Scheduling on parallel platforms

Denis Trystram
trystram@imag.fr
October 2009
Content

• Context and Introduction
• Definitions and basic results
• Communication Delays
• Taking into account new characteristics
• Parallel Tasks
• On-line and new directions
Taxinomy of Applications

- Regular
  - off-line
  - clairvoyant
    - off-line (batch)
    - mixed
  - unpredictable (not clairvoyant)
    - on-line

- Irregular

- multi-applications
  - on-line
Let $G=(V,E)$ be a weighted graph

$(i,j)\in E$ iff $i<<j$ (partial order)

The vertices are weighted by the execution times.

The arcs are weighted by the data to be transferred from a task to another.
Example:
computing $C = AB$ by Strassen

Matrices $A$ and $B$ are partitioned by quadrant.

$$C_{12} = A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22}$$
Identifying the tasks

\[ T_1 = A_{11} + A_{12}; \quad T_2 = A_{21} - A_{11}; \]
\[ T_3 = A_{12} - A_{22}; \quad T_4 = A_{21} + A_{22}; \]
\[ T_5 = A_{11} + A_{22}; \]
\[ U_1 = B_{11} + B_{22}; \quad U_2 = B_{11} + B_{12}; \]
\[ U_3 = B_{21} - B_{11}; \quad U_4 = B_{12} - B_{22}; \]
\[ U_5 = B_{21} + B_{22}; \]
\[ P_1 = T_5 * U_4; \quad P_2 = T_4 * B_{11}; \]
\[ P_3 = A_{11} * U_4; \quad P_4 = A_{22} * U_3; \]
\[ P_5 = T_1 * B_{22}; \quad P_6 = T_2 * U_2; \]
\[ P_7 = T_3 * U_5; \]
\[ C_{11} = P_1 + P_4 - P_5 + P_7; \quad C_{12} = P_3 + P_5; \]
\[ C_{21} = P_2 + P_4; \quad C_{22} = P_1 + P_3 - P_2 + P_6; \]
Strassen’s Task Graph
Scheduling: Formal Definition

The problem of scheduling graph $G = (V,E)$ weighted by function $p$ on $m$ processors:

(without communication)

Determine the pair of functions $(\text{date},\text{proc})$ subject to:

• respect of precedences

\[ \forall (i,j) \in E : \text{date}(j) \geq \text{date}(i) + p(i,\text{proc}(i)) \]

• objective: to minimize the makespan $C_{\text{max}}$
3 fields notation

[Graham,Lenstra-Lageweg-Veltman1990]

b1|b2|b3
[Lenstra-Lageweg-Veltman, 1990]

b1|b2|b3

• b1 - resources and model
[Lenstra-Lageweg-Veltman, 1990]

b1 | b2 | b3

• b1 - resources and model
• b2 - graph and schedule
[Lenstra-Lageweg-Veltman, 1990]

\begin{itemize}
  \item b1 - resources and model
  \item b2 - graph and schedule
  \item b3 - objective
\end{itemize}
[Lenstra-Lageweg-Veltman, 1990]

b1|b2|b3

• b1 - resources and model
• b2 - graph and schedule
• b3 - objective

Example: $P_\infty |prec,pj|C_{\text{max}}$
Parameters of a Problem

• b1 - implicit, BSP, LogP, $P^\infty$, P or Pm, Q, R
• b2 - prec, tree, diamond / dup, pmtn, pj, Cij
• b3 - $C_{\text{max}}, \sum C_i$, overhead, stretch
Example
Scheduling without communication \((m=3)\)
Theoretical Models

**PRAM:** modèles de référence pour la classification.

**Shared-memory:** ordonnancement pur, sans délais de communication. Grain fin et faiblement couplé.

**Distributed-memory:** prise en compte des communications (UET-UCT) explicites et modèles élargis (linéaires, LogP, etc..).

**Grappes et Grilles:** nouveaux paramètres.
Central Scheduling Problem

\[ P \mid \text{prec, pj} \mid C_{\text{max}} \] is NP-hard [Ulmann75]

Thus, we are looking for good heuristics.

