HAECubie: A Highly Adaptive and Energy-Efficient Computing Demonstrator

Franz Eichhorn, Waltenegus Dargie, Christoph Möbius, and Kateryna Rybina

Chair of Computer Networks, Faculty of Computer Science,
Technical University of Dresden, 01062 Dresden, Germany
Email: firstname.lastname@tu-dresden.de

***Abstract*—The amount of data that are computed, stored, and shared over the Internet is rising at an unprecedented scale. This has necessitated more servers to be deployed and drastic improvement in the capacity of individual servers. However, several independent studies also reveal that resources are not optimally utilised in most existing data centres and server clusters. Since the introduction of server virtualization and cloud computing, the research community has proposed several workload aggregation and dynamic consolidation techniques, however, most of these techniques are either theoretical and rely on simulation environments or use real servers but static workloads or benchmarks. In reality, the workload of data centres fluctuates as a function of time and servers frequently experience both overloading and underutilised conditions. In this paper we introduce the HAECubie demonstrator we developed and deployed to experimentally evaluate the scope and usefulness of dynamic workload consolidation in a server cluster and to quantitatively analyse the relationship between energy/power consumption and the utility (performance) that can be achieved through workload consolidation. Our demonstrator is a video hosting platform and enables Internet users to stream videos of variable length. The number of users accessing the HAECubie as well as the duration of videos they stream are modelled as stochastic processes based on realistic estimation of the workload of existing video hosting platforms.**

***Index Terms*—Adaptation, demonstrator, energy-efficient computing, energy-efficient server, energy-utility function, workload consolidation, video hosting platform, video streaming**

I. Introduction

The amount of data hosted and processed by Internet-based servers and data centers is increasing at a remarkable speed. This trend will stay in future mainly for two reasons. Firstly, as more users replace their desktop environments with smart phones and tablet computers, the ease with which they capture and share their experience on the spur of the moment increases significantly. Secondly, the introduction of private and public cloud computing will accelerate the transfer of a large amount of data from terminal devices to back-end servers. According to the recent Cisco Global Cloud Index, global data center IP traffic will reach 554 exabytes per month by 2016 (in comparison, it was 146 exabytes per month in 2011). Similarly, the global cloud IP traffic will increase from 57 exabyte per month (in 2011) to 355 exabytes per month by 2016. Likewise, the magnitude of workload per installed cloud server will increase by more than twofold by 2016 compared to the workload per installed server in 2011 [1].

This has resulted in a growing demand for more server deployment as well as for more computing power per individual server. The estimated worldwide server deployment in 2010 has been 40 million units [2], but additional servers have been steadily deployed since then. The latest figure from the International Data Corporation (IDC)[1] reveals that more than 8 million server units have been shipped in 2014. As far as capacity per individual server is concerned, systems with operation capacity in the range of exaflops and an optical link number of around 10^8 are expected by the year 2020 [3].

Unfortunately, independent studies indicate that computing resources are not efficiently utilised in many existing data centres and server clusters. For example, in a typical Twitter server, the CPU utilization is less than 20% and the RAM utilization is between 40 and 50% [4]. Likewise, in a typical Google server, the CPU utilization is between 25 and 35% whereas the RAM utilization is approximately 40% [5]. In Amazon's EC2 cloud environment, the CPU utilization per server is between 3 and 17% [6]. At the same time, the idle power consumption of existing servers is between 50 and 60% of their peak power consumption [7], [8].

The introduction of server virtualisation has enabled a more efficient resource utilisation because it is possible to aggregate (consolidate) several virtual machines in a single physical machine. As a result, a substantial body of work exists on live virtual machine migration and runtime virtual machine consolidation. The main aim of these approaches is to balance the demand for and the supply of resources in a data centre or a server cluster and to switch off idle or underutilised servers. However, most of the proposed approaches are either theoretical and rely on simulation environments or use real servers but static workloads or benchmarks. In reality, the workload of data centres fluctuates as a function of time and servers frequently experience both overloading and underutilised conditions. In this paper we introduce the HAECubie demonstrator we developed to experimentally evaluate the scope and usefulness of dynamic workload consolidation in a server cluster and to quantify the energy-utility trade-off it introduces. Our demonstrator is a video hosting platform which enables Internet users to stream videos of variable

[1] http://www.idc.com/getdoc.jsp?containerId=prUS24476413 (last visited December 22, 2014).

length. The number of users who access the HAECubie as well as the duration of videos they stream are modelled as stochastic processes based on realistic estimation of the workload of existing video hosting platforms.

The rest of this paper is organized as follows: In Section II, we review related work. In Section III, we present the hardware and software architecture of our demonstrator and describe building blocks. In Section IV, we give a detail account of the video hosting and consolidation as well as our workload generation strategies. In Section V, we quantitatively evaluate the energy-utility trade-off of the different consolidation strategies. Finally, in Section VI, we provide concluding remarks and outline future work.

