Classical and Iterative MapReduce on Azure

Cloud Futures Workshop

Microsoft Conference Center Building 33, Redmond, Washington

June 2 2011

Geoffrey Fox

gcf@indiana.edu

http://www.infomall.org http://www.salsahpc.org

Director, Digital Science Center, Pervasive Technology Institute

Associate Dean for Research and Graduate Studies, School of Informatics and Computing

Indiana University Bloomington

Work with Thilina Gunarathne, Judy Qiu

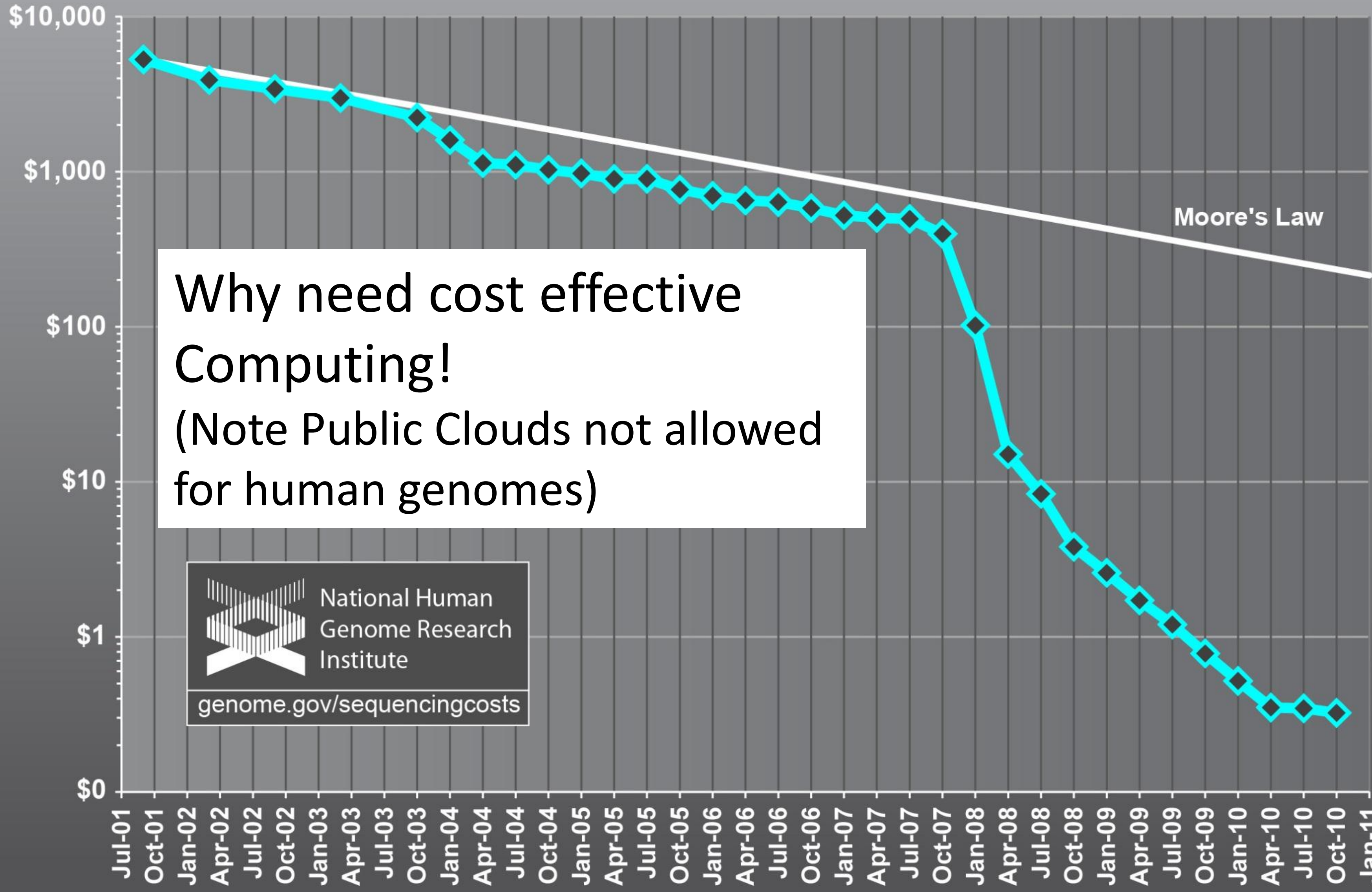
Twister introduced in Jaliya Ekanayake's PhD Thesis

https://portal.futuregrid.org

Simple Assumptions

- Clouds may not be suitable for everything but they are suitable for majority of data intensive applications
  - Solving partial differential equations on 100,000 cores probably needs classic MPI engines
- Cost effectiveness, elasticity and quality programming model will drive use of clouds in many areas such as genomics
- Need to solve issues of
  - Security-privacy-trust for sensitive data
  - How to store data – “data parallel file systems” (HDFS) or classic HPC approach with shared file systems with Lustre etc.
- Programming model which is likely to be **MapReduce** based initially
  - Look at high level languages
  - Compare with databases (SciDB?)
  - **Must support iteration for many problems**

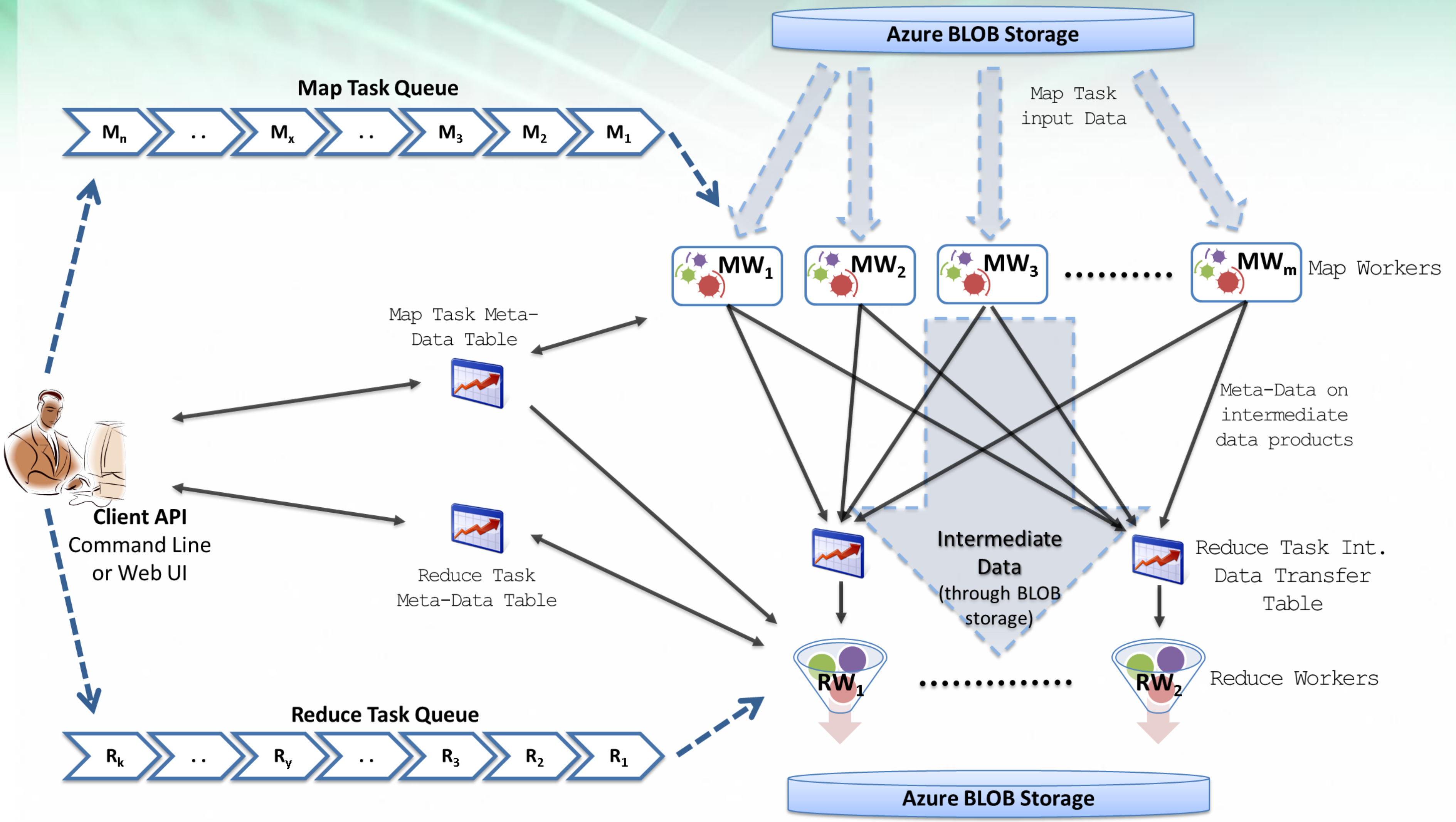
Cost per Megabase of DNA Sequence



Why need cost effective Computing!

