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-Francois.Pineau@ens-lyon.fr

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

GDT GRAAL
July 5, 2007

Outline

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 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.

Bag-of-tasks Applications

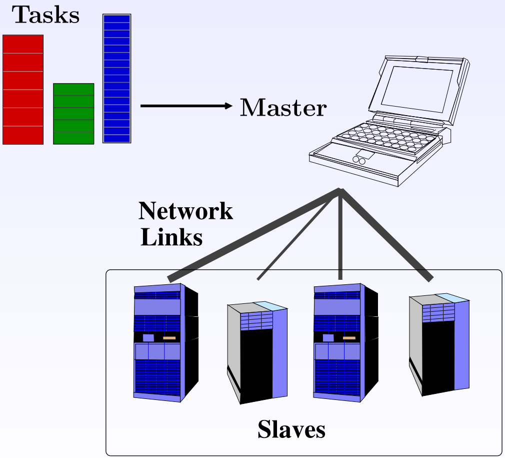
Bag of tasks

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-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 $\mathcal{A}_k$
- $\mathcal{A}_i$ is composed of $\Pi^{(i)}$ tasks.
- $w^{(i)}$: amount of computation of a task of $\mathcal{A}_i$
- $\delta^{(i)}$: amount of communication of a task of $\mathcal{A}_i$
- $r^{(i)}$: release date of $\mathcal{A}_i$
- $\mathcal{C}^{(i)}$: completion time of $\mathcal{A}_i$

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: 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 $\mathcal{A}_k$,

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

Objective function

- Makespan

$$\max \ \mathcal{C}^{(i)} \text{ or } \mathcal{C}^{(max)}$$

Objective function

Objective function

- Makespan

$$\max \ \mathcal{C}^{(i)} \text{ or } \mathcal{C}^{(max)}$$

Problem of satisfaction of the clients

Objective function

Objective function

- ~~Makespan~~
- Sum flow

$$\sum \{\mathcal{C}^{(i)} - r^{(i)}\}$$

Objective function

Objective function

- ~~Makespan~~
- Sum flow

$$\sum \{\mathcal{C}^{(i)} - r^{(i)}\}$$

Problem of starvation

Objective function

Objective function

- ~~Makespan~~
- ~~Sum flow~~
- Max flow

$$\max \{\mathcal{C}^{(i)} - r^{(i)}\}$$

Objective function

Objective function

- ~~Makespan~~
- ~~Sum flow~~
- ~~Max flow~~

$$\max \left\{ \mathcal{C}^{(i)} - r^{(i)} \right\}$$

Small applications can wait a long time

Objective function

Objective function

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

$$\max \frac{\mathcal{C}^{(i)} - r^{(i)}}{Size\ of\ \mathcal{A}_i}$$

Objective function

Objective function

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

$$\max \frac{\mathcal{C}^{(i)} - r^{(i)}}{\textit{Size of } \mathcal{A}_i}$$

$\textit{Size of } \mathcal{A}_i = \Pi^{(i)}$?

Objective function

Objective function

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

$$\max \frac{\mathcal{C}^{(i)} - r^{(i)}}{Size\ of\ \mathcal{A}_i}$$

$Size\ of\ \mathcal{A}_i = w^{(i)}$?

Objective function

Objective function

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

$$\max \frac{\mathcal{C}^{(i)} - r^{(i)}}{\textit{Size of } \mathcal{A}_i}$$

$\textit{Size of } \mathcal{A}_i = \Pi^{(i)} * w^{(i)}$?

Outline

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

Simple problem

Problem

- Unique bag-of-tasks $\mathcal{A}_0$
- Large $\Pi^{(0)}$

Simple problem

Problem

- Unique bag-of-tasks $\mathcal{A}_0$
- Large $\Pi^{(0)}$

Objective

- Minimizing the makespan

Simple problem

Problem

- Unique bag-of-tasks $\mathcal{A}_0$
- Large $\Pi^{(0)}$

Objective

- ~~Minimizing the makespan~~
- Maximizing the throughput

Simple problem

Problem

- Unique bag-of-tasks $\mathcal{A}_0$
- 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

$$\left\{\begin{array}{l} \text{MAXIMIZE } \rho^{*(0)} = \sum_{u=1}^{p} \rho_u^{*(0)} \\ \text{SUBJECT TO} \\ \rho_u^{*(0)} \frac{w^{(0)}}{s_u^{(0)}} \leq 1 \\ \rho_u^{*(0)} \frac{\delta^{(0)}}{b_u} \leq 1 \\ \sum_{u=1}^{p} \rho_u^{*(0)} \frac{\delta^{(0)}}{\mathcal{B}} \leq 1 \end{array}\right. \tag{1}$$

