Algorithmes d'ordonnancement pour les nouveaux supports d'exécution

Pierre-François Dutot

Laboratoire ID-IMAG

18 October 2004

Scheduling algorithms for new execution platforms

Pierre-François Dutot

Laboratoire ID-IMAG

18 October 2004

Fact

Computing power will never outgrow users imagination.

Bigger computers allow :

- better weather forecast
- medical research (protein modeling)
- astro-physics simulation
- ...

Fact

Computing power will never outgrow users imagination.

Bigger computers allow :

- better weather forecast
- medical research (protein modeling)
- astro-physics simulation
- ...


TOP500 SUPERCOMPUTER SITES
Performance Development
100 PFlops
1 PFlops
100 TFlops
10 TFlops
1 TFlops
100 GFlops
10 GFlops
1 GFlops
100 MFlops
Performance
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
#1
#500
Sum
813.23 TF
35.86 TF
624 GF
1.12 TF
59.7 GF
0.42 GF
23rd List / June 2004
http://www.top500.org/

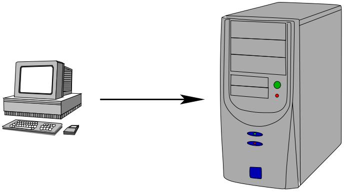
There are two options to increase the available computer power :

- Either buy a bigger computer,
- Or use several computers.

Question

We need to decide where and when to compute.

There are two options to increase the available computer power :

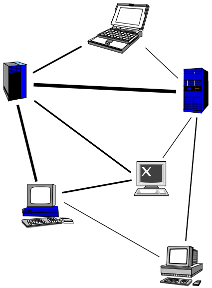


- Either buy a bigger computer,
- Or use several computers.

Question

We need to decide where and when to compute.

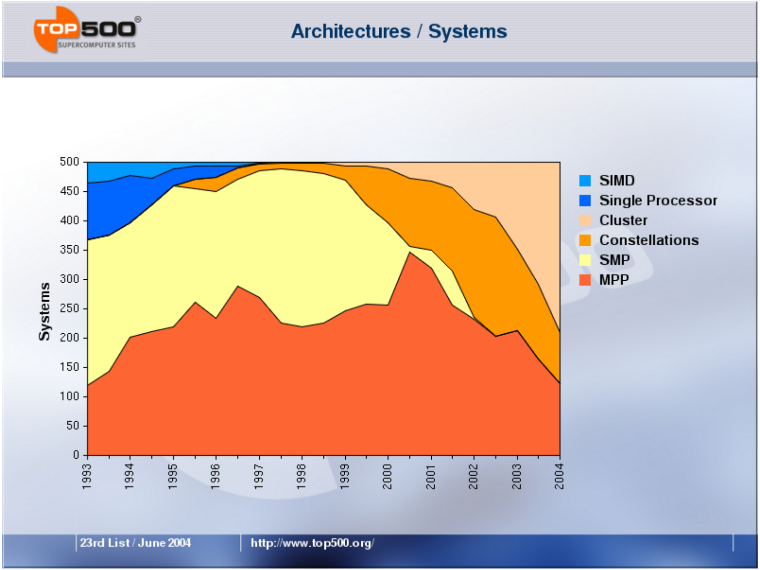
There are two options to increase the available computer power :



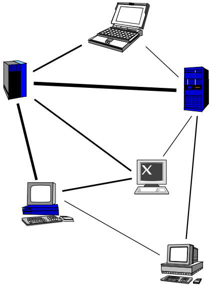
- Either buy a bigger computer,
- Or use several computers.

Question

We need to decide where and when to compute.


TOP500 SUPERCOMPUTER SITES
Architectures / Systems
Systems
500
450
400
350
300
250
200
150
100
50
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
SIMD
Single Processor
Cluster
Constellations
SMP
MPP
23rd List / June 2004
http://www.top500.org/

There are two options to increase the available computer power :

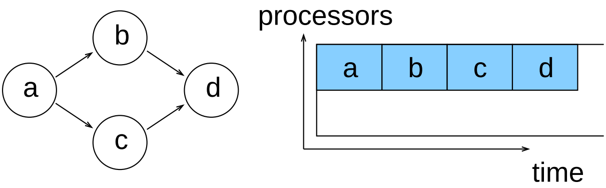


- Either buy a bigger computer,
- Or use several computers.

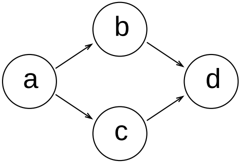
Question

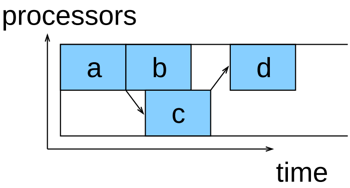
We need to decide where and when to compute.

