Introduction to Scientific Workflow Management and the Kepler System

Ilkay Altintas¹, Terence Critchlow², Zhijie Guan¹, Ayla Khan³, Scott Klasky⁴, Jeff Ligon⁵, Bertram Ludaescher⁶, Pierre Mouallem⁵, Steve Parker³, Norbert Podhorszki⁶, Bill Romine², Mladen Vouk⁵

1. San Diego Supercomputer Center
2. Lawrence Livermore National Laboratory
3. University of Utah
4. Oak Ridge National Laboratory
5. North Carolina State University
6. University of California at Davis
Others on the team

• Developers:
  • Zhengang Cheng (NCSU), George Chin (PNNL), Arie Shoshani (LLBL), Claudio Silva (U-Utah).

• End-Users:
  • John Blondin (NCSU), Matt Coleman (LLNL), Scott Klasky (ORNL), Doug Swesty (SUNY/SB), …. 

• More ..
  • ..including numerous Kepler and Ptolemy developers and app users!
Overview

• Motivation
• Introduction
• Architecture
• Existing Workflows
• Hands on
• Wrap-up
Topics

I. Brief Definition of the Scientific Workflows with Motivational Examples

II. User and Technical Requirements for Scientific Workflows, Existing Systems
DOE Scientific Data Management Center

- **Goals**: integrate and deploy software-based solutions to **efficiently and effectively manage** large volumes of **data** generated by scientific applications.

SDM Center focuses on the application of known and emerging data management technologies to scientific applications.

[Diagram showing data processing and analysis]

**http://sdm.lbl.gov/sdmcenter/**

Funded under SciDAC
A Typical SDM Scenario

Control Flow Layer

Applications & Software Tools Layer

I/O System Layer

Storage & Network Resources Layer

Flow Tier

Work Tier

Task A: Generate Time-Steps
Task B: Move TS
Task C: Analyze TS
Task D: Visualize TS

Simulation Program
Data Mover
Post Processing
Parallel R
Terascale Browser

Parallel NetCDF
PVFS
Sabul
SRM
HDF5 Libraries

NetCDF
PVFS
Sabul
SRM
HDF5 Libraries

SC06/Kepler Tutorial/V7/Nov-06
Scientific Process Automation (SPA)

- SPA = Scientific Workflow layer in the SDM Center 3-tier architecture
- Part of the Kepler Scientific Workflow System and Collaboration

[Diagram showing layers and components related to SPA, including WorkFlow Management Engine (Kepler), Scientific Workflow Components, Data Mining and Feature Identification, Parallel R Statistical Analysis, Efficient indexing (Bitmap Index), Efficient Parallel Visualization (pVTK), Storage Resource Manager (To HPSS), Parallel NetCDF, Parallel I/O (ROMIO), Parallel Virtual File System.]
What are Scientific Workflows?

• Workflows:
  • Workflow management systems help in the *construction* and *automation* of scientific problem-solving processes that include executable sequences of components and data flows.
  • Scientific workflow systems often need to provide for load balancing, *parallelism*, and complex *data flow* patterns between servers on distributed networks.
  
  • ... *aiming to solve complex scientific data integration, analysis, management, visualization tasks*
  • in plainer English: doing hard and/or messy stuff, and making it look easy.
Why use Scientific Workflows?

- It’s too hard to keep moving my data, launching analysis services, and visualizing my data.
- If I don’t archive my data before the end of my run, I will lose data.
- I want to analyze my data as soon as it’s generated.
- I want to keep track of everything in the workflow (provenance) in case there is a problem.
- I don’t want to learn 10 million technologies just to continue running on the large computers.
- I write my own analysis routines, and I don’t want to learn new packages just to do my basic science.
Two typical types of Workflows for SC

- **Real-time Monitoring (Server Side Workflows)**
  - Job submission.
  - File movement.
  - Launch Analysis Services.
  - Launch Visualization Services.
  - Launch Automatic Archiving.
- **Post Processing (Desktop Workflows)**
  - Read in Files from different locations.
  - File movement.
  - Launch Analysis Services.
  - Launch Visualization Services.
  - Connect to Databases.
- Obviously there are other types of workflows.
  - Do you guys have different types?
A few days in the life of Sim Scientist. Day 1 -morning.

- **8:00AM Get Coffee, Check to see if job is running.**
  - Ssh into jaguar.ccs.ornl.gov (job 1)
  - Ssh into seaborg.nersc.gov (job 2) (this is running yea!)
  - Run gnuplot to see if run is going ok on seaborg. This looks ok.
- **9:00AM Look at data from old run for post processing.**
  - Legacy code (IDL, Matlab) to analyze most data.
  - Visualize some of the data to see if there is anything interesting.
  - Is my job running on jaguar? I submitted this 4K processor job 2 days ago!
- **10:00AM scp some files from seaborg to my local cluster.**
  - Luckily I only have 10 files (which are only 1 GB/file).
- **10:30AM first file appears on my local machine for analysis.**
  - Visualize data with Matlab. Seems to be ok. 😊
- **11:30AM see that the second file had trouble coming over.**
  - Scp the files over again... Dohhh
A few days in the life of Sim Scientist. Day 1 evening.

- **1:00PM Look at the output from the second file.**
  - Opps, I had a mistake in my input parameters.
  - Ssh into seaborg, kill job. Emacs the input, submit job.
  - Ssh into jaguar, see status. Cool, it’s running.
  - bbcp 2 files over to my local machine. (8 GB/file).
  - Gnuplot data.. This looks ok too, but still need to see more information.

- **1:30PM Files are on my cluster.**
  - Run matlab on hdf5 output files. Looks good.
  - Write down some information in my notebook about the run.
  - Visualize some of the data. All looks good.
  - Go to meetings.

- **4:00PM Return from meetings.**
  - Ssh into jaguar. Run gnuplot. Still looks good.
  - Ssh into seaborg. My job still isn’t running……

- **8:00PM Are my jobs running?**
  - ssh into jaguar. Run gnuplot. Still looks good.
  - Ssh into seaborg. Cool. My job is running. Run gnuplot. Looks good this time!
And Later

- **4:00AM yawn... is my job on jaguar done?**
  - Ssh into jaguar. Cool. Job is finished.
  - Start bbcp files over to my work machine. (2 TB of data).
- **8:00AM Bbcp is having troubles.**
  - Resubmit some of my bbcp from jaguar to my local cluster.
- **8:00AM (next day).**
  - Still need to get the rest of my 200GB of data over to my machine.
- **3:00PM My data is finally here!**
  - Run Matlab. Run Ensight. Oppps.... Something’s wrong!!!!!!!!!! Where did that instability come from?
- **6:00PM finish screaming!**
And 2 years from now…. 

- **Simulations /computers** are getting larger and **more expensive** to operate.  
  - In Fusion, large runs will be using >50K cores/wallclock hours, to understand turbulent transport in ITER size reactors.  
  - The cost of a simulation approaches $0.6M (power, cooling, system cost averaged over 5 years).

- **Data Sizes** are getting larger.
  - Large simulations produce 2 TB/simulation (today), 100TB/simulation(week) in the future.

