Collaborative provenance for workflow-driven science and engineering

Altıntaş, İ.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Scientific Workflows

“It is unworthy of excellent men to lose hours like slaves in the labor of calculation which could be relegated to anyone else if machines were used.”

– Gottfried Wilhelm von Leibnitz, German philosopher and mathematician

Current technology significantly accelerates the scientific problem solving process by allowing scientists to access data remotely, distribute job execution across remote parallel resources, efficiently manage data, and even perform observations in remote environments via technologies like sensor networks and cameras. The simplest examples of the transformation from manual science to automated science are the usage of personal computers to record scientific activity and the way scientists publish and search papers on online databases. A few more advanced technologies within the scientific process include sensor-based observatories to collect data in real-time, supercomputers to run simulations, domain-specific data archives that gives access to heterogeneous data and online interfaces to distribute computational experiments and monitor resources. As simulations and experiments move into the petascale regime, the orchestration of long-running data and compute intensive tasks is becoming a major requirement for the successful steering and completion of scientific investigations.

Although an increasing amount of middleware has emerged in the last few years to achieve remote data access, distributed job execution, and data management, orchestrating these technologies with minimal overhead still remains a difficult task for scientists. This push by the complexity of today’s scientific problems and state-of-the-art computer science and technology, resulted in a group of technologies developed for making the automation of scientific process more efficient and faster, with a goal to help scientists via easier utilization of technology, now called scientific workflows, a.k.a., problem solving environments (Altintas et al. 2006b). A scientific workflow is the process of combining data and processes

\[\text{This chapter is based on (Altintas et al. 2004b), (Ludäschter et al. 2006), (Altintas et al. 2006b), (Altintas et al. 2004c), (Baldrige et al. 2005), (Sudholt et al. 2006), (Jaeger-Frank et al. 2006a), (Vouk et al. 2007), (Wang et al. 2009b), (Ludäschter et al. 2009), (Barseghian et al. 2010) and (Altintas et al. 2010b) co-authored by Altintas.}\]
into a configurable, structured set of steps that implement semi-automated computational solutions of a scientific problem.

Scientific workflow management systems are useful for conducting computational studies today by making technological advances more approachable through integrative interfaces and providing appropriate abstractions of the usage of the necessary underlying resources to help scientists to focus on their research. Scientific workflow systems promote scientific discovery by providing tools and methods to generate scientific workflows via extensible and customizable graphical user interfaces for scientists from different scientific domains. They support computational experiment creation, execution, sharing, reuse and provenance tracking. Some scientific workflow systems provide efficient ways to connect to the existing data and integrate heterogeneous data from multiple resources. Shortly, scientific workflow systems allow scientists to develop formal, customizable, reusable and extensible definitions of all or part of a scientific process and execute them efficiently.

Since the initiation of the scientific workflows in the late 1990s and their initial applications, especially for solving visualization challenges, there have been significant developments in technology (Ludäscher and Goble 2005, Taylor et al. 2007). Followed by the World Wide Web and new advances in network and data research, we have witnessed the introduction of component-oriented frameworks, data and computational grids, Service-oriented architecture and Web Services, semantic data models and tools to increase domain-specificity, virtual organizations, peer-to-peer networks, virtualization and cloud computing, portals and social networks, just to name a few. Scientific workflows have evolved to satisfy different scientific requirements, computational technologies and scientific approaches that transform the scientific method. Moreover, as we learn more about efficient ways to design and execute them, they mature from art to commodity, enabling and impacting scientific studies with a number of advantages.

2.1 Example: Sea Surface Temperature MatchUp Workflow

Sea Surface Temperatures (SSTs) are important resources for many oceanographic, biological, and ecological analyses. Many types of SSTs datasets are available, from buoys and shipboard sensors to satellite-borne instruments. To facilitate the integration and comparison of these heterogeneous data sources, the Real-time Environment for Analytical Processing project, or REAP, has developed two Kepler-based (Ludäscher et al. 2006) scientific workflows to retrieve and perform statistical analyses on SSTs (Barseghian et al. 2010), which together are named the SSTs MatchUp Workflow.

