

The OCEAN Computational Market and its Standardized XML Documents

Mark J. Tobias
University of Florida
mjtobias@cise.ufl.edu

Michael P. Frank
University of Florida
mpf@cise.ufl.edu

Abstract

Recent technological innovations and falling hardware prices have contributed to the emergence of large-scale computational economies. At no other time in human history have so many had access to such powerful computing resources. And yet many of these resources lie idle, possibly running a screen saver or not running at all. The OCEAN [10] system, currently in development at the University of Florida, is a computational market that harnesses these resources for the economic benefit of its human users. Computational markets are composed of buyers and sellers of CPU time, and the market itself, which facilitates trade. In our development, we have noted that locating a seller(s) for a buyer is a particularly important aspect of the market's operation. We propose that buyers and sellers describe their computations and resources using standardized XML documents. The market can use these documents to locate sellers for buyers, even if the sellers are found in other markets.

1. Introduction

There are currently millions of human users whose computing resources are connected via the Internet. It has been observed by numerous sources [3,5,10,11,13,14,15,16] that these resources are frequently idle. This observation has led to the emergence of various projects that harness these resources for some benefit.

The projects that have received the most attention are the SETI@home project, and the various initiatives led by Distributed.net. Both of these projects are maintained on a strictly voluntary basis. In the case of SETI@home, users donate idle CPU cycles for the analysis of data collected by the Arecibo radio telescope in the search for intelligent extra terrestrial life [18]. Distributed.net has used the donated CPU time of many users to solve computationally intense problems, such as cracking encryption codes [3].

However, there are some recently formed commercial ventures and research projects whose goal is to harness idle CPU cycles for the economic benefit of

their human users. Such projects are referred to as computational markets.

A computational market is a marketplace in which a CPU cycle is the traded commodity. The major components of a computational market are the users, the computational resources associated with them, and the underlying market mechanisms that facilitate trade. Note that the users do not need to be human; in many instances, a user is a programmed agent acting on behalf of a human user [4,8]. In general, human users in computational markets are buyers or sellers. The buyers have a computation that needs to be performed, and the sellers have access to the idle resources that can execute the computation in question. The buyer and seller(s) will agree on some sort of payment mechanism, the program for the computation will be transported to the seller(s), and the computation will be performed. The computational market's role in all of this is to provide an environment in which these interactions may take place.

2. History of Computational Markets

Possibly the first computational market was described by Sutherland [20]. He demonstrated how auction methods were used to allocate computer time to users of the PDP-1 at the Aiken Computation Laboratory at Harvard. The users were allowed to use the computer for some time period. The hours of the day would be divided into regular time slots. Users were assigned different amounts of currency based on their project's importance. Then, the users would submit bids for time slots. For a given time slot, the user who submitted the highest bid would have use of the computer [20]. By using the auction methods described, a monetary value was associated with computation time, which is a basic feature of computational markets. This paper is important because it may be the first application of economic principles to the problem of computer resource allocation.

Another significant research project regarding computational markets was implemented by Shoch and Hupp [19]. Their paper describes a "worm", characterized by a couple of traits. First, it was parallelizable—it could be broken into many discrete

segments, each running on a different host. Also, the worm was able to sniff out idle machines on a network [19]. Although this project was not concerned with the economics of computation, it still represented a significant step toward fully functional computational markets. This observation lies in the fact that the worm's traits are important features of programs used in computational markets. For one, idle machines in a network must be located. Also, users of programs that can be parallelized stand to benefit most from computational markets, because they achieve the highest utilization of available resources.

