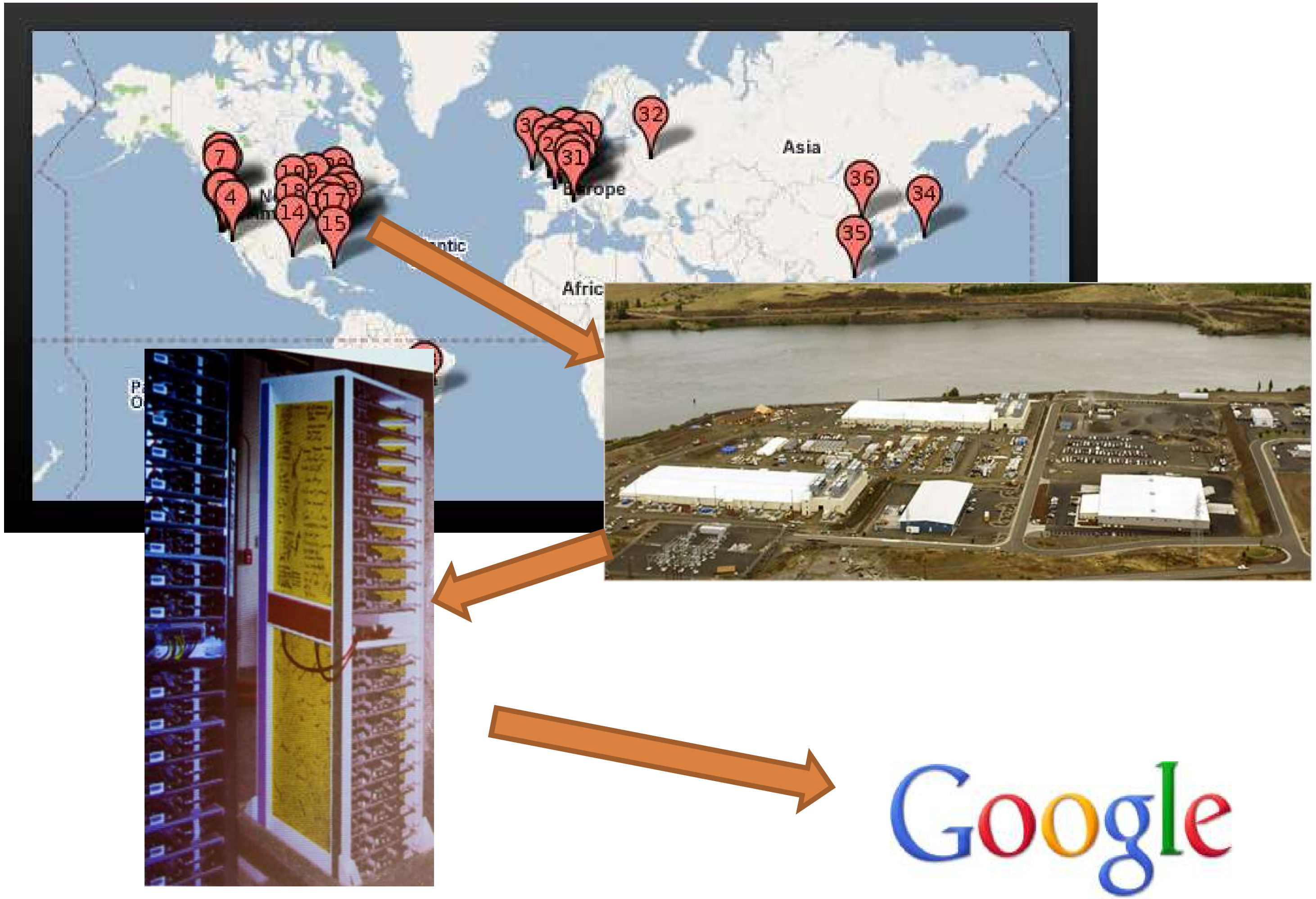
DCS: A Fast, Scalable, Flexible Device-Centric Server Architecture

Jangwoo Kim

E-mail: jangwoo@snu.ac.kr
Web: https://hpcs.snu.ac.kr/~jangwoo

High Performance Computer System (HPCS) Lab
Department of Electrical and Computer Engineering
Seoul National University

Major IT companies run datacenters

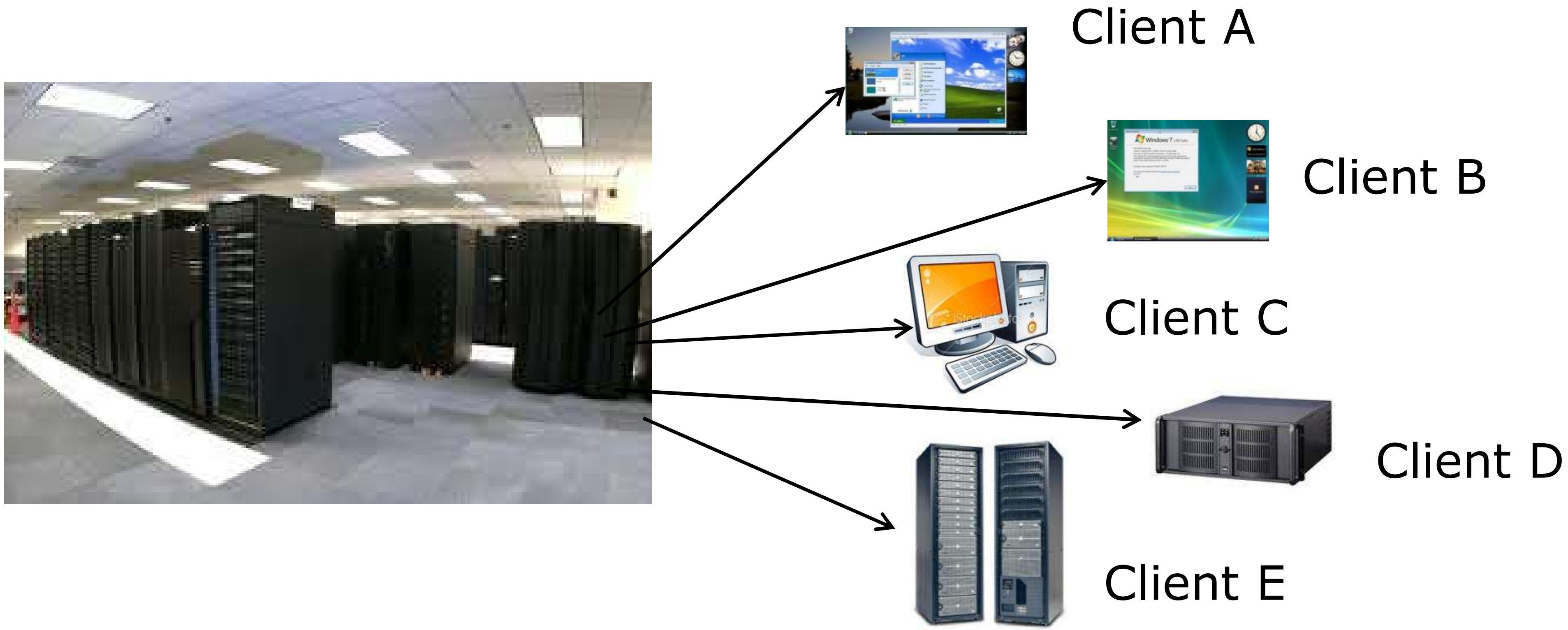


Google

Datacenter infra market is huge.

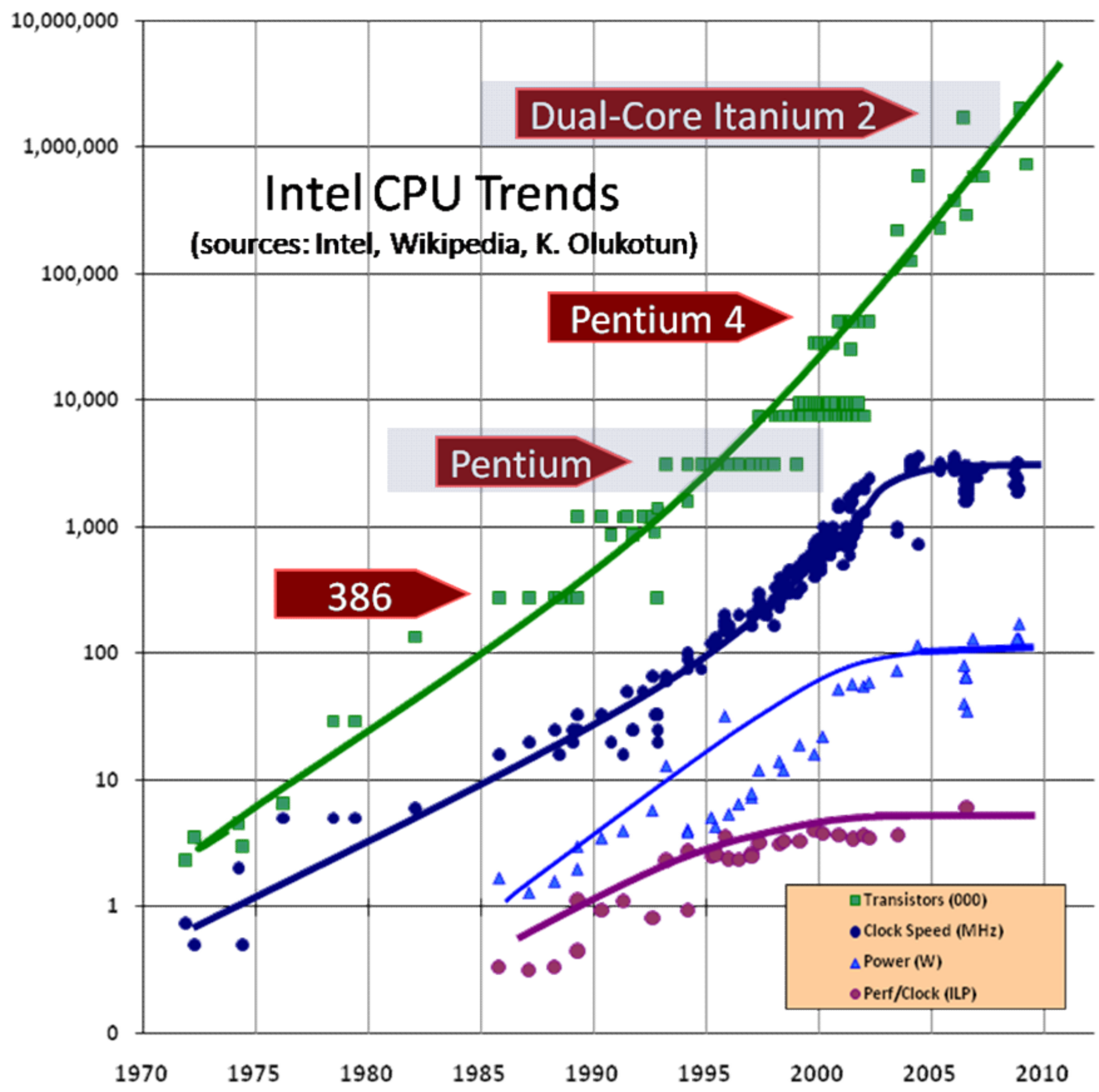
All others use the datacenters

- **Buy a SW/HW platform as a service**

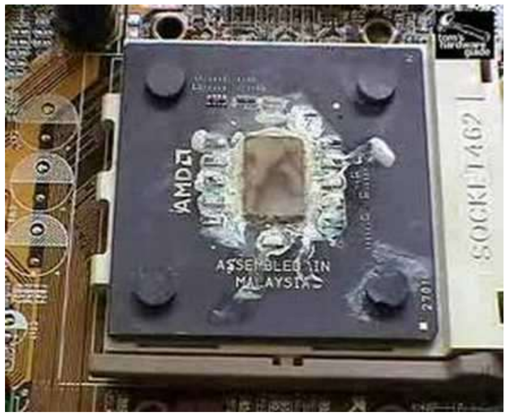


Again, datacenter infra market is huge.

Moore’s Law is Dead



What is the use
of extra transistors?



