A Market-Based Scheduler for JXTA-Based Peer-to-Peer Computing System

Tan Tien Ping[1], Gian Chand Sodhy[1], Chan Huah Yong[1], Fazilah Haron[1] and Rajkumar Buyya[2]

[1]School of Computer Science
Universiti Sains Malaysia
11800 Penang, Malaysia
{tienping, sodhy, hychan, fazilah} @cs.usm.my
[2]Grid Computing and Distributed Systems Laboratory
Department of Computer Science & Software Engineering
University of Melbourne, Australia
raj@cs.mu.oz.au

Abstract. Peer-to-Peer (P2P) computing is said to be the next wave of computing after client-server and web-based computing. It provides an opportunity to harness a lot of idle peer-resources such as desktop computers across the Internet, for solving large-scale computing applications. Each peer is autonomous and it needs incentive for sustained contribution of its resources to P2P applications. In addition, a flexible and efficient job scheduling is needed to harvest the idle computing power as cheaply and economically as possible.
This paper introduces an economic based job scheduler for mapping jobs to resources in P2P computing environment. The scheduler has been implemented with the Compute Power Market (CPM) system developed using Sun JXTA P2P technology. Our scheduler can be configured depending on users' quality of service requirements such as the deadline and budget constraints. Our scheduler follows a hierarchy scheme. The design allows multiple consumers and multiple providers to schedule and run jobs. To allow wider support for a wide variety of applications, the system is designed to allow easy 'plug in' of user applications.

1 Introduction

P2P computing has been touted as the next wave of computing after client-server and web-based computing [1][22]. The P2P computing paradigm harnesses resources such as storage, computing cycles, contents and human presence that are available at the edge of the Internet. An advantage of P2P computing model is that everyone can contribute their resources while being autonomous. Another advantage of P2P is that a lot of otherwise unused resources can be harvested for the development of science, engineering, and business. Different types of P2P systems have been developed to support file sharing, distributed computing, collaboration, searching, instant messaging and mobile devices. Example systems and applications [22] include Napster, ICQ, Jabber, Gnutella, FreeNet and SETI@Home [14].

A. Laganà et al. (Eds.): ICCSA 2004, LNCS 3046, pp. 147–157, 2004.

P2P computing systems aim to exploit the synergies that result from co-operation of autonomous peers. For this cooperation to be sustainable, peers (resource contributors) need an incentive [8][12][15]. Efforts such as SETI@Home [6] are successful in attracting a large number of contributors due to (a) their exciting application theme—search for an extraterrestrial intelligence— and (b) their objective of sharing results with the public. There is a huge potential in creating and transforming P2P networks into a computing marketplace that brings together providers and consumers. In such marketplace, peers (providers) gain economic incentive by providing access to their resources; and they also get encouraged to offer value-added services. The consumers benefit by gaining access to large-scale resources on demand and having an ability to select resources by based on their quality of service requirements.

In [18], we proposed a market-based resource management and job scheduling system, called Compute Power Market (CPM), for P2P computing on Internet-wide computational resources as market-based systems offer economic incentive for resource providers and also support the regulation of supply-and-demand for resources [8][11][15]. The CPM primarily comprises of markets, resource consumers, resource providers and their interactions. Over the last three years [19], our team has carried out an implementation of CPM using Sun's JXTA [16] P2P computing framework. It supports various economic models for resource trading and matching service consumers and providers. The CPM components that represent markets, consumers and providers are Market Server, Market Resource Agent, and Market Resource Broker (MRB).

This paper focuses on a market-based scheduler implemented as a component of CPM's MRB. The broker is responsible for providing master-worker style application scheduling services by discovering and selecting suitable resources that match user requirements. The CPM scheduler adopts scheduling algorithms, originally developed for Grid environments [13], and incorporates them within P2P computing-based CPM system.

2 Related Work

Although job scheduling on parallel and distributed systems has been investigated extensively in the past, they are limited to cooperative and dedicated environments or system-centric in nature. Moreover, scheduling in different environments such as P2P often involves different challenges and policies, for instance variation of resource availability with time and the presence of large-scale heterogeneity. A number of projects have investigated scheduling of computations on Internet-based distributed systems. They include AMWAT [4], Condor [20], XtremWeb [23], Entropia [24], AppLes[5], Nimrod-G[11], Javelin++ [10], and Java Market [8].

