# Current Research in Computer Architecture at UVA

Kevin Skadron,
University of Virginia,
Dept. of Computer Science

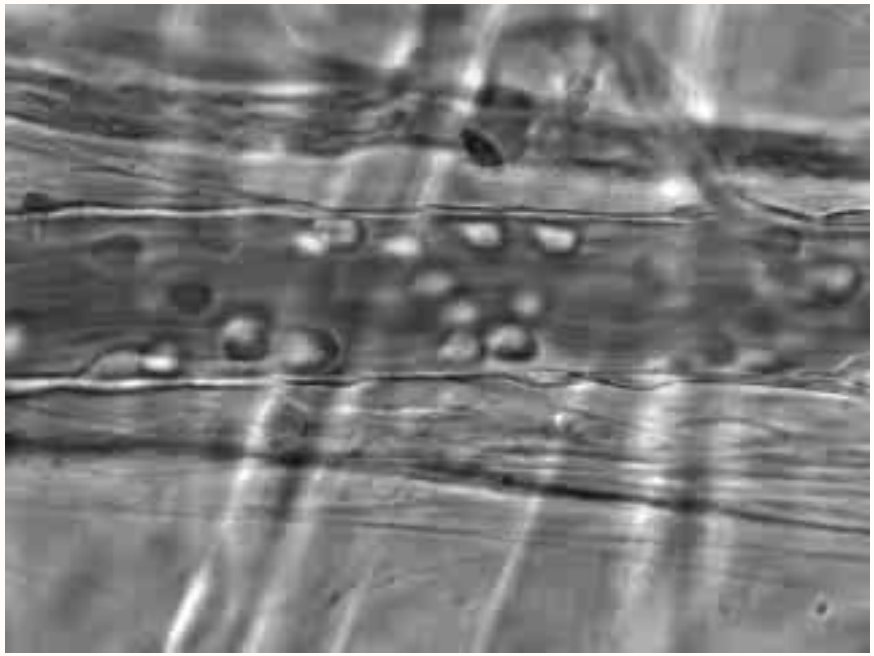
# A Few Topics from my Group

- GPUs for computational science
- Temperature-aware processor design

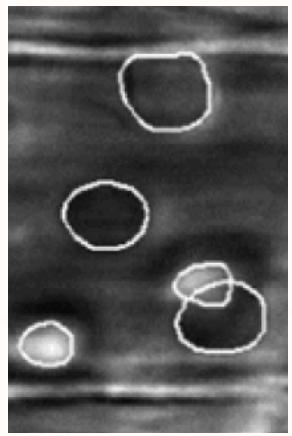
# Illustrative Case Study from Systems Biology

- Leukocyte detection and tracking in video microscopy
  - Understand inflammatory processes and treatment
  - Manual measurement is tedious and error-prone
  - 100X speedup possible even with discrete GPU, but…
  - Required non-trivial, architecture-aware reorganization
  - IPDPS'09

# Leukocyte Detection and Tracking



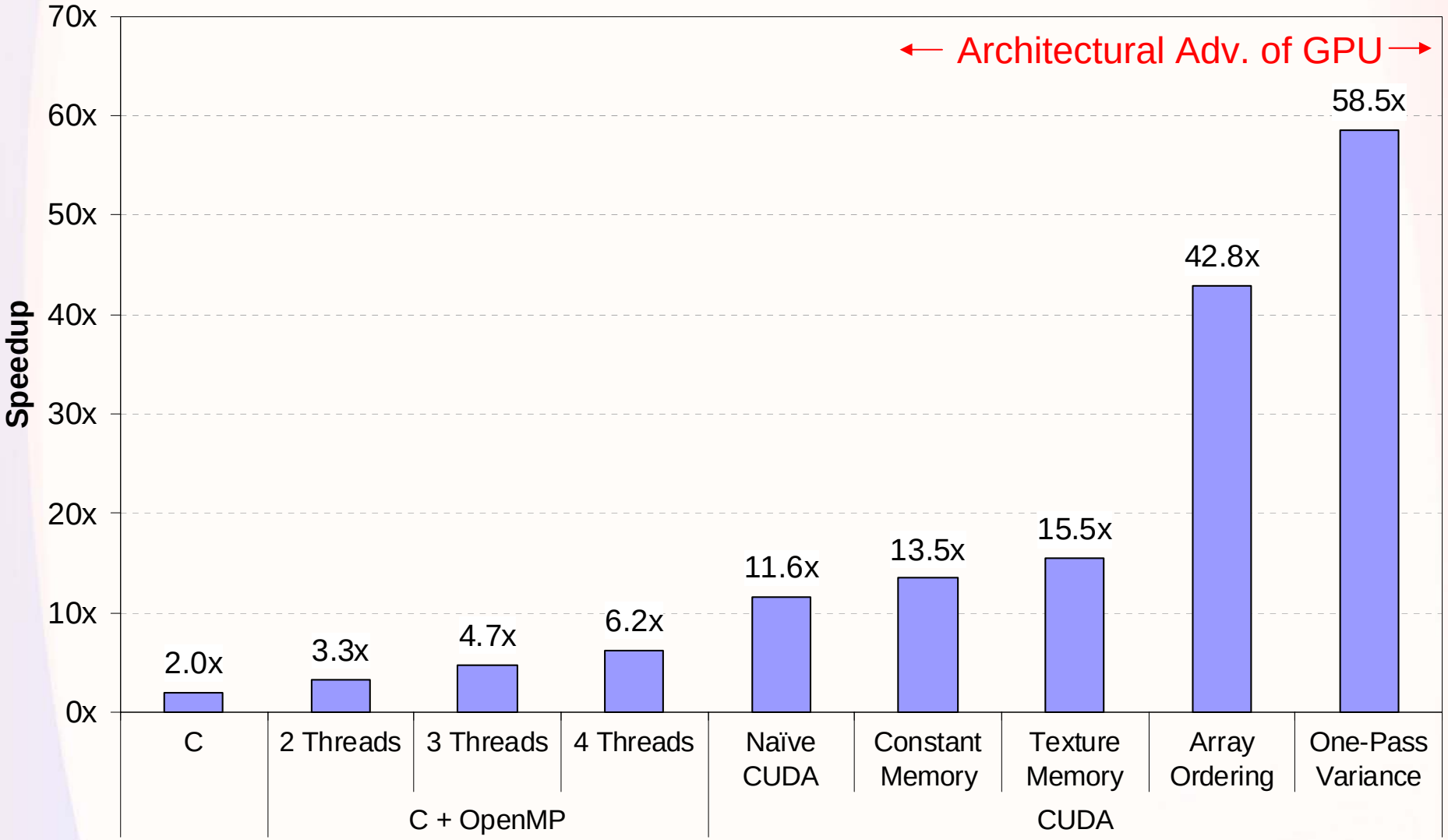
Zoomed:



- Video processing challenges:
  - Need to track velocity of rolling leukocytes
  - Leukocytes can be dark or light, overlap
  - Multiple layers of vessels
  - Jitter due to breathing of subject