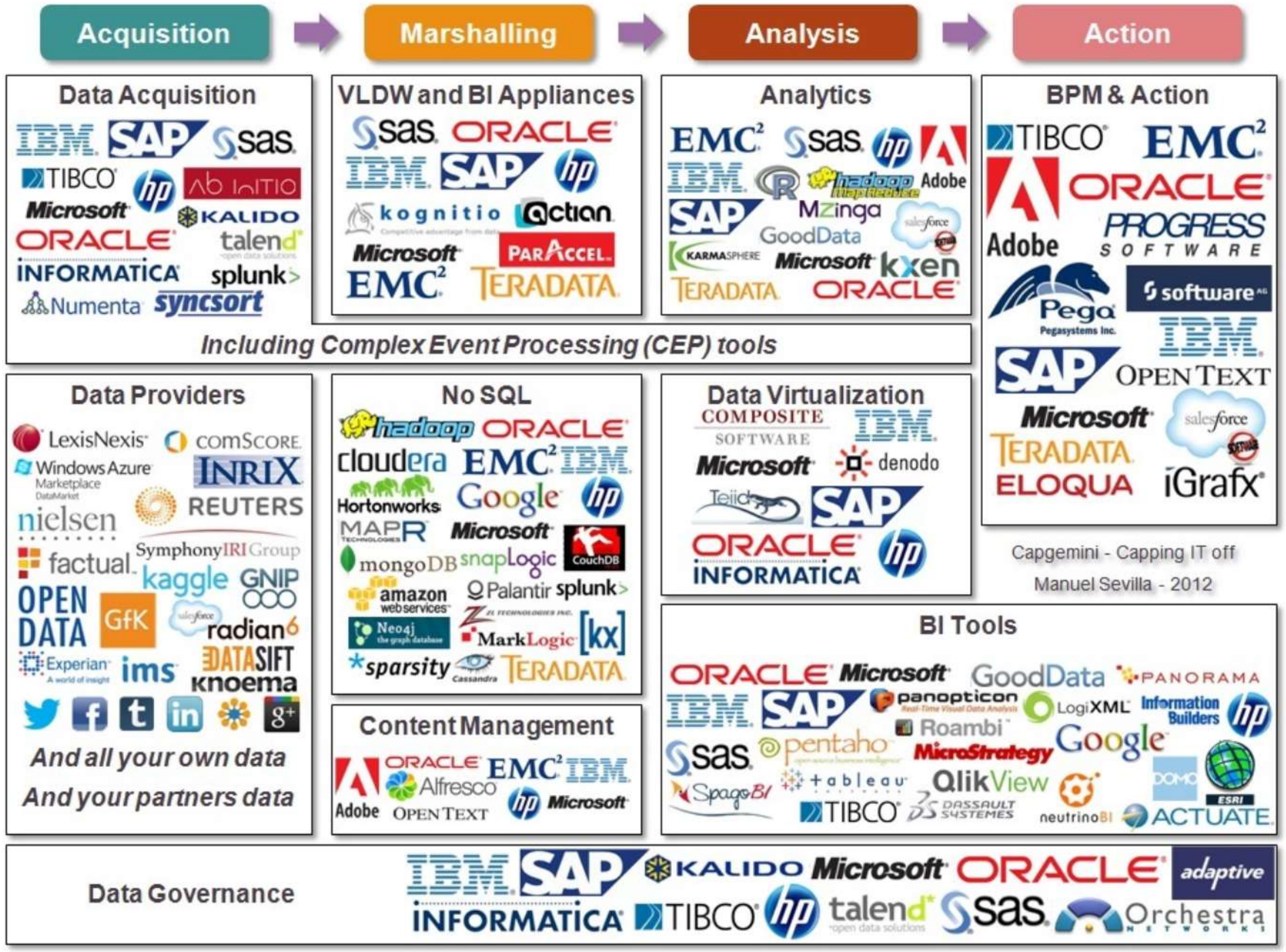
can’t build a faster CPU
due to the power ceiling

CPU is NOT the 1st-class citizen any more

VS

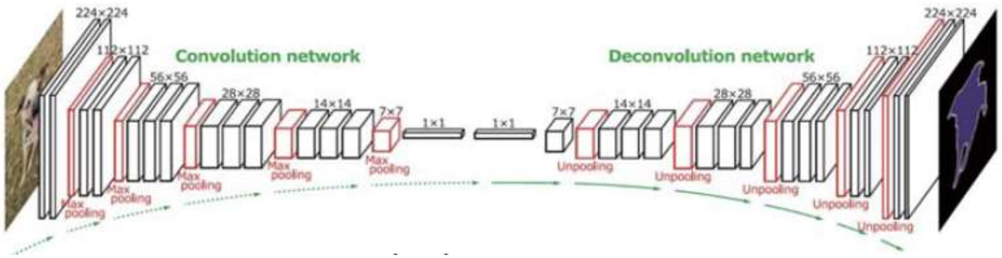
"Un-CPU" devices now dominate
the performance, power, and costs.

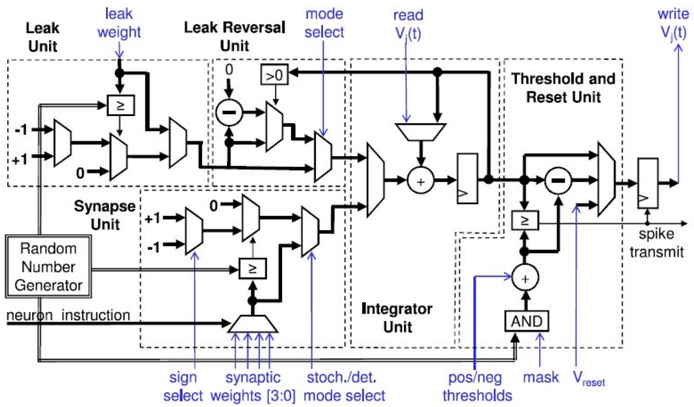
Every company now deals with big data

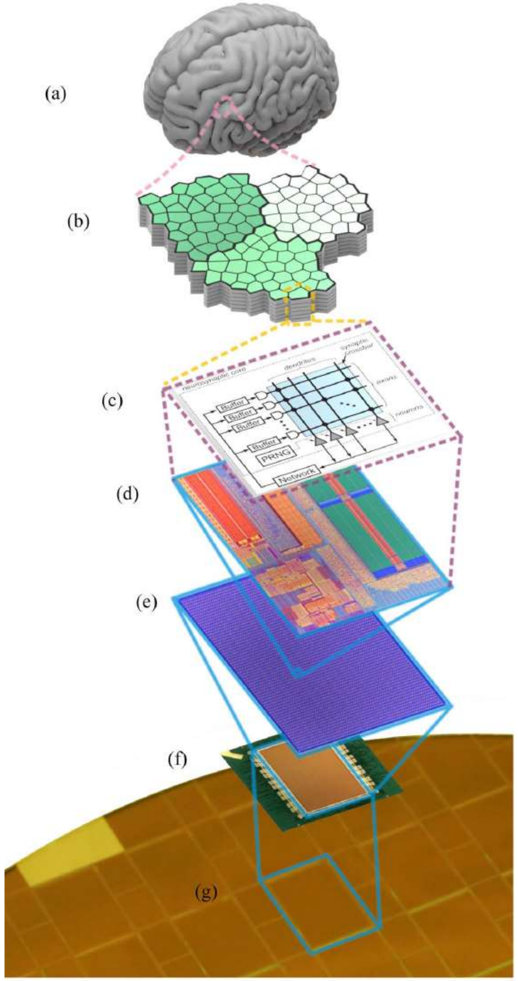


Storage infra market is EVEN larger!

Neuromorphic computer is coming







Brain-inspired computing → New World?

HPCS
High Performance Computer System Lab

- **Message #1** (for system engineers)

We must build a **datacenter-friendly, intelligent server**

(e.g., cloud, big data, artificial intelligence)

- **Message #2** (for system engineers)

The advantage must come from **emerging devices**

(e.g., Memory, SSD, GPU, ASIC, ..)

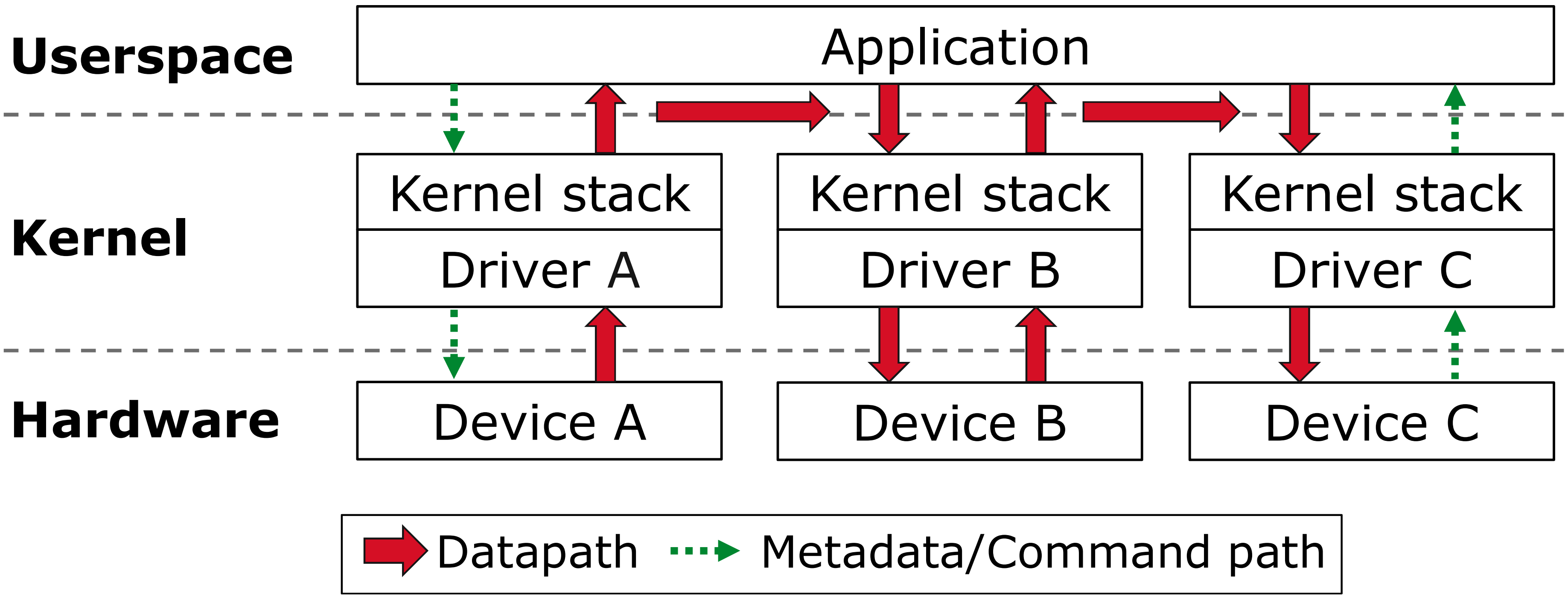
My solution:

Let’s use our intelligent server architecture!
“DCS: Device-Centric Server Architecture”

Three papers appeared in
- 2018 ACM/IEEE International Symposium on Computer Architecture (ISCA)
- 2017 ACM/IEEE International Symposium on Microarchitecture (MICRO)
- 2015 ACM/IEEE International Symposium on Microarchitecture (MICRO)

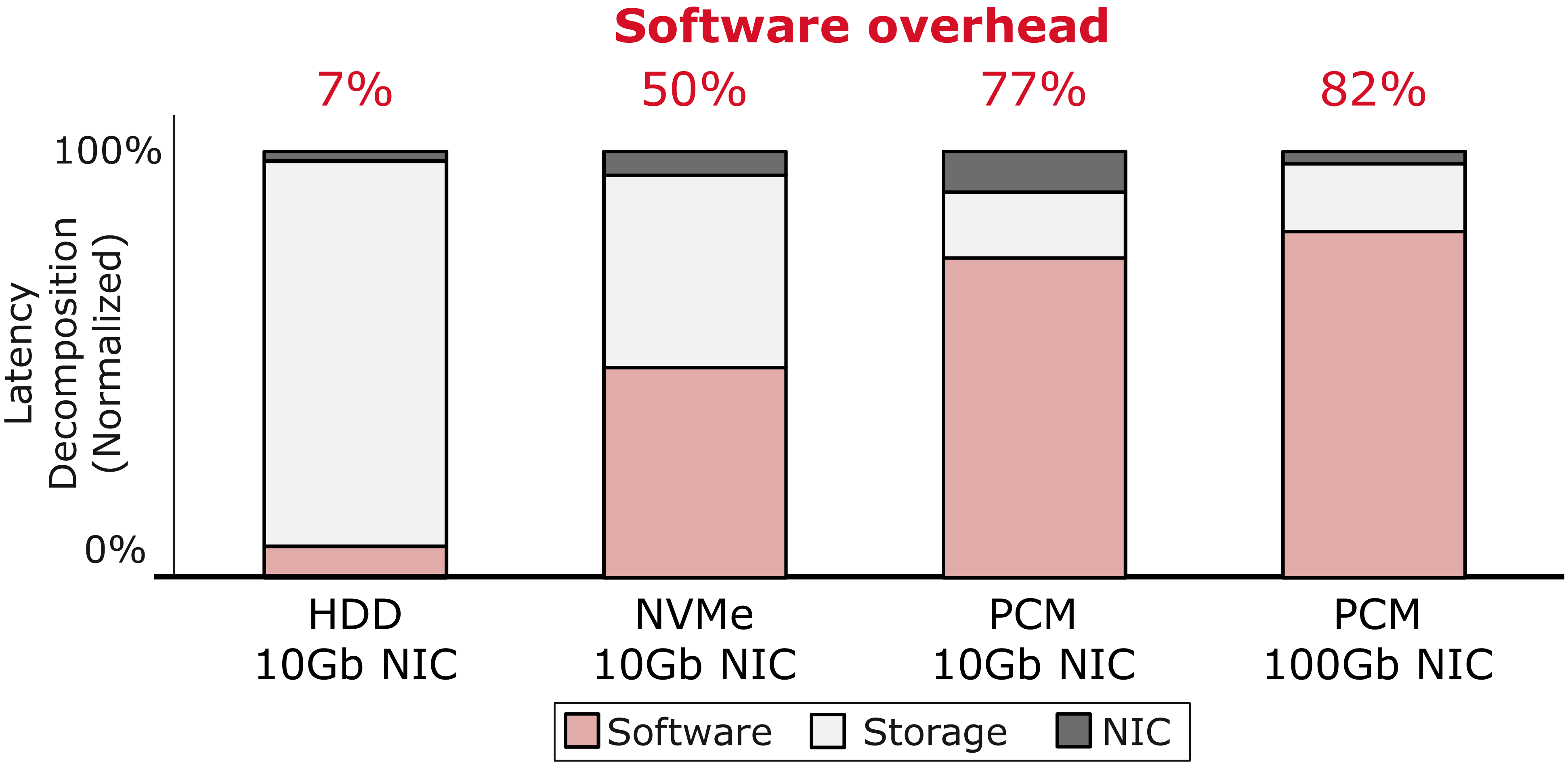
Existing servers do not work

- **Host-centric device management**
  - Host manages every device invocation
  - Frequent host-involved layer crossings
    - Increases latency and management cost