II. Related Work

Beloglazov et al. [9] proposed several heuristics for dynamic VM consolidation, which they divide into three subtasks: (1) The detection of overloaded and underutilised servers, (2) the selection of candidate virtual machines for migration, and (3) the identification of target servers to which virtual machines can be migrated. Moreover, they considered fixed and adaptive thresholds to detect underutilised and overloaded situations. The proposed heuristics are based on the statistical analysis of historical data pertaining to the CPU utilisation of real workload traces obtained from PlanetLab [10]. The first heuristic defines the median of the absolute deviations from the median of the data set to adaptively determine thresholds. The second heuristic sets the upper CPU utilisation threshold depending on the difference between the third and the first quartiles of the CPU utilisation. The third heuristic is a local regression method and estimates the $k + 1$ CPU utilisation of a server from the last k observations of the CPU utilisation. In addition, the authors propose three algorithms to identify candidate virtual machines for migration. In the first algorithm, the candidate virtual machines are those which have the shortest migration time. In the second, the virtual machines which have the highest correlation with the CPU utilisation of the source virtual machines are identified. The third algorithm chooses virtual machines randomly.

The virtual machine placement itself is defined as a bin packing problem, where the virtual machines represent the items to be allocated, the bin sizes are the servers' CPU capacities, and the prices are the power consumption of the servers. It is solved by first sorting virtual machines in decreasing order of their CPU utilisations and then by allocating to a destination server virtual machines which cost the lowest price (the lowest increment of power consumption due to the VM allocation). Once overloaded servers are dealt with, the strategy addresses underutilised servers. Virtual machines from these servers are migrated to other servers as long as this can be done without overloading the latter. The performances of the proposed heuristics were evaluated using the CloudSim simulation framework[2]. The simulated IaaS cloud environment consisted of 800 heterogeneous servers, of which half of them were HP ProLiant ML110 G4 servers with dual core Intel Xeon processors 3040 (1860 MHz) and the other half were HP ProLiant ML110 G5 servers with dual core Intel Xeon processors 3075 (2660 MHz). Both types of servers were assigned 4 GB RAM and 1 GBps network bandwidth. They consider more than 1000 virtual machines which resembled Amazon's EC2 virtual machine instance types[3] with the only difference that each virtual machine was assigned a single core and an appropriately scaled amount of RAM. The authors reported that the combination of the regression and the minimum migration time approaches produced the best results when both energy saving and minimisation of SLA violation were considered at the same time. The combination of the fixed threshold overloading heuristic with the minimum migration time produced the highest energy saving but at the price of a higher SLA violations.

Similarly, Verma et al. [11] employ bin-packing to consolidate virtual machines by assuming that (1) a mechanism exists for predicting the optimal minimum size of a VM to meet its SLA goals, and (2) servers can be monotonously ordered according to their energy efficiency, meaning, if some server s_i is more energy-efficient for some application a_k than another server s_j, then s_i is more energy-efficient than s_j for *any* other application. To place and re-locate virtual machines in a cluster, they first sort all servers in decreasing order of their energy efficiency and all virtual machines in decreasing order of the optimium size (i.e. necessary allocation of resources). Secondly, they compute a mapping of virtual machines in such a way that starting with a theoretical utilization of 0 for all servers, they allocate as many virtual machines from the sorted list to the most-energy efficient server until its capacity is filled. The process is continued with the second most energy-efficient server and the remaining virtual machines, until all virtual machines are mapped to some server. Further iterations are made to efficiently utilise resources.

Xu et al. [12] propose a virtual machine consolidation strategy which aims to avoid SLA violation while minimising energy consumption. This is done by minimising interferences during migration and co-location of virtual machines. The authors employ a multi-dimensional supply-demand model to compute the interference of a migrated virtual machine on CPU, network I/O, CPU cash and memory bandwidth. Based on this quantity, candidate virtual machines are identified. The authors tested their approach using different workloads (including, SPEC CPU2006[4], netperf[5], Hadoop[6], and SPECweb2005[7]). The proposed consolidation strategy was tested on 10 homogeneous physical servers each having two quad-core Intel Xeon E5620 2.40 GHz processors, 12 MB shared LLC, 24 GB memory, and 800 GB NFS storage. The physical servers were connected via 1 Gbps Ethernet switches. The servers as well as 50 virtual machines were running

[2]https://code.google.com/p/cloudsim/

[3]http://aws.amazon.com/de/ec2/instance-types/

[4]https://www.spec.org/cpu2006/

[5]http://www.netperf.org/netperf/

[6]https://hadoop.apache.org/

[7]https://www.spec.org/web2005/

CentOS5.5 with Linux 2.6.18.8 kernel. Small VM instances were allocated with 1 virtual CPU core and 1.7 GB RAM, while large VM instances were allocated 4 virtual CPU cores and 7.5 GB RAM. Experimental results show that the proposed strategy can improve the performance of CPU and memory intensive workloads by 16% - 28% and network-intensive workloads by 45%-65% in comparison to First Fit Decreasing (FFD) [11] and Sandpiper [13] consolidation strategies. Even though the proposed approach is self-sufficient, it can be integrated into existing load balancing (Sandpiper [13]) or power-aware (FFD [11]) VM consolidation strategies.

