Modelling Low Power Compute Clusters for Cloud Simulation

Gabor Kecskemeti, Wajdi Hajji, Fung Po Tso
Department of Computer Science
Liverpool John Moores University
Liverpool, United Kingdom
Email: {g.kecskemeti, p.tso}@ljmu.ac.uk; w.hajji@2015.ljmu.ac.uk

***Abstract*—In order to minimise their energy use, data centre operators are constantly exploring new ways to construct computing infrastructures. As low power CPUs, exemplified by ARM-based devices, are becoming increasingly popular, there is a growing trend for the large scale deployment of low power servers in data centres. For example, recent research has shown promising results on constructing small scale data centres using Raspberry Pi (RPi) single-board computers as their building blocks. To enable larger scale experimentation and feasibility studies, cloud simulators could be utilised. Unfortunately, state-of-the-art simulators often need significant modification to include such low power devices as core data centre components. In this paper, we introduce models and extensions to estimate the behaviour of these new components in the DISSECT-CF cloud computing simulator. We show that how a RPi based cloud could be simulated with the use of the new models. We evaluate the precision and behaviour of the implemented models using a Hadoop-based application scenario executed both in real life and simulated clouds.**

***Keywords*-cloud computing; IaaS; low power; simulation; containers; docker**

I. Introduction

Infrastructure as a service (IaaS) clouds offer unprecedented flexibility and elasticity for virtual infrastructure maintainers, while eliminating the burdens of actually operating data centres and physical infrastructures. The always-on and always-ready to scale nature of a modern cloud requires IaaS providers to utilise a wide variety of cutting-edue technologies at the background (e.g., virtualisation, containers, orchestrators etc.) to stay competitive. One of the promising ways to increase competitiveness is to go "green" and "low-cost" – through applying novel low power architectures as core building blocks of the data centres. For example, ARM-based compute nodes, which have been widely adopted in today's Smartphones and tablet's, could be used instead of the now widespread Intel-based ones.

One of the examples is Raspberry Pi (RPi) cloud [1], which emulates a scale model of production cloud data centre. The initial version of RPi cloud has been constructed out of 56 RPi's, and subsequent version has more than 200 nodes interconnecting with leaf-spine topology and drawing power from USB hubs that are plugged to main power from a regular wall socket. With this scale, it is still possible to build the whole miniature cloud inside an office. However, when this number grows, it is obviously becoming more prohibitive to construct even a miniature scale model. To overcome the limitation in scalability, the need for building a simulator that allows studying the behaviour of low-power clouds is prominent.

This paper presents our initial attempt to model clouds comprising solely of such low power devices on top of the DISSECT-CF cloud computing simulator [2]. With our model, compute characteristics of raw and containerised low power devices, meaning that they run containers such as Docker [3] as a mean of virtualisation, are handled distinctively differently. The model also provides an easy transition from simulations designed for regular clouds, allowing cloud providers to experiment with low power devices while utilising previously validated simulation designs for their data centres.

We evaluated our extensions to the DISSECT-CF cloud simulator with a scenario utilising Apache Spark and HDFS executed in both a real life and a simulated system (both of them incorporate 12 RPi 2 model B nodes). Using the scenario, we show that the simulator is capable of modelling both the CPU and network utilisation. We have evaluated the models for precision and we have shown that CPU times are predicted with low error margin. Nevertheless, we have measured an error of 15% in the simulator's instantaneous CPU utilisation predictions. We believe this is reasonable because there are a number of reasons can contribute to unpredictable spikes in resource utilisation during real-life workloads. Hence, accurate CPU time prediction suggests correct estimates for average resource utilisation.

The rest of this paper is organised as follows. First, in Section II, we continue with the discussion of the state-of-the-art and its relation to our newly introduced models, extensions and techniques. Next, in Section III, we discuss the extensions applied to the DISSECT-CF cloud simulator aiming at large-scale experimentation with low power clouds. Later, Section IV discusses a case study using HDFS and Apache Spark and evaluates the previously overviewed extensions in contrast to a real life cloud. Finally, Section V concludes our work and provides an insight into our planned future works.

II. Related Work

In this section, we survey related literature in the areas of low-power compute cluster and cloud simulation.

