Kepler + Hadoop : A General Architecture Facilitating Data-Intensive Applications in Scientific Workflow Systems

Jianwu Wang, Daniel Crawl, Ilkay Altintas

San Diego Supercomputer Center,
University of California, San Diego

Introduction

- **Goals**
  - Easily compose MapReduce applications in workflow
  - Easily connect MapReduce applications with other programs via workflow
  - Efficiently execute them in the Hadoop environment
- **Advantages : combination of characteristics**
  - Scientific workflow: GUI, component reuse and sharing, task composition
  - MapReduce: parallelism, scalability, automatic data partitioning, load balancing, fault tolerance

Background – Kepler

- **Actor-oriented Modeling**
  - All these actors inherit the same interfaces, such as *prefire()*, *fire()* and *postfire()*
- **Model of Computation**
  - Synchronous Data Flow (SDF) director: actors execute sequentially
  - Process Network (PN) director: each actor has its own execution thread and execute in parallel
- **Others**
  - Actor customization
  - Actor reuse and sharing locally or publicly through the Kepler actor repository

Background – MapReduce and Hadoop

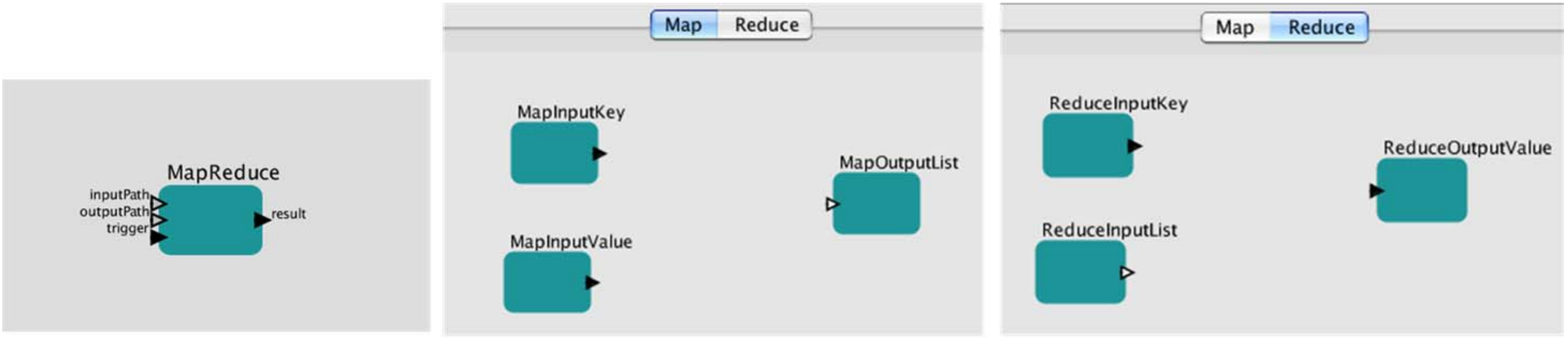
- **MapReduce**
  - A parallel and scalable programming model for data-intensive computing
  - Input data is automatically partitioned onto multiple nodes and programs are distributed and executed in parallel on the partitioned data blocks.

map (k1,v1) → list(k2,v2)

reduce (k2,list(v2)) → list(v2)

- **Hadoop**
  - Open source implementation of MapReduce.
  - Consists of MapReduce runtime system and a distributed file system, called HDFS.
  - One Hadoop node, called *master*, dispatches tasks and manages the executions of the other Hadoop nodes, i.e., *slaves*

