Supporting Data-Intensive Distributed Computing

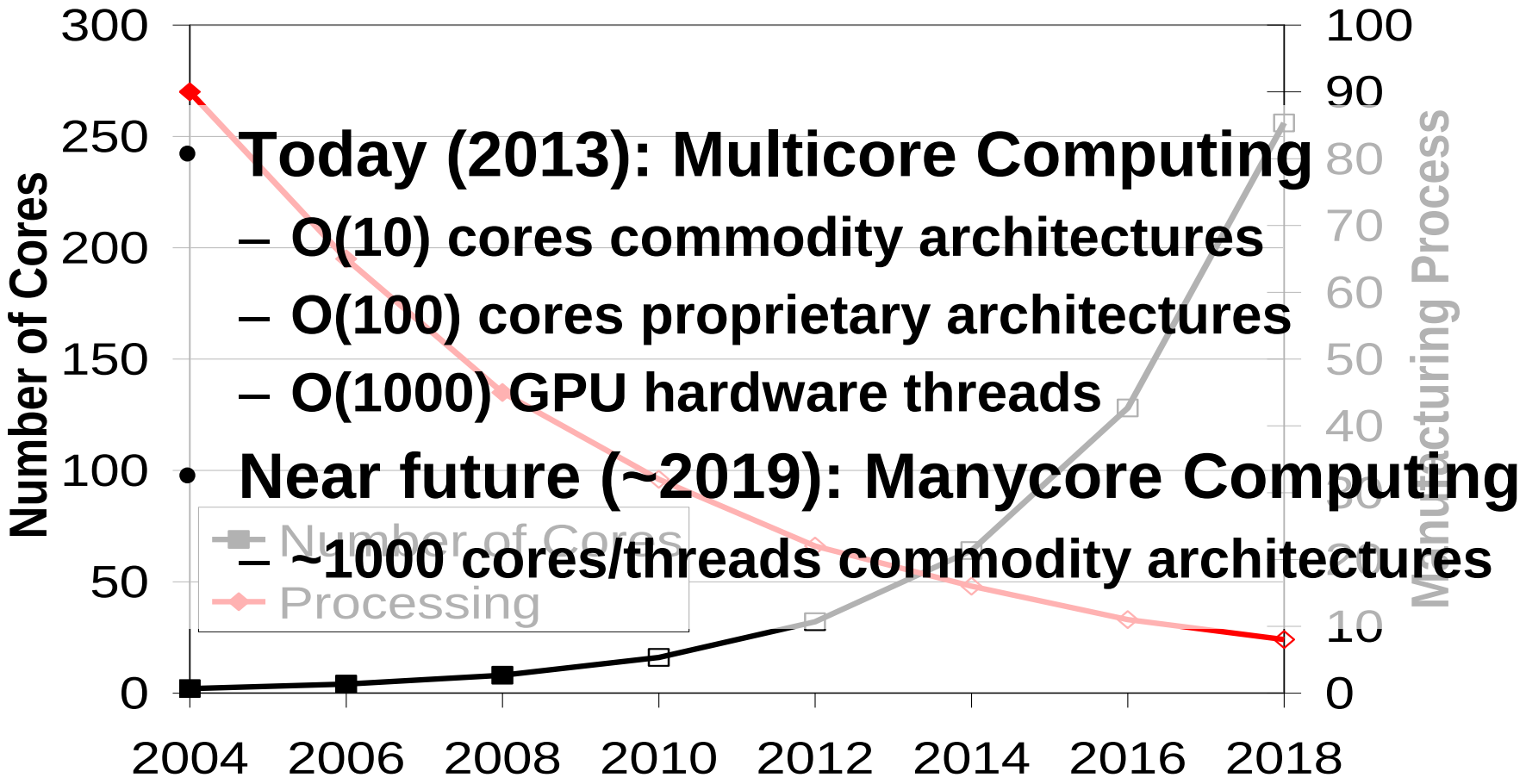
Ioan Raicu
Computer Science Department, Illinois Institute of Technology
Math and Computer Science Division, Argonne National Laboratory

