



CS 159

Two Lecture Introduction

Parallel Processing:

A Hardware Solution

&

A Software Challenge

# Outline

- ★ Hardware Solution (Day 1)



- ★ Software Challenge (Day 2)

- ★ Opportunities



# Key Points from Day 1

## Hardware Solution

- Parallel Processing is Essentially an Evolution in
  - Micro- and Macro-Architecture Hardware
    - That provides a Solution to:
      - The Heat and Power Wall
      - The Limitations of ILP
      - Cost-Effective Higher Performance
- **HW Paradigm Shift Occurring (esp. Micro-level)**
  - **More Cores; Not a Faster Clock or more ILP**

# Outline

- ★ Hardware Solution



- ★ Software Challenge

- Technical
- Business

- ★ Opportunities

# Software Challenge - Technical

## Change in Hardware Requires Change in Software

- ❖ The Car (Hardware) has Changed
  - From a Sequential Engine To a Parallel Engine
- ❖ The Driver (Software) Must Change Driving Techniques
  - Otherwise, Sub-Optimal Performance will Result



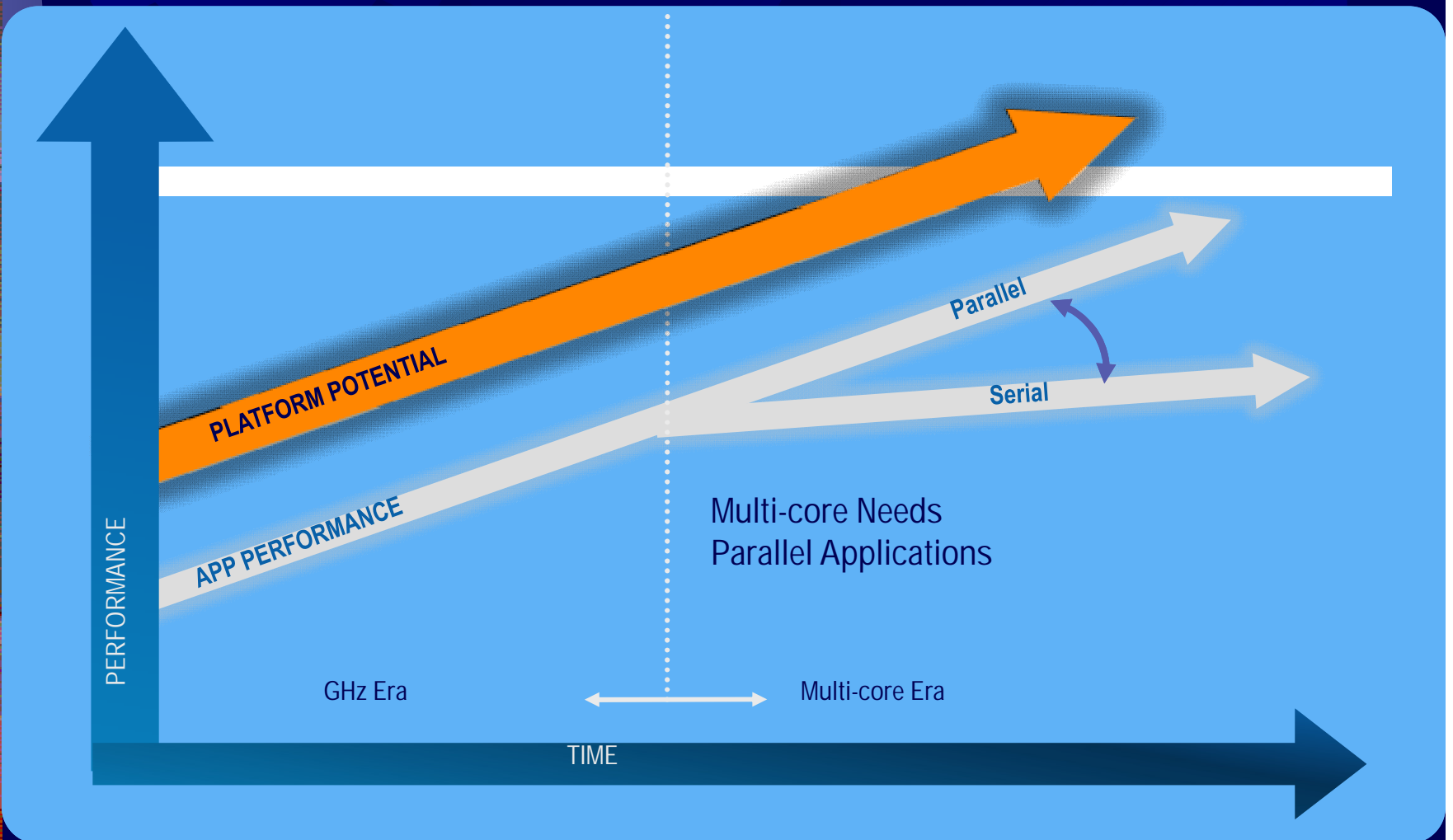
Car  
&  
Driver

Hardware  
&  
Software



# Software Challenge - Technical

- Because HW is going Parallel, so must the SW in order to get performance gains from HW platform



# Software Challenge - Technical Overview

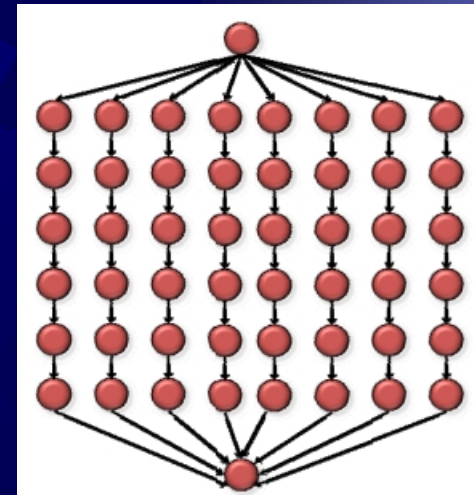
- ❖ The Challenge:

Cannot Extract Parallelism Without User Support

- ❖ The Goal: Make Parallel Programming the Mainstream Method for Improving SW Performance

- ❖ But Parallel Programming is Harder in all Aspects:

- Design & Re-engineering
- Debugging
- Testing
- Profiling
- Scaling



# Software Challenge - Technical Design & Re-engineering

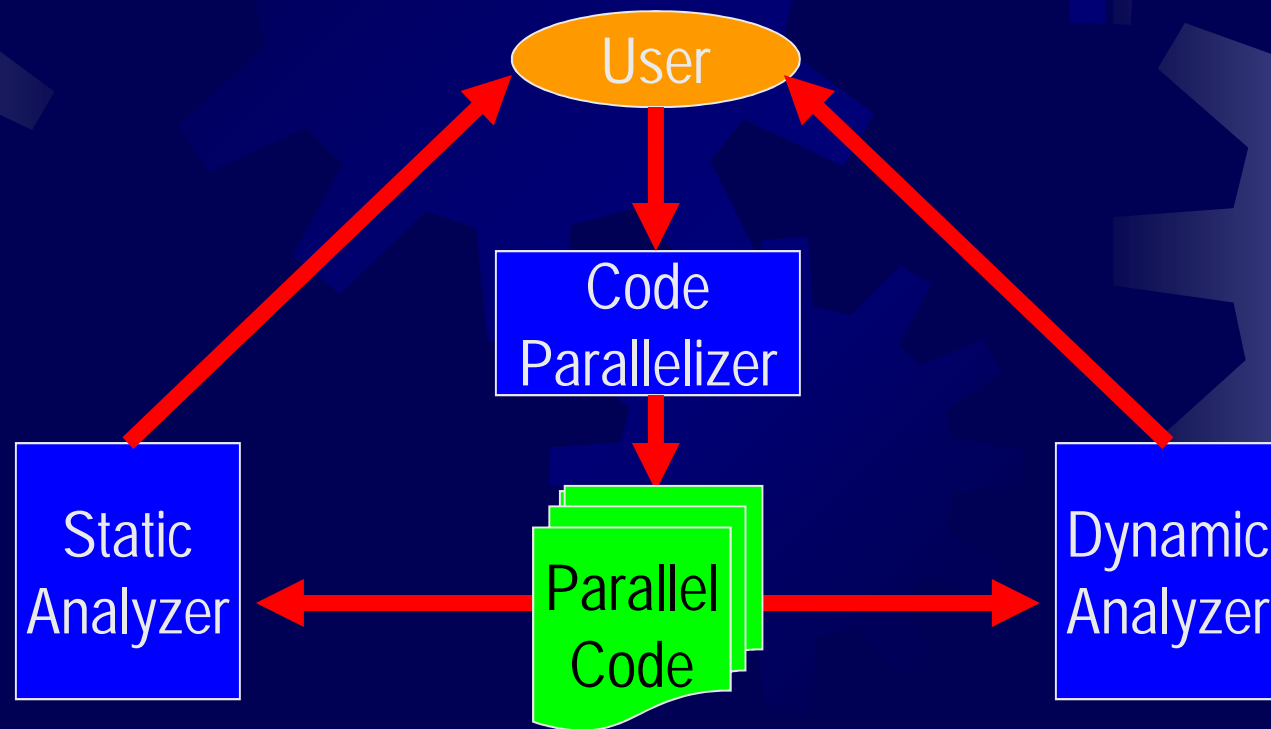
- ❖ Increased Complexity will Require More Careful Analysis
  - Parallelism Adds Temporal Dimension to Problem
  - Hard for Humans to Think about Parallel Events
  - Large Permutation of Operation Interleavings Possible
- ❖ Increased Programmer Expertise Needed in:
  - The Application Domain and Algorithms
  - Source Code Parallelization Techniques
  - Communication & Synchronization
  - Performance Optimization





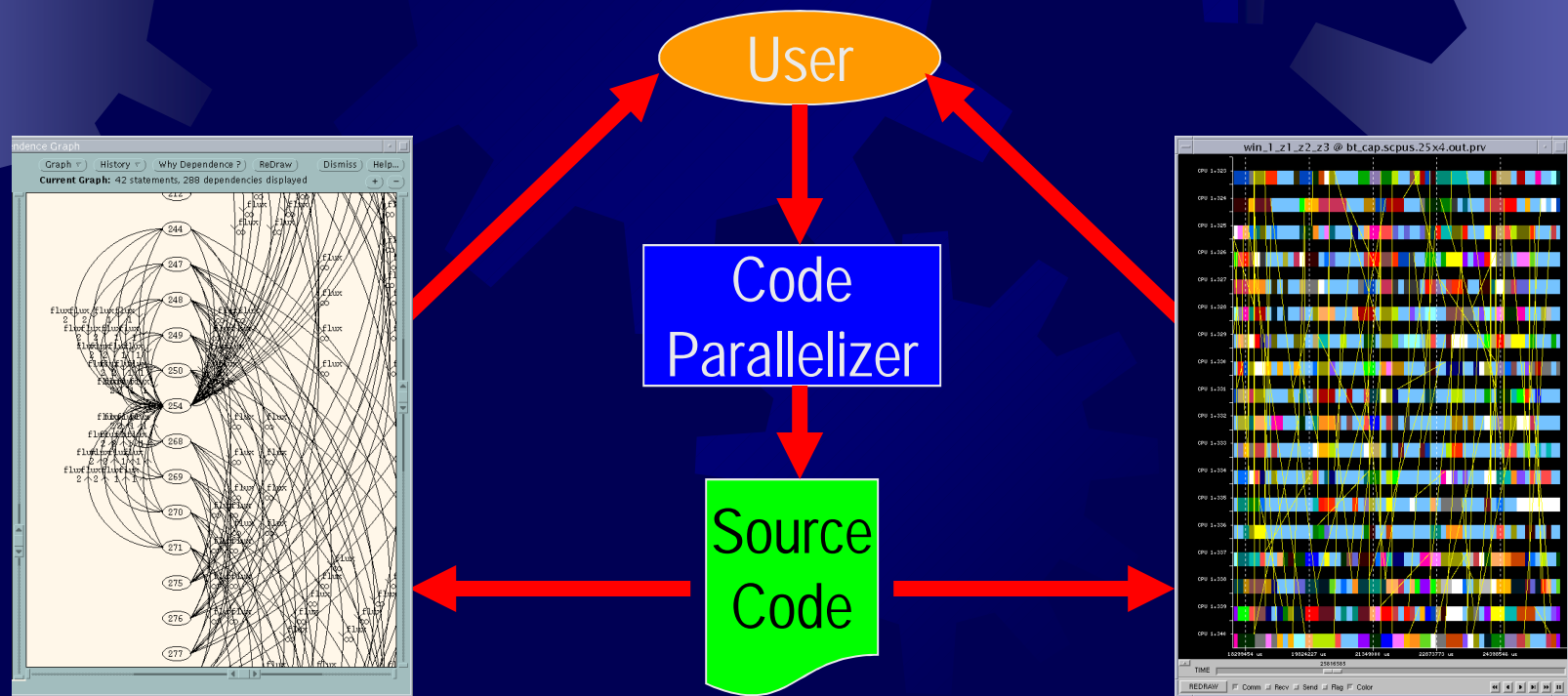
# Code Parallelization Tools

- Static Dependency Analyzer: Draws Call-Graph Structure
- Dynamic Profile Analyzer: Plots Thread Activity vs. Time
- Code Parallelizer: Parallel Language, Paradigm, Compiler



