SeDuCe: a Testbed for Research on Thermal and Power Management in Datacenters

Jonathan Pastor
IMT Atlantique - Nantes
jonathan.pastor@imt-atlantique.fr

Jean Marc Menaud
IMT Atlantique - Nantes
jean-marc.menaud@imt-atlantique.fr

***Abstract*—With the advent of Cloud Computing, the size of datacenters is ever increasing and the management of servers and their power consumption and heat production have become challenges. The management of the heat produced by servers has been experimentally less explored than the management of their power consumption. It can be partly explained by the lack of a public testbed that provides reliable access to both thermal and power metrics of server rooms. In this article, we propose SeDuCe, a testbed that targets research on energy and thermal management of servers, by providing public access to precise data about the power consumption and the thermal dissipation of 48 servers integrated in Grid'5000 as the new *ecotype* cluster. We present the chosen software and hardware architecture for the first version of the SeDuCe testbed, and propose some improvements that will increase its relevance.**

Keywords–Datacenters; Scientific testbed; Thermal management; Power management; Green computing.

I. INTRODUCTION

The advent of web sites with a global audience and the democratization of Cloud Computing have led to the construction of datacenters all over the world. Datacenters are facilities that concentrate from a few servers up to hundreds of thousands of servers hosted in rooms specially designed to provide energy and cooling for the servers. These facilities are widely used for applications from web services to *High Perfomance Computing* (HPC).

In recent years, the size of datacenters is ever increasing, which leads to new challenges such as designing fault tolerant software to manage at large scale the servers and energy management of server rooms. On the latter challenge, many research efforts have been conducted [1] [2], most of them focusing on the implementation of on demand power management systems, such as Dynamic voltage scaling (DVFS) [3] [4] and vary-on vary-off (VOVO) [5] [6]. Some work has been made to extend existing scientific testbeds with power monitoring of experiments: for example Kwapi [7] enables researchers to track the power consumption of their experiments conducted on Grid'5000.

On the other hand, the thermal management of servers has been less explored, a large part of the existing work considering only simulations [8]. This can be explained, partly, by the difficulty of conducting experiments involving thermal monitoring of servers: to ensure that the data recorded experimentally is valid, experimentions must be conducted on a testbed that contains many temperature sensors, not only positioned on cooling systems, but also at the front and the back of each server of the racks.

In addition, such a testbed must enable reproducible experimentations, by providing its users with a full control on experimental conditions like setting the temperature of the environment of their experiments and by exposing its data in a non misleading way, via a well documented Application Programming Interface (API).

Finally, as power management and temperature management of servers are related problems [9], there is a need for a testbed that enables users to access to both thermal and power data of servers.

As far as we know, there is no public testbed that enables researchers to work on both energy and thermal aspects of servers functionning. The objective of the SeDuCe - Sustainable Data Centers - project is to propose such a testbed: the SeDuCe testbed enables its users to use, in the context of the new *ecotype* cluster of the Grid'5000 infrastructure [10], 48 servers located in 5 airtight racks with a dedicated Central Cooling System (CCS) positioned inside one of the rack. In parallel of conducting the experiment by leveraging the tools provided by Grid'5000, users can get access to thermal and power data of the testbed via a web portal and a user-friendly API. The stability of experimental conditions is guaranteed by hosting the testbed in a dedicated room equipped with a secondary cooling system (SCS) that enables a precise thermoregulation of the environment outside the cluster. As resources of the testbed are made publicly available via the Grid'5000 infrastructure, all its users are able to perform reproducible research on thermal and power management of servers. The rest of the paper is structured as follows. In Section II, we describe the SeDuCe testbed, in Section III an experimental validation of the testbed is conducted. In Section IV we detail the future work on SeDuCe. Finally, we conclude in Section V.

