Real-Time Power Cycling in Video on Demand Data Centres using Online Bayesian Prediction

Vicent Sanz Marco, Zheng Wang, Barry Porter
School of Computing and Communications, Lancaster University, UK

***Abstract*—Energy usage in data centres continues to be a major and growing concern as an increasing number of everyday services depend on these facilities. Research in this area has examined topics including power smoothing using batteries and deep learning to control cooling systems, in addition to optimisation techniques for the software running inside data centres. We present a novel real-time power-cycling architecture, supported by a media distribution approach and online prediction model, to automatically determine when servers are needed based on demand. We demonstrate with experimental evaluation that this approach can save up to 31% of server energy in a cluster. Our evaluation is conducted on typical rack mount servers in a data centre testbed and uses a recent real-world workload trace from the BBC iPlayer, an extremely popular video on demand service in the UK.**

***Keywords*-Energy; data centre; prediction; video on demand**

I. Introduction

Video on demand services have become increasingly popular over the last decade, with services like Netflix, Amazon Instant Video, and IP-based broadcast solutions.

These services are supported by data centres, which stream requested content to users. In many cases these facilities use dedicated servers (rather than shared cloud hosting) so that available latency and bandwidth, and therefore Quality of Service (QoS), is predictable. Data centres like this also often *over-provision* resources based on a projected peak load to ensure that all requests can be served with acceptable QoS at peak times [1]. This tends to mean that servers are not fully utilised at all times, however, leading to energy waste [2]. These services are also known to experience fluctuations in demand over the course of a day, week, month and year as new popular content is released [3].

At the same time, data centres are known to be significant energy consumers, with a broad range of efforts under way to reduce this level of energy consumption, such as, power distribution [4], improving cooling system efficiency [5] or data centre network architectures [6], among other techniques.

We propose an approach in which we use real-time prediction to determine the number of servers needed at any point in time, allowing those that are not currently needed to be temporarily switched off until demand increases. To enable this our approach uses three novel features for a video on demand service, including both prediction and content distribution within the server cluster:

- A predictive model of which videos will and will not be requested by users in the near future (considering the upcoming 30 minutes of requests), using a fused model of the last two hours of requests together with the same two hour period from the previous week.
- A framework in which popular content is co-located on a subset of available servers in the cluster, increasing the probability that servers with unpopular content can be turned off when no requests are projected for them.
- The use of short *intro clips*, distributed to all servers in the cluster, which allow users to view the first two minutes of a video while a server with the full copy is being turned on, in case of prediction errors.

We use standard Linux rack mount servers, with a user-level signal to switch servers off and a Wake-on-LAN signal to turn them on. Our approach is the first example of a predictive model that is automatically and continuously updated in real-time based on ongoing requests for a video on demand service. Our approach is tested in real-world case using real trace data from BBC iPlayer achieving 31% reduction in energy use without impacting QoS

II. Background

The British Broadcasting Corporation (BBC) is a UK-based public service broadcaster of radio and television, in addition to an extensive online platform. The BBC iPlayer platform is an online streaming service that allows users to watch and listen to live TV and radio programmes, across 9 TV channels and 57 radio stations. In addition, users can access any content from across these channels that was broadcast in the last 30 days. The iPlayer supports a range of client devices, including mobile phones, tablets and PCs.

By 2013, BBC iPlayer was the second most popular on-demand streaming application in the UK after YouTube, with around 40% of UK households using it regularly [7].

The BBC operates a set of dedicated data centres around the UK for streaming iPlayer videos; our approach targets the software infrastructure of these facilities. The data set that we utilise for experimental evaluation is real iPlayer data from two weeks of access logs, from 3rd of October 2016 to 16th of October 2016. We use this data to validate our approach on a test server cluster by replaying the trace in real-time to simulate the same client request pattern.

