Fundamental Properties of Market-Based CPU Scheduling across the Internet

Brad Penoff

University of British Columbia
201-2366 Main Mall
Vancouver, BC V6T 1Z4 CANADA
penoff@cs.ubc.ca

Abstract

This paper outlines the issues around scheduling CPU-time across the Internet. Many have used market-based approaches to solve this problem, however each failed to produce a thorough, realistic solution. We offer some desired properties an ideal market-based solution would need and offer suggestions to existing solutions for satisfying these fundamental properties.

1. Introduction

Over time, the computational strength available to users has rapidly increased through the development of faster individual CPUs. As this has occurred, there has continued to be a development of computationally intensive applications that still yearn for further extensions of computational power to do their work more quickly. Faster and inexpensive networking speeds, expansion of computer sales, and rapid growth in popularity of the Internet indicate that, in theory, there exists an additional supply of reachable computing power for such applications. It is therefore a possibility that extra computational capabilities could come from the collaboration of networked computers willing to donate their resources, particularly their CPU cycles.

Under this scenario, there would be users wishing to run their applications and also hosts with extra CPU cycles to give away. Once these respective sides are known, there needs to be a means of matching members of these two parties. One decision that needs to be made is whether or not to assign a CPU to a particular task. This is typically defined as a problem of resource allocation. However, there is also often a need to know when exactly a computation should be performed. When this time element is a factor in addition to the allocation of the resource, this is termed to be a scheduling problem. As such, scheduling is just a more specific instance of resource allocation.

Scheduling in distributed computer systems has long been researched, both in single and multiple CPU situations. Many proposed solutions have revolved around the use of heuristics, and others involve the use of deterministic mathematical models. Particularly, research in one branch has involved the use of economic algorithms in what have been called market-based scheduling strategies. Originally, such strategies were used in single CPU setups [8] but more recently, they have been applied in networked, multi-CPU setups. Within this branch even, researchers have taken various approaches, each with their own goals and assumptions. Based on the results from these approaches, in this paper, we will demonstrate what properties an ideal market-based approach should possess when trying to realistically solve the problem of allocating CPU-time across the Internet.

This paper is set up as follows. After this introduction, we move on to briefly describe some key technical issues surrounding the general problem of Internet-based allocation of CPU-time. From there, after some initial assumptions, we present and argue for what we believe are fundamental properties which should be required by any realistic, market-based solution to CPU-time scheduling across the Internet.

2. Technical Issues

The allocation of CPUs over the Internet brings up several technical issues independent of the means chosen to attempt to perform the actual scheduling. Firstly, there may be concerns about whether a given application can even be run on different platforms in a heterogeneous environment due to lack of portability. But supposing the applications can run, the next question is why a host would be motivated in the first place to make their resources eligible for use. Provided they are just nice, they may still have trust concerns since someone else is using their resources, possibly maliciously. Trust factors can go both ways though. How can we say that hosts are not telling lies about the answers calculated?

Even more issues come up when tackling this scheduling problem. First, most systems are not connected to the Internet at all time. People coming and going might affect which hosts can still help perform computations not to mention if an application currently being run needs to even continue if the user who submitted the job has recently vanished. Not only must the scheduling approach be fault tolerant, but it must also bear in mind the scale that they intend to accommodate; here since we are on the Internet, the potential numbers could be quite vast.

Other key issues crop up around communication costs. Obviously, the total cost of communication and computation should be less than the time saved if a user were to simply do the computations themselves. There are several communication costs though, and they can be further split up and analyzed separately. There are the costs associated with initially establishing who needs which resources and who can supply these facilities. Further, there will be costs for doing the actual CPU assignment, for example, by passing the actual executable from the user to the host. In the case of parallel applications, there may also be a need to coordinate calculations with tasks running elsewhere. In an IO-bound application, the quality of task assignments would therefore be heavily dependent on the latency between the nodes chosen by the mechanism.