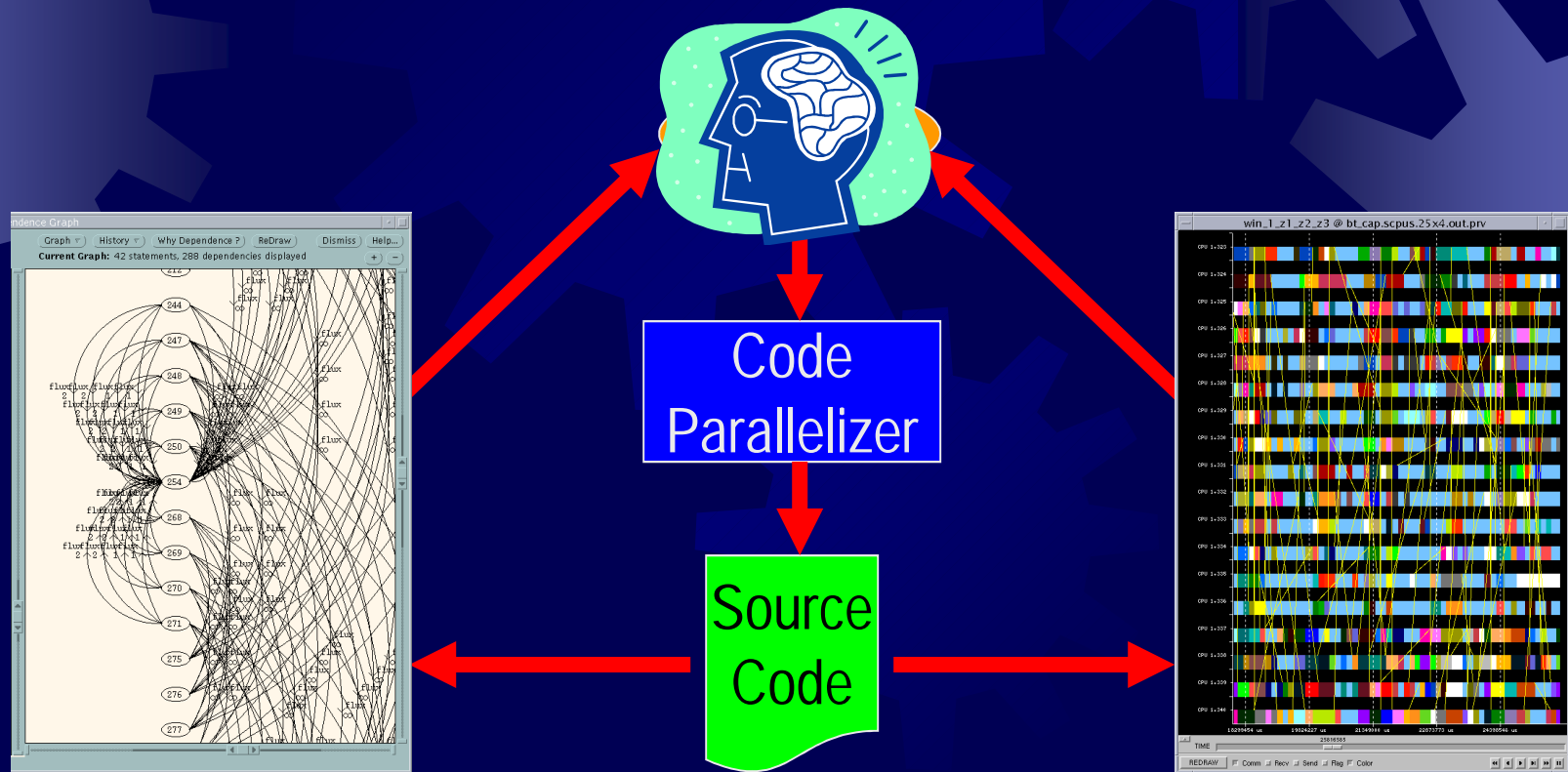
# User Role in Code Parallelization

- Run & Interpret Static Data Dependency Analysis
- Run & Interpret Dynamic Profile Analysis
- Drive Code Parallelizer Transformations



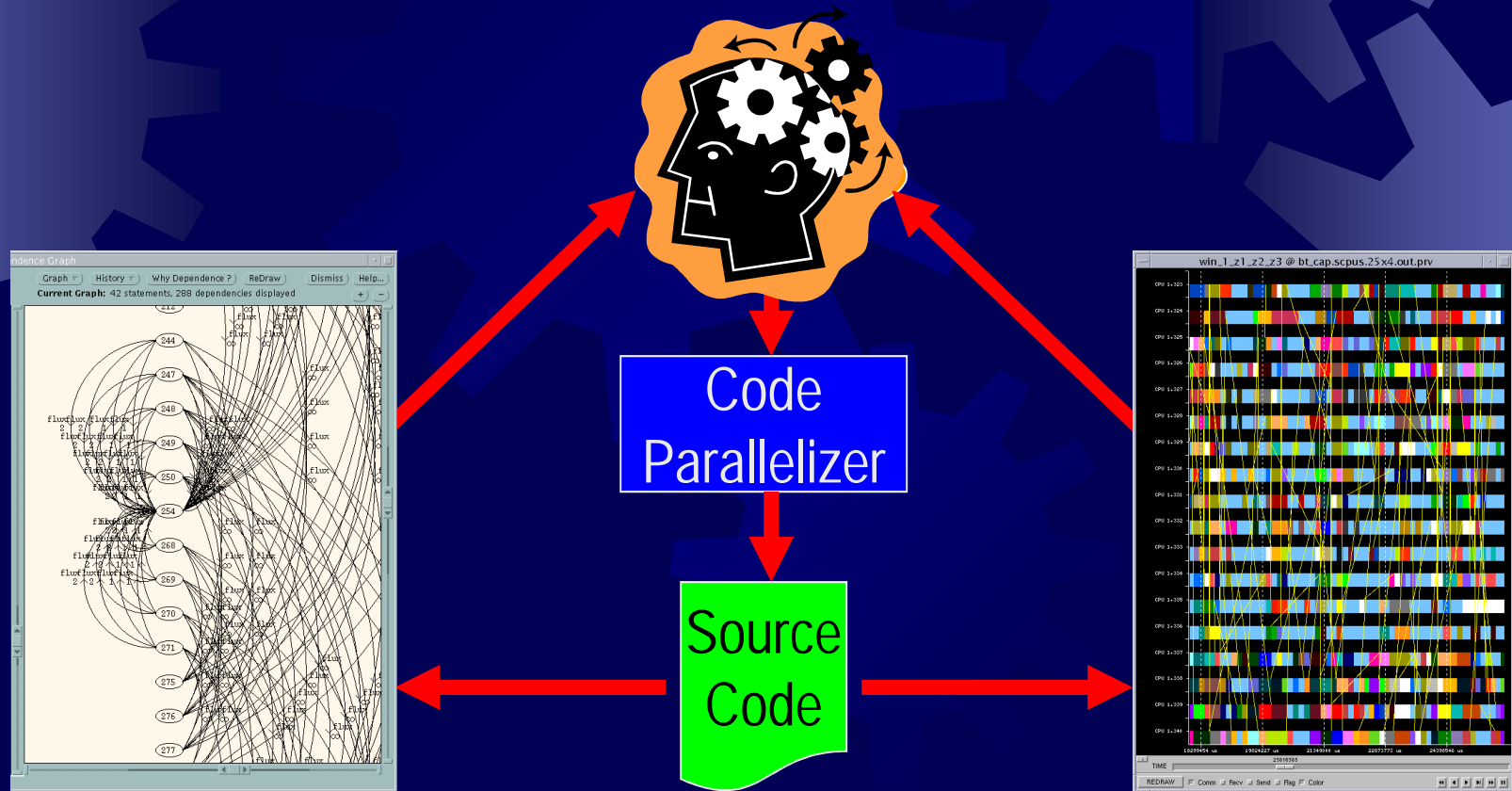
# User Knowledge in Code Parallelization

- High-Level Application Code and Algorithms
- Low-Level Thread and Communication Profile
- Source Code Parallelization & Optimization Techniques



# User Knowledge in Code Parallelization

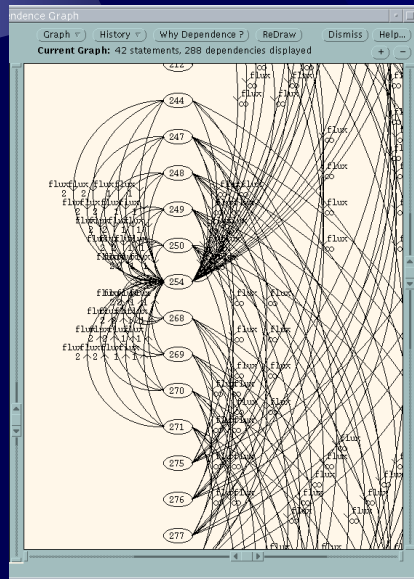
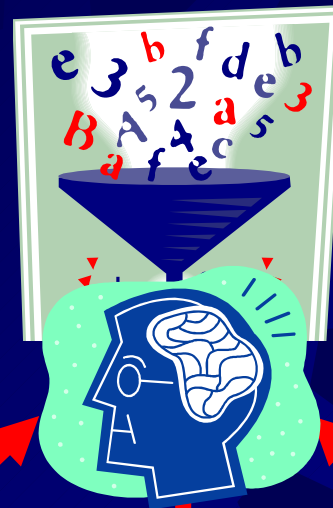
- High-Level Application Code and Algorithms
- Low-Level Thread and Communication Profile
- Source Code Parallelization & Optimization Techniques



# Goal: Assist The User

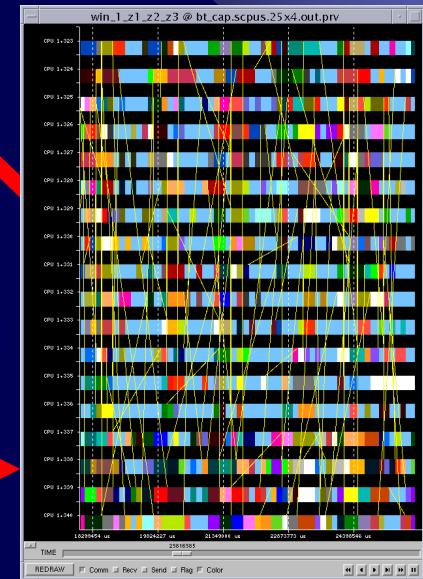
*Help User Focus on  
Relevant Information  
from Analyses*

*Help User Focus on  
the Code with Best  
Potential Speedup*



Code  
Parallelizer

Source  
Code



# Ultimate Goals

## An Ideal Set of Parallelization Tools Would:

- Off-Load User as Much as Possible
- Make Parallelization Easier and More Efficient
  - Maximize Code Performance Gain
  - Minimize Analysis and Transformation Time
- Perform Data Fusion on Static and Dynamic Analysis
  - Filter, Correlate, and Interpret the Results
- Produce Correct, Bug-Free Parallel Code
- Increase Degree of Automation

**But, Current SW Tools Still Need Further Development**



# Software Challenge - Technical Tools

- ❖ Lack of Tools Compounds Problem
  - Existing Tool Chain only for Sequential Programming
- ❖ Need New Parallel Programming Tools & Infrastructure
  - Effective Models for Parallel Systems
  - Constructs to make Parallel Architecture more Visible
  - Languages to More Clearly Express Parallelism
  - Reverse Engineering Analysis Tools
    - To Assist with Conversion of Sequential to Parallel
      - Especially for Optimized Sequential Code



# Software Challenge - Technical Race Conditions

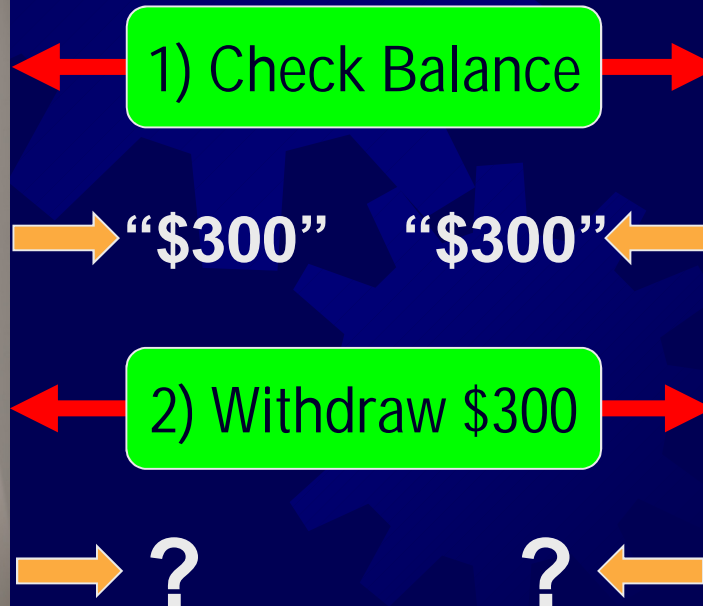
- ❖ Parallelism can Give Rise to a New Class of Problems
  - Caused by the Interactions Between Parallel Threads
- ❖ Race Condition:
  - Multiple Threads Perform Concurrent Access to the Same Shared Memory Location**
- ❖ Threads “Race” Against Each Other
  - Execution order is assumed but cannot be guaranteed
  - Outcome depends on which one wins (by chance)
  - Results in Non-Deterministic Behavior



# Software Challenge - Technical

## ATM Race Condition Example

Joint Bank  
Account  
\$ 300

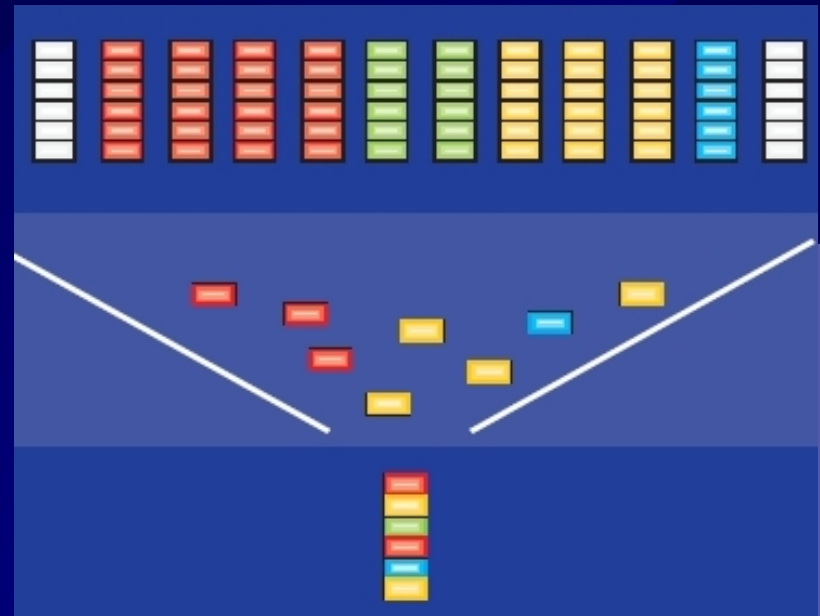
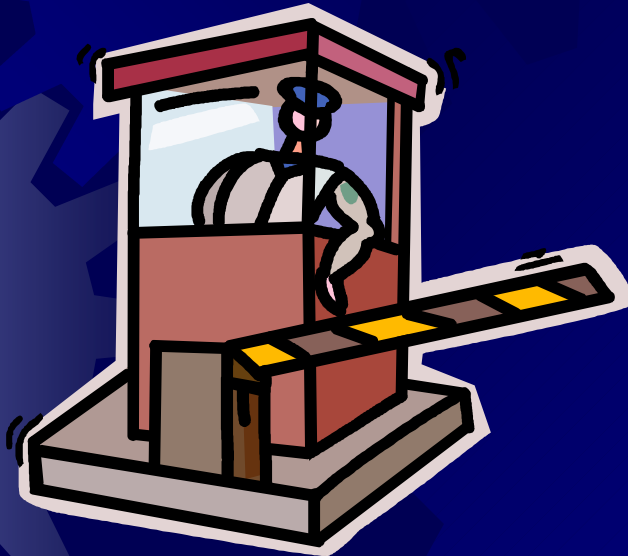




