Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing

Vladimir Stantchev[1,2,3] and Christian Schröpfer[2]

[1] International Computer Science Institute, Berkeley CA 94704, USA
vstantch@icsi.berkeley.edu
[2] Technische Universität Berlin, Berlin, Germany
[3] FOM Fachhochschule fuer Oekonomie und Management, Berlin, Germany

Abstract. Emerging grid computing infrastructures such as cloud computing can only become viable alternatives for the enterprise if they can provide stable service levels for business processes and SLA-based costing. In this paper we describe and apply a three-step approach to map SLA and QoS requirements of business processes to such infrastructures. We start with formalization of service capabilities and business process requirements. We compare them and, if we detect a performance or reliability gap, we dynamically improve performance of individual services deployed in grid and cloud computing environments. Here we employ translucent replication of services. An experimental evaluation in Amazon EC2 verified our approach.

Keywords: QoS and SLA Negotiation, Assurance, Service-oriented computing.

1 Introduction

Service-oriented architecture (SOA) is an architecture that combines elements of software architecture and enterprise architecture. It is based on the interaction with autonomous and interoperable services that offer reusable business functionality via standardised interfaces. Services can exist on all layers of an application system (business process, presentation, business logic, data management). They may be composed of services from lower layers, wrap parts of legacy application systems or be implemented from scratch. Typically, services at the business process layer are described as *business services*, while services at the lower implementation level are described as *technical services*.

1.1 Emerging Grid Computing Infrastructures for Services

Datacenters and cloud computing environments are grid computing infrastructures that are emerging as platforms for provision of technical services. An example for such an environment is Amazon EC2, recently evaluated from the user perspective at Harvard [1]. The development and extension of tools to monitor and control such infrastructures is part of large research projects, e.g., at Stanford and UC Berkeley [2]. On the other side, the mapping of business process requirements at the infrastructure level in such environments is rarely addressed.

N. Abdennadher and D. Petcu (Eds.): GPC 2009, LNCS 5529, pp. 25–35, 2009.

1.2 Challenges

A successful service offering has two main objectives: to provide the needed functionality and to provide the needed Quality of Service (QoS). QoS parameters are part of the run-time related nonfunctional properties (NFPs) of a service and present one of the main research challenges in service-oriented computing [3]. Contrary to design-time related NFPs (e.g., language of service or compliance), run-time related NFPs are performance oriented (e.g., response time, transaction rate, availability). They can change during runtime – when times of extensive concurrent usage by many users are followed by times of rare usage, or when failures occur.

An approach to measure and dynamically adapt service performance in grid and cloud computing environments to such changes can ensure continuous meeting of service levels defined at the business level. This is an even more challenging task in such IT infrastructures that are not owned or controlled directly by the enterprise. Specifically, such approach should consider service reconfiguration at runtime, as changes in service implementation are not realistic.

NFPs of services (both technical and human, as well as their combinations) are typically specified in Service Level Agreements (SLAs). They are typically defined at the level of a business process but need to be addressed at the level of IT infrastructures. Thereby several technical services are orchestrated in order to provide business services for a business process. SLAs are negotiated between the process owner and the service provider who have to agree upon them.

This work proposes a straightforward way to negotiate business process SLAs between a process owner and a service provider and to enforce these SLAs at the level of grid and cloud computing infrastructures – if we formalize both the service level requirements of the process owner and the capabilities of the technical services in the grid (cloud) using a similar structure, we can compare them in an automated way. Based on such comparisons, we can negotiate and provide optimized service configurations in the grid (cloud) and thereby enforce the SLAs of the business process in the QoS characteristics and the service levels of these technical services.

1.3 Work Structure

The remainder of this work is structured as follows: Section 2 gives an overview of our proposed approach and puts it in the context of related research in the areas of service-oriented computing, QoS-aware platforms and grid workflow optimization. Section 3 describes the formalization of service and QoS levels as a main precondition for the negotiation of SLAs. Section 4 deals with performance and availability as NFPs that are representative for the approach and their enforcement in grid and cloud computing environments. In Section 5, we present an experimental evaluation of the approach. Thereby our solution was deployed in Amazon EC2 and demonstrated the viability and applicability of SLA formalization and subsequent QoS enforcement in cloud computing infrastructures.

