Combining Local and Grid Resources in Scientific Workflows (for Bioinformatics) *

Ann-Charlotte Berglund[1], Erik Elmroth[2], Francisco Hernández[2], Björn Sandman[2], and Johan Tordsson[2]

[1] The Linnaeus Centre for Bioinformatics, Uppsala University
ann-charlotte.sonnhammer@lcb.uu.se, www.lcb.uu.se
[2] Department of Computing Science and HPC2N, Umeå University
{elmroth, hernandf, c01bsn, tordsson}@cs.umu.se, www.gird.se

Abstract. In this work we examine some issues arising when using both local and Grid resources in scientific workflows. We have previously addressed and illustrated the benefits of a light-weight and generic workflow engine that manages and optimizes Grid resource usage. Extending on this earlier effort, we here illustrate how a client tool adapted for bioinformatics employs the engine to interface with Grid resources. We also explore how to define data flows (suitable for bioinformatics) transparently integrating local and Grid subworkflows. In addition, we examine the benefits of parameter sweep workflows and introduce a mechanism for describing this type of workflows in an abstract and concise manner. Finally, we employ the above mechanisms to reduce the complexity of gene sequence analysis.

Key words: Grid, workflow, data flow, SPMD, bioinformatics

1 Motivation and Background

To date many available tools (e.g., [1–3]) for scientific workflows have sophisticated functionalities that enable e.g., design, enactment, and scheduling of workflows, while providing support for data management and fault tolerance. However, most of these tools have at least one of three shortcomings: (1) the tool is a monolithic application where the above listed functionalities cannot easily be extracted and reused, (2) the tool targets a specific scientific domain and to adapt it to new domains is a major, if not impossible, endeavor, and (3) the tool is focused on workflow execution on client machines ignoring the benefits of using Grid resources for computationally or data intensive tasks.

In previous work [4, 5] we have discussed these problems and argued in favor of a loosely coupled design where workflow capabilities are exposed as a composable set of workflow services with clear separation of concerns. By orchestrating

* This research was conducted using the resources of the High Performance Computing Center North (HPC2N). Financial support has been provided by The Swedish Research Council (VR) under contract 621-2005-3667.

these services, environments tailored to the needs of a certain group of users can be constructed with much less effort compared to a (re)implementation of a full-featured system. Our implementation of the Grid Workflow Enactment Engine (GWEE) [4] demonstrates the feasibility of this approach.

This contribution is a direct continuation of these efforts. We identify the following four contributions, each described in a separate section of the paper. First, we illustrate the feasibility of the concept of a composable set of workflow capabilities by demonstrating how a client tool, adapted for use within bioinformatics, can be built on top of the general-purpose workflow engine (GWEE). We next discuss two issues in mixed client and Grid (data flow) workflows, namely how to transparently transfer data between the client machine and the Grid, and how to simplify parameter sweep studies. Finally, we use our results to reduce the complexity of a particular bioinformatics problem - gene sequence analysis.

2 GWEE Details and the Workflow Client Tool

Before discussing the data management issues encountered when combining Grid and local workflows, we give a brief overview of the GWEE and the client tool. The GWEE [4] is a Grid workflow enactment engine that is exposed as a Grid service. A strict separation between workflow logic and execution of individual tasks (e.g., computational jobs and file transfers) enables independence of workflow description language and interoperability with multiple Grid middlewares. The GWEE uses a data flow model of computation in which task dependencies define the execution order of the workflow. Workflows driven (in part or in full) by a control flow are supported via dependencies that send tokens to initiate the next task. Workflow tasks consist of three parts (1) a set of input ports, (2) a set of output ports, and (3) a process that maps the input ports to the output ports. Connections between output ports to input ports define task dependencies.

Basic client functionality. Workflows are designed, in a drag and drop manner, by connecting tasks together in a graph that specifies a control and/or a data flow. Users can start and cancel workflows, as well as pause and resume a running workflow. The client can handle mixed Grid and local workflows. Simple tasks that are not computationally intensive, e.g., preparation of input files, are typically performed on the client side while long running jobs can be included in a (Grid) subworkflow that is sent of to the GWEE for enactment on the Grid. The state of (also running) workflows can be stored to, and loaded from disk. The client can hence be disconnected from the GWEE during the execution of a long Grid subworkflow. This is useful e.g., when running the client on a laptop during travel. The client tool can be updated of Grid subworkflow progress either by synchronous requests or by notifications.

3 Local and Grid Data Flows

Most of the tasks executed in the Grid are *command line* applications and their input is typically obtained from the environment or from command line argu-

ments. This introduces the difficulty of mapping the arguments to the corresponding input or output ports for the correct arrangement of dependencies delineating the flow of data. Our solution for this problem is a task-template where the workflow designer can specify the correct port-argument mapping. By filling out a single form (generated from the task-template), users can specify input, output, and options for their workflow instances, without being concerned with low-level details such as the format used to describe Grid tasks.