Usually "where and when" is depicted in a Gantt diagram :

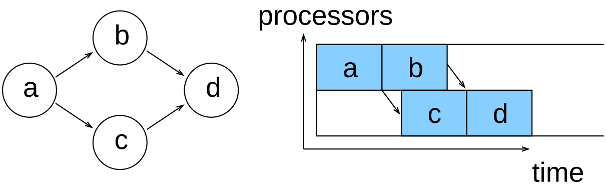


Usually "where and when" is depicted in a Gantt diagram :





Usually "where and when" is depicted in a Gantt diagram :



Task characteristics

- predictable or unpredictable
- identical or different
- independent or precedence constrained
- sequential or multiprocessor
  multiprocessor tasks are :
  - rigid or moldable

Machine characteristics

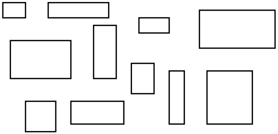
- off-line or on-line
- homogeneous or heterogeneous processors
- homogeneous or heterogeneous links
- simple topology or any graph

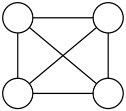
Task characteristics

- predictable or unpredictable
- identical or different
- independent or precedence constrained
- sequential or multiprocessor
  multiprocessor tasks are :
  - rigid or moldable

Machine characteristics

- off-line or on-line
- homogeneous or heterogeneous processors
- homogeneous or heterogeneous links
- simple topology or any graph



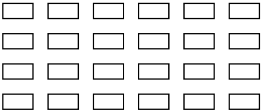


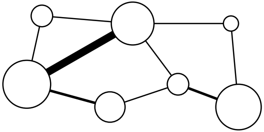
Task characteristics

- predictable or unpredictable
- identical or different
- independent or precedence constrained
- sequential or multiprocessor
  multiprocessor tasks are :
  - rigid or moldable

Machine characteristics

- off-line or on-line
- homogeneous or heterogeneous processors
- homogeneous or heterogeneous links
- simple topology or any graph



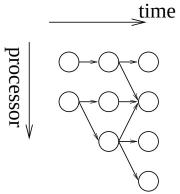


Outline

Outline

Moldable tasks concept

- fine grain execution graphs are replaced by boxes
- execution time depends on the number of processors



#proc	2	3	4
t	5	4	3
W	10	12	12

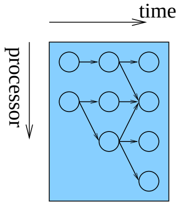
Monotony hypothesis

When p increases :

- t is nonincreasing
- W is nondecreasing

Moldable tasks concept

- fine grain execution graphs are replaced by boxes
- execution time depends on the number of processors



#proc	2	3	4
t	5	4	3
W	10	12	12

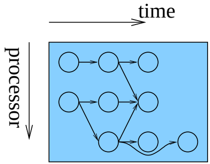
Monotony hypothesis

When p increases :

- t is nonincreasing
- W is nondecreasing

Moldable tasks concept

- fine grain execution graphs are replaced by boxes
- execution time depends on the number of processors



#proc	2	3	4
t	5	4	3
W	10	12	12

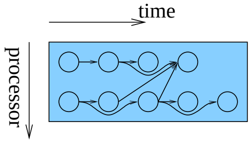
Monotony hypothesis

When p increases :

- t is nonincreasing
- W is nondecreasing

Moldable tasks concept

- fine grain execution graphs are replaced by boxes
- execution time depends on the number of processors



#proc	2	3	4
t	5	4	3
W	10	12	12

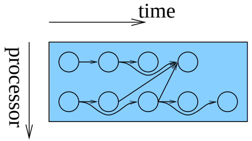
Monotony hypothesis

When p increases :

- t is nonincreasing
- W is nondecreasing

Moldable tasks concept

- fine grain execution graphs are replaced by boxes
- execution time depends on the number of processors



#proc	2	3	4
t	5	4	3
W	10	12	12

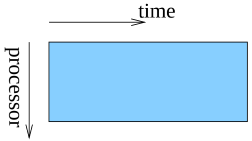
Monotony hypothesis

When p increases :

- t is nonincreasing
- W is nondecreasing

Moldable tasks concept

- fine grain execution graphs are replaced by boxes
- execution time depends on the number of processors



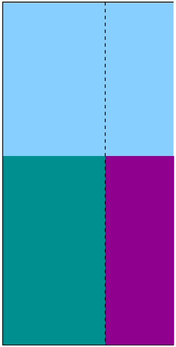
#proc	2	3	4
t	5	4	3
W	10	12	12

Monotony hypothesis

When p increases :

- t is nonincreasing
- W is nondecreasing

Previous results

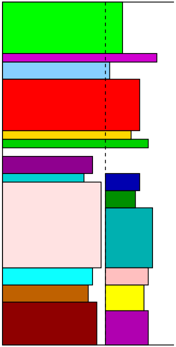


[Mounié et al. 01] gave a $\frac{3}{2}$ approximation algorithm for independent tasks.

Algorithm

- partition tasks
- make a few transformations
- build a shelf schedule

Previous results



[Mounié et al. 01] gave a $\frac{3}{2}$ approximation algorithm for independent tasks.

Algorithm

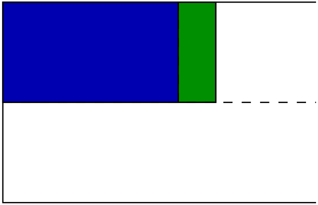
- partition tasks
- make a few transformations
- build a shelf schedule

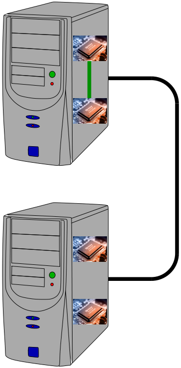
Outline

1. Introduction
2. Moldable Tasks
   - Presentation of the Model
   - Hierarchical Scheduling
   - Bicriteria Scheduling
3. Master-Slave Tasking
   - Presentation of the Model
   - Polynomial Algorithms
   - NP-Hardness
4. Conclusion