July 18th, 2013
DePaul University

DataSys: Data-Intensive Distributed Systems Laboratory

- **Research Focus**
  - Emphasize designing, implementing, and evaluating systems, protocols, and middleware with the goal of supporting **data-intensive applications on extreme scale distributed systems**, from many-core systems, clusters, grids, clouds, and supercomputers
- **People**
  - Dr. Ioan Raicu (Director)
  - 6 PhD Students
  - 2 MS Students
  - 4 UG Students
- **Contact**
  - http://datasys.cs.iit.edu/
  - iraicu@cs.iit.edu

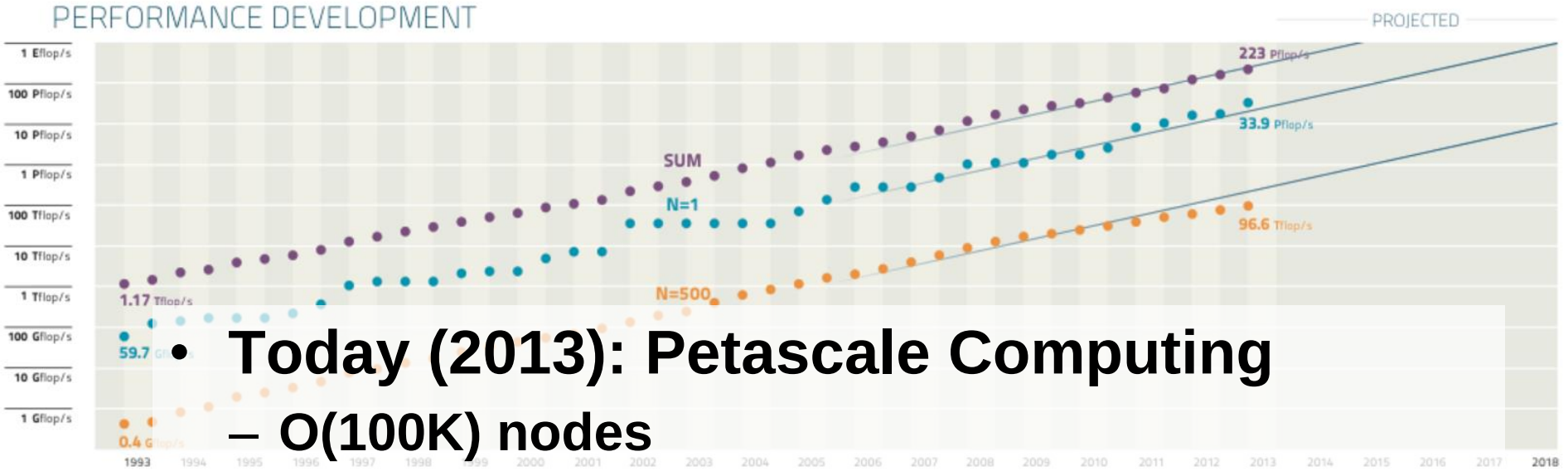
Manycore Computing



- **Today (2013): Multicore Computing**
  - **O(10) cores commodity architectures**
  - **O(100) cores proprietary architectures**
  - **O(1000) GPU hardware threads**
- **Near future (~2019): Manycore Computing**
  - **~1000 cores/threads commodity architectures**

Pat Helland, Microsoft, The Irresistible Forces Meet the Movable Objects, November 9th, 2007

