MOHA: Many-Task Computing meets the Big Data Platform

Jik-Soo Kim*, Cao Nguyen*†, Soonwook Hwang*†

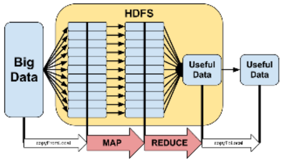
* National Institute of Supercomputing and Networking at KISTI
Daejeon, Republic of Korea
Email: {jiksoo.kim, cao, hwang}@kisti.re.kr

† University of Science & Technology (UST)
Daejeon, Republic of Korea

Table of Contents

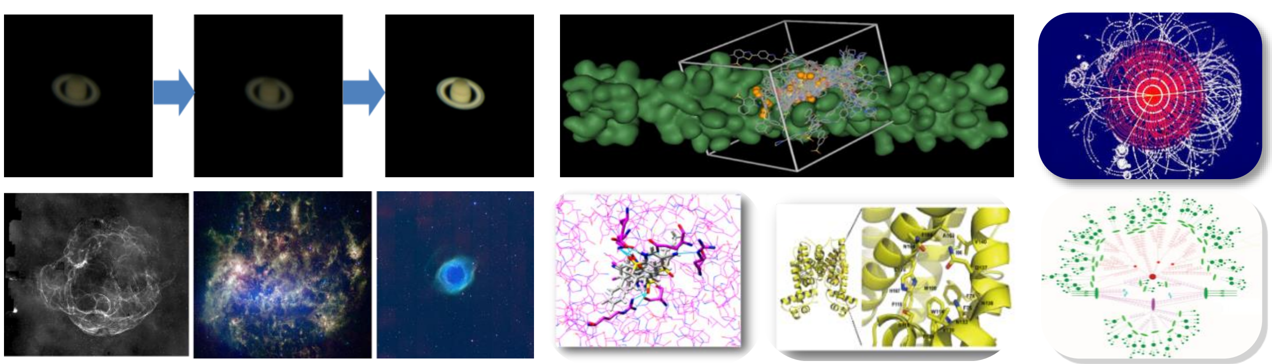
Introduction

- **Distributed/Parallel computing systems to support various types of challenging applications**
  - **HTC (High-Throughput Computing)** for relatively long running applications consisting of *loosely-coupled* tasks
  - **HPC (High-Performance Computing)** targets efficiently processing *tightly-coupled parallel* tasks
  - **DIC (Data-intensive Computing)** mainly focuses on effectively leveraging *distributed storage systems* and *parallel processing frameworks*



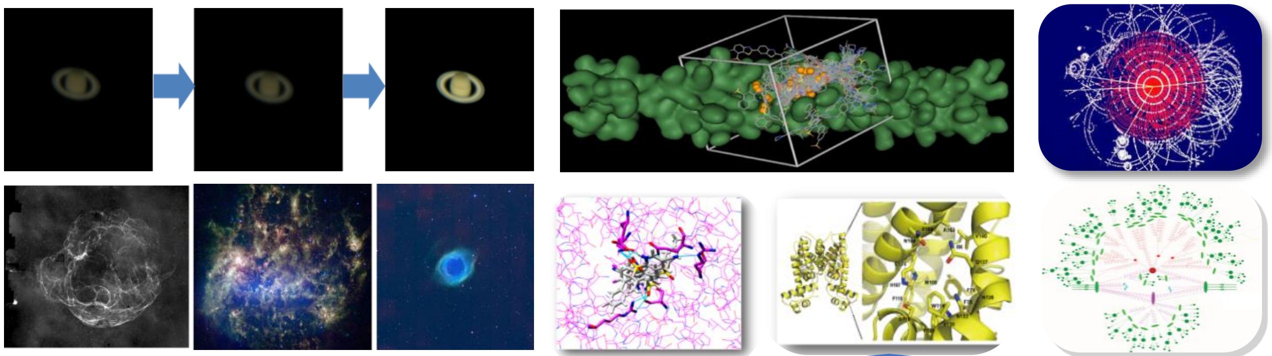
Introduction

- **Many-Task Computing (MTC) as a *new computing paradigm* [I. Raicu, I. Foster, Y. Zhao, MTAGS'08]**
  - A very **large** number of tasks (millions or even billions)
  - Relatively **short** per task execution times (sec to min)
  - **Data** intensive tasks (i.e., tens of MB of I/O per second)
  - A large **variance** of task execution times (i.e., ranging from hundreds of milliseconds to hours)
  - Communication-intensive, however, not based on message passing interface but through **files**



astronomy, physics, pharmaceuticals, chemistry, etc.

Introduction



Many-Task Computing Applications

astronomy, physics, pharmaceuticals, chemistry, etc.

