Computing the throughput of replicated workflows on heterogeneous platforms

Matthieu Gallet,

joint work with Anne Benoit, Bruno Gaujal and Yves Robert

CNRS
INRIA GRAAL project-team
Laboratoire de l’Informatique du Parallélisme
École Normale Supérieure de Lyon, France

ALEAE, August 25, 2009
Introduction

Our problem

- A fully connected, heterogeneous platform
- Many instances of the same linear workflow made of $n$ stages
- Some stages may be replicated on several processors (two successive data sets are processed on distinct processors)
Introduction

- Periodic schedules
- The mapping of each stage is given
- Processors allocated to a same stage are served in a Round-Robin fashion
- A processor is devoted to a single stage
- **How to determine the throughput?**
- Throughput: average number of processed instances per time unit
- Period: inverse of the throughput
  - Easily determined without replication
  - Harder problem when stages are replicated
  - Use of Timed Petri Nets models to solve it
Introduction

- Periodic schedules
- The mapping of each stage is given
- Processors allocated to a same stage are served in a Round-Robin fashion
- A processor is devoted to a single stage
- **How to determine the throughput?**
- Throughput: average number of processed instances per time unit
- Period: inverse of the throughput
  - Easily determined without replication
  - Harder problem when stages are replicated
  - Use of Timed Petri Nets models to solve it
Introduction

- Periodic schedules
- The mapping of each stage is given
- Processors allocated to a same stage are served in a Round-Robin fashion
- A processor is devoted to a single stage
- How to determine the throughput?
- Throughput: average number of processed instances per time unit
- Period: inverse of the throughput
  Easily determined without replication
  Harder problem when stages are replicated
  Use of Timed Petri Nets models to solve it
Introduction

- Periodic schedules
- The mapping of each stage is given
- Processors allocated to a same stage are served in a Round-Robin fashion
- A processor is devoted to a single stage

**How to determine the throughput?**

- Throughput: average number of processed instances per time unit
- Period: inverse of the throughput
  - Easily determined without replication
  - Harder problem when stages are replicated
  - Use of Timed Petri Nets models to solve it
Outline

Introduction

Framework

Timed Event Graphs models

Computing mapping throughputs

Conclusion
Outline

Introduction

Framework

Timed Event Graphs models

Computing mapping throughputs

Conclusion
Notations

- Exemple of workflow:

```
S_0 \rightarrow F_0 \rightarrow S_1 \rightarrow F_1 \rightarrow S_2 \rightarrow F_2 \rightarrow S_3
```

- Exemple of platform:

```
P_0 (\Pi_0 \text{ FLOPS}) \rightarrow b_{0,1} \rightarrow P_1 (\Pi_1 \text{ FLOPS}) \rightarrow b_{1,2} \rightarrow P_2 (\Pi_2 \text{ FLOPS}) \rightarrow b_{2,3} \rightarrow P_3 (\Pi_3 \text{ FLOPS})
```

- Two communication models:
  1. **Overlap One-Port**: data set $i+1$ is received during transmission of data set $i-1$ and computation of data set $i$
  2. **Strict One-Port**: receptions, computations and transmissions of the results are serialized
Replication model

- No redundant computations
- If $S_k$ is replicated onto $P_1$, $P_2$ and $P_3$:
  
  \[
  \begin{align*}
  \text{\textbackslash} & S_k \text{ on } P_1: \text{ data sets } 1, 4, 7, \ldots \\
  \ldots & S_{k-1} \quad \text{\textbackslash} & S_k \text{ on } P_2: \text{ data sets } 2, 5, 8, \ldots \\
  \text{\textbackslash} & S_k \text{ on } P_3: \text{ data sets } 3, 5, 9, \ldots \\
  \end{align*}
  \]

- Processors are served in a Round-Robin fashion, even if they have different speeds
- $C_{\text{exec}}(k)$ cycle-time of processor $P_k$
- **Overlap One-Port** model:
  
  \[
  C_{\text{exec}}(k) = \max \{ C_{\text{in}}(k), C_{\text{comp}}(k), C_{\text{out}}(k) \}
  \]

- **Strict One-Port** model:
  
  \[
  C_{\text{exec}}(k) = C_{\text{in}}(k) + C_{\text{comp}}(k) + C_{\text{out}}(k)
  \]

- the maximum cycle-time $M_{\text{ct}} = \max_{1 \leq k \leq p} C_{\text{exec}}(k)$ is a lower bound for the period
Outline

Introduction

Framework

Timed Event Graphs models

Computing mapping throughputs

Conclusion
Key ideas

- Event Graph: Petri Net, such that each place is linked to a single outgoing transition and a single incoming transition.

- Timed Petri Net:
  - The production of tokens takes some time during a firing.
  - A transition cannot be fired again before the end of its current firing.

- Communications and computations are modeled by transitions.

- Dependences are modeled by places between transitions.