Hierarchical scheduling

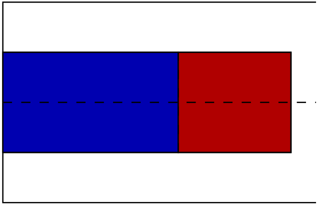
With two levels of communication, t is not a function of p anymore :

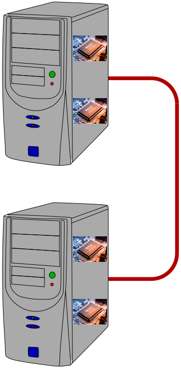




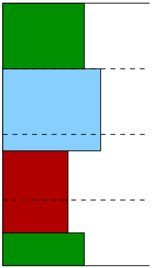
Hierarchical scheduling

With two levels of communication, t is not a function of p anymore :





To keep writing t as a function of p, we introduce a placement rule :



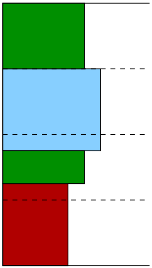
Placement rule

For any given allocation :

- fill as many multiprocessors as possible
- group remaining processors in the same multiprocessor

This placement minimizes the number of clusters used by a task. We can prove that it is the best placement for biprocessors.

To keep writing t as a function of p, we introduce a placement rule :



Placement rule

For any given allocation :

- fill as many multiprocessors as possible
- group remaining processors in the same multiprocessor

This placement minimizes the number of clusters used by a task. We can prove that it is the best placement for biprocessors.

Contiguity

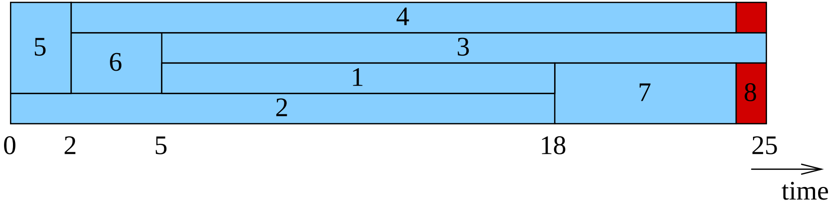
Tasks may not always be represented with rectangles.

Tasks	1	2	3	4	5	6	7	8
1 proc.	13	18	20	22	6	6	12	3
2 proc.	13	18	20	22	3	3	6	1.5
3 proc.	13	18	20	22	2	3	6	1
4 proc.	13	18	20	22	2	3	6	1

Contiguity

Tasks may not always be represented with rectangles.

Tasks	1	2	3	4	5	6	7	8
1 proc.	13	18	20	22	6	6	12	3
2 proc.	13	18	20	22	3	3	6	1.5
3 proc.	13	18	20	22	2	3	6	1
4 proc.	13	18	20	22	2	3	6	1



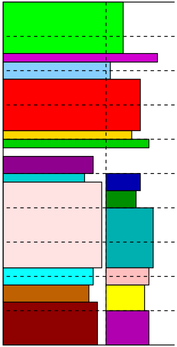
Summary

Problem

We consider :

- independent moldable tasks
- hierarchical platform
  - identical processors
  - fully connected clusters of size 2^q
- objective function : makespan

Algorithm [SPAA01]



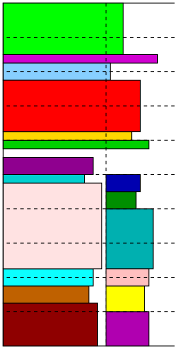
Using the placement rule, we get :

- same guaranty as the homogeneous case for biprocessors and quadriprocessors
- $(2 - \frac{2}{2^q})$ for other values of q

Keypoint of the proof

Remainders are reduced to powers of 2 and sorted.

Algorithm [SPAA01]



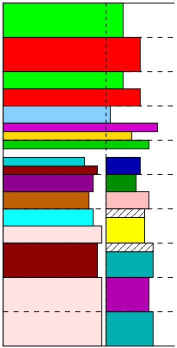
Using the placement rule, we get :

- same guaranty as the homogeneous case for biprocessors and quadriprocessors
- $(2 - \frac{2}{2^q})$ for other values of q

Keypoint of the proof

Remainders are reduced to powers of 2 and sorted.

Algorithm [SPAA01]



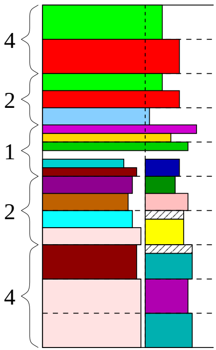
Using the placement rule, we get :

- same guaranty as the homogeneous case for biprocessors and quadriprocessors
- $(2 - \frac{2}{2^q})$ for other values of q

Keypoint of the proof

Remainders are reduced to powers of 2 and sorted.

Algorithm [SPAA01]



Using the placement rule, we get :

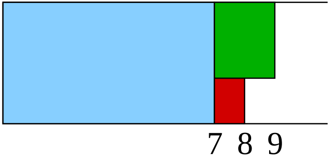
- same guaranty as the homogeneous case for biprocessors and quadriprocessors
- $(2 - \frac{2}{2^q})$ for other values of q

Keypoint of the proof

Remainders are reduced to powers of 2 and sorted.

Outline

Until now we only used the makespan criterion. However there are other possible objective functions such as the minsum criterion.



$$\max_i(C_i) = 9$$

$$\sum_i C_i = 24$$

$$\max_i(C_i) = 9$$

$$\sum_i C_i = 12$$

Summary

Problem

We consider :

- independent moldable tasks
- identical processors
- fully connected
- objective function : makespan and minsum

Preliminary definition

ρ-MSWP algorithm

A ρ approximation algorithm solving the Maximum Scheduled Weight Problem (*MSWP*) takes as input :

- a set of weighted jobs
- a deadline D

Selects some jobs, and produces :

- a schedule of length ρD
- with as much weight as the optimal schedule does in D units of time.