- **Demand for ‘real-time’ monitoring/analysis of simulations.**
- **Demand for ‘fast-reliable data movement’ to local machines for ‘post processing’.”
Workflows to the rescue!

- In our demo section you will see us automate this process.
- **Job submission** starts services on ORNL IB cluster (ewok).
- Files are automatically **moved** from Cray XT3 to ORNL IB cluster.
- Files are converted from **binary** to hdf5 files.
- Files accumulate until they >= 6GB. Then they are **tarred**.
- Files use hsi commands to place tar files into **HPSS**. (xml file describes which files are in which tar files).
- **Hdf5** file is read into **SciRun** service which creates a jpeg.
- **Jpeg** files create an mpeg file via a mpeg service.
- Jpeg and mpeg files are **moved** to web portal.
- Hdf5 files are **archived** to PPPL.
Why do Pflop computing scientists care?

Typical situation for Sim Scientist.
- We run on 1 – 60K processors, producing lots of data.
- Typical method of work.
  - Prepare input data for smaller simulations.
  - Iterate until we come up with the correct parameters for the large run.
  - Run the large simulation only at a handful number of locations (usually <4).
  - Must Archive results. Must be of the correct size archives on HPSS.
  - Must move some data over to our local clusters for analysis after the simulation.
  - Did we make a mistake with the input parameters? Is something going wrong? Fix the code/input: start the run over again.
    - Wow, I just wasted 100K CPU hours because I missed a – sign. Duhh.
  - Where are all of my files? I want to look at the temperature in the 200 time slice, where is it on HPSS.
Doesn’t everyone analyzed data on 100’s of processors?

- **100’s** maybe for the ‘truly’ advanced users.
  - In the future, some of the services will require this amount of processors for analysis.
  - Mostly determined by RAM.
  - Most analysis done on <4 processors.
- Workflow can execute services on clusters which operate on many processors.
  - Job scheduling an issue. Needs to be scheduled when the simulation starts.
  - Can waste cycles if we do not carefully design the workflow.
- **Memory is the key reason for parallel services.**
  - Use a round-robin technique on multiple processors.
Post Processing Workflow: A day in the life of Sim Scientist

• 9:00AM Get Diet Coke, decide which runs/experimental data to analyze.
• 9:30AM Start to download files from HPSS from NERSC and ORNL.
• 10:00AM Move files from NERSC/ORNL to local desktop machine. Smallish data (10GB/location).
• 11:00AM Start IDL, and compute various post processing quantities.
• 11:30AM look at the data from the simulations, and grab data from a database which has experimental data.
• 1:00PM Move some more data from ORNL to local desktop to compare to more experimental data.
  • Save the plot from Matlab to Postscript to include in paper.
  • Write down results into notebook, copy figure into notebook.
• 2:00PM Think about results, and decide on new analysis routines to write in the future.
• 4:00PM Start moving more data from NERSC to local desktop.
What’s changing in his life?

- **Collaboration.**
- More clusters, more simulations.
  - Just analyze the data where we run. Don’t move the data.
    - But...
      - What if the network goes (I’m on a plane,...).
      - What if the resource is not available for my late-breaking analysis before the BIG conference?
  - OK, but what about the large data?
    - OK. Large data will be server-side analysis. Not DESKTOP. But can run workflow on a server.
- **Data from multiple resources**
  - V&V = data from multiple simulations/experiments.
- **But can’t we just run VISIT/SciRun?**
  - Yes. But need to orchestrate the data movement from different sources, track the provenance, and perhaps use multiple analysis/visualization packages, then a workflow system can help.
How do we help this scientist?

- The workflow is the “glue” for the scientists.
- The scientists hooks up all of the analysis routines.
- The director makes sure that the data movement occurs, and is reliable, and secure.
- All of the tedious portions of ssh, start this program, is removed by the workflow automation.
- The workflow will be able to keep the provenance information which allows the user to understand how they processed the dataset.
  - This enables the scientist to compare new data with old data.
So what are the requirements?

• Must be **EASY to use.**
  • If you need a manual, then **FORGET IT!**
    (some may not agree)
  • Good user support, and long-term DOE support. 😊
• The workflow should **work for all of my workflows.**
  • NOT just for the Petascale computers.
  • And on multiple platforms!
• Must be **easy** to incorporate my own services into the workflow.
  • Services can be defined as a stand alone program.
  • What if I have stand-alone libraries?
  • Eventual CCA integration.
• Must be **customizable by the users.**
  • Users need to easily change the workflow to work with the way users work.
And more requirements.

- Long-term requirements. [NOT being worked on yet].
  - Autonomics.
    - Adapt to the network/resources to deal with failures.
  - Interactive User Changes.
    - For long-running simulations, the user may want to change the current workflow.
    - Faster data movement in the workflow? For latency-sensitive movement.
- High Quality front-end for the end-user interaction.
- You tell us!
Increasing Usage of Technology in Geosciences

- Online data **acquisition** and **access**
- Managing **large databases**
  - Indexing data on spatial and temporal attributes
  - Quick subsetting operations
- Large scale **resource** sharing and management
- **Collaborative** and **distributed** applications
- **Parallel** gridding algorithms on large data sets using **high performance computing**
- **Integrate** data with other related data sets, e.g. geologic maps, and hydrology models
- Provide **easy-to-use** user interfaces from portals and scientific workflow environments
SWF Systems Requirements

- **Design tools**— especially for non-expert users
- **Ease of use**— fairly simple user interface having more complex features hidden in the background
- **Reusable generic** features
  - Generic enough to serve to different communities but specific enough to serve one domain (e.g. geosciences) → customizable
- **Extensibility** for the expert user
- **Registration, publication & provenance** of data products and “process products” (=workflows)
- **Dynamic plug-in** of data and processes from registries/repositories
- **Distributed** WF execution (e.g. Web and Grid awareness)
- **Semantics** awareness
- **WF Deployment**
  - as a web site, as a web service, “Power apps” (a la SciRUN II)
The Big Picture: Supporting the Scientist

From “Napkin Drawings”…

… to Executable Workflows

Conceptual SWF

Executable SWF

Here:

John Blondin, NC State Astrophysics
Terascale Supernova Initiative
SciDAC, DOE
Promoter Identification Workflow (PIW)

**Step 1:** Microarray Analysis
- microarray data

**Step 2:** Clusfavor Analysis
- Gene ID
  - GDBank sequence retrieval
  - cDNA sequence
  - genomics search

**Step 3:** NCBI BLAST search
- genomic sequence

**Step 4:** Transfac search

**Step 5:** New candidate target genes
- Consensus sequence

**Step 6:** Promoter Identification
- promoter data
- Transcription factor binding

**Step 7:** Promoter Model generator

**Step 8:** NCBI BLAST search
**CPES Fusion Simulation Workflow**

- **Fusion Simulation Codes:** (a) GTC; (b) XGC with M3D
  - e.g. (a) currently 4,800 (soon: 9,600) nodes Cray XT3; 9.6TB RAM; 1.5TB simulation data/run

