Beyond Moore's Law

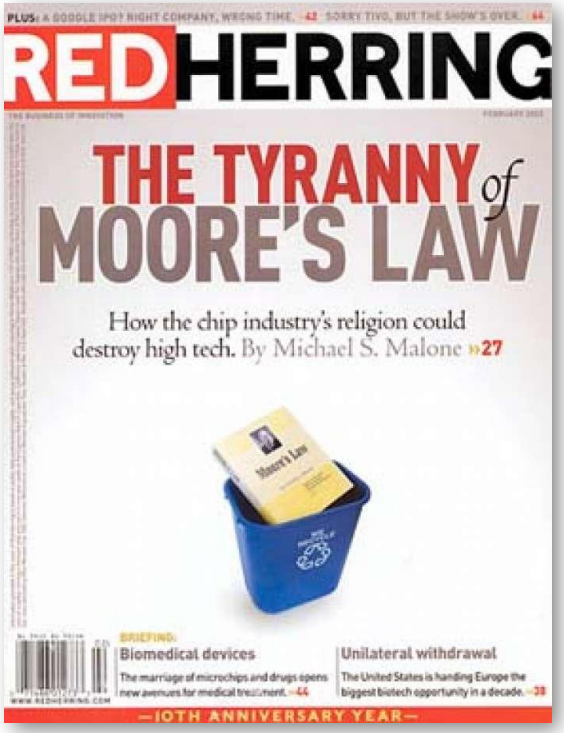
What happens when the greatest truth of our lives ends?

Michael Gat

Socal Python, October 2018

“The greatest truth of our lives”

- For most of our lives, computer hardware advances have followed “Moore’s Law”
- Invisible but relentless
- So pervasive, we take it for granted



A brief history of computer chips

- 1958-9: Invented (Kilby, Lehovec, Noyce, Hoerni)
- 1960: First produced (Fairchild Semiconductor, Jay Last)
- 1965: Moore’s Law (revised 1975)

Moore's Law

- Number of transistors on an integrated circuit (chip) will double every 24 months, for the chip with the lowest cost per transistor.
  - Originally was 12 months, revised 1975
  - Even Gordon Moore had no idea where "18 months" came from
- Cost is part of the "law!"
- Effectively: "double the transistors at the same cost every two years."

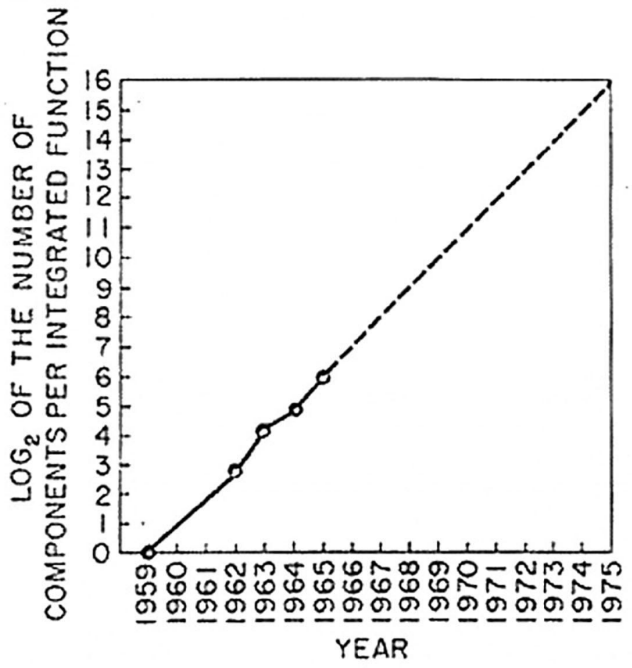


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

A brief history of computer chips

- 1958-9: Invented (Kilby, Lehovec, Noyce, Hoerni)
- 1960: First produced (Fairchild Semiconductor, Jay Last)
- 1965: Moore’s Law (revised 1975)
- 1989: First million-transistor chip
- 2005: First billion-transistor chip
- 2017:
  - 6-7 billion in “mainstream” microprocessors
  - 19 billion in extreme-performance/cost microprocessors*

*But not at lowest possible cost, so not really following “the law”

A self-fulfilling prophecy

- An entire industry scheduled around it
- Related industries (software) trailed along
- “The hardware-software spiral” as business strategy
- CPU and memory-hogging practices were deemed to be OK and even desirable!
  - It meant customers always needed the next generation
  - Ka-ching!

