Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems

Jeffrey Shneidman[1], Chaki Ng[1], David C. Parkes[1]
Alvin AuYoung[2], Alex C. Snoeren[2], Amin Vahdat[2], and Brent Chun[3]
[1]*Harvard University,* [2]*University of California, San Diego,* [3]*Intel Research, Berkeley*

Abstract

Using market mechanisms for resource allocation in distributed systems is not a new idea, nor is it one that has caught on in practice or with a large body of computer science research. Yet, projects that use markets for distributed resource allocation recur every few years [1, 2, 3], and a new generation of research is exploring market-based resource allocation mechanisms [4, 5, 6, 7, 8] for distributed environments such as Planetlab, Netbed, and computational grids.

This paper has three goals. The first goal is to explore why markets can be appropriate to use for allocation, when simpler allocation mechanisms exist. The second goal is to demonstrate why a new look at markets for allocation could be timely, and not a re-hash of previous research. The third goal is to point out some of the thorny problems inherent in market deployment and to suggest action items both for market designers and for the greater research community. We are optimistic about the power of market design, but we also believe that key challenges exist for a markets/systems integration that must be overcome for market-based computer resource allocation systems to succeed.

1 Is there a Problem?

During the past decade, we have witnessed the emergence of systems that are owned, deployed, and used by multiple self-interested stakeholders. Consider the differences between traditional distributed systems and current distributed environments, such as Planetlab, Netbed, and computational grids. Current environments have the following properties:

Many resources, many users, and more complicated needs. Multiple self-interested parties can simulatenously supply and consume sets of resources (e.g., machine time, bandwidth). Users can demand large sets of disparately controlled resources, creating a large combinatorial allocation problem not easily solved by techniques like social pairwise agreements.

Resource demand exceeds resource supply. Previous work has graphically demonstrated this problem on Planetlab, where the machine load is many times the system capacity [9]. Scientific computing (grid) users expect this to be a problem as they deploy experimental testbeds [10].

No job selection by committee. The scale and design goals of these systems preclude an administrative body to handle resource allocation.

Incentives and external constraints limit supply. Political, financial, and geographic limitations prevent additional hardware deployments to solve all cases of resource contention. Unlike commercial servers that have a financial incentive to support their peak user load, resource providers in shared environments usually have little incentive to add resources to the shared system.

Testbed-sensitive experimentation. In some shared environments (e.g., Planetlab), the network itself is the target of research. A *tragedy of the commons* [11] can develop where overlapping usage consumes resources to the point of disutility and users are unable to run certain class of measurement experiments accurately or at all.

Computer systems have reached the point where the goal of distributed resource allocation is no longer to maximize utilization; instead, when demand exceeds supply and not all needs can be met, one needs a policy for making resource allocation decisions. Researchers (Planetlab central, Grid planners, etc.) have

started to consider more intelligent ways of allocating resources than simple best effort, or randomized allocation schemes.

These methods can involve a *social policy* for resource distribution. A policy is simply a set of rules for allocation when resource demand exceeds resource supply. One candidate policy is to seek *efficient* usage, which directs a mechanism to allocate resources to the set of users who have the highest utility for the use of the resources. Other social policies exist, such as those that favor small experiments, or favor underrepresented stakeholders, or (if money is involved) seek maximal revenue generation. One can also implement a mixture of policies to meet a complex social goal.

Past deployments of distributed system schedulers (e.g. Condor [12]) focused on maximizing utilization, and were not designed to support complex social policy. Today's schedulers must take full utilization as the common case and focus on solving the resulting resource contention problems.

In this paper, we explore the idea of using market-based mechanisms to address resource allocation problems in distributed systems. In Sections 2 and 3 we explore how markets may be a useful (and perhaps required) tool in this research and why they warrant new consideration by systems researchers. However, there are special challenges that arise when markets are used for computational resource allocation. These challenges, presented in Section 4, could prove overwhelming depending on the response from the systems community and our collective ability to address these concerns.

We feel that now is a critical time for the systems community to consider the various resource allocation capabilities that should be supported in next-generation distributed systems, before an uninformed decision or simple necessity leads to a less desirable, de facto standard.

2 The Role of Markets

If one is interested in performing policy-directed resource allocation, one should consider allocation schemes that are based around a market.

A market is a way for buyers and sellers to exchange goods. Applied to computer resource allocation, the traded goods could be the right to use a certain amount of system resources on a set of machines. When demand exceeds supply, markets provide a goal-oriented way of allocating resources among competing interests while meeting some social goal. One natural goal is to maximize overall "happiness" or utility of the users. When users have complex needs, achieving this goal is not easy for either the individual users and for the system tasked with making the allocation decision. We will return to these issues in Section 4, but for now we consider the advantages of markets for computational resource allocation.

