COLOCATION GAMES
And Their Application to Distributed Resource Management

Jorge Londoño[†] Azer Bestavros[‡] Shang-Hua Teng[þ]
jmlon@cs.bu.edu *best@cs.bu.edu* *steng@cs.bu.edu*
Computer Science Department
Boston University

Abstract

We introduce Colocation Games as the basis of a general framework for modeling, analyzing, and facilitating the interactions between the various stakeholders in distributed/cloud computing environments, where resources are offered in an open marketplace to independent, rational parties interested in setting up their own applications. Virtualization technologies enable the partitioning of such resources so as to allow each player to dynamically acquire appropriate fractions of the resources. When all the components are under the control of a single administrative domain, this leads to an standard optimization problem, but when infrastructure providers make available their resources in a marketplace, and from there customers acquire the resources, the global optimization framework is no longer appropriate. Rather, in this paper we use a game-theoretic framework in which the assignment of players to resources is the outcome of a strategic "Colocation Game". Although we show that determining the existence of an equilibrium for colocation games in general is NP-hard, we present a number of simplified, practically-motivated variants of the colocation game for which we establish convergence to a Nash Equilibrium, and price of anarchy bounds. In addition to these analytical results, we present an experimental evaluation of implementations of some of these variants. Experimental evaluation corroborates our analytical results and also illustrates how colocation games offer a feasible distributed resource management alternative for self-organizing systems, in which the adoption of a global optimization approach would be neither practical nor justifiable.

[†] Supported in part by the Universidad Pontificia Bolivariana and COLCIENCIAS–Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología "Francisco José de Caldas".
[‡] Supported in part by NSF awards CCF-0820138, CSR-0720604, EFRI-0735974, CNS-0524477, and CNS-0520166.
[þ] Supported in part by NSF award CCR-0635102.

1 Introduction

Motivation and Scope: *Cloud computing* has emerged as a compelling paradigm for the deployment of distributed applications and services on the Internet. By relying on virtualized resources, users are able to easily deploy, scale up or down, and migrate their applications seamlessly across computing resources offered by one or more infrastructure providers. More importantly, virtualization enables performance isolation, whereby each application is able to acquire appropriate fractions of shared fixed-capacity resources, ensuring that the application would meet minimal Quality of Service (QoS) requirements.

Cloud computing *users* leverage resources (servers, networks, and storage) hosted by a *provider* and are billed for what they use based on a preset resource-quantum per unit-time price, like a metered utility. It is common the case (*e.g.*[1]) where this cost is bourn by the user whether or not the user consumes the full capacity of the instance. Clearly, it is possible for multiple users to share the use of a cloud resource as long as their aggregate utilization does not exceed its capacity. While such colocation would be attractive to users (as it would lower their costs) it is not attractive to the provider (as it would reduce their profits). This suggests that providers have no incentive to "optimize" the assignment of users to resources – indeed, they have the exact opposite incentive. This implies that it is up to each user in the cloud to minimize its own cost through colocation with other users.

To summarize, given a user's ability to unilaterally migrate from one resource to another in order to reduce its own cost, cloud resource allocation and acquisition is better viewed through a game theoretic perspective. To that end, in this paper we introduce *Colocation Games* as the basis of a general framework for modeling, analyzing, and facilitating the interactions between the various stakeholders in a cloud computing environment.

Related work: Economic models have been used in prior Grid resource management frameworks (*e.g.*, [3, 4]). Typically, these models depend on a broker and users abide to its decisions. This authority does not exist under our model.

On the other hand, algorithmic game-theory has many examples ([5, 9]) where the system self-organizes reaching an equilibrium state, which may be within a bound from the social optimum (*Price-of-Anarchy*). Within these, the closest to our problem are cost-sharing games [7, 2] which deal with the distribution of resource costs among the users sharing it. In particular, *cooperative cost sharing games* [7] deal with the problem of assigning the shares of the costs such that the selected users are all satisfied with the outcome. On the other hand, Anshelevich et al[2] present a pure-strategies cost sharing scheme applied to network formation games (and generalized to allocation of subsets of resources) where the cost is equally shared among all users sharing a single resource. The disadvantage of the first model is that in many cases such self-stabilizing cost-share assignments do not exist, may not be budget balanced or are not fair amongst the users. The limitation of the second model is that it only applies for unweighted resources.

