Intro to GeMTC and Swift
+
GeMTC and Swift/T Tutorial

Scott Krieder

About Me

- 3rd year PhD Student
- Research Assistant DataSys Laboratory
- Teaching Assistant, Dept. Computer Science
  - CS350, CS351, CS450 (involved w/ CS550, CS554)
- IIT Starr/Fieldhouse Research Fellow
  - In collaboration w/ Argonne National Laboratory and Computation Institute (UChicago)
- Guest Graduate Student, Argonne National Lab
- Research
  - Many-Task Computing
  - Hardware Accelerators
  - HPC, HTC, Distributed Systems

DataSys Data-Intensive Distributed Systems Laboratory

ILLINOIS INSTITUTE OF TECHNOLOGY

Argonne NATIONAL LABORATORY

Lecture Outline

Lecture: (11:25pm-12:40pm)

- GeMTC (:55)
  - Motivation
    - Distributed Systems, HPC, MTC GPGPU.
  - GeMTC
    - Architecture, Design
    - Apps
  - Future Work
    - MTACS
    - Xeon Phi
- Swift/T (:20)
  - Slides as paper was presented. (CCGrid’13)

Hands on Outline

Hands On: (12:45pm-1:45pm)

- CUDA
  - SDK Examples
    - DeviceQuery
  - Vector-Add
- GeMTC
  - Vector-Add
- Swift/T
  - Vector-Add

Acknowledgements

Thank you to:

Dr. Ioan Raicu - Advisor

Benjamin Grimmer - IIT Undergrad

Dustin Shahidehpour, Jeff Johnson-IIT Alumni(Orbitz, Microsoft)

Dr. Justin Wozniak - ANL Computer Scientist

Michael Wilde - ANL Software Architect & UChicago CI Fellow

ILLINOIS INSTITUTE OF TECHNOLOGY

DataSys Data-Intensive Distributed Systems Laboratory

Argonne NATIONAL LABORATORY

Computation Institute

THE UNIVERSITY OF CHICAGO

Publications

- CCGrid'14 - [Preparing Submission]
- STARR/Fieldhouse Research Fellowship '12-'13
- Scott Krieder, Ioan Raicu, “Towards the Support for Many-Task Computing on Many Core Computing Platforms” - IEEE/ACM Supercomputing 2012 (SC’12) - Salt Lake City, UT (11/2012)
- Scott Krieder, Ioan Raicu - “Early Experiences in running Many-Task Computing workloads on GPUs” - XSEDE 2012 - Chicago, IL (07/2012)
- Scott Krieder, Ioan Raicu - “An Overview of Current and Future Accelerator Architectures” - Greater Chicago Area System Research Workshop - Chicago, IL (05/2012)

Lecture Outline

Lecture: (11:25pm-12:40pm)

- GeMTC (:45)
  - **Motivation**
    - Distributed Systems, HPC, MTC GPGPU.
  - GeMTC
    - Architecture, Design
    - Apps
  - Future Work
    - MTACS
    - Xeon Phi
- Swift/T (:30)
  - Slides as paper was presented. (CCGrid’13)

Distributed Systems

- Many machines
- Network
- Common Goal
- Fault Tolerant
- Heterogeneous

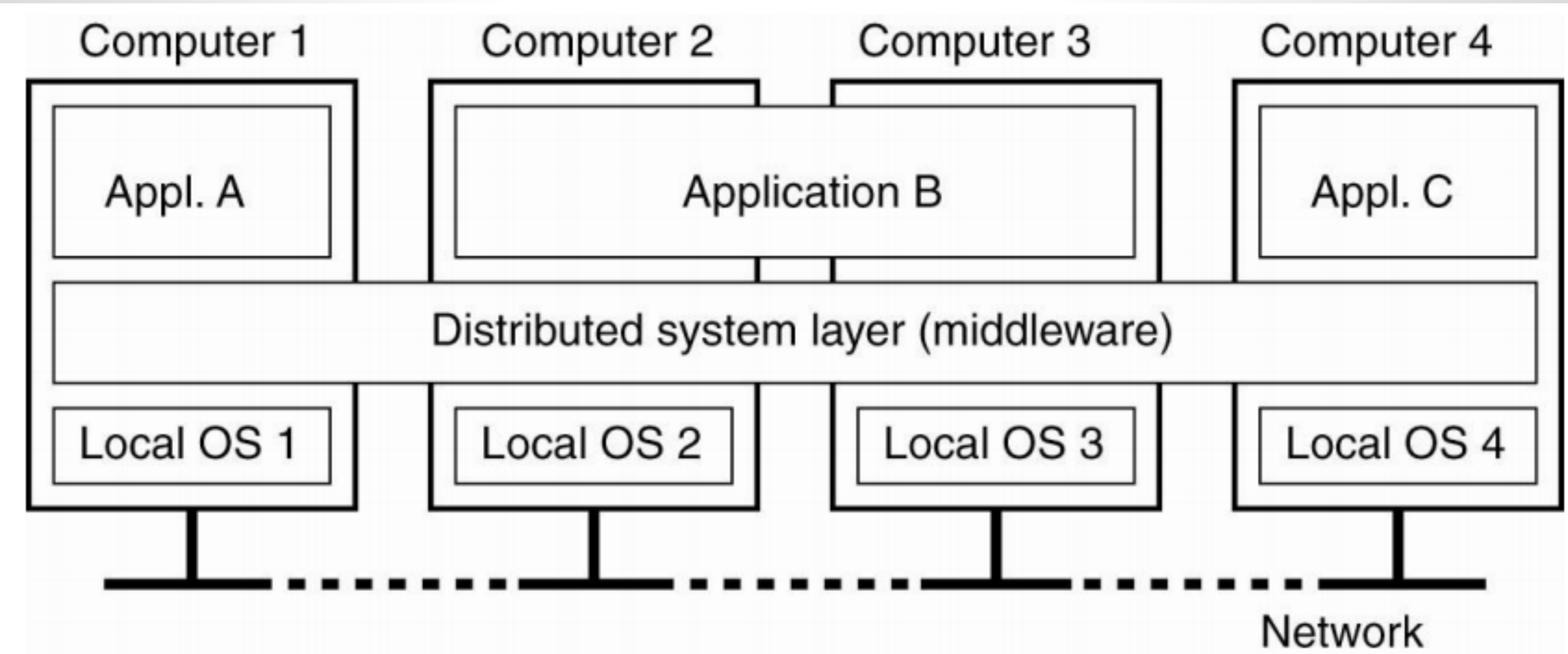


*image from Tannebaum - "Distributed Systems"

Supercomputing

Key Characteristics:

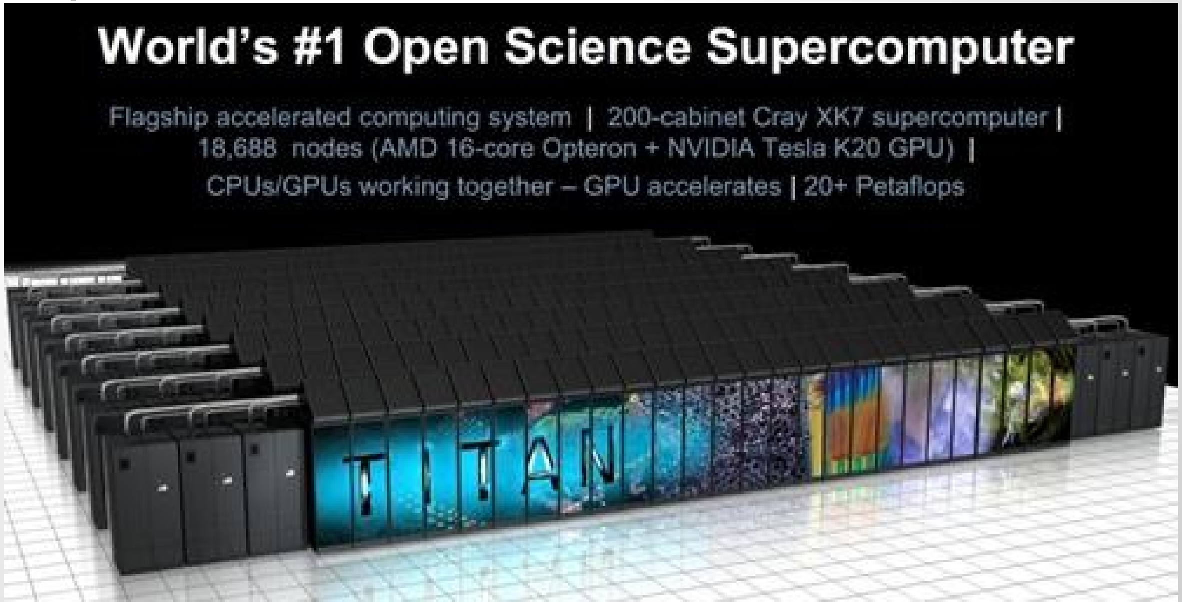
- Evolving from homogeneous to increasingly hybrid
- High speed network / Fast
- High Performance Computing (HPC)

Advantages

- "Tightly Coupled"
- Large Resources
- Short Time
  - hours/days/weeks

Growing Shortcomings:

- programmability
- fault tolerance



High Throughput Computing (HTC)

Key Characteristics:

- Loosely coupled
- Robustness
- Reliability
- Jobs per month/year

Advantages:

- Programmability
- Fault tolerance

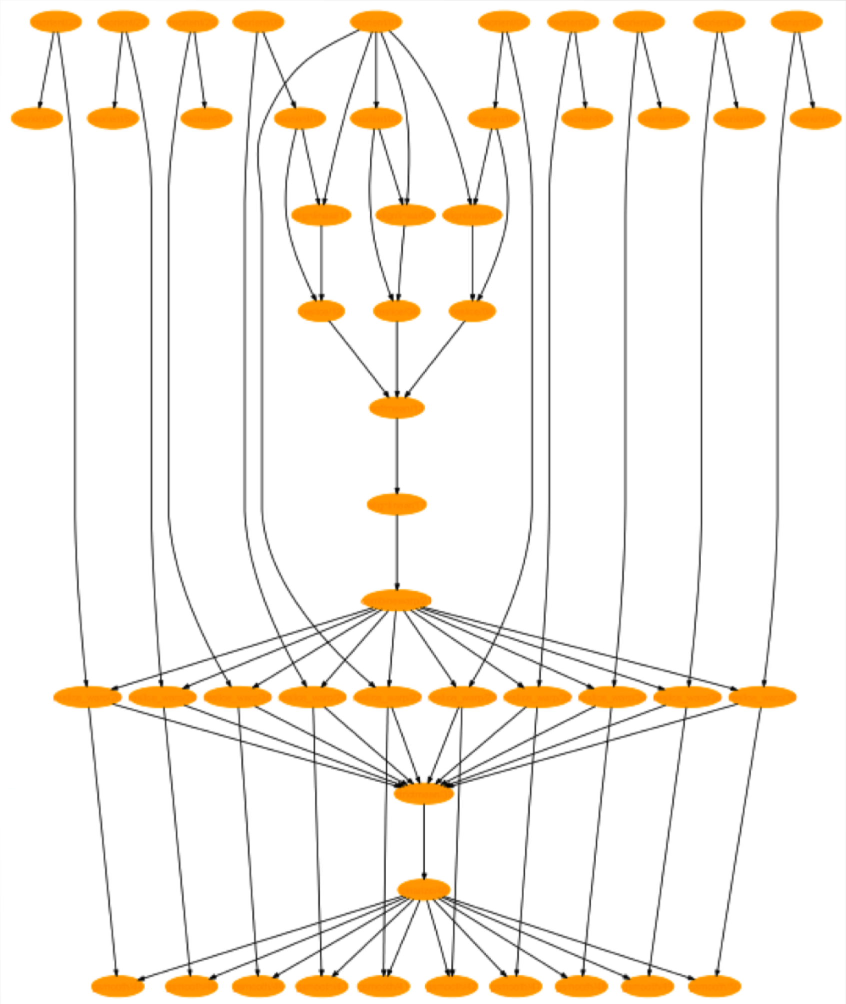
Shortcomings:

- Efficiency
- Large focus on pleasingly parallel
- Bag-of-tasks pattern

Many-Task Computing (MTC)

MTC emphasizes:

- bridging HPC/HTC
- many resources
  - short period of time
- many computational tasks
- dependent/independent tasks
- tasks organized as DAGs
- primary metrics are seconds

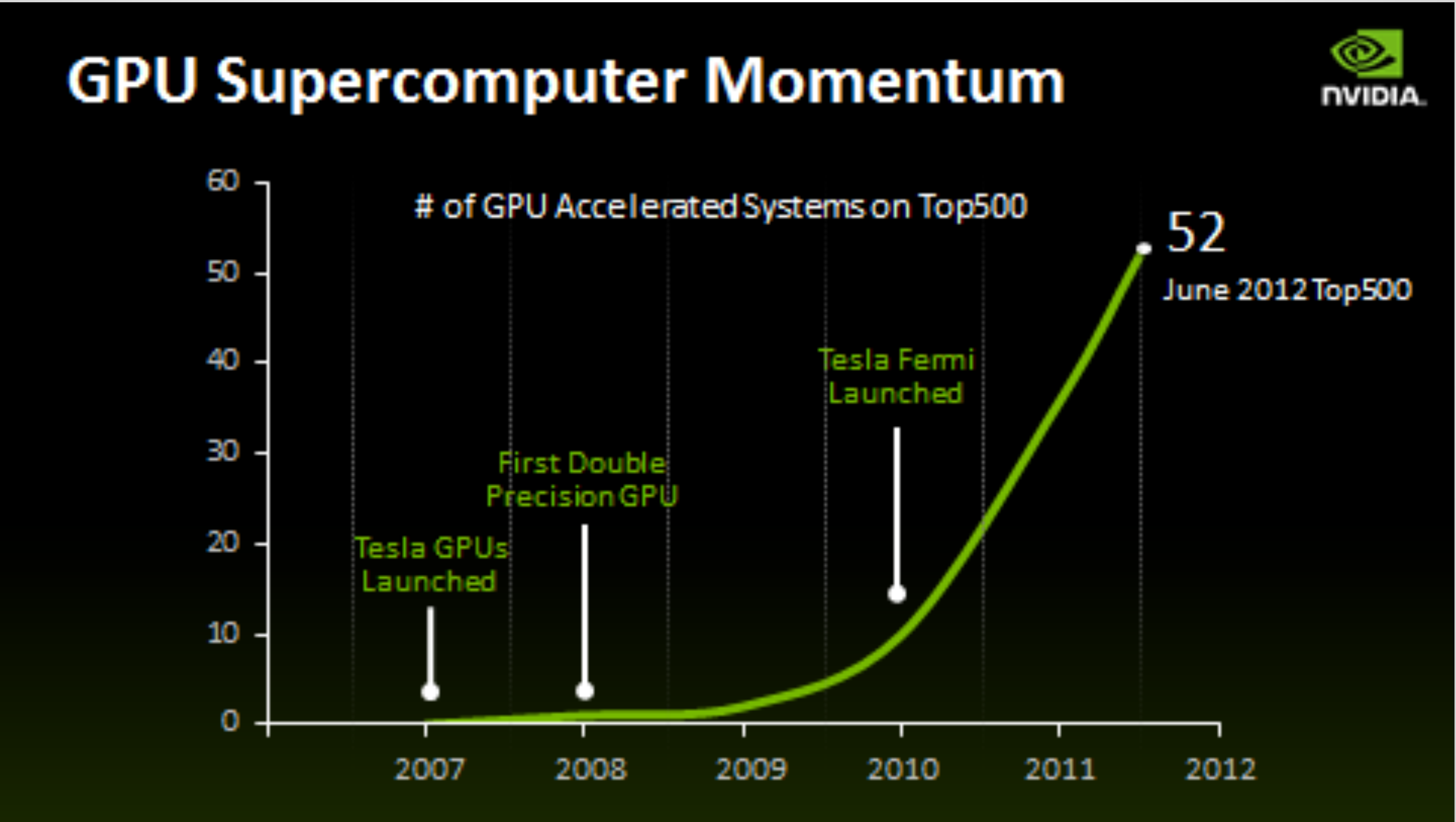


Advantages:

- Improve fault tolerant
- Maintain efficiency
- Programmability & Portability
- support embarrassingly parallel and parallel applications

GPGPU Motivation

- Host CPU offload work to GPU
- Relieves CPU
- 52 in June 2012, 62 in November 2012