II. TESTBED DESIGN

A. Ecotype: a Grid'5000 cluster dedicated to the study of power and thermal management of servers

In [11], we introduced our initial work on the *ecotype* cluster: we have builded the "ecotype" cluster, which contains 48 servers, and is integrated in the Grid'5000 infrastructure: any Grid'5000 user can reserve servers of the *ecotype* cluster and conduct experiments on them by using the usual Grid'5000 tools. The testbed is designed for research related to power and thermal management in datacenters: during an experiment, a user can access in real time to information regarding the temperature of the servers involved in its experiment, and get the power consumption of any parts of the testbed (servers, switches, cooling systems, etc.), or control some parameters of

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-667-5

the testbed, such as setting temperature targets for the cooling systems of the cluster.

Servers of the *ecotype* cluster are based on DELL PowerEdge R630 and contains a pair of Intel Xeon E5-2630L v4 CPUs (10 cores, 20 threads per CPU), 128GB of RAM, and 400GB Solid State Disk (SSD). The CPUs have been designed to have a lower power consumption than other CPUs of the XEON 26XX serie, with a Thermal Design Power (TDP) of 55W. Each server is connected via two 10GBe links to the Grid'5000 production network, and via a single 1GBe link to the Grid'5000 management network. For instance, the Grid'5000 production network is used for transferring the disk images required to deploy an experiment or to support communications between experimental components, while the management network is mainly used by the Grid'5000 backend to communicate with management cards of servers to turn them on and off. Additionally, each server is certified to work in hot environments where temperature can be up to 35°C. These hardware specifications will enable users to perform experiments at different levels of temperature.

The cluster is composed of 5 air-tights racks (Z1, Z2, Z3, Z4, Z5) based on the *Schneider Electric IN-ROW* model. These air-tights racks are equipped with Plexiglas doors, and create a separation between the air inside the racks and the air from outside the racks. As shown on Figure 1, one rack (Z3) is used for the cooling the cluster by hosting a dedicated Central Cooling System (CCS), while remaining racks are computing racks and are dedicated to hosting servers. The racks are connected and form two alleys: a cold alley at the front of servers and a hot alley at their back.

As depicted by Figure 1, each computing rack hosts 12 servers, and is organized following two layouts of server positions: one layout where servers are organised in a concentrated way with no vertical space between servers (Z1 and Z2), and a second layout where servers are spaced at 1U intervals (Z4 and Z5).

We have deliberately chosen to use these two layouts: they will enable users to study the impact of the server density over the temperature and the power consumption of servers.

In addition to the servers, the cluster also contains three network switches that are in charge of connecting servers to the production network and the management network of the Grid'5000 infrastructure. Three racks (Z2, Z4, Z5) are hosting each one a network switch.

The 5 racks of the cluster are based on *Schneider Electric IN-ROW* racks. This rack model creates an inside airtight environment for servers, and guarantees that the environment outside the cluster has a limited impact on temperatures inside the racks. The temperature inside the cluster is regulated by the CCS, which is connected to a dedicated management network and implements a service that enables to remote control the cooling and to access its operating data with the Simple Network Management Protocol (SNMP) protocol. The CCS has several temperature sensors located at different parts of the racks, which are in charge of checking that the temperature inside racks is under a specified temperature threshold. It is possible to change the temperature that the CCS have to maintain inside the racks (*Cooling Temperature Target* parameter), and also change the temperature of the air injected by the CCS in the cold aisle (*Air supply Temperature Target* parameter). This will, in addition to the fact that servers have been designed to work in hot environments, enable users perform their experiments at several levels of temperature.

Regarding the temperature outside the cluster, it is regulated by the SCS which is mounted from the ceiling of the server room: the SCS is in charge of maintaining a constant temperature in the server room, and thus it prevents any event outside the racks to disturb the experiments that are conducted on the SeDuCe testbed.

Finally, we have installed several "Airflow management panels" between each pair of servers: they improve the cooling efficiency by preventing the mixing of cold air and hot air inside the racks.

