

# *21<sup>st</sup> Century Software*

Andrew Herbert  
Microsoft Research  
Cambridge, UK

26<sup>th</sup> September 2007

# Rethinking software for the 21<sup>st</sup> century

- In the 1970s, computer software was designed in large part to overcome hardware limitations
- In the 21st century, many of these limitations no longer apply. We have an abundance of computing resources
- This is changing how we think about software

# Personal Computing in 1970...



# Caution

- Much of what I will say relates primarily to personal computing
- There are many computer applications that are still held back by hardware
  - Large server systems such as Google or Hotmail
  - High Performance Computing applications for science, such as genomics, computational physics and chemistry

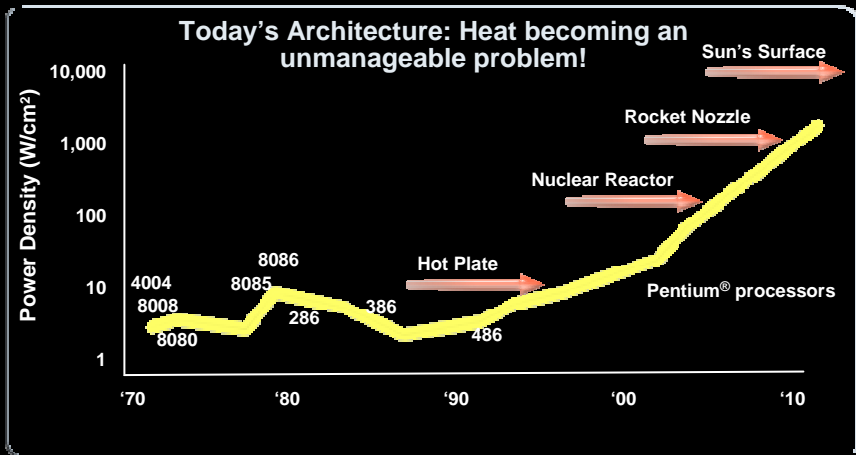
# Moore' Law (1967)

- Not really a “law”, but an observation, intended to hold for “...the next few years...”
- $(N_t/A)_{t_1} = (N_t/A)_{t_0} * 1.58^{t_1-t_0}$  (t in years)
  - $N_t$ : number of transistors; A: area
- Moore's observation has held for 35 years and has sustained the personal computer industry
- NB Moore's Law is about transistor count, not speed...

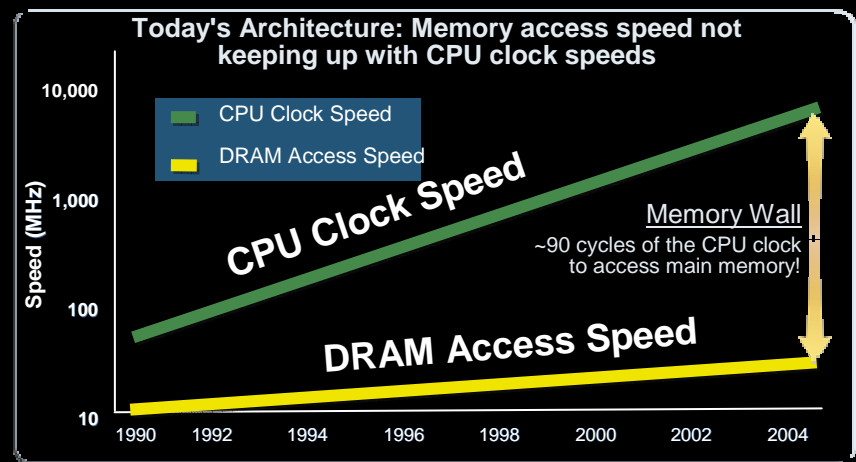
# Implications for Processors

- More complex designs: rich instruction sets, pipelining, out of order execution, speculative execution, caching
- More than one processor on a chip (homogeneous multi-processor)
- More than one processor on a chip, with specialized functions,
  - Graphics performance is improving much faster than CPU performance

# The Intel perspective...



Intel Developer Forum, Spring 2004 - Pat Gelsinger



Modern Microprocessors - Jason Patterson

“... we see a very significant shift in what architectures will look like in the future ... fundamentally the way we've begun to look at doing that is to move from instruction level concurrency to ... multiple cores per die. But we're going to continue to go beyond there. And that just won't be in our server lines in the future; this will permeate every architecture that we build. All will have massively multicore implementations.”

