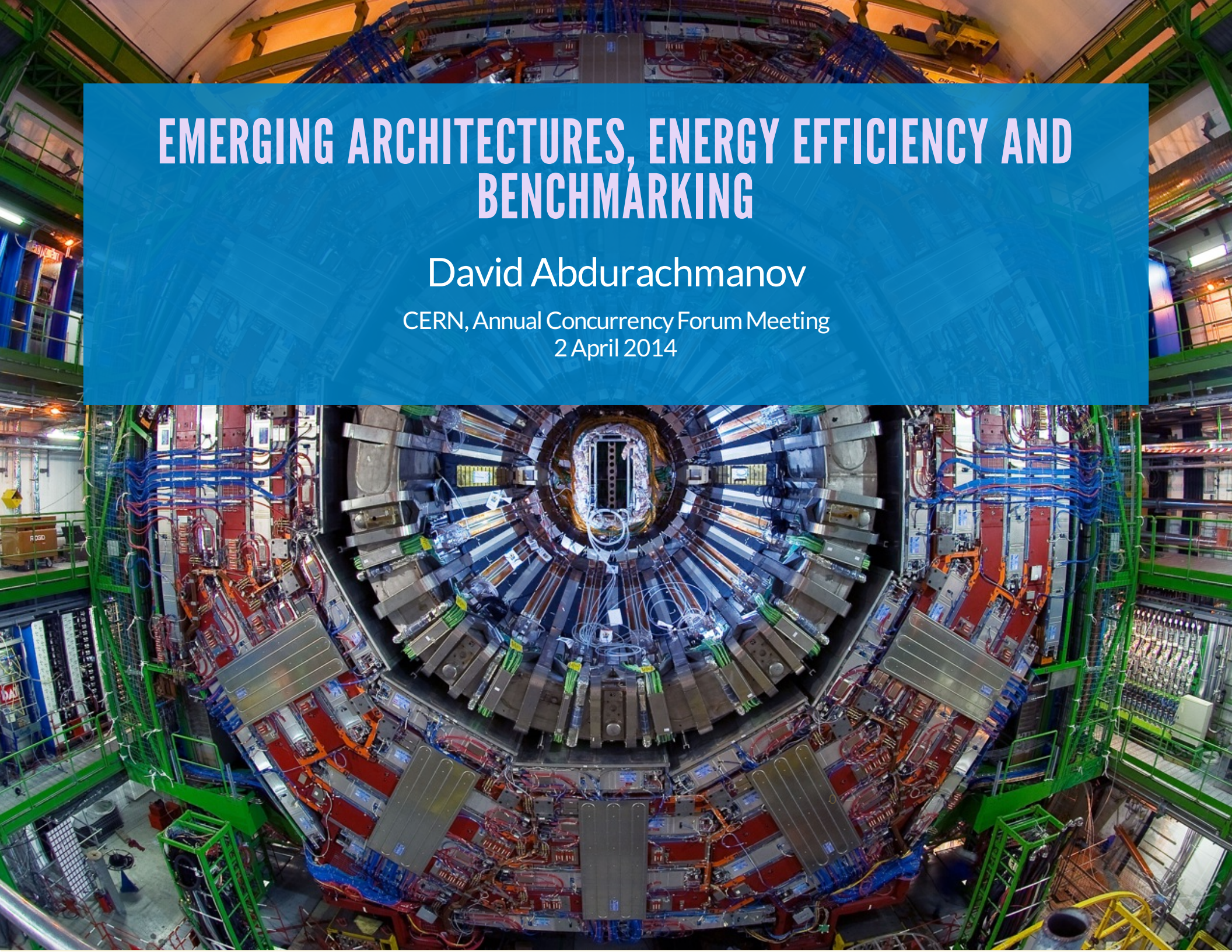


EMERGING ARCHITECTURES, ENERGY EFFICIENCY AND BENCHMARKING

David Abdurachmanov

CERN, Annual Concurrency Forum Meeting
2 April 2014

NEW HARDWARE

There are two different types of new hardware.

General purpose CPUs: ARMv7, ARMv8, and Xeon Phi [native mode].

A full native port to a new architecure is required.

Co-Processor: GPUs and Xeon Phi [offloading mode].

Running CUDA 5.5/6.0 and OpenCL 1.1 Full-Profile. We offload some parts of computation to a co-processor. No port is required, execution is handled by libraries.

Heterogeneous environment becoming a common practice, e.g., CPU + Co-Processor, big.LITTLE.

