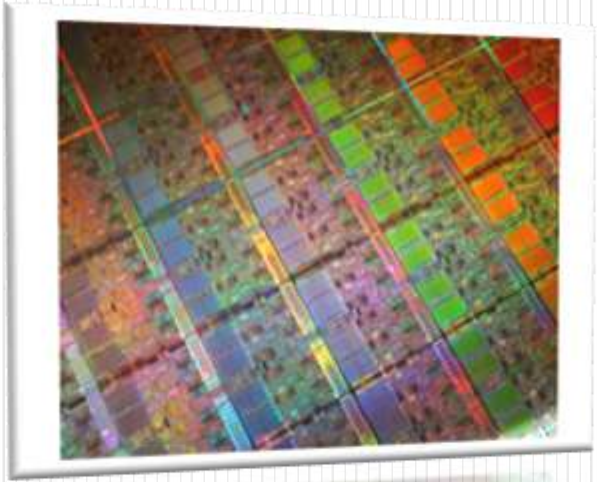
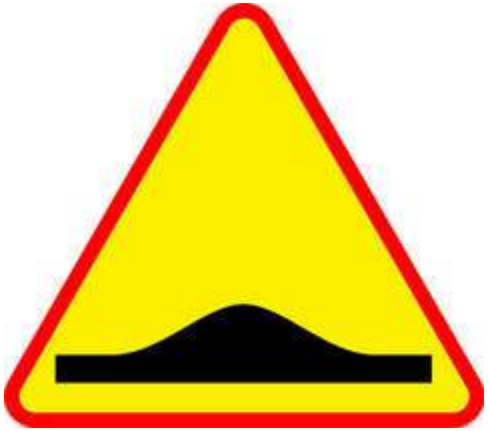
Multicore and Cloud Computing – Time to Start Afresh

James Larus
Microsoft Research

High Confidence Software and Systems Conference
May 18, 2009



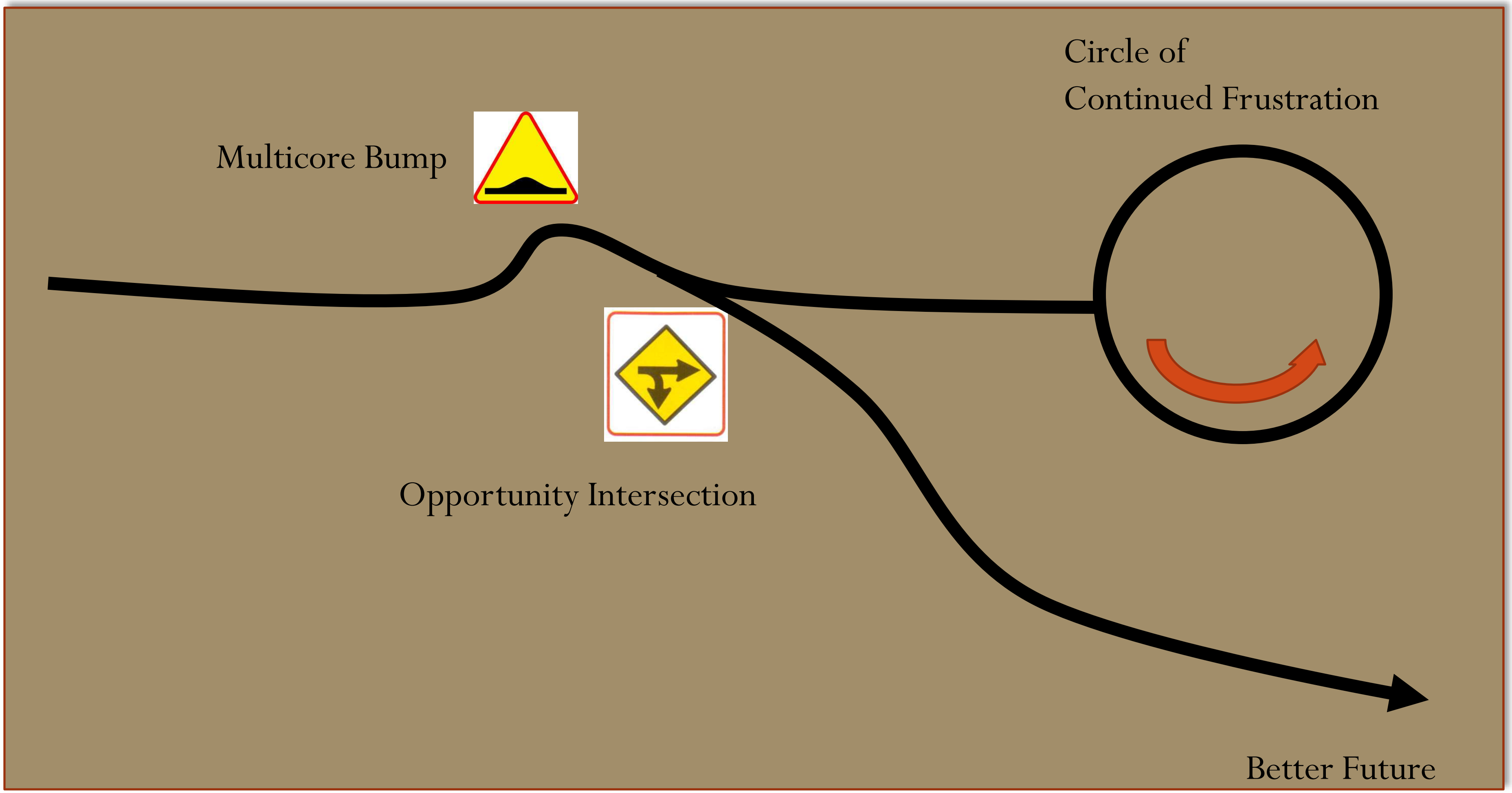
The Multicore Revolution



or

Microsoft Research

Computing Road Map

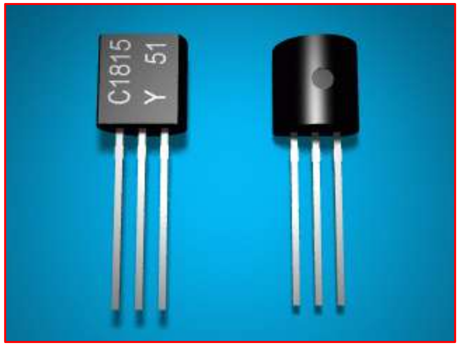


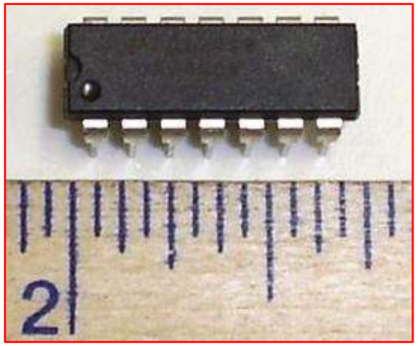
Microsoft Research

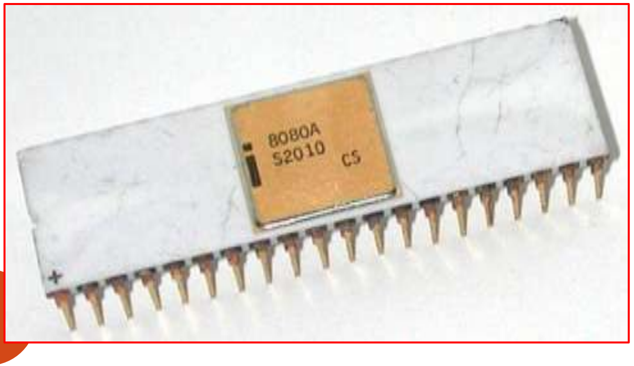
“A crisis is a terrible thing to waste”

- Multicore revolution will change how software is built and sold
- Disruptive change offers opportunity for improvement
- Seize this opportunity to build robust and reliable software
  - Ensure new software is better than old software

Si Is Destiny

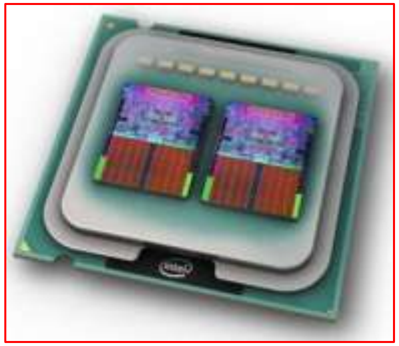






MICRO SOFT
intel

Multicore Destiny

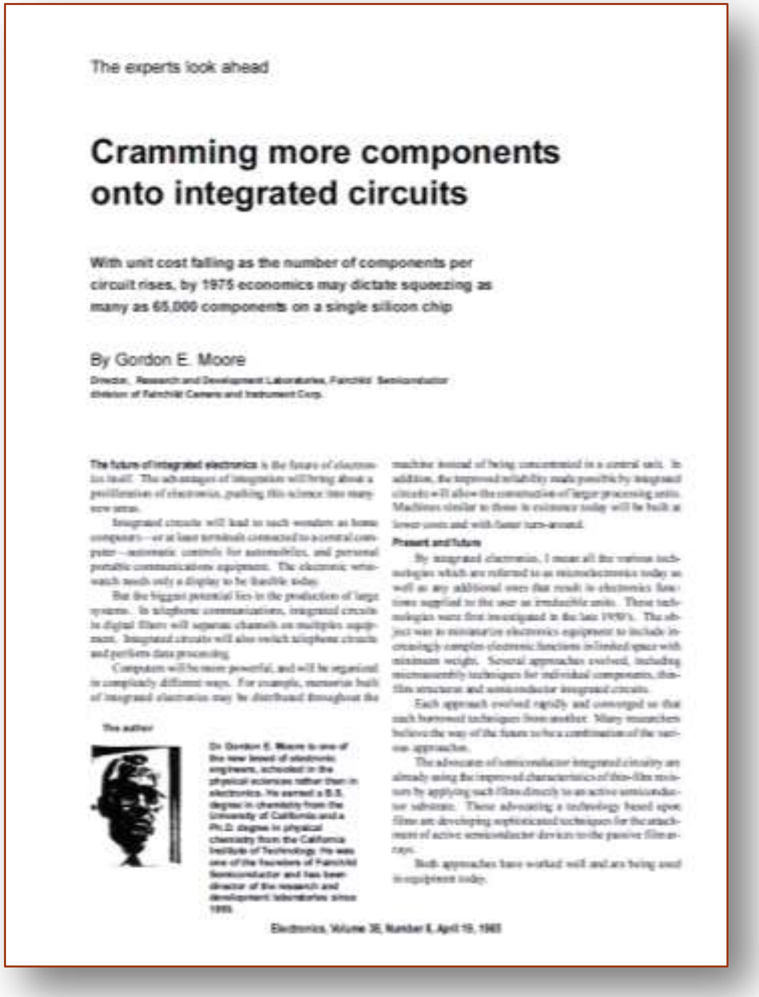


?

?

Microsoft® Research

Moore’s Law



The experts look ahead

Cramming more components onto integrated circuits

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip

By Gordon E. Moore
Director, Research and Development Laboratories, Fairchild Semiconductor division of Fairchild Camera and Instrument Corp.

The future of integrated electronics is the future of electronics itself. The advantages of integration will bring about a proliferation of electronics, pushing this science into many new areas.

Integrated circuits will lead to such wonders as home computers—or at least terminals connected to a central computer—automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.

But the biggest potential lies in the production of large systems. In telephone communications, integrated circuits in digital filters will separate channels on multiplex equipment. Integrated circuits will also switch telephone circuits and perform data processing.

Computers will be more powerful, and will be organized in completely different ways. For example, memories built of integrated electronics may be distributed throughout the machine instead of being concentrated in a central unit. In addition, the improved reliability made possible by integrated circuits will allow the construction of larger processing units. Machines similar to those in existence today will be built at lower costs and with faster turn-around.

Present and future

By integrated electronics, I mean all the various technologies which are referred to as microelectronics today as well as any additional ones that result in electronics functions supplied to the user as irreducible units. These technologies were first investigated in the late 1950's. The object was to miniaturize electronics equipment to include increasingly complex electronic functions in limited space with minimum weight. Several approaches evolved, including microassembly techniques for individual components, thin-film structures and semiconductor integrated circuits.

Each approach evolved rapidly and converged so that each borrowed techniques from another. Many researchers believe the way of the future to be a combination of the various approaches.

