Efforts to Bridge Theory and Practice on Distributed Scheduling Algorithms
Initial research results

Laércio Lima Pilla, Johanne Cohen
pilla@lri.fr
Agenda

Motivation
Approach
Evaluation
Conclusion
Motivation

Increase in scale of HPC application and platforms → more potential for issues with load imbalance (*dynamic behavior, performance variations, failures ...*)

© Cyril FRESILLON / IDRIS / CNRS Photothèque
Motivation

Increase in scale of HPC application and platforms → more potential for issues with load imbalance (*dynamic behavior, performance variations, failures ...*)

Solution for load imbalance at run-time: *dynamic load balancing*

Problem with usual dynamic load balancing: *overhead of centralizing information, sequential decisions*
Motivation

How to avoid centralizing information? Distributed load balancing.

Examples:

- RTSs with work stealing (e.g., PaRSEC)
- Lifflander et al. Work Stealing and Persistence-based Load Balancers for Iterative Overdecomposed Applications (HPDC 2012)
- Menon, Kale. A Distributed Dynamic Load Balancer for Iterative Applications (SC 2013)
- Freitas et al. A Batch Task Migration Approach for Decentralized Global Rescheduling (SBAC-PAD 2018)
- Pebay, Lifflander. Distributed Load Balancing Utilizing the Communication Graph (Charm++ Workshop 2019)
- Freitas et al. PackStealLB: A Scalable Distributed Load Balancer based on Work Stealing and Workload Discretization (Pre-print)
Motivation

How to avoid centralizing information? Distributed load balancing.

Examples:

- RTSs with work stealing (e.g., PaRSEC)
- Menon, Kale. *A Distributed Dynamic Load Balancer for Iterative Applications* (SC 2013)
- Pebay, Lifflander. *Distributed Load Balancing Utilizing the Communication Graph* (Charm++ Workshop 2019)
- Freitas et al. *PackStealLB: A Scalable Distributed Load Balancer based on Work Stealing and Workload Discretization* (Pre-print)

… not many distributed algorithms used in practice
Motivation

What about searching in the field of distributed algorithms?

• Balls into bins problems
• Game theory (e.g., no-regret learning)
• Average consensus
• ...
Motivation

What about searching in the field of distributed algorithms?

• Balls into bins problems
• Game theory (e.g., no-regret learning)
• Average consensus
• ...

Abstract models, little to no information

Search for bounds, convergence, Nash equilibria ...

Objective: bridge the gap between theory and practice on distributed scheduling algorithms

Approach: search for ways to adapt and improve distributed scheduling algorithms from the literature to be used in HPC scenarios
Approach

Take an algorithm from the literature and expand on it.


Structure of a round:

For each task $b$ do in parallel:

- Let $i_b$ be the current resource of task $b$
- Choose resource $j_b$ uniformly at random
- Let $X_{ib}(t)$ be the current load of resource $i$
- Let $X_{jb}(t)$ be the current load of resource $j$
- If $X_{ib}(t) > X_{jb}(t)$ then:
  - Move task $b$ from $i_b$ to $j_b$ with probability $1 - \frac{X_{jb}(t)}{X_{ib}(t)}$

- $m$ atomic, unitary tasks
- $n$ identical resources
- $\epsilon$-Nash equilibrium in $O(\log \log m)$
- Perfect balance expected in $O(\log \log m + n^4)$
Information to add: average resource load ($\bar{x} = m/n$)

Organize the resources in three categories:

- **Underloaded**: $X_{i,j}(t) < \bar{x}$
- **Average-loaded**: $\bar{x} \leq X_{i,j}(t) \leq \bar{x} + \varepsilon$
- **Overloaded**: $\bar{x} + \varepsilon < X_{i,j}(t)$

Use this information to define which resources can send tasks ($\textit{sources}$) and receive tasks ($\textit{destinations}$)
## Approach

### Variations over Selfish + average load

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sup</strong> (only overloaded)</td>
<td>Sup_Inf (most restrictive)</td>
<td>Sup_Avg</td>
<td>Sup_Sup</td>
</tr>
<tr>
<td><strong>Avg</strong> (average-load or above)</td>
<td>Avg_Inf</td>
<td>Avg_Avg</td>
<td>Avg_Sup</td>
</tr>
<tr>
<td><strong>Inf</strong> (all resources)</td>
<td>Inf_Inf</td>
<td>Inf_Avg</td>
<td>Inf_Sup (original)</td>
</tr>
</tbody>
</table>
**Approach**