A. Raspberry Pi and Raspberry Pi Compute Cluster

Since its launch in 2012, the RPi has quickly become one of the best-selling computers and has stimulated various interesting projects across both industry and academia that fully exploit the low-cost low power full feature computer [1], [4]–[8]. As of September 2016, the total number of units sold worldwide has passed 10 million [9].

Iridis-pi [7] and Glasgow Raspberry Pi Cloud [1] are among the first to use a large collection of RPi boards to construct clusters. Despite their similarity in hardware construction, their nature is distinctively different. Iridis-pi is an educational platform that can be used to inspire and enable students to understand and apply high-performance computing and data handling to tackle complex engineering and scientific challenges. On the contrary, the Glasgow RPi cloud is an educational and research platform which emphasises on development and understanding virtualisation and Cloud Computing technologies. Other similar RPi clusters include [8], [10], [11].

B. Cloud simulation

Limited accessibility to actual Cloud testbeds has made researchers to resort to simulated environments. Cloud Computing simulators include CloudSim [12], GreenCloud [13], iCanCloud [14] and MDCSim [15]. GreenCloud [13] is an extension of the NS2 network simulator for evaluation of energy-aware Cloud DCs. The main strength of GreenCloud is the detailed modelling of communication within a DC network. MDCSim [15] is a commercial discrete event simulator that models specific hardware characteristics of different DC components such as servers, communication links, and switches. On the contrary, iCanCloud [14] is a hypervisor simulator specifically aimed at simulating instance types provided by Amazon. Some tools that can simulate an entire Cloud stack include CloudSim [12] and DISSECT-CF [2]. However, CloudSim provides limited or no support for more realistic and complex applications composed of communicating tasks and workflows, enabling no or limited cross-layer interaction. In comparison, DISSECT-CF allows access to internal cloud information and accurately models power consumption of IaaS infrastructures.

One new direction for virtualisation is to use container-based virtualisation. This was first realised by ContainerCloudSim [16] which was built on top of CloudSim. To the best of our knowledge, this work is the first attempt to model and simulate an ARM-based low-power cloud with and without using containers virtualisation.

III. Modelling and simulation extension

In order to model ARM based low power systems, first, we started with the analysis of our previously collected dataset from a RPi cluster which runs a range of Apache Spark workloads [17] at Liverpool John Moores University. During these experiments, two Spark applications were run and thoroughly monitored on the cluster of 12 RPi mini computers. The applications were both run directly in both the cluster's native environment and in a container based virtualised environment. In other words, in the latter case each RPi has docker container that was set up to completely occupy its resources. We monitored and captured a range of data in order to identify the application behaviour on a low-power cluster including CPU, network and memory utilisation for before, during and after the particular workload was executed on the cluster. The CPU and memory utilization monitoring was done with the tool `vmstat`. While we used a custom script to collect network loads. In all cases, the utilisation was collected directly at the host level.

In this paper, we focus on the previously collected CPU and network related datasets. Our analysis on these data sets aims at finding the relationship between how containerised workloads would differ compared to directly utilising such low power devices. Figure 1 presents an example trace collected. Clearly, the data is noisy, especially considering the first and last few seconds where there was no activity related to our Apache Spark executions. Thus, we first analysed the noise and found that there is approximately 2.24% of CPU utilised for native RPis' use while dockerised RPis used 2.5%. This already shows that on average a dockerised RPi could be 6% slower in some scenarios.

Next, we focused our attention on the parts of the trace where the application was really running (i.e., all trace entries with over 10% CPU utilisation – securely over the 2.5% average noise –, or significant – 10k/sec or more – network traffic). This filtering allows us to more easily find correlations between data points as the varying length of the idle noise is mostly excluded from the results (the idle noise length varies because the metering data collection was not started and terminated at the exact same moments). A summarised CPU utilisation chart is shown in Figure 2. There we charted how the growing size of data (ranging between 1GiB-6GiB depicted as $1g, 4g$ and $6g$ in the figure) increased the average CPU utilisation over each machine in the cluster. Note, $pi1$ is the master machine, it only plays a control role in the application's execution.

