Utility-based High Performance Digital Library Systems

Hussein Suleman

Department of Computer Science, University of Cape Town,
Private Bag X3, Rondebosch, 7701, South Africa
hussein@cs.uct.ac.za

Abstract. Many practical digital library systems have had to deal with scalability of data collections and/or service provision. Early attempts at enabling this scalability focused on data/services closely coupled with or tightly integrated with various high performance computing platforms. This inevitably resulted in compromises and very specific solutions. This paper presents an analysis of current high performance systems and motivates for why utility computing can subsume existing models and better meet the needs of generic scalable digital library systems.

1 Introduction

Cloud computing is the current but ill-defined buzzword in computing for the use of abstract online resources [15] - utility computing is a specific subset of cloud computing whereby virtual computing and storage resources are acquired (or released) on-demand without the consumer of the resource knowing their physical location [10]. The most common scenario is the need for a large numbers of computers for a short period of time, for example to index a large ingested collection. Utility computing is arguably more cost-effective than other forms of high performance computing (HPC) because of the efficiency of shared resources for which the management is essentially outsourced.

Designers of digital library systems (DLSes) have long grappled with the problem of how to construct systems that meet multiple criteria, including elements of scalability and preservation of data [7]. As HPC technologies have emerged and matured, DLSes were designed to exploit them. These include symmetric multiprocessors, clusters and grids. Utility computing, a relatively newer paradigm, is arguably a better fit because it provides an ideal compromise among features to enable high performance digital libraries that can be provisioned as needed (i.e., on-demand).

The rest of this paper provides an overview of current high performance computing technology and how these are utilised in the provision of digital library services, and then how an alternative digital library system can be designed based on utility computing. The focus is on service provision rather than data scalability as service provision must be built over utility computing platforms while scalable data storage is often a basic service.

2 High Performance Computing

High performance computing is about the construction of hardware and software systems to process a computationally-intensive job. This often is distinguished from high throughput computing, which is about systems to process a large number of jobs. As these are not completely disjoint, the former term has become the norm to describe both forms of computing.

Many forms of HPC have been adopted in computer systems, including: cluster computing, grid computing, cloud/utility computing, edge computing, symmetric multiprocessing, multicore computing, general-purpose graphics processing units, cell processors, field programmable gate arrays, server farms and volunteer computing. Each of these is discussed briefly below.

2.1 CPU Models

The earliest form of scalability in online systems was effected using server farms. These are individual machines, each able to respond to requests from the Web, with possibly a shared database backend or independent replicated data stores (if the site is mostly read-only). Server farms are effective when the majority of processing does not require database updates - in this case the round-robin IP address resolution from the DNS system automatically distributes the load evenly among the servers. DSpace implementers have recommended the use of such a technique for some degree of scalability because it is supported by the Web server and the architecture of DSpace [11].

Modern multicore CPUs contain multiple processing units in a single microchip. Each core is capable of logic and arithmetic operations, with a single shared memory for all cores. On-chip caches can be shared in a multicore CPU to allow for a high degree of integration. Multicore machines are supported by operating systems at the process/thread level - the operating system is able to allocate different processes to run on different cores simultaneously. Thus, an application that is appropriately split into sub-tasks will automatically take advantage of the additional cores. Most desktop computers and servers use multicore CPUs, although most programming models are still serial.

Symmetric multiprocessing refers to computers that contain multiple independent CPUs on separate microchips. These also are managed by modern operating systems at the process level and behave very similarly to multi-core systems. They are not, however, as efficient for inter-process communication because this is between-chip rather then within-chip. SMP systems were popular in early supercomputers and were the focus of early attempts to achieve scalability in digital library systems [4].

Mosix [5] is an operating system tool that binds a group of networked machines into a single virtual operating system. The programmer or end user sees a group of machines as a single machine and Mosix handles the distribution of processes and inter-process communication. Mosix has the advantages of SMP from an end-user perspective, but has a much slower interconnecting network between processors. Mosix, SMP and multicore systems were used in experiments with parallelising metadata harvesting - and it was shown that the same applications would work on all architectures with varying performance [19].

