Distributed Execution Architectures in Kepler

Jianwu Wang[1], Daniel Crawl[1],
Ilkay Altintas[1], Chad Berkley[2], Matthew B. Jones[2]

[1] *San Diego Supercomputer Center, UCSD*
[2] *National Center for Ecological Analysis and Synthesis, UCSB*

Outline

- Distributed Execution Architectures in Kepler
- Master-Slave Distributed Execution Architecture in Kepler
- MapReduce Distributed Execution Architecture in Kepler
- Comparison between the Above Two Architectures

Part I

- **Distributed Execution Architectures in Kepler**
- Master-Slave Distributed Execution Architecture in Kepler
- MapReduce Distributed Execution Architecture in Kepler
- Comparison between Master-Slave and MapReduce Distributed Execution Architectures

Distributed Execution Requirements in Kepler

- Various requirements on distributed execution in different environments, examples:
  - Ad-hoc network resources
  - Web service resources
  - Cluster resources
  - Grid resources
  - Cloud resources
  - …

Kepler

Distributed Execution Supports in Kepler

- Kepler integrated frameworks and libraries to support the requirements
  - Remote method invocation (RMI) for ad-hoc network resources
  - Axis Web service libraries for Web Service invocation
  - Ssh session libraries (JSch) for remote execution and job submission on clusters
  - Globus libraries for Grid computing

Kepler

Three Distributed Execution Levels in Kepler

- **Workflow level:** the whole workflow can be executed in distributed environments
  - Example: Web service for Kepler workflow execution
- **Actor level:** distributed computing and data resources can be utilized in an actor
  - Example: Web service actor in Kepler
- **Sub-workflow level:** sub-workflows can be executed in distributed environments
  - Example: Master-Slave and MapReduce Distributed Execution

Advantages of Using Workflow System for Distributed Execution

- **Reuse existing workflows**
  - Easily transform workflow from centralized execution to distributed execution
- **Transparent implementation**
  - Hide diverse distributed techniques from users, such as different job schedulers
  - Just drag-and-drop, no coding is needed
- **Optimal execution**
  - (Semi-)automatically get the best execution plan
- **Provenance support**
- **Fault tolerance**

