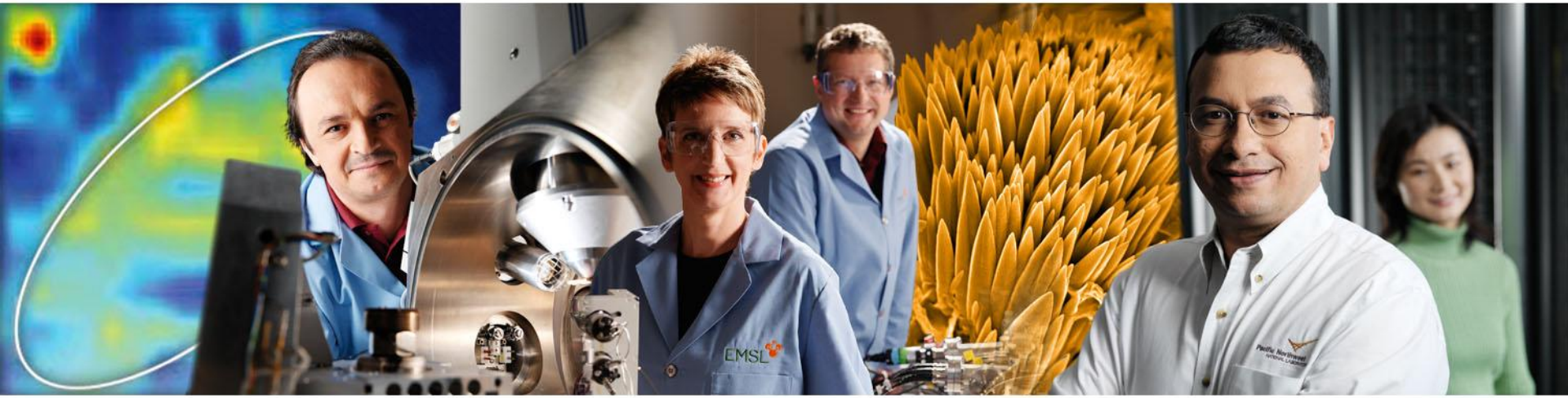
EMSL

Supercomputing for the Masses: Killer-Apps, Parallel Mappings, Scalability and Application Lifespan

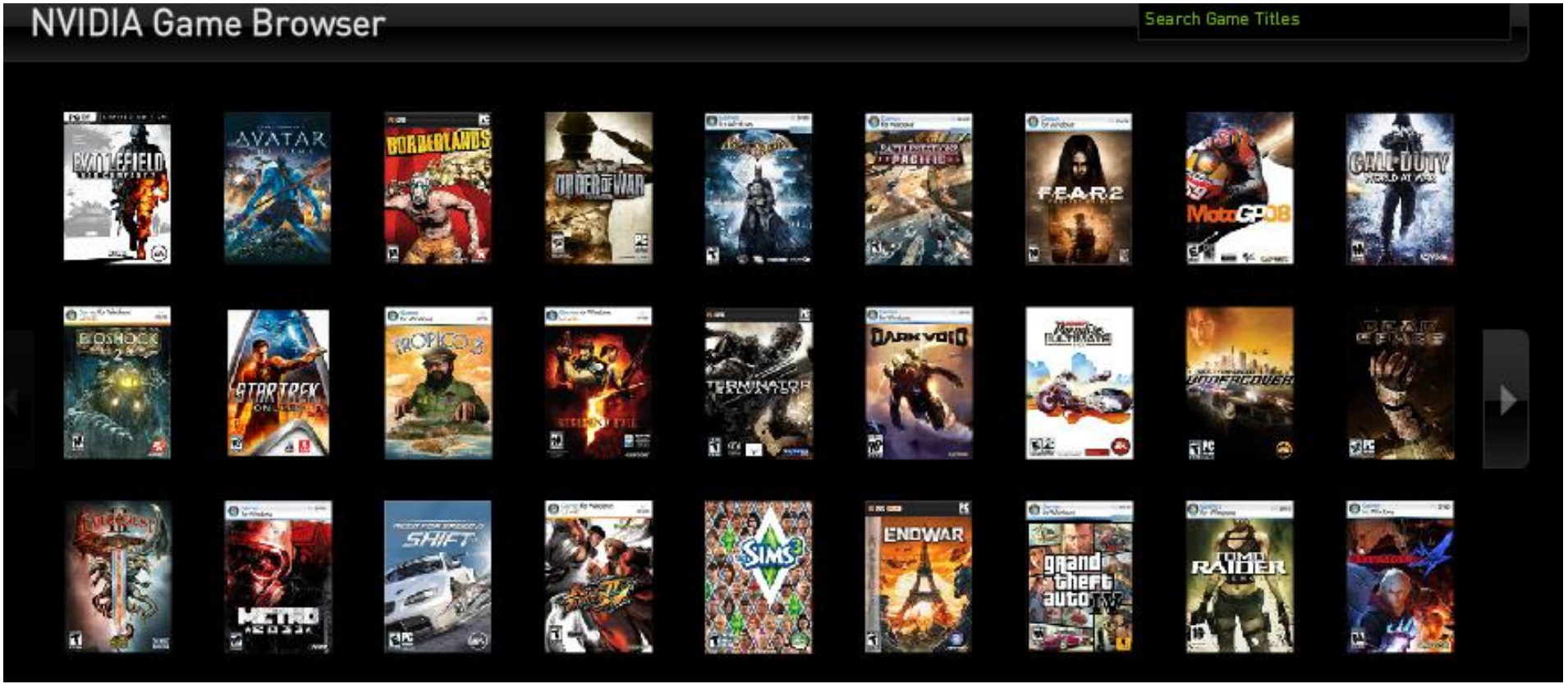
Rob Farber, Senior Scientist, PNNL



Killer Apps

occur when personal vision matches technical capability to fulfill a market demand

