Coopération en Informatique Parallèle

Denis TRYSTRAM
Grenoble INP - IUF (labo LIG)

octobre, 2013
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Parallel everywhere

Informally, **parallelism** is the simultaneous use of multiple computing devices (whatever they are).

**Today.** (Example of molecular dynamics):

- The codes are multi-scale (classical dynamic and quantum chemistry).
- Several hundreds of thousands atoms.
- Sophisticated visualization modules.
- In Situ simulations (adaptive).
HPC (High Performance Computing)

**TOP500**: an international organization created in 1993. Website updated twice a year.

Main goal: to rank the most powerful computing platforms. Historical data, stats, evolution tendencies (energy, accelerators), ...
Petit intermède culturel...

Performance scale: $10^3$

- Mega
- Giga (Giant)
- Tera (monster)
- Peta
- Exa
- ...
Evolution of computing: The TOP’500

http://www.top500.org/statistics/perfdevel/
maximum performance (in PetaFlops) and efficiency (average on peak)

- Tianhe-2 Milky Way-2 (China) – 33.86
- Titan (Cray - USA) – 17.59
- Sequoia (IBM - USA) – 17.17
- K-computer (Fujitsu, Japan) – 10.51
Evolution during these last 20 years
Predictions for the next years
A deeper look: evolution by countries
A deeper look at the countries in 2013
Repartition of the running applications
Number of cores
Type of architectures
Order of magnitude

Comparison with today computing platforms in term of maximal performances (in GigaFlops).
In the TOP’500 from 1993, the most powerful machine reached 59 GigaFlops.
20 years later, the basic processors are more powerful (the Galaxy – or Iphone – under Android are not too far!).
The computing landscape today...

New and future high performance computing platforms

- Computational grids
- Desktop grids (volunteer computing)
- Hybrid multi- and many-cores (accelerators)
- Clouds
Common characteristics in all kind of most platforms

- Very large number of resources
- Distributed features (hierarchical structure)
- Heterogeneity (with accelerators)
- Uncertainties on data at any level
- Energy limitations

The challenge today is on the **automation and universality** of software (and tuning) tools
Emergence of new parallel platforms

The evolution of high-performance execution platforms leads to physical or logical distributed entities (called organizations) which have their own local rules. For instance, CiGri platform in Grenoble. Each organization is composed of multiple users who compete for the resources, and they aim at optimizing their own objectives. Such systems are often hierarchical (many-core).

Proposal:
To create a general framework for studying the resource allocation problem for most situations and study in this context the concept of cooperation.
Multi-organization scheduling

Informally, a set of users have some applications to execute on distributed resources. These resources belong potentially to multiple organizations that may have their local control and rules. The objectives of the users are not necessarily the same, but they are related to a metric on the completion times (i.e. the finishing times) of the jobs.
Synthetic view of the problem
More formally

The problem is to allocate the jobs to the available resources according of a certain objective. Then, the jobs are scheduled locally.

The set of jobs is available at time 0, the execution is performed by a series of successive time frames (called batches).
Formally

The parallel platforms are viewed as entities which have all their local rules. Each physical entity is composed of users who compete for resources with their own needs or wishes.

Generic problem
to allocate jobs to available computing (or networking) distributed resources aiming at optimizing some objective. Then, execute them locally.

We have to determine when and where to execute the jobs. This corresponds to two interleaved problems, namely $\pi$ (allocation) and $\sigma$ (scheduling).
Some notations

- $m$ "machines"
- $n$ jobs de duration $p_j$ for $1 \leq j \leq n$ (sometimes the jobs are them-selves parallel applications)
- $N$ users, owning each $n_i$ jobs

Practically (in existing systems), several priority queues which manage the jobs according to some policy (FCFS and its variants like back filling).
Problems classification

Key parameters

- Users
- Applications (jobs)
- Resources
- Control
- Objective(s)
Methodology for the analysis

- Modelization and formal definition of the problem
- Complexity analysis (NP-completeness, inapproximation)
- Algorithms design
- Analysis (worst case bounds, simulations, realistic testbeds)
Methodology for the analysis

- Modelization and formal definition of the problem
- Complexity analysis (NP-completeness, inapproximation)
- Algorithms design
- Analysis (worst case bounds, simulations, realistic testbeds)

The idea is to study various problems with the same way (and create adequate common tools).
Brief recall on complexity

