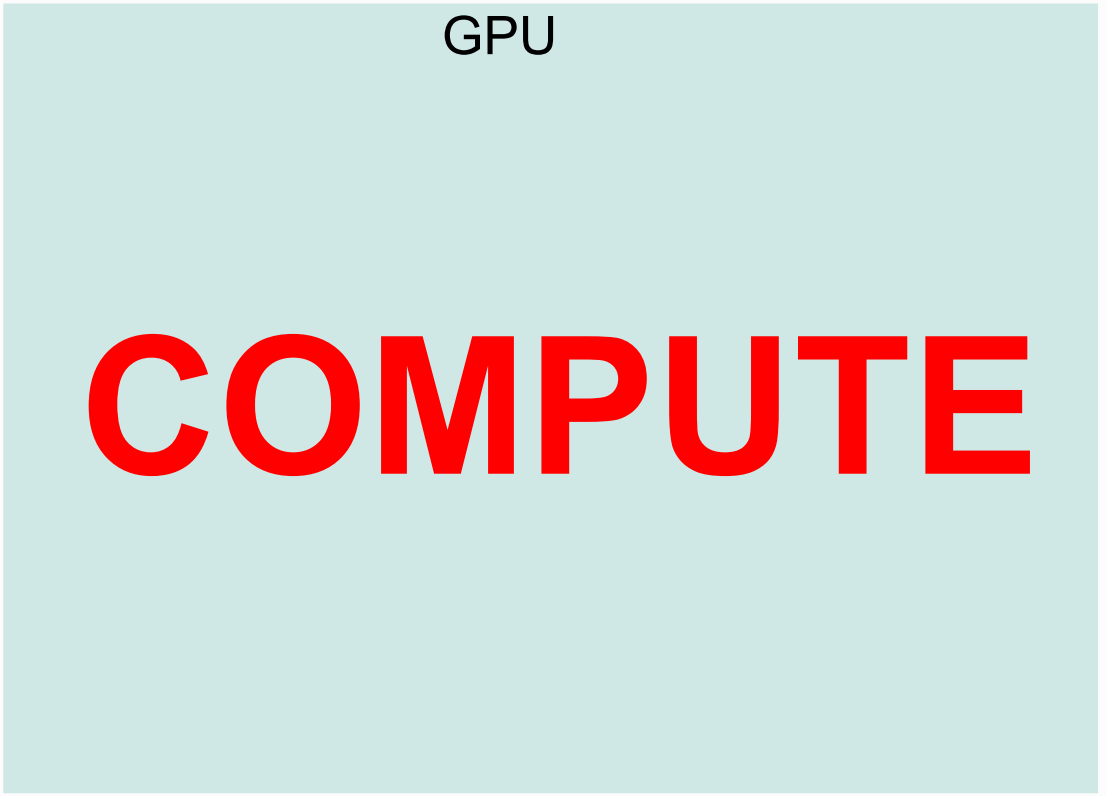
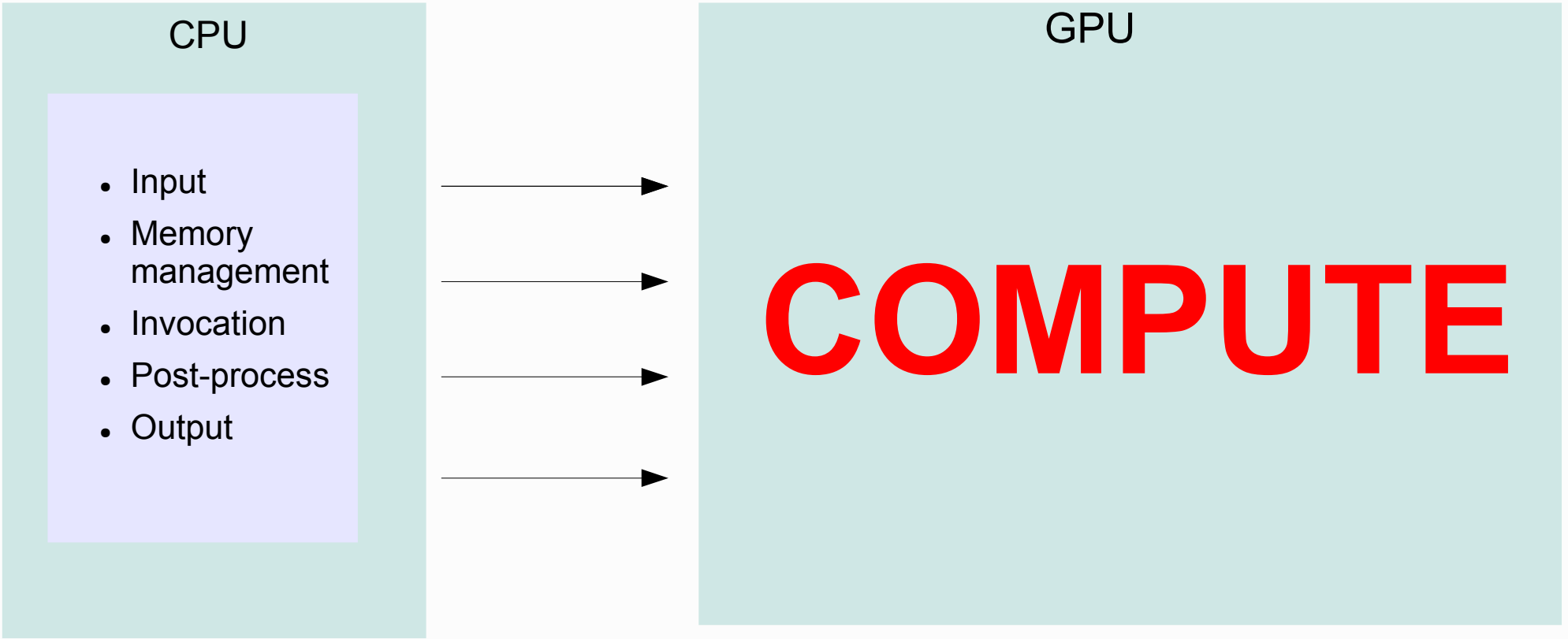
Set GPUs free!
I/O support for GPUs

Mark Silberstein, Emmett Witchel
UT Austin

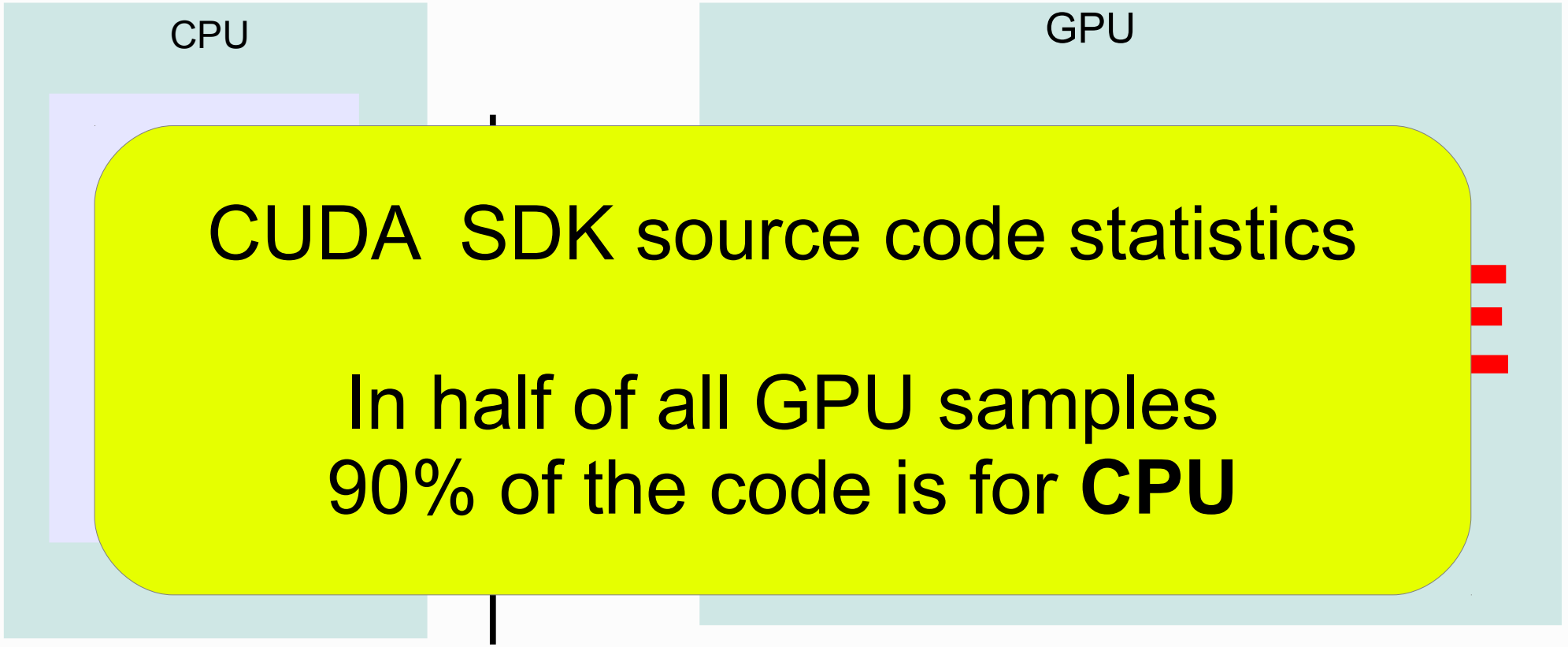
We think GPU....



We mean GPU**+CPU**



For every line of GPU code there are **10** lines of CPU code



Why? GPU kernels are isolated

CPU

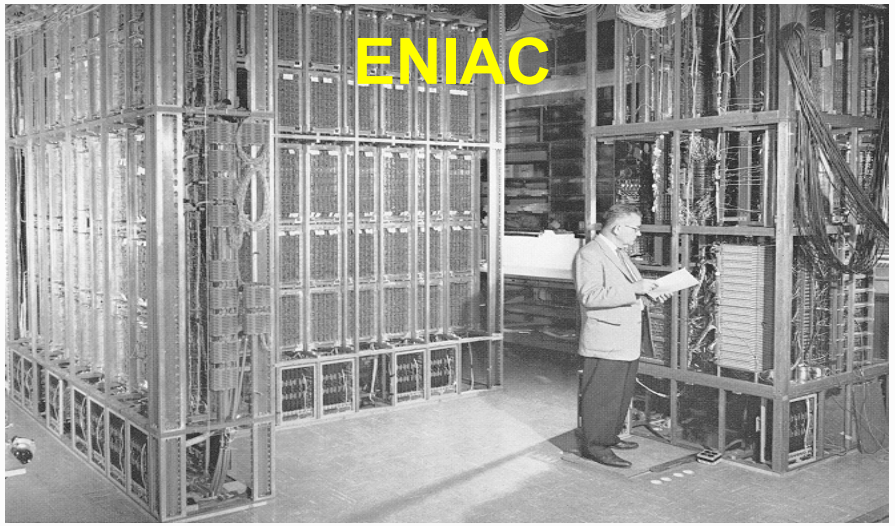
- Input
- Memory management
- Invocation
- Post-process
- Output

GPU

COMPUTE

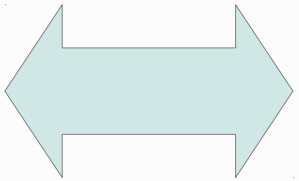
Find similarities...

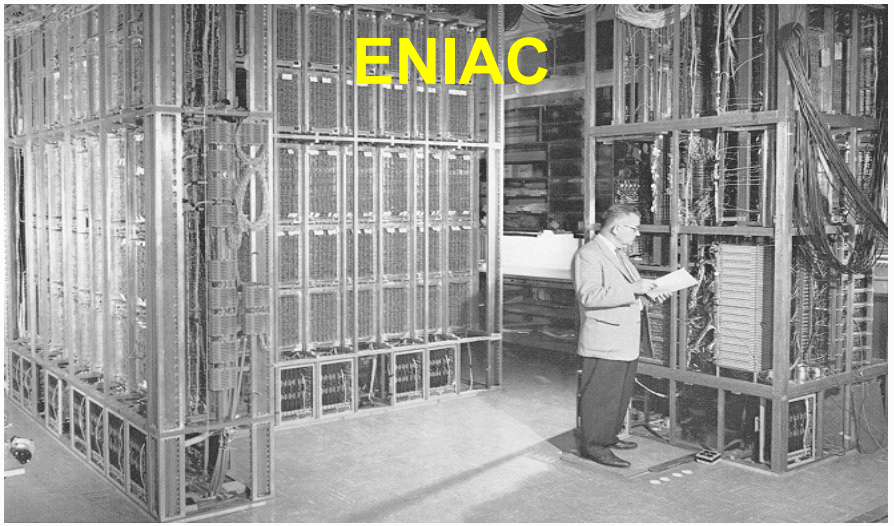
?



ENIAC: the "Electronic Numerical Integrator and Calculator"
[U.S. Army photo]

No I/O system support!

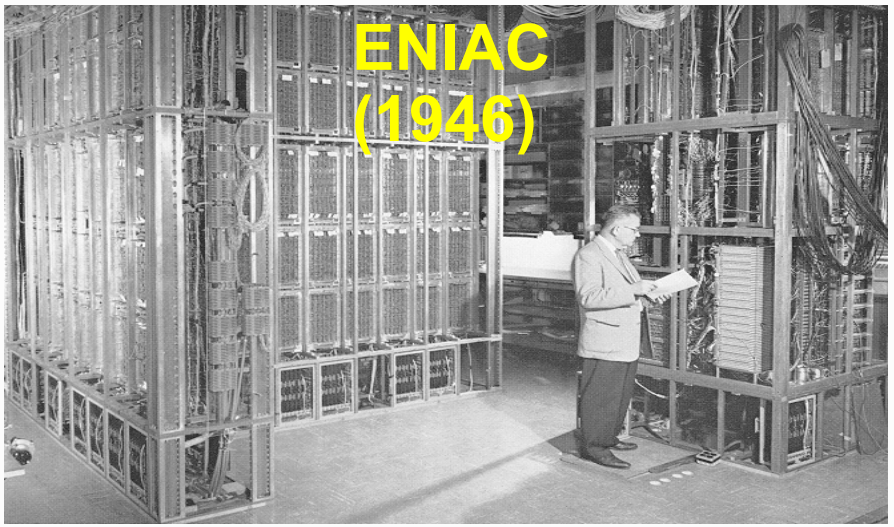




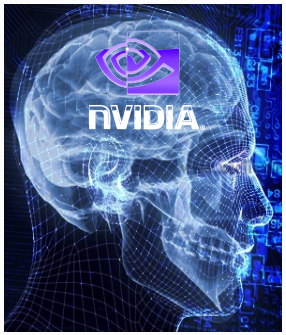
ENIAC: the "Electronic Numerical Integrator and Calculator"
[U.S. Army photo]

- **GPU programs cannot** initiate I/O
  - Cannot read/write files
  - Cannot communicate via network

I/O is the next step in the GPU evolution



ENIAC: the "Electronic Numerical Integrator and Calculator"
[U.S. Army photo]



