Game Theory Based Load Balanced Job Allocation in Distributed Systems

Anthony T. Chronopoulos

Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, USA
atc@cs.utsa.edu

Load balancing: problem formulation

Load balancing

Given a large number of jobs, find an allocation of jobs to computers optimizing a given objective function (e.g. total execution time or total cost).

Motivation for a game theoretic approach

- Computational resources are distributed and used by many users having different requirements.
- Users are likely to behave in a selfish manner and their behavior cannot be characterized using conventional techniques.

Outline

- Taxonomy of Load Balancing Approaches
- A Noncooperative scheme
- A Cooperative Scheme
- A Dynamic Scheme
- Application to Grid Models
- Future Work

Categories of load balancing policies

- Static policies: base their decision on collected statistical information about the system.
- Dynamic policies: base their decision on the current state of the system.

Job Classification

- Jobs in a distributed system can be divided into different classes (multi-class or multi-user) based on their nature (e.g. arrival rates, execution times etc).
- So, the objective of the load balancing schemes can be to provide
  - a *system-optimal* solution where all the jobs are regarded to belong to one group (one class).
  - an *individual-optimal* solution where each job optimizes its own response time.
  - a *class-optimal (user-optimal)* solution where the jobs are classified into finite number of classes (users) based on their nature and each user tries to optimize the response time of her own jobs.

Load Balancing Approaches

I. *Global approach*
II. *Non-cooperative approach*
III. *Cooperative approach*

I. Global approach

- Only one decision maker that minimizes the response time of the entire system over all jobs.
- Algorithms:

  [Tantawi & Towsley '85][Tang & Chanson '00] nonlinear optimization
  [Kim & Kameda '92] efficient algorithm
  [Li & Kameda '94] tree and star networks

- Several decision makers (e.g. jobs, users) minimize their own response time independently of the others and they all eventually reach an equilibrium.
- At the equilibrium a decision maker cannot receive any further benefit by changing its own decision.
- This situation can be modeled as a non-cooperative game.
- Solutions:
  Wardrop equilibrium - infinite # of decision makers.
  Nash equilibrium - finite # of decision makers.
- Algorithms:
  [Kameda '97] Wardrop equilibrium;
  [Roughgarden '01] Stackelberg game;
  [Grosu '05] Nash equilibrium;

III. Cooperative approach

- Several decision makers (e.g. jobs, computers) cooperate in making the decisions.
- Each of them will operate at its optimum.
- Decision makers have complete freedom of preplay communication to make joint agreements about their operating points.
- This situation can be modeled as a cooperative game.
- Algorithms:

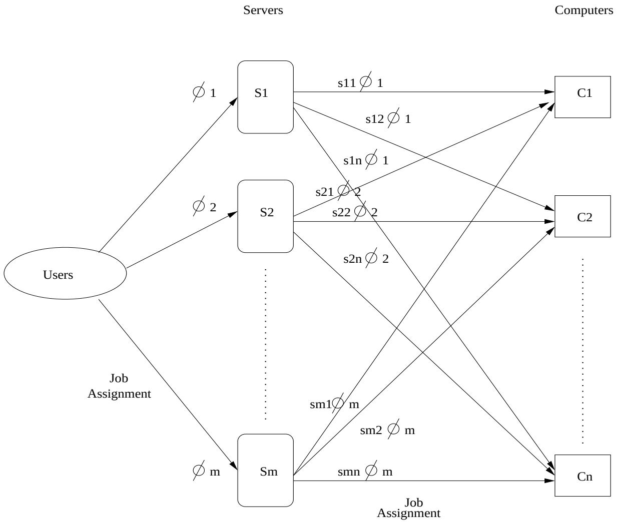
  [Grosu IPDPS'02] ,[Penmatsa IPDPS'06] cooperative game among computers.

