MATRIX: MAny-Task computing execution fabRIc at eXascale

Ke Wang
Data-Intensive Distributed Systems Laboratory
Computer Science Department
Illinois Institute of Technology

CS554: Data-Intensive Computing, IIT
February 4th, 2015

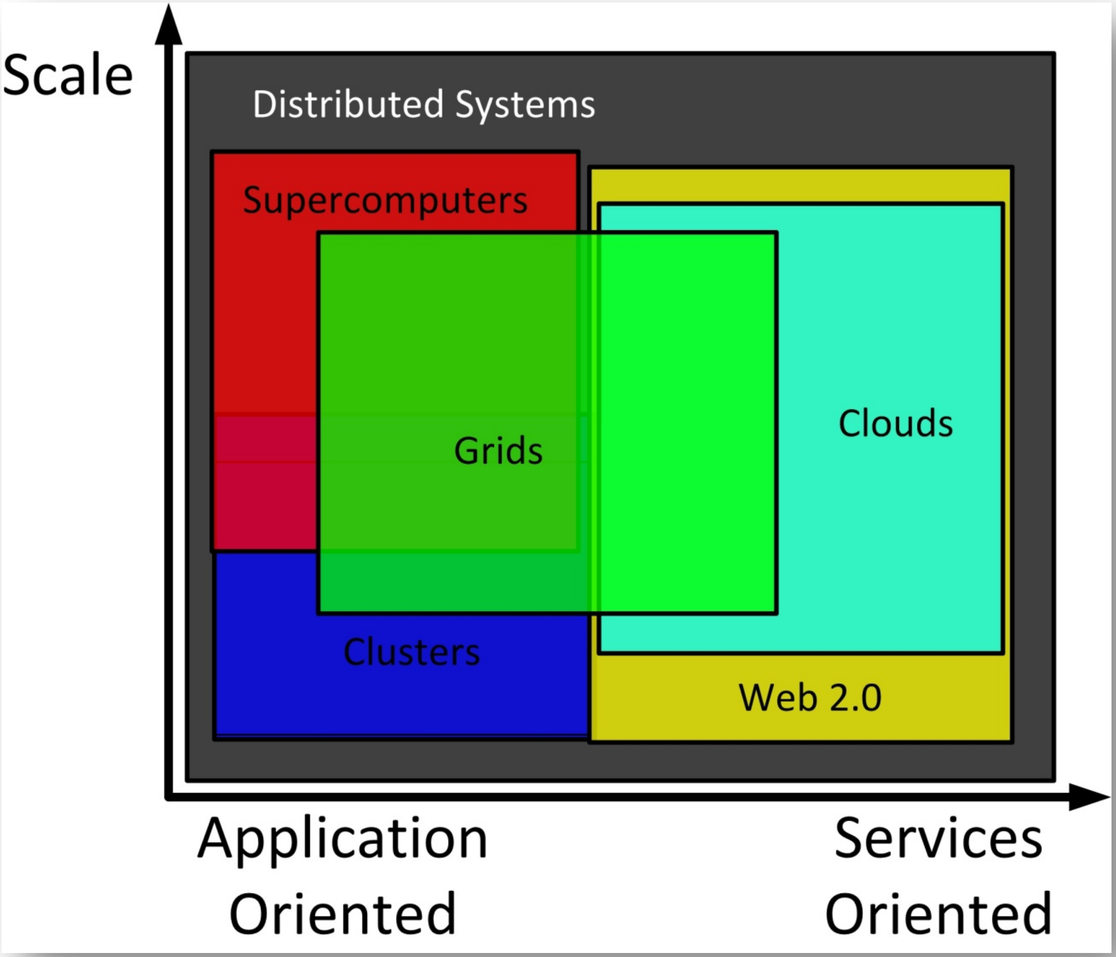
Outline

- Introduction & Motivation
- Problem Statement
- Proposed Work
- Evaluation
- Conclusions
- Future Work

Outline

- **Introduction & Motivation**
- Problem Statement
- Proposed Work
- Evaluation
- Conclusions
- Future Work

Distributed Systems



Exascale Computing

– 1Billion cores

http://s.top500.org/static/lists/2014/06/TOP500_201406_Poster.pdf

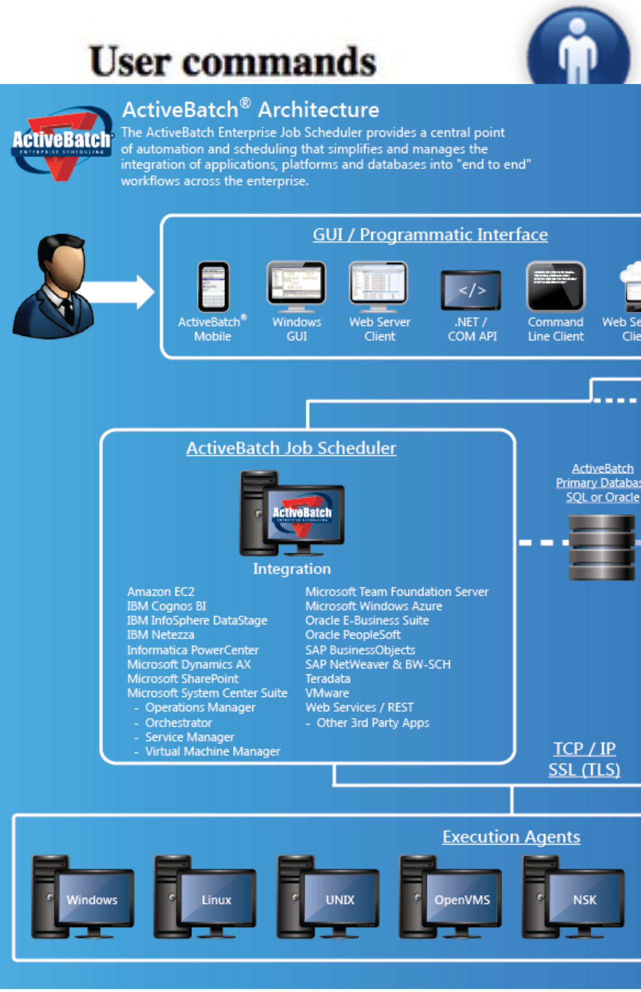
http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf

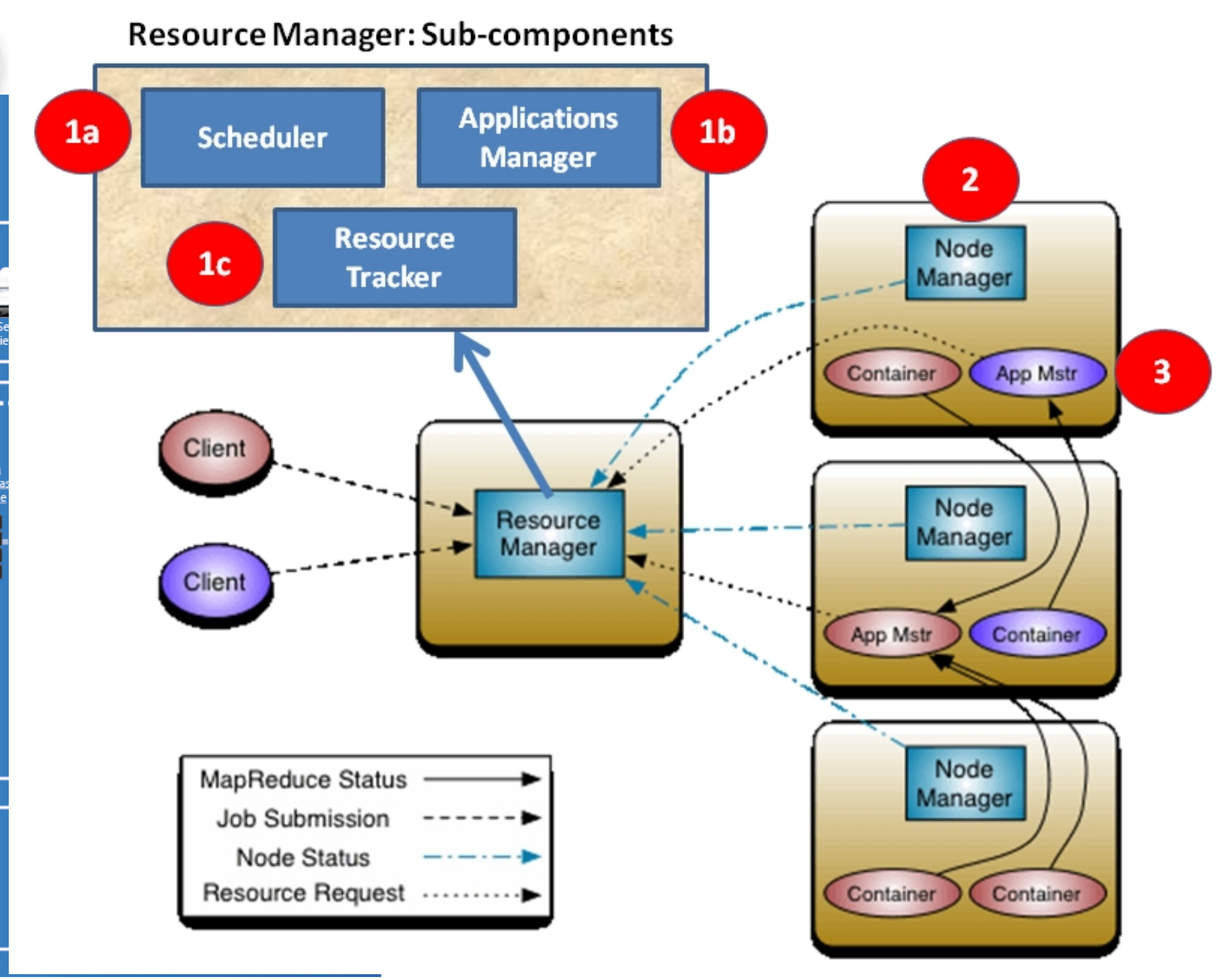
MATRIX: MAny-Task computing execution fabRIc at eXascale