- Graphics processors and games: killer apps that created a huge market



Technical capability

- Market forces evolved GPUs into massively parallel GPGPUs (General Purpose GPUs).
- 250+ million CUDA-enabled GPUs says it all!
- CUDA: put supercomputing in the hands of the masses.
  - December 1996, ASCI Red the first teraflop supercomputer
  - Today: kids buy GPUs with flop rates comparable to systems available to scientists with supercomputer access in the mid to late 1990s.
    - GeForce 480 1.35 TF/s peak 32-bit
    - Newegg.com: less than $500

Remember that Finnish kid who wrote some software to understand operating systems? Inexpensive commodity hardware enables:

- New thinking
- A large educated base of developers

A perfect storm of opportunities and technology

(Summary of Farber, *Scientific Computing*, "Realizing the Benefits of Affordable Teraflop-capable Hardware")

- **Multi-threaded software is a *must-have*** because manufacturers were forced to move to multi-core CPUs
  - The failure of Dennard's scaling laws meant processor manufacturers had to add cores to increase performance and entice customers
- Multi-core is disruptive to single-threaded legacy apps
  - Businesses and research efforts will not benefit from new hardware unless they invest in multi-threaded software
  - **Lack of investment risks stagnation and losing to the competition**
- Competition is fierce, the new technology is readily available and it is inexpensive!
  - **Which software and models? Look to those that are:**
    - Widely adopted and have withstood the test of time
    - Look at CUDA and the CUDA model

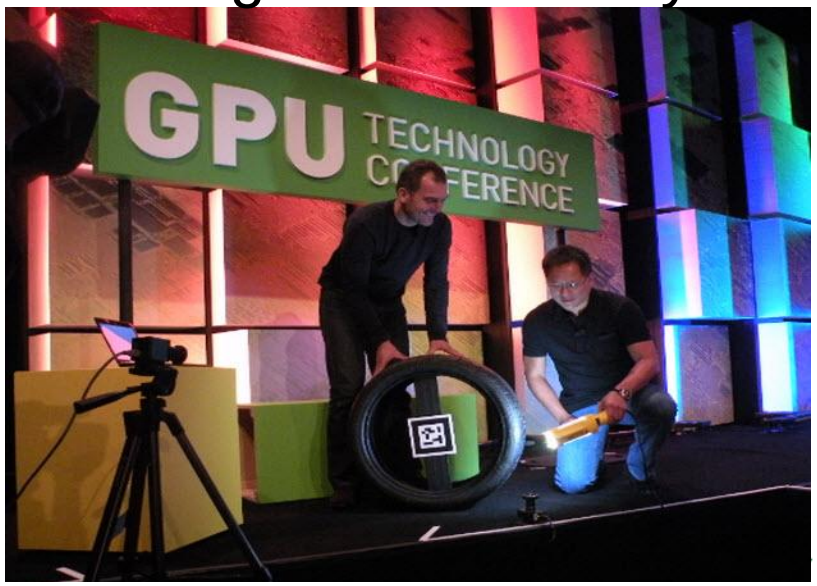
CUDA is not the only game in town
(but will be a focus in this talk)

- Android/Iphone - mobile is huge

(2008) (2009)

5 weeks
- 3.5M downloads
- Over $100K net

Augmented Reality



Jen-Hsun with RTT at 2009 GTC

CUDA is a game changer!

- CUDA enables orders of magnitude faster apps:
  - **10x** can make computational workflows more interactive (even *poorly* performing GPU apps are useful).
  - **100x** is disruptive and has the potential to fundamentally affect scientific research by removing time-to-discovery barriers.
  - **1000x** and greater achieved through the use of the NVIDIA SFU (Special Function Units) or multiple GPUs … Whooo Hoooo!
- In a few slides: examine CUDA + Graphics = Wow!

CUDA was adopted amazingly fast!

- February 2007: The initial CUDA SDK was made public.
- Now: CUDA-based GPU Computing is part of the curriculum at over 360 universities.
  - MIT, Harvard, Cambridge, Oxford, the Indian Institutes of Technology, National Taiwan University, and the Chinese Academy of Sciences.

The numbers have changed!

HPC Source

Redefining What is Possible

Perfect storm of opportunities delivers fresh approaches

Rob Farber

General purpose graphics processor unit (GPGPU) technology has arrived during a perfect storm of opportunities. Multi-threaded software is now a necessity as x86 and other conventional processor designs have been forced to adopt a multi-core approach. From dual core cell phones to IBM Power 7 systems that will support well over a million concurrent threads of execution, parallelism is now the path to performance.

Legacy applications and research efforts that do not invest in multi-threaded software will not benefit from modern multi-core processors, because single-threaded and poorly scaling software will not be able to utilize extra processor cores. As a result, computational performance will plateau at or near current levels, placing the projects that depend on these legacy applications at risk of both stagnation and loss of competitiveness.

Graphic processors have matured into general purpose computational devices at exactly the right time to be considered in this industry-wide retooling to utilize multi-threaded parallelism. To put this in very concrete terms, any teenager (or research effort) from Beijing, China, to New Delhi, India, can purchase a teraflop capable graphics processor and start developing and testing massively parallel applications. Table 1 shows two inexpensive teraflop-capable offerings from AMD and NVIDIA that are available now for purchase.

These devices represent a peak floating-point capability that was beyond anything available for the most advanced high performance computing (HPC) users until Sandia National Laboratory performed a trillion floating-point operations per second in December 1996 on the ASCI Red supercomputer. I wonder how many of those proposals for leading-edge research using a teraflop supercomputer can be performed today by students anywhere in the world using a few GPGPUs in a workstation with a fast RAID disk subsystem and a decent amount of host system memory.

