PREPRINT of: Datenbank Spektrum (2012) 12:193–203.
DOI 10.1007/s13222-012-0100-z

Scientific Workflows and Provenance: Introduction and Research Opportunities

Víctor Cuevas-Vicenttín · Saumen Dey · Sven Köhler · Sean Riddle · Bertram Ludäscher

16 July 2012 / Accepted: 25 July 2012 / Published online (Springer): 3 October 2012

Abstract Scientific workflows are becoming increasingly popular for compute-intensive and data-intensive scientific applications. The vision and promise of scientific workflows includes rapid, easy workflow design, reuse, scalable execution, and other advantages, e.g., to facilitate "reproducible science" through provenance (e.g., data lineage) support. However, as described in the paper, important research challenges remain. While the database community has studied (business) workflow technologies extensively in the past, most current work in scientific workflows seems to be done outside of the database community, e.g., by practitioners and researchers in the computational sciences and eScience. We provide a brief introduction to scientific workflows and provenance, and identify areas and problems that suggest new opportunities for database research.

Keywords Scientific workflows · Provenance

1 Introduction

A scientific workflow is a description of a process for accomplishing a scientific objective, usually expressed in terms of tasks and their dependencies [76,59,25]. Scientific workflows aim to accelerate scientific discovery in various ways, e.g., by providing workflow ***A****utomation*, ***S****caling*, ***A****daptation* (for reuse), and ***P****rovenance* support; or ASAP for short. For the automated execution of repetitive tasks, e.g., *batch processing* a set of files in a source directory to produce a set of output files in a target directory, shell scripts are traditionally used. Common processing examples include data (re-)formatting, subsetting, cleaning, analysis, etc. Compute-intensive workflows often result from computational science *simulations*, e.g., running climate and ocean models, or other simulations ranging from particle-physics, chemistry, biology, ecology, to astronomy, and cosmology [58]. Scientific workflows can be simple, linear chains of tasks, but more complex graph-structured dependencies are also common; e.g., one can think of tools like `Make` and `Ant` as special cases of workflow automation [6].

Workflows need to be *scalable* and *fault-tolerant* to execute reasonably fast in the presence of compute-intensive tasks and "Big Data". For example, a *parameter sweep* experiment may require running a program thousands of times with slightly altered input parameters, thus consuming many compute cycles and producing so much data that manually managing it quickly becomes impossible. Distributed Grid or cloud computing technologies [24,86,94] and other parallel frameworks such as MapReduce [23] and dataflow process networks [54,3,82] can be used to scale workflow execution by exploiting parallel resources.

In addition to optimizing compute cycles, workflows should also save costly "human cycles", e.g., by supporting user-friendly workflow designs that are easy to *adapt and reuse*. Similarly, scientific workflows should encourage modeling at an appropriate level of granularity, so that the underlying experimental process is effectively documented, thus improving communication and collaboration between scientists, and experimental reproducibility. Many current systems allow to record the *provenance* (i.e., processing history and lineage) of results, and to monitor runtime execution. Provenance data can then be queried, analyzed, and visualized to gain a deeper understanding of the results, or simply to "debug" a workflow or dataset by tracing its lineage back in time through the workflow execution.

Dept. of Computer Science, University of California, Davis
E-mail: ludaesch@ucdavis.edu

In the following, we introduce and describe different aspects and research challenges of scientific workflows (Section 2) and provenance (Section 3). We provide a short summary and conclusions in Section 4.

2 Scientific Workflows

We give a brief overview of various aspects of scientific workflows, and then summarize related research challenges. Our terminology and perspective are influenced by our work on Kepler [57], a scientific workflow system built on top of Ptolemy II [33].

2.1 Workflow Models of Computation (MoCs)

Figure 1 depicts a simple example workflow in the form of a dataflow process network [50,53]. Boxes represent *actors* (computational steps) which consume and/or produce *tokens* (data) that are sent over uni-directional *channels* (FIFO queues). In a process network, actors usually execute as independent, continuous processes, driven by the availability of input tokens. Alternatively, in some variant models, actors may be *invoked* by a *director* (a kind of scheduler) which coordinates overall execution. Thus, different models of computation (MoCs) can be implemented via different directors [33].

Let W be a workflow consisting of actors connected through dataflow channels.[1] With W we can associate a set of *parameters* $\bar{p}$, input datasets $\bar{x}$, and output datasets $\bar{y}$ (not shown in Fig. 1). A *model of computation* (MoC) M prescribes how to execute the parameterized workflow $W_{\bar{p}}$ on $\bar{x}$ to obtain $\bar{y}$. Therefore, we can view a MoC as a mapping $\mathsf{M} : \mathcal{W} \times \bar{P} \times \bar{X} \to \bar{Y}$ which for any workflow $W \in \mathcal{W}$, parameter settings $\bar{p} \in \bar{P}$, and inputs $\bar{x} \in \bar{X}$, determines a workflow output $\bar{y} \in \bar{Y}$, i.e., $\bar{y} = \mathsf{M}(W_{\bar{p}}(\bar{x}))$. Most workflow systems employ a single MoC, while Kepler inherits from Ptolemy II several of them (and also adds new ones).

The `PN` (process network) MoC, e.g., is modeled after Kahn process networks [50], where each actor executes as an independent, data-driven process. Thus, channels in `PN` correspond to *unbounded* queues over which ordered token streams are sent, and actors in `PN` *block* (wait) only on read operations, i.e., when not enough tokens are available on input ports. Process networks naturally support *pipeline-parallelism* as well as *task-* and *data-parallelism*. In the `SDF` (synchronous dataflow) model, each actor has fixed token consumption and production rates. This allows the director to construct a *firing schedule* prior to executing the workflow, to replace unbounded queues by buffers of fixed size, and to execute workflows in a single thread, invoking actors according to the static schedule [54].

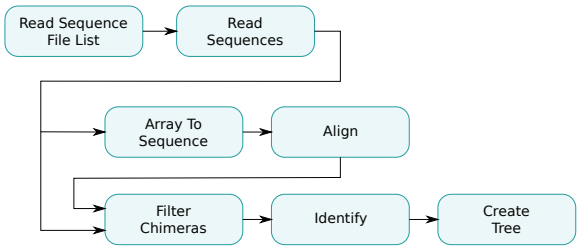

Fig. 1 A simplified bioinformatics workflow that reads a list of lists of genetic sequences: each sequence is aligned; each sequence group is then checked for chimeras, which are filtered out; the rest of the group is passed through. The remaining aligned sequences are identified via a reference database, and all identified sequences are arranged into a phylogenetic tree.

