CS260 – Lecture 1
Yan Gu

Algorithm Engineering
(aka. How to Write Fast Code)

Introduction to the course

Many slides in this lecture are borrowed from the first lecture in 6.172 Performance Engineering of Software Systems at MIT. The credit is to Prof. Charles E. Leiserson, and the instructor appreciates the permission to use them in this course.

CS260: Algorithm Engineering Lecture 1

Why care performance?

Introduction to modern computing system

Course policies

Software Properties

- **There are many things that are also important in programming**
  - Compatibility, functionality, reliability, correctness, debuggability, robustness, portability, … and more

- **If the programmers are willing to sacrifice performance for other properties, why study performance?**

Time is money, it buys other things

- **There are many things that are also important in programming**
  - Compatibility, functionality, reliability, correctness, debuggability, robustness, portability, … and more

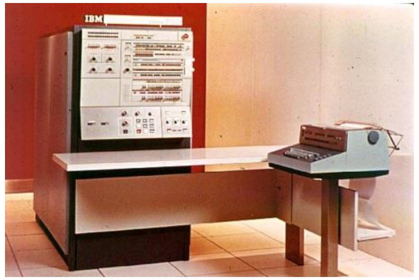
- **Performance is the currency of computing. You can often "buy" needed properties with performance**

- **Better performance means to get better results in a limited amount of time**
  - For an iterative numerical algorithm, spending more time means better accuracy
  - For a learning algorithm, training for more time means better model

Computer Programming in the Early Days

Performance optimization and engineering were common, because machine resources were limited

IBM System/360



Launched:	1964
Clock rate:	33 KHz
Data path:	32 bits
Memory:	524 Kbytes
Cost:	$5,000/month

DEC PDP-11

Launched:	1970
Clock rate:	1.25 MHz
Data path:	16 bits
Memory:	56 Kbytes
Cost:	$20,000

Apple II

Launched:	1977
Clock rate:	1 MHz
Data path:	8 bits
Memory:	48 Kbytes
Cost:	$1,395

Many programs strained the machine’s resources

- Programs had to be planned around the machine
- Many programs would not “fit” without intense performance engineering

Lessons Learned from the 70's and 80's

Donald Knuth

Premature optimization is the root of all evil. [K79]

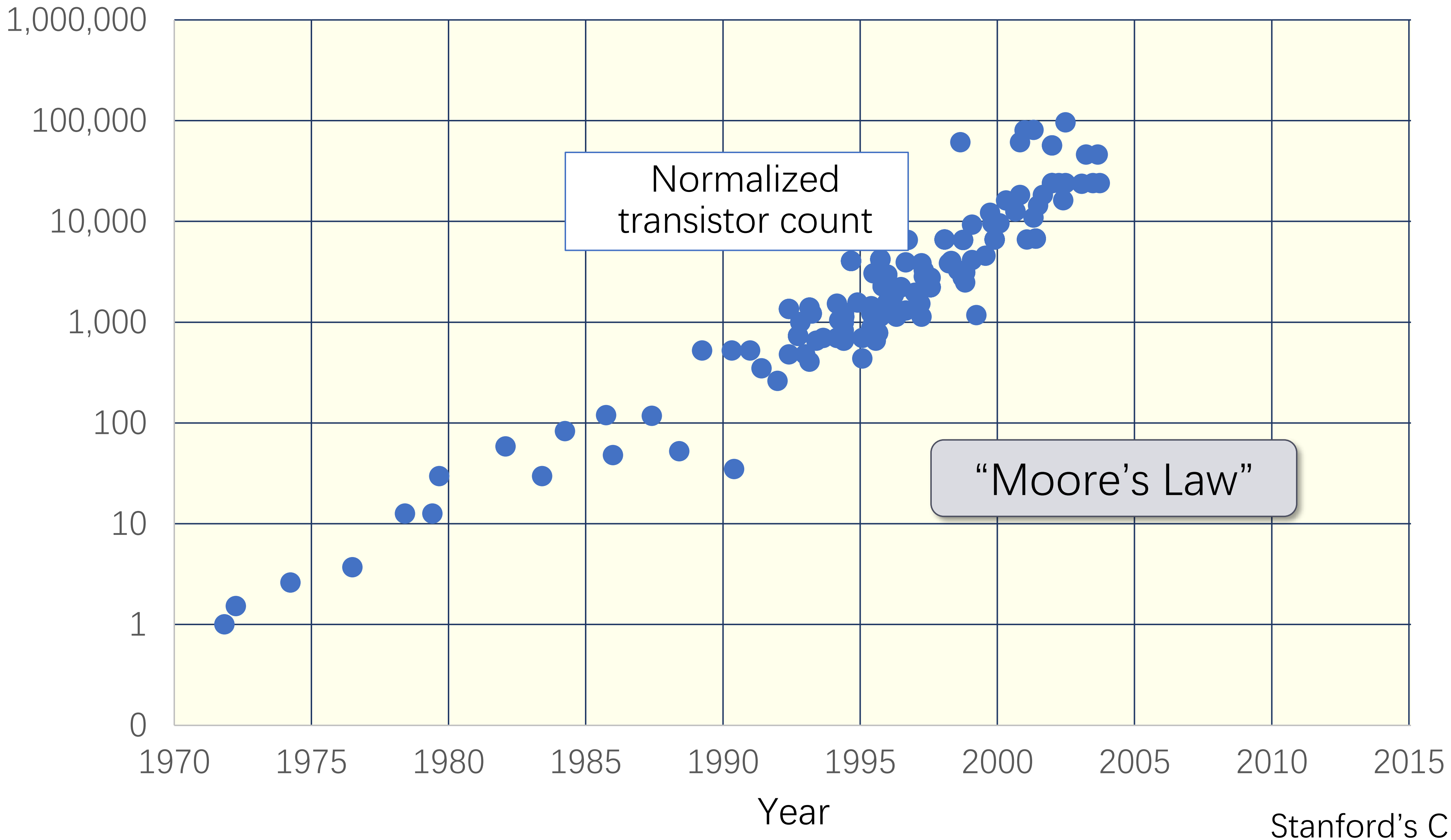
More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason — including blind stupidity. [W79]