Variations over Selfish + average load

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup (only overloaded)</td>
<td>Sup_Inf (most restrictive)</td>
<td>Sup_Avg</td>
<td>Sup_Sup</td>
</tr>
<tr>
<td>Avg (average-load or above)</td>
<td>Avg_Inf</td>
<td>Avg_Avg</td>
<td>Avg_Sup</td>
</tr>
<tr>
<td>Inf (all resources)</td>
<td>Inf_Inf</td>
<td>Inf_Avg</td>
<td>Inf_Sup (original)</td>
</tr>
</tbody>
</table>

### Structure of a round for **SelfishAL_Sup_Inf**:

For each task $b$ in an overloaded resource do in parallel:

- Let $i_b$ be the current resource of task $b$
- Choose resource $j_b$ uniformly at random
- Let $X_{ib}(t)$ be the current load of resource $i$
- Let $X_{jb}(t)$ be the current load of resource $j$
- If $j_b$ is underloaded ($X_{jb}(t) < \bar{x}$) then:
  - Move task $b$ from $i_b$ to $j_b$ with probability $1 - \frac{X_{jb}(t)}{X_{ib}(t)}$
Evaluation

Based on a simple simulator @ https://github.com/llpilla

Why simulation?

• The number of resources needed to obtain results can be large
• Easier debugging
• Easier to reproduce results

Metrics

• Number of rounds for convergence
• Number of task migrations
• Number of load checks
Evaluation

Hypotheses

1. When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

2. Limiting the set of source resources decreases the total number of task migrations.

3. Limiting the set of source resources decreases the total number of load checks.

4. Limiting the set of destination resources decreases the total number of task migrations.

5. Limiting the set of destination resources decreases the total number of load checks.

6. The original task distribution in a scheduling scenario affects the number of rounds an algorithm takes to converge.

7. The higher the number of resources in a scheduling scenario, the higher the number of rounds it takes to balance the load.
Evaluation

Additional details about the simulation

• All tasks make decisions based on the information available at the start of the round
  • No worries about ordering messages
• The simulation is stopped if an algorithm does not converge in 1000 rounds
• Convergence = no resource has a load 5% over the average resource load
## Evaluation

### Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Total number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of resources</strong></td>
<td>10, 50, 100, 500</td>
<td>4</td>
</tr>
<tr>
<td><strong>Average number of tasks per resource</strong></td>
<td>50, 100</td>
<td>2</td>
</tr>
<tr>
<td><strong>Type of tasks</strong></td>
<td>Unitary</td>
<td>1</td>
</tr>
<tr>
<td><strong>Initial task distribution</strong></td>
<td>1 with all tasks in resource 0</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>10 normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 exponential</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 exponential CDF</td>
<td></td>
</tr>
</tbody>
</table>

248 scenarios x 9 schedulers x 10 samples = **22320 samples**
Evaluation

Examples of initial load distributions (100 resources, 10000 tasks)

248 scenarios x 9 schedulers x 10 samples = 22320 samples
**Evaluation**

**Hyp. 1:** When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

Mean number of rounds for convergence

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup (only overloaded)</td>
<td>12.956</td>
<td>4.281</td>
<td>4.732</td>
</tr>
<tr>
<td>Avg (average-load or above)</td>
<td>21.606</td>
<td>4.191</td>
<td>4.541</td>
</tr>
<tr>
<td>Inf (all resources)</td>
<td>11.896</td>
<td>33.761</td>
<td>41.677</td>
</tr>
</tbody>
</table>
Hyp. 1: When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

Mean number of rounds for convergence

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup (only overloaded)</td>
<td>12.956</td>
<td>4.281</td>
<td>4.732</td>
</tr>
<tr>
<td>Avg (average-load or above)</td>
<td>21.606</td>
<td>4.191</td>
<td>4.541</td>
</tr>
<tr>
<td>Inf (all resources)</td>
<td>11.896</td>
<td>33.761</td>
<td>41.677</td>
</tr>
</tbody>
</table>