Finally, let `DAG` be a MoC that restricts the workflow graph W to a directed, *acyclic* graph of task dependencies, e.g., as in Condor's DAGMan [78]. In the `DAG` MoC, each actor node $A \in W$ is executed only once, and A is executed only after all $A' \in W$ preceding A (denoted $A' \prec_W A$) have *finished* their execution. We make no assumption whether W is executed sequentially or task-parallel, but only require that any `DAG`-compatible schedule for W satisfy the partial order $\prec_W$ induced by W. A `DAG` director can obtain the legal schedules of $\prec_W$ via a topological sort of W. Finally, note that the `DAG` model can easily support task- and data-parallelism, but not pipeline-parallelism.

Research Issues. What are suitable MoCs that best satisfy the different requirements of scientific workflows? Apart from the basic dataflow MoCs above, there are many other formalisms that could be used as foundational models for scientific workflows. For example, for the Taverna system [67] a workflow model equivalent to the λ-calculus is described in [80]. For business workflows, Petri nets have been used as a rich and solid foundation, with many theoretical results and practical analysis tools readily available. It is interesting to study to what extent scientific workflows could employ the models and analysis tools developed for business workflows. Curcin & Ghanem [20] ask whether a single system (or a single MoC in our terminology) can cover the requirements of different domains, and conclude

> *"... it is highly unlikely that standardization will occur on any one system, as it did with BPEL in the business process domain."*

Indeed, the control-oriented modeling common in business process management, and dataflow-oriented mod-

[1] Here we ignore a number of details, e.g., actor *ports*, sub-workflows "hidden" within so-called *composite actors*, etc.

eling in scientific workflows reflect different ways of thinking about workflows [60,75].

Different formalisms also imply different modeling and analysis opportunities. In databases, e.g., the relational algebra, relational calculus, and cost-based models yield algebraic, semantic, and cost-based optimization techniques, respectively (pushing selections, query minimization, join-ordering, etc.) Petri net models, on the other hand, allow detailed analysis of concurrent execution behavior, e.g., properties like reachability, liveness, and boundedness. Dataflow networks are amenable to behavioral analysis and verification [54,40,53].

In summary, the quest for suitable models of computation, e.g., to adequately represent computations and to expose and exploit different forms of parallelism, continues. A possible direction are hybrid models [44,88], which combine techniques from databases, concurrency models, and stream-processing.

2.2 Workflow Execution

As mentioned in the introduction, workflows need to be scalable to handle compute-intensive and big data loads. Both implicit and explicit approaches have been used to distribute and parallelize workflow execution. Consider, e.g., MapReduce [23] and its popular open source implementation Hadoop. In [85] an approach is described which allows particular Kepler actors to be distributed onto a Hadoop cluster, i.e., the workflow engine is used for orchestration, but is itself not distributed. On the other hand, Nimrod/k [3] (also built on top of Kepler) uses an implicit technique that allows multiple invocations of an actor to execute simultaneously on parallel resources. The approach requires no special configuration to use, but assumes that actors do not maintain state between invocations. Vrba *et al.* [81] propose to use Kahn process networks directly to model parallel applications, and argue that this MoC is a flexible alternative to MapReduce. They also report efficiency gains of their framework when comparted to Phoenix, a MapReduce framework specifically optimized for executing on multicore machines.

Recently, the availability of cloud computing has offered new computational resources to many fields in science. Wang & Altintas [84] report on early experiences with the integration of cloud management and services into a scientific workflow system; Zinn *et al.* [94] propose an approach to support streaming workflows across desktop and cloud platforms.

Research Issues. The problem of efficient, scalable workflow execution is intricately linked to the underlying workflow MoC: the more parallelism is exposed by a workflow language and MoC, the more opportunities there are for exploiting it. An algebraic approach for workflow optimization, well-suited for parameter sweeps, is presented in [72]. In essence data are represented by relations, while actors are mapped to operators that either invoke a program or evaluate a relational algebra expression. The semantics of the operators enables workflow optimization by means of rewriting. Analysis and optimization of dataflow process networks [54, 53] and approaches that combine dataflow, MapReduce, and other parallel techniques with database technologies are also promising [74,79,87,88,10]. Last not least, the reemergence of Datalog in real-world, distributed, and workflow applications [43,46,2] presents unique opportunities for database researchers interested in workflows and provenance [27].

2.3 Workflow Design

Scientific workflow design shares characteristics with component-based development, serviced-oriented design, and scripting, in that preexisting software components (viewed as *black boxes*) and services are "glued", i.e., wired together to form larger applications. In order to save human cycles, scientific workflow design should be easy and fast, and ideally feel more like storytelling and less like programming. Abstractions like "boxes-and-arrows" and flow-charts are often used to develop graphical versions of workflows in a GUI, or to visualize workflow designs, even if they have been specified textually. Depending on the workflow model, different graph formalisms might be used. The simplest designs can be thought of as linear sequences of processing steps, possibly with a stream-processing model like a UNIX pipe. On the other end of the spectrum are complex workflow graphs that can be nested, involve feedback loops, special control-flow elements, etc. For example, Taverna workflows can be nested, have dataflow and control-flow edges, and support a streaming MoC. In addition, Kepler workflows can also have cycles, e.g., to model feedback-loops, or even combine different MoCs when nesting workflows: a top-level `PN` workflow, e.g., may have `SDF` subworkflows [42]. While there can be practical reasons to employ such sophisticated models [73], complex workflow structures can make it more difficult to adapt and reuse workflows.[2]

When independently developed, third-party tools and services are combined into workflows and science

[2] Similarly, in business process modeling, more abstract models, e.g., BPMN, and simple, structured models (e.g., series-parallel graphs) can be easier to understand and reuse than unstructured or lower-level models, e.g., Petri nets.

mashups [45], the resulting designs can be "messy" and may involve complex wiring structures and various forms of software *shims*[3], i.e., adapters that transform data so that (part of) the output of one step is made to fit as an input for another step [48]. According to [56], a study of 560 scientific workflows from the myExperiment repository [22] showed that over 30% of workflow tasks are shims. The proliferation of shims and complex wiring is a limiting factor to wider workflow adoption and reuse. Such designs are hard to understand (shims distract from the "real science") and difficult to maintain: shims and complex wiring make designs "brittle", i.e., sensitive to changes in data structures and workflow steps. Complex designs also limit the use of workflows for documenting and communicating the ideas of the experiment, and require expert developers with specialized programming skills, thus increasing the cost for workflow development and maintenance.

BioMoby [19] aims to improve workflow design by annotating inputs and outputs of actors with semantic type information and enabling the system to provide a wide range of common conversions automatically and transparently. The approach relies on domain experts to create semantic tags and the appropriate conversions. Wings [41] is another system that uses semantic representations to automate various aspects of workflow generation. The approach described in [11,12] facilitates service composition and thus scientific workflow design by exploiting structural types, semantic types, and schema constraints between them. Under certain assumptions, schema constraints can be used to automatically generate the desired schema mappings [35], allowing scientists to more easily connect workflow components [11]. The "templates and frames" approach of [71] aims at simplifying workflow design via a structured composition of control-flow and dataflow.