The knowledge available amongst agents will also play a role with regards to its capabilities. One simple assumption could be that each agent has full access to all information in the system. This information would include exact runtimes, exact desires for users and hosts, exact arrival and departure times for agents, etc. However, under most circumstances, all of this information will not be available for one of several reasons. On the one hand, agents may contain conflicting and competing private information. On the other hand, it may be impossible to obtain exact information in the first place, for example, an accurate, universally compatible benchmark of a task's computational requirements may not be achievable. Similarly, a user or host could go offline due to some unforeseen networking failure; expecting to know everything ahead of time would typically be unrealistic, so the scheduling means must compensate.

3. Assumptions in Market-Based Approaches

As the previous section has demonstrated, there are many inherent issues in scheduling CPU-time across the Internet. All scheduling solutions must bear these issues in mind and attempt to address each as well as they can. Accordingly, market-based scheduling approaches are no exception.

When using markets as a means of scheduling CPU-time over the Internet, often the way that it is basically framed is quite similar from one approach to the next [1, 5, 7, 10]; it is in the fine details which they differ. The goods in the micro-economy are some quantification of required computation e.g. wall time, operations, tasks, etc. Typically, there are two types of agents: the users wishing to run their intensive applications are viewed as the auction buyers, with those lending out their CPU cycles treated as the sellers. All agents share recognition of some common electronic currency. Bids and asks are made for the goods in terms of this common currency.

One aforementioned trait of trying to collaborate across the Internet is the fact that a given agent on the Internet could become disconnected at any time. Our mechanisms must therefore deal with such catastrophes, and their models must not make any assumptions about the arrival times or persistence of a given agent. All of the approaches studied here did this appropriately by treating their respective markets like an online algorithm.

There are other properties common amongst market-based approaches. One is that everyone desires a mechanism that is computationally feasible. As we will see, this wish often requires tradeoffs elsewhere in the mechanism. Another property which all share is the desire to minimize communication introduced in the markets overhead. This is handled in most approaches through the use of sealed-bid based auctions.

These are the only assumptions we have made across all micro-economies. We have deliberately not assumed certain design characteristics simply because they differ quite drastically within approaches. Examples of those not assumed are items like bidding language expressiveness, market type, currency assumptions, and information availability, to name a few. The choices the various approaches took with regards to these overlooked assumptions will be the fuel to the fire of our statement and discussion of the fundamentals for market-based approaches to scheduling CPU-time across the Internet.

4. Fundamentals in Market-Based Approaches

As indicated earlier, opening the playing field up to potentially include any system on the Internet simply indicates that the scale could be quite vast. This tells us that any realistic usage of markets would therefore not have the market itself be a bottleneck. This leads us to our first property:

(1) The market chosen must not be a centralized solution.

In the POPCORN approach [7], many market types are experimented with but each shares the trait of being a centralized solution. They excuse this as being appropriate by saying that each computational chunk must be “CPU-time-consuming enough relative to the market overhead”. This is an obvious fact of all systems relationship to their

overhead but essentially what they are saying here is that whatever the degree of bottleneck their approach presents, it is the market agents that are responsible for dealing with it. A decentralized solution would instead strive to avoid the existence of a bottleneck in the first place by placing less computational burden on a central market.

Decentralized approaches have been vastly researched for scheduling. Here, each agent calculates its own bidding strategy, based on local information. Walsh [10, 11] discusses a decentralized protocol that was designed for scheduling. Spawn [9] also takes a decentralized approach.

Bredin [1] has an approach that primarily is concerned with mobile code that moves around a network to sets of required computational resources. In this sense, they claim that their code mobility "provides an extra layer of fault tolerance" in that if one of the resources their code had wanted to use were to suffer catastrophe, then that code could simply instead choose another similar resource. Essentially, this is true of any market-based approach, so this is by no means unique. But here, in his setup, this relocation occurs through a piece of code successfully bidding for a resource in some central auction[1]. If the auctioneer itself were to become unreachable, the entire approach would come to a standstill. This again proves the need for a decentralized approach.