Resource Manager

- Manages resources
  - compute
  - storage
  - network
- Job scheduling/management
  - resource allocation
  - job launch
- Data management
  - data movement
  - caching

Resource Manager Examples





Motivation

Decade

1970s

1980s

1990s

2000s

2010s

Medical Image Processing: Functional Magnetic Resonance

Chemistry Domain: MolDyn

Molecular Dynamics: DOCK

Production Runs in Drug Design

Economic Modeling: MARS

Large-scale Astronomy Application Evaluation

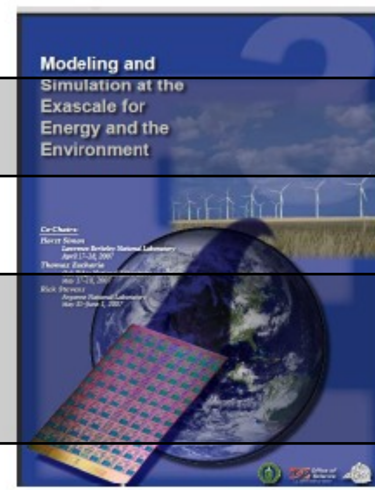
Astronomy Domain: Montage

Data Analytics: Sort and WordCount

Ensemble simulations are important for future exascale platforms

One approach to dealing with uncertainty is to perform multiple ensemble runs (parameter sweeps) with various combinations of the uncertain parameters. Since the space of parameters will be of high dimension, we will have to address the challenges of designing efficient parameter sweep methods for highdi-mensional spaces. Recent advances in approximation theory and data mining methods, such as sparse grids, offer new approaches to this problem. Furthermore, recent results in approximation theory can be used to guide us in using exascale computing power to search for efficient methods.

(Source: Horst Simon et al. Modeling and Simulation at the Exascale for Energy and the Environment)



http://elib.dlr.de/64768/1/EnSIM_-_Ensemble_Simulation_on_HPC_Computes_EN.pdf

Outline

- Introduction & Motivation
- **Problem Statement**
- Proposed Work
- Evaluation
- Conclusions
- Future Work

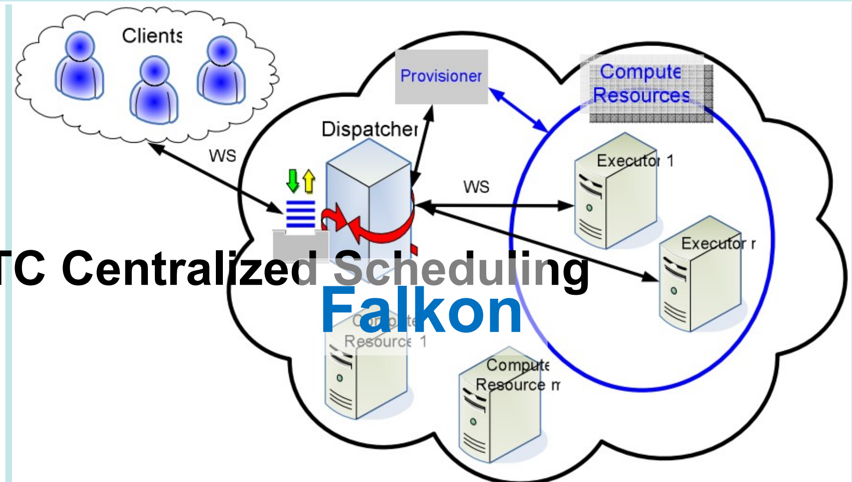
Problem Statement
Job Scheduling System Challenges

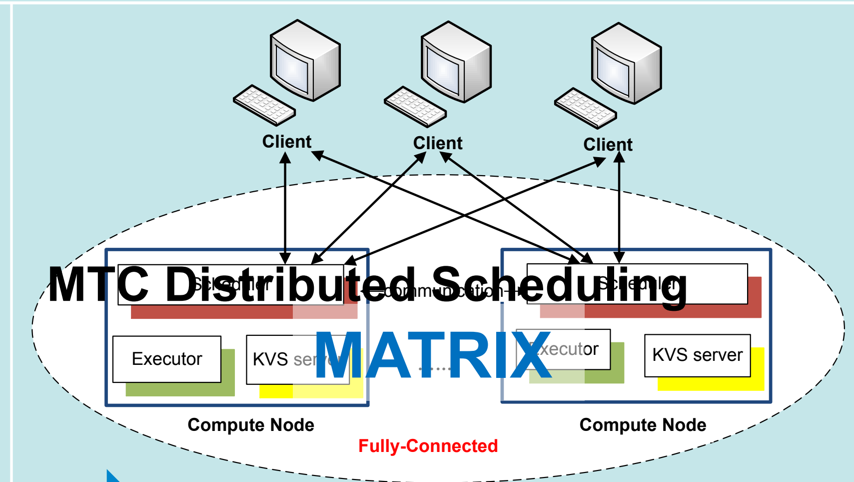
- **Scalability**
  - System scale is increasing
  - Workload size is increasing
  - Processing capacity needs to increase
- **Efficiency**
  - Allocating resources fast
  - Making fast enough scheduling decisions
  - Maintaining a high system utilization
- **Reliability**
  - Still functioning well under failures

Outline

- Introduction & Motivation
- Problem Statement
- **Proposed Work**
- Evaluation
- Conclusions
- Future Work

Job Scheduling Systems





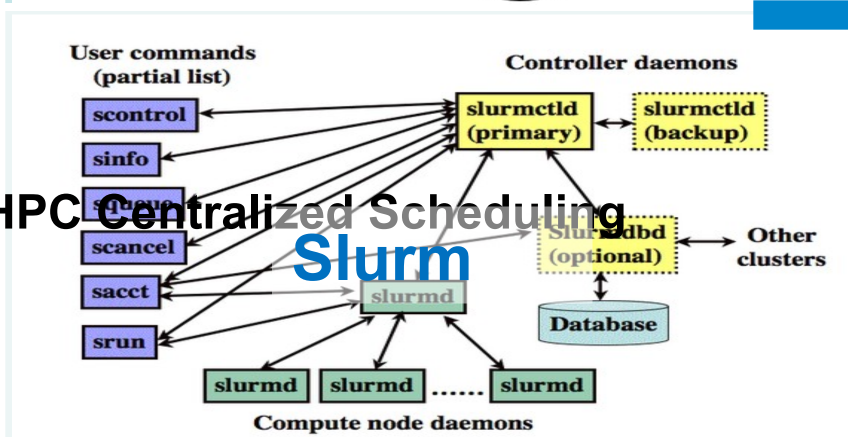
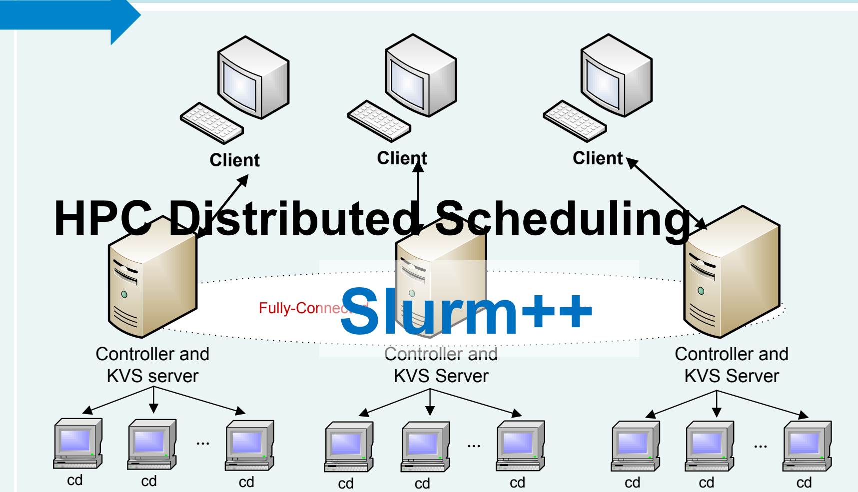
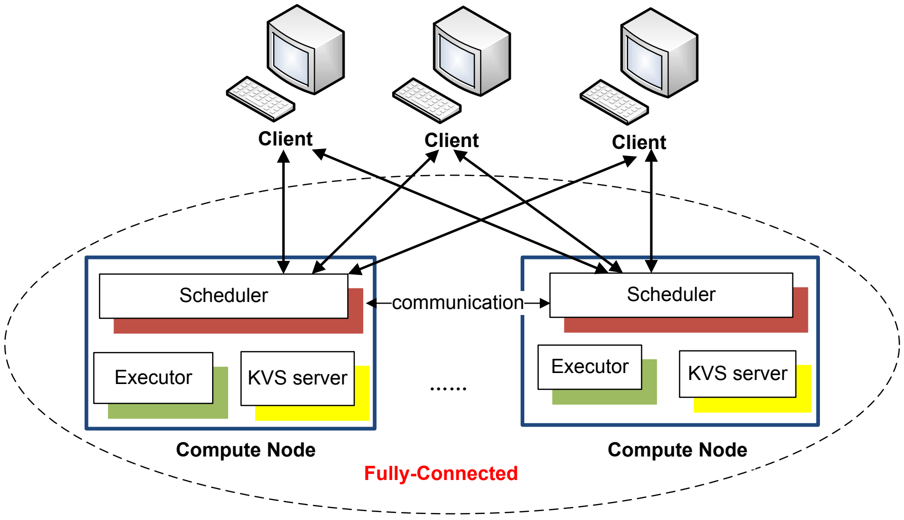