After filtering all traces and calculating the cumulative CPU time and network usage values for both applications with all three problem sizes, we ended up with averages for both dockerised and direct application execution. In Figure 3, we show the relation between the average CPU utilisation of the direct execution and the average % difference of cumulative CPU time between the dockerised and

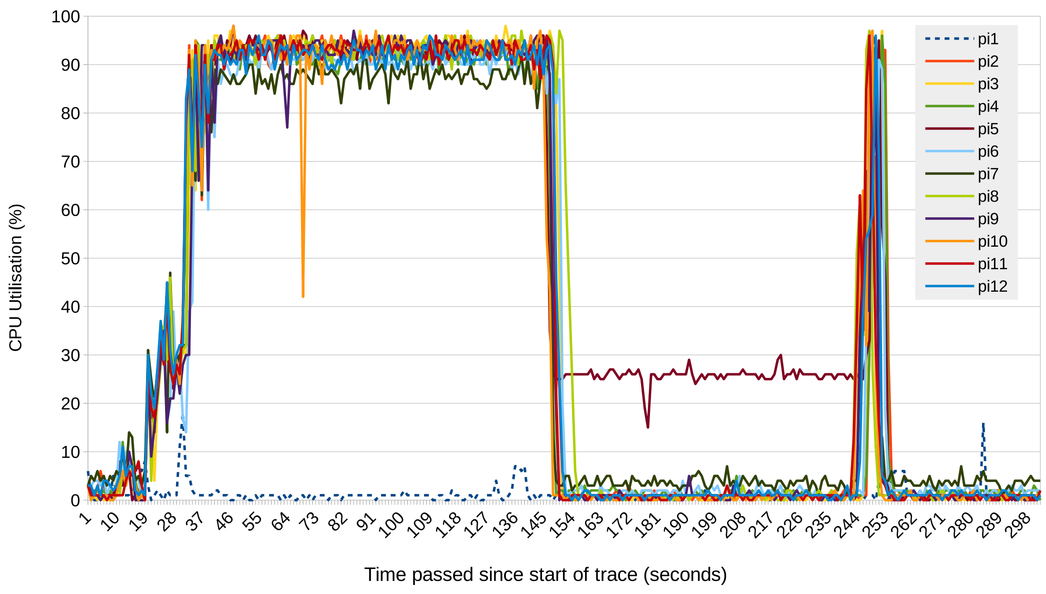

Figure 1: Example raw data collected regarding CPU utilisation.
Note: node $pi1$ (shown with the dashed line) acted as the master, while nodes $pi2 - 12$ were its workers

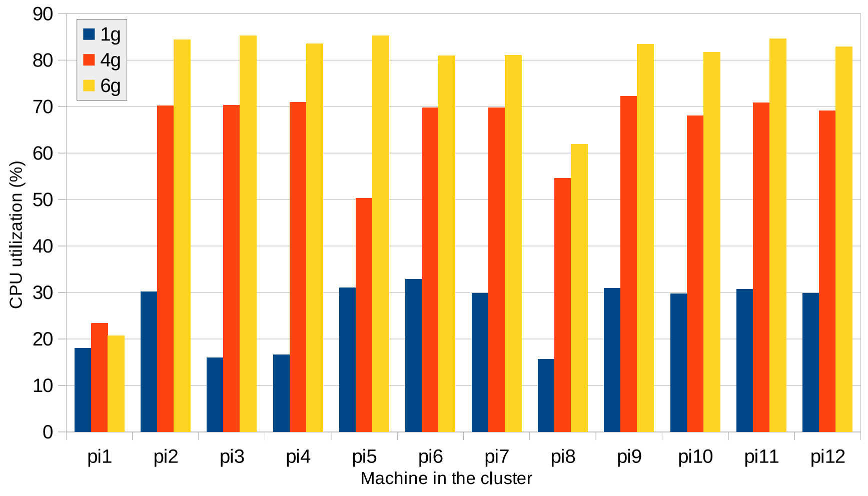

Figure 2: Example filtered data summaries regarding CPU utilisation

direct executions. As visible, the lower the CPU utilisation the less impact docker has on the average CPU time spent by each scenario. In contrast, the network utilisation figures are constant (apart from the expected variance because of the noise in the data). As expected the use of docker did not influence the amount of data transferred.