SSTs datasets are very large and stored at many different organizations, so only small subsets may be compared. To facilitate the analysis of these subsets, a Match Up database can be created to store the individual measurements from different data sets with corresponding temporal and spatial locations. The workflow in Figure 2.1(a) builds a Match Up database based on user-supplied parameters including which SSTs datasets to use, the number of samples (i.e., size of the subset) and the timespan and region of ocean. For each sample, the work-
flow randomly chooses a time and location within the user-supplied ranges, and downloads the corresponding SST measurements from each data set into the database. Many SST data sets are stored in OPeNDAP servers, which the workflow accesses through an OPeNDAP Actor in Kepler. Actors are the data processing elements in Kepler and they communicate with each other through well-defined data interfaces called Tokens. Additionally, RDBMS actors write data to the Match Up database. Once a Match Up database has been created, the Analysis workflow (shown in Figure 2.1(b)) creates a Keyhole Markup Language (KML) file for Google Earth that displays each samples location and time along with the average and standard deviation of the temperatures between the datasets (shown in Figure 2.1(c)). This workflow performs these calculations using R and Python actors.

2.2 Requirements for Scientific Workflows

Scientific workflows vary in their requirements and the technologies they use as we switch between disciplines. In spite of the fact that the development in scientific workflows is motivated by application pull from different scientific disciplines and projects, and the disciplines have their own wish lists and technical requirements, it is possible to unify these require-

---

1 Open-source Project for a Network Data Access Protocol (OPeNDAP) website: http://opendap.org/
2 Google’s KML Documentation: http://code.google.com/apis/kml/documentation/
ment and come up with a set of common requirements useful for multiple disciplines. Consider, e.g., the Cyberinfrastructure desiderata of a geoscientist and a computational chemist. A geoscientists wish list might involve online data acquisition and access, managing large databases, indexing data on spatial and temporal attributes, quick subsetting operations on the databases, large scale resource sharing and management, collaborative and distributed applications, parallel gridding algorithms on large data sets using high performance computing, integrating data with other related data sets, e.g., geologic maps and hydrology models, and easy-to-use user interfaces from portals and scientific workflow environments (Altintas et al. 2010b). A computational chemist’s wish list is comprised of performing synchronously many embarrassingly parallel calculations with different molecules, configurations, methods and/or other parameters, using existing data, codes, procedures, and resources with the usual sophistication and accuracy, letting different programs seamlessly communicate with each other, integrating computational experiment preparation and analysis steps, distributing jobs automatically onto a computational grid, having interfaces to database infrastructures to obtain starting structures and to allow data mining of results, and tools and scientific workflow environments that are as easy-to-use, general, flexible, manageable and reusable as possible (Sudholt et al. 2006). While these requirements have different focuses, their fundamental requirements are similar, i.e., querying and integrating data sets, execution of domain specific codes and algorithms, using different tools through easy-to-use unified interfaces. When these requirements are extended to the other sciences, a list of requirements that are generically applicable to different disciplines via customization and domain-knowledge integration emerge.

The following requirements are a summary based on the interactions with a variety of scientists from scientific disciplines involving molecular biology, ecology, astrophysics, fusion, geosciences, oceanography, computational chemistry and phylogeny:

- Design tools that are scalable and especially helpful to non-technical users (Bowers and Ludäscher 2005)
- Easy to use, fairly simple user interfaces having more complex features hidden in the background (Ngu et al. 2008)
- Hierarchical component and service composition (Altintas et al. 2004c) to reduce complexity and enhance design reusability
- Reusable features that are generic enough to serve different communities but customizable enough to serve one scientific domain
- Extensibility for the expert user
- Registration, publication and provenance of data and process products (Altintas 2008)
- Seamless access to and dynamic plug-in of data and processes from registries/repositories (Abramson et al. 2005)
2.3 Life-cycle of Scientific Workflows

The various phases and steps associated with developing, deploying and executing scientific workflows form a scientific workflow life-cycle. (Deelman and Gil 2006), (Gil et al. 2007) and (Ludäscher et al. 2009) summarize different aspects of this life-cycle. The steps shown in build and run phases in Figure 2.2 are largely supported by existing workflow systems, although often using different approaches and techniques. Share and learn phases often include a larger infrastructure containing workflows, thus may or may not depend on the scientific workflow system being used other than their dependency on the provenance and data associated to build and run steps.

