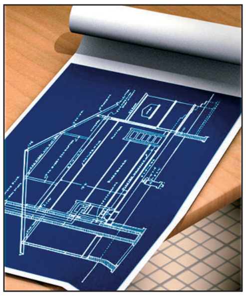

Workflow Planning on a Grid

Craig W. Thompson, Wing Ning Li, and Zhichun Xiao • *University of Arkansas*

Workflows are used in scientific, engineering, industrial, legal, and business applications to process many kinds of data, such as genomes, supply chains, and mailing lists. Workflows that involve thousands to millions of files and up to terabytes of data are a challenge that grid computing is tackling. Another challenge is how to automate the specification of large workflows to avoid human error — AI planning techniques can help. In this column, we sketch how we're addressing these problems with the prototype workflow planning system we're developing at the University of Arkansas.

Workflow Management

The Workflow Management Coalition (WfMC, www.wfmc.org/standards/model.htm) defines *workflow* as "the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules." They define a workflow management system as "a system that defines, creates, and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants, and, where required, invoke the use of IT tools and applications."

At one level of abstraction, we can represent a workflow as a directed graph with operators (or tasks) at the vertices (see Figure 1). Each operator takes inputs from data sources or from the outputs of predecessor operators in the graph, performs some computation, and produces either outputs that feed other operators or final, desired outputs. Large industrial workflows take as inputs tens to millions of files and routinely process terabyte-sized data sets. Operators are often domain- and application-specific.

Logically, a workflow-management system can be implemented as a set of services that hide its implementation details. That is, they can hide whether the operators execute on a single machine or are distributed across several machines, how big the operands are, and how long the processing takes (as long as it's fast enough). Users can implement small workflows on a single machine using home-brew technology, but curious minds want to know how to deal with large workflows.

Although we can use relational database-management systems to process large workflows, their capabilities don't always match the necessary requirements: the system must copy data to the relational DBMS, but generic relational operators don't match domain-specific workflow tasks, and relational systems might be overwhelmed by the throughput requirements for large workflows. When data sets consist of files of independent records, there's considerable opportunity for processing records in parallel, which relational systems might not exploit.

Many organizations are discovering that grid computing provides a better match for their workflow needs. The Open Grid Forum (OGF, www.ogf.org) defines a grid as a "system that is concerned with the integration, virtualization, and management of services and resources in a distributed, heterogeneous environment that supports collections of users and resources (virtual organizations) across traditional administrative and organizational domains (real organizations)" (https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.ogsa-wg/docman.root.working_drafts.glossary/doc13938).

A grid consists of many computers (sometimes thousands) connected by a network. A large file can be distributed across many machines, each of which executes the same workflow operator on different parts of the file in parallel in main memory. As Fig-

 1089-7801/07/$25.00 © 2007 IEEE

ure 2 shows, a workflow management system might execute many workflows simultaneously. Each workflow depends on a collection of workflow operators. These, in turn, depend on a grid layer and possibly on other storage, indexing, security, and system-management operations.

Workflow Planning and Our Prototype

The term *workflow automation* usually refers to automating a collection of tasks, but humans must still manually specify the workflow graph. Some organizations have hundreds of full-time data experts building or modifying workflows. A workflow developer, using a GUI selects operators, arranges them into a graph, and connects output fields of one operator to input fields of another. Manually specifying a correct and efficient workflow isn't easy even for an experienced domain expert — the expert must know detailed information and data dependencies for all operators. The manual specification process is inefficient and error-prone when workflows are large. Invalid or inefficient workflows reduce the grid's overall performance. In spite of the data sets' large size and the massive compute power needed to run large workflows, workflow specification rather than workflow execution is often the bottleneck in overall workflow-processing performance. Can we automate the workflow-specification process?

It helps if we think of a workflow as *planning* to satisfy a goal. If we consider the start state as a collection of input files and the end state as a collection of desired output files, and we are given a set of operators, then we can use state space search or problem reduction algorithms from artificial intelligence to find a set of operators that will map inputs into outputs. Over the past year, we developed a workflow planning system. Figure 3 shows how our prototype works.[1,2] In our prototype, each file is a collection of records, each consisting of named fields.