3 CPM and JXTA P2P Framework

JXTA is a open source project initiated by Sun Microsystems. One of its main objectives is to develop a standard protocol that all P2P applications can utilize, since there are a lot of P2P applications which cannot operate, interact and utilize each other's service. Worldwide developers have contributed to new services, for example file sharing chatting and others.

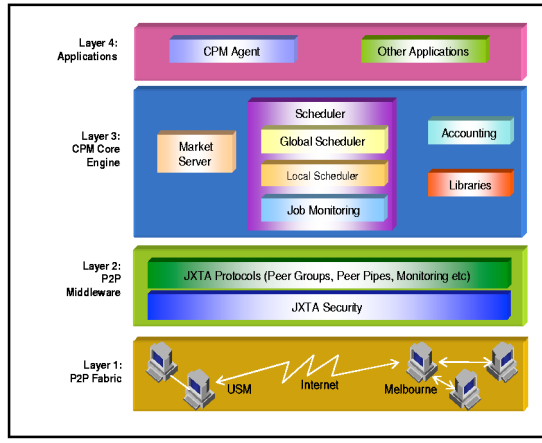

Fig. 1. CPM/P2P Framework.

Compute Power Market (CPM) is a market-based resource management service developed under JXTA. Through CPM, consumer and resource owner can trade computational resources over a P2P network. Figure 1 shows the components of CPM organized into layers. Layer 1 connects geographically distributed compute devices through JXTA network across the Internet. Layer 2 is the middleware comprising JXTA protocols and security. The Core Engine layer contains the main CPM modules, while the application layer contains programs that utilize the functionality of CPM modules.

The scheduler interacts with other modules in CPM to achieve the overall objective. A global scheduler of CPM Market Resource Broker (MRB) interacts with the CPM Market Agent to retrieve available resources in the market. With this information, the global scheduler can then schedule tasks to selected resources. Local scheduler will then schedule the execution of the tasks. Global scheduler and local scheduler will provide an accounting module and billing information, which will be used to bill the customer accordingly.

4 Scheduler Architecture

The scheduler is one of the main components in CPM. It is responsible for assigning tasks from customer to resource providers. The scheduler will be operating in a market environment, where it schedules tasks based on deadline and budget. The type of jobs which can take advantage of our scheduler are those that can be partitioned into smaller tasks and executed in an independent manner such as Monte Carlo simulation, image processing applications, molecular docking and others.

In general, schedulers can be categorized according to their scope. They are the global scheduler and the local scheduler. The global scheduler is also known as macro-scheduler, while local scheduler is also called a micro-scheduler [3]. The global scheduler decides the resource to which the user job is to be allocated based on a certain global scheduling policy, but local scheduler chooses the job to be run on the local system based on the local system policy.

Schedulers can be categorized according to their architectural model. The three commonly used models for organisation and structuring of schedulers are centralized, hierarchical, and distributed models [2][21]. The CPM scheduler follows a hierarchical model. The main components of a CPM scheduler are global scheduler, job monitoring, resource discovery and local scheduler.

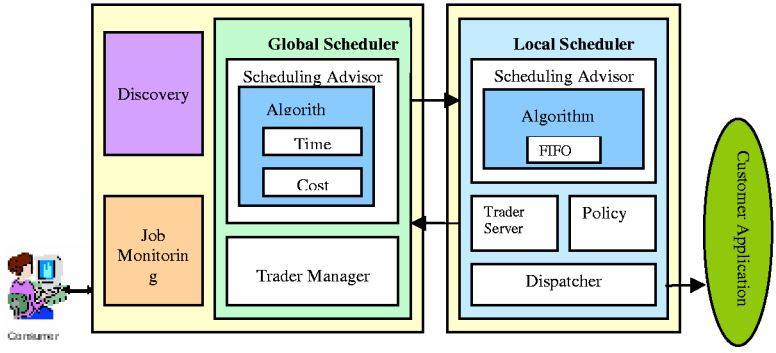

Fig. 2. CPM scheduler architecture.

