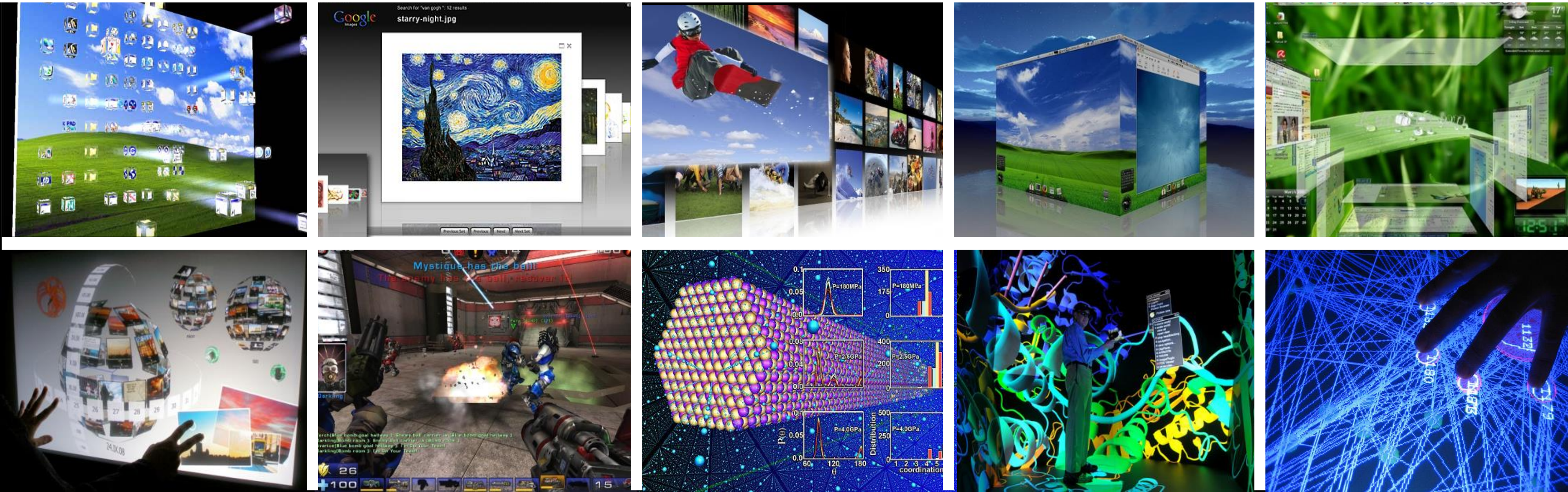
Gdev: First-Class GPU Resource Management in the Operating System

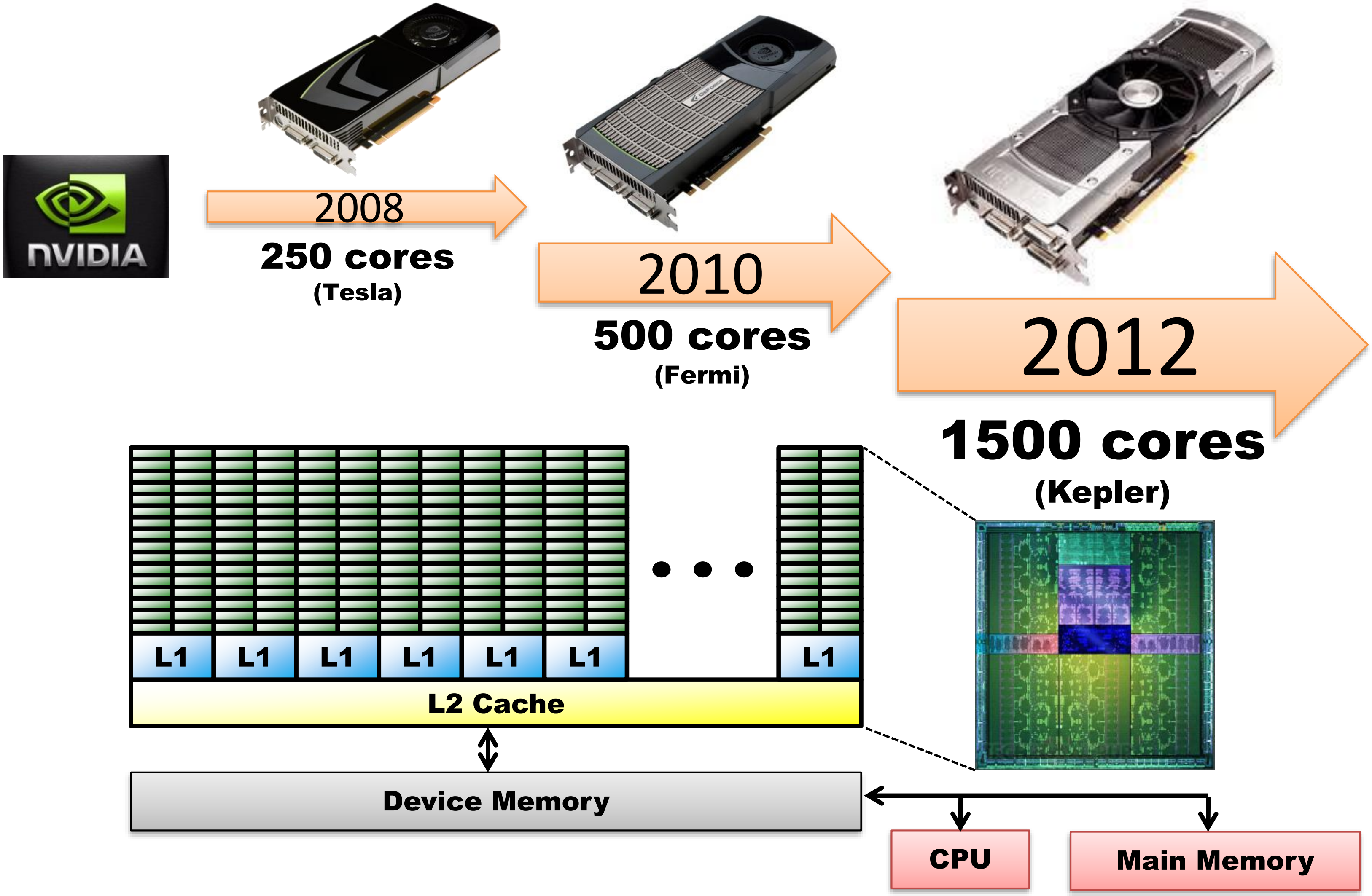
Shinpei Kato
Nagoya University

Michael McThrow Carlos Maltzahn Scott Brandt
UC Santa Cruz



GPUs embrace “many cores”.

Graphics Processing Unit (GPU)



2008

250 cores

(Tesla)

2010

500 cores

(Fermi)

2012

1500 cores

(Kepler)

