Twister4Azure: Parallel Data Analytics on Azure

Judy Qiu and Thilina Gunarathne

School of Informatics and Computing
Indiana University
{xqiu, tgunarat}@cs.indiana.edu

1. Motivation

Our research is to clarify which applications are best suited for Clouds; which require HPC and which can use both effectively and here we discuss a variety of application varying from simple pleasing parallel problems through sophisticated parallel data mining. This work is a collaboration between Cyberinfrastructure, bioinformatics and biology on large scale data intensive life sciences problems formulated as a pipeline of data storage, analysis, and visualization[1]. At the core of the pipeline, parallel programming paradigms such as MapReduce and MPI provide powerful large-scale data processing capabilities. The long-term goal is enabling cost effective and readily available analysis tool repository that removes the barrier of research in broader community -- making parallel high performance data analytics in the Cloud a reality.

2. Integrating Scientific Challenge: A Typical Bioinformatics Pipeline

New sequencing technology provides data samples with a throughput of 1 trillion base pairs per day and this rate will increase. A typical data pipeline is shown in Fig. 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, dimension reduction and visualization to identify new gene families[2]. The initial parts of the pipeline fit the MapReduce (e.g. Hadoop) or many-task Cloud (e.g. Azure) model but the latter stages involve parallel linear algebra for the data mining. It is highly desirable to simplify the construction of distributed sequence analysis pipelines with a unified programming model, which motivated us to design and implement Twister4Azure. Twister[3, 4] and Twister4Azure[5] interpolate between MPI and MapReduce and, suitably configured, can mimic their characteristics, and, more interestingly, can be positioned as a programming model that has the performance of MPI and the fault tolerance and dynamic flexibility of the original MapReduce.

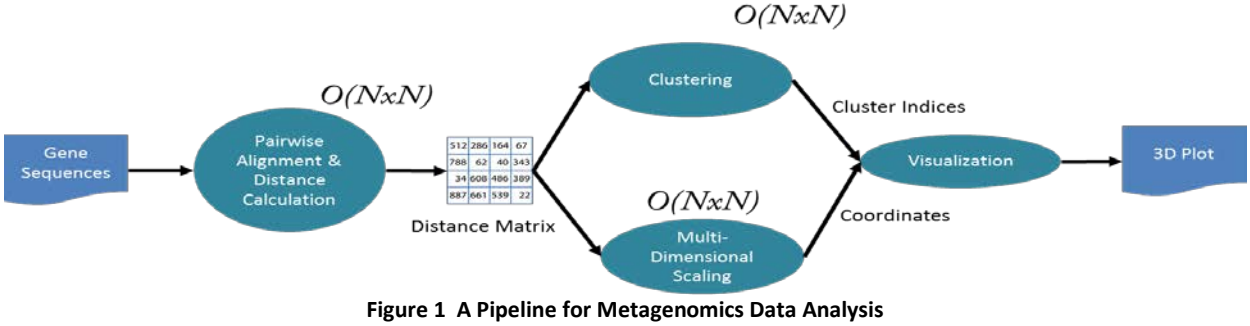

Figure 1 A Pipeline for Metagenomics Data Analysis

3. Technologies and Applications

Our applications can be classified into three main categories based on their execution pattern, namely pleasingly parallel computations, MapReduce computations and iterative MapReduce computations. Twister4Azure distributed decentralized iterative MapReduce runtime for Windows Azure Cloud, which is the successor to MRRoles4Azure (Fig. 2) MapReduce framework and the classic Cloud pleasingly parallel framework [6], was used as the distributed cloud data processing framework for our scientific computations. Twister4Azure (Fig. 3) extends the familiar, easy-to-use MapReduce programming model with iterative extensions, enabling a wide

array of large scale iterative as well as non-iterative data analysis and scientific applications to utilize Azure platform easily and efficiently in a fault-tolerant manner. It supports all three categories of applications.