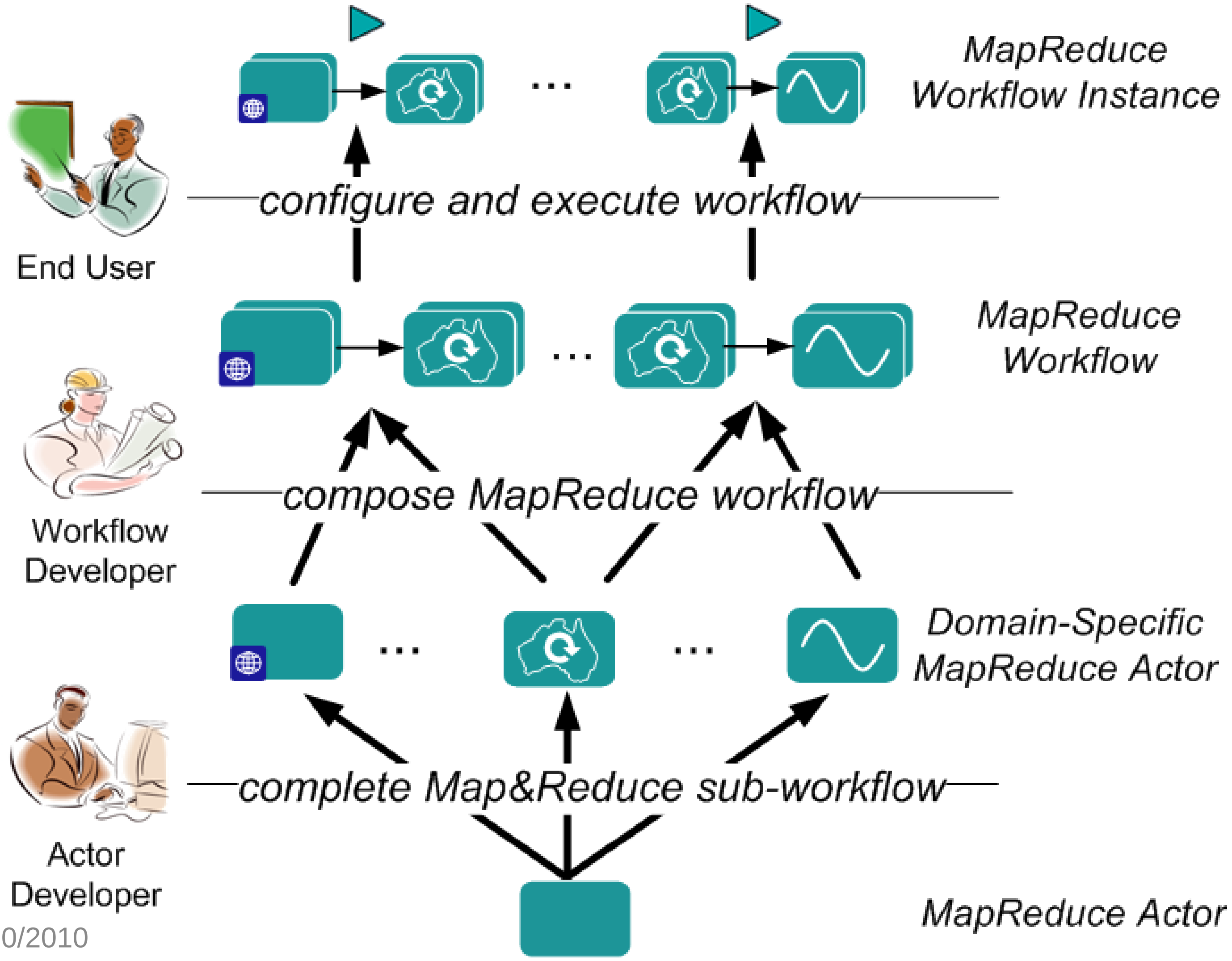
MapReduce Actor in Kepler



(a) (b) (c)

(a) MapReduce actor. (b) Map sub-workflow in MapReduce actor. (c) Reduce sub-workflow in MapReduce actor.