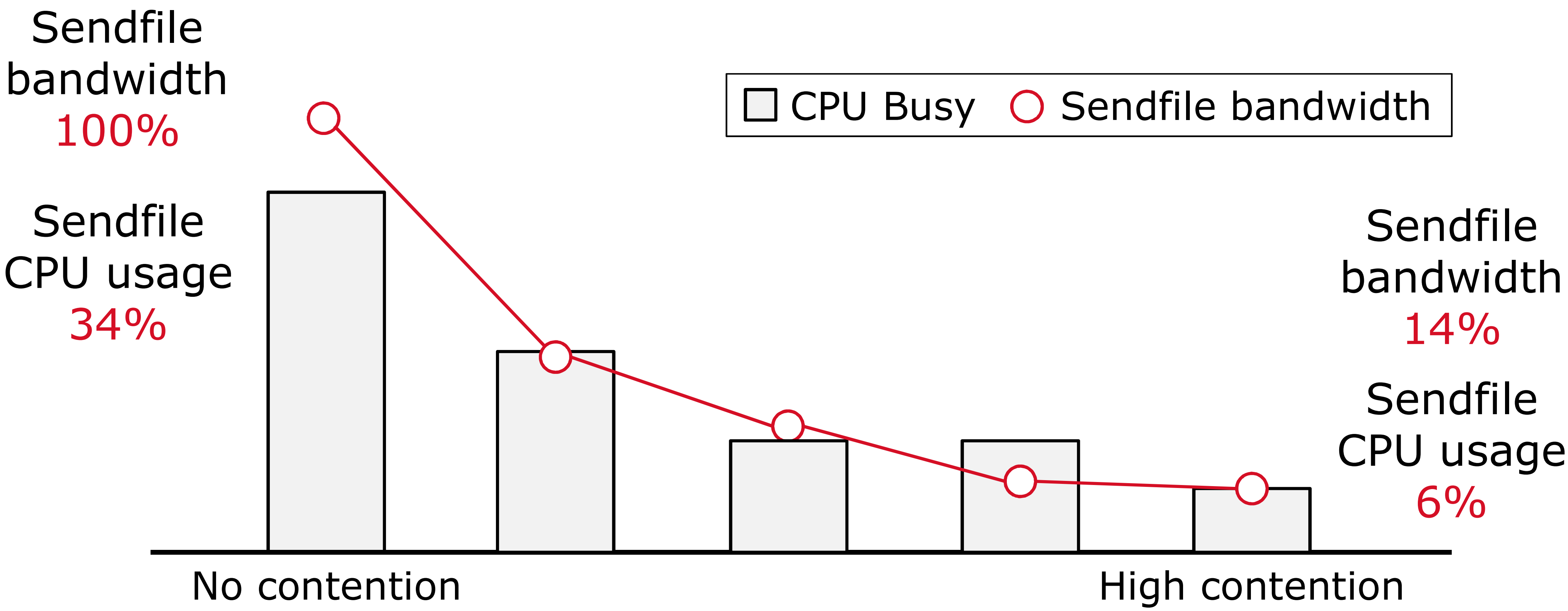
Latency: High software overhead

- **Single sendfile: Storage read & NIC send**
  - Faster devices, more software overhead



Cost: High host resource demand

- **Sendfile under host resource (CPU) contention**
  - Faster devices, more host resource consumption



*Measured from NVMe SSD/10Gb NIC

Limitations of existing work

- **Single-device optimization**
  - Do not address inter-device communication

    e.g., Moneta (SSD), DCA (NIC), mTCP (NIC), Arrakis (Generic)
- **Inter-device communication**
  - Not applicable for unsupported devices

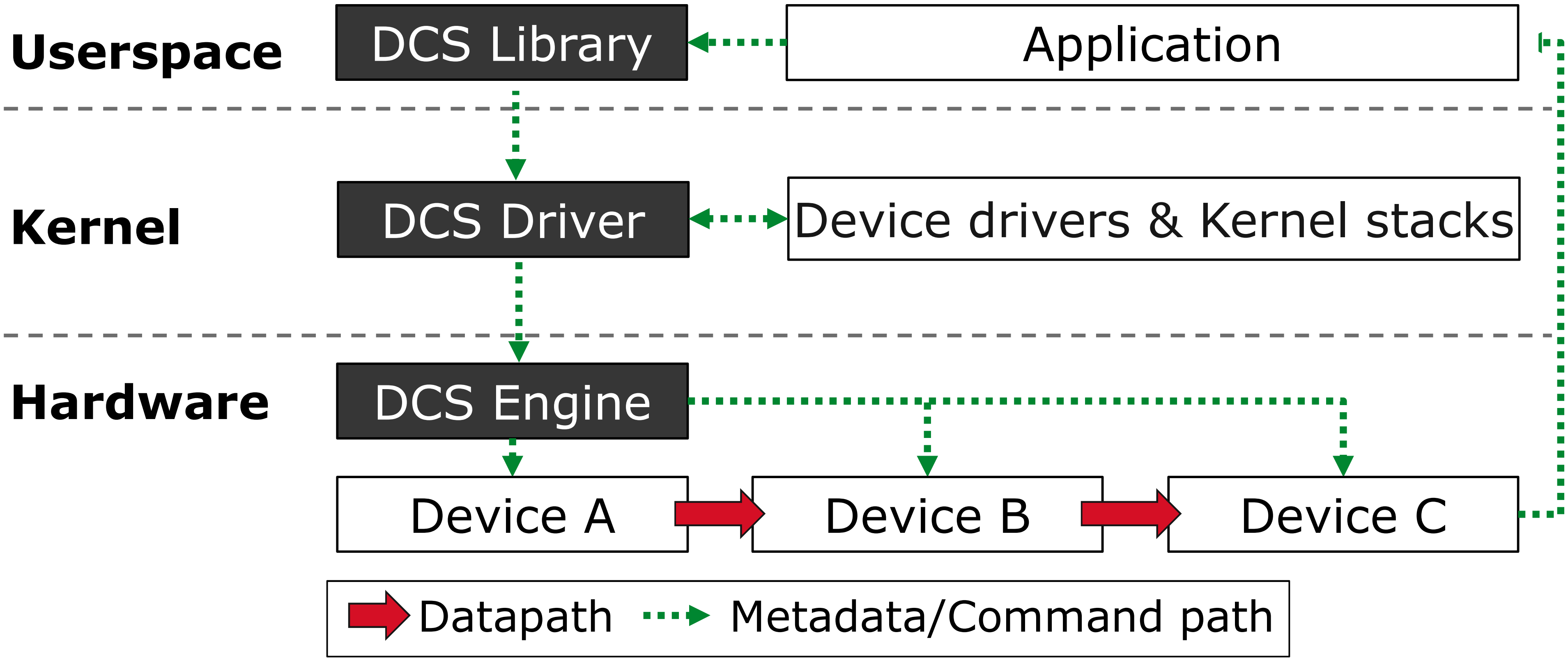
    e.g., GPUNet (GPU-NIC), GPUDirect RDMA (GPU-Infiniband)
- **Integrating devices**
  - Custom devices and protocols, limited applicability

    e.g., QuickSAN (SSD+NIC), BlueDBM (Accelerator–SSD+NIC)

Need for fast, scalable, and generic inter-device communication

Our solution: Device-Centric Server

- **Minimize host involvement & data movement**



Single command → Optimized multi-device invocation

DCS: Benefits

- **Selective, D2D transfer**
  - Faster data delivery, lower total operation latency

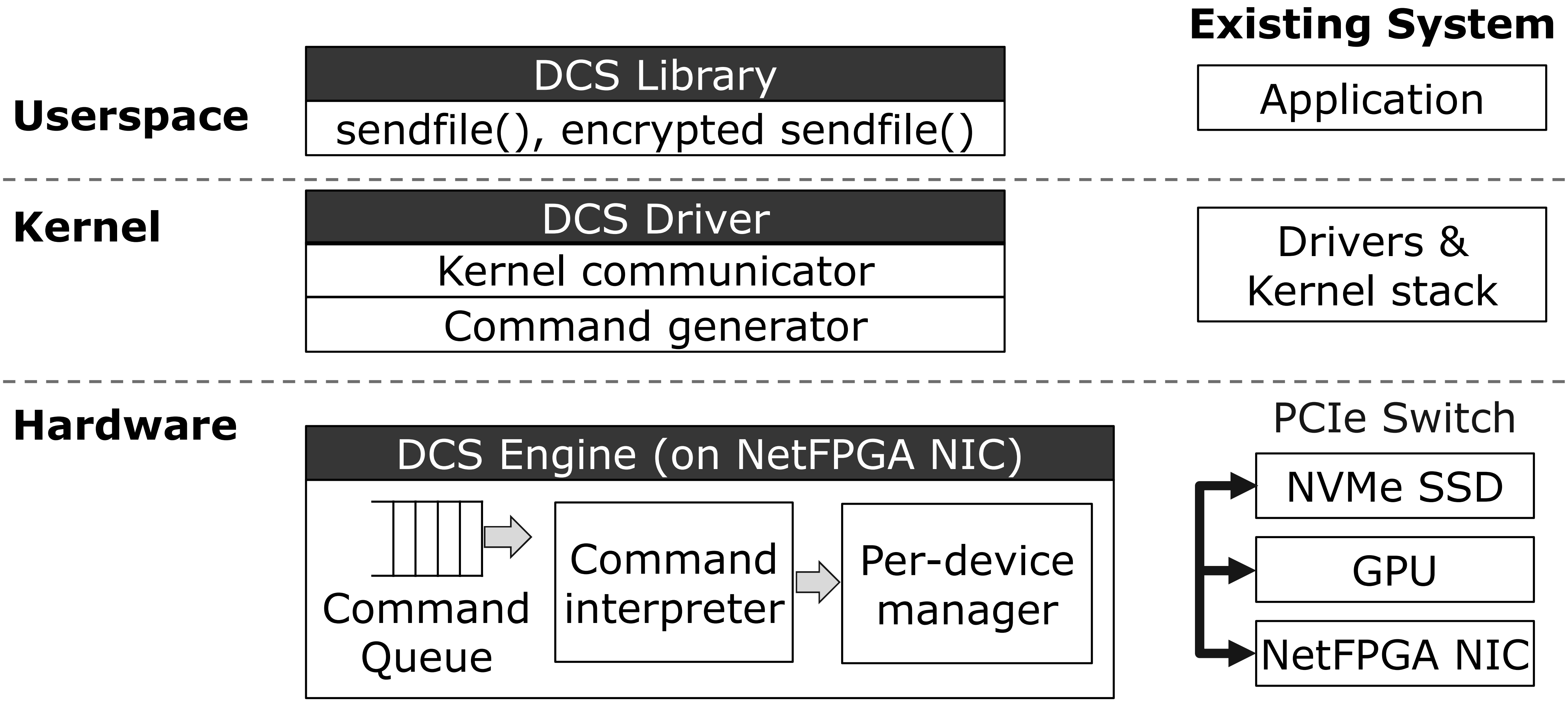
- **Better host performance/efficiency**
  - Resource/time spent for device management now available for other applications

- **High applicability**
  - Relies on existing drivers / kernel supports / interfaces
  - Easy to extend and cover more devices

Device-Centric Server Components

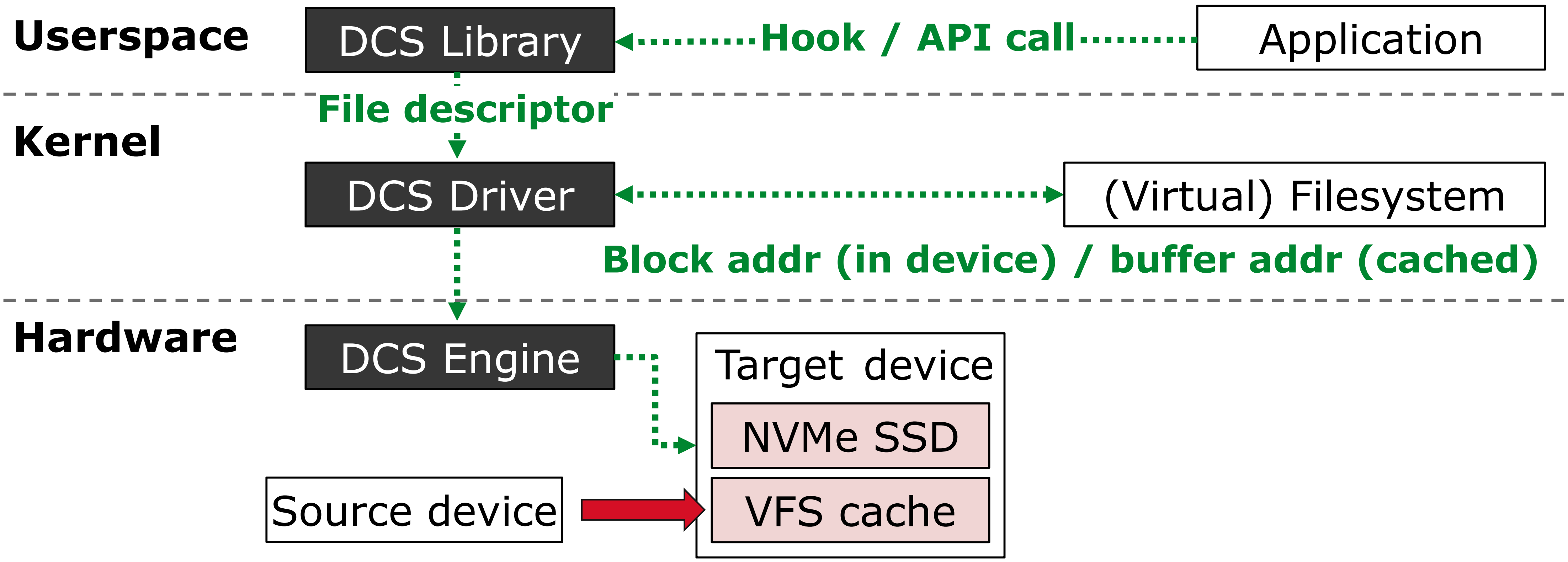
- **DCS Engine**
  - A custom HW device to selectively connect devices
- **DCS drivers**
  - Convert commodity devices to work with DCS engines
- **DCS library**
  - OS library to hook with the existing system calls
- **DCS applications**
  - Applications developed or tuned for DCS systems