The next major research effort was the Agoric Papers [4, 8]. Although there is no implementation associated with these publications, they established important groundwork for modeling distributed computation in economic terms. Although Sutherland's work [20] was important for using economic principles to solve resource allocation problems, it did not address the problem in distributed systems. Drexler and Miller approached this problem by describing auction mechanisms for allocating distributed resources. In one of their solutions, a seller auctions off its resources to multiple competing buyers. Escalator algorithms are used such that buyers only submit an initial bid. Each bid increases at a constant rate until the highest bid is determined. The highest bidder obtains the seller's resources [4]. Drexler and Miller continued, describing initial market strategies, and how they achieve stable pricing mechanisms. Later work [22] has proven Drexler and Miller's algorithms to be effective for achieving this goal. The Agoric Papers made two important contributions to computational markets. First, they provided market-based mechanisms to allocate distributed resources. And second, they addressed how stable pricing mechanisms can be achieved in computational markets. These are essential for a successful computational economy.

The Spawn project [22] claimed to be the first implementation of a market-based computational system. Indeed, its implementation incorporated the most fundamental aspects of a computational market. Spawn's principal features involved an approach to computational resource management based on economic principles, and the exploitation of idle time. This included assigning priorities based on monetary funding units, and the use of price information to control adaptive expansion [22]. The resource allocation methods, specifically processor scheduling, were largely based on Drexler and Miller's work. Spawn's methods to harness idle processors are similar to those employed by the Condor system [2]. Perhaps the most important use of Spawn was in examining the price dynamics of a computational economy. This is relevant because the market must exhibit stable pricing mechanisms. And in fact, the experiments carried out with the system confirm that

computational economies that exhibit stable pricing and growth properties can be designed.

With the advent of the Java programming language, some of the most difficult aspects of computational markets were eliminated. Java makes the problem of migrating executable programs between platforms trivial. Of course, this does not benefit the programs written in other non-portable languages.

The Internet and World Wide Web make it possible for potentially millions of human users to participate in computational markets. With this dramatic increase in the number of users, the problems of scalability and stable pricing mechanisms are magnified.

The POPCORN [13,16] project was the first computational market to focus on the benefits of using Java programs. The most interesting feature of POPCORN was its notion of the traded commodity. Regev and Nisan [16] noted that the idea of trading in CPU time is not exact since it is processor dependent. Due to the fact that they were concerned with Java programs, they established the basic tradable good as a "JOP", or Java Operation. Each program takes some number of JOP's to execute, and the price of executing a program is proportional to its number of JOP's [16]. This convention made price estimation simpler, but only applied to Java programs.

3. Current Computational Market Implementations

Commercial ventures of note are those initiated by Popular Power [14] and the ProcessTree Network [15]. Both of these endeavors promise some form of economic benefit to their human users. Although it is not yet a commercial venture, OCEAN, currently under development in the Computer & Information Science & Engineering Department of the University of Florida, is another example of such a system.

Both Popular Power and ProcessTree provide a facility for users to download specialized client software. The client software allows the user to become a seller of resources. When the user is connected to the Internet, the client software will download a buyer's computation, executing it when the seller's resources become idle [14,15].

Each of the commercial ventures employs similar mechanisms to disburse payments to sellers. They have noted [14,15] that one work unit does not have a significant cash value associated with it. Therefore payment will accrue over time in an account. When the account reaches a minimal value, payment can be disbursed in the form of a check, or as discounts to various online services. We propose a similar payment mechanism with the OCEAN. However, we believe that users prefer to receive cash rather than discounts. Also,

disseminating many small checks to users is impractical. Therefore we propose that payments be made through an online account with Paypal [12] or a similar service.

4. The OCEAN

OCEAN is an acronym for Open Computation Exchange & Arbitration Network. OCEAN distinguishes itself from other projects in a number of ways. First, the market itself should be able to profit from the transaction that occurs between a buyer and a seller(s). This is done in commercial computational markets, but not research projects. Second, all systems up to this point have employed an auctioning mechanism to allocate resources. However, with OCEAN we propose to use a slightly different mechanism to allow buyers and sellers to agree on a price. We implement a model based on transactions that occur in the stock market. Also, we envision a more democratic approach to allow buyers to use the computational market. The two aforementioned commercial systems allow sellers to sign up easily, but do not provide the same mechanism for buyers. OCEAN provides a truly “open” system—one which can be used by anyone.

