Batch Scheduling for Identical Multi-Tasks Jobs on Heterogeneous Platforms

Jean-Marc Nicod (Jean-Marc.Nicod@lifc.univ-fcomte.fr)
Sékou Diakité, Laurent Philippe

- 16/05/2008
Laboratoire d’Informatique de l’Université de Franche-Comté - Besançon
2nd “Scheduling in Aussois” workshop
Presentation Outline

Problem definition
Algorithms evaluation
  Presentation
  Experiences
  Synthesis
Steady state for small batches
  Means of action
  Reduce period size
  Experiences
Conclusion and futur works
Outline

Problem definition
Algorithms evaluation
  Presentation
  Experiences
  Synthesis
Steady state for small batches
  Means of action
  Reduce period size
  Experiences
Conclusion and futur works
Execution Platform:
- non oriented graph, $G = (P, L)$:
  - $P$: processors $p_i, i \in [1, m]$
  - $L$: communication links $l_j, j \in [1, c]$

Jobs:
- DAGs without fork (intrees), $J = (T, D)$:
  - $T$: tasks $t_k, k \in [1, n]$
  - $D$: tasks dependencies $d_l, l \in [1, d]$
- $N$ instances of the same job.
Scheduling Problem

Characteristics:
- Each task of a job must be performed by a specific function,
- $F$ is the set of the functions needed to process a job,
- Each execution resource provides a subset of $F$,
- The execution resources are unrelated.

Problem:
- Schedule a batch of $N$ jobs $J$
- Objective function: $C_{\text{max}}$
- $R_m \mid \text{intrees}, \text{batch of identical jobs} \mid C_{\text{max}}$
N instances of the same job:

1 platform for execution:

<table>
<thead>
<tr>
<th>Type</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>∞</td>
<td>∞</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>10</td>
<td>10</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure: Job
Grid:
- Image processing: filters, 2D or 3D reconstructions,
- Servers provides an application set.

Micro-Factories:
- Composed of cells: assembly, treatments, ...
- Less geographical constraints,
- Products are micro-metric → easily buffered
Outline

Problem definition

Algorithms evaluation
  Presentation
  Experiences
  Synthesis

Steady state for small batches
  Means of action
  Reduce period size
  Experiences

Conclusion and future works
Classical (Off-line):
- schedule a DAG on an heterogeneous platform,
- $C_{\text{max}}$ optimization $\rightarrow$ NP-Hard,
- batch of jobs: no use of identical jobs.

Steady state technics (Off-line):
- flow optimization: maximize the throughput
- optimal solution on heterogeneous platforms,
- use the identical job characteristic,
- does not take starting/ending into account.

On-line:
- batch $\rightarrow$ waiting queue,
- schedule ready tasks,
- no use of identical job characteristic,
**On-line** scheduling:
- simple, assign tasks on the fly,
- respect dependencies,
- used as reference.

**Genetic Meta-heuristic GATS[Daoud05]:**
- improves list scheduling,
- good results on DAG schedule,
- needs to be adapted to batches: period,
- performances of a standard heuristic on batches of jobs?

**Steady State[Beaumont04]:**
- optimal for batches of infinite size,
- performances on finite size batches?
Steady State
Overview

- Definition and resolution of a linear program:
  - define the constraints of the problem,
  - flow: solutions are time ratios per resources.

- Compute a cyclic schedule:
  - construct allocations with respect of the ratios,
  - 1 port model: communication intervals.

- Execution:
  - starting,
  - cyclic schedule,
  - ending.
MAXIMIZE \( \rho = \sum_{i=1}^{p} \text{cons}(p_i, t_f) \),

UNDER THE CONSTRAINTS

\[
\begin{cases}
(1) p_i, t_k \in T, \alpha(p_i, t_k) = \text{cons}(p_i, t_k) \times c_{i,k} \\
(2) p_i, t_k \in T, 0 \leq \alpha(p_i, t_k) \leq 1 \\
(3) p_i, \sum_{t_k \in T} \alpha(p_i, t_k) \leq 1
\end{cases}
\]
Steady State Solution

Figure: Job

Table: Cost: \( c \)