MapReduce Actor Usage



MapReduce Workflow Instance

End User

configure and execute workflow

MapReduce Workflow

Workflow Developer

compose MapReduce workflow

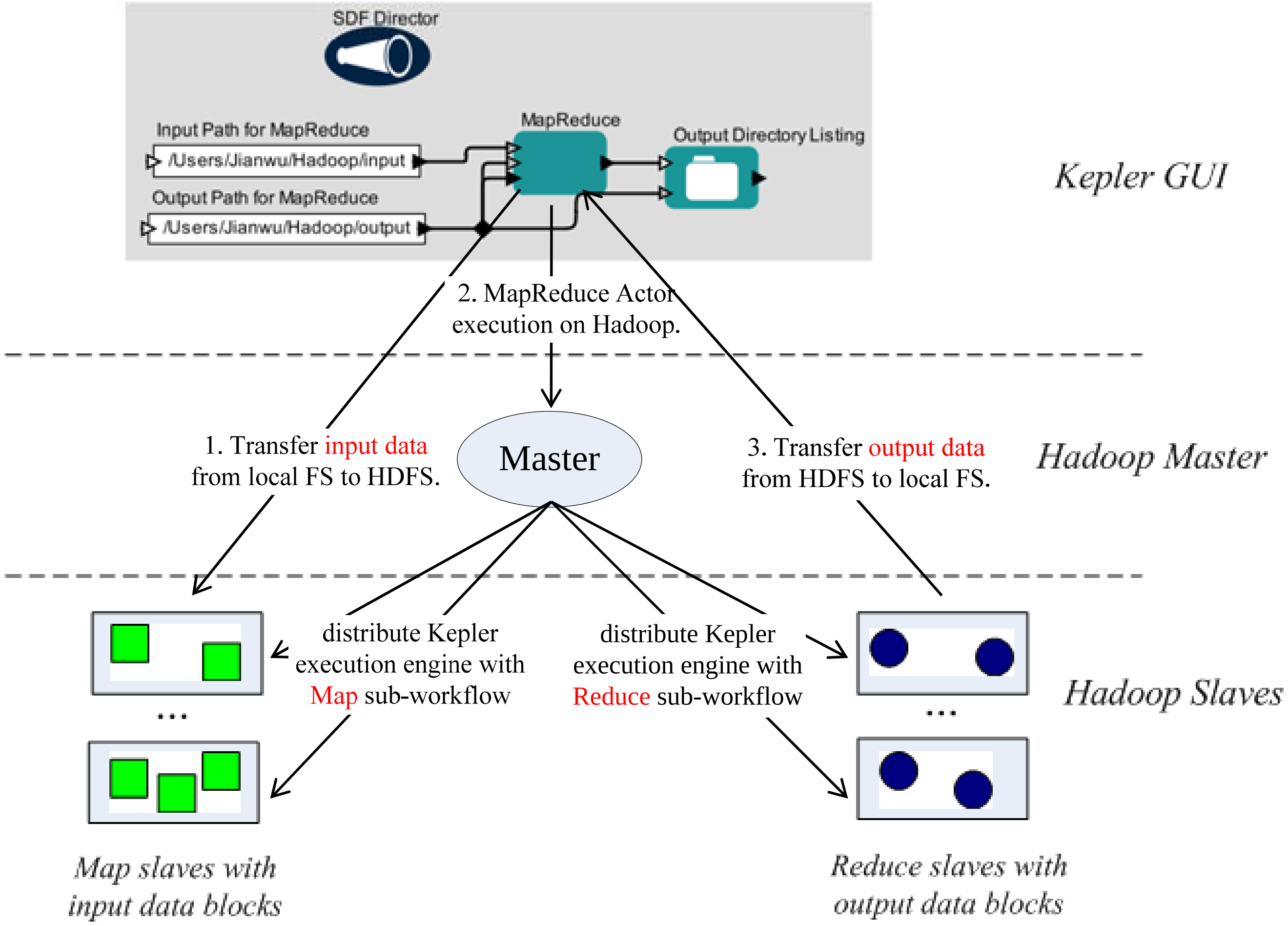
Domain-Specific MapReduce Actor

Actor Developer

complete Map&Reduce sub-workflow

MapReduce Actor

MapReduce Actor Execution in Hadoop



Kepler GUI

2. MapReduce Actor execution on Hadoop.

1. Transfer input data from local FS to HDFS.

3. Transfer output data from HDFS to local FS.

Hadoop Master

distribute Kepler execution engine with Map sub-workflow