- **BUILD**: Scientific workflow building involves composition of process flows out of lower-level components either from scratch or by extending an existing workflow. This phase often involves a graphical user interface that allow placing, customizing and linking scientific workflow components, e.g., (Ludäscher et al. 2006, Oinn et al. 2006),

Figure 2.2: Steps in the life-cycle of workflow design, execution and experimentation.

- Distributed (Altintas et al. 2004a) and detached (Jaeger-Frank et al. 2006b) workflow execution, e.g., Web, Grid and Cloud awareness
- Semantics awareness (Bowers and Ludäscher 2004) using ontologies and domain knowledge
- Execution monitoring, failure recovery (Craw et al. 2008, Mouallem et al. 2010), smart re-runs (Altintas et al. 2006a) and interactive user steering
Callahan et al. 2006), based on a conceptual workflow specification. The designed workflow can then be saved and reopened in an exchange format, e.g., SCUFL (Oinn et al. 2006) and MoML (Ludäscher et al. 2006), or as a script, e.g., Python. In addition to this visual programming model, some workflow systems, e.g., Pegasus (Deelman et al. 2004), use scripting to build scientific workflows more in line with the traditional programming.

- **SHARE**: Once designed and saved, scientific workflows can be published through public repositories or shared with a group of colleagues who can then run, extend or validate the design of a workflow. Some repositories allow for searching for workflows or provide capabilities to share workflows with a limited social network (Goble et al. 2010). Deploying, i.e., packaging and publishing, workflows through e-Science infrastructures or a repository can be provided by the workflow system or can be done manually. Sharing of workflows (and the results of workflow runs) helps the continuity of the scientific workflow life-cycle by increasing reuse and extensibility.

- **RUN**: Ability to run scientific workflow is supported by all scientific workflow management systems at different levels. As a first step, this phase includes scheduling and execution planning, e.g., resource planning, allocation, data staging, functions. Once workflow execution is planned, it can be executed. The execution step is managed by a workflow engine that uses one or more models of computation as a basis, e.g., dataflow, and includes provenance tracking, fault tolerance and other execution related functionality depending on the workflow system. Most systems also provide a way to monitor the progress of execution workflows and data flowing through the workflow execution in the run phase.

- **LEARN**: An important phase of the workflow life-cycle is learning from previous workflow executions. This phase often includes reviewing of the results of previous runs, browsing and analyzing the accumulated provenance information and reporting of workflow execution results. When provided by a workflow system, this phase enhances the exploratory process and leads to future studies based on existing workflow-driven research.

### 2.4 Advantages and Limitations of Scientific Workflows

Workflows enormously improve data analysis, especially when data is obtained from multiple sources and generated by computations on distributed resources and various analysis tools. These advances in systematic analysis of scientific information made possible by workflows have unleashed a growing need for automated data-driven applications that also provide scientists with interfaces to design, create, execute, share, reuse scientific workflows, and collect and manage the provenance of the data and processes with little overhead. The scientific workflow approach offers a number of advantages:
• **Scientific workflows formalize the scientific process.**
  A scientific workflow provides a structured definition of one or more steps towards solving a scientific problem. At its simplest, this definition is either **textual**, a formal representation of tasks and dependencies, or **visual**, a graph of tasks and dependencies. This formal definition is akin to the code of a computer program (or scripts) and it is an important step for achieving collaborative work as well as to maximize reuse of existing work. It can then be executed, shared, customized, extended and published as shown in Figure 2.2.