Zhu et al. [14] proposed a consolidation framework to improve energy efficiency by placing together virtual machines which have complementary resource consumption characteristics. They define a distance metric that quantifies the peak resource demand of applications for all resources (CPU, memory, storage, network) together with an affinity score that favours applications with a direct communication link between them. Virtual machines are placed together if they are "near" to each other according to the distance metric. The authors evaluated their approach with two clusters each comprising of 64 servers. In one of the clusters, the servers integrate AMD Opteron 250 processors and in the other Intel Xeon E5345 processors. The authors claim that their approach was able to reduce the overall energy consumption by up to 55% compared to when the virtual machines were executed separately. The associated cost is increased execution time. When virtual machines with dissimilar resource consumption characteristics were co-located, execution times increased by almost 10%. The effect was pronounced when virtual machines with similar characteristics were executed. For example, for two CPU-intensive applications, execution times doubled, and for two memory-intensive applications, execution times increased by 41%.

In summary, the proposed approaches are tested with either simulation environment or static benchmarks. Often, the power and energy consumptions are estimated instead of measured. In contrast, our demonstrator uses real computing nodes (albeit their computing capability is scaled down) and a dynamic workload. We also use direct measurement to account for the power and energy consumption of individual nodes as well as the system as a whole.

III. Architecture

The HAECubie emulates a small-scale video hosting platform consisting of 30 nodes. Each node is represented by a Cubieboard2 platform with the following resource specification: 1 GB memory, dual-core ARM coretex-A7, 100 Mbps Ethernet, and 32 GB storage on SD Card. The nodes are divided into six groups and each group is connected to a 1 Gbps switch which is in turn connected to a master switch with the same capacity. The master switch is connected to a router so that video streaming requests can be received via the Internet. The organisation of nodes into groups enables dynamic power management at core, node, switch, and cluster levels. One extra node, the *master node*, is directly connected to the master

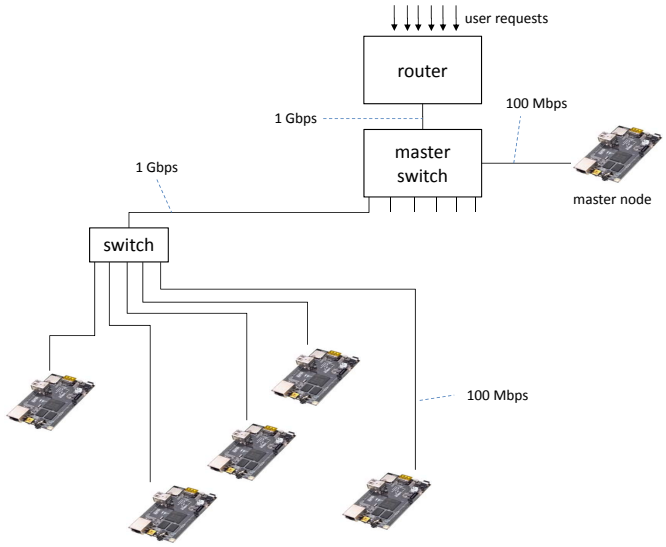

Figure 1. The hardware architecture of the HAECubie demonstrator.

switch and is in charge of managing the system as a whole. All incoming requests are accepted by the master node which then forwards the requests to one of the nodes. Individual nodes are directly responsible for streaming videos to users. Fig. 1 shows the hardware architecture of the HAECubie demonstrator.

At the software level, the video hosting platform consists of four services running on each node and five services on the master node. The services running on each node are a video management service, a monitoring service, an activity service, and a migration service. Each of these services have instances running on the master node as well. The master node in addition runs a power management service. Fig. 2 displays the software architecture of the video hosting platform.

At the node level, the video management service is responsible for streaming videos to users and for tracking and reporting the status of a video to the video management service running on the master node. The video management service at the master node is responsible for receiving and forwarding user requests and for calculating and updating the popularity of a video. It is also responsible for predicting the popularity of a video, which is useful for determining which sub-cluster a video should belong to (to be discussed in the next section). The monitoring service gathers statistics pertaining to the resource and power consumption of a node and reports them to the master. The monitoring service at the master collects and aggregates these statistics and stores them into the database. All tasks pertaining to adaptation, migration, and power management inside the demonstrator are asynchronous tasks; therefore, a mechanism is required to ensure that these tasks are executed to their completion and all deadlines are respected. The activity service will be notified when a task is issued and completed, so that it ensures these tasks are executed as intended. The activity service at the master overlooks the activities of individual nodes.

