Scheduling Bag-of-Tasks Applications

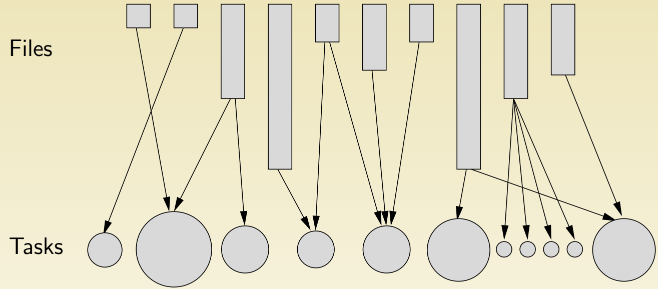
Theoretical Results, Practical Environments, and Perspectives

Arnaud Legrand

CNRS - INRIA - University of Grenoble

ASTEC09

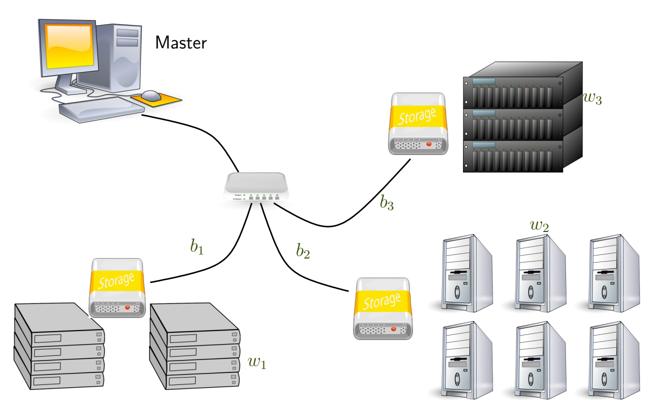
June 4, 2009



Difficulties:

- File sharing;
- Task size heterogeneity;
- Task/cluster affinity;
- Large number of Tasks (small tasks issue + scheduling complexity).

Platform Description

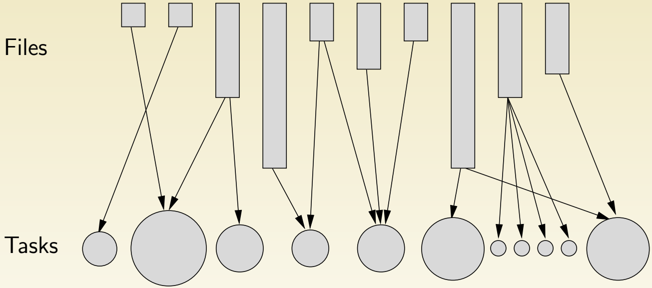


Outline

Outline

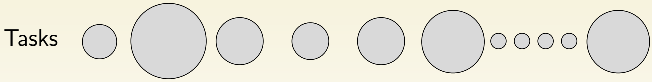
No Input Files

Sometimes, input files are very small or are transfered before hand.



No Input Files

Sometimes, input files are very small or are transfered before hand.



In this case, we end up with some packing problem.

Theoretical Results

The homogeneous case $\langle P||C_{\max}\rangle$

- Any List schedule is a $(2 - 1/m)$ approximation [Cof76].
- The $(2 - 1/m)$ bound is tight for SPT.
- LPT is a $(\frac{4}{3} - \frac{1}{3m})$ approximation [Gra69].

Theoretical Results

The homogeneous case $\langle P||C_{\max}\rangle$

- Any List schedule is a $(2 - 1/m)$ approximation [Cof76].
- The $(2 - 1/m)$ bound is tight for SPT.
- LPT is a $(\frac{4}{3} - \frac{1}{3m})$ approximation [Gra69].

The uniform case $\langle Q||C_{\max}\rangle$ [CK98]

- Any List schedule is a $\frac{1}{\alpha}(2 - 1/m)$ approximation [LD97, Li08], where α is the ratio between the faster and the slower machine.
- Admits a PTAS but no FPTAS (unless you fix m).