We distinguish between easy problem (those for which there exists polynomial algorithms) and hard problems. These problems are characterized by their ability to check if a solution is valid (in polynomial time).
We do not expect exact solutions for big instances, thus, we are looking for approximation algorithms running in reasonable times.
Example
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Starting smoothly with well-known results

A preliminary basic problem

- **Users**: single or multiple, uniform or heterogeneous
- **Jobs**: sequential, parallel (rigid or malleable), divisible loads
- **Resources**: single, identical, hierarchical, heterogeneous
- **Control**: centralized or distributed
- **Objectives**: $C_{\text{max}}$, $\sum C_i$, stretch
Classical $P||C_{max}$ problem

Informally, this corresponds to the situation of a single (homogeneous) cluster. Scheduling $n$ independent jobs on $m$ arbitrary parallel identical processors aiming at minimizing $C_{max}$.

**Complexity:**
The problem is weakly NP-hard [Ullman’75].
Solving by a list algorithm

**Algorithm framework:**
List-scheduling [Graham’69] (greedy) whose principle is to build a list of ready jobs, and to execute any of these jobs as soon there are available processors. This algorithm has a guarantee in the worst case.

**Remarks:**
(asymptotically) optimal algorithm for a large number of jobs.
Analysis 1

The idea is based on a geometrical proof on the Gantt chart:

\[ m \cdot C_{\text{max}} = W + S_{\text{idle}} \quad (\text{where} \ W = \sum p_j) \]
Analysis 2

Proposition.
List scheduling is a 2-approximation.

\[ m \cdot C_{\text{max}} = W + S_{\text{idle}} \]
Proposition.
List scheduling is a 2-approximation.

\( m \cdot C_{\text{max}} = W + S_{\text{idle}} \)

Lower bounds:
\( C^*_{\text{max}} \geq \frac{W}{m} \) and \( C^*_{\text{max}} \geq p_{\text{max}} \)
Proposition.
List scheduling is a 2-approximation.

\[ m \cdot C_{\text{max}} = W + S_{\text{idle}} \]

Lower bounds:
\[ C_{\text{max}}^* \geq \frac{W}{m} \text{ and } C_{\text{max}}^* \geq p_{\text{max}} \]

\[ S_{\text{idle}} \leq (m - 1) \cdot p_{\text{max}} \leq (m - 1) \cdot C_{\text{max}}^* \]

\[ C_{\text{max}} = \frac{W}{m} + \frac{S_{\text{idle}}}{m} \leq (1 + \frac{m-1}{m}) \cdot C_{\text{max}}^* \]
Tightness for general list scheduling

**Proposition.**
The worst case bound of \(2 - \frac{1}{m}\) for list scheduling is tight.
improvement: the LPT rule

Based on the tightness of the 2-approximation ratio, the bound can be improved by considering the specific LPT policy (largest processing times first):

```
<table>
<thead>
<tr>
<th>15</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>
```
improvement: the LPT rule

Based on the tightness of the 2-approximation ratio, the bound can be improved by considering the specific LPT policy (largest processing times first):

Approximation bound: \( \frac{4}{3} \) (for \( m \geq 2 \))
Tightness of LPT

Proposition.
The worst case bound of $\frac{4}{3} - \frac{1}{3m}$ for LPT is tight.
Synthesis

List is a very nice framework which realizes a good trade-off between simplicity and efficiency. It can be extended to many cases, sometimes it is possible to analyze theoretically.

- other objectives ($\sum C_i$ with the "reverse" SPT policy which is optimal for $n$ independent jobs on $m$ machines)
- taking into account communication costs
- Parallel rigid jobs
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Problem

- **Users**: single or multiple, uniform or heterogeneous
- **Type of applications**: sequential, parallel rigid or malleable, divisible loads
- **Resources**: single, identical, hierarchical, heterogeneous
- **Control**: centralized or distributed
- **Objectives**: $C_{\text{max}}$, $\Sigma C_i$, stretch
Rigid jobs correspond to parallel applications
Problem statement

Problem:
Given $n$ independent rigid jobs to be scheduled on one cluster composed of $m$ identical machines minimizing the makespan $C_{max}$.
Complexity results for one strip

The problem is obviously NP-hard.
Complexity results for one strip

The problem is obviously NP-hard.