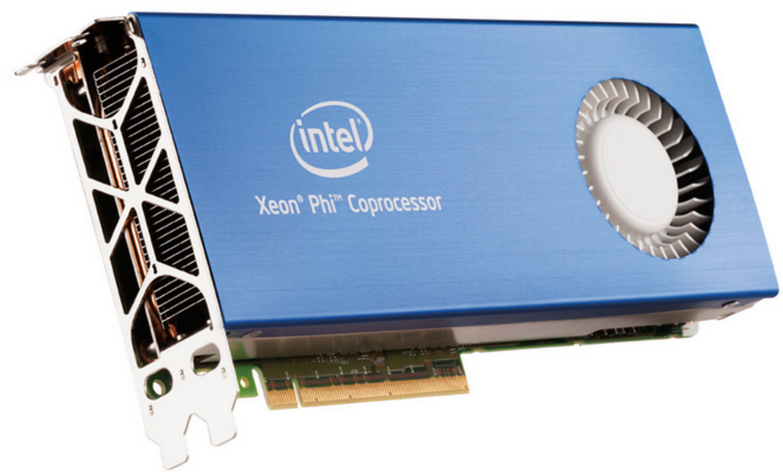
Accelerator Architecture

GPU

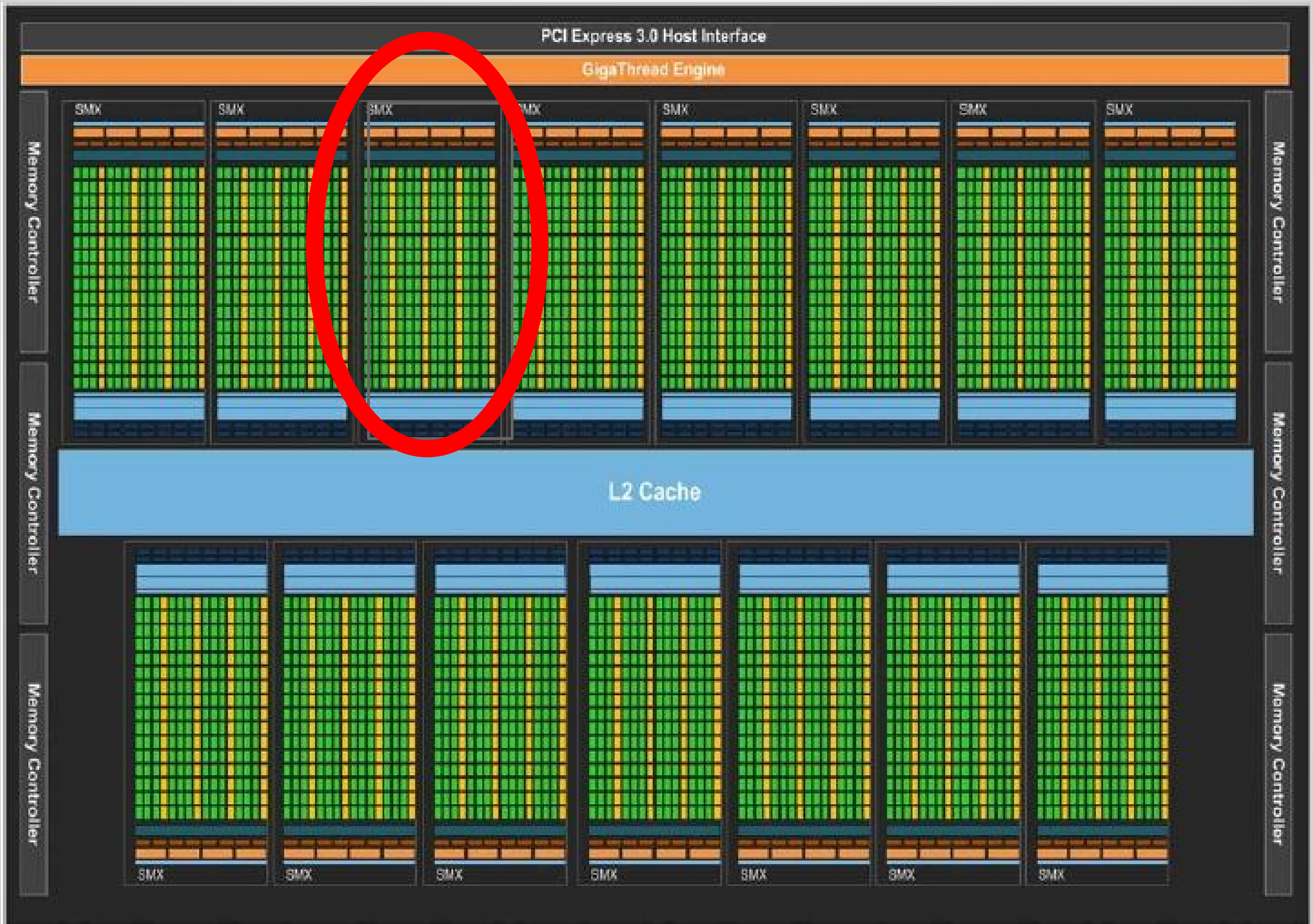
- Streaming Multiprocessors (15 SMXs on Kepler K20)
- Warps
  - 32 threads in a warp
  - 192 warps
    1. hardware available
    2. ind. compute

Coprocessors

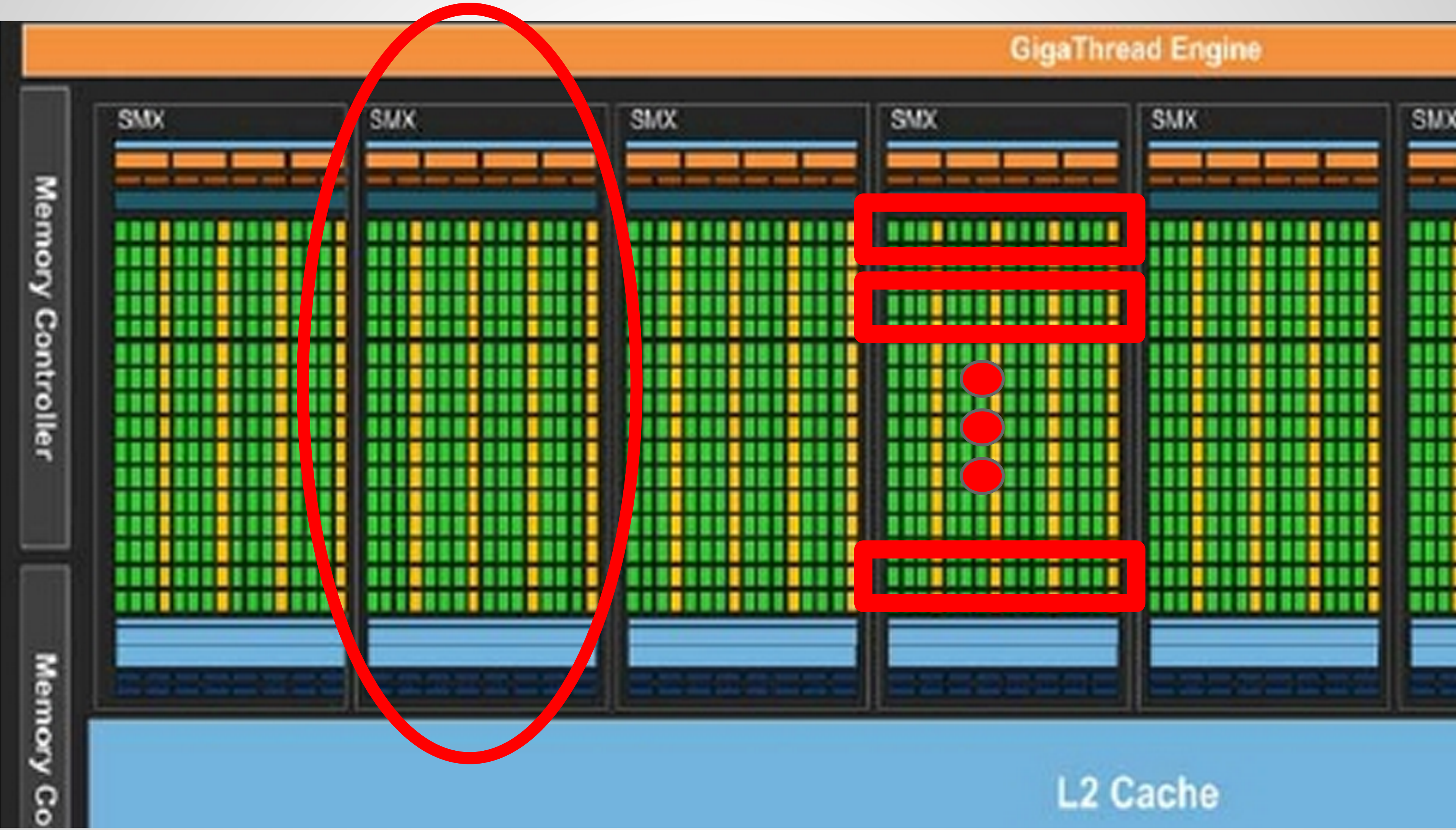
- Intel Xeon Phi
  - 60 cores * 4 threads per core = 240 hardware threads



GPU Block Diagram - Highlighting SMX

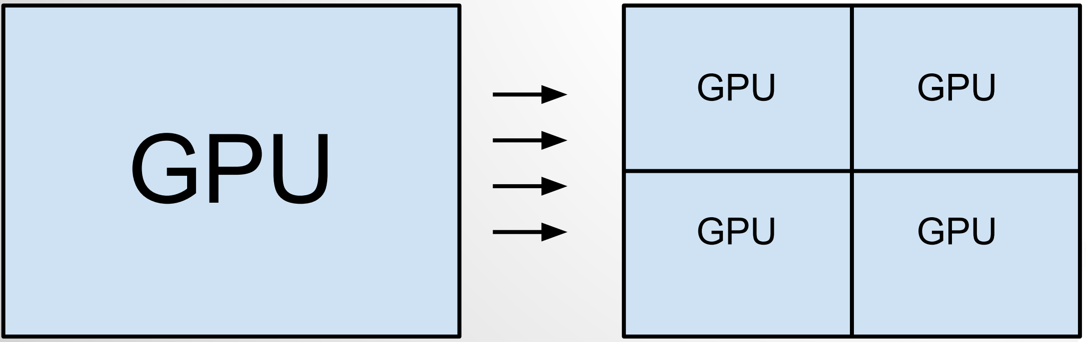


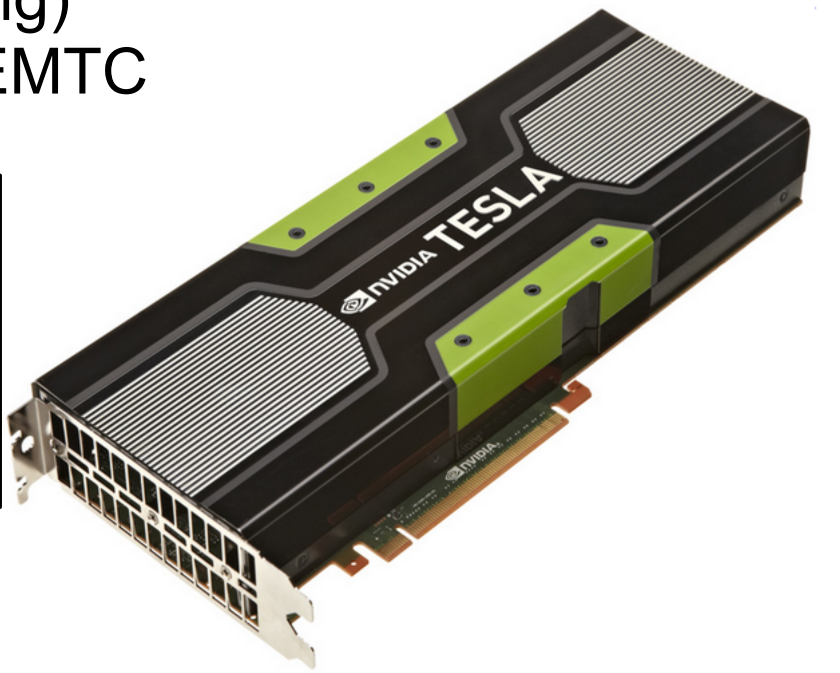
Highlighting SMX and Warps



Concept Overview

- Several works combine GPUs
- Split single GPU into many pieces
  - tiny cluster for compute
- 2 ideas
  - Framework managing the GPU = GeMTC (GPU enabled Many Task Computing)
  - Virtualization = Palacios + GEMTC





