Andrzej Nowak - Bio

- **2005-2006 Intel Corporation**
  - IEEE 802.16d/e WiMax development
  - Linux kernel performance optimizations research
- **2006 Master Engineer diploma in Computer Science**
  - Distributed Applications & Internet Systems
  - Computer Systems Modeling
- **2007-2008 CERN openlab**
  - Multi-core technologies
  - Performance monitoring
  - Systems architecture

Theme: Towards Reconfigurable High-Performance Computing
Lecture **2**

Multi-core Architectures

Andrzej Nowak

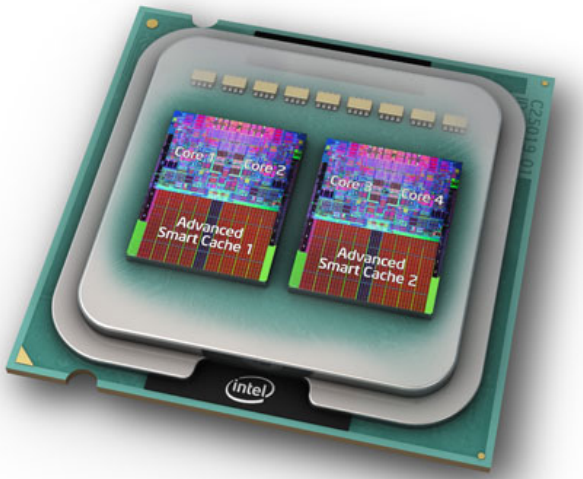
CERN openlab (Geneva, Switzerland)

Inverted CERN School of Computing, 3-5 March 2008

Introduction

- **Objectives:**
  - Explain why multi-core architectures have become so popular
  - Explain why parallelism is such a good bet for the near future
  - Provide information about multi-core specifics
  - Discuss the changes in computing landscape
  - Discuss the impact of hardware on software

- **Contents:**
  - Hardware part
  - Software part
  - Outlook



THE FREE RIDE IS OVER

Recession looms?

Fundamentals of scalability

- **Scalability – “readiness for enlargement”**
- **Good scalability:**
  - Additional resources yield additional performance
- **Poor scalability:**
  - Additional resources yield little or no additional performance past a certain point
- **Failed scalability scenarios:**
  - Increasing resources causes the design to collapse
- **Scaling up (vertically): improving single systems**
- **Scaling out (horizontally): adding additional systems**

Moore's Law (1)

- **An observation made in 1965 by Gordon Moore, the co-founder of Intel Corporation:**
  - "The amount of transistors in an integrated circuit will double every 24 (18) months"
- **What is Moore's Law?**
  - A self fulfilling prophecy?
  - A common observation?

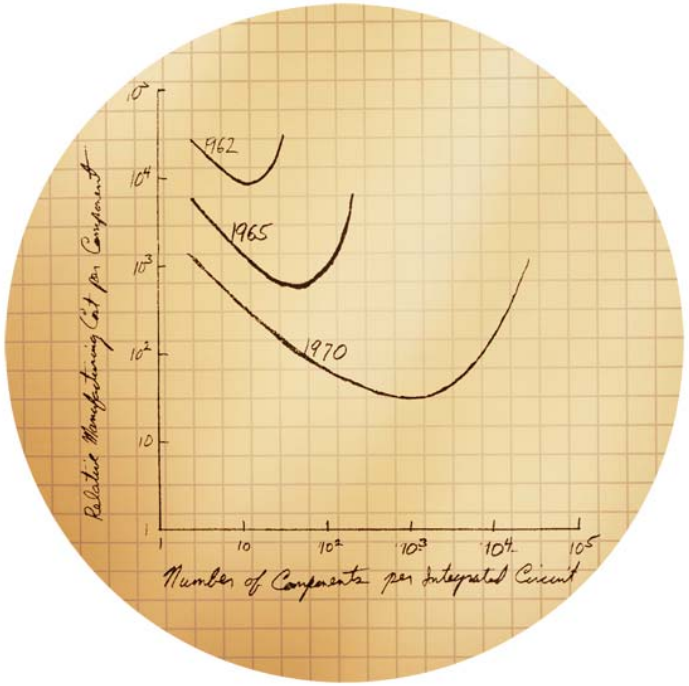


Image: Intel Corporation

Moore's Law (2)

