My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
I made it to the 8 CCGSC workshops!
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!😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
My historical perspective

- I made it to the 8 CCGSC workshops!
- 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! 😊
Did anything change in the meantime?

- 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! 😊
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 😞
Did anything change in the meantime?

- 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! 😊
  - He says that only embarrassingly parallel applications can be deployed on the grid.
  - Clearly, he is over optimistic! 😞
Did anything change in the meantime?

- 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! 😊
  - He says that only embarrassingly parallel applications can be deployed on the grid.
  - Clearly, he is over optimistic! 😞
Did anything change in the meantime?

- 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! 😊
- He says that only embarrassingly parallel applications can be deployed on the grid
- Clearly, he is over optimistic! 😞
Did anything change in the meantime?

- 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! 😊
- 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?
The great talk you’ve been expecting

Revisiting matrix product on heterogeneous platforms

Jack Dongarra, Zhiao Shi, UT Knoxville

Jean-François Pineau, Yves Robert, Frédéric Vivien, ENS Lyon
The great talk you’ve been expecting

Revisiting matrix product on heterogeneous platforms

Eh wait!
Experiments are not ready?!
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
Outline

1. Target problem
   - Fully homogeneous platforms
   - Bus platforms
   - General platforms

2. Simulations

3. Divisible Loads Using the Multiport Switch-Model

4. Conclusion
1 Target problem
   • Fully homogeneous platforms
   • Bus platforms
   • General platforms

2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

4 Conclusion
Example

Architecture

$P_0$

$P_1$

$P_2$

$P_3$

$P_4$

$T = 0$

$P_1$

$P_2$

$P_3$

$P_4$
Example

Architecture

\[ T = 0 \]
Example

Architecture

Diagram showing a tree structure with nodes labeled $P_0$, $P_1$, $P_2$, $P_3$, and $P_4$.

The diagram illustrates the architecture with nodes and their respective states at time $T = 0$.
Example

Architecture

\( P \)

\( T = 0 \)

\( P_1 \)

\( P_2 \)

\( P_3 \)

\( P_4 \)
Example

Architecture

Target problem
Simulations
Divisible Loads
Conclusion

Architecture diagram:
- Node $P_0$ is the root of the tree, with $P_1$, $P_2$, $P_3$, and $P_4$ as its children.
- Each node represents a process or task.
- The diagram illustrates the divisibility of loads across processes.

Example graph:
- Processes $P_1$, $P_2$, $P_3$, and $P_4$ are depicted.
- The graph shows the execution times and dependencies between processes.
- $T = 0$ indicates the initial state or time point.

The diagram visually represents the architecture and divisibility of loads across processes.
Example
Framework

- Master-slave platforms
- New: Distributed loads

Problem

Redistribution of data

Goal: Minimize overall processing time

Data models

- Independent tasks
- Divisible loads
Problem

Redistribution of data

Goal: Minimize overall processing time

Data models

- Independent tasks
- Divisible loads
Framework

- Master-slave platforms
- **New**: Distributed loads

**Problem**

Redistribution of data

**Goal**: Minimize overall processing time

**Data models**

- Independent tasks
- Divisible loads
### Related Work

#### Known results

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

#### Our approach

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

#### Independent tasks

Redistribution algorithms
- NP-completeness (Kremer)
- Optimality for particular cases: homogeneous ring topologies
Related Work

**Known results**

- 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)

**Our approach**

Redistribution algorithms

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

**Known results**
- 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

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

- Star network \( S = P_0, P_1, \ldots, P_m \)
- Communication cost \( c_i \)
- Computing power \( w_i \)
- Initial data \( L_i \)
- Independent and identical tasks
- Linear cost model
- Bidirectional one-port model

Objective function:
Minimize makespan
Model

- Star network $S = P_0, P_1, \ldots, P_m$
- Communication cost $c_i$
- Computing power $w_i$
- Initial data $L_i$
- Independent and identical tasks
- Linear cost model
- Bidirectional one-port model
- Objective function: Minimize makespan
Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers

Principle: Local optimization of current makespan
Best-Balance Algorithm - BBA

Homogeneous communication links
Homogeneous workers

Principle: Local optimization of current makespan
Fully homogeneous platforms

Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers

Principle: Local optimization of current makespan
Fully homogeneous platforms

Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers

Principle: Local optimization of current makespan
Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers

Principle: Local optimization of current makespan
Fully homogeneous platforms

Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers

Principle: Local optimization of current makespan
Best-Balance Algorithm - BBA

- Homogeneous communication links
- Homogeneous workers

Principle: Local optimization of current makespan
Theorem

**Best-Balance Algorithm** calculates an optimal schedule $S$ on a fully homogeneous star network.
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:
   $$d_1 \leq d_2 \leq \cdots \leq d_d$$
2. $\sigma \leftarrow \emptyset; \ t \leftarrow 0$
3. for $i := 1$ to $n$ do
4. \hspace{1em} $\sigma \leftarrow \sigma \cup \{i\}$
5. \hspace{1em} $t \leftarrow t + w_i$
6. \hspace{1em} if $t > d_i$ then
7. \hspace{2em} Find job $j$ in $\sigma$ with largest $w_j$ value
8. \hspace{2em} $\sigma \leftarrow \sigma \setminus \{j\}$
9. \hspace{2em} $t \leftarrow t - w_j$
10. end if
11. end for
## Moore-Based Binary-Search Algorithm - MBBSA

**Principle:**
- Moore’s algorithm
- Schedule within $M$
- Binary search

**Moore’s algorithm**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Order the jobs by non-decreasing deadlines: $d_1 \leq d_2 \leq \cdots \leq d_d$</td>
</tr>
<tr>
<td>2</td>
<td>$\sigma \leftarrow \emptyset$; $t \leftarrow 0$</td>
</tr>
<tr>
<td>3</td>
<td>$\textbf{for } i := 1 \textbf{ to } n \textbf{ do}$</td>
</tr>
<tr>
<td>4</td>
<td>$\sigma \leftarrow \sigma \cup {i}$</td>
</tr>
<tr>
<td>5</td>
<td>$t \leftarrow t + w_i$</td>
</tr>
<tr>
<td>6</td>
<td>$\textbf{if } t &gt; d_i \textbf{ then}$</td>
</tr>
<tr>
<td>7</td>
<td>Find job $j$ in $\sigma$ with largest $w_j$ value</td>
</tr>
<tr>
<td>8</td>
<td>$\sigma \leftarrow \sigma \setminus {j}$</td>
</tr>
<tr>
<td>9</td>
<td>$t \leftarrow t - w_j$</td>
</tr>
<tr>
<td>10</td>
<td>$\textbf{end if}$</td>
</tr>
<tr>
<td>11</td>
<td>$\textbf{end for}$</td>
</tr>
</tbody>
</table>

**Bus platforms**
- Homogeneous communication links
- Heterogeneous workers
- Makespan $M$
Determination of senders and receivers

$P_1$

$P_2$

$P_3$

$P_4$
Determination of senders and receivers

\( P_1, P_2, P_3, P_4 \)

\( t \)}}
Computation of deadlines

- $P_1$
- $P_2$
- $P_3$
- $P_4$

$t$
Scheduling step
Scheduling step
Scheduling step

P_1

P_2

P_3

P_4

0

M

\[ t \]
Scheduling step

- $P_1$
- $P_2$
- $P_3$
- $P_4$

$t$
MBBSA- Phase 4

Scheduling step

- $P_1$
- $P_2$
- $P_3$
- $P_4$
Scheduling step

P_1

P_2

P_3

P_4
Scheduling step

Bus platforms

MBBSA- Phase 4

0

M

t
MBBSA- Optimality

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).
Dealing with fully heterogeneous platforms

**Difficultly:** Who is sender, who is receiver?

\[ M = 12 \]

<table>
<thead>
<tr>
<th>Worker</th>
<th>c</th>
<th>w</th>
<th>load</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1 )</td>
<td>1</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>( P_2 )</td>
<td>8</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>( P_3 )</td>
<td>1</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>( P_4 )</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
NP-completeness

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.*
Proof: Reduction to 3-partition

Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem

Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem

\[
\begin{align*}
T+1 & \quad P \\
\frac{S}{4} & \quad E \\
x_1 & \quad P_1 \\
x_2 & \quad P_2 \\
x_3 & \quad P_3 \\
x_4 & \quad P_4 \\
x_5 & \quad P_5 \\
x_6 & \quad P_6 \\
E+S & \quad Q_1 \\
E & \quad Q_0
\end{align*}
\]
Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem

Proof: Reduction to 3-partition problem

\[ T+1 \]

\[ E \]

\[ P_1 \]

\[ E \]

\[ P_2 \]

\[ E \]

\[ P_3 \]

\[ E \]

\[ P_4 \]

\[ E \]

\[ P_5 \]

\[ E \]

\[ P_6 \]

\[ E+S \]

\[ Q_1 \]

\[ E \]

\[ Q_0 \]
Proof: Reduction to 3-partition problem

Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem
Proof: Reduction to 3-partition problem

Diagram showing a tree structure with various nodes labeled $P_0$, $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $Q_1$, and $Q_0$. The diagram illustrates the reduction to the 3-partition problem with nodes and edges representing the division of elements into sets $E$, $E+S$, and $E+2S$. The diagram includes labels such as $T+1$, $S$, $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$, and $S/4$.
Proof: Reduction to 3-partition problem

Proof: Reduction to 3-partition problem
## Impact of Heterogeneity

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. Hom.</td>
<td>simple greedy algorithm</td>
</tr>
<tr>
<td>Hom. Het.</td>
<td>complicated algorithm</td>
</tr>
<tr>
<td>Het. Hom.</td>
<td>?</td>
</tr>
<tr>
<td>Het. Het.</td>
<td>NP-strong</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. Hom.</td>
<td>simple greedy algorithm</td>
</tr>
<tr>
<td>Hom. Het.</td>
<td>complicated algorithm</td>
</tr>
<tr>
<td>Het. Hom.</td>
<td>?</td>
</tr>
<tr>
<td>Het. Het.</td>
<td>NP-strong</td>
</tr>
</tbody>
</table>

### Heuristics

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

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. Hom.</td>
<td>simple greedy algorithm</td>
</tr>
<tr>
<td>Comm. Het.</td>
<td>complicated algorithm</td>
</tr>
<tr>
<td>Het. Hom.</td>
<td>?</td>
</tr>
<tr>
<td>Het. Het.</td>
<td>NP-strong</td>
</tr>
</tbody>
</table>

Heuristics

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

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Comm. Hom.</td>
<td>simple greedy algorithm</td>
</tr>
<tr>
<td>Hom. Het.</td>
<td>complicated algorithm</td>
</tr>
<tr>
<td>Het. Hom.</td>
<td>?</td>
</tr>
<tr>
<td>Het. Het.</td>
<td>NP-strong</td>
</tr>
</tbody>
</table>

Heuristics

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

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hom. Hom.</td>
<td>simple greedy algorithm</td>
</tr>
<tr>
<td>Hom. Het.</td>
<td>complicated algorithm</td>
</tr>
<tr>
<td>Het. Hom.</td>
<td>?</td>
</tr>
<tr>
<td>Het. Het.</td>
<td>NP-strong</td>
</tr>
</tbody>
</table>

Heuristics

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

1. Target problem
   - Fully homogeneous platforms
   - Bus platforms
   - General platforms

2. Simulations

3. Divisible Loads Using the Multiport Switch-Model

4. Conclusion
SimGrid:
Simulator for distributed applications

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

\[- c_i: 1..100 \]
\[- w_i: 1..100 \]
\[- L_i: 0..50 \]
Simulations

SimGrid:
Simulator for distributed applications

- 4 platform types
- 1000 instances
- 10 workers
- Random variables
  - $c_i$: 1..100
  - $w_i$: 1..100
  - $L_i$: 0..50
Distance from the Best Heuristic

Heterogeneous platform

Distance [%] from the best

accumulated frequency [%]

BBA
50% of BBA schedules differ less than 8% from the best heuristic.
Distance from the Best Heuristic

Heterogeneous platform

![Graph showing distance from the best heuristic for MBBSA on a heterogeneous platform. The x-axis represents distance from the best [%], and the y-axis represents accumulated frequency [%]. The graph demonstrates the performance of MBBSA compared to the best heuristic.]
Distance from the Best Heuristic