<table>
<thead>
<tr>
<th>Type</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>C</td>
<td>( \infty )</td>
<td>10</td>
<td>10</td>
<td>( \infty )</td>
</tr>
<tr>
<td>D</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table: Consumption: \( \text{cons} \), objective \( \rho = 2/25 \)

<table>
<thead>
<tr>
<th>Job</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0^A )</td>
<td>7/200</td>
<td>-</td>
<td>-</td>
<td>9/200</td>
</tr>
<tr>
<td>( t_1^B )</td>
<td>3/100</td>
<td>1/20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( t_2^C )</td>
<td>-</td>
<td>1/20</td>
<td>3/100</td>
<td>-</td>
</tr>
<tr>
<td>( t_3^D )</td>
<td>-</td>
<td>-</td>
<td>7/100</td>
<td>1/100</td>
</tr>
</tbody>
</table>
Outline

Problem definition

Algorithms evaluation
  Presentation
  Experiences
  Synthesis

Steady state for small batches
  Means of action
  Reduce period size
  Experiences

Conclusion and future works
efficiency = \frac{\text{makespan}_o}{\text{makespan}_r}

efficiency = \frac{N}{\rho \times \text{makespan}_r}

- \text{makespan}_o: \text{lower bound}
  - \text{makespan}_o: \frac{N}{\rho},
  - \text{reference time}

- \text{makespan}_r: \text{makespan} resulting from experience,

- Simulation results (SimGrid),

- Communications neglected.
Figure: Job $j_0$

Table: Grid $G_0$

Global results:
- Steady state tends toward optimal,
- GATS good for small batches, then collapses,
- *on-line* is constant.

Figure: Execution of batches $j_0$ on $G_0$
Figure: Job $j_1$

Table: Grid $G_0$

- Sames tasks as $j_0$,
- GATS collapses earlier (250 instances vs. 300).

Figure: Execution of batches $j_1$ on $G_0$
**Figure:** Job $j_2$

<table>
<thead>
<tr>
<th>Type</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
<th>$p_5$</th>
<th>$p_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>B</td>
<td>$\infty$</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>$\infty$</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

**Table:** Grid $G_0$

Steady state uses $p_6$ (A: 1000),

**Figure:** Execution of batches $j_2$ on $G_0$
The efficiency of GATS decreases starting at 200 instances.

Figure: Execution of batches $j_2$ on $G_1$
Outline

Problem definition

Algorithms evaluation
  Presentation
  Experiences
  Synthesis

Steady state for small batches
  Means of action
  Reduce period size
  Experiences

Conclusion and future works
**Experiences**

### Synthesis

<table>
<thead>
<tr>
<th>Small batches: 50</th>
<th>Medium batches: 100</th>
<th>Large batches: 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-line</td>
<td>Steady</td>
<td>GATS</td>
</tr>
<tr>
<td>0.93</td>
<td>0.93</td>
<td>0.99</td>
</tr>
</tbody>
</table>

**Table:** Mean efficiency

- **Synthesis:**
  - GATS: up to 200,
  - Steady state: from 500.

- **Time consumption for 1000 instances:**
  - steady state: 0.08s,
  - on-line: 35.04s,
  - GATS: 1799.68s,
Outline

Problem definition
Algorithms evaluation
  Presentation
  Experiences
  Synthesis
Steady state for small batches
  Means of action
  Reduce period size
  Experiences
Conclusion and future works
Steady state for small batches

Means of action

Aim: improve (decrease) the global *makespan* for small batches,

The steady state phase is optimal.

Schedule starting/ending phases on the heterogeneous platform:
- NP-Hard → find a good schedule,
- reduce the work in initialization/ending phases.