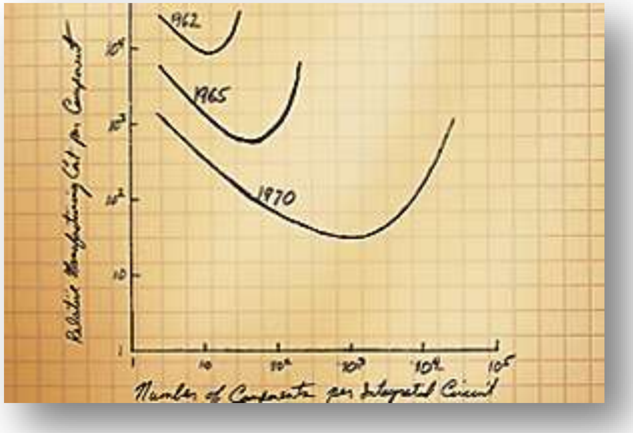
The advocates of semiconductor integrated circuitry are already using the improved characteristics of thin-film resistors by applying such films directly to an active semiconductor substrate. Those advocating a technology based upon films are developing sophisticated techniques for the attachment of active semiconductor devices to the passive film arrays.

Both approaches have worked well and are being used in equipment today.

The author

Dr. Gordon E. Moore is one of the new breed of electronic engineers, schooled in the physical sciences rather than in electronics. He earned a B.S. degree in chemistry from the University of California and a Ph.D. degree in physical chemistry from the California Institute of Technology. He was one of the founders of Fairchild Semiconductor and has been director of the research and development laboratories since 1959.

Electronics, Volume 38, Number 8, April 19, 1965



40%/yr improvement in transistor density ⇒ doubling every other year

Microsoft® Research

Moore’s Law Enforced

Intel® 8080 processor
Introduced 1974
Initial clock speed

2 MHz

Number of transistors

4,500

Manufacturing technology

6µ

1974

1,500x

18,000x

133x

Quad-Core Intel® Xeon® processor (Penryn)
Dual-Core Intel® Xeon® processor (Penryn)
Quad-Core Intel® Core™2 Extreme processor (Penryn)
Introduced 2007
Initial clock speed

> 3 GHz

Number of transistors

820,000,000

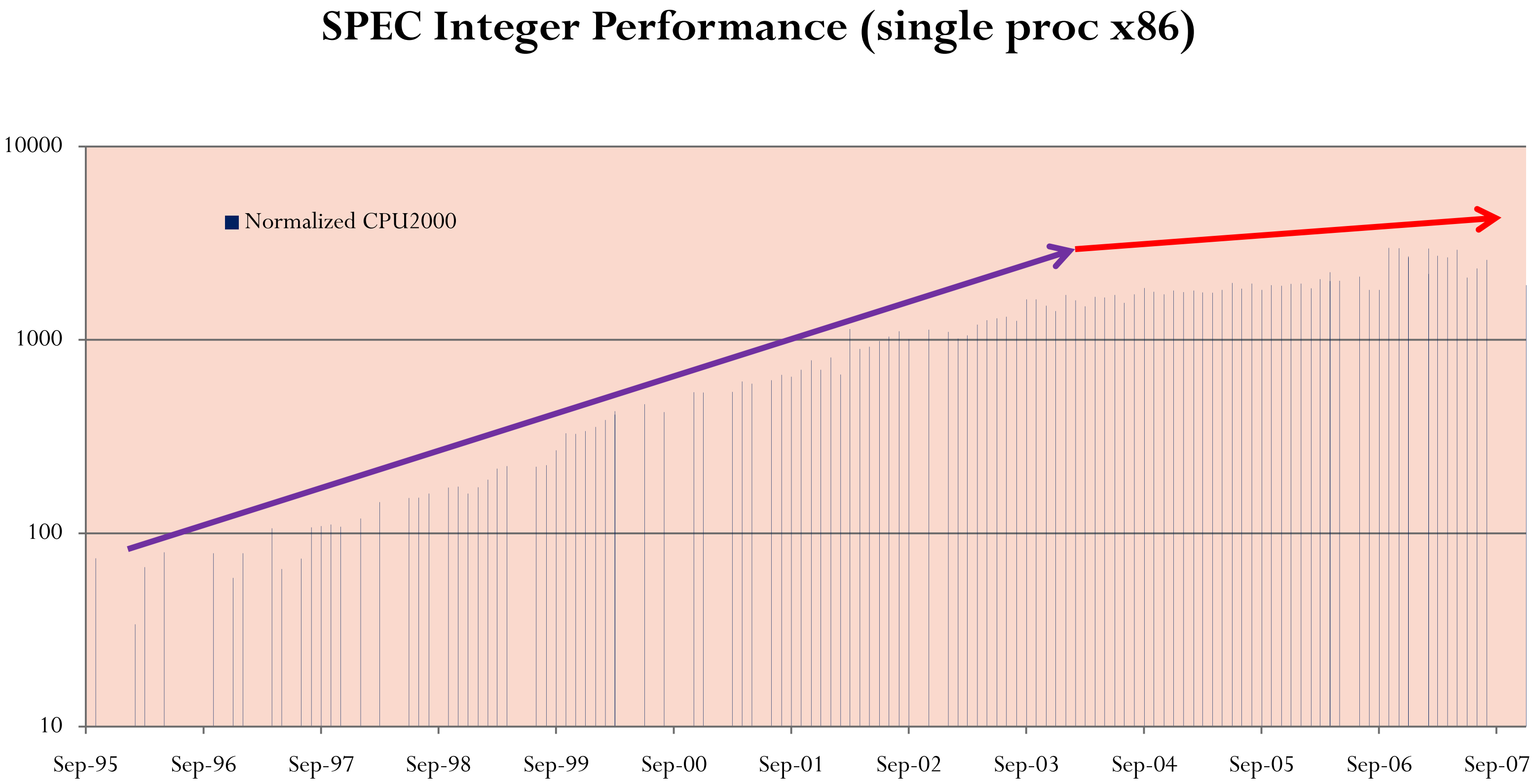
Manufacturing technology

45nm

2003

Intel, “Evolution of a Revolution,” http://download.intel.com

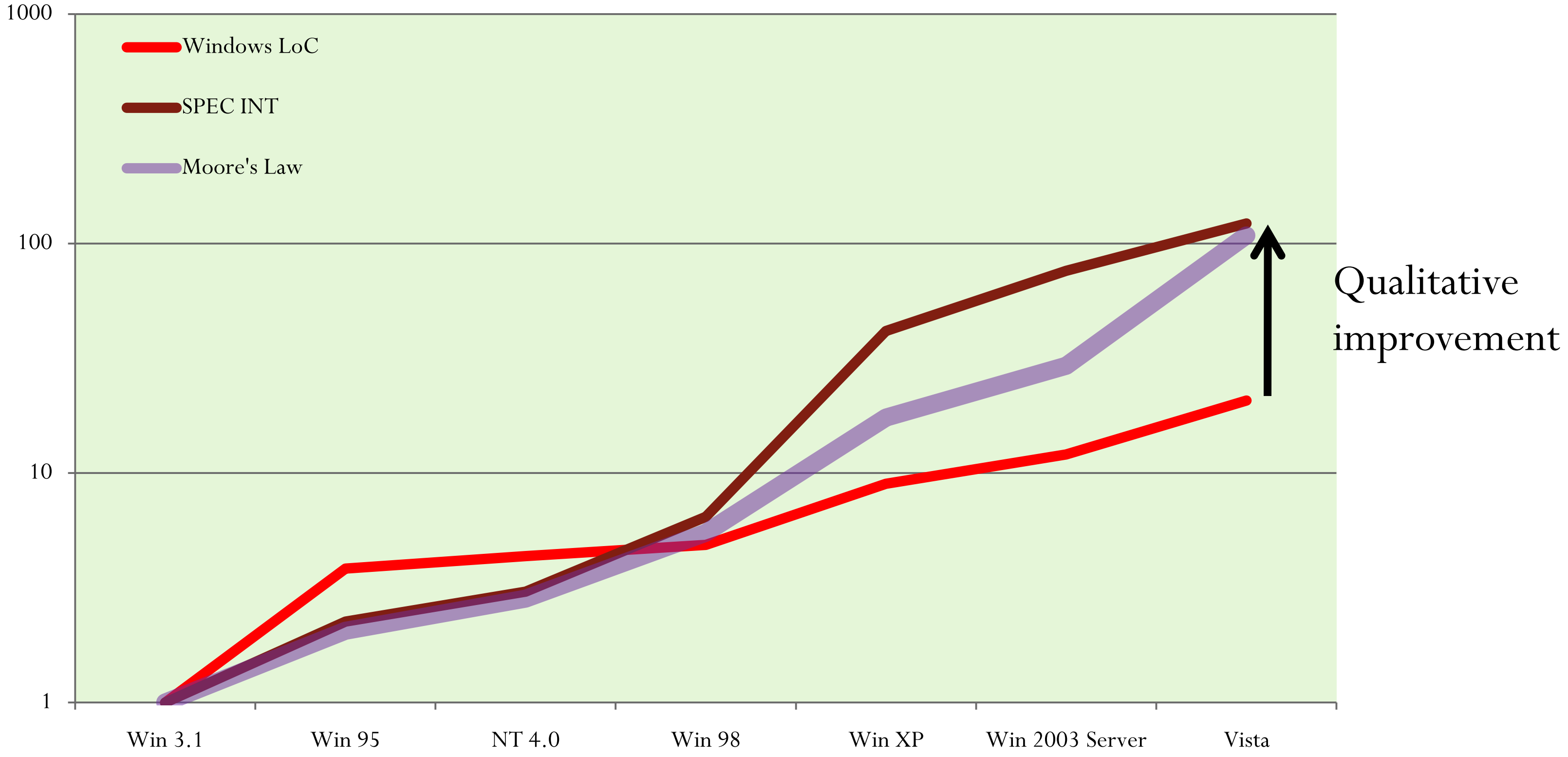
Moore’s Dividend



Outline

- Where was Moore's Dividend spent?
  - Software size
  - Software functionality
  - Programming complexity
- Is parallel computing a plausible successor?
- Parallel computing models
- Impact on computing

ΔCode Size < ΔProcessor Speed



Wikipedia estimates of LoC. Does not measure code shipped to customers.
SPEC normalized between SPEC95 and SPEC2000.

Microsoft Research

Where Moore's Dividend Was Spent

- Processor performance consumed by changes in:
  - Software size
  - Software functionality
  - Programming complexity

Microsoft Research

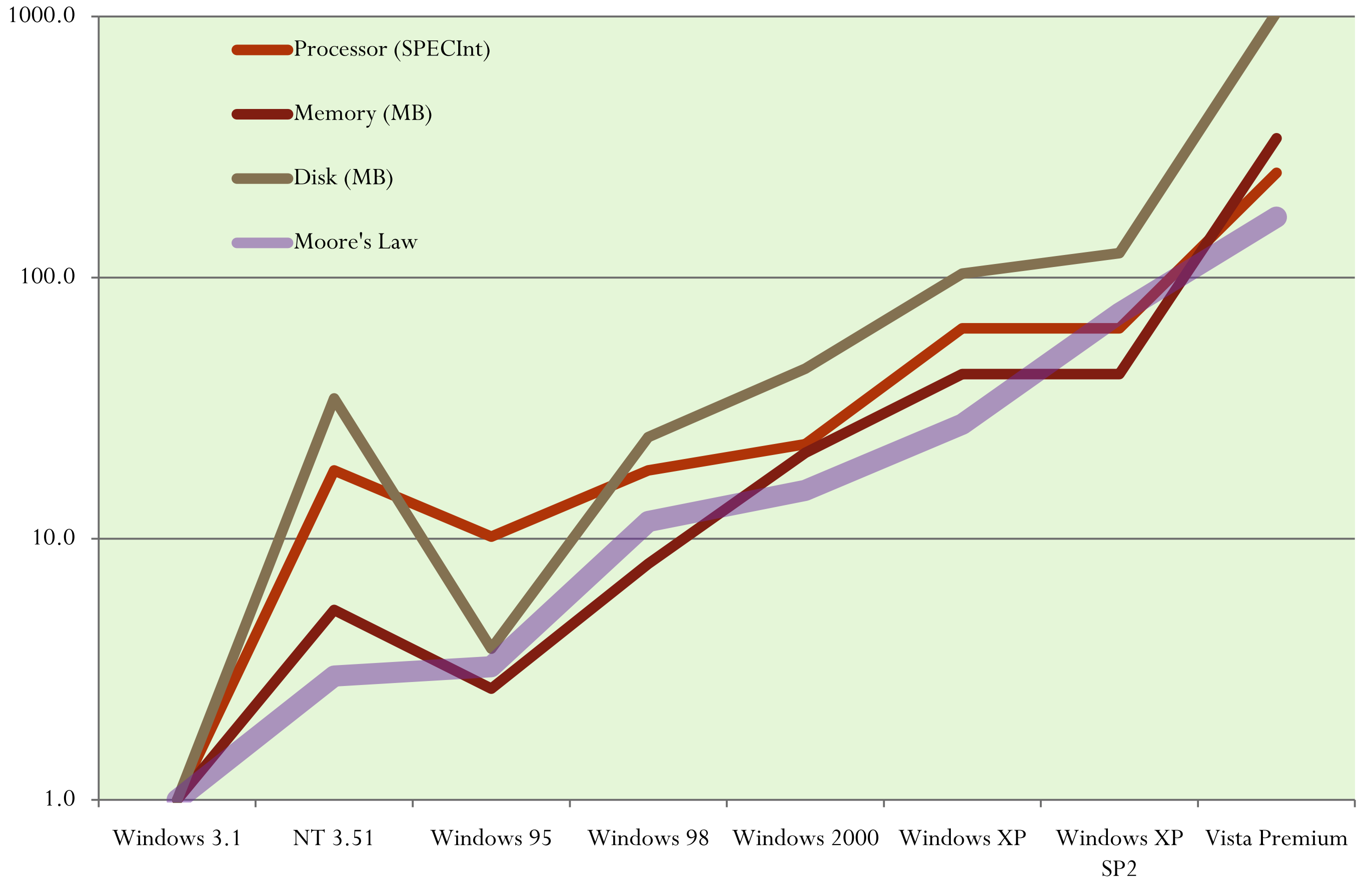
Expectations Evolve Since 1981

- 1 bit display
- 25 lines of 80 chars (4K)
- 16-640k memory
- Console
- stdio.h
- Single task, single address space
- No protection
- etc.