Based on the findings in the previously discussed figure, we conclude that a polynomial approximation of the relation between the spent CPU time could be made as follows:

$$t_{docker}^{cpu} := (\alpha U_{direct}^2 + \beta U_{direct} + \gamma) t_{direct}^{cpu} \qquad (1)$$

, where t_{docker}^{cpu} is the expected CPU utilisation time of the docker equivalent of an application that ran for t_{direct}^{cpu}. The utilisation time is dependent on the average CPU utilisation (U_{direct}) of the application during its entire execution. While $\alpha, \beta \quad and \quad \gamma$ are serving as constants in the model. With the current dataset, these constants are estimated as:

$$\begin{aligned} \alpha &:= 0.695 \\ \beta &:= 0.45 \\ \gamma &:= 0.895 \end{aligned} \qquad (2)$$

A. *Extending DISSECT-CF with the new model*

As larger scale experiments would be costly to implement in real life, we implemented the new model developed in the previous subsection. The implementation was done by

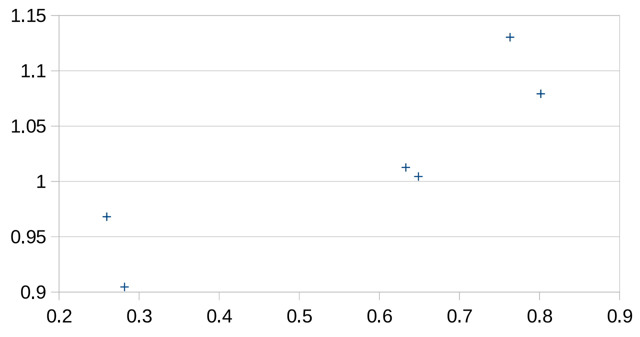

Figure 3: The relation of direct and dockerised CPU utilisation while running the same workload

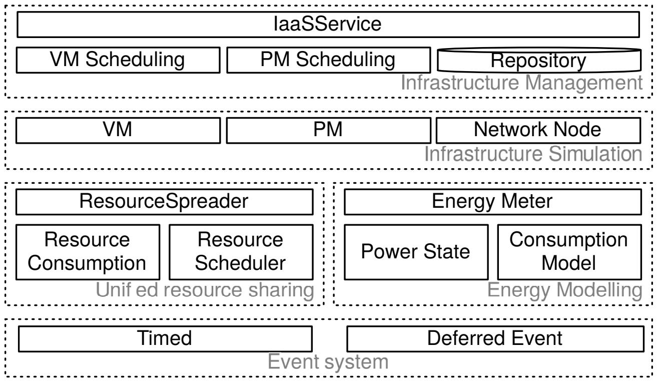

Figure 4: The architecture of the DISSECT-CF simulator

extending the DISSECT-CF cloud simulator [2] which was selected as the foundation as it is maintained by one of the authors and its performance for large-scale simulations is already promising.

DISSECT-CF is a compact, highly customizable open source[1] cloud simulator with the special focus on the internal organisation and behaviour of IaaS systems. Figure 4 presents its architecture. It groups the major components with dashed lines into subsystems. Each subsystem is implemented as independently from the others as possible. There are five major subsystems each responsible for a particular aspect of internal IaaS functionality: (i) event system – for a primary time reference; (ii) unified resource sharing – to resolve low-level resource bottleneck situations; (iii) energy modelling – for the analysis of energy-usage patterns of individual resources (e.g., network links, CPUs) or their aggregations; (iv) infrastructure simulation – to model physical and virtual machines as well as networked entities; and finally (v) infrastructure management – to provide a real life cloud like API and encapsulate cloud level scheduling.

Our extension concerns the infrastructure simulation layer, namely, we have extended the `VirtualMachine` class of the simulator with an `ARMDocker` subclass. This subclass takes over of the new compute task registration operation of the virtual machine and applies Eq. 1 to alter the user provided task size (i.e., the task size is adjusted to be larger if the expected CPU utilisation of the VM is high). As the equation depends on CPU utilisation, `ARMDocker` decomposes larger tasks into smaller pieces, allowing the CPU utilisation to influence each piece individually. The piece count is determined automatically depending on the number of tasks running parallel in the simulated docker container. If there are no parallel tasks, there is no decomposition. In contrast with multiple parallel tasks, the resolution of the pieces are set so even the shortest running task would have a chance to readjust its task sizes when necessary. This minimises the performance impact of the task decomposition (i.e., because of the need to simulate more compute tasks than the user originally asked about), but still allows the task size to be modelled after our observations.

