Complexity results for throughput and latency optimization of replicated and data-parallel workflows

Anne Benoit and Yves Robert

GRAAL team, LIP
École Normale Supérieure de Lyon

June 2007

Introduction and motivation

- Mapping workflow applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline and fork workflows

Introduction and motivation

- Mapping workflow applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline and fork workflows

Introduction and motivation

- Mapping workflow applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline and fork workflows

Introduction and motivation

- Mapping workflow applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline and fork workflows

Introduction and motivation

- Mapping workflow applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline and fork workflows

Introduction and motivation

- Mapping workflow applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline and fork workflows

Rule of the game

- Consecutive data-sets fed into the workflow
- Period T_{period} = time interval between beginning of execution of two consecutive data sets (throughput=$1/T_{\text{period}}$)
- Latency $T_{\text{latency}}(x)$ = time elapsed between beginning and end of execution for a given data set x, and $T_{\text{latency}} = \max_x T_{\text{latency}}(x)$
- Map each pipeline/fork stage on one or several processors
- Goal: minimize T_{period} or T_{latency} or bi-criteria minimization

Rule of the game

- Consecutive data-sets fed into the workflow
- Period T_{period} = time interval between beginning of execution of two consecutive data sets (throughput=$1/T_{\text{period}}$)
- Latency $T_{\text{latency}}(x)$ = time elapsed between beginning and end of execution for a given data set x, and $T_{\text{latency}} = \max_x T_{\text{latency}}(x)$
- Map each pipeline/fork stage on one or several processors
- Goal: minimize T_{period} or T_{latency} or bi-criteria minimization

Replication and data-parallelism

Replicate stage $\mathcal{S}_k$ on $P_1, \ldots, P_q$

$$\ldots \mathcal{S}_{k-1} \begin{cases} \swarrow \\ -- \\ \searrow \end{cases} \begin{array}{l} \mathcal{S}_k \text{ on } P_1\text{: data sets } \mathbf{1, 4, 7, \ldots} \\ \mathcal{S}_k \text{ on } P_2\text{: data sets } \mathbf{2, 5, 8, \ldots} \\ \mathcal{S}_k \text{ on } P_3\text{: data sets } \mathbf{3, 5, 9, \ldots} \end{array} \begin{cases} \searrow \\ -- \\ \swarrow \end{cases} \mathcal{S}_{k+1} \ldots$$

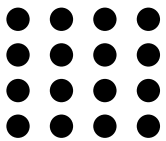
Data-parallelize stage $\mathcal{S}_k$ on $P_1, \ldots, P_q$

Replication and data-parallelism

Replicate stage $\mathcal{S}_k$ on $P_1, \ldots, P_q$

$$\ldots \mathcal{S}_{k-1} \begin{cases} \swarrow \\ -- \\ \searrow \end{cases} \begin{array}{l} \mathcal{S}_k \text{ on } P_1\text{: data sets } \mathbf{1, 4, 7, \ldots} \\ \mathcal{S}_k \text{ on } P_2\text{: data sets } \mathbf{2, 5, 8, \ldots} \\ \mathcal{S}_k \text{ on } P_3\text{: data sets } \mathbf{3, 5, 9, \ldots} \end{array} \begin{cases} \searrow \\ -- \\ \swarrow \end{cases} \mathcal{S}_{k+1} \ldots$$

Data-parallelize stage $\mathcal{S}_k$ on $P_1, \ldots, P_q$

$\mathcal{S}_k$ $(w = 16)$  $\Rightarrow$ P_1 $(s_1 = 2)$: ●●●●●●●●

P_2 $(s_2 = 1)$: ●●●●

P_3 $(s_3 = 1)$: ●●●●

Major contributions

- Complexity results for throughput and latency optimization of replicated and data-parallel workflows
- Theoretical approach to the problem
  - definition of replication and data-parallelism
  - formal definition of T_{period} and T_{latency} in each case
- Problem complexity: focus on pipeline and fork workflows

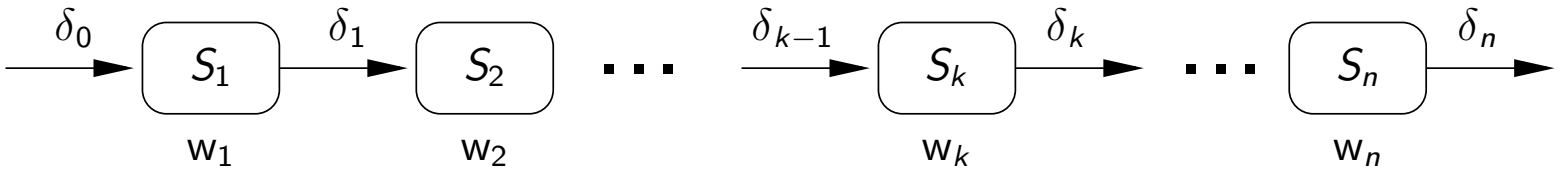
Major contributions

- Complexity results for throughput and latency optimization of replicated and data-parallel workflows
- Theoretical approach to the problem
  - definition of replication and data-parallelism
  - formal definition of T_{period} and T_{latency} in each case
- Problem complexity: focus on pipeline and fork workflows

