Scheduling multi-task applications on heterogeneous platforms

Anne Benoit, Jean-François Pineau, Yves Robert and Frédéric Vivien

Laboratoire de l’Informatique du Parallélisme
École Normale Supérieure de Lyon, France

Jean-François.Pineau@ens-lyon.fr

http://graal.ens-lyon.fr/~jfpineau

GDT GRAAL
July 5, 2007
Outline

1. Framework
2. Theoretical study
   - Steady state scheduling
   - Off-line study
   - Extension
3. Experiments
4. Conclusion
Bag-of-tasks Applications

Bag of tasks described by:

- the number of tasks
- the amount of computation of a task
- the amount of communication of a task
- their release date

On-line scheduling.
Bag-of-tasks Applications

Bag of tasks described by:

- the number of independent tasks
- the amount of computation of a task
- the amount of communication of a task
- their release date

On-line scheduling.
Bag of tasks

described by:

- the number of independent, identical tasks
- the amount of computation of a task
- the amount of communication of a task
- their release date

On-line scheduling.
Bag-of-tasks Applications

Bag of tasks described by:

- the number of *independent, identical* tasks
- the amount of computation of a task
- the amount of communication of a task
- their release date

On-line scheduling.
Platform model

Master

Network

Links

Slaves

Tasks
### Master-slaves platform

#### The master
- Receive the bags of tasks
- Send the tasks to the processors
- Bounded multi-port model

#### The processors
- Parallels
  - Identical
  - Uniform
- Related
Master-slaves platform

The master
- Receive the bags of tasks
- Send the tasks to the processors
- Bounded multi-port model

The processors
- Parallels
  - Identical
  - Uniform
- Related
Notations

Tasks

- $n$ bags-of-tasks applications $A_k$
- $A_i$ is composed of $\Pi^{(i)}$ tasks.
- $w^{(i)}$: amount of computation of a task of $A_i$
- $\delta^{(i)}$: amount of communication of a task of $A_i$
- $r^{(i)}$: release date of $A_i$
- $C^{(i)}$: completion time of $A_i$
Notations

Platform

- $p$ processors,
- $B$: bound of the multi-port model.
- $b_u$: bandwidth of the link between the master and $P_u$,
- $s_u$: computational speed of worker $P_u$, 
Notations

Platform

- $p$: processors,
- $\mathcal{B}$: bound of the multi-port model.
- $b_u$: bandwidth of the link between the master and $P_u$.
- $s_u^{(k)}$: computational speed of related worker $P_u$ with tasks of $A_k$. 
Objective

Scheduling the tasks to the processors in order to process this tasks

- according to the constraints,
  - of the processors
  - of the tasks

- optimizing an objective function
Objective function

- Makespan

\[ \max C^{(i)} \text{ or } C^{(\max)} \]
Objective function

- Makespan

\[
\max C^{(i)} \text{ or } C^{(\text{max})}
\]

*Problem of satisfaction of the clients*
Objective function

- Makespan
- Sum flow

\[ \sum \{ C(i) - r(i) \} \]
Objective function

- Makespan
- Sum flow

\[
\sum \{C^{(i)} - r^{(i)}\}
\]

Problem of starvation
Objective function

- Makespan
- Sum flow
- Max flow

\[
\max \{ C(i) - r(i) \}
\]
Objective function

- Makespan
- Sum flow
- Max flow

\[ \max \{ C(i) - r(i) \} \]

*Small applications can wait a long time*
Objective function

- Makespan
- Sum-flow
- Max-flow
- Max Stretch

\[
\max \frac{C(i) - r(i)}{\text{Size of } A_i}
\]
Objective function

- **Makespan**
- **Sum flow**
- **Max flow**
- **Max Stretch**

\[
\text{max } \frac{C(i) - r(i)}{\text{Size of } A_i}
\]

