CIS 602-01: Computational Reproducibility

Scientific Workflows

Dr. David Koop
Containers to Bridge Multiplicity Issues

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container...

...that can be manipulated using standard operations and run consistently on virtually any hardware platform

[Docker, Inc., 2016]
Dockerfiles, Images, and Containers

- Dockerfiles are used to build Docker images
- A Docker image can be used to run a Docker container
- Containers can be composed together using Docker Compose
Dockerfile

# base image: last debian release
FROM debian:wheezy

# name of the maintainer of this image
MAINTAINER Anthony.Baire@irisa.fr

# install the latest upgrades
RUN apt-get update && apt-get -y dist-upgrade

# install nginx
RUN apt-get -y install nginx

# set the default container command
# -> run nginx in the foreground
CMD ["nginx", "-g", "daemon off;"]

# Tell the docker engine that there will be something listening on the tcp port 80
EXPOSE 80

[A. Baire, 2016]
Technical Challenges in Reproducibility

- Dependency Hell
- Imprecise Documentation
- Code rot
- Barriers to adoption and reuse in existing solutions

[C. Boettiger, 2015]
Docker Features for Reproducibility

• Integrating into local development environments
• Modular reuse (Docker compose)
• Portable environments (snapshots)
• Public repository for sharing (DockerHub)
• Versioning

[C. Boettiger, 2015]
Best Practices

• Use Docker containers during development
• Write Dockerfiles instead of installing interactive sessions
• Adding tests or checks to the Dockerfile
• Use and provide appropriate base images
• Share Docker images and Dockerfiles
• Archive tarball snapshots

[C. Boettiger, 2015]
The dependency problem in packages

Packaging instructions and metadata

Filesystem /usr

building package

Filesystem /usr with package in it

[R. Garbas, 2016]
nix(os) solution for the dependency problem

Packaging instructions and metadata $\rightarrow$ building package A $\rightarrow$ Filesystem /nix/store/$\text{hash}$-$\text{name}$-$\text{version}$

Packaging instructions and metadata $\rightarrow$ building package B $\rightarrow$ Filesystem /nix/store/$\text{hash}$-$\text{name}$-$\text{version}$

[R. Garbas, 2016]
Using containers as part of a workflow

- rabix: https://github.com/rabix/rabix
- bioboxes: http://bioboxes.org
Workflows

• (Business) Workflow Definition [Software AG]

“An orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes that transform materials, provide services, or process information.”
Business Workflow

Diagram showing the steps of a business workflow: Enter trip facts, Change trip facts, Trip facts, Check trip facts, Trip facts consistent, Corrections required, Trip facts sent back for correction, Trip reimbursement of trip facts rejected, Reimbursement of expenses approved, Rejected, Correct, Notification, Expenses dept., Superiors.
Business Workflows

• Also know as “Business Process Management” or “Business Process Engineering”

• Roots in office automation (think assembly lines for offices)
  - Each person has certain roles
  - For different processes, there are flows on how decisions are made or transferred

• Newer: “Web Service Choreography”
Scientific Workflows

• Scientific Workflow Definition [Ludäscher et al.]

  “[P]rocess networks that are typically used as data analysis pipelines or for comparing observed and predicted data, and that can include a wide range of components, e.g., for querying databases, for data transformation and data mining steps, for execution of simulation codes on high performance computers”
There is a growing interest in scientific workflows as can be seen from a number of recent events, e.g., the Scientific Data Management Workshop [SDM03], the e-Science Workflow Services Workshop [eSc03], the e-Science Grid Environments Workshop [eSc04], the Virtual Observatory Service Composition Workshop [GRI04], the e-Science LINK-Up Workshop on Workflow Interoperability and Semantic Extensions [LIN04], and last not least, various activities as part of the Global Grid Forum (e.g, [GGF04]), just to name a few. Scientific workflows also play an important role in a number of ongoing large research projects dealing with scientific data management, including those funded by NSF/ITR (GriPhyN, GEON, LEAD, SCEC, SEEK, ...), NIH (BIRN), DOE (SciDAC/SDM, GTL), and similar e-fforts funded by the UK e-Science initiative (myGrid, DiscoveryNet, and others). For example, the SEEK project [SEE] is developing an Analysis and Modeling System (AMS) that allows ecologists to design and execute scientific workflows [MBJ+04]. The AMS workflow component employs a Semantic Mediation System (SMS) to facilitate workflow design and data discovery via semantic typing [BL04]. Thus SEEK is a good example of a community-driven project in need of a system that allows users to seamlessly access data sources and services, and put them together into reusable workflows. Indeed SEEK is one of the main projects contributing to the cross-project Kepler initiative and workflow system discussed below.

Aspects and Types of Workflows. Scientific workflows often exhibit particular "traits", e.g., they can be data-intensive, compute-intensive, analysis-intensive, visualization-intensive, etc. The workflows in Sections 2.1.1, 2.1.2, and 2.1.3, e.g., exhibit different features, i.e., service-orientation and data analysis, re-engineering and user interaction, and high-performance computing, respectively. Depending on the intended user group, one might want to hide or emphasize particular aspects and technical capabilities of scientific workflows. For example, a "Grid engineer" might be interested in low-level workflow aspects such as data movement and remote job control. Having workflow components (or actors) that operate at this level will be beneficial to the Grid engineer. Conversely, a scientific workflow system should hide such aspects from analytical scientists (say an ecologist studying species richness and productivity).

The Kepler system aims at supporting very different kinds of workflows, ranging from low-level "plumbing" workflows of interest to Grid engineers, to analytical knowledge discovery workflows for scientists, and conceptual-level design workflows that might become executable only as a result of subsequent refinement steps [BL05].

