EECS 583 – Class 21
Research Topic 3: Compilation for GPUs

University of Michigan

December 12, 2011 – Last Class!!

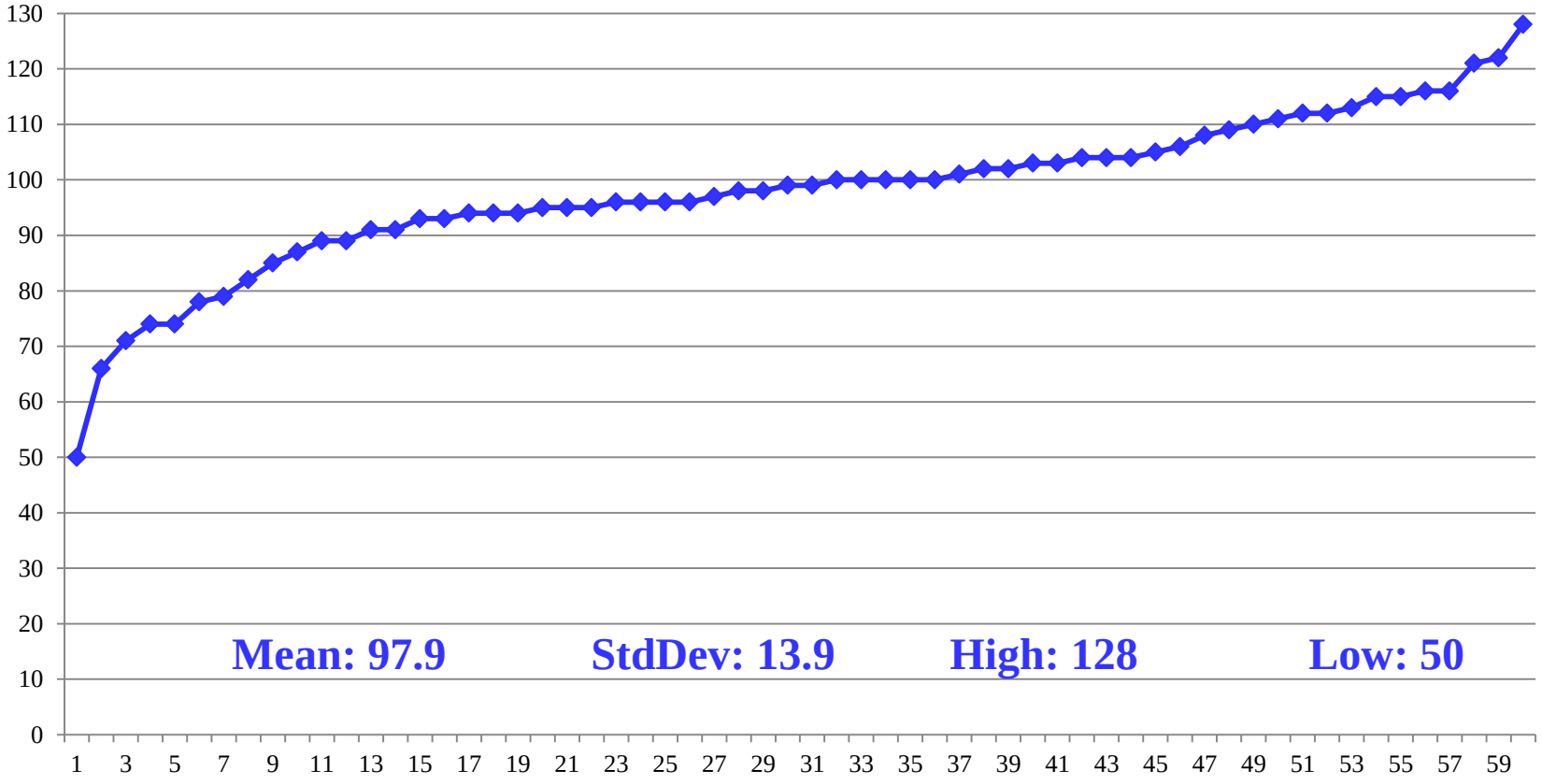
Announcements & Reading Material

- This class reading
  - "Program optimization space pruning for a multithreaded GPU," S. Ryoo, C. Rodrigues, S. Stone, S. Baghsorkhi, S. Ueng, J. Straton, and W. Hwu, *Proc. Intl. Sym. on Code Generation and Optimization*, Mar. 2008.
- Project demos
  - Dec 13-16, 19 (19th is full)
  - Send me email with date and a few timeslots if your group does not have a slot
  - Almost all have signed up already