[1]available from: https://github.com/kecskemeti/dissect-cf

IV. Validation and scalability experiments

To analyse our implementation we have turned our attention to the previously mentioned experiments and modelled them in the now extended simulator. In all experiments, we have a 12 node cluster of the following machine: RPi 2 Model B, which has a 900 MHz quad-core ARM Cortex-A7 CPU, 1 G RAM, and a 100 Mbps Ethernet connection. Each node was used either directly or indirectly through Docker containers fully occupying each machine. The operating system (OS) installed on the RPi's is Raspbian. Also on each node, we installed Spark 1.4.0 and Hadoop 2.6.4 for its HDFS. We configured node 1, i.e., $pi1$, as a master for Hadoop and Spark, and others, i.e., $pi2-12$, as workers. In both native and virtualised environments, we have run both *Wordcount* and *Sort* jobs on our low-power cluster with job sizes varying between 1GB, 4GB, and 6GB, representing small, medium and large job sizes respectively. The exact same experiment was replicated in the simulated infrastructure (the simulator was running on a machine with an Intel Core i7-4790 processor equipped with 16GBs of RAM and 500GBs of disk). In both the simulated and real life infrastructures, we monitored the execution time, the network throughput and CPU utilisation of each node. In the real life infrastrucutre, we measured the CPU utilisation with the `vmstat` tool, while the network throughput was collected via a custom script. In the simulated infrastructure we used DISSECT-CF's `MonitorConsumption` class and we attached it to the corresponding simulated PIs. Both the real life and the simulated monitors allowed CPU and network utilization figures to be collected on a per second basis.

Figure 5 presents the CPU utilisation behaviour of the simulated docker container (in this particular example we used the trace of the 6GB *Wordcount* application running on an average worker node). As its input to estimate the dockerised behaviour, the simulation was using the real life

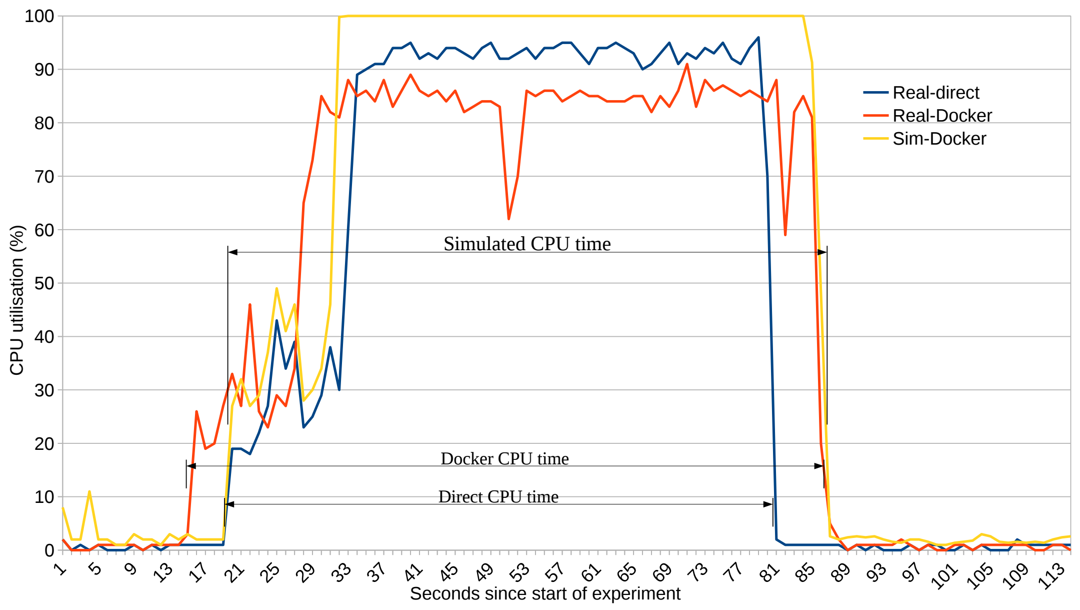

Figure 5: Example simulated CPU utilisation in contrast to real life data collected - using the traces of machine $pi12$

