Mutual Influence of Application- and Platform-Level Adaptations on Energy-Efficient Computing

Kateryna Rybina, Waltenegus Dargie, René Schöne, Somayeh Malakuti

Faculty of Computer Science, Technical University of Dresden, 01062 Dresden, Germany
Email:{kateryna.rybina, waltenegus.dargie, rene.schoene, somayeh.malakuti}@tu-dresden.de

***Abstract*—We experimentally investigate the mutual influence of application- and platform-level adaptations in a virtualized cluster environment. At the application level, applications can adapt to a changing execution environment by dynamically exchanging components that enable them to trade energy for utility and vice versa. Likewise, at the platform level, virtual machine monitors can migrate virtual machines from one server to another either to consolidate workloads and switch-off underutilized servers or to distribute the workload of overloaded servers. Our experiment quantify impacts of various types of adaptations on QoS, power consumption, and energy-overhead.**

***Keywords*—Adaptation, cloud computing, energy-efficient computing, virtualization, virtual machines migration, migration costs**

I. Introduction

Internet servers and data centers consume a large amount of energy even though most of them are underutilized much of the time. For example, in a typical Twitter server, CPU utilization is less than 20% and RAM utilization is between 40% and 50% [1]. Likewise, in a typical Google server, CPU utilization is between 25% and 35% and RAM utilization is approximately 40% [2]. In Amazon's EC2 cloud environment, the CPU utilization per server is between 3% and 17% [3]. It must be remarked that the idle power consumption of a server is between 50% and 60% of its peak power consumption [4], [5].

Different dynamic power management strategies can be implemented at various levels of abstractions to optimize energy consumption. At the operating system level, an energy-aware scheduler can perform dynamic task scheduling or dynamic voltage and frequency scaling [4], [6], [7]. Decisions made by the operating system usually have a scope of seconds and milliseconds or even less. In virtualized environments, idle or underutilized servers can entirely be turned off by first migrating the virtual machines to servers which can be optimally loaded. The same approach can be employed to achieve load balancing. This type of adaptation, known in the literature as server or workload consolidation [8], has a scope of several minutes or hours. Similarly, at the application level, application-specific decisions can be made to utilize resources efficiently, such as prioritizing tasks and user requests as well as trading quality of service for energy and vice versa.

An interesting research challenge is how to benefit from the gains that can be obtained from all levels of abstraction. The challenge primarily stems from the fact that the hardware resources (the servers), the distributed execution platform (the virtual machines), and the applications may belong to different owners and each may have a different view of the whole system. For example, the application provider has a sufficient knowledge of the workload that can be generated by its users but does not have sufficient knowledge pertaining to the resource distribution at the hardware level. Similarly, at the platform level, the virtual machine monitor has sufficient knowledge of the resource utilization statistics of the whole server but may only indirectly infer – from resource utilization – the workload statistics at the application level. It may not know, however, the various adaptation strategies the application provider may have at its disposal. Therefore, it is desirable to investigate the extent to which energy-aware adaptations at different levels of abstraction influence each other and whether they can be coordinated in a meaningful way, so that optimal energy saving can be attained and foreseeable side-effects can be mitigated. In this paper, we experimentally investigate the mutual influence of energy-aware adaptations at the application and platform levels and their contribution to minimize the overall energy consumption of a server. At the application level, we consider the dynamic reconfiguration of application components by switching to different variants or implementations to deal with changes in resource availability or quality of service. At the platform level, we migrate virtual machines to manage underutilized and overloaded servers.

The rest of this paper is organized as follows. In Section II, we provide background information concerning application and platform-level adaptations and their mutual influence. In Section III, we describe in detail our experiment setting and the experiments we carried out. In Section IV, we analyze the results of the experiment and interpret them. Finally, in Section V, we provide concluding remarks and outline our future work.

II. Background

One of the mechanisms widely used to make applications energy-aware is to provide them with multiple alternative components from which they can dynamically choose and reconfigure themselves [9]. The components should enable the application to trade energy for utility and vice versa. For example, the different components can have different resource requirements and, as a result, different quality of service (QoS). Götz et al. [10] propose the notion of energy contracts with which they specify the relationship between the resource consumption of an application component and the QoS that can be achieved with it. This contract along with linear-integer programming is used to optimally configure an

application. Similarly, Cohen et al. [11] propose the notion of "energy types" (special forms of data types in programming languages) as means to represent multiple variants of applications by labeling data and operations with resource consumption information and execution phases of applications, which have distinct patterns of energy consumption. Malakuti and Wilke [12] propose "energy aspects" to implement energy-aware features into legacy applications in a modular fashion. Sampson et al. [13] provide a mechanism for programmers to annotate whether an operation requires an approximate or a precise computation. For precise computation ample resources are required in contrast to approximate computation.

