A Hierarchical Framework for Cross-Domain MapReduce Execution

Yuan Luo[1], Zhenhua Guo[1], Yiming Sun[1],
Beth Plale[1], Judy Qiu[1], Wilfred W. Li [2]
[1] School of Informatics and Computing, Indiana University
[2] San Diego Supercomputer Center, University of California, San Diego

ECMLS Workshop of HPDC 2011, San Jose, CA, June 8th 2011

Background

- The MapReduce programming model provides an easy way to execute embarrassingly parallel applications.
- Many data-intensive life science applications fit this programming model and benefit from the scalability that can be delivered using this model.

A MapReduce Application from Life Science: AutoDock Based Virtual Screening

- AutoDock:
  - a suite of automated docking tools for predicting the bound conformations of flexible ligands to macromolecular targets.
- AutoDock based Virtual Screening:
  - Ligand and receptor preparation, etc.
  - A large number of docking processes from multiple targeted ligands
  - Docking processes are data independent

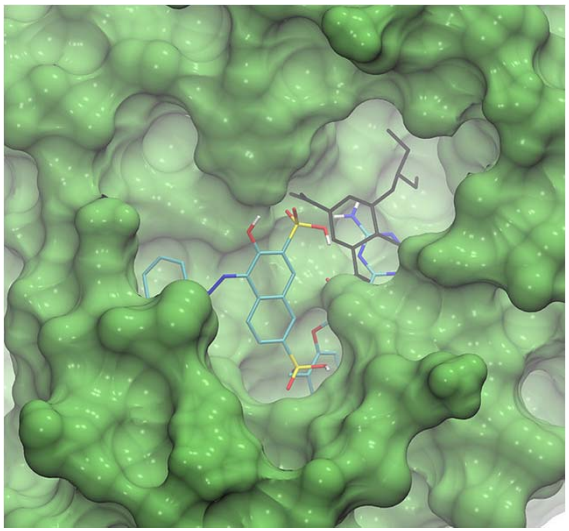


Image source: NBCR

Challenges

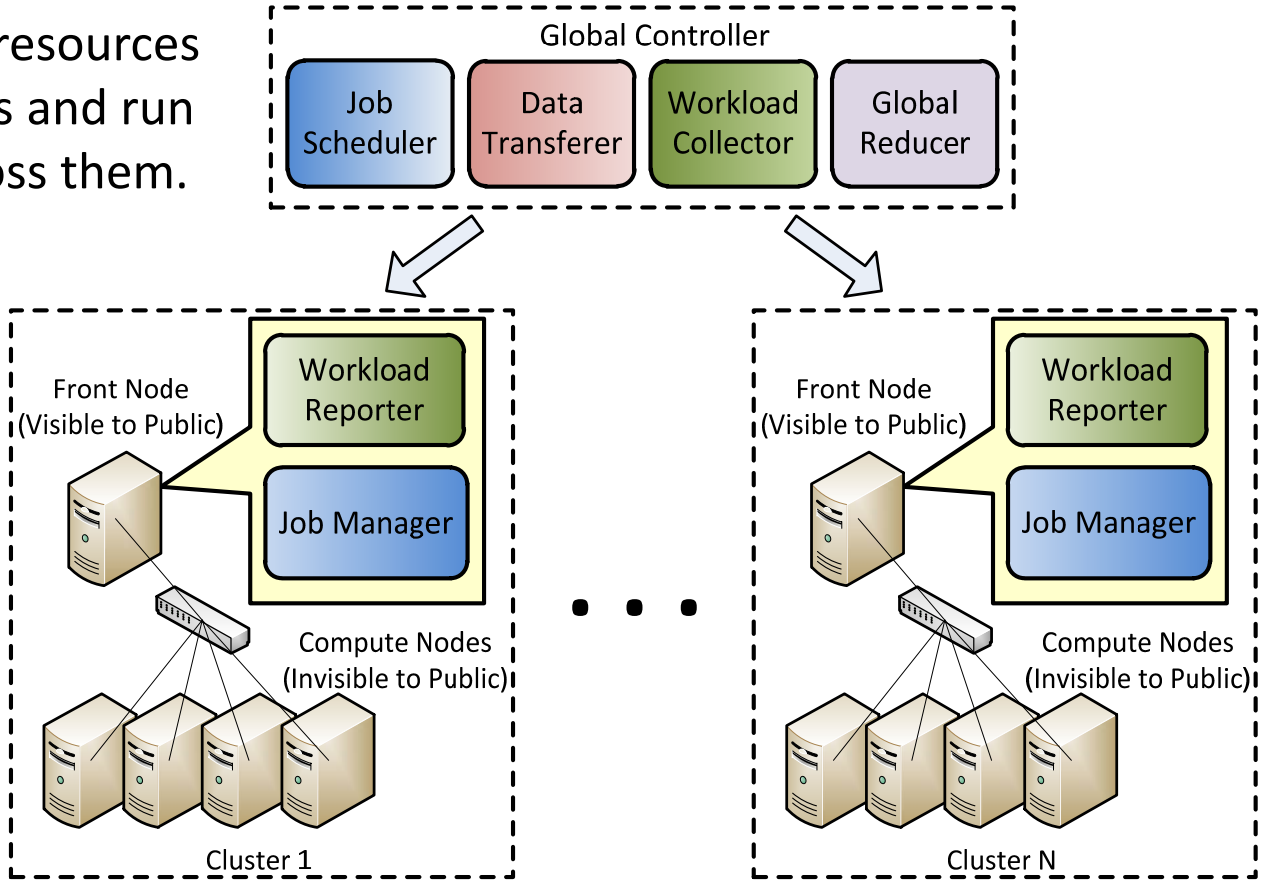
- Life Science Applications typically contains large dataset and/or large computation.
- Only small clusters are available for mid-scale scientists.
- Running MapReduce over a collection of clusters is hard
  - Internal nodes of a cluster is not accessible from outside