• **Scientific workflows are easier to share, deploy under different platforms, customize and rerun, extend and reuse compared to the traditional computing approaches.**
  Scientific workflows document processes being performed in scientific studies. A scientific workflow definition resembles a recipe (workflow) based on a set of ingredients (data) and cooking process (computations). Similar to the execution of a workflow, a cook starts with ingredients and parameters, e.g., number of servings, and applies the recipe transforming the initial ingredients with new ones and coming up with a final product, in this case, a meal for number of servings. Some intermediate products could be prepared and used in the process. Similarly, once a scientific workflow is built, it can be reused to execute with different datasets or customized through its on-the-fly updateable parameters. Each time a workflow is shared, it can be fully or partially reused, extended or re-run. Sharing increases the possibility of reuse of a workflow and the collaborative performance of the scientists using it.

• **Scientific workflows can help with management of complexity and usability of scientific processes.**
  Most scientific workflows allow for hierarchical composition of tasks by conceptually grouping sets of components as conceptual steps. These hierarchal units are akin to procedures in programming languages. The conceptual groups, i.e., sub-workflows, also help to translate from the low-level programming concepts to the high-level scientific concepts, bridging the gap between the workflow developer and the workflow user.

• **Scientific workflows provide interfaces to different technologies from a unified interface and allow for interaction with multiple tools and resources simultaneously.**
  Scientific workflows often provide a one-stop-shop interface that encapsulates multiple technologies. They consolidate the usage of similar technologies under a conceptual abstraction, e.g., a data transfer component that provide data transfer using GridFTP, scp and SRB based on users credentials and parameter selections.

• **Scientific workflows can be annotated with domain-knowledge.**
  Some scientific workflows allow for semantic annotations to data and actor schemas (Bowers and Ludäscher 2005) using conceptual information from shared ontologies.
These semantic annotations can then be used to support scientific workflow design, discovery, reuse, and validation in the presence of thousands of potentially useful actors and datasets.

- **Scientific workflows make it possible to track provenance of the data and processes.**

Provenance information is useful in scientific workflows in a variety of ways. The collected information could be used for evaluation of the results as well as for mining different patterns during workflow design, keeping the association of results to processes, making it it easier to validate/regenerate results and processes, and comparing different workflow versions. How provenance information is used depends on the particular user as different workflow users need information about different phases of the workflow.

- **Scientific workflows can help with execution monitoring, fault tolerance and smart re-running of a computational experiment.** Capturing provenance information in scientific workflows is not only useful for determining data dependencies, but also for a wide range of queries including fault tolerance and usage statistics. It is not uncommon for a scientific workflow builder to query provenance logs to analyze the history of the runs of a workflow or to recover from a system failure. As collaborative scientific workflow environments provide users with reusable shared workflows, collection and usage of provenance information in a generic way that could serve multiple data and computational models become vital. Scientific workflows can provide efficient improved reusability and maintenance of workflows and components (actors), automated provenance management, “smart” re-running of different versions of workflow instances, monitoring of long running tasks, and support for fault-tolerance and recovery from failures.

Of all the listed advantages, provenance tracking is a very important feature of scientific workflow systems as it helps track the origin of scientific end products, validate and repeat experimental processes that were used to derive these scientific products. The lifecycle of scientific workflow provenance starts with workflow design and execution. Provenance collection for scientific workflows must have the ability to create and maintain associations between workflow inputs, workflow outputs, workflow definitions, and intermediate data products. We will revisit scientific workflow provenance in the next chapter and throughout this thesis as a basis for the presented research.

**Limitations of Scientific Workflows.** Although the scientific workflows provide a significant set of advantages as listed above, they also have several limitations which affects their adoption by the computational scientists. First of all, scientific workflows are dependent on third-party applications and toolkits they wrap and orchestrate. The ability to link commodity applications is a benefit, but it also poses some problems. Like any programming-in-the-large (DeRemer and Kron 1975) product, changes to workflows can become difficult, since these
components are generally not designed to be integrated with other tools and scientific workflow tools treat them as black boxes. The versioning and packaging of third party applications, components and services is often not managed by scientific workflow systems, which in turn has effects on the sustainability and backwards compatibility of a scientific workflow as a scientific artifact. The interfaces for the components that are required to run a workflow might change over time, the standards, exchange languages (e.g., Web Service Resource Framework (Czajkowski et al. 2004)) and technologies (e.g., Java version) that a workflow system supports might evolve making the third party tools not functional under new versions of the system.

