NICOLAS DUBE

SuperComputing Futures: the Next Sharing Paradigm for HPC Resources

Economic Model, Market Analysis and Consequences for the Grid

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en génie électrique
pour l'obtention du grade de Philosophiae Doctor (Ph.D.)

Faculté des sciences et de génie
UNIVERSITÉ LAVAL
QUÉBEC

2008

Abstract

Across computer science, finance and economics, this thesis is fundamentally an incentive engineering endeavor to contribute a pervasive and sustainable high-performance computing resource exchange system. Borrowing from both fundamental and technical analysis, the presented model builds market ratios and indices in order to balance user's and provider's sharing actions. While these resource indices do not constitute an absolute indicator to follow and therefore should not constrain the trading process; they nevertheless bring the essential market assessment, or *price anticipation*, to any Grid user before posting a market order.

In many ways similar to conventional commodity exchanges, the proposed *Grid Exchange* enables trading of resources through double-auctions and *Grid Credits*, a valuation instrument much more *liquid* than conventional bartering. Since there is no such thing as a "standardized" contract in HPC, the framework makes possible to buy or sell *multi-commodity* resource "sets", called *requirement sets* on the user side, and *component sets* on the provider side.

Simulation results demonstrate the economic model efficiency and capacity to adapt to changing market conditions. Through thousands of random events, market estimators show their effectiveness at hinting *HPC traders* on what their market positioning should be.

Formally, this economic model is nothing but a non-cooperative, zero-summed game, that eventually reaches equilibrium points. Users and providers are encouraged to trade on the market, at their will, and at the price they feel inclined to buy for, or sell for, whatever that price may be, as the "market is always right".

This *strategyful* approach thus paves the way to a more effective Grid resources sharing metaphor, where *share*, in order to be *fair*, can now be weighted. In the end, this new sharing paradigm for the Grid instantiates an entire new economy with countless possibilities; consumer and provider market orders strategies, third-party brokers, technology speculators and future HPC architectures risk hedging are only some of the aspects that may now be envisioned.

Résumé

À la croisée des chemins du génie informatique, de la finance et de l'économétrie, cette thèse se veut fondamentalement un exercice en ingénierie économique dont l'objectif est de contribuer un système novateur, durable et adaptatif pour le partage de resources de calcul haute-performance. Empruntant à la finance fondamentale et à l'analyse technique, le modèle proposé construit des ratios et des indices de marché à partir de statistiques transactionnelles. Cette approche, encourageant les comportements stratégiques, pave la voie à une métaphore de partage plus efficace pour la Grid, où l'échange de ressources se voit maintenant pondéré.

Le concept de *monnaie de Grid*, un instrument beaucoup plus liquide et utilisable que le troc de resources comme telles est proposé: les *Grid Credits*.

Bien que les indices proposés ne doivent pas être considérés comme des indicateurs absolus et contraignants, ils permettent néanmoins aux négociants de se faire une idée de la valeur au marché des différentes resources avant de se positionner.

Semblable sur de multiples facettes aux bourses de commodités, le *Grid Exchange*, tel que présenté, permet l'échange de resources via un mécanisme de double-encan. Néanmoins, comme les resources de super-calculateurs n'ont rien de standardisé, la plate-forme permet l'échange d'ensemble de commodités, appelés *requirement sets*, pour les clients, et *component sets*, pour les fournisseurs.

Formellement, ce modèle économique n'est qu'une autre instance de la théorie des jeux non-coopératifs, qui atteint éventuellement ses points d'équilibre. Suivant les règles du "libre-marché", les utilisateurs sont encouragés à spéculer, achetant, ou vendant, à leur bon vouloir, l'utilisation des différentes composantes de superordinateurs.

En fin de compte, ce nouveau paradigme de partage de resources pour la Grid dresse la table à une nouvelle économie et une foule de possibilités. Investissement et positionnement stratégique, courtiers, spéculateurs et même la couverture de risque technologique sont autant d'avenues qui s'ouvrent à l'horizon de la recherche dans le domaine.

Pour toi, maman, en l'honneur d'une vieille gageure...

"One must have chaos in oneself to give
birth to a dancing star"
Friedrich Nietzsche

Acknowledgements

First and foremost, I would like to thank Marc, my research supervisor, for giving me enough latitude to work on *integrative*, rather than *incremental*, research; this certainly made a huge difference on this thesis outcome. I would also like to thank my father, Michel, and my sister, Marie-Hélène, for their financial, moral and emotional support through all those years of graduate studies. Thanks to Virginie who lived with me through most of it, to Simon for understanding the daily Ph.D. student challenges and doubts, to Vincent for helping me out with everything else, to Martin (Boutch), Jonas, Louis-Philippe, Max and all my close friends that probably saved my last chunk of sanity. My gratitude goes also to Mora and Mia, for maintaining my feet warm all those long hours and pulling me from the computer once in a while; to Beethoven, Mahler, Tchaikovsky and all the others for their company in the late hours. Last but certainly not the least, my greatest appreciation goes to my beloved mother, Francine Ouellet (1950-1998), to whom this thesis is dedicated, for letting a borderline hyperactive kid grow up without too much restriction, encouraging my differences while making sure I would not get segregated and stimulating my creativity while protecting me to hurt myself too much...

Remerciements

Je voudrais d'abord remercier mon directeur de recherche, Marc, pour m'avoir laissé l'opportunité de travailler en recherche intégrative, car un "os" incrémental à gruger pendant autant d'années ne m'aurait probablement jamais rassasié. Un gros merci à mon père, Michel, et ma soeur, Marie-Hélène, pour leur support financier, logistique, moral et émotionnel aux cours de ces longues années aux études graduées. Merci aussi à Virginie, pour en avoir vécu l'essentiel à mes côtés, à Simon, pour sa compréhension, au quotidien, des états d'âme d'un étudiant au doctorat, à Vincent, pour m'avoir aidé avec à peu près tout le reste, à Martin (Boutch), Jonas, Louis-Philippe, Max et tous mes amis, trop souvent négligés, mais qui m'ont probablement sauvé de la folie. Un merci tout spécial à Mora et Mia, pour m'avoir gardé les pieds au chaud toutes ces heures et m'avoir forcé à laisser de côté mon ordinateur de temps en temps; à Beethoven, Mahler, Tchaikovsky et tous les autres pour m'avoir tenu compagnie aux petites heures de la nuit. Finalement, ma plus grande gratitude va à ma mère, Francine Ouellet (1950-1998), à qui je dédie cette thèse, pour avoir laissé grandir un enfant quasi hyperactif sans trop le restreindre, m'avoir encouragé dans mes différences sans que je me sente rejeté et avoir stimulé ma créativité en prenant bien soin de s'assurer que mes excès ne causent pas ma perte...

Contents

List of Figures

Chapter 1

Introduction

As Moore's Law [112] celebrated it's 40th anniversary in 2005, almost unchallenged, we are now seeing microprocessors featuring 45nm transistors, multiple cores, several levels of cache and many fancy features all integrated on die. Although Gordon Moore's original prediction, as shown in Figure 1.1, referred to the densification of features in silicon, many misinterpreted it as the doubling of processor clock rates every 18 to 24 months. Until recently, chip founders were able to enhance processor clocks at this pace because shrinking transistors brought energy savings and architectural advantages. Since a smaller gate needed less energy to switch state, integrated circuits voltage could be lowered, therefore reducing thermal dissipation and enabling higher clock rates. Additionally, for a similar die and wafer size, smaller transistors made possible architectural tweaks like deeper pipelines, allowing higher clock rates but not without lowering instructions per clock (IPC). Although this ever increasing clock cycle "marketing halo" sustained sales growth for some time, it concealed deeper and deeper pipelines and an increased design complexity, bringing in fact limited performance improvement for most applications.

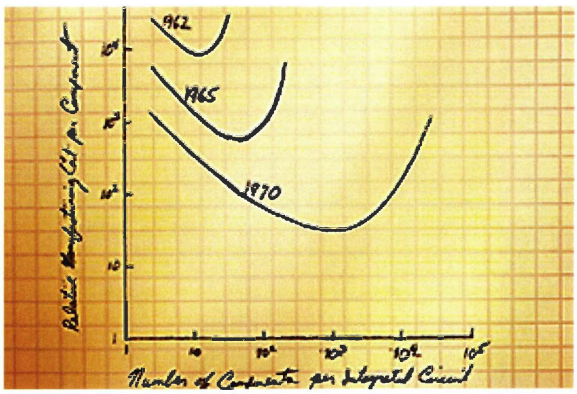

Figure 1.1: *Gordon Moore's original graph [112]*

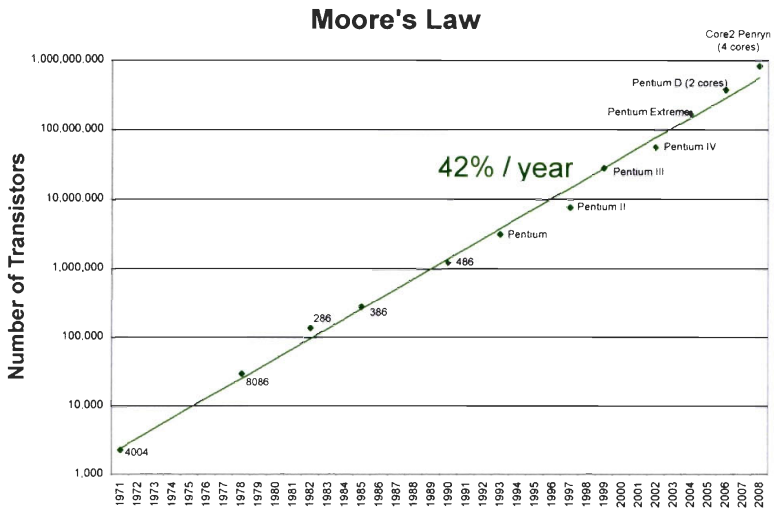

Figure 1.2: *Moore's Law over 4 decades [25]*

While processor throughput was steadily increasing (Figure 1.2), DRAM ICs were not able to sustain the same pace. This reality, known as the *memory gap*, introduces a major memory bandwidth problem, preventing many applications to scale further. Although processor manufacturers started using multiple levels of cache to compensate this issue, it is still the prevalent limiting factor for many scientific and commercial applications. Looking at Figure 1.3, one could think that because of the recent CPU clock slight drop, the memory gap may be narrowing. This assessment would ignore the fact that these lower clock CPUs not only have multiple cores, but smaller pipelines and several integer and floating point units, many features that significantly boost the bandwidth requirement per socket. While DDR3 DIMMs brings enhanced bandwidth up to 12.8 GB/s (from DDR2's 6.4 GB/s) with a lower power consumption, it has little chance to fulfill next generation CPUs data appetite.

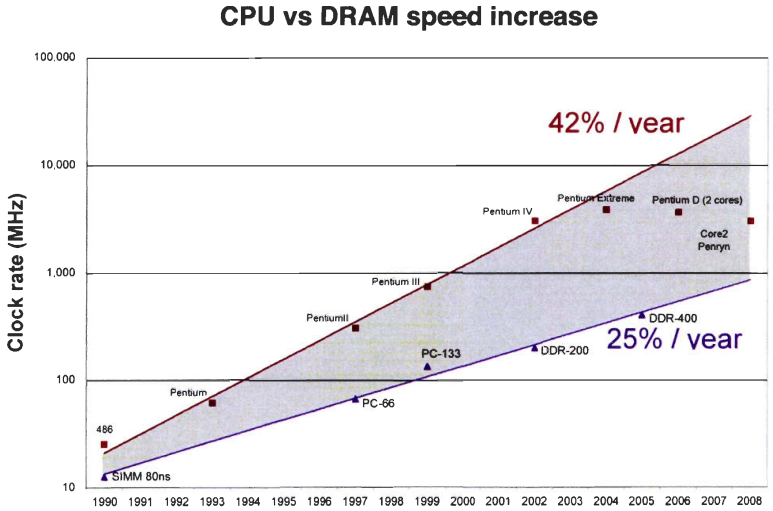

Figure 1.3: *The Memory Gap [25]*

The otherwise most forgotten issue in today's computing world involves data. Indeed, the computer component that experienced the fastest increase in the last 20 years has nothing to do with processing; hard drives capacities have stood a 72% yearly increase and, thanks to perpendicular recording, now bring us to 1TB of storage in a single disk. While in principle this sounds like very good news as more and more data can be collected, it also means that all of this data must be analyzed in some way, complicating even more the bandwidth issue.

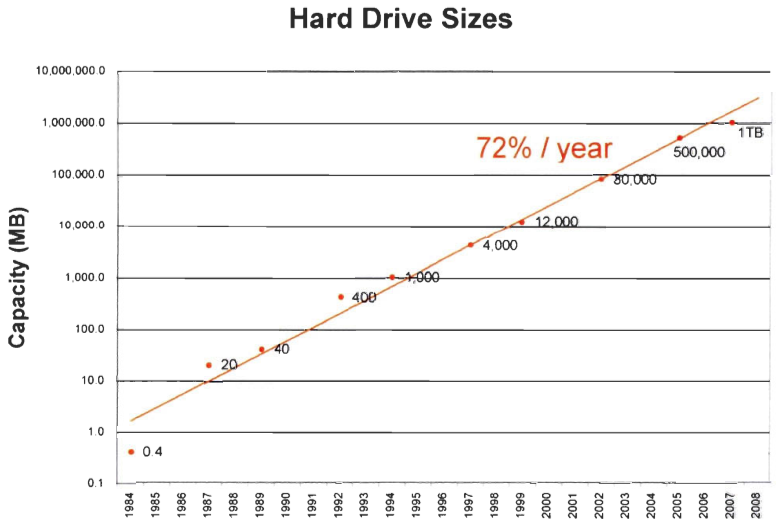

Figure 1.4: *Hard Drive capacity increase over 20 years [25]*

These trends show us that computer engineering and computer science main focus must no longer be processing in itself but rather moving the data in and out of the processing units. From the system engineer perspective, good choices will have to be made when designing multi-socket multi-cores computers. Should memory banks be linked to each CPU and CPUs linked together or should all CPUs go through a faster but unique memory controller? Should we start considering lower clocked architectures like IBM's BlueGene for more mainstream systems? Clearly, we'll have to start building more balanced computers with respect to processing, cache, RAM, hard disk and general I/O throughput.

For computer scientists and software engineers, it means the era of sequential programming is really over. Software developers now have to get a much better understanding of the hardware they will be running on; letting the default compiler figure it all is no more a valid option. Undergrads must now be taught how to use not only debuggers but memory mappers and application profilers. Mastering software development is now about being able to leverage multi-threading, using more efficiently memory accesses, defining better data read and writes patterns on hard drives and making sure a code clogging the I/O bus does not limit the application from behaving normally.

All in all, supercomputing has come to the desktop.

1.1 Science and Supercomputing

In the past, scientists used mathematics and somewhat "limited" experimental apparatus to conduct research that led to new discoveries. In today's world, an almost infinite variety of sensors are now linked to data-collection computers in order not only to extend the scope of an experiment, but also its scale and precision. Therefore, most scientists quickly run out of computing resources as they not only want to analyze experimental data but also to simulate costly experiments in order to allocate budgets and human resources to the most promising ones. While not necessarily being expert programmers, they quickly understand the purpose of supercomputers but are rapidly caught in complexity as they enter the sometimes quite tortuous road of parallelizing their code.

Weather forecasting was one of the very first mainstream and non-military uses of supercomputers and continues to drive today's massive computational requirements to accurately predict temperature, wind, humidity and pressure levels, which all have major impacts on transportation, agriculture and pretty much every aspect of human life. As supercomputing capacity evolve, models can be refined and predictions become more accurate.

Earthquake simulations and real-time monitoring are currently being run in some seismically active regions of the planet, supercomputing applications that could eventually save thousands of lives.

In the industry, fluid dynamics models are now computed for every car, plane, train or boat to be constructed in order to optimize its shape; limiting drag and thus boosting energy efficiency. Besides, for every complex structure to be built, finite elements models are executed on supercomputers around the world, maximizing the construction strength, resistance to nature events while minimizing its cost.

Scientists around the globe now envision taming problems that were way beyond belief a few years ago. For instance, the Large Hadron Collider (LHC) ATLAS experiment will require ~50 000 CPUs and 25 PB of storage for 2008 rising to ~240 000 CPU sockets and more than 200 PB by 2012 [87]; requirements so impressive any single supercomputing center could possibly support alone. Such projects are therefore designed to scale over several sites and thus require the collaboration of multiples labs scattered across many countries; supporting the need for a decentralized infrastructure for computational and storage resources.

These are just a few examples of scientific endeavors to be tackled in the years to come. Some codes run more efficiently on specific supercomputer types, others are more versatile. The next section flies over most architectures outlining their strenghs and weaknesses.

1.2 Supercomputer architectures

1.2.1 Vector supercomputers

Vector processing research started in the early 1960s within the Westinghouse Solomon project to finally end up in the ILLIAC IV in 1972, a 100 MFLOPS machine. The famous Cray-1 vector supercomputer followed 4 years later with a 240 MFLOPS performance thanks to its 8 vector registers (of 64 x 64-bit words each) and *vector chaining* [1]. Followed the Cray-2, Cray X-MP and Cray Y-MP, machines that enabled Cray to stay leaders in the supercomputing market until the early 90's. As of today, Cray now sells the X1E, a 147 TFLOPs vector supercomputer, the XMT, a massively multithreaded SMP with 128 TB of memory and the XT3/4, a large scale NUMA architecture built around AMD CPUs and HyperTransport.

NEC SX series, introduced in 1983 (SX-1), are now reaching 144 TFLOPS in a single SX-8R system with 4096 vector CPUs. NEC's most famous supercomputer is undeniably the Earth Simulator, a custom built vector supercomputer that held the top-500 list highest spot from 2002 to 2004 with its 36.5 TFLOPS.

Vector supercomputers are especially well suited for data intensive application since they are able to compute in parallel large chunks of data for a unique instruction (SIMD). This characteristic brings notable speedup in memory accesses since larger chunks are loaded into registers every cycle, therefore reducing the memory latency penalty. Hence, fluid dynamics and weather forecasting are two examples of especially well suited applications for vectors.

As some commodity processors now incorporate 128 bits vector processing units in addition to the traditional integer and floating point units, some of them also implement a feature called *fused multiply-add*, a FP unit improvement where an addition and a multiplication instruction can be run simultaneously in a single execution unit (recall Cray-1). Because of their massive end-user market, commodity processors makers have been able to leverage many formerly very high-end features of vector processors into low cost chips. While there is still a niche market for vector supercomputers in the very high-end national labs, their prohibitive development costs and limited market share makes their price/performance ratio unattractive for most general-purpose supercomputing facilities. For instance, while Cray owned a 40% market share in the early 90's, they now have about 3%. They nevertheless hold rank #5 and #12 of June 2008 Top500 list thanks to DOE's Jaguar XT4 (205 TFLOPS) and Sandia's Red Storm (102.2 TFLOPS) [23], neither Jaguar nor Red Storm being vector machines however.

[1] also known as *vector instructions pipelining, enabled on the Cray-1 by the implementation of addition/substraction and multiplication in distinct hardware [91]*

1.2.2 BlueGene

The BlueGene project is a supercomputer architecture designed to address many shortcomings of current supercomputer designs. This research project led by IBM also involves Lawrence Livermore National Laboratory and the United States Department of Energy. The first generation, called BlueGene/L integrates dedicated nodes for compute and I/O. Each node is composed of a single processor and associated DRAM chips (Figure 1.5 right). Each processor is a dual-core PowerPC 440 700 MHz with "double-hummer FPUs" able to issue 4 FP operations per cycle, for a total of 5.6 GFLOPS per chip. BlueGene/L is built on 3 communication networks, a 3D toroidal network for peer-to-peer communications, a collective network for one to all communications and an interrupt network for global barriers. The I/O nodes are also linked to an ethernet network to communicate with the outside world. Although BlueGene/L builds on many network infrastructures, all the nodes are independent and therefore not cache-coherent. The compute nodes run a minimalist OS and support only 1 user process. While BlueGene is very efficient on an *acquisition $/FLOPS* ratio, it is probably even more impressive on an *operation $/FLOPS* ratio because its power consumption is very low; in fact less than 20 KW / rack for 1024 compute nodes and associated I/O nodes.

Other members of the BlueGene family include BlueGene/C, BlueGene/P and BlueGene/Q. BlueGene/C, now renamed Cyclops64 has in fact very little in common with BlueGene/L apart from being launched within the same initiative to counter the Earth Simulator in the early 2000's. Cyclops64 (Figure 1.5 left) is a cellular architecture aiming to implement a "supercomputer on a chip" thanks to its 80 64 bits cores running at 500 MHz on a single chip. On a system scale, this means 1.1 PFLOPS and 13.8 TB RAM in 72 racks. BlueGene/P, second offspring, has just been unveiled and is designed to deliver 1-3 PFLOPS with quadcore 850 MHz PowerPC 450 processors. BlueGene/Q, last publicly known architecture in the family, is expected to develop a 3 to 10 PFLOPS performance in the 2010 timeframe.

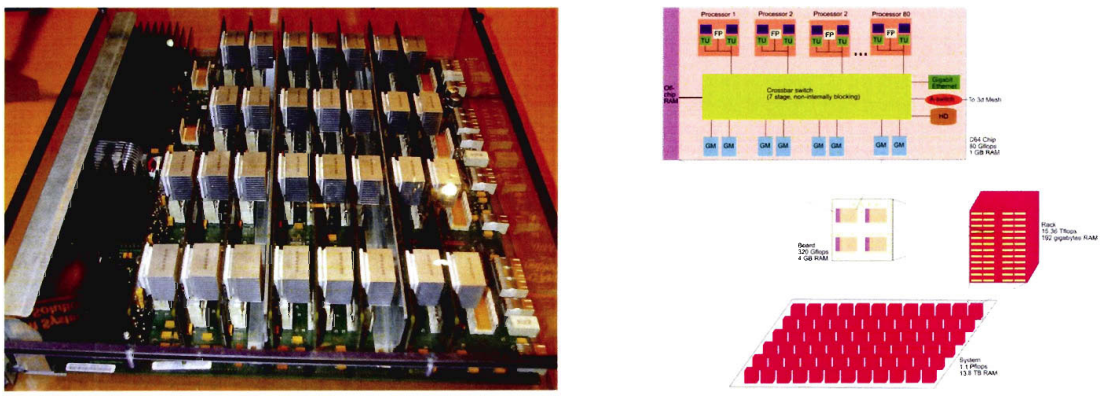

Figure 1.5: *IBM BlueGene node card and the Cyclops64 architecture [12]*

The one consideration that may not be totally straightforward for most academic general purpose supercomputing facilities is that although the BlueGene architecture presents some very effective hardware, its software development model isn't trivial for the average HPC application scientist. BlueGene therefore emphasize the previous statement that actual and future advances in hardware lead to a complete rethink of conventional software development paradigms.

1.2.3 Symmetric/Massive Multi-Processors (SMP/MMP)

Although SMPs and MMPs dominated the supercomputing world until the late 90's, as shown on Figure 1.6 [23], they are now being pushed to a supporting actor role as clusters take an ever increasing market share. By definition, Symmetric Multi-Processors are supercomputers with multiple cpus and memory banks on a unified motherboard, enabling a uniform access to memory for each and every process running on the machine. However, to develop bigger systems, SMP vendors had to leave the crossbar interconnect and move to ccNUMAs (cache-coherent Non Uniform Memory Access). Massive Multi-Procesors (MMPs) thus involve multiple processors linked through an high-speed proprietary interconnect technology, but presenting separate memory banks. These large scale NUMA supercomputers like the SGI Altix 4700, Hitachi SR11 and IBM pSeries nevertheless address the entire memory space on a system level enabling the programmer to deploy data all-over. One must just not forget that memory locality is still the one aspect in tweaking applications on such systems.

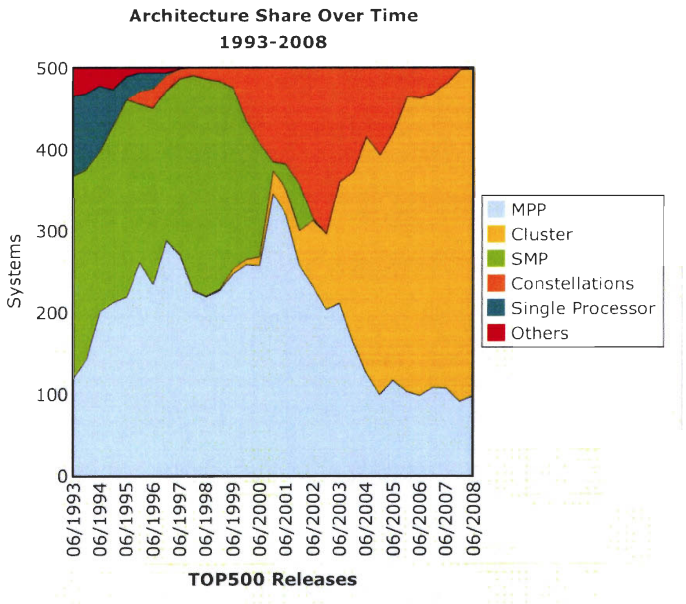

Figure 1.6: *Top500 Supercomputer architectures development over time [23]*

Shared memory among processors is a major advantage in many applications. In fact, for some very tightly coupled problems, SMP's powerful data exchange model outperforms any cluster alternative. However, a major drawback of SMPs is their prohibitive price. Their operating system and software stacks are also in most part proprietary; although this may seem as an advantage in regards to support, their limited user base quickly comes into play when tracking down an uncommon bug. While SMP programs were, for the most part, developed with the OpenMP [21] library, MPI versions of some SMP codes are starting to exhibit impressive, and sometimes improved, performance figures.

1.2.4 Clusters

As witnessed on Figure 1.6, clusters are the fastest growing architecture type on the TOP500 list. On the very low end, "serial" clusters are an aggregation of individual high-throughput server nodes linked together with low-cost interconnect, like gigabit ethernet. They present the lowest cost/gigaflops ratio and are widely employed to run large sets of independent jobs, as in many commercial applications, such as web farming.

The past few years have witnessed the emergence of a class of machines called "capability clusters", based on "commodity" hardware, either 1u nodes or blades, and linked together with high-throughput and low latency interconnects like Myrinet or InfiniBand. These machines have become very popular since they present an interesting tradeoff, leveraging the cost/gigaflop ratio of serial clusters and providing close to an SMP ability to run large coupled jobs. With the latest improvements in interconnection technologies, we are now capable of linking together cluster nodes at speeds up to 40 Gbps with latencies under 4 μs. This reality induces an important market pressure on traditional supercomputer vendors as low priced servers can now be competitively linked together. Therefore, if a researcher is able to divide the memory space needed by each job, and port his code to a message passing library like MPI [18], he may be able to get much more processing power per dollar out of a cluster.

In addition to interconnect technology improvements, cluster nodes have evolved recently from the single processor 1U server to multiple cores architectures in different form factors, from the 4U box to the very dense blade chassis. As blades now support 4 quad-core hyper-threaded processors and up to 64GB of memory; yesterday's limited commodity rack servers have really become small SMPs, able to run many highly-coupled threads on an entry-level priced box. Supporting this market pressure are also high-end libraries like MPI and MPI-2 that facilitates processor to processor and processor to storage communications. Henceforth, while a lot of HPC programs used to be coded with OpenMP, the leading paradigm has switched to MPI since it runs smoothly on almost any architecture.

Among the various research institutions are then pools of applications spanning from the embarrassingly parallel set of jobs to some tightly coupled shared memory code. Traditionally, scientists with ever increasing application size, complexity and processing requirements acquired bigger and bigger supercomputers specially suited for their needs. While Vector Computers provided SIMD datasets a notable speedup, SMPs brought a unique addressable memory space for large undividable data. Although these architectures present a high *cost/gigaflop* ratio, they support some codes that couldn't just be split up nicely in addition to a lot of legacy code yet to be ported to MPI. In addition, because of their somewhat easier programming model, there is still a case supporting the acquisition of these machines where the supercomputer efficiency is not measured by summing gigaflops but rather in some higher level research productivity measure, accounting programmer's time, for instance.

Pushed by the relatively low price point of commodity clusters, supercomputers have now become an almost ubiquitous research equipment in most universities and science laboratories around the world. Whether Symmetric/Massive Multi-Processors (SMP/MMP), Vector Computers, or Clusters, they provide a fundamental processing resource for many research fields. They nevertheless constitute a significant infrastructure to buy and operate and because any specific supercomputing infrastructure is still very expensive, very few sites are able to acquire each and everyone of them. This reality feeds the need for sharing among sites to make sure every scientists can have access to the best suited architecture.

1.3 Grid Computing

Grid computing, in one of it's original meaning, referred to distributed computing on a pool of desktop computers and workstations. Heterogeneous resources, loose network coupling and non-deterministic availability made grid computing a cheap but unreliable platform for massive computing. Furthermore, one of the biggest challenges of grid computing concerned the security risks associated with the forking of a user's application on remote compute nodes without any well-established authentication scheme. Most frameworks provided a remote execution environment and an API to ease the deployment of grid applications. Numerous commercial grid computing solutions existed such as Entropia [6] (defunct) and Univa UD [24]. Developed at the University of California in Berkeley, BOINC (Berkeley Open Infrastructure for Network Computing) [2, 33], the prevailing open-source flavor, provides a generic platform for distributed computing using volunteered computers. Distributed under the GNU Lesser General Public License (LGPL) [9], anybody can download the source and build his own massively distributed project using a single linux server. Volunteered computers can be running Windows, MacOS, Linux or Unix. BOINC, an evolution of the SETI@home platform, has seen several non-profit projects adopting it since its inception.

1.4 The Grid

A few years ago, the term "grid computing" encountered a fundamental meaning shift from the concept of distributed computing on a loosely coupled set of workstations to the far more complex "meta-computing" paradigm involving elaborate resources federation. In its actual definition [75, 74], "The Grid" main goal is to integrate, virtualize, manage and share resources and services on the scale of distributed, heterogeneous, evolving and multi-institutional *virtual organizations* (VO). In fact, grid computing now refers to the integration of supercomputers, workstations, sensors, databases and many other devices, regardless of their location and administrative domain. To become truly successful and sustainable, 'The Grid" [71, 72, 35], has to address the following aspects:

- *Multiple administrative domains.* The resources are distributed across multiple institutions geographically located around the planet and without any trust relationship by default. The infrastructure must support resource aggregation while respecting local management and usage policies.
- *Heterogeneity.* The computing resources, for example, will exhibit great disparities over the CPU architecture, memory availability, network interconnect, etc. User applications must be able to migrate smoothly and transparently from one hardware architecture to the other.
- *Scalability.* Grid deployments, like any technology, start off on a small scale and can grow exponentially with the growing community interest. The core infrastructure must therefore be lightweight and efficient to support thousands of nodes.
- *Adaptability and fault-tolerance.* By themselves, grids are unstable and unreliable. Fault-tolerance and adaptative middleware must consequently be put in place to provide an illusion of reliability by means of underlying replication and migration.
- *Extensibility.* Grid protocols must be standardized and extensible to support the integration of future devices. This requirement makes the difference between a transient technology and one that will last.

From a pure computational service model, grid computing evolved to an infrastructure including data services, application services, information services and knowledge services. Key to this integration and widespread adoption is standardization, supporting the need for a common entity to define the Grid architecture and its underlying protocols. This entity, formed in 1999, is the Open Grid Forum (OGF) and the architecture they proposed is presented in Figure 1.7. It has the form of a hourglass where multiple and heterogeneous user

applications and underlying fabrics are linked with a limited and optimized set of protocols and middlewares.

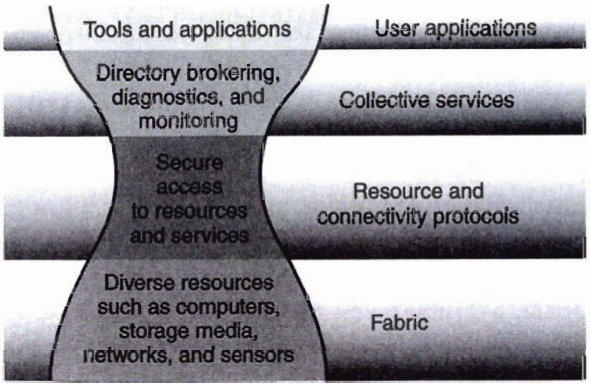

Figure 1.7: *OGF Grid architecture [72]*

The Open Grid Services Architecture (OGSA) [93, 97] is a standardized service-oriented framework built on Web service standards with semantics, additions, extensions and modifications relevant to Grids. OGSA is designed around the composition pattern or building block approach where a set of capabilities or functions is built or adapted as required, from a minimalist set of initial capabilities, to meet a need (see Figure 1.8). It is also worth noting that this architecture is not layered or object-oriented and services are loosely coupled peers that, either alone or as part of an interacting group of services, realize the capabilities of OGSA through implementation, composition or interaction with other services.

The OGSA found its first implementation in the OGSI (Open Grid Services Infrastructure) [143], a standard defining the building blocks of the distributed infrastructure including: grid service description and instances, service state, naming and name resolution, service life cycle, fault type and service groups. However, it did not define higher level behaviors regarding authentication, policy management, monitoring, data aggregation and resources federation. These functionalities where to be defined and implemented by the higher level abstraction layers.

OGSI has now been replaced by the Web Services Resources Framework [69]. The WSRF is in fact a complete refactoring of OGSI protocols using the Web Services Description Language (WSDL). Like most Web Services protocols, it is also using SOAP as the primary message exchange format. WSRF therefore defines an approach to modeling, accessing and managing state while targeting four main design principles: service discovery, service composition, specialization and extensibility.

OGF standards find their premier implementation in the Globus toolkit, detailed in section 1.4.3.. The following sections outline other mainstream grid architectures available.

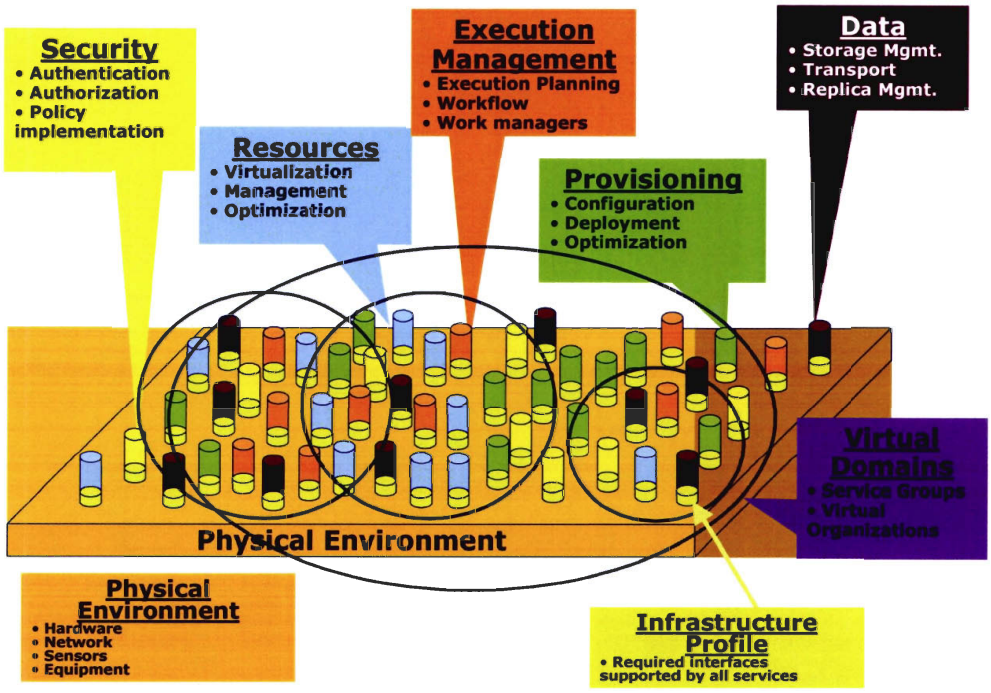

Figure 1.8: *The OGSA framework, where high-level services are composited using smaller building blocks [93]*

1.4.1 Legion

Legion [14] was born at University of Virginia in 1993 anticipating the networks bandwidth improvement of the late 90's. Project designers envisioned a generic, robust and scalable software infrastructure [86, 85] they could eventually transfer to industry [1]. In the Legion world, everything is an object. Legion API provides basic abstractions application developers can override to implement fancier behaviors. Legion objects are accessed through a three layer model starting with an object address (OA) bound to a Legion Object IDentifier (LOID) and ultimately to a local descriptor. Legion implements a *lazy coherence* scheme where the LOID - OA binding can become stale and updated only when a client calls the binding. This *update upon invocation* mechanism limits the network usage of a fully coherent data model but it introduces a potential overhead on any object access. By building such a generic infrastructure, Legion architects tried to provide every functionality to everybody. Like other IDL types development environments [16, 17], Legion is a little untargeted and therefore inefficient for compute intensive distributed computing.

1.4.2 Condor

Condor [5, 142, 78, 107, 106] is a distributed architecture maintained at UW-Madison since 1988. The name Condor comes from the idea of *hunting for idle CPU cycles*. It is a High-Throughput Computing (HTC) environment providing massive amounts of CPU power over a long period of time (whereas HPC can provide tremendous power on a short period). It is therefore primarily targeted at compute intensive, task-parallel jobs. It can be deployed on an heterogeneous pool of resources containing HPC clusters and uncoupled workstations. It provides a job queuing mechanism, scheduling policy, resource monitoring, and resource management. Condor implements checkpointing, cross-platform libraries and remote system calls where an I/O call in a job running on a remote machine will be forwarded to the machine the job was submitted from. By maintaining all data on the submission machine, Condor provides enhanced security eliminating the need for a remote account. However, this data model precludes densely coupled applications or data driven distributed computations to perform efficiently. Condor is still far more than an elementary grid environment; scheduling functionalities enable users to submit jobs in a queue and Condor will match the jobs requirements wilh resources available using its ClassAd mechanism. This matchmaking feature brings more users to the Condor pool since CPU requesters can specify their needs and CPU suppliers limit their contribution.

1.4.3 Globus

Globus [70, 8] is by far the foremost global grid software infrastructure since the late 90's. As a full-featured toolkit, Globus provides software for security, information management, resource allocation, data handling, communication, fault detection and portability. It is packaged as a set of components that can be used either independently or together to develop applications. From the very beginning, it has been developed as an open source project to foster code reuse, rapid widespread adoption and continual enhancements to the framework. From a mostly prototypal version 1.0 in 1998 to the first true version 2.0 in 2002, Globus, now version 4, has become the "de facto standard" for grid computing.

Globus World is an impressive conference taking place every year that pushes the implementation of the standards defined by the OGF. The latest Globus versions (v4) are built on the Open Grid Service Architecture (OGSA) standard and its underlying WSRF, both designed to favor code reuse, implementation flexibility and complex behaviors composition. While OGSA clearly sets the design principles for a wide scale aggregation of HPC clusters and workstations grids, its Globus evolving implementation will have to target functionality without sacrifying efficiency, scalability and ease of use.

Figure 1.9 exhibits the latest Globus version (GT 4.0.7) features and subsystems.

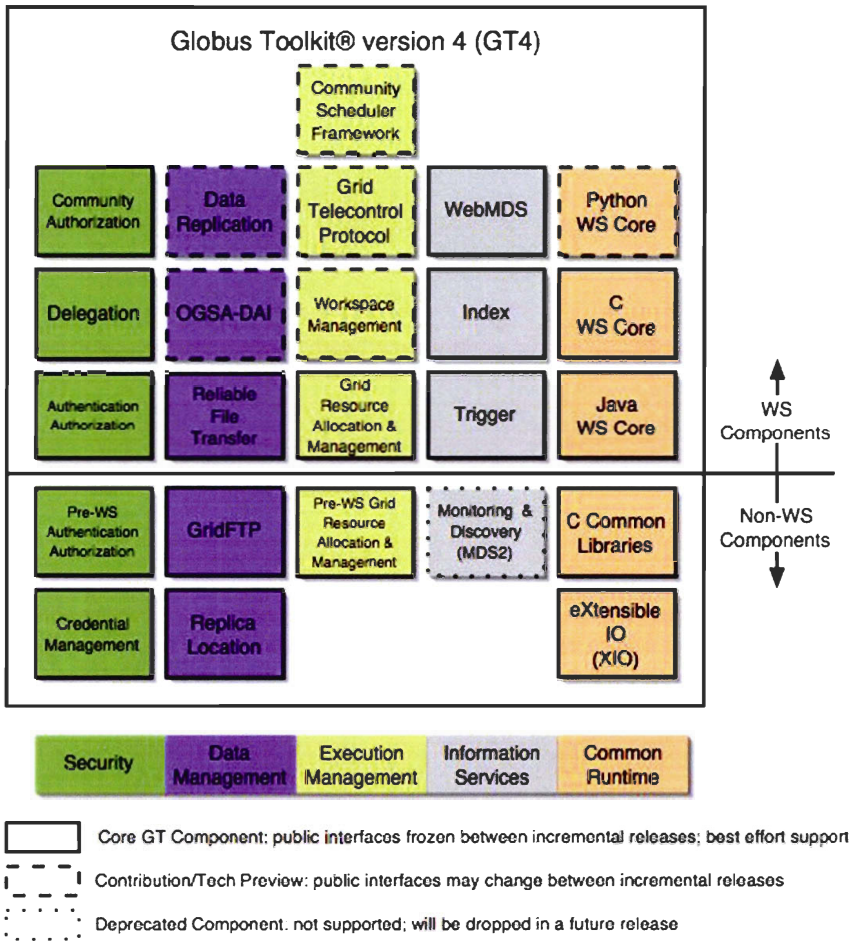

Figure 1.9: *Globus Toolkit v4 Components [77, 8]*

Security [105, 154, 152, 153]

The Grid Security Infrastructure (GSI) evolved significantly since its first implementation in GT1 where all it did was to provide message protection and authentication. While GT2 GSI version brought X.509 proxy certificates and Community Authorization Service (CAS), GT3 ported GSI to the web services standard and GT4 introduced the latest features. GT4 security framework basic component is *Credential Management*, that includes 2 modules: SimpleCA providing a simplified certification for issuing credentials to globus users and services, and MyProxy, an online credential repository that eliminates the need to copy private keys and certificates between machines. *Authentication and Authorization* are supported in 2 modules, the pre-WS version provides support for GridFTP , GRAM and other pre-WS widely used Globus components while the Web Services version brings message and transport level

security and the authorization framework to the newly adopted WSRF standard. GT4 new *Delegation* service provides an interface for delegation of credentials to a hosting environment in a way similar to a tier 2 CA, allowing credentials sharing across this environment and credentials renewal, a much anticipated feature. *Community Authorization Servers (CAS)* allow virtual organizations to express policy for resources distributed across multiple sites. This higher level authorization framework encapsulates local access control mechanisms and is able to grant fine-grained access rights to any individual resource scattered across the VO.

Data Management [31, 97, 32, 49, 30]

GridFTP is probably the one Globus component with the most widespread adoption. Some sites have indeed installed the whole Globus distribution for the sole purpose of using it. GridFTP provides a high-performance, secure and reliable data transfer protocol optimized for wide-area networks. Based on the well-known FTP protocol, it implements additional features required by Grids. GT4 version even supports striping, a mechanism similar to *BitTorrent* [56] where a file transfer may be split among multiple servers and clients to alleviate a single machine network bottleneck. The *Replica Location Service (RLS)* allows the registration and discovery of replicas. It implements data coherence mechanisms across VOs by mapping logical names for data items to target names. Users are then able to find replicas for a logical name among the various storage subsystems. *Reliable File Transfer (RFT)* is in fact a SOAP encapsulation of several lower level GridFTP operations. It provides an enhanced interface to control GridFTP parameters (TCP buffer size, parallel streams, etc.) and the ability to monitor with more detail grid file transfers. The *Data Replication Service (DRS)* is a WRSF service built by aggregation of the RFT and DLS components. Its primary function is to ensure that a specified set of files exist on a storage site. It begins by issuing a RLS request, then transfers the files pointed out using RFT and then registers the newly created replicas to RLS. *OGSA-DAI* permits the use of databases (relational, files, XML) across Grid deployments. It is a Java-based framework that can perform SQL queries using web services interfaces and is able to join multiple database requests through a single service.

Execution Management [65, 98, 135, 59, 73, 76]

Fundamental building block since the very first Globus versions, the *GRAM (Grid Resources Allocation and Management* service provides a standardized interface for requesting and using remote system resources. It provides a uniform and flexible interface to local schedulers, file staging before and after job execution and, as an improvement in GT4, the ability to select the account under which the remote job will be run. *WS GRAM* implements a web services

equivalent of the non-WS based GRAM. The new *Workspace Management Service* allows a Grid client to dynamically create and manage a workspace (currently implemented as a Unix account) on a remote site. It can create individual or groups of accounts and the included account service is able to manage account access policies and time to live (TTL), a truly interesting feature for one-time executions. The *Globus Teleoperations Control Protocol* is used to remotely control sensors for large scale geographically distributed experiments like earthquake engineering. It is in fact the WSRF version of NEESgrid protocol, the well known earthquake experiment initiative. Last, but not least, the *Community Scheduler Framework (CSF)*, the WSRF equivalent to GARA, extends GRAM features by allowing the submission, control and monitoring of jobs at the Grid level without the exact knowledge of the final execution site. It also brings the advance reservation capability for users and forwards these requests to local schedulers. Finally, it enables the creation of Grid-level queues of jobs, each with individually definable policies.

Information Services [165, 58, 99]

This portion of the Globus Toolkit has been completely shifted to the web services standard and therefore the pre-WS MDS2 is no longer supported. *WebMDS* brings monitoring and discovery of resources to users without the need to install any additional software since it publishes MDS4 information through a standard web browser interface. System administrators can customize data layouts using HTML form options and may bring any useful data into the framework thanks to plugins and XSLT transforms. Information services also include a *Trigger* component that monitors data across the Grid and, upon a pre-defined rule match, performs various actions, like sending an email when a system goes down. The *Index* service is a form of collector that aggregates information from scattered Grid resources and presents them in a single location.

Common Runtime

Globus *eXtensible I/O* library provides a single API (open/close/read/write) for file I/O across Grid deployments. Supported underlying protocols include TCP, UDP, file, HTTP, GSI, GSS-API_FTP and telnet. *C Common Libraries* provides a uniform basis for data types, system calls and data structures used across the Globus Toolkit. The *Python WS Core*, *C WS Core* and *Java WS Core* provide development frameworks to create WSRF-enabled services and clients conforming to the WS-Resource and WS-Notification specifications.

1.5 The Grid: Past, Present, Future

With the arrival of high-bandwidth backbones in the late 90's, computer scientists started dreaming about a worldwide federation of supercomputing resources: *The Grid* [72, 74, 75]. In the Grid, every single compute node, scientific sensor or data repository is virtualized and accessible to anybody, anywhere, anytime. Thanks to its impressive scale, the Grid enabled the computation of problems never envisioned before.

By design, the Grid was also to favor a better utilization of computing resources through inter-site sharing. Since some tightly-coupled problems run more efficiently on SMPs while others get more throughput on clusters, the ideal Grid virtualizes the execution environment and is able to match a problem with the optimal supercomputer at any given site. The Grid would therefore bring enhanced productivity through increased load and higher efficiency.

The term "grid" comes from the analogy with the electrical power grid [50]. But how far does this comparison hold? In both contexts, we are talking about a power exchange grid. To start off, computing grids link together producers and consumers just like electrical distribution networks. Also, large scale grids like the TeraGrid [22] or the EU DataGrid [7] involve an elaborate hierarchy with main production and distribution centers. Both domains rely on a complex infrastructure that must be maintained to provide good quality of service (QoS). Production facilities are complex equipments to be taken care by specialized staff. While electricity involves the transformation of different forms of energy (hydraulic, solar, wind, fossil), CPU power does the same with electricity. In both scenarios, power availability changes over time.

Computational power and electricity are therefore quite similar forms of goods. Where the similarities end is when we start to consider the social and economical aspects surrounding their exchange. Would anybody consider an electrical distribution network where consumers would get power for free, but relying on the producer's good-will? Large compute grids have been made possible in the last decade thanks to substantial government infrastructure investments, just as electrical networks came to light in the 50's. As the electrical grid grew and became widespread and sustainable, governments progressively transferred the infrastructure to industry. As of today, most electrical grids are owned and maintained by private sector companies and while some grids are still owned by public corporations, most of these corporations are profit oriented. While some forms of energy are renewable, most imply some form of limited supply and therefore consumers must be made more responsible. Even large electricity producing public corporations like *Hydro-Quebec* now realize that discounted energy pricing to local users is a bad idea in a global economy since it inevitably leads to wasting of otherwise valuable energy.

In principle, nobody is against a willingly sharing compute Grid. Indeed, most scientists agree when writing grant papers to funding agencies for HPC resources. C3.ca (now Compute Canada) Long Range Plan [3] reflects this general commitment to sharing. The recently approved 120M$ National Platform Fund (NPF) [4] awarded by the Canadian Foundation for Innovation (CFI) requires that the supercomputing infrastructure to be deployed across Canada will have to be available for each and every scientist from coast to coast. Since regional consortiums are planning to buy different while complementary hardware architectures, it is an additional justification supporting sharing on a national level.

But, in practice, how is this sharing system going to work? While a system administrator is overwhelmed by request from local users, how is he going to find time to support external users? And most importantly, what would be the fundamental incentive for him to do so? While CFI and other national funding agencies oblige sharing between sites, is there any accountable measure verifying this requirement? Would timeslots on one consortia vector supercomputer be traded equally with timeslots on some other consortia serial cluster? How valuable is computing on a capacity cluster compared to an high-end SMP? How much is a cluster node worth? Its components? Its network interface? These are unanswered questions that could seriously harm any future government investment in HPC if not taken seriously. From now on, sharing is no longer a possibility, it's mandatory.

On the user side, the picture is probably even fuzzier. While nobody is willing to get into the debate questioning the value of science, we all know that some problems are inherently more complex to run in HPC while others are straightforward. Most scientists with HPC needs lack this understanding and it quickly becomes quite complicated to weight their requests on the various allocation committees. Also part of the problem, following the allocation of free compute cycles, is their inefficient use, as pinpointed by most site operators. As for electricity, potable water, oil or gas, any free or undervalued resource tends to be wasted. But how "expensive" would be a data-driven application compared to a compute-intensive one? A tightly-coupled application versus an embarrassingly-parallel? One with strong deadlines? One with high quality of service (QoS) requirements?

Up to now, funding was provided to visionary researchers in the field to design the emerging grid. Building this kind of architecture involved high risk levels and the resulting product was far too unstable to be accounted rigorously. This explains the actual free and undervalued nature of Grid resources. However, the Grid infrastructure is almost out of its implementation transient state and can now provide a stable and definable quality of service. Computational power and other Grid resources must now be considered as valuable goods that can be assessed over time. *Grid Economy* is an emerging research field trying to define resource exchange policies and systems to put them in place.

1.6 Grid Economy

Grid Economists claim that the Grid will have to change its trading model to grow to a sustainable infrastructure [41, 28, 149, 53, 133]. Indeed, without minimizing the virtues of socialism, there is a case supporting the fact that HPC resources could be far better utilized if they would not be free. Unfortunately, few supercomputer site-administrators feel an obligation to optimize and share their resources with the global community; their day-to-day agenda being already too overbooked to consider these aspects. On the other hand, the actual way compute timeslots are attributed is based mostly on the type of science being done and on the political influence of the scientist behind it. There is absolutely no incentive for them to tune their code or optimize the search space in their data. This general lack of efficiency in HPC therefore comes from both the absence of reward for the operator and the consumer's lack of liability. It is the logical consequence of the free nature of resources and this is the main reason why computational economy could be the inevitable next step for the Grid.

Grid economy is a relatively recent research field that targets the development of the market-driven infrastructure for resource exchange across sites and users. Additional features to existing Grid frameworks would first have to include a banking system where user credits would be stored along with their Grid certificate. Following this scheme, a brokering infrastructure has to be implemented to exchange Grid credits between users and providers. Last but not least, a centralized trading system, or *Grid Exchange* is needed to post orders and match them accordingly. The following outlines current Grid Economy frameworks.

1.6.1 OCEAN

The OCEAN [20, 123] project began at M.I.T. in the late 90's and is now homed at University of Florida's Computer and Information Science and Engineering department. OCEAN stands for Open Computation Exchange and Auctioning Network. OCEAN implements a peer-to-peer design where any user can host an OCEAN compute node and any developer can build his own distributed task-parallel application. OCEAN therefore brings the accessibility of wide-scale P2P communities to grid computing. If implemented appropriately, this project's ideas present a clear success potential since it is based on an economic model involving a double-auction mechanism matching consumer demand with the cheapest producer offer. This could initiate a lot of interest from users willing to sell compute time and application developers willing to have a cheap alternative over dedicated HPC. OCEAN can therefore be seen as a market-driven BOINC. However, this project is in the very early stages and numerous implentation obstacles (schedulers, security, accounting, etc.) need to be tackled.

1.6.2 GridBus

The GridBus (Grid Business) [46, 44] project is developed at the Grid Computing and Distributed Systems (GRIDS) Laboratory of the University of Melbourne, Australia. It is an all-inclusive system that regroups the following components targeting the objective of building the grid economy infrastructure.

Grid Service Broker (GSB) [145]: Acts on a global grid scope and makes scheduling decisions based on users requirements like deadline, budget and QoS. It matches the demand with resources characteristics found in the Grid Market Directory (GMD).

Grid Market Directory (GMD) [164]: Database for publication and discovery of resources distributed across virtual organizations. Customers and high-level schedulers browse the GMD to match suitable offers with demands.

GridBank [39]: Maintains consumers and providers accounts where usage records are updated with job allocation and payments. It can be used in co-operative and competitive grid computing environments.

G-Monitor [126]: Web-portal for managing grid applications execution (initiation, monitoring and steering). It is in fact a front-end to GSBs or schedulers such as Libra or Nimrod-G [42, 27]. It provides credentials management et instantiates a Grid proxy for remote execution.

GridScape II [82]: GridScape provides a standardized architecture and essential components for the installation and monitoring of grid testbeds. It acts as a portal gathering information from distributed sources and presents them together seamlessly within a single interface.

Alchemi [108]: Built on Microsoft .Net framework, Alchemy bridges Windows based computers to the predominant UNIX grid. Grid nodes running Alchemy can be dedicated or voluntary. Alchemi is in fact an equivalent to BOINC, although limited to Windows.

GridSim [45]: Simulation toolkit capable of modeling various heterogeneous resources such as PCs, workstations, cluster nodes and SMPs. It can be used to simulate batch or sequential resources allocation systems evaluating the performance of various scheduling algorithms and heuristics.

Libra [133]: Economy-driven cluster scheduler implementing QoS on an homogeneous cluster. It tries to optimize the workload according to the "budget" allocated for each job and their deadlines. Since it's a user-centric scheduler, the optimization function targets maximum user satisfaction.

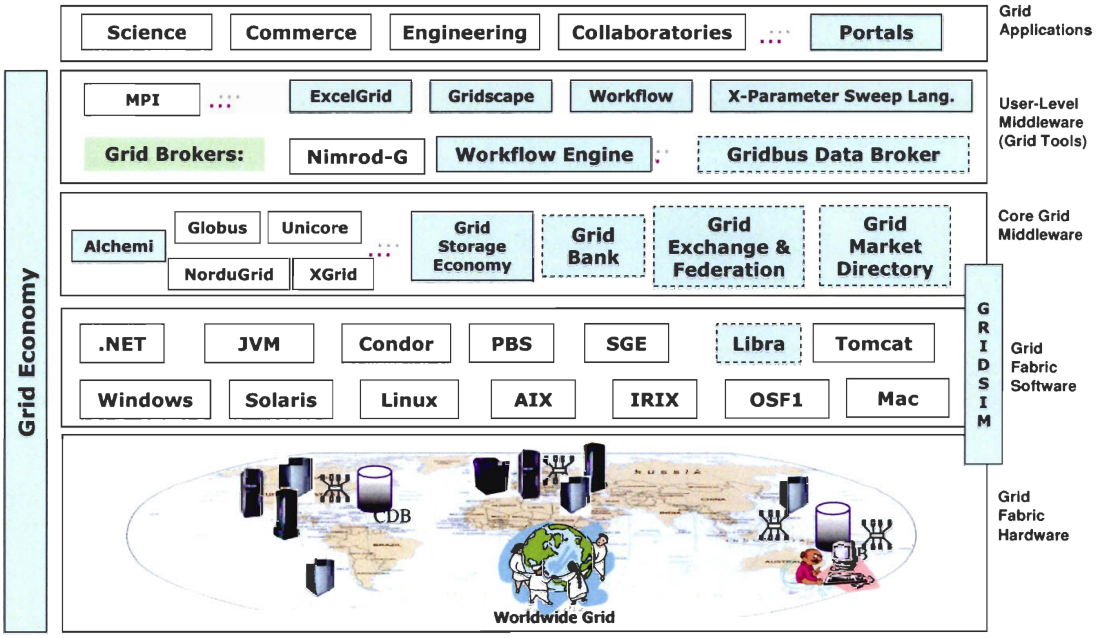

Figure 1.10: *GridBus Components [11]*

The GridBus project tries to reinvent Globus or other existing toolkits in many ways but fails to get widespread acceptance by the community. The Grid Service Broker, acting as a meta-scheduler, makes little sense in a true Grid Economy since any scheduling becomes totally irrelevant as timeslots will be traded per se and neither users nor providers will be willing to rely on a third party but unique broker to manage their resources. The Grid Market Directory could be used as a Grid economy building block although it would be more useful if it acted like a *clearing house* matching *sell* and *buy* orders like conventional commodity markets. Unfortunately, it also poorly integrates with GSI and other Globus hooks and therefore would make its general adoption difficult. GridBank, as an accounting framework, presents some essential features to support trading. It does not, however, integrate GSI certificates and would have to be extended to support inter-consortia exchanges. G-Monitor really brings no substantial improvement over Globus MDS as GridScape is also the equivalent of Globus Index service. As mentioned earlier, Alchemi is a pale copy of BOINC as it is a .NET only framework while BOINC, an established and well-supported toolkit [33], runs on Windows, MacOSX and Linux. GridSim may present some interest although it is not clear that learning to use and configure it may not actually be longer than running simulations on more generic tools like MATLAB. Finally, Libra, labeled "economy-driven scheduler" is in fact a local scheduler with a tweaked optimization function minimizing user "credit expenses" following their requirements. Although widely published, most of the GRIDS laboratory work tries to reinvent existing frameworks, and doing so, fails to nail fundamental social and market-related issues preventing the Grid to become a true economy.