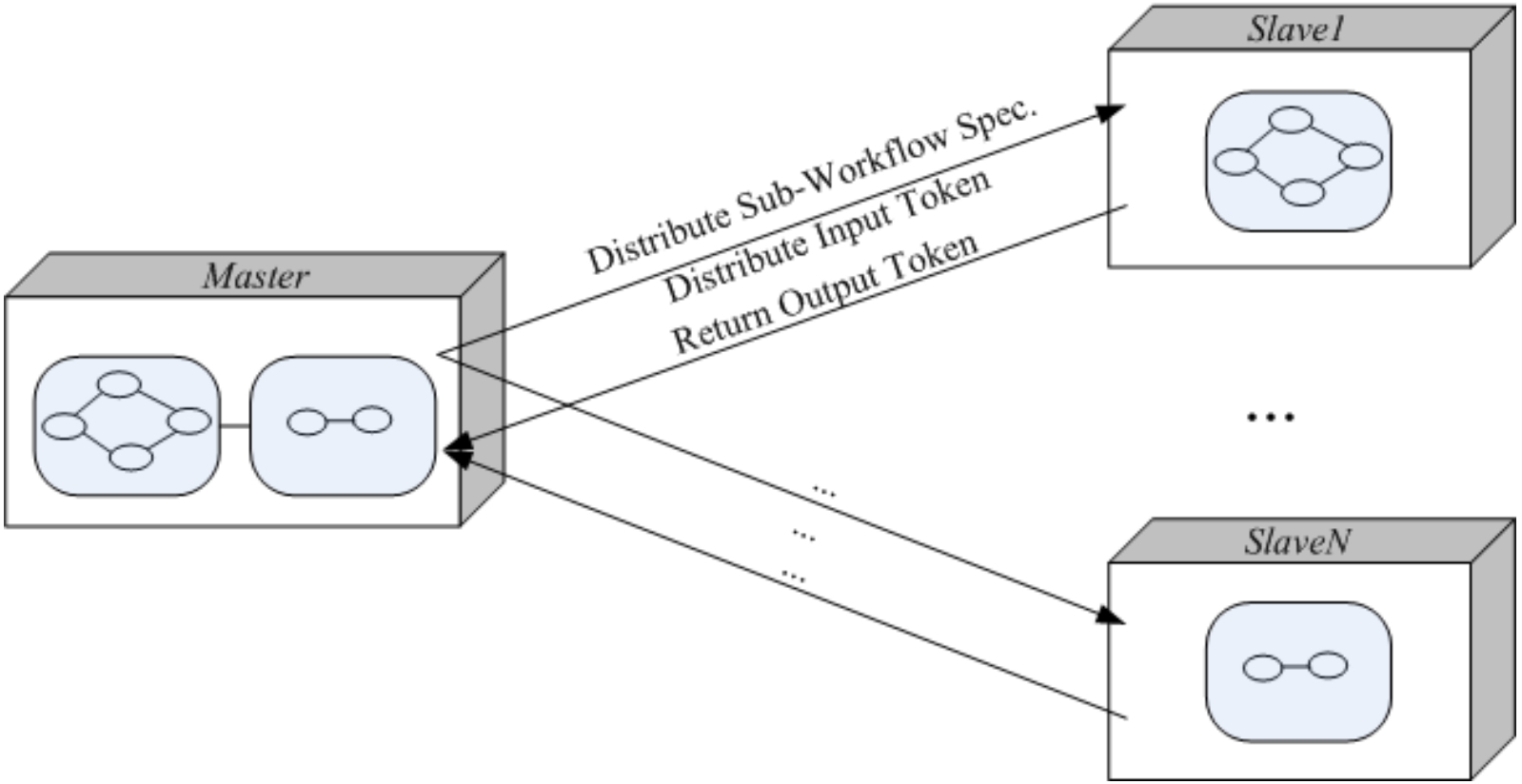
Kepler

Part II

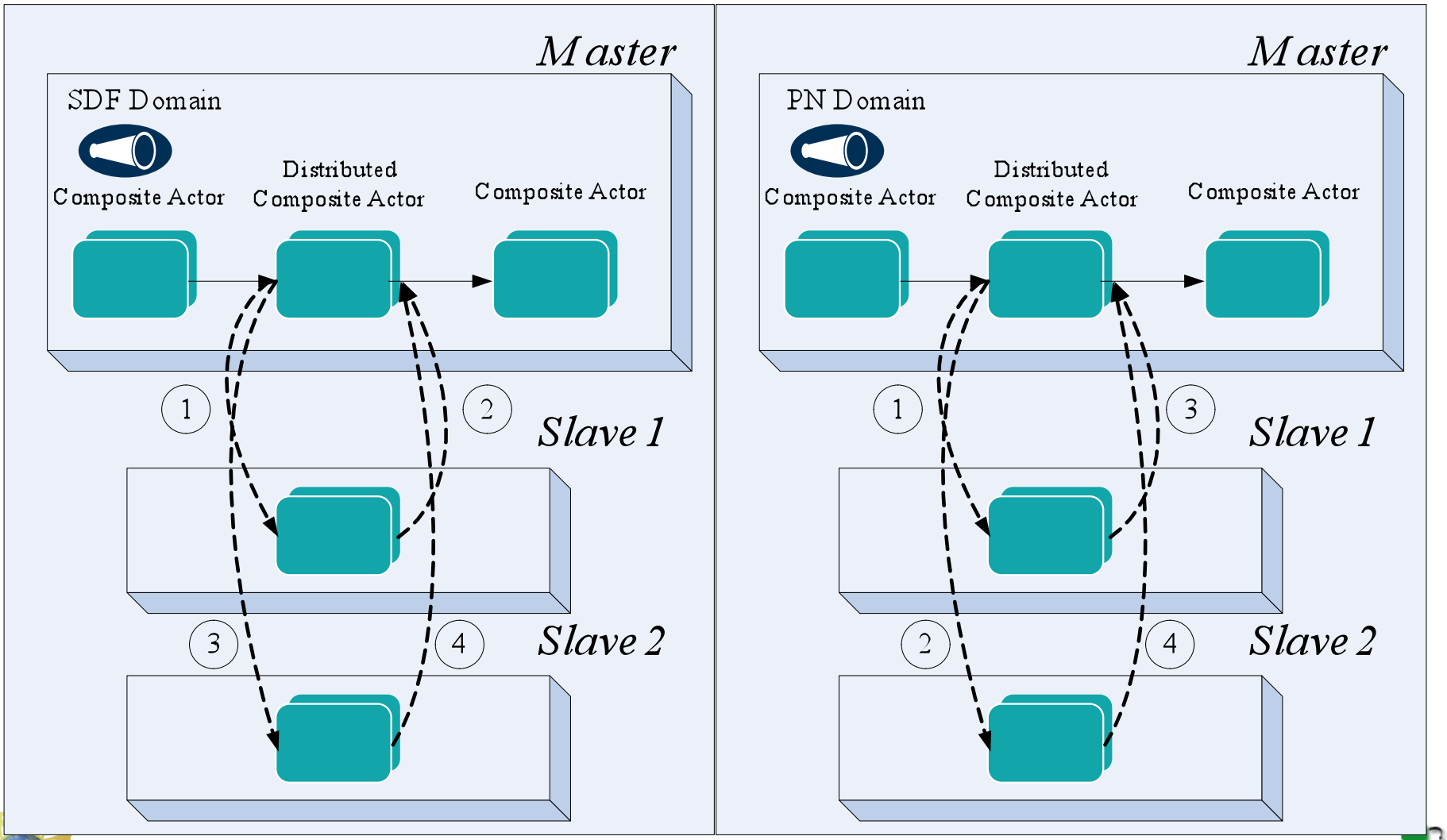
- Distributed Execution Architectures in Kepler
- **Master-Slave Distributed Execution Architecture in Kepler**
- MapReduce Distributed Execution Architecture in Kepler
- Comparison between Master-Slave and MapReduce Distributed Execution Architectures

