Data replication in large-scale data management systems

Uras Tos

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*
Cotutelle internationale avec *Université Ege, Turquie*

Présentée et soutenue le *27/06/2017* **par :**

URAS TOS

Réplication de données dans les systèmes de gestion de données à grande échelle

JURY

ESTHER PACITTI	Professeur à l'Université Montpellier II, France	Rapporteur
ALI YAZICI	Professeur à l'Universite Atılım, Turquie	Rapporteur
PATRICK VALDURIEZ	Directeur de Recherche à l'INRIA Sophia Antipolis, Méditerranée, France	Examinateur
ABDELKADER HAMEURLAIN	Professeur à l'Université Toulouse III, France	Directeur de Thèse
TOLGA AYAV	Professeur Assistant à l'Institut des Technologies d'Izmir, Turquie	Co-Directeur de Thèse
RIAD MOKADEM	Maître de Conférences à l'Université Toulouse III, France	Encadrant

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeurs de Thèse :
Abdelkader HAMEURLAIN et *Tolga AYAV*
Rapporteurs :
Esther PACITTI et *Ali YAZICI*

to mom and dad

Acknowledgments

I am sure there are many definitions of what a PhD is and what it entails along the lines of conducting independent research and pushing the boundaries of human knowledge. However, I can only attest to my perspective, as an individual who just went through the experience. Facing an immense problem and the emotional concern that comes attached, the feelings of the PhD candidate slowly turns into content as the problem is slowly chipped away with concentration, dedication and steady effort. For me, much like an athlete who pushes the limits of physical strength with repeated exercise, a PhD is a perpetual challenge to one's intellect for advancement. The fulfillment of conquering a PhD-level problem is rare in a person's life. Being awarded an opportunity towards this goal is indubitably has the utmost importance. For that reason, I am grateful to Institut de Recherche en Informatique de Toulouse (IRIT) and its director, Prof. Michel Daydé for accepting me as a PhD student.

The contribution of a thesis increases even more with external evaluation by veteran researchers of the relevant field. Hence, I thank to the reviewers Prof. Esther Pacitti and Prof. Ali Yazıcı, and examiner Prof. Patrick Valduriez for their honest and valuable feedback on my work.

Working on a specialized problem for a long time enables a PhD candidate to gain in-depth knowledge of that field. However, it may also cause the PhD candidate

to fall into the pitfall of getting lost along the way. Fortunately, I had the pleasure of working with four brilliant people to guide me during my PhD.

Prof. Abdelkader Hameurlain, with decades of experience under his belt, always shared his vision with me and reminded me not to miss the big picture while I obsessed with the details. Not just the knowledge of my field, but I also learned a great deal about life in academia from him and for that, I am grateful.

Dr. Tolga Ayav have always been a pleasure to work alongside, with his friendly attitude. He was also my advisor during my MSc and I probably would not have pursued an academic career wihout his encouragement. I am thankful to him for everything he has done for so many years.

I also thank Dr. Sebnem Bora for her positive attitude and appreciation of my efforts. She always had a way of showing her support with uplifting emotions.

Last but not least, I thank Dr. Riad Mokadem for his close support on every detail of my thesis. Not only an advisor, but he has also been a life saver in many day-to-day needs I had during my visits to France. I will certainly miss our discussions and I am honored to consider him as a friend.

I thank my colleagues at IRIT, Izmir Institute of Technology and Ege University for providing me a friendly workplace to conduct research. Also, I am grateful to The Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing financial support during my studies. I am also offering my sincerest thanks to all my friends, especially Kamil and Mehmet for vesting their confidence in my success.

Lastly but most importantly, I am deeply grateful to my parents, Nurşen and Turan for having raised me as an individual who values science, reason and logic above all else for guiding me in life. Also my little sister Eda, who always been there for me no matter what. I owe them everything for what I am today.

Abstract

Data replication in large-scale data management systems

In recent years, growing popularity of large-scale applications, e.g. scientific experiments, Internet of things and social networking, led to generation of large volumes of data. The management of this data presents a significant challenge as the data is heterogeneous and distributed on a large scale.

In traditional systems including distributed and parallel systems, peer-to-peer systems and grid systems, meeting objectives such as achieving acceptable performance while ensuring good availability of data are major challenges for service providers, especially when the data is distributed around the world. In this context, data replication, as a well-known technique, allows: (i) increased data availability, (ii) reduced data access costs, and (iii) improved fault-tolerance. However, replicating data on all nodes is an unrealistic solution as it generates significant bandwidth consumption in addition to exhausting limited storage space. Defining good replication strategies is a solution to these problems.

The data replication strategies that have been proposed for the traditional systems mentioned above are intended to improve performance for the user. They are difficult to adapt to cloud systems. Indeed, cloud providers aim to generate a profit in addition to meeting tenant requirements. Meeting the performance expectations

of the tenants without sacrificing the provider's profit, as well as managing resource elasticities with a pay-as-you-go pricing model, are the fundamentals of cloud systems.

In this thesis, we propose a data replication strategy that satisfies the requirements of the tenant, such as performance, while guaranteeing the economic profit of the provider. Based on a cost model, we estimate the response time required to execute a distributed database query. Data replication is only considered if, for any query, the estimated response time exceeds a threshold previously set in the contract between the provider and the tenant. Then, the planned replication must also be economically beneficial to the provider. In this context, we propose an economic model that takes into account both the expenditures and the revenues of the provider during the execution of any particular database query. Once the data replication is decided to go through, a heuristic placement approach is used to find the placement for new replicas in order to reduce the access time. In addition, a dynamic adjustment of the number of replicas is adopted to allow elastic management of resources.

Proposed strategy is validated in an experimental evaluation carried out in a simulation environment. Compared with another data replication strategy proposed in the cloud systems, the analysis of the obtained results shows that the two compared strategies respond to the performance objective for the tenant. Nevertheless, a replica of data is created, with our strategy, only if this replication is profitable for the provider.

Keywords: Cloud Computing, Database Queries, Data Replication, Performance Evaluation, Economic Benefit

Résumé

Réplication de données dans les systèmes de gestion de données à grande échelle

Ces dernières années, la popularité croissante des applications, e.g. les expériences scientifiques, Internet des objets et les réseaux sociaux, a conduit à la génération de gros volumes de données. La gestion de telles données qui de plus, sont hétérogenes et distribuées à grande échelle, constitue un défi important.

Dans les systèmes traditionnels tels que les systèmes distribués et parallèles, les systèmes pair-à-pair et les systèmes de grille, répondre à des objectifs tels que l'obtention de performances acceptables tout en garantissant une bonne disponibilité de données constituent des objectifs majeurs pour l'utilisateur, en particulier lorsque ces données sont réparties à travers le monde. Dans ce contexte, la réplication de données, une technique très connue, permet notamment: (i) d'augmenter la disponibilité de données, (ii) de réduire les coûts d'accès aux données et (iii) d'assurer une meilleure tolérance aux pannes. Néanmoins, répliquer les données sur tous les nœuds est une solution non réaliste vu qu'elle génère une consommation importante de la bande passante en plus de l'espace limité de stockage. Définir des stratégies de réplication constitue la solution à apporter à ces problématiques.

Les stratégies de réplication de données qui ont été proposées pour les systèmes traditionnels cités précédemment ont pour objectif l'amélioration des performances

pour l'utilisateur. Elles sont difficiles à adapter dans les systèmes de cloud. En effet, le fournisseur de cloud a pour but de générer un profit en plus de répondre aux exigences des locataires. Satisfaire les attentes de ces locataire en matière de performances sans sacrifier le profit du fournisseur d'un coté et la gestion élastiques des ressources avec une tarification suivant le modèle 'pay-as-you-go' d'un autre coté, constituent des principes fondamentaux dans les systèmes cloud.

Dans cette thèse, nous proposons une stratégie de réplication de données pour satisfaire les exigences du locataire, e.g. les performances, tout en garantissant le profit économique du fournisseur. En se basant sur un modèle de coût, nous estimons le temps de réponse nécessaire pour l'exécution d'une requête distribuée. La réplication de données n'est envisagée que si le temps de réponse estimé dépasse un seuil fixé auparavant dans le contrat établi entre le fournisseur et le client. Ensuite, cette réplication doit être profitable du point de vue économique pour le fournisseur. Dans ce contexte, nous proposons un modèle économique prenant en compte aussi bien les dépenses et les revenus du fournisseur lors de l'exécution de cette requête. Nous proposons une heuristique pour le placement des répliques afin de réduire les temps d'accès à ces nouvelles répliques. De plus, un ajustement du nombre de répliques est adopté afin de permettre une gestion élastique des ressources.

Nous validons la stratégie proposée par une évaluation basée sur une simulation. Nous comparons les performances de notre stratégie à celles d'une autre stratégie de réplication proposée dans les clouds. L'analyse des résultats obtenus a montré que les deux stratégies comparées répondent à l'objectif de performances pour le locataire. Néanmoins, une réplique de données n'est crée, avec notre stratégie, que si cette réplication est profitable pour le fournisseur.

Mots-clés: Systèmes cloud, requêtes de base de données, réplication de données, évaluation de performances, profit économique

Contents

Chapter 1: Introduction **17**

1.1 Context 18

1.1.1 Data Replication 19

1.1.2 Cloud Computing 21

1.2 Motivations 23

1.3 Contributions 26

1.4 Publications 28

1.5 Organization of the Thesis 28

Chapter 2: State of the Art **31**

2.1 Introduction 32

2.2 Data Replication in Data Grid Systems 34

2.2.1 Existing Classifications 35

2.2.2 Proposed Classification 38

2.2.2.1 Data Grids with Multi-tier Architecture 38

2.2.2.2 Data Grids with Bandwidth Hierarchy Consideration 41

2.2.2.3 Other Hierarchical Data Grid Architectures 44

2.2.2.4 Data Grids with Peer-to-peer Architecture 47

2.2.2.5 Data Grids with Hybrid Architecture 49
2.2.2.6 Data Grids with General Graph Architecture 50
2.2.3 Analysis 52
2.3 Data Replication in Cloud Systems 54
2.3.1 Data Replication to Satisfy Objectives Other Than Performance 56
2.3.2 Data Replication for Performance Objective without Economic Consideration 60
2.3.3 Data Replication for Performance Objective with Economic Consideration 63
2.3.4 Analysis 65
2.4 Conclusion 66

Chapter 3: Proposed Data Replication Strategy 69

3.1 Introduction 70
3.2 Cloud Topology 72
3.3 Data Replication Issues 74
3.3.1 Replication Decision 75
3.3.2 Replication Degree 77
3.3.3 Replica Placement 79
3.3.4 Replica Retirement 82
3.4 A Cost Model for DB Query Processing 83
3.4.1 Response Time Estimation 84
3.4.1.1 Estimating Response Time of Database Queries . . . 84
3.4.1.2 Intra-operator Parallelism 87
3.4.1.3 Inter-operator Parallelism and Pipeline Chains . . . 88
3.4.2 Resource Consumption 91

3.4.2.1 CPU Consumption 92
3.4.2.2 I/O Consumption 93
3.4.2.3 Network Consumption 94
3.4.2.4 Total Resource Consumption 95
3.5 An Economic Model for DB Query Processing in the Cloud 96
3.5.1 Estimating Provider Profit 97
3.5.2 Estimating Provider Revenue 99
3.5.3 Estimating Provider Expenditures 100
3.5.3.1 CPU Cost 101
3.5.3.2 I/O Cost 102
3.5.3.3 Network Cost 103
3.5.3.4 Penalties 104
3.6 Conclusion 105

Chapter 4: Performance Evaluation 107

4.1 Introduction 108
4.2 Simulation Environment 109
4.2.1 Cloud Simulator 110
4.2.2 Simulated Cloud Topology 112
4.2.3 Simulated Query Load 113
4.2.4 Simulation Parameters 115
4.2.5 Comparison of Data Replication Strategies 116
4.3 Simulation Results 117
4.3.1 Measured Metrics 117
4.3.2 Average Response Time 118
4.3.3 Number of Replications 119

4.3.4 Storage Usage 120
4.3.5 Network Usage 121
4.3.6 Number of SLA Violations 122
4.3.7 Monetary Expenditures of the Provider 123
4.4 Conclusion 124

Chapter 5: Conclusion and Future Work **127**
5.1 Summary of Studies 128
5.2 Future Directions 130

Bibliography **135**

List of Figures

2.1 Multi-tier data grid architecture. 39

2.2 A hierarchical data grid architecture based on network-level locality. . 42

2.3 Peer-to-peer data grid architecture. 47

2.4 An example hybrid data grid architecture (sibling tree). 49

2.5 An example data grid architecture with scale-free topology. 51

3.1 An example cloud topology showing regions and subregions (datacenters). 73

3.2 Query plans that are considered by the response time estimation model. 86

3.3 Pipeline chains in left-deep, right-deep and bushy query plans (Garofalakis and Ioannidis, 1996). 89

3.4 Dependencies among pipeline chains in left-deep, right-deep and bushy query plans. 90

4.1 The cloud topology realized in the simulation. 113

4.2 Response time of the queries during simulation. 119

4.3 Consumption of bandwidth with respect to network hierarchy. 121

4.4 Total costs of the provider during the simulation. 123

List of Tables

1.1 Publications that resulted from the studies described in the thesis. . . 29

4.1 Simulation parameters used in the performance evaluation. 116

4.2 Simulation results. 118

Chapter 1

Introduction

Abstract

In this chapter, the context of this thesis is explained in terms of the targeted data management system and the aim of the data replication strategy. Furthermore, the motivation behind the proposed data replication strategy is described as well as the conditions that necessitate it. Publications resulted from this thesis alongside the contributions are also provided in this chapter.

Contents

1.1 Context **18**

1.1.1 Data Replication 19

1.1.2 Cloud Computing 21

1.2 Motivations **23**

1.3 Contributions **26**

1.4 Publications **28**

1.5 Organization of the Thesis **28**

1.1 Context

A noticeable aspect of our lives in the new millennium is that we are now surrounded with a plethora of interconnected services that generate tremendous amount of data. In the last decade, growing popularity of social networks, Internet of things and cloud-based software services immensely increased the amount of data flowing across the globe. This data is at such a large-scale, even storing it presents significant challenges. Processing this ever-increasing scale of data, is however a completely huge challenge in on itself (Hameurlain and Morvan, 2016). Dealing with large-scale data inevitably strains the capabilities of many traditional systems, including centralized database management systems (DBMS).

Dealing with the vastness of large-scale data, several large-scale data management systems have been introduced over the past decades. Among those, some noticeable examples are parallel and distributed systems, peer-to-peer (P2P) systems, data grid systems, and more recently cloud systems. Each of these data management systems have specific properties in their design. These properties may however, be a double edged sword in data management; while providing benefits in one aspect, they can also bring challenges in other issues. A good example would be implementing data replication in a distributed environment. It may provide a performance benefit, but also make it more difficult dealing with data update operations.

Data access performance and availability are other significant issues that must be addressed by any large-scale data management system. Users expect to have a certain level of service quality when it comes to accessing their data. Frequent data access on a global-scale data inevitably poses a significant hurdle for the service providers. Risk of overloaded computational resources and network links, finite storage availability are just a few examples that concern service providers. Service

providers are bound to satisfy a certain, acceptable quality of service in a cost-effective way to their customers.

In a scenario where the response time of a query is at an undesirably high level for the tenant, the service provider may use some replication solution, e.g. data and task replication, to improve the response time of the query. Sometimes, the executing server may not have enough available CPU, or other necessary resource to process the query with an acceptable response time. This may be a frequent occurrence in a multi-tenant data management environments. In this case, the provider might want to replicate the query to be executed on other servers where there is enough resources available for execution (Wang et al., 2014). On the other hand, there may be some cases where the bottleneck for query execution is not due to some local resource such as the CPU but due to transferring some necessary remote data residing on a different server. If available bandwidth to that particular remote server is not enough, the response time guarantee may not be satisfied due to slow data access. In this case, replicating the associated data closer to the requestor server may improve the performance problem. While it is possible that these performance issues can be solved with some form of replication, e.g. data or task replication, the proposed data replication strategy in this thesis deals data replication aspect of query execution in the cloud (Tos et al., 2016, 2017b,a).

1.1.1 Data Replication

On ensuring performance, or rather satisfying an agreed upon performance level; service providers can benefit from a plethora of choices. Among these, data replication is a very well known and researched data management technique that has been used for decades in many systems. Benefits of data replication include increased perfor-

mance by strategic placement of replicas, improved availability by having multiple copies of data sets and better fault-tolerance against possible failures of servers.

When tenant queries are submitted to the data management system, depending on the execution plan, e.g. the number of joins, they may require a number of relations in order to carry on with the execution. Naturally, in a large-scale environment where relations are fragmented and distributed geographically in multiple servers, not all required data may be present on the executing node itself. Considering that a query is processed on multiple servers according to inter-operator and intra-operator parallelism, the likelihood of some remote data to be shipped from faraway servers is a realistic possibility. In cases when the network bandwidth capability to the remote servers are not abundant, e.g. due to remote data being at a geographically separate location, a bottleneck that may ultimately lead to a response time dissatisfaction may occur during query execution process.

In order to ensure the satisfaction of query response time objective, the bottleneck data should be identified heuristically to be selected for possible replication before the query is even started executing. Also, when to trigger the actual replication event to start is another important decision that must be made for the same goal. Deciding how many replicas to create and how to retire the unused replicas must also be dealt with further down the road in the data replication decision process. Strategic placement of the newly created replicas plays a key role in reducing data access latency and improving response time satisfaction. Undoubtedly, all of these replication decisions should be made from a cost-effective point of view to ensure the economic benefit of the provider, which is especially important in the economy-based large-scale systems such as cloud computing.

Dealing with the mentioned issues of data replication, a good data replication

strategy must be able to decide in a meaningful way; (i) *what to replicate* to correctly determine which fragments of relations are in need of replication, (ii) *when to replicate* to be able to respond the change in demand of data in a timely manner to quickly resolve performance problems, (iii) *how many replicas to create* to avoid wasting precious resources such as storage to keep the costs down and retire unnecessary replicas accordingly, and finally (iv) *where to replicate* to strategically place newly created replicas to ensure tenant performance expectations are met and any possible penalties are avoided. Moreover, all of these decisions should be based on some criteria that are consistent with the aims of both the tenant and the provider.

1.1.2 Cloud Computing

In cloud computing, physical resources are abstracted and rented to a multitude of tenants. While the advantages and disadvantages of abstraction of physical resources is entirely another topic that merits its own research discussion, a well known benefit of this new way of resource provisioning is the elastic scaling of resources on demand, without interruption (Hameurlain and Mokadem, 2017). Indeed, Foster et al. (2008) define cloud computing as directly quoted below.

> *A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.*

Cloud providers share the abstracted physical resources among their tenants. In return, tenants pay the rent associated with their share of services acquired from the cloud provider, according to the pay-as-you-go pricing model. This economic

relationship between the provider and the tenants is another new aspect of cloud systems in dealing with data management.

As expected from them, tenants demand the best possible performance for their applications. However, best performance is a holy grail, which is very difficult and very costly for the provider to supply. These challenges are further amplified when best performance is continuously demanded in a dynamically changing environment such as cloud systems. It is therefore not realistically possible for a provider to offer the ultimate best performance to the tenants in a cost-effective way (Tos et al., 2016). As any other economic enterprise, cloud providers have to be in pursuit of maximizing their profits. While the intricacies of market economies are beyond the scope of this thesis, it is safe to say that in a competitive market where the cloud providers are pressured to offer their services at lower prices to attract tenants; only realistic option is to reduce the costs of these services for the provider. Unfortunately, *low cost* and *best performance* are two goals that often contradict with each other. These conflicting goals are therefore must be regulated in such a way that is acceptable for both parties in this economic relationship. Instead of best performance, providers offer their tenants a threshold performance level as a service guarantee.

The economic relationship between the provider and the tenant is clearly defined in the *Service Level Agreement* (SLA) (Stantchev and Schröpfer, 2009; Buyya et al., 2009). SLA is a legally binding contract that regulates and protects the interests of the parties involved. Among the terms of this contract, *Service Level Objective*s (SLO) are particularly interesting for this thesis. SLOs are the agreed upon set of objectives, e.g. performance, availability etc. that are included in the SLA to specifically define the quality of the service the provider must supply to the tenants.

Breach of the SLA terms often result in some consequence for the provider, including monetary compensation as a penalty (Xiong et al., 2011).

Many applications designed for the cloud environment handle data differently. Some would prefer storing fast flowing data in an unstructured way, while others may require a relational model to take advantage of a structured data store that is subject to a schema. Therefore, data management in cloud systems is not a single, fit-for-all solution. In that sense, this thesis focuses on a specific data management context, namely database management systems in the cloud systems. In the considered cloud environment, database relations are fragmented and distributed to many servers around the globe. These servers are contained in datacenters that are located in various geographical regions. Each server, datacenter and region are interlinked with network connections that vary in both bandwidth capacity and cost. In this heterogeneous cloud environment, cloud providers process database queries from multiple tenants, to satisfy a threshold performance level and return a profit at the same time.

1.2 Motivations

A good way to understand both data replication and new challenges that are brought to the table by data management in the cloud is to study how data replication is implemented on traditional systems that precede cloud computing. Traditional systems such as data grids precede cloud systems in historical context, therefore many data replication strategies that are proposed for the cloud is somewhat adaptations of their counterparts for data grid systems in order to address some specific challenge of the cloud. Consequently, it allows us to better understand how the challenges of

data replication change in conjunction with the challenges of the cloud and how the proposed strategies adapted their implementation to address these challenges.

A data management system in the cloud can be elastically scaled on demand, without interruption. A sensible approach for data replication should take advantage of this property of the cloud. A common way of replicating data in the traditional systems, i.e. data grid, is creating as many replicas as possible to attain maximum resource utilization to provide best performance. In cloud systems, such a data replication strategy may not be economically beneficial for the provider, since creation of a large number of replicas can result in a wasteful resource utilization and in return, reduced profit. As mentioned before, in cloud systems, three major questions of what to replicate, when to replicate, and where to replicate (Ranganathan and Foster, 2001) must be answered in such a way to satisfy the performance in an economically feasible way (Tos et al., 2016).

There is a number of efforts in the literature that studied data replication in the cloud systems. Many of them focus just on satisfying the availability SLO (Silvestre et al., 2012; Sun et al., 2012a). In a typical cloud environment, where frequent queries are placed on a large-scale data, having low response time is crucial for the tenants. However, performance guarantees, e.g. response time, are often not offered by cloud providers as a part of the SLA. In order to resolve this issue, there are several works proposed (Kouki et al., 2011; Sakr and Liu, 2012) in the literature to include the response time guarantees in the SLA. Dealing with data replication, only a few studies are particularly interested in improved response time (Wei et al., 2010; Bai et al., 2013; Janpet and Wen, 2013; Zhang et al., 2014). In addition, even fewer of those studies (Bonvin et al., 2010a; Ghanbari et al., 2012) are taking economics of the cloud into account.

Considering the state of the art on data replication in cloud systems, as mentioned, a vast majority of the existing studies focus on criteria other than performance, e.g. availability. Among the minority of those that deal with performance, an often noticed pattern is that the performance is not considered as an objective but a measured result of data replication as a consequence of having replicas in the system. A very few number of data replication strategies for cloud systems are actually focused on satisfying a performance objective. Furthermore, among those minority, the examples that target a relational database management system that is operating in the cloud is minuscule. Therefore, only a significantly small number of existing studies are valuable for the specific set of problems that are mentioned throughout this thesis study.

This thesis is therefore built on the motivation to address the mentioned problems and shortcomings of the existing data replication strategies in cloud systems that deal with database queries. More specifically, the proposed data replication strategy in this thesis focuses on a novel solution to satisfy performance guarantees to the tenants, as well as ensuring profitability of the provider while executing queries in a relational database system situated in the cloud.

In this thesis, a strategy for *Achieving query Performance in the cloud via a cost-Effective data Replication* (APER) is proposed to deal with database queries for OLAP applications. APER focuses on the simultaneous satisfaction of both the response time objective and provider profit. We consider left-deep, right-deep and bushy query plans. In a given query plan, before the execution, APER identifies the pipeline chains that are responsible for a response time dissatisfaction. APER estimates the response time of each operator in pipeline chains, taking into account both the inter-operator and intra-operator parallelism. If a required fragment of

a relation is predicted to cause a bottleneck during the execution, that particular fragment is considered for replication. The estimated response time with the inclusion of this particular replication must be less than a certain SLO response time threshold. A placement is found for the new replica through a heuristic that reduces both resource consumption and monetary cost. Furthermore, the number of replicas is dynamically adjusted over time. Carrying out the replication also depends on another estimation, namely the provider profit estimation. We estimate both the revenues and expenditures of the provider when executing a query in a multi-tenant context. If the execution of the query is estimated to be still profitable for the provider with any possible new replicas, only then the data replication is performed. This constitutes a challenge that consists of maximizing the provider profit while minimizing the expenditures as much as possible.

1.3 Contributions

A number of contributions have been made in the duration of this thesis study. These contributions can be summarized as follows.

(i) *A complete data replication strategy that satisfies performance SLO and profitability of the provider simultaneously.* The simultaneous satisfaction of two key criteria, namely response time satisfaction and profitability of provider are pursued for each query execution. When a query is submitted for execution, proposed data replication strategy identifies whether data replication is necessary and takes the corresponding action that will result in the desired effect in terms of performance and profit. If a data replication event is necessary to take place, how many replicas to create is also another issue dealt by the data

replication strategy. Where to place the replicas are determined strategically to satisfy the SLA with most amount of profit.

(ii) *A cost model for estimating response time of executing database queries in cloud computing context.* When tenants submit queries to the cloud, they expect a timely response time that is in accord with the SLA. The cost model therefore, estimates whether a submitted query can be processed with an acceptable response time that satisfies the SLA. If the query is estimated to violate the response time objective, proposed cost model determines the reason by identifying possible data access bottlenecks in the query plan. Bottleneck data is then considered for replication. Response time estimation of database queries has already been a well studied topic in the literature (Lanzelotte et al., 1994; Özsu and Valduriez, 2011). As a result, the proposed cost model takes into account the pipelining, inter-operator and intra-operator parallelism and resource consumption of queries by standing on the shoulders of the existing studies.

(iii) *An economic model of database query execution in the cloud that takes data replication into account.* This economic model estimates the profitability of the provider by estimating the monetary cost of executing each and every query. Estimated cost of execution is compared with the estimated revenue per query in order to predict the profit generated by the provider for any particular query execution. Naturally, the provider aims to return some profit while satisfying tenant requirements, therefore the proposed profit estimation is employed as a decision criterion in the proposed data replication strategy.