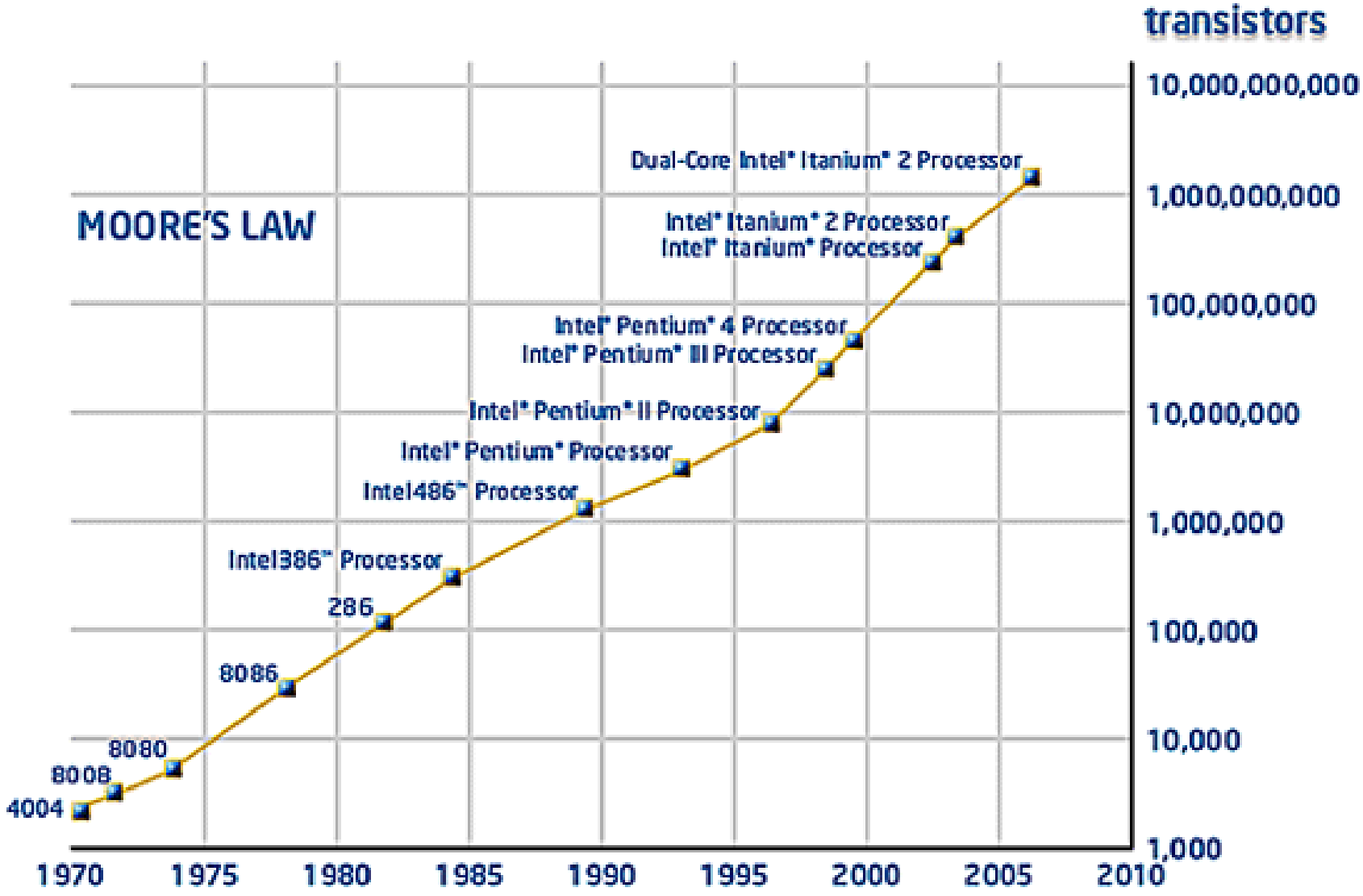


Image: Intel Corporation

Improving CPU performance

- **Common ways to improve the performance of a CPU on the hardware level**

Technique	Advantages	Disadvantages
Frequency scaling	Nearly no design overhead, immediate scaling	Some manufacturing process overhead, leakage problems
Architectural changes	Increased versatility, performance	Huge design and manufacturing overhead, minor (20%) speedups
Simultaneous multi-threading	Medium design overhead, up to 30% performance improvement	Requires more memory, single thread performance hit (~10-15%)
Cloning chips (MCP)	Minimal design overhead	Requires parallel software & more memory, inter-chip communication difficult
Adding processing cores	Small design overhead, easy to scale, 50%+ performance improvement	Requires parallel software or more memory

The good old days (1)

- **In the good old days one just had to upgrade the frequency of a CPU / BUS to get better performance**
- **The painful lesson of the Intel Pentium 4 (Netburst microarchitecture)**
  - Ca. 20% better performance than Pentium III (rough figure)
  - Good scaling to large frequencies – in theory up to 10 GHz
  - Surprise: very serious leakage problems (20-30W cited)
    - 130nm -> 90nm process transition
    - Relatively unsophisticated materials used for production
  - AMD chose “more thinking” instead of “more speed” and began to win in certain market segments
- **In a nutshell: power dissipation problems are closing this avenue, and call for improvements even in modern designs**

The good old days (2)

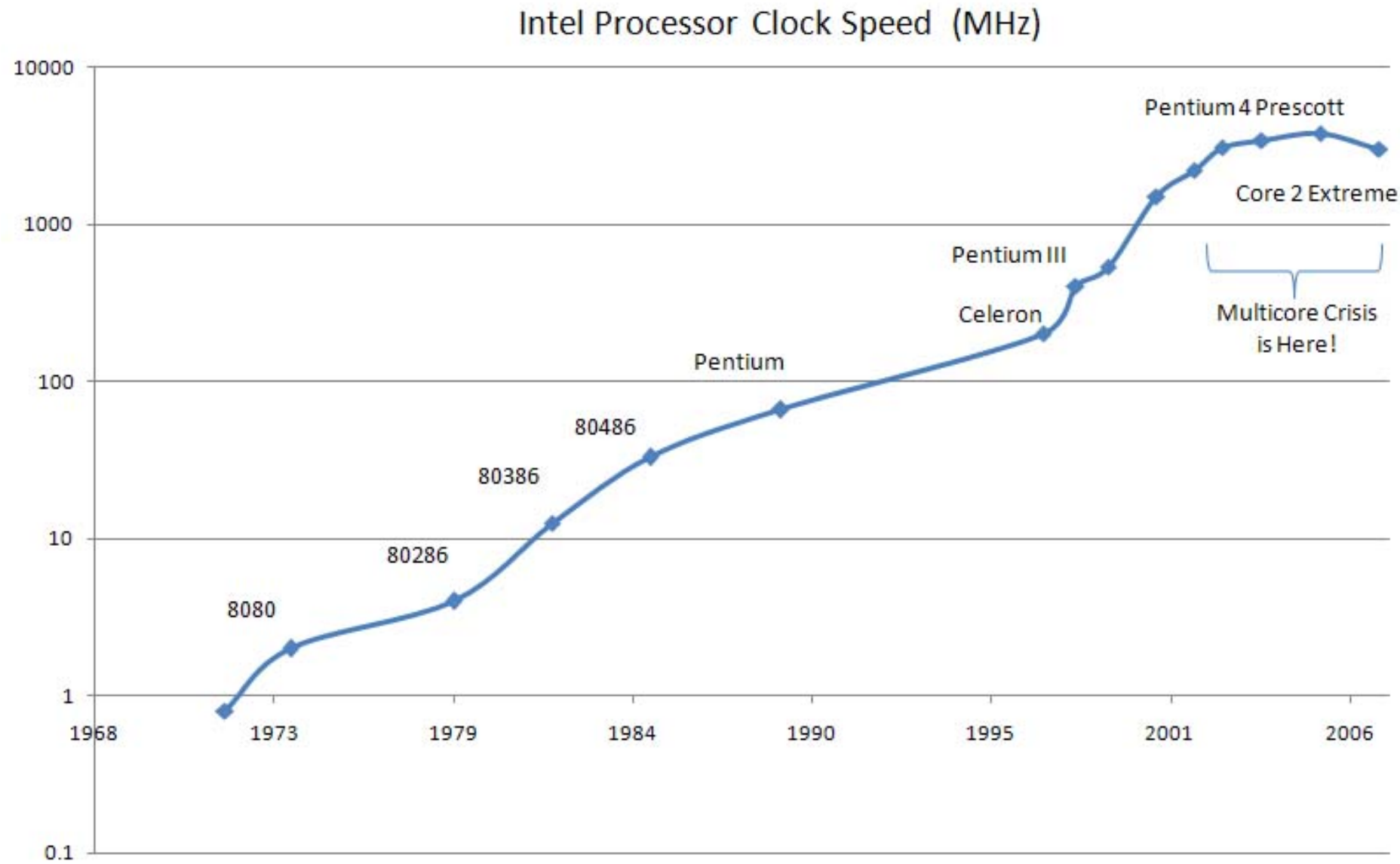
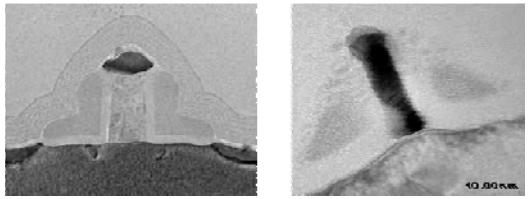