- 24 bit display
- 1280x1024 (64M)
- 1-4GB memory
- GUI
- Window system
- Multi-tasking, virtual address space
- Sophisticated security
- etc.

Microsoft Research

Recommended Windows Configurations



Microsoft Research

Legacy Compatibility

- Features monotonically increases
  - Office user uses 10% of features
  - Everyone uses a different 10% and 100% used
- Legacy compatibility sets floor

	Relative to WinXP						
	Size Increase		New Software		Legacy Code		
	Files	Lines	New Files	Lines added or churned	Files untouched	Edited Files	Original Lines
Win 2k3	1.43	1.42	1.11	1.13	0.73	0.93	0.78
Vista	1.80	1.46	1.07	1.03	0.80	1.00	0.94

Improvement Has Performance Cost

- Improvements are pervasive
  - Abstract model for many needs becomes less efficient
  - Generality precludes optimization
- Example: print spooling
  - Security, notification – 1.5-4x
  - Color management, better text handling – 2x
  - Resolution
    - 300*300 dip @ 1bit → 600*600 @ 24bits (1MB → 96MB)
  - Memory latency and bandwidth

Microsoft Research

Where Moore's Dividend Was Spent

- Processor performance consumed by changes in:
  - Software size
  - Software functionality
  - Programming complexity

Microsoft Research

Increased Abstraction

- High-level programming languages
  - Object-oriented (C++, Java, C#)
  - Interpreted (VB, Perl, Python, Ruby, etc.)
- Rich, abstract libraries
  - C++ Standard Template Library (STL)
  - Java class libraries
  - .NET platform
- Domain-specific language/systems
  - Ruby on Rails
    - RoR = 1/3 PHP < Java < C

Less Program Optimization

- Increased performance and memory size dulls programmers' edge
  - Gates changed "READY" to "OK" in Altair Basic to save 5 bytes
- Little understanding of processor performance models
  - Who really understands cache behavior?
- Increasing reliance on compiler optimization
  - Uniformly "good" quality
  - Sometimes 10-100x off hand-written code
- Performance is not an abstraction
  - Cuts across software abstractions
  - Think globally, act locally

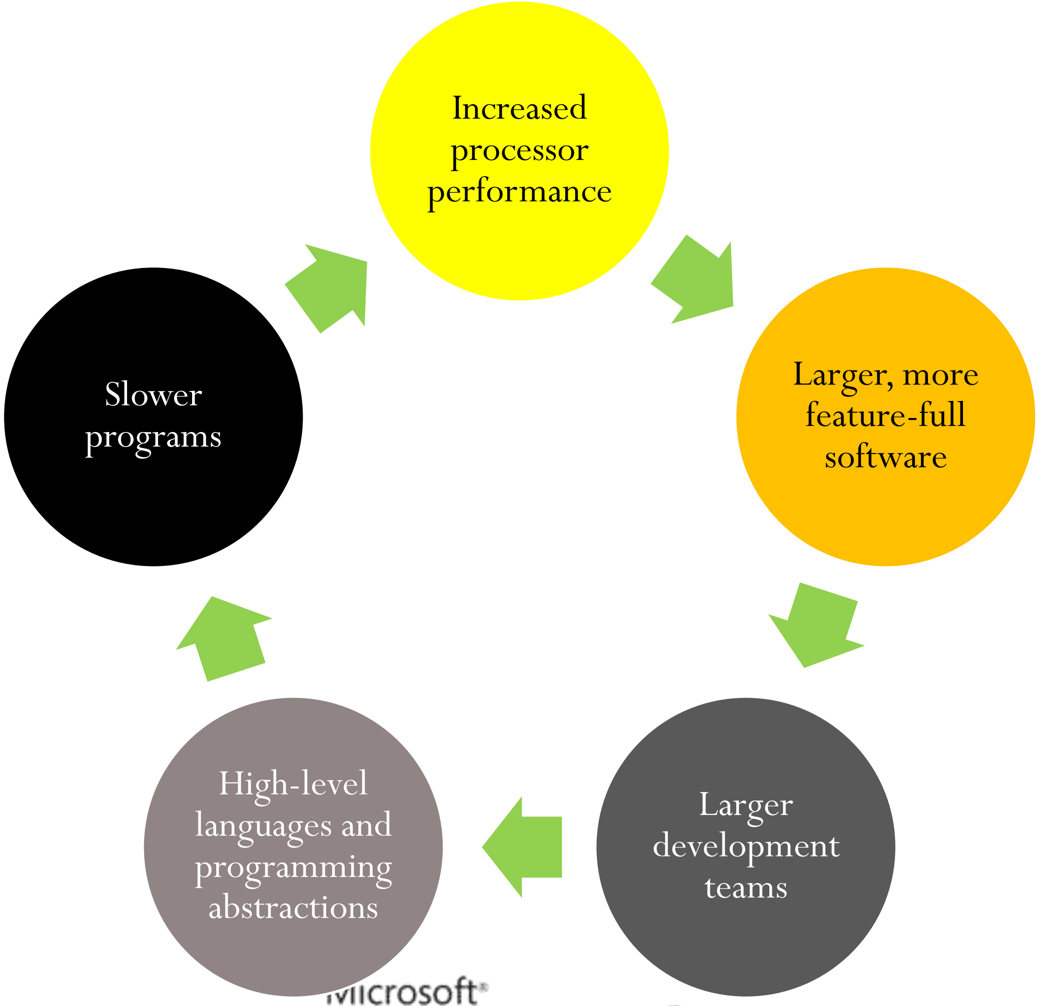
This is not bad!

- Increased abstraction improves productivity and enables richer functionality
- Without abstraction, modern software is beyond human comprehension
  - SAP Business Suite is 319 million LoC

OS	MLoC
Red Hat Linux 7.1	30
Debian 3.0	104
Debian 4.0	283
Mac Os X 10.4	86
Windows XP	40
Windows Vista	50

Source: Wikipedia.org

Software Development, c. 1950 – 2005



Increased processor performance

Larger, more feature-full software

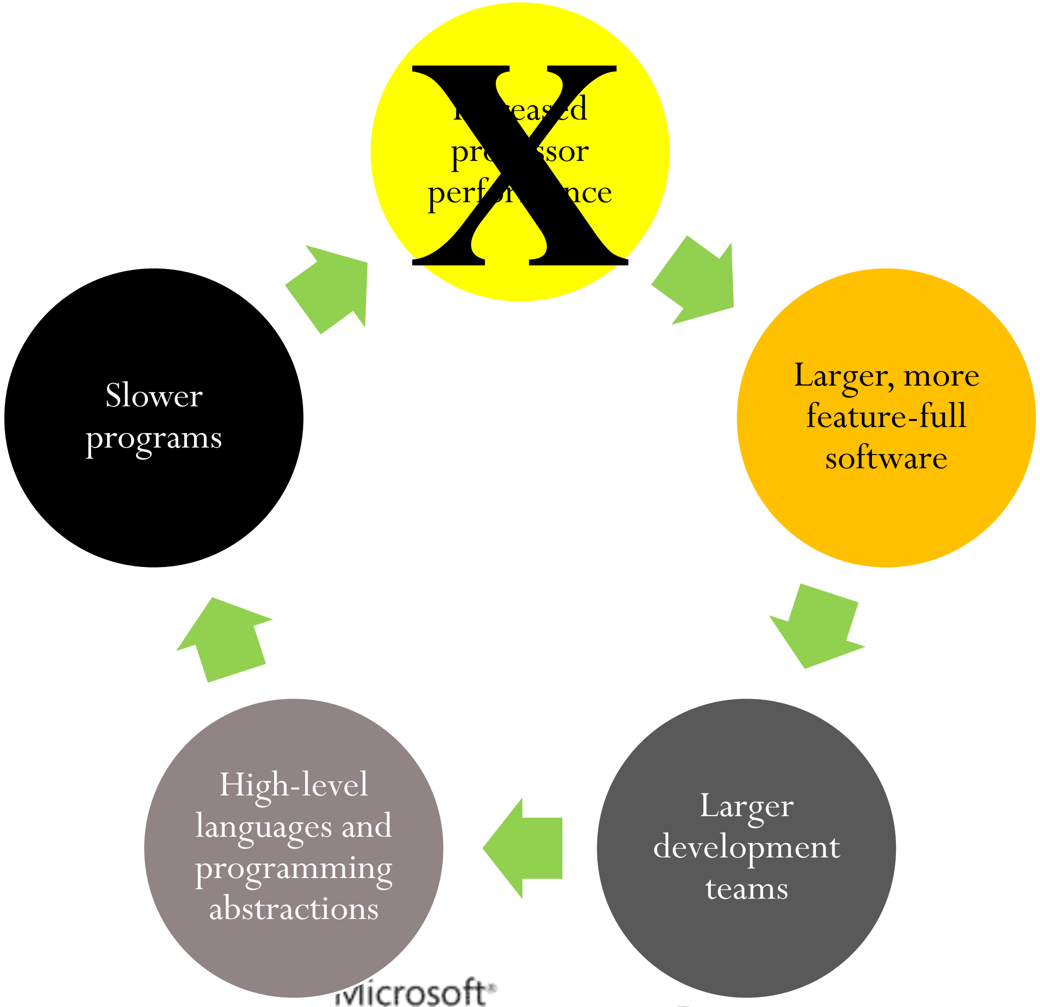
Larger development teams

High-level languages and programming abstractions

Slower programs

Microsoft Research

Software Development, RIP 2005?


X
Increased processor performance
Larger, more feature-full software
Larger development teams
High-level languages and programming abstractions
Slower programs

Microsoft Research

Outline

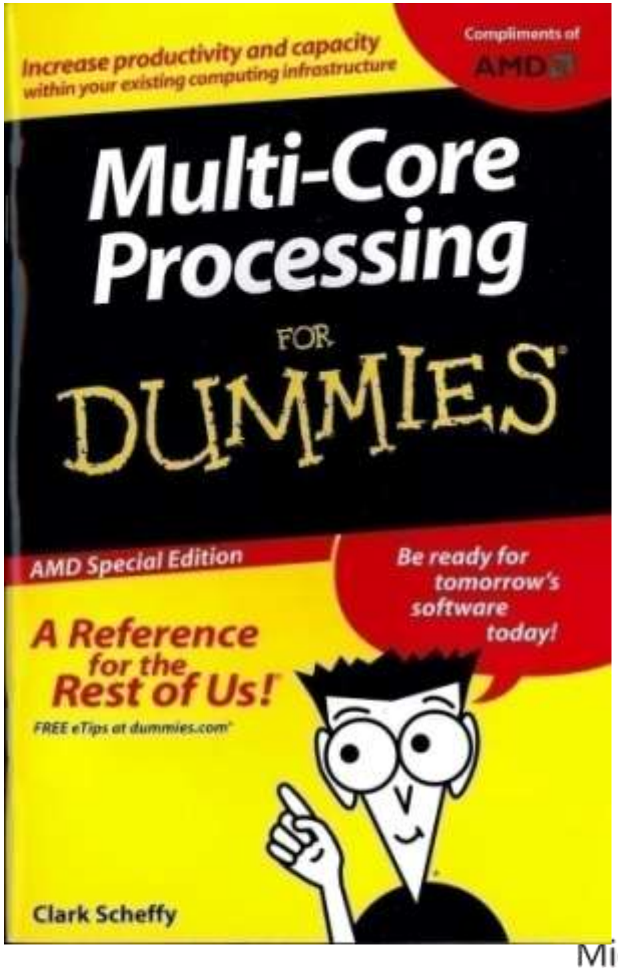
- Where Moore's Dividend was spent?
- Is parallel computing a plausible successor?
- New parallel computing models
- Impact on computing

Can Multicore Supplant Moore's Dividend?

