Kepler: An Extensible System for Design and Execution of Scientific Workflows

Ilkay Altintas[1], Chad Berkley[2], Efrat Jaeger[1], Matthew Jones[2], Bertram Ludäscher[1], Steve Mock[1]
[1]San Diego Supercomputer Center (SDSC), University of California San Diego
[2]National Center for Ecological Analysis and Synthesis (NCEAS), University of California Santa Barbara
{berkley, jones}@nceas.ucsb.edu, {altintas, efrat, ludaesch, mock}@sdsc.edu

Abstract. *We present the Kepler system for modeling and enacting scientific workflows. Kepler is unique in that it seamlessly combines high-level workflow design with execution and runtime interaction, access to local and remote data, and local and remote service invocation. We give an overview of the system, highlight some specific features, outline the planned (hands-on) demonstration, and report on the implementation status and next steps.*

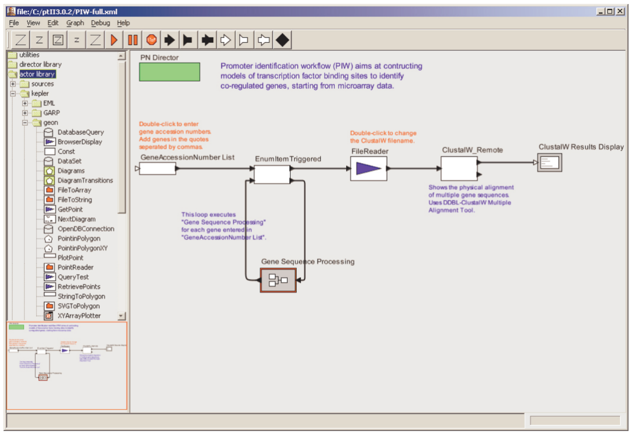
1. Scientific Workflows and Kepler Features

Progress in science depends on the quantitative and repeatable analysis of data from a variety of sources. Most scientists conduct analyses and run models in several different software and hardware environments, mentally coordinating the export and import of data from one environment to another. *Scientific workflows* can be seen as a formalization of this ad-hoc process so that scientists can design, execute, monitor, re-run, and communicate analytical procedures repeatedly and with minimal effort.

Scientific workflows (short: SWFs) are superficially similar to business process workflows but have several challenges not present in the scenario for business workflows [6]. For example, they often operate on large, complex, and heterogeneous data, can be computationally intensive, and produce complex derived data products that may be archived for use in re-parameterized runs or other workflows. Moreover, unlike business workflows, SWFs are often dataflow-oriented as witnessed by a number of recent academic systems (e.g., DiscoveryNet, Taverna, Triana, , …) and commercial systems (Pipeline-Pilot from Scitegic, Discovery-Station from Inforsense etc.). In a sense, SWFs are often closer to signal-processing and data streaming applications than they are to control-oriented business workflow applications. We have thus based the development of Kepler on the mature, dataflow-oriented Ptolemy II system (short: Ptolemy) [12]. Kepler inherits many advanced features from Ptolemy, and numerous extensions and numerous new features have been added recently for supporting scientific workflows:

- To the best of our knowledge, Kepler is the only available SWF system supporting multiple, precisely specified *execution models* via so-called directors (Figure 1). Workflows can be nested arbitrarily, and subworkflows can inherit their execution semantics from the parent, or introduce a new local director.
- Kepler's intuitive GUI for design and execution and its *actor-oriented modeling* paradigm make it a very versatile tool for SWF design, prototyping, execution, and reuse for both workflow engineers and end users.
- Kepler is *extensible* in various ways: e.g., its *web service harvester* allows the user to plug-in and execute new system components instantaneously, making it a powerful application integration tool. The underlying Ptolemy system has been designed with extensibility in mind, and new types of actors and directors (for specialized execution semantics) can be added to the system very easily.
- Kepler includes a number of specific extensions to support SWF applications, e.g., a *design actor* for rapid prototyping, *data transformation actors* (XSLT, Perl, etc.) for linking "almost but not quite fitting" web services together, *data access and query actors*, for querying database sources and other structured and annotated (EML) sources, a *browserUI* actor for injecting user control and input, as well as output of legacy applications anywhere in a workflow via the user's local web browser.
- Last not least, Kepler is a cross-project, open source activity, with an active and growing community of developers and users. The source code is freely available, and so is a preliminary packaged version for end users. The first official pre-release, Kepler v0.8 is planned for an upcoming e-Science meeting in May 2004 and will be distributed through the Kepler web site [8] and a "Kepler-to-Go" CD.