L1 **L1** **L1** **L1** **L1** **L1** • • • **L1**

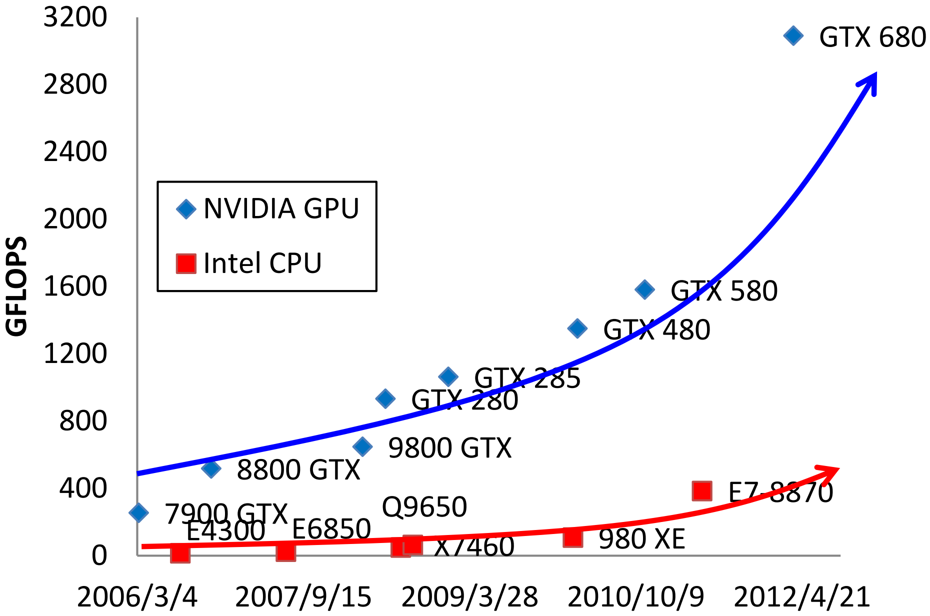
L2 Cache

Device Memory

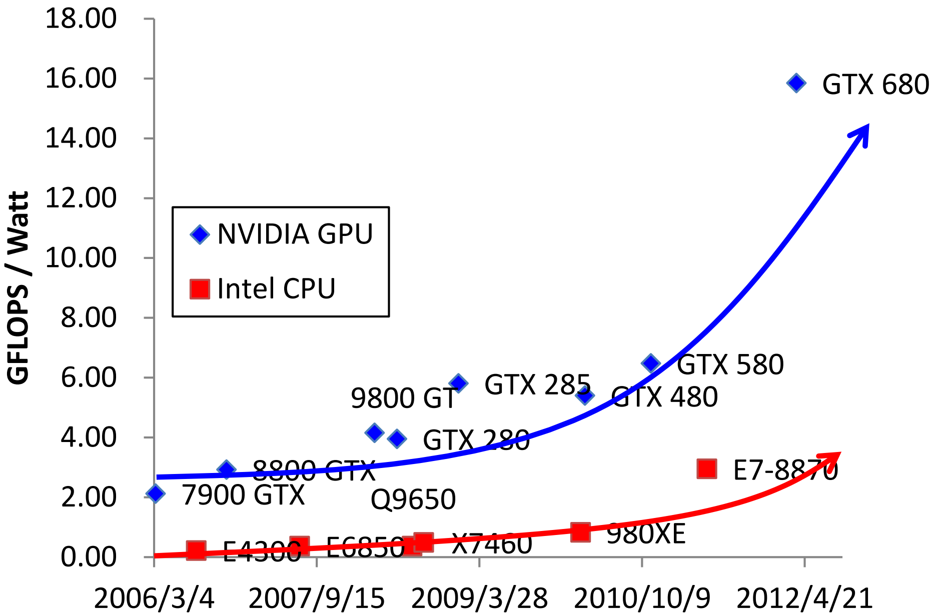
CPU

Main Memory

Performance Trend

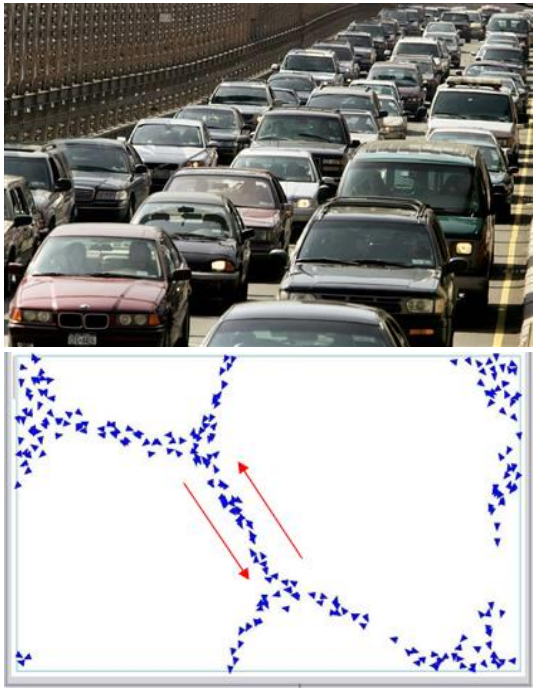


Single Precision Performance

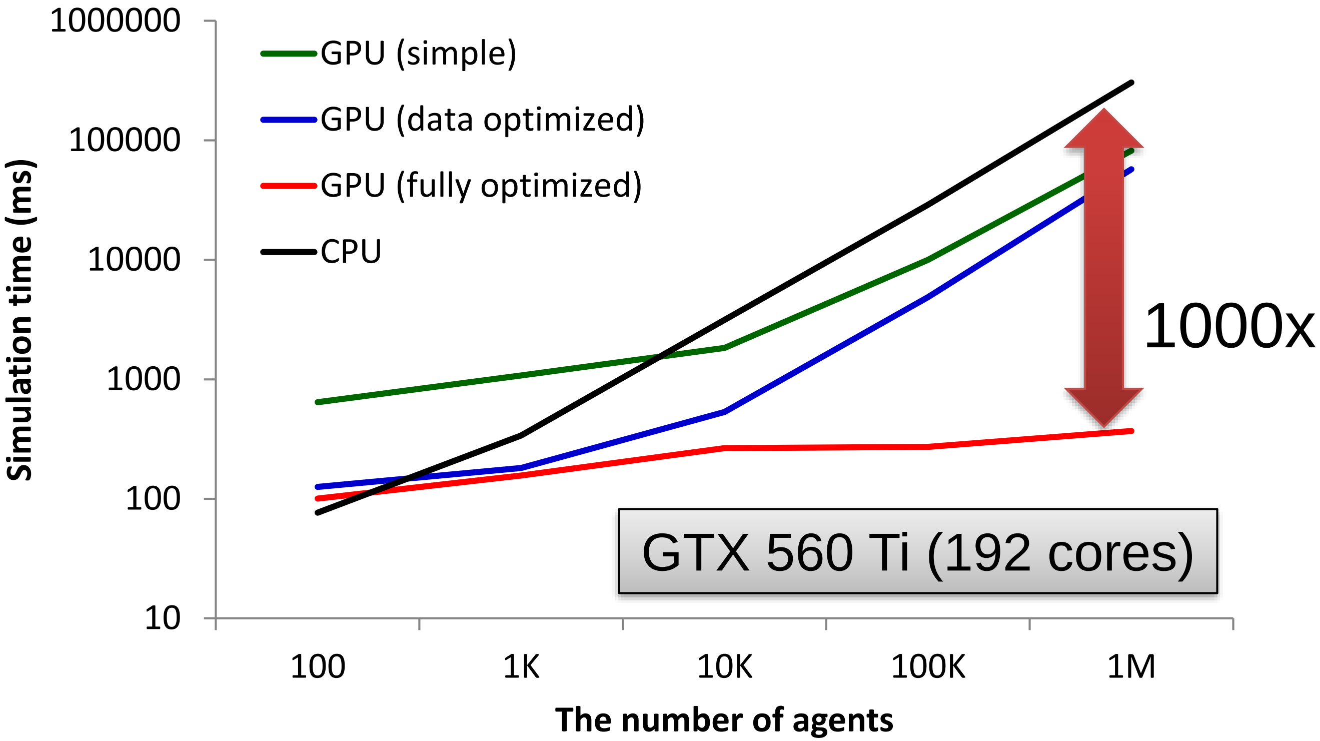


Performance per Watt

GPUs Suit Science



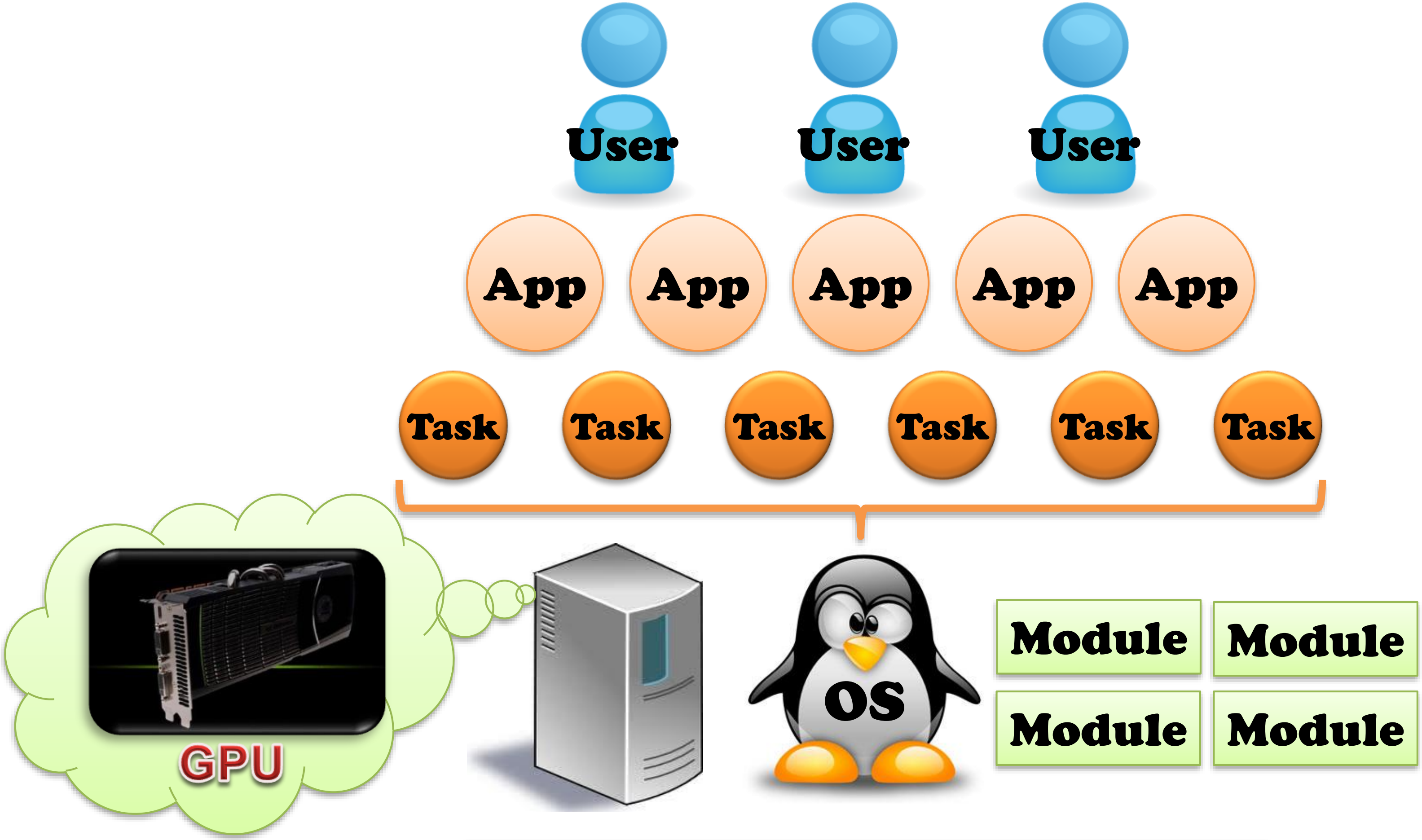
Traffic Simulation



1000x

GTX 560 Ti (192 cores)

Not Yet “General-Purpose”