The homogeneous case $\langle P||C_{\max}\rangle$

- Any List schedule is a $(2 - 1/m)$ approximation [Cof76].
- The $(2 - 1/m)$ bound is tight for SPT.
- LPT is a $(\frac{4}{3} - \frac{1}{3m})$ approximation [Gra69].

The uniform case $\langle Q||C_{\max}\rangle$ [CK98]

- Any List schedule is a $\frac{1}{\alpha}(2 - 1/m)$ approximation [LD97, Li08], where α is the ratio between the faster and the slower machine.
- Admits a PTAS but no FPTAS (unless you fix m).

The unrelated case $\langle R||C_{\max}\rangle$

- A "complex" (LP+rounding) 2-approximation [LST90].
- Not approximable within $3/2 - \varepsilon$ for any $\varepsilon > 0$ [LST90].
- Admits a FPTAS for the case where m is constant.

And in "Practice"?

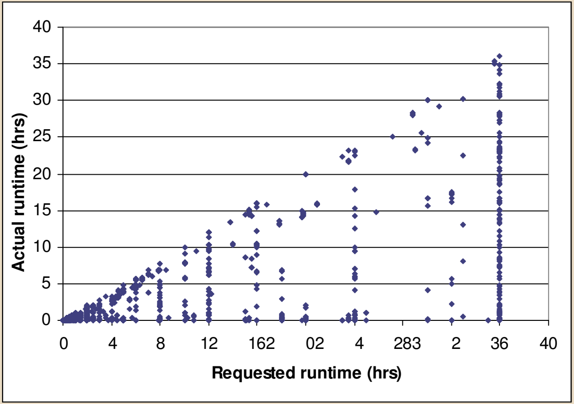
The greedy approach Min-min (SPT), Max-Min (LPT), Sufferage (takes affinity into account) [MAS+99]. $O(mn^2)$.

while there remains a task to schedule **do**
    for each unscheduled task T_i **do**
        for each processor P_j **do**
            Evaluate completion time $CT(T_i, P_j)$
        Evaluate schedule cost $C(T_i) = f(CT(T_i, P_1), \ldots, CT(T_i, P_m))$
    Choose task T_b with the best schedule cost.
    Find out the best processor $P_{b'}$ of T_b.
    Schedule T_b on $P_{b'}$

Practical Issues

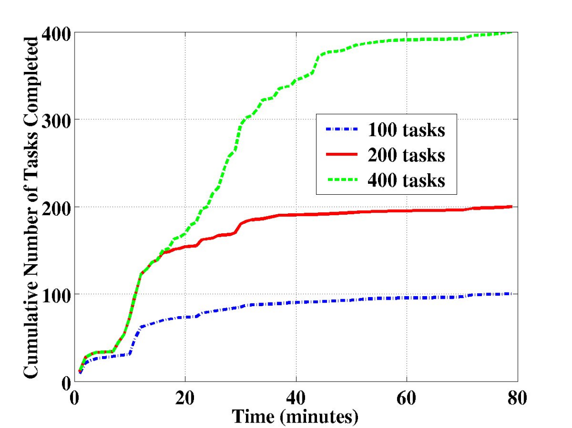
How do you get an estimate of the p_i?

You cannot trust user estimates [MF01]



The last finishing task issue

FCFS scheduling on a desktop Grid [KTB+04]



Practical Issues

How do you handle millions of tasks?

Small tasks induce large overheads.

- They "pollute" the queues.
- Most batch scheduler submission mechanism are ineffective in this setting.
- You have to pay the latency for small file transmission...
- ...or setup mechanisms like advance submission, buffering, pipelining, ...
- $n^2, n \log n$ may be impractical. Need for low-complexity algorithms.

Looking for the "optimal" does not make sense anymore.

$\leadsto$ Small jobs are a pain!