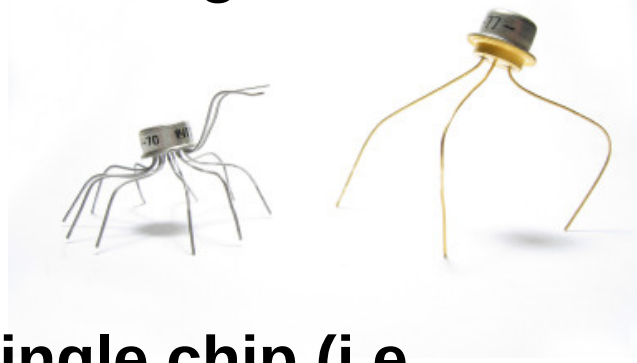
Image source unknown

The move to multi-core (1)

- **The free ride is over**
- **Employing better manufacturing processes leads to die space savings; more die space = more transistors**
  - 90nm, 65nm and 45nm today
  - 32nm, 22nm and 10nm in the works



- **Chip makers decided that the best (cheapest) use of the extra transistors is adding additional processing cores**
- **4 or 8 cores with up to 64 threads today**
- **Hundreds of cores tomorrow?**



- **Other avenues: more functionality on a single chip (i.e. adding a GPU, crypto, etc)**

The move to multi-core (2)

- **Multi-core long used in DSP, embedded and network processors**
- **Many household items contain multi-core processors:**
  - iPOD – 2 core ARM CPU
  - The PS3 – IBM Cell CPU
  - The Xbox 360 – 3-core Xenon CPU (PowerPC based, SMT capable)
  - GPUs, like the NVIDIA 8800 series – up to 128 mini cores

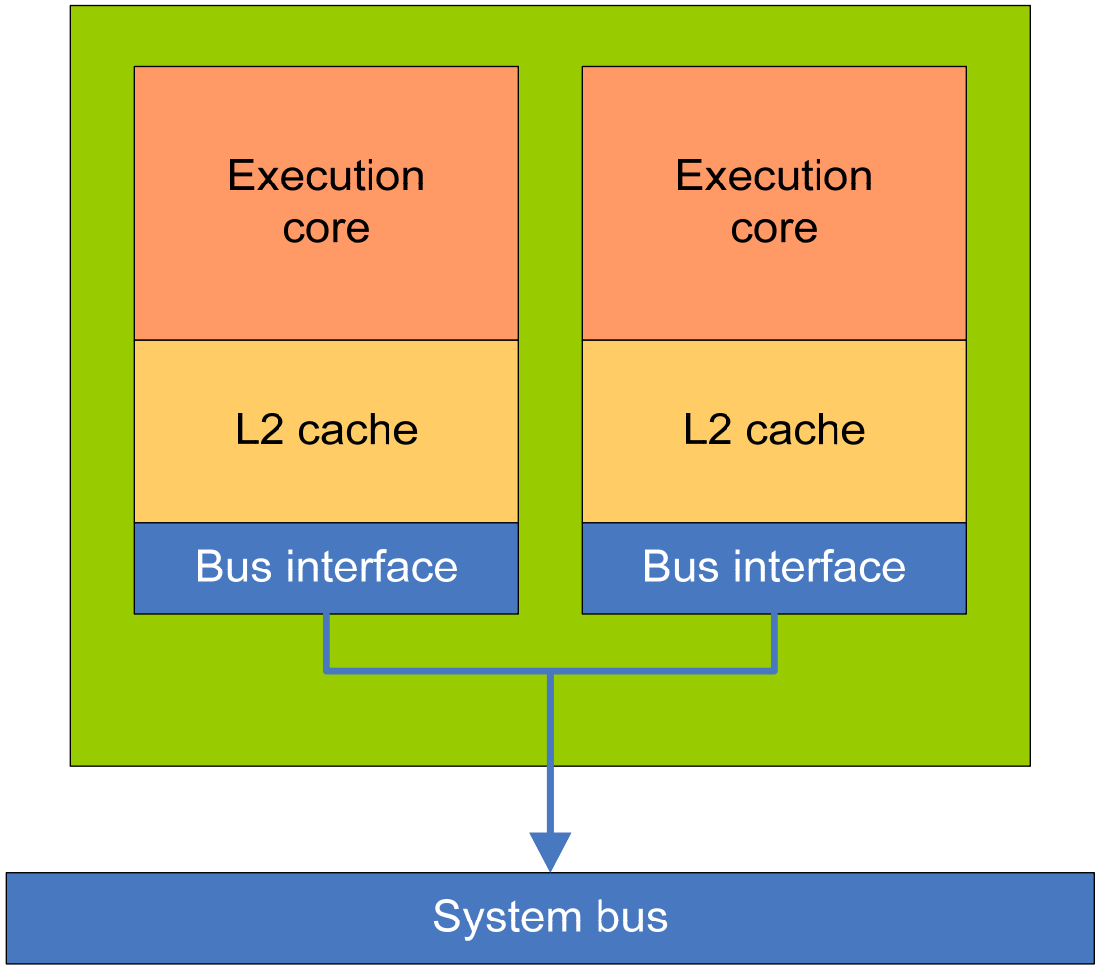
Hardware multi-threading is making a comeback

- **Intel's Montecito (Itanium 2) – up to 35% improvement cited**
  - 2 cores, 2 non-simultaneous threads per core ("temporal multithreading")
  - Switches over to the other thread in case of a high latency event (i.e. page fault)
- **Sun's Niagara 1 (UltraSPARC T1)**
  - 8 cores, 4 non-simultaneous threads per core
  - More fine-grained switching – after every instruction
  - A thread is skipped if it triggers a large latency event
  - Single thread slower, but throughput very high
- **Sun's Niagara 2 (UltraSPARC T2)**
  - 8 cores, 8 non-simultaneous threads per core
  - MySQL compiled with Sun's compiler runs 2.5x faster than with gcc –O3
- **Intel chips from Nehalem (Core 3 – Q4 '08) onwards will feature Hyper Threading**
  - Simultaneous multi-threading (only on superscalar CPUs)
  - Takes advantage of instruction level parallelism

