Distributed Computing in Kepler
Ilkay Altintas
Lead, Scientific Workflow Automation Technologies Laboratory
San Diego Supercomputer Center, UCSD
(Joint work with Matthew Jones)

6th Biennial Ptolemy Miniconference
Berkeley, CA
May 12, 2005

Distributed Computation is a Requirement in Scientific Computing

Scientific workflows do scientific computing!

- Increasing need for data and compute capabilities
- Data and computation should be combined for success!
  - HEC + Data management/integration
Kepler and Grid Systems
-- Early Efforts --

- Some Grid actors in place
  - Globus Job Runner, GridFTP-based file access, Proxy Certificate Generator
  - For one job execution! Can be iterated…
- SRB support
- Interaction with Nimrod and APST
- Grid workflow pattern:
  STAGE FILES -> EXECUTE -> FETCH FILES
  Execute==Schedule -> Monitor & Recover

Issues: Data and process provenance, user interaction, reporting and logging

NIMROD and APST

GOAL: To use the expertise in scheduling and job maintenance
Distributed Computing is Team Work

- Login to, create, join Grids & role-based access
- Access data
- Execute services
- Discover & use existing workflows
- Design, share, annotate, run and register workflows

So our distributed computing framework should support collaborations!
...as well as it should keep control of scheduling and provenance information...

Goals and Requirements

- Two targets:
  - Distributing execution
    - Users can configure Kepler Grid access and execution parameters
    - Kepler should manage the orchestration of distributed nodes.
    - Kepler will have the ability to do failure recovery
    - Users can be able to detach from the workflow instance after they and then connect again
  - Supporting on the fly online collaborations
    - Users can log into Kepler Grid and form groups
    - Users can specify who can share the execution
Peer-to-Peer System Satisfies These Goals

- A peer-to-peer network:
  - Many or all of the participating hosts act both as client and server in the communication
- The JXTA framework provides:
  - Peers
  - Peer Groups
  - Pipes
  - Messages
    - Queries and responses for metadata
    - Requests and responses to move workflows and workflow components as .ksw files
    - Data flow messages in executing workflows

Creating KeplerGrid using P2P Technology

Setting up Grid parameters

Register as a peer

Configure
Creating KeplerGrid using P2P Technology

Creating, Joining & Leaving Grids

P2P/JXTA Director
- Decides on the overall execution schedule
- Communicates with different nodes (peers) in the Grid
- Submits distributable jobs to remote nodes
- Can deduce if an actor can run remotely from its metadata
- Configuration parameters:
  - Group to join
- Can have multiple models:
  - Using a “master peer” and static scheduling is the current focus

Work in progress…
Creating KeplerGrid using P2P Technology

Provenance, Execution Logs and Failure Recovery

- Built in services for handling failures and resubmission
  - Checkpointing
  - Store data where you execute it & send back metadata
  - The “master peer” collects the provenance information

How can we do it without having a global job database?

Work in progress...

Status of Design and Implementation

- Initial tests with Grid creation, peer registration and discovery
- Start with a basic execution model extending SDF
- Need to explore different execution models
  - More dynamic models seem more suitable
- Big design decisions to think on:
  - What to stage to remote nodes
  - Scalability
  - Detachability
  - Certification and security
To sum up…

- Just distributing the execution is not enough
  *Need to think about the usability of it!*
- Need to have sub-services using the JXTA model for
  - peer discovery,
  - data communication,
  - logging,
  - failure recovery.
- Might need more than one domain for different types of distributed workflows

Questions?..

Thanks!

Ilkay Altintas
altintas@sdsc.edu
+1 (858) 822-5453
http://www.sdsc.edu