Solutions

- Allocating a large Virtual Cluster
  - Pure Cloud Solution
- Coordinating multiple physical/virtual clusters.
  - Physical clusters
  - Physical + Virtual clusters
  - Virtual clusters

Hierarchical MapReduce

Gather computation resources from multiple clusters and run MapReduce jobs across them.



Features

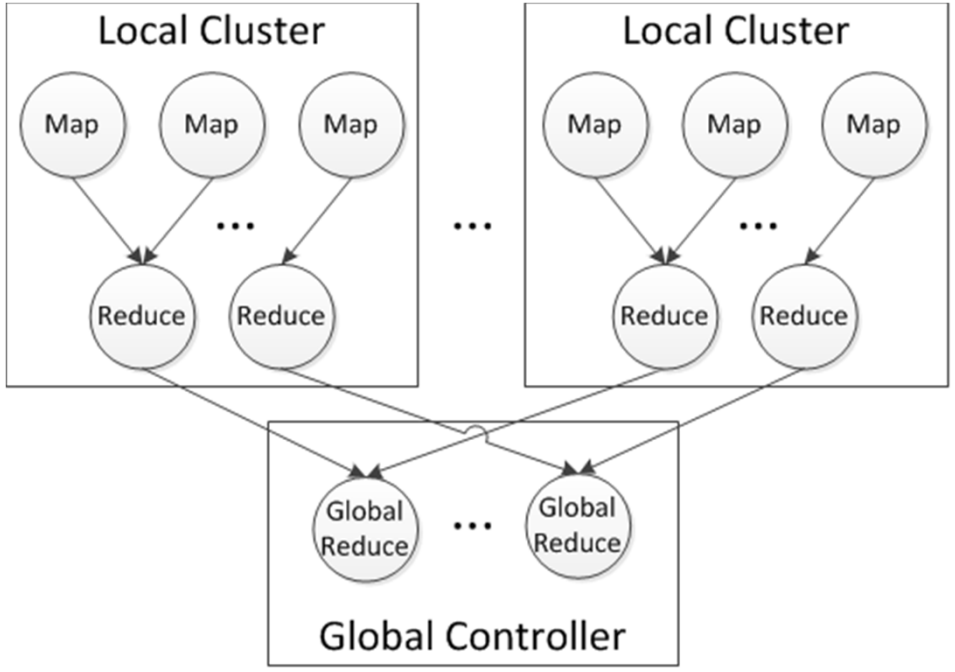
- Map-Reduce-GlobalReduce Programming Model
- Focus on Map-Only and Map-Mostly Jobs

  map-only, map-mostly, shuffle-mostly, and reduce-mostly *
- Scheduling Policies:
  - Computing Capacity Aware
  - Data Locality Aware (development in progress)

