Scheduling on clusters and grids

Grégory Mounié, Yves Robert et Denis Trystram

ID-IMAG

6 mars 2006
1. Some basics on scheduling theory
   - Notations and Definitions
   - List scheduling

2. Taking into account Communications
   - Basic Delay Model
   - More sophisticated models

3. Grid: towards non-standard models
   - Parallel tasks
   - Divisible tasks

4. Concluding Remarks
Basic references

- Chapitre 3 (Gestion de ressources) "Informatique Répartie", Trystram, Slimani et Jemni editeurs, Hermes, 2005.
- Joseph Leung "Handbook of Scheduling", Chapman & Hall, 2004
Traditional scheduling – Framework

Application = \( \text{DAG } G = (T, E, p) \)
- \( T \) = set of tasks
- \( E \) = precedence constraints
- \( p(T) \) = computational cost of task \( T \)
  (execution time)
- \( c(T, T') \) = communication cost (data sent from \( T \) to \( T' \))

Platform
- Set of \( p \) (identical) processors

Schedule
- \( \sigma(T) \) = date to start the execution of task \( T \)
- \( \pi(T) \) = processor assigned to it
Traditional scheduling – Constraints

Data dependencies

If \((T, T') \in E\) then

- if \(\pi(T) = \pi(T')\) then
  \[\sigma(T) + p(T) \leq \sigma(T')\]

- if \(\pi(T) \neq \pi(T')\) then
  \[\sigma(T) + p(T) + c(T, T') \leq \sigma(T')\]

Resource constraints (sequential tasks)

\[\pi(T) = \pi(T') \Rightarrow [\sigma(T), \sigma(T) + p(T)] \cap [\sigma(T'), \sigma(T') + p(T')] = \emptyset\]
Traditional scheduling – Objective functions

**Makespan** or total execution time

\[ C_{\text{max}}(\sigma) = \max_{T \in \mathcal{T}} (\sigma(T) + p(T)) \]

Other classical objectives:

- Sum of completion times (with its weighted variant)
- With arrival times: maximum flow (response time), or sum flow
- More oriented to fair solutions: maximum stretch, or sum stretch
Some basics on scheduling theory
Taking into account Communications
Grid : towards non-standard models
Concluding Remarks

Notations and Definitions
List scheduling

Scheduling problem

Computational units are identified and their relations are analyzed.

Scheduling
Determine when and where computational units will be executed.
Let $G = (V, E)$ be a weighted directed acyclic graph iff (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.

Notice that some colleagues are considering variants (bipartite, Data flow, etc..)
Some basics on scheduling theory
Taking into account Communications Grid: towards non-standard models
Concluding Remarks

Precedence Task Graph

Denis.Trystram@imag.fr, ID - IMAG, Grenoble

Scheduling on clusters and grids
The problem of scheduling graph $G = (V, E)$ weighted by function $p$ on $m$ processors: (without communication) Determine the pair of functions $(\sigma, \pi)$ subject to the respect of precedences:

$$\forall(i, j) \in E : \sigma(j) \geq \sigma(i) + p(i, \pi(i))$$

Usual objective: to minimize the makespan ($C_{\text{max}}$)

**Theorem**

*Minimizing the makespan is NP-Hard [Ullman 75]*
One step further

This problem remains NP-Hard even in relaxed cases: (independent tasks, trees, etc.)

Consequence

We have to find "efficient" heuristics
The evaluation of a heuristic for a criterion $\omega$.

**Definition (Competitive Ratio)**

A real number $\rho$ such that $\forall$ instance $\mathcal{I}$, $\omega(\mathcal{I}) = r(\mathcal{I})\omega^*(\mathcal{I})$ with $\rho = \sup(r(\mathcal{I}))$
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

Theorem

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

Démonstration.

*by reduction from CLIQUE*

Consequence

*It is impossible to find a heuristic better than 4/3 iff $P \neq NP$*
Lower Bounds

We are looking for simple algorithms that have good competitive ratios.
List scheduling is such a nice framework.
List scheduling

- **Initialization:**
  1. Priority queue = list of free tasks (tasks without predecessors) sorted by priority
  2. \( t \) is the current time step: \( t = 0 \).