.



Promoter identification workflow (PIW) aims at contructing models of transcription factor binding sites to identify co-regulated genes, starting from microarray data.

Double-click to enter gene accession numbers. Add genes in the quotes seperated by commas.

Double-click to change the ClustalW filename.

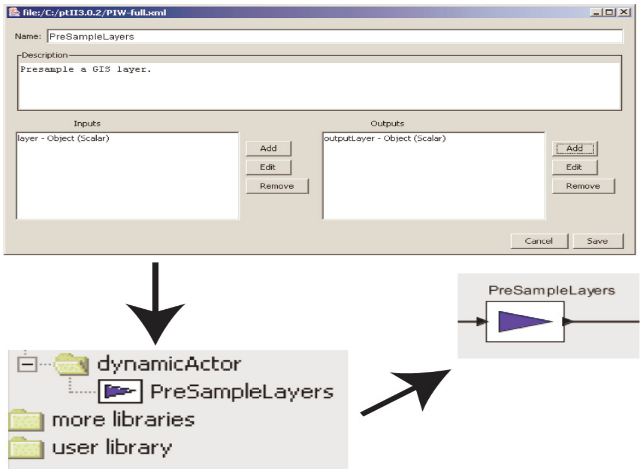
Shows the physical alignment of multiple gene sequences. Uses DDBL-ClustalW Multiple Alignment Tool.

This loop executes "Gene Sequence Processing" for each gene entered in "GeneAccessionNumber List".

Figure 1. Kepler workflow editor (Ptolemy's Vergil) showing actor libraries (left), design workspace (right), the chosen Process Network director (green), and inline comments.

2. Design and Exchange of Workflows

Using Kepler, scientists can capture workflows in a format that can easily be exchanged, archived, versioned, and executed. Kepler contains libraries of reusable modules (processing steps) called *actors*, which can act as data sources, sinks (various displays), data transformers, analytical steps (e.g., Matlab scripts), or more generally any computation step which can be invoked as a (web) service. An actor can have multiple *input* and *output ports*, through which streams of data tokens enter and exit the actor. Additionally, actors may have *parameters* to define specific behavior: e.g., for a database actor, the query to evaluate and its return type (set-at-a-time or tuple-at-a-time), for a web service actor, the method to execute, for a user-defined simulation or analytical model, the initial state and parameter settings, etc. Kepler can perform both design-time (static) and run-time (dynamic) type checking on the workflow and data. Actors run as local Java threads by default, but can spawn distributed execution via web (and soon grid) services, as well as through its foreign languages interface.



Name: PreSampleLayers

Description: Presample a GIS layer.

Inputs: layer - Object (Scalar)

Outputs: outputLayer - Object (Scalar)

Figure 2. The design actor tool creates a prototype "stub" actor, compiles it, and then adds it to the actor library, from where it can be dragged onto the workspace.

Prototyping Workflows. The system allows scientists to prototype a workflow before implementing the actual actor code needed for workflow execution. Kepler's *design actor* can be seen as a "blank slate" which prompts the scientist for critical information about an actor, e.g., the actor's name, and the number, names, and types of its input and output ports (Figure 2). Once the user has prototyped an actor in this way, a corresponding stub is compiled and added to the user's library. The user can then use this stub on the workflow canvas to prototype a workflow. In particular, since stubs are typed according to the user's information, the system can catch static type errors at the very early design stage. When trying to run a prototype workflow, an invoked stub actor simply open a dialog indicating that the workflow implementation is incomplete. When used in this way, Kepler becomes a convenient tool for designing, prototyping and documenting workflows.

