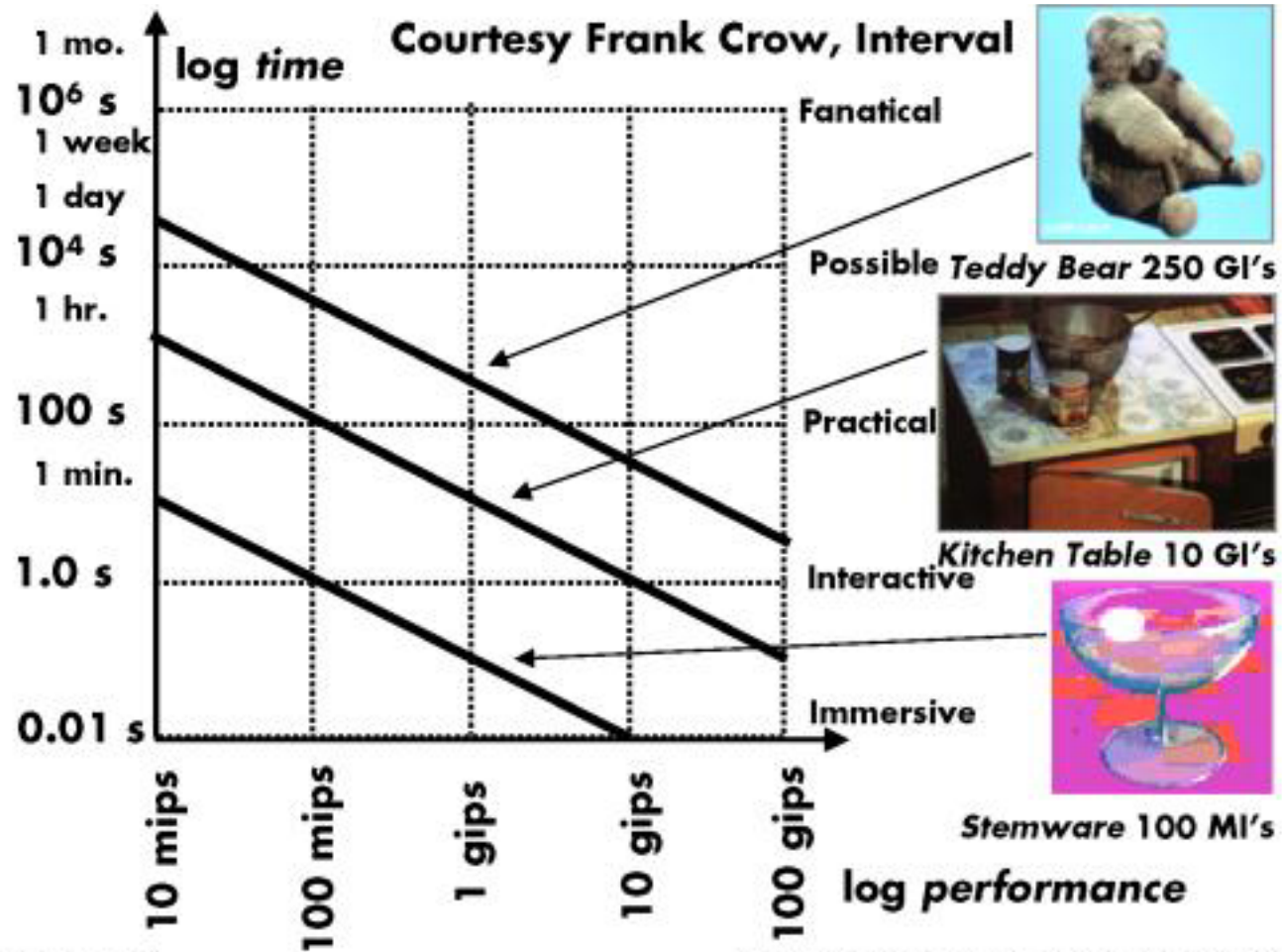

The Future: What's Next for GPUs?