Practical Issues

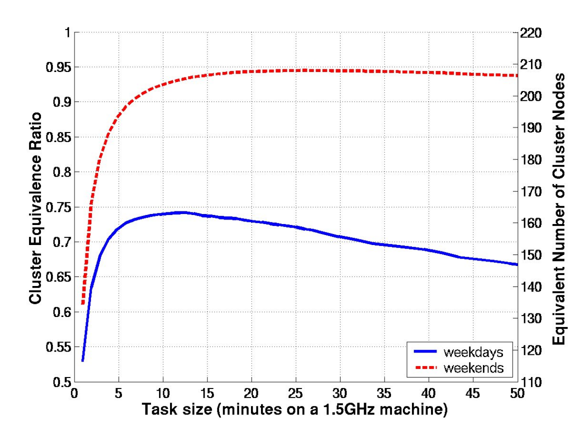
How do you handle failures?

- Jobs are migrated on another machine whenever the resource is reclaimed (Condor).
- Jobs get suspended whenever the resource is reclaimed and resumed later whenever it is available.
- Jobs get suspended whenever the resource is reclaimed. It the resource is not available again shortly, the jobs is considered as lost and resubmitted elsewhere (Entropia).
- Jobs get killed whenever the resource is reclaimed (OAR).
- Jobs are submitted with a deadline, no more communication next; if the deadline is missed the job is lost and resubmitted (BOINC).

$\rightsquigarrow$ Large jobs are a pain too!

Practical Issues

There is an "optimal" job size [KTB+04]



Many Grid Projects Handle This Kind of Applications

- Example:

  Condor Univ. of Wisconsin, started in 1988 [TTL02]
  APST Univ. of California, San Diego, started in 1998 [HC02]
  CiGRI Univ. of Grenoble, started in 2001 [YGR07]
  BOINC Univ. of Berkeley, started in 1999 with SETI@home [And04]
  OurGrid Federal Univ. of Campina Grande, started in 2003 [CBA+06]

Many Grid Projects Handle This Kind of Applications

- Example:

  Condor Univ. of Wisconsin, started in 1988 [TTL02]
  APST Univ. of California, San Diego, started in 1998 [HC02]
  CiGRI Univ. of Grenoble, started in 2001 [YGR07]
  BOINC Univ. of Berkeley, started in 1999 with SETI@home [And04]
  OurGrid Federal Univ. of Campina Grande, started in 2003 [CBA+06]
- Even when "clever" scheduling algorithms are available, most of the time a simple workqueue is used.
- Replication at the end is used to deal with the last finishing task issue.

Many Grid Projects Handle This Kind of Applications

- Example:

  Condor Univ. of Wisconsin, started in 1988 [TTL02]
  APST Univ. of California, San Diego, started in 1998 [HC02]
  CiGRI Univ. of Grenoble, started in 2001 [YGR07]
  BOINC Univ. of Berkeley, started in 1999 with SETI@home [And04]
  OurGrid Federal Univ. of Campina Grande, started in 2003 [CBA+06]

- Even when "clever" scheduling algorithms are available, most of the time a simple workqueue is used.
- Replication at the end is used to deal with the last finishing task issue.

And it works well!!

Another option is to use alternative submission mode.

Many Grid Projects Handle This Kind of Applications

- Example:

  Condor Univ. of Wisconsin, started in 1988 [TTL02]
  APST Univ. of California, San Diego, started in 1998 [HC02]
  CiGRI Univ. of Grenoble, started in 2001 [YGR07]
  BOINC Univ. of Berkeley, started in 1999 with SETI@home [And04]
  OurGrid Federal Univ. of Campina Grande, started in 2003 [CBA+06]

- Even when "clever" scheduling algorithms are available, most of the time a simple workqueue is used.
- Replication at the end is used to deal with the last finishing task issue.

  And it works well!!

  Another option is to use alternative submission mode.

- Very small tasks are a real issue though and aggregating is tedious.
- Very large tasks are a real issue too because of potential failure.

Outline

Natural Extension

- The min-min/max-min/sufferage heuristics naturally extend to this problem [CLZB00]. Xsufferage was a natural extension to take the cluster aspect into account. $\rightsquigarrow O(mn^2 + mnf)$.