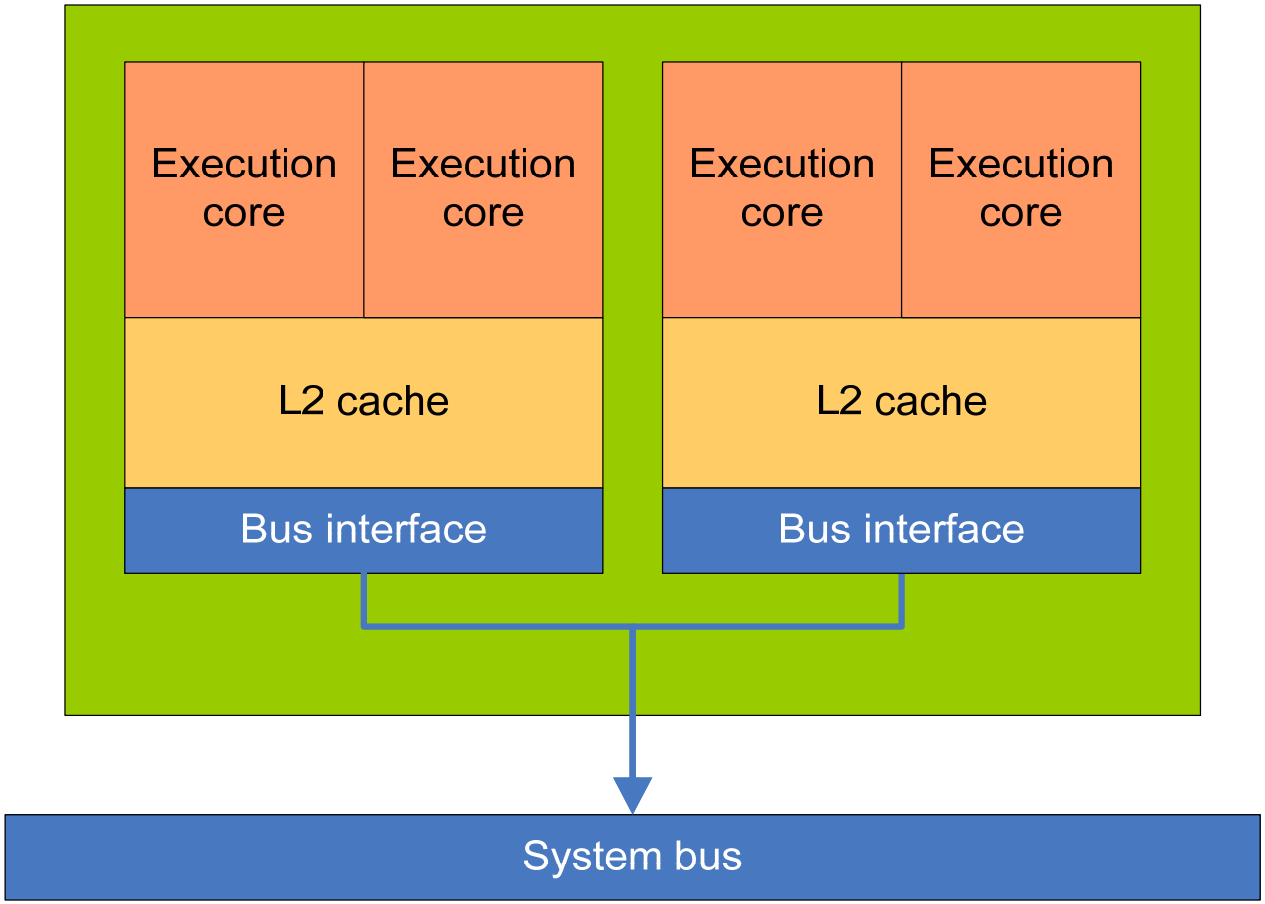
BASIC MULTI-CORE ARCHITECTURES

Common goods

Multi-core architectures – Intel Pentium D

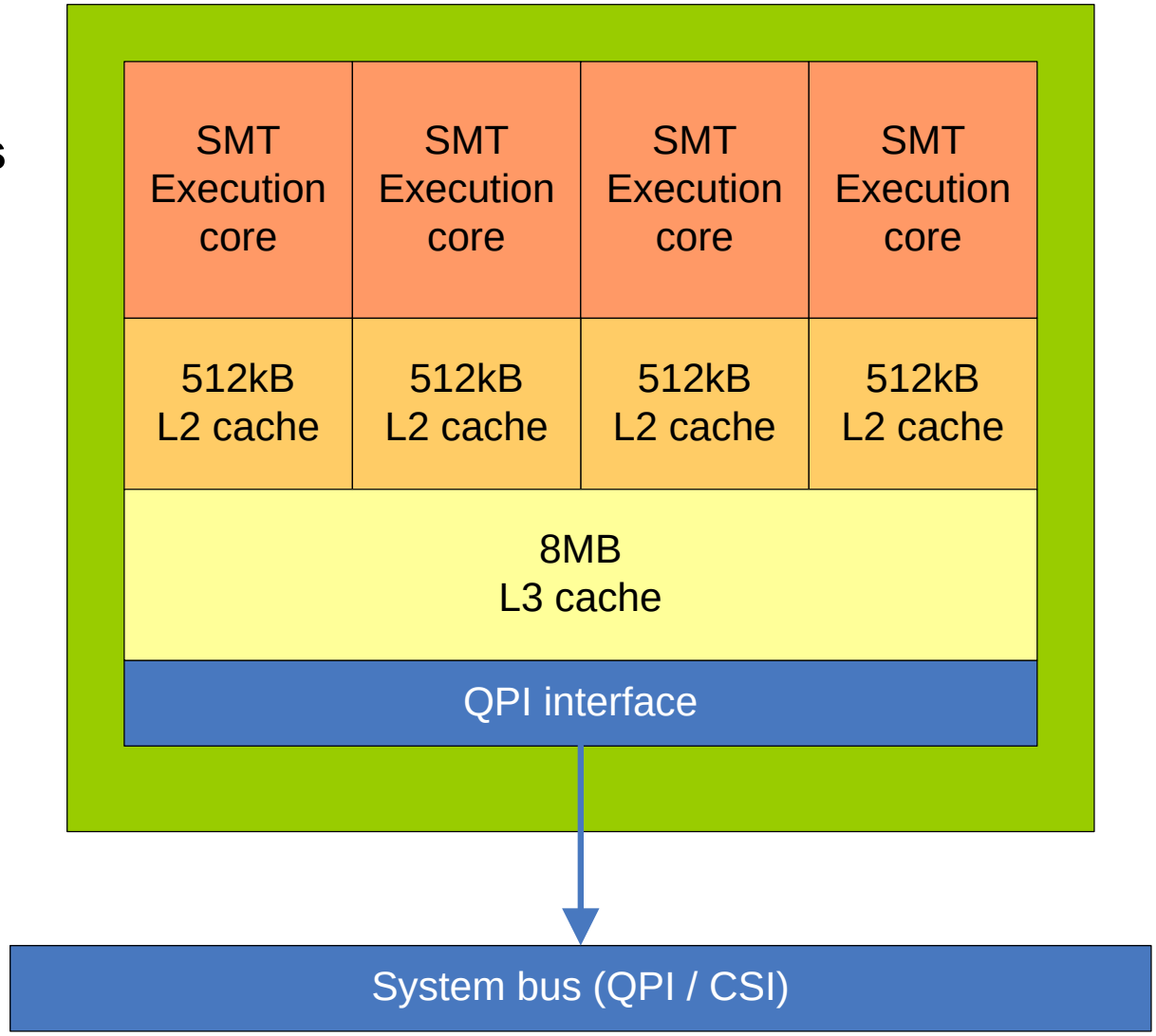


Multi-core architectures – Intel Core 2



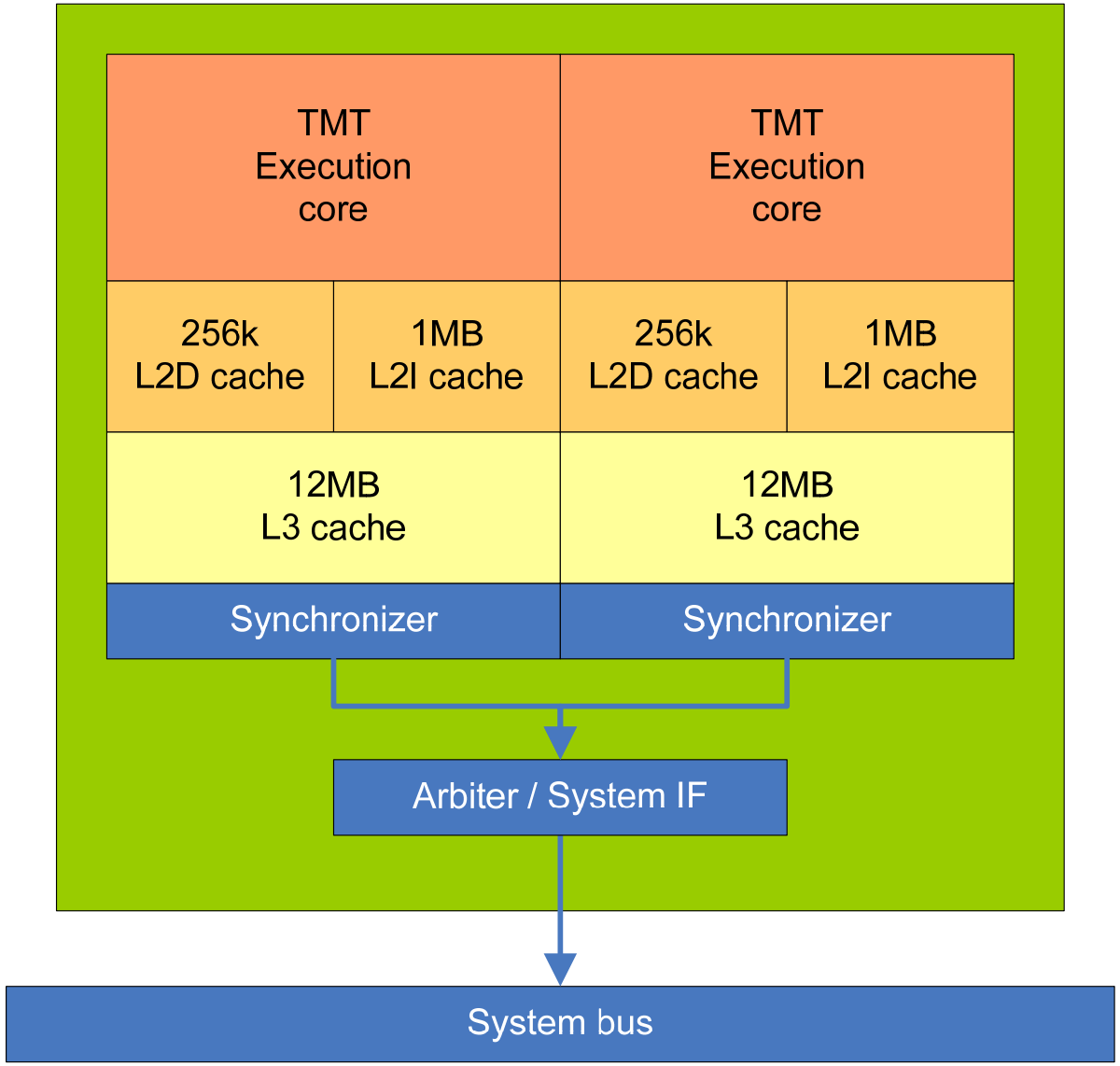
Multi-core architectures – Intel “Nehalem”

- **Release: YE 2008**
- **4-8 cores, 2 SMT threads per core**