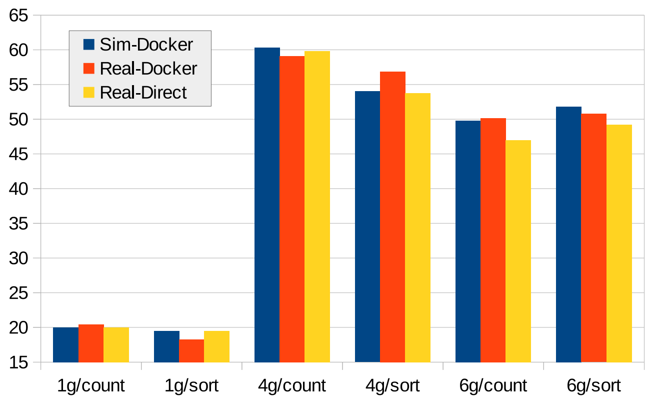

Figure 6: Average CPU utilisation prediction compared to real life results

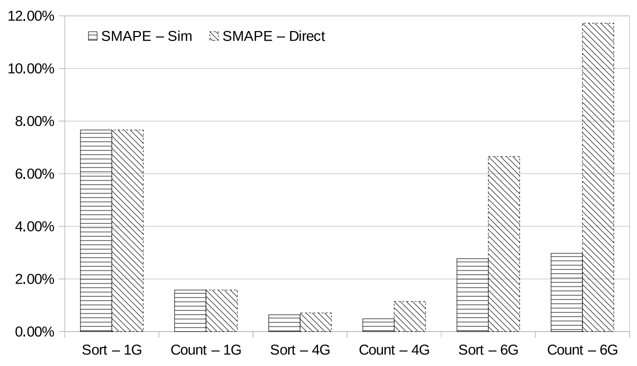

Figure 7: Symmetric Mean Absolute Percentage Error (SMAPE) of the total CPU time prediction

trace shown as "$real - direct$" in the figure. It can be observed that the simulation (see "$sim-docker$" in the figure) closely matches (within the relative error margin of 3 %) the runtime of the real-life docker run (see "$real - docker$" in the figure). On the other hand, the model leaves room for improvement in the predicted CPU utilisation of the system while using docker (peak utilisation could be over 15% more in the predicted case than in real life). The figure also depicts how the runtimes (i.e., $CPUtime$) were calculated for each case.

While the individual execution traces could have significant errors for prolonged periods, the model offers better results when it is compared in terms of predicted average CPU utilisation. This is exemplified in Figure 6, where we show how the simulation compares to the real-life measurements collected and averaged from all the traces of each individual application (i.e., the average CPU utilisation of all cluster members was averaged while it was running a particular application). The mean absolute error (MAE) of the simulator's average execution time prediction compared to the real life docker runs was 1.58%.

In contrast, in Figure 7, we present how the runtimes of the various Spark processes on each RPi node were predicted by the simulator. Here we measured the CPU time for all trace files (i.e., 72 traces for each experiment: docker, direct and simulated). The measurements were taken on the same way we have already shown in Figure 5. Then we calculated the SMAPE values to estimate the quality of the prediction. As the prediction is using the direct traces as its inputs, we have charted the figure in comparison to them. Thus we not only show how well the simulator predicts but how

Table I: Simulation runtimes for the various scenarios

App	Size	Sim Time (ms)
Count	1G	153
Count	6G	223
Sort	1G	201
Sort	6G	213

well does it perform compared to not adjusting the direct traces (i.e., reusing them as if they were the predictions of the simulator). To conclude, the larger the CPU load (and the workload size) the more likely the simulator achieves significantly better SMAPE values. The high SMAPE values on the smallest workloads are caused by the relatively high noise on the collected instantaneous CPU utilisation values. In our future works, we will revise the measurement methodology so it could offer less noisy CPU utilisation values for these cases allowing the evaluation of the model in such underutilisation related scenarios as well.