Outline

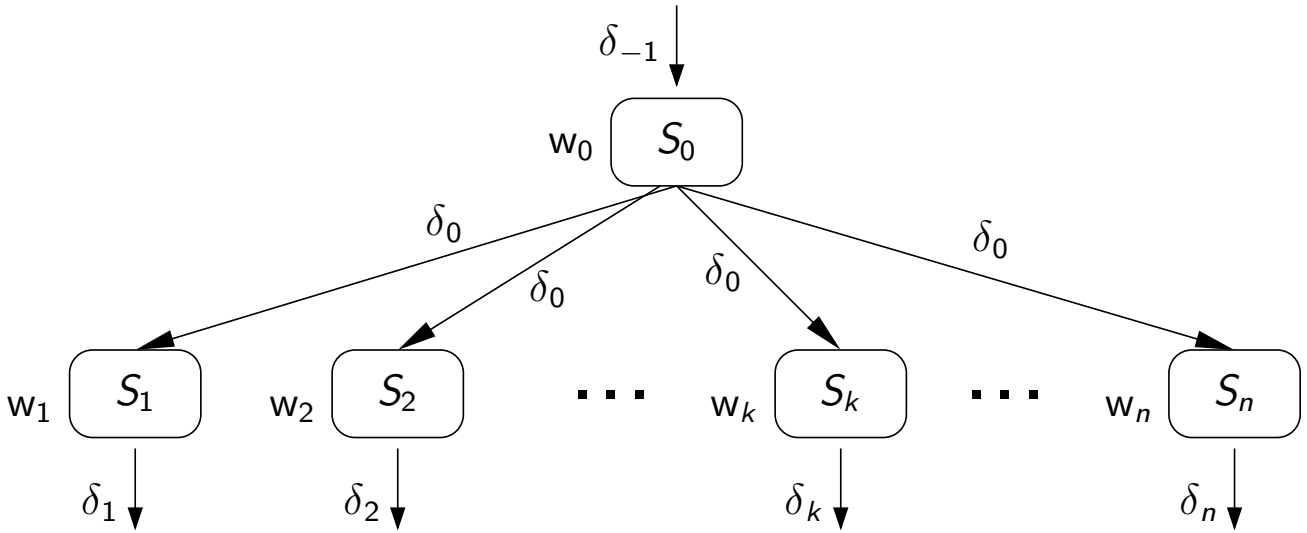
Outline

Pipeline graphs



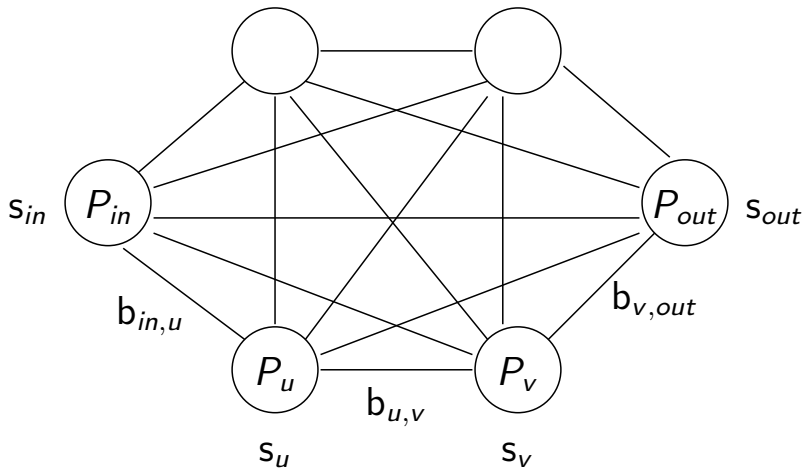
- n stages $\mathcal{S}_k$, $1 \leq k \leq n$
- $\mathcal{S}_k$:
  - receives input of size δ_{k-1} from $\mathcal{S}_{k-1}$
  - performs w_k computations
  - outputs data of size δ_k to $\mathcal{S}_{k+1}$

Fork graphs



- $n + 1$ stages $\mathcal{S}_k$, $0 \leq k \leq n$
  - $\mathcal{S}_0$: root stage
  - $\mathcal{S}_1$ to $\mathcal{S}_n$: independent stages
- A data set goes through stage $\mathcal{S}_0$, then it can be executed simultaneously for all other stages