Exascale Computing



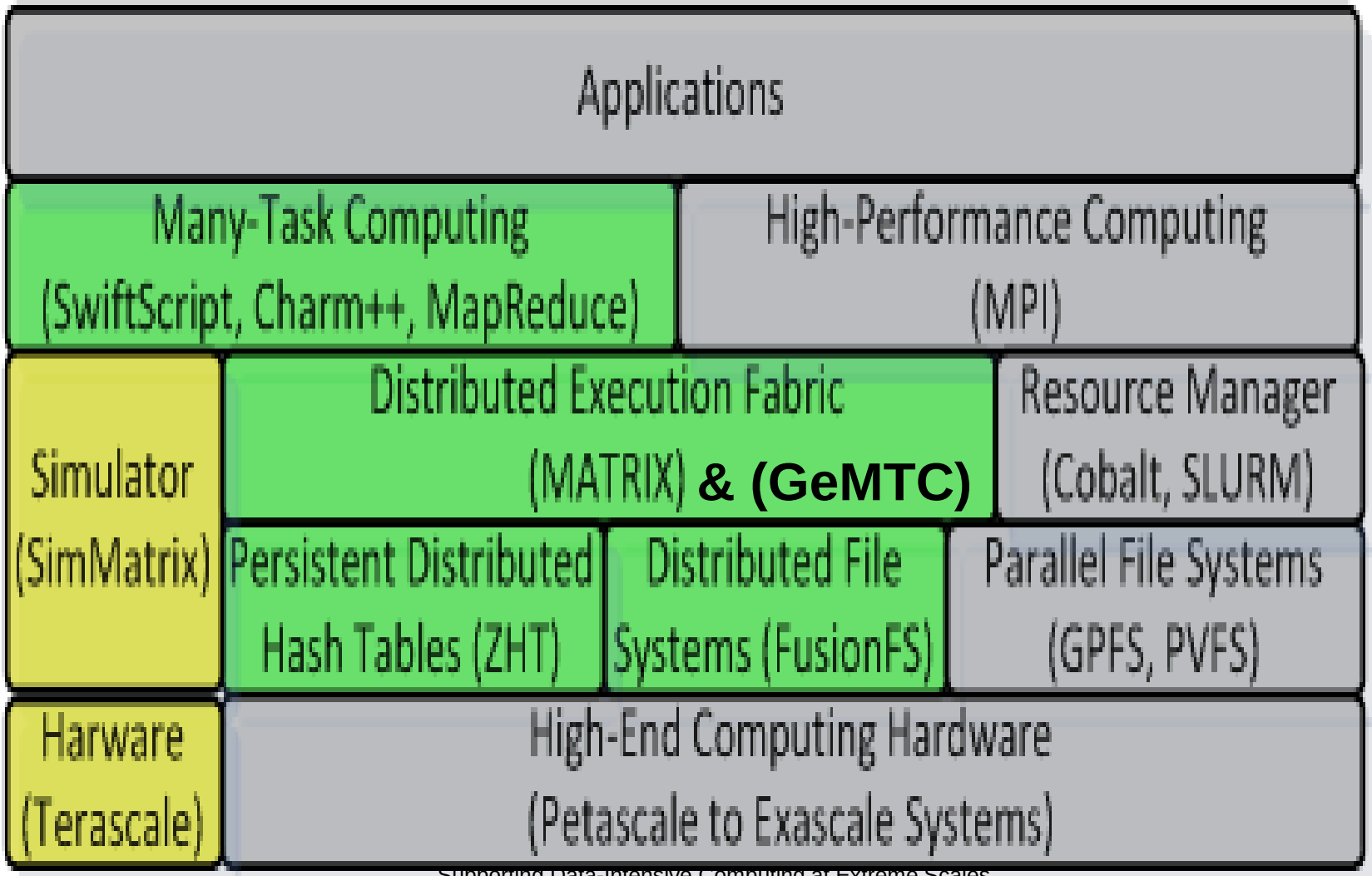
- **Today (2013): Petascale Computing**
  - **O(100K) nodes**
  - **O(1M) cores**

