Hybrid cloud and cluster computing paradigms for life science applications

Judy Qiu[1,2§], Jaliya Ekanayake[1,2]*, Thilina Gunarathne[1,2]*, Jong Youl Choi1[1,2]*, Seung-Hee Bae[1,2]*, Hui Li[1,2]*, Bingjing Zhang[1,2]*, Tak-Lon Wu[1,2]*, Yang Ryan[1,2]*, Saliya Ekanayake[1,2]*, Adam Hughes[1,2]*, Geoffrey Fox[1,2]*

[1]School of Informatics and Computing, Indiana University, Bloomington, IN 47405, USA

[2]Pervasive Technology Institute, Indiana University, Bloomington, IN 47408, USA

*These authors contributed equally to this work

[§]Corresponding author

Email addresses:

JQ: xqiu@indiana.edu

JE: jekanaya@indiana.edu

TG: tgunarat@indiana.edu

JYC: jychoi@indiana.edu

SHB: sebae@indiana.edu

LH: lihui@indiana.edu

BJZ: zhangbj@indiana.edu

TLW: taklwu@indiana.edu

SE: sekanaya@indiana.edu

YR: yangruan@indiana.edu

AH: adalhugh@indiana.edu

GCF: gcf@indiana.edu

Abstract

Clouds and MapReduce have shown themselves to be a broadly useful approach to scientific computing especially for parallel data intensive applications. However they have limited applicability to some areas such as data mining because MapReduce has poor performance on problems with an iterative structure present in the linear algebra that underlies much data analysis. Such problems can be run efficiently on clusters using MPI leading to a hybrid cloud and cluster environment. This motivates the design and implementation of an open source Iterative MapReduce system Twister.

Comparisons of Amazon, Azure, and traditional Linux and Windows environments on common applications have shown encouraging performance and usability comparisons in several important non iterative cases. These are linked to MPI applications for final stages of the data analysis. Further we have released the open source Twister Iterative MapReduce and benchmarked it against basic MapReduce (Hadoop) and MPI in information retrieval and life sciences applications.

The hybrid cloud (MapReduce) and cluster (MPI) approach offers an attractive production environment while Twister promises a uniform programming environment for many Life Sciences applications. We used commercial clouds Amazon and Azure and the NSF resource FutureGrid to perform detailed comparisons and evaluations of different approaches to data intensive computing. Several applications were developed in MPI, MapReduce and Twister in these different environments.

Background

Cloud computing [1] is at the peak of the Gartner technology hype curve [2], but there are good reasons to believe that it is for real and will be important for large scale scientific computing:

1) Clouds are the largest scale computer centers constructed, and so they have the capacity to be important to large-scale science problems as well as those at small scale.
2) Clouds exploit the economies of this scale and so can be expected to be a cost effective approach to computing. Their architecture explicitly addresses the important fault tolerance issue.
3) Clouds are commercially supported and so one can expect reasonably robust software without the sustainability difficulties seen from the academic software systems critical to much current cyberinfrastructure.
4) There are 3 major vendors of clouds (Amazon, Google, and Microsoft) and many other infrastructure and software cloud technology vendors including Eucalyptus Systems, which spun off from UC Santa Barbara HPC research. This competition should ensure that clouds develop in a healthy, innovative fashion. Further attention is already being given to cloud standards [3].
5) There are many cloud research efforts, conferences, and other activities including Nimbus [4], OpenNebula [5], Sector/Sphere [6], and Eucalyptus [7].
6) There are a growing number of academic and science cloud systems supporting users through NSF Programs for Google/IBM and Microsoft Azure systems. In NSF OCI, FutureGrid [8] offers a cloud testbed, and Magellan [9] is a major DoE experimental cloud system. The EU framework 7 project VENUS-C [10] is just starting with an emphasis on Azure.
7) Clouds offer attractive "on-demand" elastic and interactive computing.