Project Demos

- Demo format
  - Each group gets 30 mins
    - Strict deadlines because many back to back groups
    - Don't be late!
  - Plan for 20 mins of presentation (no more!), 10 mins questions
    - Some slides are helpful, try to have all group members say something
    - Talk about what you did (basic idea, previous work), how you did it (approach + implementation), and results
    - Demo or real code examples are good
- Report
  - 5 pg double spaced including figures – same content as presentation
  - Due either when you do you demo or Dec 19 at 6pm

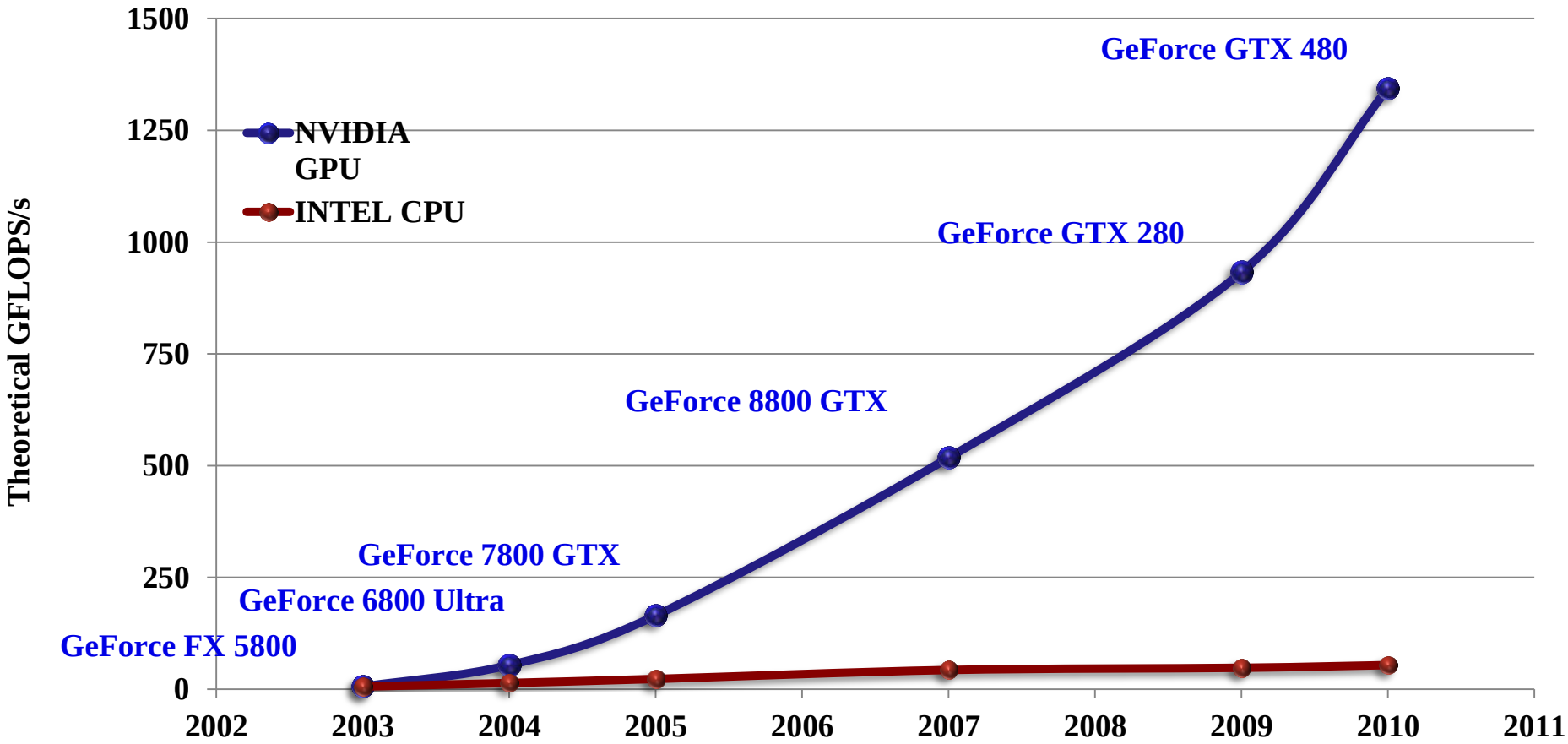
Midterm Results



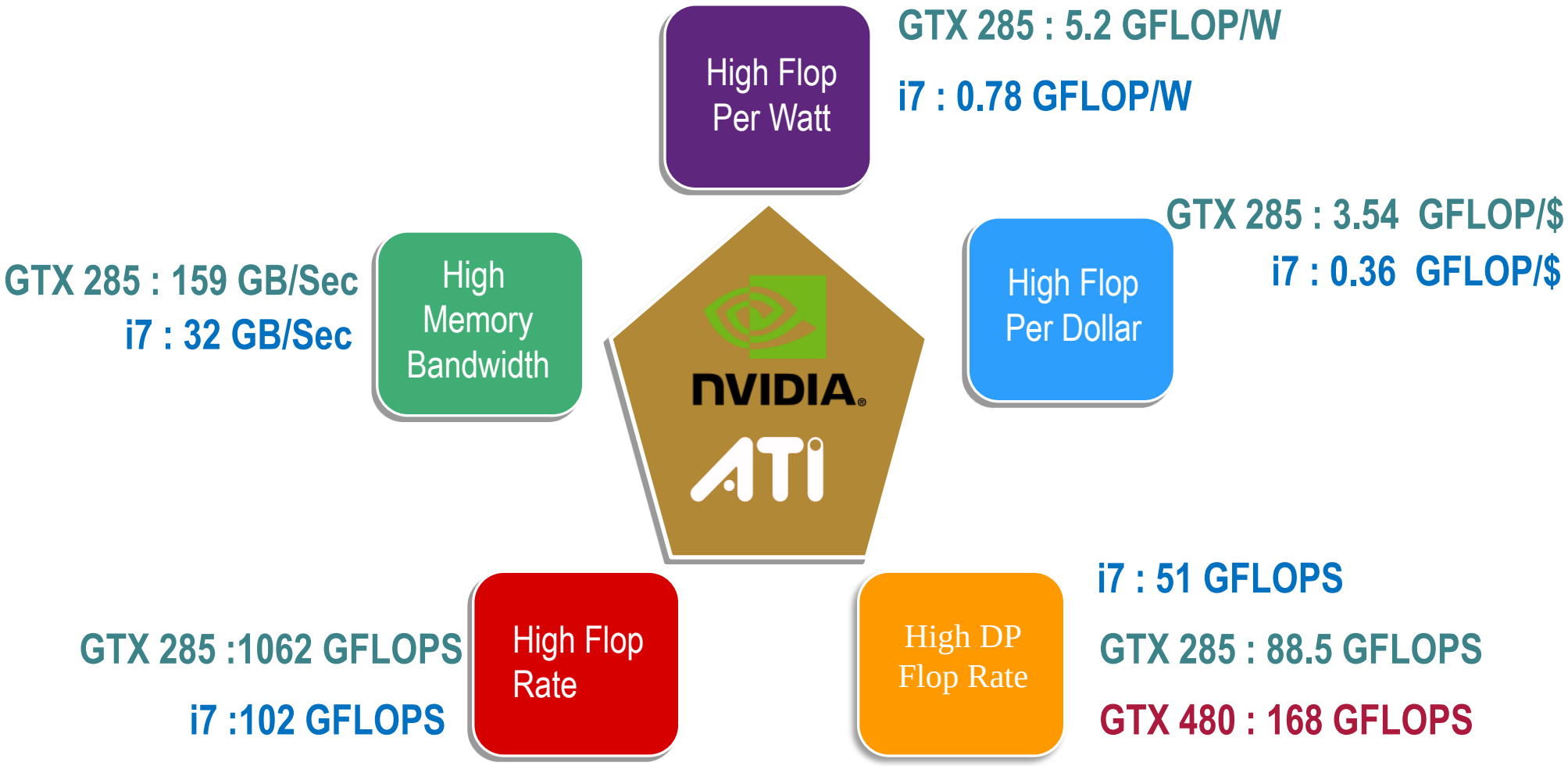
Answer key on the course webpage, pick up graded exams from Daya

If you did poorly, all is not lost. This is a grad class, the project is by far most important!!

