Algorithms and scheduling techniques for heterogeneous platforms

Yves Robert

École Normale Supérieure de Lyon
Yves.Robert@ens-lyon.fr
http://graal.ens-lyon.fr/~yrobert

joint work with
Olivier Beaumont, Anne Benoit, Larry Carter, Henri Casanova,
Jack Dongarra, Jeanne Ferrante, Arnaud Legrand, Loris Marchal, Frédéric Vivien

Cetraro, July 2, 2008
Algorithm design and scheduling already difficult with homogeneous machines

On heterogeneous platforms, it gets worse
Outline

1. Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2. Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3. Conclusion
Outline

1 Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2 Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3 Conclusion
Outline

1. Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2. Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3. Conclusion
Use $q \times q$ blocks to harness efficiency of Level 3 BLAS.
Matrix product on a $3 \times 4$ homogeneous 2D-grid
Algorithm on (heterogeneous) 2D grids

Matrix product on a $3 \times 4$ heterogeneous 2D-grid
2D load balancing (1/2)

Objective: \( \max \sum_{i=1}^{p_1} r_i \times \sum_{j=1}^{p_2} c_j \) s.t. \( r_i \times w_{ij} \times c_j \leq 1 \)

Maximize total number of elements processed within one time unit
2D load balancing (2/2)

Given $p$ processors, how to arrange them along a 2D grid of size $p_1 \times p_2 \leq p$ ...

... so as to optimally load-balance the work of the processors

- Search among all possible arrangements of $p_1 \times p_2$ processors as a $p_1 \times p_2$ grid
- For each arrangement, solve optimization problem
  - NP-hard 😞
Given $p$ processors, how to arrange them along a 2D grid of size $p_1 \times p_2 \leq p$ ...

... so as to optimally load-balance the work of the processors

- Search among all possible arrangements of $p_1 \times p_2$ processors as a $p_1 \times p_2$ grid
- For each arrangement, solve optimization problem
- **NP-hard 😞**
Matrix product on heterogeneous clusters

Matrix product with 13 heterogeneous processors
Optimization

How to compute the \textit{area} and \textit{shape} of the \( p \) rectangles?

- **Load-balancing computations** assign \textit{areas} proportional to speeds
- **Minimizing communication overhead** choose \textit{shapes}:
  - total communication volume
    \[
    \hat{C} = \sum_{i=1}^{p} (h_i + v_i)
    \]
    \textit{sum} of the half perimeters of the \( p \) rectangles
  - for parallel communications:
    \[
    \hat{M} = \max_{i=1}^{p} (h_i + v_i)
    \]

Both problems NP-hard 😞
How to compute the *area* and *shape* of the $p$ rectangles?

- **Load-balancing computations** assign *areas* proportional to speeds
- **Minimizing communication overhead** choose *shapes*:
  - total communication volume
    $\hat{C} = \sum_{i=1}^{p} (h_i + v_i)$
    *sum* of the half perimeters of the $p$ rectangles
  - for parallel communications:
    $\hat{M} = \max_{i=1}^{p} (h_i + v_i)$

- **Both problems NP-hard 😞**
Outline

1 Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2 Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3 Conclusion
Platform model

- **Star network** master $M$ and $p$ workers $P_i$
- $X.w_i$ time-units for $P_i$ to execute a task of size $X$
- $X.c_i$ time-units for $M$ to send/rcv msg of size $X$ to/from $P_i$
- Master has no processing capability
- Enforce **one-port** model

**Memory limitation:** only $m_i$ buffers available for $P_i$
→ at most $m_i$ blocks simultaneously stored on worker
Strategy for homogeneous processors

- **Natural memory management**
  - Assign one-third for each of $A$, $B$ and $C$
  - **Example**: $m = 21 \Rightarrow 7$ buffers per matrix

- **Optimal memory management**
  - Find largest $\mu$ s.t. $1 + \mu + \mu^2 \leq m$
  - Assign 1 buffer to $A$, $\mu$ to $B$ and $\mu^2$ to $C$
  - **Example**: $m = 21 \Rightarrow 1$ for $A$, 4 to $B$ and 16 to $C$
Strategy for homogeneous processors

