Supporting Preemptive Task Executions and Memory Copies in GPGPUs

C. Basaran K. D. Kang

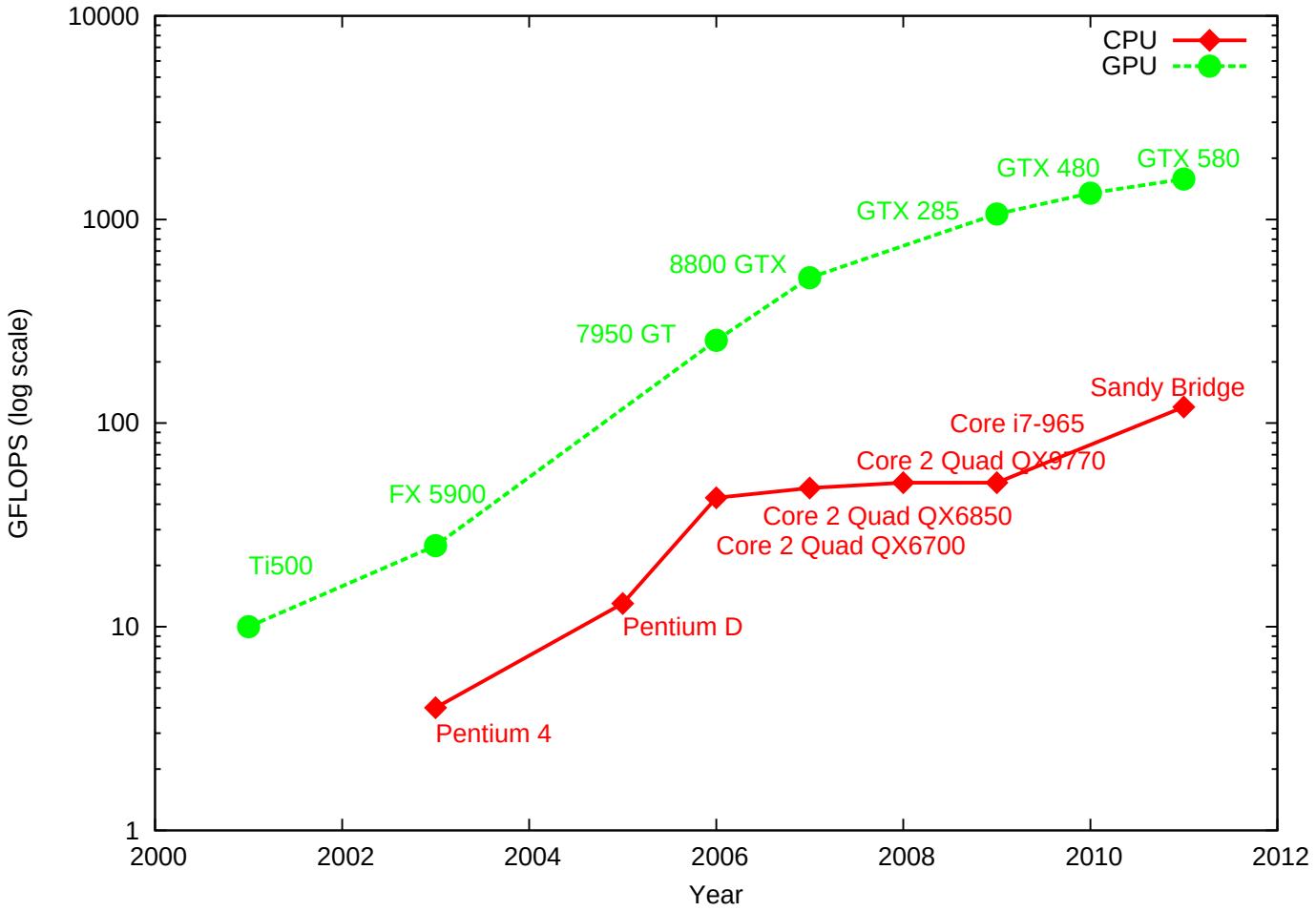
Department of Computer Science
State University of New York at Binghamton

ECRTS 2012

Why GPU for real-time computing?

- Nvidia Fermi hosts 512 Single Instruction Multiple Threads (SIMT) cores
  - Each thread can keep track of its state
  - Zero context switching overhead
- Well written code can achieve up-to 10x speed-up over the CPU
- Cost/Performance yield is high compared to the CPU
- GPUs can improve real-time application performance significantly