Distributed Composite Actor

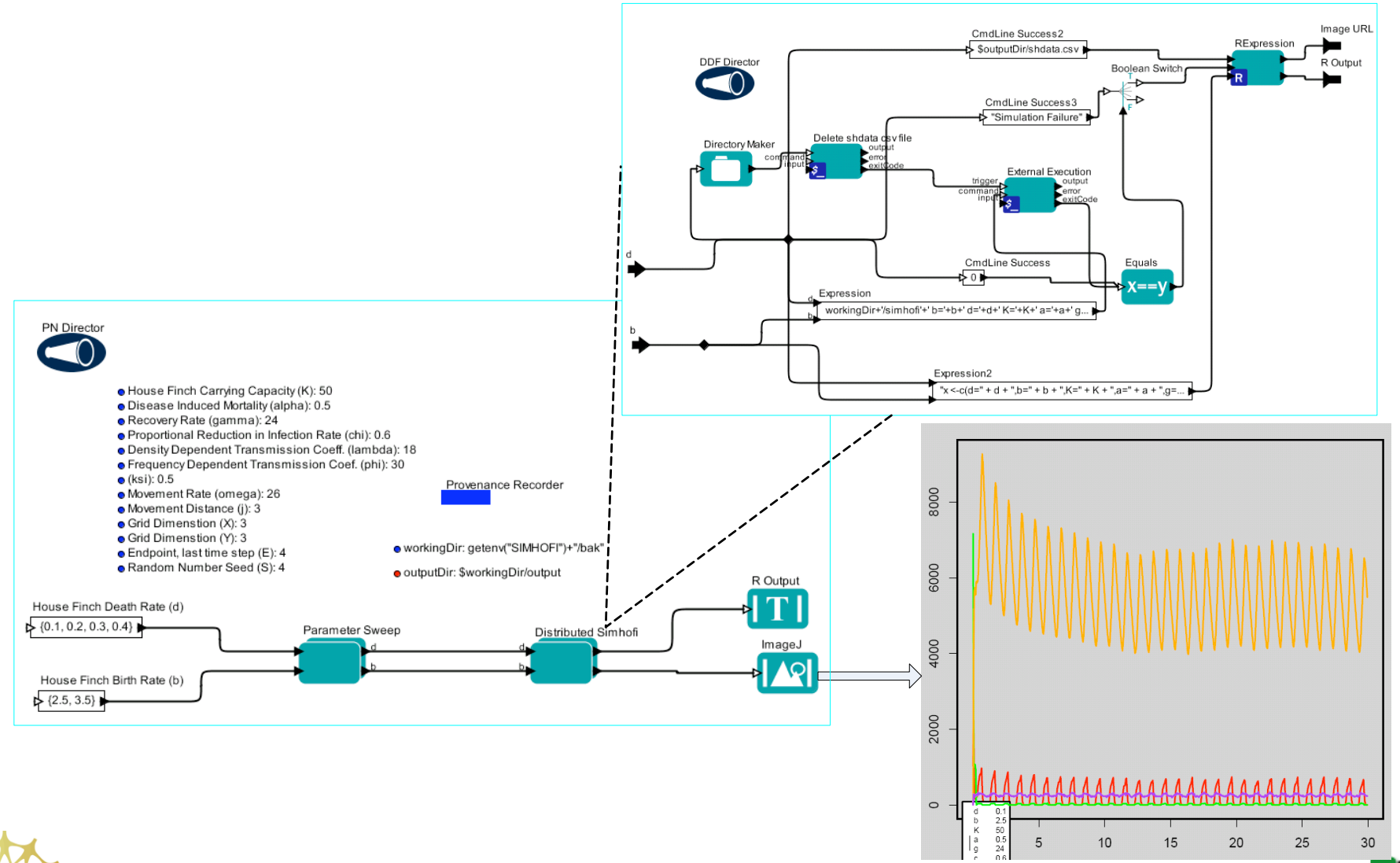
- As the role of Master, each token received by Distributed Composite Actor is distributed to a Slave node, executed, and the results returned.