The platform



- p processors P_u, $1 \leq u \leq p$, fully interconnected
- s_u: speed of processor P_u
- bidirectional link $link_{u,v} : P_u \longrightarrow P_v$, bandwidth $b_{u,v}$
- one-port model: each processor can either send, receive or compute at any time-step

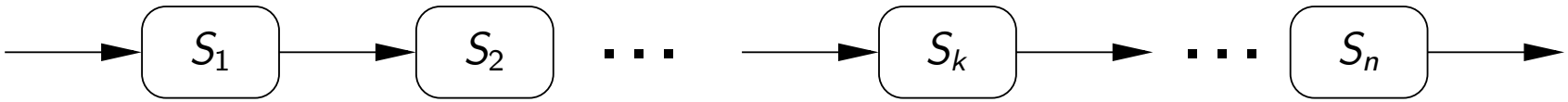
Different platforms

Fully Homogeneous – Identical processors ($s_u = s$) and links ($b_{u,v} = b$): typical parallel machines

Communication Homogeneous – Different-speed processors ($s_u \neq s_v$), identical links ($b_{u,v} = b$): networks of workstations, clusters

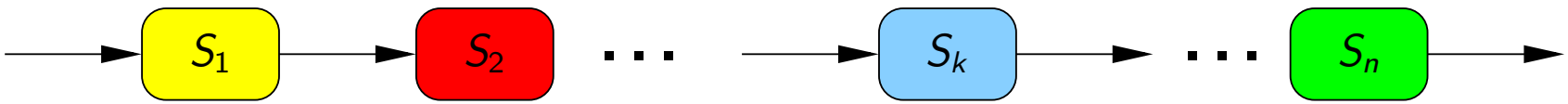
Fully Heterogeneous – Fully heterogeneous architectures, $s_u \neq s_v$ and $b_{u,v} \neq b_{u',v'}$: hierarchical platforms, grids

Back to pipeline: mapping strategies



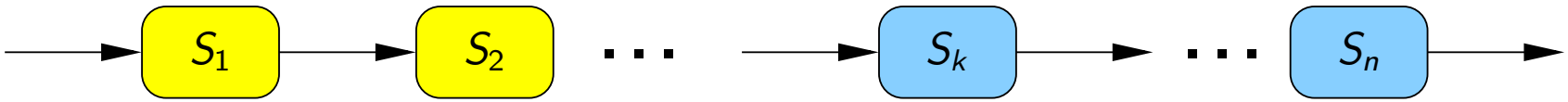
The pipeline application

Back to pipeline: mapping strategies



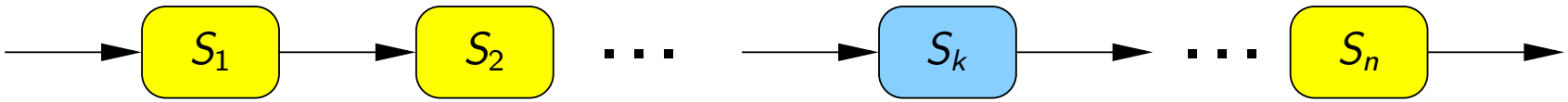
One-to-one Mapping

Back to pipeline: mapping strategies



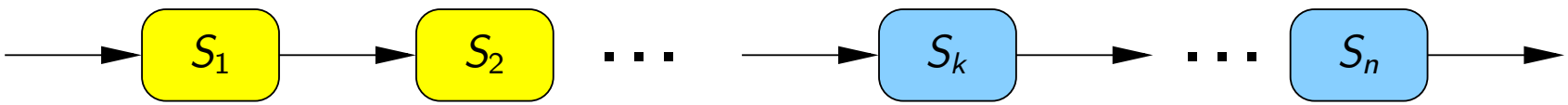
Interval Mapping

Back to pipeline: mapping strategies



General Mapping

Back to pipeline: mapping strategies



Interval Mapping

In this work, Interval Mapping

Chains-on-chains

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

Chains-on-chains

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

Chains-on-chains

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

5 7 3 4 | 8 1 3 8 | 2 9 7 | 3 5 2 3 6

$T_{\text{period}} = 20$

Chains-on-chains

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

5 7 3 4 | 8 1 3 8 | 2 9 7 | 3 5 2 3 6

$T_{\text{period}} = 20$

NP-hard for different-speed processors, even without communications

Outline

Working out an example

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$T_{\text{period}} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($T_{\text{latency}} = 17$)

Optimal latency?

$T_{\text{latency}} = 12$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$ ($T_{\text{period}} = 12$)

Min. latency if $T_{\text{period}} \leq 10$?

$T_{\text{latency}} = 14$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

Working out an example

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$T_{\text{period}} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($T_{\text{latency}} = 17$)

Optimal latency?

$T_{\text{latency}} = 12$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$ ($T_{\text{period}} = 12$)

Min. latency if $T_{\text{period}} \leq 10$?