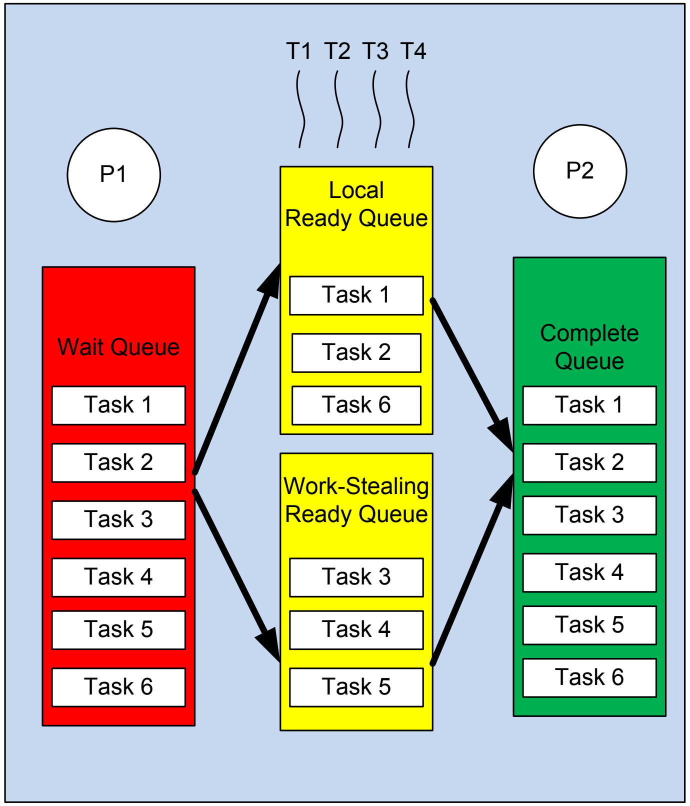
Figure 1. SLURM components



MATRIX: MAny-Task computing execution fabRIc at eXascale

MTC Distributed Scheduling

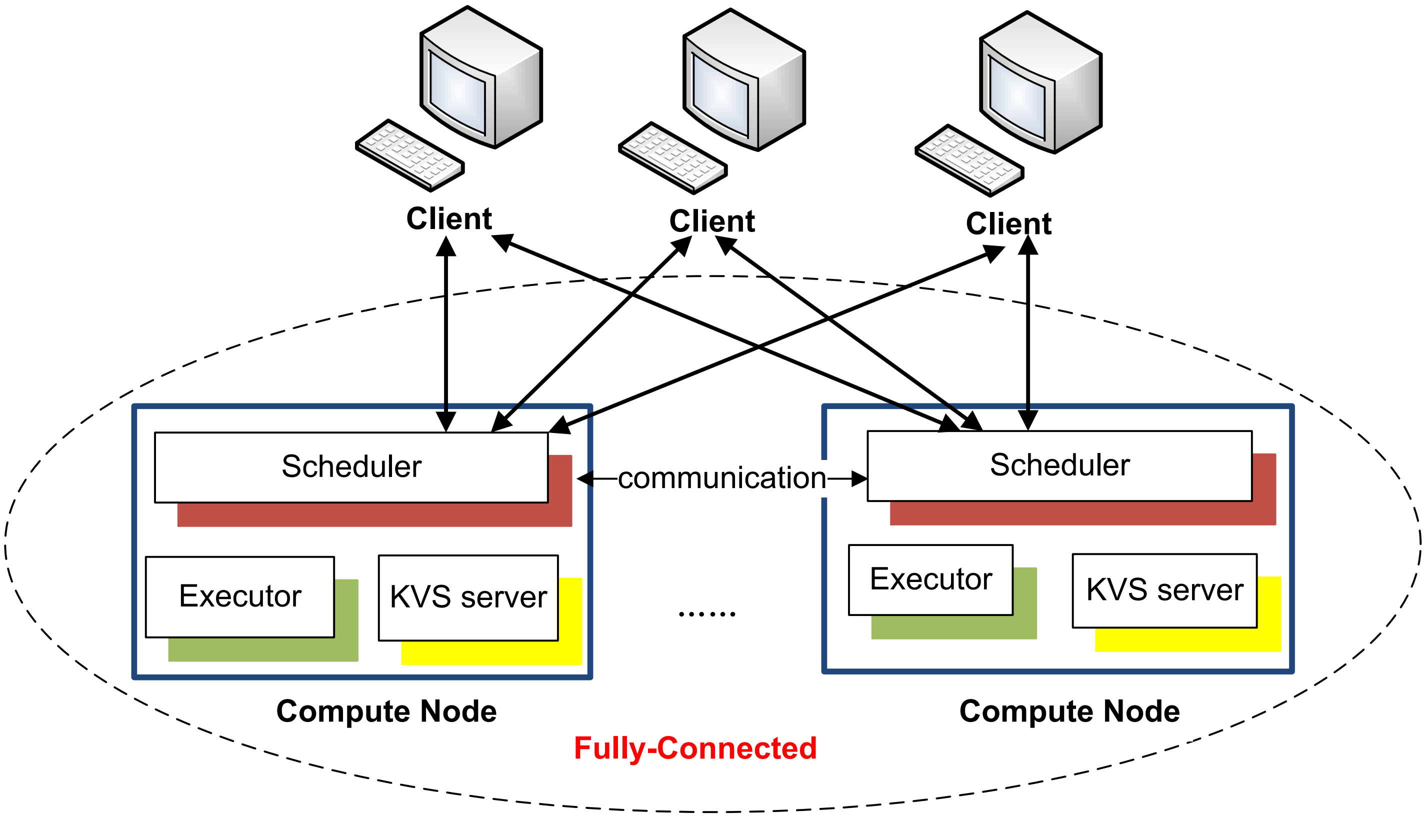




Scheduler Specification

MATRIX: MAny-Task computing execution fabRIc at eXascale

MTC Scheduling Architecture



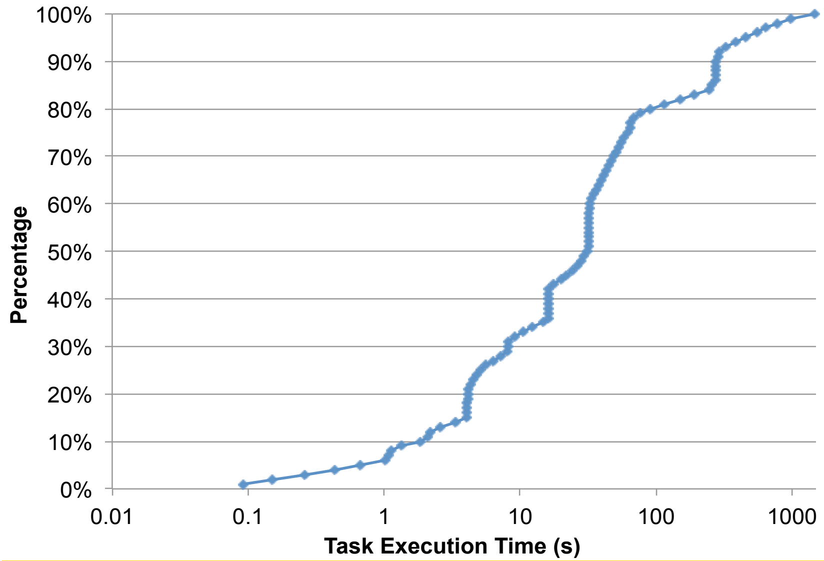
MATRIX: MAny-Task computing execution fabRIc at eXascale