Contributions: We present a game-theoretic model for the interaction of rational, selfish players sharing resources in a distributed environment, where users can easily relocate their tasks subject to QoS constraints. The general model describes the behavior of a wide range of *self-organizing* distributed systems, where all interactions are guided by players' selfish goals. Under this general setting, we show that the existence of a Nash equilibrium is an NP-complete problem. Next, we explore the mechanism design problem of creating a cost function that induces a particular (desirable) user behavior. In particular we explore the goal of maximizing resource utilization so that all users can perform their tasks subject to their QoS guarantees, but minimizing the total (social) cost of the allocated resources. We present a simplified version of the colocation game, called *the Process Colocation Game*, for which we provide analytical bounds on convergence and price of anarchy.

We also present empirical results obtained from two sets of experiments. One based on synthetically-generated workloads to explore the characteristics of the game under a wide range of settings. The other based on PlanetLab traces as to evaluate the game dynamics under realistic scenarios. In addition to corroborating our analytical findings, our experimental results suggest that colocation games could be used as the basis for building distributed, on-line, self-organizing systems, in which the adoption of a global optimization approach (centralized or distributed) would be neither practical nor justifiable.

2 Colocation Games: Definition and Applications

Model and Notation: We use a labelled graph $G =< V, E >$ to model the infrastructure (cloud) resources that are available to the users. Nodes in G represent standalone resources, whereas edges represent relationships between these resources. Examples of standalone resources include processors and storage. Examples of relationships between standalone resources include communication links and spatio-temporal adjacencies, both of which would be represented as edges in G. Here we note that edges in G may be directed or undirected.

Another labelled graph T describes the set of cloud resources and underlying relationships that are necessary to support a specific user application. We refer to this graph as the user requested task graph. Vertices and edges in T have the same meaning as those in the hosting graph G. In a hosting graph, labels specify *supply* attributes such as unit capacities and unit prices of processing or communication links. In a requested task graph, labels specify *demand* attributes such as the minimum CPU utilization and storage needed by a standalone process, or the minimum bandwidth tolerable by communicating processes, *etc.* in the task.

As illustrated in Figure 1, a set of requested task graphs (one per user) constitutes the overall *workload* to be hosted on the infrastructure graph G. A *mapping* M of the set of request graphs to the hosting graph constitutes a configuration underscoring a specific assignment of users to resources. A *valid configuration* is one wherein supply meets demand. Given a valid configuration, the infrastructure provider expects to be paid for any resource in G used by at least one task (user), but not for (idle) resources to which no tasks were mapped. The price charged per resource is fixed, independent of the number of users sharing that resource. The cost incurred by a user is given by a *cost function* which apportions the price of each resource in G among all users with tasks mapped to that resource. The cost function can be seen as the marketplace mechanism that governs and induces symbiotic relationships among rational, selfish agents (the users). In this paper, we adopt a specific form of cost functions that may be conceived as fair – namely, those that split the fixed cost of a resource among tasks in some proportional (*e.g.*, linear) fashion based on the utilization of that resource by the various tasks assigned to it.

The General Colocation Game (GCG): Given a hosting graph $G =< V, E >$ labelled with a *resource capacity vector* (R) and a *price* (P), and given a collection of tasks, each in the form of a graph $T_i =< V_i, E_i >$, where vertices and edges are labelled with a *weight* underlying a resource utilization vector (W), the *General*

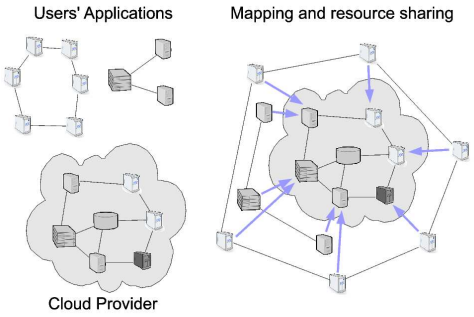

Figure 1: The Colocation Game

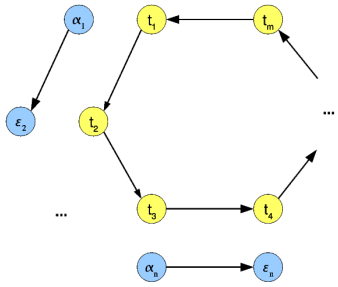

Figure 2: Examples of Colocation Games with no Nash Equilibria.

