A call to Arm(s)

Edward Vielmetti for Orchestructure

Outline of talk

History of Arm-based systems

The recent past (or, how I got here)

The state of the art (or, what I’m working on now)

A look at the future

Change comes from below

Mainframes (IBM) displaced by minicomputers (DEC)

Minicomputers (DEC) displaced by workstations (Sun)

Workstations (Sun) displaced by PCs (Intel)

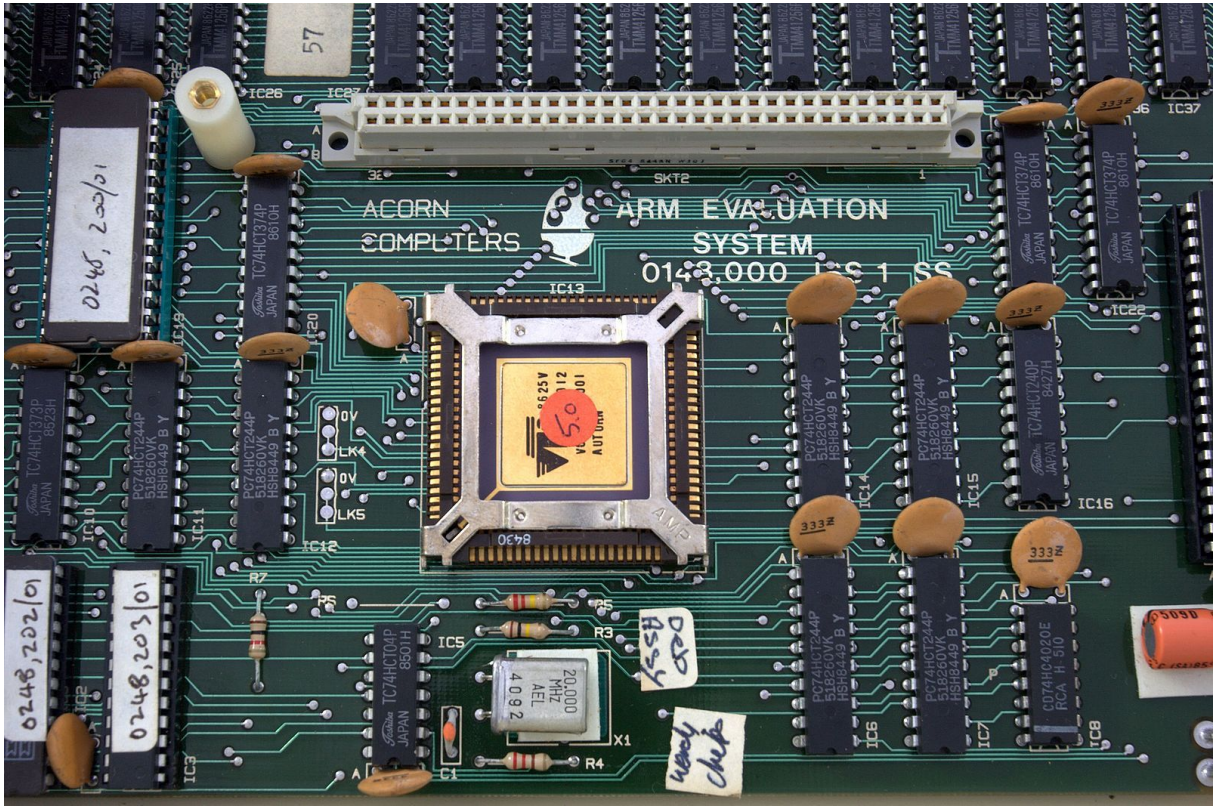
PCs (Intel) displaced by cell phones (Arm)

Why? More popular products with higher run rate lead to larger production runs, which in turn leads to more rapid advancement in production technology and better price/performance.

History of Arm, 1985

First Arm evaluation system: Steve Furber and Sophie Wilson are engineers. Project started in 1983, finished 18 months later.

25,000 transistors, low power 32-bit processor.



When reusing photograph under CC licence please credit photograph to "Peter Howkins".

Arm, 1992

Acorn Archimedes A3020, 1992

British design used in UK schools.

ARM250 chip, 32-bit, 12 Mhz. Runs RiscOS.

Photo: Vintage CPU

History of Arm: Apple Newton (1993)

Arm in the year 2000: first “smart phone”

“The ARM7TDMI core, first developed for digital cellular handsets, is ARM’s most popular offering for the Wireless market and today can be found in millions of handsets worldwide, spanning such wireless standards as GSM, CDMAone, D-AMPS and PDC.” - ARM annual report, 2000

ARM core designs licensed for manufacture by many semiconductor companies.

