Computational Storage: Beyond the Storage Status Quo

Niclas Hedam

nhed@itu.dk

Data-Intensive Systems and Applications

Department of Computer Science

DASYA

Data-Intensive Systems and Applications

Storage Status Quo

- We generate data as never before [1].
  - 44 zetabytes stored worlwide early 2020 (A ZB is 10^{21} bytes).
  - We cannot process efficiently all this data efficiently. Why?

1. Throughput is an issue!
   - The throughput of storage devices has increased exponentially [2].
     - For example, reading 1 MB from SSD.
       - 50 ms in 1990, 5 ms in 2000, 500 µs in 2010, 50 µs in 2020 [3].
   - The throughput of memory has increased only linearly [2].
     - For example, compressing 1 KB of memory.
       - 362 µs in 1990, 11 µs in 2000, 2 µs in 2010, 2 µs in 2020 [3].
   - **We cannot process stored data fast enough!**
2. To be processed, data must be moved from where it is stored to a central processing unit [4].
   - Moving data requires much more energy than processing it.
   - **We cannot process stored data in a sustainable way!**

[1] https://seedscientific.com/how-much-data-is-created-every-day/
[2] Picoli, I. L., Bonnet, P., & Tözün, P. LSM Management on Computational Storage.
[3] https://colin-scott.github.io/personal_website/research/interactive_latency.html
[4] Mutlu, O., Ghose, S., Gómez-Luna, J., & Ausavarungnirun, R. (2019). Processing data where it makes sense: Enabling in-memory computation. Microprocessors and Microsystems, 67, 28-41.

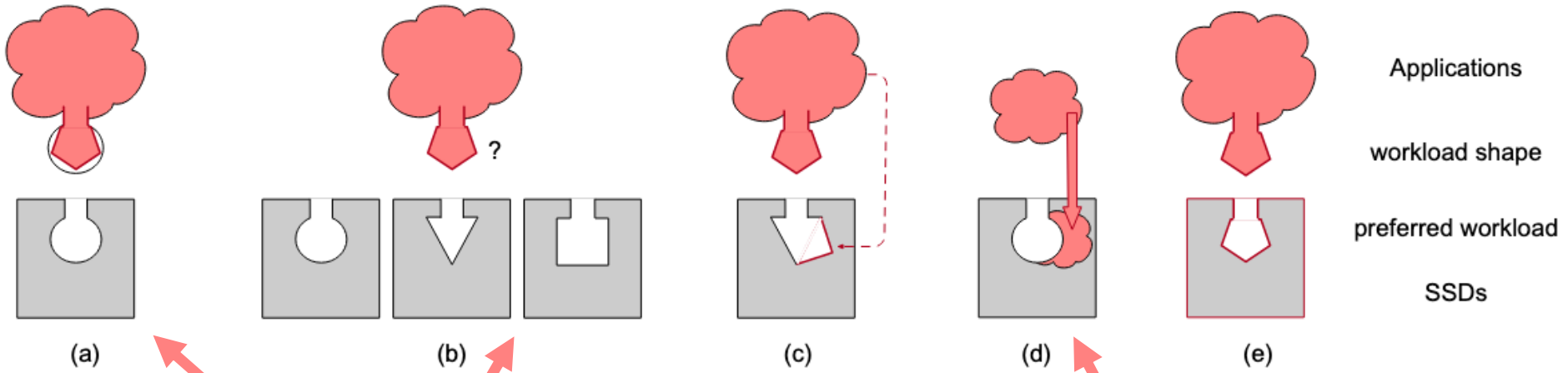
Storage devices today

Traditional approach
Applications unilaterally adapts to the SSD characteristic

Alternative approach
Applications developers find the best matching SSD and unilaterally adapts to the SSD characteristic

Configuration approach
Applications configure some aspects of the SSD for better performance

Computational Storage
Applications can either move parts of its logic to the SSD or allow the application to define the SSD in its entirety



Status Quo

My Research

Figure from:
Lerner, A., & Bonnet, P. (2021, June). Not your Grandpa's SSD: The Era of Co-Designed Storage Devices. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 2852-2858).

A pratical example – Summing up a list

- On the right we have a computer.
  - CPU is fast.
  - SSD is fast.
  - Bus & memory is slow.
- Status Quo looks like this.
  - The application wants to sum up a list.
  - The application requests data from SSD.
  - The whole list is transferred to the RAM.
  - The application sums up in CPU.
- My research looks like this.
  - The application wants to sum up a list.
  - The data sends a small program to the SSD describing how to sum up the list.
  - The result of the program is transferred to the CPU.