William Wulf

Michael Jackson

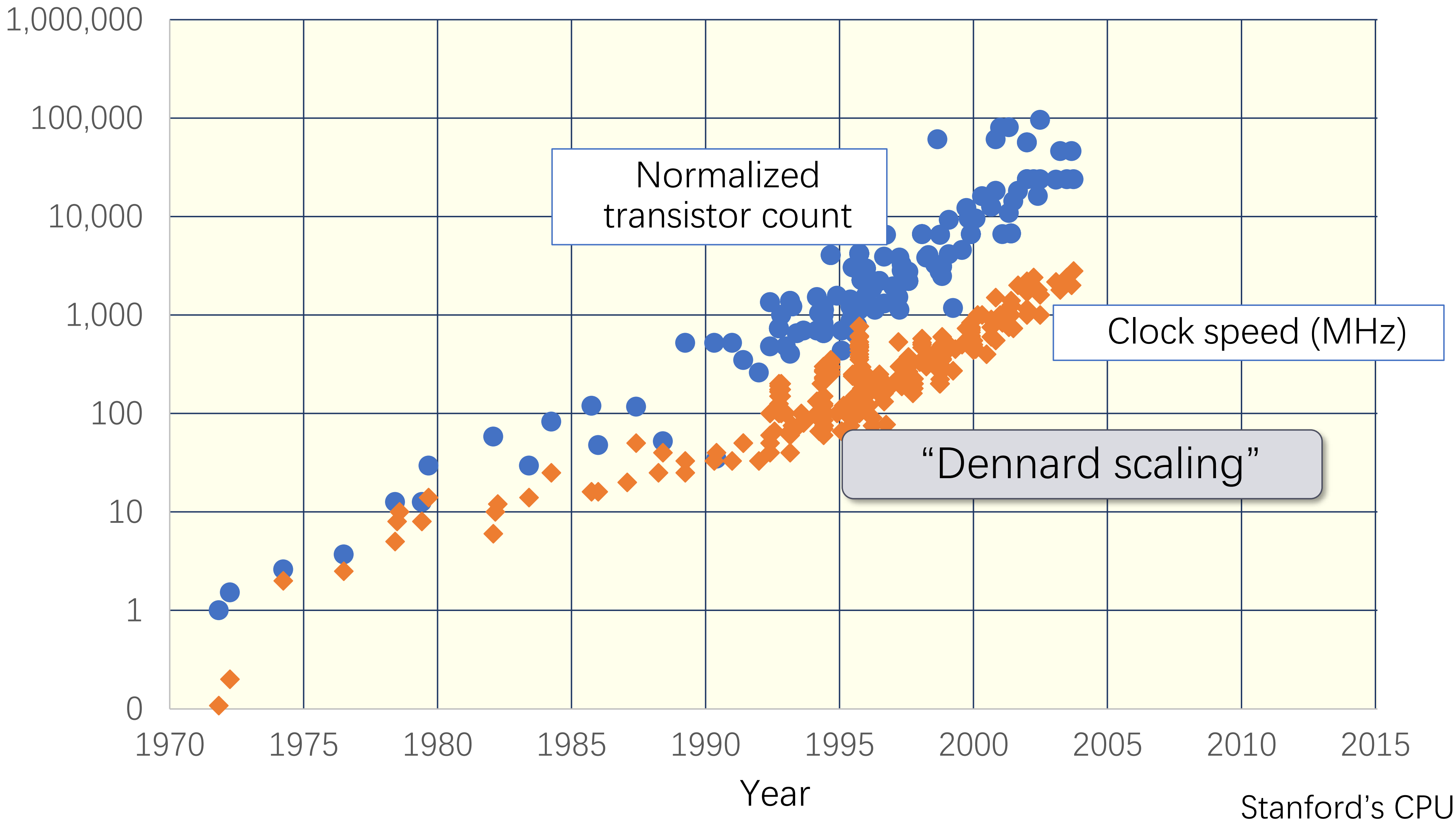
The First Rule of Program Optimization: Don't do it.
The Second Rule of Program Optimization — For experts only: Don't do it yet. [J88]

Technology Scaling Until 2004



Stanford's CPU DB [DKM12]

Technology Scaling Until 2004



Stanford’s CPU DB [DKM12]

Advances in Hardware

Apple computers with similar prices from 1977 to 2004

Apple II

Launched:	1977
Clock rate:	1 MHz
Data path:	8 bits
Memory:	48 KB
Cost:	$1,395

Power Macintosh G4

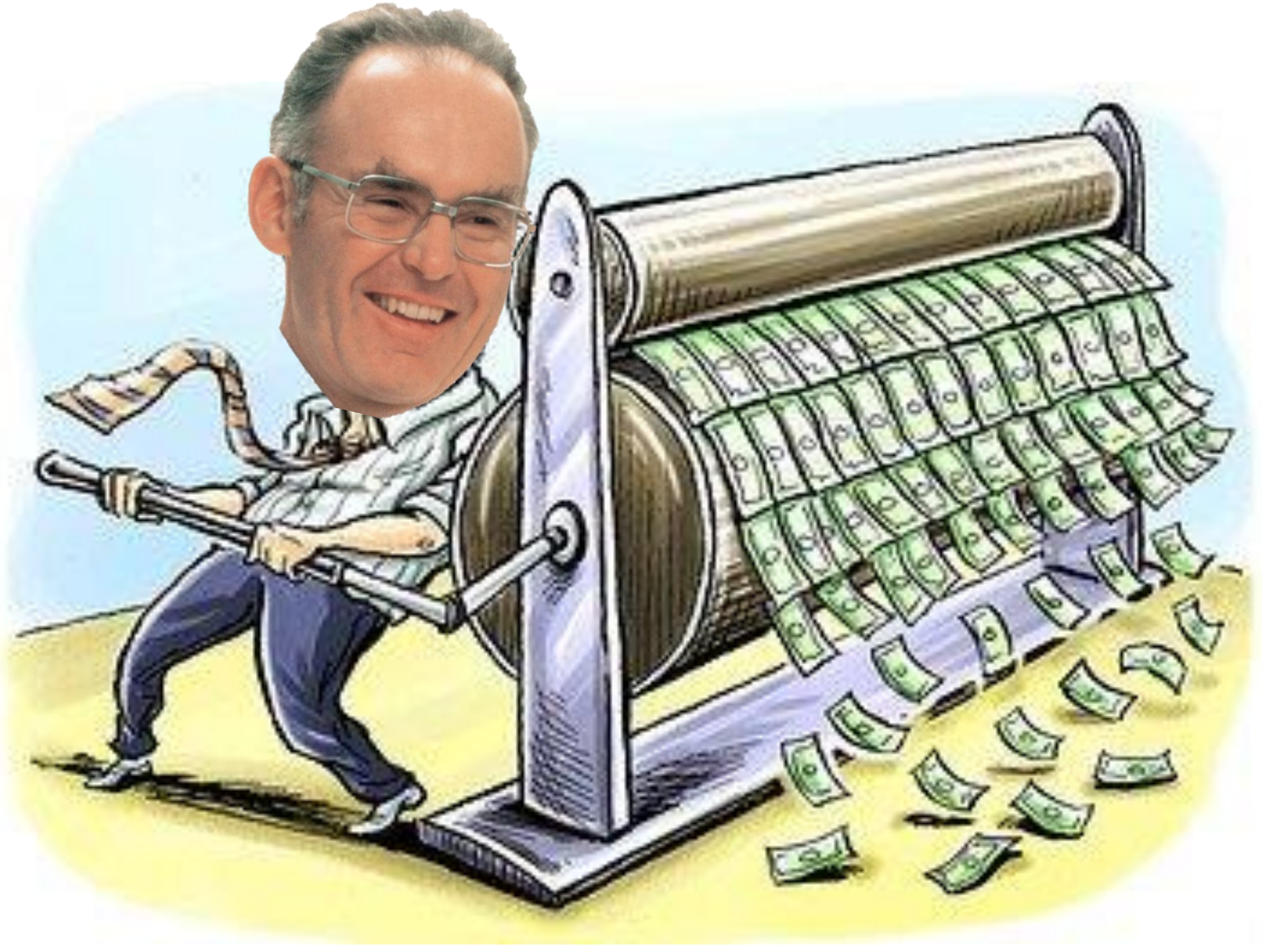
Launched:	2000
Clock rate:	400 MHz
Data path:	32 bits
Memory:	64 MB
Cost:	$1,599