In a virtualized environment, application-level adaptations can be complemented by VM-level adaptations (for VM-level adaptation, refer, for example, to [14], [15], [16]). Whereas the benefits of application- and VM-level adaptations have been investigated separately, we investigate them jointly. For our prototype we used video transcoding in a video hosting application (such as YouTube) and different versions of off-the-shelf transcoders – `FFMPEG`, `MENCODER`, and `HANDBRAKE`. The transcoders have different resource consumption characteristics as well as transcoding speed. Users upload videos of different formats and sizes to the hosting application, so that they can be viewed by other users. However, viewers may request the videos in different formats and, therefore, it is the task of the application to transform the videos from their original formats to different popular formats and store them.

Often video transcoding takes place as a background batch job, giving the application as well as the platform ample opportunity for resource- and energy-aware adaptations. During video transcoding, the application can trade performance (transcoding time) for energy/resource consumption (or vice versa) without compromising the quality of the videos by choosing different types of transcoders. Fast transcoders usually consume a large amount of resources as well as energy while slow transcoders have longer transcoding time but consume less resources. When the physical server is underutilized, it is reasonable to choose fast transcoders, so that as many videos as possible can be transcoded in a short time, but when the server is overloaded, slow transcoders can be chosen. However, despite these choices, a server can still remain underutilized or overloaded and the platform which hosts and manages the virtual machines can decide to migrate them. In doing so, the performance of some of the applications being migrated can be affected. It is this mutual influence we would like to investigate.

III. Experiment Setting

For our experiment, we employed two "homogeneous" servers and a network attached storage connected with each other via a Gigabit Ethernet switch. The two servers run Fedora 15 (Linux kernel v. 2.6.38, x86_64) as operating system, KVM as hypervisor and `libvirt` for managing platform virtualization and virtual machines. Each server employs two Intel i5-680 dual core 3.6 GHz processors, 4 GB DDR3-1333 SDRAM memory and a Gigabit Ethernet Network Interface Card (NIC). The NAS system consists of Intel Xeon E5620 Quad-Core 2.4 GHz processor, 10 GB DDR3-1333 SDRAM memory and Gigabit Ethernet NIC. To measure the overall power consumption of both servers, we used Yokogawa WT210 digital power analyzers. We created four virtual machines. Three of these run the `lookbusy` benchmark, which generates a synthetic, fixed and predictable amount of workload on CPUs and keep chosen amounts of memory active (this way, we could underutilize or overload servers). The fourth virtual machine runs one of the transcoders. Each virtual machine is allocated a single virtual core but the RAM allocation varies: For the three virtual machines running the benchmarks we reserved 400 MB, 600 MB, and 800 MB RAM, whereas for the transcoder application we reserved 1200 MB. We reserved a 100 MBps network bandwidth for the live migration of virtual machines. For all the experiments, the transcoders converted a 3.4 MB video file from VP6 to MPEG4 format. The target video has 640×320 pixels resolution. Requests to convert video files are generated at a fixed interval of 21 seconds. To obtain CPU and memory usage statistics, we ran `dstat` on both servers and the virtual machines.

A. Experiment Scenarios

We aim to measure the mutual influence of the application- and platform-level adaptations by quantifying the energy-utility trade-offs during adaptation. A utility refers to the quality of service that can be obtained for a particular system configuration or energy budget. In our experiment, we considered two extreme scenarios which are the causes of adaptations, namely, system underutilization and overload. We use empirical thresholds to specify these scenarios: If the average CPU utilization of a server is above 90%, we describe this as an overload condition because the system becomes noticeably slow. On the other hand, if the supply of CPU cycles in both servers is greater than the CPU cycles being utilized by all virtual machines, then we consider the servers as underutilized. These definitions are valid under the assumption that there is no resource bottleneck elsewhere. If, on the other hand, the the average CPU utilization of one of the servers is less than ($100\% -$ the average CPU utilization of the other server), we consider the first server as underutilized because its workload can be transferred to the second and it can be switched off. Alternatively, the underutilized server can replace a slow transcoder by a fast transcoder and increase its CPU utilization. We considered both application-level adaptation (binding to different transcoders) and VM-level adaptation (live migration of virtual machines) to deal with underutilized or overloaded conditions and evaluated the effects of these adaptation strategies on the power and energy consumptions of the servers as well as the video transcoding time.

