

Future of Computing: Moore's Law & Its Implications + High-Performance Computing

15-213: Introduction to Computer Systems

27th Lecture, Dec. 1, 2016

Instructors:

Randy Bryant

Moore's Law Origins



April 19, 1965

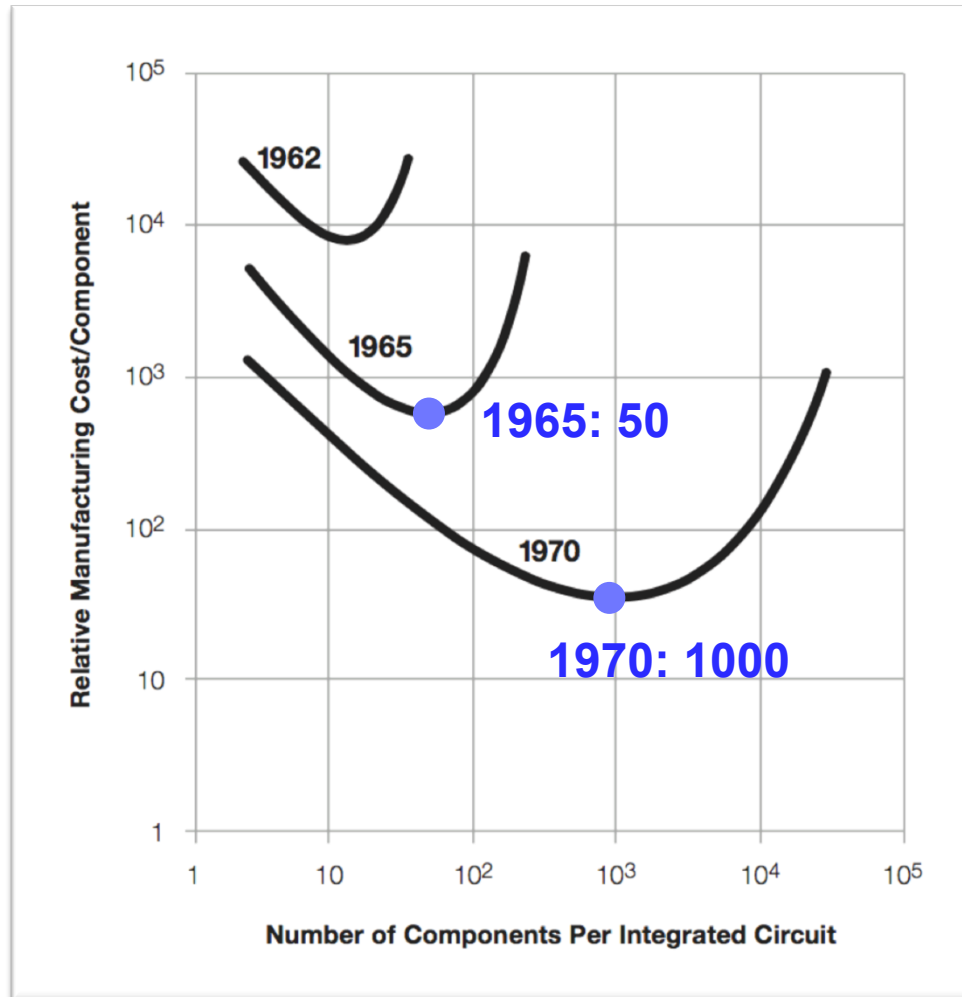
Cramming more components onto integrated circuits

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.

Moore's Law Origins



Moore's Thesis

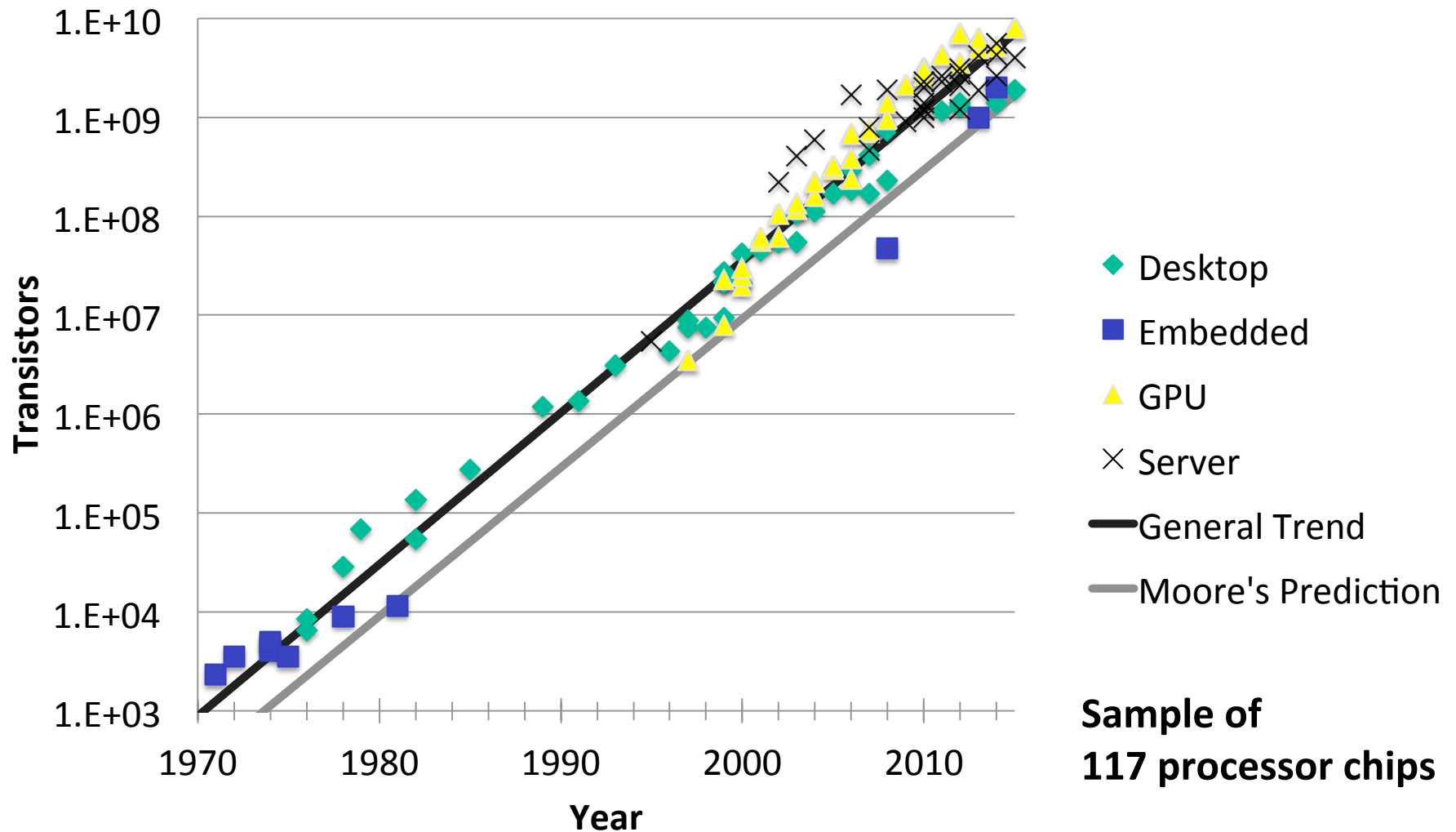
- Minimize price per device
- Optimum number of devices / chip increasing 2x / year

Later

- 2x / 2 years
- "Moore's Prediction"

Moore's Law: 50 Years

Transistor Count by Year



What Moore's Law Has Meant



■ 1976 Cray 1

- 250 M Ops/second
- ~170,000 chips
- 0.5B transistors
- 5,000 kg, 115 KW
- \$9M
- 80 manufactured



■ 2014 iPhone 6

- > 4 B Ops/second
- ~10 chips
- > 3B transistors
- 120 g, < 5 W
- \$649
- 10 million sold in first 3 days