Cell processors [14] contain a main controlling core but many sub-cores for execution of data-parallel tasks. Cell processors were made popular by their incorporation into the Playstation 3 video games consoles for graphics processing.

General-purpose Graphics Processing Units (GPGPUs) [17] exploit the increasing computational power of microchips on commodity graphics cards, which are able to perform some operations in parallel, thus resulting in much faster processing than traditional CPUs. As such, they are useful for a large number of computation-intensive tasks. Tools for programming GPUs are still in their infancy and high end GPUs are not yet considered commodity components.

Field-Programmable Gate Arrays (FPGAs) [9] are microchips that can perform specific tasks in hardware which is reconfigured at runtime. This allows for maximum performance, although there are limitations in the size of FPGAs and there is still a need for a CPU for general purpose processing. FPGAs are most suited to highly parallel computation-intensive tasks.

There are currently no well-known digital library applications that explicitly use cell processors, GPGPUs or FPGAs.

2.2 Machine/Storage Models

Cluster computing is when a group of co-located standalone computers - usually commodity machines - are interconnected into a LAN using a high speed network fabric. Each machine on the cluster runs an independent task or set of tasks, with coordination taking place over the high speed network. Clusters are suited to problems which can be decomposed into separate processes with some communication. The amount of communication is closely related to the speed of the interconnect, so general purpose clusters tend to use fairly high-end interconnects such as Gigabit Ethernet. Clusters have been used in scientific computing for space-partitioned discrete event simulations. Most commercial search engines such as Google use a cluster architecture for their indexing and querying operations because of the high memory bandwidth required for efficient indexing (whether for search or browse indices) [6].

Grid computing refers to a coordinated sharing of separately-owned resources in a wide area network, typically overlaid on the Internet [13]. Grid computing encompasses a suite of technologies to control access to resources, support data transfers and manage processes on remote systems. As the computational and storage resources are potentially far apart, the network interconnect is not as controlled and speeds are typically far lower than with clusters. As a result, grids are suitable where there is little communication among processes and where the data to be transferred is small relative to the amount of computation to be performed. In the digital library community, grids have been used for both computation and storage. Storage Resource Broker and iRODS [18] are used to create data (storage) grids over hetergeneous resources. DILIGENT [8] is a digital library management system that used a grid infrastructure to interconnect services distributed over a wide area.

Edge computing is the use of computers at the edges of the network to perform computation, instead of machines at the core. While this is not rigorously defined, there are some examples in practice that qualify as edge computing. Content distribution networks are one example, where content and sometimes services are distributed to sites closest to the end-user. This makes the service provision scalable by reducing the network distance from user to service. Another example is the use of the user's desktop machine to perform some of the computation instead of all computation happening on servers. Asynchronous Javascript and XML (AJAX) Web applications are able to scale to a far greater degree because much of the computation occurs on the clients' computers. In the Bleek and Lloyd project, a prototype search engine was created in AJAX to demonstrate how ranked retrieval could be performed on the client using local data, thus eliminating the need for network interaction and resources on the server [20].

Volunteer computing is the use of idle desktop computers to perform computation for which it is not possible to secure dedicated high performance systems. SETI@Home popularised this form of distributed computing for the large-scale analysis of astronomical data and the technology has since evolved into the general purpose BOINC framework for sharing CPU cycles [3]. Volunteer computing has great potential for scalability but sometimes limited applicability because of the uncertain widely distributed resources and unclear trust relationships. No known digital library systems are using volunteer computing.

Figure 1 illustrates key differences among these machine models, arranged from centralised to distributed. The dotted line cutting across the categories indicates how utility computing compares against the other machine models.

