

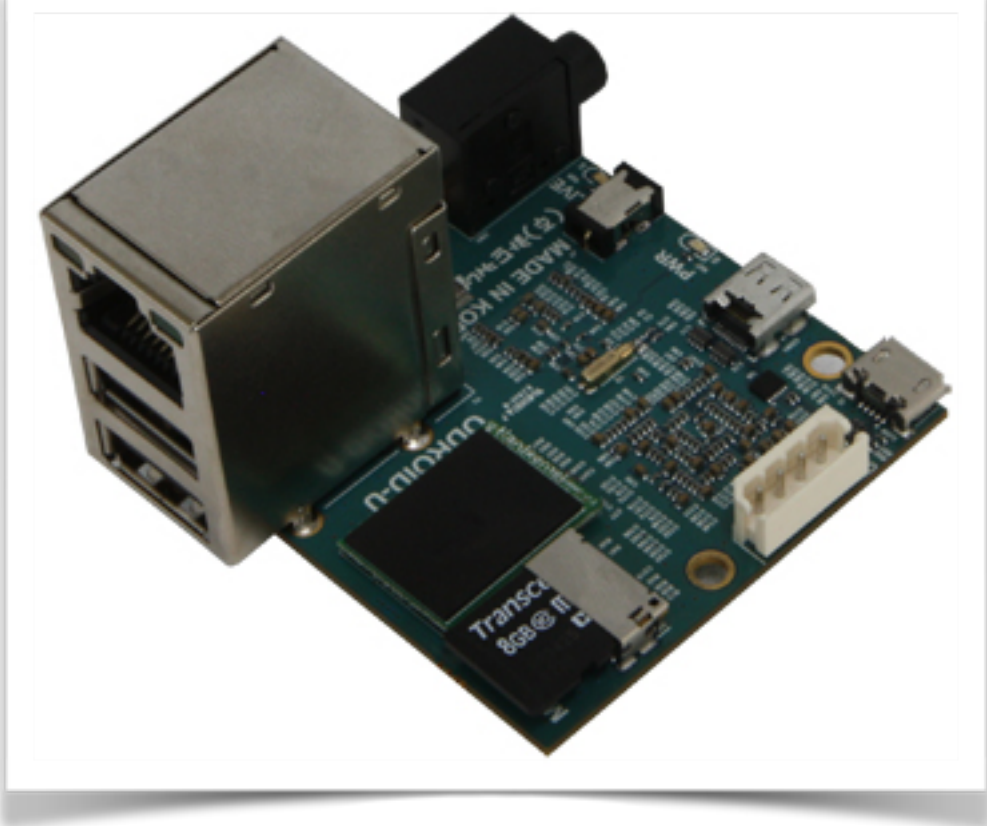
CMS - Status of new
architectures and technologies

Peter Elmer, Princeton University
Giulio Eulisse, Shahzad Muzaffar, FNAL
David Abdurachmanov, Vilnius University

Introduction

- This presentation contains a mix of information on the status of of several new technologies/architectures
- Some of it has been shown in earlier in one form or another at ACAT, by David here and elsewhere, but there are some small updates
- Most of this is still in "raw" form and not even fully discussed or presented in CMS, so take this as just an informal "technology" discussion.
- We will have to see how these things will eventually be used, but there are some promising aspects.

ARM Demonstrator - ODROID U2

- Initial tests done with a small 32bit/ ARMv7 development board
 - Exynos4412 Prime CPU
 - 1.7GHz Cortex-A9 quad core
 - 2GB L-DDR memory (total)
 - eMMC, microSD, 2xUSB2.0, 10/100Mbps Ethernet
 - \$89 (~\$233 with cables, cooling fan, 64GB eMMC, power adaptor, ...)
- 
- Fedora 18 ARMv7-A, hard floats, gcc 4.8, ODROID kernel

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 (list price?)
- servers with the new ARMv8/64bit cores, expected next year, will likely be the product that will either create (or not) sufficient market share
- Dell "Copper" servers:
 - Current: 48 x quad-core Marvell Armada XP SoC = 192 ARMv7 cores, 1GB/core (3GB visible?), 3U rack?
 - Future: 48 x quad-core=192 ARMv8 cores, 2GB/core, 750W in a 3U rack mount?

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

First Benchmarks - Simulation (no output)

Type	Cores	TDP 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

First Benchmarks - Simulation (no output)

Type	Cores	full Power	Events/ min/core	Events/ min/Watt
Exynos4412 Prime @ 1.704GHz	4	6W?	1.14	0.76
Xeon L5520 @ 2.27GHz	2x4	240W	3.50	0.12
Xeon E5-2630L @ 2.0GHz	2x6	270W	3.33	0.15

Benchmarks - Notes

- These are *very* quick and dirty benchmarks, this is a work in progress. Numbers are "indicative", not final. Only very basic checks have been that results are consistent. ROOT output is still off.
- 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. For those (second table "full power") I used some numbers from Bernd.
- 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.

Dictionary Issues

- Comparing ARMv7-A and AMD64 executions integer-overflow was detected, which resulted in a code path on AMD64 not being executed
- A number of checks were implemented by Philippe in ROOT for (and similar) issues:
 - *“...missing the compiled STL collection proxy for the data member of compiled class.”*
- Checks/fixes are available in ROOT 5.34.09

Dictionary Issues

- In CMSSW_6_2_X we see 220+ errors generated by ROOT:
 - a missing dictionary (a collection) for data member or a base class
 - unknown type
 - missing streamer or dictionary
 - discarding <type>, no [dimension]
- *genreflex* does not share the same logic w/ rest of ROOT. No errors received generating dictionaries.
- ROOT 5.34.09 and fixed errors to land in CMSSW_7_0_X (new release series just opening)

Checkpoint-Restart

- It is desirable in certain circumstances to "checkpoint" the state of a unix process, or set of processes, to disk with the possibility of restarting it at a later time.
- This can be done in an application-specific custom fashion, but it requires the addition and maintenance of dedicated code.
- A generalized technology capable of checkpointing all types of applications is thus desirable. In fact such technologies have been in use in High Performance Computing (HPC) and batch systems since more than 20 years.

