A High-Level Distributed Execution Framework for Scientific Workflows

Jianwu Wang[1], Ilkay Altintas[1],
Chad Berkley[2], Lucas Gilbert[1], Matthew B. Jones[2]

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

Outline

- **Introduction**
- **Background**
  - Scientific Workflow Specification Structure
  - Requirements for Distributed Execution using Scientific Workflows
- **Our Conceptual Architecture**
- **Working Mechanisms**
- **Case Study**
- **Conclusion and Future Work**

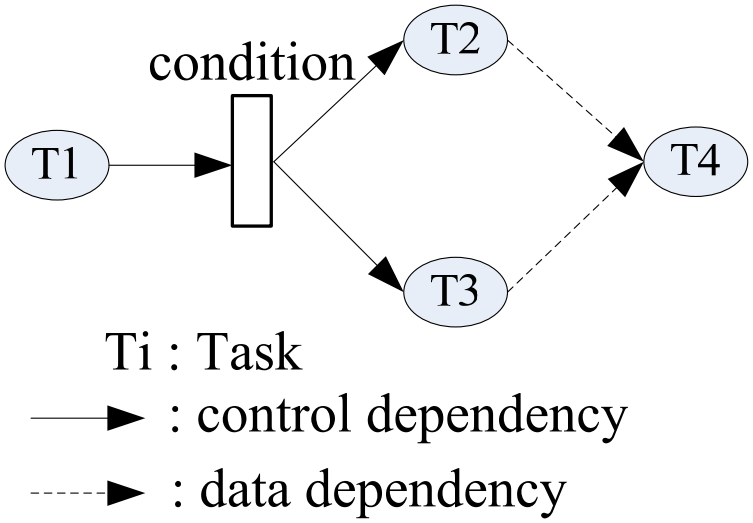
Introduction

- Scientific workflow can help domain scientists solve scientific problems.
- Most workflow systems centralize execution, which often causes a performance bottleneck.
- Distributed execution of scientific workflows is a growing and promising way to achieve better execution performance and efficiency.

Scientific Workflow Specification Structure

Focus: basic scientific workflow specification structure

- tasks
- data dependencies
- control dependencies



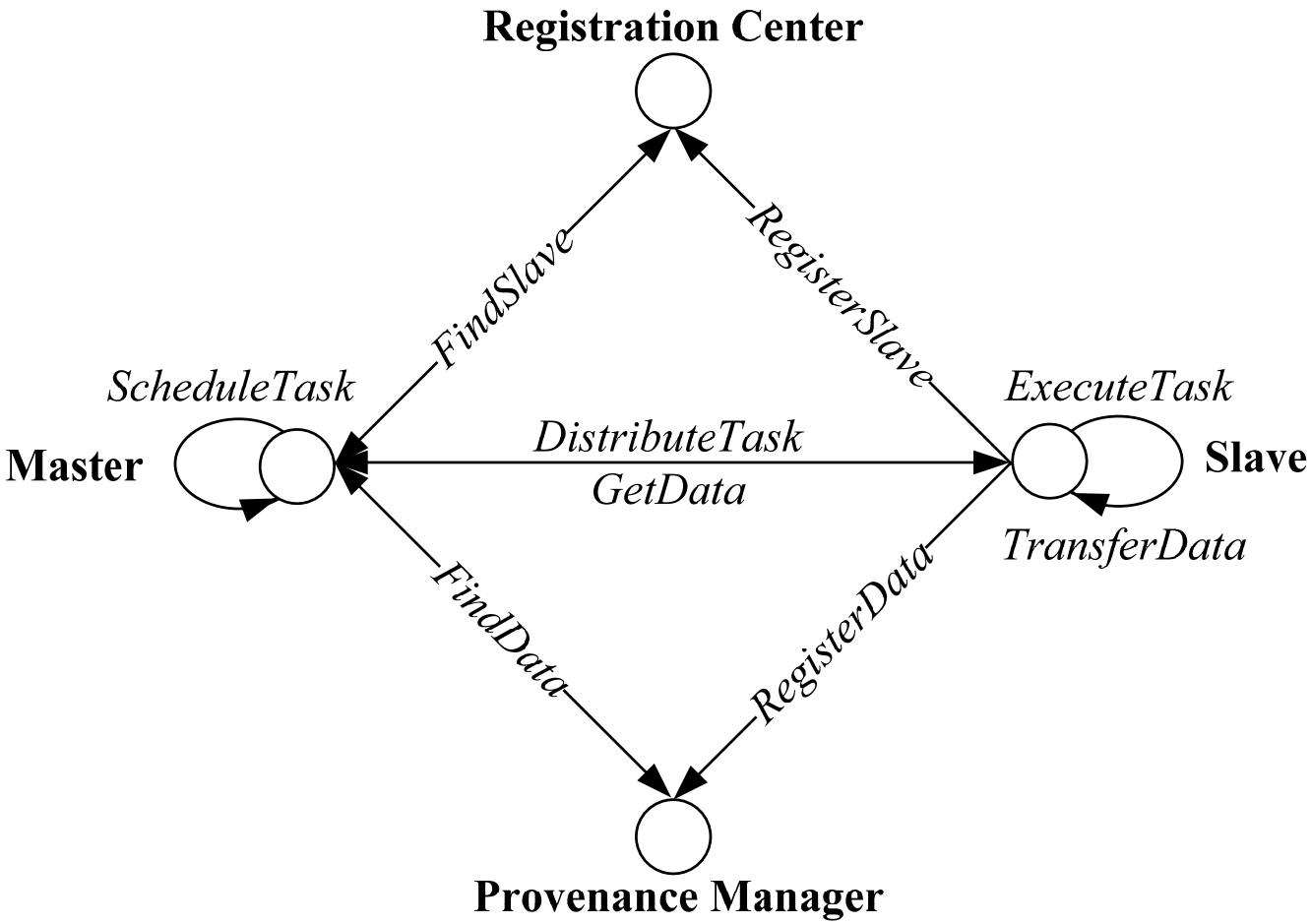
Requirements for Distributed Execution using Scientific Workflows