Finally, we present the real-life runtimes (i.e., acquired while running on the simulator on the Core i7 machine discussed above) of the simulated executions in Table I. Although the problem size (in terms of total CPU time spent across all RPis) for the RPis has increased to $9.16\times$ between the scenarios Count 1G and Sort 6G, the execution time did not significantly grow during their simulation. Thus we expect the simulator to be capable of handling significantly larger scaled systems and still provide sufficiently accurate results.

V. Conclusions

Nowadays, more and more energy conscious data centres are built. The green computing efforts bring alternative, low-power CPUs to the data centres. To exploit the full potential of these low power CPUs, simulators could be utilised to identify new use cases and adopt these systems to better match their expected use. Low power data-centres and virtualisation is a neglected combination in recent simulators. Thus in this paper, we have presented a new model for docker containers in low power environments. Next, we have implemented the model in the cloud computing simulator called DISSECT-CF. Then, we have evaluated the implemented extension's performance and accuracy by comparing the simulated results to real life traces collected from a 12-node RPi 2 cluster. Our results have shown an accuracy with low MAE (Mean Absolute Error).

In our future works, we are aiming at improving the precision of the simulation on smaller granularity CPU utilisation predictions. Also, we will analyse the possible scale of the simulated RPi-based cloud and its limiting factors. Next, as several other devices are playing a key role in low-power data-centres, we plan to extend our models towards other low-power devices such as the Cavium Thunder X2[2]. Finally, we plan to introduce specialised application models for frequently used scientific and commercial applications in low power environments.

Acknowledgements

This work is partially supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/P004407/1 and EP/P004024/1.

Software availability

This paper described the behaviour and features of DISSECT-CF version 0.9.8. Its source code is open and available (under the licensing terms of the GNU LGPL 3) at the following website:
`https://github.com/kecskemeti/dissect-cf`

References

[1] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The Glasgow Raspberry Pi cloud: A scale model for cloud computing infrastructures," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. IEEE, 2013, pp. 108–112.

[2] G. Kecskemeti, "DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds," *Simulation Modelling Practice and Theory*, vol. 58P2, pp. 188–218, November 2015.

[3] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[4] S. Jain, A. Vaibhav, and L. Goyal, "Raspberry Pi based interactive home automation system through e-mail," in *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on*, Feb 2014, pp. 277–280.

[5] V. Vujovic and M. Maksimovic, "Raspberry Pi as a Wireless Sensor node: Performances and constraints," in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. IEEE, 2014, pp. 1013–1018.

[6] S. L. Fernandes and J. G. Bala, "Low Power Affordable and Efficient Face Detection in the Presence of Various Noises and Blurring Effects on a Single-Board Computer," in *Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1*. Springer, 2015, pp. 119–127.

[7] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'brien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2014.

[2]http://www.cavium.com/ThunderX2_ARM_Processors.html

[8] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily *et al.*, "Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 2. IEEE, 2013, pp. 170–175.

[9] Raspberry Pi Foundation, "Ten millionth Raspberry Pi, and a new kit," https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/, 2016, accessed on Sep 2016.

[10] M. F. Cloutier, C. Paradis, and V. M. Weaver, "Design and analysis of a 32-bit embedded high-performance cluster optimized for energy and performance," in *Hardware-Software Co-Design for High Performance Computing (Co-HPC), 2014*. IEEE, 2014, pp. 1–8.

[11] A. Anwar, K. Krish, and A. R. Butt, "On the use of microservers in supporting hadoop applications," in *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*. IEEE, 2014, pp. 66–74.

[12] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. e. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[13] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. Khan, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec., pp. 1–5.

[14] A. Nuñez, J. Vžquez Poletti, A. Caminero, J. Carretero, and I. Llorente, "Design of a new cloud computing simulation platform," in *Computational Science and Its Applications - ICCSA 2011*, ser. Lecture Notes in Computer Science, 2011, vol. 6784, pp. 582–593.

[15] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. Das, "MDCSim: A multi-tier data center simulation, platform," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 31 2009-Sept. 4, pp. 1–9.

[16] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers," *Software: Practice and Experience*, vol. to appear, 2016, *doi:*10.1002/spe.2422.

[17] W. Hajji and F. P. Tso, "Understanding the performance of low power Raspberry Pi Cloud for big data," *Electronics*, vol. 5, no. 2, p. 29, 2016.