Throughput optimization for micro-factories subject to task and machine failures

Anne Benoit, LIP, ENS Lyon, France
A. Dobrila, J.-M. Nicod and L. Philippe, LIFC, France

April 19, 2010
APDCM 2010, in conjunction with IPDPS 2010, Atlanta, USA
Mapping several tasks onto a set of machines, to optimize the throughput of the application, in the context of a micro-factory

Tasks and machines subject to failures:
- failures attached to tasks (intrinsic difficulty of the task)
- failures attached to tasks and machines (more or less efficient machines for a given task)

Failures impact the throughput: some data sets are lost; mono-criterion problem of throughput maximization
Mapping several tasks onto a set of machines, to optimize the throughput of the application, in the context of a micro-factory

Tasks and machines subject to failures:
- failures attached to tasks (intrinsic difficulty of the task)
- failures attached to tasks and machines (more or less efficient machines for a given task)

Failures impact the throughput: some data sets are lost; mono-criterion problem of throughput maximization
Mapping several tasks onto a set of machines, to optimize the throughput of the application, in the context of a micro-factory

Tasks and machines subject to failures:
- failures attached to tasks (intrinsic difficulty of the task)
- failures attached to tasks and machines (more or less efficient machines for a given task)

Failures impact the throughput: some data sets are lost; mono-criterion problem of throughput maximization
The micro-factory

- Pieces composed of micro-metric elements
- Use of dynamic modules of production
- Still in laboratories (mechanical aspects, elementary actuators as piezo-electric beams ...)
- Nothing on scheduling
- Particular DAG (in-tree)
Outline

1. Framework
   - Application
   - Platform
   - Failure model
   - Optimization problem

2. Complexity results

3. Heuristics

4. Simulations
Application

- Set $\mathcal{N}$ of $n$ tasks: $\mathcal{N} = \{T_1, T_2, \ldots, T_n\}$
- Set $\mathcal{T}$ of $p$ task types with $n \geq p$ and type function $t : [1..n] \rightarrow \mathcal{T}$
- In-tree structure

Figure: Example of application
Platform

- Set $\mathcal{M}$ of $m$ machines: $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$
- Fully connected graph
- Machine $M_u$ can perform task $T_i$ in time $w_{i,u}$
- Two tasks of same type have same execution time on $M_u$:
  $\forall i, i' \in [1, n] \quad t(i) = t(i') \Rightarrow w_{i,u} = w_{i',u}$
Platform

- Set $\mathcal{M}$ of $m$ machines: $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$
- Fully connected graph
- Machine $M_u$ can perform task $T_i$ in time $w_{i,u}$
- Two tasks of same type have same execution time on $M_u$:
  $\forall i, i' \in [1, n] \quad t(i) = t(i') \implies w_{i,u} = w_{i',u}$
Failure model

- Failure attached to tasks and machines
- Task $T_i$ mapped on machine $M_u$: $f_{i,u} = \frac{l_{i,u}}{b_{i,u}}$
- $l_{i,u}$ lost products for $b_{i,u}$ processed products
- Two tasks of same type fail with same rate on $M_u$: $orall i, i' \in [1, n] \quad t(i) = t(i') \Rightarrow f_{i,u} = f_{i',u}$
Failure model

- Failure attached to tasks and machines
- Task $T_i$ mapped on machine $M_u$: $f_{i,u} = \frac{l_{i,u}}{b_{i,u}}$
- $l_{i,u}$ lost products for $b_{i,u}$ processed products
- Two tasks of same type fail with same rate on $M_u$: $\forall i, i' \in [1, n] \; t(i) = t(i') \Rightarrow f_{i,u} = f_{i',u}$
Objective function

- Find an allocation function \( a : [1..n] \rightarrow [1..m] \)
- Goal: maximize the throughput
  - Number of products computed by task \( T_i \) within one period, where \( T_j \) is the unique successor of \( T_i \) (or \( x_j = 1 \)):
    \[
    x_i = \frac{1}{1 - f_{i,a(i)}} \times x_j
    \]
- Period: time between the output of two successful products:
  \[
  \text{period}(M_u) = \sum_{a(i)=u} x_i \times w_{i,u}
  \]
Objective function

- Find an allocation function $a : [1..n] \rightarrow [1..m]$
- Goal: maximize the throughput
- Number of products computed by task $T_i$ within one period, where $T_j$ is the unique successor of $T_i$ (or $x_j = 1$):

$$x_i = \frac{1}{1 - f_{i,a(i)}} \times x_j$$

- Period: time between the output of two successful products:

$$\text{period}(M_u) = \sum_{a(i)=u} x_i \times w_{i,u}$$
Rules of the game

- One-to-one mapping

(a) 