- Double cores instead of increasing speed
- NO, at least without major innovation
  - Sequential code
  - Lack of parallel algorithms
  - Difficult programming
  - Few abstractions

Microsoft® Research

Some Confusion Out There



Increase productivity and capacity within your existing computing infrastructure

Compliments of AMD

Multi-Core Processing FOR DUMMIES

AMD Special Edition

Be ready for tomorrow's software today!

A Reference for the Rest of Us!

FREE eTips at dummies.com

Clark Scheffy

Help applications run more smoothly and with less frustration

Increase your computer's response time

Multi-core processors provide capabilities beyond what is possible with single-core processors. In fact, they represent the next step in processor design, offering significant gains in performance at a time when software is ready to take advantage of it. And with AMD's revolutionary Direct Connect Architecture, AMD multi-core technology helps eliminate the bottlenecks caused by the front-side bus in legacy system architectures.

Whether you're a home user or an IT manager, using multi-core technology from AMD can improve performance and decrease downtime.

Discover how to:

Experience better system performance

Use multi-core to benefit the enterprise and the digital home

Protect your computing investments

THE DUMMIES WAY

Explanations in plain English

"Get in, get out" information

Icons and other navigational aids

Top ten lists

A dash of humor and fun

Get smart! @ www.dummies.com

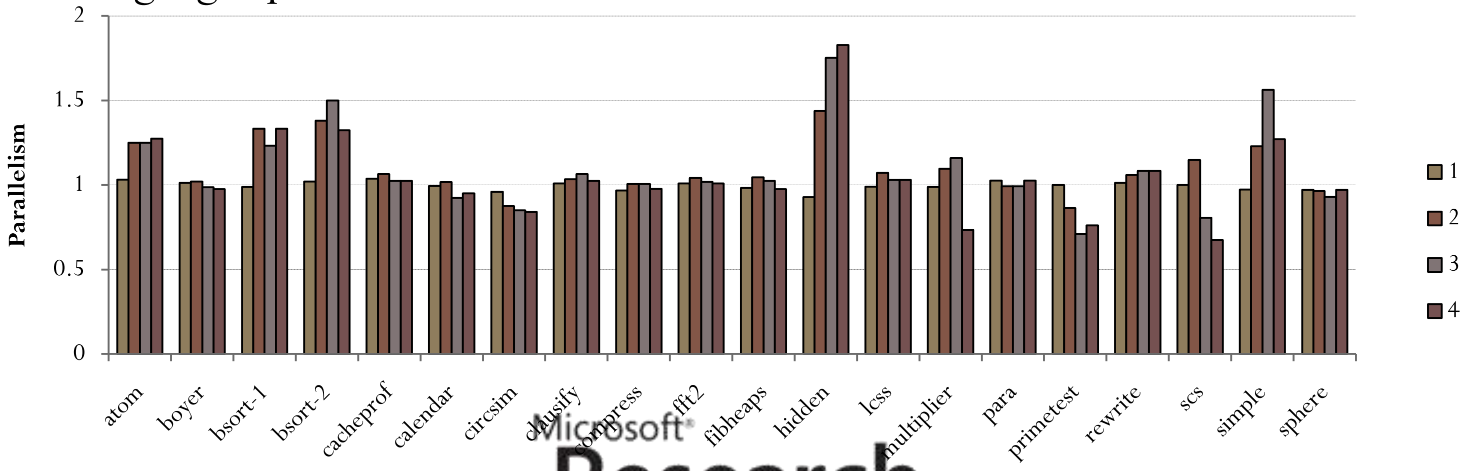
- ✓ Find listings of all our books
- ✓ Choose from many different subject categories
- ✓ Sign up for eTips at etips.dummies.com

ISBN: 0-470-04607-X
Part Number: 41344A
Not resaleable

For Dummies®
A Branded Imprint of
WILEY

Sequential Code

- Existing code is sequential
  - <u>Series</u> of decisions/actions
  - Difficult to change execution model
- Failed parallel compiler effort in ‘80s-’90s
  - Compiler cannot change fundamental programming model
- Failed instruction-level parallelism in 90’s-00’s
  - Dynamic mechanisms cannot find more than 2–4x parallelism
- Artifact of problems & thinking
  - Not language specific



Harris & Singh [ICFP 07]

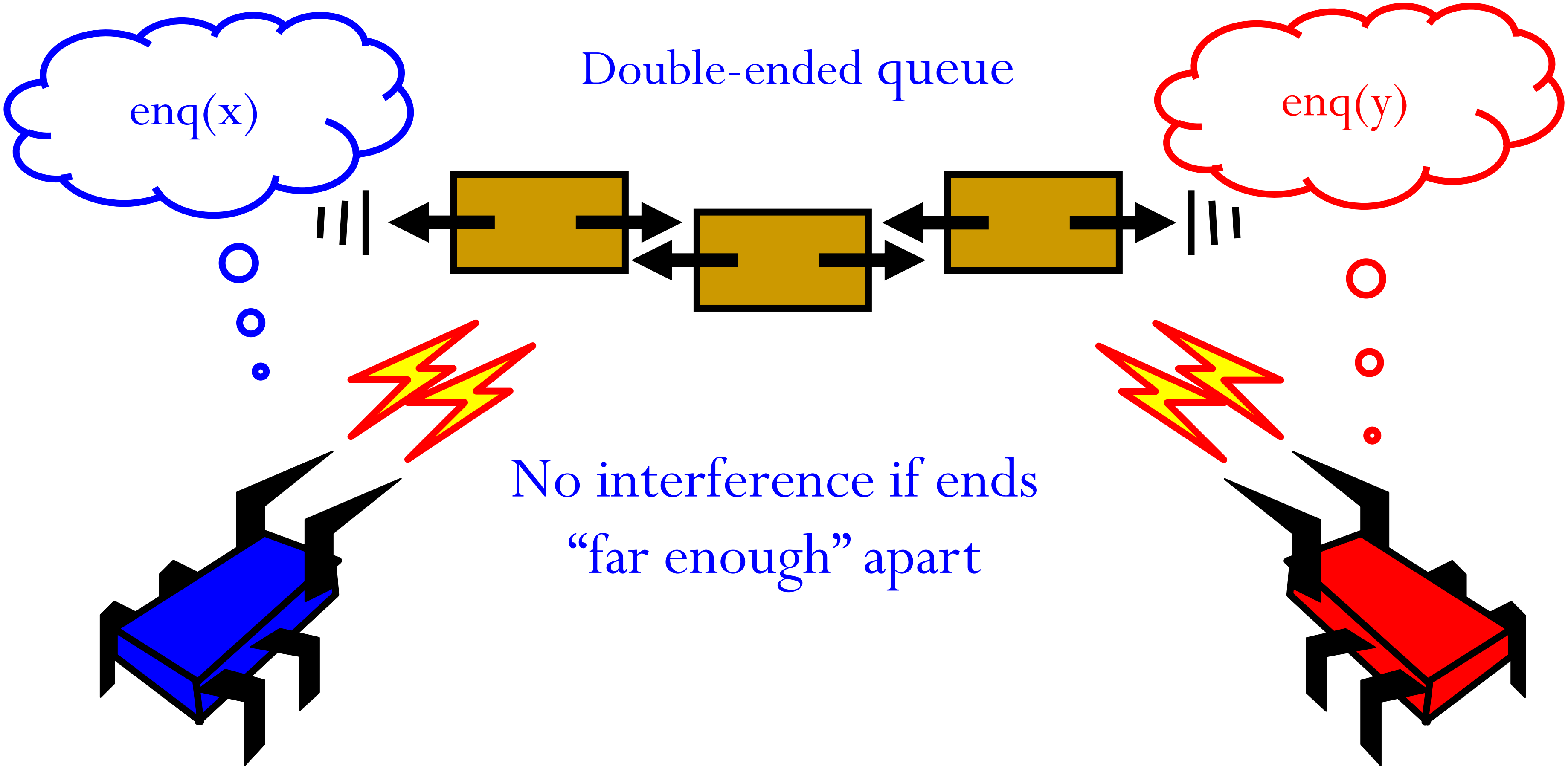
Parallel Algorithms

“In the context of sequential algorithms, it is standard practice to design more complex algorithms that outperform simpler ones (for example, by implementing a balanced tree instead of a list). For non-blocking algorithms, however, implementing more complex data structures has been prohibitively difficult.

[Herlihy, Luchangco, Moir, Scherer, PODC 2003]

Discussing a concurrent red-black tree
(data structures 101).

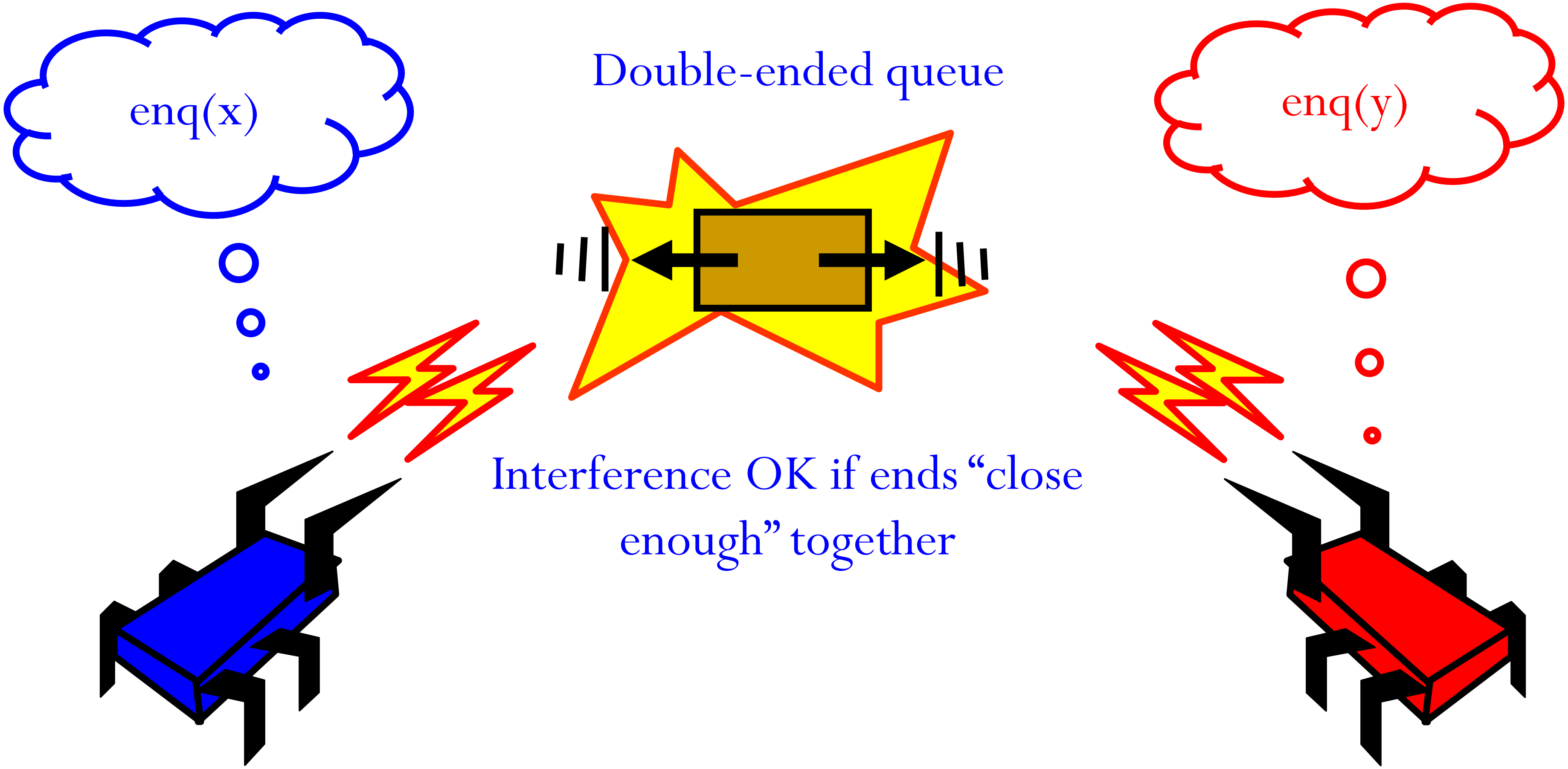
Sadistic Homework (c. Maurice Herlihy)



