Storage Exchange: A Global Trading Platform for Storage Services

Martin Placek and Rajkumar Buyya

Grid Computing and Distributed Systems Laboratory and
NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{mplac,raj}@csse.unimelb.edu.au

Abstract. The Storage Exchange (SX) is a new platform allowing storage to be treated as a tradeable resource. Organisations with varying storage requirements can use the SX platform to trade and exchange storage services. Organisations have the ability to federate their storage, be-it dedicated or scavenged and advertise it to a global storage market. In this paper we discuss the high level architecture employed by our platform and investigate a sealed Double Auction market model. We implement and experiment the following clearing algorithms: maximise surplus, optimise utilisation and an efficient combination of both.

1 Introduction

The Internet has proven to be a source of many exciting wide-area distributed computing applications, enabling its users to share and exchange resources across geographic boundaries. It is in this context we introduce the Storage Exchange (SX). Consumers and providers are able to submit their storage requirements and services along with budgetary constraints to the SX, which in turn employs a market model to determine successful trades. The motivation and long term goal behind our research and development of the SX platform has been to achieve Autonomic [1] management of storage. We envisage *Consumers* and *Providers* will employ brokers which may purchase or sell storage in an autonomic manner based on the organisations requirements.

The SX platform can be used in a collaborative manner, where participants use the model to exchange services for credits, or alternatively in an open marketplace where enterprises trade storage services. Whether in a collaborative or enterprise environment the incentives for an organisation to use our SX platform include: (i) *monetary gain:* Institutions providing storage services (*Providers*) are able to better utilise existing storage infrastructure in exchange for monetary gain. Institutions consuming these storage services (*Consumers*) have the ability to negotiate for storage services as they require them, without needing to incur the costs associated with purchasing and maintaining storage hardware. (ii) *common objectives:* There may be organisations which may wish to exchange

storage services as they may have a mutual goal such as preservation of information [2]. (iii) *Spikes in Storage Requirements:* Research organisations may require temporarily access to mass storage [3] (e.g. temporarily store data generated from experiments) and in exchange may provide access to their storage services. (iv) *donate:* Institutions may wish to donate storage services, particularly if these services are going to a noble cause.

There are many considerations which need to be made when building a global scale platform such as the SX: security, high-availability, fault tolerance, reputation, monetary issues, consistency, operating environment, just to name a few. We have chosen to focus our efforts on the core components by proposing and developing a platform upon which we are able to develop a market model and begin the work necessary to realise the Storage Exchange.

2 Related Work

Applying economic models to manage computational resources has been the focus of much recent research [4–6]. These papers discuss the application of economic principles to manage the scheduling of jobs in a large scale environment such as the Grid [7]. Examples of different economic models and systems which use them include: (i) Commodity Market model (Mungi [8] and NimrodG [9]), (ii) Posted price model (NimrodG [9]), (iii) Auction model (Spawn [10] and Popcorn [11]) (iv) Barter model (Stanford Archival Repository Project [2], and MojoNation [12]).

FreeLoader [3] aggregates unused desktop storage to provide low-cost solution to storing massive datasets. Its specifically designed for research institutions which need to store large scientific datasets. This scenario is particularly useful for scientists engaged in high performance computing, where handling large datasets is common. FreeLoader aims to handle large immutable files (write-once-read-many). Farsite [13] is another system which demonstrates the resource potential to be gained from scavenging unused storage. Farsite operates within the boundaries of an institution providing a storage service logically similar to a file-server found in corporate environments.

Cooper et al [2] propose a bartering storage system for preserving information. Institutions which have common requirements and storage infrastructure can use the framework to barter with each other for storage services. The bartering model relies on their to be a *double coincidence of wants* [14]. OceanStore [15] is a globally scalable storage utility, providing paying users with a durable, highly available storage service by utilising untrusted infrastructure. Mungi [8] is Single-address-space operating system which employs economic principles to manage storage quota. MojoNation [12] uses digital currency (*Mojo*) to encourage users to share resources on its network, users which contribute are rewarded with *Mojo* which can be redeemed for services.