What Moore's Law Has Meant

■ 1965 Consumer Product



■ 2015 Consumer Product

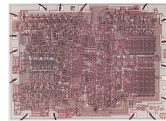


**Apple A8 Processor
2 B transistors**

Visualizing Moore's Law to Date

If transistors were the size of a grain of sand

Intel 4004
1970
2,300 transistors



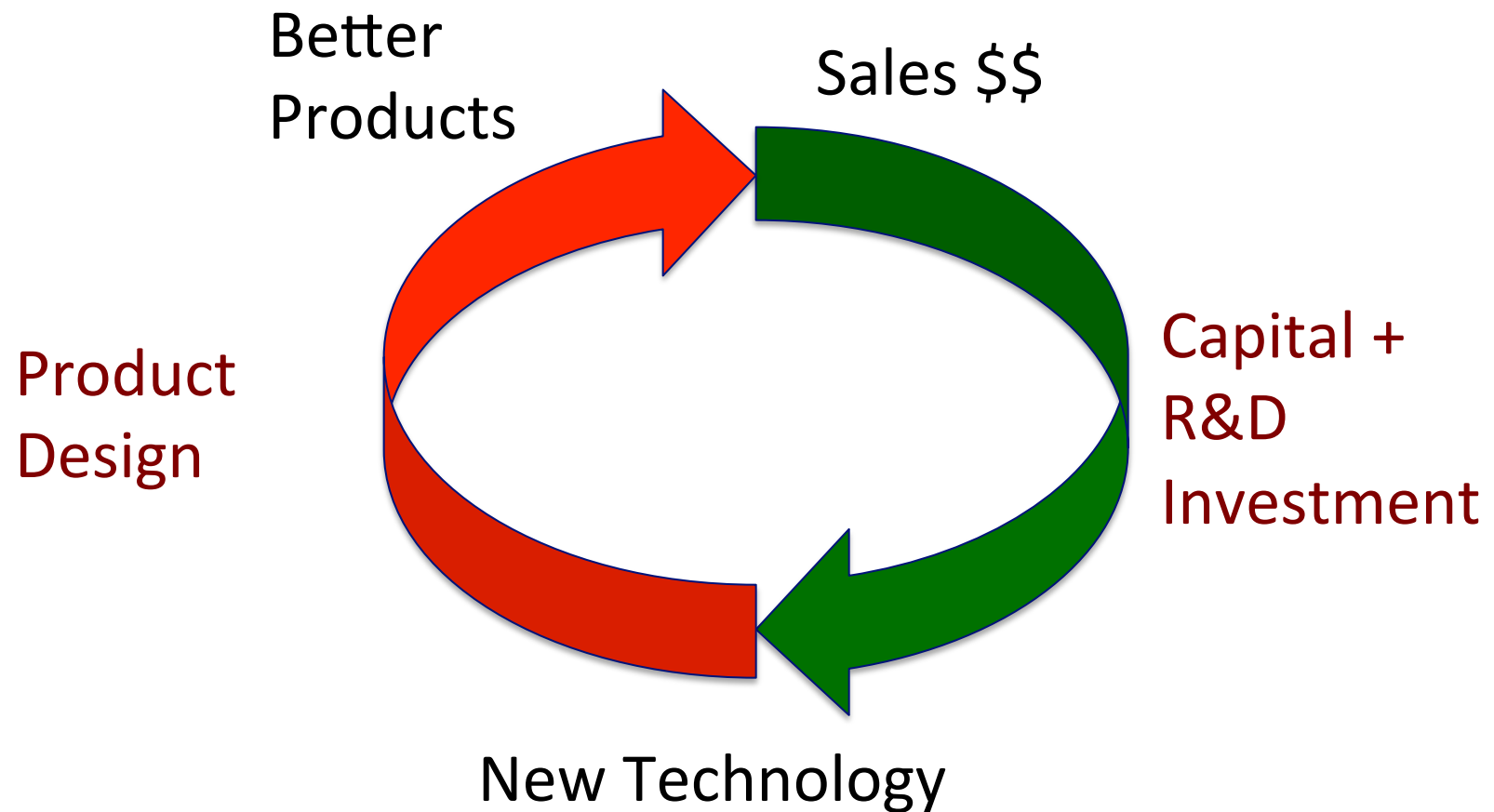
0.1 g

Apple A8
2014
2 B transistors



88 kg

Moore's Law Economics

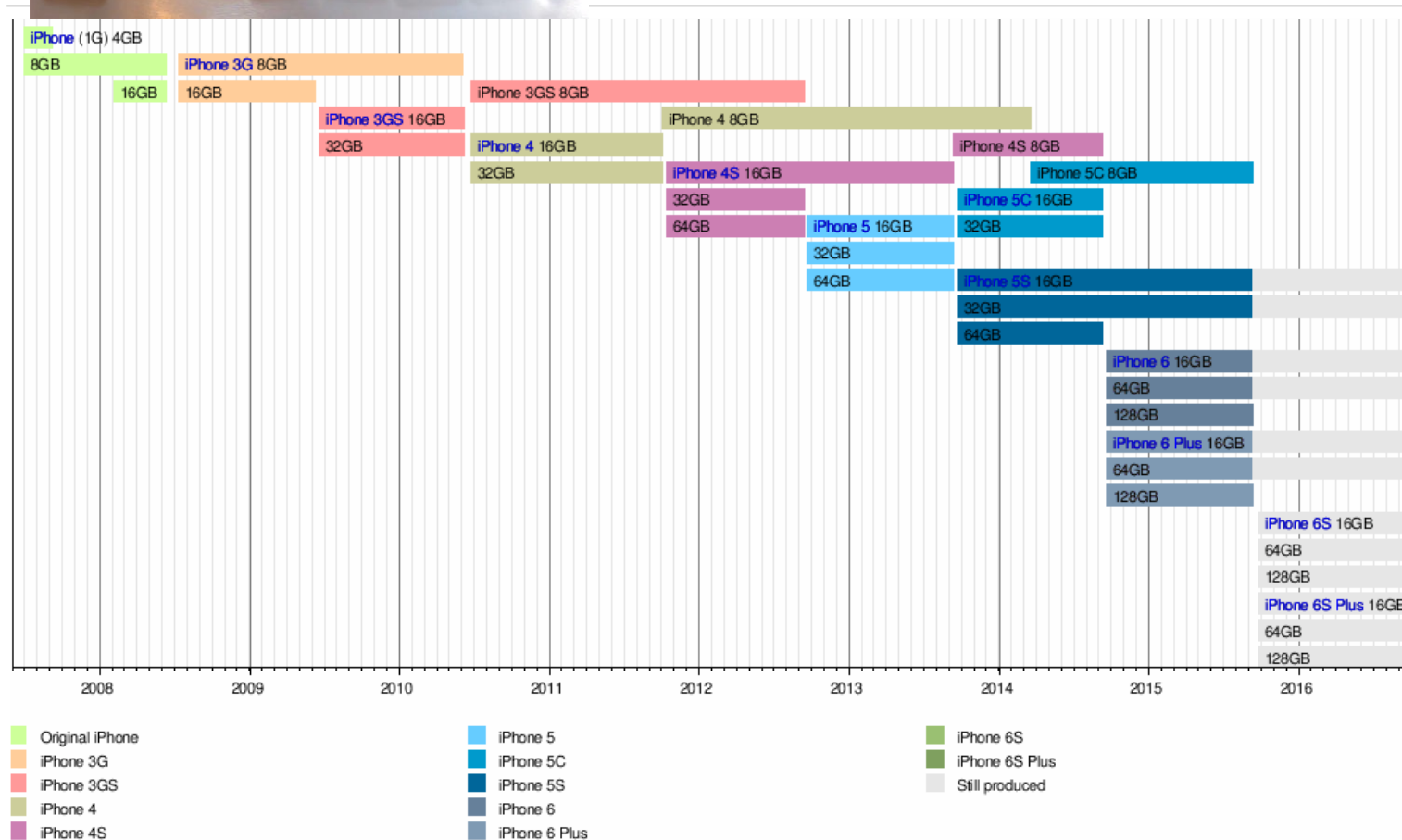


**Consumer products sustain the
\$300B semiconductor industry**

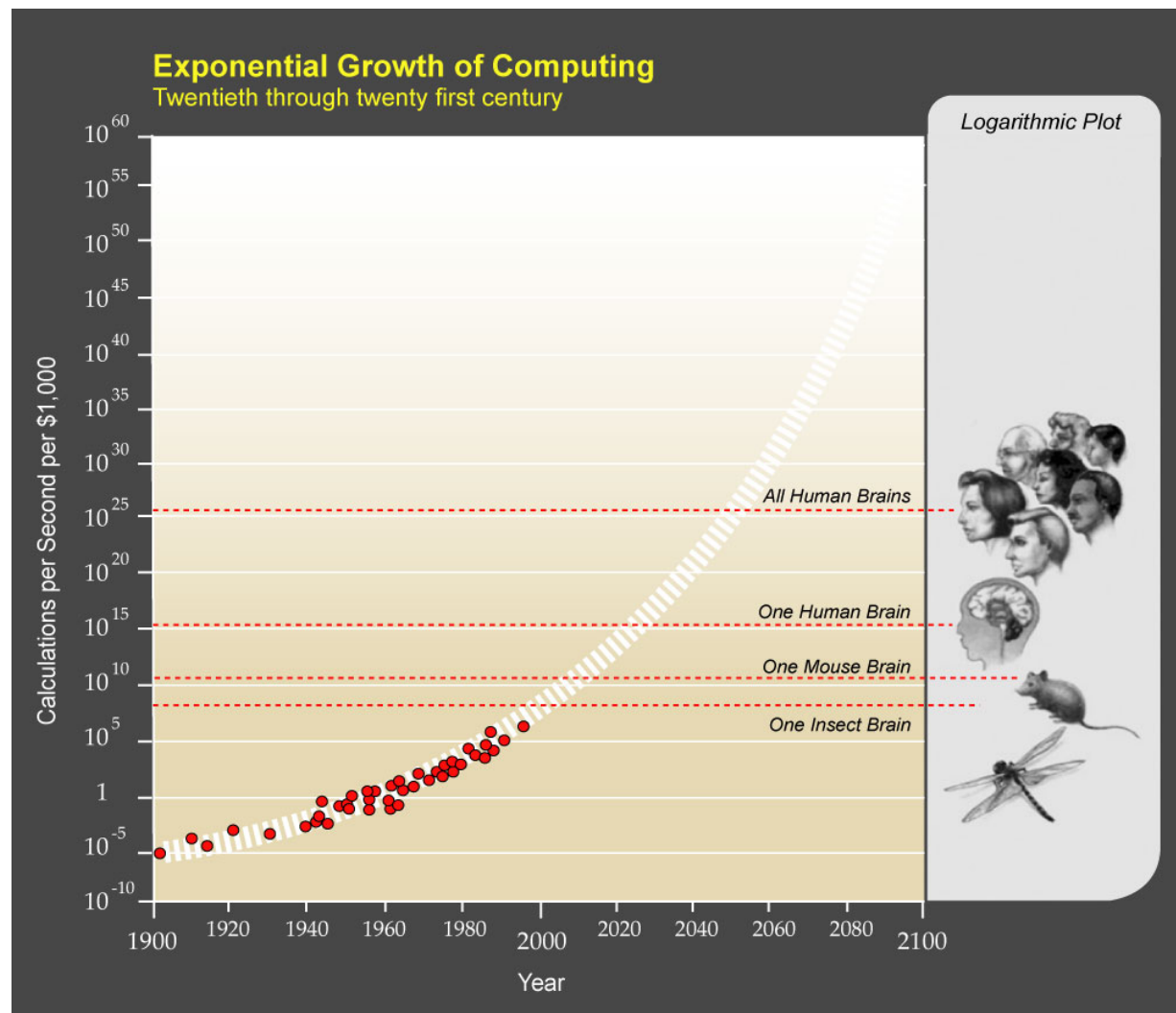
What Moore's Law Has Meant



9 generations of iPhone since 2007



What Moore's Law Could Mean



What Moore's Law Could Mean

■ 2015 Consumer Product



■ 2065 Consumer Product



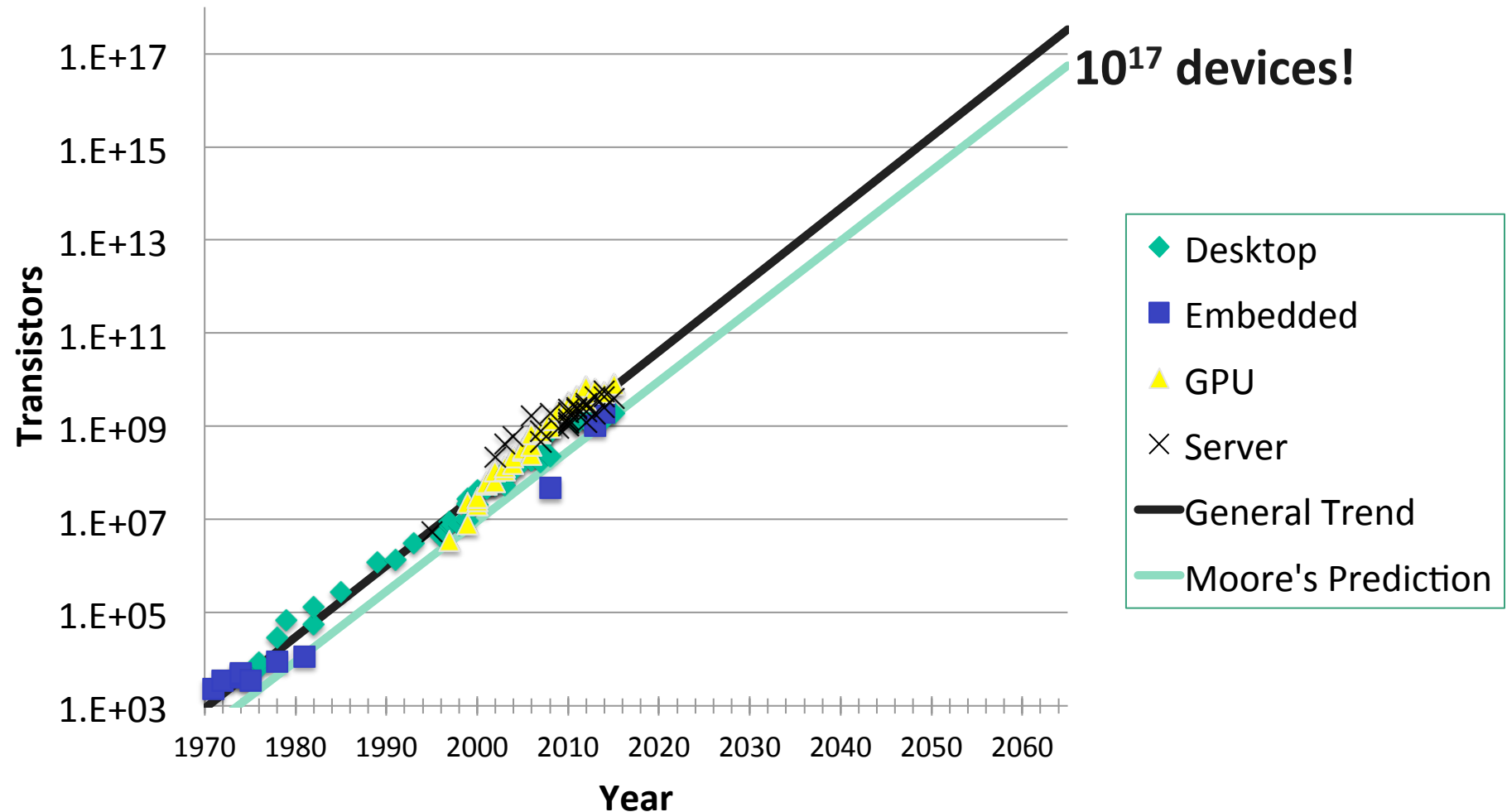
- Portable
- Low power
- Will drive markets & innovation

Requirements for Future Technology

- **Must be suitable for portable, low-power operation**
 - Consumer products
 - Internet of Things components
 - Not cryogenic, not quantum
- **Must be inexpensive to manufacture**
 - Comparable to current semiconductor technology
 - $O(1)$ cost to make chip with $O(N)$ devices
- **Need not be based on transistors**
 - Memristors, carbon nanotubes, DNA transcription, ...
 - Possibly new models of computation
 - But, still want lots of devices in an integrated system

Moore's Law: 100 Years

Device Count by Year



Visualizing 10^{17} Devices

*If devices were the size of
a grain of sand*



0.1 m^3

3.5×10^9 grains



1 million m^3

0.35×10^{17} grains

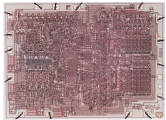
Increasing Transistor Counts

1. **Chips have gotten bigger**
 - 1 area doubling / 10 years
2. **Transistors have gotten smaller**
 - 4 density doublings / 10 years

Will these trends continue?

Chips Have Gotten Bigger

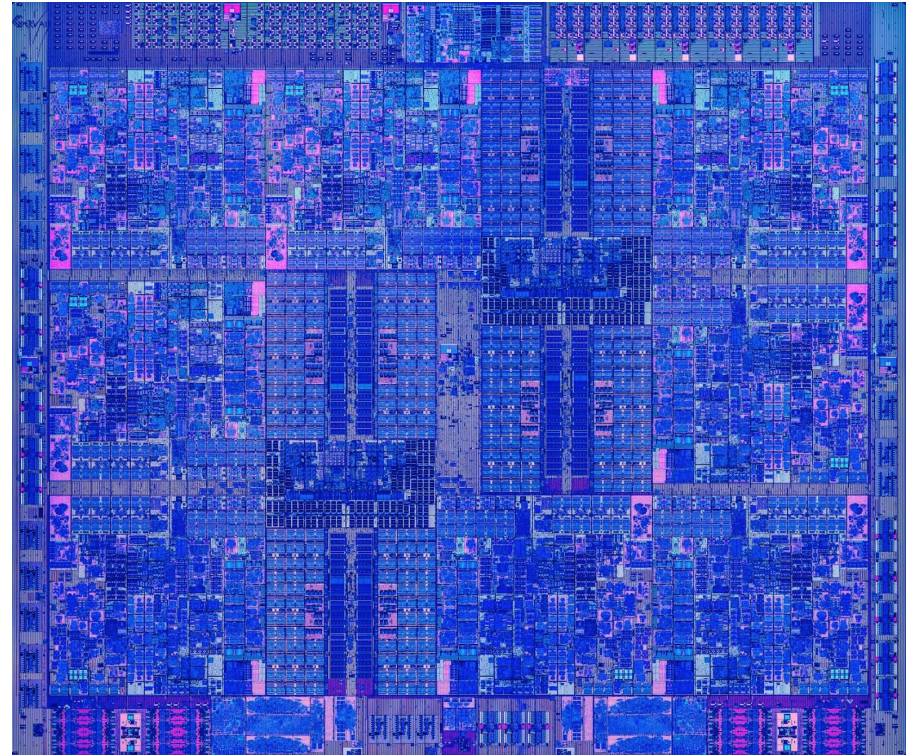
Intel 4004
1970
2,300 transistors
12 mm²