Workflow Documentation and Exchange. Kepler work-flows can be exchanged in XML using Ptolemy's own Modeling Markup Language (MoML) [9]. Because of this XML serialization, workflows can be easily shared and discussed with colleagues, and the workflow description itself be used as documentation (metadata) in the users' research projects. We plan to extend (or replace, depending on standardization efforts) MoML in the future to support workflow versioning and handling of provenance information for derived data products. In this way, researchers can return to previously "check-pointed" states as needed.

3. Data Access, Querying, and Transformation

Kepler has actors to access, query, and transform data.

Database Access and Querying. An important data source in SWFs are databases. Currently, Kepler includes two database actors: The *DBConnect* actor has input parameters for the usual connection information, e.g., database URL, user-name and password, and outputs a *database connection token t* to be used by any down-stream *DBQuery* actor that needs it. The latter takes as input the database connection token *t*, an SQL query *q*, and result-type parameter indicating whether *q*'s output should be passed downstream as a single token (set-at-a-time) or as a sequence of tokens (tuple-at-a-time). The actual database communications used by DBConnect and DBQuery are through JDBC. In the near future, we will add additional database querying actors, e.g., schema-aware actors that expose individual relational attributes as output ports (similar to the EML actor described next), and GUI-enabled actors that support query formulation in a visual query-by-example (QBE) style.

EML-Source Actor. EML, the Ecological Metadata Language [7] is an XML-based metadata specification for describing ecological and biological datasets. EML contains both physical and logical information about datasets. The *EML-Source* actor (Figure 3) uses EML to ingest (i.e., access and download) heterogeneous datasets into Kepler based on physical and logical EML metadata about the source to be ingested.

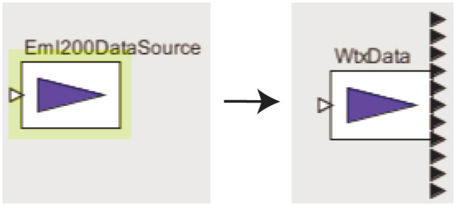


Figure 3. The EML-Source Actor (left) automatically configures itself with one output port for each logical attribute in the imported dataset (right).

Once the EML-Source actor parses the EML metadata, it uses information from the physical metadata to read the data file in its native format. It then uses the logical information to create one output port for each attribute (column) in the data file (see Figure 3). The ports are typed like other ports in Kepler, based on information contained in the EML metadata. At execution time, the datasets is emitted record-at-a-time, with each port sending the token corresponding to the port's attribute. The EML actor allows Kepler to ingest a multitude of heterogeneous data (as long as it is described in EML), making it an very flexible tool for domain scientists who often have to deal with many data and file formats.

Data Transformation Actors. When composing scientific workflows from individual actors and web services into more complex analytical pipelines, often the problem arises that two services, say A and B, cannot be linked together although the chain of actors A→B may be completely reasonable at the conceptual level. The reason for this is that when actors and web services are designed in isolation, their respective data formats (A's output and B's input) are often incompatible. In these cases, we can introduce a *data transformation actor* DT between A and B, i.e., (A→DT→B) which can transform data from A's output schema into a format which fits B's input schema. Currently Kepler includes transformation actors for a number of languages including XSLT [21] and XQuery [20] for XML data, and Perl for text-structured data. For data tokens which are sent in the native Java model, the built-in transformations actors from Ptolemy can be used.