http://www.top500.org/

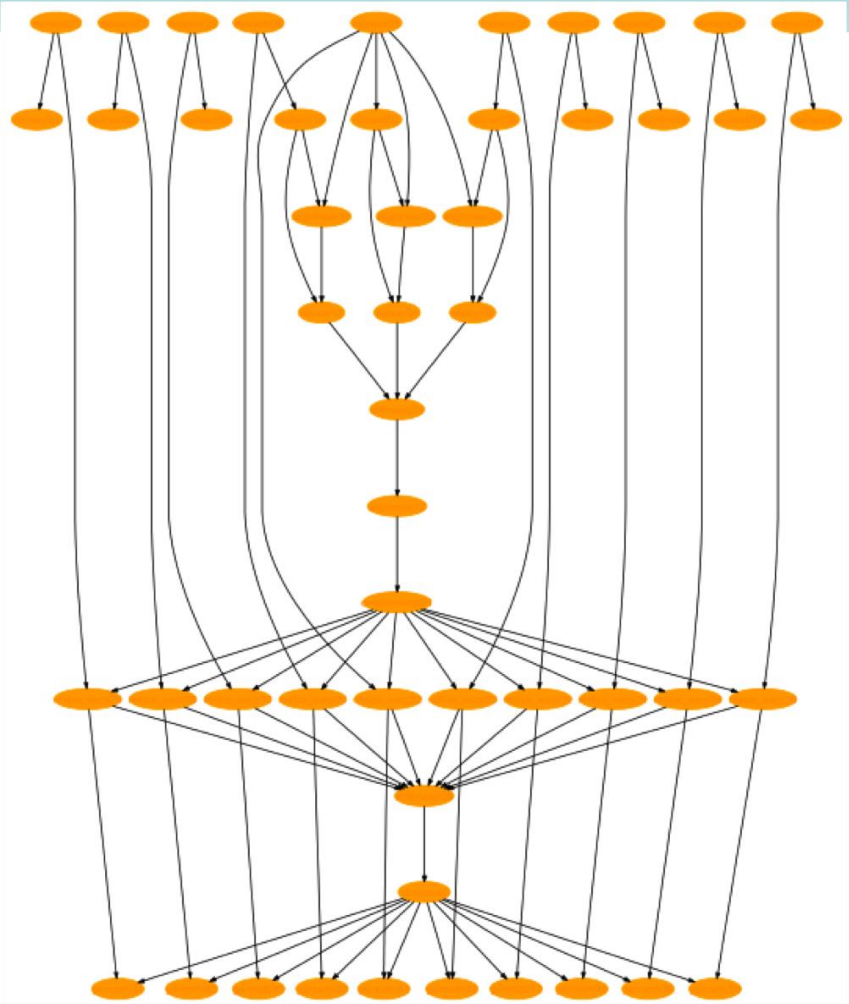
- **Near future (~2018): Exascale Computing**
  - **~1M nodes (10X)**
  - **~1B processor-cores/threads (1000X)**

http://s.top500.org/static/lists/2013/06/TOP500_201306_Poster.png

Proposed Software Stack in Large-Scale Distributed Systems



Many-Task Computing (MTC)



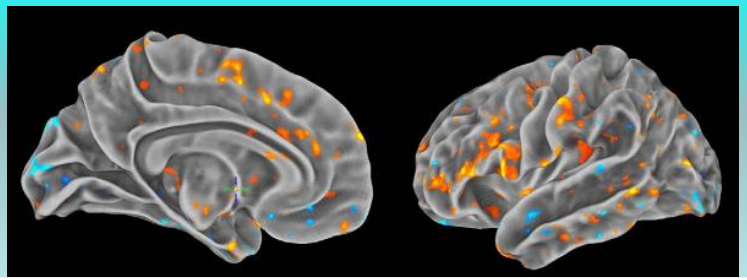
MTC emphasizes:

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

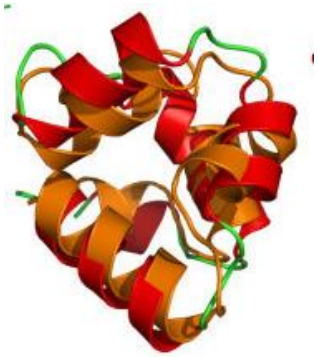
Advantages:

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

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

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 Applications

Field	Description	Characteristics	Status
Astronomy	Creation of montages from many digital images	Many 1-core tasks, much communication, complex dependencies	E
Astronomy	Stacking of cutouts from digital sky surveys	Many 1-core tasks, much communication	E (Falkon)
Biochemistry	Analysis of mass-spec data for post-translational protein modifications	10,000 – 100,000 K jobs for proteomic searches using custom serial codes	D
Biochemistry	Protein folding using iterative fixing algorithm, also exploring other biomolecule interactions	100s to 1000s of 1-1000 core simulations & data analysis	O
Biochemistry	Identification of drug targets via computational screening	Up to 1M x 1 core	O (Falkon)
Bioinformatics	Metagenome modeling	1000's of 1-core integer programming problems	D
Business economics	Mining of large text corpora to study media bias	Analysis and comparison of 70M+ text files of news articles	D
Climate	Ensemble climate model runs and analysis of output data	10s to 100s of 100-1000 core simulations	E
Economics	Generation of response surfaces for various economic models	1K to 1M 1-core runs (10K typical), then data analysis	O
Neuroscience	Analysis of functional MRI datasets	Comparison of images; connectivity analysis with SEM, many tasks (100K+)	O
Radiology	Training of computer aided diagnosis algorithms	Comparison of images; many tasks, much communication	D
Radiology	Image processing and brain mapping for neurosurgical planning research	1000's of MPI application executions	O