Power Macintosh G5

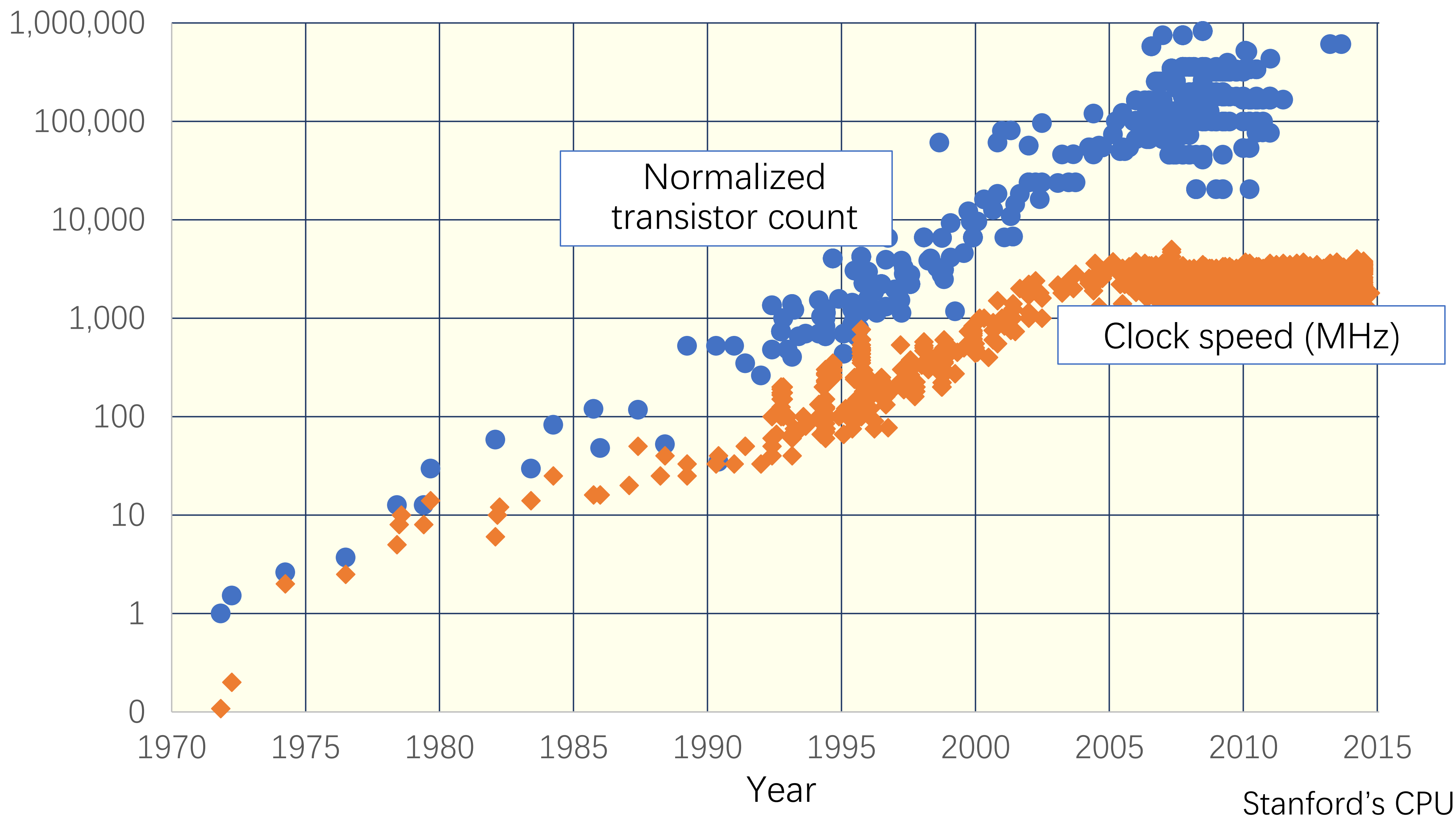
Launched:	2004
Clock rate:	1.8 GHz
Data path:	64 bits
Memory:	256 MB
Cost:	$1,499

Until 2004

Moore's Law and the scaling of clock frequency
= printing press for the currency of performance

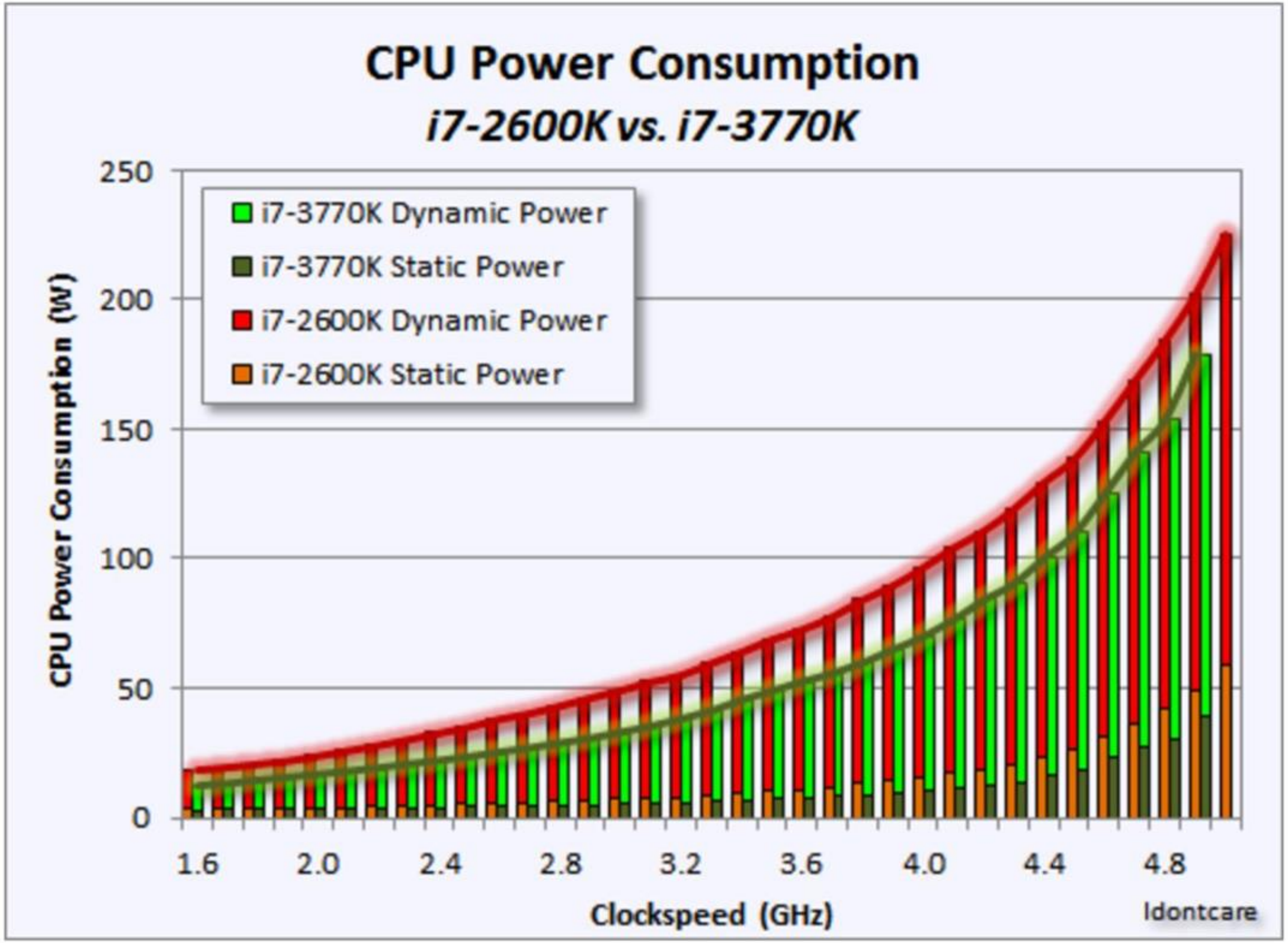


Technology Scaling After 2004



Stanford's CPU DB [DKM12]