Distributed Composite Actor Behaviors with Different Computation Models

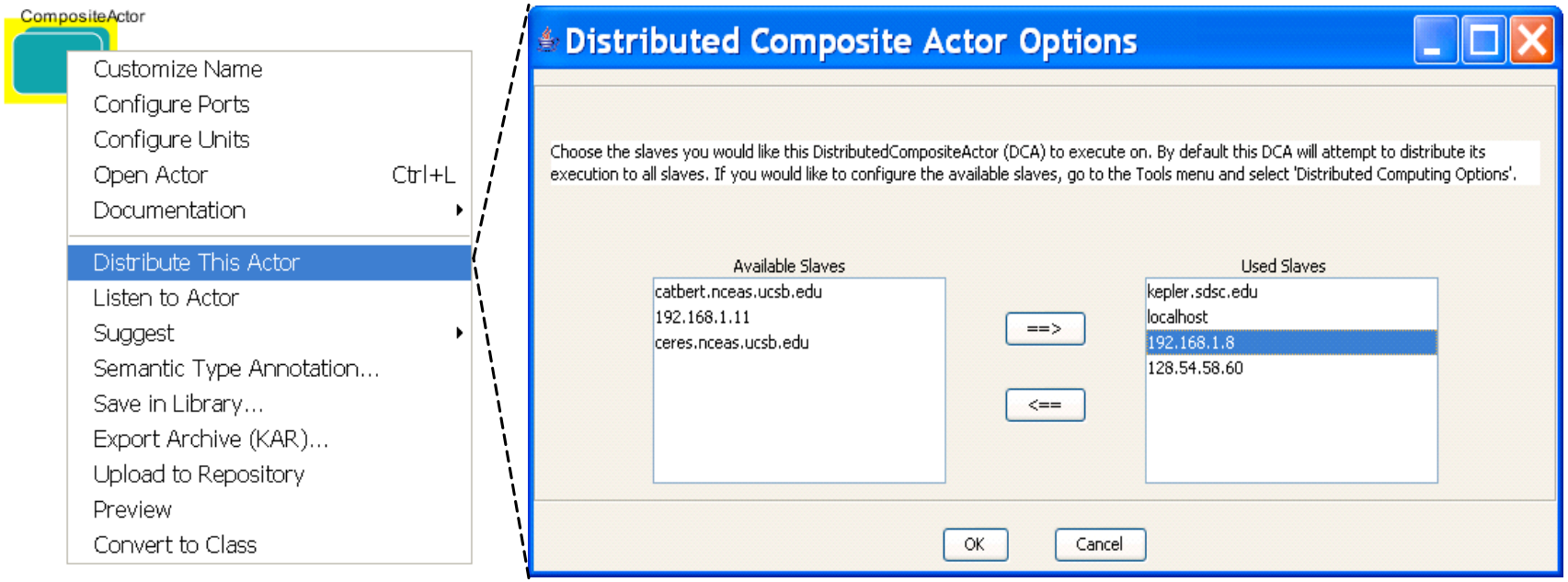


Demo Workflow

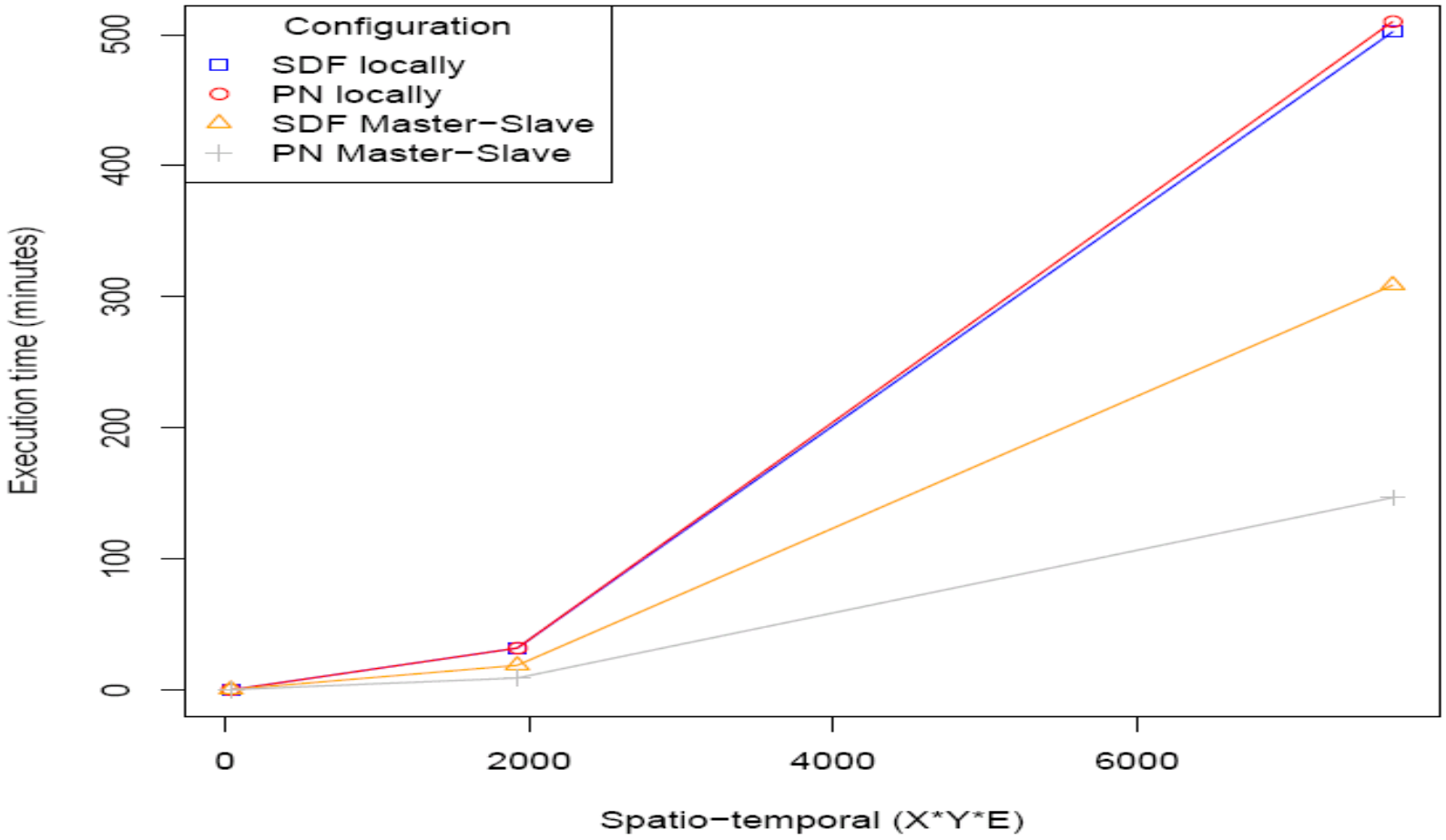


Usability

- Users use the DistributedCompositeActor just like the common composite actor
- Interaction for execution environment transition