$T_{\text{latency}} = 14$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

Working out an example

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$T_{\text{period}} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($T_{\text{latency}} = 17$)

Optimal latency?

$T_{\text{latency}} = 12$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$ ($T_{\text{period}} = 12$)

Min. latency if $T_{\text{period}} \leq 10$?

$T_{\text{latency}} = 14$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

Working out an example

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$T_{\text{period}} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($T_{\text{latency}} = 17$)

Optimal latency?

$T_{\text{latency}} = 12$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$ ($T_{\text{period}} = 12$)

Min. latency if $T_{\text{period}} \leq 10$?

$T_{\text{latency}} = 14$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

Example with replication and data-parallelism

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Replicate interval $[S_u..\mathcal{S}_v]$ on $P_1, \ldots, P_q$

$$\ldots \mathcal{S} \begin{array}{ll} \diagup & \\ - & \mathcal{S}_u \ldots \mathcal{S}_v \text{ on } P_1\text{: data sets } \mathbf{1, 4, 7, \ldots} \\ - & \mathcal{S}_u \ldots \mathcal{S}_v \text{ on } P_2\text{: data sets } \mathbf{2, 5, 8, \ldots} \\ \diagdown & \mathcal{S}_u \ldots \mathcal{S}_v \text{ on } P_3\text{: data sets } \mathbf{3, 5, 9, \ldots} \end{array} \begin{array}{l} \diagdown \\ - \\ - \\ \diagup \end{array} \mathcal{S} \ldots$$

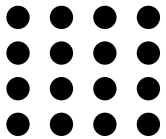
$$T_{\text{period}} = \frac{\sum_{k=u}^{v} w_k}{q \times \min_i(s_i)} \text{ and } T_{\text{latency}} = q \times T_{\text{period}}$$

Example with replication and data-parallelism

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Data Parallelize single stage $\mathcal{S}_k$ on $P_1, \ldots, P_q$

$\mathcal{S}$ $(w = 16)$ 

$\Rightarrow$ P_1 $(s_1 = 2)$:

P_2 $(s_2 = 1)$:

P_3 $(s_3 = 1)$:

$$T_{\text{period}} = \frac{w_k}{\sum_{i=1}^{q} s_i} \text{ and } T_{\text{latency}} = T_{\text{period}}$$

Example with replication and data-parallelism

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

Example with replication and data-parallelism

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$\mathcal{S}_1 \overset{\text{DP}}{\rightarrow} P_1 P_2$, $\mathcal{S}_2 \mathcal{S}_3 \mathcal{S}_4 \overset{\text{REP}}{\rightarrow} P_3 P_4$

$T_{\text{period}} = \max(\frac{14}{2+1}, \frac{4+2+4}{2 \times 1}) = 5$, $T_{\text{latency}} = 14.67$

Example with replication and data-parallelism

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$\mathcal{S}_1 \overset{\text{DP}}{\rightarrow} P_1P_2$, $\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \overset{\text{REP}}{\rightarrow} P_3P_4$

$T_{\text{period}} = \max(\frac{14}{2+1}, \frac{4+2+4}{2\times 1}) = 5$, $T_{\text{latency}} = 14.67$

$\mathcal{S}_1 \overset{\text{DP}}{\rightarrow} P_2P_3P_4$, $\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$

$T_{\text{period}} = \max(\frac{14}{1+1+1}, \frac{4+2+4}{2}) = 5$, $T_{\text{latency}} = 9.67$ (optimal)

Outline

Interval Mapping for pipeline graphs

- Several consecutive stages onto the same processor
- Increase computational load, reduce communications
- Partition of [1..n] into m intervals $I_j = [d_j, e_j]$ (with $d_j \leq e_j$ for $1 \leq j \leq m$, $d_1 = 1$, $d_{j+1} = e_j + 1$ for $1 \leq j \leq m-1$ and $e_m = n$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

$$T_{\text{period}} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

$$T_{\text{latency}} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

Interval Mapping for pipeline graphs

- Several consecutive stages onto the same processor
- Increase computational load, reduce communications
- Partition of [1..n] into m intervals $I_j = [d_j, e_j]$ (with $d_j \leq e_j$ for $1 \leq j \leq m$, $d_1 = 1$, $d_{j+1} = e_j + 1$ for $1 \leq j \leq m-1$ and $e_m = n$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

$$T_{\text{period}} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

$$T_{\text{latency}} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

Interval Mapping for pipeline graphs

- Several consecutive stages onto the same processor
- Increase computational load, reduce communications
- Partition of [1..n] into m intervals $I_j = [d_j, e_j]$ (with $d_j \leq e_j$ for $1 \leq j \leq m$, $d_1 = 1$, $d_{j+1} = e_j + 1$ for $1 \leq j \leq m-1$ and $e_m = n$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

$$T_{\text{period}} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

$$T_{\text{latency}} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

Interval Mapping for pipeline graphs

- Several consecutive stages onto the same processor
- Increase computational load, reduce communications
- Partition of [1..n] into m intervals $I_j = [d_j, e_j]$ (with $d_j \leq e_j$ for $1 \leq j \leq m$, $d_1 = 1$, $d_{j+1} = e_j + 1$ for $1 \leq j \leq m-1$ and $e_m = n$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

$$T_{\text{period}} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

$$T_{\text{latency}} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{\text{alloc}(j-1),\text{alloc}(j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b_{\text{alloc}(j),\text{alloc}(j+1)}} \right\}$$