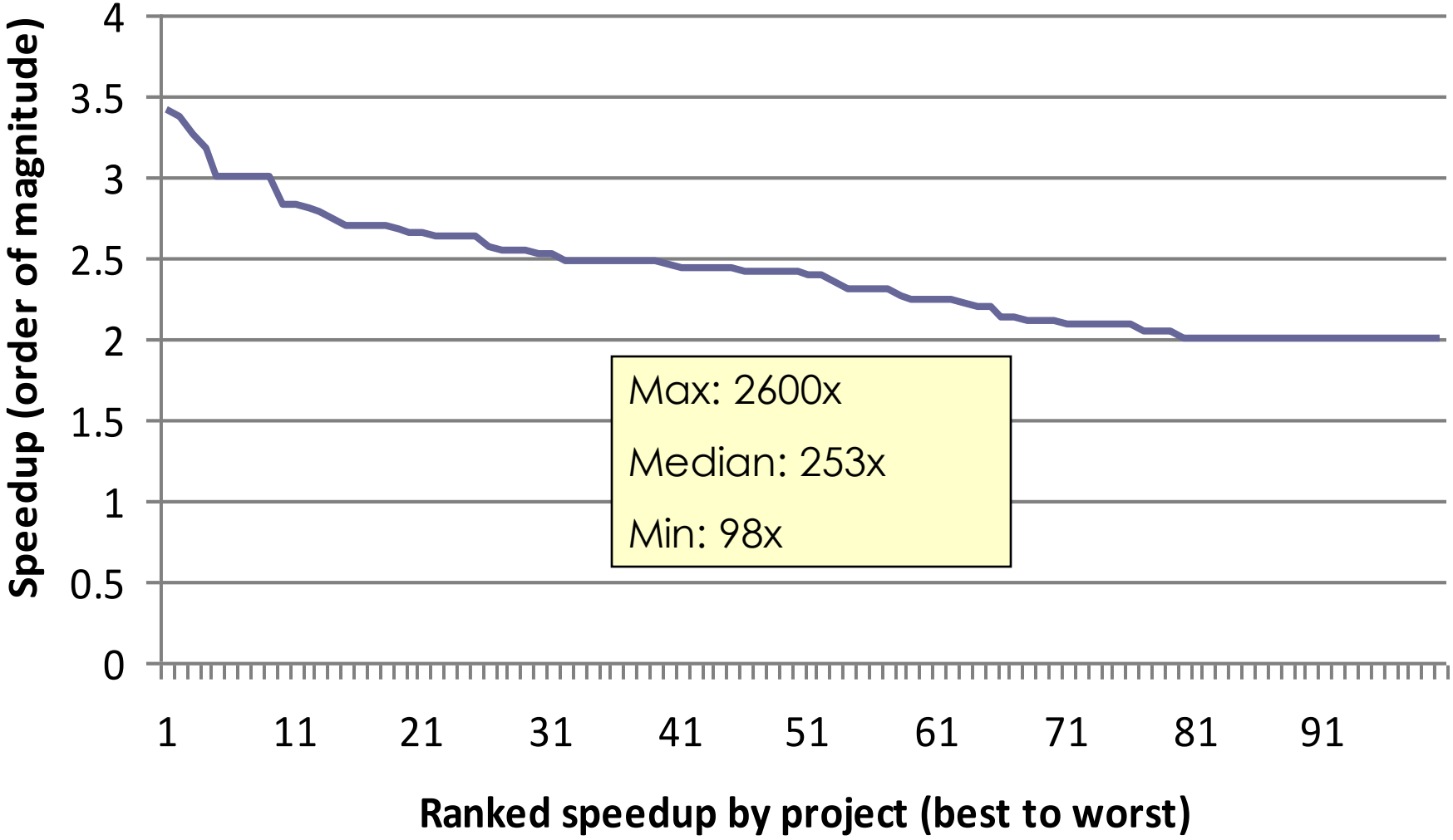
From personal experience, current GPGPU flop rates meet or exceed the computational capability to which I had access as a scientist in the theoretical division at Los Alamos National laboratory in the late 1990s. In addition, the machines I used were shared with other users, while current GPGPUs are inexpensive enough to be dedicated for use by a single individual. Installing four high-end GPUs in a workstation can create a machine with a peak flop rate comparable to the large MPP2 supercomputer that Pacific Northwest National Laboratory (PNNL) made available to users just a few years ago.

Competition is fierce in both commercial and academic circles, which is why commodity supercomputing in the hands of the masses is going to have a huge impact on both commercial products and scientific research. Plus, GPGPU technology has made the competition global and

HPC Source 6 SC10 Special Edition

http://e-ditionsbyfry.com/Olive/ODE/SCISupp/Default.aspx?href=SCI/2010/11/03&pageno=06&view=document

Application speed says it all!
(fastest 100 apps in the NVIDIA Showcase Sept. 8, 2010)



URL: http://www.nvidia.com/object/cuda_apps_flash_new.html click on **Sort by Speed Up**

Orders of magnitude increased performance in an extraordinary number of fields

- Spanning a wide-range of computational, data driven, and real-time applications:
  - Computational finance
  - Medical
  - Quantum chemistry simulations
  - Molecular modeling and electrostatic potentials
  - Diffusion
  - Fluid flow
  - Systems of differential equations
  - Data driven problems such as microscopy
- Many can be considered killer apps in their field.

An example: the Metropolis algorithm 300x – 1000x

- Among the ten algorithms that have had the greatest influence on the development and practice of science and engineering in the 20th century (Beichl, Sullivan, 2000).
- Plays a significant role in statistics, econometrics, physics and computing science.
  - For some applications, MCMC simulation is the only known general approach for providing a solution within a reasonable time (Diaconis, Saloff-Coste, 1995).
- CUDA version reported to be 300x to 1000x faster (Alerstam, Svensson, Engels, 2008).