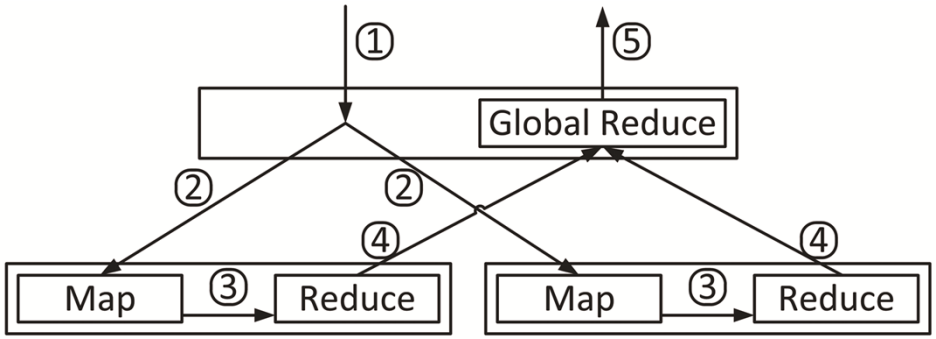
* Kavulya, S., Tan, J., Gandhi, R., and Narasimhan, P. 2010. An Analysis of Traces from a Production MapReduce Cluster. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '10). IEEE Computer Society, Washington, DC, USA, 94-103.

Programming Model

Function Name	Input	Output
Map	(k^i, v^i)	(k^m, v^m)
Reduce	$(k^m, [v_1^m, \dots, v_n^m])$	(k^r, v^r)
Global Reduce	$(k^r, [v_1^r, \dots, v_n^r])$	(k^o, v^o)



Procedures



1) A job is submitted into the system.
2) global controller to local clusters.
3) Intermediate pairs are passed to the Reduce tasks.
4) Local reduce outputs (including new key/value pairs) are send back to the global controller .
5) The Global Reduce task takes key/value pairs from local Reducers, performs the computation, and produces the output.

Computing Capacity Aware Scheduling

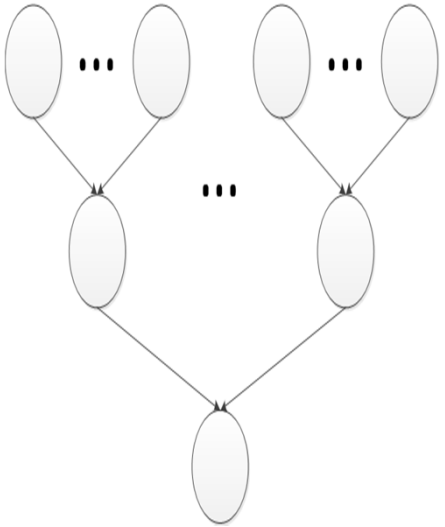
- $MaxMapper_i = \rho_i \times NumCore_i$
  - *ρ_i is defined as maximum numbers of mappers per core.*
- $\gamma_i = MaxMapper_i - MapperRun_i,$
  - γ_i is the number of available Mappers on $Cluster_i$
- $Weight_i = \frac{\gamma_i \times \theta_i}{\sum_{i=1}^{N} \gamma_i \times \theta_i},$
  - θ_i is the computing power of each cluster;
- $JobMap_{x,i} = Weight_i \times JobMap_x$
  - $JobMap_{x,i}$ is the number of Map tasks to be scheduled to $Cluster_i$ for job x,

MapReduce to run multiple AutoDock instances

1) ***Map***: AutoDock binary executable + Python script summarize_result4.py to output the lowest energy result using a constant intermediate key.
2) ***Reduce***: Sort the values values corresponding to the constant intermediate key by the energy from low to high, and outputs the results.
3) ***Global Reduce***: Sorts and combines local clusters outputs into a single file by the energy from low to high.