In the underutilized scenario VM 4 (running one of the transcoders) executed in the first server while the remaining three virtual machines executed in the second. In the overload scenario, all four virtual machines executed on the second server while the first server was turned off. The workloads of the virtual machines vary over time in a controlled fashion to create a transition from an underutilized condition to an overloaded condition and vice versa.

IV. Evaluation

A. Underutilized Scenario

When the demand for computing resources is less than the supply of resources, either the active virtual machines

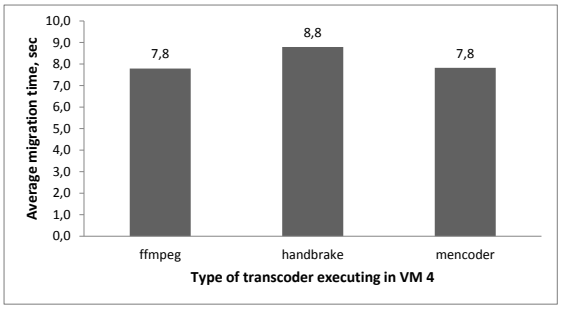

Fig. 1: The average time required to migrate virtual machines encapsulating the three different transcoders. In all the cases, the network bandwidth during migration was 100 MBps.

can request more resources in order to increase their speed of execution or resources can be switched-off to reduce their idle power consumption. The best way to reduce idle power consumption is to switch-off an entire server. To do so, all the virtual machines it hosts should first be migrated without stopping the applications from executing. The price of workload consolidation has four dimensions: the virtual machine migration latency (duration), a potential deterioration in the quality of the service being provided, the power consumption of both the source and target servers due to the need to coordinate migration, and the energy-overhead of virtual machine migration. In the underutilized scenario, we migrated VM 4 from Server 1 to Server 2 and shut down Server 1.

1) Migration Time: The migration time is defined as the time interval between the point at which the migration starts at the source server and the point at which the virtual machine begins execution at the target server. Live migration has two phases: In the first phase, the RAM content and the state of the virtual machine are iteratively copied to the target server while the VM executes in the source server. In the second phase, the VM is stopped briefly to prevent further memory updates, the latest updates are copied to the target server, and the VM is started in the target server [17], [18]. These phases require computation and communication resources and introduce VM down-time (phase two). As a result, the application's quality of service may degrade. Figure 1 shows the time required to migrate VM 4 from Server 1 to Server 2 when the VM was executing the three transcoders and when both servers were underutilized. The migration time is higher for HANDBRAKE, because it uses the largest amount of memory.

2) Impact of Migration on QoS: Figure 2 shows the impacts of VM migration on transcoding time. The transcoding time increased slightly for all the transcoders during migration. For example, it took 17.4 seconds on average for FFMPEG to transcode a single video when there was no migration but with migration it took on average 20.3 seconds. The migration time of VM 4 when FFMPEG was executing was 7.8 seconds on average (see Figure 1). This means, that the transcoder executed approximately 12.5 seconds in Server 2 and it had to share resources with the three virtual machines (overall CPU utilization of Server 2 exceeded 60% after the migration of VM 4). Consequently, transcoding time increased.

3) Power Consumption: Figure 3 shows the power consumption of both Server 1 and Server 2 in the underutilization

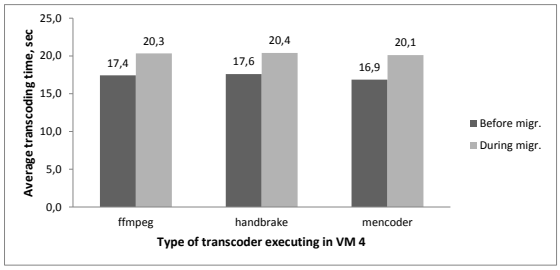

Fig. 2: The average transcoding time of the three transcoders during migration when one of the servers was underutilized.