Result: We don't know how to code well

- You *should* learn in school... but...
- CS Professors are as captive to big software companies as business school "professors" are to Wall Street
- Schools increasingly have access to the "latest and greatest," so nobody has to struggle with limits
- Students aren't stupid
  - They know what they need to know to get a job

Bad programmers with A+ degrees

- Maybe taught all the things they “should” know
  - Given the varied paths into programming these days, maybe not
- In the absence of need, forgotten after the final project
- Might not know what bad code is costing the world
- At this point, the managers don’t know either because nobody’s really been forced to learn in years

There’s a looming problem

- Moore’s Law no longer holds
  - Intel admit it
  - nVidia admit it
  - International Tech Roadmap for Semiconductors dropped it (2016)
- Exponential advance has been the reality for so long that we take it for granted and we should not

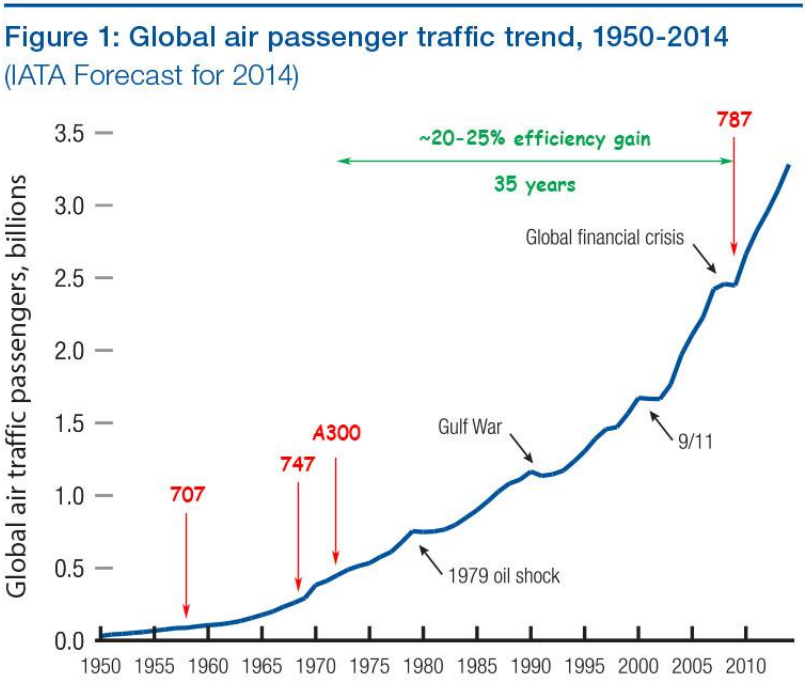
But wait...

- Quantum computing is going to save us!
- Distributed computing is going to save us!
- GPUs are still going strong!
- What about those singularity people?
- Aren’t the machines going to take over?
- Everybody is still saying things are awesome and the pace of change is still rapid!
- *Mostly the same argument...*