Size of \( A_i = \Pi^{(i)} \) ?
## Objective function

- **Makespan**
- **Sum flow**
- **Max flow**
- **Max Stretch**

\[
\text{max } \frac{C(i) - r(i)}{\text{Size of } A_i}
\]

\[\text{Size of } A_i = w^{(i)} \]
Objective function

- Makespan
- Sum flow
- Max flow
- Max Stretch

\[
\max \frac{C(i) - r(i)}{\text{Size of } A_i}
\]

\[
\text{Size of } A_i = \prod(i) \ast w(i)
\]
Outline

1. Framework

2. Theoretical study
   - Steady state scheduling
   - Off-line study
   - Extension

3. Experiments

4. Conclusion
Simple problem

**Problem**

- Unique bag-of-tasks $A_0$
- Large $\Pi^{(0)}$
Simple problem

**Problem**
- Unique bag-of-tasks $A_0$
- Large $\Pi^{(0)}$

**Objective**
- Minimizing the makespan
Simple problem

Problem
- Unique bag-of-tasks $A_0$
- Large $\Pi^{(0)}$

Objective
- Minimizing the makespan
- Maximizing the throughput
Simple problem

**Problem**
- Unique bag-of-tasks $A_0$
- Large $\Pi^{(0)}$

**Objective**
- Minimizing the makespan
- Maximizing the throughput
- Throughput of worker $P_u$: $\rho_u^{*(0)}$
- Total throughput $\rho^{*(0)} = \sum_{u=1}^{p} \rho_u^{*(0)}$
Linear program

\[
\begin{align*}
\text{MAXIMIZE} & \quad \rho^*(0) = \sum_{u=1}^{p} \rho_u^*(0) \\
\text{SUBJECT TO} & \\
\rho_u^*(0) \frac{w^{(0)}}{s_u^{(0)}} & \leq 1 \\
\rho_u^*(0) \frac{\delta^{(0)}}{b_u} & \leq 1 \\
\sum_{u=1}^{p} \rho_u^*(0) \frac{\delta^{(0)}}{\mathcal{B}} & \leq 1
\end{align*}
\]

(1)
Linear program

\[
\begin{align*}
\text{MAXIMIZE} & \quad \rho^*(0) = \sum_{u=1}^{p} \rho_u^*(0) \\
\text{SUBJECT TO} & \\
\rho_u^*(0) \frac{w(0)}{s_u(0)} & \leq 1 \\
\rho_u^*(0) \frac{\delta(0)}{b_u} & \leq 1 \\
\sum_{u=1}^{p} \rho_u^*(0) \frac{\delta(0)}{B} & \leq 1
\end{align*}
\]

Rational solution
Resource selection \( \rho^*_u(0) = 0 \)

Master sends tasks to workers using the 1D-load balancing algorithm:

- the first worker to receive a task is the one with largest throughput
- each participating worker \( P_u \) has already received \( n_u \) tasks, the next worker to receive a task is chosen as the one minimizing

\[
\frac{n_u + 1}{\rho^*_u(0)}
\]
Resource selection ($\rho_u^*(0) = 0$)
Master sends tasks to workers using the 1D-load balancing algorithm:

- the first worker to receive a task is the one with largest throughput
- each participating worker $P_u$ has already received $n_u$ tasks, the next worker to receive a task is chosen as the one minimizing

$$\frac{n_u + 1}{\rho_u^*(0)}$$
Feasible schedule

Resource selection \((\rho_u^{*}(0) = 0)\)

Master sends tasks to workers using the 1D-load balancing algorithm:

- the first worker to receive a task is the one with largest throughput

- each participating worker \(P_u\) has already received \(n_u\) tasks, the next worker to receive a task is chosen as the one minimizing

\[
\frac{n_u + 1}{\rho_u^{*}(0)}
\]
Feasible schedule

Resource selection \((\rho_u^{*(0)} = 0)\)

Master sends tasks to workers using the 1D-load balancing algorithm:

- the first worker to receive a task is the one with largest throughput
- each participating worker \(P_u\) has already received \(n_u\) tasks, the next worker to receive a task is chosen as the one minimizing

\[
\frac{n_u + 1}{\rho_u^{*(0)}}
\]
Approximation of the best execution time:

\[ MS^*(k) = \frac{\Pi(k)}{\rho^*(k)}. \]

Real execution time:

\[ C^{(k)} = r^{(k)} + MS^{(k)} \]

In general:

\[ MS^{(k)} \geq MS^*(k) \]
Approximation of the best execution time:

\[ MS^*(k) = \frac{\prod(k)}{\rho^*(k)}. \]

Real execution time:

\[ C^{(k)} = r^{(k)} + MS^{(k)} \]

In general:

\[ MS^{(k)} \geq MS^*^{(k)} \]
Back on multi-applications problem

Approximation of the best execution time:

\[ MS^*(k) = \frac{\Pi^{(k)}}{\rho^*(k)}. \]

