ENGINEERING MATHEMATICS
AND COMPUTING LAB

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Development and implementation of a temperature monitoring system for HPC systems

Martin Baumann, Fabian Gebhart, Oliver Mattes, Sotirios Nikas, Vincent Heuveline

Preprint No. 2017-07

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
Preprint No. 2017-07

Affiliation of the Authors

Martin Baumann[a], Fabian Gebhart[a], Oliver Mattes[a], Sotirios Nikas[a], Vincent Heuveline[a]

[a] *Heidelberg University, University Computing Center (URZ)*

Impressum

Heidelberg University
Interdisciplinary Center for Scientific Computing (IWR)
Engineering Mathematics and Computing Lab (EMCL)

Im Neuenheimer Feld 205,
69120 Heidelberg
Germany

W. Karl, J. Keller (Eds.): PARS 2017
Proc. 27th PARS-Workshop

Development and implementation of a temperature monitoring system for HPC systems

Martin Baumann[1], Fabian Gebhart[1], Oliver Mattes[1], Sotirios Nikas[1], Vincent Heuveline[1]

Abstract: In the context of high-performance computing (HPC), the removal of released heat is one challenging topic due to the continuously increasing density of computing power. A temperature monitoring system provides insight into the heat development of an HPC cluster. The effectiveness of this is directly related to the number of sensors, their placing and the accuracy of the temperature measurements. Monitoring is important not only to investigate the efficiency of the cooling system for purposes of detecting defective operation of the HPC system, but also to improve the cooling of the servers and by this the achievable performance. The main purpose of a fine-grained and unified temperature monitoring is the possibility to optimize the applications and their execution regarding the temperature spreading on HPC systems. Based on this, we present a highly flexible and scalable – in terms of cable length and number of sensors – and at the same time budget-friendly monitoring infrastructure. It is based on low-cost components such as Raspberry Pi as monitoring client and a setup using the DS18B20 digital thermometer as temperature sensor. Focus is given on the selection of adequate temperature sensors and we explain in detail how the sensors are assembled and the quality assurance is done before these are used in the monitoring setup.

Keywords: temperature monitoring; HPC monitoring; energy efficiency

1 Introduction

It is well-known that computer systems need cooling to prevent hardware damages and to increase the lifetime of the hardware. For modern CPUs there is an additional relation between achievable performance and temperature. These CPUs adjust their frequency based on temperature (among other things). The technologies for controlling the processor operating frequency are e.g. Turbo Boost for Intel CPUs and Turbo CORE for AMD CPUs. For example the Intel Xeon E5-2640v3 (Haswell) CPU has a base frequency of 2.60 GHz but can run up to the maximum turbo frequency of 3.40 GHz which has a clear influence on the achievable performance. For reasons of reproducibility (e.g. for code optimization or scaling tests), a homogeneous CPU frequency within all identical HPC servers over time is necessary. Therefore, the temperature of the HPC system has to be monitored and controlled.

Nowadays, new energy-efficient cooling technologies are used to reduce the energy consumption for cooling HPC clusters. For example the Knürr® DCD cooling

[1] Heidelberg University, University Computing Center (URZ)

doors [Ve], as used in the University Computing Center (URZ), are fanless heat exchanger modules in the rear door of server racks. Chilled water is pumped through that heat exchanger which cools down the hot air which is blown through these heat exchanger rear doors. This technology promises room-neutral cooling of racks. A fine-grained and flexible temperature monitoring system makes it possible to analyze such technologies in real conditions which is a first step for improvements.

To gain insight into the spatial and temporal distribution of the heat, temperature has to be monitored at a fine granularity with an adequate temporal resolution. Many servers and network components are equipped with temperature sensors providing temperature information on a server-level. However, to gain a deeper understanding of the overall heat development cycle within and around the HPC racks, additional temperature sensors outside the server hardware are needed. These sensors can be positioned in the near neighborhood of the racks or inside the rack. From sensors that are located in front and behind a rack, the heating/cooling effect of that specific rack on the room climate can easily be quantified. Sensors at different positions within a rack can help to optimize the ways of how IT components (servers, network, storage, etc.) should be positioned within a rack for approximately homogeneous temperature distributions. The development of a new monitoring system can be motivated by requirements in higher scalability in the number of sensors, the cost-effectiveness of the monitoring system and its sensors, more flexible extendibility in the variety of sensors or some specific functionalities that need to be available.