Freeloader [3] and Farsite [13] both demonstrate the storage potential that exists by scavenging storage from workstations. The following works [8–10, 12] apply economic principles to effectively manage and foster the trade and ex-

change of services. The Storage Exchange aims to combine storage scavenging and economic principles to create a global platform allowing institutions to federate, trade, exchange and manage storage services.

3 System Overview

There are four main components which make up the SX platform, the Storage Client, Storage Broker, Storage Provider and the Storage Exchange itself (Figure 1). The SX platform has been designed to operate on global network such as the Internet, allowing organisations across geographic boundaries to trade and utilise storage services. Organisations have the ability to trade storage based on their current requirements, if a Storage Broker detects an organisation is running low on storage it may purchase storage, alternatively if it finds that there is an abundance of storage it has the ability to lease out the excess storage. The rest of this section discusses each of the components:

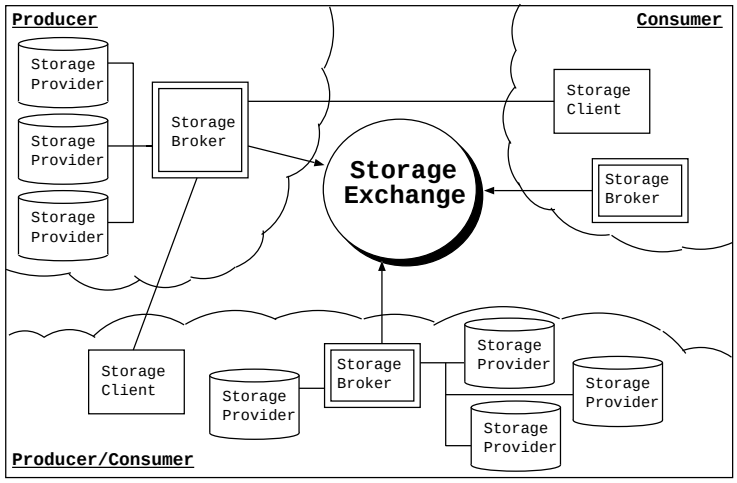

Fig. 1. Storage Exchange: Platform Architecture

Storage Provider: The Storage Provider is deployed on hosts within an organisation chosen to contribute their available storage. Whilst we envision the Storage Provider to be used to scavenge available storage from workstations, there is no reason why it can not be installed on servers or dedicated hosts. The Storage Provider is responsible for keeping the organisations broker up to date with various usage statistics and service incoming storage requests from Storage Clients.

Storage Client: An organisation wishing to utilise a negotiated storage contract will need to use a Storage Client. A user will configure the Storage Client

with the storage contract details. The Storage Client then uses these details to authenticate itself with the provider's Storage Broker and upon successful authentication the Storage Client requests a mount for the volume. The provider's Storage Broker then looks up the Storage Providers responsible for servicing the storage contract and instructs them to connect to the Storage Client. Upon receiving a successful connection from the Storage Provider, the Storage Client provides an interface to the user (e.g. local mount point).

Storage Broker: For an organisation to be able to participate in the SX platform they will need to use a Storage Broker. The Storage Broker enables the organisation to trade and utilise storage services from other organisations. The Storage Broker needs to be configured to reflect how it should best serve the organisations interests. From a consumer's perspective the Storage Broker will need to know the organisations storage requirements and the budget it is allowed to spend in the process of acquiring them. From the Provider's perspective the Storage Broker needs to be aware of the available storage and the financial goals it is required to reach. Upon configuration, a Storage Broker will contact the Storage Exchange (SX) with its requirements.

Storage Exchange (SX): The Storage Exchange component provides a platform for Storage Brokers to advertise their storage services and requirements. The SX is a trusted entity responsible for executing a market model and determining how storage services are traded. When requests for storage are allocated to available storage services the Storage Exchange generates a storage contract. The storage contract contains a configuration of the storage policy forming a contract binding the provider to fulfill the service at the determined price. In a situation where either the provider or consumer breaches a storage contract, the SX will keep a record of reputation for each organisation which can be used to influence future trade allocations.