Real execution time:

\[ C^{(k)} = r^{(k)} + MS^{(k)} \]

In general:

\[ MS^{(k)} \geq MS^*(k) \]
Stretch:

\[ S^k = \frac{MS^{(k)}}{MS^{*}(k)} \]

Throughput \( \rho^{(k)} \) defined by:

\[ MS^{(k)} = \frac{\Pi^{(k)}}{\rho^{(k)}}. \]

**Objective**: max-stretch:

\[ S = \max_{1 \leq k \leq n} S^k \]
Stretch:

\[ S^k = \frac{MS^{(k)}}{MS^{*}(k)} = \frac{\rho^{*}(k)}{\rho(k)} \]

Throughput \( \rho^{(k)} \) defined by:

\[ MS^{(k)} = \frac{\Pi^{(k)}}{\rho(k)}. \]

**Objective:** max-stretch:

\[ S = \max_{1 \leq k \leq n} S^k \]
Stretch:

\[ S^k = \frac{MS^{(k)}}{MS^{\ast}(k)} = \frac{\rho^{\ast}(k)}{\rho(k)} \]

Throughput \( \rho^{(k)} \) defined by:

\[ MS^{(k)} = \frac{\prod^{(k)}}{\rho(k)}. \]

**Objective**: max-stretch:

\[ S = \max_{1 \leq k \leq n} S^k \]
Outline

1. Framework

2. Theoretical study
   - Steady state scheduling
   - Off-line study
   - Extension

3. Experiments

4. Conclusion
Off-line

- Computing all the $MS^*(k)$, $\forall 1 \leq k \leq n$
- Binary search on the max-stretch
- For each candidate value $S'$, we know that:

\[
\forall 1 \leq k \leq n, \quad \frac{MS(k)}{MS^*(k)} \leq S'
\]

\[
\forall 1 \leq k \leq n, \quad C(k) = r(k) + MS(k) \leq r(k) + S' \times MS^*(k)
\]
Off-line

- Computing all the $MS^*(k)$, $\forall 1 \leq k \leq n$
- Binary search on the max-stretch
  - For each candidate value $S'$, we know that:

$$\forall 1 \leq k \leq n, \frac{MS(k)}{MS^*(k)} \leq S'$$

$$\forall 1 \leq k \leq n, C^{(k)} = r^{(k)} + MS^{(k)} \leq r^{(k)} + S' \times MS^*(k)$$
Computing all the $MS^*(k)$, $\forall 1 \leq k \leq n$

Binary search on the max-stretch

For each candidate value $S'$, we know that:

$$\forall 1 \leq k \leq n, \quad \frac{MS(k)}{MS^*(k)} \leq S'$$

$$\forall 1 \leq k \leq n, \quad C^{(k)} = r^{(k)} + MS^{(k)} \leq r^{(k)} + S' \times MS^*(k)$$
Off-line

- Computing all the $MS^*(k)$, $\forall 1 \leq k \leq n$
- Binary search on the max-stretch
- For each candidate value $S'$, we know that:

$$\forall 1 \leq k \leq n, \quad \frac{MS(k)}{MS^*(k)} \leq S'$$

$$\forall 1 \leq k \leq n, \quad C^{(k)} = r^{(k)} + MS^{(k)} \leq r^{(k)} + S' \times MS^*(k)$$
Deadlines

We set:

\[ d^{(k)} = r^{(k)} + S' \times MS^*(k) \]  \hspace{1cm} (2)

**Definition:** Epochal times

\[ t^{(j)} \in \{ r^{(1)}, \ldots, r^{(n)} \} \cup \{ d^{(1)}, \ldots, d^{(n)} \} \]

such that

\[ t^{(j)} \leq t^{(j+1)}, \ 1 \leq j \leq 2n - 1 \]

Divide the total execution time into intervals whose bounds are epochal times.
We set:

\[ d^{(k)} = r^{(k)} + S' \times MS^*(k) \] (2)

**Definition:** Epochal times

\[ t^{(j)} \in \{r^{(1)}, ..., r^{(n)}\} \cup \{d^{(1)}, ..., d^{(n)}\} \]

, such that

\[ t^{(j)} \leq t^{(j+1)}, \quad 1 \leq j \leq 2n - 1 \]

