An Experimental Study of Centralized and Decentralized Service Orchestration Approaches

Abul Ahsan Md Mahmudul Haque, Weihai Yu, Anders Andersen

University of Tromsø

N-9037 Tromsø, Norway

{Mahmudul.Haque, Weihai.Yu, Anders.Andersen}@uit.no

Abstract

Composite web services can be orchestrated either by centralized or decentralized approaches. Centralized approaches have been widely adopted due to easiness in process management. However, there are certain scenarios where decentralized approaches can be better alternatives. We investigate some performance characteristics of different orchestration approaches. We will use the empirical results as a guideline of our future research.

1. Introduction

Service orientation is a design paradigm for cost-effective construction and integration of sophisticated applications within and across organizational boundaries. Now, new-generation web applications provide open APIs in terms of web services, which are readily invocable by a wider range of applications. New and complex composite services are being constructed by aggregating existing web services. Service orchestration is the coordination of invocations in-between constituent services of the composite services. It has certain level of intricacies, including (1) nondeterminism, for the difficulty of anticipating the behavior of the external services in advance, (2) partial observability, since the status of the overall service is commonly obscure to other services, and (3) complex dependencies, as component web services mostly depend on the output of other services. Moreover, the orchestration process might need to handle data flow differently from the control flow[1].

Several methods and techniques have been proposed to deal with those issues. One of the main distinctions among them is between centralized orchestration approaches and decentralized approaches [2-5]. To understand the performance of these orchestration approaches, we carry out an empirical study that eventually helps us explore future direction in this important research area.

2. Web-service Orchestration Approaches

In the literature, an instance of a composite service is often called a business process or simply a process. Typically dedicated central engines orchestrate processes and maintain their runtime status in order for composite services to accomplish the overall goal. When a component service is done, or when some fault occurs, a service site sends a message back to the engine, either as a return message or as a callback. Monitoring and management of process executions become relatively straightforward. However, this can easily hinder overall performance while the number of services to be orchestrated gets larger. Furthermore, it is often difficult to choose a feasible location of the central engine when business logic crosses enterprise boundaries [6].

Decentralized orchestration approaches have therefore been introduced to overcome these limitations. In decentralized orchestration, there are multiple workflow engines in a distributed infrastructure where each of the engines orchestrates part of the original

This paper was presented at the NIK 2011 conference. For more information see http://www.nik.no/

composite web-service specification. The engines communicate directly with each other in an asynchronous manner to transfer data and control when necessary. Here, service orchestration and process management, including fault handling and recovery, and strategies to mitigate business constraint violation, become more challenging due to the absence of centralized states [7]. We categorize decentralized approaches into instantiation-based or messaging-based.

In instantiation-based approaches [2], a process is instantiated before its execution. During the instantiation, resources and controls are allocated to the subordinated engines based on an analysis of the process structures. One of the major limitations of instantiation-based approach is the waste of resources for those parts that might not be executed, such as some of the alternative paths or for the process that might roll back at an early stage. Furthermore, web services are typically implemented within stateless servers for the sake of scalability. Thus pre-allocation of resources is hardly acceptable, especially for the services that might not even be executed.

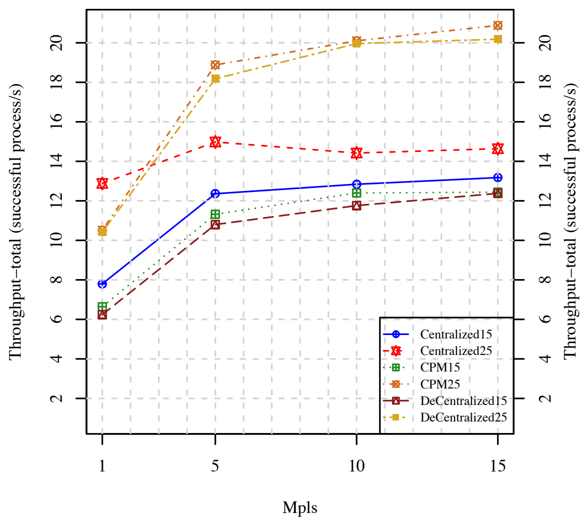
In messaging-based approaches, information like execution order of activities is carried in messages. For example, in the Continuation-Passing Messaging (CPM), a message contains two parts: continuations and environment [6]. Here a continuation contains the control flow of the execution whereas an environment contains information about process status and process-aware data. The service sites can independently interpret the messages and conduct the execution of services. New continuations and environments are generated either by the interpreted messages or by the outcome of service executions. The results of the service executions and the remaining activities of the process are carried in new messages to the subsequent service sites. Therefore services orchestration is actually a sequence of message exchanges and interpretations. To facilitate process recovery, messages also contain compensation continuations, which are recovery plans automatically generated during process execution.

3. Performance

We developed prototypes of three orchestration approaches: (1) with central engines, (2) decentralized with continuation-passing messaging, and (3) decentralized with instantiation of control prior to process execution. The prototypes run in the Omnet++ simulator[8]. We choose simulation over running prototype applications for performance study, as using simulation we have better control over a wider variety of configurations and runtime parameters, and thus can obtain better understanding of the factors that influence the performance.

In the simulations, service sites are located in several 100 Mbps LANs and these LANs are interconnected with a 10 Gbps backbone network. A composite service consists of 2 parallel branches and each branch has a sequence of 4 service invocations. These service sites are chosen randomly from the set of 15 and 25 sites. The second service in the second branch has a 20% chance of throwing a fault.