Natural Extension

- The min-min/max-min/sufferage heuristics naturally extend to this problem [CLZB00]. Xsufferage was a natural extension to take the cluster aspect into account. $\rightsquigarrow O(mn^2 + mnf)$.
- Another greedy approach with comparable performance [GRV04] but better complexity: $O(mn \log n + mnf)$.

```
for each processor P_j do
    for each unscheduled task T_i do
        Evaluate OBJ(T_i, P_j)
    Build the list L(p_j) of the tasks sorted by OBJ
while there remains a task to schedule do
    for each processor P_j do
        Let T_i be the first element of L(p_j)
        Evaluate completion time CT(T_i, P_j)
    Pick a (T_i, P_j) with minimum completion time.
    Schedule T_i on P_j
```

Hypergraph Partitioning

- A hypergraph $H = (V, N)$ ($n_k \subseteq V$ for each $n_k \in N$).
- Let $\Pi = (V_1, \ldots, V_K)$ be a K-way partition of V.

  For a $n_k \in N$, we can define $\Lambda_k = \{V_i | V_i \cup n_k \neq \emptyset\}$.

Hypergraph Partitioning

- A hypergraph $H = (V, N)$ ($n_k \subseteq V$ for each $n_k \in N$).
- Let $\Pi = (V_1, \ldots, V_K)$ be a K-way partition of V.

  For a $n_k \in N$, we can define $\Lambda_k = \{V_i | V_i \cup n_k \neq \emptyset\}$.
- Each net n_k is weighted with $w(n_k)$ and each pin $v_j \in V$ is weighted with $W(v_j)$.

  The weight W_i of a part V_i is the sum of the weights of its elements.

Hypergraph Partitioning

- A hypergraph $H = (V, N)$ ($n_k \subseteq V$ for each $n_k \in N$).
- Let $\Pi = (V_1, \dots, V_K)$ be a K-way partition of V.

  For a $n_k \in N$, we can define $\Lambda_k = \{V_i | V_i \cup n_k \neq \emptyset\}$.
- Each net n_k is weighted with $w(n_k)$ and each pin $v_j \in V$ is weighted with $W(v_j)$.

  The weight W_i of a part V_i is the sum of the weights of its elements.

Definition.

[The K-way hypergraph partitioning problem] Find a K-way partition minimizing

$$CutSize(\Pi) = \sum_{n_k \in N} w(n_k)(|\Lambda_k| - 1)$$

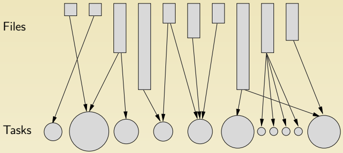
while maintaining the balance

$$\frac{W_{max} - W_{avg}}{W_{avg}} \leq \varepsilon$$

Hypergraph Partitioning (cont'd)

- There is a natural correspondence between our bags of tasks and a hypergraph.

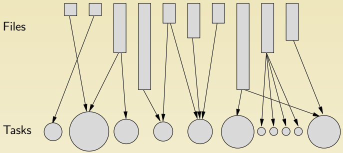
  Tasks ≈ pins and Files ≈ nets.



Hypergraph Partitioning (cont'd)

- There is a natural correspondence between our bags of tasks and a hypergraph.

  Tasks $\approx$ pins and Files $\approx$ nets.

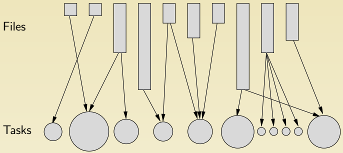


- In a homogeneous setting, finding a K-way partition amounts to minimize file transfers while ensuring a good load balance of computations.

Hypergraph Partitioning (cont'd)

- There is a natural correspondence between our bags of tasks and a hypergraph.

  Tasks ≈ pins and Files ≈ nets.



- In a homogeneous setting, finding a K-way partition amounts to minimize file transfers while ensuring a good load balance of computations.
- Adapting the idea to an heterogeneous is slightly more tricky [Kay06] but worth the effort since there are efficient heuristics for the K-way partition problem.
  - Such heuristics have complexity $O(mn \log n + cnf)$.
  - They have up to 32% of improvements compared to Min-min, Sufferage and such (for high communication to computation ratio).
  - They run more than 10 times faster.

