

Architecture of a Commercial Enterprise Desktop Grid: The Entropia System

Andrew A. Chien, achien@cs.ucsd.edu

Entropia, Inc Dept of Computer Science & Engineering
10145 Pacific Heights University of California, San Diego
Suite 800 9500 Gilman Drive
San Diego, CA 92121 La Jolla, CA 92093

Abstract

Distributed Computing, the exploitation of idle cycles on desktop PC systems, offers the opportunity to increase the available computing power by orders of magnitude (10x - 1000x). However, for desktop PC distributed computing to be widely accepted within the enterprise, the systems must achieve high levels of efficiency, robustness, security, scalability, unobtrusiveness, and manageability.

We describe the Entropia Desktop Grid Computing System, detailing its internal architecture and philosophy in attacking these key problems. Key aspects of the Entropia system include the use of: 1) scalable web/database technology for system management, 2) application namespaces for machines, 3) binary sandboxing technology for security and unobtrusiveness, and 4) open integration model to allow applications from many sources to be incorporated. These applications are enabled with nearly zero effort, and achieve good performance. We describe the use of the system both from an applications and an enterprise IT perspective.

The Entropia desktop grid system has been deployed in a wide range of commercial environments and used for a range of applications from areas including bioinformatics, molecular modeling, and monte carlo financial simulations.

Introduction

For over four years, the largest computing systems in the world have been based on “distributed computing”, the assembly of large numbers of PC’s over the Internet. These “grid” systems sustain multiple teraflops continuously by aggregating hundreds of thousands to millions of machines and demonstrate the utility of such resources for solving a surprisingly wide range

of large-scale computational problems in data mining, molecular interaction, financial modeling, etc. These systems have come to be called “distributed computing” systems and leverage the unused capacity of high performance desktop PC’s (up to 2.2 Gigahertz machines with multi-gigaop capabilities[1]), high-speed local-area networks (100 Mbps to 1Gbps switched), large main memories (256MB – 1GB configurations), and large disks (60 to 100 GB disks). Such “distributed computing” or desktop grid systems leverage the installed hardware capability (and work well even with much lower performance PC’s), and thus can achieve a cost per unit computing (or Return-On-Investment) superior to the cheapest hardware alternatives by as much as a factor of five or ten. As a result, distributed computing systems are now gaining increased attention and adoption within the enterprises to solve their largest computing problems and attack new problems of unprecedented scale. For the remainder of the paper, we focus on enterprise desktop grid computing. We use the terms distributed computing, high throughput computing, and desktop grids synonymously to refer to systems that tap vast pools of desktop resources to solve large computing problems – both to meet deadlines or to simply tap large quantities of resources.

For a number of years, a significant element of the research and now commercial computing community has been working on technologies for Grids [2-6]. These systems typically involve servers and desktops, and their fundamental defining feature is to share resources in new ways. In our view, the Entropia system is a desktop Grid which can provide massive quantities of resources and will naturally be integrated with server resources into an enterprise Grid [7, 8].

While the tremendous computing resources available thru distributed computing present new

opportunities, harnessing them in the enterprise is quite challenging. Because distributed computing exploits existing resources, to acquire the most resources, capable systems must thrive in environments of extreme heterogeneity in machine hardware and software configuration, network structure, and individual/network management practice. The existing resources have naturally been installed and designed for purposes other than distributed computing, (e.g. desktop word processing, web information access, spreadsheets, etc.); the resources must be exploited without disturbing their primary use.

To achieve a high degree of utility, distributed computing must capture a large number of valuable applications – it must be easy to put an application on the platform – and secure the application and its data as it executes on the network. And of course, the systems must support large numbers of resources, thousands to millions of computers, to achieve their promise of tremendous power, and do so without requiring armies of IT administrators.

The Entropia system provides solutions to the above desktop distributed computing challenges. The key advantages of the Entropia system are the ease of application integration, and a new model for providing security and unobtrusiveness for the application and client machine. Applications are integrated using binary modification technology without requiring any changes to the source code. This binary integration automatically ensures that the application is unobtrusive, and provides security and protection for both the client machine and the application's data. This makes it easy to port applications to the Entropia system. Other systems require developers to change their source code to use custom APIs or simply provide weaker security and protection[9-11]. In many cases, application source code may not be available and recompiling and debugging with custom APIs can be a significant effort.

The remainder of the paper includes:

- an overview of the history of distributed computing (desktop grids),
- the key technical requirements for a desktop grid platform: efficiency, robustness, security, scalability, manageability, unobtrusiveness, and openness/ease of application integration,
- the Entropia system architecture, including its key elements and how

it addresses the key technical requirements,

- a brief discussion of how applications are developed for the system, and
- an example of how Entropia would be deployed in an enterprise IT environment.