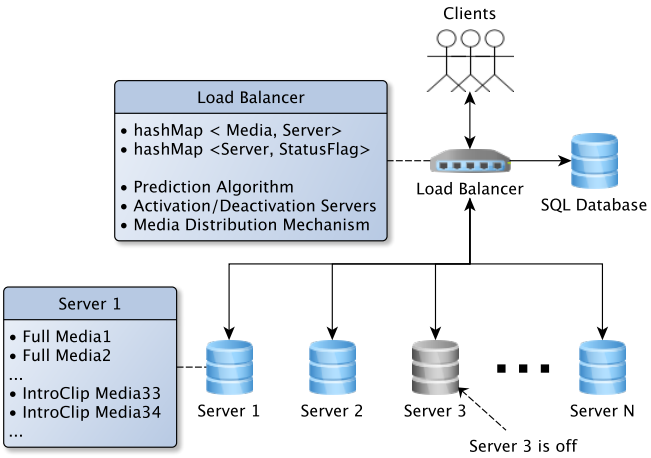

Figure 1: Overview of our system architecture.

III. Approach

An overview of our system architecture is shown in Fig. 1. A server acting as a load balancer receives video requests from clients and directs them to an appropriate media server. It also runs our prediction algorithm and controls the activation and deactivation of servers.

The load balancer contains a hash map storing which content is present on which media servers (where each content item may be replicated across multiple media servers) in addition to a second hash map containing an on/off status flag for each media server reflecting its current status.

When a client requests a video, the load balancer checks which media servers can serve that video, checks their current load level, and also verifies whether or not they are currently on. If all media servers that can serve the full video are switched on, the client request is simply directed to the server with the lowest load in that group.

Otherwise, if no servers that contain the full video are switched on, or if only some such servers are switched on but are at peak capacity (where the peak capacity level of each server is defined via a local policy), this implies there is another server available to serve the full video which is currently switched off. In this case, the load balancer sends a *powerOn* signal to that server and directs the client to another server which contains an *intro clip* version of the video. This means that the prediction algorithm made an error in its estimation of which servers need to be turned on, triggering our error mitigation strategy of just-in-time server start-up. Such prediction errors are automatically learned by our prediction model to help avoid them in the future.

Intro clips used in our approach are two minutes long and so can have a much higher replication factor across all media servers. Two minutes was selected because, in our tests, media servers needed an average of 46 seconds to turn on and be operational after receiving a *powerOn* message.

When a client is reaching the end of the intro clip version of their requested video, the load balancer redirects that client to the newly activated media server with the full-length version, allowing a seamless handoff. This is imperceptible to users, who have the impression of one continuous stream.

A. Media distribution

Our approach requires two decisions to be made on the distribution of media content: the replication factor of *full versions* of each media item, and the size and replication factor of *intro clips*. Because not all servers are switched on at all times, the *co-location* of full media content items is also important; we choose to co-locate popular media on the same servers to increase the probability that popular media can be served without needing to power up another machine, therefore heuristically reducing the overall amount of power cycling that takes place. The replication factor, and co-location affinity, of full media content is recalculated once per day based on observations of popularity in that day.

In detail, our media distribution protocol stores all *intro clips* for the entire media library at *all servers* in the cluster.

For full versions of media items, we define a peak load capacity of each media server, which is set to 100 simultaneous connections for our experiments[1]. We then attempt to ensure that no server needs to handle more than this number of simultaneous connections by sufficiently replicating popular content across servers in the cluster. This replication is performed between 3 am and 4 am, reflecting studies that show relatively few user connections to data centres between 1 am and 6 am compared to the rest of the day [8].

Our strategy for media distribution consists of two steps: *distribution* and *replication*. Firstly, each server is given a rank index from 1 to N, where N is the number of servers in the cluster and 1 is considered to be the highest rank. The way in which this ranking is assigned is configurable; in this paper we use the performance and capacity level of servers as our ranking metric. Each video is then ranked based on its total number of requests in the last 24 hours, where the highest ranked video has the highest number of requests. The contents of the entire media library V are divided across the set of servers N in the cluster, such that the highest-ranked server has the V/N most popular videos, and the following server contains the next V/N popular videos from the library, etc. At the end of this process a cleanup phase occurs in which any videos that do not belong at a server according to the above assignment protocol are deleted.

