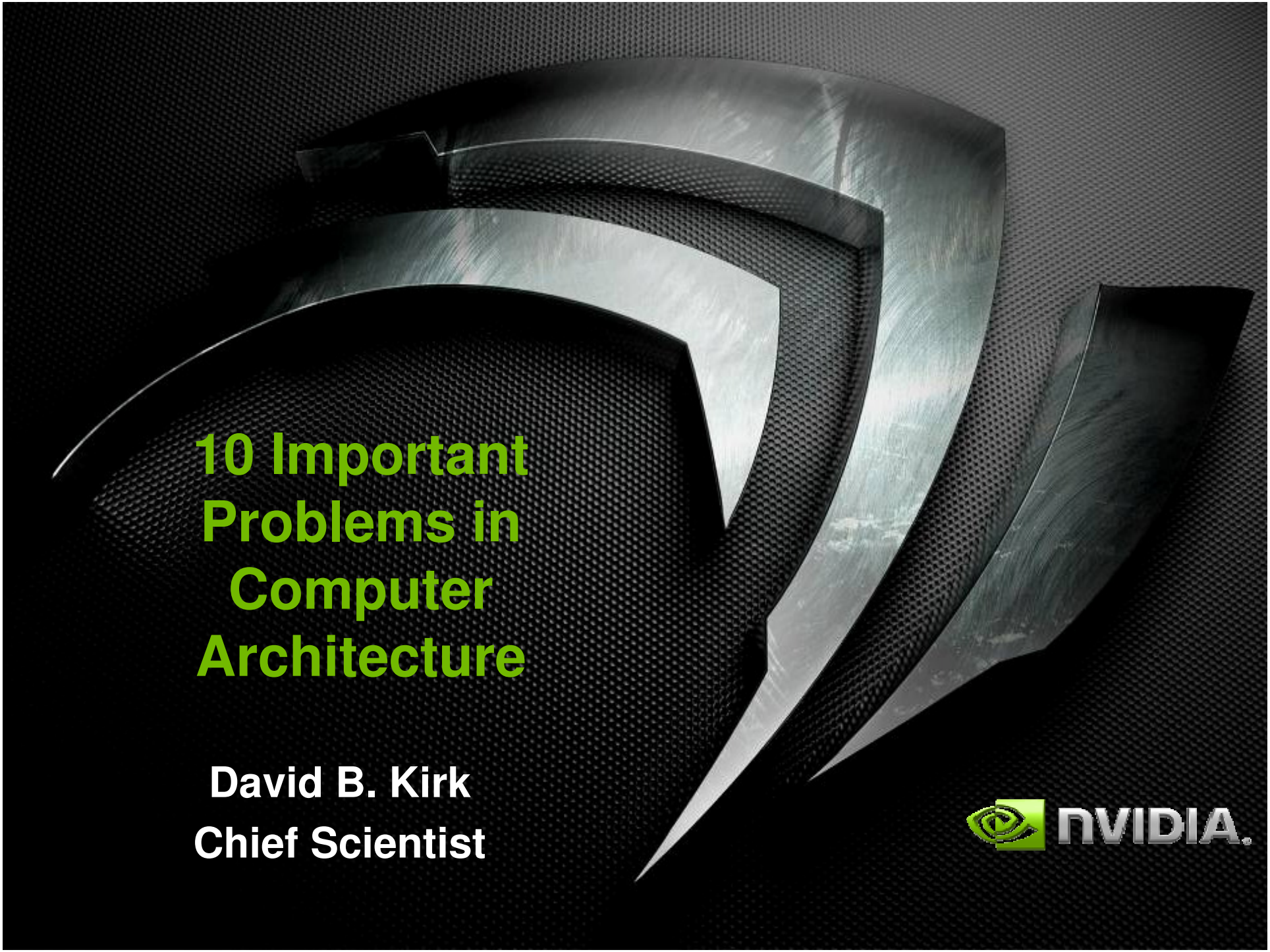

10 Important Problems in Computer Architecture
David B. Kirk
Chief Scientist
NVIDIA.

“Perfect Storm” Topic



- Potential to offend EVERYONE
