Experience and Practice Teaching an Undergraduate Course on Diverse Heterogeneous Architectures

Eitan Frachtenberg, Reed College

November 14, 2021

Workshop on Education for High-Performance Computing (EduHPC'21)

Motivation

- Heterogeneous computing growing everywhere: low-end, desktop, supercomputers.
- The transition requires new tools, programming models, and **training**.
- Analogous to the transition from single-core to multi-core.

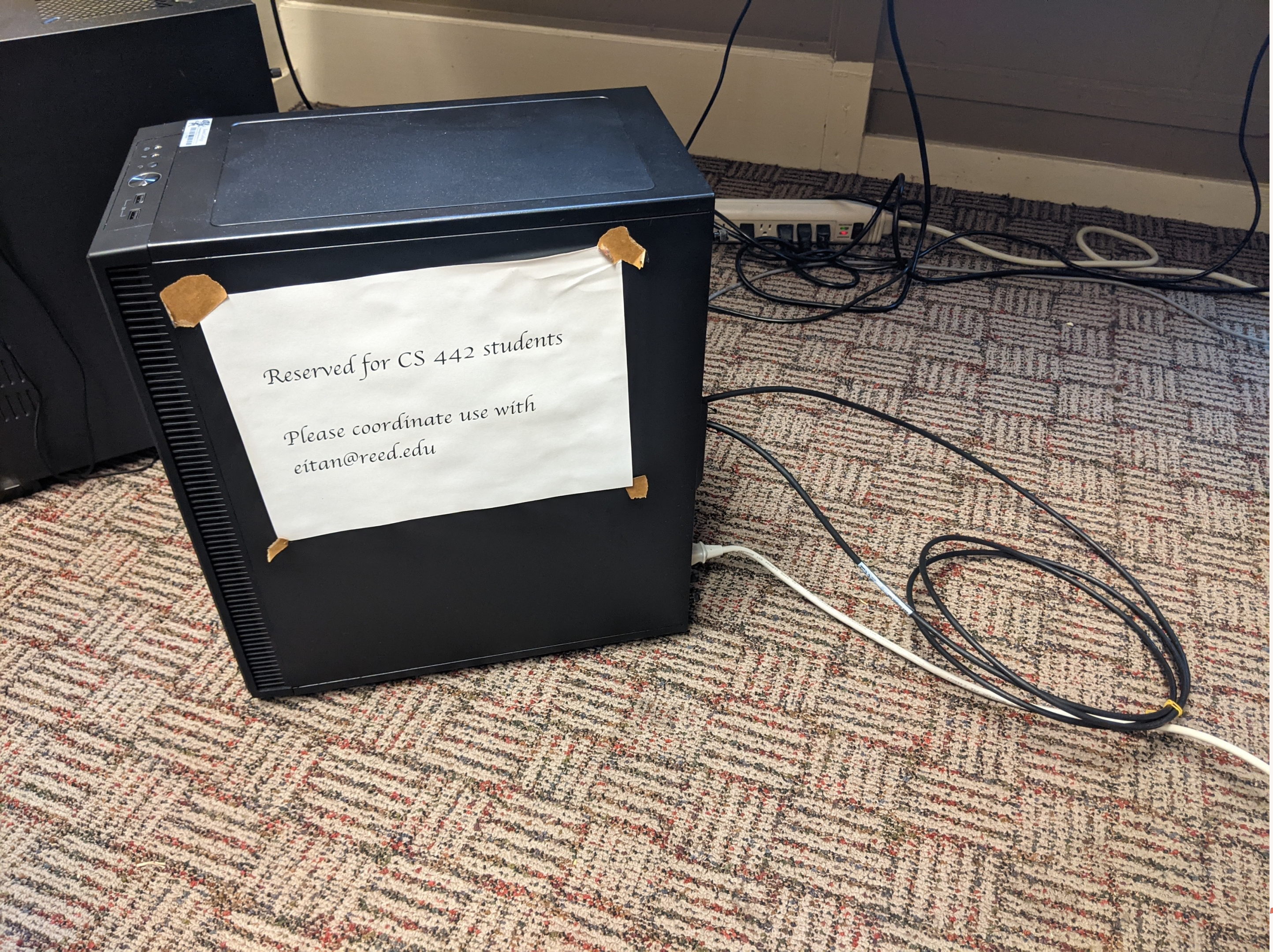
Goals

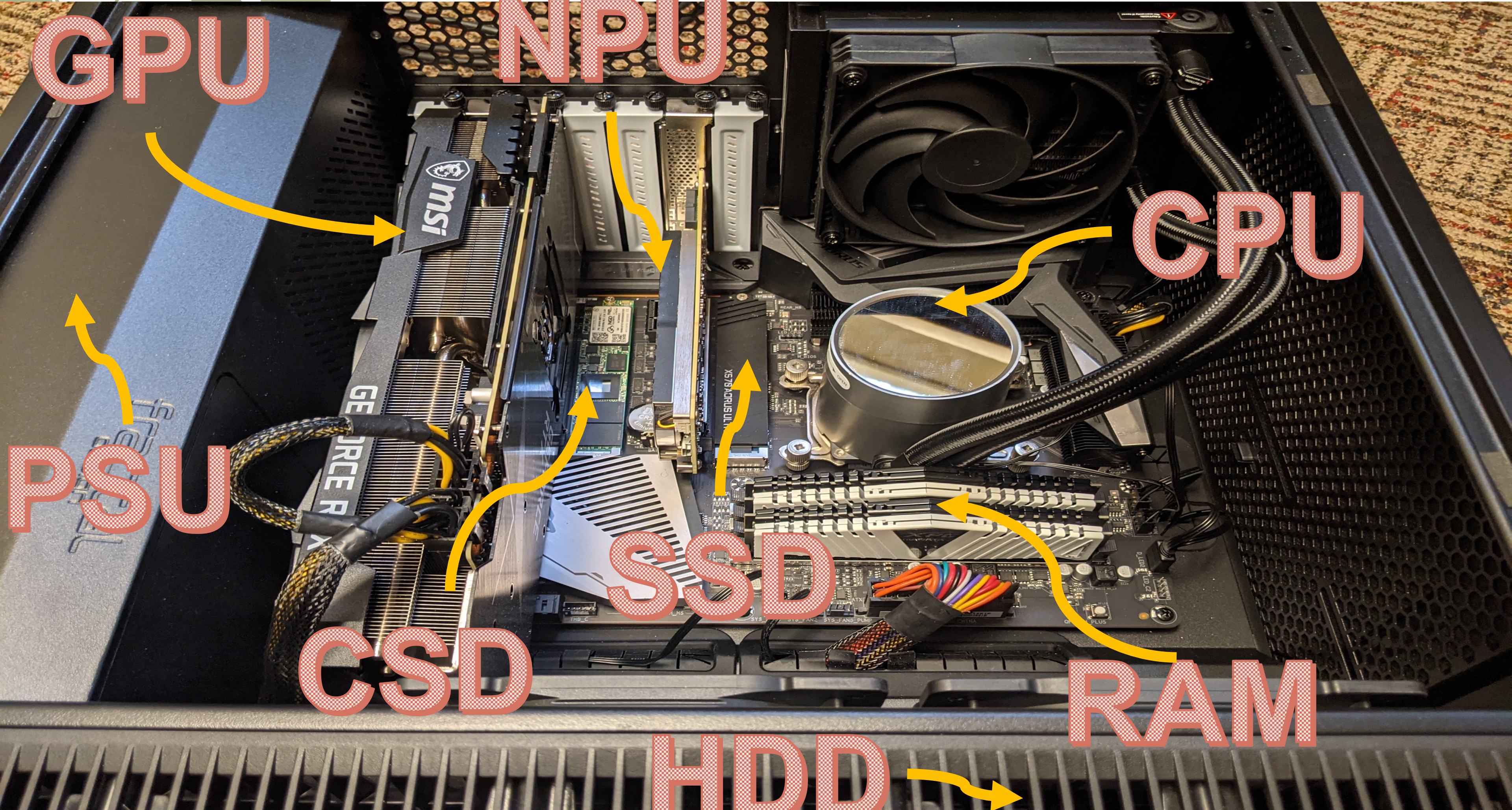
- 1) Understand the challenges and applications of heterogeneous computing through exposure to a **breadth** of contemporary research.
- 2) Practice the tools of heterogeneous computing by blending academic research with **hands-on** work.
- Develop familiarity with and appreciation for the typical systems research project **workflow**.

Class required CS389 (“computer systems”) as prerequisite and in-person attendance. Because many students were off-campus during the pandemic, final enrollment was only 7 students.