Research Issues. How can we further simplify workflow design, make workflows less brittle w.r.t. change, and thus more easily evolvable and reusable? How can workflow development become more like storytelling and less like programming? The COMAD model (*Collection-Oriented Modeling and Design*), implemented as a Kepler director [61,62,32], and the related VDAL model (*Virtual Data Assembly Lines*) [92,91] use a conveyor-belt assembly-line metaphor to create workflow designs that are mostly linear, and thus easier to understand and modify. Instead of using shims and complex wiring to ship just the required data fragments to only those actors where they are immediately needed, the conveyor-belt approach instead provides every actor with a subscription mechanism to "pick-up" only the relevant parts

[3] A physical shim is a thin strip of metal for aligning pipes.

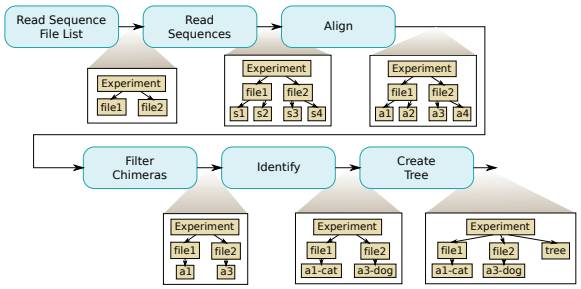

Fig. 2 Workflow redesign from Fig. 1 with COMAD/VDAL. Actors can modify parts of the shared data collections that are streamed through the pipeline; Read Sequences, e.g., processes each file entry, replacing it with a collection of the same name containing all sequences found in the file.

of the nested (XML-like) data stream for processing. The resulting data is added back to the data stream on the fly. Every actor thus gets a chance to work on those parts of the data stream that it has subscribed to, leaving other parts untouched. In this way, changes to those other parts, i.e., outside an actor's *read scope*, will not affect that actor's functionality, making the overall approach much more change resilient. As a result, even more so than on a physical assembly line, steps can be easily added, swapped, replaced, or removed in these approaches [62,91,32]: For example, Figure 2 shows a COMAD redesign of the workflow from Fig. 1. Note that the shim actor ArrayToSequence[4] is no longer needed, as the system takes care of the shim problem (a type mismatch, requiring a `map`-like iteration) using a mechanism based on XPath-like actor configurations (read and write scopes). Similarly, the branching has been eliminated, since Align passes data through to FilterChimeras, resulting in a simpler, more user-friendly design.

As another example, in a *curation workflow* [31] data and metadata undergo various quality checks (e.g., *Do lat/long-coordinates agree with geo-locations? Is the location spelled right? Are the dates plausible?*), and subsequent clean-up steps: Here, adding, removing, or replacing an actor will not "break" the workflow, but only change its curation behavior gracefully.

In contrast, in conventional workflows or scripts, complex wiring and control-flow usually prevent simple actor insertions, replacements, or deletions, and workflow changes are much more difficult and error-prone.

These techniques are initial steps towards scientific workflow design for "mere mortals" [62], but more work is needed. In COMAD, e.g., users have to configure actors by devising simple queries in the style of XPath or XQuery. Further advances would aim, e.g., at improv-

[4] This shim actor turns a data array token into a sequence of individual data tokens.

ing data and schema-level support in workflows. Based on structural and semantic information one could develop an "auto-config" option for assembly-line workflow designs, where not only shims get absorbed by the system infrastructure, but where actor scope configurations (queries) are automatically inferred. The user-friendly, linear designs currently result in a trade-off between human and compute cycles: all data flows through all "stations" (actors) in this workflow model. New analysis techniques could be developed that keep workflow user-views simple, while optimizing executable workflow plans [93]. Workflow design technology might also benefit from functional programming ideas, e.g., concatenative, point-free programming and *arrows* [47].

2.4 Workflow Management and Reuse

Workflow reuse can happen at multiple levels: a scientist may reuse a workflow with different parameters and data, or may modify a workflow to refine the method; workflows can be shared with other scientists conducting similar work, so they provide a means of codifying, sharing, and thus spreading the workflow designer's practice. A prominent scientific workflow repository is myExperiment [22], which aims to "make it easy to find, use and share scientific workflows and other Research Objects, and to build communities"; it currently has more than 5000 members in 250 groups, and a share of over 2000 workflows.[5] There is also some work on business process model repositories [90], but the scientific community is more likely to share workflows as a means to accelerate scientific knowledge discovery, whereas the incentive to publicly share business processes or ETL workflows is limited by commercial interests [18].

Research Issues. Cohen-Boulakia & Leser argue that *"a wider adoption of SciWFM will only be achieved if the focus of research and development is shifted from methods for developing and running workflows to searching, adapting, and reusing existing workflows"* [18]. They propose a number of research directions and problems to better support users of workflows and workflow repositories; e.g., new ways to describe what users search (*workflow sketches*); search for similar (sub-)workflows; and searching and querying of workflow runs and provenance. The construction of workflow repositories also entails the development of techniques to deal with the heterogeneity of workflow specifications and metadata, considering aspects such as versioning, view management, configuration, and context management. The development of specific languages for scientific workflows might again borrow ideas from similar efforts for business processes. BPQL [26], e.g., enables queries over the structure of a process as well as temporal queries addressing their potential behavior; a graph matching algorithm for workflow similarity search is described in [29]. Given the different nature and structure of business and scientific workflows [60], it is interesting to study how techniques from one area might be adapted or extended for the other.

[5] as of July 2012; see http://www.myexperiment.org

3 Provenance and Scientific Workflows

Provenance is information about the origin, context, derivation, ownership, or history of some artifact [17]. In the context of scientific workflows a suitable *model of provenance* (MoP) should be based on the underlying model of computation (MoC). We can derive a MoP from a MoC by taking into account the assumptions that a MoC entails, and by recording the *observables* it affords. In this way, a MoP captures or at least better approximates real data dependencies for workflows with advanced modeling constructs. A provenance trace T can be built from a workflow run object R, by ignoring irrelevant observables I, and adding non-functional observables M that are deemed relevant. With slight abuse of notation, we can say that $T = R - I + M$, i.e., a trace T (a MoP object), is a "trimmed" workflow run R (a MoC object), for which some observables are ignored (I) while others (additional metadata) are modeled (M). More formally, the implementation of a MoC M of a scientific workflow system defines an *operational semantics*, which in turn can be used to define a notion of "natural processing history" or *run* $R_{\bar{x}\rightsquigarrow\bar{y}}$ for $\bar{y} = \mathsf{M}(W_{\bar{p}}(\bar{x}))$. Given a run $R_{\bar{x}\rightsquigarrow\bar{y}}$, the semantics given by M allows us to check whether $R_{\bar{x}\rightsquigarrow\bar{y}}$ is a "faithful" (or *legal*) execution of $\bar{y} = \mathsf{M}(W_{\bar{p}}(\bar{x}))$. For example, the order of actor firings in a legal run $R_{\bar{x}\rightsquigarrow\bar{y}}$ must conform to $\prec_W$; and $\bar{p}, \bar{x}$, and $\bar{y}$ must appear in $R_{\bar{x}\rightsquigarrow\bar{y}}$ as inputs and outputs, respectively.