- Competitive ratio \( r \):

maximum over all instances of \( \frac{\omega}{\omega^*} \)

The schedule \( S \) is said \( \rho \)-competitive iff \( r(\sigma) \leq \rho \)
Some results

Pinf | prec, pj | Cmax is polynomial (longest path)
Pm | prec, pj=1 | Cmax is still open for m>2
P2 | prec, pj=1 | Cmax is polynomial

[Coffman-Graham72]
List scheduling

**Principle:** build first the list of ready tasks and execute them with any greedy policy (in any order when they are available).

\[ Pm \mid prec, pj \mid C_{\text{max}} \text{ is 2-competitive} \]
Analysis of list scheduling

We start from the end of the schedule:

\[ \omega = \frac{W + \text{idle}}{m} \]

where \( W \) is the total work

The idea of the proof is to bound the term \( \text{idle} \)
While there exist some time slots with idle periods:
there is one active task which is linked with Tj
We continue from $T_i$ until it remains no idle time.
Proof:

\[
idle \leq (m-1)l_{ch} \leq (m-1)t_{\infty}
\]

\[
\frac{W}{m} \leq \omega^*
\]

\[
\omega \leq \omega^* + \frac{m-1}{m}t_{\infty}
\]

As the critical path is also a lower bound of the optimum:

\[
\omega \leq \left(2 - \frac{1}{m}\right)\omega^*
\]
Worst case

The bound is tight:

Consider \((m-1)m\) UET tasks and 1 task of length \(m\)

\[
\omega = 2m - 1 \\
\omega^* = m
\]
Anomalies [Graham]

Weights: (4,2,2,5,5,10,10)

C=14
Anomalies [Graham]

All weights have one unit less:

(3,1,1,4,4,9,9)

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>C=20</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>
Lower bounds

Basic tool:

Theorem of impossibility [Lenstra-Shmoys’95]

- given a scheduling problem and an integer \( c \), if it is NP-complete to schedule this problem in less than \( c \) times, then there is no schedule with a competitive ratio lower than \( (c+1)/c \).
Application

Proposition

The problem of deciding (for any UET graph) if there exists a valid schedule of length at most 3 is NP-complete.

Proof: by reduction from CLIQUE
Application

Proposition

The problem of deciding (for any UET graph) if there exists a valid schedule of length at most 3 is NP-complete.

Proof: by reduction from CLIQUE

Corollary: a lower bound for the competitive ratio of $Pm|\text{prec,}pj=1|C_{\text{max}}$ is $4/3$. 
(finer) Upper Bound

Consider problem \( P | \text{prec}, pj=1 | C_{\text{max}} \)

**Proposition**

There exists a (list-)algorithm whose performance guarantee is \( 2-2/m \) [Lam-Sethi,77] [Braschi-Trystram,94].

**Proof** adequate labeling of the tasks plus a priority based on the critical path.
Taking communications into account: the delay model

Introduced by [Rayward-Smith, 87]

- Total overlap of communications by local computations
- Possible duplication
- Simplified communications (unitary in the basic paper)
- No preemption allowed
Formal Definition

The problem of scheduling graph $G = (V, E)$ weighted by function $p$ on $m$ processors:

(with communication)

Determine the pair of functions $(\text{date,proc})$ subject to:

- respect of precedences

$\forall (i, j) \in E : date(j) \geq date(i) + p(i, proc(i)) + c(i, j)$

- objective: to minimize the makespan $C_{\text{max}}$
Basic delay model

Comparing with no communication:

• Handling explicitly the communications is harder than the basic scheduling model
Scheduling with small delay with and without duplication
Scheduling with UCT delay with and without duplication
Brent’s Lemma

• Property:

let $\rho$ be the competitive ratio of an algorithm with an unbounded number of processors. There exists an algorithm with performance ratio $\rho + 1$ for an arbitrary number of processors.
Principle

Gantt chart for $m^*$ processors

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
</table>
m processors
m processors
Proof

\[ \omega_m \leq \omega^*_\infty + \omega_\infty \]

(Similar to Graham’s bound)

\[ \omega_\infty \leq \rho \omega^*_\infty \]

\[ \omega^*_\infty \leq \omega^*_m \]

Thus, \( \omega_m \leq (\rho + 1) \omega^*_m \)
Consequences: trivial Upper Bound

• As $P_{\text{inf}} | \text{prec, pj}=1| C_{\text{max}}$ is optimal (competitive ratio of 1), then:

$P | \text{prec, pj}=1 | C_{\text{max}}$ is 2-competitive.