Intel Developer Forum, Spring 2004  
 Pat Gelsinger  
 Chief Technology Officer, Senior Vice President  
 Intel Corporation  
 February, 19, 2004

## Intel Cancels Top-Speed Pentium 4 Chip

Thu Oct 14, 6:50 PM ET Technology - Reuters  
 By Daniel Sorid

Intel ...canceled plans to introduce its highest-speed desktop computer chip, ending for now a 25-year run that has seen the speeds of Intel's microprocessors increase by more than 750 times.

# Obsolete software idea 1: Single-threaded programs

- With uniprocessors, there is no compelling reason to try to use parallelism and write “concurrent” programs
  - Your program will probably run slower due to thread creation, destruction, and switching
  - Your program is harder to debug (“Heisenbugs”)
- Now we don’t have a choice!
- Attempts to tease the parallelism out of a sequential program automatically haven’t worked out very well
- We need better education, better languages, and better tools, since building concurrent programs is hard
  - *Data driven computation - e.g. “Map-Reduce”*

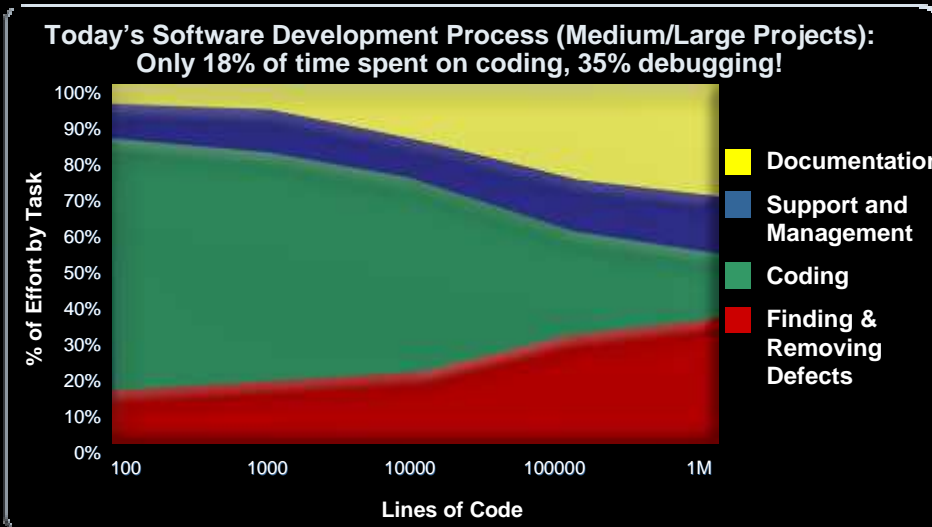
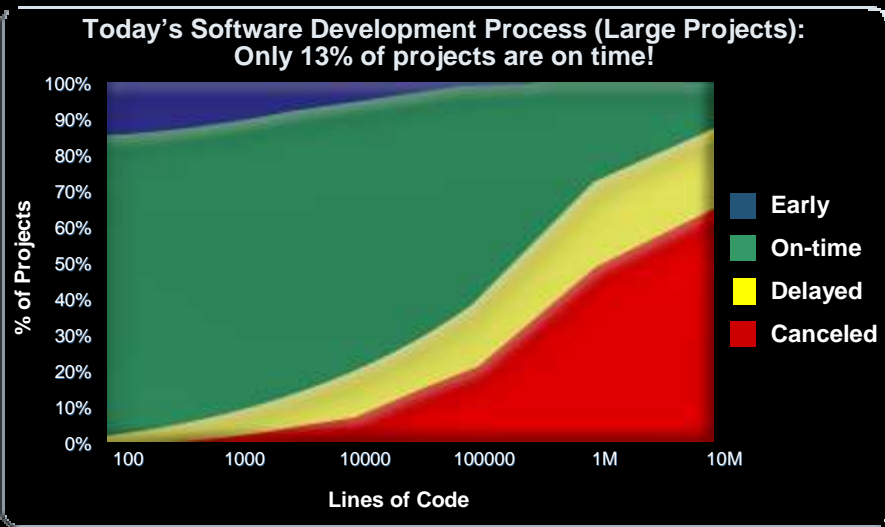
## Obsolete software idea 2:

### Low-level programming languages

- Errors in handling array and heap storage are one of the leading causes of system unreliability in C and C++
- Switching to languages like Java and C#, which provide automatic storage management, makes many types of errors impossible
- Until recently, programmers argued that these languages were too expensive in space and time
  - Today, neither is an issue
- High level languages also allow programs that are easier to understand (and maintain)
  - This is a high cost to the industry, since software evolves
- Functional programming finally comes of age?