Preliminary definition

ρ-MSWP algorithm

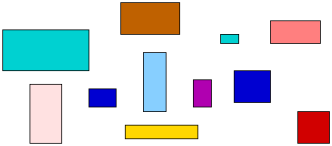
A ρ approximation algorithm solving the Maximum Scheduled Weight Problem (*MSWP*) takes as input :

- a set of weighted jobs
- a deadline D

Selects some jobs, and produces :

- a schedule of length ρD
- with as much weight as the optimal schedule does in D units of time.

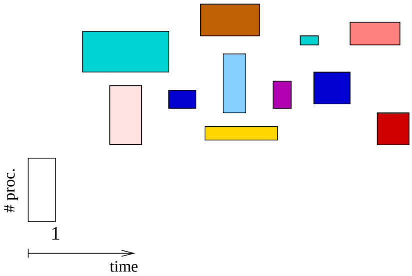
We improved an execution scheme presented by [Hall et al. 96] :



Algorithm

- find the smallest possible execution time t_{min}
- make a box of size $2\rho t_{min}$
- fill the box with as much weight as possible (with a ρ-*MSWP* algorithm)
- double the size of the box and continue

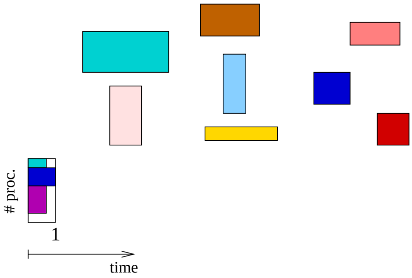
We improved an execution scheme presented by [Hall et al. 96] :



Algorithm

- find the smallest possible execution time t_{min}
- make a box of size $2\rho t_{min}$
- fill the box with as much weight as possible (with a ρ-*MSWP* algorithm)
- double the size of the box and continue

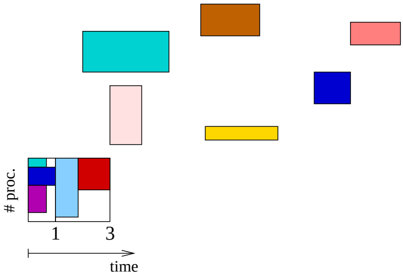
We improved an execution scheme presented by [Hall et al. 96] :



Algorithm

- find the smallest possible execution time t_{min}
- make a box of size $2\rho t_{min}$
- fill the box with as much weight as possible (with a ρ-*MSWP* algorithm)
- double the size of the box and continue

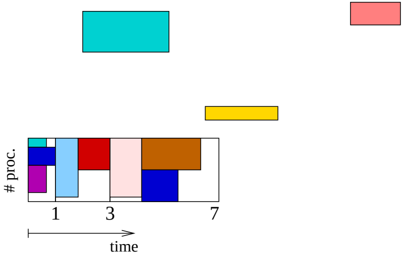
We improved an execution scheme presented by [Hall et al. 96] :



Algorithm

- find the smallest possible execution time t_{min}
- make a box of size $2\rho t_{min}$
- fill the box with as much weight as possible (with a ρ-*MSWP* algorithm)
- double the size of the box and continue

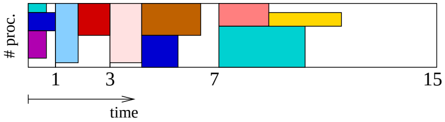
We improved an execution scheme presented by [Hall et al. 96] :



Algorithm

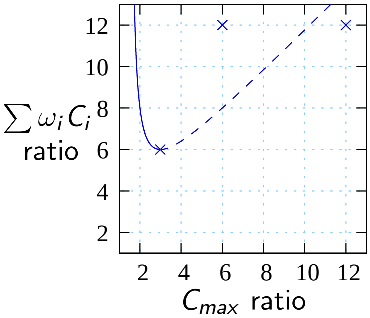
- find the smallest possible execution time t_{min}
- make a box of size $2\rho t_{min}$
- fill the box with as much weight as possible (with a ρ-*MSWP* algorithm)
- double the size of the box and continue

We improved an execution scheme presented by [Hall et al. 96] :



Algorithm

- find the smallest possible execution time t_{min}
- make a box of size $2\rho t_{min}$
- fill the box with as much weight as possible (with a ρ-*MSWP* algorithm)
- double the size of the box and continue