User User User

App App App App App

Task Task Task Task Task Task

GPU

OS

Module Module

Module Module

GPU Resource Management

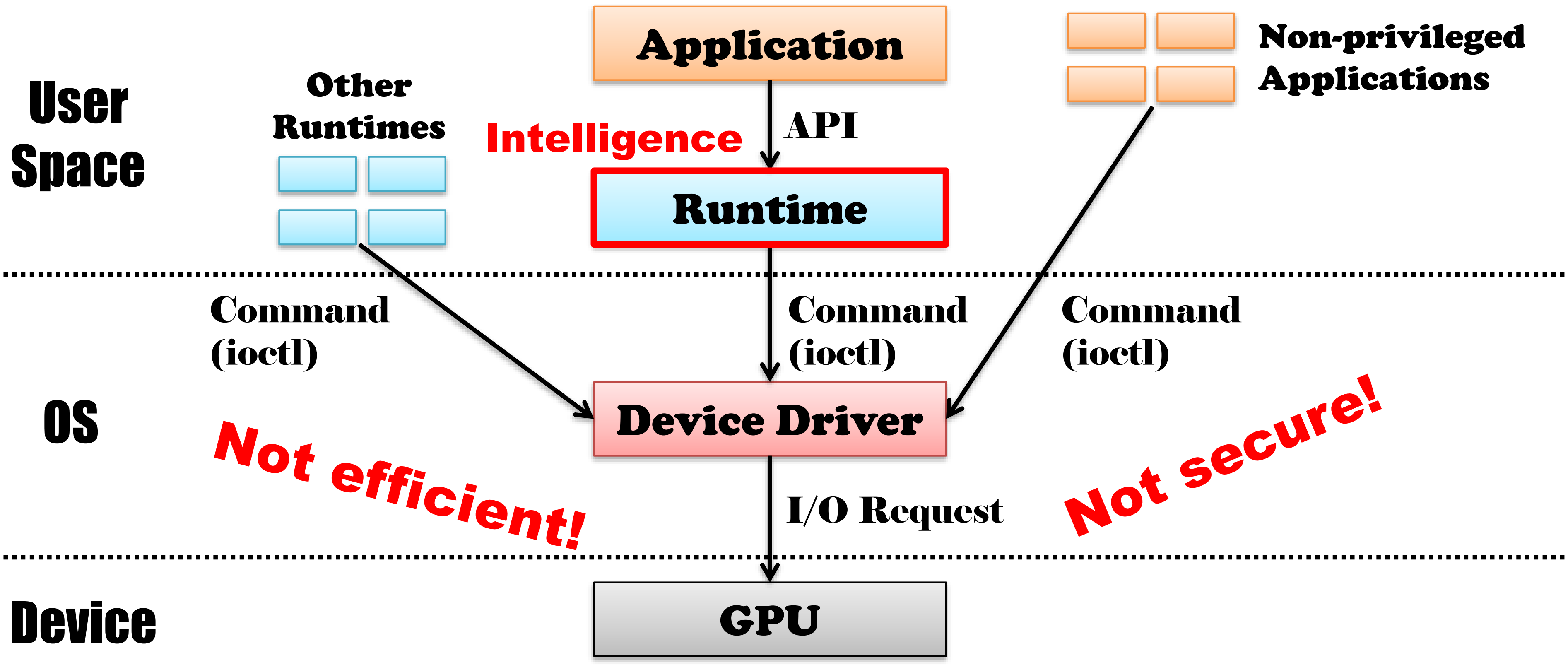
Gdev

- New approach to GPU resource management
  - – Allows the OS as well as user-space applications to use GPUs.
- New functions of GPU resource management
  - – Shared device memory (IPC)
  - – Data swapping
  - – System-level virtualization
- Open-source implementation

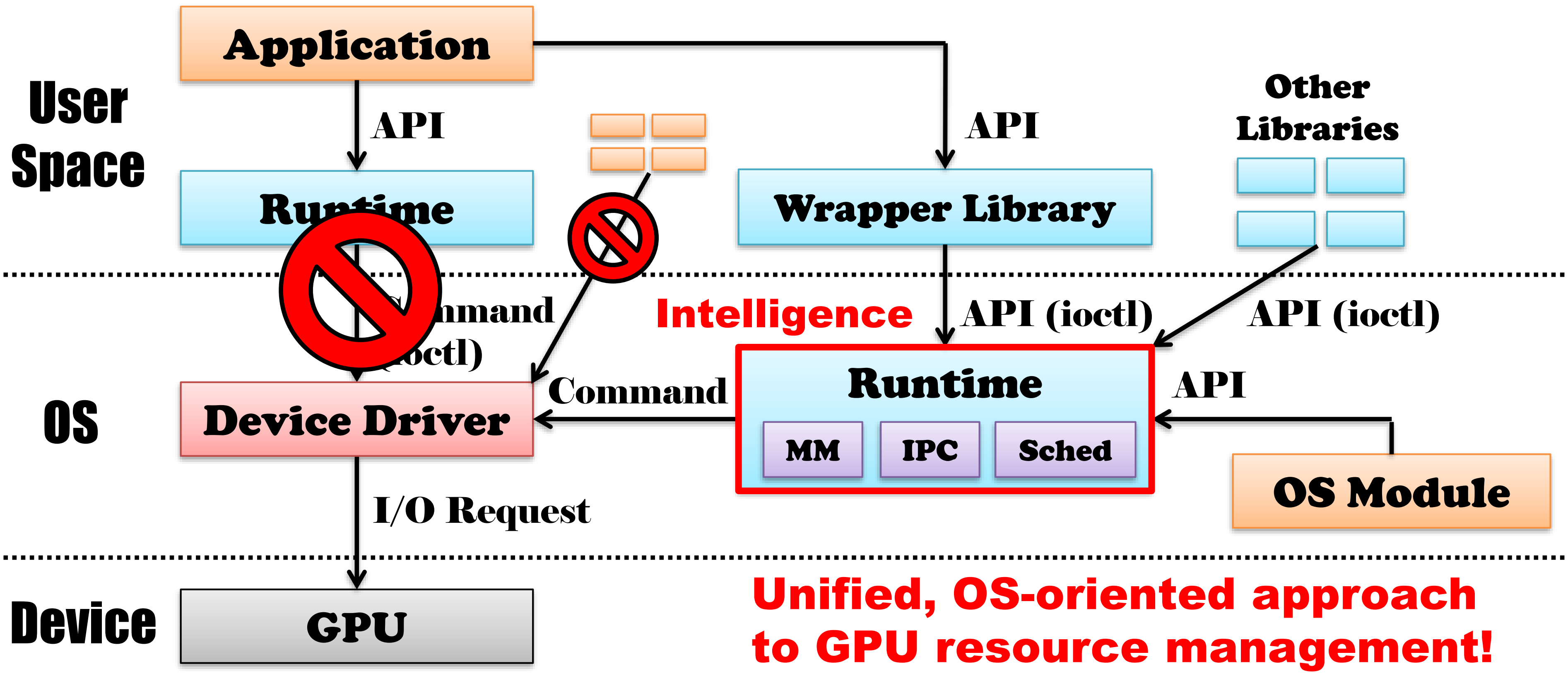
Outline

- Motivation
- Approach
- GPU Resource Management
- Evaluation
- Conclusion

Traditional Naïve Approach


User Space
OS
Device
Other Runtimes
Application
API
Intelligence
Runtime
Non-privileged Applications
Command (ioctl)
Command (ioctl)
Command (ioctl)
Device Driver
I/O Request
GPU
Not efficient!
Not secure!

Gdev Approach

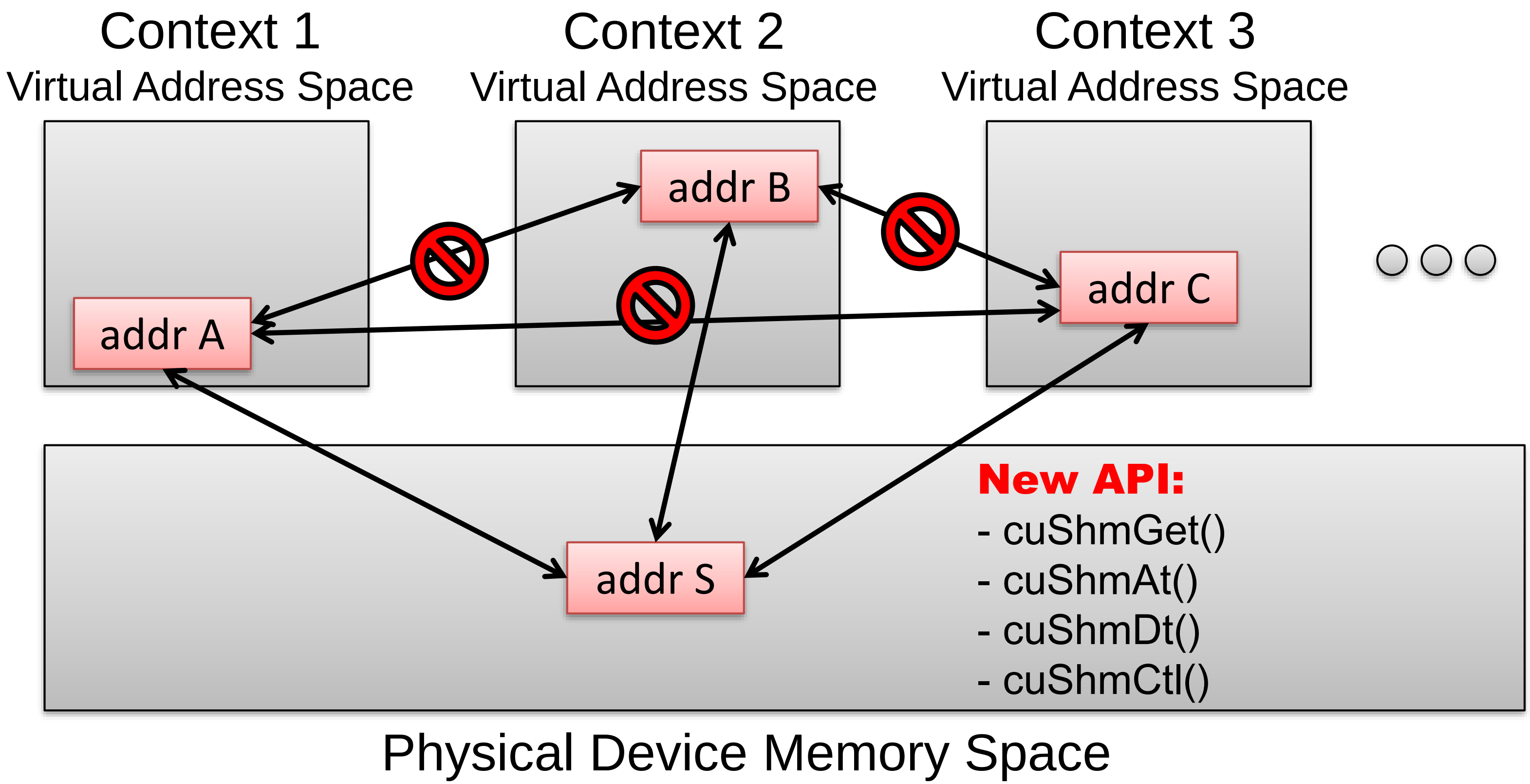

User Space
OS
Device
Application
API
Runtime
mmand
ioctl)
Device Driver
I/O Request
GPU
Command
API
Wrapper Library
Other Libraries
Intelligence
API (ioctl)
API (ioctl)
Runtime
MM
IPC
Sched
API
OS Module

Unified, OS-oriented approach to GPU resource management!