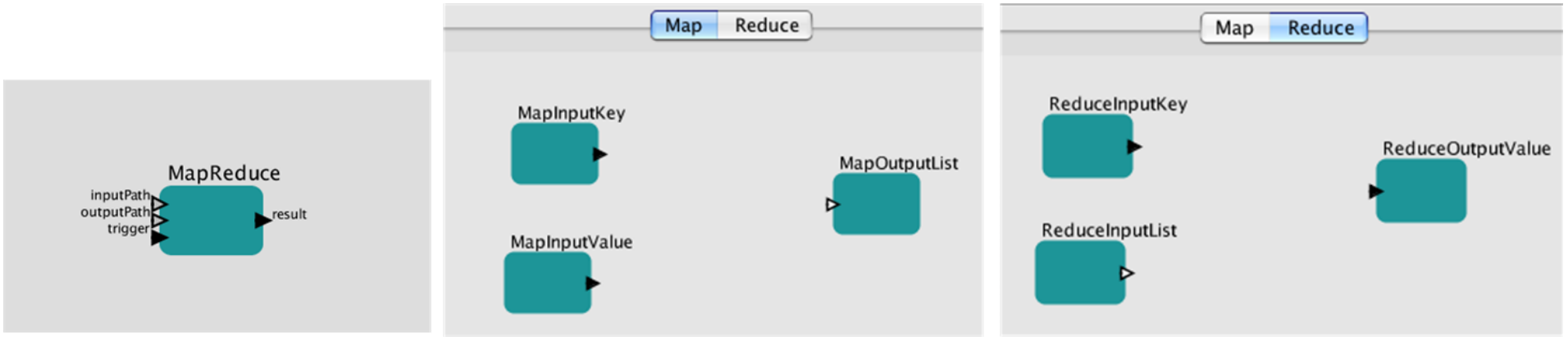
Performance Experiment



Part III

- Distributed Execution Architectures in Kepler
- Master-Slave Distributed Execution Architecture in Kepler
- **MapReduce Distributed Execution Architecture in Kepler**
- Comparison between Master-Slave and MapReduce Distributed Execution Architectures