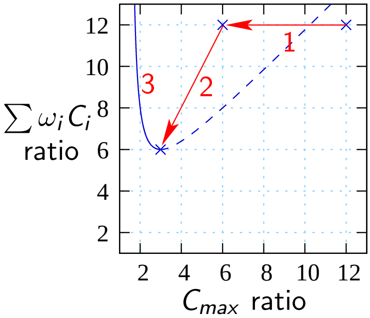


Improvements

1. off-line
2. better ρ-*MSWP* algorithm
3. parameter α

(makespan ;minsum) guaranty

$$\left(\frac{\alpha}{\alpha-1}\rho; \frac{\alpha^2}{\alpha-1}\rho\right)$$

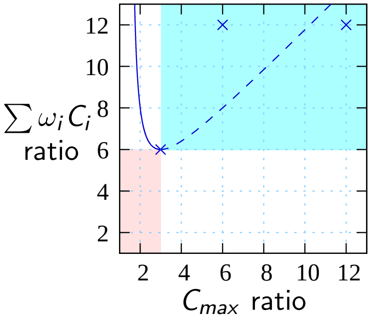


Improvements

1. off-line
2. better ρ-*MSWP* algorithm
3. parameter α

(makespan ;minsum) guaranty

$$\left(\frac{\alpha}{\alpha-1}\rho; \frac{\alpha^2}{\alpha-1}\rho\right)$$



Improvements

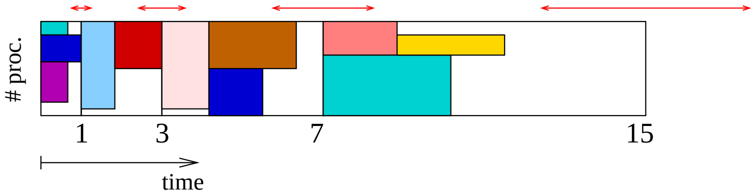
1. off-line
2. better ρ-*MSWP* algorithm
3. parameter α

(makespan ;minsum) guaranty

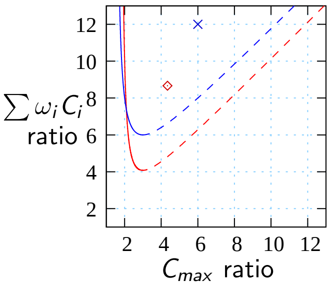
$$\left(\frac{\alpha}{\alpha-1}\rho; \frac{\alpha^2}{\alpha-1}\rho\right)$$

Randomization scheme

The worst cases are when a task is close to the time limits. We move randomly these limits.



Randomization scheme [Algorithmica(submitted)]



Multiplying the time scale by a random $\beta \in]\frac{1}{\alpha}; 1]$ we get :

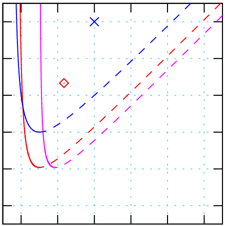
$$E\left[\sum_{i=1}^{n} w_i \bar{C}_i\right] \leq \frac{\alpha\rho}{\ln(\alpha)} \sum_{i=1}^{n} w_i C_i^*$$

Mean guaranties

$$\left(\left(1 + \frac{1}{\ln(\alpha)}\right)\rho; \frac{\alpha}{\ln(\alpha)}\rho\right)$$

$$\boxed{(3; 4.08) < (3.66; 4.33)}$$

Randomization scheme [Algorithmica(submitted)]



Multiplying the time scale by a random $\beta \in]\frac{1}{\alpha}; 1]$ we get :

$$E\left[\sum_{i=1}^{n} w_i \bar{C}_i\right] \leq \frac{\alpha\rho}{\ln(\alpha)} \sum_{i=1}^{n} w_i C_i^*$$

Mean guaranties

$$\left(\left(1 + \frac{1}{\ln(\alpha)}\right)\rho; \frac{\alpha}{\ln(\alpha)}\rho\right)$$

$$\boxed{(3; 4.08) < (3.66; 4.33)}$$

This scheme can be used in several cases, depending on the underlying ρ-*MSWP* algorithm :

- rigid parallel tasks
- moldable tasks
- hierarchical moldable tasks

We may also use it in an on-line setting

Outline

Applications

Features

We are considering applications with the following nice properties :

- small instruction set
- large data set
- computation times are constant

We use independent identical tasks.

Applications

Features

We are considering applications with the following nice properties :

- small instruction set
- large data set
- computation times are constant

We use independent identical tasks.

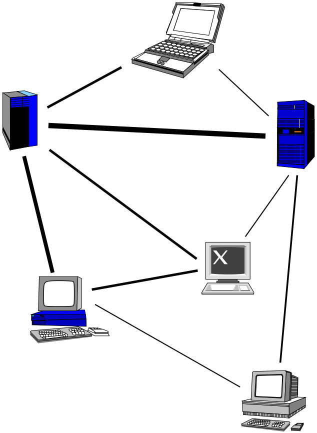
Applications

Some close matches are :

- parameterized computation (CiGri)
- SETI@home
- Mersenne prime search
- Décrypthon

This problem is related to divisible load tasks [Cheng & Robertazzi 88]

Platforms

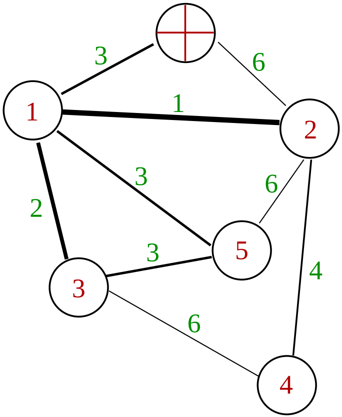


Definition

We consider heterogeneous platforms :

- heterogeneous links
- heterogeneous processors
- centralized data

Platforms



Definition

We consider heterogeneous platforms :

- heterogeneous links
- heterogeneous processors
- centralized data

Why heterogeneous ?

As hardware evolves, one site often has very different kinds of computers available.

Some homogeneous processors graphs may also be seen as heterogeneous chains. [Li 02]

A heterogeneous cluster ranked 7^{th} in the last Top500 ranking.