CPU vs. GPU: Peak Performance

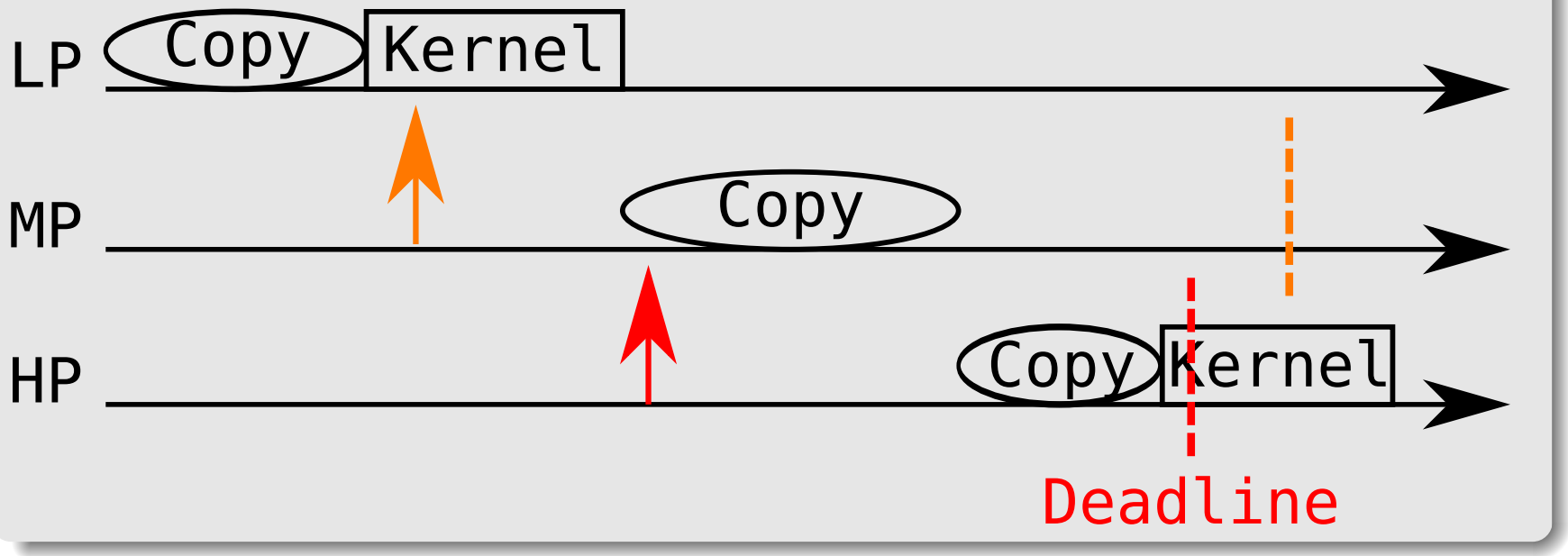


Priority Inversion

- GPU is command driven
  - Copy input data to device
  - Execute device code on copied data, produce output
  - Copy device output to the host memory
- GPU commands are non-preemptive
  - Subject to priority inversions
  - Relatively hard to do schedulability analysis