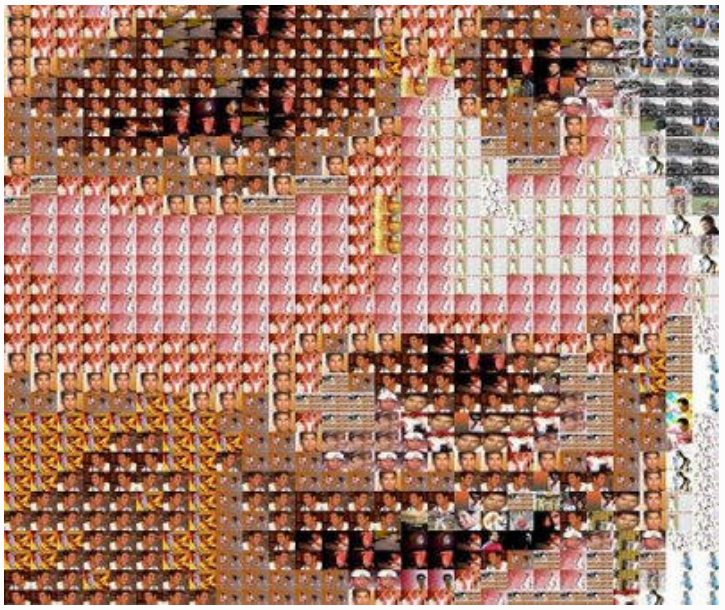
I/O is the next step in the GPU evolution

grator and Calculator"