- **Next generation interconnect (QPI)**
- **Advanced cache management**
- **Exclusive L2 and shared L3 caches**



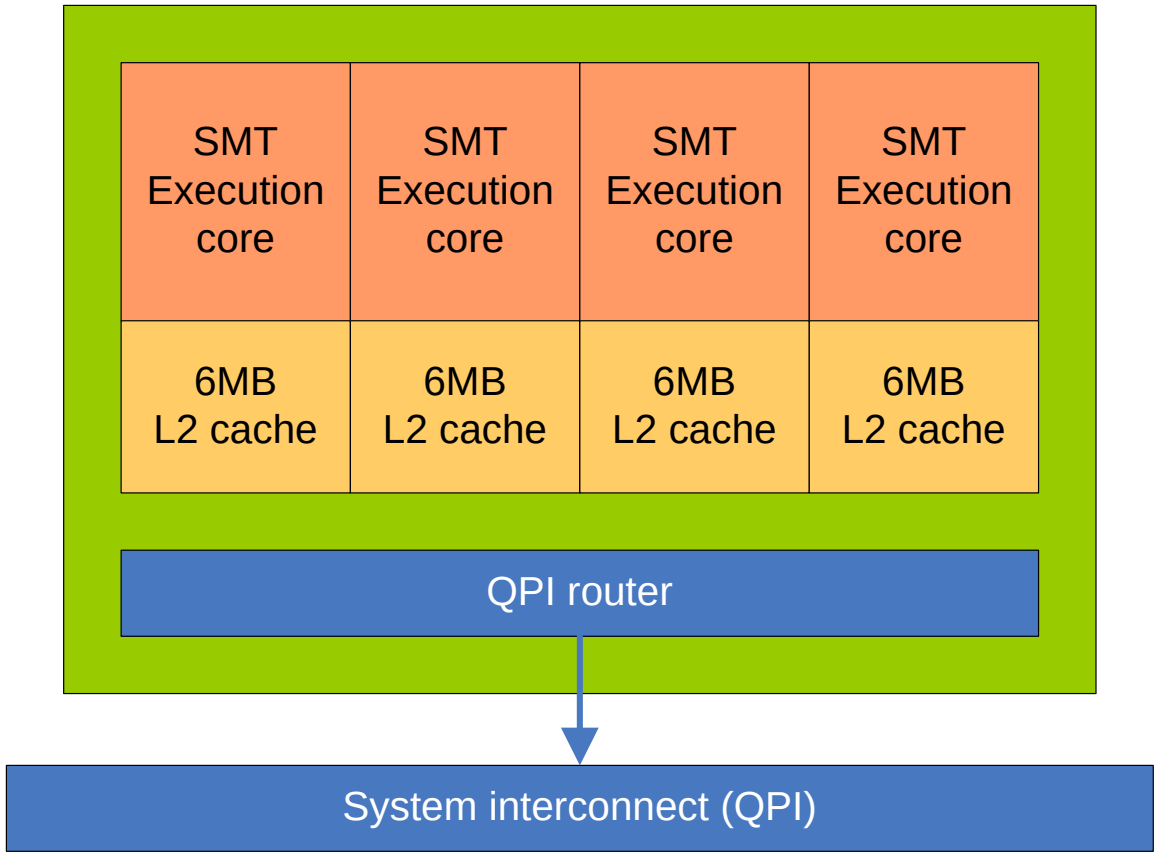
Based on undisclosed data, might vary from actual product

Multi-core architectures – Intel Itanium 2 (“Montecito”)



Multi-core architectures – Intel Itanium 3 (“Tukwila”)