MTC Distributed Scheduling Challenges

- **Fine-grained Workloads**
- **Load Balancing**
- **Data-Aware Scheduling**

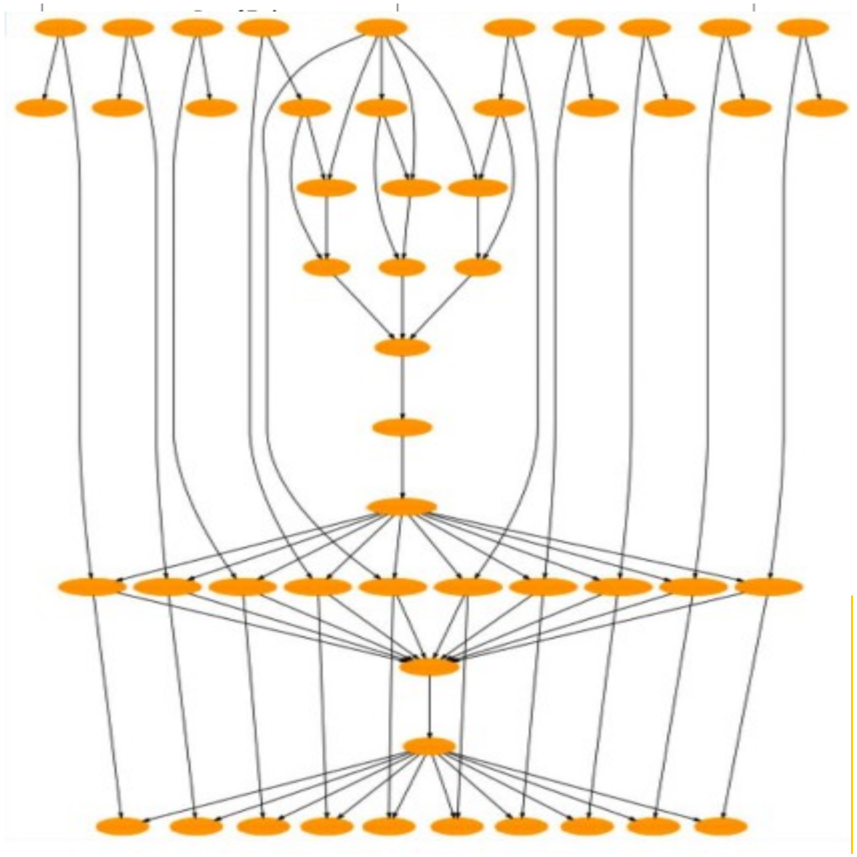
Fine-grained Workloads

MTC application trace



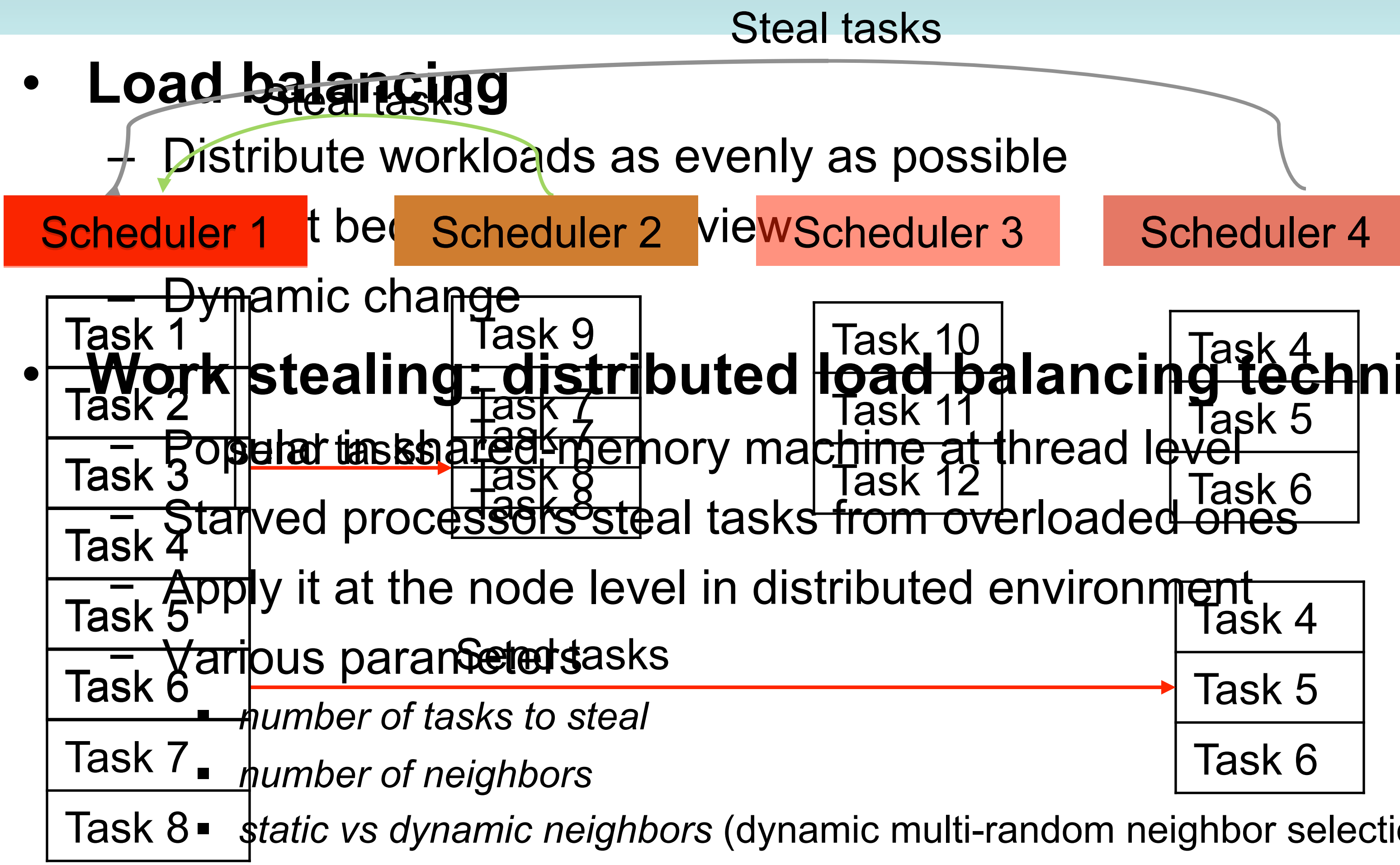
- **17-month period on IBM BG/P machine**
  - Minimum length: 0 sec
  - Maximum length: 1470 sec
  - Medium length: 30 sec
  - Average length: 95 sec