  - If I don’t mention your pet project

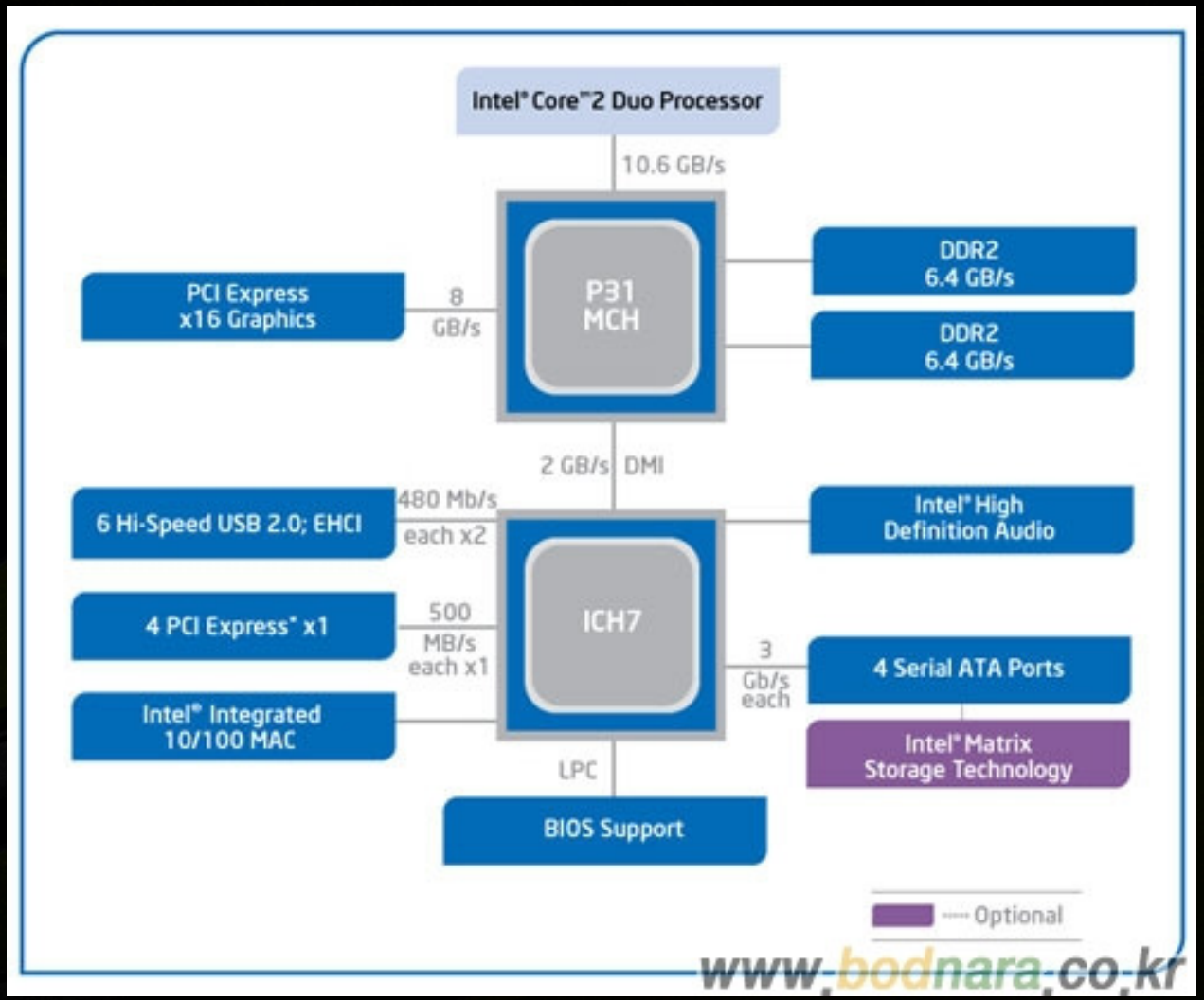
  or
  - If I do
- Potential to benefit EVERYONE
  - If I don’t mention your pet project

  or
  - If I do

Don’t take it personally!