Heterogeneity allows for bigger computing grids

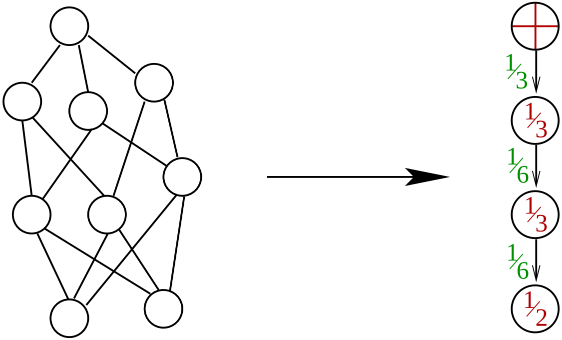
Why heterogeneous ?

As hardware evolves, one site often has very different kinds of computers available.

Some homogeneous processors graphs may also be seen as heterogeneous chains. [Li 02]

A heterogeneous cluster ranked 7^{th} in the last Top500 ranking.

Heterogeneity allows for bigger computing grids


1/3
1/3
1/6
1/3
1/6
1/2

Why heterogeneous ?

As hardware evolves, one site often has very different kinds of computers available.

Some homogeneous processors graphs may also be seen as heterogeneous chains. [Li 02]

A heterogeneous cluster ranked 7^{th} in the last Top500 ranking.

Heterogeneity allows for bigger computing grids

Why heterogeneous ?

As hardware evolves, one site often has very different kinds of computers available.

Some homogeneous processors graphs may also be seen as heterogeneous chains. [Li 02]

A heterogeneous cluster ranked 7^{th} in the last Top500 ranking.

Heterogeneity allows for bigger computing grids

Communications

1-port $\Rightarrow$ One send at a time
$\Rightarrow$ One receive at a time
We can still send, receive and compute at the same time.

No overhead, no gap: Communication times are linear in link speed and datasize.

No routing: A node can only speak to its neighbours, which can forward the task further.

Communications

1-port ⇒ One send at a time
⇒ One receive at a time
We can still send, receive and compute at the same time.

No overhead, no gap: Communication times are linear in link speed and datasize.

No routing: A node can only speak to its neighbours, which can forward the task further.

Communications

1-port $\Rightarrow$ One send at a time
$\Rightarrow$ One receive at a time
We can still send, receive and compute at the same time.

No overhead, no gap: Communication times are linear in link speed and datasize.

No routing: A node can only speak to its neighbours, which can forward the task further.

Our goal

Let n be the number of tasks and t the makespan.

Three similar goals :

1. given n, minimize t
2. given t, maximize n
3. given n and t provide a schedule if it is possible

Our goal

Let n be the number of tasks and t the makespan.

Three similar goals :

1. given n, minimize t
2. given t, maximize n
3. given n and t provide a schedule if it is possible

Our goal

Let n be the number of tasks and t the makespan.

Three similar goals :

1. given n, minimize t
2. given t, maximize n
3. given n and t provide a schedule if it is possible

Summary

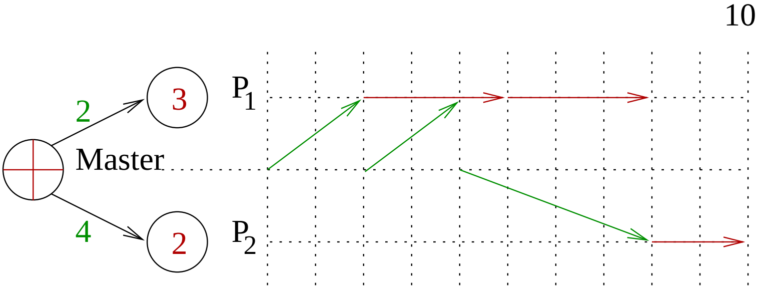
Definition

We consider :

- independent identical tasks
- heterogeneous processors
- heterogeneous links
- communications are *one-port*
- objective function : makespan

A schedule

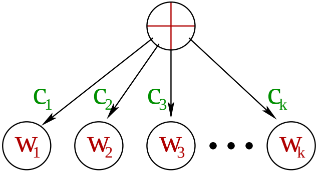
Here is the Gantt chart of a schedule :



The numbers are (respectively) the time needed to send/compute

Previous results

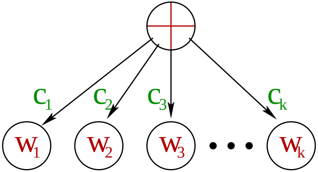
[Beaumont et al. 02] provided an optimal algorithm for fork-graphs which is polynomial in both n and t.



Only one shared resource : the outbound link from the master
$\Longrightarrow$ bandwith-centric allocation.

Previous results

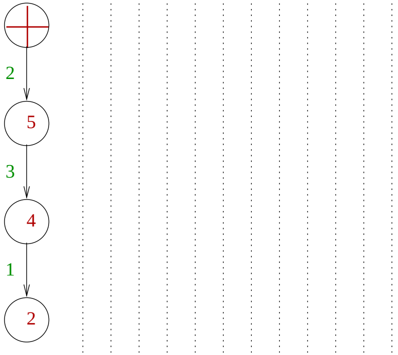
[Beaumont et al. 02] provided an optimal algorithm for fork-graphs which is polynomial in both n and t.



Only one shared resource : the outbound link from the master $\Longrightarrow$ bandwith-centric allocation.

Outline

Heterogeneous Chains



We kept this idea of not spending too much time communicating.