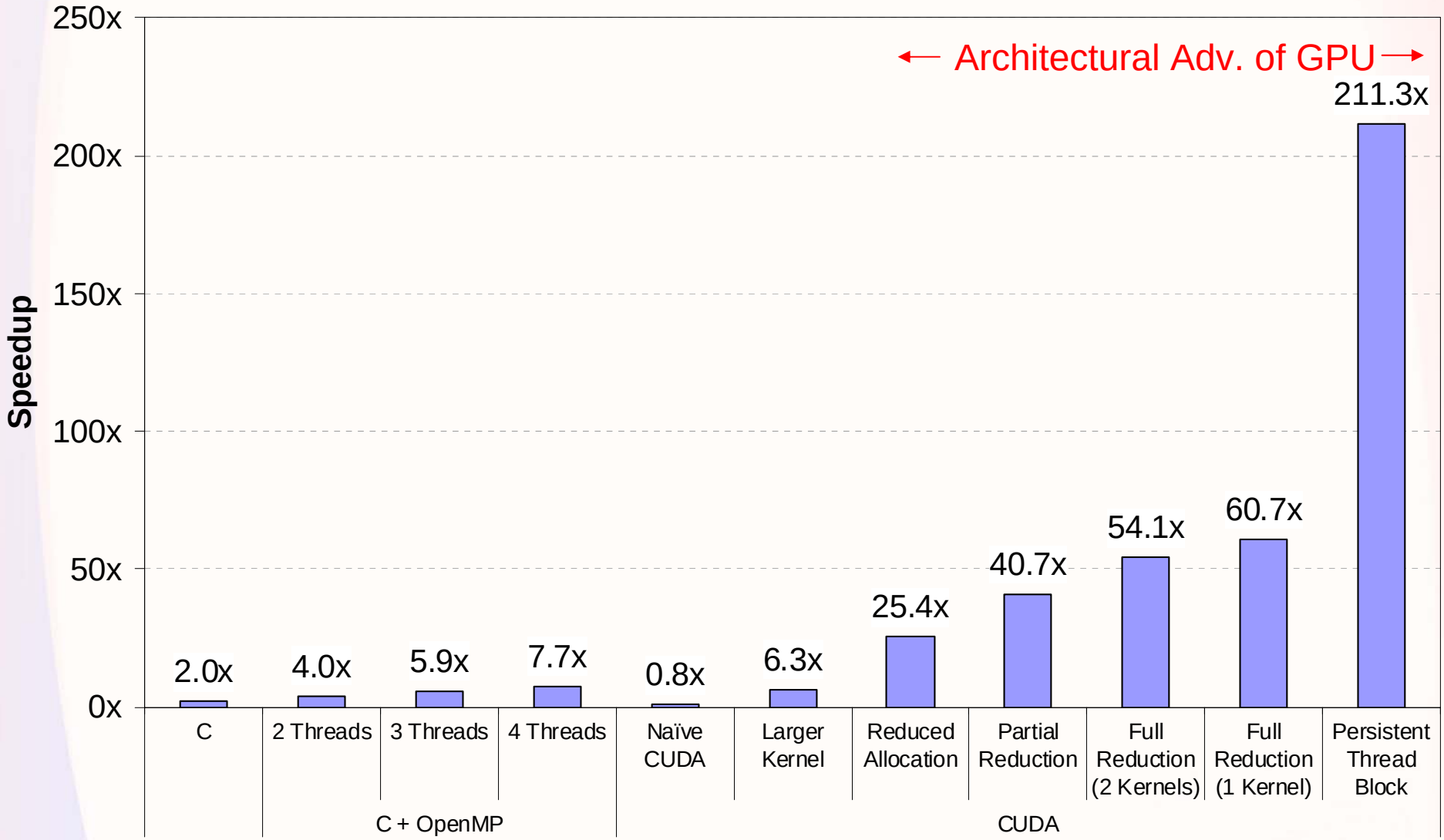
NVIDIA GTX 280

# Detection: CUDA Optimizations



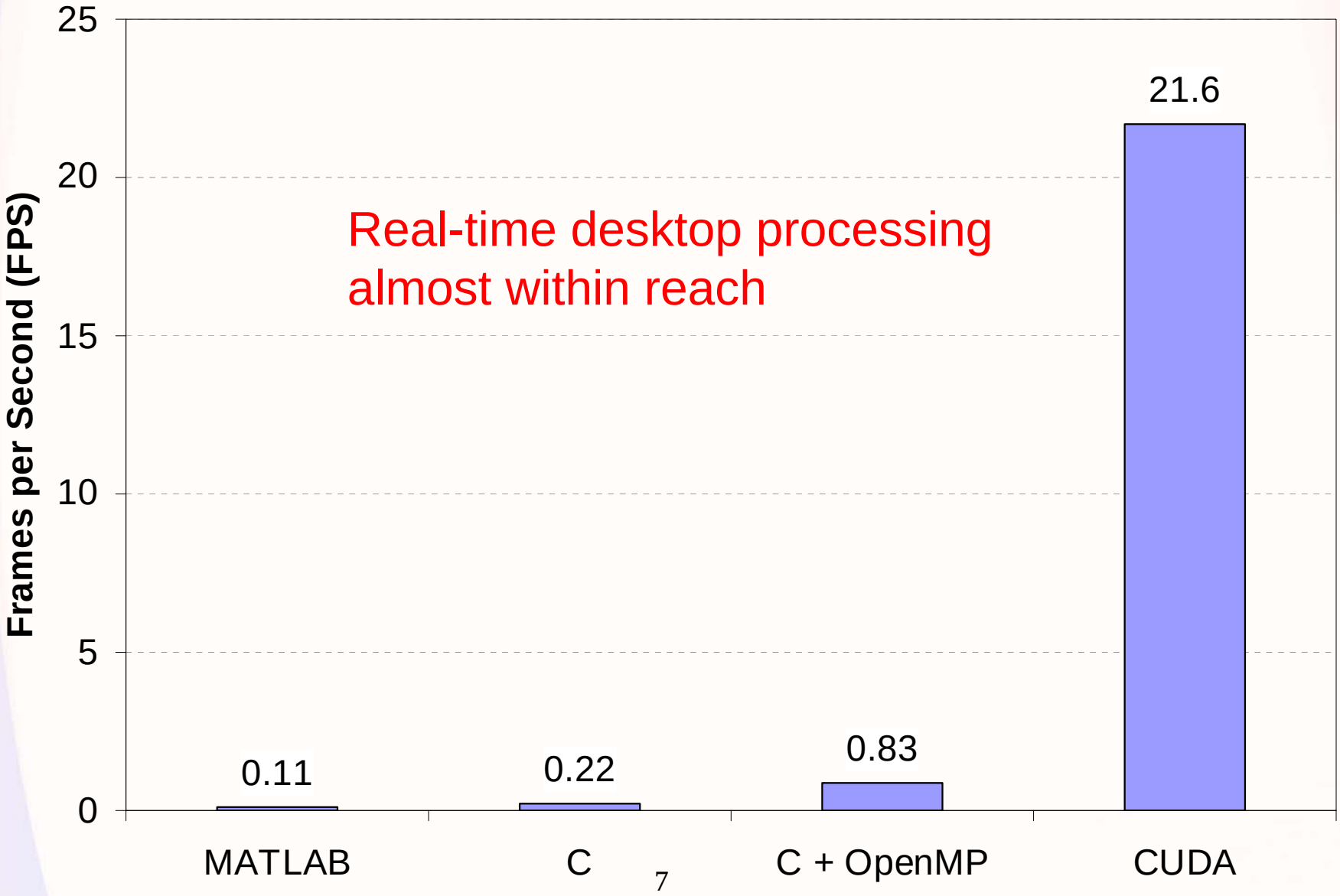
NVIDIA GTX 280

# Tracking: CUDA Optimizations

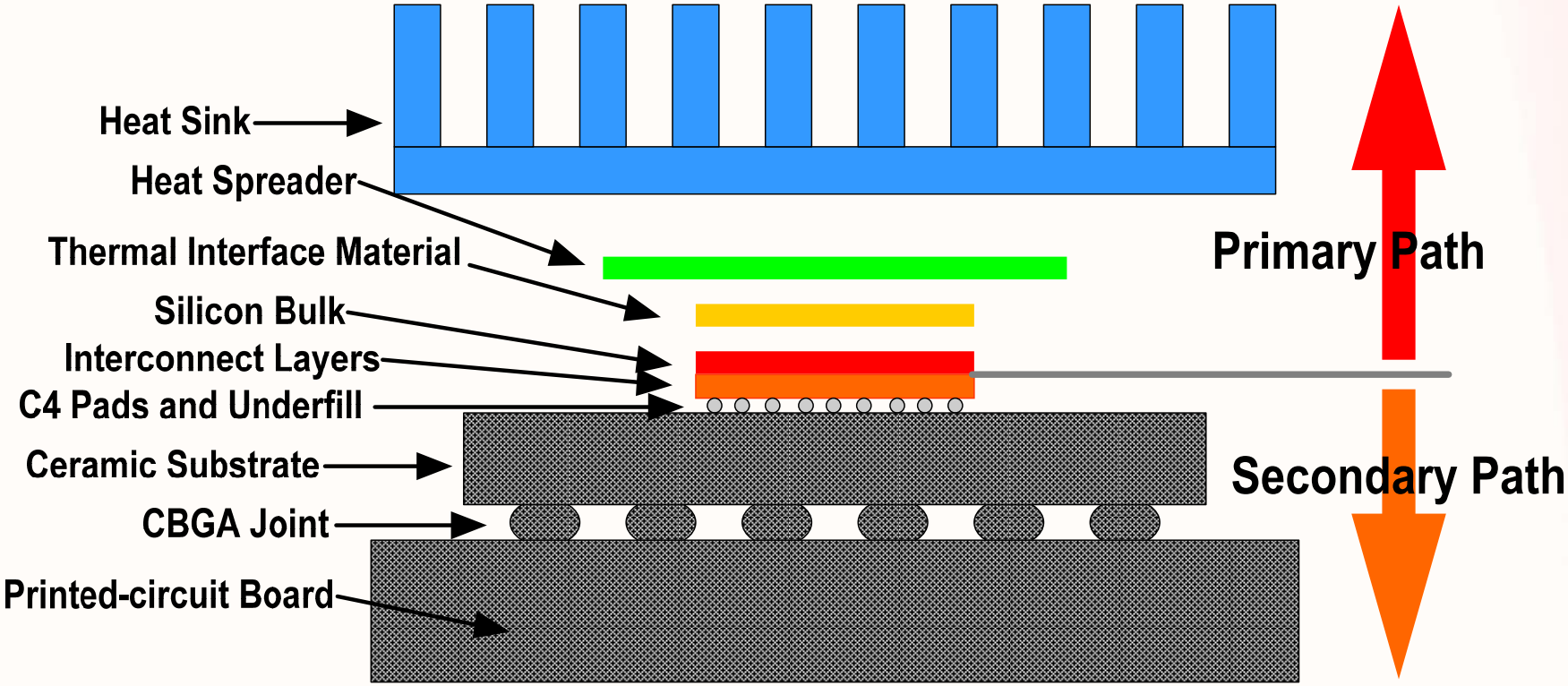


NVIDIA GTX 280

# Frame Rate Approaching Real Time



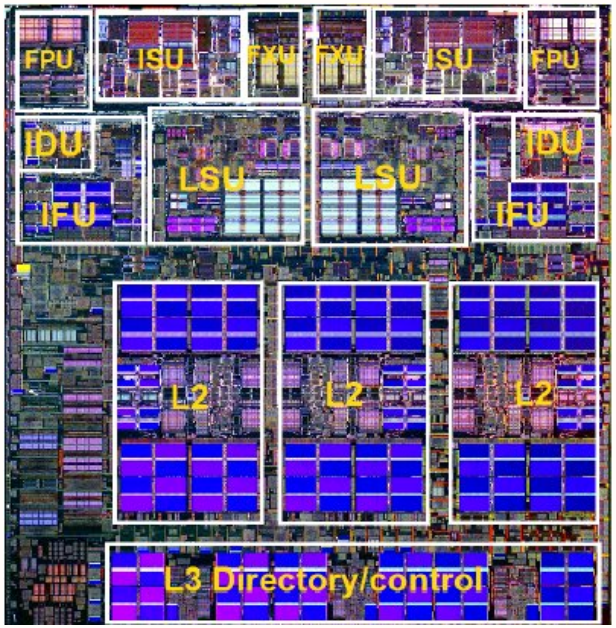
# Thermal Modeling

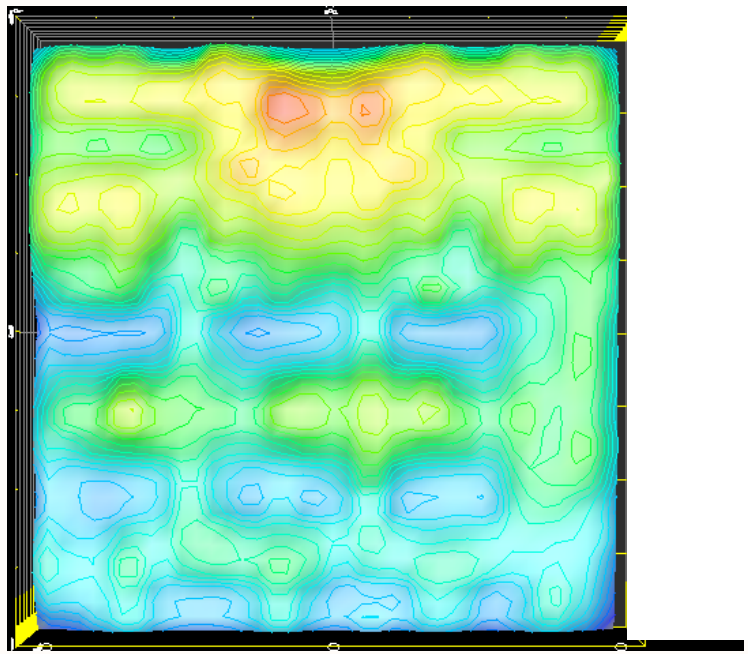



- Multiple layers
- Both silicon and package
- Primary and secondary paths
- Can add more layers for 3D chips

# Cooling Dictated by Hotspots

- High cooling capacity “wasted” on most of the chip’s area



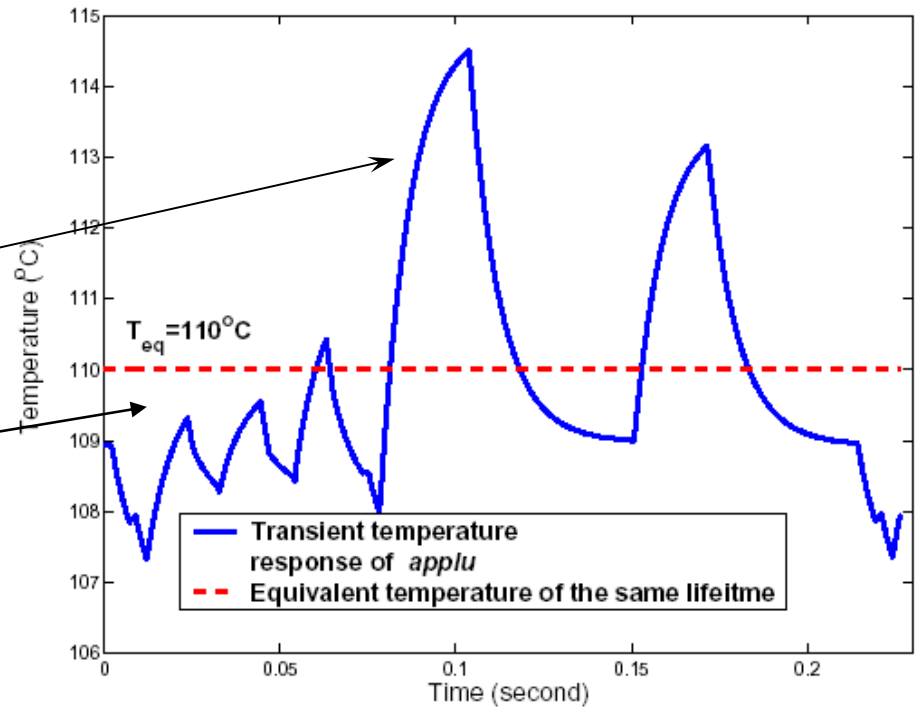
**IBM POWER5**

# Aging as f(T)

- Reliability criteria (e.g., DTM thresholds) are typically based on worst-case assumptions
- But actual behavior is often not worst case
- So aging occurs more slowly
- This means the DTM design is over-engineered!
- We can exploit this, e.g. for DTM or frequency (IEEE Micro 2005)

Spend

Bank



# The Visual Vulnerability Spectrum

Jeremy Sheaffer
University of Virginia
Department of Computer Science

# Motivation

- Transient errors can cause undesirable artifacts, such as:

# Motivation

- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors

# Motivation

- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors
  - Single texel errors

# Motivation

- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors
  - Single texel errors
    - Which might be stretched