distribute Kepler execution engine with Reduce sub-workflow

Hadoop Slaves

Map slaves with input data blocks

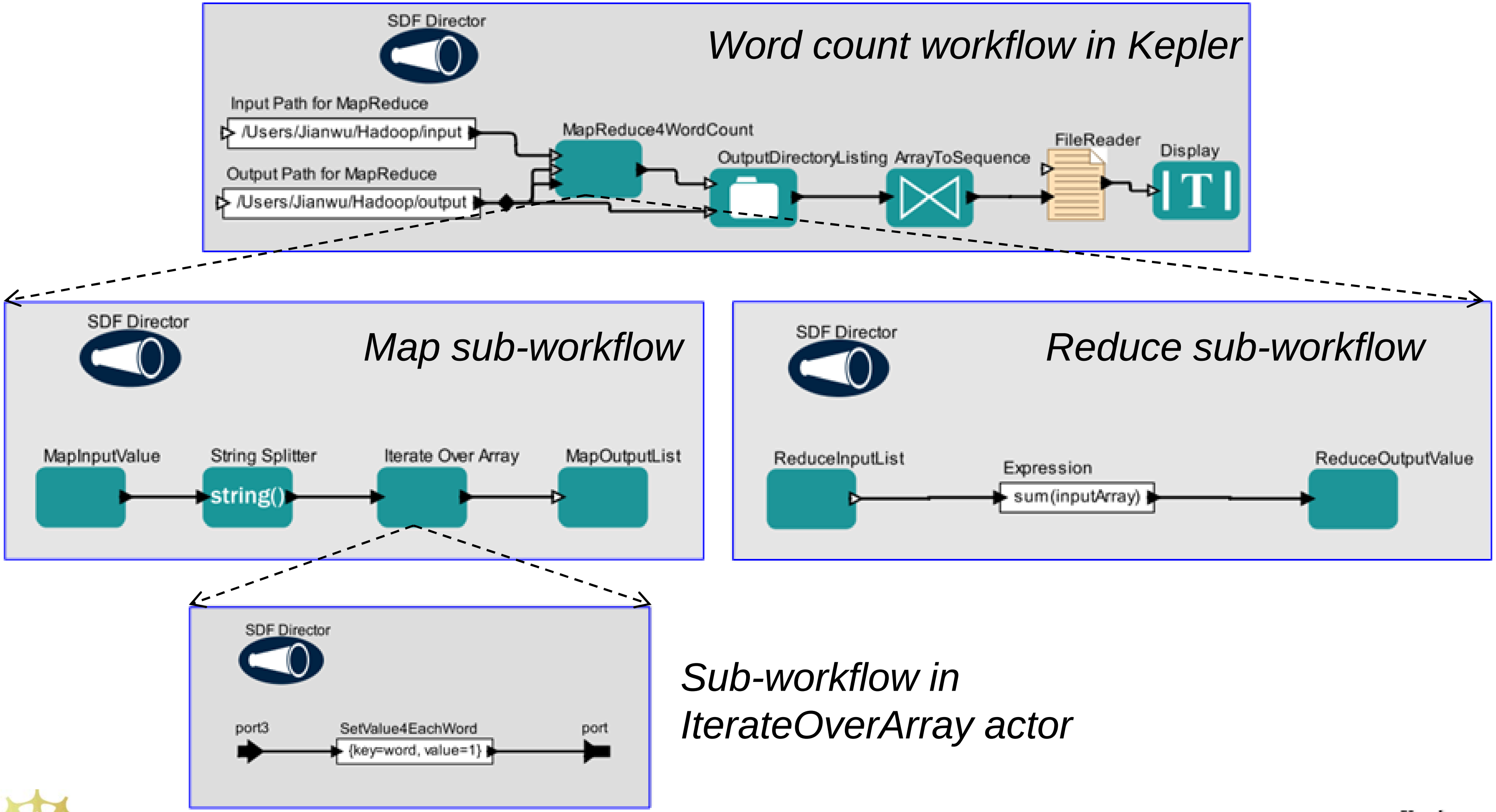
Reduce slaves with output data blocks

Execution Semantics in MapReduce Actor

```
map (k1, v1) {
   initialize Kepler execution engine for Map sub-workflow
   send k1 to Kepler engine via MapInputKey actor
   send v1 to Kepler engine via MapInputValue actor
   execute Map sub-workflow
   get list(k2, v2) from Kepler engine via MapOutputList actor
   emit  list(k2, v2)
}

reduce (k2, list(v2)) {
   …
}
```