4.1 Scheduler Components

Global scheduler is responsible for the scheduling of a customer's job. The components of a global scheduler are:

- Scheduling Advisor – Schedule tasks to local scheduler based on a specific scheduling algorithm.
- Scheduling Algorithm – Two types of heuristics scheduling algorithms are supported. They are Cost Optimized and Time Optimized scheduling algorithms [13]. Cost Optimized algorithm schedules tasks for customer where their main concern is cost, while making sure that deadline is reached. Time Optimized

scheduling tries to complete the tasks as soon as possible and within the specified cost.

- Trader Manager – Negotiate price with Trader Server at local scheduler for an agreeable price.

Job Monitoring module monitors the status of tasks and triggers the global scheduler to reschedule if failure occurs. This includes tasks which do not execute or complete within expected period of time or tasks which have failed. The discovery component is used for searching available resource providers over the network.

The CPM local scheduler can accept tasks from more than one global scheduler but only one task is executed at a time. The components of a local scheduler are:

- Scheduling Advisor – It accepts or rejects a given task based on resource provider's specified scheduling policies. It schedules the execution of tasks based on local scheduling algorithm.
- Scheduling algorithm – Support scheduling based on first-in-first-out (FIFO) method.
- Trader Server – Negotiate price with Trader Manager.
- Policy – User can specify the scheduling policy which decides what tasks get accepted and what task does not. Three types of policies are supported: minimum price of the total given tasks, total maximum time length acceptable by resource provider and availability.
- Dispatcher – Dispatches customer's application for execution. Two types of application are supported currently. They are java applications and Windows executable programs.

4.2 Scheduling Activities

Scheduling can be divided into 5 steps. The steps involved are:

i) Resource discovery, where resources are searched for and found.
ii) Resource trading, which involves retrieving the necessary information for selecting and scheduling tasks at the next stage.
iii) Scheduling, where tasks are matched to resources, using a scheduling algorithm.
iv) Execute and monitor the task on resources used.
v) Rescheduling: It handles reassignment of failed tasks or schedule variation to due to the change in the availability of resources.

4.2.1 Resource Discovery

Before a customer can schedule tasks to peers, resources need to be discovered. In JXTA, resources, services and others are represented as advertisements. The CPM resource advertisements contain information such as price, speed of computation and others, which are published by resource providers. There are 3 ways to discover advertisements, either through local cache, direct discovery or indirectly through a rendezvous peer [16]. A Rendezvous peer is a special peer, which provides the service of discovering other peers or resources.

4.2.2 Resource Trading

The trading module in the global scheduler and the local scheduler is used for negotiating agreeable price. In the current implementation, there is not much interaction involved. We use a flat-price market model whereby providers advertise their resources using resource advertisements. Global scheduler that discovers the resource advertisements will then decide which peer to hire depending on a few criteria, such as cost, speed and completion time. The information of cost and speed can be retrieved from the resource advertisement. However for the completion time, the global scheduler needs to query the local schedulers. Communication between global scheduler and local scheduler is done via JXTA pipe service. The benefit of using pipes for communication is that it hides the need to know each other's IP address and port number, using only peer IDs. Besides that, the communication can cross firewalls. These are important aspects, especially in P2P environment.

An "order ID" is also being returned when a global scheduler queries for the completion time at a local scheduler. The order ID is used to resolve contention when two or more global schedulers are interested in a peer at the same time. When a global scheduler wants to make a purchase, it needs to supply the order ID, which will change when an order is successfully made.

4.2.3 Scheduling

The global scheduler allocates tasks to a local scheduler through a contract. A contract is an agreement between the global scheduler and the local scheduler on the award of one or more tasks to it. Each task can be different in terms of functionality, runtime, input files, etc. However, they must not have any dependencies among them. A contract received by a local scheduler is put into a queue, and processed in FIFO order. A task from the contract will be processed. The required files, such as application and data files, will be fetched when the task is about to run.

