# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(()</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! <sup>(C)</sup>

# My historical perspective

- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I wondered what I should do this year?
- ullet Maybe I should find a nice little scheduling problem!  $\odot$

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

2/35

- Parallel algorithm design and scheduling were already difficult tasks with homogeneous machines
- On heterogeneous platforms, it gets worse
- $\bullet$  Patrick Geoffray went from kindergarten to Myricom but he's still a kid!  $\textcircled{\sc c}$
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 🙁

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

2/35

- Parallel algorithm design and scheduling were already difficult tasks with homogeneous machines
- On heterogeneous platforms, it gets worse
- Patrick Geoffray went from kindergarten to Myricom but he's still a kid! <sup>(1)</sup>
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 🙁

2/35

- Parallel algorithm design and scheduling were already difficult tasks with homogeneous machines
- On heterogeneous platforms, it gets worse
- $\bullet$  Patrick Geoffray went from kindergarten to Myricom but he's still a kid!  $\textcircled{\sc c}$
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 🙁

2/35

- Parallel algorithm design and scheduling were already difficult tasks with homogeneous machines
- On heterogeneous platforms, it gets worse
- Patrick Geoffray went from kindergarten to Myricom but he's still a kid!  $\textcircled{\sc c}$
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 🙁

2/35

- Parallel algorithm design and scheduling were already difficult tasks with homogeneous machines
- On heterogeneous platforms, it gets worse
- Patrick Geoffray went from kindergarten to Myricom but he's still a kid!  $\textcircled{\sc c}$
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 😊

# A nice little embarrassingly parallel application



- One (divisible load) application running on each cluster
  ⇒ Which fraction of the job to delegate to other clusters?
- Different communication-to-computation ratios
  ⇒ How to ensure fair scheduling and good resource utilization?

Conclusion

4/35

The great talk you've been expecting

# Revisiting matrix product on heterogeneous platforms

#### Jack Dongarra, Zhiao Shi, UT Knwoville

Jean-François Pineau, Yves Robert, Frédéric Vivien, ENS Lyon



Simulations

Divisible Loads

Conclusion

4/35

The great talk you've been expecting



▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

# Scheduling and Data Redistribution Strategies on Star Platforms

Loris Marchal, Veronika Rehn, Yves Robert and Frédéric Vivien

GRAAL team, LIP École Normale Supérieure de Lyon

September 2006

Target problem 0000000000	Simulations	Divisible Loads	Conclusion
<b>o</b> ''			
Outline			6/35

# Target problem

- Fully homogeneous platforms
- Bus platforms
- General platforms

# 2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

# 4 Conclusion

Target problem	Simulations	Divisible Loads	Conclusion
Outline			7/35

# 1 Target problem

- Fully homogeneous platforms
- Bus platforms
- General platforms

# 2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

# 4 Conclusion

Target problem	Simulations	Divisible Loads	Conclusion
Example			8/35





Target problem	Simulations	Divisible Loads	Conclusion
- ·			
Example			8/35





Target problem	Simulations	Divisible Loads	Conclusion
Example			0/25





Target problem	Simulations	Divisible Loads	Conclusion
Example			0/25





Target problem	Simulations	Divisible Loads	Conclusion
Example			8/35





Target problem	Simulations	Divisible Loads	Conclusion
Example			8/35



P.

(Р1

P

 $(P_4)$ 



▲□▶ ▲圖▶ ★園▶ ★園▶ - 園 - の��

Target problem	Simulations	Divisible Loads	Conclusion
Example			8/35







◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 の�?

