Scheduling in Grid Computing using Master-Slave Scheduling Model

Peter N. Nyumu
Professor: Alfredo Goldman

Mac0461

December 15, 2009
Introduction
  Brief Description and Applications
  Scheduling

Single-Master Master-Slave Systems
  No-Wait in Process

Final
Motivation

- Research in my undergraduate work
Master-Slave scheduling model, involves two sets of processors
Master process and Slave processor
Brief Description

- Master-Slave scheduling model, involves two sets of processors
- Master process and Slave processor
- The master processors are responsible of preprocessing and postprocessing of work orders
- The slave processors are responsible for the actual execution of the orders
Master-Slave Model

![Diagram of Master-Slave Model]
Two different schedule

1. No-wait-in schedule
Two different schedule

1. No-wait-in schedule
   - Each slave task must be scheduled immediately after the corresponding preprocessing task finishes
   - Each postprocessing task must be scheduled immediately after the corresponding slave task finishes.
Two different schedules

2. Canonical schedule

- No preemptions
- Slave tasks begin as soon as their corresponding preprocessing tasks complete.
- Postprocessing tasks are done in the same order as the slave tasks complete and as soon as possible.
Two different schedules

2. Canonical schedule

- Satisfies the following properties:
  - No preemptions
  - The preprocessing tasks begin on the master machine at time 0 and complete at time $\sum a_i$
  - Slave tasks begin as soon as their corresponding preprocessing tasks complete.
  - Postprocessing tasks are done in the same order as the slave tasks complete and as soon as possible.
Application of Master-Slave model

- parallel computing
- semiconductor testing
- industrial applications
Unconstrained Minimum Finish Time Problem (UMFT)

- UMFT problem is NP-hard.
- Apply the canonical schedule.
- Can rearrange the master tasks so that all preprocessing tasks complete before any postprocessing task starts.
- For any canonical schedule $S$, $\frac{C^S}{C^*} \leq 2$ and the bound is tight.
A better bound is achieved by applying the following heuristic:

- Let $S_1 = \{ i : a_i \leq c_i \}$ and $S_2 = \{ i : a_i > c_i \}$
- Reorder the jobs in $S_1$ according to nondecreasing order of $b_i$.
- Reorder the jobs in $S_2$ according to nonincreasing order of $b_i$.
- Generate the canonical schedule in which the $a$ tasks of $S_1$ precede those of $S_2$. 
A better bound is achieved by applying the following heuristic:

- Let $S_1 = \{i: a_i \leq c_i\}$ and $S_2 = \{i: a_i > c_i\}$
- Reorder the jobs in $S_1$ according to nondecreasing order of $b_i$.
- Reorder the jobs in $S_2$ according to nonincreasing order of $b_i$.
- Generate the canonical schedule in which the $a$ tasks of $S_1$ precede those of $S_2$.
- $\frac{C_H}{C^*} \leq \frac{3}{2}$ and bound is tight.
Order Preserving Minimum Finish Time (OPMFT)

- We have same order of preprocessing and postprocessing
- Apply canonical schedule
Order Preserving Minimum Finish Time (OPMFT)

- We have same order of preprocessing and postprocessing
- Apply canonical schedule
- It's possible to construct an $O(n \log n)$ algorithm, by defining a canonical order preserving schedule (COPS)
- There is an OPMFT schedule which is a COPS in which the preprocessing order satisfies that, jobs with $c_j > a_j$ come first, jobs with $c_j = a_j$ come next, and the jobs with $c_j < a_j$ come last.
Canonical Reverse Order Schedules (CROS)

▶ construction of reverse order processing
Canonical Reverse Order Schedules (CROS)

- construction of reverse order processing
- *It works as follows:*

1. The master preprocesses the $n$ jobs in the order $\sigma$.
2. The slave $i$ begins the slave processing of job $i$ as soon as the master completes its preprocessing.
3. The master begins the postprocessing of the last job in $\sigma$ as soon as its slave task is complete.
4. The master begins the postprocessing of job $j \neq k$ at the later of the two times ($a$) when it has finished the postprocessing of the successor of $j$ in $\sigma$, and ($b$) when slave $j$ has finished $b_j$.
Canonical Reverse Order Schedules (CROS)

- construction of reverse order processing
- *It works as follows:*
  - the master preprocesses the *n* jobs in the order *σ*
  - slave *i* begins the slave processing of job *i* as soon as the master completes its preprocessing.
  - the master begins the postprocessing of the last job in *σ* as soon as its slave task is complete
  - the master begins the postprocessing of job *j* ≠ *k* at the later of the two times (*a*) when it has finished the postprocessing of the successor of *j* in *σ*, and (*b*) when slave *j* has finished *b* *j*.
The Minimize Finish Time (MFTNW), subject to the no-wait-in-process constraint.
The Minimize Finish Time (MFTNW), subject to the no-wait-in-process constraint.

The Order-Preserving version of MFTNW.
No-Wait in Process

- The Minimize Finish Time (MFTNW), subject to the no-wait-in-process constraint.
- The Order-Preserving version of MFTNW.
- The Reverse-Order version of MFTNW.
Questions