Fork graphs

- map any partition of the graph onto the processors
- *q intervals*, $q \leq$ p
- first interval: $\mathcal{S}_0$ and possibly $\mathcal{S}_1$ to $\mathcal{S}_k$
- next intervals of independent stages

Fork graphs

- map any partition of the graph onto the processors
- q *intervals*, $q \leq$ p
- first interval: $\mathcal{S}_0$ and possibly $\mathcal{S}_1$ to $\mathcal{S}_k$
- next intervals of independent stages
- $T_{\text{period}} = ?$

Fork graphs

- map any partition of the graph onto the processors
- *q intervals*, $q \leq$ p
- first interval: $\mathcal{S}_0$ and possibly $\mathcal{S}_1$ to $\mathcal{S}_k$
- next intervals of independent stages

- $T_{\text{period}} = ?$
- depends on the com model: is it possible to start com as soon as $\mathcal{S}_0$ is done? Which order for com?

-

Fork graphs

- map any partition of the graph onto the processors
- *q intervals*, $q \leq$ p
- first interval: $\mathcal{S}_0$ and possibly $\mathcal{S}_1$ to $\mathcal{S}_k$
- next intervals of independent stages
- Informally: $T_{\text{period}} =$ max time needed by processor to receive data, compute, output result
-
-

Fork graphs

- map any partition of the graph onto the processors
- *q intervals*, $q \leq$ p
- first interval: $\mathcal{S}_0$ and possibly $\mathcal{S}_1$ to $\mathcal{S}_k$
- next intervals of independent stages
- Informally: $T_{\text{period}} =$ max time needed by processor to receive data, compute, output result
- $T_{\text{latency}} =$ time elapsed between data set input to $\mathcal{S}_0$ until last computation for this data set is completed
-

Fork graphs

- map any partition of the graph onto the processors
- *q intervals*, $q \leq$ p
- first interval: $\mathcal{S}_0$ and possibly $\mathcal{S}_1$ to $\mathcal{S}_k$
- next intervals of independent stages
- Informally: $T_{\text{period}} =$ max time needed by processor to receive data, compute, output result
- $T_{\text{latency}} =$ time elapsed between data set input to $\mathcal{S}_0$ until last computation for this data set is completed
- Simpler model for formal analysis

Back to a simpler problem

- No communication costs nor overheads
-
-
-
-

Back to a simpler problem

- No communication costs nor overheads
- Cost to execute $\mathcal{S}_i$ on P_u alone:

$$\frac{w_i}{s_u}$$

Back to a simpler problem

- No communication costs nor overheads
- Cost to execute $\mathcal{S}_i$ on P_u alone: $\frac{w_i}{s_u}$
- Cost to data-parallelize $[\mathcal{S}_i, \mathcal{S}_j]$ ($i = j$ for pipeline; $0 < i \leq j$ or $i = j = 0$ for fork) on k processors $P_{q_1}, \ldots, P_{q_k}$:

$$\frac{\sum_{\ell=i}^{j} w_\ell}{\sum_{u=1}^{k} s_{q_u}}.$$

  Cost = T_{period} of assigned processors
  Cost = delay to traverse the interval

-
-

Back to a simpler problem

- No communication costs nor overheads
- Cost to execute $\mathcal{S}_i$ on P_u alone: $\frac{w_i}{s_u}$
- Cost to data-parallelize
- Cost to replicate $[\mathcal{S}_i, \mathcal{S}_j]$ on k processors $P_{q_1}, \ldots, P_{q_k}$:

$$\frac{\sum_{\ell=i}^{j} w_\ell}{k \times \min_{1 \leq u \leq k} s_{q_u}}.$$

Cost $= T_{\text{period}}$ of assigned processors
Delay to traverse the interval = time needed by slowest processor:

$$t_{\max} = \frac{\sum_{\ell=i}^{j} w_\ell}{\min_{1 \leq u \leq k} s_{q_u}}$$

Back to a simpler problem

- No communication costs nor overheads
- Cost to execute $\mathcal{S}_i$ on P_u alone: $\frac{w_i}{s_u}$
- Cost to data-parallelize
- Cost to replicate
- With these formulas: easy to compute T_{period} for both graphs, and T_{latency} for pipeline graphs

Latency for a fork?