Operators read and write files (collections of records that each have the same fields) processing input records into output records. In our application domain, fields are typed so that only certain output fields can match certain input fields of other operators. In our prototype, a backtracking-based search algorithm automatically finds necessary operators within the operator set and connects them into legal workflows. The prototype system is able to automatically produce all legal workflows.

A workflow automation system has many benefits. Because we've eliminated specification errors, the planning system generates only legal workflows. In fact, in one case, our prototype generated an unexpected workflow that helped experts debug the operator specifications. In addition, customer service is enhanced when we can provide higher throughput capabilities, and training new workflow experts is simpler.

What's Missing in Our Prototype

Our workflow planner is still a prototype and more work is needed. The current model simplifies workflow automation by ignoring some important details; for example, the planner doesn't handle workflow operators whose parameters change the operator's specification. When parameters don't affect the input and output specification or the operator's ordering constraints, they can be manually specified after the system has automatically created a workflow. We need to better understand other uses of parameters.

Earlier, we stated that grids are useful for handling large workflows, and we rejected relational database-management systems as inadequate for handling very large workflows. In retrospect, we might have thrown the baby out with the bathwater in that we can learn much from relational systems.

Our prototype doesn't yet handle operators that map a signature of input fields to the same signature of output fields — for instance, a sort operator that takes a file and returns a sorted file or a relational select statement that takes a table and returns a subset of

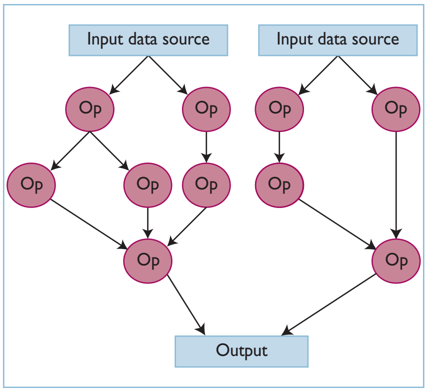

Figure 1. An example workflow. Workflows can have hundreds of steps and can operate on data from many data sources. They can execute in parallel both within and between operations.

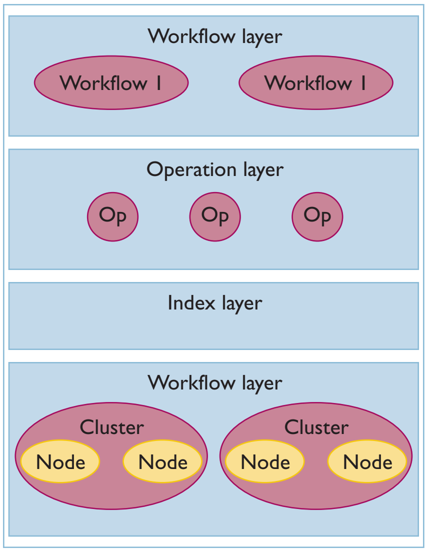

Figure 2. Workflow-management system. A workflow depends on operations. Many are custom and some are common relational operations that depend on an indexing layer for efficient implementation. Workflow operations and indices are implemented on top of a grid abstract machine consisting of nodes partitioned into clusters.