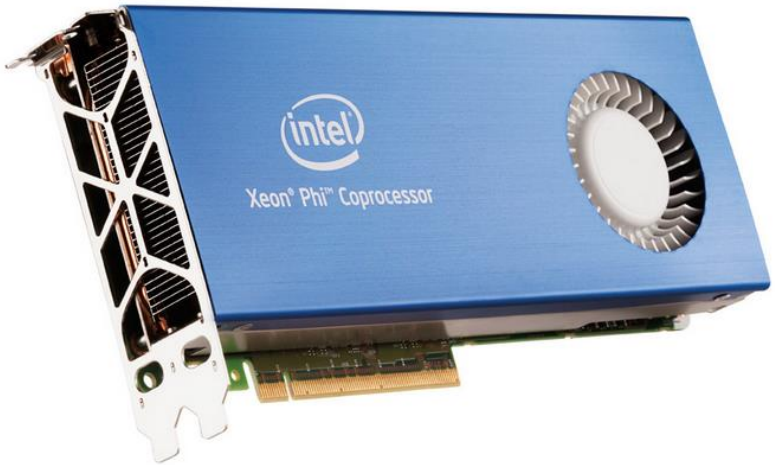
Accelerator Architecture

GPU

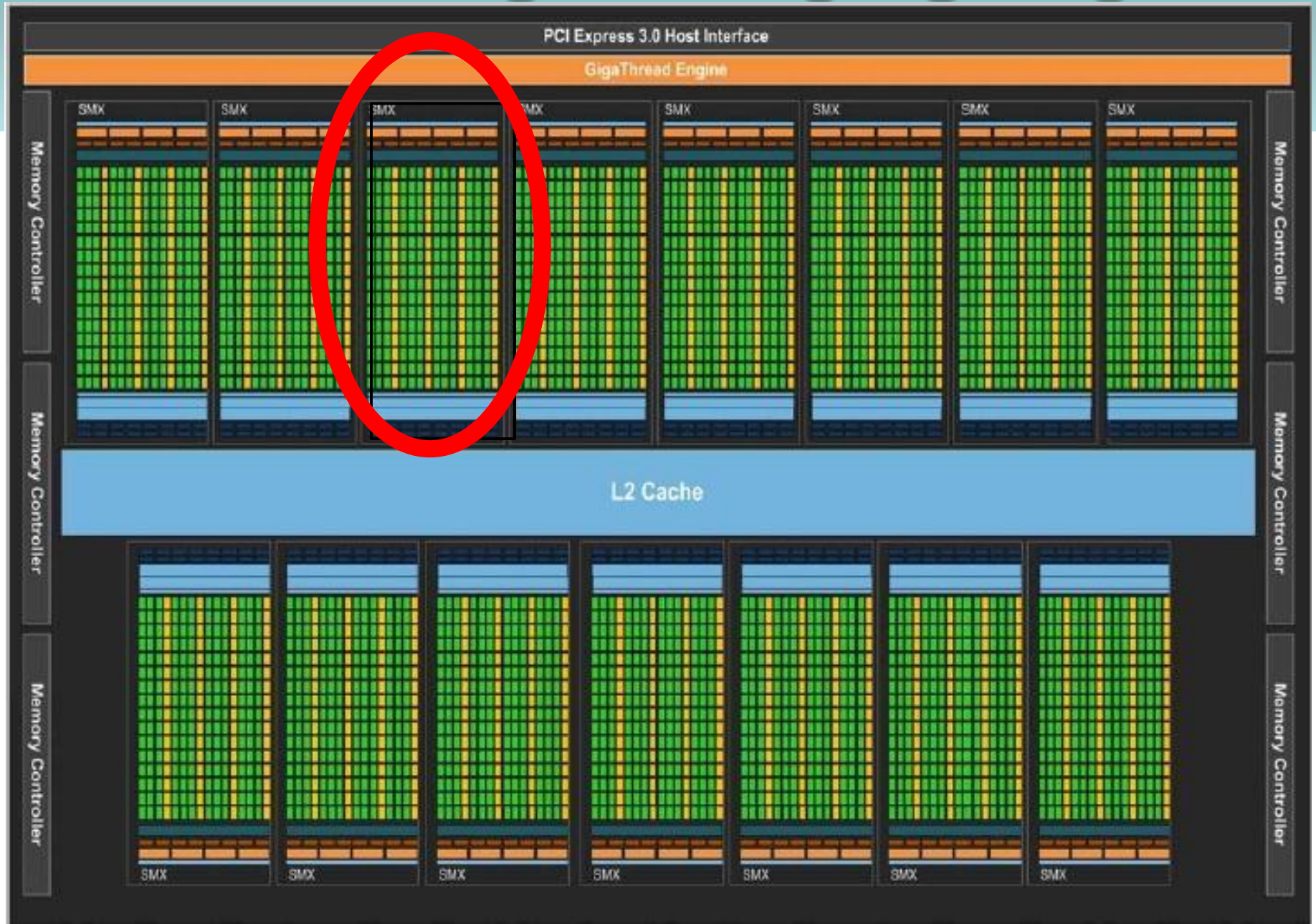
- Streaming Multiprocessors (15 SMXs on Kepler K20)
- Warps
  - 32 threads in a warp
  - 192 warps
    i. hardware available
    ii. ind. compute