- partition of stages into q sets $\mathcal{I}_r$ $(1 \leq r \leq q \leq p)$
- $\mathcal{S}_0 \in \mathcal{I}_1$, to k processors $P_{q_1}, \ldots, P_{q_k}$
- $t_{\max}(r)$ = delay of r-th set $(1 \leq r \leq q)$, computed as before
- flexible com model: computations of $\mathcal{I}_r$, $r \geq 2$, start as soon as computation of $\mathcal{S}_0$ is completed.
- s_0 = speed at which $\mathcal{S}_0$ is processed:
  - $s_0 = \sum_{u=1}^{k} s_{q_u}$ if $\mathcal{I}_1$ is data-parallelized
  - $s_0 = \min_{1 \leq u \leq k} s_{q_u}$ if $\mathcal{I}_1$ is replicated
- $$T_{\text{latency}} = \max\left(t_{\max}(1), \frac{w_0}{s_0} + \max_{2 \leq r \leq q} t_{\max}(r)\right)$$

Latency for a fork?

- partition of stages into q sets $\mathcal{I}_r$ $(1 \leq r \leq q \leq p)$
- $\mathcal{S}_0 \in \mathcal{I}_1$, to k processors $P_{q_1}, \ldots, P_{q_k}$
- $t_{\max}(r)$ = delay of r-th set $(1 \leq r \leq q)$, computed as before
- flexible com model: computations of $\mathcal{I}_r$, $r \geq 2$, start as soon as computation of $\mathcal{S}_0$ is completed.
- s_0 = speed at which $\mathcal{S}_0$ is processed:
  - $s_0 = \sum_{u=1}^{k} s_{q_u}$ if $\mathcal{I}_1$ is data-parallelized
  - $s_0 = \min_{1 \leq u \leq k} s_{q_u}$ if $\mathcal{I}_1$ is replicated
-

$$T_{\text{latency}} = \max\left(t_{\max}(1), \frac{w_0}{s_0} + \max_{2 \leq r \leq q} t_{\max}(r)\right)$$

Latency for a fork?

- partition of stages into q sets $\mathcal{I}_r$ $(1 \leq r \leq q \leq p)$
- $\mathcal{S}_0 \in \mathcal{I}_1$, to k processors $P_{q_1}, \ldots, P_{q_k}$
- $t_{\max}(r)$ = delay of r-th set $(1 \leq r \leq q)$, computed as before
- flexible com model: computations of $\mathcal{I}_r$, $r \geq 2$, start as soon as computation of $\mathcal{S}_0$ is completed.
- s_0 = speed at which $\mathcal{S}_0$ is processed:
  - $s_0 = \sum_{u=1}^{k} s_{q_u}$ if $\mathcal{I}_1$ is data-parallelized
  - $s_0 = \min_{1 \leq u \leq k} s_{q_u}$ if $\mathcal{I}_1$ is replicated

$$T_{\text{latency}} = \max\left(t_{\max}(1), \frac{w_0}{s_0} + \max_{2 \leq r \leq q} t_{\max}(r)\right)$$

Latency for a fork?

- partition of stages into q sets $\mathcal{I}_r$ $(1 \leq r \leq q \leq \mathrm{p})$
- $\mathcal{S}_0 \in \mathcal{I}_1$, to k processors $P_{q_1}, \ldots, P_{q_k}$
- $t_{\max}(r)$ = delay of r-th set $(1 \leq r \leq q)$, computed as before
- flexible com model: computations of $\mathcal{I}_r$, $r \geq 2$, start as soon as computation of $\mathcal{S}_0$ is completed.
- s_0 = speed at which $\mathcal{S}_0$ is processed:
  - $\mathrm{s}_0 = \sum_{u=1}^{k} \mathrm{s}_{q_u}$ if $\mathcal{I}_1$ is data-parallelized
  - $\mathrm{s}_0 = \min_{1 \leq u \leq k} \mathrm{s}_{q_u}$ if $\mathcal{I}_1$ is replicated
- $$T_{\text{latency}} = \max\left(t_{\max}(1), \frac{\mathrm{w}_0}{\mathrm{s}_0} + \max_{2 \leq r \leq q} t_{\max}(r)\right)$$

