ARM in ATLAS (Performance of ARMs vs our Software, news)

Rolf Seuster

27. March. 2013

Introduction: Why ARM ?

- origin: computer from ACORN developed in 80ies running ARM: ACORN (Advanced) RISC MACHINE
- some 30 years later sold abundantly: smartphones, small consoles, tablets, etc.
  - cost per unit O(10euro)-O(100euro)
- demand for more computing power also for smartphones, tablets: quad-cores with >2GHz available, still being somewhat power efficient
- Grid: powerusage in computing centers one big part of total costs: power efficiency becomes more and more important

ARM becoming mainstream

- since recently ARM well supported by linux community, kernel as well as gcc
  - next 64 bit platform for ARM supported
  - gcc 48 has further improvements for ARM
  - ARM also supports SIMD-like instructions
- ARM 64bit platform 'AArch64' announced, first hardware expected in 2014 (also from AMD !!)
- main question: How well does ATLAS code run on ARM 32 bit and ARM 64 bit ? Can it deliver more "events per Watt" ?

Linux and ARM

- latest Ubuntu well supported for many ARM SoC (system on chip)
  - Linaro(linaro.org): non-profit consortium working on supporting Android and Linux
- latest Fedora 18 now also available for ARM, so not impossible that native RHEL will also run on ARM in future, maybe even next RHEL7 ?
  - private efforts: RedSleeve (redsleeve.org) ARM clone based on RHEL6
- future: if we ever support ARM, important to have decent distribution supported for both ARM and X86

ATLAS' investigations so far

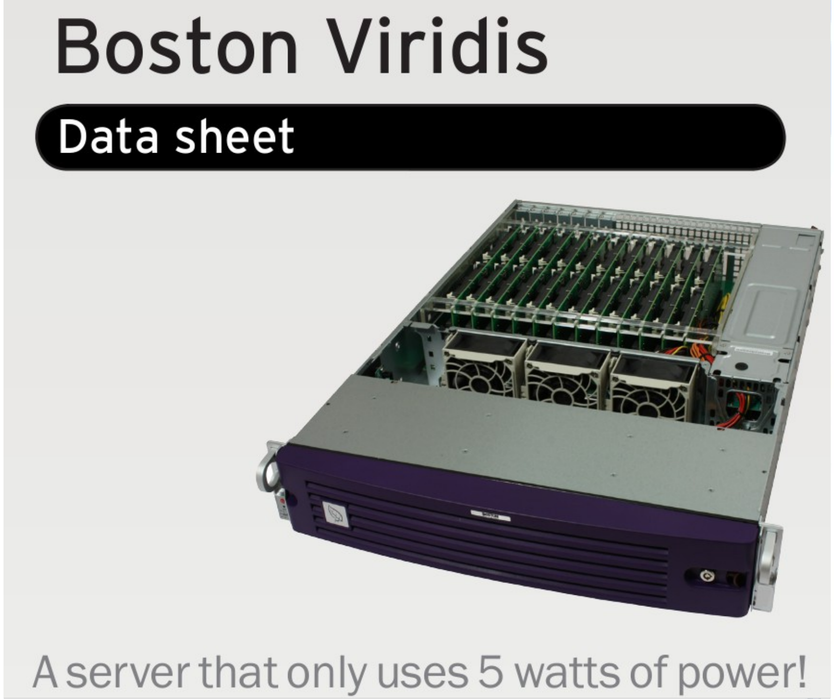
- Stefan Kluth (MPI) running on Boston Viridis, run HepSpec and compare to Intel
- me running on two different ARM small computers, run just examples for fastjet, geant and compile root

- in the following slides, I present very preliminary results about power efficiency from ATLAS studies

Stefan's finding:

- HepSpec on Boston Viridis ARM Cortex-A9 @ 1.1GHz vs. some Intel machines

Platform	HS06/W per single core
ARM A9	10.4 / 5.5 = 1.89
E5-2670	339 / 300 = 1.13
E5645	179 / 250 = 0.72
E5620	130 / 250 = 0.52



Boston Viridis

Data sheet

A server that only uses 5 watts of power!

- ARM clear winner in HS06 / Watt however: initial cost for ARM server too high ! currently > 2x as expensive as Intel solution

my findings:

- testbed, running fastjet a la “make check” on
  - dual Cortex-A9, 1.2 GHz, 1GB: ubuntu 12.04
  - Intel(R) Xeon(R) CPU E5530, 2.40GHz
  - Intel(R) Core(TM) i7-3612QM CPU, 2.10GHz

ARM	Xeon	Core-i7	ARM/Xeon	ARM/Core-i7
275.75	40.09	34.24	6.88	8.05
252.00	28.57	21.27	8.82	11.85
233.12	25.95	18.00	8.98	12.95
145.31	15.14	11.58	9.60	12.55

==> ARM core is ~ 1/10 of Intel Core

- notes: gcc 46 for ARM used – better with 47 ? also: single core – not full WS !!

my findings: cont'd

- scaling to full workstations:
  192 core WS (300W) ARM server against
  8(16 hyperthreaded) Intel WS (500W – estim!)

  Intel: 1 * 8 * 1.3 / 500 W = 0.021 / W

  ARM: 1/10 * 192 / 300 W = 0.064 / W
- one would get ~3 more per Watt from ARM based of fastjet's “make check” similar to what LHCb and Stefan sees, but one needs ~10x more CPUs to archieve similar performance

Future:

- Newest addition to ATLAS "Computing FARM"

5V 2A Power supply

Cubox ARM computer
ARM Cortex A9, 800MHz
eSata connector and
RHEL clone running
"RedSleeve"

with 250GB 2.5inch HDD
connected via eSata

NEW: root filesystem now on eSata – machine feels **much** faster now ...

Compiling on/for ARM

- Cubox with 0.8 GHz single core too slow to do full compilation of big packages (compiling gcc48 took ~20h)
  - running will not profit from the eSata interface ...
- qemu emulates plenty of non-x86 CPUs
  - emulated ARM cpu runs much slower that x86, but so does real ARM cpu ...
    runs simple programs as bash OK:

```
> qemu-arm -L testARM/rsel6-rootfs  testARM/rsel6-rootfs/bin/bash      (*)
bash-4.1$
```

  - so far couldn't run compiler inside qemu to produce usable output (furthermore: do I actually want to run it inside qemu ??)

[*] testARM/rsel6-rootfs : location of redsleeve root filesystem

Compiling on/for ARM $^{\text{cont'd}}$

- cross compilers provided by linaro look more promising:
  - should run at ~native speed on Intel CPUs, full use of multi core
  - linaro uses crosstools-NG for compilation – does it suit our needs as well ? some programs assume host==target and compile programs which run during compilation ...
  - combination with qemu etc maybe ?
  - or follow LHCb's approach ?

Future cont'd

- will follow up with low priority ARM platform
  - gcc48 just compiled OK on Cubox – about to start next round of comparisons: root benchmarks, geant examples, fastjet, etc...
- with help of SFT or LHCb compile whole stack of externals to start compiling ATLAS code
- plan to re-configure Cubox to use kernel from RedSleeve, then try to install AFS
  - would help many things e.g. cross compiling

Conclusion

- ARM looks promising, would be even more if we start up computing now ...
- work done on porting code can be valueable in future (also for BlueGene, other SC, etc)
  - scripts, procedures, experience
  - excellence cluster in Germany looking for full time persons (one looking at ARM ...)
- have another closer look at ARM64 when it comes out in 2014