Much scientific computing can be performed on clouds [11], but there are some well-documented problems with using clouds, including:

1) The centralized computing model for clouds runs counter to the principle of "bringing the computing to the data", and bringing the "data to a commercial cloud facility" may be slow and expensive.
2) There are many security, legal, and privacy issues [12] that often mimic those of the Internet which are especially problematic in areas such health informatics.
3) The virtualized networking currently used in the virtual machines (VM) in today's commercial clouds and jitter from complex operating system functions increases synchronization/communication costs. This is especially serious in large-scale parallel computing and leads to significant overheads in many MPI applications [13-15]. Indeed, the usual (and attractive) fault tolerance model for clouds runs counter to the tight synchronization needed in most MPI applications. Specialized VMs and operating systems can give excellent MPI performance [16] but we will consider commodity approaches here. Amazon has just announced Cluster Compute instances in this area.
4) Private clouds do not currently offer the rich platform features seen on commercial clouds [17].

Some of these issues can be addressed with customized (private) clouds and enhanced bandwidth from research systems like TeraGrid to commercial cloud networks. However it seems likely that clouds will not supplant traditional approaches for very large-scale parallel (MPI) jobs in the near future. Thus we consider a hybrid model with jobs running on classic HPC systems, clouds, or both as workflows could link HPC and cloud systems. Commercial clouds support "massively parallel" or “many

tasks" applications, but only those that are loosely coupled and so insensitive to higher synchronization costs. We focus on the MapReduce programming model [18], which can be implemented on any cluster using the open source Hadoop [19] software for Linux or the Microsoft Dryad system [20, 21] for Windows. MapReduce is currently available on Amazon systems, and we have developed a prototype MapReduce for Azure.

Results

Metagenomics - A Data Intensive Application Vignette

The study of microbial genomes is complicated by the fact that only small number of species can be isolated successfully and the current way forward is metagenomic studies of culture-independent, collective sets of genomes in their natural environments. This requires identification of as many as millions of genes and thousands of species from individual samples. New sequencing technology can provide the required data samples with a throughput of 1 trillion base pairs per day and this rate will increase. A typical observation and data pipeline [22] is shown in Figure 1 with sequencers producing DNA samples that are assembled and subject to further analysis including BLAST-like comparison with existing datasets as well as clustering and visualization to identify new gene families. Figure 2 shows initial results from analysis of 30,000 sequences with clusters identified and visualized using dimension reduction to map to three dimensions with Multi-dimensional scaling MDS [23]. The initial parts of the pipeline fit the MapReduce or many-task Cloud model but the latter stages involve parallel linear algebra.

State of the art MDS and clustering algorithms scale like $O(N^2)$ for N sequences; the total runtime for MDS and clustering is about 2 hours each on a 768 core commodity cluster obtaining a speedup of about 500 using a hybrid MPI-threading implementation on 24 core nodes. The initial steps can be run on clouds and include the calculation of a distance matrix of N(N-1)/2 independent elements. Million sequence problems of this type will challenge the largest clouds and the largest TeraGrid resources. Figure 3 looks at a related sequence assembly problem and compares performance of MapReduce (Hadoop, DryadLINQ) with and without virtual machines and the basic Amazon and Microsoft clouds. The execution times are similar (range is 30%) showing that this class of algorithm can be effectively run on many different infrastructures and it makes sense to consider the intrinsic advantages of clouds described above. In recent work we have looked hierarchical methods to reduce $O(N^2)$ execution time to O(NlogN) or O(N) and allow loosely-coupled cloud implementation with initial results on interpolation methods presented in [23].

One can study in [22, 25, 26] which applications run well on MapReduce and relate this to an old classification of Fox [27]. One finds that Pleasingly Parallel and a subset of what was called "Loosely Synchronous" applications run on MapReduce. However, current MapReduce addresses problems with only a single (or a "few") MapReduce iterations, whereas there are a large set of data parallel applications that involve many iterations and are not suitable for basic MapReduce. Such iterative algorithms include linear algebra and many data mining algorithms [28], and here we introduce the open source Twister to address these problems. Twister [25, 29] supports applications needing either a few iterations or many iterations using a subset

of MPI – reduction and broadcast operations and not the latency sensitive MPI point-to-point operations.