The migration service at the node level is responsible for mi-

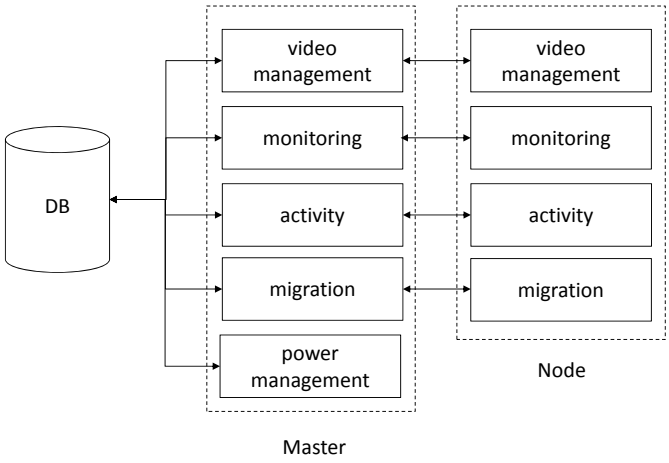

Figure 2. The software architecture of the HAECubie demonstrator.

grating videos between nodes, in a distributed manner, but the migration strategy comes from the migration service residing on the master node, which notifies nodes of these strategies. Finally, the power management service is responsible for turning on and off nodes and switches. The services interact with each other and with the database to exchange relevant information. In addition, the software platform provides a web-based administration support, so that the status of the entire cluster can be viewed at runtime and parameters which are relevant for adaptation (for example, popularity threshold) can be modified and adaptation policies can be manually selected. Fig. 3 displays the image of our demonstrator.

IV. Video Hosting

We deployed 1800 videos on the HEACubie which have different length and duration, the shortest video being 14 KB (with a streaming duration of approximately 1 s) and the longest 50 MB (with a streaming duration of approximately 100 s). In the beginning, the videos are randomly placed on all nodes. They gain popularity points every time they are requested and this way accumulate popularity over time. In order to accommodate sudden traffic surges, each node can utilise only up to 40% of its network bandwidth.

A. Consolidation

The purpose of workload consolidation is to dynamically balance the demand for and the supply of resources inside HAECubie, so that underutilised nodes can be switched off and the workload of overloaded nodes can be reduced to avoid a resource bottleneck. Because of the limited computing and communication resources in our demonstrator, we migrate data (videos) at runtime instead of virtual machines. Our demonstrator supports four types of consolidation strategies:

- All On (ON). This strategy turns on all nodes without performing any migration or load-balancing. It serves as a reference to all the other strategies.
- All On - Balanced (AOB). Uses all nodes, but tries to improve throughput and latency by distributing popular videos (by migrating them at runtime) on all nodes, so that bandwidth utilisation in the cluster is optimised.
- Load Only (LO). Activates the number of nodes that are necessary to meet the demands of the short term load of the cluster (which is estimated based on the statistics of recently accessed videos). It uses load-balancing to distribute popular videos among active nodes. The basic assumption behind this strategy is that recently streamed videos are more likely to be requested.
- Popularity (POP). This strategy classifies videos as popular and unpopular and nodes as class *A* and *B*. Class *B* nodes host predominantly unpopular videos and they are candidates to be switched off. They will be turned on *on demand*, when unpopular videos they are hosting are requested. In addition, the strategy balances the load of all active nodes.

To quantify the energy-utility trade-off that arises due to dynamic workload consolidation, we define streaming latency and throughput as utility. We evaluated the relationship of these two utilities with the power and energy consumption of the entire cluster.

B. Workload Prediction

The video management service at the master node uses a mean square estimation filter to predict the popularity of each video (the number of views) for the next time slot based on which it determines the resource demand of the HAECubie.

At time $t-1$ the video management service estimates the popularity of video x for the time slot $[t-1, t]$ based on the statistics it has up to that point in time. This quantity is denoted by $p_e(t)$. Meanwhile, between $t-1$ and t, it counts (measures) the number of requests for video x and this quantity is denoted by $p_m(t)$. Then, the estimated popularity of video x for the time slot $[t, t+1]$ is given as:

$$p_e(t+1) = p_e(t) + \alpha_x \left[p_m(t) - p_e(t)\right], 0 < \alpha_x < 1 \quad (1)$$

The expected error between the predicted and the actually observed popularity (the term inside the square brackets in Equation 1) determines the value of α_x. If this error is small (i.e., if the measurement is correlated with the estimation), then α_x will be large, otherwise, α_x will be small. Note that Equation 1 can be rewritten as:

$$p_e(t+1) = \alpha_x p_m(t) + (1-\alpha_x) p_e(t) \quad (2)$$

$$\begin{aligned} p_e(t+1) = & \\ &= \alpha_x p_m(t) + (1-\alpha_x)\left[\alpha_x p_m(t-1) + (1-\alpha_x) p_e(t-1)\right] \\ &= \alpha_x p_m(t) + \alpha_x(1-\alpha_x) p_m(t-1) + (1-\alpha_x)^2 p_e(t-1) \\ &= \alpha_x p_m(t) + \ldots + \alpha_x(1-\alpha_x)^{t-1} p_m(1) \end{aligned} \quad (3)$$