Model Flaws

Of course there are many practical aspects that are not taken into account in the previous model:

- Storage are not infinite.
- The one-port model for distributing the files is simplistic.
- NFS saturation occurs inside the cluster.
- Getting the file size and dependencies is OK but getting the processing time is even more difficult in a heterogeneous setting.

And in Practice ?

Many grid projects handle (more or less) this kind of applications. However. . .

- Even if “clever” scheduling algorithms are available, no grid middleware implements them.

And in Practice ?

Many grid projects handle (more or less) this kind of applications. However. . .

- Even if "clever" scheduling algorithms are available, no grid middleware implements them.
- OurGrid has a simple workqueue with storage affinity (and replication) that works as good as the Xsufferage heuristic [SNCBL04] but does not require any information about the tasks.

And in Practice ?

Many grid projects handle (more or less) this kind of applications. However. . .

- Even if "clever" scheduling algorithms are available, no grid middleware implements them.
- OurGrid has a simple workqueue with storage affinity (and replication) that works as good as the Xsufferage heuristic [SNCBL04] but does not require any information about the tasks.
- Local file managements are LRU anyway but the EGEE project has been investigating a lot about this with simulations.

And in Practice ?

Many grid projects handle (more or less) this kind of applications. However. . .

- Even if "clever" scheduling algorithms are available, no grid middleware implements them.
- OurGrid has a simple workqueue with storage affinity (and replication) that works as good as the Xsufferage heuristic [SNCBL04] but does not require any information about the tasks.
- Local file managements are LRU anyway but the EGEE project has been investigating a lot about this with simulations.
- Most of the time, file staging is a pain for users because the middleware support is lame. . .
- . . . but none of the previous problems is that hard in practice.

Outline

State of the Art

Campaign completion is important, not task completion!

- Few projects have a notion of campaign and of users.

State of the Art

Campaign completion is important, not task completion!

- Few projects have a notion of campaign and of users.
- Most middleware are inspired of “classical” batch schedulers and use strategies like first-come first-served (are we optimizing max-flow ?) with backfilling.

State of the Art

Campaign completion is important, not task completion!

- Few projects have a notion of campaign and of users.
- Most middleware are inspired of "classical" batch schedulers and use strategies like first-come first-served (are we optimizing max-flow ?) with backfilling.
- Some batch schedulers have a notion of fair sharing but it is not efficient at campaign-level.

State of the Art

Campaign completion is important, not task completion!

- Few projects have a notion of campaign and of users.
- Most middleware are inspired of "classical" batch schedulers and use strategies like first-come first-served (are we optimizing max-flow ?) with backfilling.
- Some batch schedulers have a notion of fair sharing but it is not efficient at campaign-level.

Interesting question (for us) How do you handle?

- fairness between users, institutions;
- campaign size heterogeneity;
- campaign heterogeneity (e.g. compilation test campaigns);
- interference with other jobs (support for best-effort).

State of the Art

Campaign completion is important, not task completion!

- Few projects have a notion of campaign and of users.
- Most middleware are inspired of "classical" batch schedulers and use strategies like first-come first-served (are we optimizing max-flow ?) with backfilling.
- Some batch schedulers have a notion of fair sharing but it is not efficient at campaign-level.

Interesting question (for us) How do you handle?

- fairness between users, institutions;
- campaign size heterogeneity;
- campaign heterogeneity (e.g. compilation test campaigns);
- interference with other jobs (support for best-effort).

Important question (for developers)

- Hardware heterogeneity (portability issues);
- Security (user and resource point of view);
- Failures.

Fairness Management in Various Projects

CONDOR Work-queue FCFS.

APST Yuck! If several APST daemons use the same set of resources, the sharing is determined by the resource local sharing policy.

CIGRI Work-queue FCFS.
Could use the OAR "fair-sharing" mechanism: keep track of user's time consumption in a sliding window and use a simple ORDER BY at the task scheduling level.

BOINC Volunteers do not want to work for *any* project. Some sharing emerge from volunteer's priorities.