In the following we first introduce scientific workflows by means of several examples taken from different projects and implemented using the Ptolemy ii-based Kepler system [KEP]. We then discuss typical features of scientific workflows and from this derive general requirements and desiderata for scientific workflow systems. We take a closer look at underlying technical issues and challenges in Section 3.

2.1 Example Workflows

2.1.1 Promoter Identification

Figure 1 shows a high-level, conceptual view of a typical scientific knowledge discovery workflow that...
Scientific Workflows

- Manage **data-intensive, complex** analyses
- Orchestrate different tools
- Structured computation
- Abstraction for more general understanding
- Enable automation, reproducibility, sharing
Business Workflows vs. Scientific Workflows

- **Business Workflows**
  - Decision-oriented
  - Emphasis on control-flow
  - Often involve many people
  - Often stateful

- **Scientific Workflows**
  - Data-oriented
  - Emphasis on data-flow
  - Usually involve a small group
  - Usually stateless

---

**Figure 1:** Conceptual (“napkin drawing”) view of the Promoter Identification Workflow (PIW) [ABB+03]
Types of Scientific Workflows

• Knowledge Discovery
  - Often executed once, changed, and new version executed again
  - Emphasis of VisTrails workflow system

• Automation
  - Used when same process is repeated for different data
  - Users often use higher-level interfaces to the workflows

• Coordination
  - Example: High-Performance Computing (HPC)
    • Move files from local machines to HPC resources
    • Coordinate execution on HPC resources
Workflows and e-Science: An overview of workflow system features and capabilities

E. Deelman, D. Gannon, M. Shields, I. Taylor
Definitions

• Workflow orchestration: defining graph of tasks to manage a task
• Workflow: a template for such an orchestration
• Workflow instance: a specific instantiation of a workflow for a particular problem and tool
Fig. 1. The workflow lifecycle: composition, mapping, execution, and provenance capture.

or comparing every workflow solution that exists today, we focus here on the scientist’s or distributed-application developer’s view and attempt to extract a feature set that encapsulates the functionality exposed by various workflow systems in an attempt to create a generalised taxonomy for the field as a whole rather than comparing specific solutions. Therefore, the systems we include in this paper only constitute a subset of the many workflow systems that exist within the research and commercial communities and provide an illustrative means for the justification of our feature set.

It is hoped that such a taxonomy of features will be useful not only to scientists or application-developers when considering which features might be important for their problem domain but also to the workflow community as a whole in order to provide some basis for classification of systems in the future and possible interoperability. It is as inconceivable at this stage to see a one-size-fits-all workflow system that encompasses all of the desirable features in one solution as it is to ask for one programming language to be best for all problems. Therefore any decision about the choice of one workflow system or systems used must be informed in order to reduce the development time of applications as a whole in this area.

2. Workflow capabilities and the workflow lifecycle

At one level a workflow is a high-level specification of a set of tasks and the dependencies between them that must be satisfied in order to accomplish a specific goal. For example, a data analysis protocol consisting of a sequence of pre-processing, analysis, simulation and post-processing steps is a typical workflow scenario in e-Science applications. At the level of representation and execution, a workflow is a computer program and it can be expressed in any modern programming language. However, the task of writing a computer program in Perl or Java or Python to orchestrate a set of tasks on a wide-area distributed system goes well beyond the programming skills or patience of most scientific users. The goal of e-Science workflow systems is to provide a specialized programming environment to simplify the programming effort required by scientists to orchestrate a computational science experiment.

In general we can classify four different phases of the workflow lifecycle (as seen in Fig. 1):

1. Composition, Representation and Data Model: the composition of the workflow (abstract or executable) through a number of different means e.g. text, graphical, semantic.
2. Mapping: Involves the mapping from the (abstract) workflow to the underlying resources.
3. Execution: Enactment of the mapped workflow on the underlying resources.
4. Metadata and Provenance: The recording of metadata and provenance information during the various stages of the workflow lifecycle.

During workflow composition the user creates a workflow either from scratch or by modifying a previously designed workflow. The user can rely on workflow component and data catalogs. The workflow composition process can be iterative, where portions of the workflow need to be executed before subsequent parts of the workflow are designed. Once the workflow is defined, all, or portions of the workflow can be sent for mapping and then execution. During that phase various optimizations and scheduling decisions can be made. Finally, the data and all associated metadata and provenance information are recorded and placed in a variety of registries which can then be accessed to design a new workflow. Even though we delineate data recording as a separate phase of the workflow lifecycle, this activity can and often is part of the workflow execution.

In the following four sections, we examine these four areas in detail by extracting a feature set within each category to define the common aspects of functionality that are inherent across workflow systems in general.
Phases of Workflow Lifecycle

- Composition: defining the workflow, may be new or based on another workflow
- Mapping: map (abstract) workflow to resources
- Execution: running the workflow
- Metadata and Provenance: recording information about the workflow at the various steps
Workflow Composition

- Textual Composition
- Graphical Workflow Editing
- Community Services (e.g. Web services)
- Semantic Composition (automated reasoning, etc.)
Workflow Representation

- Directed acyclic graphs: outputs to inputs (BPEL, MoML, etc.)
- Petri nets (e.g. Grid-Flow)
- UML: activity diagrams (e.g. Askalon)
- CommonWL
Common Workflow Language

M. R. Crusoe
Workflow Execution Models

• Control flow model
• Data-flow model
Mapping workflows to resources

• Understand resource constraints
• Allow user to define mapping
• Use scheduler (e.g. DAGMan) to map
• Optimizations possible…
Execution Concerns

- Single machine versus grid, cloud…
- Fault tolerance
- Adaptive workflow
Pegasus

R. F. da Silva
Workflows and Reproducibility

• How do workflows aid with reproducibility?
• How do workflows aid with reuse?
• What issues do workflows present with reproducibility?
• What about interoperability?