4.1 Overview

As mentioned previously, the basic constituents of any computational market are buyers, sellers, and the market itself. OCEAN is no different in this respect.

The major entities that exist in the OCEAN are *buyers*, *sellers*, and *arbitrators*. Buyer and seller assume the same meaning as discussed previously. The arbitrator is the major party responsible for facilitating trade between buyers and sellers.

4.2 Architecture

At the highest level, the OCEAN is a system of buyers and sellers, interacting through the arbitrator(s). Either a buyer or a seller may initiate a trade. A buyer has a computation to be executed and a seller has access to computing resources (Fig 1). The most common resource type is a CPU, but others include memory, network bandwidth and graphics hardware. Once a trade is initiated, the arbitrator finds a suitable match for the initiator.

In a market where large numbers of buyers and sellers interact, the arbitrator is a major performance bottleneck. A solution to this is to distribute the arbitrator’s functionality over many nodes. But buyers and sellers need not be concerned with this. In fact, this does not affect the architecture of the system, only the implementation. Whether the arbitrator is distributed or not, it still serves the same purpose.

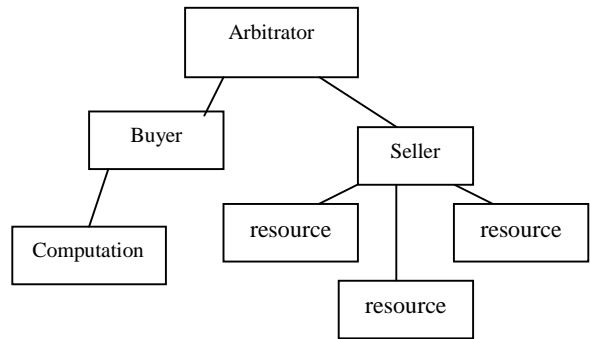


Fig 1: Major participants in the OCEAN

There is also the possibility that buyers and sellers interact with multiple competing markets. In this case there will be an arbitrator that represents each market. Users will consult an arbitrating agent, whose role is to find an arbitrator for an initiating buyer or seller (Fig 2). The agent’s principal function is to serve as a directory of arbitrators. The arbitrating agent must be operated by a non-profit, non-partisan organization to ensure fair competition between markets.

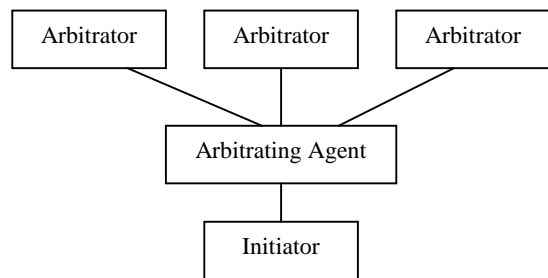


Fig 2: Multiple arbitrators and an arbitrating agent

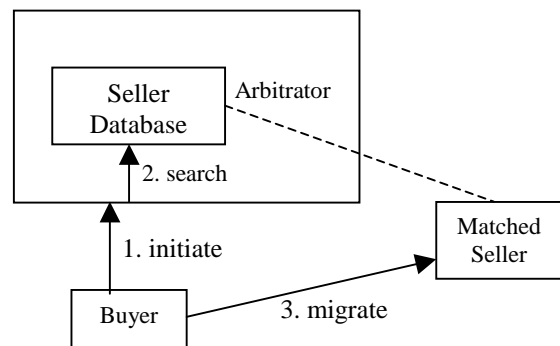


Fig 3: Buyer-initiated Transaction

An arbitrator must have access to databases of buyers and sellers. These databases contain all of the buyers and sellers known to the market. Upon initiation of a trade by a buyer, the seller database will be searched for a seller that can perform the buyer's computation. Similarly, when a seller initiates a trade, the buyer database will be searched for a computation that the seller can perform. In either case, once a buyer is matched with a seller(s), the computation is migrated to the seller(s) (Fig 3). In the event that no match is found, the market may wait for a suitable match, or it may consult another market for help.

