Scheduling Methods for Divisible Loads in Distributed Environments

Student: Nguyen The Loc
Supervisor: Professor Takuya Katayama

Foundations of software laboratory, School of IS - JAIST
March 2007
The need for Divisible Load

Definition: "divisible load" is the kind of workload that can be divided into arbitrary, independent chunks

Divisible load arises in many areas of science and technology

- Protein sequence analysis (http://hmmer.wustl.edu)
- Simulator of Cellular Microphysiology (http://www.mcell.cnl.salk.edu)
- Parallel and Distributed Video Processing, and Multimedia (http://www.cosy.sbg.ac.at/~uhl/pdivm.html)
- Feature extraction and edge detection in image processing
- ...
Scheduling problem

Given:

• Heterogeneous computation platform: 01 master, $N$ workers. The configuration (computation power, network bandwidth etc.) of workers are different

• An amount of divisible workload $L_{\text{total}}$ resides at the master.

• Star topology.

In which proportion should the workload be scheduled so that the execution time (or makespan) is minimum?
Star shape network

$L_{\text{total}}$

Master

chunk$_{j,i}$

Worker 1

Worker $i$

Worker N-1

Worker N

$S_i$, $B_i$, $nLat_i$, $cLat_i$

front-end: communication and computation concurrently
Research context

Existing scheduling algorithms in dedicated environments:

- **Divisible Load Theory (DLT) and MI algorithm (V. Bharadwaj, D. Ghose, T.G. Robertazzi, 1996):** simple computation model (homogeneous platform, linear cost model).

- **LP (O. Beaumont, 2003):** Heterogeneous, affine cost, but lack of an idle time control mechanism and worker selection policy.

- **UMR (Y. Yang, 2005):** the worker selection policy is not efficient (based on intuition). Bandwidth and latency are overlooked.
MI (V. Bharadwaj 1996)

Simple computation model:

- Homogeneous platform: CPU speed and bandwidth of workers are equal.
- Overlooks the latencies.
- Lack of worker selection policy.

Definition of worker selection problem:

Given a load partition plan. How to select the best workers that can process the $L_{total}$ such that the makespan is minimal?
LP (O. Beaumont 2003)

LP is better than MI:
• Heterogeneous platform
• Latencies were considered (affine cost model)
• Multiple, identical rounds.
• Fix the execution time of rounds are constant for all processors.

Weakness:
• Large worker’s idle time.
• No worker selection.
UMR (Y. Yang 2005)

- Increase round policy
- Induction relation on chunk size, full bandwidth utilization.
- Find out the optimal number of rounds and chunk sizes.

Drawbacks:
- The computation of chunk size relies on CPU power only, while bandwidth and communication latency are ignored.
- The worker selection based on the intuition and the experience.
My algorithm: MRRS

MRRS (Multi-Round scheduling with Resource Selection)

- Static algorithm.
- Multi-round scheduling.
- MRRS is inspired by UMR, therefore full bandwidth utilization and optimal number of rounds.
- CPU power, network bandwidth and latencies play identical roles in the scheduling.
- 2 worker selection strategies: Greedy, Branch & Bound.
Simulation tool

SimGrid version 3.0:

- H. Casanova, A. Legrand and M. Quinson.
- SimGrid is a joint project between University of Hawai, ID Laboratory (Grenoble, France) and University of Nancy (France).
- Available online at http://simgrid.gforge.inria.fr/
- SimGrid is simulation tool designed for simulation of the scheduling algorithms in distributed environments.
- SimGrid has been used to evaluate the UMR and LP algorithms, as well as about more than 50 studies since 2000.
Simulation results