Hardware is already here, we need software abstractions

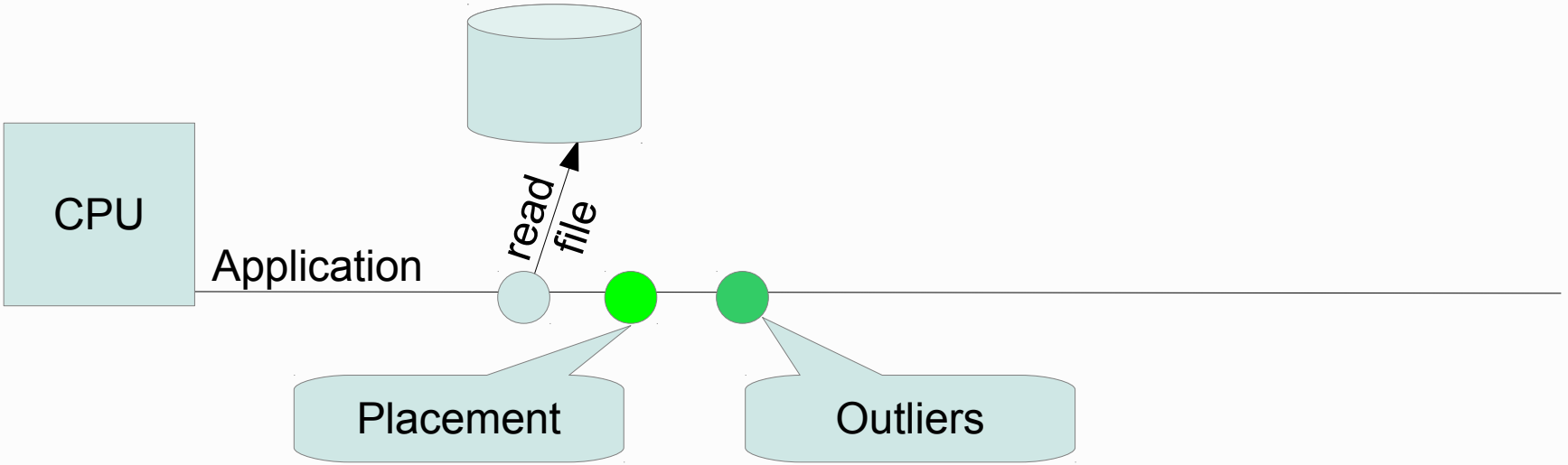
I/O intensive applications

- Collage from a set of images
- Data-dependent control flow

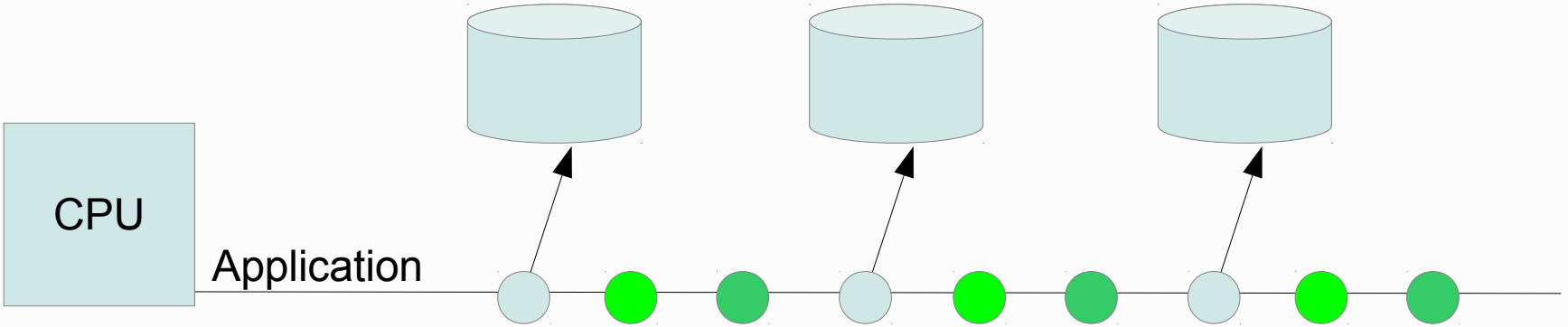


http://www.codeproject.com/Articles/36347/Face-Collage

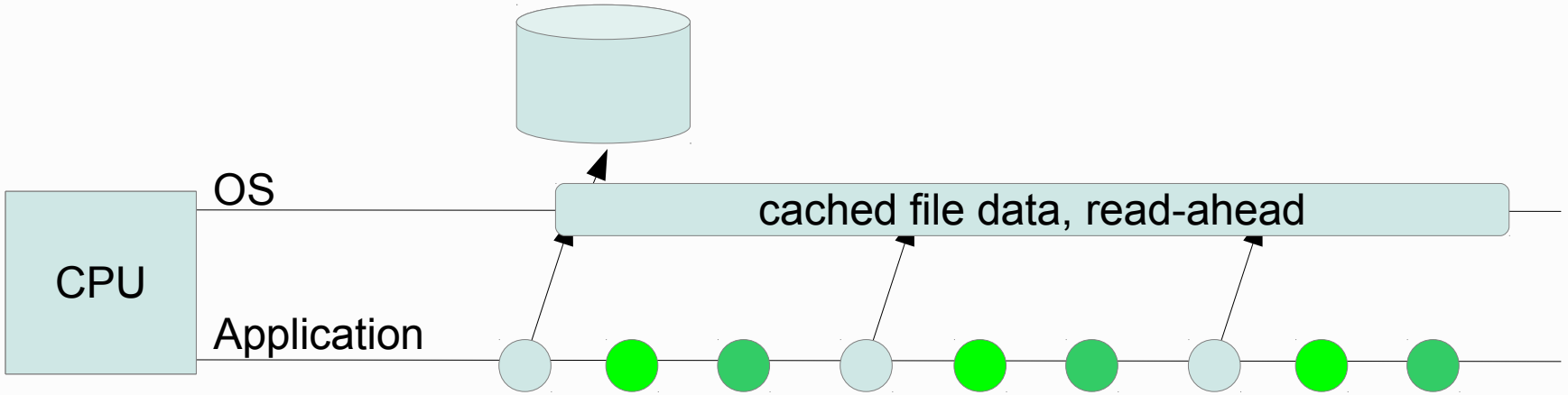
I/O intensive applications on CPU



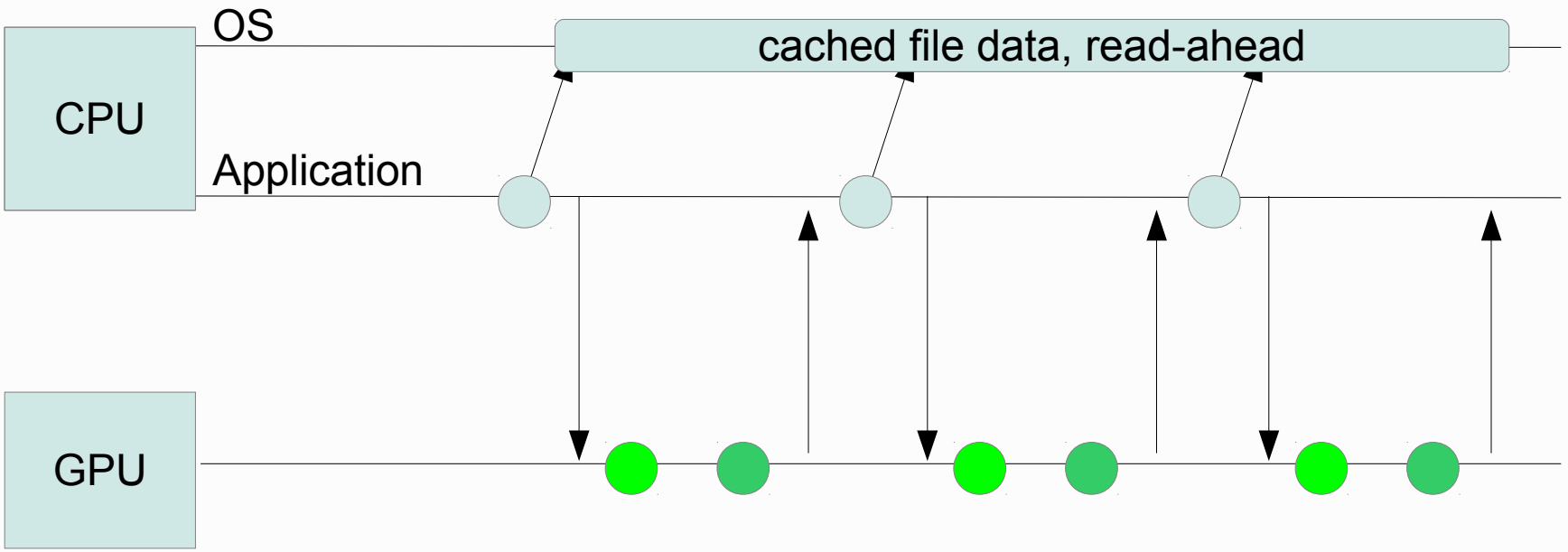
I/O intensive applications on CPU



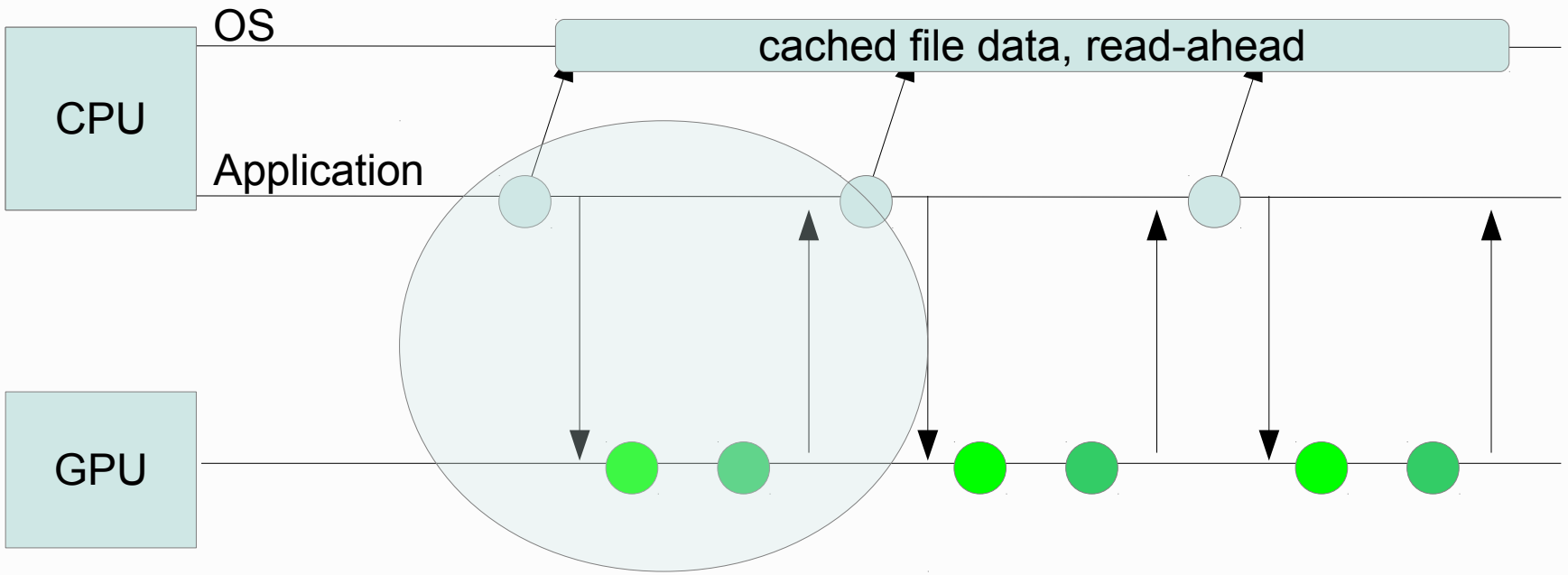
OS I/O subsystem helps



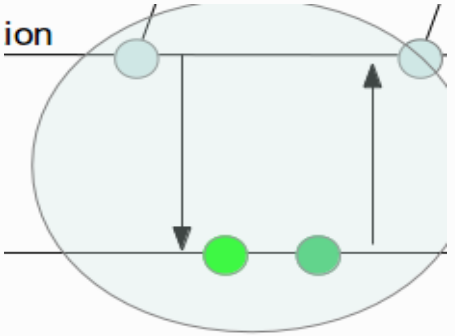
Offloading computations to GPU

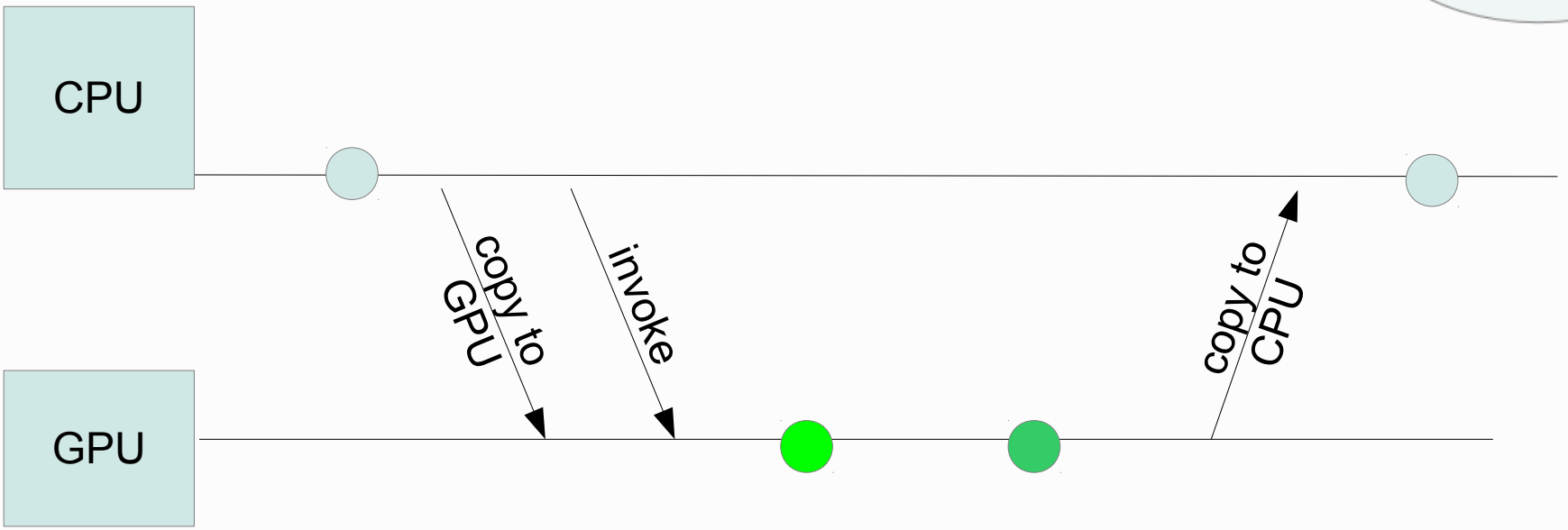


Offloading computations to GPU

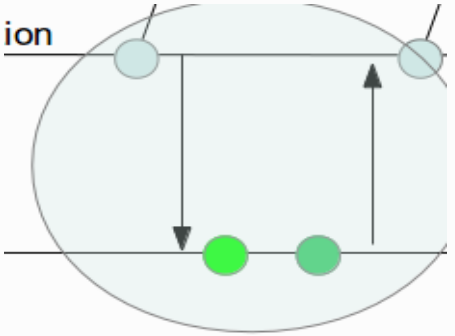


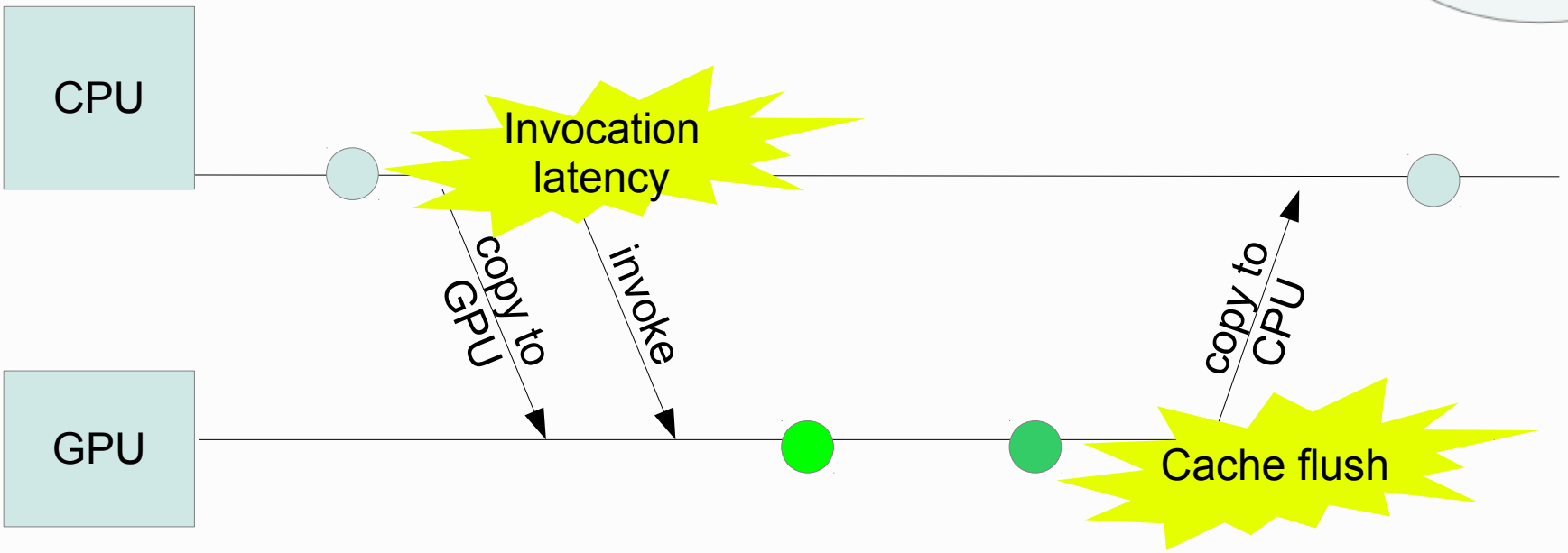
Offloading computations to GPU



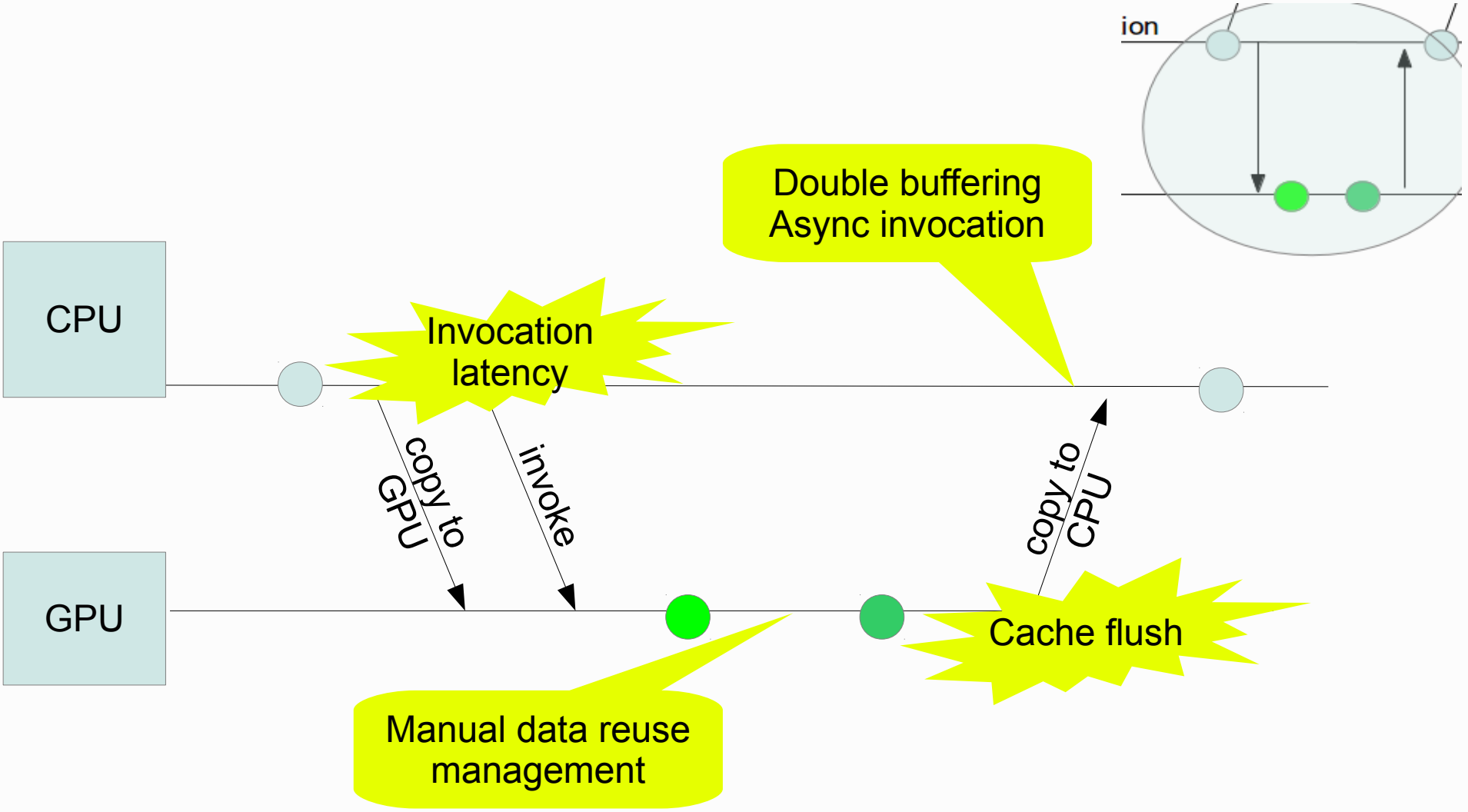


Kernel termination overheads





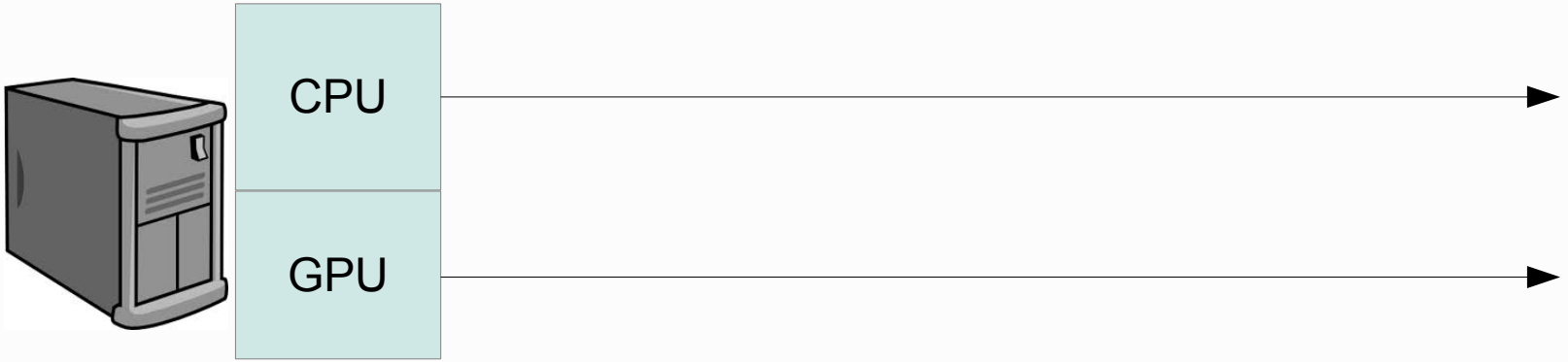
I/O implementation complexity



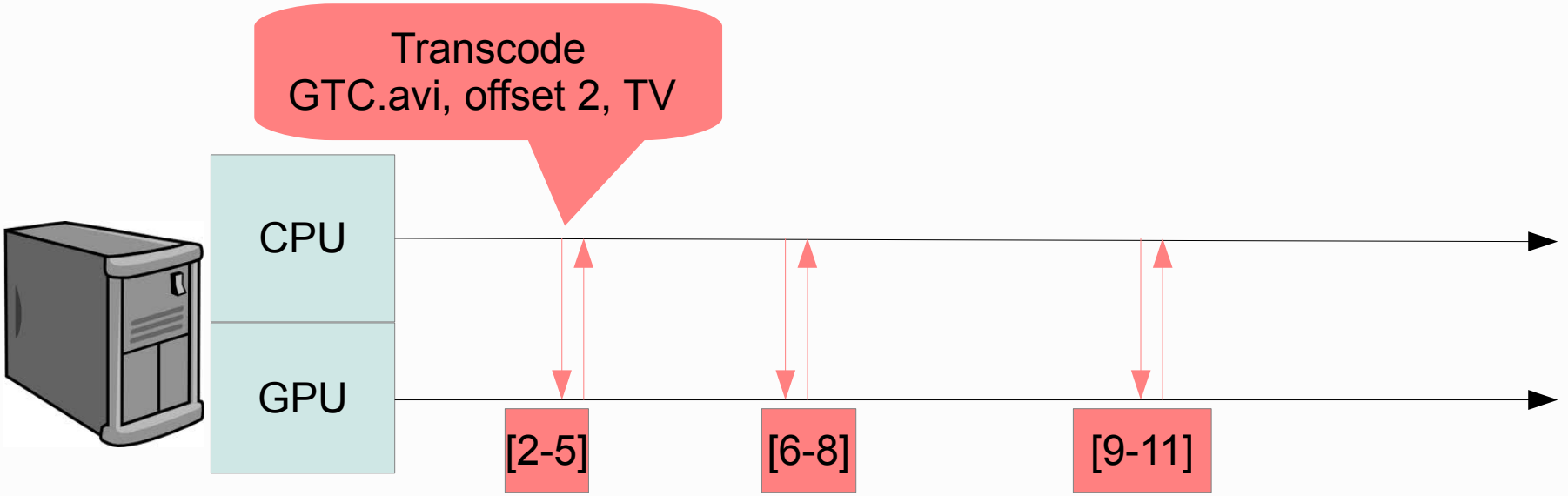
What's wrong with CPU-side I/O

- **P***erformance*
  - Typically requires GPU kernel termination
- **P***rogrammability*
  - CPU-GPU ping-pong complicates the code
- **P***ortability*
  - Ad hoc and incompatible with future/other GPUs

