Energy-Efficient Building Blocks For Rack Scale Computing

Work In Progress

Rami Alkubaty

Fachhochschule Südwestfalen
University of Applied Sciences

Contents

- Motivation
- Approach
- Initial Experiments and First Insights
- Next Steps

Motivation

- Rack scale systems are present or will be present in various business domains

- Various requirements
  - Energy efficiency
  - Performance
  - Cost
  - …and many others

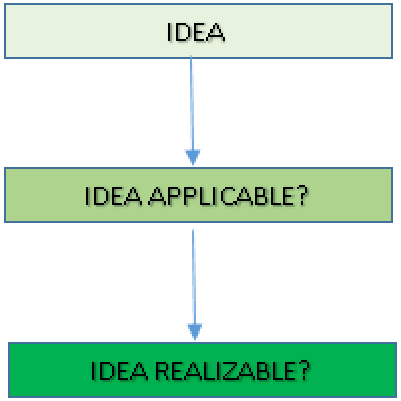
- Various load characteristics from very static to highly fluctuating

Image: http://www.techrepublic.com

Motivation: Our Focus

- Unit of consideration: The rack
- It gets load
  - from customers, or
  - datacenter coordinator

- We consider scenarios with
  - Highly fluctuating load
  - Individual target requirements
    - High performance
    - Energy efficiency
    - Different tradeoffs between energy and performance
    - Dynamic changes of these requirements



Approach: High Level

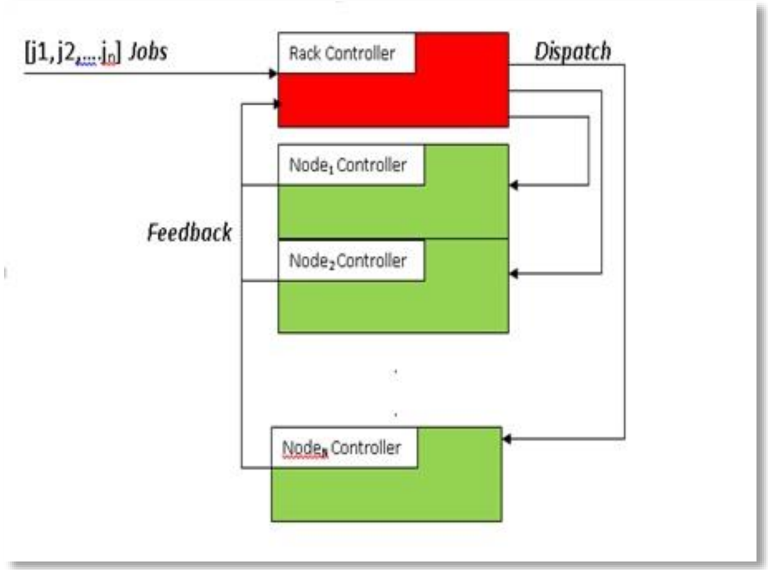
- Tasks associated with information about energy-performance trade-off

- Two-level control system:
  - Rack controller:
    - coarse grained load distribution

  - Node Controller:
    - fine grained decision how to deal with load

  - Feedback channel:
    - reports on load-status
    - “evaluates” RC decision

- **FOCUS: NODE**



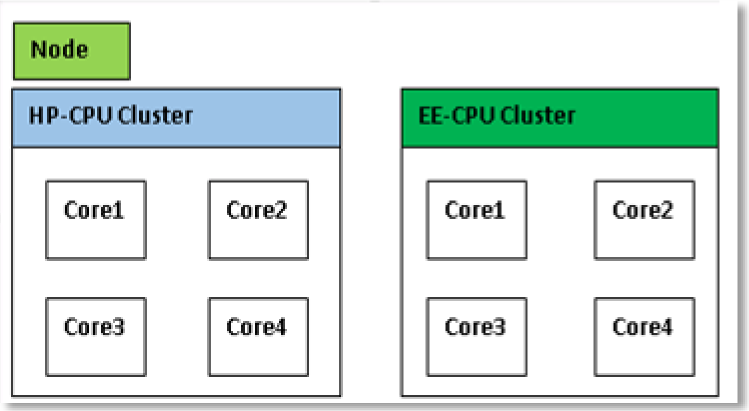
Fachhochschule Südwestfalen
University of Applied Sciences

Approach: Heterogeneity

- Heterogeneity is the way to go!