For monitoring the climate situation of server racks and their environment, several hardware and software products from different industrial vendors exist. *Rittal Corp.* offers the *CMC III* system which contains a central controller that can build serial interconnections between monitoring sensors by a Controller Area Network (CAN) bus, cf. [Ri]. The available sensors are integrated into small enclosures with two RJ-45 connectors which are used to build up the CAN bus using specific CAN bus cables. Different versions of the central controller CMC III exist which contain up to 4 or 32 sensors. *Raritan Inc.* offers a solution containing the *EMX smart rack controller* that serves as central unit of the IP-based monitoring system, cf. [Raa]. Two versions of the controller exist, both with RJ-12 ports, RJ-45 ports, USB ports and RS-485 ports for connecting a maximum of 16 respectively 128 sensors (using specific sensor hubs). The controller can be accessed via web interface e.g. to view the sensor values and configure the thresholds. The associated temperature sensors *DPX-T1* come with the corresponding RJ-12 plug and a resolution of 2 °C, cf. [Rab]. The manufacturer *AKCP* also offers a similar solution aimed to monitor HPC systems. Several components such as temperature-, humidity-, motion-, gas-, sound- and powersensors, cameras and Single Network Management Protocol (SNMP) enabled web-based monitoring devices are offered in order to build a monitoring system. The monitoring device, called *sensorProbe*, with either two, four or eight RJ-45 ports exists in various models. For example the *sensorProbe8* comes with eight RJ-45 ports, runs on a Linux operating system (OS) and includes TCP/IP and a built-in webserver with SNMP functionality, cf. [AKa]. The offered temperature

sensors are similar to the ones we use. The semiconductor microprocessor controlled *TMPXX Temperature Sensor* for example consists of CAT 5 cables and RJ-45 plugs connected to a sensor probe with a resolution of 0.5 °C, cf. [AKb]. All three exemplary solutions, CMC III, EMX smart rack controller and AKCP sensorProbe support different types of sensors (e.g. temperature, humidity, pressure, etc.) and allow for reading the measurements via the widely-used SNMP. Hence, flexible monitoring systems can be set-up with these commercial products and can be integrated into the infrastructure of existing monitoring tools.

The aforementioned commercial solutions, however, have some limitations as well. The maximum number of sensors for the presented solutions is 32 or 128, hence the scalability is limited. For each commercial solution, there exists a list of compatible sensors which cannot easily be extended by the customers which may be a restriction: In cases of changing requirements on the sensors during a project (e.g. varying measuring ranges that should be addressed or changing demands in the quality of the sensors or changing time resolution for the measurements), new sensors with specific specifications need to be added into the monitoring system. Additionally, there is a comparably heavy financial burden going along with the commercial solutions: A monitoring system consisting of only 32 temperature sensors build up using each of the aforementioned products would cost about between 3.000€ and 4.000€. It has to be taken into account that for many projects the number of needed sensors is far larger. The limited scalability in the number of sensors, the limited range of compatible sensors and comparatively high costs for setups with a high number of sensors are motivations to develop an own modular and flexible monitoring system based on cheap but powerful standard components. In general, the use of self-developed and open platforms is advantageous over commercial products whenever a product will no more be supported or provided by the manufacturer, since spare parts or new extensions might not be available any more. An open solution can be extended easily by individuals, institutions or a community which is why we published our code. If needed at limited effort new sensors, demand-oriented custom features and characteristics of the monitoring system can be added and developed. All types of measurements and information of the HPC cluster itself and its infrastructure (e.g. state of the workload of running computing jobs, or power consumption as measured by the power distribution unit (PDU), control of cooling fans) can be included into the monitoring while keeping time stamps synchronized. The increased level in flexibility of a powerful monitoring with different building blocks and extension possibilities is what is needed in scientific projects and justifies the additional work building up a custom monitoring system over buying a ready solution.

Now there are some non-commercial projects of people and institutions developing or employing their own solutions. However, these activities are not published which is why we started our project. Since the beginning of our project we were in contact with two groups developing sensor systems for monitoring HPC systems. The Regional Computing Center at Universität Hamburg (RRZ) operates a temperature monitoring system for eight HPC racks. The system consists of 30 pre-assembled

thermal probes based on the digital sensor *DS18B20* (same sensor as the one used in Heidelberg), fitted with 3-min Molex connectors to a bus of two segments connected at the center to form a T-shaped topology. Four-wire telephone cable and RJ-45 ISDN distributors (after removal of termination resistors) have been repurposed for the modular bus construction. The bus master is a DS2480B-based USB adapter connected to a standard server. The temperature measurements are read periodically (via a cron job[2]) from the bus using the 1-Wire File System (owfs[3]). They are stored locally and also sent via Secure Shell (SSH) to the institution's central Nagios[4] monitoring server. The second group at the computing center of Mainz University in the department *HPC Systems/System Security* developed a sensor system based on a Raspberry Pi for monitoring the temperature and humidity of the surrounding of an HPC cluster. Each Raspberry Pi is connected via a serial interface to a self-developed electronics circuit board which connects the sensors. The controller reads the values of the sensors via a bus protocol and forwards the measurements to the Raspberry Pi. The electronic board and the Raspberry Pi are assembled together into an enclosure and receive electric power via a Power-over-Ethernet (PoE) module. Each such system monitors one computer rack. The measurement data is sent to a central Ganglia[5] monitoring server where the data can be stored and visualized using a web interface. The developed solutions differ among other aspects in the specific sensors that are used, the way the cabeling is realized, and the way the measurement data is read and processed.

This work proposes a modular and flexible monitoring infrastructure consisting of temperature sensors, the cabling and a simple way of monitoring the temperature data using a Raspberry Pi. How to use standard cable connectors and bus boards to build large bus systems that allow for flexible changes and extension will be addressed. With focus on the details of the technical construction and a fast way of reading out the measured data of a high number of sensors, we extend our preliminary work [BNGar], presented at the bwHPC symposium in Heidelberg, 2016. Additionally, a validation procedure which ensures the expected quality of the implemented sensors is presented.