John Owens

Department of Electrical and Computer Engineering
Institute for Data Analysis and Visualization
University of California, Davis

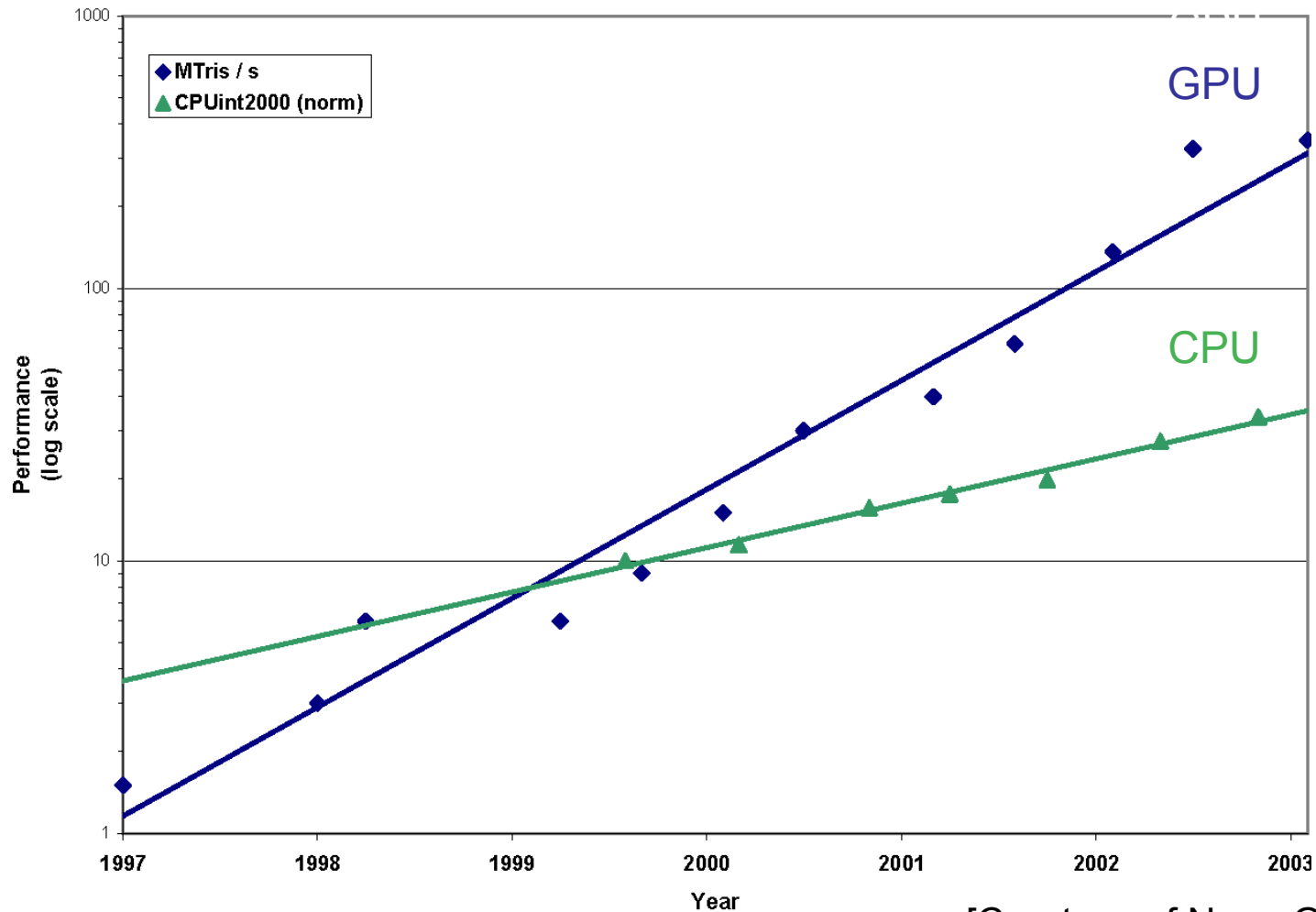
Off-line to on-line to real-time ...