1.7 Utility computing, market-based scheduling, resource trading economics and other related work

Back in 1968, I.E. Sutherland wrote what would have to be the first paper on market-driven scheduling[140]. The system described allocated "yens" to users of Harvard's PDP-1. They then used this currency to place bids for compute time on the schedule board (see Figure 1.11). Faculty and graduate students were allocated 10 yens, undergrads 1 yen and staff 5 yens. Maintenance and demonstration purposes were assigned large supply of yens for obvious reasons. Although users recovered their yens after the completion of their job, outstanding bids for future computer time could not exceed a user total yen allocation. This aspect is particularly important since it prevents "currency starvation" that could lead to computer idle time. Users and projects priorities are therefore indirectly assigned through the various yen allocation levels. While market-based, this system gave access to even the most "impoverished" undergrad students, at times of low demand. Sutherland concludes by stating that this system has been the most effective and led to higher computer utilization than any other scheduling system used before.

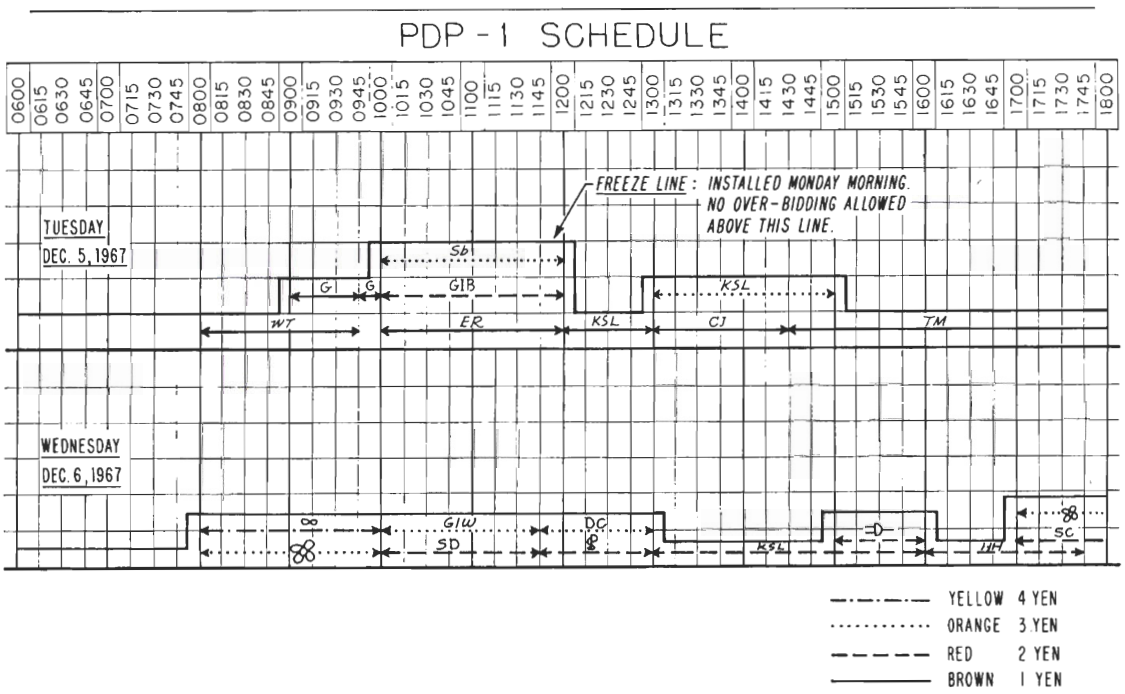

Figure 1.11: *Sutherland's PDP-1 auctioning system schedule board [140]*

Ferguson, Sairamesh, Yemini and Nikolaou developed microeconomic algorithms for load balancing in distributed processing systems [66, 67, 68, 131]. Their market-based model covered processing time and network costs on a mesh of homogeneous processors. Upon entry in the system, *Jobs* received an initial allocation of money, currency they then use to bid for CPU time and pay for communication expenses (if needed) between processors. They demonstrated that such *user-centric* scheduling techniques produced an enhanced resource allocation compared to traditional *system-centric* schedulers targeting global optimization. This improvement is mostly due to the competitive nature of the system where the law of supply and demand regulates selfish behavior of agents, whether suppliers or consumers. They also supported the idea that such economic systems present important structural benefits since the underlying algorithms are inherently decentralized and modular. One important flaw in their analysis though is that the allocation of money is performed arbitrarily, and as funding policies are a cornerstone issue in any market-driven system, it could lead to many undesirable situations from complete starvation to monopoly. Also, while their model applied reasonably well to the 1990's reality, it is too limited in our world of heterogeneous resources and fails to address the far from obvious pricing assessment issue that consumers and suppliers have to deal with, especially in today's highly complex Grids.

Miller and Drexler [111, 61] proposed pricing algorithms for computational and storage resources in *Agoric Open Systems*. The word *agoric* comes from the greek *agora*, meaning "open market place". They introduced a consumer pricing function following an escalator pattern with a slope proportional to the user willingness to close transactions. For storage and memory, they followed a *landlord* paradigm and defined the pricing strategy such that the "rental price" would stabilize at a level where supply equals demand (the *market-clearing rate*). While their work is one of the very few actually proposing fundamental algorithms for a computational resources market, they fall short in considering only one aspect of the trade equation in both compute and storage: buyer side for the former and seller side for the latter.

1.7.1 Economy-driven Networks of Workstations

Enterprise [110] is a decentralized market-driven scheduler for distributed computing on networks of workstations. To begin the trade sequence, a *client* posts a request for bids that includes a description of the task to be run, an estimation of processing time and a numerical task priority. *Contractors* with spare CPUs reply with bids including estimated completion time for the client's announced task. The client then has to evaluate the bids within a predefined period and awards its task to the best bidder. Although the *Enterprise* system allows for mutual selection through its 3-steps process, there is no possibility for either the client or the contractor to get any insight on the market value of their respective task or resource

since all deals are managed independently and no centralized market information repository exists. The market paradigm proposed by this architecture is therefore very limited and in many ways economically incorrect.

Waldspurger & al. developed *Spawn* [149], another market-based computational system that runs on a network (mostly LAN) of idle heterogeneous high-performance workstations running Unix. Using Monte Carlo simulations as a prototypal application, they explored issues of fairness in resource distribution, currency as a form of priority, price equilibria, the dynamics of transients and scaling to large systems. *Spawn* employs a *sealed-bid, second-price* auction mechanism. "Sealed" refers to the notion that bidding agents have no access to other agents bid information and "second-price" means that the highest bidder will pay the price offered by the second-highest bidder. In such a system, if there is a unique bid for a resource, the bidder gets it for free, in the absence of competition. For a grid of idle workstations, this scheme can make sense; for HPC resources with high operating costs, a producer would probably prefer running idle to reduce its electrical bill. One major limitation in *Spawn* lies in its scalable design that implements a different "market" on each machine; this leads to high price volatility between compute workstations as there is no centralized "bulletin board" for comparison purposes. While *Spawn* prove to be an interesting tool to distribute task on a pool of workstations, its conceptual limitation to processing time makes it inappropriate for a wider resource trading economy that would have to include memory, disk, networking, and other computing features.

The POPCORN project [129, 128, 121] extends *Spawn* local resources concept to the global computing scale of workstations linked to the internet. Implemented in Java, POPCORN enables coarse-grained applications to compute securely on distant nodes. In the POPCORN market, Java Operations (JOPs) are traded between seller and buyer against *popcoins*. The price for each *computelet* (the POPCORN applet for remote execution) is proportional to the amount of JOPs needed for successful execution. This JOPs figure is approximated using a piggy-backed benchmark on the *computelets*. Project leaders, Regev and Nisan, investigated 2 forms of exchange for *popcoins*. First is a mode where sellers connect to a market web site, input the information and start selling JOPs while running an applet in their web browser. The other mode, quite imaginative, starts up when a "seller" visits a web page of some interest to him. This web page, presenting a POPCORN logo, starts up an applet in the "seller's" browser, in background. In this mode, the "seller" is "paying" for the information on the web page while contributing JOPs in the background. While limited to a grid of loosely coupled PCs running Java applets, the POPCORN project investigation of economic foundations is one of the most interesting as they have simulated pricing behaviors and equilibrium points for Vickrey auctions, double auctions and clearing house auctions.

Others have worked on economic proposals for peer-to-peer distributed computing [146,

63, 88, 141] to address the free-rider problem of users connecting in and out of such systems. With an economic incentive, users tend to be far more reliable while contributing.

1.7.2 Market-based resource allocation and scheduling

Mariposa is a large scale database system developed at the University of California, Berkeley [137, 138]. Its intent is to store millions of objects across thousands of autonomous sites with very large storage capacities. As the consequence of the inherent management complexity of such a distributed system with traditional techniques, *Mariposa* implements an economic paradigm where each site seeks maximum profit by buying, selling and processing storage objects. While limited to database applications, *Mariposa* design constitutes another proof supporting the fact that market-driven approaches tend to be far more scalable and maintainable than deterministic ones.

Stoica's microeconomic Scheduler for Parallel Computers [136] implements an auction-based approach for parallel systems with identical processors. At the beginning of every time-slice, the processors initiate a sealed bid first-price auction where the client with the highest bid "wins" the CPU for the specified timeframe. A client then assigns a savings account to a job to pay for the computation. In this system, if the job fails to terminate before the pre-allocated timeframe, it can continue using CPU resources as long as it has the money to pay for it. While this system addresses the non-trivial problem of market-based scheduling on parallel systems while previous work targeted networks of workstations [66, 110, 149, 129], it fails to provide pricing information to clients and, using sealed bid first-price auctions, bring high-volatility, a drawback nobody wants for any sustainable economy-driven system.

Chun and Culler advocate for a market-based resource management system for batch scheduling on clusters of workstations called REXEC [53, 54]. Implemented on top of a stride scheduler, REXEC users assign a scalar *tickets* figure to their jobs. While executing, REXEC accounting system charges users proportionally to their job *tickets* over the sum of all executing jobs *tickets*. While this system demonstrated in simulation a higher aggregate user utility, users have no guaranteed cost for execution as it depends on the total demand for the execution timeslot. Under more realistic economic circumstances, users might have choosen to run their job on lower demand timeslots. They have later refined the scalar utility feature of REXEC proposing a linearly time-decaying utility function [55]. They demonstrated by simulation that their *FirstPrice* auction model improved Shortest Job First strategy by as much as 14x for highly parallel workloads. Since users are forced to assess their job, the scheduling algorithm gets more insight on which jobs to prioritize. But again, *FirstPrice* scheduling presents the same fundamental economic flaw in the absence of *a priori* price agreement.

Irwin & al. [96] present an evolution of this user-centric or value-based approach by incorporating an urgency component related to the rate at which the value of a job decreases as it waits in the queue. Their *FirstReward* scheduling algorithm is configured to balance the risk of keeping a task waiting with the reward of completing it immediately.

Popovici and Wilkes [127] refined *FirstPrice* [55] and *FirstReward* [96] schedulers by introducing *FirstProfit*, *FirstOpportunity* and *FirstOpportunityRate*. All three algorithms are tightly coupled to an admission control mechanism that, upon verification of a job profitability, will choose to accept or reject it. These new algorithms offer a controllable tradeoff between risk and potential reward, allowing, for example, the greedy service provider to tune the parameters for higher profit but at a higher risk.

Developed at HP Labs by Lai & al., *Tycoon* [104] is another market based distributed resource allocation system based on proportional share. Along with other market-driven schedulers, it forces users to differentiate the value of their jobs in order to increase to overall utility. *Tycoon* implements *continuous bids* where users express a bid for a resource r on a given host h at a given price point b and for a period of t seconds. On each host, the *auctioneer* calculates b_i/t_i for each bid i and allocates resources in proportion to bids. While this continuous bidding scheme alleviates user's burden of posting bids on the market for each job, it does not guarantee resource allocation prior to execution, a major impediment to most users with computational deadlines.

SHARP [79] is an architecture for distributed resource management, resource control and sharing across sites and trust domains. Its stands for *Secure Highly Available Resource Peering* and is based around *timed claims* that expire after a specified period, following a classical *lease* model [84]. In fact, SHARP implements *tickets* that are a form of *soft claim* that does not guarantee resource availability for the specified timeframe. Tickets enforce probabilistic resource allocation in a manner similar to airplane tickets reservation that only becomes "real" once the boarding pass is printed. This allows SHARP system managers to overbook [144] resources and to defer *hard claims* or lease allocation until execution. SHARP also presents a very strong security model to exchange claims between agents, either site agents, user agents or 3rd party brokers, that achieves identification, non-repudiation, encryption, and prevents man-in-the-middle and replay attacks.

As SHARP was prototyped on a PlanetLab deployment, Irwin & al. extends its reach [95] to the Cluster-On-Demand project [47, 48]. Their implentation, called *Shirako* allocates and configures *virtual clusters* that can be either physical servers or Xen [38] virtual machines. They demonstrated an enhanced adaptation to changing load through the use of a brokering infrastructure and better scaling performance thanks to decentralization, common results across utility computing literature.

Irwin, Chase, Grit and other members of the NICL lab at Duke University revisited [94] the concept of *self-recharging* currency proposed previously [140, 150]. A *self-recharging* currency means that the purchasing power of any individual is automatically restored after a predefined delay, a mechanism that prevents hoarding and starvation at the same time. They implemented this model in *Cereus*, the evolution of *Shirako* [95] and the COD project [48]. In principle, a *self-recharging* currency is an interesting idea, especially for bootstrapping a computational economy. In the long run though, more sustainable economic models may have to be put in place, allowing, for instance, currency earned by resource providers to flow back in the economy.

Yeo and Buyya [163] proposed a pricing model for utility-driven management and allocation of resources on a cluster. Their model implements an accounting scheduler where users pay *à posteriori* for their effective resource usage. While this model generates a higher payoff under high demand for the provider, there is no assurance for the consumer that the provider won't load the resources at run-time (deliberately or not), reducing available resources thus raising the price paid by the consumer. Few users may be eager to adopt this model without any billing upper bound.

While many others have worked on market-driven compute time brokers and schedulers [51, 128, 81, 90, 27, 43, 133, 109], few dynamic pricing models exist as it is considered that human decision would determine job valuation. Utility functions used in most market scheduling approaches imply a strong consumer's knowledge of a job's value; an assessment most application scientists are quite far away. Furthermore, while a job might retain an intrinsic value related to its outcome, this is only one part of the assessment equation in a true market environment; resource availability at specific time-frames must also be taken into account and reflected in the original consumer's bid. For these reasons, market-based scheduling has remained mostly a research topic failing to acquire a sustainable user base in any of the aforementioned implementations.

1.7.3 Other, more fundamental, related work

Wellman & al. developed *WALRAS* [156] (after the 19th-century French economist Léon WALRAS), a market-oriented programming environment. By *market-oriented programming (MOP)*, Wellman refers to a general approach of deriving solutions to distributed resource allocation problems by computing the competitive equilibrium of an artificial economy. As a prototype environment for specifying and simulating computational markets, *WALRAS* implements mechanisms to specify various sorts of agents, auctions and bidding protocols. Upon the reception of users and providers bids, *WALRAS* uses an algorithm similar to Léon Wal-

ras original *tatonnement* [151] mechanism where the various resource prices are adjusted iteratively as there is an excess of supply or demand to reach an equilibrium point, at the *market-clearing price*. While *WALRAS*, as a prototype *MOP* framework, provided good insight on the fundamentals of multicommodity markets (transportation, networking routes, spectrum auctions, etc.), its application to today's grid resource exchange is limited because, by design, *WALRAS* assumed *gross substitutability* between resources in a bid vector as a sufficient condition for stability [34]; in the HPC world though, resource *complementarities* are the common case, and the trading architecture must be able to handle it properly.

Feldman and Lai supplemented [64] *Tycoon's price-anticipating* scheme analytically and through simulation. Recalling *Tycoon's* continuous bidding mechanism where a user commits credits per time period and receives the ratio of his bid to the sum of bids for that resource, they demonstrate that this approach reaches an equilibrium point where *strategic* users can optimize their utility. Comparing this mechanism to a social optimum function with high efficiency but poor fairness, they show that this approach brings high utility uniformity while also improving fairness in the allocation.

Lai recently published [103] an analysis of many fundamental aspects of resource allocation markets. The strategic behavior of users leading to a *Tragedy of the Commons* [89] being the main reason supporting market-based allocation, Lai demonstrates that markets provide higher utility than *proportionnal-share* scheduling and improve resource allocation fairness. He also shows that even without real money involved or varying level of utilization, pricing predictability should not be a concern in the long run.

Ng, Parkes, Shneidman & al. worked on theoretical foundations of market-driven computational systems [119, 118, 117, 134, 124]. In particular they have strived to define *strategyproof* economic mechanisms where resource allocation and negotiation are incentive compatible and where users treat other users resources as their own. Implementing this paradigm is *Mirage* [52], a microeconomic resource allocation system based on repeated combinatorial auctions (a form of double auctions expressing tradeoffs between various options) for a tested of 148 nodes wireless sensors. *ICE* [124], an Iterative Combinatorial Exchange, further extends the combinatorial auctions concept in a framework that could be used to trade airport takeoff and landing or wireless spectrum allocations. Within such a system, a bidder could declare that he is willing to pay 1M$ for Quebec city, Montreal and Ottawa spectrum while releasing Calgary's. While interesting theoretically, combinatorial auctions may not be the right solution for trading grid resources because of the inter-dependencies within a resource set. In addition, while *strategyproof* mechanisms may seem desirable, especially to bootstrap an economy, we will show later that strategic behavior may not, in fact, be harmful for a sustainable market.

One of the most interesting works investigating the realms of a true Grid Economy is the *G-commerce* project [162, 161, 160, 159, 158]. Wolski & al. are among the very few that consider an unregulated market system where consumers and providers act in their own interest for the system to eventually reach an equilibrium. In [161], the Grid resource allocation is studied under two distinct market conditions: commodity markets and auctions. Both trading strategies are then compared in terms of price stability, market equilibrium, consumer productivity and provider efficiency. Their simulations and results showed that both approaches reach equilibrium and price stability although the auction model tends to be far more volatile than the commodity one, a logical result. Lastly, *G-commerce* implements, like many others, the concept of renewable currency for users while imposing an expiration date on all currency allocation. On the other side of the equation, producers earn credits but have no way of redeeming them. They also have to agree to let a job run until completion, once accepted. Without diminishing the importance of this work as it represents probably the only attempt to truly define the economic foundations of the Grid; the renewable currency, the impossibility to re-inject earned credits in the economy and the obligation to let a job run until completion are 3 constraints that contradict a freely evolving Grid market.

Part of the most recent developments, *Bellagio* [36] is a distributed resource discovery and market-based allocation system for federated distributed computing infrastructures proposed by AuYoung & al. Designed to work over PlanetLab [125], *Bellagio* uses *strategy-proof* [119] mechanisms forcing users to reveal their true valuation of resources in the form a combinatorial auctions to be cleared following a second-price Vickrey algorithm. In practice, *Bellagio* uses SWORD [122] to provide resource discovery to users following a pre-specified resource set. Following SWORD's response, users then express *candidate bids* in the *XOR* [120] language, reflecting tradeoffs they are willing to make in order to get varying levels of a resource, but at different price points. While *Bellagio* probably incarnates the actual best shot at defining and implementing a working market-driven resource exchange, its design around combinatorial auctions increases significantly the matchmaking algorithms complexity, lessening users understanding of the system and imposing a significant computational burden on the clearing house authority as this problem is NP-complete [132]. Also, while AuYoung and al. follows the ideology that *strategy-proof* allocation is a "no-brainer", one must be careful since following this model, resource provider's influence on pricing is almost nonexistent and this approach completely neglects the resource pricing assessment most users are far from knowing and thus provides no guidelines preventing uncontrolled market volatility.

While some others have analyzed economic acpects and auctions protocols that could be used in a resource economy [139, 155, 102], most of the these approaches targeted *market-clearing* prices for combinatorial auctions while others used Vickrey auctions where the provider has no influence. As it will be demonstrated, *clearing* the market and minimizing users expenses might not be the holy grail to design economic mechanisms after.

1.8 Problem Statement

As previously discussed, the actual "socialistic" Grid sharing paradigm fails to achieve high levels of efficiency and falls short in guaranteeing any specific resource access to remote distributed users. Indeed, the actual lack of provider accountability leads to unequal quality of service for most external users. On the other side, few users exploit supplied resources efficiently, as there is no incentive to do so. Even with grid frameworks and meta-schedulers trying to maximize global resource usage, aggregate user utility remains poor because users cannot express their time-varying eagerness for resources. As of today, the Grid dynamics are very difficult to model and as meta-schedulers make resource allocation decisions at runtime, their control loop struggles against non-linear feedback, as load fluctuates. Although the feedback-based control theory may have seemed a viable scheduling strategy at first sight, under these mechanisms, system response is described by stochastic models. Unfortunately, such models are nonexistent for the Grid and may prove very hard to define in the forseable future.

While burgeoning Grid authorities are willing to make resources readily available to early adopters as a way of cultivating a user community, resource costs must be considered if the Grid is to become pervasive. The *free-rider* problem is an issue that has been studied and in some manner addressed for peer-to-peer systems [56, 29] but the incentive engineering problem for the Grid fails to be addressed adequately.

Previous work done in *utility computing*, *market-based scheduling* and more fundamental resource trading economics provide empirical and analytical evidence supporting the effectiveness of market-based mechanisms for scheduling computational resources. In such an economy, every agent makes individual decisions, selfishly trying to maximize its utility. As prices in a market fluctuate in accordance with the supply and demand of resources, the system is able to dynamically adapt to resource contention. Under such a model, users are able to express their preference for specific kinds of resources at specific timeframes, a feature unimplementable in any control-based global resource optimization scheduling algorithm.

Because the decision process in inherently decentralized, market-based systems are able to achieve soaring scalability and redundancy, avoiding the single point of failure of conventional meta-schedulers, with minimal communication and computational overhead. In addition to these technical advantages, such exchange platforms, following fundamental economic principles, also bring better interoperability between the various sub-Grids as agreeing on the rather general trading mechanisms is far easier than linking different meta-scheduling frameworks and their intrinsic policies. Thus, distributed schedulers only have to be glued to high-level trading protocols, a task far easier than supporting true meta-scheduling. Hence-

forth, instead of entering meta-scheduling architectures technical debates, as common economic principles have a broad understanding in the community; they therefore present a definite bonus toward the acceptation of a Grid resource exchange system.

In the past few years, big corporate players have proposed computational services "on-demand" through various initiatives [57, 157]. While these services fulfill a need for pharmaceuticals and other businesses alike not willing to deploy and operate an extensive datacenter, they fail to conquer other markets outside of these niches. The pricing proposed reflects operational costs along a substantial profit margin based on the risk and expertise inherent to running an HPC infrastructure, but ignores true market fundamentals. In the academic world, these corporate initiatives inspire little interest as academics prefer to own the infrastructure and share it within the community. For the most part, resources are acquired, deployed and operated locally by scientists and support staff. Nevertheless, for academic Grids to become sustainable infrastructures, a pricing scheme reflecting operational costs, even using a virtual currency, may be an idea worth considering.

In previous work, pricing has been solely defined by users through english auctions, Vickrey auctions and proportional continuous bidding. As resource providers had barely no grasp on the market dynamics, this situation creates a restricted market instantiation regulated by consumers alone. While neglecting providers input could, in principle, simplify the trading system, it will inevitably lead to high volatility linked to application scientists humors. Since resource providers on academic Grids are funded through governmental agencies, one may wonder why they could be willing to have some leverage on pricing. Hence, as operating expenses tend to be a cornerstone issue for most HPC sites, a provider may prefer sending a compute node to sleep, saving some portion of the ever-ascending energy expense, rather than "earning" almost nothing in a second-price auction, an impossible alternative in any of the proposed systems in the literature.

Others have developed *market-clearing* algorithms similar to *clearing-houses* in commodity markets. For sure, *clearing* a market maximizes utility as all resources find jobs under high demand and all jobs find resources under low demand. This assumes that users and providers are willing to trade at the *clearing price* to be computed by the framework, whatever it may be. In some commodity markets (e.g. London gold), the *market price* is thus defined to make sure that commodities don't get unused or undelivered, as both *hedgers* (see chapter 2) want to secure the trade. As we will see throughout this thesis, Grid resources present some fundamental differences to these commodities and both users and providers may not be willing to "clear the market" at all cost.

To start, Grid resources are commodities that are produced at runtime, or "on the spot". Therefore, their exchange presents more similarities with *spot markets* for energy or foreign

currency exchange (FOREX) than usual commodity markets like oil, metals, soybeans or coffee. Since an HPC site administrator does not invest *a priori* to manufacture or grow the commodity, he may not be willing to sell at any market price settled by a clearing house while incurring the electricity expense to operate. Similarly, on the electricity spot market, hydro-electric producers sell their surplus but they normally won't sell energy at a price where it would be preferable to shut down turbines and leave the water in the reservoir.

In addition, and most importantly, resources on the Grid are to be traded in sets. Although the actual market-based approaches were mostly restricted to CPU time, HPC users must be able to specify joint requirements not only in CPU time but in RAM, storage, networking, software stack and many other features needed by an application. This is the most important difference with traditional commodity market where standardized amounts of a unique and identical good are traded and delivered. It is thus obvious that a unique market-clearing price for heterogeneous sets of goods cannot be computed.

Combinatorial auctions, as used in *Bellagio*, provide little help in this regard as they express tradeoffs a user is willing to make between various resource sets. For example, a user may be willing to pay a price X for a 2 GHz CPU, 1 GB of RAM and 10 GB of storage while he would also be willing to pay 1.5X for a 2 GHz CPU but with 2 GB of RAM and 40 GB of storage. In a combinatorial auction, the bids are mutually exclusive and the clearing house must iterate to a market equilibrium maximizing utility. This system does not, however, provide any price assessment mechanism for resources within a set neither does it provide any *asking* capability for a double auction involving the provider. In principle, combinatorial auctions may eventually prove useful for Grid resources trading, but, as of today, they address a problem yet to be expressed as many far more fundamental issues need to be dealt with first.

The *price-anticipating* algorithm described previously had the advantage of securing a price to compute at a given time. But its inherent flaws are probably bigger than other approaches implementing dynamic pricing as users in this scenario have absolutely no guarantee on the amount of resources they will get for the pre-defined cost. It's like ordering a car for 100 000$ without knowing if you'll get a sub-compact or a roadster. Nevertheless, by definition, price *anticipation*, and not *pre-determination*, may be a worthwhile concept that we may want to keep in mind.

The idea of *renewable currency* originally proposed by Sutherland and then re-used by many looks very interesting at first sight. Indeed, this open-loop mechanism automatically re-feeds users as they consume their allocated currency in exchange of computational resources. The main problem in this approach is the determination of the credits allocation rate to individual users. While it may turn out to be useful to bootstrap an economy, policies must be put in place to prevent *hoarding* that would most probably lead to periodic market *corner-*

ing and high volatility. Since it is an open-loop system, once the currency is consumed on the provider resources, that currency gets thrown away. This paradigm fails at implementing realistic economic principles where currency must be considered as a scarce and precious commodity, as computing resources are, to be exchanged and, most importantly, liquid.

Others have tried to limit *strategic* behaviors arguing that such users inevitably lead to a *Tragedy of the Commons*. What *Strategyproof Computing* advocates may get wrong is that commodity markets actually rely on speculators to limit volatility as they provide, since they massively outnumber actual resource providers and consumers, the fundamental and necessary market entropy. The inefficient use or waste of resources by users may not, in fact, be a negative consequence of their selfishness, but rather reflects fundamental flaws in the pricing scheme, market-clearing algorithm or currency allocation policy.

For the Grid to become pervasive, the incentive issue will have to be addressed adequately. This undertaking is fundamentally an exercise in economics or incentive engineering where policies and mechanisms to put them in place are tightly related [92]. Therefore, policies, mechanisms and implementation will have to be designed altogether following an iterative development process rather than a waterfall *policy to implementation* approach. The development path taken by existing grid economy frameworks is probably worse as they provide an implementation lacking sound economic fundamentals.

Ensuing the illustration of the various shortcomings inherent to actual Grid models and systems, the ideal Grid resource sharing framework would have to implement the following:

Unrestricted market dynamics

Although this requirement might seem vague at first sight, it is by far the most important principle as it allows strategic behaviors, hedgers and traders, consumers and providers influence on the market and limits the use of consumer biased mechanisms like Vickrey auctions. It addresses the fundamental incentive issue for both providers and consumers while bringing accountability on both sides. It also enables the implementation of a system where operating costs can be injected and valued. In addition, by definition, *unrestricted market dynamics* are fundamentally decentralized and therefore can foster very scalable implementations.

Hybrid futures/spot commodity trading model

As it will be presented later in further detail, Grid resources and the way they should be traded are comparable in many ways to conventional futures. However, their production is much more similar to energy commodities that are traded "on the spot". While a market-clearing house may not be the absolute right answer for HPC resource exchange, clearing the market may prove to be a valuable design objective and mechanisms encouraging it should be implemented.

Resource sets expression

Scientific applications on supercomputers have requirements that go far beyond CPU performance. Therefore, consumers must be able to express their needs for RAM, storage, networking, software, etc. Similarly, resource providers possess compute nodes with extensive features sets with respect of the aforementioned components. Both should then be able to express their requirements or components to be asked or offered on the market in a far richer syntax.

Price assessment and anticipation

As requirements and components sets are to be very descriptive, this poses a major hurdle for market trading as the comparables become very hard to find. Stock markets rely on financial data like earnings per share (EPS) or price/earnings ratio (P/E) to analyze the fundamentals of a position. Commodity exchanges trade "at the market" and therefore pricing is provided *de facto*. A Grid exchange system implementing a rich expression syntax must then be able to provide the pricing assessment to both users and providers in order to match traders and limit volatility. This price assessment scheme must not only consider the various resources traded in sets but their relative supply and demand over time, therefore providing the desirable *price anticipation* concept mentioned previously.

Closed-loop currency policy

A futures market is fundamentally a bartering system between commodities for specific timeframes, should they be material goods or currencies. Fictional currencies express the need for a liquid and un-perishable trading counterpart. They find their value in the confidence traders have in them and as they can be weighted against desirable goods, currencies are indeed a very precious instrument. Therefore, a Grid exchange system must reflect these fundamental currency properties. There should be ways to reuse consumed currencies for the acquisition of other goods. This Grid currency liquidity requirement has been totally forgotten until now and must be implemented. While the bootstrapping of the market and the initial currency allocation may be handled by a governing entity, the system should be allowed to re-feed itself for the most part.

This thesis presents an innovative Grid resource sharing system attempting to answer most limitations of existing models and frameworks while addressing these five fundamental requirements. Sitting across computer engineering and economics, it is, first of all, a rather philosophical exercise in incentive engineering and finance, and, because good concepts must be defined with respect to real life implementation, a system engineering undertaking to identify the major design aspects to follow in order to build a working framework.

1.9 Thesis Outline

Chapter 1: Introduction

This first chapter introduced High Performance Computing (HPC) and the Grid from a rather large perspective to the narrower field of utility computing and the Grid economy. It covered the actual architectures to be eventually traded on such a resource exchange system and outlined the various features of the widespread generic Grid frameworks. It then described the foremost shortcomings of existing Grid exchange models sculpting the requirements for a much more efficient system. These requirements serve as a foundation supporting the following chapters.

Chapter 2: Finance 101

Intended primarily for computer engineers and scientists, this chapter introduces some essential economic and financial principles to understand subsequent material. Auctions models, futures trading, spot markets, fundamental and technical analysis are covered with a level of detail relevant to Grid resource market exchange scenarios.

Chapter 3: An Innovative Market-based Grid Sharing Model

This chapter presents the proposed model for HPC resources trading alongside its underlying algorithms and innovative features. It details how resource sets are defined and traded and how, from statistical supply and demand data, resource market indices are computed. From these indices, this chapter develops the key pricing assessment functions both consumers and providers would use to steer their market positioning.

Chapter 4: Model Simulation and Analysis

As a real world deployment would have been an undertaking too extensive to be completed within a Ph.D. curriculum, this chapter simulates the model and pricing functions using meaningful stochastic distributions of requirements and components in order to mimic behavioral economic patterns of users. Equilibrium states are then reached and studied under different market scenarios.

Chapter 5: From Infancy to Adulthood

Building on the 2 preceding chapters, macro-economic considerations as well as market bootstrapping related issues are covered throughout these pages. Questions surrounding quality of service, trust management and trading strategies are then explored. Interesting financial concepts emerging from the proposed Grid Exchange paradigm are then envisioned, providing leads for further studies.

Chapter 6: Conclusion

Revisiting the key concepts presented throughout this work, this chapter emphasizes the major contributions. As this thesis lays foundation for many subsequent research projects, this chapter also points out a few interesting ideas worth further exploration.

Chapter 2

Finance 101

This chapter provides the financial foundations to understand and appreciate the Grid economic model presented in this thesis. Commodity and spot markets, technical and fundamental analysis as well as auction models are quickly covered. The goal is not to provide an in depth analysis of each of these topics as many books already do that, but to outline the relevant fundamental notions useful to computer engineers and scientists. A reader with strong financial background may want to skip directly to section 2.5.

2.1 Commodity Markets

2.1.1 Overview

The word "commodity" comes from the french word "commodité" that means "convenience". Indeed, commodities traded are not only the very fundamental life essentials but also the ones supporting the quality of life we have been used to. Rice, corn, wheat, soybeans, beef, pork, cocoa, coffee, sugar and even orange juice represent the essence of commodity exchanges as everybody eats, everyday. Agricultural commodities now also include cotton, oats and soybean derivatives. Also, precious metals (gold, silver, platinum, palladium), rare metals (germanium, cadmium, molybdenum...) as well as industrial metals (copper, lead, zinc, tin, aluminum, nickel and even recycled steel) are all traded around the world in various exchanges. Energy commodities like crude oil, natural gas and uranium are also driving large volumes on exchanges. As commodity exchanges are so widespread, any essential good one may think of is probably already traded in some exchange around the world as long as it can

be "commoditized"[1] and able to support enough trading volume in the long run.

Commodity trading all began in the middle of the nineteenth century as businessmen started organizing market forums to facilitate the buying and selling of agricultural commodities. While attending these meetings, farmers and grain merchants agreed on quality standards, trading units (quantity of goods), delivery timeframes and established rules of business. By the end of the century, more than 1600 exchanges had sprung up at major railheads and ports. In the early 20th century, centralized warehouses were constructed in major urban centers like Chicago, a city that had a strategic location close to the agricultural plains while sitting on the Great Lakes. As of today, Chicago is still one of the foremost commodity exchange of the 30 major exchanges in the world.

Since 2003, the Chicago Climate Exchange (CCX) has been trading six greenhouse gas emissions in order to reduce their propagation in the atmosphere. The European Climate Exchange and the Intercontinental Exchange have since joined the environmental battle. Although not widespread thanks to countries not willing to support the Kyoto agreements, carbon emissions futures introduce a very interesting incentive to protect the environment; as big corporate players, cities and governments have to buy the right to pollute the planet. Within such markets, after defining global emissions yearly targets, traders around the world would drive prices higher as the targets would get lower, year after year. Hence, large corporations could then save money by modernizing and reducing emissions since polluting would not be free anymore.

Commodities used to be solely traded in the *Pits* where floor brokers open-outcried[2] sell and buy orders. Since the early 2000's a lot of these trading floors have been transfered to electronic exchanges that are faster, cheaper, more efficient for users, less prone to manipulation by brokers and can run around the clock.

The daily dollar volume of commodities traded around the world is simply astonishing. In fact, the total dollar value of commodity traded is greater than the dollar value of stocks traded plus the dollar value of mutual funds traded daily in all exchanges in the world. Even more astonishing, millions and millions of dollars change hands everyday yet there are less than a hundred commodities traded. While most people ignore commodity trading, fortunes are made and lost in a matter of minutes on commodity markets. In fact, between 75% to 90% of traders loose on these markets while the remaining others (10% to 25%) make big money. Commodity trading is all about limiting losses and leveraging profits.

[1] meaning "standardized" in the way that any delivery from any provider is considered the same

[2] shouting and using hand signals to transfer information

2.1.2 Futures Contracts

Commodity markets are also called *futures markets* since the commodities bought and sold are to be delivered at some point in the future. A *futures contract* is the basic instrument of exchange in the futures markets. Each contract is for a pre-defined quantity of some commodity or financial instrument, the contract size. It also stipulates how the price is quoted and the minimum price fluctuation. Since the contract size is standardized for each commodity, the minimum trading unit is one contract. For instance, a wheat contract is 5 000 bushels, a gold contract is 1 000 ounces and a lumber contract is 160 000 board feet. On each trade, a commodity trader would then send buy or sell orders for X contracts of a commodity Y.

A contract is a legally binding agreement for delivery of a commodity at some time in the future. Although there is no paperwork involved, a commodity trader is nevertheless entering a contractual obligation that can be met in 2 ways. The first option is making or taking delivery of the actual commodity. That option is by far the exception on commodity markets as it represents less than 3% of trades[147]. 97% of trades are called *offsets* where a trader is offsetting a position by doing the opposite sale or purchase of a specific quantity of a given commodity prior to the expiration (and delivery) of a contract. Since contracts are standardized, this can be done easily.

For every commodity, contracts are also very precise on the quality of the goods to be delivered. For example, silver contracts require silver to be delivered in ingot form and 99.99% pure. Therefore, silver cannot be traded in any other form or at any lower quality on futures markets. The only negotiable aspect of a contract, for any commodity, on any exchange, is price.

To start trading a given commodity, one needs to know how prices are quoted for that commodity. For example, cattle and hogs are quoted by cents per pound, grains in dollar and cents per bushel and gold in dollars and cents per ounce. The minimum price fluctuation of a contract, or *tick*, multiplied by the contract standardized size, leads to the minimum dollar value fluctuation of a contract. For example, the minimum price fluctuation for corn is 1/4 cents per bushel; since a corn contract is for 5 000 bushels, the minimum dollar value fluctuation of a corn contract is 12.50$.

Most contracts implement daily price limits: the maximum amount a market can move above or below the previous day closing price. Soybeans, for example, are limited to 50 cents and therefore if the markets closes at 5.40$ a given day, it cannot close higher than 5.90$ or lower than 4.90$ the following day. These limits exist to control the volatility that may arise on a market following some dramatic event that could cause the market to plummet or skyrocket. Hence, a market may become *locked limit up* when there is an overabundance of

buyers versus sellers at the *limit-up price*, or *locked limit down* when there is an overabundance of sellers versus buyers at the *limit-down price*.

Every individual market has specific trading hours set by its Exchange. Cattle, for example, is traded from 9:00 AM to 1:00 PM on the Chicago Mercantile. As more and more exchanges become electronic, trading hours may become obsolete in the very near future.

Last but not least, every contract is for a specific delivery month, and some commodities can be only be traded for a few delivery months in a year. Wheat on CBOT, as an example, can be traded for March, May, July, September and December of each year. The last trading day for a futures contract is the last day of the delivery month, a situation no speculator would want to find himself in. On the futures markets, a trader must take great care not to keep a position up to delivery, unless he's a producer or consumer of the specific good. When that happens, the trader receives a warehouse bill stating that X amount of the commodity is stored somewhere and instructions for delivery are required (a trader would not get delivered 5 000 bushels of wheat in its backyard the very first day of the delivery month). He then must find a buyer for that commodity or incur warehousing expenses while trying to sell in the next trading month (which can be 3 months away). For metals, that may not be a big problem (while still expensive), for live cattle, it may pose some more challenging logistical problem. The other way around, a speculator should not find himself in a position of delivering a commodity he does not possess, as finding 1 000 ounces of gold may not be a trivial endeavor.

On the London Metal Exchange (LME), where most industrial metals are traded, *prompt dates* are used instead of delivery months. The reference pricing is for the 3 months contract, it is also the most active. In practice, it means that if you buy or sell a 3-month silver contract on November 7th, you have a February 7th contract. In principle, you would liquidate your position at the price of the *cash* or *spot* contract at the expiration, on February 7th. You can, however, liquidate at any time and the price is then defined by an interpolation of the 3 months and the spot contract at that time. Spot markets can therefore be seen as an instantiation of futures exchanges, only with very short term. For instance, the spot FOREX (foreign exchange market) trades currencies with a 2-day delivery date. Energy spot markets, like electricity, can have delivery within a few minutes.

To summarize, *futures contracts* specify the quantity, quality and delivery period for any given commodity. Any commodity exchange also specifies how the price is quoted, the minimum price fluctuation, or *tick*, and the value of that fluctuation. Some markets implements pricing limits to control volatility and most conventional *pits* restrict trading hours.

2.1.3 Hedgers vs Speculators

Both buyers and sellers that are physically involved in the actual delivery of a commodity are called *hedgers*. In general finance terms, *hedging* is a method for limiting risk by taking an opposite position on the instrument to be traded. For instance, because a large coffee roaster and distributor has to sign contracts with supermarkets several months in advance for delivery, he might want to assess his raw coffee costs (green beans) for that forseable future in order to define it's reselling price. On the other end, the coffee producer in Nicaragua would want to establish the market price for the same months in order to plan production ahead. By protecting the price of the commodity to be delivered, both want to *hedge* the coffee market, but in opposite directions. While hedgers enjoy price protection, they incur the responsibility of either producing or buying the good traded, by contractual means.

In practice, the commodity producer will do a *short hedge* in order to protect the value of an inventory or future production. For example, if an oil producer extracts 100 000 barrels per month, and the cash price is 70$ a barrel in September, he may be willing to hedge his November production, as the crude oil November future sells at 80$. He therefore goes short for 100 crude oil November contracts (each contracts is for 1 000 barrels) and protect his production at 8 M$. In practice, most producers won't actually deliver that very contract. It is, in fact, a form of insurance. If the spot price for crude oil ends up below 80$ in November, the producer will compensate the loss on the cash market with the profit realized on the futures market. The other way around, if the spot price is higher than 80$, the cash market profit will cope for the loss inherent to the futures contract. The same mechanism applies for long hedges and commodity buyers. For manufacturing companies that want to maintain stable their product prices for a few months, hedging alleviates the need to operate large warehouses to pile up raw material.

In general terms, the *basis* refers to the difference between the market price at the exchange and the price at some other location. For example, if a 160 000 board feet (one 73 foot rail car) CME lumber contract calls for delivery in the Chicago train station, and the transport cost for that rail car from Chicago to Boston is 2 000$, then the basis in Boston runs at "plus 182 points" (1 point = 0.10$ / 1 000 bd. ft. = 11$ / contract).

The *basis risk* refers to the difference between the cash price and the futures price. A *widening* of the basis (cash prices have fallen to a greater degree than futures) would imply a *basis loss* on a short hedge and a *basis gain* on a long hedge. Correspondingly, a *narrowing* of the basis would lead to a basis gain on the short hedge and a basis loss on the long hedge.

Since all commodities are not necessarily traded in exchanges, *cross-hedging* is a widespread practice where, for instance, airlines hedge their jet fuel costs (or trucking companies their

diesel costs) by going long on heating oil futures, as the cash market for these are closely related. The variation between the traded commodity and the commodity useful to the cross-hedger is also called the *basis risk*. For instance, heating oil cash prices may climb faster following several very cold days as jet fuel stocks remain stable. But by any means, the basis risk is always much smaller than the *flat price risk*, the price risk without hedging.

Although without hedgers there would not be any commodity market, less than 3% of the contracts bought and sold on the various exchanges are actually fulfilled. Hence, more than 97% are *liquidated* before their expiry. Liquidating a position is done by trading an opposite contract before the delivery period, or *offsetting* the position. *Speculators*, or *traders* are thus driving the futures markets volume. To make money, traders either buy low and sell high (*going long*) a futures contract or sell high and later buy low (*going short*) that same contract.

By assuming the risk hedgers want to avoid, speculators have a stabilizing effect on prices. Because they tremendously outnumber hedgers, speculators build the essential market volume and provide the fundamental liquidity[3]. In fact, although most traders are never involved with actual delivery (unless they make a mistake) of goods, without them, exchanges would suffer from high volatility and low liquidity, 2 dramatic conditions that would prevent futures markets to be sustainable.

2.1.4 Longs, Shorts, Margins and Leverage

Buying something at a low price and reselling it for a higher price is a concept everybody understands. In the financial vocabulary, it is referred as taking a *long position*. Here is an example on how to make money going long in the futures markets:

1. Trader Paul buys a 5000 bushels contract of September 2007 wheat on COMEX at 6.25 1/4 a bushel on January 15th, 2008, an investment of 31 262.50$[4].
2. On March 15th, 2008, a report from the national weather forecasting bureau predicts a hot and rainy summer in the plains of North America, due to a consequence of el-niño.
3. After opening at 6.35 1/2, the wheat market locks down the limit at 5.85 1/2 fearing an oversupply of wheat in the summer.
4. Paul, being an intrepid speculator (!), believes that the feared oversupply will not materialize and keeps his position.

[3] a "liquid" market is a market where sellers find buyers and vice-versa

[4] in theory. In practice, not really, as Paul would buy on its *margin*

5. On August 15th, 2008, the feared el-niño effect never materialized and producers, afraid of being forced to dump their wheat production on Africa, had put little pressure on the crops all summer long and the oversupply became a shortage of wheat.

6. Paul liquidates his position that very day selling his contract at 6.55 3/4, *going long*, and making a profit of 1525$.

This example could be easily understood by a kid running a roadside lemonade counter. But a lot of financial markets, and especially futures, can work the other way around: by first selling a contract at a high price and later buying the good to fulfill delivery at a lower price. Bill Gates, after selling DOS to IBM for several millions, went to a backyard programmer to buy it for 50 k$. That may be the most famous (and profitable) *short position* ever took by anybody! More generally, on the commodity markets, a short position would look like this:

1. Speculator Richard sells 4 contracts (40 000 pounds each) of October 2008 live-cattle on CME at 98.275 on November 27th, 2007. A live-cattle contract is expressed in points and 1 point equals 0,01$ / 100 pounds, or 4$, contract size. The minimum price fluctuation is 0.00025$ / pound, or 2.5 points, or 10$. Therefore, he just went short of 157 240$.

2. On July 16th, 2008, the FDA reports a bovine spongiform encephalopathy outbreak in Montana, the feared "mad-cow syndrome".

3. Japan instantly closes its meat US imports and the live-cattle goes down the limit.

4. The next day, the market opens at 95.100 and goes down the limit, again, at 92.100

5. In that over-abundance of sellers, Richard covers his shorts at 92.100, or 147 280$, making a profit of 9 960$.

Leverage is one fundamental aspect of commodity trading that makes it so different of conventional stock markets. To buy 100 shares of company XYZ, trading at 97$, one must cash out 9 700$. On commodity markets, because the traded entity is inherently worth something, a trader is not required to provide upfront the total dollar value of the contracts traded. Indeed, the *clearinghouse* only requires a security deposit, called *margin*. Every exchange has different margin requirements but typically, one can enter most futures markets with margins between 5% to 10% of the traded contracts dollar value.

Although the margin might seem a quite interesting concept for the newbie, it is a double-edged sword, as it does not, in any way, limit the amount of loss one can incur. On the stock markets, one can invest 25 000$ in a company, if that company goes bankrupt, that investor

looses 25k$, but nothing else. In contrast, with the leverage the margin provides, one can either "make a killing", or, very painfully, get "killed".

Revisiting the prior long example on wheat trading, we'll see how important leverage can be. For CBOT wheat contracts, the *initial margin* is 2 025$. In September 2007, trader Paul has placed 2 500$ in his account and then committed 2 025$ for his 31 262.50$ wheat contract, or 6.5% or the contract dollar value. During the mid-March crisis, his contract was worth 29 275$. This paper loss of 1 987,50$ did indeed become very real when Paul received a *margin call* from his broker because CBOT wheat contracts have a *maintenance margin* of 1 500$. While the initial margin is required to enter a position, the maintenance margin defines the minimum amount that has to remain, at all time, in the trader account. Therefore, at 2 500$ - 1 987,50$ = 512,50$, Paul was 987.5$ under the required maintenance margin, and his broker required him to send a check of 1 512.50$ to cover the initial margin. This is an important thing to remember, when a trader account goes below the maintenance margin level, he must cover the initial margin amount. By the time the wheat market recovered to Paul's initial position, he had the right to reclaim his margin covering check. When he got out in august, selling his contract for 32 787,50$, and making a profit of 1 525$, Paul also recovered his initial margin in order to enter other positions. In retrospective, Paul made 1 525$ by investing 2 025$ in less than a year, 75% revenue. Considering the wheat market only went up 5% (from 6.25 1/4 to 6.55 3/4), such a profit is just astonishing. Again, 75% in less than a year, that's sweet leverage. If Paul would have gone out in March, he would have lost 98% of his initial investment; leverage, again, but the other way around.

2.1.5 Brokers, Clearing Houses and the Pits

As for stock markets, there are several kinds of brokers on the commodity markets but they can generally be sorted to 2 distinct categories: discounters and full-service brokers. Typically, discounters, on online brokers, charge lower commissions to execute orders than full-service brokers. They also offer less service. For a self-directed speculator trading on electronic markets, discounters may be the right option. For a novice commodity trader, full-service brokers provide advice on which month to enter a position, how to limit the risk involved and avoid mistakes like overtrading and not maintaining a sufficient margin. For a commodity traded in the pits, large firms also tend to hire better floor brokers (bigger, taller, more aggressive or just more efficient); this can be a substantial advantage in a volatile market as better floor brokers tend to limit *slippage*. Slippage is a term referring to the difference between the price your order gets filled and the market price (last trade). In a volatile market (especially when you might want to trade), a good floor broker is worth many times the higher commission rate.

While most commodity *Exchanges* around the world have been replaced by electronic trading, some conventional *open-outcry* floors, or *pits*, still remain in the US. Following Exchange rules, only the lowest seller is allowed to advertise his offer. Similarly, only the highest bidder is allowed to yell out his bid. Therefore, no one can bid lower than the highest bid and no one can sell higher than the lowest offer. The last trade price becomes the market price, a mechanism known as *price-discovery*. When buyers are more aggressive than sellers, prices go up and inversely, when sellers are taking buyers bid more rapidly, prices go down. This is how prices are determined, which may fit, or not, someone's investment plan, but, like the old adage says, "the market is always right".

The Exchange is made up of several member firms like large financials, national-scale banks and major commodity producers and buyers. Each member is required to deposit 10% of the firm's capital in a guarantee fund. If a member goes bankrupt, all other members of that exchange are required to cover that member's losses on a pro-rata basis. On top of that, the clearing member must send all required margins for its customers contracts. In the event of a very volatile market, these margins can be adjusted by the Exchange and the clearing member is required to contact its customers to adjust them accordingly. Finally, if a customer fails to pay for his losses, the clearing member is obligated to cover them before the end of the trading session, and deal with his client afterwards. With these many levels of fallbacks, Exchanges are highly dependable and provide the appropriate confidence levels to all traders.

At the end of each trading session, the *clearinghouse* debits and credits trader accounts with their respective losses and wins. While many contracts change hands everyday, money does so too, and any trader can be assured to see his account balance adjusted accordingly. The clearinghouse is also responsible for delivery in the cash market. The *first notice day* for a contract is the first day a contract can actually be delivered. For most commodities, it is the first trading day of the active month. For some others, it is the last trading day of the previous month. The last day shorts can make delivery is also specified by the Exchange, it is usually several days before the active month end. Worth noting, it's the shorts responsibility to actually execute delivery. Also, longs are fulfilled in their order of appearance, oldest first. In practice, if the cash market is higher than the pending contracts in the first notice day, most shorts will wait to deliver. The other way around, delivery probability increases. The bottom line: don't trade the active month unless you are an experienced futures trader.

2.1.6 Order Types

The most conventional and better understood order type is the **market order**, to buy or sell "at the market" price. When such an order comes in, the floor broker must fill it immediately

at the next best price. In a double auction scheme, the advertised *bid* is the highest price a buyer (or several buyers) is (are) willing to pay. The *offer*, or *asked price*, is the lowest price a seller (or several sellers) is (are) willing to sell their contracts. Unless the floor broker is willing to forward the difference, a client would typically buy at the offer and sell at the bid. This difference, also called the *bid-to-offer spread*, is an inherent cost to a double auction scheme involving brokers and is otherwise negligible on most healthy exchanges. While by posting a market order a client cannot specify a precise bid or offer, he can have a very good confidence that his order will be filled, and within a reasonable price range.

A **limit order** specifies the worst-case pricing a client is willing to enter the market. Following such an order, the floor broker can't pay more than the limit on a buy order and sell for less than the limit on a sell order. Typically, a limit buy order would be like'buy at 100 or better" (100 or lower). Suppose the market is trading at 120 when the order comes in, then goes down to 100 and goes back up 150. A client would then expect to be filled at 100; unfortunately, that may not be the case if the market only "touched" the 100 mark and climbed back up right after. That client may have been lucky but most probably, some other order got filled and that client just missed entering the market. Therefore, while limit orders secure a price fill, they do not, in any way, assure the order will actually get filled.

The **stop order** is frequently used to restrain the loss on a position that would not be working the way it should, it is also referred as a *stop loss order*. A stop order is an order that activates and becomes a market order once the stop point is touched. Intuitively, a *buy stop* is posted by a short above the market and a *sell stop* it put up by a long under the market. For example, a trader could enter the oil market long at 80 and sell at 75 stop. As long as the market goes up, the stop order doesn't get activated. If the market goes down and touches 75, the floor broker receives a market order to sell. That order could get filled at 74.95 or even 75.05 (if the market ticks up right after touching 75) but essentially, the trader loss is then limited to the stop.

A common instrument used to lock in profits is called the *trailing stop*. In the previous example, lets assume the market moved up to 86, using a trailing stop, the trader would have moved his stop to 81, protecting his initial investment and covering commissions. If the market went further up to 95, the trailing stop would have moved to 90, protecting profits.

Stop orders can also be used to enter the market. For instance, if the market trades crude oil at 92 and if a speculator believes that if crude oil reaches the 95 mark, it will continue going up, he could place a buy stop at 95. As long as crude oil stays below the 96 mark, he does not enter any position. Once his order gets executed, he could place a sell stop at 92 to make sure the market really moves the intended way. Sell stops can also be used to enter a new short position under the market.

A *stop limit* is a stop order that activates a limit order instead of a market order. If a market reaches a predefined mark, the stop limit order must be executed at that mark, and that mark only. This type of orders see little application as they defeat the purpose of a stop order in a fast moving market.

The **market if touched order**, or MIT, is the mirror of a stop order. While buy stop orders are placed above the market, buy MITs are placed under. Similarly, sell stop orders are placed under the market while MIT sell orders are placed above. They can be used to take in profits at a predefined mark or to enter a market when a speculator thinks it is hitting an inflection point. In our previous example, if the speculator believed the market would back down to 90 once it touched the 95 mark, he would have placed an MIT to sell at 95.

One last type of order worth noting is the **One Cancels the Other**, or OCO order. With such an order, a trader specifies the upper bound (profit taking) and the lower bound (stop loss) when entering a position. For example, a trader can go long gold at 400 instructing his broker to sell at 425 or 390 stop; one cancels the other. If he had placed 2 distinct orders and the gold market went up to 425 first, then down to 390, he would have took profits at 425 but would have later found himself in an undesired short 390 position.

2.1.7 Options

In the financial markets, *options* are an instrument where the buyer acquires the possibility, or "option", of entering a position (long or short) at some point in the future. Because entering the position is at the buyer's discretion, depending on whether or not *exercising* the option would be profitable, the option seller or *writer* charges a premium when issuing the option. For the buyer, options provide unlimited profit with limited risk. This surely sounds great, but it comes with a price: the premium.

In practice, options are traded on the same markets then commodities and they can be converted in the underlying futures contract; this is called the *right to exercise*. Based on futures contracts, they include all the specifications: quantity, quality, delivery and a *strike price*, the pre-defined price at which the option can be exercised. The strike price for the various options are set by the Exchange in a manner similar to conventional futures contracts.

A *bullish*[5] trader would acquire a **call** option and receive a long position, if exercised. Similarly, a *bearish*[6] trader would buy a **put** option to get a short position, once exercised.