- **While it remains some tasks to execute:**
  1. Add new free tasks, if any, to the queue. If the execution of a task terminates at time step \( t \), suppress this task from the predecessor list of all its successors. Add those tasks whose predecessor list has become empty.
  2. If there are \( q \) available processors and \( r \) tasks in the queue, remove first \( \min(q, r) \) tasks from the queue and execute them; if \( T \) is one of these tasks, let \( \sigma(T) = t \).
  3. Increment \( t \).
List scheduling

- Priority level (off-line)
  - Use critical path: longest path from a task with no predecessor to an exit node
  - Computed recursively by a bottom-up traversal of the graph
- Implementation details
  - Cannot iterate from \( t = 0 \) to \( t = C_{\text{max}}(\sigma) \) (exponential in problem size)
  - Use a heap for free tasks valued by priority level
  - Use a heap for processors valued by termination time
  - Complexity \( O(|V| \log |V| + |E|) \)
Analysis of List scheduling

Theorem

**Performance guarantee of list scheduling**

\[ C_{\text{max}}(\sigma) \leq C^*_{\text{max}} \left( 2 - \frac{1}{m} \right) \]
Some basics on scheduling theory
Taking into account Communications
Grid: towards non-standard models
Concluding Remarks

Analysis

Processors

Time

Denis.Trystram@imag.fr, ID - IMAG, Grenoble

Scheduling on clusters and grids
Some basics on scheduling theory
Taking into account Communications
Grid: towards non-standard models
Concluding Remarks

Analysis

Processors

Time

Interval with idle time

Denis.Trystram@imag.fr, ID - IMAG, Grenoble
Scheduling on clusters and grids
Some basics on scheduling theory
Taking into account Communications
Grid: towards non-standard models
Concluding Remarks

List scheduling

Analysis

Processors

Intervals with idle time

Time
Analysis

Theorem (List scheduling analysis)

\[ C_{\text{max}}(\sigma) = \frac{W + \text{Idle}}{m} \]

where

- \( \frac{W}{m} \leq C_{\text{max}}^* \)
- at most \((m - 1)\) idle processors
  - \( \text{Idle} \leq (m - 1) T_\infty \)
  - \( T_\infty \leq C_{\text{max}}^* \)

Corollary

\[ C_{\text{max}}(\sigma) \leq (2 - \frac{1}{m}) C_{\text{max}}^* \]
Brent’s Lemma

Lemma (Brent)

Let $\rho$ be the competitive ratio of an algorithm with an unbounded number of processors. There exists an algorithm with performance ratio $2\rho$ for an arbitrary number of processors.

Since there exists an optimal algorithm for scheduling a graph with unbounded number of processors, there is a 2-approximation algorithm for $m$ fixed (this is another way for looking at the Graham’s bound).
Brent’s Lemma

Diagram showing a bar chart with vertical bars, possibly representing a scheduling algorithm or resource allocation scenario.
Brent’s Lemma
Brent’s Lemma

Some basics on scheduling theory
Taking into account Communications
Grid: towards non-standard models
Concluding Remarks

Notations and Definitions
List scheduling

Scheduling on clusters and grids
Brent’s Lemma

\[ m \]

\[ \text{Denis.Trystram@imag.fr, ID - IMAG, Grenoble} \]

Scheduling on clusters and grids
Brent’s Lemma

\[ m \]

Diagram showing \( m \) with bars representing tasks or intervals, indicating scheduling or allocation patterns in a scheduling context.
The proof is quite similar to Graham’s analysis

\[
C_{\text{max}} = \sum \left\lceil \frac{\text{Work}(t)}{m} \right\rceil
\]

thus (in a Graham’s way)

\[
C_{\text{max}} \leq C_{\text{max}}^\infty + \sum \left\lceil \frac{\text{Work}(t)}{m} \right\rceil
\]
Let us consider a task with two successors.

Complexity: This model is more complicated than the central scheduling problem (Lenstra et al. 1990). Scheduling a graph with communication on a unbounded number of processors is NP-hard.
List scheduling – With communications

ETF *Earliest Task First*

- Dynamically recompute priorities of free tasks
- Select free task that finishes execution first (on best processor), given already taken scheduling decisions
- Higher complexity $O(|V|^3 p)$
- May miss “urgent” tasks on the critical path

There exists a performance guaranty. No efficient algorithm is known for large communication delays.
Other approaches

Two-steps : clustering + load balancing :

- DSC Dominant Sequence Clustering $O((|V| + |E|) \log |V|)$
- LLB List-based Load Balancing $O(C \log C + |V|)$ ($C$ number of clusters generated by DSC)
HEFT : Heterogeneous Earliest Finishing Time

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

2. Allocation
   - For current task \( T_i \), determine best processor \( P_q \) :
     - minimize \( \sigma(T_i) + \omega_{iq} \)
   - Enforce constraints related to communication costs
   - Insertion scheduling: look for \( t = \sigma(T_i) \) s.t. \( P_q \) is available during interval \([t, t + \omega_{iq}]\)

3. Complexity: same as MCP without/with insertion
The goal is to take into account local communication overhead. Such computations are costly on network with high throughput.
Fork tree scheduling in the delay model

**Fork tree**

- **Instance**: A fork tree task graph. The communication cost are modelized with the delay model. Unbounded number of processors.