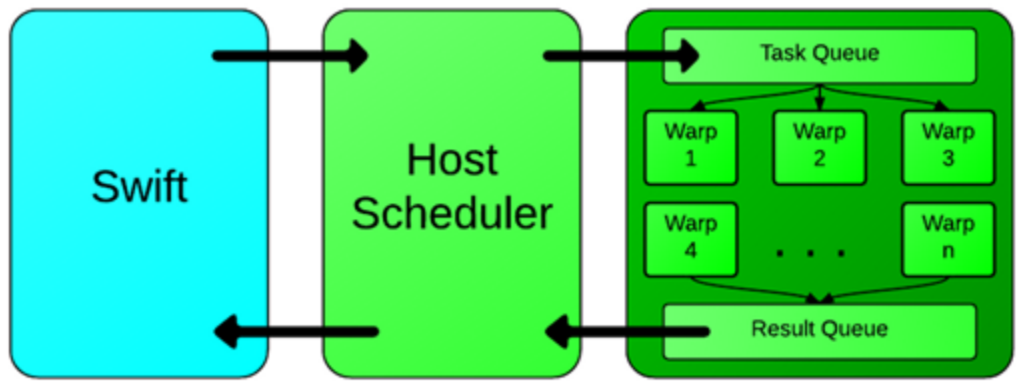
Lecture Outline

Lecture: (11:25pm-12:40pm)

- GeMTC (:45)
  - Motivation
    - Distributed Systems, HPC, MTC GPGPU.
  - **GeMTC**
    - Intro
    - Swift/T + Apps
    - Architecture, Design
    - Features: Memory, API
  - Future Work
    - MTACS
    - Xeon Phi
- Swift/T (:30)
  - Slides as paper was presented. (CCGrid'13)

Proposed Work

"GEMTC: **G**PU **E**nabled **M**any-**T**ask **C**omputing"



Motivation: No support for Many-Task Computing (MTC) on Accelerators!

Goals:

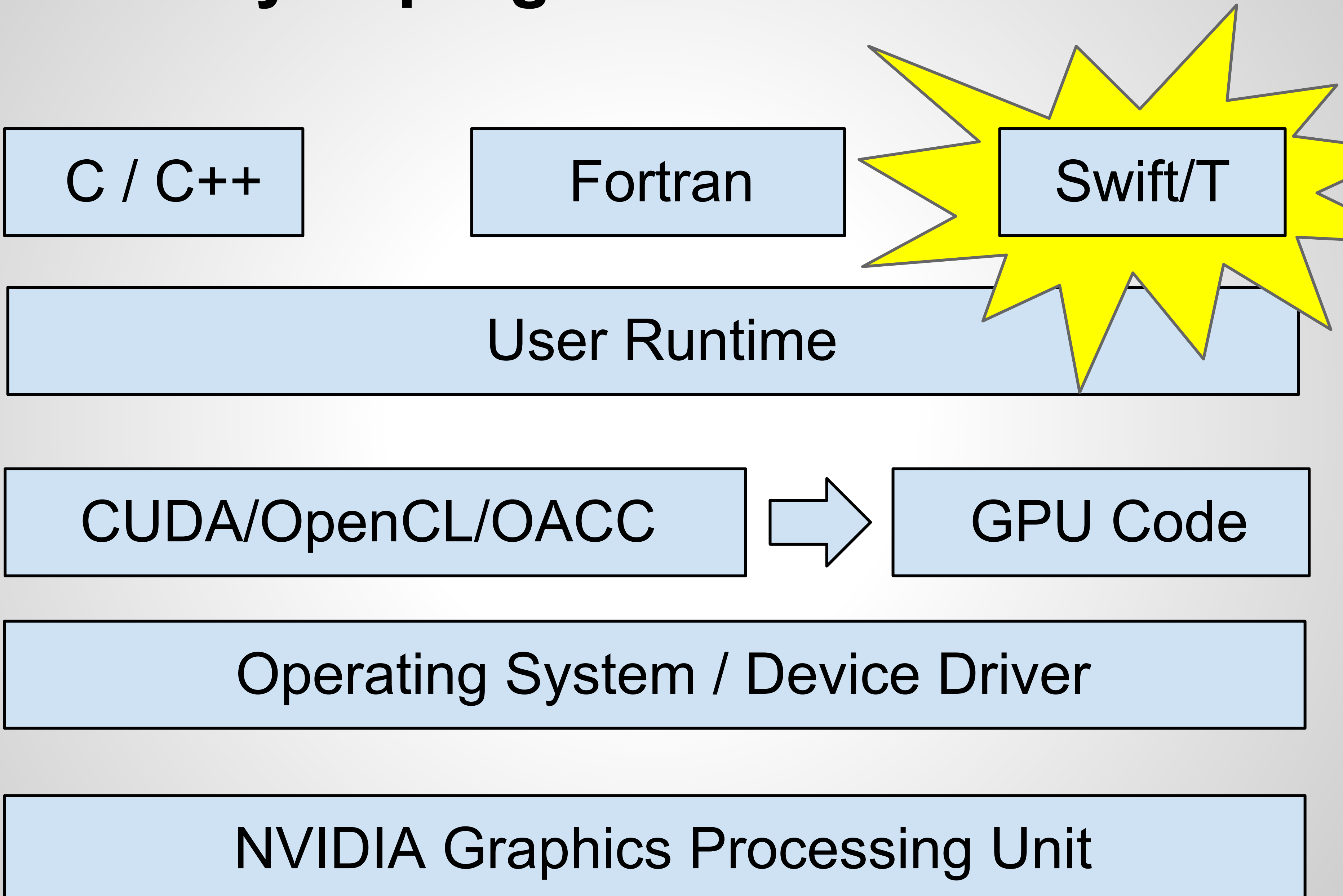
1) MTC support
2) programmability
3) efficiency
4) MIMD on SIMD
5) Increase concurrency 16 to 192 (12x)

Approach:

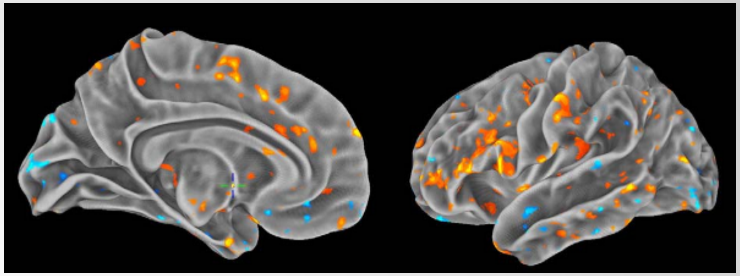
Design, implement middleware:

1) manages GPU
2) spread host/device
3) Workflow system support (Swift/T)

How do you program GPUs?



Swift/T and Applications



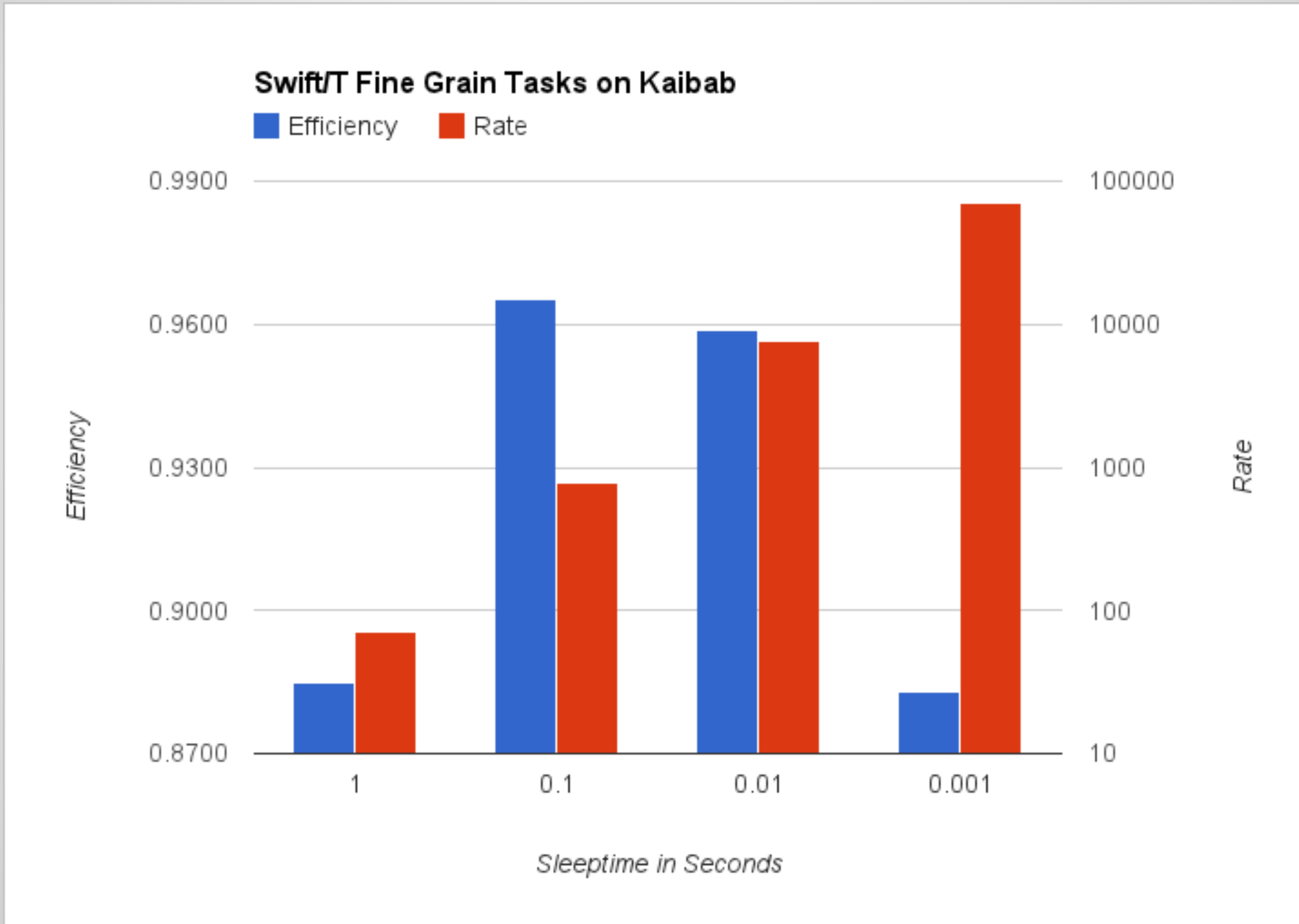
- Swift/T
  - Active research project (CI UChicago & ANL)
  - Parallel Programming Framework
  - Throughput ~25k tasks/sec per process
  - Shown to scale to 128k cores
- Application Domains Supported
  - Astronomy, Biochemistry, Bioinformatics, Economics, Climate Science, Medical Imaging

