Open-source active cooling system for GPU and FPGA accelerator cards

Rodolfo Manin* and Alberto Saa†

Instituto de Matemática, Estatística e Computação Científica, Universidade de Campinas, Campinas, SP, Brazil

August 10, 2022

Abstract

Typically, high-performance Graphics Processing Units (GPU) and Field Programmable Gate Arrays (FPGA) accelerators are high-power demanding elements, designed to be mounted in specialized racks with intense forced airflow systems to assure their functioning within proper thermal limits. We present here an open-source, Arduino-based, programmable active cooling system for GPU and FPGA cards that allow their installation in common computer cases. Only standard components and simple 3D-printed pieces are employed in our design. We built and have been continuously using this cooling system for a GPU NVIDIA Tesla V100 and an FPGA Xilinx Alveo U280. The setup is inexpensive and has proved to be flexible and extremely reliable. Furthermore, it can be easily adapted for any other card or element requiring similar cooling.

Metadata Overview

Main design files: `https://github.com/albertosaa/ActiveCooling`
Target group: scientists and engineers demanding high performance computing, datacenter managers.
Skills required: Arduino programming – easy, 3D printing – easy, electronics and mechanical assembly – easy.
Replication: The authors built and have continuously used the setup for a GPU NVIDIA Tesla V100 and an FPGA Xilinx Alveo U280.
See section "Build Details" for more detail.

Keywords

High Performance Computing (HPC), GPU, FPGA, active/passive cooling.

---

*ORCID: 0000-0002-6154-340X, email: rodolfo@ime.unicamp.br
†ORCID: 0000-0003-1520-4076, email: asaa@ime.unicamp.br

(1) Overview

Introduction

High-performance Graphics Processing Units (GPU) and Field Programmable Gate Arrays (FPGA) accelerators are widely employed nowadays in intensive scientific computing and machine learning applications. The cutting-edge models are typically cooled by a passive heat sink system and designed to be installed in some specific, high-cost, PCIe/NVLink racks [1] with a strong front-to-back forced airflow system to assure their operation within proper thermal limits. Fig. 1 illustrates common passively cooled NVIDIA GPU card. Under intense use, the power consumption of a typical high-end accelerator card can easily

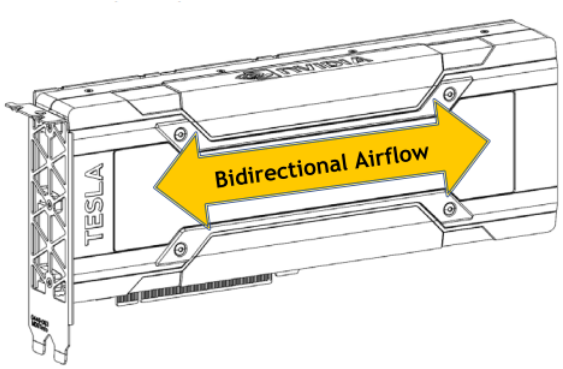

Figure 1: Schematic representation of the passive cooling airflow for a GPU NVIDIA Tesla P100, see [2].

surpass 300 W [3], requiring a minimal fully ducted airflow through the card's interior heat sink of approximately 35 cubic feet per minute (CFM) [2]. Such operational features clearly discourage any alternative installation of these high-power demanding accelerator cards in nonspecific racks.

However, it is indeed not uncommon to have some of these cards without the proper racks to install them. Having this situation in mind and our own recent necessities, we have designed and built an inexpensive open-source, Arduino-based, programmable active cooling system for a GPU NVIDIA Tesla V100 and for an FPGA Xilinx Alveo U280 mounted in a common PC motherboard in a standard rack-mounted computer case. Only off-the-shelf components and simple 3D-printed pieces are employed in our design. For each card, we used a customized 3D-printed air duct and a dedicated cooler fan with pulse width modulation (PWM), activated by our Arduino-based controller circuit, with a temperature sensor placed at the air exit port at the back of the card. The controller circuit is configured to adjust linearly the power fan for a temperature range $[T_{\min}, T_{\max}]$ that can be set accordingly to the particular card and typical operation cases. However, since the Arduino-based circuit is highly flexible, virtually any other controlling strategy and monitoring of the power fan can be easily implemented at the Arduino code.

We have extensively tested our cooling system and it has proved to be very

flexible and reliable. Both accelerator cards are now being continuously used in a high performance computing environment. Finally, our setup can be easily adapted for any other card or element with similar heat sink cooling necessities.