- **GOAL:**
  - automate remote simulation **job submission**
  - continuous **file movement** to secondary **analysis cluster** for dynamic visualization & simulation control
  - ... with **runtime-configurable observables**

![Diagram of Workflow]

- **WF design & implementation:** Norbert Podhorszki (UC Davis)
- **Overall architect (& prototypical user):** Scott Klasky (ORNL)
**CPES Analysis Workflow**

- Concurrent analysis pipeline (@Analysis Cluster):
  - convert; analyze; copy-to-Web-portal
  - easy configuration, re-purposing

**Overall architect (& prototypical user): Scott Klasky (ORNL)**

**WF design & implementation: Norbert Podhorszki (UC Davis)**
Scientific Workflow Systems

• Combination of
  • Data management, integration, analysis, and visualization steps
  • Larger, automated "scientific process"

• Mission of scientific workflow systems
  • Promote “scientific discovery” by providing tools and methods to generate scientific workflows
  • Provide an extensible and customizable graphical user interface for scientists from different scientific domains
  • Support workflow design, execution, sharing, reuse and provenance
  • Design frameworks which define efficient ways to connect to the existing data and integrate heterogeneous data from multiple resources

• Make technology useful through user’s computer!!!
Discussion

• What types of workflow are you most interested in?
• Which technologies do you use in your workflows?
• How do you automate your workflows now?
Overview

• Motivation
• Introduction
• Architecture
• Existing Workflows
• Hands on
• Wrap-up
Topics

III. Foundations of Scientific Workflows

IV. The Kepler Collaboration and System Architecture
Summary of Scientific Workflows so far …

- Need for SWFs from eScience, CI, and SDM

- Goal: (semi-)**automation** of tasks

- **Optimize** not just CPU & memory, but **human time**!
  - Scripting (e.g. Python, Perl) is not for everybody, harder to maintain, modify, share, explain; no built-in provenance, pipeline parallelism; collaboration harder
Kepler is a Scientific Workflow System

www.kepler-project.org

- ... and a cross-project collaboration
- Latest release available from the website
- Builds upon the open-source Ptolemy II framework

**Ptolemy II**: A laboratory for investigating design

**KEPLER**: A problem-solving environment for Scientific Workflow

**KEPLER** = “Ptolemy II + X” for Scientific Workflows
Kepler is a Team Effort

Other contributors:
- Chesire (UK Text Mining Center)
- DART (Great Barrier Reef, Australia)
- National Digital Archives + UCSD-TV (US)
- …

Contributor names and funding info are at the Kepler website!!
Co-development in Kepler

**SQL database access (JDBC)**

- Queries:
  - Database connection
  - Database query
  - XML result

**GEON Dataset Generation & Registration**

- Workflows used to extract gravity lat long point from a remote database and generate shapefiles using a web service.

**% Makefile**

```
% Makefile
$> ant run
$> ant run
```

**Matt et al. (SEEK)**

**Efrat (GEON)**

**Ilkay (SDM)**

**Yang (Ptolemy)**

**Xiaowen (SDM)**

**Edward et al. (Ptolemy)**

Extract the array `xmlToken` elements and translate them into strings.

```
XPathFactory

Constant

ArrayToElement

TokenToString

xmlns:
```

Remove begin and end tags from the element string.

```
XPathFactory

Constant

StringLength

ArrayEnd

ArrayStart

StartElement

EndElement
```

**DOE Scientific Data Management Center – Scientific Process Automation**
DOE Scientific Data Management Center – Scientific Process Automation

Vergil is the GUI for Kepler

- Actor ontology and semantic search for actors
- Search -> Drag and drop -> Link via ports
- Metadata-based search for datasets
Actor Search

• Kepler Actor Ontology
  • Used in searching actors and creating conceptual views (= folders)

Currently 160+ Kepler actors added!
DOE Scientific Data Management Center – Scientific Process Automation

Data Search and Usage of Results

• **Kepler DataGrid**
  – Discovery of data resources through local and remote services
  – SRB,
  – Grid and Web Services,
  – Db connections
  – Registry of datasets on the fly using workflows

A simple example of using EML data. First, a search is done in the pane to locate an EML-described data set, which is dragged onto the workflow canvas. The EML data source is added to the workflow, and contacts the EcoGrid server to download the data and configure the service. After being configured, it displays the ports from the EML data source, which are then mapped into an XY scatterplot.
Scientific Workflow Design: Challenges
Actor-oriented Modeling And Design (AMAD)

While many systems (including Kepler) support execution ... support for SWF conceptual modeling and design is lacking.

- Formal models for scientific workflows
  Based on Actor-Oriented Modeling
- Mechanisms for discovery, reuse, and adaptation of existing workflows and components
  plus a rich Type System (“hybrid” types)
- End-to-end workflow development (methods and frameworks), especially for early stages
  Modeling Primitives
  (Adapters & Replacement; strategies)
Actor-Oriented Modeling

Actors

- single component or task
- well-defined interface (signature)
- generally a passive entity: given input data, produces output data
Actor-Oriented Modeling

Ports

- each actor has a set of input and output ports
- denote the actor’s signature
- produce/consume data (a.k.a. tokens)
- parameters are special “static” ports
Actor-Oriented Modeling

Dataflow Connections

- actor “communication” channels
- directed (hyper) edges
- connect output ports with input ports
- merge step + distribute step
Actor-Oriented Modeling

Sub-workflows / Composite Actors
- composite actors “wrap” sub-workflows
- like actors, have signatures (i/o ports of sub-workflow)
- hierarchical workflows (arbitrary nesting levels)
Actor-Oriented Modeling

Directors

- define the **execution semantics** of workflow graphs
- executes workflow graph (some schedule)
- sub-workflows may have different directors
- enables reusability
Some actors in place for...

- Actors that wrap **CCA** components
- Generic **Web Service** Client and Web Service Harvester
- Customizable **RDBMS query** and update
- **Command Line** wrapper tools
  - local, ssh, scp, ftp, etc.
- Some **Grid** actors
  - Globus Job Runner, GridFTP, Certificate Generator
- **SRB** support
- Native **R** and **Matlab** support
- Interaction with parameter sweep tools
  - **Nimrod** and **APST**
- Communication with ORBs through actors and services
- Image Processing, Gridding, Visualisation Support
- Textual and Graphical Output
- Even wrapping subsystems, such as SCIRun
- ...more generic and domain-oriented actors...
Some KEPLER Actors (out of 160+ … and counting…)

Diagram showing various KEPLER Actors such as WebService, DatabaseQuery, XSLT, BrowserUI, FileFetcher, FileStager, SDF Director, Globus, and more.
Some actor extensions in the works...

- Large-scale robust data mover

- Application-driven extensions (SDM):
  - Access to/integration with other SDM components
    - PnetCDF, PVFS(2), MPI-IO, parallel-R, ASPECT, FastBit, CCA support...