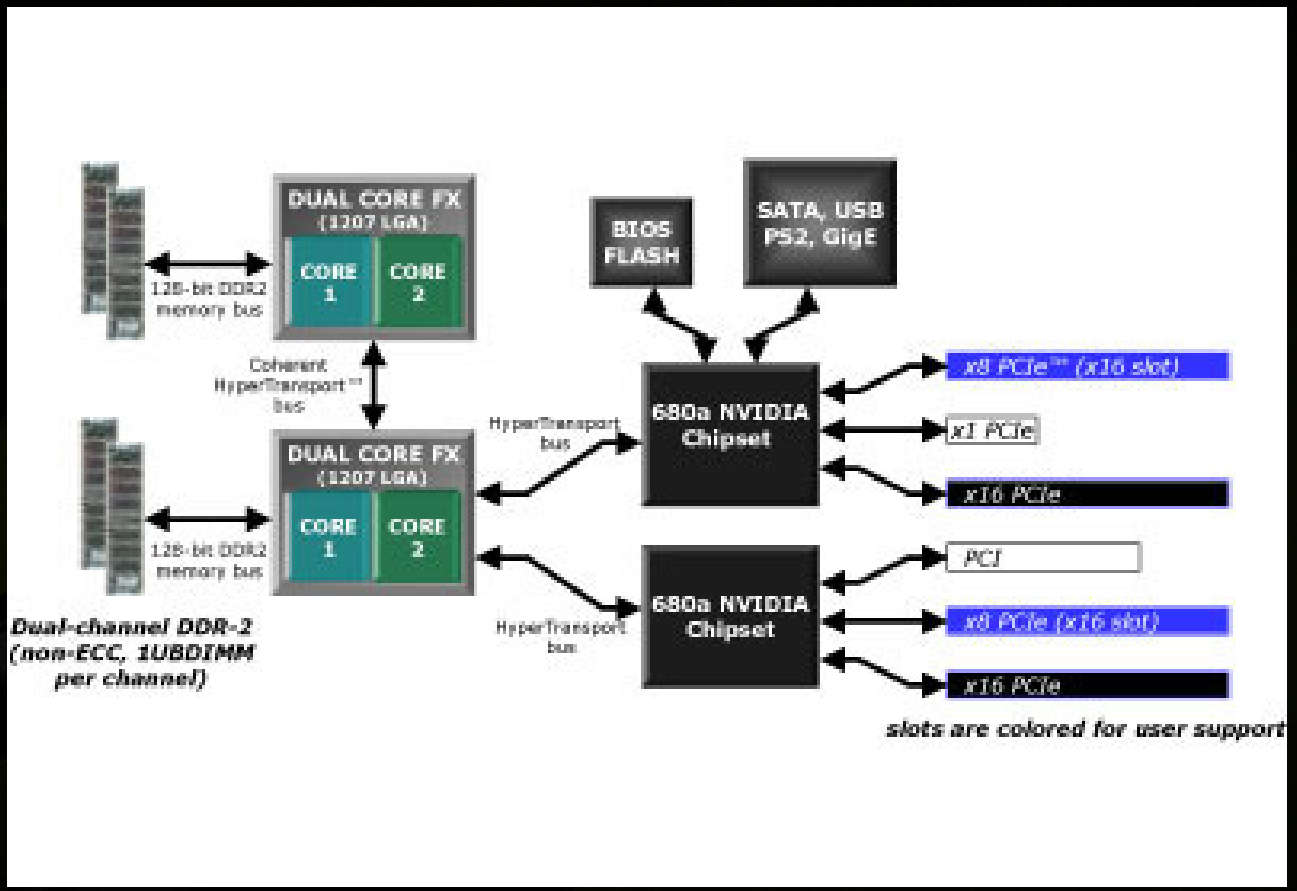
© NVIDIA Corporation 2008

Computer Block Diagram?