For most practical cases, the autocorrelation of p falls to zero after the k-th sample. Hence only the recent k samples are considered for estimation:

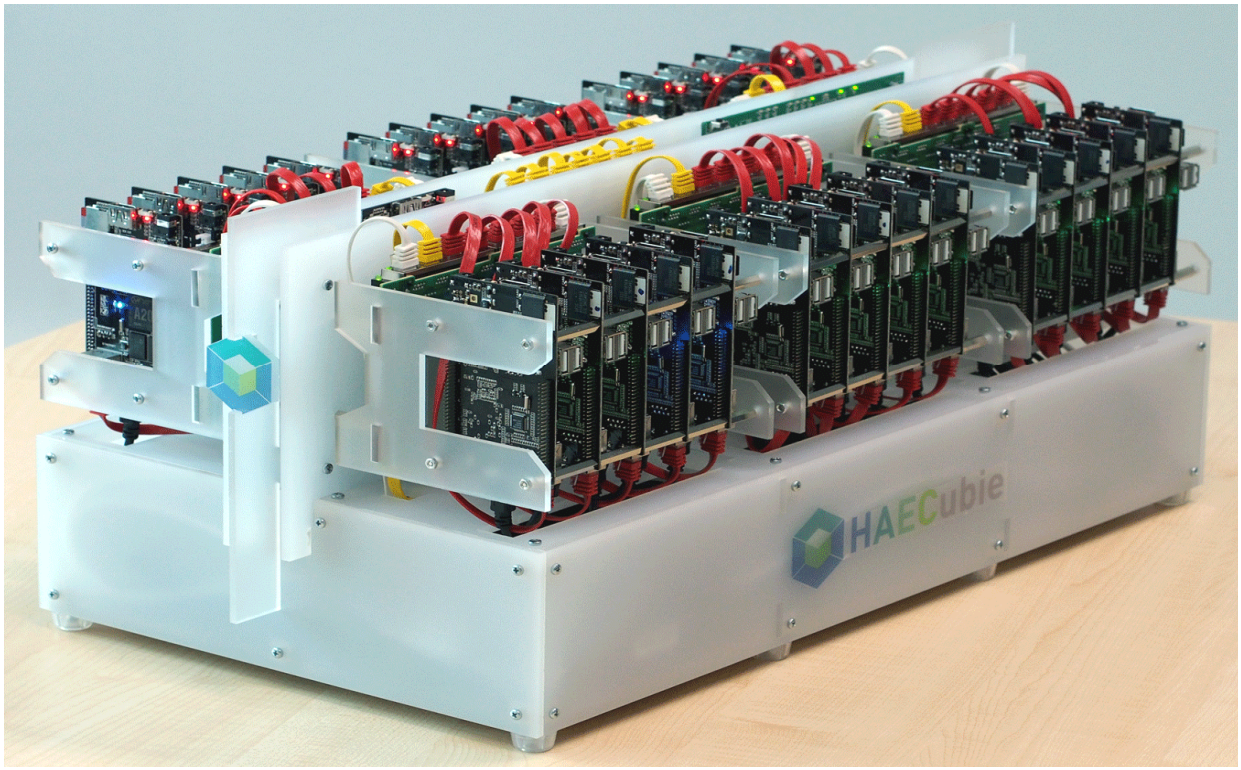

Figure 3. The HAECubie demonstrator.

$$p_e(t+1) = \alpha_x p_m(t) + ... + \alpha_x (1-\alpha_x)^{t-k} p_m(k) \quad (4)$$

Once the master's video management service estimates the popularity of each video for the next time slot, it computes the overall expected workload by aggregating the video popularities. This is useful to estimate the number of nodes required to handle the workload.

For the POP strategy, the master node defines a popularity threshold based on a specified energy-utility function. Those videos which have popularities above this threshold belong to Class *A* nodes, which should be always on. Those videos which have popularities below this threshold belong to Class *B* nodes, which should be switched off but can be switched on *on demand*, when unpopular videos are requested. Furthermore, the master notifies each node to which Class it belongs along with the popularity of the videos the node hosts. Based on this information each node migrates those videos it hosts which do not belong to its Class. Migration takes place on peer-to-peer basis, without the participation of the master node. Once Class *B* nodes are free of Class *A* videos, then the power management service puts them all into a sleep state, i.e., they will be switched off. Consequently, the access time for unpopular videos is relatively high, because of the booting time required to make a Class *B* node active.

