Energy-Aware Service Execution

Waltenegus Dargie, Anja Strunk, and Alexander Schill
Chair for Computer Networks
Faculty of Computer Science
Technical University of Dresden
01062 Dresden, Germany
Email: {waltenegus.dargie, anja.strunk, alexander.schill}@tu-dresden.de

***Abstract*—The energy consumption of ICT infrastructures has increased considerably in the recent years. This has resulted in extensive research on dynamic power management strategies as well as data centre design and placement. The main problem with most of the proposed or existing approaches is that they do not fully take the distributed nature of and strong logical dependencies between executed services into account. However, without a comprehensive knowledge of the wider relationships between services, local power management strategies may be ineffectual or can even result in high aggregate energy cost. Understanding this relationship is useful for fine-grained energy-aware computing. For example, services that run on underutilised servers can be stopped or seamlessly migrated to other servers, so that the underutilised servers can be turned off. Alternatively, a re-binding process can be used if the cost of service migration is high. Such advantages can be fully exploited if the dependency between services is properly understood and meaningfully modelled. This paper introduces a conceptual architecture for an energy-aware service execution platform and compares three optimisation mechanisms to support dynamic service migration and rebinding.**

***Index Terms*—Context-awareness, dynamic power management, energy-efficient servers, power consumption estimation, service execution, service oriented architecture**

I. Introduction

The energy consumption of servers and data centres is becoming a significant research issue. The issue is particularly interesting, since high energy consumption does not necessarily correspond with high performance. Figure 1 displays an actual measurement of the energy consumption and workload distribution of a high performance computer (SGI Altix 4700 featuring 1024 dual-core Intel Itanium processors and 6.5 TB main memory) at the Centre for Information Services and High Performance Computing (ZIH), Technical University of Dresden. As can be seen in the figure, even though the workload significantly varied throughout the day, the energy consumption of the server remained fairly constant.

A wide range of power management strategies have been proposed to optimise the energy consumption of computing systems. At the macro level, some of these strategies aim at optimal placement of servers and services as well as data [24], chiefly to reduce the energy cost of cooling and communication. At the micro level, energy-aware scheduling [20], resource virtualisation [14], and service consolidation [23] have been proposed, chiefly to reduce the energy cost of

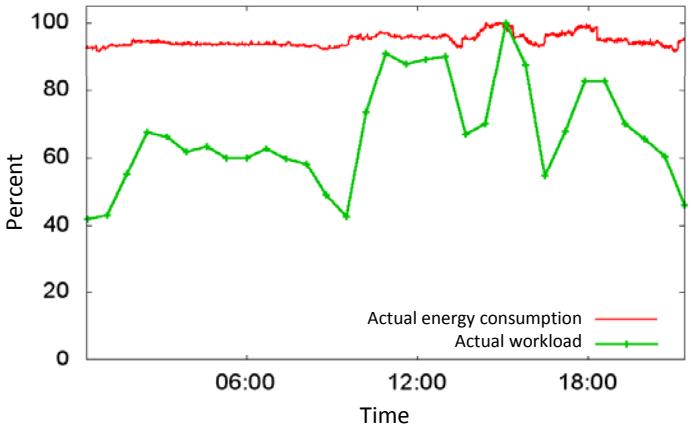

Fig. 1. A disproportional energy consumption of a high performance server at the Technical university of Dresden. The measurement was taken on June 20, 2008.

runtime service execution. Complementary to these, dynamic voltage and frequency scaling [16] as well as dynamic and selective switching on/off of hardware resources have been proposed to reduce the idle power consumption of servers. The main problem with most of the proposed or existing approaches is that they do not fully take the distributed nature of and strong logical dependencies between executed services into account. Without a comprehensive knowledge of the wider relationships between services, power management strategies can be either ineffectual or even result in high aggregate energy cost [25].

Traditionally, applications have been developed from a scratch; all the services that make up an application belong to it at all times. This practice is now being replaced by the service-oriented architecture in which services are independently developed, deployed, and managed. Applications are described as abstract business processes that can be executed by invoking a number of available services at runtime. An essential aspect of this approach is that service providers and their customers negotiate for utility based Service Level Agreements (SLAs) to determine the costs and penalties of service execution based on the achieved performance levels.