Divide the total execution time into intervals whose bounds are epochal times.
We set:

\[ d^{(k)} = r^{(k)} + S' \times MS^*(k) \]  

(2)

**Definition:** Epochal times

\[ t^{(j)} \in \{ r^{(1)}, ... , r^{(n)} \} \cup \{ d^{(1)}, ... , d^{(n)} \} \]

, such that

\[ t^{(j)} \leq t^{(j+1)}, \ 1 \leq j \leq 2n - 1 \]

Divide the total execution time into intervals whose bounds are epochal times.
run each application $A_k$ during its whole execution window $[r(k), d(k)]$, use a different throughput on each interval $[t(j), t(j+1)]$, $r(k) \leq t(j)$ and $t(j+1) \leq d(k)$.

Notation:

- $\rho_u^{(k)}(j)$: throughput achieved by $A_k$ during interval $[t(j), t(j+1)]$ on processor $P_u$
- $\rho^{(k)}(j)$: global throughput of $A_k$ during this period.
Intervals

- run each application $A_k$ during its whole execution window $[r^{(k)}, d^{(k)}]$.
- use a different throughput on each interval $[t^{(j)}, t^{(j+1)}]$, $r^{(k)} \leq t^{(j)}$ and $t^{(j+1)} \leq d^{(k)}$.

Notation:
- $\rho_u^{(k)}(j)$: throughput achieved by $A_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor $P_u$
- $\rho^{(k)}(j)$: global throughput of $A_k$ during this period.
run each application $A_k$ during its whole execution window $[r^{(k)}, d^{(k)}]$,

use a different throughput on each interval $[t^{(j)}, t^{(j+1)}]$, $r^{(k)} \leq t^{(j)}$ and $t^{(j+1)} \leq d^{(k)}$.

Notation:

- $\rho^{(k)}_u(j)$: throughput achieved by $A_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor $P_u$
- $\rho^{(k)}(j)$: global throughput of $A_k$ during this period.
run each application $A_k$ during its whole execution window $[r^{(k)}, d^{(k)}]$.

use a different throughput on each interval $[t^{(j)}, t^{(j+1)}]$, $r^{(k)} \leq t^{(j)}$ and $t^{(j+1)} \leq d^{(k)}$.

Notation:

- $\rho_u^{(k)}(j)$: throughput achieved by $A_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor $P_u$
- $\rho^{(k)}(j)$: global throughput of $A_k$ during this period.
Linear program

\[
\forall 1 \leq k \leq n, \quad \sum_{[t(j), t(j+1)]} \rho^{(k)}(j) \times (t(j+1) - t(j)) = \Pi^{(k)}
\]

\[
\forall 1 \leq k \leq n, \forall 1 \leq j \leq 2n - 1, \rho^{(k)}(j) = \sum_{u=1}^{p} \rho^{(k)}_u(j)
\]

\[
\forall 1 \leq j \leq 2n - 1, \forall 1 \leq u \leq p, \sum_{k=1}^{n} \rho^{(k)}_u(j) \frac{w^{(k)}_u}{s^{(k)}_u} \leq 1
\]

\[
\forall 1 \leq j \leq 2n - 1, \forall 1 \leq u \leq p, \sum_{k=1}^{n} \rho^{(k)}_u(j) \frac{\delta^{(k)}_u}{b_u} \leq 1
\]

\[
\forall 1 \leq j \leq 2n - 1, \sum_{u=1}^{p} \sum_{k=1}^{n} \rho^{(k)}_u(j) \frac{\delta^{(k)}_u}{B} \leq 1
\]
Algorithm

- Computing all the $MS^*(k)$, $\forall 1 \leq k \leq n$
- Binary search on the max-stretch
- For each candidate stretch
  - compute the $t^{(j)}$
  - resolve the linear program

Theorem

The previous scheduling algorithm finds the optimal max-stretch in polynomial time.
Algorithm

- Computing all the $MS^*(k)$, $\forall 1 \leq k \leq n$
- Binary search on the max-stretch
- For each candidate stretch
  - compute the $t^{(j)}$
  - resolve the linear program

Theorem

The previous scheduling algorithm finds the optimal max-stretch in polynomial time.
Proof (1/3)

Part 1

A given max-stretch \( S' \) is achievable if and only if the linear program has a solution.