It was also alluded to that in order for a host to offer their resource on the Internet, more than likely they will need some sort of motivation since the Internet is an environment where not everyone necessarily cares intimately about each other already. We assumed that a market-based approach would have some notion of a common currency. Regarding this currency, we must then establish yet another property:

(2) Money, regardless if it is fake or not, must represent incentive and priority. Agents must value having more money.

In our own consumer world, humans generally value having more money than not. The same must be analogously true of problems framed in terms of markets in order for this money to fully represent an agent's genuine incentive. Bredin's approach genuinely fails to capture this property with its version of currency. Here, the programs themselves possess the money, and when the money is spent, it evaporates in that the sellers do not keep track of their obtained funds. They say that there "is no additional utility for an agent to have a positive endowment after completing all of its jobs". In other words, having extra money in the end is therefore not valued. We believe that Bredin's approach would more accurately model incentive if instead the agents persistently possessed the money.

POPCORN, for example, appropriately encourages collaboration by satisfying property 2. Exchanges of currency take place with each sale, with the auctioneer making no profit i.e. the auction is budget balanced. The market centrally and persistently maintains agents' current funding allotments. Agents prefer to minimize their expenditures and maximize their gains, successfully motivating CPU sharing and competition.

As mentioned, there are also many trust factors encountered when collaborating across the Internet. Sellers could worry about malicious use of their facilities. Technical answers to this have included what is known as sandboxing, or the placing of restrictions to resource access [4]. Buyers could also worry about false reports of computed values. For this, cryptology-based solutions have been proposed [3]. Specifically, in the case of markets-based approaches, one could imagine that a given agent could forge some electronic currency. These trust issues combined bring us to our next property:

(3) Solutions should not have any assumed trust.

Most approaches currently do not take this property fully into account. Perhaps, this is because their focus was more on studying the use of markets applied to this particular resource allocation problem. For instance, SPAWN even admits it is lacking in security when mentioning future work. In POPCORN, the extent of the security is that the money itself is always kept within the accounts kept by the market, which requires a password to login. In terms of real-world deployment though, any approach would be a more realistic solution having the appropriate security extensions like sandboxing and cryptological login and answer checking.

Moving onward, we mentioned in our technical issues discussion that another concern which scheduling approaches must have is with regards to the amount of knowledge available to the agents at the time of preference submission. We established that a realistic scenario would typically not be all knowing. This leads us to our next property:

(4) Agents must work as best they can from imperfect information.

Approaches like Bredin's make the unrealistic assumption that there is perfect information across agents. Obviously, this could be more realistic if it was not the case. The MAJIC system [5] takes a different approach that we believe satisfies property 4. MAJIC is a system that seeks to design a general-purpose resource allocation scheme through the use of markets. By resources, they mean not just CPUs but potentially printers, databases, services, etc.

[1] Bredin's focus is on a centralized approach but briefly makes reference to how a potential decentralized alternative would look.

One could imagine that with complex services, like printing, users may want particular properties of a printer and the services it can perform. It therefore would be unrealistic to only have perfect matches since an available printer may satisfy some of the sought-after properties. Accordingly, their matching mechanisms are able to estimate the closeness of a match by requiring buyers to submit a parameter search engine that fully expresses their resource property preferences. Having the buyers submit this engine to the market along with their bid disables them from being able to strategize their bid amounts with regards to the current available sellers. The market hides the preference lists to the sellers and the property lists of the sellers from the buyers. Agents are induced to partially reveal some private information to the mechanism in exchange for obtaining a more desirable schedule. All agents in the system have imperfect information with respect to each other, as would be the case on the Internet.