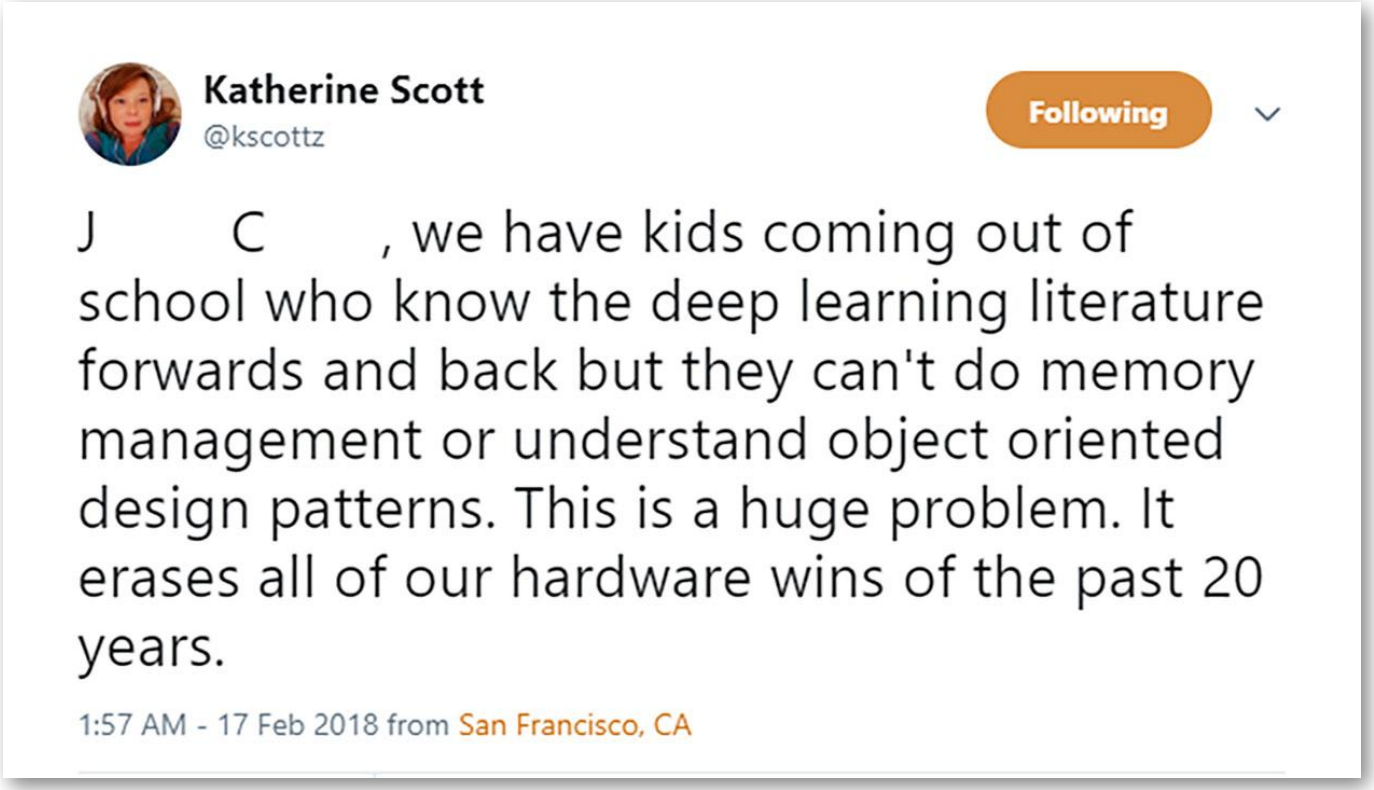
We've seen this before

"Jet age"

- The most change to most people's lives was after the tech stabilized
- It was increased reach of stable tech that made for societal change
- Max hype just as maturity happened!
- Instead we got a cramped flying bus
- Lots of engineers never adjusted
- Smart people knew it at the time



Smart people know this today too



Katherine Scott
@kscottz

Following

J C , we have kids coming out of school who know the deep learning literature forwards and back but they can't do memory management or understand object oriented design patterns. This is a huge problem. It erases all of our hardware wins of the past 20 years.

1:57 AM - 17 Feb 2018 from San Francisco, CA

We need to be better software engineers

- The basics of computer science are no longer optional
  - Object oriented patterns
  - Memory management
  - Optimizing CPU use
  - Big-O (do you know what it is?)
- Knowing another language is probably not optional
  - Python is amazing and the best
  - But not best for everything
  - Those other cases will become a big deal

Understand the trade-offs!

- Engineering is where science and economics intersect
- It is about trade-offs!
- We need to understand the trade-offs at every level
- Critical in “pay by the cpu-second” cloud environments
- If you got into it for the “art” or even just the “science” you might need to re-evaluate.

What should we learn:

- Basic math concepts
- Data Structures
- Fundamental Algorithms
- Object-oriented patterns
- Memory and I/O management
- Databases
- *Another language!*

Basic concepts

- Induction
- Integer functions
- Permutations/Factorials
- Binomials, Harmonics, Fibonaccis
- Generating functions
- Big O-notation and why it matters!
- What are the trade-offs?