DCS: Architecture overview



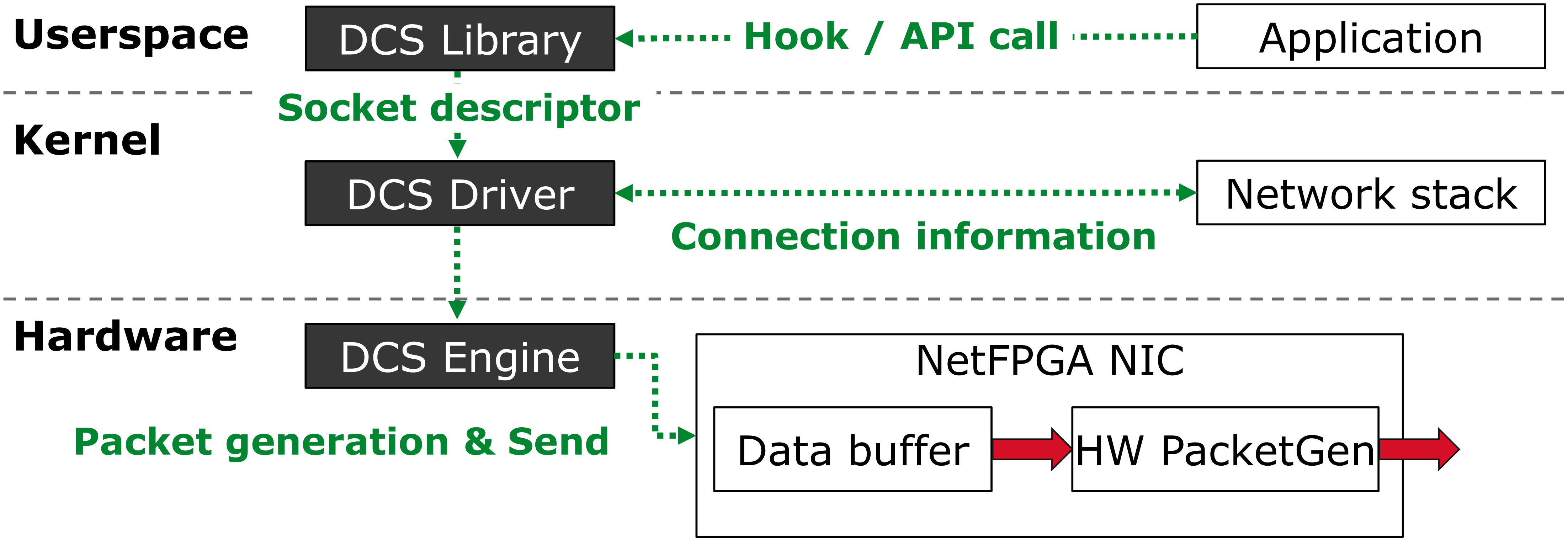
Fully compatible with existing systems

Communicating with storage



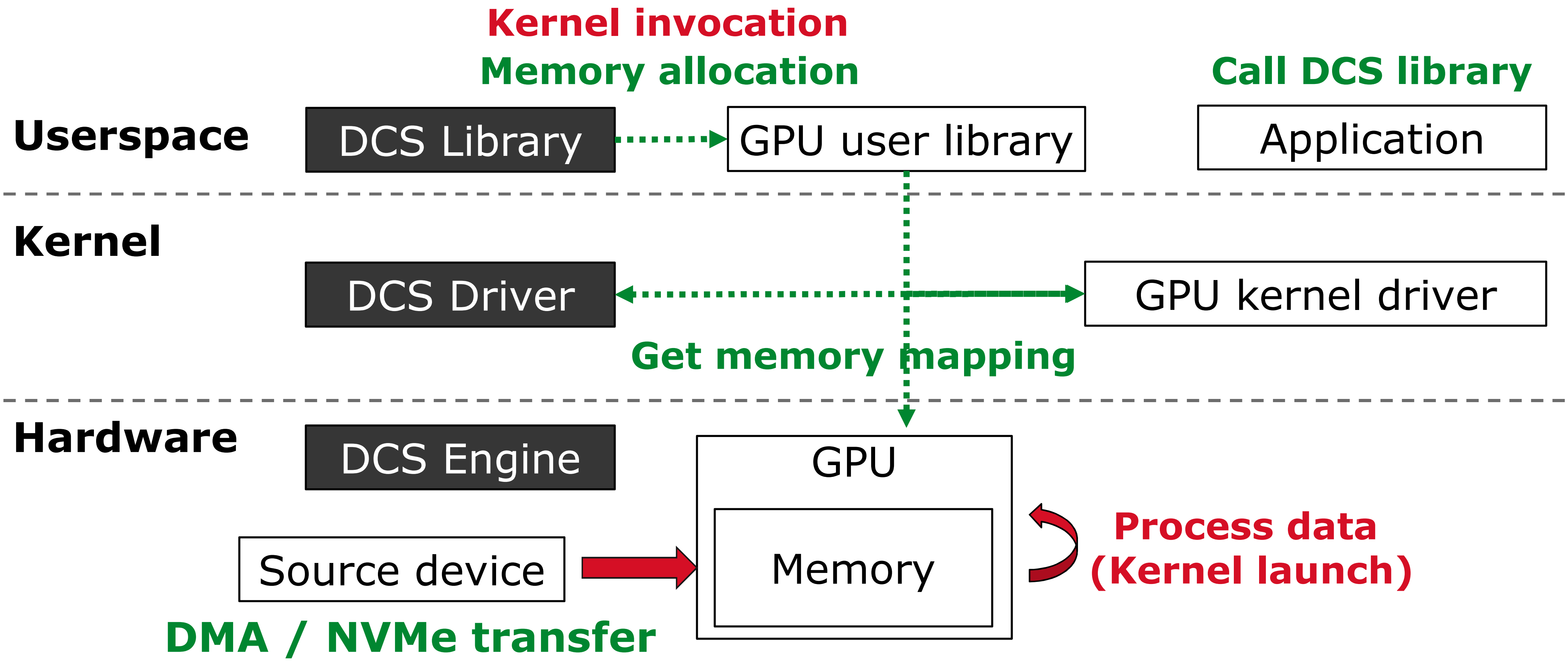
Data consistency guaranteed

Communicating with network interface



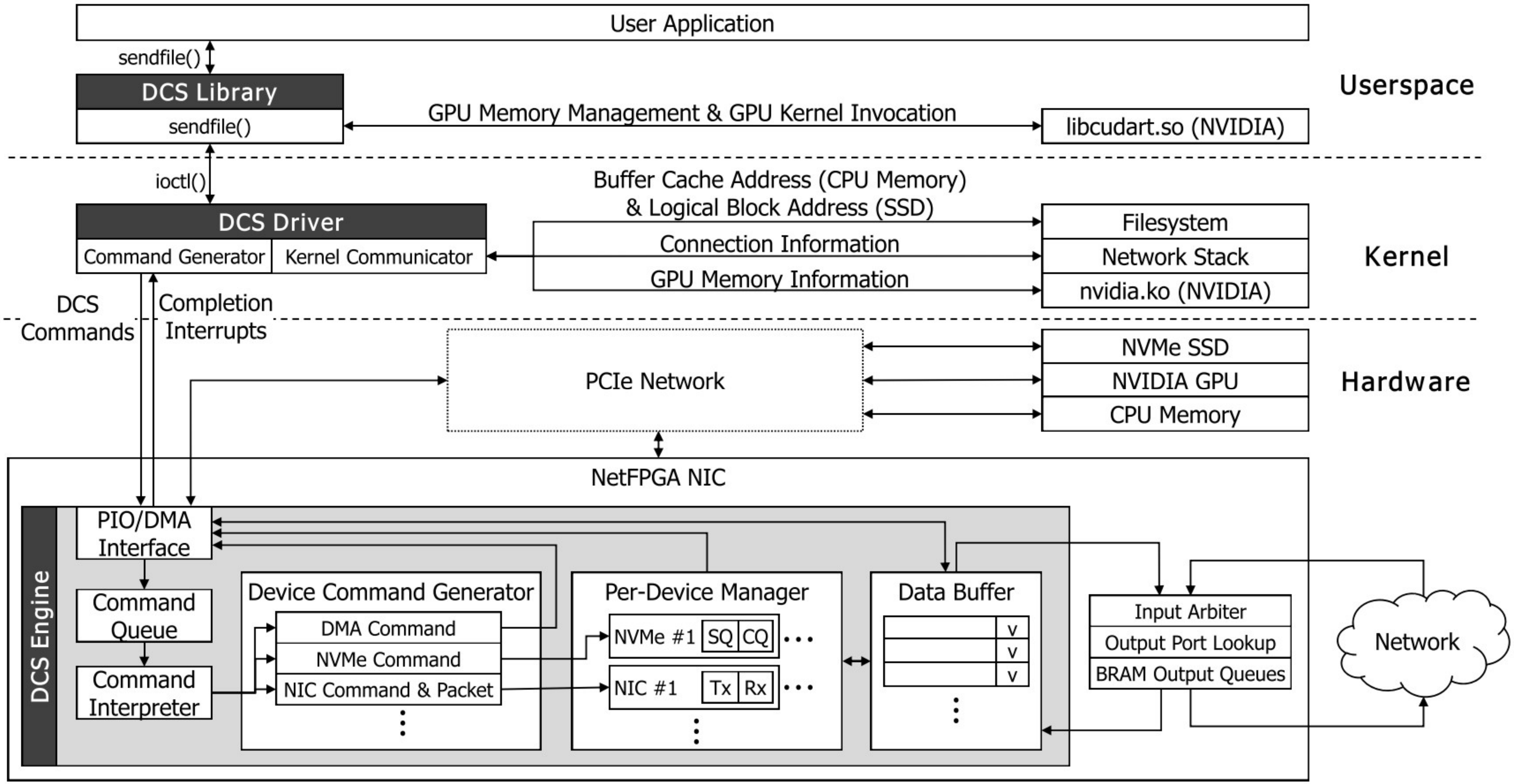
HW-assisted packet generation

Communicating with accelerator



Direct data loading without memcpy

DCS sytem in a big picture!



