

A Strategyproof Mechanism for Scheduling Divisible Loads in Distributed Systems

Daniel Grosu Thomas E. Carroll

Department of Computer Science
Wayne State University
Detroit, Michigan 48202, USA
{dgrosu, tec}@cs.wayne.edu

The 4th International Symposium on Parallel and Distributed
Computing,
Lille, France, July 4–6 2005.

Road Map

- Introduction
- Divisible Load Scheduling
- Mechanism Design Theory
- Proposed Mechanism
- Experimental Results
- Conclusion

- We combine Divisible Load Scheduling and Mechanism Design Theory.
- Divisible Load Scheduling (DLS)
 - DLS is concerned with *arbitrarily divisible* loads.
 - All elements in the load demand an identical type of processing.
 - The load can be partitioned into any number of load fractions.
 - The processors are assumed *obedient*.

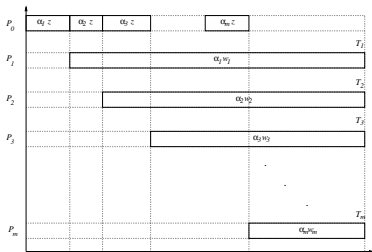
- In the real world, there is no *a priori* motivation for cooperation. Processors will manipulate the scheduling algorithm if it is beneficial to do so.
- Mechanism Design Theory
 - Reconciles private interests with social goals.
 - The processors are modeled as *autonomous agents* who behave *selfishly* (strategic node model).
 - The agents are *rational*. Their behavior is driven by their desire to maximize their utility.
 - Incentives are provided to the agents. The incentives are designed to induce the desired behavior.

- How can we obtain the optimal schedule when employing strategic processors?
- Augment Divisible Load Scheduling with incentives.

System Model

- Distributed system with bus interconnection.
- $S = \{P_0, P_1, \dots, P_m\}$ is the set of processors.
- P_0 is the master (control) processor. P_0 does not have processing capabilities and it can only communicate with a single processor at any give time (*i.e.*, one-port model).

Divisible Load Scheduling (DLS) Problem



- P_i is characterized by w_i , where w_i is the time taken by P_i to process a unit load.
- The fraction of load assigned to P_i is α_i .
- P_i executes its assigned load in time $\alpha_i w_i$.
- z is the time to communicate a unit load from P_0 to any P_i .
- P_0 communicates α_i units of load to P_i in $\alpha_i z$.
- P_i finishes in time $T_i(\alpha) = z \sum_{j=1}^i \alpha_j + \alpha_i w_i$.

BUS-LINEAR Scheduling Problem

Definition

Let $T(\alpha) = \max\{T_1(\alpha), \dots, T_m(\alpha)\}$ be the total execution time, where $\alpha = \{\alpha_1, \dots, \alpha_m\}$. *BUS-LINEAR scheduling problem* is defined as $\min_{\alpha} T(\alpha)$ such that $\alpha_i \geq 0$ ($i = 1, \dots, m$) and $\sum_{i=1}^m \alpha_i = 1$.

Theorem

The optimal solution for the BUS-LINEAR problem is obtained when all processors simultaneously finish executing their assigned load, i.e., $T_1 = \dots = T_m$.

Theorem

Any load allocation order is optimal for the BUS-LINEAR problem.

BUS-LINEAR Scheduling Problem (cont.)

Algorithm

BUS-LINEAR

Input: Time to process a unit load: w_1, \dots, w_m ;

Time to communicate a unit load: z ;

Output: Load fractions: $\alpha_1, \dots, \alpha_m$;

1. **for** $j = 1, \dots, m - 1$ **do**

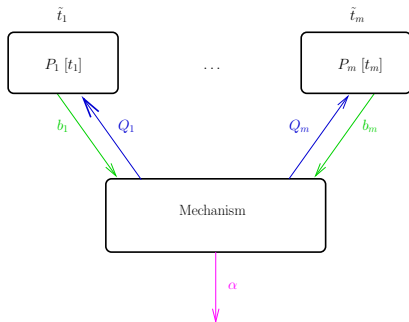
$$k_j \leftarrow \frac{w_j}{z + w_{j+1}}$$

$$2. \alpha_1 \leftarrow \frac{1}{1 + \sum_{i=1}^{m-1} \prod_{j=1}^i k_j}$$

3. **for** $i = 2, \dots, m$ **do**

$$\alpha_i = \alpha_1 \prod_{j=1}^{i-1} k_j$$

Mechanism Design Theory



- $\alpha(\mathbf{b}) = (\alpha_1(\mathbf{b}), \dots, \alpha_m(\mathbf{b}))$, where $\mathbf{b} = (b_1, \dots, b_m)$
- $\tilde{t}_i \geq t_i$
- $Q_i(\mathbf{b}, \tilde{\mathbf{t}})$, where $\tilde{\mathbf{t}} = (\tilde{t}_1, \dots, \tilde{t}_m)$
- $U_i(\mathbf{b}, \tilde{\mathbf{t}}) = Q_i(\mathbf{b}, \tilde{\mathbf{t}}) + V_i(\alpha(\mathbf{b}), \tilde{\mathbf{t}})$