4 Trading Storage

This section covers topics key to making trading storage possible and begins by covering storage policies; which provide a way to quantify storage being traded. Followed by a discussion on the Double Auction (DA) market model and clearing algorithms we have investigated.

Storage Policy: Storage policies provide a way to quantify a storage service, this is essential regardless of chosen market model. Systems such as the one proposed in [16] use Storage Policies as a way to specify high-level Quality of Service (QoS) attributes, effectively abstracting away error prone low-level configurables from the administrator. Our use of Storage Policies allow Storage Brokers to quantify the service which they wish to lease out or acquire. When a trade is determined the storage policy will form the basis for a storage contract containing details of SLA (Service Level Agreement). The attributes which make up a storage policy are as follows:

1. **Storage Service Attributes:**
   (a) **Capacity**(C)**:** Storage Capacity (GB) of volume.
   (b) **Upload Rate** (U)**:** Rate (kb/sec) of transfer to the volume.
   (c) **Download Rate** (D)**:** Rate (kb/sec) of transfer from the volume.
2. **Duration:**
   (a) **Time Frame** (T)**:** Lifetime (sec) of storage policy.

Market Model: Decades of research and experiments [17–20] show that Double Auctions (DA) are effective and efficient market model. DAs have been shown to quickly converge towards a Competitive Equilibrium (CE). The CE is the intersection point of true demand and supply curves, yielding allocations which are near 100% efficient. From an economic stand point DAs are a sound and efficient market model. In a Double Auction (DA) [18] both buyers (Consumers) and sellers (Providers) may submit offers to buy and sell respectively. Providers and Consumers submit asks and bids simultaneously and hence participate in a *Double*-sided auction. The process of clearing determines the way in which trades are allocated amongst the asks and bids. There are two ways in which clearing may take place, continuously or periodically. Double Auctions cleared continuously are refered to as Continuous Double Auctions (CDA) and compatible bids and asks are cleared instantaneously. The New York Stock Exchange (NYSE) and Chicago Commodities market both employ a CDA market model. Double Auctions may also be cleared periodically, these are refered to as Clearinghouse (CH) or Call Markets. Bids and Asks are submitted sealed to a clearinghouse, which periodically processes the queued up bids and asks to determine a market clearing price. Call Markets are used to determine opening prices in continuous markets such as the NYSE.

As well as being economically sound there are two attractive features of Double auctions which come to our attention, (i) many trades can be cleared in an instant and using a sealed model (ii) the need to continuously broadcast the current market status to all participants is removed. Studies comparing Double Auctions [21, 22] with other auction protocols (Dutch, English, First Price Sealed bid) found that Double Auctions possess least communication overhead. These remarkable properties have motivated our research and subsequent application of a DA market model in our SX platform. The Storage Exchange is responsible for executing a Clearinghouse variation of the DA model, which involves accepting sealed offers from provider and consumer brokers and periodically allocating trades amongst the queued up offers using a clearing algorithm. Consumers submitting bids do so in the form of Storage Request Bids (SRB). A SRB consists of a Storage Policy detailing the storage service and a bid price $SRB = (C, U, D, T, \$)$. A Provider submits a Storage Service Ask (SSA) representing the storage service they wish to lease out. An SSA consists of a Storage Policy representing the storage service they are selling, along with a cost function $SSA = (C, U, D, T, CostFunction(C, U, D, T))$. The cost function represents the Providers responsible and determines a cost based on Storage Policy attributes. The Storage Exchange uses the cost function to determine how much

a Consumers would need to pay based on their Storage Policy. To achieve this the Storage Exchange substitutes consumers Storage Policy attributes into the Providers cost function to determine a price.