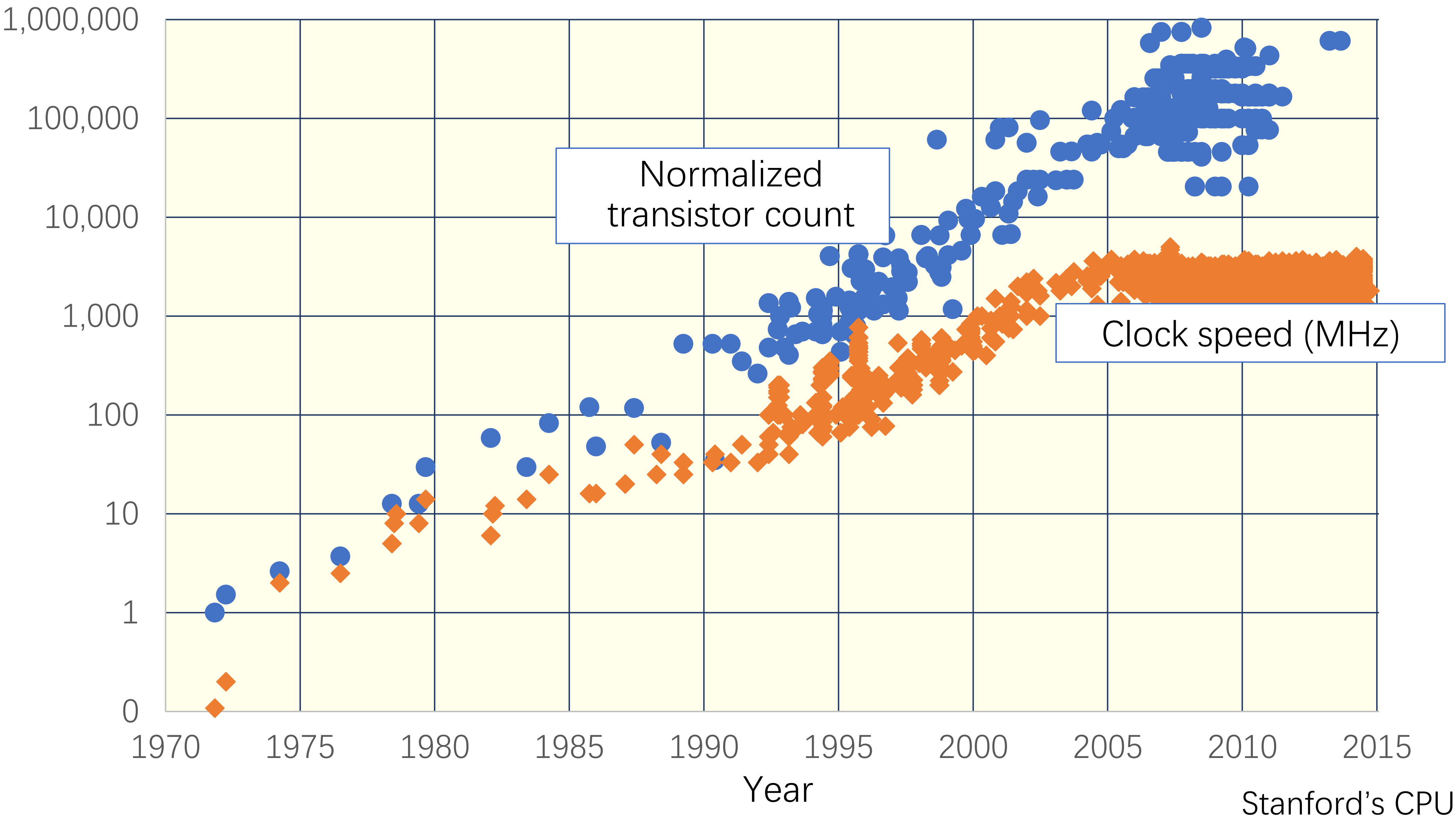
Power Density



- **Dynamic power ∝ capacitive load × voltage^2 × frequency**
- **Static power: maintain when inactive (leakage)**
- **Maximum allowed frequency determined by processor's core voltage**

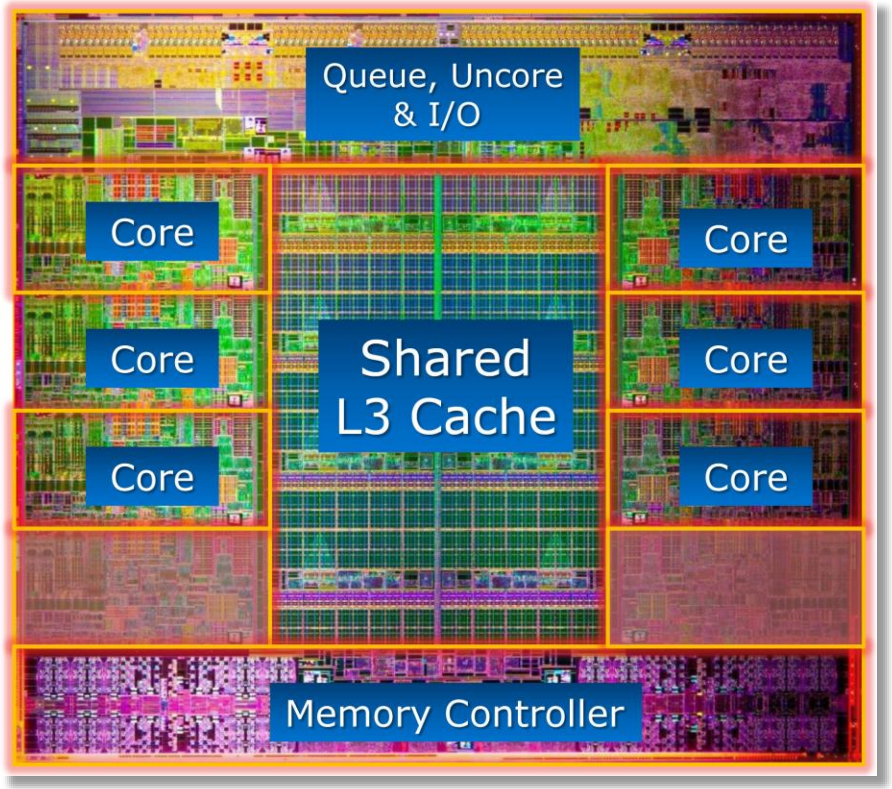
Image credit "Idontcare" from forums.anadtech.com

Technology Scaling After 2004



Stanford's CPU DB [DKM12]