Background

The idea of distributed computing has been described and pursued as long as there have been computers connected by networks. Early justifications of the ARPANET [12] described the sharing of computational resources over the national network as a motivation for building the system. In the mid 1970's, the Ethernet was invented at Xerox PARC, providing high bandwidth, local-area networking. This invention combined with the Alto Workstation presented another opportunity for distributed computing, and the PARC Worm[13] was the result. In the 1980's and early 1990's several academic projects developed distributed computing systems that supported one or several Unix systems[11, 14-17]. Of these, the Condor Project is best known and most widely used. These early distributed computing systems focused on developing efficient algorithms for scheduling, load balancing, and fairness. However, these systems provided no special support for security and unobtrusiveness, particularly in the case of misbehaving applications. Further, they do not manage dynamic desktop environments, limit what is allowed in application execution, and have significant per machine management effort.

In the mid-1980's, the parallel computing community began to leverage first Unix workstations[18] and in the late 1990's low-cost PC hardware[19, 20]. Clusters of inexpensive PCs connected with high-speed interconnects were demonstrated to rival supercomputers. While these systems focused on a different class of applications, tightly-coupled parallel, these systems provided clear evidence that PC's could deliver serious computing power.

The growth of the Worldwide Web (WWW)[21] and exploding popularity of the Internet created a new much larger scale opportunity for distributed computing. For the first time, millions of desktop PC's were connected to wide-area networks both in the enterprise and in the home. The number of machines potentially accessible to an Internet-based distributed

computing system grew into the tens of millions of systems for the first time. The scale of the resources (millions), the types of systems (windows PC's, laptops), and the typical ownership (individuals, enterprises) and management (intermittent connection, operation) gave rise to a new explosion of interest in and a new set of technical challenges for distributed computing.

In 1996, Scott Kurowski partnered with George Woltman to begin a search for large prime numbers, a task considered synonymous with the largest supercomputers. This effort, the "Great Internet Mersenne Prime Search" or GIMPS[22, 23], has been running continuously for over five years with over 200,000 machines and has discovered the 35th, 36th, 37th, 38th, and 39th Mersenne primes – the largest known prime numbers. The most recent was discovered in November 2001 and is over 4 million digits.

The GIMPS project was the first project taken on by Entropia, Inc, a startup commercializing distributed computing. Another group, distributed.net[24], pursued a number of cryptography related distributed computing projects in this period as well. In 1997, the best-known Internet distributed computing project SETI@home[25] began and rapidly grew to several million machines (typically about 0.5 million active). These early Internet distributed computing systems showed that aggregation of very-large-scale resources was possible and that the resulting system dwarfed the resources of any single supercomputer, at least for a certain class of applications. But these projects were single-application systems, difficult to program and deploy, and very sensitive to the communication-to-computation ratio. A simple programming error could cause network links to be saturated and servers to be overloaded.

The current generation of distributed computing systems, a number of which are commercial ventures, provide the capability to run multiple applications on a collection of desktop and server computing resources[9, 10, 26, 27]. These systems are evolving towards a general-use compute platform. As such, providing tools for application integration and robust execution are the focus of these systems. Grid technologies developed in the research community [2, 3] have focused on issues of security, interoperation, scheduling, communication, and storage. In all cases, these efforts have been focused on Unix servers. For example, the vast majority if not all Globus and Legion activity has been done on Unix servers.

Such systems differ significantly from Entropia, as they do not address issues that arise in a desktop environment, including dynamic naming, intermittent connection, untrusted users, etc. Further, they do not address a range of challenges unique to the Windows environment, whose five major variants are the predominant desktop operating system.

Requirements for Distributed Computing

Desktop Grid systems begin with a collection of computing resources, heterogeneous in hardware and software configuration, distributed throughout a corporate network, and subject to varied management and use regimens and aggregate them into an easily manageable and usable single resource. Furthermore, a desktop grid system must do this in a fashion that ensures that there is little or no detectable impact on the use of the computing resources for other purposes. For end-users of distributed computing, the aggregated resources must be presented as a simple to use, robust resource. Based on our experience with corporate end-users, the following requirements are essential for a viable enterprise desktop grid solution:

Efficiency – The system harvests virtually all of the idle resources available. The Entropia system gathers over 95% of the desktop cycles unused by desktop user applications.

Robustness – Computational jobs must complete with predictable performance, masking underlying resource failures.

Security – The system must protect the integrity of the distributed computation (tampering with or disclosure of the application data and program must be prevented). In addition, the desktop grid system must protect the integrity of the desktops, preventing applications from accessing or modifying desktop data.