(Note Public Clouds not allowed for human genomes)

MapReduceRoles4Azure Architecture



Azure Queues for scheduling, Tables to store meta-data and monitoring data, Blobs for input/output/intermediate data storage.

Future Grid https://portal.futuregrid.org

SALSA HPC

MapReduceRoles4Azure

- Use distributed, highly scalable and highly available cloud services as the building blocks.
  - Azure Queues for task scheduling.
  - Azure Blob storage for input, output and intermediate data storage.
  - Azure Tables for meta-data storage and monitoring
- Utilize eventually-consistent , high-latency cloud services effectively to deliver performance comparable to traditional MapReduce runtimes.
- Minimal management and maintenance overhead
- Supports dynamically scaling up and down of the compute resources.
- MapReduce fault tolerance
- http://salsahpc.indiana.edu/mapreduceroles4azure/

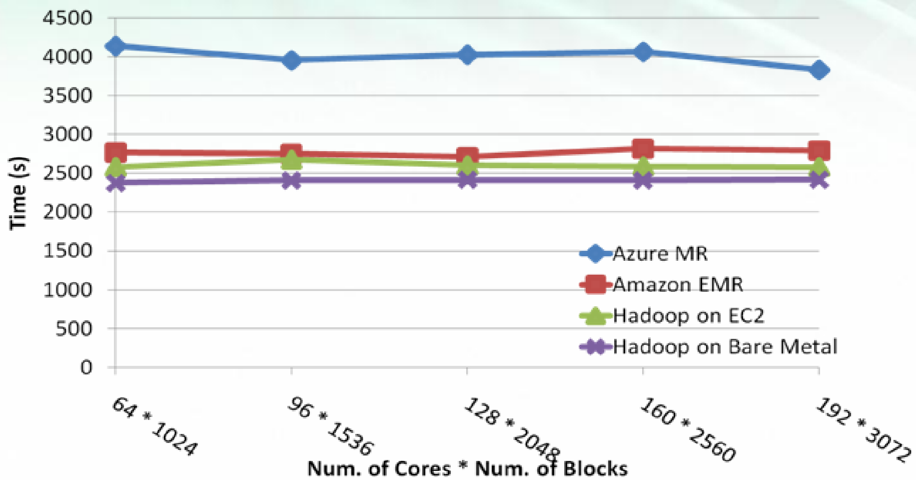
MapReduceRoles4Azure Performance

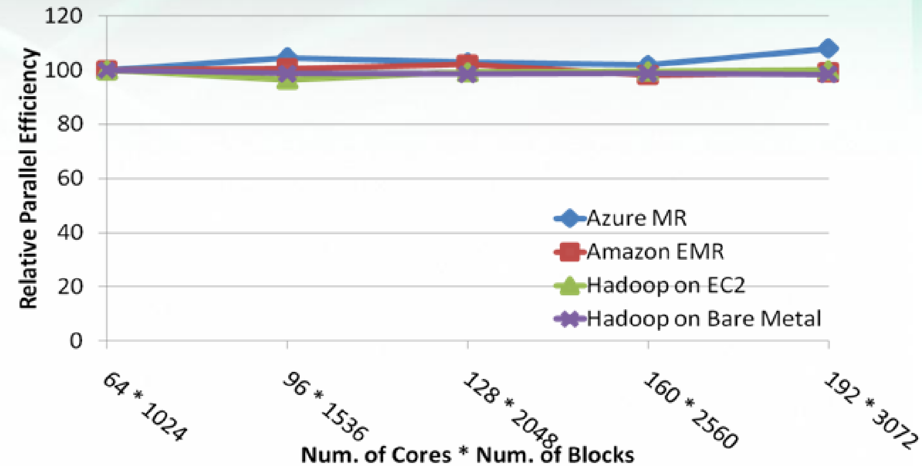
- Parallel efficiency

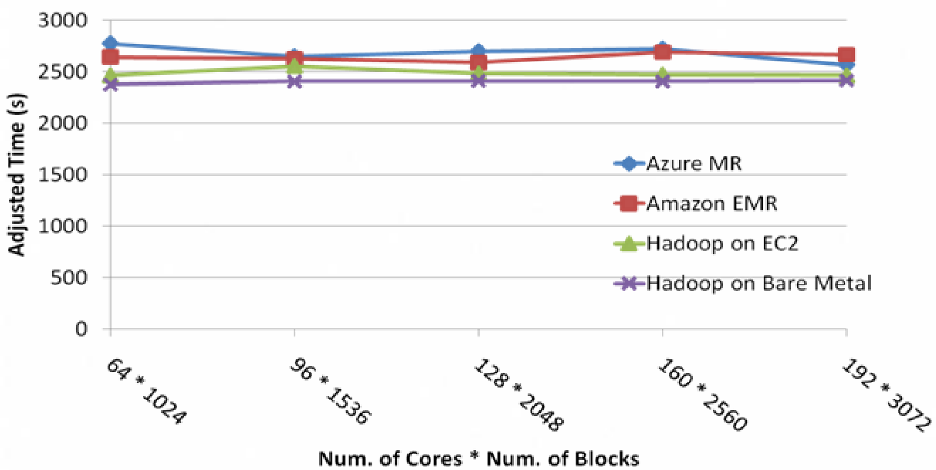
$$Parallel\ Efficiency\ (Ep) = \frac{T(1)}{pT(\rho)}$$

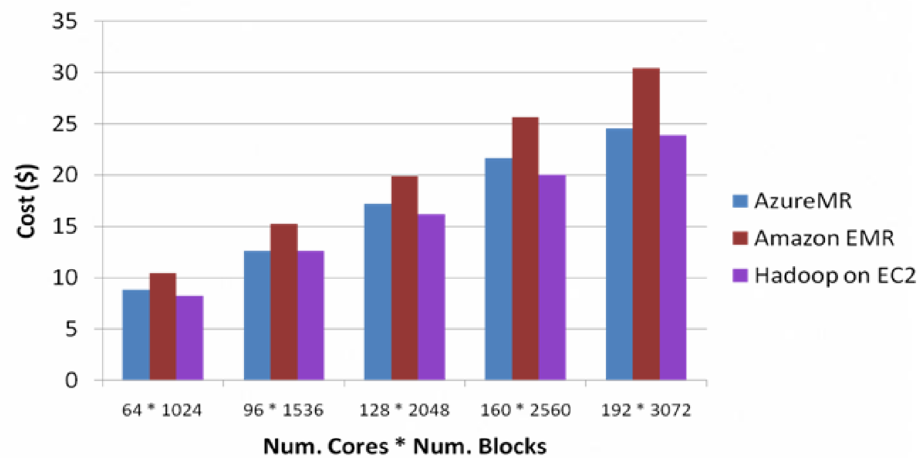
- AzureMapReduce
  - Azure small instances – Single Core (1.7 GB memory)
- Hadoop Bare Metal -IBM iDataplex cluster
  - Two quad-core CPUs (Xeon 2.33GHz),16 GB memory, Gigabit Ethernet per node
- EMR & Hadoop on EC2
  - Cap3 – HighCPU Extra Large instances (8 Cores, 20 CU, 7GB memory per instance)
  - SWG – Extra Large Instances (4 Cores, 8 CU, 15GB memory per instance)