(iv) *A detailed performance evaluation study to validate the proposed data replica-*

tion strategy. In a simulation environment, the proposed strategy is compared to another data replication approach. The experiments highlight the difference between the two strategies in a simulation scenario where frequent queries are processed in the cloud. Part of this contribution also includes the modifications necessary to CloudSim (Calheiros et al., 2011) simulation tool, which does not support data replication out-of-the-box. Extending CloudSim is necessary to accurately simulate a multi-tenant cloud environment that processes database queries with data replication.

1.4 Publications

During the research period, the studies described in this thesis manuscript have resulted in preparation of a few publications that demonstrate the contributions. These publications are shown in Table 1.1.

1.5 Organization of the Thesis

While the publications resulted from this thesis study already describe many of our contributions, a coherent narrative is crucial to convey the details of the proposed strategy in a more readable way. As a result, this thesis manuscript is organized as follows.

Chapter 2 discusses the state of the art on some important concepts that concern this thesis. A detailed discussion of existing data replication strategies is provided with respect to various data management systems. This discussion helps to convey the justification of target data management system that is considered in this thesis, namely database management systems operating in the cloud.

Table 1.1: Publications that resulted from the studies described in the thesis.

№	Publication	Status	Notes
1	Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Dynamic replication strategies in data grid systems: a survey. *The Journal of Supercomputing*, 71(11):4116–4140, 2015. ISSN 0920-8542. doi: 10.1007/s11227-015-1508-7	Published	
2	Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. A performance and profit oriented data replication strategy for cloud systems. In *Intl IEEE Conference on Cloud and Big Data Computing (CBDCom)*, pages 780–787. IEEE, jul 2016. ISBN 978-1-5090-2771-2. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0125	Published	Received best paper award
3	Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Ensuring performance and provider profit through data replication in cloud systems. *Cluster Computing*, (under review), 2017a	Under review	
4	Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Achieving query performance in the cloud via a cost-effective data replication strategy. *International Journal of Web and Grid Services*, (under review), 2017b	Under review	

The main contribution of the thesis is described in Chapter 3. How the proposed data replication strategy decides which database relations to replicate, when to trigger the replication event, how many replicas to create during each replication and where to place the newly created replicas with respect to the cost model and the economic model are discussed here. Also, retirement of unnecessary replicas are also described in this chapter. Furthermore, this chapter introduces a cost model and an economic model of database query processing in the cloud. The cost model of query processing focuses especially on the response time estimation of the database

queries, with respect to inter-operator and intra-operator parallelism. Additionally, how the proposed response time estimation takes into account the consumption of CPU, I/O and network resources by each operator in the query execution plan is discussed. The economic model deals with estimating the monetary cost of each query execution. Moreover, the economic impact of the replication decisions is also dealt by this model. How the provider profit is estimated with respect to monetary resource cost of each query execution is also covered here.

Performance evaluation study of the proposed data replication strategy is presented in Chapter 4. In a simulation environment, the proposed strategy is pit against another strategy in a simulation scenario to demonstrate how the proposed data replication strategy satisfies performance guarantees for the tenant and profitability of the provider simultaneously.

Chapter 5 concludes the thesis manuscript with an overall discussion of some advantages and shortcomings of the proposed data replication strategy, the difficulties encountered during the thesis studies and overall concluding remarks on doing research in this research area. Furthermore, a discussion of some possible future directions in this research area is also provided in this chapter.

Chapter 2

State of the Art

Abstract

In this chapter, a detailed state of the art on data replication in various data management systems is provided. The existing data replication strategies in several large-scale data management systems are analyzed with respect to their key properties and how they take advantage of the data management system they are targeting.

Contents

2.1 Introduction **32**

2.2 Data Replication in Data Grid Systems **34**

2.2.1 Existing Classifications 35

2.2.2 Proposed Classification 38

2.2.3 Analysis 52

2.3 Data Replication in Cloud Systems **54**

2.3.1 Data Replication to Satisfy Objectives Other Than Performance 56

2.3.2 Data Replication for Performance Objective without Economic Consideration 60
2.3.3 Data Replication for Performance Objective with Economic Consideration 63
2.3.4 Analysis 65
2.4 Conclusion **66**

2.1 Introduction

Data replication is a very well-known data management technique that has been commonly adopted by many traditional systems, including (i) database management systems (DBMS) (Kemme et al., 2010), (ii) parallel and distributed systems (Özsu and Valduriez, 2011), (iii) mobile systems (Guerrero-Contreras et al., 2015) and (iv) other large-scale systems including P2P (Spaho et al., 2015) and data grid systems (Tos et al., 2015).

In the large-scale data management systems where the data is distributed geographically, i.e. at the scale of wide area networks (Goel and Buyya, 2006). Frequent access to data would strain network links, overload remote data stores, and overall degrade computational performance. On the other hand, placing local copies of data sets at each node is costly and simply not realistic. As a result, placement of data plays an important role in all large-scale data management systems.

Dealing with data placement problem, data replication is interested in strategically placing copies of data in order to increase availability, access performance, reliability, and fault-tolerance, as well as to reduce bandwidth usage, and job completion times. Many replication strategies have been proposed (Ranganathan and

Foster, 2001; Tang et al., 2005; Park et al., 2004; Chang and Chang, 2008) to achieve such goals.

Every data replication strategy should be able to address several problems (Ranganathan and Foster, 2001). (i) *What data should be replicated?* It is generally not feasible to replicate each data set, therefore establishing a meaningful criteria on choosing what to replicate is important. (ii) *When should the replication take place?* Establishing a good trade-off on when to replicate is crucial as replicating too early might be wasteful on resources, while replicating too late may not yield the full benefits of replication. (iii) *How many replicas should be created?* An optimal, or at least a near-optimal number of replicas should ideally be present in the system to balance the benefits and costs of the replication. While having too many replicas may be wasteful, having too little can be equally undesirable as it may not be enough to satisfy a desired service quality. (iv) *Where should the replicas be placed?* Generally, placing replicas closer to the clients with the most access requests may improve overall performance during frequent data accesses.

Achieving an optimal replica configuration across an entire data management system is an NP-hard problem (Tang and Xu, 2005; Du et al., 2011). Albeit not providing exactly an optimal solution, existing data replication strategies very widely use heuristics to achieve a desired replica configuration throughout the target large-scale environments with at least a near-optimal solution. These data replication strategies set some trade-offs and generally aim to excel in at least one aspect of service quality with minimal undesirable consequence.

In the following subsections, we study the existing data replication strategies in some large-scale data management systems that are relevant to the work put forward in this thesis. Even though the main contribution of this thesis is in the

cloud systems context, the state of the art on data replication also contains other data management systems in order to better understand how data replication strategies are evolved with respect to the properties of each individual data management environment.

2.2 Data Replication in Data Grid Systems

The notion of *grid computing* emerged in 1990s as a way to establish a distributed processing infrastructure to satisfy information handling needs of institutions that perform advanced science experiments (Foster, 2001). These institutions contribute computational resources to the grid as virtual organizations, which in turn share these resources in a federated manner.

Data grid is a specialized grid infrastructure that provides a scalable data management solution for science experiments that generate a large amount of data (Chervenak et al., 2000). It provides a globally-spanned heterogeneous environment to research institutions. Data grid is independent of the research areas of these institutions as the many diverse fields take advantage of the data grid including physics (Segal, 2000; Takefusa et al., 2003; Hoschek et al., 2000), astronomy (Deelman et al., 2002), biology (Maltsev et al., 2006), and climate science (Chervenak et al., 2003). A common emergence among these experiments is, all of them generate immense amounts of data that is often in the petabytes region (Hoschek et al., 2000; Tatebe et al., 2002).

Often times, data sets are generated by some experiment performed at an institution and the resulting data sets are processed by other interested institutions which are located in various parts of the world. Processing these data sets require

data to be shipped from their origin to the requestors. Frequent data access, combined with precious and limited bandwidth availability potentially cause disastrous data transfer times. Therefore, data replication is an inevitable necessity in data grid systems.

2.2.1 Existing Classifications

There has been a vast number of efforts in the literature on data replication in data grid systems. Each data replication strategy focus on achieving a clearly defined objective, e.g. availability and performance. In this sense, it is possible to classify these replication strategies with respect to some clearly defined criteria existing in their design. Some of these classifications are based on static vs. dynamic classification (Chervenak et al., 2002; Cibej et al., 2005), while some others deal with centralized vs. decentralized replication strategies (Ma et al., 2013; Dogan, 2009; Amjad et al., 2012). Push-based vs. pull-based classification (Nicholson et al., 2008; Chervenak et al., 2008; Dogan, 2009; Steen and Pierre, 2010) also exists in the literature, as well as a classification based on the objective function (Mokadem and Hameurlain, 2015).

Arguably the most common classification scheme is static vs. dynamic replication. In static replication, all replication decisions are made before the system is operational and replica configuration is not changed during operation (Chervenak et al., 2002; Cibej et al., 2005; Loukopoulos and Ahmad, 2004; Fu et al., 2013). On the other hand, in dynamic replication, what, when, and where to replicate are decided as a response to the changing trends of data grid (Park et al., 2004; Tang et al., 2005; Chang and Chang, 2008; Nicholson et al., 2008). In a non-changing grid environment, where nodes do not join or leave the grid and file access patterns

are relatively constant, static replication might be the better alternative. Compared to dynamic replication, static replication do not have the overhead caused by replication decisions and management. On the other hand, when replicas need to be periodically reconfigured according to changing access patterns, it causes significant administrative overhead and affects scalability and optimal resource use of the system. In a dynamic environment where grid nodes are free to join or leave, and access patterns change over time, dynamic replication excels static replication.

Centralized vs. decentralized replication depends on what entity will control the replication decision process (Ma et al., 2013; Dogan, 2009; Amjad et al., 2012). Centralized replication strategies contain a central authority to control all aspects of data replication. All decision metrics are either collected by or propagated to this central authority. Replication decisions are made by this point of control and all the other nodes report to it. In contrast, decentralized approach requires no central control mechanism to exist in the system. Nodes themselves decide on how replication will occur. Each approach has its advantages and drawbacks. Centralized replication is easier to implement and generally more efficient, as a single entity is responsible for all the decisions and has knowledge about every aspect of the data grid. On the other hand, the central authority is also a point of failure, thus is not ideal for reliability and fault-tolerance. Decentralized replication is good for reliability as there is no single point of failure in the system and the system can still behave predictably even a number of nodes are lost. However, having no central control and nodes acting on incomplete information about the state of the system may yield non-optimal results, e.g. excessive replication (Ranganathan et al., 2002).

On replication of any particular data, there are two actors involved. Former is the server that hosts the data, and the latter is the requestor that pulls the data to

its local storage. Push vs. pull based classification is focusing on the fact that which of these two actors trigger the replication event (Nicholson et al., 2008; Chervenak et al., 2008; Dogan, 2009; Steen and Pierre, 2010). In push based replication, replication event is triggered by the originator of data, as the server pushes data sets to clients. Servers receive requests from a number of clients, thus they require enough information about the state of the system to be able to trigger replication. Therefore, push based replication is often proactive. In pull based replication, replication event is triggered by the clients. Pull based replication can be regarded as reactive, since replication is realized on-demand. Client-side caching is also regarded as pull replication due to the fact that in this form of caching, clients decide to temporarily store data in their local storage (Steen and Pierre, 2010).

Considering the fact that data replication aims to minimize or maximize some objective, it is possible to make a classification with regard to the definition of this objective function (Mokadem and Hameurlain, 2015). One popular approach is to improve data locality (Ranganathan and Foster, 2001; Tang et al., 2005). In this case, the aim is to place replicas as close to the clients as possible. Some strategies take this aim further by heuristically identifying popular data sets and increase their locality (Shorfuzzaman et al., 2009). Cost based objective functions enable replication decisions to take a number of parameters into account (Rahman et al., 2005; Andronikou et al., 2012; Mansouri and Dastghaibyfard, 2013). In these works, replication decision is generally made according to the output of a mathematical model that take into account collective file access statistics, bandwidth availability, replica sizes, etc. Market-like mechanisms also exist in some works (Bell et al., 2003; Goel and Buyya, 2006). In these efforts, data is regarded as tradable goods. During a replication event, clients tend to buy data from remote servers that offer the lowest

price while remote servers try to sell their data to return some profit.

2.2.2 Proposed Classification

In the following subsections, we study the state of the art on data replication in data grid systems with respect to a classification based on data grid architecture This novel classification is the main contribution of one the publications (Tos et al., 2015) resulted from this thesis study.

2.2.2.1 Data Grids with Multi-tier Architecture

Multi-tier architectures follow the data grid model of GriPhyN project (Ranganathan and Foster, 2001). It is hierarchical in nature, and it has a well-defined, strict topology. On the other hand, due to this strict organizational structure, multi-tier architectures are not very flexible to allow arbitrary addition of removal of nodes. Multi-tier data grid is organized in four tiers. Tier 0 denotes the source, e.g. CERN, where the data is generated and master copies are stored. Tier 1 represents national centers, Tier 2 shows the regional centers, Tier 3 consists of work groups, and Tier 4, contains desktop computers as depicted in Figure 2.1. In this model, generally, the storage capacity increases from bottom to the upper levels of the hierarchy.

Taking advantage of the hierarchical architecture, Ranganathan and Foster (2001) paved the way by proposing six dynamic replication strategies for multi-tier data grid. These strategies are, *No Replication or Caching*, *Best Client*, *Cascading Replication*, *Plain Caching*, *Caching plus Cascading Replication*, and *Fast Spread*. No Replication or Caching is implemented as a base case for comparing other strategies to a no-replication scenario. In Best Client strategy, access history records are kept for each file on the grid. When a certain threshold is reached, that file is replicated

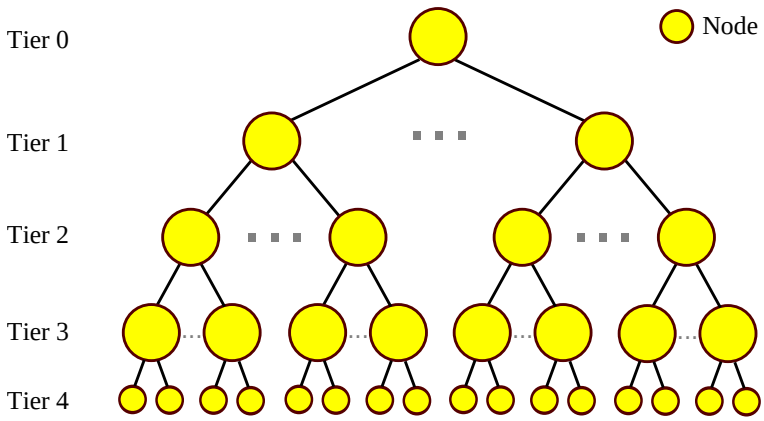


Figure 2.1: Multi-tier data grid architecture.

only on the client that generates most requests. Cascading Replication introduces a tiered replication strategy, in which, when a threshold for a file is exceeded at the root node, a replica is placed at the level on the path towards the best client, progressively. In Plain Caching, the client requests a file and stores it locally. Caching plus Cascading Replication combines Cascading Replication and Plain Caching. Fast Spread is the final strategy in which, upon client file requests, a replica of the file is placed on each tier on the path to the client. Popularity and file age are used as parameters to select files for the replica replacement approach. In simulations with three different access patterns, they show that Best Client strategy performs worst. Fast Spread works better with random data access patterns and Cascading Replication performs better when locality exists in data access patterns.

Some other data replication strategies also deal with data popularity according to the access histories of files (Cui et al., 2015). *Simple Bottom-Up* (SBU) and *Aggregate Bottom-Up* (ABU) strategies by Tang et al. (2005) are good examples of this idea. These strategies identify what files to replicate by analyzing file access history. When an access threshold is exceeded, SBU places replicas close to the nodes that request files with higher frequencies. ABU, on the other hand, calculates

the aggregate access records for each sibling of a node and passes this information to higher tiers until the root node is reached. At each level, replication decision is made when aggregate access values pass a predefined threshold. Both strategies employ *Least Recently Used* (LRU) (Arlitt et al., 2000) replica replacement approach. In the performance evaluation, ABU yields the best average response time and average bandwidth cost among studied strategies.

Also dealing with access popularity of files, Shorfuzzaman et al. (2009) propose two dynamic replication strategies for multi-tier data grid, *Popularity Based Replica Placement* (PBRP), and its adaptive counterpart, *Adaptive-PBRP* (APBRP). PBRP aims to balance storage utilization and access latency trade-off by replicating files based on file popularity. The replication strategy is run periodically in a way that access records are aggregated bottom-up and replica placement is done in a top-down manner. APBRP improves PBRP by introducing an adaptive access rate threshold. In simulations, APBRP shows improvement over PBRP while both strategies perform better than Best Client, Cascading, Fast Spread, and ABU in terms of job execution time, average bandwidth use, and storage use.

In addition to file popularity based on access frequencies, some data replication strategies also deal with spatial locality among files. A good example is *File Reunion* (FIRE) by Abdurrab and Xie (2010) strategy. FIRE assumes that there is a strong correlation between a group of jobs and a set of files. Based on this assumption, FIRE aims to reunite the file set onto the servers by means of replication. Replication is performed when a file is not locally available, and there is enough storage space to store it. If there is not enough storage space, a file with a lower group correlation degree is removed before replicating the new file. In a simulation scenario, FIRE performed better than *Least Frequently Used* (LFU) (Arlitt et al., 2000) and LRU

replication strategies.

On improving spatial locality of files, Khanli et al. (2011) proposed *Predictive Hierarchical Fast Spread* (PHFS) as an improvement over Fast Spread by Ranganathan and Foster (2001). PHFS works in three stages. In monitoring stage, file access records from all clients are collected in a log file. In analyzing stage, data mining techniques are used to discover the relationships between files. For a file A, any file B with a relationship greater than a threshold is considered in the *predictive working set* (PWS) of A. In the final stage replication configuration is applied according to the calculated PWSs. They left performance evaluation for a future study but showed on an example that PHFS improved access latency over Fast Spread.

2.2.2.2 Data Grids with Bandwidth Hierarchy Consideration

Data grids usually comprises participation from a number of institutions. These institutions although not necessarily, but usually located in various parts of the world. This geographical diversity is reflected on how a data grid architecture is substantiated in terms of network links (Figure 2.2). At the local network level, the institutions may enjoy the benefits of high-speed network links. However, over large geographical distances the Internet infrastructure is generally used. The problem with transferring large-scale data over Internet is potentially high delays and transfer times due to the lack of abundance in bandwidth. Some data replication strategies take advantage of this heterogeneity in network bandwidth capability.

One of the earliest studies to consider bandwidth hierarchy is the *Bandwidth Hierarchy Replication* (BHR) by Park et al. (2004). In their approach, they present that bandwidth between regions, e.g. countries, are narrower compared to bandwidth

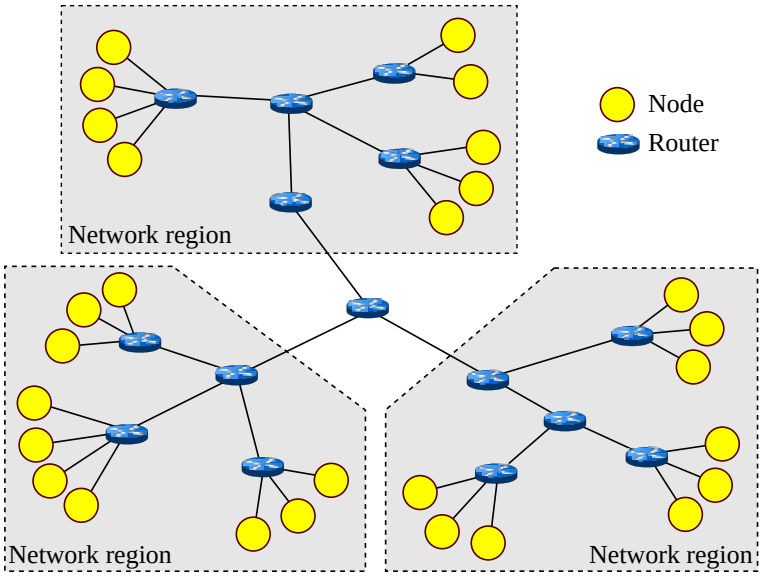


Figure 2.2: A hierarchical data grid architecture based on network-level locality.

available inside a region. BHR replicates popular files as many times as possible within a region, where intra-region bandwidth is abundant. In simulations, BHR performs better than delete LRU and delete oldest replication when narrow inter-region bandwidth or low node storage space exists. However, as the inter-region bandwidth or available storage space of the nodes increase, BHR performs similarly to the traditional strategies.

An improved implementation of BHR was proposed by Sashi and Thanamani (2011) as *Modified BHR* algorithm. In their strategy, the data is generated at the master site and replicated to region headers before any jobs are scheduled on the grid. They assume that the files accessed by a node, will also be accessed by nearby nodes and popular files will be accessed more frequently. Replicas are only placed in the region header and the node that makes the most requests. The access records are kept in the region header and least frequently used replicas are chosen as the deletion strategy. Modified BHR algorithm is compared with no replication, LFU,

LRU and BHR in a simulation study and the results show improved mean job time than to other strategies.

While BHR and its variants dealt with a two level bandwidth hierarchy, some other strategies considered a three level network hierarchy to better represent a realistic network topology. Horri et al. (2008) presented *3-Level Hierarchical Algorithm* (3LHA) for this purpose. In this work, the first level consists of regions, i.e. having low bandwidth availability. Levels two and three represent local area networks (LAN) and clients in the LANs, respectively. When a client accesses a file, if it has enough storage, the file is replicated. However, if files needed to be deleted before the replication, first, the local files that also already exist on the LAN are chosen for deletion. Then, the local files that already exist in the region are considered for deletion, and if there is still not enough space available, other local files are deleted. They compared their strategy with BHR and LRU and showed that the proposed strategy performs better in terms of mean job time.

Mansouri and Dastghaibyfard (2012) extended 3LHA and proposed *Dynamic Hierarchical Replication* (DHR) strategy. They emphasize that 3LHA places replicas in all of the requestor sites. On the other hand, DHR creates a per-region ordered list of sites with respect to the number of accesses to a file. The site that is at the top of the order is chosen to place the new replica. By placing replicas at best sites, DHR aims to lower storage cost and mean job execution time. They compare the effectiveness of DHR against no replication, LFU, LRU, BHR, and 3LHA. The results show that DHR shows better job execution times compared to other studied strategies, especially when grid sites have smaller storage space. In another paper, Mansouri and Dastghaibyfard (2013) also added economic cost model calculation to DHR, and presented *Enhanced Dynamic Hierarchical Replication* (EDHR). By

predicting future economic value of files, they made better assessment of which replicas will not be beneficial and get deleted, and which files will be beneficial and get replicated. Simulations indicate that EDHR yields even better mean job times than DHR. Another variant of 3LHA is the *Modified Dynamic Hierarchical Replication Algorithm* (MDHRA), which is another extension of DHR strategy by Mansouri et al. (2013). In MDHRA, replica replacement decision mechanism is altered to take last request time, number of accesses, and size of the replica into account. They note that the new approach improve the availability of valuable replicas. Simulations show that, compared to DHR and other studied strategies, MDHRA performs better in terms of mean job completion time and effective network usage. However, performance evaluation does not include EDHR.

Instead of periodically reacting the changing trends in the system, *Pre-fetching and Prediction based Replication Algorithm* (PPRA) (Beigrezaei et al., 2016). PPRA uses predictive statistical methods to discover the relationships between files. Therefore, when a file is requested by a client, PPRA pre-fetches any related files and replicates them if necessary. Replica managers continuously collect and update file access logs to extract the patterns in file requests.

2.2.2.3 Other Hierarchical Data Grid Architectures

Data replication strategies of this type are still designed for a hierarchical data grid architecture. However, they do not strictly target the four level multi-tier data grid (Ranganathan and Foster, 2001) or they do not necessarily consider the bandwidth hierarchy. Still, they assume the data is produced at the origin of the hierarchy and processed by clients at the lower levels. In most cases, there are middle level intermediate nodes in the hierarchy that are used for storing data.

Improving data locality through replicating popular files is a common focus for many studies. An access-weight based dynamic replication strategy is proposed by Chang and Chang (2008). Their work, *Latest Access Largest Weight* (LALW), defines a strategy for measuring popularity of files on the grid, calculating the required number of replicas, and determining sites for replica placement. Recently accessed files have larger weights and, the replica placement is based on weighted access frequencies. LALW shows similar total job execution times compared to LFU while consuming less storage space and having more effective bandwidth usage.

Another strategy called *Popular File Replicate First* (PFRF) by Lee et al. (2012) employs a threshold-based popularity measure to replicate the top 20% popular files to every grid site. Files are replicated to destination sites from the closest site that holds the required files. In a simulation scenario using five access patterns, PFRF shows improved performance on average job turnaround time, average data availability, and bandwidth cost ratio metrics.

Dynamic Optimal Replication Strategy (DORS) by Zhao et al. (2010) is yet another popularity-based replication strategy. A file is replicated when the number of replicas of that particular file is less than a dynamic threshold. Replicas are valued according to their values, which depend on access frequency and access cost of the replicas. DORS performs better than LFU and LRU in terms of mean job execution time and effective network use metrics.

Other works focus on correlation between files or file fragments. This way, it is possible to predict, to some degree, which files are going to be accessed after certain files. Saadat and Rahmani (2012) propose *Pre-fetching Based Dynamic Data Replication Algorithm* (PDDRA) with the assumption that members of a virtual organization (VO) have similar interests in files. PDDRA predicts the future accesses

of files and pre-replicates those. When a file is requested, PDDRA scans the logs and determines which files follow that file, and which of the follower files has the greatest number of accesses. PDDRA shows better performance than other strategies in terms of mean job execution time and effective network usage under all simulated access patterns.

Combination of the popularity concept with correlation among file accesses is also a visited issue in the literature. *Replication Strategy based on Correlated Patterns* (RSCP) (Hamrouni et al., 2015) and *Replication Strategy based on Maximal Frequent Correlated Patterns mining* (RSMFCP) (Qin et al., 2017) use data mining techniques to discover closely related files. Once these correlated file groups are identified, their replication are always carried out in a group. If there is not enough storage on the target node, older replicas are retired in order to make space.

In a similar approach but this time for a different goal, *Branch Replication Scheme* (BRS) (Pérez et al., 2010) takes advantage of the relationship between disjoint sub-replicas of a single file, whose replicas are placed on different nodes. With this approach, however, BRS aims to create high levels of fault-tolerance without increasing the storage use.

Improved data locality of popular files is not the only aim of replication, as evidenced by Meroufel and Belalem (2013). They propose a replication strategy called *Placement Dynamic* (PD) that determines a minimal the number of replicas to ensure a certain level of availability without degrading performance. In PD strategy, placement of replicas and failures in the system are taken into account. If a failure suspicion is observed, data is moved to other nodes in the system to maintain availability. Authors compared PD to a random replication approach in simulations performed with FTSim. Results show that PD demonstrates better

recovery times and satisfies availability compared to random replication.