Noncooperative Load Balancing Game among Users ([Grosu'05])

- We consider a distributed system that consists of n heterogeneous computers shared by m users.
- User j is a player and she must find a load balancing strategy $\mathbf{s}_j = (s_{j1}, s_{j2}, \ldots, s_{jn})$ that minimizes the expected response time of her jobs.
- The expected response time of user j is given by:

$$D_j(\mathbf{s}) = \sum_{i=1}^{n} s_{ji} F_i(\mathbf{s}) = \sum_{i=1}^{n} \frac{s_{ji}}{\mu_i - \sum_{k=1}^{m} s_{ki} \phi_k}$$

The Distributed System Model



Nash Equilibrium

Nash Equilibrium

A Nash equilibrium of the load balancing game defined above is a strategy profile $\mathbf{s}$ *such that for every user* j*:*

$$\mathbf{s}_j \in \arg\min_{\tilde{\mathbf{s}}_j} D_j(\mathbf{s}_1, \ldots, \tilde{\mathbf{s}}_j, \ldots, \mathbf{s}_m)$$

OP_j: User j Optimization Problem

$$\min_{\mathbf{s}_j} D_j(\mathbf{s})$$

subject to the constraints:

$$s_{ji} \geq 0, \qquad i = 1, \ldots, n$$

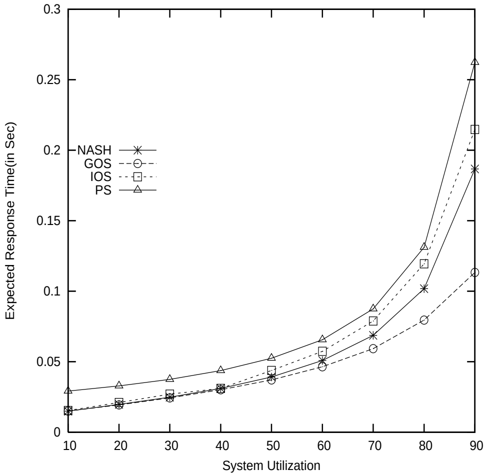
$$\sum_{i=1}^{n} s_{ji} = 1$$

$$\sum_{k=1}^{m} s_{ki}\phi_k < \mu_i, \qquad i = 1, \ldots, n$$

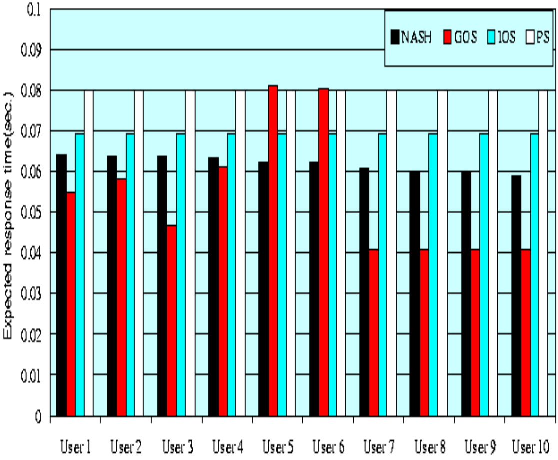
Summary of results

- The new algorithm (NASH) is compared with three other existing load balancing schemes: Proportional Scheme (PS), Global Optimal Scheme (GOS) and Individual Optimal Scheme (IOS).
- GOS minimizes the expected execution time for all the jobs of all users in the entire system.
- NASH finds the Nash equilibrium solution (*i.e.* it minimizes the expected execution time for all the jobs of each user).
- IOS finds the Wardrop equilibrium solution and PS is not optimal.

The expected response time for entire system (non-cooperative game approach)



The expected response time for each user (non-cooperative game approach)



Cooperative Load Balancing

- We consider a distributed computer system that consists of n heterogeneous computers (nodes) interconnected by a communication network.
- The load balancing problem is formulated as a cooperative game among the computers and the communication subsystem.
- The Nash Bargaining Solution (NBS) is the solution for our cooperative load balancing game which provides a Pareto optimal and fair solution.