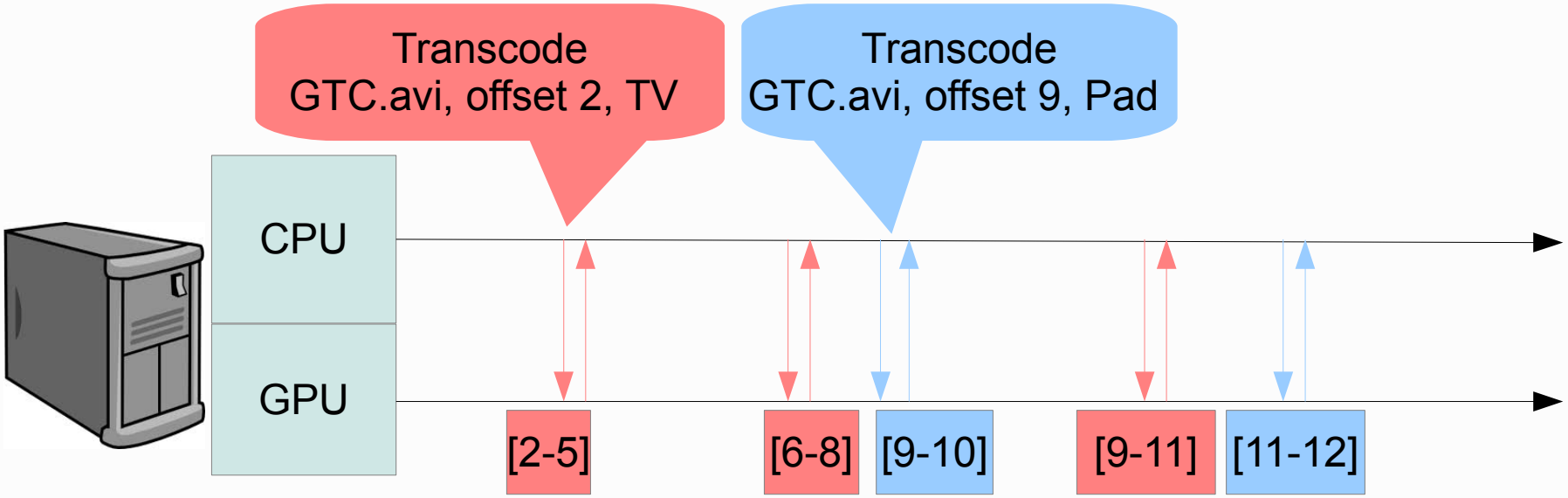
Streaming server



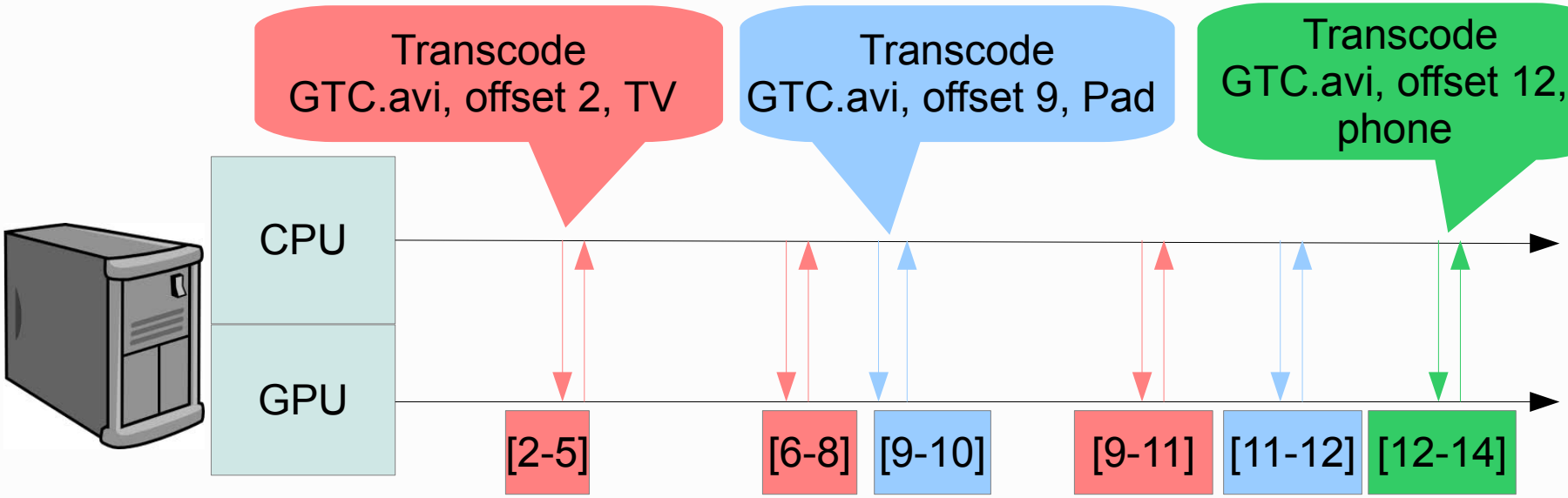
Streaming server



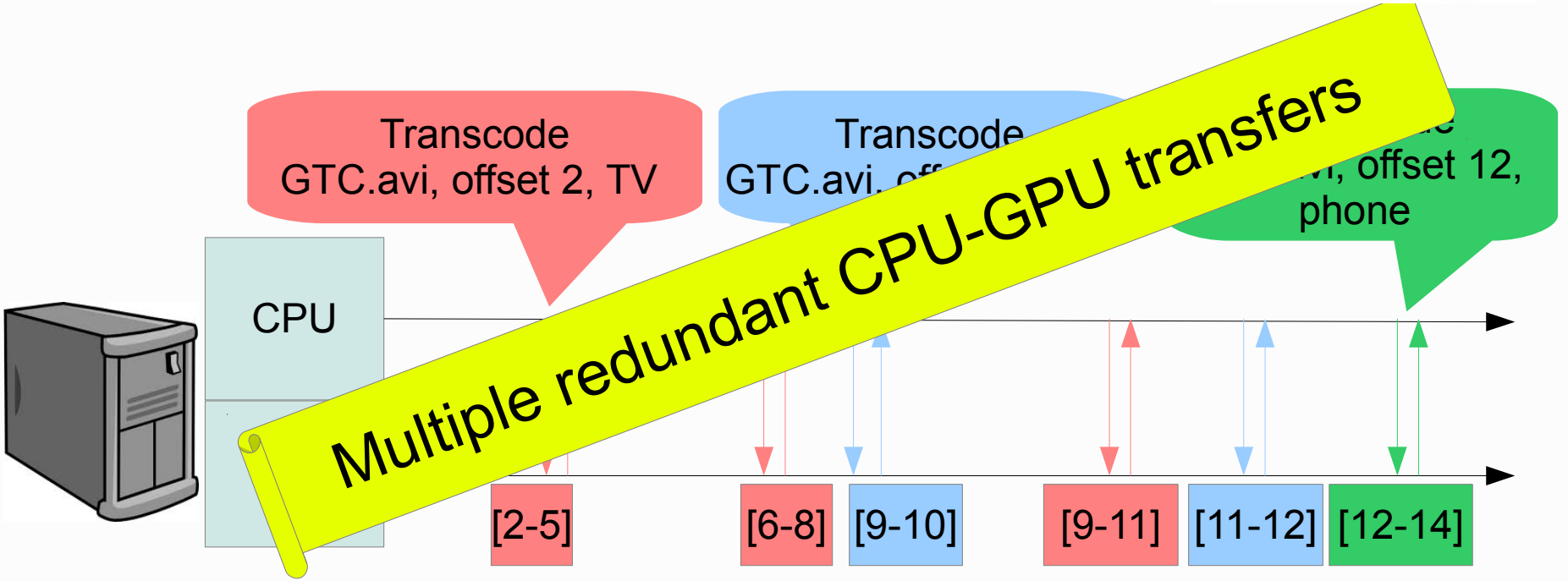
Streaming server



Streaming server



Streaming server



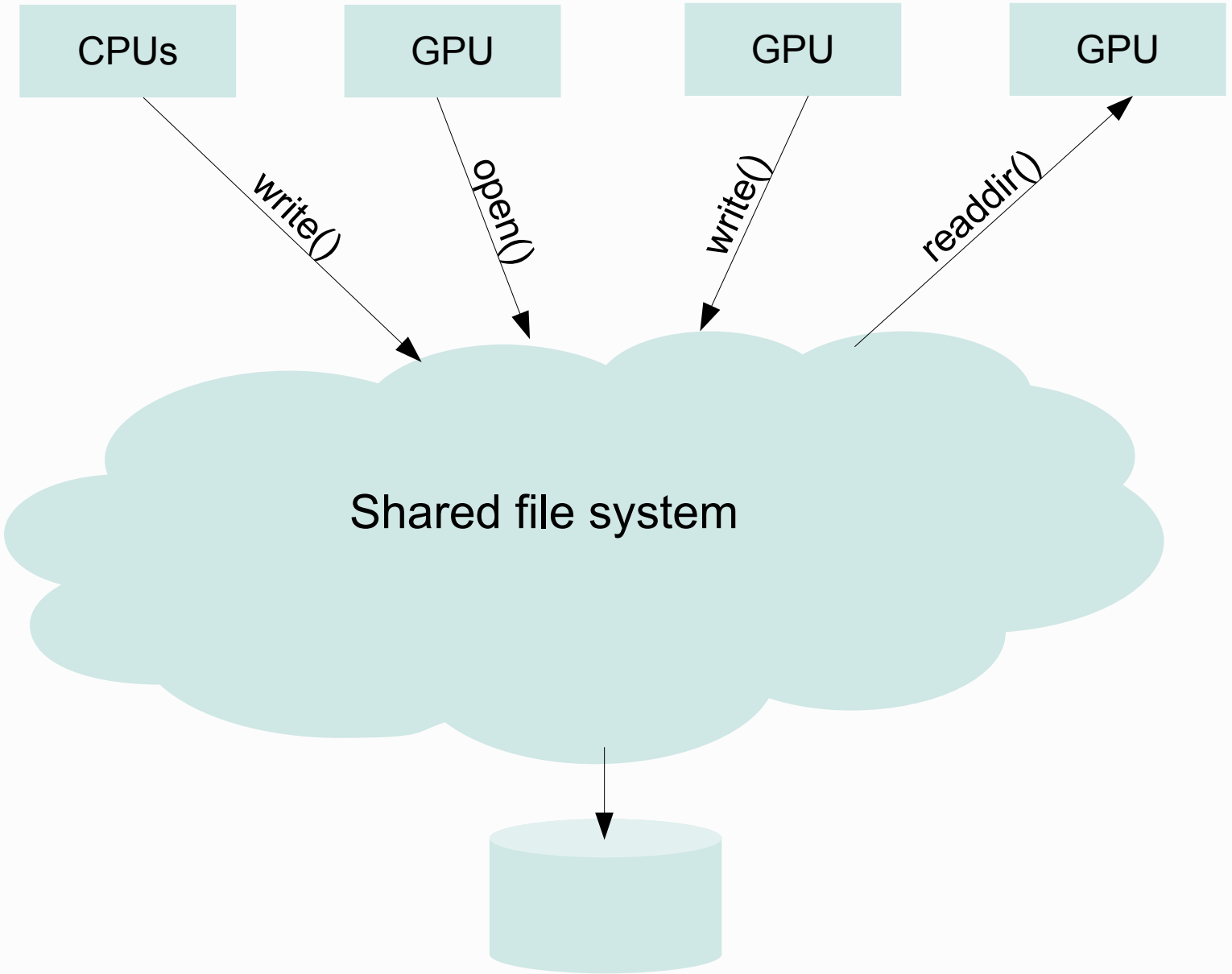
Wanted: I/O services **on GPUs**

- Familiar I/O primitives and semantics
  - Files (read/write/close/open), pipes
  - Sockets
- Direct access to I/O devices (without explicit CPU programming)
  - *Aka* generic GPUDirect
- Interprocess communication with peer CPUs/GPUs

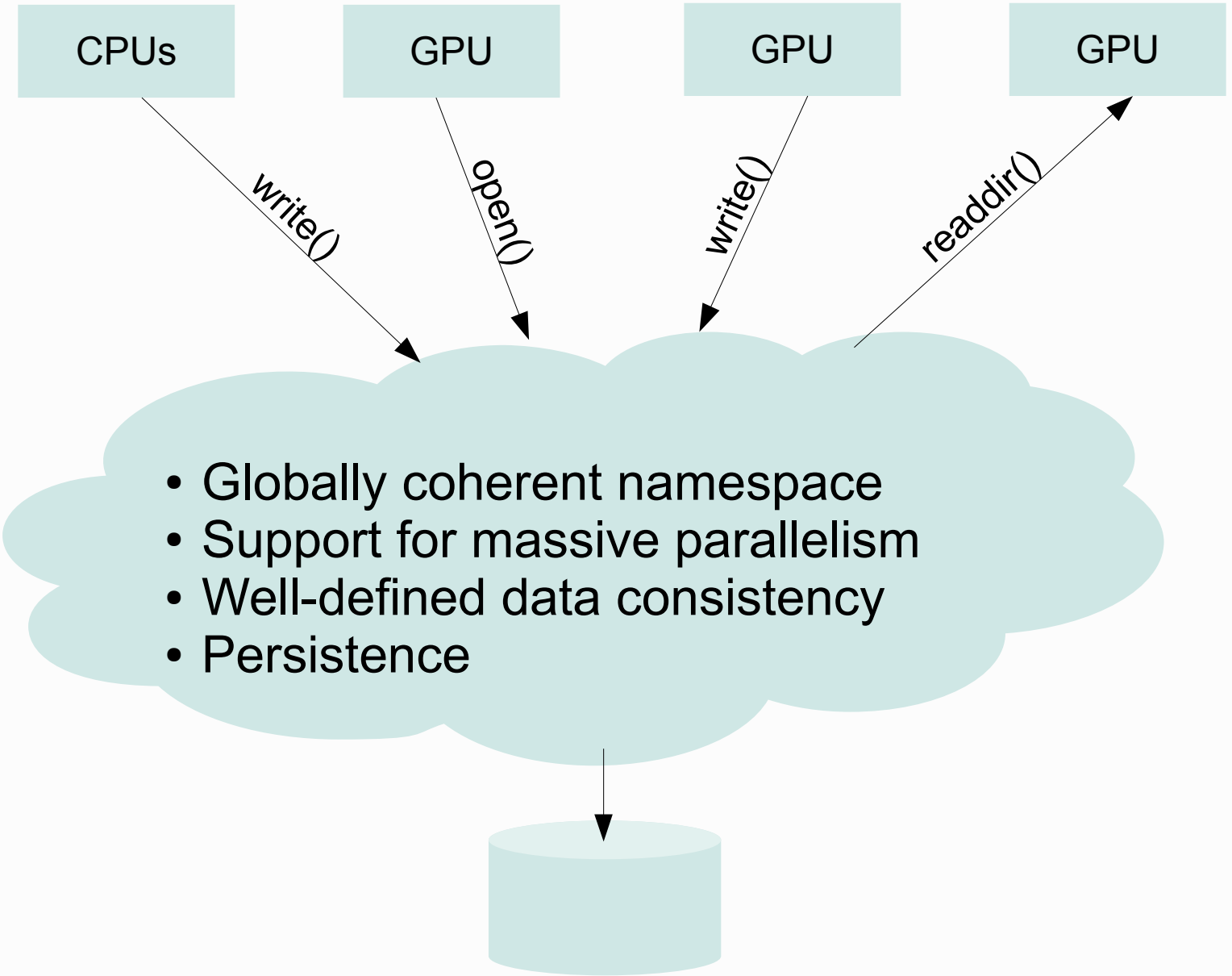
Wanted: I/O services **on GPUs**

- Familiar I/O primitives and semantics
  - Files (read/write/close/open) **This talk**
  - Sockets, pipes
- Direct access to I/O devices (without explicit CPU programming)
  - *Aka* generic GPUDirect
- Interprocess communication with peer CPUs/GPUs

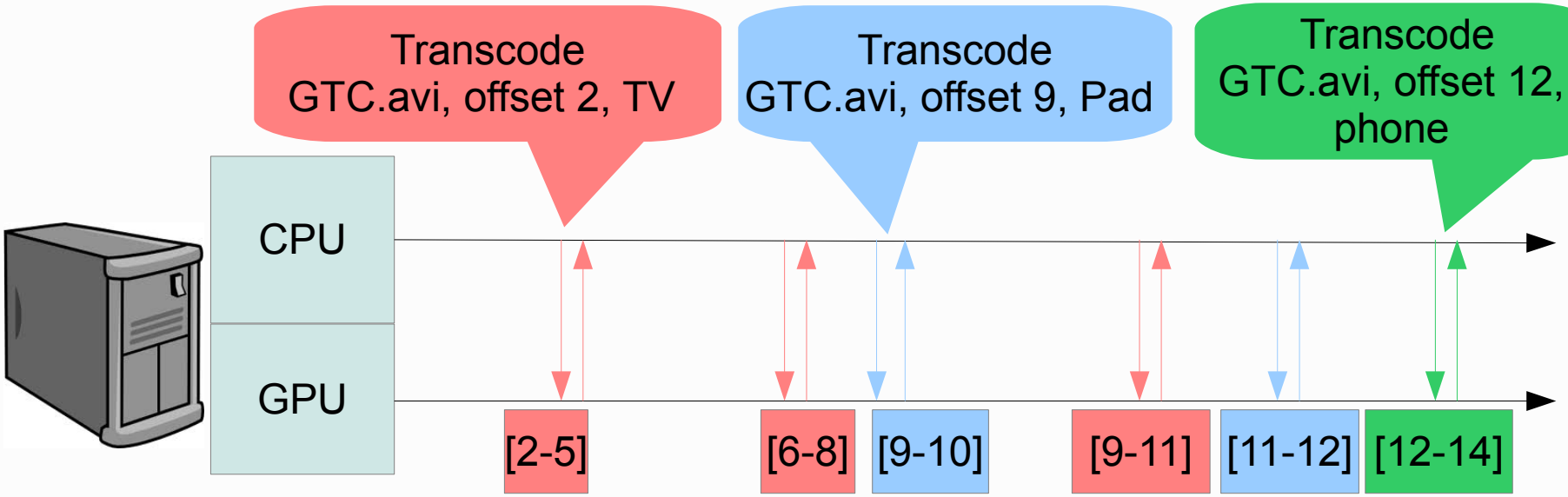
Inter-**processor** shared file system (GPUFS)