# Motivation

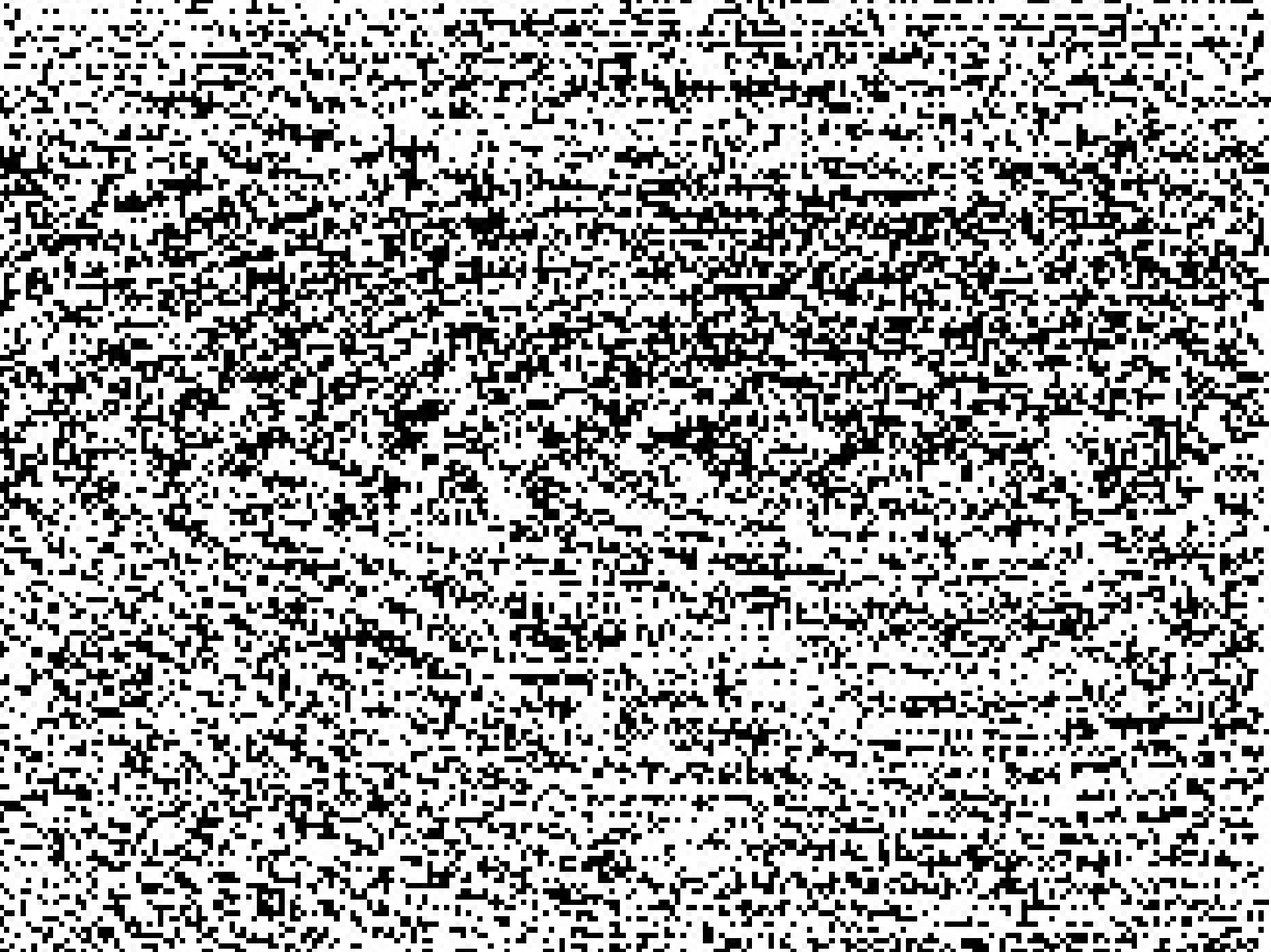
- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors
  - Single texel errors
    - Which might be stretched, interpolated

# Motivation

- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors
  - Single texel errors
    - Which might be stretched, interpolated, or repeated

# Motivation

- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors
  - Single texel errors
    - Which might be stretched, interpolated, or repeated
  - Single vertex errors

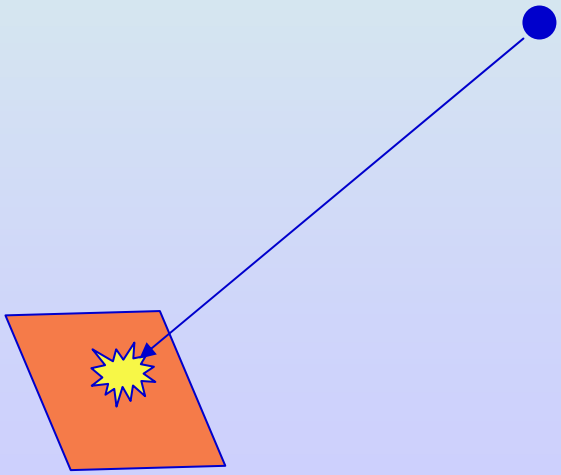


# Motivation

- Transient errors can cause undesirable artifacts, such as:
  - Single pixel errors
  - Single texel errors
    - Which might be stretched, interpolated, or repeated
  - Single vertex errors
  - Corrupt a frame
  - Crash the computer
  - Corrupt rendering state