- Rack: different computers
  - We are NOT considering this

- Node:
  - heterogeneous processors having the same ISA (Instruction Set Architecture)

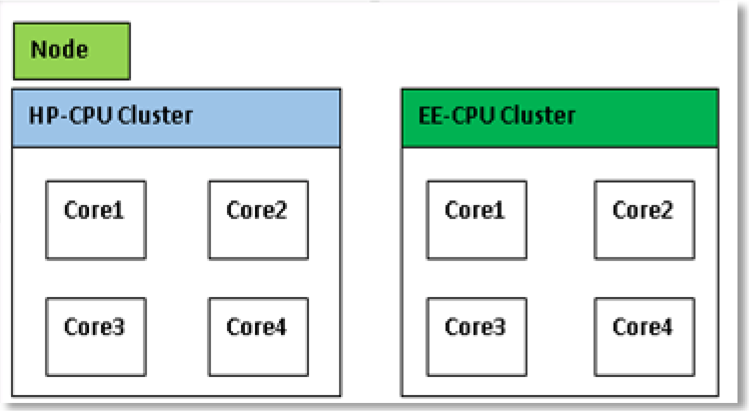


Approach: Challenge

- ***How to use heterogeneous processors efficiently?***

  There is no magic receipt!
  Analysis (Statistical, heuristic,…)?

  No, our approach considers the system as a black-box



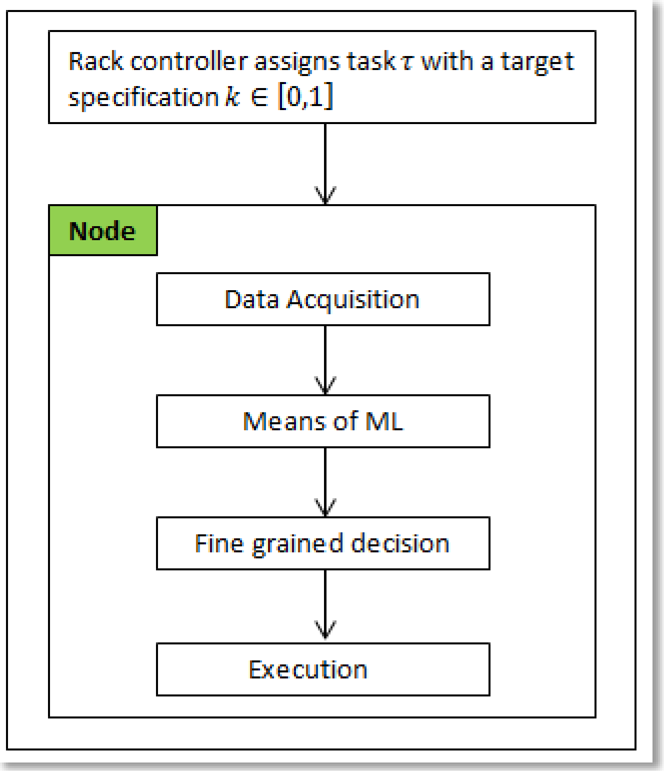
Approach: Black Box

- Black box can be realized by the means of Machine Learning

- Using Machine Learning means we need to:
  - know if patterns exist
    if so:
    - acquire data
    - build mathematical model

- Data Acquisition: *Performance Monitoring Counters* (PMCs) (and Energy measurements)

- Mathematical Model: Unsupervised Learning (later on!)


Rack controller assigns task τ with a target specification $k \in [0,1]$
Node
Data Acquisition
Means of ML
Fine grained decision
Execution

Approach: Summary

- **We think:**

i) Tackling energy-efficiency & performance tradeoff with CPU **heterogenity** (same ISA) within the node

ii) Considering systems (also Rack Scale Systems) as **black box** to decouple diversity & rapid development

