

# Iterative MapReduce for Azure Cloud

Thilina Gunarathne, Judy Qiu, Geoffrey Fox

School of Informatics and Computing / Pervasive Technology Institute  
Indiana University, Bloomington.  
{tgunarat, xqiu, gcf}@indiana.edu

## ABSTRACT

MapReduce distributed data processing architecture has become the de-facto data-intensive analysis mechanism in compute clouds and in commodity clusters, mainly due to its excellent fault tolerance features, scalability, ease of use and the simpler programming model. MapReduceRoles for Azure (MR4Azure) is a decentralized, dynamically scalable MapReduce runtime we developed for Windows Azure Cloud platform using Microsoft Azure cloud infrastructure services as the building blocks. This paper presents Twister4Azure, which adds support for optimized iterative MapReduce computations to MR4Azure, based on the concepts of Twister Iterative MapReduce framework. Twister4Azure enables a wide array of large scale iterative data analysis and scientific applications to utilize Azure platform easily and efficiently, while preserving the fault tolerance, decentralized and dynamic scheduling features of MR4Azure. Both MR4Azure and Twister4Azure take advantage of the scalability, high availability and the distributed nature of cloud infrastructure services to avoid single point of failures, bandwidth bottlenecks and management overheads.

## Keywords

MapReduce, Azure, Iterative

## 1. INTRODUCTION

Cloud computing platforms offer easily accessible and horizontally scalable compute power over the wire, which can be effectively utilized for large scale computational analysis. While cloud platforms can be used with relatively very low management overhead and very low startup costs, they also present unique challenges for the traditional computational frameworks such as MPI. MapReduce distributed computing architecture, introduced by Google, can be seen as an emerging data intensive analysis architecture, which allows the users to harness the power of cloud computing platforms more effectively and easily than their traditional counterparts.

The Microsoft Azure is a cloud computing platform offering on demand computing services such as Windows Azure Compute, Storage BLOB service, Queue service, Table service etc. Azure Compute is a platform as a service infrastructure allowing the users to lease hourly charged virtual machine instances in the form of various Roles (eg: Worker Role, Web Role, VM Role). The Azure storage queue is an eventual consistent, reliable, scalable and distributed message queue service, which is ideal for small, short-lived, transient messages. The Azure Storage BLOB service provides a distributed storage service where users can store and retrieve any type data as a BLOB through a web services interface. Azure Storage Table service provides scalable non-relational highly available structured data storage. However Azure platform currently do not offer a distributed computing framework, other than the simple queue based model. Goal of MR4Azure and Twister4Azure is to facilitate the efficient execution of MapReduce and iterative MapReduce applications in the Microsoft Azure cloud platform.

## 2. MAPREDUCE ROLES for AZURE

MR4Azure is a distributed decentralized MapReduce runtime for Windows Azure cloud platform that utilizes Azure cloud infrastructure services. MR4Azure overcomes the latencies of cloud services by using sufficiently coarser grained map and reduce tasks. It overcomes the eventual data availability of cloud storage services through re-trying and by explicitly designing the system to not rely on the immediate availability of data across all distributed workers. MR4Azure uses Azure Queues for map and reduce task scheduling, Azure Tables for metadata storage and monitoring data storage, Azure BLOB storage for data storage (input, output and intermediate) and the Window Azure Compute worker roles to perform the computations.