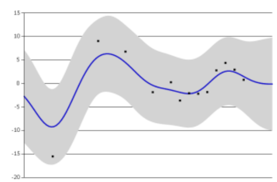
A very **large** # of tasks

millions or even billions

Relatively **short** per task execution time

seconds to minutes

A large **variance** of task execution times



from hundreds of milliseconds to hours

Data intensive tasks

tens of MB of I/O per second

Communication through **files**

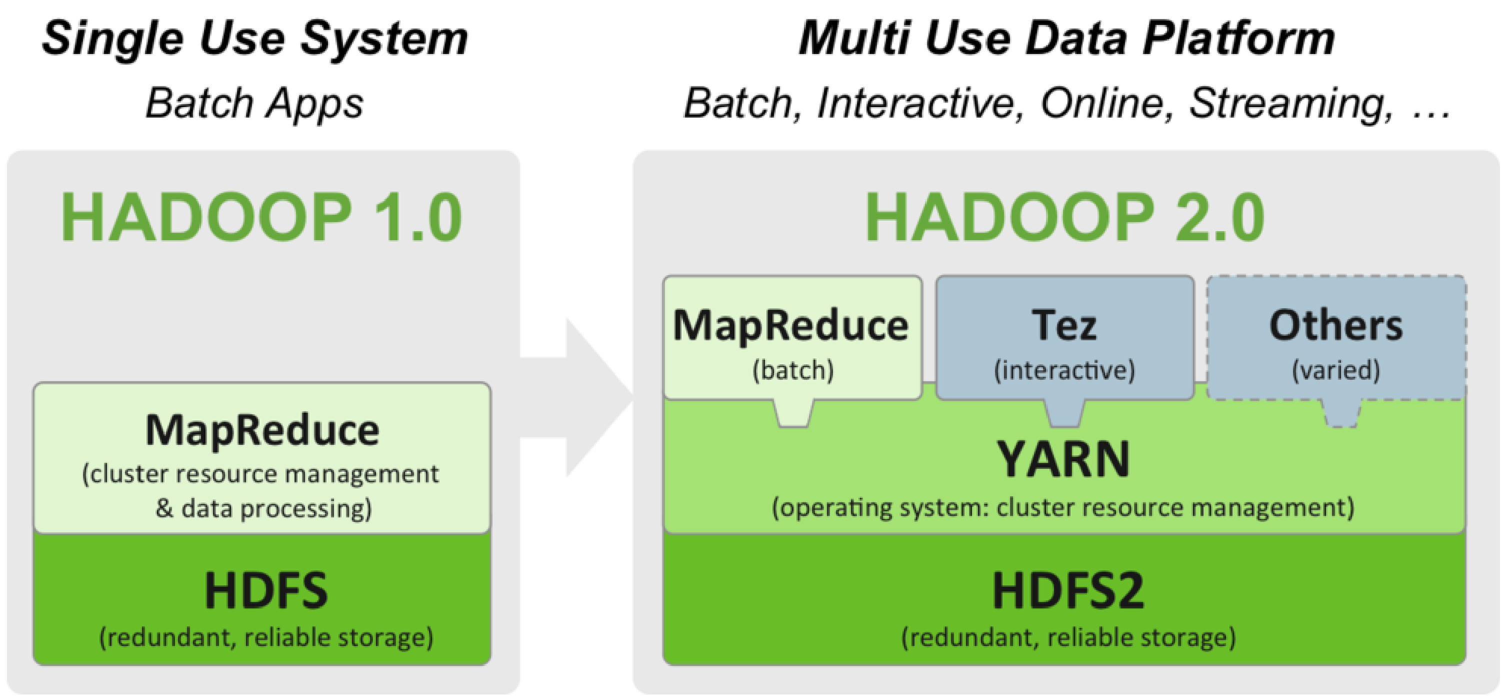
High-Performance Task Dispatching

Dynamic Load Balancing

***Another Type* of Data-intensive Workload**

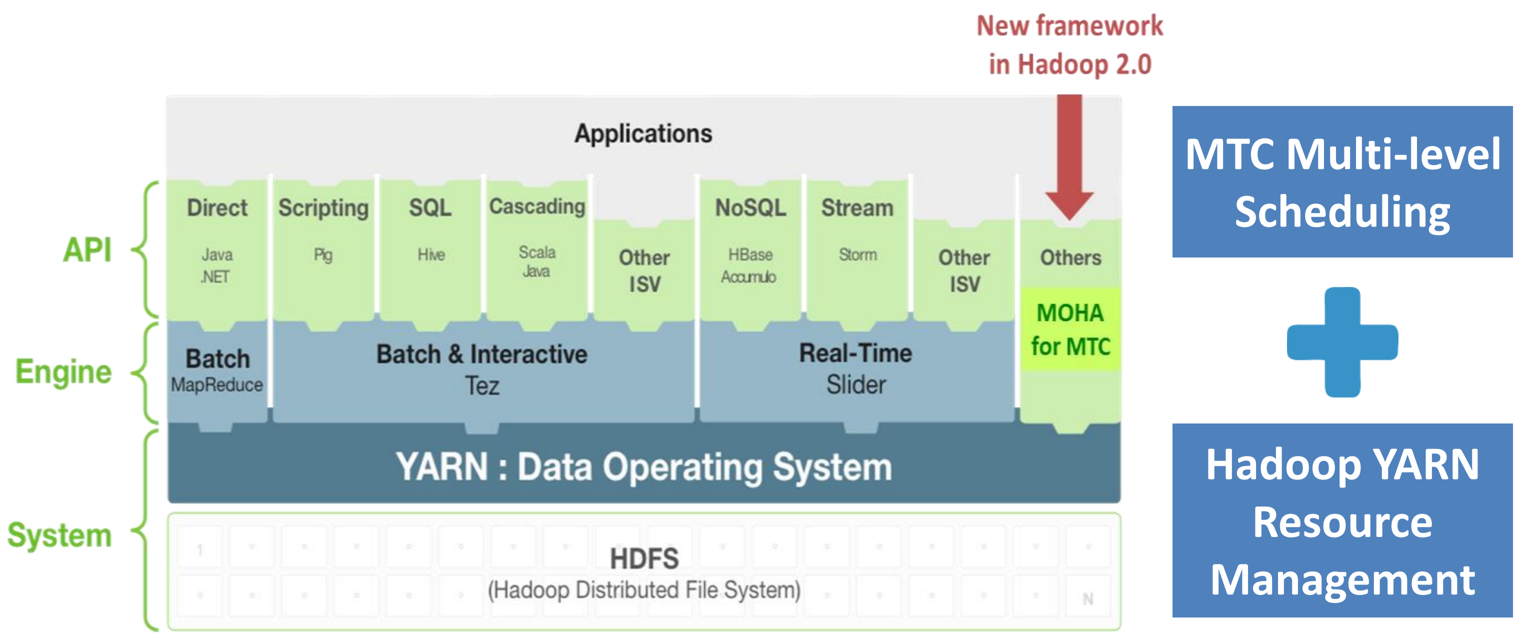
Introduction

- **Hadoop, the *de facto standard* "Big Data" store and processing infrastructure**
  - with the advent of **Apache Hadoop YARN**, Hadoop 2.0 is evolving into **multi-use data platform**
    - ✓ harness *various* types of data processing workflows
    - ✓ *decouple* application-level scheduling and resource management



Introduction

- **This paper presents**
  - **MOHA (Many-task computing On HAdoop)** framework which can effectively *combine* Many-Task Computing technologies with the existing Big Data platform Hadoop
    - ✓ developed as one of *Hadoop YARN applications*
    - ✓ transparently *cohost* existing MTC applications with other Big Data processing frameworks in a single Hadoop cluster



Related Work

- **GERBIL: MPI+YARN** **[L. Xu , M. Li, A. R. Butt, CCGrid'15]**
  - A framework for transparently co-hosting unmodified **MPI applications** alongside MapReduce applications
    - ✓ exploits YARN as the model agnostic resource negotiator
    - ✓ provides an easy-to-use interface to the users
    - ✓ allows realization of **rich data analytics workflows** as well as efficient data sharing between the MPI and MapReduce models within a single cluster

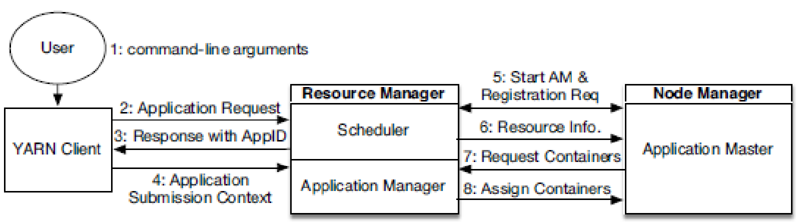


Fig. 1. Steps of launching an application in YARN.

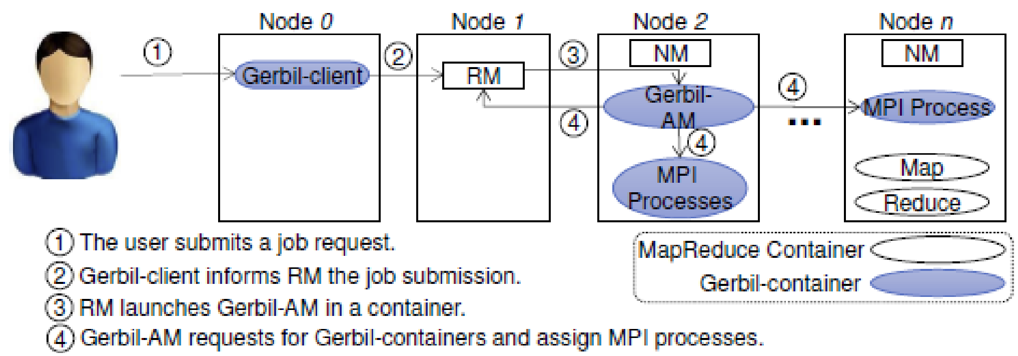


Fig. 2. GERBIL architecture for running MPI on YARN.

