Experiences with
OGCE Workflow Suite Empowering
LEAD, GridChem, BioVLAB, UltraScan Science Gateways

Suresh Marru, Raminder Singh, Marlon Pierce
Pervasive Technology Institute
Indiana University
Acknowledgements

• OGCE Workflow Suite is inherited from a decade of work by Extreme Computing Lab and many PhD’s under the guidance of Dr. Dennis Gannon.
• Special Thanks to Dennis Gannon, Beth Plale, Kelvin Droegemeier and the LEAD TEAM.
• Includes contributions from: Alek Slominski, Satoshi Shirasuna, Gopi Kandaswamy, Yi Huang, Yogesh Simmhan, Wei Lu, Srinath Perera, Chathura Herath, Lavanya Ramakrishnan, Eran Chinthaka, Thilina Gunarathne and the Lanka Software Foundation.
Nancy’s Outline

• Choices made in evaluating and selecting a workflow tool
• Description of how these tools are implemented in a gateway
• Presentation of the user interface
• Description of tasks performed behind the scenes
• Discussion of experience with tools - pros and cons
High Level LEAD Architecture

Workflow graph

User Portal

Portal server

Data Catalog service

MyLEAD Agent service

MyLEAD User Metadata catalog

Workflow Engine

Fault Tolerance & scheduler

Event Notification Bus

Application services

Compute Engine

Data Storage

Workflow graph

Providence Collection service

Data Management Service

Event Notification Bus
Scientists and Educational Interactions

Lowering the barrier for using complex end-to-end technologies

Democratize
Empower
Facilitate

Application Developers (typically Scientists)
Researchers

End Users

OGCE
Open Grid Computing Environments

TeraGrid
Analyze & Predict

Discover & Visualize

Research & Reproducibility

Education & Outreach
LEAD Workflow Requirements

• Run jobs on-demand on TeraGrid.
• Deadline driven workflows (severe weather tracking)
• Users ranging from 8th grade students to seasoned researchers.
• Run jobs on Multiple TeraGrid resources to increase turn-around time.
• Must be able to integrate to Portal with very user friendly web interface.
# Workflow Survey in 2003

(https://www.extreme.indiana.edu/swf-survey/)