Overall Implementation and design

Fig. 2 depicts our complete setup for a GPU NVIDIA Tesla V100 and for an FPGA Xilinx Alveo U280 mounted in a common PCIe motherboard (ASUS PRIME Intel H510M-E) in a standard rack-mounted computer case. The 3D-

Figure 2: The complete active cooling setup for a GPU NVIDIA Tesla V100 (top) and for an FPGA Xilinx Alveo U280 (bottom).

printed ducts in high temperature Polylactic Acid (PLA), the fans, and the controller circuit boards are clearly visible. See Fig. 3 for some details of the duct design for the GPU NVIDIA Tesla V100. For further information, build details, and pertinent source files, see the full build instructions at [4]. The airflow is provided by a dedicated $60 \times 60 \times 38$ mm 4-pin PWM cooler fan with a maximum airflow of 53 CFM. To guarantee mechanical stability and avoid vibrations, the fan must be fixed at the computer case, and not at the card. To this end, it is important to employ a computer case with a transverse bracket for fans as those ones depicted in Fig. 2.

The Arduino-based fan controller circuit is depicted in Fig. 4. All pertinent details can be found in the full build instructions at [4]. The controller circuit can

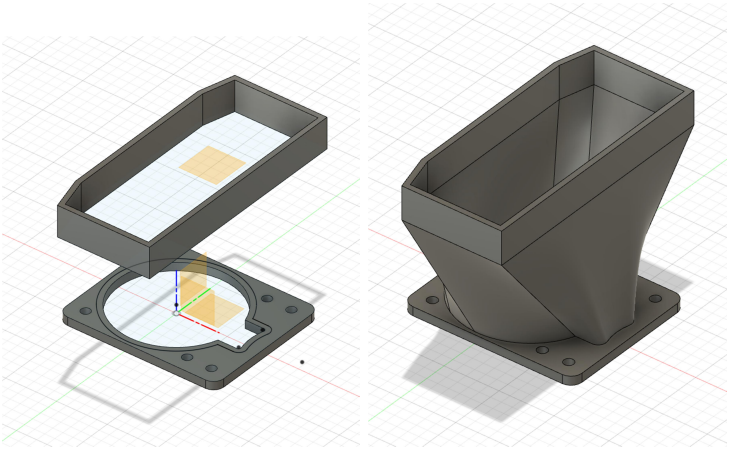

Figure 3: The design of the 3D-printed duct for the GPU NVIDIA Tesla V100. Left: 3D model of the customized end connections for the card and for the fan. Right: A standard 3D modeling software tool is used to generate the smooth structure connecting the end connections, giving origin to the duct. All design details, instructions, and specifications are available at [4].

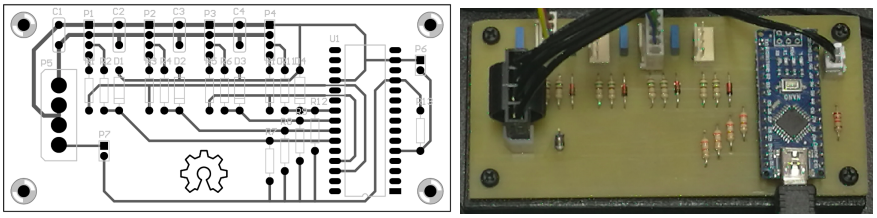

Figure 4: The fan controller based in an Arduino Nano microcontroller. Left: the circuit board. Right: The assembled controller circuit, which can control up to two pairs of 4-pin PWM fans ($P1, P2$ and $P3, P4$ connectors). The NTC thermistor is connected to $P6$. The 12 V ATX high power supply is connected to $P5$. In our case, the Arduino 5 V supply is being provided by the USB connection, but it can also be supplied by the ATX power on P5 if jumper P7 is closed. For further details about the electronics, see the full build instructions at [4].

activate up to two pairs of 4-pin PWM fans with independent programs. We are limited to only two independent programs since the Arduino nano has only two PWM channels with 16-bits timer, the remaining four 8-bits channels are inadequate to control the fans. In our case, a second common $120 \times 120 \times 38$ cooler fan, pairwise activated with the main cooler, is placed at the front part of the computer case to provide enough airflow into the case. The temperature sensor, a Negative Temperature Coefficient (NTC) thermistor, is placed at the air exit port in the back of the card, see Fig. 5. We have programmed the Arduino to adjust linearly the power fan for the thermistor temperature in the range $[T_{\min}, T_{\max}]$. In our case, for continuous use in a high-performance computing environment, we have set the fan power at 40% and at 100%, respectively, for