Vendor Solution: Multicore

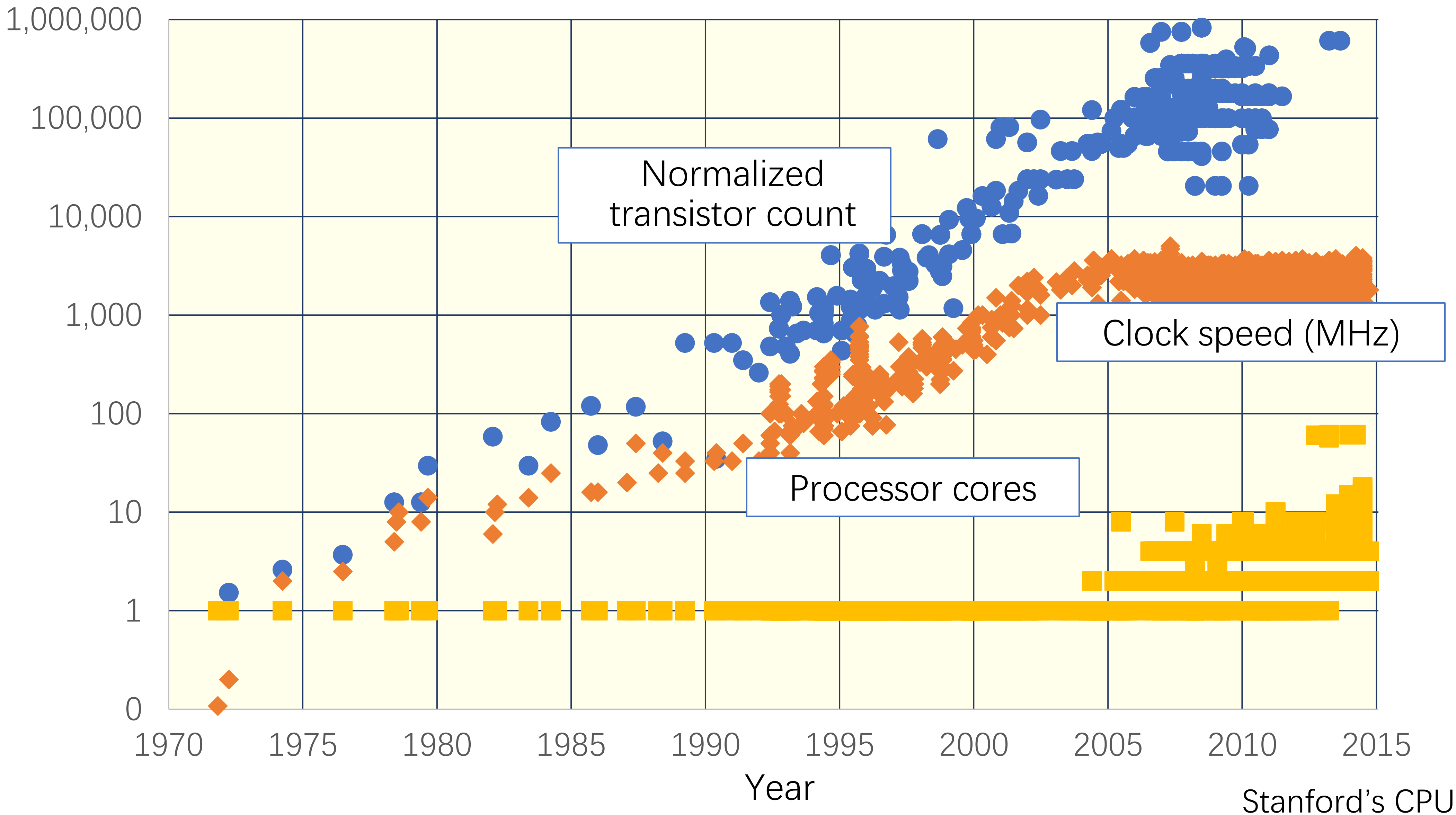


Intel Core i7 3960X (Sandy Bridge E), 2011

- 6 cores / 3.3 GHz / 15-MB L3 cache

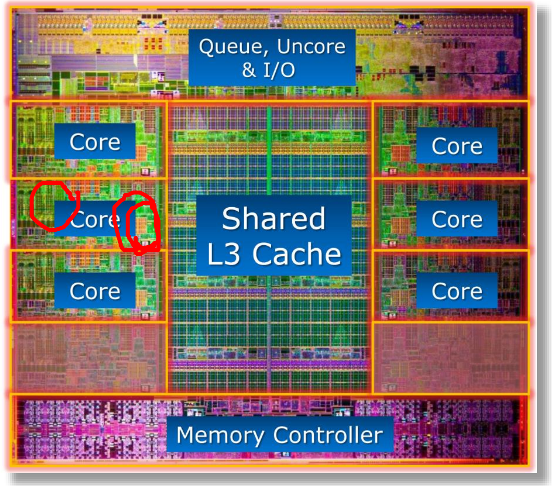
- To scale performance, processor manufacturers put many processing cores on the microprocessor chip
- Each generation of Moore's Law potentially doubles the number of cores

Technology Scaling

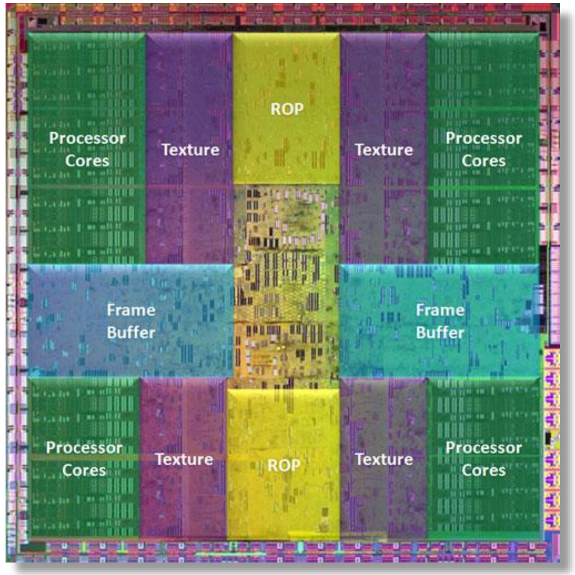


Stanford's CPU DB [DKM12]

Performance Is No Longer Free



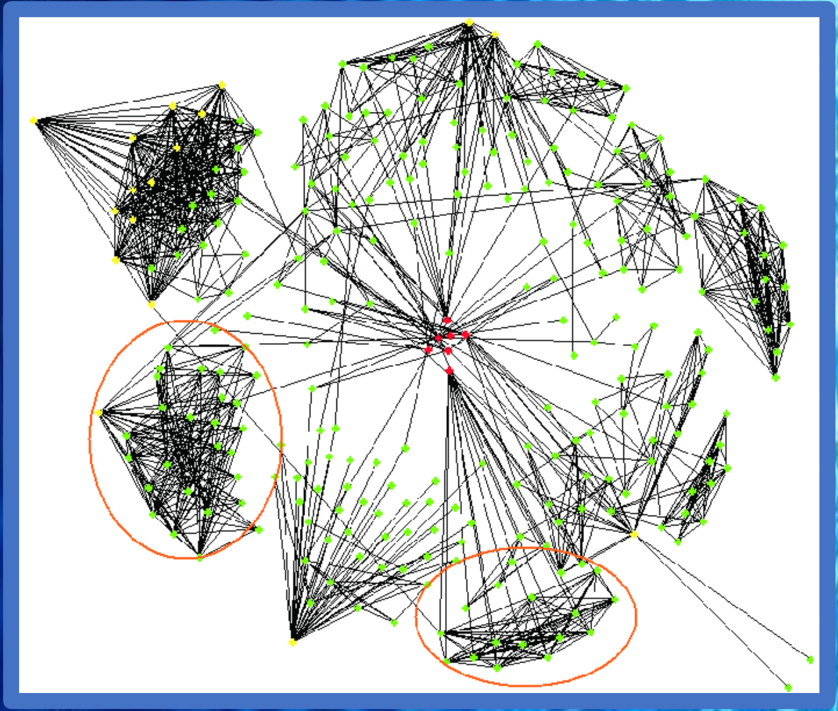
2011 Intel Skylake processor

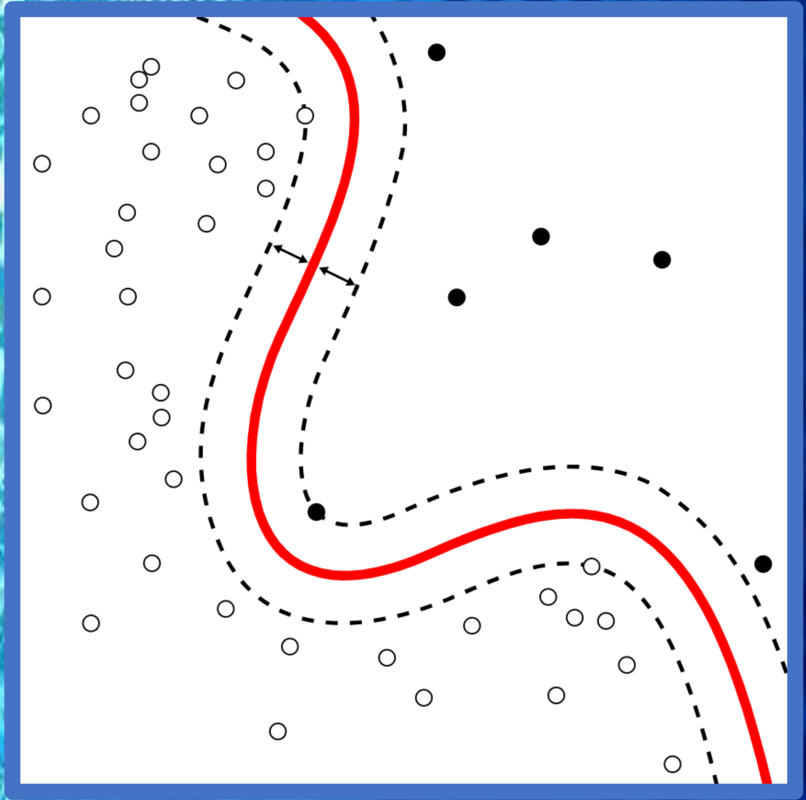


2008 NVIDIA GT200 GPU

- **Moore's Law continues to increase computing ability**
- **But now that performance looks like big multicore processors with complex cache hierarchies, wide vector units, GPUs, FPGAs, etc.**
- **Generally, algorithms must be adapted to utilize this hardware efficiently!**