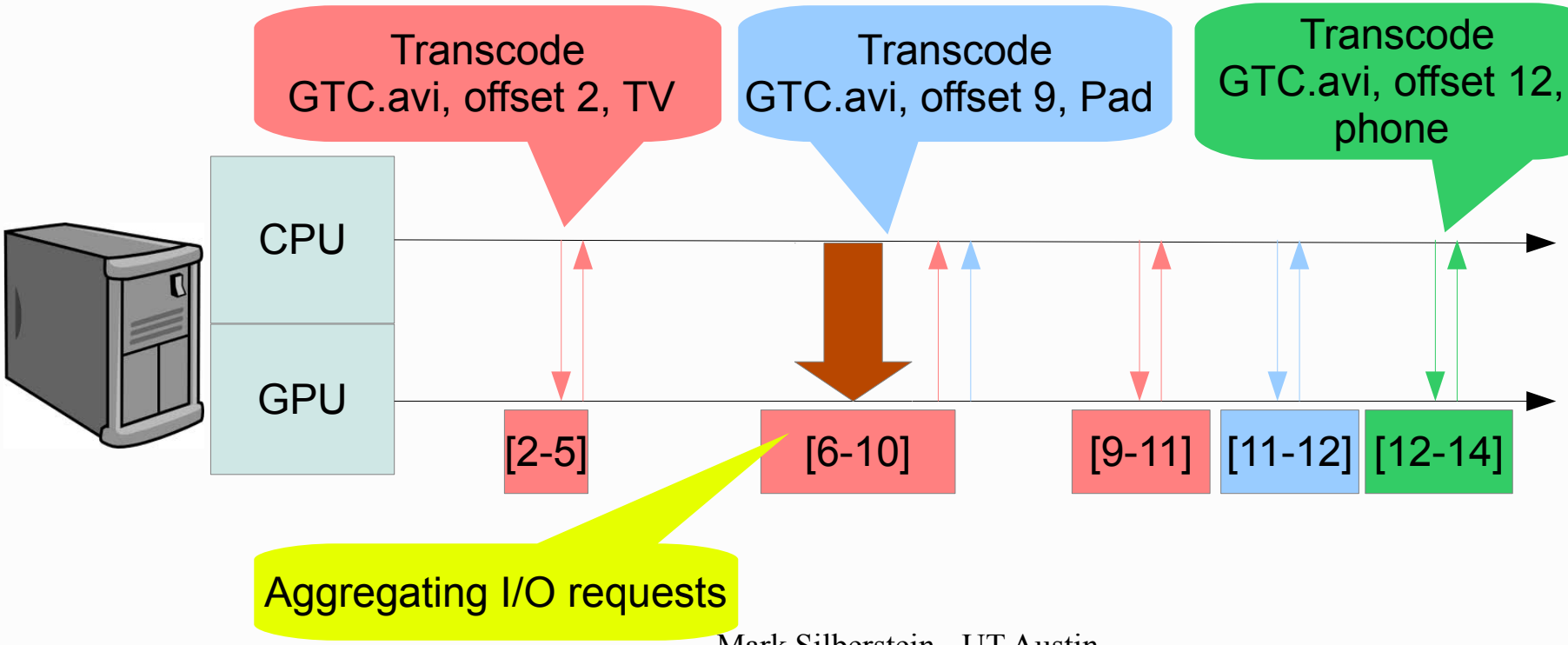
Inter-**processor** shared file system (GPUFS)



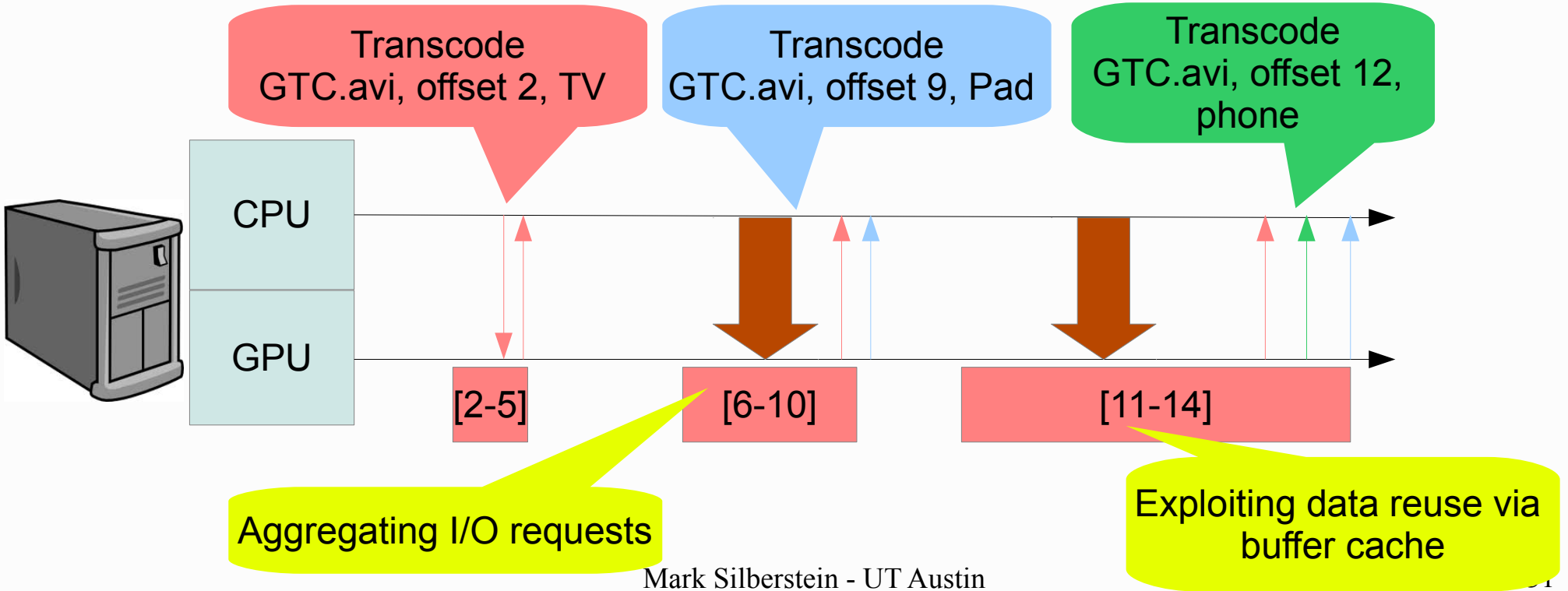
Streaming server – without GPUFS



Streaming server – with GPUFS



Streaming server – with GPUFS

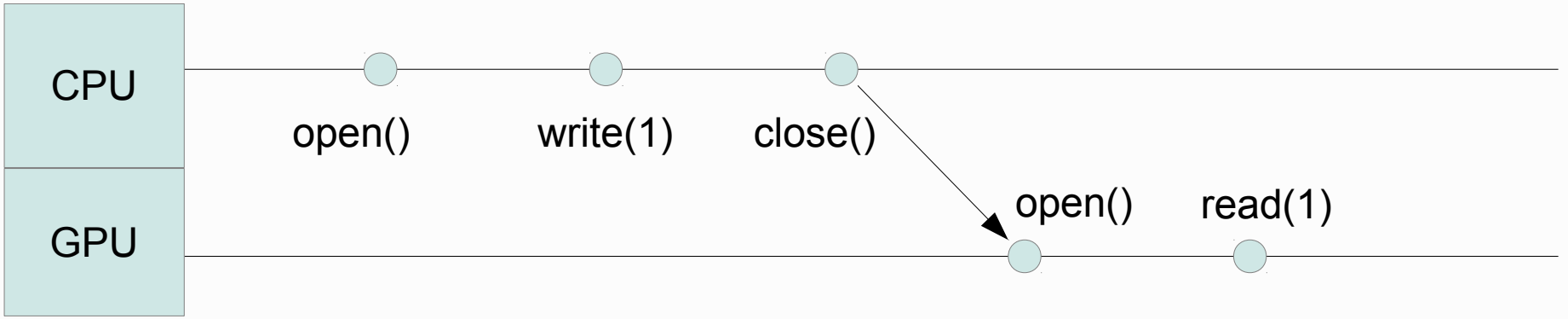


GPUFS features

- Fully functional file API in a **running** GPU kernel
- File descriptors stay open across multiple kernels of the same CPU process
- File system hints for performance
  - e.g. “streamable” - effectively a named pipe

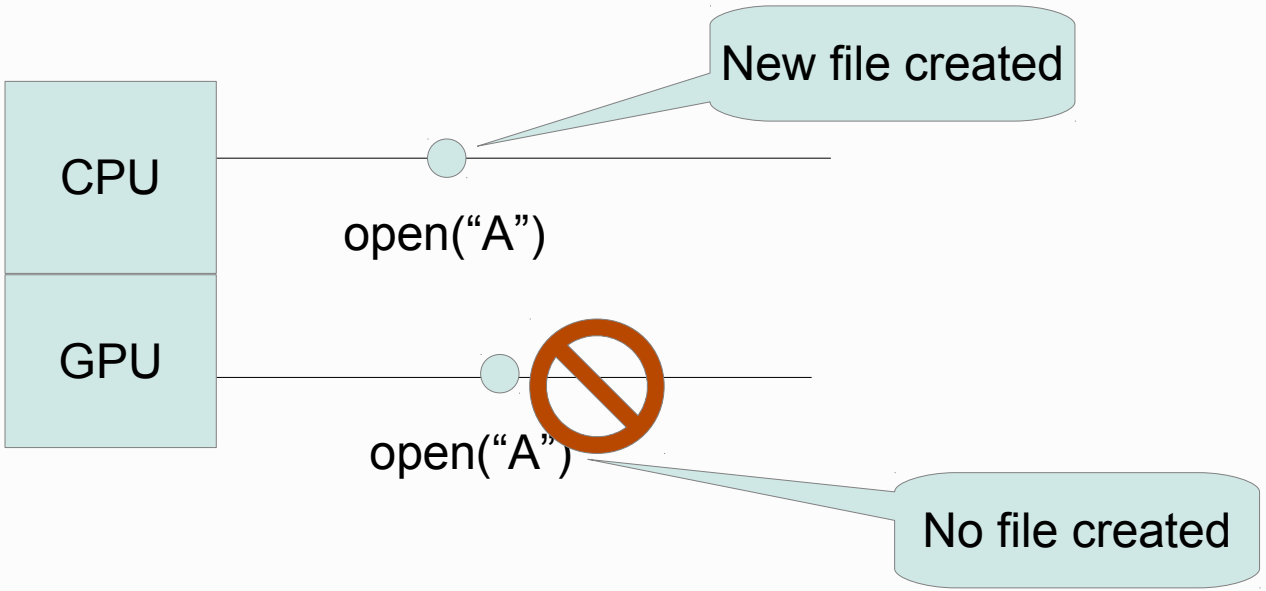
GPUFS semantics - Data

- Data consistency: as in distributed FS
  - No guarantees for concurrent writes to the same file

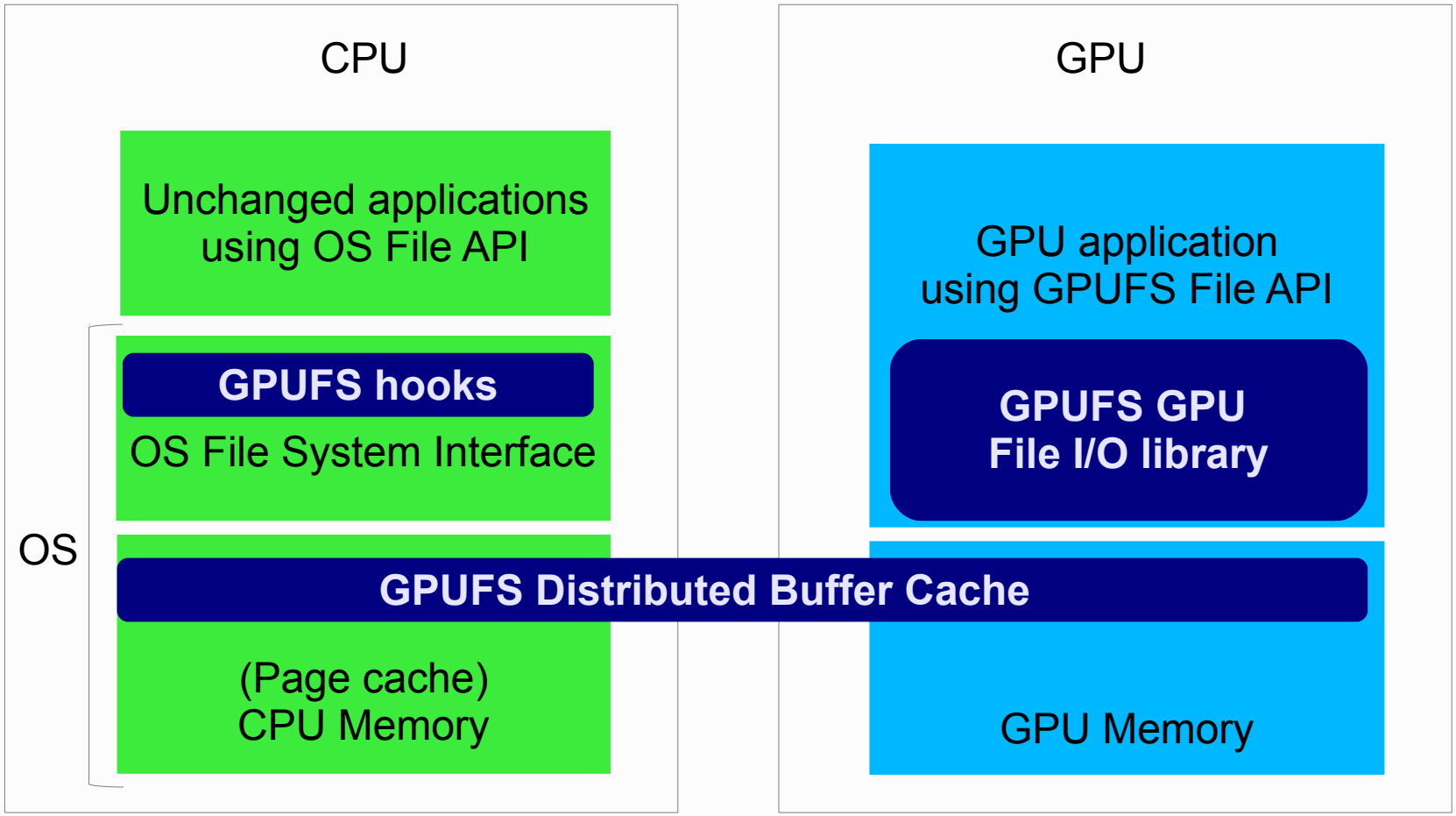


GPUFS semantics – File Names

- Name space consistency: as in local FS
  - no two files can be created with the same name
  - used for lock files