• As $P_{\text{inf}} | \text{dup, prec, pj, cij}| C_{\text{max}}$ is 2-competitive, then:

$P | \text{dup, prec, pj, cij = 1} | C_{\text{max}}$ is 3-competitive
List scheduling with communication delays

Solution for UET and UCT [Rayward-Smith]:
3-competitive algorithm.

Solution for general graphs:

The principle is to add a term proportional to the sum of the communications on the longest path [Hwang-Chow-Anger-Lee,89]. This term is not bounded.
More sophisticated algorithms than list-algorithms

Formulation of $P|\text{prec,}pj=1,ci_j=1|C_{\text{max}}$ as a ILP.

$X_{ij}$ are the decision variables 0 if task allot(i)=allot(j)
Solving as an ILP

Objective: minimize $(C)$

Constraints:

$\forall i \in V, date(i) + 1 \leq C$

$\forall i \in V, date(i) \geq 0$

$\forall (i,j) \in E, date(i) + 1 + X_{i,j} \geq date(j)$

$\sum_{j} X_{i,j} \geq \text{deg}(i)$

$X_{i,j} = 0, 1$
Solving as an ILP

Solve the LP with $x_{ij}$ real numbers in $[0,1]$ and then, relax the solution: $x_{ij} < 0.5$ are set to 0, the others are set to 1

Property: this algorithm is $4/3$-competitive.
Clustering Algorithms

Principle: unbounded number of processors.
Starting from the smallest granularity, the tasks are gathered into subsets of tasks.

Property:
Critical path or maximum independent sets.
Influence of the duplication

\[ P_{inf|prec,pj,cij \leq 1,\text{dup}|C_{max}} \]

is polynomial [Colin-Chretienne,90]

Idea: Find a spanning tree of minimum (local-) weights and schedule it by duplicating all the leaves.
Colin-Chrétienne
Duplication with a fixed number of processors

\[ P|\text{prec},pj=1,cij=1,\text{dup}|C_{\text{max}} \]

is 2-competitive [Hanen-Munier,97]

**Idea:** by applying a list scheduling with duplication of parts of paths.
Synthesis

small communication delays

unbounded number of proc.

no duplication

- trees, SP, bipartite polynomial
- UET-UCT NP-hard

duplication

- polynomial

m processors

UET-UCT NP-hard

- trees, bipartite NP-hard
- interval order polynomial
Scheduling with large delay

This problem is harder than with small communication delay

No competitive algorithm is known at this time with a constant ratio (linear in the granularity factor)
Consider $P \mid prec, pj=1, c>1 \mid C_{max}$

The best lower bound known at this time is $1+1/(g+3)$ [Bampis-Gianakos-Konig,98]

Practically, if $g<<1$ not interesting...
Large communication delays
upper bound

Consider again $P | prec, pj=1, c>1 | C_{max}$

The best upper bound known at this time is $(c+2)$ [Bampis-Gianakos-Konig, 97].

Another way to obtain this result is the trivial (list) algorithm which starts with no communication and systematically insert a communication between the computation steps...
Synthèse

grands délais de communication

infinité de processeurs
- duplication $\pi > 1$ et $c > 1$
  - NP-difficile
- pas de duplication

m processeurs
- arbres binaires complets et $m = 2$
  - polynomial
- arbres binaires $\pi > 1$ et $c > 1$ et $m = 2$
  - NP-difficile

biparti
- polynomial

arbres
- NP-difficile
Two natural extensions of the delay models are towards uniform (Q) and unrelated (R) processors.

NP-hard for very simple problems

NP-hard for 1 machine plus a set of (m-1) identical machines
Scheduling independent chains

Qm|chains,pj=1,c=1|Cmax is strongly NP-hard while Pm|chains,pj=1,c=1|Cmax is polynomial (linear).
Example: scheduling chains on 2 processors \((v_1=1, v_2=2)\).

\[\omega \geq \max \left( \frac{v_2(n_1+n_2)}{v_1+v_2}, n_1 \right) = 10\]

Idea: compute the maximum number of tasks to allocate to the slowest processor.
\[\alpha v_2 + n_1 - \alpha < \omega^*\]
\[\alpha = 2\]
Alternative models: BSP

BSP is a programming paradigm [Valiant,90] whose principle is a series of independent steps of computations and communication-synchronization.
Alternative models: BSP

BSP is a programming paradigm [Valiant, 90] whose principle is a series of independent steps of computations and communication-synchronization.