Cooperative Load Balancing Game

The cooperative load balancing game consists of:

- n computers and the communication subsystem as *players*;
- The *set of strategies*, X, is defined by the following constraints:

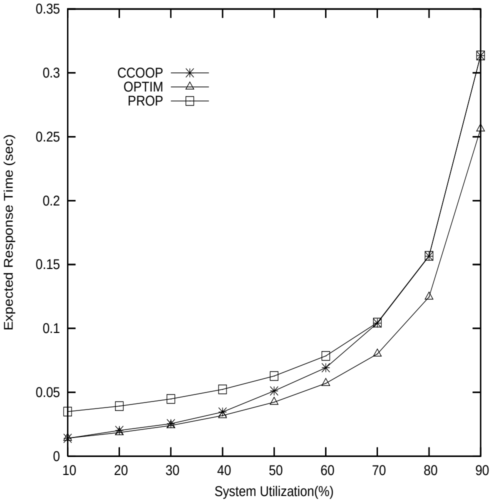
$$\beta_i < \mu_i, \qquad i = 1, \ldots, n \tag{1}$$

$$\sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \phi_i = \Phi, \tag{2}$$

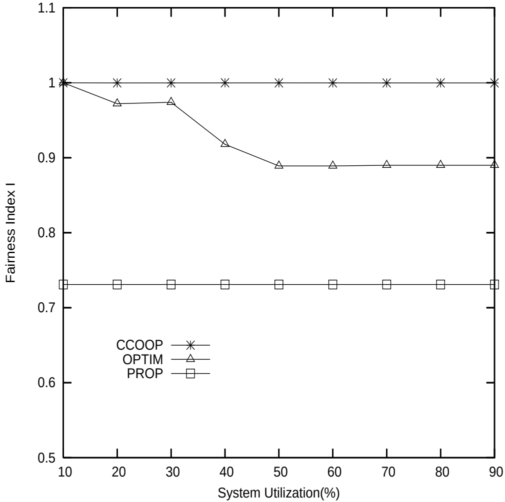
$$\beta_i \geq 0, \qquad i = 1, \ldots, n \tag{3}$$

- For each computer i, $i = 1, \ldots, n$, the *objective function* $f_i(X) = D_i(\beta_i)$; for the communication subsystem, the *objective function* $f_{n+1}(X) = G(\lambda)$; $X = [\beta_1, \ldots, \beta_n, \lambda]^T$. The goal is to minimize simultaneously all $f_i(X)$, $i = 1, \ldots, n+1$.
- For each player i, $i = 1, \ldots, n+1$, the initial performance $u_i^0 = f_i(X^0)$, where X^0 is a zero vector of length $n+1$.

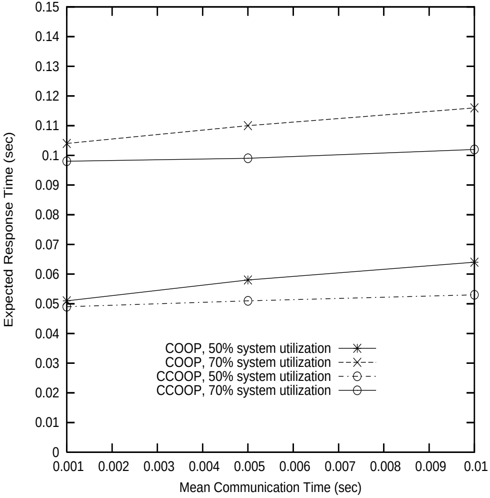
Performance evaluation: Expected response time (cooperative game approach)



Performance evaluation: Fairness index (cooperative game approach)



Communication Time vs Expected Response Time (cooperative game approach)



Dynamic Load Balancing

Distributed dynamic scheme components:

- *Information policy:* The number of jobs waiting in the queue to be processed (queue length) is used as the state information. This state information is exchanged between the nodes every P time units.
- *Transfer policy:* When a job arrives at a node, the transfer policy component determines whether the job should be processed locally or should be transferred to another node for processing.
- *Location policy:* If the job is eligible for transfer, the location policy component determines the destination node for remote processing.