4.3 Implementation

The buyer publishes a bid price, or price per unit time. The bid price is the price to complete a whole job. The seller publishes an ask price. This is measured as price per unit time. In markets where only Java programs are executed, ask price can be expressed in price per JOP [16]. The buyer must estimate the time or the number of operations to complete the job. Once one of these is known, the arbitrator has an estimate for the value of the computation, and an estimate for the cost of execution. At this point, the arbitrator can match a buyer and a seller.

But how are buyers and sellers matched? The algorithm we propose is as follows. The buyer offering the highest bid price per standardized unit of resource and the seller with the lowest ask price will agree on a trade. This is fair because these two entities are the most willing to trade. Then match the next-highest bid and the next-lowest ask, and so on. Prices can be very close to continuous to attain the fairest possible scheme.

The bid price should be larger than the ask price. The difference, the bid-ask spread, becomes revenue for the market and must cover the cost of the market's operation. After the market's operations are accounted for, any remaining revenue constitutes a profit for the market.

However, in order for the market to remain competitive, the arbitrator should not take the entire bid-ask spread as revenue. Instead, some fraction of the spread should be taken as revenue, and the rest used to reduce the bid-ask spread before the transaction is made. Thus half of the remaining spread is given back to the buyer and the other half given back to the seller. The arbitrator will set a minimum spread at which it will conduct a transaction, in order to guarantee that its fraction of the spread will always be enough to cover its costs and ensure at least a small profit.

The market participants may want to implement specific strategies to ensure their goals are met. For example, a buyer may need to obtain resources very quickly, and therefore must bid higher than any other buyer to fulfill this requirement. To facilitate this mechanism, the arbitrator must publish the prices at

which trades occur. At the very least, buyers and sellers will have the ability to query for the most recent trading prices.

4.3.1 Anatomy of a Transaction

Fig 4 shows the steps that a transaction in the OCEAN will typically follow.

1. A buyer or seller (initiator) publishes its presence to the market.
2. An arbitrator is found, either automatically, or through an arbitrating agent.
3. The arbitrator gathers price information from the initiator.
4. The arbitrator searches its database(s) for the best match for its initiator.
5. If no match is found, another arbitrator can be consulted, or the initiator will have to wait until its request can be fulfilled.
6. A match is found. The buyer contacts the seller(s). They exchange contracts with digital signatures, if required.
7. The buyer's computation is migrated to the seller's resources, and it is executed.
8. Once the computation is complete, the arbitrator collects funds from the buyer's account, disburses funds to the seller's account, and deposits its share.
9. The results of the transaction are logged to a database.

Fig 4: Anatomy of a Transaction

5. Future Research in the OCEAN

The pairing of buyers with sellers is a fundamental problem in computational markets. Buyers and sellers must agree on many discrepancies before computation can begin. The most important of these is price. If a buyer and a seller cannot reach an agreement on a computation's cost, the seller will not execute the buyer's computation. In addition, security is a major concern in computational markets. If a buyer has potentially sensitive data associated with its computation, the seller(s) must ensure that the data's integrity will not be compromised. Security in computational markets is an involved topic, and deserves much future work. Another issue is that of special needs requested on the part of the buyers. If the buyer has a special requirement, fast graphics hardware for example, the seller should make it known that it has adequate resources for such a specialized computation. Also, incompatible platforms are a concern for all markets, excluding those which only support programs written in Java.

It is imperative that the computational market can match buyers and sellers appropriately. There must exist a mechanism whereby buyers fully describe their computations and sellers fully describe the resources at their disposal. By “fully describe,” we mean that all pertinent information is present. Some information the buyers must supply is bid price, required security measures, special resource requirements, platform dependencies, and perhaps an estimate of the computation’s complexity. The sellers, in turn, must supply information such as ask price, security measures supported, any special hardware available, various hardware metrics, and the supported platform(s). Certainly there are many other types of information that should be supplied as well. For example, the seller should indicate its available network bandwidth. Alternatively, the arbitrator could independently assess this. In cases where a buyer has massive stores of input data, a seller with low network bandwidth would be unsuitable due to unacceptable data transfer delay. Another piece of useful information would be whether the buyer’s computation is parallel. In this case, the market could find many sellers for one buyer. Parallel jobs executed in computational markets are another complex area; much future research can be devoted to this topic.