Scheduling under BSP is finding a tradeoff between load-balancing and number of CS
Coming back to the example
Scheduling in BSP
Parameters of BSP

• Latency (minimum time between communications)
• computing an h-relation (hg+s)
• Interest: model based on a cost function
Complexity under BSP

• Simple problems under the delay model become hard under BSP

• However, it seems possible to design good competitive algorithms (for instance for scheduling independent chains).
Alternative models: LogP

Need of computational models closer to the actual parallel systems [Culler et al.]: 4 parameters.

- \( L \) latency
- \( o \) overhead
- \( g \) gap
- \( P \) number of processors
Alternative models: LogP

No overlap.
Alternative models: LogP

No overlap.
Alternative models: LogP

No overlap.

The delay model is a LogP-system where $o=g=0$
Scheduling the previous example in LogP
Complexity of LogP

Of course, LogP seems (is?) harder.

It is true for

\((\text{LogP})_{\text{P=2}} \mid \text{Fork,pj} \mid \text{Cmax}\) and 

\((\text{LogP})_{\text{P=inf}} \mid \text{Fork,pj} \mid \text{Cmax}\)
Scheduling a fork graph under LogP

This problem is NP-hard. LogP is harder. Too hard?
Alternative model

Independent applications are submitted locally on a cluster. They are represented by a precedence task graph.

An application is a parallel rigid job.

Let us remind briefly the model. See Feitelson for more details and classification.
Local queue of submitted jobs

... J3 J2 J1

Cluster
Rigid jobs: the number of processors is fixed.
Runtime $p_i$

# of required processors $q_i$
Scheduling rigid jobs: Packing algorithms

Scheduling independent rigid jobs may be solved as a 2D packing Problem (strip packing). List algorithm (off-line).
Alternative models: Malleable Tasks
Malleable Tasks

Communications are implicit

Natural link with applications:

• Partitioning the graph into routines.

• Parallel routines that can be analyzed easily (prediction of performances, using for instance PRAM algorithms or library routines).
Malleable Tasks

Informal definition:

A malleable task (MT) is a computational unit that can itself be executed in parallel on an arbitrary number of processors.
Exemple

MT Graph

MT Scheduling
Advantage of MT

The granularity is large, thus, it allows to neglect communications between MT or at least to consider the SCT assumption…

The performance analysis of each MT can give a rather good execution time estimation
Taking into account the communications

We introduce a penalty factor for representing the global overhead (communications plus synchronizations plus sequential character).
Le temps d’exécution parallèle décroit avec le nombre de processeurs et la pénalité augmente.
Definition of Inefficiency factor of task $T$ on $i$ processors whose execution time is $\text{exec}(T,i)$:

$$
\mu(T,i) = \frac{\text{exec}(T,i)i}{\text{exec}(T,1)}
$$

Expected Properties:

$$
\mu(.,i) \uparrow
$$

$$
\frac{\mu(.,i)}{i} \downarrow
$$
Formal definition

Scheduling of the MT-graph \( G = (V,E) \) on \( m \) processors:

Find two functions \((\text{date}, \text{allot})\) subject to:

- resource constraint
  \[
  \forall \tau : \sum_{i \in \text{slot } \tau} \text{allot}(i) \leq m
  \]

- respect of precedences
  \[
  \forall (i,j) \in E:
  \text{date}(j) \geq \text{date}(i) + t(i, \text{allot}(i)) + C_{i,j}
  \]

- objective: minimizing the makespan \( C_{\text{max}} \)
On-line scheduling
Constructing a batch scheduling

**Analysis**: there exists a nice result which gives a guaranty for an execution in batch function of the guaranty of the scheduling policy inside the batches.
Analysis [Shmoys]

previous last batch  last batch

\( r_n \)

(last job)

Cmax
Proposition

\[ C_{\text{max}} \leq 2\rho C^*_{\text{max}} \]
Analysis

Tk is the duration of the last batch

$$\rho C^*_{\text{max}} \geq r_n + T_k$$

On another hand, $D_{k-1} \leq r_n$ and $\forall i, T_i \leq \rho C^*_{\text{max}}$

$$C_{\text{max}} = D_{k-1} + T_{k-1} + T_k$$

Thus: $C_{\text{max}} \leq 2\rho C^*_{\text{max}}$
Conclusion

We have presented and discussed the problem of scheduling in the context of Parallel Processing.

There is an important impact of the computational model on the performances.

Communications are crucial and have to be optimized. Partitioning sounds more important than internal scheduling.