Colocation Game is the pure-strategies game, in which each task is able to make a move whereby, if possible, the task modifies a valid mapping M into another M' so as to minimize its own cost, given by a function $c_M(T_i)$ for the cost of task T_i when hosted in G according to a mapping M:

$$c_M(T_i) = \sum_{j \in \{V_i, E_i\}} P_j \frac{w_{ij}}{U_j} \quad (1)$$

where w_{ij} is the weight (or utilization) imposed on resource j by task T_i, P_j is the price of the resource, and U_j is the total utilization of the resource.

The *social cost* of GCG for a given mapping M is the sum of the costs of all tasks $s = \sum_{\forall T_i} c_M(T_i)$.

The Process Colocation Game (PCG): PCG is a restricted (simpler) version of GCG. In a PCG, a task graph consists of a *single vertex* representing an independent process that needs to be assigned to a single resource. In a PCG, the cost function for process i when mapped to resource j is $c_j(i) = P_j \cdot w_i / U_j$.

Assuming all processes (users) are rational and selfish, the only move that a process would make in PCG is one that results in a reduction of its own cost *and* would also benefit other processes with which the process would be collocated as a result of the move. It can be easily shown that a user's move from a to b is a *valid move* if it satisfies

$$U'_b > U_b \quad \text{and} \quad \frac{P_b}{U'_b} < \frac{P_a}{U_a} \quad (2)$$

Applications of Colocation Games: We conclude this section with a set of distributed resource management problems, illustrating how each such problem may be cast as a colocation game: 1. *Content Distribution Networks:* where the goal is to cache content chose to the customers, and dynamic resource allocation is important to cope with demand variability; 2. *Service Oriented Architectures* where services need to be instantiated while satisfying resource demands; 3. *Virtual Machine Colocation:* where whole VMs may self-arrange to minimize cost while satisfying their established SLAs.

3 Analytical Results

Nash Equilibrium of GCG: The GCG does not necessarily have a Nash Equilibrium (NE), as illustrated by the example in Figure 2. Here the hosting graph is an m-vertex ring, each vertex has unit capacity. Each of the n ($2 \leq n < m$) tasks consists of two connected vertices, with utilization requirements $1/2 < \alpha_i \leq \alpha_{max} < 1$ and $0 < \epsilon_i < 1 - \alpha_{max}$, respectively. Feasible configurations are consecutive or disjoint nodes. As $n < m$, there will always be at least one edge ($\alpha \rightarrow \epsilon$) connecting a pair of unmatched nodes, as some segment of the hosting graph will not be used. The cost function implies that any task with a free ϵ node will move to match its ϵ node to a neighbor's α node. This leaves another ϵ node unmatched and the process repeats forever.

Theorem 1. *Determining whether a GCG has a Nash Equilibrium is NP-Complete.*

Proof. Omitted due to space limitations. Please refer to the full version [6]. □

Nash Equilibrium of PCG: As opposed to GCG, PCG always converges to a Nash Equilibrium as stated by the following theorem

Theorem 2. *PCG converges to a Nash Equilibrium under better response dynamics.*

Since PCG may not converge to a minimal element, it might be useful to know the *Price of Anarchy* (PoA), *i.e.* the ratio of the worst-case cost in an equilibrium to the cost in a socially-optimal solution. Figure 3 illustrates an example where the equilibria are not optimal. Here resources have unit capacity, the configuration on the left is the social optimal (OPT) and the one on the right is a NE. The large processes have demands $1/2 < l \leq 2/3 - 2e/3$, and the small processes have demand $s = 2e/3$, with $1/e \geq 8$. The PoA is $2/3$. In fact, this example illustrates the worst case when resources are homogeneous, as stated in the following theorem:

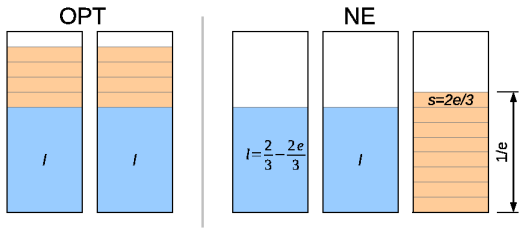

Figure 3: Example illustrating PoA for PCG with homogeneous resources

Theorem 3. *The PoA for homogeneous PCG is 3/2 while the PoA for heterogeneous PCG is 2.*

It is also possible to bound the number of moves that it takes for the game to reach a NE. In the case of homogeneous resources total number of moves is $O(n^2)$ if we impose a minimum cost improvement threshold.