Figure 5: Location of the temperature sensor (NTC thermistor), at the air exit port in the back of the card.

temperatures below $T_{\min} = 40\,°\text{C}$ and above $T_{\max} = 55\,°\text{C}$. For the intermediate temperatures, the fan power is linearly increased. Notice that one could use some commercially available alternative fan controller circuit as, for instance, the ZFC39 PWM PC CPU Fan Temperature Controller [5], for implementing simple controlling strategies as the linearly increasing one. However, one of the greatest advantages of our Arduino-based circuit is namely its high flexibility, allowing virtually any other programmable controlling strategy and monitoring for the power fan. Our setups are installed in an air-conditioned datacenter, and the cards can work at full power with internal temperatures comfortably below their working limit of 70 °C.

Universality of the design

Our design can be easily applied for any other card or element with similar heat sink cooling necessities. The Arduino can be programmed to implement virtually any controlling strategy and monitoring of the airflow.

(2) Quality control

Safety

There are very few security hazards in the assembling and operation of the setup. The computer power source must remain unplugged during the assembling. The GPU and FPGA cards must never work without the proper cooling system. For instance, our GPU NVIDIA Tesla V100 and FPGA Xilinx Alveo U280, without the proper airflow, reach the shutdown temperature (around 90 °C) in some few minutes of operation. Running these cards for long times in temperatures above their proper limits (around 70 °C) can permanently damage them.

Calibration

Since we use thermal components, some calibrations are indeed required. The NTC thermistor is incorporated in our circuit by means of a simple voltage divider, see Fig. 6, whose output voltage V_{out} is read by one of the Arduino analog input pins. From V_{out}, the thermistor resistance $R(T)$ is determined. The function $R(T)$ for an NTC thermistor is given by the well-known Steinhart-Hart

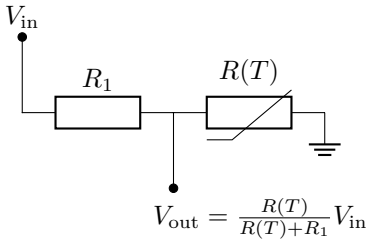

Figure 6: Simple passive voltage divider with an NTC thermistor $R(T)$ and a resistor R_1. The output voltage $V_{\rm out}$ is read by one of the Arduino analog entries. Since $V_{\rm in}$ and R_1 are known, $R(T)$ can be determined from the value of $V_{\rm out}$. See [4] for further details.

equation [6]

$$T^{-1} = a + b \log R + c \log^3 R, \tag{1}$$

where the absolute scale is used for the temperature T. Hence, to convert between T and R, we need the three parameter a, b, and c, which can be determined, for instance, by a least square fitting from a set of measurements (T_i, R_i), $i = 1, \ldots, n$, with $n > 2$. However, for narrow temperature ranges, as in the present case, the last term in the Steinhart-Hart equation can be neglected, and we have the more familiar relation

$$\beta \left(T^{-1} - T_*^{-1}\right) = \log \frac{R}{R_*}, \tag{2}$$

where we have set $a = T_*^{-1} - b \log R_*$ and $b = \beta^{-1}$. We use (2) in our Arduino code [4]. The simplest of-the-shell NTC thermistors are characterized by their nominal resistance R_* for $T_* = 25\,^\circ\mathrm{C}$ and the parameter β. In our case, we employed thermistors with $R_* = 50\,\mathrm{k}\Omega$ and $\beta = 3950\,\mathrm{K}$. These parameter, which are input data for the Arduino code, can be easily double-checked with some simple temperature and resistance measurements.

However, the fundamental point of our calibration is not the thermistor characterization, but the management of the difference between the card functioning temperature, measured internally at its circuit components, and the temperature measured by the thermistor in the air exit port of the card. It is natural to expect that the thermistor temperature be smaller than the card internal temperature, but the difference can depend on many variables, from component details to environment conditions. The temperature range $[T_{\rm min}, T_{\rm max}]$ for the control of the fan power must be determined on a case-by-case basis. We suggest to set initially our configuration ($T_{\rm min} = 40\,^\circ\mathrm{C}$ with power fan $P = 40\%$ and $T_{\rm max} = 55\,^\circ\mathrm{C}$), and adjust it in order to have the fan running with its power around 70% for the cards in intense continuous use. It is not advisable to run the fans at full power in normal conditions since some temperature overshooting can and typically occur, requiring a quick momentary increase of the power fan.

The construction and the assembling of the customized air ducts require some careful positioning and measurements of the cards and the computer case brackets, see the full build instructions [4] for further details.