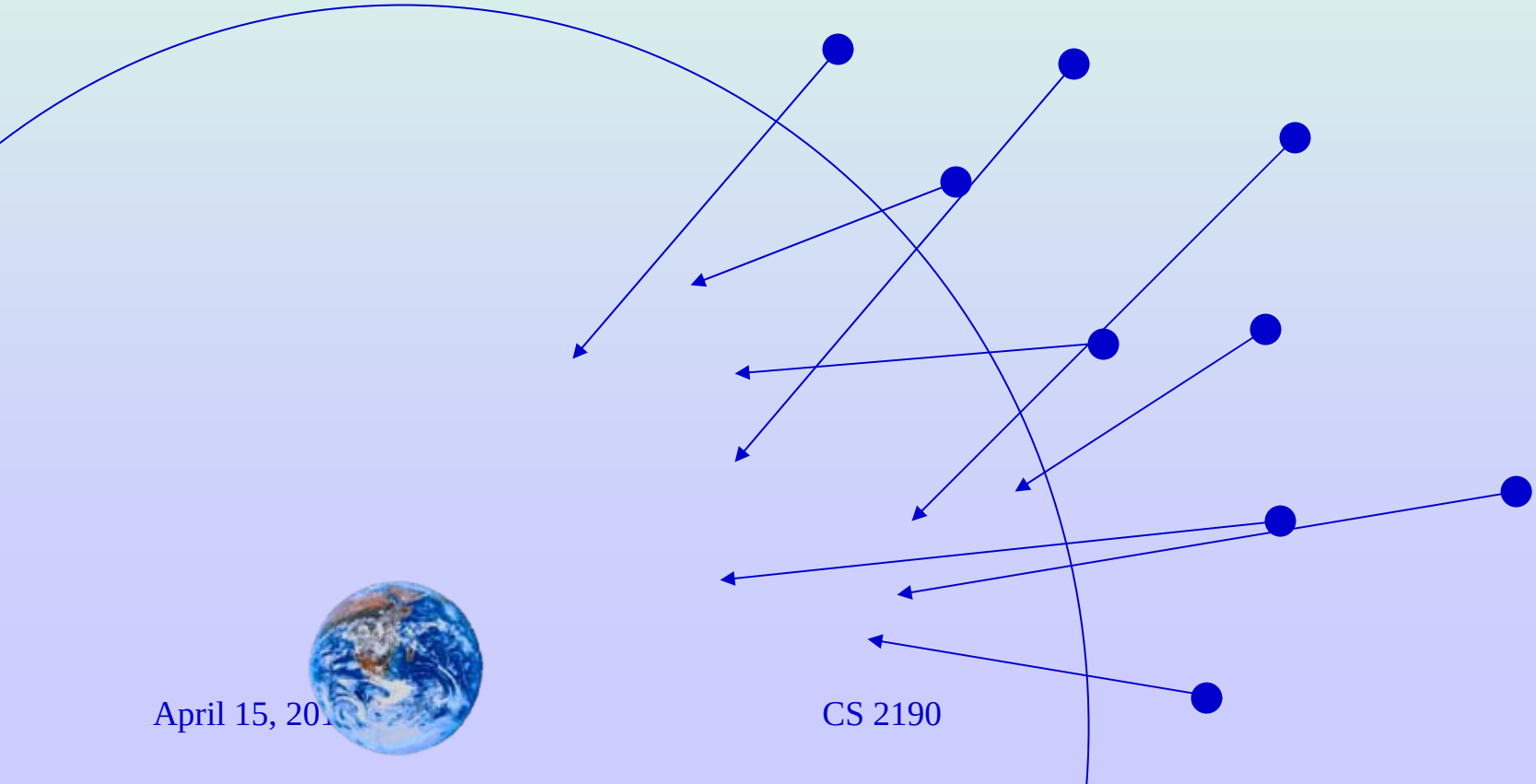
# Causes of Transient Faults

- Traditional causes
  - Cosmic radiation—gamma particles



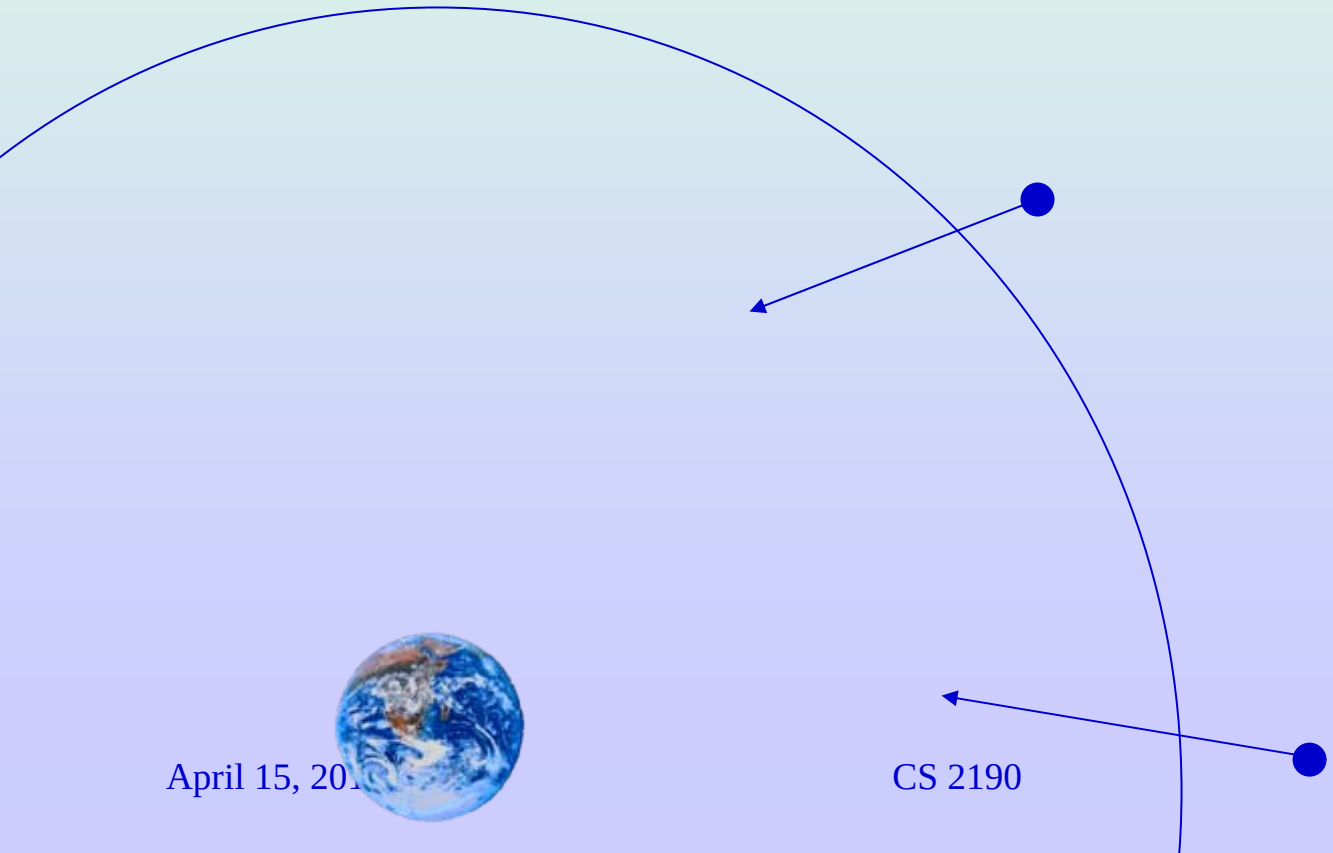
# Causes of Transient Faults

- Traditional causes
  - Cosmic radiation—gamma particles



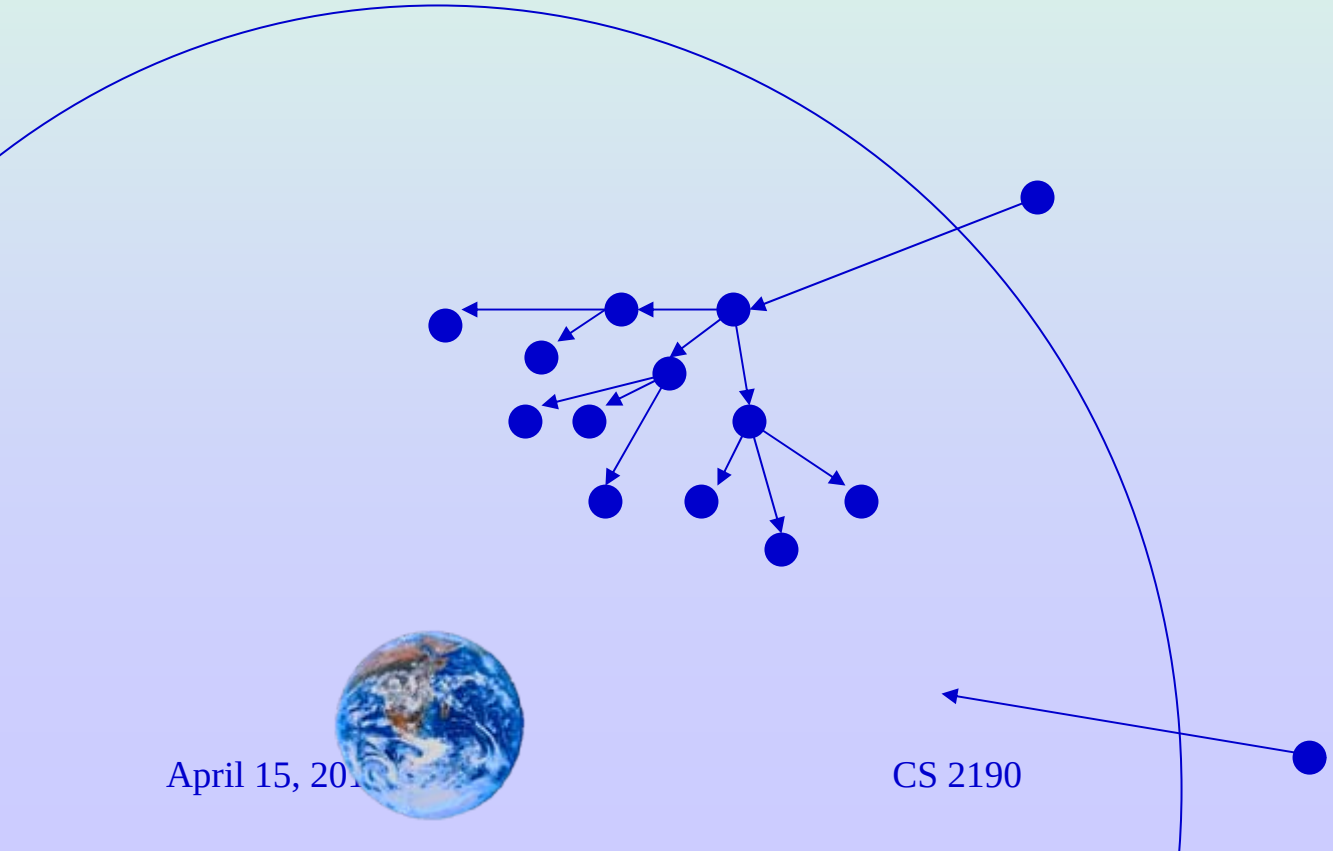
# Causes of Transient Faults

- Traditional causes
  - Cosmic radiation—gamma particles



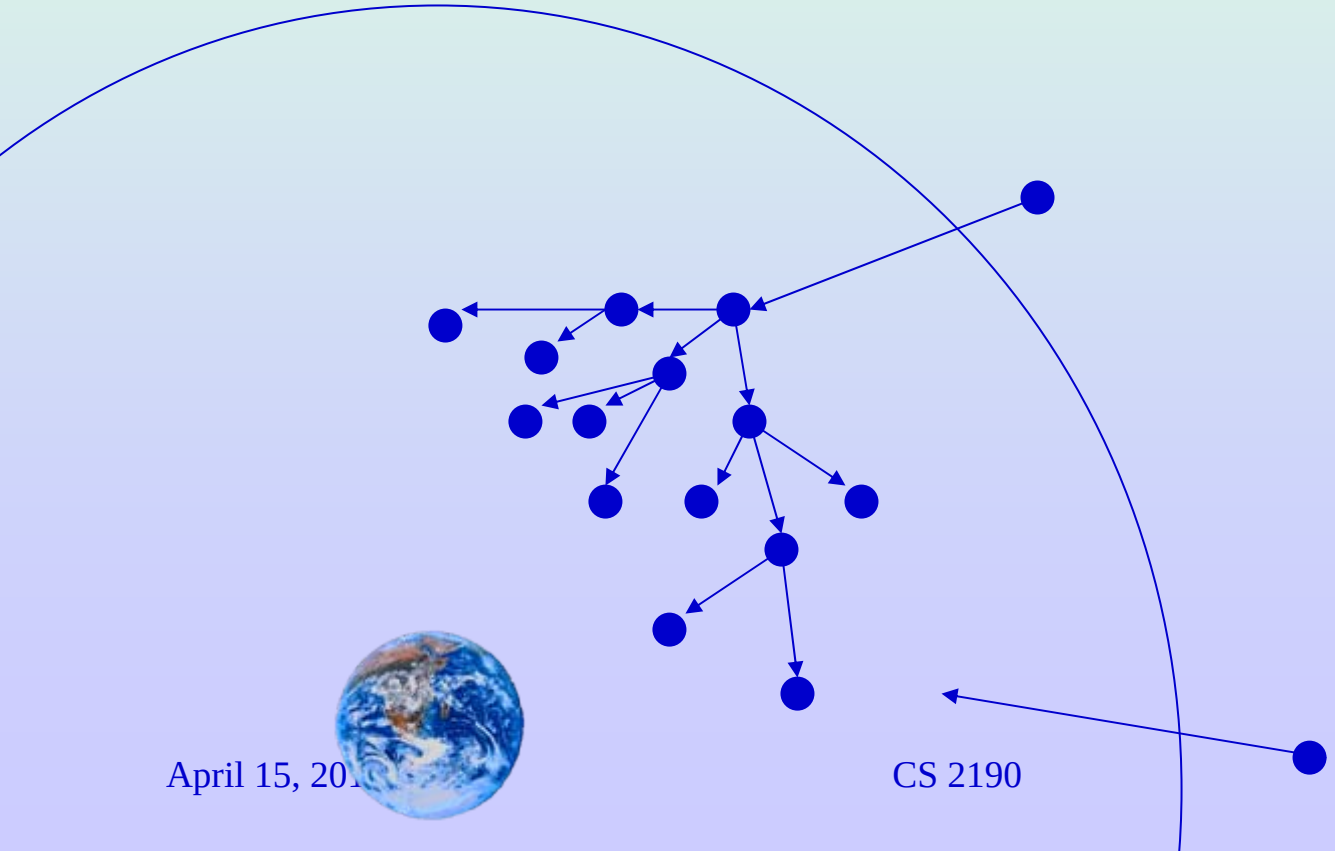
# Causes of Transient Faults

- Traditional causes
  - Cosmic radiation—gamma particles



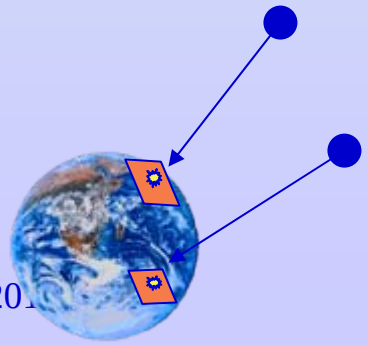
# Causes of Transient Faults

- Traditional causes
  - Cosmic radiation—gamma particles



# Causes of Transient Faults

- Traditional causes
  - Cosmic radiation—gamma particles



# Causes of Transient Faults

- Cosmic radiation—gamma particles
  - Soft Error Rate (SER) is proportional to cosmic ray flux
    - Flux at sea level is about 1 particle / $cm^2$ second
    - Maximum flux of about 100 particles / $cm^2$ second occurs at airplane altitudes
    - Particles at higher altitudes tend to have higher energies due to less cascading
- Terrestrial radiation—alpha particles
  - Initially discovered at nuclear test sites in the '50s
- Called *soft errors* or *single event upsets (SEUs)*

# Transient Fault Mitigation Techniques (CPU)

- ECC and parity
  - These protect memory but not combinational logic
    - Until recently, memory has been the primary concern and ECC and parity the primary solutions
- Scrubbing
  - Used in conjunction with ECC to reduce 2-bit errors
- Hardware fingerprinting or state dump with rollback
  - Poorly evaluated
- Larger or radiation-hardened gates
  - Increases the critical charge $Q_{crit}$
- Redundancy
  - Primarily employed to protect logic
  - Also sometimes used for memory

# Reliability Through Redundancy

- Primary topic in recent transient fault reliability literature
- Many clever ideas proposed and (sort of) evaluated, including
  - Triple redundancy with voting
    - Boeing 777 uses triple redundancy in all fly-by-wire components and triple redundancy in all computers for ‘triple triple’ or 9× redundancy
  - Lockstepped processors
  - Redundant Multithreading
    - CRT—Chip-level Redundantly Threaded processors
    - SRT—Simultaneous and Redundantly Threaded processors
      - The concepts of a ‘Sphere of Replication’ and leading and trailing threads
      - LVQs—Load value queue
      - BOQs—Branch outcome queue
      - Compiler assisted techniques like the checking store buffer (CSB)

# Architectural Vulnerability Factor

$$AVF = \frac{\sum_{b \in B} t_b}{|B| \times \Delta t}$$

# Architectural Vulnerability Factor

$$AVF = \frac{\sum_{b \in B} t_b}{|B| \times \Delta t}$$

- This is not applicable to graphics hardware

# AVF is Not Good for Graphics

- Primarily because it assumes that all bits that influence the computation are equally important
  - Similarly, with AVF, any corruption of an ACE bit gives a ‘wrong answer’

# Visual Vulnerability Spectrum

- We note that many transient faults in graphics workloads do not matter and propose the Visual Vulnerability Spectrum to characterize them
- The VVS consists of three orthogonal axes
  - Extent—how many pixels will be affected by an error
  - Magnitude—how severe is the error within the affected region
  - Persistence—how long will the error effect the output
- For an error to be important, it must rank high on all three axes

# Important Structures

- Matrix Stack
- Scissor, depth and alpha test enable bits and functions
- Viewport and clip plane function coefficients
- Depth range
- Lighting enable bits
- Culling enable bits
- Various polygon state, including fill mode, offset and stippling
- Various texture state, including enable, active texture, and current texture unit
- Current drawbuffer
- Uniform and control-related shader state

# Unimportant State

- The framebuffer
- Shader data registers
- Antialiasing state

# Mitigation Techniques for Graphics Applications

- Periodic Detection
  - Requires reliable backing store and driver support
  - Takes advantage of ‘acceptable error’
  - Techniques
    - Refresh-based
      - Piggyback error detection on DRAM refresh
    - Demand
      - Piggyback detection on use
      - Example: CRC on vertex array, checked as read
      - Analogs for texture reads?

# Conclusions

- Architectural vulnerability increasingly deserves the attention of the graphics community
  - But AVF is a poor metric for graphics computation
- The Visual Vulnerability Spectrum provides a more useful metric
  - Extent, persistence, magnitude
- Graphics hardware design suggests some novel simple error mitigation techniques
  - Partitioned protection, periodic detection

# Thank you!

- Questions?

# Traditional Causes

- Well understood and quantified
- Very important in super-computing installations
  - On order of 10 transient faults/day
- Memory is the primary victim, not logic
- Ziegler shows that only 1 in 40000 incident particles collides with silicon crystalline structure
  - Assuming a 1cm$^2$ processor
    - Approximately 1 collision/11 hours at sea level
    - 1/6 minutes in an airplane
    - Not all are high enough energy to cause errors
    - Clearly not very important for traditional graphics!