Observables. The records of a run $R_{\bar{x}\rightsquigarrow\bar{y}}$ of a workflow execution of $\bar{y} = M(W_{\bar{p}}(\bar{x}))$ are built from basic *observables* associated with M. For $\mathsf{M} = \texttt{DAG}$, the observables of a run are the single *firings* of an actor A, together with the inputs d and outputs d' of the firing, recorded as $\mathsf{fired}(d, A, d')$. We may dissect firing events into smaller observables, e.g., into two records of the form $\mathsf{received}(A, m_1(d))$ and $\mathsf{sent}(A, m_2(d'))$, and a third record of the form $\mathsf{caused}(m_1, m_2)$. This can be useful for MoCs such as CSP (*Communicating Sequential Processes*) or MPI (*Message Passing Interface*), where actors react to different messages, not just the read (input/token consumption) and write (output/token pro-

duction) that we used here for DAG. Similarly, job-based (or *Grid*) workflow systems are usually based on the DAG MoC. For these, fired(d, A, d') may be modeled differently, e.g., based on a job's start and finish events.

3.1 Models of Provenance (MoPs)

The models for provenance used in scientific workflow systems include custom models such as the RWS (read, write, state-reset) model [13], and community efforts such as the Open Provenance Model (OPM) [68]. OPM was developed as a minimal, generic standard for provenance, not just for workflows. In OPM, events are recorded when actors consume tokens (*used* events) and produce tokens (*wasGeneratedBy* events). Thus, storing provenance data can effectively make persistent the data tokens that have been flowing across workflow channels, either through the actual data or by handlers.

Event records typically include an identifier for the actor (e.g., a service or program). The specific port or parameter of the actor associated with the data token can also be recorded, which corresponds to the *role* in OPM's *usedBy* and *generatedBy* events. Each event is generally recorded along with its timestamp. While OPM only considers processes, additional information can be recorded like the notion of state-reset events and rounds in RWS, which define logical units of work. The collection of event records defines a directed acyclic graph (DAG) that represents the execution history.

The OPM also defines *wasTriggeredBy* and *wasDerivedFrom* relationships. The former denotes that an instance of an actor execution is causally linked to a preceding actor execution, while the later denotes that a data artifact results (at least partially) from processing an earlier artifact. Under an extended interpretation of the OPM *used* and *wasGeneratedBy* relations, *wasTriggeredBy* and *wasDerivedFrom* relations can be inferred, which is not always the case in terms of the original standard. The *agents* in the OPM, correspond in scientific workflow systems either to software entities that execute workflow components, or to users that initiate and monitor the execution of a workflow. Both cases can be represented in OPM by the *wasControlledBy* relation.

Research Issues. While OPM and its W3C successor PROV are gaining popularity, by design they leave out specifics of MoCs and custom MoPs. Elements specific to workflows are not present, like firing constraints [27] or the workflow structure. OPM's temporal semantics is somewhat ambiguous as pointed out in [69]. An interesting area of research is the development of richer provenance models, corresponding to different workflows systems and MoCs. For example, the DataONE[6] Provenance Working Group develops a unified MoP reflecting scientific workflows provenance from different systems (initially: Kepler, Taverna, VisTrails, and R).

3.2 Capturing Provenance

Provenance in workflows is not limited to the execution of a fixed workflow, but can also include the history of the workflow design, i.e., *workflow evolution provenance*. VisTrails [37] keeps track of all the changes that have led to new versions of a given workflow. Formally, a *vistrail* is a tree in which nodes represent workflow versions and edges correspond to operations in a change algebra, such as *addConnection* or *addModule*.

Difficulties in capturing provenance arise in practice, as scientific workflow systems are built on and interoperate with other systems, e.g., databases, parallel computing platforms, web services, scripting languages, etc. Provenance data originating from lower-level components needs to be made accessible to the workflow system, e.g., resource usage statistics, failures, and repeated execution attempts in parallel programs. Swift [39] captures such information from high level SwiftScript programs by means of wrapping scripts running in the background. Each level of abstraction associated with a software layer may include different provenance observables: e.g., the PASS system supports provenance at the file system, workflow engine, script language, and browser levels [70].

Research Issues. It is desirable to keep the overhead for capturing provenance to a minimum. An interesting possibility is to study declarative, domain-specific language for provenance in computing systems. This will allow the user to define relevant provenance information at the desired granularity.

With the use of cloud computing and key-value stores for scientific computing, it becomes challenging to capture the necessary provenance data, since these platforms can only be accessed through interfaces.[7] With respect to recording overhead, data poses the greatest challenge, since in some cases the size of provenance data can exceed the combined input and output data [15]. Thus, with large-scale data it is essential to identify the information to record and to employ efficient data management techniques.

[6] http://www.dataone.org/

[7] See, for example, Amazon's Simple Storage Service (S3) http://aws.amazon.com and Simple Workflow Service (SWS)

3.3 Storage and Querying

Scientists want to use provenance data to answer questions such as: *Which data items were involved in the generation of a given partial result?* or *Did this actor employ outputs from one of these two other actors?* Such questions addressed over the provenance data represented by a directed graph translate into two well-known types of directed graph queries: reachability and regular path queries.

A *reachability query* over a directed graph G returns pairs (x, y) of nodes, connected by a path in G. The paths themselves are often not computed. In addition to simple graph-traversal and transitive closure algorithms, specialized techniques can be used, e.g., using simpler structures such as chains or trees to compute and compress the transitive closure. In [49], a *path-tree* is introduced along with additional techniques to yield a more efficient solution. An approach specifically for provenance graphs is introduced in [8]: a labeling scheme captures parallel instantiations (forks) along with loops; labels are of logarithmic length and generated in linear time.

Reachability queries can be computed by RDBMSs with simple extensions. While *transitive closure* is not a first-order query, it is still a *maintainable* relation through first-order (plus aggregation for some cases) queries [30]. This database technique is used, e.g., in the Swift system [39]. A recent extension of Swift [38] now uses a SQL function with a `RECURSIVE` clause.