- **Release: YE2008**
- **Estimated 40GFlops / socket @ 2.5 GHz**
- **24MB L2 cache**
- **Next generation interconnect (QPI)**
- **30% improvement over “Montecito” (Itanium 2) per core**
- **Socket compatible with Xeon MP**



ADVANCED MULTI-CORE ARCHITECTURES

Just a taste

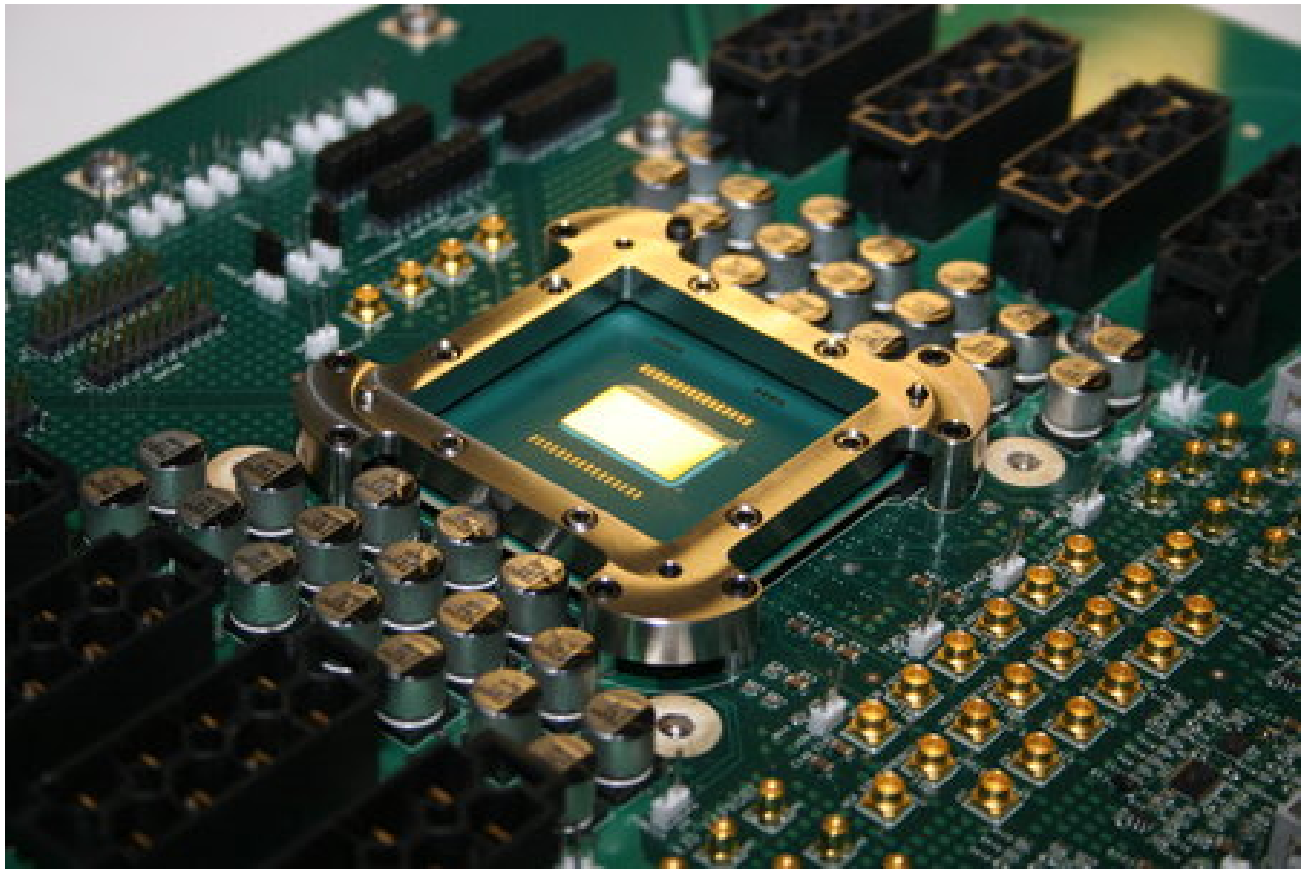
Multi-core architectures – Intel Larrabee

- **45nm process**
- **1.7 – 2.5 GHz, > 150W**
- **2 memory controllers, 2 texture samplers**
- **16-24 in order cores for pixel/vertex shading**
- **LRB core characteristics:**
  - 4 threads
  - capable of 2 double-precision FP ops per cycle
  - 32kB L1 cache, 256kB L2 cache
  - Rumor has it that the architecture is based on the Pentium MMX

SPECULATIVE INFORMATION. Source: ArsTechnica

Multi-core architectures – Intel Polaris

- **80 cores**
- **~1 TFLOPs @ ~50 Watts, ~2 TFLOPS @ ~200 Watts**



Multi-core architectures – NVIDIA G80

- **128 stream processors**
- **330 GFlops (today's general purpose CPUs have ~10)**
- **150W**
- **Top of the line graphics hardware (along with the G92)**

Multi-core architectures – IBM Cell

- **Joint SONY / IBM / Toshiba design used primarily in the Playstation 3 gaming console**
- **Central PowerPC based processing element**
  - 2 threads
  - RISC, in order
  - AltiVec
- **A number of additional vector processing elements (SPEs)**
  - In general: 8 available
  - In the PS3: 6 available + 1 for the OS + 1 locked
- **CELL based servers and blades are available on the market**
- **IBM is building a CELL based supercomputer – Roadrunner**

Multi-core architectures – IBM Xenon

- **Used in the XBOX 360**
- **Three cores**
  - 3.2 GHz
  - 2 threads (SMT)
  - VMX SIMD instructions
  - In-order
- **Up to 116 GFLOPS cited**

