

Integration of Distributed Services into Workflow Applications with Optimization of Time, Cost and Energy Consumption

Paweł Czarnul

Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology, Poland
pczarnul@eti.pg.gda.pl <http://fox.eti.pg.gda.pl/~pczarnul>

Abstract. The paper presents a concept, an algorithm and a system that allow integration of various types of services into complex workflow applications. Furthermore, an approach is presented for optimization of service selection considering service execution times, costs and energy consumption when some services may fail.

Keywords: workflow management, QoS, energy management, grid computing, cloud computing

1 Introduction

Integration of various services run on various commodity computers, servers and clusters have become possible thanks to various distributed architectures and technologies such as grid, cloud computing and volunteer computing. This requires QoS-aware service selection. The more computing resources are used the more important it is to consider not only execution time and cost but also energy consumption. Section 2 refers to selected related work on integration of services in distributed environments, energy management and motivations for a new solution. Section 3 provides details regarding the proposed solution with a description of the platform in section 3.1, algorithm in section 3.2 and implementation in section 3.3 while section 4 concludes the work.

2 Related Work and Motivations

SOA [1] is a software architecture that is built on the concept of loosely coupled distributed services published by independent providers. The services can be integrated into complex scenarios and invoke one another.

Grid computing [2] is about controlled resource sharing where different Virtual Organizations (VOs) offer hardware and software resources that are accessed through grid middlewares such as Globus Toolkit, Unicore and others. High level grid systems allow easy access to the distributed resources, in particular services for running applications and data management. Differences in user management, authorization methods etc. are hidden by the grid middleware.

Cloud computing [3] considers outsourcing the following: SaaS – Software as a Service, IaaS – Infrastructure as a Service, PaaS – Platform as a Service. The client

does not need to know details of where and how particular services are implemented and is interested mostly in the reliable delivery.

Volunteer computing engages computers of Internet clients to perform parts of a global task [4].

Green computing is energy-aware processing that is focused on minimization of energy consumption [5].

When considering a particular distributed task, the goal is to support interoperability and QoS management when integrating services to accomplish complex tasks. A complex task is usually modeled as workflow application represented by a directed graph $G(V, E)$ without loops in which V is a set of vertexes representing workflow tasks while E is a set of edges which denote dependencies between the tasks. For each task t_i , there is a set of services S_i which contains services each of which is capable of executing task t_i . The services in S_i offer the given functionality on particular and possibly different terms such as the execution time, cost and possibly others such as reliability, conformance, accessibility [6]. The standard optimization goal is to assign one service out of S_i to each task t_i so that a certain global optimization goal is minimized or maximized subject to additional constraints. For instance, a typical optimization goal is to minimize the workflow execution time with a bound on the total cost of selected services [7]. Since there can be many servers or commodity computers from which services are offered, there arises a need for energy management of such machines.

Thus the author proposes how to extend this workflow model to take into account the energy consumption of these servers and computers. This is all done considering possible service or node failures. Furthermore, the model and implementation allows incorporation of both services published by their providers from particular resources as well as services able to find resources to execute arbitrary code.

3 Proposed Solution

3.1 A Platform for Integration of Various Types of Services

BeesyCluster [8] is a middleware that allows integration of various types of services. It allows publishing particular services by their providers from their own locations. In this case, the provider can define parameters of such services such as costs and manage the services dynamically i.e. withdraw a service or offer new services. Furthermore, BeesyCluster allows offering access to particular computational resources by allowing the user to upload codes to particular clusters or servers and run such applications. Finally, a BeesyCluster user may be offered access to an account on a server or a cluster with access to both hardware and software options there with a possibility to launch applications with graphical interfaces through a Web browser. Additionally, the BeesyCluster layer can match one of general accounts of servers/clusters not owned by any of the users to the code the client wants to run (Figure 1).

The author proposes that such services are integrated into workflow applications with management of time, cost and energy consumption as presented in Section 3.2.

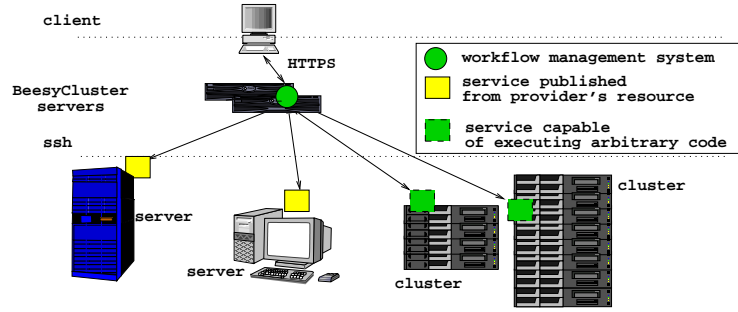


Fig. 1. BeesyCluster and Services

3.2 Optimization of Time, Cost and Energy Consumption for Workflow Applications

For optimization of workflow execution, usually there is a trade-off between the execution time and cost. Namely, faster services (e.g. installed on faster hardware) will usually be more costly.

Similarly, there is a trade-off between execution time and energy consumption if we consider that services may fail.