MTC application DAGs



Load Balancing

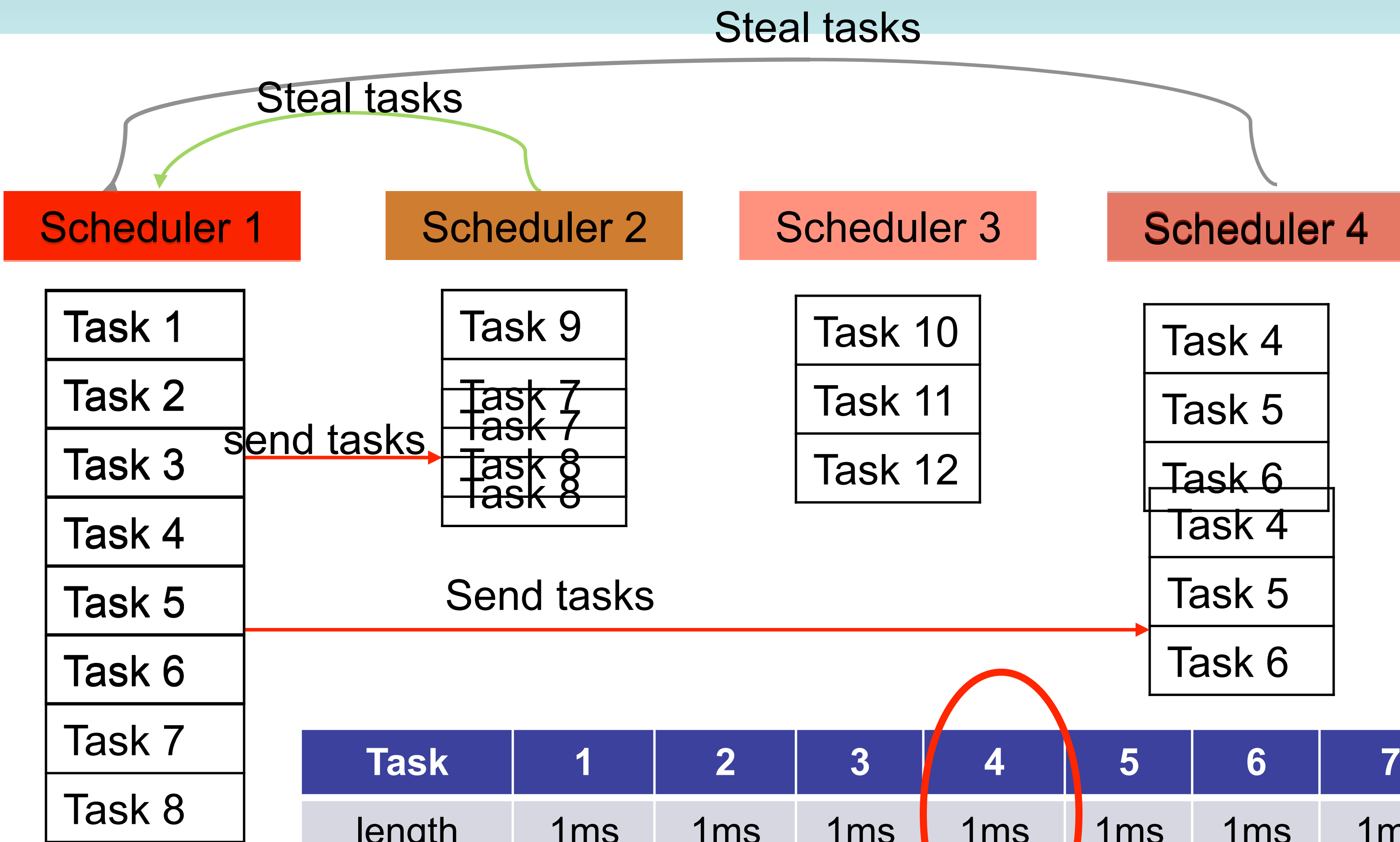
- **Load balancing**
  - Distribute workloads as evenly as possible
  - t be … view
  - Dynamic change
- **Work stealing: distributed load balancing technique**
  - Popular in shared-memory machine at thread level
  - Starved processors steal tasks from overloaded ones
  - Apply it at the node level in distributed environment
  - Various parameters
    - *number of tasks to steal*
    - *number of neighbors*
    - *static vs dynamic neighbors* (dynamic multi-random neighbor selection)
    - *polling interval* (exponential back-off)



Data-Aware Scheduling

- **Big-data era**
  - data-intensive applications
  - Data-flow driven programming models (MTC)
  - Workflow, task execution framework
- **Work Stealing**
  - Original work stealing is data locality-oblivious
  - Migrating tasks randomly comprise data-locality
  - Propose a Data-aware Work Stealing technique

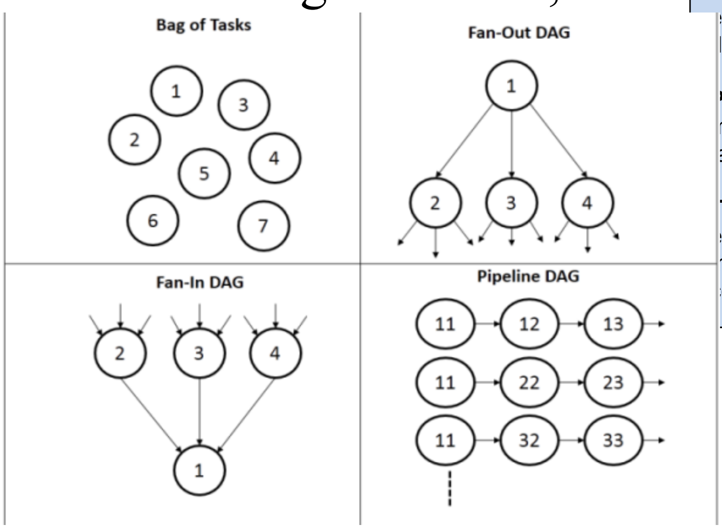
Locality-Oblivious Work Stealing



Task	1	2	3	4	5	6	7	8
length	1ms	1ms	1ms	1ms	1ms	1ms	1ms	1ms
Data size	5GB	20KB	1MB	20GB	30KB	0B	20KB	1MB

Data-Aware Work Stealing

```
typedef    TaskMetaData
{
    int    num_wait_parent;    // nu
    vector<string> parent_list;    //
    vector<string> data_object;    /
    vector<long> data_size;    // da
    long all_data_size;    // all data
    vector<string> children;    // ch
}
```



Wait Queue (WaitQ): holds tasks that are waiting for parents to complete

Dedicated Local Ready Queue (LReadyQ): holds ready tasks that can only be executed on local node

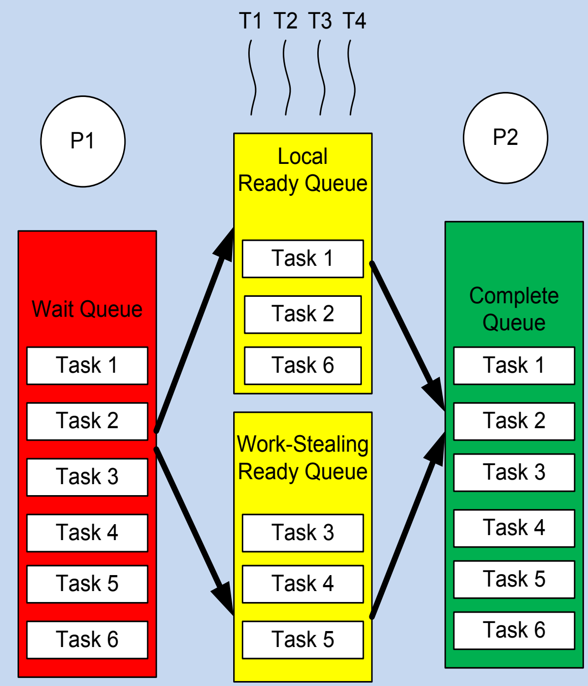
Shared Work-Stealing Ready Queue (SReadyQ): holds ready tasks that can be shared through work stealing

Complete Queue (CompleteQ): holds tasks that are completed

P1: a program that checks if a task is ready to run, and moves ready tasks to ither ready queue according to the lecision making algorithm

'2: a program that updates the task netadata for each child of a completed ask

'1 to T4: executor has 4 (configurable) xecuting threads that executes tasks in he ready queues and move a task to omplete queue when it is done



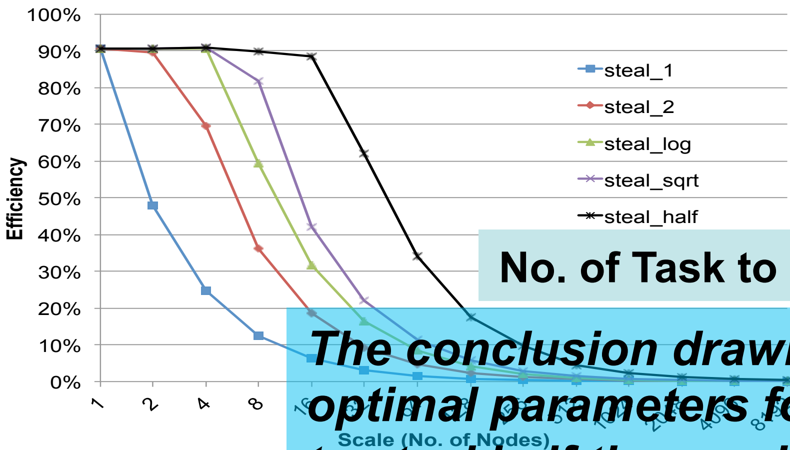
Scheduling Policies

- **MLB**
  - Maximized Load Balancing
- **MDL**
  - Maximized Data-Locality
- **RLDS**
  - Rigid Load balancing and Data-locality Segregation
- **FLDS**
  - Flexible Load balancing and Data-locality Segregation

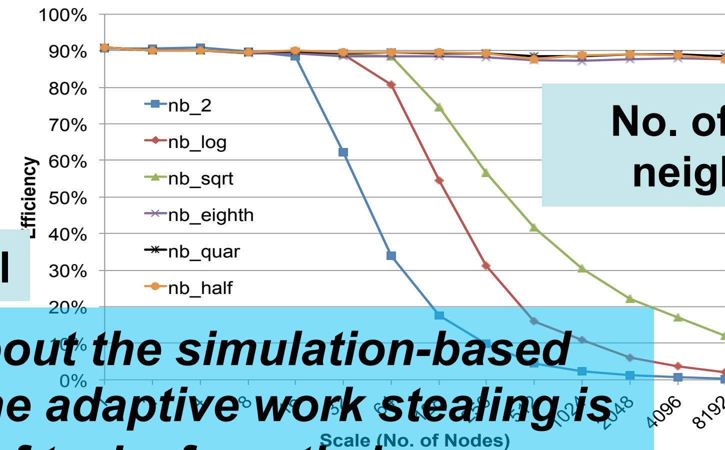
SimMatrix: SIMulator for MAny-Task computing execution fabRIc at eXascale