Consider an arbitrary solution that achieves \( S' \).

\[
\text{nb}(j, k, u) = \text{number of tasks for } A_k \text{ on } P_u \text{ during the interval } [t(j), t(j+1)],
\]

Averaged throughput:

\[
\bar{\rho}_u^{(k)}(j) = \frac{\text{nb}(j, k, u)}{t(j+1) - t(j)},
\]

\[
\bar{\rho}^{(k)}(j) = \sum_{u=1}^{\rho} \bar{\rho}_u^{(k)}(j).
\]
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution

Consider an arbitrary solution that achieves $S'$.

$$nb(j, k, u) = \text{number of tasks for } A_k \text{ on } P_u \text{ during the interval } [t(j), t(j+1)],$$

Averaged throughput:

$$\bar{\rho}_{u}^{(k)}(j) = \frac{nb(j, k, u)}{t(j+1) - t(j)},$$

$$\bar{\rho}^{(k)}(j) = \sum_{u=1}^{\rho} \bar{\rho}_{u}^{(k)}(j).$$
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution

Consider an arbitrary solution that achieves $S'$.

$\text{nb}(j, k, u) = \text{number of tasks for } A_k \text{ on } P_u \text{ during the interval } [t(j), t(j+1)]$,

Averaged throughput:

$$\bar{\rho}^{(k)}(j) = \frac{\text{nb}(j, k, u)}{t(j+1) - t(j)},$$

$$\bar{\rho}(k)(j) = \sum_{u=1}^{\rho} \bar{\rho}^{(k)}(j).$$
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution.

Consider an arbitrary solution that achieves $S'$.

\[ \text{nb}(j, k, u) = \text{number of tasks for } A_k \text{ on } P_u \text{ during the interval} \]
\[ [t(j), t(j+1)], \]

Averaged throughput:

\[ \bar{\rho}_u^{(k)}(j) = \frac{\text{nb}(j, k, u)}{t(j+1) - t(j)}, \]

\[ \bar{\rho}^{(k)}(j) = \sum_{u=1}^{p} \bar{\rho}_u^{(k)}(j). \]
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution

$$\{\rho_u^{(k)}(j), \rho^{(k)}(j)\}$$

are a valid solution of the linear program:
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution

The first equation is satisfied:

$$
\sum_{[t(j), t(j+1)]} \rho^{(k)}(j) \times (t^{(j+1)} - t^{(j)}) =
$$

$$
\sum_{[t(j), t(j+1)]} \sum_{u=1}^{p} \rho^{(k)}(j) \times (t^{(j+1)} - t^{(j)})
$$
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution

The first equation is satisfied:

$$\sum_{[t(j), t(j+1)]} \rho^{(k)}(j) \times (t(j+1) - t(j)) =$$

\[\sum_{[t(j), t(j+1)]} \rho^{(k)}(j) \times (t(j+1) - t(j)) = \sum_{[t(j), t(j+1)]} \text{nb}(j, k, u)\]
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution.

The first equation is satisfied:

$$
\sum_{[t(j), \ t(j+1)]} \rho^{(k)}(j) \times (t^{(j+1)} - t^{(j)}) = \Pi^{(k)}
$$
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution.

The second equation is satisfied by definition.
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution

The third equation is satisfied:

$$\sum_{k=1}^{n} \frac{\rho_{u}^{(k)}(j)}{s_{u}^{(k)}} \frac{w^{(k)}}{s_{u}^{(k)}} = \sum_{k=1}^{n} \frac{nb(j, k, u)}{t(j+1) - t(j)} \cdot \frac{w^{(k)}}{s_{u}^{(k)}}$$

But we have

$$\sum_{k=1}^{n} \frac{nb(j, k, u)}{s_{u}^{(k)}} \frac{w^{(k)}}{s_{u}^{(k)}} \leq t(j+1) - t(j)$$
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution.

The third equation is satisfied:

$$\sum_{k=1}^{n} \rho_{u}^{(k)}(j) \frac{w^{(k)}}{s_{u}^{(k)}} = \sum_{k=1}^{n} \frac{nb(j, k, u)}{t(j+1) - t(j)} \cdot \frac{w^{(k)}}{s_{u}^{(k)}}$$

But we have

$$\sum_{k=1}^{n} nb(j, k, u) \frac{w^{(k)}}{s_{u}^{(k)}} \leq t(j+1) - t(j)$$
Proof (1/3)

Part 1

A given max-stretch $S'$ is achievable if and only if the linear program has a solution.