Optimization problem

Given

- an application graph (n-stage pipeline or $(n+1)$-stage fork),
- a target platform (*Homogeneous* with p identical processors or *Heterogeneous* with p different-speed processors),
- a mapping strategy with replication, and either with data-parallelization or without,
- an objective (period T_{period} or latency T_{latency}),

determine an interval-based mapping that minimizes the objective

16 optimization problems

Optimization problem

Given

- an application graph (n-stage pipeline or $(n+1)$-stage fork),
- a target platform (*Homogeneous* with p identical processors or *Heterogeneous* with p different-speed processors),
- a mapping strategy with replication, and either with data-parallelization or without,
- an objective (period T_{period} or latency T_{latency}),

determine an interval-based mapping that minimizes the objective

16 optimization problems

Optimization problem

Given

- an application graph (n-stage pipeline or $(n+1)$-stage fork),
- a target platform (*Homogeneous* with p identical processors or *Heterogeneous* with p different-speed processors),
- a mapping strategy with replication, and either with data-parallelization or without,
- an objective (period T_{period} or latency T_{latency}),

determine an interval-based mapping that minimizes the objective

16 optimization problems

Bi-criteria optimization problem

- given threshold period $\mathcal{P}_{\text{threshold}}$, determine mapping whose period does not exceed $\mathcal{P}_{\text{threshold}}$ and that minimizes T_{latency}
- given threshold latency $\mathcal{L}_{\text{threshold}}$, determine mapping whose latency does not exceed $\mathcal{L}_{\text{threshold}}$ and that minimizes T_{period}

Outline

Complexity results

Without data-parallelism, *Homogeneous* platforms

Objective	period	latency	bi-criteria
Hom. pipeline	-		
Het. pipeline	Poly (str)		
Hom. fork	-	Poly (DP)	
Het. fork	Poly (str)	NP-hard	

Complexity results

With data-parallelism, *Homogeneous* platforms

Objective	period	latency	bi-criteria
Hom. pipeline	-		
Het. pipeline	Poly (DP)		
Hom. fork	-	Poly (DP)	
Het. fork	Poly (str)	NP-hard	

Complexity results

Without data-parallelism, *Heterogeneous* platforms

Objective	period	latency	bi-criteria
Hom. pipeline	Poly (*)	-	Poly (*)
Het. pipeline	NP-hard (**)	Poly (str)	NP-hard
Hom. fork	Poly (*)		
Het. fork	NP-hard	-	

Complexity results

With data-parallelism, *Heterogeneous* platforms

Objective	period	latency	bi-criteria
Hom. pipeline	NP-hard		
Het. pipeline	-		
Hom. fork	NP-hard		
Het. fork	-		

Complexity results

Most interesting case:

Without data-parallelism, *Heterogeneous* platforms

Objective	period	latency	bi-criteria
Hom. pipeline	Poly (*)	-	Poly (*)
Het. pipeline	NP-hard (**)	Poly (str)	NP-hard
Hom. fork	Poly (*)		
Het. fork	NP-hard	-	

No data-parallelism, *Heterogeneous* platforms

- For pipeline, minimizing the latency is straightforward: map all stages on fastest proc
- Minimizing the period is NP-hard (involved reduction similar to the heterogeneous chain-to-chain one) for general pipeline
- Homogeneous pipeline: all stages have same workload w: in this case, polynomial complexity.

- Polynomial bi-criteria algorithm for homogeneous pipeline

No data-parallelism, *Heterogeneous* platforms

- For pipeline, minimizing the latency is straightforward: map all stages on fastest proc
- Minimizing the period is NP-hard (involved reduction similar to the heterogeneous chain-to-chain one) for general pipeline
- Homogeneous pipeline: all stages have same workload w: in this case, polynomial complexity.

- Polynomial bi-criteria algorithm for homogeneous pipeline

Lemma: form of the solution

Pipeline, no data-parallelism, *Heterogeneous* platform

Lemma

If an optimal solution which minimizes pipeline period uses q processors, consider q fastest processors $P_1, ..., P_q$, ordered by non-decreasing speeds: $s_1 \leq ... \leq s_q$.
There exists an optimal solution which replicates intervals of stages onto k intervals of processors $I_r = [P_{d_r}, P_{e_r}]$, with $1 \leq r \leq k \leq q$, $d_1 = 1$, $e_k = q$, and $e_r + 1 = d_{r+1}$ for $1 \leq r < k$.

Proof: exchange argument, which does not increase latency

Lemma: form of the solution

Pipeline, no data-parallelism, *Heterogeneous* platform

Lemma

If an optimal solution which minimizes pipeline period uses q processors, consider q fastest processors $P_1, ..., P_q$, ordered by non-decreasing speeds: $s_1 \leq ... \leq s_q$.
There exists an optimal solution which replicates intervals of stages onto k intervals of processors $I_r = [P_{d_r}, P_{e_r}]$, with $1 \leq r \leq k \leq q$, $d_1 = 1$, $e_k = q$, and $e_r + 1 = d_{r+1}$ for $1 \leq r < k$.