The goal to assemble external components that are not necessarily designed to fit, e.g., syntactically or semantically incompatible, also creates a need for building extra components for data transformation, called shims. (Lin et al. 2009b) reported that 30% of the tasks within the 560 workflows in myExperiment (Goble et al. 2010) were shim services. Shims are not always available, requiring the users to first build shims, which can be difficult. The frequent use of shims result in workflows that are cluttered and harder to reuse. There are, however, some existing efforts to overcome this limitation, e.g., (Zinn et al. 2009) develop a Process Networks-based (Lee and Parks 1995) model of computation in which actors are more configurable through scope expressions that replace some of the shims by high-level actor configurations. Secondly, although scientific workflow system can be regarded as visual programming tools, they still require users to learn a new language, i.e., the workflow language offered by the system they choose to adopt. Each scientific workflow system comes with its own language and engine, requiring users to think in a certain way, mostly in a variation of the dataflow programming paradigm, e.g., (Kahn 1974). Many current scientific workflow systems also have limited or no support for debugging, exception handling, and fault-tolerance. Other useful capabilities for scientific workflows include transparent, reliable, high-throughput data transport or the seamless use of cluster and Cloud resources. Unfortunately, current systems usually require the workflow developer to explicitly code such capabilities into the workflow, i.e., they are not built into the workflow engine.

2.5 Scientific Workflow Systems

Numerous scientific workflow systems exist today for scientific computing. In this section we’ll give an overview of some of these scientific workflow management systems, review Kepler in detail and discuss the common components of the reviewed systems.

The open-source Taverna workbench (Oinn et al. 2006, Hull et al. 2006, Tan et al. 2010) for design and execution of scientific workflows is developed by the myGrid\(^3\) Team. Used primarily by bioinformatics, computational chemistry, medical imaging, social science and astronomy communities, Taverna offers an intuitive graphical user interface that guides the computational scientists with the help of domain specific information. The Taverna scientific

\(^3\)myGrid website: http://www.mygrid.org.uk/
workflow graphs consist of linked *processors*. Processors represent Web Services, command line tools, and other data processing and transformation components. In addition, the system allows for plug-in processors by the user community. The workflow are saved in an XML form using SCUFL (Oinn et al. 2006) and executed using a computational model based on the lambda calculus. Provenance tracking is an important feature of Taverna focusing on collecting and persistently storing information during workflow runs, and querying the collected information to answer the scientists questions regarding the provenance of their data.

The open-source Triana problem solving environment (Taylor 2006) is developed at Cardiff University. Triana also provides an intuitive visual interface to design and execute workflows by dragging and dropping a variety of data analysis tools that are built-in and integrated by users. The environment has been used in a range of tasks, such as signal, text and image processing. Once designed, Triana workflows can be imported from or saved in a variety of forms, mainly the Web Service Flow Language (WSFL) (Leymann 2001). Triana supports distributed execution of parts of a workflow on a peer-to-peer network.

Pegasus (Deelman et al. 2004) is a framework for mapping scientific workflows onto distributed resources including Grid and Cloud-based systems. It allows users to customize various aspects of the system including the target execution system, resource definition and selection, task clustering, data staging and data fetching. Based on an abstract workflow defined by the end user, Pegasus finds a mapping of the available resources to execute the defined workflow on distributed resources. The framework uses the allows for partition-level fault recovery. Abstract workflows saved in Pegasus is saved in DAX format which is the execution independent XML representation of a directed acyclic graph.

Vistrails (Callahan et al. 2006) is an open-source scientific workflow system mainly targeted at supporting data exploration and visualization. A distinguishing feature of Vistrails is its comprehensive provenance infrastructure for rapidly-evolving and explorative workflows. The system allows users to navigate through the provenance archives in XML or relational form via an intuitive user interface.