Using MapReduce Actor for Word Count



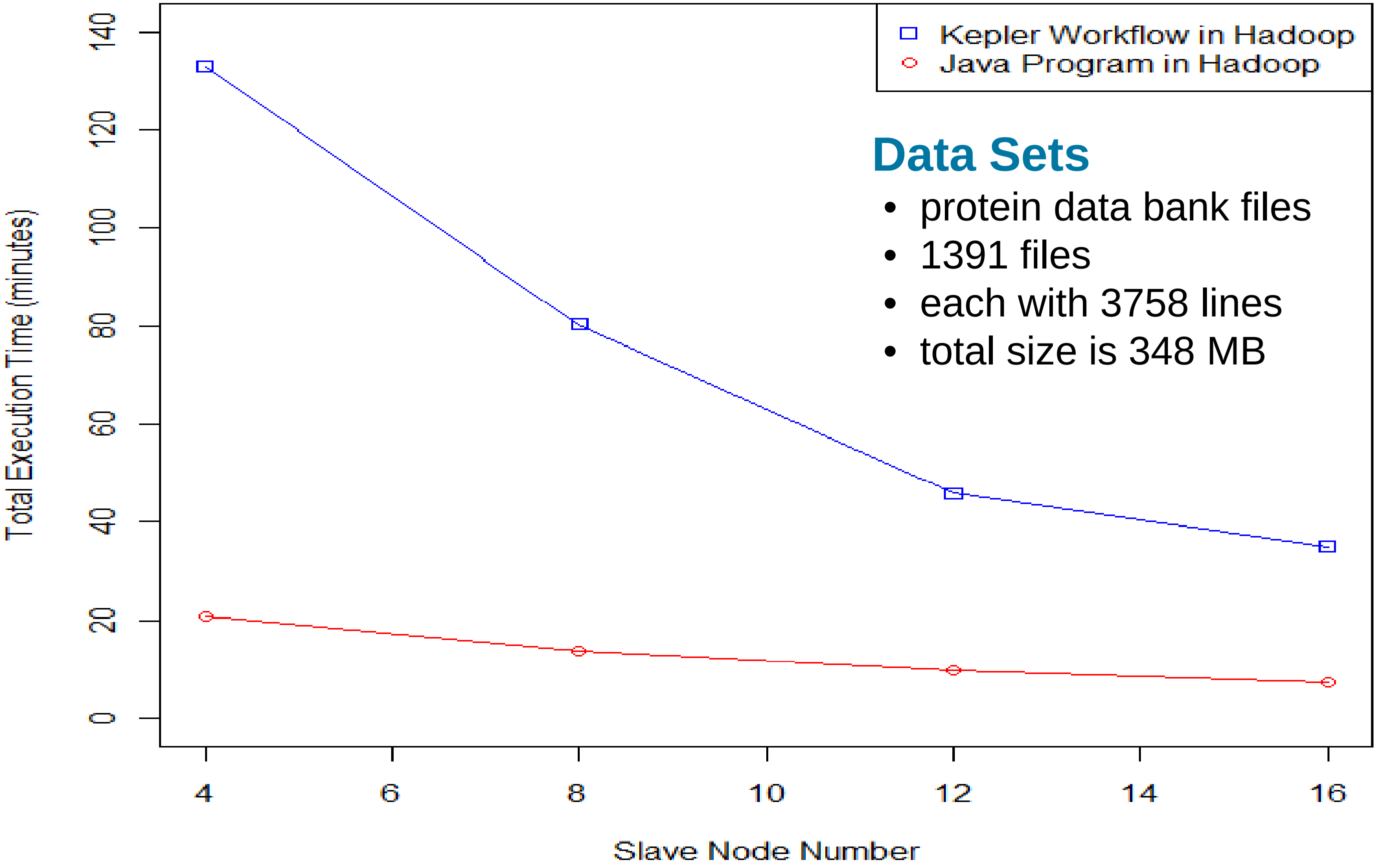
Word count workflow in Kepler

Map sub-workflow

Reduce sub-workflow

Sub-workflow in IterateOverArray actor