Swift lets you write parallel scripts that run many copies of ordinary programs concurrently, using statements like this:

```
foreach protein in proteinList {
  runBLAST(protein);
}
```

Images from Swift Case Studies - http://www.ci.uchicago.edu/swift/case_studies/

Swift/T Fine Grain, 80 W= 20 Nodes*4PPN

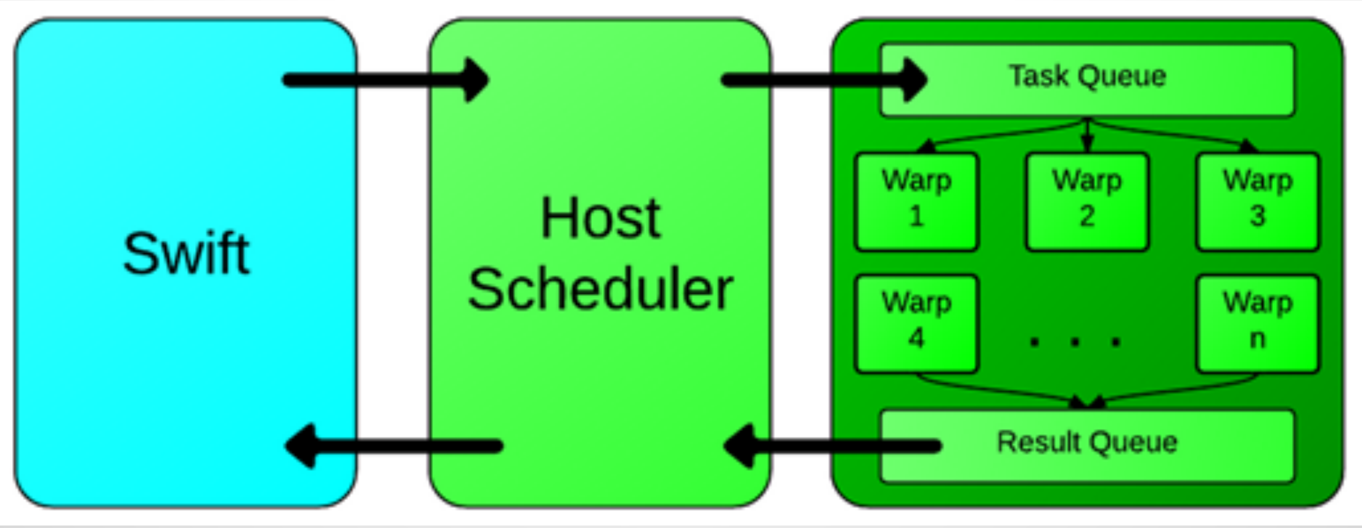


GEMTC Applications

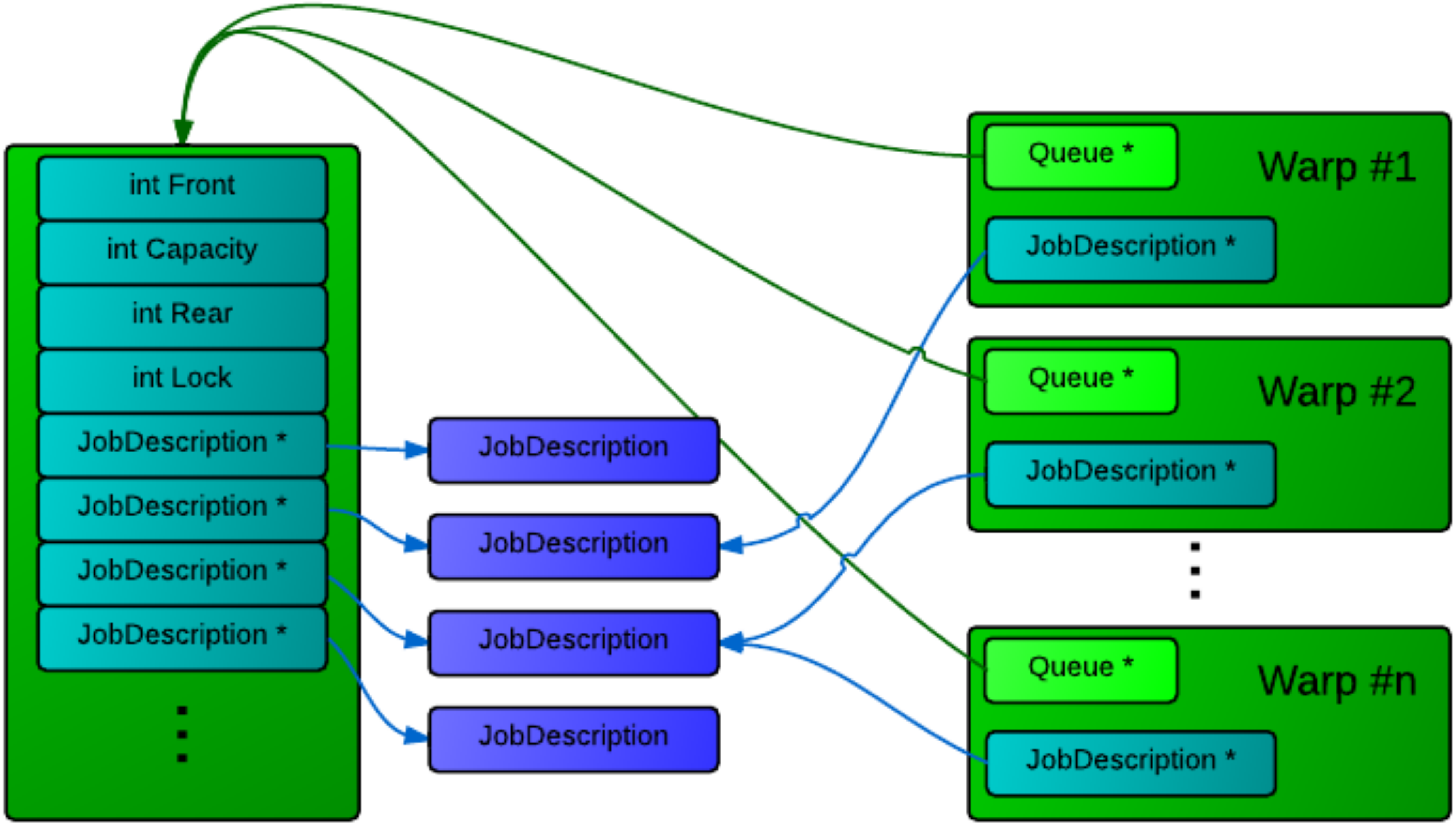
- Performance Benchmarks
  - sleep
  - sleep-data-move
  - matrix-multiply
  - vector-add
- Proxy Applications
  - MDProxy, Molecular Dynamics
- Scientific Applications (under development)
  - OOPS, Protein Folding
  - SciColSim, Collaboration graph analysis

GEMTC Task Flow

- Task submitted
  - description, (taskID, taskType, parameters)
- SuperKernel runs as daemon on device
  - incoming work queue
  - outgoing results queue
  - warp picks up task, executes task, return result
- Host checks results, returns to Swift/T