Deploying a computational market for resource allocation in the systems domain can benefit two research constituencies. The first constituency, which will be ignored for the rest of this paper, are the experimental economists and economically-minded computer scientists. Rarely are economists actually given the opportunity to deploy a market or a whole economy, let alone several for comparison. Computational mechanism design [13] is an emerging topic partly because the results apply to many different domains, and there is some merit in asking systems researchers to be research subjects as they attempt to use some market mechanism for their own work.

But systems researchers (the second constituency) are much more interested in knowing if these proposed market allocation projects and their system offspring solve real problems in distributed resource allocation. There are many programmatic alternatives to markets in resource allocation. These include simple first come-first served allocation, reservation systems, and more elaborate systems such as automated voting schemes or other devices. Unlike these simpler ideas, market-based systems can naturally address the new-world system characteristics described in Section 1. Namely, market-based systems can:

Provide a "socially optimal" project director to resolve overdemand. Unlike simpler mechanisms, markets can support a rich set of social goals, such as finding an efficient allocation decision. The most natural way to reach an efficient decision is to require users to quantify their perceived benefit of winning their resource request. A market encourages participants to use resources wisely and tries to make an overall usage decision to maximize overall value.

Provide incentives for growth. Markets are often used along with a currency that can be used to express value and acts as a medium of exchange.[1] If a currency is *open* and can be used to acquire a multitude of goods and services, then this currency can be used to incent resource providers to expand their services. In contrast, a *closed* currency can incent growth only if the receiver of the currency has some use for its receipt. One can use currency to create a medium to allow a market's "invisible hand"

[1] Currency is a natural means toward easy valuation expression, but there are other allocation algorithms that do not require currency. An example are the matching algorithms that link Medical Interns and Residents in the United States [14]. In this setting, medical students and residency programs bid on each other using a prioritization scheme, and these bids are resolved with a winner determination algorithm. At first blush, a matching market does not seem appropriate to systems resource allocation problems, where sellers have no preference of who uses their resources.

to reward those who provide useful resources to the network. Markets provide a vetted set of payment rules that can be used to transfer currency between buyers and sellers.

Provide a vocabulary to describe complex resource bundles. In any system, be it administrative or market-based, users need a mechanism to express their resource holdings and desires. Markets, which have been used for decades to capture difficult resource allocation problems (e.g. energy markets, wireless spectrum auctions, airline landing slot exchanges), can also be used to capture the intricacies of systems problems. Bidding languages have been studied for their tradeoffs between expressivity and compactness [15], and existing languages can be directly applied to computer resources.

Link Cross-Testbed experimentation. Multiple closed distributed systems that run in parallel can offer unique resources such as access to specific scientific equipment. One can imagine a physics researcher willing to provide access to their Beowulf cluster [16] but wishing to consume resources produced by data collectors at a CERN [17] on a completely separate network. Linked market-based mechanisms could be used to quantify the value of the cluster time sold in one network and the value of a CERN resource purchased in another network in a manner similar to how real economies are linked through a a currency exchange. Ongoing research into exchange mechanisms for computational systems could make this vision feasible [7].

3 Not Déjà Vu All Over Again

The idea of using markets and pricing computer resources is quite old. Pricing policies received considerable attention at the dawn of modern multi-user time sharing systems. Papers in the late 1960's were dedicated to automated pricing policies for computer time [18, 19, 20]. As research, this work was short-lived. The complexity of these schemes relative to their benefit, combined with the environment of time-shared systems (mostly cooperative, mostly controlled by a single entity) quickly made pricing for shared resource allocation a low priority. Shared resource allocation remained a hot topic in operating systems, but the goal in this research was maximizing *utilization* through clever scheduling. In contrast, schedulers that promote social goals such as *efficient* usage have not been as widely investigated.

This said, there have been past systems that take a market approach to resource allocation [1, 2]. How, then, will new research into markets for distributed resource allocation be any different? We believe that a number of developments make the timing right to revisit the question of whether market-based models are both appropriate and, more importantly, required for emerging computational environments. New research can take advantage of the following developments:

Pressing demand. Past market-based systems never saw real field testing, and contention was often artificially generated. Today, a deployed market system could have immediate usage and solve real resource conflicts. Real usage data will help researchers calibrate and evaluate their market-based resource schedulers. Previous mechanism designs were not able to take advantage of user feedback to drive the mechanism design process.