# Software Challenge - Technical Race Conditions

- ❖ Race Conditions are Especially Hard to Detect & Debug
  - Errors are Very Subtle
    - No Apparent “Failure” Occurs
    - Program Continues to Run “Normally”
    - Program Completes “Normally”
  - Errors are Intermittent
    - Hard to Reproduce and Diagnose
  - Errors Can Slip Through SQA Testing Process
    - Potential Lurking Bug
- **Most Common Error in Parallel Programs**

# Software Challenge - Technical Semaphores

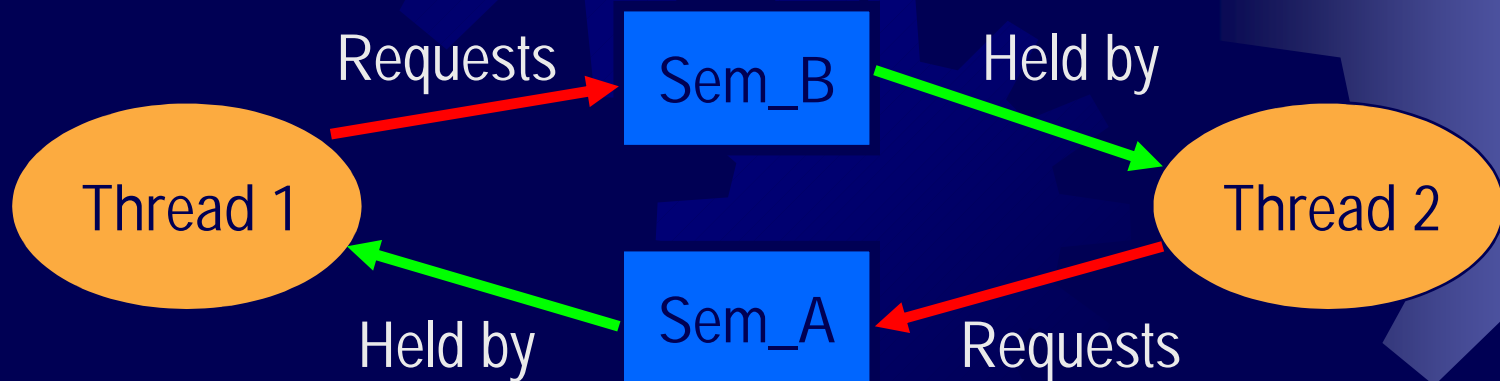


- ❖ Semaphores Offer a Solution to Race Conditions
  - However Semaphores themselves can cause Problems:
    - Introduce Overhead
    - Can Create Bottlenecks
      - Mutually Exclusive (one-at-a-time) Access

# Software Challenge - Technical Deadlock

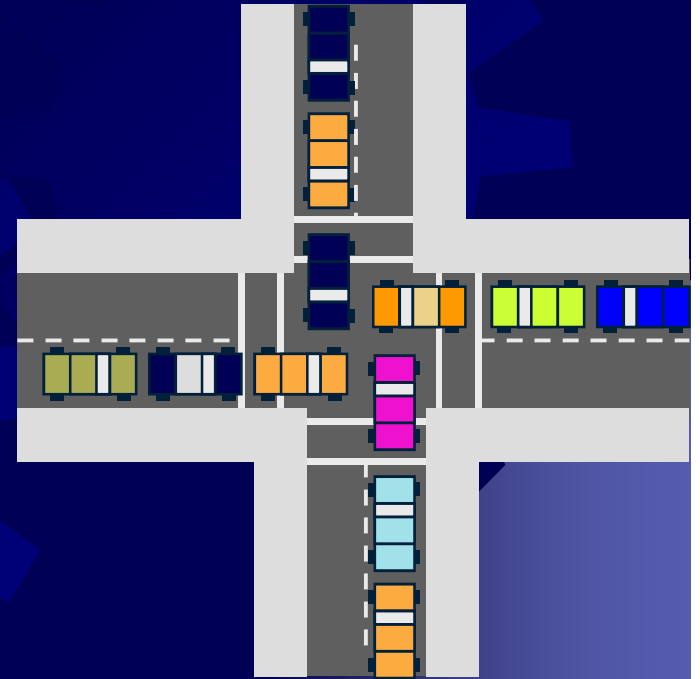
- ❖ Another Potential Problem Arising From Parallelism
- ❖ Deadlock:

Two or More Threads are Blocked because  
Each is Waiting for a Resource Held by the Other

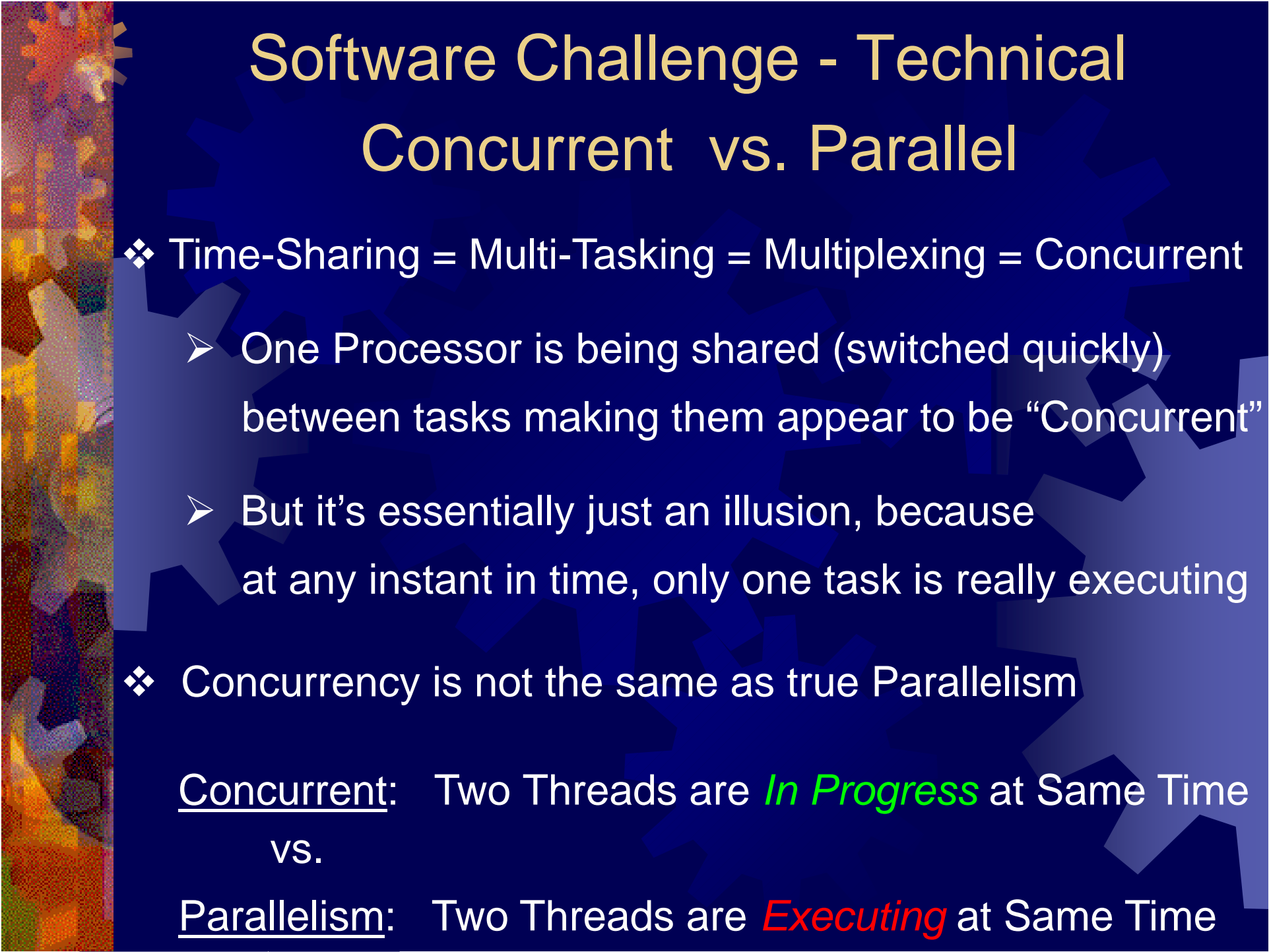


# Software Challenge - Technical Deadlock

- ❖ Not as Hard as Race Conditions
  - Errors are More Obvious
    - System Usually Freezes
- ❖ But Similar to Race Conditions
  - Errors are Intermittent
    - Hard to Detect, Reproduce, Diagnose, Debug
  - Errors Can Slip Through SQA Testing Process
    - Potential Lurking Bug



➤ Another Common Error in Parallel Programs



# Software Challenge - Technical

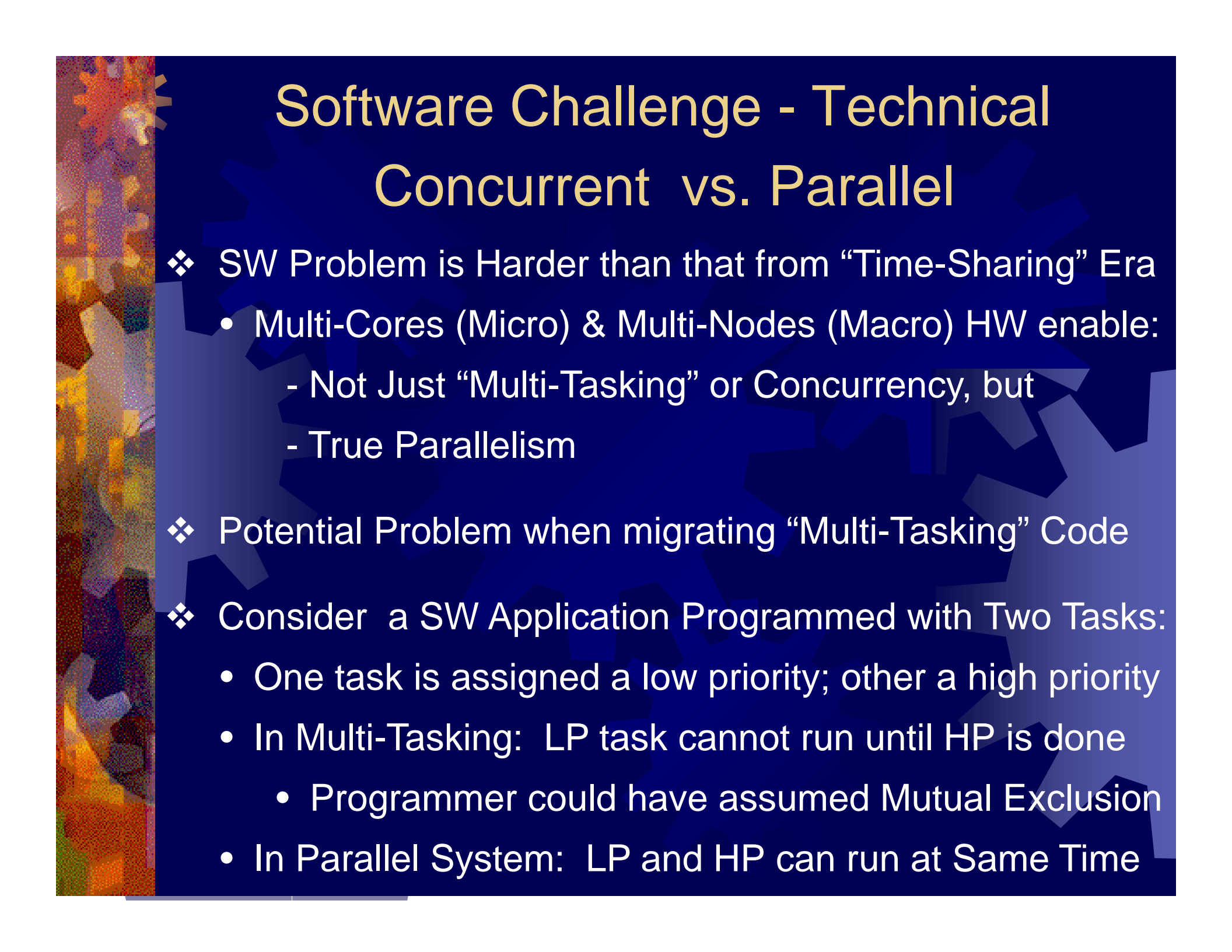
## Concurrent vs. Parallel

- ❖ Time-Sharing = Multi-Tasking = Multiplexing = Concurrent
  - One Processor is being shared (switched quickly) between tasks making them appear to be “Concurrent”
  - But it’s essentially just an illusion, because at any instant in time, only one task is really executing
- ❖ Concurrency is not the same as true Parallelism