Yet another common desire that we mentioned was minimizing communication costs. Within the market, this is done through the use of sealed-bid auctions. Let us say that a buyer has a parallel application consisting of several tasks. The tasks are considered complementaries in economic terms since the user will value all of them getting executed more than the sum of having only run each task[1]. Additionally, the user wants to minimize his own coordination communication costs between his respective tasks. The user needs to somehow make the market aware of these desires. This requirement leads to our next fundamental property:

(5) The bidding language should be expressive enough to efficiently relay agents' desires but not overly expressive as to compromise computational feasibility.

The bidding language's expressiveness of approaches like Bredin's, POPCORN, and SPAWN are mostly limited to bid ranges and execution time restraints. Each could be bettered by of course allowing agents to relay more information. But adding expressiveness cannot guarantee that in all cases the globally optimal allocation will be established. In fact, this never occurs in such an online algorithm [2]. Optimal allocations are intractable to obtain. Operating under the assumption that we also desire computational feasibility, it can then be said that we must compromise slightly solution quality, no matter how robust our bidding language is. The main point though, is that having more information certainly cannot decrease the quality of calculated allocations.

This property is better illustrated within MAJIC. We mentioned above that bids in MAJIC also accept a parameter search engine. Not only does this information help the agents operate under imperfect information, but also such a tool empowers agents to more expressively describe their preferences. Computationally, this expressiveness is limited to the complexity of the parameter space. In complicated parameter spaces, MAJIC leaves it up to the individual agents to maintain computational feasibility by supplying as good an estimate of a mapping to their utilities as they can. Similarly, Wellman [12] also allows for parameterization of preferences through the submission of a matching algorithm by an agent to the market.

We said that due to computational considerations, we must often settle for a sub-optimal result. In doing so, we must have particular interests in the result's quality. This leads us to our next fundamental property:

(6) The approach should guarantee convergence to a quality solution.

In this case, we must firstly be concerned that we even get any result, and secondly that the solution is one of high quality. Bredin obtains Nash equilibrium in his produced schedules but does so assuming perfect information, so he satisfies property 6 but at the expense of property 4. Also, it would be impossible to expect to find equilibrium for all cases in more complex markets than Bredin's. His approach is for a single good; it has been shown that in single good auctions, there always exists equilibrium. However, in an auction with multi-goods such as the parallel task example mentioned above, it is possible that no equilibrium exists [12].[2] We therefore must settle for some other degree of acceptability in order to still be computationally feasible.

Unfortunately, although auctions with multi-goods are realistic, a comprehensive understanding of their properties is still lacking. Reeves [6] studies solution quality within simultaneous multi-good auctions. His work is not specifically dealing with CPU allocation, but his findings are still relevant. In his work, a different auction is run for each scheduled time slot on a given resource. Auctions halt and clear the goods once an amount of time passes without a new admissible bid for any of the auctions[3]. In his study, he focuses on markets that possess complementaries since they are often the cause of scenarios where no equilibrium exists. As we have seen, complementaries tend to be quite common in real markets, so understanding the effects is of particular importance.

Reeves finds that agents in multi-good markets containing complementaries are particularly at risk of the so-called exposure problem. This problem is when an agent desires a particular combination of goods but it must expose itself to the risk that it very well could get caught with an incomplete set of these goods. His study of potential

[1] Complementaries cannot arise in a single-unit scheduling problem [12].

[2] If agents have additive preferences for completing multiple single-unit jobs, then equilibrium exists [12].

[3] Quiescence is often the term used for this idea of reaching a steady state where new bids have ceased to continue arriving.

equilibrium in such markets takes in him on a journey of various agent strategies. For myopic best response strategy, the research finds that it "is not even approximately optimal" even if all bidders are having this same strategy. He tries to continue using evolutionary strategy search techniques but finds that the strategy space is simply too huge to produce many findings. He concludes by saying that despite the lack of findings, simultaneous markets are still decentralized and therefore a realistic analogy to the way the world really works; in this way, a deeper understanding of multi-good auctions will certainly be needed eventually. Current gaps in knowledge tell us that the satisfaction of property 6 is not yet quantifiable but instead should be made by a fair judgment.