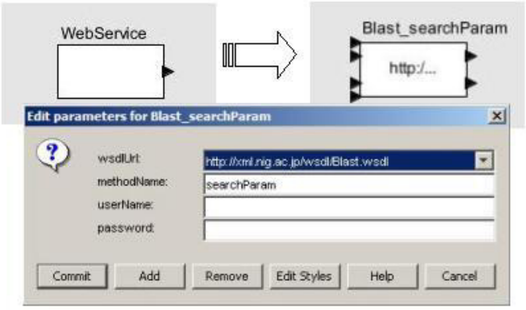
4. Workflow Execution

Kepler can execute processes locally either within the default Kepler environment (Java), or within another native environment (compiled code, or code interpreted by another environment such as Perl). In addition, the different processes implemented by actors can be executed in a distributed way, using web and grid services. By default, remote service calls behave as if they were atomic steps in the overall execution model and thus just look like native local actors to the director which orchestrates their execution.

Distributed Execution: Web-/Grid-Service Actors. Kepler's web and grid services actors allow scientists to utilize computational resources on the net in a distributed scientific workflow. Kepler's generic ***WebService actor*** provides the user with an interface to seamlessly plug-in and execute any WSDL-defined web service [18]. The user can instantiate the generic web service actor (Fig. 4, left) by providing the WSDL URL and choosing the desired web service operation. The actor then automatically *specializes* itself and adds ports with the correct inputs and outputs as given by the WSDL. The so instantiated actor (Fig. 4, right) acts as a proxy for the web service being executed. Similarly, a generic actor for grid services can be specialized into specific actors for executing remote services on the grid [12]. For conveniently plugging in a whole set of (possibly related) services, a ***web service harvester*** has been developed. It can be used to instantaneously import all web services found on a web page or in a UDDI repository [17].

Grid Actors. In addition to generic web services, Kepler also includes specialized actors for executing jobs on the grid, e.g., actors for certificate-based authentication (*ProxyInit*), grid job submission (*GlobusGridJob*), and drid-based data access (DataAccessWizard, *GridFTP*). Each of these actors access specific grid-based services using Open Grid Services Architecture (OGSA) interfaces and the Java CoG Kit [2,12].

Other Execution Environments. Supporting foreign language interfaces (as supposed to native Java execution) gives the user flexibility to reuse existing analysis components and to target appropriate computational tools. For example, Kepler (through Ptolemy) already includes a Matlab actor and a Python actor .



wsdlUrl: http://xml.nig.ac.jp/wsdl/Blast.wsdl
methodName: searchParam
userName:
password:

Figure 4. Instantiating a Web Service actor

We plan a to add further actors for execution of SAS, C++, and R(S+) code.

5. System Demonstration

We will demonstrate the Kepler system using scientific workflow examples from a number of different domains:

- ***Biology/Genomics***: The first demonstration shows how popular remote data sources and services such as Genbank and BLAST from NCBI can be integrated into a molecular biology workflow [1,4]. A version of this workflow is currently being used to automate a process called "Promoter Identification" and has led to significant time savings for the user. Since this workflow can include long-running tasks, we also use it to demonstrate a novel *notification-actor*, which can be used to send notifications to the user via email, e.g., to a cell phone or PDA.
- ***Ecology***: Our experience with metadata-driven data ingestion using the EML-Source actor has shown that generic analytical pipelines can be easily reapplied to heterogeneous data sources without manual data massaging by the scientist. This allows rapid exploratory data visualization for scientists

unfamiliar with data sources. It also enables the use of previously inaccessible data sources in complex analytical workflows. We will demonstrate the GARP algorithm (Genetic Algorithm for Rule set Production) in a scientific workflow that predicts species range distributions based on niche modeling theory [15]. Incorporating this into Kepler has broadened the data available, made archiving model predictions easier, and enabled substitution of competing prediction algorithms in the workflow, all contributing to our ability to predict the effects of climate change and invasive species on the environment.

***Geology*:** We will also demonstrate two scientific workflows illustrating geosciences applications: The ontology-enabled map integration (OMI) workflow shows how different state geologic maps can be integrated based on common integration ontologies for classifying rock types. Another demonstration exhibits an iterative, multi-level approach for classifying minerals based on their chemical composition. Both of these workflows illustrate a novel *BrowserUI* actor, which can incorporate user input and system output via a standard web browser, in the middle of any scientific workflow.