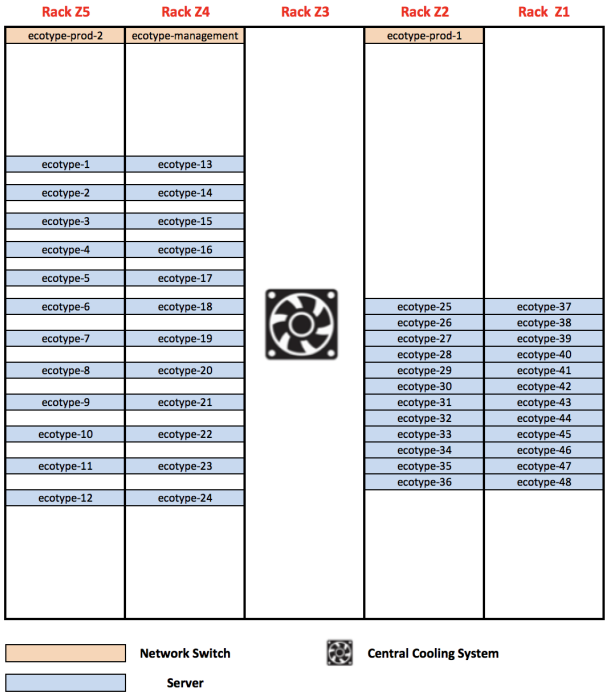

Figure 1. Layout of the ecotype cluster (front view)

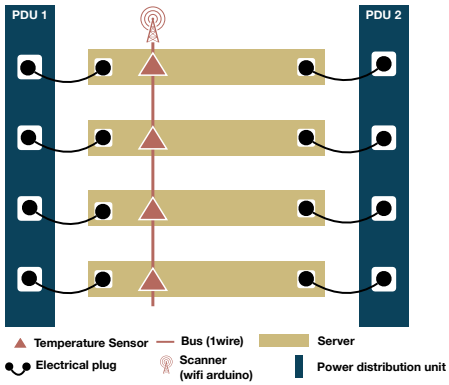

Figure 2. Back view of a server rack

B. Power Monitoring

The power consumption of each element composing the cluster (servers, network switches, cooling fans, condensators, etc.) is monitored and stored in a database, at a frequency of one hertz (one record per second).

Electrical plugs of servers and network switches are connected to Power Distribution Units (PDUs), which are in

charge of ensuring that servers and network switches can meet their power needs. Each computing racks contains two PDUs, and each server of a computing rack has two electrical plugs. As depicted in Figure 2, the two electrical plugs of a server are connected to two different PDUs, which enables servers to implement electrical redundancy. In turn, the PDUs share power consumption of servers and network switches via a dedicated service network: the power consumption of each power plug can be fetched by issuing an SNMP request to the PDU to which it is connected. In turn, the PDU provides the power consumption of each of its outlets.

The energy consumption of the CCS can similarly be fetched via SNMP requests: the CCS implements a SNMP service which is able to provide the overall power consumption and the power consumption of each of its internal part such as the condensator or the fans. On the other hand, the SCS does not implement any built-in networking access, and thus cannot share its metrics with any component over a network. To solve this problem, we instrumented several parts of the SCS by using a *Fluksometer* [12]: a *Fluksometer* is a connected device that can monitor several electrical metrics (power consumption, voltage, amperage, etc.) and expose their values over a network via a web-service at a frequency of one hertz.

Finally, we have added an additional system that tracks overall power consumption of servers, switches and the CCS. This additional system is based on the *Socomec G50 metering board* [13], and enables to check the soundness of the aforementioned source of power consumption. These additional metrics are fetched by using the modbus protocol.

C. Temperature Monitoring

To track the thermal behavior of the *ecotype* cluster, each server is monitored by a pair of temperature sensors: one sensor is positioned at the front of the server (in the cold aisle) and another sensor is positioned at the back of the server (in the hot aisle).