SWG Sequence Alignment Performance







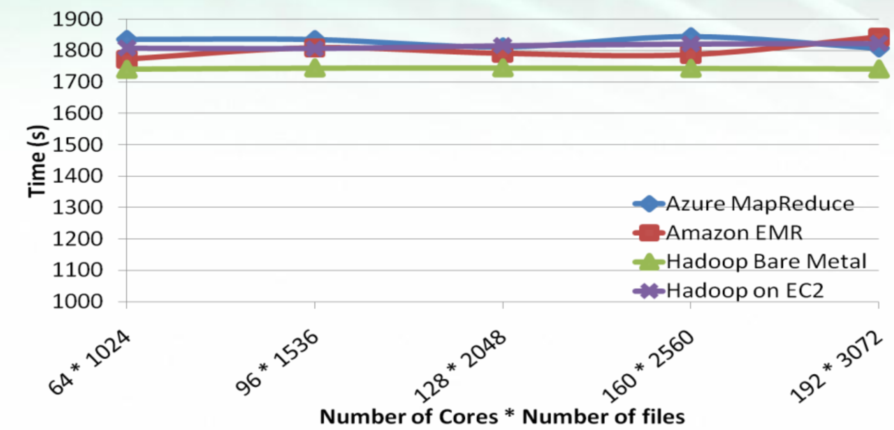


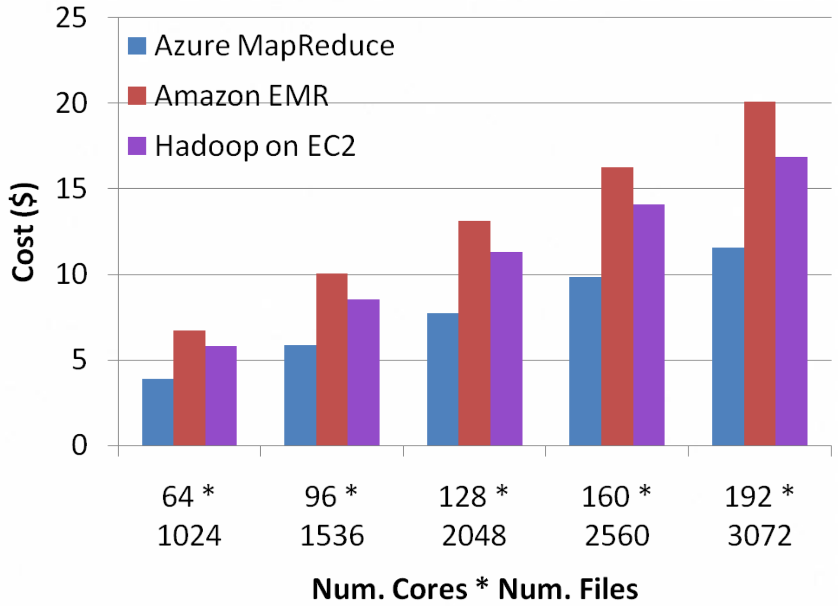
Smith-Waterman-GOTOH to calculate all-pairs dissimilarity

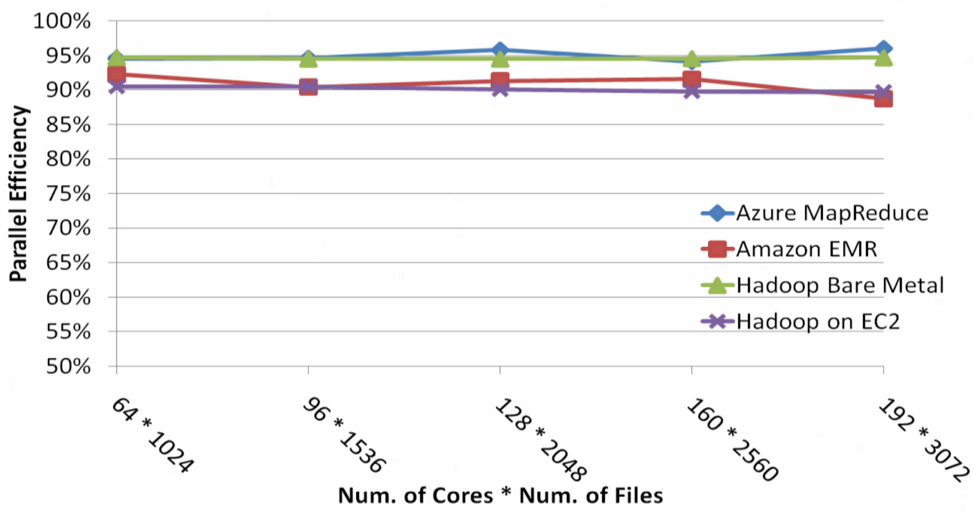
Future Grid https://portal.futuregrid.org

SALSA HPC

CAP3 Sequence Assembly Performance





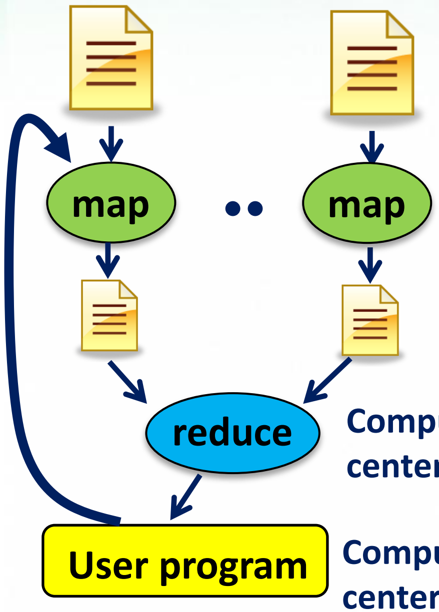


Future Grid https://portal.futuregrid.org

SALSA HPC

Why Iterative MapReduce? K-Means Clustering

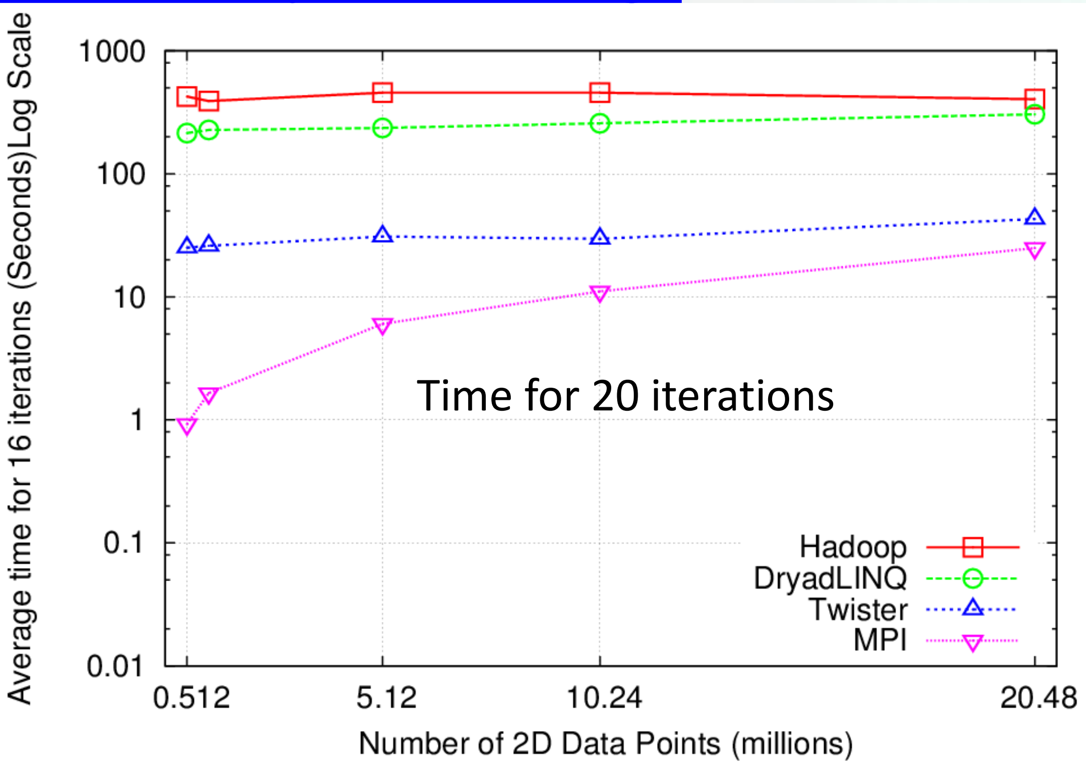
http://www.iterativemapreduce.org/



Compute the distance to each data point from each cluster center and assign points to cluster centers

