# Towards a temperature monitoring system for HPC systems

Martin Baumann, Sotirios Nikas, and Fabian Gehbart

Computing Centre, Heidelberg University

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 which is important to investigate the efficiency of the cooling system and allows for defect detection. We present a scalable and flexible monitoring solution based on low-cost components with a Raspberry Pi as a monitoring client and propose a set-up for monitoring the temperature of an HPC cluster.

## 1 Introduction

In the IT rooms of the Heidelberg University Computing Centre, a highly efficient cooling technology is realized where the outdoor air temperature is used for water chilling (free cooling). The racks are equipped with water-cooled heat exchanger back doors to cool the exhausting air to ambient air temperature (details can be found in [1]). To gain a deeper understanding of the heat situation surrounding or inside the computer racks, a high number of temperature sensors is required. We describe what sensors we are using, how we connect these to a big bus system, and how we read and manage the temperature values.

## 2 Sensor technology and flexible bus-cabling

For this project, we use the *Dallas DS18B20* temperature sensor, cf. [2], which is a digital thermometer with three pins (ground, voltage, data), see Fig. 1 (a). This sensor can measure temperature values between -55 °C and +125 °C with an accuracy of ±0.5 °C from -10 °C to +85 °C. Its resolution can be chosen between 9 and 12 bit leading to a precision from 0.5 °C to 0.0625 °C. Due to its low price of about €1 per sensor, the simplicity of the 1-Wire protocol, and the relatively high precision and temperature range the *DS18B20* is well suited for building up a temperature monitoring system for an HPC cluster.

Several vendors are offering *DS18B20* sensors pre-assembled on cables. However, we are assembling the sensors by ourselves to be able to use standard network patch cables, since those cables have good shielding and for the RJ-45 connectors many adapters exist. Only three of the eight wires of a patch cable are used for the sensor, five wires remain available for other purposes (e.g. other sensors). The assembling process consists of bisecting a patch cable and soldering

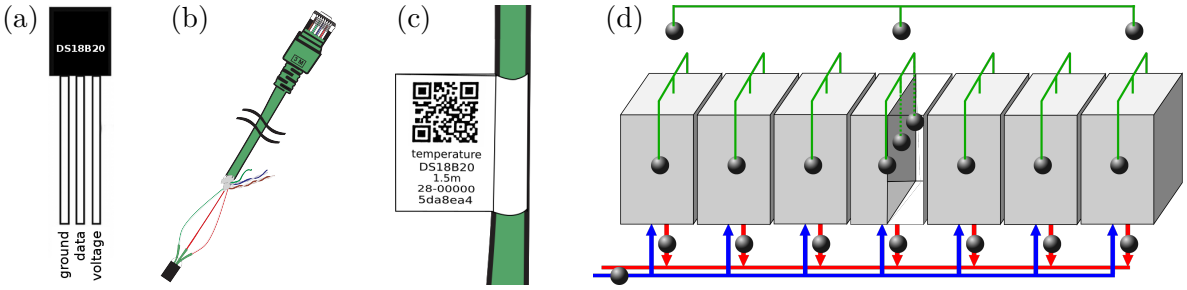


Figure 1: (a) Illustration of the sensor *DS18B20*, (b) the assembled sensor cable with RJ-45 connector without heat-shrinking tubes, (c) label containing information about the sensor and the cable length, and (d) the sensor positioning in the environment of a server rack row (sensors indicated by black circles).

the three pins of a sensor to three specified wires of a patch cable, see Fig. 1 (b). Heat-shrinking tubes of different diameters are used to avoid any short circuit between the three sensor pins and hold together the wires for higher resilience.

Since we are using RJ-45 connectors, a big bus system comprising a large number of sensors can be built using standard patch cables and RJ-45 bus boards which are offered by various vendors. In the end, the bus system is connected to the general purpose input output (GPIO) pins of a Raspberry Pi. By reading the sensors separately (see next section), each sensor's unique ID can be determined. This ID will be exploited for the identification of each single sensor in a bus system of many connected sensors. For practical reasons, we equip each assembled cable with a label containing the sensor's model name, ID, and cable length. To support automated reading, the information is additionally encoded in a QR code which is placed on the label, see Fig. 1 (c).

## 3 Reading sensor values within a monitoring system

For reading the sensor values, we have written a short Python program that makes use of the 1-Wire protocol. This tool is executed on a Raspberry Pi allowing to read the values of all sensors in the bus system. Before the sensors are installed in the IT room for production operation, their functional capability and precision are evaluated. To do so, all sensors are positioned alongside and simultaneous temperature measurements are conducted. Additionally, the response time of the sensors, i.e. the time the sensors need to follow changing temperatures, can be considered for the validation process. Thus, the outliers can be identified and sorted out.

There are several influence factors that can limit the maximum number of sensors that can be connected. These include the cabling (e.g. the length of the cables) and the ohmic resistance of the pull up resistor (we used a resistor with 270 Ω). It turns out that even the Linux Kernel of the Raspberry Pi's operating system has some influence (later versions seem to be better; we used the kernel version 4.4.14-v7+). In our set-up, the maximum number of sensors that could be read out by the Raspberry Pi in one single bus was 161 sensors.

For the realization of a monitoring system for a high number of temperature sensors existing monitoring tools such as Zabbix or Nagios can be employed. For that purpose, a Simple Network Management Protocol (SNMP) agent was developed to run on the Raspberry Pi that allows communication using the SNMP, cf. [3], which is a standard protocol for information management in IP networks. The client sends temperature values to the monitoring server on

a regular basis where the data is stored for a long time. The server can have powerful visualization and analysis features and can also be extended by custom features. Additionally, the client can observe the temperature values with a higher temporal resolution (e.g. to recognize outliers within seconds) and can send *traps* to the monitoring server in case of overheating or other noticeable situations.

## 4 On-site installation

As mentioned before, the cooling technology employed in the IT rooms of the Heidelberg University Computing Centre is based on passively cooled back doors. In this technology, the heat development of the environment of an HPC system primarily depends on the heat balance of the air flowing into and leaving the server racks. The outflowing air temperature again is determined by the temperature within the server rack and the effect of cooling by passing through the back door. The back door serves as heat exchanger which increases the water temperature flowing through the back door while cooling the air.

For an overall picture of the spatial temperature distribution, several temperature values must be known: the air temperature in front, inside and behind of each server rack and the water temperature before and after flowing through each server's back door. In contrast to the warm water (depending on the rack's energy consumption), the cool water temperature is approximately the same for each rack, for which reason it is sufficient to use only one sensor. For the determination of the HPC cluster's surrounding temperature climate, some sensors are positioned above the rack row. Overall, there are four sensors installed at each rack, plus one additional sensor per rack row for the cool water temperature, plus few sensors for the air temperature above the rack row, see Fig. 1 (d).

In the next phase of the project, additional sensors for measuring the energy consumption will be integrated to the bus system and will be included in the monitoring. The relation between the energy consumption of the HPC system while running different applications and the associated heat development will then be analysed.

## Acknowledgements

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

[1] e3computing: eCube cooling technology for data centers, http://www.e3c.eu, March 2015.

[2] Dallas Semiconductor: Data sheet for DS18B20 programmable resolution 1-Wire® digital thermometer, http://www.dalsemi.com/.

[3] T. Hochstrasser: Konzeption und Entwicklung eines verteilten Systems zum Sensor-Monitoring, Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, Institut für Informatik, 08-2016.