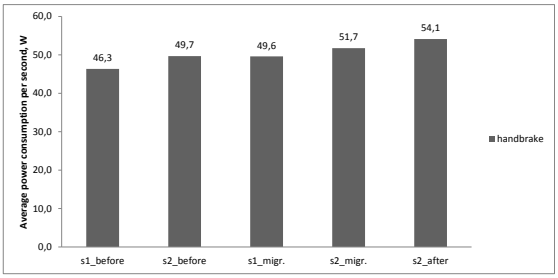

Fig. 3: The average power consumption of both servers for the underutilized scenario when HANDBRAKE was running.

scenario in five different experimental settings. HANDBRAKE was used as transcoder inside VM 4. The power consumption of both servers has been low before migration but during migration it increased. Since Server 1 was switched-off after migration, the power consumption of Server 2 is of interest after migration. As expected, the power consumption of Server 2 increased to accommodate VM 4 but by approximately 4 W only, clearly showing the benefit of server consolidation whenever this is possible, because running both Servers requires 45 W more than running only Server 2, as shown in Figure 3.

4) Energy Overhead: The energy overhead of migration can be expressed as the extra energy consumed by both server due to migration. Mathematically, it can be expressed as:

$$E_{overhead} = \left(P_{s1-dmig} - P_{s1-bmig}\right) \times t_{mig} + \left(P_{s2-dmig} - P_{s2-bmig}\right) \times t_{mig} \quad (1)$$

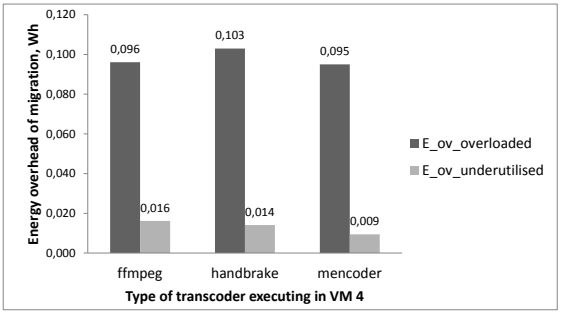

Fig. 4: The energy overhead of migration.

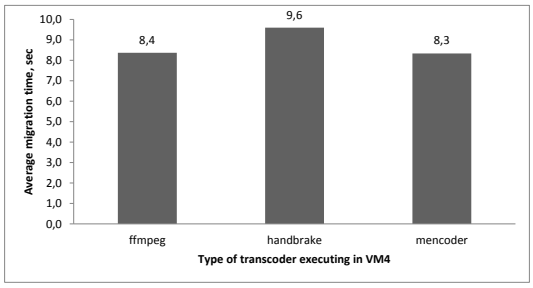

Fig. 5: The average migration time of VM 4 when the three transcoders executed in the overloaded scenario.

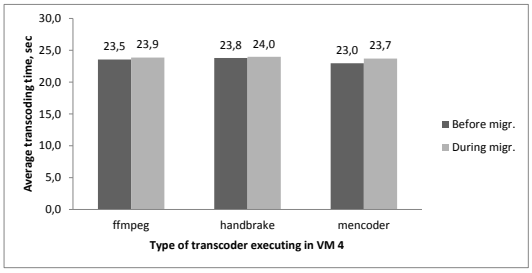

Fig. 6: The average transcoding time during VM migration when Server 2 was overloaded.

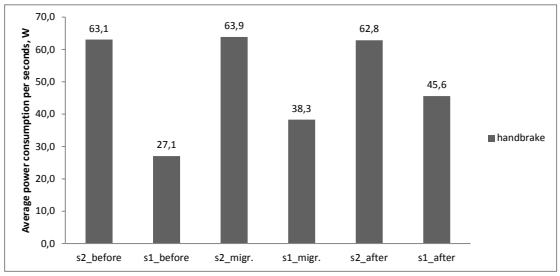

Fig. 7: The average power consumption of the two servers. Migration took place because Server 2 was overloaded.

where t_{mig} refers to the migration time; $P_{s1-dmig}$ and $P_{s2-dmig}$ refer to the average power consumption of Server 1 and Server 2, respectively, during migration. $P_{s1-bmig}$ and $P_{s2-bmig}$ refer to the average power consumption of Server 1 and Server 2, respectively, before migration. The energy overhead is significantly low when the two servers have sufficient resources, regardless of which of the transcoders was used, as can be seen in Figure 4. In summary, the energy overhead of migration plus the extra time introduced in the transcoding time make up the cost of service consolidation in our example scenario.