2 An Approach for SLA Mapping

When a company is in control of its internal IT infrastructure, business analysts and developers can define service level requirements during design time and can actively select and influence components in order to meet these requirements. However, in cloud computing environments SLAs are typically provided for basic platform services (e.g., system uptime, network throughput). Business processes typically expect service levels for the technical services they integrate (e.g., order submission in less than 1 second). How can we bring these two worlds together?

This work proposes an approach for SLA mapping between business processes and IT infrastructures. It is based on a method for the assurance of NFPs and includes three major tasks (see Figure 1): (i) Formalization of business process requirements at the business side and of service capabilities at the IT infrastructure. Both are specified in a formal way, using a predefined service level objective structure and predefined NFP terms. (ii) Negotiation of service capabilities at the IT infrastructure that correspond to the formalized business process requirements: Here, we assess whether the aggregated technical services provide the expected service levels to meet business process requirements under different load hypotheses. Within this comparison we also calculate the aggregated service level, using the performance metrics of the individual technical services. Based on the result of this comparison we can decide where to apply replication in the next step. A reasoner or comparing unit must understand both structure of the statements and used NFPs on the business and the infrastructure side. (iii) Enforcement of business process SLAs at the IT infrastructure level: Here, we apply translucent parallelization of service processing using multiple nodes in a datacenter environment [4]. Replication can be enacted to improve service levels regarding response time, transaction rate, throughput and availability, respectively reliability.

2.1 Related Work

The SOA-specific aspects of major architectural concerns, such as service visualization [5], integration [6], and service selection [7] have been consistently

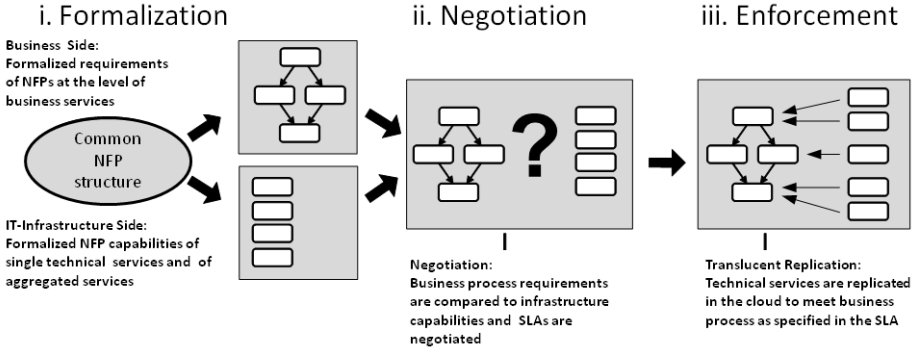

Fig. 1. Approach Overview

addressed by researchers. The existing standards for specification of QoS characteristics in web service environments can be grouped according to their main focus: software design/process description (UML Profile for QoS and QML - QoS Modeling Language [8]), service/component description (WS-Policy) and SLA-centric approaches (WSLA - Web Service Level Agreements [9], WSOL - Web Service Offerings Language [10], SLAng - Service Level Agreement definition language [11] and WS-Agreement [12]).

Much work has been done in the area of QoS-aware web service discovery [13], QoS-aware platforms and middleware [14,15,16,17], and context-aware services [18]. However, all of these approaches do not address adaptive enforcement of NFPs, but rather deal with the composition of services where the aggregation of predefined NFP levels would satisfy a specific requirement. Of particular interest are approaches that allow the "gridification" of specific applications, e.g., in [19] where a set of programs for inferring evolutionary trees is ported to the grid platform XtremWeb-CH [20]. Approaches such as shared memory have also been proposed for such tasks [21]. There is ongoing research in the area of adaptive optimization, more specifically in the areas of grids (e.g., grid workflow optimization [22]) and parallel database operations [23].