Target problem	Simulations	Divisible Loads	Conclusion
Framework			9/35

- Master-slave platforms
- New: Distributed loads

#### Problem

Redistribution of data Goal: Minimize overall processing time

# Data models

- Independent tasks
- Divisible loads

Target problem	Simulations	Divisible Loads	Conclusion
Framework			9/35

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

- Master-slave platforms
- New: Distributed loads

### Problem

# Redistribution of data

Goal: Minimize overall processing time

# Data models

- Independent tasks
- Divisible loads

Target problem	Simulations	Divisible Loads	Conclusion
Framework			0/35

▲ロト ▲御 ト ▲ 臣 ト ▲ 臣 ト の Q @

- Master-slave platforms
- New: Distributed loads

### Problem

Redistribution of data Goal: Minimize overall processing time

# Data models

- Independent tasks
- Divisible loads

**Related Work** 

Simulations

Divisible Loads

10/35



#### Our approach



### Independent tasks

- Application: BOINC (e.g. Einstein@home)
- NP-completeness for different task sizes

# Divisible load theory

- Perfect parallel jobs
- Optimal algorithms for video processing (Altilar, Paker)

### Redistribution algorithms

- NP-completeness (Kremer)
- Optimality for particular cases: homogeneous ring topologies

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Related Work

Simulations

Divisible Loads

10/35



#### Our approach



# Independent tasks

- Application: BOINC (e.g. Einstein@home)
- NP-completeness for different task sizes

# Divisible load theory

- Perfect parallel jobs
- Optimal algorithms for video processing (Altilar, Paker)

#### Redistribution algorithms

- NP-completeness (Kremer)
- Optimality for particular cases: homogeneous ring topologies

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

**Related Work** 

Simulations

Divisible Loads

10/35





### Independent tasks

- Application: BOINC (e.g. Einstein@home)
- NP-completeness for different task sizes

# Divisible load theory

- Perfect parallel jobs
- Optimal algorithms for video processing (Altilar, Paker)

#### Redistribution algorithms

- NP-completeness (Kremer)
- Optimality for particular cases: homogeneous ring topologies

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへ⊙

Model

#### Si

Div



- Star network  $S = P_0, P_1, \ldots, P_m$
- Communication cost c<sub>i</sub>
- Computing power w<sub>i</sub>
- Initial data L<sub>i</sub>
- Independent and identical tasks

- Linear cost model
- Bidirectional one-port model
- Objective function: Minimize makespan

Model

#### Simulations

Divisible Loads



- Star network  $S = P_0, P_1, \ldots, P_m$
- Communication cost c<sub>i</sub>
- Computing power *w<sub>i</sub>*
- Initial data L<sub>i</sub>
- Independent and identical tasks

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

- Linear cost model
- Bidirectional one-port model
- Objective function: Minimize makespan
Simulations

Divisible Loads

Conclusion

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



#### Principle: Local optimization of current makespan

12/35

Simulations

Divisible Loads

Conclusion

12/35

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



Simulations

Divisible Loads

Conclusion

12/35

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



#### Principle: Local optimization of current makespan

Simulations

Divisible Loads

Conclusion

12/35

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



Simulations

Divisible Loads

Conclusion

12/35

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



Simulations

Divisible Loads

Conclusion

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



Simulations

Divisible Loads

Conclusion

12/35

Fully homogeneous platforms

# Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers



#### Principle: Local optimization of current makespan

Target problem ⊙●○○○○○○○○○	Simulations	Divisible Loads	Conclusion
Fully homogeneous platforms			
BBA- Optimality			13/35

#### Theorem

BEST-BALANCE ALGORITHM calculates an optimal schedule *S* on a fully homogeneous star network.