Three rules for fast GPU codes

1. Get the data on the GPU (and keep it there!)
   - PCIe x16 v2.0 bus: 8 GiB/s in a single direction
   - 20-series GPUs: 140-200 GiB/s

2. Give the GPU enough work to do
   - Assume 10 μs latency and 1 TF device
   - Can waste ($10^{-6} * 10^{12}$) = 1M operations

3. Reuse and locate data to avoid global memory bandwidth bottlenecks
   - 10^{12} flop hardware delivers 10^{10} flop when global memory limited
   - Can cause a 100x slowdown!

Application lifespan
SIMD: a key from the past

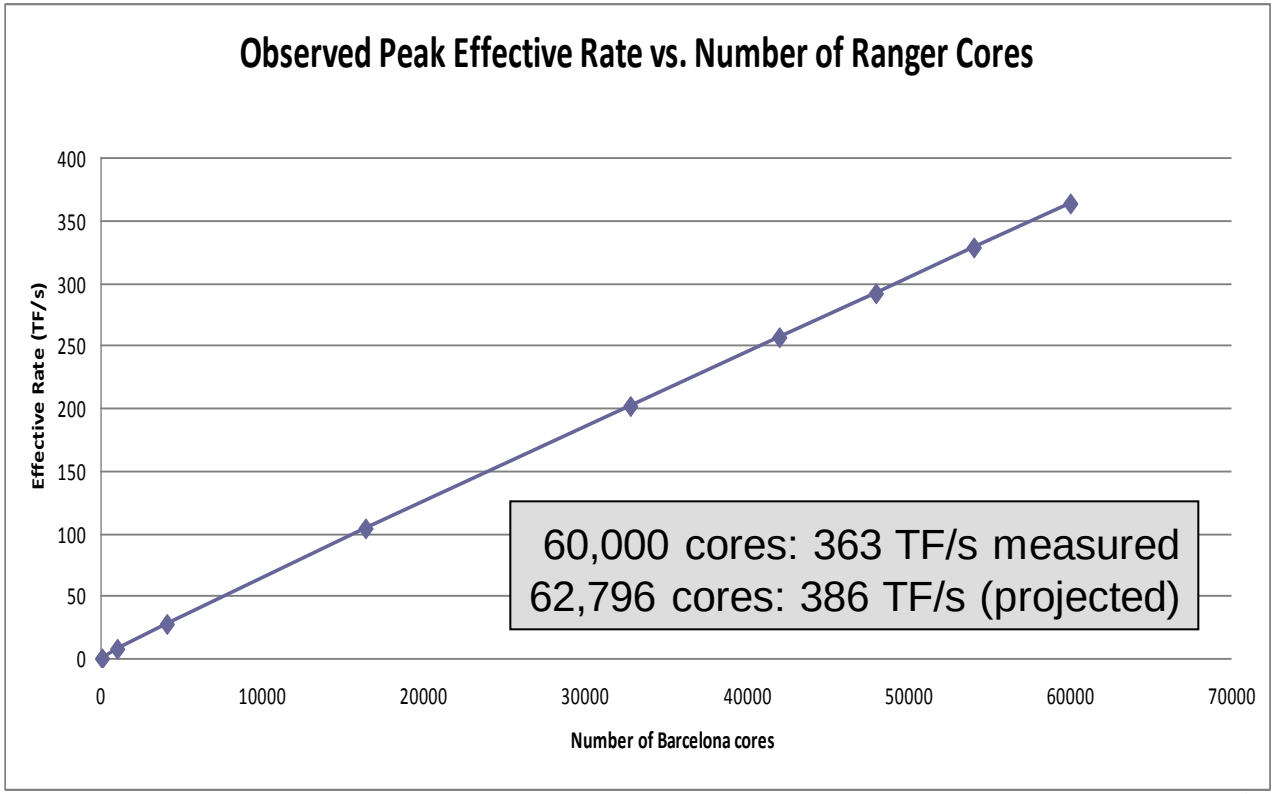
Farber: general SIMD mapping from the 1980s

- Acknowledgements: Work performed at or funded by the Santa Fe Institute, the theoretical division at Los Alamos National Laboratory and various NSF, DOE and other funding sources including the Texas Advance Computer Center.