Improved operating system infrastructure. Past systems had to deal with limitations in infrastructure, such as a lack of user authentication or kernel-supported resource isolation. Today, systems research has produced tools like BSD Jails, Xen, and Linux CKRM [21, 22], which are already in use to provide resource isolation, can be adopted to enforce allocation decisions.

Expressive market design. Previous work used bidding languages that have been artificially limited in their expressive power. During the past decade, tremendous advances have been made in the theory and practice of expressive market design. Current mechanisms can support combinatorial bidding, which more naturally captures resource needs. For instance, modern bidding languages can easily represent any logical combination of goods, such as AND, OR, XOR, and CHOOSE. This expressive power did not exist in previous mechanism deployments.

Scalable mechanisms. Solving large resource contention problems has traditionally been computationally expensive. Fortunately, significant advances have been made in the theory of solving large-scale mixed-integer optimization problems, which is an underlying technology well-suited to implementing market problems. This theory is now reflected in off-the-shelf solvers such as CPLEX. Significant breakthroughs have arisen from the use of cutting plane techniques, branch-and-cut, and preprocessing to achieve efficient solving.

4 Markets/Systems Integration Challenges

Despite our general optimism, the ultimate success of a deployed mechanism is measured in usage, and usage depends on a number of factors typically overlooked by computer science researchers. Ease of use may trump mechanism features. People may be willing to accept

the limitations of simpler systems (eg: first-come first-served, or randomized allocation) if market-based systems are seen as too complex, or if they fail in other ways, even if accepting a simpler system means ignoring some of the characteristics described in Section 1.

In this section, we articulate the roadblocks that must be addressed to make a market/systems integration successful. In our opinion, these challenges are not in the market details. Rather, we think that the biggest challenges to their adoption in systems will come from understanding, supporting, and using these mechanisms. After presenting each challenge, we consider *action items* for the general systems community, as well as for systems market designers where appropriate. In our view, a markets/systems integration could fail if these challenges are not overcome:

Allocation Policy Must be Explicit. One of the uncomfortable realities of a market is that it forces user communities to confront their social allocation rules. Do people want allocative efficiency? Do people want testbeds to be self-sustaining through policies that imply taxation? Do people want to favor jobs from underrepresented users? Other real-world uses of markets have had definite mandates. As an example, after years of running a lottery to allocate wireless spectrum, the U.S. Congress wised up to the resulting allocation inefficiency (not to mention the possibilities of revenue generation with the government as the initial sole seller), and mandated that the F.C.C. to employ an efficient allocation mechanism. This was a clear social choice, and necessarily meant the F.C.C. used a market.

Community Action Items: There is no general mandate in the systems community for the social goal of an allocation scheme. If the systems community cares about simpler goals than efficiency or revenue generation, than systems market designers should not be trying to develop auction mechanisms. Where should this mandate come from? HotOS participants? Planetlab Central? Grid users?

Dividing Up Resources as a Seller. Unlike many other markets, there are complex and not commonly understood systems interactions between computer resources, complicating the allocation decision. Consider a system that allocates three hard resources, CPU, memory, and disk: An allocation of memory is meaningless unless there is some small CPU associated with the allocation. If virtual memory is involved, it is likely that disk also needs to be allocated, but that the effects of swapping will dominate the time required to run the experiment. Either these associations are explicit, in which case minimum resource bundles must be purchased, or there are side effects that constrain the allocation based on the characteristic of winning bids.

Systems Market Designer Action Items: While the tools (like CKRM [22]) for partitioning resources are being developed, they still have a long way to go to capture pertinent resources and even trivial resource interactions.

Predicting Needs as a Buyer. It is difficult to describe precisely the level of resources required to run an experiment or job. Depending on the inputs to a program, the ideal level of resource consumption can vary dramatically.

Moreover, there is a tangible penalty for misestimating resource need, since these bids are made in advance of when the resources will actually be available. In order to match enough buyers with sellers, current market-based resource allocation schemes batch allocations into blocks of time. The time scale of this batch system can be minutes or days ahead of when the resources will actually be made available. This means that users must predict their resource needs in advance. A resource underbid will prove unsatisfying if won, while a resource overbid (with the same value) is less likely to win because of competition from more efficient users. Requiring users to predict their resource need is new user behavior, and this forecasting problem can be difficult.