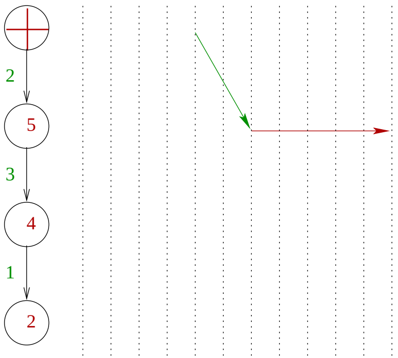
Algorithm

Starting from the end, for each task :

1. Try every processor
2. Choose the "cheapest" option (wrt communications)

Complexity is $O(np^2)$.

Heterogeneous Chains



We kept this idea of not spending too much time communicating.

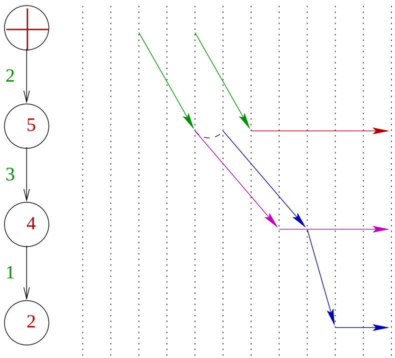
Algorithm

Starting from the end, for each task :

1. Try every processor
2. Choose the “cheapest” option (wrt communications)

Complexity is $O(np^2)$.

Heterogeneous Chains



We kept this idea of not spending too much time communicating.

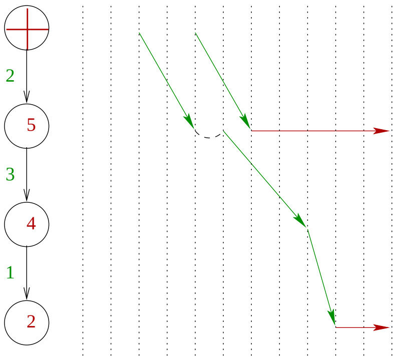
Algorithm

Starting from the end, for each task :

1. Try every processor
2. Choose the “cheapest” option (wrt communications)

Complexity is $O(np^2)$.

Heterogeneous Chains



We kept this idea of not spending too much time communicating.

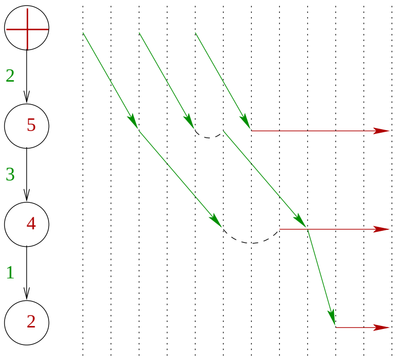
Algorithm

Starting from the end, for each task :

1. Try every processor
2. Choose the "cheapest" option (wrt communications)

Complexity is $O(np^2)$.

Heterogeneous Chains



We kept this idea of not spending too much time communicating.

Algorithm

Starting from the end, for each task :

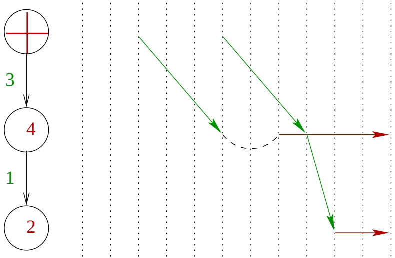
1. Try every processor
2. Choose the "cheapest" option (wrt communications)

Complexity is $O(np^2)$.

Heterogeneous Chains

Keypoint of the proof

Induction on the sub-chains.



We kept this idea of not spending too much time communicating.

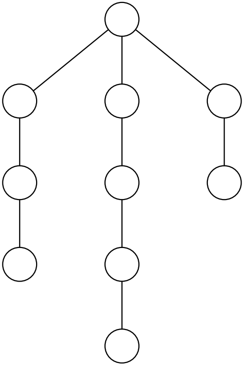
Algorithm

Starting from the end, for each task :

1. Try every processor
2. Choose the “cheapest” option (wrt communications)

Complexity is $O(np^2)$.

Heterogeneous Spiders [IPDPS03]



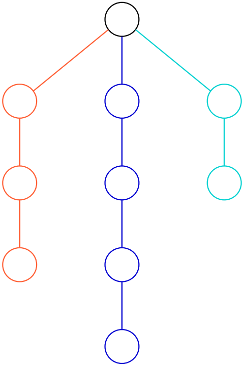
Definition

A spider is a collection of chains with a single master.

Algorithm

1. Compute the optimal schedule for each chain
2. Replace each chain by a fork
3. Compute the optimal schedule for the fork
4. Revert to a spider schedule

Heterogeneous Spiders [IPDPS03]



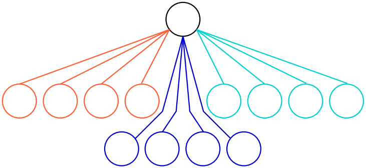
Definition

A spider is a collection of chains with a single master.

Algorithm

1. Compute the optimal schedule for each chain
2. Replace each chain by a fork
3. Compute the optimal schedule for the fork
4. Revert to a spider schedule

Heterogeneous Spiders [IPDPS03]



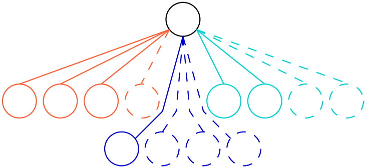
Definition

A spider is a collection of chains with a single master.