From a service-oriented architecture point of view, the placement of computational resources can be described using

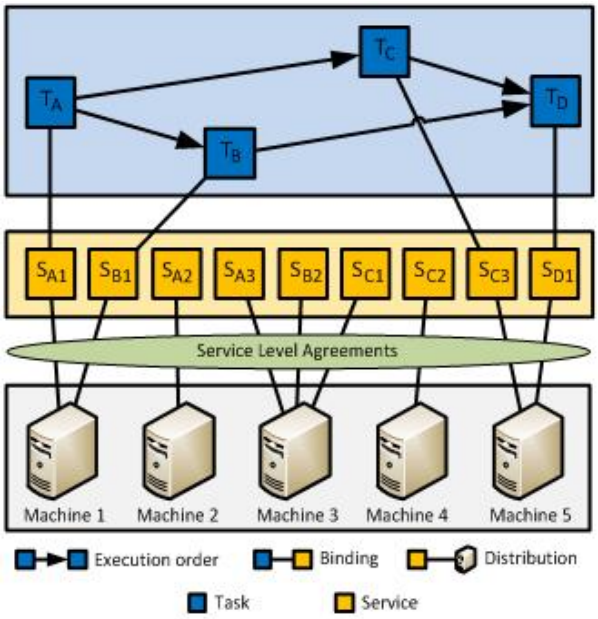

Fig. 2. Relationship between applications, services, and hardware resources. Top: application layer. Middle: service layer. Bottom: computing resources and the software that manges them

Figure 2. At the bottom, there are hardware resources and operating systems that manage them. In the middle, there are distributed software services which provide certain functionalities to the top layer process descriptions. At the top, there are applications which are described as business processes (workflow). These applications bind to the middle layer services at runtime. Logically, a service execution platform resides in the middle layer to facilitate service discovery, selection and binding; and to ensure that contractual agreements are respected by the applications as well as the service providers.

An application or a composition thereof consists of several subtasks, each subtask directly binding to a software service. During a runtime execution, services are invoked according to their execution order; and data between the services are exchanged according to the data flow structure of the composition. In Figure 2, the service composition starts with the invocation of the service S_{A1} as an implementation of the subtask T_A, followed by the concurrent executions of the services S_{B1} and S_{C3} as the implementations of subtasks T_B and T_C, respectively. The composition finishes with the execution of S_{D1} to carry out the subtask T_D. The infrastructure provider guarantees each service a formally established quality of service as defined by the SLAs. Hence, the infrastructure provider distributes the hosted services among the physical machines to ensure that the SLAs are satisfied even at the system's peak load. In most cases, the physical machines that host the services are utilised between 10 and 60 percent of their peak capacity and, hence, a static distribution of services is inefficient [1].

Understanding the distributed nature of service execution is useful to avoid conditions that lead to underutilising or overloading servers. The separation of concerns between the application development (composition) and the service deployment as well as the hardware infrastructure usage and the infrastructure provision and management enables adaptive and energy aware decisions to be made at various levels of abstractions. At the application level, the order of service execution can be rearranged to defer service execution for a latter time to prevent servers from being temporarily overloaded. Alternatively, services can be executed ahead of their schedule in case a server is idle or underutilised. At the service level, service rebinding and service migration can be carried out to consolidate services and to run an optimal number of servers in a cluster. Unlike previous approaches focusing on live migration of virtual machines (which are potentially responsible for managing several services) [7], service migration is fine grained, efficient, and quick. Moreover, as complementary to service migration, dynamic service rebinding enables to bind to alternative services in case the cost of migration (particularly, in terms of execution delay) violates some existing SLAs.

In this paper, a conceptual architecture for supporting energy-aware service execution is presented. The architecture enables energy-aware adaptations at various levels of abstraction. Moreover, the paper compares three types of optimisation strategies – taboo search, genetic algorithm, and simulation annealing – that enable fast and efficient service rebinding.

The rest of this paper is organised as follows: Section II summarises related work. Section III discusses an energy-aware service execution and presents a conceptual architecture to support dynamic service migration and rebinding. In Section IV, three types of optimisation strategies to enable dynamic rebinding are discussed and evaluated. Finally, in Section V, a discussion is given to summarise the contributions and observation of this work and to layout a future work.

II. Related Work

Fan et al. [12] investigate the energy consumption of Google's web service infrastructure and find out that the idle state power consumption of most of the servers is above 50% of the peak power consumption. The same study suggests that the amount can be reduced to 10% of the peak power consumption by implementing power management strategies. The study, however, does not indicate how the reduction can be achieved. A comprehensive analysis of the net productivity of enterprise IT operating as a whole system (servers, storage, tape, networking, power distribution, and cooling systems) is given in [3]. Similarly, Hamilton [15] introduces the concept of work done per dollar and work done per joule to measure the efficiency of servers. The relevance of the study is in its practical analysis of the full-load performance of existing servers. Jourjon et al. [18] analytically model the energy consumption of a peer-to-peer content distribution network. The model extends the one initially proposed by Irani et al. [17], which expresses the energy consumption of wireless sensor networks. Both models attempt to quantify the cost of data processing and communication as well as the idle state

of servers. The model of Jourjon et al. is used by an off-line scheduler to estimate the number of distributed servers required to efficiently support the exchange of multimedia content of known volume. While the communication aspect of the analytic model is not relevant to the us, the general approach to quantify the energy cost of various concerns is interesting.