Compute new cluster centers

Compute new cluster centers

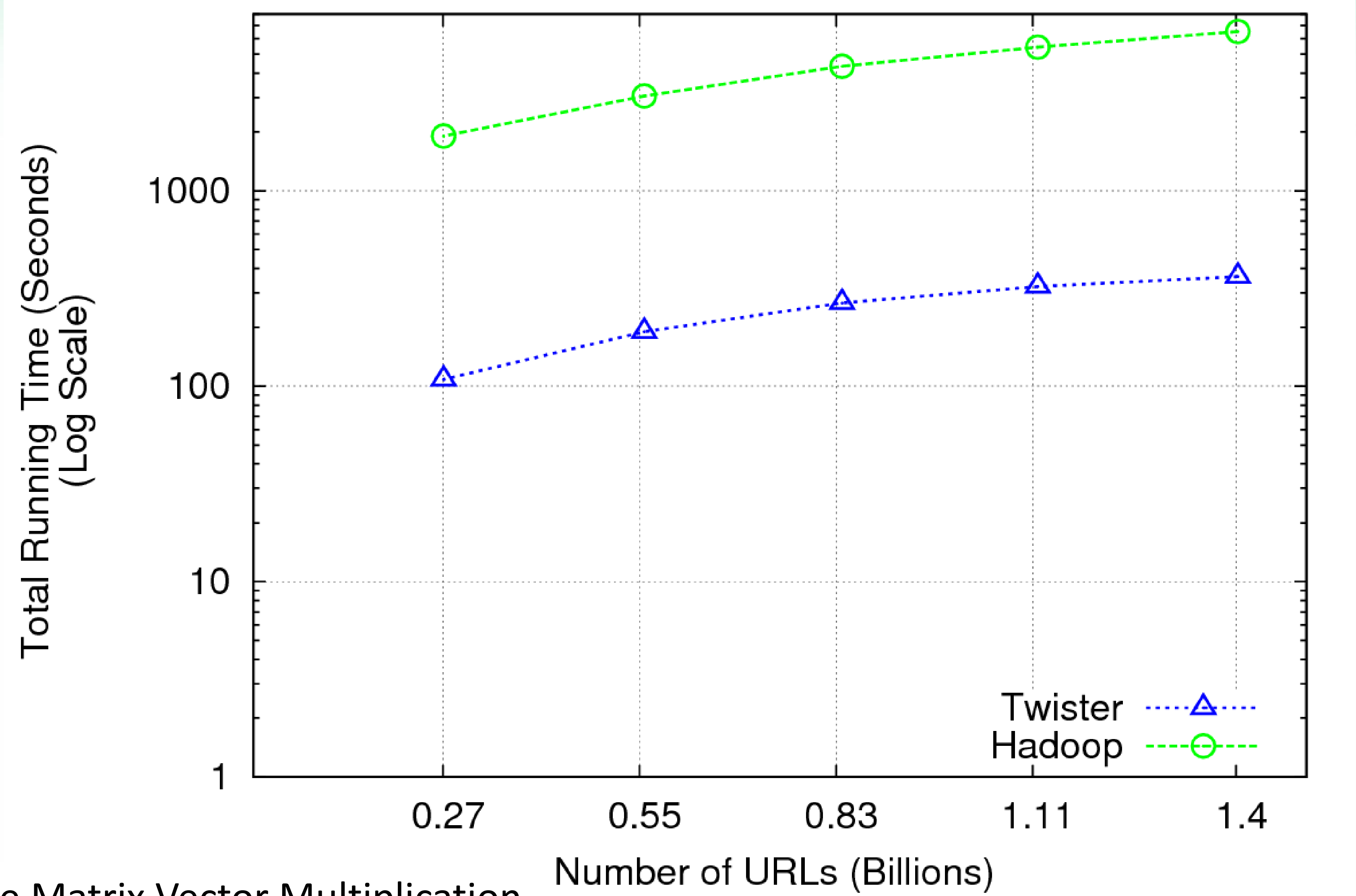


- Iteratively refining operation
- Typical MapReduce runtimes incur extremely high overheads
  - New maps/reducers/vertices in every iteration
  - File system based communication
- Long running tasks and faster communication in Twister enables it to perform close to MPI

Future Grid https://portal.futuregrid.org

SALSA HPC

Performance of Pagerank using ClueWeb Data (Time for 20 iterations) using 32 nodes (256 CPU cores)

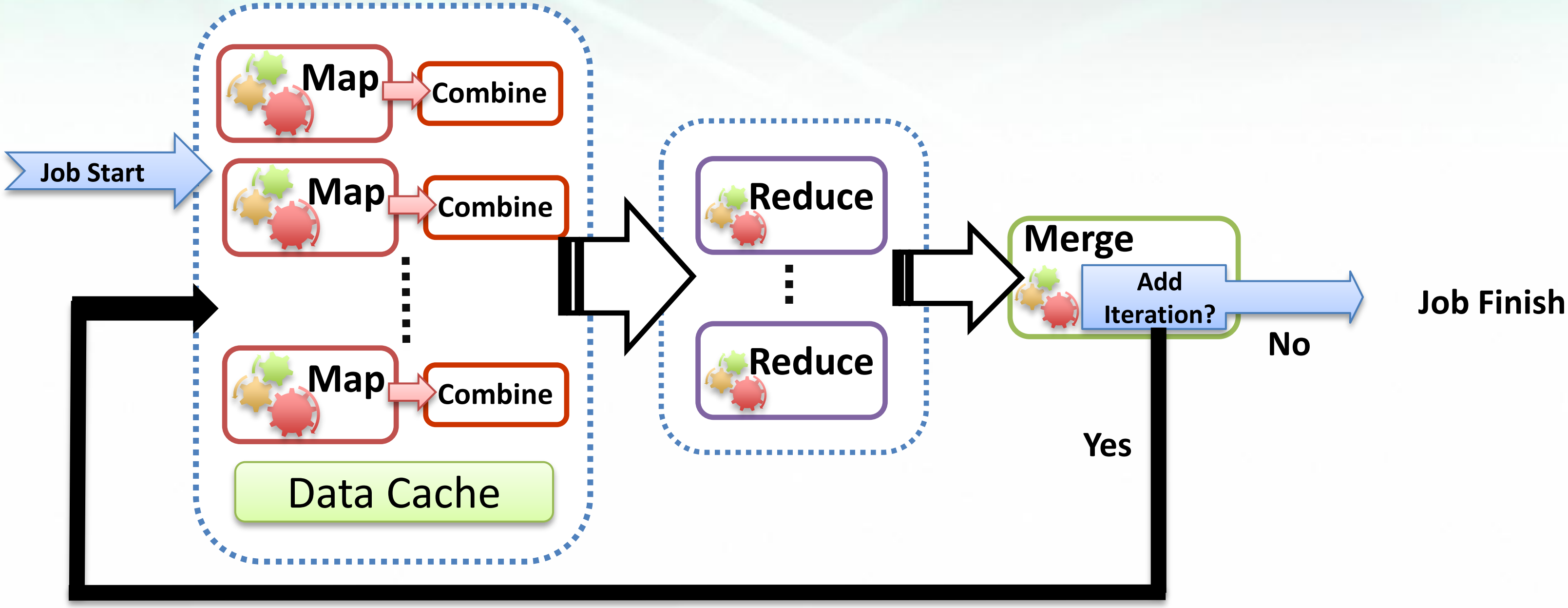


Iterate Matrix Vector Multiplication
(Power method for Eigenvector)

https://portal.futuregrid.org

SALSA HPC

High Level Flow Twister4Azure

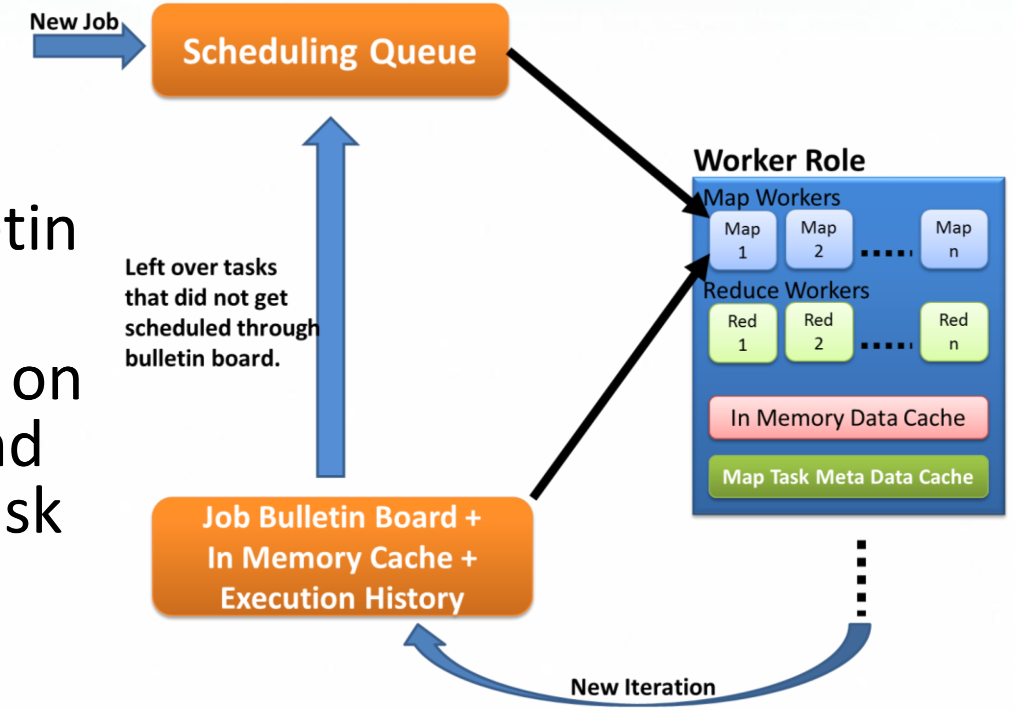


Hybrid scheduling of the new iteration

- Merge Step
- In-Memory Caching of static data
- Cache aware hybrid scheduling using Queues as well as using a bulletin board (special table)