It is $\frac{3}{2}$-inapproximable unless $P = NP$
Algorithms for one strip

- List Scheduling is still a \((2 - \frac{1}{m})\)-approximation for non-continuous case only! Introduced by Graham in 1975 and revisited recently by Eyraud, Mounié and Trystram in IPDPS 2007.
- Steinberg/Schiermeyer: fast 2-approx available for both versions
- Jansen: very costly \((\frac{3}{2} + \epsilon)\)-approx available for both versions
- Kenyon-Remila: AsymptoticFPTAS available for both versions
Including extra rules

HF (Highest first) is a natural extension (similar to the LPT rule).

**Bad news:** no better approximation as for general list (open question)

**Good news:** nice dominance rule
Analysis of HF policy

HF (Highest first) is a list algorithm that sorts the jobs by non-increasing order of their height.

high utilization zone (I) (more than 50% of processors are busy)
Approximation algorithm - analysis

**Proposition.**

The previous algorithm is a 3-approximation even for irregular strip sizes.

![Diagram showing zones and approximations](image)
Analysis of HF policy

Local schedules

Multi-org LB

Compact

load balance
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Problem

- **Users**: single or multiple, uniform or heterogeneous
- **Type of applications**: sequential, **parallel rigid** or malleable, divisible loads
- **Resources**: single, identical, **hierarchical**, heterogeneous
- **Control**: centralized or distributed
- **Objectives**: $C_{\text{max}}$, $\sum C_i$, stretch
**Problem:** Given \( n \) independent rigid tasks and \( k \) cluster (cluster \( Cl_i \) owns \( m_i \) machines), schedule all the jobs minimizing the makespan \( C_{\text{max}} \).

- The continuous Vs non continuous discussion still holds..
Negative result for Multi-Strip

- multi-SP are 2-inapproximable unless $P = NP$, even for $k = 2$ strips

**YES-Instance**

$C_{max}^* = 1$

**NO-Instance**

$C_{max}^* = 2$
Toward positive results for multiSP

- How getting a $\frac{5}{2}$ ratio for regular multiSP?
Toward positive results for multiSP

- How getting a $\frac{5}{2}$ ratio for regular multiSP?
- Using LS again: consider the last finishing task (of size $(p, q)$) starting at time $s$
Toward positive results for multiSP

- How getting a $\frac{5}{2}$ ratio for regular multiSP?
- Using LS again: consider the last finishing task (of size $(p, q)$) starting at time $s$

\[
W \geq ks(m - q) \quad \text{and thus}
\]
\[
C_{max} = s + p
\]
\[
\leq \frac{W}{k(m - q)} + p
\]
\[
\leq \frac{m}{(m - q)\frac{W}{km} + p}
\]
\[
\leq C^*_{max}(\frac{m}{m - q} + 1)
\]
Toward positive results for multiSP

- How getting a $\frac{5}{2}$ ratio for regular multiSP?
- Using LS again: consider the last finishing task (of size $(p, q)$) starting at time $s$
- Thus we need **small** values of $p$ and $q$

\[
W \geq ks(m - q) \text{ and thus }
\]
\[
C_{\text{max}} = s + p
\]
\[
\leq \frac{W}{k(m - q)} + p
\]
\[
\leq \frac{m}{(m - q)\frac{W}{km} + p}
\]
\[
\leq C^*_\text{max} \left( \frac{m}{m - q} + 1 \right)
\]
Toward positive results for HSP

We want (for example) $p \leq \frac{C_{\text{max}}}{2}$ and $q \leq \frac{m}{2}$, directly leading to a $\frac{5}{2}$ ratio.
Toward positive results for HSP

- We want (for example) $p \leq \frac{C_{\text{max}}}{2}$ and $q \leq \frac{m}{2}$, directly leading to a $\frac{5}{2}$ ratio.