Twister [29] supports iterative computations of the type needed in clustering and MDS [23]. This programming paradigm is attractive as Twister supports all phases of the pipeline in Figure 1 with performance that is better or comparable to the basic MapReduce and on large enough problems similar to MPI for the iterative cases where basic MapReduce is inadequate. The current Twister system is just a prototype and further research will focus on scalability and fault tolerance. The key idea is to combine the fault tolerance and flexibility of MapReduce with the performance of MPI.

The current Twister, shown in Figure 4, is a distributed in-memory MapReduce runtime optimized for iterative MapReduce computations. It reads data from local disks of the worker nodes and handles the intermediate data in distributed memory of the worker nodes. All communication and data transfers are handled via a Publish/Subscribe messaging infrastructure. Twister comprises three main entities: (i) Twister Driver or Client that drives the entire MapReduce computation, (ii) Twister Daemon running on every worker node, and (iii) the broker network.

We present two representative results of our initial analysis of Twister [25, 29] in Figure 5 and 6.

We showed “doubly data parallel” (all pairs) application like pairwise distance calculation using Smith Waterman Gotoh algorithm can be implemented with Hadoop, Dyrad, and MPI [30]. Further, Figure 5 shows a classic MapReduce application already studied in Figure 2 and demonstrates that Twister will perform well in this limit, although its iterative extensions are not needed. We use the conventional efficiency defined as T(1)/(pT(p)), where T(p) is runtime on p cores. The results shown in Figure 5 were obtained using 744 cores (31 24-core nodes). Twister outperforms Hadoop because of its faster data communication mechanism and the lower overhead in the static task scheduling. Moreover, in Hadoop each map/reduce task is executed as a separate process, whereas Twister uses a hybrid approach in which the map/reduce tasks assigned to a given daemon are executed within one Java Virtual Machine (JVM). The lower efficiency in DryadLINQ shown in Figure 5 was mainly due to an inefficient task scheduling mechanism used in the initial academic release [21]. We also investigated Twister PageRank performance using a ClueWeb data set [31] collected in January 2009. We built the adjacency matrix using this data set and tested the page rank application using 32 8-core nodes. Figure 6 shows that Twister performs much better than Hadoop on this algorithm [32], which has the iterative structure, for which Twister was designed.

Conclusions

We have shown that MapReduce gives good performance for several applications and is comparable in performance to but easier to use [33] (from its high level support of parallelism) than conventional master-worker approaches, which are automated in Azure with its concept of roles. However many data mining steps cannot efficiently use MapReduce and we propose a hybrid cloud-cluster architecture to link MPI and

MapReduce components. We introduced the MapReduce extension Twister [25, 29] to allow a uniform programming paradigm across all processing steps in a pipeline typified by Figure 1.

Methods

We used three major computational infrastructures: Azure, Amazon and FutureGrid. FutureGrid offers a flexible environment for our rigorous benchmarking of virtual machine and "bare-metal" (non-VM) based approaches, and an early prototype of FutureGrid software was used in our initial work. We used four distinct parallel computing paradigms: the master-worker model, MPI, MapReduce and Twister.

The authors declare that they have no competing interests.

Authors' contributions

JQ participated in study of the hybrid Cloud and clustering computing pipeline model. JE, TG, JQ, HL, BZ, TLW carried out Smith Waterman Gotoh sequence alignment using Twister, DryadLINQ, Hadoop, and MPI. JYC, SHB and JE, contributed on parallel MDS and GTM using MPI and Twister. YR, ES and AH studied workflow and job scheduling on clusters. GF participated in study of Cloud and parallel computing research issues.