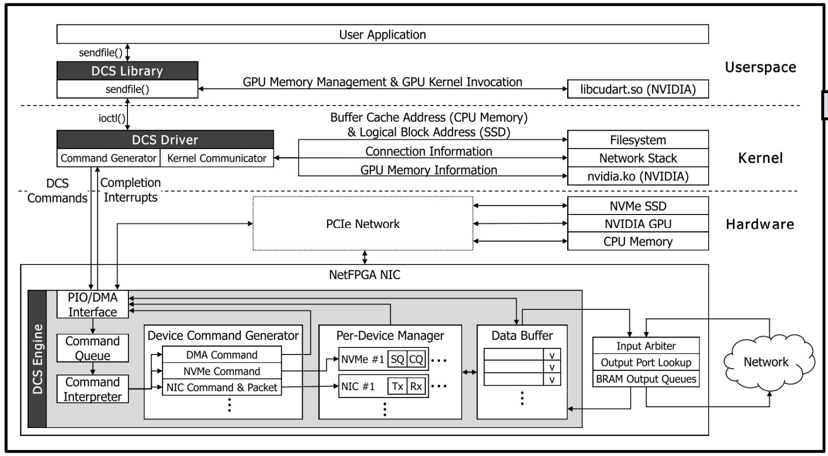
Experimental setup

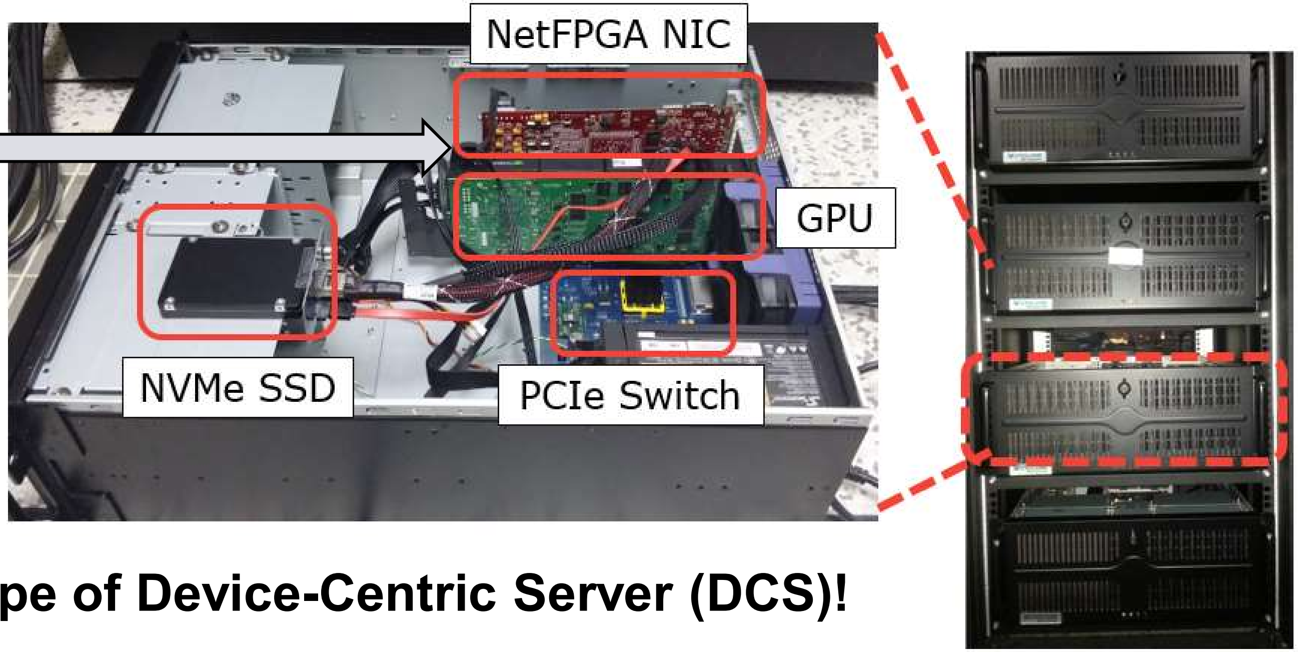
- **Host: Power-efficient system**
  - Core 2 Duo @ 2.00GHz, 2MB LLC
  - 2GB DDR2 DRAM

- **Device: Off-the-shelf emerging devices**
  - Storage: Samsung XS1715 NVMe SSD
  - NIC: NetFPGA with Xilinx Virtex 5 (up to 1Gb bandwidth)
  - Accelerator: NVIDIA Tesla K20m
  - Device interconnect: Cyclone Microsystems PCIe2-2707 (Gen 2 switch, 5 slots, up to 80Gbps)

DCS prototype implementation

- **Our 4-node DCS prototype**
  - Can support many devices per host



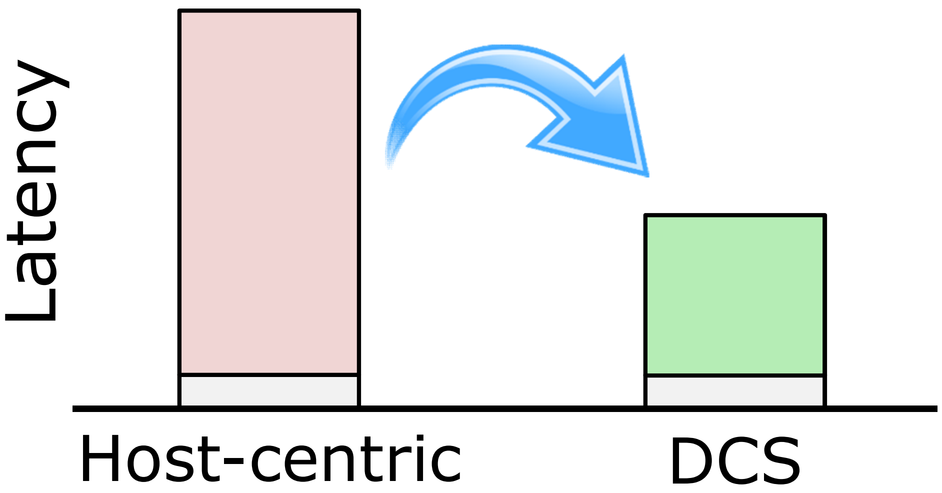


A working prototype of Device-Centric Server (DCS)!

Reducing device utilization latency

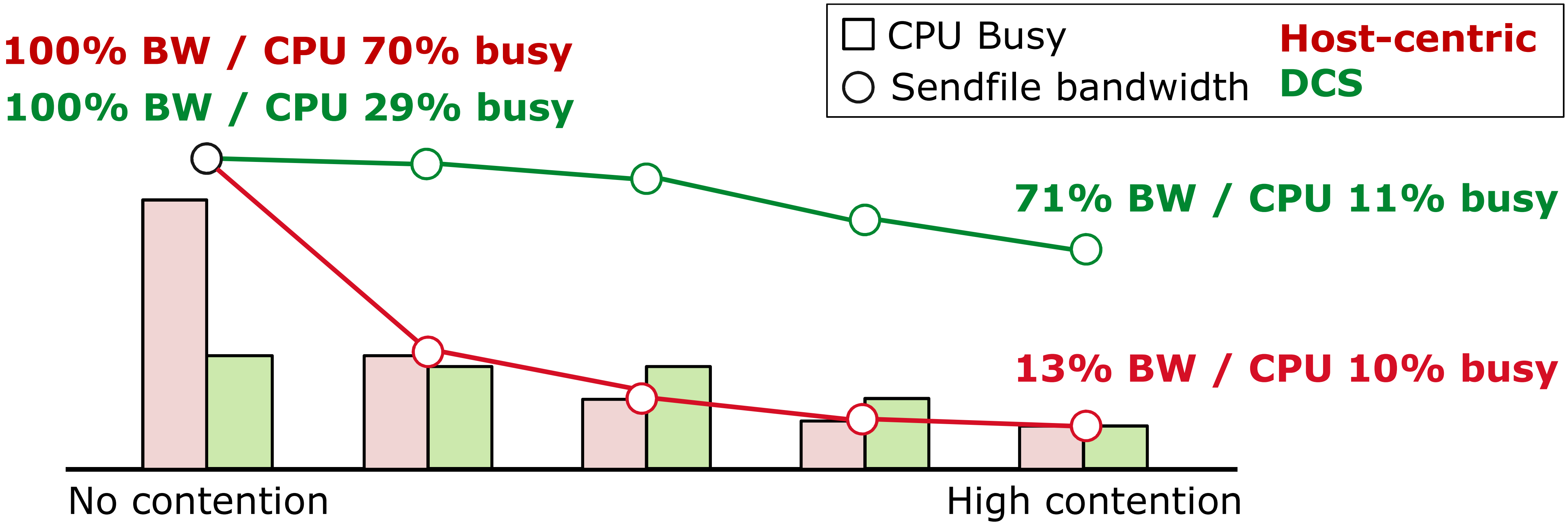
- **Single sendfile: Storage read & NIC send**
  - Host-centric: Per-device layer crossings
  - DCS: Batch management in HW layer

2x latency improvement
(with low-latency devices)



Host-independent performance

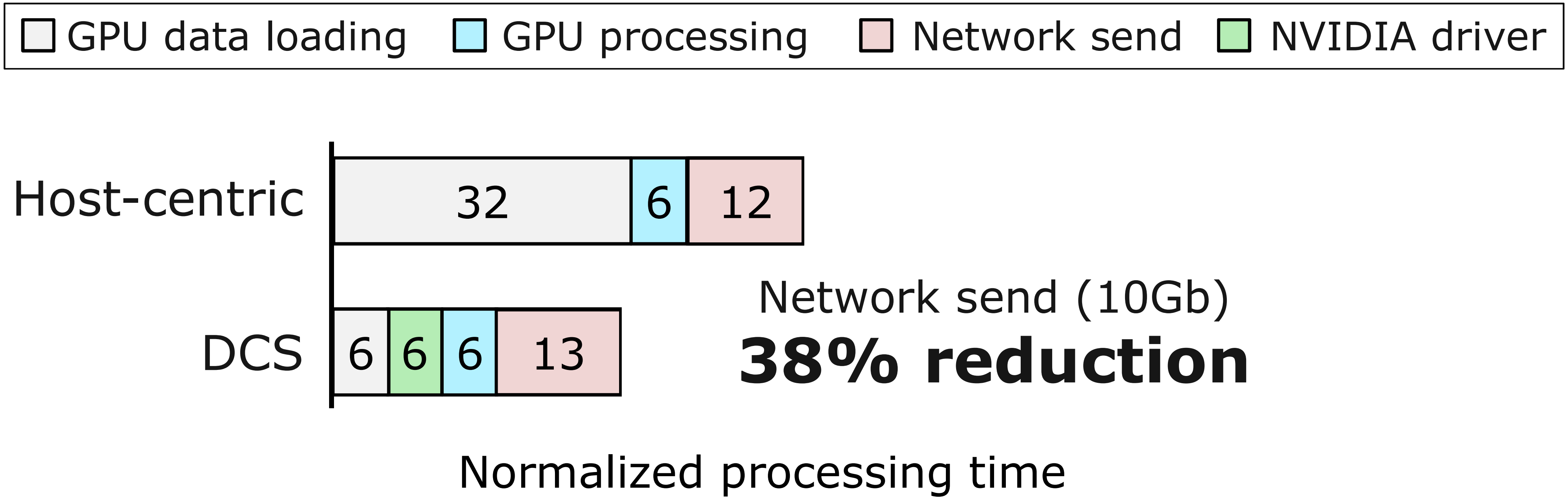
- **Sendfile under host resource (CPU) contention**
  - Host-centric: host-dependent, high management cost
  - DCS: host-independent, low management cost



High performance even on weak hosts

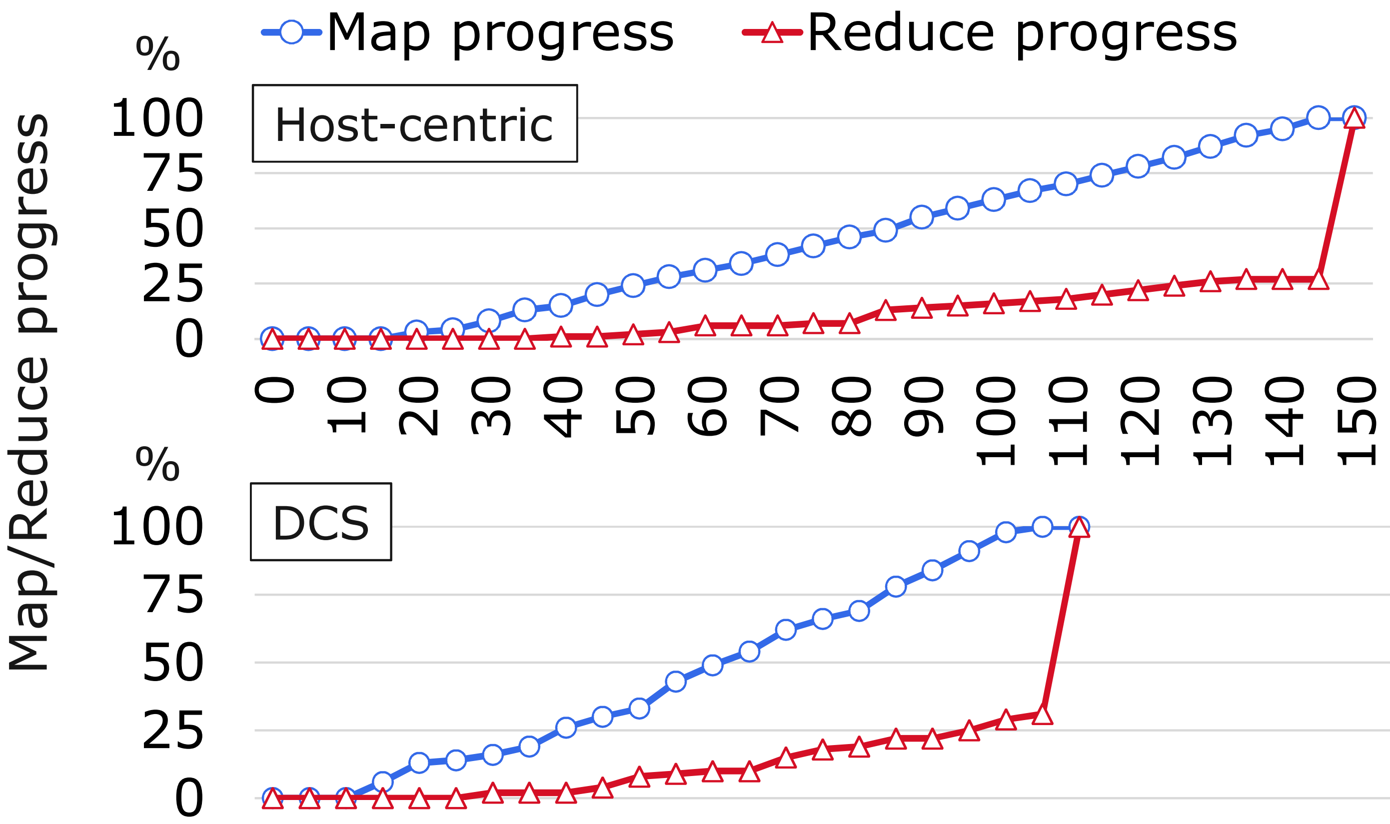
Multi-device invocation

- **Encrypted sendfile (SSD → GPU → NIC, 512MB)**
  - DCS provides much efficient data movement to GPU
  - Current bottleneck is NIC (1Gbps)