As depicted by Figure 2, each temperature sensor is part of a bus (based on the *1wire* protocol) connected to a *Scanner* (based on an Arduino that implements wifi communication) in charge of gathering data produced by temperature sensors of the bus. As the front and the back of each server is monitored by temperature sensors, each computing rack has in total two *Scanners* and two buses: a front bus for monitoring the cold aisle and a back bus dedicated to the hot aisle. *Scanners* fetch temperatures from their sensors at a frequency of one reading per sensor every second.

Temperature sensors are based on the DS18B20 sensor produced by "Maxim Integrated" [14] that costs approximately 3$ per sensor. According to the specifications provided by the constructor, the DS18B20 is able to provide a temperature reading every 750ms with a precision of 0.5°C between -10 °C and 85 °C.

The choice of the DS18B20 has been motivated by the fact that the DS18B20 sensor is able to work as part of an *1wire* bus. In the context of the SeDuCe infrastructure, 12 DS18B20 sensors are connected together to form an *1wire* bus, and a *Scanner*, based on an *nodeMCU* arduino with built-in wifi capabilities, fetches periodically their temperature readings. The current version of the firmware used by *Scanners* scans an 1wire bus every second, and then pushes temperature data to a *Temperature Registerer* service, as illustrated in Figure 3.

We also developed a contextualisation tool to generate firmwares for the *Scanners*. It leverages the PlatformIO framework [15] to program a *Scanner* that pushes data to a web-service. Using this contextualisation tool is simple: a developer needs to define a program template in a language close to C language and marks some parts of code with special tags to indicate that these parts need to be contextualized with additional information, such as initializing a variable with the ID of a *Scanner* device or with the address of a remote web-service (such as the one that will receive temperature records). The contextualisation tool takes this program and a context as input parameters, analyses the template program, and completes parts that requires contextualisation with information provided in the context, which results in valid C language source file. Then, the firmware is compiled and automatically uploaded to *Scanners* via their serial ports. By leveraging this contextualisation tool, we can remotely configure *Scanners* and update their firmware "on the fly".

D. SeDuCe portal

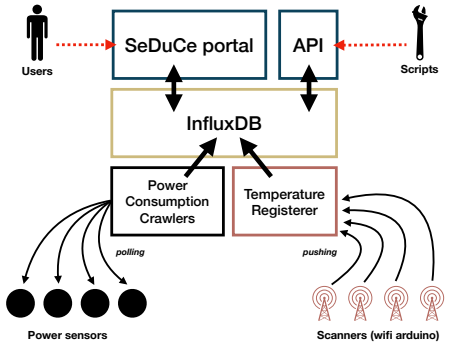

Figure 3. Architecture of the SeDuCe portal

To help users to easily access power and thermal metrics generated by the SeDuCe testbed, we developed a platform that exposes publicly two components: a web portal [16] and a documented API [17].

As illustrated by Figure 3, the web portal and the API fetch data from a Time Series Database (TSDB) based on *InfluxDB* [18]. *InfluxDB* enables to store a large quantity of immutable time series data in a scalable way. In the background, *InfluxDB* creates aggregates of data by grouping periodically data from a same series. These aggregated sets of data enable the web portal to promptly load data used for visualization.

Two kind of components are in charge of inserting data in the database : the *Power consumption crawlers* and the *Temperature Registerer*. *Power consumption crawlers* are programs that are in charge of polling data from PDUs, Socomecs, Flukso, the CCS and the SCS. In turn, this data is inserted in the database. On the other hand, the *Temperature Registerer* is a web service that receives temperature data pushed from *nodeMCU* arduino devices, and inserts it in the database.

The web portal and the API are both written in Python and leverage the "Flask" micro web framework [19]. The API component makes an extensive use of the Swagger framework [20] which automatises the generation of complete REST web services and their documentations from a single description file (written in JSON or YAML). This choice has enabled us to focus on the definition and the implementation of the API, by reducing the quantity of required boilerplate code.