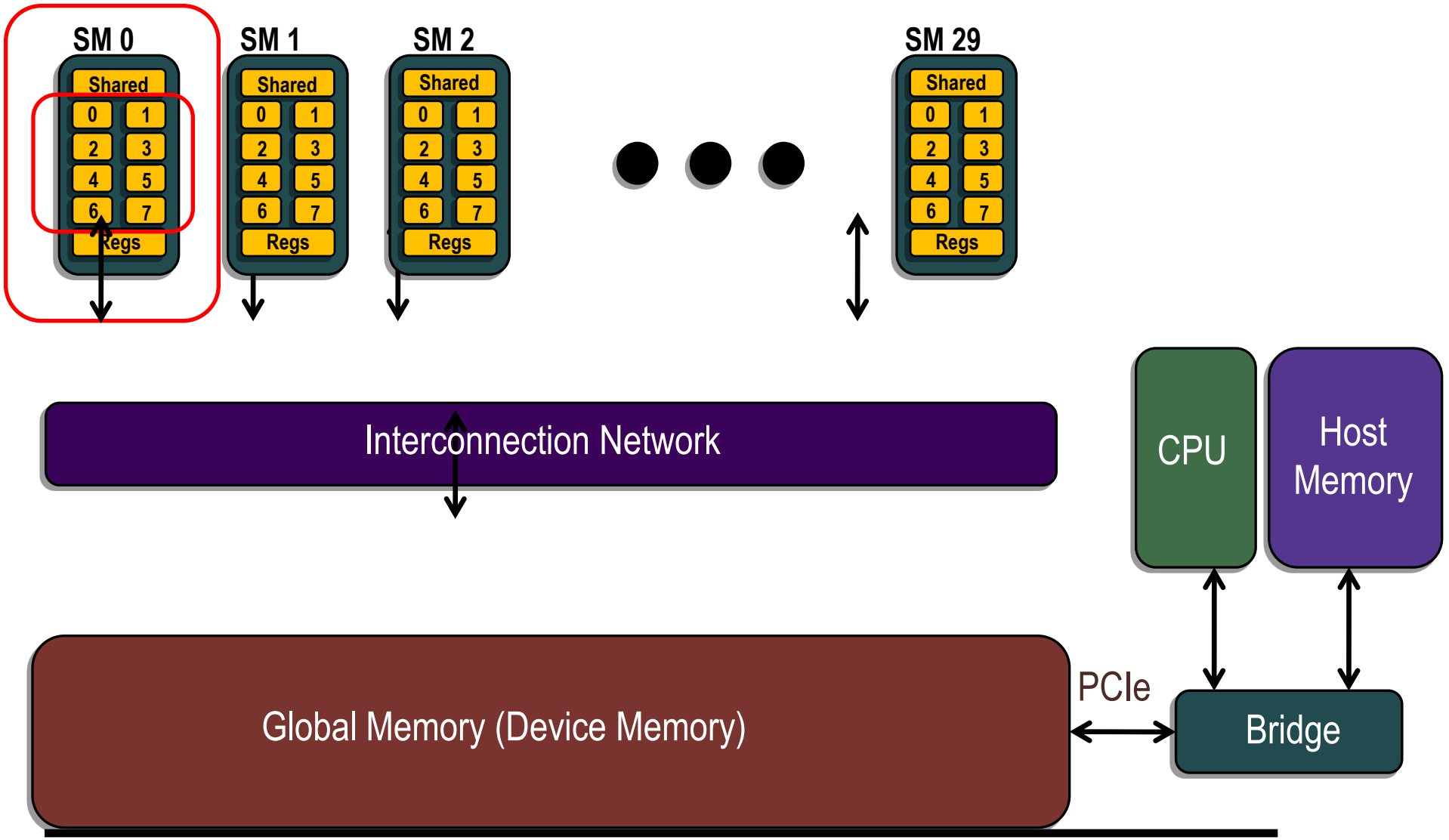
Why GPUs?