Data





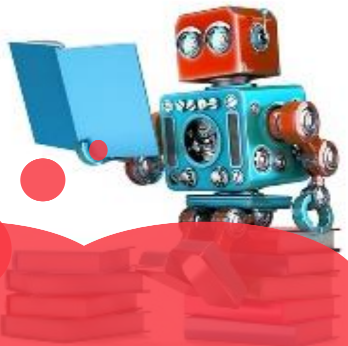
The data size can easily reach hundreds GB to TB level

Everyone wants performance!

Database / Data warehouses

Data mining / Data science

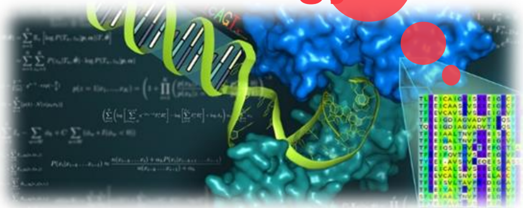
Machine learning / Artificial intelligence



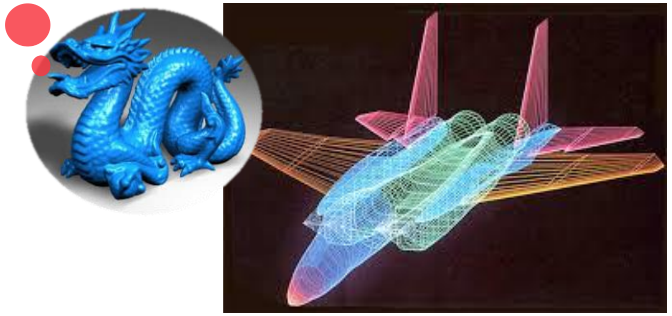
Get Faster!

Many, many others

Computational biology

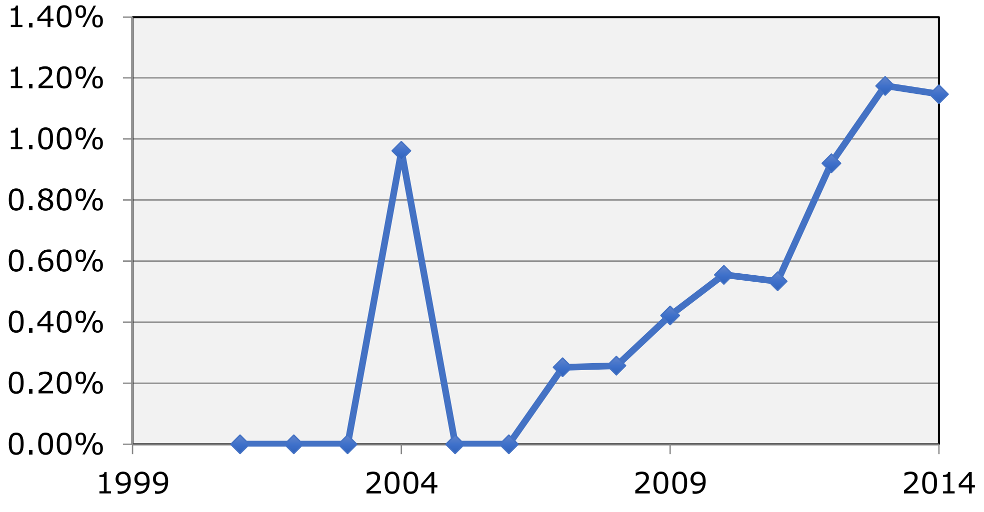


Computer graphics / computational geometry

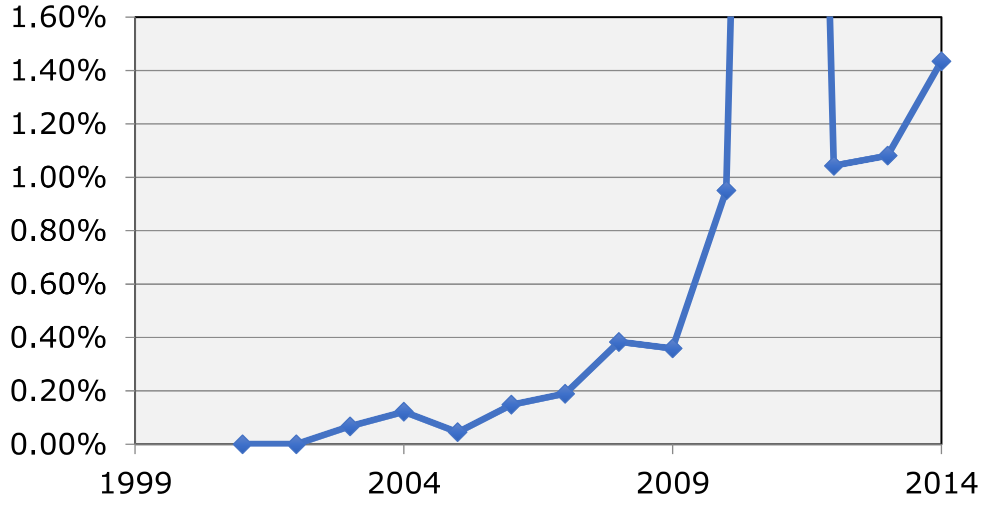


Software Bugs Mentioning "Performance"

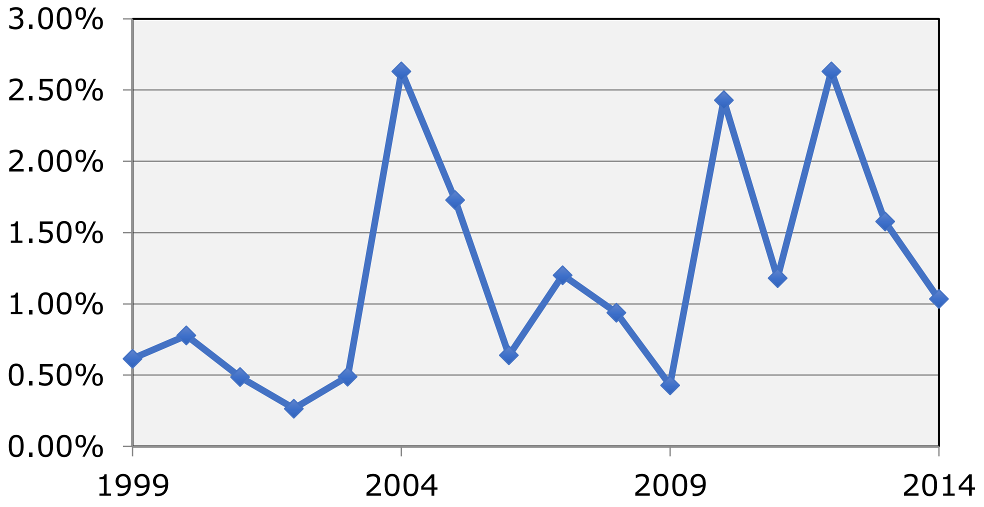
Bug reports for Mozilla "Core"