(b) 

(c)
Rules of the game

- One-to-one mapping

- Specialized mapping

\[ t(1) = t(3) = t(5) = 1 \text{ and } t(2) = t(4) = 2 \]
Outline

1. Framework
2. Complexity results
   - One-to-one mappings
   - Specialized mappings
3. Heuristics
4. Simulations
Complexity results of one-to-one mappings

Failures attached only to tasks

\[ f_{i,u} = f_i \rightarrow \text{minimum weight matching in a bipartite graph} \]
Complexity results of one-to-one mappings

- **Failures attached only to tasks**
  \[ f_{i,u} = f_i \rightarrow \text{minimum weight matching in a bipartite graph} \]

- **Homogeneous machines and linear chains**
  \[ w_{i,u} = w, \text{ linear chains} \rightarrow \text{min. weight matching in bipartite graph} \]
Complexity results of one-to-one mappings

Failures attached only to tasks

\[ f_{i,u} = f_i \rightarrow \text{minimum weight matching in a bipartite graph} \]

Homogeneous machines and linear chains

\[ w_{i,u} = w, \text{ linear chains} \rightarrow \text{min. weight matching in bipartite graph} \]

- **Period** = \( x_1 \times w \) since \( w_{i,u} = w \) and \( x_1 = \max_{1 \leq i \leq n} x_i \)

\[ \rightarrow \text{problem of minimization of } x_1 = \prod_{1 \leq i \leq n} \frac{1}{1-f_{i,a(i)}} \]

\[ \rightarrow \text{equivalent to a minimum weight matching in bipartite graphs, which can be found in polynomial time} \]
Complexity results of one-to-one mappings

Failures attached only to tasks
\[ f_{i,u} = f_i \rightarrow \text{minimum weight matching in a bipartite graph} \]

Homogeneous machines and linear chains
\[ w_{i,u} = w, \text{ linear chains} \rightarrow \text{min. weight matching in bipartite graph} \]

Homogeneous machines and general applications
\[ w_{i,u} = w, f_{i,u} = f_u, \text{ in-tree applications} \rightarrow \text{NP-hard} \]
Complexity results of one-to-one mappings

**Failures attached only to tasks**

\[ f_{i,u} = f_i \rightarrow \text{minimum weight matching in a bipartite graph} \]

**Homogeneous machines and linear chains**

\[ w_{i,u} = w, \text{ linear chains} \rightarrow \text{min. weight matching in bipartite graph} \]

**Homogeneous machines and general applications**

\[ w_{i,u} = w, f_{i,u} = f_u, \text{ in-tree applications} \rightarrow \text{NP-hard} \]

- reduction from 3-PARTITION: given a set \( \{z_1, \ldots, z_{3n}\} \) of \( 3n \) integers, and an integer \( Z \) such that \( \sum_{1 < j \leq 3n} z_j = nZ \), does there exist \( n \) independent subsets \( B_1, \ldots, B_n \) of \( \{z_1, \ldots, z_{3n}\} \) such that for all \( 1 \leq i \leq n \), \( \sum_{z_j \in B_i} z_j = Z \)?
Complexity results of one-to-one mappings

Failures attached only to tasks

\[ f_{i,u} = f_i \rightarrow \text{minimum weight matching in a bipartite graph} \]

Homogeneous machines and linear chains

\[ w_{i,u} = w, \text{ linear chains} \rightarrow \text{min. weight matching in bipartite graph} \]

Homogeneous machines and general applications

\[ w_{i,u} = w, f_{i,u} = f_u, \text{ in-tree applications} \rightarrow \text{NP-hard} \]

- Instance with \(3n + 1\) tasks and processors: \(n\) linear chains of 4 tasks sharing the same final task, \(w = 1, f_{3n+1} = 0, f_u = \frac{2^u - 1}{2^u}, K = 2^Z\)
Complexity results of specialized mappings

Homogeneous setting, failures attached only to tasks

- NP-hard, even for linear chain with $f_{i,u} = f_i$ and $w_{i,u} = w$
Complexity results of specialized mappings

Homogeneous setting, failures attached only to tasks

- NP-hard, even for linear chain with $f_{i,u} = f_i$ and $w_{i,u} = w$

- Problem equivalent to 2-PARTITION:
  - Given a set $\{a_1, ..., a_n\}$ of $n$ integers, does it exist a subset $I$ such that $\sum_{i \in I} a_i = \frac{1}{2} \sum_{1 \leq j \leq n} a_j$?
  - Instance with $n$ tasks ordered as a linear chain, two machines, $w = 1$, $f_i = \frac{a_i - a_{i+1}}{a_i}$, $K = \frac{1}{2} \sum_{1 \leq j \leq n} a_j$
  - We prove that $x_i = a_i$, and then one must two-partition tasks between the two machines
Solving the specialized mapping problem