3 Formalization and Negotiation of SLAs

Figure 2 shows the structure we have recently proposed [24,25] for formalization of business process service level objectives (SLOs) and technical service capabilities. The figure also contains sample service level statements about response time, throughput and transaction rate. These statements are then stored with the service description (service capabilities) respectively with the business process definition (business process SLO) and are the starting point for the negotiation of SLAs. An example for a statement about the service capability is "The transaction rate of the service is higher than 90 transactions per second in 98% of the cases as long as throughput is higher than 500 kB/s." An example for a

	NFP	Predicate	Metric (Value, Unit)	Percentage	Qualifying conditions (QC)		
SLO pattern					NFP	Predicate	Metric
SLO examples	Response time	less than	100 ms	in 95 % of the cases **if**	transaction rate	less than	10 tps
	Throughput	higher than	1000 kB/s	in 95 % of the cases **if**	transaction rate	less than	10 tps
	Transaction rate	higher than	90 tps	in 98 % of the cases **if**	Throughput	higher than	500 kB/s

Fig. 2. Structure of Service Level Objectives (SLOs) and Examples (tps - Transactions per second)

requirement business process level is "The transaction rate of the process should be higher than 50 transactions per second in 97% of the cases while throughput is higher than 500 kB/s."

In this work we deal with the negotiation and the enforcement of performance-related NFPs as part of SLAs and therefore focus on their formalization. However, our scheme can also be used to describe further aspects of SLAs, e.g., design-time related NFPs, such as cultural (e.g. language), legal (e.g. Sarbanes-Oxley-Compliance, Basel-II-Compliance), organizational (partner list), service usage-related (e.g. GUI simplicity) and trust-related (e.g. Customer rating, experience of provider).

In our proposed approach, the process owner specifies service level requirements as expected from the business perspective. At the IT infrastructure side we evaluate different replication configurations of technical services in grid and cloud computing environments such as Amazon EC2. Both requirements and service capabilities of the different configurations are then formalized and compared. Furthermore, we can start a negotiation between process owner and service provider based on this comparison. Thereby, the replication configuration that meets (or is closest to) the business requirements is selected and is the starting point for the actual SLA. The transparent cost model of Amazon Web Services allows us to put different price tags for the required service levels (e.g., a business process that needs higher transaction rates has to pay more), thus allowing for real activity-based IT costing. Furthermore, we can use the SLO structures in a supply-oriented way, contrary to the demand-oriented approach we present here. Thereby, we can specify combinations of NFP levels that represent different generic SLAs (e.g., "Gold", "Standard"', "Cost-optimized" [26], or "Time-Critical", "Load-Critical", "Dependability-Critical").

4 QoS Enforcement of SLAs in Grid and Cloud Computing Environments

In order to satisfy the SLAs of a business process we should look at ways to represent and control NFPs at the level of a technical service. While cloud computing environments specify service levels for basic platform operations (e.g., system uptime), we focus on the improvement of service levels for specific technical services that are composed to provide the needed business service. One example is the composition of the technical services `GetOrder()` and `ClearPayment()` to provide the business service *Order Placement*. In this work we show exemplary how we can improve performance-oriented NFPs, particularly response time and transaction rate, as well as dependability for single and composed technical services.

4.1 Performance

A general and broadly accepted definition of performance is to observe the system output $\omega(\delta)$ that represents the number of successfully served requests (or transactions) from a total of input $\iota(\delta)$ requests during a period of time.

$$\omega(\delta) = f(\iota(\delta)) \tag{1}$$

This definition of performance corresponds to transaction rate as NFP – the system guarantees to process n requests during time period t. The performance of a serial composed service chain is determined by the performance of the service with the lowest performance. Let us assume that we compose a service chain from Service 1, Service 2 and Service 3. If Service 1 and Service 3 are providing high transaction rate (e.g., 500 requests per second) and Service 2 is providing a much lower transaction rate (e.g., 50 requests per second), our composed service will only serve 50 requests (or actually less than 50 requests) per second overall. We can easily calculate the average response time from the transaction rate by dividing the time period through the number of requests. Furthermore, we can also measure further performance metrics, such as worst-case execution time (WCET) if we need to specify them in the SLA.