Related Work

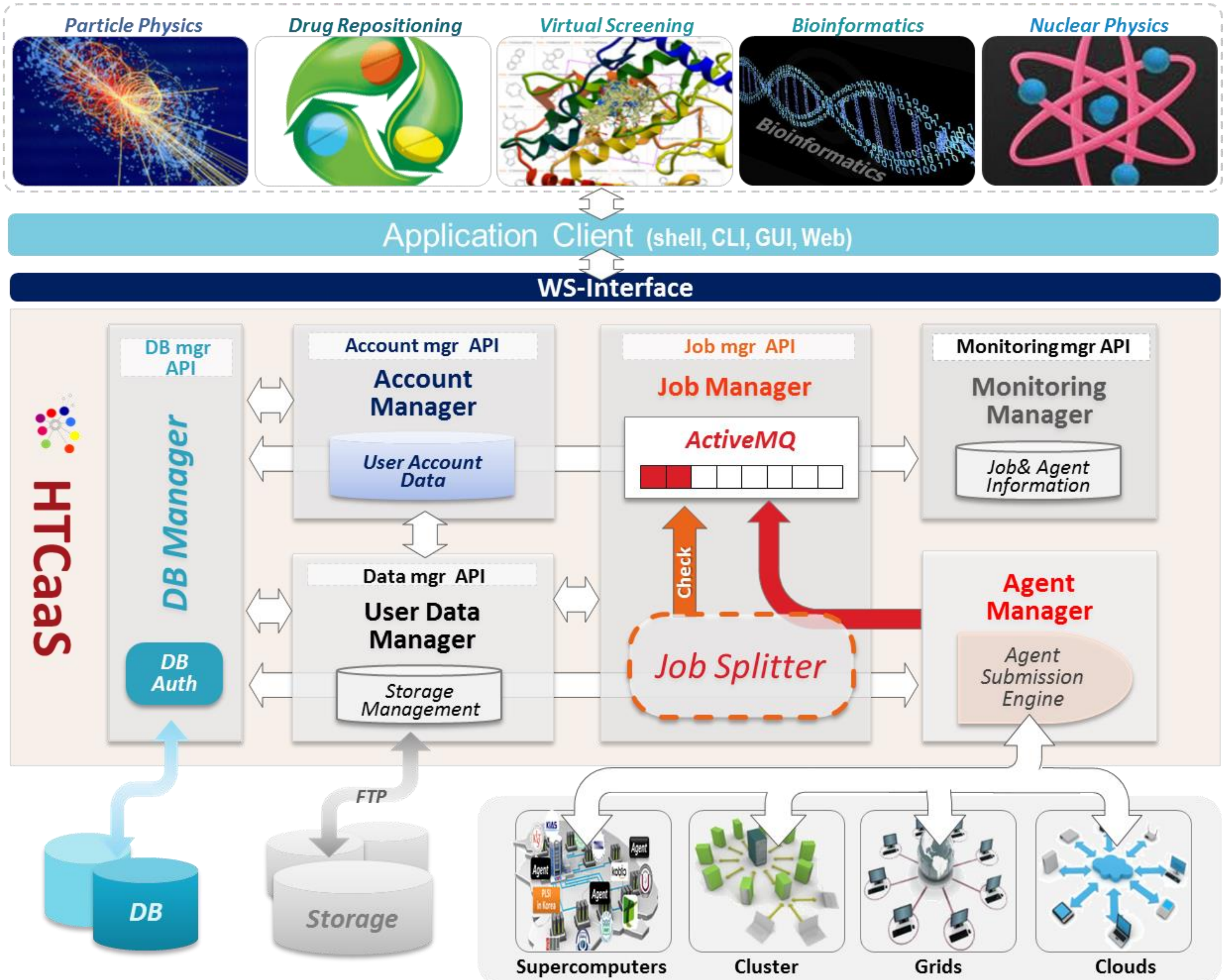
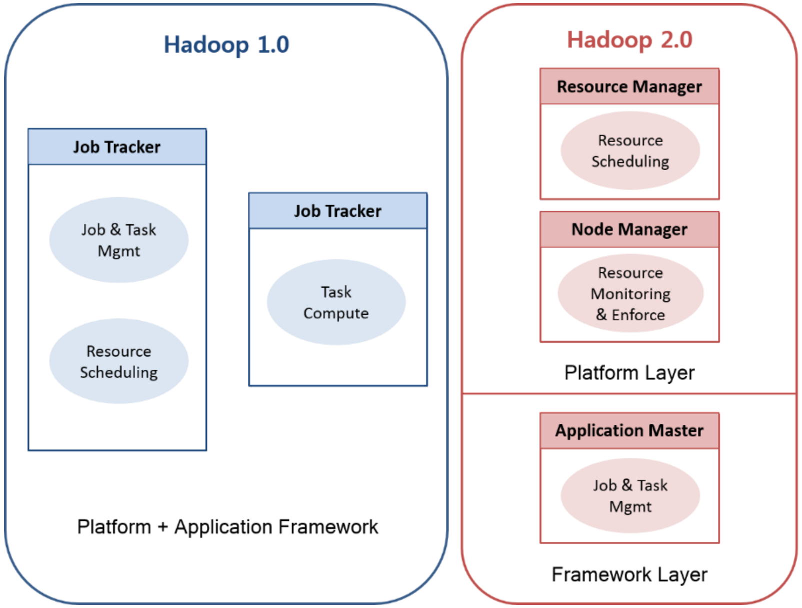
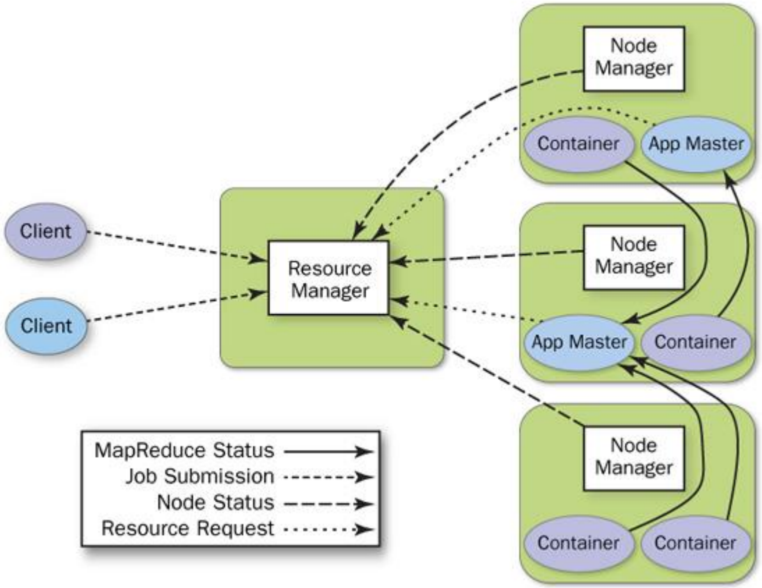


Table of Contents

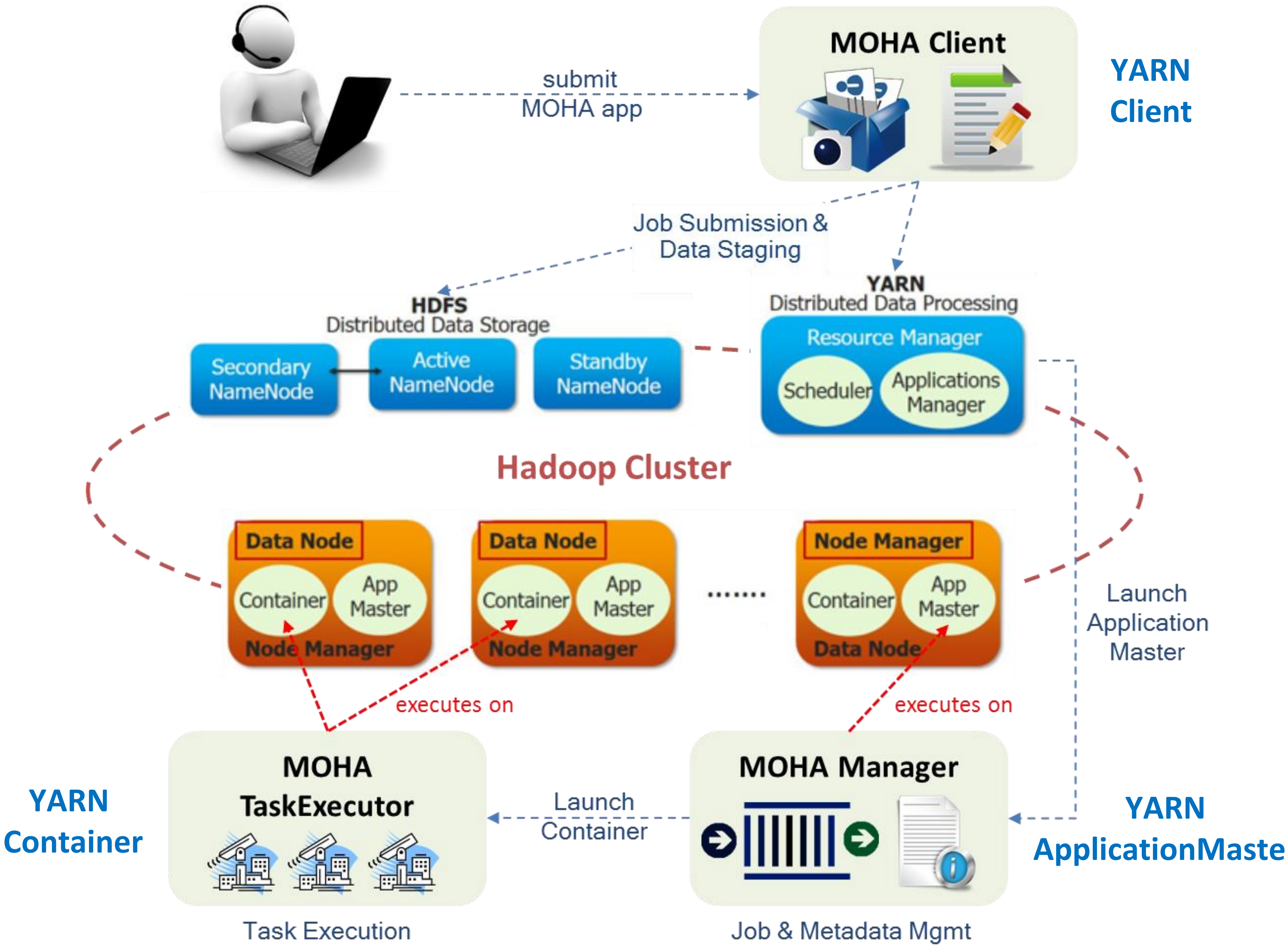
Hadoop YARN Execution Model

- **YARN separates all of its functionality into *two* layers**
  - **platform layer** is responsible for resource management (*first-level scheduling*)
    - ✓ **Resource Manager, Node Manager**
  - **framework layer** coordinates application execution (*second-level scheduling*)
    - ✓ **ApplicationMaster → New MOHA Framework !**