Section 2 and 3 describe what kind of sensors are used, how these are assemble on standard patch cables, how to connect these to a big bus system and how to read the temperature values. To quantify the quality of the sensor measurements, the sensors were exposed to a course of changing temperature. The result and the validation of the sensors with respect to their accuracy are presented in section 4. Hereafter in section 5, our sensor placement strategy for the racks of an HPC cluster and the related water-based cooling infrastructure used in Heidelberg is described. The paper is completed with concluding remarks and an outlook of further developments.

[2] https://cron-job.org/en/
[3] http://owfs.org
[4] https://www.nagios.org/
[5] http://ganglia.sourceforge.net/

2 A scalable and cheap temperature monitoring sensor network

In this section we will present our setup and describe it in detail. We use digital sensors, since those allow (in contrast to analogue ones) bus wiring, i.e. multiple sensors can be connected via one single cable and varying topologies are possible. After evaluating different sensors we choose the the digital temperature sensor *Dallas DS18B20*, cf. [Da]. Each single sensor is identified by its unique ID assigned by the manufacturer. The sensor *DS18B20* is a digital thermometer with three pins (ground, voltage, data), see Fig. 1(a). Its mode of operation ranges from $-55\,^{\circ}\mathrm{C}$ to $+125\,^{\circ}\mathrm{C}$ with an accuracy of $\pm 0.5\,^{\circ}\mathrm{C}$ from $-10\,^{\circ}\mathrm{C}$ to $+85\,^{\circ}\mathrm{C}$. The digital resolution can be chosen between 9 and 12 bit leading to a precision from $0.5\,^{\circ}\mathrm{C}$ to $0.0625\,^{\circ}\mathrm{C}$, for all later purposes the 12 bit resolution was chosen to exclude avoidable rounding errors. Due to its low price of about 1€ per sensor, the simplicity of the *1-wire protocol*, as well as the relatively high precision and temperature range the *DS18B20* is well suited for building up a temperature monitoring system for HPC clusters.

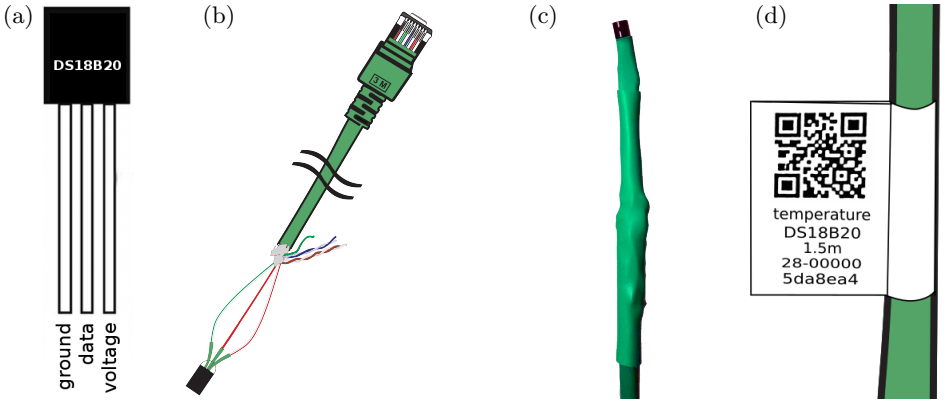

Fig. 1: (a) Illustration of the sensor *DS18B20*, (b) the assembled sensor cable with RJ-45 connector without heat-shrinking tubes, (c) assembled cable with heat-shrinking tubes, (d) label containing information about the sensor and the cable length.

Several vendors offer *DS18B20* sensors pre-assembled on cables. However, we assemble the sensors by ourselves for two reasons. The first reason is that we want to make use of standard network patch cables, because they have good shielding and many adapters exist for the commonly used RJ-45 connectors. The use of RJ-45 standard plugs facilitates the creation of big bus systems of varying topologies comprising a large number of sensors since standard patch cables and RJ-45 bus boards can be utilized (the latter are offered by various vendors). The second reason is that we want to be able to use the cabling in future, e.g. for a second sensor bus or other purposes. Currently, only three of the eight wires of a patch cable are used for the sensor, five wires remain available for possible future usage. The assembling process consists of bisecting a patch cable and soldering the three pins of a sensor to three specified wires of a patch cable, see Fig. 1(b). Heat-shrinking

tubes of three different diameters are used to avoid any short circuit between the three sensor pins and hold together the wires for higher resilience, see Fig. 1(c).

Each digital sensor has a unique ID, which is needed to distinguish the sensor values of the different sensors once these are connected in a large bus system. To determine the ID, the assembled sensor is connected to a Raspberry Pi and the sensor's unique ID can be read by checking the name of the mounted *1-wire* devices (see next section for details). For practical reasons, we equip each cable of an assembled sensor with a label containing the sensor's model name, ID, and cable length. To support automated reading, the label also includes the information encoded in a QR code, see Fig. 1(d). The QR code includes a JSON object with the aforementioned information.