Clark et al. [7] introduce the idea of live Virtual Machine migration as a mechanism to consolidate computing resources, reduce management complexity and speed up the response to business dynamics. They employ a pre-copy mechanism in which pages of memory are iteratively copied from the source machine to the destination host. A page level protection hardware is used to ensure a consistent transfer of data. Likewise, Bradford et al. examine the delay associated with live migration of a VM, both within a local area network and a wide area network environments. Whereas the migration within a local area network environment takes a few seconds (three seconds), migration across a wide area network takes 68 seconds. Liu et al. [19] propose a conceptual architecture to support live VM migration, which consists of a managed environment, a migration manager, and a monitoring service. The managed environment consists of a host of computing resources (virtual machines, physical machines, remote commands on VMs and applications) whose operation can be adaptively controlled. The migration manager is responsible for triggering live migration and makes decisions pertaining to the placement of virtual machines on physical servers. The monitoring service monitors resource utilization, the workload created by applications, and the power consumption of physical servers. It is the main source of knowledge regarding the workload and power distribution in the cloud. A heuristic algorithm is used to estimate workload distribution and trigger migration. While migration is a useful strategy, it is not the aim of this paper to strive for VM migration. There are two reasons for this: (1) In a service-oriented environment, a single VM may be used by a large number of services; in which case, VM migration cannot take place without first migrating or de-installing all the services it hosts. (2) It may not be necessary to migrate inactive services that are managed by a VM. In contrast, we focus on migration and re-binding at the service level to achieve a more flexible and efficient way of energy-aware computing.

Binder and Suri [2] classify hardware resources in the Cloud as dispatchers, file servers, and compute servers. The first two run permanently while a dynamic power management technique is applied on the compute servers based on the present and anticipated workload distribution in the cloud. A probabilistic algorithm is employed on the dispatcher servers to determine (1) to which of the compute servers requests should be dispatched for processing; and (2) which of the compute servers should be turned off. Each compute server hosts a node manager for monitoring idle time and average service response time (metrics relevant for the probabilistic algorithm to estimate workload and transition periods). The service dispatch algorithm is in many respects similar to most process scheduling and load balancing algorithms in operating systems as well as job scheduling in grid computing. The main difference is that whereas the latter algorithms aim at performance (CPU utilisation, throughput, response time, turnaround time) and fairness, the former aims at consolidating services and keeping the number of running servers as small as possible without infringing service level agreements. The proposed algorithm and the classification of resources as a key aspect of the power management technique are most suitable for client-server architectures. Moreover, service requests (and thereby, service arrival rates) are considered to be independent from each other. Apparently, in a service oriented computing environment, services are logically dependent on other services. Consequently, local energy-aware adaptations certainly have impact on the energy consumption of servers elsewhere. The magnitude of this impact should be quantified to justify local adaptations.

Chen et al. [4] propose a statistic-based energy-efficient workload generation for MapReduce computation. The work exploits knowledge of inter-job arrival rates as well as the ratio between input and shuffle, shuffle and output and output and input data to make decisions concerning cluster size and configuration parameters such as the number and distribution of workers in a cluster. Similar work by the same authors [5] provides an extensive analysis of the energy consumption of various concerns (read/write operations, distributed file system replication, sorting, shuffle operations that involve memory, network cards and disks, etc.) of a MapReduce job. The main contribution of this latter work is its quantification of the energy consumption of servers in terms of the amount of useful data they processed (for example, the amount of sorted records per joule). Based on the analysis, the authors propose a simple energy model that can be employed to estimate an optimal workload distribution. The model, though applied in a narrow context, can be extended to analyse the energy consumption of logically dependent distributed services.

III. Architecture

Figure 3 displays a partial view of the conceptual architecture of the energy-aware service execution platform. The main task of the platform is to search and bind to the services which accomplish the tasks that are described in the business process description (workflow) at the application level. The services are distributed and selectively replicated in a cluster of servers. Additionally, the service execution platform monitors the violation of SLA's during service execution and ranks the reliability of services for future binding.

To achieve energy-awareness, the service execution platform gathers context information from all levels of abstractions and carries out adaptive decisions in return. At the application layer, it rearranges the execution order of services and differs service execution for a latter time in case the current execution order overloads or underutilise resources. At the service layer, it starts and stops services and moves them from server to server. At the infrastructure level, it selectively switches on or off hardware resources.