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-667-5

All the components depicted in Figure 3 are implemented as micro-services. Our system is able to register 200 metrics per seconds with minimal hardware requirements (it is currently hosted on a single computer). In the case we add more sensors to our testbed, it is likely that the existing components would be sufficient. In the case that one of the component would not be able to cope with the additional workload, it would be easy to setup an high availability approach by using a load balancer such as *NGINX* that would forward requests to a pool of instances of the component.

III. EXPERIMENTATION

To illustrate the potential of the SeDuCe platform, we conducted two experiments that mix energetic and thermal monitoring. The objective of the first experiment was to reproduce a known scientific result by using both temperature and power data from the SeDuCe testbed, while the objective of the second experiment was to study the impact of parameters of the CCS over the overall power consumption.

A. First experiment: studying the impact of idle servers on cooling

The goal of this first experiment is to verify that the data produced by the SeDuCe testbed is reliable, by designing an experiment that will use both the thermal and the power data produced by the testbed. These data would be used to reproduce a scientifically validated observation, such as the impact of idle servers on the power consumption and the temperature of a server room. Such experiment has been conducted in the past [9], however as far as we know there is no public testbed that would enable researchers to reproduce this result: by reproducing this result on the SeDuCe testbed, we think that it would demonstrate the soundness of our approach and the usefulness of our testbed.

1) Description of the experiment: To illustrate the scientific relevance of our testbed, we wanted to reproduce the observations made by third party publication [9].

In [9], authors have highlighted an interesting fact: in a datacenter idle servers (i.e. servers that are turned on while not being used to execute any workload) have a significant impact on power consumption and heat production. We decided to try to reproduce this observation.

For this experiment, servers of the *ecotype* cluster are divided in three sets of servers:

- *Active servers*: servers with an even number (ecotype-2, ecotype-4, ..., ecotype-48) were executing a benchmark that generates a CPU intensive workload.
- *Idle servers*: a defined quantity (0, 6, 12, 18, 24 servers) of servers with an odd number (ecotype-1, ecotype-3, ..., ecotype-47) was remaining idle.
- *Turned-off servers*: remaining servers were electrically turned off.

and during one hour we recorded the power consumption of the CCS and the average temperature in the hot aisle of the *ecotype* cluster. The CPU intensive workload was based on the "sysbench" tool : the goal was to stress CPUs of each servers, resulting in an important power consumption and a bigger dissipation of heat. To guarantee the statistical significance of the measurements, each experimental configuration was repeated 5 times, leading to a total number of 25 experiments.

We executed two sets of experiments: one with the SCS turned-on (Figure 4) and the other while the SCS was turned off (Figure 5). The objective of turning off the SCS was to identify the impact of the SCS over the CCS.

2) Results: Figure 4 plots the cumulated power consumption of the CCS and the average temperature in the hot aisle of the cluster with the SCS enabled.

First, it is noticeable that as the number of idle nodes increases, both the energy consumed by the SCS and the temperature in the hot aisle of the rack increase. This can be explained by the fact that an idle node consumes some energy and produces some heat, which increases the workload of the CCS.

The second element highlighted by Figure 4 is that the impact of idle nodes is not linear: the red line representing the CCS consumption follows an exponential pattern and the blue line representing the average temperature in the hot aisle follows a sigmoid pattern. The exponential pattern of the power consumption of the CCS can be explained by the fact that the heat produced by a given server has an impact on the temperature of surrounding servers, thus creating hot spots in the the cluster. The CCS has it own monitoring of the temperature of servers thanks to many temperature sensors located in the racks (these sensors are independent to the ones we installed on the buses). These hot spots are detected by some of the many temperature sensors of the CCS and this leads to an activation of the CCS to reduce the temperature of hot spots to the temperature target configured in the CCS. The more the number of idle servers increases, the more the CCS must be activated to maintain its temperature target. On the other hand, the sigmoid pattern of the average temperature in the hot aisle is explained by the fact that when the number of idle servers is higher than 12, the functionning of the CCS is more intensive and thus the additional production of heat by server is absorbed by the CCS, and thus the average is growing at a slower rate.