3 Reading values from the sensor network

The Raspberry Pi is a popular low cost ARM-based mini-computer, which aims to introduce coding and computer science to all kinds of learners. It is widely-used for scientific applications including all types of monitoring purposes [SA16, IEB15]. With its huge community, the Linux based single-board computer is well-documented. It is not only a flexible and low-cost component, but it is also easy to be used and therefore perfectly fits our needs.

The *DS18B20* sensors can be connected to the general purpose input output (GPIO) pins of a Raspberry Pi as shown in Fig. 2. More sensors can be added by connecting these in parallel, i.e. ground to ground, data to data and voltage to voltage.

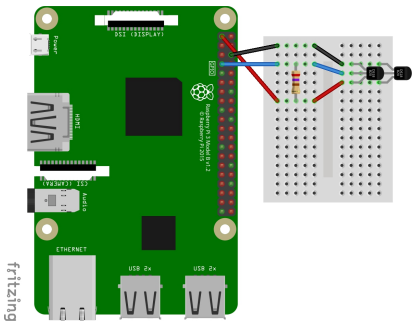

Fig. 2: Two temperature sensors are connected in parallel to the GPIO pins of a Raspberry Pi; a pull-up resistor is needed additionally.

Based on each individual sensor ID, the value of each sensor can be determined. Using the *1-wire protocol*, the Raspberry Pi is capable of reading the temperature data of all connected sensors. For example, the value of the sensor with ID "28-000005da34fa" can be read from the following device file:

```
/sys/bus/w1/devices/28-000005da34fa/w1_slave
```

Operating in the highest resolution (i.e. 12 bit) the temperature conversion time (i.e. the time the sensor needs to convert the analog into a digital signal) of the *DS18B20* sensor is 750 ms. Reading the sensors sequentially leads to a linear scaling of the overall reading time with the number of sensors connected, which is no longer reasonable for the purpose of live monitoring in case of a large number of sensors. To overcome this issue, we make use of the Python threading library[6], which provides a high-level threading interface. With this, interlaced accesses to multiple sensor device files are possible, so the conversation time of each sensor can be hidden. The resulting time consumption for one loop over all temperature sensors is much shorter compared to the time needed for sequential reading. Having a setup of 176 sensors takes 16 s instead of 132 s without multithreading. The Python code for reading out the temperature data using multithreading can be found on our project repository energy-efficiency-monitoring[7]. For people that are using owfs for reading the sensors values, the sequential reading time can be reduced by activating the so-called "simultaneous access".

The extension of the network, i.e. connecting more sensors, requires decreasing the resistance of the used pull-up resistor. Hence, bigger pull-up resistors (i.e. above 1 kΩ) are recommended for smaller networks. Our investigations have shown that the implementation of a rather large network with an increasing number of connected sensors as well as increasing total cable length has to be counterbalanced by decreasing the pull-up resistance. Besides this effect we also found two additional factors to influence the maximum number of sensors to be read. The Raspberry Pi model generation as well as the Linux Kernel version appeared to play a critical role. Our optimized final setup for temperature monitoring in the HPC cluster consists of a Raspberry Pi 3 Model B running the Debian based OS Raspbian (Version 8 *jessie*) on Linux Kernel version 4.4.38-v7+ connected to 176 Dallas *DS18B20* temperature sensors with a total cable length of 616 m and a pull-up resistor having 270 Ω. In various experiments, it turned out that the topology of the sensor network (star, linear and mixed topology) does not influence the performance of the system (i.e. response time of all sensors and reliability).

4 Quality of the sensors

Before the sensors are installed in the IT room for production operation, their functional capabilities and precisions are tested for determining outliers that will be replaced. To this end, all sensors are fastened side by side to a mobile frame so that the cables do not shield the sensors from the ambient air temperature. Based on the measurements, the mean temperature and the difference of each sensor to the mean are evaluated and analyzed to determine outliers.

The mobile frame with the attached sensors is first located in a room of constant temperature, after some time it is moved to another room with slightly warmer

[6] https://docs.python.org/2/library/threading.html
[7] https://github.com/uhd-urz/energy-efficiency-monitoring

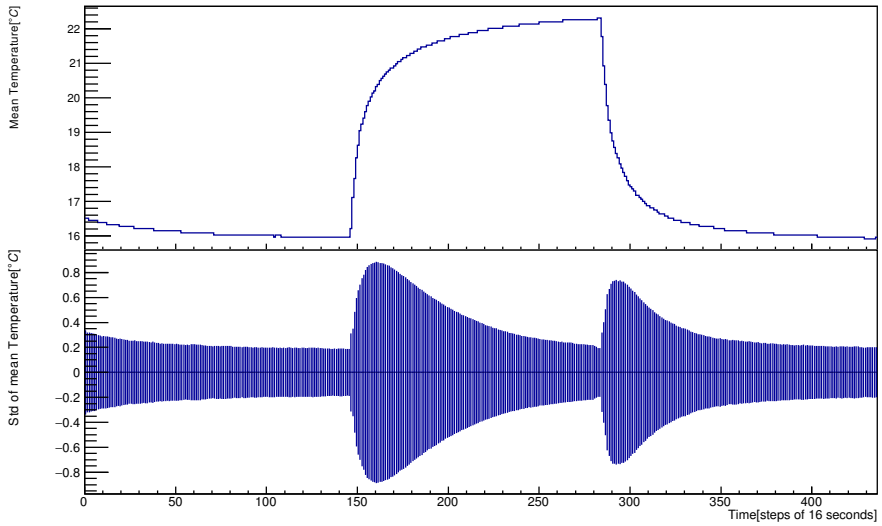

Fig. 3: Mean temperature (upper plot) and the standard deviation (lower plot) of 176 sensors exposed to a course with changing temperatures.