Checkpoint-Restart - Interesting Use Cases

- Avoiding CPU-intensive initialization steps in frequently run applications, perhaps avoid need for conditions or other loading
- Reproducibility of problems in long running jobs for debugging
 - The application can be "replayed" from a point just before the error or crash, rather than from the beginning
- In situations where resources are being used opportunistically, it can be used to efficiently give access back to the "owner" and then later restart when resources are free again
- In interactive applications, the current state can be saved ("workspace")
- For long-running parallel applications sensitive to hardware failure, the state of calculations can be saved periodically to allow restart.

DMTCP

- Distributed MultiThreaded CheckPointing package (DMTCP), developed at Northeastern University (NEU), <http://dmtcp.sourceforge.net>

Key Features

Userspace checkpointing, no kernel-level access required

Checkpoints multithreaded applications

Checkpoints distributed applications

Can handle fork, exec, ssh, open file descriptors, TCP/IP sockets, etc.

Minimum runtime overhead

Optional compression of checkpoint images

Open source

Works on linux and supports a wide range of kernels

DMTCP - CMS Example

- Quick test with CMS Framework-based generation/simulation application, memory footprint ~750MB RSS
- ~10s required to create compressed checkpoint image, 220MB
- 1-2s for uncompressed checkpoint

```
Begin processing the 1st record. Run 1, Event 1, LumiSection 1 at 18-May-2013 05
:10:01.557 CEST
Begin processing the 2nd record. Run 1, Event 2, LumiSection 1 at 18-May-2013 05
:10:01.584 CEST
WARNING: G4QPDGToG4Particle is deprecated and will be removed in GEANT4 version
10.0.
Begin processing the 3rd record. Run 1, Event 3, LumiSection 1 at 18-May-2013 05
:10:10.014 CEST
Begin processing the 4th record. Run 1, Event 4, LumiSection 1 at 18-May-2013 05
:10:14.434 CEST
Begin processing the 5th record. Run 1, Event 5, LumiSection 1 at 18-May-2013 05
:10:18.362 CEST
```

Trigger checkpoint externally while processing event #5

DMTCP - CMS Example

- And restart:

```
vocms101> ./dmtcp_restart_script_a9bf217912ea647-48000-5196f087.sh
dmtcp_restart (DMTCP + MTCP) 2.0
Copyright (C) 2006-2011 Jason Ansel, Michael Rieker, Kapil Arya, and
Gene Cooperman
```

```
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; see COPYING file for details.
(Use flag "-q" to hide this message.)
```

```
Begin processing the 6th record. Run 1, Event 6, LumiSection 1 at 18-May-2013 05
:15:53.486 CEST
```

```
Begin processing the 7th record. Run 1, Event 7, LumiSection 1 at 18-May-2013 05
:15:56.754 CEST
```

```
G4Fragment::CalculateExcitationEnergy(): WARNING
```

```
Fragment: A = 26, Z = 12, U = -3.520e-01 MeV IsStable= 1
```

```
P = (-9.200e+01, 1.075e+02, -2.607e-01) MeV E = 2.420e+04 MeV
```



This was on x86-64, also ARM and Xeon Phi supported

Comments and ideas for next steps

- Note that there is a "closed world" assumption, i.e. external connections need to be closed and reopened
- External trigger for checkpoint asynchronous with Framework states, however a callable API also exists
- Can imagine using Framework services and/or USR signal (and handler) to make checkpoint synchronous

Xeon Phi

- 60-ish lightweight cores with big vector units, coprocessor packaging on PCIe bus
- Practical difficulties even to play with it:
 - Cross compilation from x86-64 required
 - Intel compiler required
 - No software environment available
- Offload vs direct running on the card (future?)

Xeon Phi

- Probably not sensible (or performant) to run entire CMS applications on the Xeon Phi, but it would facilitate tests to have some software development environment
- Solution: produce a CMSSW release subset with a smaller set of externals available (cross-compiled) and a subset of the CMSSW itself (code which compiles with icc)
- Mechanically it implies that one can create a SCRAM development area on the x86-64 host and code checked out and built will automatically be cross compiled for the Xeon Phi

Xeon Phi - CMSSW release subset status

- Release CMSSW_6_2_0_pre7 with arch slc6_mic_gcc472 available
- Externals: root, boost, xrootd, clhep, python, pythia8, roofit, fastjet, gsl, many other smaller packages (not all 125 yet, though)
- Separate work on direct compilation of CMSSW code with icc underway. I expect we will have iterations on the release to make a larger subset of code available. Just one dummy CMSSW package included in this "subset" release at the moment.
- Should help facilitate some types of prototyping on Xeon Phi, without having to start from main() and by-hand Makefile

Cross-compilation notes

- Mostly external were build by just setting CXX="icpc -fPIC -mmic" and CC="icc -fPIC -mmic" and --host=x86_64-k1om-linux to configure scripts for cross compilation
- Boost: Patched couple of files [b] and used TOOLSET intel
- Python: Needs to build it twice, once for build system and once for Xeon Phi cross compilation.
- Fastjet: without -msse3
- GSL: Fixed configure script to not run test when cross-compiling
- OpenSSL: Configured w/o -fstack-protector and --with-krb5-flavor
- Root: Patched to build few executables without -mmic, Built without fftw3, castor and dcap dependency

The End