Proof: exchange argument, which does not increase latency

Binary-search/Dynamic programming algorithm

- Given latency L, given period K
- Loop on number of processors q
- Dynamic programming algorithm to minimize latency
- Success if L is obtained

- Binary search on L to minimize latency for fixed period
- Binary search on K to minimize period for fixed latency

Binary-search/Dynamic programming algorithm

- Given latency L, given period K
- Loop on number of processors q
- Dynamic programming algorithm to minimize latency
- Success if L is obtained

- Binary search on L to minimize latency for fixed period
- Binary search on K to minimize period for fixed latency

Dynamic programming algorithm

- Compute $L(\mathrm{n}, 1, q)$, where $L(m, i, j)$ = minimum latency to map m pipeline stages on processors P_i to P_j, while fitting in period K.

$$L(m,i,j) = \min_{\substack{1 \leq m' < m \\ i \leq k < j}} \begin{cases} \frac{m.\mathrm{w}}{\mathrm{s}_i} & \text{if } \frac{m.\mathrm{w}}{(j-i).\mathrm{s}_i} \leq K \quad (1) \\ L(m',i,k) + L(m-m',k+1,j) & (2) \end{cases}$$

- Case (1): replicating m stages onto processors $P_i, ..., P_j$
- Case (2): splitting the interval

Dynamic programming algorithm

- Compute $L(\mathrm{n}, 1, q)$, where $L(m, i, j) =$ minimum latency to map m pipeline stages on processors P_i to P_j, while fitting in period K.

$$L(m,i,j) = \min_{\substack{1 \leq m' < m \\ i \leq k < j}} \begin{cases} \frac{m.\mathrm{w}}{\mathrm{s}_i} & \text{if } \frac{m.\mathrm{w}}{(j-i).\mathrm{s}_i} \leq K \quad (1) \\ L(m', i, k) + L(m - m', k+1, j) & (2) \end{cases}$$

Initialization:

$$L(1,i,j) = \begin{cases} \frac{\mathrm{w}}{\mathrm{s}_i} & \text{if } \frac{\mathrm{w}}{(j-i).\mathrm{s}_i} \leq K \\ +\infty & \text{otherwise} \end{cases}$$

$$L(m,i,i) = \begin{cases} \frac{m.\mathrm{w}}{\mathrm{s}_i} & \text{if } \frac{m.\mathrm{w}}{\mathrm{s}_i} \leq K \\ +\infty & \text{otherwise} \end{cases}$$

Dynamic programming algorithm

- Compute $L(\mathrm{n}, 1, q)$, where $L(m, i, j)$ = minimum latency to map m pipeline stages on processors P_i to P_j, while fitting in period K.

$$L(m,i,j) = \min_{\substack{1 \leq m' < m \\ i \leq k < j}} \begin{cases} \frac{m.\mathrm{w}}{\mathrm{s}_i} & \text{if } \frac{m.\mathrm{w}}{(j-i).\mathrm{s}_i} \leq K \quad (1) \\ L(m', i, k) + L(m - m', k+1, j) & (2) \end{cases}$$

- Complexity of the dynamic programming: $O(\mathrm{n}^2.\mathrm{p}^4)$
- Number of iterations of the binary search formally bounded, very small number of iterations in practice.

Outline

Related work

Subhlok and Vondran– Extension of their work (pipeline on hom platforms)

Chains-to-chains– In our work possibility to replicate or data-parallelize

Mapping pipelined computations onto clusters and grids– DAG [Taura et al.], DataCutter [Saltz et al.]

Energy-aware mapping of pipelined computations [Melhem et al.], three-criteria optimization

Mapping pipelined computations onto special-purpose architectures– FPGA arrays [Fabiani et al.]. Fault-tolerance for embedded systems [Zhu et al.]

Mapping skeletons onto clusters and grids– Use of stochastic process algebra [Benoit et al.]

Conclusion

- Mapping structured workflow applications onto computational platforms, with replication and data-parallelism
- Complexity of the most tractable instances → insight of the combinatorial nature of the problem
- Pipeline and fork graphs, extension to fork-join
- *Homogeneous* and *Heterogeneous* platforms with no communications
- Minimizing period or latency, and bi-criteria optimization problems
- Solid theoretical foundation for study of single/bi-criteria mappings, with possibility to replicate and data-parallelize application stages

Conclusion

- Mapping structured workflow applications onto computational platforms, with replication and data-parallelism
- Complexity of the most tractable instances → insight of the combinatorial nature of the problem
- Pipeline and fork graphs, extension to fork-join
- *Homogeneous* and *Heterogeneous* platforms with no communications
- Minimizing period or latency, and bi-criteria optimization problems

- Solid theoretical foundation for study of single/bi-criteria mappings, with possibility to replicate and data-parallelize application stages

Future work

Short term

- Select polynomial instances of the problem and assess complexity when adding communication
- Design heuristics to solve combinatorial instances of the problem

Longer term

- Heuristics based on our polynomial algorithms for general application graphs structured as combinations of pipeline and fork kernels
- Real experiments on heterogeneous clusters
- Comparison of effective performance against theoretical performance