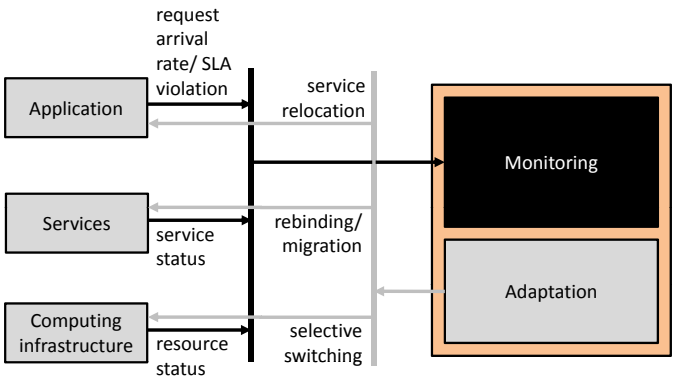

Fig. 3. A partial view of the energy-aware service execution

The existence of logical dependencies between the distributed services that make up an application is the basis for energy-aware service execution. Knowledge of these dependencies is formally obtained from a workflow and the Service Level Agreements between the application and the actual services to which it binds. From the workflow, order of service execution, input-output dependencies and temporal dependencies can be obtained. The Service Level Agreements set constraints on, among other things, service response time and data quality.

A. Context

Unlike previous approaches, we believe that the use of context [9], [10] plays an important role in energy-aware service execution. At the infrastructure level, knowledge of the power consumption of the hardware resources, the efficiency and capacity of the power supply unit, the different operational modes (in terms of voltage and frequency) of the hardware subsystems, etc. is useful. At the service level, the status of the services, the size of the services, the different operations they support (read, write, update (a combination of read and write operations), search, filter, communicate, etc.) as well as the average energy they consume when these operations are carried out, are useful. At the application level, user requests arrival rates, average read operations per user, average write operations per user, ratio of read and write operations in update requests, etc., are useful.

For example, the voltage levels of the power supply units of most light weight servers are directly affected by the amount of current drawn from the supply units. Whereas some power consumption estimation models assume an invariable (constant) voltage (3.3V, 5V, 12V) during the calculation of power consumption, this knowledge should be taken into account both when estimating the power consumption of a future workload and when modelling underutilising and overloading conditions. Figure 4 shows how the +5 V supply line that connects the power supply unit with the hard disk, which was connected to an AMD Dual Core Athlon 64 X2 server, varies during a write operation. As can be seen, the supply voltage (which should be constant) fluctuates throughout, but as the hard disk draws a significant amount of current, the fluctuation got worse, thereby, significantly reducing the efficiency of the power supply (in that the current that can be drawn from the power supply reduced). This implies that, an extended heavy write-workload can potentially cause the hard disk to consume a power which is not proportional to the work done (in terms of the number of bits written per watt).

Likewise, a stochastic knowledge of the power consumption of individual hardware resources during the transition between active and off states is useful to make realistic estimation of the cost of power transitions. Again, in the literature, power transition costs are modelled as static quantities. For example, Chiasserini and Rao [6] use a static approach to estimate the minimum shut-down state of a computer to justify selective switching:

$$\tau_{th} = max\left(0, \frac{(P_{on} - P_{off,on})\, t_{off,on}}{P_{off} - P_{on}}\right) \qquad (1)$$

where P_{on} is the mean power consumption in the idle (underutilised) state; $P_{off,on}$ is the transition cost (from the off to an active state) in terms of power consumption; and P_{off} is the power cost of the server in the off state. The time period $\tau_{th} + t_{off,on}$ is defined as the minimum break-even time, i.e., the minimum length of the device's idle period to save energy. The equation neglects the delay and power cost of transition from the idle to the off state. Even so, the transition cost from the off to the active state, namely, $P_{off,on}$, is considered to be a deterministic quantity. A similar model can also be found in [22].

While this can be useful for simple servers, our own observation suggests that the power cost of state transition should better be modelled as stochastic quantity. Moreover, this cost significantly differs from subsystem to subsystems as well. Figure 5 displays the cumulative distribution functions of the power consumption of the motherboard taken from the 12V supply line and the CPU taken from the 12V supply line.

Therefore rich context information pertaining to computing resources enables fine-grained decisions to be made at different abstraction layers.

B. Consolidation

To make a server operate in its optimal region or to switch it off altogether, services should be consolidated by migrating them or by rebinding to alternative services. Service rebinding has the same goal as service migration, but it is carried out when the cost of service migration is high. A rebinding process releases a service that runs on an underutilised or overloaded server and rebinds to a similar service that runs on an optimally configured server.

Both service migration and rebinding require the live transfer of state information from one server to another. Thus, the main tasks in consolidating services are to (1) identify the services whose migration or deactivation produces the optimal energy utilisation; (2) estimate the cost of migration and/or rebinding; and (3) identify target servers on which the services should be migrated or alternative services should be