# Near Future Causes

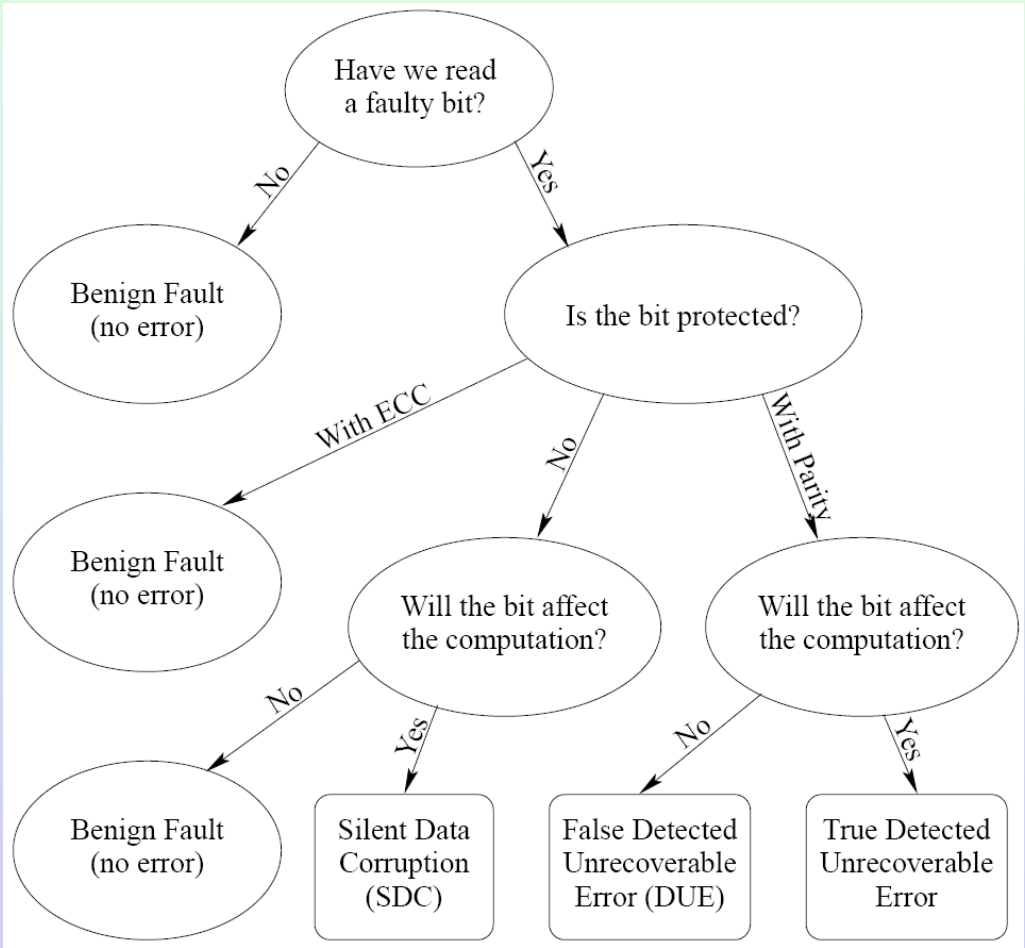
- External EM noise
- EM noise from crosstalk
- *di/dt* and voltage droop
- Parameter variations

- Most of these can be at least partially accounted for through architectural or circuit level techniques
  - e.g. capacitors to compensate for *di/dt*
- Overclocking exacerbates these problems

# Near Future Causes

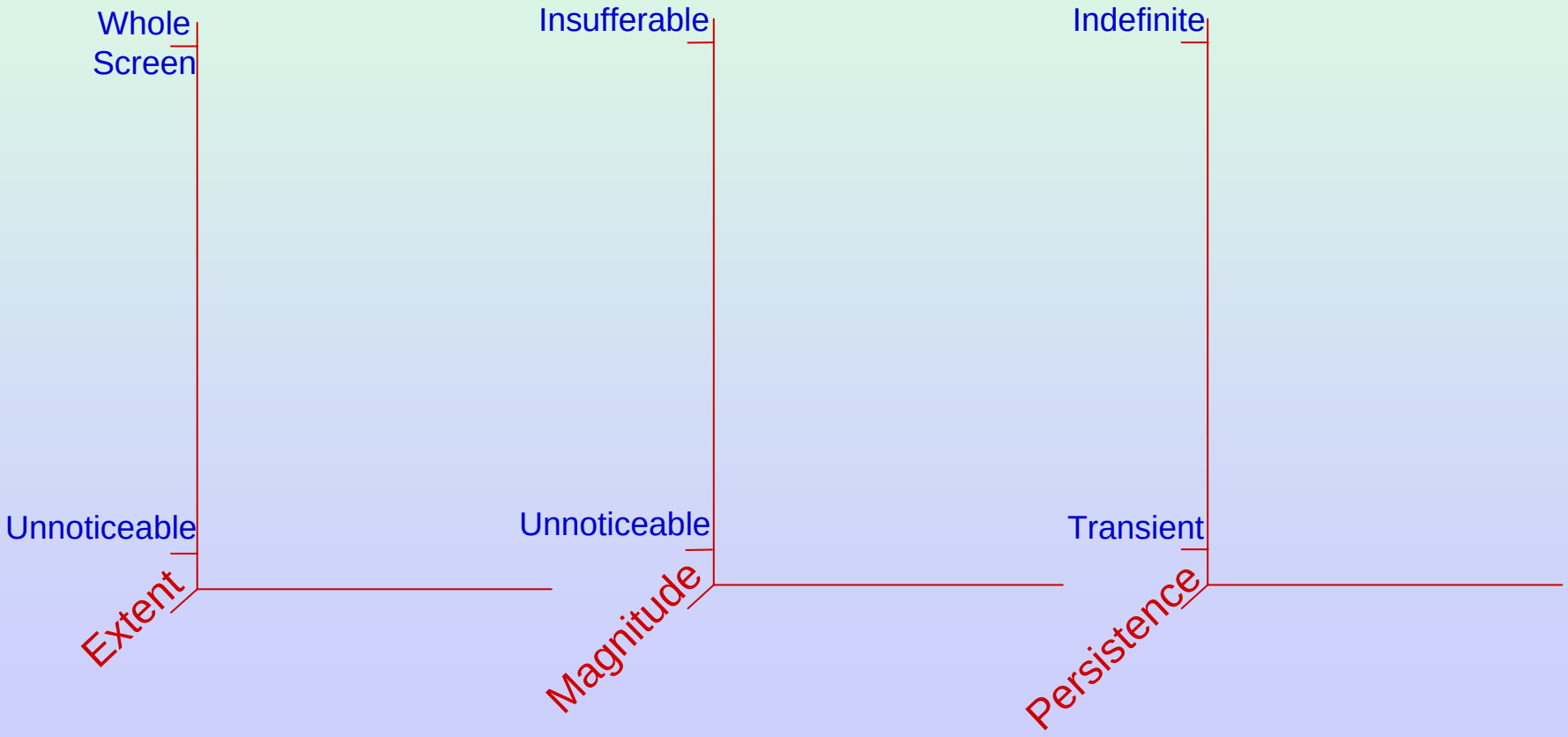
- Neither well understood nor well quantified
  - Borkar shows exponential growth of transient errors at a rate of 8%/generation
  - Very little other literature exists
    - Primarily because nobody yet understands how to analyze the problem

# Architectural Vulnerability Factor



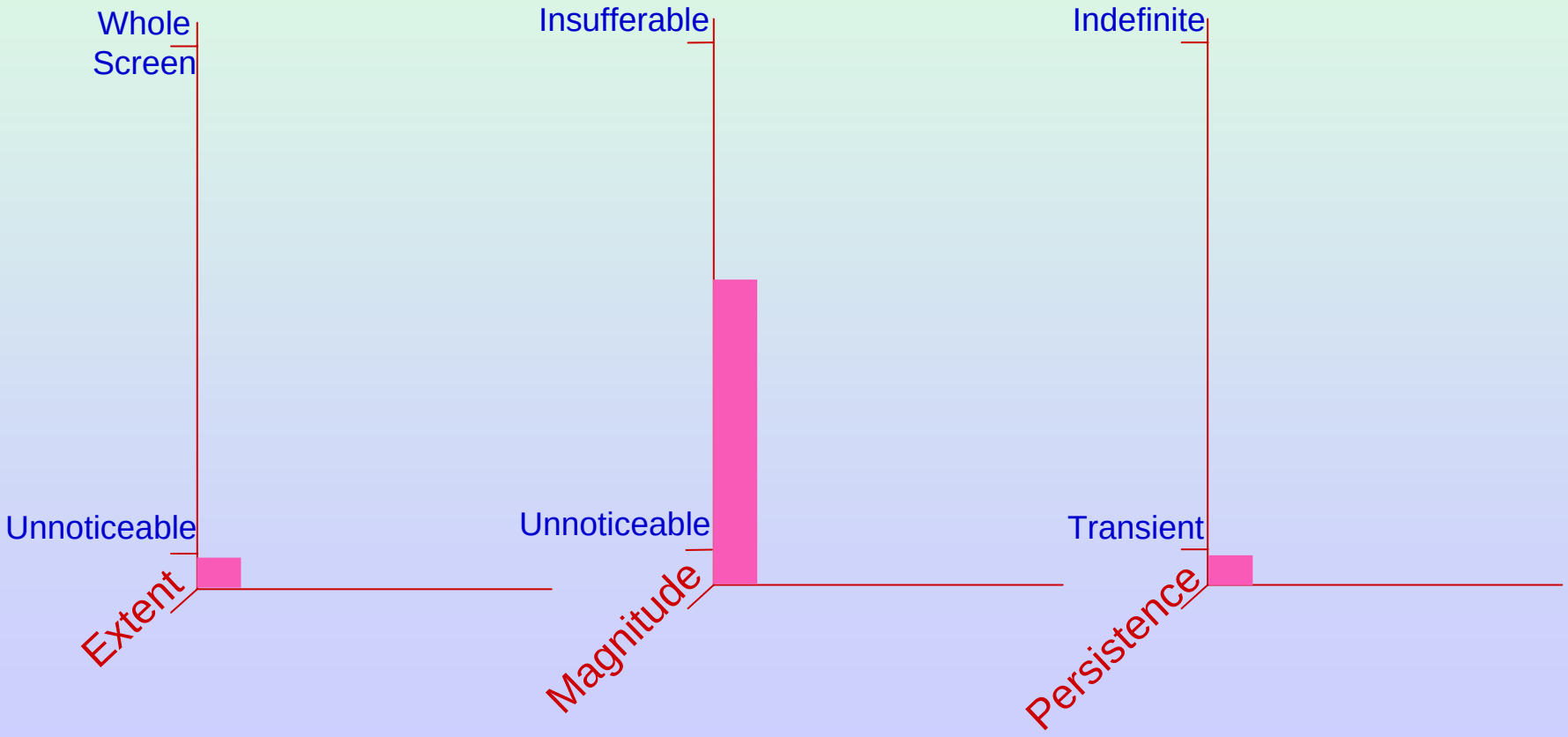

- DUE—Detectable Unrecoverable Error
- SDC—Silent Data Corruption
- ACE—required for Architecturally Correct Execution
- AVF—Architectural Vulnerability Factor
  - The likelihood that a transient error in a structure will lead to a computational error