GPUFS design sketch



GPUFS CUDA API

- open – must be called for each thread block
- close – must match every open
- read/write – supported but discouraged
- pread/pwrite – fully parallel warp/block bulk-synchronous
- open(O_CREAT|O_EXCL) for lock files
- readdir/stat

Code sample – bulk-synchronous

```
__shared__ int fd;
// invoke in one thread
one_thread(0,fd=open(filename,O_RDONLY));
if (fd<0) __bailout();
// bulk-synchronous read by all threads
gpu_pread(fd,offset,size,buffer);
// invoke in one thread
one_thread(0,close(fd));
```

Same parameters for all threads in TB

Preliminary results
Measuring overheads

- 256MB file, 1MB GPU pages
- 1 page per thread block
- TESLA C2070

	CPU page cache (PCIe)	GPU page cache (Global memory)
Maximum bandwidth	5.4 GB/s	103 GB/s
`gpu_pread` `gpu_pwrite`	2.7 GB/s	42 GB/s

Preliminary results
Measuring overheads

- 256MB file, 1MB GPU pages
- 1 page per thread block
- TESLA C2070

	CPU page cache (PCIe)	GPU page cache (Global memory)
Maximum bandwidth	5.4 GB/s	103 GB/s
`gpu_pread` `gpu_pwrite`	2.7 GB/s	42 GB/s

`open, read/write,close:`
sync with CPU, file table update,
memory allocation,transfers

Preliminary results
Measuring overheads

- 256MB file, 1MB GPU pages
- 1 page per thread block
- TESLA C2070

	CPU page cache (PCIe)	GPU page cache (Global memory)
Maximum bandwidth	5.4 GB/s	103 GB/s
`gpu_pread` `gpu_pwrite`	2.7 GB/s	42 GB/s

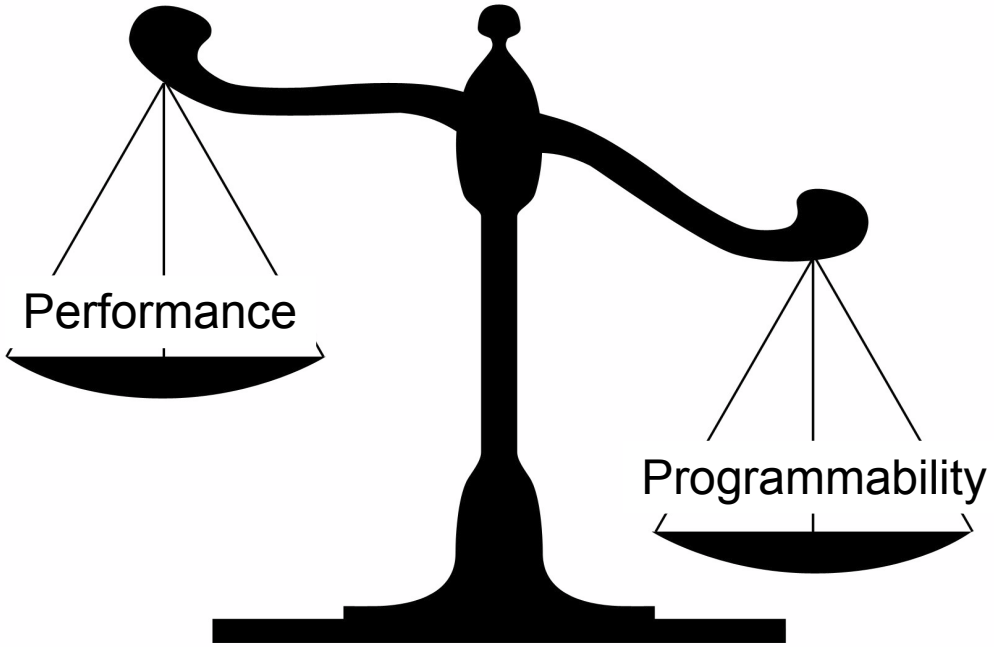
Manageable overheads for increased programmability

Summary
I/O services on GPUs

- Dynamic working set
- Simpler CPU-like development
- Portability
- Forward compatibility
- Interoperability with legacy programs
- Coordination with peer GPUs and CPUs

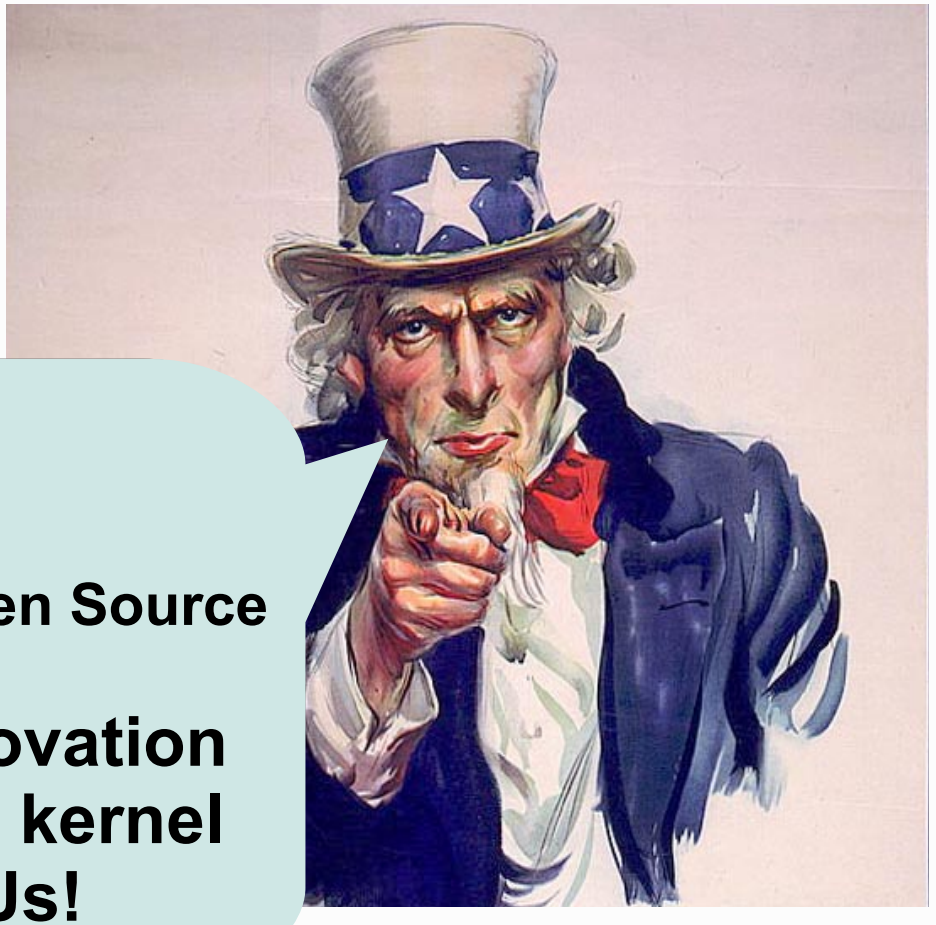
Summary
I/O services on GPUs

- Dy
- Sin
- Po
- Fo
- Int
- Co



Allow broader adoption outside the HPC community

Before I leave...

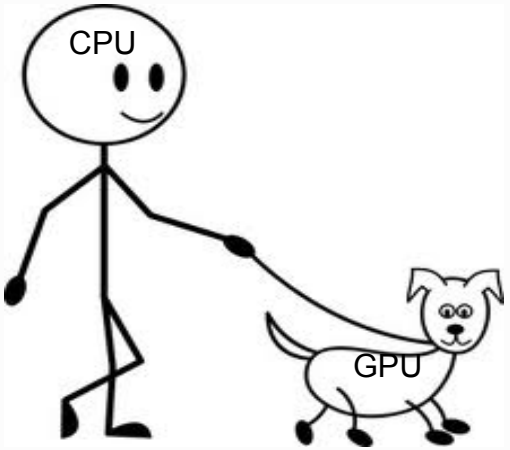


NVIDIA!

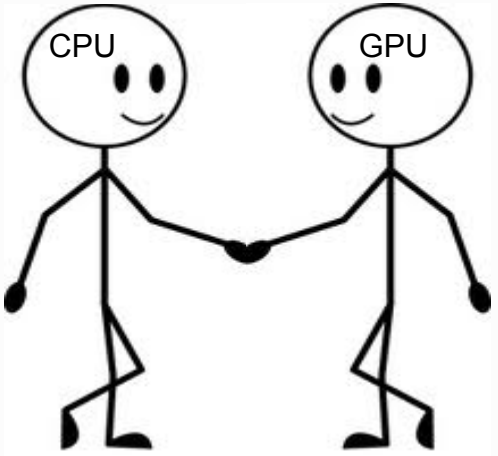
Open API is NOT Open Source

Enable GPU innovation by providing OS kernel APIs for GPUs!

Set GPUs free!



Without I/O support



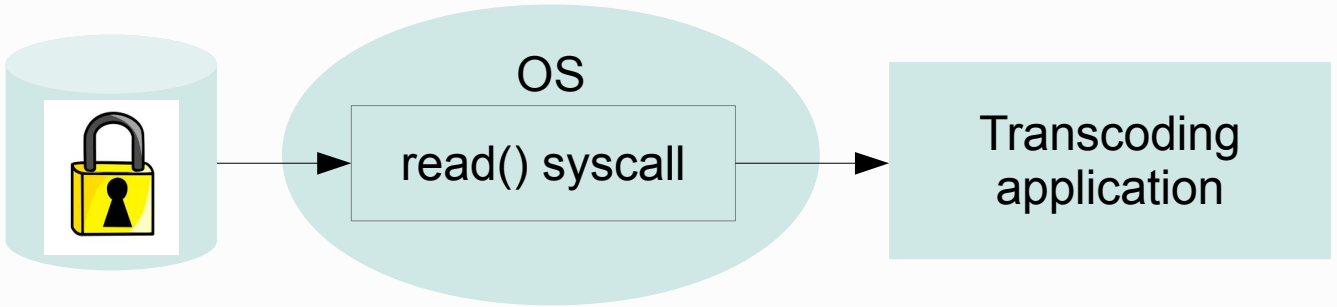
With I/O support

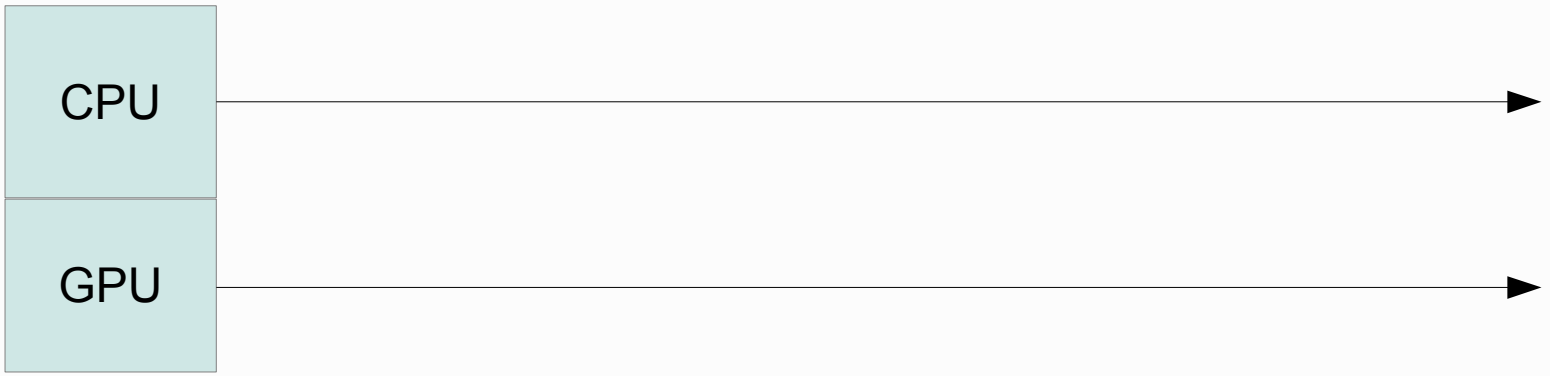
Thank you

Backup

Adding Encryption Support

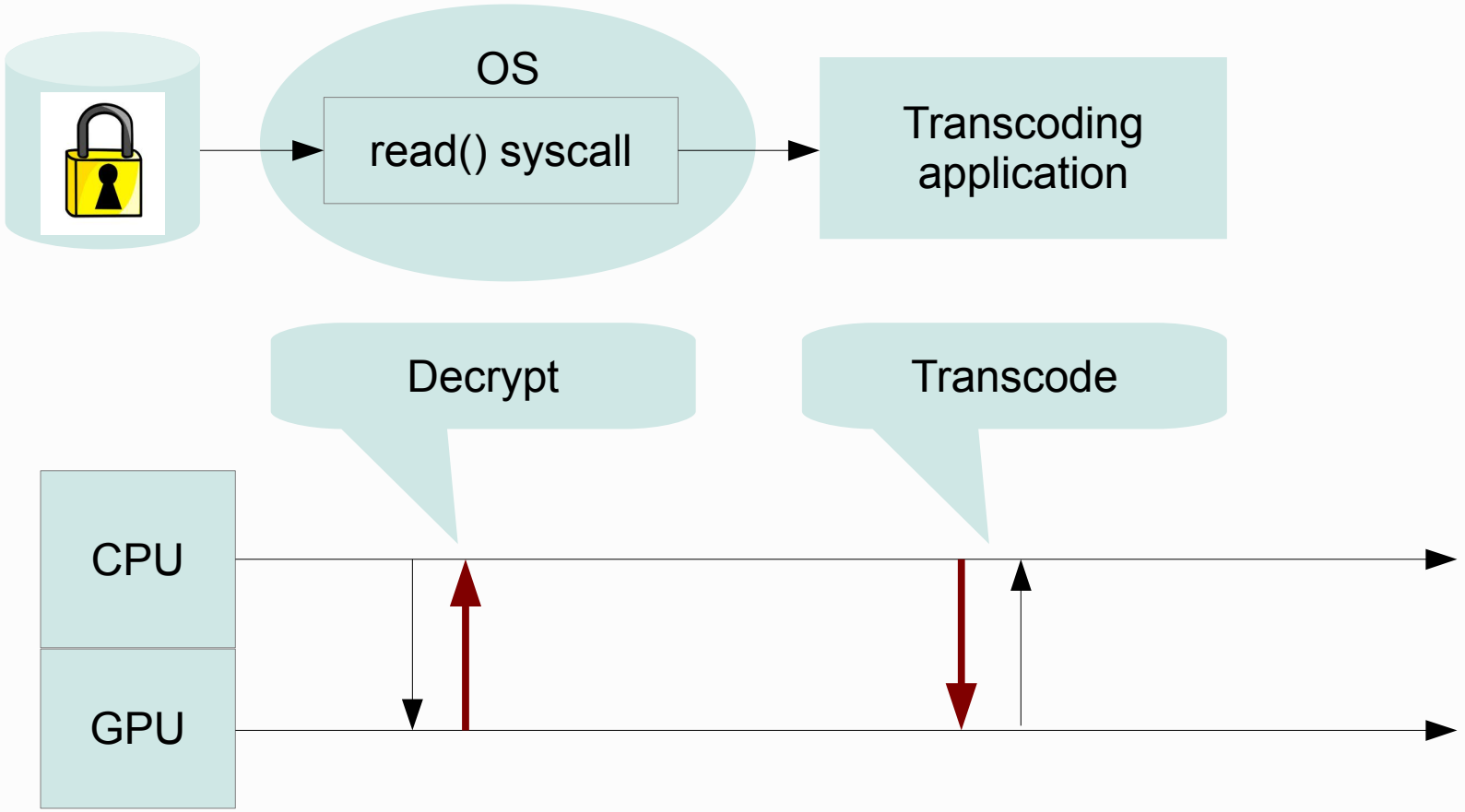
Streaming server with GPU-accelerated encrypted storage