An extended form of reachability query is presented in [66], where the authors consider fine-grained provenance data resulting from Taverna workflows that also operate on lists. Their approach is based on using the workflow specification as an index. This is particularly useful to reduce search time for focused queries, where users are interested only in selecting the inputs related to one specific output.

A *regular path query* (RPQ) over an edge-labeled, directed graph G returns all pairs of nodes (x, y) which are connected in G via a path π whose labels spell a word that matches a given regular path expression R. Regular path expressions are built similar to regular expressions (e.g., concatenation $R_1 \cdot R_2$, alternation $R_1 | R_2$, Kleene-star R^*, etc.) but can also include graph-specific extensions such as R^{-1}, denoting edge reversal.

The evaluation of RPQs via simple paths (i.e., paths without repeated nodes) is NP-complete [63] in general, but drops to PTIME when allowing non-simple paths as well. The traditional approach to evaluate RPQs is to view the graph as a non-deterministic finite automaton, then construct an equivalent deterministic finite automaton to be used as an (often very large) index. To achieve a more efficient evaluation, a heuristic approach is presented in [52], based on finding rare labels and using them as starting points for bi-directional searches. Each search corresponds to a sub-query, and the combination of results yields the result for the original query. The fact that rare labels are chosen restricts the search significantly and thus the overall computation required.

An alternative way to evaluate RPQs is the translation to Datalog queries. A direct translation is given in [4], whereas [77] provides another, optimized method.

Query languages able to express RPQs include those originally developed for semistructured data integration systems such as STRUDEL [36] and proposed extensions on SPARQL. Such languages and other alternatives are discussed in greater depth in [89]. However, graph queries issued directly against physical data representations (e.g. XML or RDF) can be difficult to express and expensive to evaluate. In [7] QLP, a high-level query language for provenance graph queries is introduced to address these shortcomings. It provides constructs for querying both structure and lineage information complemented with optimization and lineage-graph reduction techniques.

Visual querying is also well suited for provenance data. In particular, VisTrails [37] includes a visual query-by-example interface; additionally, it can perform keyword search on provenance data.

Research Issues. Efficient storage and querying of large amounts of provenance data remain important areas of research, in particular, as more provenance data is being produced and shared. The use of information retrieval techniques in databases has received significant attention (e.g. [55]), their application specifically to provenance data represents another interesting research direction. Provenance data can also be seen as (detailed) variants of event logs of information systems, thus providing interesting new areas and applications of process mining [1]. This approach could allow discovering a workflow from provenance data, monitoring its conformance to the expected behavior, or improving the workflow by identifying, e.g., opportunities for optimization and parallelization.

3.4 Interoperability

Today's scientific experiments are very complex, involving multiple teams working in collaboration and using various scientific workflow systems to design and execute respective workflows. In this collaborative setting, an output data product of a workflow is often used as an

input data product of another workflow. Thus, to completely understand a data product we need its provenance information along with provenance information of all the dependent data products. Thus, we need to integrate all the provenance information to effectively answer all provenance queries.

Heterogeneous MoPs. Most scientific workflow systems provide a method for recording provenance. However, these systems use specific provenance and storage models. For example, Kepler records OPM-based provenance into a relational database, whereas COMAD employs its own provenance model and stores provenance information into an XML file. In order to answer provenance queries, we need to develop methods to integrate provenance information from various MoPs. A mediation-based approach to solve this problem is presented in [34], whereas in [64] a framework and common data model for traces is proposed.

Lack of Shared Data Identification. While working collaboratively, a scientist may copy a data product into his local system and then perform some formatting tasks before using it as an input data product. The copying and formatting tasks may give rise to a different identifier for the same data, as the identification management is being done by individual systems in isolation. Thus, it is important to identify these copies so that they are linked appropriately and correct dependencies are established. Missier *et al.* [65] observe this problem and provide a prototypical foundation toward solving it.

3.5 Provenance Applications

Privacy-aware Provenance. While provenance information is very useful, it often carries sensitive information causing privacy concerns, which can be tied to data, processes, and workflow specifications. It is possible to infer the value of a data product, the functionality (being able to guess the output of an actor given a set of inputs) of a process, or the execution flow of the workflow. In [21] a mathematical foundation to achieve ε-privacy for a process or a workflow is developed. This model is able to compute the input/output combinations for which the required level of ε-privacy is achieved.

SecurityViews [16] are developed in order to provide a partial view of the workflow through a role-based access control mechanism, and by defining a set of access permissions on processes, channels, and input/output ports as specified by the workflow owner at design time.

Two techniques are developed in [28] to customize the provenance information based on scientists privacy and data sharing requirements. This approach adheres to a provenance model in which the scientist wants to share the customized provenance data.

While trying to honor the privacy concerns, current techniques remove the private information without providing clear guarantees of what queries could be answered using the customized provenance. Also, as the provenance information often is very large, it is important to develop techniques so that privacy and publication requirements can be expressed at a higher level (for e.g. workflow specification, data collection, etc).

Information Overload. Provenance data captured by executing a workflow is often larger than the size the actual data [14]. ZOOM*UserViews [9] provides a partial, zoomed-out view of a workflow, based on the user-defined distinction between relevant and irrelevant actors. Provenance information is captured based on this view. Another approach in [28] does not impose any restriction in capturing provenance information. It allows scientists to specify the customization requests on the provenance information to remove the irrelevant parts, which allows scientists to better understand the relevant parts of the provenance information.

The volume explosion of provenance information gives rise to two very challenging issues: how to effectively and efficiently (i) visualize and browse provenance information under different levels of abstraction, and (ii) specify what are the relevant portions of the provenance information.

Debugging. Provenance recording also enables a natural way of debugging workflows. For instance, data values can easily be inspected and checked for correctness. In addition, by comparing a workflow description with a trace of its run, actors that never fire can easily be detected. For this purpose, a static analysis technique is described in [95] to infer an abstract provenance graph (APG) from a VDAL style workflow description described earlier. The APG allows identifying incorrect configurations and actors that are never fired.

An interesting research issue is how to use provenance data potentially including time-stamps to analyze the efficiency of a workflow execution. Independent subworkflow executions would be easily identifiable from data dependencies in a trace graph. Those independent subworkflows should optimally be executed in parallel and time-stamps in the trace would allow finding places with suboptimal scheduling.

Fault Tolerance. Similarly to database recovery using log files, provenance can be used to efficiently recover faulty workflow executions as shown in [51]. The trace

contains all data items processed and created before the fault as well as the status of invocations at the time of the fault. Thus, successfully completed invocations can be skipped and data that was in the intra-actor queues at the time of the fault is restored. The challenges in developing recovery techniques are that (1) actors can be invoked multiple times and maintain state from one invocation to the next; (2) data can be transported between actors outside of the queue-based infrastructure, circumventing provenance recording; (3) non-trivial scheduling algorithms are used for multiple actor invocations, which are based on data availability.