- Execution of Tasks on Remote Nodes
- Distributed Node Discovery
- Peer-to-Peer Data Transfer
- Provenance of Distributed Execution
- Distributed Monitoring
- Transparent Implementation
- Reuse of Existing Workflows
- Failure Recovery

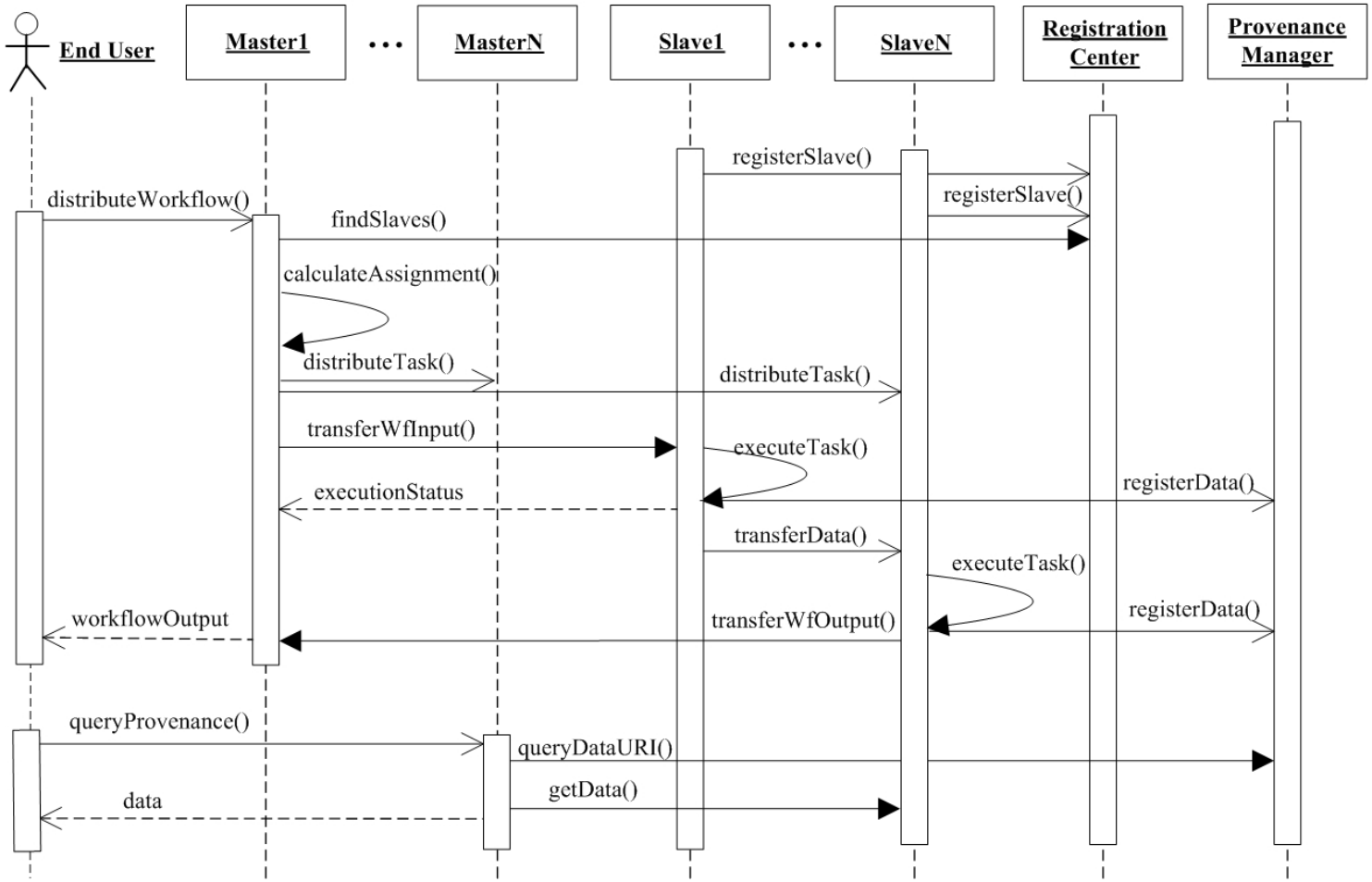
Our Goals

- ***Easy-to-use***
- ***Comprehensive***
- ***Adaptable***
- ***Extensible***
- ***Efficient***

Conceptual Architecture



Interaction sequence of a distributed scientific workflow execution