Concurrent: Two Threads are *In Progress* at Same Time

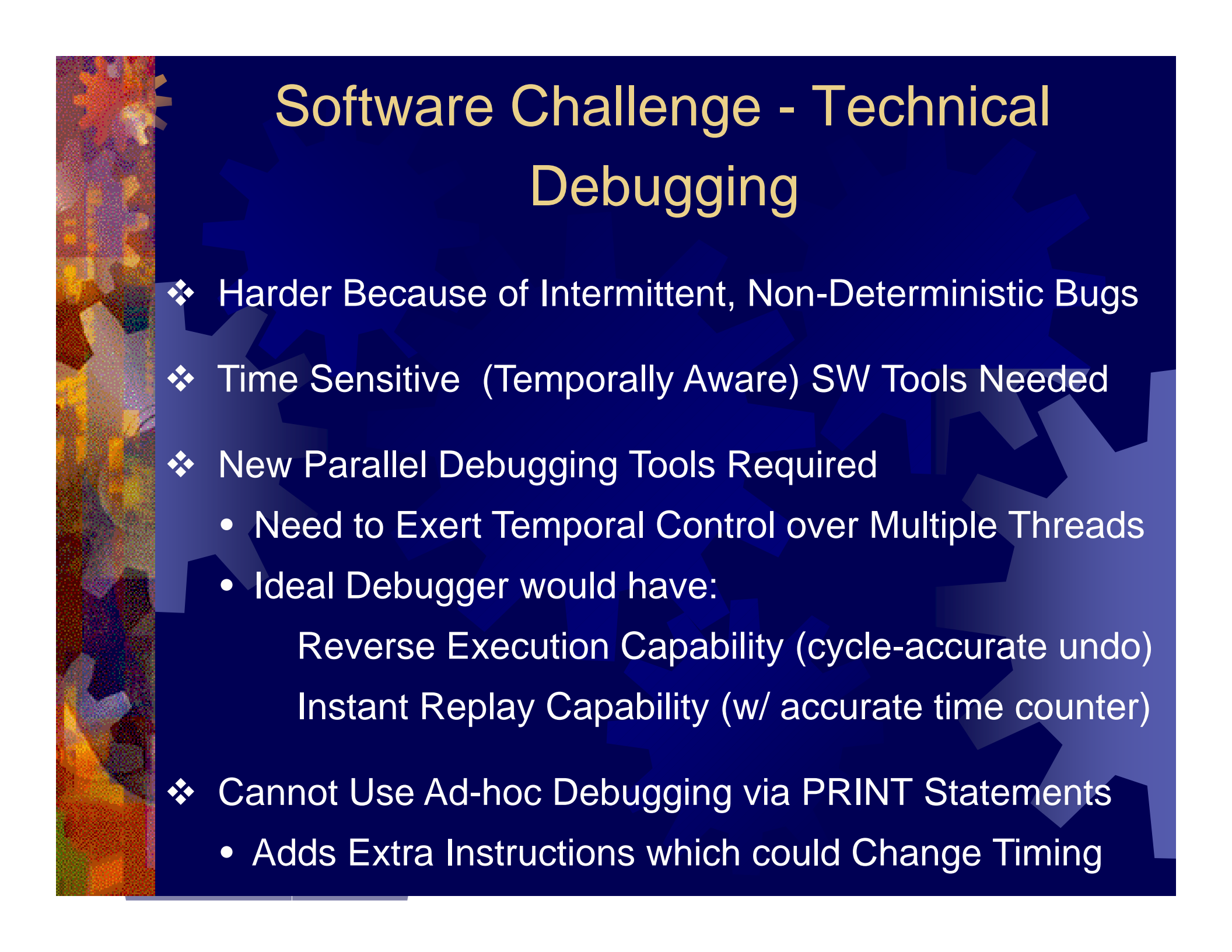
vs.

Parallelism: Two Threads are *Executing* at Same Time



# Software Challenge - Technical Concurrent vs. Parallel

- ❖ SW Problem is Harder than that from “Time-Sharing” Era
  - Multi-Cores (Micro) & Multi-Nodes (Macro) HW enable:
    - Not Just “Multi-Tasking” or Concurrency, but
    - True Parallelism
- ❖ Potential Problem when migrating “Multi-Tasking” Code
- ❖ Consider a SW Application Programmed with Two Tasks:
  - One task is assigned a low priority; other a high priority
  - In Multi-Tasking: LP task cannot run until HP is done
    - Programmer could have assumed Mutual Exclusion
  - In Parallel System: LP and HP can run at Same Time



# Software Challenge - Technical Debugging

- ❖ Harder Because of Intermittent, Non-Deterministic Bugs
- ❖ Time Sensitive (Temporally Aware) SW Tools Needed
- ❖ New Parallel Debugging Tools Required
  - Need to Exert Temporal Control over Multiple Threads
  - Ideal Debugger would have:
    - Reverse Execution Capability (cycle-accurate undo)
    - Instant Replay Capability (w/ accurate time counter)
- ❖ Cannot Use Ad-hoc Debugging via PRINT Statements
  - Adds Extra Instructions which could Change Timing



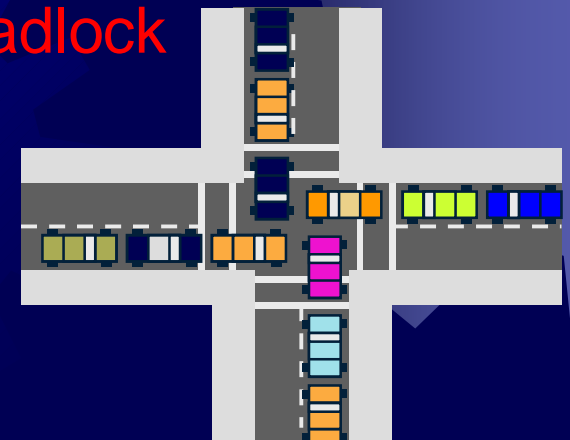
# Software Challenge - Technical Testing

- ❖ Simple Code Coverage Metrics Insufficient
  - e.g.) Just Tracking Statement or Branch Executions
- ❖ Need to Consider Other Code Executing in Parallel
  - Want to Test All Possible Instruction Interleavings
  - Otherwise, Code Would Not Be Fully Exercised
    - Especially Important to Check Interactions In Time

Race Condition

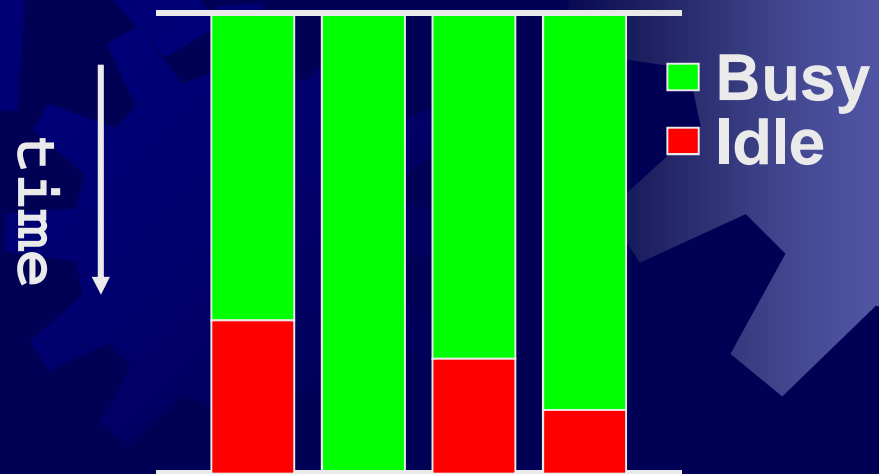


Deadlock



# Software Challenge - Technical Performance Profiling and Tuning

- ❖ Important to Know Which Code (sections) to Optimize
  - Concentrate on “Hot” Spots
- ❖ Harder in Parallel Because Must Consider:
  - Thread Creation & Synchronization Overhead
  - Communication to Computation Ratio
  - Thread Balancing

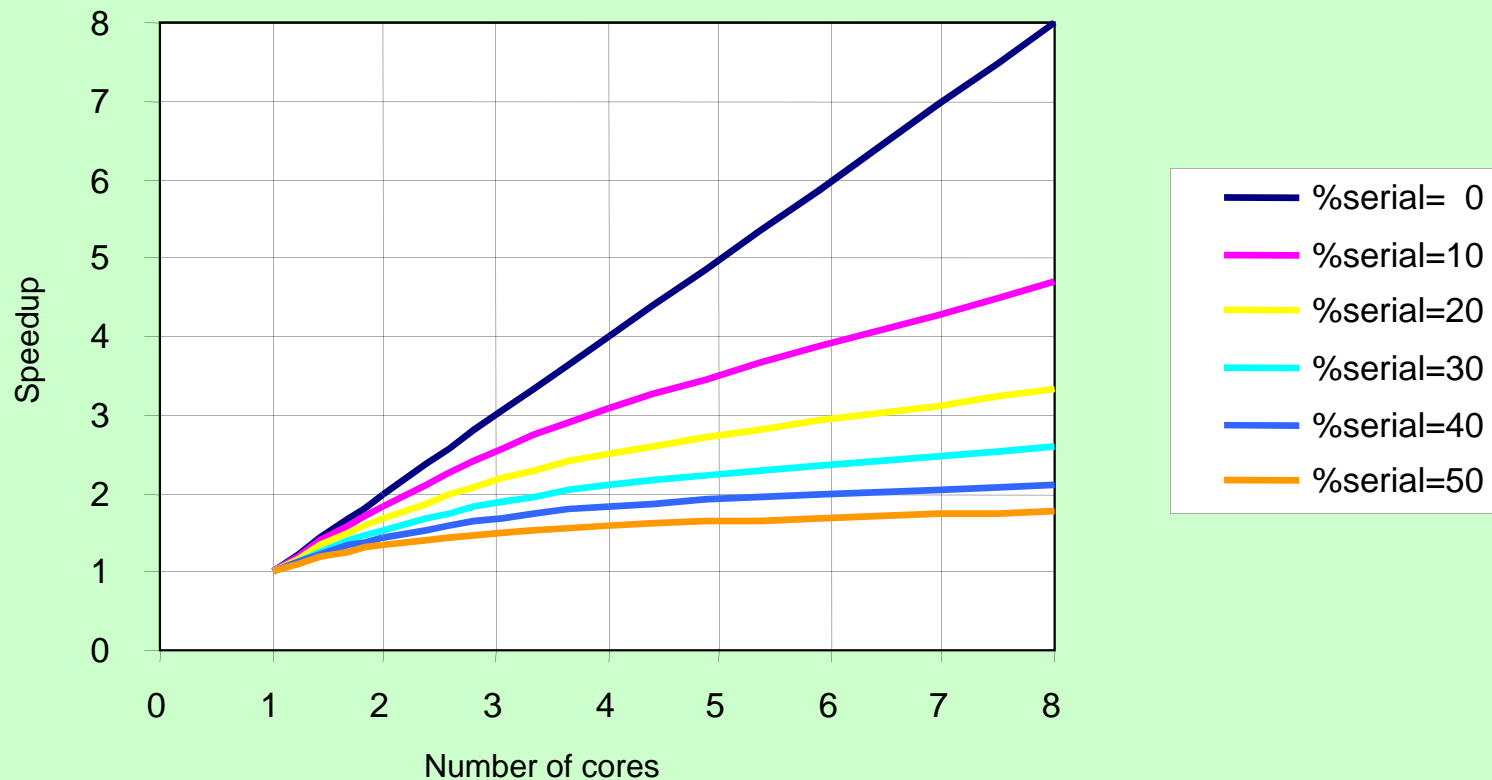


# Software Challenge - Technical

## Amdahl's Law

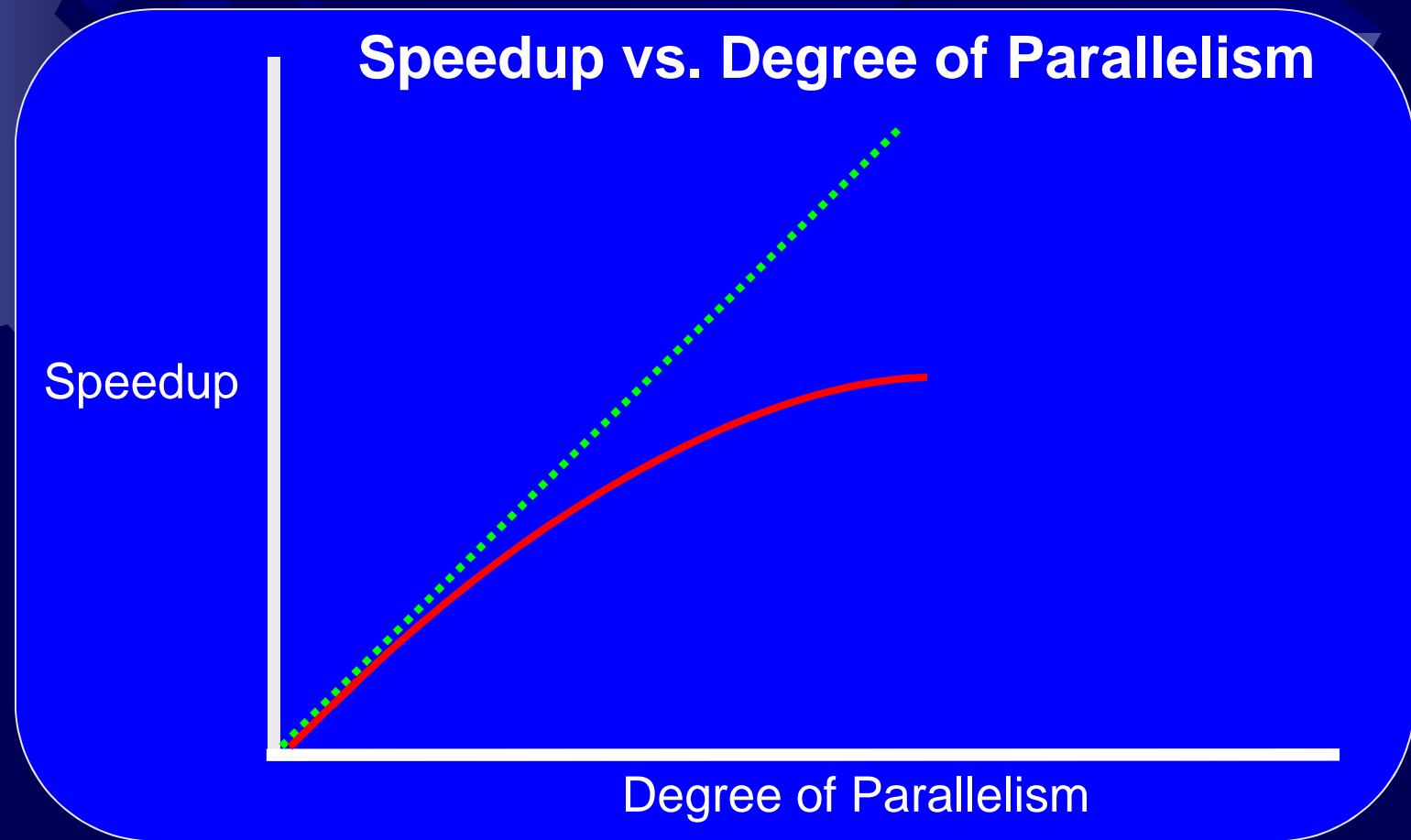
- ❖ Parallel Speedup is Limited by the Amount of Serial Code