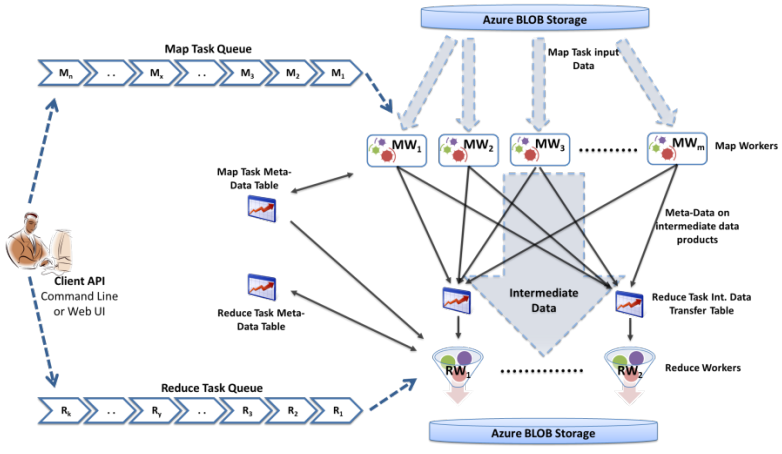

Figure 2 MRRoles4Azure Architecture

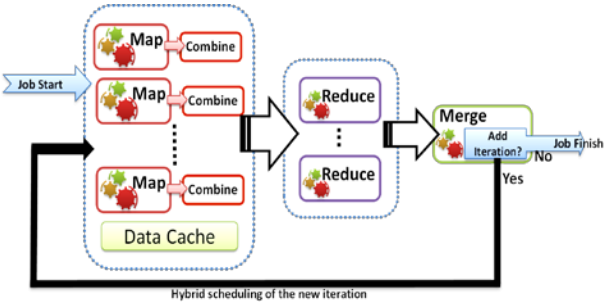

Figure 3 Twister4Azure programming model

Twister4Azure utilizes the eventually-consistent, high-latency Azure cloud services effectively to deliver performance comparable to traditional (non-iterative) MapReduce runtimes and outperforms MapReduce for iterative computations. Twister4Azure provides multi-level caching of data across iterations as well as among workers running on the same compute instance and utilizes a novel hybrid task scheduling mechanism to perform cache aware scheduling with minimal overhead. Twister4Azure also supports data broadcasting, collective communication primitives as well as invoking of multiple MapRedue applications inside an iteration.

3.1 Pleasingly parallel Computations

We performed Cap3 sequence assembly (Fig. 4), BLAST+ sequence search and dimension reduction interpolation computations on Azure using this framework. The performance and scalability are comparable to traditional MapReduce runtimes [6]. For Cap3, we assembled up to 4096 FASTA files (each containing 458 reads) in less than one hour using 128 Azure small instances with a cost of around 16$. With BLAST+, the execution of 76800 queries using 16 Azure large instances was less than one hour with a cost of around 12$.

3.2 MapReduce Type Computations

We performed Smith Waterman-GOTOH (SWG) pairwise sequence alignment computations on Azure [3][4] with performance and scalability comparable to the traditional MapReduce frameworks running on traditional clusters (Fig. 5). We were able to perform up to 123 million sequence alignments using 192 Azure small instances with a cost of around 25$, which was less than the cost it took to run using Amazon ElasticMapReduce.

3.3 Iterative MapReduce Type Computations

The third and most important category of computation is the iterative MapReduce type applications. These include majority of data mining, machine learning, dimension reduction, clustering and many more applications. We performed KMeans Clustering (Fig. 6) and Multi-Dimensional Scaling (MDS) (Fig. 7) scientific iterative MapReduce computations on Azure cloud. MDS consists of two MapReduce computations (`BCCalc` and `StressCalc`) per iteration and contains parallel linear algebra computations as its core. Fig. 6 shows performance measurements of Azure are comparable or a factor of 2 to 4 better than those of the traditional MapReduce runtimes deployed on up to 256 instances and for jobs with tens of thousands of tasks.