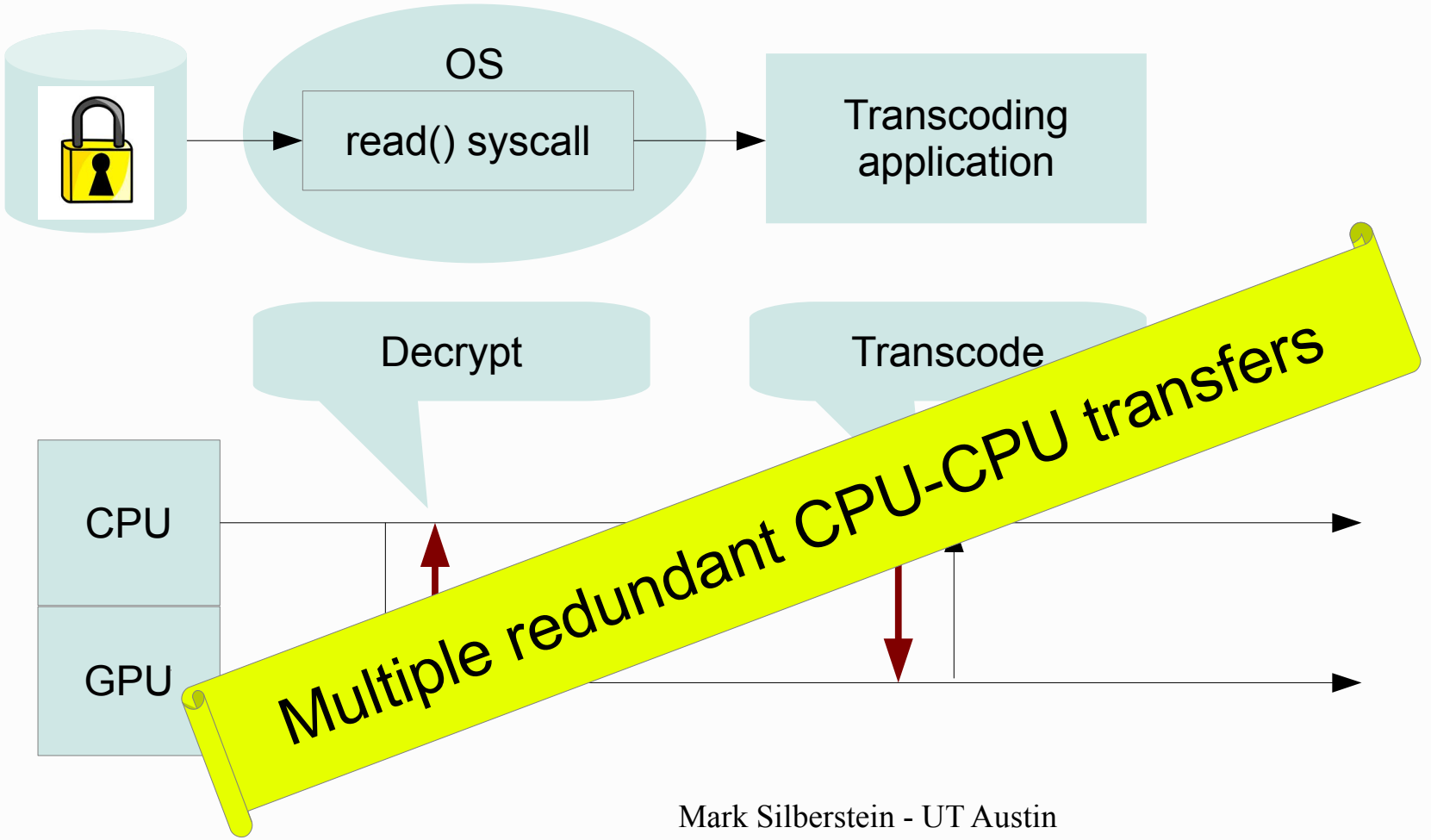
Adding Encryption Support

Streaming server with GPU-accelerated encrypted storage



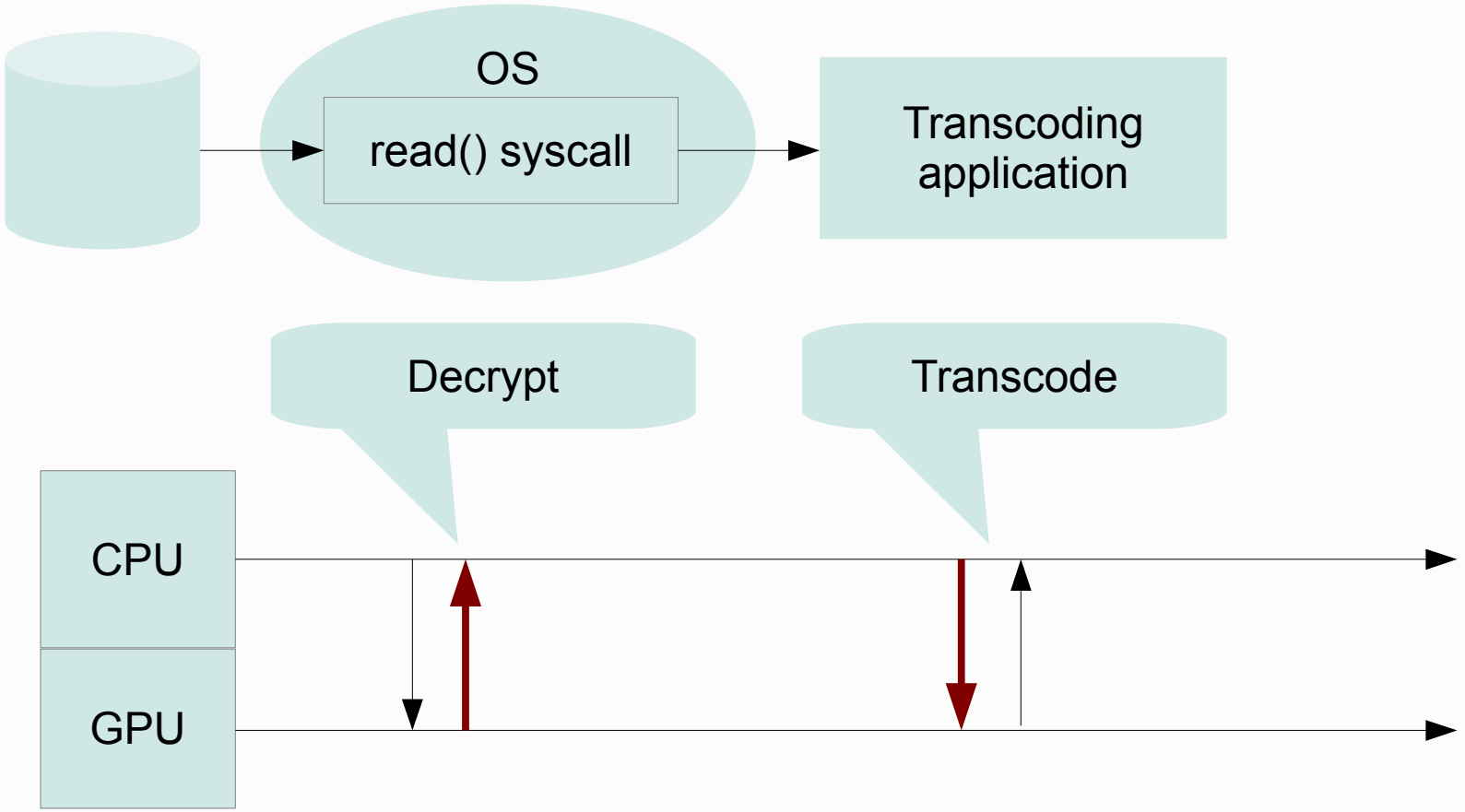
Adding Encryption Support

Streaming server with GPU-accelerated encrypted storage



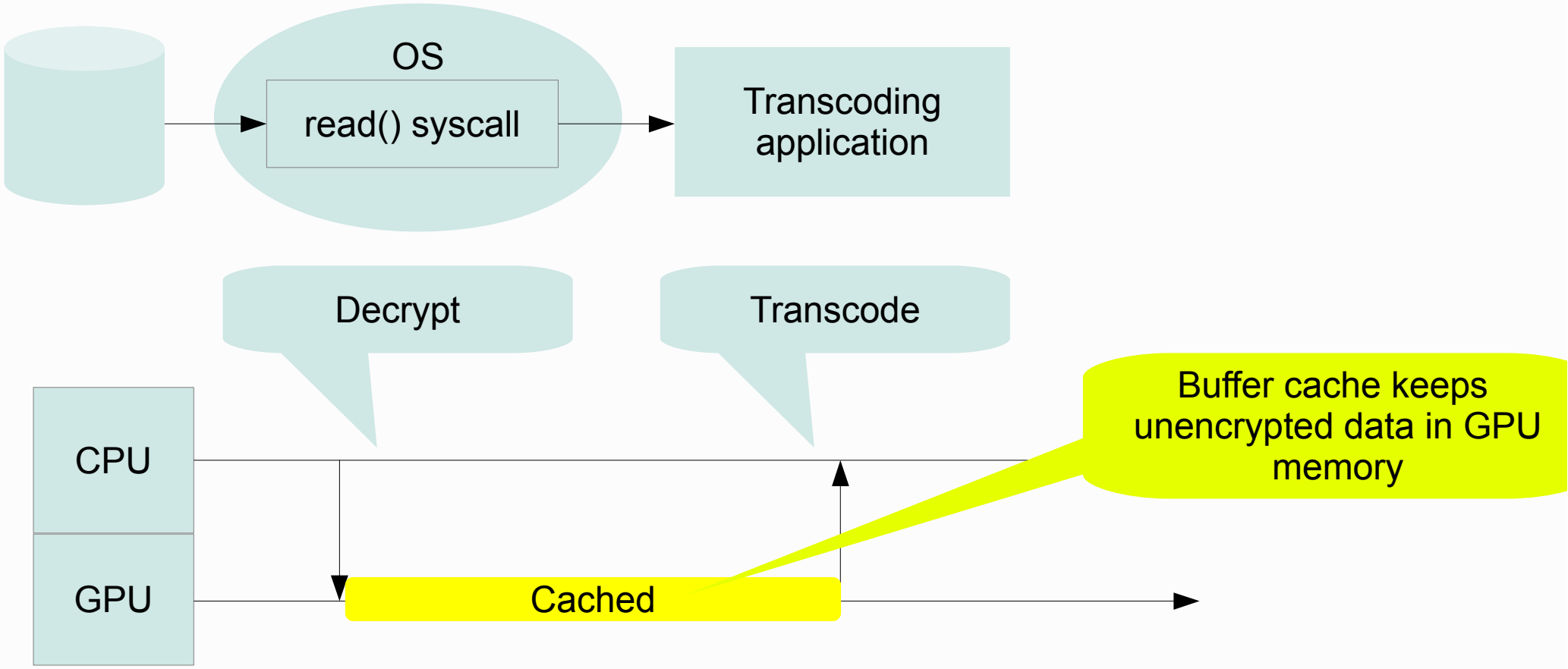
Without IDFS

Streaming server with GPU-accelerated encrypted storage



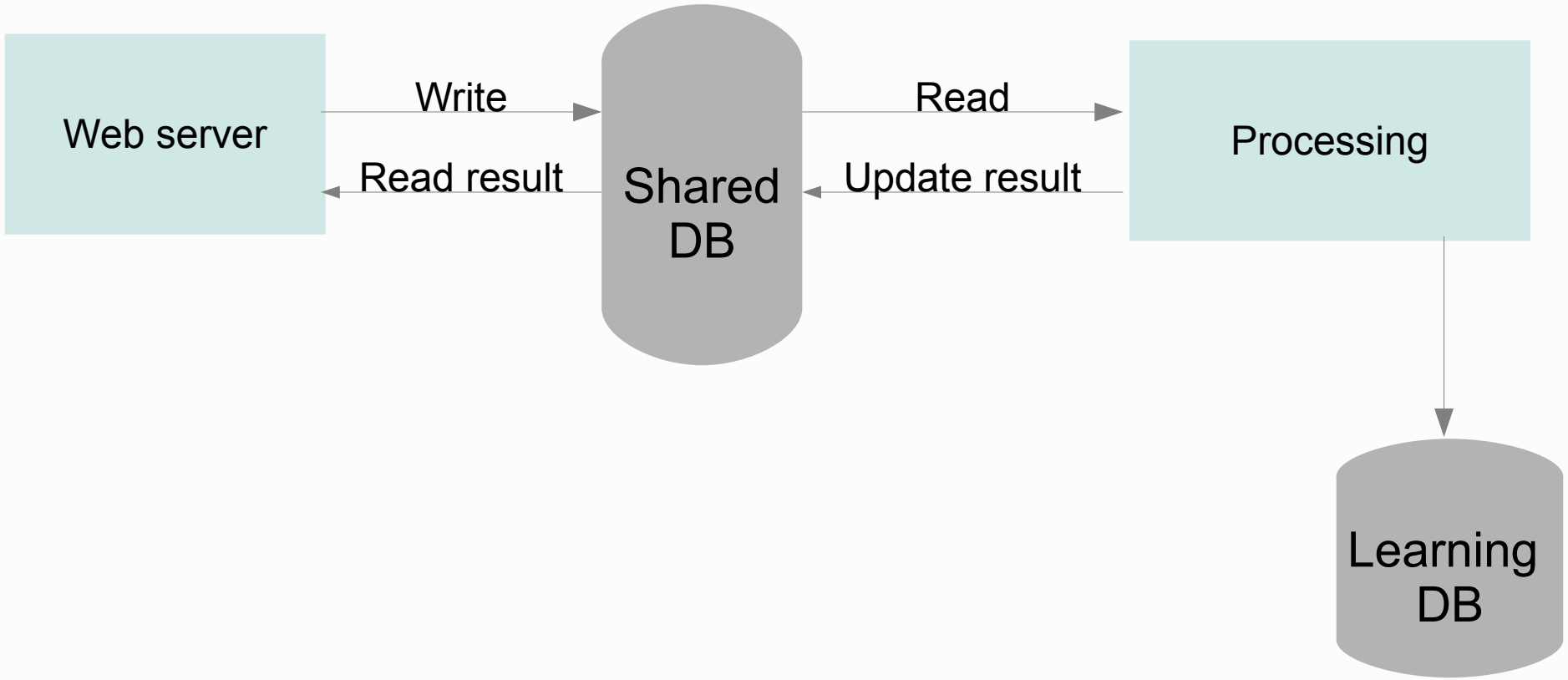
With IDFS

Streaming server with GPU-accelerated encrypted storage

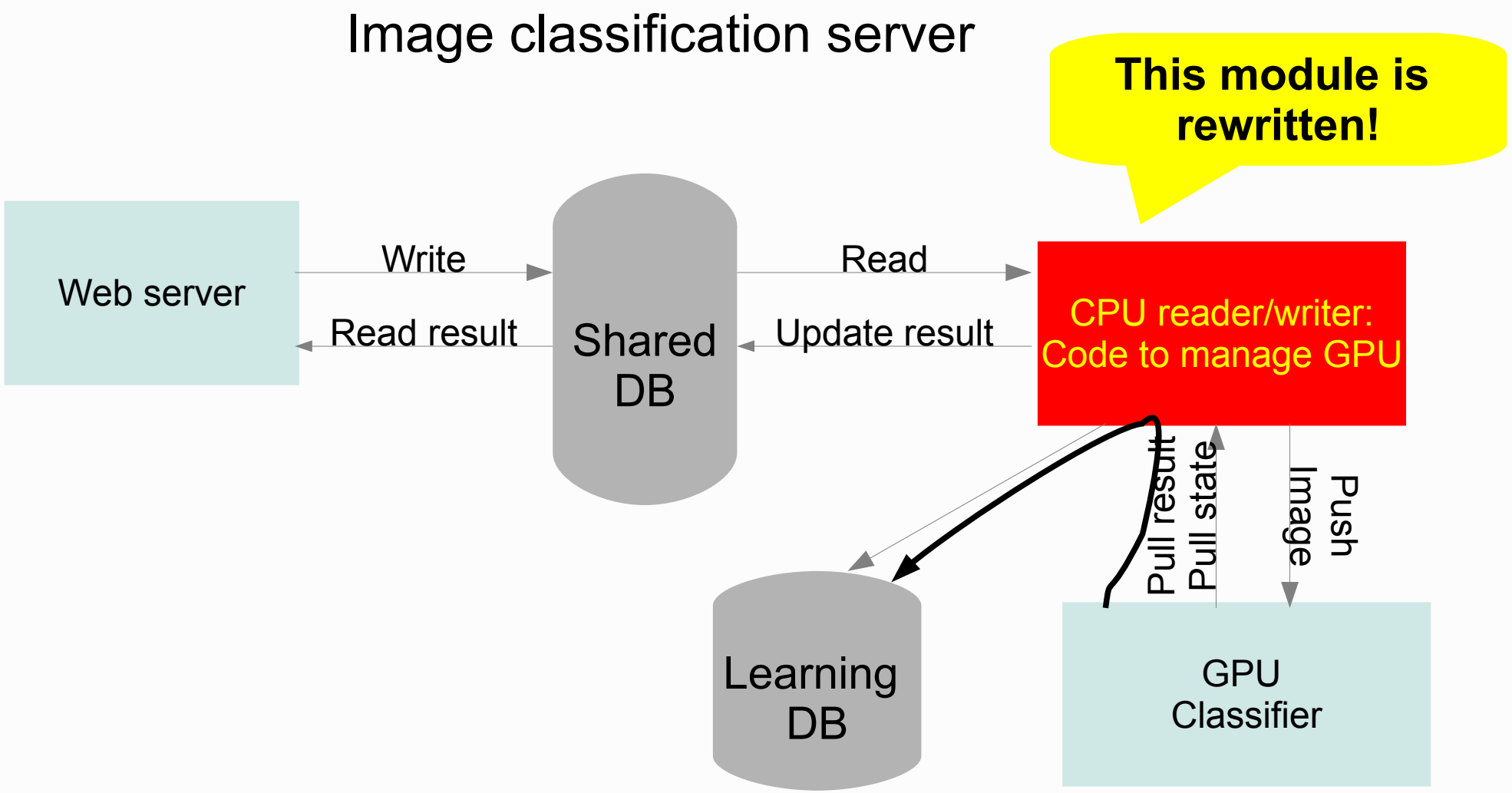


No OS I/O support (3)