Better Response Computation: It can be shown by reduction from the integral knapsack problem that the best response computation is NP-complete. Fortunately the convergence condition only requires a better response, which can be computed using either a dynamic programming technique (DPKP) or a branch & bound search. For the latter we implemented two versions, one based on the bread-first search (BFS), and the other on a depth-first search (DFS).

4 PCG: Experimental Evaluation

Data Sets and PCG Variants: In our experiments, we used synthetically-generated as well as trace-driven workloads, which we applied to unidimensional and multidimensional variants of PCG under both a homogeneous and heterogeneous resource model. Our *synthetic workloads* give us the flexibility in exploring the parameter space and more importantly, enable us to experiment with workloads for which we know (by construction) the socially-optimal solution. Our *trace-driven workloads* were constructed using publicly available PlanetLab traces of CoMon [8]. The traces we used give us snapshots of PlanetLab server capacities as well as the utilization of the slices assigned to the various tasks collocated on each server. PlanetLab traces describe multidimensional server capacities as well as slice utilizations (corresponding to the utilizations of a slice's CPU, memory, uplink, and downlink). This makes this traces also a good test case for multidimensional PCG.

Metrics: To characterize the PCG game dynamics, we evaluated a number of metrics, namely: (1) the *colocation ratio* which we define to be the ratio of worst/optimal social cost (for synthetic workloads) and of worst/best social cost (for trace-driven workloads); (2) the total number of move *trials* until an equilibrium is reached; and (3) the total number of actual *moves* until an equilibrium is reached. The colocation ratio characterizes the (in)efficiency of the colocation resulting from playing the game, and is bounded by the price of anarchy (PoA). The number of trials gives us insight as to the total time it takes for the game to reach an equilibrium. The number of moves gives us insight in the overhead involved in relocating players (tasks) since each relocation involves migration costs, *etc*.

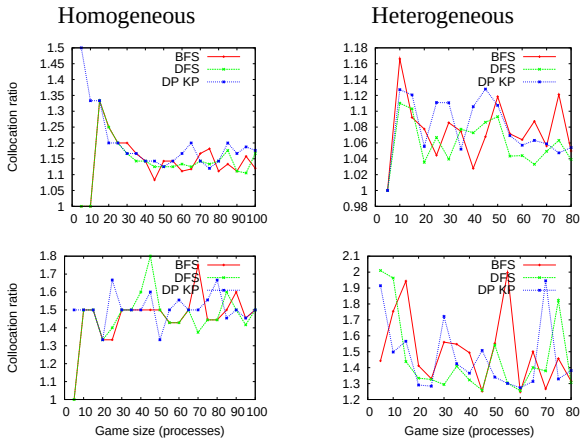

Figure 4: Median (top) and worst-case (bottom) colocation ratio (synthetic)

As shown earlier, verifying that the game is at a NE is an NP-hard problem. Thus, the criterion we used to declare that an equilibrium has been reached was to set a threshold on the number of consecutive trials attempted without resulting in a move. The thresholds we used were such that doubling them (*e.g.*, from 500 to 1,000) did not produce a significant change our metrics.

Colocation Efficiency for Synthetic Workloads: Figure 4 shows the median and worst case (over 100 samples) of the colocation ratios for synthetically-generated workloads in both homogeneous and heterogeneous settings. Recall that in a homogeneous (heterogeneous) setting all resources are (not) of equal capacities. These results show that the PoA bound (of $3/2$ for homogeneous cases and 2 for heterogeneous cases) for the colocation ratio holds for the median (1D). In the worst-case, there were a few samples above the bound, which we attribute to the approximate better-response computation and the threshold for detecting an equilibrium.

The results in Figure 4 show that the colocation ratio tends to decrease (*i.e.*, better efficiency) as the number of players increases, which bodes well for large-scale deployments. Also, our results show that colocation efficiency was basically independent of the better-response heuristic in use.

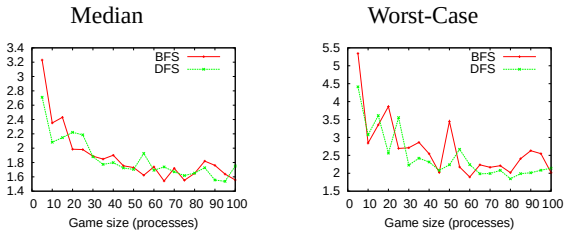

Figure 5: Colocation ratio (PlanetLab)