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 のへで

#### Bus platforms

# Moore-Based Binary-Search Algorithm - MBBSA

- Homogeneous communication links
- Heterogeneous workers
- Makespan M

## Principle:

- Moore's algorithm
- Schedule within M
- Binary search

#### Moore's algorithm

#### Bus platforms

# Moore-Based Binary-Search Algorithm - MBBSA

- Homogeneous communication links
- Heterogeneous workers
- Makespan M

## Principle:

- Moore's algorithm
- Schedule within M
- Binary search

#### Moore's algorithm

1: Order the jobs by non-decreasing deadlines:  $d_1 < d_2 < \cdots < d_d$ 2:  $\sigma \leftarrow \emptyset$ :  $t \leftarrow 0$ 3: **for** i := 1 to *n* **do** 4:  $\sigma \leftarrow \sigma \cup \{i\}$ 5:  $t \leftarrow t + w_i$ 6: if  $t > d_i$  then Find job *j* in  $\sigma$  with 7: largest  $w_i$  value  $\sigma \leftarrow \sigma \setminus \{j\}$ 8:  $t \leftarrow t - w_i$ 9: end if 10: 11: end for

Target problem ○○○●○○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase 1	+ 2		15/35

#### Determination of senders and receivers



▲□▶ ▲圖▶ ▲≣▶ ▲≣▶ = 差 = のへで

Target problem ○○○●○○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase 1	+ 2		15/35

#### Determination of senders and receivers



▲□▶ ▲圖▶ ▲≣▶ ▲≣▶ = 差 = のへで

Target problem ○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	e 3		16/35

#### Computation of deadlines



◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 \_ のへで

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



・ロト・日本・日本・日本・日本・日本

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



◆□▶ ◆□▶ ◆臣▶ ◆臣▶ ─臣 ─のへで

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



<ロト <回ト < 注ト < 注ト

æ

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



・ロト・日本・日本・日本・日本・日本

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



・ロト・雪ト・雪ト・雪・ 今日・

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

Target problem ○○○○○●○○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Phase	4		17/35



・ロト・雪ト・雪ト・雪・ 今日・

Target problem ○○○○○○●○○○○	Simulations	Divisible Loads	Conclusion
Bus platforms			
MBBSA- Or	otimality		18/35

#### 18/35

#### Theorem

- (i) MBBSA succeeds to build a schedule  $\sigma$  for a given makespan M, if and only if there exists one.
- (ii) Binary search algorithm returns in polynomial time an optimal schedule  $\sigma$  for bus platforms (homogeneous communication links and heterogeneous workers).

Simulations

Divisible Loads

Conclusion

General platforms

Dealing with fully heterogeneous platforms

19/35

#### Difficulty: Who is sender, who is receiver?

M = 12			
Worker	с	W	load
<i>P</i> <sub>1</sub>	1	1	13
P <sub>2</sub>	8	1	13
<i>P</i> <sub>3</sub>	1	9	0
P <sub>4</sub>	1	10	0



▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 のへで

Target problem ○○○○○○○○○○○	Simulations	Divisible Loads	Conclusion
General platforms			
NP-completeness			20/35

# Scheduling Problem for Master-Slave Tasks on a Star of Heterogeneous Processors

#### Definition (SPMSTSHP)

Let N be a star-network. Let T be a deadline.

"Is it possible to redistribute tasks and process them in time T?".

#### Theorem

NP-complete in the strong sense.

Simulations

Divisible Loads

Conclusion

21/35

General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



Simulations

Divisible Loads

Conclusion

General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



21/35

Simu

Divisible

Target problem 000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 … のへで

21/35

Simu

Divisible Loads

Conclusion

21/35

Target problem 0000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 … のへで

Sim

Di

Conclusior

21/35

Target problem 0000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



Si

ations

Divisible Load

21/35

Target problem 000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



Sin

mulations

Divisible Load

21/35

Target problem 0000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 … のへで

Simulations

Divisible Load

21/35

Target problem 0000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



Simi

ulations

Divisible Load

Conclusion

21/35

Target problem 0000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 … のへで

Sim

imulations

Divisible Load

Conclusion

21/35

Target problem 0000000000 General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 … のへで