When we introduce parallelism through functional replication we can ideally double the processing performance. The replication of the service with the lowest transaction rate (Service 2) leads to an overall increase of the transaction rate for the composed service. Therefore, replication has advantageous effects on service chain performance when no replica synchronization is required. This includes transaction rate, throughput and response time as parameters of SLAs.

4.2 Dependability

Dependability integrates several attributes: availability, reliability, safety, integrity, and maintainability. These are defined as follows [27]:

- availability denotes the readiness to provide a correct service,
- reliability denotes the continuity of service provision,
- safety is an attribute that assures there are no catastrophic consequences on the user and the environment.
- integrity denotes that there are no improper changes of the system.
- maintainability denotes that a system can undergo changes and repairs.

There are four categories of approaches to attain dependability [27]: fault prevention, fault tolerance, fault removal, and fault forecasting. In the context of this work we focus on availability as attribute and on fault tolerance as approach to attain it.

Availability needs to be quantified in the SLAs so that we can negotiate and enforce it. It has been defined traditionally as a binary metric that describes whether a system is "up" or "down" at a single point of time. A common extension of this definition is to compute the average percentage of time that a system is available during a certain period – this is a typical availability measure that describes a system as having `99.999%` availability, for example.

There are several extended definitions of availability that address the inherent limitations of the traditional definition – availability should be considered as a spectrum, rather as a binary metric, as systems can have various degraded, but operational, states between "up" and "down". Furthermore, the definition does not consider QoS aspects.

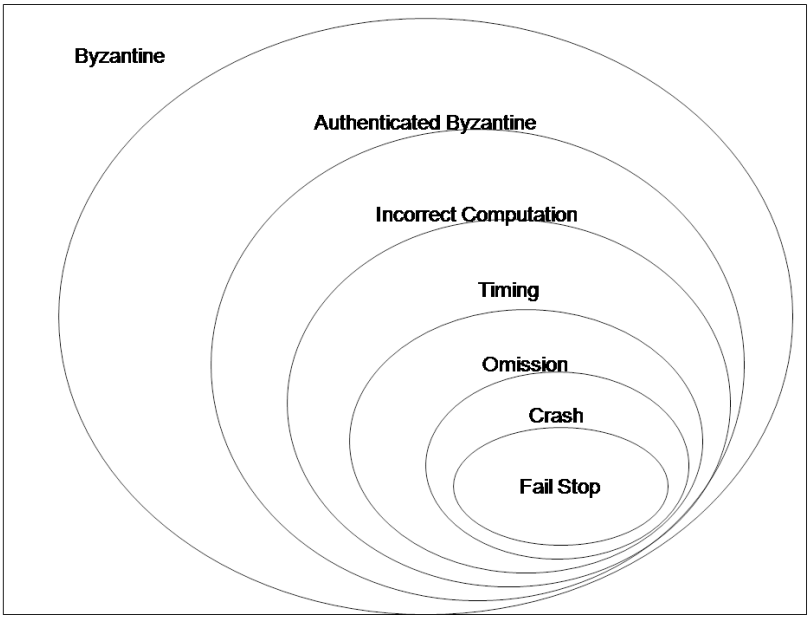

Fig. 3. Fault Model used in the SLO Structures

One possibility is to measure availability by examining variations in system QoS metrics over time [28]. The authors state that the particular choice of QoS metrics depends on the type of system and suggest performance and degree of fault tolerance as obvious metrics for server systems. In this case, performance would mean requests satisfied (successfully served) per second. This corresponds to our definition of performance in the previous subsection. In order to specify degrees of fault tolerance we need an underlying fault model. We use a model that was also used in [29] (see Figure 3). The model was originally proposed in [30] and extended in [31]. It incorporates several aspects that are typical for technical services in cloud computing environments, as compared to traditional distributed systems (e.g., trust issues).