What can be done on starting/ending phases?
- Keeping steady state optimal:
  - re-organise affectations to reduce the period size,
  - resolve dependencies inside the period.
- Deterioration of steady state:
  - reduce the number of instances per periods
Steady state schedule

Figure: Job $j_3$

Table: Grid $G_2$
Outline

Problem definition
Algorithms evaluation
  Presentation
  Experiences
  Synthesis
Steady state for small batches
  Means of action
  Reduce period size
  Experiences
Conclusion and futur works
Steady state for small batches

Example

**Figure:** Job

**Table:** Cost

<table>
<thead>
<tr>
<th>Type</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
<td>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**Table:** Consumption

<table>
<thead>
<tr>
<th>Job</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0^A$</td>
<td>$7/200$</td>
<td>$-$</td>
<td>$-$</td>
<td>$9/200$</td>
</tr>
<tr>
<td>$t_1^B$</td>
<td>$3/100$</td>
<td>$1/20$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$t_2^C$</td>
<td>$-$</td>
<td>$1/20$</td>
<td>$3/100$</td>
<td>$-$</td>
</tr>
<tr>
<td>$t_3^D$</td>
<td>$-$</td>
<td>$-$</td>
<td>$7/100$</td>
<td>$1/100$</td>
</tr>
</tbody>
</table>

Period $= 200$. 
Steady state for small batches

Algorithm

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0^A$</td>
<td>7/200</td>
<td>-</td>
<td>-</td>
<td>9/200</td>
</tr>
<tr>
<td>$t_1^B$</td>
<td>3/100</td>
<td>1/20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$t_2^C$</td>
<td>-</td>
<td>1/20</td>
<td>3/100</td>
<td>-</td>
</tr>
<tr>
<td>$t_3^D$</td>
<td>-</td>
<td>-</td>
<td>7/100</td>
<td>1/100</td>
</tr>
</tbody>
</table>

Table: Consumptions: $cons$

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0^A$</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>$t_1^B$</td>
<td>6</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$t_2^C$</td>
<td>-</td>
<td>10</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>$t_3^D$</td>
<td>-</td>
<td>-</td>
<td>14</td>
<td>2</td>
</tr>
</tbody>
</table>

Table: Integer consumptions: $consInt$
Steady state for small batches

Initial steady state schedule $S$
- $P$: period, $P$ : LCM of the matrix denominators,
- $\rho$: throughput, $\rho = a/b$, reduced fraction.

Let $P_{\text{min}}$ be the minimum possible period
- $P_0 = b$,
- $P_{\text{min}} = \alpha \times P_0, \alpha \in [1, P/b]$
- Find $P_{\text{min}}$, that respect the constraints:
  - For lines $L_j$: $\sum_{i \in L_i} \text{cons}(i, j) = \rho$,
  - For columns $C_i$: $\sum_{j \in C_j} \text{cons}(i, j) \times w_{ij} < 1$

2nd “Scheduling in Aussois” workshop - 19/05/2008
Steady state for small batches

Example

**Figure:** Job

**Table:** Cost

<table>
<thead>
<tr>
<th>Type</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
<td>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**Table:** Consumption

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0^A$</td>
<td>7/200</td>
<td>-</td>
<td>-</td>
<td>9/200</td>
</tr>
<tr>
<td>$t_1^B$</td>
<td>3/100</td>
<td>1/20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$t_2^C$</td>
<td>-</td>
<td>1/20</td>
<td>3/100</td>
<td>-</td>
</tr>
<tr>
<td>$t_3^D$</td>
<td>-</td>
<td>-</td>
<td>7/100</td>
<td>1/100</td>
</tr>
</tbody>
</table>

**Table:** Modified consumption

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0^A$</td>
<td>1/25</td>
<td>-</td>
<td>-</td>
<td>1/25</td>
</tr>
<tr>
<td>$t_1^B$</td>
<td>1/50</td>
<td>3/50</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$t_2^C$</td>
<td>-</td>
<td>1/25</td>
<td>1/25</td>
<td>-</td>
</tr>
<tr>
<td>$t_3^D$</td>
<td>-</td>
<td>-</td>
<td>3/50</td>
<td>1/50</td>
</tr>
</tbody>
</table>
Steady state for small batches

Algorithm

Aim: maximize CD of the ConsInt matrix that respect the constraints,

- Optimisation problem with integers.
  - Constraints programming with finite domains (swi-prolog)
  - Exponential complexity but the problem is small