[5] speculator thinking the market will go up
[6] speculator thinking the market will go down

American options are Exchange-based, they are the most widespread and can be exercised anytime time prior to expiration. *European options*, the other style, can be exercised only at expiration date; they are generally cheaper and found most of the time for OTC (over the counter) options.

As we have just seen, options bring an unlimited profit and a limited risk to the buyer. While selling an option brings unlimited risk, options markets are generally in favor of sellers. On a stable market, if the market goes the opposite direction or if the market moves to the option strike price without touching it, the option writer is making the profit. This is why professional traders, that believe they have a fairly good understanding of the market dynamics, represent the vast majority of options writers.

Options can also be a very useful tool for hedging, as they can guarantee revenue without limiting the profits. Revisiting the example of section 2.1.3, instead of locking his oil production revenue at 80$ a barrel, the oil producer could have bought a put option at 80$ strike minus 2$ per barrel. By buying a put instead of going short, although the hedger incurred the premium cost if the barrel sells for 82$ or less in the cash market, the option premium can be calculated in his expenses long in advance. In addition, while "locking in" a 78$ selling price, the premium just bought the producer an unlimited profit possibility if the oil barrel skyrockets.

Synthetic options are a well-known option strategy where a long contract is covered by a put (synthetic call) and a short contract is covered by a call (synthetic put). The main reason not to enter directly in a put or call position and going synthetic comes from the enhanced flexibility hence brought, especially under high-volatility. By entering a futures position and the reverse option position at the same price, the speculator is always capable of "washing-in" the option in the futures market. If a market goes limit-lock several days in a row, a synthetic option gives an added benefit: the option to wait before moving (in case the market would rally up in your direction). If the market goes the futures position, the option would become irrelevant, and could just be seen as a form of insurance the trader was willing to pay. Nevertheless, if the market breaks down the other way, such options could rise from the dead and become valuable again. To limit the risk involved with option selling, writers can *cover* selling calls by going long on the market and selling puts by going short.

Spreads are also a widespread technique in the options market. They are more flexible than futures spreads because, for instance, a *vertical spread* can be constructed by buying 2 call (or 2 put) options for the same month, but at different strike prices. *Straddles* are options spreads involving both a put and a call at a given strike. *Strangles* are the same, but with different strike prices. *Ratio* spreads are variants of straddles involving unequal number of calls or puts for the pre-defined strike price.

2.2 Fundamental Analysis

The previous section outlined the major aspects surrounding commodity trading in a level of detail relevant to understand this thesis contributions. Many books on futures and options trading have been published and references may be consulted at the end of this chapter. While the previous section explained *how* futures are traded around the world in the Exchanges, this section and the following will describe *why* traders enter positions, whether they analyze the market fundamentally or technically.

Fundamental analysis is probably the most respected and maybe even "worshipped" analysis technique, technical analysis being the much less venerated brother. As both methods try to predict future market movements, fundamentals refers to global factors that impact supply and demand and therefore, pricing. In the following, we will outline some important factors that affect the financial, energy, agricultural and metal commodities.

Financial futures drive the largest volume on Exchanges, a long way ahead of physical commodities. They include interest rates, stock indices and currencies. They influence each other and are closely linked to many other external factors. *Inflation*, mostly reported through the *consumer price index* and the *producer price index* is one of the major actors affecting interest rates, as the cost of borrowing money must be higher than inflation for the lender to make any profit. Most of the time presenting an inverted relationship to inflation, *unemployment* greatly affects interest rates as it closely reflects the global demand for credit. The *balance of trade*, representing the ratio between imports and exports has a major influence on currencies and interest rates, since a net exporter will earn more foreign capital than what is spent externally. The Federal Reserve in the United States (or equivalent in any industrialized country) represents the foremost political lever on the economy. Through its *discount rate* (the rate at which the Fed lends money to banks), a government can influence the economy in slowing or accelerating the access to credit in order to slow inflation or stimulate the economy. Many other aspects like retail sales, housing starts and gross domestic product influence financial commodities. While most interesting from the economic standpoint, they nevertheless are high abstraction derivatives that find few equivalent in the bootstrapping of a computational economy.

The most preeminent energy commodity is by far the crude oil contract traded on NYMEX. It represents 20% of the world's entire trade; money and its derivatives being the only bigger market. Heating oil and unleaded gasoline, both crude oil derivatives, are quoted in contracts of 42 000 gallons and follow crude oil prices. Natural gas is the emerging energy commodity as more and more homes and industries are using it as a heating source. Energy markets are influenced by environmental factors such as weather and seasonality, where harsh winters see

prices rising abruptly. Politics also have a significant impact on crude oil pricing whether it depends on OPEC (Organization of the Petroleum Exporting Countries) production directives or the various instability issues that may arise in the middle-east.

Agricultural futures fundamentals are easily understood by everybody since they have been around forever. For obvious reasons, weather and seasonality are even more influential on crops pricing and just the shadow of a drought or flood can move the market limit up. Government policies may also affect crops pricing through protectionist measures or acreage restrictions sometimes used to protect against a market collapse under specific export constraints. Last but not least, the Chicago Board of Trade distributes a weekly report about the quantity of corn, wheat, soybeans and oats in the CBOT licensed elevators; this weekly report sometimes moves the market to the limit if traders fear a *squeeze*, where shorts can't find enough grain to deliver.

Meat markets regroup feeder cattle, young steers and heifers of 600 to 800 pounds, exchanged in contracts of 50 000 pounds, that must be fed up to 1 000 to 1 300 pounds, where they can be sold as live cattle, in contracts of 40 000 pounds. The pork industry, in contrary to its beef counterpart, feeds hogs from birth to the ready to slaughter product: lean hogs. The other market for pigs is frozen pork bellies, the raw material for bacon. Meats are largely influenced by consumers tastes, which are somewhat linked to seasonality, where holidays see rising demand for ham and summer barbecues drives the beef demand. Also, the slight mention of the *mad cow syndrome* or any such disease can induce high volatility on the markets. Finally, feeder costs are to be accounted since they lead to *accumulation*, under low feeder cost, and *liquidation*, following rising corn prices.

Referred as the *softs*, sugar, coffee, cocoa, cotton, orange juice and lumber are mostly driven by their stock to usage ratios and the yields per acreage of the various production sites.

Metals can be divided into 2 distinct categories: precious and industrial. Precious metals include gold, platinum and silver. Back in 1816, Great Britain decided to support its currency by an equivalent gold reserve, which forced all other industrialized countries to do the same in order to support their own currency against the British pound. In 1971, Nixon cancelled the US dollar convertibility to gold, which turned the currency into a purely speculative instrument, that must now be managed by controlling the markets volatility and confidence in it through interest rates and Treasury Bonds. Gold is still a refuge instrument for a lot of investors when the markets become unstable. Although mainly used in the industry, platinum worldwide production is under 100 tons annually. It is therefore considered as a precious metal and usually trades at a higher price point than gold. Because Platinum production is very limited, the 2 major producers, located in South-Africa, can pretty much set the market price. Silver, an hybrid precious / industrial metal has long been used in photographic film on

top of many other applications; the emergence of digital photography could lower its course but the rising of India, where silver is the precious metal of choice, will probably account for more than a global worldwide move to digital cameras. Industrial metals include copper, aluminum, zinc, nickel, lead, tin and palladium. They are mostly influenced by economic activity in the developed nations. LME and COMEX inventories are also an important factor as they reflect the production / consumption ratio at any point in time. Finally, wars drive metal prices, especially copper, as they are much needed in arsenal fabrication.

On the stock markets, fundamental analysis relies on the same global factors than commodities: interest rates, inflation, etc. In addition though, a good fundamental analyst will pick stocks based on the financial statements of the companies, looking at profits versus earnings ratios, anticipated growth, return on equity, stock dilution and earnings per share. These are very important leads in deciding to enter or leave a position. This data provides an essential comparison tool between any given stock and a comparable company or the industry average. Warren Buffet, also called the *Oracle of Omaha*, made his fortune in being able to pick the undervalued stocks, based solely on fundamental data.

2.3 Technical Analysis

On the commodity markets, there is no such thing as *insider trading*[7] and nothing could prevent a large cocoa producer/consumer like Nestlé to move on the futures market following an internal crop report from their plants in the Ivory Coast. Therefore, an external speculator is largely disadvantaged since he does not have a man walking the cocoa fields and sending him reports. However, if a report comes in to Nestlé's headquarters in Switzerland stating that the crop is getting affected by the "witch's tail disease", Nestlé's purchase office people will probably get busy at hedging their risk in buying cocoa futures in London and New-York. Since Nestlé is no small player on the cocoa market, they will leave "footprints in the sand".

Technical analysis is all about watching for these events, even traces, that could be signs the market is moving either way. Charts, open interest, volume, resistance and support, moving averages and many other tools are used in trying to guess where the market will move. This section quickly covers the mainstream predictors.

Although good technicians, or *chartists* employ a great number of fancy tools, technical analysis (TA) is really just about studying the supply and demand in a market and trying to

[7] *insider trading* is forbidden on the stock markets and closely watched by the Securities and Exchange Commission (SEC), as it involves the buying or selling of a security by someone who has access to material but nonpublic information; typically a firm director or one of his family member and/or related brokers.

predict if the *trend* will continue or break. In looking at past prices and volume, technicians do not try to assess a position intrinsic value, as fundamentalists would do, but instead try to match patterns on current price charts to estimate future activity.

2.3.1 Trends

A *trend* is simply the general direction a market is headed for any given stock, commodity or security, as shown in Figure 2.1 a). However, trends are not always that easy to comprehend, as shown in Figure 2.1 b).

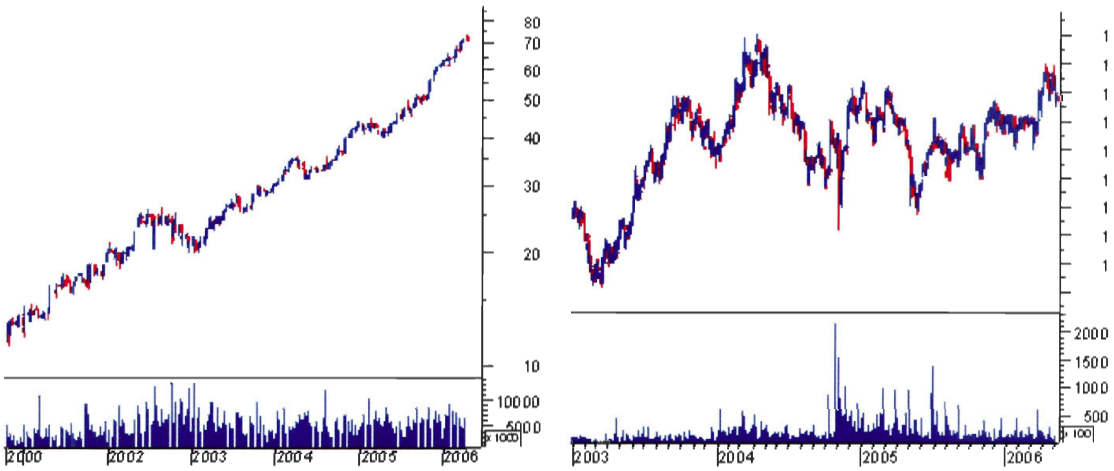

Figure 2.1: *a) An easily understandable trend and b) not quite [13]*

Therefore, an *uptrend* would be defined as a series of higher highs and higher lows while a *downtrend* would be a series of lower lows and lower highs. If highs and lows on a chart stay roughly horizontal, some say the trend is *sideways* while some others consider there is just no trend. When a low falls below the previous low on an uptrend, this is called a *trend breaker*, an event of great interest for most technicians, as it may point out to a *trend reversal*.

Trend lengths are also to be considered and, depending on a trader preferences, some may be more valuable than others. A long-term trend is generally considered to be of several years on the stock markets, while a intermediate trend is between one and 3 months and a short-term being less than a month. Figure 2.2 outlines the various trend lengths.

The most commonly used *trendlines* are called *support* and *resistance*, they are drawn by joining successive peaks (resistance) and troughs (support) on a chart, as shown in Figure 2.3. They indicate turning points on a market, typically when the market becomes *bullish* or

Figure 2.2: *Trend lengths [13]*

bearish[8]. Support and resistance are generally seen at round numbers where a lot of traders believe the market has reached the top or the bottom and will start going the opposite direction. Therefore, in general, higher volume can be witnessed when approaching either trendline, meaning higher volatility and orders getting filled with more slippage. A trader must therefore be very careful when posting orders near resistance or support.

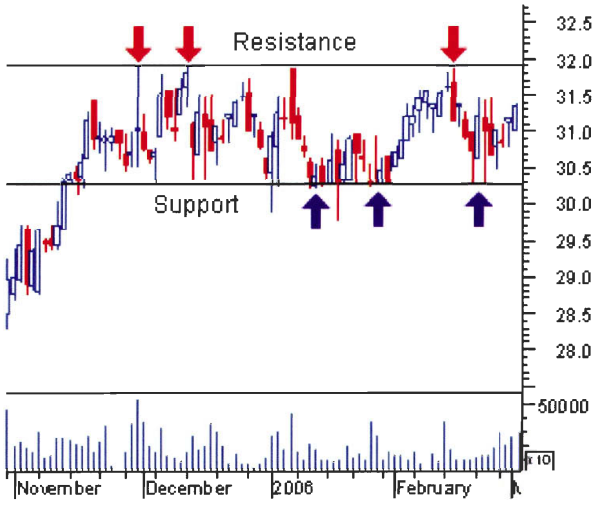

Figure 2.3: *Support and resistance [13]*

If the markets *punches* through resistance or *plummets* through support, this often means the balance between supply and demand has shifted and new trendlines are about to be defined. Rather frequently, this event leads to a *role reversal* where support becomes resistance

[8] *bulls* are buyers and believe the market will go up while *bears* are sellers and think the market will go down.

or vice-versa. Figure 2.4 shows an example of this phenomenon on the Wal-Mart stock (VMT) between 2003 and 2006.

Figure 2.4: *A role reversal example [13]*

2.3.2 Volume

Volume indicates the number of trades that occur on any given security over a specific period of time. It is indicated using a bar chart attached to the bottom of a security price chart, as shown in Figure 2.5.

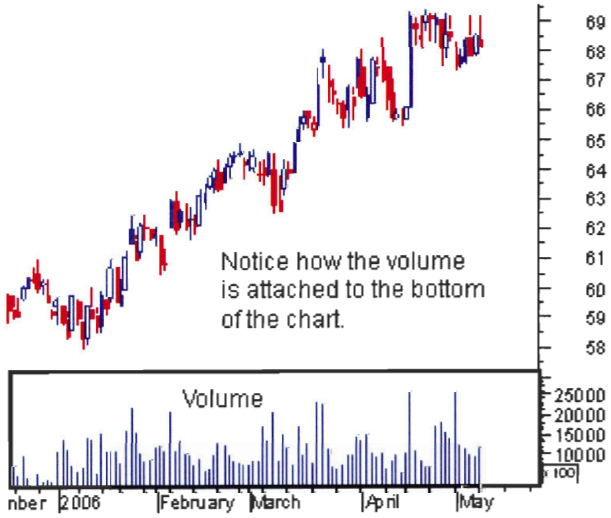

Figure 2.5: *Volume chart [13]*

It is important to understand that volume is different to *open interest* which calculates the cumulative number of longs, or the total number of contracts not closed on any particular day. To illustrate the difference, here is a quick example:
-Day 1: Adam buys 1 wheat contract from Brian: Volume = 1, Open Interest = 1
-Day 2: Charlie buys 8 wheat contracts from Dominic: Volume = 8, Open Interest = 9
-Day 3: Adam sells 1 wheat contract to Dominic: Volume = 1, Open Interest = 8
-Day 4: Eddy buys 5 wheat contract from Charlie: Volume = 5, Open Interest = 8

Worth noting on this example is the trade on Day 4, basically switching 5 open contracts from Charlie to Eddy (Charlie kept 3), a swap that changed noting to open interest while generating volume.

Analyzing volume has a lot to do with crowd psychology since typically, volume moves with the trend. When volumes becomes a very important tool to look at is to identify trend reversals. Most usually, on an upward trend, when volumes starts to decrease, it might be a sign the trend is about to reverse. When that happens, one must also monitor volume indicators to validate the price move; a reversal with little volume is nothing to worry about as there is still confidence in the market that it will catch up. However, if volumes is high on an uptrend reversing, a long should probably get out.

2.3.3 Charts

On the financial markets, charts typically represent the closing price for each day, week or month a stock or commodity is traded, depending on the chart timescale. Intraday charts show price movement from the opening bell to the closing bell. The line chart is the most common and widely published.

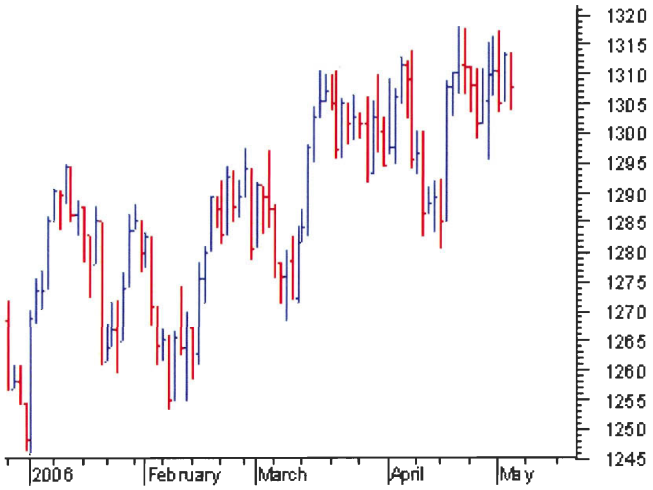

Figure 2.6: *A bar chart [13]*

Bar charts (Figure 2.6) extends line charts by plotting high and lows for the period, and draws opening and closing prices using a dash on the left side or right side of the vertical bar, respectively.

Candlestick charts (Figure 2.7) are a slight variation of bar charts as they add some visual information about the day price movement (between opening and close) using the line thickness, where large variations dictates wider vertical lines.

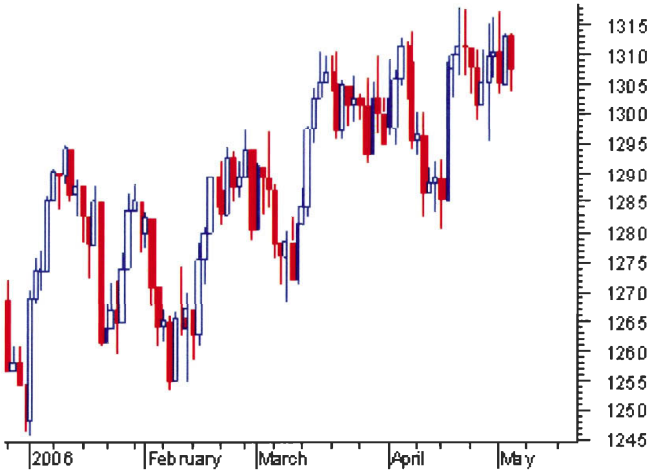

Figure 2.7: *A candlestick chart [13]*

2.3.4 Chart Patterns

One of the foundations of technical analysis is the strong belief that history repeats itself and therefore, chart patterns should repeat themselves also. This section quickly covers the mainstream patterns chartists look at to identify trend continuation or reversals in order to make efficient buy or sell decisions. Obviously, there is no such thing as a "sure shot" pattern, but many TA advocates have had some success following them over several trades.

Head and Shoulders: This pattern (Figure 2.8 a) is a reversal signal showing a weakening in the successive highs, after hitting twice the level of resistance, or *neckline*. The inverse head and shoulders pattern (not illustrated) follows the same principle, and foresees an uptrend.

Cup and Handle: Shown in Figure 2.8 b), this is a continuation pattern pointing to an uptrend once the handle portion has been confirmed. This pattern timescale is normally spread across several months to more than a year.

Double Tops and Bottom: As in Figure 2.8 c), this intermediate to long term reversal pattern illustrates a security trying to punch through resistance (or through support, for double

bottoms) twice, unsuccessfully, then reversing and going down.

Triangles: Considered over a couple of weeks to several months, triangles (Figure 2.8 d) are most often breakout signals, either pointing to an uptrend (ascending triangle), or to a downtrend (descending triangle, not shown).

Pennant and Flag: Also viewed as *consolidation*, the pennant and flag patterns (Figure 2.8 e) are characterized by an uptrend, a consolidation phase of 1 to 3 weeks and are completed once the trend goes up again, abruptly. The pennant differs from the flag by its converging trendlines. Both patterns can also be observed on downtrends.

Triple Tops and Bottoms: Although not widespread as their double counterpart, these patterns indicates trend reversals in the same way, as shown in Figure 2.8 f). They can be confusing to many chartists as there is no easy way to determine, after the second top (or bottom), if the pattern is about to take the downtrend or hit support again to lead to a triple top pattern.

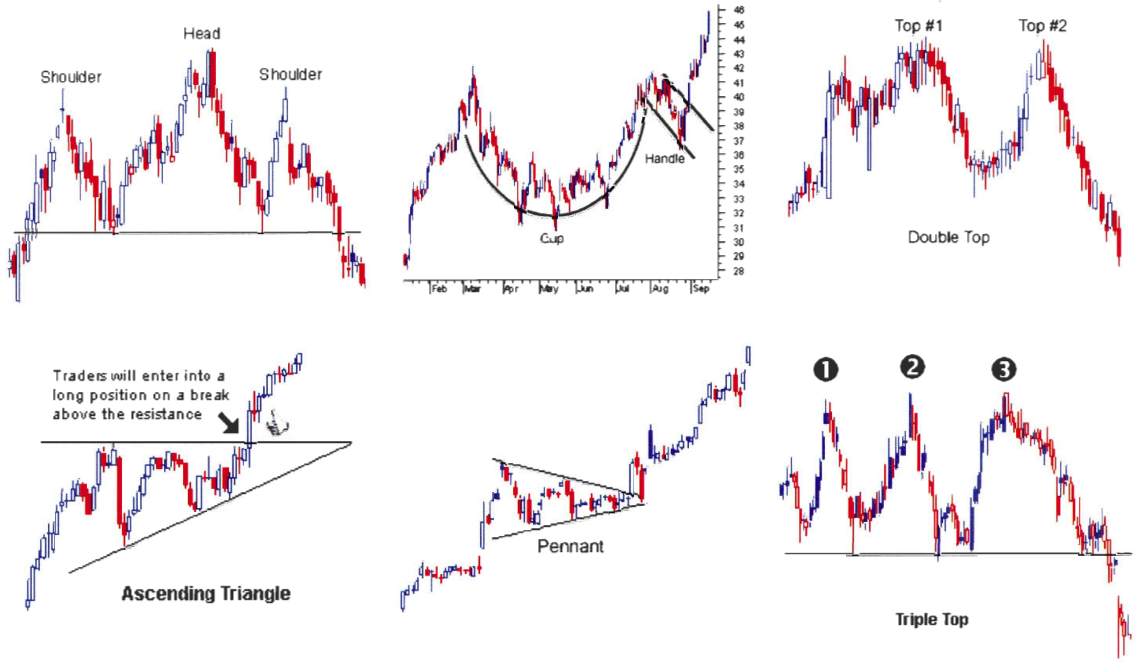

Figure 2.8: *Chart patterns: a) head and shoulders, b) cup and handle, c) double top, d) ascending triangle, e) pennant, f) triple top [13]*

This section outlined very quickly some widely used patterns applied by technical analysts to steer their decision about entering or exiting the market. Many other patterns exists and much are presented in the references at the end of this chapter.

2.3.5 Moving Averages

Seen by many technical analysts as the most useful tool, moving averages smooth out variations in price movement in order to get a better idea of a security overall trend. While fundamentally all based on the same principles, simple, linear and exponential moving averages differ on the weighting they apply on each price point, the latter placing increased importance on more recent data.

Simple Moving Averages

Most common method of calculation, a simple moving average (SMA) adds all price over the considered period and divides by the number of prices in the period. As shown in Figure 2.9, the longer the SMA period, the less sensitive it is to "noise" and the longer trend it outlines. The one drawback of simple moving averages it that they give an equal weight to all values over the considered period, and since more recent values should logically be more relevant when computing the moving average to any given point, SMAs tend to underestimate the momentum.

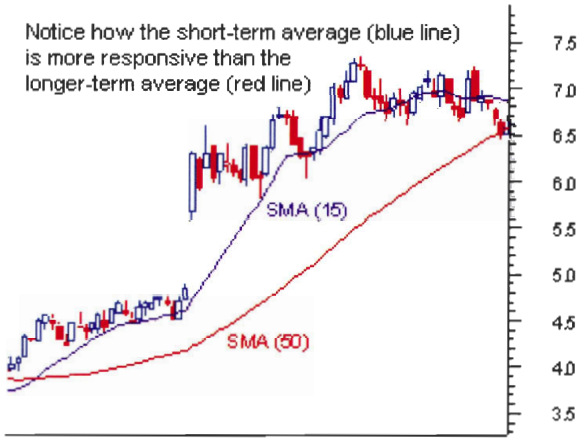

Figure 2.9: *Simple moving averages [13]*

Linear Weighted Averages

Although not widely used as the exponential moving average, the linear weighted average multiplies each data point by its position over the considered period. For instance, in a 10-day moving average, the actual data point would be multiplied by 10 and then each preceding data point by 9, 8, 7... an so on, to be subsequently divided by the sum of multipliers.

Exponential Moving Averages

While based on the same principle as the linear weighted average, the exponential moving average (EMA) computes the weighting factor following an exponential function, therefore placing even more importance to recent data than its linear counterpart. As shown in Figure 2.10, it is often used in conjunction to the SMA as it is more responsive and can sometimes predict SMA variations.

Figure 2.10: *Exponential moving average [13]*

As mentioned previously, moving averages are one of the widespread tool used in TA, mainly to identify trend reversals as well as support and resistance levels. For instance, when the price of a security falls below or breaks through a long period SMA, there is a good chance of trend reversal, as shown in Figure 2.11.

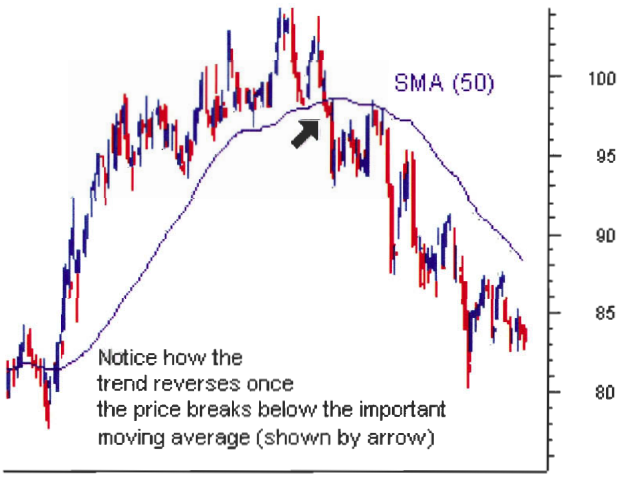

Figure 2.11: *Identifying a trend reversal from a moving average [13]*

The other indicator closely watched by chartists is the crossover between 2 moving averages, either SMA vs EMA or similar with different periods (Figure 2.12).

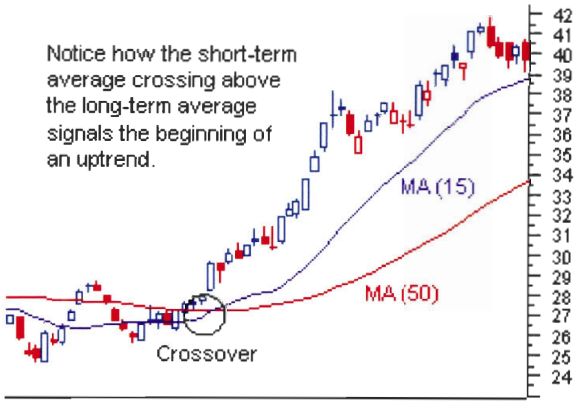

Figure 2.12: *Crossover between 2 moving averages [13]*

Last but not least, long period moving averages (Figure 2.13) are also used to identify support and resistance levels, and since many technical analysts rely on them, they are often quite dependable.

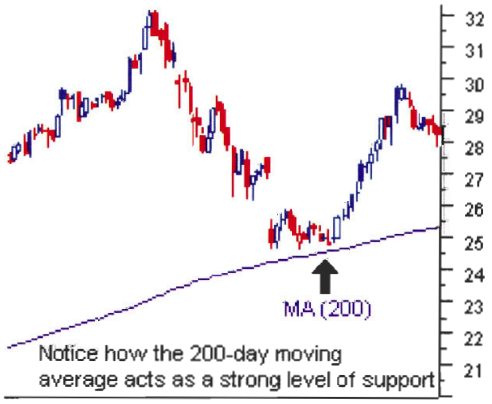

Figure 2.13: *Support level from a 200-day moving average [13]*

2.3.6 Other Indicators

Indicators are tools derived from charts and volume information used to outline trends but also other important factors like momentum, money flow and volatility. They are used most often to forge buy and sell signals (leading indicators) but also to confirm trends (lagging indicators). Most indicators have a bounded range (between 0 and 100 for example); these bounded indicators are called oscillators. Many books and publications are focused on TA indicators (see references at the end of this chapter), following is a brief description of some of the most common indicators.

Accumulation/Distribution Line

This indicator measures the ratio between buyers and sellers over a pre-defined time period. It is calculated using this equation:

$$AccDist = CLV * Period's\_Volume \tag{2.1}$$

where CLV is the Close Location Value that reflects the closing price of a security over the relative range of trading for a fixed period. A value of +1 means the security closed at the high value for the period, -1 the low value, and 0 being halfway between high and low.

$$CLV = \frac{(Close - Low) - (High - Close)}{(High - Low)} \tag{2.2}$$

For example, if a security CLV for a given day is 0,2327 and the day trading volume is 1 million shares, then +232 700 will be added to the accumulation distribution line, pointing to an upward, or buying, trend.

Moving Average Convergence

This indicator, also known as the MACD (moving average convergence divergence) plots the difference between two EMA, against a centerline, where both EMAs are equal. In addition, an exponential of the MACD itself, called the "signal", is plotted in order to get a better idea of the short term momentum over the long term. As shown on Figure 2.14, buy signals are generated when the MACD crosses up the signal line, and sell signals are when the MACD crosses down the signal line.

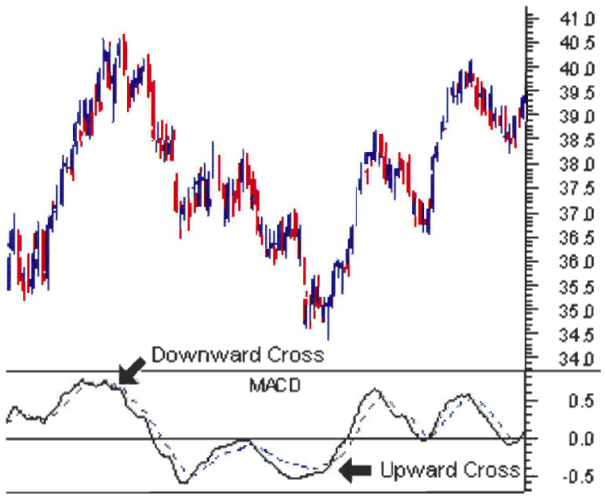

Figure 2.14: *Moving Average Convergence/Divergence and signal line [13]*

Relative Strength Index

The Relative Strength Index, or RSI, is used to determine overbought or oversold conditions of a security. It is calculated using this equation:

$$RSI = 100 - \frac{100}{1 + RS} \tag{2.3}$$

where

$$RS = \frac{\text{sum of closing prices of updays}/n}{\text{sum of closing prices on downdays}/n} \tag{2.4}$$

where n is the considered trading period lenght.

Used in conjunction with other indicators, the RSI (Figure 2.15) is considered to send a buy signal when higher that 70 (or 80 for more conservative chartists) and a sell signal when below 30 (or 20).

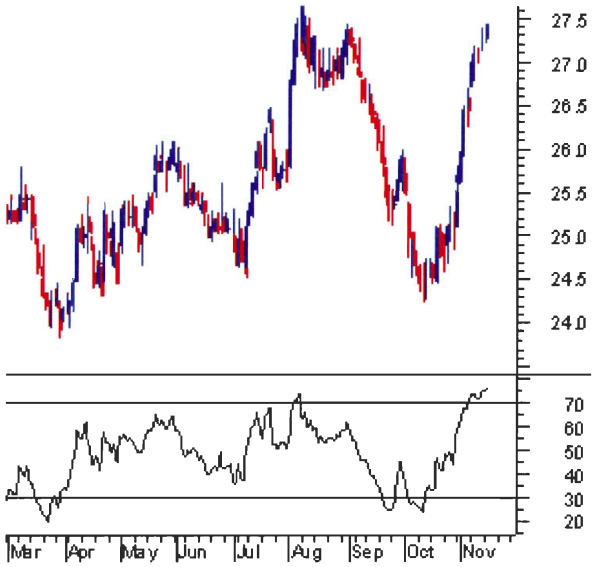

Figure 2.15: *Relative Strength Index [13]*

Stochastic Oscillator

The theory supporting the stochastic oscillator relies on the frequent observation that in a downward market, prices tend to close near their previous low and in an upward market, prices close near their previous high. This indicator is computed using the following formula:

$$SO = 100 * \frac{C - L14}{H14 - L14} \tag{2.5}$$

where C is the closing price, $L14$ the low of the 14 previous trading sessions and $H14$, the highest priced traded over the same period. As shown on Figure 2.16, buy and sell signals are triggered when the stochastic oscillator is over 80 and under 20, respectively.

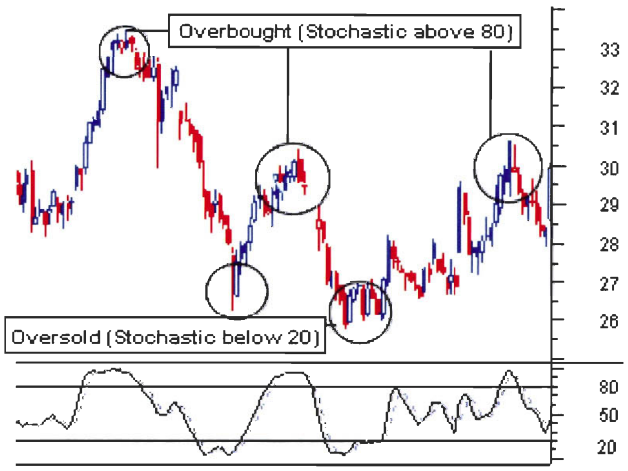

Figure 2.16: *Stochastic Oscillator [13]*

2.3.7 Fundamental vs Technical Analysis

It's an ever-ending debate between fundamentalists and technical analysts about the motivation behind entering or exiting any position. Following the *efficient market hypothesis*, fundamentalists argue that the market price of any security is always the right one, making any historical analysis useless. Stated differently, they strongly believe the *fundamentals* of any stock, commodity or option are always reflected in the actual price and there is no profit to make in the short term. Therefore, most fundamentalists pick stocks with good ratios, statements and balance sheets, companies that should improve their value on the longer run.

Chartists believe all the information they need about a stock can be found in charts. They base their theory on 3 assumptions: 1) the market discounts everything, 2) prices move in trends and 3) history tends to repeat itself. They therefore monitor uptrends, downtrends and sideways trends, highlighting trendlines and volume in trying to extrapolate higher abstraction figures like channels, chart patterns, resistance and support. On top of that, they use indicators and oscillators to get buy and sell signals on any given position.

On the commodity markets, technical analysis is much more widespread than on stock markets as trading on *material but non-public information* is not illegal. Hence, since large corporations, either producers or consumers of any commodity, cannot make any significant move on the market without leaving traces, the argument for TA makes sense.

There is a great deal of literature on both approaches, and as this chapter's intent is to give a broad introduction to the financial markets in order to get a better understanding of the following chapter, choices had to be made on the various aspects to cover. Much had to be left behind and can be further investigated through the references at the end of this chapter.

2.4 Auction Models and Price Setting Mechanisms

According to ancient Greek scribes, auctions have been used since 500 B.C. to trade goods, animals and even people. More typically, in the general economic theory, auctions are a method to assess the value of a commodity that has an undetermined or variable price. Normally, a *supply auction* refers to an auction with many sellers and 1 buyer, a *demand auction* has 1 seller and many buyers while a *double auction* is a many to many relationship. This section covers the most common auction types: English, Dutch, French, Vickrey and Double.

English Auction

The english auction is by far the most widely known auction type where the auctioneer starts the auction at the *reserve price* and accepts increasing bids until nobody wants to bid higher. It is also known as the *open-outcry auction* as every bidder shouts his bid, until no one is willing to speak up. Variations exist where, for instance, bidders make hand signals or raise bidding paddle instead of shouting out loud. Also, the reserve price can be kept private in order to prevent bidder coalitions to under-bid the asset. English auction can also be carried out for multiple units, where the n highest bidders win an identical asset. Most of the time, the english auction is in the seller's favor, as bidders, especially if unexperienced, sometimes get carried away in the bidding process. Therefore, overbidding frequently occurs during english auctions and the price paid by the winner is sometimes much higher than the *marginal utility*[9].

Dutch Auction

While fundamentally similar to chinese auctions, Dutch auctions are acknowledged around the world because of the tulip auction performed every year by the royal gardens in the Netherlands. In this type of auction, the price for 1 to n goods, is set at a high asking price then incrementally lowered until one or n buyers are willing to pay the auctioneer's price. Every winning participant pays the last announced price. Dutch auctions are particularly useful to auction goods within short timeframes.

Dutch auctions are often used for shares repurchase. Within this process, a company posts the number of shares, or quota, it is willing to repurchase and invites its shareholders to tender their stock within a pre-determined price range. It then compiles the offer's number of shares and price and buys back the shares from the cheapest offer all the way up to the pre-defined quota, at the price where the quota is met.

[9] the value of an additional unit of the auctioned good: the price paid by the second highest bidder.

Google followed a similar process for its IPO[10] by asking prospective buyers to submit their bid for Google shares. Following this *Open IPO* process, underwriters then settled the price at 85$ and issued shares to anybody that was willing to pay 85$ or higher, but at 85$ for everybody.

French Auction

Also known as *tâtonnement*, this auction type was invented by French mathematician, and Nobel Prize, Leon Walras in 1874. Therefore also called a *Walrasian auction*, this process incorporates all the vendors and buyers bid quantity and price, and calculates a unique price level that will result in a maximum quantity being traded. This computation is done in seconds each day at the London Bullion Market for gold. This auction method can also be used for stocks, but since it is particularly efficient with many bids, it used to be executed at the opening when many orders were pending; with the advent of the internet and after hours trading, it is less used for stocks.

Vickrey Auction

The Vickrey auction, from the name of its creator, William Vickrey, is a *sealed-bid second price* auction [11]. *Sealed-bid*, because no buyers knows other people bids and *second-price*, because the winner bidder pays the second highest bid. For multiple units, the Vickrey auction becomes a *uniform price auction* where the price paid is the highest non-winning bid and is the same for all winning buyers.

The foremost appeal for this type of auction lies in the fact that it is *incentive compatible* in forcing every bidder to reveal his foreseen utility by privately stating asset valuation. It does not, however, maximize the seller revenue neither does it provide any help for price discovery in the case of an asset of undetermined value. While *strategy-proof*[12] in principle, there is nothing preventing a seller to use shill bids to increase profit.

Double Auction

Although somewhat different from other auction types, double auctions are used around the globe in exchanges and therefore represent the foremost trading support tool. In a double auction, multiple sellers offer various quantities of the traded good at different *ask* price points. In order for the market to behave, only the lowest ask price can be advertised in an open-outcry or published on an electronic market. Similarly, multiple buyers posts bids for

[10] initial public offering

[11] the appellations *Vickrey* and *sealed-bid second-price* auction are used interchangeably in the literature

[12] since bidders reveal their true utility

varying quantities of the asset at different price points and only the highest one can go public. When they crossover, a transaction can be completed. This is why most exchanges publish bid and ask prices and not only the last price at which a transaction occurred.

2.5 Finance and Grid Resources

This chapter introduced some relevant economic and financial concepts used across worldwide markets today. Although most Grid resources like processors, memory, storage, networking and software can be seen as commodities, there is no such thing as a standardized contract for HPC and there will never be as application requirements are too diversified.

In addition, balance sheets, income statements, earnings per share or any such fundamental data do not find any comparison in the supercomputing world. There is, however, fundamental technological trends in the computing industry in general, trends similar to some observed on the commodity markets. Nevertheless, HPC resources exhibit a lifetime cycle much shorter than any other commodity as they present some initial scarcity and high demand, followed by a market saturation and finally depreciation, all within a few years.

For any market to emerge and be sustainable, the price discovery issue has to be addressed appropriately. Since standardization is not an option, there is a need to provide price discovery for both buyers and sellers in order for them to enter the market and trade. This mechanism will provide the essential assessment insight much needed to control volatility and support trading on a daily basis. It also has to be adaptable, more adaptable than any other price discovery mechanism, as Grid resources come and go faster than anything else.

As we will see later, the double auction mechanism is a natural choice for trading these resources as it favors neither the seller nor the buyer. In order to close as many transactions as possible and limit the time spent trading, both sellers and buyers could then potentially implement semi-automated to fully-automated trading algorithms.

The next chapter presents the proposed multi-commodity economic model for the Grid, incorporating fundamental and technical backgrounds, dynamic and adaptative price discovery, heterogeneous "contracts" and a double-auction trading platform.

2.6 References

Commodity Trading

1. George Kleinman. *Trading Commodities and Financial Future: A Step by Step Guide to Mastering the Markets*. 3rd edition. Financial Times Press, 2004. [101]
2. Richard E. Waldron. *Futures 101 : An Introduction to Commodity Trading*. 2nd edition. Squantum Publishing Company, 2003. [148]
3. Helyette Geman. *Commodities and Commodity Derivatives: Modelling and Pricing for Agriculturals, Metals and Energy*. Wiley, 2005. [80]

Technical Analysis

1. John J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. 2nd edition. Prentice Hall, 1999. [113]
2. Charles D. Kirkpatrick and Julie R. Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians*. Financial Times Press, 2006. [100]

Economics

1. John Nash. *Non-Cooperative Games*. dissertation, 1950. Reprinted in Harold Kuhn and Sylvia Nasar, editors. *The Essential John Nash*. Princeton University Press, 2007. [116]
2. Gerard Debreu. *Theory of Value: An Axiomatic Analysis of Economic Equilibrium*. Yale University Press, 1972. [60]
3. Leon Walras. *Elements of Pure Economics or the Theory of Social Wealth*. Harvard University Press, 1954. [151]
4. Bryan Ellickson. *Competitive Equilibrium: Theory and Applications*. Cambridge University Press, 1994. [62]
5. Robert Gibbons. *Game Theory for Applied Economists*. Princeton University Press, 1992. [83]
6. Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997. [114]

Other Interesting References

1. Leonid Hurwicz. *The Design of Mechanisms for Resource Allocation*. American Economic Review, 1973. [92]
2. Kenneth Arrow and Leonid Hurwicz. *Studies in Resource Allocation Processes*. Cambridge, 1977. [34]

Chapter 3

An Innovative Market-based Grid Exchange Model

Scheduling is a research subject that has inspired a plethora of literature and is still one very hot topic in the computer science world. For a single computer, where, most of the time, CPU cycles are abundant compared to application processing requirements, good scheduling algorithms support concurrent multi-tasking and provide overall interface responsiveness. In the HPC world, these usability requirements find little echo as for the most part, HPC applications are strictly throughput oriented and need to execute as fast as possible, without much user interaction while running.

HPC schedulers and even Grid scale meta-schedulers have nevertheless tried to optimize job allocation following the same paradigms used in modern operating system kernels, balancing jobs sizes, lengths and other requirements. But rarely have users been able to express the value of a job being completed before a deadline or on any given specific supercomputer architecture.

Utility computing is trying to address this problem by allowing users to express a variable eagerness for specific resources at different timeframes, then using this utility expression to steer scheduling decisions.

However, in order for any resource valuation to be well established, the process must follow 2 requirements. First, the resource has to be limited, or stated otherwise, users have to compete against each other to gain access to it. Second, the instrument for valuation expression has also to be finite and limited too. Following these 2 conditions, users would then transfer the value inherent to a precious but abstract valuation instrument into precious as well, but very real, HPC resources.

For any competition to occur, users have to be aware of others interest in a resource, and how big that interest is. This enables *market positioning*, where users are willing to bid on a resource, but not to exceed a given price-point. In a similar fashion, resource producers must be informed of others offerings, as they may find value in differentiation rather than competing against bulk and cheap siblings.

By compelling users to spend some precious instrument to gain access to resources, utility computing will bring more responsible computing practices, leading to enhanced data-sets being studied and more efficient code deployed across the infrastructure. Similarly, producers, while trying to maximize revenue, will find value in differentiation as well as higher QoS levels. For both sides, the transformation of a precious valuation instrument into limited HPC resources and vice-versa, brings more efficient and more responsible computing, 2 objectives that are currently poorly addressed by actual frameworks.

There is therefore a need for a *price discovery* mechanism that would be used by both users and providers to steer their positioning.

This chapter presents the fundamental contribution of this thesis in defining and detailing an economic model where both users and providers would be able to interact in a freely evolving market. This rather philosophical incentive engineering exercise will show that conventional scheduling may not, in fact, be the right paradigm to follow on an HPC Grid as advanced reservation coupled to a sound economic model could handle it all.

While based on commodity and spot markets where standardized contracts are traded in multiples of well-defined goods, the proposed model enables the exchange of non-standardized resource sets and extracts market indices from relevant previous transactions. In addition to such multi-commodity sets being non-trivial to handle, the system is also able to support variable allocation time lengths, heterogeneous job processor counts and unlimited transaction-time to execution-time differential.

Although any computing instance could be exchanged on this market, from a core to an entire MMP, we choose to focus the trading unit around a single processor (or 1 "socket"), a scale compromise understood and widely used by most scientists and operators. Hence, memory, storage, networking and every other additional resource will be expressed on a single processor scale. Limiting the scale as such and thus defining some kind of standardized *contract size* not only facilitates the understanding of the proposed model, but it also limits the complexity of the system, a very important aspect especially when bootstrapping the system, in order to test scalability but also to support a pricing scheme based on comparables.

In a fashion similar to trading months on the commodity markets, we also choose to limit

traded timeslots to 6 hour periods and we consider similar every timeslot within a trading month. Again, this is to ease initial deployment for scalability reasons while also limiting pricing volatility. Nevertheless, nothing would prevent the managing entity to progressively augment timeslot granularity in order to pinpoint more precisely execution timeframes once the system is up and running in stable mode.

The following therefore presents a multi-commodity trading model standardized around a single processor where commodity bundles are traded in multiples of a single processor with co-allocated resources, and potentially over several computation periods.

We start by defining how these multi-commodity resource sets will be expressed, matched and traded. We then develop a pricing assessment approach based on a proximity function using statistical information from "similar" previous deals.

In the economic literature, Rosen [130] and later Brown [40] have developed a theory on *hedonic prices* where, for instance, they argue a car pricing could be extrapolated from its implicit, or *hedonic*, characteristics. In [130], Rosen demonstrates the existence of market equilibrium as well as the motivation for consumption or production decisions. While proved theoretically, the hedonic prices have seen few applications up to now. As most goods, like cars, most frequently enter the economy in a simpler form and become more and more complex over time; markets simply adapt to that rising complexity. In addition, any conventional good presents some well-known production costs and therefore its pricing on the market becomes rather straightforward. These considerations have up to now limited the application of hedonic prices theory to a certain extent.

HPC resources, however, have never been traded in a freely evolving market *per se*. They are otherwise intrinsically complex, from the very first market order. In addition, production costs, especially in the academic community, are somewhat difficult to nail down. For sure, power and cooling costs can be assessed quite precisely, but acquisition cost amortization, as every installation is somehow discounted, presents a bigger challenge.

This thesis therefore presents one of the few empirical demonstrations of some hedonic pricing derivative approach applied to HPC resources trading. This chapter presents the economic model foundations and chapter 4 follows with a market simulation of the model. Consequent issues related to contract standardization verification, quality of service, credentials validation and trust management are then discussed in the following chapter. Chapter 5 also further analyses currency allocation, governing entity involvement and other macroeconomic matters.

3.1 Resources Sets

In order to be able to express both user application requirements and provider computational resource features, we define several types of resources that can be combined in *requirement sets*, for applications, and *component sets*, for compute resources.

By definition, a *requirement set* will list the required features in terms of CPU, RAM, storage, networking, etc., that an application needs to execute successfully.

In a similar fashion, the *component set* details the available features of a provider's HPC resource. Noteworthily, component sets tend to be more extensive than requirement sets as they are an exhaustive list of *available* features while requirement sets are *mandatory*.

The following presents the various types of resources to be handled by the system, and examples for each case. It is important to remember that these lists are not intended to be exhaustive as the economic model presented here is dynamic, fully adaptive and can therefore assimilate any future technology.

The potential explosion of resource elements to be handled by the system is not, by itself, problematic, since only the ones presenting any market value would really be considered by any trader polling information to enter a position.

We therefore start with a set of resources $\mathbf{\Phi}$ from which job requirements $R \subset \mathbf{\Phi}$ and processor related components $C \subset \mathbf{\Phi}$ can be expressed. From this extensive set of resources, we then define 5 types: processing, memory, storage, networking and software. While these resource types will be the ones covered within the body of this thesis, it is important to note that the model is not limited to them and can be further extended to include many others.

$$\mathbf{\Phi} = \Phi_{\text{proc}} \cup \Phi_{\text{mem}} \cup \Phi_{\text{disk}} \cup \Phi_{\text{net}} \cup \Phi_{\text{soft}} \tag{3.1}$$

3.1.1 Types of Resources

Processing resources

They include processor architectures, clock speeds, number of cores, cache sizes, vector units, in-silicon features like fused multiply-add as well as enhanced instruction sets like SSE4. They also include processor accelerator cards, FPGAs, GPGPUs and the like. We can

therefore expand Φ_{proc} to several sub-types:

$$\begin{aligned}
\Phi_{\text{proc_arch}} &= \{\text{x86, itanium, intel, amd, power, bluegene, sparc, nec}\} \\
\Phi_{\text{proc_model}} &= \{\text{woodcrest, clovertown, penryn, santa-rosa, barcelona, power5, niagara, rocks}\} \\
\Phi_{\text{proc_clock}} &= \{1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0\}\ \text{(GHz)} \\
\Phi_{\text{proc_cores}} &= \{1, 2, 4, 6, 8, 16, 32\} \\
\Phi_{\text{proc_L2}} &= \{1, 2, 4, 6, 8, 12, 16\}\ \text{(MB)} \\
\Phi_{\text{proc_inst}} &= \{\text{128bit, SSE3, SSE4, fused_mult_add}\} \\
\Phi_{\text{proc_coproc}} &= \{\text{FPGA, GPGPU, ClearSpeed}\}
\end{aligned} \tag{3.2}$$

Leveraging on this rather flexible expression syntax, we could also define several resource characteristics based on micro-benchmarks specs in terms of floating point or integer performance, such as:

$$\begin{aligned}
\Phi_{\text{proc_linpack}} &= \{4, 8, 12, 16, 20, 24, 32, 40, 48, 64, 96, 128\ \}\ \text{(GFlops)} \\
\Phi_{\text{proc_specint}} &= \{20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}\ \text{(2006, rate, base)}
\end{aligned} \tag{3.3}$$

Memory Resources

For a growing number of applications, memory size and bandwidth are becoming a cornerstone issues. We thus develop Φ_{mem} to include not only RAM quantity per processor but also RAM type and clock:

$$\begin{aligned}
\Phi_{\text{mem_size}} &= \{1, 2, 4, 8, 12, 16, 24, 32, 48, 64, 128, 256\}\ \text{(GB)} \\
\Phi_{\text{mem_type}} &= \{\text{DDR1, DDR2, DDR3, FBDDR2}\} \\
\Phi_{\text{mem_clock}} &= \{533, 667, 800, 1066, 1333\}\ \text{(MHz)}
\end{aligned} \tag{3.4}$$

While the addressable memory space within a "box"[1] would seem like the logical choice to define resource sets after (as it represents most memory-savvy applications essential requirement), it would have led to a too extensive and thus uncorrelated expression set for memory resources. Despite this apparent limitation, a job presenting some large addressable memory requirement can still be expressed through a multiplication of processor count by the amount of memory per processor; then, at the same time, stating a specific cache-coherent interconnect. Using this approach, the trading system is then able to *express* and, most importantly, *compare* any job memory requirement or compute server memory component, from blades to large MMPs, all within a reduced and more manageable memory sizes resource set.

[1] around 64 GB for today's typical cluster node and up to 128TB for the latest SGI's Altix 4700 (ccNUMA)

For memory bandwidth dependent codes, it could also be interesting to add memory performance references such a the STREAM benchmark:

$$\Phi_{\text{mem\_STREAM\_8core}} = \{120, 160, 200, 240, 280, 300\} \text{ (MB/s)} \tag{3.5}$$

As the use of benchmarks can be very tricky, especially if linked to some economical value, one will have to make sure the benchmarks used are widely known codes with reproductible results by a third-party validating authority, or by any user willing to verify if he did get what he paid for. This aspect falls into the contract standardization verification and QoS issues that will be discussed in section 5.3.

Storage Resources

Following processor and memory requirements, storage management is becoming a major headache for many site operators as the amount of data produced by scientific applications in the past few years has just exploded. Not only application storage requirements can now be on the order of several terabytes but read and write performance, thanks to parallel filesystems like *Lustre* [15] and *GPFS* [10], are now in the order of several GB/s per stream. Our storage expression syntax hence goes as follow:

$$\Phi_{\text{disk\_size}} = \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 768, 1024\} \text{ (GB)} \tag{3.6}$$
$$\Phi_{\text{disk\_bw}} = \{16, 32, 64, 128, 256, 512, 1024\} \text{ (MB/s)}$$

The storage capacities and speeds expressed here reflect the whole spectrum of storage alternatives, from a local disk to a parallel file system over an high-performance interconnect. When expressing storage needs, both user and provider must make sure their resource level reflects the size and speed available to a single processor, analogously to memory, such that a local scratch disk bandwidth should be divided by 2 on a dual-socket motherboard. Here also, a bandwidth requirement validation algorithm would have to be put in place in order to make sure the advertised bandwidth is fulfilled on execution.

Networking Resources

High-Performance Computing has always been a driving force for networking in general and more specific interconnect technologies. Using this economic model, scientists and site operators will now be able to express their interconnect preferences in terms of technology,

bandwidth and latency. Hence:

$$\begin{aligned} \Phi_{\text{net\_tech}} &= \{\text{100T, GigE, 10GigE, IB\_SDR, IB\_DDR, Myrinet, NumaLink4}\} \\ \Phi_{\text{net\_bw}} &= \{10, 100, 1000, 2000\}\ \text{(MB/s)} \\ \Phi_{\text{net\_latency}} &= \{50, 10, 5, 1, 0.5, 0.1\}\ (\mu\text{s}) \end{aligned} \tag{3.7}$$

Once again, while stating a technology and associated bandwidth in its component set, a provider must take great care to divide by the number or processors on the motherboard. For example, a single InfiniBand DDR host connection adapter (HCA) on a double socket motherboard would have to be stated with 1000 MB/s of bandwidth (instead of 2000 MB/s) as the bandwidth is shared between the 2 sockets. Also, the latency figure expressed here is the worst-case roundtrip ping latency between 2 nodes and through spine switches.

Software Resources

Many scientific applications not only have hardware requirements but also frequently rely on libraries, compilers, drivers and middlewares. Math libraries, genomic tools like BLAST, specific InfiniBand driver versions and higher-level middlewares like Globus or a specific scheduler capability such as on-demand provisioning are all examples of software requirements an application can have. We therefore expand our syntax as such:

$$\begin{aligned} \Phi_{\text{soft\_package}} &= \{\text{BLAST\_2.2.14, GROMACS\_3.3.3, FLUENT\_6.3, GLOBUS\_4.0.6}\} \\ \Phi_{\text{soft\_driver}} &= \{\text{OFED\_1.2.5, InfiniPath\_2.1}\} \\ \Phi_{\text{soft\_other}} &= \{\text{MOAB\_5.2, VMware\_1.0.3}\} \end{aligned} \tag{3.8}$$

Worth noting in these examples is the software resource version expression in the syntax. As many codes are built around specific library versions, one must make sure the provided computational resource does supply the right version in order for the code to run bug-free.

3.1.2 Consumers Requirement Sets

A scientific application requirement set $\mathcal{R}$ can then be expressed as follow:

$$\mathcal{R} = \{\text{x86}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 2.4_{\text{proc\_clock}}, 4_{\text{mem\_size}}, 2_{\text{disk\_size}}\} \tag{3.9}$$

This resource set expression would thus specify very little requirements, probably denoting an embarassingly[2] parallel application. All types of applications could then be expressed

[2] an application that scales very easily thanks to little inter-process communications

like this, all the way to the tightly-coupled SMP code on a parallel filesystem:

$$\begin{aligned}\mathcal{R} = \{&\text{itanium}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 2.0_{\text{proc\_clock}}, 4_{\text{mem\_size}}, 800_{\text{mem\_clock}}, \\ &4_{\text{disk\_size}}, 256_{\text{disk\_bw}}, \text{Numalink4}_{\text{net\_tech}}, \text{BLAST\_2.2}_{\text{soft\_package}}\}\end{aligned} \tag{3.10}$$

It is therefore very easy for any scientist to express his HPC needs from the very common serial cluster to the vector computer, coming across InfiniBand and Myrinet capability clusters and SMP/MMPs.