Rational solution

Linear program

$$\left\{\begin{array}{l}
\textsc{Maximize } \rho^{*(0)} = \sum_{u=1}^{p} \rho_u^{*(0)} \\
\text{SUBJECT TO} \\
\rho_u^{*(0)} \frac{w^{(0)}}{s_u^{(0)}} \leq 1 \\
\rho_u^{*(0)} \frac{\delta^{(0)}}{b_u} \leq 1 \\
\sum_{u=1}^{p} \rho_u^{*(0)} \frac{\delta^{(0)}}{\mathcal{B}} \leq 1
\end{array}\right. \tag{1}$$

Rational solution

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

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

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

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

Back on multi-applications problem

Approximation of the best execution time:

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

Real execution time:

$$\mathcal{C}^{(k)} = r^{(k)} + MS^{(k)}$$

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:

$$\mathcal{C}^{(k)} = r^{(k)} + MS^{(k)}$$

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:

$$\mathcal{C}^{(k)} = r^{(k)} + MS^{(k)}$$

In general:

$$MS^{(k)} \geq MS^{*(k)}$$

Stretch

Stretch:

$$\mathcal{S}^k = \frac{MS^{(k)}}{MS^{*(k)}}$$

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

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

Objective: max-stretch:

$$\mathcal{S} = \max_{1 \leq k \leq n} \mathcal{S}^k$$

Stretch

Stretch:

$$\mathcal{S}^k = \frac{MS^{(k)}}{MS^{*(k)}} = \frac{\rho^{*(k)}}{\rho^{(k)}}$$

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

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

Objective: max-stretch:

$$\mathcal{S} = \max_{1 \leq k \leq n} \mathcal{S}^k$$

Stretch

Stretch:

$$\mathcal{S}^k = \frac{MS^{(k)}}{MS^{*(k)}} = \frac{\rho^{*(k)}}{\rho^{(k)}}$$

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

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

Objective: max-stretch:

$$\mathcal{S} = \max_{1 \le k \le n} \mathcal{S}^k$$

Outline

Off-line

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

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

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

Off-line

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

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

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

Off-line

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

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

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

Off-line

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

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

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

Deadlines

We set:

$$d^{(k)} = r^{(k)} + \mathcal{S}' \times MS^{*(k)} \tag{2}$$

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.

Deadlines

We set:

$$d^{(k)} = r^{(k)} + \mathcal{S}' \times MS^{*(k)} \tag{2}$$

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.

Deadlines

We set:

$$d^{(k)} = r^{(k)} + \mathcal{S}' \times MS^{*(k)} \tag{2}$$

Definition: Epochal times

$$t^{(j)} \in \{r^{(1)}, \dots, r^{(n)}\} \cup \{d^{(1)}, \dots, d^{(n)}\}$$

, 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.

Intervals

- run each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor P_u
- $\rho^{(k)}(j)$: global throughput of $\mathcal{A}_k$ during this period.

Intervals

- run each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor P_u
- $\rho^{(k)}(j)$: global throughput of $\mathcal{A}_k$ during this period.

Intervals

- run each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor P_u
- $\rho^{(k)}(j)$: global throughput of $\mathcal{A}_k$ during this period.

Intervals

- run each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$ during interval $[t^{(j)}, t^{(j+1)}]$ on processor P_u
- $\rho^{(k)}(j)$: global throughput of $\mathcal{A}_k$ during this period.