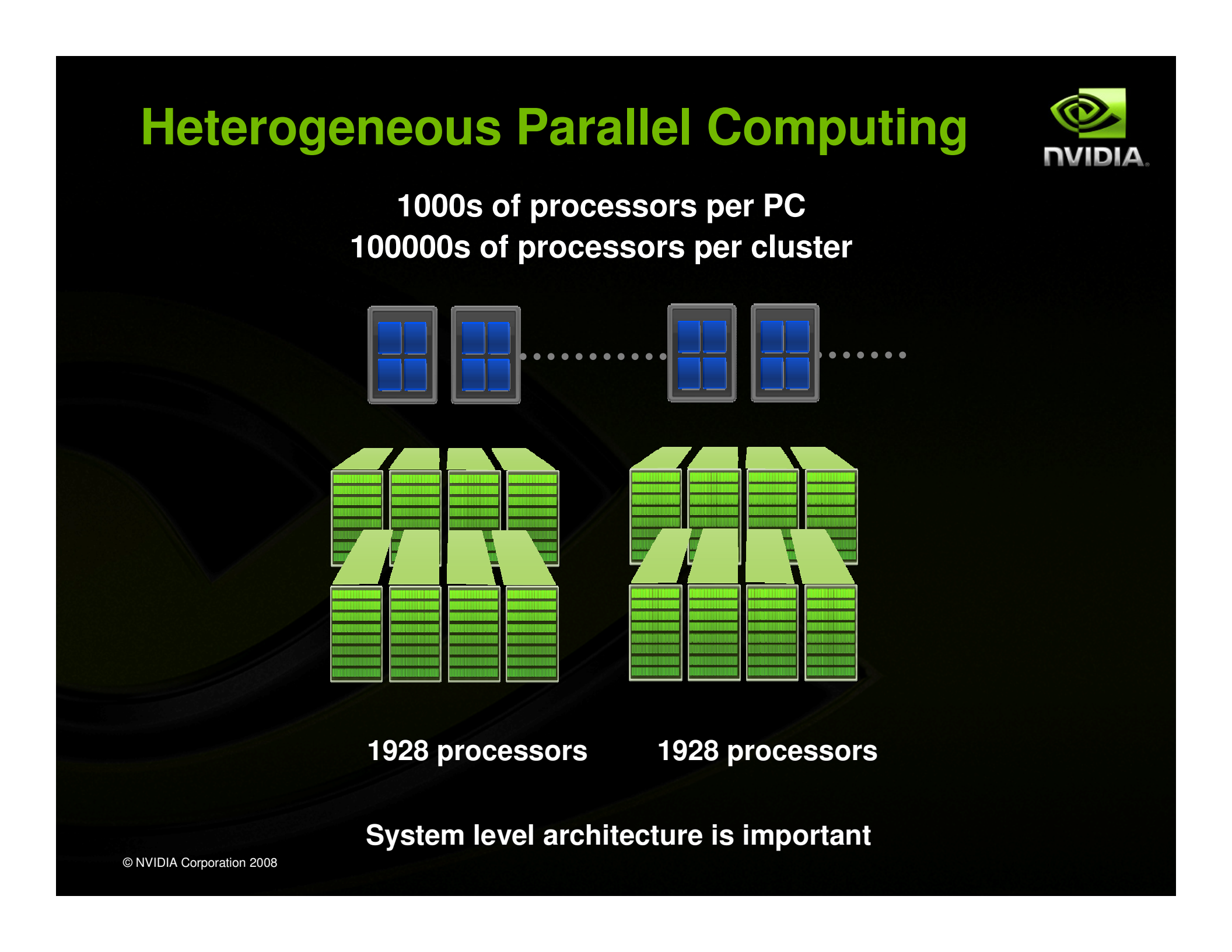
© NVIDIA Corporation 2008

Better…



© NVIDIA Corporation 2008

Heterogeneous Parallel Computing



1000s of processors per PC
100000s of processors per cluster

1928 processors 1928 processors

System level architecture is important

© NVIDIA Corporation 2008

Systems of Systems



© NVIDIA Corporation 2008

Hardware is Software (and vice versa)



- HW challenges and failures persist
  - Show up as SW issues
- HW successes and achievements disappear
  - No longer SW problems
- SW problems inspire new HW efforts
- SW solutions mask HW shortcomings

Talk about HW and SW together

© NVIDIA Corporation 2008

Reliable SW



- SW systems have poor reliability
- Bigger systems, worse reliability
  - Example: OS
    - HUGE system – many millions of lines of code
    - Pretty amazing it works at all
    - How many bugs left?
- Solutions?
  - Verifiability
  - Provability
  - HW support

© NVIDIA Corporation 2008

Reliable HW



- Historical: RAS efforts (Reliability / Availability / Serviceability)
- Fully redundant HW
  - No single point of failure, low overall MTBF
- Error / fault
  - Detection / Correction / recovery
- Dynamic reconfiguration
- No crashes. Ever. Really!
  - Soft resets

Problem: $$$$...