Apple A8
2014
2 B transistors
89 mm²

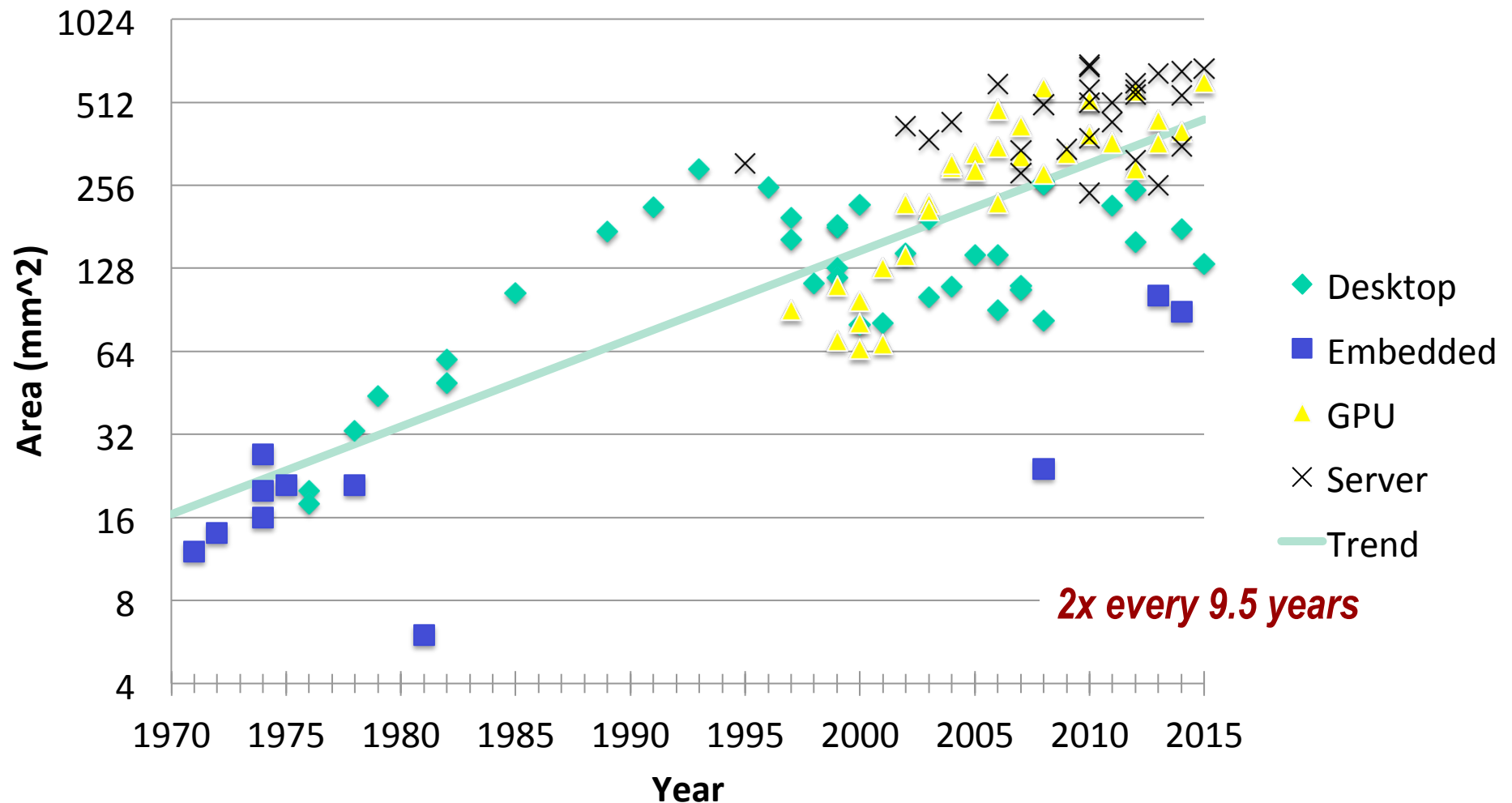


IBM z13
205
4 B transistors
678 mm²



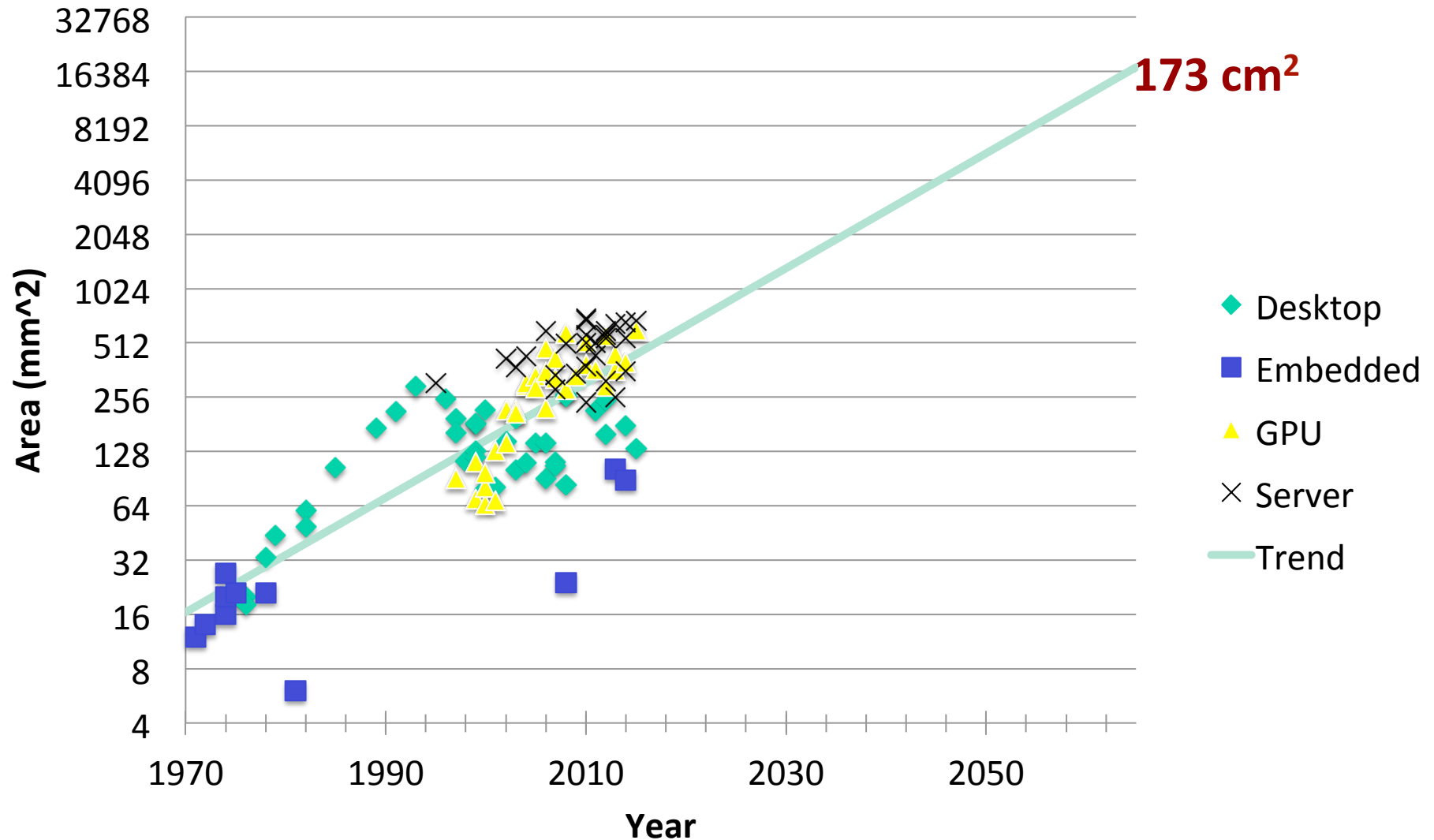
Chip Size Trend

Area by Year



Chip Size Extrapolation

Area by Year



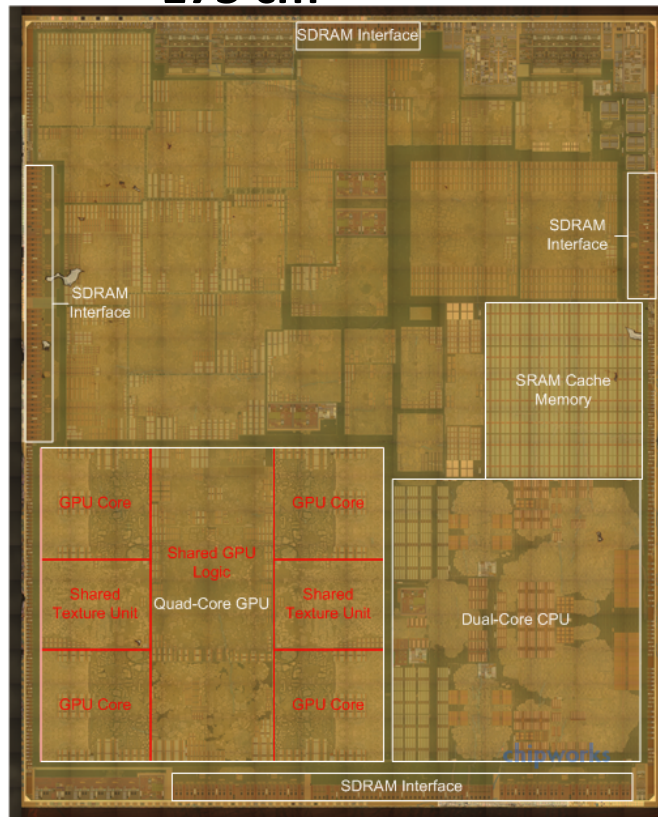
Extrapolation: The iPhone 31s

Apple A59

2065

10^{17} transistors

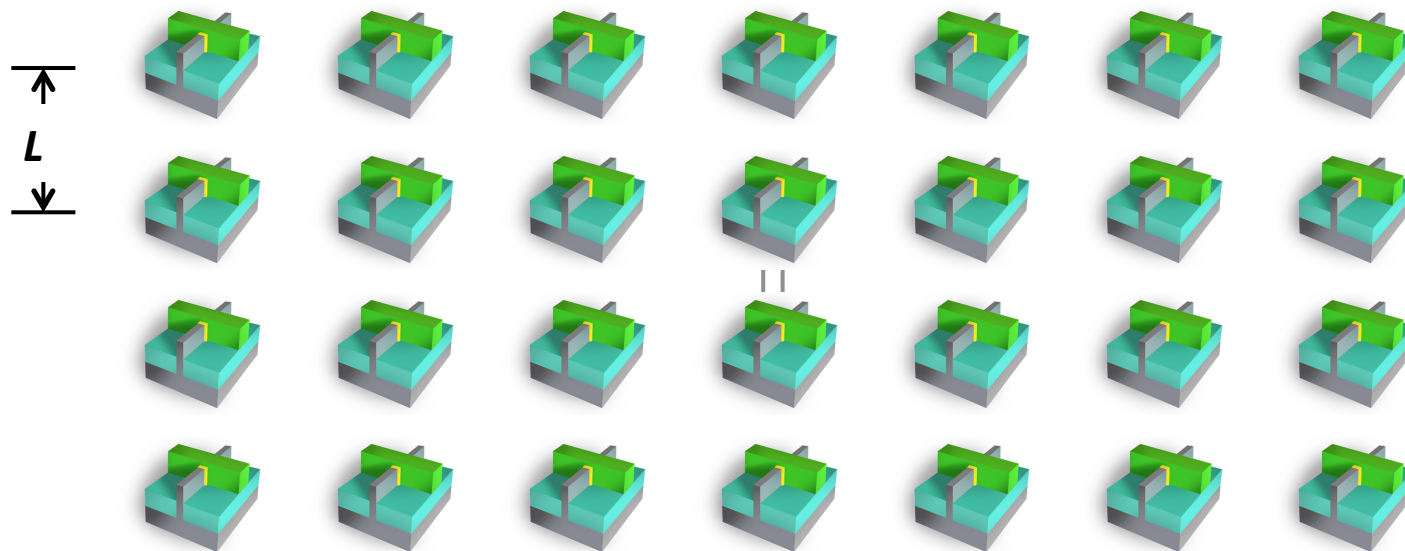
173 cm^2



Transistors Have Gotten Smaller

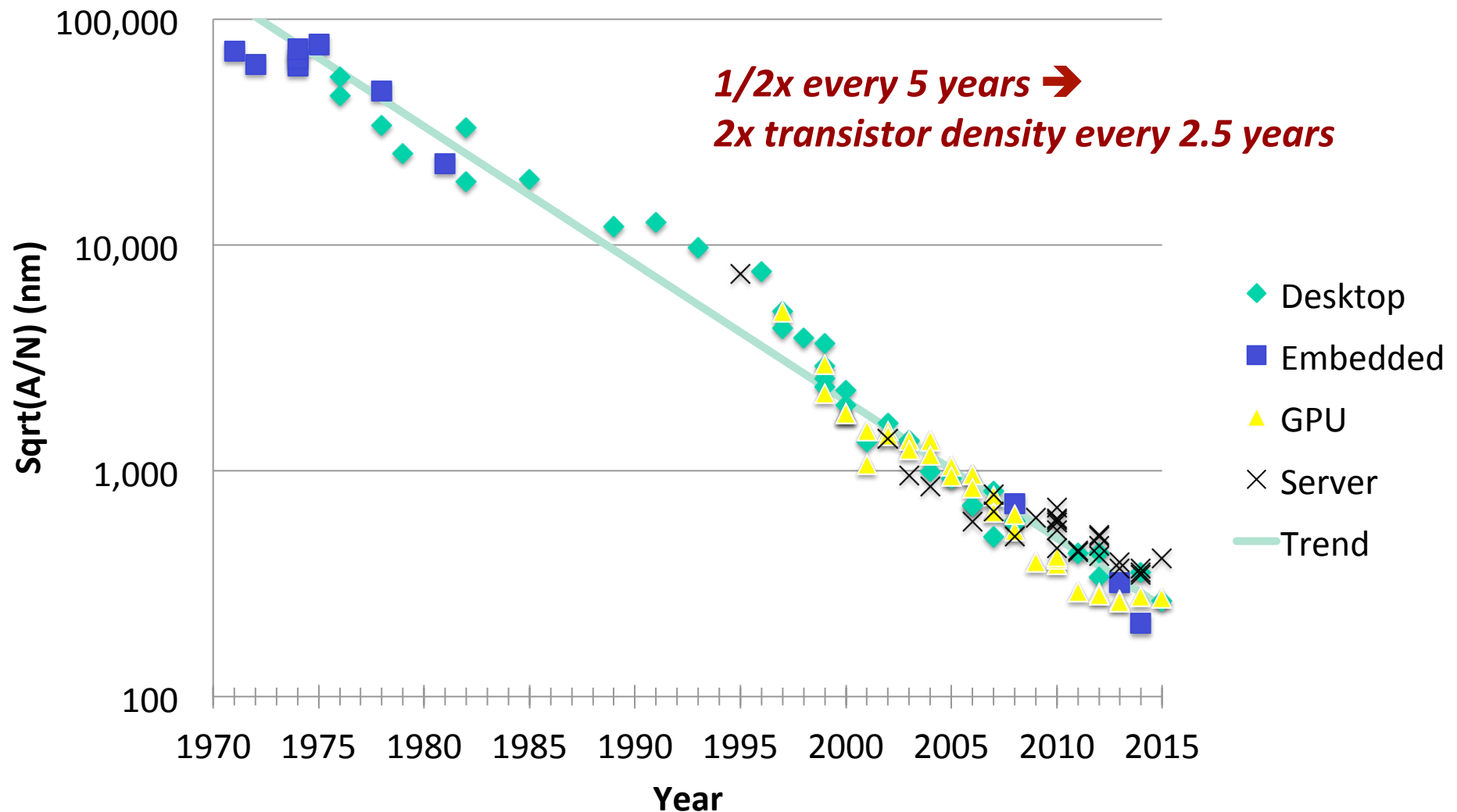
- Area A
- N devices
- Linear Scale L

$$L = \sqrt{A / N}$$



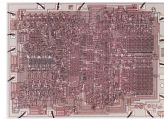
Linear Scaling Trend

Linear Scale by Year

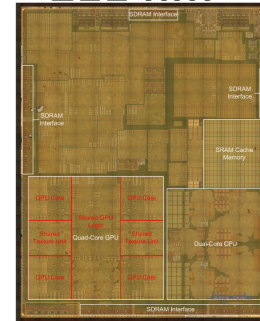


Decreasing Feature Sizes

Intel 4004
1970
2,300 transistors
 $L = 72,000$ nm

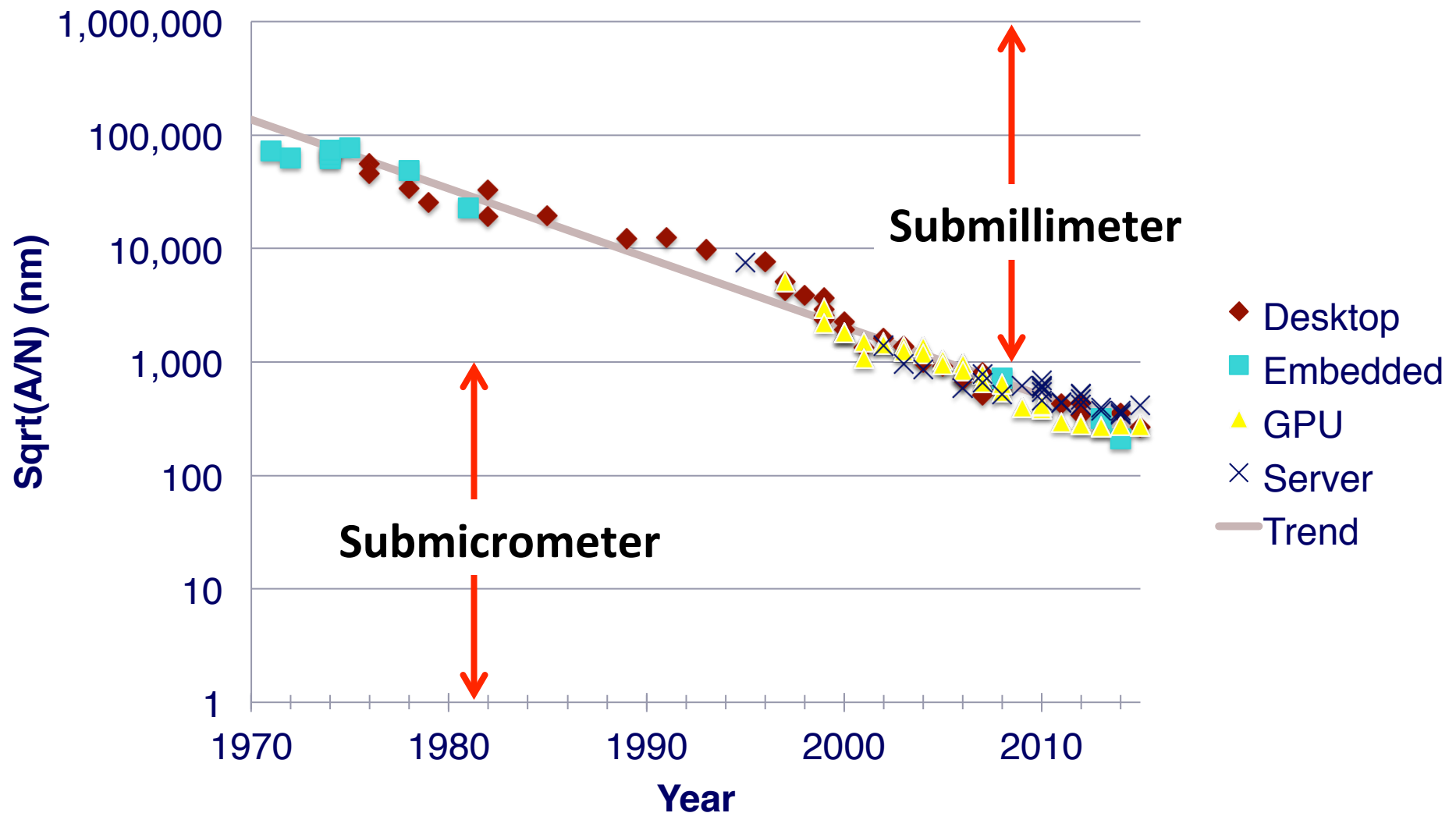


Apple A8
2014
2 B transistors
 $L = 211$ nm

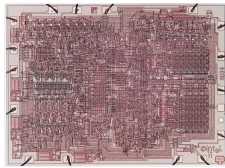
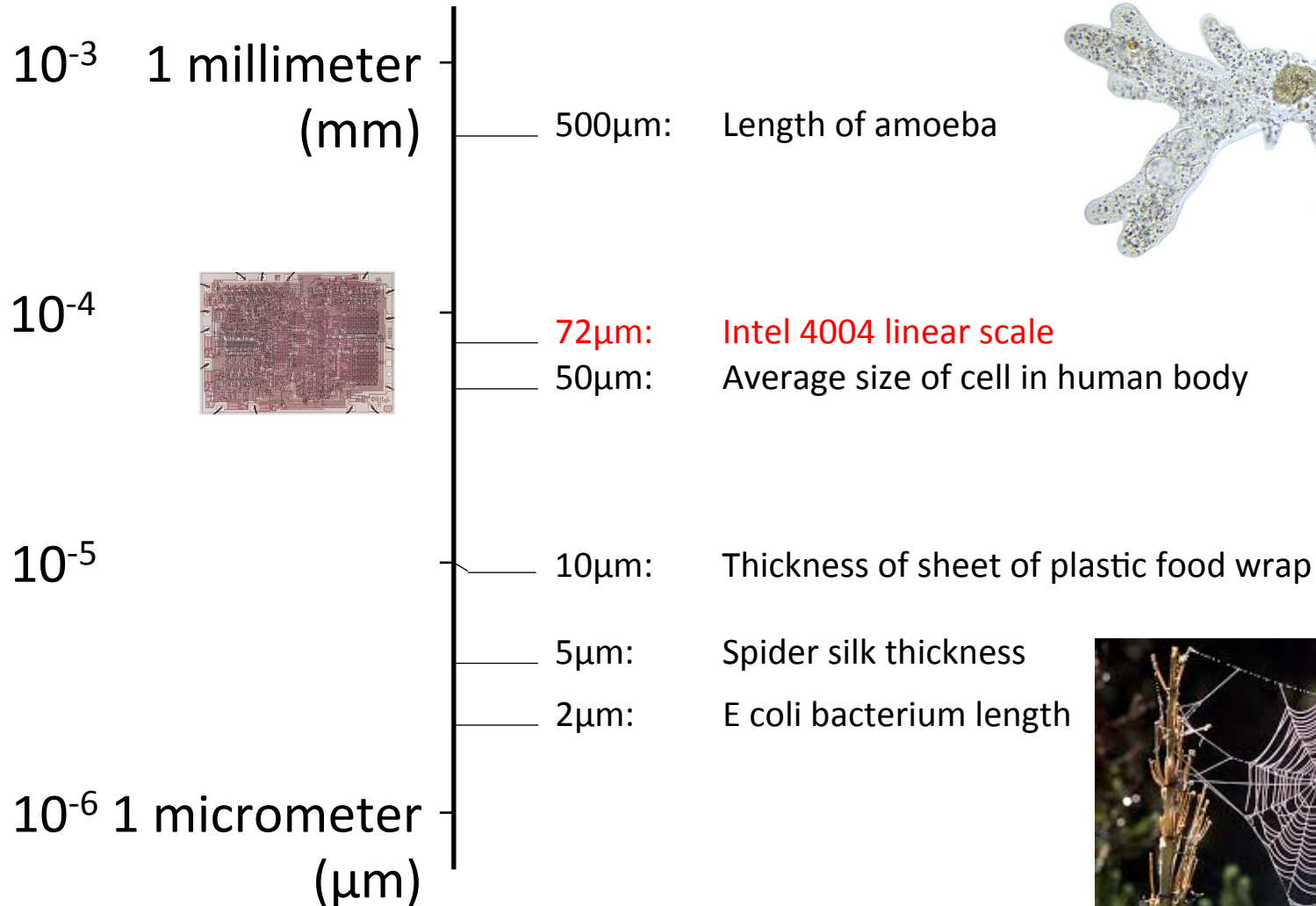