The Connection Machine

This mapping for Neural Networks …

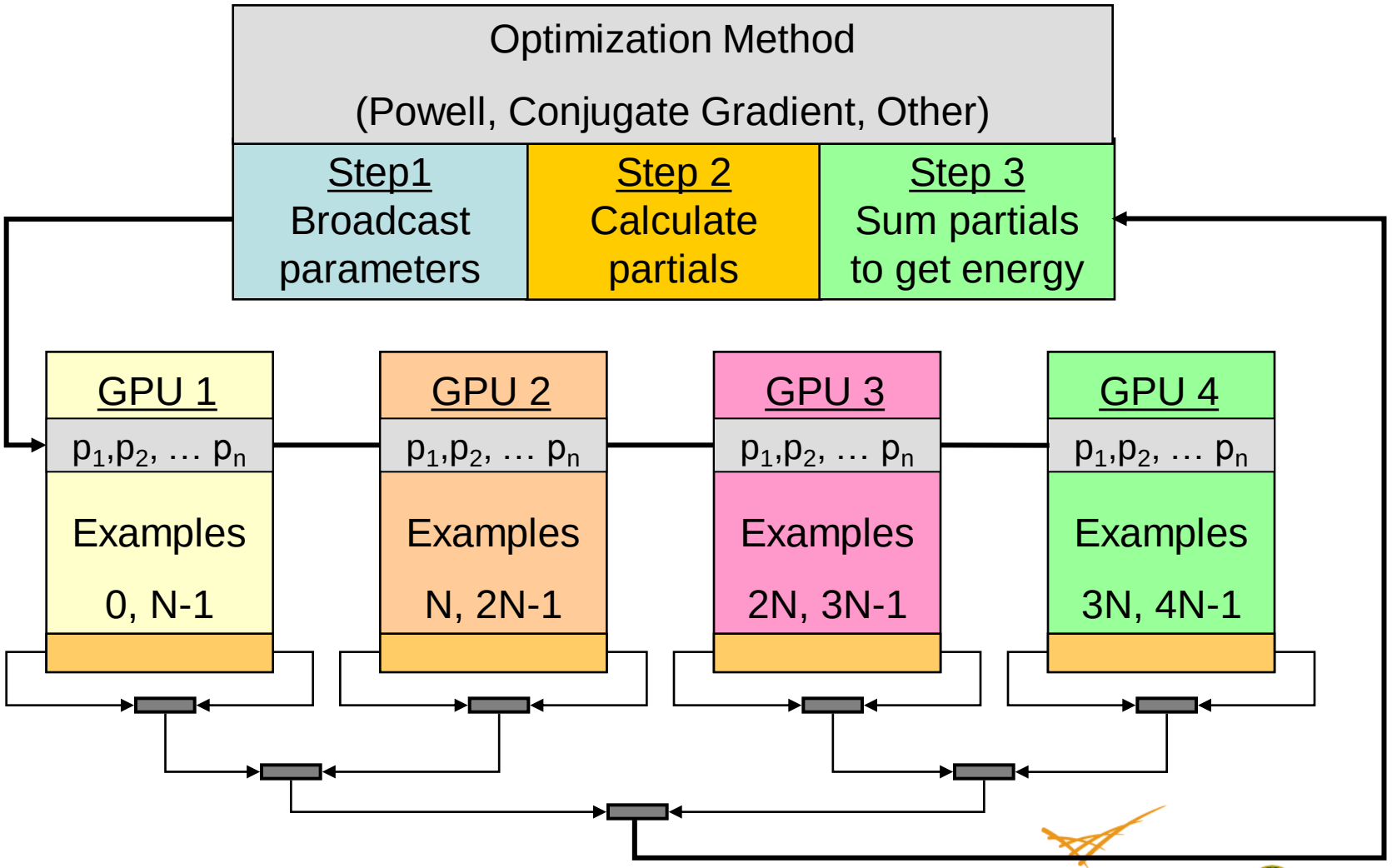
"Most efficient implementation to date" (Singer 1990), (Thearling 1995)



Results presented at SC09 (courtesy TACC)

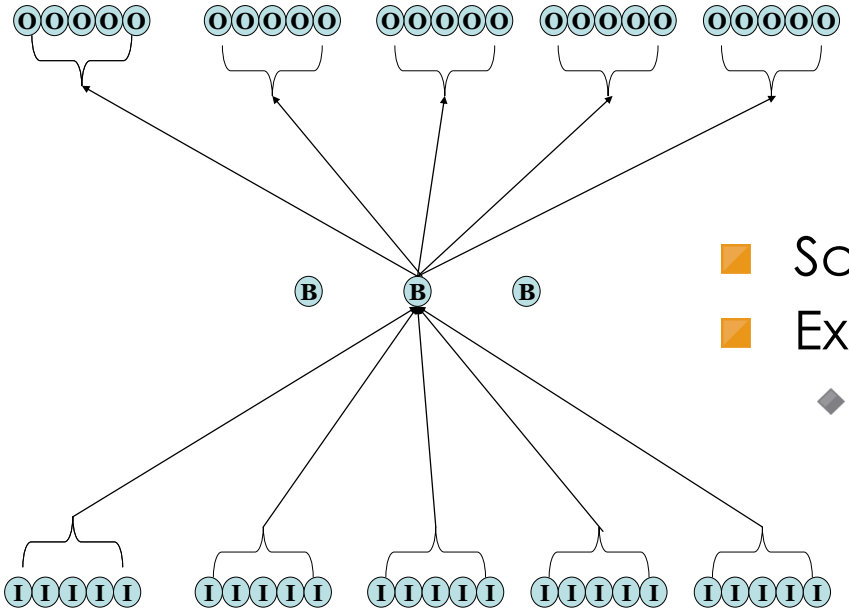
The Parallel Mapping

energy = objFunc(p_1, p_2, … p_n)



One example: Principle Components Analysis (PCA)

- A widely used technique in data-mining and data reduction
  - Demonstrate a method proposed by Sanger (1989)



- Scales according to data
- Extends to Nonlinear PCA (NLPCA)
  - E. Oja, J. Karhunen, L. Wang, and R. Vigario, 1995

This is a general mapping
(think of your own applications!)

- Optimization
- Locally Weighted Linear Regression (LWLR) LWLR
- Neural Networks
- Naive Bayes (NB)
- Gaussian Discriminative Analysis (GDA)
- k-means
- Logistic Regression (LR)
- Independent Component Analysis (ICA)
- Expectation Maximization (EM)
- Support Vector Machine (SVM)
- Others: (MDS, Ordinal MDS, etcetera)

Results

=

The Connection Machine

$* C_{NVIDIA}$

(where $C_{NVIDIA} >> 1$)

What is C_{NVIDIA} for modern x86_64 machines?