2.2.2.4 Data Grids with Peer-to-peer Architecture

In peer-to-peer (P2P) data grid architectures, there is no central authority to make and enforce replication decisions. Grid nodes themselves collect the measurements of metrics used in these decisions. They act in an autonomous way and normally possess enough functionality to act as both servers and clients at the same time. This decentralized grid architecture shown in Figure 2.3 allows high volatility, as nodes can connect to any part of the grid and leave without notice.

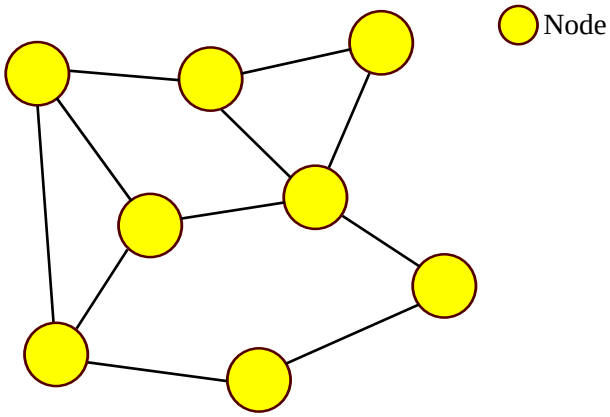


Figure 2.3: Peer-to-peer data grid architecture.

In an early effort by Ranganathan et al. (2002), a model-driven dynamic replication strategy keeps a minimum required number of replicas in P2P data grid to satisfy a desired availability level. Their model takes node stability, data transfer time between nodes, storage cost of files into account. While their system expectedly outperforms a static replication approach they highlight the fact that independent acting of nodes on incomplete information sometimes lead to unnecessary replication.

A common approach in the P2P grid where there is no apparent hierarchy among

the nodes is to rely on network link connectivity degree to decide on the minimization of data transfer times. Chettaoui and Charrada (2014) capitalize on this idea with DPRSKP, *Decentralized Periodic Replication Strategy based on Knapsack Problem*. DPRSKP selects what files to replicate by creating a prioritized list of the popularity and availability of each file. Replicas of popular files are then placed on nodes that are stable and having good bandwidth to the requestor nodes.

Also dealing with network distances when making replication decisions, Abdullah et al. (2008) propose two dynamic replication strategies, *Path and Requestor Node Placement Strategy*, and *N-hop Distance Node Placement Strategy*. Path and Requestor Node Placement Strategy replicates files on all nodes on the path to the requestor node, including the requestor itself. In N-hop Distance Node Placement Strategy, replicas are placed on all neighbors of the provider node with a distance of N. These two strategies increase availability and decrease response time at the expense of using more bandwidth.

Another data replication strategy by Challal and Bouabana-Tebibel (2010) maximizes the distance between identical replicas and minimizes the distance between non-identical replicas. They increase availability and ensure that each node has replicas of a different file in its vicinity. Another interesting aspect of their strategy is they supply the data grid with a strategic initial placement of replicas before the jobs are started.

Focusing on network topology is not the only consideration of the replication strategies for P2P data grids. Bell et al. (2003) present an economy-based data replication strategy that considers the data grid as a marketplace. Files represent tradable goods on this market. Computing elements purchase files and aim to minimize their purchasing cost. Similarly, storage elements try to maximize their profits

and make investments based on file access predictions to increase revenue. This strategy reduces total job execution times in sequential file access, however LRU performs better in some specific access patterns.

2.2.2.5 Data Grids with Hybrid Architecture

Hybrid data grid architectures generally combine at least two other architectures with different properties. For example, a replication strategy can be aimed at a sibling tree hybrid architecture, which combines P2P-like inter-sibling communication with hierarchical parenthood relationships as depicted in Figure 2.4. This is a very specific type of data grid architecture that is not considered by many data replication strategies.

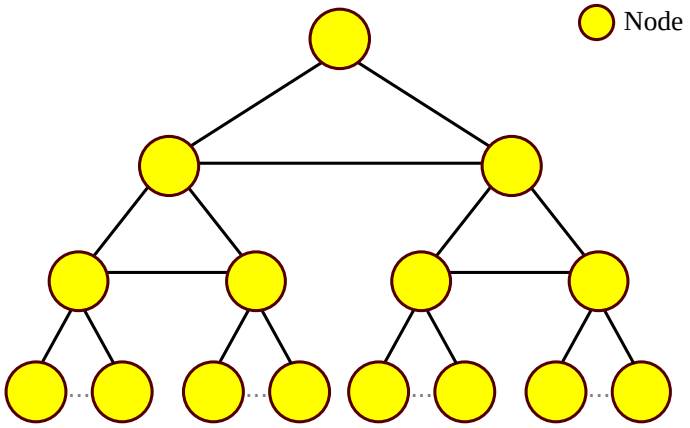


Figure 2.4: An example hybrid data grid architecture (sibling tree).

An example data replication strategy for this category is presented by Lamehamedi et al. (2002), a hybrid replication strategy that combines the hierarchical architecture with P2P features. They implemented a cost model and based the replication decisions on how the gains of the replication measure against the costs. A runtime component constantly monitors the grid to collect important parameters, i.e. replica size, and network status. These information used in the calculation of

the replication costs. They noted that average response time is improved as replicas are placed closer to clients.

Two-Way Replication (TWR) by Rasool et al. (2009) combines a multi-tier architecture with P2P-like features. In the target architecture, each node (except at the leaf level) is connected to its siblings as well being connected to its parent. Replication decision is handled by a central authority called *Grid Replication Scheduler* (GRS). GRS targets the files that have higher-than-average access frequency and replicates them at the parent of the client that generate the most requests. In terms of response time, TWR performs similarly to Fast Spread while consuming less resources.

2.2.2.6 Data Grids with General Graph Architecture

In general graphs, nodes are freely connected (Figure 2.5) without a particular topological enforcement for hierarchy. From a scalability point of view, these architectures are at an advantage because there is no strict limitation on the organization of the nodes. Data replication strategies that are targeting the scale-free, social network based data grid architectures and other general strategies that do not focus on one particular architecture are classified in this subsection.

In general graph data grids, nodes are interconnected with varying degrees of connectivity. This impacts the inter-node data transfer capabilities of the nodes as some nodes having easier access to different parts of the grid. *Dynamic Multi-replicas Creation Algorithm* (DMRC) (Chen et al., 2010) takes advantage of this scale-free grid architecture. DMRC measures the degree of distribution of nodes and place replicas on nodes with higher degrees. It also uses a cost model to calculate costs of placing replicas on candidate nodes.

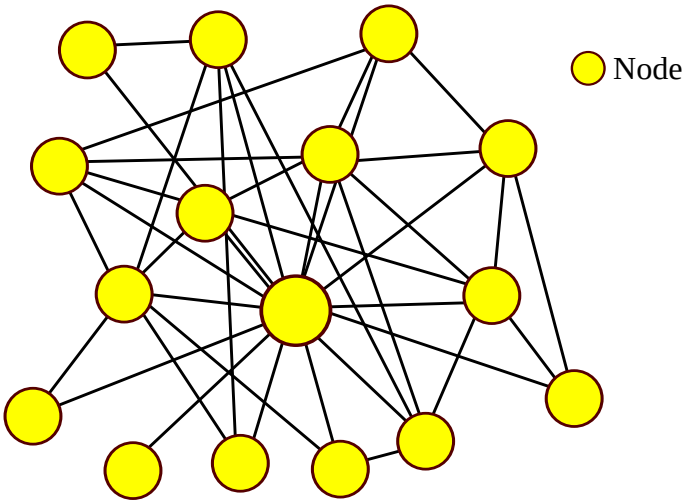


Figure 2.5: An example data grid architecture with scale-free topology.

Benefiting from replica placement with respect to network distance has also been studied by Rahman et al. (2005) in a multi-objective replication strategy. They use p-median and p-center models to select nodes for placing replicas. The p-median model finds p replica placement nodes to optimize the request-weighted average response time. The p-center model selects p replication nodes to minimize maximum response time. Their strategy aims to minimize p-median model by restricting the increase in the p-center objective. By doing this, they minimize average response time without having a requestor too far from a replication node.

Other strategies deal with improved service quality by means of data replication. In this respect, Andronikou et al. (2012) present a data replication strategy that focuses on a new metric called measuring data importance. The importance of data is defined as maximizing profits by satisfying quality of service requirements of the system. They proposed a greedy algorithm and an adaptable heuristic algorithm to make replication decisions. They compared these algorithms to show that the heuristic approach outperformed the greedy algorithm in terms of execution speed.

Minimize Data Missing Rate (MinDmr) (Lei et al., 2008) is another data replication strategy that improves service quality by satisfying availability. Two data availability metrics, *System File Missing Rate* (SFMR) and *System Bytes Missing Rate* (SBMR) are proposed to represent the ratio of the missing number of files and amount of data to total files, respectively. All files have weights according to their availability, number of predicted future accesses, number of copies, and size. The files with lower weights are called cold data and files with higher weights are called hot data. During replica replacement, cold data is deleted first, and hot data has the greater probability of replication.

Not all replication strategies propose new metrics to improve quality of service. As some earlier strategies *Enhanced Fast Spread* (EFS) (Bsoul et al., 2011) extends the existing Fast Spread (Ranganathan and Foster, 2001) replication strategy to improve data locality. In EFS, a replica is created only under two conditions. (i) When enough storage is available, or (ii) replica to be created is more important than the replicas it is replacing. The replica replacement decision is based on a dynamic threshold that takes the number of requests, frequency of requests, size of the replica, and last request time into account.

2.2.3 Analysis

Studying data replication strategies in data grid systems revealed that these strategies mainly target and take advantage of some grid architecture. Among these, more relaxed architectures offer easier data access, with multiple network routes to remote sites. In applications where frequent remote requests are a necessity, these architectures are found to be more suitable than other architectures with stricter topologies. It is safe to say that in the architectures where nodes are connected in a

less restrictive manner, the response time is decreased, regardless of the replication strategy used (Tos et al., 2015) compared to more sparsely interconnected data grid architectures.

Access patterns of files are shaped by user requests (Dogan, 2009; Sashi and Thanamani, 2011). In data grids, stored files are read by clients that process them. Some data processing tasks may require a number of files contained in a data set. In those cases, a data set contains a number of files that have some correlation between their access frequencies. Furthermore, a newly added file to the grid may gain a popularity, or contrarily a popular file of today may not be popular in the future. There are a number of different access patterns used in the literature, including random access patterns that are generated from statistical distributions, e.g. Gaussian and Zipf, and sequential access pattern models. While some studies use three (Ranganathan and Foster, 2001) or even five (Shorfuzzaman et al., 2009; Saadat and Rahmani, 2012; Lee et al., 2012) different access patterns for evaluating performance, others argue for using a more realistic access pattern (Adamic and Huberman, 2002), e.g. Zipf-based. Nevertheless, a sensible model of data access patterns is crucial for realistic evaluation of performance.

Many data replication strategies aim to increase availability. Performance enhancement is often times not directly aimed but still obtained as a consequence of having multiple replicas of files. Multiple replicas allows requestor nodes to have multiple access paths to files, and almost always guarantees lower data transfer times. As having these benefits are tempting, there are replication strategies that always replicate (Ranganathan and Foster, 2001; Dogan, 2009), or create as many replicas as possible (Park et al., 2004; Sashi and Thanamani, 2011). Data grids are purpose-built and its resources are shared among institutions in a federated way

(Foster et al., 2008). Maximizing resource utilization for maximizing desired benefits is a common theme among these data replication strategies.

Grid resources, while being distributed among the institutions in a federated way, are still not free. A drawback of aiming maximized resource utilization in data replication is the increased resource cost incurred on the data grid infrastructure. Some strategies deal with resources being finite and limited, e.g. BHR (Park et al., 2004), most of them do it in terms of resource abundance but not the monetary aspect. As a result, economic impact of the replication is not a frequently visited issue by many proposed data replication strategies in data grid systems. An example case is when a traditional strategy aims to increase availability by filling all of the available storage. Considering the cost of increased storage capacity, it is apparent that replication strategies that create as many replicas as possible will create an economic burden on both the consumer and the service provider.

2.3 Data Replication in Cloud Systems

In the last decade, cloud computing has established itself as a popular computing paradigm. Cloud providers offer seemingly infinite amount of resources to meet ever-increasing storage and computational needs of the tenants (Foster et al., 2008) by benefiting from filling datacenters with commodity hardware. Elastic scaling of abstracted resources (Kouki and Ledoux, 2013) also opened the doors for the cloud providers to offer an economy-based service model (Suleiman et al., 2011). These seemingly infinite resources are rented to tenants as a utility with pay-as-you-go pricing model (Philip Chen and Zhang, 2014).

In a typical cloud environment, where frequent access requests are placed on a

large-scale data, having low response time and high availability is crucial for the tenants. As in many systems, a number of data replication strategies have been proposed (Alami Milani and Jafari Navimipour, 2016; Tabet et al., 2017) to improve the service quality through SLA-awareness in the cloud systems.

Part of the proposed SLA-aware data replication strategies are tenant-centric (Zhao et al., 2014). Some propose replication strategies that focus on minimizing consumption of a certain cloud resource (Vulimiri et al., 2015), e.g. bandwidth, while others deal with replication of databases to satisfy the SLA without considering monetary impact of the replication decisions (Sousa and Machado, 2012).

Performance guarantees, e.g. response time, are often not offered to the tenants by cloud providers as a part of the SLA. Many of the data replication strategies in the cloud focus just on satisfying the availability SLO (Silvestre et al., 2012; Sun et al., 2012a). The lack of performance guarantees in SLAs is also true for the commercial clouds offered by Amazon[1], Google[2], and Microsoft[3]. In order to address this shortcoming, there are several works proposed (Kouki et al., 2011; Sakr and Liu, 2012) in the literature to propose mechanisms to include response time guarantees in the SLA. Dealing with data replication, only a few studies are particularly interested in improved response time (Wei et al., 2010; Bai et al., 2013; Janpet and Wen, 2013; Zhang et al., 2014). Furthermore, even fewer of those studies (Bonvin et al., 2010a; Ghanbari et al., 2012) are taking economic impact of data replication into account.

Data replication is an integral part of the cloud and many distributed data management systems on the cloud such as Google File System (GFS) and Hadoop File System (HDFS) already implement some form of replication. As an example,

[1]http://aws.amazon.com/s3/sla/
[2]https://cloud.google.com/storage/sla
[3]http://azure.microsoft.com/en-us/support/legal/sla/

HDFS creates triple replicas in a rack-aware fashion to enhance both the performance and availability (Lee et al., 2015). However, as the scale of the cloud systems enlarge as the number of datacenters and their geographical disparities increase to the scale of Internet, it is necessary for data replication strategies to address this trend.

The following subsections classify the existing data replication strategies in cloud systems according to their objective and whether they consider the monetary impact of data replication in the replication decision process. Since one of the contributions of this thesis is on the performance satisfaction through data replication, the classification is deliberately based on the satisfaction performance objective or satisfaction of objectives other than performance.

Of course, boundaries of classifying data replication strategies in the cloud may sometimes become not clearly apparent due to a certain amount of overlap in the classification criteria. For example, many strategies that focus on increased availability also observe improved performance even though performance may not be an objective in such strategies. This performance benefit occurs as a consequence of having multiple replicas of data sets. As mentioned earlier, a significant portion of data replication strategies in the cloud ignore the economic impact of data replication. Following classification is done in such a way that highlights this phenomenon in the existing strategies.

2.3.1 Data Replication to Satisfy Objectives Other Than Performance

Arguably the most widely considered objective by data replication in the cloud is the satisfaction of a desired level of availability. Availability of data sets often depends on the availability of the nodes hosting them or their system load, as overloaded

nodes will not be able to serve requests. Increasing availability of data therefore be satisfied by increasing the number of replicas accordingly.

Cost-effective Dynamic Replication Management (CDRM) (Wei et al., 2010) highlights that having too many replicas does not increase availability but instead results in diminished returns. CDRM calculates and maintains a minimum number of replicas to satisfy a given level of availability. In the considered cloud storage, data sets are fragmented into several blocks. Authors calculate the availability of data sets depending on the availability of each of their fragments. With this information, CDRM calculates a required number of replicas for each fragment. Placement of replicas is performed in such a way that balances the load on all sites. Reducing access skew ensures that all of the fragments are served without causing a bottleneck. As a consequence of balanced load, the authors also observe increased access performance. Investigating the relationship between the number of replicas and the level of availability has been the focus of other studies as well (Myint and Naing, 2011). Another approach that is very similar to CDRM (Sun et al., 2012a) also deals with finding minimum number of replicas from a data popularity point of view in cloud storage clusters. In this work, not all data but only a subset of files which exceed an access threshold are considered for replication.

It is also possible to achieve a desired quality of service from an availability standpoint by calculating the minimum level of redundancy with respect to heterogeneous cloud resources (Pamies-Juarez et al., 2011). In these heterogeneous environments, availability calculation is performed with respect to each cloud site having different reliability levels. This heterogeneity in reliability also determines how many replicas can a cloud site is trusted to host as well. The issue of reliability can also be handled as a fault-tolerance problem by mathematically modeling failure rates of cloud sites

(Sun et al., 2012b). This way, the mentioned study makes it possible to satisfy the availability objective through a fault-tolerance framework based on data replication.

Modeling a data placement strategy to improve availability can be also be achieved through custom economic cost models. Some present an auction model to implement a replica placement policy in a large-scale cloud storage environment (Zhang et al., 2014). If the desired availability level cannot be maintained, a bidding is held to determine some placement for a new replica. Bidding price is dependent on several properties of the nodes including failure probability, network bandwidth and available space. Some performance increase is also observed as a consequential benefit of data replication. Others deal with a similar problem in a multi-cloud context (Abouzamazem and Ezhilchelvan, 2013), where tenants receive services from multiple cloud providers. These examples use a custom economic model that is tailored to take into account the heterogeneity caused by resource and pricing variations among multiple cloud providers. Similarly to existing strategies, this economic model is used in order to minimize a cost model(Miglierina et al., 2013).

Some other strategies deal with availability as a reliability issue regarding the cloud sites (Li et al., 2011, 2016). While availability is a frequently considered objective, what sets these two strategies apart is their consideration of cost-effectiveness in satisfying availability. By determining the duration to keep replicas according to a reliability metric, these strategies can optimize the number of replicas for each file, hence saving provider costs on storage resource.

In another similar effort, Skute is introduced by Bonvin et al. (2010a) as a cost-efficient data replication strategy based on a virtual economy that takes into account marginal utility, storage usage and query load. Virtual nodes act autonomously and they periodically announce their rent to other nodes. Moreover, nodes also accu-

mulate wealth by answering queries and spend this wealth on other nodes to store replicas on them, according to their rent. Skute self organizes replica configuration among the virtual nodes to minimize communication cost while maximizing economic benefit. Even though the strategies of this type utilize some form virtual economy, it is not an actual monetary cost model of the provider-tenant relationship (Bonvin et al., 2010b).

Increasing availability by replicating popular data closer to the locations that originate most amount of requests is also a frequently researched problem. Some strategies even propose mechanisms to predict future accesses based on historical access records to preemptively replicate data to meet the increased demand (Ridhawi et al., 2015). Moreover, some other strategies deal with data popularity that peaks for a short amount of time, and then tapers off (Qu and Xiong, 2012). In order to deal with the quick burst of popularity, these data replication strategies quickly respond and change the replica configuration accordingly. Increasing data locality based on popularity also decreases network consumption (Mengxing et al., 2013).

Zeng et al. (2016) deal with replica creation and placement in cloud storage systems dealing with balanced load. In their work, cloud hierarchy is described as service providers buying services from cloud vendors and in turn, clients buying services from these service providers. Authors aim to minimize costs of the service providers while maximizing storage utilization for users. They measure quality of service and calculate an importance metric of data sets. What to replicate is then determined according to the importance metric. Replica placement is performed in a way that maximizes data transfer volume per unit expenditure.

An interesting objective for data replication is to specifically target some expenditure that is particularly interesting in the cloud. In this respect, some data

replication strategies (Boru et al., 2015a,b) focus on minimizing energy consumption and communication delays through data replication. These strategies take data access frequencies into account for a periodical replica reconfiguration. During this periodical assessment, the replication strategy predicts a future value of data sets that include energy and bandwidth demand. According to this estimation, a suitable placement is found for the replicas to minimize their power and bandwidth consumption in the following time periods. Another strategy (Maheshwari et al., 2012) with a similar aim takes the idea further by integrating data placement strategy to be linked with the power controller for a storage cluster. This way, replicas can be reconfigured to scale the cluster up or down and turn of inactive sites according to workload changes in order to save power.

2.3.2 Data Replication for Performance Objective without Economic Consideration

While availability-focused strategies can observe some performance improvement as a result of data replication, this performance increase is not generally resulted from an enforcement by the SLA. In other words, performance is usually not among the service guarantees put forward in the SLA contract to the tenant (Tos et al., 2016) in the strategies of the previous subsection.

The strategies in this subsection commonly takes performance SLA into account and elastically adjust the number of replicas in some way to ensure performance guarantees to the tenant. Some strategies such as RepliC (Sousa and Machado, 2012) perform this task in an elastic multi-tenant database environment. RepliC monitors the system utilization against workload changes in order to handle the variation by directing transactions to the replicas with available resources. As a

result, RepliC strategy satisfies QoS with minimal SLA violations.

Increasing data locality, more specifically spatial locality through data replication yields increased performance with strategically placing replicas closer to the requestor sites. Even though some strategies do not directly consider performance as an objective, increased performance is observed as a consequence of data replication (Lee et al., 2015). Another form of increasing locality of data for improved performance is focusing on temporal locality. In this manner, some strategies argue that frequently accessed data sets will likely stay popular in the future (Jayalakshmi and P, 2015) and decide what to replicate according to the mentioned popularity assessment. This is a simple but effective strategy for satisfying performance guarantees as evidenced by RTRM strategy (Bai et al., 2013). When accessing popular data is causing a higher-than-desired average response time, these data sets are replicated to resolve the performance issue. What is interesting about RTRM is that it replicates not all but just popular data in order to conserve resource usage.

Multi-objective optimization model for data replication is another interesting approach, as described by MORM strategy (Long et al., 2014). Establishing mathematical models for multiple objectives including availability, response time, network latency; data placement can be performed according to an optimization function that takes these objectives into account. This way it is possible to satisfy more than one SLO simultaneously.

The idea of dynamic provisioning of data is employed in database management systems in the cloud as well. Sakr and Liu (2012) introduced an SLA-aware customer-centric strategy, in which the database servers are scaled in and out according to SLA requirements. As the main SLA objective for the decision process, they chose the total execution time of transactions. In the proposed strategy, cloud sys-

tem is closely monitored and cloud providers declaratively define application specific rules to adaptively scale resources. While SLA-aware provisioning is beneficial for scaling, economic impact of the replication for the cloud provider is not considered by this strategy.

While performance benefit can be had by balancing the load of sites, it is also possible to increase performance by decreasing network delays in data access. Sharov et al. (2015) present such a data replication strategy for leader-based cloud storage. One replica of each data set is elected as a leader that coordinates the replication tasks. While all replicas may be used for a read operation, leaders often respond to reads or write transactions. The authors state that placement of leaders impact the performance. They propose three algorithms to handle the placement of the leaders, voters and the replicas to reduce the latency of the data access operations. They show that their strategy improves data access latency by up to 50% in a distributed file system.

Not all data replication strategies deal with performance as a temporal measure of data processing (e.g. response time). Especially in content delivery networks, performance objective may just consist of delivering the data to the requestor. In this sense, AREN (Silvestre et al., 2012) is a good example of enforcing SLA as a data delivery guarantee. AREN performs popularity-aware data replication to optimize bandwidth usage to minimize SLA violations and reduce storage consumption to improve overall user experience.

On a similar performance objective, Zhao (2013) deals with replication delay, i.e. how quickly a database server refreshes all replicas. If the replication delay is longer than a predetermined threshold, new replicas are created to resolve performance degradation. This strategy aims to satisfy the SLA in a tenant-centric fashion,

hence the provider profit is not considered. In another study (Sakr et al., 2011) discuss an elastic, cloud-based database management system that enables automatic management of data management tasks, including a rule based implementation of replication decisions.

2.3.3 Data Replication for Performance Objective with Economic Consideration

Kumar et al. (2014) proposed SWORD, a workload-aware data placement and replica selection scheme. Authors introduced a new metric named query span, which is the average number of nodes used in execution of a query. Their approach aims to minimize query span in order to reduce the communication overhead, resource consumption, energy footprint, and transaction cost. They claim that SWORD deals with performance degradation with incremental repartitioning of data. Although provider profit is not a focus of this study, the authors show the effectiveness of their work by doing an experimental analysis to measure query span and transaction times.

Minimizing cost of resources can also be considered in a heterogeneous cloud context (Mansouri et al., 2017). Datacenters in different locations can have varying costs for storage, network etc. for placing replicas. A replication strategy should take advantage of this heterogeneous pricing while responding with replication to future variations in workloads. As a common theme in many other distributed environments, reducing network transfers between datacenters through replication to both ensure performance due to faster access to data and provider costs by utilizing more expensive network links less frequently is also present in the cloud systems context (Vulimiri et al., 2015).

Xiong et al. (2011) present a resource management strategy that uses machine learning techniques to return an optimum amount of profit from the tenants in a multi-tenant cloud database context. They employ a predictive model to determine a CPU and memory allocation configuration that yields a minimum amount of penalty cost for a given workload. The authors vary the number of replicas to balance the cost of hosting and managing the replicas against the improvement in the performance.

A dynamic data replication strategy is proposed by Gill and Singh (2016) for a heterogeneous cloud environment. Their strategy keeps performance and availability at a desired level while optimizing the cost of replication. The cost of replication is calculated by the unit replication cost per datacenter and whether a replica is placed there. A metric called replica factor is used to calculate weighted access frequencies. Replica factor is compared with a threshold for making replication decision. Placement of replicas is performed with an optimization technique that gets lowest replication cost with highest value of replicas.

Janpet and Wen (2013) designed a data replication strategy to minimize data access time by finding the shortest access path to data objects. They model access frequency, delay and replication budget to find the closest, most suitable node for replica placement. Replication budget is predefined and it is only used as a limiting factor for the users in such a way to regulate number of replicas. A detailed economic relationship between the users and the cloud provider is not addressed. The experimental study shows that by placing data objects closer to the nodes with high access frequency, response time is improved.