History of Arm: iPhone 1 (2007)

ARM11 CPU

90nm process

Samsung 32-bit CPU at 412 Mhz

2005 Arm mobile market share 98%

History of Arm: Raspberry Pi (2012)

Pi 1: 700 Mhz single-core Broadcom ARMv6

Pi 3 B+: 1.4 Ghz quad-core Broadcom ARMv8

19 million sales by March 2018

Cell phone-derived parts on cheap ($35) computer designed to be suitable for education.

Surprisingly useful for industrial and general-purpose computing use.

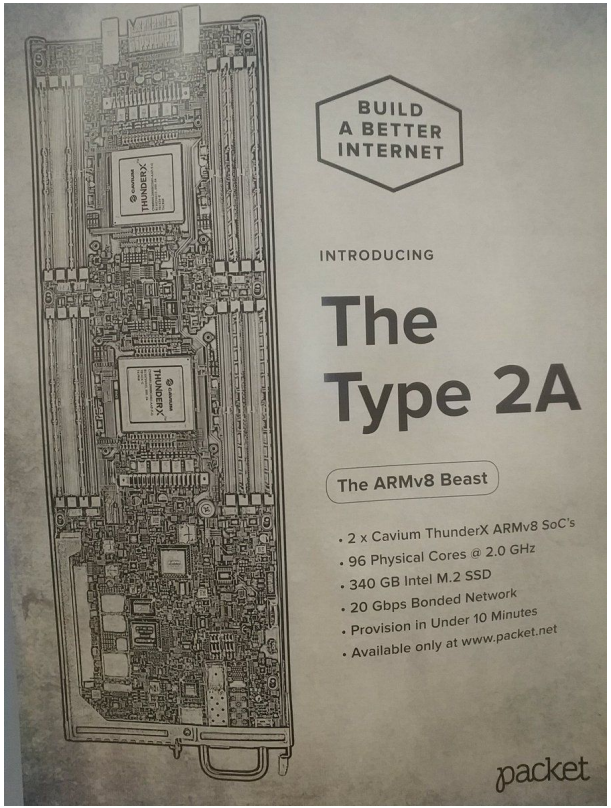
The recent past: Cavium ThunderX at Packet (2016)

Compare “Type 2A” with Raspberry Pi:

CPU: 96 cores at 2 Ghz (vs 4 cores at 1.4 Ghz)
Memory: 128 GB (vs 1 GB)
Network: 20 Gbps (vs 300 Mbps)

This is how I got started in Arm in the data center.

Fortunately: I skipped a previous generation of data center grade systems that hadn’t been as well engineered.



arm64 software, 2016 vs 2018

OS: Ubuntu 16.04

Software: Many things mostly work if you compile them yourself

Bugs: Small set of bugs that appear repeatedly

Performance: So much faster than a Raspberry Pi; we love having 96 cores; compiles seem fast, except for the link step.

OS: Ubuntu, Debian, CentOS, Fedora, RHEL*, SUSE, OpenSUSE, FreeBSD

Software: Pre-built binary distributions for many (most?) software stacks; generally works; performance issues known.

Bugs: Mostly squashed.

Performance: On optimized codes, compare with Intel Xeon at ⅔ the power consumption. Not all compilers are 100% optimized yet. “Low hanging fruit” to do so if you know Arm asm.

What do people use these systems for?

Build machines for IoT and embedded codes - native code compilation is about 5x as fast as emulated code. (Resin)

Build machines for languages and libraries to test for and optimize for the next generation of hardware (Go, LLVM, Node.JS, many more)

Edge workloads - web-scale cloud workloads, one thread per remote system, scale out horizontally. (Cloudflare)

235 watt Xeon vs 150 watt Centriq (Cloudflare)

Compatibility, the hit list of most important

Boot firmware and network drivers.

Operating systems and distributions - get lots of software ported all at once, ensure system stability.

Languages and libraries - start by getting everything to pass all its tests, and then start to work on build system performance and the performance of generated code.

Containers and orchestration. Work on and worry about packaging so that applications run easily.

Scientific computing and big data - not our direct customers, but stay informed.