Working Mechanisms (1/5)

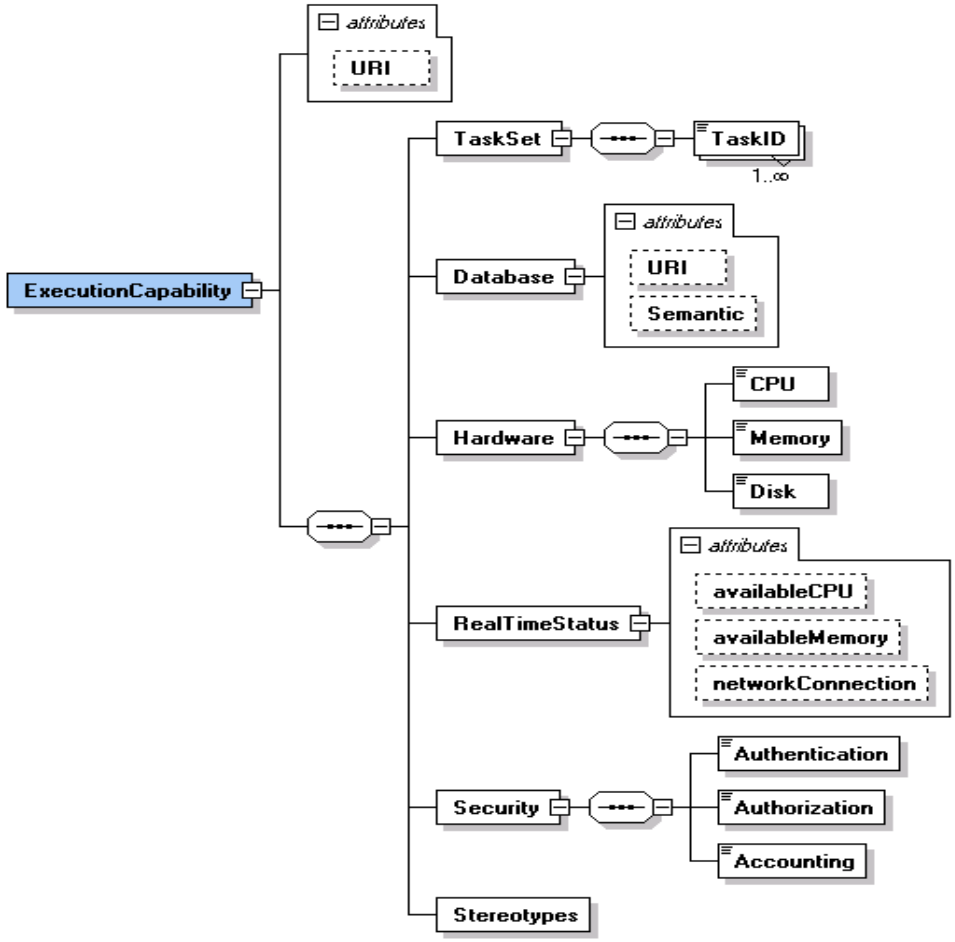
- **Decoupling of the Workflow Specification from the Execution Model**
  - Ability to use existing workflow specifications with both centralized and distributed execution models, i.e., workflow engine
    - Simply replacing the Director in Kepler
- **Peer-to-Peer Data Transfer**
  - A corresponding pipeline for each data dependency
  - Data flows from source Slave to destination Slave(s) directly