## List of Scientific Workflows (Alphabetical)

<table>
<thead>
<tr>
<th>Name</th>
<th>Status, availability</th>
<th>Prerequisites</th>
<th>Tooling</th>
<th>Workflow Standards</th>
<th>Grid integration</th>
<th>Partial integration</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astaron</td>
<td>NA</td>
<td>Java 1.1, 1.2, CoG</td>
<td>Java GUI</td>
<td>Custom XML language (ADWL)</td>
<td>GT2, CoG</td>
<td>NA</td>
<td>A Java runtime for ADF, that has resource broker, resource monitoring, (meta)scheduler etc.</td>
</tr>
<tr>
<td>ADWL</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>ADWL is an abstract Grid workflow language for describing workflows (graphs with loops) at a high level of abstraction used in Astaron</td>
</tr>
<tr>
<td>BioOpen</td>
<td>Cerberus</td>
<td>Conda DAGMan</td>
<td>?</td>
<td>NA</td>
<td>Condor</td>
<td>?</td>
<td>Built on top of Condor, workflow as an add-on to provide on-demand data generation (&quot;sandbox&quot;)</td>
</tr>
<tr>
<td>D'Agostino</td>
<td>Mozilla License</td>
<td>JDK 1.1.1</td>
<td>Java GUI</td>
<td>NA</td>
<td>?</td>
<td>?</td>
<td>Modules compiled into dataflow optimized for data mining and B2C2 (Knowledge Discovery in Databases)</td>
</tr>
<tr>
<td>DAGMan</td>
<td>Part of Condor, source code under GPL</td>
<td>Condor</td>
<td>Integrated with Condor command line tools</td>
<td>NA</td>
<td>GT2 (Condor-G)</td>
<td>Under work</td>
<td>Part of Condor, very well integrated</td>
</tr>
<tr>
<td>EDIF</td>
<td>DiscoverNet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EUROGRID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EWS</td>
<td>GRIDS superpanel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GridAnt</td>
<td>Soon?</td>
<td>Java JRE 1.4, CoG</td>
<td>Graphical workflow visualizer (AXL)</td>
<td>Built on top of Apache ANT</td>
<td>via CoG</td>
<td>NA</td>
<td>Merged ANL GridAnt and NCSA OGRE</td>
</tr>
<tr>
<td>GridPort</td>
<td>3.0 alpha</td>
<td>JRE, CoG, JBoss</td>
<td>NA</td>
<td>NA</td>
<td>via CoG</td>
<td>NMI OGCE, Jetison, Hotpage 3.0</td>
<td>Non-standard description of job sequences</td>
</tr>
<tr>
<td>Grid Genesis</td>
<td>1.0 (to be released under open source license)</td>
<td>Java JRE 1.4</td>
<td>Built on top of Mozilla using Java GUI</td>
<td>Internal DMG is transformed into new scripting language called XPL (XML based)</td>
<td>GT3</td>
<td>Under investigation</td>
<td>Main focus on making Drag and drop GUI</td>
</tr>
<tr>
<td>Grid Service Broker</td>
<td>Part of the Genesis Broker, code available under GPL</td>
<td>Java, CoG</td>
<td>Command line, Java based API</td>
<td>GT2, Altair</td>
<td>GT2, Altair</td>
<td>Yes, with G-monitor that supports multiple devices (?)</td>
<td>Supports parameter computing for compute and data grid applications, being extended to support advanced workflows,</td>
</tr>
<tr>
<td>GRMS</td>
<td>CSL (new paper)</td>
<td>Java, CoG</td>
<td>command line and GridSphere Web portal</td>
<td>DAM/PeerNet</td>
<td>Pre-WS GT</td>
<td>yes (GridSphere)</td>
<td>GRMS is a persistent GSI enabled Web Service in GridLab project</td>
</tr>
<tr>
<td>Plenary for Eclipse</td>
<td>ISSD</td>
<td>Java 1.4 (or above)</td>
<td>Eclipse 3.1</td>
<td>Directed Cycle Labelled Graphs, stored in custom XML format (Graph Flow Diagrams) - supports cycles</td>
<td>WSFL, WSFDE, G14, SSH, Condor</td>
<td>Built-in monitoring tools (Eclipse 40P and Web based)</td>
<td>Grid workflows fully integrated with the Eclipse user experience; Extensible with your own plugins to call any kind of service</td>
</tr>
<tr>
<td>Plenary for Eclipse</td>
<td>ISSD (open source license - certified?)</td>
<td>Java</td>
<td>Java GUI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plenary for Eclipse</td>
<td>ISSD</td>
<td>Java 1.2</td>
<td>Java GUI (Virtual GSI Job Authoring)</td>
<td>GAADL is based on POI Nets</td>
<td>GT2</td>
<td>Job Builder</td>
<td>Allows queue retirement during execution (such as add file transfer node)</td>
</tr>
<tr>
<td>Plenary for Eclipse</td>
<td>Summer/WF?</td>
<td>CoG2 GUI</td>
<td>Visualiser2</td>
<td>NA</td>
<td>GT2, GT2A, GT2E02, SSH, Condor</td>
<td>GT2, Altair</td>
<td>NMI OGCE.org</td>
</tr>
</tbody>
</table>

---

**OGCE**

Open Grid Computing Environments

**TeraGrid**
OGCE Workflow Suite

• Generic Service Toolkit
  – Tool to wrap command-line applications as web services
  – Handles file staging & job submission and monitoring
  – Extensible runtime for security, resource brokering & urgent computing
  – Generic Factory service for on-demand creation of application services

• XRegistry
  – Information repository for the OGCE workflow suite
  – Register, search, retrieve & share XML documents
  – User & hierarchical group based authorization

• XBaya
  – GUI based tool to compose & monitor workflows
  – Extensible support for compiler plug-ins like BPEL, Jython, SCUFL
  – Dynamic Workflow Execution support to start, pause, resume, rewound of workflow executions

• Apache ODE Scientific Workflow Extensions
  – XBaya GUI integration for BPEL Generation
  – Asynchronous support for long running workflows
  – Instrumented with fine grained monitoring

• Eventing System
  – Supports both WS-Eventing and WS-Notification Standards
  – Very scalable
  – Persistent Message Box for clients behind firewalls and with intermittent network glitches.
Generic Application Service Factory
WS-BPEL

• Business Process Execution Language for Web Services (WS-BPEL)
  – De-facto standard for specifying web service based business processes and service compositions

• Basic activities
  – Invoke, Receive, Assign..