Heterogeneous platform

- Distance from the Best Heuristic
- Accumulated frequency [%]
- Distance [%] from the best
Distance from the Best Heuristic

Heterogeneous platform

- BBA
- MBBSA
- R-BSA
## Standard Deviation

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
</tr>
<tr>
<td>Hom Hom</td>
<td>0</td>
</tr>
<tr>
<td>Hom Het</td>
<td>0.0006</td>
</tr>
<tr>
<td>Het Hom</td>
<td>0.4007</td>
</tr>
<tr>
<td>Het Het</td>
<td>0.3516</td>
</tr>
</tbody>
</table>
Standard Deviation

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Comm.</th>
<th>Component</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
<td>MBBSA</td>
<td>R-BSA</td>
</tr>
<tr>
<td>Hom</td>
<td>Hom</td>
<td>0</td>
<td>0.0107</td>
</tr>
<tr>
<td>Hom</td>
<td>Het</td>
<td>0.0006</td>
<td>0.0181</td>
</tr>
<tr>
<td>Het</td>
<td>Hom</td>
<td>0.4007</td>
<td>0.0173</td>
</tr>
<tr>
<td>Het</td>
<td>Het</td>
<td>0.3516</td>
<td>0.0284</td>
</tr>
</tbody>
</table>
## Standard Deviation

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
</tr>
<tr>
<td>Hom Hom</td>
<td>0</td>
</tr>
<tr>
<td>Hom Het</td>
<td>0.0006</td>
</tr>
<tr>
<td>Het Hom</td>
<td>0.4007</td>
</tr>
<tr>
<td>Het Het</td>
<td>0.3516</td>
</tr>
</tbody>
</table>
## Standard Deviation

<table>
<thead>
<tr>
<th>Platform type</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
</tr>
<tr>
<td>Hom Hom</td>
<td>0</td>
</tr>
<tr>
<td>Hom Het</td>
<td>0.0006</td>
</tr>
<tr>
<td>Het Hom</td>
<td>0.4007</td>
</tr>
<tr>
<td>Het Het</td>
<td>0.3516</td>
</tr>
</tbody>
</table>
Outline

1. Target problem
   - Fully homogeneous platforms
   - Bus platforms
   - General platforms

2. Simulations

3. Divisible Loads Using the Multiport Switch-Model

4. Conclusion
Framework

- Switch as master
- $m$ workers
- Computation speed $s_i$
- Bandwidth $b_i$
- Divisible load $\alpha_i$
- Linear cost model
- Overlapped unbounded switch model
Goal: Every worker finishes at the same time
Redistribution Strategy

Goal: Every worker finishes at the same time

Diagram showing redistribution among workers $P_1$, $P_2$, $P_3$, $P_4$, and $P_5$. The goal is to ensure all workers finish at the same time $T_{new}$. The diagram uses different colors to indicate the tasks assigned to each worker over time $T$. 
Solution for Divisible Loads

Imbalance of a worker $\delta_i$

**Linear program**

Minimize $T$,  
under the constraints

\[
\begin{align*}
(1a) & \quad |\delta_i| \leq T \times b_i \\
(1b) & \quad \delta_i \geq \alpha_i - T \times s_i \\
(1c) & \quad \sum_i \delta_i = 0
\end{align*}
\]

(1)

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 $\gamma_{i,j}$

\[
\gamma_{i,j} = \frac{f_{i,j}}{T_0}
\]
Solution for Divisible Loads

Imbalance of a worker $\delta_i$

Linear program

\[
\begin{align*}
\text{Minimize } & \quad T, \\
\text{under the constraints} & \\
(1a) & \quad |\delta_i| \leq T \times b_i \\
(1b) & \quad \delta_i \geq \alpha_i - T \times s_i \\
(1c) & \quad \sum_i \delta_i = 0 \\
\end{align*}
\]

(1)

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 $\gamma_{i,j}$

\[ \gamma_{i,j} = \frac{f_{i,j}}{T_0} \]
Outline

1 Target problem
   - Fully homogeneous platforms
   - Bus platforms
   - General platforms

2 Simulations

3 Divisible Loads Using the Multiport Switch-Model

4 Conclusion
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
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
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
Assess the impact of new architectural characteristics

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

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