Theoretical GFLOPS/s
1500
1250
1000
750
500
250
0
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
NVIDIA GPU
INTEL CPU
GeForce GTX 480
GeForce GTX 280
GeForce 8800 GTX
GeForce 7800 GTX
GeForce 6800 Ultra
GeForce FX 5800

Efficiency of GPUs



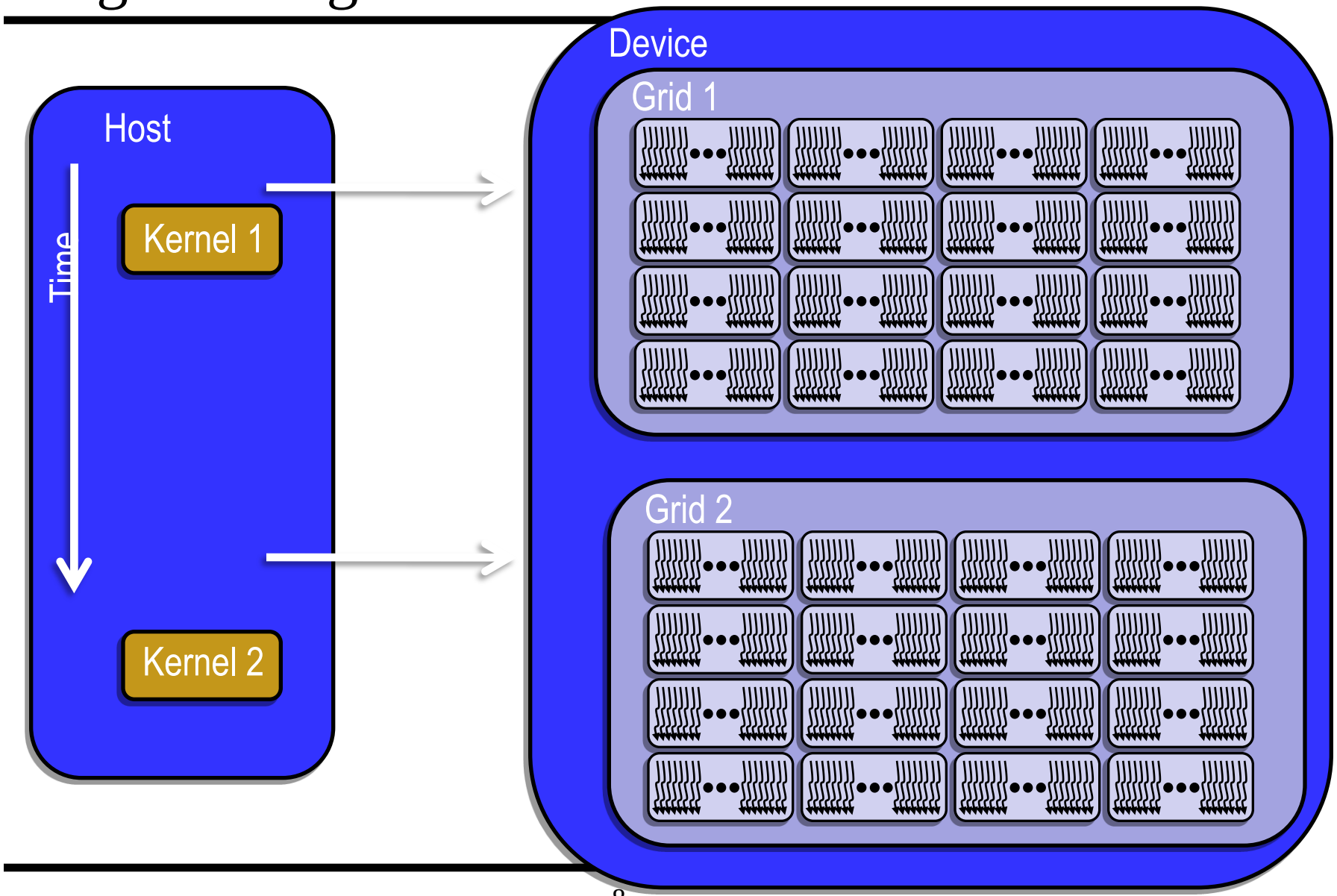
GPU Architecture



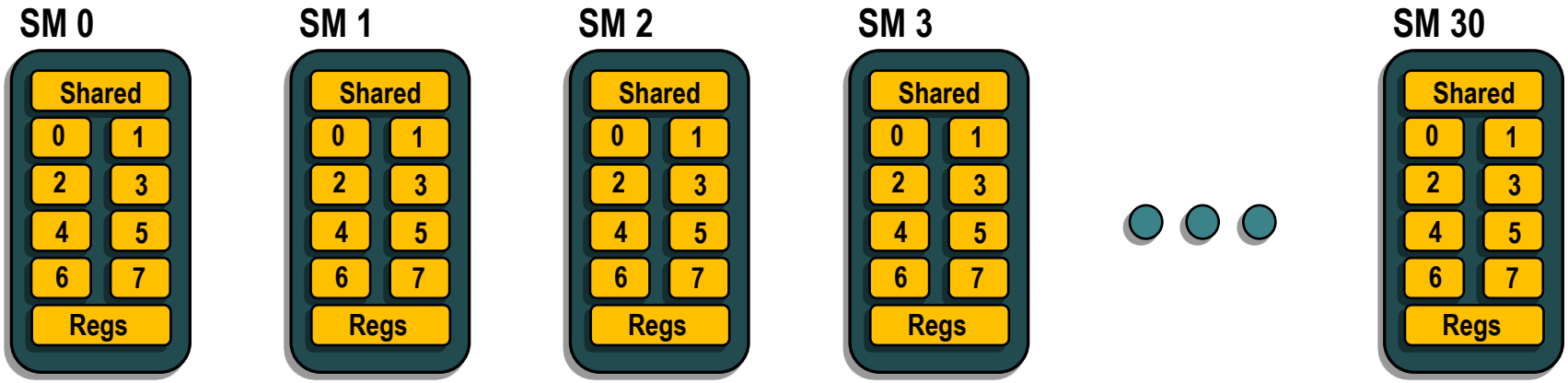
CUDA