Clearing Algorithms: Periodically the Storage Exchange allocates trades amongst queued up SRBs with SSAs, the manner in which it does so is determined by the clearing algorithm it employs. We propose and investigate the following clearing algorithms in the context of our SX platform:

1. *First fit:* SRBs are allocated to SSAs on a first fit basis. An SSA is deemed to fit if it has the storage resources required by the SSA and the cost function returns a price within the SSA bid amount. SRBs are processed in the order which they have been queued up.
2. *Maximise Surplus:* This clearing algorithm aims to maximise the profit of the auction. An SRB is allocated to an SSA which results the maximum difference between Consumers bid price and result of Providers cost function.
3. *Optimise Utilisation:* This algorithm focuses on achieving better utilisation by trying to minimize the *left overs* that remain after an SRB is allocated to an SSA. A measure of fit is calculated (Algorithm 1) between an SRB and each SSA. A large measure of fit indicates that the remaining ratios have a large spread amongst each of the *Storage Service Attributes* and therefore would result in an SSA with potentially more waste, whereas a small population variance would indicate that the remaining *Storage Service Attributes* within the SSA would have less waste. Upon calculating a measure of fit between the considering SRB and each SSA, we allocate it to the SSA which returned the smallest measure of fit. SRBs are processed in the order which they have been queued up.

Algorithm 1 MeasureOfFit(S,A)

1: **Input**: Storage Request Bid S, Storage Service Ask A
2: **Output**: Measure of Fit F
3: $A = \{a_1, a_2, ..., a_n\}$//*Storage Service Attributes*
4: //belonging to Available Storage Policy
5: $S = \{s_1, s_2, ..., s_n\}$//*Storage Service Attributes* belonging to Storage Request
6: // calculate a remaining ratio for each of Storage Service Attributes
7: $R = \{r_1 = \frac{a_1 - s_1}{a_1}, r_2 = \frac{a_2 - s_2}{a_2}, ..., r_n = \frac{a_n - s_n}{a_n}\}$
8: // calculate the population variance amongst the remaining ratios
9: $F = \frac{1}{n}\sum_{i=1}^{n}(r_i - u_R)^2, where\ \ u_R = \frac{1}{n}\sum_{i=1}^{n} r_i$

4. *Max-Surplus/Optimise Utilisation:* This clearing algorithm (Algorithm 2 incorporates the last two allocation strategies and aims to draw a balance between the two. Parameter (k) serves to bias the balance, ($0.5 < k <= 1$) means that importance will be given to utilisation, whereas a $k(0 <= k < 0.5)$ will give importance to achieving a better surplus. Algorithm 2 is applied to every SRB, in the order which they have been queued up.

Algorithm 2 Max-Surplus/Optimise Utilisation Algorithm

Input: Storage Request Bid S, Storage Service Asks A, Balance k
Output: Selected Storage Policy P
$F \leftarrow \{\emptyset\}$ // a set to store MeasureOfFit values
$M \leftarrow \{\emptyset\}$ // a set to store Surplus calculations
for all $availableStoragePolicy \in A$ **do**
  if $availableStoragePolicy$ has greater resource attributes than S and S bid price is greater than $availableStoragePolicy$ reserve **then**
    $F \leftarrow F \cup \text{MeasureOfFit}(S, availableStoragePolicy)$
    $M \leftarrow M \cup \text{surplus}(S, availableStoragePolicy)$
  end if
end for
$minSurplus = \min(M)$, $worseFit = \max(F)$
$deltaMeasureFit = worseFit - \min(F)$, $deltaSurplus = \max(M) - minSurplus$
$currentHighScore =$ Large Negative Number
for all $availStorePl \in A$ **do**
  $ratioBetterFit = (worseFit - \text{MeasureOfFit}(S, availStorePl))/deltaMeasureFit$
  $ratioBetterSurplus = (\text{surplus}(S, availStorePl) - minSurplus)/deltaSurplus$
  $score = k * ratioBetterFit + (1 - k) * ratioBetterSurplus$
  if $score > currentHighScore$ **then**
    $currentHighScore = score$
    $P \leftarrow \{availStorePl\}$ // assign Storage Policy with max score
  end if
end for