© NVIDIA Corporation 2008

Parallel Programming



- Express real-world problems
  - Mix of Serial / Parallel components
  - 90 / 10 rule (most programmers not very skilled)
- CUDA is a good step, but it isn't the end-all
  - Bet you didn't think I'd say that ☺
- Future opportunities
  - Problem decomposition (automatic?)
  - Higher level programming model / better tools
  - Locality / Communication
    - Pointers – (don't) pass "&x" around
  - Event management
    - Signaling / Exceptions

© NVIDIA Corporation 2008

Memory



- Memory wall: BW
  - How do we keep scaling memory BWs?
- Memory wall: Power
  - High speed interconnects draw lots of power

- Memory size
  - Good: grows with Moore's Law
  - Bad: not even remotely keeping up with data size

- Recurring theme: locality, communication

© NVIDIA Corporation 2008

Locality: Eliminate / Respect Space-time Constraints



- Programming models naïve WRT physics
  - Not “game physics”
  - Space-time physics
- With 1 processor, locality matters… a little
  - Cache hits
- With 2+ processors, locality matters… more
  - Pollution, migration, sharing, synchronization
- With many processors, locality matters… more than anything
  - Time & distance
  - No computing, only waiting

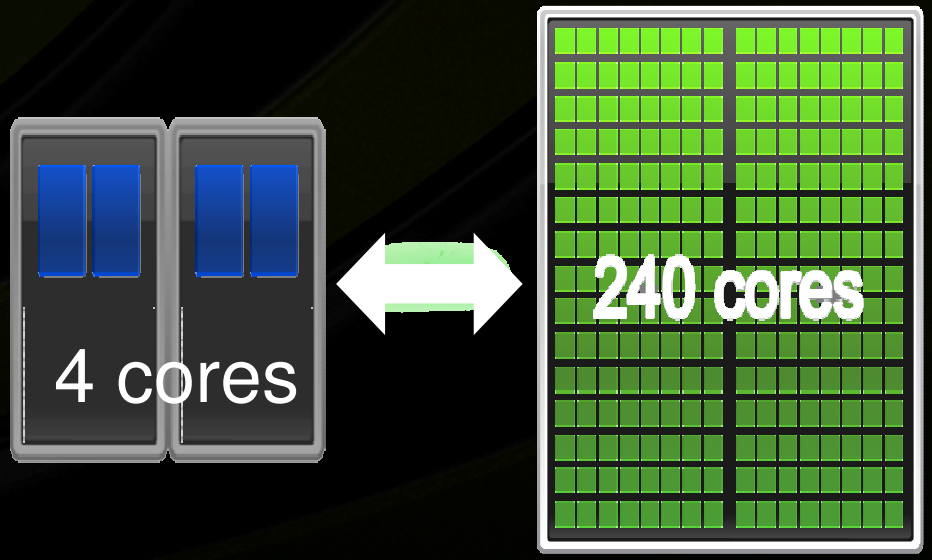
© NVIDIA Corporation 2008

Application Software
Industry Standard C Language

Numerics Engine

cuFFT | cuBLAS | cuDPP

System	CUDA Compiler	CUDA Tools
PCI-E Switch	C Fortran	Debugger Profiler



© NVIDIA Corporation 2008

Threading: MIMD, SIMD, and SIMT (oh my!)



- MIMD (or, in the limit, serial SISD)
  - Easy to understand / build
  - Ultimately limited flexibility & scalability
- SIMD
  - Still pretty simple to understand
  - HW Efficient, but constrained SW / tools
- SIMT
  - Harder to understand (and, to explain!)
  - Very Powerful model
    - That's why 100x speedups are possible
  - Needs growth and development