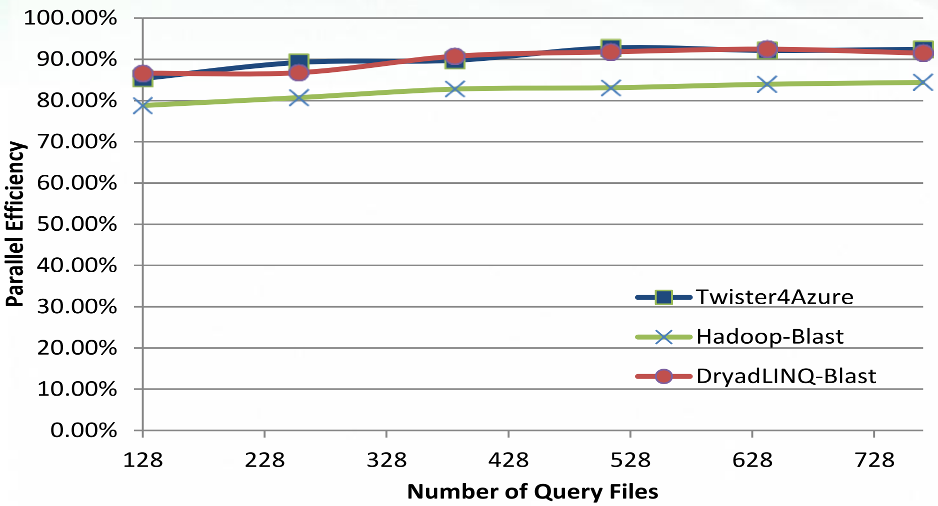
Cache aware scheduling

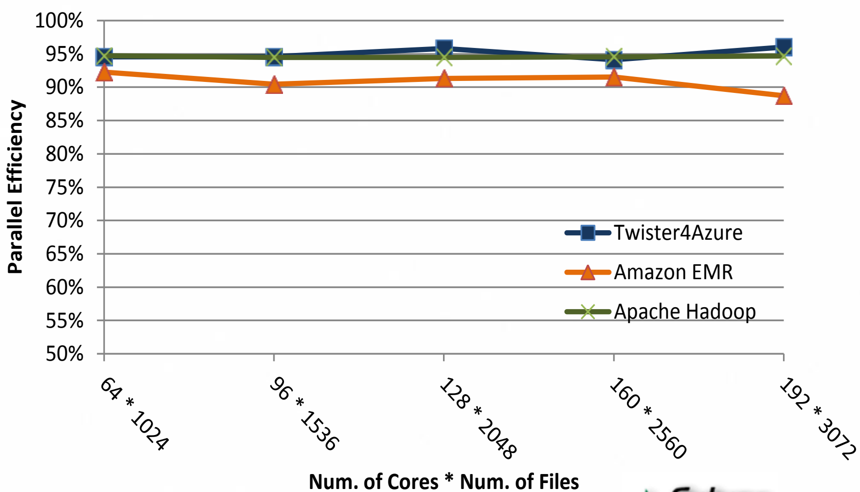
- New Job (1st iteration)
  – Through queues
- New iteration
  – Publish entry to Job Bulletin Board
  – Workers pick tasks based on in-memory data cache and execution history (MapTask Meta-Data cache)
  – Any tasks that do not get scheduled through the bulletin board will be added to the queue.



Twister4Azure Performance Comparisons

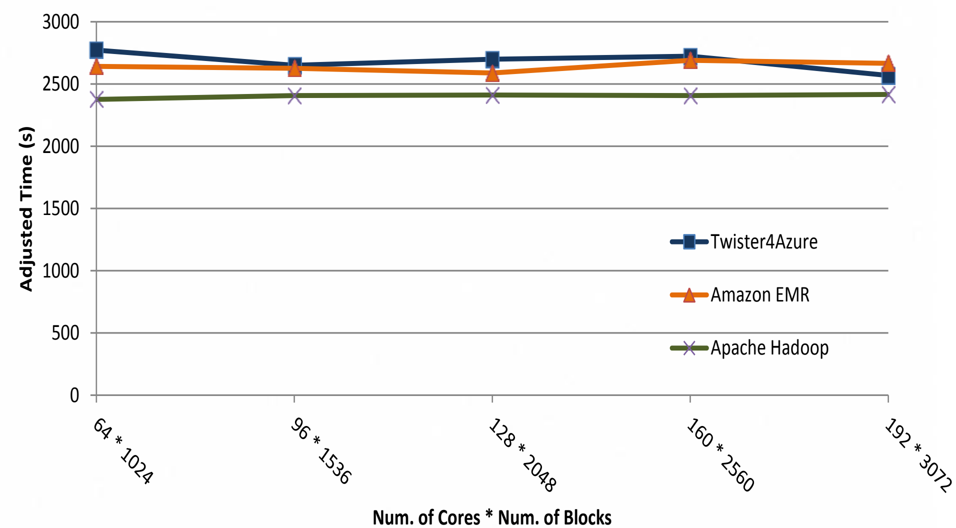
BLAST Sequence Search





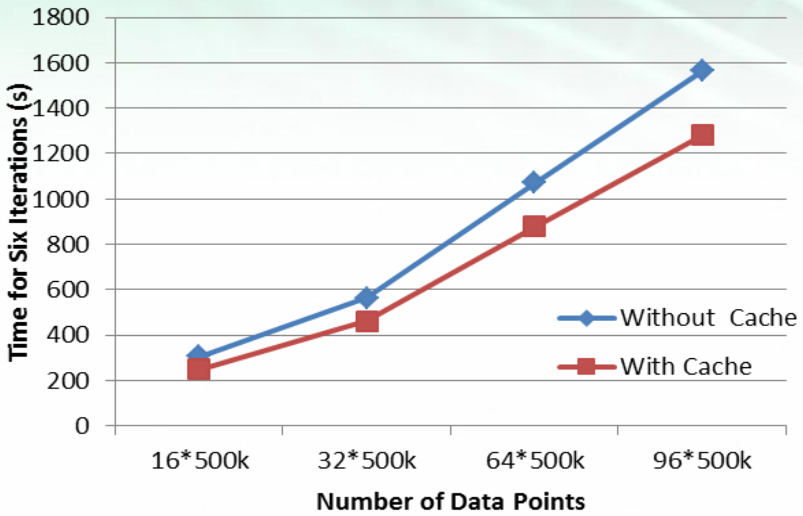
Cap3 Sequence Assembly

Smith Waterman Sequence Alignment

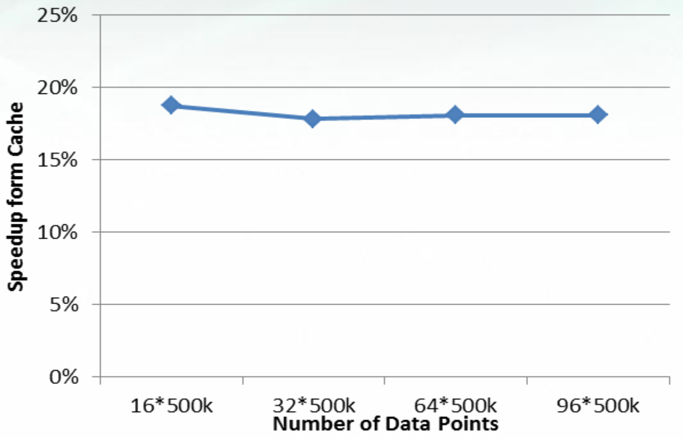


https://portal.futuregrid.org

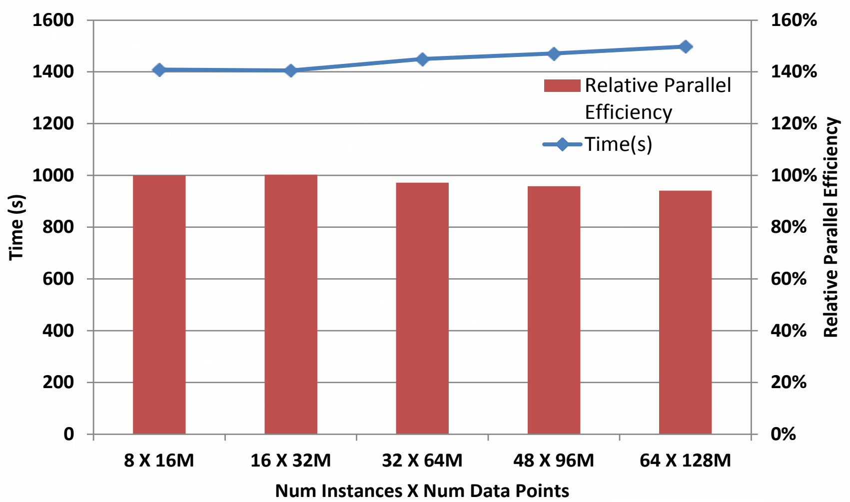
Twister4Azure Performance – Kmeans Clustering



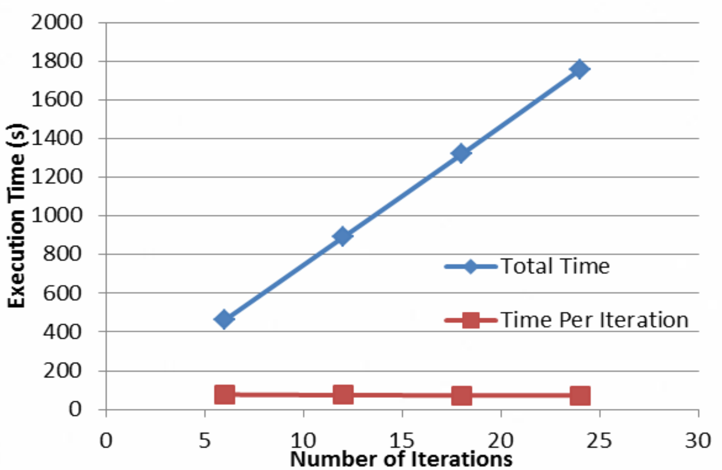
Performance with/without data caching.