Working Mechanisms (2/5)

- **Transparent Implementation**
  - Define technology selection rule, detect and adapt to the context of real situations
  - Ease of deployment
    - Each node running workflow instance can act as an execution endpoint in either the Master or Slave role.

Working Mechanisms (3/5)

- **Capability-Based Slave Registration**



Slave Execution Capability Metamodel

Working Mechanisms (4/5)

- **Automatic Constraint-Based Task Scheduling**
  - Match user requirements with Slave execution capabilities to get optimal task scheduling solutions
    - Meet both functional requirements and non-functional constraints
  - Need new task scheduling algorithms
    - Run-times of some tasks vary with different input configuration
    - Take the task's input and configuration values into account

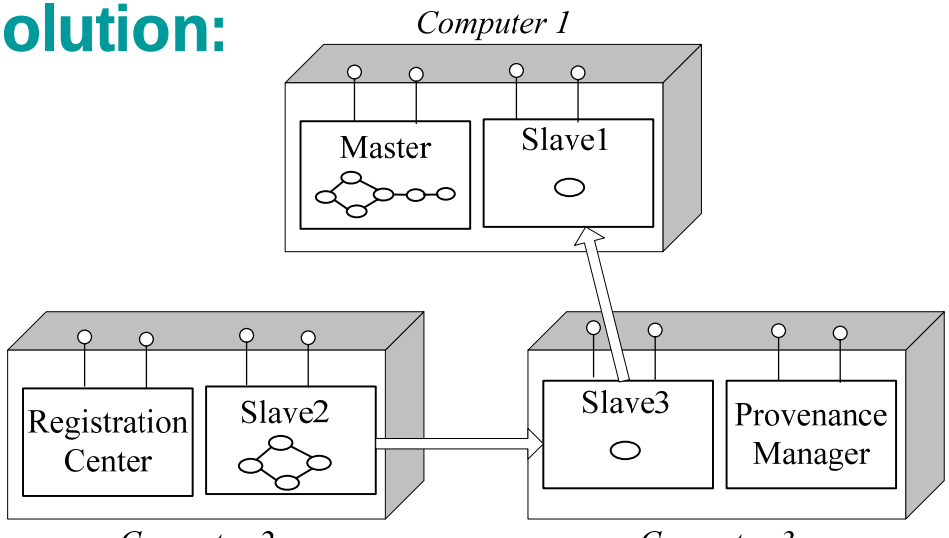
Working Mechanisms (5/5)

- **Broker based Provenance Management**
  - **Centralized**
    - Inefficient to store the data content
  - **Decentralized**
    - Efficient, but difficult to query and integrate the data in the future
  - **Broker-based**
    - *Tradeoff between functionality and efficiency*
      - Each slave records the data locally and register it to Provenance Manager.
      - Provenance Manager only record the reference info
      - The Master node can get the data content from the corresponding Slaves

Case Study

- **Scenario:**
  - A group of three scientists collaboratively construct a workflow with tasks in their sub-domains.
    - The workflow can't be executed as a whole on any of their computers.
  - They hope to:
    - Connect their computers *(Computer 1, Computer 2, Computer 3)* to execute the workflow
    - Track the provenance information

- **Solution:**



Conclusion and Future Work

- **A high-level distributed execution framework**
  - Based on requirements from the Kepler community
- **Discuss its main working mechanisms.**
- **Main focus on its usability in terms of adoption in our community**
  - Refine the design details
  - Finish implementation in Kepler
  - Evaluate it with applications

A quick demo…

Simhofi workflow: Terrestrial ecology

(With Parviez Hosseini from Princeton University)

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

Related Work

- **Several scientific workflow systems support distributed execution.**
  - Triana
    - Peer-to-peer execution
    - Intuitive graphical user interface
  - Pegasus
    - Execute workflows in Grid environments
    - Provenance support
  - ASKALON
    - Service repository to share service
    - Data repository to share data