AutoDock MapReduce input fields and descriptions

Field	Description
ligand_name	Name of the ligand
autodock_exe	Path to AutoDock executable
input_files	Input files of AutoDock
output_dir	Output directory of AutoDock
autodock_parameters	AutoDock parameters
summarize_exe	Path to summarize script
summarize_parameters	Summarize script parameters



Experiment Setup

- **Cluster Nodes Specifications.**
  - FG: FutureGrid , IU: Indiana University

Cluster	CPU	Cache size	# of Core	Memory
Hotel (FG)	Intel Xeon 2.93GHz	8192KB	8	24GB
Alamo (FG)	Intel Xeon 2.67GHz	8192KB	8	12GB
Quarry (IU)	Intel Xeon 2.33GHz	6144KB	8	16GB

- PBS allocated 21 nodes per cluster
  - 1 namenode, 20 datanode
- set $\rho_i = 1$ so that
  - $MaxMapper_i = \rho_i \times NumCore_i$
- AutoDock Version 4.2 on each cluster
- 6,000 ligands and 1 receptor.
- *ga_num_evals* = 2,500,000

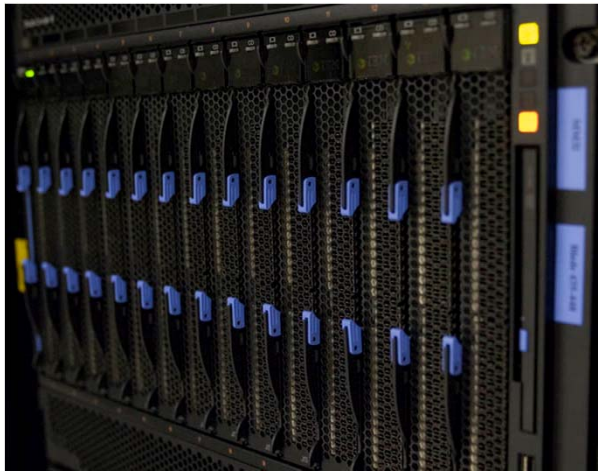


Image Source: Indiana University

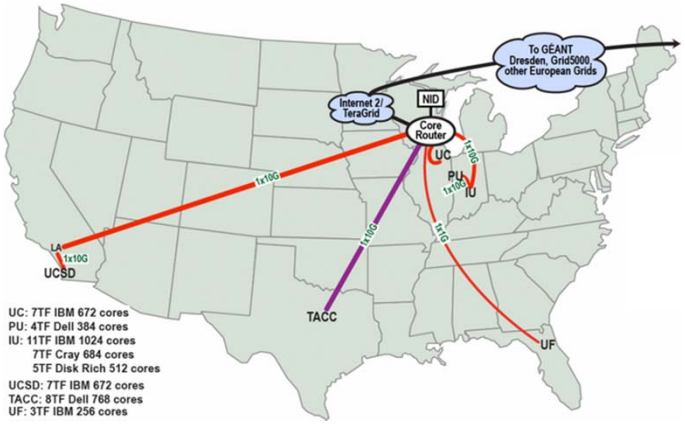


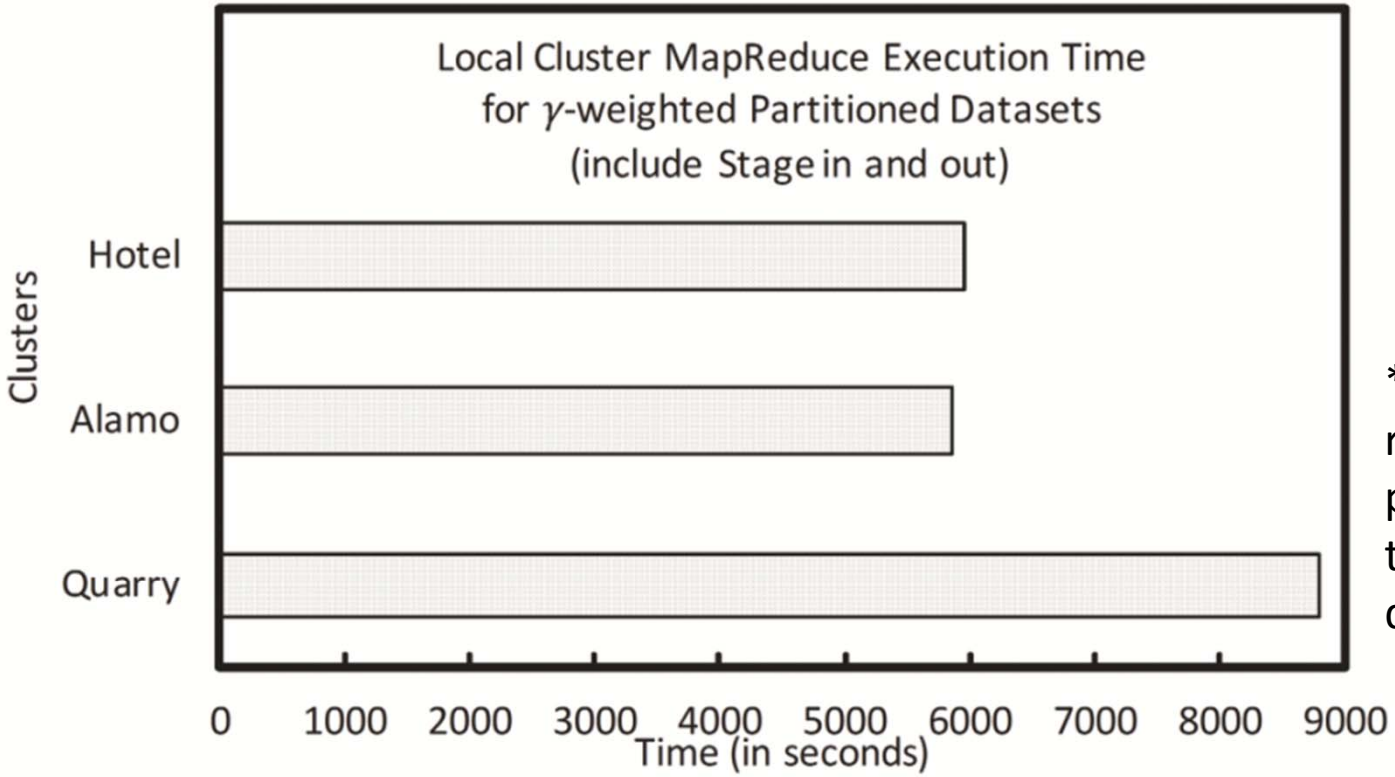
Image Source: FutureGrid

Evaluation

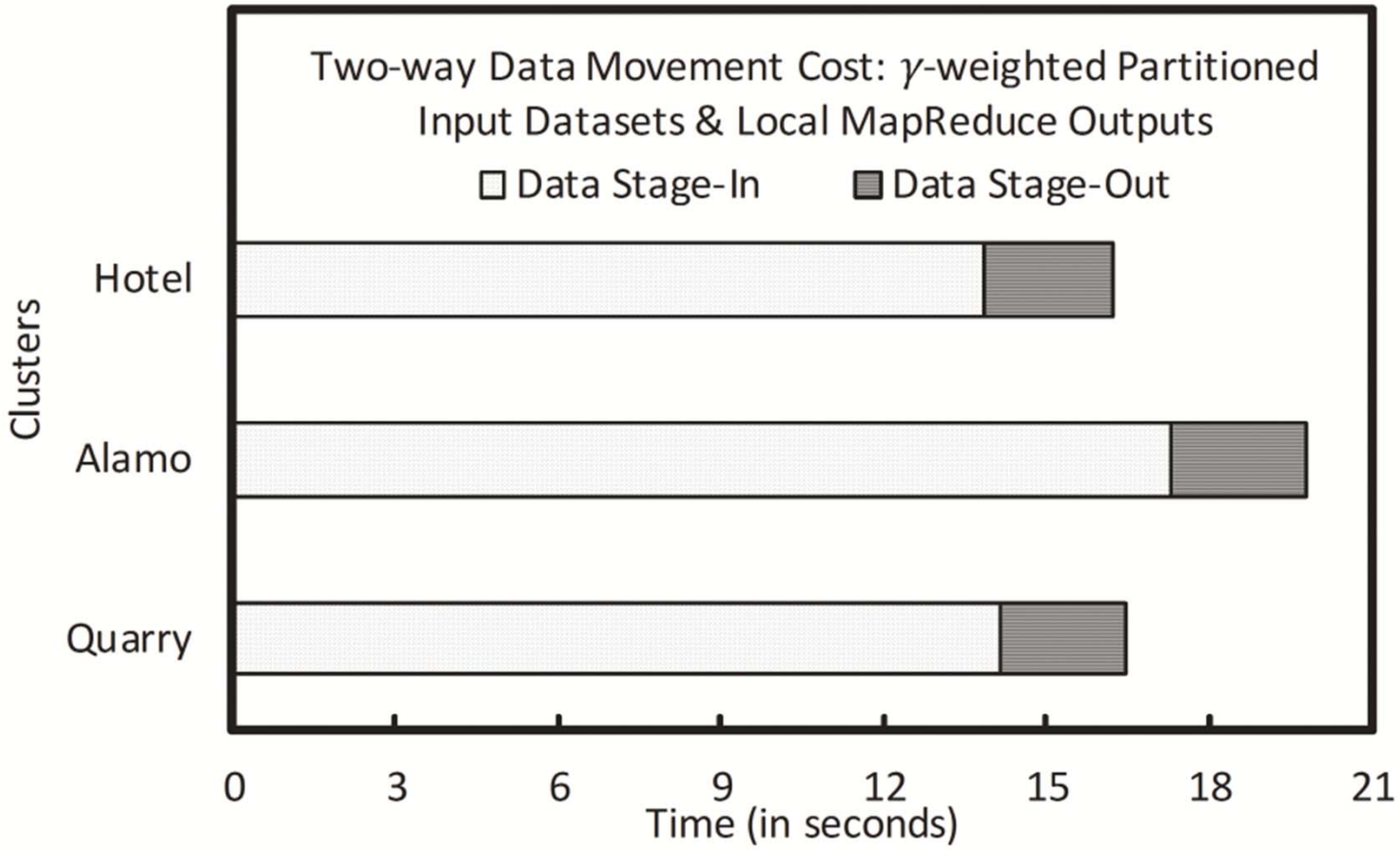
γ-weighted dataset partition:

set $\theta_i = C$, where C is a constant, $\gamma_1 = \gamma_2 = \gamma_3 = 160$