C. Workload Generation

Video streaming requests have two statistically independent components, namely, the request arrival rate (**r**) and the streaming duration (or video size) of each requested video (**s**). For most practical scenarios both components are random variables. Thus, a workload **w** can be express as $\mathbf{w} = \mathbf{rs}$. We expressed the workload of our demonstrator by the joint probability density function $f(\mathbf{w}) = f(\mathbf{r}, \mathbf{s})$, which can be expressed as the convolution of the probability density functions of **r** and **s**:

$$f(w) = \int_0^{w/s} f(w/s)\, f(s) ds \quad (5)$$

The most plausible way of generating a realistic workload for our demonstrator is by reusing traces and log files from actual video hosting platforms. Unfortunately, existing platforms do not make traces and log files available for the public nor do they permit web crawler due to non-disclosure agreements and privacy concerns. Nevertheless, in the literature there exist several stochastic models that can estimate particular aspects of real-world workloads based on already available traces and log files. For example, some of these models estimate the distribution of file size and video popularity growth in YouTube, Youku, Dailymotion, and Metacafe [15], [16], [17], [18]. Hence, we convolved existing models, to generate the workload of our demonstrator. For the detail analysis of our workload generation strategy, we refer the reader to [8].

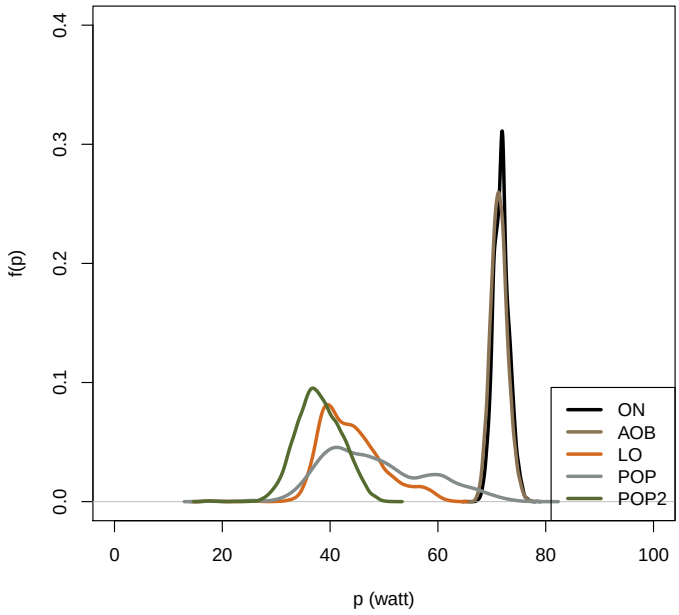

Figure 4. The density function of the power consumption of the different consolidation strategies.

V. Evaluation

To evaluate the energy-utility trade-off that can be achieved by the different consolidation strategies, we generated a stochastic workload and supplied it to the HAECubie. But the statistical parameters (the density functions of the workload size and the workload arrival rate) are the same for all the strategies. For each experiment the demonstrator streamed videos as per user requests for one hour. We used an extra 32-core server (with Intel Xeon E5-4603 processors and 16 GB RAM) to generate and feed the workloads to our HAECubie. In our evaluation we considered the overall energy consumption and the average power consumption of the system (the HEACubie only), the average throughput in terms of the number of bits per second the demonstrator was able to stream, and the video streaming latency (the time between the reception of a request and the beginning of streaming).

A. Energy and Power Consumption

Whereas the energy consumption is a measure of the cost of energy incurred to run the system, the power consumption is a measure of how much electrical load the system introduces on the power supply line. The amount of load the system should produce is specified by the contract between the client and the power supplier because violating this agreement may make the power supply line unstable and disturb the normal operation of other systems which share the same line. Since for our case each experiment was conducted for one hour, it is possible to directly infer the energy consumption of the system from its average power consumption.

Fig. 4 displays the density function of the power consumption of the HAECubie for the different strategies. Understandably, the always on strategy (ON) consumed the largest amount of power as well as energy. The always on with load-balancing strategy (AOB) has consumed the next largest. Apparently, the gain in power consumption as a result of load-balancing has been counterbalanced by an additional adaptation cost that was needed to migrate videos between nodes.

For the POP strategy, we considered two popularity thresholds: 0.25 (POP) and 0.125 (POP2). In the first, videos, the popularity of which is below the 0.25 quantile in the popularity distribution belong to Class *B* nodes which are normally switched off. In the second, the quantile threshold is 0.125.

Therefore, the POP strategy resulted in larger power and energy consumptions compared to the POP2 strategy, since during POP2 more nodes belonged to Class *A* than during POP. The LO strategy, which keeps all nodes active on which recently (in the past 300 s) accessed videos reside and turn off all other nodes except when they are demanded performed better than the POP strategy in terms of power and energy consumption.

B. Throughput

Throughput is a measure of the rate at which the HAECubie streams videos to the users. Interestingly, the ON strategy is the one which performed the poorest (as can be seen in Fig. 5), apparently, as a result of frequent bandwidth bottleneck at some nodes. This confirms to the significance of load balancing in server clusters and data centres. Nevertheless, load-balancing alone does not produce the highest throughput even if there is a surplus of computing and communication resources. In fact, all the consolidation strategies yield relatively higher throughputs compared with the ON and AOB strategies. The highest throughput was achieved when the POP strategy was used followed by the POP2 strategy, confirming to the fact that accurate estimation of the videos popularity enables to accurately balance the demand for and the supply of resources in video hosting platforms.