Simulations

Divisible Loads

Conclusion

21/35

General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



Simulations

Divisible Loads

Conclusion

21/35

General platforms

# Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 … のへで

Target	problem
00000	000000

Simulations

Divisible Loads

・ロト ・ 理 ト ・ ヨ ト ・ ヨ ト

э

General platforms

# Impact of Heterogeneity

Platfor	m type	
Comm.	Comp.	Difficulty
Hom.	Hom.	simple greedy algorithm
Hom.	Het.	complicated algorithm
Het.	Hom.	
Het.	Het.	NP-strong

#### Heuristics

- BBA
- MBBSA
- R-BSA: Reversed-Binary Search Algorithm Combination of greedy algorithm and binary search
| Target | problem |
|--------|---------|
| 00000  | 000000  |

イロト 不得 トイヨト イヨト

æ

General platforms

# Impact of Heterogeneity

Platfor	m type	
Comm.	Comp.	Difficulty
Hom.	Hom.	simple greedy algorithm
Hom.	Het.	complicated algorithm
Het.	Hom.	?
Het.	Het.	NP-strong

- BBA
- MBBSA
- R-BSA: Reversed-Binary Search Algorithm Combination of greedy algorithm and binary search

Target	problem
00000	000000

イロト 不得 トイヨト イヨト

æ

General platforms

# Impact of Heterogeneity

Platform type		
Comm.	Comp.	Difficulty
Hom.	Hom.	simple greedy algorithm
Hom.	Het.	complicated algorithm
Het.	Hom.	?
Het.	Het.	NP-strong

- BBA
- MBBSA
- R-BSA: Reversed-Binary Search Algorithm Combination of greedy algorithm and binary search

Target	problem	
00000	000000	

Divisible Loads

イロト 不得 トイヨト イヨト

э

General platforms

# Impact of Heterogeneity

Platform type		
Comm.	Comp.	Difficulty
Hom.	Hom.	simple greedy algorithm
Hom.	Het.	complicated algorithm
Het.	Hom.	?
Het.	Het.	NP-strong

- BBA
- MBBSA
- R-BSA: Reversed-Binary Search Algorithm Combination of greedy algorithm and binary search

Target	problem
00000	000000

General platforms

# Impact of Heterogeneity

Platform type		
Comm.	Comp.	Difficulty
Hom.	Hom.	simple greedy algorithm
Hom.	Het.	complicated algorithm
Het.	Hom.	?
Het.	Het.	NP-strong

- BBA
- MBBSA
- R-BSA: Reversed-Binary Search Algorithm Combination of greedy algorithm and binary search

Outline 23/35	Target problem 0000000000	Simulations	Divisible Loads	Conclusion
	Outline			23/35

#### Target problem

- Fully homogeneous platforms
- Bus platforms
- General platforms

### 2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

### 4 Conclusion

▲□▶ ▲圖▶ ▲≣▶ ▲≣▶ = 差 = のへで

Target problem 0000000000	Simulations	Divisible Loads	Conclusion
Simulations			24/35

#### SIMGRID:

#### Simulator for distributed applications

- 4 platform types
- 1000 instances
- 10 workers
- Random variables

- c<sub>i</sub>: 1..100
- w<sub>i</sub>: 1..100
- *L<sub>i</sub>*: 0..50

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Target problem 0000000000	Simulations	Divisible Loads	Conclusion
Simulations			24/25

### SIMGRID:

#### Simulator for distributed applications

- 4 platform types
- 1000 instances
- 10 workers
- Random variables

- c<sub>i</sub>: 1..100
- w<sub>i</sub>: 1..100
- L<sub>i</sub>: 0..50

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

Target problem 00000000000	Simulations	Divisible Loads	Concl
Trace Tests			2

### BBA

_									
				lo		5		10	
master_	m_proce	master	Process			× >	× 1		
machine	ss_t	master-	Process						
alar and	m_proce	sbrver£in	Process		$\sim$	R I	$\sim$ '		
staveo	ss_t	processi	Process						
alary a l	m_proce	สสพลาสิเก	Process		8		2		
NAVET	ss_t	processi	Process						
alau a 2	m_proce	abavventuri	Process				,		
siavez	ss_t	processi	Process						
Sauch	m_proce	sbrownfun	Process		,	3	, ,		
staves	ss_t	processi	Process						