5 Performance and Evaluation

Implementation: The Storage Provider and Storage Client components have been written in C. The Storage Client utilises the FUSE library [23] to provide a local mount point of the storage volume in user space. The Storage Broker and Storage Exchange have both been written in Java. Interactions between the Broker, Provider and Client have been implemented and tested. We have been able to successfully mount a replicated storage volume utilising scavenged storage made available by Providers. Communication between components is carried out via TCP socket communication. The Storage Exchange accepts offers from Storage Brokers and employs a clearing algorithm to allocate trades. Our performance evaluation focuses on the Storage Exchange and comparing the different clearing algorithms it employs.

Experiment Setup: We randomly generate a series of bids (SRB) and asks (SSA) which comply to the posting protocol used by Consumers and Providers. The cost functions in the SSAs are linear. The parameters we have used to generate our random set of offers are outlined in Table 1. Each experiment executed represents a single clearing period, that is assume the set of bids and asks generated were queued up over some period of time by the Storage Exchange, our experiment focuses on the sole process of clearing at the end of that period. With every experiment the same set of orders are loaded in the same order in the

Storage Exchange to ensure each clearing algorithm is executed in exactly the same manner. Parameters with ranges are assigned with a randomly generated numbers within the specified range. Whilst our scenario has many more bids (600) than asks (50), the asks contain much larger storage service attributes, which would imply that Providers have a large quantity of storage they wish to sell to many consumers.

Parameter	*Description*	*Values*
SRB	Number of Storage Request Bids	600
SRC_{range}	Storage Request Capacity range (GB)	5 - 50
SRU_{range}	Storage Request Up Rate range (kb/sec)	5 - 50
SRD_{range}	Storage Request Down Rate range (kb/sec)	5 - 50
$SRDU$	Storage Request Duration (sec)	20000
SSA	Number of Storage Service Asks	50
SAC_{range}	Storage Ask Capacity range (GB)	50 - 500
SAU_{range}	Storage Ask Up Rate range (kb/sec)	100 - 1000
SAD_{range}	Storage Ask Down Rate range (kb/sec)	100 - 1000
$SADU$	Storage Ask Duration (sec)	20000

Table 1. Experiment Parameters

Results: Our experiment results have been broken down into four plots. The first two plots (Figure 2 and 3) focus on budget aspects while the second set of plots (Figure 4 and 5) focus on utilisation achieved. The horizontal axis in all the plots represents the number of bids that have been processed. We can see from the Auction Surplus plot that the *Maximise Surplus* algorithm achieves far better surplus than either *first fit* or *Optimise Utilisation*, but performs poorly in utilisation plots (Figure 4 and 5) which in turn has a bad impact on *Ask budget met*. The *Optimise Utilisation* algorithm achieves a far better utilisation (Figure 4 and 5) than *Maximise Surplus*, so much so it achieves the best in percentage of Ask budget met. Even though it performs well in utilisation it achieves a poor result in auction surplus. Finally when we apply *Max-Surplus/Optimise Utilisation* clearing algorithm we are able to achieve best Auction Surplus ($k = 0.75$) whilst achieving better utilisation than *Maximise Surplus*.

6 Conclusion

The SX platform provides organisations with various storage services and requirements the capability to trade and exchange these services. Our platform aims to federate storage services, allowing organisations to find storage services which better meet their requirements whilst better utilising their available infrastructure. Organisations are able to scavenge storage services across their network of workstations and with the use of the SX platform lease it out globally. The Storage Exchange serves as a foundation for further research and develop-