A resource is selected based on the resource's price versus customer's budget and resource's completion time (next available time) versus customer's deadline. We provide two scheduling algorithms for a user to choose, depending on their priority in cost and speed [12]. In cost optimization scheduling, the scheduler will try to schedule as many tasks as possible to the cheapest provider as long as deadlines are not exceeded. As for time optimization, the scheduler will schedule to make sure tasks are completed as soon as possible, taking care that budget is not exceeded.

4.2.4 Execution and Monitoring

When a task is ready for execution, the local scheduler will request the necessary data or files from the global scheduler. After the required data is available, the task can be executed. The module that is responsible for the execution of customer applications is the Dispatcher. The dispatcher learns of the type of application and then executes the right command to run the application. The current implementation supports 2 types of applications, Java programs and Windows executable programs.

Job monitoring module is responsible for monitoring the status of the tasks. When a task fails to execute or complete within expected period, job monitoring module will abort the task and reschedule it to another peer. To avoid the possibility of a failed

task being moved from one peer to another, user can set the maximum number of times a task can be rescheduled.

4.2.5 Rescheduling

Scheduler will reschedule a task when it is triggered by the job monitoring module. Rescheduling a task is just like scheduling, where it is reassigned to a new provider. When the scheduler cannot find a suitable resource, customers may update their QoS parameters such as budget and/or deadline constraints, so that resources that meet the new criteria can be found.

5 Scheduling Experiments and Results

We have developed a simple image processing application as a simulation. The application is used for segmenting an object from an image using adaptive thresholding method [17].

The objective of the experiments are to see how the scheduler schedules a task under different budget and deadline constraints. We conducted our test in a local area network (LAN) with bandwidth of 100Mbps. Table 1 shows the hardware used We have configured a job consisting of 40 tasks on a Window machine to be scheduled to 4 provider machines (labeled as Win2000, XP1, XP2 and Linux), the specifications of which are shown in Table 1. Each task uses an input image involving an estimated 28141 million operations, which will be processed by an image processing application as shown in Figure 3. All machines are benchmarked using a standard application where the number of floating point operations they can perform in a second is known. The price is set as tokens per hour and the value of tokens can be mapped to a real currency. Each task consists of an image with the same dimension and parameters. We then schedule the tasks out to resources where objects in the image are segmented and returned to the customer.

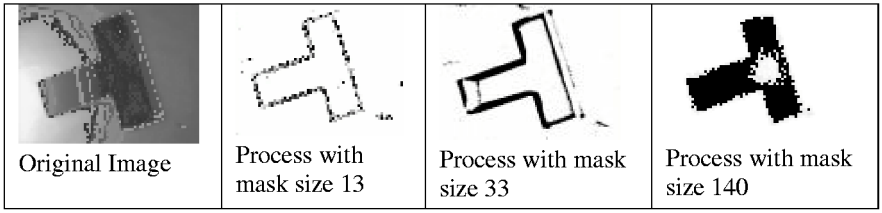

Fig. 3. Processed images with different mask size which have been resized.

In the first experiment, we have allocated 500 tokens with the deadline of 1 hour to perform the job, which consists of 40 tasks. We select Time Optimization as the scheduling algorithm to schedule the jobs. Figure 4 shows the task completion time by the machines. From the graph we can see that all tasks completed before the deadline (i.e. 3600 seconds). Machines XP1 and XP2 have been scheduled the most

Table 1. CPM testbed and hardware configuration

PC	Specification			Description			Label
	OS	Memory (GHz)	RAM (MB)	Role	Benchmark (mflops)	Price (token/hour)	
1	Win2K	1	256	Provider	130	80	Win 2000
2	WinXP	2.4	512	Provider	180	300	XP2
3	WinXP	2.4	512	Provider	180	250	XP1
4	RedHat	0.486	256	Provider	50	40	Linux

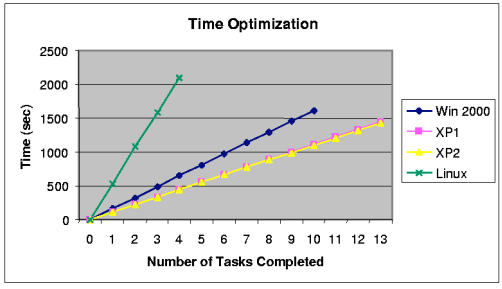

Fig. 4. Graph showing Time Optimization scheduling of 40 tasks. Budget allocated is 500 tokens with time deadline of 1 hour.

tasks (13 tasks), since both have the highest processing speed and the budget allocated is sufficient. Therefore, in cases where budget is more than required, cost becomes a less important constraint, and the dominant factor which determines the completion time is the number of resource providers and their respective processing speeds.