An execution of a service program at a service site takes on average 100ms. A central engine takes on average 5ms to dispatch an activity. A service site takes on average 10ms to interpret a CPM message. With central engines, a service site takes on average 2ms to dispatch an invocation.



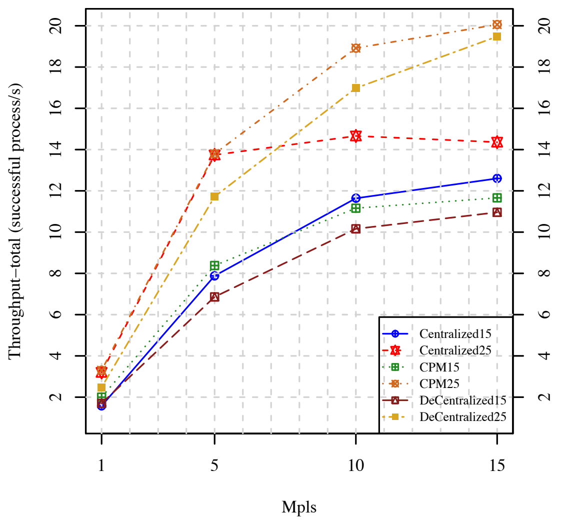


Figure 1: Throughput with no backbone delay

Figure 2: Throughput with backbone delay

To understand the impact of network delays, we run two sets of experiments: (1) the backbone network has no delay (2) the backbone network incurs 50ms delay. Furthermore, to understand the scalability of the approaches, each set of experiments is divided into two groups: one with 15 service sites and another with 25 service sites. To measure the performance under different workload, we model the workload of a site by its multiprogramming level, or MPL, which is the number of concurrent service executions at that site. So MPL 6 means that each site concurrently executes 6 services most of the time. Initially, a fixed number of processes are fed into the system. A new process is started as soon as an existing one terminates. In centralized orchestration and in instantiation-based decentralized approaches, the message sizes depend on both the sizes of the input and output of the service operations. In CPM, the message sizes are also dependent on the sizes of the continuations and environments, which eventually change during the execution of the process.

Figure 1 shows the aggregate throughput of all service sites (measured in the number of successfully completed processes per second) when the backbone network does not incur any delay. Figure 2 shows the throughput when the backbone network incurs delay. In the figures, Centralized stands for the centralized approach, CPM for continuation-passing messaging, and DeCentralized for instantiation-based decentralized approach. The numbers 15 and 25 are appended with those notations to mention the number of service sites in the experiments. Figure 1 depicts that with 15 service sites or at low system loads with 25 server sites, the centralized approach outperforms both CPM and pre instantiated approaches; however, with 25 service sites, when the load increases, the throughout flattens gradually. As can be seen from the figures, decentralized approaches scale well in highly loaded situations. The reason is that in case of the centralized approach, when the workload increases the central engine becomes congested, while in case of decentralized approaches, there is no obvious performance bottleneck. In Figure 2, both of the decentralized approaches outperform the centralized approach. It seems therefore that backbone network is more sensitive to the number of messages than the sizes of the messages. Furthermore, within decentralized approaches, CPM has higher throughput than the instantiation-based approach, indicating that larger messages in CPM are less costly than an additional instantiation phase of the instantiation-based approach.

4. Future Work

Our empirical study shows that different orchestration approaches have different performance characters. In lightly loaded engines, centralized approach has better throughput; however, when the load is increased then CPM approach gets better throughput than both centralized approaches and instantiation-based decentralized approaches. Although the centralized approach is still ideal due to the ease of process management and monitoring; however, there are situations where decentralized approaches are better alternatives, for example, whenever there is no appropriate place for the central engine or when the engine gets congested. Given these observations, it would be suitable to adopt a new orchestration approach, which can dynamically switch between orchestration approaches based on the composite threshold of its current load or a combination of load and other important parameters for instance, the number of pending messages in the engine. Thus determining the threshold requires better understanding about the involved service sites of the composite service. We believe further exploration of this idea could lead to some significant results and a truly dynamic orchestration approach for service orchestration is crucial in todays open world of web applications.

Reference:

1. Marconi, A. and M. Pistore, *Synthesis and Composition of Web Services.* Formal Methods for Web Services, 2009. **5569**: p. 89-157.
2. Benatallah, B., M. Dumas, and Q.Z. Sheng, *Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services*, in *Distributed and Parallel Databases*2005.
3. Muth, P., et al., *From Centralized Workflow Specification to Distributed WorkflowExecution.* J. Intell. Inf. Syst., 1998. **10**(2): p. 159-184.
4. Schneider, J., B. Linnert, and L.O. Burchard. *Distributed workflow management for large-scale grid environments*. in *Applications and the Internet, 2006. SAINT 2006. International Symposium on*. 2006.
5. Chafle, G.B., et al., *Decentralized orchestration of composite web services*, in *Proceedings of the 13th international World Wide Web conference on Alternate track papers &amp; posters*2004, ACM: New York, NY, USA. p. 134-143.
6. Yu, W., *Scalable Services Orchestration with Continuation-Passing Messaging*, in *First International Conference on Intensive Applications and Services,2009. INTENSIVE '09.* 2009, IEEE: Valencia. p. 59-64.
7. Wang, M., K.Y. Bandara, and C. Pahl, *Integrated Constraint Violation Handling for Dynamic Service Composition*, in *Proceedings of the 2009 IEEE International Conference on Services Computing*2009, IEEE Computer Society. p. 168-175.
8. Andra, et al., *An overview of the OMNeT++ simulation environment*, in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*2008, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Marseille, France. p. 1-10.