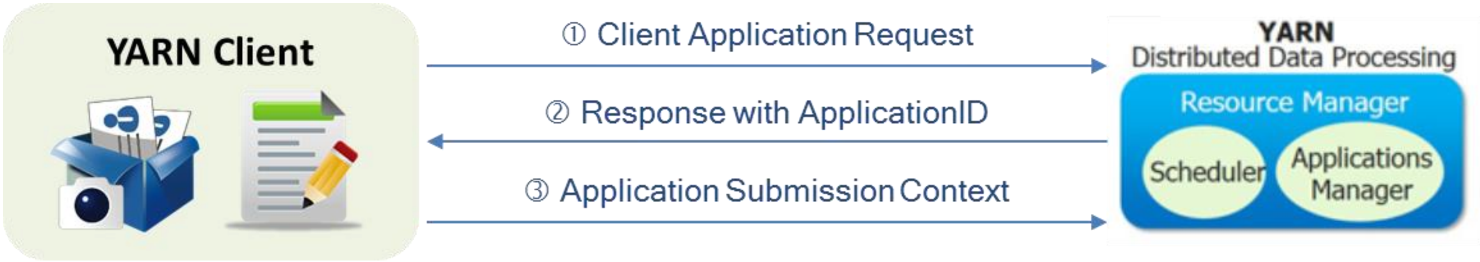
MOHA System Architecture



MOHA System Architecture

▪ MOHA Client

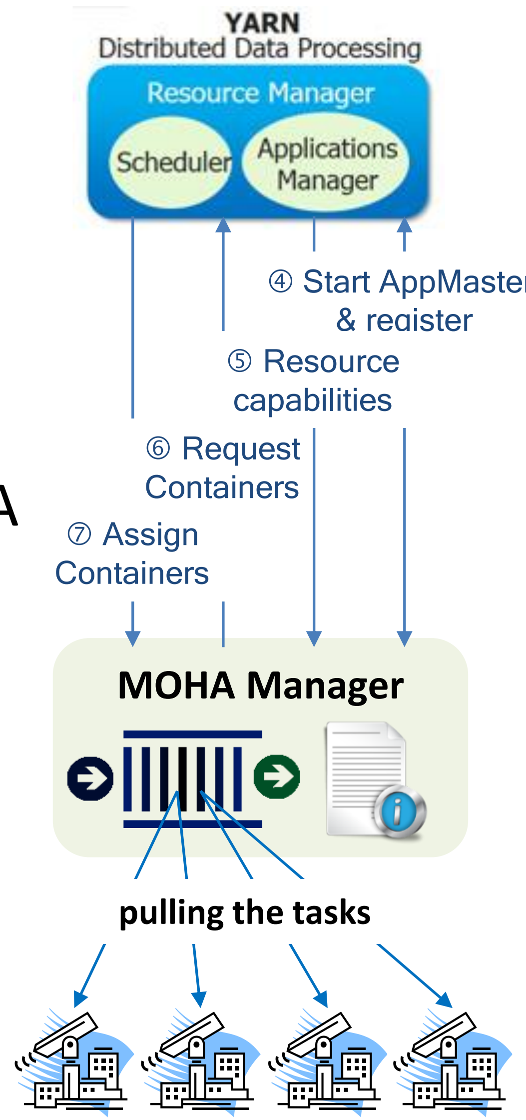
- **submit a MOHA job** and performs **data staging**
  - ✓ A MOHA job is *a bag of tasks* (i.e., a collection of multiple tasks)
    - ▪ provides a simple JDL(Job Description Language)
  - ✓ upload required data into the HDFS
    - ▪ application input data, application executable, MOHA JAR, JDL etc.
- prepare an execution environment for the MOHA Manager based on *YARN's Resource Localization Mechanism*
  - ✓ required data are *automatically* downloaded and prepared for use in the local working directories of containers by the NMs



MOHA System Architecture

- **MOHA Manager**
  - create and launch **MOHA job queues**

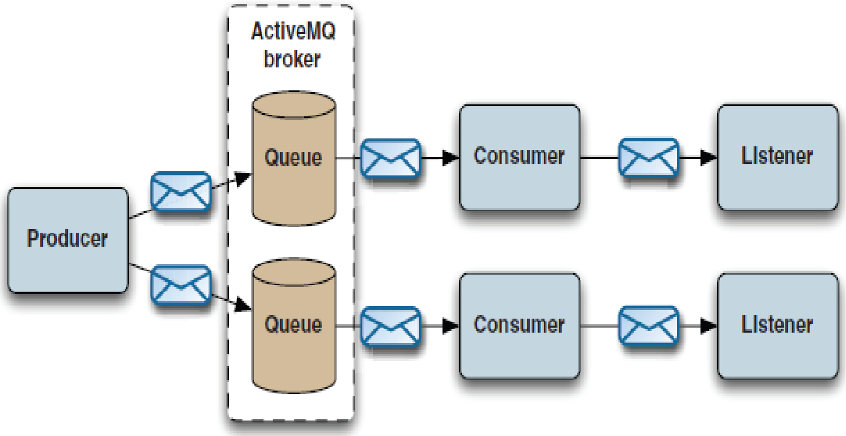
    Apache Kafka
    A high-throughput distributed messaging system.
    ActiveMQ
  - **split** a MOHA job into multiple tasks and insert them into the queue
  - get containers **allocated** and **launch** MOHA TaskExecutors
- **MOHA TaskExecutor**
  - **pull** the tasks from the MOHA job queues and process them
    - ✓ monitor and report the task execution



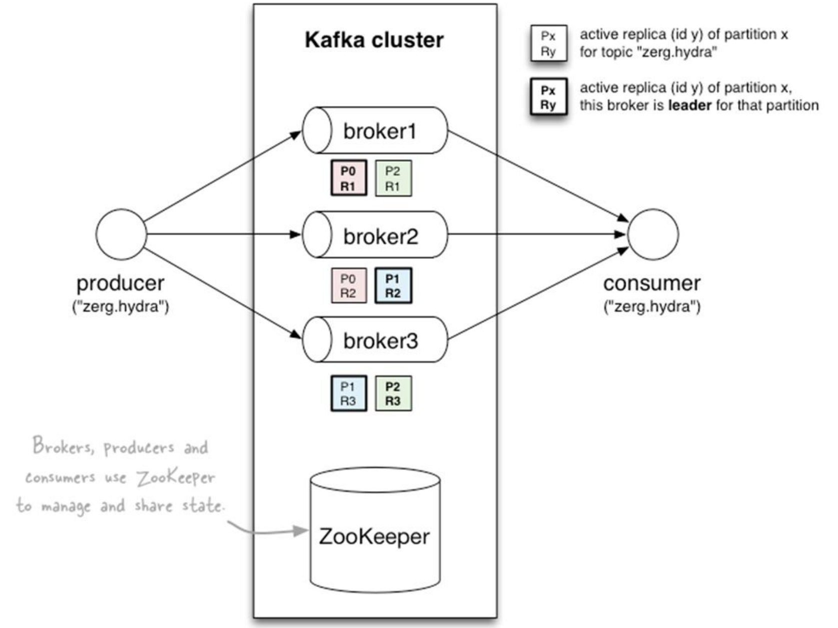
"Multi-level Scheduling Mechanism"

MOHA System Architecture

- **Apache ActiveMQ**
  - a message broker in Java that supports AMQP protocol
  - does not support any message delivery guarantee
  - **cannot scale very well** in larger systems



- **Apache Kafka**
  - an open source, distributed publish and consume service introduced by **LinkedIn**
  - gathers the logs from a large number of servers, and feeds it into HDFS or other analysis clusters
  - **fully distributed** and provides **high throughput**