Linear program

$$\begin{cases} \forall\ 1 \le k \le n, \displaystyle\sum_{\substack{[t^{(j)},\ t^{(j+1)}] \\ t^{(j)}\ \ge\ r^{(k)} \\ t^{(j+1)}\ \le\ d^{(k)}}} \rho^{(k)}(j) \times (t^{(j+1)} - t^{(j)}) = \Pi^{(k)} \\ \forall\ 1 \le k \le n, \forall\ 1 \le j \le 2n-1, \rho^{(k)}(j) = \displaystyle\sum_{u=1}^{p} \rho_u^{(k)}(j) \\ \forall\ 1 \le j \le 2n-1, \forall\ 1 \le u \le p, \displaystyle\sum_{k=1}^{n} \rho_u^{(k)}(j) \frac{w^{(k)}}{s_u^{(k)}} \ \le\ 1 \\ \forall\ 1 \le j \le 2n-1, \forall\ 1 \le u \le p, \displaystyle\sum_{k=1}^{n} \rho_u^{(k)}(j) \frac{\delta^{(k)}}{b_u} \ \le\ 1 \\ \forall\ 1 \le j \le 2n-1, \displaystyle\sum_{u=1}^{p}\sum_{k=1}^{n} \rho_u^{(k)}(j) \frac{\delta^{(k)}}{\mathcal{B}} \ \le\ 1 \end{cases} \tag{3}$$

Algorithm

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

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 $\mathcal{S}'$ is achievable if and only if the linear program has a solution

Consider an arbitrary solution that achieves $\mathcal{S}'$.

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

Averaged throughput:

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

$$\overline{\rho}^{(k)}(j) = \sum_{u=1}^{p} \overline{\rho}_u^{(k)}(j).$$

Proof (1/3)

Part 1

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

Consider an arbitrary solution that achieves $\mathcal{S}'$.

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

Averaged throughput:

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

$$\overline{\rho}^{(k)}(j) = \sum_{u=1}^{p} \overline{\rho}_u^{(k)}(j).$$

Proof (1/3)

Part 1

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

Consider an arbitrary solution that achieves $\mathcal{S}'$.

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

Averaged throughput:

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

$$\overline{\rho}^{(k)}(j) = \sum_{u=1}^{p} \overline{\rho}_u^{(k)}(j).$$

Proof (1/3)

Part 1

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

Consider an arbitrary solution that achieves $\mathcal{S}'$.

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

Averaged throughput:

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

$$\overline{\rho}^{(k)}(j) = \sum_{u=1}^{p} \overline{\rho}_u^{(k)}(j).$$

Proof (1/3)

Part 1

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

$\{\overline{\rho}_u^{(k)}(j), \overline{\rho}^{(k)}(j)\}$ are a valid solution of the linear program:

Proof (1/3)

Part 1

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

The first equation is satisfied:

$$\sum_{\substack{[t^{(j)},\ t^{(j+1)}] \\ t^{(j)} \geq r^{(k)} \\ t^{(j+1)} \leq d^{(k)}}} \overline{\rho}^{(k)}(j) \times (t^{(j+1)} - t^{(j)}) =$$

$$\sum_{\substack{[t^{(j)},\ t^{(j+1)}] \\ t^{(j)} \geq r^{(k)} \\ t^{(j+1)} \leq d^{(k)}}} \sum_{u=1}^{p} \overline{\rho}_u^{(k)}(j) \times (t^{(j+1)} - t^{(j)})$$

Proof (1/3)

Part 1

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

The first equation is satisfied:

$$\sum_{\substack{[t^{(j)},\ t^{(j+1)}] \\ t^{(j)} \geq r^{(k)} \\ t^{(j+1)} \leq d^{(k)}}} \overline{\rho}^{(k)}(j) \times (t^{(j+1)} - t^{(j)}) = \sum_{\substack{[t^{(j)},\ t^{(j+1)}] \\ t^{(j)} \geq r^{(k)} \\ t^{(j+1)} \leq d^{(k)}}} \mathrm{nb}(j,k,u)$$

Proof (1/3)

Part 1

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

The first equation is satisfied:

$$\sum_{\substack{[t^{(j)},\ t^{(j+1)}] \\ t^{(j)} \geq r^{(k)} \\ t^{(j+1)} \leq d^{(k)}}} \overline{\rho}^{(k)}(j) \times (t^{(j+1)} - t^{(j)}) =$$

$$\Pi^{(k)}$$

Proof (1/3)