temperature, and finally moved back to the first room. The mean value of the measured temperature and the standard deviation are shown in Fig. 3. The temperature values converge from above the room temperature in the first room. In the first phase of converged temperature measurements (between time step #125 and #140), the mean temperature is approximately constant and the standard deviation has converged to approx. 0.2 ℃. After time step #145 the sensors are moved (almost instantaneously) into the warmer room. At the beginning the mean temperature increases very fast but decelerates finally. The rate of change of the mean temperature seems to depend on the difference to the constant temperature in the room, i.e. the higher the temperature difference the stronger the rate of change of the measured temperature. After time step #284, the sensors are moved back to the cold room and the opposite development of the measured values can be seen. Between time step #410 and #425 is a second phase of converged temperature measurements, in which the mean temperature is again approximately constant and the standard deviation is around 0.2 ℃.

For the detection of outliers among the sensors, we investigated the two phases of converged temperature measurements in the colder room in more detail. Figure 4 illustrates the maximal deviation over time during the two time spans, from step #125 to #140 and from #410 and #425, as histogram plot. For each sensor j, the maximal absolute value of the difference between the sensors measurement $t_j(i)$ at

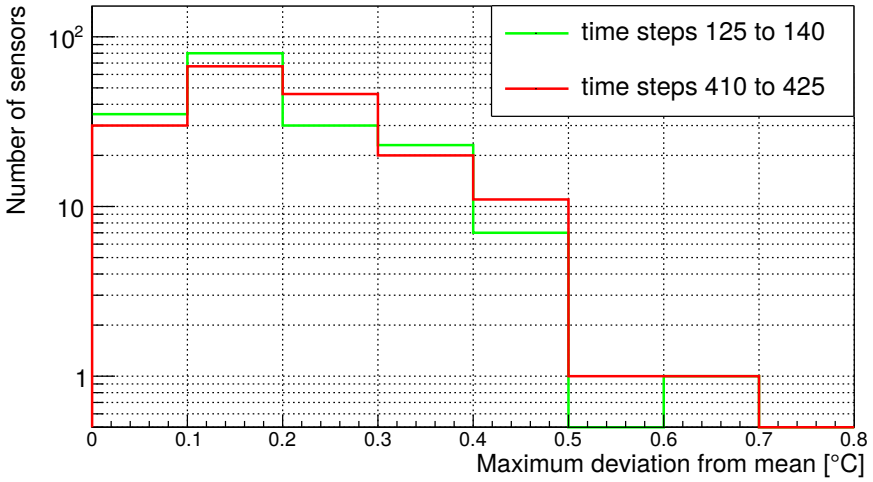

Fig. 4: Histogram of the maximal deviation of the 176 temperature sensors from their mean for two periods of time where the temperature measurements have converged.

time step i and the mean value of all sensors at that time is taken, i.e.

$$E_j, a \;:=\; \max_{i \in I_a} \left\| t_j(i) \;-\; \frac{1}{176} \cdot \sum_{k=1,\ldots,176} t_k(i) \right\| \tag{1}$$

where $j \in \{1,\ldots,176\}$, $a \in \{1,2\}$, $I_1 = \{125,\ldots,140\}$ and $I_2 = \{410,\ldots,425\}$. In Fig. 4 it can be seen, that the qualitative behavior of the sensor's deviation from the mean is similar in the two time periods. In the two time periods I_1 and I_2, there is one sensor that has a deviation of more than 0.5 ℃ which is the accuracy given in the manufacturer specification. It turns out that it is the exact same sensor having the highest deviation in both periods of time and therefore, this sensor is considered as outlier and removed from the sensor system. The remaining sensors meet the expected accuracy and will be used in production.

According to the vendor of the *DS18B20* sensor, a systematic error is going along with all bandgap-based sensors. This sensor type has been calibrated by the vendor by comparing the measured values T_M to a high-quality approximations T_R of the temperature. The error between these two can be described statistically, where second order polynomials can be taken to approximate the mean error. Figure 1 in [Da] shows the typical error curve of that sensor, where the error is defined as $T_{Error} := T_M - T_R$. The general procedure of the validation is described in [Ma]. We fitted the typical error curve given in [Da] by a second order polynomial function f as shown in Fig. 5. Thus, the measured temperature T_M can be compensated by this polynomial function by:

$$T_C := T_M - f(T_M), \tag{2}$$

where T_C is then supposed to be a better approximation of the temperature than the measurement T_M itself. We added this compensation as an optional feature to our implementation available on our aforementioned project repository.