3.1.3 Providers Component Sets

Any supercomputing resource can be expressed similarly through a provider's component set, $\mathcal{C}$, such as:

$$\begin{aligned}\mathcal{C} = \{&\text{x86}_{\text{proc\_arch}}, \text{clovertown}_{\text{proc\_model}}, 4_{\text{proc\_cores}}, 2.0_{\text{proc\_clock}}, 8_{\text{proc\_L2}}, \\ &4_{\text{mem\_size}}, \text{FBDDR2}_{\text{mem\_type}}, 800_{\text{mem\_clock}}, 64_{\text{disk\_size}}, 128_{\text{disk\_bw}} \\ &\text{IBDDR}_{\text{net\_tech}}, 1000_{\text{net\_bw}}, 5_{\text{net\_latency}}, \text{OFED\_1.2.5}_{\text{soft\_driver}}\}\end{aligned} \tag{3.11}$$

Intuitively, component sets tend to be much more extensive as they express the *available* resources on a given machine while requirement sets, *needed* for the job to execute, tend to be more specific and less numerous.

In order to match a consumer's requirement set with the right provider component set, both sets will be compared and only the matching market orders will be presented to both parties. Hence, the more exhaustive a component set is, the more chances it has to get matched against diverse requirement sets. Reciprocally, the less elaborate the requirement set is, the more matching offers it gets.

Most component sets intrinsically represent an upper bound for many types of resources like processor clock, memory size and frequency, disk size and bandwidth and interconnect throughput. They can therefore get matched against any lower or equal level of service requirement.

Hence, the set $\mathcal{A}_b$ of admissible component offers o for bid b with requirements $\mathcal{R}_b$ is given by:

$$\mathcal{A}_b = \{o \in O \mid \mathcal{R}_b \cap \mathcal{C}_o = \mathcal{R}_b\}, \tag{3.12}$$

where O is the set of all market offers presenting component sets described by $\mathcal{C}_o$.

3.2 Supercomputing Contracts

As previously discussed, a standardized contract, following the commodity trading definition, is not a concept that can be straightforwardly transfered to high-performance computing. As mostly every single job requires a unique and heterogeneous set of computational features, it is just impossible to define a comprehensive contract to be traded on the market. However, since similarities between job requirements and supercomputer components through processing, memory, storage, networking and software are ubiquitous, higher abstraction correlations may be possible.

In the futures market, deals are concluded for multiples of a standardized unit, either barrels, bushels or pounds. In the HPC world in general, the equivalent could be the number of cores, processors or compute nodes. While processor sockets are used for the purpose of this analysis, the atomic trading entity could be defined otherwise in any other market instantiation.

In addition to quantity, futures contracts are bound to a specific timeframe, or delivery period. For commodities, the delivery month is there to buy flexibility for the provider, as the delivery of physical goods is somewhat difficult to pinpoint precisely. Since HPC resources are "leased" rather than sold, and since they do not exhibit any physical delivery and warehousing issues like conventional commodities, their exchange can be made much more flexible than futures. Although HPC resources are somewhat produced on the spot and could be traded, like electricity, in a very short term SPOT market, there is a strong need for advanced reservation throughout the community[3], a fact supporting the futures model. Hence, in addition to contract *sizing*, we also have to consider contract *timing* aspects.

3.2.1 Contract Sizing

Up to here, the number of processors required for a job to execute has been pushed aside deliberately. While being a fundamental aspect of any requirement set, it had to be left out in order to introduce the model representation for core features like processor, memory and storage.

We choose to use a single processor and its inherent features as well as its linked components (memory, storage, networking and software) as the core trading unit. The resource set expression syntax is thus expanded to include processor count. Consequently, a client is able

[3]because most users need to guarantee access to future resource cycles, in order to match some experiment results deadline, for example.

to express the required number of processors, by specifying processor count before the set:

$$\begin{aligned}\mathcal{R} =&(\mathbf{24_{procs}}) \\ &@\{\mathrm{x86}_{\mathrm{proc\_arch}}, 2_{\mathrm{proc\_cores}}, 2.4_{\mathrm{proc\_clock}}, 4_{\mathrm{mem\_size}}, 2_{\mathrm{disk\_size}}\}\end{aligned} \tag{3.13}$$

For resource providers, expressing the available number of processors introduces a more complex syntax as there must be a way for them to specify a minimum and maximum number of processors as well as a pre-defined processor count increment, inherited the interconnect infrastructure for example. Hence, C becomes:

$$\begin{aligned}\mathcal{C} =&(\mathbf{24}, \mathbf{24}, \mathbf{288_{procs}}) \\ &@\{\mathrm{x86}_{\mathrm{proc\_arch}}, \mathrm{clovertown}_{\mathrm{proc\_model}}, 4_{\mathrm{proc\_cores}}, \\ &2.0_{\mathrm{proc\_clock}}, 8_{\mathrm{proc\_L2}}, 4_{\mathrm{mem\_size}}, \mathrm{FBDDR2}_{\mathrm{mem\_type}}, \\ &800_{\mathrm{mem\_clock}}, 64_{\mathrm{disk\_size}}, 128_{\mathrm{disk\_bw}}, \mathrm{IBDDR}_{\mathrm{net\_tech}}, \\ &1000_{\mathrm{net\_bw}}, 5_{\mathrm{net\_latency}}, \mathrm{OFED\_1.2.5}_{\mathrm{soft\_driver}}\}\end{aligned} \tag{3.14}$$

where a provider is willing to trade 24 processor increments up to 288 processors, a specification that probably correlates to his motherboard and interconnect characteristics, where, for example, dual-socket motherboards linked over InfiniBand DDR switches induce building blocks of 12 nodes, or 24 processors.

The rationale supporting the use of processors, or sockets, as the default trading unit instead of processor cores or compute nodes comes from an in-depth analysis of the various technological aspects and their relationship to the market. First, processor cores are always packaged all-together, they share the same cache, memory, and system bus. It would therefore be quite complicated for any user or provider to split processor connected resources per core, and even so, guaranteeing any level of service would quickly turn out to be very tricky as there is no way to divide a system bus evenly for each and every core. Hence, processing cores are probably one level of granularity too small.

Scaling upwards, it could appear natural to trade compute nodes instead of processors. The number of processors per box would then become a standardized processing resource element, similar to cores in the current resource expression syntax. Memory and interconnection host connection adapters (HCAs) would thus be stated on the compute node scale. Although this concept makes a lot of sense for clusters, either serial or more tightly linked together, for SMP/MMPs and other specialized architectures like BlueGene, it quickly becomes very hard to apply. How could a provider split a unified MMP if the model would force him to state components at the entire compute server level? Within the proposed model, that provider is not only able to split his supercomputer, but split it in a meaningful and maybe

more profitable way such that some processors could be resold with large shared memory for each of them and the remaining procs would be offered with low memory, but still high efficiency networking and very low latency.

In addition, trading at the compute server level would introduce far larger ranges for many components such as memory and storage. By trading on the processor scale, the same 4GB/processor (for example) memory resource element can be used, and compared, for any architecture, from the single cpu cluster 1u node to a 512 sockets MMP supercomputer.

No matter how flexible the trading model is, cluster owners may persist selling their resources on the compute node scale, as co-scheduling within a box is not a challenge still entirely addressed. While one can choose to sell a dual-socket InfiniBand cluster by multiples of 2 processors, or 24 processors if the denominator is the switch fabric, another provider could instead opt to sell his quad-socket, quad-core nodes on individual contracts, targeting jobs requiring low latency for up to 16 threads. In the end, it all comes down to market positioning.

Using the processor socket as the standardized trading unit thus represents the right trade-off in terms of scale and expression syntax capability. As we will see later with more detail, for market evaluation and pricing purposes, processor count will not be treated as another type of resource but as a multiplicating factor over the atomic resources expressed in the set.

3.2.2 Contract Timing and Delivery

Recalling chapter 2, some commodities, like oil, can be traded for all 12 months and others, like CBOT wheat, have 5 delivery months a year. Some traders prefer the long-term months while others enter the sometimes more volatile short-term to active months. Although we could define a similar trading timeframe architecture for HPC resources, where any time-lapse such as a week or a month could be considered as valid "delivery" month; we choose not to take that path because such precisely bound timeframes would introduce non-linearities in pricing, and fundamentally, there is no reason for differences in HPC resources market pricing between the last day of any given period and the first day of the following.

While we could enable trading up to 3 to 5 years in advance (more that 5 years would mean trading future, very "volatile", technology[4]), we choose to limit the trading period to the next 365 days in order to facilitate the economy bootstrapping and ease the trading

[4]Such trading, especially in the form of options, could turn out to be very interesting, but is outside the scope of this section, see section 5.9 for a more detailed discussion of these aspects

framework scaling. Once the Grid market exchange has become stable, more distant futures could be enabled by the exchange management entity.

What we also have to consider is the time duration of HPC resources. Similar to electric kWh, HPC resources are a "power over a period of time" type of resource. It then all comes down to the time window to be considered as the element of trading. Even though that window could be refined to an hour, we choose to limit the trading unit to 6 hours. Once again, this is to ease the proposed Grid Exchange system deployment but also because, as of today's technology, overall scientific code scaling, on-demand provisioning and pre-fetching considerations limit the value of a 1 hour timeslot on the Grid.

Hence, at any point in time, any scientist or resource provider will be able to trade $365 * 4 = 1460$ resource timeslots.

Our application requirements expression therefore gets extended to this form:

$$\begin{aligned}\mathcal{R} = (&24_{\text{procs}}, \mathbf{4_{ts}}) \\ &@\{\text{x86}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 2.4_{\text{proc\_clock}}, 4_{\text{mem\_size}}, 2_{\text{disk\_size}}\}\end{aligned} \tag{3.15}$$

where the consumer is able to express a need for 24 contiguous hours (4 x 6 hours timeslots) of computing, on 24 processors presenting the specified characteristics.

On the provider side, the component offer expression is extended in a similar fashion:

$$\begin{aligned}\mathcal{C} = (&24, 24, 288_{\text{procs}}, \mathbf{28_{ts}}) \\ &@\{\text{x86}_{\text{proc\_arch}}, \text{clovertown}_{\text{proc\_model}}, 4_{\text{proc\_cores}}, \\ &\quad 2.0_{\text{proc\_clock}}, 8_{\text{proc\_L2}}, 4_{\text{mem\_size}}, \text{FBDDR2}_{\text{mem\_type}}, \\ &\quad 800_{\text{mem\_clock}}, 64_{\text{disk\_size}}, 128_{\text{disk\_bw}}, \text{IBDDR}_{\text{net\_tech}}, \\ &\quad 1000_{\text{net\_bw}}, 5_{\text{net\_latency}}, \text{OFED\_1.2.5}_{\text{soft\_driver}}\}\end{aligned} \tag{3.16}$$

where the provider is offering a week (28 timeslots) of computation on its resources. However, that figure must be enhanced because in a similar fashion to processor count, a provider is able to specify both the minimum number of timeslots he is willing to trade per contract, as well as a pre-defined timeslot count increment:

$$\begin{aligned}\mathcal{C} = (&24, 24, 288_{\text{procs}}, \mathbf{4}, \mathbf{2}, \mathbf{28_{ts}}) \\ &@\{\text{x86}_{\text{proc\_arch}}, \text{clovertown}_{\text{proc\_model}}, 4_{\text{proc\_cores}}, \\ &\quad 2.0_{\text{proc\_clock}}, 8_{\text{proc\_L2}}, 4_{\text{mem\_size}}, \text{FBDDR2}_{\text{mem\_type}}, \\ &\quad 800_{\text{mem\_clock}}, 64_{\text{disk\_size}}, 128_{\text{disk\_bw}}, \text{IBDDR}_{\text{net\_tech}}, \\ &\quad 1000_{\text{net\_bw}}, 5_{\text{net\_latency}}, \text{OFED\_1.2.5}_{\text{soft\_driver}}\}\end{aligned} \tag{3.17}$$

where the provider is not willing to enter any trade for less than 24 processors for 24 hours, scaling up by 24 processors or 12 hours increments all the way up 288 processors and a week of computation.

Matching bids and offers, in terms of processor counts and available timeslots, then becomes the exchange framework duty. The Grid Exchange, similar to a conventional commodity exchange, but for Grid resources, must thus match the number of processors requested in the available offers as well as it must fit the consumer's job length in the remaining contiguous timeslots if the matching provider resource slots have already been partly sold.

Of course, consumers will specify the timeframe for which the bid is valid, should the requested timeslots be anytime in December 2008 or within the third week of November, for example. After the bid gets posted on the market, the Grid Exchange will reply with the matching offers, listing them either cheapest first or earliest first. Users can then choose to trade at any provider's offer or just leave their bid on the market and wait for a provider to match it. We thus expand our notation to support bid timing:

$$\mathcal{R} = (24_{\text{procs}}, 4_{\text{ts}}, \mathbf{01/12/08...31/12/08_t}) \\ @\{\text{x86}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 2.4_{\text{proc\_clock}}, 4_{\text{mem\_size}}, 2_{\text{disk\_size}}\} \tag{3.18}$$

And provider offer expression in a similar way:

$$\mathcal{C} = (24, 24, 288_{\text{procs}}, 4, 2, 28_{\text{ts}}, \mathbf{8/12/08_t}) \\ @\{\text{x86}_{\text{proc\_arch}}, \text{clovertown}_{\text{proc\_model}}, 4_{\text{proc\_cores}}, \\ 2.0_{\text{proc\_clock}}, 8_{\text{proc\_L2}}, 4_{\text{mem\_size}}, \text{FBDDR2}_{\text{mem\_type}}, \\ 800_{\text{mem\_clock}}, 64_{\text{disk\_size}}, 128_{\text{disk\_bw}}, \text{IBDDR}_{\text{net\_tech}}, \\ 1000_{\text{net\_bw}}, 5_{\text{net\_latency}}, \text{OFED\_1.2.5}_{\text{soft\_driver}}\} \tag{3.19}$$

where the provider states the starting point for the 28 contiguous timeslots offered. Here again, while using similar expression syntax, consumer bids and provider offers are different, the first stating a period over which the requested timeslots can be fit, the latter posting a contiguous time-lapse starting on the specified date.

We have just witnessed how consumers and providers can submit their respective bids and offers to a new trading entity, the Grid Exchange, in terms of computational features (processor, memory, storage, networking and software), processor count, resource allocation time length and execution timeframe. While that was mostly an exercise in set theory, we are now ready to tackle the more fundamental economic system engineering problem.

3.3 Grid Credits: the Grid Exchange Currency

One of the fundamental principles supporting trading on any market is the exchange of a well-known, standardized and most importantly very liquid instrument against much less convertible assets. In order to trade supercomputing resources, we must therefore identify a currency to be used on this market. While any existing currency such as the US dollar or the Euro could be used as the trading counterpart, it is preferable to instantiate a new currency, specific to the HPC world: Grid Credits (gc).

The rationale leading to the use of an abstract currency comes from many reasons. First, in the beginning, the Grid market is to be used mostly by scientists willing to trade between nationwide research labs. As national research funding agencies subsidize both the scientists and the labs, this market is, by itself, a rather enclosed one that can support a private currency.

In addition, most researchers still see HPC resources as a free asset they need to be provided in order for them to pursue their scientific undertakings. While graduate students, post-docs and lab equipment must be paid for with real dollars, there is still a sense in the community that supercomputing cycles must be free. Hence, the sole consideration of using real dollars to buy HPC cycles builds walls of opposition almost instantly.

The problem of using a real currency would be even more dangerous while bootstrapping the economy. As it will inevitably be quite volatile in the beginnings, few researchers would be willing to invest their very precious research funds in such an unpredictable market. Feelings of unfairness or inequity rising from unjustified initial market movements could just kill the market's volume very quickly. The use of an abstract currency alleviates much of these considerations since that currency will have no intrinsic value before it gets traded.

In the long run, when the Grid economy has stabilized, Grid Credits will have a value on their own, a value that could then be traded back and forth against any other currency, as conventional currencies are currently traded against each other (see section 5.8).

By that time, the Grid might be ready to open to corporate markets and other industrial users. Consequently, that could lead to a sustainable financing model for academic HPC infrastructure through the reselling of cycles to corporate users.

Many consequences from the use of Grid Credits and their matching to conventional currencies can be envisioned right away. From cost optimization for both users and providers, to return on investment calculation for funding agencies, a liquid but stable Grid currency sets the table to many improvements over the actual sharing model.

3.4 Supercomputing Resources Pricing

On the stock markets, traders use both fundamental and technical information to enter and exit positions. Price over earnings, earnings per share and a plethora of other ratios are used to compare stocks against each other. While futures markets do not have such ratios, demand curves, open interest and volume are examples of guidance information that influence speculators decisions. For any type of market, such metrics build the foundations of asset valuation, or *price discovery*. In order for a Grid Exchange infrastructure to be sustainable and pervasive, similar comparative data must be provided such that both scientists and site operators can make enlightened trading decisions.

This section therefore presents an economic algorithm to extract market information from the maze of job requirement bids and supercomputer component offers in order to get the essential pricing discovery mechanism useful to both the application scientist, or consumer, and the supercomputer operator, or provider. This market pricing information enables both of them to estimate the value of their combined resources statement and steer their positioning while limiting global market volatility.

However, as mostly every single HPC contract agreement will be different, there is just no straightforward way to compare them against each other. Nevertheless, most trades will include some processor, memory, storage, networking or software element. From these sub-components market correlations, the economic algorithm presented here is able to extract pricing information that can be used to steer users and providers decisions.

As the core contribution of this thesis, the presented economic model builds the foundation to a sustainable Grid Exchange where *fairshare* is no longer some remote abstract objective but finds its roots through quantitative weighting of traded resource.

3.4.1 The Resource Index

Since mostly every transaction on the Grid Exchange will be different, we have showed that it is just impossible to define a standardized HPC contract. Therefore, in order to give a good market positioning assessment to traders, we choose to compute resource market indices for each individual requirement or component.

Thus, for every resource element in $\Phi = \Phi_{\text{proc}} \cup \Phi_{\text{mem}} \cup \Phi_{\text{disk}} \cup \Phi_{\text{net}} \cup \Phi_{\text{soft}}$, we define a resource index $\mathbf{I}_{\phi}(t)$ that represents the market valuation for any such asset over time. For

example, the 2GB/processor RAM index could run at 257 gc on the September 7th, 2008, 12:00 timeslot. These indices, reflecting the market demand and offer for any resource can then be used by both the provider and the consumer to estimate their respective resource sets. Hence, for a consumer willing to enter the market with a requirement set described in Equation 3.20, assuming there have been transactions closed prior to that market order and from which we can extract market information and thus resource indices; we then use the hypothetical resource indices in Equation 3.21 to compute that consumer's requirement set market pricing estimate:

$$\begin{aligned}\mathcal{R} = (&24_{\text{procs}}, 1_{\text{ts}}, 15/12/08 - 18:00_{\text{t}}) \\ &@\{\text{x86}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 2.4_{\text{proc\_clock}}, 4_{\text{mem\_size}}, 2_{\text{disk\_size}}\}\end{aligned} \tag{3.20}$$

where:

$$\begin{aligned}\mathbf{I}_{\text{x86}_{\text{proc\_arch}}} &= 307gc \\ \mathbf{I}_{2_{\text{proc\_cores}}} &= 35gc \\ \mathbf{I}_{2.4_{\text{proc\_clock}}} &= 80gc \\ \mathbf{I}_{4_{\text{mem\_size}}} &= 84gc \\ \mathbf{I}_{2_{\text{disk\_size}}} &= 70gc\end{aligned} \tag{3.21}$$

And therefore, this user requirement set pricing estimate, per processor and for timeslot t, $p_{\mathcal{R}}(t)$ would be:

$$\begin{aligned}p_{\mathcal{R}}(t) &= \sum_{\phi \in \Phi_{\mathcal{R}}} \mathbf{I}_{\phi}(t) \\ &= \mathbf{I}_{\text{x86}_{\text{proc\_arch}}} + \mathbf{I}_{2_{\text{proc\_cores}}} + \mathbf{I}_{2.4_{\text{proc\_clock}}} + \mathbf{I}_{4_{\text{mem\_size}}} + \mathbf{I}_{2_{\text{disk\_size}}} \\ &= 307 + 35 + 80 + 84 + 70 \\ &= 635gc\end{aligned} \tag{3.22}$$

Expanding to t timeslots and n processors, the total requirement set pricing estimate $\mathcal{P}_R$ becomes:

$$\begin{aligned}\mathcal{P}_R = n * \int_t p_{\mathcal{R}}(t) &= n * \int_t \sum_{\phi \in \Phi_{\mathcal{R}}} \mathbf{I}_{\phi}(t) \\ &= 24 * 635gc = 15240gc\end{aligned} \tag{3.23}$$

Using this price discovery mechanism, the user is then able to get a cost approximation for his requirement set. Although there is no guarantee there will be any offer in the "predicted" price range, this assessed value gives the user a starting point to establish his market positioning as resource indices reflect global market trends. Following the analysis of his

requirement set estimate, the user will evaluate the matching offers ("matching" in terms of resources and timeframe) and decide either to trade at the provider's price point or to post his bid on the market, at any price point below or over the requirement set estimate.

On the provider side, the algorithm basically works the same over component sets. However, as component sets tend to be much more extensive than requirement sets, we need to find an alternative to account for this difference.

We thus introduce the notion of *double-index* where for each resource element of Φ, the requirement index $\mathbf{I}^R_\phi(t)$ is always higher or equal to the component index $\mathbf{I}^C_\phi(t)$.

$$\forall \phi \in \Phi | \mathbf{I}^R_\phi(t) >= \mathbf{I}^C_\phi(t) \tag{3.24}$$

The motivation for a double index scheme comes from the fact that the brokering system won't match job requirements on limited node components. Indeed, in almost every transaction, the consumer will pay in some way for components not listed in his requirements, but still presenting a positive market ratio[5]. Since we must interpolate every component index when closing a deal, the requirement indices must be higher to compensate for unrequested, but still unusable by others, components. For example, in a transaction where the provider is advertising a 2.8 GHz processor but the user is requesting only a 2.2 GHz, while both peers may close the deal at some pricing middle point, which resource element should the index calculation system consider? Any of the 2.2, 2.4, 2.6 and 2.8 GHz processor clock resource elements could be considered as valid closing point. Even with *continuous* (see section 5.2) indices, closing at an eventual 2.5 GHz mark would probably not do it, as the index calculation algorithm cannot extrapolate which side "pulled the deal" more than the other, especially counting in all the other types of resources in the transaction. By considering distinct indices from the consumer and the provider standpoint, we alleviate this issue.

Following this rationale, using component indices, a provider is able to get a pricing estimate for its advertised resources:

$$\begin{aligned} \mathcal{C} = (&12, 12, 48_{\text{procs}}, 1_{\text{ts}}, 15/12/08 - 18:00_{\text{t}}) \\ @\{&\text{x86}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 2.8_{\text{proc\_clock}}, 8_{\text{mem\_size}}, 32_{\text{disk\_size}}\} \end{aligned} \tag{3.25}$$

Before translating this component expression set into a pricing estimate, we must emphasize the notion of *sub-resource inclusiveness* in provider offers. For resources like processor clock, memory size, and storage, the site operator advertised components reflect the ultimate level of service he is able to supply. While the framework does not require the provider to list exhaustively every component included by default, the component set expression inherently

[5] with a demand greater than 0

translates to more extensive forms, or series, that gets posted on the market:

$$\begin{aligned}\mathcal{C} = (&12, 12, 48_{\text{procs}}, 1_{\text{ts}}, 15/12/08 - 18:00_{\text{t}}) \\ @\{&\text{x86}_{\text{proc\_arch}}, 1_{\text{proc\_cores}}, 2_{\text{proc\_cores}}, \\ &2.0_{\text{proc\_clock}}, 2.2_{\text{proc\_clock}}, 2.4_{\text{proc\_clock}}, 2.6_{\text{proc\_clock}}, 2.8_{\text{proc\_clock}}, \\ &1_{\text{mem\_size}}, 2_{\text{mem\_size}}, 4_{\text{mem\_size}}, 8_{\text{mem\_size}}, \\ &2_{\text{disk\_size}}, 4_{\text{disk\_size}}, 8_{\text{disk\_size}}, 16_{\text{disk\_size}}, 32_{\text{disk\_size}}\}\end{aligned} \tag{3.26}$$

And thus, every component included in the set brings value to the provider offer, and consequently, every component index must be added in order to get the right processor timeslot earning $p_{\mathcal{C}}(t)$ and then the total earnings figure a provider could expect $\mathcal{P}_C$:

$$\mathcal{P}_C = n * \int_t p_{\mathcal{C}}(t) = n * \int_t \sum_{\phi \in \Phi_{\mathcal{C}}} \mathbf{I}_{\phi}^{C}(t) \tag{3.27}$$

where n reflects the maximum number of processors the provider is offering, and t the timeframe available. Since component indices are generally much lower than their requirement counterpart, the pricing estimate thus obtained can be matched against bids with higher, but less numerous, requirement indices.

As for the consumer side example, we can use some hypothetical component indices to compute provider market pricing estimate:

$$\begin{aligned}&\mathbf{I}_{\text{x86}_{\text{proc\_arch}}} = 141gc, \\ &\mathbf{I}_{1_{\text{proc\_cores}}} = 30gc, \quad \mathbf{I}_{2_{\text{pcores}}} = 32gc, \\ &\mathbf{I}_{2.0_{\text{proc\_clock}}} = 39gc, \quad \mathbf{I}_{2.2_{\text{pclock}}} = 67gc, \quad \mathbf{I}_{2.4_{\text{pclock}}} = 38gc, \quad \mathbf{I}_{2.6_{\text{pclock}}} = 11gc, \quad \mathbf{I}_{2.8_{\text{pclock}}} = 3gc, \\ &\mathbf{I}_{1_{\text{mem\_size}}} = 6gc, \quad \mathbf{I}_{2_{\text{msize}}} = 33gc, \quad \mathbf{I}_{4_{\text{msize}}} = 62gc, \quad \mathbf{I}_{8_{\text{msize}}} = 39gc, \\ &\mathbf{I}_{2_{\text{disk\_size}}} = 34gc, \quad \mathbf{I}_{4_{\text{dsize}}} = 59gc, \quad \mathbf{I}_{8_{\text{dsize}}} = 36gc, \quad \mathbf{I}_{16_{\text{dsize}}} = 8gc, \quad \mathbf{I}_{32_{\text{dsize}}} = 1gc\end{aligned} \tag{3.28}$$

hence, for this case:

$$p_{\mathcal{C}}(t) = \sum_{\phi \in \Phi_{\mathcal{C}}} \mathbf{I}_{\phi}^{C}(t) = 639gc \tag{3.29}$$

Meaning that, following historical market statistics, that provider could expect to receive about 639 gc from every processor timeslot he offers on the market.

The requirement and component indices used here were not in fact totally hypothetical, as they were taken from section 4.4 results. What it shows is that although the consumer and provider pricing estimates are based on different indices, they converge to a coherent figure of 635 to 639 gc.

The closing price at which this transaction will end up is an entire other story, as the provider could be willing to somewhat ignore market indices and sell its resources at twice the price. In the case our client would be in desperate need for such resources, he could end up paying such a figure. The other way around, the consumer could go cheap, under 300 gc for instance, and take the "utility risk" a provider would match his market order.

As we have seen earlier, the consumer may be concerned about the total cost for his requirement set over the necessary timeframe and number of processors. While in principle the provider could also be concerned about total earnings, he will most probably enter positions by stipulating a price per processor timeslot, stating $p_C(t)$ rather than $\mathcal{P}_C$. Doing so gives much more matchmaking flexibility to the trading infrastructure as it is then able to sell the provider components in sub-sets of processors and timeslots, a much more complicated task if the requested price is for the entire offer.

As important as the idea of sub-resources inclusiveness, this concept of multiple matching for provider offers is brought by the framework itself and eases significantly resource matchmaking on the Grid Exchange. Hence, for any matching bid and offer, as long as the remaining timeslots and processor counts in the partly pre-sold offer fulfill the new bid requirements, a transaction can be concluded if pricing fits. In practice, the trading system will simply truncate the available processor timeslots when a provider offer gets partly traded. As providers are able to specify the minimum and incremental counts for processors and timeslots, this gives them a adequate leverage on the structure of trades they could get in.

Last but not least, it is important to remember that market indices are simply a statistical figure, calculated from previous transaction closing prices, and are not, in any way, some definitive values to be followed, mandatorily. Computed using a simple moving average over the previous deals information, as providers and consumers steer their market positioning, resource indices will simply adapt to reflect these new realities. Following the Grid Exchange inception, as we will see throughout chapter 4, market indices initialization values are not a major concern, as they will quickly converge to reflect market events. The next sections introduce their calculation methodology, following the underlying market ratios.

3.4.2 Market Ratios

We have just seen how to use market indices to get a pricing estimate before entering a position, for both types of traders. As market indices intrinsically reflect the demand and offer trends for any type of resource, we used previously concluded transactions information to compute them. In fact, for any such previous transaction, we get several resource indices by splitting the closing price to every resource concerned in the deal, in proportion to their respective *market ratios*.

Hence, we introduce market ratios that divide the number of times a resource has been stated in the requirement sets over the number of times it has been offered in the component sets, in previous bids and offers, and over the considered timeframe. By maintaining a database with time relevant counts for every requirement $\eta_\phi^{\mathrm{R}}(t)$ and component $\eta_\phi^{\mathrm{C}}(t)$, we thus obtain the various market ratios:

$$\rho_\phi(t) = \frac{\eta_\phi^{\mathrm{R}}(t)}{\eta_\phi^{\mathrm{C}}(t)}, \quad \forall \phi \in \mathbf{\Phi} \tag{3.30}$$

These ratios are thus an indication of a resource demand divided by its offer. The question then becomes which bids and offers to consider when calculating market ratios. For sure, concluded transactions reflect true market fundamentals as in such a case, the consumer and provider have agreed. However, considering only concluded deals would fail to represent the entire picture as in some cases, many users may have been willing to buy the resources or, inversely, many providers may have posted equivalent offers and only one got to trade.

But considering every bid and offer to compute market ratios may hide an even bigger pitfall. Hence, some consumers could just post bids at a price point so low, no one would ever trade with them. The inverse is also true for providers with skyrocketing offers.

When calculating market ratios, we should therefore weight appropriately bids and offers whether or not they were in an "expected" price range, or, stated otherwise, "at the market" or not. In addition, considering a specific execution timeframe, it is logical to think that bids and offers posted for timeframes near the one considered should be weighted in with more emphasis than the ones 6 months apart. Section 5.1 studies these 2 issues and details an approach, but for now, we will just consider all bids and offers as equal when it comes to compute market ratios.

3.4.3 Resource Indices Calculation

As consumers are not paying for individual resources but rather for their requirement set as a whole and as providers are reselling the entire contract and not the individual components, we must develop an algorithm to interpolate the value brought by each independent resource to the transaction, for both sides of the deal. As deals are concluded for contracts with these many intrinsic resource elements, resource indices are therefore calculated from market demand and offer data applied on historical transaction closing prices.

Market ratios are computed using statistics on requirements and components over the set of previous bids and offers. In order to illustrate the resource index calculation, we use these 3 hypothetical previous bids:

$$\mathcal{R}_1 = (32_{\text{procs}}) \quad (3.31)$$
$$@\{\text{x86}_{\text{proc\_arch}}, 2.8_{\text{proc\_clock}}, 2_{\text{mem\_size}}, 2_{\text{disk\_size}}\}$$
$$\mathcal{R}_2 = (8_{\text{procs}}) \quad (3.32)$$
$$@\{\text{power5}_{\text{proc\_arch}}, 4_{\text{mem\_size}}, 32_{\text{disk\_size}}, \text{IBDDR}_{\text{net\_tech}}\}$$
$$\mathcal{R}_3 = (16_{\text{procs}}) \quad (3.33)$$
$$@\{\text{sparc}_{\text{proc\_arch}}, 2_{\text{proc\_cores}}, 8_{\text{mem\_size}}, 128_{\text{disk\_size}}, 128_{\text{disk\_bw}}\}$$

and 3 previous offers:

$$\mathcal{C}_1 = (8, 4, 32_{\text{procs}}) \quad (3.34)$$
$$@\{\text{sparc}_{\text{proc\_arch}}, 2.0_{\text{proc\_clock}}, 2_{\text{proc\_cores}}, 16_{\text{mem\_size}},$$
$$800_{\text{mem\_clock}}, 256_{\text{disk\_size}}, 256_{\text{disk\_bw}}, \text{GigE}_{\text{net\_tech}}\}$$
$$\mathcal{C}_2 = (8, 8, 48_{\text{procs}}) \quad (3.35)$$
$$@\{\text{x86}_{\text{proc\_arch}}, 3.0_{\text{proc\_clock}}, 2_{\text{proc\_cores}}, 4_{\text{mem\_size}},$$
$$667_{\text{mem\_clock}}, 32_{\text{disk\_size}}, 32_{\text{disk\_bw}}, \text{GigE}_{\text{net\_tech}}\}$$
$$\mathcal{C}_3 = (8, 8, 8_{\text{procs}}) \quad (3.36)$$
$$@\{\text{power5}_{\text{proc\_arch}}, 2.2_{\text{proc\_clock}}, 1_{\text{proc\_cores}}, 8_{\text{mem\_size}},$$
$$667_{\text{mem\_clock}}, 64_{\text{disk\_size}}, 64_{\text{disk\_bw}}, \text{IBDDR}_{\text{net\_tech}}\}$$

Presenting relatively limited requirements in terms of memory, storage and interconnect, $\mathcal{R}_1$ reflects a typical embarrassingly-parallel application. With higher requirements for memory and storage but most notably necessitating an InfiniBand DDR interconnect, $\mathcal{R}_2$ points towards the tightly-coupled profile. As for $\mathcal{R}_3$, the large memory size, disk size and disk bandwidth expressed in the set depict some form of data-driven code.

On the provider side, the machine offered in $\mathcal{C}_2$ is a GigE x86 serial cluster, matching the needs expressed in $\mathcal{R}_1$. While the components in $\mathcal{C}_3$ portray a much higher-end machine, we can easily establish that the InfiniBand interconnect could have made the deal with $\mathcal{R}_2$. In the case of $\mathcal{C}_1$, the important storage bandwidth and capacity are probably the expression a SAN based storage infrastructure linked to the compute nodes.

For all 3 deals, different processor architectures have been specified in order to illustrate the difference of such mutually exclusive resources over memory and storage, that can be mixed and matched in any deal. Also worth noting is the absence of timeframe information, as such information is not relevant, for now, for index calculation purposes.

Using these 6 previous bids and offers, that might have ended up being traded together (this information is irrelevant for ratio calculation purposes), we compute market ratios in for timeframe t_1 in table 3.1. Because we consider so few events (only 6), it leads to many non-existent ratios. Over a more realistic and much more extensive set of previous events, most ratios would be non-zero.

A new bid market order then gets posted to the Exchange:

$$\begin{aligned}\mathcal{R}_4 = (24_{\text{procs}}) \\ @\{\text{x86}_{\text{proc\_arch}}, 2.4_{\text{proc\_clock}}, 2_{\text{proc\_cores}}, 4_{\text{mem\_size}}, 2_{\text{disk\_size}}\}\end{aligned} \tag{3.37}$$

Just after, an new offer enters the market and end up trading with $\mathcal{R}_4$ at 1000 gc:

$$\begin{aligned}\mathcal{C}_4 = (12, 12, 48_{\text{procs}}, 1_{\text{ts}}, 15/12/08 - 18:00_{\text{t}}) \\ @\{\text{x86}_{\text{proc\_arch}}, 2.8_{\text{proc\_clock}}, 2_{\text{proc\_cores}}, 8_{\text{mem\_size}}, 32_{\text{disk\_size}}\}\end{aligned} \tag{3.38}$$

from these 2 events, we can update our ratios from timeframe t_1 to this transaction timeframe t_2, as shown in columns 5 to 7 of table 3.1.

From the newly updated ratios, we can compute requirement and component market indices by proportionally retro-propagating every single deal price $\Pi_\$$:

$$\mathbf{I}^R_\phi(t) = \frac{\rho_r(t)}{\sum_{\phi \in R} \rho_\phi(t)} \times \Pi_\$ \tag{3.39}$$

$$\mathbf{I}^C_\phi(t) = \frac{\rho_c(t)}{\sum_{\phi \in C} \rho_\phi(t)} \times \Pi_\$ \tag{3.40}$$

Recall that since requirement sets are always smaller or equal to component sets for any given transaction, we have to compute distinct resource indices for them. Taking the previous

Resource	$\eta_\phi^R(t_1)$	$\eta_\phi^C(t_1)$	$\rho_\phi(t_1)$	$\eta_\phi^R(t_2)$	$\eta_\phi^C(t_2)$	$\rho_\phi(t_2)$	$\mathbf{I}_\phi^R(t_2)$	$\mathbf{I}_\phi^C(t_2)$
$\text{x86}_{\text{proc\_arch}}$	32	48	0.67	56	96	0.58	324	206
$\text{power}_{\text{proc\_arch}}$	8	8	1.00	8	8	1.00		
$\text{sparc}_{\text{proc\_arch}}$	16	32	0.50	16	32	0.50		
$2.0_{\text{proc\_clock}}$	0	88	0.00	0	136	0.00		0
$2.2_{\text{proc\_clock}}$	0	56	0.00	0	104	0.00		0
$2.4_{\text{proc\_clock}}$	0	48	0.00	24	96	0.25	140	89
$2.6_{\text{proc\_clock}}$	0	48	0.00	0	96	0.00		0
$2.8_{\text{proc\_clock}}$	32	48	0.67	32	96	0.33		117
$3.0_{\text{proc\_clock}}$	0	48	0.00	0	48	0.00		
$1_{\text{proc\_cores}}$	0	88	0.00	0	136	0.00		0
$2_{\text{proc\_cores}}$	16	80	0.20	40	128	0.31	173	110
$4_{\text{proc\_cores}}$	0	0	0.00	0	0	0.00		
$1_{\text{mem\_size}}$	0	88	0.00	0	136	0.00		0
$2_{\text{mem\_size}}$	32	88	0.36	32	136	0.24		85
$4_{\text{mem\_size}}$	8	88	0.09	32	136	0.24	134	85
$8_{\text{mem\_size}}$	16	40	0.40	16	88	0.40		142
$16_{\text{mem\_size}}$	0	32	0.00	0	32	0.00		
$533_{\text{mem\_clock}}$	0	88	0.00	0	88	0.00		
$667_{\text{mem\_clock}}$	0	88	0.00	0	88	0.00		
$800_{\text{mem\_clock}}$	0	32	0.00	0	32	0.00		
$2_{\text{disk\_size}}$	32	88	0.36	56	136	0.41	229	145
$4_{\text{disk\_size}}$	0	88	0.00	0	136	0.00		0
$8_{\text{disk\_size}}$	0	88	0.00	0	136	0.00		0
$16_{\text{disk\_size}}$	0	88	0.00	0	136	0.00		0
$32_{\text{disk\_size}}$	8	88	0.09	8	136	0.06		21
$64_{\text{disk\_size}}$	0	40	0.00	0	40	0.00		
$128_{\text{disk\_size}}$	16	32	0.50	16	32	0.50		
$256_{\text{disk\_size}}$	0	32	0.00	0	32	0.00		
$32_{\text{disk\_bw}}$	0	88	0.00	0	88	0.00		
$64_{\text{disk\_bw}}$	0	40	0.00	0	40	0.00		
$128_{\text{disk\_bw}}$	16	32	0.50	16	32	0.50		
$256_{\text{disk\_bw}}$	0	32	0.00	0	32	0.00		
$\text{GigE}_{\text{net\_tech}}$	32	48	0.67	32	48	0.67		
$\text{IBDDR}_{\text{net\_tech}}$	8	8	1.00	8	8	1.00		

Table 3.1: Example of resource statistics.

example resource sets $\mathcal{R}_4$ and $\mathcal{C}_4$ for a transaction concluded at $1000gc$, we can compute both $\mathbf{I}^R_{4_{\text{mem\_size}}}$ and $\mathbf{I}^C_{4_{\text{mem\_size}}}$ market indices at deal time t_2:

$$\mathbf{I}^R_{4_{\text{mem\_size}}}(t_2) = \frac{0.24}{1.79} \times 1000gu = 134gc \tag{3.41}$$

$$\mathbf{I}^C_{4_{\text{mem\_size}}}(t_2) = \frac{0.24}{2.82} \times 1000gu = 85gc \tag{3.42}$$

Looking at table 3.1, we can observe that all 5 requirements expressed by the bidder were attributed a requirement index. Following market trends expressed through ratios, we can see that of the $1000gc$ paid, almost 1/3 probably went for the x86 architecture, because of an overall high demand for it. While every provider could supply 2GB of storage per processor, a fact that could have led to a low ratio; once again, a large number of bids required it, pushing the ratio up to 0.41. Looking at the various ratios for t_1 and t_2, it could be hard to believe that a 2GB of storage per processor presents an equal or even higher value than 4GB or 8GB. But it all comes down to demand and offer, if every provider is offering 32 GB / proc, there is just no differentiating factor from 2GB to 32GB. From a consumer standpoint, if no one is interested in more that 2GB, there is simply no value in having more than 2GB.

Intuitively, any resource requirement index will always be higher than its component index because, for any deal, component sets tend to be more extensive than requirement sets. For very specific resources like BLAST permutations, for example, equal indices could be computed in the case where all jobs required only this feature and all sold nodes provided it without any other component in the set.

Resource market indices are therefore calculated dynamically over time. From a system point of view, the real-time value for an index could be simply its last deal value or a moving average of relevant deal values in time. Section 5.2 further details time-interpolation of indices, for now, we will consider market indices are averaged over every past transaction.

Remembering the rationale supporting the double-index scheme came from the fact that requirement sets are *necessary*, and thus less extensive than component sets, stating the provider's *available* resources. Thus, for mostly every HPC "contract" to be exchanged, there won't be a 1 to 1 mapping between the requirement set and the component set, and therefore a dual indices interpolation methodology had to be implemented in order to reflect this reality.

We have now seen how to extract resource pricing indices from previously concluded transactions and how to use them to estimate any user or provider market positioning. The next chapter demonstrates this methodology implementation in simulations of several thousand events.

Chapter 4

Model Simulation and Analysis

This chapter presents a simulation and its results for the economic model presented previously in this thesis. In order to demonstrate the inherent pricing mechanism effectiveness, 8 different parameters were used. Processor architecture, processor clock frequency, memory size per processor, memory speed, available storage capacity (per processor), storage bandwidth (per processor), network bandwidth (per processor) and available software components represent the fundamental resource types that could eventually be traded on a Grid Exchange. While the simulation focuses on these 8 resource types, nothing prevents the addition of various other HPC features for trading and any resource range could be expanded without loss of generality.

For each resource, market ratios and indices are computed following the equations presented in chapter 3. Using these indicators, market estimates for aggregated bid requirements or offer components are computed. As it will be shown in the following sections, these market pricing estimates are rather accurate, and confirm the theoretical claim that multi-commodity trading can be achieved, analyzed and forecasted, using the proposed model.

The various resource types and related intrinsic instantiations chosen for simulation purposes do not reflect any subjective interest of any kind. NEC processors, IBM's Power, Sun's Sparc as well as the general x86 architecture where used to depict a trading system close to today's reality. While their choice and simulation significance could be discussed in many ways, this debate would not serve any real purpose as these simulated resources sole objective is to demonstrate the model usefulness.

4.1 Generating Bids and Offers

In this simulated market instantiation, each bid and offer is a set of 8 elements such that:

$$\Phi_{bid} = [\phi_{proc\_arch}, \phi_{proc\_clock}, \phi_{mem\_size}, \phi_{mem\_freq}, \phi_{stor\_size}, \phi_{stor\_bw}, \phi_{net\_bw}, \phi_{soft}] \quad (4.1)$$

and

$$\Phi_{offer} = [\phi_{proc\_arch}, \phi_{proc\_clock}, \phi_{mem\_size}, \phi_{mem\_freq}, \phi_{stor\_size}, \phi_{stor\_bw}, \phi_{net\_bw}, \phi_{soft}] \quad (4.2)$$

For each resource type, the following instantiations are possible:

$$\begin{aligned} \Phi_{proc\_arch} &= [\text{x86; Power; Sparc; NEC}] \quad (4.3)\\ \Phi_{proc\_clock} &= [\text{2.0; 2.2; 2.4; 2.6; 2.8; 3.0}](\text{GHz})\\ \Phi_{mem\_size} &= [\text{1; 2; 4; 8; 16; 32; 64}](\text{GB})\\ \Phi_{mem\_speed} &= [\text{533; 667; 800; 1066; 1333}](\text{MHz})\\ \Phi_{stor\_size} &= [\text{2; 4; 8; 16; 32; 64; 128; 256}](\text{GB})\\ \Phi_{stor\_bw} &= [\text{16; 32; 64; 128; 256; 512}](\text{MB/s})\\ \Phi_{net\_bw} &= [\text{5; 10; 50; 100; 500; 1000; 2000}](\text{MB/s})\\ \Phi_{soft} &= [\text{GLOBUS; GROMACS; BLAST; FLUENT}] \end{aligned}$$

Using every specific resource position within their respective resource sets, we can generate bids and offers such that:

$$\Phi_{bid} = \Phi_{offer} = [[1...4], [1...6], [1...7], [1...5], [1...8], [1...7], [1...7], [1...4]] \quad (4.4)$$

where, for instance:

$$\Phi_{bid} = [1, 3, 4, 2, 7, 3, 6, 1] \quad (4.5)$$

would be the expression for a bid requiring an x86 processor running at 2.4 GHz (or higher), with at least 8 GB memory per processor (at 667 MHz), 128 GB of storage capacity per processor, 64 MB/s of I/O bandwidth, 2000 MB/s of interconnect bandwidth between the sockets and the GLOBUS package installed on the nodes.

That bid could get matched against an offer presenting the following components:

$$\Phi_{offer} = [1, 5, 5, 3, 7, 4, 6, 1] \quad (4.6)$$

Or, in a more extensive form:

$$\Phi_{offer} = [\text{x86, 2.8, 16, 800, 128, 128, 1000, GLOBUS}] \quad (4.7)$$

While these 2 examples had a specified requirement or component for each of the 8 resources, this is not mandatory and bid requirements could be expressed using 0 when there is no required level of service for a given resource. Hence:

$$\Phi_{bid} = [1, 0, 2, 0, 1, 0, 0, 0] \tag{4.8}$$

Would translate a much less demanding requirement set for an x86 processor with 2 GB of memory and 2GB of hard disk capacity. On the component side, although offers will tend to be more expressive than bids, it is possible for a provider not to provide any level of service for a given type of resource.

Obviously, the processor architecture must match in order for a deal to be possible. Although some higher abstraction processes depending more on a specific software framework than on the processor architecture (and thus expressing ϕ_{proc_arch} with 0) could conceivably be traded, this simulation instantiates a specific processor architecture for each bid and offer. The reason supporting that fact is to witness the behavior of a type of resource widely expressed and matched one to one.

Software resources are considered in a similar fashion with the exception that not every bid and offer will present such a requirement or component. In a real market instantiation, multiple software resources could be asked or made available within a single bid or offer, but since that would not bring any significant additional data for simulation analysis, we choose to limit software instances to 1.

For all other 6 resource types, as long as the component level of service expressed in the offer is higher or equal to the level of service required in the bid, a transaction can be concluded.

Hence, for any simulation run, we randomly generate a population of bids and offers where for each resource type, the resource instantiation expectation is given by Figure 4.1. As we can see, every single bid and offer will specify a processor architecture, while 80% of bids will require some minimum processor clock and 90% of offers will make it available. Some resources, like memory speed, may be less significant as mostly every provider will offer it (90%) while very few bids (10%) will require it.

For every resource in a bid or in an offer that has been flagged for instantiation following the probabilities in Figure 4.1, the simulation software then uses the probability distributions presented in Figure 4.3 to determine which level of service to instantiate.

As we can can observe in Figure 4.3, the processor architecture distribution expresses an under-demand for x86 (40% demand over 50% offer) while the NEC architecture is under

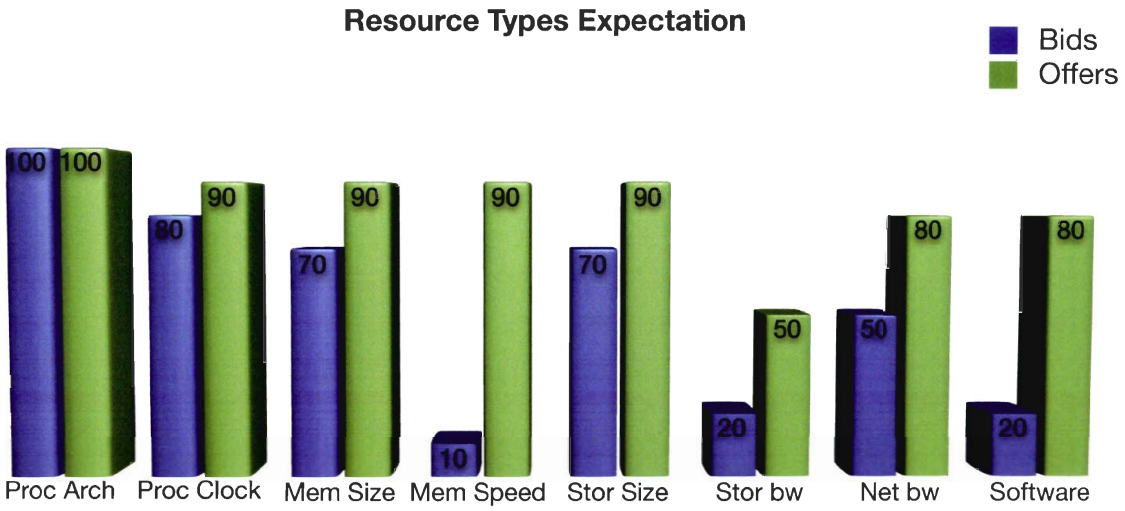

Figure 4.1: *Resource types expectation for bids and offers*

high market pressure in a 2 to 1 ratio (20% vs 10%). Both the Power and Sparc architectures are balanced (20% to 20%) . Of course, on both sides, summing the various probabilities adds up to 100%. For software resources, it is a similar scenario with GROMACS and FLUENT presenting high and low demand, respectively.

For the other 6 resource types, we use discrete gaussian distributions centered on a given element (μ) and with a variance of 1 (σ^2), then normalized such that the sum of elements adds up to 100%. For most resources, offers are randomly distributed at a slightly higher point than bids in order to mimic a typical market. Another aspect worth noting is that any offer component, for these 6 resource types, intrinsically covers every sub-level, as they can be referred to as *series*, rather than unordered *sets*. Therefore, the memory size probability distribution, for example, looks more like Figure 4.2.

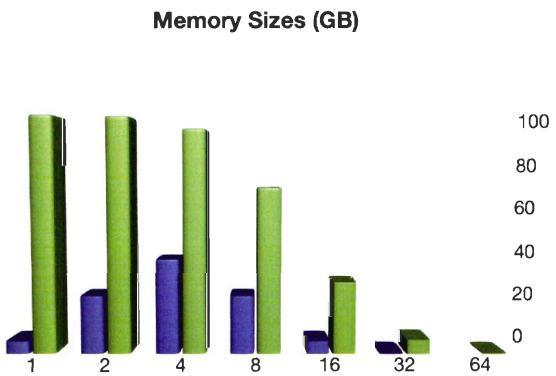

Figure 4.2: *Effective memory sizes probability distribution*

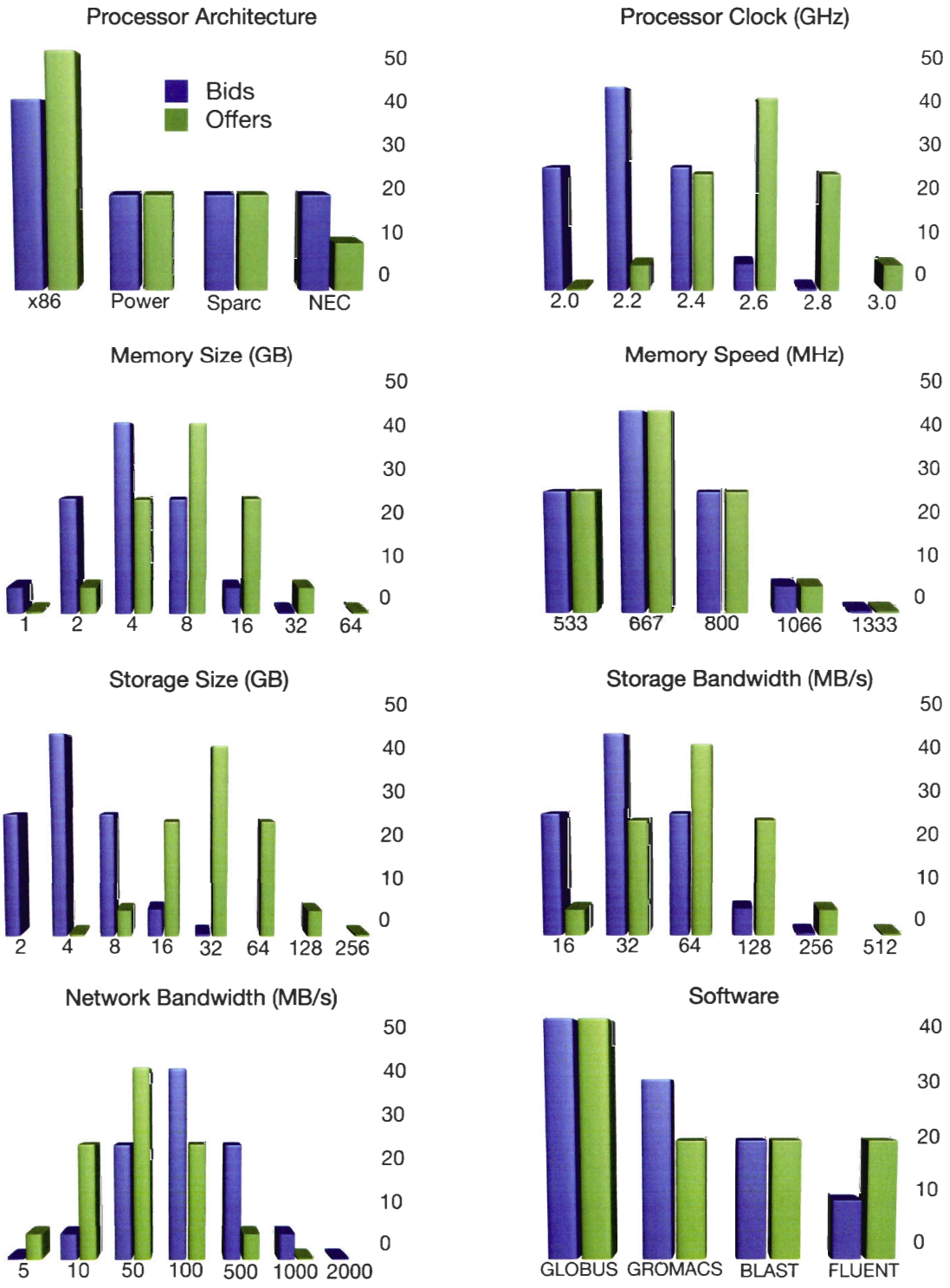

Figure 4.3: *Probability distributions for bids and offers*

4.2 Simulation Parameters

The simulation main function (`runmarket(nbevent, ratio, ratiomem, dealsmem, bidtol, offertol, initflag)`) takes 7 parameters for each execution.

The first parameter, `nbevent` specifies the number of events to randomly generate. For example, in a simulation run of 10 000 events, the framework will randomly generate a bid or an offer, check if it matches against a previous (and unmatched) corresponding offer or bid and then move on to the the next iteration.

The `ratio` parameter steers the market around buyers or sellers; a 0.5 ratio being a balanced market, 0.1 meaning there are 10% buyers vs 90% sellers and vice-versa. For our 10 000 events example run, a 0.5 ratio would lead to approximately 5 000 bids and 5 000 offers.

Parameters `ratiomem` and `dealsmem` tweak the system entropy, trying to limit volatility without hampering too much system adaptation to changing market conditions. Section 4.7 is dedicated to their study.

Parameter `initflag` is used to either initialize (when set to 1) market ratios and indices to their default values or, when disabled, to use the previous simulation run data. By default, market ratios are initialized to 0 (Figure 4.4) while both requirement and component indices are initialized to 100 *gc* (figures 4.5 and 4.6). In order to limit market instability in the beginning, these bootstrapping values are further studied in section 4.5 . For now, trying to establish a more representative market index for each resource type would probably lead us to the same kind of debate that comes to place when trying to decide which supercomputer presents the best value for any given group of scientists and applications. In the end, we will show that it does not matter at all as the economic model stabilizes rather quickly to values that reflect true market fundamentals.

