Tit-for-Tat Distributed Resource Allocation

Colin Dixon Tanya Bragin Arvind Krishnamurthy Tom Anderson
University of Washington
{ckd,tbragin,arvind,tom}@cs.washington.edu

ABSTRACT

Distributed computing infrastructures have risen in popularity over the past decade, however resource allocation on these—largely federated—systems remains a practically unsolved problem. Previous attempts have either relied upon central management of resources, which is infeasible in a federated model, and market-based economies which we classify into systems using global, transferrable currency and systems using local, transferrable currency. We present an alternate approach which represents pair-wise, tit-for-tat relationships as local, non-transferrable currency. This approach promises to provide a solution that does not require a globally trusted third party and is easier to deploy and maintain than previous systems.

1. PROBLEM DEFINITION

Recently developed large-scale, distributed, federated computing infrastructures like PlanetLab [2], VINI [4] and the currently planned GENI [1] require resource allocation mechanisms which provide fairness, efficiency, and reasonable performance for each scheduled task while honoring federated control.

In addressing this problem, we keep the following four goals in mind. First, a solution should work in a federated environment where there are many different administrative domains which may not entirely trust one another. Second, systems should not require a globally trusted third party as this explicitly undermines several of the attractive properties of federated environments. Third, per-domain configuration should be kept to a minimum and preferably eliminated as excessive configuration often leads to difficulties in deploying and maintaining systems. Fourth, the provided allocations should give reasonable performance with respect to some notions of fairness and efficiency.

Previous work on distributed resource allocation can be coarsely categorized based on who issues credit or currency, and whether it can be moved. Systems like Bellagio [3] and Tycoon [7] use *global, transferrable currency*, which is issued by a globally trusted mint and easily transfered between parties. These systems cast resource allocation as a constrained optimization problem where each user has a utility function describing the locally perceived benefit of each possible resource allocation. With this formalism in place, it is easy to define efficiency as the ratio of the the total benefit of the current assignment to the maximum possible benefit and fairness as the minimum ratio between two different user's benefits [5]. These two metrics are almost always at odds because the most efficient solution will give all of a given resource to whoever placed the highest value on it in their utility function. The global currency is used to solve this problem, by having each party bid on resources, thus constraining the set of possible solutions to ones which are both efficient and fair. The use of a globally trusted mint to produce all currency in effect gives the mint total power, which seems at odds with the goals of a federated infrastructure.

SHARP [6], on the other hand does away with global currency by having each resource provider issue tickets which can later be probabilistically exchanged for resources. Furthermore, these tickets can be subdivided and reissued to other parties. In this way, SHARP uses *local, transferrable currency* and does not require global trust in a centralized mint. SHARP differs from Bellagio and Tycoon in that it provides a general mechanism for building such systems, but does not aim to explicitly define what policies should be deployed, instead providing some examples of possible configuration. Also, we believe that most of the complexity involved in SHARP comes from the fact that currency can be transferred requiring many mechanisms to secure the currency and agents which can reason about transferrable currency.

While general properties of market-based systems and strategies can be analyzed in a game theoretic manner [5], they rely on accurate utility functions. Defining and maintaining such utility functions requires very smart algorithms, very smart people or both which violates our principle of minimal complexity and configuration. Even with an accurate local utility function, building a bidding-agent to maximize utility on a fixed budget is far from trivial.

Because we feel that the minimal configuration and lack of global trust are extremely important for the adoption of a resource allocation scheme we propose an alternate strategy based on *local, non-transferrable currency*. The ability to transfer currency enables 3-way trades, which are critical in the situations where different entities have non-fungible offerings. On the other hand, in distributed computing infrastructures it is uncommon for the varying administrative domains to provide functionally distinguishable resources as resources usually vary in amount, but not in type. Instead, we maintain currency locally within each domain in the form of credit given to other domains for providing resources in the past, creating pair-wise, tit-for-tat relationships between administrative domains.

2. TIT-FOR-TAT ALLOCATION

In this section, we propose a tit-for-tat strategy for exchanging resources. Each *administrative domain* has a set of one or more *nodes* which are most likely physical machines

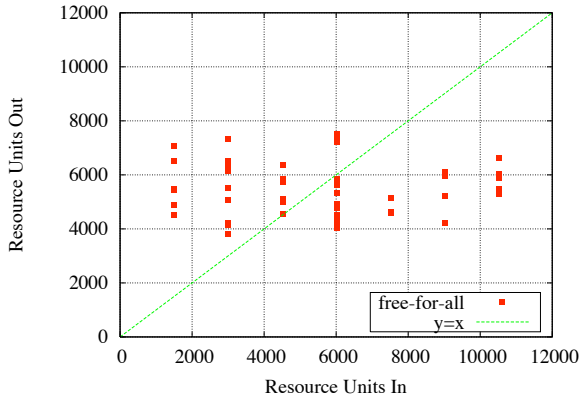

Figure 1: Free-for-All Fairness.