When the distribution stage is complete, our replication stage is performed. This is used to control the load of servers, copying videos among the servers to avoid exceeding specified peak loads. It works by predicting the number

[1]Determining the actual peak capacity of a server depends on a large range of factors, including its network bandwidth, the network infrastructure characteristics in the data centre, and the kinds of clients that connect to a server (i.e., smartphones may take less bandwidth than PC users as their videos have lower bitrates). We assume that peak capacity can be estimated by network administrators and assigned as a policy for each server.

of connections that each video will receive in each hour of the day, and copying a video to the next highest ranked server if the server's overall connection volume exceeds the peak load specification in any hour; this copying is repeated until videos are sufficiently replicated that peak load capacity is not exceeded or there are no more servers to copy to.

B. Server activation and deactivation

During the course of a day our framework needs to turn servers on and off as user activity changes. To turn a server off we use an agent running on each server which listens for *powerOff* commands, sending a shutdown signal to the operating system when this command is received.

Turning a server on is more complex, with several possible protocols that can be used, such as Intelligent Platform Management Interface (IPMI) or Wake-on-LAN (WoL). In our current implementation we use Wake-on-LAN. It is important to ensure that the length of *intro clips* is longer than the average startup time of a server, measured from the time at which a *powerOn* signal is sent. We measure this to be 46 seconds on average and use 2 minutes as our intro clip length to allow for unexpected delays in server startup.

C. Prediction

We use prediction to help determine when to turn servers on and off. Servers are turned on either proactively when it is predicted that they will be needed in the next 5 minutes, or reactively when a client request has arrived which cannot be served by a media server that is currently on. Servers are turned off when they have no clients using them and it is predicted that they will not be needed within the following 5 minutes and so can be powered down to save energy.

Prediction is performed in real-time using two data sources: a window w_r of the most recent media requests, of configurable length, and a window w_h of media requests from the period as w_r but from one week ago. Each request item in these two windows contains the time at which a user made a request, the media item requested, and the length of time they watched that item. The data for w_r is stored on the load balancer, while w_h is retrieved from an SQL database.

Both streams of data are fed together into a prediction model, for which we use Naive Bayes. This is done incrementally for the two streams, so that time step n from w_r and w_h are fed together into the prediction model, followed by time step $n+1$ from w_r and w_h, and so on. This effectively results in the predictive model's output being informed with equal weight by the data from both time windows.

For this paper we use 2 hours as the length of w_r and w_h, with a forward prediction duration of 30 minutes. This forward prediction indicates which media items are likely to be requested at each time point through that 30-minute window, and can thus be used to determine which servers likely need to be switched on versus those that can safely be switched off during that 30-minute time window.

The prediction model is continually updated on a rolling basis every 1 minute using Naive Bayes Inference [9], moving w_r and w_h ahead in step with real time as more requests arrive, incrementally feeding new requests into the model and removing requests from the model that are no longer within the prediction data time window.

IV. Experimental Setup

In this section we present our experiment setup, including the model by which we characterise energy consumption. In the following section we then present our results.

A. Hardware and Software Platforms

Hardware infrastructure: Our experiments attempt to replicate the BBC iPlayer service data centre setup, using a data centre testbed and a real trace of user requests from the iPlayer to model actual user dynamics over a two week period. Specifically, we use five machines for our experiments. Four are used as media servers, which are Dell PowerEdge R210 II rack mount servers. Each has a 16-core Intel Xeon CPU E5-2620 v4 @ 2.10GHz with 16 GB of DDR4 RAM. A fifth server is used as a load balancer, which is a Dell Optiplex 9020 with 4-core Intel i7-4790 CPU @ 3.60Hz with 8 GB of DDR3 RAM.