Maximum Theoretical Speedup from Amdahl's Law



# Software Challenge - Technical Parallel Scaling

- ❖ Potentially Negative ROI due to Parallel Overhead





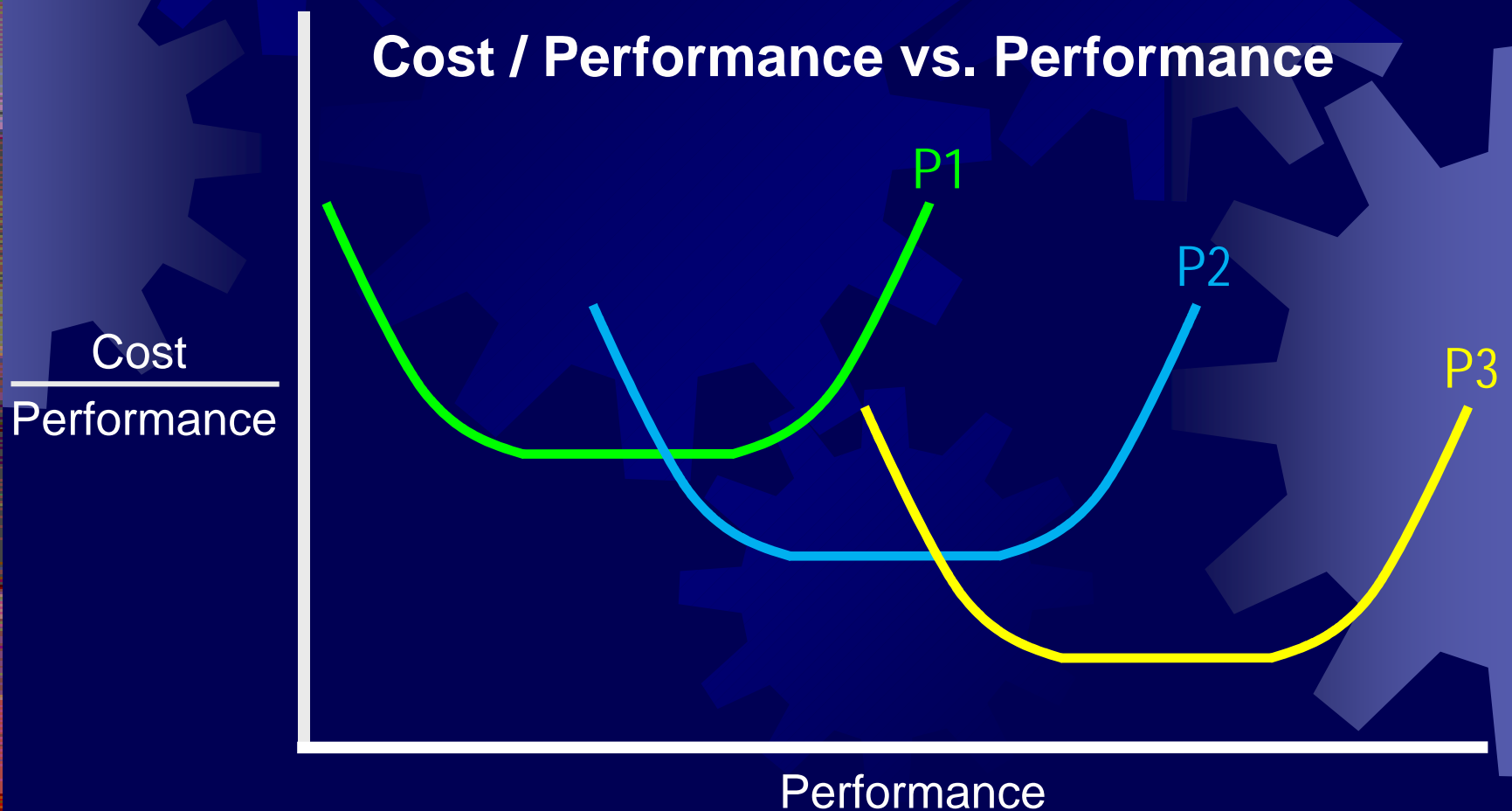
# Software Challenge - Technical Parallel Scaling

- ❖ Implications of Amdahl's Law:
  - Diminishing Marginal Rates of Return from Parallelism
    - It will be Hard to get good Parallel Scaling from SW
  - Eliminating Sequential Code is Important
    - Even Small Amounts of Serial Execution can Render a Parallel Machine Ineffective
- Applications That Lack Sufficient Parallelism Will Be Performance Dead Ends

# Software Challenge - Business

## Changing Technology Curves is Hard

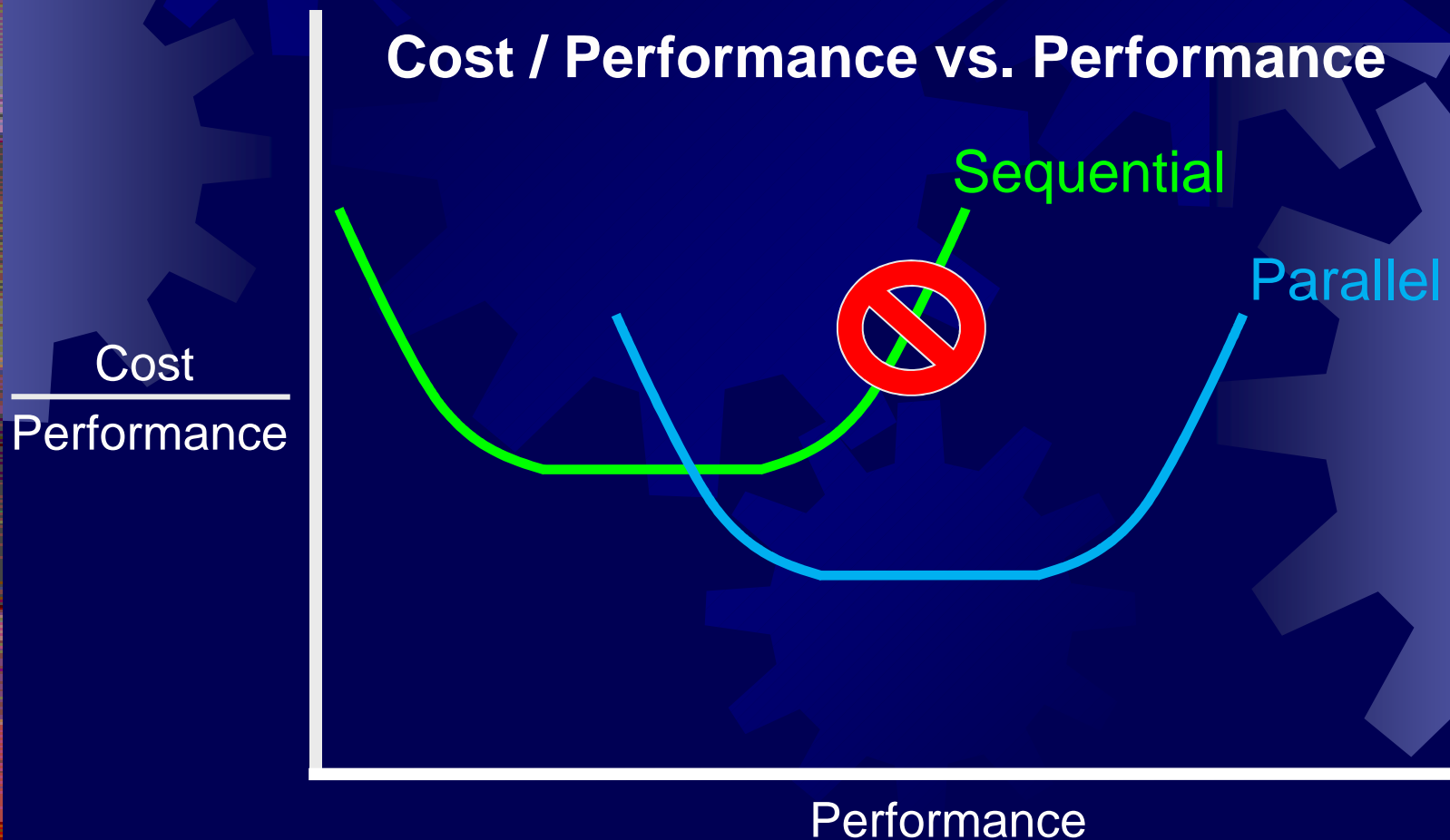
- ❖ New Technology Curves Generally Appear “Down Right”



# Software Challenge - Business

## Changing Technology Curves is Hard

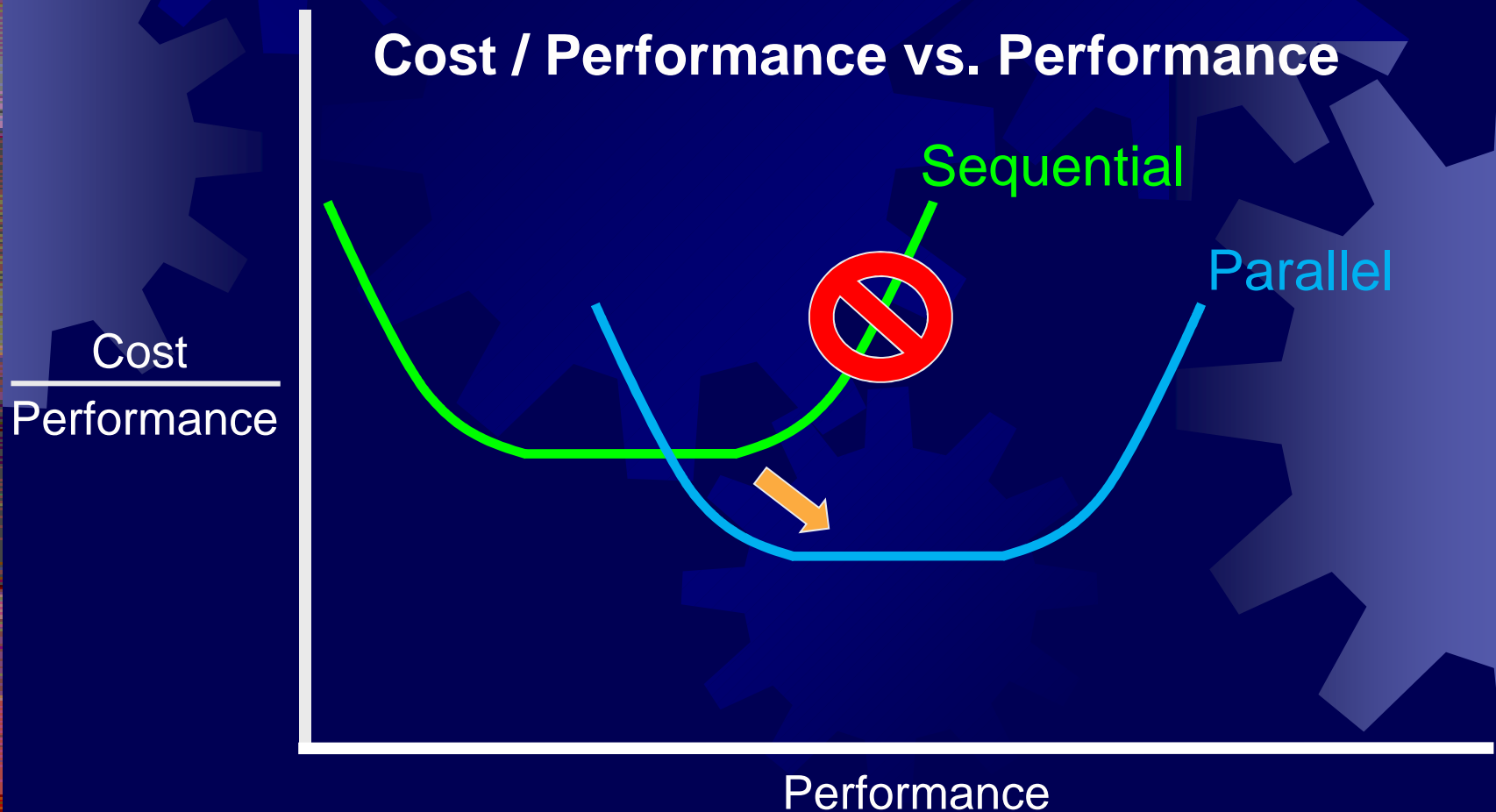
- ❖ Never Ride Technology Curve “Up” into Over-Utilization