Number of cases that did not converge in 1000 rounds

Inf_Sup: 9 times
Inf_Avg: 7 times
### Hyp. 1: When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

**Mean number of rounds when converged**

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sup (only overloaded)</strong></td>
<td>12.956</td>
<td>4.281</td>
<td>4.732</td>
</tr>
<tr>
<td><strong>Avg (average-load or above)</strong></td>
<td>21.606</td>
<td>4.191</td>
<td>4.541</td>
</tr>
<tr>
<td><strong>Inf (all resources)</strong></td>
<td>11.896</td>
<td>33.761</td>
<td>41.677</td>
</tr>
</tbody>
</table>

**Note:** Mean number of rounds when converged.
**Evaluation**

**Hyp. 1:** When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

### Mean number of rounds when converged

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup (only overloaded)</td>
<td>12.956</td>
<td>4.281</td>
<td>4.732</td>
</tr>
<tr>
<td>Avg (average-load or above)</td>
<td>21.606</td>
<td>4.191</td>
<td>4.541</td>
</tr>
<tr>
<td>Inf (all resources)</td>
<td>11.896</td>
<td>31.023</td>
<td>38.182</td>
</tr>
</tbody>
</table>

### Median number of rounds for convergence

<table>
<thead>
<tr>
<th>Source \ Dest</th>
<th>Inf (only underloaded)</th>
<th>Avg (average-load and below)</th>
<th>Sup (all resources)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup (only overloaded)</td>
<td>9</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Avg (average-load or above)</td>
<td>13</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Inf (all resources)</td>
<td>7</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Evaluation

**Hyp. 1:** When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

Comparison between variations and the original Selfish

Value of a scenario = \( \text{avg. rounds for variation} - \text{avg. rounds for Selfish} \)
Evaluation

Hyp. 1: When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.
Evaluation

Hyp. 1: When compared to the original Selfish algorithm, the addition of the average resource load information decreases the number of rounds necessary for convergence.

Extra lessons

Using only underloaded destinations -> bad idea

Using overloaded and average-loaded as sources -> it can be a good idea
**Evaluation**

*Hyp. 6:* The **original task distribution** in a scheduling scenario affects the number of rounds an algorithm takes to converge.

<table>
<thead>
<tr>
<th>Inf_Sup (original)</th>
<th>Avg_Avg</th>
<th>Sup_Inf (most restrictive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdfexponential</td>
<td>exponential</td>
<td>normal</td>
</tr>
<tr>
<td>44.83</td>
<td>34.82</td>
<td>3.59</td>
</tr>
<tr>
<td>exponential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45.83</td>
<td>4.54</td>
<td></td>
</tr>
<tr>
<td>normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.59</td>
<td>5.88</td>
<td></td>
</tr>
<tr>
<td>single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.59</td>
<td>5.88</td>
<td></td>
</tr>
</tbody>
</table>

Average number of rounds

Initial task distribution
Hyp. 6: The original task distribution in a scheduling scenario affects the number of rounds an algorithm takes to converge.
Hyp. 7: The higher the number of resources in a scheduling scenario, the higher the number of rounds it takes to balance the load.
Hyp. 7: The higher the number of resources in a scheduling scenario, the higher the number of rounds it takes to balance the load.

Evaluation

**Avg_Avg - 50 tasks per resource**

5000 tasks

**Avg_Avg - 100 tasks per resource**

5000 tasks
Conclusion

First steps to bridge the gap between theory and practice. Hypotheses hold so far.

Next steps

• Run experiments in larger scale
• Run experiments with non-unitary tasks
• Add more information and techniques
• Test in real HPC systems after filtering variations
Conclusion

Final remarks

We need **standard load balancing simulators**. *Do you know of any?*

* Cannot always count on a supercomputer reservation
* Make it easier to compare algorithms and results

We need **standard load balancing benchmarks with well-documented parameters**.

* Task Bench is a nice step in the right direction but it is not focused in load balancing
Efforts to Bridge Theory and Practice on Distributed Scheduling Algorithms
Initial research results

Thank you.

Laércio Lima Pilla, Johanne Cohen
pilla@lri.fr