Linear Scaling Trend

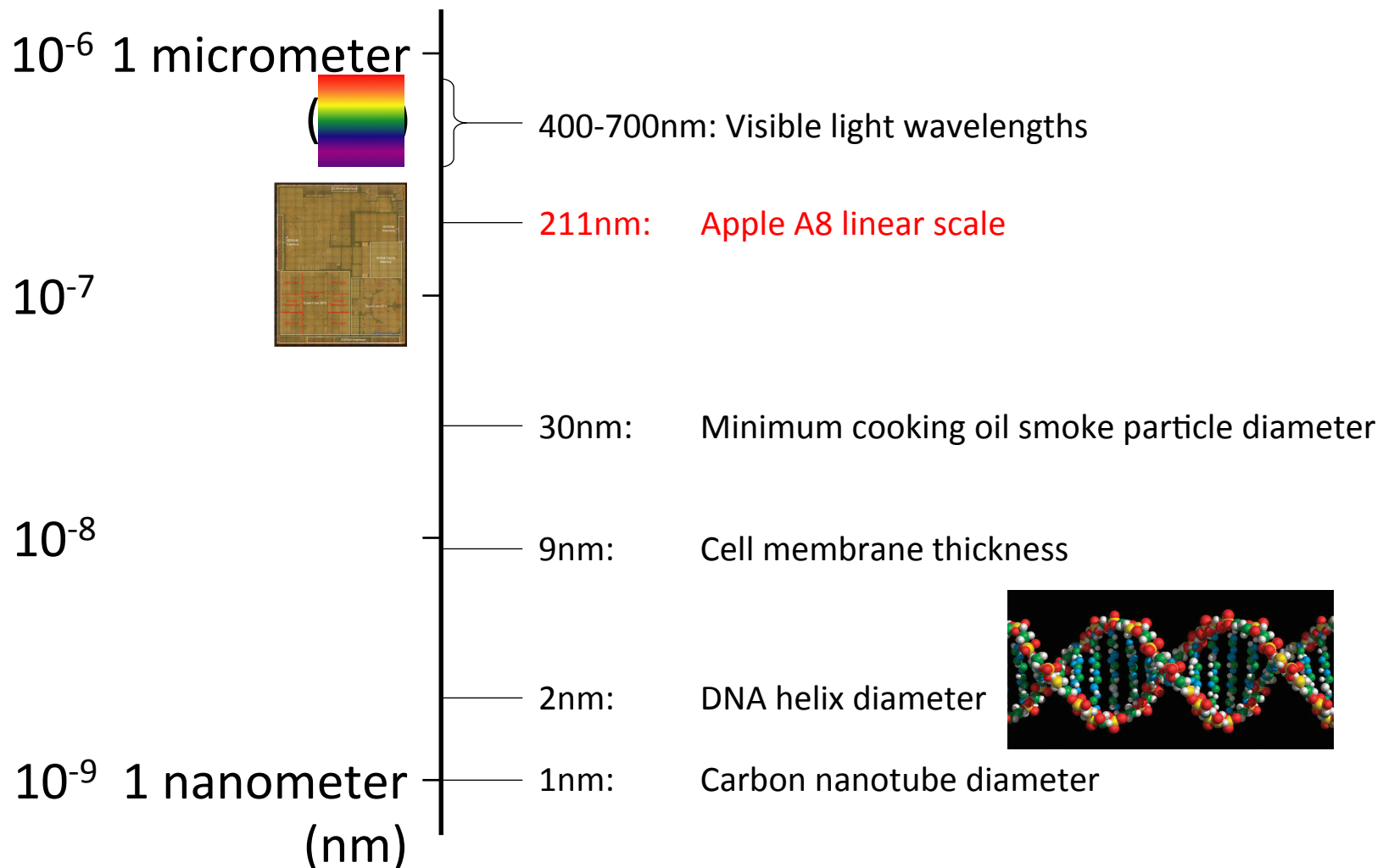
Linear Scale by Year



Submillimeter Dimensions

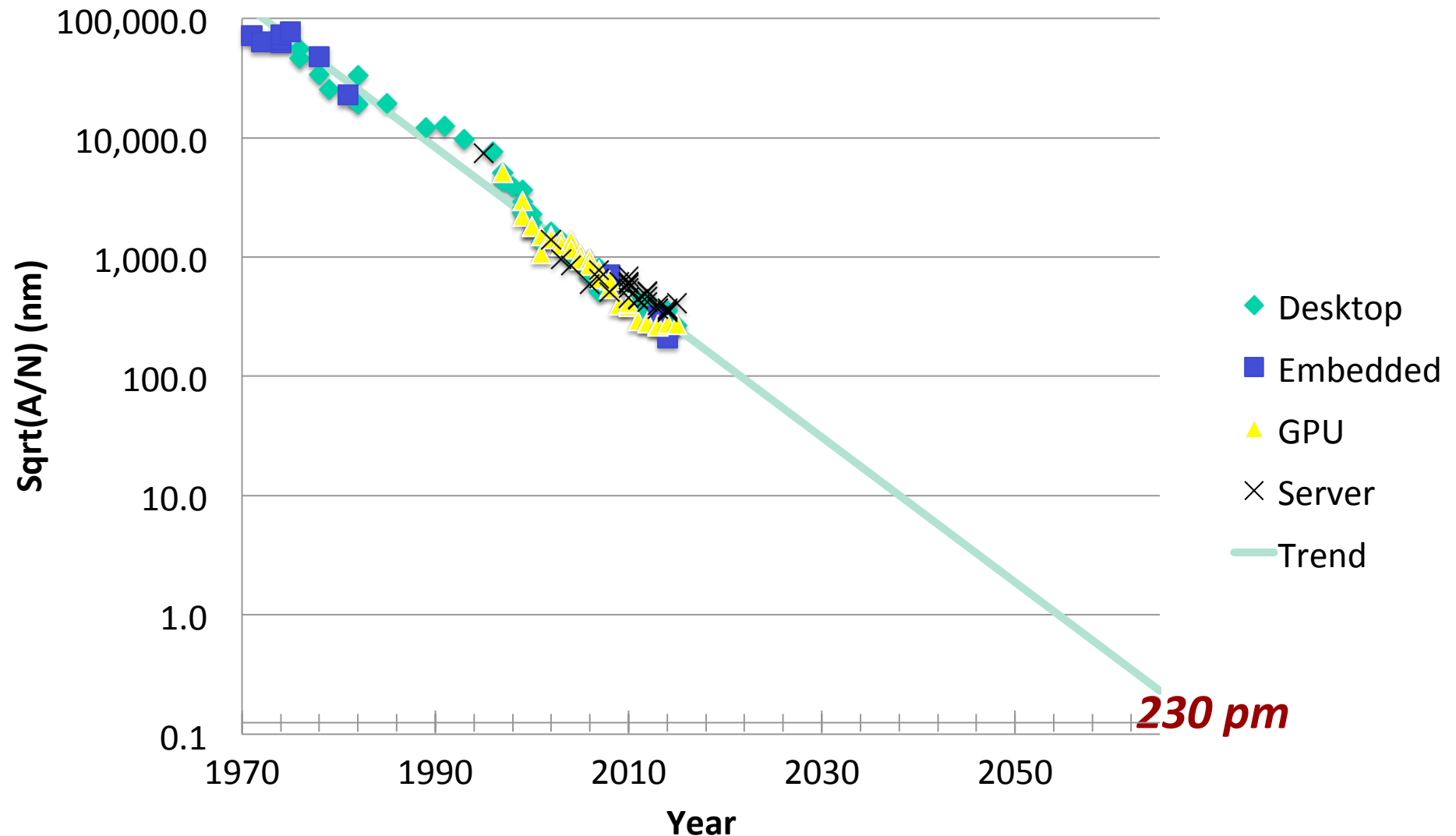


Submicrometer Dimensions

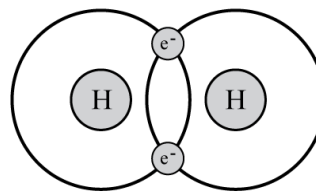
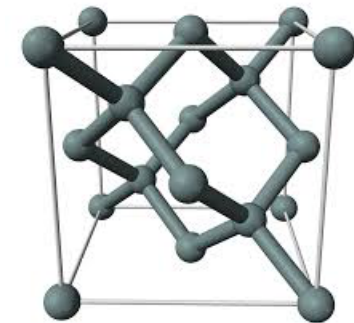
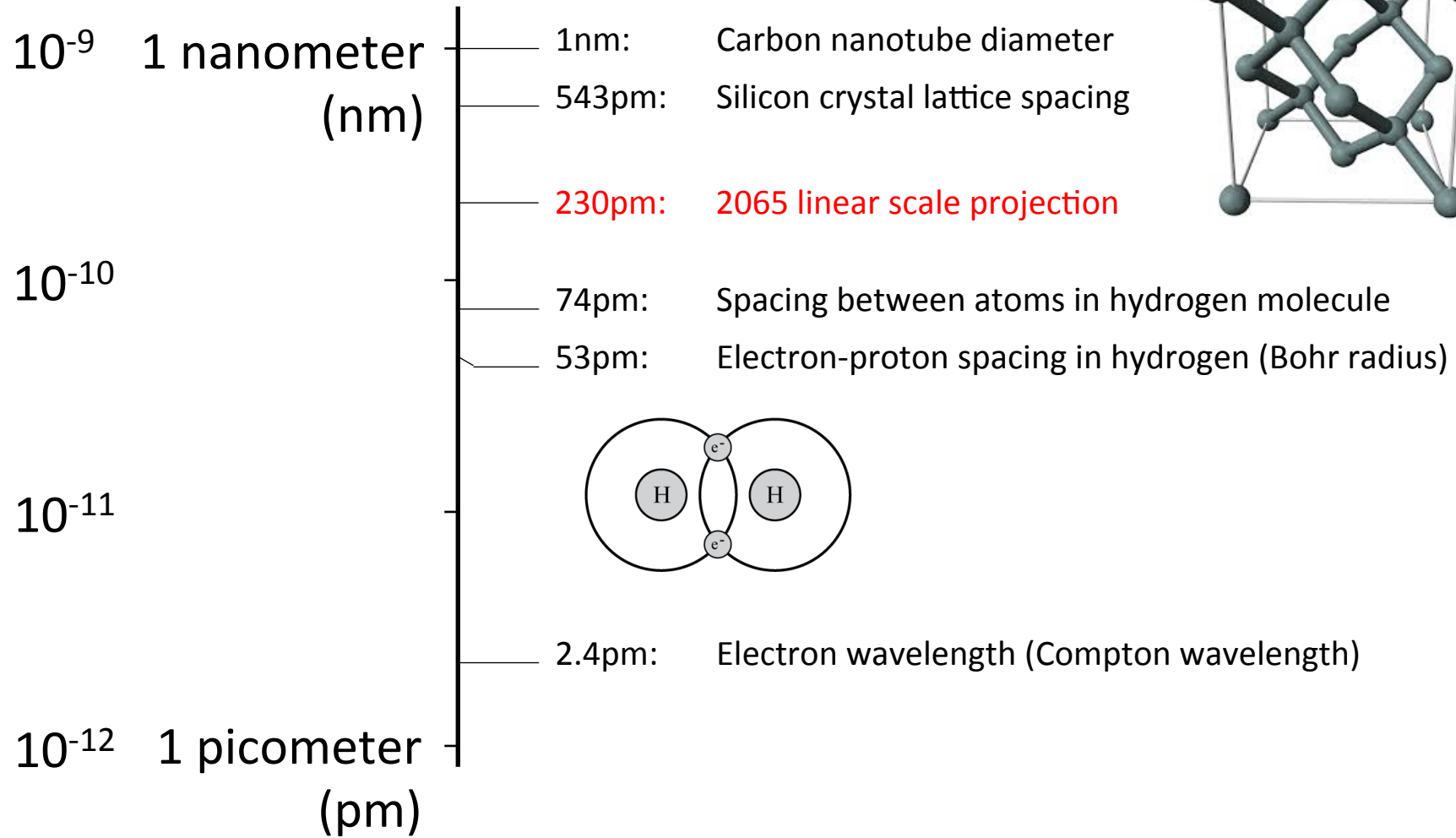


Linear Scaling Extrapolation

Linear Scale by Year



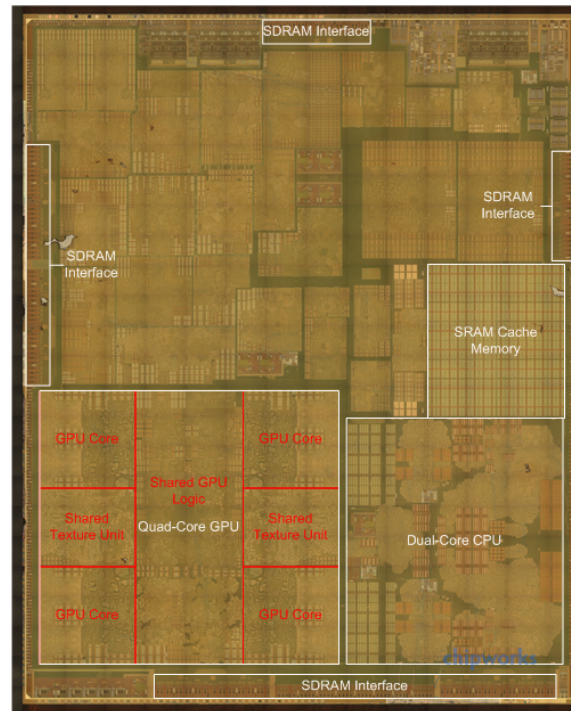
Subnanometer Dimensions



Reaching 2065 Goal

■ Target

- 10^{17} devices
- 400 mm^2
- $L = 63 \text{ pm}$

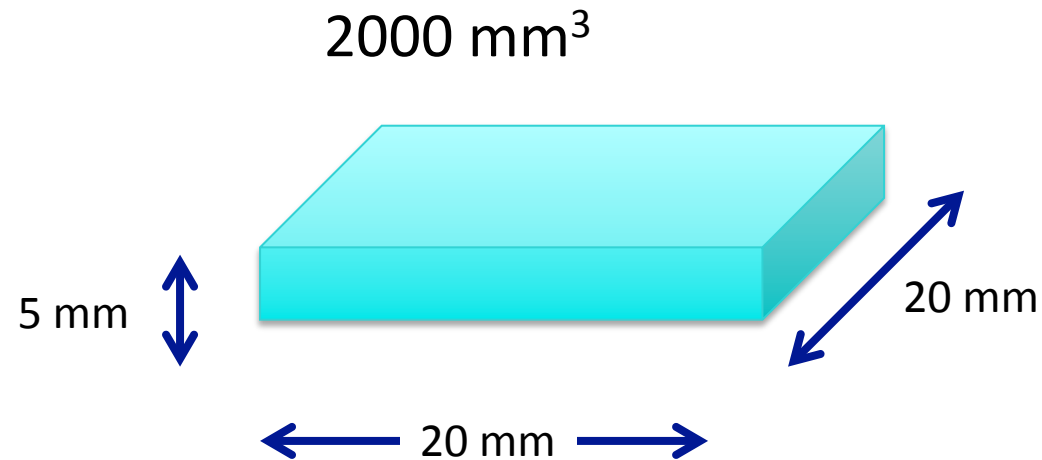


■ Is this possible?

No!

Not with 2-d
fabrication

Fabricating in 3 Dimensions



■ Parameters

- 10¹⁷ devices
- 100,000 logical layers
 - Each 50 nm thick
 - ~1,000,000 physical layers
 - To provide wiring and isolation
- $L = 20$ nm
 - 10x smaller than today



2065 mm³

3D Fabrication Challenges

■ Yield

- How to avoid or tolerate flaws

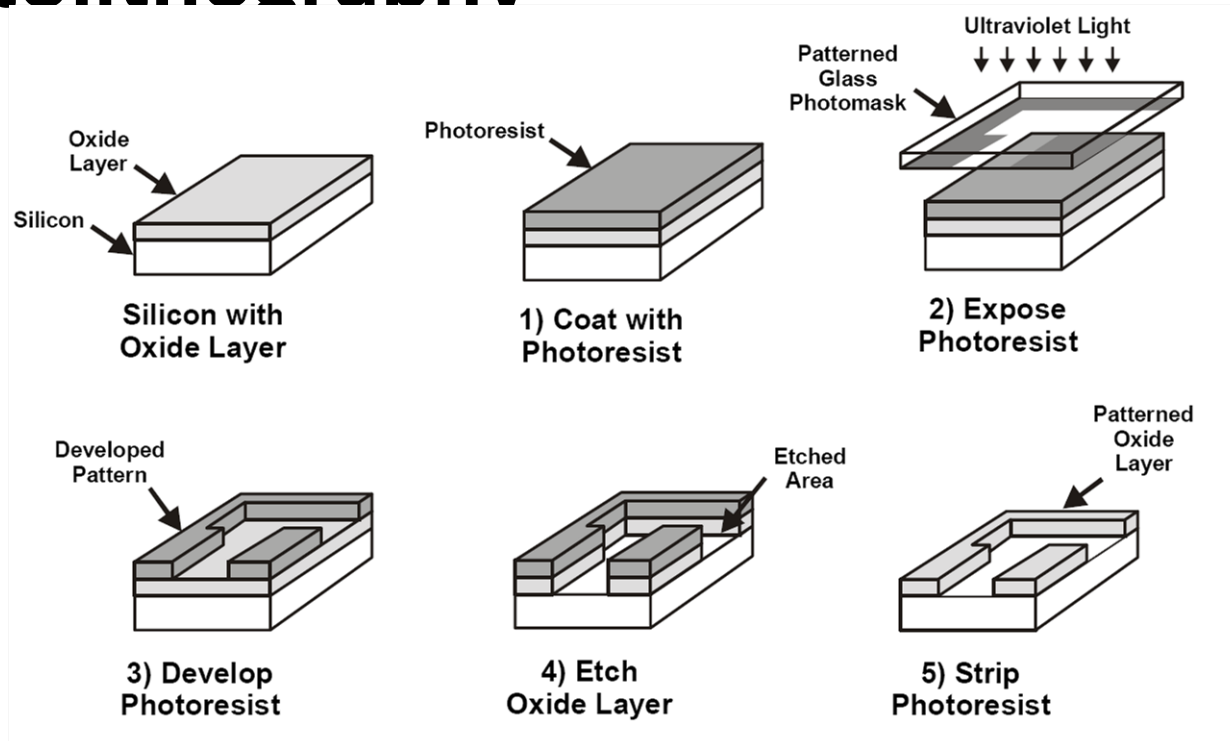
■ Cost

- High cost of lithography

■ Power

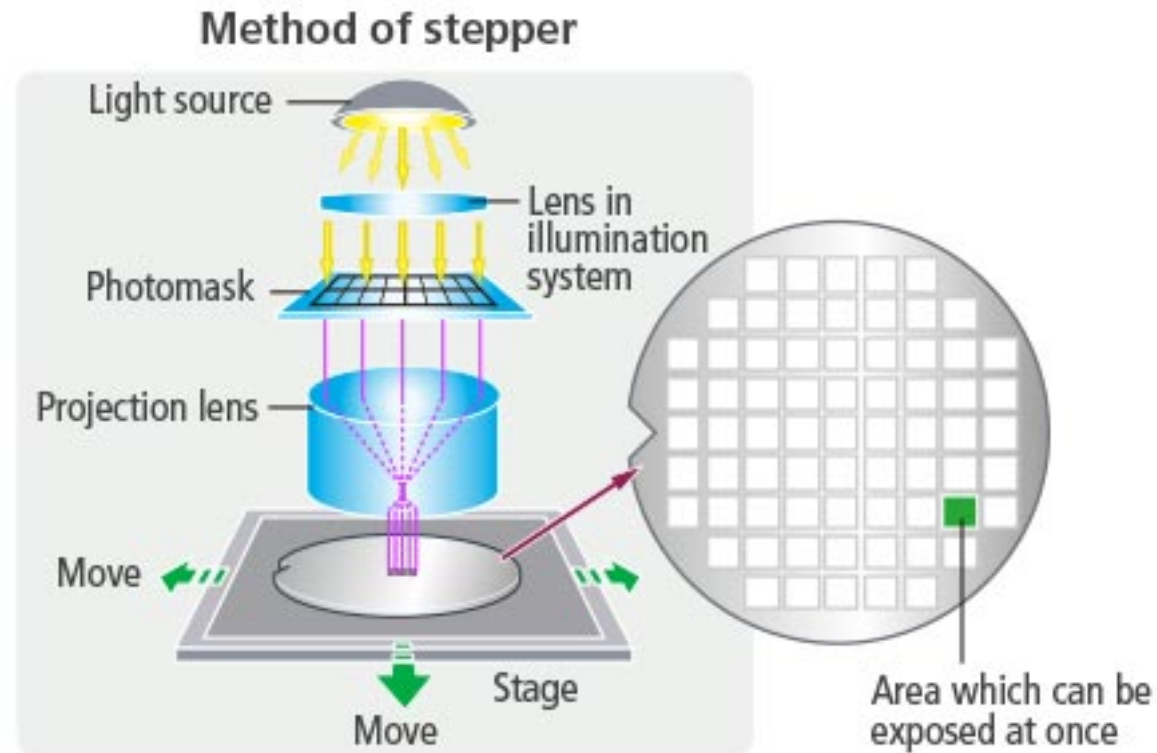
- Keep power consumption within acceptable limits
- Limited energy available
- Limited ability to dissipate heat

Photolithography



- Pattern entire chip in one step
- Modern chips require ~60 lithography steps
- Fabricate N transistor system with $O(1)$ steps

Fabrication Costs



■ Stepper

- Most expensive equipment in fabrication facility
- Rate limiting process step
 - 18s / wafer
- Expose 858 mm² per step
 - 1.2% of chip area

Fabrication Economics

■ Currently

- Fixed number of lithography steps
- Manufacturing cost \$10–\$20 / chip
 - Including amortization of facility

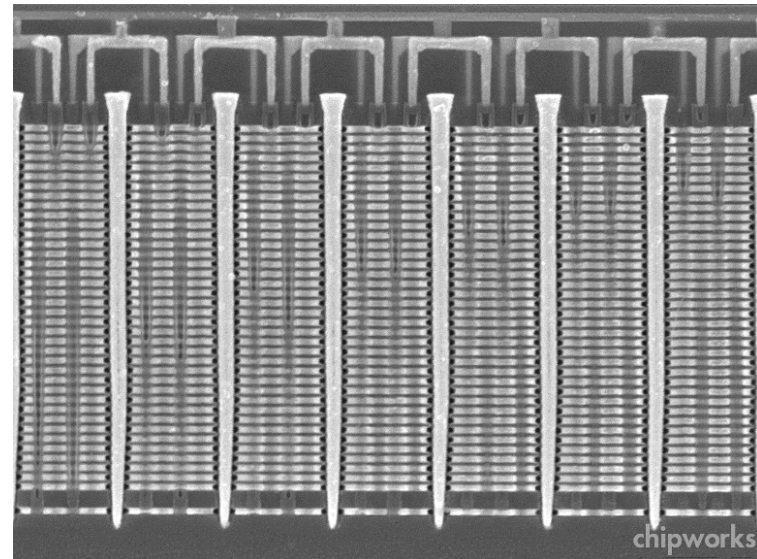
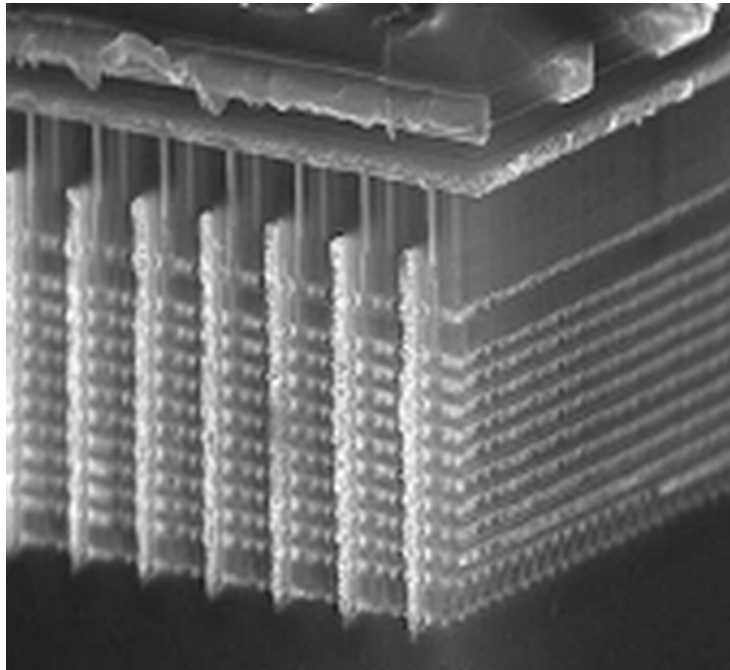
■ Fabricating 1,000,000 physical layers

- Cannot do lithography on every step

■ Options

- Chemical self assembly
 - Devices generate themselves via chemical processes
- Pattern multiple layers at once

Samsung V-Nand Flash Example



- Build up layers of unpatterned material
- Then use lithography to slice, drill, etch, and deposit material across all layers
- ~30 total masking steps
- Up to 48 layers of memory cells
- Exploits particular structure of flash memory circuits

Meeting Power Constraints



- 2 B transistors
- 2 GHz operation
- 1—5 W

***Can we increase number of devices
by 500,000x without increasing
power requirement?***



- 64 B neurons
- 100 Hz operation
- 15—25 W
 - Liquid cooling
 - Up to 25% body's total energy consumption

Challenges to Moore's Law: Economic

■ Growing Capital Costs

- State of art fab line ~\$20B
- Must have very high volumes to amortize investment
- Has led to major consolidations



Dennard Scaling

- Due to Robert Dennard, IBM, 1974
- Quantifies benefits of Moore's Law

■ How to shrink an IC Process

- Reduce horizontal and vertical dimensions by k
- Reduce voltage by k

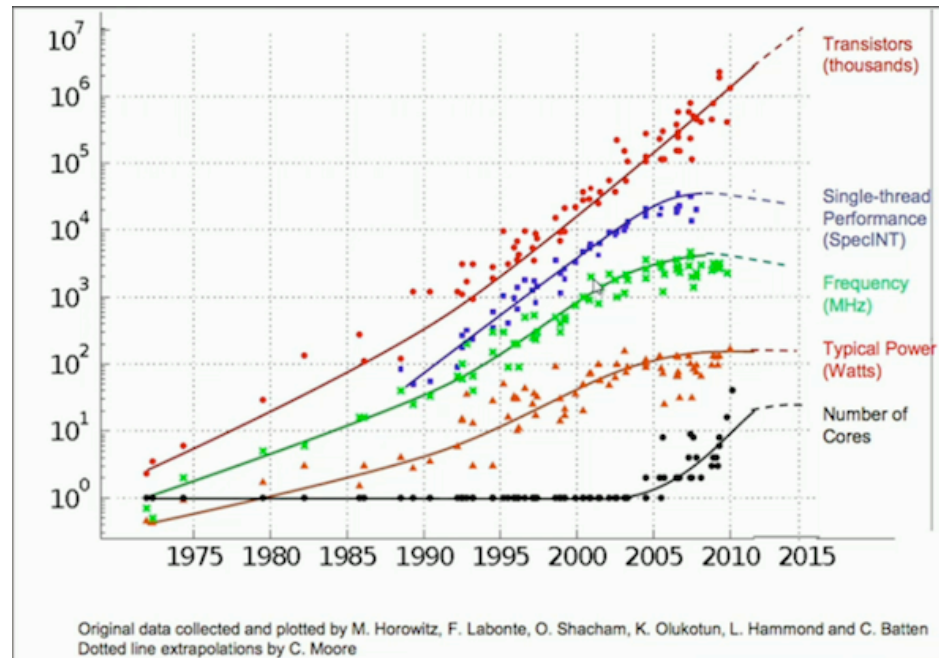
■ Outcomes

- Devices / chip increase by k^2
- Clock frequency increases by k
- Power / chip constant

■ Significance

- Increased capacity and performance
- No increase in power

End of Dennard Scaling



■ What Happened?

- Can't drop voltage below $\sim 1V$
- Reached limit of power / chip in 2004
- More logic on chip (Moore's Law), but can't make them run faster
 - Response has been to increase cores / chip

Final Thoughts about Technology

- **Compared to future, past 50 years will seem fairly straightforward**
 - 50 years of using photolithography to pattern transistors on two-dimensional surface
- **Questions about future integrated systems**
 - Can we build them?
 - What will be the technology?
 - Are they commercially viable?
 - Can we keep power consumption low?
 - What will we do with them?
 - How will we program / customize them?