© NVIDIA Corporation 2008

GPU Computing Key Concepts

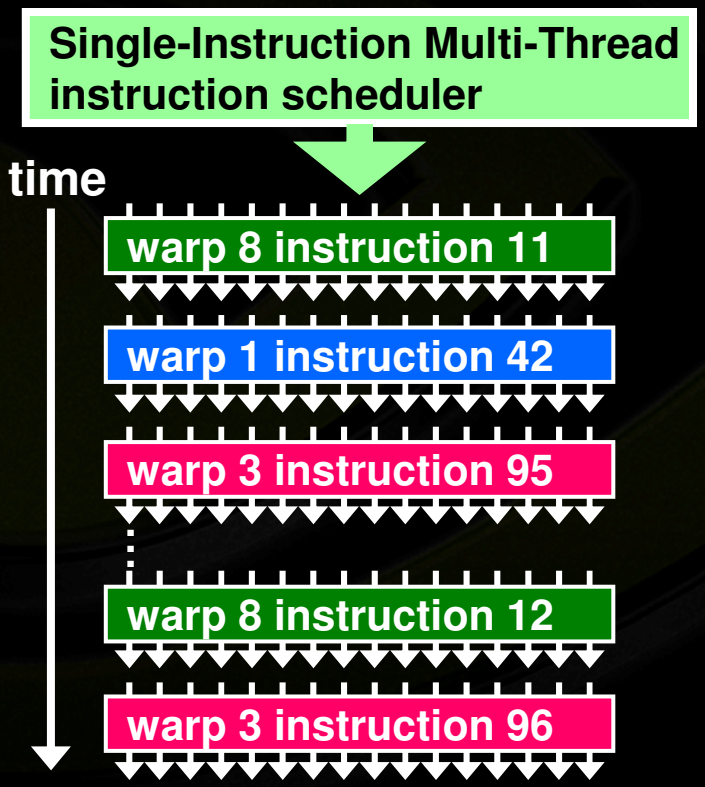


- Hardware (HW) thread management
  - HW thread launch and monitoring
  - HW thread switching
  - Tens of thousands of lightweight, concurrent threads
  - Real threads: PC, private registers, …
- SIMT execution model
- Multiple memory scopes
  - Per-thread private memory
  - Per thread-block shared memory
  - Global memory
- Using threads to hide memory latency
- Coarse grain thread synchronization

© NVIDIA Corporation 2008

SIMT Multithreaded Execution





- **SIMT: Single-Instruction Multi-Thread executes one instruction across many independent threads**
  - **Warp: a set of 32 parallel threads that execute a SIMT instruction**
  - **SIMT provides easy single-thread scalar programming with SIMD efficiency**
- **Hardware implements zero-overhead warp and thread scheduling**
- **SIMT threads can execute independently**
  - **SIMT warp diverges and converges when threads branch independently**
  - **Best efficiency and performance when threads of a warp execute together**