Scalability – Desktop grids must scale to the 1,000's, 10,000's, and even 100,000's of desktop PC's deployed in enterprise networks. Systems must scale both upward and downward – performing well with reasonable effort at a variety of system scales.

Manageability – With thousands to hundreds of thousands of computing resources, management and administration effort in a desktop grid cannot scale up with the number of resources. Desktop grid systems must achieve manageability that requires no incremental human effort as clients are added to the system. A crucial element is that the desktop grid cannot increase the basic desktop management effort.

Unobtrusiveness – Desktop grids share resources (computing, storage, and network resources) with other usage in the corporate IT environment. The desktop grid's use of these resources should be unobtrusive, so as not to interfere with the primary use of desktops by their primary owners and networks by other activities.

Openness / Ease of Application Integration – Desktop grid software is a platform which supports applications which in turn provide value to the end-users. Distributed computing systems must support applications developed with varied programming languages, models, and tools – all with minimal development effort.

Together, we believe these seven criteria represent the key requirements for distributed computing systems.

Entropia System Architecture

The Entropia system addresses the seven key requirements by aggregating the raw desktop resources into a single logical resource. The aggregate resource is reliable, secure, and predictable, despite the fact that the underlying raw resources are unreliable (machines may be turned off or rebooted), insecure (untrusted users may have electronic and physical access to machines) and unpredictable (machines may be heavily used by the desktop user at any time). The logical resource provides high performance for applications through parallelism while always respecting the desktop user and his or her use of the desktop machine. Furthermore, the single logical resource can be managed from a single administrative console. Addition or removal of desktop machines is easily achieved, providing a simple mechanism to scale the system as the organization grows or as the need for computational cycles grows.

To support a large number of applications, and to support them securely, we employ a proprietary binary sandboxing technique that

enables any Win32 application to be deployed in the Entropia system without modification and without any special system support. Thus, end-users can compile their own Win32 applications and deploy them in a matter of minutes. This is significantly different than the early large-scale distributed computing systems which required extensive rewriting, recompilation, and testing of the application to ensure safety and robustness.

Layered Architecture

The Entropia system architecture consists of three layers: physical management, scheduling, and job management (see Figure 1). The base, the Physical Node Management layer, provides basic communication and naming, security, resource management, and application control. The second layer is Resource Scheduling, providing resource matching, scheduling, and fault tolerance. Users can interact directly with the Resource Scheduling layer through the available APIs, or alternatively through the third layer, Job Management, which provides management facilities for handling large numbers of computations and files. Entropia provides a Job Management system, but existing job management systems can also be used.

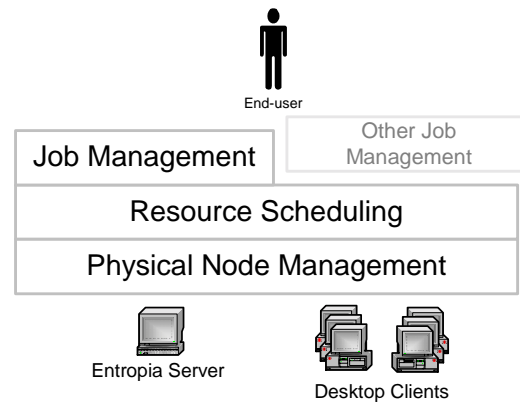


Figure 1. Architecture of the Entropia Distributed Computing System. The Physical Node Management layer and Resource Scheduling layer span the servers and client machines. The Job Management layer runs only on the servers. Other (non-Entropia) Job Management systems can be used with the system.

Physical Node Management – The desktop environment presents numerous unique challenges to reliable computing. Individual client machines are under the control of the desktop user or IT manager. As such, they can be shutdown, rebooted, reconfigured, and be disconnected from the network. Laptops may be

offline or just off for long periods of time. The Physical Node Management layer of the Entropia system manages these and other low-level reliability issues.

The Physical Node Management layer provides naming, communication, resource management, application control, and security. The resource management services capture a wealth of node information (e.g. physical memory, CPU speed, disk size and free space, software version, data cached, etc.), and collect it in the system manager.

This layer also provides basic facilities for process management including file staging, application initiation and termination, and error reporting. In addition, the physical node management layer ensures node recovery, terminating runaway and poorly behaving applications.

The security services employ a range of encryption and binary sandboxing technologies to protect both distributed computing applications and the underlying physical node. Application communications and data are protected with high quality cryptographic techniques. A binary sandbox controls the operations and resources that are visible to distributed applications on the physical nodes, controlling access to protect the software and hardware of the underlying machine.

Finally, the binary sandbox also controls the usage of resources by the distributed computing application. This ensures that the application does not interfere with the primary users of the system – it is unobtrusive – without requiring a rewrite of the application for good behavior.