B. Overloading Scenario

The different transcoders generate different amount of workload on the CPU and have different RAM utilization profiles. For example, `FFMPEG` and `MENCODER` induce a relatively high CPU workload (92.5%) but have comparatively low memory footprint (279 MB). On the contrary, `HANDBRAKE` generates relatively little workload on the CPU (91%) while requiring more memory (348 MB). These differences in resource utilization can be exploited to carry out application-specific adaptations when the server becomes overloaded. However, despite these possibilities, VM-level adaptation may still be required because the application-level adaptation may not produce appreciable change in the system configuration. A VM-level adaptation during an overloading situation takes place by starting an additional server and offloading some of the virtual machines from the overloaded server to the new server. For our case, we migrated VM 4 to the new server and quantified the cost of migration.

1) Migration Time: Figure 5 shows the migration time for the three transcoders. The migration time increased for all the transcoders, which is plausible due to the scarcity of resources when compared with the underutilized situation.

2) Impact of Migration on QoS: Figure 6 shows the impact of VM migration on the transcoding time, which barely increased. This phenomena demonstrates an interesting aspect of load-balancing: *during* migration, transcoding certainly slowed down adding a few extra seconds on the transcoding time, but this cost was compensated by the rich availability of resources at the target server which facilitated the transcoding process for the remaining time. The overall result is a near zero impact of migration on the QoS. For example, for the FFMPEG transcoder, the average transcoding time was 23.5 seconds when the server was overloaded and no migration was performed. The average migration time for this transcoder was 8.4 seconds. Hence, if the transcoder began a new task just at the beginning of migration, then it would execute on average 15.5 seconds on the new server to complete its task. Hence, the delay introduced during the 8.4 seconds of migration would be compensated during the remaining 15.5 seconds execution with ample resources.

3) Power Consumption: The cost of load-balancing can be clearly seen by considering the power consumption of the two servers. If we take the case of `HANDBRAKE` as an example (see Figure 7), when all the virtual machines executed on Server 2, its overall power consumption was on average 63.1 W. The idle power consumption of Server 1 (when there was no virtual machine running on it) was on average 27.1 W. During the migration of VM 4 from Server 2 to Server 1, the overall power consumption of both servers increased and that of Server 1 rather significantly. After migration, the power consumption of Server 2 reduced only slightly while the power consumption of Server 1 increased even further, because the transoder was executing in it and in this case the transcoder had plenty of resources and transcoded at a higher speed. This case demonstrates the trade-off between power consumption and utility. Load-balancing reduced the transcoding time of each video. However, the power consumption of both servers increased significantly.

4) Energy Overhead: Compared with the underutilized scenario, the energy overhead of migration becomes significantly high for the case of the overloaded scenario (Figure 4). The increment in the average power consumption in both servers during migration as well as the longer migration time contributed to the large amount of energy overhead. The power consumption of both servers changed on average by 12 W during migration when VM 4 was migrated from the

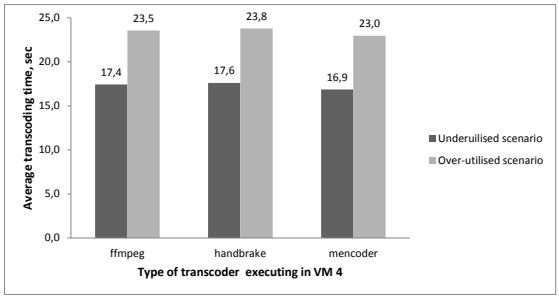

Fig. 8: Average transcoding time when one of the servers was either underutilized or overloaded.

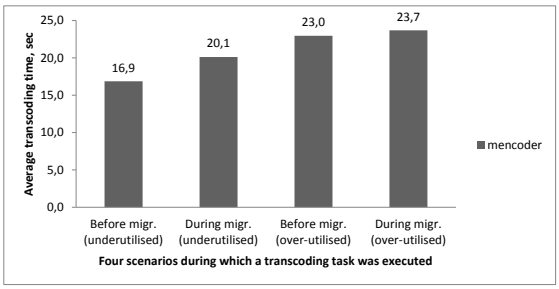

Fig. 9: MENCODER's average transcoding time.

overloaded server whereas it changed on average by 5.3 W when the VM was migrated from the underutilized server.