© NVIDIA Corporation 2008

Secure Computing

- Elimination of identity theft
- Elimination of spam, bots…



- Intellectual Property protection

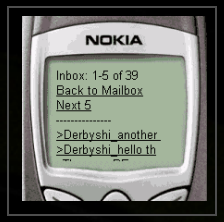
- Privacy

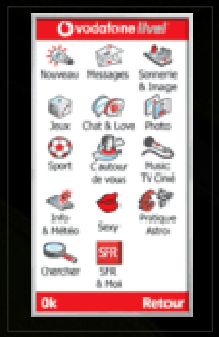


- Safety

© NVIDIA Corporation 2008

Compelling User Interface









© NVIDIA Corporation 2008

Immersive, High Fidelity Displays



High Dynamic Range



Portable

Home / Office



© NVIDIA Corporation 2008

True 3D Interfaces

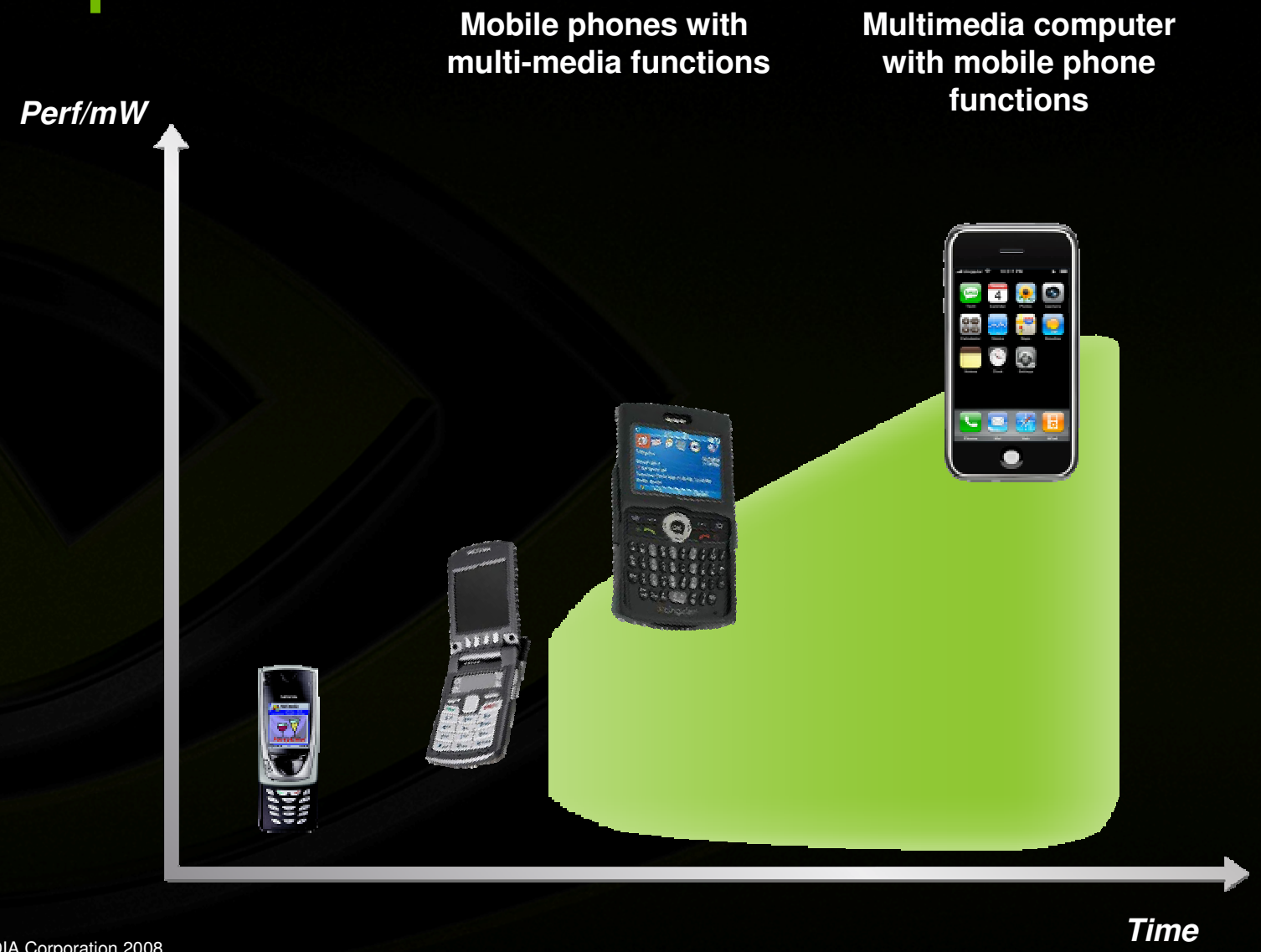


- 3D immersive visual interfaces for computers
- Vista / MacOS still 2D
- What does real 3D look like?
- How can we better interface computers to people?

- Why doesn't this topic ever go away?
  - Very little progress
  - Computers and devices still hard to use
  - We are 3D / visual beings
    - Our world is 3D
    - We know how to use it

© NVIDIA Corporation 2008

Mobile Devices become “real computers”