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
	0	0	0	0	0	0	
	0	0		0	0	0	
		0		0		0	
				0			

Figure 4.4: *Initial ratios*

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	100.00	100.00	100.00	100.00	100.00	100.00	
	100.00	100.00		100.00	100.00	100.00	
		100.00		100.00		100.00	
				100.00			

Figure 4.5: *Bootstrapping values for requirement indices*

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
	100	100	100	100	100	100	
	100	100		100	100	100	
		100		100		100	
				100			

Figure 4.6: *Bootstrapping values for component indices*

The last 2 parameters `bidtol` and `offertol` express the tolerance of buyers and sellers over the market estimate computed by the framework. To establish the highest trading price on the buy side, the bid's market estimate is simply multiplied by `bidtol`. Similarly, we simply multiply an offer estimate by `offertol` (most frequently equal to 1 / `bidtol`) to get the lowest acceptable closing price for the seller. A bid, for instance, might get a market estimate of 1 000 *gc*, if `bidtol` is set to 1.2, that bid would automatically trade with the lowest matching offer, as long as that offer price is equal or lower than 1 200 *gc*. On the other side, if an offer component set gets estimated to 1 500 *gc*, and `offertol` is set to 0.90, that offer will get matched against the highest bidder, and a trade will occur if that bid price is higher or equal to 1 350 *gc*. This parameter is very useful to study traders confidence in the market and its indices. At the beginning, tolerances must be high in order to compensate for market volatility; therefore, `bidtol` and `offertol` are initialized to 5.0 (500%) and 0.2 (80%), meaning that bids can trade at up to 5 times their market estimate while offers can cut 80% of theirs, and trade at 1/5th. After a few hundred iterations, tolerances can be lowered without hurting the number of concluded transactions.

Over the next few sections, we will start from these default values and then modify each of them independently in order to evaluate their impact on the various market health indicators, such as volatility versus stability, ratios representability, pricing estimates accuracy, number of concluded transactions, etc.

4.3 Step by Step Market Startup

Throughout the next sections, we will be simulating the market's behavior for an undefined timeframe, as the system inner workings are just the same, for every trading period to be considered. Hence, timing information is put aside for now, but will be revisited in section 5.2.

We then start by initializing ratios and indices to their default values to go ahead with the random generation of bids and offers. For demonstration purposes, lets analyze the first 10 random events of an hypothetical simulation run:

1. $\Phi_{offer_1} = [2, 4, 0, 2, 6, 0, 0, 0]@1300gc$
   Since all component indices are initialized to 100 gc, it leads us to a 1300 gc pricing estimate for this offer since:

$$\begin{aligned}\mathcal{P}_C &= \sum_{\phi \in \Phi_C} \mathbf{I}_\phi^C \\ &= I^C_{\text{power}_{proc\_arch}} \\ &\quad + I^C_{2.0_{proc\_clk}} + I^C_{2.2_{proc\_clk}} + I^C_{2.4_{proc\_clk}} + I^C_{2.6_{proc\_clk}} \\ &\quad + I^C_{533_{mem\_spd}} + I^C_{667_{mem\_spd}} \\ &\quad + I^C_{2_{stor\_sz}} + I^C_{4_{stor\_sz}} + I^C_{8_{stor\_sz}} + I^C_{16_{stor\_sz}} + I^C_{32_{stor\_sz}} + I^C_{64_{stor\_sz}} \\ &= 1300gc\end{aligned} \tag{4.9}$$

2. $\Phi_{bid_1} = [1, 2, 5, 0, 3, 0, 3, 0]@500gc$
   While an offer has to account every single component an its implicit sub-levels of service, bid requirements pricing estimates are calculated using the exact service level required. Then:

$$\begin{aligned}\mathcal{P}_R &= \sum_{\phi \in \Phi_\mathcal{R}} \mathbf{I}_\phi^R \\ &= I^R_{\text{x86}_{proc\_arch}} + I^R_{2.2_{proc\_clk}} + I^C_{16_{mem\_sz}} + I^C_{8_{stor\_sz}} + I^C_{50_{net\_bw}} \\ &= 500gc\end{aligned} \tag{4.10}$$

   As there is no matching offer for this bid, the simulation framework generates a few other random events.

3. $\Phi_{offer_2} = [2, 5, 4, 3, 4, 2, 3, 1]@2300gc$

4. $\Phi_{bid_2} = [2, 1, 0, 0, 2, 0, 4, 0]@400gc$

5. $\Phi_{offer_3} = [1, 5, 0, 1, 0, 4, 4, 1]@1600gc$

6. $\Phi_{offer_4} = [1, 5, 6, 2, 5, 0, 0, 2]@2000gc$

7. $\Phi_{offer_5} = [1, 5, 5, 0, 7, 1, 3, 4]@2300gc$
   Once that offer gets posted on the exchange, the matching bid_1 is immediately found: $\Phi_{bid_1} = [1, 2, 5, 0, 3, 0, 3, 0]@500gc$
   Since `offertol` is initialized to 20%, a trade is concluded at the bidder's price (as the offer just entered the market): 500 gc. While presenting a pricing estimate error of 78%, it is still under the limit specified by the provider and trading therefore occurs.

   At that point in time, ratios have evolved to:

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
0.33	0.20	0	0	0	0	0	0
0.50	0.20	0	0	0.25	0	0	0
0	0	0	0	0.25	0	0.33	0
0	0	0	0	0	0	1.00	0
	0	0.50	0	0	0	0	
	0	0		0	0	0	
		0		0		0	
				0			

Figure 4.7: *Ratios after a few (7) events*

And requirement indices get updated accordingly (only affected indices are displayed in Figure 4.8, others remain at the default 100gc value):

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
102.48							
	62.11						
				77.65		102.48	
		155.28					

Figure 4.8: *Calculated requirement indices from the first transaction*

Where, for instance:

$$I^R_{16_{mem\_sz}} = 500gc * \frac{0.50}{0.33 + 0.20 + 0.50 + 0.25 + 0.33} = 155.28gc \tag{4.11}$$

Component indices are also calculated using equations developed in chapter 3:

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
80.10	48.54	0		0	0	0	
	48.54	0		60.68		0	
	0	0		60.68		80.10	
	0	0		0			0
	0	121.36		0			
				0			
				0			

Figure 4.9: *Calculated component indices from the first transaction*

Again, for example:

$$I^{C}_{16_{mem\_sz}} = 500gc * \frac{0.50}{0.33 + 0.20 + 0.20 + 0.50 + 0.25 + 0.25 + 0.33} \\ = 121.36gc \tag{4.12}$$

As witnessed on the component indices table 4.9, although this provider offered high levels of service in terms of processor clock and storage size, the market, at that point in time, had no demand for it. Hence, the concerned ratios equalled 0 and back-propagating the transaction price assigned no value to these higher level components. The other way around is also true, as shown by the memory size, which saw no demand for other service levels than 16 GB per processor.

In practice, both bid and offer pricing estimates use an average of their respective deal calculated indices over the period specified by the `dealsmem` parameter. When component indices are set to 0, they are just ignored when computing the relevant average and following offer pricing estimate. Hence, the next bid or offer to enter the market would be estimated using the requirement indices in table 4.10 or component indices in 4.11.

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
102.48	100.00	100.00	100.00	100.00	100.00	100.00	100.00
100.00	62.11	100.00	100.00	100.00	100.00	100.00	100.00
100.00	100.00	100.00	100.00	77.65	100.00	102.48	100.00
100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	100.00	155.28	100.00	100.00	100.00	100.00	
	100.00	100.00		100.00	100.00	100.00	
		100.00		100.00		100.00	
				100.00			

Figure 4.10: *Averaged requirement indices after first transaction*

For demonstration purposes, we continue generating events until another transaction occurs:

8. $\Phi_{bid_3} = [2, 3, 4, 0, 2, 0, 3, 1]@604.96gc$

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
80.10	48.54	100	100	100	100	100	100
100	48.54	100	100	60.68	100	100	100
100	100	100	100	60.68	100	80.10	100
100	100	100	100	100	100	100	100
	100	121.36	100	100	100	100	
	100	100		100	100	100	
		100		100		100	
				100			

Figure 4.11: *Averaged component indices after first transaction*

9. $\Phi_{bid_4} = [2, 1, 0, 0, 3, 0, 0, 0]@277.65gc$
   That bid could get traded with $offer_2$, but since $offer_2$ pricing is not within 500% of bid_1 estimate, no deal occurs. This market behavior is totally appropriate as $offer_2$ reflects a rather extensive component set while bid_1 is minimal. Hence, the concerned provider would probably prefer trading with a more demanding customer while our bid_1 user would rather shoot for cheaper resources.

10. $\Phi_{bid_5} = [2, 2, 4, 3, 1, 1, 2, 0]@662.11gc$
    Again using updated requirement indices of table 4.8, this bid gets matched against $offer_2$, within the 500% `bidtol`, and a transaction is concluded at 2300 gc.

    As ratios are now:

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
0.33	0.40	0	0	0.25	0.33	0	0.50
2.00	0.40	0	0	0.50	0	0.33	0
0	0.20	0	1.00	0.50	0	0.67	0
0	0	0.67	0	0	0	1.00	0
	0	0.50	0	0	0	0	
	0	0		0	0	0	
		0		0		0	
				0			

Figure 4.12: *Ratios after 10 events*

Back propagating the 2300 gc transaction price over concerned resources leads to requirements indices in table 4.13 and component indices in table 4.14.

For future transactions, we average transactional requirement indices to obtain table 4.15. As we can see, $I^R_{2.0_{proc\_clk}}$, the only requirement concerned in both deals, got averaged to $(62.11gc + 184.74gc)/2 = 123.43gc$

The same averaging mechanism applies to component indices, as shown in table 4.16 (bold values).

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
				115.46	152.41		
923.69	184.74					152.41	
			461.85				
		309.44					

Figure 4.13: *Transaction #2 closing price back-propagation over requirement indices*

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
	118.71	0	0	74.20	98.92	0	148.39
593.55	118.71	0	0	148.39	0	98.92	
	59.35	0	296.77	148.39		197.85	
	0	197.85		0			
	0						

Figure 4.14: *Transaction #2 closing price back-propagation over component indices*

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
102.48	100.00	100.00	100.00	115.46	152.41	100.00	100.00
923.69	123.43	100.00	100.00	100.00	100.00	152.41	100.00
100.00	100.00	100.00	461.85	77.65	100.00	102.48	100.00
100.00	100.00	309.44	100.00	100.00	100.00	100.00	100.00
	100.00	155.28	100.00	100.00	100.00	100.00	
	100.00	100.00		100.00	100.00	100.00	
		100.00		100.00		100.00	
				100.00			

Figure 4.15: *Averaged requirement indices after 10 events and 2 transactions*

Proc Arch	Proc Clock	Mem Size	Mem Spd	Stor Size	Stor bw	Net bw	Soft
80.10	83.63	100	100	74.2	98.92	100	148.39
593.55	83.63	100	100	104.54	100	98.92	100
100	59.35	100	296.77	104.54	100	198.98	100
100	100	197.85	100	100	100	100	100
	100	121.36	100	100	100	100	
	100	100		100	100	100	
		100		100		100	
				100			

Figure 4.16: *Averaged component indices after 10 events and 2 transactions*

4.4 Default Simulation Run

As starting point, we consider a base case simulation run with the following parameters:
`runmarket(10 000, 0.5, 2000, 2000, 5.0, 0.2, 1)`
where we run the system for 10 000 events, with a ratio of 0.5 leading to approximately 5 000 bids and 5 000 offers, a system memory of 2000 events (bids and offers) and 2000 transactions, where bids can trade at up to 5 times their market estimate while offers can trade at 1/5th, in order to bootstrap the market and cope for initial market volatility and where indices and ratios are initialized to their default value.

Figure 4.17 exhibits market ratios for the 4 processor architectures and their respective market indices. While bids and offers are generated randomly, they converge on the expected ratios following the distributions in Figure 4.3; the NEC vector processor architecture presenting a 2 for 1 demand ratio, while Power and Sparc are at 1, and x86 at 4/5.

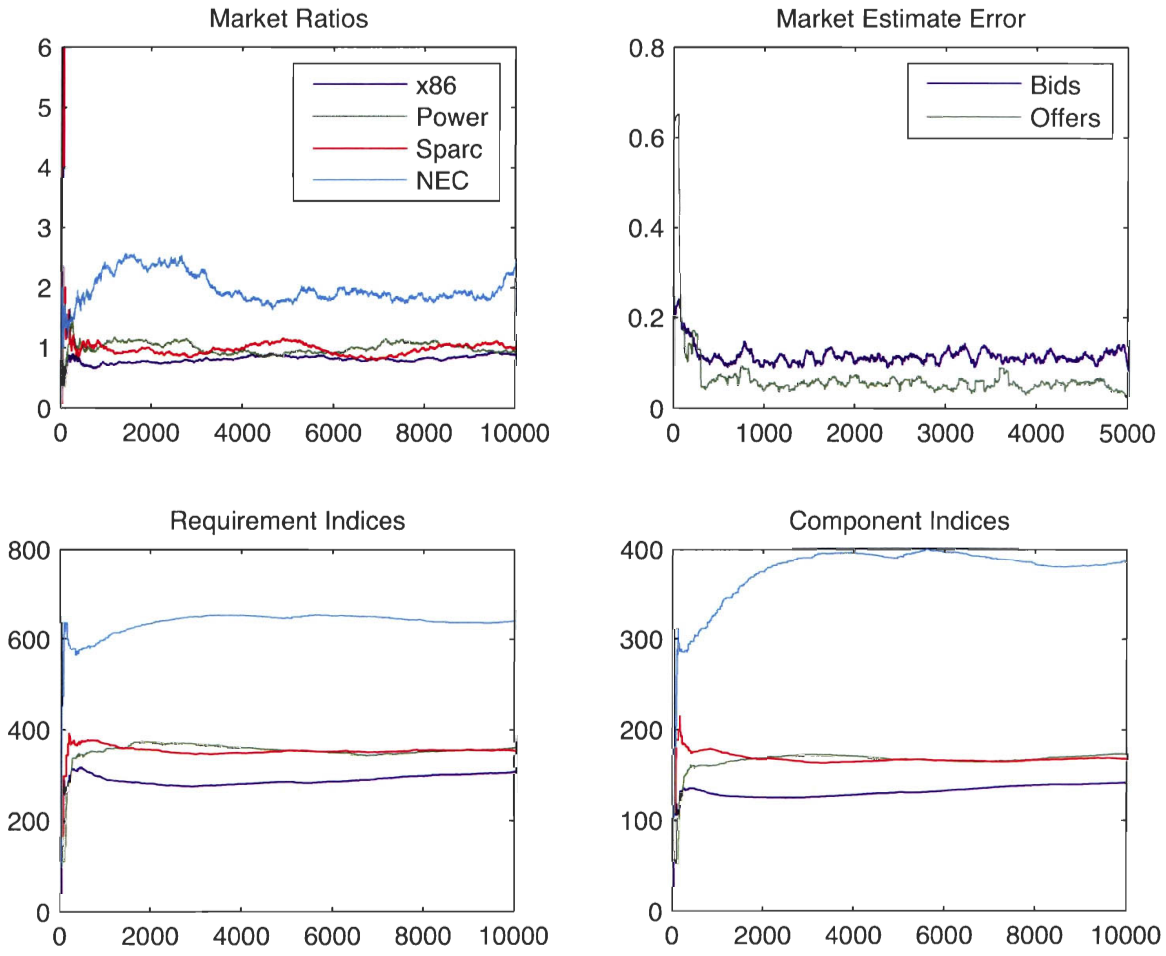

Figure 4.17: *CPU types ratios, indices and market estimate error for 10 000 events*

Since bids and offers are generated randomly, ratios vary over time, and the more events for any resource type, the less volatile its ratio becomes. Hence, comparing NEC (10% / 20%) to x86 (40% / 50%), the least common cpu architecture depicts a more volatile behavior.

Looking at requirement and component indices, we notice all 4 processor architectures reach an equilibrium point reflecting global market tendencies. The observation that component indices equilibrium tend to be one half of their respective requirement indices has no meaning in itself, it simply reflects the market instantiation from the 47 resource sub-types and their bid and offer probability distributions.

The market estimate error graph in 4.17 depicts the system's prediction accuracy, in terms of relative error from the market estimate to the transaction closing price (0.2 being 20%) , for users requirement sets (in blue) or providers component sets (in green). As shown on Figure 4.18, we can see that bids entering the market trade on the lower range while newly posted offers take the upper portion of the graph. This is totally normal as a user entering the market will close the transaction with the cheapest provider matching his bid. Similarly, the provider will maximize his profits by trading with the highest bidder he can satisfy. While managing multiple commodities at once, this system is equivalent to a commodity market pit where only the cheapest short and the highest long get screamed out loud. As tolerances are very high (20% of pricing estimate for offers and 500% for bids), closing prices are distributed on a rather wide range.

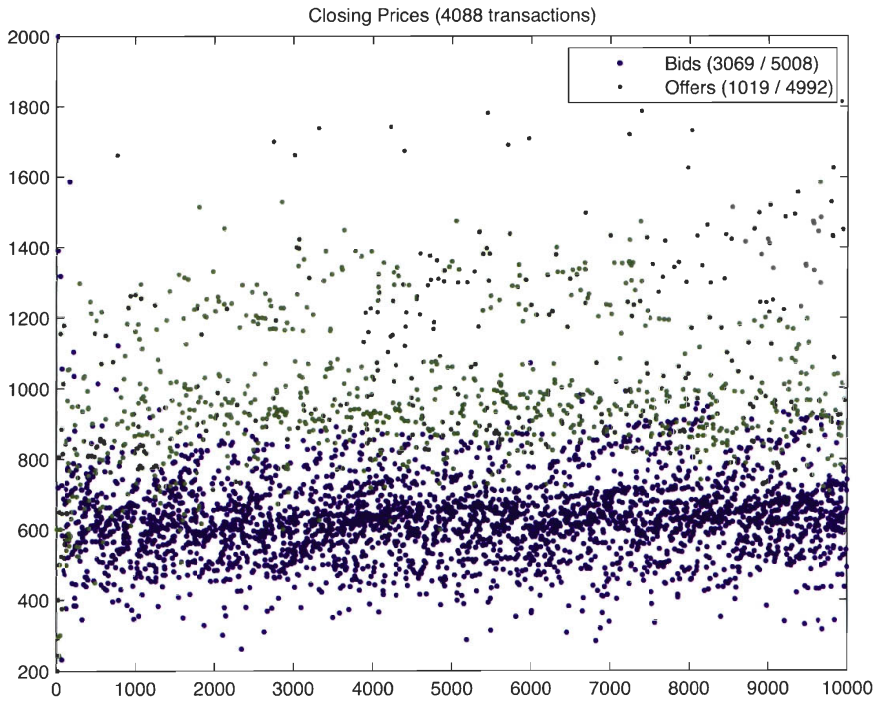

Figure 4.18: *Transaction closing prices for 10 000 events*

Over 10 000 events, in this case 5008 bids and 4992 offers, 4088 transactions occurred. This means that 82% of generated events ended up being traded, and only 18% ended up orphan. As we will see later, adjusting tolerances will have some influence on the percentage of concluded transactions, but for now, we can simply witness the effectiveness of this multi-commodity trading system. In addition, it is worth noting market stability, from indices graphs but also through closing prices in Figure 4.18, where no unjustified inflation occurs.

Zooming on the first 1000 events (Figure 4.19) gives a better view of the initial market volatility that cannot be avoided when bootstrapping an economy. As we can observe, the initial Sparc offering showed up when 4 pending bids were already registered, leading to an initial ratio jump to 4. Two subsequent bids for Sparc then entered the market to push its ratio up to 6. Nevertheless, the following Sparc offers brought it down to the expected ratio (1) from the probability distributions in Figure 4.3. In this simulation, while the NEC processor architecture also experienced initial oscillation, the x86 architecture "suffered" less, logically, since it is the one architecture with the most bids and offers.

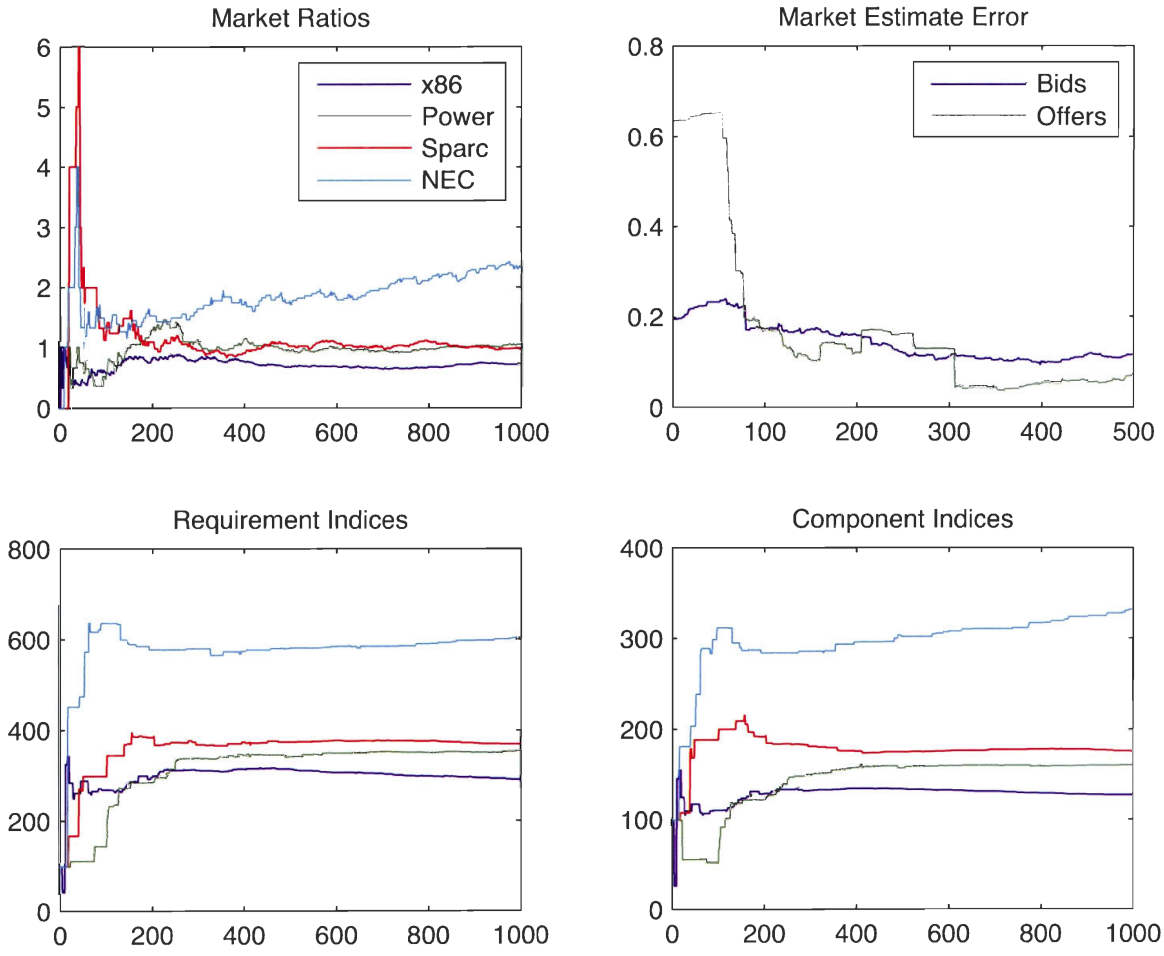

Figure 4.19: *CPU types ratios, indices and market estimate error for initial 1000 events*

While initial offer estimates experienced a rather high relative error due to initialization of both requirements and components indices to 100 *gc*, we can see that this error quickly drops to about 10%, an error range very acceptable for any such market estimator.

Looking at Figure 4.20, we can observe the various memory size indices in their transient state, then quickly adjusting to a market value reflecting their ratios. Although it may appear counter-intuitive the 32 and 64 GB of memory per processor are trading at a lower price than their smaller counterpart, it simply reflects the market's demand and offer. While the offer distribution is centered on 8GB (Figure 4.3), the demand distribution is centered on 4GB and leads to the greatest demand over offer ratio. This does not mean, however, that providers don't get any benefit from offering 32 or 64GB of memory, but rather that on an overall market scheme, the biggest value they get for their memory component comes in smaller chunks and the bigger allocations do bring incremental additional value, although maybe not as much as they would wish. As these 2 resource levels are not frequently expressed neither in bids or offers, we can see their indices staying at the initialization value for quite a while.

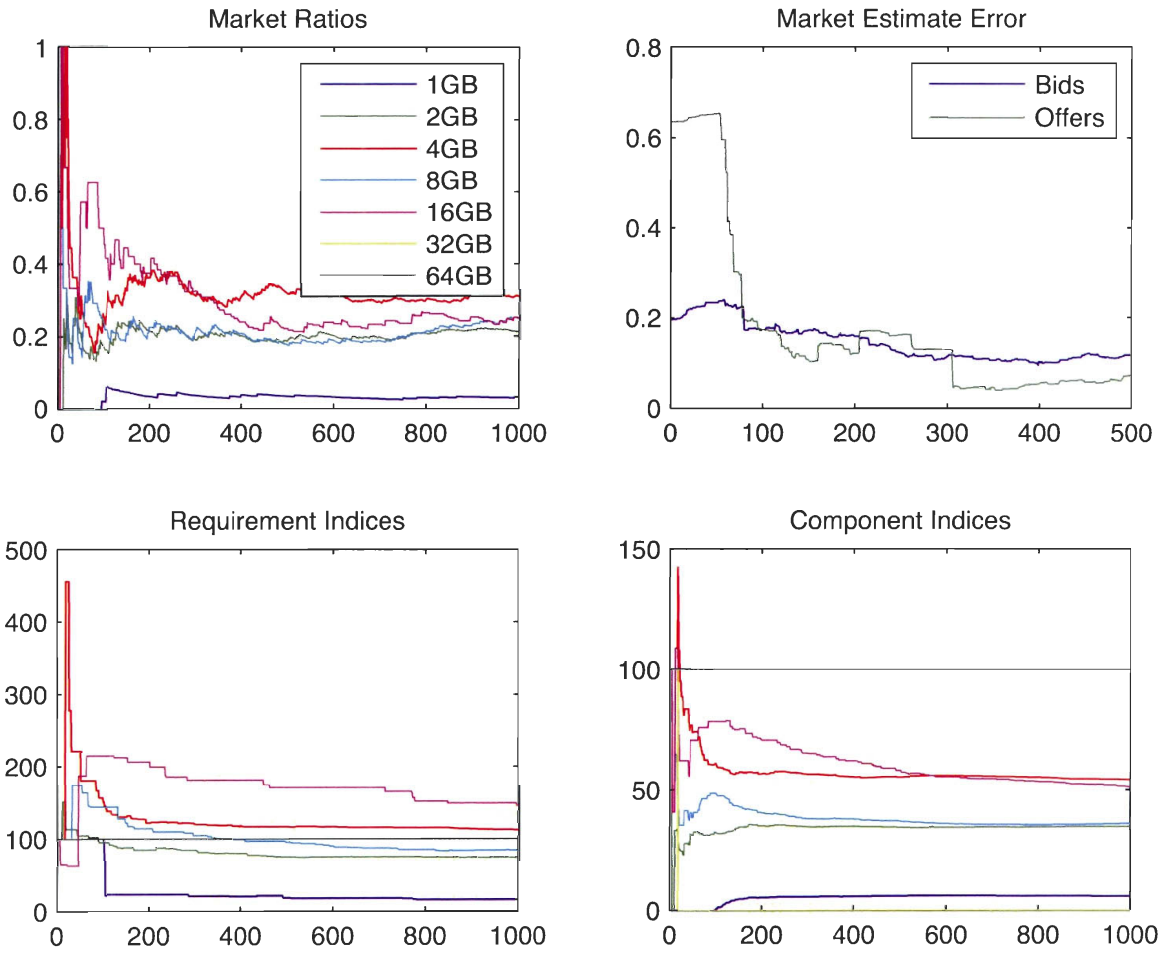

Figure 4.20: *Memory sizes initial market data*

Compared to memory sizes, network bandwidth resources exhibits the inverse distribution (Figure 4.3), were the crossover leads to an over-demand for higher-end networking infrastructure. Hence, Figure 4.21 displays networking ratios and indices were the biggest bandwidth required or offered, the higher the price goes. Over the initial 1000 events, no trade occurred for the 2000 MB/s networking resource levels. This is why its requirement and component indices stayed at the initialization value of 100 *gc*. As soon as it would get traded, we could expect their ratio and indices to be quite high, following the very high demand for them.

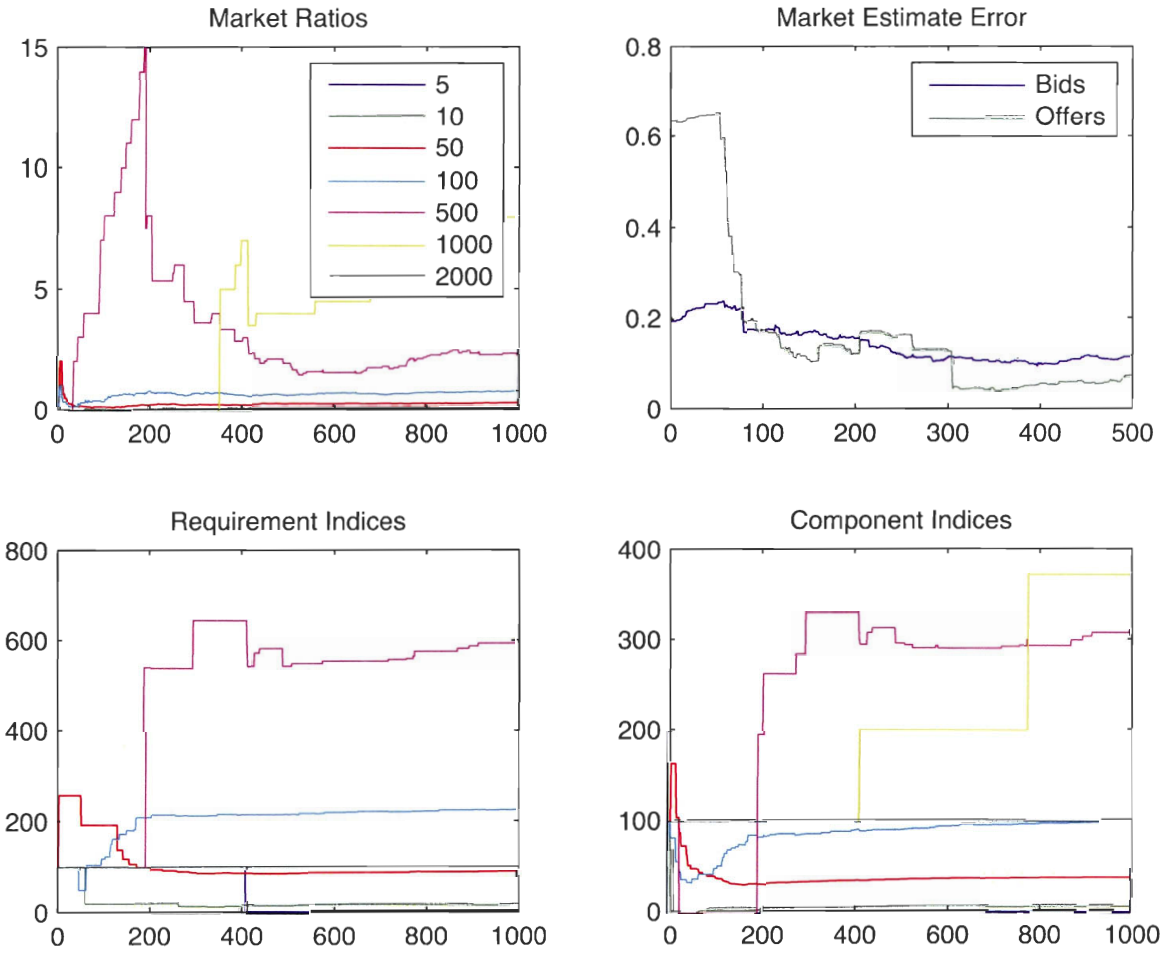

Figure 4.21: *Network bandwidth resources initial 1000 events*

Looking at processor architectures, memory sizes and network bandwidth graphs, we can see that for this totally random simulation run, most ratios and indices were quite settled to their steady state after 200 events, or only about 100 bids and 100 offers, a major argument supporting the adoption of this economic model for HPC resources trading. Appendix A includes the 5 other resource types graphs.

4.5 Economy Bootstrapping Volatility

As we have just seen in Figure 4.19, bootstrapping an economy inevitably leads to a quite unstable transient mode. For instance, steady state pricing index levels are largely influenced by the initial transaction closings. Running the simulation framework for another 1000 events can lead to somewhat different pricing figures, as shown in 4.22.

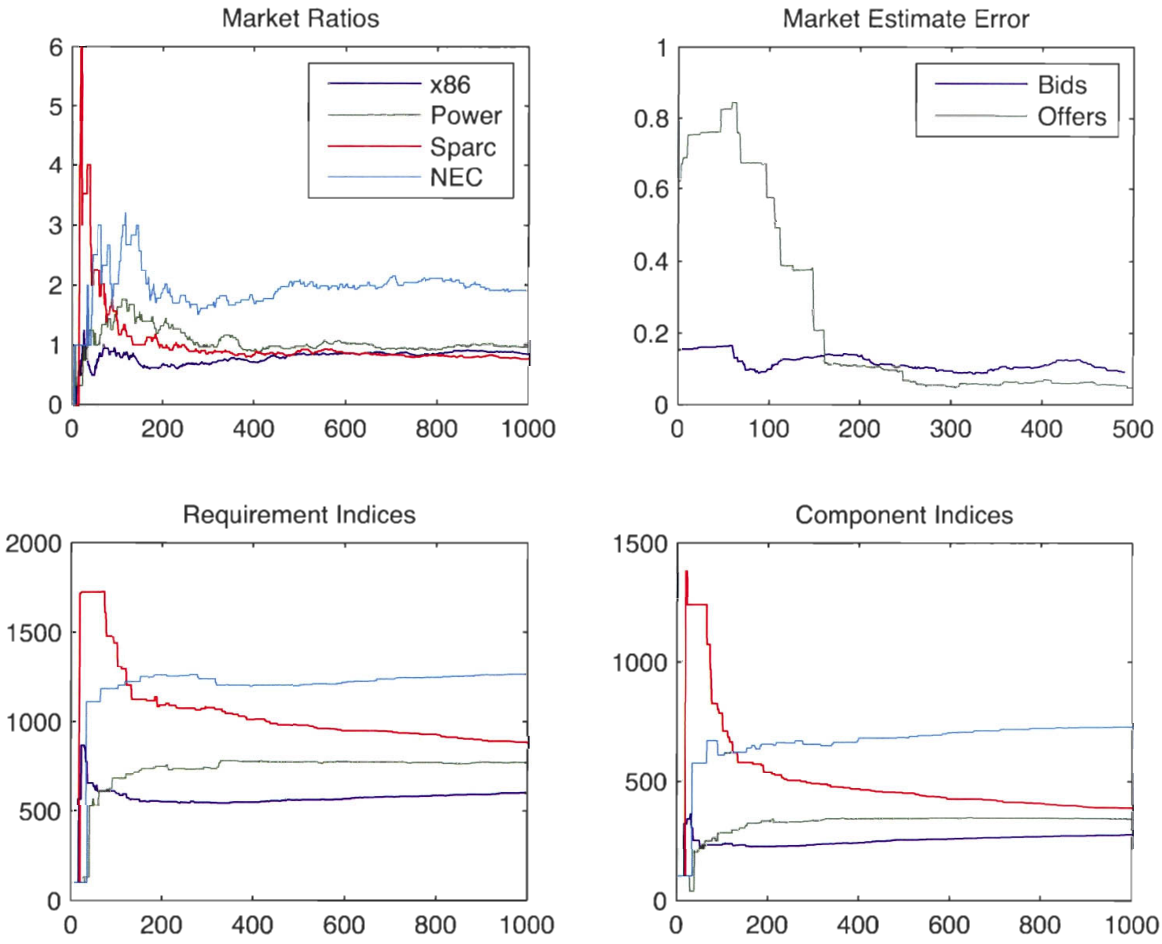

Figure 4.22: *CPU types simulation results for another 1000 events run*

Although both requirement and component indices converged to similar relative price points, their absolute value is quite different, the NEC processor requirement now trading around 1200gc after 1000 events while it traded for about 600 gc in the previous run. While that difference could be perturbing at first sight, it is really no big deal, as it only establishes the value of a Grid Credit on the market, and not the value of HPC resources themselves. As a comparison, saying the Big Mac Index [26] runs at 280 yen in Japan while it is at 3.41$ in the United States gives little information; bringing everybody back to a common currency makes more sense: Japan Big Macs selling for 2.66 US dollars.

As Grid Credits are an abstract currency, the initial volatility and steady state index settling points should not be feared. As most initial trades will be concluded by users and providers understanding the system's economics, this will largely limit volatility in comparison to random events generated by a computer. Hence, the value these initial traders will see in the Grid Credits will establish the long term pricing of HPC resources. In the end, it will only influence the potential exchange rate between a real world currency and Grid Credits.

On Figure 4.22, it is otherwise interesting to observe indices following the more unstable initial ratios, as Sparc traded high following a rocketing initial public offering (IPO) to eventually slowly decline to its expected relative price point. Consequently, in order to limit initial market instability, we choose to initialize component indices at 50 *gc* rather than 100 *gc*. Figure 4.23 plots the results. As ratios and indices converge to their expected values, the most interesting thing to observe is that we got rid of the rather high initial offers market estimate relative error (∼65% and ∼80%) we had in the preceding runs (charts 4.19 and 4.22), limiting it to a much lower 25% figure.

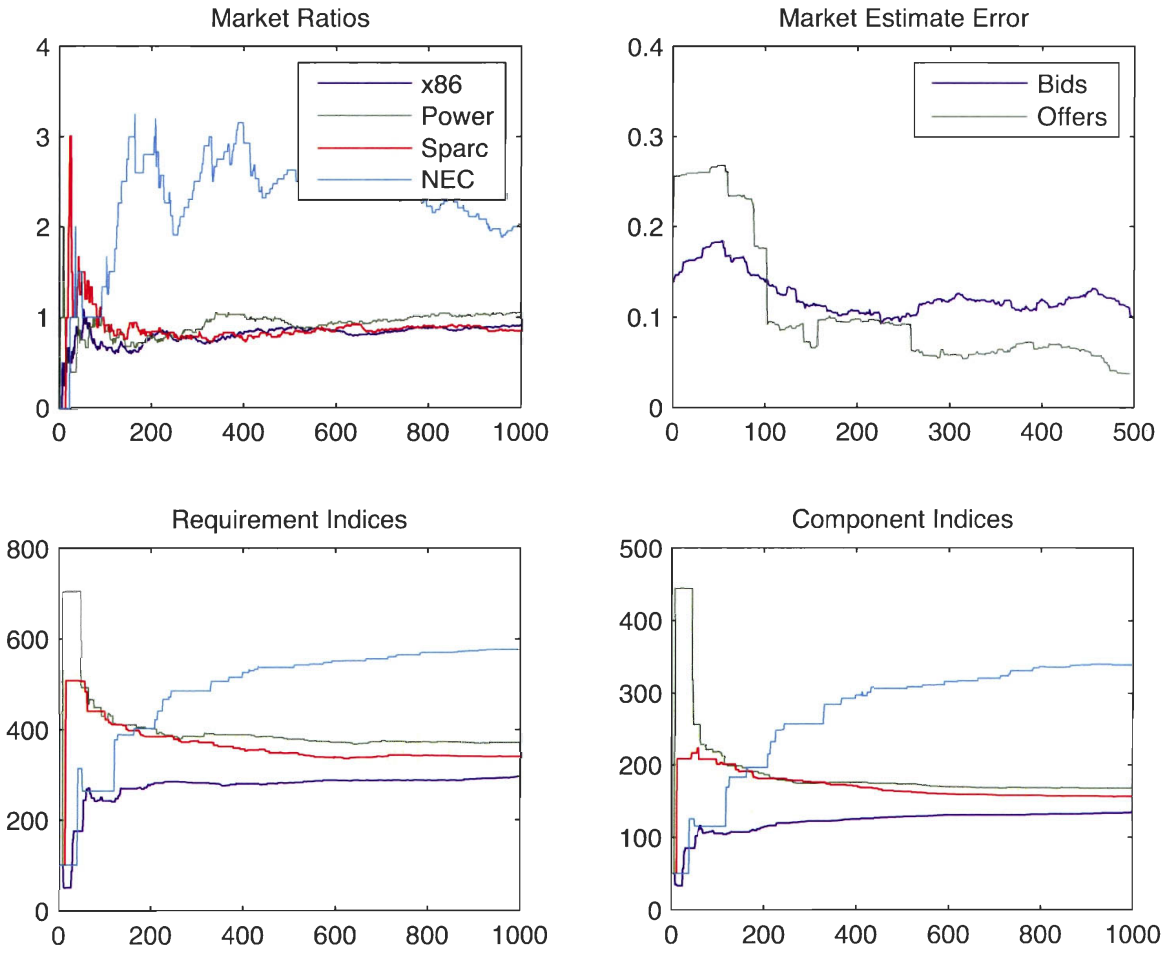

Figure 4.23: *Bootstrapping the economy with 50 gc default component indices*

4.6 Traders Confidence in Market Indices

In this section, we will modify both buyers and sellers tolerance (`bidtol` and `offertol`) over calculated market estimates. On the bid side, it is used to get the trading signal when a bid enters the market and finds a matching offer (if there are multiple matching offers, then the cheapest one is the one retained): if `bidprice*bidtol >= offerprice`, a transaction is concluded at `offerprice`. Following an offer creation event, when that offer enters the market and gets compared to the highest bid, if `offerprice*offertol <= bidprice`, the deal is closed at `bidprice`.

We thus lower the tolerances from the very high 500% `bidtol` and 20% `offertol` used in the previous section to 125% and 80%, respectively. Figure 4.24 shows the results. We can see straightforwardly that volatility is much lower, an observation that makes plenty of sense as lower tolerances lowers the system's instability.

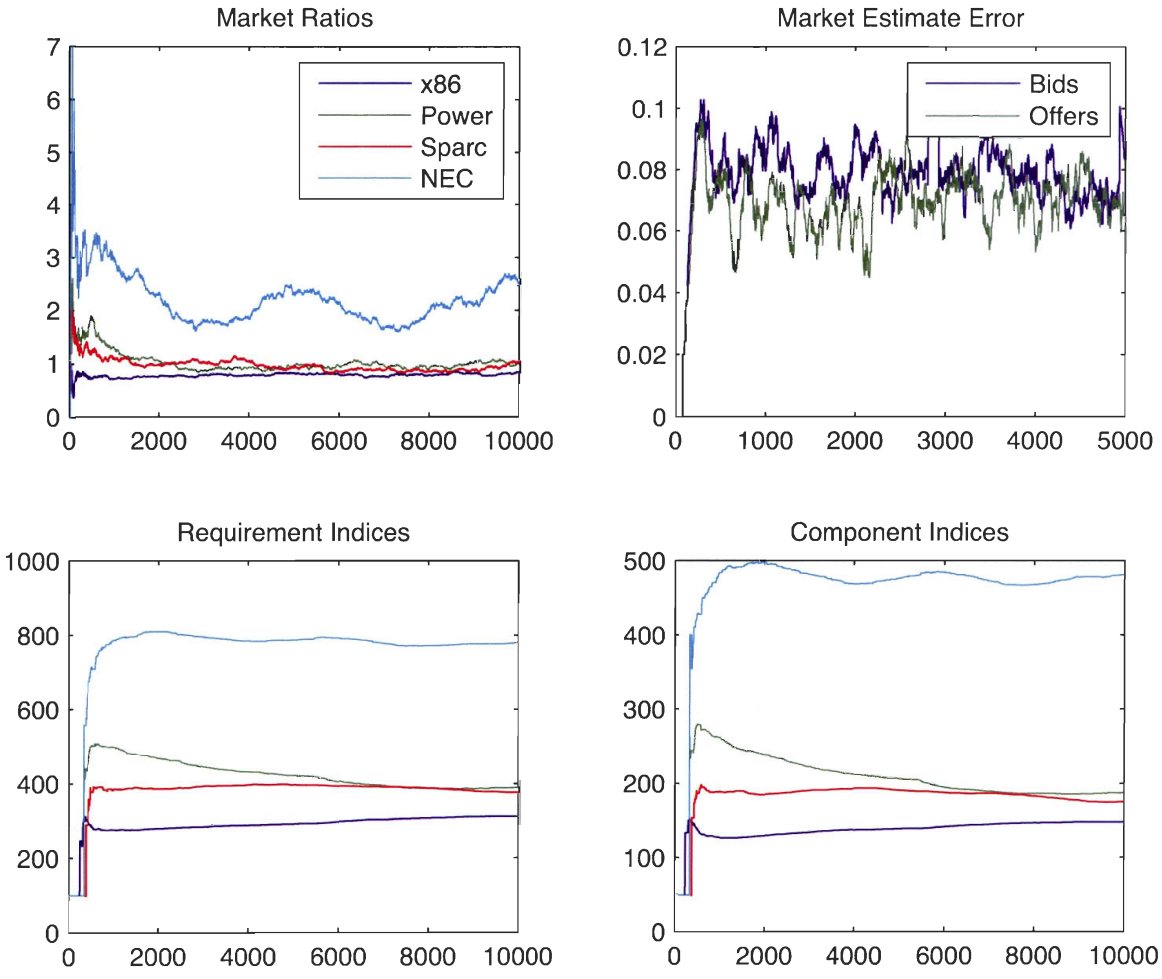

Figure 4.24: *Lowering Bid tolerance to 125% and offer tolerance to 80%*

Zooming on the first 1000 events as shown in Figure 4.25, we can see that it took a while for a trade to occur, about 200 events. This is logical since tolerances are low, thus matching bids and offers with default indices had to be more tricky. However, we can notice that as soon as indices started adjusting, the market constrained volatility much more effectively and it continued over the entire 10 000 events as the average bid estimate error dropped from 11.6% to 7.9% and the mean offer estimate error went from 11.1% to 6.9%.

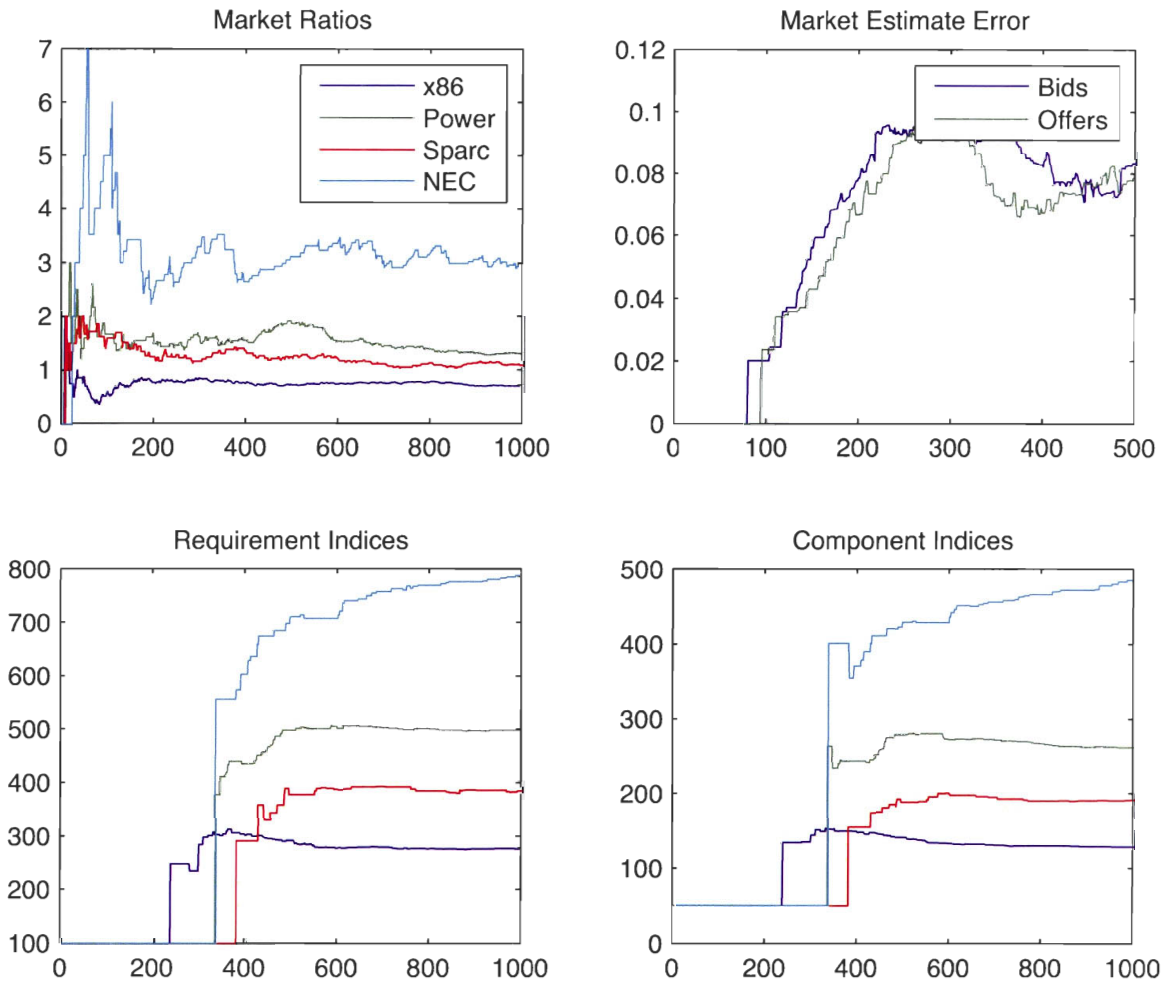

Figure 4.25: *Initial 1000 events with lowered tolerances*

In the end, these improvements would mean nothing if significantly fewer transactions occurred. Looking at Figure 4.26 a), we can see 3828 transactions took place. That's 77% of market orders that got fulfilled, a tiny 5% drop from the previous 82% figure while tolerances got dropped significantly (from 500% to 125%). Pushing even further, we lower tolerances to 110% for bids and 90.9% for offers. Figure 4.26 b) shows the 2878 transaction closing prices, a fulfillment ratio of 58%, not bad at all considering we are dealing with a bootstrapping market and hence unstabilized estimators.

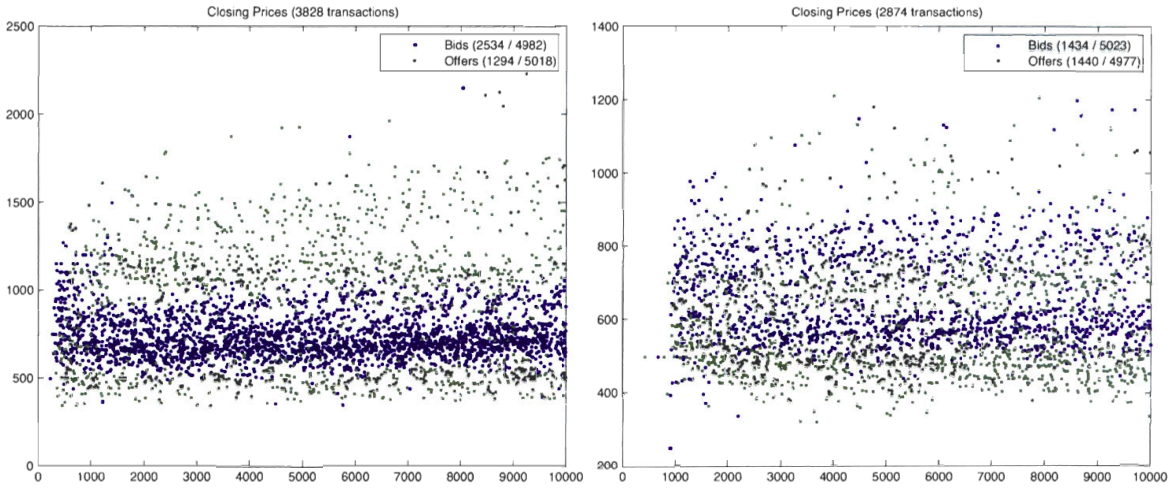

Figure 4.26: *Transactions with lowered tolerances a) 25% (left), b) 10% (right)*

The one drawback inherent to lowered tolerances is the delay before trading starts. For the 125% tolerances, that delay was about 200 events, a still reasonable figure. However, when lowered to 110%, tolerances delayed trading to the 1000th event. For obvious reasons, in a real market, traders will have higher tolerances while bootstrapping the market, and then, as the estimators stabilize, their tolerances should lower. Figure 4.27 shows a simulation run where tolerances are initialized to 200% and 50%, then lowered to 125% and 80% at the 500th event, and subsequently lowered again to 110% and 90.9% at the 2000th event mark.

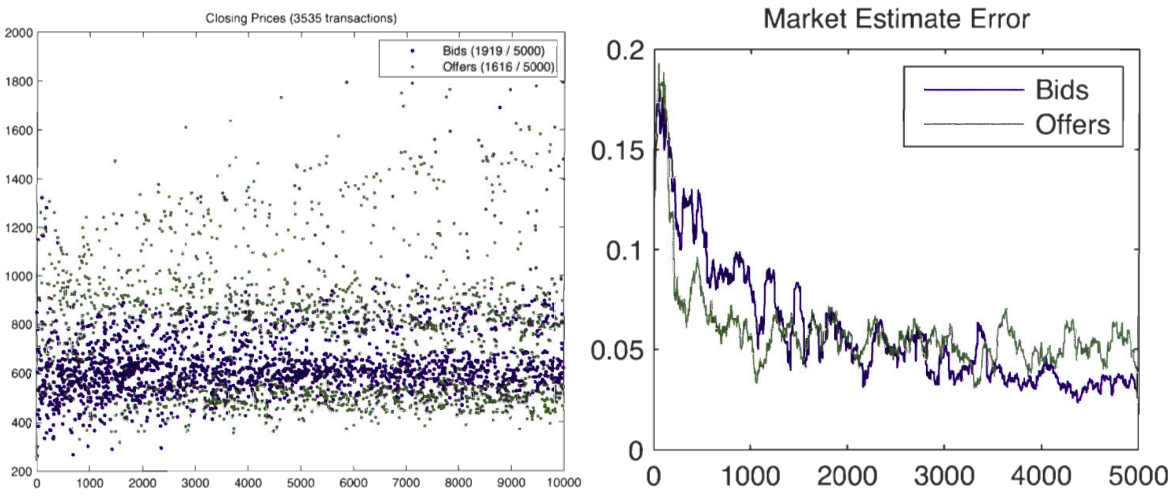

Figure 4.27: *Transactions with adaptative tolerances (2.0, 1.25, 1.10)*

As we can see, trading starts as soon as the 4th event, 3535 deals are concluded, a 71% completion ratio, and market estimators error drops to a 5% range as soon as we hit the 2000th mark. In appendix B are other graphs for this adaptative tolerances simulation run.

4.7 Market Entropy Analysis

Up to here, we have used a 2000 event memory buffer for ratio and index calculations. In practice, it means that resource ratios are calculated from the last 2000 events (bids and offers) posted on the market. As for indices, their calculation was made by averaging resources indices calculated from the last 2000 transaction closing prices.

The system's memory, or more precisely in the case considered, the market's momentum, is a very important variable to study. Following signal analysis general principles, a memory buffer too small leads to an unstable system while one too extensive hampers the system's ability to adapt. Hence, in this section, we will study the market's momentum from the system bootstrapping stabilization perspective. The next section will follow studying adaptation to changing market conditions.

Figures 4.28, 4.29, 4.30 and 4.31 depict the market response to bids and offers generated by the default probability distributions (Figure 4.3) for 100, 500, 1000 and 2000 events/deals memory, respectively. Looking at Figure 4.28, we can observe instability all over the ratios, requirement and component indices. Although index means seem to converge on the expected pricing, the oscillation around the expected price point precludes the use of such a low system memory. Figure 4.29, with a 500 events/transactions memory, is already a major improvement over 4.28, although we can still witness some oscillation, however presenting a lower frequency, but still too much to base a trading system confidence on it. On 4.30, oscillations are not a problem anymore and indices stabilized quickly to their steady-state pricing range. Last but not least, Figure 4.31 plots a market simulation with the default system memory of 2000 events/transactions. On most simulation runs, a 2000 events/closings memory looks perfect. However, in this particular case, the NEC processor sees an unplanned ratio jump around the 5000th event (following truly random events), and indices only start to account for it about 2000 events later, and without the system response we could expect.

Looking at these 4 graphs, we can observe that ratios tend to be more unstable than indices. Since ratios are computed on events, with a higher inter-arrival rate, their base frequency is simply higher than indices, computed on completed transactions, with a maximum frequency that would be half the ratio's, if every single bid and offer would get matched. Hence, looking at the various options, for this market instantiation, with these resource types and levels of service, following their respective probability distributions, a 2000 events memory seems to be the right compromise for ratios. As for indices memory window, it's hard to conclude, as most of their instability came from their dependency on their respective ratios. The next section then studies system adaptation to changing market conditions and how indices adapt to it.