HIGH-PERFORMANCE COMPUTING

Comparing Two Large-Scale Systems

■ Oakridge Titan



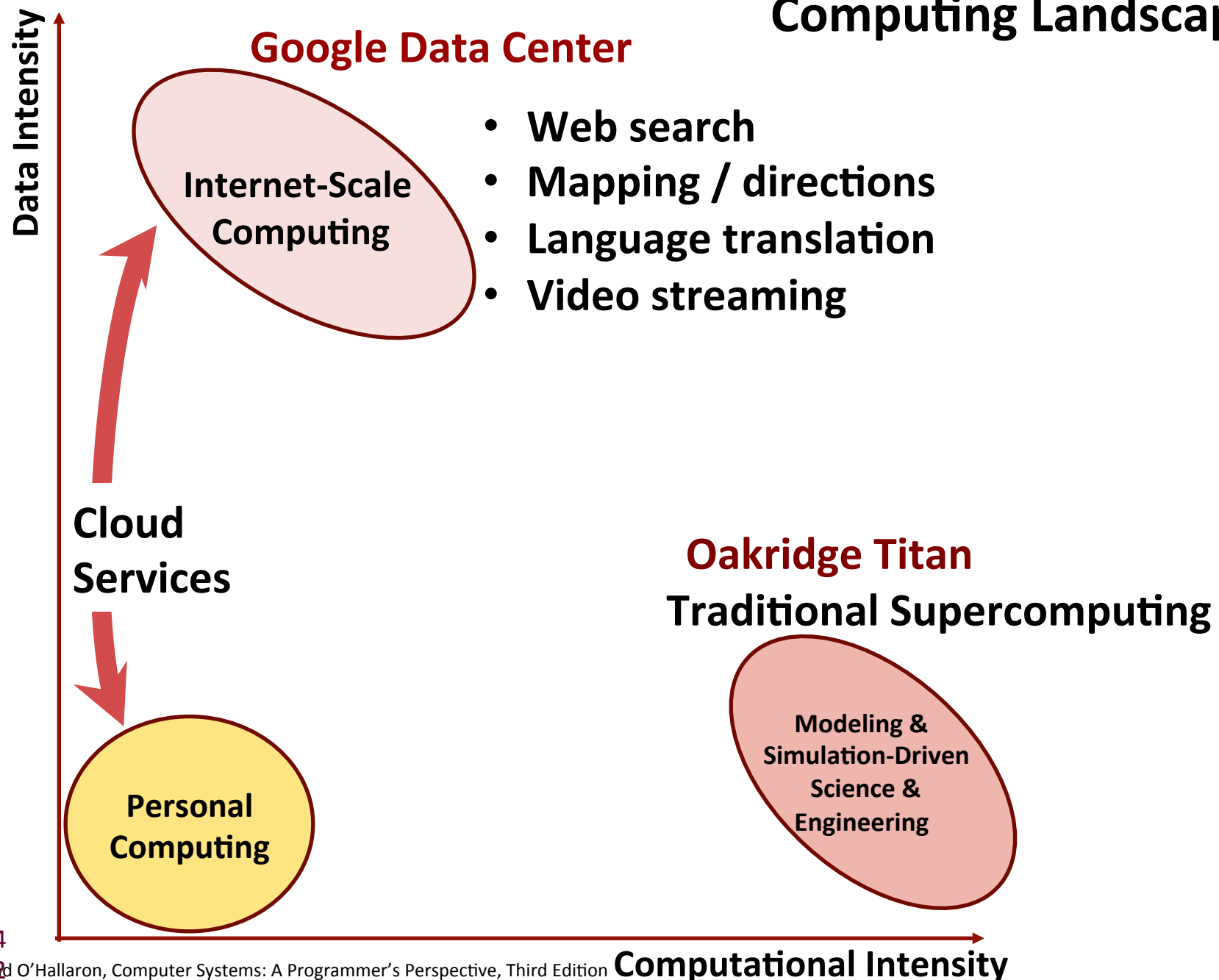
- Monolithic supercomputer (3rd fastest in world)
- Designed for compute-intensive applications

■ Google Data Center



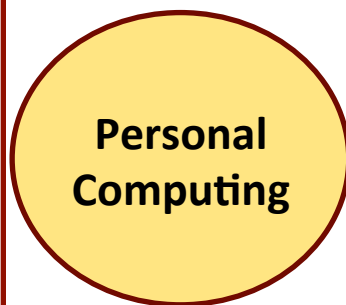
- Servers to support millions of customers
- Designed for data collection, storage, and analysis

Computing Landscape

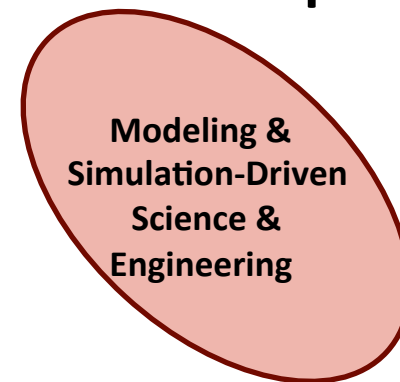


Supercomputing Landscape

Data Intensity

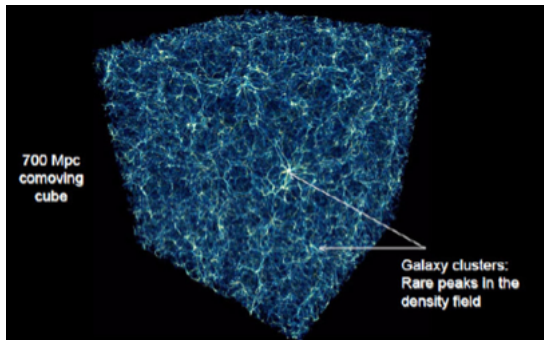


Oakridge Titan
Traditional Supercomputing

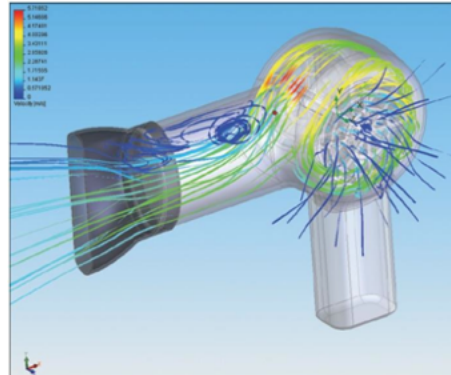


Computational Intensity

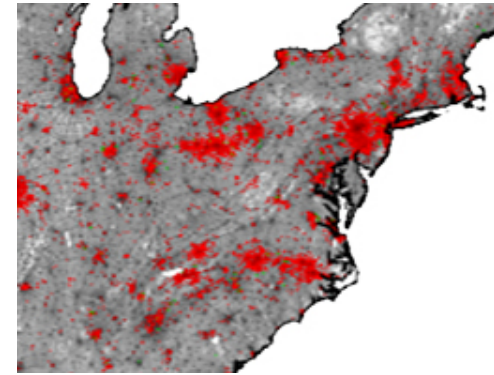
Supercomputer Applications



Science



Industrial Products



Public Health

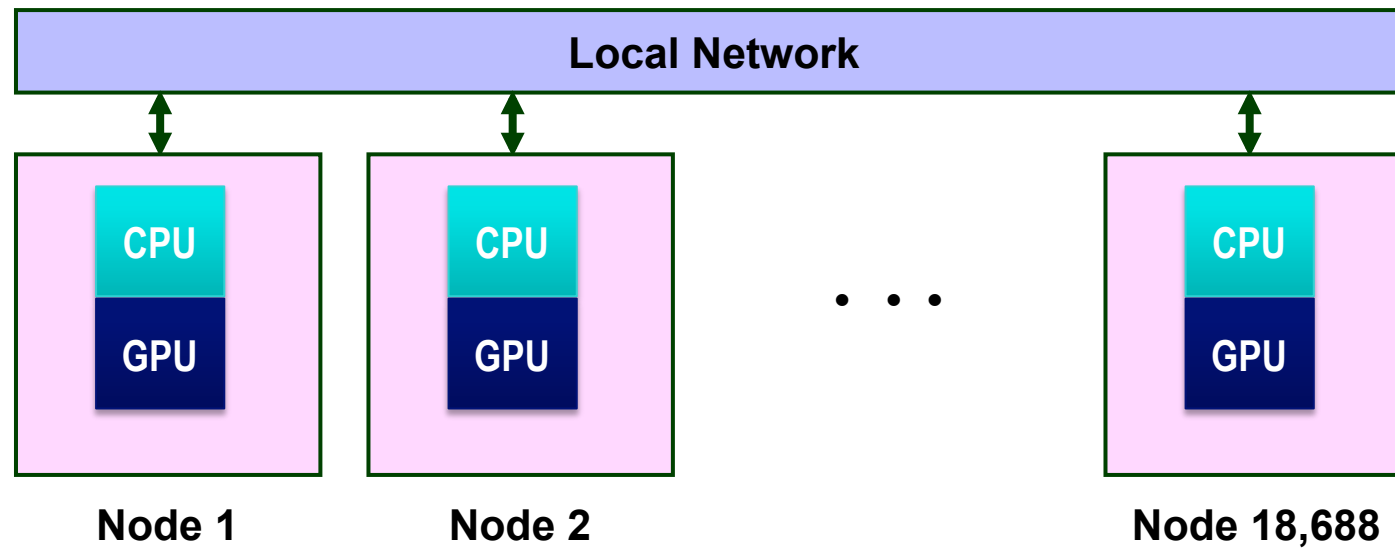
■ Simulation-Based Modeling

- System structure + initial conditions + transition behavior
- Discretize time and space
- Run simulation to see what happens

■ Requirements

- Model accurately reflects actual system
- Simulation faithfully captures model

Titan Hardware



■ Each Node

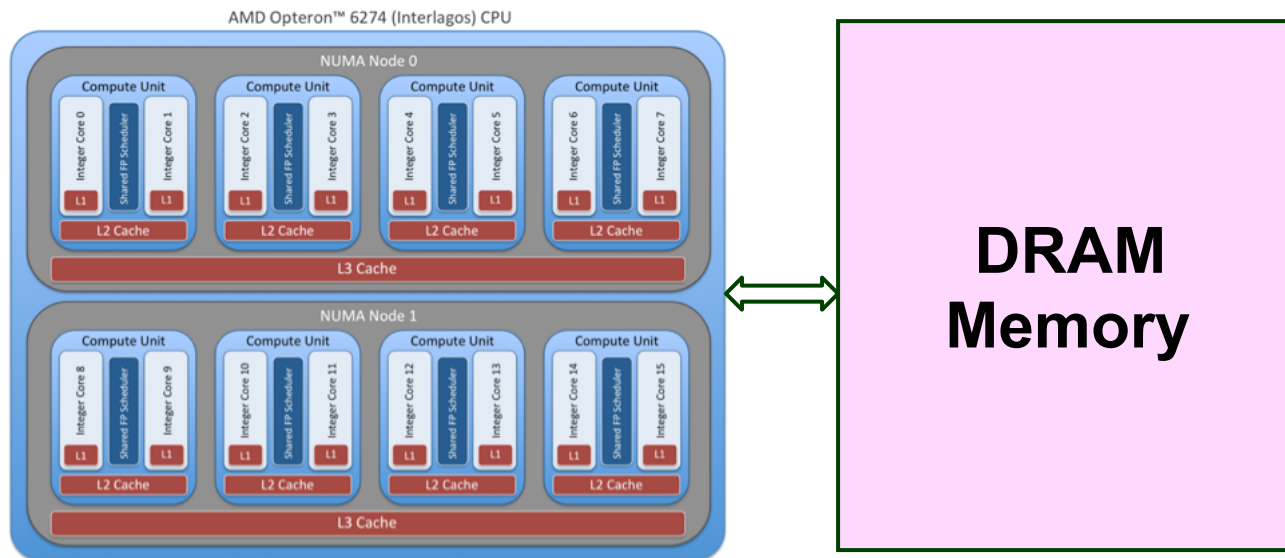
- AMD 16-core processor
- nVidia Graphics Processing Unit
- 38 GB DRAM
- *No disk drive*

■ Overall

- 7MW, \$200M



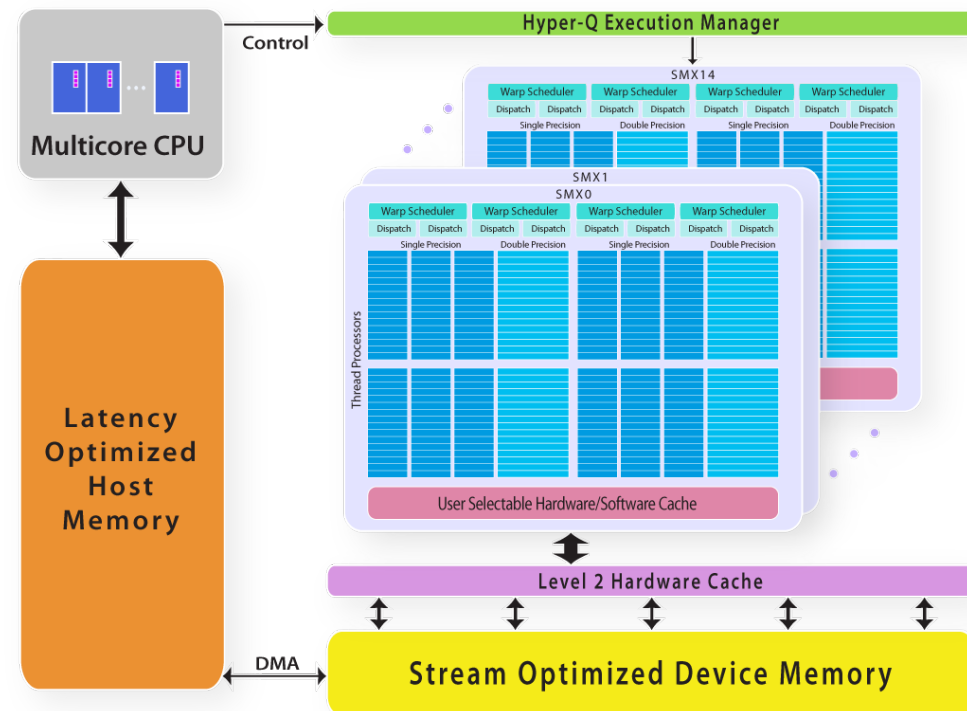
Titan Node Structure: CPU



■ CPU

- 16 cores sharing common memory
- Supports multithreaded programming
- $\sim 0.16 \times 10^{12}$ floating-point operations per second (FLOPS) peak performance

Titan Node Structure: GPU



©2013 The Portland Group, Inc.

■ Kepler GPU

- 14 multiprocessors
- Each with 12 groups of 16 stream processors
 - $14 \times 12 \times 16 = 2688$
- Single-Instruction, Multiple-Data parallelism
 - Single instruction controls all processors in group
- 4.0×10^{12} FLOPS peak performance

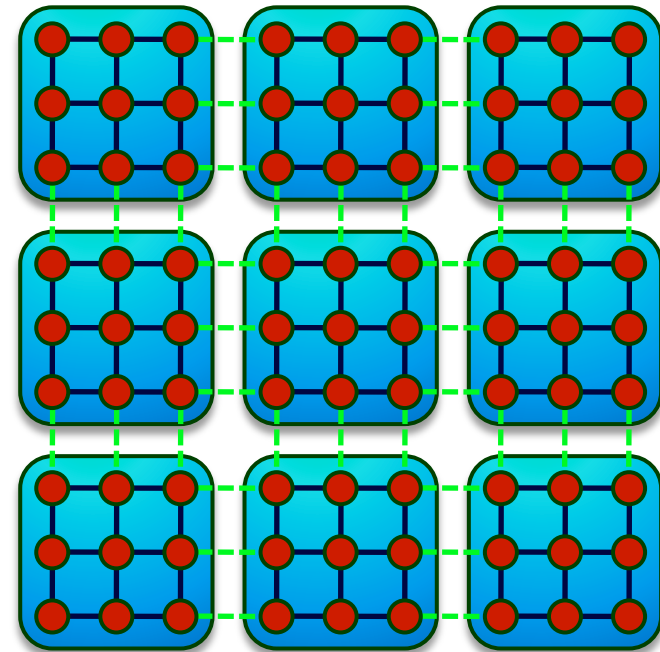
Titan Programming: Principle

■ Solving Problem Over Grid

- E.g., finite-element system
- Simulate operation over time

■ Bulk Synchronous Model

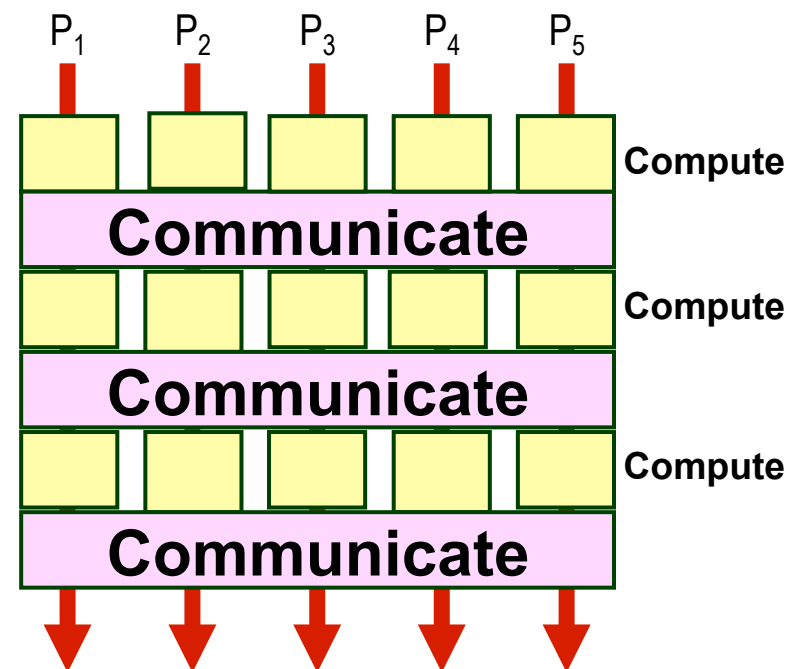
- Partition into Regions
 - p regions for p -node machine
- Map Region per Processor



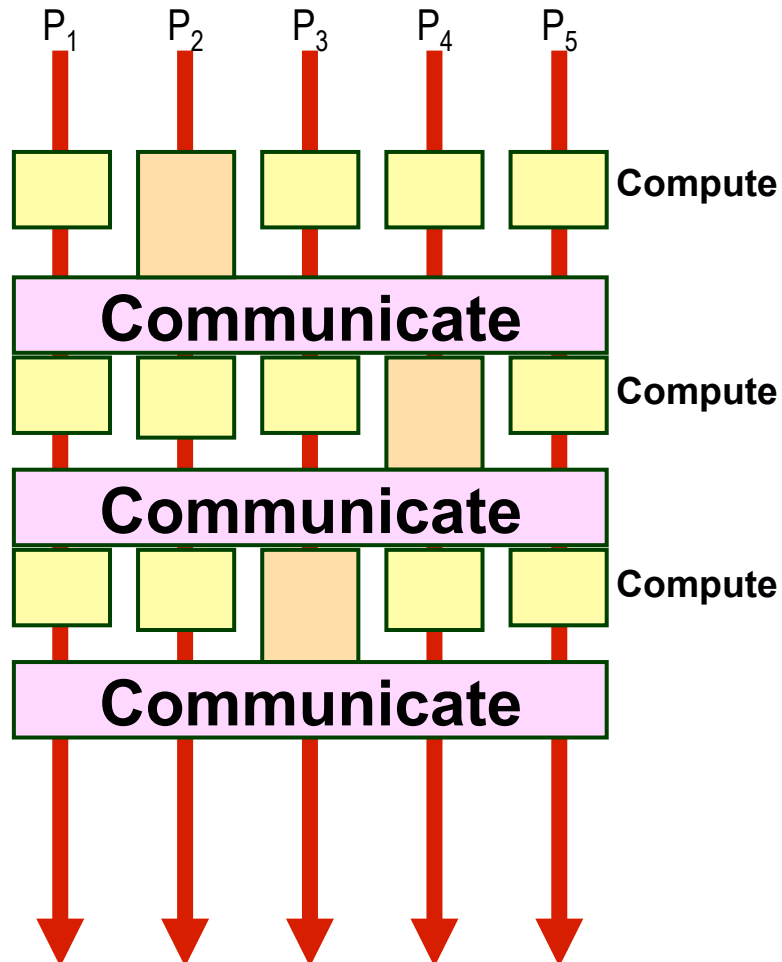
Titan Programming: Principle (cont)