Acknowledgements

We appreciate Microsoft for their technical support. This work was made possible using the computing use grant provided by Amazon Web Services which is titled "Proof of concepts linking FutureGrid users to AWS". This work is partially funded by Microsoft "CRMC" grant and NIH Grant Number RC2HG005806-02. This document was developed with support from the National Science Foundation (NSF) under Grant No. 0910812 to Indiana University for "FutureGrid: An Experimental, High-Performance Grid Test-bed." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

1. Armbrust, M., Fox, Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M., **Above the Clouds: A Berkeley View of Cloud Computing.** Technical report, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf
2. Press Release: *Gartner's 2009 Hype Cycle Special Report Evaluates Maturity of 1,650 Technologies*. http://www.gartner.com/it/page.jsp?id=1124212
3. Cloud Computing Forum & Workshop. NIST Information Technology Laboratory, Washington DC: May 20 2010. http://www.nist.gov/itl/cloud.cfm
4. Nimbus Cloud Computing for Science. http://www.nimbusproject.org/
5. OpenNebula Open Source Toolkit for Cloud Computing. http://www.opennebula.org/
6. Sector and Sphere Data Intensive Cloud Computing Platform. http://sector.sourceforge.net/doc.html
7. Eucalyptus Open Source Cloud Software. http://open.eucalyptus.com/
8. FutureGrid Grid Testbed http://www.futuregrid.org
9. Magellan Cloud for Science. http://magellan.alcf.anl.gov/, http://www.nersc.gov/nusers/systems/magellan/
10. European Framework 7 project starting June 1 2010 VENUS-C Virtual multidisciplinary EnviroNments USing Cloud infrastructure.
11. Recordings of Presentations Cloud Futures 2010. Redmond WA, April 8-9 2010. http://research.microsoft.com/en-us/events/cloudfutures2010/videos.aspx
12. Lockheed Martin Cyber Security Alliance: Cloud Computing Whitepaper. April 2010. http://www.lockheedmartin.com/data/assets/isgs/documents/CloudComputingWhitePaper.pdf

13. Walker, E., **Benchmarking Amazon EC2 for High Performance Scientific Computing**. *USENIX*. vol. 33(5), Oct 2008. http://www.usenix.org/publications/login/2008-10/openpdfs/walker.pdf
14. Ekanayake, J., Qiu, X. H., Gunarathne, T., Beason, S., Fox, G. **High Performance Parallel Computing with Clouds and Cloud Technologies**. Book chapter to *Cloud Computing and Software Services: Theory and Techniques*, CRC Press (Taylor and Francis), ISBN-10: 1439803153, 2010. http://grids.ucs.indiana.edu/ptliupages/publications/cloud_handbook_final-with-diagrams.pdf
15. Evangelinos, C. and Hill, C. N., **Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2**. In *CCA08: Cloud Computing and its Applications*. 2008: Chicago ILL USA.
16. J. Lange, K. Pedretti, T. Hudson, Dinda, P., Cui, Z., Xia, L., Bridges, P., Gocke, A., Jaconette, S., Levenhagen, M., and Brightwell, R., **Palacios and Kitten: New High Performance Operating Systems For Scalable Virtualized and Native Supercomputing**. In *24th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2010)*. 2010: Atlanta, GA, USA.
17. Fox, G. White Paper: *FutureGrid Platform FGPlatform: Rationale and Possible Directions)*, 2010. http://grids.ucs.indiana.edu/ptliupages/publications/FGPlatform.docx
18. Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters**. *Commun*. ACM 51(1): 107-113.
19. Open source MapReduce Apache Hadoop. http://hadoop.apache.org/core/