Software infrastructure: We deploy our systems on Ubuntu Server with Linux kernel 16.04. Our framework is implemented in Java, for which we use version 1.8.0_91. Additionally, our prediction model in particular made use of Weka 3 [10], a machine learning algorithm collection library that can be used by Java applications. Our long-term iPlayer request data is stored in a MySQL database.

B. Evaluation Methodology

Although we have access to the set of user requests for content, we do not have access to the actual media content to which requests refer. We therefore created similar content that matches the request sequence. For this purpose we created 80 unique videos for requests spanning 7 days. Our experiments were conducted in real-time by simulating client requests from the trace, such that 1 second in the real world is equivalent to 1 second in the experiments.

The trace covers a period from 00:00 on 3rd of October 2016 to 23:59 on 16th of October 2016. Our experiments use the second half of this data, from 00:00 on 10th of October 2016 to 23:59 on 16th of October 2016, so that we can use the first half as our historical prediction window w_h.

At the beginning of our experiments all servers were turned on, as the prediction model has not yet gathered any data. Consequently, during the first two hours of 10th of October all machines are on and our framework is collecting the information required about user connections and media demand. After this two first hours, windows w_r and w_h have enough information to start performing the user demand prediction mechanism. While the prediction is performed

Power at Full Load	130W
Measured Idle Power	44W
Off	20W

Table I: Server power consumption

continually, the media distribution algorithm is executed once every day between 3 am and 4 am.

Energy Consumption: During the course of our experiment we did not have a mechanism to directly measure the actual energy consumption of a server (for example by attaching energy monitoring equipment). Instead, we estimate this using a simple energy model in which servers are in one of three states: off, idle and full load.

Table I shows server power consumption in each state, according to manufacturer specifications. In the table, 'full load' represents the energy utilisation when the machine has 100% CPU load and 'off' its Wake-on-LAN standby power.

$$Watt\text{-}hours = \frac{1}{60} * \begin{bmatrix} (minFullLoad * fullLoadWatts) \\ +(minIdle * idleWatts) \\ +(minOff * offWatts) \end{bmatrix} \quad (1)$$

Using these three states, we use Eq. 1 to model the total energy consumption of a server over an hour. In the equation, *MinFullLoad*, *minIdle* and *minOff* variables are the minutes in an hour that a server is in the full load, idle and off states, respectively. *FullLoadWatts*, *idleWatts* and *offWatts* are the values in Table I for full load, idle and off statuses.

For instance, if a server spends 30 minutes switched off, 20 minutes in idle mode and 10 minutes at full load, then, using this equation, we can estimate that the server has consumed approximately 46.33 Wh in that time.

Using Eq. 1, our experiments compare the data centre energy spent in four cases:

- **Default**: The data centre is not using our predictive mechanism nor our media distribution mechanism, such that media is randomly distributed across media servers.
- **Predictive**: The data centre is using our predictive mechanism but not our media distribution mechanism (again using random media distribution).
- **Distributed**: The data centre is using our media distribution mechanism but not our predictive mechanism.
- **Both**: The data centre is using our predictive mechanism and our media distribution mechanism.

V. Experimental Evaluation

In this section we first show the energy performance of the above four test cases. We then provide a detailed prediction accuracy analysis, including the effects of using different window sizes, and finally we present a comparison of different machine learning approaches used for prediction.

A. Energy Consumption

We examine energy consumption both on the first day of the experiments, when our prediction model is starting from no information, and also across the whole seven days.

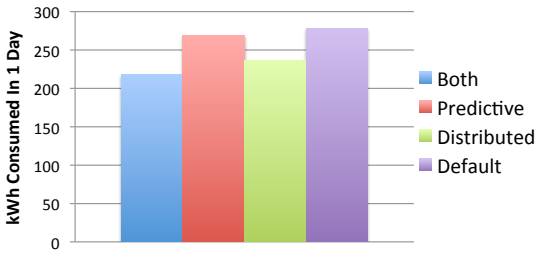

Figure 2: Power consumption in first day experiments.