One recent adaptation of traditional methods for better availability to the world of SOA is proposed in [32]. It involves replication of technical services across multiple, wide-area sites. Typically, we need to provide strong consistency between the replicas in order to provide better availability. This makes the application of the approach problematic, particularly in cloud computing environments - the overhead we introduce to ensure strong consistency generally has a negative impact on performance.

4.3 Evaluation and Improvement of IT Infrastructure Capabilities

While these general aspects of replication are hardly surprising, there are different ways *where* and *how* we can replicate technical services in cloud computing

environments. The concept of architectural translucency [4] defines three levels for replication in SOA platforms (hardware, operating system, and serviceware) and proposes replication techniques and mechanisms for the evaluation of their effects on NFPs [33,34]. Using such concepts we are able to evaluate different replication configurations at the level of IT infrastructure, formalize the results of these evaluations as SLOs, and select the configuration that best meets the requirements of the business process.

5 Experimental Evaluation

For a series of experimental evaluations we deployed WSTest 1.1 [35] as a generic benchmark in Amazon EC2 [1] as a grid and cloud computing infrastructure. Our business process requirements were specified in SLO structures similar to the one in Figure 2. We tested different configurations of translucent replication as specified in [34,33] with the objective to find settings that best match the SLOs. Client requests were simulated using the second of the two test methodologies described in [35] and Mercury LoadRunner's SOAP client. Specifically, 25 client machines that closed the connection at the end of each request were used. Each of these machines (Dell, 2 GHz Core Duo, 2 GB RAM, Gigabit Ethernet) runs Mercury LoadRunner agents and can generate approx. 2000 concurrent requests. Every replication setting was tested for 120 minutes with 1 second think time before a request. This corresponds to some 7200 requests that were sent to each setting. These 120 minutes tests were automated and repeated 10 times on 21 consecutive days. Figure 4 shows an overview of the results. All replication settings provided better service levels as the original configuration. Furthermore, there were two replication settings (3 and 4) that satisfied the requirements of the

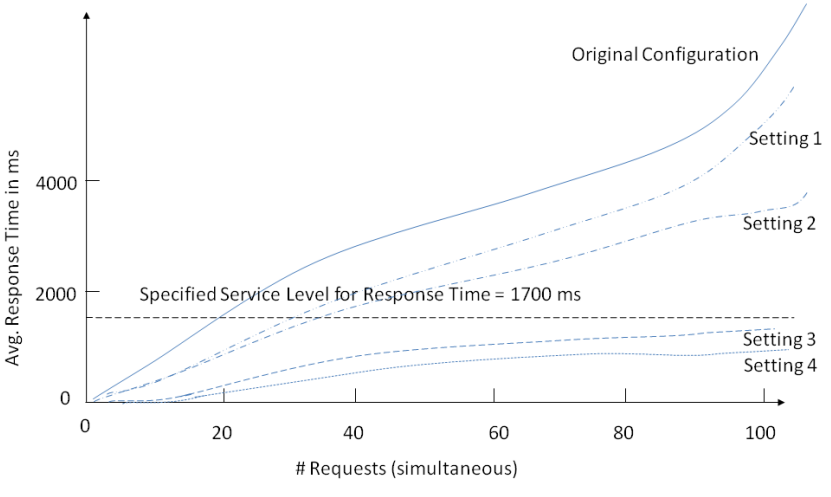

Fig. 4. Overview of Results

business process and thus allowed the mapping of its SLAs at the level of the IT architecture. These two settings caused only marginally higher costs compared to the original configuration in Amazon EC2.