Some MoCs such as COMAD use complex data structures which are exchanged between actor invocations in fragments and therefore implicit dependencies between data artifacts exist that have to be maintained. This requires special consideration during the recovery process. Furthermore, the stateful layer in each actor handling the scope matching requires new techniques in order to allow an efficient recovery.

4 Summary and Conclusions

Scientific workflow systems can help scientists design and execute computational experiments efficiently, but many research issues remain to be solved. We have given an introduction and overview on scientific workflows and provenance and highlighted a number of research areas and problems. *Business* workflows (and business process modeling) have been and are being studied extensively by the database community. With this paper we hope to help trigger or reignite interest in the database community to address some of the challenges in *scientific* workflows. After all, database researchers were among the first to explore the challenges in scientific workflows and to apply database technologies towards them [83,5]. In the era of data-driven scientific discovery and Big Data, there was probably never a better time for researchers to embrace the challenges and opportunities in scientific workflows and to advance the state-of-the-art in data-intensive computing.

Acknowledgements Work supported in part by NSF awards OCI-0830944, OCI-0722079, DGE-0841297, and DBI-0960535.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st edn. Springer (2011) 7
2. Abiteboul, S., Bienvenu, M., Galland, A., Rousset, M.: Distributed datalog revisited. Datalog Reloaded pp. 252–261 (2011) 3
3. Abramson, D., Enticott, C., Altinas, I.: Nimrod/K: Towards Massively Parallel Dynamic Grid Workflows. In: Supercomputing Conference. IEEE (2008) 1, 3
4. Afrati, F., Toni, F.: Chain queries expressible by linear datalog programs. In: Deductive Databases and Logic Programming (DDLP), pp. 49–58 (1997) 7
5. Ailamaki, A., Ioannidis, Y., Livny, M.: Scientific workflow management by database management. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 190–199 (1998) 9
6. Amin, K., von Laszewski, G., Hategan, M., Zaluzec, N., Hampton, S., Rossi, A.: GridAnt: A Client-Controllable Grid Workflow System. In: Hawaii Intl. Conf. on System Sciences (HICSS). IEEE (2004) 1
7. Anand, M.K., Bowers, S., Ludäscher, B.: Techniques for efficiently querying scientific workflow provenance graphs. In: Proceedings of the 13th International Conference on Extending Database Technology, EDBT '10, pp. 287–298. ACM, New York, NY, USA (2010) 7
8. Bao, Z., Davidson, S.B., Khanna, S., Roy, S.: An optimal labeling scheme for workflow provenance using skeleton labels. In: SIGMOD, pp. 711–722 (2010) 7
9. Biton, O., Cohen-Boulakia, S., Davidson, S.: Zoom* userviews: Querying relevant provenance in workflow systems. In: VLDB, pp. 1366–1369 (2007) 8
10. Borkar, V., Carey, M., Grover, R., Onose, N., Vernica, R.: Hyracks: A flexible and extensible foundation for data-intensive computing. In: Intl. Conf. on Data Engineering (ICDE) (2011) 3
11. Bowers, S., Ludäscher, B.: An ontology-driven framework for data transformation in scientific workflows. In: Data Integration in the Life Sciences (DILS), pp. 1–16 (2004) 4
12. Bowers, S., Ludäscher, B.: Actor-oriented design of scientific workflows. Conceptual Modeling (ER) pp. 369–384 (2005) 4
13. Bowers, S., McPhillips, T., Ludäscher, B., Cohen, S., Davidson, S.B.: A model for user-oriented data provenance in pipelined scientific workflows. In: Intl. Provenance and Annotation Workshop (IPAW) (2006) 6
14. Braun, U., Garfinkel, S., Holland, D., Muniswamy-Reddy, K., Seltzer, M.: Issues in automatic provenance collection. Provenance and annotation of data pp. 171–183 (2006) 8
15. Chapman, A.P., Jagadish, H.V., Ramanan, P.: Efficient provenance storage. In: SIGMOD, pp. 993–1006 (2008) 6
16. Chebotko, A., Chang, S., Lu, S., Fotouhi, F., Yang, P.: Scientific workflow provenance querying with security views. In: Web-Age Information Management (WAIM), pp. 349–356 (2008) 8
17. Cheney, J., Finkelstein, A., Ludäscher, B., Vansummeren, S.: Principles of Provenance (Dagstuhl Seminar 12091). Dagstuhl Reports **2**(2), 84–113 (2012). http://dx.doi.org/10.4230/DagRep.2.2.84 5
18. Cohen-Boulakia, S., Leser, U.: Search, adapt, and reuse: the future of scientific workflows. ACM SIGMOD Record **40**(2), 6–16 (2011) 5
19. Consortium, T.B.: Interoperability with Moby 1.0—It's better than sharing your toothbrush! Briefings in Bioinformatics **9**(3), 220–231 (2008) 4
20. Curcin, V., Ghanem, M.: Scientific workflow systems—can one size fit all? In: Biomedical Engineering Conference (CIBEC) (2008) 2
21. Davidson, S., Khanna, S., Roy, S., Boulakia, S.: Privacy issues in scientific workflow provenance. In: Intl. Workshop on Workflow Approaches to New Data-centric Science (2010) 8