Case	First day		Seven days	
Both	£8.27	60.39 kWh	£613.88	640.59 kWh
Predictive	£1.33	9.74 kWh	£65.36	68.20 kWh
Distributed	£5.77	42.15 kWh	£282.75	295.05 kWh

Table II: Energy and financial saving in first and seven days, compared to default case.

Fig. 2 shows the energy consumed during the first day of experiments in the above four scenarios. By examining scenario separately we are able to more clearly see how much each element of our approach contributes to overall energy. For each scenario we repeat the experiment twice and average the results; note that because we run these experiments on real hardware and in real-time, this represents a total of 8 weeks of experimentation time.

For the first scenario, in which we only use our prediction method (with random media distribution), we see an energy saving of 1.11% compared to the baseline of random media distribution always-on servers. This demonstrates that, even without a smart media distribution system in which popular items are co-located, there is still some chance that servers can be turned off in a realistic trace.

In the second scenario, in which we use our media distribution approach but not our prediction model, we see an energy saving of 13.8% compared to the baseline case. This is because more servers are running at 'idle' more of the time, as popular content is co-located on fewer servers.

Finally, when using *both* our prediction model and media distribution approach, we see an energy saving of 19.62%, due to some servers being switched off some of the time.

Table II shows an estimate of the money saved in the first day if the system is implemented with our prediction model and/or media distribution mechanism[2].

Fig. 3 shows the energy consumed during the entire seven days of experiments according to the same four different scenarios. As we can see in the figure, we achieved an energy saving percentage of 15.23% using only our prediction model, 26.12% using only the media distribution system, and 31.1% using both mechanisms.

The last two columns in Table II estimate financial sav-

[2]Based on a UK average energy cost of 13.69p/kWh in September 2016, https://www.sust-it.net/energy-calculator.php?tariff=9

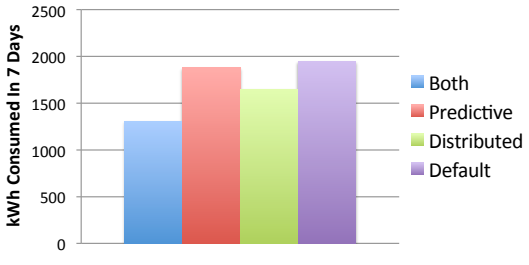

Figure 3: Power consumption in seven day experiments.

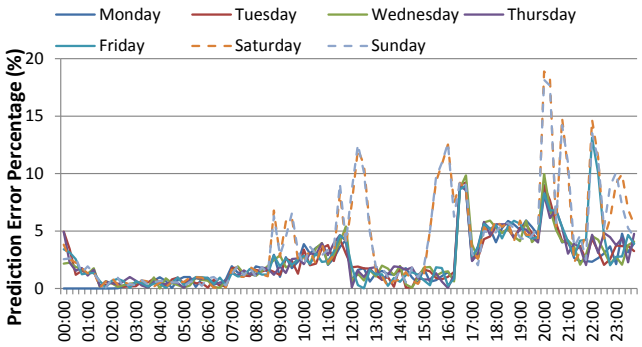

Figure 4: Prediction error percentage per day.

ings over seven days compared to the default case. If we extrapolate these results out to an entire year, we estimate that the money saved will be approximately £31,921.76.

B. Prediction Accuracy

Prediction errors occur when a user requests media from a server that is not currently powered on. This results in the user being directed to an intro clip version of their media stream while the server that contains the full version is turned on; correspondingly there is unexpected extra load at servers hosting *intro clips*. In this section we examine the frequency of prediction errors and their causes. Prediction error is calculated as the ratio between the total number of media requests in a 15 minute period, and the number of prediction errors that occur in the same time period.

Fig. 4 shows the prediction error percentage per day, divided into time slots of 15 minutes. Note that the first two hours on Monday have 0 error because the model does not yet have enough information to make predictions.