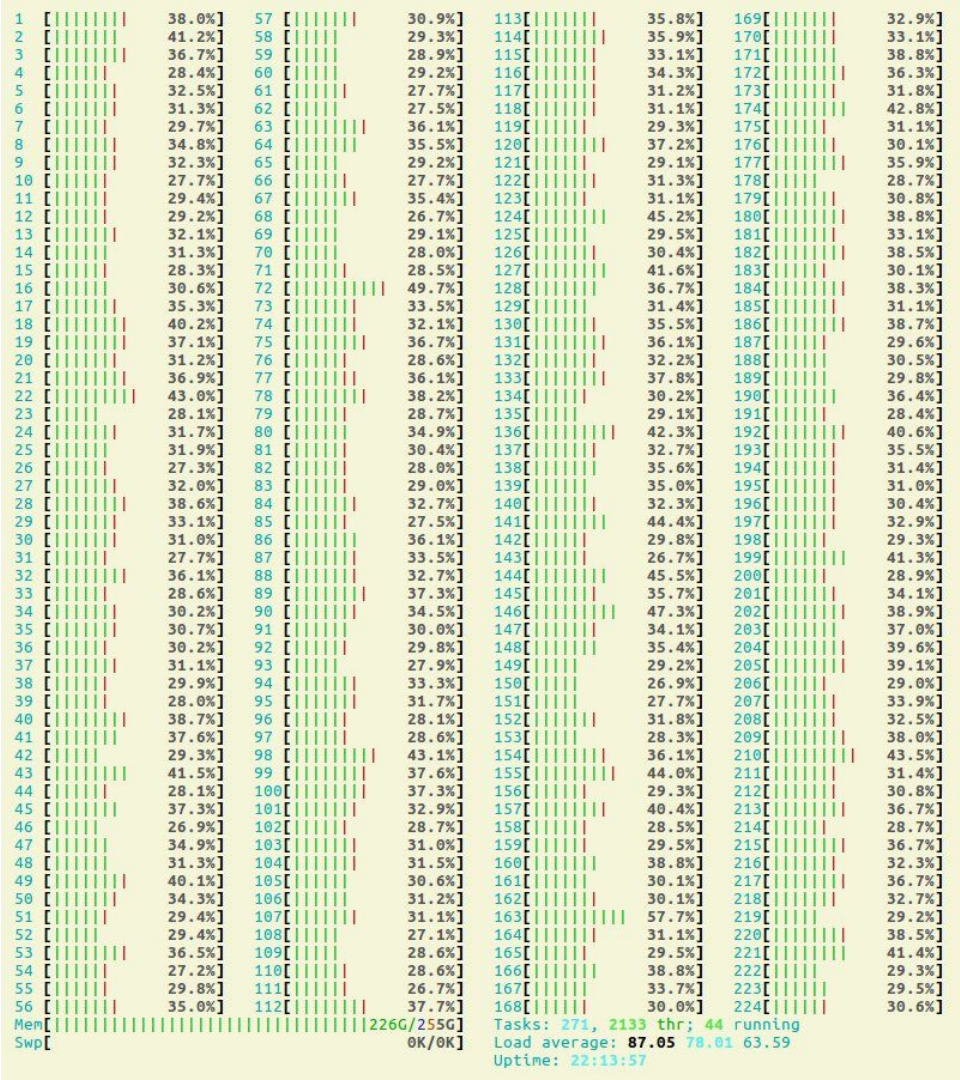
Web-scale data management - make no small designs on how many different systems around the world will be running in a single distributed system.

htop on 224 thread TX2

Once you have this many cores at your disposal, you never ever want to go back.

With 224 threads, a load average of 87 is nearly idle.

There are lots of issues around parallelism that never get touched in other system designs that rely on fewer, faster cores.



The story of Go (golang)

Go 1.7: bug in Getpagesize() results in crash in Docker file systems when you layer your filesystems deeply

Go 1.8: Getpagesize() fixed, but no binary distributions; have to bootstrap the software yourself.

Go 1.9: binary release for arm64; performance still slow on crypto routines.

Go 1.10: Improvements in manycore compile times. Cloudflare reports Arm vs Intel benchmarks.

Go 1.11: teams from Arm, Qualcomm, Cloudflare, others start work on performance on arm64. 100s of patches. Release due summer 2018.

Never underestimate the power of a good, well-researched, comparative benchmark to light a fire under a team.

The state of the art: hardware

What's here:

New data center grade systems on the near horizon (Q2, Q3 2018) from Cavium, Qualcomm, Ampere.

Steady stream of sub-$100 single-board computers: Rockchip, Qualcomm, NXP, Beaglebone, Marvell, Pineboard, Raspberry Pi.

Windows on Arm laptop from Qualcomm.