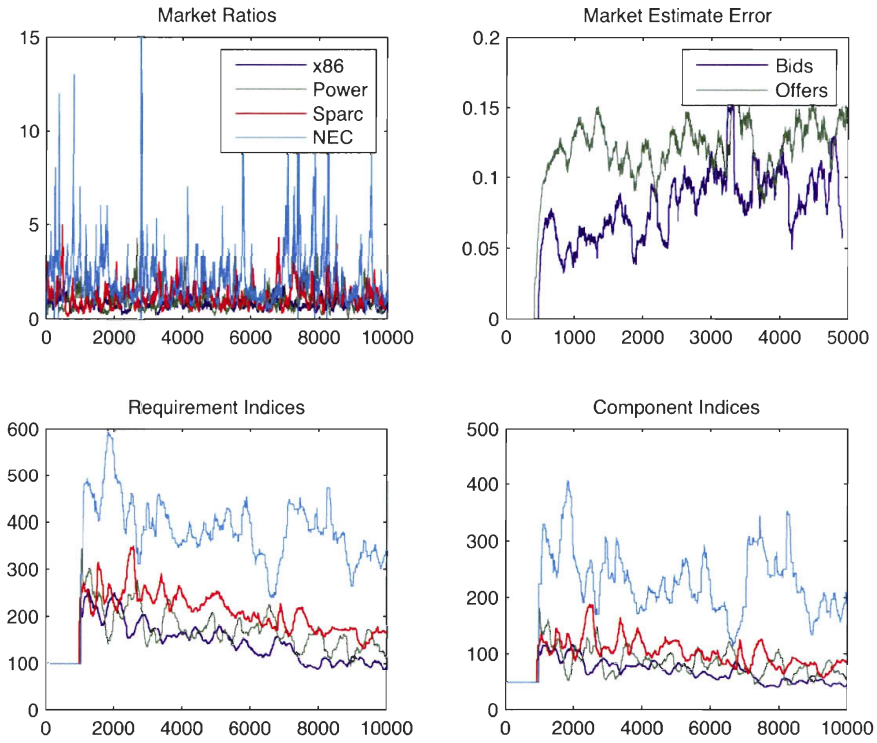

Figure 4.28: *Market entropy with 100 events/deals memory*

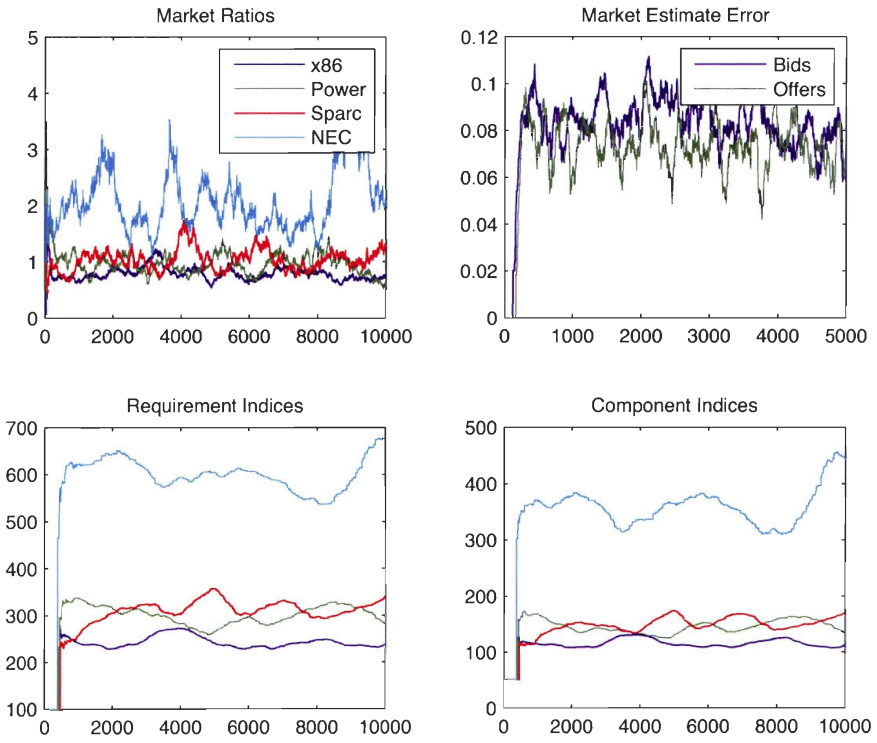

Figure 4.29: *Market entropy with 500 events/deals memory*

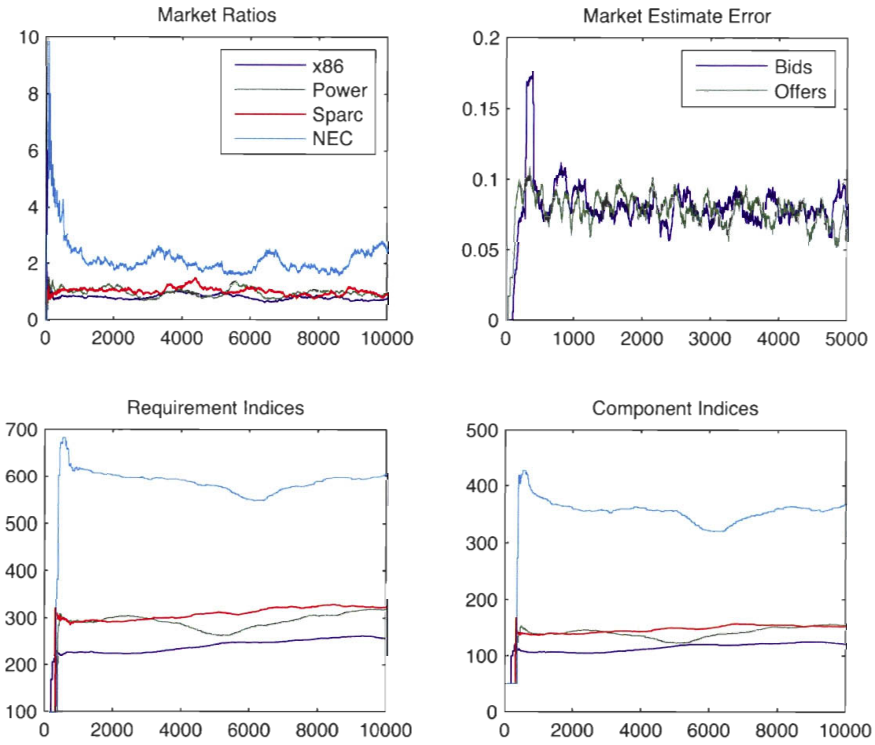

Figure 4.30: *Market entropy with 1000 events/deals memory*

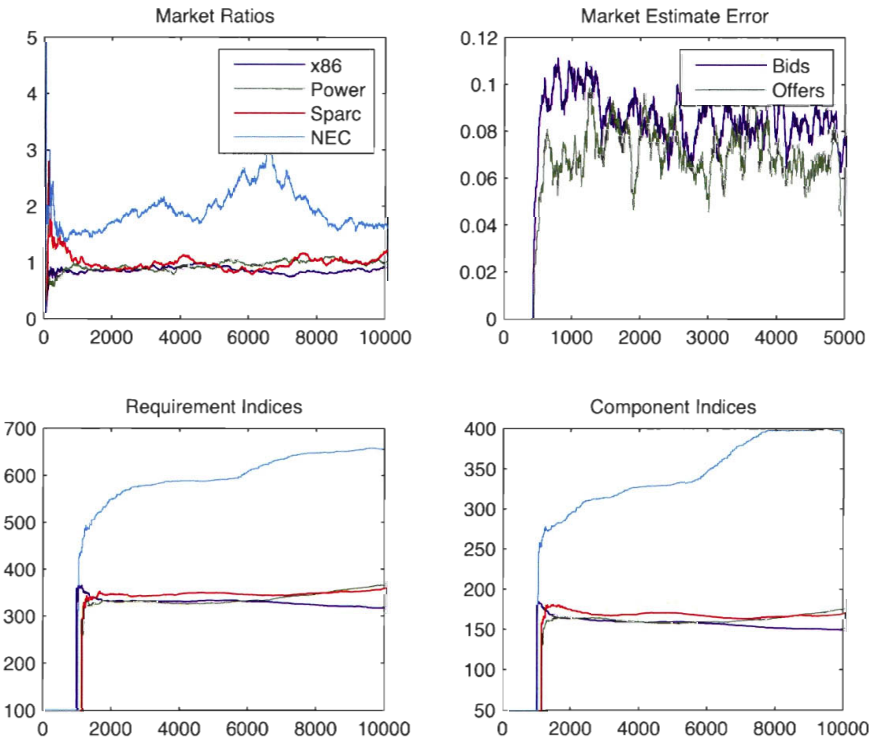

Figure 4.31: *Market entropy with 2000 events/deals memory*

4.8 Market Adaptation to Changing Demand and Offer

By default, bids and offers for processor architectures are distributed according to Figure 4.32 a). In order to simulate how our market representation adapts to change, let's suppose a new climate prediction code, one of the prime uses for NEC based systems, gets released on the Power architecture. As this new code presents major benefits over the previous NEC version, it pushes our probability distributions such that NEC falls in an over-supply situation (5%/10%) and Power, from a balanced demand, jumps to a 35/20 demand over offer ratio, as seen on Figure 4.32 b).

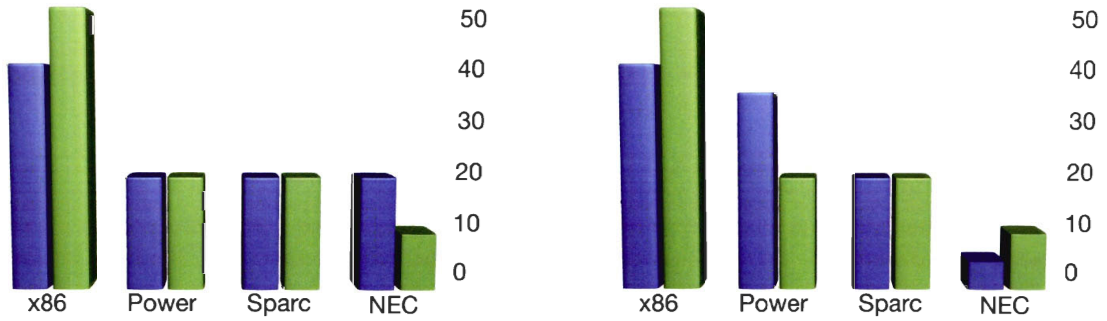

Figure 4.32: *Processor types probability distributions: a) default, b) new market conditions*

We thus bootstrap the simulation with the default probability distribution in Figure 4.32 a) until the 2000th event. At this very moment, we switch probability distributions to the ones in 4.32 b). Tolerances are initialized at 125% (bids) and 80% (offers) and are kept unchanged in order to focus on the system's adaptation analysis.

As we can see on Figure 4.33, ratios quickly start adapting around the 2000th event. After 4000 events with the new probability distribution, they have stabilized to their new expected values. However, indices tend to have a longer adaptation period. Hence they reach anticipated market values by the 10 000th event and NEC's and Power's indices cross-over occurs around the 7000th event, while it happened around the 3000th for their respective ratios. This is largely due to the fact that indices are calculated by retro-propagating a transaction closing price to every single requirement and component concerned in the deal in proportion to their market ratios. Hence, before indices start adapting, ratios must have done so. Since indices are averaged over the last 2000 transactions (and not events, for obvious reasons), this adds up to the adaptation period.

In order to reduce our indices response lag, we re-use the first 2000 events and recompute the simulation, this time with a 1000 transactions memory. Comparing to Figure 4.33 and remembering probability distributions have been switched after 2000 events, Figure 4.34

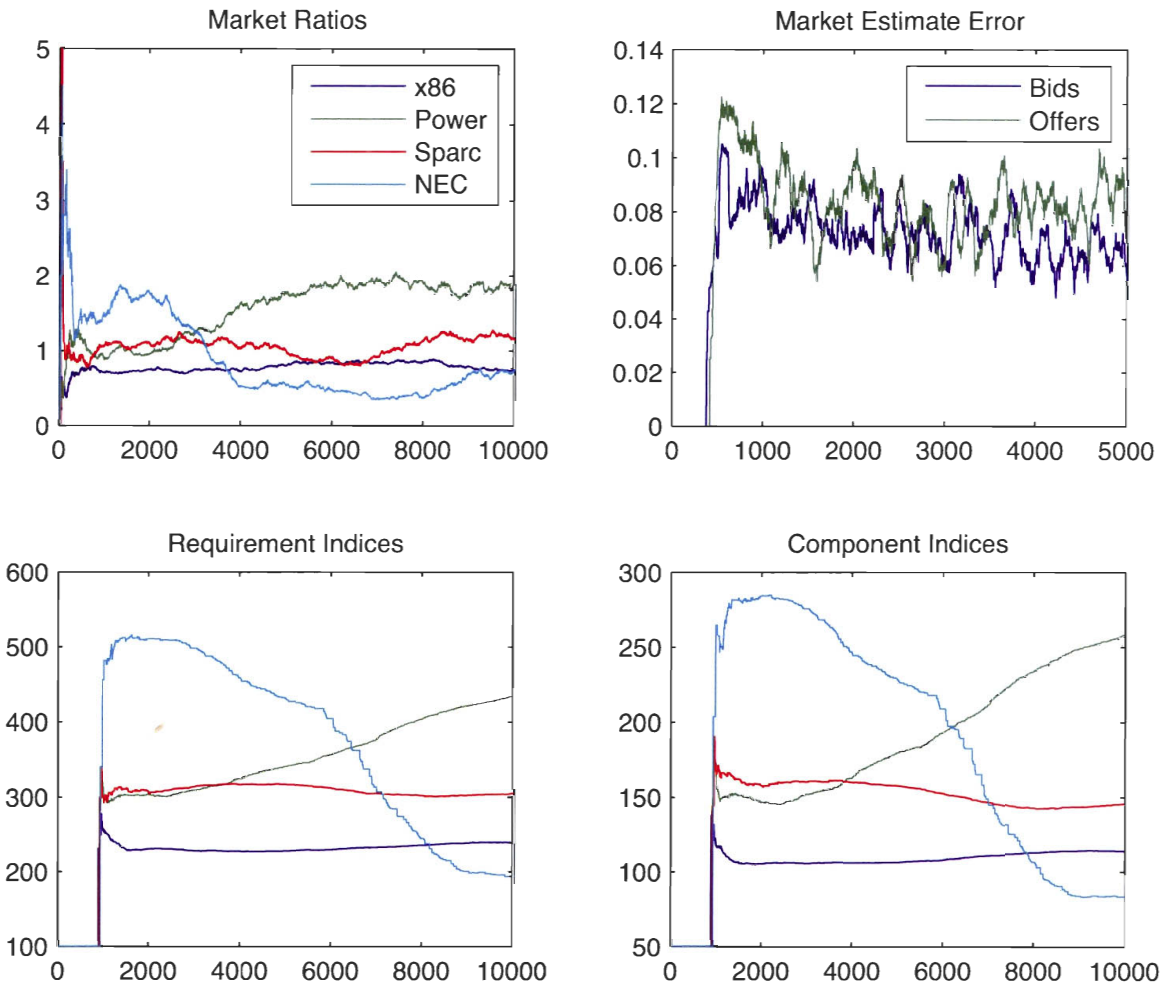

Figure 4.33: *Ratios and indices adapting to changing market conditions*

exhibits a more responsive system behavior, where indices cross-over around the 5000th event and have mostly reached their new equilibrium point around the 8000th, or 6000 events after the market conditions changed, while it took 8000 events before.

Pushing even further, we lower the transactional memory to the 500 previous deals, as shown in Figure 4.35. As we can see, the system is somewhat unstable because of a too small memory buffer. Hence, under both market bootstrapping and permanent state conditions, a market entropy consisting of a 2000 events and 1000 transactions for ratios and indices calculation, respectively, seems to be the best compromise and should thus be kept as default parameters. As a complement, other resources ratios and indices for a 1000 events and 2000 transactions memory are added to appendix C.

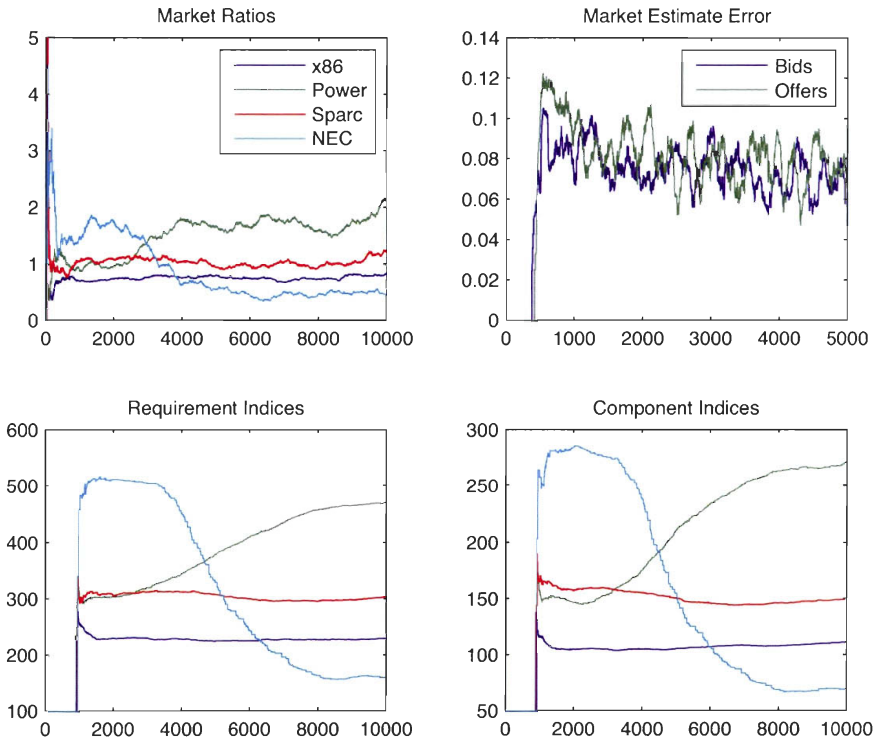

Figure 4.34: *Market adaptation to changing demand and offer, with a 1000 events memory*

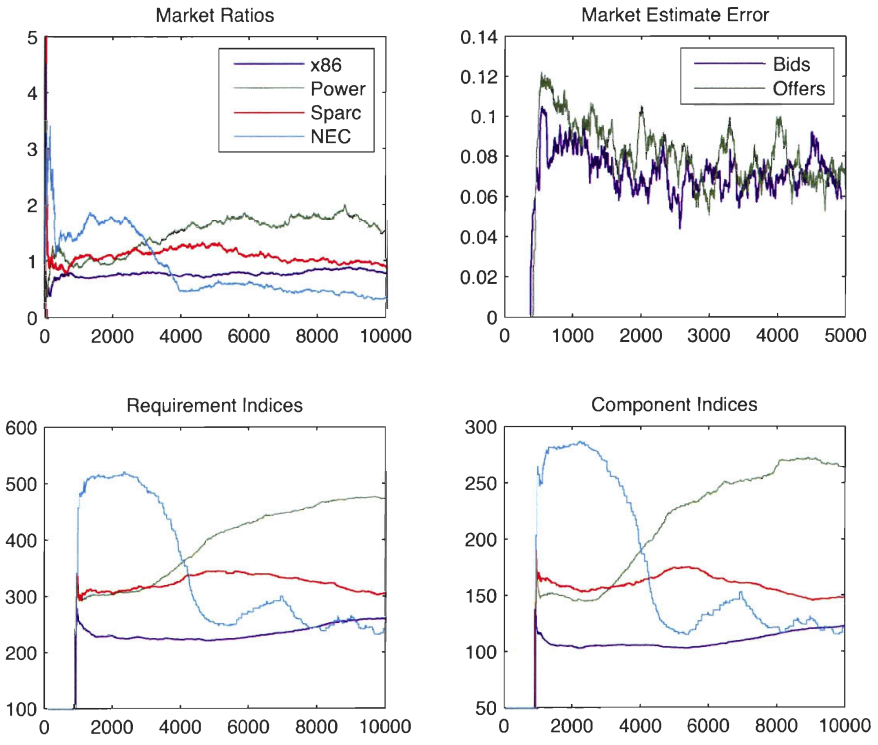

Figure 4.35: *Index oscillation because of a too small system's memory (500 events)*

4.9 Bullish vs Bearish Market

This section focuses on the variations in demand and offer on the macroeconomic scale. Hence, it is no longer a matter of changing probability distributions within resource sets but rather of dealing with global market tendencies. To do so, we vary the different simulation parameters in order to simulate a bullish or bearish market [1].

Pre-computing an initial 2000 events set with default probability distributions, we start by modifying the 0.5 ratio between bids and offers to 0.75 in order to generate a 3 for 1 over-demand and we leave all others parameters to their default values. Figure 4.36 presents the results.

The other way around, we reuse the initial 2000 events and repeat the simulation, this time with a 0.25 ratio, generating a 3 for 1 oversupply. Figure 4.37 plots the system's response.

In both cases, as ratios take significant jumps, the various indices stability is quite astonishing at first sight. In 4.36, with ratios going up abruptly, we would expect their respective indices to follow and even go through the roof, as there is 3 buyers for every seller. Inversely, in 4.37, shouldn't indices plummet to the floor?

Looking at transaction closing prices confirms that preoccupying behavior observed in ratios, with almost no inflation in 4.36 and insignificant deflation in 4.37.

This behavior, while counter-intuitive, is in fact absolutely correct. Simply put, it is the demonstration through simulation of a system following John Nash's competitive equilibrium theory [116]. By itself, this is a major result supporting this thesis proposed economic model as a valid and stable system for HPC resources trading.

Pushing the analysis further, indices have remained mostly stable in both cases because there has been no change in either short or long strategies. As their tolerances remained the same and their market entry pricing stayed at the proposed market estimate, there was just nothing pushing the market either way. Looking at transactions in Figure 4.36, we can see that from the 2000th event, when ratios changed, it became a sellers market, with mostly every offer being posted finding a bid to trade with. Even though the market matched newly generated offers with the highest reward bid (reason for the slight inflation), as sellers strategies remained the same, while most of the time they got into a closing within a shorter timeframe, they could have waited longer for a more profitable deal.

[1] Recall a *bullish* market is a market with a strong demand, favoring longs (sellers), with prices going up while a *bearish* market suffers oversupply, pushing prices down, and thus in favor of shorts (buyers).

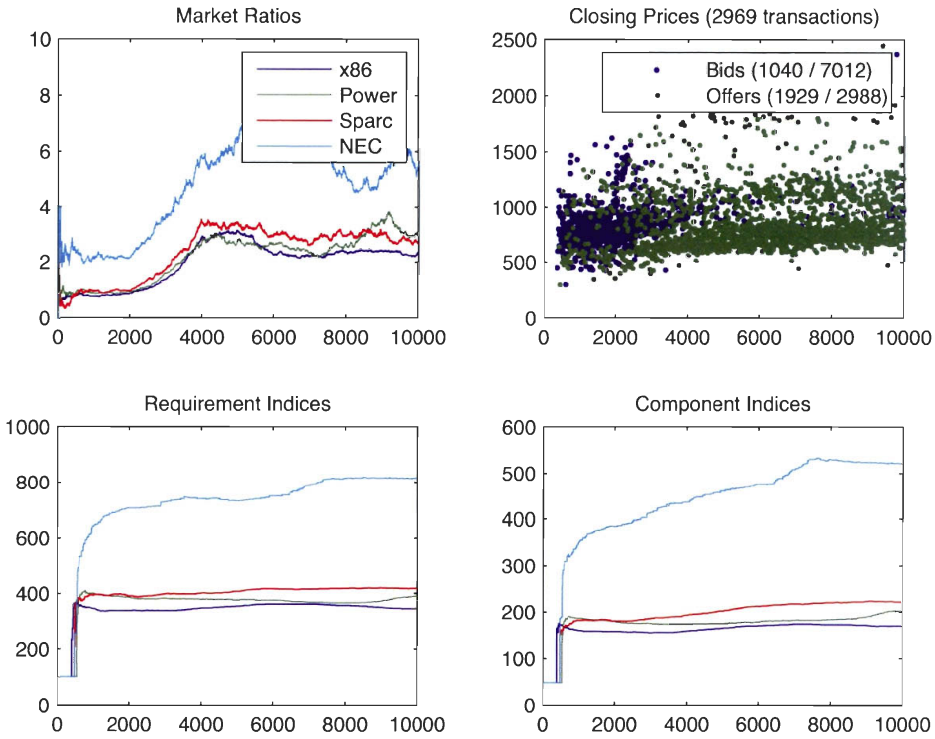

Figure 4.36: *Market adaptation following a 3 for 1 jump in demand*

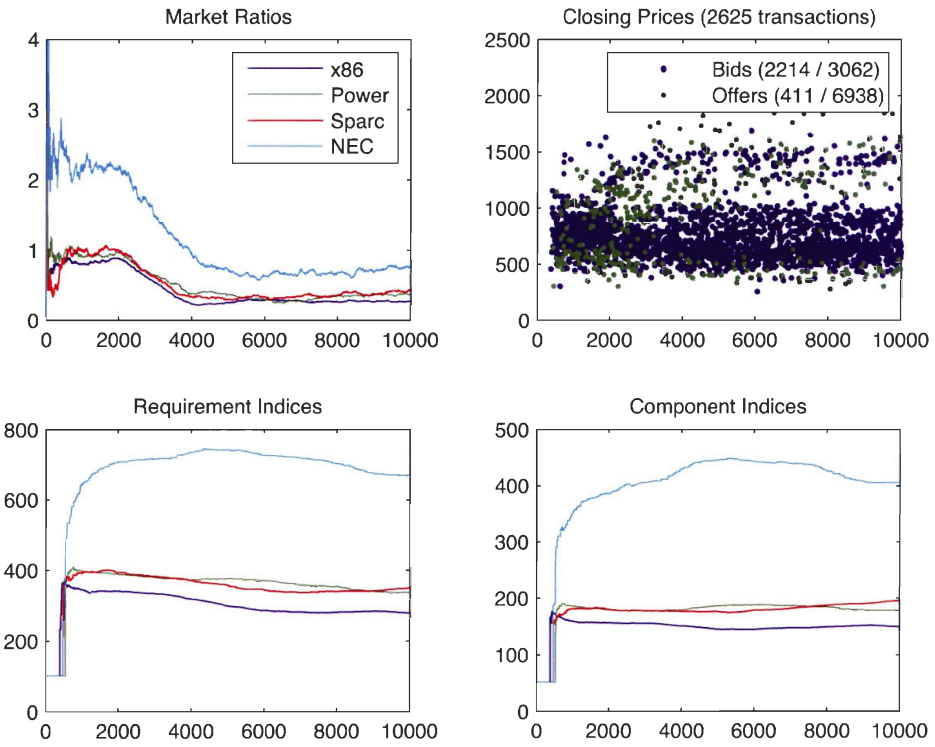

Figure 4.37: *Market adaptation following a 3 for 1 jump in supply*

Similarly, following the 0.25 ratio change, the market became a buyers market in Figure 4.37. As every entering bid quickly found an offer to trade with, and since buyers did not lower their tolerances, although the market matchmaking system got them the cheapest offer available (reason for the minimal deflation), they just did not put enough discounting pressure on the sellers.

While these conclusions hold true for most indices, NEC requirement and component indices look contradictory. The rise observed in 4.36 is mostly due to NEC ratio oscillation, caused by a lack of system memory for that type of resource, as it is the one resource experiencing the largest variation while presenting the fewer events. Trying to resolve that issue, an approach with adaptative ratios is proposed in section 5.1.

Pushing our market equilibrium analysis further, in order for the system to reflect over-demand or oversupply in the closing prices, we should expect a balanced mix of closing bids and closing offers in the transaction graph (an almost equal mix of blue and green dots).

To do so, we start by limiting sellers tolerance over the proposed market estimate from 0.8 to 1.0, meaning sellers can now trade at their market estimate or higher, but not any lower. Figure 4.38 plots the results. Likewise, we limit buyers tolerance from 1.25 to 1.0 in the over-demand case, as shown on Figure 4.39.

Already, we can start witnessing inflation and deflation. In Figure 4.38, processor architectures requirement indices grew up 18% from the 2000th event to the 10000th. While transaction closing prices do not appear to have gained that much on when looking at the graph, they went up from an average of 773 gc between the 1000th event and the 2000th event to an average of 1024 gc for the last 1000 events, a non-negligible increase of 32%. In Figure 4.39, processor architectures requirement indices went down 18.8% on average and closing prices came from 773 gc to 713 gc, an 8% drop.

In order to simulate an even more realistic market where buyers an suppliers can enter a position at any price point, we add 2 parameters to our simulation function. Hence, on Figure 4.40, sellers enter the market at 125% of their estimated pricing, with no tolerance, while buyers have not changed anything in their strategy, their entry pricing remaining at the market and their tolerance staying at 1.25. From that somewhat limited increase in market order pricing on the long side, we witness some major bullish behavior, processor architectures jumping 60% on average, and transactions going from 773 gc to 1457 gc, an 89 % increase. Figure 4.41 presents the bearish equivalent, with consumer market orders entering at 80% of their estimate, without tolerance and under a 3 for 1 oversupply scenario; every other parameter remaining fixed. As anticipated, we witness a market crumble, with a 42% drop for processors and closing prices going down from 773 gc to 472 gc, or -39%.

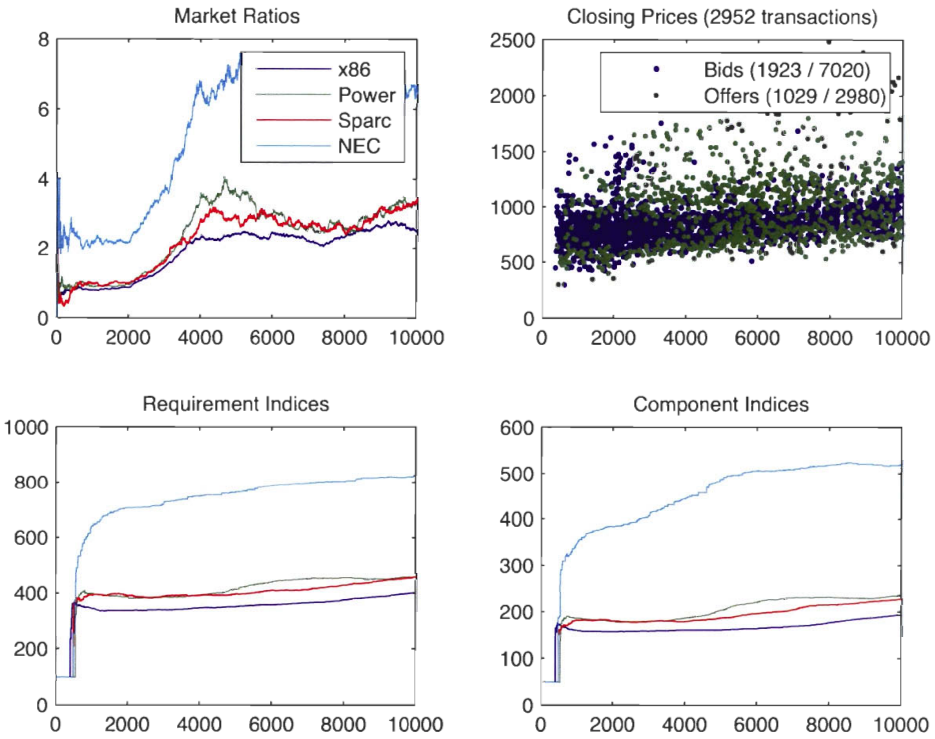

Figure 4.38: *Market adaptation following a rise in demand and lowered sellers tolerance*

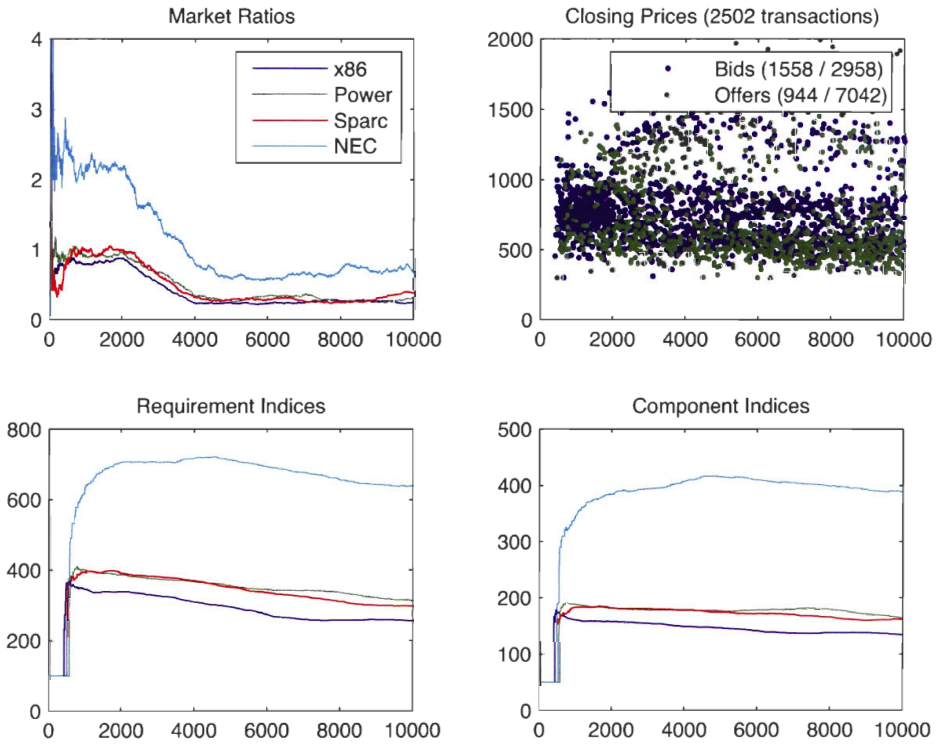

Figure 4.39: *Market adaptation following a rise in supply and lowered buyers tolerance*

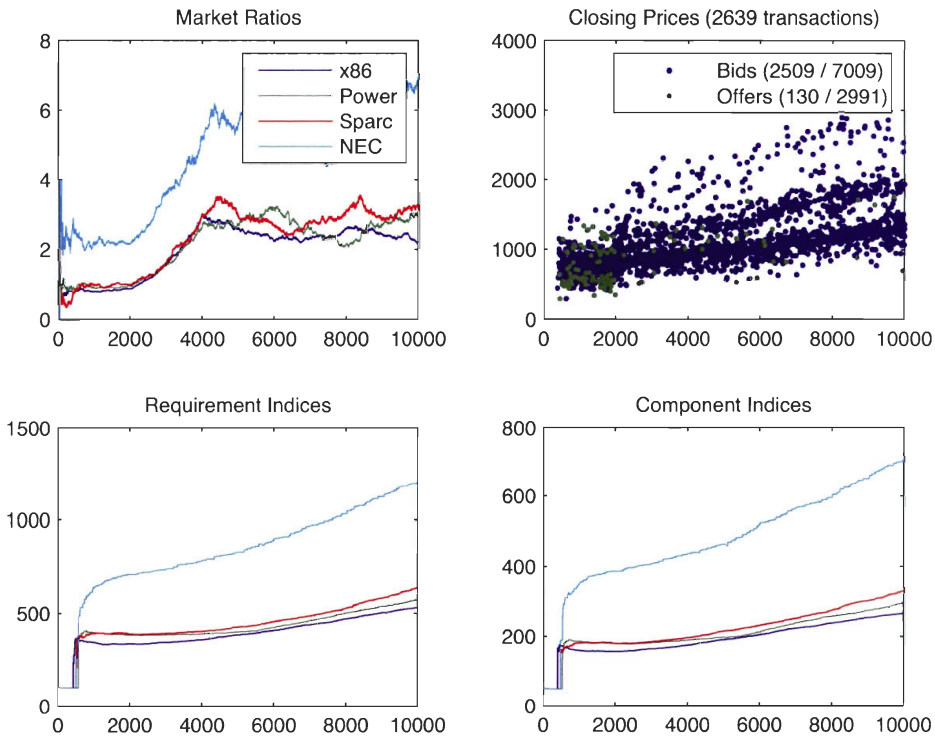

Figure 4.40: *Market adaptation following a rise in demand and higher priced offers*

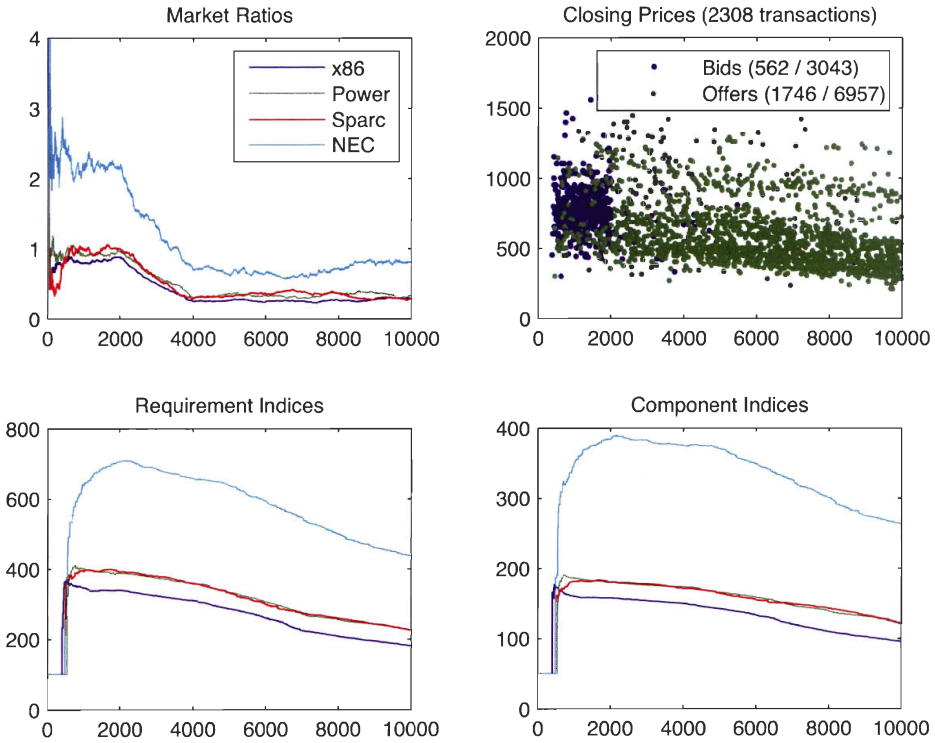

Figure 4.41: *Market adaptation following a rise in supply and lower priced bids*

Last but not least, we restart the simulation from Figure 4.40 and randomly generate another 10 000 events, but this time with a balanced demand/supply ratio, and all other parameters kept to their default values, bid and offer tolerances fixed at 1.25 and 0.8 respectively, and market orders entering "at the market".

As we can observe on Figure 4.42, ratios quickly settle down to their default values right after the 10 000 event mark and indices stabilize to the price points they were then at. This is expected as for any market, if there is no reason for indicators to change, they should just stay at their current price points.

One of the most interesting figures to analyze is the closing prices graphs. As we can see, during the bullish push, providers entered positions at 125% the market and refused to trade until a bid came up matching their price. As soon as the market re-equilibrated and providers came back to their default positioning scheme, they started trading once again when entering the market, as shown by the green dots coming back after the 10 000th event.

While many other market equilibrium simulation runs could be conducted, this section demonstrated the proposed HPC resources trading system effectiveness, stability and adaptability to the multiple market conditions that could eventually arise.

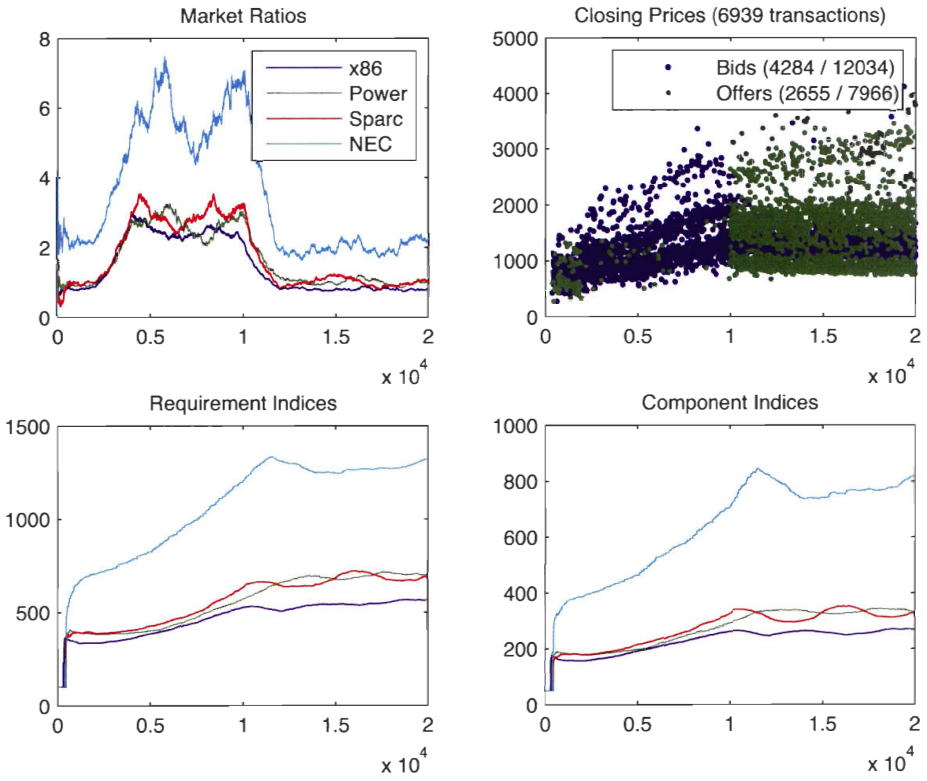

Figure 4.42: *Market settles down after a bullish period*

4.10 Introduction of New Resources

As HPC resources evolve rapidly, in fact much faster than any typical commodity, a Grid Exchange system must be able to support the addition of new resources at any point in time. As the proposed trading system is already operating in a *multi-commodity* mode, it is straightforward to add a new resource sub-type or an entirely new resource type like interconnect latency, for example.

As a demonstration, we therefore introduce a new processor architecture simply by modifying the default probability distribution (Figure 4.43 left) used in the previous sections, right after the 2000th event. In this example case, we now enable users and providers to specify the version of the Power processor required or offered. Considering the "Power" resource referred in fact to a "Power5" processor, we simulate the introduction of the "Power6" processor in the market. As anyone could expect, for any new version of a processor architecture, demand is high while offer is limited for the Power6; the new processor is thus experiencing over-demand while the Power5 is left in oversupply. Consequently. the previous Power 20% bid and 20% offer probability distribution then becomes a 10% / 15% ratio for Power5 and 10% / 5% for Power6, as shown on Figure 4.43 (right).

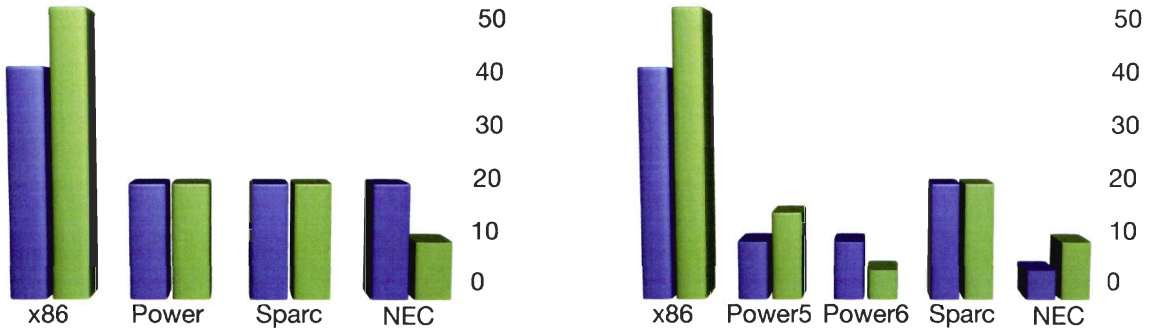

Figure 4.43: *Introduction of a new processor architecture resource (right)*

Figures 4.44 and 4.45 exhibit the system's response following the ratios modification after the 2000th event. In 4.44, the Power6 ratio, requirement index and component index are left to their default values of 0, 100 *gc*, and 50 *gc*. In 4.45, we initialize the Power6 IPO by copying the 3 market indicators from the Power resource, now labelled Power5. As we can see, it makes little difference, neither for ratio, indices nor system adaptation; an evidence supporting the fact that the market reaches his equilibrium point for each and every resource, regardless of default initialization values.

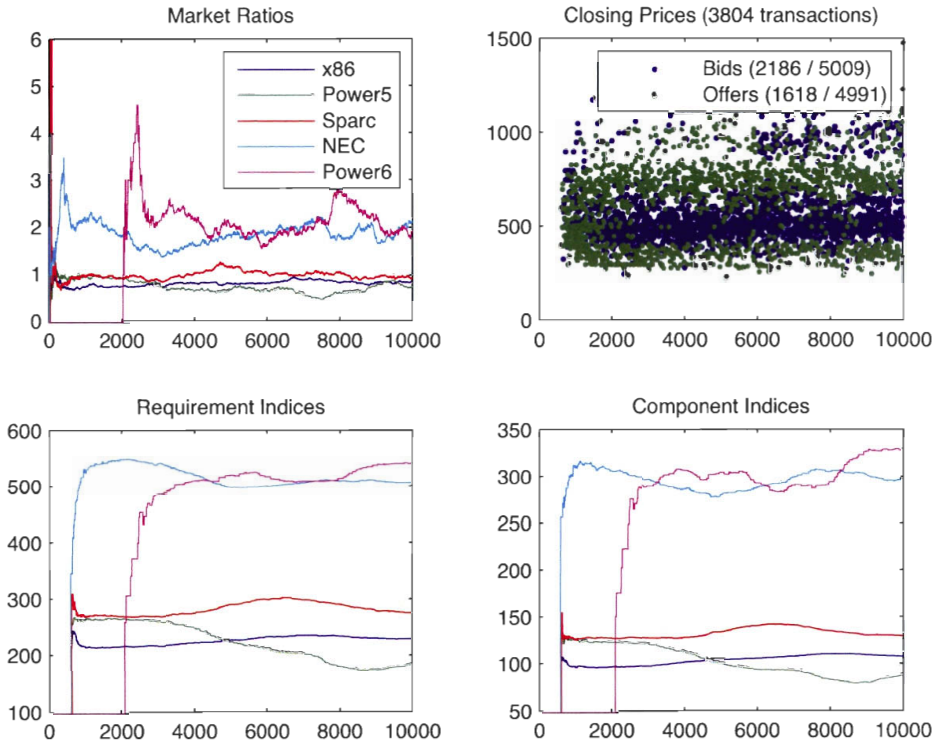

Figure 4.44: *Introduction of a new processor architecture resource*

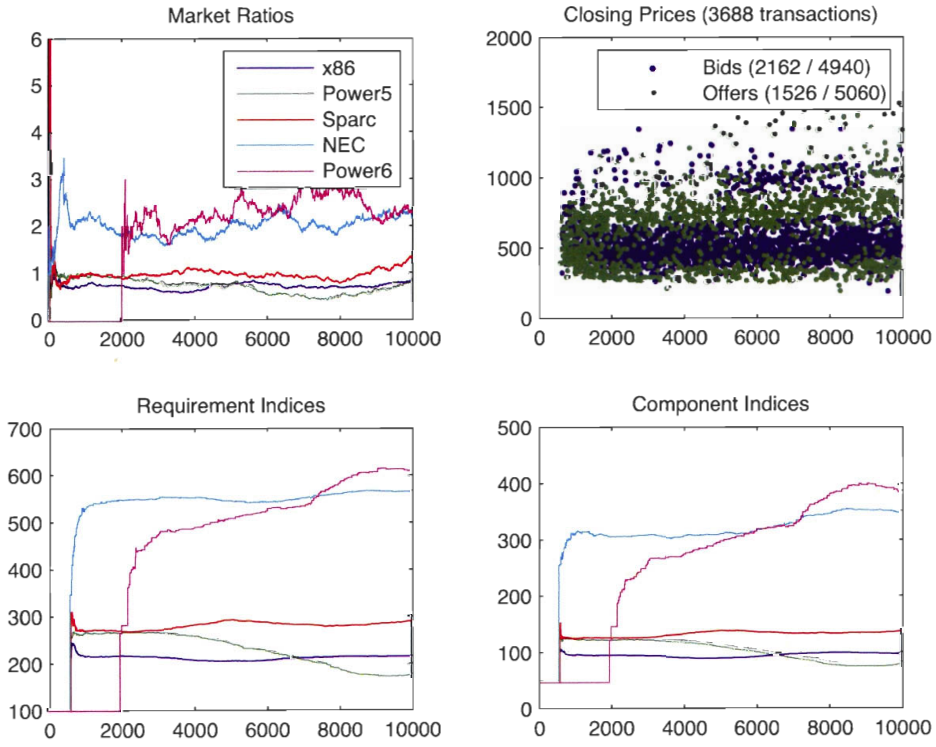

Figure 4.45: *Introduction of a new processor architecture resource*

4.11 The Quest for Market Indicators

This chapter demonstrated the application of a dual index scheme that supports the Grid Exchange stability, adaptability and efficiency. One might wonder why this thesis develops this entire bid and offer index duality and why a unified index scheme could not have been developed? The very first problem that comes up when it comes to trade HPC resources refers to the settlement point. If a user requests 4GB of memory in his bid, it means any provider offering 4GB or more can match it. From this sole consideration, we are already far from a standardized conventional commodity contract. Hence, we could define market indices after the resource level required on a trade, the resource level offered in the same transaction, or any mix of the twos.

As a simple demonstration, Figure 4.46 exhibits a market following an indicator based solely on bid requirements. For this market indicator, component indices are completely put away, and thus on each transaction we still compute requirement indices, but only them. To estimate their market positioning, providers then use the requirement indices of their resources highest available level of service, or stated otherwise, trading all of their resources at their maximum utility, what buyers would pay for these resource levels if they required it.

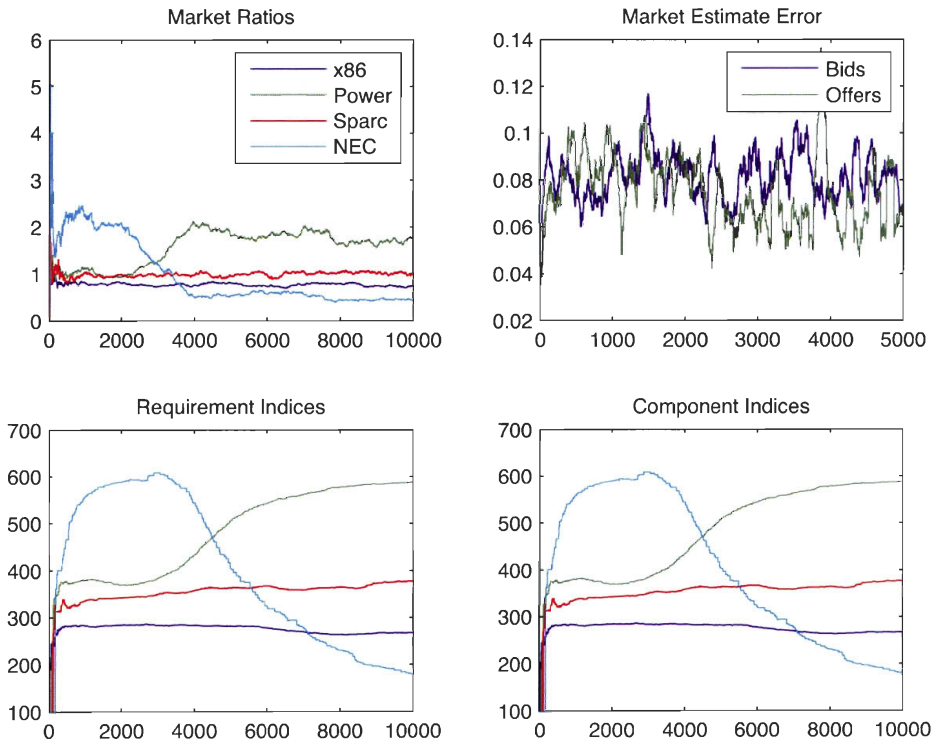

Figure 4.46: *A single index market indicator based on requirements*

Right away, we can observe this market indicator stabilizes to expected relative values for processor architectures and its error rate seems comparable to the double index developed before. However, comparing this single index adaptation to changing market conditions with our previously defined double-index (Figure 4.47), it is a clear looser as it takes more than twice the time to adapt. This is in big part due to the fact that the double-index represents the market in a broader way, as on each transaction, component indices are updated from the required resource level to the offered level, thus leading to a faster adaption of providers estimated pricing, leading to an enhanced adaptation in closing prices.

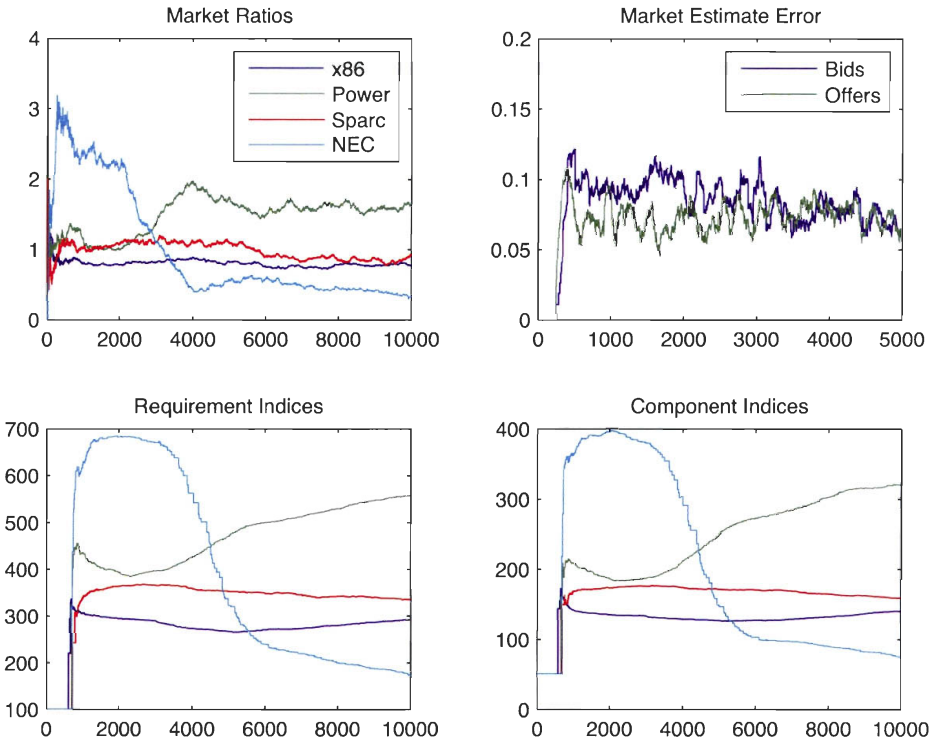

Figure 4.47: *Double-index scheme presented in this thesis*

As both single and dual-index market predictors error rates are comparable, it implies a good market efficiency for both. What's also interesting to observe is that absolute requirement indices settle to similar values in both cases. As transactions are concluded without a much higher error rate on the provider side, it means that our previously defined component indices are a valid sub-distribution of a resource value within the range of a component offer.

As a curiosity, we then define a unified resource index after the provider's top resource level, in Figure 4.48. In this scenario, the transaction closing price is split amongst the component resources top levels (and not split across the components resources sub-ranges), and requirement levels are completely ignored. For bidding purposes, buyers simply use these newly defined indices, in some way reflecting what the market would charge them, taking

into account a provider's maximum level of service for each resource.

Unsurprisingly, the market just skyrockets since users are now estimating their bid from the ratios and indices of top level provider components, although most users probably requested lower requirement levels, in previously concluded transaction. As these component levels present the more valuable ratios (recall Figure 4.2), market indices are simply pushed up following simple economic rationale.

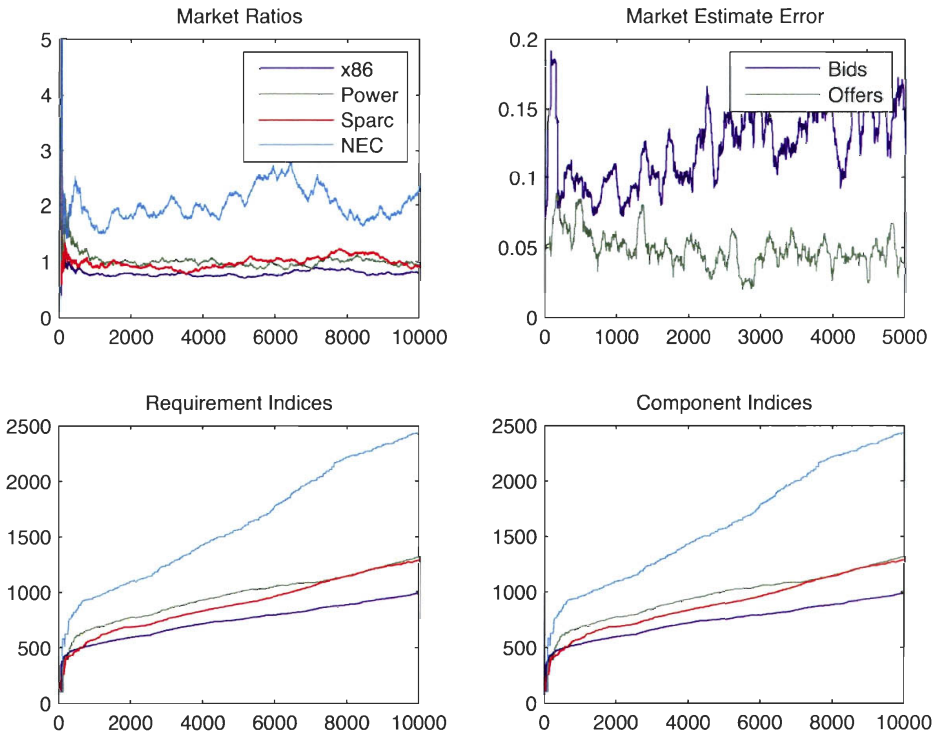

Figure 4.48: *A single index market indicator based on components top-levels*

This chapter presented a market simulation where buyers and sellers followed market estimators within specified tolerances to assess their positioning on a Grid Exchange. In some way, is it a false representation as there is just no way to model traders emotional behaviors and the sometimes un-rational moves on the market. Nevertheless, all things being equal, especially on such a complex goods trading instantiation, there is a good probability most traders would simply follow the market as their prime business is in the scientific applications of HPC, and not in HPC resources trading itself.