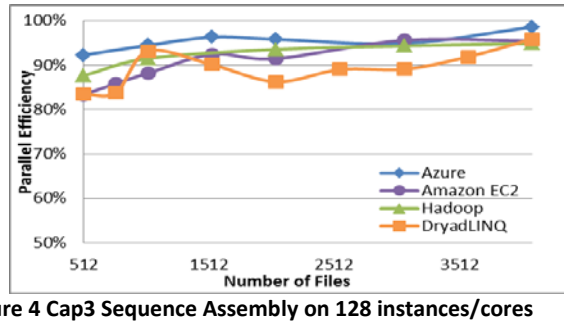

Figure 4 Cap3 Sequence Assembly on 128 instances/cores

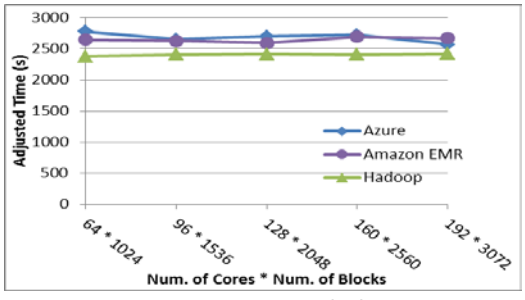

Figure 5 SWG Pairwise Distance Calculation Performance

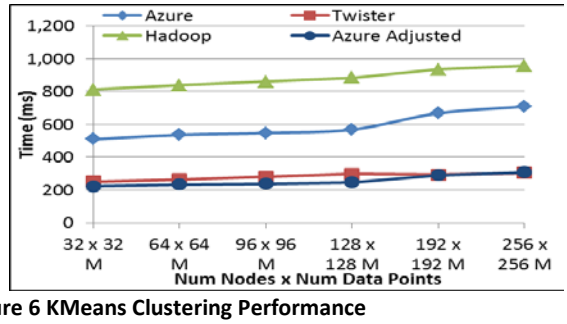

Figure 6 KMeans Clustering Performance

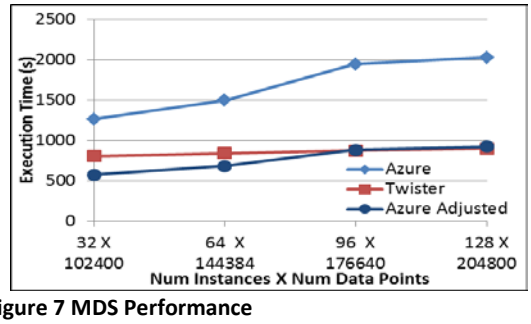

Figure 7 MDS Performance

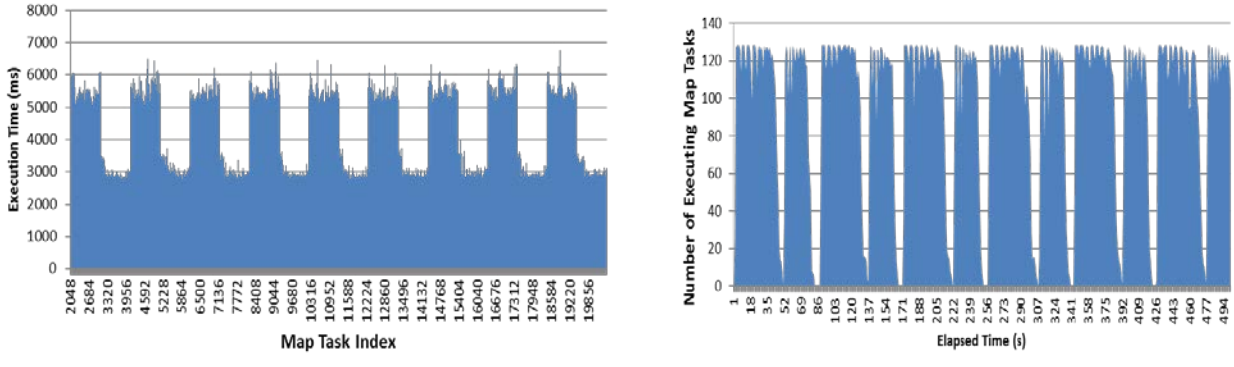

Figure 8 Using AllGather primitive together with MMF caching

4. Performance Considerations for Large Scale Iterative Applications on Azure

It is possible to optimize iterative MapReduce computations by caching the loop-invariant input data across the iterations. We use the Azure Blob storage as the input data storage for Twister4Azure computations. Twister4Azure supports local instance (disk) storage caching as the simplest form of data caching. Local storage caching allows the subsequent iterations (or different applications or tasks in the same iteration) to reuse the input data from the local storage rather than fetching them from the Azure Blob Storage. This resulted in speedups of more than 50 % over a non-cached MDS computation of the sample use case.

