ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Decentralization of BPEL Using Various Processes.

A.V.A Sushama Sarvani
M.Tech Student,
Department of CSE
Vignan'sNirula Institute of Technology and Science for Women, Pedapalakaluru, Guntur-522 005

Dr. Paturi Radhika
Associate Professor,
Department of CSE
Vignan'sNirula Institute of Technology and Science for Women, Pedapalakaluru, Guntur-522 005

ABSTRACT:

This article presents BPELcube, a framework comprising a scalable architecture and a set of distributed algorithms, which support the decentralized enactment of BPEL processes. In many application domains, BPEL processes are long-running, they involve the exchange of voluminous data with external Web services, and are concurrently accessed by large numbers of users. In such context, centralized BPEL process execution engines pose considerable limitations in terms of scalability and performance.To overcome such problems, a scalable hypercube peer-to-peer topology is employed by BPELcube in order to organize an arbitrar number of nodes, which can then collaborate in the decentralized execution and monitoring of BPEL processes. Contrary to traditional clustering approaches, each node does not fully take charge of executing the whole process; rather, it contributes to the overall process execution by running a subset of the process activities, and maintaining a subset of the process variables. Hence, the hypercubebased infrastructure acts as a single execution engine, where workload is evenly distributed among the participating nodes in a finegrained manner. An experimental evaluation of BPELcube and a comparison with centralized and clustered BPEL engine architectures demonstrates that the decentralized approach yields improved process execution times and throughput

Index Terms— Composite Web Services, Processes, Business Process Management, Simulation of Business Processes

INTRODUCTION:

THe *Web Services Business Process Execution Language* abbreviated to WS-BPEL or BPEL, is widely considered the *de facto* standard for the implementation of executable service-oriented business processes as compositions of Web services. The language specification defines a set of *activities* to support synchronous and asynchronous interactions between a process and its clients, as well as between a process and external Web services. Moreover, a number of *structured* activities are used to implement typical control flow units such as sequential or parallel execution, ifelse statements, loops, etc. Hence the control flow of a business process is realized by a number of activities, which are appropriately ordered and put together. The BPEL language also provides the necessary elements to support the expression of common programming concepts such as scope encapsulation, fault handling, compensation, and thread synchronization.

Data handling is realized by means of *variables*, which are conveniently used by a BPEL process to hold the data that are generated and/or consumed upon execution of its constituent activities. Thus the various activities of a process are able to share data with each other simply by reading from and writing to one or more of the process variables. To date, most of the available solutions for the execution of BPEL processes have been designed and operate in a centralized manner, whereby an orchestrator component running on a single server is responsible for the execution of all process instances, while all relevant data are maintained at a single location (i.e. the server hosting the BPEL engine). Clearly, such approach cannot scale in the presence of a potentially large number of simultaneous, long-running process

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

instances that produce and consume voluminous data. While in some cases clustering techniques are supported and can be employed to address the scalability issue, the deployment and maintenance of clusters consisting of two or more centralized BPEL engines sets requirements on the underlying hardware resources, which cannot be always fulfilled by the involved organizations. Furthermore, clustering could be proved an inefficient approach under certain conditions, as it cannot overcome the emergence of bottlenecks that are caused by specific activities of a BPEL process. Hence, a more fine-grained workload distribution approach is called for to ensure scalability of the BPEL execution engine at lower cost. In the following section, we introduce a motivating scenario from the environmental domain so as to better capture the problem in real-world terms.

Existing system

In order to facilitate the development, delivery and reuse of environmental software models, service orientation has been recently pushed forward by several important initiatives1;2;3 and international standardization bodies4 in the environmental domain. In the light of those efforts, both geospatial data and geo-processing units are exposed as Web services, which can be used as building blocks for the composition of environmental models in the form of BPEL processes several challenges arise upon this paradigm shift. Efficient execution and monitoring of long-running environmental processes that consume and produce large volumes of data, in the presence of multiple concurrent process instances are among the prominent issues that one should effectively deal with.

PROPOSED SYSTEM

In an effort to address situations such as the one described previously, we introduce a framework comprising a scalable Peer-to-Peer (P2P) architecture and a set of distributed algorithms to support the decentralized enactment of BPEL processes. Our framework, dubbed BPELcube hereinafter, particularly focuses on the improvement of the average process execution times, and the enhancement of the overall throughput of the execution infrastructure in the presence of multiple, concurrent and long-running process instances. BPELcube is mainly characterized by the following features: *Fully decentralized, P2P-based BPEL engine architecture*.