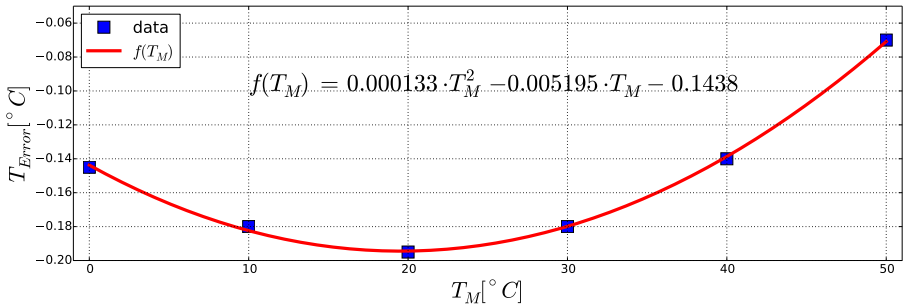

Fig. 5: The fitting function f approximates the mean error of the typical error curve given by the vendor of the sensor.

Coming back to Fig. 3 it can be clearly seen that the sensors obtain an effect of inertia due to the time-depending temperature course. This effect is limiting the informative value of the measurements. Since the temperature values of the environment of an HPC cluster can change (depending on the workload and other parameters) fast, the effect of inertia should not be neglected. It is planned to address the effect in future work, as outlined in the last section of this work.

5 On-site installation

The cooling technology employed in the IT rooms of the Heidelberg University Computing Center is based on a passive cooling concept, cf. [e315]. The heat released by the servers and other IT components within a rack will be evacuated by the fans of the servers through the read door of the rack. The read door serves as heat exchanger, i.e. heat in the air is cooled down while the cold water running through the door is heated up, see Fig. 6 (left). The heat development of an HPC system environment primarily depends on the heat balance of the air flowing into and leaving the server racks. The out-flowing air temperature again is determined by the temperature within the server rack and the effect of cooling by passing through the rear door.

For an overall picture of the spatial temperature distribution, several temperature values must be known: the air temperature in front, inside and behind each server rack as well as the water temperature before and after flowing through each server rear door. In contrast to the warm water - which depends on the energy consumption of the specific rack - the cold water temperature is approximately constant for every rack. Therefore, it is sufficient to use only one sensor for measuring the cold water temperature. For the determination of the HPC cluster's surrounding temperature climate, three additional sensors are positioned above each rack row.

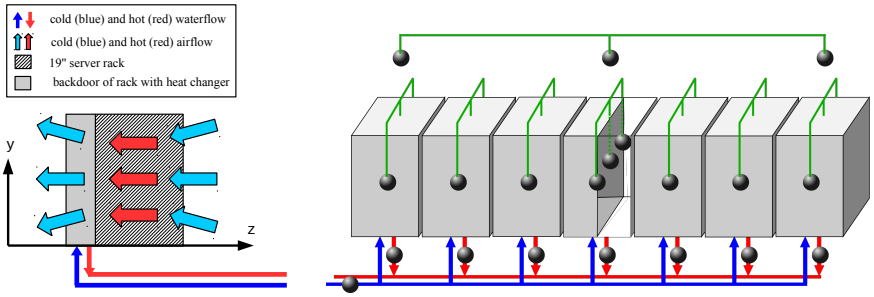

Fig. 6: Sketch of the operating principle of the cooling system on a rack basis at the Heidelberg University Computing Center (left) and the sensor positioning in the environment of a server rack row (right) sensors are indicated by black spheres.

Overall, there are four sensors installed at each rack, plus one additional sensor per rack row for the cool water temperature, plus three sensors for the air temperature above the rack row, compare Fig. 6 (right).

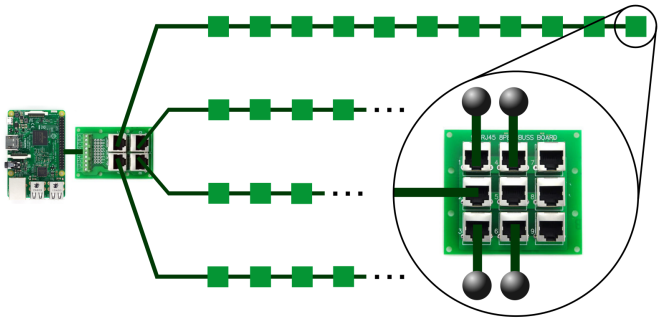

Fig. 7: Illustration of the star topology: The Raspberry Pi is connected via its GPIO connector to the sensor bus which consists of four branches of eleven bus boards serving as hubs. Each bus board is connected to four sensors indicated by black spheres.

Typically HPC systems are built into several racks standing next to each other which in turn are arranged in rack rows. In our case, we a considering a machine room consisting of four rack rows, each consisting of ten racks. This structure of racks can be reflected by the use of a star topology with four branches (for the four rack rows), compare Fig. 7. In each branch we install eleven RJ-45 bus boards. Ten of which correspond to the ten racks in that row and the eleventh bus board for the sensors of the rack row (i.e for measuring the cold water plus the air above the rack row). This setup leads to a network of 176 sensors in total.