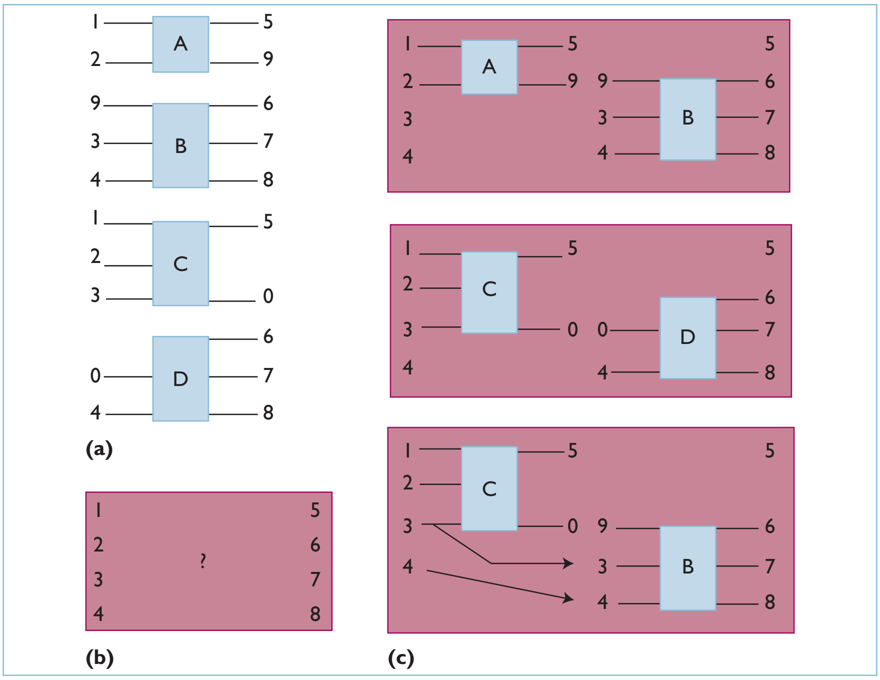

Figure 3. A prototype workflow planning system. This diagram shows (a) workflow operators A – D, (b) a workflow problem, and (c) how the operators can connect to solve the problem. For each operator, assume inputs on the left and outputs on the right, that both 0 and 9 can be matched to 9, and, for the rest, that a field can be matched only to itself.

the same table. We need to add specifications, such as `sorted(f)` to indicate that a file `f` is sorted, and selection criteria as in SQL. When we begin to list these kinds of extensions, we expect to find that all relational algebra operators will be useful in workflows (such as `project`, `select`, `join`, `union`, `sort`). So our collection of workflow operators will be a superset of relational algebra operators.

Our prototype is currently limited in that it doesn't handle algebraic equivalences for relational algebra operators, including commutativity and associativity transformations, which could help us develop heuristics to push selections and projections through joins. Often, we can plan to evaluate domain-specific operators earlier or later in a workflow, but we don't yet provide a way to declare how these alternative operator orderings are logically equivalent.

Our prototype doesn't yet support cost-based optimization. The current solution factors in an operator's cost but not the true cost as it's measured in real workflows. Relational optimizers progressed from ad hoc heuristics to cost-based planning, and we can predict a similar need if relational query plans are really subsets of workflow plans.

We can't yet state integrity statements such as "Do not deliver advertising solicitations to minors" or region-specific regulations as separate statements that can be woven into a workflow plan via query-modification or clause-resolution techniques.

We can't yet declare which users have access rights to which data sets and then provide privacy or security assurance that the data is isolated and partitioned.

We could use relational systems to specify several of the functions just described. Our problem with current-day relational technology is that current implementations aren't open and flexible enough to accommodate very large highly parallel grid workflow operations dominated by domain-specific operators. We'd be making a mistake to discount relational technology even if we can't yet use today's relational systems. So an integration challenge exists. We don't just want workflow planning on a grid – we want the best from the right mix of relational, grid, and workflow systems.

We also observe that when a Web service is specified by two sets of parameters – the set of inputs for a SOAP request and the set of outputs for SOAP response – then this is similar to our data transformation operators. As formulated, our field-based workflow-planning problem is similar to the Web service composition (WSC) problem.[3,4] Solving the problem of finding the Web services using a discovery service (UDDI) while connecting the Web services automatically to a user's request remains an active research area. In a related vein, we don't currently handle macro operators that are essentially workflow operators that contain workflow subroutines.

Analogies to relational database technology also give rise to considerations of whether a declarative business language can be defined to specify workflows – possibly a superset of SQL like various object query languages or as proposed in other work.[5] Figure 4 shows how the architecture of such a business language might map down to a workflow-execution plan in a similar way to how SQL is translated to a query-execution plan.