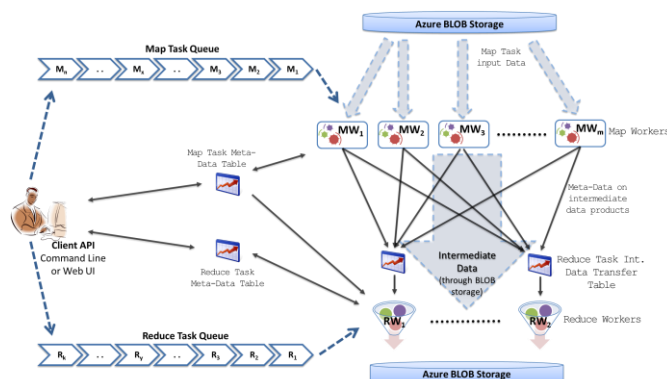
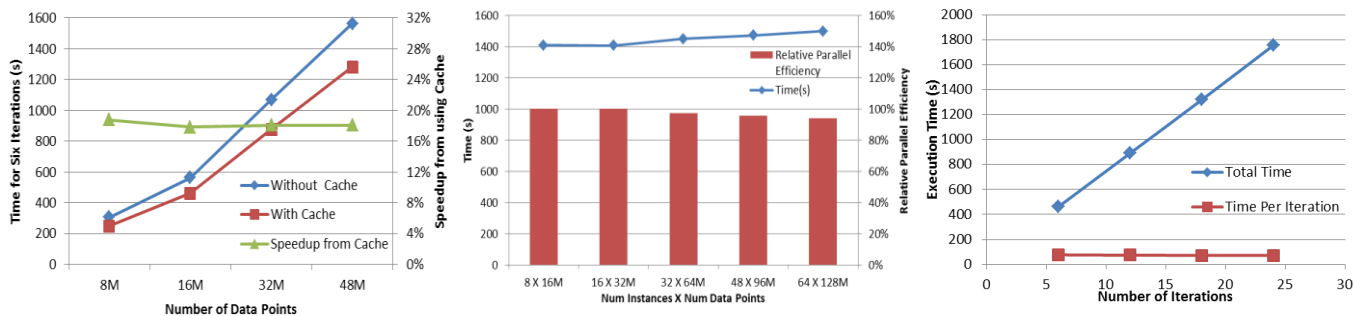


Figure 1 MapReduce for Azure Architecture

In order to withstand the brittleness of cloud infrastructures and to avoid possible single point of failures, MR4Azure was designed as a decentralized control architecture which does not rely on a central coordinator or a client side driver. MR4Azure provides users with the capability to dynamically scale up/down the number of computing resources. The map and reduce tasks of the MR4Azure runtime are dynamically scheduled using global queues achieving natural load balancing given sufficient amount of tasks. MR4Azure handles task failures and slower tasks through re-execution and duplicate executions. MapReduce architecture requires the reduce tasks to ensure the receipt of all the intermediate data products from Map tasks before starting the reduce phase. Since ensuring such a collective decision is not possible with the direct use of eventual consistent tables, MR4Azure uses additional data structures on top of Azure Tables for this purpose. In Gunarathne et al.[1] we show that MR4Azure performs comparably to the other popular MapReduce runtimes.

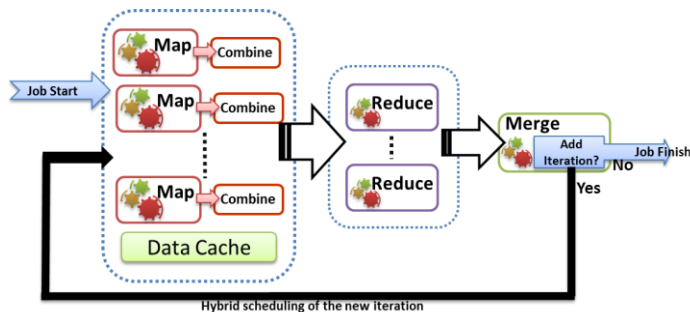
## 3. TWISTER for AZURE

There exists many data analysis as well as scientific computation algorithms that rely on iterative computations, where each iterative step can be easily specified as a MapReduce computation. Twister4Azure extends the MR4Azure to support such iterative MapReduce executions, drawing lessons from Twister [2] iterative MapReduce framework.