Experiment 1: Execution on Different Cluster Nodes



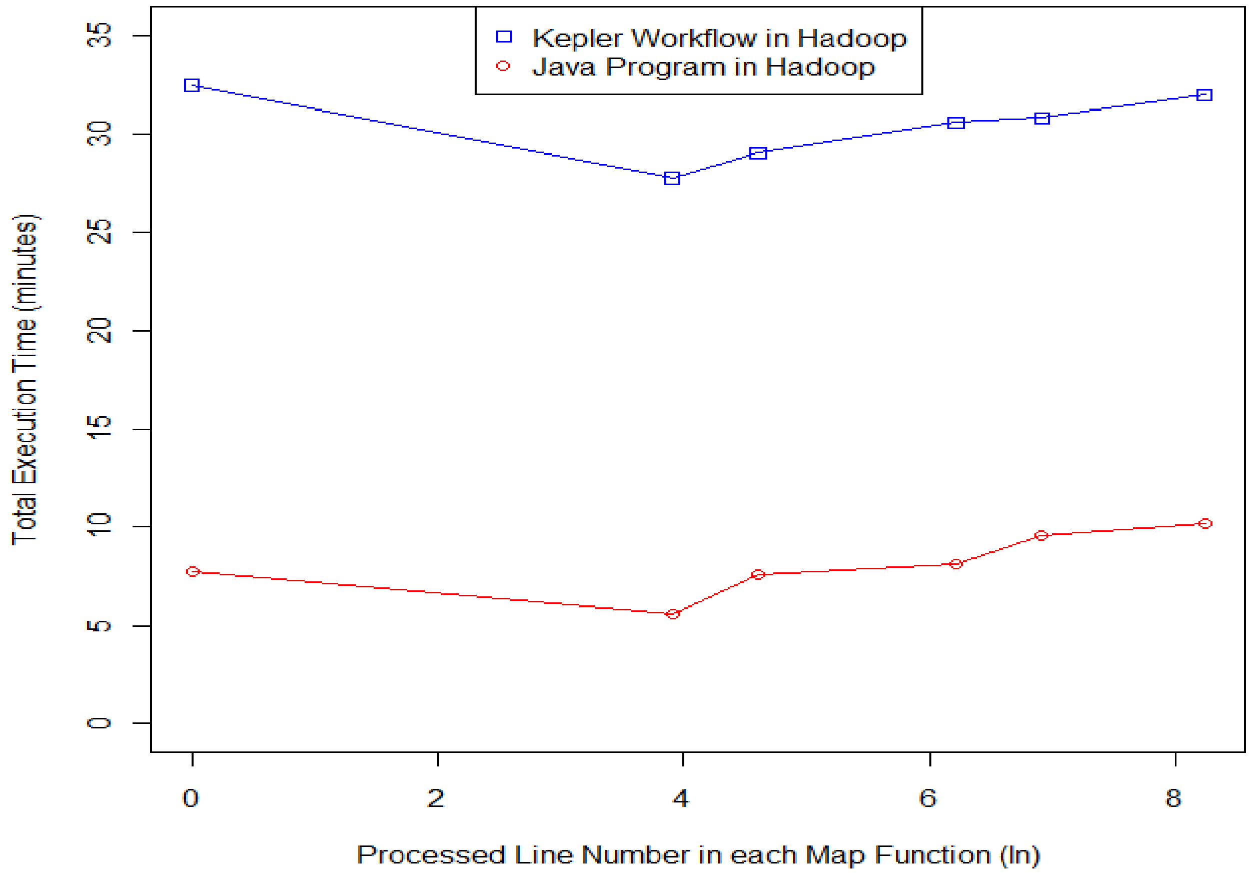
Data Sets

- protein data bank files
- 1391 files
- each with 3758 lines
- total size is 348 MB