- **A discrete event simulator**
  - Simulates MTC task execution framework
  - 1M nodes, 1B cores, 100B tasks
- **Explore the scalability**
  - fully distributed MTC-architecture
  - work stealing technique
  - data-aware work stealing technique

Exploration of Work Stealing

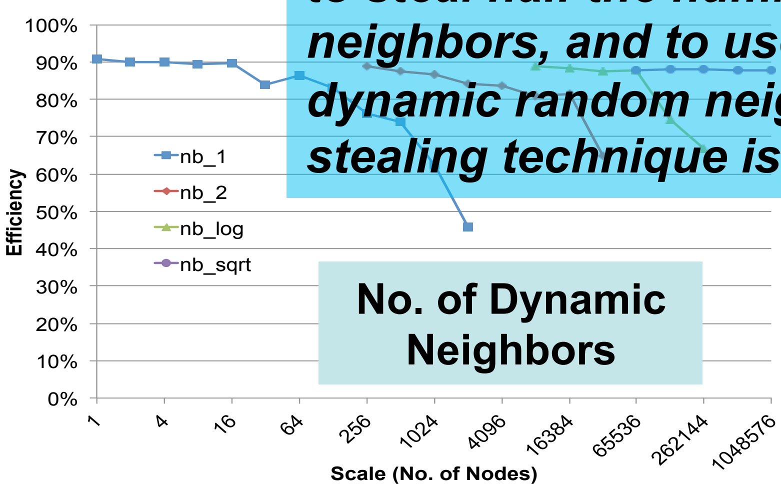


No. of Task to Steal

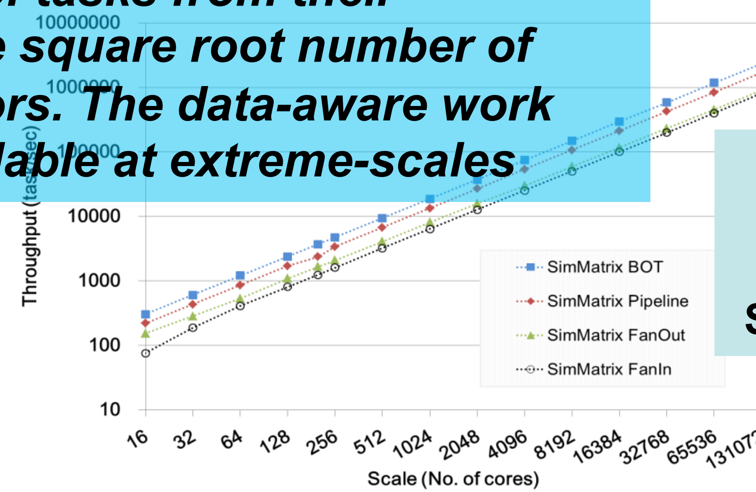


No. of Static neighbors

The conclusion drawn about the simulation-based optimal parameters for the adaptive work stealing is to steal half the number of tasks from their neighbors, and to use the square root number of dynamic random neighbors. The data-aware work stealing technique is scalable at extreme-scales



No. of Dynamic Neighbors



Data-aware Work Stealing

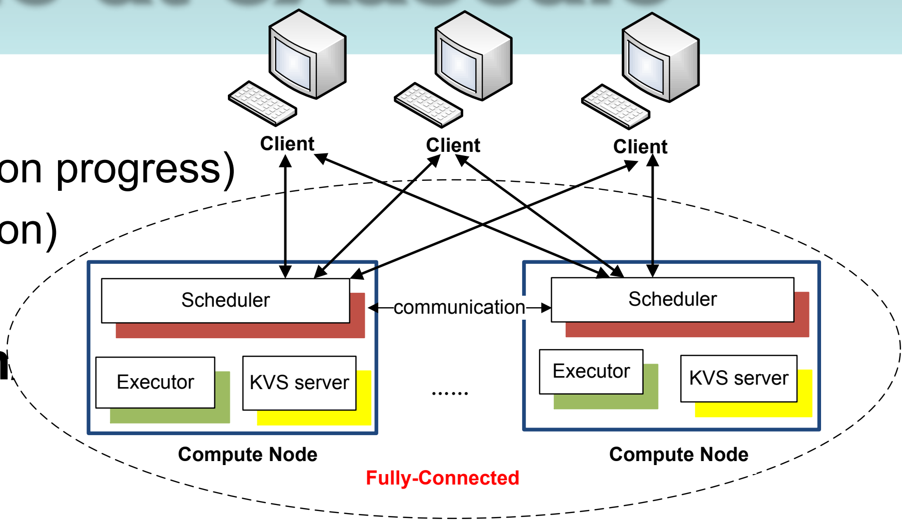
MATRIX: MAny-Task computing execution fabRIc at eXascale

Simulations have shown that the distributed architecture and the work stealing technique are scalable, now we can do real implementations

MATRIX: TASK EXECUTION FRAMEWORK FOR MTC

MATRIX: MAny-Task computing execution fabRIc at eXascale

- **MATRIX components**
  - Client (submit tasks, monitoring execution progress)
  - Scheduler (scheduling tasks for execution)
  - Executor (execute tasks)
- **MATRIX code (https://github.com**
  - 5K lines of C++ code
  - 8K lines of ZHT code
  - 1K lines of auto-generated code from Google protocol buffer

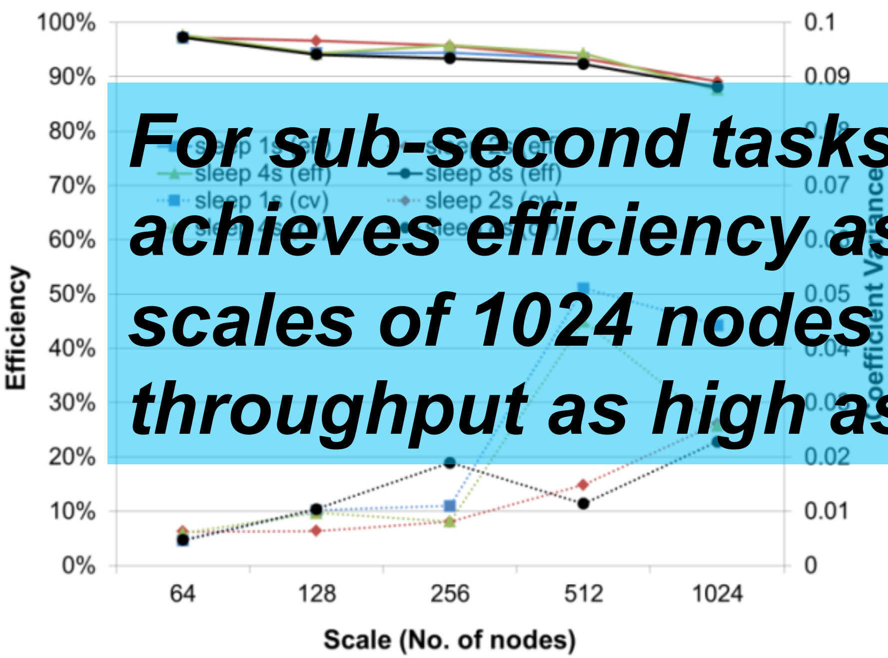


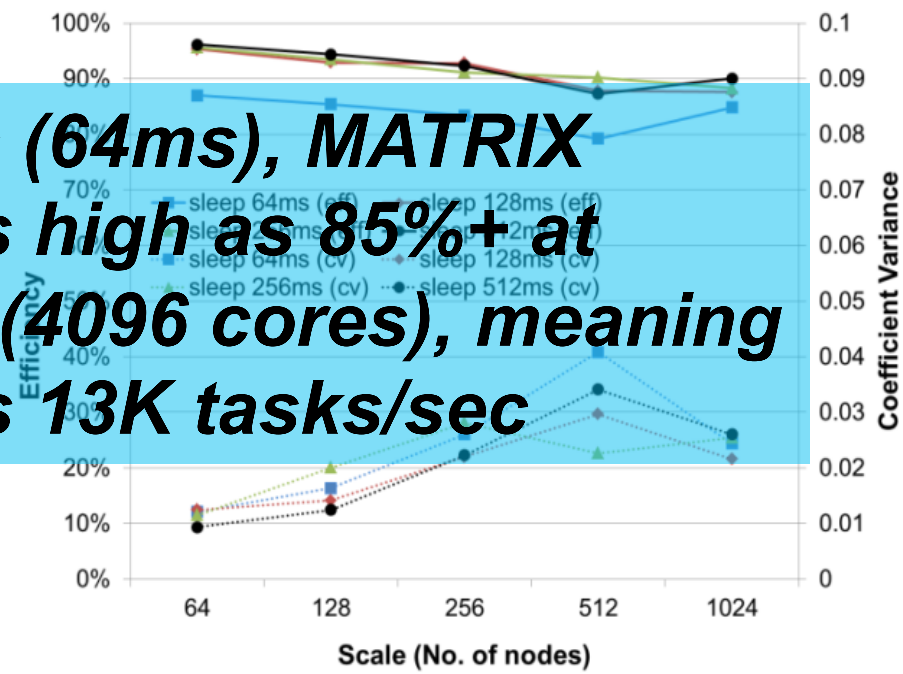
- **MATRIX goal**
  - Scalable distributed scheduling system for MTC applications
  - The prototype has been finished
  - working on running real applications
- **Testbeds:**
  - IBM Blue Gene/P/Q supercomputers (4K cores)
  - Probe Kodiak cluster (200 cores)
  - **Amazon EC2 (256 cores)**