Resource Scheduling – A desktop grid system consists of resources with a wide variety of configurations and capabilities. The resource scheduling layer accepts units of computation from the user or job management system, matches them to appropriate client resources, and schedules them for execution. Despite the resource conditioning provided by the physical node management layer, the resources may still be unreliable (indeed the application software itself may be unreliable in its execution to completion). Therefore the Resource Scheduling layer must adapt to changes in resource status and availability, and to high failure rates. To meet these challenging requirements the Entropia system can support multiple instances of heterogeneous schedulers.

This layer also provides simple abstractions for IT administrators, which automate the majority of administration tasks with reasonable defaults, but allow detailed control as desired.

Job Management – Distributed computing applications often involve large overall computation (thousands to millions of CPU hours) submitted as a single large job. These jobs consist of thousands to millions of smaller computations and often arise from statistical studies (i.e. Monte Carlo or Genetic algorithm), parameter sweep, or database search (bioinformatics, combinatorial chemistry, etc.). Because so many computations are involved, tools to manage the progress and status of each piece, in addition to the performance of the aggregate job in order to provide short, predictable turnaround times are provided by the job management layer. The job manager provides simple abstractions for end users,

Entropia’s three layer architecture provides a wealth of benefits in system capability, ease of use by end-users and IT administrators, and for internal implementation. The modularity provided by the Entropia system architecture allows the physical node layer to contain many of the challenges of the resource operating environment. The physical node layer manages many of the complexities of the communication, security, and management, allowing the layers above to operate with simpler abstractions. The resource scheduling layer deals with unique challenges of the breadth and diversity of resources, but need not deal with a wide range of lower level issues. Above the resource

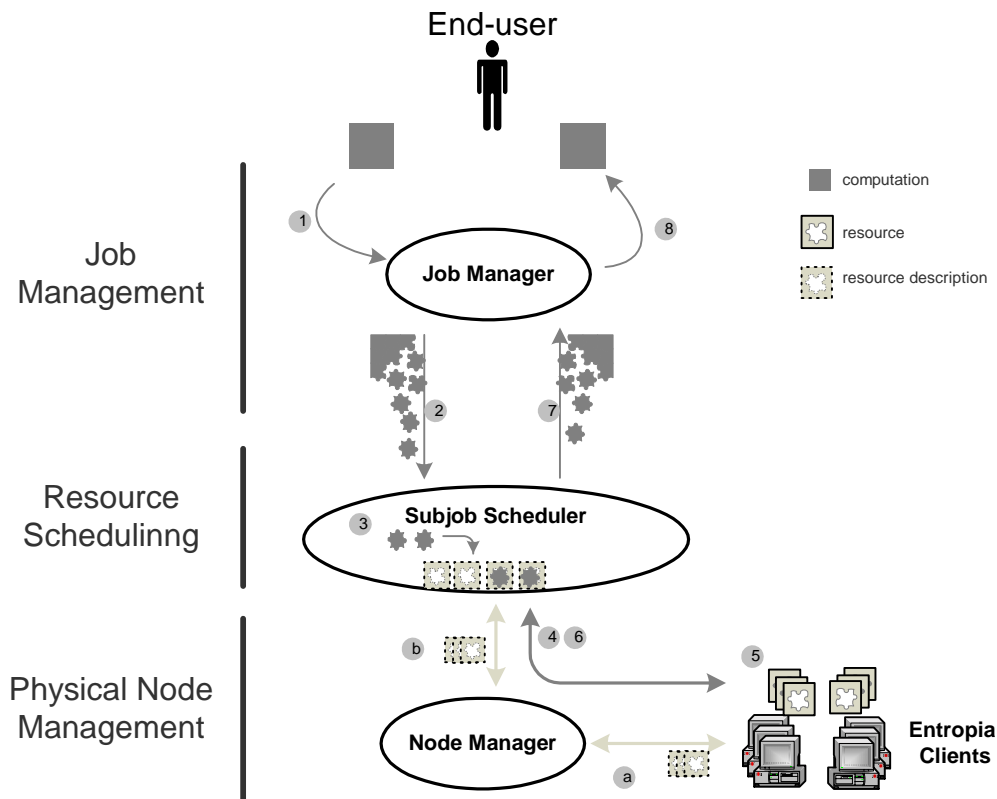


Figure 2: Application Execution on the Entropia System. End-user submits computation to Job Management (1). The Job Manager breaks up the computation into many independent “subjobs” (2) and submits the subjobs to the resource scheduler. In the mean time, the available resources of a client are periodically reported to the Node Manager (a) that informs the Subjob Scheduler (b) using the resource descriptions. The Subjob Scheduler matches the computation needs with the available resources (3) and schedules the computation to be executed by the clients (4,5,6). Results of the computation are sent to the Job Manager (7), put together, and handed back to the end-user (8).

delivering a high degree of usability in an environment where it is easy to drown in the data, computation, and the vast numbers of activities.

scheduling layer, the job management layer deals with mostly conventional job management issues. Finally, the higher-level abstractions presented by each layer support the easy

enabling of applications. This process is highlighted in the next section.