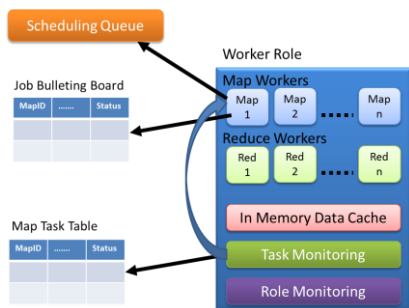


**Figure 2. KMeans iterative MapReduce performance using (a) 6 iterations with/without data caching, 16 instances (b) Scaling Speedup - 10 iterations with caching (c) Increasing number of iterations using 16M data points with data caching, 16 instances**

Twister4Azure introduces an additional “Merge” step to MapReduce programming mode, that executes after the reduce phase (map->reduce->merge). Merge task receives all the reduce task outputs as its input. Since Twister4Azure do not have an always connected client driver due to reliability reasons, the decision to continue with more iteration or to finish the computation after the current iteration needs to be made in the Merge step. Iterative computations often rely on a set of static data that remain fixed across iterations and a set of transient dynamic data between iterations. Twister4Azure introduces an in-memory DataCache to store the reusable static data across the iterations, avoiding the downloading and parsing cost of such data from Azure BLOB storage for every iteration. Each worker role will have one managed DataCache with a given memory limit.



**Figure 3 Twister4Azure Computation Flow**



**Figure 4 Twister4Azure Map Worker Architecture**

Since Twister4Azure do not have a central coordinator which can utilize the knowledge about cached data products to assign tasks to workers, scheduling tasks to take advantage of data caching presents a significant challenge. At the same time, it’s desirable to preserve the dynamic scheduling, fault tolerance and dynamic scalable features of MR4Azure in the new scheduling mechanism. In order to address these issues, Twister4Azure utilizes a hybrid scheduling approach using a combination of Queues and Azure Tables. The first iteration of Twister4Azure would be identical to

MR4Azure and will get scheduled only through Azure queues. Twister4Azure uses a special table as a “BulletinBoard”, where the tasks are advertised from second iteration onwards. Map Workers first query this bulletin board to identify any intersection between the data items they have in their DataCache vs the data items needed for the advertised tasks. With this mechanism the static data for iterative MapReduce computations will get reused from the second iteration onwards. In the meantime idle workers (newly joined or a worker who has finished processing all the tasks for the cached data) will be able to pick up tasks directly from the queue and will use the AzureTables and the monitoring infrastructure to identify the tasks that are already processed. This also ensures that Twister4Azure retains the fault tolerance features of MR4Azure.

Figure 2 presents the performance of KMeans clustering Twister4Azure implementation using Azure small instances. Each data set contained 8-128 million 20-dimensional data points. Cached KMeans computation showed ~18% speedup over non-cached computation, when used with this particular data set. We expect the speedups to be even more significant for more data intensive iterative computations. Figure 2 (c) shows that the Twister4Azure was able to sustain the performance with increasing number of iterations.

## 4. CONCLUSION

Twister4Azure and MR4Azure provide MapReduce distributed computing runtimes for the Windows Azure cloud platform. Twister4Azure presents a decentralized iterative extension to MapReduce distributed computing model, enabling the users to easily and efficiently perform large scale iterative data analysis and scientific computations on Azure cloud platform. Twister4Azure utilizes a novel hybrid scheduling mechanism based on Azure Tables and Azure Queues to provide the caching of static data across iterations in iterative computations. Twister4Azure and MR4Azure can also be perceived as two examples of architectures that utilize cloud infrastructure services effectively to deliver robust and efficient applications.

MR4Azure and an alpha version of Twister4Azure are available for download at <http://salsahpc.indiana.edu/twister4azure>

## 5. REFERENCES

- [1] Gunarathne, T., Wu, T.L., Qiu, J., and Fox, G.C. 2010. MapReduce in the Clouds for Science. In *Proceedings of CloudCom 2010 Conference* (Indianapolis, December 2010)
- [2] J.Ekanayake, H.Li, B.Zhang *et al.*, 2010. Twister: A Runtime for iterative MapReduce, in *Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference* (Chicago, June 2010)