Lots of bolted together arm64 clusters running Docker and Kubernetes.

What's missing:

Developer workstation - new 24-core Synquacer / Socionext system is closest ($1250 kit). "The year of Linux on the desktop".

Single board computers with >4G of RAM.

The state of the art: software

Mostly:

Just works when you compile it yourself.

Usually:

Binary distributions available from developer or from distro.

Sometimes:

Docker multi-architecture containers.

What's missing:

5% of code that's broken for whatever reason - x86 dependencies, lack of maintainer interest, very old build systems.

Optimizations that use 100% of the arm64 assembly language hardware speedups. Some compilers (Go, notably) are waiting for optimized code to land in a release.

Turnkey CI/CD systems (like Travis, Circle etc) that can handle arm64 builds for you.

The curse of embedded systems

“The internet of shit”

Impossibly cheap chips, connected to the Internet, doing pointless things with insecure software and unmanaged firmware.

You don't care about flexibility, you don't care about adaptability, you don't care so much about reuse.

Disposable or unrepairable and vulnerable to attacks.

Photo: @dgouldin via @internetofshit

Challenges: parallel architectures

Ampere: 32 cores? (not fielded yet)

Hisilicon: 32 cores x2 (64 total)

Qualcomm: 40-48 cores, single socket

Cavium ThunderX: 48 core x2 (96 total)

Cavium ThunderX2: 54 core, 4-way thread (216)

Not enough people have used manycore systems enough for all of the performance improvement opportunities to be shaken loose.

Amdahl's Law: Parallel speedup is limited by the total time needed for the sequential (serial) part of the program.

Example: Highly parallel compiler with a relatively slow, single-threaded linker. Overall build times dominated by link times, even at make -j 96.

Challenges: non-uniform memory architectures

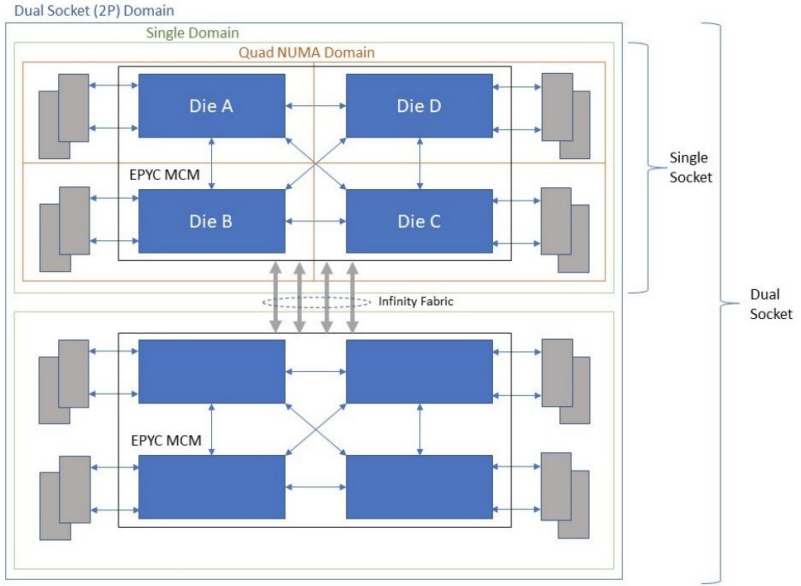
Not all memory is created equal!

Manycore processors (here we have AMD EPYC) have complex on-chip topologies, to extract the best performance out of available memory bandwidth to the CPU.

Some memory is closer than others!

Schedulers need to be aware of memory localities to wring the best performance out of parallel systems. It's possible to be memory IO bound on performance.

Figure 3: EPYC NUMA Domains



Sources: AMD & TIRIAS Research

Challenges: build systems

Lots of code has been built by increasingly complex CI and build systems. Over the past years most of that code has been written and built for x86 systems.

Emulation of Arm on x86 is relatively slow (about 10-20% of native speed, give or take). Good enough for small jobs, impossible for big jobs.

How do you build all the world's open source projects? Offer dedicated hardware for the task.

.

Travis CI

circleci

A look at the future

Arm architectures coming up from below:

1. Ever faster and more capable single-board computers, leveraging cell phone SoC designs.
2. Integrated machine learning and graphics processing capabilities
3. Tough slog in the data center against entrenched Intel interests; biggest wins in power-constrained environments
4. The year of Linux on the desktop is still next year, or the year after.

Alternative architectures coming up from below: royalty-free RISC-V designs to disrupt the bottom part of the market again.