Double-ended queue

No interference if ends "far enough" apart

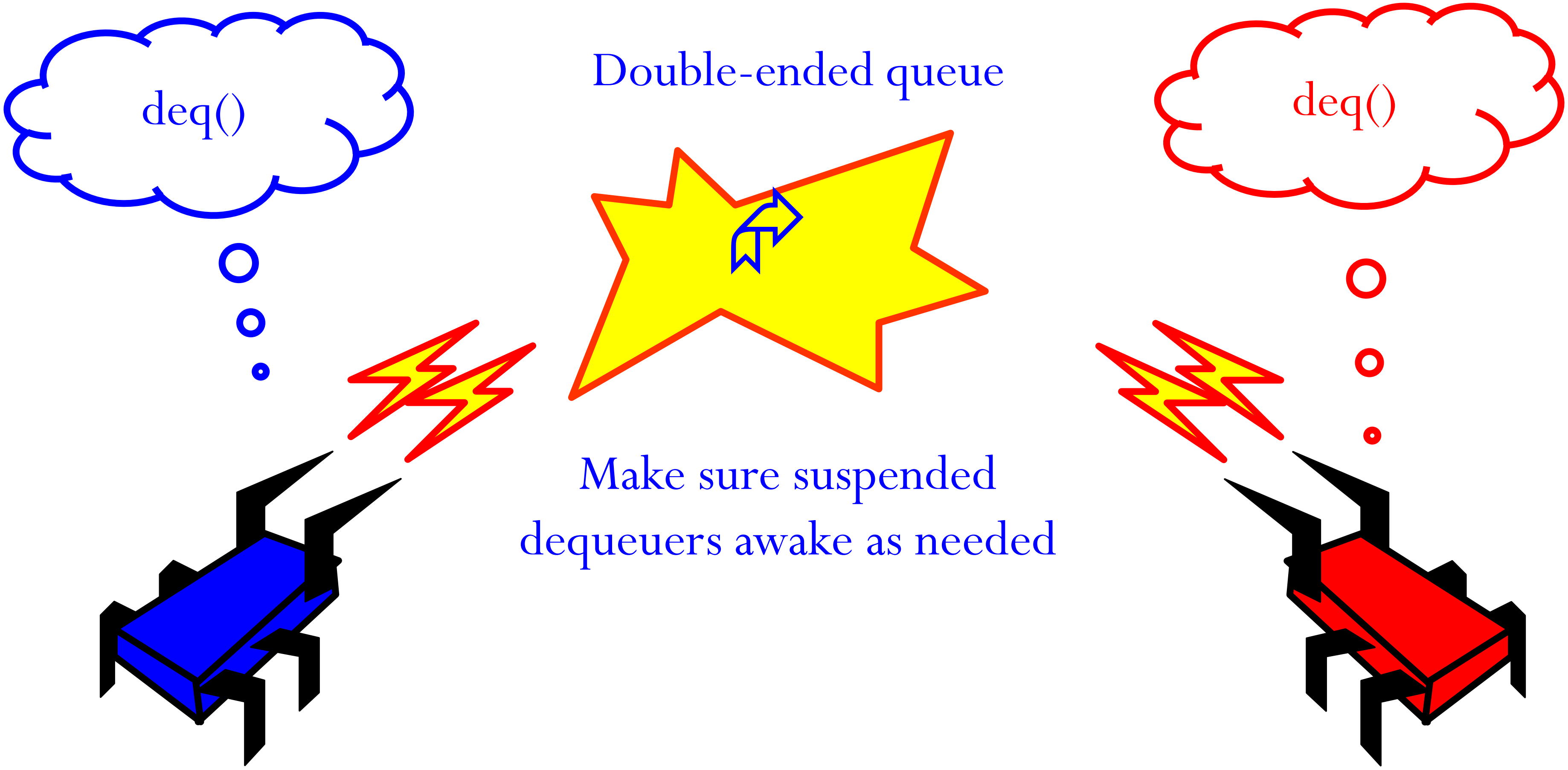
Sadistic Homework



Double-ended queue

Interference OK if ends "close enough" together

Sadistic Homework



deq()

Double-ended queue

deq()

Make sure suspended dequeuers awake as needed

You Try It …

- One lock?
  - too conservative
- Locks at each end?
  - deadlock, too complicated, etc
- Waking blocked dequeuers?
  - harder that it looks

Solution

- Clean solution is a publishable result
    - [Michael & Scott, PODC 96]
- What kind of world are we moving to when solutions to such elementary problems are publishable?

Difficult Programming

- Parallel programming is as difficult as sequential programming +
  - Synchronization
  - Data races
  - Non-determinism
  - Non-existent language and tools support

Few Parallel Abstractions

- Parallel programming models are low-level and machine-specific
  - Shared memory or message passing (~ hardware)
- Parallel programming constructs are “assembly language”
  - Thread == processor
  - Semaphore == atomic increment
  - Lock == compare & swap
- Performance models are machine-specific
- ⇒ Parallel programs are low-level and machine-specific
  - Hard to port, reuse investments, develop market, or gain economies of scale

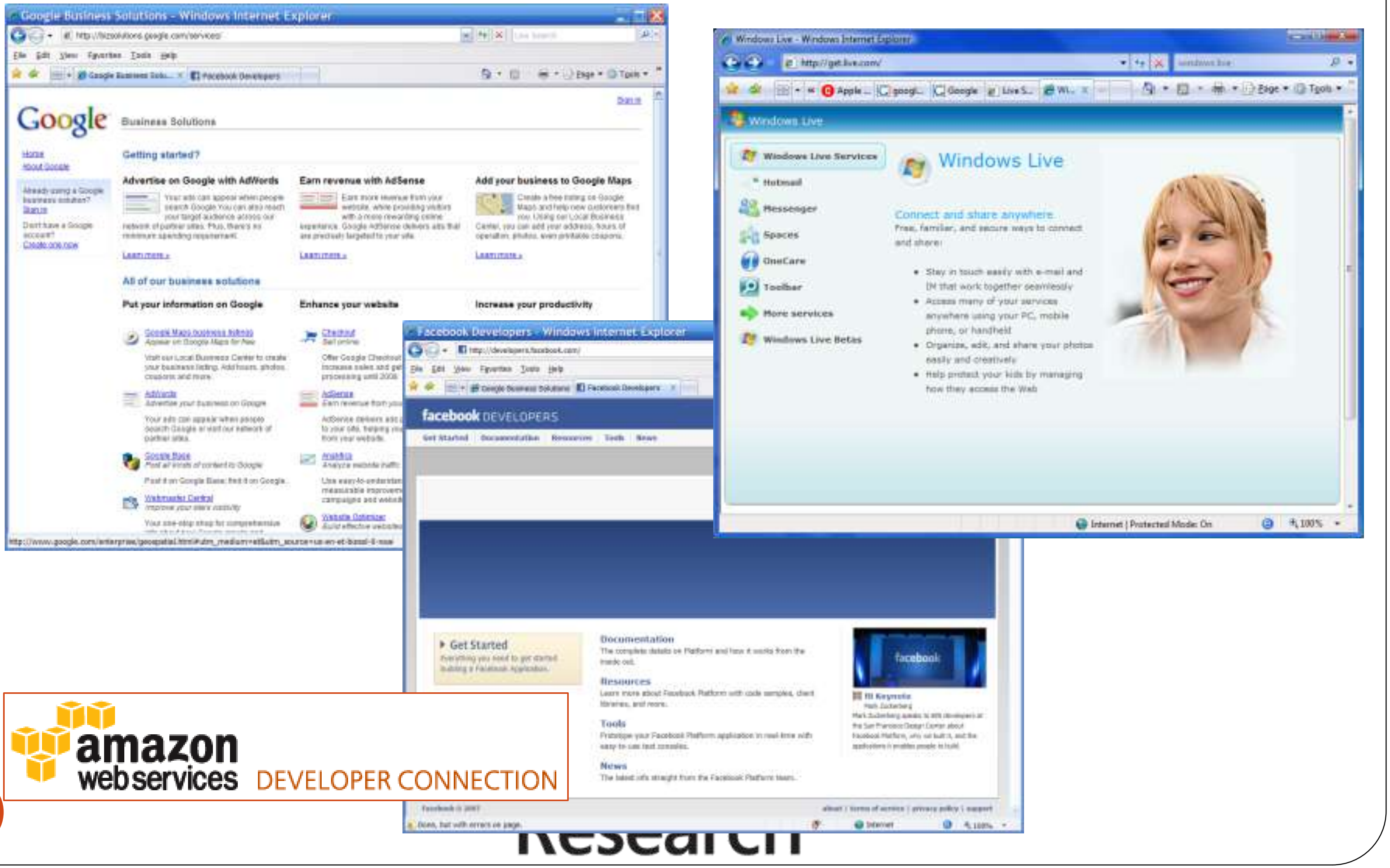
Outline

- Where Moore's Dividend was spent?
- Is parallel computing a plausible successor?
- Parallel computing models
- Impact on computing

Parallelism Will Change Computing

- Last revolution was commodity multiprocessors
  - Supplanted specialized processors and mainframes
  - “Killer micros” improved at 50%/yr
  - Software industry was born
- If existing applications and systems cannot use parallelism, new applications and systems will
  - Software + services
  - Mobile computing

Cloud Computing



Google Business Solutions - Windows Internet Explorer

http://bizsolutions.google.com/services/

Google Business Solutions

Home
About Google

Already using a Google business solution? Sign in

Don't have a Google account? Create one now

Getting started?

Advertise on Google with AdWords

Your ads can appear when people search Google. You can also reach your target audience across our network of partner sites. Plus, there's no minimum spending requirement.

Learn more »

Earn revenue with AdSense

Earn more revenue from your website, while providing visitors with a more rewarding online experience. Google AdSense delivers ads that are precisely targeted to your site.

Learn more »

Add your business to Google Maps

Create a free listing on Google Maps and help new customers find you. Using our Local Business Center, you can add your address, hours of operation, photos, even printable coupons.

Learn more »

All of our business solutions

Put your information on Google

Google Maps business listings
Appear on Google Maps for free

Visit our Local Business Center to create your business listing. Add hours, photos, coupons and more.

AdWords
Advertise your business on Google

Your ads can appear when people search Google or visit our network of partner sites.

Google Base
Post all kinds of content to Google

Post it on Google Base; find it on Google.

Webmaster Central
Improve your site's visibility

Your one-stop shop for comprehensive

Enhance your website

Checkout
Sell online

Offer Google Checkout to increase sales and get processing until 2008.

AdSense
Earn revenue from your

AdSense delivers ads to your site, helping you from your website.

Analytics
Analyze website traffic

Use easy-to-understand measurable improvement campaigns and website

Website Optimizer
Build effective websites

Increase your productivity

http://www.google.com/enterprise/geospatial.html#utm_medium=et&utm_source=us-en-et-bizsol-8-nsa

Facebook Developers - Windows Internet Explorer

http://developers.facebook.com/

facebook DEVELOPERS

Get Started | Documentation | Resources | Tools | News

Get Started
Everything you need to get started building a Facebook Application.

Documentation
The complete details on Platform and how it works from the inside out.

Resources
Learn more about Facebook Platform with code samples, client libraries, and more.

Tools
Prototype your Facebook Platform application in real-time with easy-to-use test consoles.

News
The latest info straight from the Facebook Platform team.

f8 Keynote
Mark Zuckerberg
Mark Zuckerberg speaks to 800 developers at the San Francisco Design Center about Facebook Platform, why we built it, and the applications it enables people to build.

Facebook © 2007

about | terms of service | privacy policy | support

Done, but with errors on page.