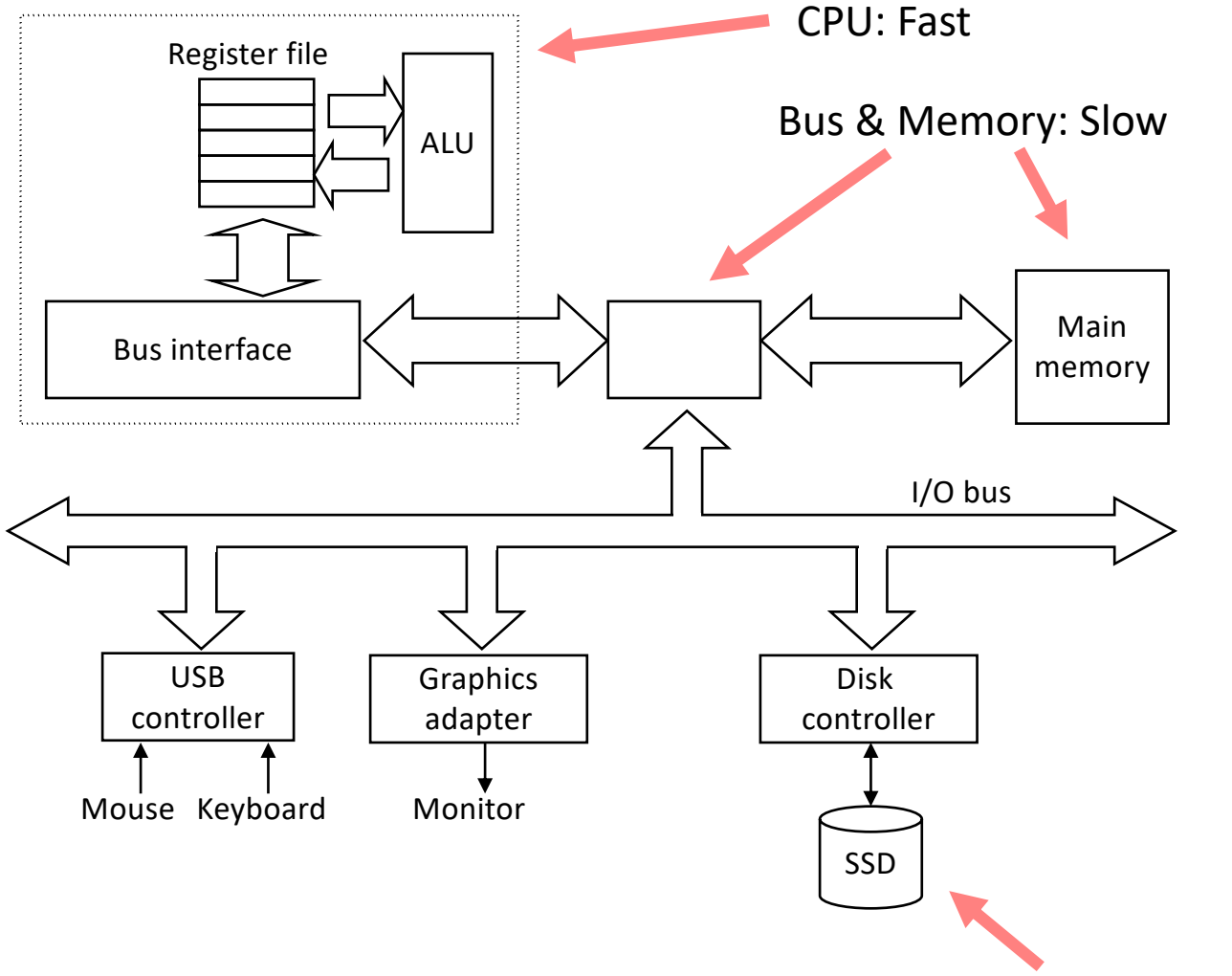


Figure from:
P. Bonnet. Operating Systems and C - Exceptional Control Flow

Whats the difference?

- Status quo requires you to retrieve the full list.
  - Depending on the list, that could be in the order of gigabytes.
  - It will block the bus while transferring, and block the CPU while summing up.
- My research requires a program to be transferred to the SSD.
  - A program is often < 1 KB.
  - While running, both the bus and CPU is free to do other work, i.e. low data movement between SSD and CPU.
  - My architecture has high throughput between data on the SSD and a processor within the SSD.

- To summarise: **more performant** and **more energy efficient**!

Challenges

- The major challenge of my PhD is defining how to offload programs to storage to process data where it is stored, rather than move data so that it can be processed.
  - How are programs represented?
  - How are programs transferred and executed?
  - How is memory managed across storage tiers?
  - How efficiently can offloaded programs be executed?
  - When is it a good idea to offload a program to storage?

Impact & Conclusion

- I already built a working device, which we are now evaluating.
  - The device uses eBPF to represent programs.
    - A world-wide standard (NVMe) will be unveiled in 2023 proposing eBPF as intermediate representation.
  - To my knowledge, my device is the **first** storage device that supports eBPF offload.
    - Next: benchmarking eBPF-based offload.