Programming Desktop Grid Applications

The Entropia system is designed to support easy application enabling. Each layer of the system supports higher levels of abstraction, hiding more of the complexity of the underlying resource and execution environment while providing the primitives to get the job done. Applications can be enabled unaware of low-level system details, yet can be run with high degrees of security and unobtrusiveness. In fact, unmodified application binaries designed for server environments are routinely run in production on desktop grids using the Entropia technology. Further, desktop grid computing versions of applications can leverage existing job coordination and management designed for existing cluster systems because the Entropia platform provides high capability abstractions, similar to those used for clusters. We describe two example application-enabling processes:

Parameter Sweep (single binary, many sets of parameters)

1. Process application binary to wrap in Entropia virtual machine, automatically providing security and unobtrusiveness properties
2. Modify your scripting (or front-end job management) to call Entropia job submission comment and catch completion notification
3. Execute large parameter sweep jobs on 1,000 to 100,000 nodes
4. Execute millions of sub jobs

Data Parallel (single application, applied to parts of a database)

1. Process application binaries to wrap in Entropia virtual machine, automatically providing security and unobtrusiveness properties
2. Design database splitting routines and incorporate in Entropia job manager system
3. Design result combining techniques and incorporate in Entropia job manager system
4. Upload your data into the Entropia data management system
5. Execute your application exploiting Entropia's optimized data movement and caching system

6. Execute Jobs will millions of sub-parts

Entropia Usage Scenarios

The Entropia system is designed to interoperate with many computing resources in an enterprise IT environment. Typically, users are focused on integrating desktop grid capabilities with other large-scale computing and data resources such as linux clusters, database servers, or mainframe systems. We give two example integrations below:

Single Submission Users often make use of both linux cluster and desktop grid systems, but prefer not to manually select resources as delivered turnaround time depends critically on detailed dynamic information such as changing resource configurations, planned maintenance, and even other competing users. In such situations, a single submission interface, where an intelligent scheduler places computations where the best turnaround time can be achieved gives end users the best performance.

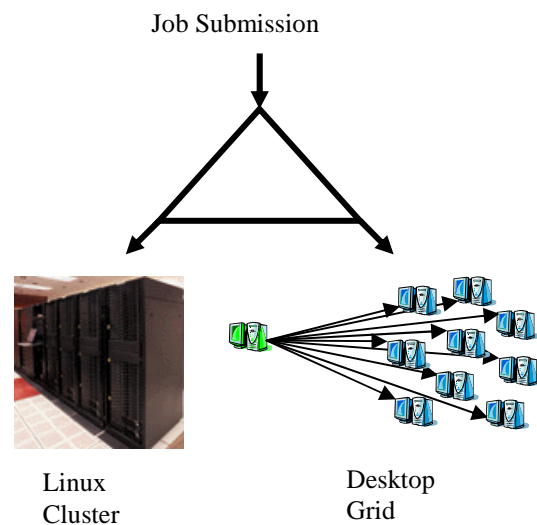


Figure 3. Single Submission to multiple Grid systems.

Large Data Application For many large data applications, canonical copies of data are maintained in enhanced relational database systems. These systems are accessed via the network, and are often unable to sustain the resulting data traffic when computational rates are increased by factors of 100 to 10,000. The Entropia system provides for data copies to be staged and managed in the desktop grid system,