© NVIDIA Corporation 2008

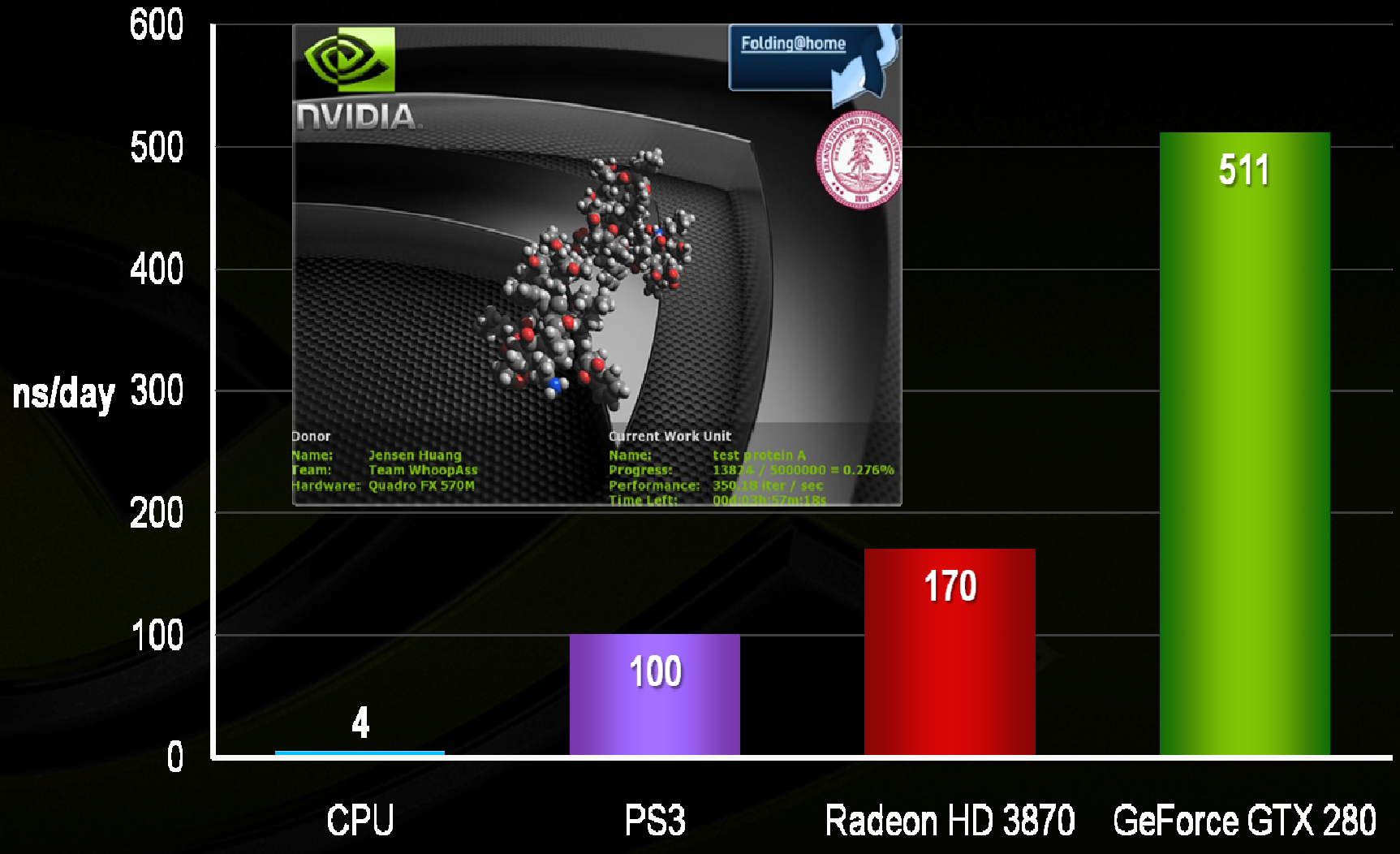
Extensible Distributed Computing



- Distributed devices
  - PC, Laptop, Media Server, TV, Phone, iPod, etc
- GPUs greatly speed up applications
  - Can we do the same for data center speedups?
  - Across the internet?
- How to use vast numbers of computers across the internet in a general purpose way?
  - Kind of a universal operation systems question
  - F@H is a baby step in this direction

© NVIDIA Corporation 2008

Folding@home on GeForce



© NVIDIA Corporation 2008

Interconnect



- Between processors / memory on a chip
- Between chips in a system
- Between systems in a cluster
- Between clusters in a grid
- Between ubiquitous, heterogeneous devices

- Wired, wireless …

- How to make scalable & extensible

© NVIDIA Corporation 2008

Power



- The multi-core panacea
  - Doesn't solve problem, only buys time
- Power-aware design and architecture
- Power-aggressive design and operation

- # of processors will rise exponentially
  - So will power, if nothing is done to stop it
  - Energy efficiency, or Ops/W



© NVIDIA Corporation 2008

That was 11 or 12



- But, who's counting!

© NVIDIA Corporation 2008


Questions?
dk@nvidia.com
http://www.nvidia.com/CUDA
http://www.nvidia.com/TESLA
NVIDIA.