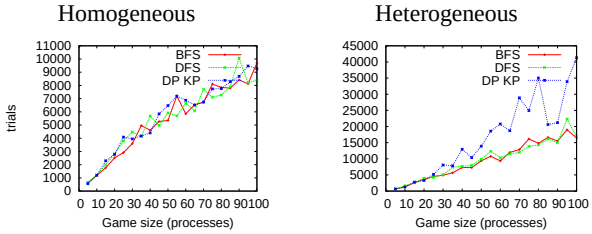

Figure 6: Worst-case trials (synthetic)

Colocation Ratio for PlanetLab Workloads: Figure 5 shows the median colocation ratio (ratio of worst/best social cost) using the task specifications derived from the PlanetLab traces. These results imply a relatively small colocation ratio, which as with synthetic workloads seems to be independent of the better response heuristic used. As shown in Figure 5, the worst-case colocation ratios were not too far off.

Convergence Speed and Overhead: Figures 6 and 7 show the number of trials it takes to reach equilibrium. They indicate that the number of trials is directly related to the number of tasks in the system and fairly independent by the heuristic used to compute the better response. The relation is essentially linear for unidimensional PCG and follows a power law for higher dimensionality PCGs.

5 Conclusion

Colocation Games offer a natural paradigm for modeling and analyzing the dynamics that are likely to result when rational, selfish parties (users) interact in an attempt to minimize the individual costs they incur to secure the shared infrastructure resources necessary to support the QoS or SLA requirements of their applications. Colocation games offer an attractive alternative to approaches that require such parties to trust infrastructure providers (who have no vested interest in minimizing user costs, and may indeed have the exact opposite incentive) or those that expect such parties to be altruistic or to accept best-effort (as opposed to reservation-based) approaches that do not guarantee performance isolation.

In this paper we introduced the general colocation game (GCG) as well as the process colocation game

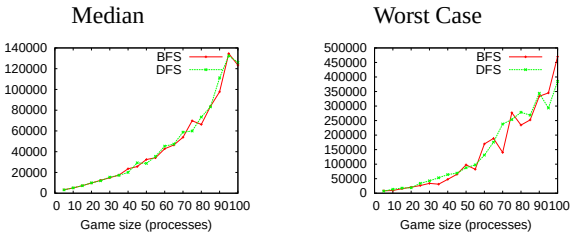

Figure 7: Number of trials (PlanetLab)

(PCG), a more restricted version of GCG, along with many variants for which convergence and PoA results could be derived. Also, using both synthetic and trace-driven workloads, we presented results from extensive empirical performance evaluation of practical and scalable implementations of the strategies underlying these games.

Colocation games are not only valuable as modeling and analysis tools, but also they provide a solid framework upon which purely distributed resource acquisition and management protocols may be conceived for emerging cloud computing, grid, and peer-to-peer overlays. In this paper we have shown that although best response computation may be expensive, computationally-efficient better-response heuristics are quite promising.

References

[1] AMAZON.COM, INC. Amazon Elastic Computing Cloud (Amazon EC2). http://aws.amazon.com/ec2/, Feb 2009.

[2] ANSHELEVICH, E., DASGUPTA, A., KLEINBERG, J., TARDOS, E., WEXLER, T., AND ROUGHGARDEN, T. The price of stability for network design with fair cost allocation. In *FOCS* (2004).

[3] BUYYA, R., ABRAMSON, D., AND GIDDY, J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. In *HPC ASIA* (2000).

[4] GOMOLUCH, J., AND SCHROEDER, M. Performance evaluation of market-based resource allocation for grid computing. *Concurrency and Computation: Practice and Experience 16*, 5 (2004), 469–475.

[5] KOUTSOUPIAS, E., AND PAPADIMITRIOU, C. Worst-case equilibria. *Lecture Notes in Computer Science 1563* (1999), 404–413.

[6] LONDOÑO, J., BESTAVROS, A., AND TENG, S.-H. Collocation games and their application to distributed resource management. Tech. Rep. BUCS-TR-2009-002, Boston University, 2009.

[7] NISAN, N., ROUGHGARDEN, T., TARDOS, É., AND VAZIRANI, V. V., Eds. *Algorithmic Game Theory*, 1st ed. Cambridge University Press, 2007.

[8] PARK, K., AND PAI, V. S. CoMon: a mostly-scalable monitoring system for planetlab. *SIGOPS Oper. Syst. Rev. 40*, 1 (2006), 65–74.

[9] ROUGHGARDEN, T., AND TARDOS, É. How bad is selfish routing? *JACM 49*, 2 (2002), 236–259.