Example Priority Inversion

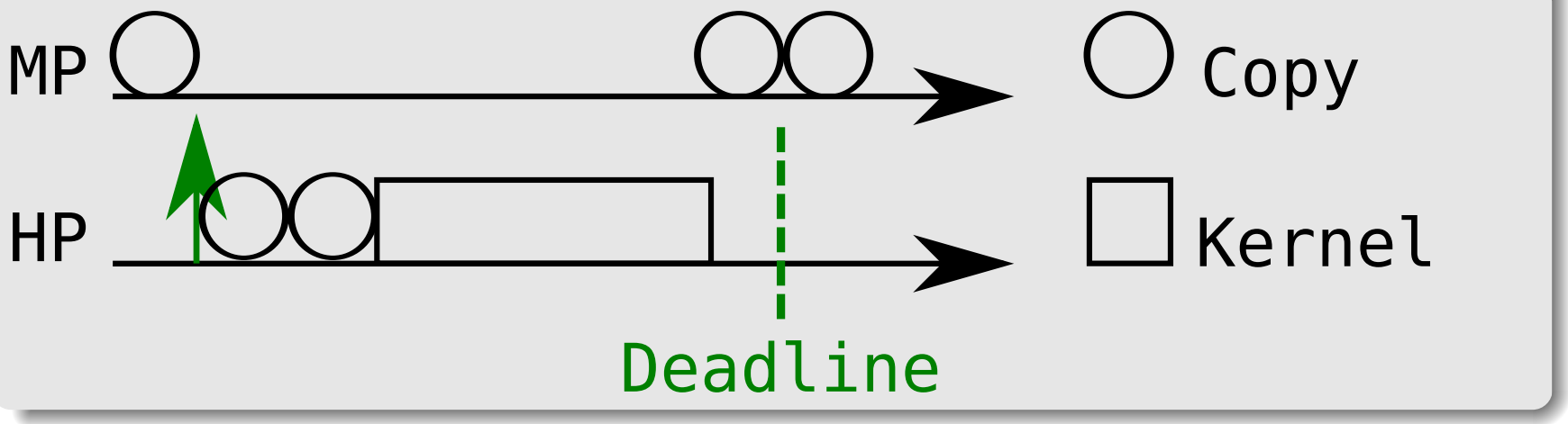
Example



Preemptive Memory Copies

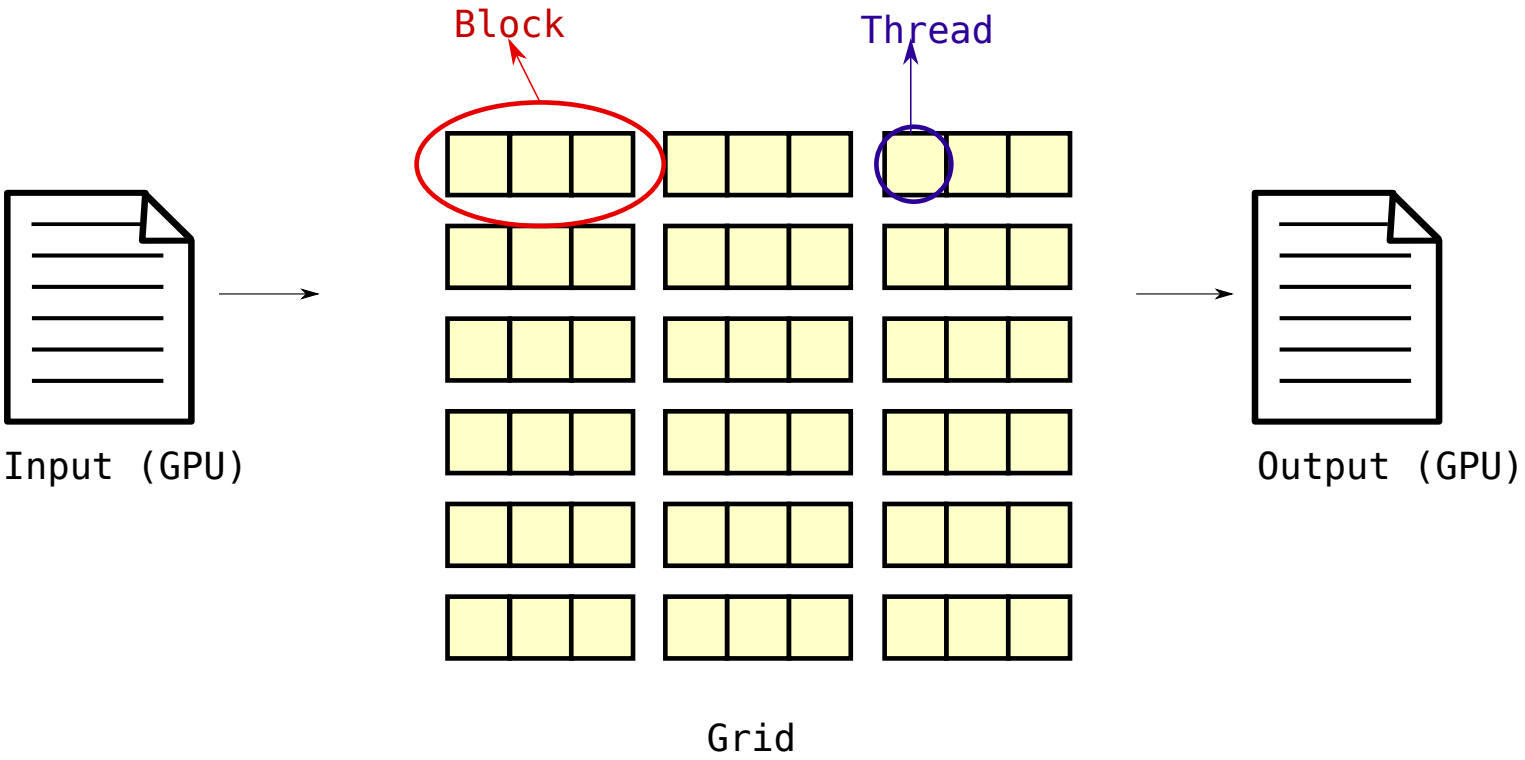
- Data transfers are non-preemptive in both directions
  - Transfer speed is predictable
- Instead of doing one big transfer, do multiple memcopies of small chunks
  - Preemption between memcopies of two chunks

Example



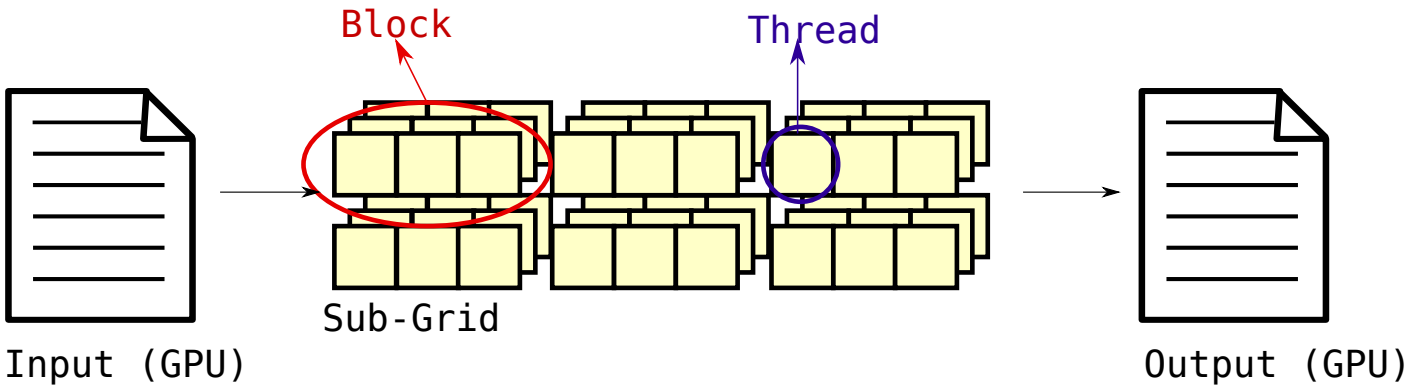
Kernel Execution

- Kernel executions are non-preemptive in GPUs
  - A kernel is executed by a grid of CUDA thread blocks
  - CPU issues the execute command and waits until completion