A small subset of data replication in the cloud deals with managing how the virtual machines are provisioned on physical hosts. According to provider cost and

performance objective (Ghanbari et al., 2012; Casas et al., 2015). These type of strategies deal with determining how many virtual machines are necessary to perform a set of tasks owned by multiple tenants. Files required for the execution are replicated alongside the virtual machines in order to address the changing demand in the task execution.

2.3.4 Analysis

While many challenges of traditional large-scale data management systems are still valid in cloud systems, there are some new issues that should be taken into account when dealing with data replication in the cloud. These new issues include elastic adjustment of the resources and renting of the services as an utility with pay-as-you-go pricing to name a few. One particular aspect of the cloud that is interesting for data replication is the provider's point of view in terms of monetary cost of operating a cloud in a multi-tenant environment (Lang et al., 2014).

In traditional systems, many available data replication strategies create as many replicas as possible to achieve higher system utilization. Especially in federated systems like data grids, maximized utilization is pursued in order to have maximum performance and prevent resources to be left idle. However, such an approach may not be economically feasible in cloud computing. Cloud providers expect to return some profit from their business relationship with the tenants. Therefore, such an aggressive data replication approach, while possibly satisfying performance, may also result in higher-than-desired operating cost for the provider. Creation and maintenance of an unnecessarily high number of replicas are therefore inevitably bound to have an impact on the provider profit.

As a result of the economic expectations of the provider, new replicas should be

added in order to satisfy SLA requirements, while the removal of replicas occurs when these objectives are satisfied over time. Since SLA requires only the satisfaction of a given level of a quality of service metric, e.g. availability, performance; ideally, the provider should aim to supply the promised service quality, without trying to overachieve. In other words, pursuing best performance for the tenant is unrealistic and most likely a wasteful endeavor for the provider. Instead, the provider should focus on delivering an acceptable quality of service that will maximize its profits.

Therefore, in cloud systems, major questions of what to replicate, when to replicate, how many replicas to create and where to place these replicas must be answered in such a way to satisfy the SLA in an economically feasible way (Tos et al., 2016).

2.4 Conclusion

In this chapter, we studied the state of the art on data replication in some large-scale data management systems. We observed how data replication strategies are adapted their objectives as the computing paradigm is shifted from federated data management systems such as data grids towards economy-based systems, namely cloud computing.

The most common theme in data replication in data grid systems is the emphasis on data access times. Most studies on data replication in data grid systems specifically focus on scientific applications that run on data grids. As multiple institutions access the data that is often generated at a single origin, e.g. physics experiments at CERN, data replication is primarily aimed towards making it easier for the clients to quickly access this data. Other issues such as monetary cost of data replication is often not a visited issue by almost all data replication strategies that are proposed

for the data grid.

On the cloud computing side of the fence, existing data replication strategies are geared towards a wider selection of scenarios. Albeit some strategies still target a scientific cloud environment, many other strategies also exist to deal with content delivery, database management systems and various other applications. As cloud computing is nowadays being immensely popular, this was not a surprising outcome. What was relatively surprising is the decision criteria used in the vast majority of the existing strategies. Cloud computing, as described by Foster et al. (2008), is an economy-based architecture. Many existing strategies deal with availability with a minority of them considering performance as well, but studying economic impact of replication is still open for new research. Only a handful of data replication strategies proposed for the cloud take into account performance and economic benefit simultaneously.

Studying the related work on data replication served us as an evidence of the need of cost-effective data replication strategies that are specifically tailored for the needs of the cloud systems, including satisfaction of SLA-based performance objectives and profitability of the cloud providers. In other words, the work presented in this chapter further reinforced the motivation behind this thesis.

Chapter 3

Proposed Data Replication Strategy

Abstract

Meeting performance expectations of tenants without sacrificing economic benefit is a tough challenge for cloud providers. In this chapter we discuss APER, the proposed data replication strategy that satisfies both performance and provider profit criteria simultaneously. APER estimates the response time of database queries. If the estimated response time is not acceptable, the bottleneck resource is identified. Then, the course of action for data replication to resolve the bottleneck through data replication is determined. Data placement is heuristically performed in such a way to satisfy the query response time at a minimal cost for the provider. Consequently, this chapter provides the details of response time and profit estimation models and the actual data replication strategy itself.

Contents

3.1 Introduction **70**
3.2 Cloud Topology **72**
3.3 Data Replication Issues **74**
3.3.1 Replication Decision 75
3.3.2 Replication Degree 77
3.3.3 Replica Placement 79
3.3.4 Replica Retirement 82
3.4 A Cost Model for DB Query Processing **83**
3.4.1 Response Time Estimation 84
3.4.2 Resource Consumption 91
3.5 An Economic Model for DB Query Processing in the Cloud . . . **96**
3.5.1 Estimating Provider Profit 97
3.5.2 Estimating Provider Revenue 99
3.5.3 Estimating Provider Expenditures 100
3.6 Conclusion **105**

3.1 Introduction

As briefly discussed in Chapter 1, simultaneous satisfaction of performance objective and provider profit is the core aim of this thesis. Proposed cost model predicts whether a newly arrived tenant query would satisfy the performance guarantee and

economic model is responsible for assessing whether processing of that particular query is profitable for the provider. Neither of these models are enough on their own to form a complete data replication strategy as they are merely tools to assist in making replication decisions. Hence, this chapter describes the proposed strategy for *Achieving query Performance in the cloud via a cost-Effective data Replication* (APER), as well as how the two models are integrated in the replication decisions at the core of the proposed strategy.

APER takes advantages of some properties of cloud systems, such as a global network hierarchy and heterogeneous resource availability and pricing. Therefore, we start with the chapter by laying out the details of the considered cloud environment and how the various aspects of the cloud would impact APER on making replication decisions. Afterwards, the details of when to start data replication, how many replicas to create and where to place the newly created replicas are discussed. Of course, all of these replication decisions are handled with the consideration of performance and provider profit objectives. For each query, APER tries to make a replication decision that satisfies the tenant's performance expectation with returning most amount of profit for the provider.

APER deals with database queries for analytical purposes, i.e. query processing for OLAP applications. Therefore, the proposed data replication strategy deals with a cost-effective replica management in a read-only data management context. Other challenges that are typically associated with data replication, e.g. consistency of data updates, are not in the scope of this thesis.

3.2 Cloud Topology

Cloud providers often establish multiple facilities in separate geographical regions for a multitude of reasons, including providing services that span across the globe. Each region may contain several subregions. These subregions are cloud facilities that host a number of nodes that provide computational power and storage to the tenants. The cloud system we consider consists of geographical regions with each region containing a number of datacenters. Expectedly, each datacenter in turn contains a number of servers, i.e. virtual machines (VM), that reside on physical hosts. Each server is equipped with computational resources, e.g. CPU, network, storage, to contribute during query execution.

Regions, datacenters and servers are interconnected via network links in a hierarchical manner. This hierarchy describes the network bandwidth as it is relatively cheaper and far more abundant inside a datacenter while it is more expensive and less abundant between regions where Internet infrastructure is usually employed. As the network hierarchy goes from inter-node links to inter-region links, bandwidth abundance decreases and bandwidth cost increases (Park et al., 2004). A typical example of this cloud hierarchy is depicted in Figure 3.1.

Tenants utilize the services they rent from the provider by placing queries to the cloud. These queries require data sets that may reside on multiple servers scattered around different geographical regions. From the tenant's perspective, it is essential for the response time of an average query to be within the threshold defined in the SLA. The cloud provider aims to satisfy the SLA with the maximum amount of profit. The essence of the proposed strategy contains two models. Former is the cost model based on a response time estimation and latter is the economic cost model used for provider profit estimation.

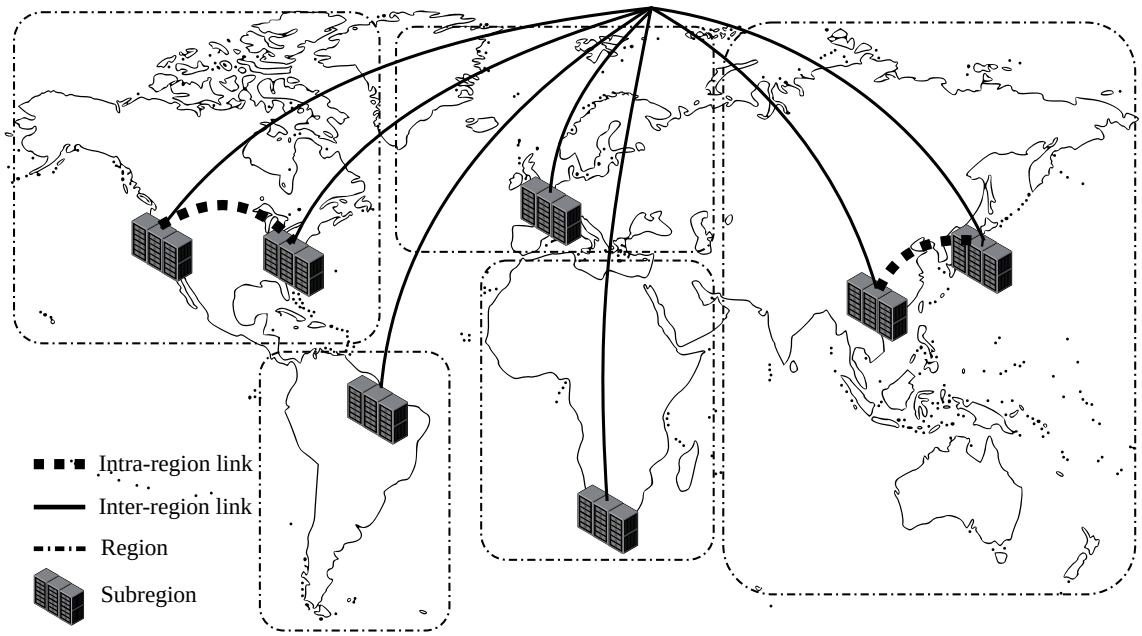


Figure 3.1: An example cloud topology showing regions and subregions (datacenters).

It should also be noted that, processing data in the cloud has many challenges including distributed execution and partitioning. The focus of our study is only on the data replication aspect of data management in the cloud.

An important characteristic that differentiates the cloud from traditional data management systems is the penalty mechanism. Should an SLA breach occur, the provider is obligated to pay an agreed upon monetary sum to the tenant (Kouki et al., 2011). In our case, when the actual response time is found to be greater than the threshold defined in the SLA, it is indicative of an SLA violation. It is therefore important to note that, penalties play an important role in the economics of the cloud.

3.3 Data Replication Issues

Any data replication strategy should be addressing a few key concerns in the decision process for replication to constitute a complete strategy that is suitable for managing data for satisfaction of a certain objective. Some of these concerns have been described in many traditional systems, particularly in data grids (Ranganathan and Foster, 2001), as what to replicate, when to replicate, and where to place the replicas. However, in economy-based data management systems, i.e. particularly in cloud systems, these decisions must be made from a cost-efficient perspective. We therefore, extend these considerations in the data replication process to take both the tenant and provider's point of views into account.

First item on the agenda of any data replication strategy should be correctly identifying what data to replicate. Especially in database query processing, a query may require multiple relations during the execution. A replication strategy should correctly identify if any of these relations poses a bottleneck that may hamper the response time of the query. Incorrectly identifying what to replicate would just result in wasted resources without any actual benefit for performance. In the proposed strategy this step is handled by the response time estimation model.

When to replicate is the next important step towards performing data replication. A set of solid criteria based on the desired goal to achieve is key at this step. Replicating too early or too frequently would result in inefficient use of resources and also reduce the performance due to frequent re-occurrence of the replication overhead. Too late or too lazily performing data replication is also harmful since it would erase the benefit of data replication due to missing the opportunity for providing actual benefit.

Degree of replication is another important issue in data replication. Too few

number of replicas may not satisfy the data access load of the queries while too many replicas would increase data management costs for the provider due to wasteful use of resources. A good middle ground is to dynamically adjust the degree of replication according to the changing demand of tenant queries.

Placement of the planned creation of replicas is equally important as the previous issues. While finding the best placement is a costly venture due to it being an NP-complete problem (Tang and Xu, 2005), finding a good placement in a timely manner to satisfy the requirements of the system is key in this step. Therefore, a data replication should find a cost-effective placement quickly in order to reduce the overhead of data replication and not lose time over trying to find the optimal placement.

Finally, the removal of unnecessary replicas should also be handled by the data replication strategy. Over time, some replicas may become unused due to changing demand in the data management system. To free up resources for future replicas and increase profitability by reducing resource consumption, these replicas should be retired from the system as the queries come and go through the cloud.

In the following sections, we discuss how APER strategy deals with these issues of data replication in a cost-effective manner to satisfy the performance guarantees.

3.3.1 Replication Decision

First, and most fundamental decision in APER strategy is whether to actually perform data replication or not. The replication decision depends on the fulfillment of two important criteria. (i) If the estimated response time ($\mathcal{T}_C$) of a query $\mathcal{Q}$ is greater than the threshold response time enforced by the SLA ($\mathcal{T}_{SLO}$) and (ii) processing of that particular query is estimated by the profit estimation (P_Q) to generate a min-

imum desired profit (P_{Th}) for the provider, with the inclusion of potential creation of any new replicas. The steps of making the decision to replicate are depicted in Algorithm 1. Of course, the profitability prediction of the query includes the cost of new replicas as well.

Algorithm 1 Replication decision on query processing.

```
1: Receive query Q
2: T_C ← Estimated response time of Q
3: if T_C > T_SLO then
4:    Select bottleneck data for replication
5:    Find a placement for new replicas that would satisfy T_C < T_SLO
6:    P_Q ← Estimated profit by executing Q with the new replicas
7:    if (P_Q > P_Th) ∧ (T_C < T_SLO) then
8:       Place the new replicas
9:    end if
10: end if
11: Continue executing Q
```

When a tenant submits a query to the cloud for processing, first order of business is to predict whether that particular query can satisfy the response time objective with the given execution plan. Response time estimation of the cost model of APER handles this task. Moreover, in the given query plan, which operator causes the bottleneck that would ultimately result in an SLA breach is identified by the response time estimation.

Dealing with a query that consists of multiple operators, it is important to measure which operator (hence what data related to that operator) will experience a bottleneck in what resource (e.g. network, CPU) in order to resolve the bottleneck and satisfy SLA through data replication. Therefore the data associated with the replication is always the bottleneck data for each replication event during query execution.

In case a query is predicted to violate the SLA performance threshold, then

comes the latter part of the replication decision. The second important criterion to fulfill is the provider profitability. Assuming that creation of some new replicas is considered, the processing of the query must still be profitable with the inclusion of the costs associated with the new replicas. Of course, this necessitates finding an economical placement for the new replicas. The provider expects to return a desired amount of profit (P_{Th}), on average, from every query execution. For this purpose, the profit estimation model predicts the satisfaction of this condition. Only the simultaneous satisfaction of the two mentioned criteria would trigger the replication to be carried out.

As described, APER makes the decision whether to replicate on arrival of each and every query. Another alternative is to perform the replica reconfiguration on a periodical basis. Of course, the calculations required for estimating response time and profit might introduce some overhead to query execution, APER has the benefit of immediately responding to changing trends in the queries arriving at the cloud. While periodical replica reconfiguration may have less overhead, an obvious drawback of such an approach would be a slow response to changing access trends. A further discussion on this is left for future work as mentioned in Chapter 5.

3.3.2 Replication Degree

Determining the number of replicas for each piece of data is an interesting aspect of data replication. Too many replicas may cause wasteful use of resources, while too few replicas would not be enough to satisfy expected performance level. Some data replication strategies predefine (Fu et al., 2013) a degree of replication for the data management system. The number of replicas may then be re-adjusted periodically. APER makes the replication decisions dynamically and on demand, on arrival of

each new query. Therefore the degree of replication is also determined in a similar manner.

When the necessity of data replication arises and the required conditions for going through with the replication is met, APER increments the number of replicas for a particular fragment by one. APER aims to place the replica in such a way that it would satisfy the needs of both the provider and the tenant. Therefore, there is no need to create more than one replica at a time for a particular fragment. It is true to say that the number of replicas is dynamically adjusted as the queries are processed in the cloud.

Naturally, a given query can consist of a number of operators that require processing of several fragments. It is possible that more than one of these fragments can be predicted to cause a bottleneck. In cases like this, of course, APER can consider replicating as many of these fragments as it is necessary to resolve the bottleneck condition to satisfy the response time. However, each of these fragments would still be replicated incrementally. In this manner, APER does not enforce an upper limit for the number of replicas of any particular fragment. Consequently, as long as it is still profitable and necessary, the number of replicas of any fragment can be incremented many times as the queries come and go.

Satisfying an agreed upon performance for the tenant is indeed a crucial part of APER. However, ensuring the availability of fragments is another objective that is a commonly present in SLAs (Stamou et al., 2013). Therefore, availability should not be ignored for the purpose of focusing on performance. There are many studies in the literature that dare dealing with replicating data for improved availability (Qu and Xiong, 2012; Abouzamazem and Ezhilchelvan, 2013). Therefore, we do not revisit this issue in this strategy. Our strategy simply maintains a minimum

number of replicas (e.g. triplication in the HDFS) to satisfy a minimum required availability for each fragment. The number of replicas are never allowed to fall below this minimum during replica removal operations.

3.3.3 Replica Placement

The two estimation models makes it possible to identify resource bottlenecks to determine what to replicate and also decide when the replication is necessary. However it is the strategic placement of the replicas that actually achieves the response time and profitability objectives during execution.

A candidate server for placing the new replica should have low load, enough storage space and sufficient network bandwidth to serve the new replica to the requestors. Furthermore, the candidate placement should also incur low cost to store the new replica.

Clouds can be heterogeneous in many ways. Storage, CPU and network costs for the provider can change from region to region, or datacenter to datacenter for several reasons, e.g. different prices of raw materials. Also, not all regions can satisfy the response time of queries originating from other regions. For example, accessing a fragment in a datacenter in Europe from another datacenter in North America can satisfy the response time SLO, while a datacenter in Asia may fail to do so. Therefore it is a very hard task to find the best placement that satisfies the response time SLO with the lowest cost.

Finding the best placement requires all servers in the cloud to be evaluated for load, storage and network resources, as well as considering the cost of storing it there. Taking into account that the cloud consists of a seemingly infinite number of servers, this approach is non-trivial. Finding a placement for the new replica

should be completed in a time sensitive manner. Lingering on for pursuing the best placement would waste time and in turn erase the benefit of actually performing the replication.

For every query execution, when the necessity of creating a new replica of a fragment arises, a placement heuristic is used by our strategy to find a good placement. Considering all geographical regions of the cloud, only a number of them may satisfy the response time SLO by having mathematically enough network bandwidth. The placement heuristic simply determines this by evaluating the network bandwidth between the requestor and originator regions. The next issue is to find which particular datacenter in these subset of regions has the lowest execution cost with the new replica. The placement heuristic reduces the search space by first eliminating unsuitable regions and datacenters successively and then finding the first suitable server instead of searching for the best one.

Let the cloud L consist of a number of geographical regions as $L = \{G_1, G_2, ..., G_i\}$. In turn, each of these geographical regions may contain a number of datacenters as $G_i = \{D_{i,1}, D_{i,2}, ..., D_{i,j}\}$. Let an estimation function $\mathcal{T}'$ in Equation 3.1 determine the predicted response time of executing a query $\mathcal{Q}$ with the assumption of a new replica of $R^m_{n,l}$ placed to region G_i. A candidate subset G' regions that can satisfy the response time SLO can be found by evaluating all regions by the $\mathcal{T}'$ function.

$$G' \subset G | \mathcal{T}'(\mathcal{Q}, R^m_{n,l}, G_i) < \mathcal{T}_{SLO} \tag{3.1}$$

In the next step, the placement focuses on the provider profit. Among the datacenters belong to the G' set of regions, placement heuristic choses the D' datacenter, which offers storage for $R^m_{n,l}$ that results in the lowest cost of execution for the query $\mathcal{Q}$, as depicted in Equation 3.2.

$$D' | \min(Cost(\mathcal{Q}, R^m_{n,l}, D_{i,j}) \wedge D_{i,j} \subset G') \tag{3.2}$$

Network bandwidth inside a datacenter uses the dedicated local network infrastructure. This highly controlled network capability is more or less uniform. Therefore if a datacenter is found to have sufficient network capability, it does not matter to store the new replica on which particular server, as long as the candidate server has low load and enough storage.

$$N_m \in D' | load(N_m) < Th_{load} \wedge freeSpace(N_m) \geq sizeof(R_{n,l}) \tag{3.3}$$

The replica is placed on the server N_m of the datacenter D'. The condition for N_m is to have enough storage space to store the fragment $R_{n,l}$ and have a computational load that is lower than a load threshold Th_{load} as shown in Equation 3.3. Th_{load} is a system parameter that is predetermined by the cloud provider and may be included

Algorithm 2 Placement heuristic for placing new replicas.

1: set $G' \leftarrow$ empty set
2: **for all** Regions $G_i \in$ Cloud L **do**
3: **if** $\mathcal{T}(\mathcal{Q}, R^m_{n,l}, G_i) < \mathcal{T}_{SLO}$ **then**
4: Add G_i to set G' {regions that has enough bandwidth to satisfy response time}
5: **end if**
6: **end for**
7: $D' \leftarrow \min(Cost(\mathcal{Q}, R^m_{n,l}, D_{i,j}) \wedge (D_{i,j} \subset G'))$ {datacenter that offers the lowest cost with the new replica}
8: **for all** Servers $N_m \in D'$ **do**
9: **if** $load(N) < Th_{load}$ **and** $freeSpace(N_m) \geq sizeof(R_{n,l})$ **then**
10: place on N_m {the server with lower-than-threshold load and enough free storage}
11: **end if**
12: **end for**

in the SLA.

Using the equations that describe the search process for candidate servers, the complete algorithm for finding placement for new replicas is depicted in Algorithm 2.

3.3.4 Replica Retirement

Over time, some replicas may get less and less frequent accesses due to changing trends in the arrival of the queries. Keeping these replicas in the system consumes precious storage and causes unnecessary costs for the provider. Therefore, retiring unused replicas is a way of eliminating burden on the provider profit. While other replication decisions are made on a per-query basis, determining which replicas are sparsely used is a task that must be undertaken with respect to usage information over time.

APER periodically evaluates all replicas to decide on their eligibility for removal. The duration of this period (Per_{remove}) for considering replica removal is determined by the provider. If the response time SLO is satisfied by the queries that require a particular relation, in a certain period (Per_{sat}), it is regarded that some replicas of the fragments of that relation may be unnecessary, therefore safe for removal. In other words, that a relation R_n can be fragmented into l fragments as $R_{n,l}$ and these fragments may have replicas on j servers as $R^j_{n,l}$. If the queries that require R_n have no difficulty in satisfying the response time over a predetermined period of time, APER may consider that there may be an abundance in the degree of replication for the fragments of R_n.

In order to determine which of the mentioned replicas are unnecessary, each of them are checked for their access histories. Among them, those with an access count below a threshold (Th_{access}) are retired from the cloud. Of course, the

number of replicas for any particular fragment is never allowed to drop below the minimum number of replicas (Th_{numRep}) to satisfy a given availability level. The replica removal algorithm is depicted in Algorithm 3. The thresholds for the removal operation and the time period to initiate replica removal parameters are left to the discretion of the provider. If cloud providers desire a more aggressive replica removal for quickly freeing up cloud resources or vice versa, they are free to adjust these parameters accordingly.

Algorithm 3 Retirement of unnecessary replicas.

```
1: for Each period of Per_remove do
2:   for all R_n relations do
3:     if Queries that require R_n satisfied SLA for a period of Per_sat then
4:       for all R_{n,l} fragments of R_n do
5:         for all R^j_{n,l} replicas of fragment R_{n,l} do
6:           if accessCount(R^j_{n,l}) < Th_access and
             numberOfReplicas(R_{n,l}) > Th_numRep then
7:             remove R^j_{n,l}
8:           end if
9:         end for
10:      end for
11:    end if
12:  end for
13:  Reset access counts of all replicas
14: end for
```

3.4 A Cost Model for DB Query Processing

As discussed in Chapter 1, APER, data replication strategy proposed in this thesis regards both the satisfaction of query performance and provider profit simultaneously. In this section, how the former criterion, namely the performance metric is treated in the proposed strategy. The data management aspect of this thesis is focused on a relational database system hosted in the cloud, response time of database

queries is regarded as the primary metric for performance.

Processing a database query in a distributed context involves a set of steps that include query decomposition (syntax and semantic analysis), resource discovery, generating query plans etc. among many other tasks that are handled by the data management system. APER starts with the assumption that, a query optimizer has already generated a near-optimal query plan. APER strategy is positioned after generating the mentioned query plan and just before carrying out processing of the query. Hence, query plan generation and other related tasks of query execution are not a focus of this thesis as APER only deals with the data replication aspect in the query execution process.

3.4.1 Response Time Estimation

The query plan generation mechanism of DBMS tries to generate a near-optimal query plan according the state of the cloud in terms of location of required relations, load of the servers etc. The generated query plan may not still satisfy performance threshold indicated in the SLA. In other words, a near-optimal query plan may not necessarily guarantee the performance expectation of the tenant. The response time estimation model in this section, is therefore aims to determine whether a given query plan is viable or not in satisfying SLA.

3.4.1.1 Estimating Response Time of Database Queries

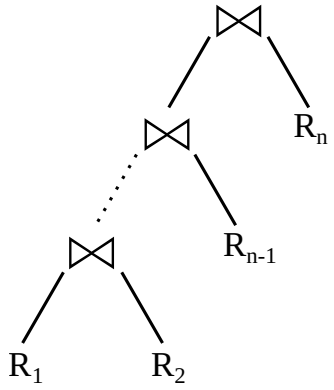
In distributed environments such as clouds, database queries can be processed by participation of a number of servers. These servers process queries with (i) inter-query parallelism as executing multiple queries in parallel in the cloud, and (ii) intra-query parallelism as executing individual parts of any single query in parallel.

During processing, database queries require relations, or fragments of relations to be present. These fragments may be scattered across multiple servers. Processing of a query constitute a multitude of operators that can be executed in a pipeline. Furthermore, multiple operators can be executed in parallel with intra-operator parallelism concept. These operators themselves may be divided to be executed in parallel on multiple servers with intra-operator parallelism, which makes the response time estimation even more challenging. Moreover, the result of an operator often can be the input of another as intermediate results. Query processing is therefore not a straightforward task, but it is a set of operations that are intertwined in a complicated manner with presence of parallel execution.

