Smith College

Computer Science

Multithreading In Java (2)

CSC352 — Week #4

Dominique Thiébaut
dthiebaut@smith.edu

Comments on Paper Summaries

- Extract information out
- Top-down approach. First paragraph = summary of whole paper. Cite paper in first sentence. Add to bibliograpy
- List the main points. Bullets are ok
- Develop one or two points
- User present tense
- Use *italics* first time a concept is introduced

Comments on Paper Summaries (cont'd)

- Don't get stuck in the details of the paper
- Two back quotes for opening double-quotes
- Follow the organization of the paper
- You may give your impressions/feedback at the end
- Grade range: **A-** to **A**

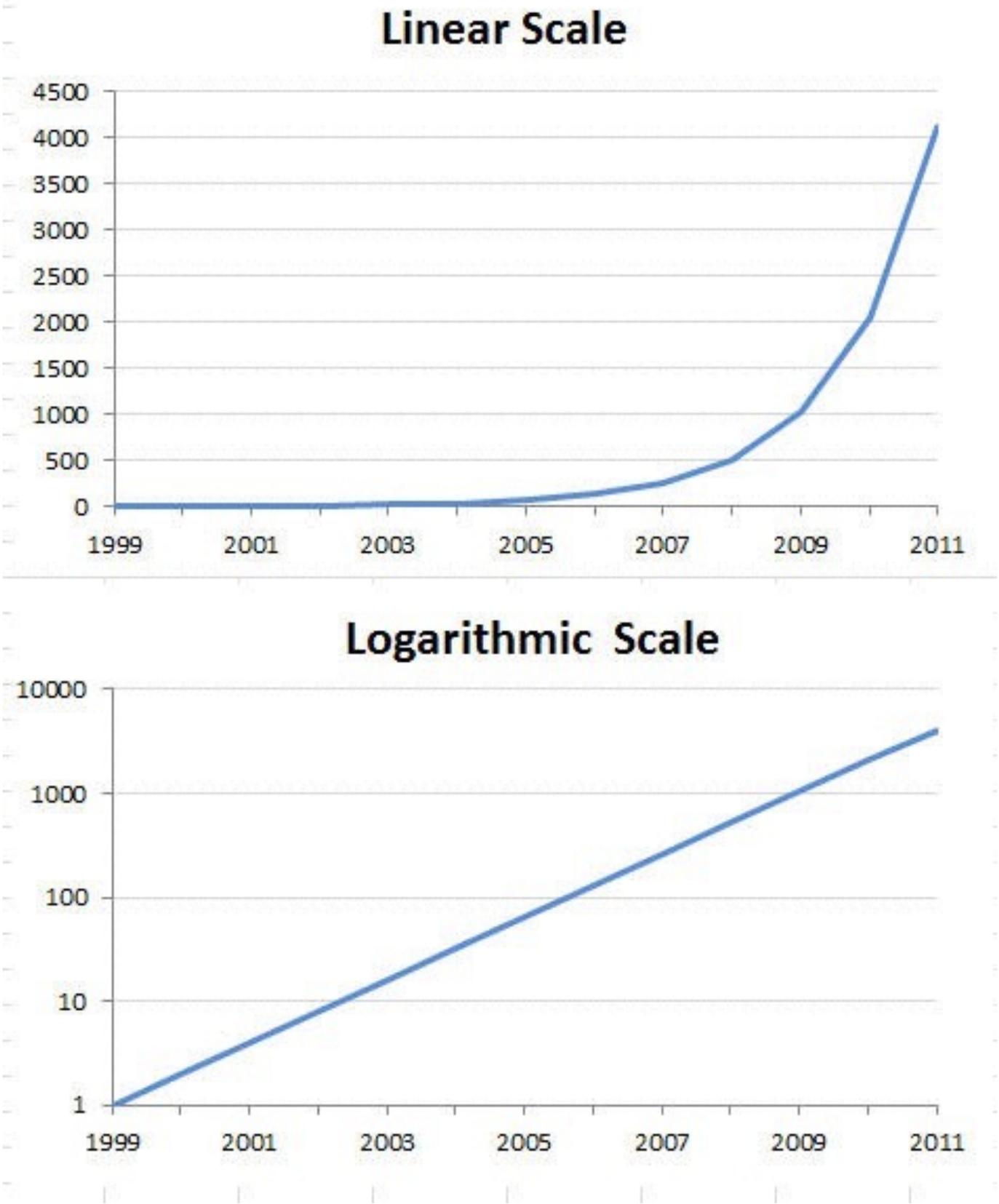
Comments on Newsletter

- Pick recent articles (1 month or less)
- Expand acronyms, but ok to use acronyms
- Boldface the keywords in each article (cloud, GPU, algorithm, TOP500, etc.)
- **Title**
  Author, *publication*, date
- Make sure you figure out the message of the article. What should one remember from having read it.
- Use present tense
- Grade range: **A-** to **A**

Comments on Berkeley Paper

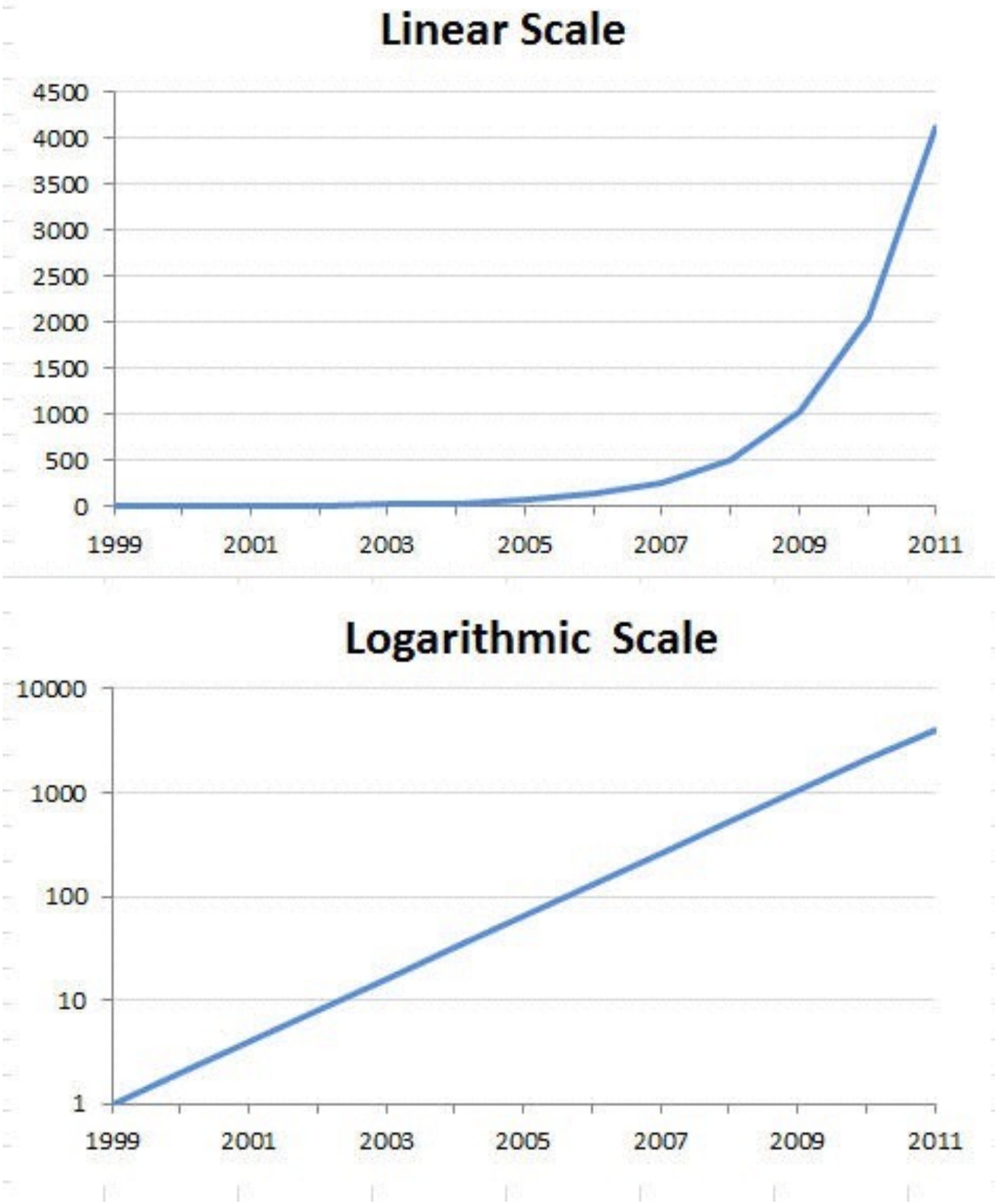
Moore's Law

Moore's Law



- Gordon Moore (Fairchild, Intel)
- 1965, doubling **every year** of components/ IC
- 1975, revised to **doubling every 2 years**