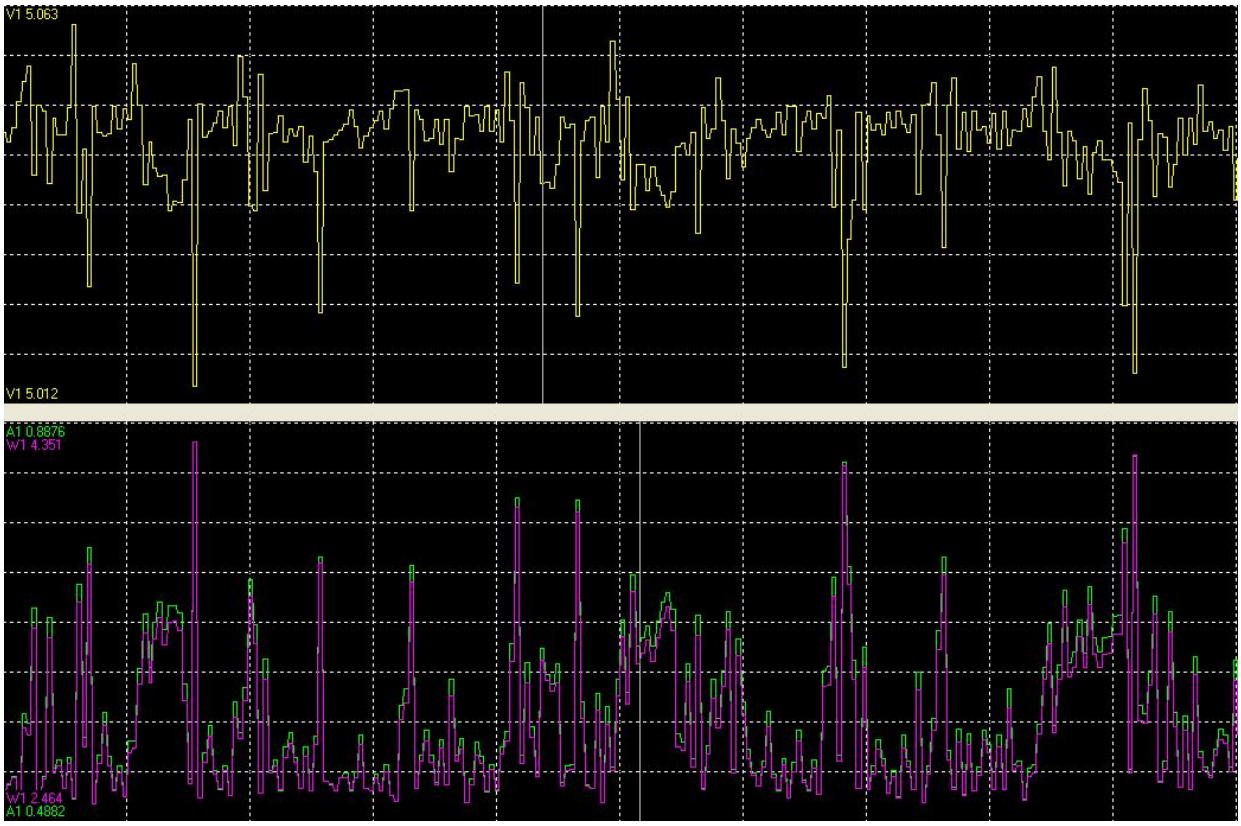

Fig. 4. The fluctuation in the supply voltage of a 5V line of a hard disk during a write operation. Top: Fluctuation in supply voltage. Bottom: Variation in drawn current (power). The Yokogawa 210 Power Analyser was used during the measurement.

started. These tasks become complex as the number of services running on the individual servers become large.

Figure 6 displays the steps required to carry out energy-aware adaptation. An event from a hardware monitor is emitted when a server is underutilised or overloaded. When the adaptation component receives this event, it identifies among the services that are actively running on the server the ones which can be migrated or replaced by equivalent services elsewhere. Then, it estimates the cost of adaptation and compares it with the energy gain that can be achieved through adaptation.

Once the services with minimal adaptation cost and optimal energy gain are chosen, the adaptation component ensures that no SLA will be violated or, if this cannot be avoided, it recomputes an adaptation cost for a different set of services. Finally, it computes the optimal target servers to which the services migrate or on which a service rebinding takes place. The last problem, i.e., finding the optimal target servers, can be modelled as a multi-dimensional, multi-objective, multi-choice knapsack problem (MMMKP) [26]. Given n target servers that have different accommodation capacities[1] (see Figure 7) and m different services each of which generates a different workload, the aim is to compute the optimal service distribution without violating any of the existing Service Level Agreements.

[1]Since each server in a cluster should operate between two energy thresholds (within the optimal operation region), the energy-aware service execution platform migrates services to or rebinds services on a target server as long as its energy consumption remains within the operating region.

IV. The Cost of Adaptation

One of the side-effects of adaptation is the delay it causes in the execution of services. Often a delay in the execution of a single service results in a chain of concomitant delays. The delay cost is higher for a service rebinding than a service migration. This is because, in addition to the cost of identifying the optimal target servers on which the newly selected services should be started, a service rebinding requires the selection of an optimal alternative service that (1) minimise the energy consumption of a service execution and (2) also satisfy all the service level objective constraints defined by the SLAs. For instance, in Figure 2, the subtasks of the service composition at the application layer can be carried out by different services at the service level. The subtask T_A can be performed by S_{A1} that runs on Machine 1. It can also be performed by the alternative services S_{A2} or S_{A3} that run on Machine 2 and Machine 3, respectively. Likewise, the subtask T_B can be performed by S_{B1} or S_{B2}, each of which runs on different machines, and so on. Hence, the complexity of choosing the optimal services increases with the size of the workflow as well as the the number of alternative services for each subtask.