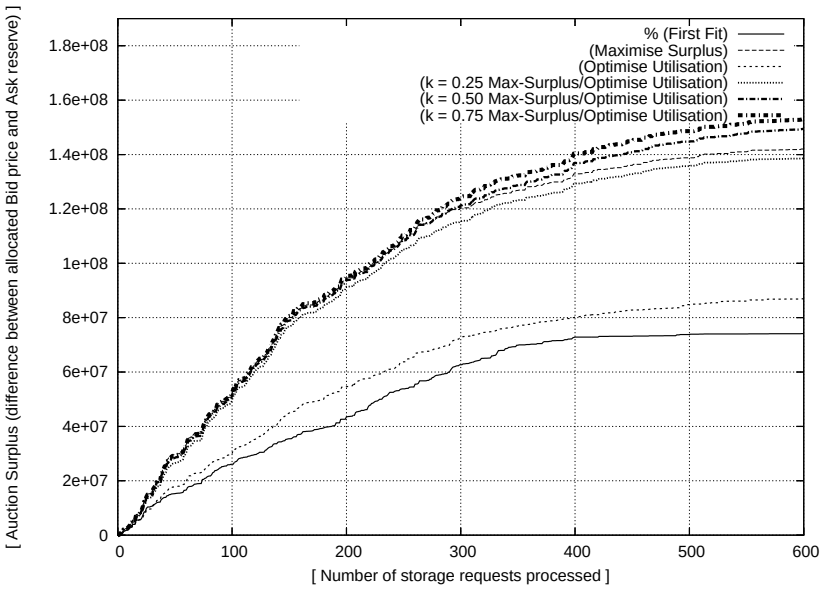

Fig. 2. Results: Auction Surplus

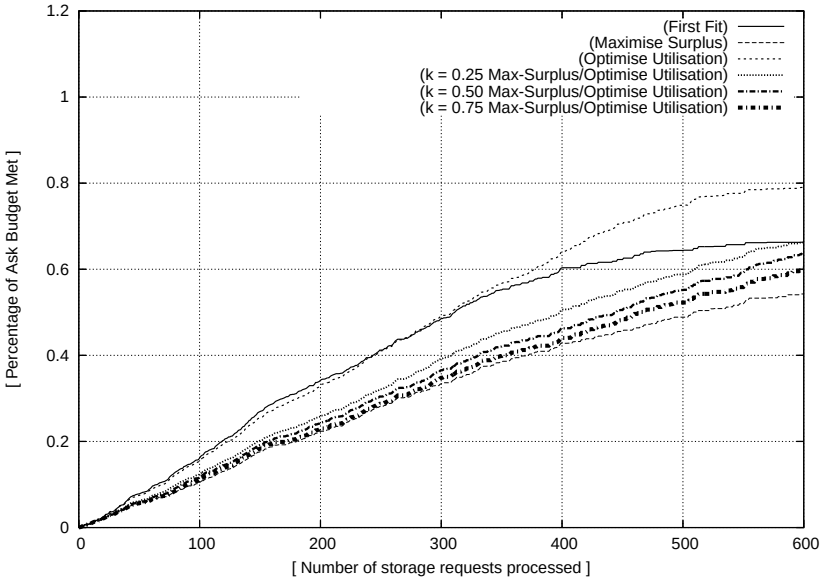

Fig. 3. Results: Percentage of Ask Budget met

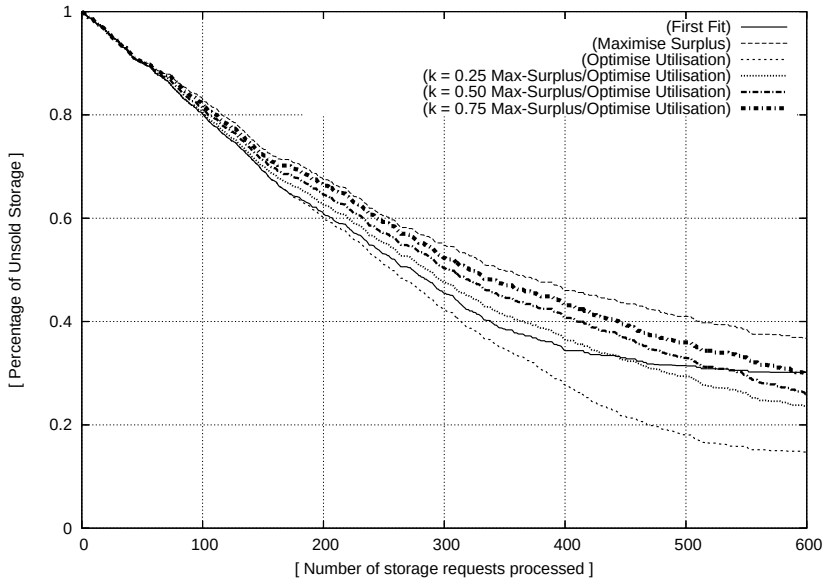

Fig. 4. Results: Percentage of Unsold Storage

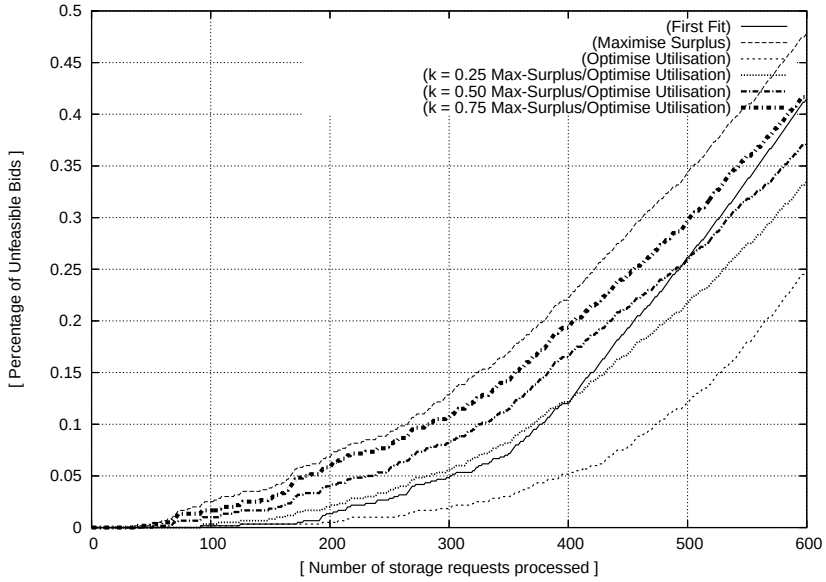

Fig. 5. Results: Percentage of Unfeasible Bids

ment into utilising economic principles to achieve Autonomic management [24] of storage services.

In this paper we discuss our SX platform and apply a sealed Double Auction market model and evaluate various clearing algorithms which aim to maximise surplus, optimise utilisation and finally combining the previous two. Our results show that combining maximise surplus and optimise utilisation algorithms achieves better utilisation and consequently the best auction surplus. A couple of areas which require further research include:

1. *Determining a clearing price:* Whilst determining a clearing price in a double auction which deals with goods that are homogeneous and divisible abstract entities such as money and shares is found by looking where supply intersects demand, this is not applicable when dealing with heterogeneous goods [25] such as storage policies.
2. *Extending Experiment:* Conduct a more detailed assessment of our clearing aglorithms by varying parameters described in Section 5. Also determining a theoretically optimal clearing result would allow us to compare and guage the efficiency of our clearing algorithms.