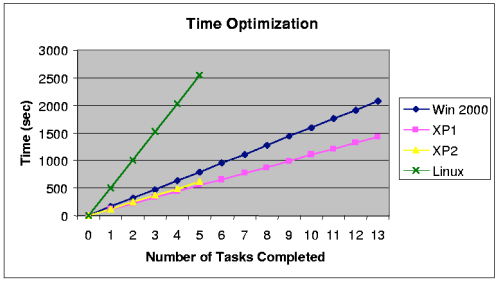

Fig. 5. Graph showing Time Optimization scheduling of 40 tasks. Budget allocated is 350 tokens with time deadline of 1 hour.

In the second experiment, we still use Time Optimization algorithm to schedule the same jobs, but now the budget has been reduced from the previous 500 tokens to 350 tokens. From Figure 5, we can see that reducing the budget will reduce the number of tasks allocated to more expensive machines. In this case the number of tasks allocated to machine XP2 has been reduced to 5. Cheaper but slower machines like Win2000, for instance, get more tasks.

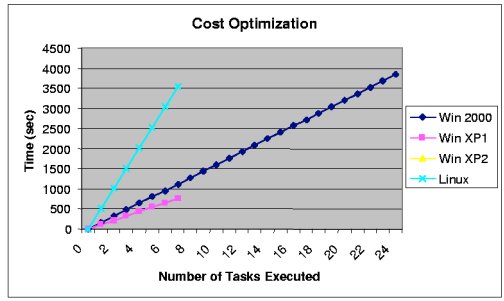

Fig. 6. Graph showing Cost Optimization scheduling of 40 tasks. Budget allocated is 300 tokens with time deadline of 1.5 hour.

For the final experiment, we use the same job again for our test. However, we change our scheduling strategy to Cost Optimization, allocating only 300 tokens, with the duration of 1.5 hours. From Figure 6, we can see that the scheduler utilizes the cheapest resources by allocating as many tasks as possible to them, for instance machines Win2000 and Linux. The most expensive resource, machine XP2, does not get any task at all. Notice that all 40 tasks are completed well before the deadline (of 5400 seconds).

From this experiment, we can deduce that if task execution time can be reasonably estimated, then the Cost Optimization scheduling algorithm provides better results in terms of budget used and completion time (within the deadline given).

6 Conclusion

We discussed a market-based scheduler for JXTA-based P2P computing system. It followed hierarchical model for system architecture. It consists of global scheduler, which is implemented as part of the CPM market-resource broker and local scheduler, which implemented as part of the CPM market resource agent. We discussed in depth the implementation of two market-based global scheduling algorithms within CPM global scheduler along with experimental and performance results.

Acknowledgements. A financial assistance from the Ministry of Science, Technology and Environment of Malaysia (MOSTE) is gratefully acknowledged. We thank Rob Gray (Monash University) and Srikumar Venugopal (University of Melbourne) for their valuable comments on the paper.

References

[1] Clay Shikey, "What is P2P… And What Isn't", http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html, O'Reilly Network. Nov 2000.

[2] Vijay Subramani, Rajkumar Kettimuthu Srividya, Srinivasan P. Sadayappan, "Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests", *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02),* Scotland, 2002.

[3] Steve J. Chapin and Eugene H. Spafford, "Support for Implementing Scheduling Algorithms Using MESSIAHS", *Scientific Programming*, 1994.

[4] Garry Shao, "Adaptive Scheduling of Master/Worker Applications on Distributed Computational Resources", *Ph.D. Thesis*, University of California, San Diego. May 2001.