### MBBSA

				6 · · · · · · · · · · · · · · · · · · ·
master_	m_proce	master	Process	
machine	ss_t	master-	Process	
dawa	m_proce	stanner2.m	Process	
SIRVED	ss_t	processi	Process	
claural	m_proce	stavvar&un	Process	
alaver	ss_t	processi	Process	
clause2	m_proce	stamentun	Process	
SHAVEZ	ss_t	processi	Process	
Saurala	m_proce	stannarfam	Process	
Slaves	ss_t	processi	Process	

Divisible Loads

### Distance from the Best Heuristic

Heterogeneous platform



500

Divisible Loads

26/35

### Distance from the Best Heuristic

Heterogeneous platform



200

Divisible Loads

## Distance from the Best Heuristic

#### Heterogeneous platform



~ ~ ~ ~

Divisible Loads

### Distance from the Best Heuristic

#### Heterogeneous platform



Divisible Loads

### Distance from the Best Heuristic

26/35

#### Heterogeneous platform



996

Target	problem
00000	

Divisible Loads

Conclusion

27/35

### Standard Deviation

Platfor	m type	Standard deviation			
Comm.	Comm. Comp.		MBBSA	R-BSA	
Hom	Hom	0	0	0.0107	
Hom	Het	0.0006	0	0.0181	
Het	Hom	0.4007		0.0173	
Het	Het	0.3516	0.0327	0.0284	

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへ⊙

Divisible Loads

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

### Standard Deviation

Platfor	m type	Standard deviation			
Comm.	Comp.	BBA	MBBSA	R-BSA	
Hom	Hom	0	0	0.0107	
Hom	Het	0.0006	0	0.0181	
Het	Hom	0.4007	0.0208	0.0173	
Het	Het	0.3516	0.0327	0.0284	

Divisible Loads

### Standard Deviation

-	Platfor	m type	Standard deviation			
	Comm.	Comp.	BBA	MBBSA	R-BSA	
-	Hom	Hom	0	0	0.0107	
-	Hom	Het	0.0006	0	0.0181	
	Het	Hom	0.4007	0.0208	0.0173	
	Het	Het	0.3516	0.0327	0.0284	

Divisible Loads

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

### Standard Deviation

Platfor	m type	Standard deviation			
Comm.	Comp.	BBA	MBBSA	R-BSA	
Hom	Hom	0	0	0.0107	
Hom	Het	0.0006	0	0.0181	
Het	Hom	0.4007	0.0208	0.0173	
Het	Het	0.3516	0.0327	0.0284	

Target problem 0000000000	Simulations	Divisible Loads	Conclusion
Outline			28/35

#### Target problem

- Fully homogeneous platforms
- Bus platforms
- General platforms

### 2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

### 4 Conclusion

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへ⊙

Framework

Simulations

Divisible Loads



- Switch as master
- *m* workers
- Computation speed s<sub>i</sub>
- Bandwidth b<sub>i</sub>
- Divisible load  $\alpha_i$
- Linear cost model
- Overlapped unbounded switch model

▲□▶ ▲圖▶ ★ 国▶ ★ 国▶ - 国 - のへで

Target problem 0000000000	Simulations	Divisible Loads	Conclusion
Redistribution Stra	ategy		30/35

#### Goal: Every worker finishes at the same time



◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Target problem 0000000000	Simulations	Divisible Loads	Conclusion
Redistribution Stra	ategy		30/35

### Goal: Every worker finishes at the same time



・ロト ・回ト ・ヨト ・ヨト

Divisible Loads

31/35

# Solution for Divisible Loads

Imbalance of a worker  $\delta_i$ 

Linear program

 $\begin{array}{ll} \text{MINIMIZE } T, \\ \text{UNDER THE CONSTRAINTS} \\ \left\{ \begin{array}{ll} (1a) & |\delta_i| \leq T \times b_i \\ (1b) & \delta_i \geq \alpha_i - T \times s_i \\ (1c) & \sum_i \delta_i = 0 \end{array} \right. \\ \end{array}$ 