Generally, after the service selected for execution of task t_i is started, it can either finish successfully or fail. In the latter case, a backup service is selected [8] and we can start a node for a backup service and run the backup service. Alternatively, nodes for possible backup services can be started in advance. This solution saves the execution time but results in a higher energy consumption. It is assumed that the nodes for the backup services are shut down after the service has completed. Now, the following parameters are proposed for service s_{ij} running on node N_{ij} to take into account energy-aware workflow execution in the context of possible service or hardware failures:

- $S_{backup\ ij}$ – a set of backup services of service s_{ij} used in the recent time frame T ,
- $nc_{backup\ ij}^l$ – for the l -th service $s_{backup\ ij}^l$ in $S_{backup\ ij}$ the number of times it was run in the recent time frame T ,
- $t_{backup\ ij}^l$ – for the l -th service $s_{backup\ ij}^l$ in $S_{backup\ ij}$ the average service running time in the recent time frame T ,
- n_{ij} – the number of runs of backup services of service s_{ij} in the recent time frame T ,
- $t_{start\ N_{backup\ ij}^l}$ – start-up time of node $N_{backup\ ij}^l$ on which service $s_{backup\ ij}^l$ runs,
- $P_{N_{ij}}$ is the average power consumption required by node N_{ij} when running service s_{ij} .

For each service s_{ij} the author proposes to consider the following parameters:

- t_{ij} – execution time of the service itself,

- c_{ij} – cost of running the service,
- $e_{ij} = (t_{ij} + t_{start\ N_{ij}})P_{N_{ij}}$ – energy consumption for the service,

It should be noted that the node on which service s_{ij} runs can be started just before the service is needed or act as a server and be active at all times because of other reasons. In any case, we consider the times and energy consumption taken by the given service s_{ij} . These parameters can be defined both for particular services published by providers and for services capable of finding nodes and executing any arbitrary code given by the user. In the latter case, virtual services are considered for each of the node on which the service might run with the time, the cost and energy parameters corresponding to the node.

It is possible that when the task is to be executed, the node on which it is to be run is no longer available. In this case, the following two solutions are considered, for each of which a virtual service is proposed:

service $s_{ij}^{delayed}$ delayed start-up – start the next best service on the node which is available at the moment the previous service failed. We can estimate its parameters as:

$$\begin{aligned}
- t_{ij}^{delayed} &= t_{ij} + \frac{1}{n_{ij}} \sum_l n c_{backup\ ij}^l (t_{backup\ ij}^l + t_{start\ N_{backup\ ij^l}}), \\
- c_{ij}^{delayed} &= c_{ij} \\
- e_{ij}^{delayed} &= e_{ij} + \frac{1}{n_{ij}} \sum_l n c_{backup\ ij}^l (t_{backup\ ij}^l + t_{start\ N_{backup\ ij^l}}) P_{N_{backup\ ij^l}}
\end{aligned}$$

service $s_{ij}^{immediate}$ immediate start-up – start the nodes used for the backup in the recent time frame T immediately which results in the following parameters for the service:

$$\begin{aligned}
- t_{ij}^{immediate} &= t_{ij} + \frac{1}{n_{ij}} \sum_l n c_{backup\ ij}^l t_{backup\ ij}^l, \\
- c_{ij}^{immediate} &= c_{ij}, \\
- e_{ij}^{immediate} &= e_{ij} + \\
&|S_{backup\ ij}| \frac{1}{n_{ij}} \sum_l n c_{backup\ ij}^l (t_{backup\ ij}^l + t_{start\ N_{backup\ ij^l}}) P_{N_{backup\ ij^l}}
\end{aligned}$$

Note that the cost of performing operations may depend on the time of day which suggests that performing computations in areas where there is night time at the moment might be preferred [9].

Consequently, for each service s_{ij} there are two services which differ in the execution time and energy consumption based on the past experience, namely service $s_{ij}^{delayed}$ and service $s_{ij}^{immediate}$. The optimization algorithm can choose the best set of services so that the required optimization goals are met even if some services have failed.

The goal of this extended formulation is to minimize the workflow execution time $t_{workflow}$ subject to the following constraints:

$\sum_{i,j} c_{ij} < B$ where B is the budget given, $\sum_{i,j} e_{ij} < E$ where E is the available budget and E is the maximum energy consumption used for the execution of the workflow application.

Following [9], a genetic algorithm can be used to solve this problem and it was implemented in the BeesyCluster middleware. Namely, the representation of the solution in the chromosome consists of the following:

1. a part with $|V|$ elements each of which contains an index of the service selected for particular task of the workflow,
2. a part with $|V|$ elements which denotes the order of execution of services assigned to different tasks if the services assigned to these tasks execute on the same node and if the order is not determined by the acyclic directed graph G .

The algorithm for execution of workflow applications using the aforementioned approach is presented in Section 3.3.

3.3 Implementation

Following [8] Beesy-Cluster uses an engine based on the Java EE architecture in which separate message driven beans are responsible for selection of a service and execution of a particular workflow task [8]. The author proposes how to extend this approach to incorporate dynamic start-up and shutdown of nodes.