The fourth and fifth equations are satisfied as well.

Intuitively, the result comes from the linearity of linear programs!
The linear program can be solved in polynomial time.

- $2n - 1$ intervals, so $O(n^2 + np)$ equations
- linear program over rational numbers,
- in theory using the ellipsoid method,
- in practice using standard software packages (glpk).
Proof (2/3)

Part 2

The linear program can be solved in polynomial time.

- $2n - 1$ intervals, so $O(n^2 + np)$ equations
- linear program over rational numbers,
  - in theory using the ellipsoid method,
  - in practice using standard software packages (glpk).
Part 2

The linear program can be solved in polynomial time.

- $2n - 1$ intervals, so $O(n^2 + np)$ equations
- linear program over rational numbers,
- in theory using the ellipsoid method,
- in practice using standard software packages (glpk).
Proof (2/3)

Part 2

The linear program can be solved in polynomial time.

- $2n - 1$ intervals, so $O(n^2 + np)$ equations
- linear program over rational numbers,
- in theory using the ellipsoid method,
- in practice using standard software packages (glpk).
Part 3

The binary search needs polynomial number of iterations.
Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

- $S^1, S^2$: given max-stretch
- $\forall S' \in [S^1, S^2]$, the order of the $t^{(i)}$ does not change
- $t^{(i)} \leftarrow t^{(i)}(S')$
The binary search needs polynomial number of iterations.

- $S^1, S^2$: given max-stretch
- $\forall S' \in [S^1, S^2]$, the order of the $t^{(i)}$ does not change
- $t^{(i)} \leftarrow t^{(i)}(S')$
Part 3

The binary search needs polynomial number of iterations.

- $S^1, S^2$: given max-stretch
- $\forall S' \in [S^1, S^2]$, the order of the $t^{(i)}$ does not change
- $t^{(i)} \leftarrow t^{(i)}(S')$
Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

New linear program:

\[
\begin{align*}
\text{Minimize } & S' \\
\text{subject to } & S^1 \leq S' \leq S^2 \\
& \forall 1 \leq k \leq n, \\
& \sum_{[t(j)(S'), t(j+1)(S')]} \rho^{(k)}(j) \times (t^{(j+1)}(S') - t^{(j)}(S')) = \Pi^{(k)} \\
& t^{(j)}(S') \geq r^{(k)} \\
& t^{(j+1)}(S') \leq d^{(k)}(S') \\
\end{align*}
\]
Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

The modified linear program has a solution if and only if a max-stretch $S' \in [S^1, S^2]$ is achievable.

At most $n(n - 1)$ stretch intervals
Part 3

The binary search needs polynomial number of iterations.

The modified linear program has a solution if and only if a max-stretch $S' \in [S^1, S^2]$ is achievable.

At most $n(n - 1)$ stretch intervals.
Outline

1. Framework

2. Theoretical study
   - Steady state scheduling
   - Off-line study
   - Extension

3. Experiments

4. Conclusion
Off-line algorithm at each release dates:

1. For each application $A_k$, count the number of tasks (if any) that have been executed
2. update $\Pi^{(k)}$
3. update $MS^*(k)$
4. determine the new optimal stretch that can be achieved as in the off-line case
Off-line algorithm at each release dates:

- For each application $A_k$, count the number of tasks (if any) that have been executed
- update $\Pi^{(k)}$
- update $MS^*(k)$
- determine the new optimal stretch that can be achieved as in the off-line case
On-line

Off-line algorithm at each release dates:

- For each application $A_k$, count the number of tasks (if any) that have been executed
- update $\Pi^{(k)}$
- update $MS^{*(k)}$
- determine the new optimal stretch that can be achieved as in the off-line case
Off-line algorithm at each release dates:

- For each application $A_k$, count the number of tasks (if any) that have been executed
- update $\Pi^{(k)}$
- update $MS^{\ast}(k)$
- determine the new optimal stretch that can be achieved as in the off-line case
Multi-level trees

- **Resource constraints unchanged**
- conservation law stating that for each application $A_k$ for each internal node