The prediction model generally works very well, particularly during week days from Monday to Friday. However, these are also clear peaks where prediction performs less well. This is evident towards the end of a day (approaching midnight) when there are far fewer user connections in general and so the trend is harder to extract. There is also a clear rise in prediction errors from 07:00, which is caused by a continuous increase in users until around 12:00, for which the prediction model needs to continually account for until the user population reaches a stable point.

Errors are also evident on weekend days (Saturday and Sunday), where we find that user activity is simply far less

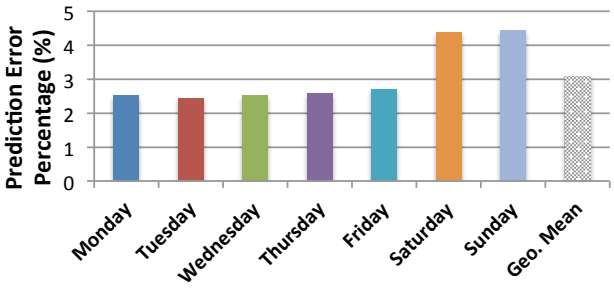

Figure 5: Average prediction error percentage per day.

predictable in general, and there are is also a higher volume of users suggesting that many of them are occasional users and therefore more erratic. The majority of peaks are brief, however, as the prediction model catches up with the change and returns to making accurate predictions.

To show the effect of different days more clearly, Fig. 5 shows the total average prediction error accuracy per day. Here we can strongly see that Saturday and Sunday have the worst average prediction accuracy due to the volume of users connected and the corresponding probability that they will make requests for 'unusual' videos. Overall however the average prediction error across our experiments is very low considering the volume of requests, at around 3.08%.

VI. Related Work

With the increasing number of everyday online services that rely on data centres, energy and cost efficiency for data centres has recently received a significant amount of attention in the research community [11], [12], [13], [14]. Here we focus specifically on power-cycling techniques.

Chun et al. [15] developed an architecture that uses heterogeneous platforms to save energy. These platforms are divided in two systems: a high performance system and a low performance system. Initially all tasks are run in the high performance system, and an energy consumption boundary is defined. When a server from the high performance system exceeds this boundary, the approach migrates all tasks from this server to a low performance system server and then turns off the previous high performance system server. Our approach avoids the use of real-time migration, which would be costly for large video files, instead using a media distribution approach based on recently observed activity.

Nam *et al.* [16] created a mechanism to turn off *network switches* in the data centre network. *Network switches* are responsible of redirecting the clients to the correspondent machine in the data centre. These machines are turned off using real-time network information, such as network state, traffic flows and servers' states. This work is complementary to ours; while this paper is concerned with turning on/off *network switches*, our approach focuses on turning on/off service-provider machines in the data centre based on a predictive machine learning model.

Zhang and Ansari [17] created a mechanism called *Hierarchical EneRgy Optimisation* (HERO) that reduces the power consumption of network devices in a data centre. The authors created a turn on/off schedule for each device in the data centre, according to a previous analysis made of system behaviour. The main shortcoming of this approach is that it is not flexible in adapting to workload fluctuations; instead we make much finer-grained, real-time predictions using only a small window of historical data.

Mathew et al. [18] present a similar approach to ours, focusing on turning off machines in Content Delivery Networks (CDNs) during periods of low computation load. They present two different theoretical algorithms to achieve this goal, fusing offline and online data. However, the approach is not evaluated under realistic conditions and does not consider factors of server power-up time and its effects on QoS; in contrast, our approach considers the complete end-to-end system requirements, including the time taken to switch servers on and the appropriate distribution of media content, and has been tested against a real-world experiment trace from the popular BBC iPlayer service.