Moore's Law



- Applies to:
  - # transistors
  - speed of processor
  - size of memory
  - # pixels in cameras
  - uProcessor prices

Plan for $10 Billion Chip Plant Shows China's Growing Pull

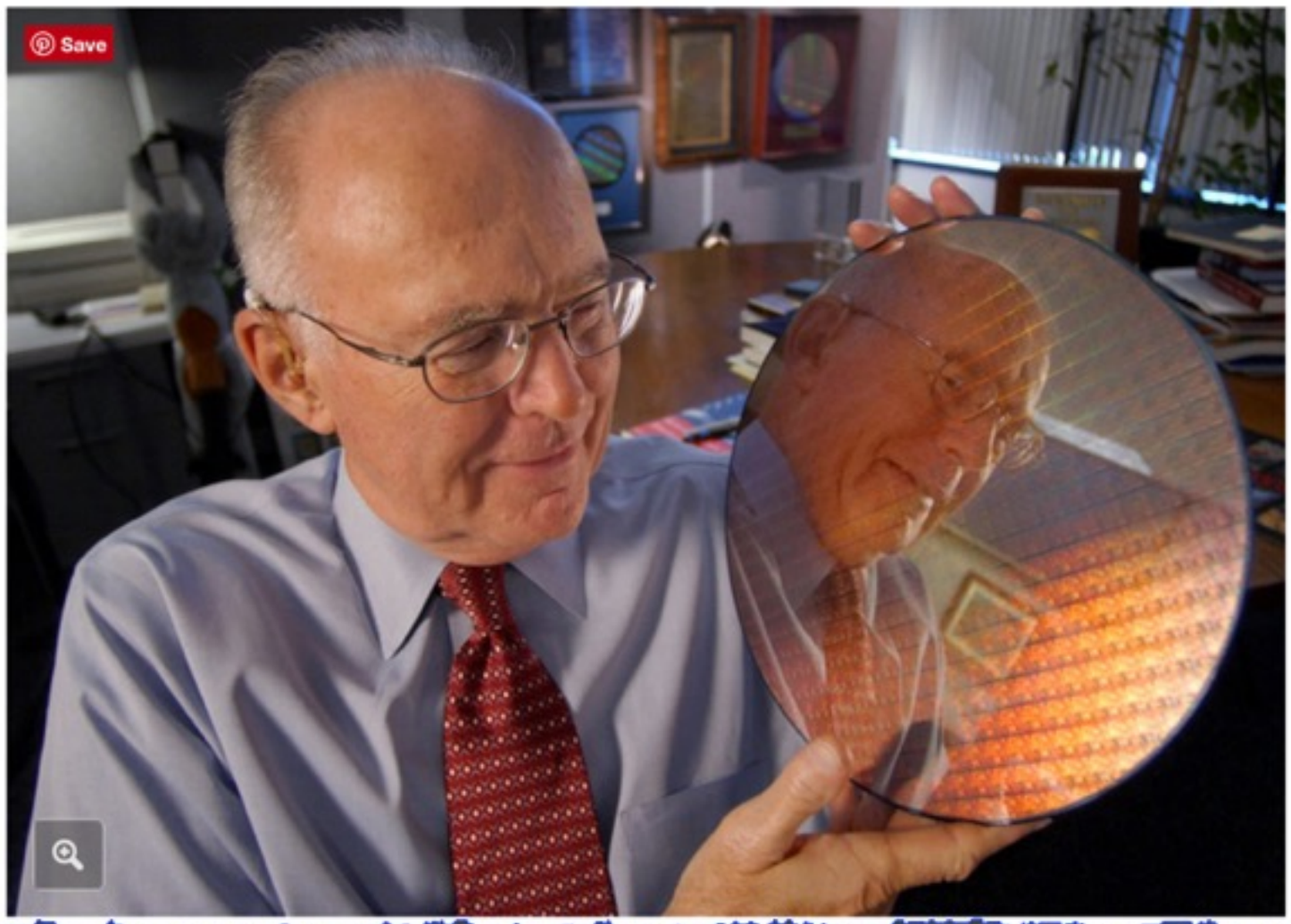
Twitter Struggles to Capitalize on Influence and Posts Lackluster Earnings

TECH FIX
A Low-Tech Guide to Becoming More Politically Active

Apple Tiptoes Into Producing Original Video but Plans to Pick Up Pace

Moore's Law Running Out of Room, Tech Looks for a Successor

By JOHN MARKOFF MAY 4, 2016



RELATED COVERAGE

Smaller, Faster, Cheaper, Over: The Future of Computer Chips SEPT. 26, 2015

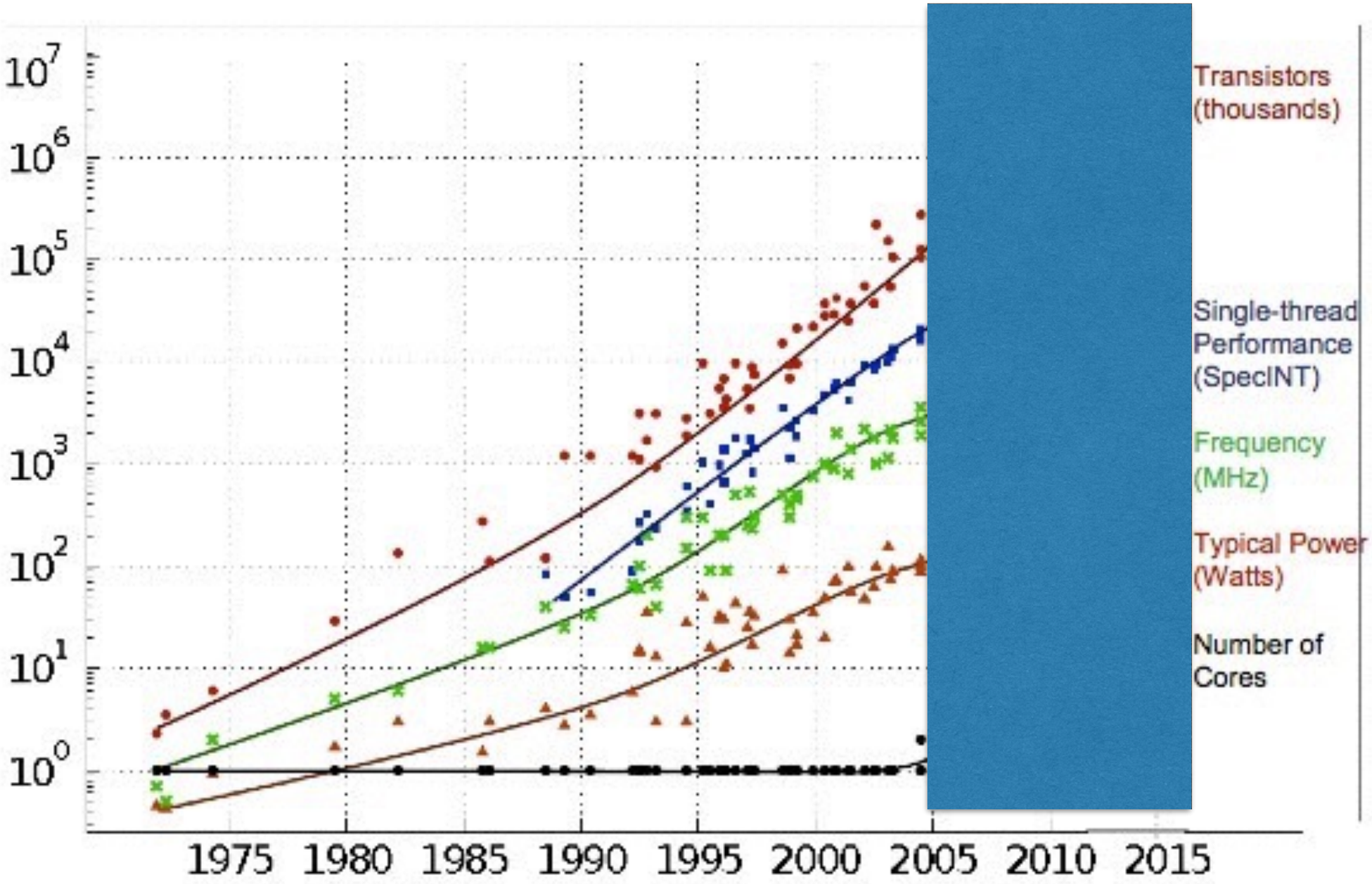
Intel to Cut 12,000 Jobs as PC Demand Plummets APRIL 19, 2016