Outline

- Motivation
- Approach
- GPU Resource Management
- Evaluation
- Conclusion

Shared Device Memory



Context 1
Virtual Address Space

Context 2
Virtual Address Space

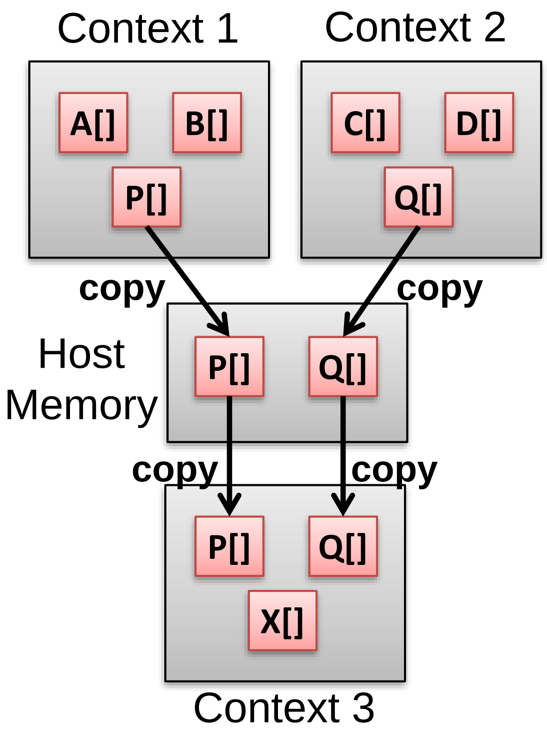
Context 3
Virtual Address Space

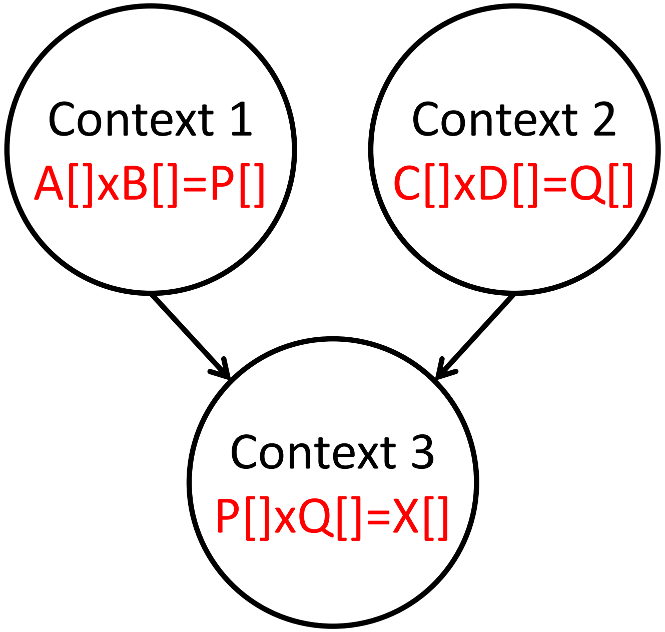
New API:
- cuShmGet()
- cuShmAt()
- cuShmDt()
- cuShmCtl()

Physical Device Memory Space

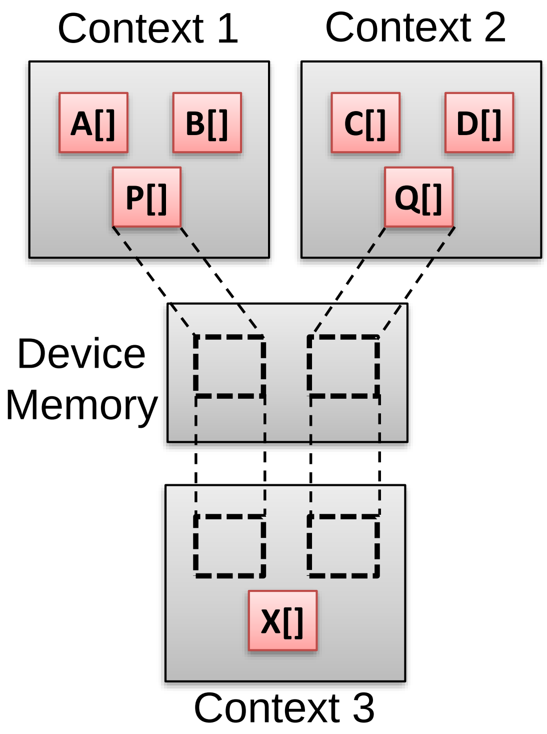
E.g., Dataflow (2x2 Tree)

No Shared Memory

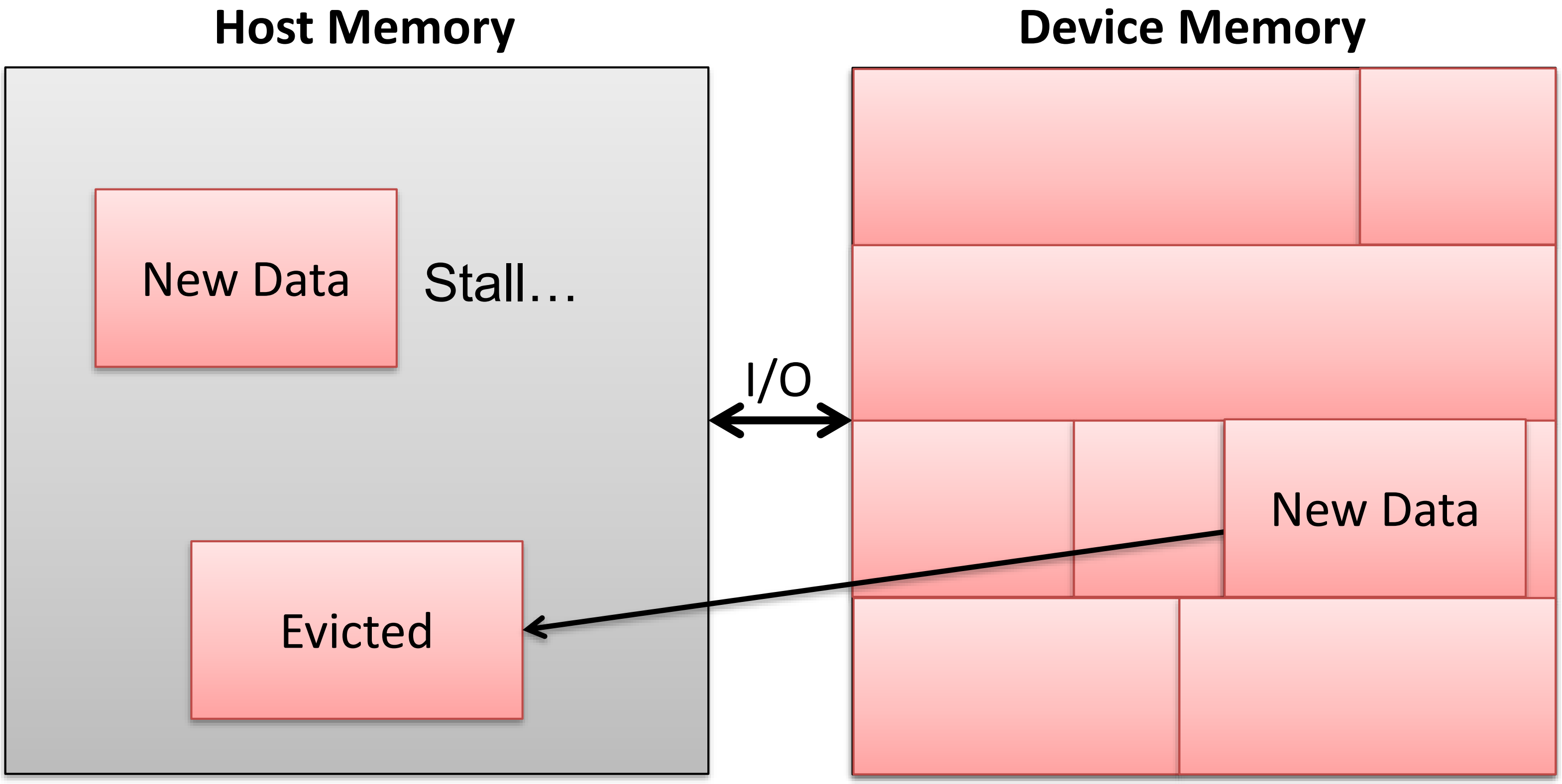

Context 1
Context 2
A[]
B[]
P[]
C[]
D[]
Q[]
copy
copy
Host
Memory
P[]
Q[]
copy
copy
P[]
Q[]
X[]
Context 3


Context 1
A[]xB[]=P[]
Context 2
C[]xD[]=Q[]
Context 3
P[]xQ[]=X[]