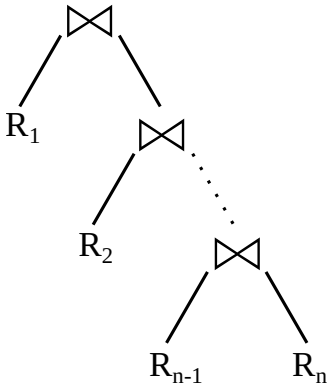
Estimating response time of a query of such complex intricacies is a significant challenge. Dealing with these issues, estimating response time of database queries has been an active research area for decades, as the response time estimation constitutes an integral part of query optimizers in DBMSs (Hsiao et al., 1994; Swami and Schiefer, 1994). Therefore in APER, we benefit from several existing studies (Tomov et al., 1999, 2004) to propose a response time estimation model that is suitable for the proposed data replication strategy. These existing studies follow an approach that estimates resource usage (e.g. CPU) by the queries. A novel aspect of response time estimation in APER is that it uses the information on estimated resource consumption by also considering the monetary cost of these resources in replication decisions.

Estimating resource usage of each query as soon as it is submitted is a significant challenge. Many issues as mentioned above, especially parallel execution complicates the task of calculating an accurate estimation. Response time estimation for a single simple query, i.e. without joins, is a relatively straightforward task (Tos et al.,

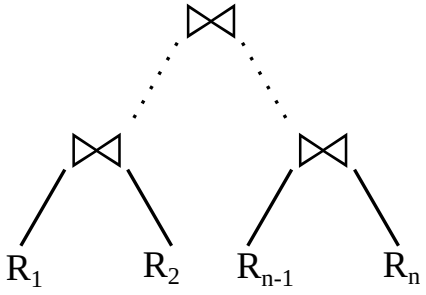
2016). In these queries, there are no dependent operations and as a result, estimating response time is just an approximation of execution and data transfer times. In contrast, estimating the response time of a query that consists of multiple join operators is significantly more difficult. Identifying pipelines, computational complexity of each operator in these pipelines, the amount of data required to be shipped from remote servers all come into play in estimating a complicated query.



(a) Left-deep query plan



(b) Right-deep query plan



(c) Bush query plan

Figure 3.2: Query plans that are considered by the response time estimation model.

In order to model the response time estimation over a realistic case, we deal with queries that consist of a multitude of hash join operations. In query plans with multiple hash joins, left-deep, right-deep and bushy plans are investigated (Figure 3.2). We design our response time estimation model to cover all three of these query plan types. Generating query execution plans, and selecting a suitable

plan for execution is not a part of this thesis. The scope of the proposed work begins when a new query with a given execution plan is estimated to not satisfy the SLA response time threshold. At that time, APER estimates whether this query plan can satisfy the performance objective, and takes action accordingly.

Initially, APER is given a query execution plan $\mathcal{P} < \mathcal{Q}, N >$ of a query $\mathcal{Q}$ that will be executed on a set of servers N. APER estimates the response time of $\mathcal{Q}$ by calculating the expected resource usage before the execution of a query takes place. Accurate estimation of the usage of these resources is a significantly hard task with considering both inter-operator and intra-operator parallelism. First step in attacking this task is to identify the pipeline chains in the given query plan to deal with inter-operator parallelism.

3.4.1.2 Intra-operator Parallelism

Intra-operator parallelism is another issue that must be addressed by the response time estimation model. Intra-operator parallelism deals with parallel execution of an operator on several servers and combining the results generated on each server afterwards. No matter how the fragmentation was done by the DBMS, i.e. horizontal or vertical, there may be response variations between the execution of an operator on different fragments due to skew, available CPU etc. As a result, the response time of an operator that is executed on multiple servers and on multiple fragments is determined by the execution of the operator on a particular one of those fragments that yields the largest amount of response time.

Let C_i be a pipeline chain in the query $\mathcal{Q}$. This pipeline chain C_i consists of a number of operators, $C_i = \{O_{i,1}, O_{i,2}, ..., O_{i,k}\}$ that are executed in a pipeline. Let an operator $O_{i,k}$ to require a relation R_n for processing. Assuming that the relation

R_n is fragmented into l fragments as $R_n = \{R_{n,1}, R_{n,2}, ..., R_{n,l}\}$, with replicas on j servers as $N = \{N_1, N_2, ..., N_j\}$; the operator $O_{i,k}$ will be executed on each of this fragments in parallel as $O^j_{i,k}$. In this case, the estimated response time of $O_{i,k}$ is determined by the longest estimated amount of time it takes for $O_{i,k}$ to execute on a certain particular fragment $R^j_{n,l}$ as shown in Equation 3.4.

$$\mathcal{T}_{O_{i,k}} = \max_j(\mathcal{T}(O^j_{i,k}), \mathcal{T}_{Tr}, \mathcal{T}_{Fin})|O^j_{i,k} \in O_{i,k} \tag{3.4}$$

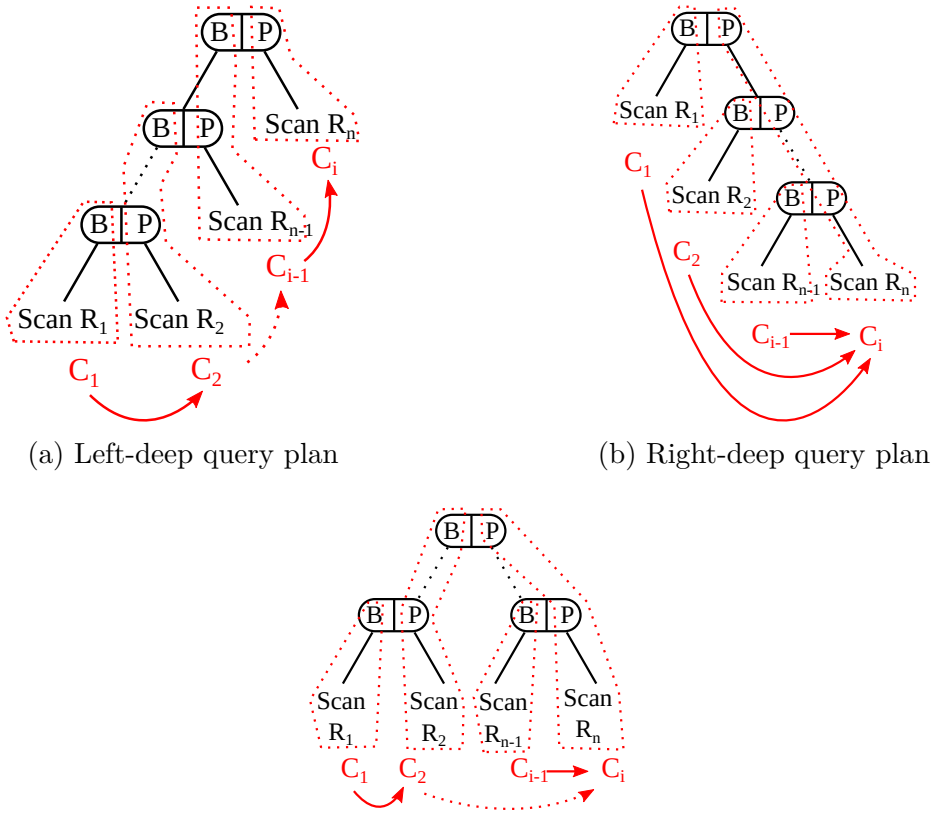
When an operator is executed on multiple fragments in parallel, the results are required to be transferred to a final destination for combining. When estimating the response time $\mathcal{T}_{O_{i,k}}$ of the operator $O_{i,k}$, Equation 3.4 takes this into account by considering transfer of results ($\mathcal{T}_{Tr}$) and producing the final result ($\mathcal{T}_{Fin}$), e.g. union operation for joining horizontally fragmented relations.

3.4.1.3 Inter-operator Parallelism and Pipeline Chains

In order to determine the pipeline chains in the query plan, at the first step, APER assesses in what order the operators are going to be executed. In this sense, when we deal with inter-operator parallelism, we take into account both independent and pipelined execution. In query execution, some operators block data flow between operators (Garofalakis and Ioannidis, 1996; Tomov et al., 2004; Wu et al., 2013) that is necessary for pipelined execution. For example, during hash join, hash tables must be built before the probing stage. Therefore, these blocking operators are identified and the query is deconstructed into pipeline chains according to them (Garofalakis and Ioannidis, 1996; Tomov et al., 2004; Wu et al., 2013).

An example of how the pipeline chains are identified in the left-deep, right-deep and bush query plans is depicted in Figure 3.3. The set of operators that belong in

each pipeline chain are executed in a pipeline, until the last operator in the respective pipeline chain, which is ultimately a blocking operator.



(a) Left-deep query plan

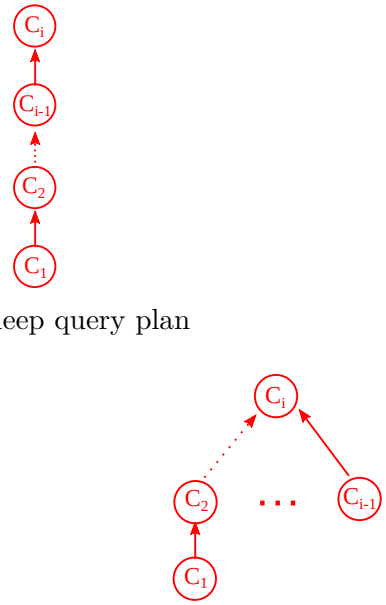
(b) Right-deep query plan

(c) Bushy query plan

Figure 3.3: Pipeline chains in left-deep, right-deep and bushy query plans (Garofalakis and Ioannidis, 1996).

As a query plan can consist of any number pipeline chains and dependencies among them, it is possible to show this relationship as a tree $Y < V, E >$ (Figure 3.4). Vertices V of this tree represent each individual pipeline chain and edges E represent the dependencies between them. With inter-operator parallelism, some pipeline chains are executed simultaneously on different servers. This parallelism is represented by each root-to-leaf path of the pipeline chain tree (Lanzelotte et al., 1994; Özsu and Valduriez, 2011). In this sense, the response time estimation of a

query can be formulated as follows.



(a) Left-deep query plan

(b) Right-deep query plan

(c) Bushy query plan

Figure 3.4: Dependencies among pipeline chains in left-deep, right-deep and bushy query plans.

Let $C = \{C_1, C_2, ..., C_i\}$ be the set of pipeline chains containing a number of operators in the query plan $\mathcal{P}$. Moreover, let $W = \{W_1, W_2, ..., W_m\}$ be the set of the root-to-leaf paths for all ℓ leaf nodes of a pipeline chain tree $Y < V, E >$, in the given query plan. Consequently, the estimated response time of the query $\mathcal{Q}$ is as shown in Equation 3.5, with respect to dependencies and between pipelines.

$$\mathcal{T}_C = \max_m \sum_1^\ell \mathcal{T}(V_\ell) | V_\ell \in W_m \tag{3.5}$$

Estimated response time $\mathcal{T}_C$ of the query $\mathcal{Q}$ is the estimated response time of the longest execution branch (in terms of response time) in the pipeline chain tree $Y < V, E >$. An execution branch in a query plan can consist of a number of pipeline chain that are executed in succession as depicted in Figure 3.4. The response time

estimation model takes this into account when dealing with multiple pipeline chains in the query plan.

Considering that C_i consists of a number of operators, $C_i = \{O_{i,1}, O_{i,2}, ..., O_{i,k}\}$ that are executed in a pipeline, some operator depend on the first generated result tuple of the previous operator further up the pipeline chain. In other words, operators that belong to the same pipeline chain does not necessarily start processing at the same time. Therefore, for operators other than the first, there is a pipelining delay, equal to the mentioned waiting duration (Lanzelotte et al., 1994; Özsu and Valduriez, 2011). In order to estimate the response time of each individual pipeline chain, this pipelining delay of each operator $O_{i,k}$ in the pipeline chain C_i, must also be estimated to be able to estimate the response time $\mathcal{T}_{C_i}$ of pipeline chain C_i (Lanzelotte et al., 1994; Özsu and Valduriez, 2011), as depicted in Equation 3.6.

$$\mathcal{T}_{C_i} = \max_k(\mathcal{T}_{O_{i,k}} + pipeDelay_{O_{i,k}}) \tag{3.6}$$

Execution of C_i starts with processing $O_{i,1}$, therefore, starting time of $O_{i,2}$ depends of the response time of $O_{i,1}$. This waiting time is the pipelining delay mentioned above. Response time estimation model recursively estimates the pipelining delay of each operator by going back from the last one to the first since the pipelining delay of the first operator in any pipeline chain is zero.

3.4.2 Resource Consumption

So far, the response time estimation dealt with how the pipeline chains are identified, how they consist of multiple operators and how the execution of these operators impact on response time estimation. The actual amount of time consumed by each

individual operator has not yet been addressed. During execution, an operator consumes CPU time resources for calculations, e.g. calculating hash tables, I/O resource for data access e.g. performing scans, and network resource for shipping fragments located in remote servers. Each of these resources consume time during query processing, hence the response time of an operator is a combination of the amount of time spent by the consumption of these resources.

In this subsection, we explain how the response time of an operator is determined with respect to its resource consumption in terms of the resources mentioned above. This process involves estimating the CPU, I/O and network usage of all operators in their respective pipeline chains (Hsiao et al., 1994). This estimation will also play a role in determining which fragments to replicate. Assuming that an operator $O_{i,k}$ in the pipeline chain C_i is executed on $R^j_{n,l}$ fragment at server j, the estimation model calculates which resource type will cause a response time violation by causing a bottleneck.

3.4.2.1 CPU Consumption

Consumption of CPU by an operator depends on how much CPU time it is necessary to be spent on actual CPU instructions. First consideration towards estimating CPU consumption is the amount of data to be processed by an operator. Naturally, the more data an operator needs to process, the longer the response time is expected to be. In this sense, we take the terms $t(R^j_{n,l})$ and $sR^j_{n,l})$ into account, which denote the number of tuples and size of a tuple in the fragment $R^j_{n,l}$, respectively.

Some operators are more computationally expensive compared to others. For example, hash table generation requires more instructions to be executed by the CPU per a single byte of data, compared to relatively simple and cheaper scan op-

eration. This variation between the operators are taken into account by the $IPB_{O_{i,k}}$ multiplier, which denotes how many CPU instructions are required by processing one byte of data by any particular operator.

$$\mathcal{T}^{CPU}_{O^j_{i,k}} = \frac{t(R^j_{n,l}) \times s(R^j_{n,l}) \times IPB_{O_{i,k}}}{r^{CPU}_j} \tag{3.7}$$

Consumption of the CPU resource by an operator is depicted in Equation 3.7. In the equation, we have dealt with data size and number of instructions required to process a unit amount of data. In order to have a result in the time unit, we divide these multipliers by r^{CPU}_j. r^{CPU}_j indicates how many instructions the CPU of the server j can process per unit amount of time. This value is scaled as the number of operators executing simultaneously on a particular server changes. If more than one operators are simultaneously processed on one server, CPU time is consequently divided among them. Therefore, r^{CPU}_j is a function of arrival rate, which affects the number of operators simultaneously executing on the servers.

3.4.2.2 I/O Consumption

During query processing, the relations that are required for that query is read from the disk of the server hosting the data. Whether it is for shipping the relation to another server or processing the relation locally in a query, these operations consume I/O resource on that particular server.

The amount of time of time spent on consuming I/O resource depends on the size of the data read from the local storage and the I/O throughput specific to that server, i.e. how many bytes can be read per unit amount of time. In both cases described above, the same I/O consumption model is used to estimate the I/O related response time component of the total response time of any particular query.

In both components, the data size of $R^j_{n,l}$ is utilized, similarly to the CPU consumption model.

$$\mathcal{T}^{I/O}_{O^j_{i,k}} = \frac{t(R^j_{n,l}) \times s(R^j_{n,l})}{r^{I/O}_j} \tag{3.8}$$

I/O consumption is relatively straightforward. On the server N_j that hosts the fragment $R^j_{n,l}$ that is relevant in the processing of an operator $O^j_{i,k}$; the fragment $R^j_{n,l}$ is read over an I/O throughput of that particular server, $r^{I/O}_j$ as depicted in Equation 3.8.

3.4.2.3 Network Consumption

When a server participates in processing of a query, ideally it is desired for this server to have the relevant data to be locally present. In some cases, a portion of required data may be required to be shipped from remote nodes. In another case, if the execution is migrated to another server due a reason that makes the initial server to be temporarily non-viable for query execution, e.g. having a temporarily high load, the associated data may also be required to another server alongside the execution. These type of query processing scenarios require the consumption of network resource. Obviously, consumption of network resource means a duration of time to be spent for transferring data. Response time estimation model also takes this into account when predicting response time of queries.

$$\mathcal{T}^{Net}_{O^j_{i,k}} = \frac{t(R^e_{n,l}) \times s(R^e_{n,l})}{r^{Net}_{j,e}} \tag{3.9}$$

The equation for estimating the network consumption (Equation 3.9) is similar to estimating I/O consumption in the way that both of them are base on the amount

of data handled. This time, data transfer occurs between two servers, therefore the estimated time spent on network transfer is determined by bandwidth $r_{j,e}^{Net}$ between the local server N_j and the remote server N_e during the shipping of the remote fragment $R_{n,l}^{e}$. We assume that messaging delay for initiating the data transfer process is constant, and included in the total transfer time.

3.4.2.4 Total Resource Consumption

The three types of resources that are consumed during processing of a query are used in calculating the response time $\mathcal{T}_{O_{i,k}^{j}}$ of the operators individually.

Of course, for each operator the consumption of these resources do not happen sequentially. Their consumption are time shared on the executing server. For example, CPU can start processing a relation that is shipped from a remote server to the local server as soon as a meaningful amount of data e.g. a page, has becomes available locally. In that sense, the total resource consumption takes time sharing into account by taking the maximum time spent with respect to time-shared utilization of different types of resources as shown as shown in Equation 3.10.

$$\mathcal{T}_{O_{i,k}^{j}} = \max((\mathcal{T}_{O_{i,k}^{j}}^{CPU} + \mathcal{T}_{O_{i,k}^{j}}^{I/O}), \mathcal{T}_{O_{i,k}^{j}}^{Net}) \tag{3.10}$$

The resource consumption model used in response time estimation inevitably makes some simplifications in order to rapidly estimate the response time of the queries on arrival. Too complex of a model would increase the computational overhead caused by data replication and in return possibly erase the performance gains expected to be made through data replication. For this purpose, we assume that CPU is occupied during I/O operations and tread the consumption of CPU and I/O to be not happen simultaneously on a single server.

It should be noted that all three types of considered resources take the data size that is relevant to a particular operator into account. In real query processing scenarios, some operators would need intermediate relations as input for processing them. Intermediate relations are outputs of other operators that precede the actual operator in the query execution plan. Since these intermediate relations simply do not exist before the execution of the query starts, the DBMS may not have statistical information about them, e.g. data size. Some research effort have been made to estimate the result sizes (Vengerov et al., 2015; Harangsri, 1998) and cardinalities (Swami and Schiefer, 1994) of the intermediate relations. We use the methods proposed by these studies to obtain a more accurate estimated response time.

3.5 An Economic Model for DB Query Processing in the Cloud

Cloud computing is a computer paradigm that brings several new challenges for data replication strategies to deal with. These challenges have been discussed in detail in Chapter 2. Among the new challenges of the cloud, a particular one is of significant interest for the data replication strategy proposed in this thesis.

In contrast to other traditional data management systems that precede the cloud, resources are rented to the tenants with an economy based model in cloud computing (Foster et al., 2008). It is indeed true as the cloud providers rent the services to the tenants in return of some monetary compensation from them. What this signify is that, operating in the cloud should take economic impact of every decision into account. More specifically, since the context of this thesis is in query execution and data replication in a multi-tenant cloud environment; the economic burden of

processing the queries and associated costs of data replication on the cloud provider is of utmost importance.

The economic model presented in this section deals with the economic aspect of the cloud while processing database queries with the presence of data replication. Replication decisions are based on satisfaction of both the response time satisfaction and provider profit generation. While the response time estimation handles the former part of the condition, the economic part of whether the execution of a query is still profitable for the provider with possible creation of new replicas is handled by the economic model.

3.5.1 Estimating Provider Profit

Dealing with database management in the cloud, requires an immense undertaking by cloud enterprises, similarly to delivering other cloud-based services. Cloud providers establish enormous facilities to host computing resources and process data, pay significant upfront investment in material and personnel, and occupy global network links with transferring data throughout the globe. Inevitably, all of these expenditures add up to a significant monetary sum for cloud providers as operating costs.

For a cloud provider, the entire sum of investment and effort is motivated by gaining economic benefit, namely returning a monetary profit (Buyya et al., 2009). As it is the case for all business enterprises, the cloud providers seek profit from their interactions with their tenants. Ideally, every interaction (e.g. query execution) with a tenant should be profitable from the provider's point of view.

While providers aim to return a healthy amount of profit for their benefit, tenants demand a good performance for a reasonable price. Focusing on supplying a service

with the best performance is often a costly pursuit for the provider (Tos et al., 2016). Eagerly replicating data to attain the lowest possible response time may result in high storage and network costs for the provider as a consequence. Among the possible replication decisions, the data replication strategy should be able to go through with the one that just satisfies the response time SLO and yield the most amount of economic benefit to the provider. A profit estimation model is therefore necessary to predict the monetary impact of the resource consumption of processing queries.

Replication decisions are made at the per-query level. As a result, similar to the response time estimation model, the profit estimation is also calculated for each individual query. Equation 3.11 shows the provider profit (P_Q) for executing a query $\mathcal{Q}$, which is the difference between the revenue and total cost ($\mathcal{C}_Q$) associated with that particular query.

$$P_Q = REV_Q - \mathcal{C}_Q \tag{3.11}$$

It is very straightforward to estimate provider profit with known revenue and expenditures. What is challenging about profit estimation lies in the difficulty of estimating revenue and expenditures for each query the provider processes for each tenant. Some properties that are specific to cloud systems such as dynamically adjusted service capability with pay-as-you-go pricing, heterogeneity of services and their unit costs with respect to global region make it more challenging to estimate the profit.

3.5.2 Estimating Provider Revenue

Cloud providers are large business entities, and as such, it is possible for a provider to have multiple revenue streams. However, in this thesis, we are only interested in the revenue generated from the tenants for query processing. Therefore, in the proposed economic model, rent is regarded as the only revenue source of the provider. In this way, APER strategy estimates the average revenue expected from processing each tenant query.

SLA terms include the details for the rent and the corresponding billing period. A billing period is an agreed upon duration of time that the tenant acquires services from the provider. The tenant pays the corresponding amount of rent (REV) to the cloud provider for the services acquired during each billing period.

Naturally, the provider desires to return a profit from each query execution. APER calculates the per query revenue and cost estimates to predict an expected profit for each query. However, in order to be able to calculate the per-query profit, the SLA should also include the maximum amount of queries the agreed upon service level handles per unit amount of time, i.e. maximum query arrival rate (AR), before scaling the service up.

From the maximum query arrival rate and the duration of the billing period (PER), it is possible to calculate an expected average revenue REV_Q per query as depicted in Equation 3.12.

$$REV_Q = \frac{REV}{PER \times AR} \tag{3.12}$$

It should be noted that, it is not realistically possible to know beforehand that, how many queries a tenant will submit to the cloud with what arrival rate. There-

fore, the revenue estimation calculates the average revenue expected from each tenant query. Assuming that the set of services are elastically scaled up and down according to the query arrival rate (and possibly other metrics) of a particular tenant, revenue estimation provides quick prediction of per-query revenue that is accurate enough for replication decisions.

3.5.3 Estimating Provider Expenditures

With a known amount of expected revenue from each query processed, next issue to address is estimating the monetary cost of processing a particular query. Estimating revenue is relatively straightforward to estimating expenditures, because revenue prediction is based on an average. In contrast, expenditures caused by a particular query depends on the cloud resources it will consume once the execution is started. Therefore, estimating monetary cost of a query is a comparatively more difficult task that involves more steps.

In the cloud environment, where distributed query processing is a natural occurrence, execution of a query $\mathcal{Q}$ may involve the participation of a number of servers. The resources on these servers may be fully or partially utilized by the execution of $\mathcal{Q}$. These resources inevitably cost money for the provider (Greenberg et al., 2009). As in response time estimation model, APER takes into account CPU cost ($\mathcal{C}_U$), network bandwidth cost ($\mathcal{C}_B$) and storage cost ($\mathcal{C}_S$) resources, as well as cost of the penalties ($\mathcal{C}_Y$) in the previous billing period. The sum of these estimated costs determines the total cost of executing a particular query for the provider. These four types of costs are shown in Equation 3.13 as they are used in the profit estimation. How each of these cost items are estimated is described in detail in the following sections.

$$\mathcal{C}_Q = \mathcal{C}_U + \mathcal{C}_B + \mathcal{C}_S + \mathcal{C}_Y \tag{3.13}$$

In real world, cloud providers have many types of costs, ranging from employee salaries to on-site physical security. These costs may not be directly linked to, or rather caused by data replication or query execution. However, provider naturally still would want to have enough profit margin to cover these costs as well. As a result, we do not set the profit condition for triggering replication to $P_Q > 0$, but rather $P_Q > P_{Th}$. P_{Th} describes a predefined threshold profit margin as the minimum amount of profit a provider desires to generate from the execution of a query.

3.5.3.1 CPU Cost

First cost item to estimate in the breakdown of total cost of processing query $\mathcal{Q}$ by the provider is predicting CPU cost ($\mathcal{C}_U$). CPU cost depends on the CPU time that is expected to be spend during the execution of a particular query $\mathcal{Q}$. Queries can be processed by the joint effort of a number of servers in a distributed environment. Therefore, CPU cost is determined by the sum of CPU time consumption of each server (U_j) that is expected to participate in the execution.

Let $R = \{R_1, R_2, ..., R_n\}$ be the set of relations with l fragments as $R_{n,l}$ that are used in the execution of $\mathcal{Q}$ and $N = \{N_1, N_2, ..., N_j\}$ be the set of servers that host j replicas of these fragments as $R^j_{n,l}$ and carry out the actual execution task. Then the processing cost is as depicted in Equation 3.14.

$$\mathcal{C}_U = \sum_j U_j \times \phi_j \tag{3.14}$$

Considering that the cloud is a heterogeneous environment, the unit processing cost (ϕ_j) may differ according to which particular servers are utilized during execution. Unit processing cost is adjusted in the estimation to reflect the regional differences. For example, high power costs in a particular geographical region may ultimately lead to a higher CPU unit cost as power consumption of a server depends on the CPU consumption.

3.5.3.2 I/O Cost

While I/O cost depend on the size of the relations that is expected to be used in processing a query, I/O cost estimation is not directly attributed to the amount of storage used by the fragments of these relations. Storage use on its own would not be enough to estimate I/O usage because in case when a new replica is created, not just one but all future queries would benefit from it.