Intel's Earnings Fall in Fourth Quarter, but Beat Expectations JAN. 14, 2016

RECENT COMMENTS

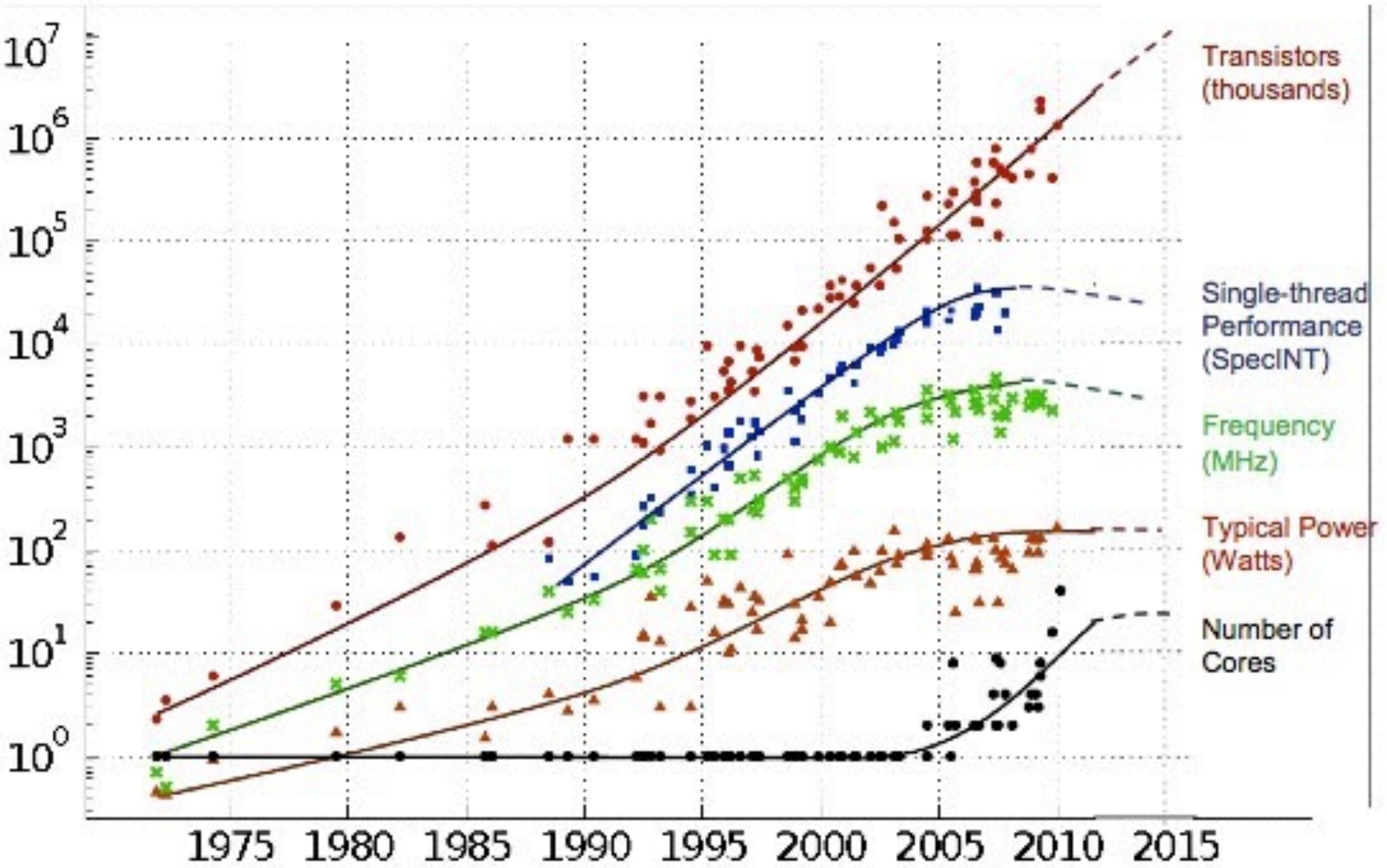
Ashley June 7, 2016
To be honest, if we want speed increases, we should forget sta looking at software. Over the years as processors have become faster,...

Moore's Law



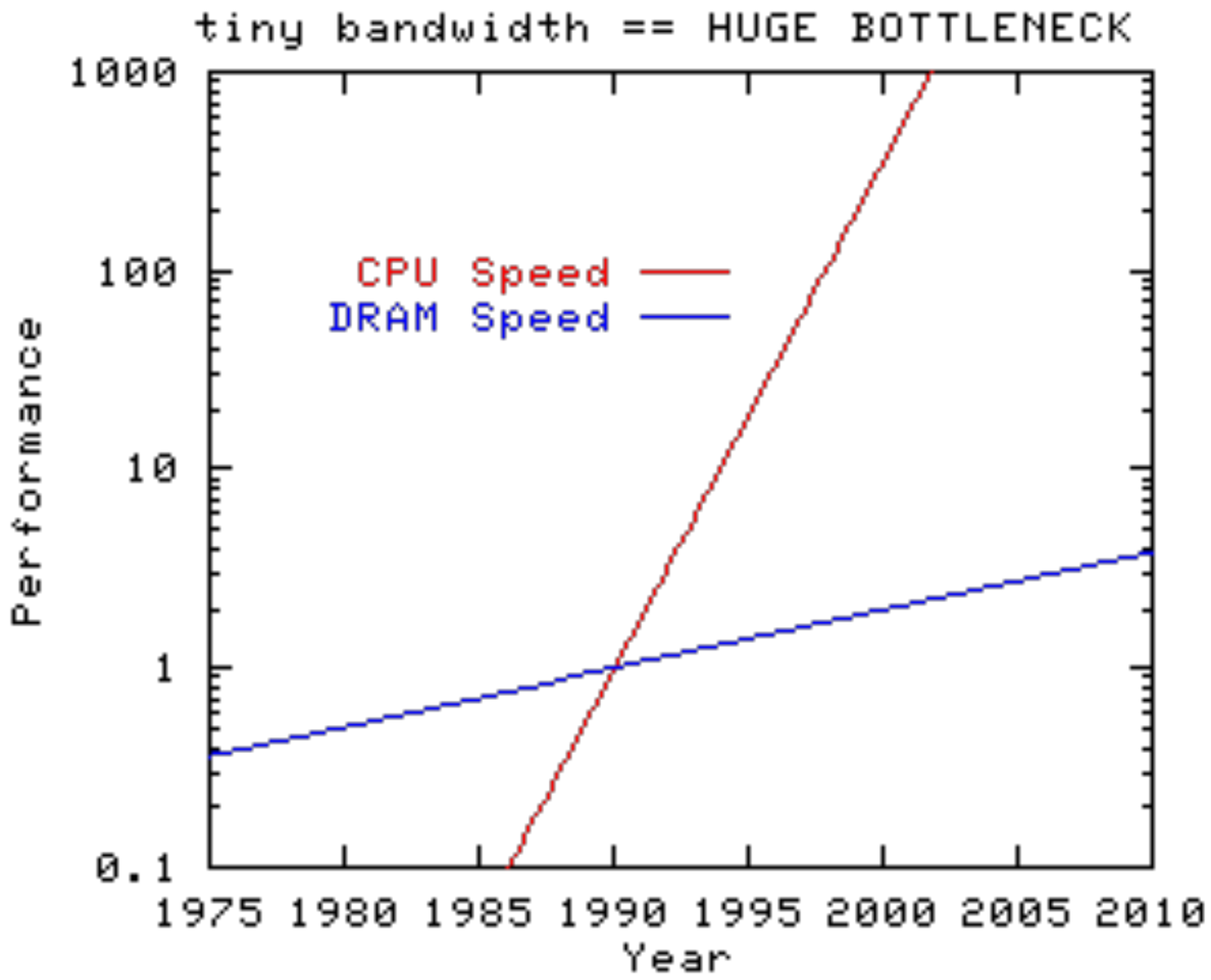
Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