■ Bulk Synchronous Model

- Map Region per Processor
- Alternate
 - All nodes compute behavior of region
 - Perform on GPUs
 - All nodes communicate values at boundaries



Bulk Synchronous Performance



- Limited by performance of slowest processor
- **Strive to keep perfectly balanced**
 - Engineer hardware to be highly reliable
 - Tune software to make as regular as possible
 - Eliminate “noise”
 - Operating system events
 - Extraneous network activity

Titan Programming: Reality

■ System Level

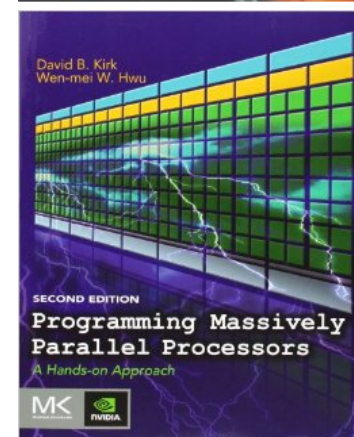
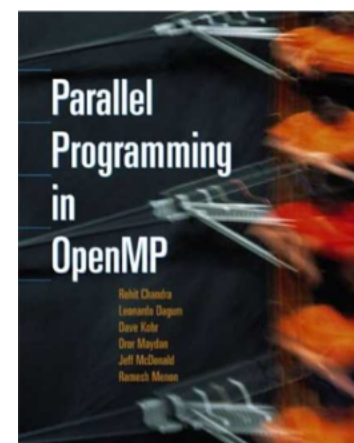
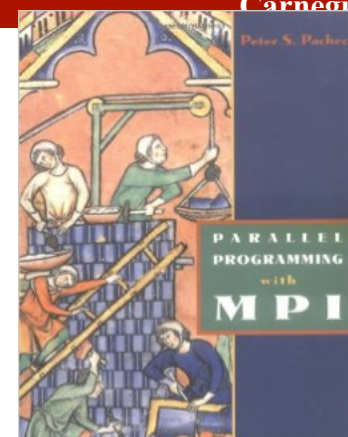
- Message-Passing Interface (MPI) supports node computation, synchronization and communication

■ Node Level

- OpenMP supports thread-level operation of node CPU
- CUDA programming environment for GPUs
 - Performance degrades quickly if don't have perfect balance among memories and processors

■ Result

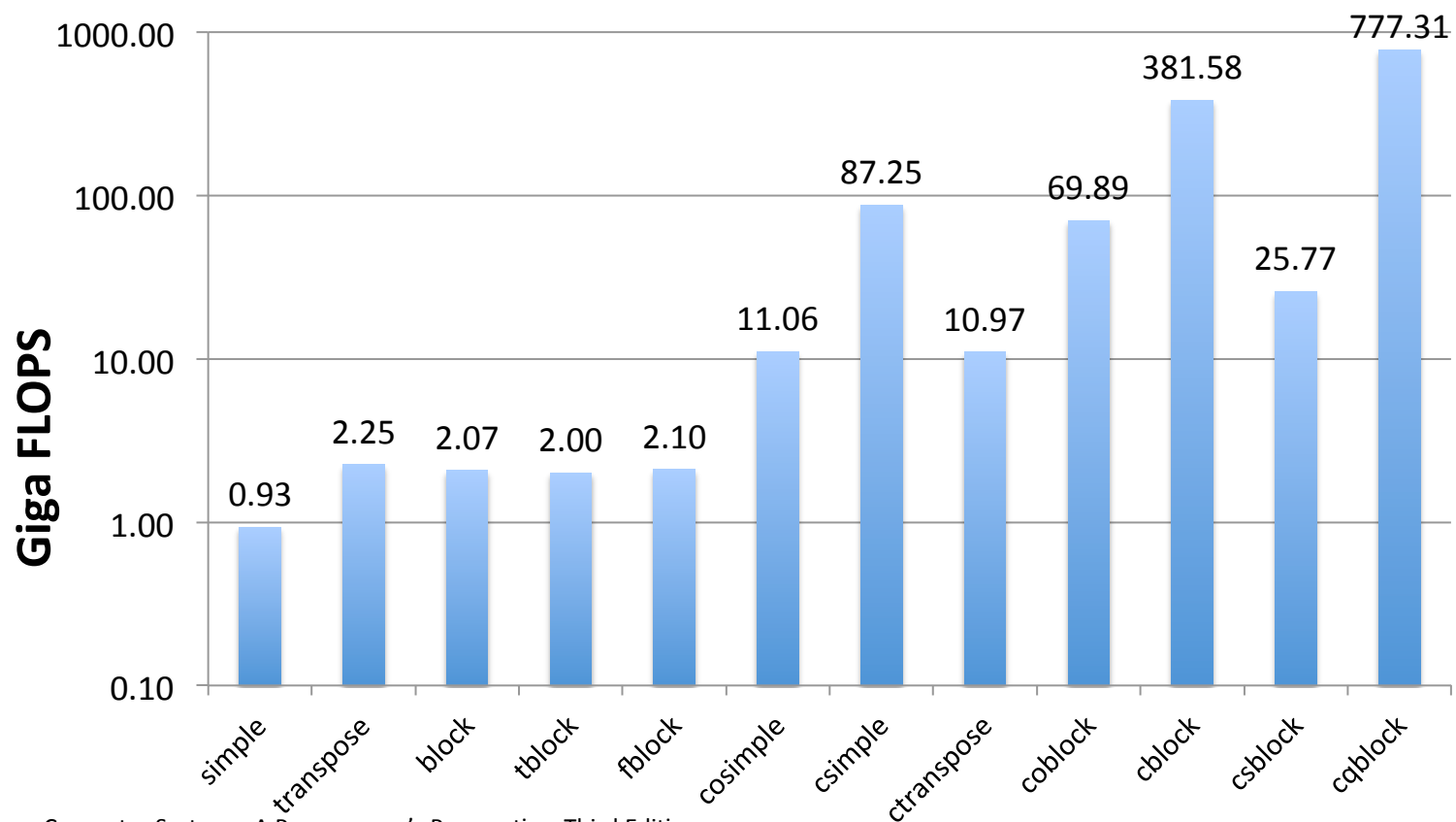
- Single program is complex combination of multiple programming paradigms
- Tend to optimize for specific hardware configuration



My GPU Experience

■ Multiply two 1024 x 1024 matrices (MM)

- 2×10^9 floating point operations
- Express performance in Giga FLOPS
- Program in CUDA and map onto nVidia GPU



Matrix Multiplication Progress

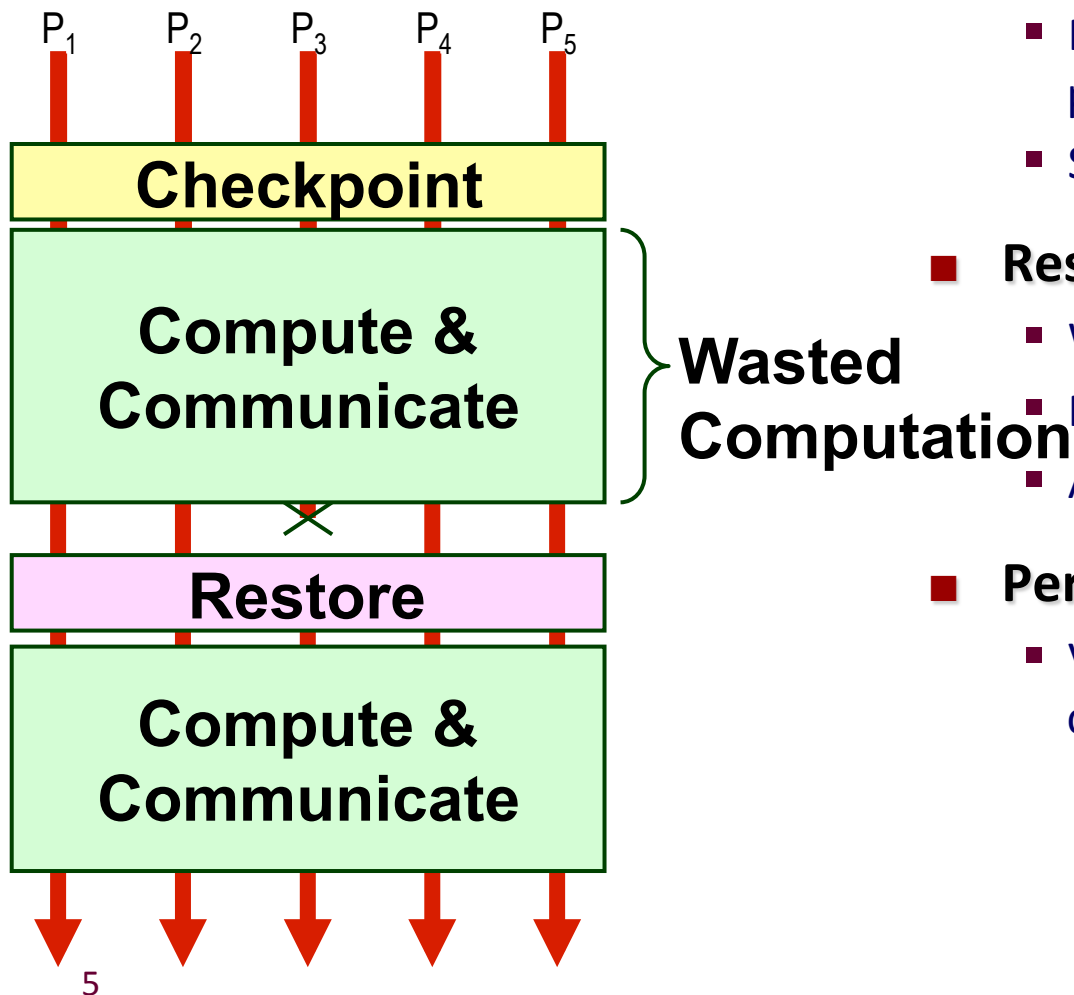
■ Versions

■ Naive	1
■ Simple parallel	11
■ Blocking	70
■ <i>nVidia Example Code</i>	388
■ Reorient memory accesses	382
■ Packed data access	777

■ Observations

- Progress is very nonlinear
 - Not even monotonic
- Requires increased understanding of how program maps onto hardware
- Becomes more specialized to specific hardware configuration

MPI Fault Tolerance



■ Checkpoint

- Periodically store state of all processes
- Significant I/O traffic

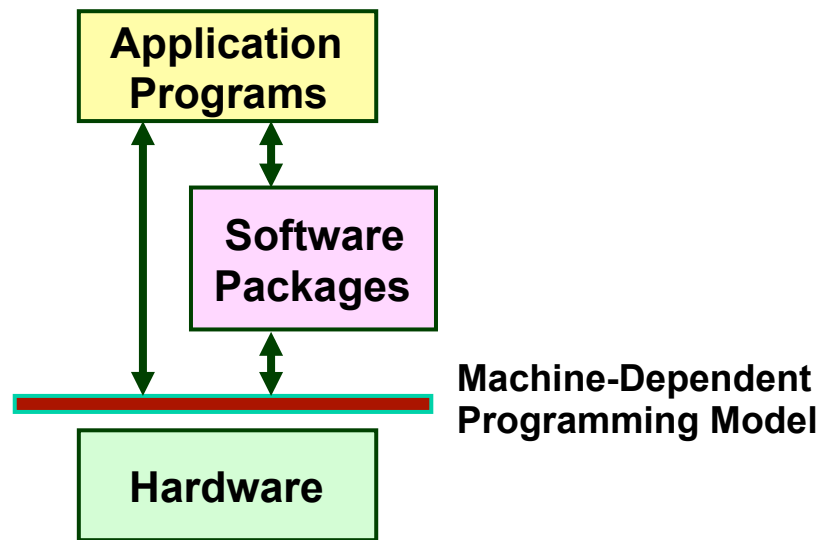
■ Restore

- When failure occurs
- Reset state to that of last checkpoint
- All intervening computation wasted

■ Performance Scaling

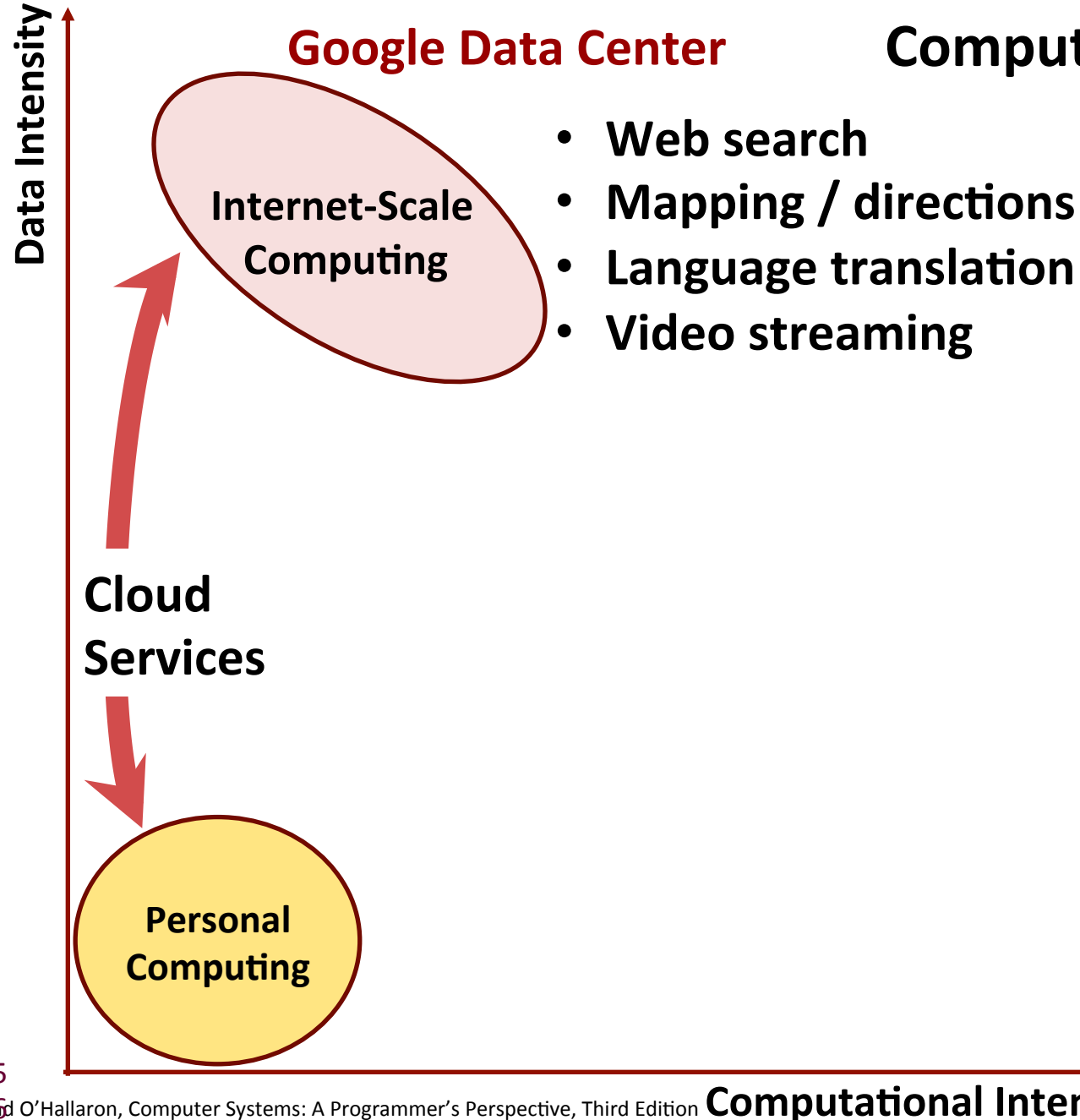
- Very sensitive to number of failing components

Supercomputer Programming Model



- Program on top of bare hardware
- **Performance**
 - Low-level programming to maximize node performance
 - Keep everything globally synchronized and balanced
- **Reliability**
 - Single failure causes major delay
 - Engineer hardware to minimize failures

Data-Intensive Computing Landscape



Internet Computing

■ Web Search

- Aggregate text data from across WWW
- No definition of correct operation
- Do not need real-time updating

■ Mapping Services

- Huge amount of (relatively) static data
- Each customer requires individualized computation



■ Online Documents

- Must be stored reliably
- Must support real-time updating
- (Relatively) small data volumes


Other Data-Intensive Computing Applications

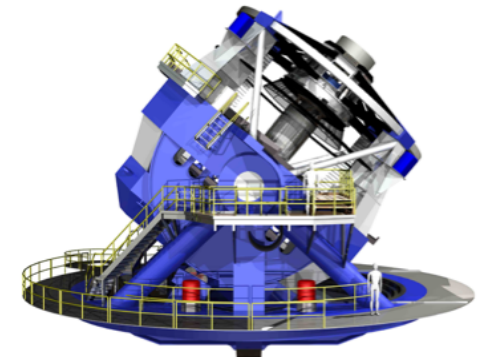
■ Wal-Mart

- 267 million items/day, sold at 6,000 stores
- HP built them 4 PB data warehouse
- Mine data to manage supply chain, understand market trends, formulate pricing strategies



■ LSST

- Chilean telescope  Large Synoptic Survey Telescope very 3 days
- A 3.2 gigapixel digital camera
- Generate 30 TB/day of image data



Data-Intensive Application Characteristics

■ Diverse Classes of Data

- Structured & unstructured
- High & low integrity requirements

■ Diverse Computing Needs

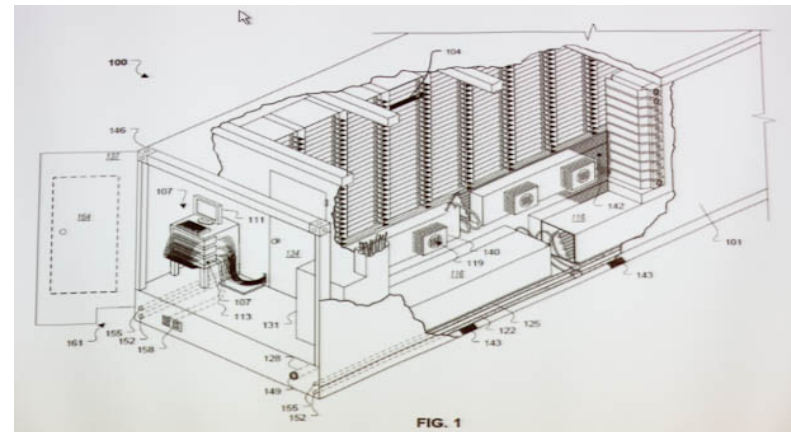
- Localized & global processing
- Numerical & non-numerical
- Real-time & batch processing