Definition

Strategyproof Mechanism

A mechanism is a *strategyproof mechanism* if every agent i with true value t_i maximizes her utility by bidding t_i independent of the bids of the other agents (*i.e.*, truth-telling is a dominant strategy).

Definition

Voluntary Participation Mechanism

A mechanism is a *voluntary participation mechanism* if

$U_i((\mathbf{b}_{-i}, t_i), (\tilde{\mathbf{t}}_{-i}, t_i)) \geq 0$ for every agent i , t_i , and other agents' bids \mathbf{b}_{-i} (*i.e.*, truthful agents never incur a loss).

Proposed Mechanism

Definition

$$V_i(\alpha(\mathbf{b}), \tilde{\mathbf{w}}) = -\alpha_i \tilde{w}_i$$

Protocol

DLS-BL:

1. After the master processor P_0 collects all the bids it computes the allocation using the *BUS-LINEAR* algorithm.
2. Once the assigned load is finished the Master processor P_0 does the following:
 - 2.1. Determines the execution values for each processor, \tilde{w}_i ($i = 1, \dots, m$).
 - 2.2. For each processor P_i , computes payment $Q_i(\mathbf{b}, \tilde{\mathbf{w}}) = C_i(\mathbf{b}, \tilde{\mathbf{w}}) + B_i(\mathbf{b}, \tilde{\mathbf{w}})$ where $C_i(\mathbf{b}, \tilde{\mathbf{w}}) = -V_i(\alpha(\mathbf{b}), \tilde{\mathbf{w}})$ and $B_i(\mathbf{b}, \tilde{\mathbf{w}}) = T_{-i}(\alpha(\mathbf{b}_{-i}), \mathbf{b}_{-i}) - T(\alpha(\mathbf{b}), (\mathbf{b}_{-i}, \tilde{w}_i))$.
 - 2.3. Sends Q_i to each P_i .
3. Each processor receives its payment and evaluates its profit $U_i(\mathbf{b}, \tilde{\mathbf{w}}) = Q_i(\mathbf{b}, \tilde{\mathbf{w}}) + V_i(\alpha(\mathbf{b}), \tilde{\mathbf{w}})$.

Proposed Mechanism (cont.)

Theorem

DLS-BL is a strategyproof mechanism.

Theorem

DLS-BL is a voluntary participation mechanism.

Experimental Results

- We study by simulation the proposed mechanism.
- We consider two distributed system of sixteen processors and one master processor.
- The systems differ in the speed of processor P_1 . The 'fast' system is characterized by $w_1 = 0.1$; the 'slow' system is characterized by $w_1 = 0.7$. The characteristics of the remaining processors are presented at the bottom.
- $z = 0.01$. The system is computationally bound.

| w_1 | | w_2 | $w_3 - w_5$ | $w_6 - w_{10}$ | $w_{11} - w_{16}$ |
|-------|------|-------|-------------|----------------|-------------------|
| fast | slow | 0.1 | 0.2 | 0.5 | 1.0 |
| 0.1 | 0.7 | | | | |

Experimental Results (cont.)

- We consider eight cases:
 - (1) $w_1 = b_1 = \tilde{w}_1$ (i.e., P_1 bids truthfully and it processes the load as reported);
 - (2) $\tilde{w}_1 > w_1 = b_1$ (i.e., P_1 bids truthfully, but processes the load at a slower rate);
 - (3) $w_1 < b_1 = \tilde{w}_1$ (i.e., P_1 bids a rate slower than its true rate, but processes the load at the reported rate);
 - (4) $w_1 = \tilde{w}_1 < b_1$ (i.e., P_1 bids a rate slower than its true rate, but it processes the load at its true rate);
 - (5) $w_1 < \tilde{w}_1 < b_1$ (i.e., P_1 processes its load slower than its true rate and bids a rate slower than its execution rate);
 - (6) $w_1 < b_1 < \tilde{w}_1$ (i.e., P_1 processes the load slower than it bid and bids a rate slower than its true rate);
 - (7) $b_1 < w_1 = \tilde{w}_1$ (i.e., P_1 bids a rate faster than its true rate, but processes the load at its true rate);
 - (8) $b_1 < w_1 < \tilde{w}_1$ (i.e., P_1 bids a rate faster than its true rate and processes the load slower than its true rate).