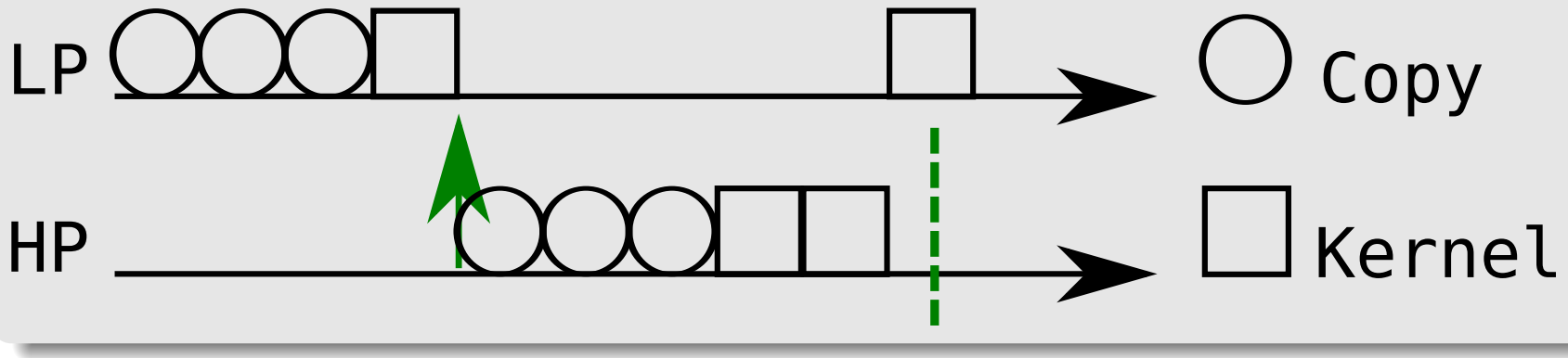
Preemptive Kernels

- Instead of executing a complete CUDA grid, execute **sub-grids**
  - Exploit lack of block level synchronization in Cuda
    - Blocks are assumed to execute independently
    - Concurrently executing blocks are unknown to each other
  - Preemption between **sub-kernel boundaries**



Kernel Preemption

Example

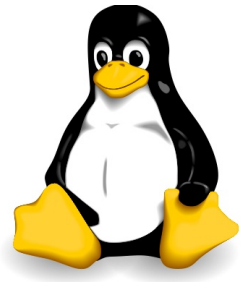


Overlapping Preemptive Memcopies and Subkernels

- Separate queues for memcopies and kernel executions per-task
  - Fixed priority scheme
  - CUDA can overlap memory copies and kernel executions of different tasks
    - As not only memcopies but also (sub)kernels are preemptive, PKM can overlap a subkernel execution and memcopy of different tasks regardless of their priorities
    - Overlapping a high priority memcopy and a **non-preemptive** low-priority kernel incurs priority inversion
  - Each queue entry stores a timestamp and task specific parameters

Performance Evaluation

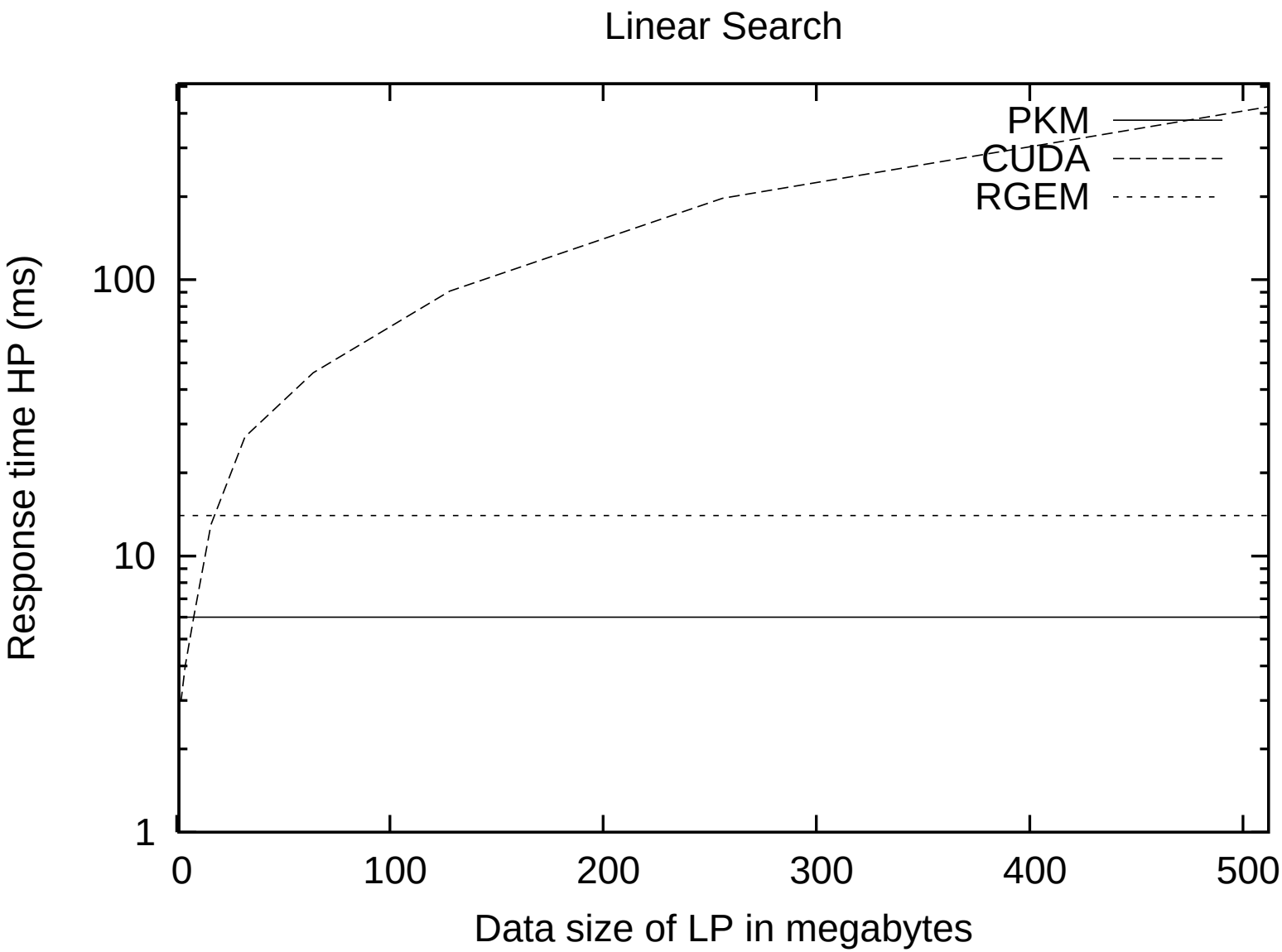
- Experimental Setup
  - GPU: NVIDIA GeForce GTX 460
  - CPU: AMD Athlon II X4 630
  - Memory: 4GB RAM, 500GB HD
  - OS: Linux 2.6.32.21 kernel
  - Micro-benchmarks:
    - Linear Search: Search the input for a given string
    - Matrix Multiplication: Multiply two matrices