# Software Complexity

- Complexity of developing, testing and supporting large scale software systems continues to escalate



Source:  
Capers Jones, Estimating Software Costs, pg. 140  
Capers Jones, Patterns of Software Systems Failure & Success

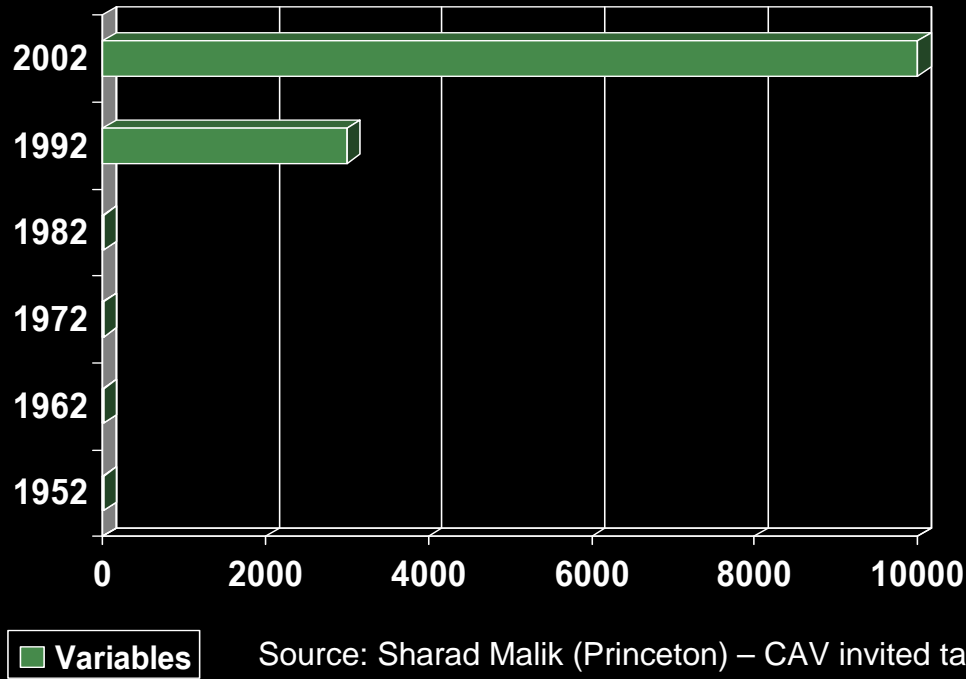


## Obsolete software idea 3:

### Verifying software quality by testing alone

- Testing is needed, but has limits:
  - Hard to find certain types of problems (e.g. concurrency)
  - Huge number of configurations is daunting
- Need more use of formal methods
  - Mathematical model of system: “specification”
  - Automated “verification” of software implementation
  - Used to be impractical due to state space size explosion and CPU speeds
  - Now routinely used in hardware design, where the cost of a bug is *much* larger
  - Increasingly used in Microsoft development

# Capacity Growth in Proof Engines for Propositional Logic



	A	B	C	D	E	F	G
1		1952	1962	1972	1982	1992	2002
2	Variables	10	10	12	15	3000	10000

# Capacity Growth in Proof Engines for Propositional Logic



Achieved through worldwide & competitive research with a focus on industrial problems

■ Variables

Source: Sharad Malik (Princeton) – CAV invited talk

	A	B	C	D	E	F	G
1		1952	1962	1972	1982	1992	2002
2	Variables	10	10	12	15	3000	10000