Hard to evolve existing systems

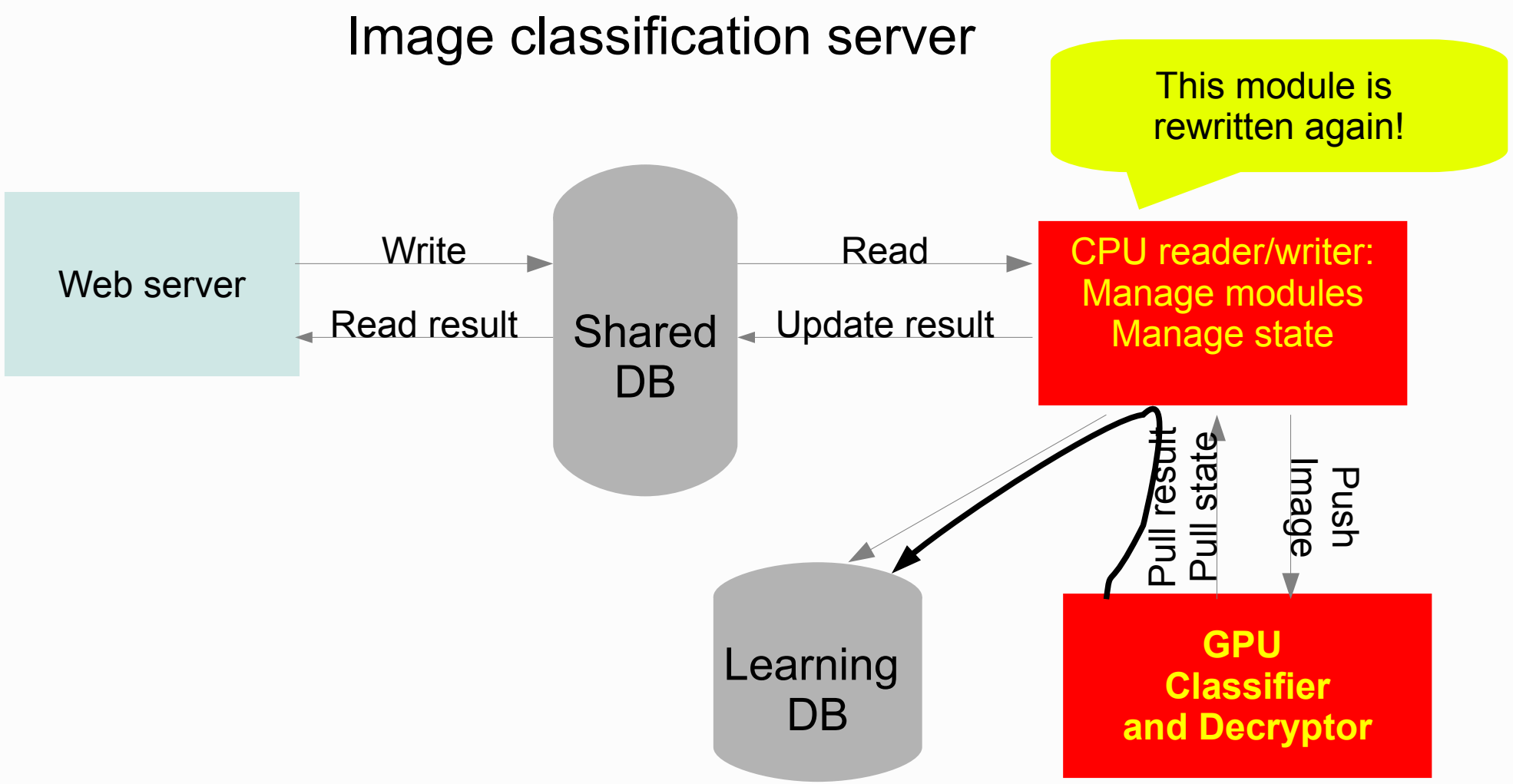
Image classification server



Adding a GPU requires rewrite

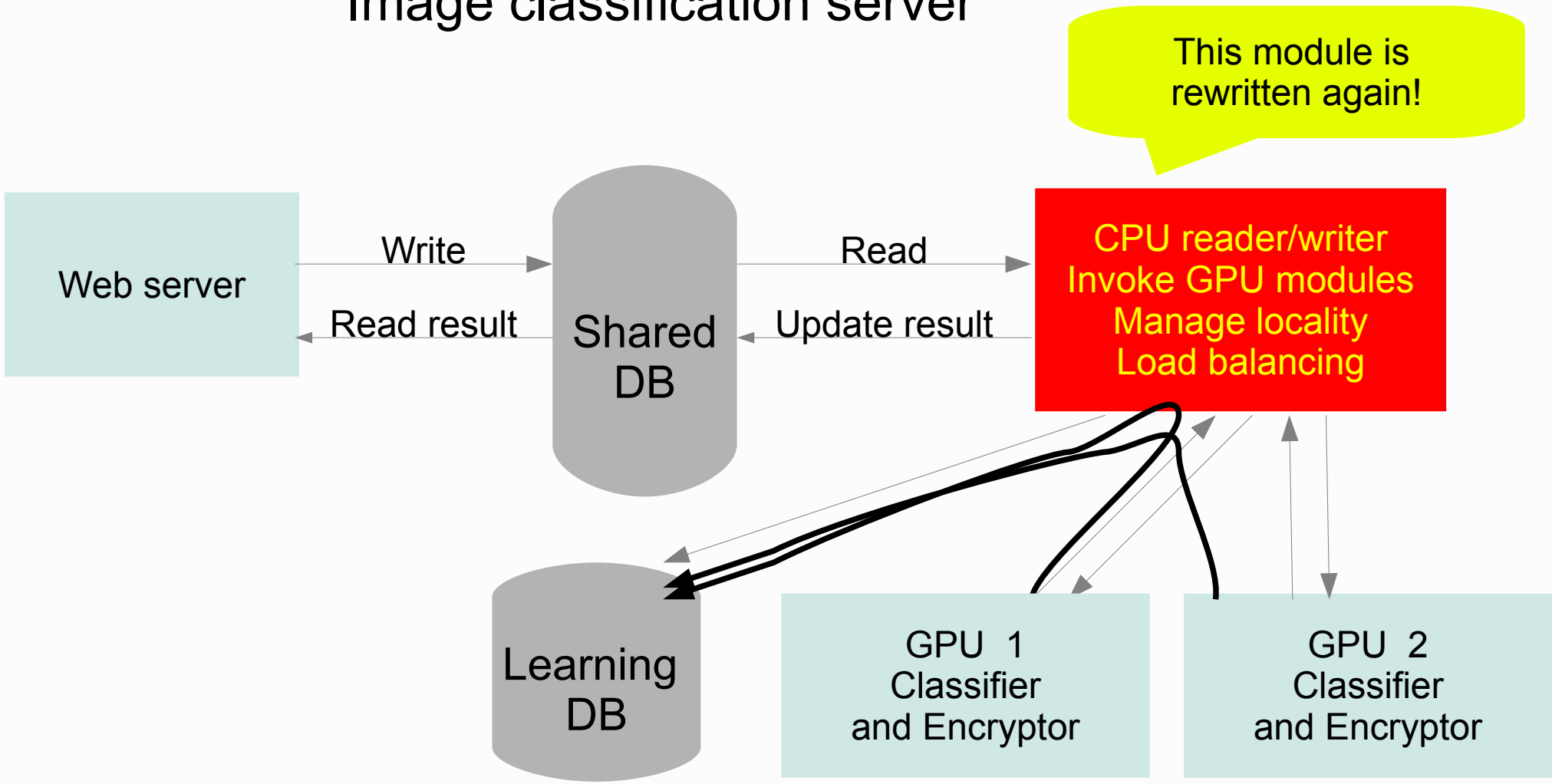


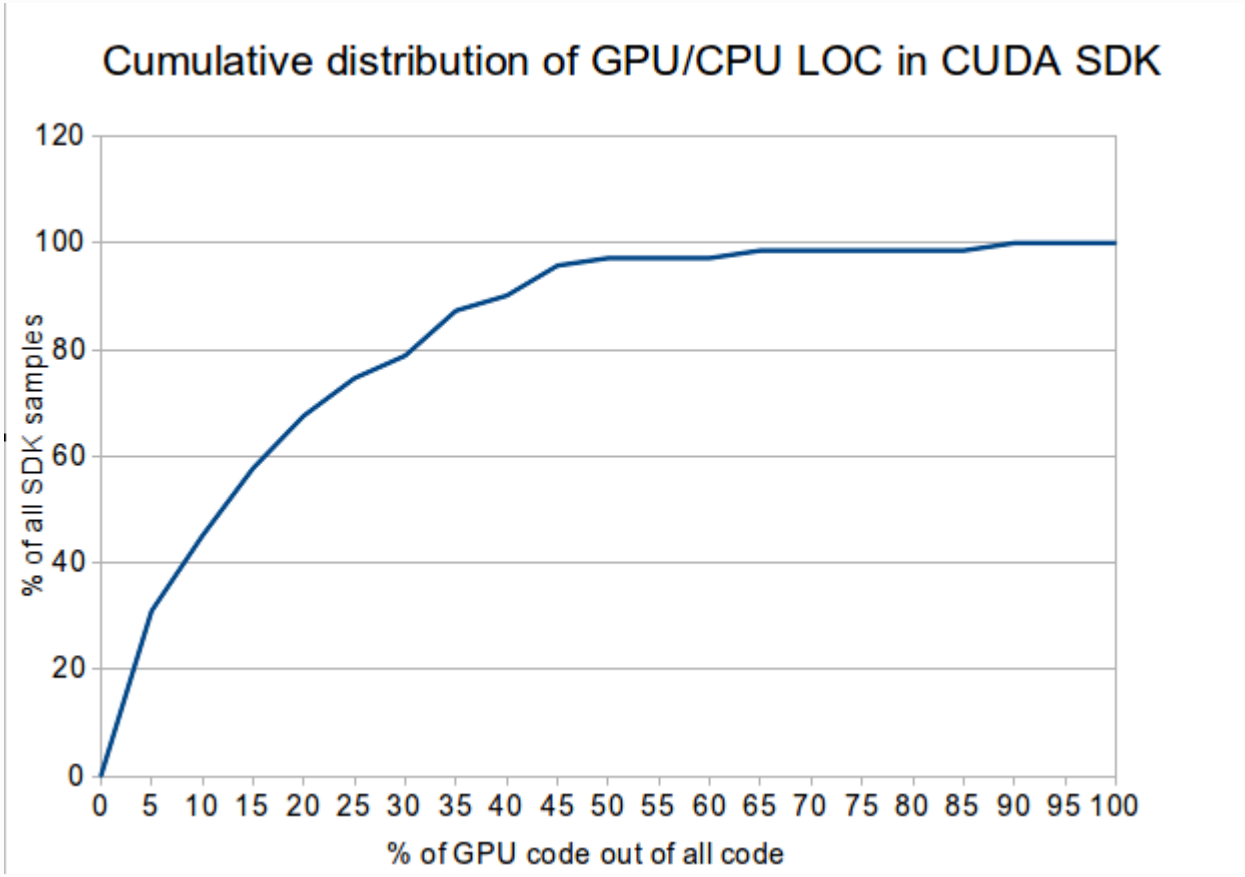
Adding new GPU module (decryption) requires rewrite



Adding another GPU requires redesign

Image classification server




Cumulative distribution of GPU/CPU LOC in CUDA SDK
120
100
80
60
40
20
0
% of all SDK samples
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
% of GPU code out of all code