- Natural memory management
  - Assign one-third for each of A, B and C
  - **Example:** $m = 21 \Rightarrow 7$ buffers per matrix

- Optimal memory management
  - Find largest $\mu$ s.t. $1 + \mu + \mu^2 \leq m$
  - Assign 1 buffer to A, $\mu$ to B and $\mu^2$ to C
  - **Example:** $m = 21 \Rightarrow 1$ for A, 4 to B and 16 to C
Example with $m = 21$
Example with $m = 21$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example with $m = 21$
Example with $m = 21$
Example with $m = 21$

<table>
<thead>
<tr>
<th></th>
<th>$C_{11}$</th>
<th>$C_{12}$</th>
<th>$C_{13}$</th>
<th>$C_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{11}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_{11}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{21}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{22}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{23}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{24}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{31}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{32}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{33}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{34}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{41}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{42}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{43}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{44}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example with $m = 21$

<table>
<thead>
<tr>
<th></th>
<th>$A_{11}$</th>
<th>$B_{11}$</th>
<th>$B_{12}$</th>
<th>$C_{11}$</th>
<th>$C_{12}$</th>
<th>$C_{13}$</th>
<th>$C_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$C_{12}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td>$C_{23}$</td>
<td>$C_{24}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$C_{31}$</td>
<td>$C_{32}$</td>
<td>$C_{33}$</td>
<td>$C_{34}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$C_{41}$</td>
<td>$C_{42}$</td>
<td>$C_{43}$</td>
<td>$C_{44}$</td>
</tr>
</tbody>
</table>
Example with $m = 21$

\[
\begin{array}{cccc}
A_{11} & C_{11} & C_{12} & C_{13} & C_{14} \\
C_{21} & C_{22} & C_{23} & C_{24} \\
C_{31} & C_{32} & C_{33} & C_{34} \\
C_{41} & C_{42} & C_{43} & C_{44} \\
B_{11} & B_{12} & B_{13} & B_{14}
\end{array}
\]
Example with $m = 21$

<table>
<thead>
<tr>
<th></th>
<th>$B_{11}$</th>
<th>$B_{12}$</th>
<th>$B_{13}$</th>
<th>$B_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{21}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$C_{11}$</th>
<th>$C_{12}$</th>
<th>$C_{13}$</th>
<th>$C_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{21}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{22}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{23}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{24}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$C_{31}$</th>
<th>$C_{32}$</th>
<th>$C_{33}$</th>
<th>$C_{34}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{41}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{42}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{43}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{44}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example with $m = 21$

<table>
<thead>
<tr>
<th></th>
<th>$A_{31}$</th>
<th>$x$</th>
<th>$x$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_{11}$</td>
<td>$B_{12}$</td>
<td>$B_{13}$</td>
<td>$B_{14}$</td>
<td></td>
</tr>
<tr>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$C_{13}$</td>
<td>$C_{14}$</td>
<td></td>
</tr>
<tr>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td>$C_{23}$</td>
<td>$C_{24}$</td>
<td></td>
</tr>
<tr>
<td>$C_{31}$</td>
<td>$C_{32}$</td>
<td>$C_{33}$</td>
<td>$C_{34}$</td>
<td></td>
</tr>
<tr>
<td>$C_{41}$</td>
<td>$C_{42}$</td>
<td>$C_{43}$</td>
<td>$C_{44}$</td>
<td></td>
</tr>
</tbody>
</table>
Example with $m = 21$

<table>
<thead>
<tr>
<th></th>
<th>$B_{11}$</th>
<th>$B_{12}$</th>
<th>$B_{13}$</th>
<th>$B_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$C_{13}$</td>
<td>$C_{14}$</td>
</tr>
<tr>
<td></td>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td>$C_{23}$</td>
<td>$C_{24}$</td>
</tr>
<tr>
<td></td>
<td>$C_{31}$</td>
<td>$C_{32}$</td>
<td>$C_{33}$</td>
<td>$C_{34}$</td>
</tr>
<tr>
<td>$A_{41}$</td>
<td>$C_{41}$</td>
<td>$C_{42}$</td>
<td>$C_{43}$</td>
<td>$C_{44}$</td>
</tr>
</tbody>
</table>
Example with $m = 21$