Twister4Azure also supports the 'in-memory caching' of the loop-invariant data across iterations. With in-memory caching, Twister4Azure fetch the data from the Azure Blob storage, parse and load them in to the memory during the first iteration. After the first iteration, these data products remain in memory throughout the course of the computation for reuse by the subsequent iterations, eliminating the overhead of reading and parsing data from disk. In-memory caching improved the average run time of the `BCCalc` map task by approximately 36% and the total run time by approximately 22% over disk based caching. Twister4Azure performs cache-invalidation for in-memory cache using Least Recently Used (LRU) policy. In a typical

Twister4Azure computation, the loop-invariant input data stays in the in-memory cache throughout the computation, while Twister4Azure caching policy will evict broadcast data for iterations after it is used.

When using in-memory caching, we started to notice occasional non-deterministic fluctuations of the Map function execution times in some of the tasks. These slow tasks, even though few, affect the performance of the computation significantly because the execution time of a whole iteration is of course dependent on the slowest task of the iteration. Even though Twister4Azure supports duplicate execution of the slow tasks, duplicate tasks for non-initial iterations are often more costlier than the total execution time of a slow task that uses data from a cache, as the duplicate task would have to fetch the data from the Azure Blob Storage. We were able to narrow down the cause of the fluctuation anomaly to the use of large amount of memory, including the in-memory data cache, within a single .NET process. We switched to use of memory-mapped files, which can be shared across multiple processes and can be used to facilitate inter-process communication.

A key idea of our research is a Map-Collective model supported by Twister4Azure where each architecture is supported by an appropriate implementation of each collective operation. Fig. 8 shows the results for MDS using an AllGather iterative MapReduce Collective primitive similar to the MPI AllGather communication primitive. The left column labels Execution time of tasks in each iteration. The taller bars represent the MDSBCCalc computation, while the shorter bars represent the MDSStressCalc computation. A pair of `BCCalc` and `StressCalc` bars represents an iteration. Number of active map tasks of the computation at a given time (A 500 second view from the 3rd iteration onwards). The wider bars represent `BCCalc` computations, while the narrower bars represent `StressCalc` computations. The gaps between the computations represent the overhead of task scheduling, reduce task execution, and merge task execution and data broadcasting.

The AllGather primitive broadcasts the Map Task outputs to all the computational nodes, and assembles them together in the recipient nodes and schedules the next iteration or the application. Usage of the AllGather primitive in MDS BCCalc computation eliminates the need for reduce, merge and the broadcasting steps in that particular computatation. In addition to improving the performance, this primitive also improves the usability of the system as it eliminates the overhead of implementing reduce and/or merge functions. Communication primitives also allow us to optimize the given operation transparently to the users where Fig. 8 presents an execution trace of a computation that utilized both memory mapped files and the AllGather primitive.

5. Related Work

CloudMapReduce [7] for Amazon Web Services (AWS) and Google AppEngine MapReduce [8] follow an architecture similar to MRRoles4Azure, in which they utilize the cloud services as the building blocks. Amazon ElasticMapReduce [9] offers Apache Hadoop as a hosted service on the Amazon AWS cloud environment. However, none of them support iterative MapReduce. Spark [10] is a framework implemented using Scala to support MapReduce like operations to query and process read-only data collections, while supporting in-memory caching and re-use of data products. Azure HPC scheduler is a new Azure feature that enables the users to launch and manage high-performance computing in the Azure environment. Azure HPC scheduler supports parametric sweeps, Message Passing Interface (MPI) and LINQ to HPC applications. Microsoft Daytona [11] is a recently announced iterative MapReduce runtime developed by Microsoft Research for Microsoft Azure Cloud Platform. It builds on some of the ideas of the earlier Twister system. Haloop [12] extends Apache Hadoop to support iterative applications and supports caching of loop-invariant data as well as loop-aware scheduling.

5. Conclusion and Future Work

We have developed Twister4Azure, a novel iterative MapReduce distributed computing runtime for Azure cloud. We implemented four significant scientific applications using Twister4Azure – MDS, Kmeans Clustering, SWG sequence alignment and BLAST+. Twister4Azure enables the users to easily and efficiently perform large-scale iterative data analysis for scientific applications on a commercial cloud platform. Twister4Azure and Java HPC Twister are key to our cross platform new programming paradigm supporting large scale data analytics.