[Courtesy of Crow/Hanrahan/Akeley]



Motivation: Computational Power



[Courtesy of Naga Govindaraju]



Semiconductor Scaling Rates

- From *Digital Systems Engineering*, WJ Dally and JW Poulton

Parameter	Current Value	Yearly Factor	Years to Double (Half)
Moore's Law (grids on a die)**	1 B	1.49	1.75
Gate Delay	150 ps	0.87	(5)
Capability (grids / gate delay)		1.71	1.3
Pins per package	750	1.11	7
Aggregate off-chip bandwidth		1.28	3
DRAM latency [courtesy Junji Ogawa]		1.07	10

** Ignores multi-layer metal, 8 layers in 2001

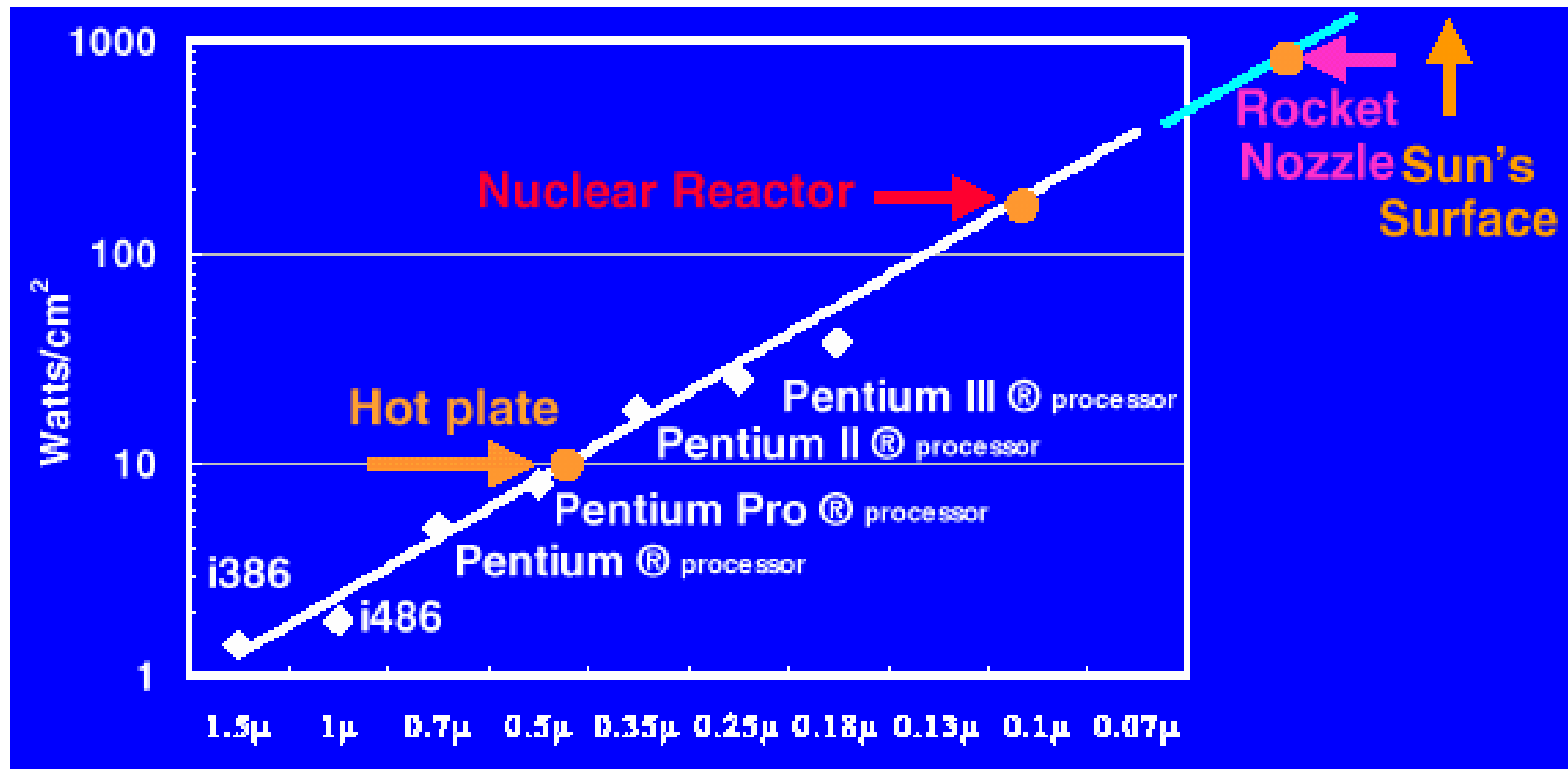


Hardware Considerations

- “Memory wall”
 - Caching and recomputation vs. communication
 - Continued migration of functionality onto GPU
 - Higher-level graphics functionality
 - Physics & simulation
 - GPGPU?
- Size of design teams
 - Intel design teams increase in size 40% / generation
 - Validation for increasingly complex designs
- Power ...