Towards many-core – hardware summary

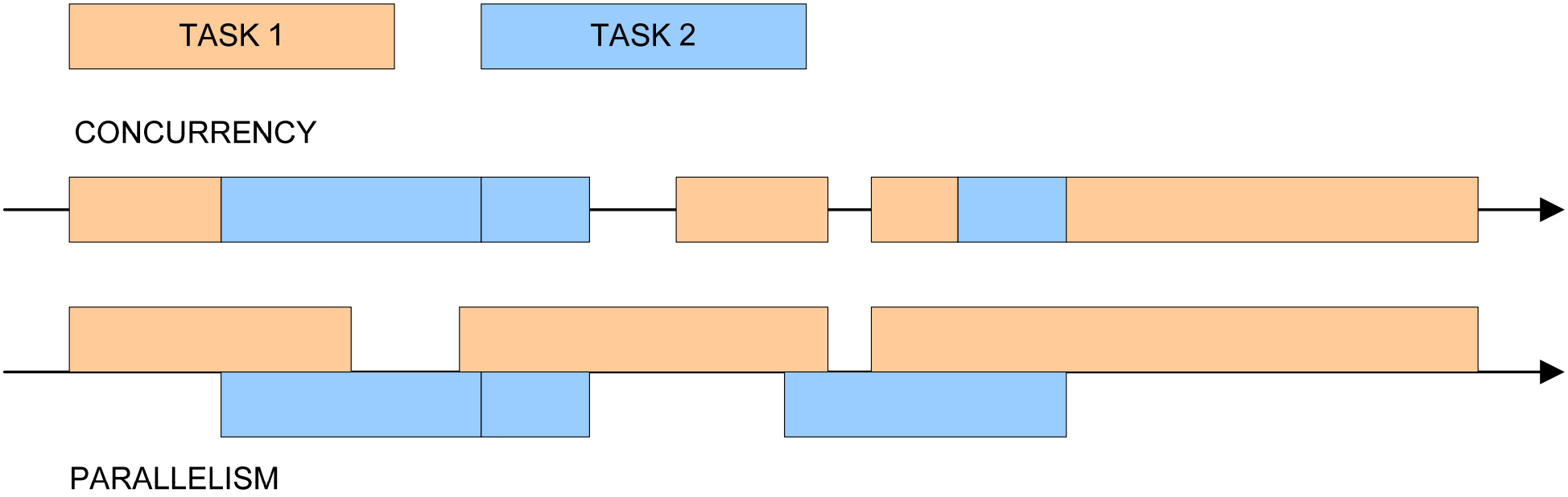
- **Sun's "Niagara 2"**
  - 64 threads running concurrently
- **Intel's Nehalem:**
  - Up to 8 cores and 16 simultaneous threads
- **Intel's Larrabee:**
  - 16-24 cores, 4 threads per core (speculated)
- **Pat Gelsinger, Intel VP on Larrabee:**
  - "different versions of the chip will have a different number of cores"

TAKE ALL YOU WANT

Eat all you take

Common parallelism concepts (1)

- **Concurrency is not parallelism**



- **Synchronization and race conditions**
- **Sharing and exclusion**
- **Deadlock and livelock**
- **Producing, consuming and starvation**

Common parallelism concepts (2)

- **Turnaround – complete one task as soon as possible**
  - Example: Reviewing a paper, each of the three collaborators reads one part
- **Throughput – complete the most tasks in a given amount of time**
  - Example: PC farm
- **Capabilities – better or more detailed results**
- **Efficiency – keeping the hardware busy**
  - Load balancing is important
- **Granularity – at which level is the work being split?**
  - Consider the overhead costs

Amdahl's Law (1)

- **Your principal enemy…**
  - …"keep your friends close, and your enemies closer"
- **Describes the relationship between the parallelized portion of code and the expected speedup**
  - P – parallelized portion
  - S – parallelized portion speedup

$$Speedup = \frac{1}{(1-P) + \frac{P}{S}}$$

Amdahl's Law (2)

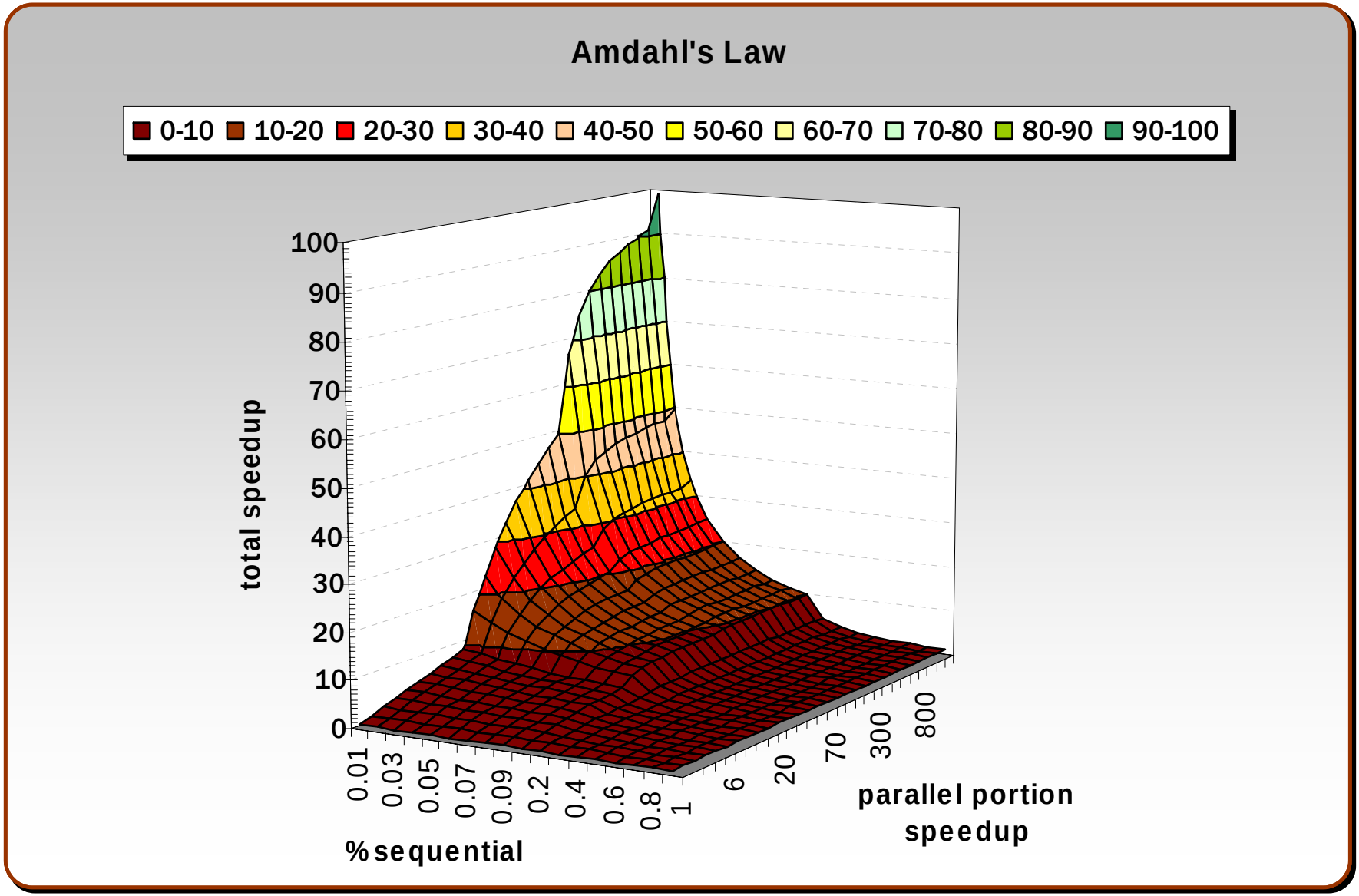
- **Baking a delicious cake**
  - Preparing the ingredients – 40% of the time (parallel)
  - Baking the cake – 60% of the time (sequential)
- **Cake process speedup with 4 people:**