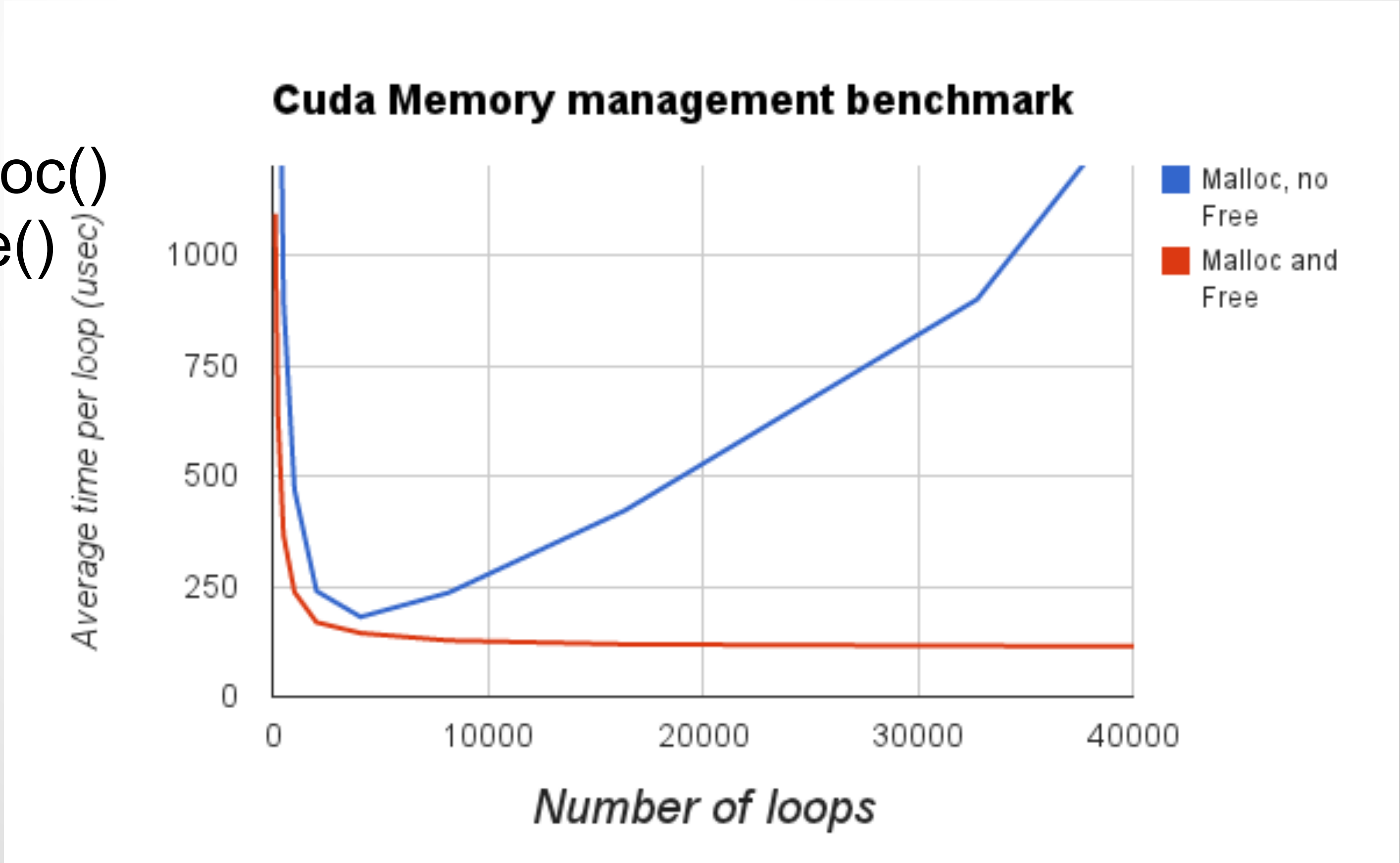
Warps and Incoming Queue



Mirror image for outgoing results queue

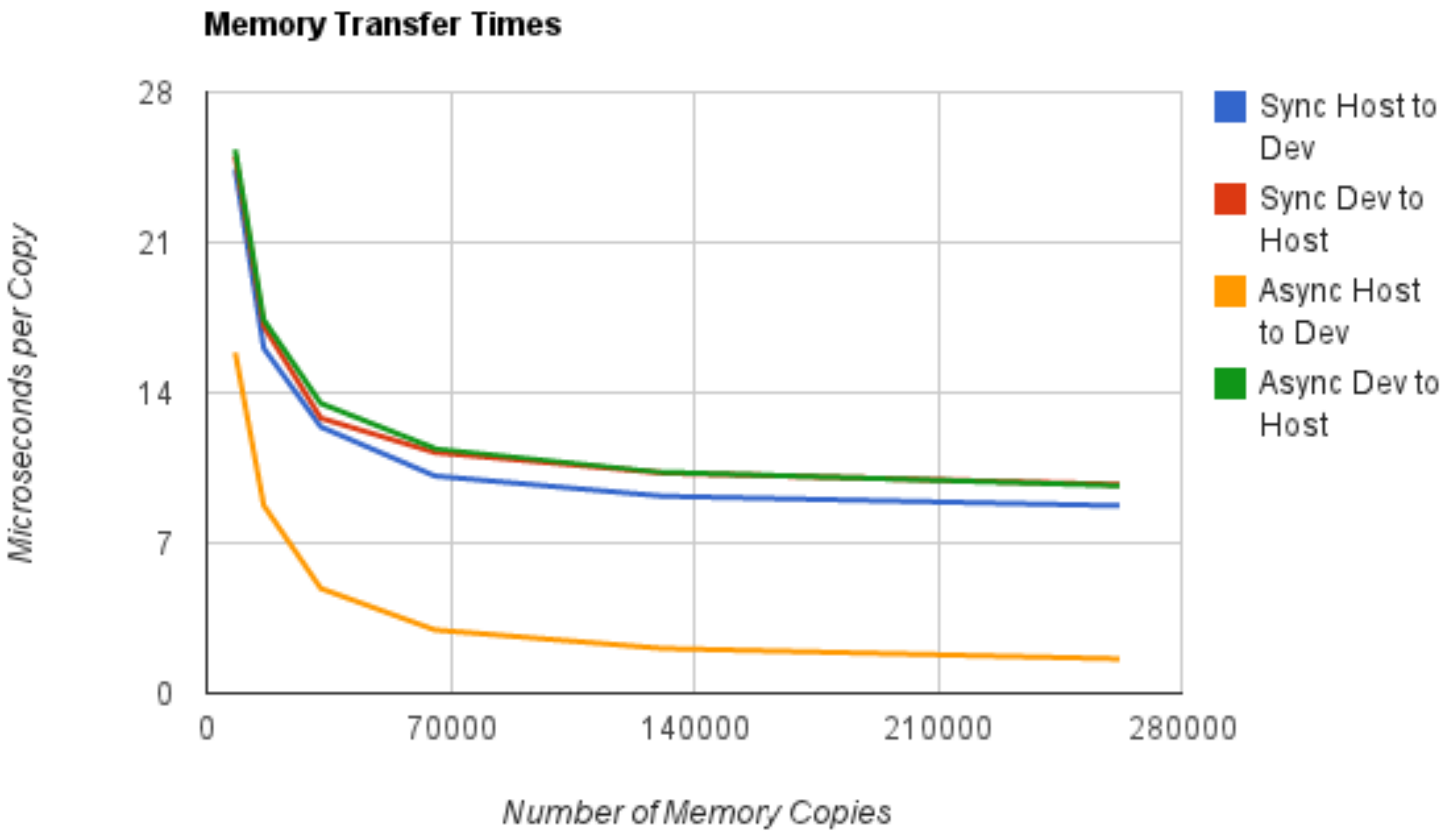
Sub Allocator - Motivation

- Malloc without free, terrible performance
- Malloc + Free ~= 110 usec
- Grab all GPU memory at the start
  - Manage memory on our own
- Memory operations
  - gemtcMalloc()
  - gemtcFree()



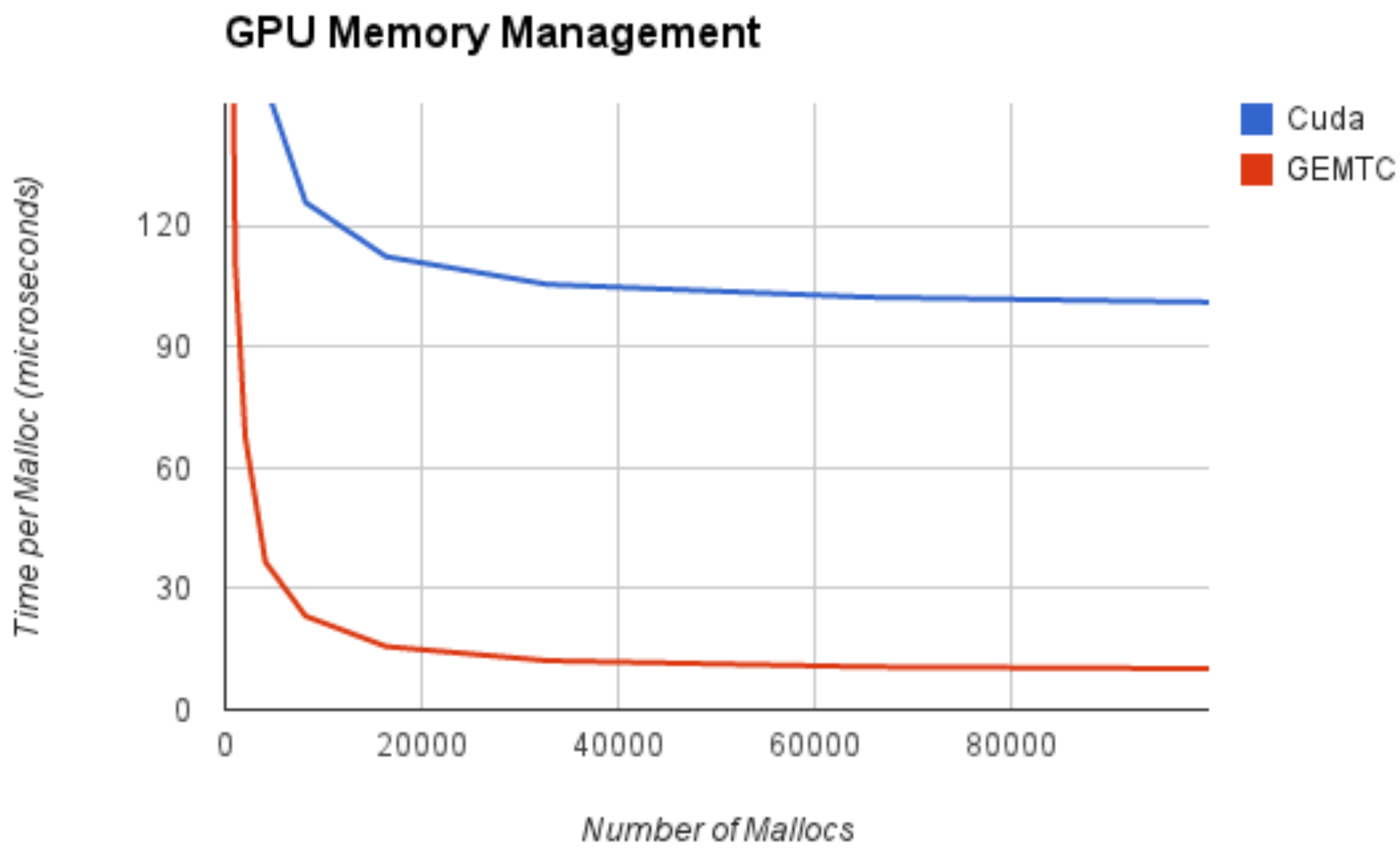
Sub Allocator - Theory

- cudaMalloc() called each task
- Communication times much lower!!