Some queries are more data intensive than the others. These queries may require a particular fragment to be read several times during an execution. Therefore, a realistic storage cost calculation should not just consider the size of the fragments but also consider the amount of disk read operations caused by accessing any particular relation fragment.

$$\mathcal{C}_S = \sum_j \sum_e \sum_n \sum_l sizeof(R_{n,l}) \times \omega_{je} \times s_{jenl} \tag{3.15}$$

$$s_{jenl} = \begin{cases} 1 & \text{if } R_{n,l} \text{ is read from node } N_e \text{ by } N_j \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

Simply, while all queries benefit from stored fragments and replicas, more storage

cost should be incurred to the data-intensive queries accordingly. Therefore, an I/O cost calculation (Equation 3.15) is performed in a similar manner as in the network cost calculation. However, this time the unit cost of I/O (ω_{je}) is used when accessing a fragment on server N_e by requestor server N_j to consider any potential read or write operations. Since the intermediate relations of pipelined operations are not stored, they are not included in the storage cost. The indices of the unit cost is used to indicate the heterogeneity of different cloud regions having different unit cost values. A storage factor s_{jenl} shows which particular replica is accessed among the many possible replicas of the same fragment, as shown in Equation 3.16.

3.5.3.3 Network Cost

Query processing in distributed environments such as cloud systems, inevitably causes some network resource to be consumed due to data transfer between servers. The cost of data transfer is another cost item that the provider has to deal with during query execution.

For a certain amount of data transfer, network cost inside a datacenter is considerably cheaper than transferring data between geographical regions. Inside a datacenter, network links are owned and operated by the cloud provider. On the other hand, inter-region data transfer usually performed over the Internet, hence yielding more cost. Furthermore, accessing different geographical regions from the same requestor server likely to net different unit costs simply due to heterogeneous bandwidth availability and pricing toward various regions. As a result, the unit cost of network transfers (ν) is dependent on where the local N_j and remote N_e servers are located in the cloud.

Since query $\mathcal{Q}$ may consist of many operators that may be executed on a num-

ber of servers, a multitude of inter-server data transfers involving several remote fragments ($R_{n,l}$) may take place. The factor b_{jenl} is used to signify which particular replica is accessed among the many possible replicas of $R_{n,l}$ in the cloud. Equations 3.17 and 3.18 show the network cost and b_{jenl} factor, respectively.

$$\mathcal{C}_B = \sum_j \sum_e \sum_n \sum_l sizeof(R_{n,l}) \times \nu_{je} \times b_{jenl} \tag{3.17}$$

$$b_{jenl} = \begin{cases} 1 & \text{if } R_{n,l} \text{ migrated from node } N_e \text{ to } N_j \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

3.5.3.4 Penalties

SLA terms mandate the provider to complete the execution of every query to satisfy the service guarantees given to the tenant. While providers take necessary precautions (e.g. data replication in our case) to prevent SLA breaches from happening, in reality there may be some queries executed with response times greater than SLO response time threshold. In these cases, the provider pays the tenants an agreed upon sum of penalty, as described in the SLA (Wu et al., 2011; Sousa and Machado, 2012).

Penalty total is not the cost of a consumed resource associated with neither data replication nor query execution. However, amount of penalties incurred to the provider depends on the effectiveness of the data replication strategy. A good data replication strategy should be able to enable query execution to meet the SLA and as a consequence, reduce the penalties for the provider.

$$\mathcal{C}_Y = \sum_t \frac{NUMQ_t \times PQB_t \times \rho}{PER_t \times AR_t} \tag{3.19}$$

Whether a query has respected SLA response time or not can be observed after the execution is completed. Therefore, the actual penalty cannot be reliably determined by an estimation. Moreover, while the resources consumed by a particular tenant query is backed by the revenue generated by that particular query, the penalties are paid to all tenants by the single provider. As Equation 3.19 shows, for each tenant t, the amount of penalties inflicted per query in the next billing period will depend on the total number of queries processed ($NUMQ_t$) and percentage of them breaching the SLA (PBQ_t) and per-query unit penalty amount (ρ). Using this formula, we include the increased cost of queries due to penalties paid by the provider.

3.6 Conclusion

In chapter, we presented APER, a dynamic data replication strategy to satisfy both the response time objective and economic benefit of the provider for processing database queries in cloud systems. APER estimates the response time of database queries before execution and predicts whether the response time objective is going to be satisfied. If a query is estimated to violate the SLA, the proposed strategy considers placing a new replica based on placement heuristic to carry on the execution with an acceptable response time. Of course, the processing of this query is still required to be profitable with the new replica. An economical model predicts the provider costs for executing that particular query and makes sure that profitability

is ensured before the replication is carried out. For all query execution events, this replication decision mechanism is evaluated to identify bottleneck data and replicate them as necessary.

Response time estimation model in APER considers the consumption of the cloud resources in left-deep, right-deep and bushy join query plans. Profit estimation is based on the estimation of both the revenues and expenditures of the provider when executing a tenant query. Based on predicted resource consumptions, we estimate the monetary cost of executing each query while expected revenue is calculated from the rent collected from the tenants.

The outputs of cost model and economic model of database query execution is utilized in every aspect of the replication decisions. Once these two criteria are met and data replication is being carried out, relevant fragments are replicated in an incremental manner. Placement of these replicas are also handled in a cost-effective manner that also satisfies the performance according to the placement heuristic. Moreover, unnecessary replicas are retired from the system over time to further improve the expenditures of the provider.

Chapter 4

Performance Evaluation

Abstract

In this chapter, the performance of APER, the proposed strategy is validated in an experimental evaluation. Throughout the chapter, we first establish the testbed for the performance evaluation. This includes selection and extension of a simulation tool for the tests, preparation of the simulated cloud topology and the generation of query load. Later, the performance of APER is studied in a comparative study against another data replication strategy. The results are analyzed in a discussion throughout the rest of the chapter.

Contents

4.1 Introduction **108**

4.2 Simulation Environment **109**

4.2.1 Cloud Simulator 110

4.2.2 Simulated Cloud Topology 112

4.2.3 Simulated Query Load 113

4.2.4 Simulation Parameters 115
4.2.5 Comparison of Data Replication Strategies 116
4.3 Simulation Results **117**
4.3.1 Measured Metrics 117
4.3.2 Average Response Time 118
4.3.3 Number of Replications 119
4.3.4 Storage Usage 120
4.3.5 Network Usage 121
4.3.6 Number of SLA Violations 122
4.3.7 Monetary Expenditures of the Provider 123
4.4 Conclusion **124**

4.1 Introduction

Another crucial step on proposing a novel data replication strategy is to validate the performance of the proposed strategy through experimental evaluation. Of course, obtaining repeatable results has a key importance in the performance evaluation study. Experimental evaluation of APER strategy can be summarized in two steps. (i) Preparation of a testbed for evaluating the performance and (ii) actually performing the experiments themselves.

First, a controlled test environment is established to represent a database management system in the cloud that will process a query load with predetermined parameters. For this purpose, we tailored CloudSim (Calheiros et al., 2011) simulation tool by customizing it to suit the need of our simulation scenarios. In CloudSim,

APER strategy is implemented as well as CDRM (Wei et al., 2010), another data replication strategy. We employed these two strategies to be used in a comparative analysis. The test bed contains all the necessary system requirements for evaluating query performance with the presence of data replication. Moreover, the testbed allows for measuring monetary costs of the provider resulting from resource consumption.

In the second part of the performance evaluation, the results are obtained from processing queries for a predetermined timespan. Of course, the necessary metrics regarding the performance analysis is collected during the tests for both of the evaluated data replication strategies. In this part, the aim is to highlight how APER strategy excels in simultaneously satisfying both the performance and provider profit objectives against the other strategy.

4.2 Simulation Environment

Cloud systems are volatile environments in terms of load and resource availability. Especially when considering that many tenants share the same set of resources through abstraction, it is possible to have continuously changing performance. This situation may prevent us from performing controlled experiments (Zeng et al., 2016) that is crucial for demonstrating the performance of the proposed strategy. Simply, we need to be able to control all properties of the cloud environment to have repeatable results. This necessity pushes us towards using a simulation environment, rather than a real cloud.

There are several aspects of the simulated environment that are important for achieving a realistic model of the desired data management system operating in the

cloud. Some noticeable issues dealt with in preparation of the simulation environment include the following. (i) Having a simulation tool that is capable of simulating database query execution in the cloud with the presence of data replication. (ii) Creation of a simulated cloud topology that reflects a realistic cloud environment, e.g. multiple geographical regions, heterogeneous resource availability etc. (iii) A query load with coherent properties such as arrival rate and average number of joins. Of course, these requirements of the simulation environment also necessitate careful selection of all simulation parameters.

4.2.1 Cloud Simulator

Creating an entire cloud topology by renting resources from a cloud provider is a very hard and expensive way of verifying the performance of any study. Furthermore, this performance evaluation study requires us to assume the role of a cloud provider, which is very likely to cause real cloud providers to hesitate donating this kind of access privilege. In order to address these concerns of researchers, several cloud simulators have been proposed in the literature (Ahmed and Sabyasachi, 2014). These tools let researchers to accurately implement performance evaluation scenarios without the burden and cost of dealing with an actual cloud environment.

All available cloud simulators focus on a key aspect of the cloud, hence designed around that key property. For example, GreenCloud simulator is specifically focuses on simulating energy consumption of datacenters. While other examples exist that are interested in some other properties of the cloud systems, as of writing this thesis manuscript, no particular cloud simulator specifically deal with data replication. Among the available simulators, CloudSim (Calheiros et al., 2011) is noticeably widely used in the literature. CloudSim is also an open source project, therefore

it is possible to tailor the simulation tool with some extensions to better suit the requirements of the performance evaluation study in this thesis.

In its standard form, CloudSim is targeted for resource provisioning in datacenters in terms of allocating virtual machines on physical hosts. It allows for the creation of simple tasks called cloudlets and models the execution of cloudlets on virtual machines. In a typical usage scenario of CloudSim, number of virtual machines are scaled with respect to the load generated by processing cloudlets. Like many other cloud simulators however, mechanisms for performing data replication is not present in CloudSim. As a result, an extension in functionality of CloudSim is necessary to enhance it with support for data replication; before realizing any simulation scenarios involving the proposed strategy.

First, the cloudlet model of CloudSim is extended to have them better represent database queries. Cloudlets are atomic tasks, independent from each other. This model is extended to have dependencies between cloudlets, which is necessary since processing a database query that may consist of operations with inter-dependencies. Furthermore, the storage model is also extended to allow for virtual machines to have their own storage, instead of CloudSims own storage model which uses a centralized storage for each datacenter. Next, the necessary changes for measuring important metrics is added to CloudSim. These include monetary cost of resources, which was not measurable in the original form of CloudSim. Most importantly, data replication support is added to CloudSim. With this extension, data replication can be triggered before processing each query or at periodical intervals depending on the requirement of the data replication strategy to be simulated. With these changes, CloudSim is ready to be used for the performance evaluation purposes of this thesis.

4.2.2 Simulated Cloud Topology

As cloud systems often offer services on a global-scale, we realized a cloud topology that consist of 5 geographical regions. These geographical regions represent continent-scale zones that contains datacenters. In turn, each region contains 3 datacenters, which host the actual computational resources. In each of these datacenters, 10 servers are realized as virtual machines. These virtual machines possess processing, storage and network capabilities to process the queries assigned to them. The regions, datacenters and virtual machines are interconnected as shown in Figure 4.1.

In the simulations, we realized a hierarchical network bandwidth capability. As the global hierarchy of the cloud goes from regions to towards smaller scales towards the VMs, the network bandwidth is changed in parallel. The network infrastructure is established in such a way that, the bandwidth capacity is more abundant and cheaper at the lower levels of the hierarchy, i.e. inside the datacenters, but less abundant and more expensive towards the higher levels, i.e. between geographical regions.

In addition to network bandwidth, computational capabilities and storage capacities of virtual machines also vary from datacenter to datacenter to simulate a heterogeneous cloud environment. This heterogeneity is not only limited to the capacities of these resources but also extended to their unit costs as well. In every region, the provider costs of these resources are varied to achieve a more realistic cloud system. Therefore, both the performance and cost of the CPU, I/O and network resources depend on the virtual machines participate the processing of any particular query. Each VM is equipped with computational resources (e.g. CPU, storage etc.) essential to perform the execution of queries. During execution, VMs

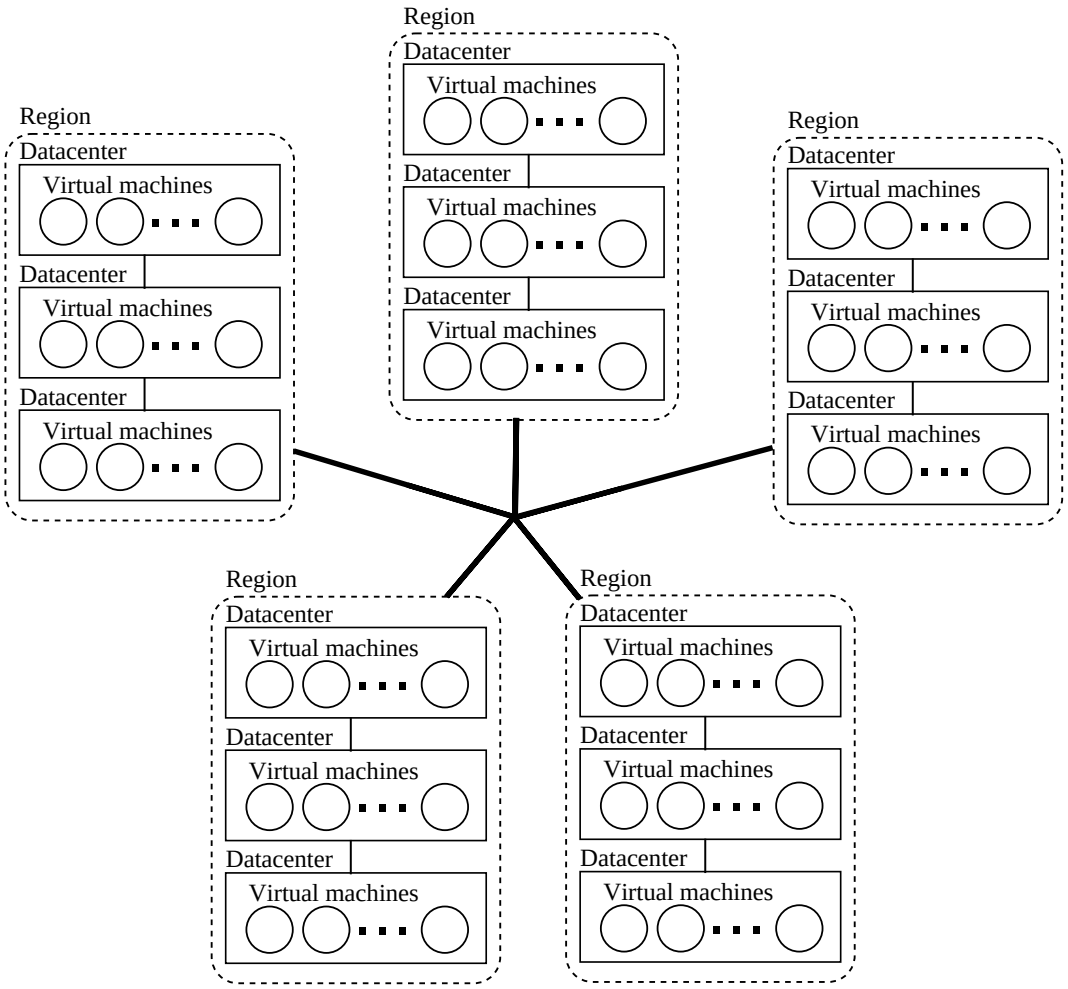


Figure 4.1: The cloud topology realized in the simulation.

can access the data sets in other VMs by remote reads or by replicating them to local storage.

4.2.3 Simulated Query Load

After having a capable simulation tool equipped with a suitable simulation topology, next item on the agenda is to generate the query plans that will serve as the load for the cloud system. This query load is processed by the cloud. Some of these queries may trigger data replication to satisfy the objectives of the studied data replication

strategies.

We generated a query load of 10000 queries. On generation, each query plan is determined to have a number of joins that vary between 1 to 5. The exact number of join operations each query to have is randomly determined on generation to create a computational variation between different queries. By dealing with different number of joins and ultimately different computational loads caused by each query, the simulation scenario is aimed to be more realistic. Of course, generated queries requires a number of relations for processing depending on the number of joins. All of the generated queries have a right-deep query plan to take advantage of parallel execution in the cloud. These queries are randomly created and submitted to the cloud during the simulation. For all simulation scenarios, same initial placement of the fragments is used to ensure fair starting conditions.

Resource allocation during query processing is not a focus of this thesis, therefore newly generated queries are accepted with their given query execution plan. This execution plan is assumed to already have the target servers that are selected for executing that particular query. In the simulations, when queries are submitted to the cloud, they start processing in the same region they are submitted to, by using servers that are most suitable with respect to their load and hosted fragments in their storage. Queries are generated at random interval with an average arrival rate and submitted to the cloud system. Generated queries require a number of fragments that is dependent on their execution plan, e.g. number of joins. Moreover, computational load generated by each query is also randomly changed from one query to another to simulate the computational load variations between queries, e.g. a simple projection versus an aggregate function.

4.2.4 Simulation Parameters

In the simulations, we used a specific topology and query load as described in the earlier sections. However, the simulation environment is further tailored with a specific set of parameters that describe the other properties of the cloud. These include, network bandwidth capacities, storage availability of the virtual machines, unit cost of resources among many other system parameters. Table 4.1 shows these specific set of system parameters used in the simulation scenarios.

The simulation parameters must be in accordance with the real cloud environments to realistically model query processing on a real cloud. Some of these parameters are relatively easier to model after real clouds, however others, especially the unit cost of resources is difficult to obtain as they are usually treated as trade secrets by the cloud providers to mask their profit margins. Therefore, we referred to the existing studies (Barroso et al., 2013) on datacenter infrastructures to realistically choose these system parameters to represent an accurate cloud environment.

Other system parameters such as the arrival rate of the queries are determined empirically to properly demonstrate the operation of the evaluated strategies in a meaningful manner. For example, a low arrival rate would result in a relaxed system that is not strained by the query load. Such a scenario may not necessitate or benefit from data replication. On the other hand, an extremely high query load that is not possibly be handled by the existing capability of the cloud would also not yield meaningful results from an opposing sense. Therefore, the selected simulation parameters allows us to evaluate performance at the middle ground of the two mentioned cases.

Table 4.1: Simulation parameters used in the performance evaluation.

Parameter	Value
VM processing capability	1000 to 2000 MIPS
VM storage capacity	10 to 20 GB
Intra-datacenter bandwidth	8 to 12 Gbit/s
Intra-region bandwidth	2.75 to 3.25 Gbit/s
Inter-region bandwidth	0.15 to 0.25 Gbit/s
Avg. Intra-datacenter delay	5 to 10 ms
Avg. Intra-region delay	25 to 50 ms
Avg. Inter-region delay	100 to 150 ms
Response time SLO	120 s
Query arrival rate	16.67 query/s
Average load by query	1000 to 7500 MI
Number of relations	30
Avg. size of a relation	600 MB
Intra-datacenter bw cost	\$0.0005 per GB
Intra-region bw cost	\$0.002 per GB
Inter-region bw cost	\$0.04 per GB
I/O cost	\$0.05 to 0.15 per TB
Processing cost	\$2 to \$4 per 10^9 MI
Penalty cost	\$0.5 per 1000 violation
Revenue	\$0.5 per 1000 query
Th_{numRep}	3

4.2.5 Comparison of Data Replication Strategies

APER strategy brings cost-effective satisfaction of query response times as a novel contribution to large-scale data management in the cloud. However, highlighting the impact of this novelty is better conveyed through a comparative performance evaluation study against a more traditional data replication strategy. Hence, we compared the performance of APER with CDRM (Wei et al., 2010), another data replication strategy proposed for cloud systems.

Main focus of CDRM is maintaining a minimum number of replicas to satisfy a given level of availability. It calculates a minimum number of replicas to sat-

isfy a given level of availability and it further increases the number of replicas if necessary according to a proposed metric called blocking probability. The authors propose a queuing model that takes into account the arrival rate and service rate for each server. According to this model, CDRM aims to reduce the impact of access skew, hence improving balanced load. New replicas are placed according to blocking probability to ensure the continuity of a balanced load throughout the cloud.

CDRM promises improved performance, however it does not take the economic aspect of data replication into account. While it achieves availability and balanced load, CDRM does not consider SLA satisfaction. Because of these properties CDRM is a very suitable candidate for comparing APER to show how the economic impact of SLA-aware data replication affect the profitability of the provider.

4.3 Simulation Results

The simulation study is conducted by evaluating the generated query load on the established cloud topology for the duration of time that is described in the simulation parameters. During the simulations, several metrics are collected for further analysis. In the following subsections, some discussion on the collected metrics are given to highlight how the two studied data replication strategies handle the query load by performing replication.

4.3.1 Measured Metrics

The performance of both APER and CDRM is analyzed over some key metrics measured during the simulations. While simulation scenarios are executed by CloudSim, several system parameters are logged to be further discussed in their respective sub-

sections. These logs are then processed at the end of the simulation to obtain values of the desired metrics. These include, average response time of the queries, total number of replication events, used storage percentage of the entire cloud as well as I/O consumption, network bandwidth consumption and finally the number of SLA violations. Table 4.2 shows the measured values of these metrics.

Table 4.2: Simulation results.

Replication strategy	APER	CDRM
Average response time (s)	64.7	351.7
Number of replications	908	796
Storage use (%)	22	14
Inter-region transfer (GB)	27.50	335.53
Intra-region transfer (GB)	301.05	149.71
Intra-datacenter transfer (GB)	302.25	147.07
Number of SLA violations	533	5005

What these metrics signify regarding the performance of the evaluated data replication strategies is discussed in the following subsections.

4.3.2 Average Response Time

Average response time metric indicates, on average, how long it takes for a query to produce a response starting from its submission. As the tenant's expectation from the provider is the satisfaction of the performance objective, average response time is key in demonstrating whether the evaluated strategies performed acceptably.

APER satisfied the response time with respect to the threshold set in the SLA. There are some queries that produced a response with a less-than ideal response times during the initial replica reconfiguration. However, after the starting period where replicas are starting to be distributed across the cloud, APER yielded ac-

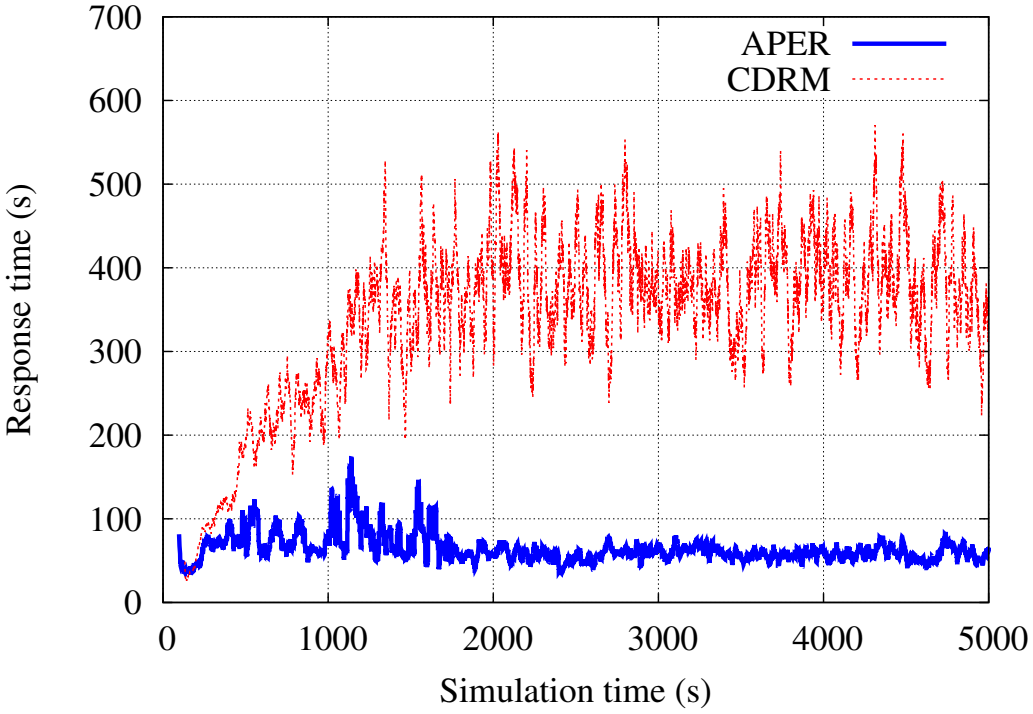


Figure 4.2: Response time of the queries during simulation.

ceptable response times throughout the simulation, as shown in Figure 4.2. Low data access times as a result of good replica placement by APER is a key factor in this result. Since balanced load does not necessarily guarantee the satisfaction of performance, CDRM provided a higher average response time compared to the SLA threshold.

4.3.3 Number of Replications

During the simulation, both strategies performed relatively similar number of replications. The marginal difference is resulted from the replication decision criteria of the two strategies. In other words, APER replicated data more often than the CDRM, in order to satisfy the response time SLO. On the other hand, CDRM did

not continue to create more replicas to satisfy the response time once the balanced load is achieved. The difference is further evidenced by the fact that, APER considers cost-effective satisfaction of response time for triggering replication while CDRM only considers access frequencies of fragments.

In an example case where a query requires an unpopular but large data, APER may perform replication if necessary due to a predicted SLA breach. On the other hand, CDRM would not replicate it even though long data transfer time would violate the SLA since unpopular data is uninteresting for CDRM. Both strategies pursue different objectives it may be possible that the satisfaction of performance necessitate more replications to be triggered compared to the satisfaction of balanced load in this specific simulation scenario.

4.3.4 Storage Usage

APER used 8% more amount of storage space compared to CDRM. While this affects the expenditures of the provider, there is a certain trade-off behind this decision. On replication decisions, APER evaluates which decision is more economically feasible for the provider, doing a remote read or replicating the data to a more suitable location. In the simulation scenarios, apparently performing data replication is more economical for the provider compared to remote reads in a larger percentage of time.

APER followed this route in order to avoid (i) high utilization of expensive network links and (ii) avoiding penalties by taking advantage of having more replicas to satisfy the response time. Of course, APER would not create these replicas if they were not profitable for the provider to have them. Lower number of replicas of CDRM can be attributed to the core decision mechanism of the strategy. In these

simulation scenarios, evidently, it takes less number of replicas to ensure availability and balanced load compared to satisfying response time objective. This is further apparent in the number of SLA breaches for the two strategies.