Power Considerations



[Courtesy Bob Colwell]

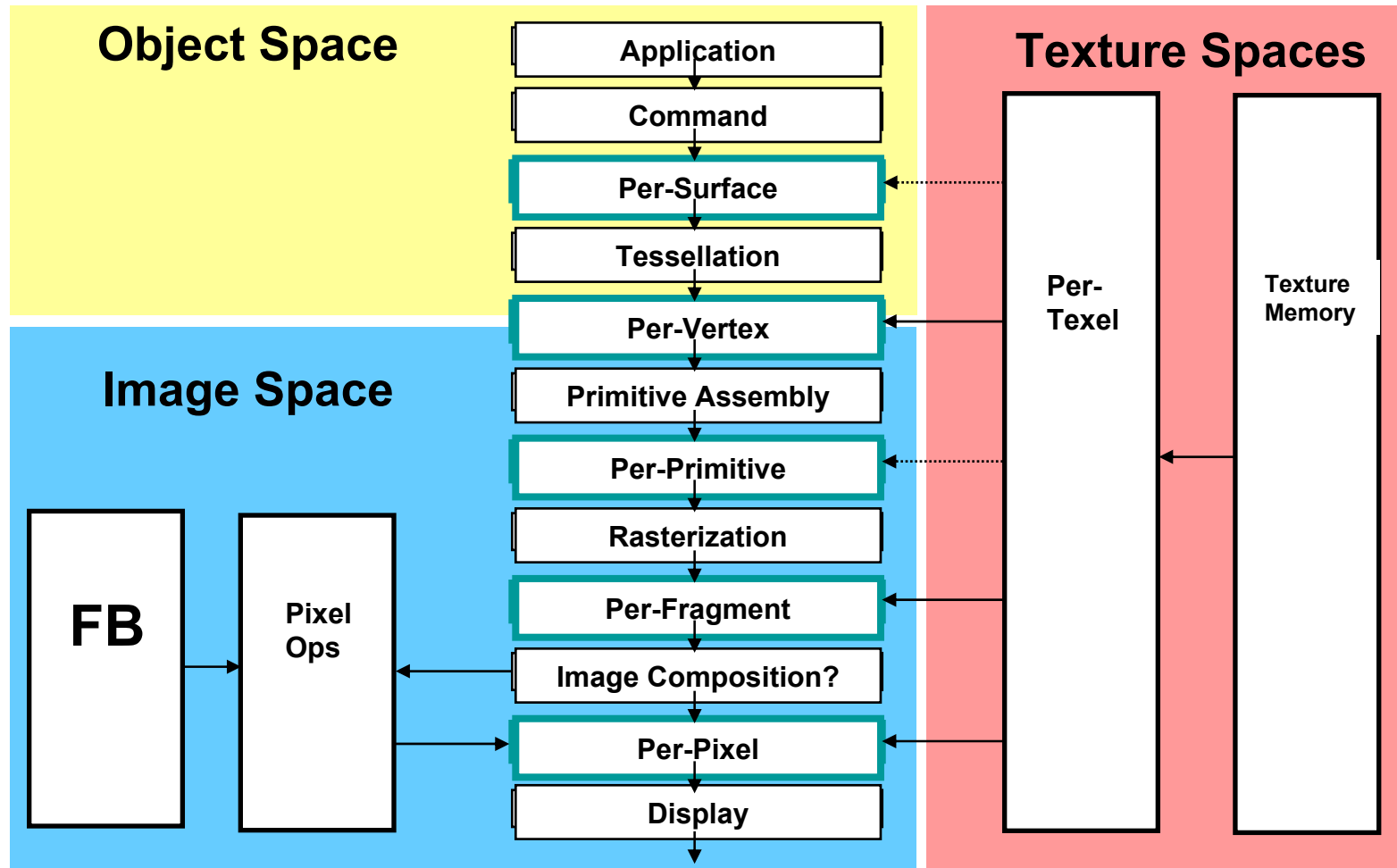


Architecture/Microarchitecture

- Current programming model:
 - MIMD or SIMD for vertex processing?
 - SIMD for fragment processing?
- Can we share units between the stages?
- To what will the instruction sets converge?
- Are these the only stages that will be programmable?
- How will the CPU interact with the GPU?
- How can we extend to multiple GPUs and multiple CPUs?



