Adaptive Performance Tuning for Internet-Based Workflows

Paolo Vercesi
ESTECO
Trieste, 34012, ITALY
paolo.vercesi@esteco.it

Alberto Bartoli
DEEI
University of Trieste
Trieste, 34127, ITALY
bartolia@univ.trieste.it

Abstract

Composition of Internet-based services exported by different organizations has quickly become a key software paradigm for engineering and scientific communities. Although the scheduling of composite service invocations in such a multi-organization, multi-tiered and geographically dispersed environment may have strong impact on performance, this issue has not been explored very much so far. In this work we study the performance of the composite service and we take the perspective of the organizations involved in its implementation. We focus on the trade-off between the performance of the composite service and the global cost incurred at the participating organizations. We seek techniques for executing a given workload by minimizing such cost, while maintaining the performance delivered to clients to an acceptable level.

1. Introduction

Engineering and scientific communities are embodying service oriented architectures to process and manage large data sets, to execute simulations and to share both data and computing resources [3, 2] . Protocols for programmatic interactions across different organizations connected to the Internet (e.g. web services, grid computing), along with the plenty of software implementing these protocols, have made it feasible the *composition* of such services across the Internet [1].

In this work we consider the obstacles to the adoption of this paradigm by taking the perspective of the organizations providing the composing service. We assume that the cost incurred by an organization for executing a job is proportional to the time spent by the job at that organization. This assumption is justified by the application domain of our interest, characterized by many large and resource-consuming jobs that are submitted in batches and do not require interaction with human users. We seek techniques for executing a given workload by minimizing such cost, while keeping performance delivered to clients at an acceptable level. Knowing that the composite service will properly balance performance and cost may be a powerful factor for encouraging organizations to indeed share their services.

Our proposal can be implemented very simply and consists in a form of admission control, that is deployed at the composite service level. To maintain performance near to the optimal level the admission control module places a dynamic upper bound on the number of jobs that may be executed concurrently and delays excess jobs. The key feature of our approach is that we do not require any hook from the participating organizations and we treat the entire workflow as a black-box. What makes the problem challenging is that performance may heavily depends on a myriad of factors, that may be unknown, may vary unpredictably and are beyond the control of the scheduling machinery—resources available at each organization, bandwidth of the links, load imposed by other systems on the organizations and on the links, just to name a few.

2. Application Domain and Overview of Our Approach

A workflow is a set of tasks with data dependencies between them. We consider composite services that can be internally modeled as workflows, i.e., workflows implemented by means of services exported by potentially different organizations connected to the Internet [4]. Workflows are described by a *schema* that defines which services need to be executed, in what order and on which data. A workflow instance, that we call a *job*, consists of an execution of a schema on a specified set of input data and it produces a set of output data. Input data is composed of input parameters and input files, similarly output data is composed of output parameters and output files. Files are uploaded/downloaded when needed, based on directives enclosed in the job. Jobs are orchestrated by a *workflow engine*, that interprets the workflow schema.

31st Annual International Computer Software and Applications Conference(COMPSAC 2007)
0-7695-2870-8/07 $25.00

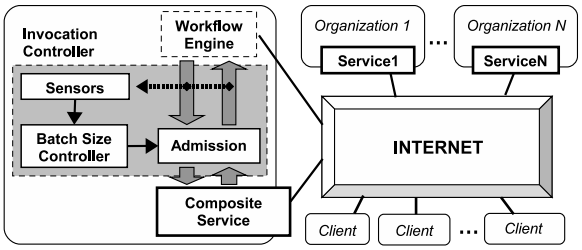

Figure 1. Physical layout

Conceptually, the workflow engine is an internal module of the composite service that needs not be exposed to clients. Figure 1 shows a possible arrangement of the various entities involved (the invocation controller module is the one that implements our approach and is outlined below).

We consider a *session* consisting of a specified number of jobs, the *session size*. The *session time* is the time needed to complete a session. The *throughput* is defined as the ratio of the session size and the session time. The *service time* is the time spent by one of the composing services for completing a job. This time includes: input data downloading, the processing time (which usually involves using licensed software), and output data uploading. We define *composite service time* the sum of the service times for each service involved in a job. We assume that the cost globally incurred by all the organizations is proportional to the composite service time.

The number of jobs concurrently being executed is called *batch size*. In systems of our interest [4] throughput grows more or less linearly with batch size, until reaching a saturation point starting from which throughput remains constant irrespective of the batch size. Composite service time grows approximately linearly with batch size, and depending on the actual system parameters it may grow more than linearly. The batch size, thus, has a significant impact on throughput and cost.

In our approach we vary the batch size dynamically and automatically so as to obtain a good trade-off between throughput and cost. Essentially, we attempt to place the batch size to the smallest value leading to maximum throughput. We say that this value is *cost-optimal*, because it is the one that exhibits minimum composite service time (amongst those that deliver maximum throughput). We do so without any explicit prior knowledge of the performance characteristics of the system and without any hook from the composing services.

We treat the workflow engine as a black-box and place an *invocation controller* in front of it (Figure 1). Jobs submitted by clients to the workflow engine are intercepted by the invocation controller. The invocation controller selects dynamically a value for the batch size and implements a form of admission control to ensure that the number of in-progress jobs does not exceed that value. Excess jobs are queued by the invocation controller, that will inject them into the workflow engine as prior jobs complete. The current value for the batch size is provided by a *control algorithm* whose inputs are estimates provided by dedicated *sensors*.

We evaluated our technique in a broad range of scenarios, by means of a detailed event-driven simulator (more details can be found in [5].) We assumed as baseline the best performance that can be obtained with a statically defined batch size value (note that, in practice, this value cannot be known in advance, due to the myriad of parameters involved which may also vary unpredictably during execution). We compared performance of our adaptive admission controller against this baseline. From these preliminary results we observed that our mechanism is indeed capable of trading off a small loss in throughput against a consistent decrease in the cost at the participating organizations.

From the point of view of clients, completing a session would thus take slightly longer. From the point of view of the participating organizations, however, executing a session would consume less resources. Each organization would thus need less resources for executing a given amount of work, thereby enabling the accommodation of further additional workloads, either within the organization or submitted from the outside. We believe that this paradigm could facilitate and encourage the participation of individual organizations in composite services. Our proposal may be an effective way to cope efficiently and automatically with the myriad of performance-critical factors that may be unknown, may vary unpredictably and are beyond the control of the scheduling machinery.

References

[1] F. Casati, S. Ilnicki, L. jie Jin, V. Krishnamoorthy, and M.-C. Shan. Adaptive and dynamic service composition in eflow. In *CAiSE '00: Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, pages 13–31, London, UK, 2000. Springer-Verlag.

[2] E. D. Gurmeet Singh, Carl Kesselman. Optimizing grid-based workflow execution. *Journal of Grid Computing*, 3(3):201–219, September 2005.

[3] G. Kola, T. Kosar, J. Frey, M. Livny, R. Brunner, and M. Remijan. Disc: A system for distributed data intensive scientific computing. In *First Workshop on Real, Large Distributed Systems (WORLDS04)*, San Francisco, CA, December 2004.

[4] P. Vercesi and A. Bartoli. On the performance of inter-oganizational design optimization systems. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, page 29. Winter Simulation Conference, 2006.

[5] P. Vercesi and A. Bartoli. Adaptive performance tuning for internet-based workflows. unpublished, January 2007.