22. De Roure, D., Goble, C., Stevens, R.: The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows. Future Generation Computer Systems **25**(5), 561–567 (2009) 4, 5
23. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. CACM **51**(1), 107–113 (2008) 1, 3
24. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K., Livny, M.: Pegasus: Mapping scientific workflows onto the grid. In: Grid Computing, pp. 131–140. Springer (2004) 1
25. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems **25**(5), 528–540 (2009) 1
26. Deutch, D., Milo, T.: A structural/temporal query language for Business Processes. J. Comput. Syst. Sci. **78**(2), 583–609 (2012) 5
27. Dey, S., Köhler, S., Bowers, S., Ludäscher, B.: Datalog as a Lingua Franca for Provenance Querying and Reasoning. In: Workshop on the Theory and Practice of Provenance (TaPP) (2012) 3, 6
28. Dey, S., Zinn, D., Ludäscher, B.: PROPUB: Towards a Declarative Approach for Publishing Customized, Policy-Aware Provenance. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM) (2011) 8
29. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Intl Conf. on Business Process Management (BPM), pp. 48–63 (2009) 5
30. Dong, G., Libkin, L., Su, J., Wong, L.: Maintaining Transitive Closure of Graphs in SQL. Int. J. Information Technology **5** (1999) 7
31. Dou, L., Cao, G., Morris, P.J., Morris, R.A., Ludäscher, B., Macklin, J.A., Hanken, J.: Kurator: A Kepler Package for Data Curation Workflows. Procedia CS **9**, 1614–1619 (2012). Demo video at `http://youtu.be/DEkPbvLsud0` 4
32. Dou, L., Zinn, D., McPhillips, T.M., Köhler, S., Riddle, S., Bowers, S., Ludäscher, B.: Scientific workflow design 2.0: Demonstrating streaming data collections in Kepler. In: Intl. Conf. on Data Engineering (ICDE) (2011) 4
33. Eker, J., Janneck, J., Lee, E.A., Liu, J., Liu, X., Ludvig, J., Sachs, S., Xiong, Y.: Taming heterogeneity - the Ptolemy approach. Proceedings of the IEEE **91**(1), 127–144 (2003) 2
34. Ellqvist, T., Koop, D., Freire, J., Silva, C., Stromback, L.: Using mediation to achieve provenance interoperability. In: Services-I, 2009 World Conference on, pp. 291–298. IEEE (2009) 8
35. Fagin, R., Haas, L., Hernández, M., Miller, R., Popa, L., Velegrakis, Y.: Clio: Schema mapping creation and data exchange. Conceptual Modeling: Foundations and Applications pp. 198–236 (2009) 4
36. Fernández, M., Florescu, D., Levy, A., Suciu, D.: Declarative specification of Web sites with S. The VLDB Journal **9**(1), 38–55 (2000) 7
37. Freire, J., Silva, C.T., Callahan, S.P., Santos, E., Scheidegger, C.E., Vo, H.T.: Managing rapidly-evolving scientific workflows. In: Intl. Annotation and Provenance Workshop (IPAW), pp. 10–18 (2006) 6, 7
38. Gadelha, L., Mattoso, M., Wilde, M., Foster, I.: Provenance query patterns for Many-Task scientific computing. Workshop on the Theory and Practice of Provenance. Heraklion, Greece pp. 1–6 (2011) 7
39. Gadelha Jr., L.M.R., Clifford, B., Mattoso, M., Wilde, M., Foster, I.: Provenance management in Swift. Future Gener. Comput. Syst. **27**(6), 775–780 (2011) 6, 7
40. Geilen, M., Basten, T.: Requirements on the execution of Kahn process networks. Programming Languages and Systems pp. 319–334 (2003) 3
41. Gil, Y., Ratnakar, V., Deelman, E., Mehta, G., Kim, J.: Wings for Pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In: National Conference on Artificial Intelligence, vol. 22 (2007) 4
42. Goderis, A., Brooks, C., Altintas, I., Lee, E.A., Goble, C.A.: Composing Different Models of Computation in Kepler and Ptolemy II. In: Intl. Conf. on Computational Science (2007) 3
43. Hellerstein, J.: The declarative imperative: experiences and conjectures in distributed logic. SIGMOD Record **39**(1), 5–19 (2010) 3
44. Hidders, J., Kwasnikowska, N., Sroka, J., Tyszkiewicz, J., Van den Bussche, J.: DFL: A dataflow language based on Petri nets and nested relational calculus. Information Systems **33**(3), 261–284 (2008) 3
45. Howe, B., Green-Fishback, H., Maier, D.: Scientific Mashups: Runtime-Configurable Data Product Ensembles. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 19–36 (2009) 4
46. Huang, S., Green, T., Loo, B.: Datalog and emerging applications: an interactive tutorial. In: SIGMOD, pp. 1213–1216 (2011) 3
47. Hughes, J.: Programming with Arrows. In: Intl. Summer School on Advanced Functional Programming, LNCS 3622, pp. 73–129 (2005) 5
48. Hull, D., Stevens, R., Lord, P., Wroe, C., Goble, C.: Treating "shimantic web" syndrome with ontologies. In: First AKT workshop on Semantic Web Services (2004) 4
49. Jin, R., Ruan, N., Xiang, Y., Wang, H.: Path-tree: An efficient reachability indexing scheme for large directed graphs. ACM TODS **36**(1), 7:1–7:44 (2011) 7
50. Kahn, G.: The Semantics of Simple Language for Parallel Programming. In: IFIP Congress, pp. 471–475 (1974) 2
51. Köhler, S., Riddle, S., Zinn, D., McPhillips, T.M., Ludäscher, B.: Improving Workflow Fault Tolerance through Provenance-Based Recovery. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 207–224 (2011) 8
52. Koschmieder, A., Leser, U.: Regular path queries on large graphs. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM) (2012) 7
53. Lee, E.A., Matsikoudis, E.: The Semantics of Dataflow with Firing. In: G. Huet, G. Plotkin, J.J. Lévy, Y. Bertot (eds.) From Semantics to Computer Science: Essays in memory of Gilles Kahn (2008) 2, 3
54. Lee, E.A., Parks, T.M.: Dataflow Process Networks. In: Proceedings of the IEEE, pp. 773–799 (1995) 1, 2, 3
55. Li, G., Feng, J., Zhou, X., Wang, J.: Providing built-in keyword search capabilities in RDBMS. The VLDB Journal **20**(1), 1–19 (2011) 7
56. Lin, C., Lu, S., Fei, X., Pai, D., Hua, J.: A task abstraction and mapping approach to the shimming problem in scientific workflows. In: Services Computing, pp. 284–291. IEEE (2009) 4
57. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice & Experience (CCPE) **18**(10), 1039–1065 (2006) 2
58. Ludäscher, B., Altintas, I., Bowers, S., Cummings, J., Critchlow, T., Deelman, E., Roure, D.D., Freire, J., Goble, C., Jones, M., Klasky, S., McPhillips, T., Podhorszki, N., Silva, C., Taylor, I., Vouk, M.: Scientific