- Further generic extensions:
  - Additional support for data-intensive and compute-intensive workflows
  - Semantics-intensive actors
  - Actors to work with other Kepler features for
    - Authentication and authorization
    - Distributed execution
    - Execution monitoring
    - Fault tolerance
Dataflow as a Computation Model

- Dataflow Model of Computation:
  - Alternative to: stored-program (von Neumann) execution
- Sound, simple, powerful model of **parallel** computation
  - Formal model: variations of Kahn Process Networks (PN)
    - “built in”: task and pipeline parallelism!
  - NOT having a locus of control makes it simple!
  - Naturally distributed model of computation

- **Asynchronous**: Many actors can be ready to fire simultaneously
  - Execution ("firing") of a node starts when (matching) data is available at a node's input ports.
    - **Locally** controlled events
      - Events correspond to the “firing” of an actor
    - **Actor**:
      - A single instruction
      - A sequence of instructions
  - Actors **fire** when all the inputs are available
Directors are the WF Engines that…

- Implement different models of computation (MoCs)
  - Define the semantics of
    - execution of actors and workflows
    - interactions between actors

Ptolemy and thus Kepler are unique in their ability to use and even combine different MoCs

- Dataflow
- Time Triggered
- Synchronous/reactive model
- Discrete Event
- Wireless
- Process Networks
- Rendezvous
- Publish and Subscribe
- Continuous Time
- Finite State Machines
More on Models of Computation (MoCs)

Directors separate the concerns of WF orchestration from Actor execution

- **Directed Acyclic Graph (DAG)**
  - Common among Grid workflows: no loops, each actor fires at most once (no streaming / pipeline parallelism)
  - Example: DAGMan

- **Synchronous Dataflow (SDF)**
  - Connections have queues for sending/receiving fixed numbers of tokens at each firing. Schedule is statically predetermined. SDF models are highly analyzable and used often in SWFs.

- **Process Networks (PN)**
  - Generalize SDF. Actors execute as a separate thread/process, with queues of unbounded size. Related to Kahn/MacQueen semantics.

- **Continuous Time (CT)**
  - Connections represent the value of a continuous time signal at some point in time ... Often used to model physical processes.

- **Discrete Event (DE)**
  - Actors communicate through a queue of events in time. Used for instantaneous reactions in physical systems.
Challenges in Scientific Workflow Design

- Describing components in the users’ terms ➔ semantic types

- Dealing with complex control-flow (e.g., case-switch, exception handling etc.) ➔ different modeling paradigms, templates and frames
• Scientific Workflow Life-cycle
  • Resource Discovery
    • discover relevant datasets
    • discover relevant actors or workflow templates
  • Workflow Design and Configuration
    • data ⇒ actor (data binding)
    • data ⇒ data (data integration / merging / interlinking)
    • actor ⇒ actor (actor / workflow composition)

• Challenge: do all this in the presence of …
  • 100’s of workflows and templates
  • 1000’s of actors (e.g. actors for web services, data analytics, …)
  • 10,000’s of datasets
  • 1,000,000’s of data items
  • … highly complex, heterogeneous data
  • price to pay for these resources: $$$ (lots)
  • scientist’s time wasted: priceless!
Approach & Kepler/SMS Capabilities

• Employ semantic extensions (ontologies) for ..
  • **Smart Search** (Resource Discovery)
  • **Smart Attach** (Data Binding)
  • **Smart Integration** (Transform/Merge Data)
  • **Smart Links** (Actor Composition, WF Design)

* by “smart” we mean these services are informed by metadata and ontology information

• Characteristics of SMS work
  • big chunk of basic computer science research (references)
  • ... but also implement this (link to Kepler)
  • ... driven by real-world use cases (link to BEAM)
  • ... on top of community ontologies (link to KR team)
Semantic Type Annotation in Kepler

- Component input and output port annotation
  - Each port can be annotated with multiple classes from multiple ontologies
  - Annotations are stored within the component metadata
Component Annotation and Indexing

- Component Annotations
  - New components can be annotated and indexed into the component library (e.g., specializing generic actors)
  - Existing components can also be revised, annotated, and indexed (hiding previous versions)
DOE Scientific Data Management Center – Scientific Process Automation

**Smart Search**

Find a component (here: an actor) in different locations (“categories”)
- ... based on the semantic annotation of the component (or its ports)
- ... needs one (or more) ontologies to “register against” (⇒ KR)

Browse for Components  Search for Component Name  Search for Category / Keyword
Smart (Data) Integration: Merge

- Discover data of interest
- ... connect to merge actor
- ... “compute merge”
  - align attributes via annotations
  - open dialog for user refinement
  - store merge mapping in MOML
- ... enjoy!
  - ... your merged dataset
Under the hood of ‘Smart Merge’

- Exploits semantic type annotations and ontology definitions to find mappings between sources
- Executing the merge actor results in an integrated data product (via “outer union”)

### Merge Result

<table>
<thead>
<tr>
<th>a1</th>
<th>a3</th>
<th>a4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5.0</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td>6.0</td>
<td>11</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0.3</td>
<td></td>
</tr>
</tbody>
</table>
Smart Linking (Workflow Design)

- Statically perform semantic and structural type checking

- Navigate errors and warnings within the workflow

- Search for and insert “adapters” to fix (structural and semantic) errors ...
Smart Linking (Data-Actor, Actor-Actor)

**Ontologies (OWL)**

- **Semantic Type** $P_s$
- **Semantic Type** $P_t$
- **Structural Type** $P_s$
- **Structural Type** $P_t$
- **Correspondence** ($\subseteq$)

**Compatible**

**Transformation**

- **Source Service** $P_s$
- **Target Service** $P_t$

**Desired Connection**

Source: [Bowers-Ludaescher, DILS’04]

Related: Wang & Goguen (SCIA)

SC06/Kepler Tutorial/V7/Nov-06
Complexity in Scientific Workflow Design

The use of “control-flow” primitives

- Managing complex, **nested** data structures (select/filter/transform)
- Fault-tolerance and exception handling

..."wired-in", e.g. via Boolean switches, complex branching and looping

Custom actors, **hand-crafted** control flow limited to sequential execution (SSDBM’03)
Modeling Control-Flow Constructs in Dataflow

- **Dataflow** in Kepler
  - Based on dataflow process networks (Kahn et al, Lee)
  - Supports pipeline parallelism (streaming data)
  - Natural paradigm for data-driven workflows
  - Efficient analysis and scheduling
  - Intuitive model for workflow designers

- **Control-Flow** in Kepler
  - Branching via if-then-else and switch-case statements
  - Iteration with multiple entry and exit points
  - Low-level actors for manipulating structure (e.g., record-to-array)

- **Problems** modeling Control-Flow directly using Dataflow
  - Overly complicated workflows; hard to understand (low-level programming), maintain, debug, extend \(\rightarrow\) limited reusability; complex re-configuration

New Design Primitives (e.g., templates & frames)

[Bowers-et-al, SciFlow’06]
Workflow Design Paradigms

- (a) Vanilla Process Network
- (b) Functional Programming Dataflow Network
- (c) XML Transformation Network
- (d) Collection-oriented Modeling & Design framework
Provenance
Kepler Provenance Framework

- **Provenance**: the history of execution and conditions applied to the workflow

  Why do we need provenance?
  - Because science evolves...
    - Recreate results and rebuild workflows using the evolution information
    - Associate the workflow with the results it produced
    - Create links between generated data in different runs, and compare different runs
    - Checkpoint a workflow and Recover from a system failure
    - Debug and explain results (via lineage tracing, ...)
    - Smart Reruns

- Types of Provenance Information:
  - **Data provenance**
    - Intermediate and end results including files and db references
  - **Process provenance**
    - Keep the wf definition with data and parameters used in the run
  - **Error and execution logs**
  - **Workflow design provenance**
Kepler Provenance Recording Utility

- Parametric and customizable
  - Different report formats
  - Variable levels of detail
    - Verbose-all, verbose-some, medium, on error
  - Multiple cache destinations
- Saves information on
  - User name, Date, Run, etc...
What other system functions does provenance relate to?