6. Implementation Status and Next Steps

Kepler inherits many advanced features such as different execution models (through directors), nested workflows, the Vergil GUI etc. from the underlying Ptolemy system. Since Ptolemy is open source and comes with an excellent and comprehensive documentation, the Kepler team (see [8] for the current members) was able to add numerous novel features within a relatively short time. The extensions are driven by actual needs as identified in scientific application projects such as SciDAC/SDM, SEEK, and GEON. The source code is freely available through the project site [8], and the first official pre-release, packaged for end users is planned for an upcoming e-Science meeting in May. The distribution will be (a) through the web site, and (b) via a self-contained "Kepler-to-Go" CD that will be handed out to the participants at the time of the conference. In particular, we plan to let participants run the above-mentioned demonstrations *on their own laptops* (several demonstrations require internet access to invoke remote services).

Acknowledgments.

Kepler includes contributors from SEEK [11], SDM Center-SPA [13], Ptolemy II [10] and Geon [5]. Work supported by NSF ITRs 022567 (SEEK), 0225673 (GEON), DOE DE-FC02-01ER25486 (SciDAC/SDM), and DARPA F33615-00-C-1703 (Ptolemy).

References

[1] BLAST: Basic local alignment search tool, http://www.ncbi.nlm.nih.gov/BLAST/

[2] I. Foster, C. Kesselman, J. Nick, S. Tuecke. 2002. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[3] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. 2002. Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation. Proceedings of the 14th Conference on Scientific and Statistical Database Management, Edinburgh, Scotland, July 2002.

[4] Genbank: National Institute of Health Genetic Sequence Database, http://www.ncbi.nlm.nih.gov/Genbank/

[5] GEON: Cyberinfrastructure for the Geosciences, http://www.geongrid.org

[6] M. Greenwood, C. Wroe, R. Stevens, C. Goble and M. Addis, Are bioinformaticians doing e-Business? Proceedings Euroweb 2002: The Web and the GRID - from e-science to e-business.

[7] Jones, M.B., C. Berkley, J. Bojilova, and M. Schildhauer, 2001. Managing Scientific Metadata, IEEE Internet Computing 5(5): 59-68.

[8] Kepler: An Extensible System for Scientific Workflows, http://kepler.ecoinformatics.org

[9] E. A. Lee and S. Neuendorffer. 2000. "MoML - A Modeling Markup Language in XML, Version 0.4," Technical Memorandum UCB/ERL M00/12, University of California, Berkeley, CA 94720, March 14, 2000. http://ptolemy.eecs.berkeley.edu/publications/papers/00/moml/

[10] Ptolemy II, http://ptolemy.eecs.berkeley.edu/ptolemyII/

[11] SEEK: Science Environment for Ecological Knowledge, http://seek.ecoinformatics.org

[12] B. Sotomayor. 2003. The Globus Toolkit 3 Programmer's Tutorial, http://www.casa-sotomayor.net/gt3-tutorial/index.html.

[13] SPA: http://kepler.ecoinformatics.org/spa.html

[14] SQL: Structured Query Language. Chamberlin, D.D., et al., "SEQUEL 2: a unified approach to data definition, manipulation and control," IBM Journal of Research and Development 20:6, pp. 560-575, 1976.

[15] Stockwell D.R.B. and D. Peters 1999. The GARP Modeling System: problems and solutions to automated spatial prediction. International Journal of Geographical Information Science 13 (2): 143-158.

[16] I. Taylor, M. Shields, I. Wang and R. Philp. 2003. Distributed P2P Computing within Triana: A Galaxy Visualization Test Case. IPDPS 2003 Conference, April 2003. http://www.gridlab.org/Resources/Papers/ipdsp_trianagalaxy_2003.pdf

[17] Universal Discovery, Description and Integration of Web Services: http://www.uddi.org

[18] Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, http://www.w3.org/TR/wsdl

[19] Extensible Markup Language, http://www.w3.org/XML/

[20] An XML Query Language, http://www.w3.org/TR/xquery/

[21] XSL Transformations, http://www.w3.org/TR/xsl