Real-world workload: Hadoop-grep

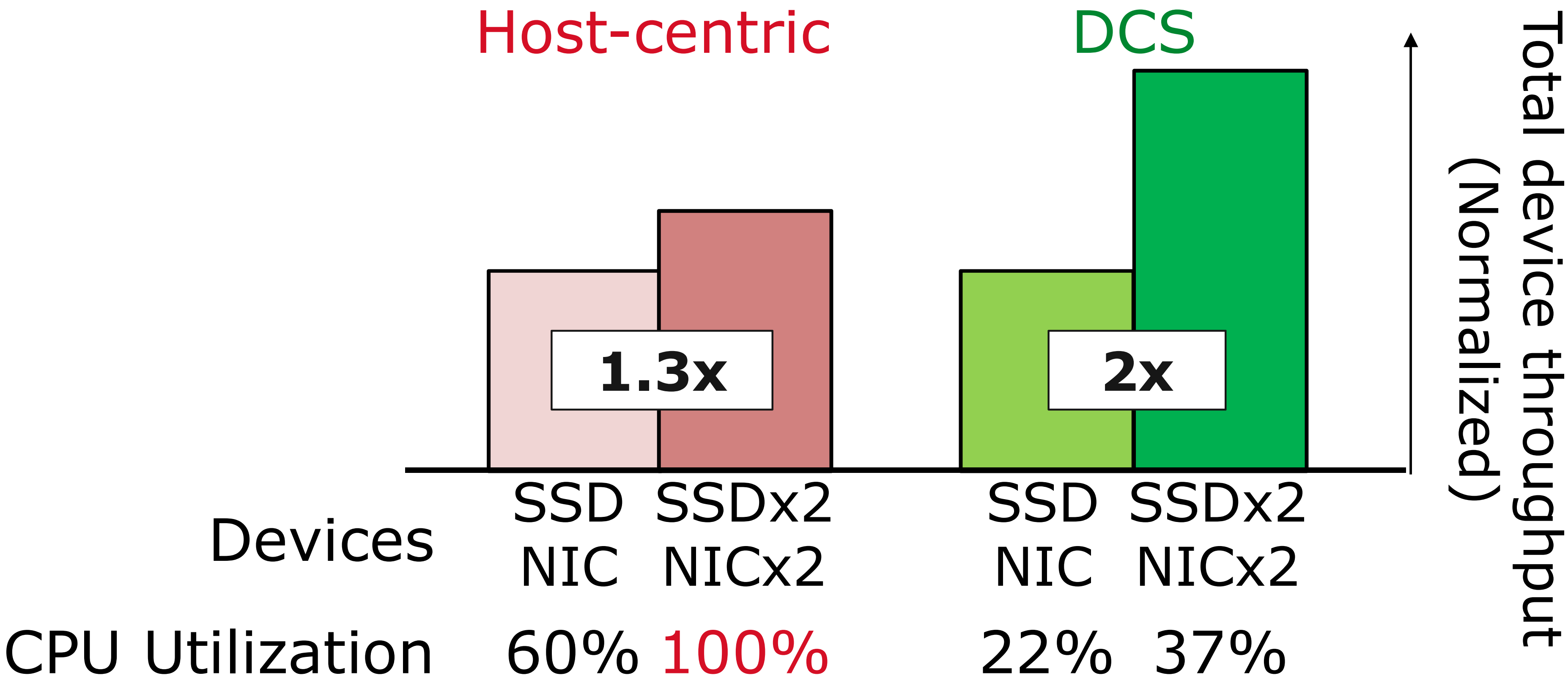
- **Hadoop-grep (10GB)**
  - Faster input delivery & smaller host resource consumption



40% faster processing

Scalability: More devices per host

- Doubling # of devices in a single host



Scalable many-device support

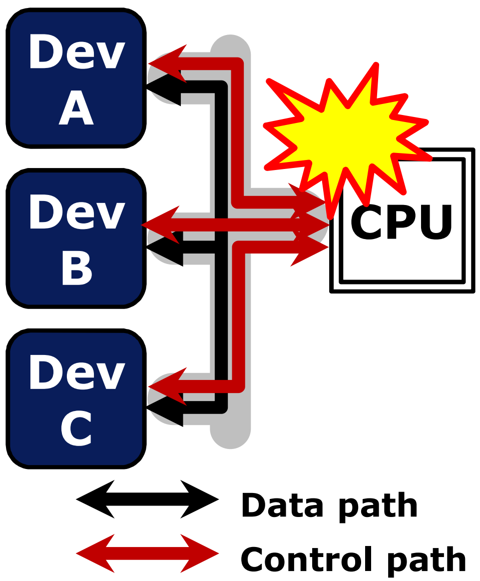
1^{st} prototype in 2015 [MICRO 2015]

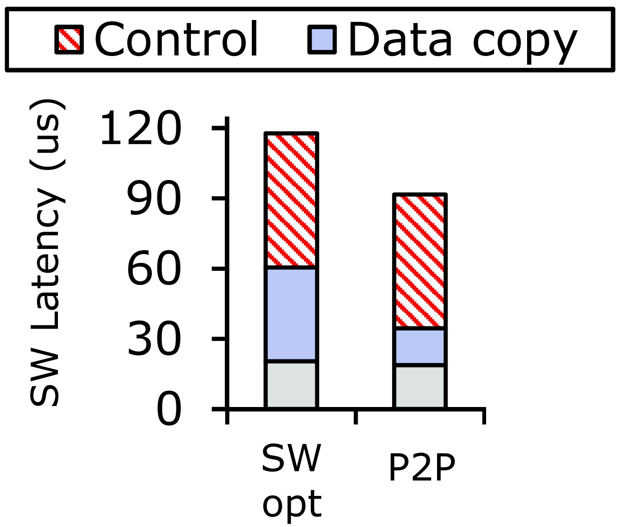
- A new server architecture: **DCS!**
  - Device latency reduction: ~25%
  - Host resource savings: ~61%
  - Hadoop speed improvement: ~40%

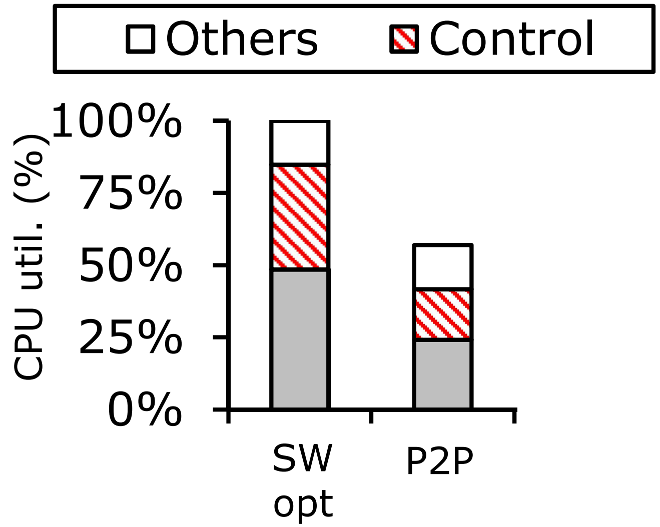
Wait. We can do even better!

Limitations of Existing D2D Comm.

- **P2P communication**
  - Direct data transfers through PCI Express → **D2D comm.**
  - **Slow, high-overhead control path becomess a killer**

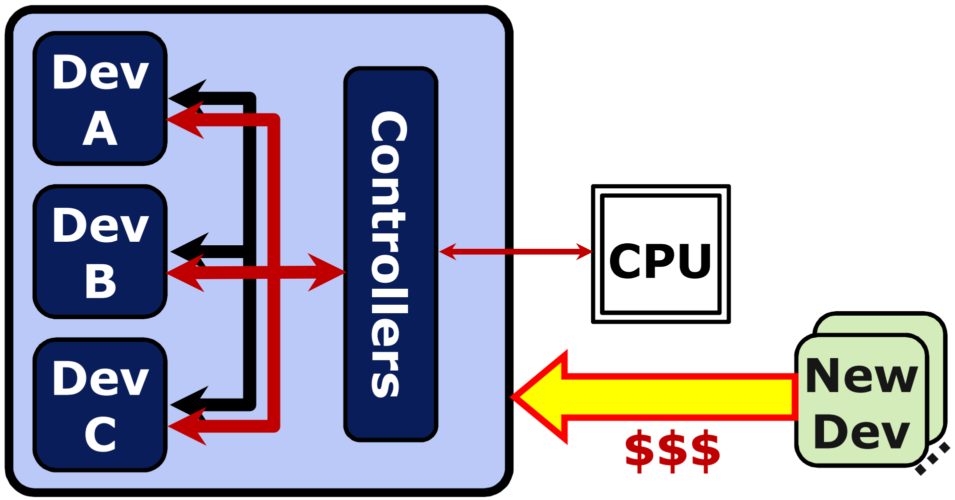






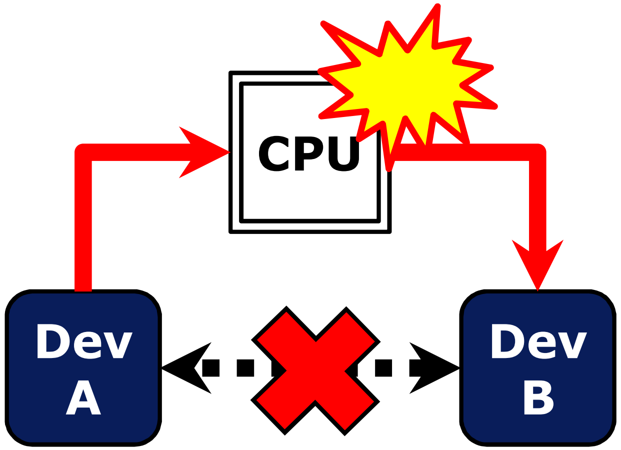
Limitations of Existing D2D Comm.

- **Integrated devices**
  - Integrating heterogeneous devices → **D2D comm.**
  - **Fast data & control transfers**
  - **Fixed and inflexible aggregate implementation**



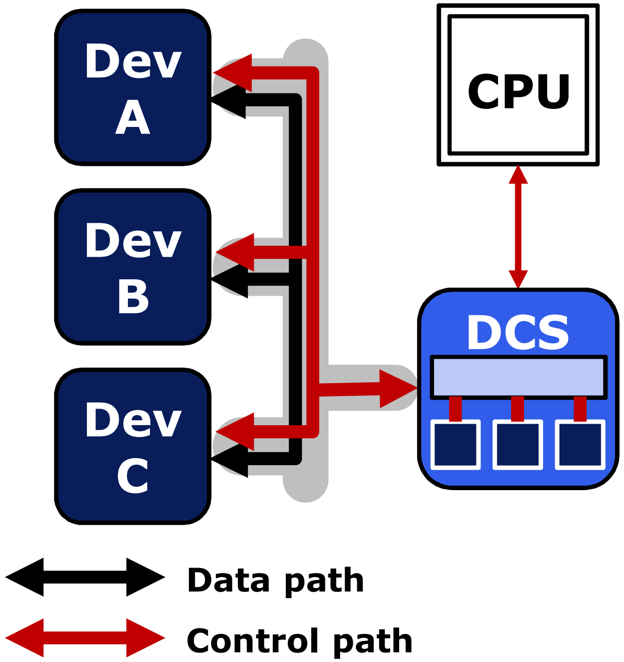
Limited Performance Potential

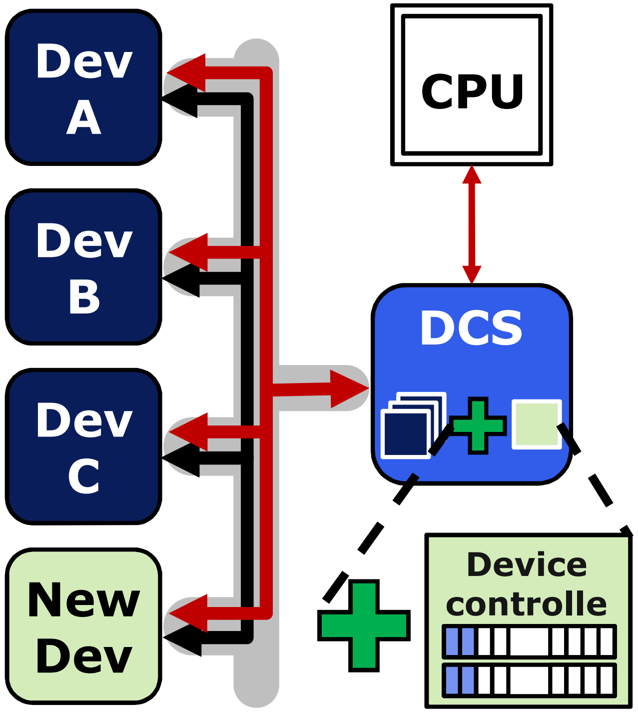
```
while (true) {
    rc_recv = recv(fd_sock, buffer, recv_size, 0);
    if (rc_recv <= 0) break;
    processing(&md_ctx, buffer, recv_size);
    rc_write = write(fd_file, buffer, recv_size);
    …
}
```