- Failure recovery
- Smart re-runs
- Semantic extensions
- Kepler Data Grid
- Reporting and Documentation
- Authentication
- Data registration

Re-run only the updated/failed parts

Documentation generation and updates
Provenance for the WF Engineer / Plumber

• A Workflow Engineer’s View
  • Monitor, benchmark, and optimize workflow performance
  • Record resource usage for a workflow execution
  • “Smart Re-run” of (variants of) previous executions
  • Checkpointing & restart (e.g. for crash recovery, load balancing)
  • Debug or troubleshoot a workflow run
  • Explain when, where, why a workflow crashed
Provenance for Domain Scientists!

- Query the lineage of a data product
  - from what data was this computed? ("real" dependencies please!)
- Evaluate the results of a workflow
  - do I like how this result was computed?
- Reuse data products of one workflow run in another
  - (re-)attach prior data products to a new workflow
- Archive scientific results in a repository
- Replicate the results reported by another researcher
- Discover all results derived from a given dataset
  - ... i.e. across all runs
- Explain unexpected results
  - ... via parameter-, dataset-, object-dependencies in the scientist’s terms (yes, you may think “ontology” here ... )
Example Scientific Workflow (Kepler)
Scientific provenance questions we can ask about a run of this workflow

- What DNA sequences were input to the workflow (this run)?
- What phylogenetic trees were output by the workflow?
- What phylogenetic trees were created (intermediate or final) by the workflow?
- What actor created this phylogenetic tree?
- What sequences input to the workflow does this consensus tree depend on?
- What input sequences were not used to derive any output consensus trees?
- What was the sequence alignment (key intermediate data) used in the process of inferring this tree?
- Which actors were involved in creating this tree?
Architecture
Kepler can be used as a batch execution engine

- **Configuration phase**
- **Subset**: DB2 query on DataStar

**Subset** → **Analyze** → **Visualize**

- **Interpolate**: Grass RST, Grass IDW, GMT...
- **Visualize**: Global Mapper, FlederMaus, ArcIMS
Kepler System Architecture
Kepler Authentication Framework

- Actors manage data, programs, computing resources in
  - Distributed & Heterogeneous environments
  - Under various secure administration
- How to use ONE system handle all of the authentication jobs?

**Data:**
- Database
- SRB
- XML
- File System
- … …

**Programs:**
- Command Line
- MPI Parallel
- Online CGI
- Web Service
- Grid Application

**Resources:**
- Mobil Device
- Laptop
- Desktop
- Cluster
- Supercomputer
- Grid

**Job Management:**
- OS
- Gondor
- PBS qsub
- GRAM
- Web Portal
- … …
Advantages of Scientific Workflow Systems

- Formalization of the scientific process
- Easy to share, adapt and reuse
  - Deployable, customizable, extensible
- Management of complexity and usability
  - Support for hierarchical composition
  - Interfaces to different technologies from a unified interface
  - Can be annotated with domain-knowledge
- Tracking provenance of the data and processes
  - Keep the association of results to processes
  - Make it easier to validate/regenerate results and processes
  - Enable comparison between different workflow versions
- Execution monitoring and fault tolerance
- Interaction with multiple tools and resources at once
- Parallelism...
Evolving Challenges For Scientific Workflows

- **Access to** heterogeneous **data and computational resources and link to different** domain knowledge
- **Interface to** multiple **analysis tools and workflow systems**
  - One size doesn’t fit all!
- **Support computational experiment creation, execution, sharing, reuse and provenance**
  - **Manage complexity, user and process** interactivity
  - **Extensions for adaptive and dynamic workflows**
- **Track provenance of workflow design (= evolution), execution, and intermediate and final results’**
- Efficient failure recovery and smart re-runs
- **Support various file and process** transport mechanisms
  - Main memory, Java shared file system, ...
Evolving Challenges For Scientific Workflows

- **Support the full scientific process**
  - Use and control instruments, networks and observatories in observing steps
  - Scientifically and statistically analyze and control the data collected by the observing steps
  - Set up simulations as testbeds for possible observatories

- **Come up with efficient and intuitive workflow deployment methods**

- **Do all these in a secure and usable way!!!**
Overview

• Motivation
• Introduction
• Architecture
• Existing Workflows
• Hands on
• Wrap-up
Topics

• Example - Terascale Supernova Initiative Workflow
• Workflow anatomy
  • Problem decomposition and abstraction
  • Processes, data and behaviors
  • Scripts
  • Data Modeling, Data Flows
  • GUI layer
  • Orchestration
  • Communications
  • Meta-data collection (provenance)
  • Error and interrupt management (fault-tolerance)
  • Virtualization
• Summary
• Handouts
Example – Computational Astrophysics

• **Circumstellar Gas-Dynamics.** The numerical hydrodynamical code VH-1 is used on supercomputers, to study a vast array of objects observed by astronomers both from ground-based observatories and from orbiting satellites.

• The two primary subjects under investigation are interacting binary stars - including normal stars like the Algol binary, and compact object systems like the high mass X-ray binary SMC X-1 - and supernova remnants - from very young, like SNR 1987a, to older remnants like the Cygnus Loop.

• Other astrophysical processes of current interest include radiatively driven winds from hot stars, the interaction of stellar winds with the interstellar medium, the stability of radiative shockwaves, the propagation of jets from young stellar objects, and the formation of globular clusters.
The Big Picture: Supporting the Scientist

From “Napkin Drawings”…

… to Executable Workflows

Conceptual SWF

Executable SWF

A workflow from the Terascale Supernova Initiative SciDAC, DOE
Scientific Workflow Automation (e.g., Astrophysics)

In conjunction with John Blondin, NC State University
Automate data acquisition, transfer and visualization of a large-scale simulation at ORNL

- **Input Data**
  - VH1+

- **Output**
  - ~500 files (<50+GB each)

- **Aggregated to**
  - ~500x500 files

- **HPSS archive**
  - Logistic Network
  - Depot
  - Local Mass Storage 14+TB

- **Provenance**
  - Local 44 Proc. Data Cluster
    - data sits on local nodes for weeks

- **Web**
  - Viz Software

- **Viz Wall**
  - Viz Client
Demo Runs and Workflow Exploration with Discussion of Components, Processes, Behaviors, Data Models, Issues, Solutions …
Some Discussion Related Slides
Discussion: Workflow
SPA: Scientific Process Automation

The Scientific Process Automation project enables scientists to easily create exploratory data flows (scientific workflows) from local and distributed components (e.g. web services).