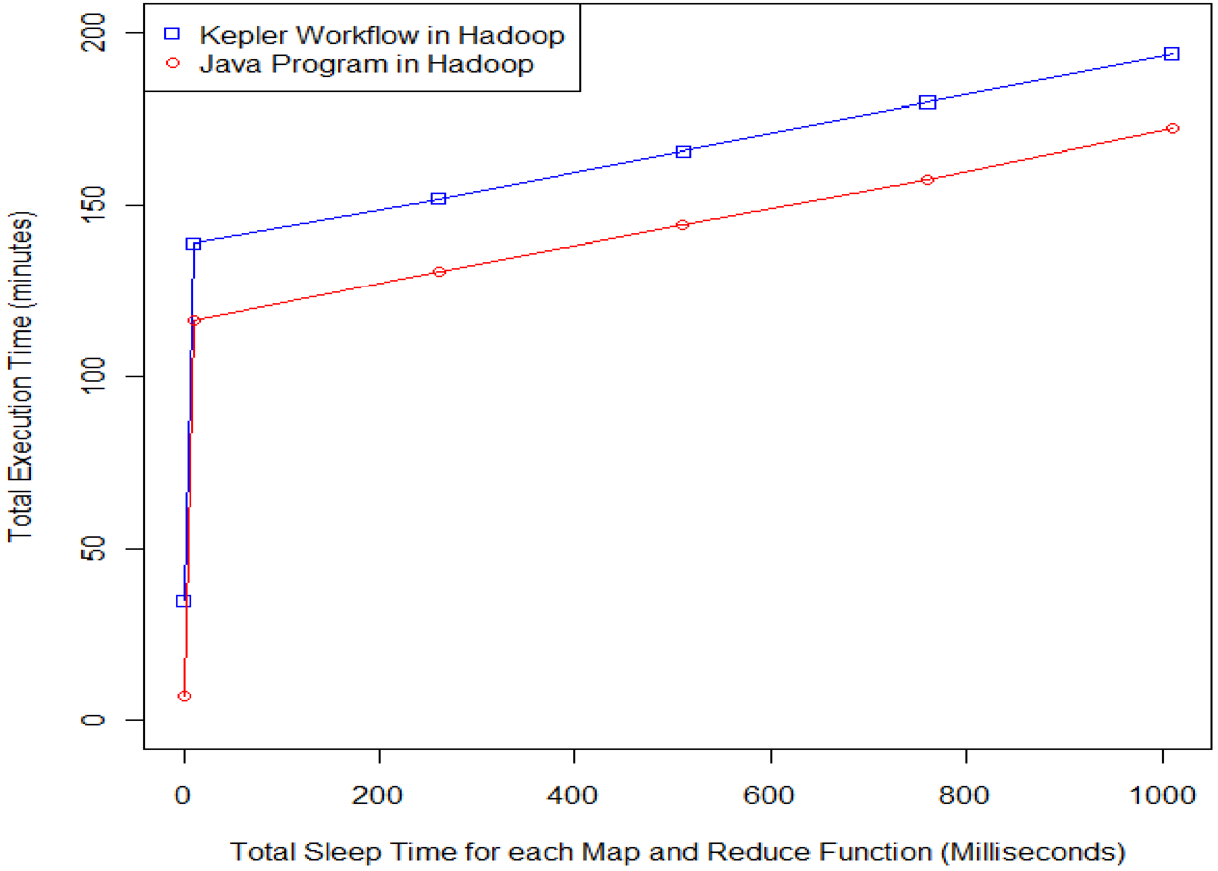
Overhead Analysis

- **Overhead Reason**
  - Kepler engine initialization
  - Map/Reduce sub-workflow parsing
- **Overhead in this Case**
  - The overhead for each Map/Reduce sub-workflow instance takes about 10 milliseconds
  - The execution time of each Map/Reduce instance in Java is much shorter (0.3 ms for Map; 0.03 ms for Reduce)
  - The whole execution number for the Map/Reduce function invocation is about 20 million