Windows Live - Windows Internet Explorer

http://get.live.com/

Windows Live

Windows Live Services
Hotmail
Messenger
Spaces
OneCare
Toolbar
More services
Windows Live Betas

Windows Live

Connect and share anywhere
Free, familiar, and secure ways to connect and share:

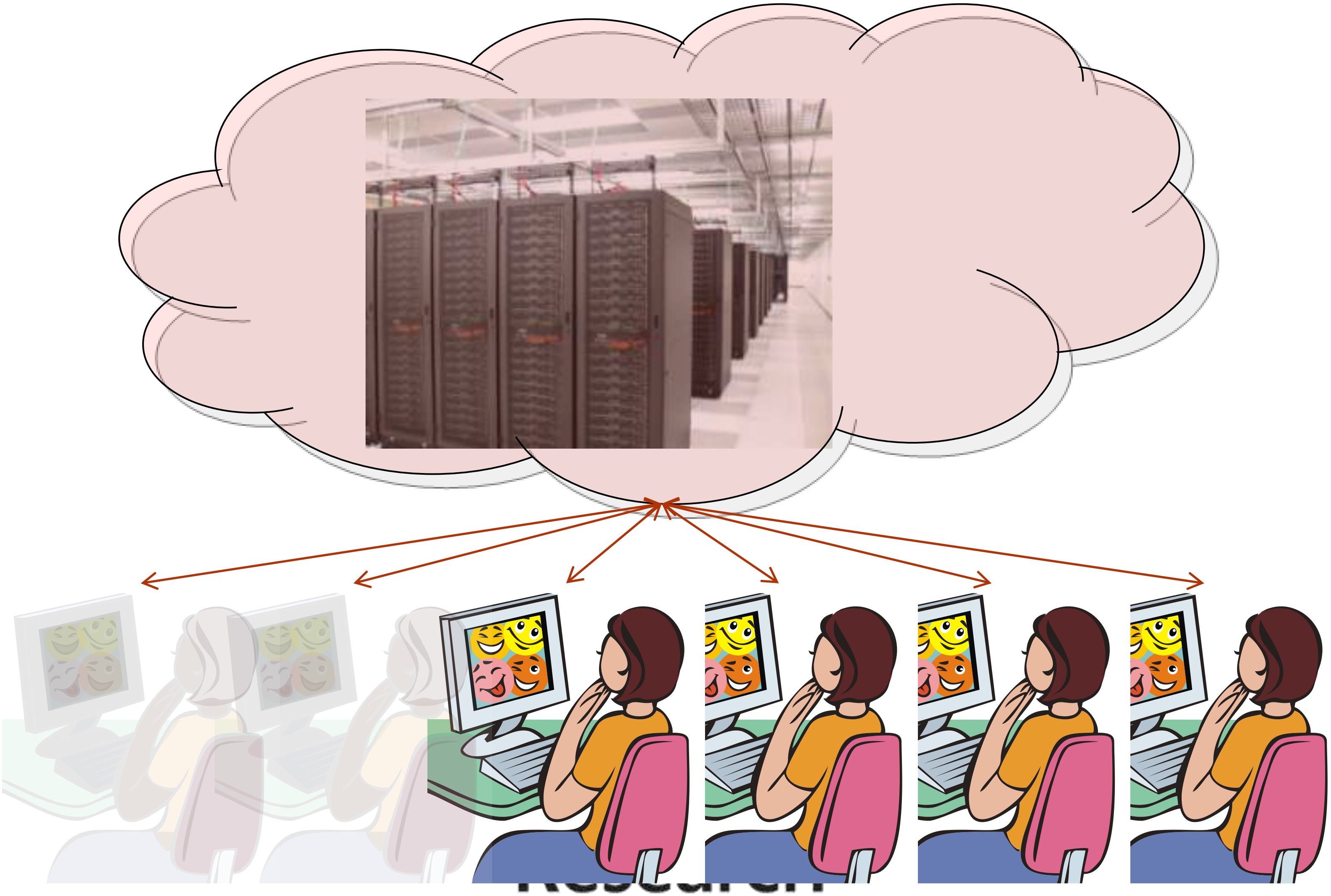
- Stay in touch easily with e-mail and IM that work together seamlessly
- Access many of your services anywhere using your PC, mobile phone, or handheld
- Organize, edit, and share your photos easily and creatively
- Help protect your kids by managing how they access the Web

Internet | Protected Mode: On

amazon webservices DEVELOPER CONNECTION

Research

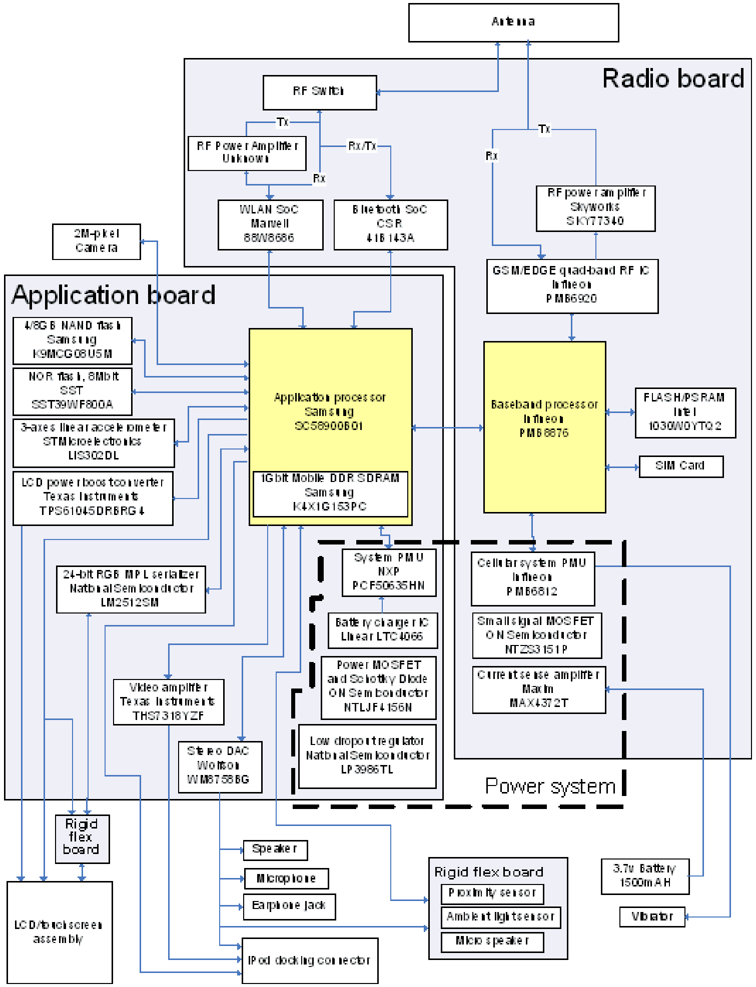
New Software Architecture



Embarrassingly Parallel

- Even sequential applications become parallel when hosted
  - Few dependencies between users
- Moore's Benefits accrue to platform owner
  - 2x cores ⇒
    - ½ servers (+ ½ power, space, cooling, etc.)
    - Or 2x service (same cost)
- Many implications for desktop platform, mobility, etc.
- Tradeoffs not entirely one-sided because of latency, bandwidth, privacy, off-line considerations; as well as capital investment, security, programming problems

Mobile is Parallel

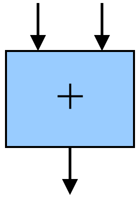


Microsoft® Research

Parallelism Reduces Energy

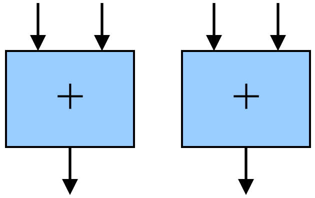
8-bit adder/compare

- 40MHz at 5V, area = 530 kμ^2
- Base power P_{ref}



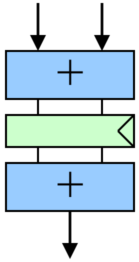
Two parallel interleaved adder/cmp units

- 20MHz at 2.9V, area = 1,800 kμ^2 (3.4x)
- Power = 0.36 P_{ref}



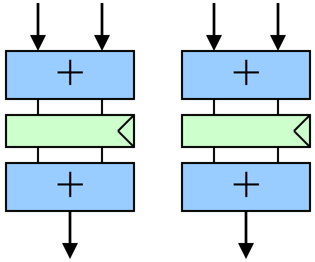
One pipelined adder/cmp unit

- 40MHz at 2.9V, area = 690 kμ^2 (1.3x)
- Power = 0.39 P_{ref}



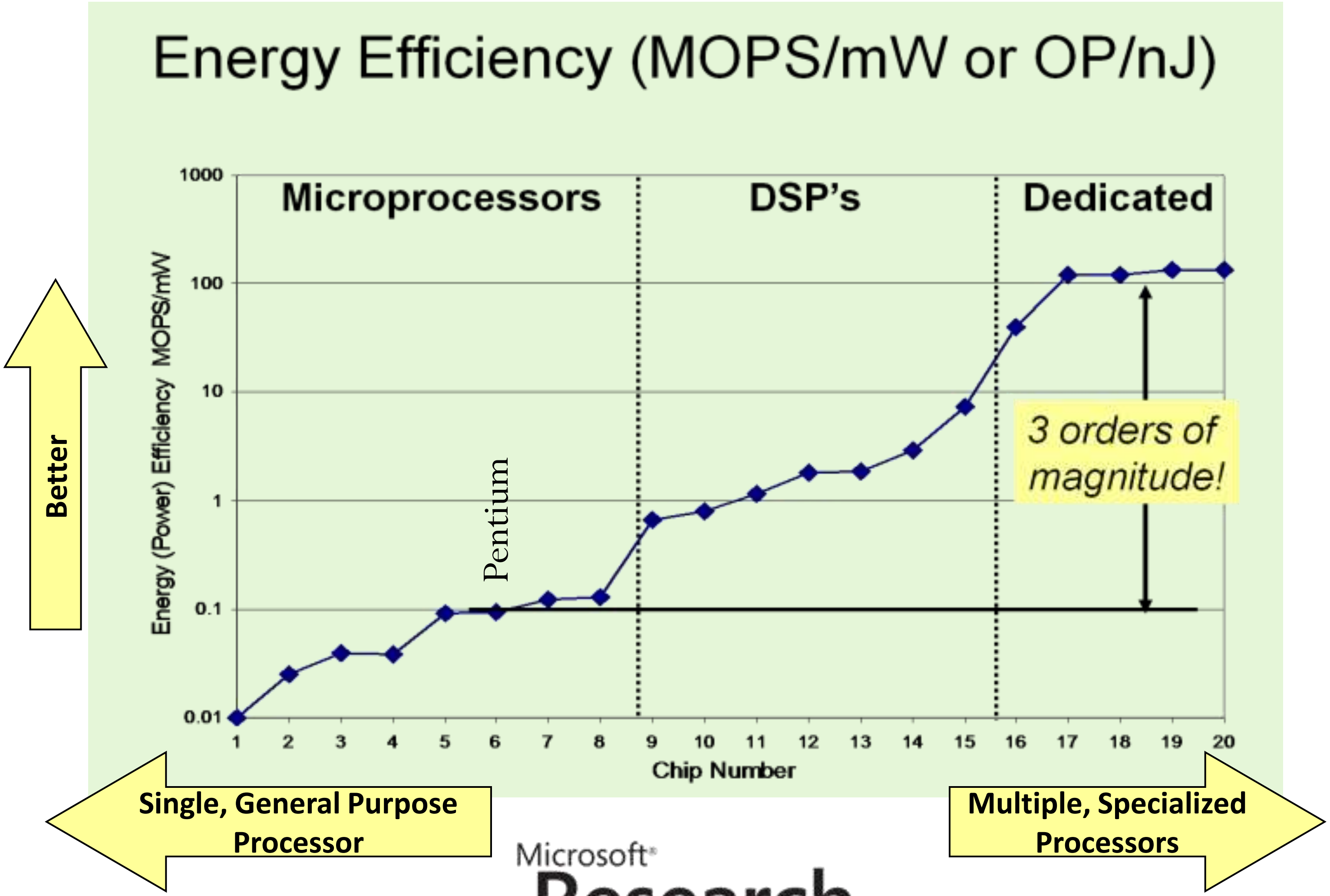
Pipelined and parallel

- 20MHz at 2.0V, area = 1,961 kμ^2 (3.7x)
- Power = 0.2 P_{ref}



Chandrakasan et. al, IEEE JSSC 27(4), April 1992. Slide from Krste Asanovic, "Clock and Power," 6.375, March 07.