- Linear PCA

	Average 100 iterations (sec)
8x core*	0.578
C2050 **	0.00482
speedup	**15x**
vs. 1 core	**120x (measured)**

* 2x Intel (quadcore) E5540s @ 2.53 GHz, openmp, SSE enabled via g++

- Nonlinear PCA

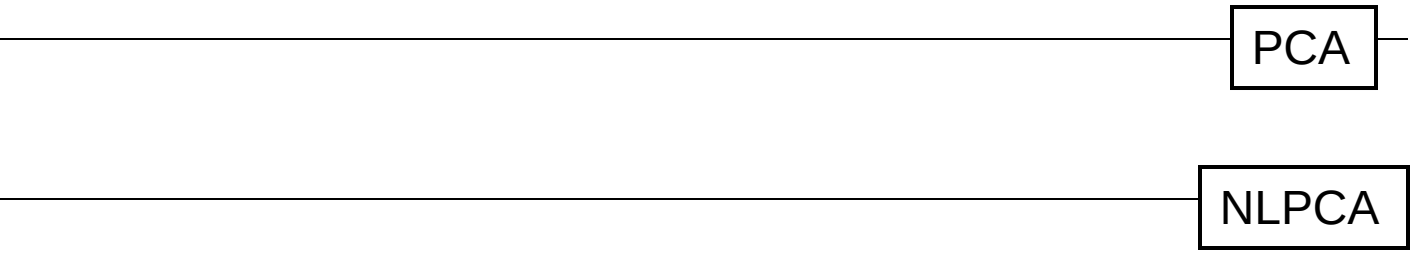
	Average 100 iterations (sec)
8x core*	0.4389
C2050 **	0.0076
speedup	**58x**
vs. 1 core	**425x (measured)**

** includes all data transfer overhead ("Effective Flops")

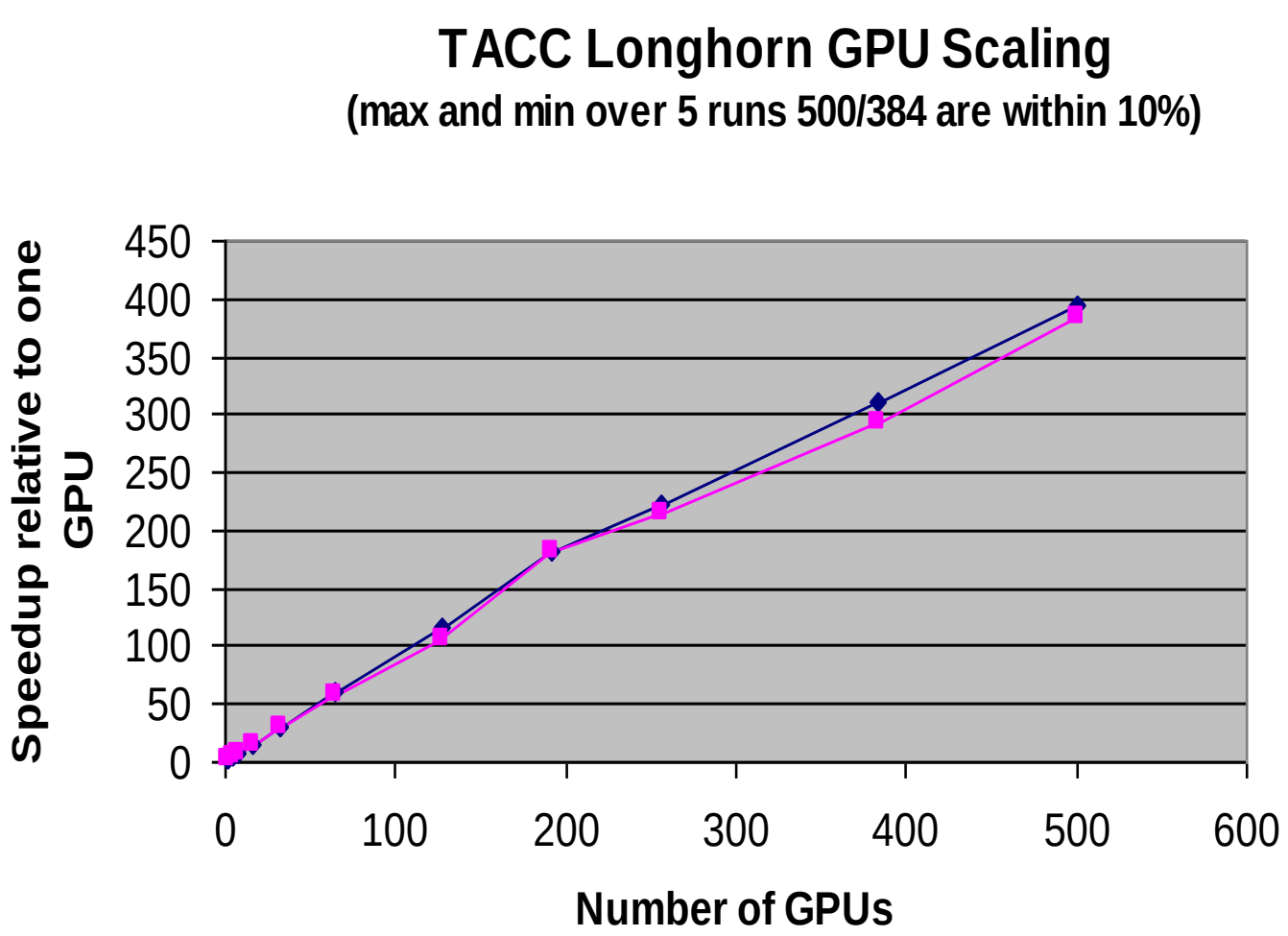
Time includes all overhead! (effective rate or "honest flops")

$$EffectiveRate = \frac{TotalOpCount}{T_{broadcastParam} + T_{func} + T_{reduce}}$$

- Memory bandwidth is key
  - More SMP cores does not translate to faster performance!
  - Previous results: single-core was faster than 1/8th of an 8-core run



Scalability across GPU/CPU cluster nodes
(In collaboration with Ted Hromadka – a UCSD grad student)



Tihane-1a
7,168
NVIDIA® Tesla™ M2050 GPUs
(4.04 MW)

Nebulae
(星雲)
Shenzhen, China

Oak Ridge National Laboratory looks to NVIDIA "Fermi" architecture for new supercomputer

NERSC experimental GPU cluster: Dirac

EMSL experimental GPU cluster: Barracuda

Looking into my crystal ball

I predict long life for GPGPU applications

- Efficient CUDA codes will stay around
  - SIMD/SPMD/MIMD mapping translate well to new architectures
  - CUDA is an excellent way to create these codes
  - Previous SIMD example is still solving important problems

- Will these applications always be written in CUDA?