Outline

- Introduction & Motivation
- Problem Statement
- Proposed Work
- **Evaluation**
- Conclusions
- Future Work

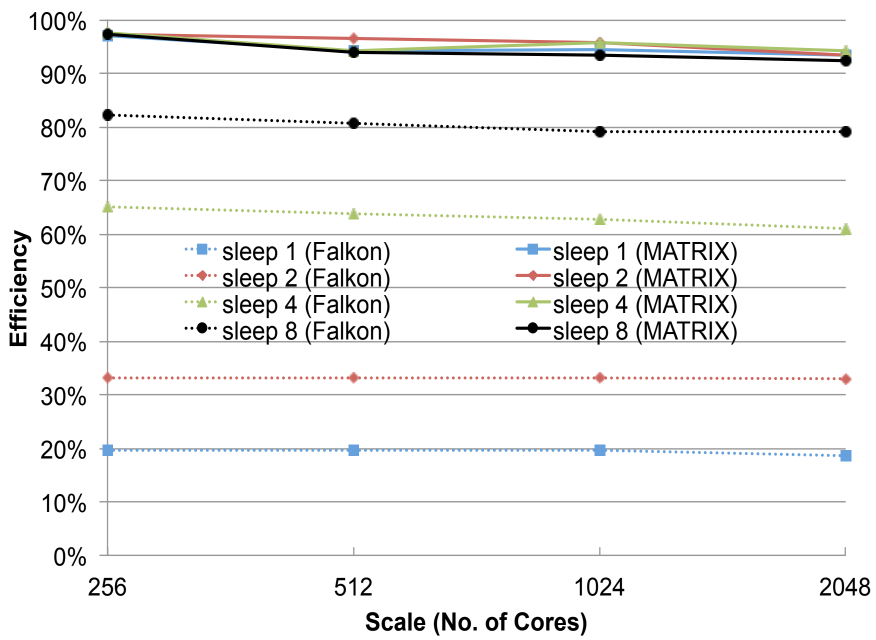
MATRIX running fine-grained tasks

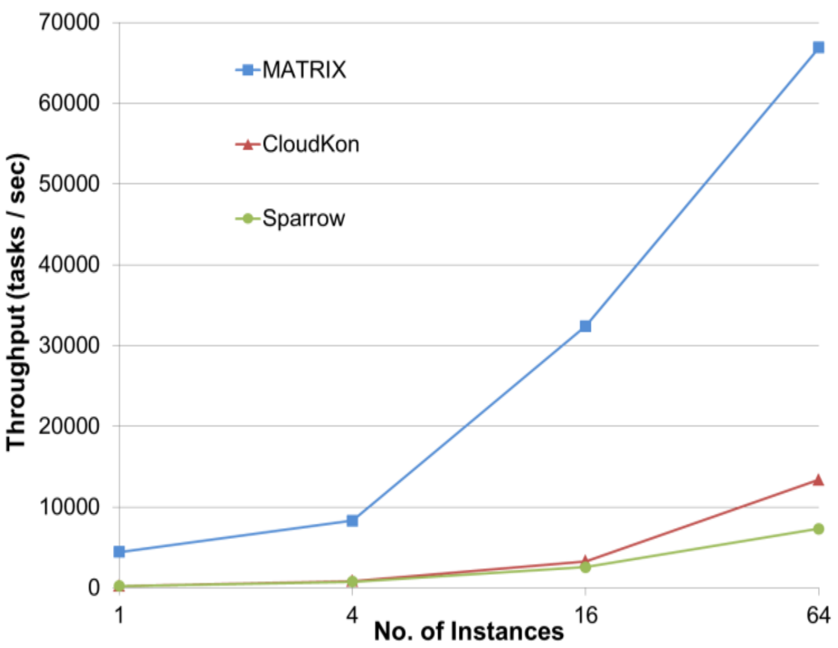




For sub-second tasks (64ms), MATRIX achieves efficiency as high as 85%+ at scales of 1024 nodes (4096 cores), meaning throughput as high as 13K tasks/sec

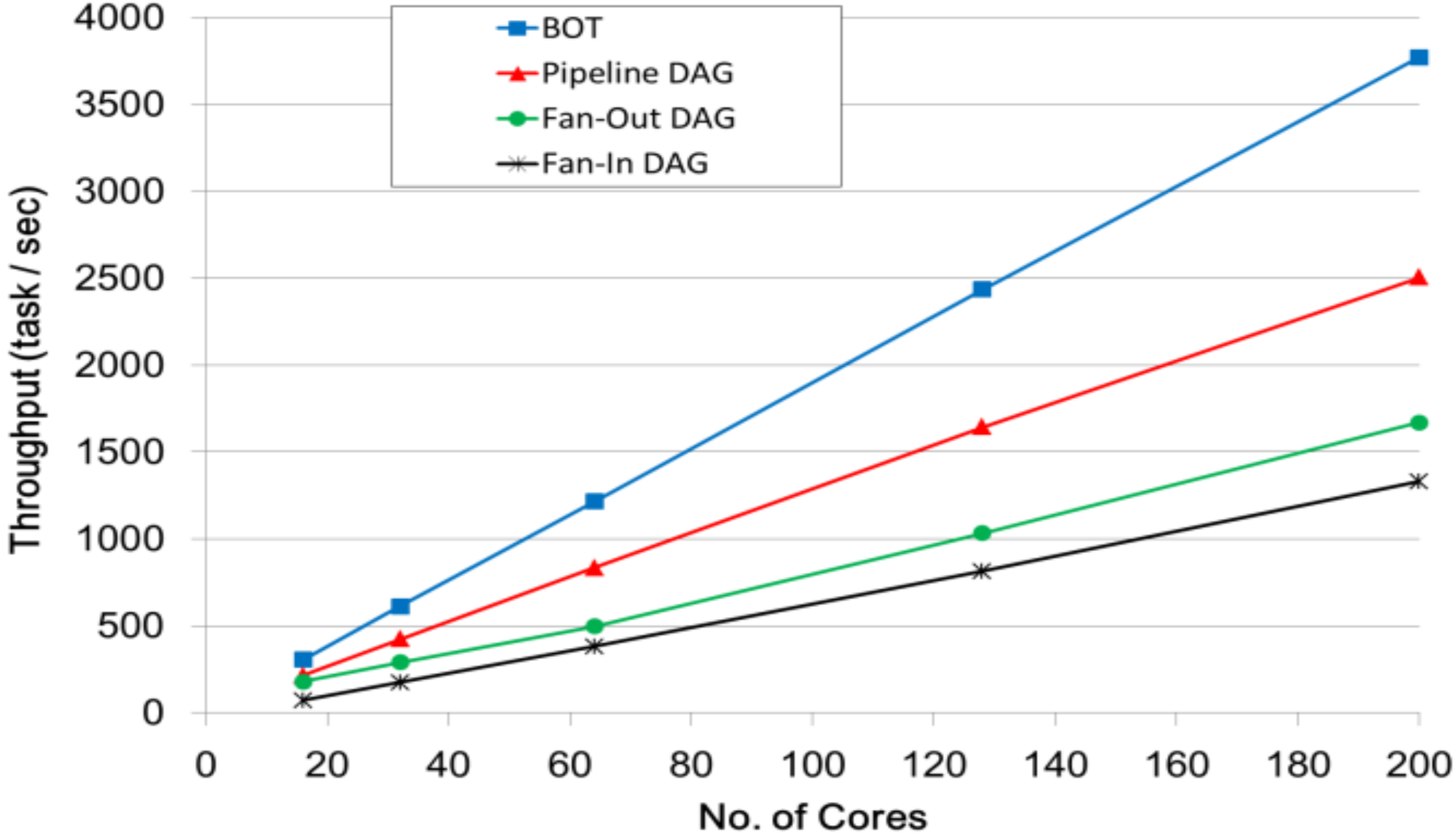
MATRIX comparisons





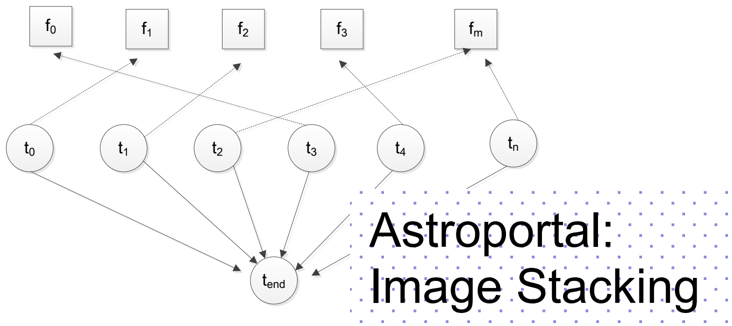
MATRIX: MAny-Task computing execution fabRIc at eXascale