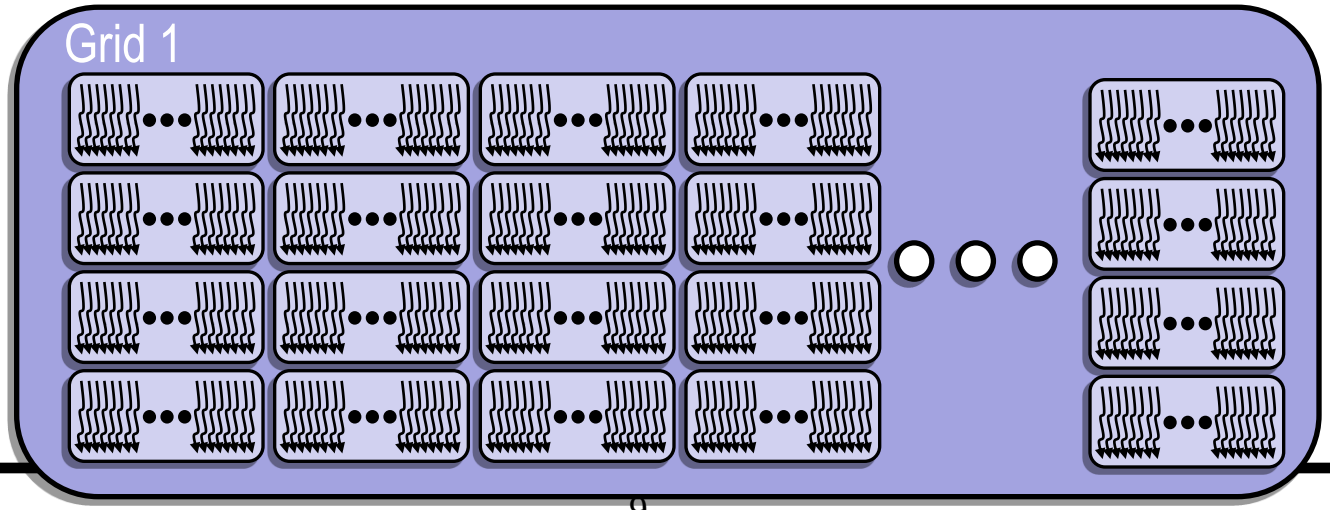
- "Compute Unified Device Architecture"
- General purpose programming model
  - User kicks off batches of threads on the GPU
- Advantages of CUDA
  - Interface designed for compute - graphics free API
  - Orchestration of on chip cores
  - Explicit GPU memory management
  - Full support for Integer and bitwise operations