We propose that buyers and sellers include standardized XML documents that fully describe their requirements and resources, respectively. When a buyer wishes to consult the market to have a computation performed, it must send the market the document associated with its computation. When a seller registers with the market, it supplies a document for each machine that will be available to the market. It is then up to the market to use these documents to appropriately match a buyer with a seller(s). Possible examples of XML documents for buyers and sellers are given in Fig 5 & 6, respectively.

```
<?xml version="1.0">
<buyer ID="98765">
  <location>
    http://www.cise.ufl.edu/~mjtobias/comp.exe
  </location>
  <platform>Windows_x86</platform>
  <complexity units="JOPS_per_second">
    100000
  </complexity>
  <security_needed>
    SSL
  </security_needed>
  <network_needed>
    28.8k
  </network_needed>
</buyer>
```

Fig 5: XML Document for a Buyer

```
<?xml version="1.0">
<seller ID="12345">
  <location>209.26.232.140</location>
  <platform>Windows_x86</platform>
  <resource_list>
    <resource>
      <CPU clock_speed="500mhz">
        </CPU>
      </resource>
    <resource>
      <network_connection>
        T1
      </network_connection>
    </resource>
  </resource_list>
  <security_supported>
    SSL
  </security_supported>
</seller>
```

Fig 6: XML Document for a Seller

Why use XML documents to describe computations and computational resources? XML offers some distinct advantages. First of all, XML documents are plain text, and thus are platform independent. Any platform that can read ASCII text (and that is almost certainly all of them) has the ability to read an XML document. This is critical for computational markets because many different computing platforms may be used in a given market. Also, XML’s data modeling capabilities can be very useful for computational markets. XML provides the means to describe a domain specific vocabulary, along with a Document Type Definition for that application domain. This is discussed in more detail in the following paragraphs.

How is a suitable seller(s) located for a given buyer? This is where an XML vocabulary for computational markets is useful. The vocabulary can be used to construct a database schema for available sellers. When a buyer makes a request to the market to perform a computation, the market will use the buyer’s associated XML document to search a database for suitable sellers.

However, what if the market cannot find a suitable seller for a buyer? Should the buyer be forced to wait for a suitable seller to matriculate? One way this problem could be solved is for the buyer to consult other existing markets to see if they have suitable sellers. Or, when the market determines that no suitable seller exists, that market may consult other markets to determine if they have any suitable sellers. In the latter case, perhaps the first market must charge the buyer a “finder’s fee”. So, there is the possibility that 1) the buyer consults different markets and 2) markets consult other markets.

This leads to the necessity for standardized Document Type Definitions (DTD’s) for all computational markets. The DTD’s will rigorously define the structure of a legal document. Realistically,

there will need to be at least two DTD's: one for buyers and one for sellers. Other possible DTD's include those devoted strictly to security issues, parallel computations, and the contract between a buyer and seller(s). As mentioned before, each document must fully describe any computation or computational resource. This means that any market, not just a specific market, can use the XML documents of a buyer or seller for their own benefit. At this point, it is useful to determine the advantages of having standardized DTD's for all computational markets. Ultimately, it will benefit all parties involved. Both buyers and sellers will be able to draw from a larger pool of users. A buyer may find a more suitable seller in another market, and vice versa. A single standard document for the buyer or seller requires the least amount of work on their part—relevant information must only be gathered once. Also, different markets consulting each other can be mutually beneficial. The referring market may be able to collect a finder's fee, while the referred market benefits from matching a buyer and a seller, as normal. Or, the two markets may split the profits associated with that type of transaction.

