

Initial Explorations of ARM Processors for Scientific Computing



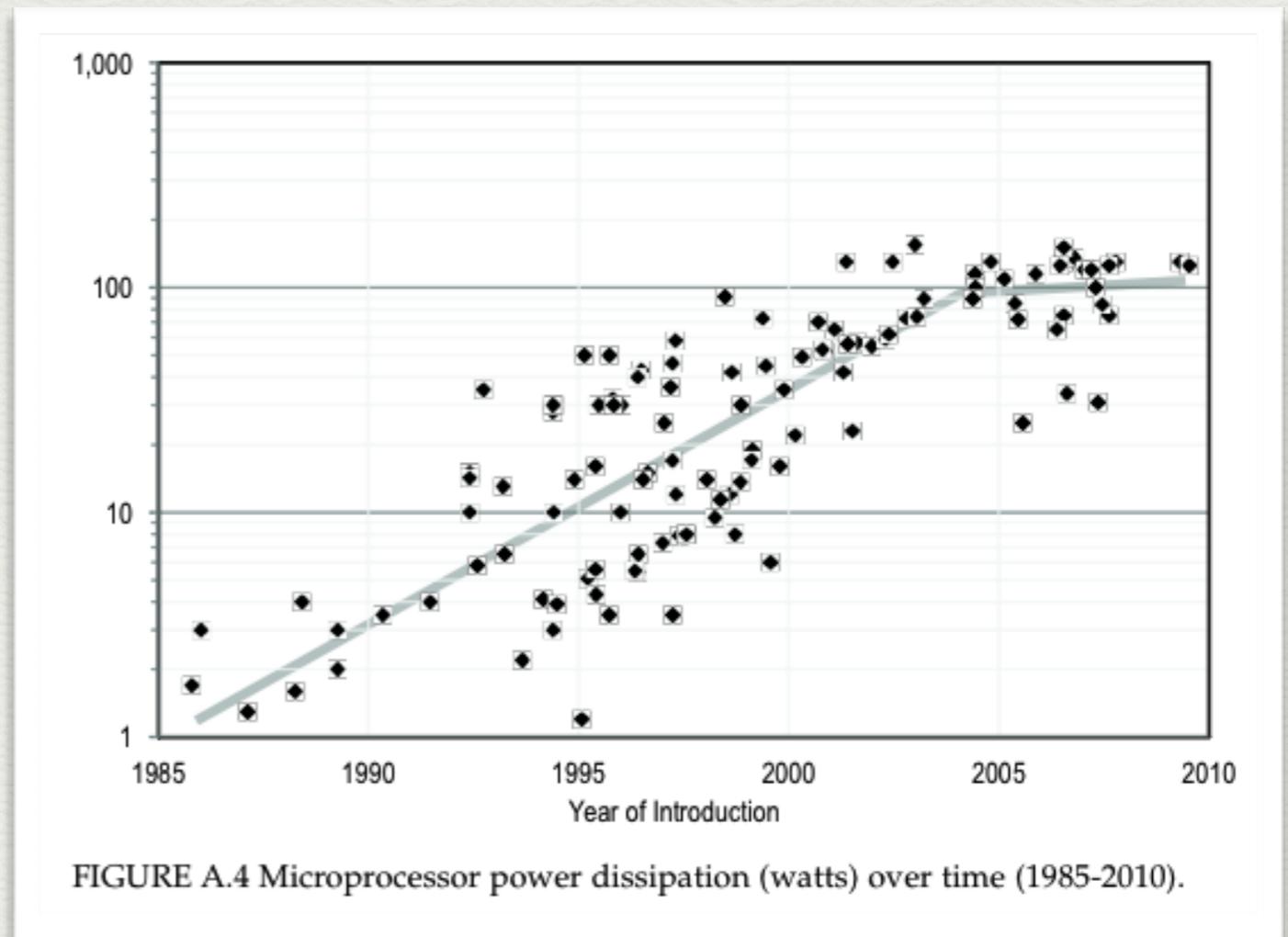
Peter Elmer - Princeton University

David Abdurachmanov - Vilnius University

Giulio Eulisse, Shahzad Muzaffar - FNAL

Power limitations for processors

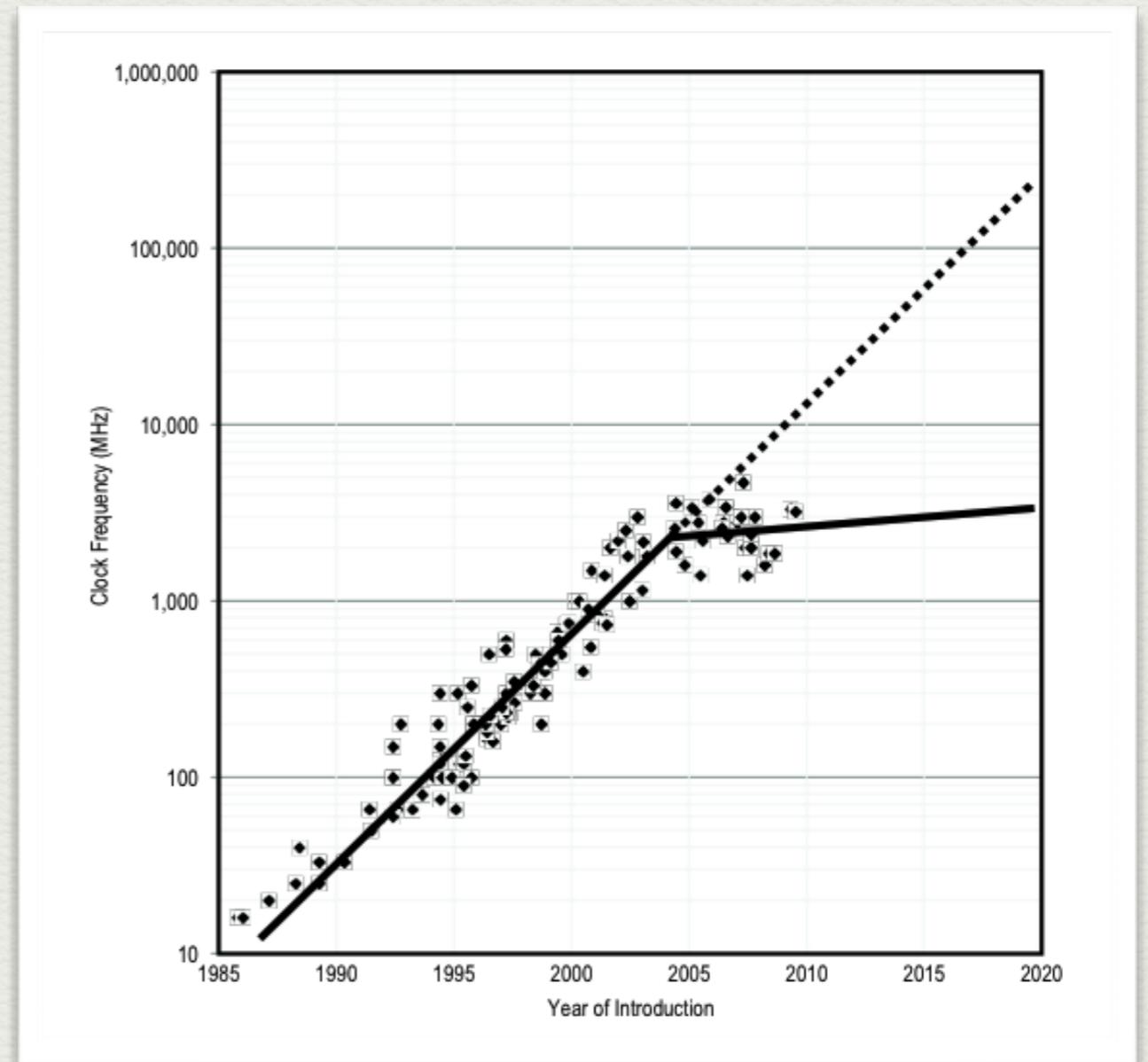
- Over the past ten years processors have hit power limitations which place significant constraints on "Moore's Law" scaling.
- The first casualty was scaling for single sequential applications, giving birth to multi-core processors.



From: "The Future of Computing Performance: Game Over or Next Level?"

The Future of Moore's Law

- Even multi-core, implemented with large "aggressive" cores is just a stop-gap. The power limitations remain. The focus is shifting to performance/watt, not just performance/price.