Shared Memory IPC


Context 1
Context 2
A[]
B[]
P[]
C[]
D[]
Q[]
Device
Memory
X[]
Context 3

Data Swapping


Host Memory

Device Memory

New Data

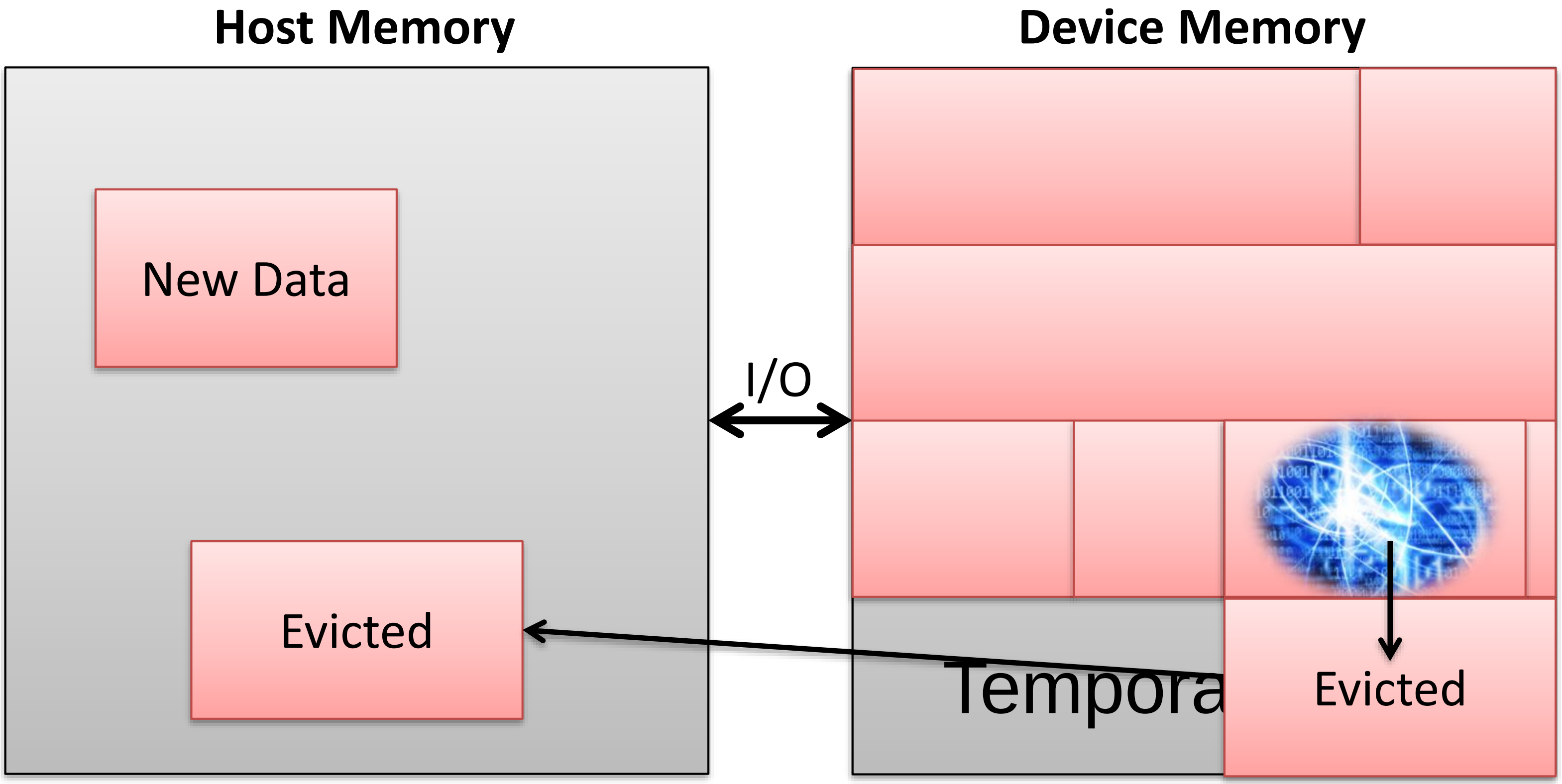
Stall…

I/O

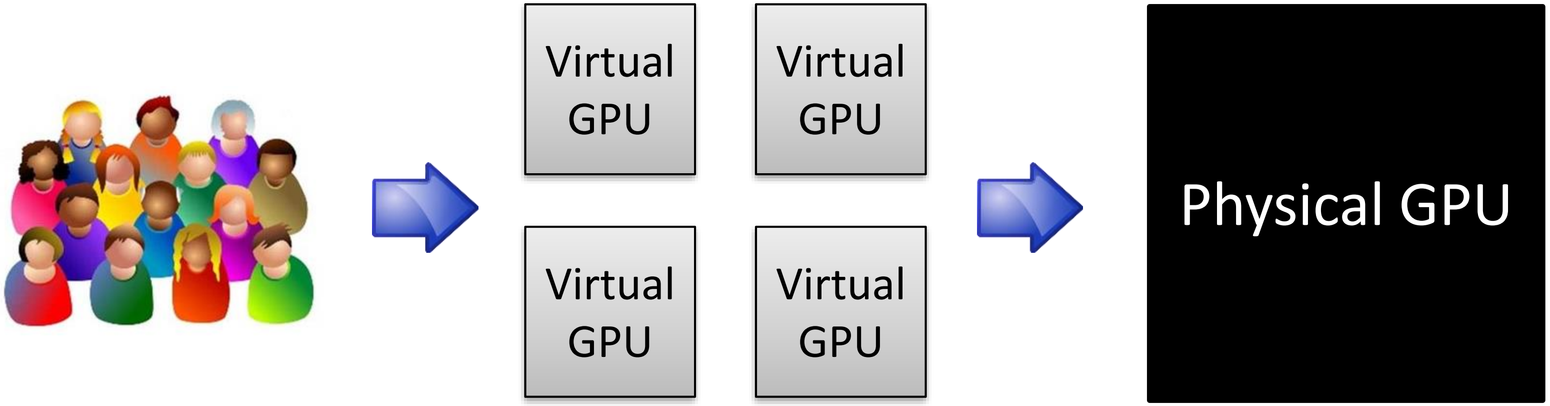
New Data

Evicted

Data Swapping (Enhanced)


Host Memory
Device Memory
New Data
Evicted
I/O
Tempora
Evicted

GPU Virtualization



Virtual GPU

Virtual GPU

Virtual GPU

Virtual GPU

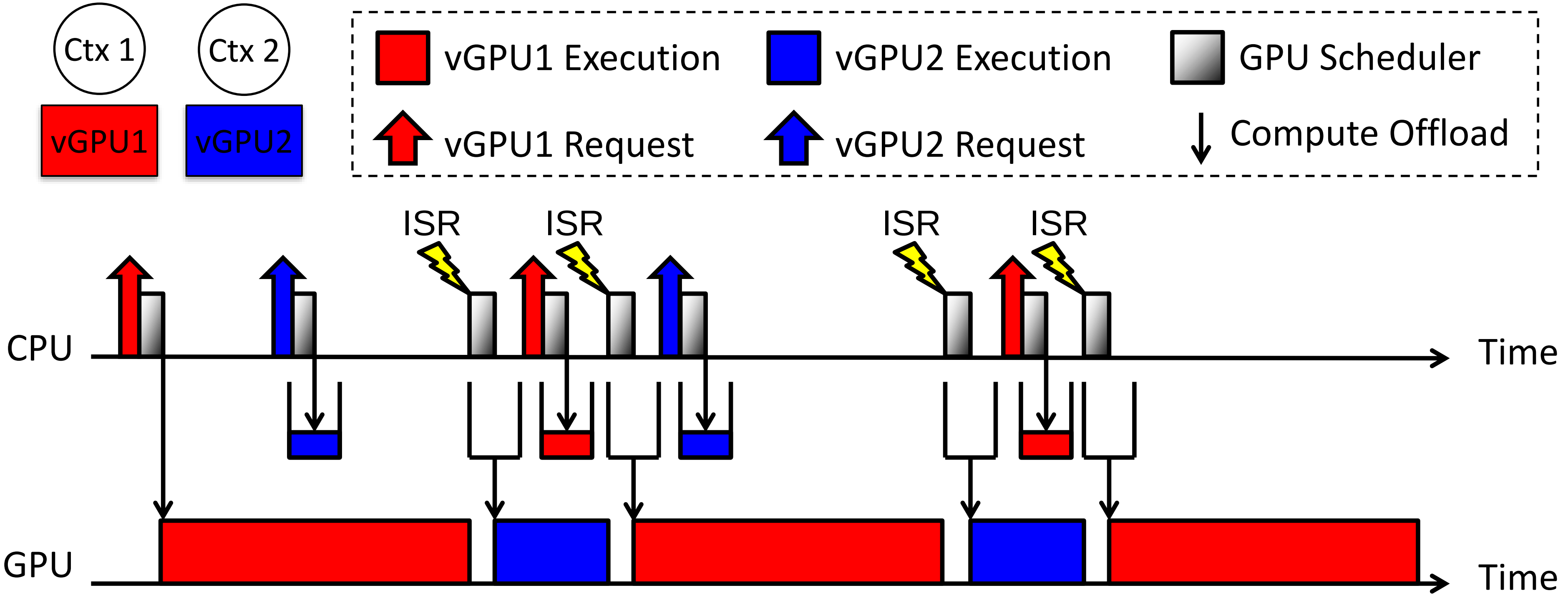
Physical GPU

`/dev/gdev0`
`/dev/gdev1`
`/dev/gdev2`
`/dev/gdev3`
…

`/dev/dri/card0`
(real device file)