Community Action Items: The general systems community should think more about building tools to help users estimate their resource needs. Perhaps users in a shared environment will have access to a best-effort staging ground where they will be able to gauge their resource usage. One can imagine future research tools (either modeling or analysis) that attempt to capture the resource profile of a wide-area application. Such tools are an open area for ongoing and future research [10]. *Systems Market Designer Action Items:* While there is ongoing research into online market mechanisms—making an allocation decision before seeing all bid activity—designers should develop markets that are less rigid in their clearing time frames, while still meeting social goals.

Valuing Resources. Utility maximizing market mechanisms are only as accurate as the values that users assign their bids (on goods that they possess, and goods that they would like to acquire). But what is a user's true value on four hours of CPU time, a week before a major conference deadline? (*Any* situation where demand exceeds supply will lead to unhappy users; a variation of this question exists in any resource allocation scenario.) Ultimately, the requirement of the market is that users place a value on their resource needs and holdings. There are several problems with calculating this value in computational systems. We label these as problems with a

well-defined currency, and in calculating and expressing valuation:

Well-Defined Currency. Almost all previously deployed computational markets have used a virtual currencies instead of real cash. The low barrier to utilization and low stakes in case of deployment error make simple closed virtual currencies attractive to developers. In these scenarios, it is all too easy to skip the monetary policy considerations that make currencies work.

For all of their bootstrapping advantages, virtual currencies require initial thought and ongoing care to function properly. Virtual currencies often suffer from a lack of liquidity, making it difficult to convert into or out of the virtual currency. As a result, these ersatz currencies are quite limited; certain users might be willing to sell resources for Euros, but not for un-exchangable Woozies. Furthermore, virtual currencies can suffer from *starvation*, as heavy consumers run out of currency to spend, *depletion* as users leave the system or hoard currency reducing the total amount of currency available to others, and *inflation* as users are added to the system with an initial credit. Previous research attempts to address the faults of virtual currency systems with monetary policies and administrative measures (e.g., [23]), but for a virtual currency to work, it must be expressive and appreciated by users.[2]

We believe that the success of a computational resource exchange will be tied to a well-defined currency. Rather than attempting to create such a currency, one could turn to real money as the medium for exchange. One reason to use a real currency is that it may increase resource contribution and ease maintenance of distributed environments. Using Netbed or Planetlab as an example, many entities are passive, light users, and may not see the value of maintaining their portion of the network beyond their initial required contribution. Whereas these users may not respond to an allocation of a closed virtual currency, they may respond to real money. Using a real currency could help increase participation in a distributed system – since supply and demand set the price of contributed resources, the network has a way of rewarding those who provide useful offerings to the network. Using a real currency also might provide a lower barrier to entry for new users and create a self-sustaining shared environment: rather than charging new organizations a fixed usage fee, or relying on external grant money for support, one can imagine transaction fees that support the development of the testbed.

We believe that there is no technical reason that prevents one from using real currencies on shared environments. There are numerous political and fairness concerns with this idea. Researchers don't like the idea of having a resource request denied because other researchers could pay more money. (We do observe that the existing research grant process potentially creates this sort of situation.) But in a world where demand exceeds supply, and one has chosen to resolve this problem efficiently, one needs some understood way of expressing valuation differences. Perhaps using a real currency is a wacky idea (that works for every other market) whose time has come?

Community Action Items: If efficiency is an important social goal, then we see valuation questions as a big challenge for the systems community. We wonder if users would be willing to try something novel (which is old hat to every other use of markets) and pay for their bids with real currency. While there are issues with this idea, it does force people to put money where their valuations are. *Systems Market Designer Action Items:* We would like to see a careful construction of a virtual currency system, or alternatively, a careful construction of an argument as to why these systems do not work. We feel that a well-defined currency is a major stumbling block to market adoption in systems.

Calculating and Expressing Valuation. It can be difficult for a user to accurately value their ideal resource bundles. There needs to be a simple and effective way for people to express their resource need and calculate its value. To stress this point, imagine a market interface that asked the user for their valuation, one question at a time, over the entire space of good combinations. This painful approach would require the user to think about their valuation for a whole slew of bundles, a time-consuming and sometimes difficult task. An area of market design that has received almost no attention for computer resources is in the user interface between the users and the mechanism. The bidding interface is the most public face of a market mechanism, and in our opinion it is this interface that has the greatest effect on user perception (and acceptance) of the mechanism as a useful tool.

Community Action Items: Be willing to give feedback to designers on how well a language/interface is at capturing your resource desires. Be willing to suffer through some bad research designs. *Systems Market Designer Action Items:* Improving price guidance and addressing

[2]One interesting note is that the new breed of multi-player online games often have a virtual in-game currency component. Operators of these online games either openly support the exchange of their currency into other real currencies [24], or attempt to keep their currency closed, effectively incenting players to open these closed currencies by spawning parallel side exchange markets [25].

valuation complexity are currently active research areas in mechanism design, and this effort will likely continue.