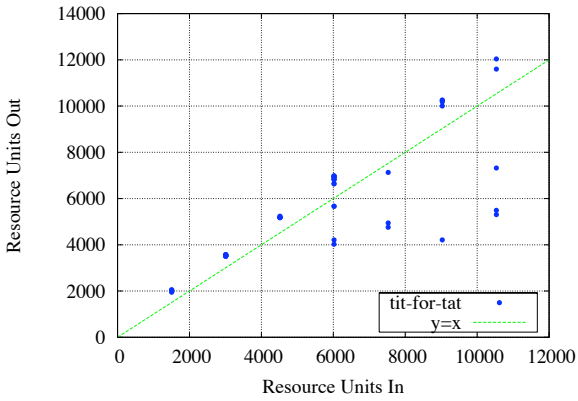

Figure 2: Tit-for-Tat Fairness.

and each node maintains *slots* corresponding to the number of *tasks* it is currently willing to run. To facilitate tit-for-tat bartering, we assume that the set of administrative domains providing resources is the same as the set of domains which consume resources.

In this environment, the challenge is for each domain to schedule all tasks contending to use its nodes into available local slots. Scheduling decisions must be made based on locally available information, such as past history of resources consumed by remote domains on local nodes and resources provided by remote nodes to local tasks. The tit-for-tat strategy is to compare the local resource usage by other domains to the resources they have provided in return and use these ratios to produce a sorted preference list of all other domains. Slots are then provided to tasks owned by the most preferred domians. Information about exchanged resources is allowed to decay over time so that recent behavior is more important, but the time frame in which this decay occurs must be large enough that credit for running tasks can actually be recouped later.

A particularly noteworthy feature of tit-for-tat allocation is that it provides incentives for administrative domains to add more nodes. If an administrative domain wants to get better performance, it will likely be able to get it if it can provide more resources to other domains thus increasing their credit. This solves a current problem in PlanetLab where there are few incentives to upgrade or add nodes other than altruism. Additionally, it would provide incentives to write "good" code for PlanetLab to avoid unnecessarily burning credit.

To test our hypotheses, we built a simulator to evaluate tit-for-tat resource allocation when compared against the current free-for-all scheme used on PlanetLab. The simulator models a PlanetLab-like environment. We assumed an exponential inter-arrival rate for tasks, power-law distributed task-length, and that tasks ran on all nodes at all sites. The following results are from experiments run with 50 sites, each having between 1 and 7 nodes, which can be seen as the 7 columns in both Figures 1 and 2. In evaluating fairness, we looked at the resources put into the system by each site as well as the resources consumed. In a perfectly fair system, no site would be allowed to extract more resources from the system than it put in. As can be seen in Figure 1 this property does not hold in the free-for-all scheduling mechanism which PlanetLab currently employs. If the system were perfectly fair the points would all fall along or below the line $y = x$, but as we can see sites appear to take roughly the same number of resources out regardless of what they put in.

In contrast to free-for-all, the tit-for-tat strategy yields much more satisfying fairness results. Figure 2 shows that most of the sites fall along a line slightly above $y = x$. The reason the points do not fall more closely along the line is because some of the larger sites (those with more nodes) request fewer resources from the system than they are capable of putting in due to randomness in the generated workload. If all sites were constantly contending for as many resources as they could get we would see points falling almost perfectly along the line $y = x$. In contrast as contention decreases, the fairness becomes limited by the workload and the system begins to look more like free-for-all, which allows for reasonable efficiency when contention is low.

3. REFERENCES

[1] GENI: Global Environment for Network Innovations. http://www.geni.net.

[2] PlanetLab. http://www.planet-lab.org.

[3] A. AuYoung, B. Chun, A. C. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, 2004.

[4] A. Bavier, N. Feamster, M. Huang, J. Rexford, and L. Peterson. In VINI veritas: Realistic and controlled network experimentation. In *SIGCOMM*, 2006.

[5] M. Feldman, K. Lai, and L. Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. In *Electronic Commerce*, 2005.

[6] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. Sharp: an architecture for secure resource peering. In *SOSP*, 2003.

[7] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an implemention of a distributed market-based resource allocation system. Technical report, HP Labs, Dec 2004.