# Software Challenge - Business

## Changing Technology Curves is Hard

- ❖ Change (“Hop Down Right”) to New Tech Curve Instead



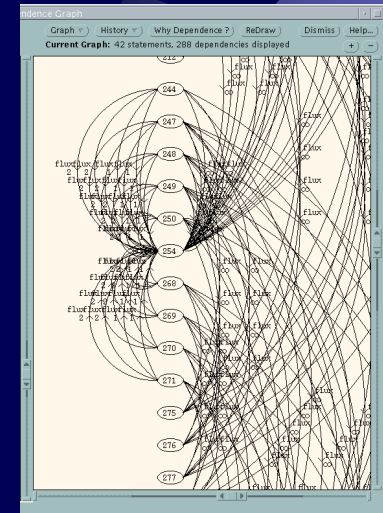


# Software Challenge - Business

## Changing Technology Curves is Hard

### ❖ Investment in Training and New Tools Required

- Learning Curve for Employees
- Entirely New SW Engineering Infrastructure
  - Design / Re-engineering
  - Debugging
  - Testing
  - Profiling
  - Scaling



### ❖ Legacy Code Needs to be Re-engineered for Parallelism



# Key Points

## Software Challenge

### ➤ Parallel Programming is Hard

- More Complex
- Lack of Tools
- New Type of Bugs
  - Race Conditions
  - Deadlocks
- Harder to Debug, Test, Profile, Tune, Scale

### ➤ Parallel Programming is a Software Challenge



# Outline

✦ Hardware Solution

✦ Software Challenge



✦ Opportunities

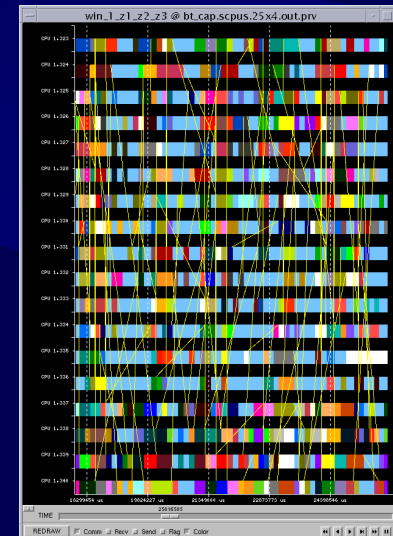
- Technical
- Business

# Opportunities - Technical

- ❖ Opportunity to Create New SW Engineering Infrastructure

- ❖ Better, Smarter Tools for

- Design / Re-engineering
    - Debugging
    - Testing
    - Profiling



- ❖ Opportunity to Re-Invent Entire SW Engineering Field

- ❖ Algorithms, Languages, Compilers, Processes.....

- ❖ Dawn of a New Era

- Second Chance (to get it right)

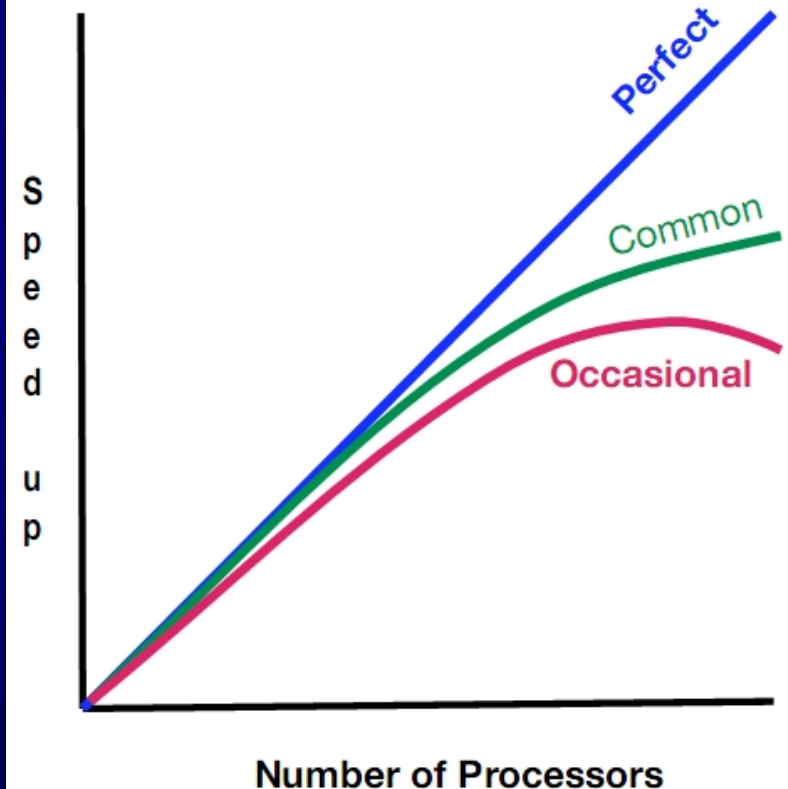
# Opportunities - Technical

- ❖ The Universe is Inherently Parallel
  - Natural Physical and Social / Work Processes
    - Weather, Galaxy Formation, Epidemics, Traffic Jam
- ❖ Can Leverage Unique Capabilities offered by Parallelism
- ❖ Add New Features via Separate Parallel Modules
  - Avoids Re-engineering of Old Module
  - More Functionality
  - No Increase in Wall Processing Times
- ❖ Speculative Computation
  - Precompute alternatives to Minimize Response Time
  - e.g.) Video Game Up / Down / Left / Right
  - More Responsive User Interfaces

# Opportunities - Technical

- ❖ (Yet) Undiscovered Technical Opportunities
- ❖ New Parallel Algorithms
- ❖ Super-Linear Speedups
  - Parallel Computer has N times more Memory
  - Larger % of SW can fit in upper Levels of Memory Hierarchy
  - “Divide and Conquer” leverages faster Mem
  - An Important Reason for using Parallel Computers

## Observed Speedup



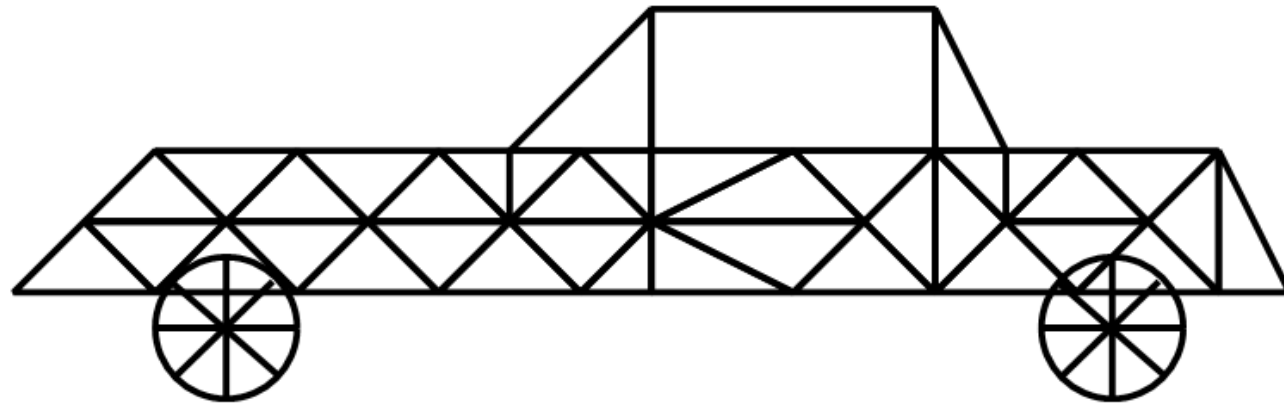
# Opportunities - Business

## High Performance Computing (HPC)

- ❖ Cloud and Parallel Processing Makes HPC Ubiquitous
- ❖ New Applications Become Possible
  - Personalized Drugs (Genetic & Molecular Profiling)
  - Stream Computing (Real-Time Analytics)
- ❖ Smarter Applications Become Possible
  - Virtual Assistants
- ❖ Efficiency Becomes Possible
  - High-Fidelity Simulations  
e.g.) Car Safety Tests



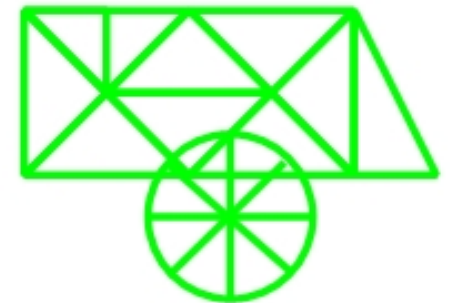
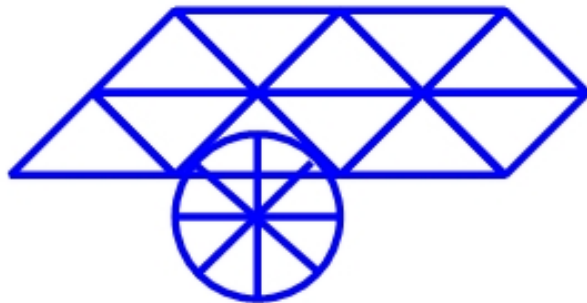
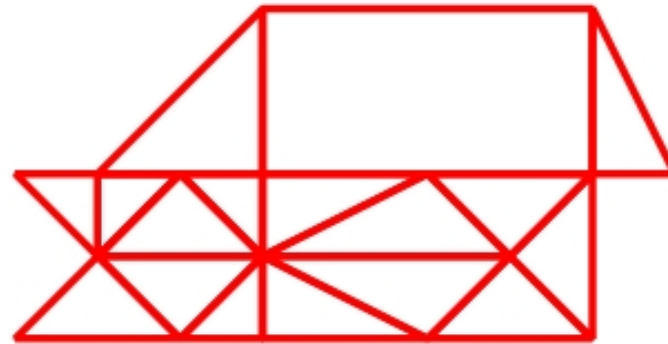




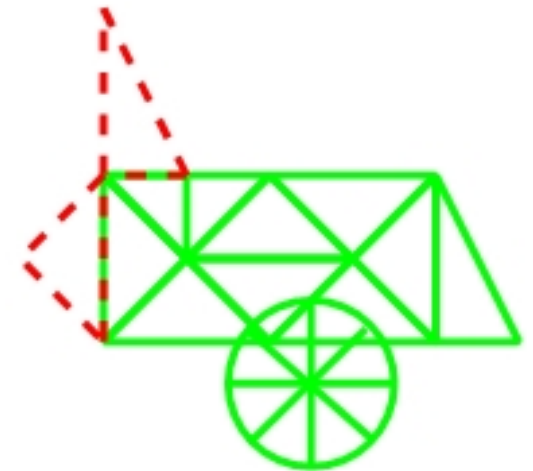
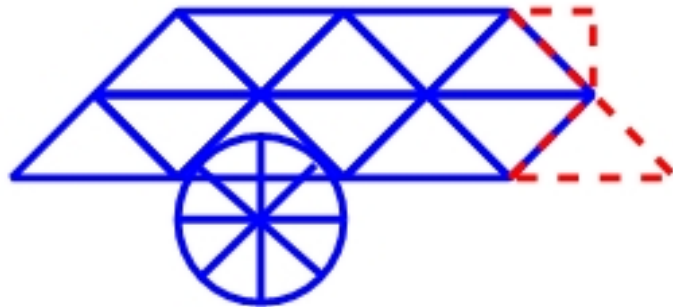
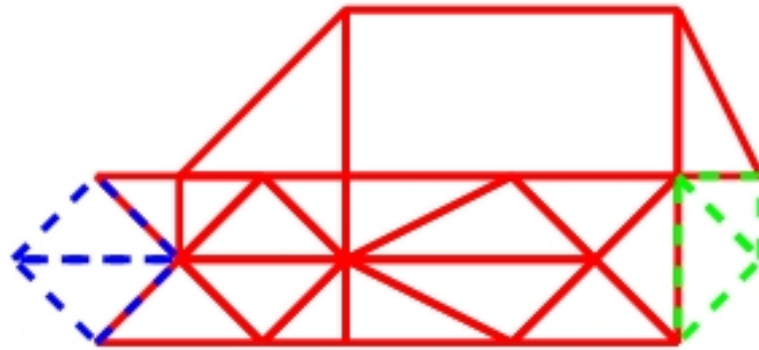
## Basic Serial Crash Simulation

- 1 For all elements
- 2     Read State(element), Properties(element),  
      Neighbor\_list(element)
- 3 For time=1 to end\_of\_simulation
- 4     For element = 1 to num\_elements
- 5         Compute State(element) for next time step,  
          based on previous state of element and its  
          neighbors, and on properties of element