6 Conclusion and Outlook

We presented an approach for negotiation of SLAs of business processes and the corresponding QoS enforcement at the level of IT infrastructure. It consists of three main tasks. The formalization of NFPs allows us to compare required and existing service levels. Using translucent replication we can meet expected service levels by automatically reconfiguring service replicas in cloud computing environments. The experimental evaluation demonstrated that we could improve service levels by over 50 % under certain load hypotheses. Furthermore, it demonstrated that we are able to keep business process service levels as specified in the SLAs continuously. Providers of cloud computing environments typically offer very flexible and detailed cost accounting, so the costs of providing and enforcing different service levels for business processes are transparent. This allows us to taylor QoS levels to the specific requirements of every business process and we support concepts such as activity-based costing in an enterprise. We are currently working on a user interface that will allow process owners to set their expected preferences regarding NFPs in an easy and convenient way. Thereby, a predefined set of service levels for response time, availability and other NFPs will correspond to a simple description (e.g. gold, standard, cost-optimized). Users will be able to select a setting using a simple user interface such as a slider and the infrastructure will automatically adapt to this setting. Furthermore, we are currently investigating ways to derive such preferences from existing processes and incorporate them in the process model repository as SLOs which will ultimately result in the automatic provision of service compositions that best meet the functional and nonfunctional requirements of the business process. Furthermore, we also plan to address limitations of distributed replication of services with respect to availability and particularly the trade-off communication vs. availability [32].

References

1. Garfinkel, S.: An evaluation of amazon's grid computing services: Ec2, s3 and sqs. Technical report tr-08-07, School for Engineering and Applied Sciences, Harvard University, Cambridge, MA (July 2007)
2. Bodık, P., Fox, A., Jordan, M.I., Patterson, D., Banerjee, A., Jagannathan, R., Su, T., Tenginakai, S., Turner, B., Ingalls, J.: Advanced Tools for Operators at Amazon.com. In: The First Annual Workshop on Autonomic Computing (2006)
3. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. Computer 40(11), 38–45 (2007)
4. Stantchev, V., Malek, M.: Architectural Translucency in Service-oriented Architectures. IEE Proceedings - Software 153(1), 31–37 (2006)

5. Eicker, S., Spies, T., Kahl, C.: Software Visualization in the Context of Service-Oriented Architectures. In: Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis (Vissoft 2007), pp. 108–111. IEEE, Los Alamitos (2007)
6. Zhang, J., Chang, C.K., Chung, J.-Y., Kim, S.W.: Ws-net: a petri-net based specification model for web services. In: IEEE International Conference on Web Services, 2004. Proceedings, July 6-9, 2004, pp. 420–427 (2004)
7. Reinicke, M., Streitberger, W., Eymann, T.: Evaluation of Service Selection Procedures in Service Oriented Computing Networks. Multi Agent and Grid Systems 1(4), 271–285 (2005)
8. Frolund, S., Koistinen, J.: Quality of services specification in distributed object systems design. In: COOTS 1998: Proceedings of the 4th conference on USENIX Conference on Object-Oriented Technologies and Systems (COOTS), Berkeley, CA, USA, p. 1. USENIX Association (1998)
9. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Service Level Agreement (WSLA) Language Specification. IBM Corporation (2002)
10. Tosic, V., Patel, K., Pagurek, B.: WSOL-Web Service Offerings Language. In: Bussler, C.J., McIlraith, S.A., Orlowska, M.E., Pernici, B., Yang, J. (eds.) CAiSE 2002 and WES 2002. LNCS, vol. 2512, pp. 57–67. Springer, Heidelberg (2002); (revised papers)
11. Lamanna, D.D., Skene, J., Emmerich, W.: SLAng: A Language for Defining Service Level Agreements. In: Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS, pp. 100–106 (2003)
12. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Global Grid Forum GRAAP-WG, Draft (August 2004)
13. Makripoulias, Y., Makris, C., Panagis, Y., Sakkopoulos, E., Adamopoulou, P., Pontikaki, M., Tsakalidis, A.: Towards Ubiquitous Computing with Quality of Web Service Support. Upgrade, The European Journal for the Informatics Professional VI(5), 29–34 (2005)
14. Yau, S.S., Wang, Y., Huang, D., Hoh, P.: Situation-aware contract specification language for middleware for ubiquitous computing. In: The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003. FTDCS 2003. Proceedings, May 28-30, 2003, pp. 93–99 (2003)
15. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
16. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1069–1075 (2005)
17. Solberg, A., Amundsen, S., Aagedal, J.Ø., Eliassen, F.: A Framework for QoS-Aware Service Composition. In: Proceedings of 2nd ACM International Conference on Service Oriented Computing (2004)
18. Tokairin, Y., Yamanaka, K., Takahashi, H., Suganuma, T., Shiratori, N.: An effective qos control scheme for ubiquitous services based on context information management. cec-eee, 619–625 (2007)
19. Abdennadher, N., Boesch, R.: Deploying phylip phylogenetic package on a large scale distributed system. In: IEEE International Symposium on Cluster Computing and the Grid, pp. 673–678 (2007)