- Thus, we first schedule carefully all jobs having $p > \frac{C_{\text{max}}}{2}$ or $q > \frac{m}{2}$ using simple structures like shelves and layers.
Toward positive results for HSP

- We want (for example) \( p \leq \frac{C_{\max}}{2} \) and \( q \leq \frac{m}{2} \), directly leading to a \( \frac{5}{2} \) ratio.
- Thus, we first schedule carefully all jobs having \( p > \frac{C_{\max}}{2} \) or \( q > \frac{m}{2} \) using simple structures like shelves and layers.
- Then, we add using LS the small remaining jobs.
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Add local constraints

- **Users:** single or multiple, uniform or heterogeneous
- **Type of applications:** sequential, parallel (rigid or malleable), divisible loads
- **Resources:** single, identical, hierarchical, heterogeneous
- **Control:** centralized or distributed
- **Objectives:** $C_{\text{max}}$, $\sum C_i$, stretch
Motivations

In the last results, different organizations shared processors and exchange jobs in order to maximize the profits of the whole community.
Motivations

In the last results, different organizations shared processors and exchange jobs in order to maximize the profits of the whole community.
Locally, an organization can act selfishly and refuse to cooperate if in the final schedule one of its (migrated) jobs could be executed earlier in one of its own processors.
The focus here is to study the impact on the global performance ($C_{max}$) of cooperation between selfish organizations. The local objectives are to minimize local $C_{max}$.

Notation:
MOSP($C_{max}$).
MOSP constraints

(a) Initial instance

(b) Global optimum without constraints
**MOSP constraints**

(c) Global optimum without constraints

(d) Optimum with local constraints
Selfishness restrictions 1
Selfishness restrictions 2

Inapproximation.

MOSP\((C_{\text{max}})\) is strongly NP-complete.
Ratio between approximation algorithms with and without selfishness restrictions: \(\geq 2 - \frac{2}{N}\)

(e) Optimal with selfishness restrictions

(f) Optimal with local constraints
Approximation algorithms

The idea is (again) to use LPT for solving MOSP with selfish restrictions.

**Phase 1:** If solving MOSP($C_{max}$), each organization applies LPT locally for its own jobs.

**Phase 2:** Global LPT: each time an organization becomes idle, the longest job that does not have started yet is migrated and executed by the idle organization.
Analysis

**Proposition:**
MOSP is a 2-approximation.

**Proof.**
- Phase 2 works as a list scheduling, so Graham’s classical approximation ratio $2 - \frac{1}{N}$ holds for all of them.
- It is feasible since the migrated jobs are always executed earlier than the original schedule; this guarantees that the *selfishness restriction is always respected* and that both $C_{max}$ of the original organization is not increased;
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Model of multi-organization scheduling

- **organizations** $O^{(u)}$ have **resources** (clusters) and some **local jobs** $\{J^{(u)}_i\}$
- system goal: global makespan $C_{\max}$
- each organization minimizes the makespan of its local jobs $C_{\max}^{(u)} = \max_i C^{(u)}_i$
- idea: move jobs across clusters to optimize $C_{\max}$
Multi-objective optimization based on constraints on organizations’ objectives

Parallel rigid jobs.

- an organization can not increase its local makespan \( C_{\text{max}}^{(u)} \) by cooperating with others
- schedule jobs locally (with makespan \( C_{\text{max}}^{(u)}(\text{loc}) \))
- optimization: \( \min \max C_{\text{max}}^{(u)} \) subject to \( \forall u : C_{\text{max}}^{(u)} \leq C_{i}^{(u)}(\text{loc}) \)
Complexity

The problem is strongly NP-hard, even if each organization has only 2 jobs.

Lower bound of $\frac{3}{2}$ on the global makespan (easy instance).
Outline of the scheduling algorithm (MOCCA)

3-approximation of the global makespan where the local constraints are not violated

1. schedule jobs locally using highest-first (HF) ordering
2. unschedule jobs that complete after $3LB$ ($LB$ is lower bound on the global makespan), sort them by HF
3. schedule large ($> m/2$) jobs backwards from $3LB$
4. schedule remaining jobs in the gaps of the schedule
Example run: first, we ensure the worst-case performance

(a) local scheduling

(b) MOCCA with gaps
Example run: ... then, we collapse the schedule.
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Local versus Global

**Strict constraints**
MOSP’s local and selfishness constraints are too strict in practice. They strongly limit the freedom of the scheduler to find a good global Cmax.

**A clear trade-off**
There is a correlation between the guarantees that we can provide individually for each organization and the global performance of the platform.
Local versus Global

**Strict constraints**
MOSP’s local and selfishness constraints are too strict in practice. They strongly limit the freedom of the scheduler to find a good global $C_{\text{max}}$.

**A clear trade-off**
There is a correlation between the guarantees that we can provide individually for each organization and the global performance of the platform.