Generalized Graphics Pipeline?

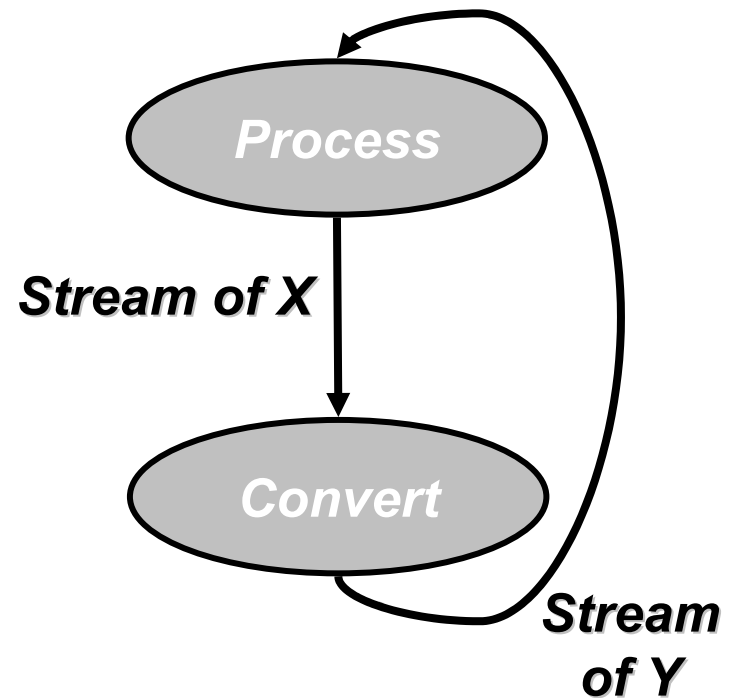


[From Akeley and Hanrahan, Real-Time Graphics Architectures]



Future GPUs?

- Programmable stages operate on primitives (“process”)
 - Fragment, vertex programs
- Hardwired or programmable stages “convert” one kind of primitive to another
 - Rasterization, composite
- Could define own pipelines!
 - Reyes, raytracing ...



GPGPU Algorithms

- Much to be done!
 - New/optimized stream algorithms
 - New features of graphics hardware
- Move from kernels to applications
 - Scientific computation
 - Simulation (game physics?) + visualization
 - What will be first “killer app” on GPUs?
- Ask for new features ...
- ... but don't lose what gives the GPU high performance



Tools and Programming Models

- CPU programmers have it easy!
 - Straightforward programming model
 - Many languages
 - Great compilers
 - Debuggers
 - Profiling and performance tools
 - Multi-CPU libraries and applications
- GPU: Long way to go
 - Vendors working hard to provide these (but targeted primarily at games)
 - Active academic research
 - Brook is a great start, but domain specific languages and other design philosophies are needed
 - People who need these tools should help design them!



What *should* we map to GPUs?

- Problems with *high compute requirements*
- Problems with *regular structure*
- Problems with *predictable communication needs*
- Problems that require *interaction with the graphics system*

- Other domains: biology, statistics, chemistry, finance ...

- Enormous opportunity at frontiers of applications, software, and hardware!



For more information ...

- Course web page:
<http://www.gpgpu.org/vis2004/>
- GPGPU home: <http://www.gpgpu.org/>
 - Mark Harris, UNC/NVIDIA
 - Research, forums, developer tools, ...
- *GPU Gems* (Addison-Wesley)
 - Vol 1: 2004; Vol 2: 2005
- Conferences: Siggraph, Graphics Hardware, GP²
 - Course notes: Siggraph '04, IEEE Visualization '04