6 Conclusion and Outlook

With the continuously increasing density of compute elements in HPC systems, the removal of the released heat is one of the major topics. Utilizing a temperature

monitoring system, an effective cooling but also an optimized application execution and higher performance can be achieved regarding the head spreading inside the HPC system. The effectiveness of this is directly related to the available information and in more detail to the number of sensors, their placing and the accuracy of the temperature measurements. In this paper we propose an open fine-grained and unified monitoring system to break through the limited scalability in the number of sensors, the limited diversity of available sensors and comparatively high costs per sensor of commercial monitoring systems. Based on this, we present a highly flexible and scalable – in terms of cable length, number and diversity of sensors – and at the same time budget-friendly monitoring infrastructure.

We presented how the digital sensor *DS18B20* can be assembled to standard cables, the development of a labeling system including QR codes and how we operate a large sensor network. The usage of standard CAT 5 cables and RJ-45 plugs offers the ability to build large bus networks of varying topology. The totality of the sensors of such a network can easily be read out in high frequency using a Raspberry Pi. The Python code for reading the sensor values is published. Hence, all components needed for building the measurement infrastructure is standard and cheap, leading to a comparably affordable and flexible sensor system for monitoring HPC systems.

Although the quality of the temperature sensors *DS18B20* is high, the measured values, in case of fast changing temperatures, are limited due to inertia effects. This might be caused by the sensor's plastic cover which is given by the design of the manufacturer. There are several approaches to improve the dynamic response behavior of the sensors, e.g. on the hardware level (reduce the plastic cover thickness) or on the level of extrapolation based on physical or mathematical models. We will address different approaches to increase the dynamic response behavior of the measurements in future works, which we presumably address within a collaboration with the RRZ of University Hamburg.

In the next phase of the project, we also will include the energy consumption to the monitoring system on the server level (build-in features of the servers), on the PDU-level (build-in features of power distribution units) and on the rack-level. For the rack-level, specific sensors based on inductive current will be investigated and presumably be built and installed to the supply lines of each rack. With this setup, we will be able to analyze the connection between energy consumption, released heat and temperature development on different levels.

For different use cases (many small jobs, few large jobs, jobs that are CPU-bound, jobs that are IO-bound, etc.) the energy and heat footprint will be determined using the monitoring system. Based on this information, we will look for *hot spots* of the HPC system at which temperature is particularly high (sometimes or frequently). Such situations can give a hint that hardware damages exist (e.g. limited functionality of fans) or suboptimal air flows (e.g. because of occlusion effects caused by disadvantageous cable routing). It could also be the case that too many IT components are built into small volume (too high energy density) or an suboptimal spreading of the IT components within the rack.

For different representative use cases, we will set the measurements in relation to the specific application or even the program phases. In general, it seems to be beneficial to account for the energy and temperature footprint of a compute job when the starting time of the job and the occupied servers are determined by the HPC system's management. However, a challenge is that the footprint of new jobs that should be run on a cluster is in general not known.

Nevertheless, we see some potential for a productive utilization of such technologies. Software packages or sub-routines like function calls or kernels that run often with similar parameters on the same HPC system will most often show similar energy and temperature properties. This information can be used for the scheduling of new application runs. Another approach would be to use the monitoring information to determine the temperature environment state of free nodes. New compute jobs then could be scheduled to run on nodes which are in a preferably relative cold environment. Additionally, the information of a continual online monitoring of the heat situation of the HPC cluster might be used to detect local heat spots and might even actively be used to reduce the load of some servers. This could be done e.g. by moving running jobs to other nodes (if possible) or by stopping some jobs and restarting these on other nodes. Such actions would be simplified in virtualized environments. A pragmatical strategy to get the full performance out of the hardware (for the jobs that need this) and the possibility of temperature optimization would be to reserve a certain number of servers per rack for dynamic load scheduling which will idle if the temperature in the rack is reduced. This could be facilitated, e.g. by a virtualized operational model of these nodes.

Acknowledgements

Some preliminary work, in particular for the conceptual design and development of the monitoring system, was done by Thomas Hochstrasser as part of his bachelor thesis [Ho16]. The authors acknowledge support by the Ministry of Science, Research and the Arts of the State of Baden-Württemberg (MWK) through the project bwHPC-C5.

References

[AKa] AKCP Data Center Monitoring: Datasheet for the *AKCP sensorProbe8*. Technical report, http://www.akcp.com. Available at `http://www.akcp.com/datasheets/Base-Units/sensorProbe-Series/sensorProbe8.pdf`.

[AKb] AKCP Data Center Monitoring: Datasheet for the *TMPXX Temperature Sensor*. Technical report, http://www.akcp.com. Available at `http://www.akcp.com/datasheets/Sensors/Environmental/TMPXX-Temperature-Sensor.pdf`.

[BNGar] Baumann, Martin; Nikas, Sotirios; Gebhart, Fabian: Towards a temperature monitoring system for HPC systems. In: Proceedings of the 3rd bwHPC-Symposium – Heidelberg 2016. heiBOOKS, Heidelberg, (to appear).

[Da] Dallas Semiconductor: Data sheet for DS18B20 programmable resolution 1-Wire® digital thermometer. Technical report, http://www.dalsemi.com/. Available at https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[e315] e3computing: eCube cooling technology for data centers. Technical report, http://www.e3c.eu/, 03 2015. Available at http://www.e3c.eu/wp-content/uploads/2015/04/2015_03_19_Cooling_technology_en.pdf.