Moore's Law

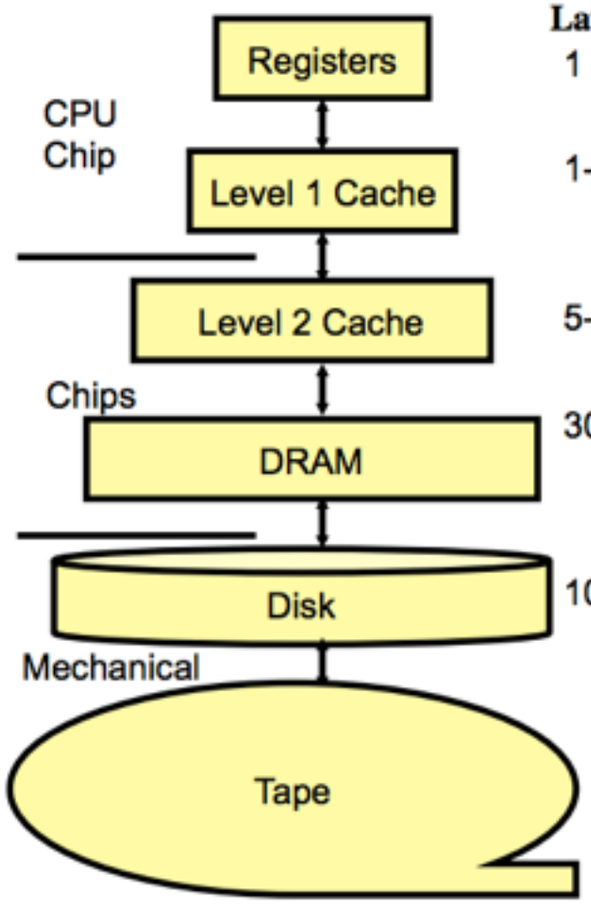


Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

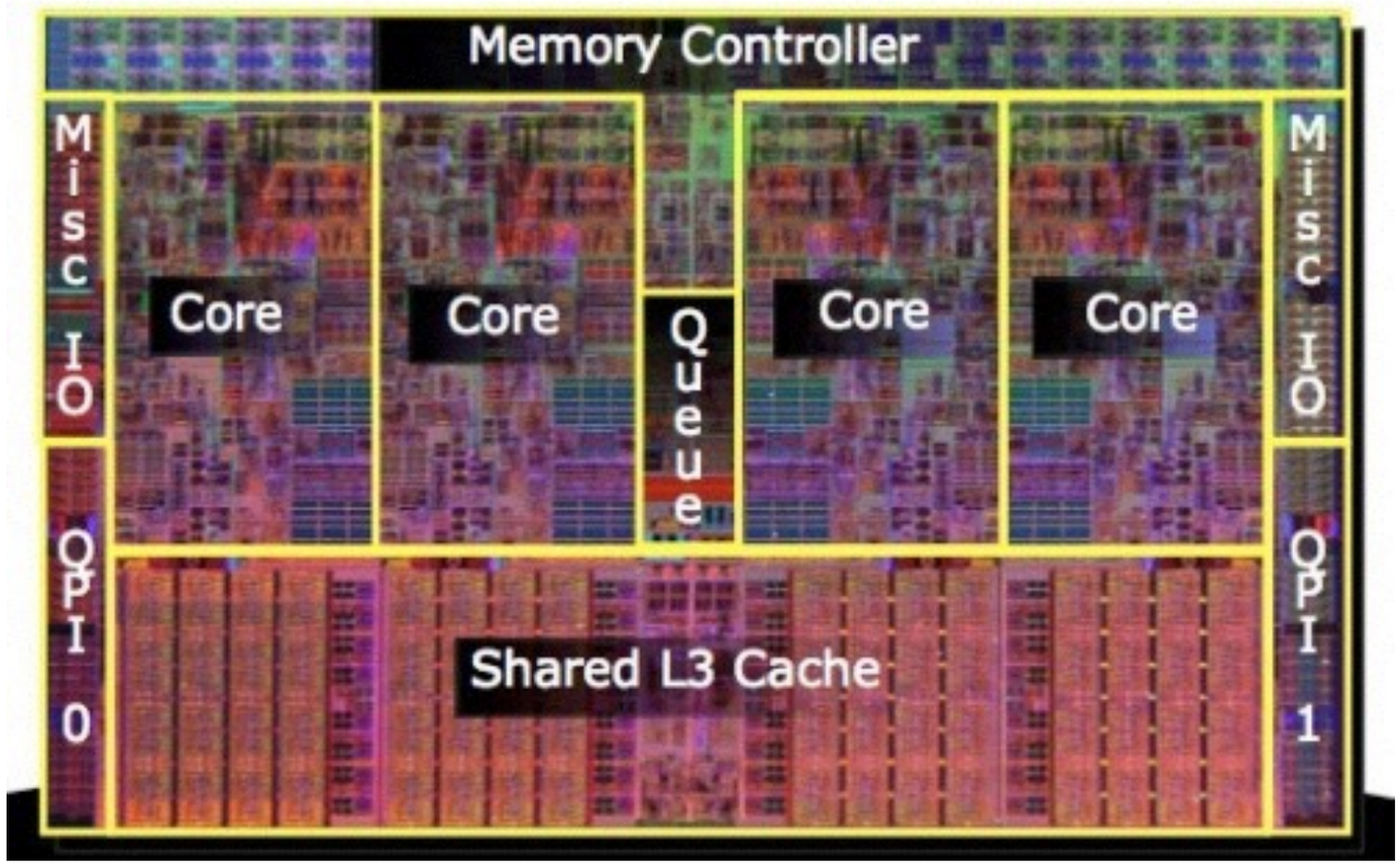
Processor/Memory Gap



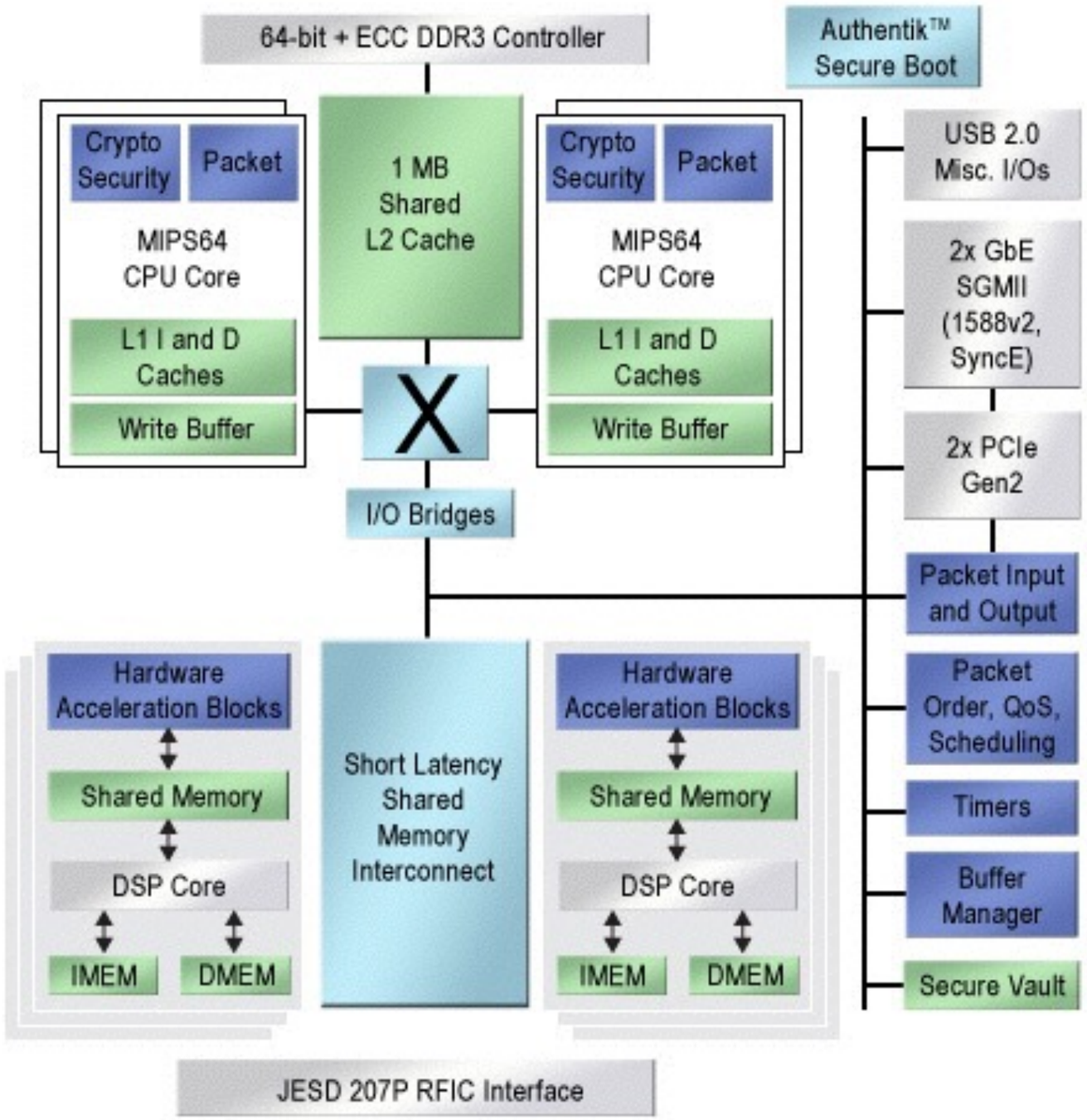
The Memory Hierarchy



	Latency	Bandwidth	
Registers	1 cyc	3-10 words/cycle < 1KB	compiler managed
Level 1 Cache	1-3cy	1-2 words/cycle 32KB -1MB	hardware managed
Level 2 Cache	5-10cy	1 word/cycle 1MB - 4MB	hardware managed
DRAM	30-100cy	0.5 words/cycle 64MB - 4GB	OS managed
Disk	10^6-10^7cy	0.01 words/cycle 4GB+	OS managed