# Storage

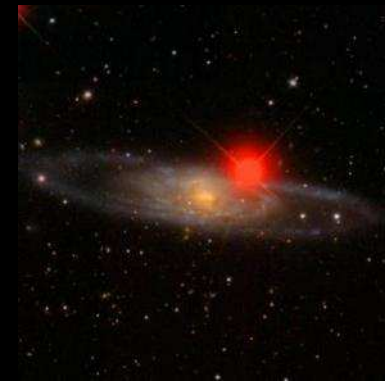
- 30GB Personal file systems



- TerraServer - 5TB



- SkyServer – 40TB



# Obsolete software idea 4: Hierarchical file systems

- Originally introduced to improve access efficiency and to give users a familiar metaphor (the office file cabinet with folders)
- Today, it has serious problems
  - A clean install of Windows and Office has 45,000+ files
  - The structure chosen by a person today will not be appropriate in six months
- Need a new way to organize things so we don't drown in information when we have a personal terabyte

# New Ways of Organizing Files

- Full text indexing
  - Microsoft Vista desktop search
  - Extend to audio and handwriting
- Object recognition for image and video search
- Huge opportunities for personalization and exploiting context
  - Timelines, work flow discovery
  - Email and IM buddies
  - Location awareness
    - Network location, GPS etc

# Example – Visual Summary

## Images in a folder



Thumbnail Viewer - Win XP



Tapestry Viewer

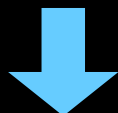
# The MSR Cambridge Digital Tapestry

[Rother, Kumar, Kolmogorov, Blake; CVPR '05]

Input Images



Block Tapestry



**Objective:** Choose informative, representative parts from many images and place them realistically



**Contributions:**  
Novel problem formulation  
Extension of optimisation technique

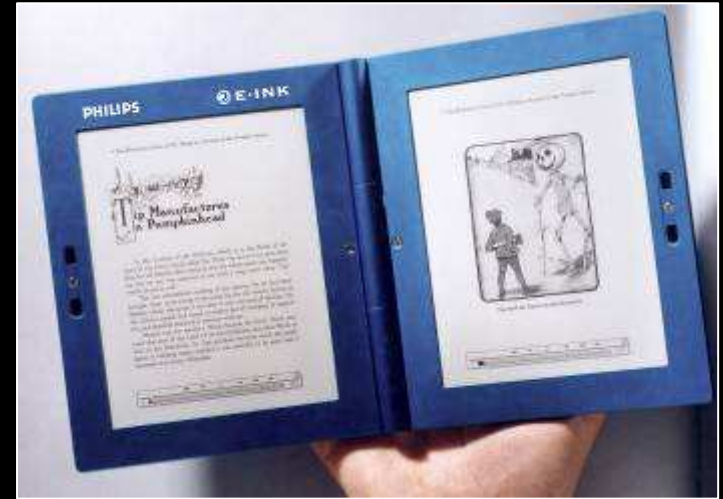
Novel “membrane blending” based on [Perez, Gangnet, Blake Siggraph '03]

Digital Tapestry

**Objective:** Remove any visual seams



# Displays



## Obsolete idea 5:

### Computer monitors (screens) on desks

- What happens when display real-estate is free?
- What happens when dynamic displays are as ubiquitous as conventional signage, or whiteboards?
- What happens when displays can be as large as your wall?
- What happens when displays are as thin as your wallpaper?
- What happens if your cell phone has an A3 display, yet retains portability?
- What happens when all of this is factored in to other trends (wireless, speed, cost, ...)?

# New models of interaction

- Combine new display formats with machine perception
  - Handwriting, gesture – Tablet PC
  - Speech
  - Touch
  - Scanning / Recognition
  - Physical objects
- The future is “interactive surfaces”

## Other things to consider

- **Everything is networked**
  - High bandwidth
  - Wireless
- **Large computer memories**
  - Factor of  $10^6$  growth in capacity
  - Latency is now biggest issue
- **Success of “machine learning” in tackling classical artificial intelligence problems:**
  - Computer vision, information retrieval, handwriting recognition
  - Machine learning = “behaviour capture” rather than “brain emulation”

# What Might Stop Moore's Law?

- Physical limits
  - “Atoms are too large, and light is too slow”
  - Today, the problem isn't making the transistors faster, it's the time for signals to propagate on the wires (latency again)
  - Power. Lots of transistors => lots of power. Cooling is hard
- Design complexity
  - Designing a billion-transistor chip takes a large team, even with good design tools
  - The “junk DNA” problem
- Economics
  - Factories are *very* expensive
- Latency: See D. Patterson, “Latency Lags Bandwidth”, CACM, October 2004

# Conclusions

- Hardware has evolved rapidly
- We haven't exploited it as well as we should
- Lots of exciting research still to do in computer science!