Data Structures

- Stacks
- Arrays, lists, records (tuples)
- Trees
  - Binary, b-trees, heaps (there are others)
- Hash lists and tables
- Graphs
- No, not the built-in ones, build them yourself!
- What are the trade-offs?

Algorithms

- Search
- Sort
- Insert
- Update
- Delete
- Using all the data structure types
- What are the trade-offs?

Object oriented patterns

- Objects
- Methods
- Classes
- Encapsulation
- Inheritance
- Polymorphism
- When to go OO, and not
- What are the trade-offs?

Another language

- Python is the best, of course...
- As an interpreted language, it has its drawbacks
  - We trade off ease of development for efficiency
- When hardware is critical, greater efficiency is desirable
  - Usually Java or C++
- Put resource-critical code in a more efficient language
  - Can still be called from Python!
- What are the trade-offs?

Other

- Statistics and Linear Algebra!
- Memory use and abuse
- I/O
- Databases
- What are the trade-offs in everything we do?

Some Resources

- Art of Computer Programming (vol 1-3)
  - Don’t buy, but read the wikipedia entry and the index. There are plenty of online resources to get you through the main topics
- Design Patterns: Elements of Reusable OO Software
  - This one’s an easier read, but again, lots of free online resources cover the material with more up to-date code
- Downey’s “Think” series (O’Reilly)
  - Have the great benefit of being in Python
- More: http://www.michaelgat.com/MooresLaw

Be prepared for

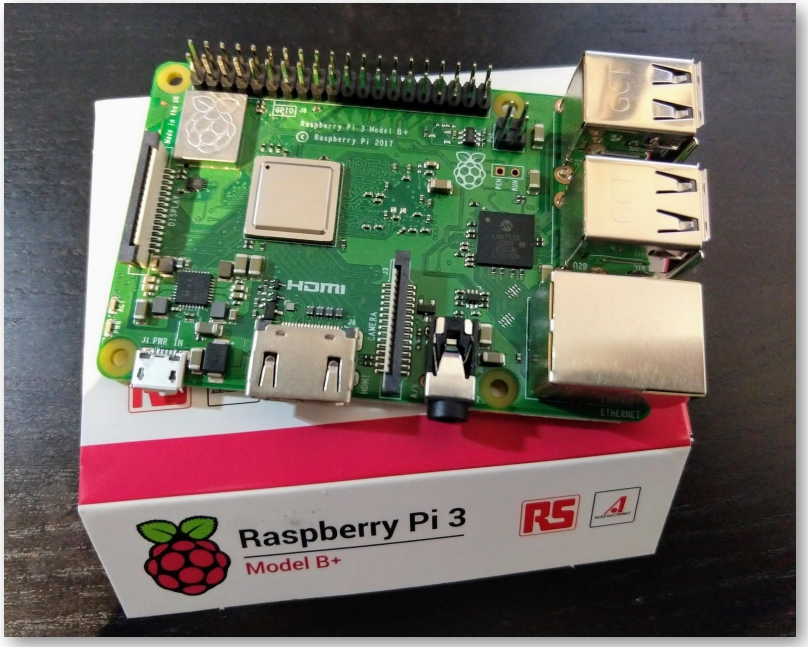
- Consolidation of environments, languages, frameworks
  - Some won’t make it
  - Some features will be adopted by survivors
  - Python is a good place to be
- More cloud
  - The efficiency of standard solutions vs. cost of bespoke niche cases
- All while applications and ideas continue to multiply
- Some trade-offs just won’t make sense