[Ho16] Hochstrasser, Thomas: Konzeption und Entwicklung eines verteilten Systems zum Sensor-Monitoring, Bachelor thesis. Master's thesis, Ruprecht-Karls-Universität Heidelberg, Institut für Informatik, 08 2016.

[IEB15] Ibrahim, Mohannad; Elgamri, Abdelghafor; Babiker, Sharief: Internet of Things based Smart Environmental Monitoring using the Raspberry-Pi Computer. Department of Electrical & Electronic Engineering, University of Khartoum, Sudan, 10 2015.

[Ma] Maxim Integrated: Technical notes for Curve Fitting the Error of a Bandgap-Based Digital Temperature Sensor. Technical report, https://www.maximintegrated.com/en.html/. Available at https://www.maximintegrated.com/en/app-notes/index.mvp/id/208.

[Raa] Raritan: EMX Smart Rack Controller. Technical report, http://www.raritan.com/. Available at http://assets.raritan.com/resources/data_sheets/raritan-ds-EMX_rack_monitoring.pdf.

[Rab] Raritan: Environmental Sensors - Data Sheet. Technical report, http://www.raritan.com/. Available at http://assets.raritan.com/resources/data_sheets/Environment-Sensors-Data-Sheet-V1191.pdf.

[Ri] Rittal: Sensor Network for Rack and Room Monitoring. Technical report, http://www.rittal.de/. Available at https://www.rittal.com/imf/none/5_2079/.

[SA16] Shete, Rohini; Agrawal, Sushma: IoT Based Urban Climate Monitoring using Raspberry Pi. In: International Conference on Communication and Signal Processing. India, 04 2016.

[Ve] Vertiv™: Vertiv™ Knürr® DCD Cooling Door. Technical report, http://www.vertivco.com/. Available at https://www.vertivco.com/en-us/products-catalog/facilities-enclosures-and-racks/.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2017-06 Simon Gawlok, Philipp Gerstner, Saskia Haupt, Vincent Heuveline, Jonas Kratzke, Philipp Lösel, Katrin Mang, Mareike Schmidtobreick, Nicolai Schoch, Nils Schween, Jonathan Schwegler, Chen Song, Martin Wlotzka: HiFlow3 Technical Report on Release 2.0

No. 2017-05 Nicolai Schoch, Vincent Heuveline: Towards an Intelligent Framework for Personalized Simulation-enhanced Surgery Assistance: Linking a Simulation Ontology to a Reinforcement Learning Algorithm for Calibration of Numerical Simulations

No. 2017-04 Martin Wlotzka, Thierry Morel, Andrea Piacentini, Vincent Heuveline: New features for advanced dynamic parallel communication routines in OpenPALM: Algorithms and documentation

No. 2017-03 Martin Wlotzka, Vincent Heuveline: An energy-efficient parallel multigrid method for multi-core CPU platforms and HPC clusters

No. 2017-02 Thomas Loderer, Vincent Heuveline: New sparsing approach for real-time simulations of stiff models on electronic control units

No. 2017-01 Chen Song, Markus Stoll, Kristina Giske, Rolf Bendl, Vincent Heuveline: Sparse Grids for quantifying motion uncertainties in biomechanical models of radiotherapy patients

No. 2016-02 Jonas Kratzke, Vincent Heuveline: An analytically solvable benchmark problem for fluid-structure interaction with uncertain parameters

No. 2016-01 Philipp Gerstner, Michael Schick, Vincent Heuveline, Nico Meyer-Hbner, Michael Suriyah, Thomas Leibfried, Viktor Slednev, Wolf Fichtner, Valentin Bertsch: A Domain Decomposition Approach for Solving Dynamic Optimal Power Flow Problems in Parallel with Application to the German Transmission Grid

No. 2015-04 Philipp Gerstner, Vincent Heuveline, Michael Schick : A Multilevel Domain Decomposition approach for solving time constrained Optimal Power Flow problems

No. 2015-03 Martin Wlotzka, Vincent Heuveline: Block-asynchronous and Jacobi smoothers for a multigrid solver on GPU-accelerated HPC clusters

No. 2015-02 Nicolai Schoch, Fabian Kiler, Markus Stoll, Sandy Engelhardt, Raffaele de Simone, Ivo Wolf, Rolf Bendl, Vincent Heuveline: Comprehensive Pre- & Post-Processing for Numerical Simulations in Cardiac Surgery Assistance

No. 2015-01 Teresa Beck, Martin Baumann, Leonhard Scheck, Vincent Heuveline, Sarah Jones: Comparison of mesh-adaptation criteria for an idealized tropical cyclone problem

No. 2014-02 Christoph Paulus, Stefan Suwelack, Nicolai Schoch, Stefanie Speidel, Rdiger Dillmann, Vincent Heuveline: Simulation of Complex Cuts in Soft Tissue with the Extended Finite Element Method (X-FEM)

No. 2014-01 Martin Wlotzka, Vincent Heuveline: A parallel solution scheme for multiphysics evolution problems using OpenPALM

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de