We consider the delay cost of a rebinding process to be the worst case of our adaptation strategy. As a result, we

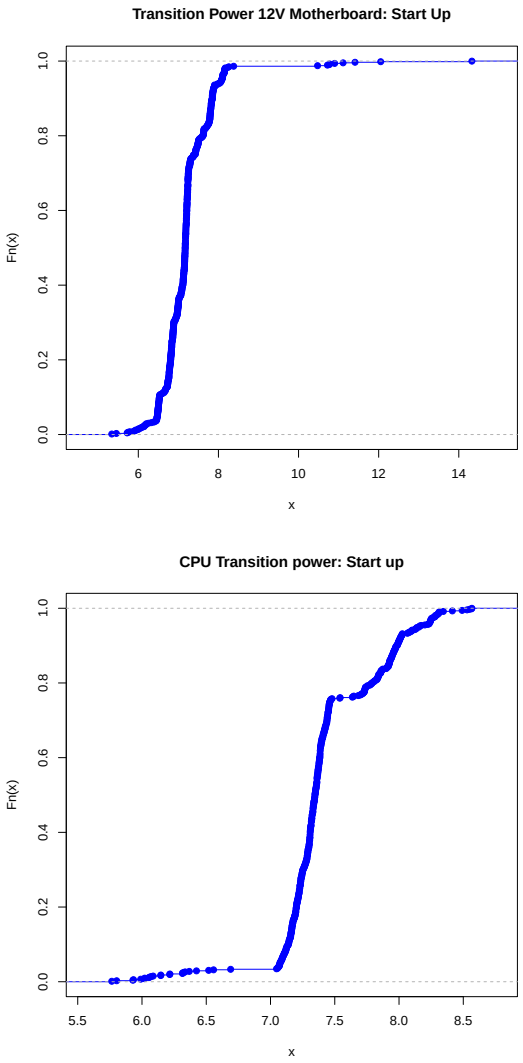

Fig. 5. The cumulative power consumption of the 12 V supply line supplying power to the mother board (top) and the CPU (Bottom). In both cases, the x-axis refers to the power consumption in Watt. The transition cost was measured on the same server running Ubuntu Server version 10.4 and the server was running idle before it was shut down.

implemented a rebinding component within the adaptation component of Figure 3 and tested the end-to-end delay of a service execution (for the entire workflow) during service binding. The rebinding component consists of three subcomponents, namely, the generator, the selector, and the core (see Figure 8). The generator is provided with a workflow description formally defined with BPEL [11] of a workflow and generates Java objects which are useful for the selector to carry out service selection. The selector is provided with the description of alternative services and a workflow and computes the optimal services for binding. This entails the evaluation of $\prod_{i=1}^{m} n_i$ task-to-service binding, where n_i is the number of equivalent services for each subtask i and m is the total number of subtasks. For instance, there are altogether 24 implementations for the composition displayed

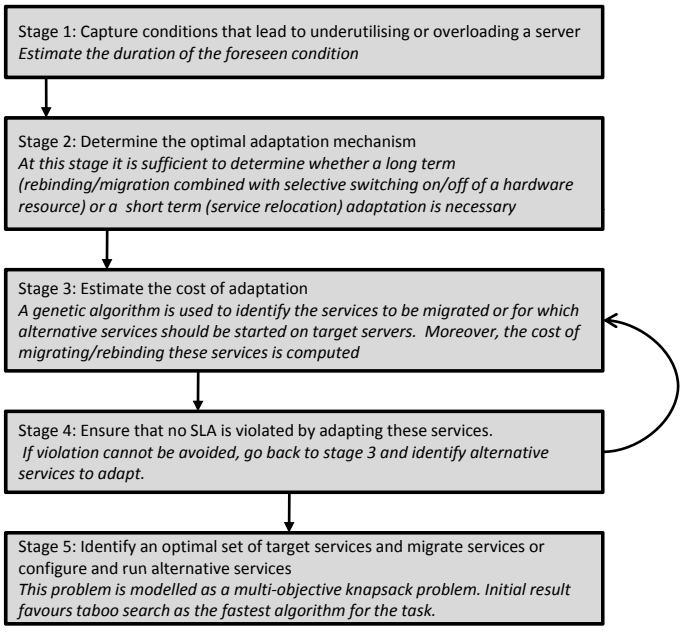

Fig. 6. A brief summary of the steps taken to undertake an adaptation

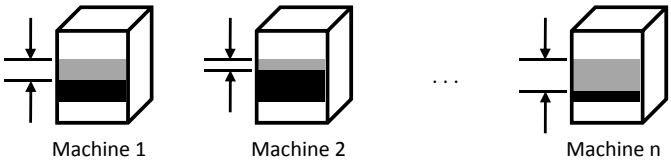

Fig. 7. Identifying target servers modelled as the knapsack problem. The n target servers have different workloads and can accommodate different types of services.

in Figure 2. The core provides the interface between the adaptation components and the client supplying the workflow, the description of the equivalent services for each subtask, and the QoS constraints. The core also synchronises the rebinding process by calling the generator to read the specifications; and the selector to output the optimal services for a composition.