Sub Allocator - Results

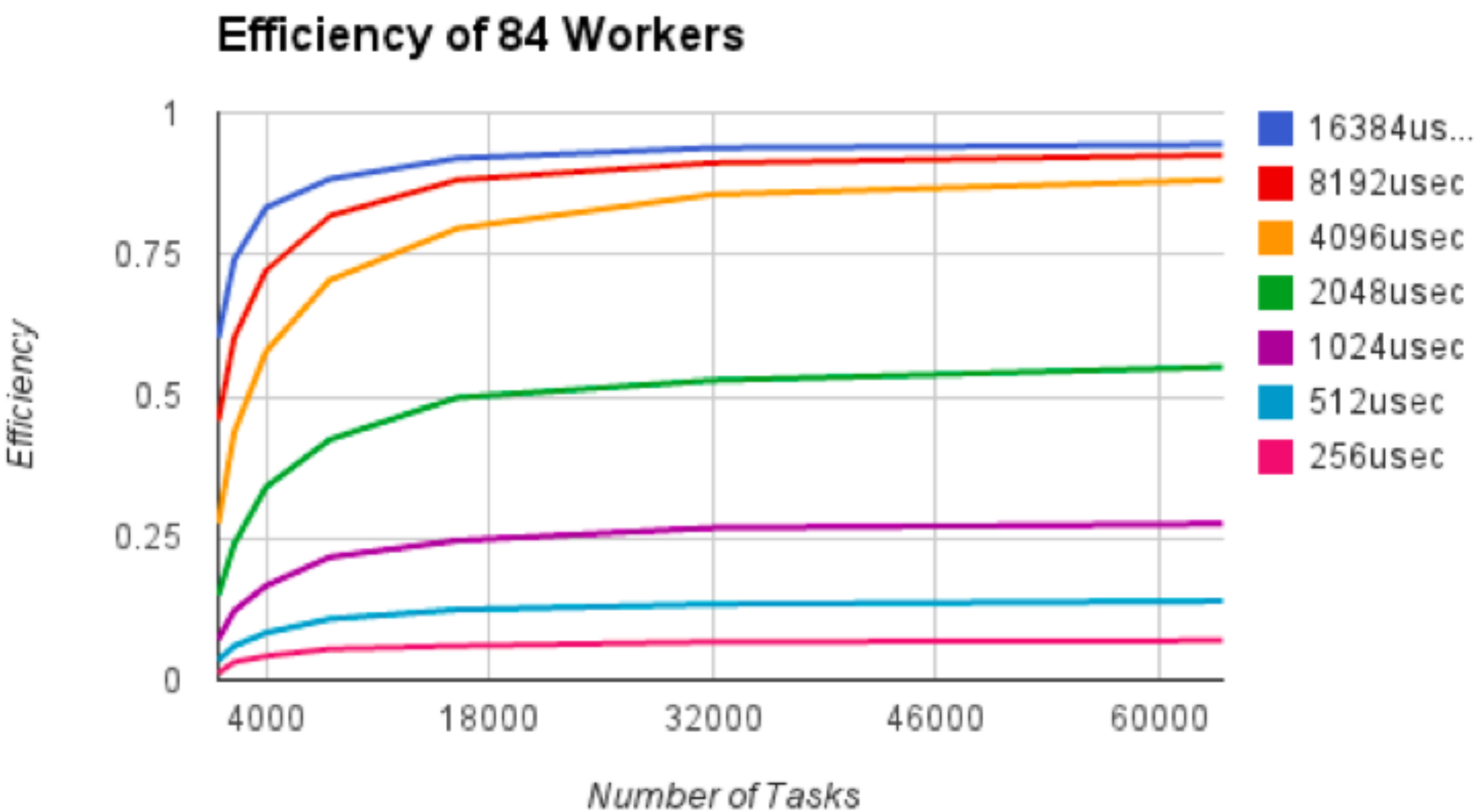
- cudaMalloc() ~= 110 usec
- gemtcMalloc() ~= 14 usec



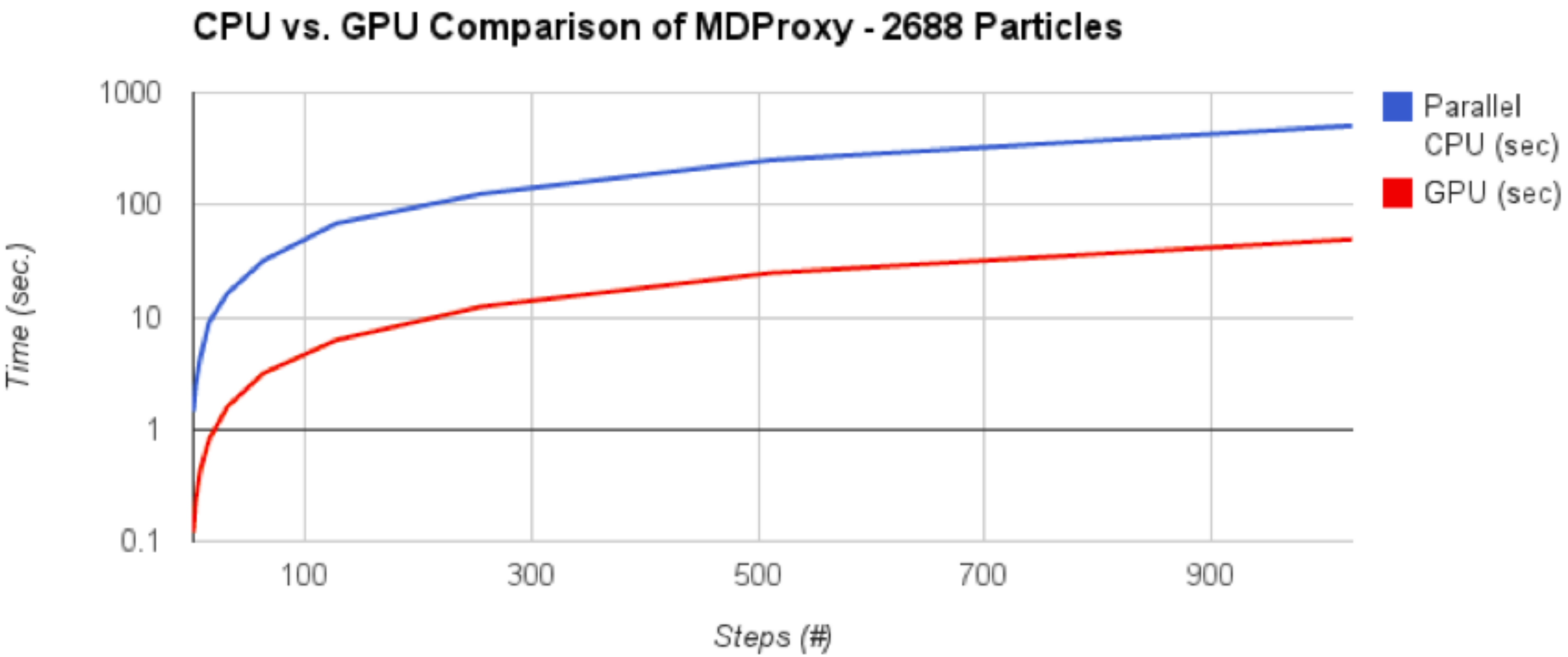
GeMTC API

- C API to interact with workflow systems (Swift/T)
- 8 functions including:
  - **Initialization/Deconstruction**
    - gemtcSetup() //starts our SuperKernel
    - gemtcCleanup() //kills SuperKernel
  - **Enqueue/Dequeue Tasks**
    - gemtcPush() // gemtcPush(1, 1000000)
    - gemtcPoll()
  - **Memory Transfer**
    - gemtcMemcpyHostToDevice()
    - gemtcMemcpyDeviceToHost()
  - **Memory Management**
    - gemtcGPUMalloc()
    - gemtcGPUFree()