4.3.5 Network Usage

Network usage highly impact the costs of the provider and also data transfer times, as the network links vary both in pricing and throughput depending on the type of the link, e.g. intra-datacenter network area. In this sense, APER performed a majority of data transfers inside regions and datacenters, as shown in Figure 4.3. This is not a coincidence. APER often chose to replicate data residing in remote geographical locations to more locally available servers, preferably in the same datacenters as the requestors are located.

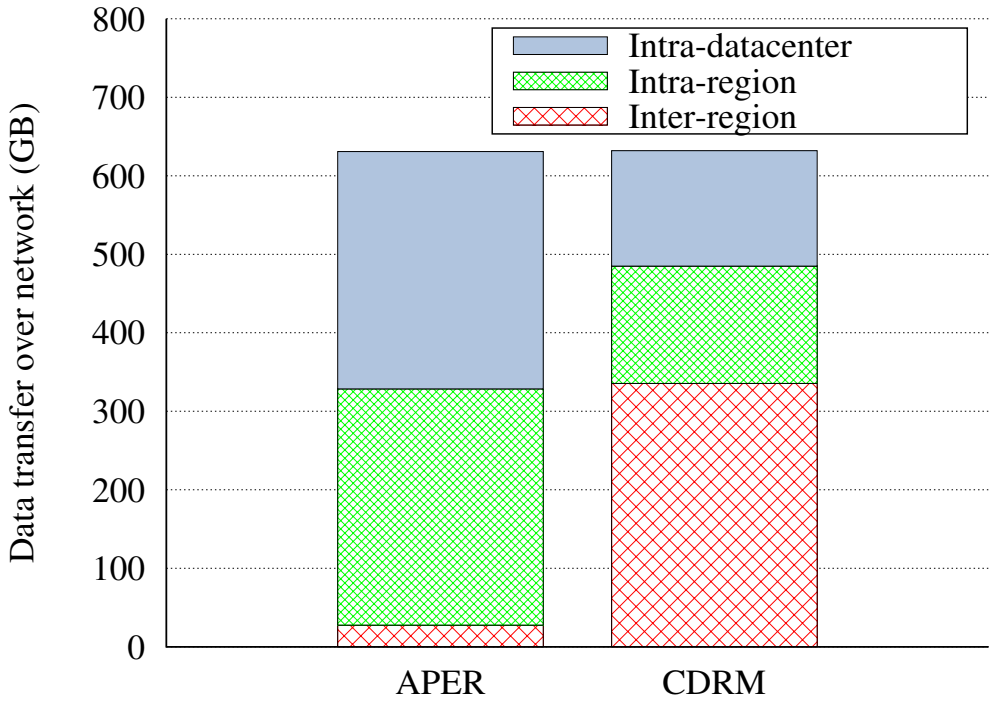


Figure 4.3: Consumption of bandwidth with respect to network hierarchy.

More frequent utilization of cheaper network links is also decreases expenditures, as APER avoids using expensive and slow inter-region links by placing replicas on servers that are cheaply accessed with good bandwidth. CDRM does not take into account the network hierarchy among servers during data placement. Instead, it performs server selection for placing new replicas according to the workload of candidate servers. As a result, there are many cases where a target server with a suitable properties for replica placement to be found in a remote datacenter or region. A drawback CDRM is high utilization of expensive and slower inter-region network links.

4.3.6 Number of SLA Violations

The number of SLA violations is the ultimate verdict of replication decisions on their effectiveness during the simulation period. If a strategy performs data replication in accordance with satisfying the SLA, the number of breaches should ideally be zero. However, initially the fragments are scattered across the cloud topology. Inevitably, during a small period at the beginning of the simulations, some violations are observed simply due to replica reconfiguration to respond query arrival rate. Therefore, a more realistic view of the number of breaches is to keep them at a minimum after the initial replica reconfiguration.

APER caused significantly less number of SLA breaches compared to CDRM during the simulation. As can be seen in the discussion of the average response time metric, some SLA violations (Table 4.2) occurred with the proposed strategy at the initial replica configuration phase. However, CDRM caused a significant number of SLA breaches as a result of the high average response time. Minimizing the number of SLA breaches is the most important measure to take in order to avoid

high penalty expenditures for the provider.

4.3.7 Monetary Expenditures of the Provider

During the simulations, cloud resources are consumed, or rather occupied for processing the computational load generated by tenant queries.

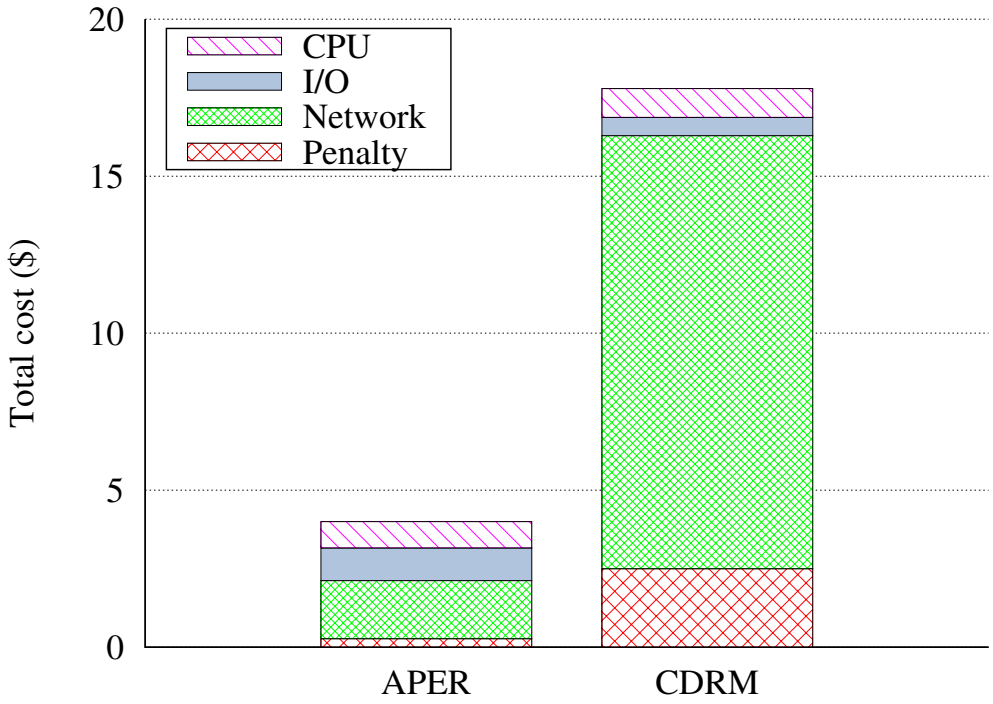


Figure 4.4: Total costs of the provider during the simulation.

The expenditures are calculated with respect to the resource consumption and their unit costs as described in the simulation parameters. Since penalty cost is not an expenditure generated by the consumption of a resource, it is calculated by using the number of SLA breaches during the simulation. Figure 4.4 shows the monetary costs of the cloud resources corresponding to their consumptions during the simulation. Apart from penalty and network costs, both strategies yielded similar resource

costs. Lower-than-threshold average response time and minimal SLA violations enabled APER to cause less expenditures for the provider in terms of penalties paid to the tenants. Furthermore, having enough replicas in strategic places allowed APER to accumulate cost savings by avoiding high utilization of the expensive network links.

4.4 Conclusion

There are several takeaways emerged from the results of the simulation study. While a data replication strategy is essential to satisfy performance guarantees, simply using a traditional strategy is not enough for the cloud provider to ensure economic benefit. Traditional strategies tend to eagerly replicate to attain best possible performance. However, in pursuit of the best performance, traditional strategies increase provider costs. Consequently, as long as the performance objective is satisfied, it is important to focus on improving the economic benefit of the provider.

Focusing on one aspect of the cloud may also result in some undesirable outcome for other aspects. An example for this phenomenon is observed in the simulations. While CDRM focuses on satisfying availability and load balancing, economic impact of the replication is not taken into account. The disregard of economic impact of replication on provider profitability resulted in an unacceptable outcome for the provider.

Furthermore, it can be said that, balanced load does not always necessarily satisfy the performance objective. APER successfully satisfied both the response time guarantee and the provider profitability by establishing a good trade-off between them. APER does not pursue best performance in a wasteful way that would ex-

haust the cloud resources. Instead, it aims to keep the performance at an acceptable threshold level to leave a healthy profit margin for the provider. In light of the performance evaluation, we believe that, simultaneous satisfaction of SLA and economic benefit should be the key ingredient in any data replication strategy operating in the cloud.

Chapter 5

Conclusion and Future Work

Abstract

This chapter concludes the thesis manuscript with a summary of the studies conducted during the research period and contributions made. The novel aspects of the proposed data replication strategy is revisited with discussing on its benefits and shortcomings. Furthermore, some future directions that may inspire researchers who are interested in doing research in this area are discussed.

Contents

5.1 Summary of Studies **128**

5.2 Future Directions **130**

5.1 Summary of Studies

Cloud computing is a computing paradigm that has been growing into popularity in more than a decade. Cloud providers rent the abstracted resources to the tenants in a economy-based model, which allows the tenants to be billed just for their share of resource usage. Moreover, elastic scaling of the resources makes it possible to quickly scale the cloud services to respond demand changes without interruption. In return for their money, tenants expect a certain level of performance from the providers. On the other hand, providers aim to minimize their expenditures to maximize their profit.

Data replication has been around for many decades to assist in achieving goals such as increased performance, improved availability and introducing fault-tolerance. Data replication deals with identification of what data needs replication, when to perform the task of replication, decide on the degree on replication, where to place the created replicas and finally retirement of the unnecessary replicas from the system. Of course, while pursuing for performance benefit through data replication, it is important to do it in a cost-effective way especially in economy-based systems such as clouds.

In this thesis we presented APER, a dynamic data replication strategy to satisfy both the response time of database queries and provider's economic benefit in the cloud. APER estimates the response time of database queries before the execution takes place and predicts whether the response time objective is going to be satisfied. If a query is estimated to violate the SLA due to the potential of producing a response time greater than the promised threshold to the tenant, the proposed strategy considers creating new replicas to resolve the performance problem. However, the query still needs to be estimated as profitable for the provider before the

replication is triggered. Therefore, both the performance objective and provider profitability criterion must be simultaneously satisfied in order to replicate data.

When data replication takes place, only the fragments that are expected to cause a bottleneck are selected for replication. There fragments are incrementally replicated to more suitable locations in the cloud. The placement of replicas is based on a placement heuristic, which finds candidate servers that are able to satisfy performance objective in a cost-effective manner. All of these replication decisions are made in such a way that reduces the resource consumption in the cloud to reduce the expenditures of the provider.

Regarding the cost model and economic model used in the replication decisions, former is responsible for estimating the response time of database queries. Response time estimation model in APER considers the consumption of the cloud resources in left-deep, right-deep and bushy join query plans. Latter model is focused on the profit estimation for the provider and it is based on the estimation of both the revenues and expenditures of the provider when executing a tenant query. Based on predicted resource consumptions, APER estimates the monetary cost of executing each query while expected revenue is calculated from the rent collected from the tenants.

We analyzed the performance of APER alongside CDRM (Wei et al., 2010), which is another data replication strategy proposed for cloud systems. The performance evaluation processed a large number of queries on a simulated cloud topology. The query load consist of different number of joins for each query and cloud topology is established to reflect a heterogeneous cloud environment to represent a realistic scenario. Results indicate that, APER satisfied performance and returned a profit for the provider by strategically placing replicas to simultaneously achieve improved

data access time and reduced resource consumption. On the other hand, CDRM resulted in high network and penalty costs due to not considering network hierarchy in data placement and high number of SLA breaches.

5.2 Future Directions

After having done in-depth research on data replication in large-scale data management systems, some new research directions or bifurcation possibilities from the research presented in this thesis have emerged. Below, we discuss some of those future work in our research area.

(i) Proposing a more efficient penalty management. On satisfaction of SLA, the ultimate measure of to what extent the tenant's expectations are satisfied by the provider is the penalty mechanism. If the provider manages to keep tenants happy by satisfying SLA, no penalty is incurred on the provider. Otherwise, some amount of penalty is paid by the provider to the tenant, with the amount depending on the degree of SLA violation. In APER, we ensure the satisfaction of SLA by estimating the response time of the the queries and make them meet threshold performance through data replication. Another possible way of achieving this result is by estimating the penalty amount due to execution of some query. This way, SLA satisfaction would still be satisfied and penalty management would become an objective of the data replication strategy, instead of a consequence. Of course, penalty minimization should also be done in a cost-effective way for the profitability of the provider.

(ii) APER makes the replication decisions before the execution of every single query that arrives at the cloud. One significant advantage of this approach is

to immediately respond to any changing trend in query load and popularity. However, one could argue that experiencing the overhead of data replication at every single query may be a high price to pay for the luxury of quickly responding to change. The alternative is to perform a periodical replication, instead of considering it at every query. In periodical replication, historical data for which queries are popular and which fragments are accessed more frequently is collected. Afterwards, this information can be used at periodical intervals to perform a global optimization for reconfiguring replicas. This approach may respond to changing trends more slowly than APER, but it can also be cheaper in terms of overhead. A good research opportunity is available here to compare both approaches and study which one is more suitable for what particular scenario.

(iii) Current economic model deals with the issue of multi-tenancy from the perspective of a single tenant. With a simplistic assumption of the provider having the same set of performance guarantees for each of its tenants, APER is capable of handling a multi-tenant cloud environment. However, each tenant having its own set of requirements from the provider complicates the matters. In this case, an extension to the economic model to take into account the heterogeneity among service level agreements with multiple tenants is necessary. This direction is worth pursuing and can open new research possibilities. Considering that queries that belong to a multitude of tenants are processed by the same set of resources, fairness of the provider towards each tenant with different levels of performance guarantees and other service level objectives would also pose interesting challenges to tackle.

(iv) Some other future research opportunities can also be pursued to evaluate the performance of the proposed strategy in various other simulation scenarios. A few example issues that may be worth researching on this matter are listed below.

(a) Performance evaluation study can be taken afar by creating some variation in certain simulation parameters. An way to accomplish this would be to impose different response time threshold values to be used for triggering data replication. This can also be conjoined with the multi-tenancy point made above, with some tenants requiring stricter response time guarantees while others may be more relaxed with their performance demand from the provider.

(b) Some variation can also be introduced in the workload generation routines used in the simulation. Currently, query load is randomly generated in terms of computational intensiveness and their arrival intervals at the cloud system. A more in-depth performance analysis can be done by extending the workload generation to invoke some intentional workload spikes to simulate a query load that resembles real-world system loads more closely.

(c) In the current simulation setup, the query load is generated with a randomly determined number of joins in order to create a load variation between each query. Another possible way to achieve this would be using a standard, well known batch of queries, more specifically TPC-H queries (Barata et al., 2015). Also it would be interesting to see the impact of query complexity (in terms of number of joins) on the response time objective. One example evaluation may consider modeling simple queries

with 1 join, medium queries with 3 joins and complex queries with 7 joins.

(d) Finally, the simulation topology can be expanded to have a larger number of VMs to better represent a realistic cloud environment. However, it should be noted that it would require a powerful hardware to accomplish as CloudSim gets increasingly resource-hungry as the number of simulated cloud topology gets larger.

Bibliography

Azizol Abdullah, Mohamed Othman, Hamidah Ibrahim, Md Nasir Sulaiman, and Abu Talib Othman. Decentralized replication strategies for P2P based scientific data grid. In *International Symposium on Information Technology (ITSim 2008)*, volume 3, pages 1–8. IEEE, 2008. ISBN 9781424423286.

Abdul Rahman Abdurrab and Tao Xie. FIRE: A file reunion based data replication strategy for data grids. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 215–223. IEEE, 2010. ISBN 978-1-4244-6987-1. doi: 10.1109/CCGRID.2010.12.

Abdallah Abouzamazem and Paul Ezhilchelvan. Efficient inter-cloud replication for high-availability services. In *2013 IEEE International Conference on Cloud Engineering (IC2E)*, pages 132–139. IEEE, mar 2013. ISBN 978-0-7695-4945-3. doi: 10.1109/IC2E.2013.27.

Lada A. Adamic and Bernardo A. Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.

Arif Ahmed and Abadhan Saumya Sabyasachi. Cloud computing simulators: A detailed survey and future direction. In *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, pages 866–872, 2014. ISBN 978-1-4799-2572-8. doi: 10.1109/IAdCC.2014.6779436.

Bahareh Alami Milani and Nima Jafari Navimipour. A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications*, 64:229–238, 2016. ISSN 10958592. doi: 10.1016/j.jnca.2016.02.005.

Tehmina Amjad, Muhammad Sher, and Ali Daud. A survey of dynamic replication strategies for improving data availability in data grids. *Future Generation Computer Systems*, 28(2):337–349, feb 2012. ISSN 0167739X. doi: 10.1016/j.future.2011.06.009.

Vassiliki Andronikou, Konstantinos Mamouras, Konstantinos Tserpes, Dimosthenis Kyriazis, and Theodora Varvarigou. Dynamic QoS-aware data replication in grid environments based on data importance. *Future Generation Computer Systems*, 28(3):544–553, mar 2012. ISSN 0167739X. doi: 10.1016/j.future.2011.02.003.

Martin Arlitt, Ludmila Cherkasova, John Dilley, Rich Friedrich, and Tai Jin. Evaluating content management techniques for web proxy caches. *ACM SIGMETRICS Performance Evaluation Review*, 27(4):3–11, mar 2000. ISSN 01635999. doi: 10.1145/346000.346003.

Xiaohu Bai, Hai Jin, Xiaofei Liao, Xuanhua Shi, and Zhiyuan Shao. RTRM: A response time-based replica management strategy for cloud storage system. In *Grid and Pervasive Computing*, number 1, pages 124–133. 2013.

Melyssa Barata, Jorge Bernardino, and Pedro Furtado. *An Overview of Decision Support Benchmarks: TPC-DS, TPC-H and SSB*, pages 619–628. Springer International Publishing, Cham, 2015. ISBN 978-3-319-16486-1. doi: 10.1007/978-3-319-16486-1_61. URL `http://dx.doi.org/10.1007/978-3-319-16486-1_61`.

Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2nd edition, 2013. ISBN 9781627050098. doi: 10.2200/S00516ED2V01Y201306CAC024.

Mahsa Beigrezaei, Abolfazle Toroghi Haghighat, Mohamd Reza Meybodi, and Maryam Runiassy. PPRA: A new pre-fetching and prediction based replication algorithm in data grid. In *6th International Conference on Computer and Knowledge Engineering (ICCKE)*, number ICCKE, pages 257–262. IEEE, oct 2016. ISBN 978-1-5090-3586-1. doi: 10.1109/ICCKE.2016.7802149.

William H. Bell, David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Kurt Stockinger, and Floriano Zini. Evaluation of an economy-based file replication strategy for a data grid. In *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)*, pages 661–668. IEEE, 2003. ISBN 0-7695-1919-9. doi: 10.1109/CCGRID.2003.1199430.

Nicolas Bonvin, Thanasis G Papaioannou, and Karl Aberer. A self-organized , fault-tolerant and scalable replication scheme for cloud storage categories and subject descriptors. In *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, pages 205–216, 2010a. ISBN 9781450300360.

Nicolas Bonvin, Thanasis G. Papaioannou, and Karl Aberer. An economic approach for scalable and highly-available distributed applications. In *IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, pages 498–505, 2010b. ISBN 9780769541303. doi: 10.1109/CLOUD.2010.45.

Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y. Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, 18(1):385–402, 2015a. ISSN 15737543. doi: 10.1007/s10586-014-0404-x.

Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y Zomaya. Models for efficient data replication in cloud computing datacenters. In *2015 IEEE International Conference on Communications (ICC)*, volume 2015-Septe, pages 6056–6061. IEEE, jun 2015b. ISBN 978-1-4673-6432-4. doi: 10.1109/ICC.2015.7249287.

Mohammad Bsoul, Ahmad Al-Khasawneh, Emad Eddien Abdallah, and Yousef Kilani. Enhanced fast spread replication strategy for data grid. *Journal of Network and Computer Applications*, 34(2):575–580, mar 2011. ISSN 10848045. doi: 10.1016/j.jnca.2010.12.006.

Rajkumar Buyya, Rajkumar Buyya, Chee Shin Yeo, Chee Shin Yeo, Srikumar Venugopal, Srikumar Venugopal, James Broberg, James Broberg, Ivona Brandic, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(June 2009):17, 2009. ISSN 0167-739. doi: 10.1016/j.future.2008.12.001.

Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, jan 2011. ISSN 00380644. doi: 10.1002/spe.995.

Israel Casas, Javid Taheri, Rajiv Ranjan, Lizhe Wang, and Albert Y. Zomaya. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. *Future Generation Computer Systems*, 2015. ISSN 0167739X. doi: 10.1016/j.future.2015.12.005.

Zakia Challal and Thouraya Bouabana-Tebibel. A priori replica placement strategy in data grid. In *International Conference on Machine and Web Intelligence*, pages 402–406. IEEE, oct 2010. ISBN 978-1-4244-8608-3. doi: 10.1109/ICMWI.2010.5647925.

Ruay Shiung Chang and Hui Ping Chang. A dynamic data replication strategy using access-weights in data grids. *The Journal of Supercomputing*, 45(3):277–295, jan 2008. ISSN 0920-8542. doi: 10.1007/s11227-008-0172-6.

Danwei Chen, Shutao Zhou, Xunyi Ren, and Qiang Kong. Method for replica creation in data grids based on complex networks. *The Journal of China Universities of Posts and Telecommunications*, 17(4):110–115, aug 2010. ISSN 10058885. doi: 10.1016/S1005-8885(09)60496-9.

Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23(3):187–200, jul 2000. ISSN 10848045. doi: 10.1006/jnca.2000.0110.

Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Bob Schwartzkopf, Heinz Stockinger, Kurt Stockinger, and Brian Tierney. Giggle: A framework for constructing scalable replica location services. In *Proceedings of the ACM/IEEE conference on Supercomputing*, number 3, pages 1–17, 2002. ISBN 076951524X.

Ann Chervenak, Ewa Deelman, Carl Kesselman, Bill Allcock, Ian Foster, Veronika Nefedova, Jason Lee, Alex Sim, Arie Shoshani, Bob Drach, Dean Williams, and Don Middleton. High-performance remote access to climate simulation data: a challenge problem for data grid technologies. *Parallel Computing*, 29(10):1335–1356, oct 2003. ISSN 01678191. doi: 10.1016/j.parco.2003.06.001.

Ann Chervenak, Robert Schuler, Carl Kesselman, and Scott Koranda. Wide area data replication for scientific collaborations. *International Journal of High Performance Computing and Networking*, 5(3):124–134, 2008.

Hanene Chettaoui and Faouzi Ben Charrada. A new decentralized periodic replication strategy for dynamic data grids. *Scalable Computing: Practice and Experience*, 15(1):101–119, 2014. ISSN 1895-1767. doi: 10.12694/scpe.v15i1.969.

Uros Cibej, Bostjan Slivnik, and Borut Robic. The complexity of static data replication in data grids. *Parallel Computing*, 31(8-9):900–912, aug 2005. ISSN 01678191. doi: 10.1016/j.parco.2005.04.010.

Zhongqiang Cui, Decheng Zuo, and Zhan Zhang. Based on the correlation of the file dynamic replication strategy in multi-tier data grid. *International Journal of Database Theory and Application*, 8(1):75–86, feb 2015. ISSN 20054270. doi: 10.14257/ijdta.2015.8.1.09.

Ewa Deelman, Carl Kesselman, Gaurang Mehta, Leila Meshkat, Laura Pearlman, Kent Blackburn, Phil Ehrens, Albert Lazzarini, Roy Williams, and Scott Koranda. GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists. In *Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing (HPDC'02)*, pages 225–234, 2002.

Atakan Dogan. A study on performance of dynamic file replication algorithms for real-time file access in data grids. *Future Generation Computer Systems*, 25(8): 829–839, sep 2009. ISSN 0167739X. doi: 10.1016/j.future.2009.02.002.

Zhihui Du, Jingkun Hu, Yinong Chen, Zhili Cheng, and Xiaoying Wang. Optimized qos-aware replica placement heuristics and applications in astronomy data grid. *Journal of Systems and Software*, 84(7):1224–1232, jul 2011. ISSN 01641212. doi: 10.1016/j.jss.2011.02.038.

Ian Foster. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, aug 2001. ISSN 1094-3420. doi: 10.1177/109434200101500302.

Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE, nov 2008. ISBN 978-1-4244-2860-1. doi: 10.1109/GCE.2008.4738445.

Xiong Fu, Xin xin Zhu, Jing yu Han, and Ru chuan Wang. QoS-aware replica placement for data intensive applications. *The Journal of China Universities of Posts and Telecommunications*, 20(3):43–47, jun 2013. ISSN 10058885. doi: 10.1016/S1005-8885(13)60047-3.

Minos N. Garofalakis and Yannis E. Ioannidis. Multi-dimensional resource scheduling for parallel queries. *SIGMOD Rec.*, 25(2):365–376, 1996. ISSN 0163-5808. doi: 10.1145/233269.233352.

Hamoun Ghanbari, Bradley Simmons, Marin Litoiu, and Gabriel Iszlai. Feedback-based optimization of a private cloud. *Future Generation Computer Systems*, 28 (1):104–111, jan 2012. ISSN 0167739X. doi: 10.1016/j.future.2011.05.019.

Navneet Kaur Gill and Sarbjeet Singh. A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers. *Future Generation Computer Systems*, 65:10–32, dec 2016. ISSN 0167739X. doi: 10.1016/j.future.2016.05.016.

Sushant Goel and Rajkumar Buyya. Data replication strategies in wide area distributed systems. In *Enterprise Service Computing: From Concept to Deployment*, page 17. 2006.

Albert Greenberg, James Hamilton, David a Maltz, and Parveen Patel. The cost of a cloud: Research problems in data center networks. *Computer Communication Review*, 39(1):68–73, 2009. ISSN 01464833. doi: 10.1145/1496091.1496103.

Gabriel Guerrero-Contreras, Carlos Rodriguez-Dominguez, Sara Balderas-Diaz, and JoseLuis Garrido. Dynamic replication and deployment of services in mobile environments. In *New Contributions in Information Systems and Technologies*, pages 855–864. Springer International Publishing, 2015.

Abdelkader Hameurlain and Riad Mokadem. Elastic data management in cloud systems. *International Journal of Computer Systems Science and Engineering*, 32 (4), jul 2017.

Abdelkader Hameurlain and Franck Morvan. *Big Data Management in the Cloud: Evolution or Crossroad?*, pages 23–38. Springer International Publishing, 2016. ISBN 978-3-319-34099-9. doi: 10.1007/978-3-319-34099-9_2.

Tarek Hamrouni, Sarra Slimani, and Faouzi Ben Charrada. A data mining correlated patterns-based periodic decentralized replication strategy for data grids. *Journal of Systems and Software*, 110:10–27, 2015. ISSN 01641212. doi: 10.1016/j.jss.2015.08.019.