OURGRID Network of favors, designed to deter free-riders.

- $v_A(A, B) =$ how much A gave to B (according to A).
- A give priority to users with the higher rank $R_A(B) = \max\{0, v_A(B, A) - v_A(A, B)\}$.

Handles lab friendship gracefully.

Outline

Conclusion

- At a coarse-grain level, campaigns are divisible and preemptible.
- Average task size and task campaign could be estimated from observations.

- I am not sure there is much work to do in scheduling for tasks (simple workqueues with replication work fine).
- I think there is a lot of work to do in campaign scheduling. We know of many online efficient algorithms for "fair" divisible tasks.

D. P. Anderson.
Boinc: a system for public-resource computing and storage.
In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10, 2004.

Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade, Lauro Costa, Alisson Andrade, Reynaldo Novaes, and Miranda Mowbray.
Labs of the world, unite!!!
Journal of Grid Computing, 4(3):225–246, 2006.

Pierluigi Crescenzi and Viggo Kann.
A compendium of np optimization problems, 1998.
http://www.nada.kth.se/~viggo/problemlist.

Henri Casanova, Arnaud Legrand, Dmitrii Zagorodnov, and Francine Berman.
Heuristics for scheduling parameter sweep applications in grid environments.
In *Heterogeneous Computing Workshop*, pages 349–363, 2000.

E. G. Coffman.
Computer and job-shop scheduling theory.
John Wiley & Sons, 1976.

R. L. Graham.
Bounds on multiprocessing timing anomalies.
SIAM Journal on Applied Mathematics, 17:416–429, 1969.

A. Giersch, Y. Robert, and F. Vivien.
Scheduling tasks sharing files on heterogeneous master-slave platforms.
In *Proc. 12th IEEE Euromicro Workshop on Parallel Distributed and Network-Based Processing (PDP'04)*, 2004.

Fran Berman Henri Casanova.
Grid Computing: Making the Global Infrastructure a Reality, chapter 33.
Wiley Publisher, Inc., 2002.

D. S. Hochbaum and D. B. Shmoys.

Using dual approximation algorithms for scheduling problems: theoretical and practical results.
J. ACM, 34:144–162, 1987.

Kamer Kaya.
Iterative-improvement-based heuristics for adaptive scheduling of tasks sharing files on heterogeneous master-slave environments.
IEEE Trans. Parallel Distrib. Syst., 17(8):883–896, 2006.
Member-Aykanat, Cevdet.

Derrrick Kondo, M. Taufer, C. Brooks, Henri Casanova, and Andrew A. Chien.
Characterizing and evaluating desktop grids: An empirical study.
In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'04)*, 4 2004.

K. Li and J. E. Dorband.
A task scheduling algorithm for heterogeneous processing.

In *Proceedings of the 5th High Performance Computing Symposium*, pages 183–188, 1997.

Keqin Li.
Performance analysis of list scheduling in heterogeneous computing systems.
International Journal of Computer Systems Science and Engineering, 4(2):103–110, 2008.

J. K. Lenstra, D. B. Shmoys, and É. Tardos.
Approximation algorithms for scheduling unrelated parallel machines.
Math. Programming, 46:259–271, 1990.

M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, and R. Freund.
Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems.
J. Parallel and Distributed Computing, 59(2):107–131, 1999.

. Ahuva W. Mu'alem and . Dror G. Feitelson.

Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling.
IEEE Trans. Parallel Distrib. Syst., 12(6):529–543, 2001.

Elizeu Santos-Neto, Walfredo Cirne, Francisco Brasileiro, and Aliandro Lima.
Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids.
In *JSSPP*, pages 210–232, 2004.

Douglas Thain, Todd Tannenbaum, and Miron Livny.
Condor and the grid.
In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.

Nicolas Capit Yiannis Georgiou and Olivier Richard.
Evaluations of the lightweight grid cigri upon the grid5000 platform.
In *3rd IEEE International Conference on e-Science and Grid Computing (eScience2007)*, Bangalore, India, dec 2007.