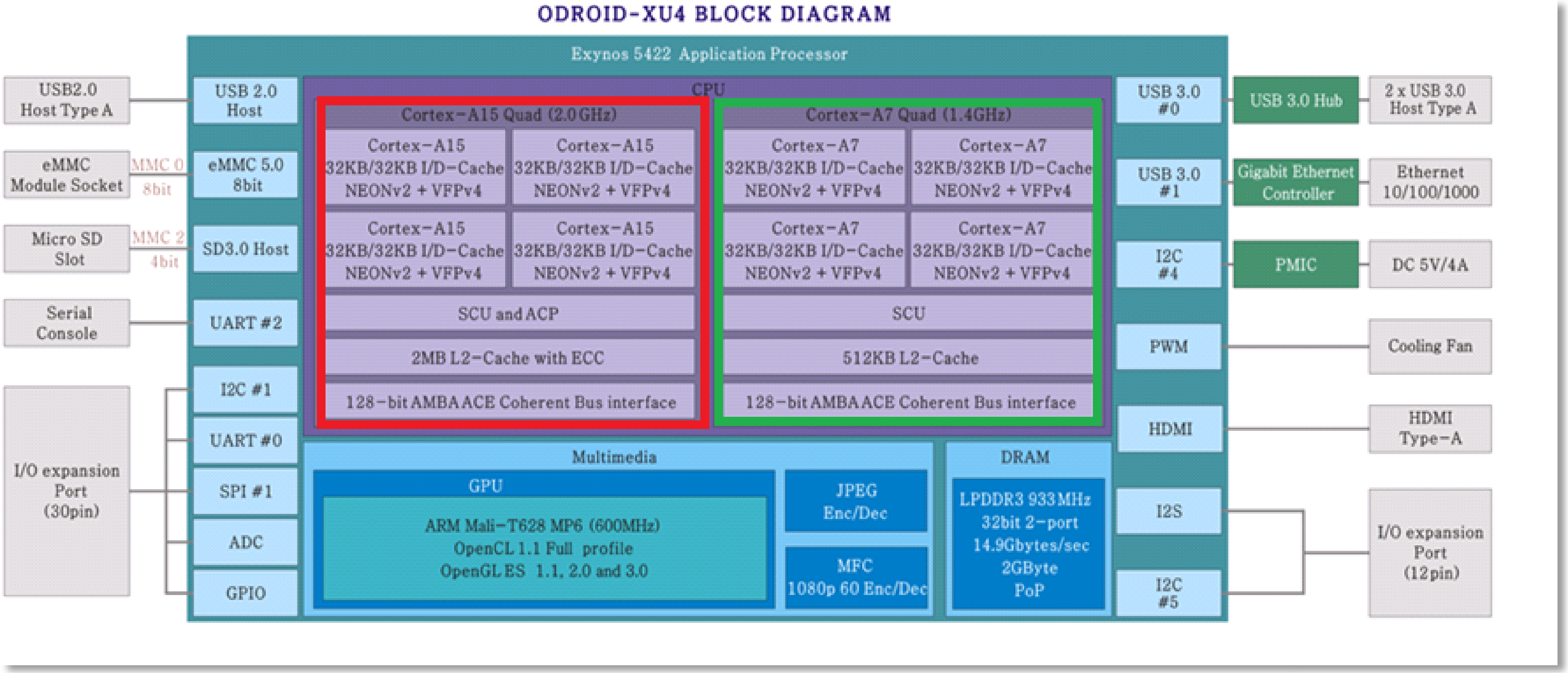
Initial Experiments And First Insights

- Bear in mind this work still in progress!
- We are still in the very early phases where we are trying to find out if this works!
- Our work is inspired by (but not based on):
- Josep LI. Berral et. al., 2010
  “Towards energy-aware scheduling in data centers using machine learning”
- Matthew J. Walker et. al. 2016
  “Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs”
- A. Weisel, F. Bellosa, 2002
  “Process cruise control: event-driven clock scaling for dynamic power management”

Initial Experiments And First Insights

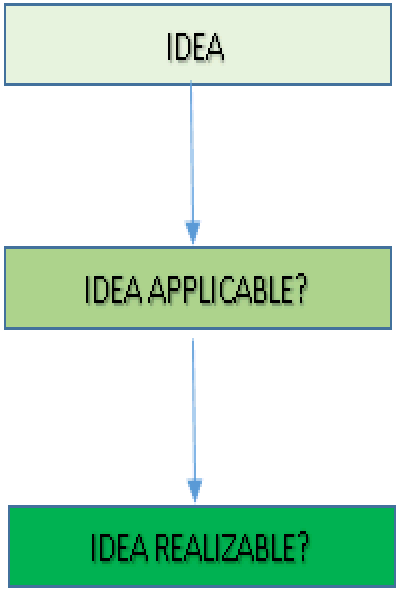
- **Experiments:**

  + Hardkernel Odroid xu4 (image: http://hardkernel.com)


ODROID-XU4 BLOCK DIAGRAM
Exynos 5422 Application Processor
USB2.0 Host Type A
eMMC Module Socket
MMC 0 8bit
Micro SD Slot
MMC 2 4bit
Serial Console
I/O expansion Port (30pin)
USB 2.0 Host
eMMC 5.0 8bit
SD3.0 Host
UART #2
I2C #1
UART #0
SPI #1
ADC
GPIO
CPU
Cortex-A15 Quad (2.0GHz)
Cortex-A15 32KB/32KB I/D-Cache NEONv2 + VFPv4
Cortex-A15 32KB/32KB I/D-Cache NEONv2 + VFPv4
Cortex-A15 32KB/32KB I/D-Cache NEONv2 + VFPv4
Cortex-A15 32KB/32KB I/D-Cache NEONv2 + VFPv4
SCU and ACP
2MB L2-Cache with ECC
128-bit AMBA ACE Coherent Bus interface
Cortex-A7 Quad (1.4GHz)
Cortex-A7 32KB/32KB I/D-Cache NEONv2 + VFPv4
Cortex-A7 32KB/32KB I/D-Cache NEONv2 + VFPv4
Cortex-A7 32KB/32KB I/D-Cache NEONv2 + VFPv4
Cortex-A7 32KB/32KB I/D-Cache NEONv2 + VFPv4
SCU
512KB L2-Cache
128-bit AMBA ACE Coherent Bus interface
Multimedia
GPU
ARM Mali-T628 MP6 (600MHz) OpenCL 1.1 Full profile OpenGL ES 1.1, 2.0 and 3.0
JPEG Enc/Dec
MFC 1080p 60 Enc/Dec
DRAM
LPDDR3 933MHz 32bit 2-port 14.9Gbytes/sec 2GByte PoP
USB 3.0 #0
USB 3.0 #1
I2C #4
PWM
HDMI
I2S
I2C #5
USB 3.0 Hub
Gigabit Ethernet Controller
PMIC
2 x USB 3.0 Host Type A
Ethernet 10/100/1000
DC 5V/4A
Cooling Fan
HDMI Type-A
I/O expansion Port (12pin)