# VVS Examples



Single Pixel

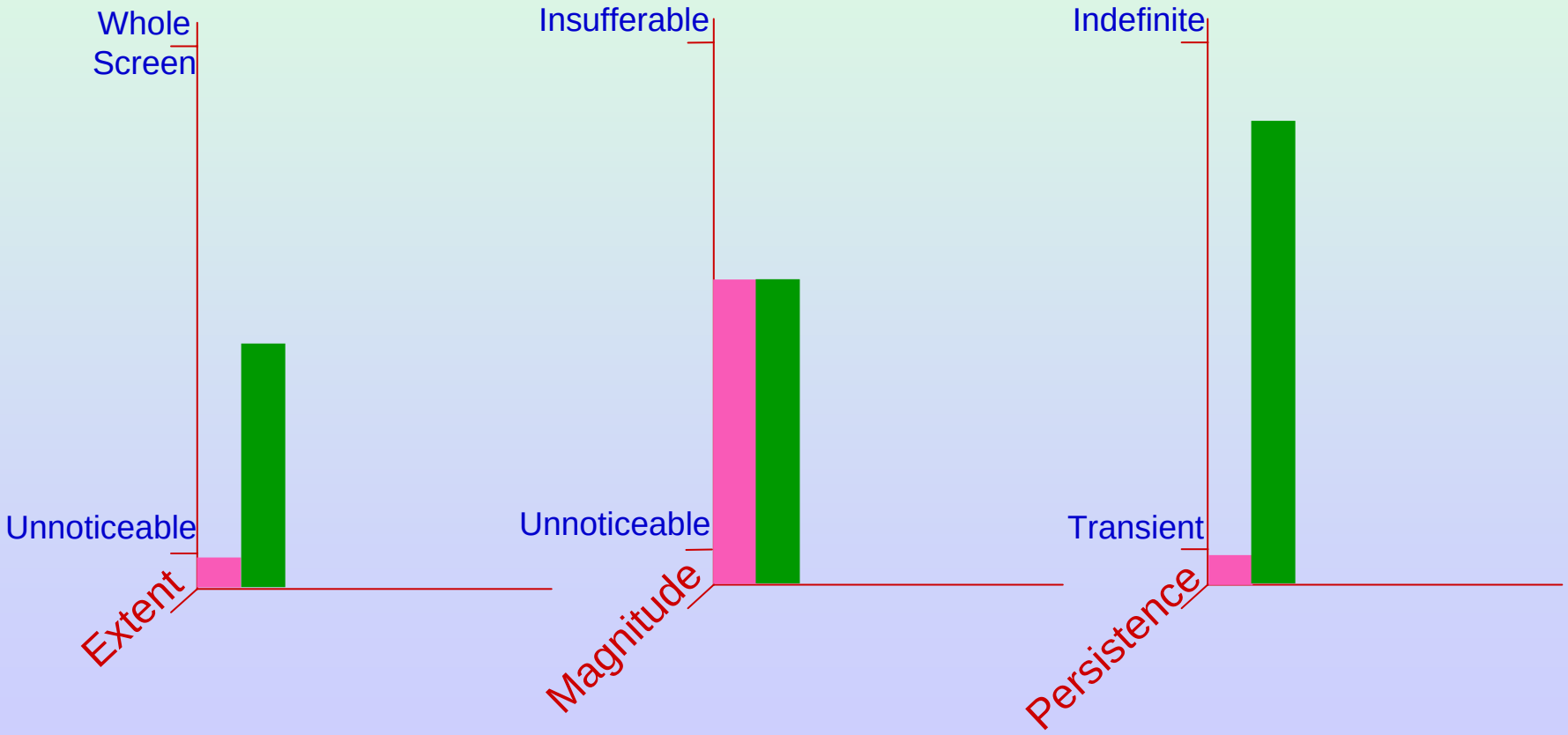
# VVS Examples



# VVS Examples

Single Pixel

Single Texel

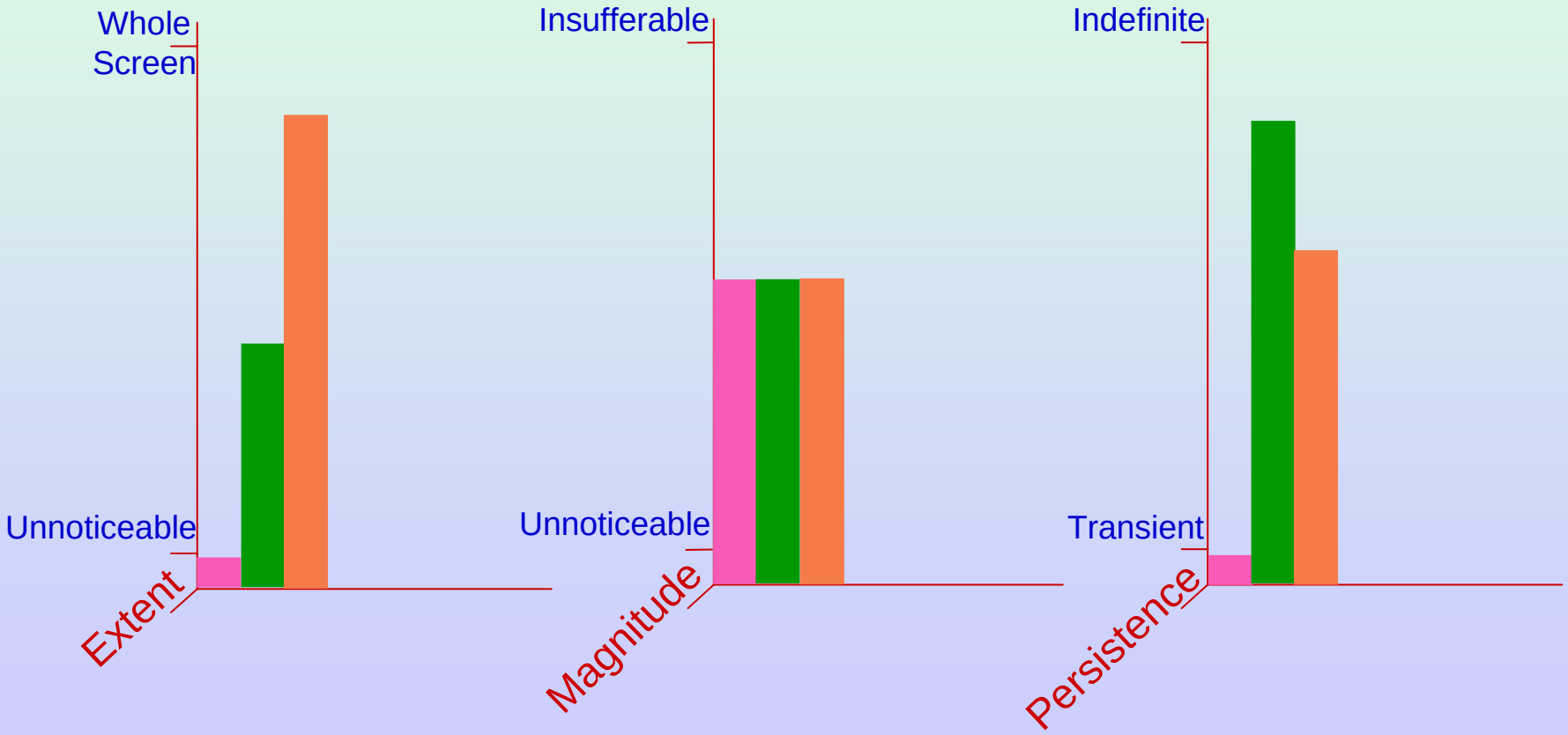


# VVS Examples

Single Pixel

Single Texel

Single Vertex



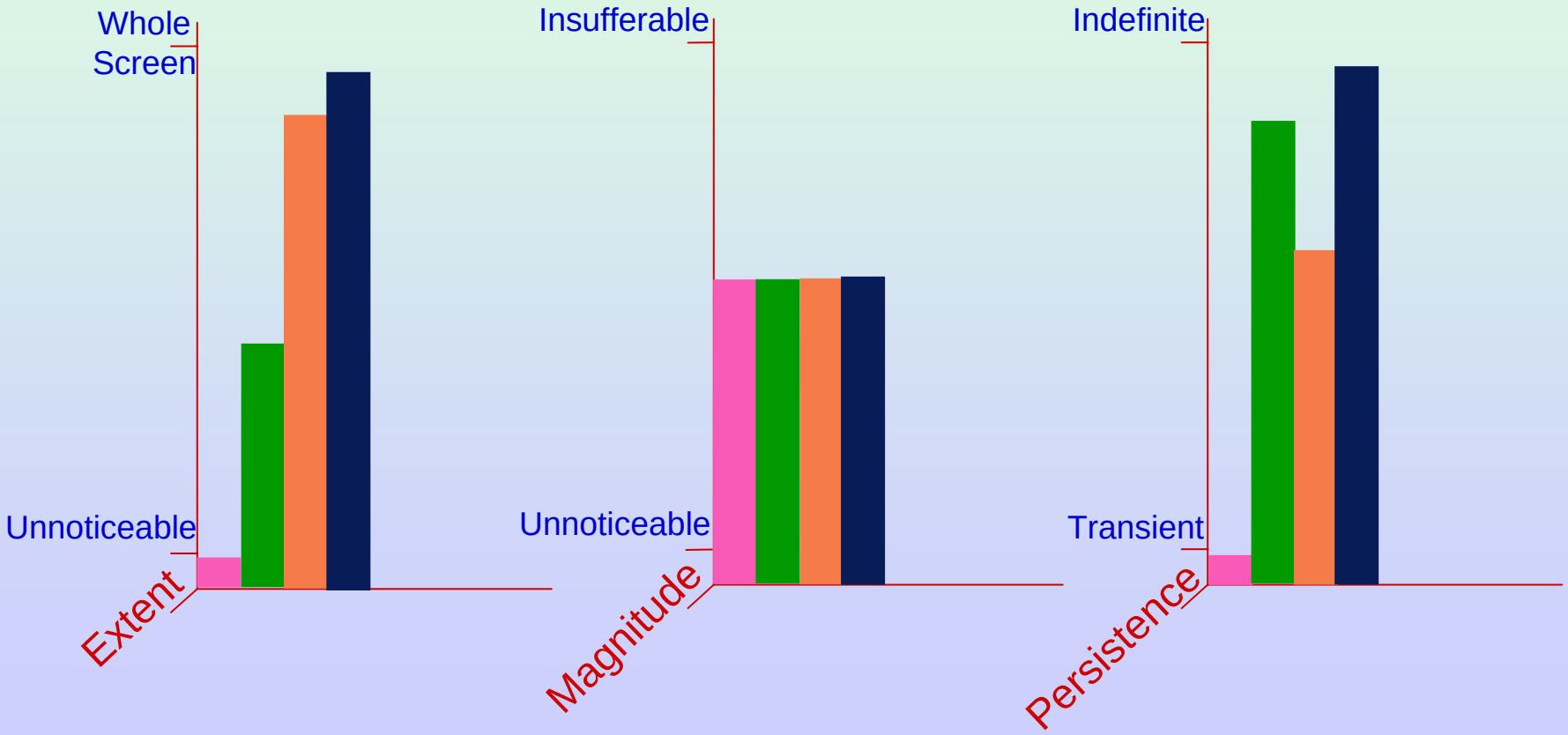
# VVS Examples

Single Pixel

Single Texel

Single Vertex

Uniform Shader State



# VVS Examples

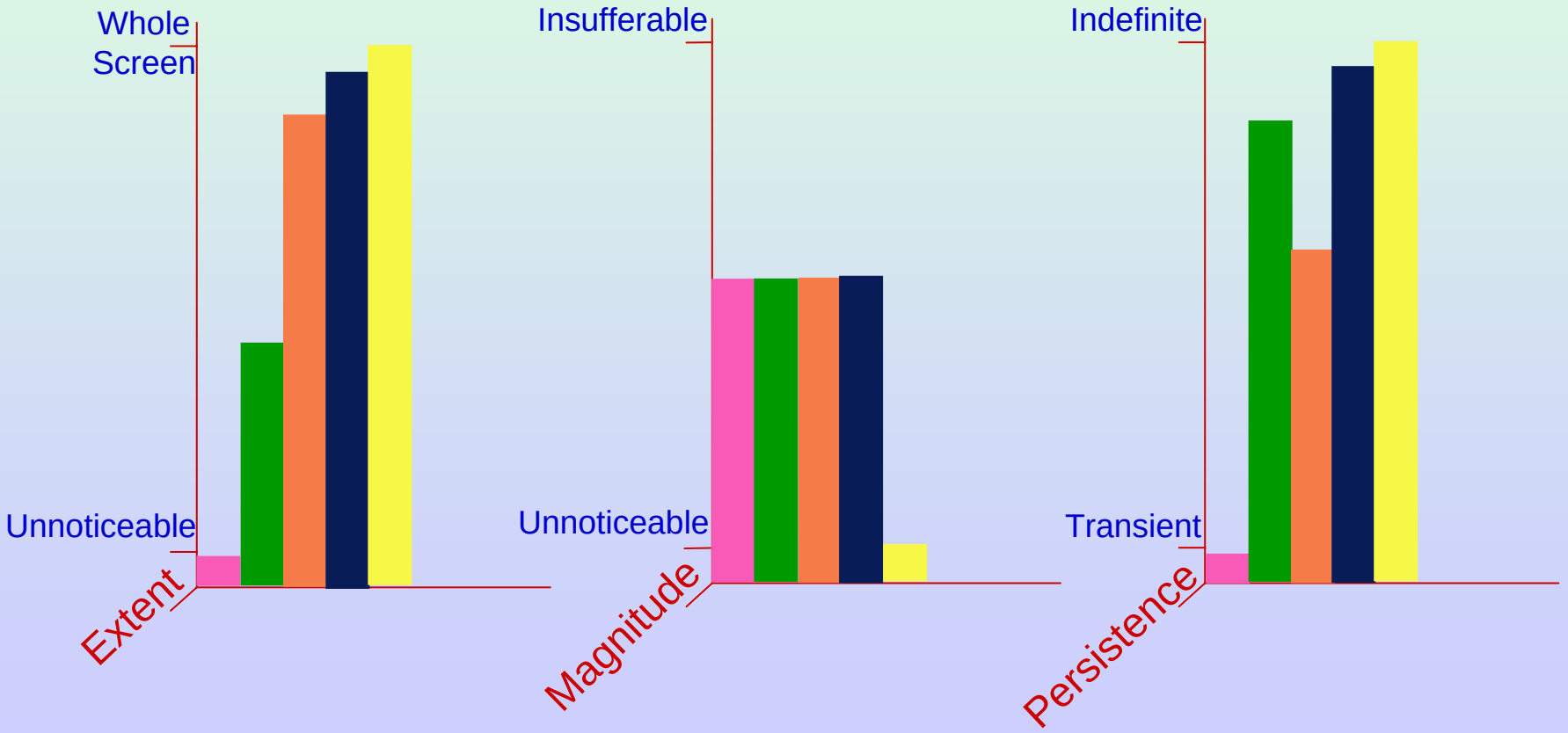
Single Pixel

Single Texel

Single Vertex

Uniform Shader State

Clear Color



# VVS Examples

Single Pixel

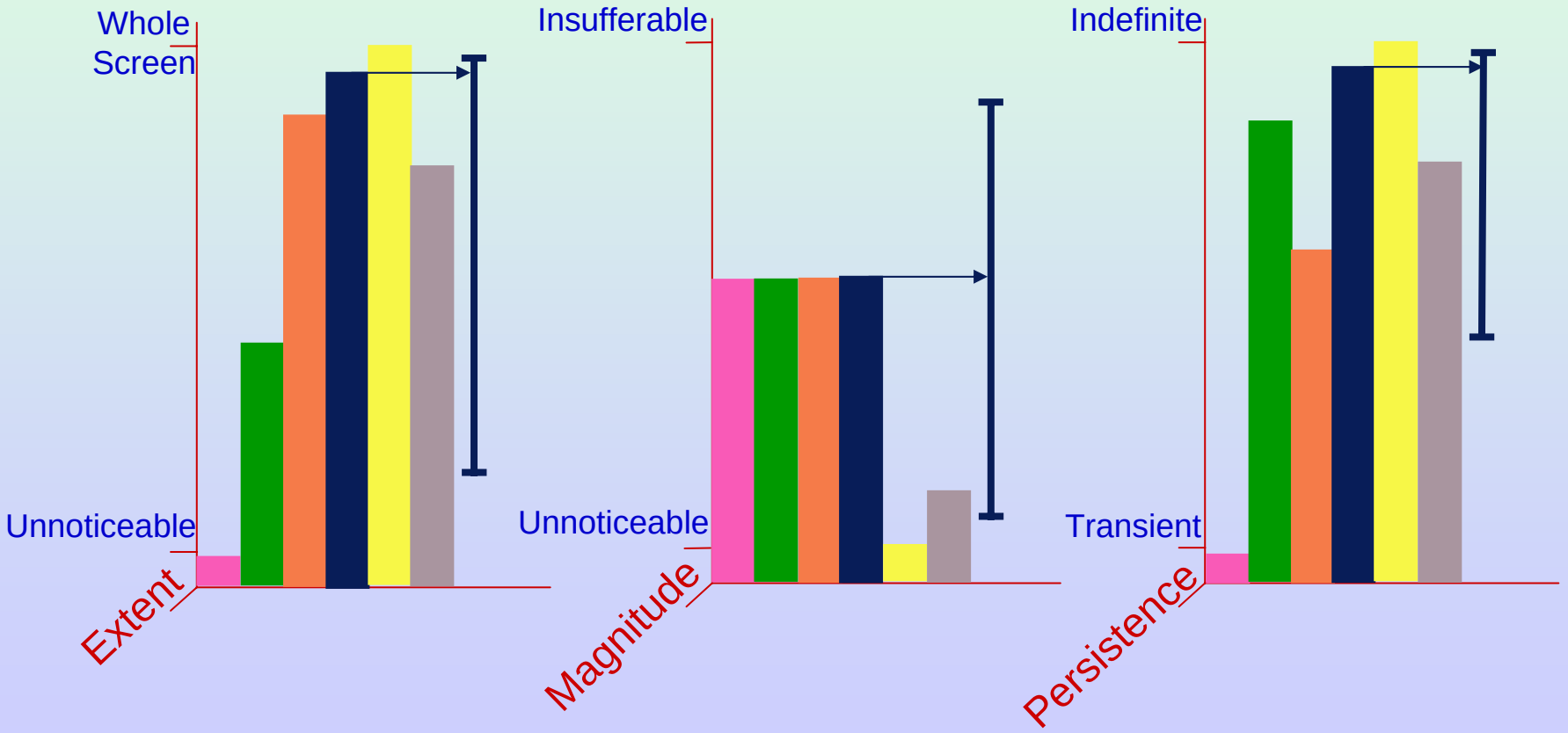
Single Texel

Single Vertex

Uniform Shader State

Clear Color

Antialiasing State



# VVS Examples

Single Pixel

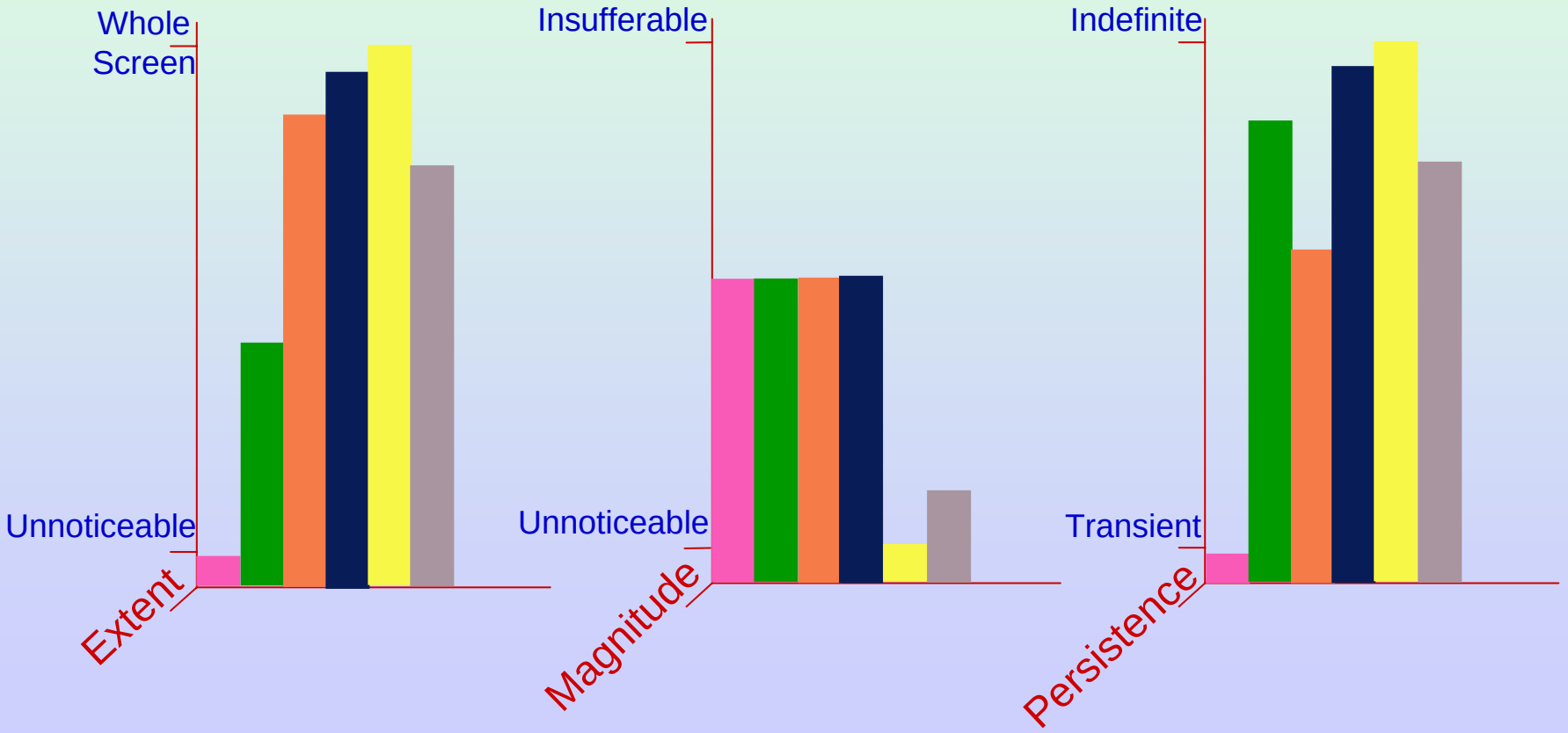
Single Texel

Single Vertex

Uniform Shader State

Clear Color

Antialiasing State



# Application of the VVS

- We analyzed the OpenGL 2.0 state vector using the VVS
  - A “proxy” for real microarchitectures
  - Has shortcomings, but a reasonable, first-order approximation of GPU state
  - We identified some sets of structures of:
    - High importance
    - Intermediate importance
    - Little importance

# Less Important State

- Various vertex array state, including size, type, stride, etc.
- Similar state for other types of arrays: texture, fog, color, etc.
- High levels of the hierarchical Z-pyramid
- Texture contents

# Mitigation Techniques for Graphics Applications

- Full protection, via ECC or similar, on small, not easily recovered important state
  - Various enable bits, matrix stacks, shader control state, clip and viewport coefficients, etc.
- Parity on slightly less important state that can be easily recovered, e.g. shader store

# Future Work

- GPGPU
  - Opportunities afforded by GPU design
  - Redundancy
    - Macro – SLI/Crossfire/video-out based
    - Micro – redundantly combine shader units in space or time