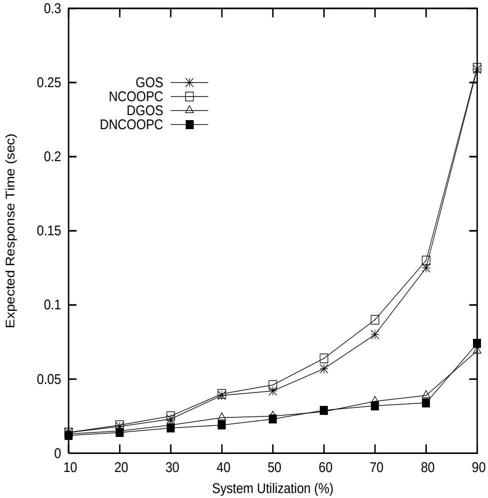
Dynamic Non-cooperative Scheme with Communication (DNCOOPC)

- The goal of DNCOOPC is to balance the workload among the nodes dynamically in order to obtain a user-optimal solution *i.e.* to minimize the expected response time of the individual users.
- The derivation of DNCOOPC is based on the static NCOOPC.

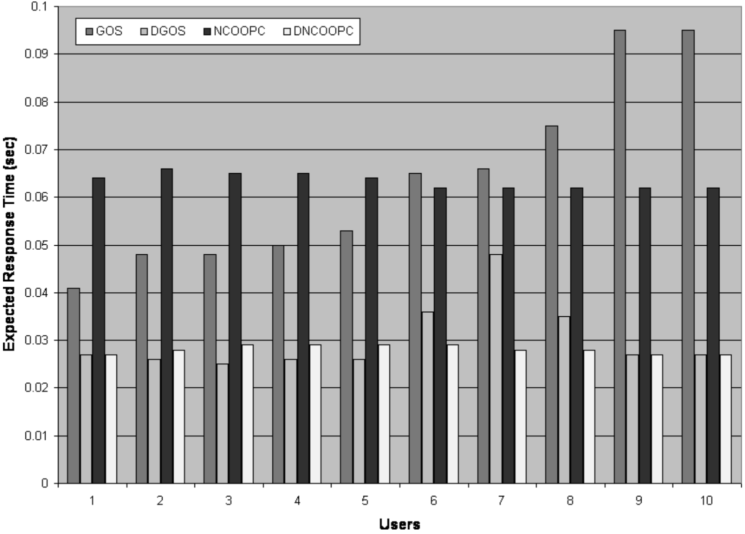
Dynamic Global Optimal Scheme (DGOS)

- The goal of DGOS is to balance the workload among the nodes dynamically in order to obtain a system-wide optimization *i.e.* to minimize the expected response time of all the jobs over the entire system.
- The derivation of DGOS is based on the static GOS similar to DNCOOPC.

The expected response time for entire system (dynamic load balancing approach)



The expected response time for each user (dynamic load balancing approach)



Job Allocation Schemes for Computational Grids (GOSP and NASHP)

- Grid is a type of parallel / distributed system which enables the sharing, selection, and aggregation of geographically distributed ‘autonomous’ resources dynamically at runtime.
- Computational grid: Tries to solve problems or applications by allocating the idle computing resources over a network or the internet
- These computational resources have different owners who can be enabled by an automated negotiation mechanism by the grid controllers

Pricing Model

- Players are the Grid Servers and the Computers
- Reserved valuations
- The server has to play an independent game with each computer associated with it to form the price per unit resource vector, p_j.
- In a system with m servers and n computers at time t, we have $m \times n$ bargaining games.

Non-cooperative Job Allocation Game

A *Non-cooperative job allocation game* consists of a set of players, a set of strategies, and preferences over the set of strategy profiles:

(i) *Players*: The m users.