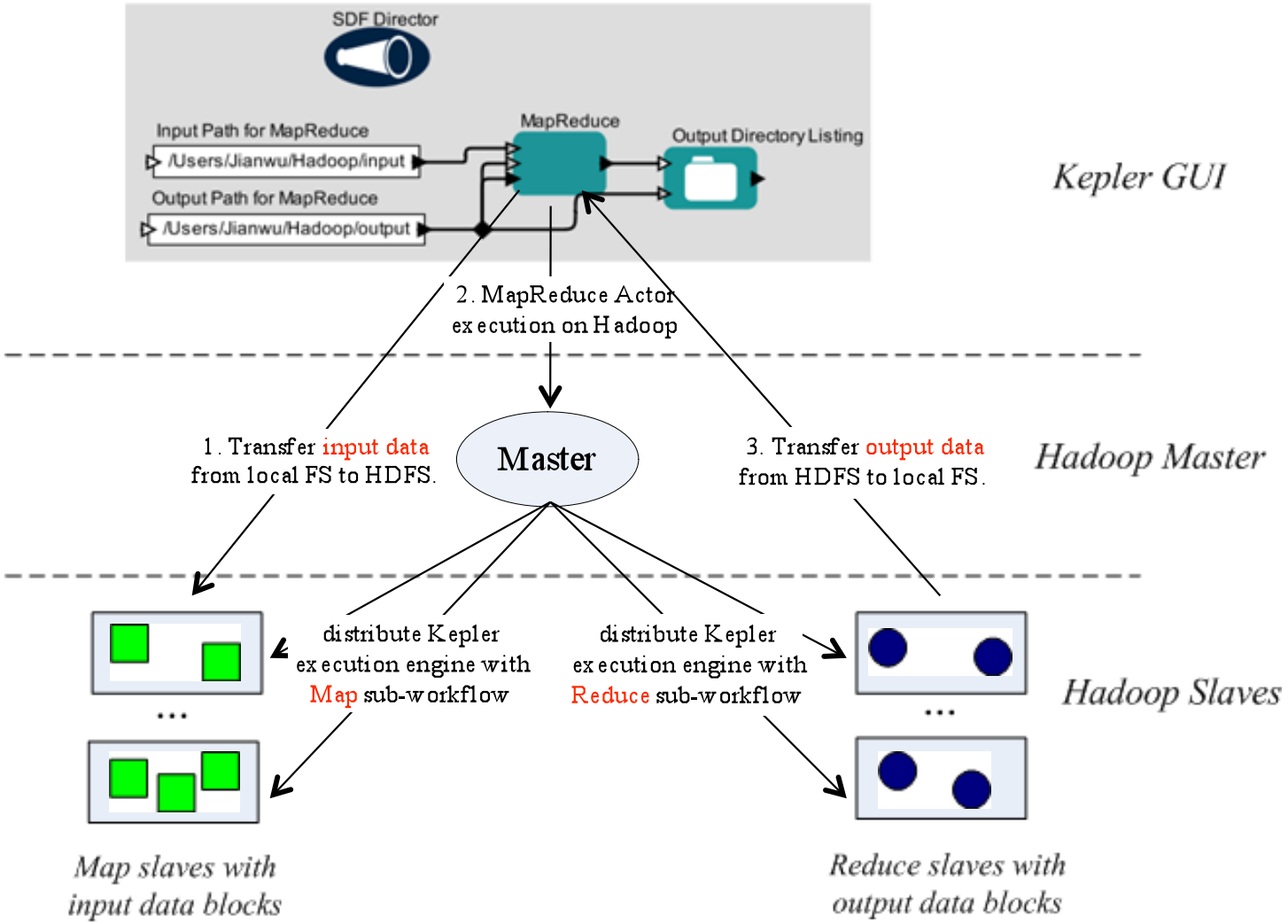
MapReduce Actor in Kepler



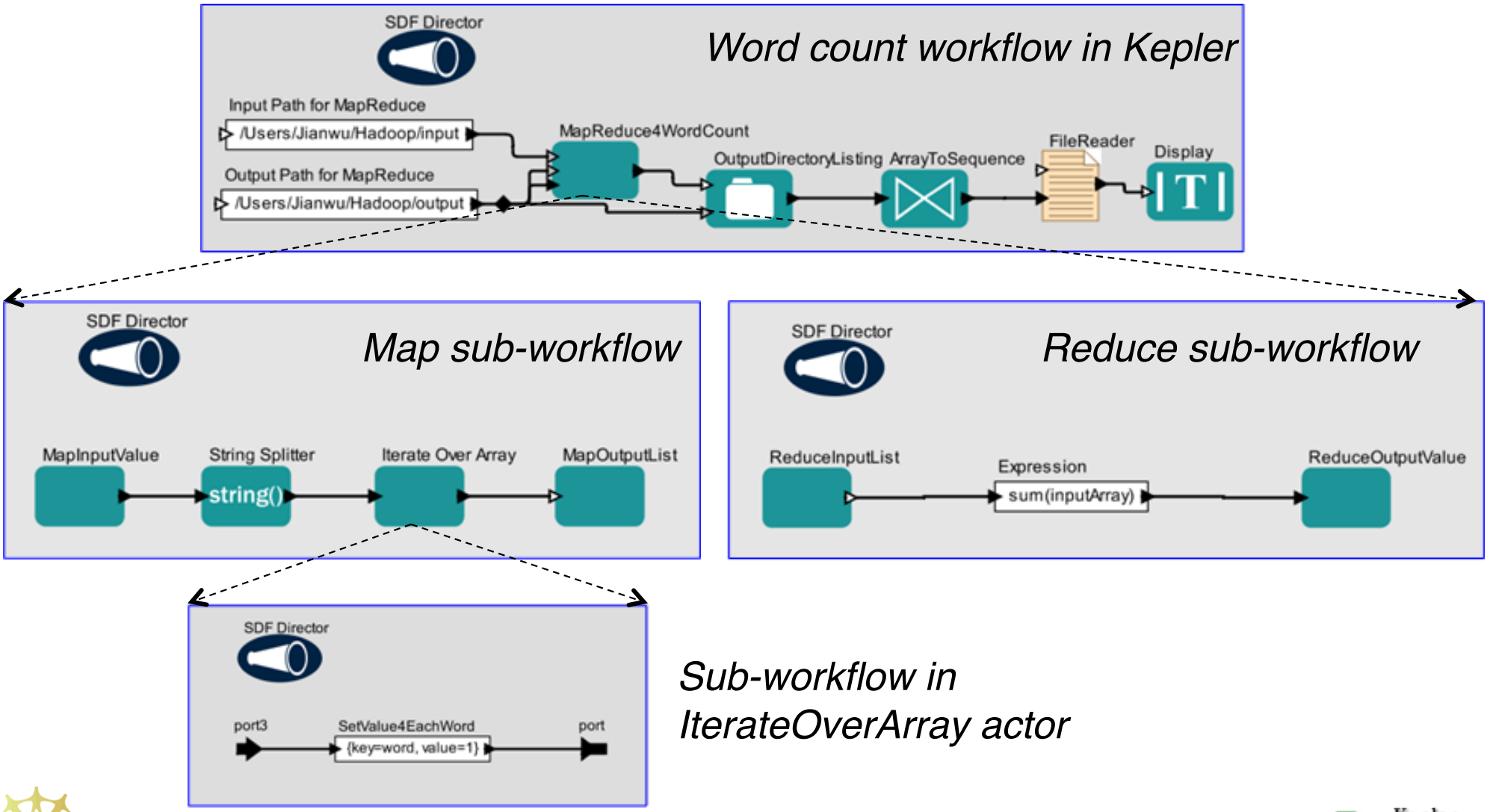
(a) (b) (c)