[5] Francine Berman et. al. "Adaptive Computing on the Grid Using AppLeS", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 4 IEEE Press, USA, Apr 2003.

[6] David Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, Dan Werthimer, "SETI@home: An Experiment in Public-Resource Computing", *Communications of the ACM,* Vol. 45 No. 11, ACM Press, USA, November 2002.

[7] Henri Casanova, Arnaud Legrand, "Heuristics for Scheduling Parameter Sweep Applications in Grid Environment", *Proceedings of the 9th Heterogeneous Computing Workshop (HCW'2000*). Cancun, Mexico, May 2000.

[8] Yair Amir, Baruch Awerbuch, and Ryan S. Borgstrom, "The Java Market: Transforming the Internet into a Metacomputer". *Technical Report CNDS-98-1*, Johns Hopkins University, 1998.

[9] Peter Cappello, Bernd Christiansen, Mihai F. Ionescu, Michael O. Neary, Klaus E. Schauser, and Daniel Wu, "Javelin: Internet-Based Parallel Computing Using Java", *Proceedings of the 1997 ACM Workshop on Java for Science and Engineering Computation*, June 1997

[10] Michael O. Neary, Sean P. Brydon, Paul Kmiec, Sami Rollins, Peter Capello, "Javelin++: Scalability Issues in Global Computing", *Future Generation Computing Systems Journal*, Vol.15(5-6):659-674, Elsevier, Netherlands, 1999.

[11] Rajkumar Buyya, David Abramson, Jonathan Giddy, "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", *Proceedings of 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000)*, Beijing, China, 2000.

[12] Rajkumar Buyya, "Economic-based Distributed Resource Management and Scheduling for Grid Computing", *Ph.D. Thesis*, Monash University Australia, April 2002.

[13] Rajkumar Buyya, Jonathan Giddy, and David Abramson, "An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications", *Proceedings of the 2nd Workshop on Active Middleware Services (AMS 2000),* Kluwer Academic Press, Pittsburgh, USA, August 1, 2000.

[14] W. T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, D. Anderson, "A new major SETI project based on Project Serendip data and 100,000 personal computers", *Proceedings of the 5th International Conference on Bioastronomy*, 1997.

[15] Carl A. Waldspurger, Tad Hogg, Bernado A. Huberman, Jeffrey O. Kephart and Scott Stornetta, "Spawn: A Distributed Computational Economy", *IEEE Transactions on Software Engineering*, IEEE Press, USA, February 1992.

[16] Brendon J. Wilson, *JXTA*, New Riders Publishing, Indiana, June 2002.

[17] Anatol Piotrowski and Sivarama P. Dandamudi, "Performance Sensitivity of Variable Granularity *Proceedings of International Conference on Massively Parallel Computer Systems*, Colorado Springs, April 1998

[18] Rajkumar Buyya and Sudharshan Vazhkudai, "Compute Power Market: Towards a Market-Oriented Grid", *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001),* Brisbane, Australia, May 15-18, 2001.

[19] Rajkumar Buyya Fazilah Haron Chan Huah Yong, "CPM on Jxta", http://compute-power-market.jxta.org/

[20] Matt Mutka and Miron Livny, "Scheduling Remote Processing Capacity In A Workstation-Processing Bank Computing System"*, Proceedings of the 7th International Conference of Distributed Computing Systems,* September 1987.

[21] Rajkumar Buyya, David Abramson, and Jonathan Giddy, "An Economy Driven Resource Management Architecture for Global Computational Power Grids", *Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*, Las Vegas, USA, June 26-29, 2000.

[22] Andy Oram (ed), *Peer-to-Peer Harnessing the Power of Disruptive Technologies*, O'Reilly Press, USA, 2001.

[23] Cecile Germain, Vincent Neri, Gille Fedak and Franck Cappello, "XtremWeb: building an experimental platform for Global Computing", *Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000)*, Bangalore, India, Dec. 2000.

[24] Andrew Chien, Brad Calder, Stephen Elbert, and Karan Bhatia, "Entropia: architecture and performance of an enterprise desktop grid system", *Journal of Parallel and Distributed Computing*, Volume 63, Issue 5, Academic Press, USA, May 2003.