  - Secure backing store
    - Suggests checkpointing-type solutions

# Acknowledgements


- This work is supported by
  - A Graduate Research Fellowship from ATI
  - NSF Grants CCF-0429765 and CCR-0306404
  - Army Research Office grant #W911NF-04-1-0288
  - And a research grant from Intel MRL
- Thanks to the reviewers for their helpful and insightful comments
- Thanks also to Shubu Mukherjee for some otherwise unavailable information

# Design Space Exploration for Low-Cost Safety Critical Architectures

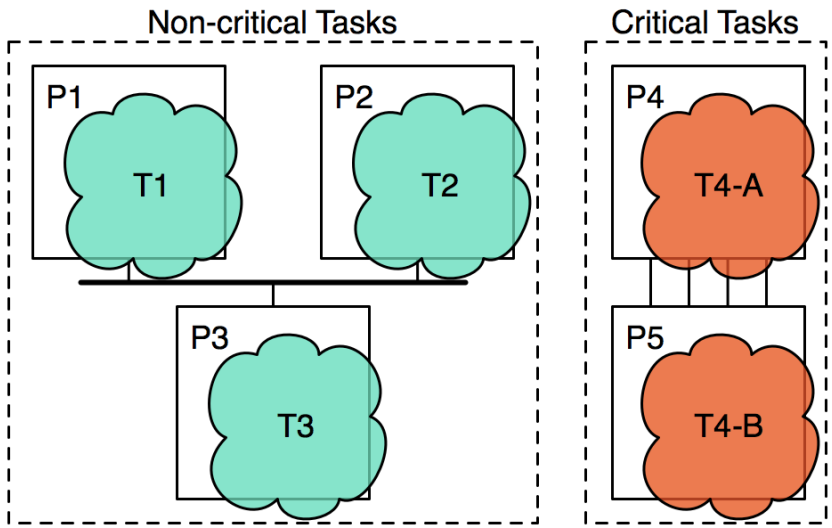
Brett H. Meyer
University of Virginia
April 15, 2010

# Motivation

- Increasing integration of safety-critical systems
- For example, cars
  - X-by-wire
  - Engine-efficiency controls
  - Driver interfaces and navigation aids
- Traditional reliable systems
  - Distributed system of single-core chips
- With emerging multi-core systems, opportunity to
  - Reduce cost with integration
  - Achieve equivalent or better reliability

# Background: Lock-step Execution

- Redundancy to address both soft-, hard-error
- Safety-critical tasks execute on coupled resources
  - Results are compared after each cycle
  - On results mismatch, retry or “limp” home
- Resources tend to be under-utilized => wasted area

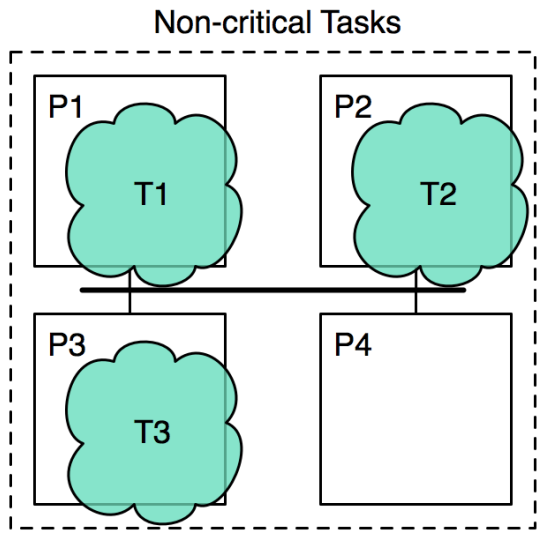


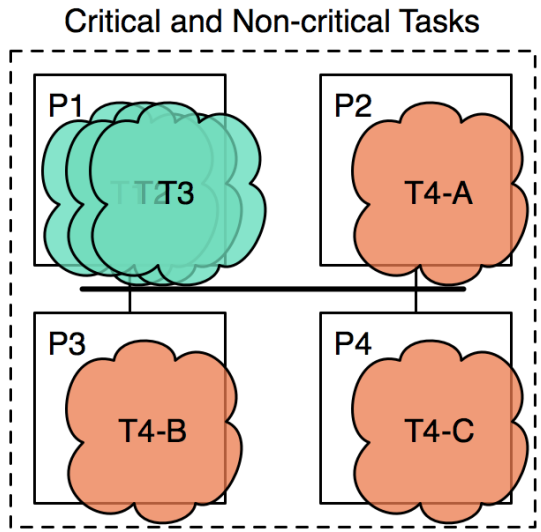
# Objectives

- Develop low-cost, reliable architectures
- Hardware, software alternatives to lockstep on dedicated resources
  - Increase hardware utilization
  - Reduce hardware cost
  - Maintain reliability

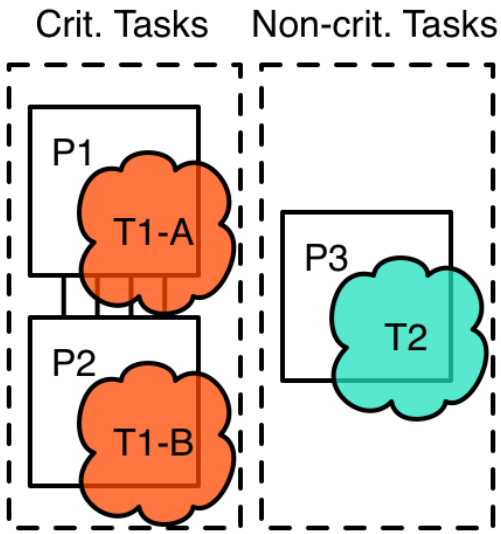
# On-demand Redundancy

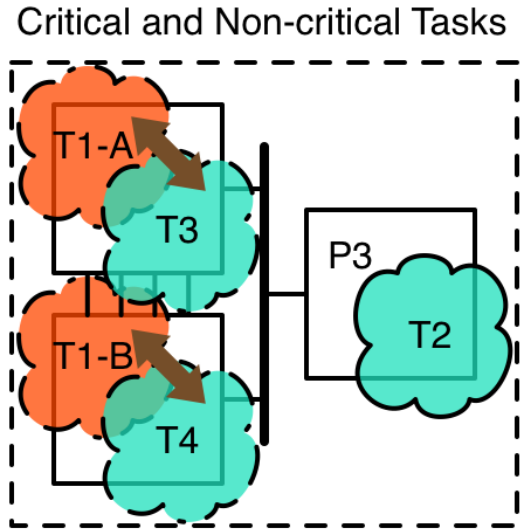
- Relaxing lock-step requirements
  - Relaxing resource dedication
  - Relaxing lock-step execution
- Two key benefits
  - Cost reduction
  - Reliability improvement
- For example, TMR for free!