Algorithm 1: reducePeriod(cons: Matrix, cost: Matrix) : Matrix

\[
\begin{align*}
    cd & \leftarrow \text{periodLength}/\text{throughputDenominator}; \\
    \text{while } cd > 1 & \text{ do} \\
        \text{newCons} & : \text{Matrix}; \\
        \text{if } \text{newCons} \leftarrow \text{reorganize}(\text{cons}, \text{cost}, \text{cd}) & \text{ then} \\
            & \text{return newCons}; \\
        \text{else} \\
            & \text{cd} \leftarrow \text{cd} - 1; \\
        \text{end} \\
    \text{end} \\
    \text{return cons};
\end{align*}
\]
Outline

Problem definition
Algorithms evaluation
  Presentation
  Experiences
  Synthesis
Steady state for small batches
  Means of action
  Reduce period size
  Experiences
Conclusion and future works
Steady state for small batches

Metric

\[
\text{efficiency} = \frac{\text{makespan}_o}{\text{makespan}_r}
\]

\[
\text{efficiency} = \frac{\text{Batch size}}{(\text{rate} \times \text{makespan}_r)}
\]

Steady state rate is optimal

- time reference: \(N/\rho\),
- lower bound for optimal makespan (\(\text{makespan}_o\)),

\(\text{makespan}_r\): makespan of the algorithm.
Figure: Job $j_3$

Table: Grid $G_3$

Period: $16/200 \Rightarrow 4/50$ (/4),

Figure: Execution of batches $j_3$ on $G_2$
Figure: Job $j_3$

Table: Grid $G_4$

Figure: Execution of batches $j_3$ on $G_3$

$96/1200 \Rightarrow 4/50 (\times 24)$,
Figure: Job \( j_4 \)

<table>
<thead>
<tr>
<th>Type</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>C</td>
<td>( \infty )</td>
<td>10</td>
<td>10</td>
<td>( \infty )</td>
</tr>
<tr>
<td>D</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table: Grid \( G_3 \)

- 48/1320 \( \Rightarrow \) 4/110 (/12),
- starting: 194 instances,
- never in steady state.

Figure: Execution of batches \( j_4 \) on \( G_2 \)
Figure: Job $j_4$

Table: Grid $G_4$

- **Type**
  - A: 20 $\infty$ 20 20
  - B: 10 10 $\infty$ 10
  - C: 10 10 10 $\infty$
  - D: $\infty$ 10 10 10

12/330 $\Rightarrow$ 4/110 (/3),
starting: 38 instances,
2 full periods for a batch of 150 jobs,
Max efficiency difference: 1%.

Figure: Execution of batches $j_4$ on $G_3$
Outline

Problem definition
Algorithms evaluation
  Presentation
  Experiences
  Synthesis
Steady state for small batches
  Means of action
  Reduce period size
  Experiences
Conclusion and future works
Conclusion and futur works

Conclusion:
- Comparison of algorithms performances,
- Minimal period while keeping steady state,
- Large gain for small batches,
- Not always possible.

Futur works:
- Keeping an optimal steady state:
  - Comparison of dependencies resolutions.
- Deterioration of steady state:
  - Reduce the number of instances per periods.
Thanks for your attention.

[diakite,nicod,philippe]@lifc.univ-fcomte.fr
Initialization prepares all the dependencies needed before entering in steady state:

- For each task or communication in the period:
  - execute all the preceding tasks/communications in the graph

Ending: finish all the remaining tasks/communications,

- Common work with Loris Marchal (LIP)
- Reduce the number of tasks computed in the starting and ending phases.
Dependencies resolution

Figure: Job $j_3$

<table>
<thead>
<tr>
<th>Type</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
<td>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table: Grid $G_5$

Figure: Execution of batches $j_3$ on $G_5$
Dependencies resolution

Initial Schedule:

Schedule with less dependencies:
Dependencies resolution

Figure: Platform

Figure: Job
1 suppressed dependency = 1 subgraph less,
Balance starting and ending,
How to optimize dependencies resolution?
  How to measure the gain?
  The more ... the less
  Are there better dependencies?
Max number of dependencies: two-partition
Reorganize the periodic schedule: heuristics
Find a good scheduling algorithm for starting and ending phases.
Link number of jobs $\leq$ period size