<table>
<thead>
<tr>
<th>A_{12}</th>
<th>x</th>
<th>C_{11}</th>
<th>C_{12}</th>
<th>C_{13}</th>
<th>C_{14}</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>C_{21}</td>
<td>C_{22}</td>
<td>C_{23}</td>
<td>C_{24}</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>C_{31}</td>
<td>C_{32}</td>
<td>C_{33}</td>
<td>C_{34}</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>C_{41}</td>
<td>C_{42}</td>
<td>C_{43}</td>
<td>C_{44}</td>
</tr>
</tbody>
</table>
Example with $m = 21$
Example with $m = 21$

<table>
<thead>
<tr>
<th>$B_{t1}$</th>
<th>$B_{t2}$</th>
<th>$B_{t3}$</th>
<th>$B_{t4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$C_{11}$</th>
<th>$C_{12}$</th>
<th>$C_{13}$</th>
<th>$C_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td>$C_{23}$</td>
<td>$C_{24}$</td>
</tr>
<tr>
<td>$C_{31}$</td>
<td>$C_{32}$</td>
<td>$C_{33}$</td>
<td>$C_{34}$</td>
</tr>
<tr>
<td>$C_{41}$</td>
<td>$C_{42}$</td>
<td>$C_{43}$</td>
<td>$C_{44}$</td>
</tr>
</tbody>
</table>

$A_{4t}$
Algorithm with identical workers

\[ c = 2, \ w = 4.5, \ \mu = 4, \ t = 100, \ \text{enroll} \ \not P = 5 \ \text{workers} \]
Algorithm with identical workers

\[ c = 2, \ w = 4.5, \ \mu = 4, \ t = 100, \text{ enroll } \mathcal{P} = 5 \text{ workers} \]
Algorithm with identical workers

\[ c = 2, \ w = 4.5, \ \mu = 4, \ t = 100, \ \text{enroll } \mathfrak{P} = 5 \ \text{workers} \]
Algorithm with identical workers

\[ c = 2, \ w = 4.5, \ \mu = 4, \ t = 100, \ \text{enroll} \ \mathcal{P} = 5 \ \text{workers} \]
Algorithm with identical workers

$c = 2, \ w = 4.5, \ \mu = 4, \ t = 100, \ enroll \ P = 5 \ workers$
Algorithm with identical workers

\[ c = 2, \ w = 4.5, \ \mu = 4, \ t = 100, \text{ enroll } \mathcal{P} = 5 \text{ workers} \]
Performance

Communication-to-computation ratio:

\[
\frac{2}{t} + \frac{2}{\mu} \rightarrow \frac{2}{\sqrt{m}}
\]

- Close to lower bound
- Enroll \( \mathcal{P} \leq p \) workers, where

\[
\mathcal{P} = \left\lceil \frac{\mu w}{2c} \right\rceil
\]

In the example, \( \mathcal{P} = \lceil 4.5 \rceil \)

- Typically, \( c = q^2 \tau_c \) and \( w = q^3 \tau_a \)
- \( \rightarrow \) resource selection \( \mathcal{P} = \left\lceil \mu q \frac{\tau_a}{2\tau_c} \right\rceil \)
Algorithms for heterogeneous platforms

- Different memory patterns for workers
- Complicated resource selection
- Complicated communication ordering
- Complicated schedule
- ...but it works fine 😊 (see experiments in papers)
Algorithms for heterogeneous platforms

- Different memory patterns for workers
- Complicated resource selection
- Complicated communication ordering
- Complicated schedule
- ... but it works fine 😊 (see experiments in papers)
Matrix product / LU for multicores?

- Multicore $\geq 100$ processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:
  Add yet another level of blocking?
Matrix product / LU for multicores?

- Multicore $\geq 100$ processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:
  
  Add yet another level of blocking?
Matrix product / LU for multicores?

- Multicore $\geq 100$ processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:
  
  *Add yet another level of blocking?*
Matrix product / LU for multicores?