WS-VLAM (Korkhov et al. 2007) is a Grid-enabled workflow management system following the Open Grid Services Architecture (OGSA) (Foster et al. 2002) and Web Service Resource Framework (WSRF) (Czajkowski et al. 2004) standards. The WS-VLAM engine supports the monitoring and interactive runtime control of the application workflows with ability to disconnect from a long running workflow without stopping it.

Askalon (Fahringer et al. 2007) programming environment for Grid computing is designed for the design, execution and optimization of applications that can harness the power of Grid computing. The environment build upon a service-oriented architecture using services for resource management, discovery, workflow scheduling and execution, and performance prediction and analysis. Askalon workflows are saved using Abstract Grid Workflow Language (AGWL) that describes Grid workflow applications at a high level of abstraction.

Swift (Wilde et al. 2009) is a scientific workflow system for specification, execution, and management of large-scale science workflows, e.g., to analyze large quantities of data or per-
form parameter studies. It combines a scripting language to define high-level computations and data mapper to access different data formats. Swift users span the physical sciences, biological sciences, social sciences, humanities, computer science, and education.

KNIME (The Konstanz information miner) (Berthold et al. 2009) is an open source data analytics, reporting and integration platform for machine learning and data mining. It is built upon the modular data pipelining concept, and provides a graphical user interface based on Eclipse for assembly of nodes for data preprocessing, namely for extraction, transformation and loading. KNIME is primarily used in pharmaceutical research with applications in other areas like customer data analysis, business intelligence and financial data analysis.

The LONI pipeline (Rex et al. 2003) is a workflow system primarily aimed at neuroimaging community for assembly of available neuroimaging tools. The LONI pipeline workflow are built based solely on external stand-alone applications or services, whose interface methods are then described in the Pipeline XML language.

P-GRADE Grid Portal (Kacsuk and Sipos 2006) is a web based environment for developing, executing and monitoring workflows and workflow based parameter study applications using various Grid middleware technologies. P-GRADE Workflow Editor enables workflow creation and modification through the portal while the portal provides support for management of Grid environments, Grid certificates, and workflow execution and monitoring.

DiscoveryNet (Rowe et al. 2003) is another visual component integration-based workflow system that provides a graphical user interface to build workflows out of existing third-party tools and services. It also allows for service providers to publish their software components for data analysis and mining. The developed workflows are saved in an XML-based format called Discovery Markup Language (DPML). DiscoveryNet technology and system was later commercialized as a series of products through InforSense⁴.

Pipeline Pilot is another commercial tool built by Accelrys⁵ for scientific data analysis and integration using scientific workflows. Pipeline Pilot provides a large set of domain-specific component libraries including bioinformatics and computational chemistry. In addition to a graphical workflow building interface, Pipeline Pilot allows users to visualize the data at each step of the process and directly view scientific data.

2.5.1 Kepler Scientific Workflow Environment

The Kepler scientific workflow system is developed by a cross-project collaboration to serve scientists from different disciplines, and is currently coordinated by the Kepler/CORE project. A large number of large-scale projects and individual researchers encompassing multiple disciplines have used Kepler to manage, process and analyze this increasing amount of scientific data. For example, in the CAMERA project (Seshadri et al. 2007, Sun et al. 2010), Kepler scientific workflows are used to create, deploy and execute applications based on anal-

---

⁴InforSense website: http://www.inforsense.com/
Figure 2.3: (a) The configurable Kepler Provenance Recorder acts as a recording component of the Kepler Provenance Architecture. (b) Kepler Provenance Architecture is designed to collect provenance information in different formats into a provenance store and provides querying API for accessing the collected information.
2.5 Scientific Workflow Systems

Figure 2.4: The reporting and workflow run manager modules in Kepler use the provenance information through a common querying API in Kepler.

ysis of metagenomics data using community provided tools (Altintas et al. 2010a). Inherited from Ptolemy II, Kepler adopts the actor-oriented modeling paradigm for scientific workflow design and execution.