20. Ekanayake, J., Gunarathne, T., Qiu, J., Fox, G., Beason, S., Choi, J. Y., Ruan, Y., Bae, S. H., Li, H. Technical Report: *Applicability of DryadLINQ to Scientific Applications*, January 30 2010. http://grids.ucs.indiana.edu/ptliupages/publications/DryadReport.pdf
21. Ekanayake, J., Balkir, A., Gunarathne, T., Fox, G., Poulain, C., Araujo, N., and Barga, R. **DryadLINQ for Scientific Analyses**. In *5th IEEE International Conference on e-Science*. 2009: Oxford UK.
22. Fox, G. C., Qiu, X. H., Beason, S., Choi, J. Y., Rho, M., Tang, H., Devadasan, N., Liu, G. **Biomedical Case Studies in Data Intensive Computing**. Keynote talk at The 1st International Conference on Cloud Computing (CloudCom 2009) at Beijing Jiaotong University, China December 1-4, 2009, Springer Verlag LNC 5931 "Cloud Computing" Jaatun, M., Zhao, G., Rong, C. (Eds), pp 2-18 (2009)
23. Bae, S. H., Choi, J. H., Qiu, J., Fox, J. **Dimension Reduction and Visualization of Large High-dimensional Data via Interpolation**. Proceedings of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010
24. Sammon, J.W. **A nonlinear mapping for data structure analysis**. *IEEE Trans. Computers*, 1969. **C-18**: p. 401-409.
25. Ekanayake, J. Computer Science PhD: *ARCHITECTURE AND PERFORMANCE OF RUNTIME ENVIRONMENTS FOR DATA INTENSIVE SCALABLE COMPUTING*. Bloomington: Indiana. 2010.
26. Fox, G. C. **Algorithms and Application for Grids and Clouds.** 22nd ACM Symposium on Parallelism in Algorithms and Architectures. 2010. http://grids.ucs.indiana.edu/ptliupages/presentations/SPAAJune14-10.pptx.

27. Fox, G. C., Williams, R.D. and Messina, P. C. **Parallel computing works!** 1994: Morgan Kaufmann Publishers, Inc. http://www.old-npac.org/copywrite/pcw/node278.html#SECTION00144000000000000000.
28. Chu, Cheng T., Sang K. Kim, Yi A. Lin, Yu, Y., Bradski, R., Ng, A. and Olukotun, K. **Map-Reduce for Machine Learning on Multicore**. In *NIPS*. 2006: MIT Press.
29. Twister Home page. http://www.iterativemapreduce.org/
30. Qiu, X., Ekanayake, J., Beason, S., Gunarathne, T., Fox, G., Barga, R., Gannon, D. **Cloud Technologies for Bioinformatics Applications**. Proceedings of the 2nd ACM Workshop on Many-Task Computing on Grids and Supercomputers (SC09), Portland, Oregon, November 16th, 2009 http://grids.ucs.indiana.edu/ptliupages/publications/MTAGSOct22-09A.pdf
31. The ClueWeb09 Dataset. http://boston.lti.cs.cmu.edu/Data/clueweb09/
32. Ekanayake, J., Li, Hui., Bingjing Zhang, B., Gunarathne, T., Bae, S.H., Qiu, J., Fox, G. **Twister: A Runtime for Iterative MapReduce**. Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010
33. Gunarathne,T., Wu,T.L., Qiu,J., and Fox, G., **Cloud Computing Paradigms for Pleasingly Parallel Biomedical Applications**. Proceedings of Emerging Computational Methods for the Life Sciences Workshop of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010

Figure 1 - Pipeline for analysis of metagenomics Data

Figure 2 - Results of 17 clusters for full sample using Sammon's version of MDS for visualization [24]

Figure 3 - Time to process a single biology sequence file (458 reads) per core with different frameworks[24]

Figure 4 - Current Twister Prototype

Figure 5 - Parallel Efficiency of the different parallel runtimes for the Smith Waterman Gotoh algorithm for distance computation

Figure 6 - Total running time for 20 iterations of PageRank algorithm on ClueWeb data with Twister and Hadoop on 256 cores