We experimented with three comparable heuristics, namely, the genetic algorithm [21], the taboo search [8], and the

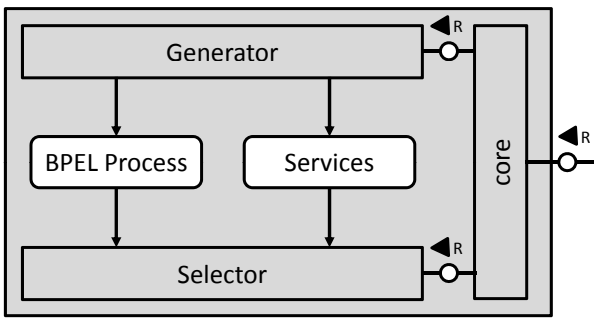

Fig. 8. The rebinding components

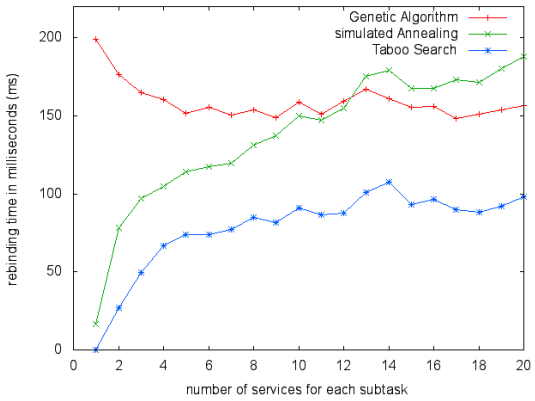

Fig. 9. Rebinding time in milliseconds using genetic algorithm, simulated annealing, and taboo search as a function of the number of alternative services for each subtask in a composition.

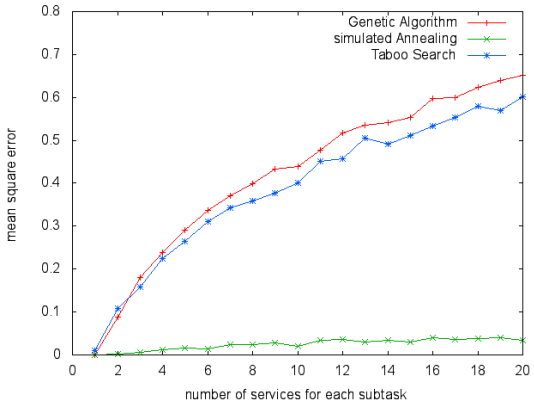

Fig. 10. The mean square error of genetic algorithm, simulated annealing and taboo search for finding the optimal services that maximize a composition's end-to-end QoS.

simulated annealing [13] as selection strategies. The efficiency of the optimisation heuristics was evaluated in terms of their performance and correctness. The performance test focuses on the time it takes for the selector to compute the optimal set of services by satisfying a prescribed set of non-functional requirements of a workflow (availability, reliability, response time, and price). To evaluate the correctness of the heuristics, we first computed the optimal services using dynamic programming, which yields the true optimal result, but its computation time was unacceptably high. Then we compared the output of the heuristics with the results obtained from the dynamic programming.

In the experiment, we measured a computation's time in which 20 subtasks were defined, each subtask having up to 7 alternative services. This results in $[20,\ 20^7]$ implementations. Figure 9 displays the computation time as a function of the number of alternative services for each subtask. The computation time of all algorithms increases with the number of services. The algorithm with the lowest selection time is the taboo search, followed by the simulated annealing. The maximum execution time of each algorithm was less than 200 ms.

To evaluate the correctness of the heuristics, we once again varied the alternative services of each subtask from 1 to 7 implementations. Each service had a randomly generated response time. Each test case was performed 50 times. The correctness is expressed by calculating the mean square error between the response time obtained from the dynamic programming and the minimum response time obtained by employing the heuristics. The mean square error as a function of the number of services per task is shown in Figure 10. The simulated annealing is the most correct algorithm in finding the minimal response time, with a mean square error that was less than 0.05.