But also, a modest proposal

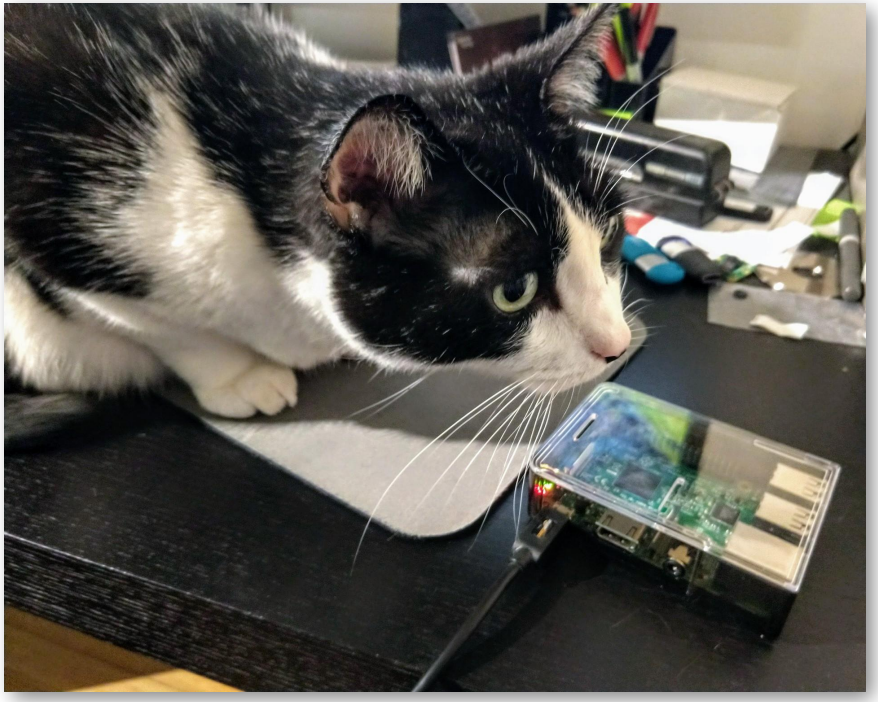
“Crappy Computers!”

OK, maybe not *that* crappy...

But how about one of these?