$Weight_i = 1/3$

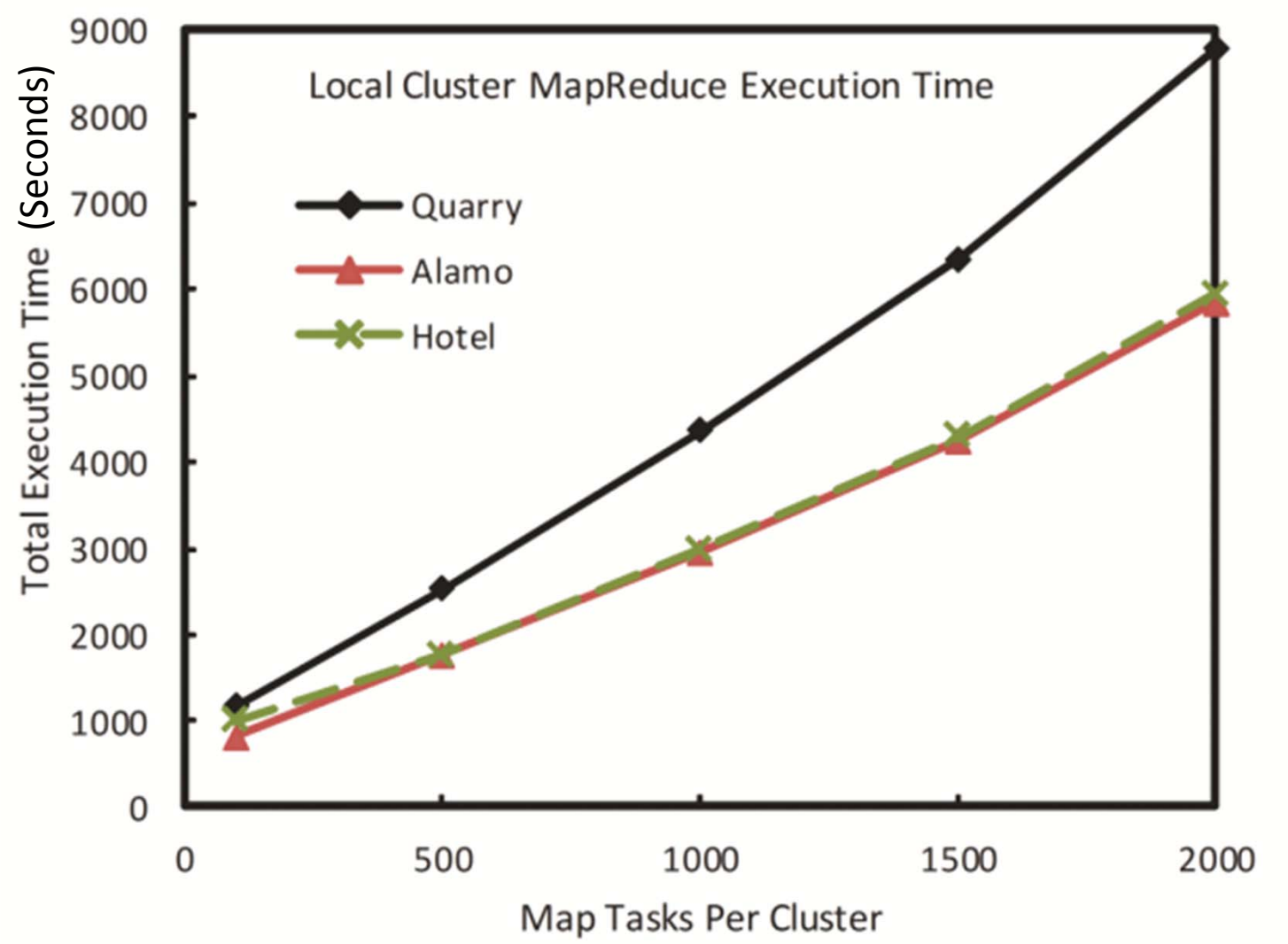


**The average global reduce time taken after processing 6000 map tasks (ligand/receptor docking) is 16 seconds.



Data Movement cost can be ignored in comparison with the computation cost

Local cluster MapReduce execution time based on different number of map tasks.