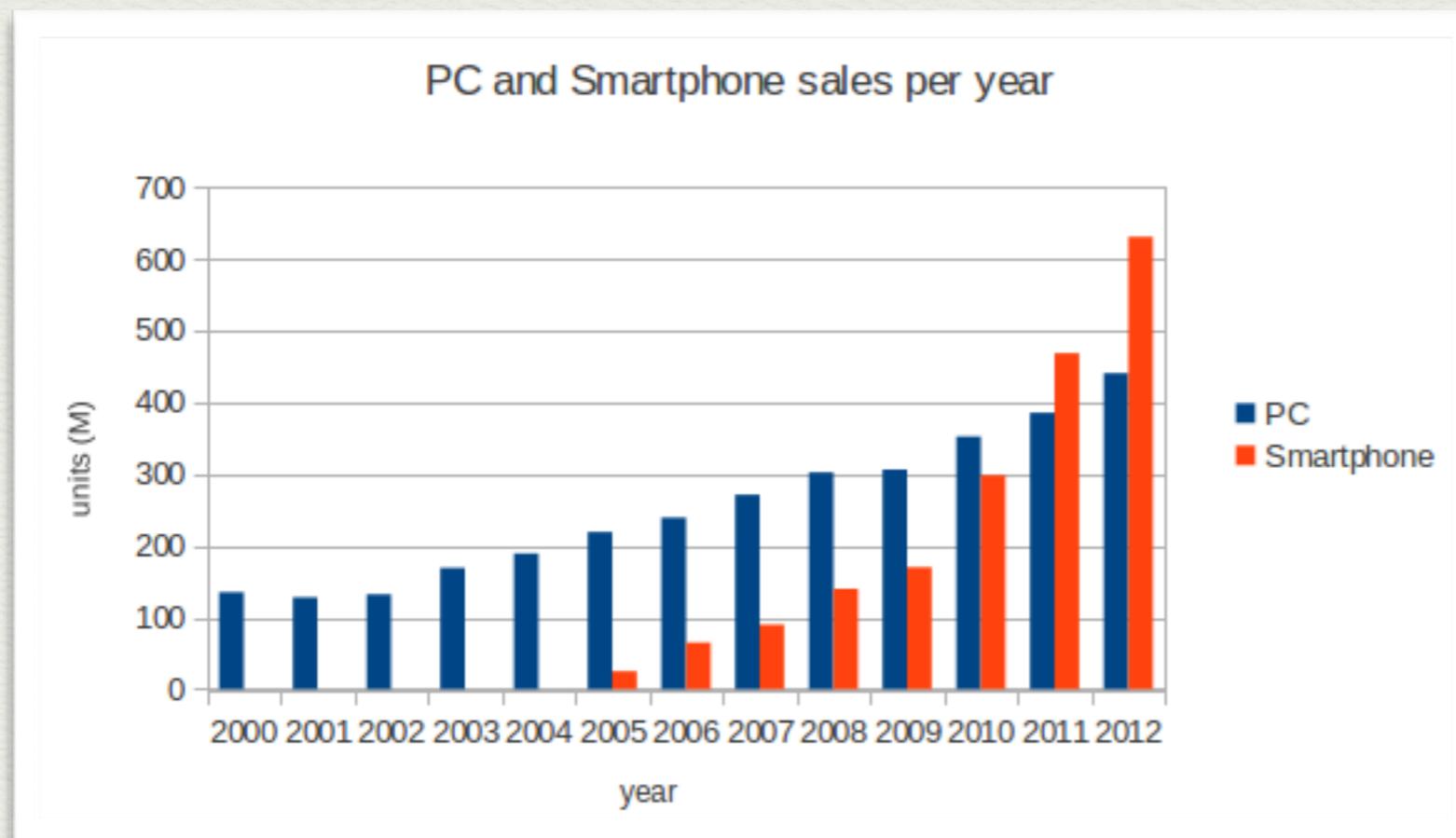
From: "The Future of Computing Performance: Game Over or Next Level?"

Power and Data Centers

- How relevant is the power use in data centers?
- For a typical 4 year server lifetime, and taking into account the full power efficiency, the power cost can already represent ~25-50% of the total cost. (Varies, but this are typical numbers.)
- In 2007 the US Environmental Protection Agency estimated the power use of servers as 1.5% of total US power output. Taking into account trends for efficiency, increased server use and capacity it is estimated that power for server use could reach 5% of the total generation in the US by 2016.
- Thus there are important economic drivers in play in the market, focused on server power use.

ARM processors

- RISC processor with a long history going back to the BBC Micro. Of interest today as the core processor used in the vast majority of mobile devices.
- Current generation ARMv7/32bit, ARMv8/64bit products expected in 2014



- Unit sales increasing dramatically in recent years (typically cost and profit/unit, however)

ARM Servers

- As power limitations have become important also in the server market, there is an opportunity for ARM to enter a market dominated currently by Intel, capitalizing on its strength in low power (high performance/watt) processors
- ARM's business model is to license their design for the processor core to others who then build the chip (typically SoC) around that.
- This flexibility is clearly interesting in the mobile device market, given the many constraints (power, size/form factor, price/feature points). Now it is becoming interesting also for the server market as they look for efficiency gains.