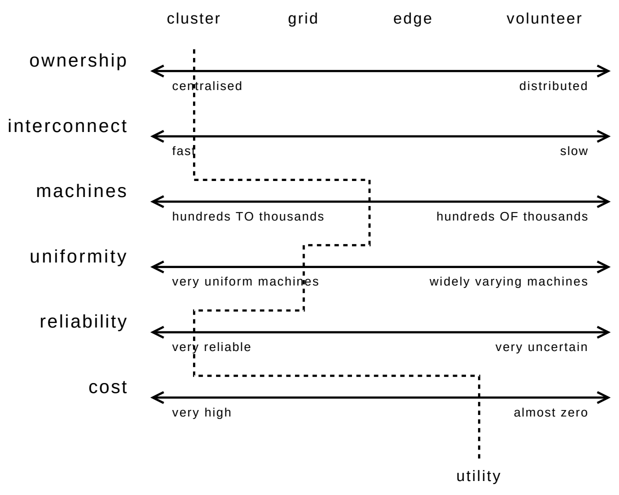

Fig. 1. HPC Machine configurations

2.3 Utility Computing

Utility computing has emerged as a generalisation of most of the existing HPC models, where the exact nature of the technology is not visible to the end-user or the programmers. Instead all interaction is via an API and the user may assume that the machines are interconnected with a high-speed network, with localised access to data.

Two core services are provided in utility clouds: virtual machines and virtual data stores. Virtual machines are machines that can be created on demand, often programmatically, and used remotely. Amazon's Elastic Compute Cloud (EC2) [1] is a popular example of this service. Virtual data stores allow the storage of a virtually unrestricted amount of information, with a uniform addressing scheme. Amazon's Simple Storage System (S3) [2] is a popular example of this service.

The provisioning of utility computing can be done either remotely as an outsourced operation, or internally using similar APIs. Eucalyptus [16] is a software suite that emulates the APIs created by Amazon, allowing an organisation to provision its own utility computing services, while still retaining the benefit of a clean separation between services and provisioning.

As shown in Figure 1, utility computing has the benefits of clusters in terms of ownership of resources being centralised, high speed interconnects, a reasonable level of homogeneity of machines and a high degree of reliability. In addition, there are generally many more machines available than in a typical cluster (grid-like), while the cost of processing is much smaller since there is no upfront and continuous hardware acquisition cost (volunteer computing-like).

3 Designing Utility Computing HPDLs

While utility computing systems may offer a single scalable data store, there is currently no single scalable machine architecture. Applications therefore must be designed to make optimal use of virtual machines, possibly adopting the various architectures of pre-utility systems [21]. Figure 2 illustrates 4 possible architectures to incorporate scalability within the utility cloud.

The Proxy architecture has a manager node act as a proxy for all external connections to service nodes. This is similar to a cluster.

The Redirector architecture has the manager node act as a lookup table for service nodes, with the external client making direct connections into the cloud thereafter. This is similar to a grid.

In the Round Robin architecture the client uses the DNS system (or a similar resolution service) to obtain the address of the next service node to use. DNS typically rotates IP addresses to balance the load but the manager must still update the DNS tables based on the number of service nodes available. This is similar to a server farm.

In the Client-side model, the manager sends a list of nodes to the client upon request and the client uses this list to rotate requests to service nodes. This is similar to an edge computing model. This model ought to have the highest level of scalability and reliability.