V. Conclusion

This paper proposes energy-aware service execution which carries out adaptation at various abstraction levels. The aim is to avoid situations that lead to underutilising or overloading physical machines. At the application level, a service execution is differed as a short term strategy. At the service level, services are either migrated to alternative physical servers or applications bind to alternative services so that services that underutilise or overload a server can be stopped. Unlike previous approaches which focus on the migration of virtual machines, service migration is fine-grained, highly flexible, and fast. At the hardware level, physical machines are selectively switched on or of to minimise idle state energy consumptions. Adaptation begins by first foreseeing underutilising or overloading conditions and by estimating the duration of these conditions. The service level objectives defined in Service Level Agreements for each physical service and the logical dependencies between services in a workflow set constraints on adaptive decisions.

The approach assumes that the workload that can be generated by the execution of each service can be estimated. Chen et al. [4] have demonstrated that this assumption is feasible. The adaptation strategies entail (1) the computation of the optimal adaptation cost, and (2) the determination of the optimal target servers to which services should migrate or on which new set of alternative services should be started. Since the target servers may have different accommodation capacities and each service to be migrated or decommissioned may be required to satisfy different set of service level objectives, we modelled the adaptation at the service level as a multi-dimensional, multi-objective, multi-choice knapsack problem (MMMKP). We evaluated three comparable heuristics for this task, namely, taboo search, genetic algorithm, and simulated annealing. The taboo search was the fastest, while the simulated annealing was the most correct one in terms of finding a global optimal solution.

In the future, we aim to transform a workflow into a probabilistic finite state machine to estimate the energy consumption of an application. The model will be useful for estimating the workload that can be generated by a service execution and for allocating the optimal amount of computing resources.

ACKNOWLEDGEMENT

The authors would like to acknowledge the German Research Foundation (DFG) for partially funding the Project under agreement: SFB 912/1 2011

REFERENCES

[1] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40:33–37, December 2007.

[2] W. Binder and N. Suri. Green computing: Energy consumption optimized service hosting. In *SOFSEM '09: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, pages 117–128, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] K. Brill. The invisible crisis in the data center: The economic meltdown of moore's law. white paper, Uptime Institute, 2007.

[4] Y. Chen, A. S. Ganapathi, A. Fox, R. H. Katz, and D. A. Patterson. Statistical workloads for energy efficient mapreduce. Technical report, EECS Department, University of California, Berkeley, 2010.

[5] Y. Chen, L. Keyes, and R. H. Katz. Towards energy efficient mapreduce. Technical report, EECS Department, University of California, Berkeley, 2009.

[6] C.-F. Chiasserini and R. R. Rao. Improving energy saving in wireless systems by using dynamic power management. *IEEE Trans. wireless comm.*, 2:1090–1100, April 2003.

[7] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[8] D. Cvijovic and J. Klinowski. Taboo search: An approach to the multiple minima problem. *Science*, 267(5198):664–666.

[9] W. Dargie. The role of probabilistic schemes in multisensor context-awareness. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, PERCOMW '07, pages 27–32, Washington, DC, USA, 2007. IEEE Computer Society.

[10] W. Dargie and T. Tersch. Recognition of complex settings by aggregating atomic scenes. *IEEE Intelligent Systems*, 23:58–65, September 2008.

[11] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[12] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 13–23, New York, NY, USA, 2007. ACM.

[13] D. Fogel. An introduction to simulated evolutionary optimisation. *IEEE Trans. neural networks*, 5:3–15, January 1994.

[14] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. *ArXiv e-prints*, 901, December 2009.

[15] J. Hamilton. Cooperative expendable micro-slice servers (cems): Low cost, low power servers for internet-scale services. In *4th Biennial conference on innovative data systems research (CIDR)*, 2009.

[16] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Trans. Comput.*, 56:444–458, April 2007.

[17] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.

[18] G. Jourjon, T. Rakotoarivelo, and M. Ott. Models for an energy-efficient p2p delivery service. In *PDP '10: Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 348–355, Washington, DC, USA, 2010. IEEE Computer Society.

[19] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen. Greencloud: a new architecture for green data center. In *ICAC-INDST '09: Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pages 29–38, New York, NY, USA, 2009. ACM.

[20] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. *SIGOPS Oper. Syst. Rev.*, 40:403–414, April 2006.

[21] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[22] A. Sinha and A. Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Des. Test*, 18:62–74, March 2001.

[23] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008. USENIX Association.

[24] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. A scalable application placement controller for enterprise data centers. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 331–340, New York, NY, USA, 2007. ACM.

[25] A. Weissel, M. Faerber, and F. Bellosa. Application characterization for wireless network power management. In *Proceedings of the International Conference on Architecture of Computing Systems (ARCS'04)*, 2004.

[26] C. Ykman-Couvreur, V. Nollet, F. Catthoor, and H. Corporaal. Fast multi-dimension multi-choice knapsack heuristic for mp-soc run-time management. In *Proceedings of the 2006 international symposium on system-on-chip*, pages 1–4, 2006.