Performance comparisons among algorithms: MRRS, UMR and LP

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Normalized Makespan</th>
<th>Rank</th>
<th>Degradation from the best (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRRS</td>
<td>1</td>
<td>0.12</td>
<td>0.65</td>
</tr>
<tr>
<td>UMR</td>
<td>1.21</td>
<td>0.88</td>
<td>21.4</td>
</tr>
<tr>
<td>LP</td>
<td>1.59</td>
<td>2</td>
<td>59.8</td>
</tr>
</tbody>
</table>
Scheduling problem in Non-dedicated environment

**Non-dedicate**: worker have to execute both Grid tasks and local tasks.

**Local tasks**: randomly appear, priority are higher than Grid tasks, lead to interrupt the execution of Grid tasks

**Gap**: there is not any study concern about scheduling for divisible load in Non-dedicated environments.

Given a computation platform:

- Heterogeneous: 01 master, $N$ workers
- Non-dedicated: workers have to execute both Grid tasks and high priority local tasks
- An amount of divisible workload $L_{total}$ resides at the master

In which proportion should the workload be scheduled so that the execution time is minimum?
Components of DSA scheduling algorithm

- Computation model that expresses the task processing on the workers.
- Mixed-tendency (L. Yang 2003) prediction method for estimating the power of workers
- MRRS algorithm to scheduling the total workload
Non-dedicated computation model

- Input process: arriving tasks consist of Grid tasks and local tasks. Local tasks arrive and interrupt the implementation of a Grid task.
- Service mechanism: first come first served. The local task's execution is preemptive.
- Worker state: available (free) and unavailable (busy)
- The local task processing on the worker is a M/M/1 queueing system
- The arrival of local tasks: Poisson distribution with arrival rate $\lambda$
- The execution process: exponential distribution with service rate $\mu$

\[ P(\lambda t) \rightarrow \text{Queue} \rightarrow \mu \rightarrow \text{Output} \]
DSA simulation: results

![Graph showing makespan (second) vs ratio: Grid task’s size / local task’s size]

- **X-axis**: Ratio: Grid task’s size / local task’s size
- **Y-axis**: Makespan (second)

The graph compares the performance of DSA and UMR under different ratios.
Components of 2PP scheduling algorithm

• The computation model that expresses the task processing on the workers (similar to the model of DSA algorithm).

• 2PP prediction method for estimating the power of workers.

• MRRS algorithm to scheduling the total workload
2PP prediction strategy

![Graph showing time (T) versus V, with stable and transient periods marked.](image)
2PP simulation: results

Arrival rate of the local tasks (tasks per second)

Makespan (100 second)

- UMR
- 2PP
2PP vs. DSA: results

![Graph showing makespan (100 seconds) against arrival rate of the local task (task/second)]

- **Makespan (100 seconds)**
- **Arrival rate of the local task (task/second)**

- **Comparison between DSA and 2PP**

Values:
- Makespan (100 seconds):
  - 0.29
  - 0.63
  - 1.11
  - 3.33
- Arrival rate of the local task (task/second):
  - 0.29
  - 0.63
  - 1.11
  - 3.33
## Contribution

<table>
<thead>
<tr>
<th>Objective</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Represent the Grid tasks processes under the effect of local tasks.</td>
<td>Non-dedicated computation model that using M/M/1 queueing system.</td>
</tr>
<tr>
<td>Estimate the CPU utilization</td>
<td>• applying strategy Mixed-tendency</td>
</tr>
<tr>
<td>Select the best subset of workers.</td>
<td>• developing prediction strategy 2PP</td>
</tr>
<tr>
<td>Find out the best way to partition and delivering the total workload.</td>
<td>• Greedy selection strategy</td>
</tr>
<tr>
<td></td>
<td>• Branch &amp; Bound strategy</td>
</tr>
<tr>
<td></td>
<td>• Static scheduling algorithm MRRS</td>
</tr>
<tr>
<td></td>
<td>• Dynamic scheduling algorithm DSA</td>
</tr>
<tr>
<td></td>
<td>• Dynamic scheduling algorithm 2PP</td>
</tr>
</tbody>
</table>
Thank you for your attention!