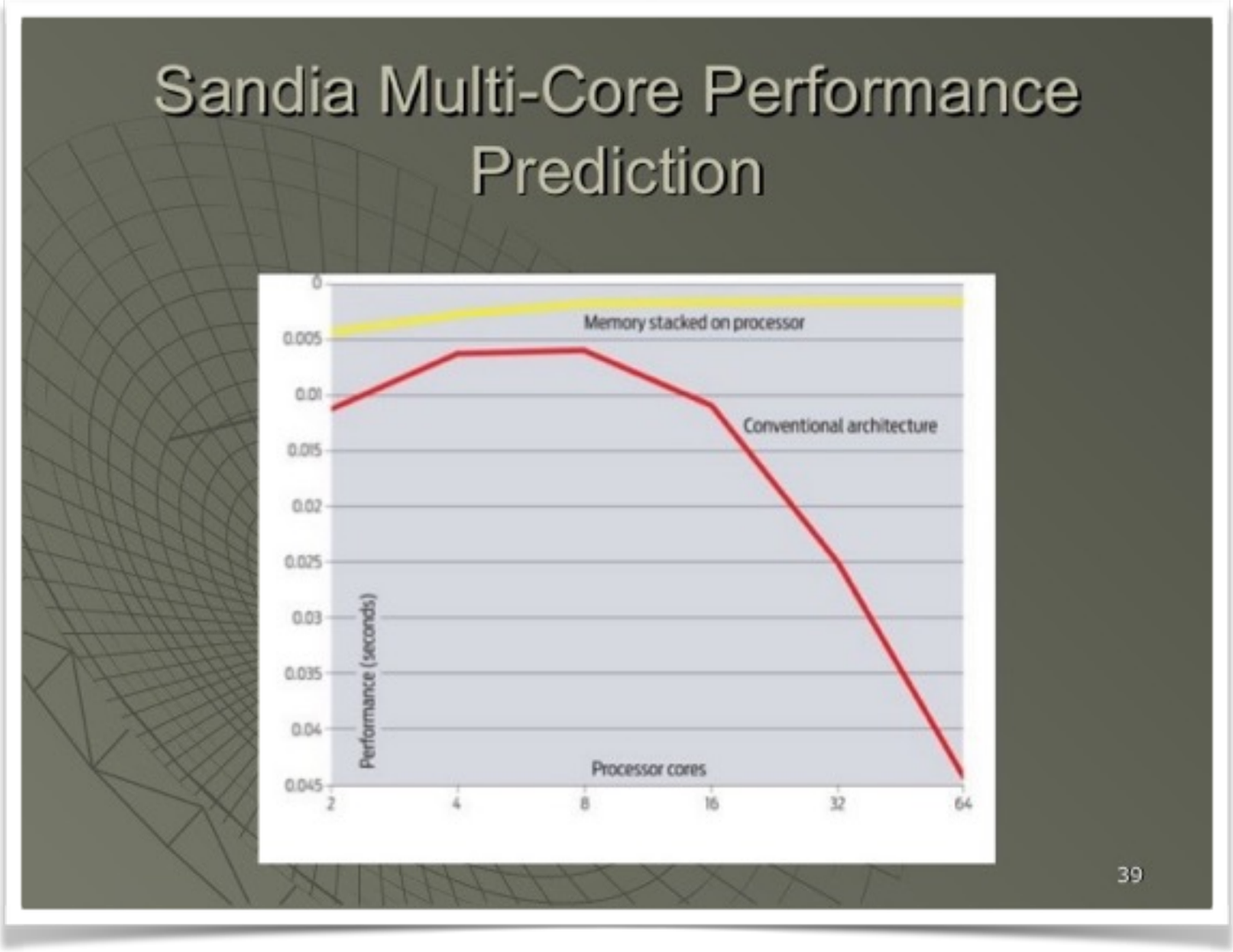
Many-Core vs Multi-Core



https://www.altera.com/technology/system-design/articles/2012/multicore-many-core.html

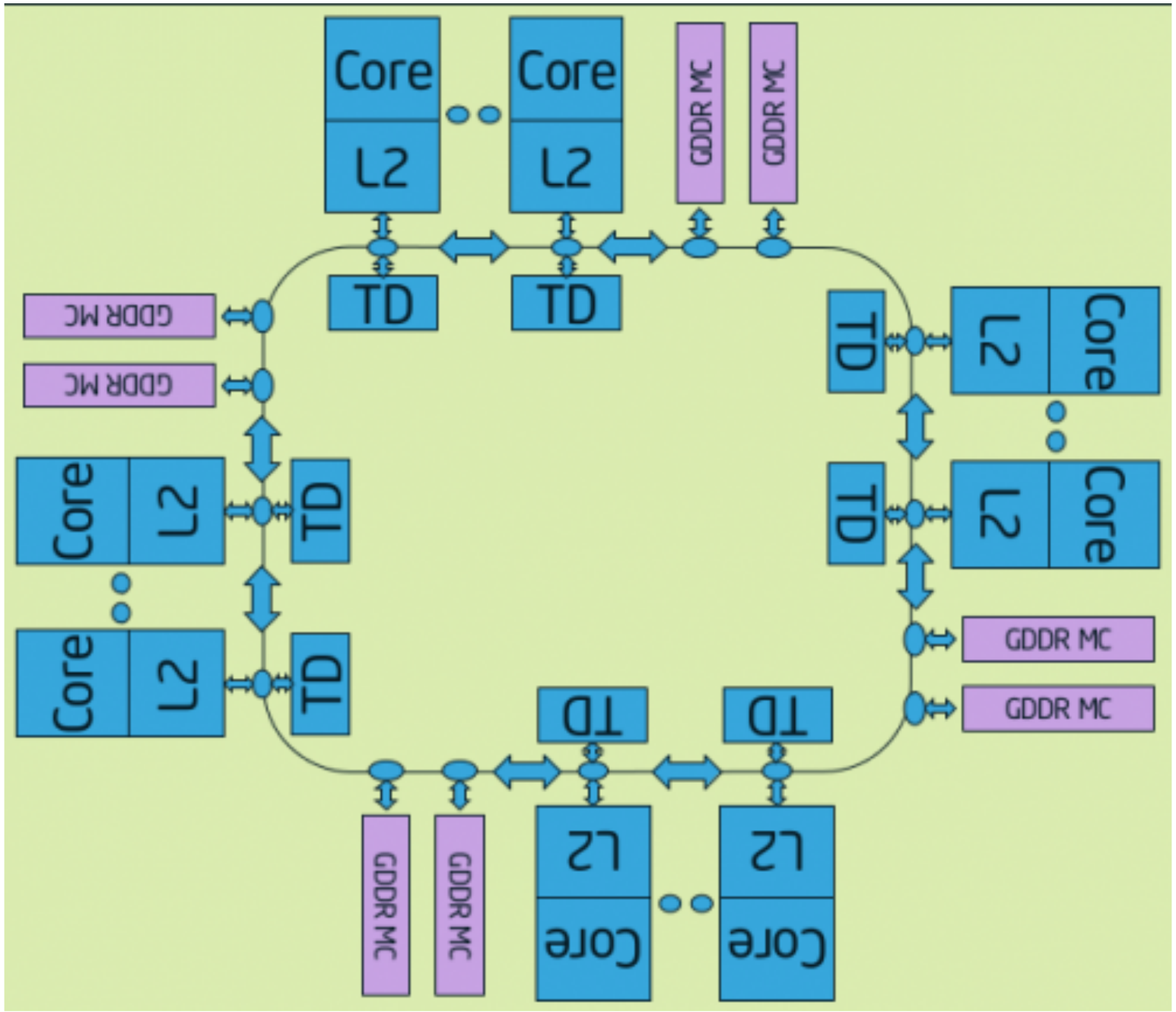
Multicore Performance

Benchmark Analysis of Multi-Core Processor Memory Contention, Simon & McGalliard, SCMG, 2009



https://image.slidesharecdn.com/8aeda7ce-5324-48d6-b6d8-26bd03fed953-150707001147-lva1-app6892/95/benchmark-analysis-of-multicore-processor-memory-contention-april-2009-39-638.jpg?cb=1436228009