- Data-parallel extensions are hot!

Thrust: a very good thing!

- http://code.google.com/p/thrust/
- The primary developers of Thrust are:
  - Jared Hoberock
  - Nathan Bell
- Others acknowledged
  - Mark Harris
  - Michael Garland
  - Nadathur Satish
  - Shubho Sengupta

```
int main(void)
{
  // generate random data on the host
  thrust::host_vector<int> h_vec(100);
  thrust::generate(h_vec.begin(), h_vec.end(), rand);

  // transfer to device and compute sum
  thrust::device_vector<int> d_vec = h_vec;
  int x = thrust::reduce(d_vec.begin(), d_vec.end(), 0, thrust::plus<int>());
  return 0;
}
```

Example from website

Expect good things from Copperhead: the data-parallel Python project!

CUDA made simple

- Most of the actual code from the PCA and NLPCA examples

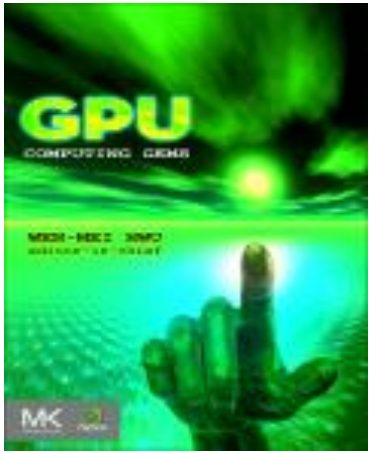
```
…
cudaMemcpyToSymbol("constP", &h_P[0], sizeof(float)*nParam, 0,
                                                    cudaMemcpyHostToDevice);

FcnOfInterest objFcn(input);

energy = thrust::transform_reduce(
                                    thrust::counting_iterator<int>(0),
                                    thrust::counting_iterator<int>(nExamples),
                                    objFcn,
                                    0.0f,
                                    thrust::plus<Real>());
```

Killer apps: the computational technology is here!

- A myriad of other examples are available
  - GPU Computing Gems edited by Wen-mei W. Hwu
  - GPGPUcomputing.net
  - The NVIDIA showcase
  - Many others



- Tool kits
  - CUFFT (IMHO an excellent package)
  - OpenVidia
  - Plus many others

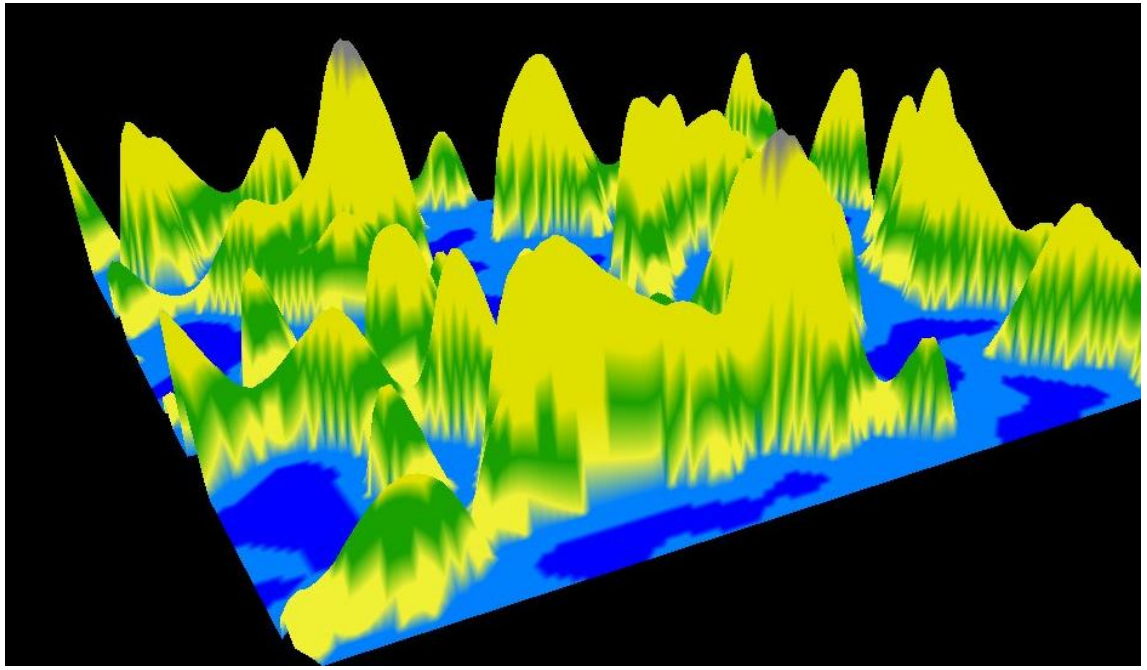
CUDA + Graphics (a potent combination!)

- **Primitive restart**: define an index value to be used as a tag that tells OpenGL that the next vertex starts a new OpenGL primitive of the same type
  - Keep the data on the GPU
  - Avoids the PCIe bottleneck
  - Variable length data works great!
  - A feature of OpenGL 3.1
- Output only: visualization, rendering and games
- Combine with vision recognition: Augmented Reality!