Coprocessors

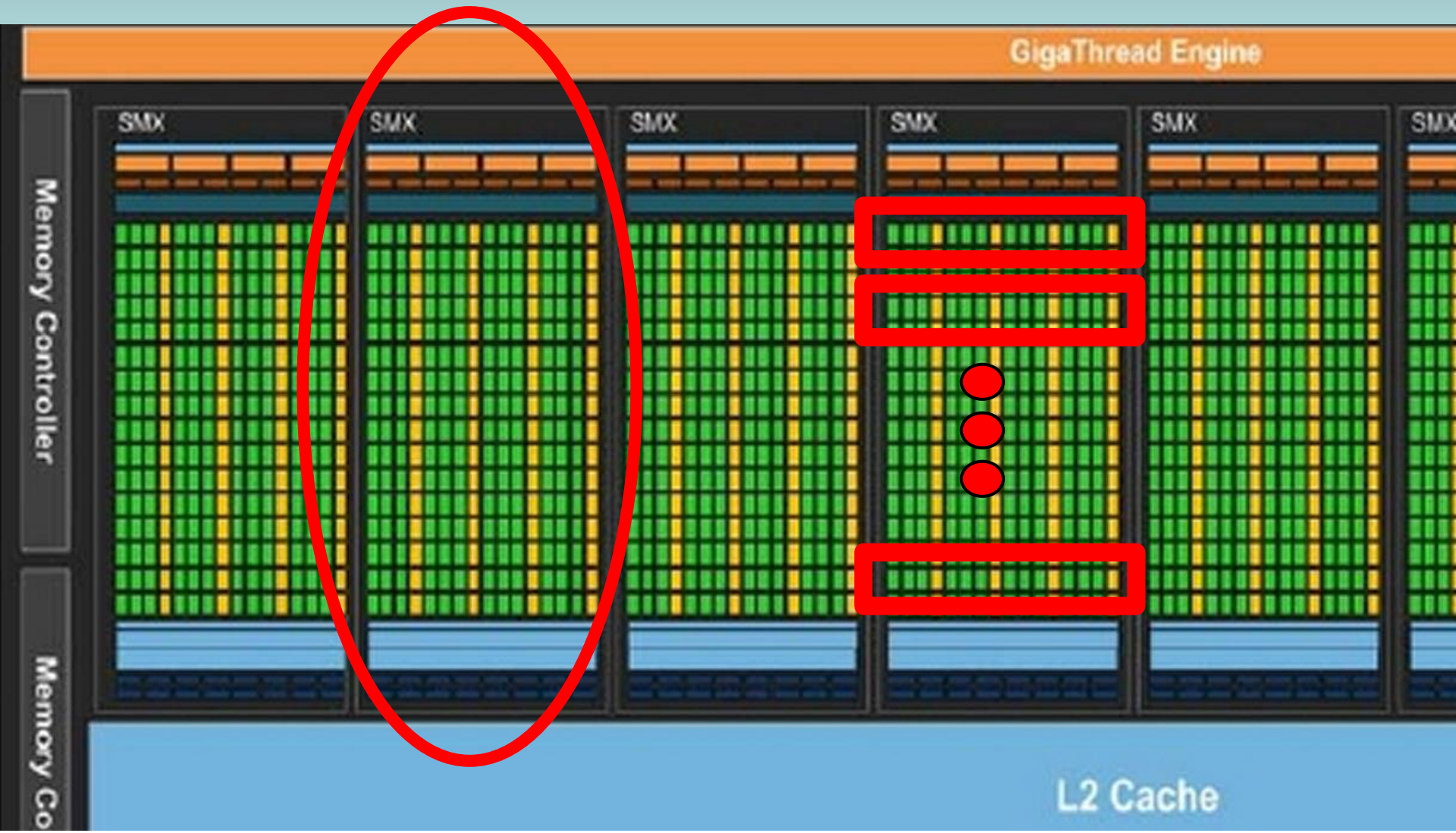
- Intel Xeon Phi
  - 60 cores * 4 threads per core = 240 hardware threads