Algorithm

1. Compute the optimal schedule for each chain
2. Replace each chain by a fork
3. Compute the optimal schedule for the fork
4. Revert to a spider schedule

Heterogeneous Spiders [IPDPS03]



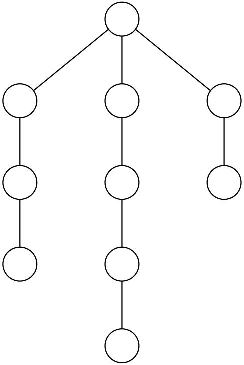
Definition

A spider is a collection of chains with a single master.

Algorithm

1. Compute the optimal schedule for each chain
2. Replace each chain by a fork
3. Compute the optimal schedule for the fork
4. Revert to a spider schedule

Heterogeneous Spiders [IPDPS03]



Definition

A spider is a collection of chains with a single master.

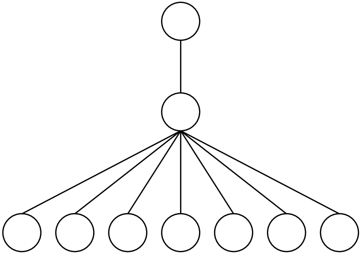
Algorithm

1. Compute the optimal schedule for each chain
2. Replace each chain by a fork
3. Compute the optimal schedule for the fork
4. Revert to a spider schedule

Outline

Trees [EJOR04]

For general trees the problem is NP-hard



- The reduction is made from 3-partition.
- The tree used in the reduction is a fork graph connected to the master node by a single link.

Results

Moldable tasks

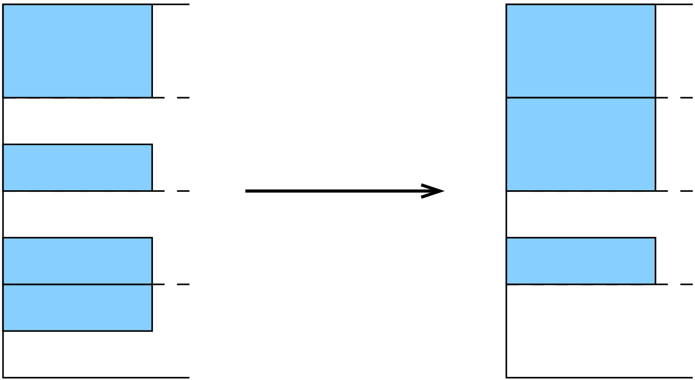
- optimal polynomial algorithm for a constrained case with precedence constraints
- efficient algorithm for hierarchical moldable tasks
- improved general scheme for bicriteria scheduling

Master-slave tasking

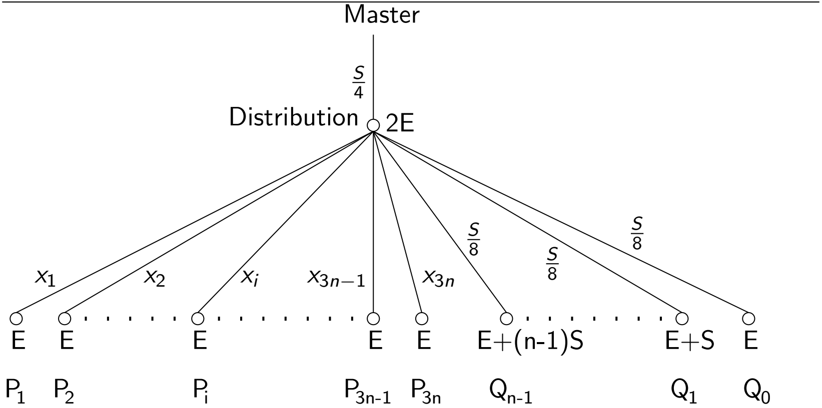
- optimal polynomial algorithm for chains and spider graphs
- NP-hardness of trees

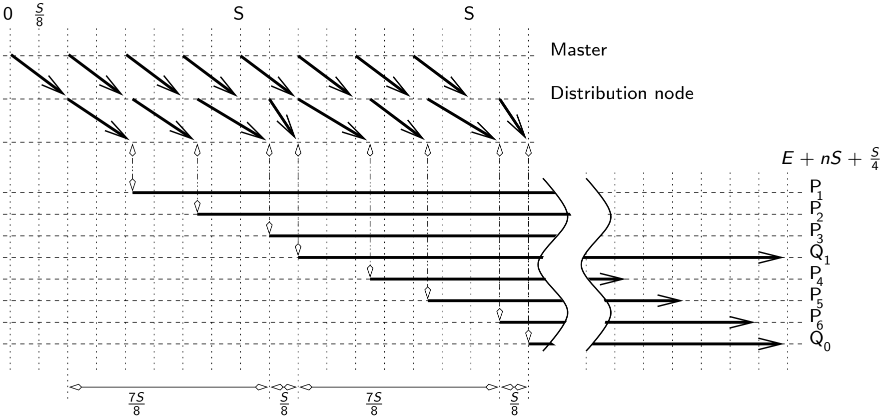
Future works

- Implementation of the algorithms within CiGri and OAR
- Promote the use of moldable tasks
- Consider other criteria for master-slave tasking
- Multicriteria algorithms for multi-users settings



$$\sum_{i=1}^{3n} x_i = n\frac{7S}{8}$$





0
$\frac{S}{8}$
S
S
Master
Distribution node
$E + nS + \frac{S}{4}$
P_1
P_2
P_3
Q_1
P_4
P_5
P_6
Q_0
$\frac{7S}{8}$
$\frac{S}{8}$
$\frac{7S}{8}$
$\frac{S}{8}$