On-Chip Networking

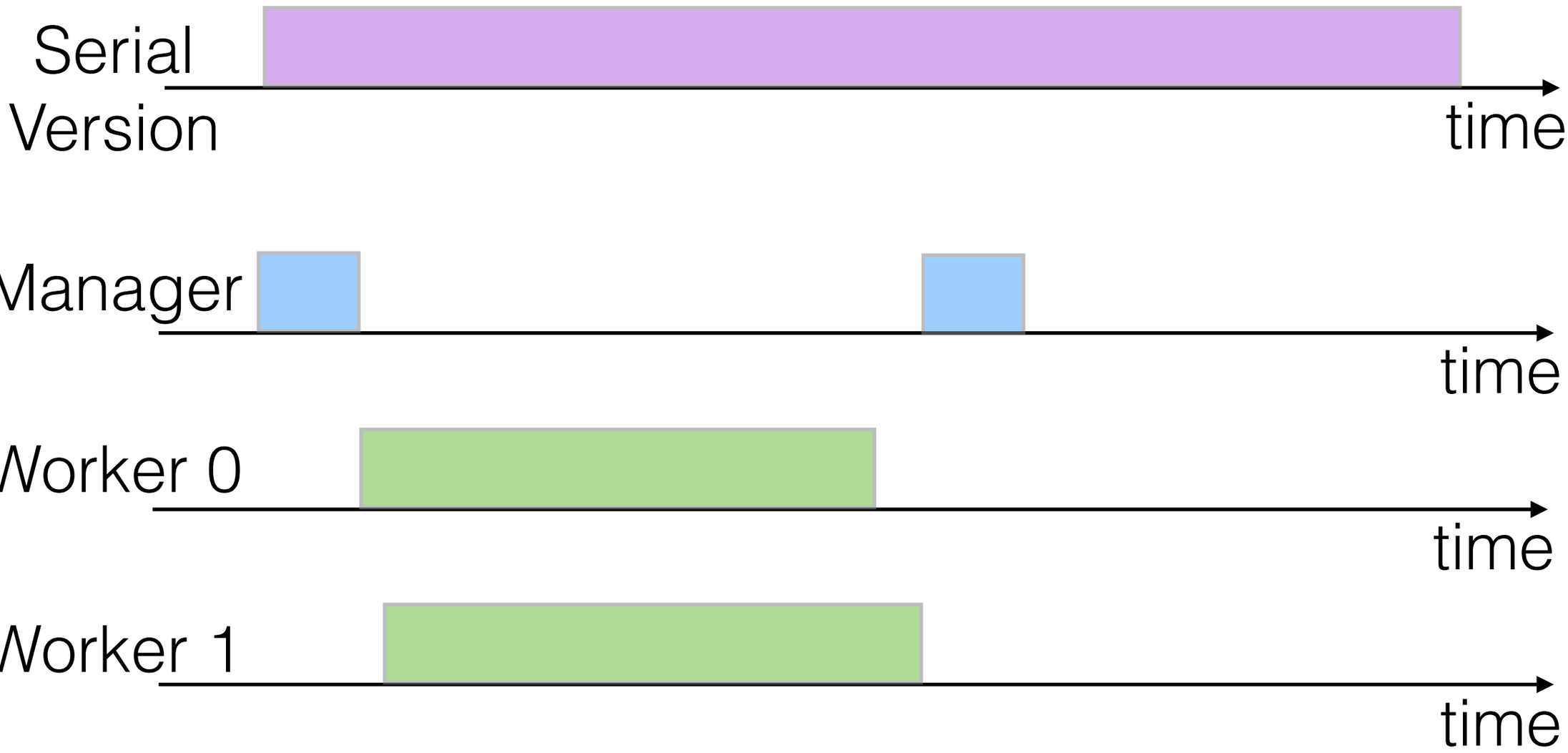


Think
Monte Carlo
Simulation

Amdahl's Law

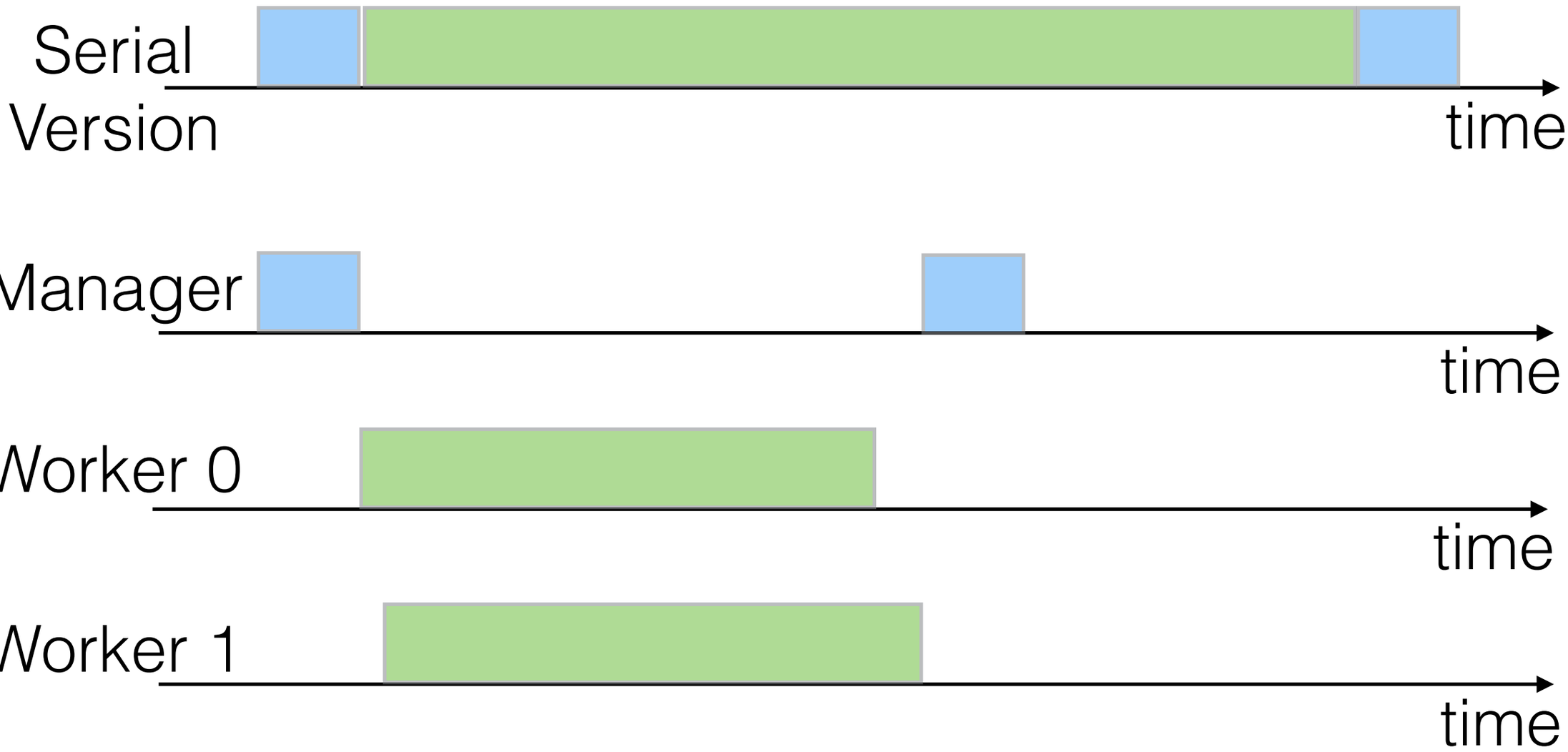
Think
Monte Carlo
Simulation

Amdahl's Law

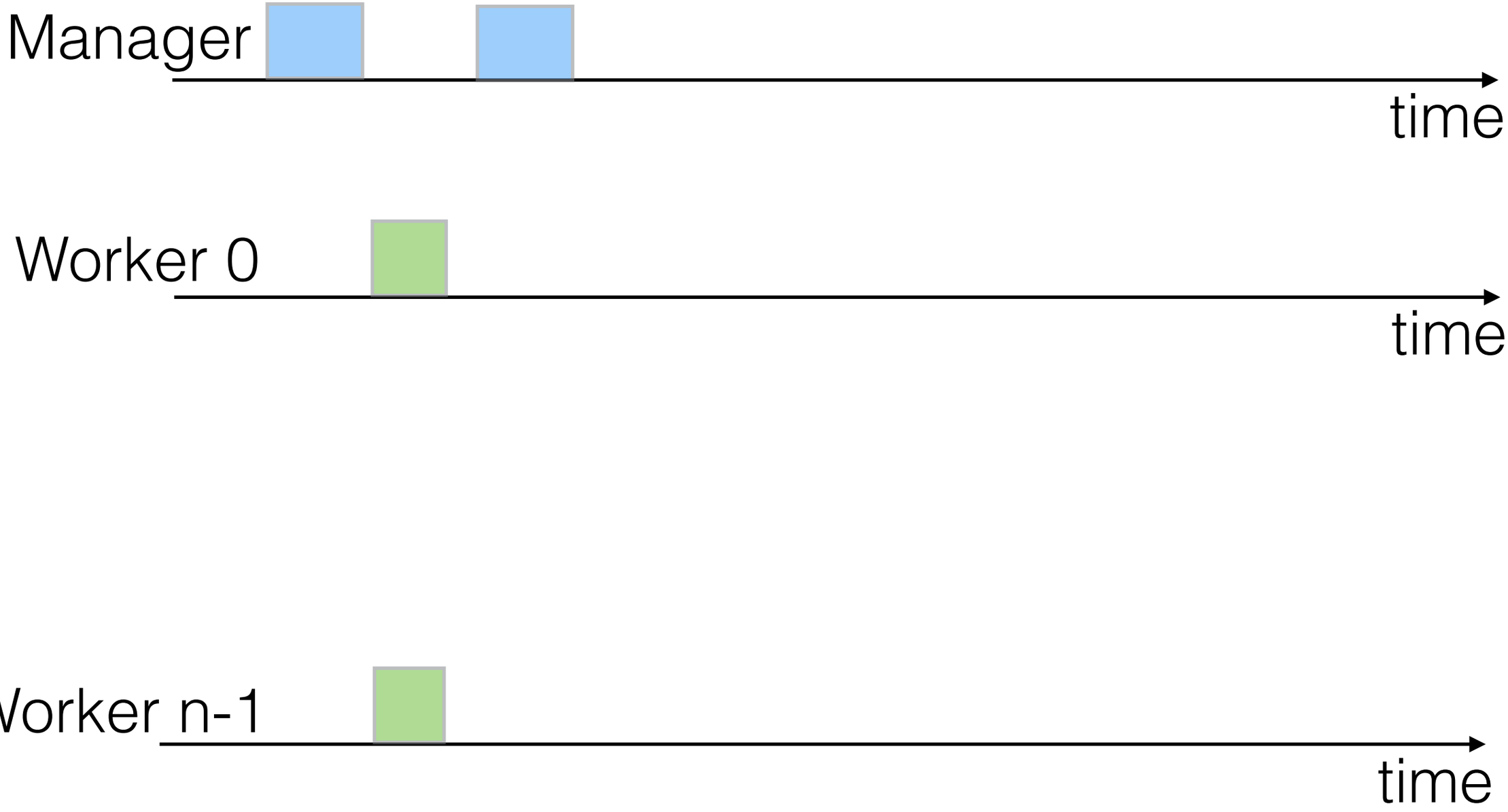


Think