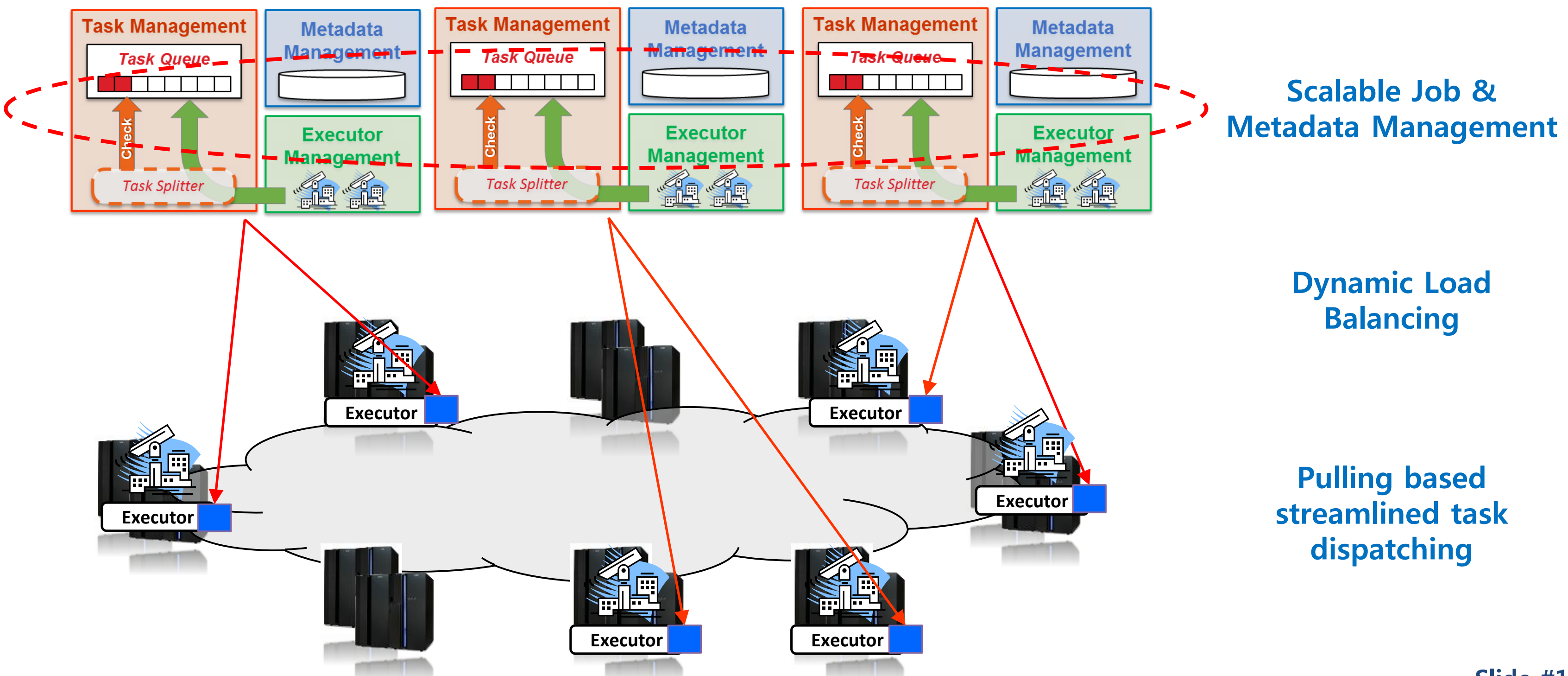


Discussion

- **MTC applications typically require**
  - much *larger* numbers of tasks
  - relatively *short* task execution times
  - substantial amount of *data* operations with potential interactions through *files*
  - ☑ high-performance task dispatching
  - ☑ effective dynamic load balancing
  - ☑ data-intensive workload support
  - ⇨ **"seamless integration"**
- **Hadoop can be a viable choice for addressing these challenging MTC applications**
  - technologies from MTC community should be effectively *converged* into the ecosystem

Discussion

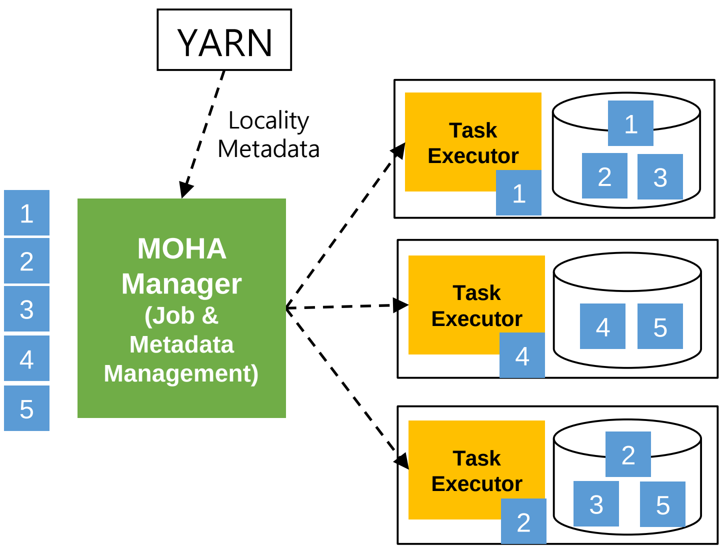
- **Potential Research Issues**
  - Scalable Job/Metadata Management
    - ✓ removing potential performance bottleneck
  - Dynamic Task Load Balancing
    - ✓ Task bundling and Job profiling techniques

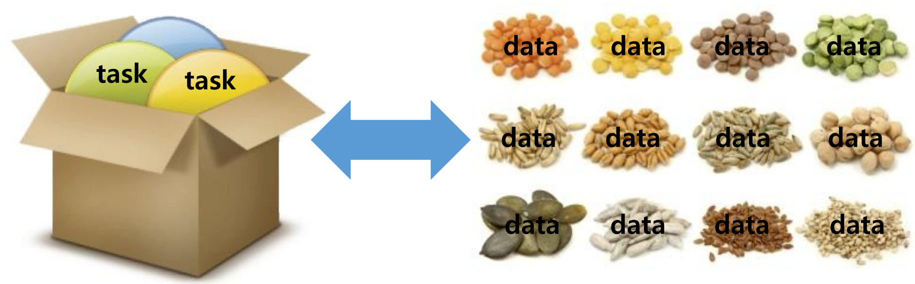


Discussion

▪ Potential Research Issues

- Data-aware resource allocation
  - ✓ leveraging Hadoop's data locality (computations *close* to data)
- Data Grouping & Declustering
  - ✓ aggregating a groups of small files ("data bundle")