Fraction of load  $f_{i,j}$ 

$$f_{i,j} = \delta_i \times \frac{\delta_j}{\sum_{k \in R} \delta_k} = \delta_i \times \frac{\delta_j}{-L}$$

Communication rate  $\lambda_{i,j}$ 

 $\lambda_{i,j} = \frac{f_{i,j}}{T_0}$ 

Computation rate  $\gamma_{i,j}$ 

$$\gamma_{i,j} = \frac{f_{i,j}}{T_0}$$

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへで

Divisible Loads

### Solution for Divisible Loads

Imbalance of a worker  $\delta_i$ 

Linear program

MINIMIZE T, UNDER THE CONSTRAINTS  $\begin{cases}
(1a) & |\delta_i| \leq T \times b_i \\
(1b) & \delta_i \geq \alpha_i - T \times s_i \\
(1c) & \sum_i \delta_i = 0
\end{cases}$ (1) Fraction of load  $f_{i,j}$ 

$$f_{i,j} = \delta_i \times \frac{\delta_j}{\sum_{k \in \mathcal{R}} \delta_k} = \delta_i \times \frac{\delta_j}{-L}$$

Communication rate  $\lambda_{i,j}$ 

 $\lambda_{i,j} = \frac{f_{i,j}}{T_0}$ 

Computation rate  $\gamma_{i,i}$ 

 $\gamma_{i,j} = \frac{f_{i,j}}{T_0}$ 

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

Outline 32/35	Target problem 0000000000	Simulations	Divisible Loads	Conclusion
	Outline			32/35

### Target problem

- Fully homogeneous platforms
- Bus platforms
- General platforms

### 2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

### 4 Conclusion

Conclusion

ヘロマ ふぼう イヨマ イロマ

.....

#### 33/35

Complete study of a difficult load-balancing problem

Scheduling and redistributing data on master-slave platforms Independent tasks:

- General case: Proof of NP-completeness in the strong sense
- Special platforms: Optimal algorithms
- Simulations: Verification of theoretical results

Divisible loads:

• Solution for general case: LP + analytical formulas

#### Perspectives

Beyond the NP-completeness: Search for approximation algorithms Extension to dynamic master-slave platforms Extension to more general interconnection networks Conclusion

#### Complete study of a difficult load-balancing problem

Scheduling and redistributing data on master-slave platforms Independent tasks:

- General case: Proof of NP-completeness in the strong sense
- Special platforms: Optimal algorithms
- Simulations: Verification of theoretical results

Divisible loads:

• Solution for general case: LP + analytical formulas

#### Perspectives

Beyond the NP-completeness: Search for approximation algorithms Extension to dynamic master-slave platforms Extension to more general interconnection networks

### Recent work at our place

### On-line scheduling heuristics for master-slave platforms

- Competitive ratios and inapproximability results
- Communication-aware heuristics

#### Collective communications

- Broadcast, multicast on heterogeneous clusters
- Resource selection for future MPI2 routines

Load-balancing

- Optimize BOINC-like applications
- Data redistribution strategies

Steady-state scheduling

- Multiple applications competing for resources
- Centralized vs fully distributed heuristics

Divisible Loads

# Scheduling for large-scale platforms

Assess the impact of new architectural characteristics

- Heterogeneity
- Irregular network topologies
- Hierarchy
- Variability (volability)

### Inject static knowledge in a (mostly) dynamic environment

- Divisible loads vs bag of tasks
- Steady-state scheduling
- Resource selection
- Evaluation
- Evaluate strategies through simulation
- SimGrid software co-developed with UCSD
- Large-scale experiments with Grid'5000

35/35