Google Data Centers

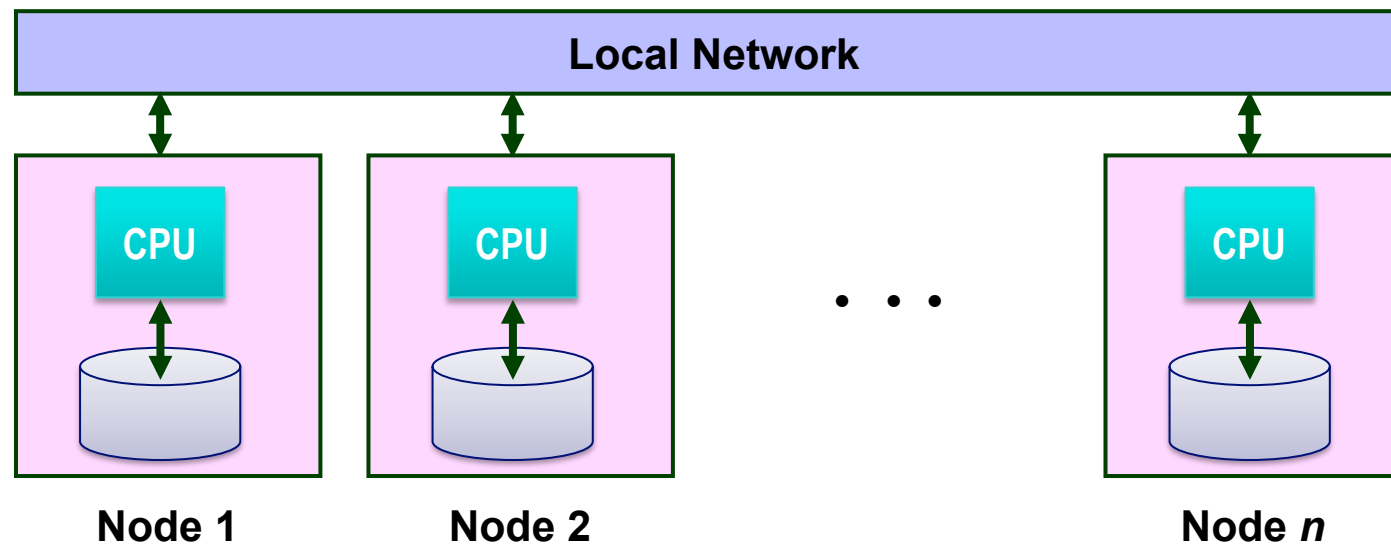


■ Dalles, Oregon

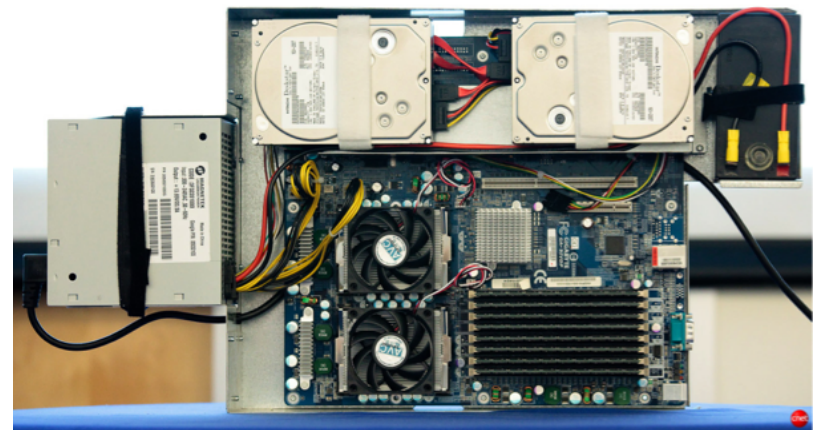
- Hydroelectric power @ 2¢ / KW Hr
 - 50 Megawatts
 - Enough to power 60,000 homes
- Engineered for low cost, modularity & power efficiency
 - Container: 1160 server nodes, 250KW



Google Cluster



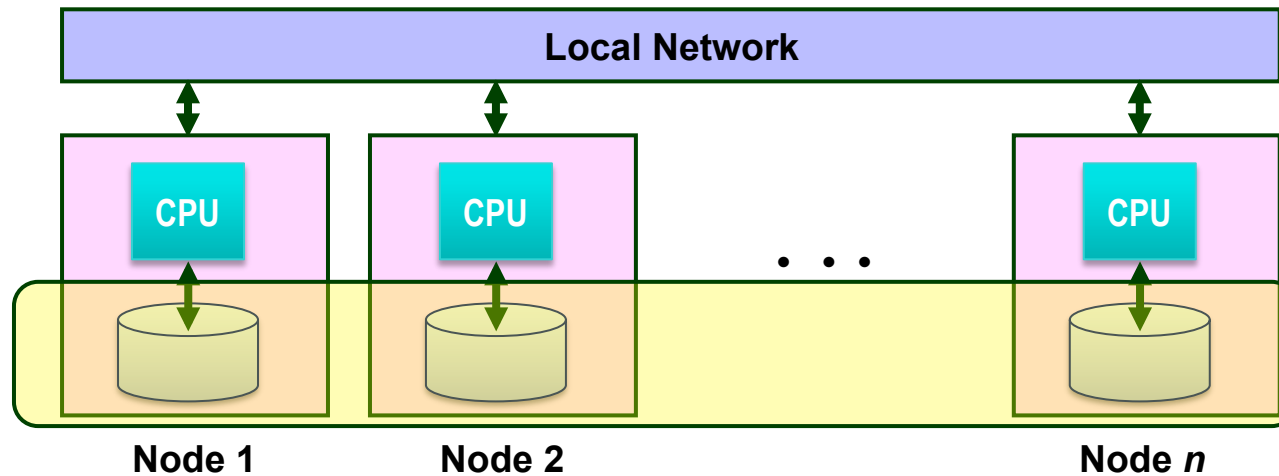
- Typically 1,000–2,000 nodes
- **Node Contains**
 - 2 multicore CPUs
 - 2 disk drives
 - DRAM



Hadoop Project



■ File system with files distributed across nodes

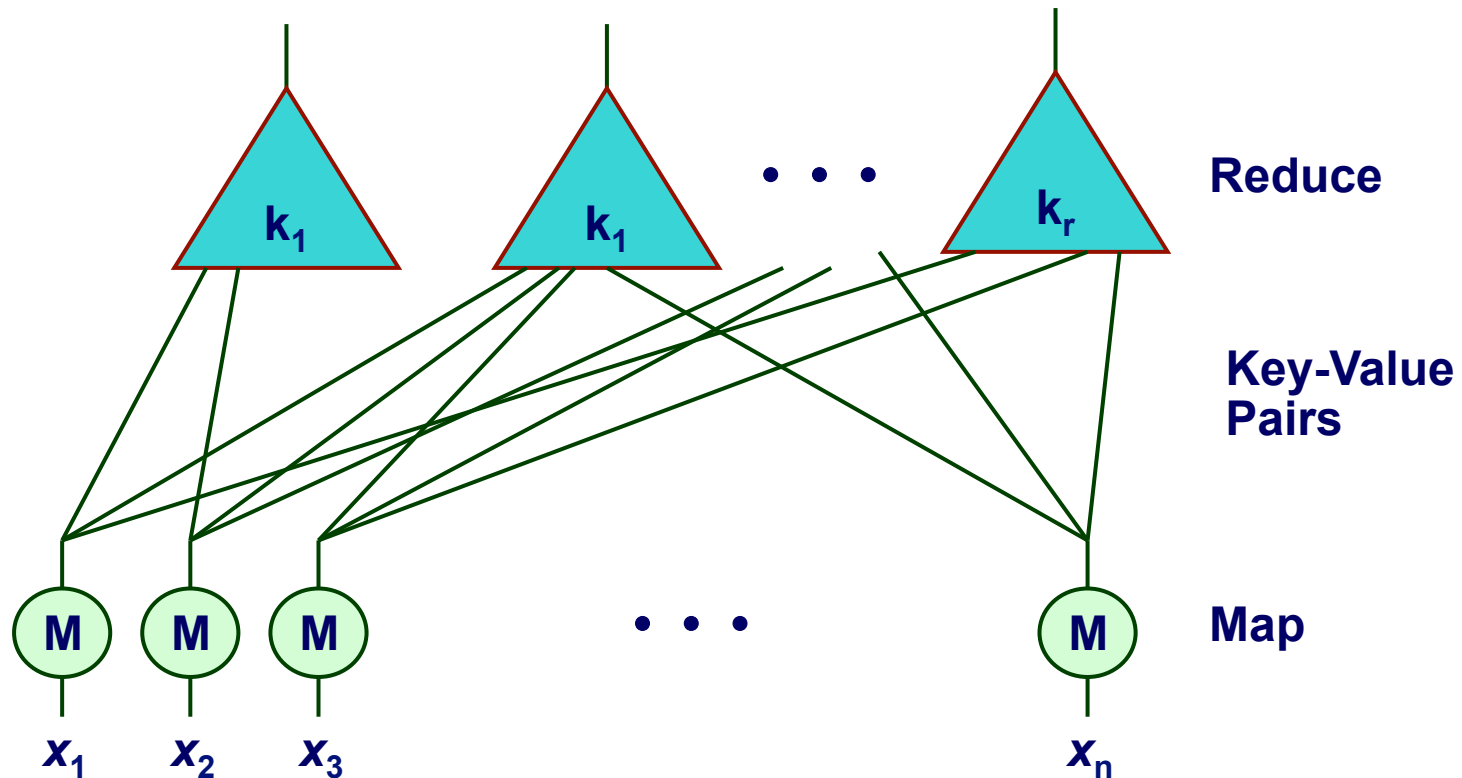


- Store multiple (typically 3 copies of each file)
 - If one node fails, data still available
- Logically, any node has access to any file
 - May need to fetch across network

■ Map / Reduce programming environment

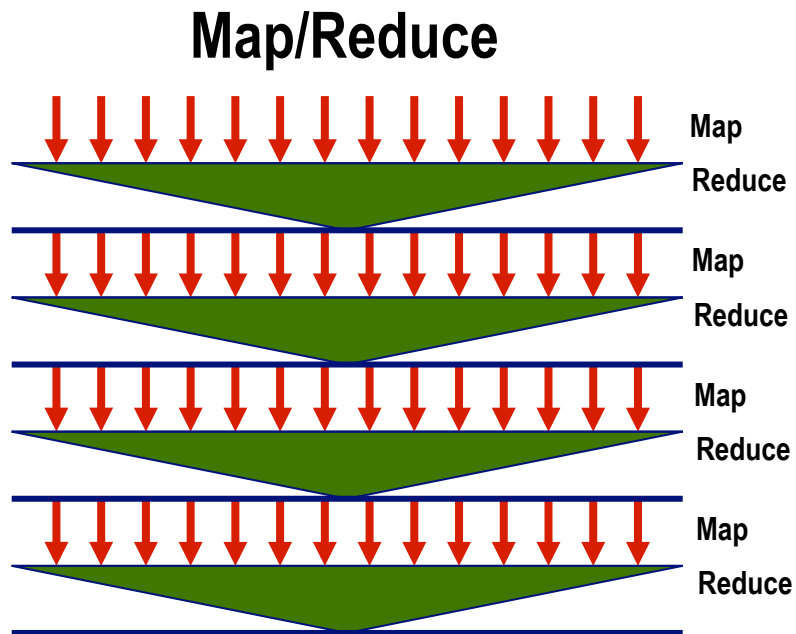
- Software manages execution of tasks on nodes

Map/Reduce Programming Model



- **Map computation across many objects**
 - E.g., 10^{10} Internet web pages
- **Aggregate results in many different ways**
- **System deals with issues of resource allocation & reliability**

Map/Reduce Operation



■ Characteristics

- Computation broken into many, short-lived tasks
 - Mapping, reducing
- Tasks mapped onto processors dynamically
- Use disk storage to hold intermediate results

■ Strengths

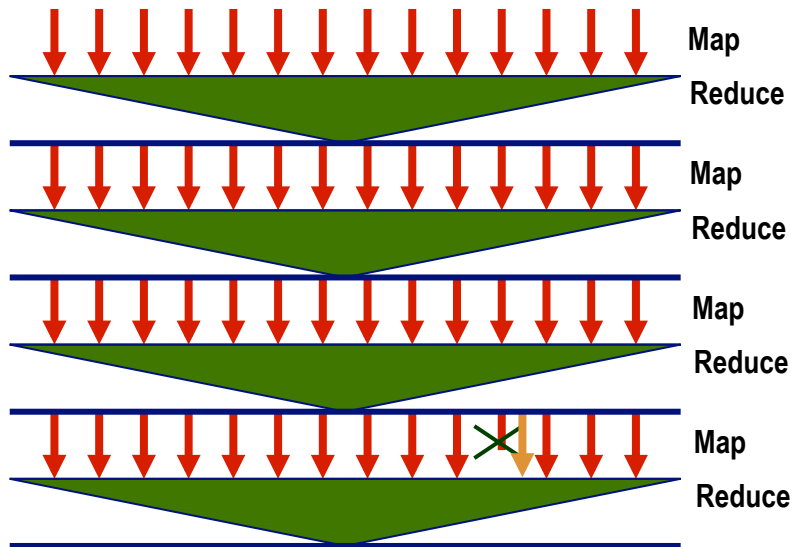
- Flexibility in placement, scheduling, and load balancing
- Can access large data sets

■ Weaknesses

- Higher overhead
- Lower raw performance

Map/Reduce Fault Tolerance

Map/Reduce



■ Data Integrity

- Store multiple copies of each file
- Including intermediate results of each Map / Reduce
 - Continuous checkpointing

■ Recovering from Failure

- Simply recompute lost result
 - Localized effect
- Dynamic scheduler keeps all processors busy

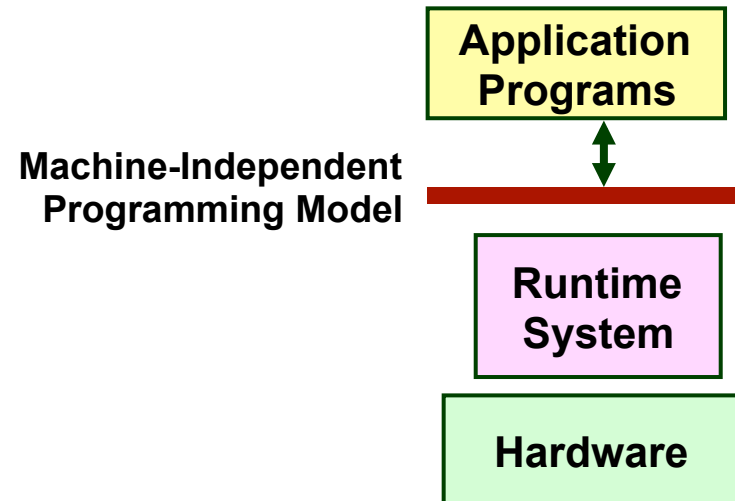
- *Use software to build reliable system on top of unreliable hardware*

Cluster Programming Model

- Application programs written in terms of high-level operations on data
- Runtime system controls scheduling, load balancing, ...

■ Scaling Challenges

- Centralized scheduler forms bottleneck
- Copying to/from disk very costly
- Hard to limit data movement
 - Significant performance factor

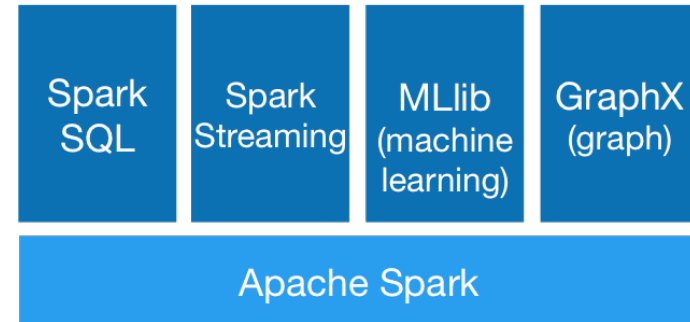


Recent Programming Systems

■ Spark Project



- at U.C., Berkeley
- Grown to have large open source community



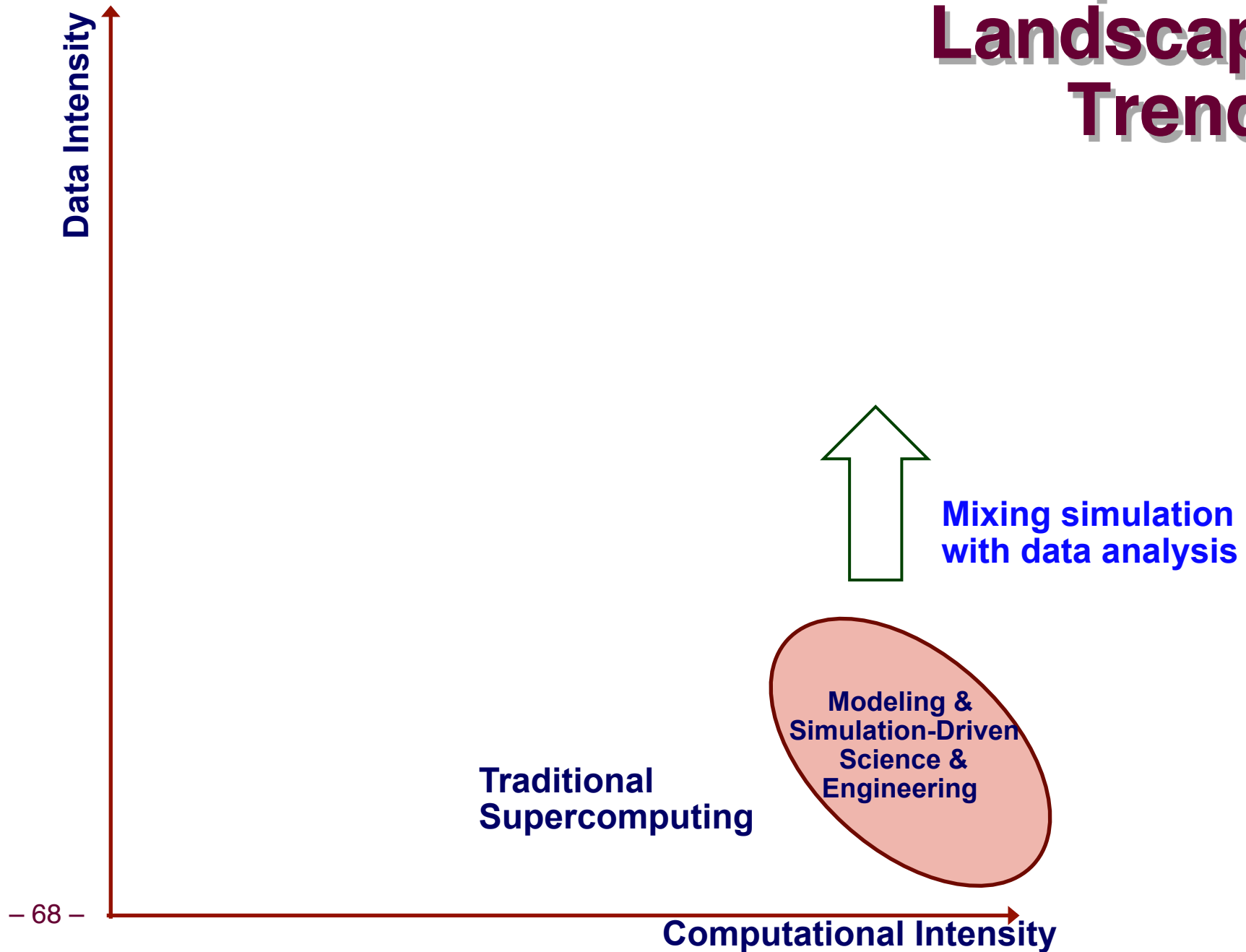
Machine Learning Startup GraphLab Gets A New Name And An \$18.5M Check

Posted Jan 8, 2015 by [Jonathan Shieber \(@jshieber\)](#)

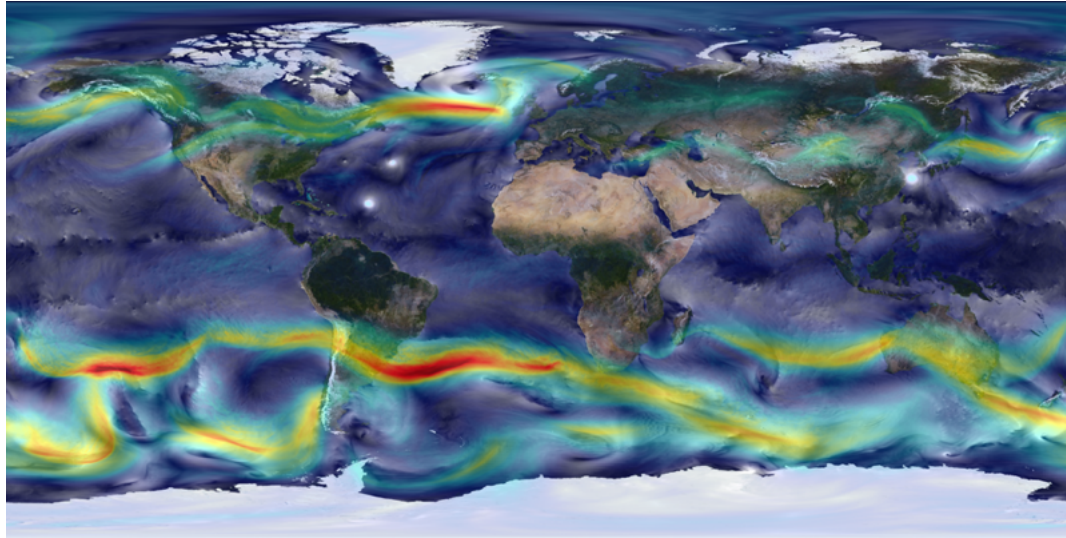
■ GraphLab

- Started as project at CMU by Carlos Guestrin
- Environment for describing machine-learning algorithms
 - Sparse matrix structure described by graph
 - Computation based on updating of node values