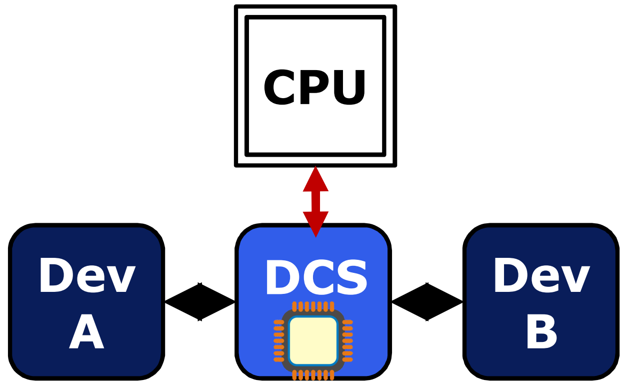
- **"Intermediate" processing between device ops**
  - Prevent applications from using direct D2D comm.
  - Cause host-side resource contention (CPU and memory)

DCS-v2: Key Ideas & Benefits






```
void ssd_to_nic()
{
  get_from_ssd(&data);
  process_in_HDC(&data);
  write_to_nic(&data);
}
```



Optimized dev. control

⇒ **Faster & scalable communication**

Generic dev. interfaces

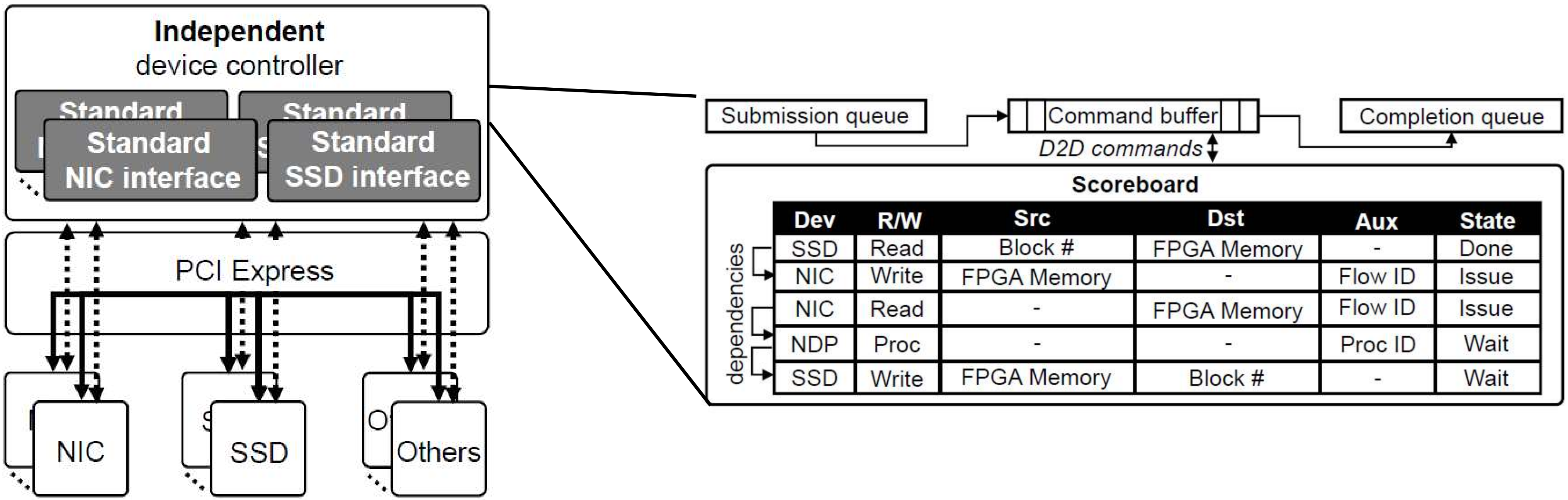
⇒ **Higher flexibility**

Near-device processing

⇒ **Higher applicability**

DCS-v2: (1) standard device interfaces

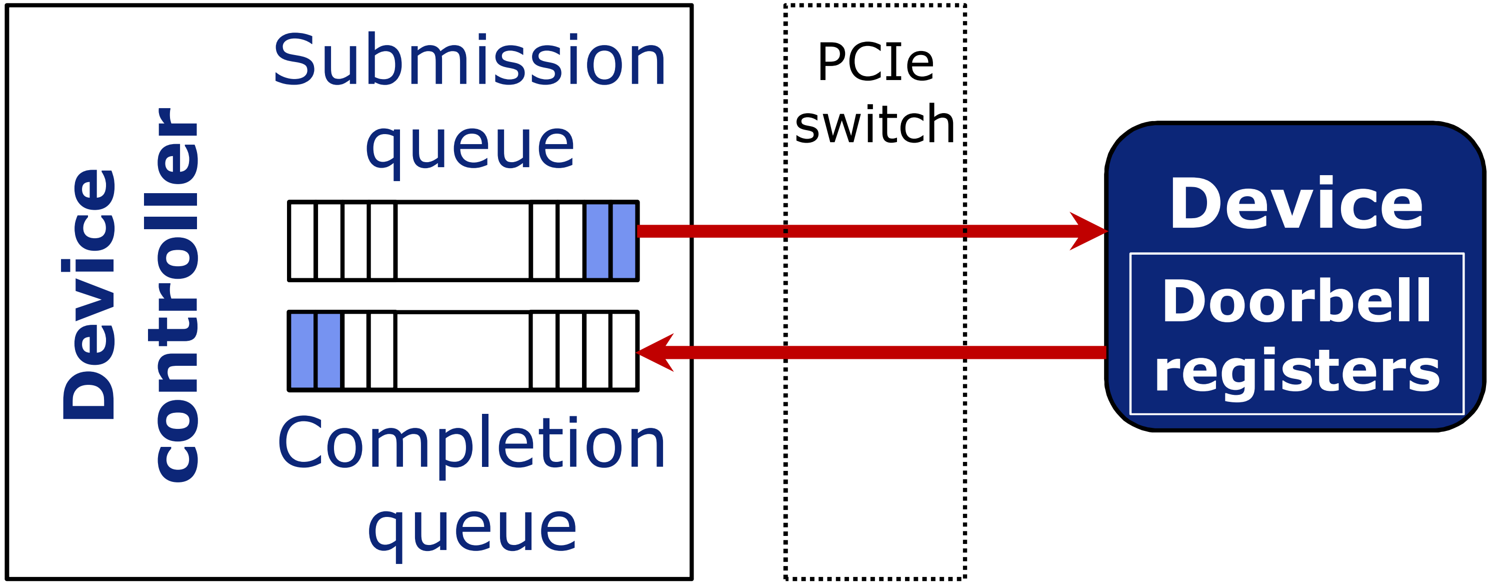
- **Standard interfaces in DCS Engine**
  - Based on “scoreboard” with independent queues
    - Keep track of (src, dst, commands, status)



Standard interface provided by FPGA

DCS-v2: (2) HW-based fast D2D “control”

- **Device ctrl functions in DCS Engine**
  - Bypass OS as much as possible
    - Handle kernel-dependent functions (e.g., recvfile)



Both data and control managed by FPGA

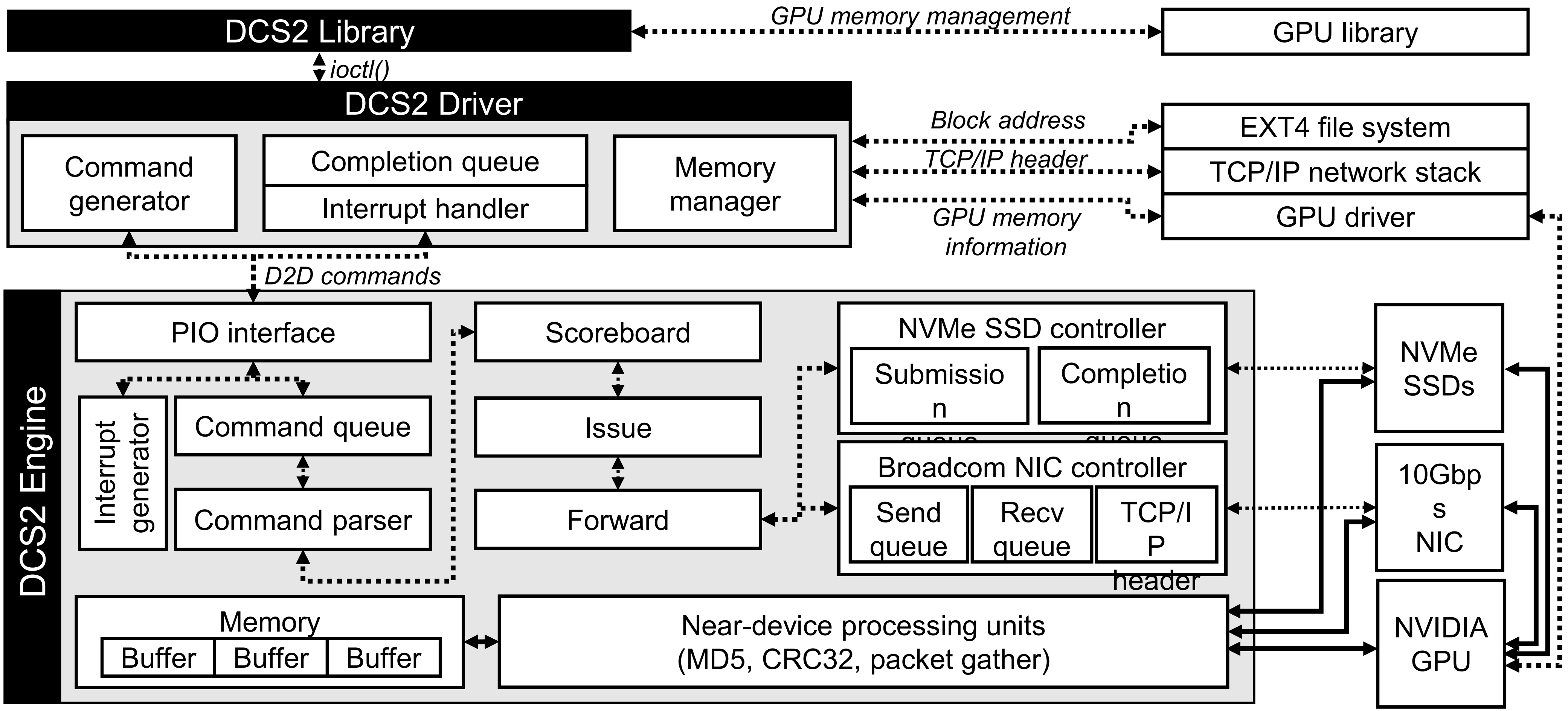
DCS-v2: (3) Near-Device Processing (NDP)

```
MD5_Init(&md_ctx);
while (true) {
    rc_recv = recv(fd_sock, buffer, recv_size, 0);
    if (rc_recv <= 0) break;
    MD5_Update(&md_ctx, buffer, recv_size);
    rc_write = write(fd_file, buffer, recv_size);
    if (recv_size != rc_write) {
        break;
    }
}
MD5_Final(md_res, &md_ctx);
```

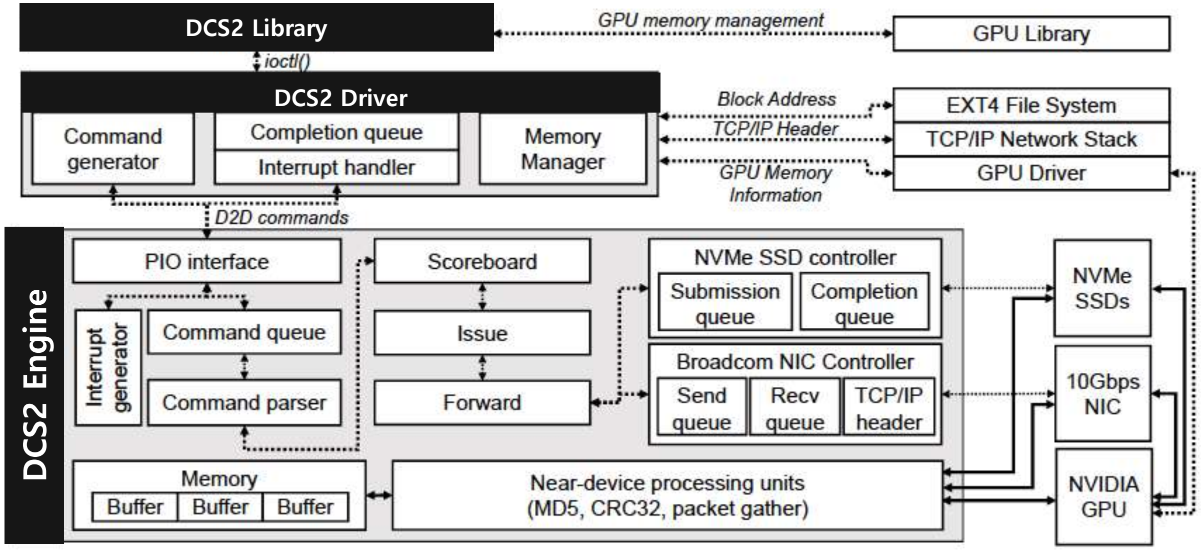
- **Intermediate processing (MD5_Update) between device Ops**
- **CPU- and memory-intensive routines in existing applications**
  - Prevent applications from using direct D2D communications
  - Cause host-side resource contention (CPU and memory)

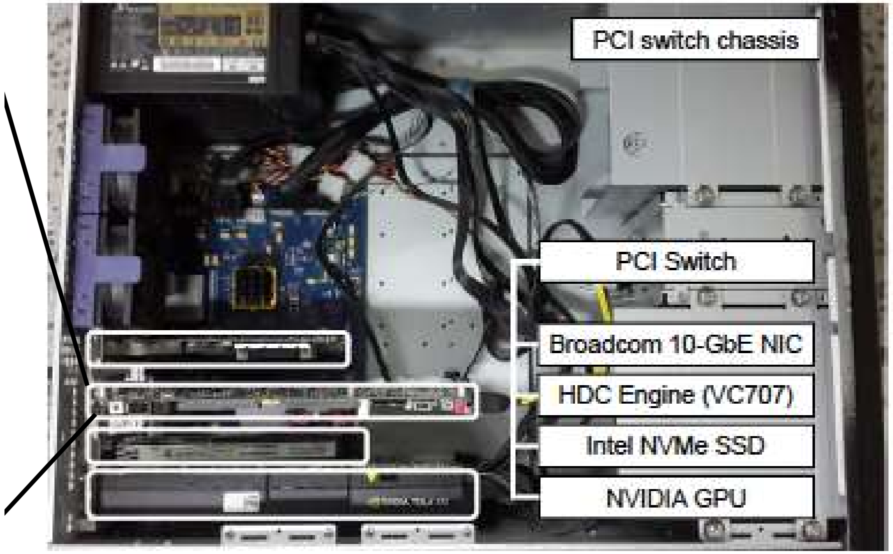
Intermediate computation by FPGA

A new DCS system in a big picture!