Monte Carlo
Simulation

Amdahl's Law

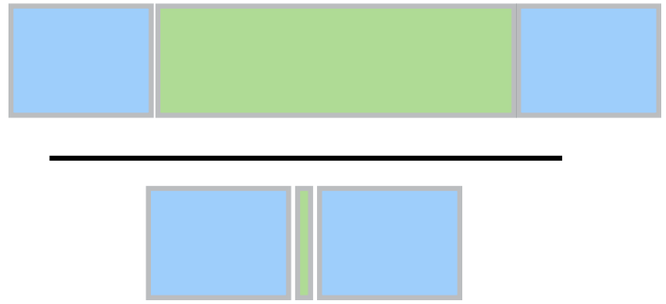


Amdahl's Law

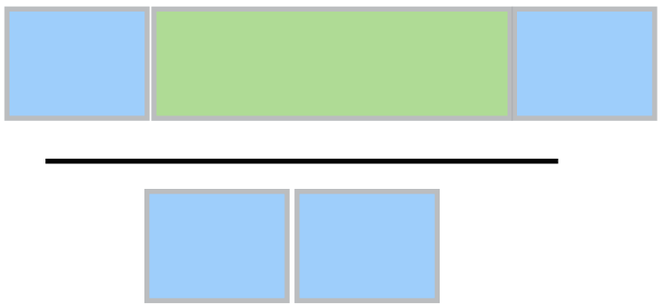


Amdahl's Law

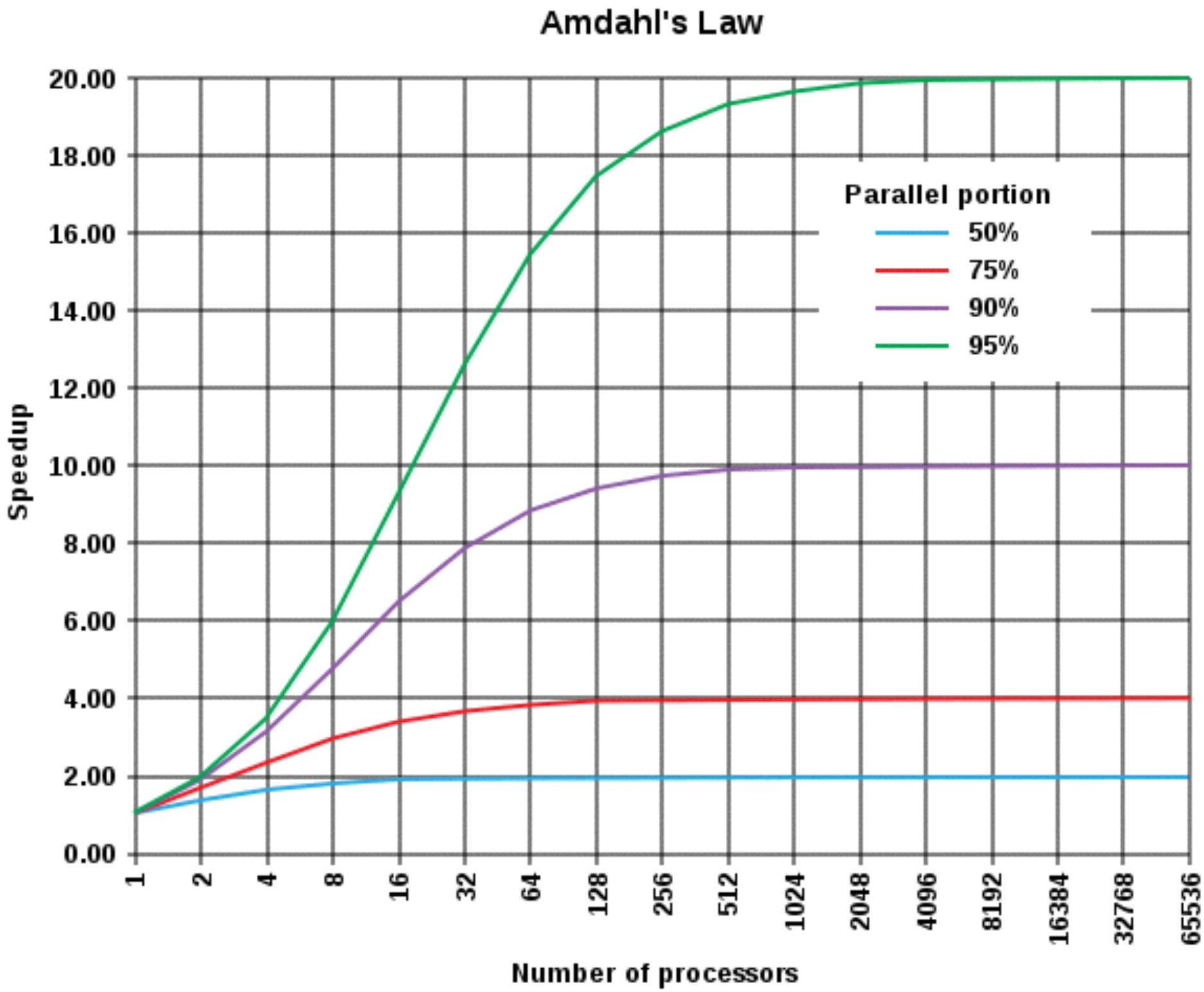
$$\text{Speedup}_N = \frac{T(1)}{T(N)} =$$