$$S_4 = \frac{1}{(1-0.4)+\frac{0.4}{4}} = 1.43$$

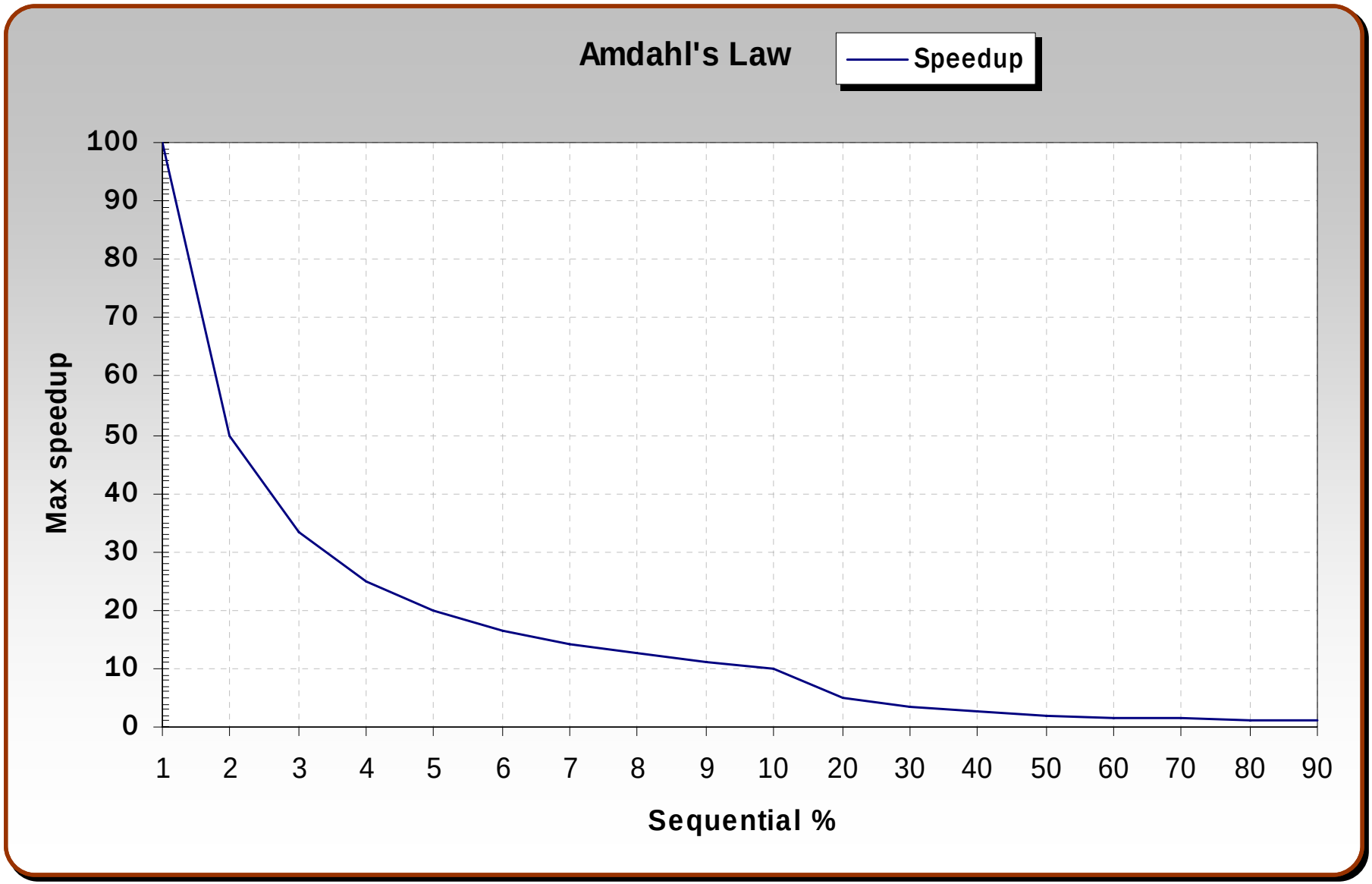
- **Maximum cake process speedup according to Amdahl's law:**

$$S_{\max} = \frac{1}{(1-0.4)+\frac{0.4}{\infty}} = 1.66$$

Amdahl's Law (3)



Amdahl's Law (4)



Gustafson's Law

- **Often when the problem size increases, the sequential portion remains constant**
- **Therefore, as the problem size increases, so do the opportunities for parallelization**
- **Let *a(n)* be the sequential portion function of the program, diminishing as *n* approaches infinity**

$$Speedup = a(n) + N(1 - a(n))$$

- **As *n* approaches infinity, the speedup approaches the number of processors N**

Levels of parallelism

- **Multistage pipelines**
- **ILP – instruction level parallelism (multiple specialized pipelines)**
  - Superscalar computers – i.e. the Core 2 can multiply, load and store at the same time
- **Thread concurrency**
  - Multiple CPUs
- **Multi-core**
- **Process concurrency (SMP – different sockets)**
  - Multiple processes on multiple CPUs
- **Cluster computing**
  - Multiple concurrent computing systems
- **Grid computing**
  - Heterogeneous environment

Parallel and multi-core programming paradigms

- **Scatter-gather algorithms**
- **Transactional memory**
- **Shared and exclusive sections of code**
- **Shared and exclusive sections of memory**
- **Vector processing**
- **Maximizing SIMD benefits**

Parallelism in popular languages

- **C, C++:**
  - "External" threading libraries – posix threads, linux threads, windows threads
- **Java, C#:**
  - Native threading
  - Some higher level tools
- **Python:**
  - Threading modules dependent on the underlying OS
- **Common traits:**
  - manual synchronization needed, low level, fine grained control
  - all of the techniques from the previous slide have to be implemented and controlled manually

Multi-core programming technologies

- **Extensions and libraries for C, C++:**
  - OpenMP
  - MPI
  - PVM
  - Intel Threading Building Blocks
- **New programming languages and methods**
  - Intel Ct (vector computing)
  - Intel STM (transactional memory)
- **How do the parallel programming paradigms map onto these languages?**

Implications

- **How do you feed 80 hungry cores?**
- **Parallelism – fine grained or coarse?**
- **Effective virtualization**
- **Memory access and bus optimization**
- **Resource sharing**

How to take advantage of multi-core architectures?

- **Know your hardware**
- **Know your software tools**
  - Compilers – they're not all the same
  - Intel Threading Tools – Thread Checker, Thread Profiler, VTune
  - Performance monitoring utilities – gprof, perfmon, Java profilers
- **Know your own software**
  - How much memory does it need?
  - How well does it scale?
  - Is it multi-threaded? Should it be multi-threaded?
  - Is it portable or architecture dependent?

Questions for the future

- **How many cores does your family need?**
- **How many cores do you, a scientist, need?**
- **How do you effectively use what you have?**
- **What is the best level to introduce parallelism? Do you need to redesign your software?**
- **GRID computing or tera-scale homogenous computers? Will virtualization be effective enough?**

Multi-core prospects

- **Multi-core designs will continue to dominate the computing landscape for at least several years**
- **The amount of “available” cores is increasing more rapidly than the amount of “available” memory**
- **The transition from single-threaded to multi-threaded software development is not easy nor pleasant, but necessary**

The multi-core tradeoff

Large amounts of computing power will be available at your disposal, but an effort will be needed in order to put them to use

CERN School of Computing

Q&A