Experimental Results (cont.)

Bids and execution values for the fast system

| Case | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|-----|-----|-----|-----|-----|-----|------|------|
| b_1 | 0.1 | 0.1 | 0.3 | 0.3 | 0.3 | 0.3 | 0.05 | 0.05 |
| \tilde{w}_1 | 0.1 | 0.3 | 0.3 | 0.1 | 0.2 | 0.4 | 0.1 | 0.2 |

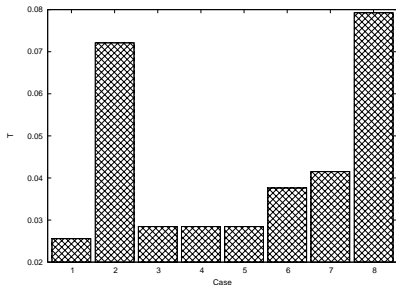
Bids and execution values for the slow system

| Case | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|
| b_1 | 0.7 | 0.7 | 0.9 | 0.9 | 0.9 | 0.8 | 0.5 | 0.5 |
| \tilde{w}_1 | 0.7 | 0.9 | 0.9 | 0.7 | 0.8 | 0.9 | 0.7 | 0.8 |

The Effect of Cheating Is Processor Order Dependent

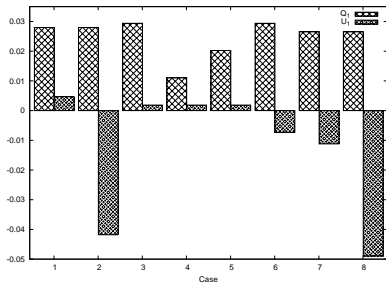
- The effect of cheating depends on processor order.
- The greatest impact occurs when P_1 cheats, the smallest impact when P_m cheats.

Cheating Increases Makespan



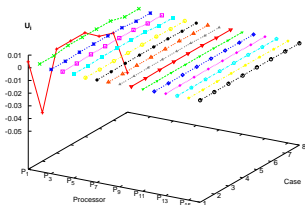
- Minimal makespan obtained when P_1 reports its true value and executes its assignment at that speed (case 1); all other configurations result in increased makespan.
- Long makespans result when $\tilde{w}_1 > b_1$ (cases 2, 6, 7, and 8).

Cheating Reduces Utility



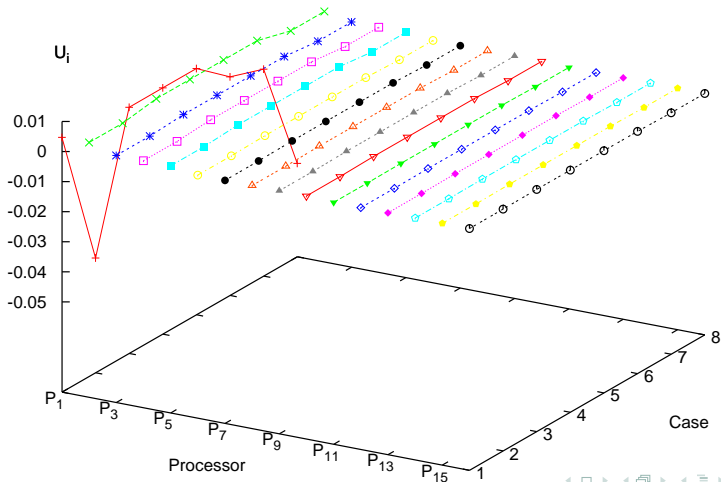
- P_1 obtains greatest utility when it reports its true value to the mechanism and executes its assignment at that speed (case 1); all other configurations result in reduced utility.
- Negative utility results when $\tilde{w}_1 > b_1$ (cases 2, 6, 7, and 8).

The Effects of Cheating On All Processors



- The utility of the other processors may increase, decrease, or remain unchanged.
- The utility never becomes negative.

The Effects of Cheating On All Processors



- We considered the design of strategyproof mechanisms in the context of Divisible Load Scheduling.
- Our solution is to augment DLS with incentives.
- The incentives induce participation and truth-telling.

Future Work

- Expand work to include other architectures (*e.g.*, tree), multiple cheating processors, and multiple jobs with priority.
- Develop distributed mechanisms for task scheduling.
- Study agent privacy in distributed task scheduling mechanisms.