Existing GPU Schedulers

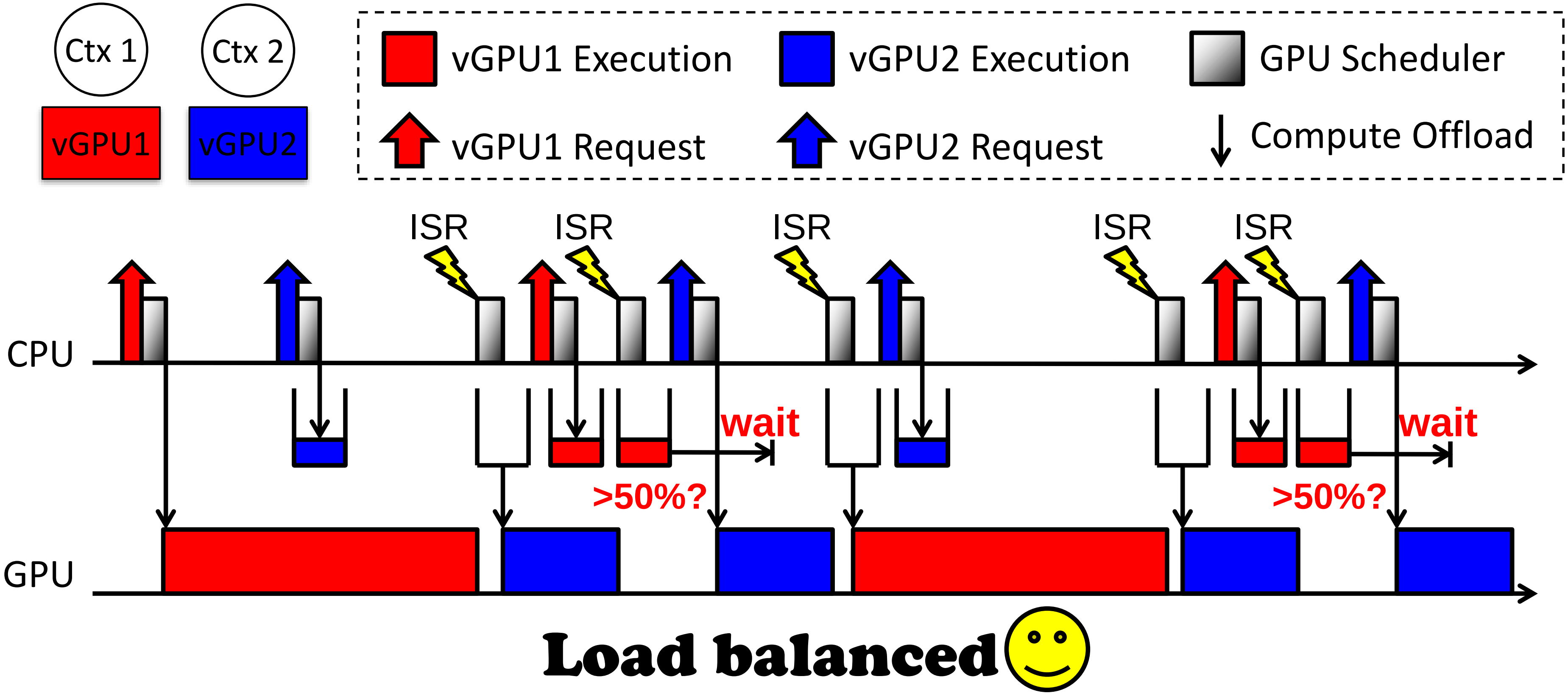
Queue and dispatch [Kato ATC11] [Kato RTSS11]


Ctx 1
Ctx 2
vGPU1
vGPU2
ISR
ISR
ISR
ISR
CPU
Time
GPU
Time

🟥 vGPU1 Execution
🟦 vGPU2 Execution
▨ GPU Scheduler
⇧ vGPU1 Request
⇧ vGPU2 Request
↓ Compute Offload

Load unbalanced!

Bandwidth-aware non-preemptive device (BAND) Scheduler


Ctx 1
Ctx 2
vGPU1
vGPU2
vGPU1 Execution
vGPU2 Execution
GPU Scheduler
vGPU1 Request
vGPU2 Request
Compute Offload
ISR
ISR
ISR
ISR
ISR
CPU
wait
wait
>50%?
>50%?
GPU

Load balanced ☺

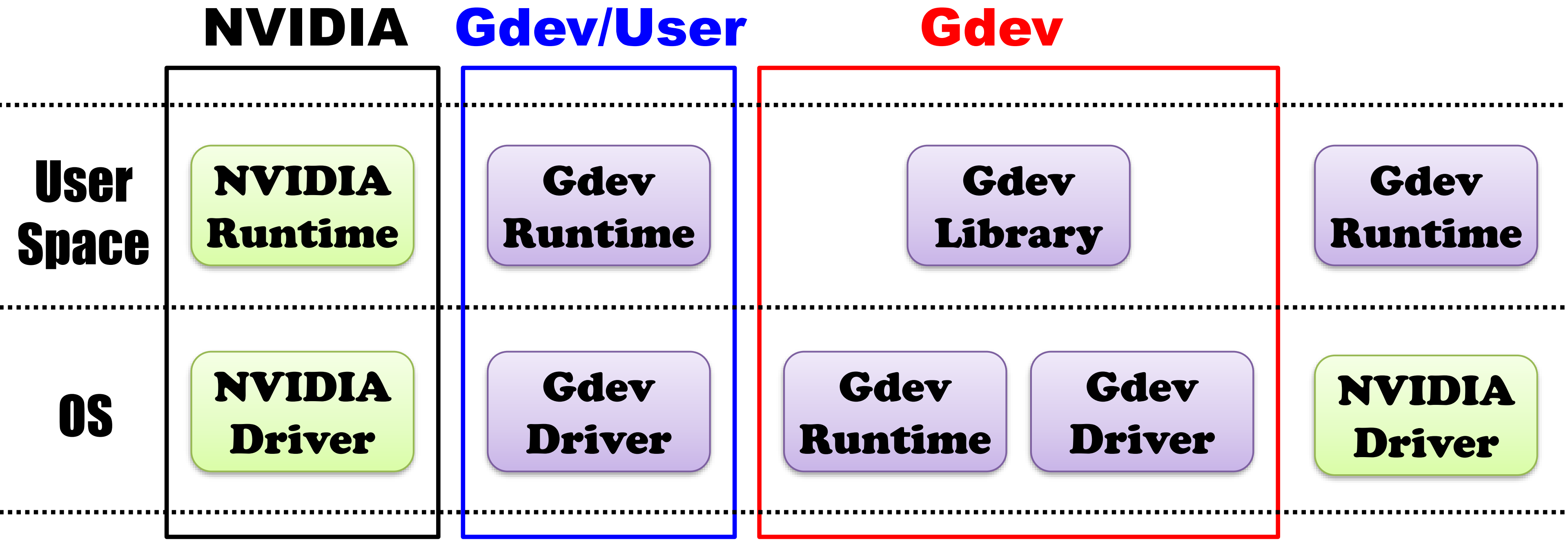
Outline

- Motivation
- Approach
- GPU Resource Management
- Evaluation
- Conclusion

Experimental Setup

- Linux kernel 2.6.39
- NVIDIA GeForce GTX 480
- Intel Core 2 Extreme QX9650
- NVIDIA CUDA Compiler 4.0 and GCC 4.4.6
- Benchmarks & Applications:
  - Rodinia benchmark [Che *et al*, IISWC’09]
  - eCryptfs encrypted filesystem
  - FAST database search [Kim *et al*, SIGMOD’10]
  - PTask dataflow benchmarks [Rossbach *et al*, SOSP’11]