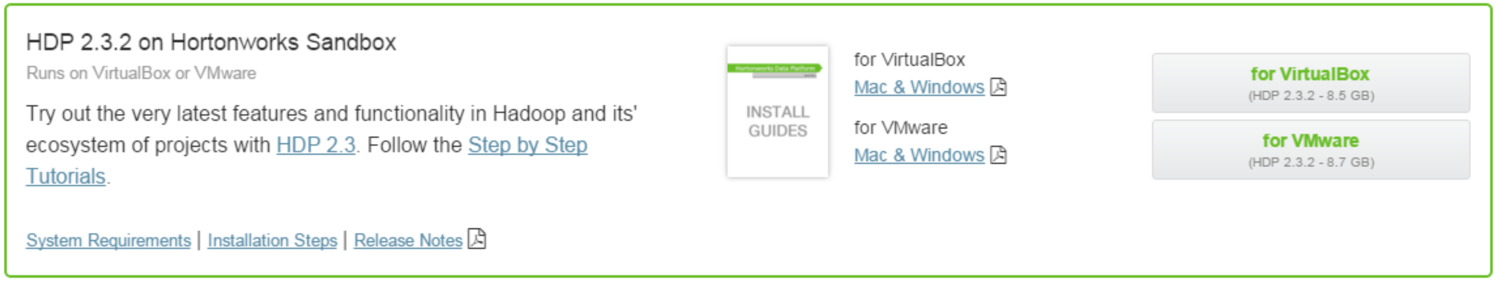


Task Bundling & Data Grouping can be closely related

Table of Contents

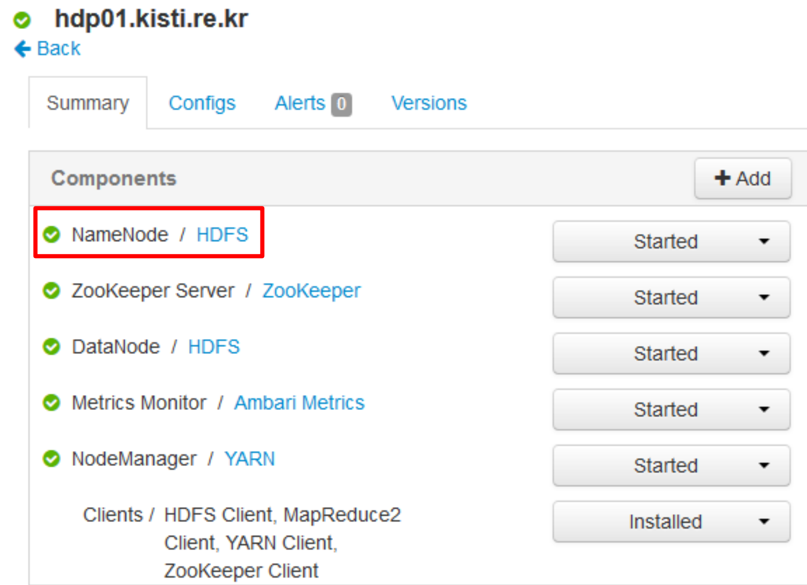
Experimental Setup

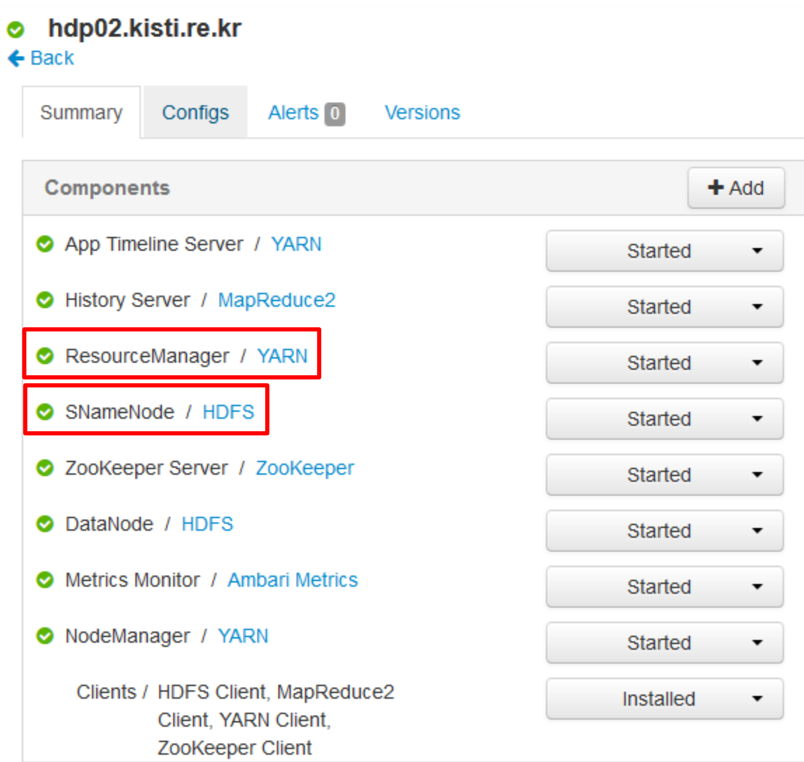
- **MOHA Testbed**
  - consists of 3 rack mount servers
    - ✓ 2 * Intel Xeon E5-2620v3 CPUS (12 CPU cores)
    - ✓ 64GB of main memory
    - ✓ 2 * 1TB SATA HDD (1 for Linux, 1 for HDFS)
  - Software stack
    - ✓ **Hortonworks Data Platform (HDP) 2.3.2**
      - automated install with Apache **Ambari**
    - ✓ Operating Systems Requirements
      - CentOS release 6.7 (Final)
    - ✓ Identical environment with the **Hortonworks Sandbox VM**

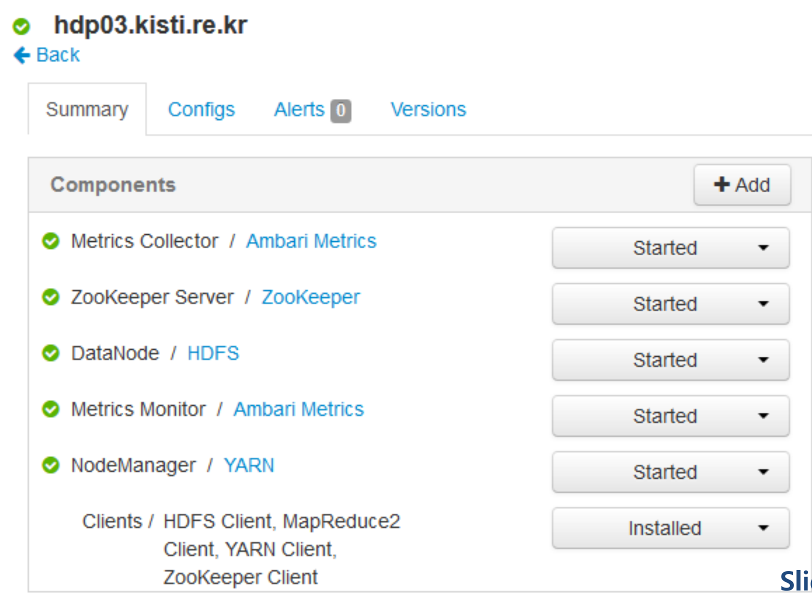


Experimental Setup

MOHA Testbed Configurations including *Masters* (YARN ResourceManager, HDFS NameNode) and *Slaves* (YARN NodeManager, HDFS DataNode) with additional Hadoop service components





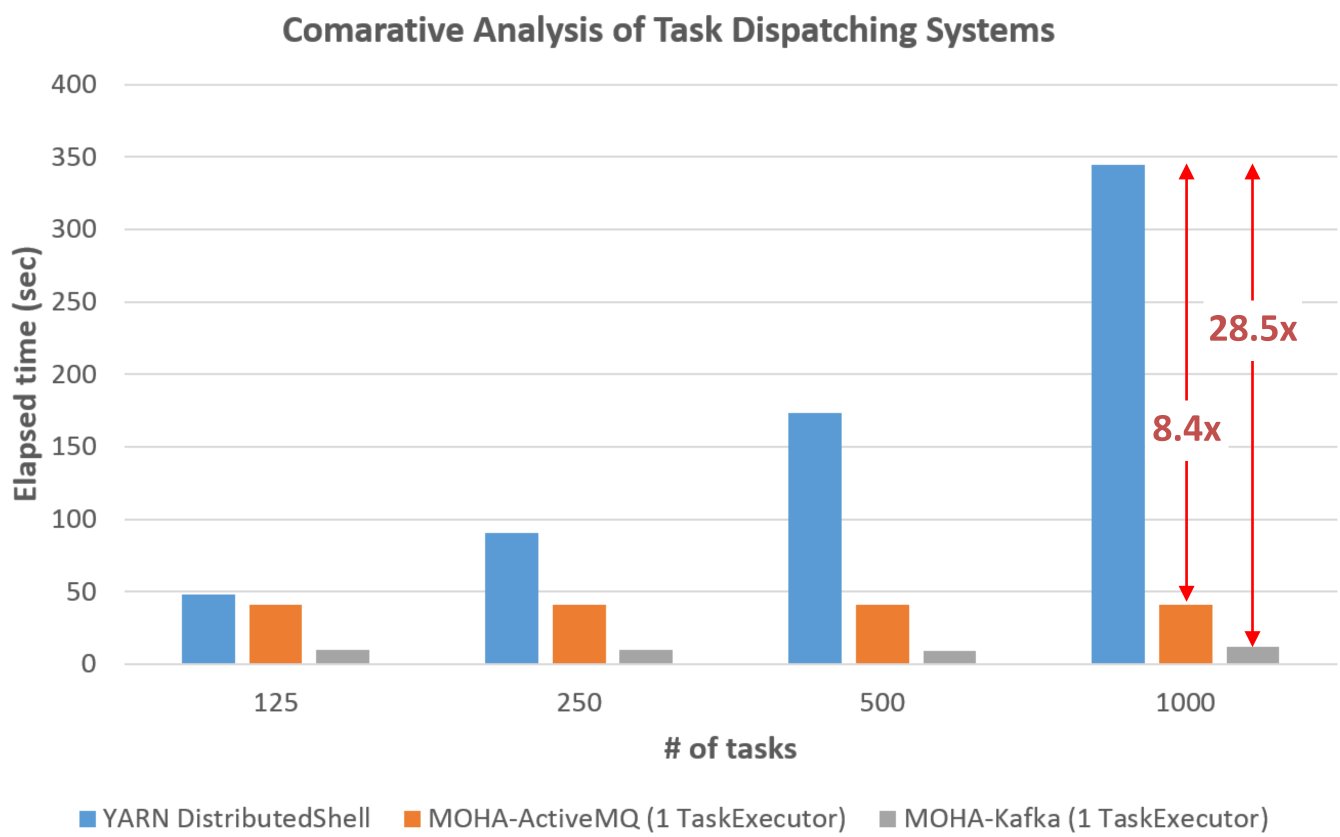


Experimental Setup

- **Comparison Models**
  - **YARN Distributed-Shell**
    - ✓ a simple YARN application that can execute shell commands (scripts) on distributed containers in a Hadoop cluster
  - **MOHA-ActiveMQ**
    - ✓ ActiveMQ running on a single node with New I/O (NIO) Transport
  - **MOHA-Kafka**
    - ✓ 3 Kafka Brokers with minimum fetch size (64 bytes)
- **Workload**
  - Microbenchmark
    - ✓ varying the # of “sleep 0” tasks
  - Performance Metrics
    - ✓ Elapsed time
    - ✓ Task processing rate (# of tasks/sec)