A PATH TO ARMV7

We acquired Exynos4412, Exynos5410, and Exynos5420 powered community development boards.

Development boards in order: ODROID U2, ODROID XU+E, and Arndale Octa.

We have support for Dell Copper server system (Marvell PJ4Bv7) in CMSDIST.

We just [31 March 2014] reiceved Kayla Platform (Tegra 3). It will be used together with NVIDIA K40.

We use Fedora 18 & 19 and Linaro Ubuntu releases.

KAYLA PLATFORM + TESLA K40

CMSSW ON ARMV7

Full ports are available for fc18_armv7hl_gcc481 and fc19_armv7hl_gcc482, but not error-free.

We continue to provide CMSSW_7_1_X integration builds each day on publicly available armcms{1,2}.cern.ch machines.

ROOT5 I/O issues continue to be ARMv7 showstopper. We can only run GEN-SIM with output disabled.

No alternative output formats are supported by CMSSW.

A PATH TO ARMV8 [AARCH64]

Work done on ARMv7 is a faundation for ARMv8 port.

We started porting using ARM Foundation Model (a full system simulation).

Architecture was named fc19_aarch64_gcc482 and is highly experimental.

Required migrating to newest RPM version, modifications to PKGTOOLS, and taking care of apt-get porting.

Only a fraction of CMSDIST is currently ported to AArch64.

PROGRESS ON AARCH64

ARM Foudation Model was replaced by QEMU (OpenSUSE fork)/chroot/binfmt_misc using application mode emulation.

Significantly improved compilation times.
No limitation for number of cores or memory.
Root permissions are required, but alternative solutions should be possible (e.g, PRoot).

Since mid-March 2014 we are running on a real server-class AArch64 hardware [under NDA].

Acquiring hardware is complicated, we started it back in June 2013.
The initial RPMs for `fc19_aarch64_gcc482` architecture works fine on a real hardware.
First AArch64 ice-on-valid-code was found in GEANT4 10.00 (`-mcpu=general` or `-mcpu=cortex-a53`) and fixed for GCC 4.9.0.

ROOT5 ON AARCH64

ROOT5 is blocked on AArch64 due to GCCXML using an old version of GCC (last GPLv2).

GCC 4.2.1 was released in 21 July 2007 and there was no hints of AArch64 at a time.

GCCXML can easily be converted to a GCC plugin supporting newest GCC versions.

Some work has been done on this, but current plugin output does not satisfy `genreflex`.
Main porting done by Alex Leach and available on GitHub as `gccxml_plugin`.

ROOT6 ON AARCH64

ROOT6 does not compile out of the box on AArch64 running Fedora 19.

ROOT6 uses "old" JIT from LLVM and not MCJIT. Neither AArch64 nor ARM64 backends supports "old" JIT. Migration to MCJIT in June/July after ROOT 6.00.

From Tim Northover (former Senior Compiler Engineer at ARM [AArch64] and now Compiler Engineer at Apple):

> *Yep. AArch64 should support MCJIT. ARM64 certainly does on Darwin and *theoretically* Linux will work, but I've not tested it. There are no plans for either to support the old JIT.*

```
cling::IncrementalExecutor::IncrementalExecutor()
target does not support JIT code generation
```

WHAT INDURSTRY WANTS?

ARMv8 comes with EUFI support. EFI GRUB instead of U-Boot.

ARMv8 comes with ACPI support. No more Device Trees (DTs).

AArch64, no interest into AArch32 and/or multi-arch support. Pure 64-bit only Linux distribution.

No vendor specific kernel configuraton options.

Sounds like yet another PC, doesn't it?

A PATH TO XEON PHI

We have access to CERN openlab machine with 4 7120P Xeon Phi accelerator cards.

Currently we are attempting to run CMSSW on Xeon Phi native mode. We constantly provide Intel C++ Compiler cross-compiled CMSSW_7_1_X builds for slc6_mic_gcc481.

Such builds are hightly broken and non-operational.

Intel C++ Compiler C++11 support is behind what is provided by open source GCC and LLVM/Clang.

A PATH TO GPUS

We received two K40 cards and two Kayla Platforms from NVIDIA.

One K40 was installed into TechLab machine with K20 card.
Antother K40 will be used with Kayla Platform [ARMv7, Tegra 3].
The board will be running Ubuntu derived distribution, most likely L4T [Linux for Tegra].

We have support for CUDA 5.5/6 and OpenCL 1.1 Full-Profile.

Non-default CMSSW toolfiles are available: `cuda`, `opencl`, and `opencl-cpp`.
`cuda` and `opencl` packages contains symlinks to the actual locations of run-time libraries and headers.
Users can change the symlinks in local CMSSW instalation.