Kepler provides a graphical user interface for designing workflows composed of a linked set of components called Actors that may execute under different Models of Computation (MoCs) (Goderis et al. 2009). Actors are the implementation of specific functions that need to be performed and communication between actors takes place via tokens that contain both data and messages. MoCs specify what flows as tokens between the actors; how the communication between the actors is achieved; when actors execute (a.k.a. fire); and when the overall workflow can stop execution. The support for multiple MoCs in Kepler is provided by components called Directors. The designed workflows can then be executed through the same user interface or in batch mode from other applications.

Kepler provides an intuitive graphical user interface and an execution engine to help scientists edit and manage scientific workflows, collect provenance information related to the developed workflows, and generate reports on their executions over time. The execution engine can be separated from the graphical user interface enabling the execution of existing workflows in batch mode in a centralized or distributed fashion (Sudholt et al. 2006, Wang et al. 2008). With built-in wizard tools, customized workflow components (actors) can be easily exported for reuse and sharing locally or publicly through the Kepler actor repository6.

---

6Kepler Analytical Component Repository: http://library.kepler-project.org/kepler/
Figure 2.5: Different Distributed Execution Pattern Techniques in Kepler are applied to numerous computational science challenges, including: (a) Map Reduce, and (b) Master-Slave, used in the Ecology domain for House Finch spatial stochastic birth-death process simulation.

Provenance Tracking in Kepler and Related System Modules

Kepler also provides a provenance framework (Altintas et al. 2006a, Ludäscher et al. 2008, Crawl and Altintas 2008) that keeps a record of chain of custody for data and derived products.
Figure 2.6: Different Parameter Sweep Execution Techniques in Kepler are applied to numerous computational science challenges, including: (a) Nimrod/K used in the Chemistry domain for quantum chemical calculations; and (b) SSH tunneling used in the Physics domain for plasma fusion simulation.

Within workflow design and execution. The Kepler Provenance Recorder (KPR) collects information about workflow structure and executions to enable users to track data generated by domain specific programs. As seen in Figure 2.3(a), KPR has plug-in interfaces for new
data models, metadata formats and storage destinations that were designed to serve the multi-disciplinary requirements of a broad user community. An architecture (see Figure 2.3(b)) including the KPR, that allows for binding different data models, collecting of application-specific provenance data and using results through a dashboard has already been created (Vouk et al. 2007). The center of this architecture is Provenance Store: a database providing physical storage and an API to access the database. The API has three components: (1) Kepler, its actors, and external scripts use a Recording API to collect and save provenance information; (2) a Query API provides different query capabilities for dashboards, and query actors in Kepler; and (3) a Management API. Three types of provenance information are collected: the workflow structure, e.g., actor types and parameter values, workflow evolution, e.g., how parameter values change over time, and workflow execution, e.g., the data products read and written by actors during execution. The provenance information collected by the recorder can be stored to multiple data models including an SQL schema. Additionally, a Query API has been implemented to retrieve provenance information from this schema and is used by the Kepler Reporting System, as seen in Figure 2.4.

In addition to KPR, Kepler provides a fault-tolerance framework (Mouallem et al. 2010) in the context of the failures observed in data- and compute-intensive workflow runs such as parameter-sweep studies. The framework is divided into three major components: (i) a general contingency Kepler actor that provides a recovery block functionality at the workflow level; (ii) an external monitoring module that tracks the underlying workflow components, and monitors the overall health of the workflow execution; and (iii) a checkpointing mechanism that provides smart resume capabilities for cases in which an unrecoverable error occurs. This framework takes advantage of the provenance data collected by the KPR to detect failures and help in fault-tolerance decision-making.

Distributed Computing in Kepler

Apart from modules for the core, provenance recording, fault tolerance, run management and reporting, Kepler provides an extensible environment for designing and executing distributed computational studies.