Reeve's market assumptions in his experiments were quite robust. Results in some approaches simply are not. This leads us to our final property:

(7) Simulations of an approach should be analogous to realistic conditions.

In the POPCORN approach, it is claimed that when running their clearinghouse double auction in an online fashion, its allocations are c-competitive with the offline optimal allocations. At first, this appears to score some points in favor of property 6. However, upon further investigation, it becomes apparent that POPCORN's evaluations may not be fully justified. Their observations were based off of models of simulated buyers and sellers whose arrival was governed by a constant Poisson process. On the Internet, arrival and departure would not be so regular. The setup assumes each buyer only has a single computational chunk to run while each seller only has the desire to run a single chunk before they leave the market. We have seen that single good markets yield nicer results. Single-good solutions also take less computation to obtain, so making such assumptions would not only be unrealistic but they could increase the perceived overall performance.

The simulation also makes assumptions about its sellers' homogeneity in that all hosts have the same computational power. This simplifies the sort of work the clearing function has to perform since it essentially has fewer criteria it needs to match. Obviously, this is an unrealistic assumption as well because computational power from hosts to hosts can vary drastically.

Another assumption made is with regards to communication overhead. Relaying information across the Internet is known to take considerable latency. Messages could even be lost and require retransmission. In their experiments, the simulated buyers are not performing over the Internet but instead within a single computer simulation. As a result, applicable communication overhead was not modeled, and it is not realized to what extent how much of a bottleneck this centralized approach is.

To more convincingly satisfy property 7, POPCORN could have instead followed a few of the steps of Reeves. Reeves uses a multi-good market with complementaries. POPCORN could also simulate message latency and heterogeneous sellers as well. Of course, this would be more complex, but also this would be a more realistic demonstration so claims like being c-competitive could be taken more seriously.

5. Conclusion

In this paper, we have listed and argued for the need of some particular fundamental properties in market-based approaches to CPU scheduling across the Internet. We have illustrated how they could be applied to existing implementations in order to result in a more realistic end-product for CPU-time scheduling. Having such products in place, the desire of accessing additional computational power across the Internet could then become more of a reality.

References

[1] Bredin, Jonathan et al. A Game-Theoretic Formulation of Multi-Agent Resource Allocation. In *Proceedings of the 2000 International Conference on Autonomous Agents*, Barcelona, Spain, June 2000.

[2] El-Yaniv, R. and A. Borodin. On-Line Computation and Competitive Analysis. Cambridge University Press, 1998.

[3] P. Golle and I. Mironov. Uncheatable Distributed Computations. In *Proceedings in RSA Conference* 2001.

[4] Jensen, Christian and Daniel Hagimont. Protection wrappers: a simple and portable sandbox for untrusted applications. In *Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications*, 104-110, 1998.

[5] L. Levy, L. Blumrosen, and N. Nisan. OnLine Markets for Distributed Object Services: the MAJIC system. In *Proceedings in SITS 2001*.

[6] DM Reeves, MP Wellman, JK MacKie-Mason, and A Osepayshvili. Exploring bidding strategies for market-based scheduling. to appear, *Decision Support Systems*.

[7] O. Regev and N. Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of First International Conference on Information and Computation Economies*, pages 148-157, Chaleston, SC, Oct. 1998. ACM Press.

[8] Sunderland, I.E. A futures market in computational time. *Communications of the ACM*, 11(6):449-451, June 1968.

[9] Waldspurger, C. A. et al. Spawn: A Distributed Computational Economy. *IEEE Trans. Software Engineering* 18: 103-117, 1992.

[10] WE Walsh and MP Wellman. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. Journal of Artificial Intelligence Research, 19:513–567, 2003.

[11] William E. Walsh and Michael P. Wellman. A market protocol for decentralized task allocation. In *Third International Conference on Multi-Agent Systems*, 325-332, 1998.

[12] MP Wellman, WE Walsh, PR Wurman, and JK MacKie-Mason. Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior* 35:271-303, 2001.