**Question**
How much can we improve the global $C_{\text{max}}$ of the entire platform if we allow some controlled degradation of the local performance?
The \( \alpha \)-Cooperative Multi-Organization Scheduling

**Definition**

We denote as \((\alpha, \beta)\) a schedule that allows the local objectives to be degraded by a factor \(\alpha\) in order to guarantee a \(\beta\)-approximation for the global \(C_{\text{max}}\).
Relaxed multi-organization scheduling

(a) Initial instance

(b) \( \alpha = \frac{3}{2}; \beta = 1 \)

(c) \( \alpha = 1; \beta = \frac{4}{3} \)
Analysis of the **degree of cooperation** as a bi-objective problem.

![Diagram with axes labeled 'local', 'global', and points labeled '3/2', '4/3']
Figure: Inapproximation points given by Family 1 for $N = 4$
Figure: Inapproximation points given by Family 1 for $N = 3, 4, 5, 6$
First 10 inapprox. points of Family 2

- (10/9; 19/10)
- (5/4; 9/5)
- (4/3; 7/4)
- (3/2; 5/3)
Coopération en Informatique Parallèle

Relaxed multi-organization scheduling

Global Cmax

Local Cmax

(2; 3/2)

(3; 4/3)

(4; 5/4) (5; 6/5)

(3/2; 5/3)

(4/3; 7/4)

(5/4; 9/5)

(10/9; 19/10)

First 10 inapprox. points of Family 2

Family 1 inapprox. points
General principle of the algorithms

(a) Classification

(b) Place large jobs

(c) Pairing
We designed a two algorithms with approx \( (2; \frac{3}{2}) \) and \( (3; \frac{4}{3}) \).

**Algorithm 1:**

- \( \mathcal{A} = \{ O^{(k)} | C^{(k)}_{\text{max}} \leq \frac{1}{2} \} \);
- \( \mathcal{B}_1 = \{ O^{(k)} | \frac{1}{2} < C^{(k)}_{\text{max}} \leq \frac{3}{4} \) and \( \nexists J^{(k)}_i \) such that \( p^{(k)}_i > \frac{1}{2} \} \);
- \( \mathcal{B}_2 = \{ O^{(k)} | \frac{1}{2} < C^{(k)}_{\text{max}} \leq \frac{3}{4} \) and \( \exists J^{(k)}_i \) such that \( p^{(k)}_i > \frac{1}{2} \} \);
- \( \mathcal{C} = \{ O^{(k)} | C^{(k)}_{\text{max}} > \frac{3}{4} \} \).

Figure: Classification
We designed a two algorithms with approx \((2; \frac{3}{2})\) and \((3; \frac{4}{3})\).

**Algorithm 2:**

- \(\mathcal{A} = \{ O^{(k)} \mid C^{(k)}_{\text{local}} \leq \frac{1}{3} \};\)
- \(\mathcal{B}_1 = \{ O^{(k)} \mid \frac{1}{3} < C^{(k)}_{\text{local}} \leq \frac{4}{9} \) and \(\not\exists J_i^{(k)}\) such that \(p_i^{(k)} > \frac{1}{3} \};\)
- \(\mathcal{B}_2 = \{ O^{(k)} \mid \frac{1}{3} < C^{(k)}_{\text{local}} \leq \frac{4}{9} \) and \(\exists J_i^{(k)}\) such that \(p_i^{(k)} > \frac{1}{3} \};\)
- \(\mathcal{C} = \{ O^{(k)} \mid C^{(k)}_{\text{local}} > \frac{4}{9} \} .\)
Coopération en Informatique Parallèle

Relaxed multi-organization scheduling

Previously known guaranteed algorithms
Algorithms 1 and 2
Outline

1. Introduction: parallelism and resource management
2. Classical Scheduling
3. Strip Packing
4. Multiple strip Packing
5. Selfishness
6. Multi-organization scheduling
7. Relaxed multi-organization scheduling
8. Conclusion
Concluding remarks

This talk illustrated a way of solving problems coming from the resource management in new cooperative parallel platforms.

Take home message

It was possible to study the problem by the degree of cooperation that could be imposed to the participants of the platform.
Current research directions

Put the user in the loop: in what extent it is possible to consider the individual objectives in such platforms? The key notion here is the **fairness**.