BENCHMARKING ARCHITECTURES

GEANT4 10.00 ParFullCMS is currently used to benchmark different architectures. The only HEP benchmark running on ARMv7, ARMv8, Xeon Phi, and x86_64 CPUs.

CMSSW would be ideal benchmark, but it doesn't run properly on ARMv7, Xeon Phi, and porting to ARMv8 is in early stages.

We also use HEP SPEC 2006 and SciMark 2.0 for general bechmarking.

BECHMARKING ENERGY-EFFICIENCY

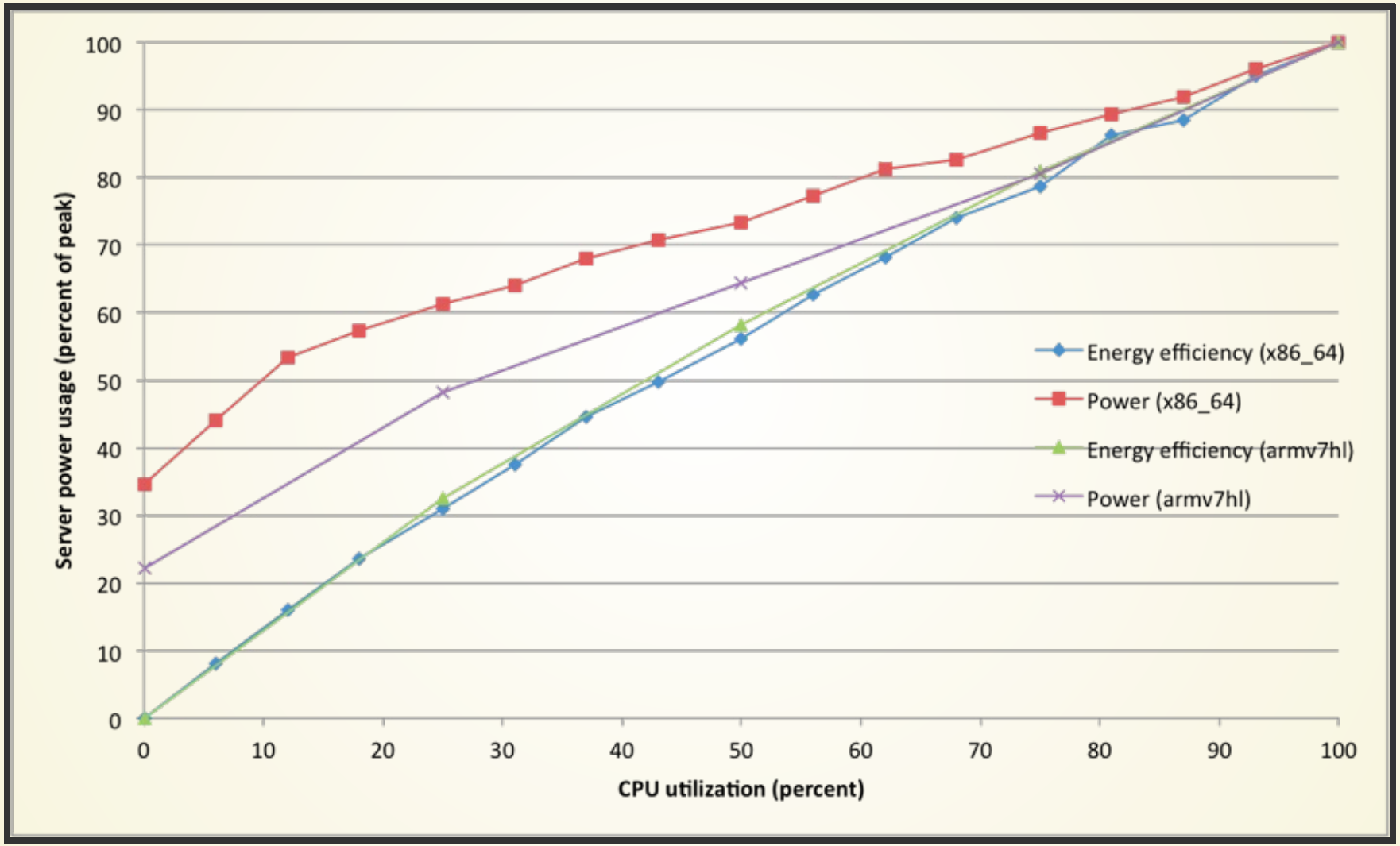
We used ODROID Smart Power to collect data on power consumption of Exynos4412, Exynos5410, and Exynos5420 dev boards.