Below is a sampling of workflows powered by SPA:

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Promoter Identification</td>
<td>A production workflow that generates a list of promoter binding sites for</td>
</tr>
<tr>
<td>Workflow</td>
<td>each of a given set gene ID members.</td>
</tr>
</tbody>
</table>

SPA is a collaboration between Lawrence Livermore National Laboratory, San Diego Supercomputer Center at UCSD, and North Carolina State University. It is based upon the Proteum II heterogeneous modeling and design framework. It is also an active participant in the Kepler project.
Kepler (v1)

This workflow automates job submission, large volume data transfer, and visualization for a large scale simulation (Astrophysics/Blondin Workflow)

Submit Job to Supercomputer via SSH; Run simulation in parallel; Prepare results for bulk transfer.

Transfer results from supercomputer to post-processing.

Collect Connection Permissions

Post-processing
- Slicing and dicing for processing by visualization cluster

Visualization Processing
- Use Ensit to generate images of the simulation

Visualization Display
- Display the image/animation
A TSI Workflow (in Ptolemy II framework)

This workflow automates job submission, large volume data transfer, and visualization for a large scale simulation (Blindin Workflow).

Right click and Configure to your Username and Password used for connection to servers. We are using SSH and SSL to provide secure communication.

Submit Job to Supercomputer via SSH

Transfer results from supercomputer to post processing.

Post-processing

Slicing and dicing for processing by visualization cluster

Use EnSight to generate images of the simulation

Invoke Sibel File Transfer

(Sibel) For each file, start “Sibel sendfile” server on the source platform, then start “Sibel receive” client on the destination platform.

Invoke Sibel SendFile

N.B. getFilesName retrieves the list of files to be transferred. Loop to transfer individual files.

Receive File List

Invoke Sibel Receive File

(Sibel) After receiving all files, start “Sibel receive” server on the destination platform, then start “Sibel sendfile” client on the source platform.
Discussion: Abstraction
**Scientific Workflow Automation (Abstract Ops)**

Needed:
A comprehensive, on-demand, end-to-end, data and workflow management. An integrated network-based framework that is functional, dependable, fault-tolerant, and supports data and process provenance.
Workflow - Abstraction

Model
Merge
Backup
Move
Split
Viz

Parallel Computation

Mass Storage
Fiber C. or Local NFS

Data Mover Channel
(e.g. LORS, SABUL, FC over SONET)

Head Node Services

Head Node Services

Web Services

Web or Client GUI

Construct
Orchestrate
Monitor/Steer
Change
Stop/Start

To VizWall
Parallel Visualization

Control
A Key Issue

- Very important to distinguish between a custom-made workflow solution and a more canonical set of operations, methods, data models and solutions that can be composed into a scientific workflow.
- Complexity, skill level needed to implement, usability, maintainability, “standardization”
  - e.g., sort, uniq, grep, ftp, ssh on unix boxes
  - SAS (that can do sorting), home-made sort,
  - LORS, SABUL, bbcp (free, but not standard), etc.
Discussion: Similarities
TSI Workflow I

In conjunction with Doug Swesty and Eric Myra, Stony Brook

Automate the transfer of large-scale simulation data between NERSC and Stony Brook
Fusion Simulation Project Workflow Pilot

*In conjunction with Scott Klasky, CESP, PPPL, ORNL Automation of simulation, transfer and analytics of FSP*
ChemInformatics Workflow
In conjunction with Resurgence Project
Automate the management and submission of jobs
SCIRun and Kepler Dataflow Integration
Incorporate SCIRun computation and visualization with the SPA workflow engine
DOE Scientific Data Management Center – Scientific Process Automation

Scientific Workflow Automation (Abstract Ops)

Needed:
A comprehensive, on-demand, *end-to-end*, data and workflow management. An integrated network-based framework that is functional, dependable, fault-tolerant, and supports data and process provenance.

---

**Data/Science/Code**
- Model
- Archive/Backup
- Slice&Dice
- Analyze (e.g.Viz)
- Merge
- Move
- Split ...

---

**Workflow**
- Construct
- Orchestrate
- Monitor/Steer
- Provenance
- Change
- Stop/Start ...

---

**RT**
- Mont.
- O/P
- Move
- Post-A

---

**Archive. Provenance**
- RT Diag
- RT Mont.

---

**Move**
- Prep
- Move
- Exec
- O/P
- Move
Scientific Workflow Automation (Major Resource Sinks)

Shift scientist’s efforts away from unnecessary data and resource management, workflow orchestration and application development to scientific research and discovery.
Discussion: Architecture and Components

Architecture
Parameterization
Data models and flows
Scripts
Communications
Orchestration
Meta-data collection (provenance)
Error and interrupt management (fault-tolerance)
Virtualization
Scientific Process Automation

Process/Data/Behavior (Workflow)
Construction, Control, Monitoring and Steering
(End-User Access)

Gateways, Brokers, Registries,
Persistence DBs, Proxies and Services
(Middleware)

Synchronous or Asynchronous Resource
Management and Data Movement
(“Heavy lifting”, actions, scripts, standard ops)

Hardware, OS, and MSS)
**Kepler System Architecture**

[Diagram showing the system architecture of Kepler, including layers for Authentication, GUI, Kepler Object Manager, SMS, Type System Ext, Smart Re-run Failure Recovery, Provenance Framework, Kepler Core Extensions, and Ptolemy.]
Elements, Issues, Solutions, ...

- Components
- Coupling (loose, tight, very tight, code-level) and Granularity (fine, medium?, coarse)
- Communication Methods (e.g., ssh tunnels, xmprpc, snmp, web/grid services, etc.)
- Scripts
- Direct and Indirect Data Flows (functionality, throughput, delays, other QoS parameters)
- Data models
- End-to-end performance (including fault-tolerance, recovery, time-outs, etc.)
- Level of abstraction
- Workflow description language(s) and exchange issues – interoperability
- “Standard” scientific computing “W/F functions”
- Other ...
Component-Based System Engineering*

- The basis is the **Component**
- Components can be **assembled** according to the rules specified by the **component model**
- Components are assembled through their **interfaces**
- A **Component Composition** is the process of assembling components to form an assembly, a larger component or an application
- Component are performing in the context of a **component framework**
- All parts conform to the **component model**
- A **component technology** is a concrete implementation of a component model

**Reusable, Substitutable, Extensible, Composable, Networked**

*Functional equivalence*

A Data Model

```
runinfo
  uid
  runid
taskname
time
info
note

1

1

M

fileinfo
  uid
  fileid
filename
type
path
size
lasttime
machines
parents
notes

M

histinfo
  uid
  runid
hostname
content
```
Example - Partial GTC & S3D Schemas
Kepler Provenance Framework

- **Provenance**: the history of execution and conditions applied to the workflow run
- **Why do we need provenance?**
  - Recreate results and rebuild workflows using the evolution information
  - Associate the workflow with the results it produced
  - Create links between generated data in different runs, and compare different runs
  - Checkpoint a workflow and Recover from a system failure
  - Debug and explain results (via lineage tracing, …)
  - Smart Reruns
- **Types of Provenance Information:**
  - Data provenance
    - Intermediate and end results including files and db references
  - Process provenance
    - Keep the wf definition with data and parameters used in the run
  - Error and execution logs
  - Workflow design provenance
Example of Runtime Data Collection

- Each run of the workflow is associated with a runid. The info is organized around it.