50K Compound Throughput Scalability

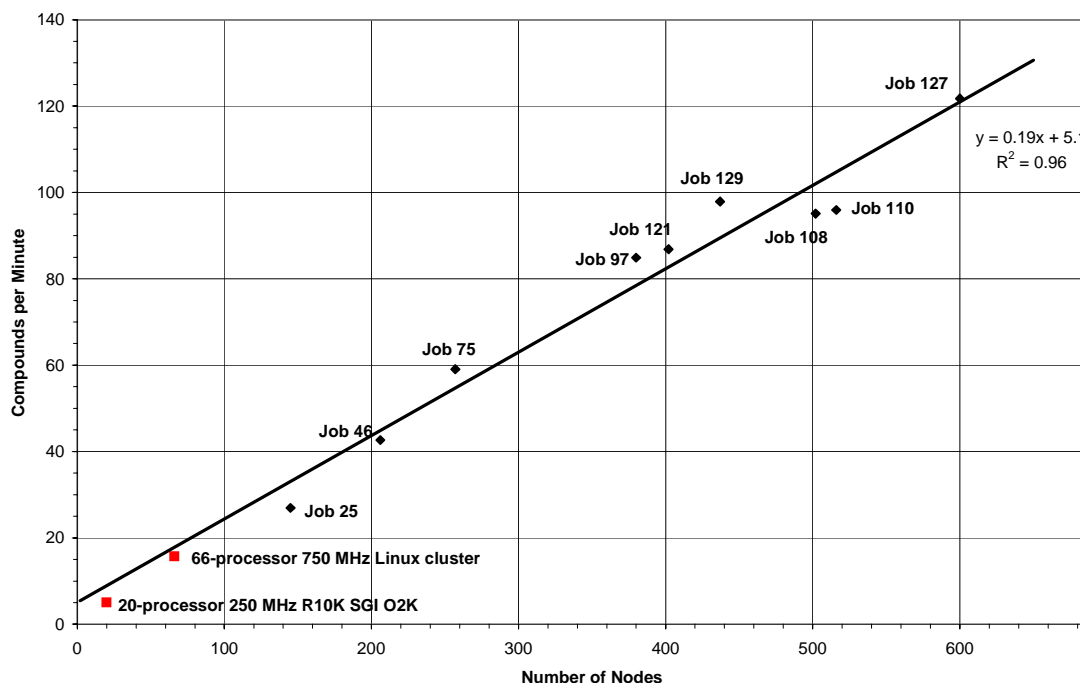


Figure 4: Scaling Of Entropia System Throughput On Virtual Screening Application.

allowing the performance demands of the desktop grid to be separated from the core data infrastructure. A key benefit is that the desktop grid can then provide maximum computational speedup.

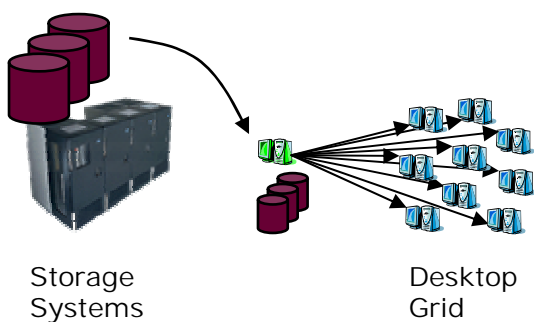


Figure 5. Data Staging in the Entropia System

Applications and Performance

Early adoption of distributed computing technology is focused on applications that are easily adapted and whose high demands cannot be met by traditional approaches whether for cost or technology reasons. For these applications; sometimes called “high throughput” applications, very large capacity provides a new kind of capability.

The applications exhibit large degrees of parallelism (thousands to even hundreds of millions) with little or no coupling, in stark

contrast to traditional parallel applications, which are more tightly coupled. These High Throughput Computing applications are the only ones capable of not being limited by Amdahl’s law. As shown in Figure 3, these applications can exhibit excellent scaling, greatly exceeding the performance of many traditional high performance computing platforms.

We believe the widespread availability of distributed computing will encourage reevaluation of many existing algorithms to find novel uncoupled approaches, ultimately increasing the number of applications suitable for distributed computing. For example, Monte Carlo or other stochastic methods that are too inefficient using conventional computing approaches may prove attractive when considering time to solution.

Four application types successfully using distributed computing include virtual screening; sequence analysis; molecular properties and structure; and financial risk analysis. We discuss the basic algorithmic structure, from a computational and concurrency perspective, the typical use and run sizes and the computation/communication ratio. A common characteristic of all of these applications is the independent evaluation, requiring several minutes or more of CPU time, of at most a few megabytes of data.

Summary and Futures

Distributed computing has the potential to revolutionize how much of large-scale computing is achieved. If easy-to-use distributed computing can be seamlessly available and accessed, applications will have access to dramatically more computational power to fuel increased functionality and capability. The key challenges to acceptance of distributed computing include robustness, security, scalability, manageability, unobtrusiveness, and openness/ease of application integration.

Entropia's system architecture consists of three layers: a physical node management layer, resource scheduling, and job scheduling. This architecture provides a modularity that allows each layer to focus on a smaller number of concerns, enhancing overall system capability and usability. This system architecture provides a solid foundation to meet the technical challenges as the use of distributed computing matures; it enables a broadening class of computations by supporting an increasing breadth of computational structure, resource usage, and ease of application integration.