Programming Model

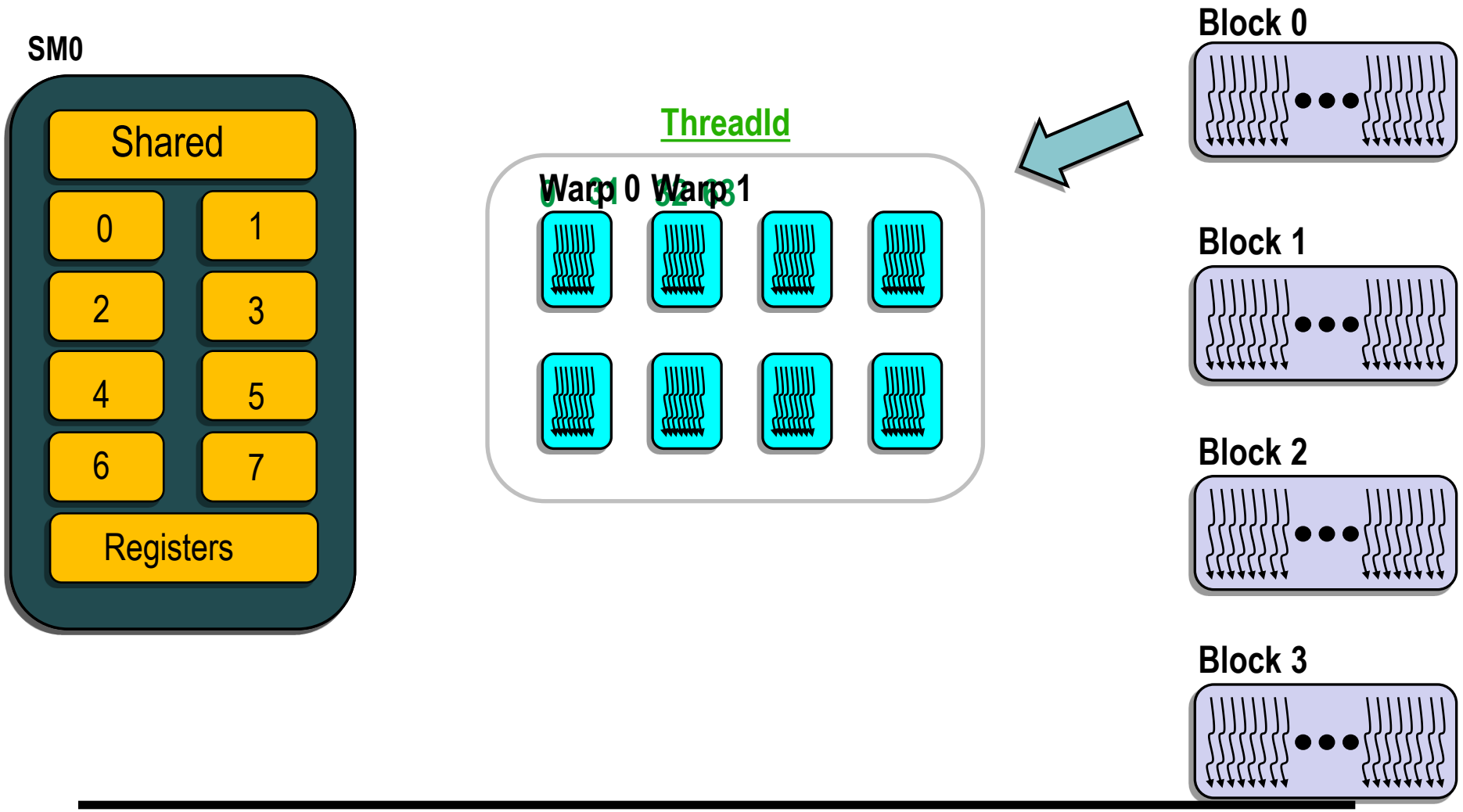


GPU Scheduling

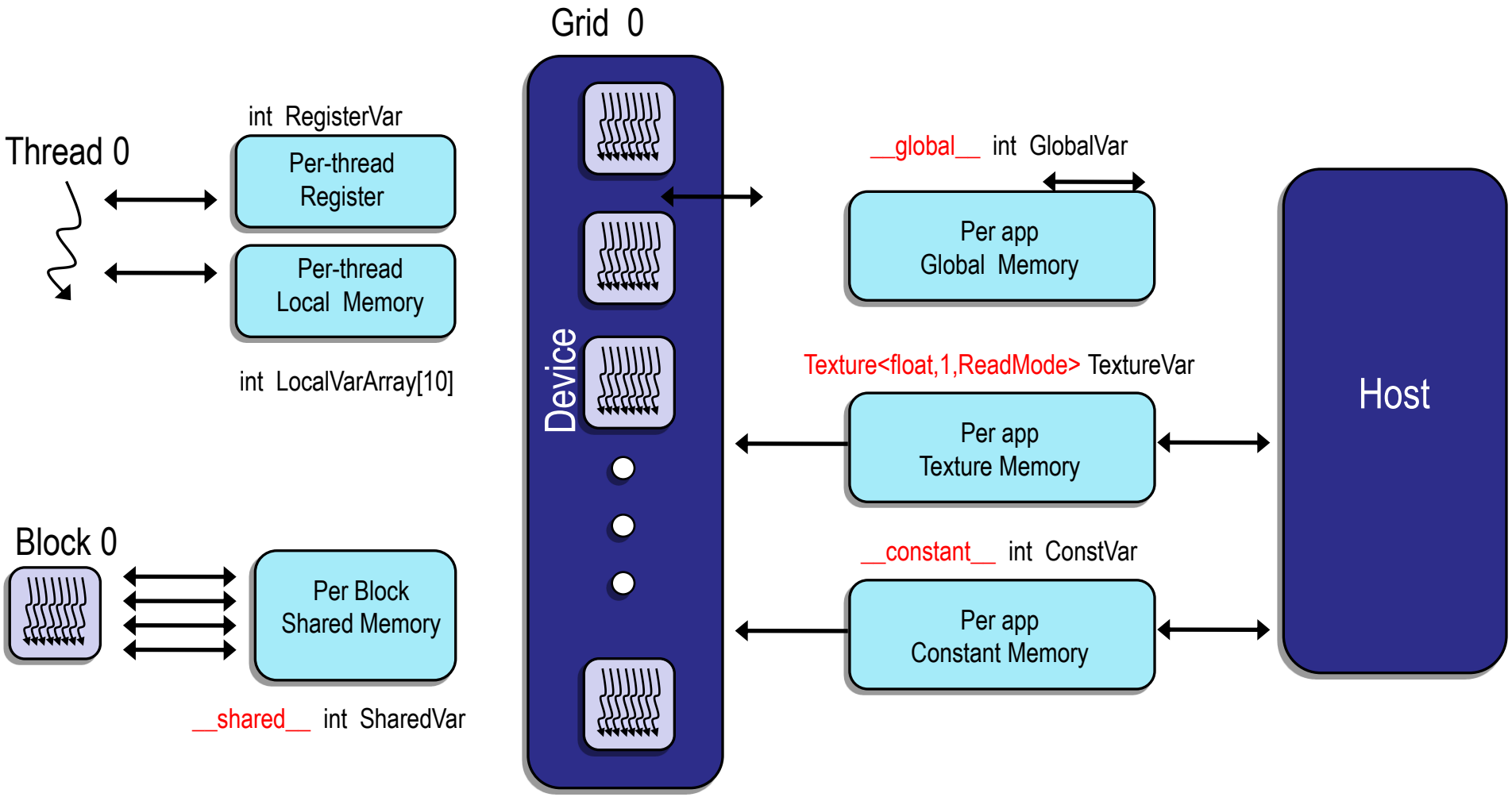




Warp Generation



Memory Hierarchy



Discussion Points

- Who has written CUDA, how have you optimized it, how long did it take?
  - Did you do tune using a better algorithm than trial and error?
- Is there any hope to build a GPU compiler that can automatically do what CUDA programmers do?
  - How would you do it?
  - What's the input language? C, C++, Java, StreamIt?
- Are GPUs a compiler writers best friend or worst enemy?
- What about non-scientific codes, can they be mapped to GPUs?
  - How can GPUs be made more "general"?