ARM Servers

- This has led to the introduction of ARM-based servers in recent years, such as the Boston Viridis:
 - 192 cores in a 2U rack mount, consuming <300W
 - 48 quad-core nodes (1.4GHz Cortex-A9)
 - \$20k (reported)
- servers with the new ARMv8/64bit cores, expected next year, will likely be the product that will either create (or not) sufficient market share

Goals of an ARMv7 test port

- We currently do linux/x86-64 and OSX/x86-64 builds, in the past we had linux/ia32 builds (including ia32 builds on x86-64). We have done partial ports in the past to linux/PowerPC and OSX/PowerPC. Each port usually flags or flushes out some number of problems both in the generality of the build system and in the software itself.
- Use an ARMv7/32bit port as a stepping stone to an ARMv8/64bit, hopefully resolving some of the eventual problems we will see.
- The capability to run applications on "small" cores will also provide an interesting initial environment for general performance studies. We expect that such "small cores", whether x86 or ARM, are part of the future in any case.
- High core counts even on 32bit ARM servers can also be useful for testing scalability of the multicore-aware framework CMS is developing.

ODROID-U2

- Do some initial tests with a small 32bit ARMv7-A ~5W development board
- Exynos4412 Prime CPU
- 1.7GHz Cortex-A9 quad-core
- 2GB LP-DDR2 memory (512MB per core)
- eMMC, microSD, 2xUSB 2.0, Ethernet RJ-45
- \$89 (\$233 with cables, cooling fan, power adaptor, 64GB eMMC, etc.)



- Fedora 18, ARMv7-A Hard Floats, gcc 4.8, ODROID kernel
- `fc18_arm7hl_gcc480`

Building for ARM

- Early build attempts done with QEMU. Slow and buggy.
- Now we have a test board: cross compilation or native builds?
- If we eventually do have proper ARMv8/64bit servers with sufficient throughput for application use, we should be able to build natively.
- CMS has also invested over the years in optimizing its build system at many levels.
- The ODROID-U2 is actually reasonably powerful, so try a native build!

Build system

- Use the same PKGTOOLS build and packaging system we use for the standard linux and OSX x86-64 builds.
- Driven by build recipes written as rpm spec files which are used (with a slight preprocessing) and a single driver script which manages the dependencies among them.
- Heterogeneous mix of build systems encapsulated in spec files. The CMS software (CMSSW) itself is built with SCRAM.
- Successfully built rpms can be uploaded to a central apt repository to allow installation elsewhere, incl. other build machines. Builds are configured to use existing rpms from apt repo, incl. installation on local disk.

Build times on ODROID-U2

- ~4 hours mostly for gcc 4.8.0, but also a small set of basic things we need for packaging:
 - rpm, apt, zlib, ncurses, nspr, sqlite, etc.
- ~12 hours for all other "externals":
 - ROOT, Geant4, Python, Fastjet, Valgrind, gdb, boost, Qt, all generators, etc. Total of ~125 packages.
- ~25.5 hours for CMS software (CMSSW) - 3.5MSLOC of C++, plus generated ROOT dictionaries

Build Issues Encountered

- No Oracle. But by construction no standard CMS grid-capable workflows can depend on Oracle. Affects a few special things.
- Minor compilation configuration issues: -m32/-m64 don't work, x86-ish assumptions leading to attempts to use SSE/AVX
- Signedness problems for char/bit-fields (Intel signed, ARM unsigned)
- Compilation of some translation units exhausted virtual memory (mostly ROOT dictionaries: refactor...)
- Patch needed for ROOT Cintex trampoline

Build Status

- All externals build except Oracle and one online-only package
- 99% of CMSSW builds: a few remaining packages require Oracle plus a few being iterative broken/fixed as we sort out various last issues.
- All build recipes/patches available from:
 - `git://github.com/cms-sw/cmsdist.git`
 - branch "IB/CMSSW_6_2_X/fc18_armv7hl_gcc480"