- Multicore ≥ 100 processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:
  Add yet another level of blocking?
Matrix product / LU for multicores?

- Multicore $\geq 100$ processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:  
  **Add yet another level of blocking?**
Outline

1 Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2 Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3 Conclusion
Outline

1. Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2. Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3. Conclusion
Model

- Task graph with $n$ tasks $T_1, \ldots, T_n$.
- Platform with $p$ heterogeneous processors $P_1, \ldots, P_p$.
- Computation costs:
  - $w_{iq} =$ execution time of $T_i$ on $P_q$
  - $\overline{w_i} = \frac{\sum_{q=1}^{p} w_{iq}}{p}$ average execution time of $T_i$
  - particular case: consistent tasks $w_{iq} = w_i \times \gamma_q$
- Communication costs:
  - data$(i,j)$: data volume for edge $e_{ij} : T_i \rightarrow T_j$
  - $v_{qr}$: communication time for unit-size message from $P_q$ to $P_r$ (zero if $q = r$)
  - $\text{com}(i,j,q,r) =$ data$(i,j) \times v_{qr}$ communication time from $T_i$ executed on $P_q$ to $P_j$ executed on $P_r$
  - $\overline{\text{com}_{ij}} = \text{data}(i,j) \times \frac{\sum_{1 \leq q,r \leq p,q \neq r} v_{qr}}{p(p-1)}$ average communication cost for edge $e_{ij} : T_i \rightarrow T_j$
Constraints

**Dependences** For $e_{ij} : T_i \rightarrow T_j$, $q = \text{alloc}(T_i)$ and $r = \text{alloc}(T_j)$:

$$\sigma(T_i) + w_{iq} + \text{com}(i, j, q, r) \leq \sigma(T_j)$$

**Resources** If $q = \text{alloc}(T_i) = \text{alloc}(T_j)$, then

$$(\sigma(T_i) + w_{iq} \leq \sigma(T_j)) \text{ or } (\sigma(T_j) + w_{jq} \leq \sigma(T_i))$$

**Makespan**

$$\max_{1 \leq i \leq n} (\sigma(T_i) + w_{i, \text{alloc}(T_i)})$$
HEFT: Heterogeneous Earliest Finish Time

- **Priority level:**
  - \( \text{rank}(T_i) = \overline{w}_i + \max_{T_j \in \text{Succ}(T_i)} (\text{com}_{ij} + \text{rank}(T_j)) \),
  - where \( \text{Succ}(T) \) is the set of successors of \( T \)
  - Recursive computation by bottom-up traversal of the graph

- **Allocation**
  - For current task \( T_i \), determine best processor \( P_q \):
    - minimize \( \sigma(T_i) + w_{iq} \)
  - Enforce constraints related to communication costs
  - Insertion scheduling: look for \( t = \sigma(T_i) \) s.t. \( P_q \) is available during interval \([t, t + w_{iq}]\)
What’s wrong?

- 😊 Nothing (still may need to map a DAG onto a platform!)
- 😞 Too many parameters to instantiate
- 😞 Absurd communication model: clique + unbounded bandwidth
- 😞 Wrong metric: need to relax makespan minimization objective
What’s wrong?

- 😊 Nothing (still may need to map a DAG onto a platform!)
- 😞 Too many parameters to instantiate
- 😞 Absurd communication model: clique + unbounded bandwidth
- 😞 Wrong metric: need to relax makespan minimization objective
What’s wrong?

- 😊 Nothing (still may need to map a DAG onto a platform!)
- 😞 Too many parameters to instantiate
- 😞 Absurd communication model: **clique + unbounded bandwidth**
- 😞 Wrong metric: need to relax makespan minimization objective
What’s wrong?