Heterogeneous Parallelism
Really Reduces Energy



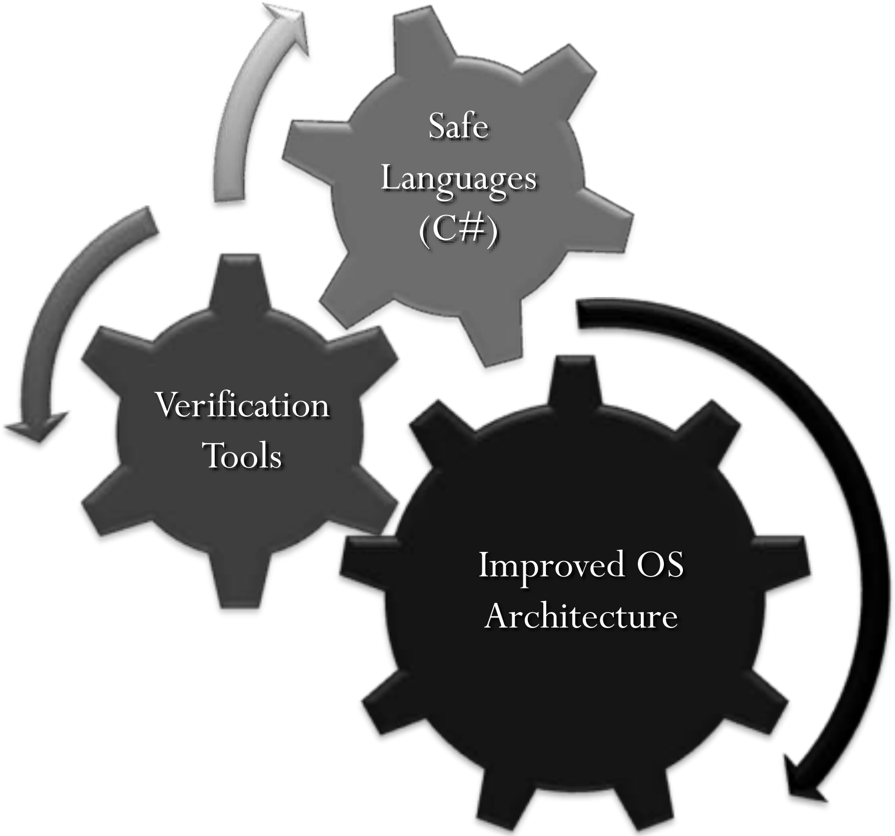
Microsoft Research

Mihai Budiu, "On The Energy Efficiency of Computation," Talk, Feb 2004.

Opportunity to Rethink Computing

- Day-to-day challenges should not obscure opportunity for major improvements in computing experience
  - PC (Mac, Linux, etc.) is not epitome of computing (I hope)
- Focus on performance can eclipse more important qualities (reliability, robustness)
  - Wasteful to use half of processors to monitor other half?
- Disruptive changes are opportunity to introduce "impossible" improvements

Singularity Project



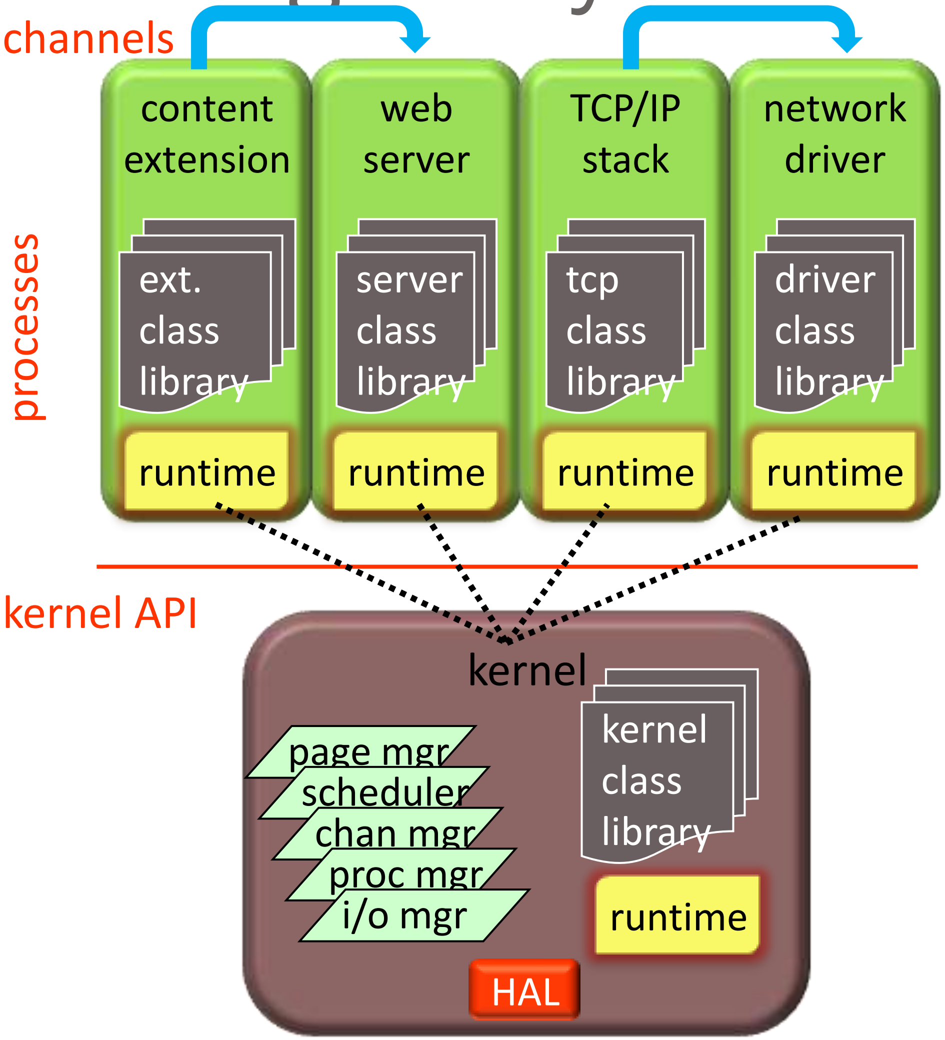
- Large Microsoft Research project with goal of more robust and reliable software
  - Galen Hunt, Jim Larus, and many others
- Started with firm architectural principles
  - Software will fail, system should not
  - System should be self-describing
  - Verify as many system aspects as possible
- No single magic bullet
  - Mutually reinforcing improvements to languages and compilers, systems, and tools

Microsoft Research

Key Tenets

1. Use safe programming languages everywhere
   - Safe ⇒ type safe and memory safe (C# or Java)
   - Everywhere ⇒ applications, extensions, OS services, device drivers, kernel
2. Improve system resilience in the face of software errors
   - Failure containment boundaries
   - Explicit failure notification model
3. Facilitate modular verification
   - Make system "self-describing," so pieces can be examined in isolation
   - Specify and check behavior at many levels of abstraction
   - Facilitate automated analysis

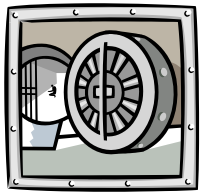
Singularity OS Architecture



- Safe micro-kernel
  - 95% written in C#
    - 17% of files contain unsafe C#
    - 5% of files contain x86 asm or C++
  - Services and device drivers in processes
- Software isolated processes (SIPs)
  - All user code is verifiably safe
  - Some unsafe code in trusted runtime
  - Processes and kernel sealed
- Communication via channels
  - Channel behavior is specified and checked
  - Fast and efficient communication
- Working research prototype
  - Not Windows replacement

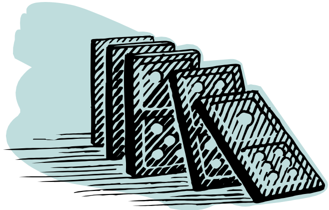
Microsoft Research

Challenge 1: Pervasive Safe Languages



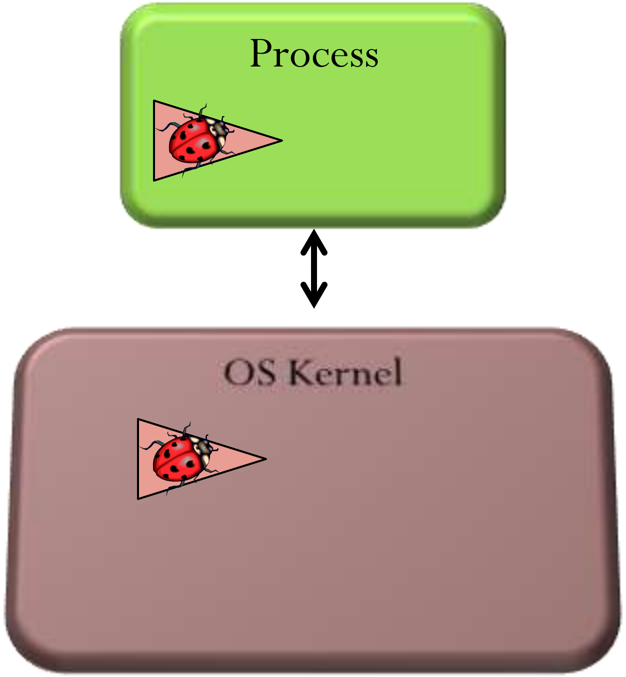
- Modern, safe programming languages
  - Prevent entire classes of (serious) defects
  - Easier to analyze
- Singularity is written in extended C#
  - Spec# (C# + pre/post-conditions and invariants)
  - Sing# adds features to increase control over allocation, initialization, and memory layout
- Evolve language to support Singularity abstractions
  - Channel communications
  - Factor libraries into composable pieces
  - Compile-time reflection
- Native compiler and runtime
  - No bytecodes or MSIL
  - No JVM or CLR

Challenge 2: Improve Resilience



- Cannot build software without defects
  - Verification is a chimera
  - (But we could still do a lot better)
- Software defects should not cause system failure
- A resilient system architecture should
  - Isolate system components to prevent data corruption
  - Provide clear failure notification
  - Implement policy for restarting failed component
- Existing system architectures lack isolation and resilience

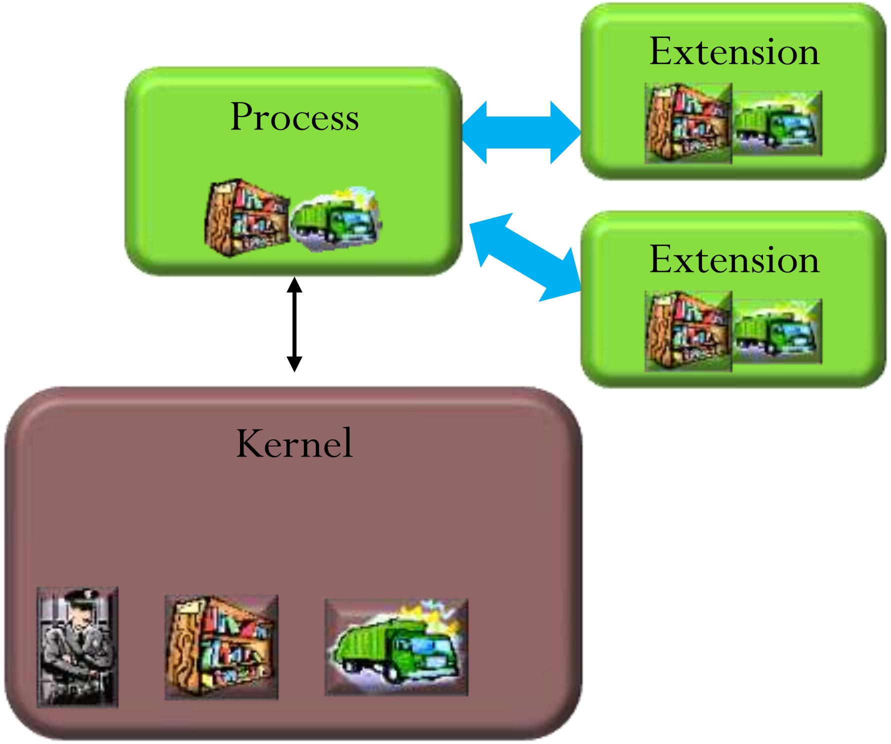
Open Process Architecture



- Ubiquitous architecture (Windows, Unix, Java, etc.)
  - DLLs, classes, plug-ins, device drivers, etc.
- Processes are not sealed
  - Dynamic code loading and runtime code generation
  - Shared memory
  - System API allow process to alter another's state
- Low dependability
  - 85% of Windows crashes caused by third party code in kernel
  - Interface between host and extension often poorly documented and understood
  - Maintenance nightmare

Microsoft Research

Sealed Processes



- Singularity processes are sealed
  - No dynamic code loading or run-time code generation
  - All code present when process starts execution
  - Extensions execute in distinct processes
    - Separate closed environments with well-defined interfaces
  - No shared memory
- Fundamental unit of failure isolation
- Improved optimization, verification, security