Part 1

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

The third equation is satisfied:

$$\sum_{k=1}^{n} \overline{\rho}_u^{(k)}(j) \frac{w^{(k)}}{s_u^{(k)}} = \sum_{k=1}^{n} \frac{\mathrm{nb}(j,k,u)}{t^{(j+1)} - t^{(j)}} \cdot \frac{w^{(k)}}{s_u^{(k)}}$$

But we have

$$\sum_{k=1}^{n} \mathrm{nb}(j,k,u) \frac{w^{(k)}}{s_u^{(k)}} \leq t^{(j+1)} - t^{(j)}$$

Proof (1/3)

Part 1

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

The third equation is satisfied:

$$\sum_{k=1}^{n} \overline{\rho}_u^{(k)}(j) \frac{w^{(k)}}{s_u^{(k)}} = \sum_{k=1}^{n} \frac{\text{nb}(j,k,u)}{t^{(j+1)} - t^{(j)}} \cdot \frac{w^{(k)}}{s_u^{(k)}}$$

But we have

$$\sum_{k=1}^{n} \text{nb}(j,k,u) \frac{w^{(k)}}{s_u^{(k)}} \leq t^{(j+1)} - t^{(j)}$$

Proof (1/3)

Part 1

A given max-stretch $\mathcal{S}'$ 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!

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).

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).

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).

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).

Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

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

Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

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

Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

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

Proof (3/3)

Part 3

The binary search needs polynomial number of iterations.

New linear program:

$$\left\{\begin{array}{l}
\textsc{Minimize } \mathcal{S}' \\
\textsc{subject to} \\
\mathcal{S}^1 \leq \mathcal{S}' \leq \mathcal{S}^2 \\
\forall\, 1 \leq k \leq n, \quad \displaystyle\sum_{\substack{[t^{(j)}(\mathcal{S}'),\ t^{(j+1)}(\mathcal{S}')] \\ t^{(j)}(\mathcal{S}') \,\geq\, r^{(k)} \\ t^{(j+1)}(\mathcal{S}') \,\leq\, d^{(k)}(\mathcal{S}')}} \rho^{(k)}(j) \times (t^{(j+1)}(\mathcal{S}') - t^{(j)}(\mathcal{S}')) = \Pi^{(k)} \\
\ldots
\end{array}\right. \quad (4)$$

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 $\mathcal{S}' \in [\mathcal{S}^1, \mathcal{S}^2]$ is achievable.

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

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 $\mathcal{S}' \in [\mathcal{S}^1, \mathcal{S}^2]$ 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

On-line

Off-line algorithm at each release dates:

- For each application $\mathcal{A}_k$, 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 $\mathcal{A}_k$, 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 $\mathcal{A}_k$, 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 $\mathcal{A}_k$, 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

Extension

Multi-level trees

- Resource constraints unchanged
- conservation law stating that for each application $\mathcal{A}_k$ for each internal node

One-port model : previous constraint:

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

Mixed-implementation of the two previous models

Return messages : for each application $\mathcal{A}_k$

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

Extension

Multi-level trees

- Resource constraints unchanged
- conservation law stating that for each application $\mathcal{A}_k$ for each internal node

One-port model : previous constraint:

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

Mixed-implementation of the two previous models

Return messages : for each application $\mathcal{A}_k$

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

Extension

Multi-level trees

- Resource constraints unchanged
- conservation law stating that for each application $\mathcal{A}_k$ for each internal node

One-port model : previous constraint:

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

Mixed-implementation of the two previous models

Return messages : for each application $\mathcal{A}_k$

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

Extension

Multi-level trees

- Resource constraints unchanged
- conservation law stating that for each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$

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

Extension

Multi-level trees

- Resource constraints unchanged
- conservation law stating that for each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$

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

Extension

Multi-level trees

- Resource constraints unchanged
- conservation law stating that for each application $\mathcal{A}_k$ 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 $\mathcal{A}_k$

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

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)

Results

Results

Eh wait!

You don't have any result yet !!

Outline

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