Pause for obligatory cat photo

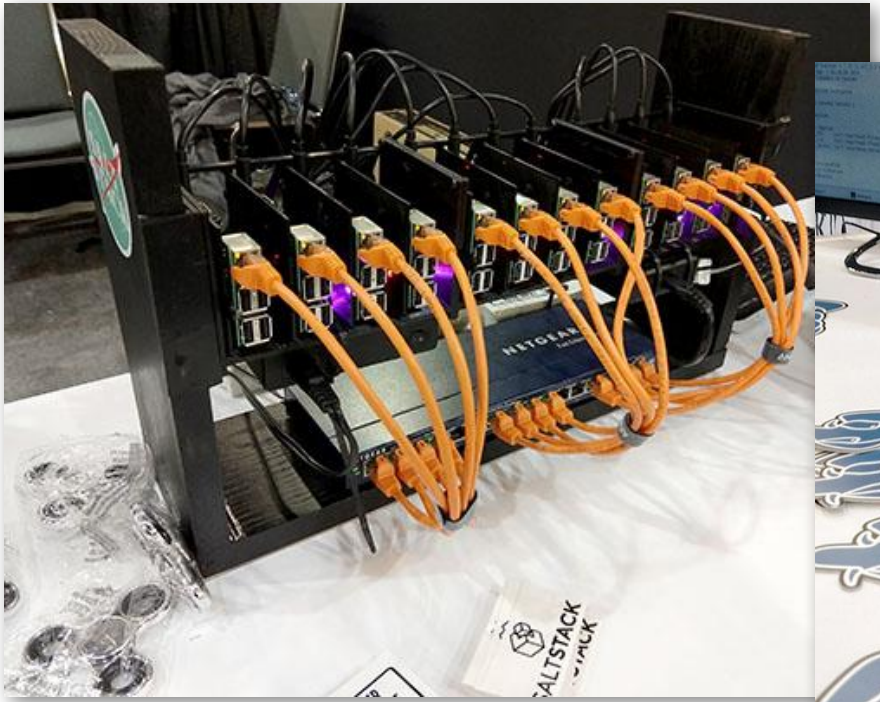


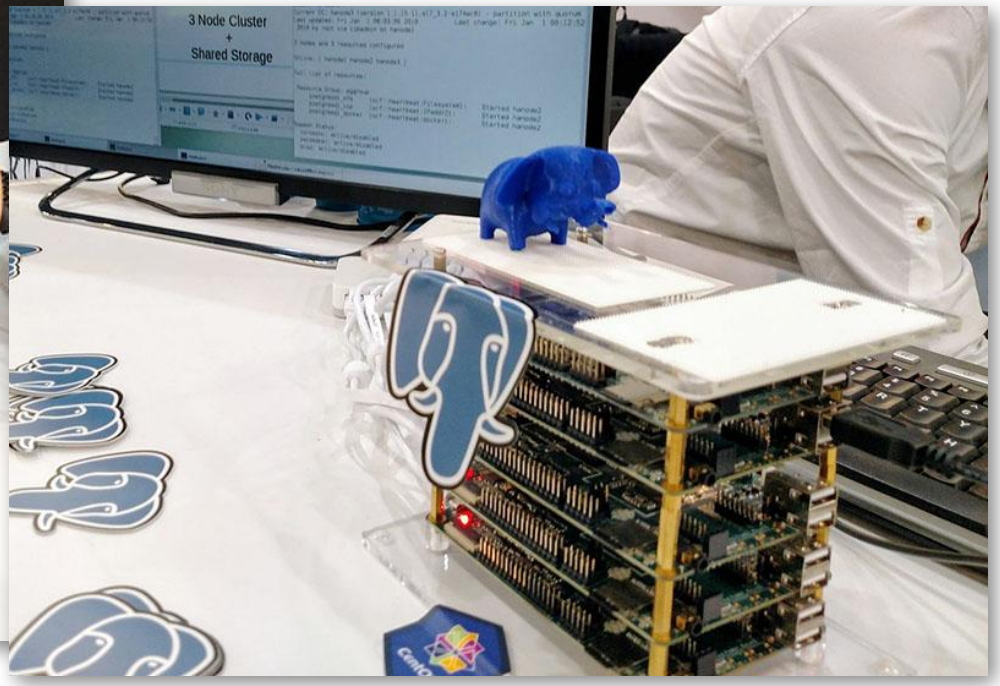
His name is Flash and he loves Raspberry Pis! (His counterpart RIP was “Floppy.”)

Great tools for forcing yourself to learn

- Limited memory, CPU, and storage bandwidth for $35
- Save and push yourself more with a Pi Zero for $10
- Do things better because you get sick of waiting!
  - And sometimes because they just blow up
- Or, really, just use that ancient laptop in the closet
  - Or Pine64, BeagleBoard, NanoPC/NanoPi, adafruit Circuit Playground
  - Anything that forces you to confront limits!
- You can get through all the CS core material on RPi
  - A lot of people could do their entire jobs on one

Very smart people are already going there





In summary

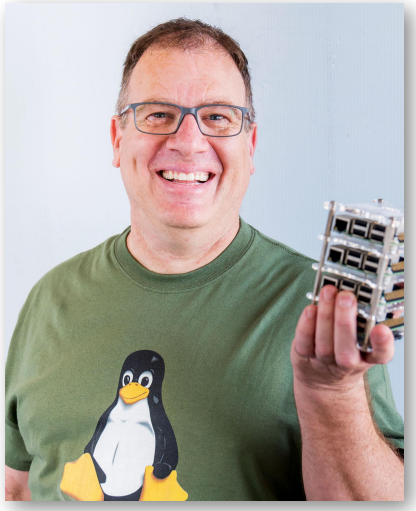
- We don’t have Moore’s Law to save us from sloppy practices anymore
- This is a generational paradigm shift
  - We don’t know where it’s going to take us, but the recent patterns won’t work
- We need to go back to basics
  - We don’t want to be “coal miners” looking for the old days
- We need to be engineers, not artists

The future is unknown, but different

- The ends of long-term trends are hard to predict
- We must be cautious and opportunistic
- Technology history suggests that efficiency/economics may become the driving force in computer tech

“The old stuff gets broken faster than the new stuff is put in its place. The importance of any given experiment isn’t apparent at the moment it appears; big changes stall, small changes spread.”
~ Clay Shirky, 2009

Thank You!



Michael Gat
@michaelgat
michaelgat.com

Slides at github.com/michaelgat/Presentations

Questions!

Not:

- Calling bullshit
- A long story
- Your resume

Floppy RIP

Floppy RIP