Distributed Execution Architectures in Kepler

Jianwu Wang\textsuperscript{1}, Daniel Crawl\textsuperscript{1},
Ilkay Altintas\textsuperscript{1}, Chad Berkley\textsuperscript{2}, Matthew B. Jones\textsuperscript{2}

\textsuperscript{1} San Diego Supercomputer Center, UCSD
\textsuperscript{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
  – …
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
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**
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

![Graph showing different configurations for execution time vs. Spatio-temporal (X*Y*E)]
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
(a) MapReduce actor. (b) Map sub-workflow in MapReduce actor. (c) Reduce sub-workflow in MapReduce actor.
MapReduce Actor Execution in Hadoop

2. MapReduce Actor execution on Hadoop

1. Transfer input data from local FS to HDFS.

3. Transfer output data from HDFS to local FS.

- Master
  - distribute Kepler execution engine with Map sub-workflow
  - distribute Kepler execution engine with Reduce sub-workflow

- Hadoop Master
  - Hadoop Slaves
    - Map slaves with input data blocks
    - Reduce slaves with output data blocks

- Kepler GUI
Using MapReduce Actor for Word Count

Word count workflow in Kepler

Map sub-workflow

Reduce sub-workflow

Sub-workflow in IterateOverArray actor
Performance Experiment for Word Count

![Graph showing the performance experiment for word count. The graph compares Kepler Workflow in Hadoop and Java Program in Hadoop across different Slave Node Numbers. The total execution time decreases as the number of slave nodes increases.]
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
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
Thanks!

• **Papers for the Above Work**

• **More Information:**
  – Distributed Execution Interest Group of Kepler: [https://dev.kepler-project.org/developers/interest-groups/distributed](https://dev.kepler-project.org/developers/interest-groups/distributed)
  – Contact: jianwu@sdsc.edu