VII. Conclusions

We have proposed a predictive power cycling approach to energy reduction for video on demand data centres. Our approach is based on a media distribution system, an *intro clip* mechanism, and a real-time prediction model. We have experimentally evaluated our approach on a real server cluster, using trace data of real user requests from the BBC iPlayer as input over a two week period. We find that, using our fused prediction model, average prediction error is low at around 3.08%. This is done using a computationally cheap real-time prediction model, without training, and which is responsive to changes in the request pattern over time.

Overall this results in an energy saving of 31% compared to an approach that leaves all servers switched on all of the time, which is the default approach in the majority of data centres today, demonstrating that our real-time power cycling approach is a promising avenue for investigation.

In future work, we intend to further improve our prediction mechanism to deal with high-demand real-time events that might appear, such as Olympic games or football matches, as well as explore the generalised application of this approach to areas beyond video on demand services.

Acknowledgements

This work was partly supported by the UK EPSRC under grants EP/M029603/1 (*Deep Online Cognition*), EP/M01567X/1 (SANDeRs), and EP/M015793/1 (DIVIDEND). We would also like to thank the BBC R&D department for curating the trace data on which our experiments are based.

References

[1] P. Delforge, "America ' s Data Centers Are Wasting Huge Amounts of Energy," *Natural Resources Defense Council (NRDC)*, vol. IB:14-08-A, no. August, pp. 1–5, 2014.

[2] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[3] Y. Elkhatib, R. Killick, M. Mu, and N. Race, "Just browsing?: Understanding user journeys in online tv," in *ACM MM*, 2014.

[4] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood, "Power Routing: Dynamic Power Provisioning in the Data Center," in *Proc. of the 15th Conference on Architectural Support for Programming Languages and Operating Systems*, 2010, pp. 231–242.

[5] D. Quirk and M. Patterson, "The "right" temperature in Datacom environments," in *ASHRAE Transactions*, vol. 116, no. PART 2, 2010, pp. 192–204.

[6] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges," *ArXiv e-prints*, Jun. 2010.

[7] D. M. Harrison, "The communications market report 2016," OFCOM, Tech. Rep., 2016. [Online]. Available: https://www.ofcom.org.uk/research-and-data/cmr/cmr16

[8] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, "Energy management for commercial servers," *IEEE Computer*, vol. 36, no. 12, pp. 39–48, 2003.

[9] G. C. Box, George EP and Tiao, *Bayesian inference in statistical analysis*, 40th ed. John Wiley & Sons, 2011.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[11] G. L. T. Chetsa, L. Lefevre, J. M. Pierson, P. Stolf, and G. Da Costa, "Beyond CPU frequency scaling for a fine-grained energy control of HPC systems," in *Proc. of the Symposium on Computer Architecture and High Performance Computing*, 2012, pp. 132–138.

[12] H. Chen, Y. Li, and W. Shi, "Fine-grained power management using process-level profiling," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 1, pp. 33–42, 2012.

[13] J. Chang, J. Meza, P. Ranganathan, A. Shah, R. Shih, and C. Bash, "Totally Green : Evaluating and Designing Servers for Lifecycle Environmental Impact," in *Proc. of the 17th Conference on Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 25–36.

[14] P. Petoumenos, L. Mukhanov, Z. Wang, H. Leather, and D. S. Nikolopoulos, "Power capping: what works, what does not," in *IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2015, pp. 525–534.

[15] J. B. Marcinichen, J. A. Olivier, and J. R. Thome, "On-chip two-phase cooling of datacenters: Cooling system and energy recovery evaluation," in *Applied Thermal Engineering*, vol. 41, 2012, pp. 36–51.

[16] T. M. Nam, N. H. Thanh, and D. A. Tuan, "Green data center using centralized power-management of network and servers," in *2016 International Conference on Electronics, Information, and Communications (ICEIC)*, January 2016, pp. 1–4.

[17] Y. Zhang and N. Ansari, "Hero: Hierarchical energy optimization for data center networks," *IEEE Systems Journal*, vol. 9, no. 2, pp. 406–415, June 2015.

[18] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *Proc. of IEEE INFOCOM*, March 2012, pp. 954–962.