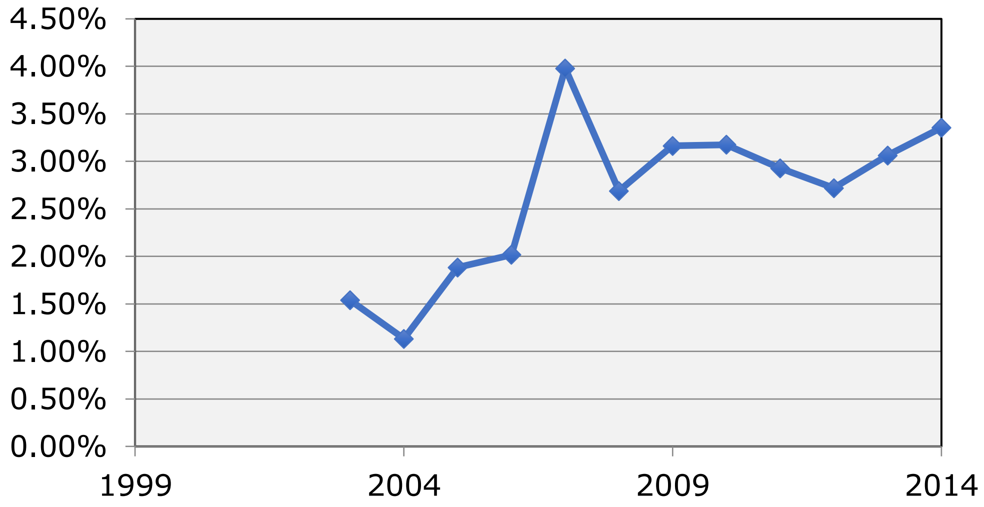
Commit messages for MySQL



Commit messages for OpenSSL

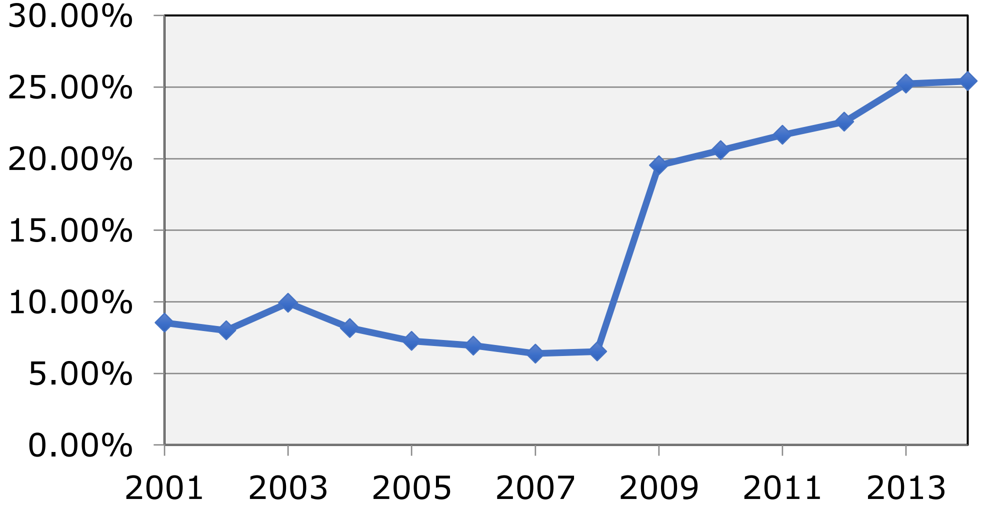


Bug reports for the Eclipse IDE

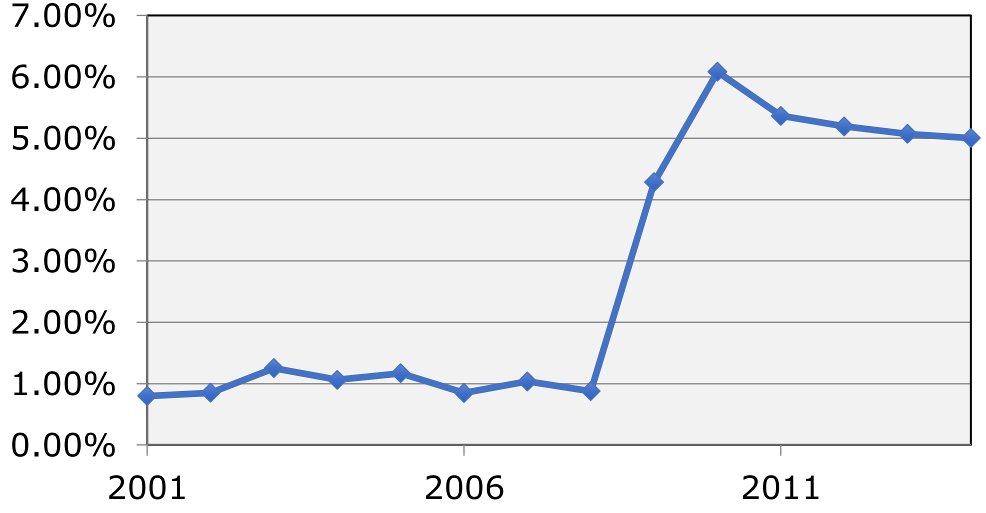


Software Developer Jobs

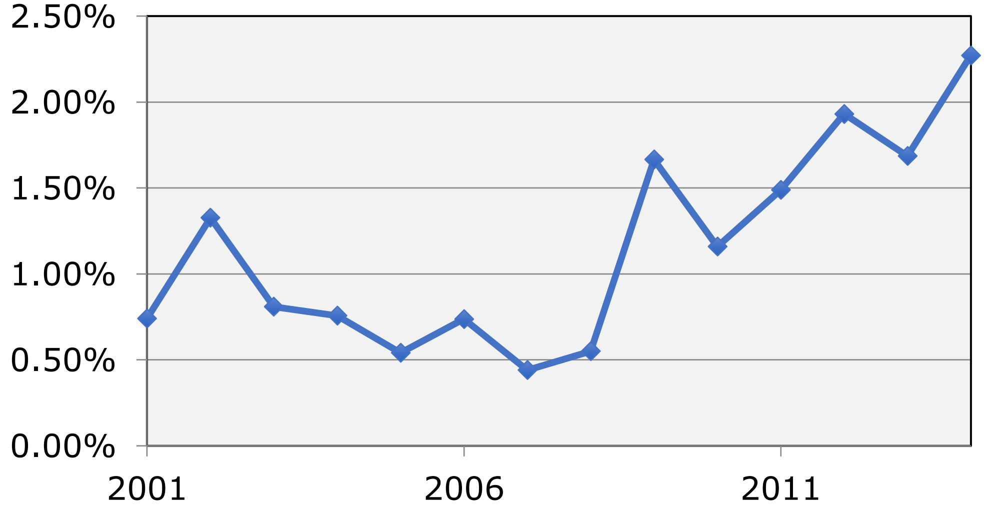
Mentioning “performance”