C. Latency

The challenge with dynamic workload consolidation is dynamic resource-pool sizing, particularly, when more sources are demanded. For our HAECubies, sometimes up to several tens of seconds are required to make videos available from nodes which are completely switched off. The time required to turn on a sleeping node depends on the workload of the system and how busy all the management services are. As a result it is a random variable.

As can be seen in Fig. 6 both the ON and AOB strategies have the smallest streaming latency. The average latency in these strategies is below 3 s. The POP strategy resulted in the highest average latency whereas the POP2 strategy is out performed only by the ON and AOB strategies.

D. Summary

Our experiments show that there is no single strategy that can outperform all the others by all accounts. Table I summarises the energy-utility trade-off that can be achieved by the different strategies. Whereas the POP strategy produces the

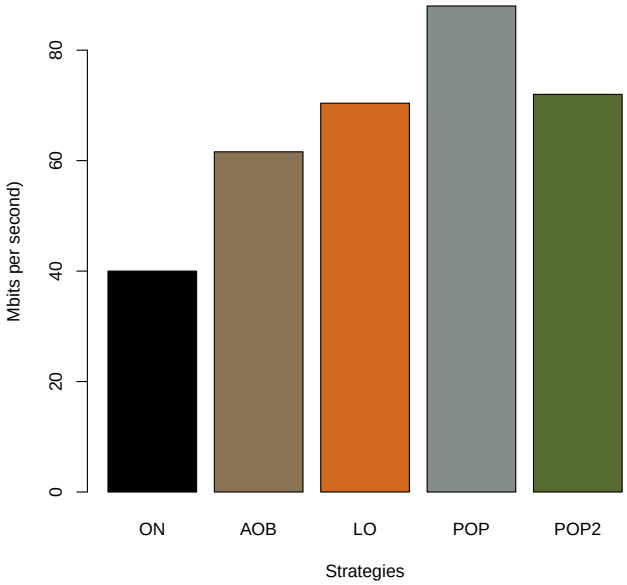

Figure 5. A summary of the throughput of the different strategies in Mbits per second.

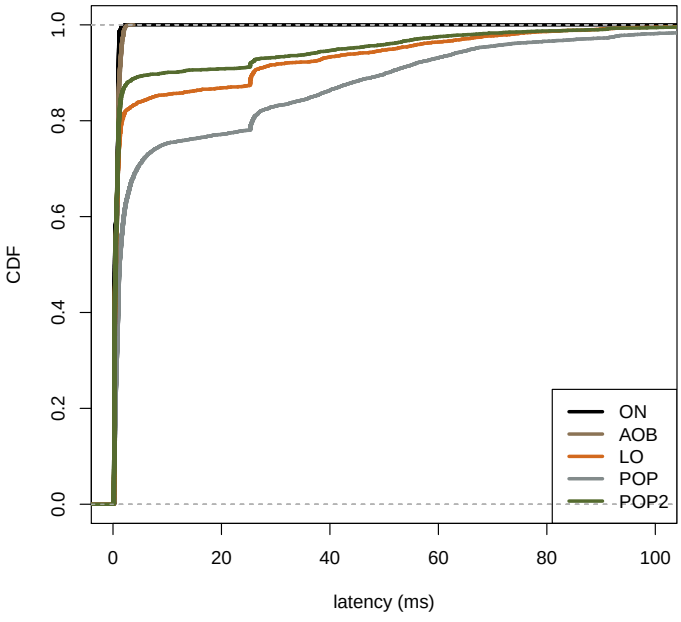

Figure 6. The CDF of the latency of the different consolidation strategies.

Strategy	ON	AOB	LOAD	POP	POP2
Energy consumption (Watts-hour)	71.626	71.3	48.3	56.1	45
Mean power consumption (Watt)	71.8	71.4	44.0	48.5	38.0
Throughput (Mbps)	40.0	61.6	70.4	88.0	72.0
Gain in % (Throughput)	0	54	76	120	80
Gain in watt (Avg. Power)	0.0	0.56	38.7	32.5	47.1

Table I
SUMMARY OF ENERGY-UTILITY TRADE-OFF FOR DIFFERENT CONSOLIDATION STRATEGIES.

largest throughput for a small amount of average power consumption, it is also the one which produces the largest delay. On the other hand, the POP2 strategy has the smallest amount of average power consumption and the third best in average latency. It has also the second best in throughput. From this it is possible to conclude that the threshold of the popularity strategy can be adjusted to achieve the optimal energy-utility trade-off for a named utility. The optimal threshold, however, should take the hardware and software architectures as well as the workload of the system into consideration.