**Workflow Execution Index**

<table>
<thead>
<tr>
<th>uid</th>
<th>runid</th>
<th>Log</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>blondin</td>
<td>10092</td>
<td>Detail</td>
<td>File Info</td>
</tr>
<tr>
<td>blondin</td>
<td>10091</td>
<td>Detail</td>
<td>File Info</td>
</tr>
<tr>
<td>blondin</td>
<td>10090</td>
<td>Detail</td>
<td>File Info</td>
</tr>
<tr>
<td>blondin</td>
<td>10089</td>
<td>Detail</td>
<td>File Info</td>
</tr>
<tr>
<td>blondin</td>
<td>10088</td>
<td>Detail</td>
<td>File Info</td>
</tr>
<tr>
<td>blondin</td>
<td>10085</td>
<td>Detail</td>
<td>File Info</td>
</tr>
</tbody>
</table>

**Workflow Execution Logs**

<table>
<thead>
<tr>
<th>uid</th>
<th>runid</th>
<th>taskname</th>
<th>time</th>
<th>info</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>blondin</td>
<td>10092</td>
<td>checkRunning</td>
<td>05/09/29 12:40:42</td>
<td>checking</td>
<td>not submitted</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>submitJob</td>
<td>05/09/29 12:40:44</td>
<td>started</td>
<td>about to submit</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>submitJob</td>
<td>05/09/29 12:40:47</td>
<td>ended</td>
<td>job submitted</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>checkWait</td>
<td>05/09/29 14:47:18</td>
<td>checking</td>
<td>running</td>
</tr>
</tbody>
</table>

**File Tracking Info**

<table>
<thead>
<tr>
<th>uid</th>
<th>runid</th>
<th>filename</th>
<th>type</th>
<th>path</th>
<th>size</th>
<th>lasttime</th>
<th>machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>blondin</td>
<td>10092</td>
<td>Rov01W1067.nc</td>
<td>regular</td>
<td>/scratch/scrbig15/blondin/Rotation/Rot009/01W1067/01W1067</td>
<td>2048009848</td>
<td>n/a</td>
<td>phoenix</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>Rov01V1067.nc</td>
<td>regular</td>
<td>/scratch/scrbig15/blondin/Rotation/Rot009/01V1067/01V1067</td>
<td>2048009848</td>
<td>n/a</td>
<td>phoenix</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>Rov01U1067.nc</td>
<td>regular</td>
<td>/scratch/scrbig15/blondin/Rotation/Rot009/01U1067/01U1067</td>
<td>2048009848</td>
<td>n/a</td>
<td>phoenix</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>Rov01P1067.nc</td>
<td>regular</td>
<td>/scratch/scrbig15/blondin/Rotation/Rot009/01P1067/01P1067</td>
<td>2048009844</td>
<td>n/a</td>
<td>phoenix</td>
</tr>
<tr>
<td>blondin</td>
<td>10092</td>
<td>Rov01D1067.nc</td>
<td>regular</td>
<td>/scratch/scrbig15/blondin/Rotation/Rot009/01D1067/01D1067</td>
<td>2048009844</td>
<td>n/a</td>
<td>phoenix</td>
</tr>
</tbody>
</table>
An Implementation

Manticore

Mysql DB

XMLRPC (Zope)

HTTP (Apache)

XMLRPC

Super Computer

SSH

Orbitty
(Python adapter)

Scientists
Discussion: Fault-Tolerance
Probability of failure is that of least one service failing.
E.g., for 3 serial service, “average” \( R = (1-0.045)^3 = 0.871 \), and in the “heavy load” case \( R = (0.8)^3 = 0.512 \)
But \( (0.985)^3 \) would yield a collective cca 0.95+
Single Service as a Parallel System

- Considerable improvement using redundancy – for m=3, service failure probability is now about 0.000091 instead of 0.045.
- Of course, redundant services must not exhibit significantly \textit{correlated} failures, either due to their (co-)location, or for algorithmic or other reasons, or the approach may not work.
Collective Behavior

- For example, with $p=0.045$, $n=3$ and $m=3$, workflow failure probability is reduced from 0.129 to less than $10^3$.
- Even $m=2$ helps a lot.
Implementation

- Problem occurs frequently in serial workflows
- Hence, mirroring of services (very common, failover is often manual and at user level), e.g.,
  - FTP
  - Web pages (http)
  - Less frequently of SOAP based services

- Issues
  - Non-standard listing of alternatives
  - Often no heart-beat
  - Often no information about reliability of individual service
  - Complex services are only near-equivalent (e.g., desire to differentiate from competition, different implementation, etc.)
Fault-Tolerant Shell

- From Cooperative Computing Lab at the University of Notre Dame ([http://www.cse.nd.edu/~ccl/software/ftsh/](http://www.cse.nd.edu/~ccl/software/ftsh/))

- ftsh - The Fault Tolerant Shell The Fault-Tolerant Shell (ftsh) is a small language for system integration that makes failures a first class concept. Intended to deal with failures that recover. For example:

  - **try for 30 minutes**
    - cd /tmp
    - rm -f data
    - for any host in xxx yyy zzz
      - wget http://${host}/fresh.data data
    - end
  - end

- If any element of the script fails, all running process trees are reliably cleaned up, and the block is tried again with an exponential backoff. You might think of this as exception handling for scripts.
FT Service - XML