General testing

The testing of the cooling system is rather simple. We used some burn-in test programs to stress the high performance cards and monitored their temperature. See [4] for further details.

(3) Application

Use case

We built and have been continuously using this cooling system for a GPU NVIDIA Tesla V100 and an FPGA Xilinx Alveo U280. Both cards are mounted in a common PCIe motherboard ASUS PRIME Intel H510M-E, installed in a standard rack-mounted computer case, running in an air-conditioned datacenter. The setup has proved to be flexible and extremely reliable. All design and built details are available at [4].

Reuse potential and adaptability

The setup can be easily adapted for any other card or element requiring similar cooling systems. The 3D-printed PLA air duct must be customized for each card and computer case, but the overall building, calibration, and testing are essentially the same. For a suggested customization procedure for the air ducts, see the full build instructions at [4].

(4) Build Details

Availability of materials and methods

Only standard, easily available, components and simple 3D-printed pieces are employed in our design. We summarize below the main employed components and materials. See [4] for the full list.

1. Arduino nano microcontroller.
2. High-performance $60 \times 60 \times 38$ mm 4-pin PWM cooler fan as, for instance,
   (a) Sanyo Denki model 9GA0612P1K611 [7] or
   (b) Delta Electronics model PFC0612DE. [8].
3. Negative Temperature Coefficient (NTC) thermistor with $R_{25} = 50\,\mathrm{k}\Omega$.
4. High temperature Polylactic Acid (PLA) filament (also called "Premium PLA". Most regular PLA must not be used with temperatures above 50 °C).
5. Adhesive EPDM rubber foam tape, usual electronics components, usual bolts, etc.

Ease of build

No advanced skills are required. The most critical step in the building is the customization of the air-ducts, which involves some careful measurements and a 3D modeling. See the full build instructions at [4] for the details of the two already built cases.

Operating software and peripherals

We use an Arduino nano microcontroller and, hence, we need the Arduino software, which is free and widely available.

Dependencies

There is no relevant dependencies for the design.

Hardware documentation and files location:

Name: GitHub

Persistent identifier: `https://github.com/albertosaa/ActiveCooling`

Publisher: Rodolfo Manin and Alberto Saa

Date published: 2022–08–10

Hardware License: CERN-OHL-S-2.0

Software and Documentation License: MIT License

(5) Discussion

Conclusions

We have presented and built an open-source, Arduino-based, programmable active cooling system for a GPU NVIDIA Tesla V100 and an FPGA Xilinx Alveo U280 cards that allowed their installation and use in common computer cases. Our inexpensive setup, which has been continuously used it in a high-performance computing environment, has proved to be flexible and extremely reliable. Furthermore, it can be easily adapted for any other card or element requiring similar cooling.

Paper author contributions

Rodolfo Manin: 3D printing of the prototype pieces; mechanical and electronic assembling, Arduino programming.

Alberto Saa: acquiring resources and funding.

Both authors contributed equally to the conceptualization, design, testing, and validation of the system, and to the paper and documentation writing.

Acknowledgements

AS thanks Vitor Cardoso and José S. Lemos for the warm hospitality at the Center for Astrophysics and Gravitation of the University of Lisbon, where this work was concluded.

Funding statement

AS acknowledges the financial support of CNPq (grant 302674/2018-7) and FAPESP (grant 21/09293-7).

Competing interests

The authors declare that they have no competing interests.

References

[1] See, for instance, the SuperMicro GPU SuperServer SYS-420GP-TNR at https://www.supermicro.com/en/products/system/GPU/4U/SYS-420GP-TNR

[2] See, for instance, the specifications of the NVidia Tesla P100 PCIe GPU accelerator at https://images.nvidia.com/content/pdf/tesla/NV-tesla-p100-pcie-PB-08248-001-v01.pdf

[3] M. Spetko, O. Vysocky, B. Jansik B, L. Riha, *DGX-A100 Face to Face DGX-2 – Performance, Power and Thermal Behavior Evaluation*, Energies **14**, 376 (2021). https://doi.org/10.3390/en14020376

[4] https://github.com/albertosaa/ActiveCooling

[5] https://www.amazon.com/Temperature-Control-Speed-Controller-Module/dp/B019P0FLHW

[6] D. Ibrahim, *Microcontroller-Based Temperature Monitoring and Control*, Newnes - Elsevier, 2002.

[7] https://products.sanyodenki.com/en/sanace/dc/dc-fan/9GA0612P1K60

[8] https://www.delta-fan.com/Download/Spec/PFC0612DE-F00.pdf