In terms of gain (by taking the outcomes of the ON strategy as a baseline), the POP2 improved throughput by 120% whereas it reduces the power consumption by 32.5% only. The highest gain in terms of saving power is achieved by the AOB strategy, but when the corresponding throughput gain is considered, the POP2 has the highest gain.

VI. CONCLUSION

In this paper we introduced our HAECubie demonstrator which we built to experimentally investigate the scope and usefulness of different adaptation (workload consolidation) strategies to achieve high performance and energy-efficient computing in data centres and server clusters. Our demonstrator consists of 30 nodes connected with each other via six Gbps switches. An additional node, a master node, is responsible for computing runtime statistics, distributing the statistics to all active nodes, and initiating load-balancing and migration at runtime.

Our demonstrator is a video hosting platform. It hosts 1800 videos of variable sizes. Initially, these videos are arbitrarily distributed in the 30 nodes but the master node keeps track of the popularity of the videos as they are requested by users. Then it aggregates the popularities of all the videos and based on the aggregated statistics estimates the future workload of the system and decides the number of nodes it should keep active in order to handle the workload. Then video migration and load-balancing take place to switch off idle or underutilised nodes and to avoid hotspots. Both migration and load-balancing take place in a distributed manner, without the participation of the master node.

We implement three types of consolidation strategies. In the AOB strategy, all nodes are active but load-balancing is used to avoid resource bottleneck. In the LO strategy, all nodes from

which videos are requested in the recent past (for our case, the past 300 s) remain active. We chose 300 s because this was the maximum time we experienced to make a sleeping node available for service. Therefore, an adaptation strategy should have at least this much lifetime to justify its usefulness. The POP strategy uses the popularity distribution of all nodes to determine the number of nodes that should be active to handle the incoming workload.

Our experiment results confirm that all strategies performed well when compared with the baseline in which all nodes were active and no load-balancing was employed. However, they also show that there was no single strategy that can achieve high energy-efficiency, low streaming latency, and high throughput at the same time. The POP strategy, with the selection of the appropriate threshold, can achieve most of these goals.

Because we used relatively simple boards to construct our demonstrator, these boards experienced failure when they were frequently switched on and off. Consequently, some of our measurements did not make sense to us during statistical analysis. Consequently, we rejected them as outliers. In future we plan to look closer into these problems and make the demonstrator more robust to failure and faults. Moreover, due to the limited bandwidth we have, we migrated videos instead of runtime containers (virtual machines). Therefore, the runtime migration service always waits until a video streaming is over before a video is migrated. We are, however, confident that this is not a major departure from the way dynamic workload or virtual machine consolidation takes place in real data centres and server clusters.

ACKNOWLEDGEMENT

This work has been partially funded by the German Research Foundation (DFG) under project agreement: SFB 912/1 2011.

REFERENCES

[1] Cisco Inc., "Cisco global cloud index: Forecast and methodology, 2011–2016," 2012.

[2] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. N. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[3] A. Benner, "Optical interconnect opportunities in supercomputers and high end computing," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, pp. 1–60, 2012.

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

[5] 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*, pp. 7:1–7:13, ACM, 2012.

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

[7] A. Brihi and W. Dargie, "Dynamic voltage and frequency scaling in multimedia servers," *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, vol. 0, pp. 374–380, 2013.

[8] C. Möbius and W. Dargie, "Statistical analysis of the workload of a video hosting server," in *Analytical and Stochastic Modelling Techniques and Applications - 21st International Conference, ASMTA 2014, Budapest, Hungary, June 30 - July 2, 2014. Proceedings*, pp. 223–237, 2014.

[9] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[10] K. Park and V. S. Pai, "Comon: A mostly-scalable monitoring system for planetlab," *SIGOPS Oper. Syst. Rev.*, vol. 40, pp. 65–74, Jan. 2006.

[11] A. Verma, P. Ahuja, and A. Neogi, "pmapper: Power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, (New York, NY, USA), pp. 243–264, Springer-Verlag New York, Inc., 2008.

[12] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, and B. Li, "iaware: Making live migration of virtual machines interference-aware in the cloud," *Computers, IEEE Transactions on*, vol. 63, pp. 3012–3025, Dec 2014.

[13] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX conference on Networked systems design anbd implementation*, NSDI'07, (Berkeley, CA, USA), pp. 17–17, USENIX Association, 2007.

[14] Q. Zhu, J. Zhu, and G. Agrawal, "Power-aware consolidation of scientific workflows in virtualized environments," technical report, Ohio State University, 2010.

[15] X. Cheng, C. Dale, and J. Liu, "Statistics and Social Network of YouTube Videos," *2008 16th Interntional Workshop on Quality of Service*, pp. 229–238, June 2008.

[16] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1357–1370, 2009.

[17] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network – Measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, 2009.

[18] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing web-based video sharing workloads," *ACM Transactions on the Web*, vol. 5, no. 2, pp. 8:1–8:27, 2011.