C. The Impact of Adaptations on QoS

Figure 8 summarizes the average transcoding time of the three transcoders. When VM 4 was executing in an underutilized server (Server 1), the average transcoding time was 16.9, 17.4, and 17.6 seconds for MENCODER, FFMPEG, and HANDBRAKE, respectively. When the server is overloaded, the transcoding time increased by approximately 6 seconds for all the transcoders, as can be seen in Figure 8. Figure 9 summarizes the different transcoding times of MENCODER when a server undergoes different execution states. As can be expected, the shortest transcoding time is achieved when the server was underutilized. VM migration in both underutilized and overloaded states increased transcoding time. However, the increment during migration should be understood in context. In case of a transition from an overloaded condition to a balanced-load condition, the extra seconds introduced during migration can be compensated by the subsequent state which provides the transcoder with sufficient resources to execute faster.

V. Conclusion and Future Work

In this paper, we experimentally evaluated the contributions of application-level and VM-level adaptations and how they trade energy consumption for utility and vice versa. The application-level adaptations were achieved by dynamically selecting different implementations of application components that have different resource requirements but also yield different utilities. For video transcoding applications, which were the subjects of our experiment, we considered binding to different transcoder implementations. We performed VM-level adaptation to consolidate servers (so that those underutilized can be switched-off) and to distribute the workload of overloaded servers. In both cases, we quantified the cost of virtual machine migration in terms of migration time, the impact of migration on the quality of the service provided by the applications (transcoding time), and energy overhead. In general, the migration cost, with the exception of the transcoding time, was significantly high when an overloaded server was involved. In our present investigation, the application- and VM-level adaptations were not coordinated. We are working on the architecture to enable this coordination, and then we will experimentally investigate the impact of this coordination on the overall energy-efficiency of an entire cluster. Furthermore, we are planning to employ formal models to detect potential conflicts and interferences between the different adaptation techniques and to avoid these problems.

References

[1] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and qos-aware cluster management," in *ASPLOS '14*. ACM, 2014, pp. 127–144.

[2] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *SoCC '12*. ACM, 2012, pp. 7:1–7:13.

[3] H. Liu, "Host server cpu utilization in amazon ec2 cloud," 2012.

[4] W. Dargie, "Analysis of the power consumption of a multimedia server under different DVFS policies," in *CLOUD '12*, June 2012, pp. 779–785.

[5] C. Mobius, W. Dargie, and A. Schill, "Power consumption estimation models for processors, virtual machines, and servers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 6, pp. 1600–1614, June 2014.

[6] A. Brihi and W. Dargie, "Dynamic voltage and frequency scaling in multimedia servers," in *AINA '13*, March 2013, pp. 374–380.

[7] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *CLOUD '11*. ACM, 2011, p. 5.

[8] Q. Zhu, J. Zhu, and G. Agrawal, "Power-aware consolidation of scientific workflows in virtualized environments," in *SC '10*. IEEE Computer Society, 2010, pp. 1–12.

[9] W. Dargie, A. Strunk, and A. Schill, "Energy-aware service execution," in *LCN '11*. IEEE, 2011, pp. 1064–1071.

[10] S. Götz, C. Wilke, S. Richly, G. Püschel, and U. Assmann, "Model-driven self-optimization using integer linear programming and pseudo-boolean optimization," in *ADAPTIVE '13*, 2013.

[11] M. Cohen, H. S. Zhu, E. E. Senem, and Y. D. Liu, "Energy types," in *OOPSLA '12*, 2012, pp. 831–850.

[12] S. Malakuti and C. Wilke, "Energy aspects: Modularizing energy-aware applications," in *GREENS '14*, 2014, pp. 23–30.

[13] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," *SIGPLAN Not.*, vol. 46, no. 6, pp. 164–174, Jun. 2011.

[14] A. Strunk and W. Dargie, "Does live migration of virtual machines cost energy?" in *AINA '13*. IEEE, 2013, pp. 514–521.

[15] K. Rybina, W. Dargie, A. Strunk, and A. Schill, "Investigation into the energy cost of live migration of virtual machines," in *SustainIT '13*. IEEE, 2013, pp. 1–8.

[16] F. Xu, F. Liu, H. Jin, and A. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11–31, Jan 2014.

[17] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI '05*. USENIX Association, 2005, pp. 273–286.

[18] F. Ma, F. Liu, and Z. Liu, "Live virtual machine migration based on improved pre-copy approach," in *ICSESS '10*, July 2010, pp. 230–233.