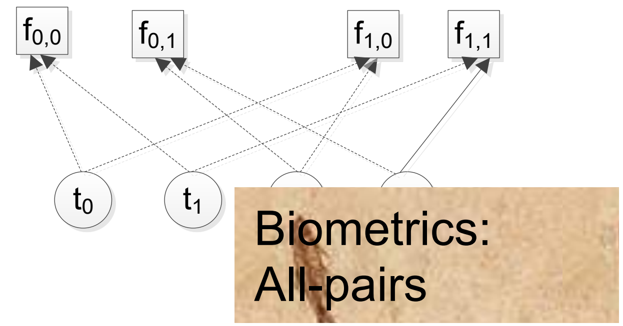
MATRIX running workload DAGs

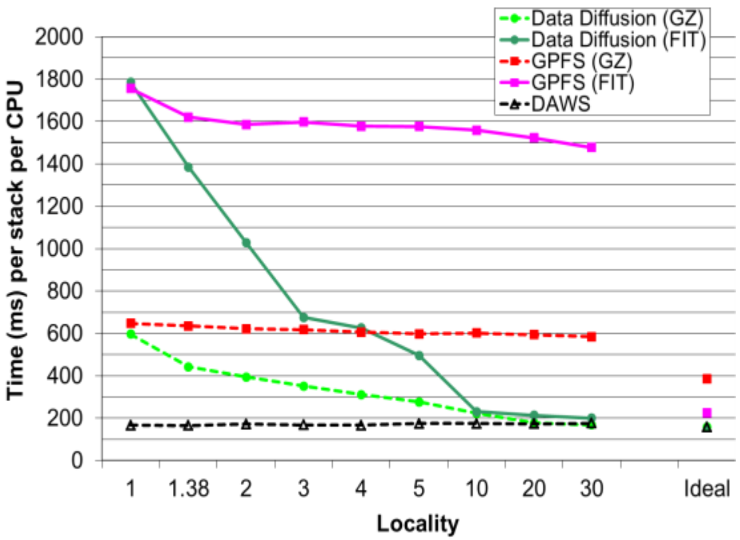


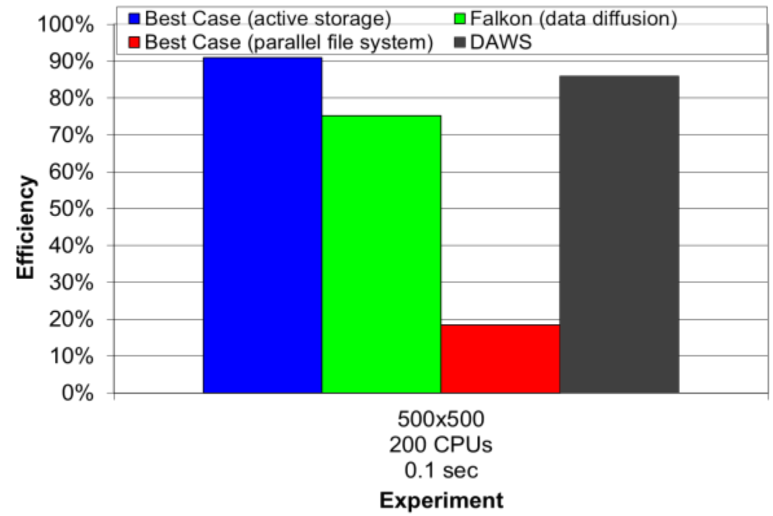
MATRIX: MAny-Task computing execution fabRIc at eXascale

MATRIX running data-intensive applications



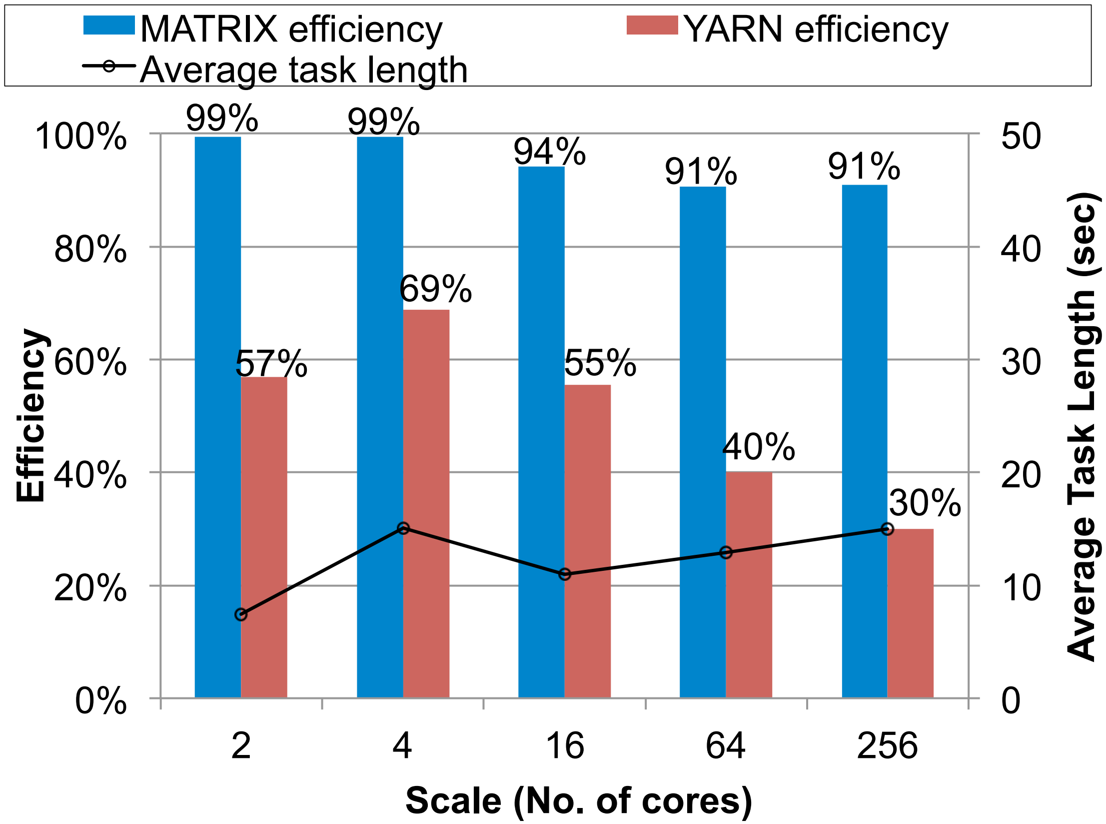






MATRIX: MAny-Task computing execution fabRIc at eXascale

MATRIX vs. Hadoop YARN



- **Bio-Informatics Application**
- **Protein-ligand Clustering**
- **5 Phases of MapReduce Jobs**
- **256MB data per node**
- **First phase has the majority of tasks**

Outline

- Introduction & Motivation
- Problem Statement
- Proposed Work
- Evaluation
- **Conclusions**
- Future Work

Conclusions

- System scale is approaching exascale
- Applications are becoming fine-grained
- Next-generation resource management systems need to be highly scalable, efficient and available
- Fully distributed architectures are scalable
- Data-aware work stealing technique is scalable

Outline

- Introduction & Motivation
- Problem Statement
- Proposed Work
- Evaluation
- Conclusions
- **Future Work**

Future Work

- **Hadoop Integration of MATRIX**
  - Hadoop scheduler is centralized
  - Replace the Hadoop scheduler with MATRIX
- **Workflow integration of MATRIX**
  - Swift workflow system
  - Will enable the execution of real scientific applications
- **File System integration of MATRX**
  - FusionFS distributed file system
  - Take care of the data storage and management
  - Expose the data to MATRIX

More Information

- More information:
  - http://datasys.cs.iit.edu/~kewang/
- Contact:
  - kwang22@hawk.iit.edu
- Questions?