Optimal solution using integer linear programming

Minimize $K$ under the constraints:

\[
\begin{align*}
\forall i & \quad \sum_u a_{i,u} = 1 \\
\forall u & \quad \sum_j t_{u,j} \leq 1 \\
\forall u \forall i & \quad a_{i,u} \leq t_{u,t(i)} \\
\forall u \forall i & \quad x_i \geq \frac{1}{1-f_{i,u}} a_{i,u} \times x_{i+1} \\
\forall u & \quad \sum_i a_{i,u} \times x_i w_{i,u} \leq K
\end{align*}
\]
Solving the specialized mapping problem

Optimal solution using integer linear programming

\[
\begin{align*}
\text{Minimize } K & \text{ under the constraints:} \\
\forall i \sum_u a_{i,u} &= 1 \\
\forall u \sum_j t_{u,j} &\leq 1 \\
\forall u \forall i \ a_{i,u} &\leq t_{u,t(i)} \\
\forall i \forall u \ x_i &\geq \frac{1}{1-f_{i,u}} \ x_{i+1} - (1 - a_{i,u}) \text{MAX}_x_i \\
\forall u \sum_i y_{i,u} w_{i,u} &\leq K \\
\forall i \forall u \ y_{i,u} &\leq a_{i,u} \text{MAX}_x_i \\
\forall i \forall u \ y_{i,u} &\leq x_i \\
\forall i \forall u \ y_{i,u} &\geq x_i - (1 - a_{i,u}) \text{MAX}_x_i
\end{align*}
\]
Outline

1. Framework
2. Complexity results
3. Heuristics
   - H1, H2, H3
   - H4, H4w, H4f
4. Simulations
Heuristics H1, H2 and H3

H1 - Random
- One constraint: respect the specialized mapping

H2 - Potential optimization
- Assign to a machine a set of tasks that it is efficient for
- Make the best use of each machine

H3 - Heterogeneity level
- Sort the machines by their heterogeneity level
- Assign first the most heterogeneous one
H4, H4w and H4f

H4 - performance
- Read the task graph backwards
- Allocate a task to the machine having the best factor: $w_{i,u} \times f_{i,u} \times x_i$

H4w - fastest
- Allocate a task to the fastest machine: $w_{i,u} \times x_i$

H4f - reliability
- Allocate a task to the most reliable machine: $f_{i,u} \times x_i$
Outline

1. Framework
2. Complexity results
3. Heuristics
4. Simulations
   - Configuration
   - Experiments
Configuration

- $m$ machines
- $p$ tasks types
- $n$ tasks
- average value on 30 simulations
- $w_{i,u}$ randomly chosen between 100 and 1000 $ms$,
- $f_i$ and $f_{i,u}$ randomly chosen between 0 and 2 %
Figure: \( m = 50, \ p = 5 \)
H1 and H4f are always less good than the other heuristics, we focus on the four other variants

Figure: $m = 50, \ p = 5$
Figure: $m = 100, \ p = 5$
Heuristics still efficient when the size of application and platform grows

Figure: $m = 100, \ p = 5$
Figure: $f_i$ platform with $m = n = 100$
With $f_{i,u} = f_i$, we can compute the optimal one-to-one solution, H4w is at 1.28 of the optimal
Figure: $m = 5$, $p = 2$
Comparison to the linear program solution for small platforms, H4w is at 1.33 of the optimal
Conclusion and future work

- Throughput maximization for micro-factories subject to task and machine failures
- Exhaustive complexity study: polynomial vs NP-hard instances
- Set of polynomial heuristics for specialized mappings; H4w is the most performing solution (focuses on execution speed and not on failure rate)

Future work:
- Task performed by several machines: possibility to divide the workload of a task and improve the throughput
- Different objective functions: makespan, bi-criteria, ...
Conclusion and future work

- Throughput maximization for micro-factories subject to task and machine failures
- Exhaustive complexity study: polynomial vs NP-hard instances
- Set of polynomial heuristics for specialized mappings; H4w is the most performing solution (focuses on execution speed and not on failure rate)

Future work:
- Task performed by several machines: possibility to divide the workload of a task and improve the throughput
- Different objective functions: makespan, bi-criteria, ...