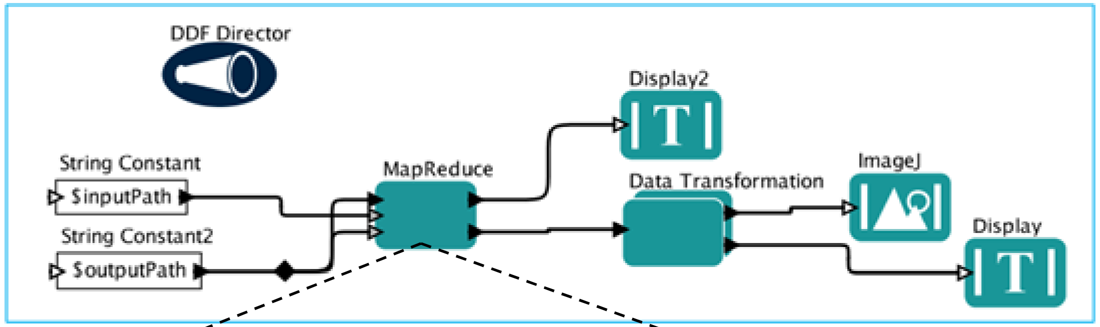
Experiment 2: Execution with Increased Data Size in Map



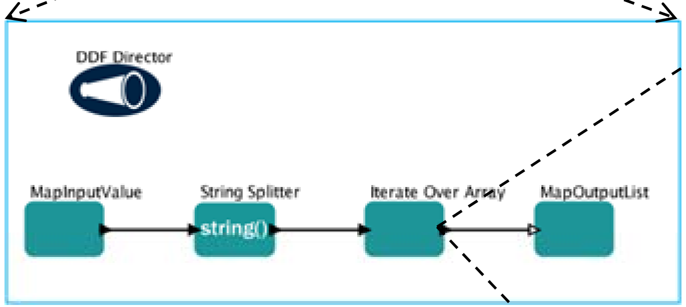
Experiment 3: Execution with Increased Execution Time in MapReduce



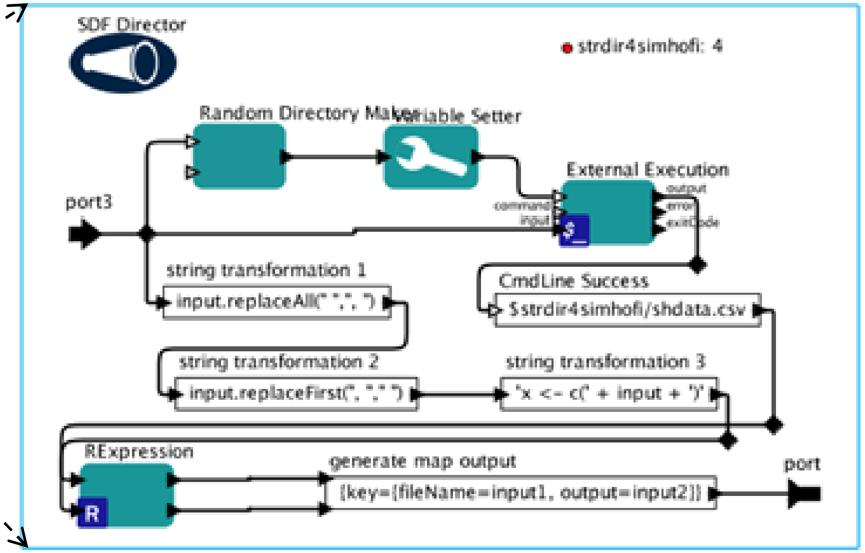
Using MapReduce Actor for A Parameter Sweep Application



A Parameter Sweep workflow in Kepler



Map sub-workflow



Sub-workflow in IterateOverArray actor

Conclusion and Future Work

- **Kepler + Hadoop : A General Architecture Facilitating Data-Intensive Applications**
  - – Easily create MapReduce sub-workflows, connect them with other tasks using Kepler
  - – Execute them efficiently and transparently via the Hadoop infrastructure
- **Future Work**
  - – Refactor to enhance its capability, performance, and robustness
  - – Apply to concrete domain-specific scientific problems

- **Acknowledgements**
  - The rest of the Kepler team
  - NSF SDCI Award for Kepler/CORE, NSF CEO:P Award for REAP, DOE SciDac Award for SDM Center, and UCGRID Project

- **For More Information:**
  - Distributed Execution Interest Group of Kepler: https://dev.kepler-project.org/developers/interest-groups/distributed
  - Contact: jianwu@sdsc.edu