- 😊 Nothing (still may need to map a DAG onto a platform!)
- 😞 Too many parameters to instantiate
- 😞 Absurd communication model: **clique + unbounded bandwidth**
- 😞 Wrong metric: need to relax makespan minimization objective
Outline

1 Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2 Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3 Conclusion
Example

Data starts here

My computer

Intermediate nodes can compute too

Internet Gateway

Cluster Host

Super-computer

Partner site

Participating PC's and workstations
Example

A is the root of the tree; all tasks start at A

Time for computing one task in C

Time for sending one task from A to B

Yves.Robert@ens-lyon.fr July 2, 2008
Example

A compute
A send

B receive
B compute

C receive
C compute
C send

D receive
D compute

Time→
Example
Example
Example
Example

A compute
A send
B receive
B compute
C receive
C compute
C send
D receive
D compute

Time →
Example

Steady-state: 7 tasks every 6 time units

Start-up

Repeated pattern

Clean-up

Yves.Robert@ens-lyon.fr July 2, 2008
Rule of the game

- Master sends tasks to workers **sequentially**, and without preemption
- Full computation/communication overlap for each worker
- Worker $P_i$ receives a task in $c_i$ time-units
- Worker $P_i$ processes a task in $w_i$ time-units
Worker $P_i$ executes $\alpha_i$ tasks per time-unit

- Computation: $\alpha_i w_i \leq 1$
- Communications: $\sum_i \alpha_i c_i \leq 1$
- Objective: maximize throughput

$$\rho = \sum_i \alpha_i$$
Solution

- Faster-communicating workers first: $c_1 \leq c_2 \leq \ldots$
- Make full use of first $q$ workers, where $q$ largest index s.t.
  $$\sum_{i=1}^{q} \frac{c_i}{w_i} \leq 1$$
- Make partial use of next worker $P_{q+1}$
- **Discard** other workers

**Bandwidth-centric strategy**
- Delegate work to the fastest communicating workers
- It doesn’t matter if these workers are computing slowly
- Slow workers will not contribute much to overall throughput
Example

Fully active

Discarded

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Communication</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 tasks to $P_1$</td>
<td>$6c_1 = 6$</td>
<td>$6w_1 = 18$</td>
</tr>
<tr>
<td>3 tasks to $P_2$</td>
<td>$3c_2 = 6$</td>
<td>$3w_2 = 18$</td>
</tr>
<tr>
<td>2 tasks to $P_3$</td>
<td>$2c_3 = 6$</td>
<td>$2w_3 = 2$</td>
</tr>
</tbody>
</table>

11 tasks every 18 time-units ($\rho = 11/18 \approx 0.6$)
Example

Compare to purely greedy (demand-driven) strategy!
5 tasks every 36 time-units ($\rho = 5/36 \approx 0.14$)

Even if resources are cheap and abundant, resource selection is key to performance.
Extension to trees

- Fully used node
- Partially used node
- Idle node

Resource selection based on **local** information (children)
Does this really work?

- Can we deal with arbitrary platforms (including cycles)?
- Can we deal with return messages?
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?
Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages?
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?
Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages?
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?
Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages? Yes
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?
Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages? Yes
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?
Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages? Yes
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)? Yes, I mean, almost!
LP formulation still works well . . .

Conservation law

\[ \forall m, n \sum_{j} \text{sent}(P_j \rightarrow P_i, e_{mn}) + \text{executed}(P_i, T_m) = \text{executed}(P_i, T_n) + \sum_{k} \text{sent}(P_i \rightarrow P_k, e_{mn}) \]

Computations

\[ \sum_{m} \text{executed}(P_i, T_m) \times \text{flops}(T_m) \times w_i \leq 1 \]

Outgoing communications

\[ \sum_{m, n} \sum_{j} \text{sent}(P_j \rightarrow P_i, e_{mn}) \times \text{bytes}(e_{mn}) \times c_{ij} \leq 1 \]
... but schedule reconstruction is harder

- 😊 Actual (cyclic) schedule obtained in polynomial time
- 😊 Asymptotic optimality
- 😞 A couple of practical problems (large period, # buffers)
- 😞 No local scheduling policy
The beauty of steady-state scheduling

**Rationale**
Maximize throughput (total load executed per period)

**Simplicity**
Relaxation of makespan minimization problem
- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity per time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving from or sending to which neighbor?