DCS-v2: a working prototype now





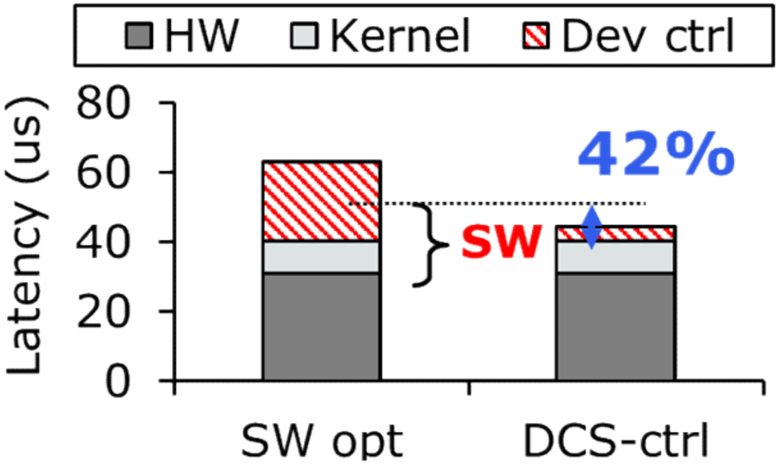
- **Off-the-shelf emerging devices**
  - Storage: Intel 750 Series SSD 400GB
  - NIC: Broadcom Corporation NetXtreme II BCM57711(10Gb)
  - Accelerator: NVIDIA Tesla K20m
  - PCIe switch: Cyclone Microsystems PCIe2-2707 (Gen2)
  - FPGA: Xilinx Virtex 7 VC 707 board

[ISCA'18]

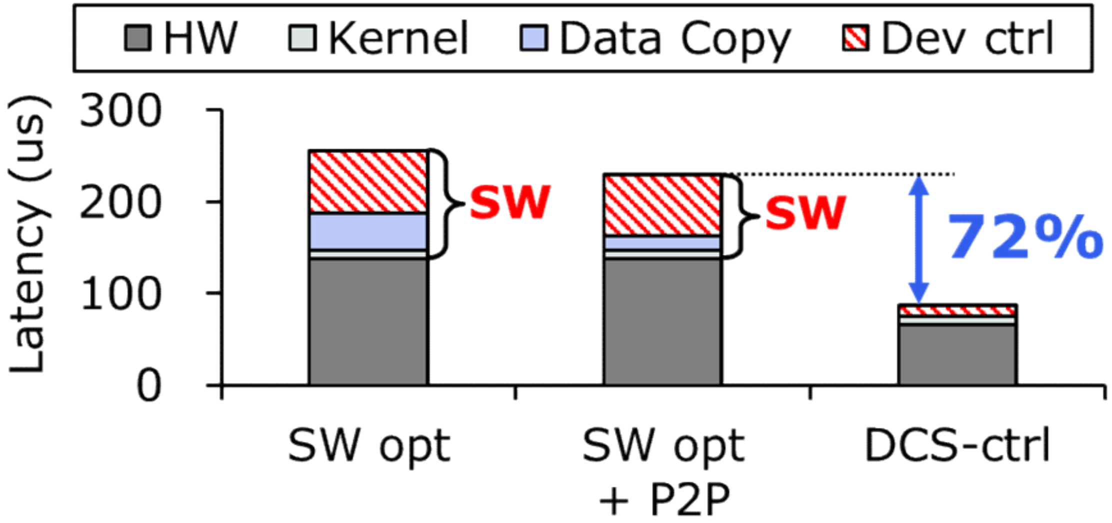
HPCS High Performance Computer System Lab

Performance: Low D2D Latency

- `encrypted_sendfile()`: **SSD → hash → NIC**
  - SW opt (+P2P): frequent boundary crossings, complex software
  - DCS-ctrl: less crossings, hardware-based device control



without processing

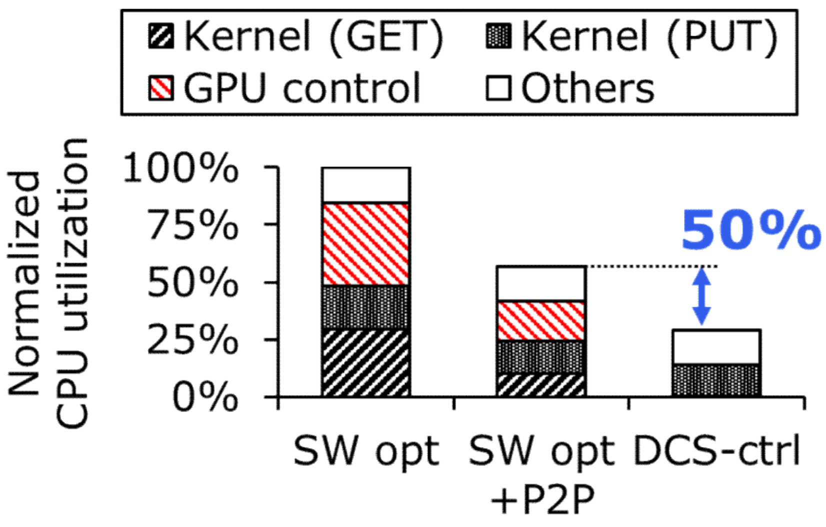


with processing (AES256)

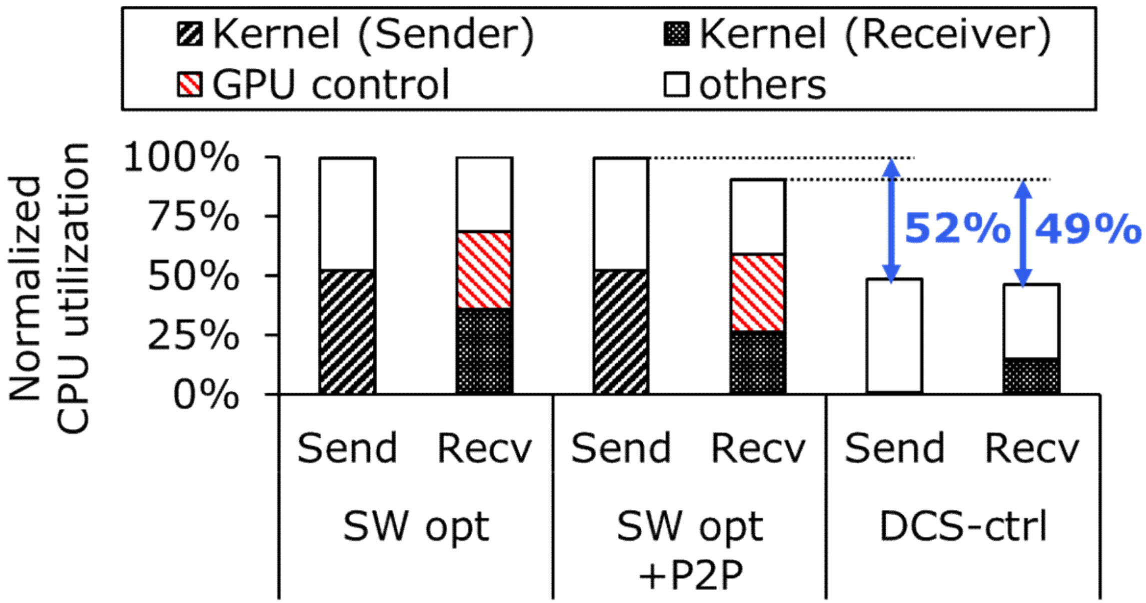
Significant performance boost!

Utilization: CPU become silent

- **Swift & HDFS workloads**
  - Offload device control & data transfers to hardware



Swift

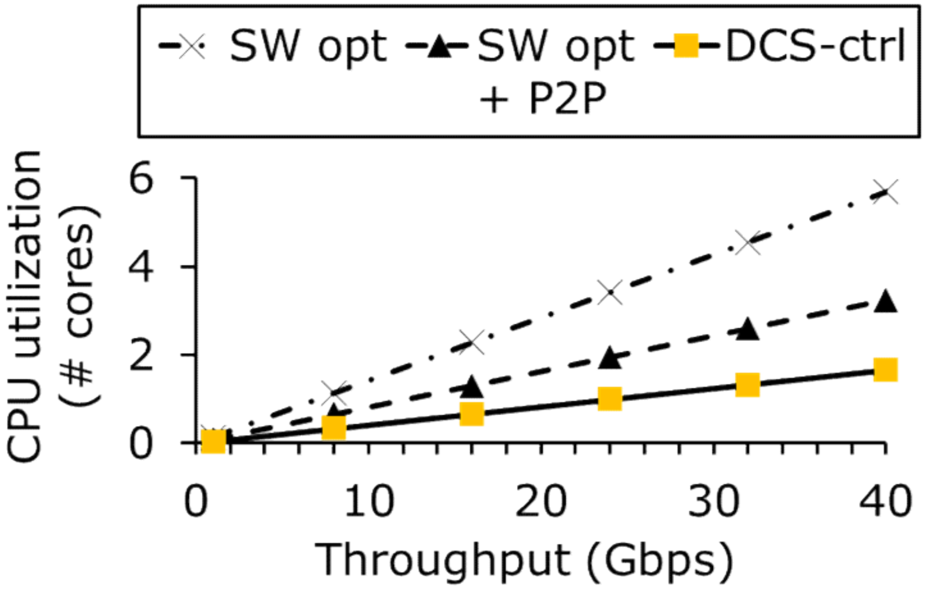


HDFS

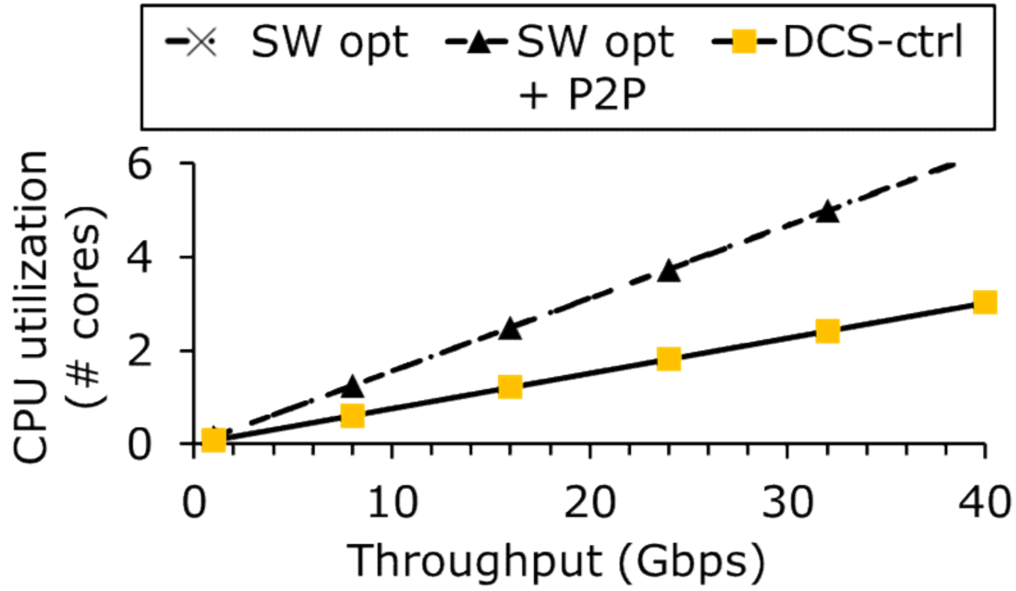
Significant host CPU saving!

Scalability: support many devices

- **Swift & HDFS workloads**
  - More CPU-efficient ➔ support more high-performance devices



Swift



HDFS

Significant scalability boost!

What we are doing now!

- New applications require new systems!

We are currently building

(1) Scale-up DCS engine

(2) Scale-out DCS engine

(3) DCS-enabled AI processing
(e.g., fast training, real-time processing)

Question?

Thank You!

Jangwoo Kim
e-mail: jangwoo@snu.ac.kr
https://hpcs.snu.ac.kr/~jangwoo