All in all, there is no such thing as a definitive market indicator, estimator, predictor or any other representation of previous events. In the end, an HPC resources trading system should just enable buyers and sellers to steer the market at their will. As for any economy, they are the only actors on market equilibrium. However, as HPC resources are complex sets of goods, there is a need to define market indicators, supporting *price-discovery*, consequently enhancing volume and, most importantly, asset liquidity.

Since the historical transaction closings information should be made available to everybody, there could be as many HPC resources valuation estimators as there would be traders on the market. The approach presented here borrowed concepts from technical analysis, building moving averages from previous transaction information. However, as this multi-commodity market inner workings could be quite complex, the back-propagation of closing prices onto the various resources enabled the extraction of some market fundamentals, similar to price over earnings ratios, that could then be used and compared to analyze any position. This thesis therefore built both fundamental and technical analysis foundations of an eventual supercomputing futures exchange; many ensuing refinements can now be envisioned.

For further study and analysis, this chapter's complete simulation results and *Matlab* code are available at this address:
`http://www.nic.qc.ca/thesis`

Chapter 5

From Infancy to Adulthood

In order to control volatility, build traders confidence and increase trading volume, any emerging market should be carefully supervised by a governing entity. This chapter therefore presents some recommended guidelines to be considered in the process of bootstrapping a Grid Exchange all the way to its development into a mature and viable economy.

Chapter 4 presented a market simulation demonstrating how ratios and indices were useful in bringing users and providers to trade on a Grid Exchange. The processor count and timeslot information were deliberately set aside as they did not shed any additional light on the analysis. As such, the number of processors requested or offered in market orders is somehow similar to contract sizing on conventional commodity markets, and thus the Exchange simply handles the matchmaking process.

However, timing information is an important factor to consider, as short-term and long-term futures can vary quite a bit, in the case traders anticipate a technology shift, for instance. But as far as simulation goes, there was just no need to study multiple timeframes scenarios since generating probablistically distributed random events imitated the core behavior of the market to be instantiated. Pushing the analysis one step ahead, in a mature and liquid Grid Economy, market orders would get posted for short-term and long-term timeframes, thus leading to somewhat "parallel" markets, as traders react to future potential events and adjust their initial long-term positions as they come all the way to the active month.

At first, fewer resources, coarser granularity in resource series like memory or processor clocks as well as less future timeframes should be enabled in order to build liquidity. As time goes, with more and more traders entering the market, resources types and sub-types could be further expanded and additional future months added. Therefore, from an initial single next month (i.e. "active") HPC futures markets, the Grid Exchange could eventually trade up

to several months, and maybe more than a year away HPC futures.

Hence, from that single month limited resources Grid Exchange, the market supervisory entity will gradually enable newer resources for trading. In doing so, it could initialize these new resources ratios and indices at some "expected" valuation, in a comparable way to a stock IPO entry pricing, or leave them all to default values. As we have seen previously, leaving them to some arbitrary default value is not a problem by itself as the market will reach equilibrium anyway. However, an adequate resource IPO market estimate could limit the initial instability and should therefore be conducted.

As for the future months to be instantiated, the active month and next month should probably be the only ones traded initially. As soon as these markets drive enough volume, then the second next month should be enabled, then the third next month and so on. These new trading months inceptions should inherit the previous month ratios and indices, in order to build on the statistical data previously acquired.

At the beginning, bids and offers could be for 6 hours timeslots to be delivered anytime within the specified month timeframe. While this is somehow mimicking conventional commodity markets, before too long, the Grid Exchange should enable narrower contract fulfillment periods as supercomputing needs tend to be more specific in terms of timing and no physical commodity delivery has to be put in place. Hence, following sufficient trading volume, the Grid Exchange management entity should eventually switch from future months to future weeks, then future days and even, eventually, future specific timeslots.

While the fundamental reason supporting resource discretization is to safeguard against illiquid assets, as time goes by, following the market's rising volume, ratios and indices should be revised to consider a larger, and more precise, spectrum of data.

This chapter introduces some enhancements to ratios and indices calculation methods to account for resource scarcity. Technical issues related to quality of service, benchmarking, trust management and credentials validation are then discussed. Traders activities and their implications are analyzed in the context of a burgeoning supercomputing economy and finally macro-economic aspects such as currency management, taxation and credits re-allocation are further explored.

Worth noting, while the next pages consider these important aspects in the transition process from the actual sharing model to a freely-evolving economy, none of them is unconditional. In the end, there will probably be as many Grid social-economic models as there are national Grids around the world, some of them being more liberal, some others more conservative; as they should reflect their users collective vision of HPC resources sharing.

5.1 Enhanced Ratios

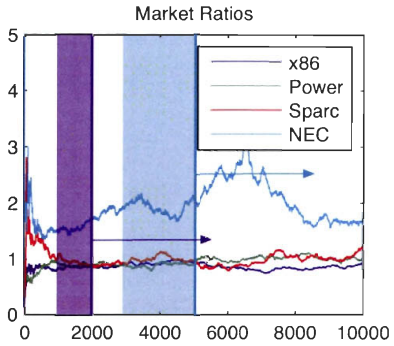

Figure 5.1: *"Adaptative" ratios*

As witnessed in the multiple simulation runs, ratios and indices for less liquid resources had a tendency to be slightly more unstable. Taking the processor architecture default probability distribution as an example, over 1000 events, 45% were for x86, 20% for Power and Sparc and only 15% for NEC. In the NEC case in particular, of the 15%, only 5% were offers. When these somewhat infrequent events occurred, ratios experienced some obvious instability. In order to compensate for scarce resources, *adaptative ratios* should be further investigated and most probably implemented in a real world deployment. Hence, instead of considering a fixed time-window for all resources, that window should expand dynamically to cope for less liquid assets. As shown on Figure 5.1, the NEC window (light blue) is much larger than the x86 window (dark blue), therefore bringing enhanced stability for the NEC ratios and greater adaptation for more frequently traded resources like the x86.

In a real world economy, in order for market ratios and indices to compensate for unreasonably high offers or low bids, *weighted market ratios* should be implemented to reflect bid and offer pricing relative gap from market estimators. For example, in the case of a bid with a market pricing estimated at 500 gc that would get posted at 100 gc, market ratios should not be influenced as much as for a bid posted "at the market". On the provider side, we should do the same for over-priced offers. Within the simulation, there was just no need to implement such a mechanism as the computer had precisely defined bounds to trade within, and hence no trader tried to steer the market. In a real world, there should be a provision against it.

Following the economy bootstrapping initial stages, available timeslots for trading will get expanded up to a year (or even more). As the market grows and becomes more and more liquid, it will enable time differentiation in the calculation of ratios. For many reasons, ratios for the March 2009 timeframe could be somewhat different from the ones in the October 2008 timeframe. Hence, we can already envision enhanced *proximity ratios* that would account in the timeframe for which a bid or an offer has been posted. For any specific resource timeslot, we could consider any such resource's previous 100 bids and offers and similar subsequent time-window for calculation purposes. Our simulation treated incoming independent events as if there were all for a unique timeslot. In a more realistic market instantiation, for any timeslot t, there would be bids and offers within the interval $t-\delta t$ and $t+\delta t$. Thus, computing ratios over such a pre-defined time-window would be a definitive improvement, for ratios, but also for indices, as demonstrated in the following section.

5.2 Market Indices Revisited

The previous section raised some interesting considerations about the time correlation of significant historical data to determine resource ratios. Market indices can also be revisited following the same reasoning. In this case, however, we may not only consider the execution timeframe for indices calculation, but also the transaction timeframe, that could influence pricing in many ways. Intuitively, we can envision HPC resources pricing starting at a base price level in a distant future, growing incrementally approaching present time, and maybe dropping sometime before the expiry of the timeslot (however still accounting operational costs, see section 5.6).

We thus introduce a new variable to consider: Δt, representing the amount of time from deal time t_0 to execution time t_1. While t_0 on itself could be considered, to limit the amount of data analyzed, we suggest to use Δt, since, combined to t_1, it intrinsically includes t_0. A larger Δt would therefore suggest a more speculative deal while a minimal Δt might exhibit a "discount price" for remaining time-slots.

For every timeslot, lets consider a 1 year range for Δt. For example, considering the 6 hours time-slots within a 10 days period as similar and using Δt increments of 10 days over the year, we would have $4 \times 10 \times 365 \div 10 = 1460$ possible input data representing a single resource index, for both $I^C_\phi(t)$ or $I^R_\phi(t)$. Since such a table would be populated by only a limited number of historical data, the question becomes how to appraise any single combination of t_1 and Δt for every resource index?

Most probably, neighboring timeslots should be market-valued in a similar fashion. As such, timeslots executing within a week should be more correlated than timeslots 6 months apart. Also, for a defined timeslot execution time, the ones dealt within a certain interval will present similar characteristics. Hence, considering a specific resource timeslot to be sold 6 months before execution, we might want to consider the same resource timeslots exchanged from 5 to 7 months before their execution.

Therefore, instead of applying a linear average over every past transaction, we propose to use a gaussian averaging method where neighboring data is more significant. For example, if we want to appraise a given resource for a timeslot on January 6, 2009 from 00h00 to 06h00, to be dealt 90 days before execution, we propose to use a 2D gaussian weighting function of this form:

$$\Gamma^{\Delta t}_{t_1} = e^{-1/2(\mathbf{x}-\mathbf{m})^T \Sigma^{-1} (\mathbf{x}-\mathbf{m})}, \tag{5.1}$$

where $\mathbf{x}$ is vector $[t_1 \ \Delta t]^T$, $\mathbf{m}$ is the mean vector of the distribution, and Σ its covariance matrix.

Figure 5.2 shows a graph of this representation for this example with $\mathbf{m}$ =[01/06/09@00h00 90 days]^T and

$$\Sigma = \begin{bmatrix} (60 \text{ hours})^2 & 0 \\ 0 & (45 \text{ days})^2 \end{bmatrix}.$$

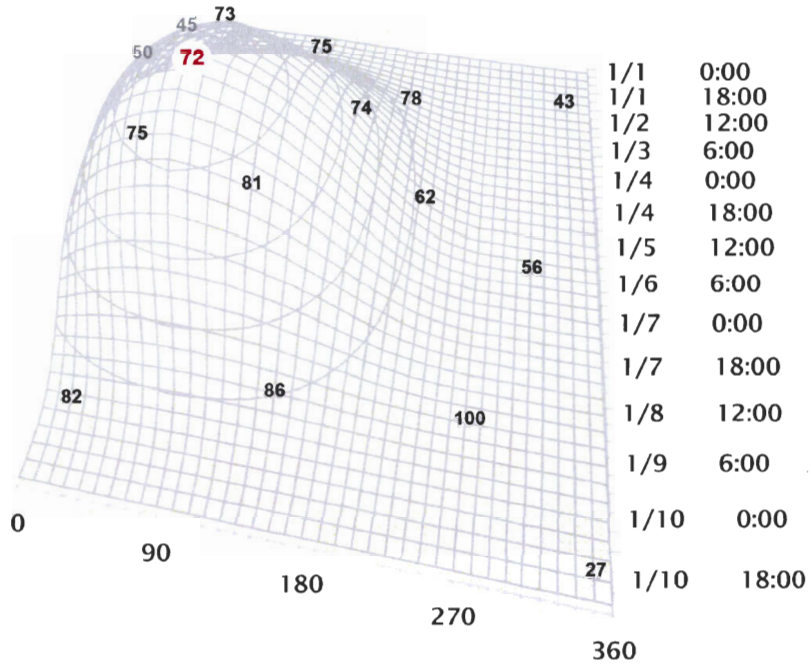

Figure 5.2: *Gaussian averaging factors applied on previously computed indices*

Hence, we state that for every timeslot t starting at t_1 and exchanged Δt before execution we can compute its resource indices $I_\phi^C(t)$ or $I_\phi^R(t)$ using neighborhood information through a gaussian approximation and then:

$$I_\phi^C(t) = \frac{\sum_{t_1 \in \tau} \sum_{\Delta t \in \omega} I_\phi^C(t_1, \Delta t) \times \Gamma_{t_1}^{\Delta t}}{\sum_{t_1 \in \tau} \sum_{\Delta t \in \omega} \Gamma_{t_1}^{\Delta t}} \tag{5.2}$$

where we consider relevant historical data by applying the gaussian factor $\Gamma_{t_1}^{\Delta t}$ to each registered deal information within the considered execution timeframe τ and deal timeframe ω. In this case τ =[01/01/09 00h00, 01/10/09 00h00] and ω = [0 days, 360 days].

Figure 5.2 presents an example where a resource index, whether $I_\phi^C(t)$ or $I_\phi^R(t)$, is appraised using this algorithm. As we can see, the table is populated by very little historical data. To compute our estimated resource index for (01/06/09 00h00, 90 days), highlighted in gray, we will consider every resource index in the table, but with a relative significance. For example, the 73 gu for (01/05/09 06h00, 90 days) is counted with a $\Gamma_{t_1}^{\Delta t}$ of 0.8983 while

(01/10/09 12h00, 350 days) at 27 gu hardly counts with $\Gamma_{t_1}^{\Delta t} = 0.0008$. Highly correlated data therefore gets a bigger impact on the computation of our market estimators.

This mechanism brings enhanced support for HPC resources trading transient states, an essential aspect for any such infrastructure to ramp-up and grow. In the beginning, historical deal information might be very scarce. Our previous approach considered all deals equal whatever their timeframe and applied a simple moving average. Although this could work on an initial limited market scale with low volume, users will eventually require a system more responsive, able to adapt faster and more representative of timing differentiations. From this concern comes the inspiration for the gaussian weighted average presented here. With very few historical data, this system would build more momentum, more quickly, even with "distant" information. In the long run, this gaussian pricing algorithm would then simply refine its approximations.

This gaussian averaged valuation algorithm is also adjustable by tweaking the variances Σ of the distribution, therefore altering the curve steepness. For example, Figure 5.3 presents a gaussian function where the variance is adjusted to consider more or less of relevant data. Also, any modified gaussian, sigmoid or asymmetric function could also be used to weight differently shorter deal-execution intervals, Δt, than longer ones.

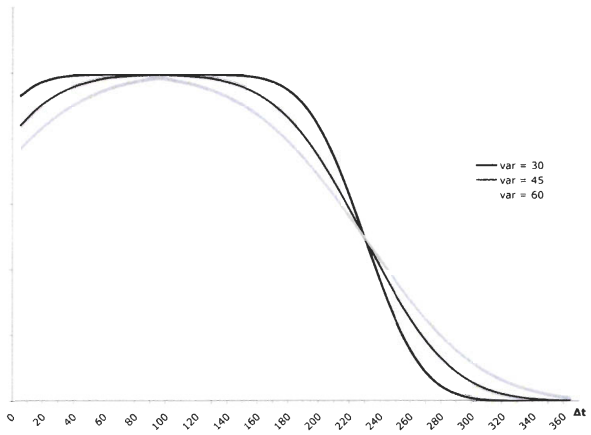

Figure 5.3: *Gaussian Algorithm Parameter Tweaking*

This section presented a gaussian averaged valuation algorithm using 2 variables, t_1 and Δt, it could however be refined to incorporate numerous other dimensions like the time of day, the day in the week, holidays, etc. Finally, one must never forget that this algorithm is only an approximation of what could cost a given resource for a specific timeslot. It is not intended to be an absolute market regulator. It is to be used by the consumer and the provider in their specific context to have an idea of what might be worth their requirement or component sets.

5.3 Contract Fulfillment and QoS

As we have seen in chapter 2, commodity contracts specify some very precise quality requirements that any provider should equal or exceed when delivering the traded good. As an independent third party, the clearing house is responsible for enforcing quality control and producing the necessary certification documents.

In order for an HPC futures market to be sustainable, similar independent quality of service (QoS) assessment mechanisms should be implemented. However, evaluating QoS on supercomputer nodes may turn out to be somewhat more complicated than checking commodities like oil or metals.

First, the *Grid clearinghouse* should validate allocated provider resources match its advertised resources levels. This could be fairly easily conducted using various benchmarks and test codes prior to allocation to the buyer. Nevertheless, in order to make sure the consumer gets what he paid for, quality of service validation routines should be executed at run-time to verify the client's code does not get co-allocated with various other codes. However, standing on the other side of the fence, no provider can guarantee 100% *user time* [1] to the client's application as there will always be system daemons that should be run simultaneously.

Hence, providers should be able to specify a level of utilization they guarantee to the application code. As penalties to be enforced by the Grid Exchange will be substantial, providers should be conservative in their various resources advertised levels. In this regard, Equation 3.27 could be modified to incorporate an advertised resource utilization factor δ_c, that could be determined automatically upon prior job runs or manually by the provider.

$$\mathcal{P}_C = n * \int_t \sum_{\phi \in \Phi_C} \delta_c \mathbf{I}_\phi^C(t) \tag{5.3}$$

Thus, for resources "shared" at run-time such as processor time or memory bandwidth, the provider can state a pre-defined level of service he is absolutely confident to deliver. For other resources like memory size or storage size, it would probably be a wiser choice to simply post a guaranteed lower level of service instead of using δ_c. Afterwards, using transactions that included several δ_c, ratios and indices calculations should be compensated following a $1/\delta_c$ factor in order not to skew market estimators.

Finally, as user satisfaction and vendor commitment definitely carry an intrinsic value in every economy, QoS information feedback, similar to *ebay ratings*, could be a good addition as trustable Grid nodes are clearly of higher value than unstable ones.

[1] as in the unix *user time*, *system time* and *idle time*

5.4 Trust Management and Credentials Validation

On the futures market, an entire hierarchy of trust starts from the Exchange, who validates the credentials of the various brokerage firms, that are then ultimately responsible for their client orders. Hence, traders are imputable to their brokers, and brokers to the Exchange. In order for them not to end up paying for their client losses, brokerage firms maintain security margins and perform *margin calls* as soon as one client position could end up losing money from an adverse market shift or because or an increased volatility leading to more slippage when exiting the position. Over this comprehensible 3-level structure, trust is thus delegated in a simple and effective manner and imputability is assured through coherent accounting procedures.

Trust and identity management, authentication and authorization have always been core issues in Grid deployments. Actually, the various Virtual Organizations (VOs) deal with this difficult question using centralized Certification Authorities (CAs) that every user and provider must trust. While a global federation of CAs might be a desirable long term goal, credential validation could simply be performed through an independent third party: the Grid Exchange Authority (GEA). Implementing a root CA for its own domain, the GEA would then perform delegation by issuing second-level CA certificates to consortia administrations that would then issue leaf-level certificates to their own users. Eventually, consortia CAs could also issue third-level CA certificates to accredited research groups principal investigators, that would then be capable of issuing credentials to their lab members.

In a manner similar to conventional commodity markets accounting, following transaction closings, the Grid Exchange should then debit and credit the various users and providers accounts. If, by any means, one user would end up incapable of covering its position, the parent authority's account should be debited.

From such an hierarchy of trust, the Exchange is able to validate and accept the various market orders it receives. However, there should be a way for the Exchange to "transfer", for any transaction, the user's and provider's credentials to each other. Hence, once a transaction is completed, in order for the provider to perform proper authentication, the Exchange would then issue a time-stamped *proxy certificate*[152] to the client. The client would then perform login on the provider nodes by transmitting that temporary certificate. Since that certificate is signed by the GEA, the provider could then authenticate the right client for the appropriate timeslot and grant him access to the traded resources. Since the Exchange also transmitted the provider's credentials to the client, he is also able to validate he is indeed talking to the right person. Therefore, the user/provider authentication and authorization scheme becomes pretty straightforward, as everybody is trusting a unique root authority, the GEA.

5.5 Market Actors

From a somewhat technical standpoint, in terms of market liquidity enhancement measures, contract standardization, quality of service enforcement, as well as trust and identity management, we have seen how relatively easy it would be to deploy and ramp up a supercomputing futures economy. We now take a step back and look at it from a financial perspective, in order to outline the major market actors, their goals, and the measures that should be taken in order to inhibit unfair market actions like *currency hoarding* or *market cornering*.

5.5.1 Hedgers

Hedgers are the prime users of any economy, their first and foremost objective is to guarantee either access or delivery of the needed or produced resources, for short or longer term outlooks, while reducing their risk at a minimum.

High-performance computing application scientists goals are in their respective fields, and, for most, not in supercomputing performance characteristics or benchmarks in any way. Therefore, all they want is to get some assurance their code and data will get executed in specific timeframes; in order to match publication deadlines, for example. In this regard, they express preoccupations similar to conventional hedgers, that are willing to get a reliable commitment on the delivery of raw materials at the base of their manufacturing process. In order to get such assurance, hedgers are willing to pay a bit more than the best deal they could get on the market in an uncertain future. Application scientists, or HPC hedgers, should manifest similar behaviors. Over time, building on their previous deals closing prices, they should be able to anticipate resource pricing and then express their computing needs in Grid Credits (gc) enabling them not only to budget their research funds expenditures, but also their gc distribution over the various projects and graduate students they supervise.

HPC resource providers, on the other side of the equation, are mostly capitalized through government funding agencies and their prime objective is simply to provide resources with high QoS to users on the Grid. While trying to maximize the gc earnings over their funding allocation period[2], they want to limit the risk of a radical new technology making their technological choice completely deprecated. Over a 2 to 3 years funding allocation period, major resources providers should then short long term positions in order to guarantee minimal gc revenues and assure their sustainability.

[2] we'll elaborate on this topic in section 5.7

5.5.2 Speculators

As we have seen previously, commodity markets volume does not come from hedgers, but from speculators, who account for more than 90% of trades for most futures. The instantiation of the Grid Exchange could thus give birth to a new breed of traders: *computer technology speculators*. In the oil business, successful traders anticipate the oil price movements better than others. It's just the same thing for any other conventional commodity. In the proposed vision, computer "geeks" would now be able to profit from their passion, but not by programming or designing computers, but by trading technology trends!

Not only would that be the coolest financial market for any computer aficionado, but their presence on the market would drive the essential liquidity a supercomputing futures economy needs to be pervasive and sustainable. Many very interesting financial instruments and market orders can then be envisioned, from HPC options, to shorts, spreads and every other concept we have seen in chapter 2. All such metaphors should be strongly encouraged by the Grid Exchange Authority, since, should it be repeated, "a liquid market is a healthy market".

5.5.3 Third Party Resource "Transformers"

The other very interesting and maybe quite profitable option emerging from the Grid Exchange for HPC gurus lies in the "transformation" of cheaper "raw" provider components into more exotic, higher abstraction resources. As an example, let's consider a rather common case where some application scientists require some high abstraction resources, like BLAST iterations, but they don't care about the kind of hardware BLAST is executed on.

Resource providers, on their end, may not be willing to support each and every version of every single HPC code available. For instance, many BLAST versions exist, either optimized for specific CPU architectures, FPGAs, or addressable memory configurations. Also, because of kernel modules dependencies or other considerations, users may request one of the many versions of BLAST and related kernel, for a specific hardware installation. But as supercomputer facility operators already have a burden of system management tasks to execute, some remote user's very specific needs are rarely of first importance. Henceforth, instead of trying to satisfy everybody, system admins could simply provide *stateless*[3] compute nodes with *on-demand provisioning*[4], and thus leave the burden of software configuration to the user purchasing compute time.

[3] without residing OS, network-booting (PXE-boot) with hard disk, if present, for local scratch purposes only

[4] where a specific system image is fetched to every node at run-time, following scheduler allocation

Users requiring high-abstraction resources and providers willing to ease their life and provide raw components... In finance, that's called *opportunity*.

Third-parties could then specialize in developing system images tuned for the various application fields, and for the multiple resource types available on the Grid Exchange. Now, this is leverage! Imagine leveraging a research group or a system administrator expertise in computing scientific datasets for an entire community, without the requirement for the expert to have a dedicated access to an equivalent supercomputing capacity. Possibilities are endless. Not only does it uncouple the expertise from the computing facility or the scientific research lab staff, but it empowers people that are good at what they do by enabling them to share their knowledge, and benefit from it.

In the end, it all comes down to liquidity management by the Grid Exchange, that should gradually enable higher-abstraction resources trading, as the economy grows and prospers.

5.5.4 Big Players

In 1979-1980, the Hunt Brothers, at the time possibly the richest family in the United-States, decided to buy precious metals as a way of hedging against inflation. Since gold could not be traded by individuals at that time (only states), they formed a silver pool with some Arabs and shortly amassed 200 million ounces of silver, or about half the world's deliverable supply. Early in 1979, silver traded at 5$ an ounce, early 1980, it went up to 54$ an ounce. As the Hunts *cornered* the silver market, stock exchanges, interest rates and about every aspect of the economy felt some side-effect. But then, the New York Metals Market, part of COMEX, changed trading rules and thanks to the Federal Reserve intervention, that game was put to an end. The Hunt brothers declared bankruptcy.

Market cornering could also be performed the other way around, with massive short positions, therefore controlling the buy side, and possibly bringing producers on their knees.

Because of their massive influence on the market, such illicit trading strategies should be carefully monitored by the Grid Exchange Authority as they both can constitute a serious threat to market stability and long term sustainability. Rules preventing the control of a significant part of a commodity's futures should be put in place. In addition, as the Grid Market Exchange will operate in a closed environment at the beginning, Grid Credits liquidity incentives should also be established.

5.6 Trading strategies and automated pricing

When defining market strategies, providers might want to consider acquisition and operation costs in their algorithm. Also, consumers might want to steer their bidding strategy depending on result deadlines. They could therefore set a basic price P^R_{base} for their requirement set or P^C_{base} for their component set and specify a weighting function between this "desired" price point and the market-driven estimate.

As an example strategy, we thus introduce two similar pricing function presenting these two components: basic (desired) and market (statistical), first for consumers:

$$P_R(t) = \alpha_R(t) P^R_{\text{base}} + (1 - \alpha_R(t)) P^R_{\text{market}}(t), \tag{5.4}$$

and then for providers:

$$P_C(t) = \alpha_C(t) P^C_{\text{base}} + (1 - \alpha_C(t)) P^C_{\text{market}}(t), \tag{5.5}$$

where the price for a requirement set P_R or a component set P_C at time t is determined by a basic price $P_{base}(t)$ and a market-driven statistical index $P_{market}(t)$ for that specific set of job requirements or supercomputer configuration. $P^C_{base}(t)$ is assessed by the resource provider upon original and maintenance costs. Likewise, the consumer can determine a basic price, $P^R_{base}(t)$, he is willing to pay for compute timeslots in the long run.

The α parameter is used to tweak the relative significance of the trader's desired price point versus market-driven inputs. It is a function of t and could vary linearly, exponentially or following any user-defined curve. This parameter shall be determined by the resource owner and, similarly, by the compute time consumer.

Intuitively, as shown in Figure 5.4 (a) , we can think of a resource provider setting his α parameter (small dots line) such that his resources pricing (black line) are starting at a base price level (red line) in a distant future, then grow incrementally approaching present time, following market demand trends (green line), and end up dropping drastically sometime before the expiry of the time-slot. This behavior is somewhat comparable to airplane ticket prices or other commodity markets. It reflects the "blue-chip" aspect of long term bidding, the speculative increment, and the fact that remaining cpu cycles, if not sold, will be lost.

Similarly, Figure 5.4 (b) shows a client strategy and related market trends. While we can observe that this client will try to bid on cheap resources in the short to medium term following his $P^R_{base}(t)$ (in yellow); coming to an important deadline, his $\alpha_R(t)$ plunges and even gets negative, therefore pushing up his pricing function $P_R(t)$ (in black), as he is willing to pay more than the market price (in blue) in order to make sure his job gets computed.

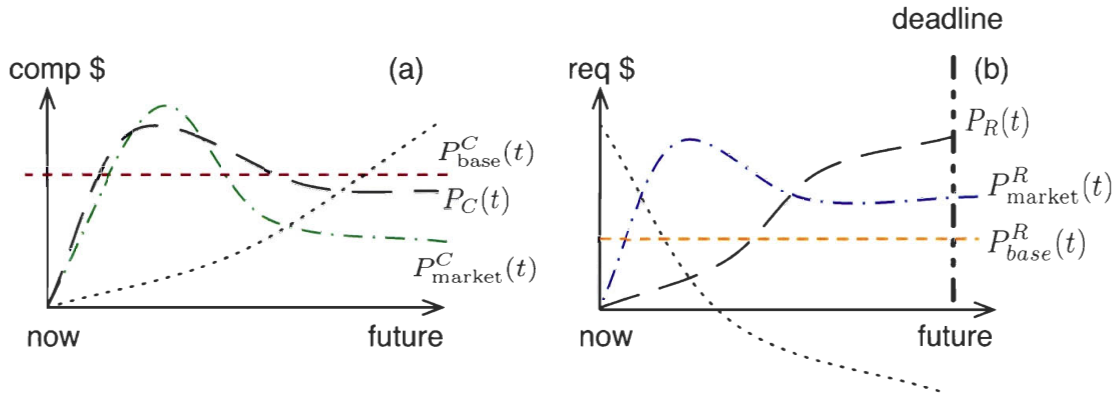

Figure 5.4: *Components (a) and Requirements (b) Pricing Graph*

Overlapping the two resulting pricing functions $P_C(t)$ and $P_R(t)$ in Figure 5.5, we can observe an opportunity window for these 2 traders starting at time t_1. Because the consumer's job length λt, and the strict deadline, the farthest possible job startup would have to be at $deadline - \lambda t$. The shaded red and blue area outlines the possible closing price range if our 2 traders would end up dealing together. Both having pre-defined market strategies, it then becomes a matter of who posts the market order before the other: if the client posts first, he would end up paying more than the other way around. However, in all cases, there is a very good chance more than 1 provider would match this client needs, or correspondingly, more than a single client would be able to use the provider's resources. Hence, instead of posting market orders for more distant futures, a better client-side strategy, in this case, would be to raise his tolerance gradually, approaching his deadline. These 2 pricing functions where mainly used as a rather simple market positioning strategy example, in the end, there could be as many strategies, as complex and tuned, as there are traders on the market.

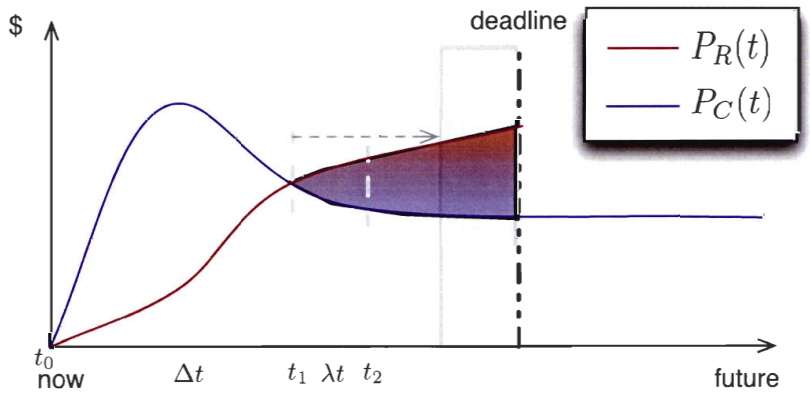

Figure 5.5: *Components (a) and Requirements (b) Pricing Graph*

5.7 The Value of Science: a Macro-Economic Exercise?

One of the first questions following the presentation of this Grid resource exchange model refers to the "value of science", or how to value, in HPC resources allocation, the research conducted in the various scientific labs...

The answer is quite simple, as nothing would really change. The use of an abstract currency, in this case called *Grid Credits*, is simply a mean to instantiate a standardized bartering instrument rather than trying to weight a good for a good, for every single resource "swap". In a manner somewhat similar to when users received allocations for specific machines on NPACI [19], the Grid Exchange Authority, tied to funding agencies, would simply allocate gc to users, starting with project principal investigators, that would then delegate a portion of their allotment to their researchers, graduate students and so on. This allocation scheme could be inherited from the funding agencies research funds allocation algorithm, issuing money to preeminent researchers and weighting that allocation depending on the various scientific fields intrinsic requirements.

At the beginning, users would receive gc from the governing agency, that would also have, directly or through some related entity, subsidized the various HPC resources across the Grid. Users would then have to spend responsibly this convertible and thus very precious asset through the multiple provider resources on the market. As providers would start earning Grid Credits, it would instantiate some form of competition between the sites, as to whom gets the best return on investment on the Grid Credits / subsidies ratio. Already, the Grid Market Exchange would have built two major incentives the actual Grid model lacks: a responsible resource usage on the user side and a enhanced quality of service commitment on the provider side, as they compete against each other.

From that high-level subsidies management perspective, we can then envision some potential delegated economical measures to be enabled by the national Grid Authority. As providers would earn Grid Credits on the Exchange, they could then be allowed to re-distribute a portion of these credits to their "related" users. Hence, from the initial National Resource Allocation Committee (NRAC), we can then imagine a local or consortia level wealth re-distribution system through re-seeded LRACs (local) and CRACs (consortium), somehow similar to provincial and lower-level social programs. As a positive side-effect, following the implementation of such a policy, every user, even if their application would have to be run externally, would have its local site efficiency and "profitability" at heart. The never-ending debates around supercomputing acquisitions could potentially come to an end; as the desired infrastructure would not anymore be the one system every individual user would push for its very own needs, but the system that would earn the most credits, on the Grid scale. Hence,

the more "profitable" a site would be, whatever the installed supercomputer architecture, the more "buying power" its related users would have on the Grid, just as citizens living in a vigorous exporting economy tend to increase their global wealth.

From that reasoning, it becomes straightforward that users would get a huge payoff to tune and optimize their codes to run on the "cheapest" (in terms of Grid Credits "cheapness") system possible. As high-end SMPs would probably carry a higher Grid market price than serial clusters, there would just be no reason to run embarrassingly parallel codes on such systems. Think about it: an incentive for users to tune their code to the best suited supercomputer architecture, wouldn't that be every site operator's dream?

Then, as Grid Credits start flowing between users, sites and consortiums, the governing entity will have to be very careful with currency management, limiting inflation, guaranteeing access to resources and making sure once the market has stabilized, that Grid Credits become a instrument with a high confidence level throughout the community.

From this sustainable but maybe not enough interventionist Grid currency management perspective, one may wonder how users not related to a "profitable and exporting" consortia would be able to survive. Again, it all comes down to the governing entity approach to the question. While they could potentially embrace economic liberalism without moderation, most chances are that they would instead establish some form of wealth sharing measures.

Income taxes could be calculated on the providers revenue and then redistributed to the less-wealthy individuals within this parallel economy. Capital taxes should also be implemented as they would prevent site administrators to keep earned credits within their account rather than redistributing it to the various users. Not only would such capital taxes enhance asset liquidity, it would also thwart currency hoarding, an unethical practice that could be used to corner the market down the road. In addition, equalization or *perequation*[5] measures could be instantiated in order to redistribute the wealth across consortias, to eventually bring the lagging ones up to speed.

In the end, the proposed economic model is an instrument that can be modeled on the community's ideology and aspirations. It provides a simple and effective way to weight resource exchanges between individuals, sites and consortias. It builds an incentive for people to share and leaves wealth management to the trusted entity, the Grid Exchange Authority or any other democratically elected institution. From liberalism to interventionism, throughout the world, as for real economies, Grids could be conjugated over the entire spectrum of social-democracy.

[5] similar to Canada's Federal Government equalization transfers

5.8 Currency Exchange

Historically, in every major infrastructure deployment, like railways and electrical grids, governments played a prime role in the early stages, financing acquisition and operational costs. As time went by, they gradually transferred some responsibilities to the private sector while carefully legislating the way they should conduct business.

While somewhat ideologically unpopular throughout the academic Grid community, the idea of gradually opening the door to the industry, both on the provider and the consumer side, may be worth considering. Although quite a few years down the road, by the time the Grid Market Exchange drives enough volume amongst the scientific community, funding agencies could be opened at enabling resource providers to re-sell some of their earned Grid Credits to private outsiders. Through a "foreign" currency exchange (forex, or FX), Grid Credits could then be converted to dollars and vice-versa. That would enable resource providers to finance rising operation costs without requiring more subsidies from funding agencies. On the user side, in principle, nothing would prevent researchers to resell their Grid Credits allocation against US dollars or vice-versa. Acting as a central bank, the Grid Authority would then want to closely monitor its currency intrinsic value, liquidity and inflation.

Nevertheless, in order for such a forex to be beneficial to everybody, the governing entity should carefully monitor the Grid Credits allocations and earnings, over the transition period from the actual sharing model to this wide-opened market. Doing so, it could initially limit the amount of Grid Credits to be exchanged to a precise percentage of a provider's monthly earnings, for example. On the consumer side, scientists could, at first, be forbidden from reselling their allocation and carefully monitored. As the economy grows and the Grid Authority's confidence in guaranteeing access to resources to its researchers, restrictive measures could be gradually softened.

As a positive side effect, opening the market to the private sector could incite some HPC vendors to enter the Grid Market Exchange and sell their "on-demand" facilities in exchange of gc. Their pricing model would therefore reflect market fundamentals, and not their own, somewhat flawed, proprietary valuation model. Their earned credits could then be re-sold through Grid FX to pharmaceuticals or scientists in need for more compute resources.

Bridging academia and industry, for both sectors consumers and providers, a currency exchange may therefore be a good thing, as it provides more liquidity to the market, an enhanced resource offering for consumers and a possibility to generate some revenue for academic sector resource providers. In the end, it is just a matter of osmosis management between the 2 worlds, an aspect to be carefully watched, especially in the beginning phases.

5.9 HPC Options

Although most people are unaware of their existence, options and derivatives drive a impressive trading volume around the world, everyday. While intrinsically more complex than stocks, derivatives and options are used by large trust funds and major economical actors (insurance companies, major resource producers and consumers, etc.) to hedge against adverse market movements.

Following a widespread and sustained installment of the Grid Market Exchange in the community and then possibly tied to the private sector, High-Performance Computing options could be an interesting instrument to consider.

Initially, traders could thus issue options through the academic Grid in a fashion similar to conventional options. As such, consumers expecting a need for future timeslots, without necessarily being 100% sure about that need (in the case of uncertain preceding experiments for instance), could acquire a *call option* and therefore receive a long position, once exercised. The option premium would therefore buy them a guaranteed access to resources, if need be.

From the provider standpoint, they could be able to benefit from an earlier than expected supercomputer deployment. Hence, assuming a January installment of new equipment, a provider would probably not want to go short before the April timeframe. However, in the case everything went perfectly when deploying the system, that provider could purchase *put options* for the March and February futures. He would then have the opportunity to exercise these options, and receive short positions, increase its revenues, without incurring the risk inherent to deployment.

The next step, tied to a well established Grid FX, could enable scientists, site operators and even HPC vendors to hedge future technology deployments. Expecting some future supercomputing architecture to accelerate their results, scientists could hedge against it in order for them to compensate, in the case it does not get delivered, and transfer their hedging profit to a larger acquisition of previous technology timeslots. Similarly, as vendors are frequently tied to processor availability constraints, they could then be able to purchase call options and exercise them in the case they could not deliver in time a specific machine.

These considerations clearly bring us into high-flying HPC financials; while it may look confusing at first sight, the intent is not to make a more complex story, but rather to illustrate how far and how sophisticated an eventual Grid Market Exchange could get. Supercomputing futures, first and foremost, address a current fundamental incentive issue. From then on, tied to the community's objectives, countless possibilities could emerge from this new paradigm.

Chapter 6

Conclusion

Across computer science, finance and economics, this thesis is fundamentally an incentive engineering endeavor to contribute a pervasive and sustainable high-performance computing resources exchange system. While the Grid has found rather successful implementations through projects like Globus [8], it fails to get widespread acceptance; its inherent sharing model being essentially flawed because of a lack of provider accountability and consumer responsibility.

Ideologically, a sharing model based on the user's and provider's "good-will" would probably be the most desirable. However, times have shown the majority of HPC users are most preoccupied by their very needs and couldn't care less for other's demands. As resource providers are already overwhelmed by their local users requirements, few find time to adequately support external researchers. Hence, if the Grid is not to implement a centralized sharing model, where funding agencies would *fairly and equally* micro-manage resource allocations and all of the Grid operations, we must envision a new sharing paradigm that would provide the fundamental sharing incentive to every user and provider.

In essence, the concept of a Grid Economy is nothing but a proposal to "weight" resource exchanges. As some form of arbitrary compensation algorithm would be doomed from its inception, this thesis proposes to use statistical market data in order to balance user's and provider's sharing actions. Instead of bartering resources *per se*, the concept of a Grid currency, a far more liquid and usable instrument, is introduced: the Grid Credits.

In the end, the proposed paradigm is nothing but a non-cooperative, zero-summed game, that eventually reaches equilibrium points [116, 115, 62]. Users and providers are then encouraged to "trade" on the market, at their will, and at the price they feel inclined to buy for, or sell for, whatever that price may be.

Resource allocation optimization throughout the Grid therefore becomes very straightforward, requiring a simple advance reservation mechanism linked to the exchange system. It is no longer a problem of complex meta-scheduling, where the control-loop struggled against the non-linear feedback of the Grid's undeterministic workload. Moreover, as the economic modular algorithms streamline the whole process and can provide guarantees over specific performance criterias, resource allocations are now conducted across the various grid sites, paving to way to a decentralized and much more scalable, robust and fault-tolerant HPC resources sharing system.

What the utility computing literature got right is the fact that users must not only be able, but strongly encouraged, to specify the value they see in having their job completed before some arbitrary deadline. However, trying to design systems to counter strategic user behaviors is probably fallacious, as some users would undoubtedly find a way around the system to get some personal benefit. Therefore, the model thus proposed allows Grid "traders" to be as strategic as they want, as long as they follow some high level regulatory rules, to be carefully applied by a Grid equivalent to the SEC (securities and exchange commission).

In addition, thanks to this economic model, both users and providers are now able to state their valuation and, as a group, steer the market accordingly. Any market-based scheduling alternative, where only users were able to influence pricing, was simply unsustainable, as any serious provider would want to have some leverage on its resources "earnings", especially at times of rising operation costs (electricity, human resources, etc.).

In many ways similar to conventional commodity exchanges, the proposed *Grid Exchange* enables trading of resources through double-auctions. However, since there is no such thing as a "standardized" contract in HPC, the framework makes it possible to buy or sell resource "sets", called *requirement sets* on the user side, and *component sets* on the provider side. In the beginning, resource "series" like memory size or interconnect bandwidth are to be discrete and coarse grained; in order to build sufficient asset liquidity. As time goes, granularity should improve to eventually enable trading over continuous resource specifications.

In such an emerging economy, consumers and providers have very little insight of what resources are really worth. A "momentum building" metaphor thus had to be designed in order to stabilize economy bootstrapping, stimulate trading and grow volume. Without any orientation mechanism, the Grid Exchange users would be lost in a noisy and unintelligible maze, as fundamentalists would be in the stock markets without income statements, balance sheets and the various financial ratios they use on a daily basis.

The market ratios and indices introduced in chapter 3 leveraged the historical transaction closing prices to provide statistical estimates for requirement sets and component sets. These

resource indices support the decision making process in a way similar to traditional fundamentals. Without such insight, traders would be lost without any factual comparison data, and markets would be quite speculative and more unstable.

Thinking about it, because of this multi-commodity trading system inner complexity, technical analysis techniques had to be borrowed and modified to provide market fundamentals. While hard-core fundamentalists could argue they could speculate on the high-level technological trends, without any common denominator to compare market orders with each other, even the Grid's "Warren Buffet" would probably get lost.

The sharing model thus presented supports a self-regulated economy while bringing objective statistical data to steer users decisions. Recall that resource indices do not constitute an absolute indicator to follow and therefore should not constrain the trading process; they nevertheless bring some market assessment, or *price anticipation*, to any Grid user before posting a market order, a definitive advantage.

Simulation in chapter 4 demonstrated the economic model efficiency and capacity to adapt to changing market conditions. Although many further improvements could be made to ratios and indices calculation algorithms, the ones proposed, through thousands of random events, showed their effectiveness at hinting both consumers and providers on what their market positioning should be, as these estimators were, on average, well within 10% (relative error) from the price at which the resources ended up being traded.

Considerations surrounding quality of service (QoS), benchmarking and trust management were then discussed, outlining the major pitfalls to watch for, and trying to define some best practices while deploying the Grid Exchange.

Some example trading strategies have been envisioned as a mean to stimulate further studies surrounding the Grid financials, and how users and providers could "play the game" over the short to long term scenarios.

Acting as a federal reserve equivalent, the Grid Exchange Authority, tightly coupled to funding agencies, would be responsible for the economy and its currency well-being, implementing a closed-loop Grid Credits policy and potentially imposing income and capital taxes to support wealth sharing measures. As the various providers would now earn Grid Credits for timeslots exchanged on the market, funding agencies would then be able to calculate return on investment (ROI) for the various sites, thus stimulating competition between them, fostering higher QoS and efficiency. On the user side, as they would now have to spend a portion of their currency allocation, it would inevitably lead to a more effective use of supercomputing resources across the Grid, through code optimizations and data set refinements.

As every scientist would now receive some basic Grid Credits allocation, and while being incited to use it more skillfully, a Grid economy could end up being more democratic than the actual model where researchers have to deploy sometimes significant efforts to gain access to HPC resources.

This *strategyful* approach thus paves the way to a more sustainable Grid resources sharing metaphor, where *share*, in order to be *fair*, can now be weighted.

In the end, this new sharing paradigm for the Grid instantiates an entire new economy with countless possibilities. Consumer and provider market orders strategies, third-party brokers through higher abstraction resources reselling, technology speculators that would sustain the market's liquidity, future supercomputer architectures risk hedging for vendors, scientists and providers, all sorts of options and derivatives; these are only some of the aspects that will be worth further research.

This thesis laid the foundations of an entire new spectrum of ideas to be investigated, researchers from both the HPC world and the economic community are then welcomed to join their knowledge in giving birth to a new system based a concept that could end up revolutionizing the way computer cycles used to be considered.

Bibliography

[1] Avaki web site. http://www.avaki.com/.

[2] Boinc web site. http://boinc.berkeley.edu/.

[3] Compute canada long range plan. http://www.c3.ca/LRP/.

[4] Compute canada web site. http://computecanada.org/.

[5] Condor web site. http://www.cs.wisc.edu/condor/.

[6] Entropia web site. http://www.entropia.com/ (offline).

[7] The EU datagrid project (web site). http://eu-datagrid.web.cern.ch/eu-datagrid/.

[8] Globus web site. http://www.globus.org.

[9] Gnu lesser general public license (lgpl). http://www.gnu.org/licenses/lgpl.html.

[10] Gpfs web site. http://www.ibm.com/servers/eserver/clusters/software/gpfs.html.

[11] Gridbus web site. http://www.gridbus.org/.

[12] International business machines (ibm) web site.

[13] Investopedia.com, your source for investing education. www.investopedia.com.

[14] Legion web site. http://www.cs.virginia.edu/ legion/.

[15] Lustre web site. http://www.lustre.org/.

[16] Mico corba: an open-source corba implementation. http://www.mico.org/.

[17] Microsoft com and dcom web site. http://www.microsoft.com/com/default.mspx.

[18] Mpi web site. http://www-unix.mcs.anl.gov/mpi/.

[19] National partnership for advanced computational infrastructure (npaci) website. http://npacigrid.npaci.edu/.

[20] Ocean project web site. http://www.cise.ufl.edu/research/ocean.

[21] Openmp web site.

[22] Teragrid web site. http://www.teragrid.org/.

[23] Top 500 supercomputers list (web site). http://www.top500.org.

[24] Univa ud web site. http://www.univaud.com/.

[25] Wikipedia.org, the free encyclopedia.

[26] Big mac currencies. *The Economist*, 1998.

[27] D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems*, 18(8):1061–1074, Oct. 2002.

[28] D. Abramson, J. Giddy, and R. Buyya. An economy driven resource management architecture for global computational power grids, Aug. 03 0.

[29] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), Oct. 02 2000.

[30] B. Allcock, J. Bester, and J. Bresnahan. GridFTP protocol specification. In *In SGF GridFTP Working Group Document*, 2002.

[31] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster. The globus striped gridftp framework and server. In *Proceedings of Super Computing 2005*, 2005.

[32] W. Allcock, I. Foster, and R. Madduri. Reliable data transport: A critical service for the grid. In *Building Service Based Grids Workshop, Global Grid Forum 11*, 2004.

[33] D. P. Anderson. BOINC: A system for public-resource computing and storage. In R. Buyya, editor, *5th International Workshop on Grid Computing (GRID 2004), 8 November 2004, Pittsburgh, PA, USA, Proceedings*, pages 4–10. IEEE Computer Society, 2004.

[34] K. J. Arrow and L. Hurwicz. *Studies in Resource Allocation Processes*. Cambridge, 1977.

[35] P. Asadzadeh, R. Buyya, C. L. Kei, D. Nayar, and S. Venugopal. Global grids and software toolkits: A study of four grid middleware technologies, July 01 2004. Comment: 19 pages, 10 figures.

[36] 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.

[37] J. Backus. Can programming be liberated from the von neumann style? A functional style and its algebra of programs. *Communications of the ACM*, 21(8), Aug. 1978.

[38] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[39] A. Barmouta and R. Buyya. Gridbank: A grid accounting services architecture (GASA) for distributed systems sharing and integration. In *17th International Parallel and Distributed Processing Symposium (IPDPS-2003)*, Los Alamitos, CA, Oct. 01 2003. IEEE Computer Society. Comment: 12 pages.

[40] J. N. Brown and H. S. Rosen. On the estimation of structural hedonic price models. *Econometrica*, 50(3):765–68, May 1982.

[41] R. Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, Melbourne, Australia, 2002.

[42] R. Buyya, D. Abramson, and J. Giddy. Nimrod/g: An architecture of a resource management and scheduling system in a global computational grid. In *4th Int. Conf. High Performance Computing in Asia-Pacific Region (HPC Asia 2000)*, 2000.

[43] R. Buyya, D. Abramson, and J. Giddy. Nimrod-G resource broker for service-oriented Grid computing. *IEEE Distributed Systems Online*, 2(7), 2001.

[44] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. *Special Issue on Grid Computing, Proceedings of the IEEE*, 93(3):698–714, March 2005 2005.

[45] R. Buyya and M. Murshed. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience*, 14(13–15):1175–1220, nov 2002.

[46] R. Buyya and S. Venugopal. The gridbus toolkit for service oriented grid and utility computing an overview and status report. In *1st IEEE Int. Workshop Grid Economics and Business Models (GECON 2004)*, 2004. Comment: 11 pages, 3 figures, 3 tables.

[47] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. *SIGOPS Oper. Syst. Rev.*, 35(5):103–116, 2001.

[48] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle. Dynamic virtual clusters in a grid site manager. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 90, Washington, DC, USA, 2003. IEEE Computer Society.

[49] A. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf. Performance and scalability of a replica location service. In *Proceedings of the International IEEE Symposium on High Performance Distributed Computing (HPDC-13)*, 2004.

[50] M. Chetty and R. Buyya. Weaving electrical and computational grids: How analogous are they?, Nov. 08 2001.

[51] B. N. Chun. *Market-based Cluster Resource Management*. PhD thesis, University of California at Berkeley, November 2001.

[52] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, may 2005.

[53] B. N. Chun and D. E. Culler. Market-based proportional resource sharing for clusters, Mar. 22 1999.

[54] B. N. Chun and D. E. Culler. Rexec: A decentralized, secure remote execution environment for clusters. In *CANPC '00: Proceedings of the 4th International Workshop on Network-Based Parallel Computing*, pages 1–14, London, UK, 2000. Springer-Verlag.

[55] B. N. Chun and D. E. Culler. User-centric performance analysis of market-based cluster batch schedulers cluster batch schedulers. In *CCGRID*, pages 30–38, 2002.

[56] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the First Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[57] C. H. Crawford, G. P. Bate, L. Cherbakov, K. Holley, and C. Tsocanos. Toward an on demand service-oriented architecture. *IBM Syst. J.*, 44(1):81–107, 2005.

[58] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the* 10^{th} *Symposium on High Performance Distributed Computing*, 2001.

[59] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the 1998 IPPS/SPDP Workshop on Job Scheduling Strategies for Parallel Processing, IPPS/SPDP'98 (Orlando, Florida, March 30, 1998)*, volume 1459 of *LNCS*, pages 62–82. Springer-Verlag, Berlin-Heidelberg-New York-Barcelona-Budapest-Hong Kong-London-Milan-Paris-Singapore-Tokyo, 1998.

[60] G. Debreu. *Theory of Value: An Axiomatic Analysis of Economic Equilibrium*. Yale University Press, 1972.

[61] K. E. Drexler and M. S. Miller. Incentive engineering: for computational resource management. In B. A. Huberman, editor, *The Ecology of Computation*, pages 133–176. North-Holland Publishing Company, Amsterdam, 1988.

[62] B. Ellickson. *Competitive Equilibrium: Theory and Applications*. Cambridge University Press, 1994.

[63] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM Press.

[64] M. Feldman, K. Lai, and L. Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 127–136, New York, NY, USA, 2005. ACM Press.

[65] M. Feller, I. Foster, , and S. Martin. Gt4 gram: A functionality and performance study. In *TERAGRID 2007 conference*, 2007.

[66] D. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. In *8th International Conference on Distributed Computing Systems*, 1988.

[67] D. F. Ferguson. *The Application of MicroEconomics to the Design of Resource Allocation and Control Algorithms*. PhD thesis, Columbia University, 1989.

[68] D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic models for allocating resources in computer systems. pages 156–183, 1996.

[69] I. Foster, K. Czajkowski, D. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. Modeling and managing state in distributed systems: the role of ogsi and wsrf. *Special Issue on Grid Computing, Proceedings of the IEEE*, 93(3):604–612, March 2005.

[70] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997. ftp://ftp.globus.org/pub/globus/papers/globus.pdf.

[71] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999.

[72] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, second edition, 2004.

[73] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proceedings of the International Workshop on Quality of Service*, 1999.

[74] I. Foster, C. Kesselman, J. Nick, S. Tuecke, and S. Fitzgerald. The physiology of the grid: An open grid services architecture for distributed systems integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, page 31, 2002.

[75] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, Fall 2001.

[76] I. Foster, A. Roy, and V. Sander. A quality of service architecture that combines resource reservation and application adaptation. In *Proceedings of the International Workshop on Quality of Service*, June 2000.

[77] I. T. Foster. Globus toolkit version 4: Software for service-oriented systems. In *NPC*, pages 2–13, 2005.

[78] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. F. Tuecke. Condor-g: a computation management agent for multi-institutional grids. *Proceedings 10th IEEE International Symposium on High Performance Distributed Computing*, 2001.

[79] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. Sharp: an architecture for secure resource peering. *SIGOPS Oper. Syst. Rev.*, 37(5):133–148, 2003.

[80] H. Geman. *Commodities and Commodity Derivatives: Modelling and Pricing for Agriculturals, Metals and Energy*. Wiley, 2005.

[81] A. Geweke. A system for batch-mode economic scheduling of a cluster of workstations. Master's thesis, University of California at Berkeley, 1997.

[82] Gibbins and Buyya. Gridscape: A tool for the creation of interactive and dynamic grid testbed web portals. In *International Workshop on Distributed Computing: Mobile and Wireless Computing, LNCS*, volume 5, 2003.

[83] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, 1992.

[84] C. Gray and D. Cheriton. Leases: an efficient fault-tolerant mechanism for distributed file cache consistency. In *SOSP '89: Proceedings of the twelfth ACM symposium on Operating systems principles*, pages 202–210, New York, NY, USA, 1989. ACM Press.

[85] A. Grimshaw and A. Natrajan. Legion: lessons learned building a grid operating system. *Proceedings of the IEEE*, 93(3):589–603, March 2005.

[86] A. S. Grimshaw and W. A. Wulf. The Legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, 1997.

[87] A. C. Group. Atlas computing: Technical design report. Technical report, CERN / LHC, http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/Html/Computing-TDR.htm, 2005.

[88] R. Gupta and A. Somani. Compup2p: An architecture for sharing of computing resources in peer-to-peer networks with selfish nodes. In *Proceedings of Second Workshop on Economics of Peer-to-Peer Systems*, June 2005.

[89] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[90] K. Harty and D. Cheriton. A market approach to operating system memory allocation. pages 126–155, 1996.

[91] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach, 3rd ed.* Morgan Kaufmann Publishers Inc., Palo Alto, CA 94303, 2002.

[92] L. Hurwicz. The design of mechanisms for resource allocation. *American Economic Review*, 63(2):1–30, May 1973. available at http://ideas.repec.org/a/aea/aecrev/v63y1973i2p1-30.html.

[93] A. S. D. B. A. D. A. G. B. H. F. M. F. S. R. S. J. T. J. V. R. I. Foster, H. Kishimoto. The open grid services architecture. Technical report, Global Grid Forum (GGF), 2005.

[94] D. Irwin, J. Chase, L. Grit, and A. Yumerefendi. Self-recharging virtual currency. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 93–98, New York, NY, USA, 2005. ACM Press.

[95] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum. Sharing networked resources with brokered leases. In *Proceedings of USENIX 2006 Annual Technical Conference*, 2006.