Experimental Results

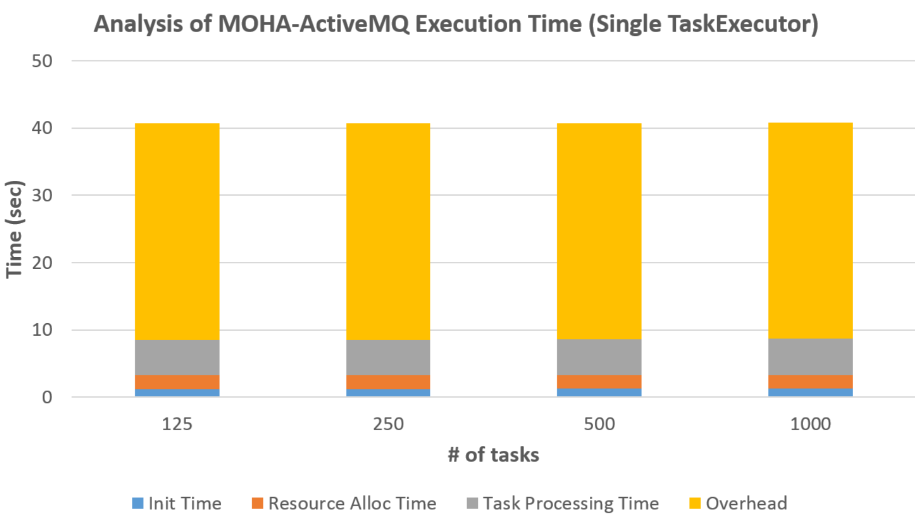
- **Performance Comparison (Total Elapsed Time)**
  - multiple resource (de)allocations in YARN Distributed-Shell
  - multi-level scheduling mechanisms enable MOHA frameworks to substantially reduce the cost of executing many tasks

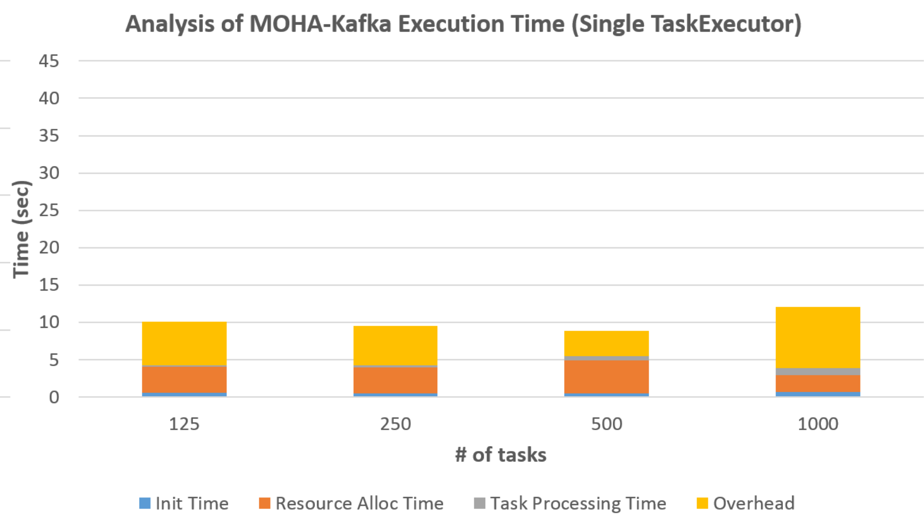


Experimental Results

Execution Time Breakdowns of MOHA Frameworks

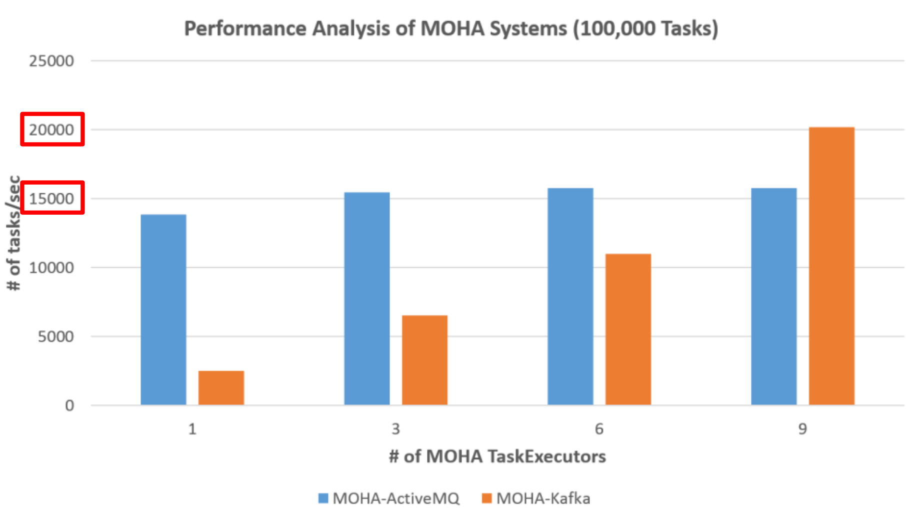
- resource allocation time of a single container can take a couple of seconds
- Overheads of MOHA-ActiveMQ are larger than MOHA-Kafka
  - ✓ due to higher memory usages in MOHA-ActiveMQ's TaskExecutor
    - relatively heavyweight ActiveMQ consumer libraries





Experimental Results

- **Task Dispatching Rate and Initialization Overhead**
  - MOHA-Kafka outperforms MOHA-ActiveMQ as the number of TaskExecutors increases (also Falkon's 15,000 tasks/sec)
    - ✓ have not fully utilized Kafka's task bundling functionality
  - Initialization Overhead
    - ✓ mostly queuing time



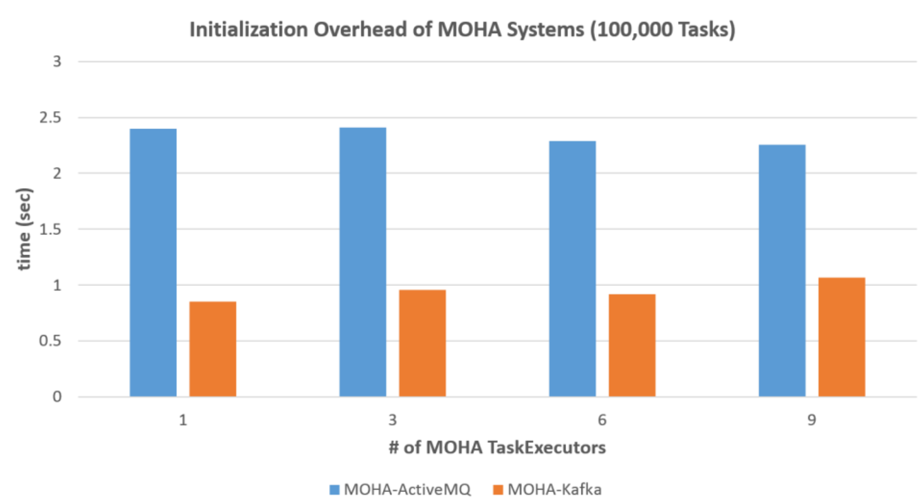


Table of Contents

Conclusion

- **Design and implementation of MOHA (Many-task computing On HAdoop) framework**
  - effectively *combine* MTC technologies with Hadoop
  - developed as *one of Hadoop YARN applications*
  - transparently *co-host* existing MTC applications with other Big Data processing frameworks in a single Hadoop cluster
- **MOHA prototype as a Proof-of-Concept**
  - can execute *shell* command based many tasks across distributed computing resources
  - substantially *reduce* the overall execution time of many-task processing with *minimal* amount of resources
    - ✓ compared to the existing YARN Distributed-Shell
  - efficiently *dispatch* a large number of tasks by exploiting *multi-level scheduling* and *streamlined task dispatching*

Future Work

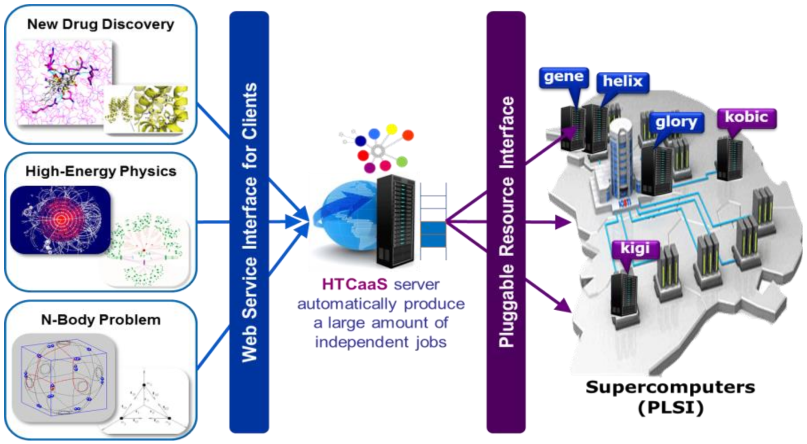
- **MOHA can bring many interesting research issues**
  - related to data grouping & declustering on HDFS, scalable job/metadata management, dynamic load balancing, etc.
  - considering applying a *new type of high-performance storage* system in HPC area such as *Lustre* on top of Hadoop
    - ✓ support relatively small data files from MTC applications by replacing conventional HDFS
  - ultimately contributing to **a new data processing framework for MTC applications in Hadoop 2.0 ecosystem**
- **Based on our years of experience to support "real scientific applications in MTC area", we plan to apply these applications on our new MOHA framework**

Thank you!