- Each path followed by the input data must be fully developed in the TPN.
**Number of paths in the system**

<table>
<thead>
<tr>
<th>Input data</th>
<th>Path in the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$P_0 \rightarrow P_1 \rightarrow P_3 \rightarrow P_6$</td>
</tr>
<tr>
<td>1</td>
<td>$P_0 \rightarrow P_2 \rightarrow P_4 \rightarrow P_6$</td>
</tr>
<tr>
<td>2</td>
<td>$P_0 \rightarrow P_1 \rightarrow P_5 \rightarrow P_6$</td>
</tr>
<tr>
<td>3</td>
<td>$P_0 \rightarrow P_2 \rightarrow P_3 \rightarrow P_6$</td>
</tr>
<tr>
<td>4</td>
<td>$P_0 \rightarrow P_1 \rightarrow P_4 \rightarrow P_6$</td>
</tr>
<tr>
<td>5</td>
<td>$P_0 \rightarrow P_2 \rightarrow P_5 \rightarrow P_6$</td>
</tr>
<tr>
<td>6</td>
<td>$P_0 \rightarrow P_1 \rightarrow P_3 \rightarrow P_6$</td>
</tr>
<tr>
<td>7</td>
<td>$P_0 \rightarrow P_2 \rightarrow P_4 \rightarrow P_6$</td>
</tr>
</tbody>
</table>

**Proposition**

Assume that stage $S_i$ is mapped onto $m_i$ distinct processors. Then the number of paths is equal to $m = \text{lcm}(m_0, \ldots, m_{n-1})$. 
Overlap One-Port model

Dependences between communications and computations

\[ S_0 \quad P_0 \quad F_0 \quad P_0 \rightarrow P_1 \quad S_1 \quad P_1 \quad F_1 \quad P_1 \rightarrow P_3 \quad S_2 \quad P_3 \quad F_2 \quad P_3 \rightarrow P_6 \quad S_3 \quad P_6 \]

\[ P_0 \quad P_0 \rightarrow P_2 \quad P_2 \quad P_2 \rightarrow P_4 \quad P_4 \quad P_4 \rightarrow P_6 \quad P_6 \]

\[ P_0 \quad P_0 \rightarrow P_1 \quad P_1 \quad P_1 \rightarrow P_5 \quad P_5 \quad P_5 \rightarrow P_6 \quad P_6 \]

\[ P_0 \quad P_0 \rightarrow P_2 \quad P_2 \quad P_2 \rightarrow P_3 \quad P_3 \quad P_3 \rightarrow P_6 \quad P_6 \]

\[ P_0 \quad P_0 \rightarrow P_1 \quad P_1 \quad P_1 \rightarrow P_4 \quad P_4 \quad P_4 \rightarrow P_6 \quad P_6 \]

\[ P_0 \quad P_0 \rightarrow P_2 \quad P_2 \quad P_2 \rightarrow P_5 \quad P_5 \quad P_5 \rightarrow P_6 \quad P_6 \]
Overlap One-Port model

Dependences due to the round-robin distribution of computations
Overlap One-Port model

Dependences due to the round-robin distribution of outgoing communications
Overlap One-Port model

Dependences due to the round-robin distribution of incoming communications
Overlap One-Port model

All dependences!
Strict One-Port model

Dependences due to the strict one-port model
Strict One-Port model

All dependences!
Outline

Introduction

Framework

Timed Event Graphs models

Computing mapping throughputs

Conclusion
Critical cycles and throughputs

- $\mathcal{C}$ is a cycle of the TPN
- $\mathcal{L}(\mathcal{C})$ is its length (number of transitions)
- $t(\mathcal{C})$ is the total number of tokens in places traversed by $\mathcal{C}$
- A critical cycle achieves the largest ratio $\max_{\mathcal{C}_{\text{cycle}}} \frac{\mathcal{L}(\mathcal{C})}{t(\mathcal{C})}$
- This ratio gives the period $\mathcal{P}$ of the system
- Can be computed in time $O(n^3m^3)$
  \[ m = \text{lcm}(m_0, \ldots, m_{n-1}) \]
The TPN has an exponential size!

However:

**Theorem.**

Consider a pipeline of \( n \) stages \( S_0, \ldots, S_{n-1} \), such that stage \( S_i \) is mapped onto \( m_i \) distinct processors. Then the average throughput of this system can be computed in time \( \mathcal{O} \left( \sum_{i=0}^{n-2} \left( (m_im_{i+1})^3 \right) \right) \).
Key ideas of the proof

- Split the TPN into $2n - 1$ columns
- Computation columns: simple problem
- Communication columns: reduction to smaller TPNs with critical cycles of same weight
The case of a communication column

- Several connected components.
- Example of connected component:
Outline

Introduction

Framework

Timed Event Graphs models

Computing mapping throughputs

Conclusion
Conclusion

▶ Even if the mapping is given, the throughput is hard to determine
▶ Examples without critical resource with both communication models
▶ Such examples remain seldom
▶ Future work: use dynamic platforms instead of static ones, and find good schedules on these platforms