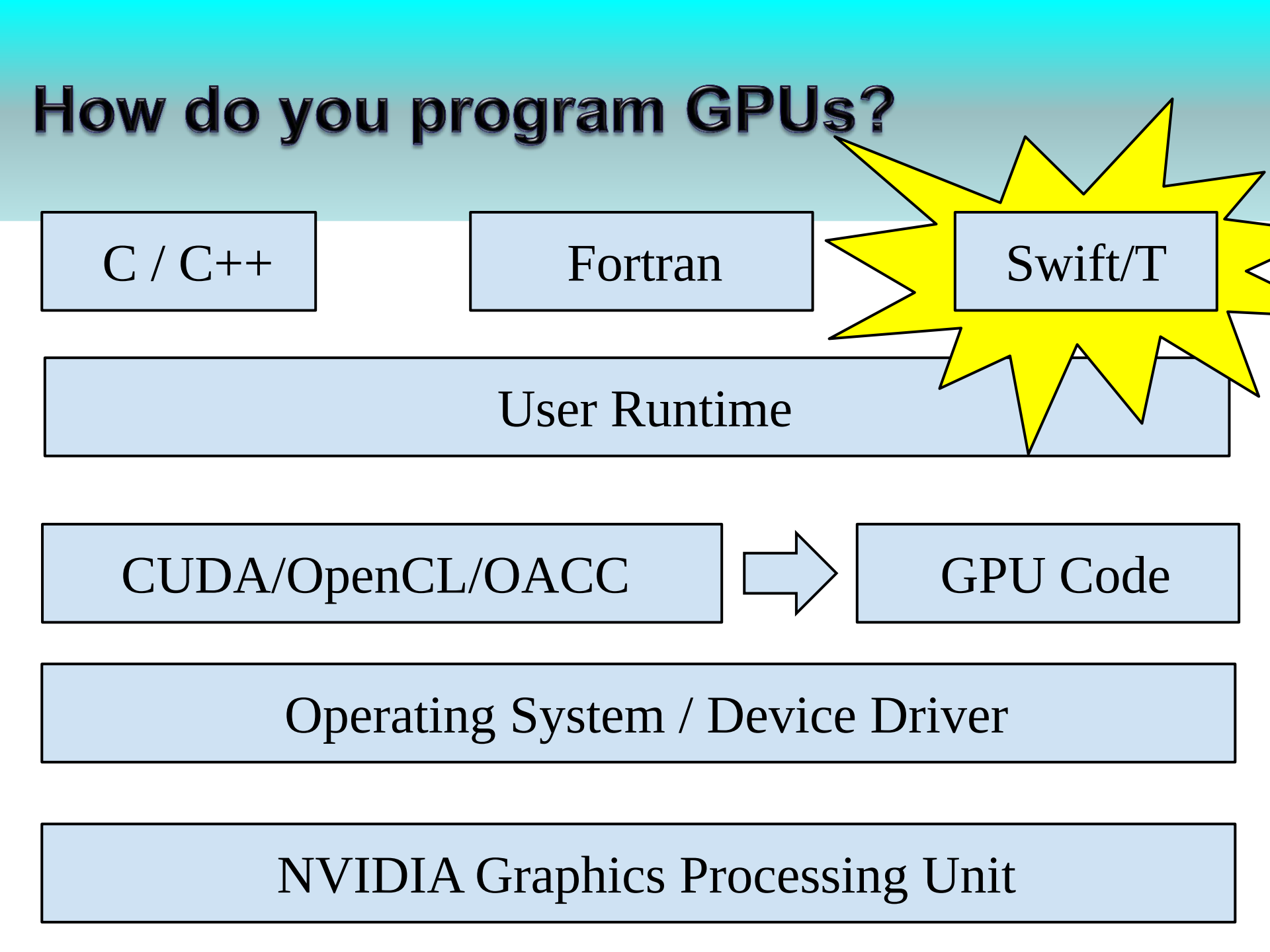
GPU Block Diagram - Highlighting SMX



Highlighting SMX and Warps



How do you program GPUs?



Interested

- Collaborations with groups seeking interesting applications
- Collaborative proposals to NSF or NIH combining Medical Imaging and Distributed Systems (Clouds, Big Data, and/or parallelism)

More Information

- More information:
  - http://www.cs.iit.edu/~iraicu/
  - http://datasys.cs.iit.edu/
- Contact:
  - iraicu@cs.iit.edu
- Questions?