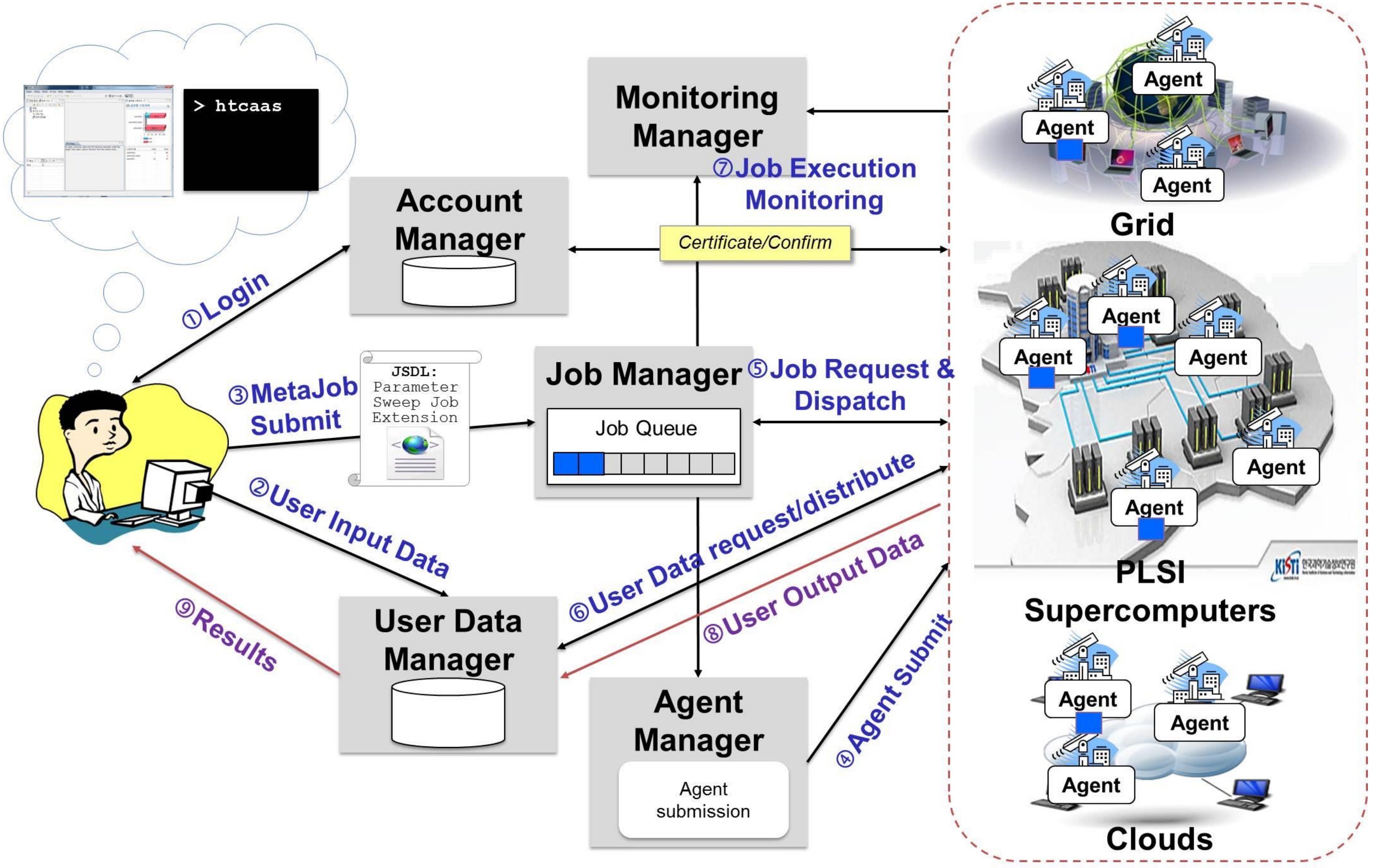
National Institute of
Supercomputing and Networking
2016

Related Work: HTCaaS

- **HTCaaS: a Multi-level Scheduling System**
  - **High-Throughput Computing as a Service**
    - ✓ *Meta-Job* based automatic job split & submission
      - e.g., parameter sweeps or N-body calculations
    - ✓ Agent-based *multi-level scheduling*
    - ✓ *Pluggable interface* to heterogeneous computing resources
    - ✓ Leveraging *local disks* of each compute node
    - ✓ Supporting many *client interfaces*
  - HTCaaS is currently running as a **pilot service** on top of PLSI
    - ✓ supporting a number of scientific applications from **pharmaceutical domain** and **high-energy physics**



Related Work: HTCaaS



Related Work: HTCaaS

- **Falkon MTC Task Dispatcher**
  - achieve **15,000 tasks/sec** dispatching performance
    - ✓ Ioan Raicu et. al, "Middleware support for many-task computing", Cluster Computing, Volume 13 Issue 3, September 2010
    - ✓ **One billion tasks (sleep 0) on 128 processors** in a Linux cluster
      - **19.2 hours** to complete
      - distributed version of the Falkon dispatcher using **four instances** on an 8-core server using **bundling of 100**

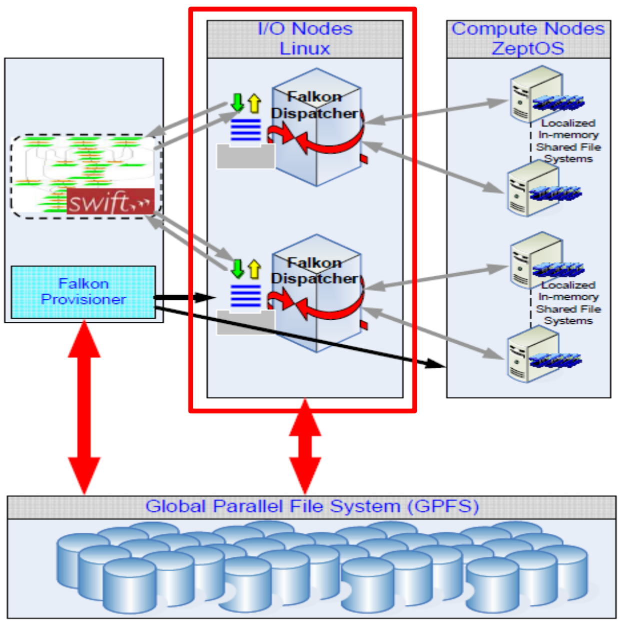


Figure 4: 3-Tier Architecture Overview

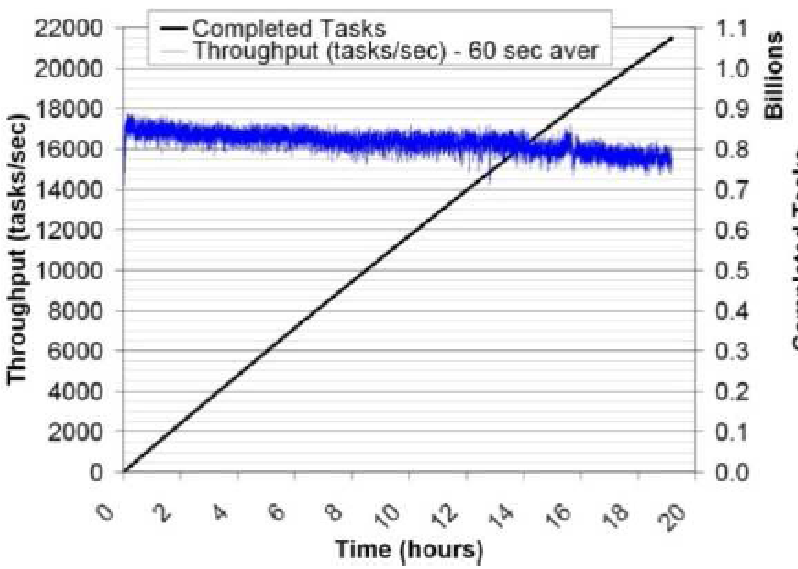


Figure 9: Endurance test with 1B tasks on 128 CPUs in ANL/UC cluster