5 Conclusion and Challenges

We feel that the time is right to explore market-based resource allocation mechanisms, but we also see a number of challenges that may hinder their applicability to systems. While there has been a general call for better resource allocation, it is not clear to us that systems researchers will be willing to accept the implications of mechanisms to achieve certain social goals. These market designs need to be debated, and if deemed valuable, deployed and evaluated "in the wild".

References

[1] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and S. Stornetta, "Spawn: A distributed computational economy," *IEEE Transactions on Software Engineering*, vol. 18, no. 2, pp. 103–177, February 1992.

[2] A. S. Tanenbaum, S. J. Mullender, and R. van Renesse, "Using sparse capabilities in a distributed operating system," in *Proceedings of the 6th International Conference on Distributed Computing Systems*, 1986, pp. 558–563.

[3] O. Regev and N. Nisan, "The popcorn market – an online market for computational resources," in *Proceedings of the 1st International Conference on Information and Computation Economies*, October 1998.

[4] R. Wolski, J. Plank, and J. Brevik, "G-commerce – building computational marketplaces for the computational grid," The University of Tennessee at Knoxville, Tech. Rep. CS-00-439, Apr. 2000.

[5] R. Buyya, J. Giddy, and D. Abramson, "A case for economy grid architecture for service-oriented grid computing," in *Proc. of HCW '01*, Apr. 2001.

[6] A. AuYoung, B. N. Chun, A. C. Snoeren, and A. Vahdat, "Resource allocation in federated distributed computing infrastructures," in *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure*, October 2004.

[7] D. Parkes, R. Cavallo, N. Elprin, A. Juda, S. Lahaie, B. Lubin, L. Michael, J. Shneidman, and H. Sultan, "ICE: An iterative combinatorial exchange," in *Proc. of ACM Conference on Electronic Commerce*, June 2005.

[8] B. A. H. Kevin Lai and L. Fine, "Tycoon: A Distributed Market-based Resource Allocation System," HP Labs, Palo Alto, CA, USA, Tech. Rep. arXiv:cs.DC/0404013, Apr. 2004.

[9] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat, "SHARP: An architecture for secure resource peering," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, October 2003.

[10] S. Grinstein, J. Huth, and J. Schopf, "Resource predictors in hep applications," in *Proc. of Computing in High Energy and Nuclear Physics (CHEP)*, September 2004.

[11] G. Hardin, "The tragedy of the commons," in *Science*, 162(1968):1243-1248.

[12] "The Condor Project: http://www.cs.wisc.edu/condor/."

[13] N. Nisan and A. Ronen, "Algorithmic mechanism design (extended abstract)," in *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 1999, pp. 129–140.

[14] A. E. Roth, "Game Theory as a Tool for Market Design," http://kuznets.fas.harvard.edu/ aroth/design.pdf.

[15] N. Nisan, "Bidding and allocation in combinatorial auctions," in *Proceedings of the 2nd ACM Conference on Electronic Commerce*, October 2000.

[16] "Beowulf Cluster Project: http://www.beowulf.org/."

[17] "CERN Particle Physics Experiments: http://www.cern.ch/."

[18] D. S. Diamond and L. L. Selwyn, "Considerations for computer utility pricing policies," in *Proceedings of the 1968 23rd ACM national conference*. ACM Press, 1968, pp. 189–200.

[19] I. E. Sutherland, "A futures market in computer time," *Commun. ACM*, vol. 11, no. 6, pp. 449–451, 1968.

[20] F. J. Corbato and V. A. Vyssotsky, "Introduction and overview of the multics system," http://www.multicians.org/fjcc1.html.

[21] P. R. Barham, B. Dragovic, K. A. Fraser, S. M. Hand, T. L. Harris, A. C. Ho, E. Kotsovinos, A. V. Madhavapeddy, R. Neugebauer, I. A. Pratt, and A. K. Warfield, "Xen 2002," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-553, Jan. 2003. [Online]. Available: http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-553.pdf

[22] "Class-based Kernel Resource Management (CKRM): http://ckrm.sourceforge.net/."

[23] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "Karma: A secure economic framework for p2p resource sharing," in *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, 2003.

[24] "Online Gaming gets Feeding Tube," http://wired-vig.wired.com/news/ebiz/0,1272,67277,00.html.

[25] "Gaming Open Market," http://www.gamingopenmarket.com/.