We have described the architecture of the Entropia system and how to apply it – both to applications and in an enterprise IT environment. The system is applicable to a large number of applications, and we have discussed virtual screening, sequence analysis, molecular modeling, and risk analysis in this paper. For all of these application domains, excellent linear scaling has been demonstrated for large distributed computing systems. We expect to extend these results to a number of other domains in the near future.

Despite the significant progress documented here, we believe we are only beginning to see the mass use of distributed computing. With robust commercial systems such as Entropia only recently available, widespread industry adoption of the technology is only beginning. At this writing, we are confident that within a few years, distributed computing will be deployed and in use in production within a majority of large corporations and research sites.

Acknowledgements

We gratefully acknowledge the contributions of the talented team at Entropia to the design and implementation of this desktop grid system. We specifically acknowledge the contributions of Kenjiro Taura, Scott Kurowski,

Brad Calder, Shawn Marlin, Wayne Schroeder, Jon Anderson, Karan Bhatia, Steve Elbert, and Ed Anady to the definition and development of the system architecture. We also acknowledge Dean Goddette, Wilson Fong, and Mike May for their contributions to applications benchmarking and understanding performance data.

The author is supported in part by the Defense Advanced Research Projects Administration through United States Air Force Rome Laboratory Contracts AFRL F30602-99-1-0534 and the National Science Foundation through the National Computational Science Alliance and NSF EIA-99-75020 GrADS.

- [1] J. Lyman, "Intel Debuts 2.2ghz Pentium 4 Chip," in *The News Factor*, 2002.
- [2] I. Foster and C. Kesselman, "The Globus Project: A Status Report," presented at IPPS/SPDP '98 Heterogeneous Computing Workshop, 1998.
- [3] A. Grimshaw and W. Wulf, "The Legion Vision of a Worldwide Virtual Computer," *Communications of the ACM*, vol. 40, 1997.
- [4] D. Barkai, *Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net*: Intel Press, 2001.
- [5] Sun Microsystems, <http://www.jxta.org>
- [6] I. Foster, *The Grid: Blueprint for a New Computing Infrastructure*: Morgan Kaufmann, 1998.
- [7] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, vol. 15, 2001.
- [8] Entropia Inc, "Entropia Announces Support for Open Grid Services Architecture," Entropia Inc., Press Release Feb 21, 2002.
- [9] United Devices, <http://www.ud.com>
- [10] Platform Computing, <http://www.platform.com>
- [11] A. Bricker, M. Litzkow, and M. Livny, "Condor Technical Summary," Department of Computer Science, University of Wisconsin, Madison, WI, Technical Report 1069, January 1992.
- [12] F. Heart, A. McKenzie, J. McQuillan, and D. Walden, "ARPANET Completion Report," BBN, Technical Report 4799, Jan 1978.
- [13] J. F. Schoch and J. A. Hupp, "The 'Worm' Programs-Early Experience with a Distributed Computation," *Communications of the ACM*, vol. 25, pp. 172-80, 1982.
- [14] C. A. Waldsburger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A Distributed Computational Economy," *IEEE Transactions on Software Engineering*, vol. 18, pp. 103-17, 1992.
- [15] M. J. Litzkow, "Remote Unix Turning Idle Workstations into Cycle Servers," presented at Proceedings of the Summer 1987 USENIX Conference, Phoenix, AZ, USA, 1987.
- [16] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-a Hunter of Idle Workstations," presented at 8th International Conference on Distributed Computing Systems, San Jose, CA, USA, 1988.
- [17] Z. Songnian, Z. Xiaohu, W. Jingwen, and P. Delisle, "Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems,"