Hardware

- Enhance a COTS-based workstation (12-core AMD 3900 CPU, 48GB RAM, 500GB SSD+3TB HDD) with **diverse** accelerators:
  - An RTX 3090 GPU with 24GB RAM
  - A Mellanox BlueField1 SmartNIC (NPU) with 2x25GBe ports, 16-core ARM processor, 16GB RAM, crypto-enabled.
  - An NGD Systems Newport 8TB CSD with 4-core ARM processor, 2GB RAM.
- A second, nearly identical machine (lower-end GPU, BF2 with 8 cores).
- Two power meters.


Reserved for CS 442 students
Please coordinate use with
eitan@reed.edu


GPU
NPU
CPU
PSU
SSD
CSD
RAM
HDD
msi

Software

- Host and NPU running Ubuntu 20.04, CSD starting with 16.04, then 20.04.
- NPU and CSD accessible via the “network” over PCI, with their own IP addresses.
- Students given log-in credentials on all three, but no root access.
- Home directories from the host mounted on NPU and CSD via NFS.

Structure

1) Introductory lectures (1 week, 3 x one hour).

2) Mini-seminar (3 weeks).

3) Benchmarks (3 weeks).

4) Individual projects and reports (7 weeks).

Lectures

1) The emerging role of heterogeneous architectures, especially GPUs; their benefits and challenges; common tools and programming models.

2) The hardware and software architecture of the BlueField SmartNIC.

3) The hardware and software architecture of the Newport CSD.

Seminar

- Students picked two recent research papers (out of a list) on two different accelerator architectures or topics.
- Each lecture, two students presented one paper each.
- The presentations were informal 20-minute unstructured talks, typically only projecting figures from the original paper, no new slides.
- The entire class was expected to engage in conversation and Q&A.
- Students were evaluated on their understanding of the topic, not on their presentation skills.
- Students were asked to explain the paper’s findings, as well as:
  - “What is your main critique of the paper?”
  - ”What in the paper could become a class project?”

Benchmarking

- After the talks (and sometimes in parallel), students evaluated two different benchmarks (out of a long list) on two different architectures.
- The benchmarks selected tested I/O, HPC, CPU cache, databases, RDMA, and Hadoop.
- Students had to compile all benchmarks and their dependencies.
- Absolute performance numbers and comparison didn't matter. Relative performance per Watt did.
- Students recorded performance and power numbers, varied workloads and parameters, and plotted results.
- All intermediate results were shared in a cloud spreadsheet for any student to comment on, accompanied by classroom discussion.

Projects

- For the 2nd half of the semester, students worked individually on projects of their choosing, employing at least one accelerator:
  - User-level RDMA-based filesystem on the NPU.
  - A power-normalized chess tournament between NPU, CPU, CSD.
  - A tree-search chess AI running on the GPU.
  - Three separate implementations of differential privacy on CSD.
  - Parallel prefix on the GPU.
- Final reports on their project (~7 pages) was expected to include:
  1. Background & bibliography.
  2. Technical description of proposed solution.
  3. Evaluation (correctness, performance, energy).
  4. Conclusion.

Challenges and lessons learned

- Superuser access: Great lesson in installing software, but some dependencies don’t work well without root installation.
- NFS mounts: convenient but slow. Fails with binary dependencies.
- Software compatibility: Different OS versions lead to broken deps.
- Hardware compatibility: Ensure correct endianness and open APIs.
- CUDA complexity: Ab initio training doesn’t fit in 7 weeks.
- COVID-19: Can’t easily measure power over Zoom.

Scalability

- One machine can only serve so-many students (< 10).
- Equipment cost can be managed with lower-end or older components.
- Technical support increases with number of machines, accelerators, and students, requiring TAs and good support contract.
  - Best to stay away from “bleeding-edge” devices.
- Scaling attention across students developing many different benchmarks and projects is challenging. Some aspects of the class can be overlapped and others augmented with TAs.
- Overall, this class structure probably can’t scale beyond ~20-30 students.

Conclusion

- Course had ambitious goals that were mostly met, partly owing to the small class size.
- Students expressed overall satisfaction with the topic, appreciation for the wide choice of topics, the seminar experience, and the practical skills they learned.
- There is no baseline to compare this class against, as it was the first instance (and during COVID-19 no less!)
- Future iterations may deemphasize CUDA programming aspects and just rely on prebuilt GPU libraries instead. Project expectations should be clearer with example final report.

Additional resources in the paper:

- Details about the selected benchmarks and projects.
- Extensive bibliography for parts 1+2,
- Additional project ideas.