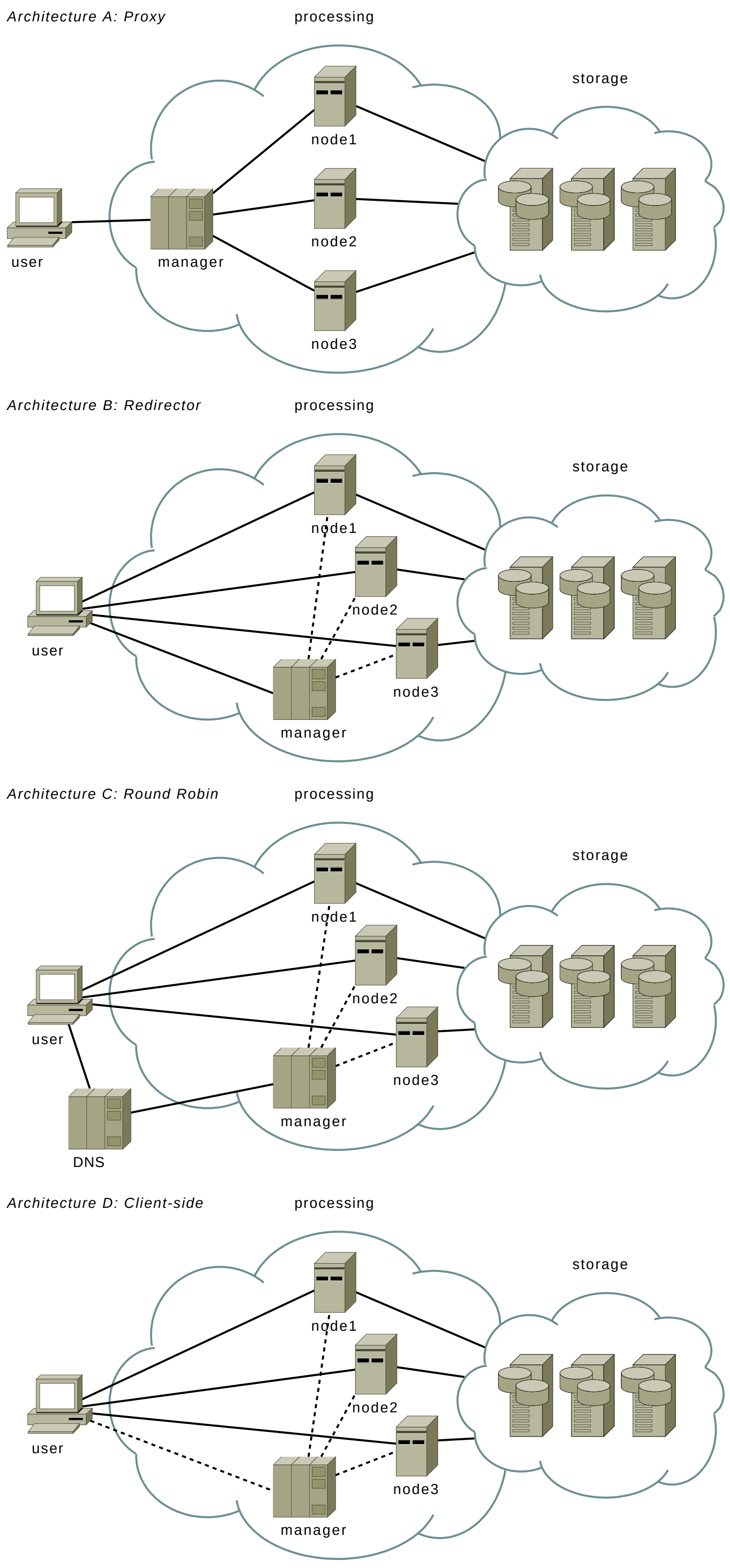

Fig. 2. Cloud Computing machine configurations

Thus, within a utility cloud, digital library services can be designed to emulate most existing HPC architectures - given that different DL services are most efficient on different architectures, utility computing offers the flexibility of multiple architectural models.

4 Conclusions and Future Work

Utility computing shows promise to provide computing facilities with the best features of multiple HPC technologies and substantial flexibility for implementing scalable systems. The system architectures presented here are currently being developed into prototype digital repository systems to experimentally verify scalability for popular services.

There is, however, still a need to balance scalability with preservation. The rapid advances in HPC technology has meant that no one option has been preferred by the DL community. Thus, there are no standard approaches to scalability and this makes preservation even more difficult. The Duraspace project is attempting to build more preservable cloud stores by replicating data across multiple cloud platforms, but this is still a work in progress [12].

Utility computing narrows the gap between HPC and on-demand computing by making resources available as required. Future work will focus on how to make use of those resources most cost-efficiently, without human intervention or with minimal intervention. This is vital for small collections to scale up in size or services without a redesign of the digital library system.

Finally, utility computing is not common in most developing countries because the Internet connection into the utility cloud is not fast enough. More work is needed on the localisation of utility clouds to ensure equal access to the technology.

5 Acknowledgements

This project is made possible by funding from University of Cape Town and NRF.