Figure 2.5 and Figure 2.6 show a set of parameter sweep execution workflows, which were designed in Kepler: Figure 2.5(a) for MapReduce (Wang et al. 2009a) used in the Ecology domain for House Finch spatial stochastic birth-death process simulation; Figure 2.5(b) for Master-Slave distributed framework (Wang et al. 2008, Wang et al. 2009b) used in
2.5 SCIENTIFIC WORKFLOW SYSTEMS

Figure 2.7: A four-layer reference architecture for scientific workflow management systems. The User Interaction Layer facilitates the interaction of users with the workflow design, execution and monitoring interfaces along with the workflow repositories. The Workflow Execution Layer consists of workflow engines that support workflow scheduling, execution and failure management. The Process Layer supports the user interaction and workflow execution layers with background processes including listeners for provenance, data and task execution management. The Physical Resource Layer provides interfaces to utilize computational and data resources.

the Ecology domain for House Finch spatial stochastic birth-death process simulation; Figure 2.6(a) for Nimrod/K, also called TDA, director (Abramson et al. 2008, Abramson et al. 2009, Smanchat et al. 2009) used in the Chemistry domain for quantum chemical calculations; and Figure 2.6(b) for Ad-hoc SSH tunneling controlled by Kepler workflows composed of generic Kepler actors and dataflow directors (Podhorski et al. 2009, Podhorski et al. 2007) used in the Physics domain for plasma fusion simulation.
2.5.2 A Reference Architecture for Scientific Workflow Management Systems

Although a set of major scientific workflow systems were discussed in this section, we recognize that this list is not a comprehensive list of all the existing systems. This list was selected based on our interactions with the scientific workflow community as examples to major trends in scientific workflow development, e.g., workflows for distributed execution on parallel, Grid and Cloud resources, visual workflow assembly vs. scripting-based workflow development, and provenance support. Based on the review of the above, this section defines a reference architecture that pulls together the typical features and components in scientific workflow systems.

As indicated in the brief descriptions above, each scientific workflow system has a workflow language specific to the (set of) engine(s) underlying each system. Workflows are either described through wiring of components using a graphical user interface or though a scripting language that then get ingested by the workflow engine for scheduling, resource provisioning and execution. Different systems optionally provide modules for provenance tracking, Grid and Cloud management, execution monitoring, reporting, data visualization and reporting. Figure 2.7 illustrates a typical four-layer architecture that draws on the common components of existing scientific workflow management systems. Note that the layers for user interaction, workflow run management, process management (for background listeners and helper processes) are generally a part of the workflow system itself. However, the physical resource layer depicts the external data and computational resources that the workflow system interacts during the execution of a workflow.

(Lin et al. 2009a) also discussed a reference architecture for scientific workflows. This architecture was used as a basis for the design and development of the VIEW SOA Solution (Lin et al. 2009a) for scientific workflows.

2.6 Usages and Current Challenges

Since the introduction of scientific workflow, they have been used for multiple steps of the scientific process mainly involving data analysis, integration, visualization and distributed execution activities. There have been significant advances in design and execution of scientific workflows making them a potential commodity for scientific research. However, the research in this area still faces some challenges to make scientific workflows more efficient and useful in the context of a scientific researchers day-to-day activities. Within these challenges are access to heterogeneous data and computational resources and link to different domain knowledge (Gil et al. 2007), ability to interface to multiple analysis tools and workflow systems (Zhao et al. 2006c, Goble et al. 2010), and comprehensive support for computational experiment creation, execution, sharing, reuse and provenance that fits the work methods of scientists while managing complexity, user and process interactivity. In order for workflow-based research to be fully efficient, the scientific workflow systems should at the minimum provide extensions for optimize failure recovery, smart re-runs, and more importantly concep-
tually consolidated file and process transport mechanisms that are technology-independent from users perspective. The biggest challenge for making scientific workflows useful still remains to be end-to-end support for scientific studies. Scientific workflows have the potential to be useful in all steps of the scientific process from observation to publication of scientific data. This requires scientific workflow management systems to have support for not only data management and analysis, but also interact with networked scientific instruments, research networks and observatories along with provenance tracking within and across experiments modeled as workflows beyond the active lifetime of a workflow including citations to it.

Summary

Definition and requirements of scientific workflows and their life-cycle, advantages, and limitations were discussed. A brief review of the features provided by some of the well-known scientific workflow systems and a typical reference architecture is also included. Next chapter focuses on challenges related to provenance within the scientific workflow life-cycle.