**Efficiency**
Optimal throughput $\Rightarrow$ optimal schedule (up to a constant number of tasks)

Periodic schedule, described in compact form
$\Rightarrow$ compiling a loop instead of a DAG!
Outline

1. Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2. Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3. Conclusion
Scheduling multiple applications

Investigate scenarios in which multiple applications are simultaneously executed on the platform
⇒ competition for CPU and network resources

- Large complex platform: several clusters and backbone links
- One (divisible load) application running on each cluster
- Which fraction of the job to delegate to other clusters?
- Applications have ≠ communication-to-computation ratios
- How to ensure fair scheduling and good resource utilization?
Minimize \( \min_k \left\{ \frac{\alpha_k}{\pi_k} \right\} \),

under the constraints

\[
\begin{align*}
(1a) \quad & \forall C^k, \quad \sum_l \alpha_{k,l} = \alpha_k \\
(1b) \quad & \forall C^k, \quad \sum_l \alpha_{l,k} \cdot \tau_l \leq s_k \\
(1c) \quad & \forall C^k, \quad \sum_{l \neq k} \alpha_{k,l} \cdot \delta_k + \sum_{j \neq k} \alpha_{j,k} \cdot \delta_j \leq g_k \\
(1d) \quad & \forall l_i, \quad \sum_{l_i \in L_{k,l}} \beta_{k,l} \leq \text{max-connect}(l_i) \\
(1e) \quad & \forall k, l, \quad \alpha_{k,l} \cdot \delta_k \leq \beta_{k,l} \times g_{k,l} \\
(1f) \quad & \forall k, l, \quad \alpha_{k,l} \geq 0 \\
(1g) \quad & \forall k, l, \quad \beta_{k,l} \in \mathbb{N}
\end{align*}
\]
Approach

- Solution to *rational* linear problem as comparator/upper bound
- Several heuristics, greedy and LP-based
- Use Tiers as topology generator, and then $S_{\text{IMGGRID}}$
Approach (cont’d)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K )</td>
<td>( 5, 7, \ldots, 90 )</td>
</tr>
<tr>
<td>( \log(bw(l_k)), \log(g_k) )</td>
<td>normal (( mean = \log(2000) ), ( std = \log(10) ))</td>
</tr>
<tr>
<td>( s_k )</td>
<td>uniform, 1000 — 10000</td>
</tr>
<tr>
<td>max-connect, ( \delta_k ), ( \tau_k ), ( \pi_k )</td>
<td>uniform, 1 — 10</td>
</tr>
</tbody>
</table>

Platform parameters used in simulation
Hints for implementation

- Participants sharing resources in a Virtual Organization
- Centralized broker managing applications and resources
- Broker gathers all parameters of LP program
- Priority factors
- Various policies and refinements possible
  ⇒ e.g. fixed number of connections per application
Outline

1 Algorithms
   - Matrix product (ScaLAPACK oriented)
   - Matrix product (MATLAB oriented)

2 Scheduling
   - Scheduling DAGs
   - Steady-state scheduling
   - Multiple applications

3 Conclusion
Tools for the road

- Forget absolute makespan minimization
- Divisible load (fractional tasks)
- Linear programming: absolute bound to assess heuristics

- Resource selection mandatory
- Stochastic models (uncertainties / failures)
- From DAG to workflow (collection of pipelined DAGs)
- Single workflow: throughput / latency / power / robustness
- Several workflows: max-min fairness, MAX stretch
Tools for the road

- Forget absolute makespan minimization
- Divisible load (fractional tasks)
- Linear programming: absolute bound to assess heuristics
- Resource selection mandatory
- Stochastic models (uncertainties / failures)
- From DAG to workflow (collection of pipelined DAGs)
- Single workflow: throughput / latency / power / robustness
- Several workflows: max-min fairness, MAX stretch

Too complicated?! 😞
Tools for the road

- Forget absolute makespan minimization
- Divisible load (fractional tasks)
- Linear programming: absolute bound to assess heuristics

- Resource selection mandatory
- Stochastic models (uncertainties / failures)
- From DAG to workflow (collection of pipelined DAGs)
- Single workflow: throughput / latency / power / robustness
- Several workflows: max-min fairness, MAX stretch

Think before coding! 😊