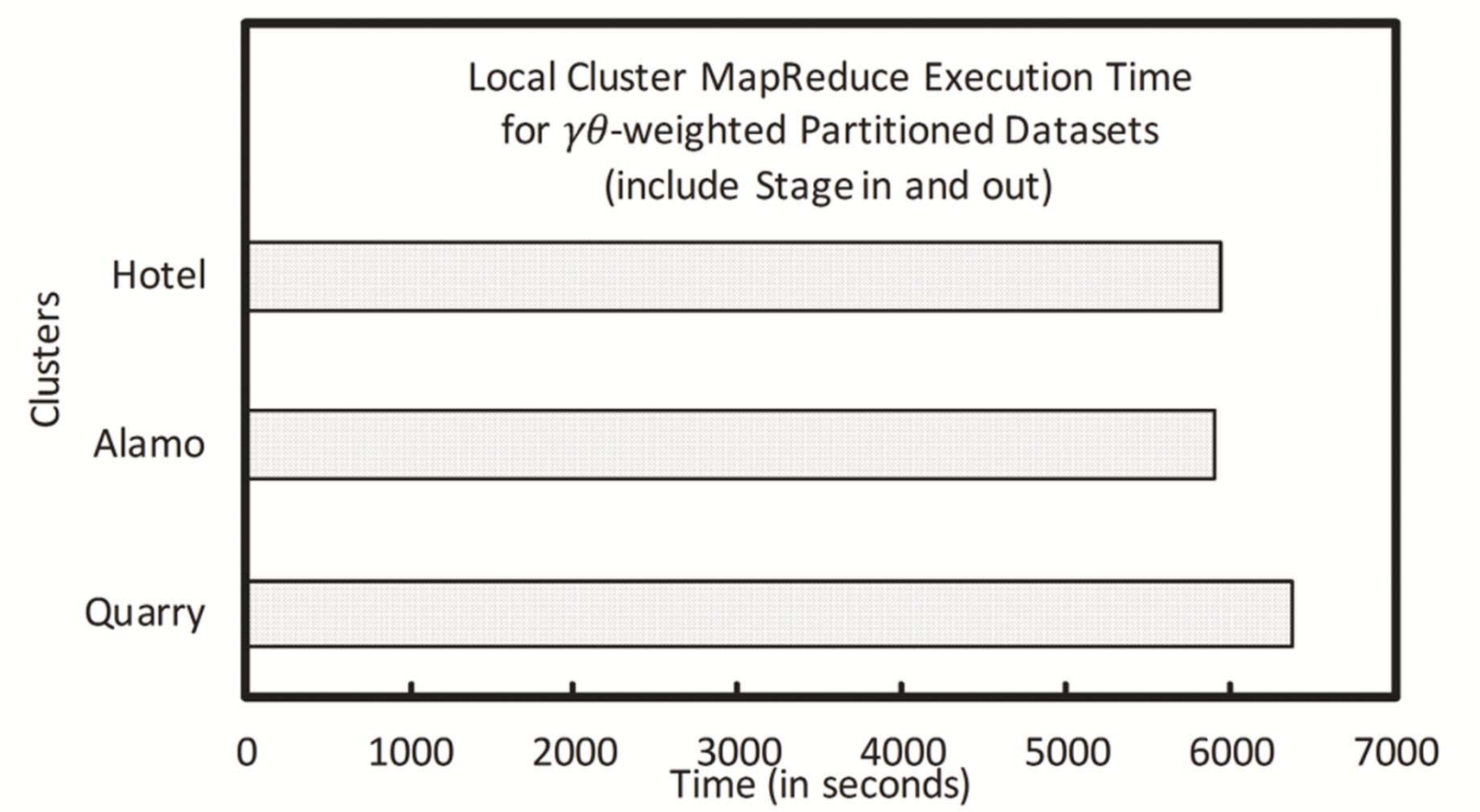


$\gamma\theta$-weighted dataset partition:

$\theta_1 = 2.93$ (Hotel), $\theta_2 = 2.67$ (Alamo), $\theta_3 = 2$ (Quarry)

$\gamma_1 = \gamma_2 = \gamma_3 = 160$

$Weight_1 = 0.3860,\ Weight_2 = 0.3505,\ Weight_3 = 0.2635$



Conclusion and Future Work

- A hierarchical MapReduce framework as a solution to run MapReduce over a collection of clusters.
- “Map-Reduce-Global Reduce” model.
- Computing Capacity Aware Scheduling
- AutoDock as an example.
- Performance Evaluation showed the workload are well balanced and the total makespan was kept in minimum.

- Performance Test for Large Dataset Applications.
  - Data transfer overhead
  - Bring Computation to Data
  - Share File System that uses local storage
  - Change θ_i in the current scheduling policy
- Replace ssh+scp glue
  - Meta-scheduler?
  - Better data movement solution
    - gridftp?
    - Distributed file system?

Acknowledgements

- This work funded in part by
  - Pervasive Technology Institute of Indiana University
  - Microsoft
- Special thanks to Dr. Geoffrey Fox for providing early access to FutureGrid.

Thanks!
Questions?

Yuan Luo, http://www.yuanluo.net
Indiana University School of Informatics and Computing
http://www.soic.indiana.edu
Indiana University Data to Insight Center
http://pti.iu.edu/d2i

Backup Slides