Exynos 5410 dev board includes current/power monitors to measure power consumption of A15 cluster, A7 cluster, memory, and GPU.

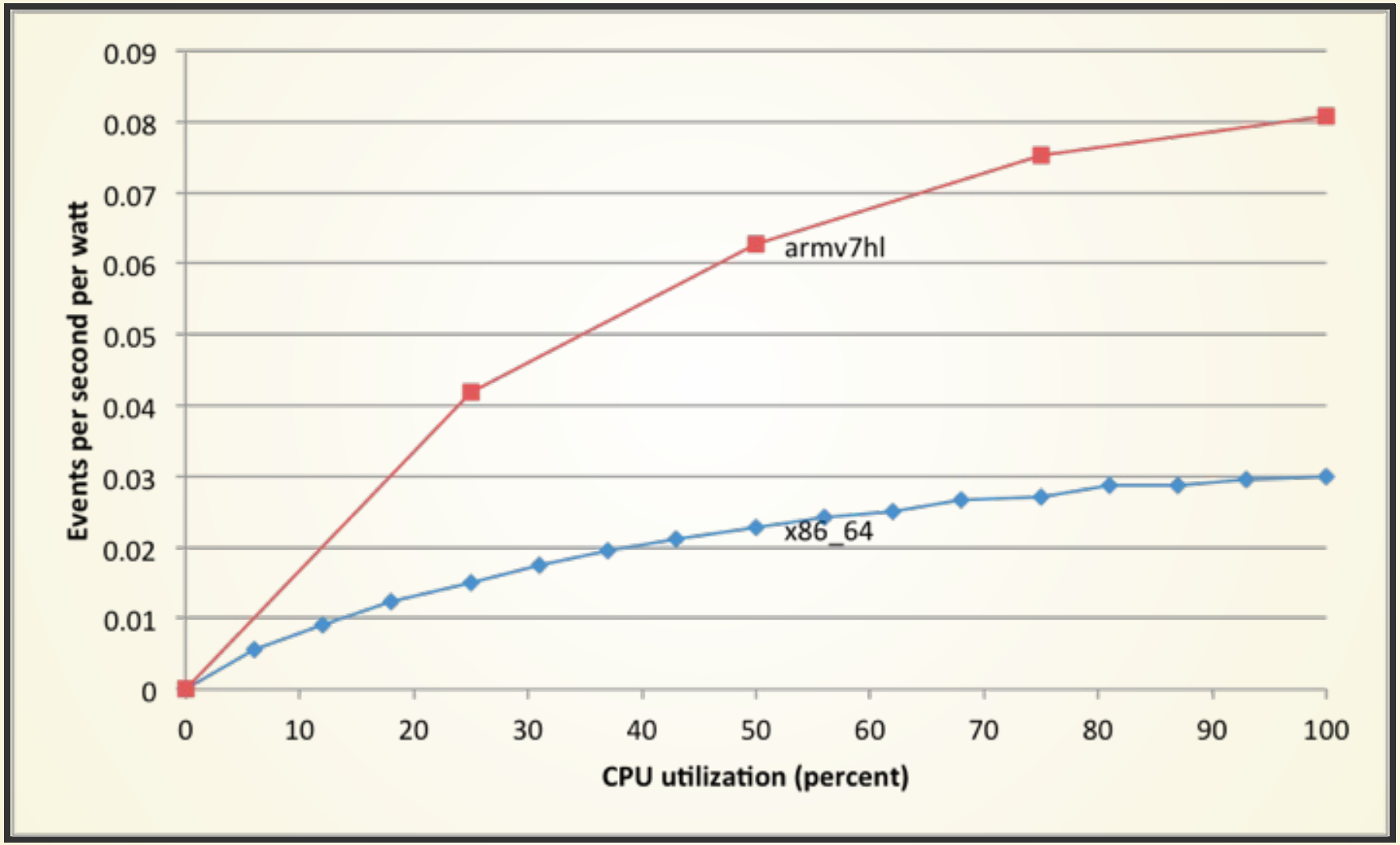
We had access to Princeton University TIGER cluster node with IPMI enabled, which reports power used by a blade.

We used this in combination with ParFullCMS to measure energy-proportionality on Exynos5420 and Xeon E5-2670.

ENERGY-PROPORTIONAL COMPUTING



PERFORMANCE/WATT



Q & A

Contact: hn-cms-sw-develtools **dot** cern **dot** ch