A primitive restart virtual world example

- Perlin Noise

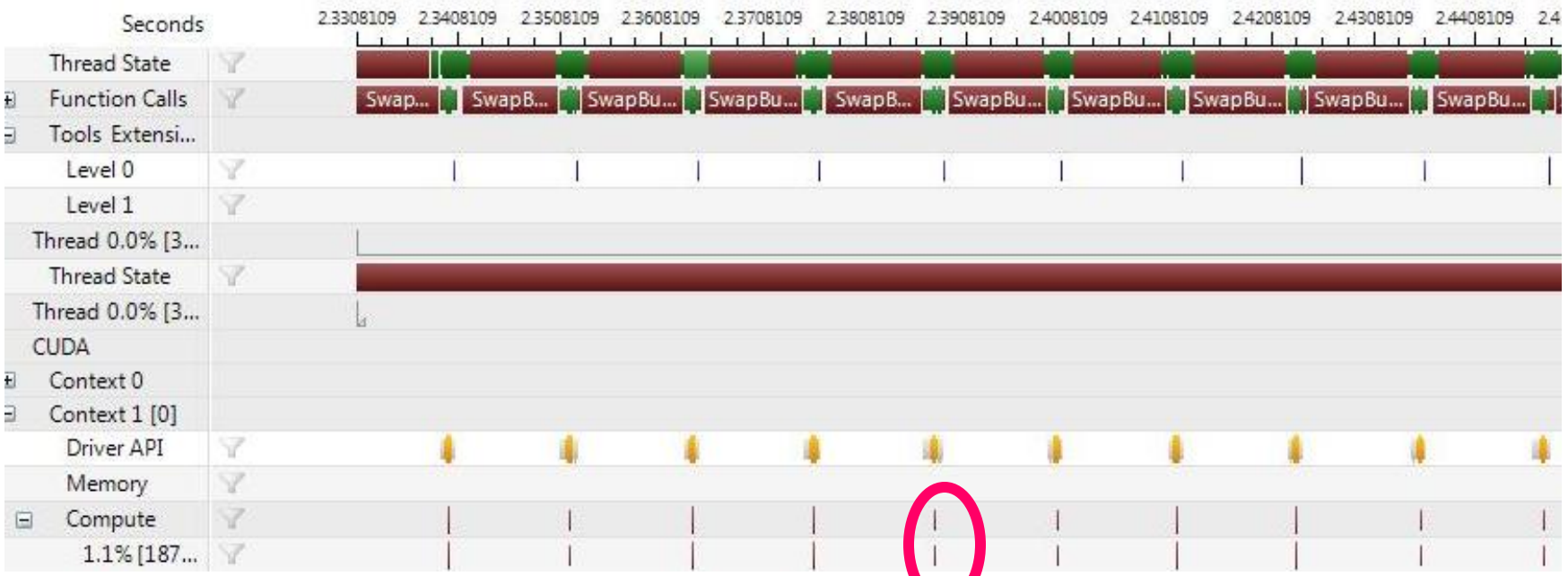
Important for games and the movie industry: Ken Perlin won an Academy Award for this noise generator



- Primitive restart can be 100 FPS faster than other rendering methods and delivers higher quality images
  - Rendering performance can be optimized by arranging the indices to achieve the highest reuse in the texture units.
  - Higher quality images can be created by alternating the direction of tessellation
  - Avoid the PCIe bus!

Farber DDJ Part 18

Parallel Nsight shows the speed



Generate a 512x512 heightmap using Perlin noise

- Primitive restart: around 60 μs.
- Multidraw: around 3,900 μs.
- Iteratively drawing each triangle fan: approximately 1,100,000 μs.

Farber DDJ Part 20

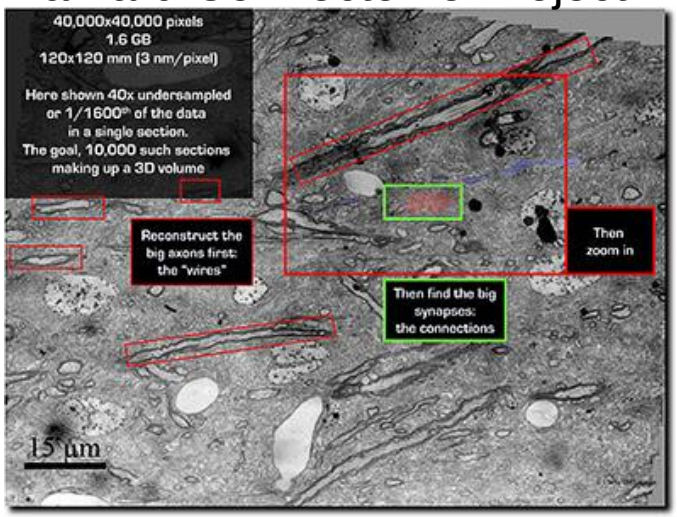
Predicting future killer apps?

Humility: five years ago I would not have believed:

- Adding four PCIe devices will give my workstation roughly the same peak flop rate as the largest PNNL supercomputer

- It is now possible to get the full 3D wiring diagram for the entire brain of a cat or mouse

Harvard Connectome Project



Killer apps: when personal vision meets technical capability

- The Connectome project: A Galilean first opportunity for scientists to examine the detailed schematic diagram that nature uses for vision and cognition.
- SC09: computers can simulate an entire cat brain
  - "**The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses**", Ananthanarayanan, Esser, Simon, and Modha (2009).
- My prediction: combining detailed brain models with sufficient computational capability will be a killer app.
  - People studied birds and (eventually) created supersonic aircraft
  - With nature's wiring diagram for vision & language, (eventually) …?

What is *your* vision?