Computing Landscape Trends



Combining Simulation with Real Data



■ Limitations

- Simulation alone: Hard to know if model is correct
- Data alone: Hard to understand causality & “what if”

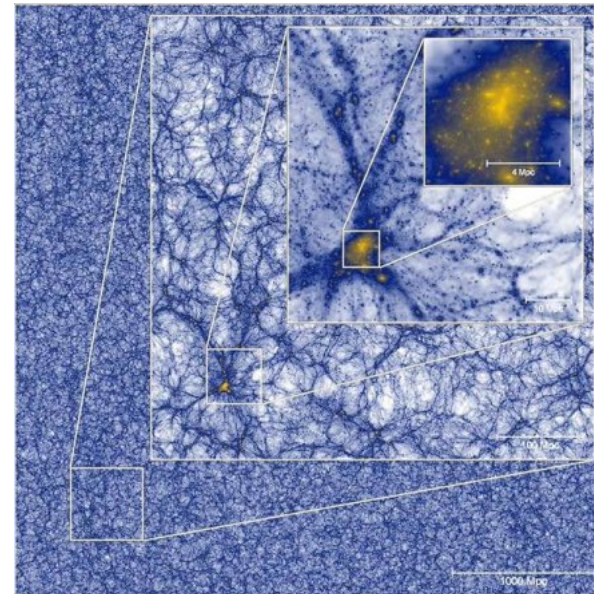
■ Combination

- Check and adjust model during simulation

Real-Time Analytics

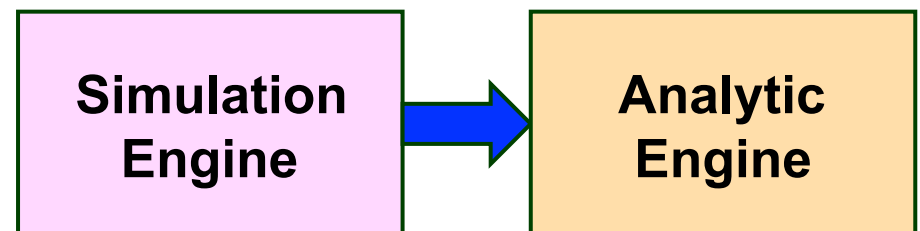
■ Millenium XXL Simulation (2010)

- 3×10^9 particles
- Simulation run of 9.3 days on 12,228 cores
- 700TB total data generated
 - Save at only 4 time points
 - 70 TB
- Large-scale simulations generate large data sets

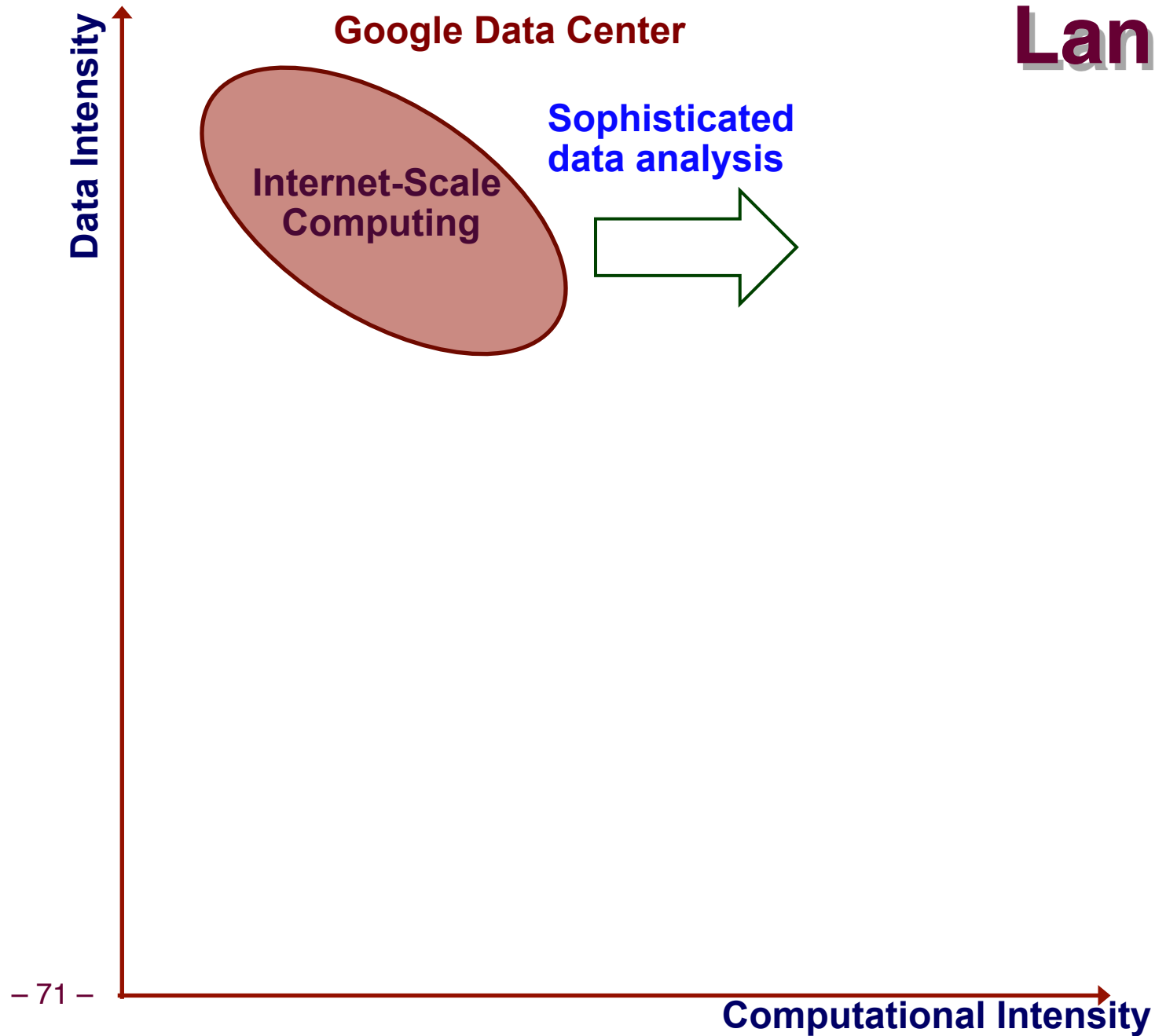


■ What If?

- Could perform data analysis while simulation is running

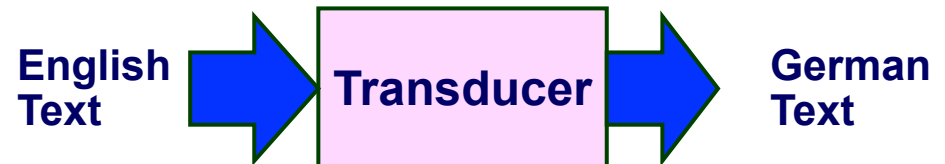
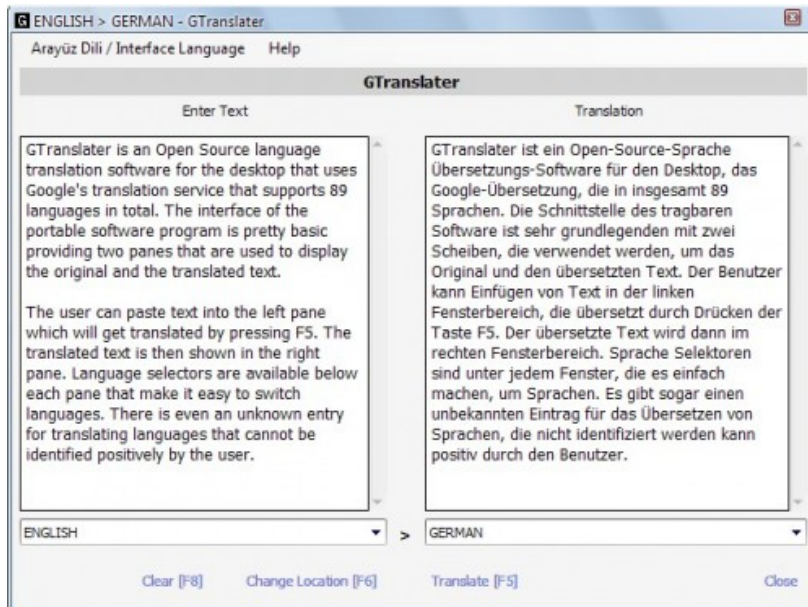


Computing Landscape Trends



Example Analytic Applications

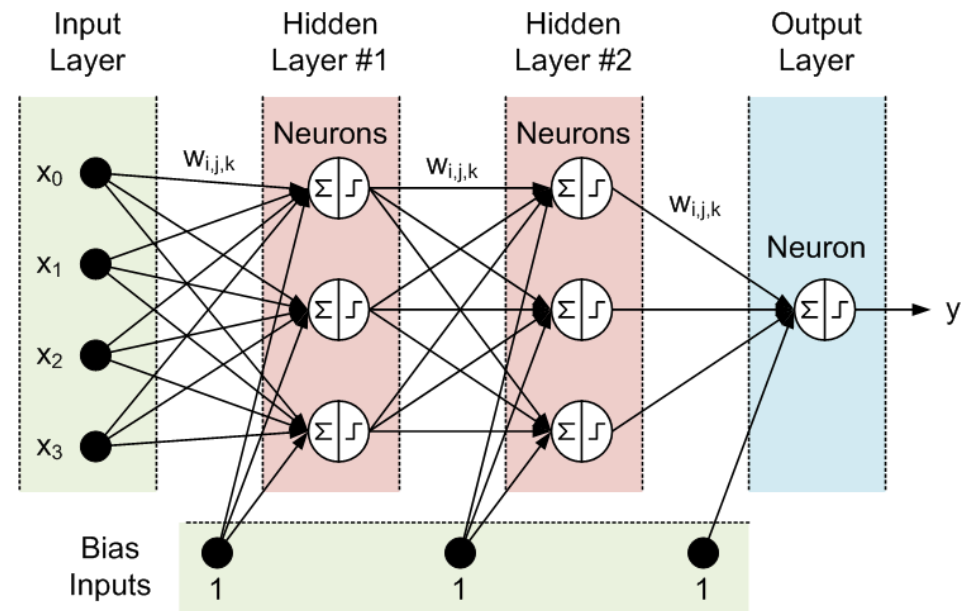
Microsoft Project Adam



Data Analysis with Deep Neural Networks

Task:

- Compute classification of set of input signals



Training

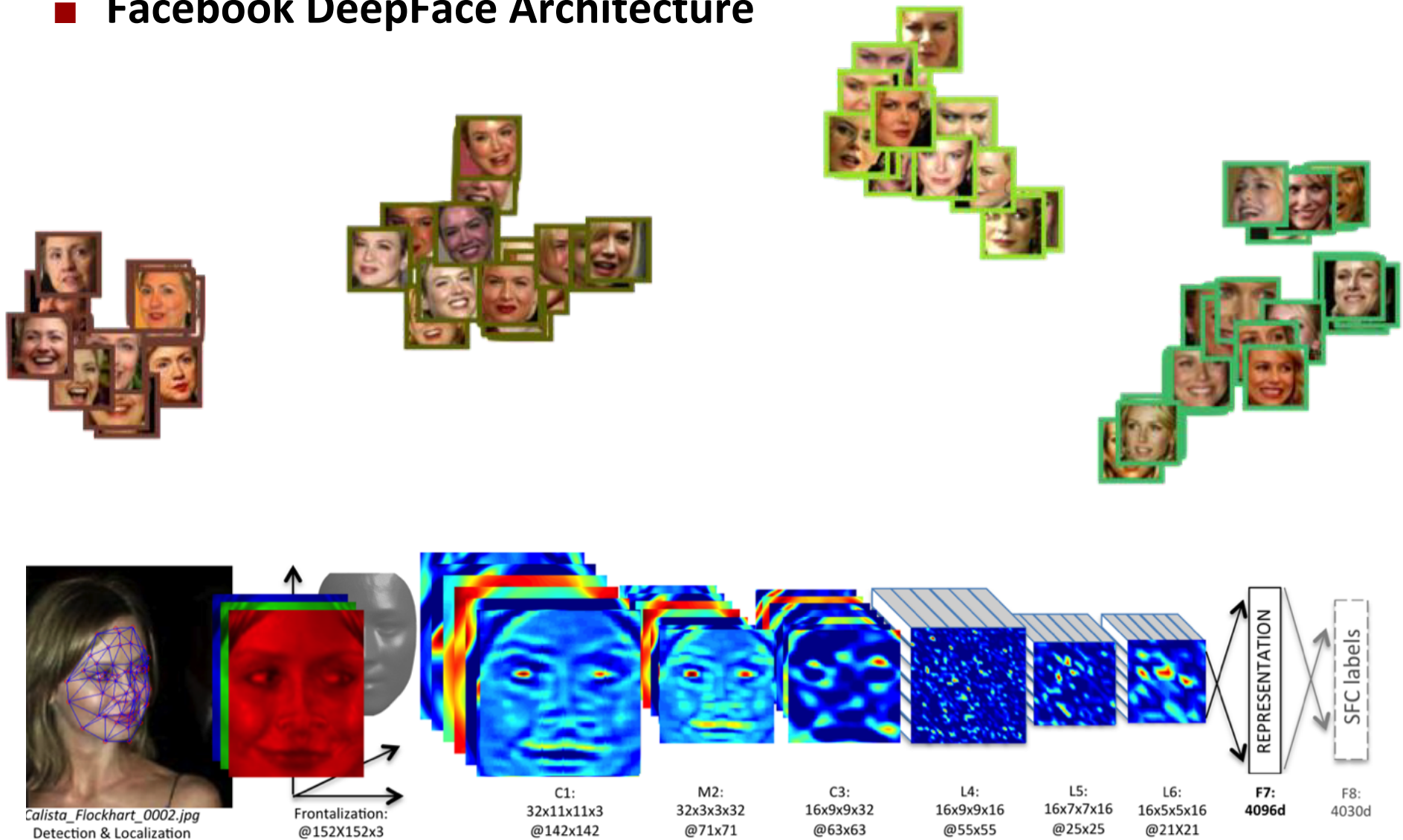
- Use many training samples of form input / desired output
- Compute weights that minimize classification error

Operation

- Propagate signals from input to output

DNN Application Example

■ Facebook DeepFace Architecture



Training DNNs



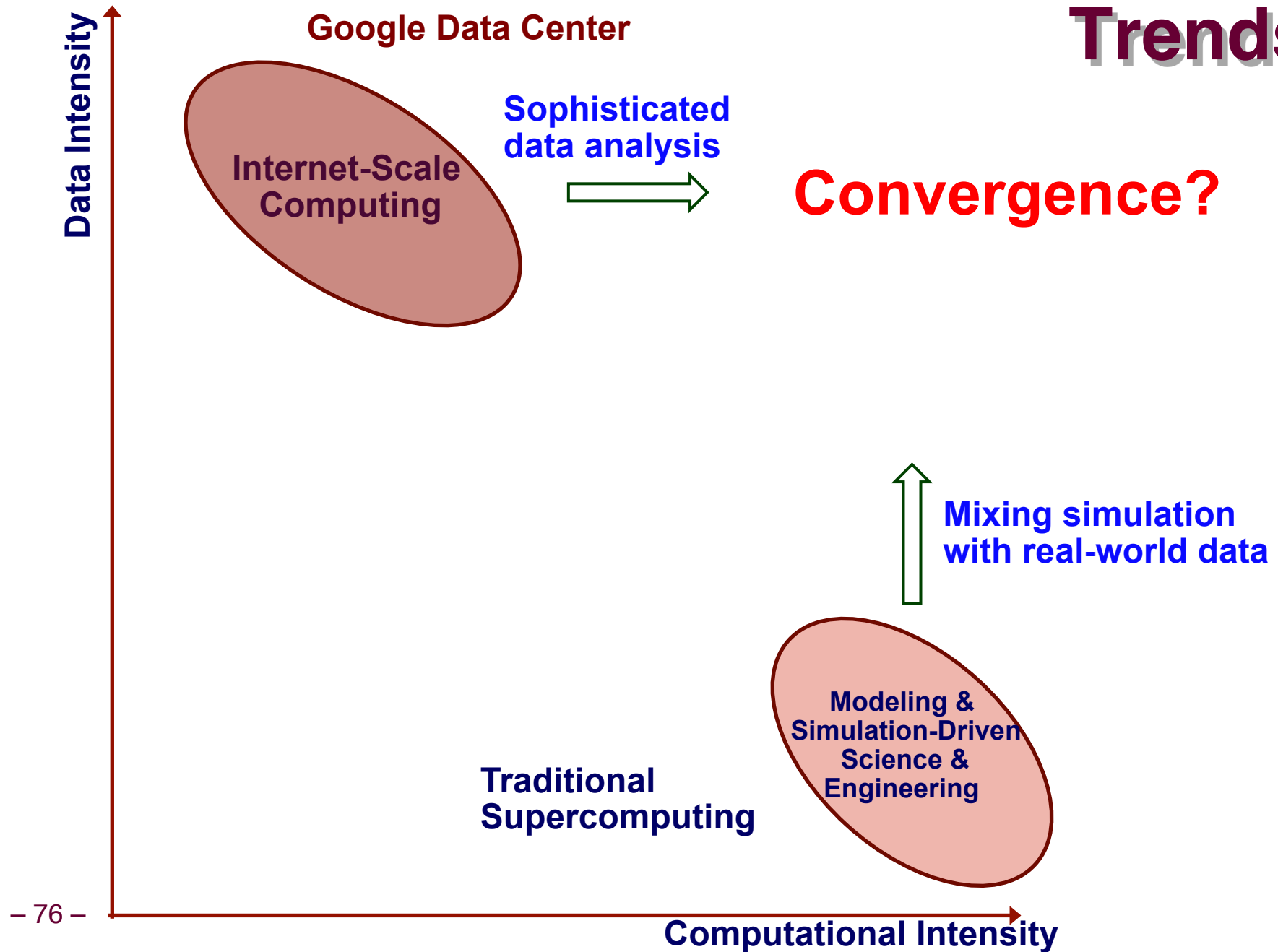
Characteristics

- Iterative numerical algorithm
- Regular data organization

Project Adam Training

- 2B connections
- 15M images
- 62 machines
- 10 days

Trends



Challenges for Convergence

Supercomputers

Data Center Clusters

Hardware

- | | |
|-----------------------------|--------------------------|
| ■ Customized | ■ Consumer grade |
| ■ Optimized for reliability | ■ Optimized for low cost |

Run-Time System

- | | |
|---------------------|------------------------|
| ■ Source of “noise” | ■ Provides reliability |
| ■ Static scheduling | ■ Dynamic allocation |

Application Programming

- | | |
|--------------------------------------|----------------------------------|
| ■ Low-level, processor-centric model | ■ High level, data-centric model |
|--------------------------------------|----------------------------------|

Summary: Computation/Data Convergence

- **Two Important Classes of Large-Scale Computing**
 - Computationally intensive supercomputing
 - Data intensive processing
 - Internet companies + many other applications
- **Followed Different Evolutionary Paths**
 - Supercomputers: Get maximum performance from available hardware
 - Data center clusters: Maximize cost/performance over variety of data-centric tasks
 - Yielded different approaches to hardware, runtime systems, and application programming
- **A Convergence Would Have Important Benefits**
 - Computational *and* data-intensive applications
 - But, not clear how to do it