CMS Integration Builds

Release cycle 6.2 -- [back to top of page](#)

day	IB	platforms	builds	RelVals	OtherTests	Q/A page
thu	CMSSW_6_2_X_2013-05-16-0200	slc5_amd64_gcc472 slc6_amd64_gcc472 slc6_amd64_gcc480 osx107_amd64_gcc472 osx108_amd64_gcc472	summary details summary details summary details summary details summary details	summary details summary details summary details	summary details summary details summary details	Q/A info Strip Charts Q/A info Strip Charts
wed	CMSSW_6_2_X_2013-05-15-0400	slc5_amd64_gcc472	summary details			Q/A info Strip Charts
wed	CMSSW_6_2_X_2013-05-15-0200	slc5_amd64_gcc472 slc6_amd64_gcc472 slc6_amd64_gcc480 osx107_amd64_gcc472 osx108_amd64_gcc472	summary details summary details summary details summary details summary details	summary details summary details summary details	summary details summary details summary details	Q/A info Strip Charts Q/A info Strip Charts
tue	CMSSW_6_2_X_2013-05-14-1400	osx108_amd64_gcc472	summary details			Q/A info Strip Charts
tue	CMSSW_6_2_X_2013-05-14-0200	slc5_amd64_gcc472 slc6_amd64_gcc480 osx107_amd64_gcc472 osx108_amd64_gcc472 fc18_armv7hl_gcc480	summary details summary details summary details summary details summary details	summary details	summary details	Q/A info Strip Charts Q/A info Strip Charts

CMS Integration Builds

Summary for CMSSW_6_2_X_2013-05-14-0200 IB on platform fc18_armv7hl_gcc480 -- [Back to IB portal](#)

error type	# of packages	total # of errors
dictError	0	0
compError	15	370
linkError	132	231
pythonError	0	0
compWarning	17	29
dwnlError	0	0
miscError	11	35
ignoreWarning	3	7
scram errors	0	unknown
scram warnings	0	unknown
libChecker	0	unknown

Log file from the BuildManager

- [Log file from the BuildManager \(check here if something completely fails\).](#)
- [Log file from "scram p" and CVS checkout.](#)
- [Log file from "scram b".](#)

For the new libchecker errors and the SCRAM errors and warnings please click on the linked number to see the details for the package.

#/status	subsystem/package	compError	linkError	compWarning	miscError	UnitTest logfile
0	CalibCalorimetry/EcalPedestalOffsets V02-00-26	1	1	-	-	-
1	CalibCalorimetry/EcalSRTools V00-01-06	1	1	-	-	-
2	CalibCalorimetry/EcalTPGTools V01-04-07	4	1	-	-	-
3	CalibTracker/SiStripDCS V02-06-02	5	2	-	-	-
4	CaloOnlineTools/HcalOnlineDb V01-06-00	18	2	-	1	-
5	CondTools/Ecal V03-03-15	57	2	-	1	-

Benchmarks - Simulation

Type	Cores	Power	Events/ min/core	Events/ min/Watt
Exynos4412 Prime @ 1.704GHz	4	4W?	1.14	1.14
Xeon L5520 @ 2.27GHz	2x4	120W?	3.50	0.23
Xeon E5-2630L @ 2.0GHz	2x6	190W?	3.33	0.21

Benchmarks - Notes

- These are *very* quick and dirty benchmarks, this is a work in progress. Numbers are "indicative", not final.
- For power I used the TDP numbers from www.cpubenchmark.net, plus the quoted number for the ODROID (roughly measured by us), obviously **not** the total power cost especially for the Xeon servers
- I used one Nehalem (Q1 2010 release) and one Sandy Bridge (Q2 2012) "L" machine, both at CERN, vocms101 and vocms18. HT was on for the latter, but I have done just quick single core benchmark tests.

Porting IgProf to ARM?

- IgProf (igprof.org) is a sampling performance and memory profiler. Some notes on the ARM port:
- ARM assembly much simpler than the `x86_64` one, all instructions are 32bit long: easier to decode. Documentation is excellent.
- However its RISC-ness introduces a few new quirks to be treated when instrumenting (conditional execution, linker peculiarity, less space for the actual instrumentation in the preamble).
- RDTSC equivalent is not available in user mode.
- `libunwind` works out of the box, performance to verify

Next Steps

- Run an integration build every day, including runtime tests
- Proper benchmarking with all cores loaded, warmed up, etc. ("by the light of day....")
- Benchmarking of the CMS reconstruction
- Detailed performance studies, comparisons and validation
- Tests with multithreaded applications: CMS framework in preparation, Geant4-MT, etc.
- Repeat on ARMv8/64bit when available

Summary

- We have done some initial explorations of the use of ARM processors and nearly completed a port to ARMv7 of the entire stack of software used by CMS
- Still very much a work in progress, but we are now beginning to run a standard `fc18_arm7hl_gcc480` integration build
- We have some first benchmarks of real applications on ARMv7 processor
- Very much looking forward to ARMv8/64bit servers and seeing whether this flies or not in the market