- **Problem**: minimizing $C_{max}$

**Theorem (Fork Tree)**

*The scheduling of a fork tree is solvable in polynomial time.*
Fork tree scheduling in the LogP model

**Instance**: A fork tree task graph. The communication cost are modelized with the LogP model. Unbounded number of processors.

**Problem**: minimizing $C_{max}$

**Theorem**

*The scheduling of a fork tree is NP-Hard.*
Fact (LogP)

A modelisation closer to reality induces often an increased complexity and a worse approximability.

Consequences

Alternative approaches are required to be able to schedule accurately and efficiently parallel applications.
Some basics on scheduling theory
Taking into account Communications
Grid : towards non-standard models
Concluding Remarks

Parallel Tasks Model

Set of independent jobs (Parallel Tasks).

Set of independent jobs (Parallel Tasks).

- Overhead
- Computation

Parallel Tasks Model

Set of independent jobs (Parallel Tasks).

- Overhead
- Computation
Parallel Tasks Model

Set of independent jobs (Parallel Tasks).

- Overhead
- Computation

Processors

Execution Time
Some basics on scheduling theory
Taking into account Communications
Grid: towards non-standard models
Concluding Remarks

Parallel Tasks Model

Set of independent jobs (Parallel Tasks).

- **Processors**
- **Execution Time**
- **Overhead**
- **Computation**
Parallel Tasks Model

Set of independent jobs (Parallel Tasks).

- **Overhead**
- **Computation**

![Graph showing parallel tasks with execution time and processors]
Batch scheduler principle
Analysis of batch scheduling

All tasks arrived in time interval $I_k$ are scheduled in time interval $I_{k+1}$
Analysis of batch scheduling

Theorem (Online batch scheduling)

On-line (batch) scheduling [Shmoys et al. – SIAM’95] : the approximation ratio is multiplied by a factor of 2.

All tasks arrived in time interval $I_k$ can not be schedule before the beginning of $I_k$. Interval $I_k$ and $I_{k+1}$ are both smaller than $\rho C_{\text{max}}^{*}$. 

Denis.Trystram@imag.fr, ID - IMAG, Grenoble

Scheduling on clusters and grids
FIFO and co. principle

- Fifo guarantees no starvation. But, it may induced large idle time ($m$ times worse than the optimal)

Tasks: 

m processors
FIFO and co. principle

- Fifo guarantees no starvation. But, it may induced large idle time ($m$ times worse than the optimal)
FIFO and co. principle

- Fifo guaranties no starvation. But, it may induced large idle time \((m\) times worse than the optimal) 
- Fifo with basic **back filling** is better
FIFO and co. principle

- Fifo guaranties no starvation. But, it may induced large idle time ($m$ times worse than the optimal)
- Fifo with basic **back filling** is better but the worst case is the same ($m$ times worse than the optimal)
FIFO and co. principle

- Fifo guarantees no starvation. But, it may induce large idle time ($m$ times worse than the optimal)
- Fifo with basic **back filling** is better but the worst case is the same ($m$ times worse than the optimal)
- Fifo with **aggressive** back filling is a list scheduling algorithm (less than twice the optimal)
**DEFINITION:** Divisible load applications can be divided into any number of independent pieces. Perfectly parallel job: any sub-task can itself be processed in parallel, and on any number of workers. This model is a good approximation for applications that consist of very (very) large numbers of identical, low-granularity computations.
Continuous solution

Fact

*Continuous solution for master-worker problem* Instead of looking for integer solution, fractional solutions are found easily to solve optimally distribution of tasks on heterogeneous processors to minimize $C_{\text{max}}$.

Heterogeneous processors, variants of topological networks, etc..

Heavy - but straightforward - techniques.

Continuous solution

Solving the problem with complex network or multiple send-receive per node is much more difficult.
steady-state scheduling

- Communication in grid may be fluidised in continuous streams between nodes, in a steady state, neglecting initialisation and clean-up.
- The full detailed order does not need to be computed

Point of view

Changing the point of view (e.g. relaxing makespan) may help to get polynomial results in large scale applications

Denis.Trystram@imag.fr, ID - IMAG, Grenoble
Easy to handle practically by filling the idle slots of a schedule.
Mixed models
Best effort jobs

Easy to handle practically by filling the idle slots of a schedule.
Easy to handle practically by filling the idle slots of a schedule.
Do not feel alone...

Working group MAO of the GdR ASR (next meeting the 23rd march, LIP6, Paris).
Hot topics

- Dealing with uncertainties and disturbances.
Hot topics

- Dealing with uncertainties and disturbances.
- Game Theory and economical approaches for an alternative way of managing complex decisions.