The proposed solution works as follows. Initially, a list of initial workflow tasks is created. The execution of parallel tasks is executed by separate message driven beans. The execution of each task is presented in Figure 2.

Before task t_i is to be executed, future nodes following the given node must be started (if not already running) depending on the types of services selected for their execution. This step is shown in the activity diagram in Figure 3.

Note that the values of the services presented above are estimates based on the past experience.

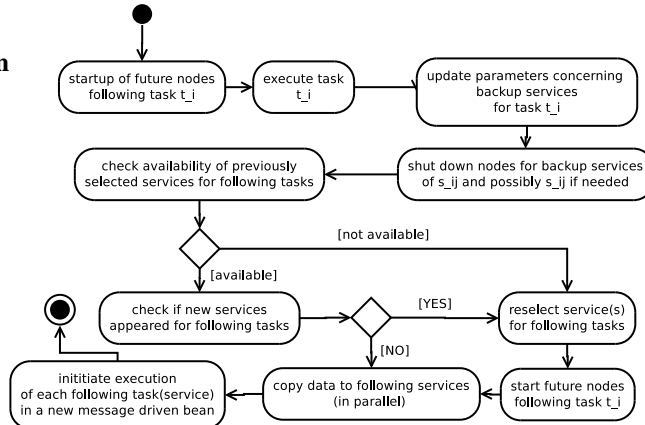


Fig. 2. Execution of a Node

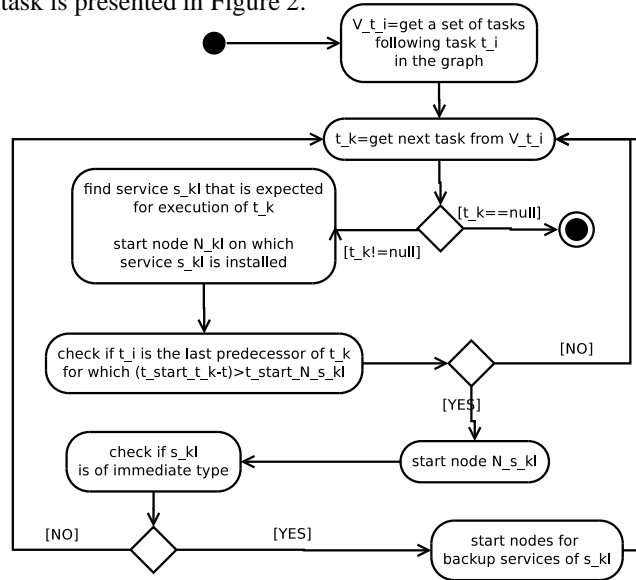


Fig. 3. Start-up of Future Nodes

For each task, after a service and possibly backup services were selected for execution and the execution of the task has finally terminated, the values concerning the backup services for the task are updated with possibly new services added to sets S_i or $S_{backup\ ij}$ and some services subtracted. It is up to the dynamic QoS algorithm, as presented by the author in [8], to decide which services are selected if others have failed or newly appeared services in the process.

Nodes are started using a Python script that uses the wake-on-LAN function.

4 Conclusions

The paper presents a concept and an environment that allows integration of various types of services: services published by particular providers from locations managed by them and services allowing to execute arbitrary code on the resources available to the system.

Furthermore, the paper presents an algorithm for selection of services for particular workflow tasks that copes with service failures by reselection of services if others have failed or have become unavailable or new services have appeared. The algorithm considers execution times, cost and energy consumption of the nodes on which the services run to minimize the workflow execution time while keeping the cost and energy consumption below predefined thresholds.

References

1. Steve Graham and Simeon Simeonov and Troufic Boubez and Doug Davis and Glen Daniels et al.: Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI. SAMS Publishing (2002)
2. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In: Open Grid Service Infrastructure WG. (2002) Global Grid Forum, <http://www.globus.org/research/papers/ogsa.pdf>.
3. Ahson, S.A., Ilyas, M., eds.: Cloud Computing and Software Services: Theory and Techniques. CRC Press (2011) ISBN 978-1-4398-0315-8.
4. Open-source software for volunteer computing and grid computing: (Boinc) University of California, <http://boinc.berkeley.edu/>.
5. Murugesan, S.: Harnessing green it: Principles and practices. IT Professional **10** (2008) 24–33
6. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.: Quality driven web services composition. In: Proceedings of WWW 2003, Budapest, Hungary (2003)
7. Yu, J., Buyya, R.: Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Scientific Programming Journal (2006) ISSN: 1058-9244, IOS Press, Amsterdam, The Netherlands.
8. Czarnul, P.: Modeling, run-time optimization and execution of distributed workflow applications in the JEE-based BeesyCluster environment. The Journal of Supercomputing (2010) 1–26 10.1007/s11227-010-0499-7, <http://dx.doi.org/10.1007/s11227-010-0499-7>.
9. Czarnul, P.: Modelling, optimization and execution of workflow applications with data distribution, service selection and budget constraints in beesycluster. In: IMCSIT. (2010) 629–636 Wisla, Poland.