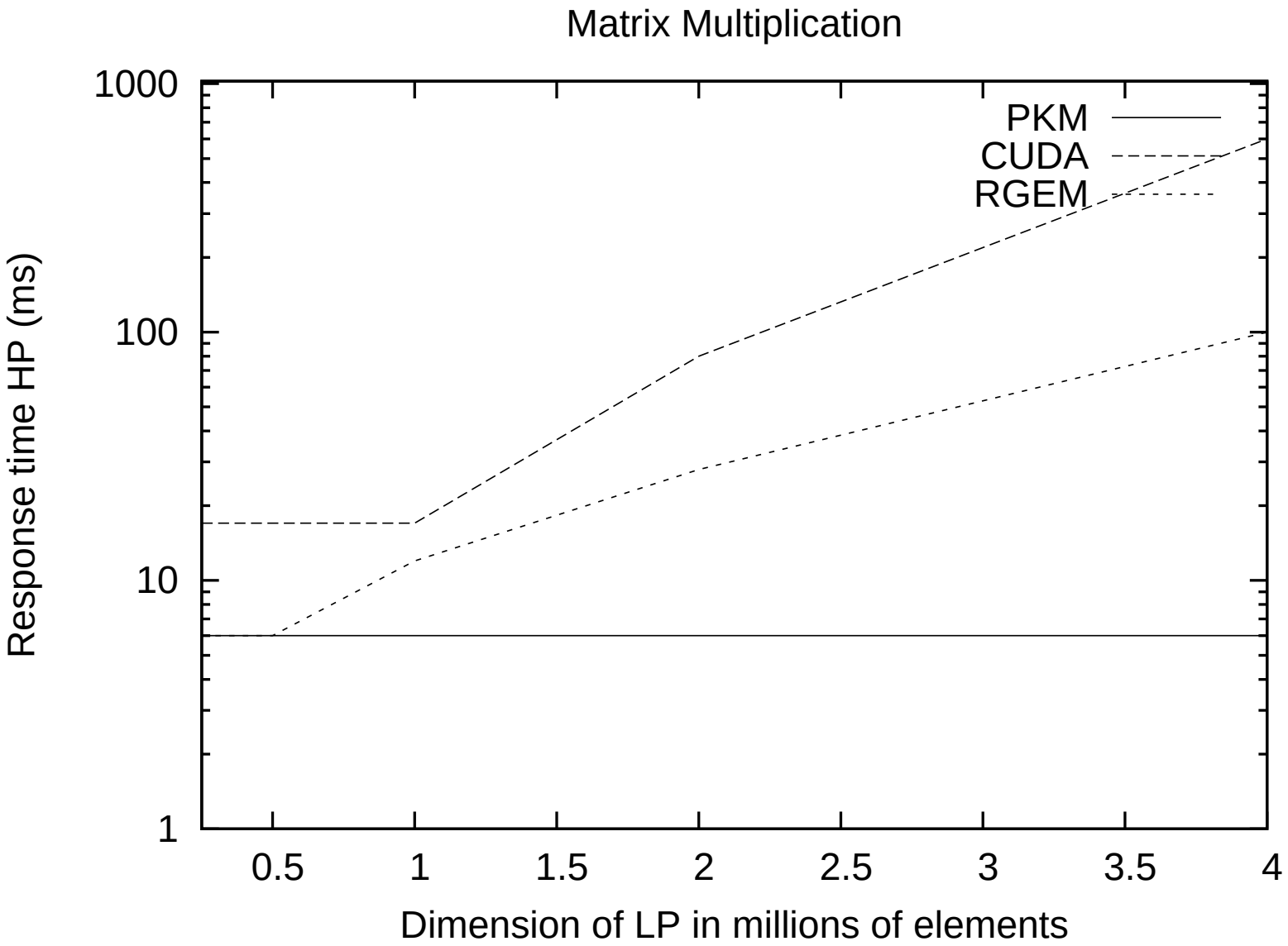
Baselines: CUDA and RGEM (RTSS '11)

- RGEM (Responsive GPGPU Execution Model) decreases the response time of high priority tasks via **preemptive memory copies**
- RGEM does **not** support **kernel preemption**
- Both PKM and RGEM use Direct Memory Access (DMA) to copy data between host and device
  - RGEM makes an extra copy between application and RGEM buffers in the OS kernel space
    - This extra copy of RGEM is eliminated in the experiments to favor RGEM