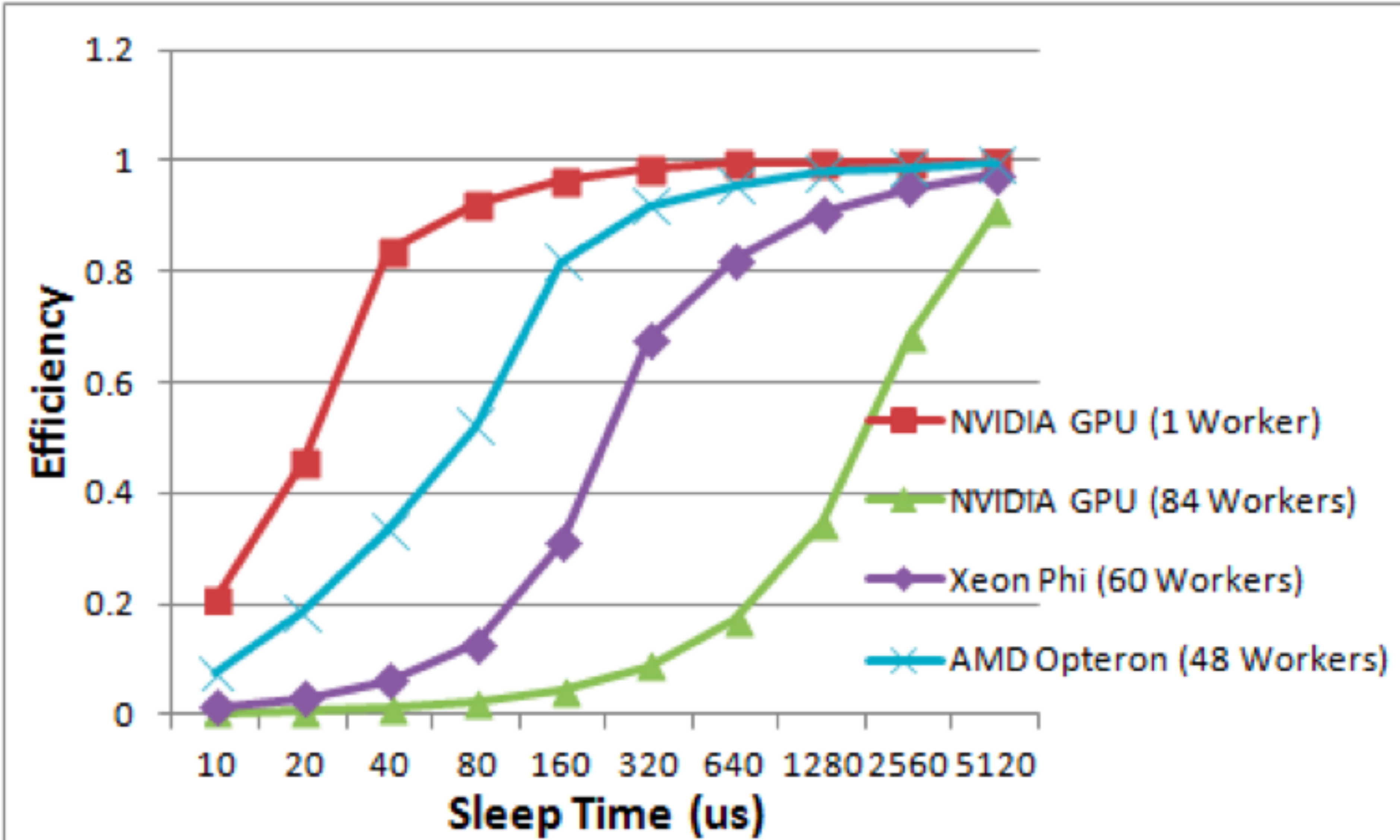
GeMTC Throughput



MDProxy CPU vs. GPU



Hardware Comparison



Conclusions

- *Integrated* GeMTC + Swift/T
  - programmability
  - scalability (multi-node support)
  - application domain
- *Evaluated* Performance Benchmarks
- Framework with ~200 independent workers per node
- MIMD on SIMD
  - MIMD collection of SIMD workers
  - unclear max efficiency for scientific applications
  - highly efficient for synthetic benchmarks

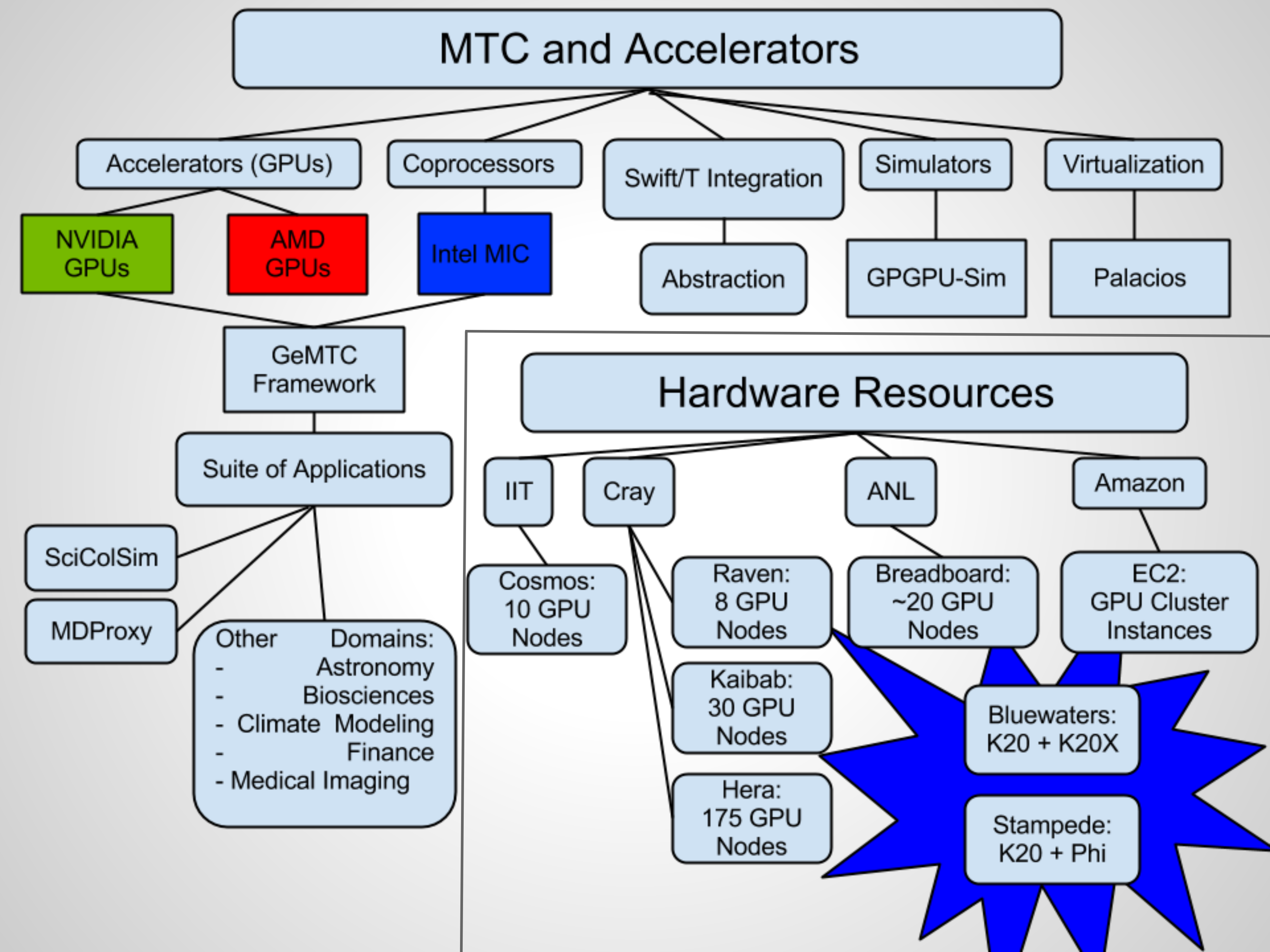
Lecture Outline

Lecture: (11:25pm-12:40pm)

- GeMTC (:45)
  - Motivation
    - Distributed Systems, HPC, MTC GPGPU.
  - GeMTC
    - Architecture, Design
    - Apps
  - **Future Work**
    - MTACS
    - Xeon Phi
    - GeMTC organization chart
- Swift/T (:30)
  - Slides as paper was presented. (CCGrid’13)

Future Work

- *Develop* Swift frameworks to evaluate additional applications
  - MDProxy
  - Protein Structure Prediction
  - Modeling of scientific knowledge acquisition in collaborative networks
- *Abstract* the Swift + GeMTC integration process for fast application deployment
- *Implement* an MTC solution for Intel Xeon-Phi and AMD GPUs.



MTC and Accelerators
Accelerators (GPUs)
Coprocessors
Swift/T Integration
Simulators
Virtualization
NVIDIA GPUs
AMD GPUs
Intel MIC
Abstraction
GPGPU-Sim
Palacios
GeMTC Framework
Suite of Applications
SciColSim
MDProxy
Other Domains:
- Astronomy
- Biosciences
- Climate Modeling
- Finance
- Medical Imaging
Hardware Resources
IIT
Cray
ANL
Amazon
Cosmos: 10 GPU Nodes
Raven: 8 GPU Nodes
Kaibab: 30 GPU Nodes
Hera: 175 GPU Nodes
Breadboard: ~20 GPU Nodes
EC2: GPU Cluster Instances
Bluewaters: K20 + K20X
Stampede: K20 + Phi

Thanks! Questions?

Scott Krieder

skrieder@iit.edu

Dr. Ioan Raicu

iraicu@cs.iit.edu

Open Source:

https://github.com/skrieder-datasys/gemtc

NVIDIA CUDA Teaching & Research

- Expect to see more GPU projects in CS5XX
- Monthly CUDA Workshops
  - Starting September 3rd 1:50 - 2:40 LS 111
- IIT Free GPUs

Jarvis

- Rocks Cluster Management
  - CentOS
- 10 GPU Nodes
  - 2 Kepler K20
  - 3 GTX 650
  - 5 GTX 480
- Node Types
  - Frontend
  - Compute

Rocks Cluster Management

- Install to
  - frontend
- Compute nodes
  - PXE boot
- NFS
  - across nodes
- SGE
  - Batch scheduler
  - Interactive
  - Scheduled

Any Questions so far?

- Recap
  - HPC
  - GPGPU, CUDA
  - GeMTC
  - Jarvis
- Switch to Swift/T slides.

Hands on Outline

Hands On: (12:45pm-1:45pm)

- CUDA
  - Installing CUDA
  - SDK Examples
    - DeviceQuery
  - Vector-Add
- GeMTC
  - Vector-Add
- Swift/T
  - Vector-Add

How can I run CUDA?

- Getting started tips:
  - https://sites.google.com/site/iitcuda/cuda-quickstart
- Do you have an NVIDIA GPU on your machine?
  - Is it CUDA compliant?
    - https://developer.nvidia.com/cuda-gpus
- Jarvis
  - pending accounts/projects

Installing CUDA

- Download CUDA here:
  - https://developer.nvidia.com/cuda-downloads

CUDA Registered Developer

- Forum access
- Extra downloads
- Developer tools
- Register here:
  - https://developer.nvidia.com/programs/cuda/register

Hands on Outline

Hands On: (12:45pm-1:45pm)

- CUDA
  - Installing CUDA
  - SDK Examples
    - DeviceQuery
  - Vector-Add
- **GeMTC**
  - Cloning the git repo.
  - Code overview.
  - Vector-Add
- Swift/T
  - Vector-Add