20. Abdennadher, N., Boesch, R.: A scheduling algorithm for high performance peer-to-peer platform. In: Lehner, W., Meyer, N., Streit, A., Stewart, C. (eds.) Euro-Par Workshops 2006. LNCS, vol. 4375, pp. 126–137. Springer, Heidelberg (2007)
21. Ibach, P., Stantchev, V., Keller, C.: Daedalus a peer-to-peer shared memory system for ubiquitous computing. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 961–970. Springer, Heidelberg (2006)
22. Wanek, H., Schikuta, E., Haq, I.U.: Grid workflow optimization regarding dynamically changing resources and conditions. Concurrency and Computation: Practice and Experience (2008)
23. Schikuta, E., Mach, W.: Optimized workflow orchestration of parallel database aggregate operations on a heterogenous grid. In: The 37th International Conference on Parallel Processing (ICPP 2008), Portland, Ohio, USA. IEEE Computer Society, Los Alamitos (2008)
24. Stantchev, V., Schröpfer, C.: Techniques for service level enforcement in web-services based systems. In: Proceedings of The 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS 2008), pp. 7–14. ACM, New York (2008)
25. Krallmann, H., Schröpfer, C., Stantchev, V., Offermann, P.: Enabling autonomous self-optimization in service-oriented systems. In: Proceedings of The 8th International Workshop on Autonomous Systems - Self Organisation, Management and Control, Berlin, New York, pp. 127–134. Springer, Heidelberg (2008)
26. Schropfer, C., Binshtok, M., Shimony, S.E., Dayan, A., Brafman, R., Offermann, P., Holschke, O.: Introducing preferences over NFPs into service selection in SOA. In: International Conference on Service Oriented Computing - International Workshop on Non Functional Properties and Service Level Agreements in Service Oriented Computing (2007)
27. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing 1(1), 11–33 (2004)
28. Brown, A., Patterson, D.A.: Towards Availability Benchmarks: A Case Study of Software RAID Systems. In: Proceedings of the 2000 USENIX Annual Technical Conference (2000)
29. Polze, A., Schwarz, J., Malek, M.: Automatic generation of fault-tolerant corba-services. Tools, 205 (2000)
30. Cristian, F., Aghili, H., Strong, R., Dolev, D.: Atomic broadcast: from simple message diffusion to byzantine agreement. Inf. Comput. 118(1), 158–179 (1995)
31. Laranjeira, L.A., Malek, M., Jenevein, R.: Nest: a nested-predicate scheme for fault tolerance. Transactions on Computers 42(11), 1303–1324 (1993)
32. Yu, H., Vahdat, A.: The costs and limits of availability for replicated services. ACM Trans. Comput. Syst. 24(1), 70–113 (2006)
33. Stantchev, V.: Effects of Replication on Web Service Performance in WebSphere. Icsi tech report 2008-03, International Computer Science Institute, Berkeley, California 94704, USA (February 2008)
34. Stantchev, V., Malek, M.: Addressing Web Service Performance by Replication at the Operating System Level. In: ICIW 2008: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services, pp. 696–701. IEEE Computer Society, Los Alamitos (2008)
35. Microsoft. Comparing Web Service Performance: WS Test 1.1 Benchmark Results for .NET 2.0, NET 1.1, Sun One/ JWSDP 1.5 and IBM WebSphere 6.0 (2006), `http://www.theserverside.net/tt/articles/content/NET2Benchmarks`