# A Distributed Car

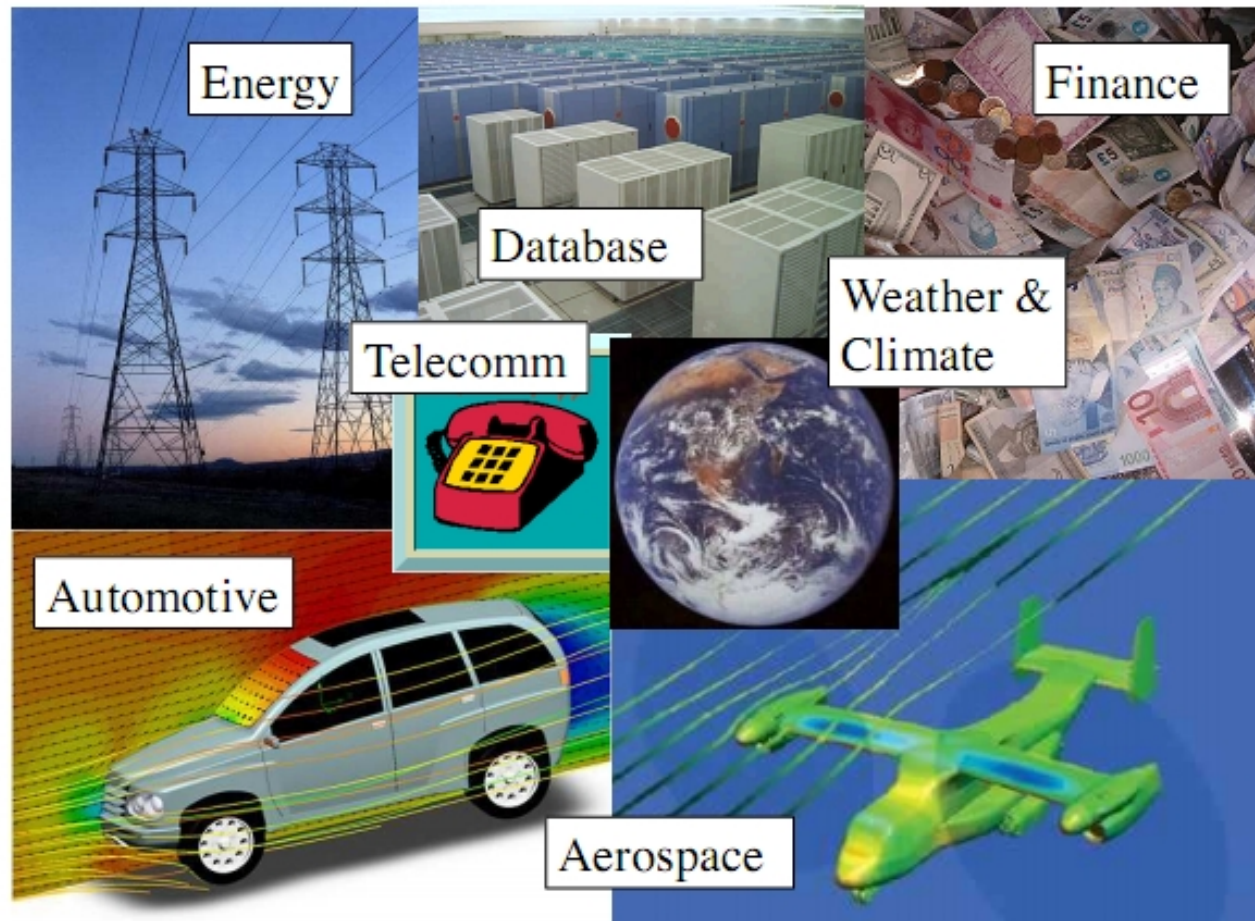


# Ghost Cells



# Opportunities - Business

## Who Uses Supercomputers?





# Opportunities - Business

## High Performance Computing (HPC)

- ❖ Grand Challenges Become Possible
- ❖ Grand Challenge was Defined by Wilson in 1987:
  - Fundamental Problem in Science or Engineering
  - Has Potentially Broad Economic and Scientific Impact
  - Could be Advanced with HPC Resources
- ❖ Grand Challenge 3T Goal:
  - 1 TeraFlop/second of Processor Power
  - 1 TeraByte of Main Memory
  - 1 TeraByte/second of I/O Bandwidth

# Opportunities - Business

## High Performance Computing (HPC)

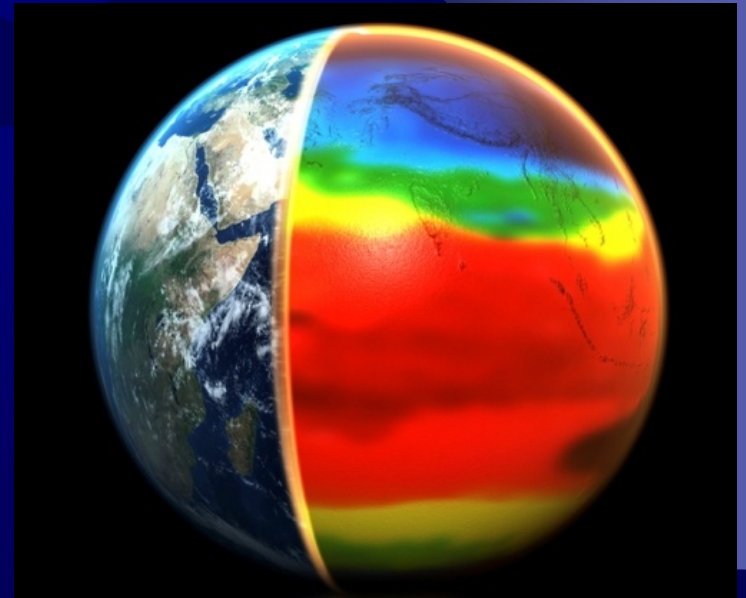
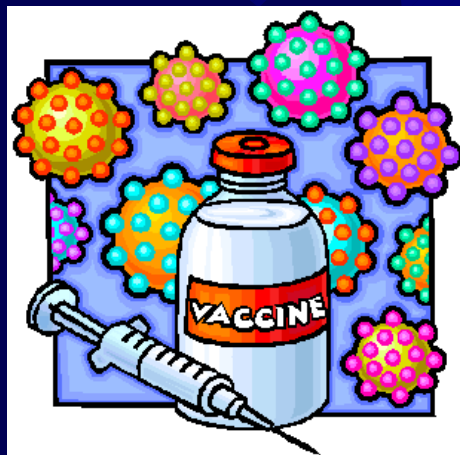
### ❖ Examples of Grand Challenges

- Data Mining and Fusion
- Hurricane Prediction
- Global Warming
- World Hunger
- Cure for Serious Diseases

REVIEW & OUTLOOK | JANUARY 8, 2010

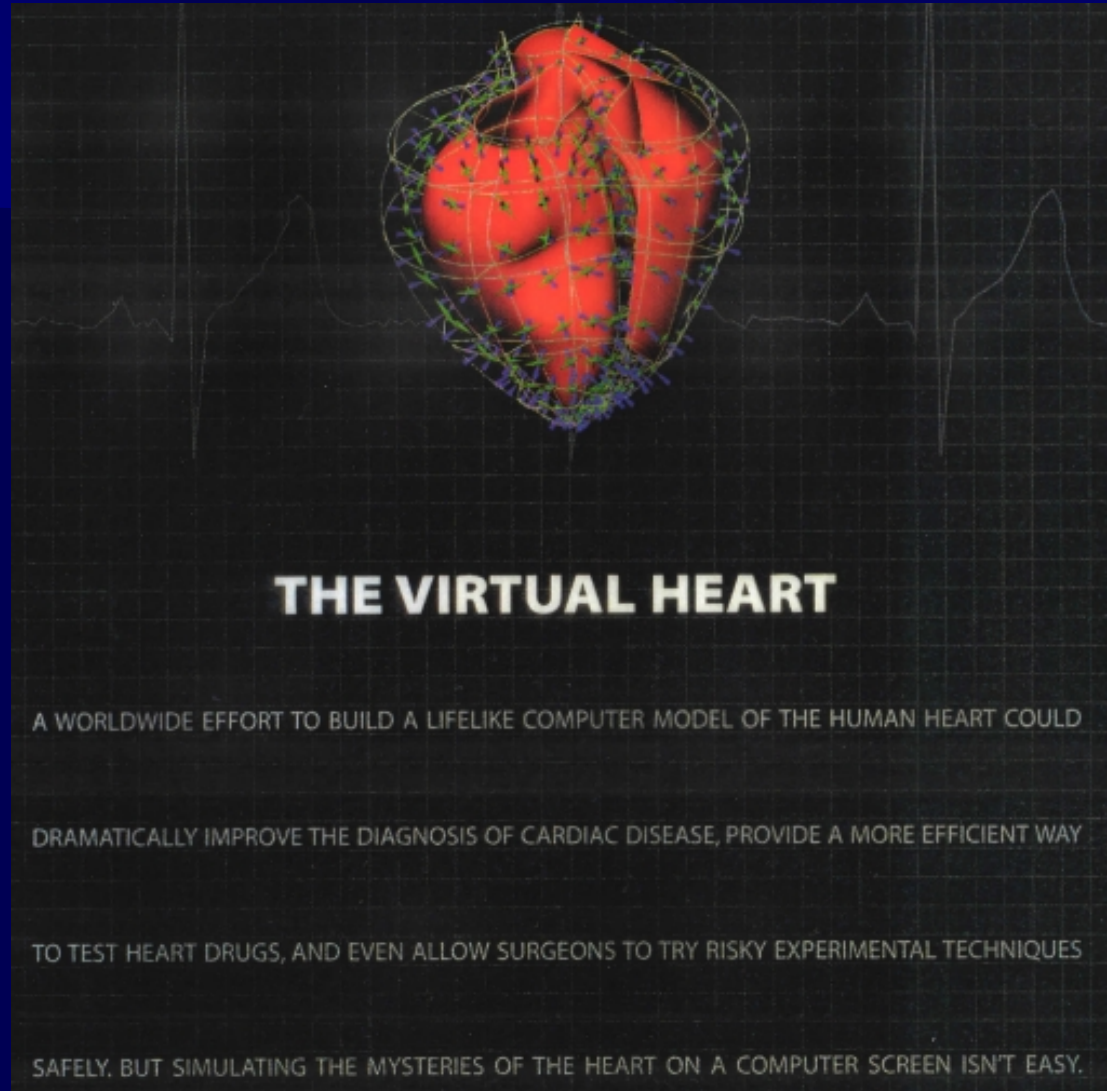
### 'A Failure to Connect the Dots'

*A lesson in the lack of bureaucratic intelligence.*



# Opportunities - Business

## New Capabilities to Benefit Mankind





# Opportunities - Business

## New Capabilities to “Entertain” Mankind ..... Maybe even “OutSmart” Mankind

### **NETWORKWORLD**

This story appeared on Network World at  
<http://www.networkworld.com/news/2010/021010-ibm-jeopardy-game.html>

## **IBM's Jeopardy-playing machine can now beat human contestants**

'Watson' may face Jeopardy public challenge within a year

By [Jon Brodtkin](#), Network World

February 10, 2010 04:58 PM ET

[IBM's](#) Jeopardy-playing supercomputer is now capable of beating human Jeopardy contestants on a regular basis, but has a ways to go before it takes on the likes of 74-time champion Ken Jennings.

IBM announced plans to build a computer that can win on Jeopardy last April, and expects to stage a public tournament involving human players and the machine within the next year or so.

### [IBM Supercomputer to Compete on Jeopardy](#)

The question-answering system, nicknamed "Watson", is already doing trial runs against people who have actually appeared on the Alex Trebek-hosted Jeopardy. Watson's competition includes people who qualified for the show but lost, people who appeared and won once, and people who appeared and won twice.

Sponsored by:

newsle



# Opportunities - Business Corporate & National Competiveness

## **IBM Reclaims Supercomputer Lead**

But stay tuned—supercomputers are getting faster,  
at an even faster rate

## **NASA Ames machine sets new speed record**

SILICON GRAPHICS PROVIDED 'COLUMBIA' SYSTEM

## **IBM Takes On Petaflop Barrier**

**N**EVER SAY THAT NOTHING good comes out of computer games. By harnessing a processor originally built for the upcoming Sony PlayStation 3, IBM is building a supercomputer that's expected to smash the petaflop barrier with a speed of 1,000 trillion calculations per second.

gress laid out \$35 million, is a base cluster that runs on the Linux operating system and uses IBM System x 3755 servers based on AMD Opteron technology. That system is slated to ship to the national lab next month.

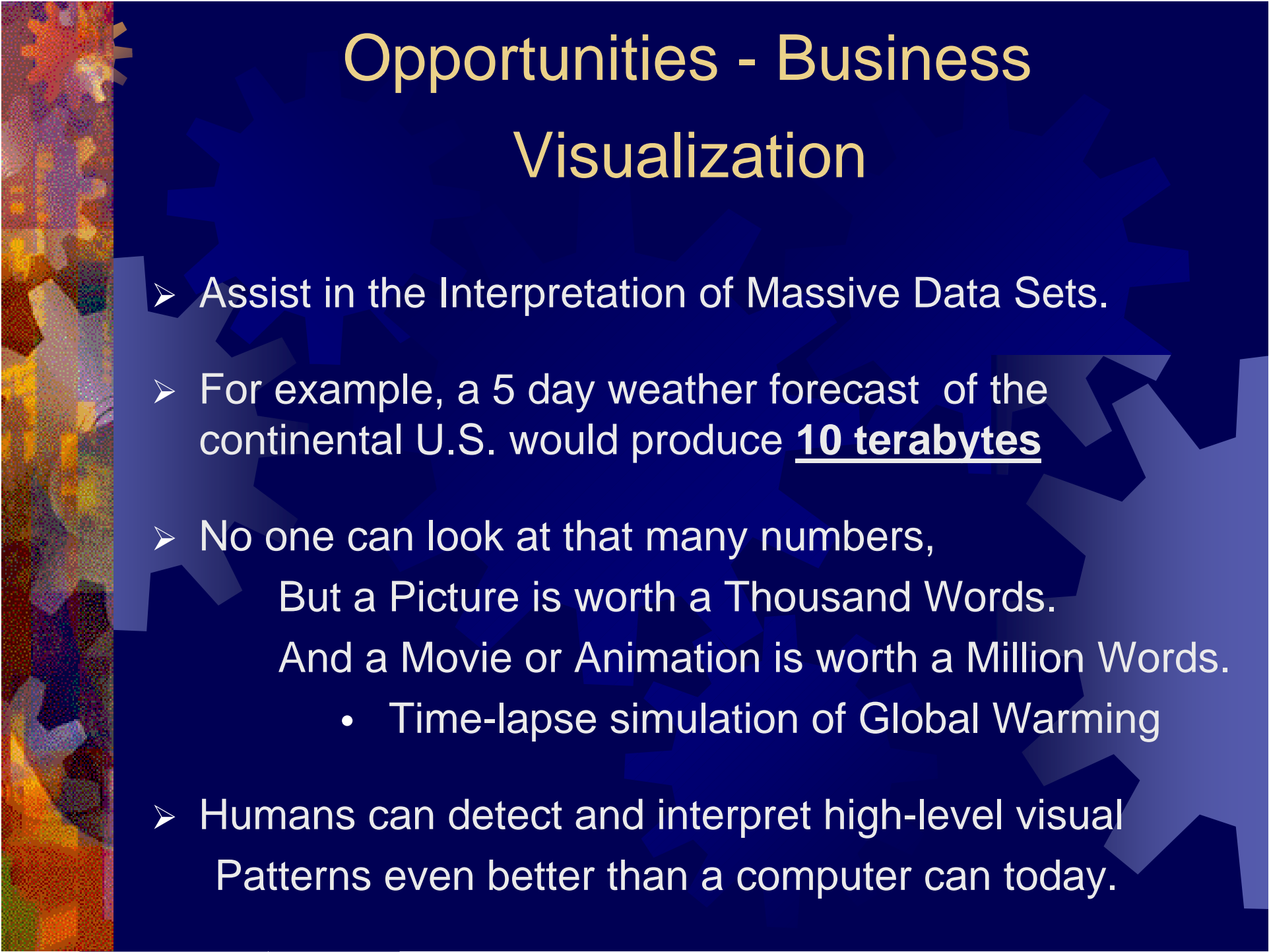
But the computer's real

The real challenge ahead lies in building software sophisticated enough to examine a calculation and decide which processor to assign to it, Addison Snell of IDC says.