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<services>
  <service-name = "Genbank">
    <bindings>
      <binding-name = "main">
        <url>http://manticore.csc.ncsu.edu/</url>
        <namespace>urn:spa.service.Genbank</namespace>
        <method>service</method>
      </binding-name>
      <binding-name = "backup1">
        <url>http://backup1.csc.ncsu.edu/</url>
        <namespace>urn:spa.service.Genbank</namespace>
        <method>service</method>
      </binding-name>
      <binding-name = "backup2">
        <url>http://backup2.csc.ncsu.edu/</url>
        <namespace>urn:spa.service.Genbank</namespace>
        <method>service</method>
      </binding-name>
    </bindings>
  </service-name>
</services>
```
Uses registered local and remote services to construct new services and/or workflows. Saves product on a server and registers it.

**Workflow Composer**

- Workflow Composer Tools

**Registries (e.g., UDDI) and Context Gateways**

- e.g., a Web Service Registry

**Users**

- Workflow Agent

“Dials” needed services/workflows and executes/runs through the services/workflow, delivers output to user.

**Global Architecture**

Services-Based Scientific Workflow System

**SOAP Service Gateways and other Service Agents**

- NCBI Services
- LLNL Services
- SDSC Service
- NCSU Services
- ORNL Services
SPA/Kepler - Web-Service Actor

- The Web Service actor’s function is to invoke web services.
- Example actor invokes a service that queries the Genbank database. This actor takes 6 parameters:
  - NameSpace: specifies the namespace of the web service
  - LocationUrl: specifies the URL where the web service is found
  - Username: specifies the username required to access the service
  - Password: specifies the password required to access the service
  - Method: specifies which method to invoke
  - ParameterName: specifies the input passed to the web service
- User has to know and explicitly enter the location and namespace of the service that needs to be invoked. The service will then be invoked with the given username, password, method and input. After it is done processing, the results are forwarded to the next actor in the workflow.
Fault-Tolerant Web-Service Actor

- In order to provide Fault Tolerance, the actor needs to know how to handle a missing service. UDDI repository can be used to store the location of the various services, and backup copies of those services.
- An example of the way services were stored in the repository:
  - SDM SPA Genbank 01
    - Description: Main Genbank Service
  - SDM SPA Genbank 02
    - Description: Identical Backup Genbank Service
- The user ONLY needs to specify keyword. The actor retrieves the matching services, and stores the service locations in a local data structure (to avoid performing another search in the case the first service on the list fails). The actor then invokes the services in succession. If complete failure occurs, actor sends an error message.
Promoter Identification Workflow

*In conjunction with Matt Coleman, LLNL*

Automate the analysis of gene expression data using a combination of web services and local analysis programs.
Some Limitations

- Issue: web service is operational but not behaving properly, for example not returning correct results, or taking a long time to process requests.
  - Run-time validation of the results before proceeding with the rest of the workflow. e.g., by submitting a sample request and comparing the result with a saved result before submitting the rest of the results.
- Case where one of the services returned in the search isn’t what we’re looking for may be avoided by providing complex search phrase
- Another approach is to have a sample input and output, then invoke the service with that sample input and compare the outputs, to make sure that the service in question is indeed what we’re looking for.
- Consensus voting, fuzzy voting, and other more complex FTS solutions are an option.
To effectively deliver on-demand computing services that are maintainable, scalable, and customizable it is essential that the tiers of virtualization are separated but can be coupled.
Summary

- Scientific Workflows - a series of structured activities and computations that arise in scientific problem-solving.
- Needed: a comprehensive, end-to-end, data and workflow management for scientific workflows. Integrated network-based framework that is functional, dependable, fault-tolerant, and supports data and process provenance.
- Shift scientists’ efforts away from resource management and application development to scientific research and discovery.
- Long history starting with numerical libraries, through tightly-coupled local and distributed problem solving environments, to a combination of end-to-end synchronous and asynchronous network-based/assisted/integrated computational, storage, and access (e.g., via web) resources.
- One such framework is the Ptolemy II based environment called Kepler.
- Department of Energy Scientific Data Management Center is participating in development of Kepler and its evaluation in a number of high-end scientific problem solving settings including bioinformatics, astrophysics and fusion domains.
“If there is time” topic

• Comparison of Kepler with other workflow support systems
  • Capabilities
  • Interoperability
  • Usage community
  • Other ...
Other SW/F Support Engines

- Classical Problem Solving Environments (tightly coupled, distributed) – been around for 20+ years, but tend to be domain specific and hand-crafted
- Open source initiatives (a number of those), many are services-based
- Proprietary solutions (including Microsoft’s Windows Workflow Foundation)

Some (Open Source) Solutions

- BioPipe
- BizTalk
- BPWS4J
- DAGMan
- GridAnt
- Grid Job Handler
- GRMS (GridLab Resource Management System)
- GWFE (Gridbus Workflow Engine)
- GWES (Grid Workflow Execution Service)
- IT Innovation Enactment Engine
- JIGSA
- Kepler
- Karajan
- OSWorkflow
- Pegasus (uses DAGMan)
- ScyFLOW
- SDSC Matrix
- SHOP2
- Taverna
- Triana
- wftk
- YAWL Engine
- WebAndFlo
- WFEE
- Etc.

http://www.gridworkflow.org/snips/gridworkflow/space/Workflow+Engines
http://www.extreme.indiana.edu/swf-survey/
Overview

- Motivation
- Introduction
- Architecture
- Existing Workflows
- Hands on
- Wrap-up discussion
Handouts

- Expectation is that attendees have a laptop with them which is networked. Hands-on remote account, access and help information.
- CD with Kepler and demos
  - Packages with any additional slides that may be used during the tutorial
  - A collection of relevant reports/papers/notes
- Contact information
Installing and Running Kepler

- In the lab:
  - We will use Virtual Computing Laboratory (VCL) to access Kepler
  - To start:
    - Double click on the VCL logo in the tutorial CD
    - Use the connection information in the handouts

- Installing Kepler from the tutorial CDs:
  - Read the README.txt
  - Double click on the Kepler version

- [http://www.kepler-project.org](http://www.kepler-project.org)
  - There’s a link to the latest release on the Kepler website
  - To install future releases!!!

- Compatible with all platforms
Opening and Running a Workflow

- Start Kepler from VCL
  - As explained in the paper handouts

- Open the "HelloWorld.xm1" under the "demos/getting-started" directory in your local Kepler folder

- Two options to run a workflow:
  - PLAY BUTTON in the toolbar
  - RUNTIME WINDOW from the run menu
Modifying an Existing Workflow and Saving It

• GOAL: Modify the HelloWorld workflow to display a parameter-based message

• Step by step instructions:
  • Open the HelloWorld workflow as before
  • From actors search tab, search for Parameter
  • Drag and drop the parameter to the workflow canvas on the right
  • Double click the parameter and type your name
  • Right click the parameter and select ‘Customize Name’, type in ‘name’.
  • Double click the Constant actor and type the following:
    • “Hello “ + name
  • Save
  • Run the workflow
Creating a HelloWorld! Workflow

- Open a new blank workflow canvas
  - From toolbar: File -> New Workflow -> Blank
- In the Components tab, search for “Constant” and select the Constant actor.
- Drag the Constant actor onto the Workflow canvas
- Configure the Constant actor
  - right-click the actor and selecting Configure Actor from the menu
  - Or, double click the actor
- Type “Hello World” in the value field and click Commit
- In the Components and Data Access area, search for “Display” and select the Display actor found under “Textual Output.”
- Drag the Display actor to the Workflow canvas.
- Connect the output port of the Constant actor to the input port of the Display actor.
- In the Components and Data Access area, select the Components tab, then navigate to the “/Components/Director/” directory.
- Drag the SDF Director to the top of the Workflow canvas.
- Run the model
Drive the point home…

• **GOAL**: Create a workflow to execute a simulation job, monitor it and post-process the results

• **Main steps**:
  - Execute simulation
    - (Create, submit, manage, monitor status)
  - Move files
  - Archive and visualize results

• **Put all these pieces together to create a hierarchical workflow**
For simplicity, everything in the back-end is simulated: Supercomputer, Staging Machines, Analytics, scp and bbcp, Etc.
Overview

• Motivation
• Introduction
• Architecture
• Existing Workflows
• Hands on
• Wrap-up discussion
The end...
...questions?
Thank you!