[96] D. E. Irwin, L. E. Grit, and J. S. Chase. Balancing risk and reward in a market-based task service. In *HPDC*, pages 160–169, 2004.

[97] K. Karasavvas, M. Antonioletti, M. Atkinson, N. C. Hong, T. Sugden, A. Hume, M. Jackson, A. Krause, and C. Palansuriya. *Introduction to OGSA-DAI Services*, volume 3458. Springer, 2005.

[98] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming Journal*, 2006.

[99] C. Kesselman, G. V. Laszewski, I. Foster, S. Fitzgerald, and S. Tuecke. A directory service for configuring high-performance distributed computations, 10 2001.

[100] C. D. Kirkpatrick and J. R. Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians.* Financial Times Press, 2006.

[101] G. Kleinman. *Trading Commodities and Financial Future: A Step by Step Guide to Mastering the Markets.* Financial Times Press, 3rd edition edition, 2004.

[102] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Comput.*, 38(5):705–717, 1989.

[103] K. Lai. Markets are dead, long live markets. *SIGecom Exch.*, 5(4):1–10, 2005.

[104] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent Grid Syst.*, 1(3):169–182, 2005.

[105] B. Lang, I. Foster, F. Siebenlist, R. Ananthakrishnan, and T. Freeman. A multipolicy authorization framework for grid security. In *Fifth IEEE Symposium on Network Computing and Application*, 2006.

[106] M. J. Litzkow and M. Livny. Experience with the Condor distributed batch system. In *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, pages 97–101, Huntsville, Alabama, USA, 1990. http://www.cs.wisc.edu/condor/doc/experience.ps.

[107] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor—A hunter of idle workstations. In *Proceedings of the Eighth International Conference on Distributed Computing Systems*, San Jose, California, 1988.

[108] A. Luther, R. Buyya, R. Ranjan, and S. Venugopal. Alchemi: A .NET-based grid computing framework and its integration into global grids. Technical Report GRIDS-TR-2003-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Dec. 2003. Comments:12 figures, 2 tables.

[109] U. Maheshwari. Charge-based proportional scheduling. Technical report, M.I.T. Laboratory for Computer Science, 1995.

[110] R. Malone, T.W. Fikes and M. Howard. Enterprise: A market-like task scheduler for distributed computing environments. pages 177–205, 1988.

[111] M. S. Miller and K. E. Drexler. Markets and computation: agoric open systems. In B. A. Huberman, editor, *The Ecology of Computation*, pages 133–176. North-Holland Publishing Company, Amsterdam, 1988.

[112] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, pages 114–117, 19Apr. 1965. Director of R&D Labs at Fairchild Semiconductor http://www.intel.com/intel/annual96/bio_moor.htm.

[113] J. J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications.* Prentice Hall, 2nd edition edition, 1999.

[114] R. B. Myerson. *Game Theory: Analysis of Conflict.* Harvard University Press, 1997.

[115] R. B. Myerson. Nash equilibrium and the history of economic theory. *Journal of Economic Literature*, 37(3):1067–1082, September 1999. available at http://ideas.repec.org/a/aea/jeclit/v37y1999i3p1067-1082.html.

[116] J. F. Nash. Equilibrium points in n-person games. *PNAS*, 36(1):48–49, 1950.

[117] C. Ng, P. Buonadonna, B. N. Chun, A. C. Snoeren, and A. Vahdat. Addressing strategic behavior in a deployed microeconomic resource allocator. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 99–104, New York, NY, USA, 2005. ACM Press.

[118] C. Ng, D. C. Parkes, and M. Seltzer. Virtual worlds: fast and strategyproof auctions for dynamic resource allocation. In *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pages 238–239, New York, NY, USA, 2003. ACM Press.

[119] C. Ng, D. C. Parkes, and M. I. Seltzer. Strategyproof computing: Systems infrastructures for self-interested parties. In *1st Workshop on Economics of Peer-to-Peer Systems*. ACM, 2003.

[120] N. Nisan. Bidding and allocation in combinatorial auctions. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[121] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over the internet - the popcorn project. In *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*, page 592, Washington, DC, USA, 1998. IEEE Computer Society.

[122] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing, 2005. HPDC-14.*, July 2005.

[123] P. Padala, C. Harrison, N. Pelfort, E. Jansen, M. P. Frank, and C. Chokkareddy. Ocean: The open computation exchange and arbitration network, a market approach to meta computing. In *Proceedings of the ISPDC'03 (International Symposium on Parallel and Distributed Computing)*, 2003.

[124] D. C. Parkes, R. Cavallo, N. Elprin, A. Juda, S. Lahaie, B. Lubin, L. Michael, J. Shneidman, and H. Sultan. Ice: an iterative combinatorial exchange. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 249–258, New York, NY, USA, 2005. ACM Press.

[125] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64, 2003.

[126] M. Placek and R. Buyya. G-monitor: Gridbus web portal for monitoring and steering application execution on global grids. Technical report, Feb. 05 2003. Comment: Technical Report, Grid Computing and Distributed Systems Lab, Dept. of Computer Science and Software Engineer, The University of Melbourne, Australia.

[127] F. I. Popovici and J. Wilkes. Profitable services in an uncertain world. In *SC'2005 Conference CD*, Seattle, Washington, USA, Nov. 2005. IEEE/ACM SIGARCH.

[128] O. Regev. Economic issues in cpu sharing system for the internet. Master's thesis, Institute of Computer Science, Hebrew University, Jerusalem, 1998.

[129] O. Regev and N. Nisan. The popcorn market - an online market for computational resources. In *ICE '98: Proceedings of the first international conference on Information and computation economies*, pages 148–157, New York, NY, USA, 1998. ACM Press.

[130] S. Rosen. Hedonic prices and implicit markets: Product differentiation in pure competition. *Journal of Political Economy*, 82(1):34, 1974.

[131] J. Sairamesh, D. F. Ferguson, C. Nikolaou, and Y. Yemini. Economic framework for pricing and charging in digital libraries. *D-Lib Magazine*, 1996.

[132] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 69–76, New York, NY, USA, 2002. ACM Press.

[133] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya. Libra: a computational economy-based job scheduling system for clusters. *Software, Practice and Experience*, 34(6):573–590, May 2004.

[134] J. Shneidman, C. Ng, D. C. Parkes, A. AuYoung, A. C. Snoeren, A. Vahdat, and B. Chun. Why markets could (but don't currently) solve resource allocation problems in systems. In *10th USENIX Workshop on Hot Topics in Operating Systems*, 2005.

[135] W. Smith, V. Taylor, and I. I Foster. Scheduling with advanced reservations. *Proceedings of the IPPS/SPDP '99 Workshop on Job Scheduling Strategies for Parallel Processing*, 1999.

[136] I. Stoica, H. Abdel-Wahab, and A. Pothen. A Microeconomic Scheduler for Parallel Computers. In *Proceedings of the IPPS '95 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–135, April 1995.

[137] M. Stonebraker, P. M. Aoki, R. Devine, W. Litwin, and M. A. Olson. Mariposa: A new architecture for distributed data. In *Proceedings of the Tenth International Conference on Data Engineering*, pages 54–65, Washington, DC, USA, 1994. IEEE Computer Society.

[138] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. An economic paradigm for query processing and data migration in mariposa. In *PDIS '94: Proceedings of the third international conference on on Parallel and distributed information systems*, pages 58–68, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[139] N. Stratford and R. Mortier. An economic approach to adaptive resource management. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems (HotOS-VII)*, 1999.

[140] I. E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, 1968.

[141] K. Tamilmani, V. Pai, and A. Mohr. Swift: A system with incentives for trading. In *Proceedings of Second Workshop on the Economics of Peer-to-Peer Systems*, June 2005.

[142] D. Thain, T. Tannenbaum, and M. Livny. Condor and the grid. *Grid Computing: Making the Global Infrastructure a Reality*, 2003.

[143] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman. Open grid services infrastructure (ogsi) v1.0. 2003.

[144] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource overbooking and application profiling in shared hosting platforms. *SIGOPS Oper. Syst. Rev.*, 36(SI):239–254, 2002.

[145] S. Venugopal, R. Buyya, and L. Winton. A grid service broker for scheduling distributed data-oriented applications on global grids. In *Proceedings of the 2nd International Workshop on Middleware for Grid Computing*, May 06 2004. Comment: 15 pages, 11 figures, 1 table.

[146] V. Vishnumurthy, S. Chandrakumar, and E. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Proceedings of the First Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[147] R. E. Waldron. *Futures 101 : An Introduction to Commodity Trading*. Squantum Publishing Company, 2003.

[148] R. E. Waldron. *Futures 101 : An Introduction to Commodity Trading*. Squantum Publishing Company, 2nd edition edition, 2003.

[149] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and S. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, Feb. 1992.

[150] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the First Symposium on Operating System Design and Implementation*, November 1994.

[151] L. Walras. *Elements d'economie politique pure; ou, Theorie de la richesse sociale*. Lausanne, Paris, 1874.

[152] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 proxy certificates for dynamic delegation. *3rd Annual PKI R&D Workshop*, 2004.

[153] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. In *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*. IEEE Press, 2003.

[154] V. Welch and T. G. S. Team. Globus toolkit version 4 grid security infrastructure: A standards perspective. Technical report, 2005.

[155] M. Wellman, W. Walsh, P. Wurman, and J. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[156] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[157] J. Wilkes, P. Goldsack, G. Janakiraman, L. Russell, S. Singhal, and A. Thomas. Eos-the dawn of the resource economy. *hotos*, 00:0188, 2001.

[158] R. Wolski. Dynamically forecasting network performance using the network weather service, 1999.

[159] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):41–49, 3 2003.

[160] R. Wolski, J. Brevik, J. S. Plank, and T. Bryan. Grid resource allocation and control using computational economies. In F. Berman, G. Fox, and A. Hey, editors, *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, 2003.

[161] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. Analyzing market-based resource allocation strategies for the computational grid. *Int. J. High Perform. Comput. Appl.*, 15(3):258–281, 2001.

[162] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 46, Washington, DC, USA, 2001. IEEE Computer Society.

[163] C. S. Yeo and R. Buyya. Pricing for utility-driven resource management and allocation in clusters. In *Proceedings of the 12th International Conference on Advanced Computing and Communication*, Ahmedabad, India, December 2004.

[164] J. Yu, S. Venugopal, and R. Buyya. Grid market directory: A web services based grid service publication directory. Technical report, Feb. 05 2003. Comment: Technical Report, Grid Computing and Distributed Systems Lab, University of Melbourne, Jan 2003.

[165] X. Zhang and J. M. Schopf. Performance analysis of the globus toolkit monitoring and discovery service, MDS2, 2004.

Appendix A

Section 4.4 Additional Figures

In this first appendix are ratios and indices additional graphs for the default simulation run of section 4.4.

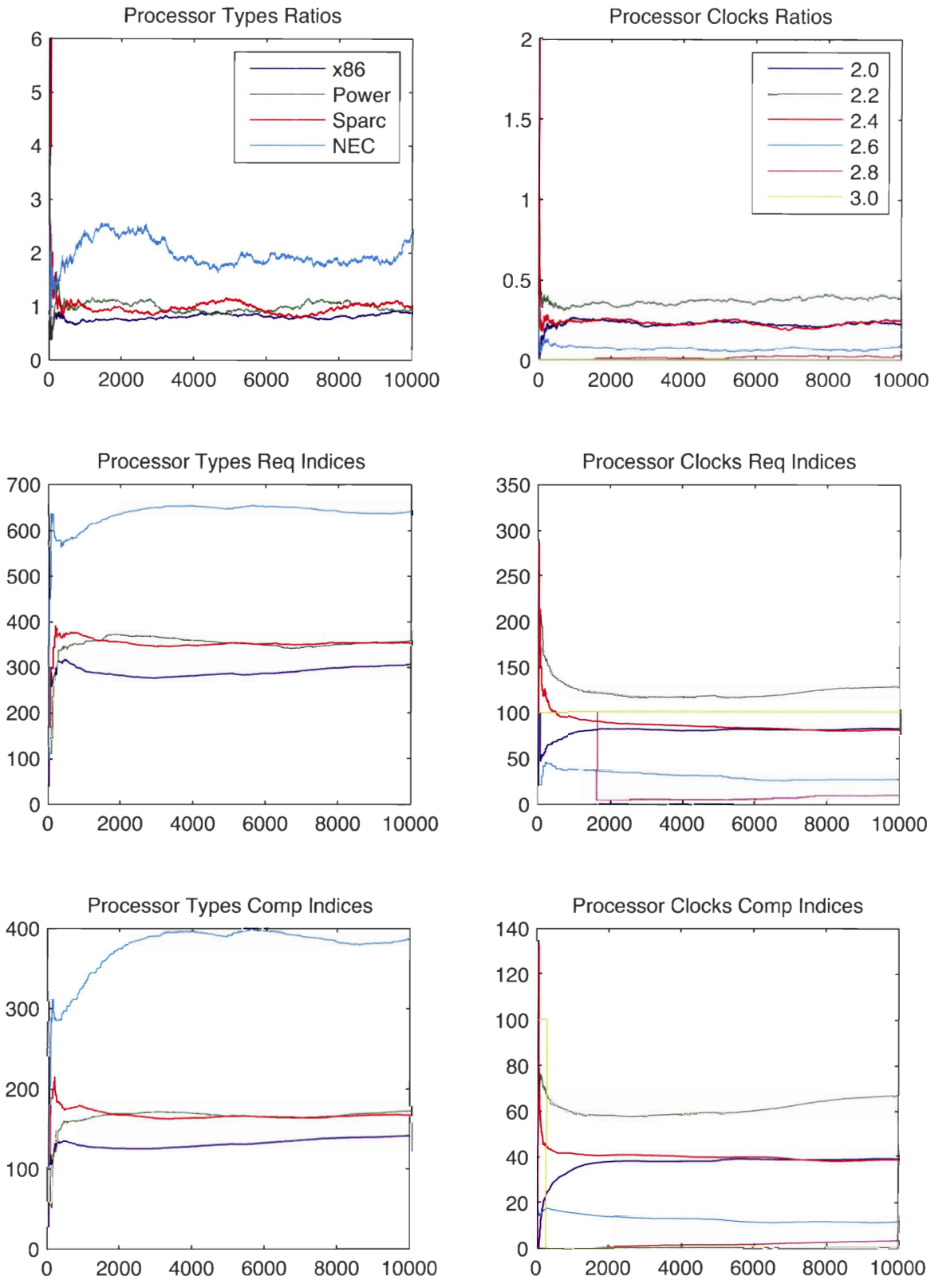

Figure A.1: *Processor types (left column) and clocks (right column) ratios and indices*

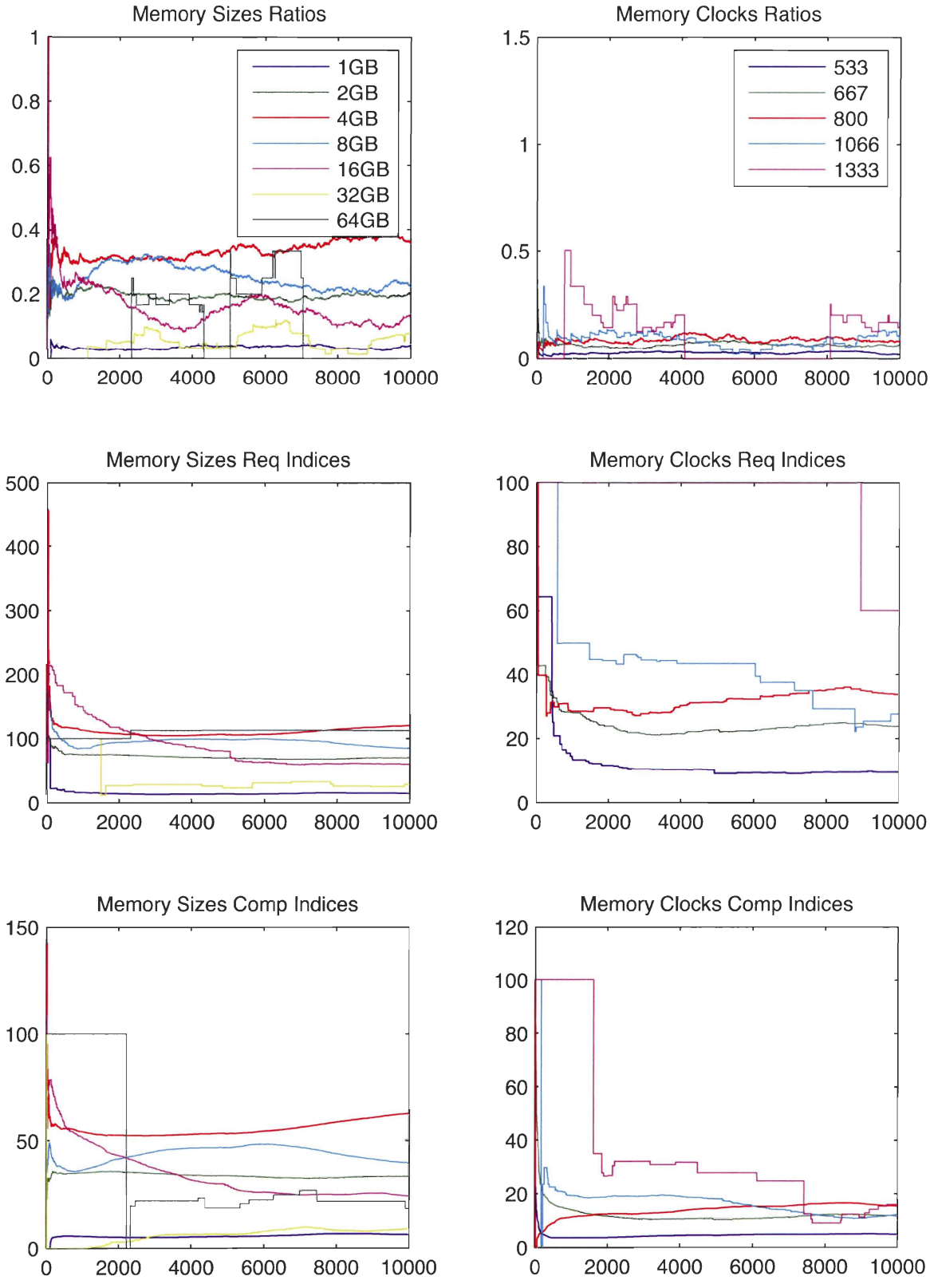

Figure A.2: *Memory sizes (left column) and clocks (right column) ratios and indices*

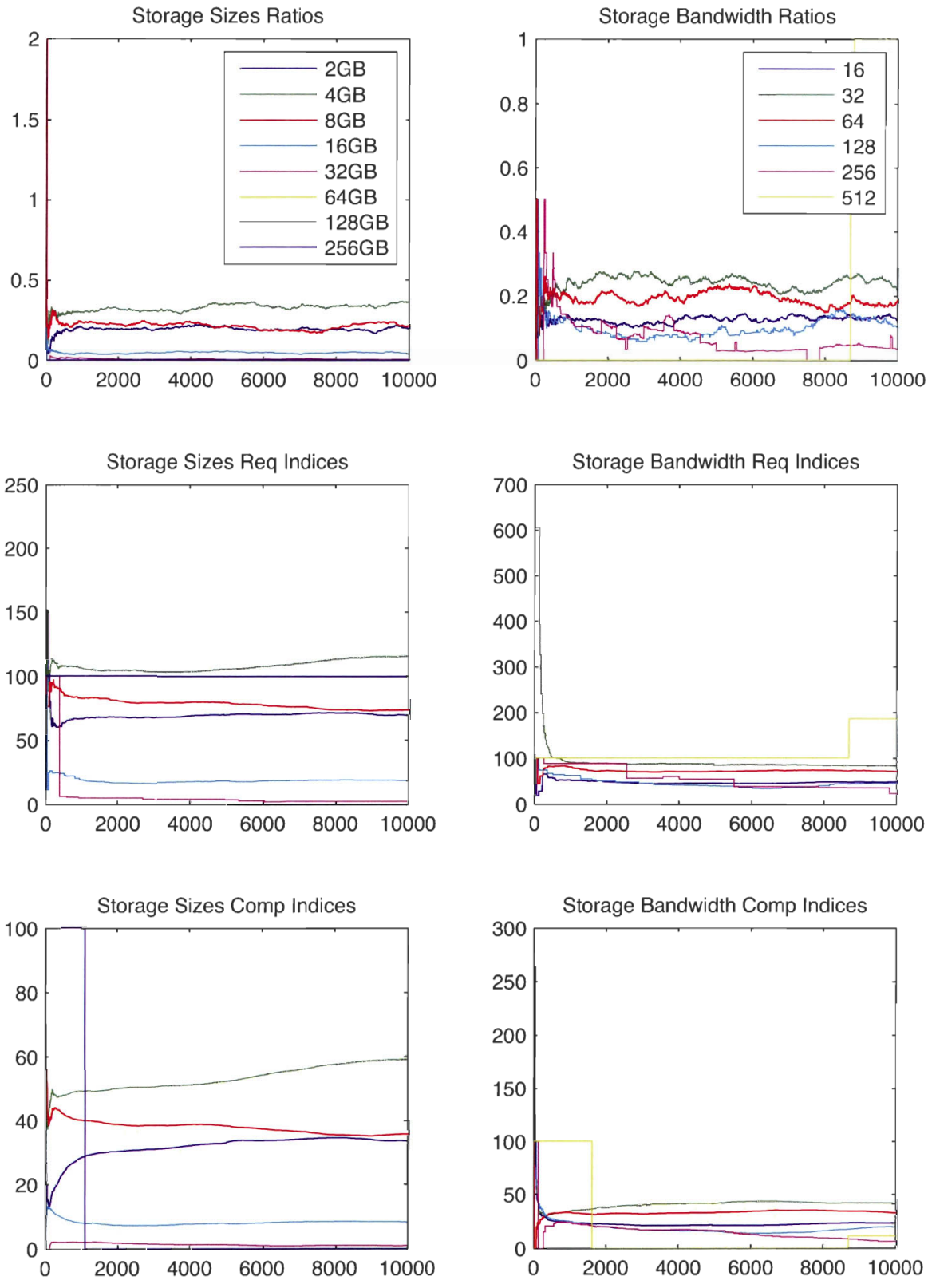

Figure A.3: *Storage sizes (left column) and bandwidth (right column) ratios and indices*

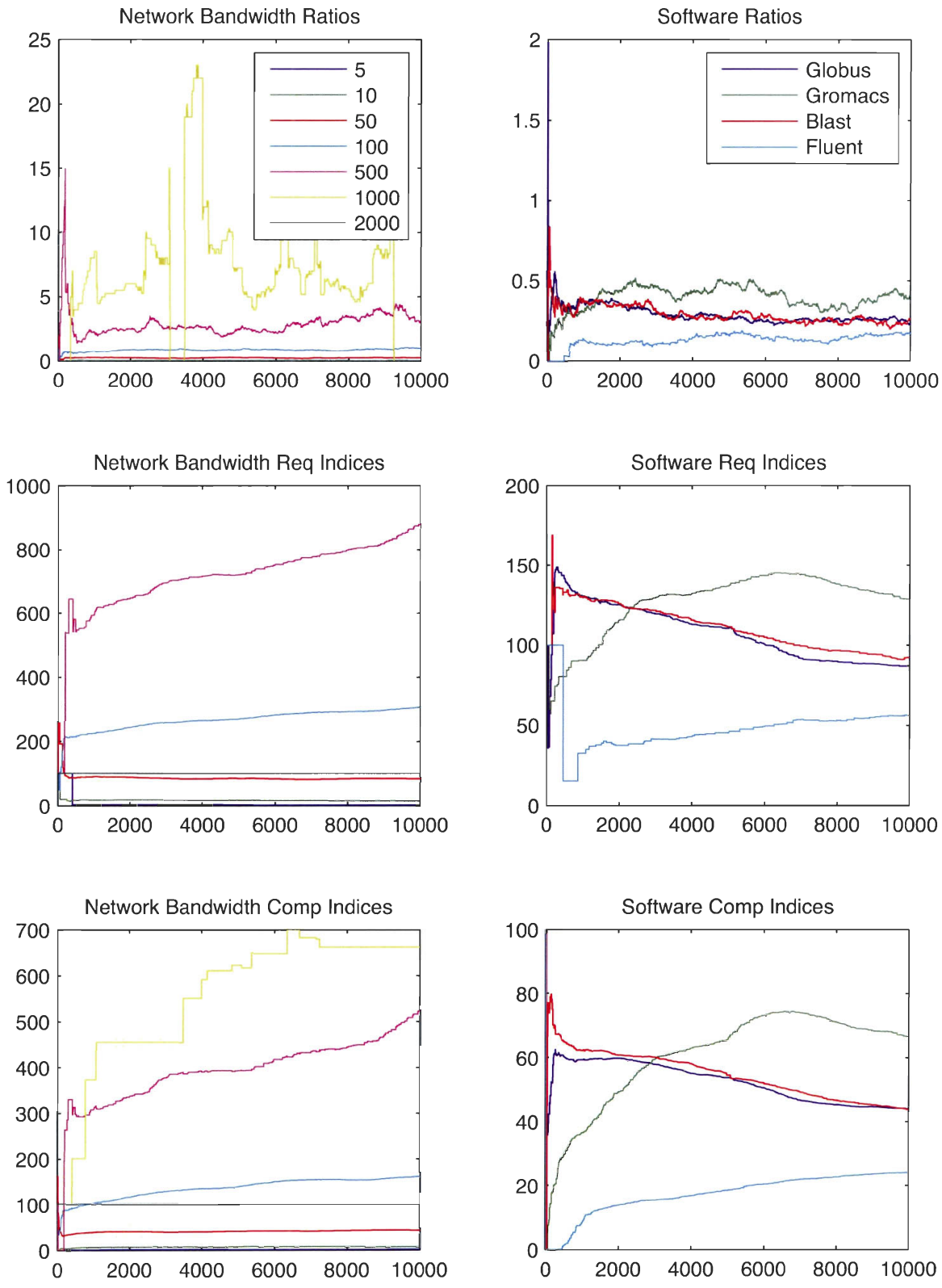

Figure A.4: *Network bandwidth (left column) and software (right column) ratios and indices*

Appendix B

Section 4.6 Additional Figures

In this appendix are included additional figures for section 4.6, where tolerances were initialized at 200%, then lowered to 125% (500th event) and then lowered again to 110% (2000th event).

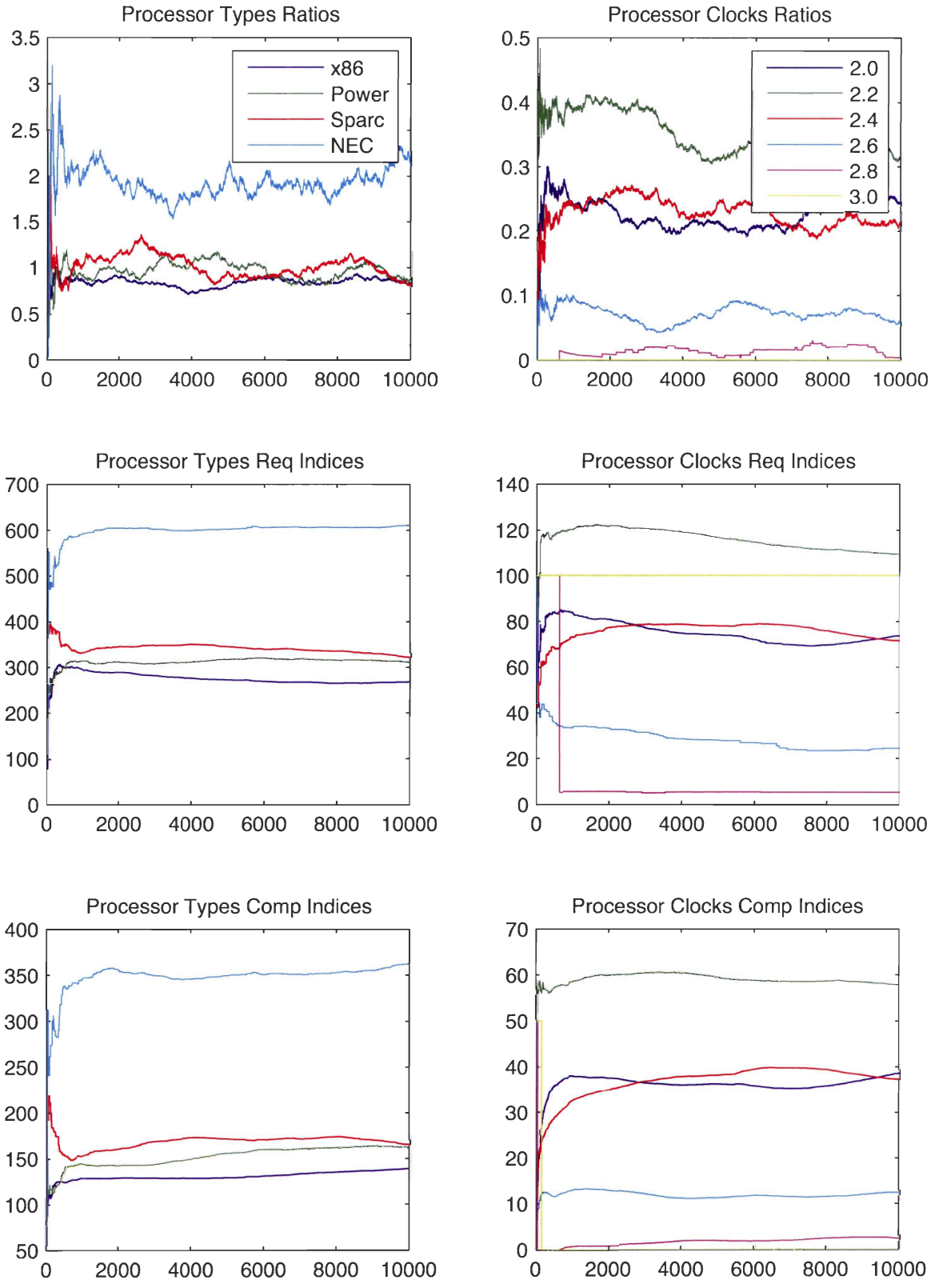

Figure B.1: *Processor types (left column) and clocks (right column) ratios and indices*

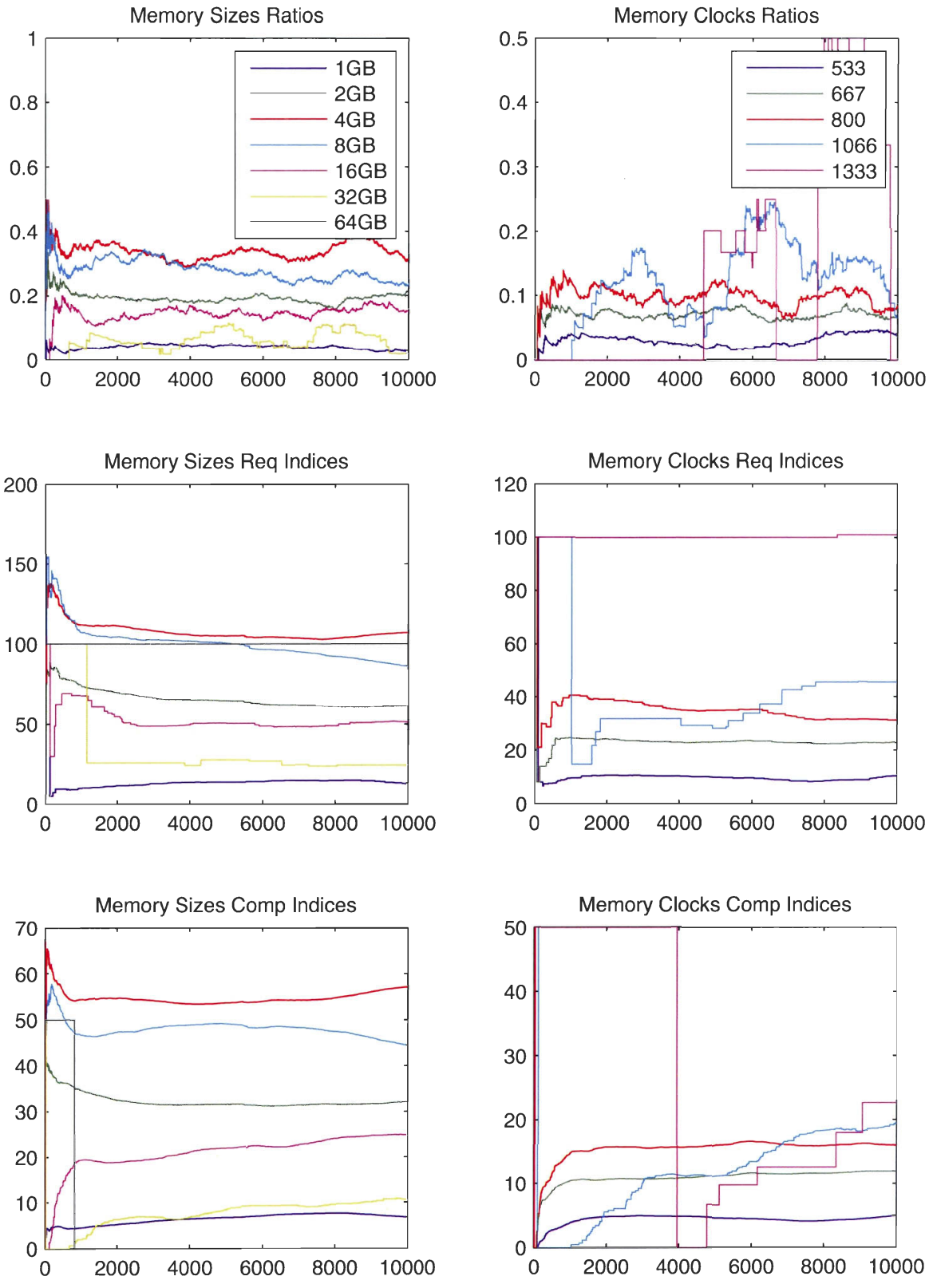

Figure B.2: *Memory sizes (left column) and clocks (right column) ratios and indices*

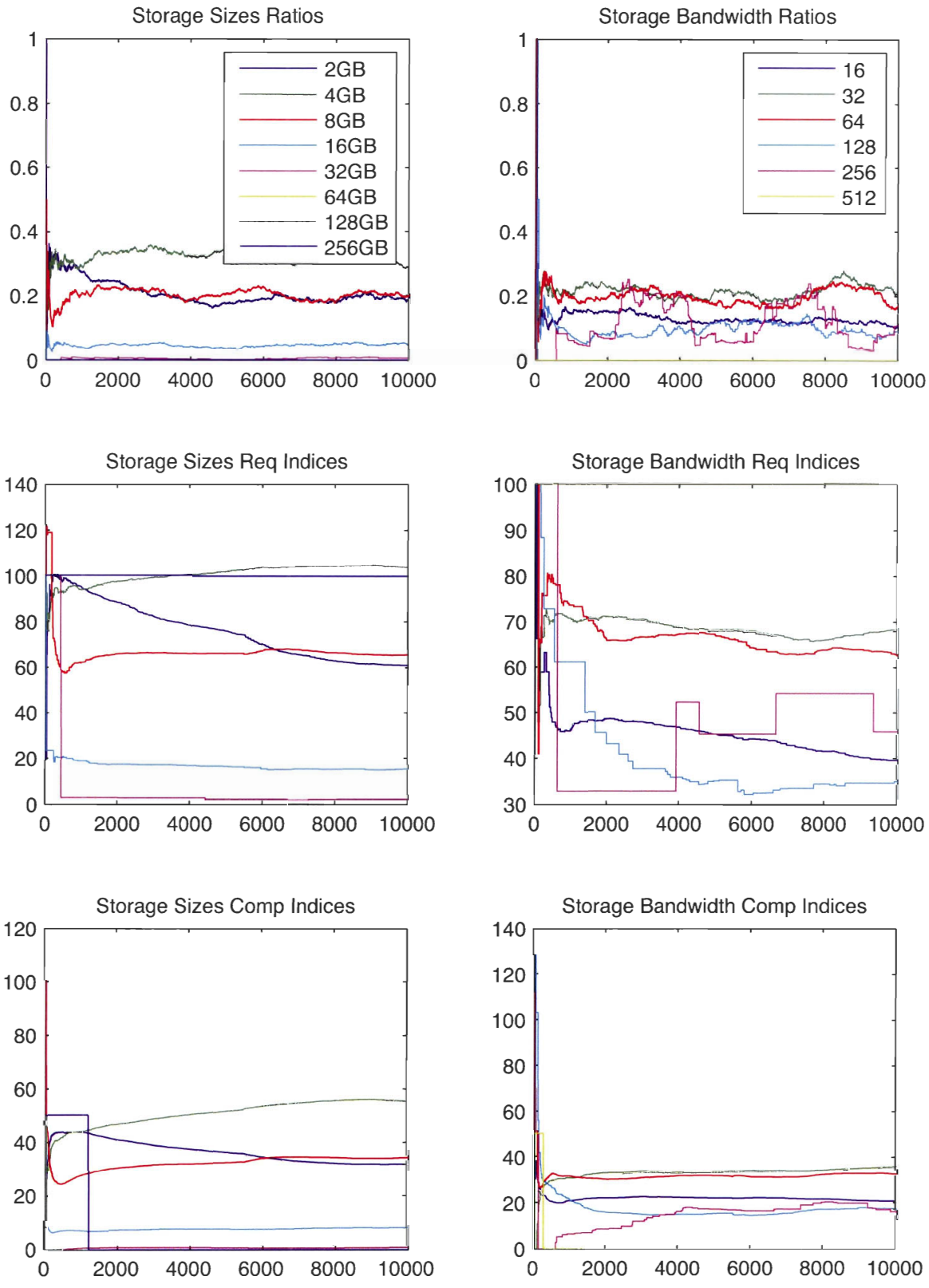

Figure B.3: *Storage sizes (left column) and bandwidth (right column) ratios and indices*

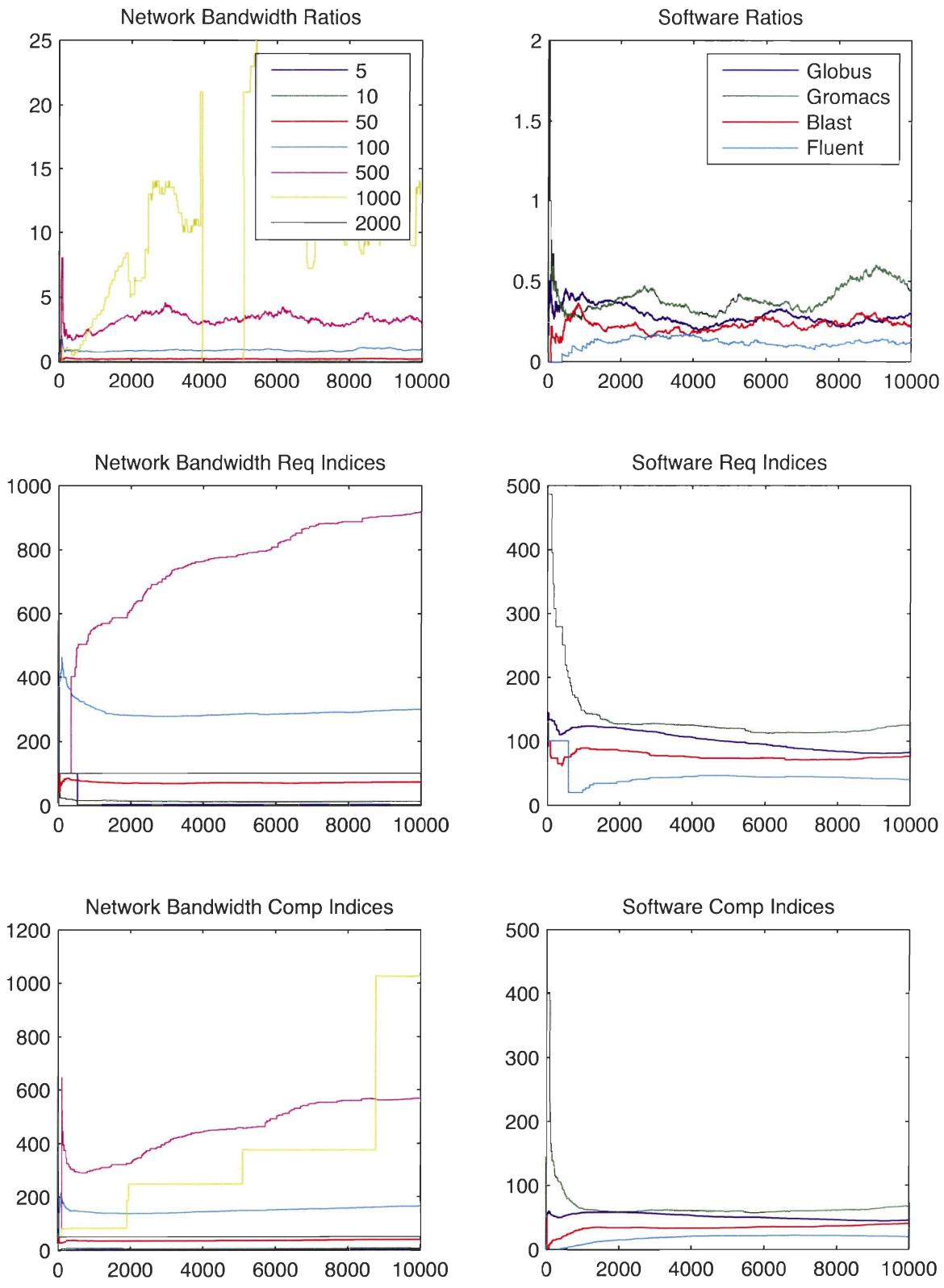

Figure B.4: *Network bandwidth (left column) and software (right column) ratios and indices*

Appendix C

Section 4.8 Additional Figures

Following changing processor architectures market conditions after the 2000th event, all other 7 resources types graphs are included here to demonstrate the system's stability, as other resources ratios and indices remained steady.

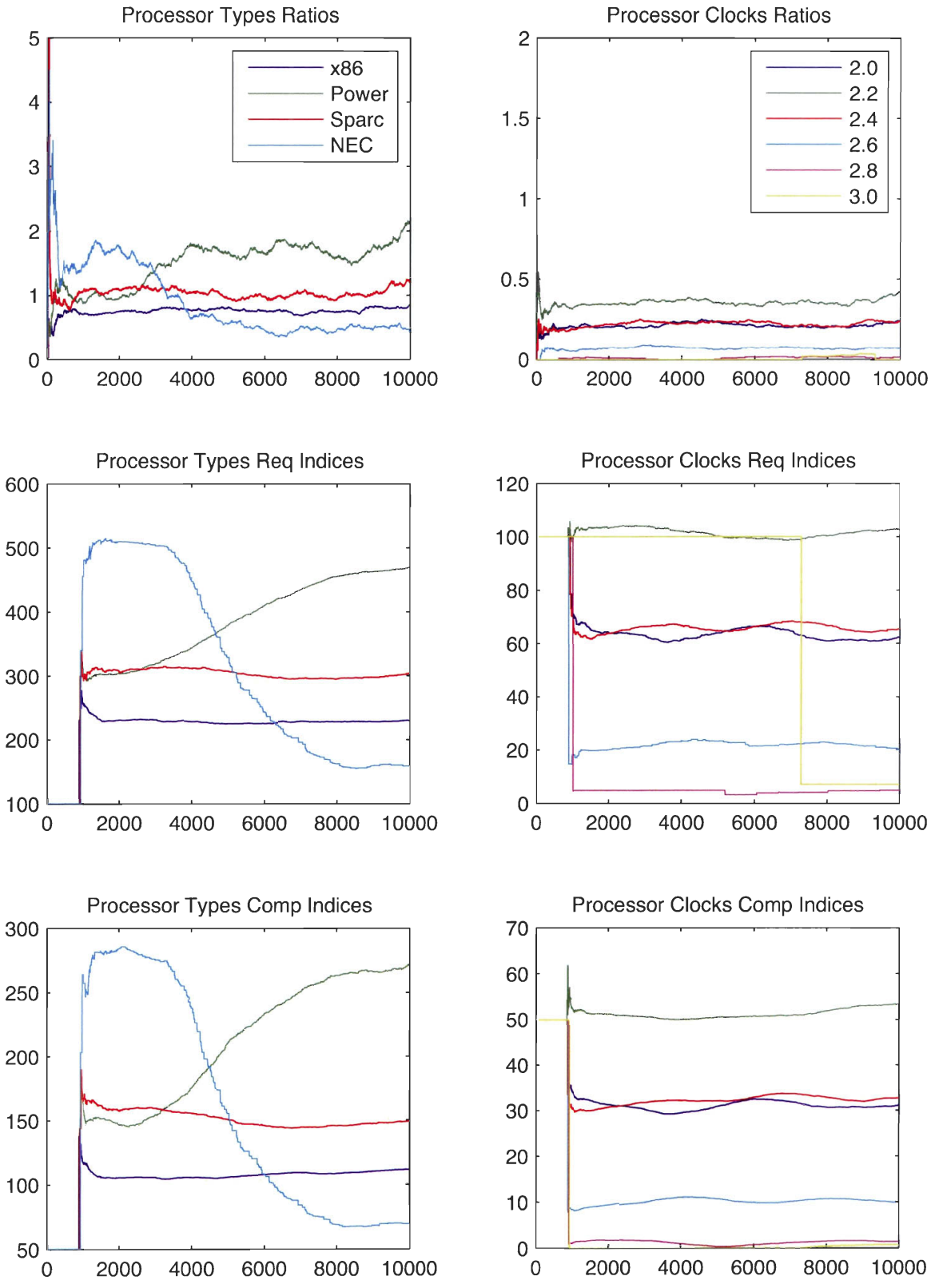

Figure C.1: *Processor types (left column) and clocks (right column) ratios and indices*

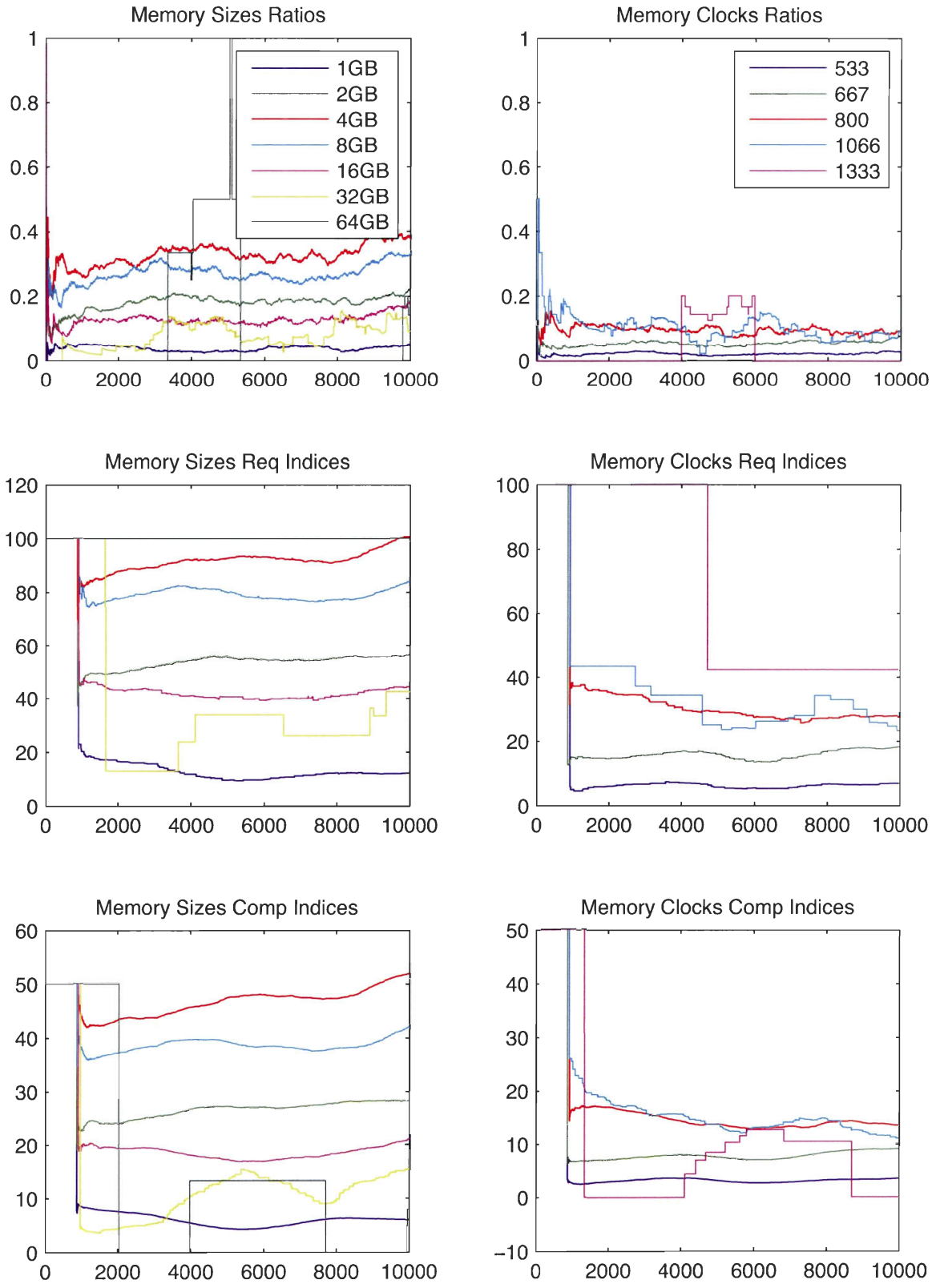

Figure C.2: *Memory sizes (left column) and clocks (right column) ratios and indices*

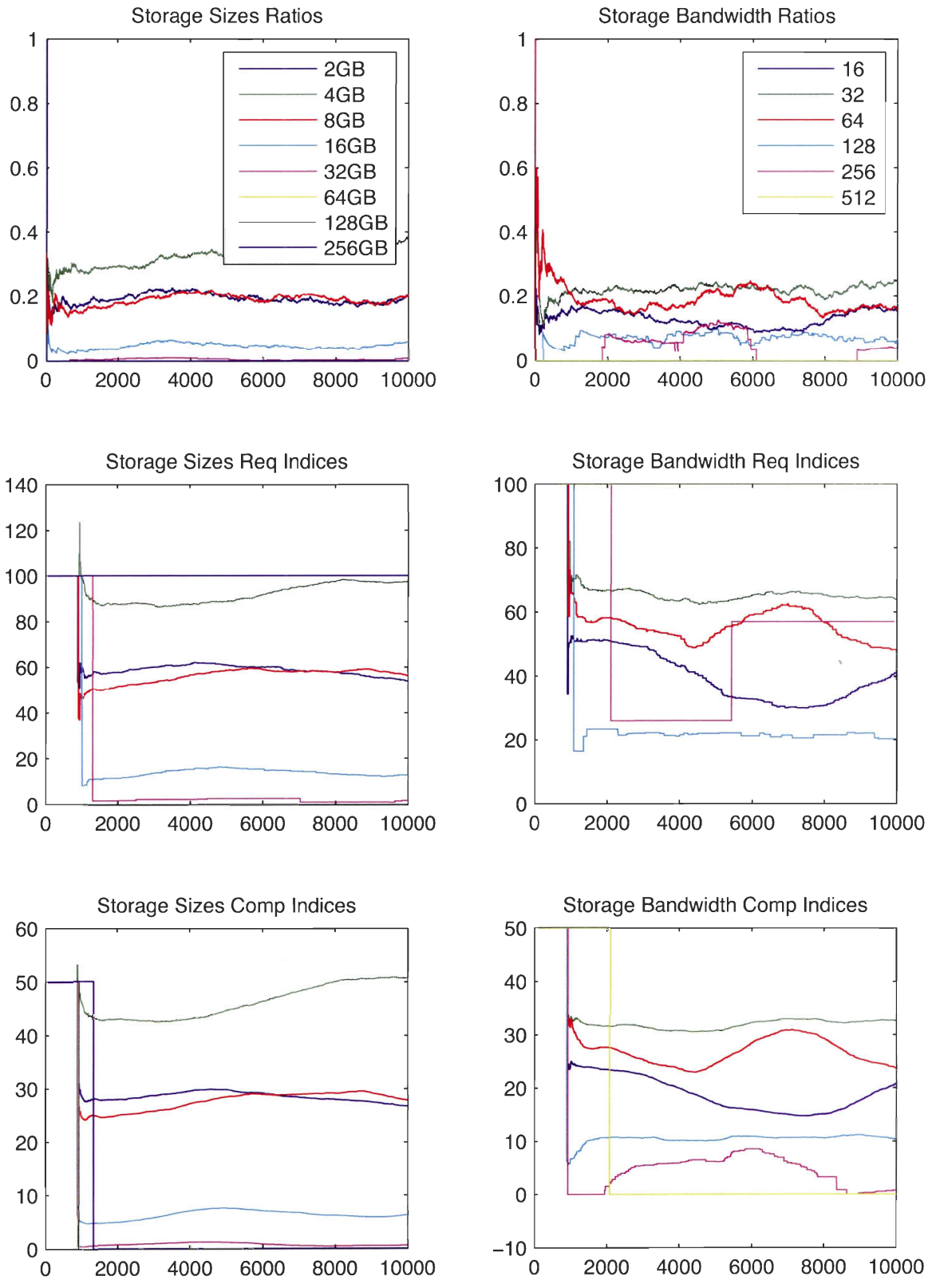

Figure C.3: *Storage sizes (left column) and bandwidth (right column) ratios and indices*

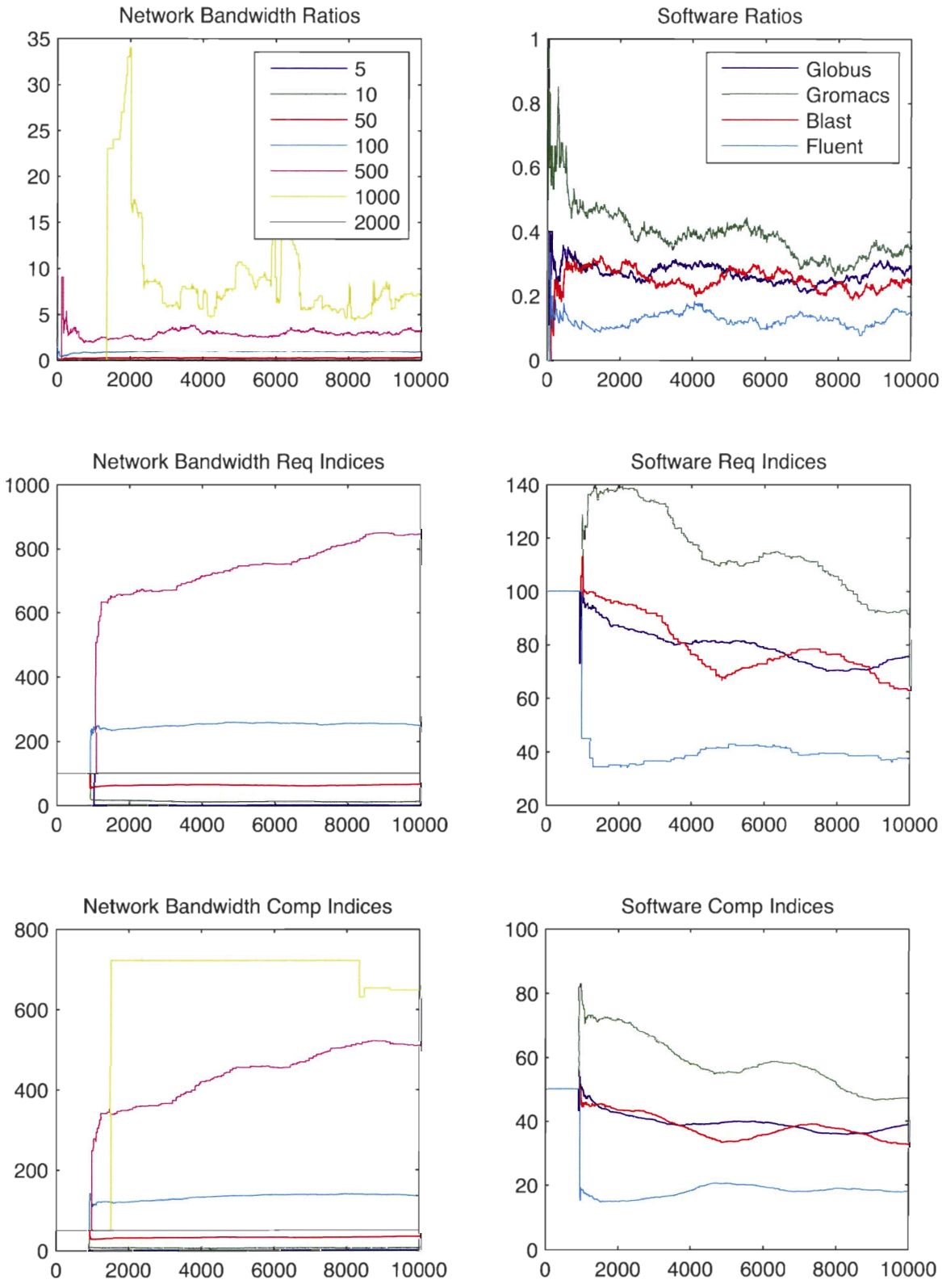

Figure C.4: *Network bandwidth (left column) and software (right column) ratios and indices*