Speedup gained using data cache



Scaled speedup



Increasing number of iterations

Future Grid https://portal.futuregrid.org

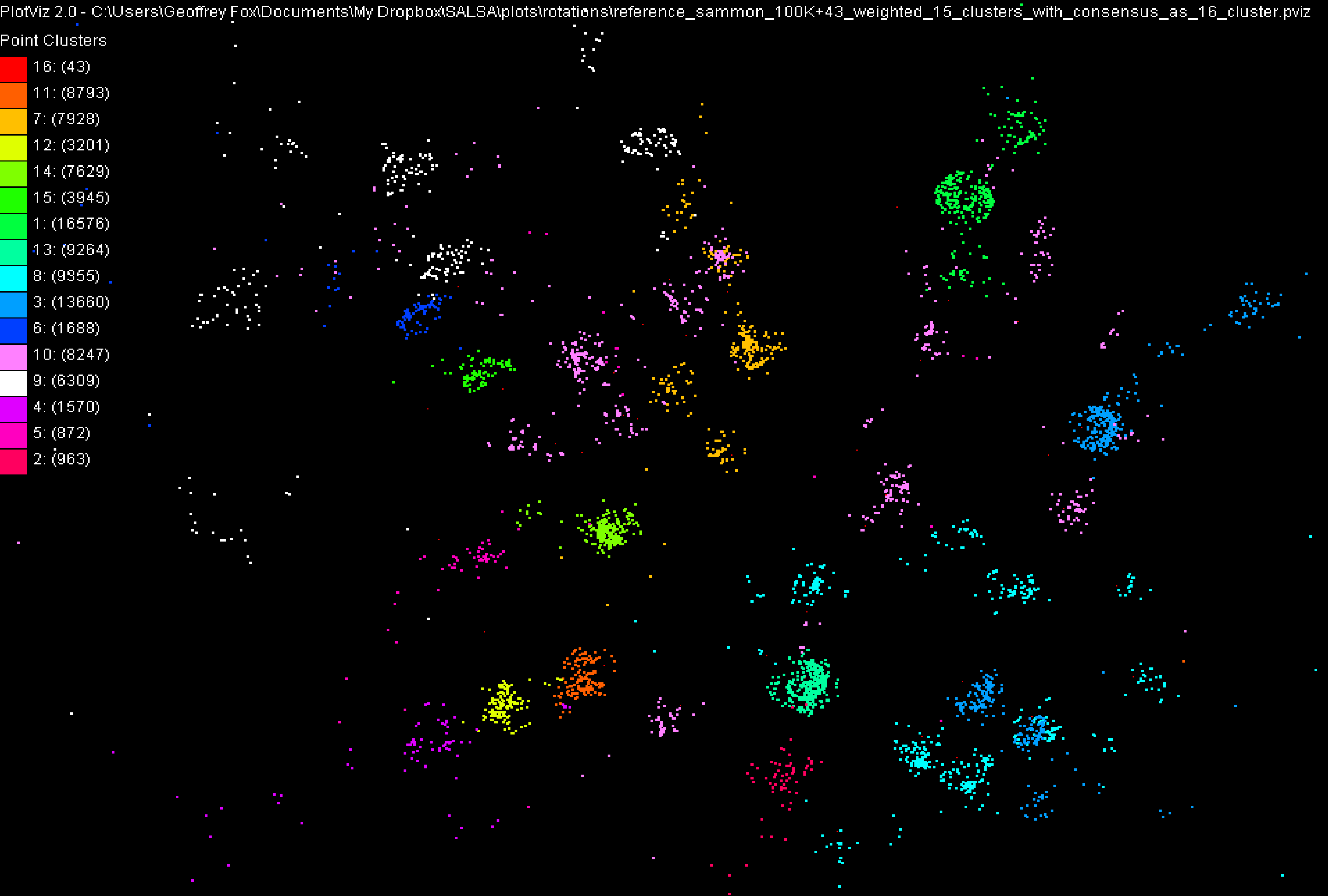
SALSA HPC

Visualizing Metagenomics

- Multidimensional Scaling MDS natural way to map sequences to 3D so you can visualize
- Minimize Stress

$$\sigma(\boldsymbol{X}) = \sum_{i<j\leq N} w_{ij}(d_{ij}(\boldsymbol{X}) - \delta_{ij})^2$$

- Improve with deterministic annealing (gives lower stress with less variation between random starts)
- Need to iterate Expectation Maximization
- N^2 dissimilarities (Needleman-Wunsch) δ_{ij}
- Communicate N positions <u>**X**</u> between steps