Runtime and Driver Choice



NVIDIA

Gdev/User

Gdev

User Space

OS

NVIDIA Runtime

NVIDIA Driver

Gdev Runtime

Gdev Driver

Gdev Library

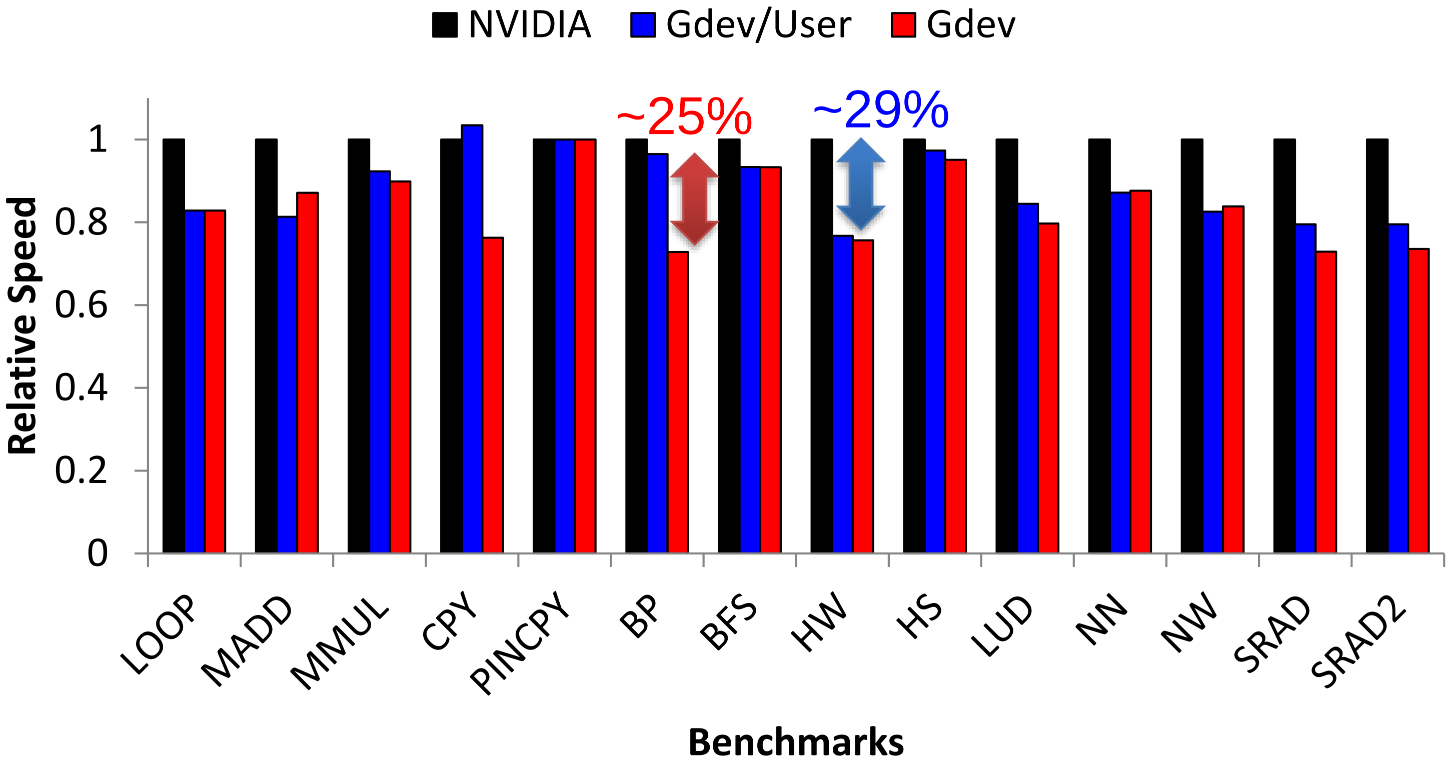
Gdev Runtime

Gdev Driver

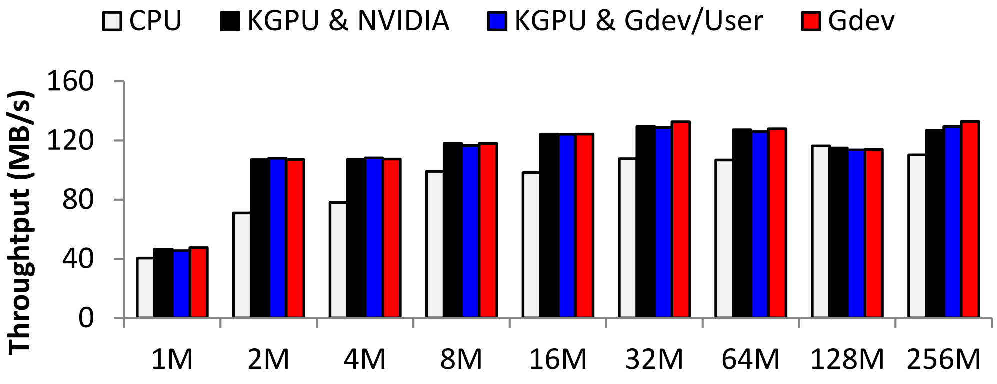
Gdev Runtime

NVIDIA Driver

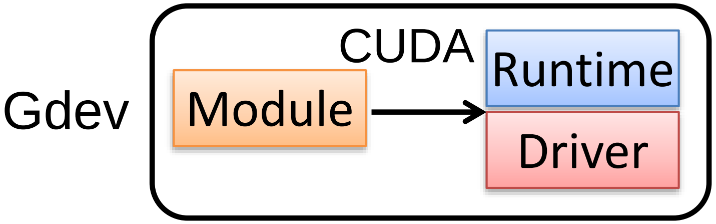
Basic Performance



eCryptfs Read&Write Throughput



Read throughput


Gdev

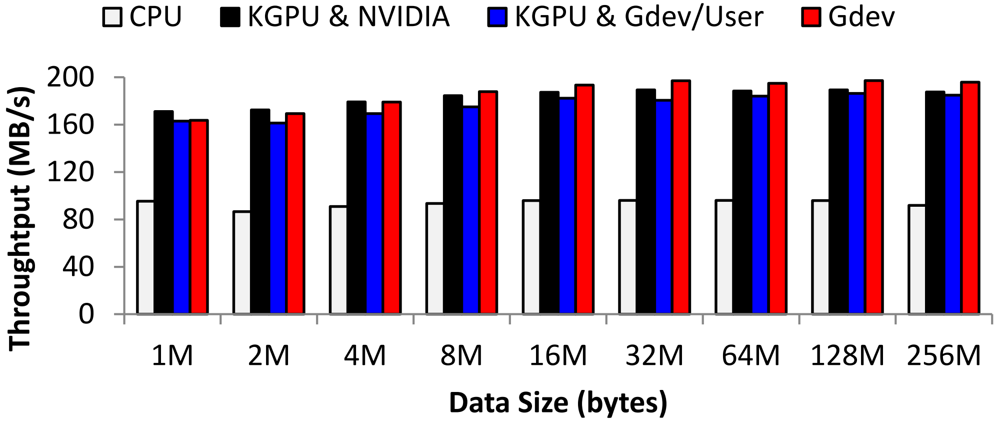
Module

CUDA

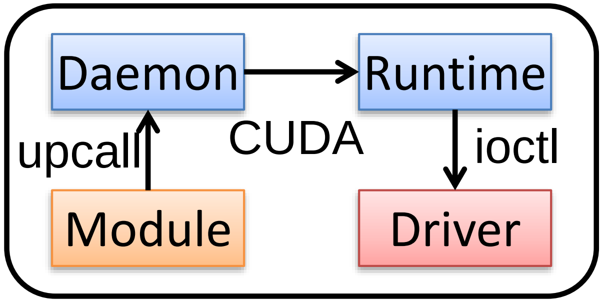
Runtime

Driver

Write throughput



KGPU
Sun *et. al.*
2012


Daemon

Runtime

upcall

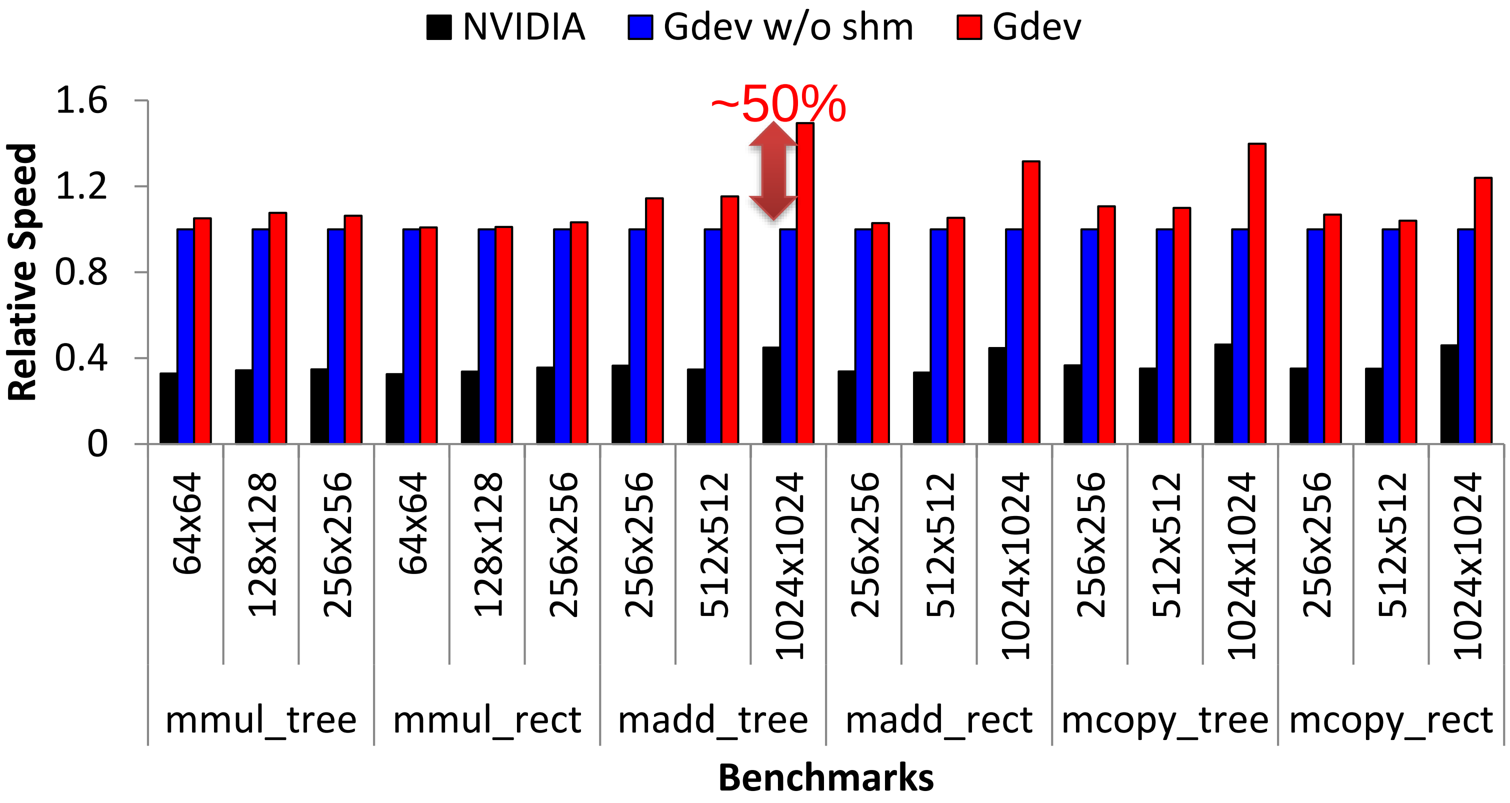
CUDA

ioctl

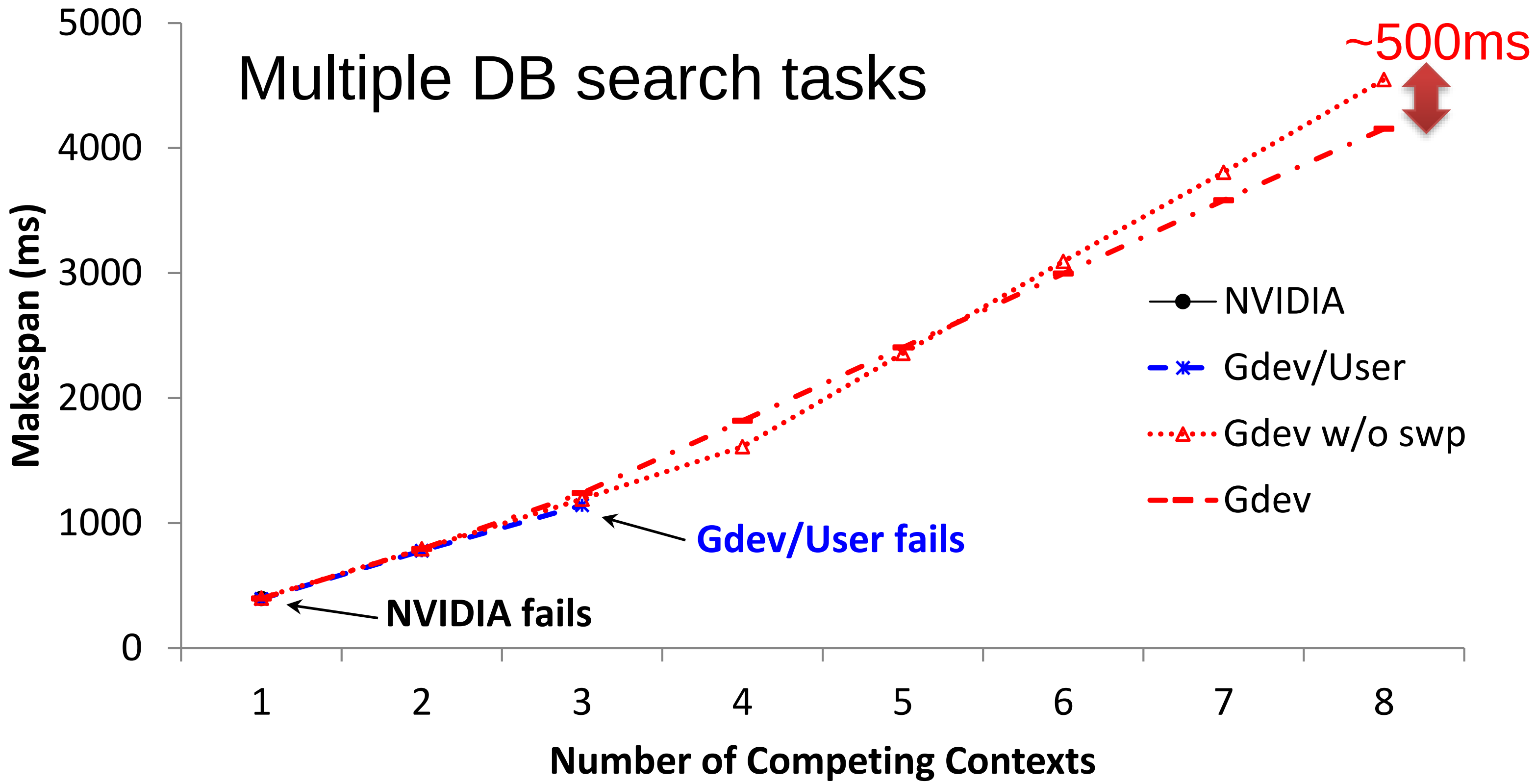
Module

Driver

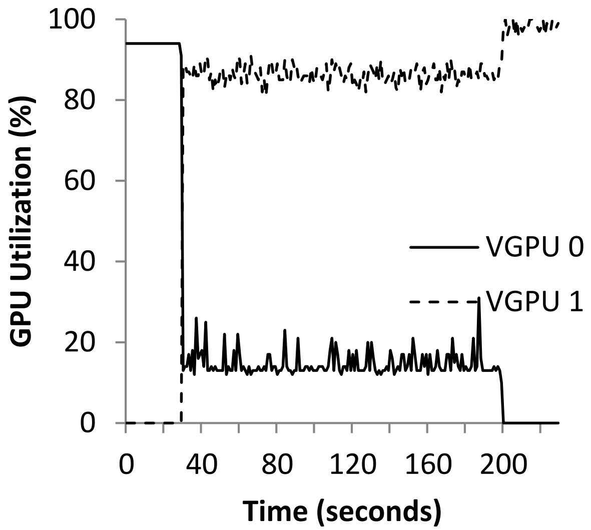
Impact of Shared Device Memory



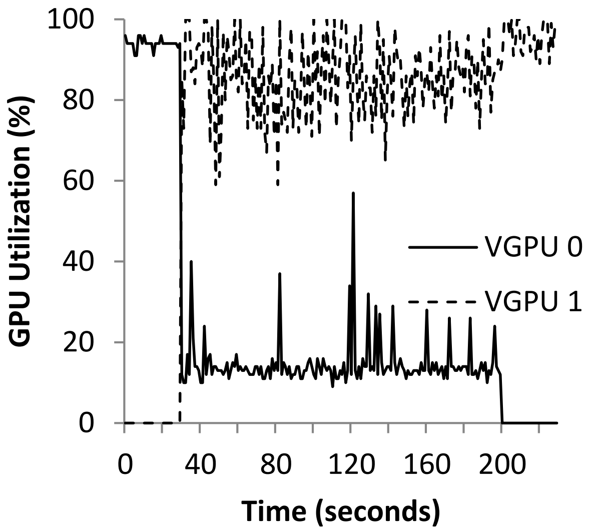
Impact of Data Swapping



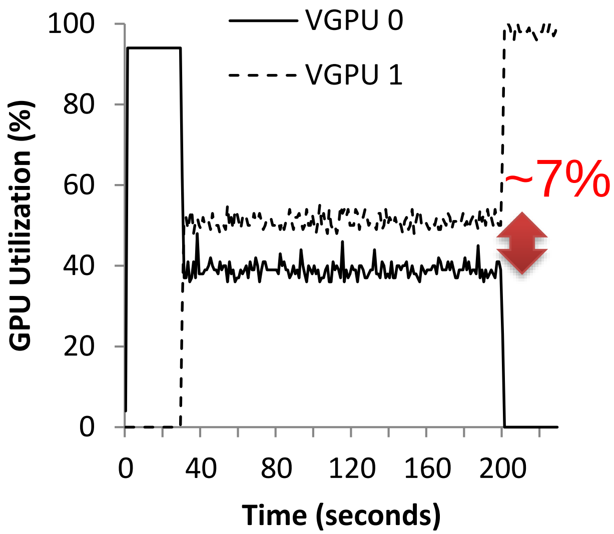