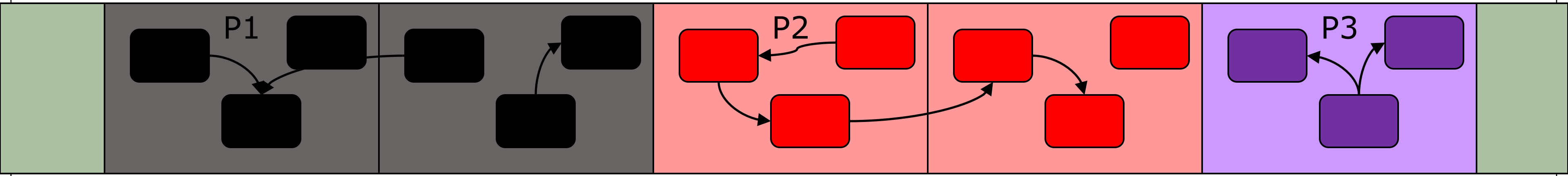
Microsoft Research

Isolation Requires Lightweight Processes

- Existing processes rely on virtual memory and protection domains
  - VM prevents reference into other address spaces
  - Protection prevents unprivileged code from access system resources
- Processes are expensive to create and schedule
  - High cost to cross protection domains (rings), handle TLB misses, and manipulate address spaces
- Cost encourages monolithic architecture
  - Expensive process creation and inter-process communication
  - Large, undifferentiated applications
  - Dynamically loaded extensions

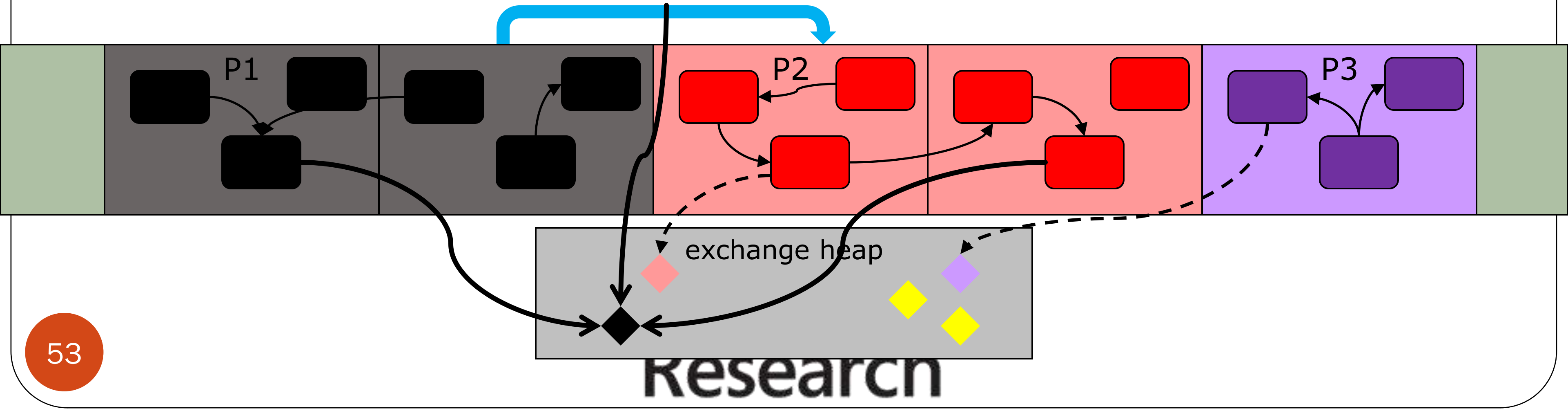
Software Isolated Processes (SIPs)

- Protection and isolation enforced by language safety and kernel API design
  - Process owns a set of pages
  - All of process's objects reside on its pages (object space, not address space)
  - Language safety ensures process can't create or mutate reference to other pages
- Global invariants:
  - No process contains a pointer to another process's object space
  - No pointers from exchange heap into process

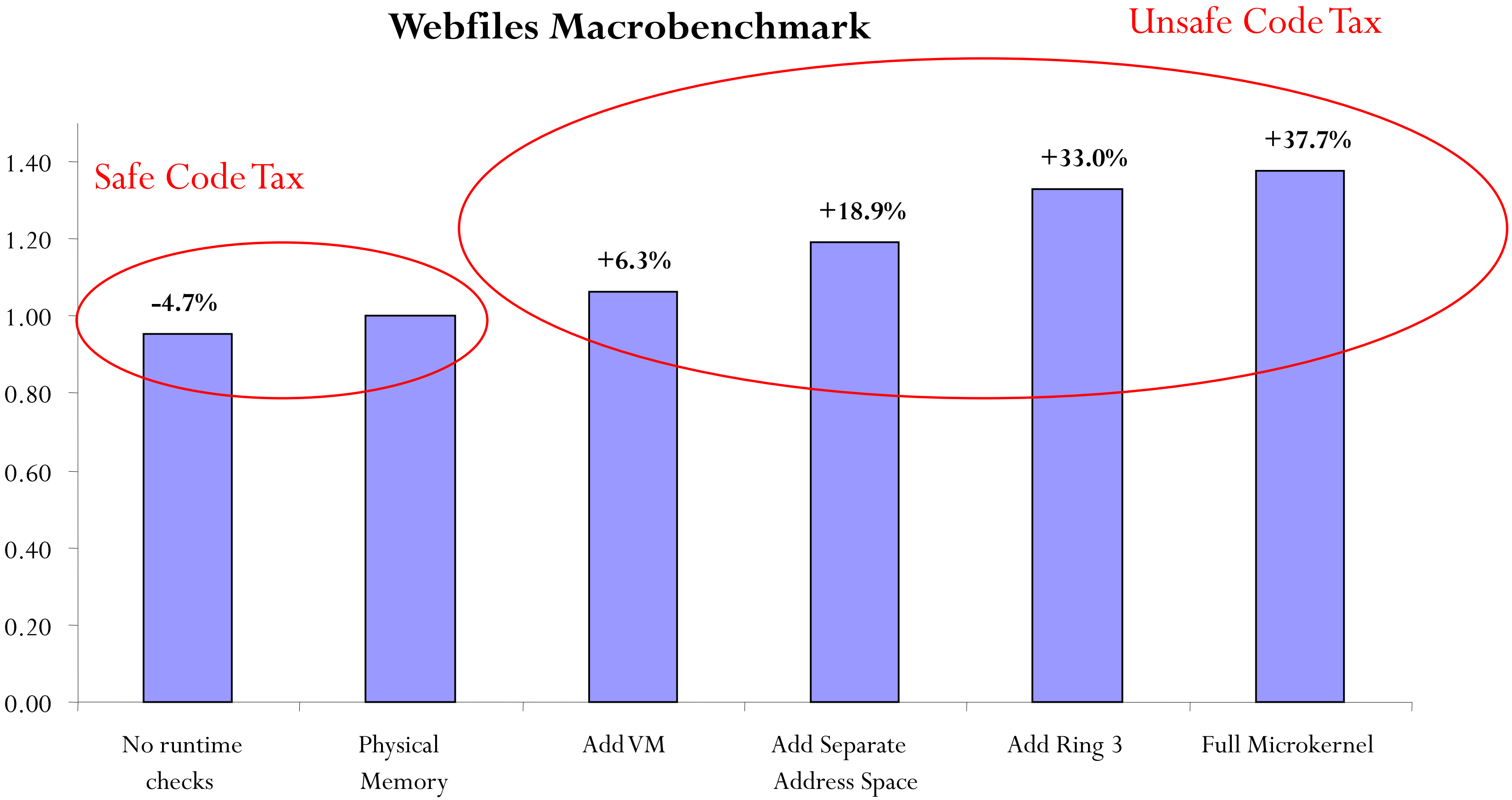


Interprocess Communications

- Channels are strongly typed (value & behavior), bidirectional communications ports
  - Messages passing with extensive language support
- Messages live outside processes, in exchange heap
  - Only a single reference to a message
- “Mailbox” semantics enforced by linear types
  - Copying and pointer passing are semantically indistinguishable
- Channel buffers pre-allocated according to contract

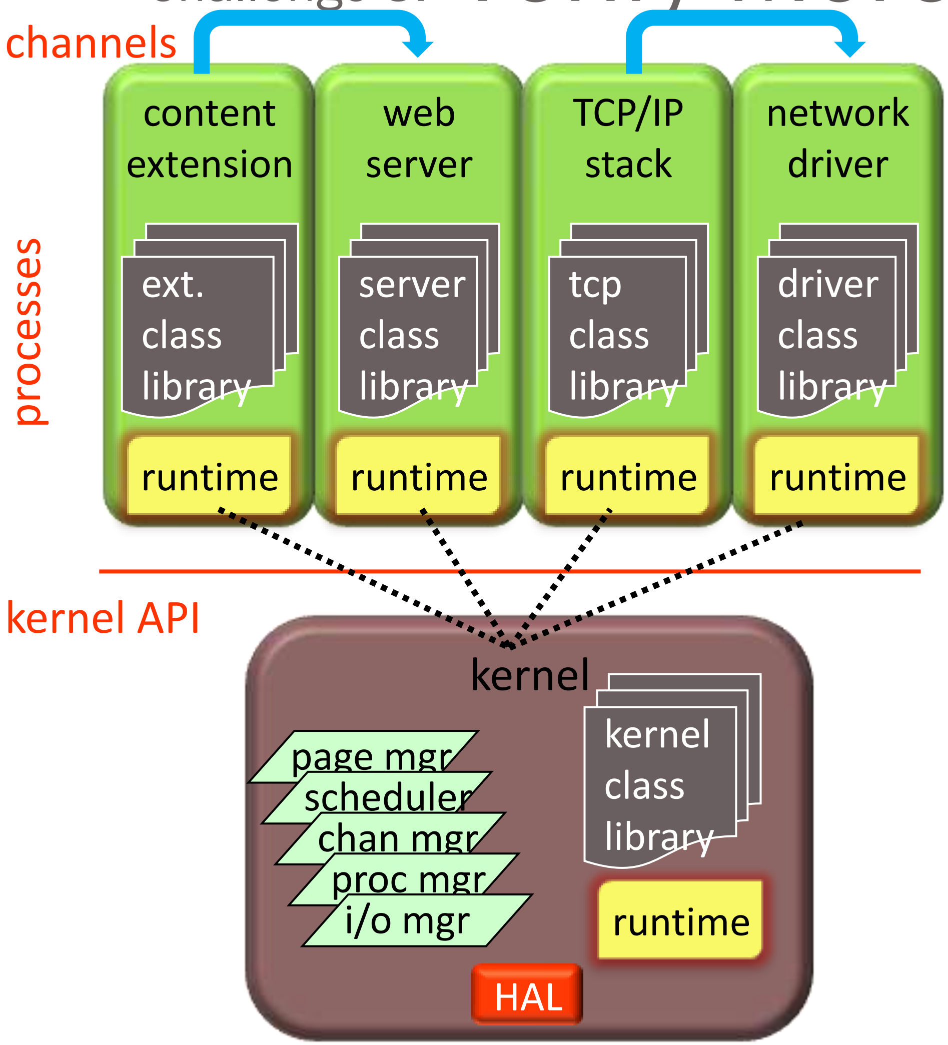


Hardware is Costly



Microsoft Research

Challenge 3: Verify More



- Process internals (code):
  - Type safety
  - Object invariants
  - Method pre- & post- conditions
  - Component interfaces
- Process externals:
  - Channel contracts
  - Resource access & dependencies
- System:
  - Communication safety
  - Hardware resource conflict free
  - Namespace conflict free
- Static verification: before code runs

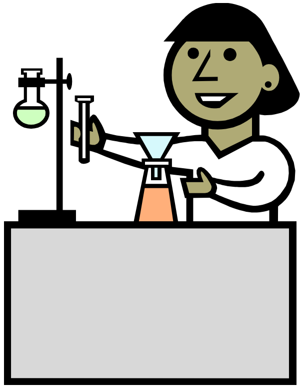
Microsoft Research

Cloud Computing Challenges

- Software stack (client and server) that is robust and reliable
  - Fail and recover, not fail and restart
  - Build on best language, tool, and software development practices
  - Security from the beginning
- Software behaves in understandable and predictable manner
  - Users have no idea what is "behind the curtain" (and don't want to)
  - Natural interfaces
- New, compelling uses for computing
  - Personal assistant

Microsoft Research

Research Community Challenges



- Rethink assumptions behind software stack
  - Multics was an amazing project, 40 years ago
  - The world has changed, so should our assumptions
- People develop software
  - Social/organization issues are huge factor
  - Tools are secondary
- Huge gap between research and practice
  - Researchers are unaware of practical issues, problems, and trends
  - Practitioners’ formal education ends when they graduate

Software

Well this place is old
It feels just like a beat up truck
I turn the engine, but the engine doesn't turn
Well it smells of cheap wine & cigarettes
This place is always such a mess
Sometimes I think I'd like to watch it burn
I'm so alone, and I feel just like somebody else
Man, I ain't changed, but I know I ain't the same

– *One Headlight,* Jakob Dylan (Wallflowers)