The overall major challenge for this research is building a system capable of handling the incredible increases in dataset sizes while solving the technical challenges of portability with scaling performance and fault tolerance using an attractive powerful programming model. Further, these challenges must be met for both computation and storage. Cloud enables persistent storage like Azure Blob. MapReduce leverages the possibility of collocating data and compute and provides more flexibility in co-locating data and computing.

Acknowledgements: This work was made possible using the computing usage grant provided by Microsoft Azure Cloud. The work reported in this document is partially funded by NIH Grant number RC2HG005806-02. We would also like to appreciate Persistent Systems for their fellowship supporting Thilina Gunarathne.

References

1. Jaliya Ekanayake, Thilina Gunarathne, and Judy Qiu, *Cloud Technologies for Bioinformatics Applications.* Journal of IEEE Transactions on Parallel and Distributed Systems, 2011. 22(6): p. 998-1011. DOI:10.1109/TPDS.2010.178 http://grids.ucs.indiana.edu/ptliupages/publications/BioCloud_TPDS_Journal_Jan4_2010.pdf
2. Judy Qiu, Thilina Gunarathne, Jaliya Ekanayake, Jong Youl Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Yang Ryan, Saliya Ekanayake, Tak-Lon Wu, Scott Beason, Adam Hughes, and Geoffrey Fox, *Hybrid Cloud and Cluster Computing Paradigms for Life Science Applications*, in *11th Annual Bioinformatics Open Source Conference BOSC 2010*. July 9-10, 2010. Boston. http://grids.ucs.indiana.edu/ptliupages/publications/HybridCloudandClusterComputingParadigmsforLifeScienceApplications.pdf.
3. J.Ekanayake, H.Li, B.Zhang, T.Gunarathne, S.Bae, J.Qiu, and G.Fox, *Twister: A Runtime for iterative MapReduce*, in *Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference June 20-25, 2010*. 2010, ACM. Chicago, Illinois. http://grids.ucs.indiana.edu/ptliupages/publications/hpdc-camera-ready-submission.pdf.
4. SALSA Group. *Iterative MapReduce*. 2010 [accessed 2010 November 7]; Twister Home Page Available from: http://www.iterativeMapReduce.org/.
5. Thilina Gunarathne, Tak-Lon Wu, Judy Qiu, and Geoffrey Fox, *MapReduce in the Clouds for Science*, in *CloudCom 2010*. November 30-December 3, 2010. IUPUI Conference Center Indianapolis. http://grids.ucs.indiana.edu/ptliupages/publications/CloudCom2010-MapReduceintheClouds.pdf.
6. Gunarathne, T., T.-L. Wu, J.Y. Choi, S.-H. Bae, and J. Qiu, *Cloud computing paradigms for pleasingly parallel biomedical applications.* Concurrency and Computation: Practice and Experience, 2011. DOI:10.1002/cpe.1780. http://dx.doi.org/10.1002/cpe.1780
7. *CloudMapReduce*. [accessed 2010 April 20]; Available from: http://code.google.com/p/cloudMapReduce/.
8. *AppEngine MapReduce*. July 25th 2011]; Available from: http://code.google.com/p/appengine-MapReduce.
9. *Amazon Web Services*. [accessed 2010 Nov 10]; Available from: http://aws.amazon.com/.
10. Zaharia, M., M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica, *Spark: Cluster Computing with Working Sets*, in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10)*. June 22, 2010. Boston. http://www.cs.berkeley.edu/~franklin/Papers/hotcloud.pdf.
11. *Microsoft Daytona*. Retrieved Feb 1, 2012, from : http://research.microsoft.com/en-us/projects/daytona/. http://research.microsoft.com/en-us/projects/daytona/.
12. Bu, Y., B. Howe, M. Balazinska, and M.D. Ernst, *HaLoop: Efficient Iterative Data Processing on Large Clusters*, in *The 36th International Conference on Very Large Data Bases*. September 13-17, 2010, VLDB Endowment: Vol. 3. Singapore. http://www.ics.uci.edu/~yingyib/papers/HaLoop_camera_ready.pdf.