One-port model: previous constraint:

$$\sum_{u=1}^{p} \sum_{k=1}^{n} \rho_{u}^{(k)} \frac{\delta^{(k)}}{\beta} \leq 1$$

Mixed-implementation of the two previous models

Return messages: for each application $A_k$

$$\delta^{(k)} \leftarrow \delta^{(k)} + \text{return}^{(k)}$$
Extension

Multi-level trees

- Resource constraints unchanged
- Conservation law stating that for each application $A_k$ for each internal node

One-port model: previous constraint:

$$\sum_{u=1}^{p} \sum_{k=1}^{n} \rho_u^{(k)} \delta^{(k)} \frac{\delta^{(k)}}{B} \leq 1$$

Mixed-implementation of the two previous models

Return messages: for each application $A_k$

$$\delta^{(k)} \leftarrow \delta^{(k)} + \text{return}^{(k)}$$
Multi-level trees

- Resource constraints unchanged
- Conservation law stating that for each application $A_k$ for each internal node

One-port model: previous constraint:

$$
\sum_{u=1}^{p} \sum_{k=1}^{n} \rho_{u}^{(k)} \frac{\delta^{(k)}}{B} \leq 1
$$

Mixed-implementation of the two previous models

Return messages: for each application $A_k$

$$
\delta^{(k)} \leftarrow \delta^{(k)} + \text{return}^{(k)}
$$
Multi-level trees

- Resource constraints unchanged
- Conservation law stating that for each application $A_k$ for each internal node

One-port model: new constraint:

$$
\sum_{u=1}^{p} \sum_{k=1}^{n} \rho_u^{(k)} \frac{\delta^{(k)}}{b_u} \leq 1.
$$

Mixed-implementation of the two previous models

Return messages: for each application $A_k$

$$
\delta^{(k)} \leftarrow \delta^{(k)} + \text{return}^{(k)}
$$
Extension

Multi-level trees

- Resource constraints unchanged
- Conservation law stating that for each application $A_k$ for each internal node

One-port model: new constraint:

$$\sum_{u=1}^{p} \sum_{k=1}^{n} \rho_u^{(k)} \frac{\delta^{(k)}}{b_u} \leq 1.$$ 

Mixed-implementation of the two previous models

Return messages: for each application $A_k$

$$\delta^{(k)} \leftarrow \delta^{(k)} + \text{return}^{(k)}$$
Multi-level trees

- Resource constraints unchanged
- Conservation law stating that for each application $A_k$ for each internal node

One-port model: new constraint:

$$\sum_{u=1}^{p} \sum_{k=1}^{n} \rho_u^{(k)} \delta^{(k)} \frac{\delta^{(k)}}{b_u} \leq 1.$$  

Mixed-implementation of the two previous models

Return messages: for each application $A_k$

$$\delta^{(k)} \leftarrow \delta^{(k)} + \text{return}^{(k)}$$
Outline

1. Framework
2. Theoretical study
   - Steady state scheduling
   - Off-line study
   - Extension
3. Experiments
4. Conclusion
The platform

Hardware

Computers of the GDSDMI cluster:

- 8 SuperMicro servers 5013-GM, with processors P4 2.4 GHz;
- 5 SuperMicro servers 6013PI, with processors P4 Xeon 2.4 GHz;
- 7 SuperMicro servers 5013SI, with processors P4 Xeon 2.6 GHz;
- 7 SuperMicro servers IDE250W, with processors P4 2.8 GHz.
- 100Mbps Fast-Ethernet switch
The tasks

Software
- MPI communications
- Modification of slave parameters

Tasks
Computation of matrices product

The linear programs are solved using glpk.
The studied algorithms

- FIFO + Round-Robin
- FIFO + MCT
- S(R)PT + MCT
- S(R)PT + Demand-Driven
- Steady-state MWMA (Master Worker Multi-applications) on each time interval
- CBSSSM (Clever Burst Steady-State Stretch Minimizing)
Eh wait!
You don’t have any result yet!!
1 Framework

2 Theoretical study
  - Steady state scheduling
  - Off-line study
  - Extension

3 Experiments

4 Conclusion
Conclusion

Key points:
- Realistic platform model
- Optimal off-line algorithm
- On-line algorithm
Conclusion

- **Key points:**
  - Realistic platform model
  - Optimal off-line algorithm
  - On-line algorithm

- **Extensions:**
  - Have some experimental results
  - Consider other objective functions