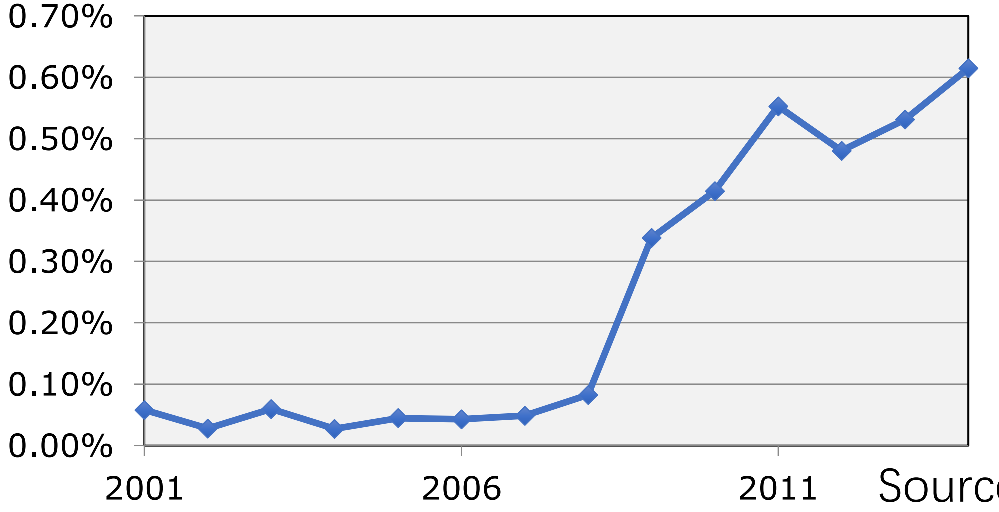
Mentioning “optimization”



Mentioning “parallel”



Mentioning “concurrency”

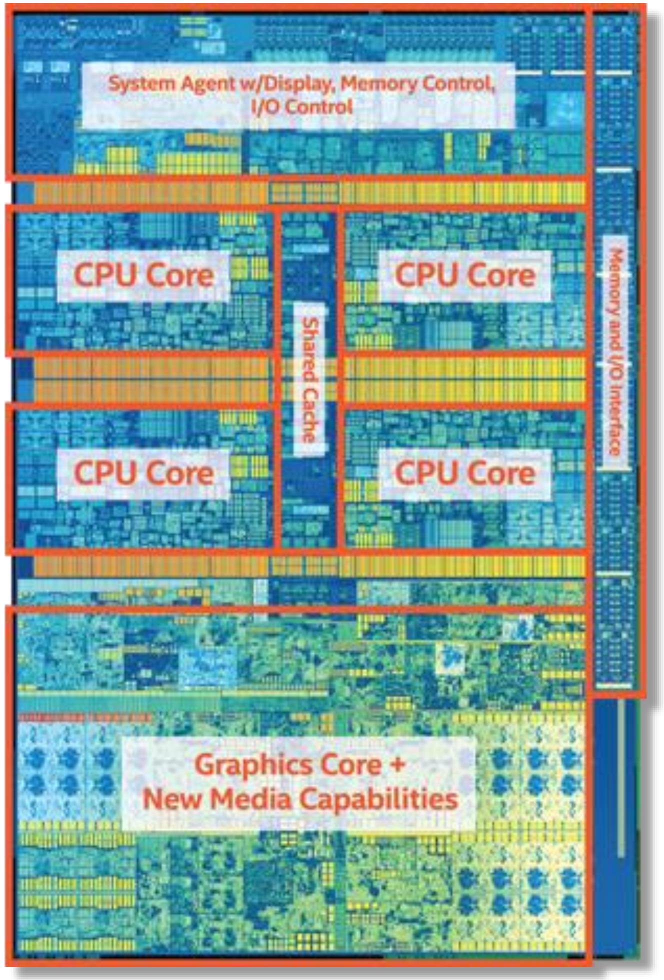


Source: Monster.com

Algorithm Engineering Is Still Hard

- A modern multicore desktop processor contains **parallel-processing cores, vector units, caches, prefetchers, GPU's, hyperthreading, dynamic frequency scaling**, etc.

- **How can we write algorithms and software to utilize modern hardware efficiently?**



2017 Intel 7th-generation desktop processor

Overall Structure in this Course

Performance Engineering

Parallelism
I/O efficiency
New Bentley rules
Brief overview of architecture

Algorithm Engineering

Sorting / Semisorting
Matrix multiplication
Graph algorithms
Geometric algorithms

EE/CS217	GPU Architecture and Parallel Programming
CS211	High Performance Computing
CS213	Multiprocessor Architecture and Programming (Stanford CS149)
CS247	Principles of Distributed Computing

This is a tough course...

- **Level of difficulties is related to course number**
  - Usually 20X, 21X are easier, and 260 has the largest number

- **You need to spend a lot of time in this course, but you can learn useful knowledge from this course**

- **This is a seminar course, and the expected outcome also includes research abilities**

Front-loading the course

- Basically there is nothing much you can do in the first several weeks. I will try to frontload materials so you will have more time for paper reading and the two projects
- Won’t work usually, but might work since we go online
- Two proposals:
  - 3:30-4:50pm
  - 4:00-5:20pm
- The overall lecture time remains the same. 13 lectures taught by me, and many slots remain empty

Logistic

- **Paper Reading - 15%**
- **Course Presentation - 20%**
- **Quiz - 10%**
- **Midterm Project - 20%**
- **Final Project - 35%**
- **Class Participation - 10% bonus**

Paper Reading – 15%

- Here you can find a list of (about 30) related papers, categorized in three topics
- You need to submit paper reviews for two papers
  - Each review should contain no less than 1000 words and no more than 3000 words (figures, tables are encouraged but not counted)
  - Describe the problem the paper is trying to solve, why it is important, the main ideas proposed, and the results obtained

Course Presentation – 20%

- Each of you will give a presentation on one of your reviewed papers
- Each should be 15-20 minutes long with slides, followed by a discussion
- It should discuss the motivation for the problem being solved, any definitions needed to understand the paper, key technical ideas in the paper, theoretical results and proofs, experimental results, and existing work
- It should also include any relevant content from related work that would be needed to fully understand the paper being presented. The presenter should also present his or her own thoughts on the paper, and pose several questions for discussion

Paper Reading and Course Presentation

- **One paper reading is due before your course presentation**
- **The other paper reading is due on May 15**
- **The presenter should send this paper review and a draft version of the slides to Yan at least two days before the presentation, and Yan will provide feedback**
- **Also, you are welcome to talk to Yan at any time**

Quiz – 10%

- A small quiz by the end of Week 4
- When we basically finished the first part of the course
- Only takes 10% of the final score
- The goal of this quiz is to guarantee you understanding the basic knowledge, which will be helpful for your final project

Midterm Project – 20%

- You need to implement one of the designated algorithms, test the performance, and write a formal report
- Topics from:
  - Sample sort (100%)
  - Semisort (110%)
  - Matrix multiplication (90%)
- Due: April 29

Final Project – 35%

- **Proposing and completing an open-ended (research) project**
- **The project will be done in groups of 1-2 people and consist of a proposal, mid-term report, final presentation, and final report**
- **Deadlines:**
  - **Proposal: May 4**
  - **Mid-term report: May 22**
  - **Final presentation: June 1-5**
  - **Final report: June 8**

Class Participation – 10% bonus

- **This is a seminar course: participate in discussion!**

- **This is a one-time course: giving me feedback at the end of the course won’t help---ask questions or provide feedback immediately!**

- **You will waste your time if you are unclear about what I’m saying. Time is the most valuable!**

Office Hour

- **Tentatively 1:30-2:30pm on Wednesday**
  - **However, since the course goes only, we can do a reservation-based office hour**
  - **Send me an email if you want to talk to me this week, and I will reply to you and reserve you a slot**
  - **Ideally during the office hour, but other times are also applicable**

This is a tough course…

- **Please decide if you really want to take this course, or you want to spend more time on LeetCode**
  - **Either case is fine, but don't complain the course load later**
- **Once the roster is fixed, we will proceed to the paper assigning**
- **For those who did not attend CS260 in Winter, it is better to spend some time in looking at the slides**
  - **Assignment 0 is given out for warm up with the course server**