Data transfers become cumbersome for data flow workflows that contain both local and Grid subworkflows. Special care is required to ensure that the output of a local task is available as input to a subsequent Grid task. The workflow client explicitly transfers the data to a Grid-available storage element and then back to the client machine once the Grid subworkflow is completed. These transfers, being control flow activities, are hidden from the user and the visual workflow displayed in the client tool remains pure data flow.

4 Parameter Sweeps in Workflows

A commonly used pattern in workflows is parameter sweep studies, where one computational task (or a subworkflow) is executed with different data (SPMD). This type of workflow requires a suitable partition of the input data (usually, and for our case, a file) and that each partition is assigned to a different task (or subworkflow). The information on how to process the data is identical for each data partition and is defined independently of the number of tasks. We accomplished this with a *regular-expression-like* mechanism for describing tasks, in particular any partition-specific input, output, and arguments. From these task descriptions, the graph is rewritten as a Grid subworkflow with a user-defined number of tasks and one data partition mapped to each task.

Embarrassingly parallel problems such as parameter sweep studies are well suited to Grid environments, where large numbers of resources are available. Even though parameter sweep workflows can be done without Grid workflows, data management in the client is simplified when the parameter sweep can be treated as one (Grid) subworkflow, instead of a (potentially large) set of individual tasks. This increased abstraction is beneficial both from a usability and a performance point of view. Other advantages include the reuse of a familiar programming paradigm akin to map-reduce or scatter-gather reductions (as done e.g., in MPI). By implementing parameter sweeps as a workflow graph rewrite on the client side, iterations of the *parallel-for* style are possible, regardless of whether or not iterations are supported by the workflow enactment engine.

5 Gene Sequence Analysis - a Bioinformatics Use Case

With the new sequencing technologies [6], the bioinformatics field is facing an avalanche of sequence data to be analyzed. As many of the computational problems are embarrassingly parallel, there is a growing interest in the bioinformatics field for Grid techniques in order to accelerate the data processing.

The analysis of biological data is typically a mix of short processing steps and larger computational parts that can easily be parallelized. These steps are all part of a scientific discovery workflow for answering a biological question. Therefore, for a workflow tool to be useful in the bioinformatics field it must have the ability to combine local and Grid subworkflows into a larger workflow. The large-scale bioinformatics analyses are made on general purpose computational grids, such as EGEE and NorduGrid, and can hence not rely on pre-installed software. All this considered, these analyses are today not done using a workflow tool. Instead, the different tasks in the analysis are typically connected by Perl scripts and are manually submitted to the computational grids. The drawbacks with this approach are that (1) it can be difficult to exactly replicate the analysis, (2) there is no easy way to reuse the scientific discovery workflow, and (3) it is difficult to analyze and verify this workflow.

As an alternative, we have implemented a small workflow using the NCBI version of the Basic Local Alignment Search Tool (BLAST) [7]. BLAST is a tool for sequence similarity searches, which is used as an analysis step in several larger workflows, for example when constructing probes for micro-array experiments and for reconstructing phylogenetic trees. Our workflow is a mixed local and Grid data flow, but uses control flow to install the BLAST binaries. The workflow also makes use of the parameter sweep mechanism to partition the BLAST database and parallelize the sequencing. This example illustrates the feasibility of our approach.

References

1. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Leen, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency Computat.: Pract. Exper. **18**(10) (2006) 1039–1065
2. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., Li, P.: Taverna: A tool for the composition and enactment of bioinformatics workflows. Bioinformatics **20**(17) (2004) 3045–3054
3. Taylor, I., Shields, M., Wang, I., Harrison, A.: The Triana workflow environment: architecture and applications. In Taylor, I., et al., eds.: Workflows for e-Science. Springer-Verlag (2007) 320–339
4. Elmroth, E., Hernández, F., Tordsson, J.: A light-weight Grid workflow execution engine enabling client and middleware independence. In Wyrzykowski, R., et al., eds.: Parallel Processing and Applied Mathematics, LNCS 4967, Springer-Verlag
5. Elmroth, E., Hernández, F., Tordsson, J., Östberg, P.O.: Designing service-based resource management tools for a healthy Grid ecosystem. In Wyrzykowski, R., et al., eds.: Parallel Processing and Applied Mathematics, LNCS 4967, Springer-Verlag
6. Hall, N.: Advanced sequencing technologies and their wider impact in microbiology. J. Exp. Biol. **9**(210) (2007) 1518–1525
7. Altschul, S.F., Madden, T.L., Schffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. **17**(25) (1997) 3389–3402