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1) (2003) 41–50
2. Cooper, B.F., Garcia-Molina, H.: Peer-to-peer data trading to preserve information. ACM Transactions on Information Systems **20**(2) (2002) 133–170
3. Vazhkudai, S.S., Ma, X., Freeh, V.W., Tammineedi, J.W.S.N., , Scott., S.L.: Freeloader: Scavenging desktop storage resources for scientific data. In: IEEE/ACM Supercomputing 2005 (SC—05), Seattle, WA, IEEE Computer Society (2005)
4. Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: G-commerce: Market formulations controlling resource allocation on the computational grid. In: International Parallel and Distributed Processing Symposium (IPDPS), San Francisco, IEEEE (2001)
5. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. The Journal of Concurrency and Computation: Practice and Experience (CCPE) (2002)
6. Weglarz, J., Nabrzyski, J., Schopf, J., eds.: Grid resource management: state of the art and future trends. Kluwer Academic Publishers, Norwell, MA, USA (2004)
7. Foster, I.T.: The anatomy of the grid: Enabling scalable virtual organizations. In: Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing, London, UK, Springer-Verlag (2001) 1–4
8. Heiser, G., Elphinstone, K., Vochteloo, J., Russell, S., Liedtke, J.: The Mungi single-address-space operating system. Software Practice and Experience **28**(9) (1998) 901–928
9. Buyya, R., Abramson, D., Giddy, J.: NimrodG: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid. In: Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region. (2000)

10. Waldspurger, C.A., Hogg, T., Huberman, B.A., Kephart, J.O., Stornetta, W.S.: Spawn: A distributed computational economy. IEEE Transactions on Software Engineering **18**(2) (1992) 103–117
11. Regev, O., Nisan, N.: The popcorn market an online market for computational resources. In: ICE '98: Proceedings of the first international conference on Information and computation economies, New York, NY, USA, ACM Press (1998) 148–157
12. Wilcox-O'Hearn, B.: Experiences deploying a large-scale emergent network. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag (2002) 104–110
13. Adya, A., Bolosky, W.J., Castro, M., Cermak, G., Chaiken, R., Douceur, J.R., Howell, J., Lorch, J.R., Theimer, M., Wattenhofer, R.P.: Farsite: federated, available, and reliable storage for an incompletely trusted environment. SIGOPS Operating Systems Review **36**(SI) (2002) 1–14
14. Richard G. Lipsey and K.Alec Chrystal: Principles of Economics 9th Edition. Oxford University Press (1999)
15. Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. In: Proceedings of ACM ASPLOS, ACM (2000)
16. Devarakonda, M.V., Chess, D.M., Whalley, I., Segal, A., Goyal, P., Sachedina, A., Romanufa, K., Lassettre, E., Tetzlaff, W., Arnold, B.: Policy-based autonomic storage allocation. In Brunner, M., Keller, A., eds.: DSOM. Volume 2867 of Lecture Notes in Computer Science., Springer (2003) 143–154
17. Smith, V.L.: An experimental study of competitive market behavior. The Journal of Political Economy **70**(2) (1962) 111–137
18. Friedman, D., Rust, J.: The Double Auction Market: Institutions, Theories and Evidence. Addison-Wesley Publishing (1993)
19. Gjerstad, S., Dickhaut, J.: Price formation in double auctions. In: E-Commerce Agents, Marketplace Solutions, Security Issues, and Supply and Demand, London, UK, Springer-Verlag (2001) 106–134
20. Rustichini, A., Satterthwaite, M.A., Williams, S.R.: Convergence to efficiency in a simple market with incomplete information. Econometrica **62**(5) (1994) 1041–63
21. Assuncao, M., Buyya, R.: An evaluation of communication demand of auction protocols in grid environments. In: Proceedings of the 3rd International Workshop on Grid Economics and Business (GECON 2006), World Scientific Press (2006)
22. Morali, A., Varela, L., Varela, C.: An electronic marketplace: Agent-based coordination models for online auctions. In: XXXI Conferencia Latinoamericana de Informática, Cali, Colombia (2005)
23. FUSE: http://sourceforge.net/projects/fuse/ (2000)
24. Pattnaik, P., Ekanadham, K., Jann, J.: Autonomic Computing and GRID. In: Grid Computing: Making the Global Infrastructure a Reality. Wiley Press, New York, NY, USA (2003)
25. Kalagnanam, J.R., Davenport, A.J., Lee, H.S.: Computational aspects of clearing continuous call double auctions with assignment constraints and indivisible demand. Electronic Commerce Research **1**(3) (2001) 221–238