References

1. Amazon.com (2009). Amazon Elastic Compute Cloud. Website http://aws.amazon.com/ec2/
2. Amazon.com (2009). Amazon Simple Storage Service (Amazon S3). Website http://aws.amazon.com/s3/
3. Anderson, D. P. (2004). BOINC: A system for public-resource computing and storage, in Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, 8 November 2008, ACM Press.
4. Andresen, D., Yang, T., Egecioglu, O., Ibarra, O. H., and Smith, T. R. (1996). Scalability Issues for High Performance Digital Libraries on the World Wide Web, ADL, pp. 139, Third International Forum on Research and Technology Advances in Digital Libraries (ADL '96), IEEE Computer Society.

5. Bar, M. (2003). openMosix, a Linux Kernel Extension for Single System Image Clustering. in Proceedings of Linux Kongress: 10th International Linux System Technology Conference, 15-16 October, Saarbrücken, Germany.
6. Barroso, L. A., Dean, J., and Hölzle, U. (2003). Web Search for a Planet: The Google Cluster Architecture, IEEE Micro, March-April 2003, IEEE Computer Society.
7. Borgman, C. L. (2002). Challenges in Building Digital Libraries for the 21st Century, in Proceedings of Digital Libraries: People, knowledge, and technology : 5th International Conference on Asian Digital Libraries, ICADL 2002, Singapore, 11-14 December 2002, Springer.
8. Candela, L., Castelli, D., Pagano, P., and Simi, M. (2005). Moving Digital Library Service Systems to the Grid, Lecture Notes in Computer Science, Volume 3664, Springer.
9. Compton, K., and Hauck, S. (2000). An Introduction to Reconfigurable Computing, IEEE Computer, April 2000, IEEE Computer Society.
10. Danielson, K. (2009). Distinguishing Cloud Computing from Utility Computing, eBizQ, 26 March 2008. Available http://www.ebizq.net/blogs/saasweek/2008/03/distinguishing_cloud_computing/
11. DSpace.org (2009). Clustering, DSpace Wiki. Available http://wiki.dspace.org/index.php/Clustering
12. Duraspace (2009). Duracloud. Website http://www.duraspace.org/duracloud.html
13. Foster, I., and Kesselman, C. (2004). The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann.
14. Gschwind, M., Hofstee, H. P., Flachs, B., Hopkins, M., Watanabe, Y., and Yamazaki, T. (2006). Synergistic Processing in Cell's Multicore Architecture, IEEE Micro, March-Apruil 2006, IEEE Computer Society.
15. Knorr, E., and Gruman, G. (2008). What cloud computing really means, Infoworld, 7 April 2008. Available http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031
16. Nurmi, D., Wolski, R., and Grzegorczyk, C. (2008). Eucalyptus : A Technical Report on an Elastic Utility Computing Archietcture Linking Your Programs to Useful Systems, Computer Science Technical Report Number 2008-10, UCSB. Available http://www.cs.ucsb.edu/research/tech_reports/reports/2008-10.pdf
17. Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., and Purcell, T. J. (2007). A Survey of General-Purpose Computation on Graphics Hardware, Computer Graphics Forum, 26(1), pp. 80-113, Blackwell-Synergy.
18. Rajasekar, A., Wan, M., Moore, R., and Schroeder, W. (2006). A prototype rule-based distributed data management system, in Proceedings of HPDC Workshop on Next-Generation Distributed Data Management, 19-23 June 2006, Paris.
19. Suleman, H. (2006). Parallelising Harvesting, in Proceedings of 9th International Conference on Asian Digital Libraries, ICADL 2006, Kyoto, 27-30 November 2006, Springer.
20. Suleman, H. (2007). in-Browser Digital Library Services, in Kovacs, Laszlo, Norbert Fuhr and Carlo Meghini (eds): Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference (ECDL 2007), pp. 462-465, 16-19 September, Budapest, Hungary. Available http://pubs.cs.uct.ac.za/archive/00000434/01/ecdl_2007_ajax.pdf
21. Wayner, P. (2008). Cloud versus cloud: A guided tour of Amazon, Google, AppNexus, and GoGrid, InfoWorld, 21 July 2008. Available http://www.infoworld.com/d/cloud-computing/ cloud-versus-cloud-guided-tour-amazon-google-appnexus-and-gogrid-122