The generation of documents should require minimal effort on the part of human users. There will be applications that generate these documents with little or no intervention from human users. For a seller, the application can be something that gathers all relevant hardware information, generates a document, and sends it to the arbitrator. For a buyer, an application could gather most of the information from an executable program. However, to indicate special needs, the user may have to input information to OCEAN, perhaps from some online form. Then a document can be generated from all information gathered.

It is our belief that forming standardized XML Document Type Definitions for buyers and sellers will be a significant contribution to the area of computational markets. The autonomous existence of many markets does not provide the greatest benefit to human users, or even to the markets themselves. By standardizing the languages that describe buyers and sellers, it will be possible for all markets and users to benefit. Only in this way can a computational market truly be "open".

Certainly existing markets have their own sophisticated database schemas in place to find sellers for buyers, or vice versa. However, this does not mean the DTD's are of no use to them. The markets can transform or parse a standard XML document originating from another market or user so that all of the information about a computation or a computational resource is available in a format that is useful to them. Conversely, a market can generate outgoing XML documents from information it keeps about buyers and sellers to consult another market or user. With this in mind, one might think of the DTD's as a sort of communication protocol for computational

markets. The DTD defines the structure of a message, and the message content is the document describing a computation or a resource.

References

- [1] Andersson, Arne and Fredrik Ygge, "Managing Large Scale Computational Markets", *Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS'98)*, IEEE Computer Society, 1998.
- [2] The Condor homepage, <http://www.cs.wisc.edu/condor>.
- [3] The Distributed.net homepage, <http://www.distributed.net>.
- [4] Drexler, K. E. and Mark S. Miller, "Incentive Engineering: for Computational Resource Management", from *The Ecology of Computation*, Bernardo Huberman (ed.) Elsevier Science Publishers, North-Holland, 1988.
- [5] Hayes, Brian, "Collective Wisdom", *American Scientist*, vol. 86, no. 2, March-April, 1998, pp. 118-122.
- [6] The LEGION homepage, <http://www.cs.virginia.edu/~legion>.
- [7] Martin, Didier, et al, *Professional XML*, Wrox Press, Birmingham, UK, 2000.
- [8] Miller, Mark S. and K. E. Drexler, "Markets and Computation: Agoric Open Systems", from *The Ecology of Computation*, Bernardo Huberman (ed.) Elsevier Science Publishers, North-Holland, 1988.
- [9] The New York Institute of Finance, *Guide to Investing*, New York Institute of Finance, New York, 1992.
- [10] The OCEAN homepage, <http://www.cise.ufl.edu/~mpf/ocean>.
- [11] Patrizio, Andy. "Discover Distributed Computing", *Byte.com*, Sep 1, 1999.
- [12] The Paypal homepage, <http://www.paypal.com>.
- [13] The POPCORN homepage, <http://www.cs.huji.ac.il/~popcorn>.
- [14] The Popular Power homepage, <http://www.popularpower.com>.
- [15] The ProcessTree Network homepage, <http://www.processtree.com>.
- [16] Regev, Ori and Noam Nisan, "The POPCORN Market—an Online Market for Computational Resources", *ICE '98, Proceedings of the first International Conference on Information and Computation Economies*, October 25-28, Charleston, SC, USA.

[17] Rheingold, Howard, "You Got the Power", *Wired Magazine*, archive 8.08, August 2000.

[18] The SETI@home homepage, <http://setiathome.ssl.berkeley.edu>.

[19] Shoch, John F. and Jon A. Hupp, "The 'Worm' Programs—Early Experience with a Distributed Computation", *Communications of the ACM*, vol. 25 no. 3, March 1982, pp. 172-180.

[20] Sutherland, I. E. "A Futures Market in Computer Time", *Communications of the ACM*, vol. 11 no. 6, June 1968, pp. 449-451.

[21] Vanhelsuwe, Laurence, "Create Your Own Supercomputer With Java", *JavaWorld*, January 1997.

[22] Waldspurger, Carl A., et al, "Spawn: A Distributed Computational Economy", *IEEE Transactions on Software Engineering*, vol. 18, no. 2, Feb 1992, pp. 103-117.