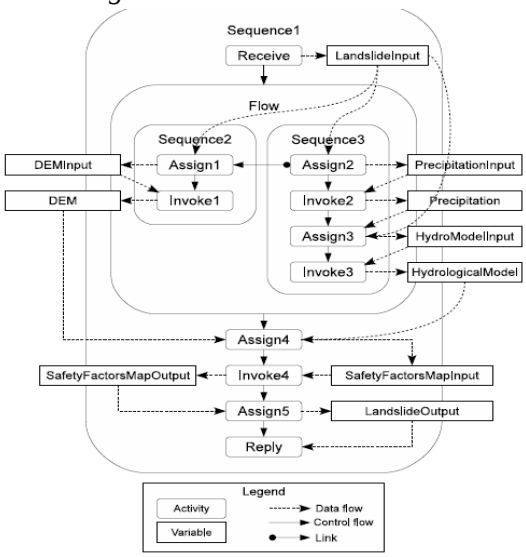

Fig. 1. A BPEL process used for the calculation of landslide probabilities in a user-specified area.

BPEL processes are deployed, executed, and monitored by a set of nodes organized in a hypercube P2P topology. Each node does not fully take charge of executing the whole process; rather, it contributes by running a sub-set of the process activities, and maintaining a sub-set of the generated process data. Thus the BPEL execution engine is fully operational without the need of any central controller components.

_ *Fine-grained distribution of process Activitie.* Decentralization of process execution fits to the nature of long-running business-to-business interactions, and significantly improves the performance and throughput of the execution infrastructure. BPEL processes are fully decomposed into their constituent activities. Large-scale parallelization is feasible as the various activities designated to run in parallel can be synchronized and executed by different nodes.

Proximity-based distribution of process variables.

Since in many application domains processes consume and produce large volumes of data, it is important that those data are distributed in order to avoid resource exhaustion situations. Our algorithms make sure that

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

the data produced by a BPEL process will be distributed across the nodes involved in its execution. Moreover, they will stay close to the process activities that produce them, thereby avoiding the unnecessary transfer of potentially large volumes of data between nodes as much as possible.

Asynchronous interaction with the client.

Even if a BPEL process is synchronous following the requestresponse communication pattern, the interaction between the client and the distributed execution engine occurs in an asynchronous, non-blocking manner. This way, the execution engine is able to serve multiple long-running process instances without the need to maintain open connections to the respective clients over long periods of time. Furthermore, while waiting for a long-running process instance to complete, clients are given the monitoring mechanisms to retrieve intermediate results, without intervening or inflicting additional delays to the process execution.

Efficient use of the available resources and balanced workload distribution.

The BPELcube algorithms ensure that all nodes available in the P2P infrastructure will contribute to the execution of BPEL processes.

The frequency of use of each node is taken into account upon load balancing, while efficient routing techniques are employed in order to achieve an even distribution of the workload at any given time and thereby avoid the emergence of performance bottlenecks. In the following section, we present an analysis of the relative literature and pinpoint the added value of our work in the context of decentralized BPEL process execution. Then, we proceed in Section 3 with the detailed presentation of our proposed approach. Examples based on the landslide BPEL process that was described in Section 1.1 are given where necessary in order to better explain the various algorithms. An experimental evaluation of our approach along with the retrieved measurements are presented and discussed in Section 4, while we conclude this paper and identify paths for future work in Section 5.

BPEL Decentralization

The decomposition and decentralized enactment of BPEL processes is a valid problem that has been the subject of many research efforts in the last years. In the following, we review a number of related results that have become available in the literature.

A P2P-based workflow management system called SwinDeW that enables the decentralized execution of workflows was proposed by Yan et al. [17]. According to the authors, the generic workflow representation model is compatible with most concrete workflow languages including BPEL, although this compatibility is not demonstrated. In any case, similar to our presented approach, SwinDeW is based on the decomposition of a given workflow into its constituent tasks, and their subsequent assignment to the available nodes of a P2P network, in order to remove the performance bottleneck of centralized engines

BPELcube Node Architecture

The main internal components of a node participating in the BPELcube engine are shown in Figure 2. The *P2P Connection Listener* acts as the entry point of each node accepting incoming requests from other nodes in the hypercube. Each request is bound to a new P2P connection, which is then passed to a *P2P Connection Handler* for further processing

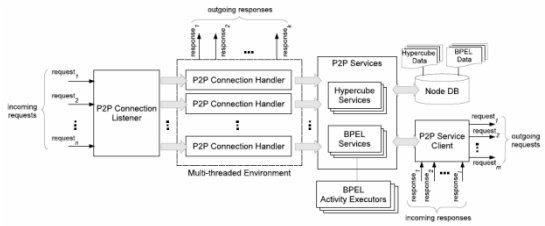

Fig. 2. Internal architecture and main components of the BPELcube node.

Process Deployment

For a BPEL process to be deployed to the BPELcube engine, a request containing a bundle with all necessary files needs to be submitted to one of the available nodes in the hypercube. In particular, this bundle contains the BPEL process specification, the

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

WSDL interface, the WSDL files of all external services, as well as any potentially required XML schemas and/or XSLT transformation files. Upon receipt of the deploy request, the node first performs a syntactic validation of the included files, and then decomposes the process into its constituent activities and variables.