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

MapReduce Actor Execution in Hadoop

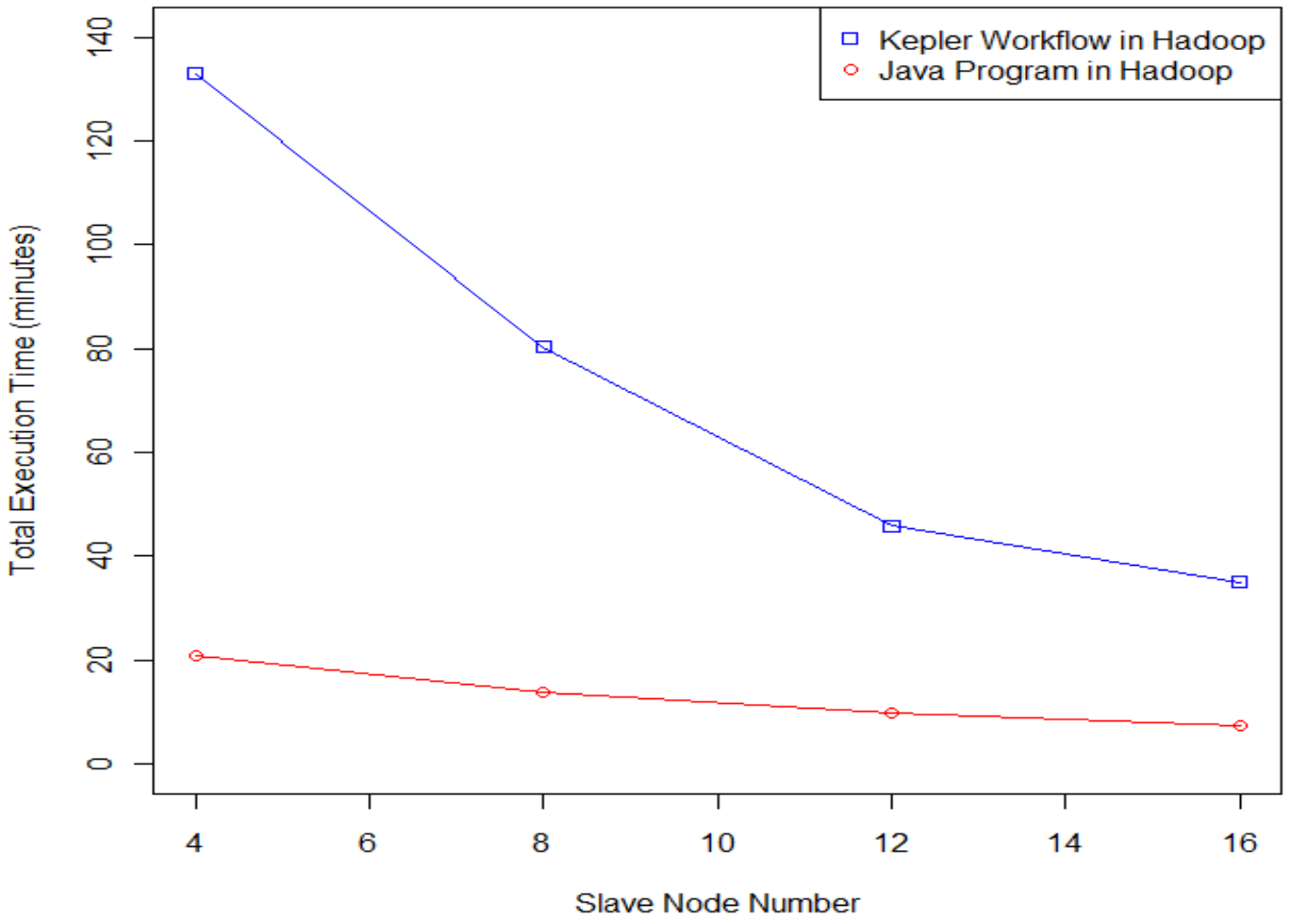


Using MapReduce Actor for Word Count



Sub-workflow in IterateOverArray actor

Performance Experiment for Word Count



Part IV

- Distributed Execution Architectures in Kepler
- Master-Slave Distributed Execution Architecture in Kepler
- MapReduce Distributed Execution Architecture in Kepler
- **Comparison between Master-Slave and MapReduce Distributed Execution Architectures**

Kepler

Commonalities

- Both have distributed data + distributed programs
- Both have master and slaves
- Both have execution engines on slaves

Main Differences

- **MapReduce**
  - Usually, all input data needs to be staged in beforehand and outputs is only accessible when the whole execution is finished
  - More suitable for large data sets, and has good scalability on clusters with numerous nodes
- **Master-Slave**
  - Inputs can be provided dynamically and get its result gradually once it is generated
  - More suitable for dynamic data distribution cases

Kepler

Thanks!

- **Papers for the Above Work**
  - J. Wang, D. Crawl, I. Altintas. *Kepler + Hadoop – A General Architecture Facilitating Data-Intensive Applications in Scientific Workflow Systems*. In Proc. of the 4th Workshop on Workflows in Support of Large-Scale Science (WORKS09) at Supercomputing 2009 (SC2009) Conf..
  - J. Wang, I. Altintas, P. R. Hosseini, D. Barseghian, D. Crawl, C. Berkley, M. B. Jones. *Accelerating Parameter Sweep Workflows by Utilizing Ad-hoc Network Computing Resources: an Ecological Example*. In Proceedings of IEEE 2009 Third International Workshop on Scientific Workflows (SWF 2009), 2009 Congress on Services (Services 2009).
- **More Information:**
  - Distributed Execution Interest Group of Kepler: https://dev.kepler-project.org/developers/interest-groups/distributed
  - Contact: jianwu@sdsc.edu

Kepler