Virtual GPU Isolation



No scheduling
(FIFO)



Xen VM Policy
(Credit)



Gdev Policy
(BAND)

Outline

- Motivation
- Approach
- GPU Resource Management
- Evaluation
- **Conclusion**

Concluding Remarks

- Gdev is an OS approach to **first-class** GPU resource management.
  - GPUs can be used by the OS.
  - GPUs can be protected by the OS.
  - GPUs can be multi-tasked by the OS.
  - Compromising basic performance to some extent.

Concluding Remarks

Gdev is **open-source**.

Facilitate systems research.

Visit http://sys.ertl.jp/gdev/.

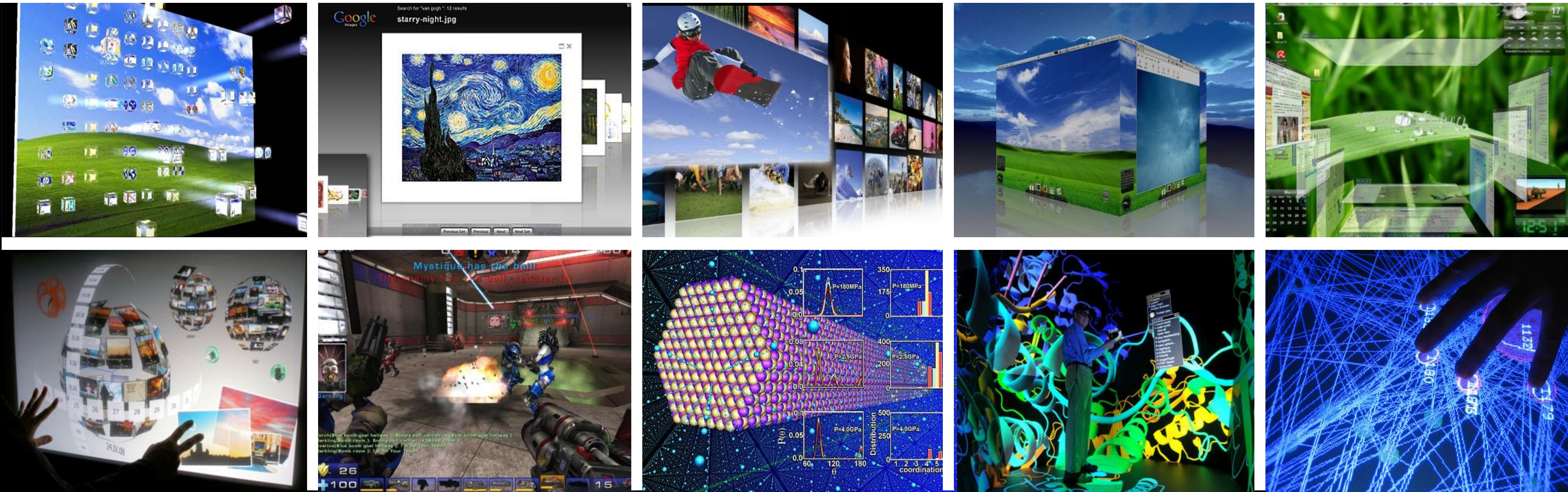
What's up-to-date:

- RAID6 erasure coding acceleration.
- Dynamic power management.
- Zero-copy between I/O devices and GPUs.

NVIDIA
GEFORCE GTX 280
EVGA
GEFORCE 9500 GT
NVIDIA
GEFORCE GTX 470M

Thank You!

Questions?


Google
starry-night.jpg
Mystique has the ball!
P=180MPa
P=2.5GPa
P=4.0GPa
coordination