Banchong Harangsri. *Query Result Size Estimation Techniques in Database Systems*. PhD thesis, 1998.

Abbas Horri, Reza Sepahvand, and Gholam Hosein Dastghaibyfard. A hierarchical scheduling and replication strategy. *International Journal of Computer Science and Network Security*, 8(8):30–35, 2008.

Wolfgang Hoschek, Javier Jaen-martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. Data management in an international data grid project. In *IEEE, ACM International Workshop on Grid Computing*, pages 77–90, 2000.

Hui-I Hsiao, Ming-Syan Chen, and Philip S. Yu. On parallel execution of multiple pipelined hash joins. *ACM SIGMOD Record*, 23(2):185–196, jun 1994. ISSN 01635808. doi: 10.1145/191843.191879.

Joolahluk Janpet and Yean-Fu Wen. Reliable and available data replication planning for cloud storage. In *IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 772–779. IEEE, mar 2013. ISBN 978-1-4673-5550-6. doi: 10.1109/AINA.2013.125.

D S Jayalakshmi and Rashmi Ranjana T P. Dynamic data replication strategy in cloud environments. In *2015 Fifth International Conference on Advances in Computing and Communications (ICACC)*, pages 102–105, 2015. ISBN 9781467369947. doi: 10.1109/ICACC.2015.79.

Bettina Kemme, Ricardo Jimenez-Peris, and Marta Patino-Martinez. *Database Replication*. Morgan and Claypool Publishers, 2010.

Leyli Mohammad Khanli, Ayaz Isazadeh, and Tahmuras N. Shishavan. PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid. *Future Generation Computer Systems*, 27(3):233–244, mar 2011. ISSN 0167739X. doi: 10.1016/j.future.2010.08.013.

Yousri Kouki and Thomas Ledoux. SCAling: SLA-driven cloud auto-scaling. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 411–412, 2013. ISBN 9781450316569.

Yousri Kouki, Thomas Ledoux, and Remi Sharrock. Cross-layer SLA selection for cloud services. In *First International Symposium on Network Cloud Computing and Applications*, pages 143–147. IEEE, nov 2011. ISBN 978-1-4577-1667-6. doi: 10.1109/NCCA.2011.30.

K. Ashwin Kumar, Abdul Quamar, Amol Deshpande, and Samir Khuller. SWORD: workload-aware data placement and replica selection for cloud data management systems. *The VLDB Journal*, 23(6):845–870, dec 2014. ISSN 1066-8888. doi: 10.1007/s00778-014-0362-1.

Houda Lamehamedi, Boleslaw Szymanski, Zujun Shentu, and Ewa Deelman. Data replication strategies in grid environments. In *Proceedings of Fifth International Conference on Algorithms and Architectures for Parallel Processing.*, pages 378–383. IEEE Comput. Soc, 2002. ISBN 0-7695-1512-6. doi: 10.1109/ICAPP.2002.1173605.

Willis Lang, Srinath Shankar, Jignesh M Patel, and Ajay Kalhan. Towards multi-tenant performance SLOs. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1447–1463, 2014.

Rosana S G Lanzelotte, Patrick Valduriez, Mohamed Zaït, and Mikal Ziane. Industrial-strength parallel query optimization: Issues and lessons. *Information Systems*, 19(4):311–330, 1994. ISSN 03064379. doi: 10.1016/0306-4379(94)90017-5.

Jungha Lee, Jaehwa Chung, and Daewon Lee. Efficient data replication scheme based on hadoop distributed file system. *International Journal of Software Engineering and Its Applications*, 9(12):177–186, 2015.

Ming Chang Lee, Fang Yie Leu, and Ying Ping Chen. PFRF: An adaptive data replication algorithm based on star-topology data grids. *Future Generation Computer Systems*, 28(7):1045–1057, jul 2012. ISSN 0167739X. doi: 10.1016/j.future.2011.08.015.

Ming Lei, Susan V. Vrbsky, and Xiaoyan Hong. An on-line replication strategy to increase availability in data grids. *Future Generation Computer Systems*, 24(2): 85–98, feb 2008. ISSN 0167739X. doi: 10.1016/j.future.2007.04.009.

Wenhao Li, Yun Yang, and Dong Yuan. A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 496–502. IEEE, dec 2011. ISBN 978-1-4673-0006-3. doi: 10.1109/DASC.2011.95.

Wenhao Li, Yun Yang, and Dong Yuan. Ensuring cloud data reliability with minimum replication by proactive replica checking. *IEEE Transactions on Computers*, 65(5):1494–1506, 2016. ISSN 00189340. doi: 10.1109/TC.2015.2451644.

Sai-Qin Long, Yue-Long Zhao, and Wei Chen. Morm: A multi-objective optimized replication management strategy for cloud storage cluster. *Journal of Systems Architecture*, 60(2):234–244, feb 2014. ISSN 13837621. doi: 10.1016/j.sysarc.2013.11.012.

Thanasis Loukopoulos and Ishfaq Ahmad. Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, 64(11):1270–1285, nov 2004. ISSN 07437315. doi: 10.1016/j.jpdc.2004.04.005.

Jianwei Ma, Wanyu Liu, and Tristan Glatard. A classification of file placement and replication methods on grids. *Future Generation Computer Systems*, 29(6): 1395–1406, aug 2013. ISSN 0167739X. doi: 10.1016/j.future.2013.02.006.

Nitesh Maheshwari, Radheshyam Nanduri, and Vasudeva Varma. Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce

framework. *Future Generation Computer Systems*, 28(1):119–127, jan 2012. ISSN 0167739X. doi: 10.1016/j.future.2011.07.001.

Natalia Maltsev, Elizabeth Glass, Dinanath Sulakhe, Alexis Rodriguez, Mustafa H Syed, Tanuja Bompada, Yi Zhang, and Mark D'Souza. Puma2–grid-based high-throughput analysis of genomes and metabolic pathways. *Nucleic acids research*, 34(Database issue):D369–D372, jan 2006. ISSN 1362-4962. doi: 10.1093/nar/gkj095.

Najme Mansouri and Gholam Hosein Dastghaibyfard. A dynamic replica management strategy in data grid. *Journal of Network and Computer Applications*, 35 (4):1297–1303, jul 2012. ISSN 10848045. doi: 10.1016/j.jnca.2012.01.014.

Najme Mansouri and Gholam Hosein Dastghaibyfard. Enhanced dynamic hierarchical replication and weighted scheduling strategy in data grid. *Journal of Parallel and Distributed Computing*, 73(4):534–543, apr 2013. ISSN 07437315. doi: 10.1016/j.jpdc.2013.01.002.

Najme Mansouri, Gholam Hosein Dastghaibyfard, and Ehsan Mansouri. Combination of data replication and scheduling algorithm for improving data availability in data grids. *Journal of Network and Computer Applications*, 36(2):711–722, mar 2013. ISSN 10848045. doi: 10.1016/j.jnca.2012.12.021.

Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. Cost optimization for dynamic replication and migration of data in cloud data centers. *IEEE Transactions on Cloud Computing*, 7161((in press)), 2017. ISSN 2168-7161. doi: 10.1109/TCC.2017.2659728.

Huang Mengxing, Ye Xianglong, Wei Sanpeng, and Zhu Donghai. A strategy of dynamic replica creation in cloud storage. In *Proceedings of the 1st International Workshop on Cloud Computing and Information Security*, number Ccis, pages 389–392, Paris, France, 2013. Atlantis Press. ISBN 978-90-78677-88-8. doi: 10.2991/ccis-13.2013.89.

Bakhta Meroufel and Ghalem Belalem. Managing data replication and placement based on availability. *AASRI Procedia*, 5:147–155, jan 2013. ISSN 22126716. doi: 10.1016/j.aasri.2013.10.071.

Marco Miglierina, Giovanni P. Gibilisco, Danilo Ardagna, and Elisabetta Di Nitto. Model based control for multi-cloud applications. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 37–43, 2013. ISBN 9781467364478. doi: 10.1109/MiSE.2013.6595294.

Riad Mokadem and Abdelkader Hameurlain. Data replication strategies with performance objective in data grid systems: a survey. *International Journal of Grid and Utility Computing*, 6(1):30–46, 2015.

Julia Myint and Thinn Thu Naing. Management of data replication for pc cluster based cloud storage system. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 1(3):31–41, 2011. doi: 10.5121/ijccsa.2011.1303.

Caitriana Nicholson, David G. Cameron, A. T. Doyle, A. Paul Millar, and Kurt Stockinger. Dynamic data replication in LCG 2008. *Concurrency and Computation: Practice and Experience*, 20(11):1259–1271, 2008. doi: 10.1002/cpe.

M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Springer New York, New York, NY, 3rd edition, 2011. ISBN 978-1-4419-8833-1. doi: 10.1007/978-1-4419-8834-8.

Lluis Pamies-Juarez, Pedro García-López, Marc Sánchez-Artigas, and Blas Herrera. Towards the design of optimal data redundancy schemes for heterogeneous cloud storage infrastructures. *Computer Networks*, 55(5):1100–1113, apr 2011. ISSN 13891286. doi: 10.1016/j.comnet.2010.11.004.

Sang Min Park, Jai Hoon Kim, Young Bae Ko, and Won Sik Yoon. Dynamic data grid replication strategy based on internet hierarchy. In *Grid and Cooperative Computing*, pages 838–846. Springer Berlin Heidelberg, 2004.

José M. Pérez, Félix García-Carballeira, Jesús Carretero, Alejandro Calderón, and Javier Fernández. Branch replication scheme: A new model for data replication in large scale data grids. *Future Generation Computer Systems*, 26(1):12–20, jan 2010. ISSN 0167739X. doi: 10.1016/j.future.2009.05.015.

C.L. Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275: 314–347, aug 2014. ISSN 00200255. doi: 10.1016/j.ins.2014.01.015.

Danyang Qin, Ruixue Liu, Jiaqi Zhen, Songxiang Yang, and Erfu Wang. Research on decentralized group replication strategy based on correlated patterns mining in data grids. In *Machine Learning and Intelligent Communications: First International Conference, MLICOM 2016, Shanghai, China, August 27-28, 2016, Revised Selected Papers*, pages 293–302. Springer International Publishing, 2017. doi: 10.1007/978-3-319-52730-7_30.

Yanzhen Qu and Naixue Xiong. RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. In *2012 41st International Conference on Parallel Processing*, pages 520–529. IEEE, sep 2012. ISBN 978-1-4673-2508-0. doi: 10.1109/ICPP.2012.3.

RM Rahman, Ken Barker, and Reda Alhajj. Replica placement in data grid: a multi-objective approach. In *Grid and Cooperative Computing-GCC 2005*, pages 645–656. Springer Berlin Heidelberg, 2005.

Kavitha Ranganathan and Ian Foster. Identifying dynamic replication strategies for a high-performance data grid. In *In Proc. of the International Grid Computing Workshop*, volume 2242, pages 75–86. Springer, 2001.

Kavitha Ranganathan, Adriana Iamnitchi, and Ian Foster. Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pages 376–381, 2002. ISBN 0769515827.

Qaisar Rasool, Jianzhong Li, and Shuo Zhang. Replica placement in multi-tier data grid. In *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 103–108. IEEE, dec 2009. ISBN 978-1-4244-5420-4. doi: 10.1109/DASC.2009.60.

Ismaeel Al Ridhawi, Nour Mostafa, and Wassim Masri. Location-aware data replication in cloud computing systems. *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 20–27, 2015. doi: 10.1109/WiMOB.2015.7347936.

Nazanin Saadat and Amir Masoud Rahmani. PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids. *Future Generation Computer*

Systems, 28(4):666–681, apr 2012. ISSN 0167739X. doi: 10.1016/j.future.2011.10.011.

Sherif Sakr and Anna Liu. SLA-based and consumer-centric dynamic provisioning for cloud databases. In *IEEE Fifth International Conference on Cloud Computing*, pages 360–367. IEEE, jun 2012. ISBN 978-1-4673-2892-0. doi: 10.1109/CLOUD.2012.11.

Sherif Sakr, Liang Zhao, Hiroshi Wada, and Anna Liu. CloudDB AutoAdmin: Towards a truly elastic cloud-based data store. In *IEEE International Conference on Web Services*, pages 732–733. IEEE, jul 2011. ISBN 978-1-4577-0842-8. doi: 10.1109/ICWS.2011.19.

K. Sashi and Antony Selvadoss Thanamani. Dynamic replication in a data grid using a modified bhr region based algorithm. *Future Generation Computer Systems*, 27 (2):202–210, feb 2011. ISSN 0167739X. doi: 10.1016/j.future.2010.08.011.

Ben Segal. Grid computing: The european data grid project. In *IEEE Nuclear Science Symposium and Medical Imaging Conference*, volume 1, pages 15–20, 2000.

Artyom Sharov, Alexander Shraer, Arif Merchant, and Murray Stokely. Take me to your leader! *Proceedings of the VLDB Endowment*, 8(12):1490–1501, aug 2015. ISSN 21508097. doi: 10.14778/2824032.2824047.

Mohammad Shorfuzzaman, Peter Graham, and Rasit Eskicioglu. Adaptive popularity-driven replica placement in hierarchical data grids. *The Journal of Supercomputing*, 51(3):374–392, dec 2009. ISSN 0920-8542. doi: 10.1007/s11227-009-0371-9.

Guthemberg Silvestre, Sebastien Monnet, Ruby Krishnaswamy, and Pierre Sens. AREN: A popularity aware replication scheme for cloud storage. In *IEEE 18th International Conference on Parallel and Distributed Systems*, pages 189–196. IEEE, dec 2012. ISBN 978-1-4673-4565-1. doi: 10.1109/ICPADS.2012.35.

Flávio R C Sousa and Javam C. Machado. Towards elastic multi-tenant database replication with quality of service. In *IEEE/ACM 5th International Conference on Utility and Cloud Computing (UCC 2012)*, pages 168–175, 2012. ISBN 9780769548623. doi: 10.1109/UCC.2012.36.

Evjola Spaho, Admir Barolli, Fatos Xhafa, and Leonard Barolli. P2P data replication: Techniques and applications. In *Modeling and Processing for Next-Generation Big-Data Technologies*, pages 145–166. Springer International Publishing, 2015.

Katerina Stamou, Verena Kantere, and Jean-Henry Morin. SLA data management criteria. In *IEEE International Conference on Big Data*, pages 34–42. IEEE, oct 2013. ISBN 978-1-4799-1293-3. doi: 10.1109/BigData.2013.6691769.

Vladimir Stantchev and Christian Schröpfer. Negotiating and enforcing QoS and SLAs in grid and cloud computing. In *Advances in grid and pervasive computing*, pages 25–35. 2009.

Maarten Van Steen and Guillaume Pierre. Replicating for performance: case studies. In *Replication*, pages 73–89. 2010.

Basem Suleiman, Sherif Sakr, Ross Jeffery, and Anna Liu. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *Journal of Internet Services and Applications*, 3(2):173–193, dec 2011. ISSN 1867-4828. doi: 10.1007/s13174-011-0050-y.

Da-Wei Sun, Gui-Ran Chang, Shang Gao, Li-Zhong Jin, and Xing-Wei Wang. Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. *Journal of Computer Science and Technology*, 27(2): 256–272, mar 2012a. ISSN 1000-9000. doi: 10.1007/s11390-012-1221-4.

Dawei Sun, Guiran Chang, Changsheng Miao, and Xingwei Wang. Building a high serviceability model by checkpointing and replication strategy in cloud computing environments. In *32nd International Conference on Distributed Computing Systems Workshops*, pages 578–587. IEEE, jun 2012b. ISBN 978-1-4673-1423-7. doi: 10.1109/ICDCSW.2012.6.

Arun Swami and K. Bernhard Schiefer. On the estimation of join result sizes. In *Advances in Database Technology - EDBT '94*, pages 287–300. 1994. ISBN 978-3-540-57818-5. doi: 10.1007/3-540-57818-8_58.

Khaoula Tabet, Riad Mokadem, Mohamed Ridda Laouar, and Sean Eom. Data replication in cloud systems: A survey. *International Journal of Information Systems in the Service Sector*, 8(3):17–33, jun 2017. doi: 10.4018/IJISSC.2017070102.

Atsuko Takefusa, Osamu Tatebe, Satoshi Matsuoka, and Youhei Morita. Performance analysis of scheduling and replication algorithms on grid datafarm architecture for high-energy physics applications. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, pages 34–43, 2003.

Ming Tang, Bu Sung Lee, Chai Kiat Yeo, and Xueyan Tang. Dynamic replication algorithms for the multi-tier data grid. *Future Generation Computer Systems*, 21 (5):775–790, may 2005. ISSN 0167739X. doi: 10.1016/j.future.2004.08.001.

Xueyan Tang and Jianliang Xu. QoS-aware replica placement for content distribution. In *IEEE Transactions on Parallel and Distributed Systems*, volume 16, pages 921–932, oct 2005. doi: 10.1109/TPDS.2005.126.

Osamu Tatebe, Youhei Morita, and Satoshi Matsuoka. Grid datafarm architecture for petascale data intensive computing. In *International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pages 102–110, 2002. ISBN 0769515827.

N. Tomov, E. Dempster, M.H. Williams, A. Burger, H. Taylor, P.J.B. King, and P. Broughton. Analytical response time estimation in parallel relational database systems. *Parallel Computing*, 30(2):249–283, feb 2004. ISSN 01678191. doi: 10.1016/j.parco.2003.11.003.

Neven Tomov, Euan Dempster, M Howard Williams, Albert Burger, Hamish Taylor, Peter J B King, and Phil Broughton. Some results from a new technique for response time estimation in parallel dbms. In *High-Performance Computing and Networking, Proceedings*, volume 1593, pages 713–721, 1999. ISBN 0302-9743.

Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Dynamic replication strategies in data grid systems: a survey. *The Journal of Supercomputing*, 71(11):4116–4140, 2015. ISSN 0920-8542. doi: 10.1007/s11227-015-1508-7.

Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. A performance and profit oriented data replication strategy for cloud systems. In *Intl IEEE Conference on Cloud and Big Data Computing (CBD-Com)*, pages 780–787. IEEE, jul 2016. ISBN 978-1-5090-2771-2. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0125.

Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Ensuring performance and provider profit through data replication in cloud systems. *Cluster Computing*, (under review), 2017a.

Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Achieving query performance in the cloud via a cost-effective data replication strategy. *International Journal of Web and Grid Services*, (under review), 2017b.

David Vengerov, Winterberry Way, San Jose, and Andre Cavalheiro Menck. Join size estimation subject to filter conditions. *VLDB*, pages 1530–1541, 2015. ISSN 21508097. doi: 10.14778/2824032.2824051.

Ashish Vulimiri, Carlo Curino, Brighten Godfrey, Thomas Jungblut, Jitu Padhye, and George Varghese. Global analytics in the face of bandwidth and regulatory constraints. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 323–336, 2015. ISBN 9781931971218.

Da Wang, Gauri Joshi, and Gregory Wornell. Efficient task replication for fast response times in parallel computation. *arXiv:1404.1328*, pages 1–20, 2014. ISSN 01635999. doi: 10.1145/2591971.2592042.

Qingsong Wei, Bharadwaj Veeravalli, Bozhao Gong, Lingfang Zeng, and Dan Feng. Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster. In *IEEE International Conference on Cluster Computing*, pages 188–196. IEEE, sep 2010. ISBN 978-1-4244-8373-0. doi: 10.1109/CLUSTER.2010.24.

Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. SLA-based resource allocation for software as a service provider (saas) in cloud computing environments. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 195–204. IEEE, may 2011. ISBN 978-1-4577-0129-0. doi: 10.1109/CCGrid.2011.51.

Wentao Wu, Yun Chi, Hakan Hacigümüs, and Jeffrey F Naughton. Towards predicting query execution time for concurrent and dynamic database workloads. *Proc. VLDB Endow.*, 6(10):925–936, 2013. ISSN 2150-8097. doi: 10.14778/2536206.2536219.

Pengcheng Xiong, Yun Chi, Shenghuo Zhu, Hyun Jin Moon, Calton Pu, and Hakan Hacigumus. Intelligent management of virtualized resources for database systems in cloud environment. In *International Conference on Data Engineering*, pages 87–98, 2011. ISBN 9781424489589. doi: 10.1109/ICDE.2011.5767928.

Lingfang Zeng, Shijie Xu, Yang Wang, Kenneth B Kent, David Bremner, and Chengzhong Xu. Toward cost-effective replica placements in cloud storage systems with QoS-awareness. *Software: Practice and Experience*, 2016. ISSN 00380644. doi: 10.1002/spe.2441.

Hong Zhang, Bing Lin, Zhanghui Liu, and Wenzhong Guo. Data replication placement strategy based on bidding mode for cloud storage cluster. In *11th Web Information System and Application Conference*, pages 207–212, 2014. ISBN 978-1-4799-5726-2. doi: 10.1109/WISA.2014.45.

Liang Zhao. *SLA-Driven Database Replication on Cloud Platforms*. PhD thesis, 2013.

Liang Zhao, Sherif Sakr, Anna Liu, and Athman Bouguettaya. SLA-driven database replication on virtualized database servers. In *Cloud Data Management*, pages 97–118. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-04765-2_7.

Wuqing Zhao, Xianbin Xu, Zhuowei Wang, Yuping Zhang, and Shuibing He. A dynamic optimal replication strategy in data grid environment. In *International Conference on Internet Technology and Applications*, pages 2–5, 2010. ISBN 9781424451432.

Data replication in large-scale data management systems

In recent years, growing popularity of large-scale applications, e.g. scientific experiments, Internet of things and social networking, led to generation of large volumes of data. The management of this data presents a significant challenge as the data is heterogeneous and distributed on a large scale.

In traditional systems including distributed and parallel systems, peer-to-peer systems and grid systems, meeting objectives such as achieving acceptable performance while ensuring good availability of data are major challenges for service providers, especially when the data is distributed around the world. In this context, data replication, as a well-known technique, allows: (i) increased data availability, (ii) reduced data access costs, and (iii) improved fault-tolerance. However, replicating data on all nodes is an unrealistic solution as it generates significant bandwidth consumption in addition to exhausting limited storage space. Defining good replication strategies is a solution to these problems.

The data replication strategies that have been proposed for the traditional systems mentioned above are intended to improve performance for the user. They are difficult to adapt to cloud systems. Indeed, cloud providers aim to generate a profit in addition to meeting tenant requirements. Meeting the performance expectations of the tenants without sacrificing the provider's profit, as well as managing resource elasticities with a pay-as-you-go pricing model, are the fundamentals of cloud systems.

In this thesis, we propose a data replication strategy that satisfies the requirements of the tenant, such as performance, while guaranteeing the economic profit of the provider. Based on a cost model, we estimate the response time required to execute a distributed database query. Data replication is only considered if, for any query, the estimated response time exceeds a threshold previously set in the contract between the provider and the tenant. Then, the planned replication must also be economically beneficial to the provider. In this context, we propose an economic model that takes into account both the expenditures and the revenues of the provider during the execution of any particular database query. Once the data replication is decided to go through, a heuristic placement approach is used to find the placement for new replicas in order to reduce the access time. In addition, a dynamic adjustment of the number of replicas is adopted to allow elastic management of resources.

Proposed strategy is validated in an experimental evaluation carried out in a simulation environment. Compared with another data replication strategy proposed in the cloud systems, the analysis of the obtained results shows that the two compared strategies respond to the performance objective for the tenant. Nevertheless, a replica of data is created, with our strategy, only if this replication is profitable for the provider.

Keywords: Cloud Computing, Database Queries, Data Replication, Performance Evaluation, Economic Benefit

---

Réplication de données dans les systèmes de gestion de données à grande échelle

Ces dernières années, la popularité croissante des applications, e.g. les expériences scientifiques, Internet des objets et les réseaux sociaux, a conduit à la génération de gros volumes de données. La gestion de telles données qui de plus, sont hétérogenes et distribuées à grande échelle, constitue un défi important.

Dans les systèmes traditionnels tels que les systèmes distribués et parallèles, les systèmes pair-à-pair et les systèmes de grille, répondre à des objectifs tels que l'obtention de performances acceptables tout en garantissant une bonne disponibilité de données constituent des objectifs majeurs pour l'utilisateur, en particulier lorsque ces données sont réparties à travers le monde. Dans ce contexte, la réplication de données, une technique très connue, permet notamment: (i) d'augmenter la disponibilité de données, (ii) de réduire les coûts d'accès aux données et (iii) d'assurer une meilleure tolérance aux pannes. Néanmoins, répliquer les données sur tous les nœuds est une solution non réaliste vu qu'elle génère une consommation importante de la bande passante en plus de l'espace limité de stockage. Définir des stratégies de réplication constitue la solution à apporter à ces problématiques.

Les stratégies de réplication de données qui ont été proposées pour les systèmes traditionnels cités précédemment ont pour objectif l'amélioration des performances pour l'utilisateur. Elles sont difficiles à adapter dans les systèmes de cloud. En effet, le fournisseur de cloud a pour but de générer un profit en plus de répondre aux exigences des locataires. Satisfaire les attentes de ces locataire en matière de performances sans sacrifier le profit du fournisseur d'un coté et la gestion élastiques des ressources avec une tarification suivant le modèle 'pay-as-you-go' d'un autre coté, constituent des principes fondamentaux dans les systèmes cloud.

Dans cette thèse, nous proposons une stratégie de réplication de données pour satisfaire les exigences du locataire, e.g. les performances, tout en garantissant le profit économique du fournisseur. En se basant sur un modèle de coût, nous estimons le temps de réponse nécessaire pour l'exécution d'une requête distribuée. La réplication de données n'est envisagée que si le temps de réponse estimé dépasse un seuil fixé auparavant dans le contrat établi entre le fournisseur et le client. Ensuite, cette réplication doit être profitable du point de vue économique pour le fournisseur. Dans ce contexte, nous proposons un modèle économique prenant en compte aussi bien les dépenses et les revenus du fournisseur lors de l'exécution de cette requête. Nous proposons une heuristique pour le placement des répliques afin de réduire les temps d'accès à ces nouvelles répliques. De plus, un ajustement du nombre de répliques est adopté afin de permettre une gestion élastique des ressources.

Nous validons la stratégie proposée par une évaluation basée sur une simulation. Nous comparons les performances de notre stratégie à celles d'une autre stratégie de réplication proposée dans les clouds. L'analyse des résultats obtenus a montré que les deux stratégies comparées répondent à l'objectif de performances pour le locataire. Néanmoins, une réplique de données n'est crée, avec notre stratégie, que si cette réplication est profitable pour le fournisseur.

Mots-clés: Systèmes cloud, requêtes de base de données, réplication de données, évaluation de performances, profit économique