• Structured activities
  – Sequence, Flow, ForEach,..
Workflow Composition, Execution & Monitoring

XBaya enables users to construct, share, execute and monitor sequence of tasks executing on their local workstations to high-end compute resources.
“A significant need exists in many disciplines for long-term, distributed, and stable data and metadata repositories”

- NSF Blue-Ribbon Advisory Panel on Cyberinfrastructure

“Metadata is key to being able to share results”

- UK e-Science Core Programme Study

More Info: Scott Jensen, Beth Plale
Service Monitoring via Events

• The service output is a stream of events
  – I am running your request
  – I have started to move your input files.
  – I have all the files
  – I am running your application.
  – The application is finished
  – I am moving the output to you file space
  – I am done.

• These are automatically generated by the service using a distributed event system (WS-Eventing / WS-Notification)
  – Topic based pub-sub system with a well known “channel”.

notification Channel

Subscribe Topic=x

Application Service Instance

publisher

generates event

listener

Application Service Instance

publisher

subscribe topic=x

Application Service Instance

publisher

subscribe topic=x

Application Service Instance

publisher

subscribe topic=x

Application Service Instance

publisher
WxChallenge: The National Collegiate Weather Forecast Contest

• Hundreds of graduate and undergraduate students nationwide make daily forecasts for selected US cities
• LEAD was introduced on a trial basis so students could run their own WRF forecasts with fine-scale grids + fine-scale data
  – 7-week pilot study funded by LEAD core
  – 75 students from 10 institutions + faculty sponsors
  – 279 on-demand forecasts made at NCSA, 0.6 TB output produced
  – More than 160 processors reserved on Tungsten 5 days/week, 10 am – 8 pm EDT
  – 78% of workflows completed, subsequent problems fixed
• Significant hands-on experience with models → opportunity for learning about CFD, physics, computing
• Decision Science Opportunity: How do students decide how, when to run model?
• Huge (but welcome) challenge for TeraGrid
• Good way to identify strengths and weaknesses in ALL components of LEAD
• Formal evaluation conducted
Lessons Learned From LEAD

• Large number of users invoking lot of small applications create a surges in load in middleware.
  – Build Reliability into Workflow Infrastructure

• Troubleshooting large scale distributed infrastructure needs coordinated debugging involving multiple experts.
  – Ingest as much as possible fault Tolerance.

• Scientists need Flexibility but too much Flexibility is called Confusion.
Workflow Architecture
GPEL

• Grid Process Execution Language
  – BPEL4WS based home grown research workflow engine
  – Supports a subset of BPEL4WS 1.1
  – One of the very early adaptations of BPEL efforts
• Specifically designed for eScience Usage
  – Long running workflow support
  – Decoupled client
Benefits of Porting to Apache ODE

- **WS-BPEL 2.0 features**
- **Improved Scalability & performance**
- **Sustainability**
  - Well supported run time with minimal custom changes
- **Minimize changes to legacy components**
- **Portability & avoid lock in**
  - Adhering to widely used open standards
  - Avoid using runtime specific
• Challenges:
  – Some apps have rich Client Gui’s, a challenge with asynchronous long running workflows
  – Workflow Verification Service
  – Parametric sweep scheduling, monitoring iteration steps, graphical composition
  – Human Interaction into Workflows
UltraScan Advanced Support

• Enhance Job Management
• Integrate Information Services
• Improve Fault Tolerance
• Integrate with Unicore and other European Grids
• Enable Workflows for Future
Packaged, Downloadable Software

• http://www.collab-ogce.org/ogce/index.php/Main_Page
Live Demo & Questions?
XBaya Usage Flow

1. Scientist/Application provider registers application description with Registry Service.
2. Workflow Author constructs the workflow with multiple wrapped application services.
3. Workflow is compiled and deployed to the ODE workflow Engine.
4. Workflow inputs are captured by XBaya and workflow is launched to ODE.
5. Workflow system and possibly some services publish notifications to the Message bus reporting the progress of the workflow.
6. XBaya monitoring system listens to notifications and color the workflow components to present workflow progress.