Our current prototype demonstrates the feasibility of automatically generating specifications for large workflows. Although an integration challenge remains with today's relational database systems, it appears that next-generation large-scale workflows will operate on grids; in many cases, their specifications will be auto-generated rather than manually specified; and next-generation workflow systems will borrow ideas from relational database systems such as integrity and security constraints and optimization.

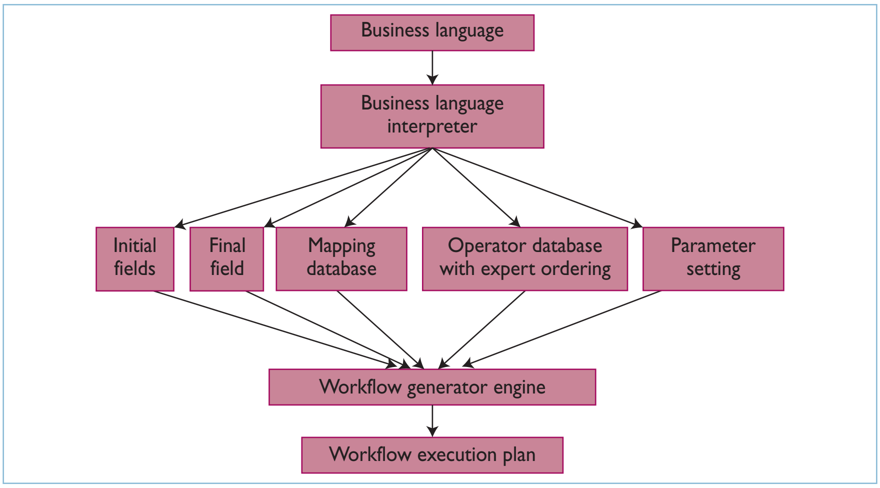

Figure 4. Architecture of a business language. A higher-level business language could raise the abstraction level so business analysts could focus on specifying the problem in terms of their domain rather than tediously hooking up workflow fields.

Acknowledgments

We thank Acxiom for providing funding for our research.

References

1. Z. Xiao, C. Thompson, and W. Li, "Automating Workflow for Data Processing in Grid Architecture," *Int'l Conf. Information and Knowledge Eng.* (IKE 06), Worldcomp 06 Pub., 2006, pp. 191–195.
2. Z. Xiao, C. Thompson, and W. Li, "Data Processing Workflow Automation in Grid Architecture," *Proc. 5th Int'l Conf. Grid and Cooperative Computing Workshops: Int'l Workshop Workflow Systems in Grid Environments* (WSGE 06), IEEE CS Press, pp. 189–195.
3. S. Oh, D. Lee, and S. Kumara, "A Comparative Illustration of AI Planning-Based Web Services Composition," *ACM SIGecom Exchanges*, vol. 5, no. 5, 2005, pp. 1–10; www.acm.org/sigs/sigecom/exchanges/volume_5/5.5-Oh.pdf.
4. C. Thompson and R. Jena, "Digital Licensing," *IEEE Internet Computing*, vol. 9, no. 4, 2005, pp. 85–89.
5. R. Hull et al., "Declarative Workflows that Support Easy Modification and Dynamic Browsing," *Proc. Int'l Joint Conf. Work Activities Coordination and Collaboration*, ACM Press, pp. 69–78.

Craig W. Thompson is a professor and the Charles Morgan Chair in Database at the University of Arkansas. His research interests include software architecture, middleware, data grids, and agent technology. Thompson has a PhD in computer science from the University of Texas at Austin. He is an IEEE Fellow. Contact him at cwt@engr.uark.edu.

Wing Ning Li is an associate professor at the University of Arkansas. His interests include design and analysis of algorithms, combinatorial optimization, parallel computing, and software reuse. Li has a PhD in computer science from the University of Minnesota. Contact him at wingning@uark.edu.

Zhichun Justin Xiao is an associate at Acxiom. His interests include parallel algorithms and workflow. Xiao has a PhD in computer science from the University of Arkansas. Contact him at justin.xiao@acxiom.com.