Figure 5 plots the cumulated power consumption of the CCS and the average temperature in the hot aisle of the cluster while the SCS is disabled. This figure highlights that the power consumption of the CCS is lower when the SCS is disabled. This can be explained by the fact that the SCS was configured to maintain a temperature of 19 °C in the outside room, which is close to the maximum temperatures in the cold aisle: as the SCS does not cool down enough the outside air, by means of thermal conduction, it warms the temperature inside the racks. As a consequence, it increases the needs in term of cooling inside the cluster, leading to an higher power consumption of the CCS.

This experimental campaign has shown that idle servers have an important impact on the power consumption of cooling systems and overall racks temperature, thus it confirms the observation made in this publication [9].

B. Second experiment: Finding the optimum cooling parameters for the CCS

1) Description of the experiment: The CCS is based on the *Schneider Electric IN-ROW* cooling system. The functioning of this cooling system can be customised by changing several parameters, such as:

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-667-5

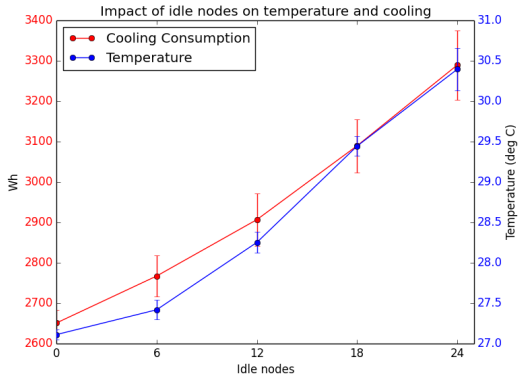

Figure 4. Central cooling consumption and average temperature in the hot

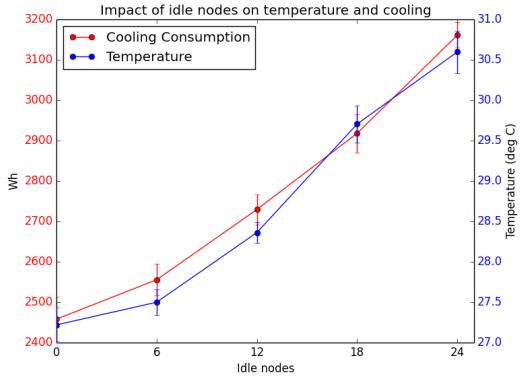

Figure 5. Central cooling consumption and average temperature in the hot aisle (SCS disabled)

- *Cooling Temperature Target*: corresponds to the temperature that the CCS tries to maintain inside the racks.
- *Air supply Temperature Target*: corresponds to the temperature of the air supplied by the CCS to the cold aisle.

In this second experiment, the impact of these two parameters is studied, in order to understand which combination of these two variables would lead to an optimal power consumption of the CCS. The experimental protocol of this second experiment is very similar to the one described in Section III-A1 as nodes where divided in two sets of servers:

- *Active servers*: these servers were executing a benchmark that generates a CPU intensive workload.
- *Turned-off servers*: these servers were electrically turned off.

We defined two configurations: an "heavy-load" configuration with 48 active servers and 0 turned-off servers, and "medium-load" configuration configuration with 24 active servers (servers with an even number) and 24 turned-off servers (servers with an odd number).

We tried several combinations of the *Cooling Temperature Target* and *Air supply Temperature Target*: the CCS's configuration was set to use these values. Each of combination has been tried in both the "heavy-load" and the "medium-load" configuration: we let the cluster in the chosen configuration for one hour, and then measured the overall power consumption of the CCS. Each experiment was repeated at least 4 times. Between each run of an experiment, we have implemented a pause step, where the CCS was configured back to its default settings (*Cooling Temperature Target* set to 23°C and *Air supply Temperature Target* set to 20°C), and all the servers were turned off until the overall temperature inside racks was under 26°C. Once the racks were cool enough, the CCS was programatically (via an SNMP API) setup to use the two cooling parameters required by the next experiment, and then the next experience would start.