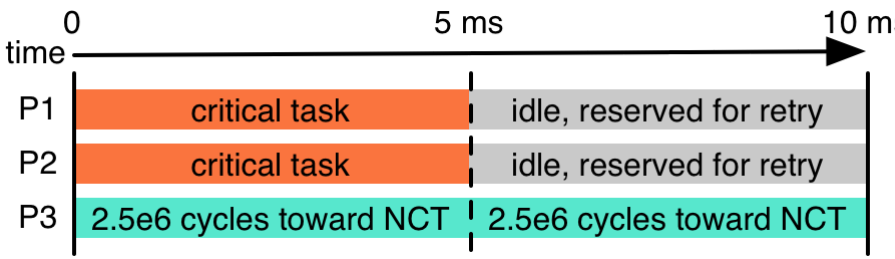


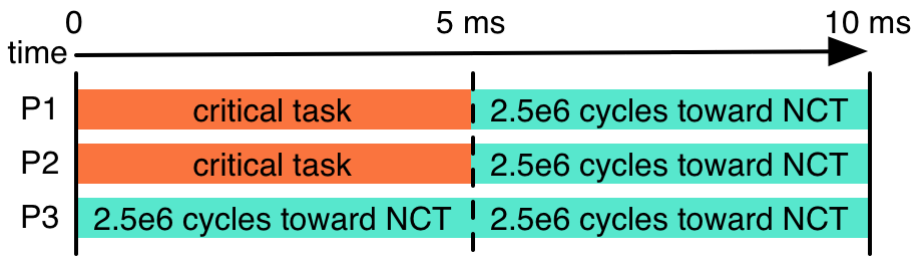



# Relaxing Resource Dedication



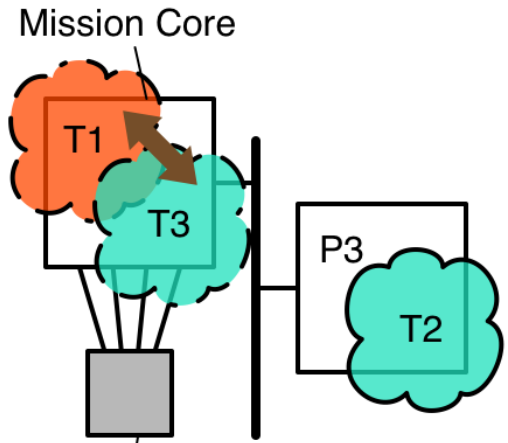



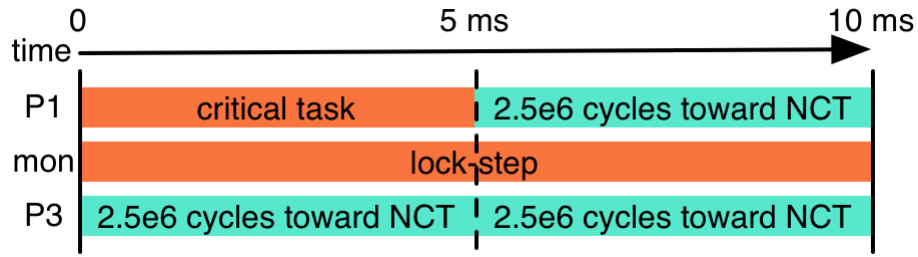




*Non-critical task workload can be increased when dedication is relaxed*

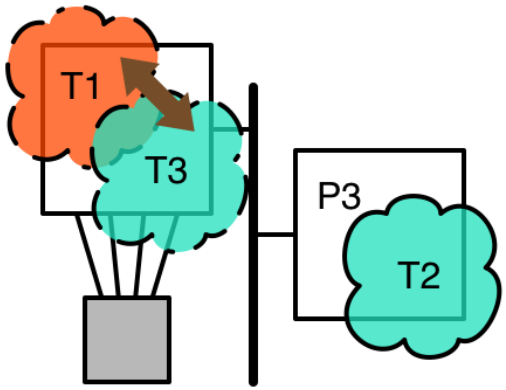
# Mission-monitor Pairs

- Mission core executes critical tasks
- Tightly-coupled monitor core verifies correct execution
- Area reduction of 50% for monitor [Toshiba]
- Monitor is not available for NCT exec

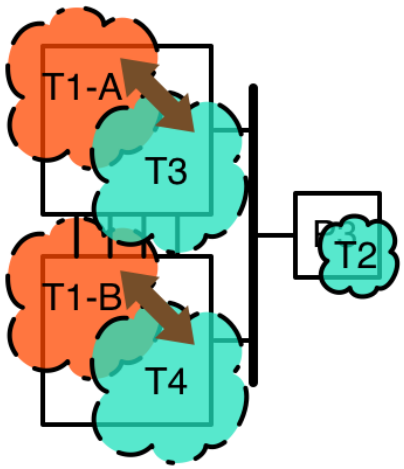


# Mission-Monitor vs. DMR




Mission-monitor

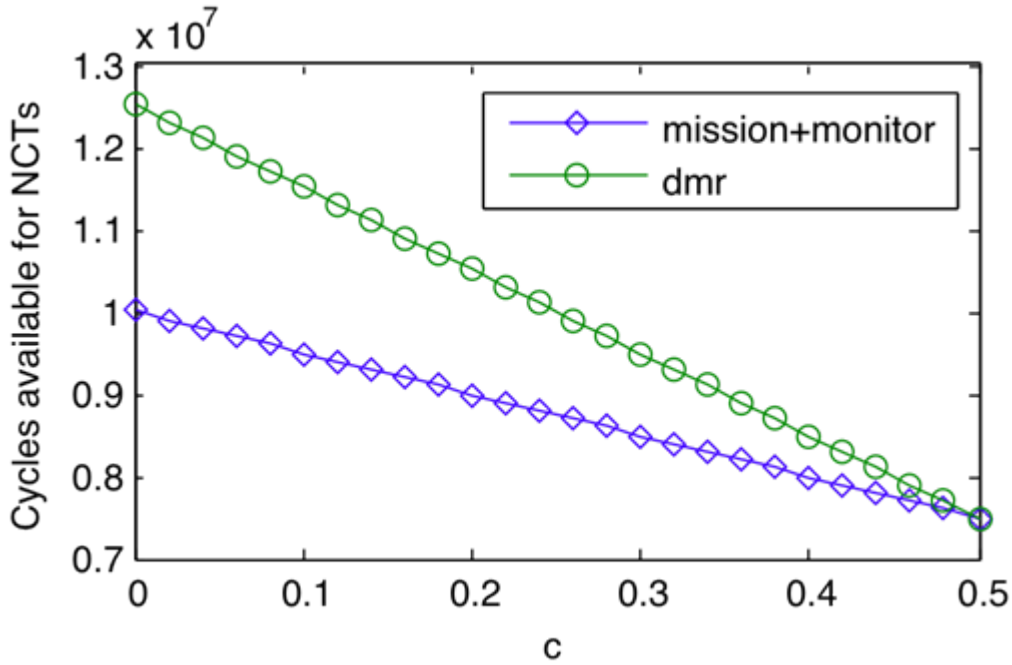



Dual modular redundancy

$$T_{DMR} = 5\,(0.5 + 2(1-c))$$

$$T_{MM} = 5\,(1 + (1-c))$$

$$\frac{T_{DMR}}{T_{MM}} = \frac{2.5 - 2c}{2 - c}$$



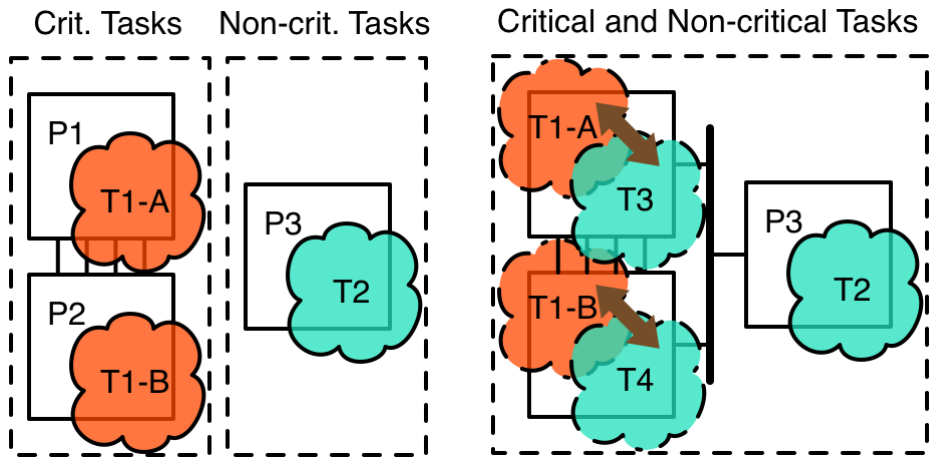
# Future Work

- Validate analytical models
- Extend experimentation to additional templates
- Explore sensitivity to overhead
  - E.g., context switching
- Explore sensitivity to application model
  - E.g., number and organization of critical tasks and non-critical tasks

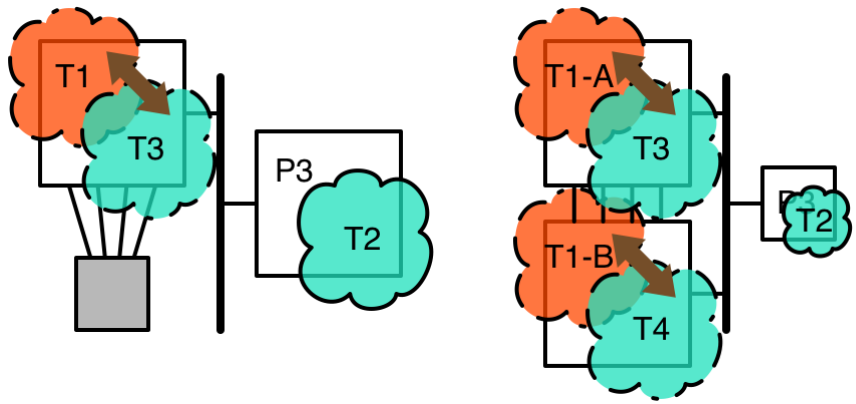
# Questions?

# Cost-neutral Analytical Comparison

- Baseline vs. Relaxed Dedication




- Mission-monitor vs. DMR
  - Both with relaxed dedication



# Experimental Setup

*How many cycles to execute non-critical tasks?*

- Processor model
  - 500/250 MHz ARM processors
- Application model
  - 10 ms scheduling interval, IPC of 1
  - Mix of critical and non-critical tasks
  - *c* – fraction of interval required for critical tasks
  - Retry slot scheduled immediately after critical tasks
  - Optimistically schedule non-critical tasks
- Failure model
  - Transient failures are rare events

# Baseline vs. Relaxed Dedication

$$T_{Base} = 5(N-2)$$

$$T_{RD} = 5\left((N-2) + 2(1-c)\right)$$

$$\frac{T_{RD}}{T_{Base}} = \frac{5\left((N-2) + 2(1-c)\right)}{5(N-2)} = \frac{N-2c}{N-2}$$