An Efficient Algorithm for Scheduling Jobs in Volunteer Computing platforms

A. Essafi, D. Trystram, Z. Zaidi

University of Tunis, University Grenoble-Alpes, University of Tunis,

March 31 - April 4, 2014

Centre CNRS "Paul-Langevin", Aussois, France
Abstract

1. Introduction
2. State of the art
3. Adaptation of HEFT to the availability constraint
4. Stable algorithm for disturbed environments
5. Validation and Empirical study
6. Conclusion and prospect
Desktop Grid

Internet: a reserve of underexploited resources

Volunteering:
- Many actors
- Acceptable performance (technological evolution)
- Latent instability
- Short lived

Collaboration:
- Long-run (Institutional)
- More or less – stable performance
- Not always evident to place
- Conflict of interest
Uncertain Platform

A. Essafi, D. Trystram, Z. Zaidi
Definitions of the Problem

Tasks, machines and objective:

- Tasks: \( n \) non preemptive independent tasks of size \( p_j \)
- Machines
  - \( m \) machines of speed of \( S_i \)
  - Machine \( i \) is only available during given intervals
- Objective
  - Makespan (Taken uncertainty into account)
Related work

Scheduling with availability

Fast Algorithms (list):

More complex Algorithms:

Scheduling with uncertainty

Efficient Algorithms:
ESSAFI and MAHJOUB (2007)

Polynomial approximation
TRYSTRAM (2007).

Scheduling in a heterogeneous environment

Algorithms: HEFT, CPOP:
WU, HARIRI (2002)
HEFT Principle

HEFT principle for DAG

Phase 1

- Calculation of the priority of each task ($rank_u$), which is based on the average calculation and communication costs.
- The task list is generated by sorting the tasks in decreasing order of $rank_u$.

Phase 2

- For most scheduling algorithms, the availability date for a processor $p_j$ is the end of execution of its last task assigned.
- Insertion policy
- Possibility of insertion of a task in an interval of inactivity
HEFT Principle

HEFT principle for DAG

Phase 1

- Calculation of the priority of each task \((rank_u)\), which is based on the average calculation and communication costs.
- The task list is generated by sorting the tasks in decreasing order of \(rank_u\).

Phase 2

- For most scheduling algorithms, the availability date for a processor \(p_j\) is the end of execution of its last task assigned.
- Insertion policy
- Possibility of insertion of a task in an interval of inactivity
HEFT adaptation to availability constraint

- $EST(i, j) = avail(j, i)$: First date in which $j$ can execute task $i$
- $EFT(i, j) = EST(i, j) + \frac{P_i}{S_j}$: End date of $i$ task if executed on processor $j$
- $rank_u(i) = \overline{P_i}$: Priority of the $i$ task
HEFT adaptation to availability constraint

- \( EST(i, j) = avail(j, i) \) : First date in which \( j \) can execute task \( i \)
- \( EFT(i, j) = EST(i, j) + \frac{P_i}{S_j} \) : End date of \( i \) task if executed on processor \( j \)
- \( rank_u(i) = \overline{P_i} \) : Priority of the \( i \) task
HEFT-A2 Algorithm

1. Compute the Rank\textsubscript{u} and the average cost of processing for all tasks
2. Sort all tasks in order of decreasing values of Rank\textsubscript{u}.
3. While there are unscheduling tasks in the list do
4. Select the first task, n\textsubscript{i}, from the list for scheduling
5. For each processor p\textsubscript{k} in the processor-set (p\textsubscript{k} \in Q) do
6. Compute the availability date for n\textsubscript{i} (avail[p\textsubscript{k}, n\textsubscript{i}])
7. Compute EFT (n\textsubscript{i}, p\textsubscript{k}) value
8. Assign task n\textsubscript{i} to the processor p\textsubscript{k} that minimize EFT of task n\textsubscript{i}.
9. endwhile
Exemple for Tasks allocation in HEFT-A2
What’s wrong with HEFT-A2?

**Risks**

- Can fill an interval availability entirely
- Uncertainty unawareness?

**Improvement**

- We must improve the allocation of tasks to machines using the availability model.
Stable algorithm for disturbed environments

Performance modeling, Kondo et al.

- Identifying correlation between resources based on their availability
- Standard classification algorithm (K-means)
- They were able to identify 5 classes of machines

The average availability of a machine in a grid is a good criterion for the classification
\[ \alpha[j] = \frac{\sum_{k=1}^{nb_{\text{intervalles}[j]}} a^k \cdot vitesse[j]}{nb_{\text{intervalles}[j]}} \]

Is the average amount of computation in one interval

- \( \alpha[0] = 0 \)
- \( \alpha[1] = 10 \)
- \( \alpha[2] = 20 \)
- \( \alpha[3] = 30 \)
- \( \alpha[4] = 90 \)
Drawback with Alpha

The alpha parameter does not take into account the dispersion pattern of intervals availability.

The alpha parameter is insufficient to characterize machines.
\[ \beta[j] = \sum_{k=0}^{n_{\text{intervalles}[j]}} (a_j^k - \bar{I}[j])^2 \]

Is the variance between availability intervals of machine j

<table>
<thead>
<tr>
<th>Valeur de alpha</th>
<th>Valeur de beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha[0] = 50</td>
<td>beta[0] = 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valeur de alpha</th>
<th>Valeur de beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha[0] = 25</td>
<td>beta[0] = 1152</td>
</tr>
</tbody>
</table>
HEFT-ASA Algorithm

1. Compute the $\text{Rank}_u$ for all tasks
2. Compute $\alpha[j]$ and $\beta[j]$ for all processors.
3. Sort all tasks in order of decreasing values of $\text{Rank}_u$. (Longest First)
4. While there are unscheduled tasks in the list do
5. Select the first task, $n_i$ from the list
6. Compute $\text{EFT}^+ (n_i) = \min_{j \in 1..m} (\text{EFT} (n_i, p_j))$
7. Let $P_{\text{cand}}$ the list of processor $p_j$ that $\text{EFT} (n_i, p_j) \leq \text{EFT}^+ (n_i) \cdot (1+\tau)$
8. Assign task $n_i$ to processor $p_j$ from $P_{\text{cand}}$ such as $\alpha[j] / \beta[j]$ is the maximum
9. endwhile
Exemple for Tasks allocation in HEFT-ASA

V0=V1=V2=V3=10
Execution Scenario

**Scenario 1** (without disturbance) Tasks are executed using the planned dates.

**Scenario 2** (with disturbance) Potential interruptions of tasks are handled by a local re scheduling mechanism on the same processor.

To do that we designed a specific simulator which supports machine profiles, task profile and disturbance in availability intervals.
Study of performance of heuristics

- Six algorithms are studied:
  - LPT
  - SPT
  - MinMin
  - MaxMin
  - HEFT-A2
  - HEFT-ASA

- Tests are performed on 10 different instances:
  - 10000 tasks
  - 1000 machines
  - $\tau = 0.2$ (empirically chosen)
  - Each instance is disturbed 30 times
Performance Comparison between Heuristics without disturbance

![Makespan of the heuristics](image)
Performance Comparison between Heuristics with disturbance

![Box plot comparing makespan for different heuristics]

- HEFT-AC
- HEFT-ACU
- LPT

A. Essafi, D. Trystram, Z. Zaidi
Conclusion and prospect

Conclusion

- We are considering uncertain Desktop grid platforms
- We adapted HEFT to schedule tasks within the schedule instead of only at the end
- We use the average and variance of the length of availability intervals to characterise the most stable machines
- HEFT-ASA is the most stable evaluated algorithm

Future works

Migration, Duplication...
Thank you for your attention