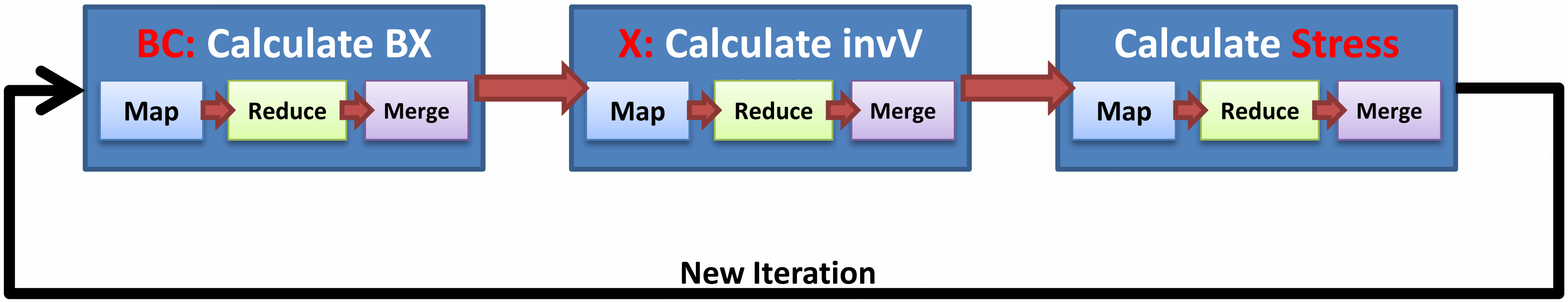


100,043 Metagenomics Sequences

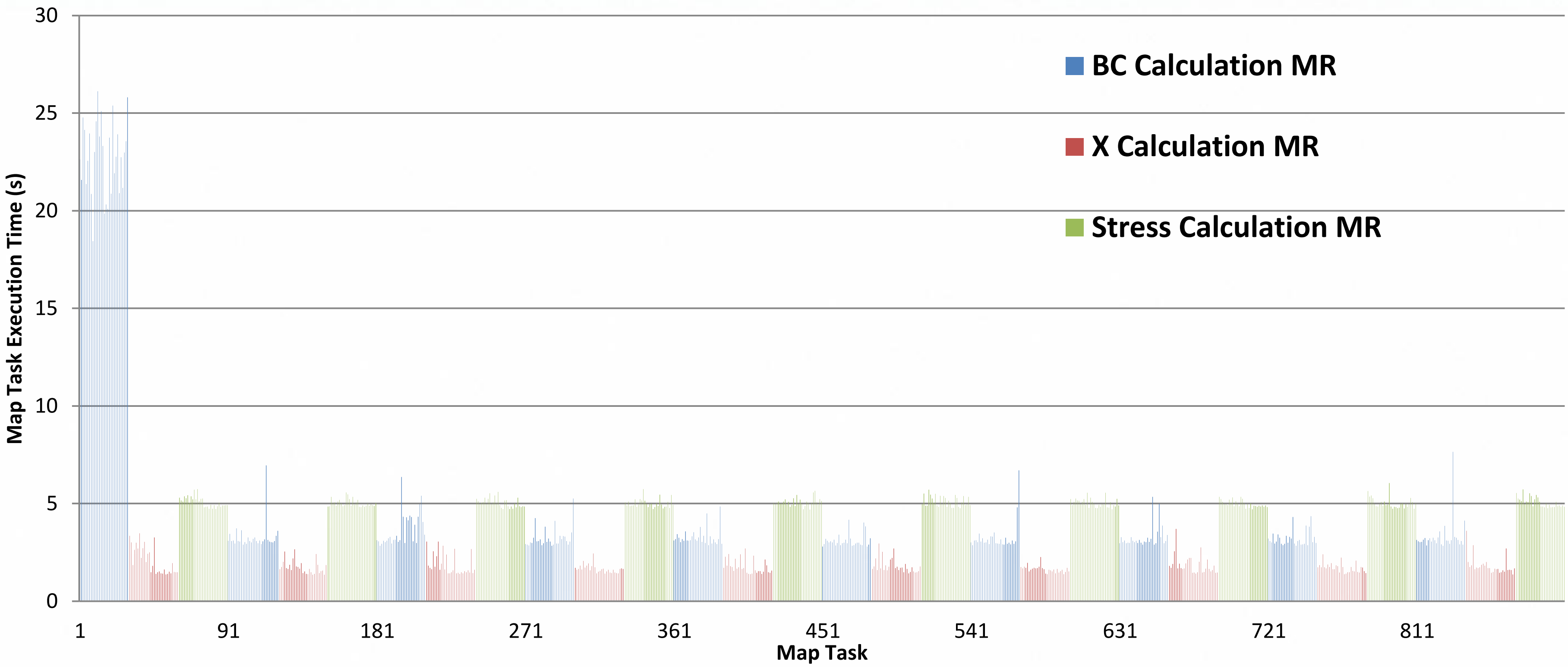
Aim to do 10 million by end of Summer

Multi-Dimensional-Scaling

- Many iterations
- Memory & Data intensive
- 3 Map Reduce jobs per iteration
- $\underline{X}_k = invV * B(\underline{X}_{(k-1)}) * \underline{X}_{(k-1)}$
- 2 matrix vector multiplications termed BC and X

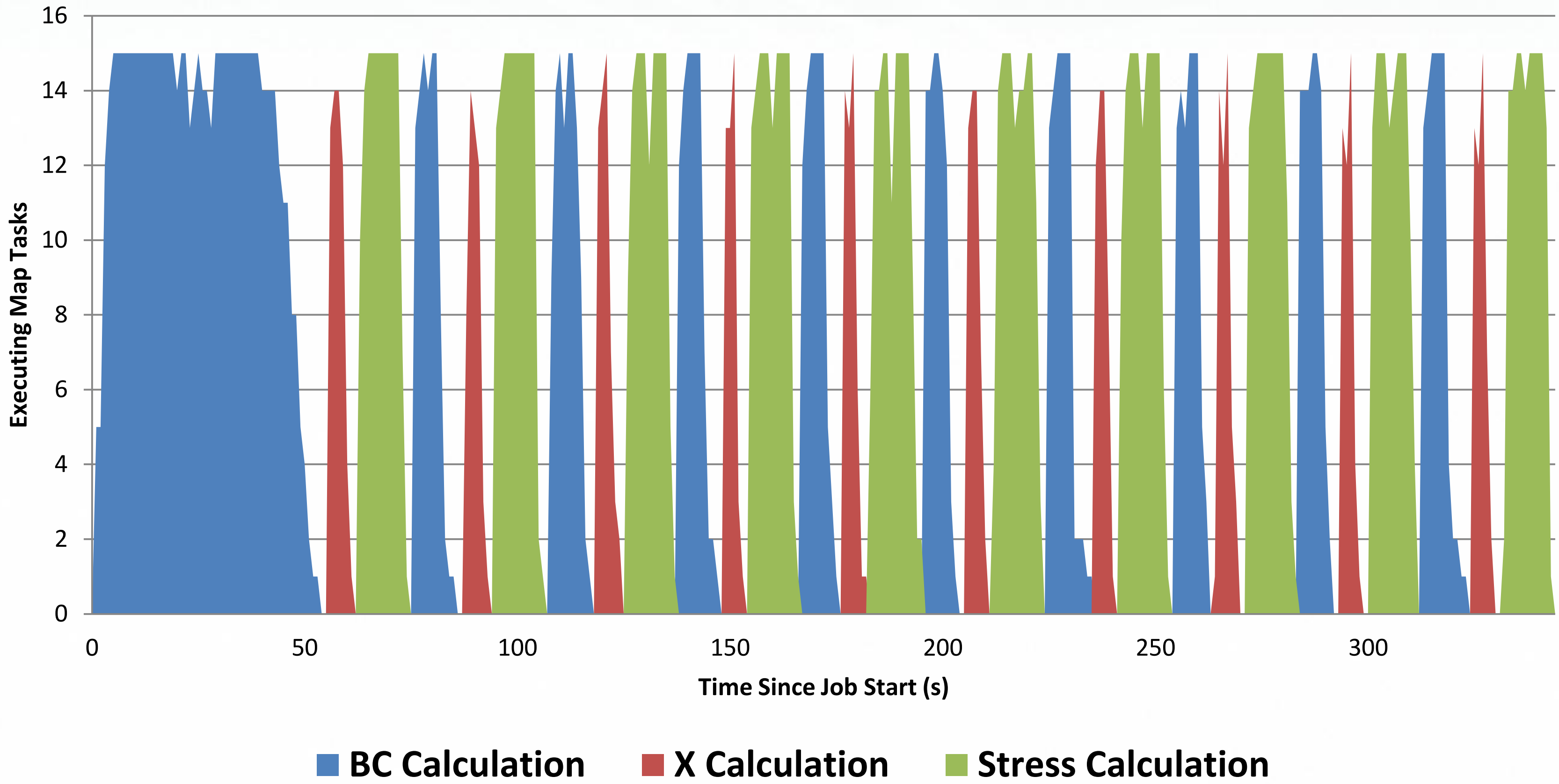


MDS Execution Time Histogram



10 iterations, 30000 * 30000 data points, 15 Azure Instances

MDS Executing Task Histogram

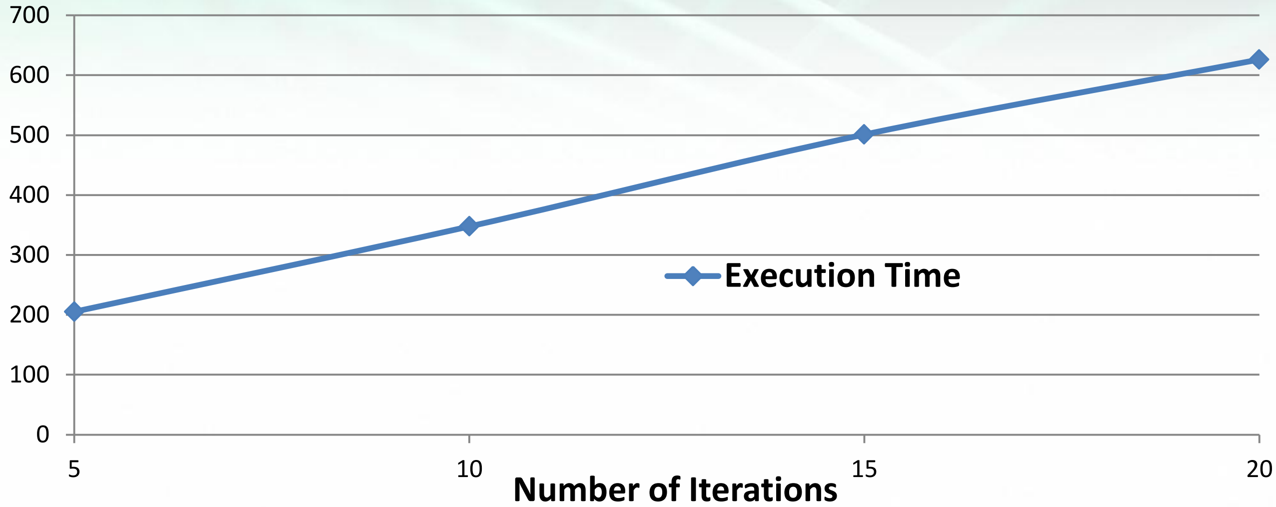


10 iterations, 30000 * 30000 data points, 15 Azure Instances

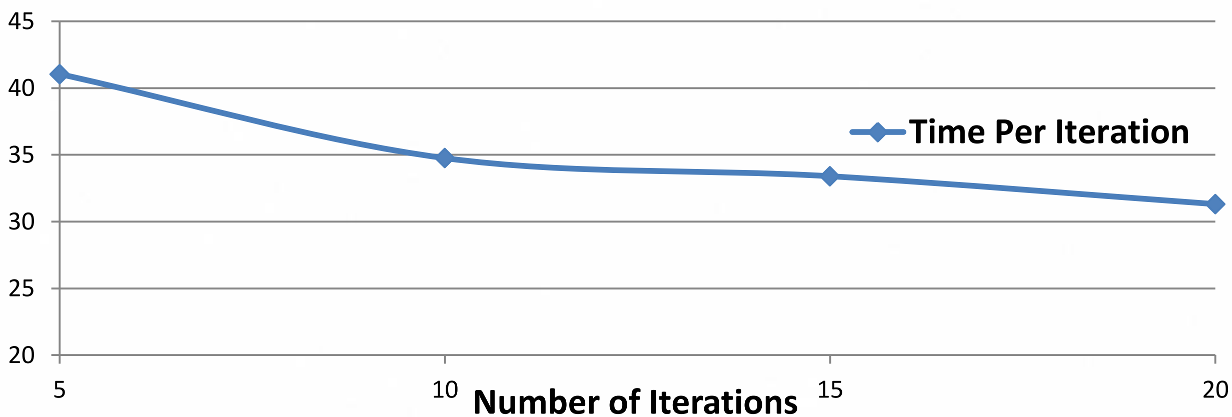
Future Grid https://portal.futuregrid.org