Process Execution

The execution of an already deployed process is triggered each time an *ExecuteProcess* request is received by one of the available nodes in the hypercube, containing the process identifier *idp* and the initial input, if any. To ensure even distribution of workload, the recipient node starts a random walk within the hypercube by means of shortest-path routing, in order to appoint the node that will actually take over the role of the execution *manager*.

The appointed manager creates a new P2P session for the execution of the process, and stores it in a tuple

Algorithm 1: Processing of a recruitment request.

input: $\langle id_{p2p}, id_p, E_a, E_v, e_m \rangle$

```
1  begin
2      e_L ← get local endpoint address
3      Get process tuple based on id_p
4      recruitment_complete ← true
5      foreach (id_a, e) ∈ E_a do
6          if e = null then
7              e ← e_L
8              Update E_a
9              recruitment_complete ← false
10             Get activity tuple based on process tuple
               and id_a
11             V_w ← get from activity tuple
12             foreach id_v ∈ V_w do
13                 (id_v, e) ← get corresponding entry
                   from E_v
14                 if e = null then
15                     e ← e_L
16                     Update E_v
17                 end
18             end
19             N ← get all hypercube neighbors
20             Broadcast timestamp of last use to all
               nodes ∈ N
21             break
22         end
23     end
24     if recruitment_complete = false then
25         w ← get LRU neighbor in lowest dimension
26         e ← get endpoint address from w
27         Send Recruitment request to e
28     else
```

Conclusions

We presented a distributed architecture based on the hypercube P2P topology along with a set of algorithms that enable the decentralized execution of BPEL processes. Our approach targets towards the improvement of the average process execution times and the enhancement of the overall throughput of the execution infrastructure, in the presence of multiple long-running process instances that involve the exchange of large data.

The presented algorithms support the decomposition of a given BPEL process and the subsequent assignment of the constituent activities and data variables to the available hypercube nodes. Execution is then performed in a completely decentralized manner without the existence of a central coordinator. Our distributed approach also provides a lightweight monitoring mechanism that does not intrude into the process execution, but rather allows the retrieval of monitoring information from the hypercube in a seamless manner. We evaluated our approach in a series of experiments, and compared it with centralized and clustered architectures in terms of performance. The retrieved measurements indicate that our hypercube-based architecture is more suitable for the execution of long-running and data-intensive processes, while it is able to accommodate more concurrent clients than the other two architectures.

Moreover, thanks to the even distribution of workload, our approach copes with large data in a more efficient manner. In future work, we aim to expand our worker recruitment algorithm so as to consider additional factors like network proximity or other QoS, which are complementary to the frequence of use of the employed nodes. This expansion will facilitate the deployment of the hypercube-based engine on less controlled settings IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. X, NO. X, MONTH 20XX 14 such as WAN networks. We are also interested in extending the proposed architecture to support Cloudbased deployment of the BPELcube engine. We anticipate that by moving BPELcube to the Cloud, we will be able to exploit elasticity capabilities for dynamically increasing or decreasing the hypercube dimension. This way, the BPELcube engine will be able to effectively and timely respond to

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

workload changes. Finally, in terms of implementation, we will investigate the use of parallel query processing techniques to further enhance the performance of BPELcube nodes, in the presence of multiple concurrently running process instances.

REFERENCES:

[1] OASIS, "Web Services Business Process Execution Language Version 2.0," Apr. 2007. [Online]. Available: http://docs.oasisopen. org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[2] B. Sch¨affer and T. Foerster, "A client for distributed geoprocessing and workflow design." *Journal of Location Based Services*, vol. 2, no. 3, pp. 194–210, 2008.

[3] A. Weiser and A. Zipf, "Web service orchestration of ogc web services for disaster management," in *Geomatics Solutions for Disaster Management*, ser. Lecture Notes in Geoinformation and Cartography, J. Li, S. Zlatanova, A. G. Fabbri, W. Cartwright, G. Gartner, L. Meng, and M. P. Peterson, Eds. Springer Berlin Heidelberg, 2007, pp. 239–254.

[4] X. Meng, F. Bian, and Y. Xie, "Research and realization of geospatial information service orchestration based on BPEL." In *Proceedings of the 2009 International Conference on Environmental Science and Information Application Technology, ESIAT 2009*. IEEE Computer Society, 2009, pp. 642–645.

[5] F. Theisselmann, D. Dransch, and S. Haubrock, "Service-oriented architecture for environmental modelling - the case of a distributed dike breach information system," in *Proceedings of the 18th World IMACS/MODSIM Congress*, 13-17 July 2009, pp. 938–944.

[6] D. Roman, S. Schade, A. J. Berre *et al.*, "Environmental services infrastructure with ontologies - a decision support framework," in *Proceedings of EnviroInfo 2009: Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools*. Shaker Verlag, 2009, pp. 307–315.