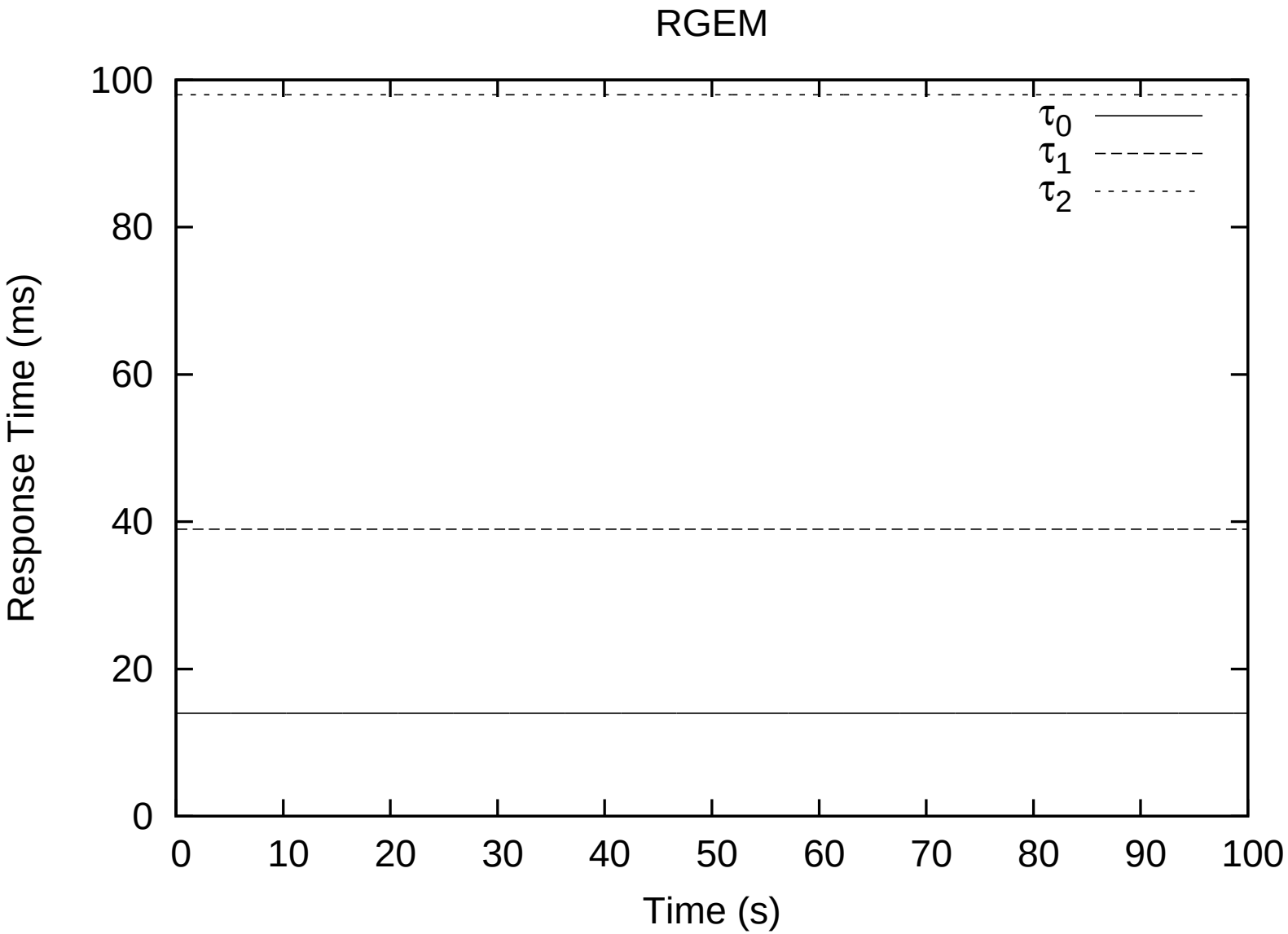
Experimental Results



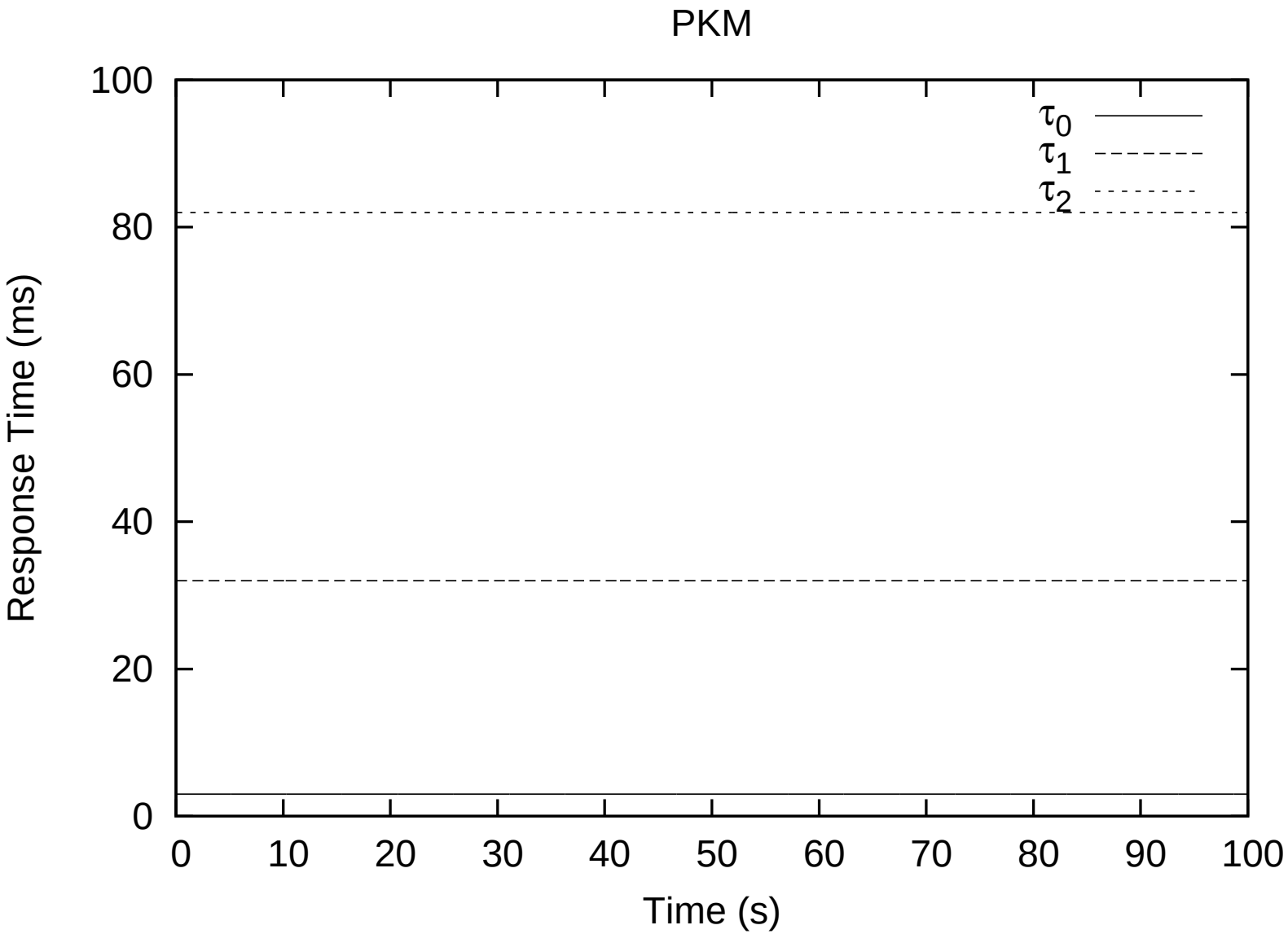
Experimental Results



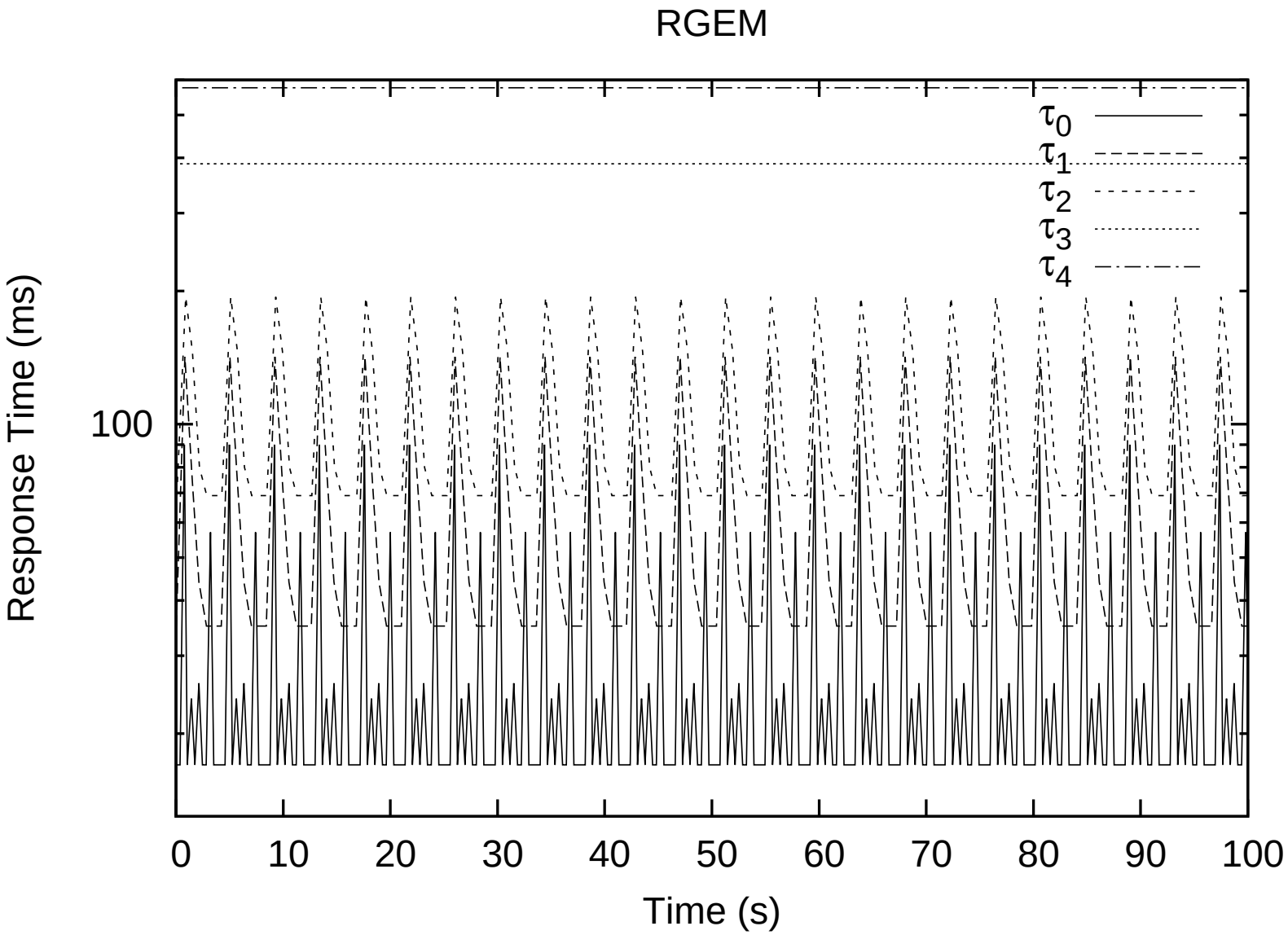
Experimental Results



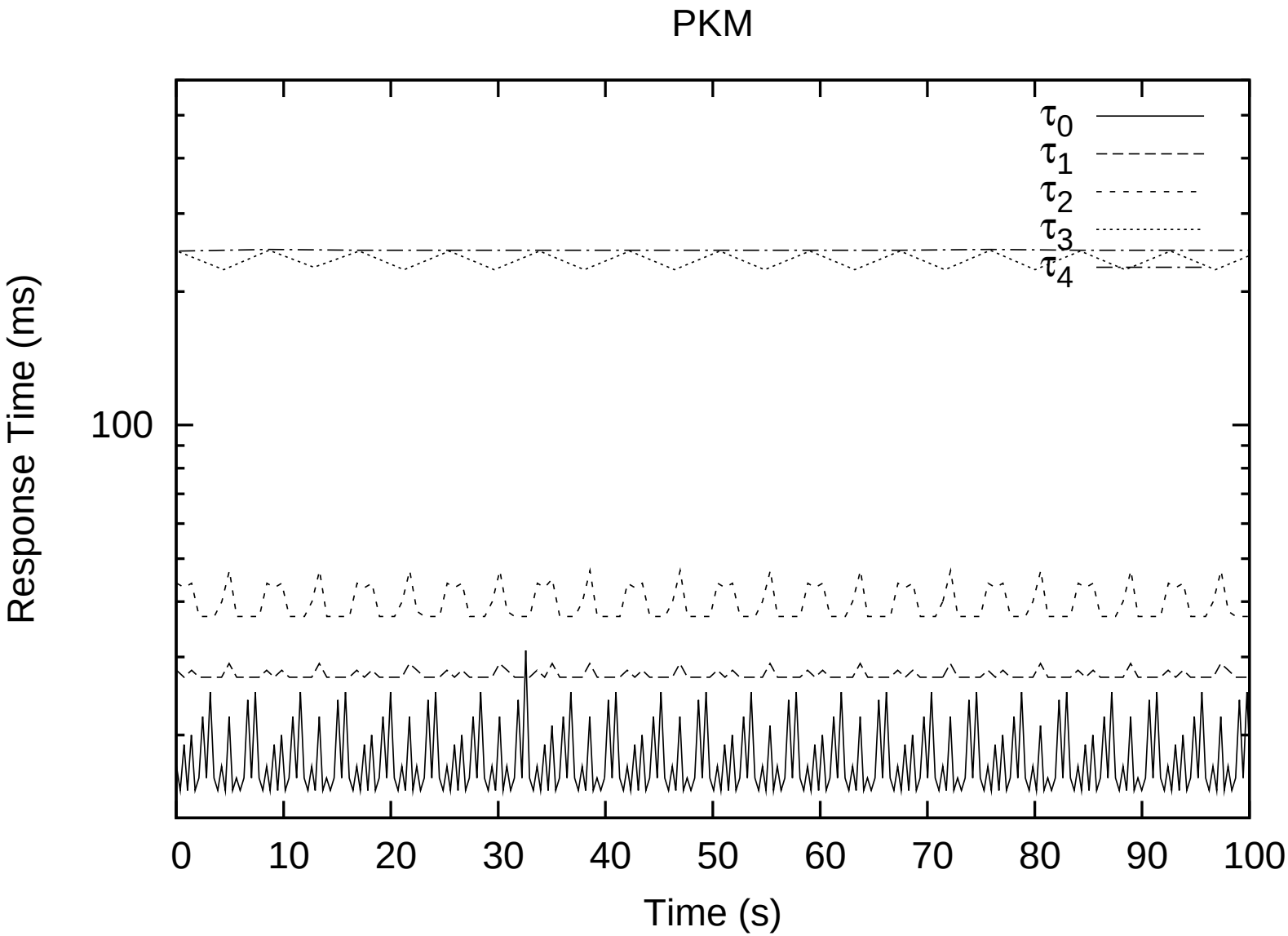
Experimental Results



Experimental Results



Experimental Results



Conclusions

- GPGPUs can provide significant computational power to real-time applications
- Preemptive copies and kernel executions to alleviate potential priority inversion
- Greatly simplifies methods to hide data transfer overhead due to preemptive memcopies and subkernel execution

Future Work

- Schedulability analysis
- Support for dynamic priority scheme
- QoS adaptation
- More advanced applications, e.g., image processing for CPS applications

Thank You

Questions?