2) Results: Figure 6 plots the cumulated power consumption of the CCS depending on the cooling parameters introduced in Section III-B1 in the case of a "medium-load" configuration. First, it is noticeable that as the *Air supply Temperature Target* increases, the overall power consumption of the CCS decreases: when it is set to 20°C, all the cumulated power consumption are over 2500 Wh, while they are lower than 2000 Wh when the air supply is set to be at 26°C, which corresponds to a decrease of 20%. Second, an high value for *Cooling Temperature Target* parameter seems also to have an impact on the overall power consumption of the CCS, as setting the *Cooling Temperature Target* to 31 °C leads to a power consumption that is more than 130 Wh over the consumption with lower *Cooling Temperature Target* values. We explain this observation by the fact that an important *Cooling Temperature Target* value creates hot spots in the hot aisle, which are detected by some of the many CCS's sensors and leads to an additional functioning of the CCS.

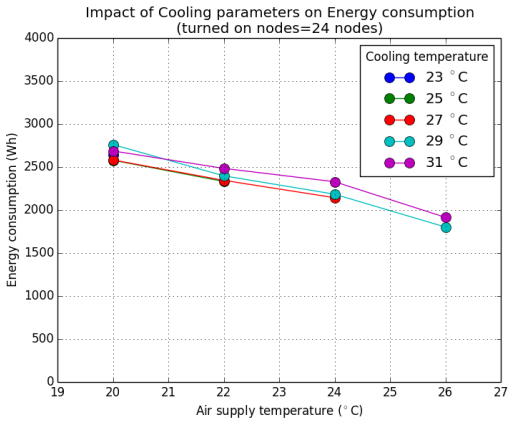

Figure 6. Central cooling consumption depending of temperature targets ("medium-load" configuration)

On the other hand, in the "heavy-load" configuration the results of the different strategies are closer. As illustrated in Figure 7, changing the values of *Air supply Temperature Target* and *Cooling Temperature Target* parameters does not seem to have an impact over the consumption of the CCS. We explain this observation by the fact that 48 active servers are an important source of heat, which requires the CCS to work continuously to maintain the target temperature, whichever cooling strategy has been used.

This second experimental campaign shows the experimental potential enabled by the the SeDuCe testbed: users can get access to thermal and power data produced during the functioning of the testbed, and they can also parameterize the configuration of the CCS. Thus, the SeDuCe testbed can help reasearchers working on the cooling of datacenters to design experiments with a fine-grained-control on experimental conditions.

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-667-5

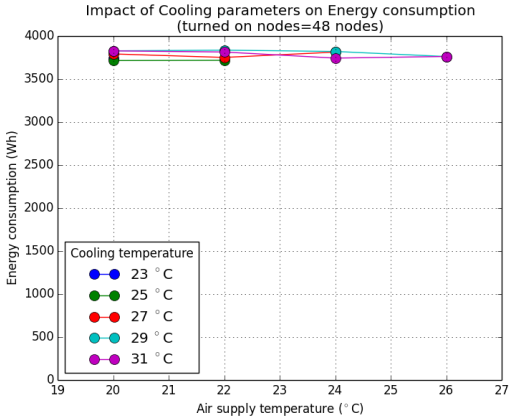

Figure 7. Central cooling consumption depending of temperature targets ("heavy-load" configuration)

IV. Future work

In this article, a first version of the SeDuCe testbed has been presented. In its current state, the SeDuCe testbed provides its users with an access to the thermal and power data generated during its functioning, and users can use this data in their work, such as studying the thermal profile of a software, or building energy efficient placement strategies. We currently see two areas of progress : one is related adding new energetic capabilities to the testbed, while the second is related to improving the precision of our thermal measurements.