IBM's Blue Gene/L is currently the fastest supercomputer in the

**1,000  
TRILLION**

The number of calculations per second Roadrunner will run



# Opportunities - Business Visualization

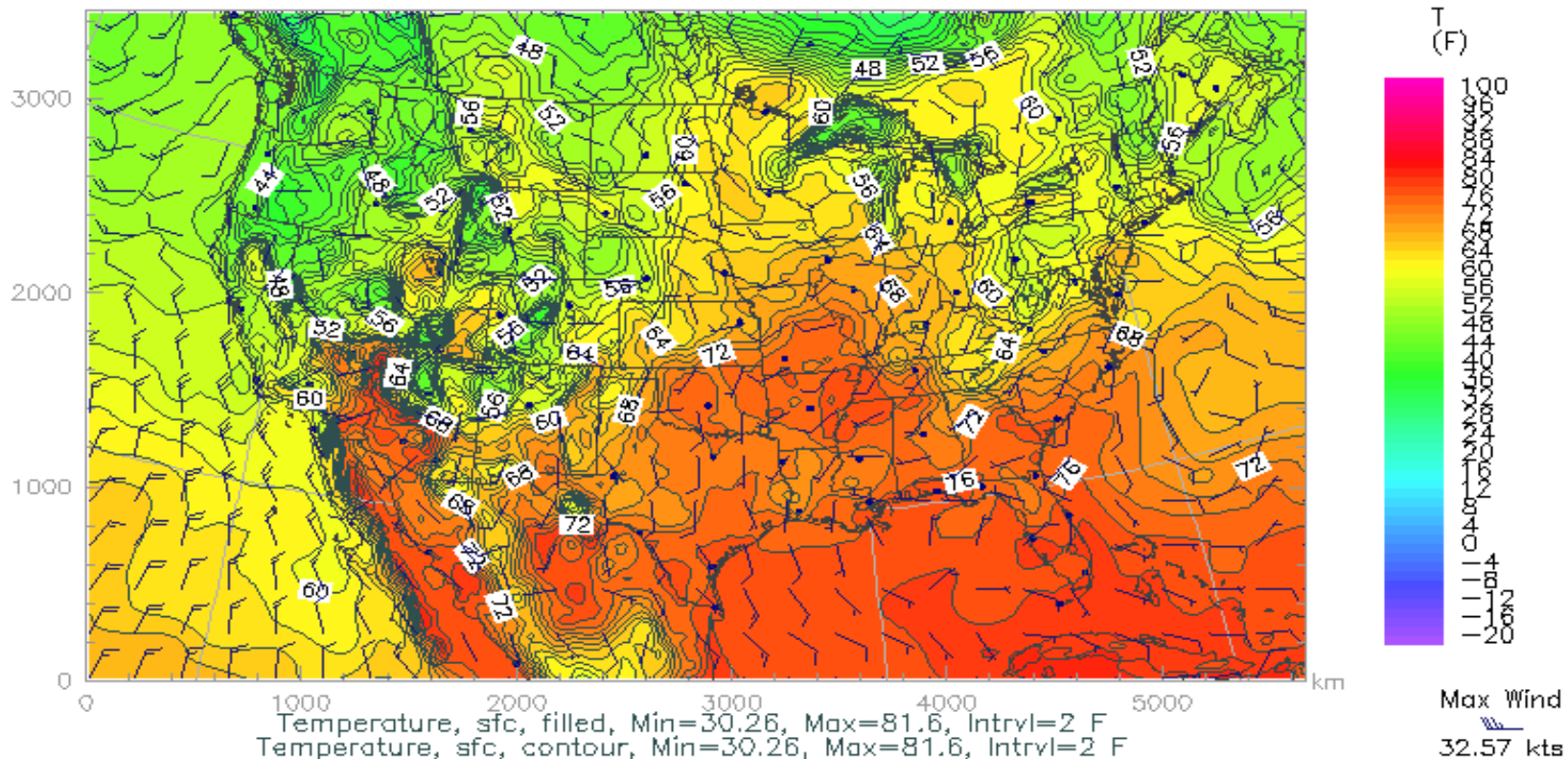
- Assist in the Interpretation of Massive Data Sets.
- For example, a 5 day weather forecast of the continental U.S. would produce 10 terabytes
- No one can look at that many numbers,  
But a Picture is worth a Thousand Words.  
And a Movie or Animation is worth a Million Words.
  - Time-lapse simulation of Global Warming
- Humans can detect and interpret high-level visual Patterns even better than a computer can today.



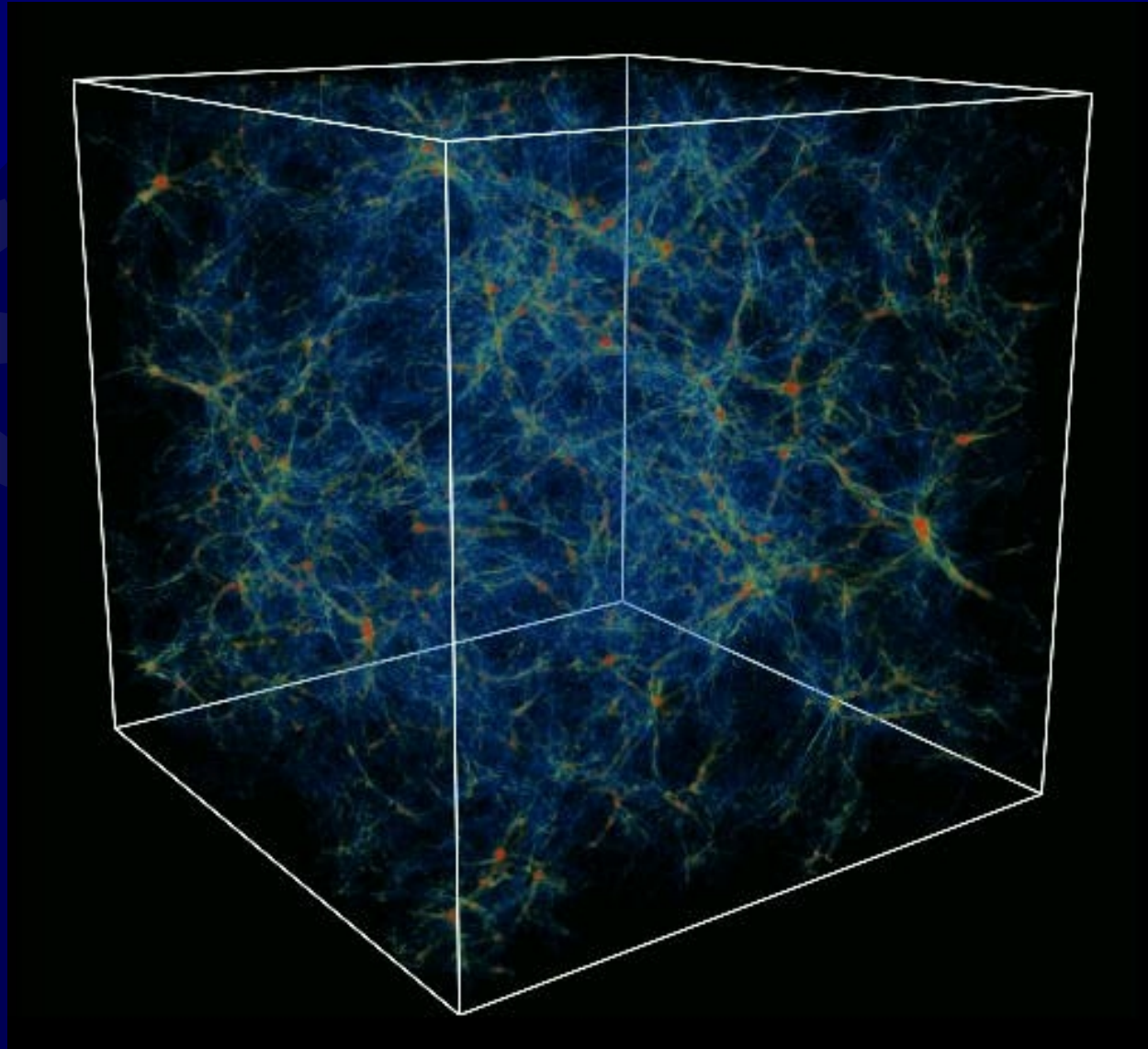
"If a picture is worth a thousand words, don't ever show Margaret any pictures!"

# Example Visualization: Weather Forecast

**Thu, 25 May 2006, 8 am CDT (13Z)**  
**Surface Temperature**



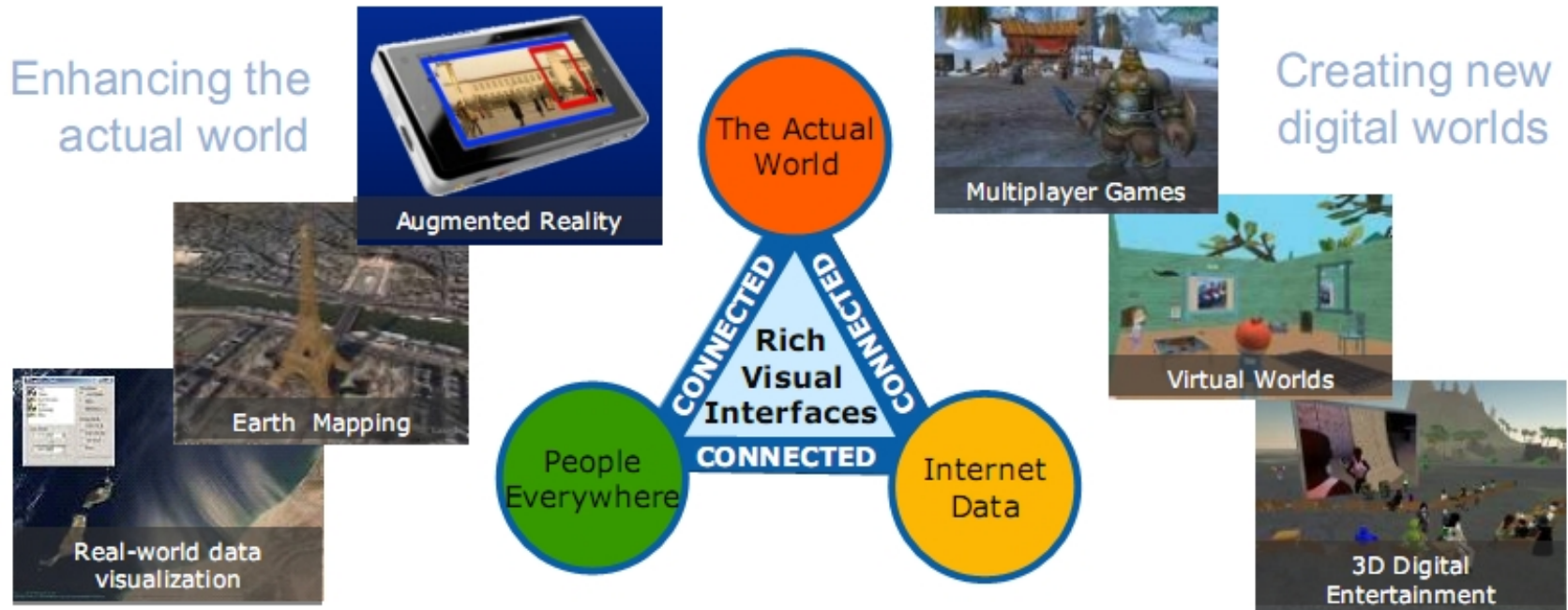
# Example Visualization: Structure of the Universe



# Opportunities - Business

## Next: Immersive Connected Experiences

Bringing the richness of VC to connected usage models such as social networking, collaboration, online gaming, & online retail.



Better content quality, social interaction

# Opportunities – Business Meeting People's Expectations About Multitasking Capabilities



Apple  
iPad  
Jan 2010

There's no multitasking. "Are you saying I can't listen to Pandora while writing a document? I can't have my Twitter app open at the same time as my browser? I can't have AIM open at the same time as my email? Are you kidding me? This alone guarantees that I will not buy this product," Gizmodo's Adam Frucci writes.

# The World is Moving Towards Parallel Processing On The Cloud



The Cloud Offers Even More Opportunities



# Opportunities - Business

## Ubiquitous HPC via Cloud

- ❖ HPC on Mobile Devices Possible
- ❖ “CloneCloud” (Berkeley Research Labs)
  - Clone Smartphone in Cloud
    - Off-load Compute-Intensive tasks
    - Conserve Mobile Device Battery Life
- ❖ “Fusion Render Cloud” for Mobile Gaming (AMD)
  - Cloud Computes Game Graphics and Compresses it
  - Mobile’s Computation Simplified to Decompression

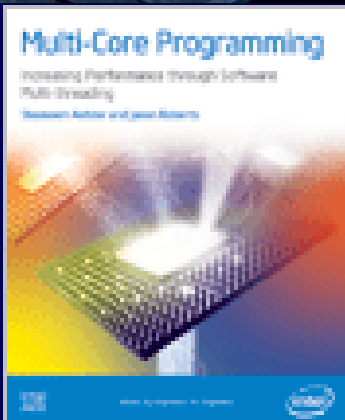


A bright sun with prominent rays is centered in a clear blue sky, surrounded by soft, wispy white clouds. The overall scene is bright and clear, symbolizing high performance and reaching the limits of the sky.

**Using Parallel Programming,  
You Can Reach The Clouds**

**And The Sky is The Limit !**

# Wondering about Parallel Programming?



Plan to Read Entire MCP Book, \*Except\* for:

- Skip Ch. 5
- Read only the first part of Ch. 8 (up to 219)
- Read only the first part of Ch. 10 (up to 265)
- Only Skim Ch. 11