Process Automation and Workflow Management. In: A. Shoshani, D. Rotem (eds.) Scientific Data Management. Chapman & Hall/CRC (2009) 1
59. Ludäscher, B., Bowers, S., McPhillips, T.: Scientific Workflows. In: T. Özsu, L. Liu (eds.) Encyclopedia of Database Systems. Springer (2009) 1
60. Ludäscher, B., Weske, M., McPhillips, T., Bowers, S.: Scientific Workflows: Business as Usual? In: Intl. Conf. on Business Process Management (BPM), pp. 31–47 (2009) 3, 5
61. McPhillips, T., Bowers, S., Ludäscher, B.: Collection-Oriented Scientific Workflows for Integrating and Analyzing Biological Data. In: Intl. Workshop on Data Integration in the Life Sciences (DILS) (2006) 4
62. McPhillips, T., Bowers, S., Zinn, D., Ludäscher, B.: Scientific Workflows for Mere Mortals. Future Generation Computer Systems **25**(5), 541–551 (2009) 4
63. Mendelzon, A.O., Wood, P.T.: Finding Regular Simple Paths in Graph Databases. SIAM J. Comput. **24**(6), 1235–1258 (1995) 7
64. Missier, P., Ludascher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M., Goble, C.: Linking multiple workflow provenance traces for interoperable collaborative science. In: Workflows in Support of Large-Scale Science (WORKS), 2010 5th Workshop on, pp. 1 –8 (2010) 8
65. Missier, P., Ludäscher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M., Goble, C.: Linking multiple workflow provenance traces for interoperable collaborative science. In: Workshop on Workflows in Support of Large-Scale Science (WORKS) (2010) 8
66. Missier, P., Paton, N.W., Belhajjame, K.: Fine-grained and efficient lineage querying of collection-based workflow provenance. In: EDBT, pp. 299–310 (2010) 7
67. Missier, P., Soiland-Reyes, S., Owen, S., Tan, W., Nenadic, A., Dunlop, I., Williams, A., Oinn, T., Goble, C.: Taverna, reloaded. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 471–481 (2010) 2
68. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., den Bussche, J.V.: The Open Provenance Model core specification (v1.1). Future Gener. Comput. Syst. **27**(6), 743–756 (2011) 6
69. Moreau, L., Kwasnikowska, N., den Bussche, J.V.: A Formal Account of the Open Provenance Model. Tech. rep., University of Southampton (2009) 6
70. Muniswamy-Reddy, K.K., Braun, U., Holland, D.A., Macko, P., Maclean, D., Margo, D., Seltzer, M., Smogor, R.: Layering in provenance systems. In: USENIX (2009) 6
71. Ngu, A., Bowers, S., Haasch, N., McPhillips, T., Critchlow, T.: Flexible scientific workflow modeling using frames, templates, and dynamic embedding. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 566–572 (2008) 4
72. Ogasawara, E., De Oliveira, D., Valduriez, P., Dias, D., Porto, F., Mattoso, M.: An Algebraic Approach for Data-Centric Scientific Workflows. Proceedings of VLDB **4**(11), 1328–1339 (2011) 3
73. Podhorszki, N., Ludäscher, B., Klasky, S.A.: Workflow automation for processing plasma fusion simulation data. In: Workflows in Support of Large-Scale Science (WORKS), pp. 35–44 (2007) 3
74. Shankar, S., Kini, A., DeWitt, D., Naughton, J.: Integrating databases and workflow systems. ACM SIGMOD Record **34**(3) (2005) 3
75. Tan, W., Missier, P., Madduri, R., Foster, I.: Service-Oriented Computing — ICSOC 2008 Workshops. chap. Building Scientific Workflow with Taverna and BPEL: A Comparative Study in caGrid, pp. 118–129. Springer-Verlag, Berlin, Heidelberg (2009) 3
76. Taylor, I., Deelman, E., Gannon, D., Shields, M. (eds.): Workflows for e-Science: Scientific Workflows for Grids. Springer (2007) 1
77. Tekle, K.T., Gorbovitski, M., Liu, Y.A.: Graph queries through datalog optimizations. In: Principles and Practice of Declarative Programming (PPDP), pp. 25–34 (2010) 7
78. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: The Condor experience. Concurrency and Computation: Practice & Experience **17**(2-4), 323–356 (2005) 2
79. Thusoo, A., Sarma, J., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. VLDB **2**(2) (2009) 3
80. Turi, D., Missier, P., Goble, C., Roure, D.D., Oinn, T.: Taverna Workflows: Syntax and Semantics. In: Intl. Conf. on e-Science and Grid Computing (2007) 2
81. Vrba, Ž., Halvorsen, P., Griwodz, C., Beskow, P.: Kahn process networks are a flexible alternative to MapReduce. In: High Performance Computing and Communications (HPCC), pp. 154–162 (2009) 3
82. Vrba, Ž., Halvorsen, P., Griwodz, C., Beskow, P., Espeland, H., Johansen, D.: The Nornir run-time system for parallel programs using Kahn process networks on multi-core machinesa flexible alternative to MapReduce. Journal of Supercomputing pp. 1–27 (2010) 1
83. Wainer, J., Weske, M., Vossen, G., Medeiros, C.: Scientific workflow systems. In: NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions. Athens, GA (1996) 9
84. Wang, J., Altintas, I.: Early Cloud Experiences with the Kepler Scientific Workflow System. Procedia CS **9**, 1630–1634 (2012) 3
85. Wang, J., Crawl, D., Altintas, I.: Kepler+Hadoop: A general architecture facilitating data-intensive applications in scientific workflow systems. In: Workshop on Workflows in Support of Large-Scale Science (WORKS) (2009) 3
86. Wieczorek, M., Prodan, R., Fahringer, T.: Scheduling of scientific workflows in the ASKALON grid environment. SIGMOD Record **34**(3), 56–62 (2005) 1
87. Wilde, M., Foster, I., Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B., Raicu, I.: Parallel scripting for applications at the petascale and beyond. IEEE Computer **42**(11), 50–60 (2009) 3
88. Wombacher, A.: Data workflow: a workflow model for continuous data processing (2010). Centre for Telematics and Information Technology, University of Twente 3
89. Wood, P.T.: Query languages for graph databases. SIGMOD Rec. **41**(1), 50–60 (2012) 7
90. Yan, Z., Dijkman, R., Grefen, P.: Business process model repositories - Framework and survey. Inf. Softw. Technol. **54**(4), 380–395 (2012) 5
91. Zinn, D., Bowers, S., Ludäscher, B.: XML-based computation for scientific workflows. In: Intl. Conf. on Data Engineering (ICDE), pp. 812–815. IEEE (2010) 4
92. Zinn, D., Bowers, S., McPhillips, T., Ludäscher, B.: Scientific workflow design with data assembly lines. In: Workshop on Workflows in Support of Large-Scale Science (WORKS) (2009) 4

93. Zinn, D., Bowers, S., McPhillips, T., Ludäscher, B.: X-CSR: Dataflow Optimization for Distributed XML Process Pipelines. In: Intl. Conf. on Data Engineering (ICDE), pp. 577–580 (2009) 5
94. Zinn, D., Hart, Q., McPhillips, T.M., Ludäscher, B., Simmhan, Y., Giakkoupis, M., Prasanna, V.K.: Towards Reliable, Performant Workflows for Streaming-Applications on Cloud Platforms. In: Intl. Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 235–244 (2011) 1, 3
95. Zinn, D., Ludäscher, B.: Abstract Provenance Graphs: Anticipating and Exploiting Schema-Level Data Provenance. In: Intl. Provenance and Annotation Workshop (IPAW), pp. 206–215 (2010) 8