- Software - Practice and Experience*, vol. 23, pp. 1305-36, 1993.
- [18] V. S. Sunderam, "PVM: A Framework for Parallel Distributed Computing," *Concurrency: Practice and Experience*, vol. 2, pp. 315-339, 1990.
- [19] A. Chien, M. Lauria, R. Pennington, M. Showerman, G. Iannello, et al., "Design and Evaluation of HPVM-Based Windows Supercomputer," *International Journal of High Performance Computing Applications*, vol. 13, pp. 201-219, 1999.
- [20] T. Sterling, *Beowulf Cluster Computing with Linux*: The MIT Press, 2001.
- [21] M. Gray, "Internet Growth Summary", MIT, <http://www.mit.edu/people/McCray/net/internet-growth-summary.html>
- [22] Entropia Inc, "Researchers Discover Largest Multi-Million-Digit Prime Using Entropia Distributed Computing Grid," Entropia, Press Release Dec. 2001.
- [23] G. Woltman, "The Great Internet Mersenne Prime Search", <http://www.mersenne.org/>
- [24] Distributed.net, "The Fastest Computer on Earth", <http://www.distributed.net/>
- [25] W. T. Sullivan, D. Werthimer, S. Bowyer, "A New Major Seti Project Based on Project Serendip Data and 100,000 Personal Computers," *Astronomical and Biochemical Origins and the Search for Life in the Universe*, 1996.
- [26] Entropia Inc, <http://www.entropia.com>
- [27] DataSynapse Inc., <http://www.datasynapse.com>
- [28] Veridian Systems, "Portable Batch System", <http://www.openpbs.org>
- [29] B. Kramer, M. Rarey, and T. Lengauer, "Evaluation of the Flexx Incremental Construction Algorithm for Protein-Ligand Docking," *PROTEINS: Structure, Functions, and Genetics*, vol. 37, pp. 228-241, 1999.
- [30] M. McGann, "Fred: Fast Rigid Exhaustive Docking", OpenEye, <http://www.eyesopen.com/fred.html>
- [31] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, et al., "Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function," *J. Computational Chemistry*, vol. 19, pp. 1639-1662, 1998.
- [32] T. J. A. Ewing and I. D. Kuntz, "Critical Evaluation of Search Algorithms for Automated Molecular Docking and Database Screening," *Journal of Computational Chemistry*, vol. 9, pp. 1175-1189, 1997.
- [33] G. Jones, P. Willett, and R. C. Glen, "Molecular Recognition of Receptor Sites Using a Genetic Algorithm with a Description of Desolvation," *J. Mol. Biol.*, vol. 245, pp. 43-53, 1995.
- [34] K. Davies, "Think", Dept. of Chemistry, Oxford University, <http://www.chem.ox.ac.uk/cancer/thinksoftware.html>
- [35] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and R. P. Mee, "Empirical Scoring Functions: I. The Development of a Fast Empirical Scoring Function to Estimate the Binding Affinity of Ligands in Receptor Complexes.," *J. Comput. Aided Mol. Design*, vol. 11, pp. 425-445, 1997.
- [36] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A. A. Friesem, C. Aflalo, et al., "Molecular Surface Recognition: Determination of Geometric Fit between Proteins and Their Ligands by Correlation Techniques," *Proc. Natl. Acad. Sci. USA*, vol. 89, pp. 2195-2199, 1992.
- [37] L. F. T. Eyck, J. Mandell, V. A. Roberts, and M. E. Pique, "Surveying Molecular Interactions with Dot," presented at Supercomputing 1995, San Diego, 1995.
- [38] H. J. Bohm, "Towards the Automatic Design of Synthetically Accessible Protein Ligands: Peptides, Amides and Peptidomimetics," *J. Comput. Aided Molec. Design*, vol. 10, pp. 265-272, 1996.
- [39] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool," *J. Mol. Biol.*, vol. 215, pp. 403-410, 1990.
- [40] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, et al., "Gapped Blast and Psi-Blast: A New Generation of Protein Database Search Programs," *Nucleic Acids Res.*, vol. 25, pp. 3389-3402, 1997.
- [41] W. Gish and D. J. States, "Identification of Protein Coding Regions by Database Similarity Search," *Nature Genet.*, vol. 3, pp. 266-272, 1993.
- [42] T. L. Madden, R. L. Tatusov, and J. Zhang, "Applications of Network Blast Server," *Meth. Enzymol.*, vol. 266, pp. 131-141, 1996.
- [43] J. Zhang and T. L. Madden, "Powerblast: A New Network Blast Application for Interactive or Automated Sequence Analysis and Annotation," *Genome Res.*, vol. 7, pp. 649-656, 1997.
- [44] S. R. Eddy, "Hmmer: Profile Hidden Markov Models for Biological Sequence Analysis", <http://hmmer.wustl.edu/>,
- [45] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Positions-Specific Gap Penalties and Weight Matrix Choice.," *Nucleic Acids Research*, vol. 22, pp. 4673-4680, 1994.
- [46] W. R. Pearson and D. J. Lipman, "Improved Tools for Biological Sequence Comparison," *Proc. Natl. Acad. Sci. USA*, vol. 85, pp. 2444-2448, 1988.
- [47] E. Birney, "Wise2: Intelligent Algorithms for DNA Searches", <http://www.sanger.ac.uk/Software/Wise2/>,
- [48] T. F. Smith and M. S. Waterman, "Comparison of Biosequences," *Adv. Appl. Math.*, vol. 2, pp. 482-489, 1981.
- [49] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, et al., "Gaussian 98." Pittsburgh PA: Gaussian, Inc., 2001.
- [50] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, et al., "General Atomic and Molecular Electronic Structure System," *J. Comput. Chem.*, vol. 14, pp. 1347-1363, 1993.
- [51] Schrödinger, "Jaguar", <http://www.schrodinger.com/Products/jaguar.html>,