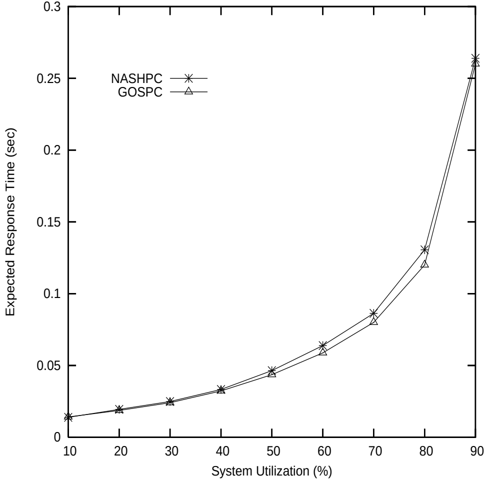
(ii) *Strategies*: Each user's set of feasible job allocation strategies.

(iii) *Preferences*: Each user's preferences are represented by its expected price (D^j). Each user j prefers the strategy profile β^* to the strategy profile $\beta^{*'}$ if and only if $D^j(\beta^*) < D^j(\beta^{*'})$.

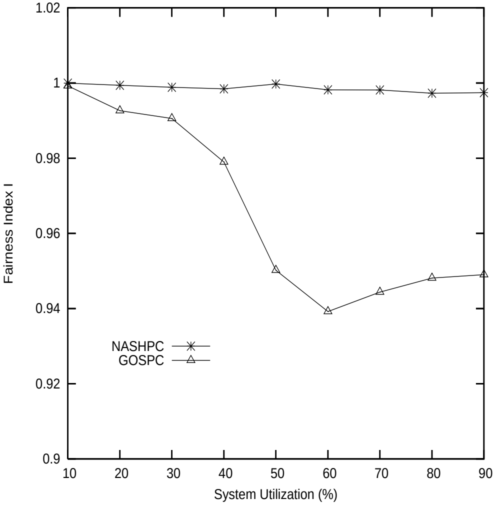
Definition: A *Nash equilibrium* of the job allocation game defined above is a strategy profile β^* such that for every user j $(j = 1, \ldots, m)$:

$$\beta^j \in \arg\min_{\tilde{\beta}^j} D^j(\beta^1, \ldots, \tilde{\beta}^j, \ldots, \beta^m) \tag{4}$$

Performance evaluation: Expected response time (price-based job allocation)



Performance evaluation: Fairness index (price-based job allocation)



Future Work

More results on load balanced job allocation in distributed systems and validation by application to the Grid computing model.

- Develop cooperative load balancing schemes for multi-class jobs by taking the communication costs and bandwidth constraints into consideration.
- Develop dynamic cooperative load balancing schemes.
- Extend the current non-cooperative load balancing scheme to include the communication costs and bandwidth constraints.
- Develop load balancing protocols based on mechanism design that work in distributed systems shared by self interested agents.
- Implement the new schemes in conjunction with job allocation schemes for grids.
- Study the performance of the algorithms using existing distributed systems simulation frameworks (e.g. SIMGRID).

References

- D. Grosu and A. T. Chronopoulos, Noncooperative Load Balancing in Distributed Systems, *Journal of Parallel and Distributed Computing*, 65(9), pp. 1022-1034, Sept. 2005.
- S. Penmatsa and A. T. Chronopoulos, Cooperative Load Balancing for a Network of Heterogeneous Computers, *Proc. of the 20th IEEE Intl. Parallel and Distributed Processing Symposium, 15th HCW*, Rhodes Island, Greece, April 2006.
- S. Penmatsa and A. T. Chronopoulos, Dynamic Multi-User Load Balancing in Distributed Systems, *Proc. of the 21st IEEE Intl. Parallel and Distributed Processing Symposium*, Long Beach, California, March 26-30, 2007.
- S. Penmatsa and A. T. Chronopoulos, Job Allocation Schemes in Computational Grids based on Cost Optimization, *Proc. of the 19th IEEE Intl. Parallel and Distributed Processing Symposium, Joint Workshop on HPGC and HIPS*, Denver, Colorado, 2005.
- S. Penmatsa and A. T. Chronopoulos, Price-based User-optimal Job Allocation Scheme for Grid Systems, *Proc. of the 20th IEEE Intl. Parallel and Distributed Processing Symposium, 3rd HPGC*, Rhodes Island, Greece, April 2006.

Thank you

Questions ?