Regarding the addition of new energetic capabilities to the testbed, during the next phase of building of the SeDuCe testbed (summer 2018), several solar panels will be installed on the roof of one building at *IMT Atlantique*. The energy produced by the solar panels will be used, either for supplying electricity to the testbed or for storage in batteries.Integrating these sources of renewable energy in the existing testbed will be a challenge, as we would like our users to be able to control, via an API, how the energy produced by solar panels is used, and to dynamically decide what quantity will be used by the testbed and what quantity will be stored in batteries. We think that these new capabilities will enable researchers to have access to all the elements required to experimentally study energy efficient placement strategies in datacenters.

In [11], we highlighted the fact that the accuracy of DS18B20 was not satisfactory enough in the cold aisle. Regarding the objective of improving the precision of the thermal measurements, we are currently investigating the addition of new temperature sensors based on the thermocouple approach. We have designed a prototype of an electronic card that embeds several thermocouples lines. We are working with some electronic assemblers to manufacture the cards and plan to install few of them in the cluster within September 2018.

V. Conclusion

In this article we have presented our initial work on building the SeDuCe testbed, a scientific testbed that targets research related to power and thermal management in datacenters, and which is integrated in Grid'5000 infrastructure as the new "ecotype" cluster. We have described the architecture of the testbed, which is built on buses of sensors, storage of power and thermal metrics in a time series oriented database (InfluxDB) and an user friendly web portal and a documented API. We have also detailed the components used in this first version of the SeDuCe testbed. We have illustrated the relevance of the SeDuCe testbed by performing two experimental campaigns that use the data produced by the testbed: the first experiment consisted in reproducing an existing scientific result, while the purpose of the second experiment was to illustrate the fine-grained-control that users of the SeDuCe testbed have over experimental conditions. Future work will focus on two areas: adding renewable energy capabilities to the SeDuCe testbed, and improving the precision of temperature sensors.

References

[1] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. ACM, 2009, pp. 41–50.

[2] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing. IEEE Computer Society, 2011, pp. 26–33.

[3] G. Von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in dvfs-enabled clusters," in Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on. IEEE, 2009, pp. 1–10.

[4] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters," Future Generation Computer Systems, vol. 37, 2014, pp. 141–147.

[5] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Dynamic cluster reconfiguration for power and performance," in Compilers and operating systems for low power. Springer, 2003, pp. 75–93.

[6] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," ACM SIGOPS operating systems review, vol. 35, no. 5, 2001, pp. 103–116.

[7] F. Clouet et al., "A unified monitoring framework for energy consumption and network traffic," in TRIDENTCOM-International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, 2015, p. 10.

[8] H. Sun, P. Stolf, and J.-M. Pierson, "Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters," Future Generation Computer Systems, vol. 71, 2017, pp. 157–170.

[9] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers." in USENIX annual technical conference, General Track, 2005, pp. 61–75.

[10] R. Bolze et al., "Grid'5000: A large scale and highly reconfigurable experimental grid testbed," The International Journal of High Performance Computing Applications, vol. 20, no. 4, 2006, pp. 481–494.

[11] J. Pastor and J.-M. Menaud, "Seduce: Toward a testbed for research on thermal and power management in datacenters," in E2DC, vol. 18, 2018, pp. 1–7.

[12] Flukso website. [Online]. Available: https://www.flukso.net/about

[13] Socomec g50 website. [Online]. Available: https://www.socomec.fr/gamme-interfaces-communication_fr.html?product=/diris-g_fr.html

[14] Technical documentation of the ds18b20 sensor. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[15] Website of the platformio project. [Online]. Available: https://platformio.org/

[16] Website of the seduce portal. [Online]. Available: https://seduce.fr

[17] Documented seduce api. [Online]. Available: https://api.seduce.fr/apidocs

[18] I. InfluxData. Website of the influxdb project. [Online]. Available: https://www.influxdata.com/

[19] A. Ronacher. Website of the flask project. [Online]. Available: http://flask.pocoo.org/

[20] Website of the swagger project. [Online]. Available: https://swagger.io/

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-667-5