$$\text{Speedup}_{oo} = \frac{T(1)}{T(oo)} =$$



If 5% of code is serial, then max speedup is $\frac{5\% + 95\%}{5\%}$



https://en.wikipedia.org/wiki/Amdahl's_law

Amdahl's Law

- Too pessimistic
- As problem size gets larger, portion of parallel code increases
- As more processors are added, more of the data can fit in memory, cache ==> gain speed in accessing data

nanometers

http://en.wikipedia.org/wiki/22_nanometer

Semiconductor manufacturing processes

10 μm — 1971
3 μm — 1975
1.5 μm — 1982
1 μm — 1985
800 nm — 1989
600 nm — 1994
350 nm — 1995
250 nm — 1997
180 nm — 1999
130 nm — 2002
90 nm — 2004
65 nm — 2006
45 nm — 2008
32 nm — 2010
22 nm — 2012
14 nm — 2014
10 nm — est. 2017
7 nm — est. 2020
5 nm — est. 2022

Half-nodes

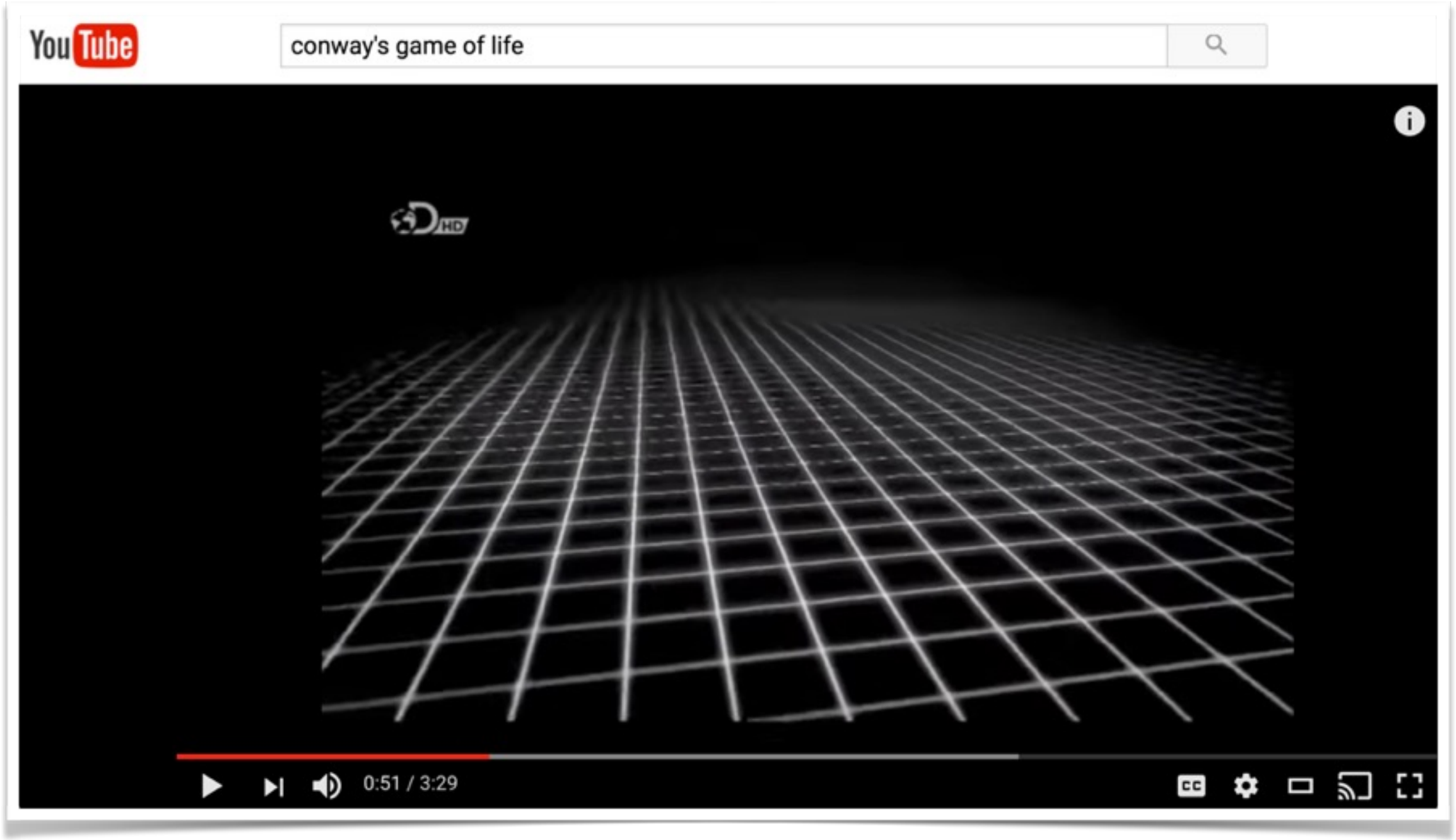
V • T

https://www.youtube.com/watch?v=qm67wbB5GmI

(1m10 - 8m20)

Programming Lab

Making the Game of Life Parallel



https://www.youtube.com/watch?v=CgOcEZinQ2I

Serial Version

- Study it

login to your **352b** account

```
getCopy GameOfLife.java
javac GameOfLife.java
java GameOfLife
```

- Run it on your laptop

- Use both dish and dish2 as the array of live cells, and see how they evolve

2-Thread Version

- As a group, discuss the different tissues associated with parallelizing the Game of Life and running it with two threads.
- List all the issues that must be addressed on the whiteboard
- How will you verify the correctness of the parallel version?
- Play-out the execution of the 2-thread program: two people or two groups play the roles of the two threads.

Could be Usefull…

- **What is a BlockingQueue?**
  BlockingQueue is a queue which is **thread safe** to insert or retrieve elements from it. Also, it provides a mechanism which blocks requests for inserting new elements when the queue is full or requests for removing elements when the queue is empty, with the additional option to stop waiting when a specific timeout passes. This functionality makes *BlockingQueue* a nice way of implementing the Producer-Consumer pattern, as the producing thread can insert elements until the upper limit of *BlockingQueue* while the consuming thread can retrieve elements until the lower limit is reached and of course with the support of the aforementioned blocking functionality.

https://examples.javacodegeeks.com/core-java/util/concurrent/java-blockingqueue-example/

Thread safe: Implementation is guaranteed to be free of race conditions when accessed by multiple threads simultaneously.

How to use a BlockingQueue

```
package com.javacodegeeks.java.util.concurrent.blockingqueue;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

public class BlockingQueueExample {

     public static void main(String[] args) throws Exception {
            BlockingQueue<Integer> bq = new ArrayBlockingQueue<Integer>(1000);
            Producer producer = new Producer(bq);
            Consumer consumer = new Consumer(bq);
            new Thread(producer).start();
            new Thread(consumer).start();
            Thread.sleep(4000);
     }
}
```

Implement the 2-Thread Game of Life in Java

Measuring Performance

- Pick setup that will not be slowed down by OS or non necessary IO operations
- Pick **best** serial algorithm available
- **Tune** the parallel version
- Keep the conditions **constant** (same grid size)
- Measure the **average** execution time of several runs for each case
- Use shell **scripts**! (See next slide)
- Pick several possible **measures of performance**
    - speedup
    - throughput
    - ?

Using Shell Scripts

http://www.science.smith.edu/dftwiki/index.php/CSC352:_Using_Bash,_an_example