SALSA HPC

MDS Performance



# Instances	Speedup
6	6
12	16.4
24	35.3
48	52.8



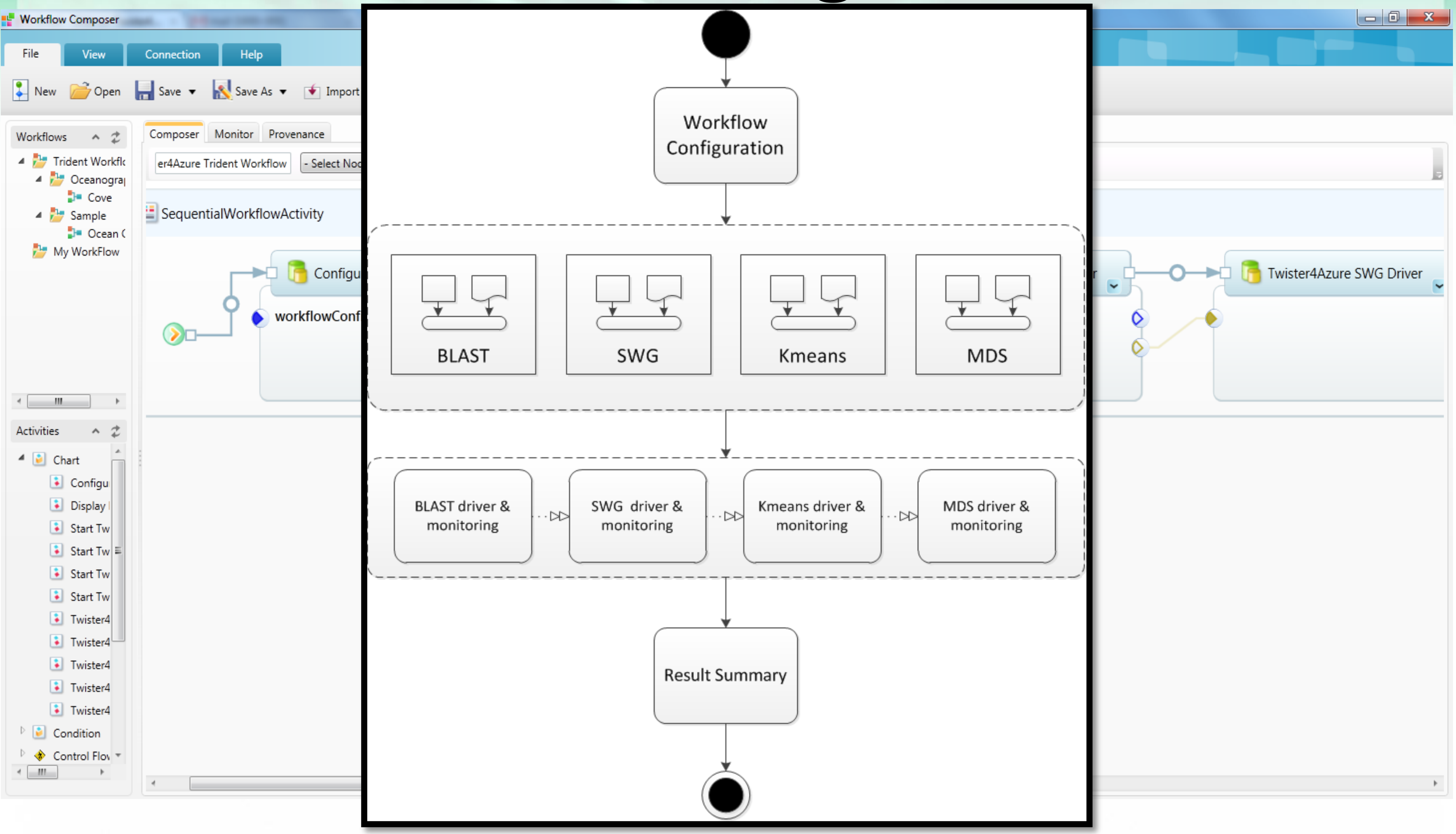
Probably super linear as used small instances

30,000*30,000 Data points, 15 instances, 3 MR steps per iteration
30 Map tasks per application

Future Grid https://portal.futuregrid.org

SALSA HPC

Trident Integration



Types of Data

- Loop invariant data (static data) – traditional MR key-value pairs
  - Comparatively larger sized data
  - Cached between iterations
- Loop variant data (dynamic data) – broadcast to all the map tasks in beginning of the iteration
  - Comparatively smaller sized data
- Can be specified even for non-iterative MR jobs

In-Memory Data Cache

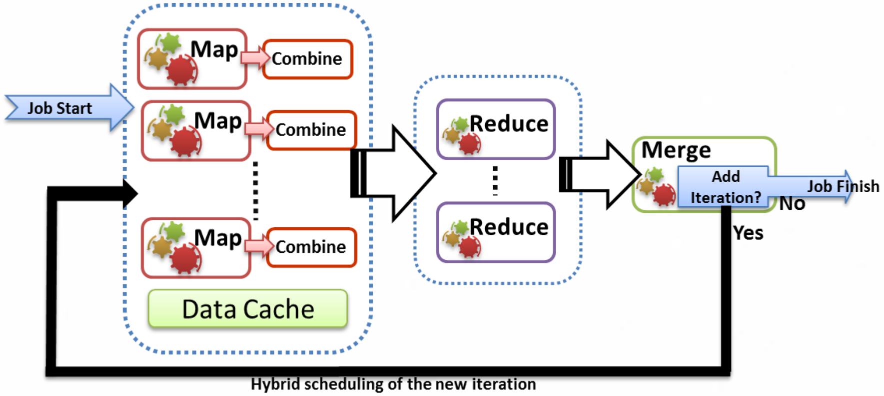
- Caches the loop-invariant (static) data across iterations
  - Data that are reused in subsequent iterations
- Avoids the data download, loading and parsing cost between iterations
  - Significant speedups for some data-intensive iterative MapReduce applications
- Cached data can be reused by any MR application within the job

Cache Aware Scheduling

- Map tasks need to be scheduled with cache awareness
  - Map task which process data ‘X’ needs to be scheduled to the worker with ‘X’ in the Cache
- Nobody has global view of the data products cached in workers
  - Decentralized architecture
  - Impossible to do cache aware assigning of tasks to workers
- Solution: workers pick tasks based on the data they have in the cache
  - Job Bulletin Board : advertise the new iterations

Merge Step

- Extension to the MapReduce programming model to support iterative applications
  - Map -> Combine -> Shuffle -> Sort -> Reduce -> Merge
- Receives all the Reduce outputs and the broadcast data for the current iteration
- User can add a new iteration or schedule a new MR job from the Merge task.
  - Serve as the “loop-test” in the decentralized architecture
    - Number of iterations
    - Comparison of result from previous iteration and current iteration
  - Possible to make the output of merge the broadcast data of the next iteration



Future Grid https://portal.futuregrid.org

SALSA HPC

Multiple Applications per Deployment

- Ability to deploy multiple Map Reduce applications in a single deployment
- Possible to invoke different MR applications in a single job
- Support for many application invocations in a workflow without redeployment

Conclusions

- Twister4Azure enables users to easily and efficiently perform large scale iterative data analysis and scientific computations on Azure cloud.
  - Supports classic and iterative MapReduce
- Utilizes a hybrid scheduling mechanism to provide the caching of static data across iterations.
- Can integrate with Trident (or other workflow)
- Plenty of testing and improvements to come!
- Open source: Please use http://salsahpc.indiana.edu/twister4azure
- Is it useful to make available as a Service?