Fachhochschule Südwestfalen
University of Applied Sciences

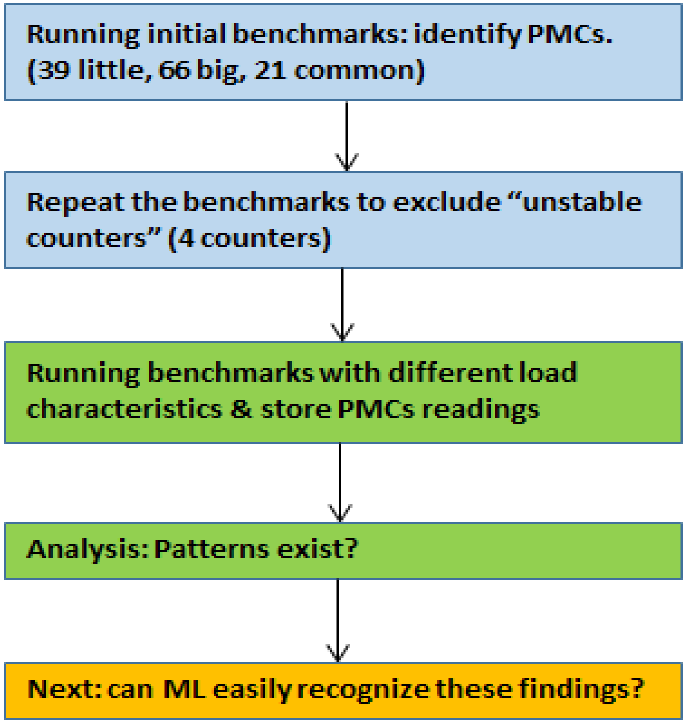
Initial Experiments And First Insights



IDEA

IDEA APPLICABLE?

IDEA REALIZABLE?



Running initial benchmarks: identify PMCs. (39 little, 66 big, 21 common)

Repeat the benchmarks to exclude "unstable counters" (4 counters)

Running benchmarks with different load characteristics & store PMCs readings

Analysis: Patterns exist?

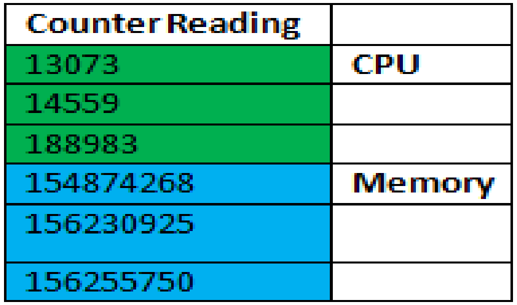
Next: can ML easily recognize these findings?

Initial Experiments And First Insights

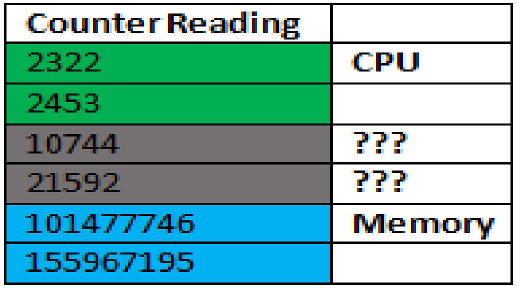
- Huge data samples.

- Empirical analysis does not show the insights all the time.

- We rely on ML



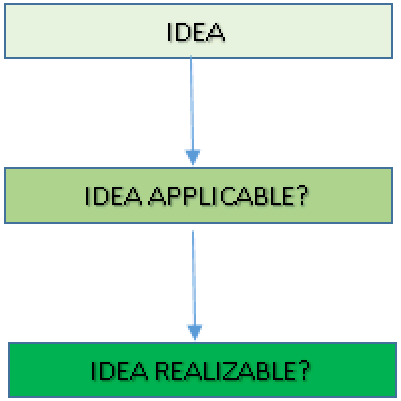
Counter Reading	
13073	CPU
14559	
188983	
154874268	Memory
156230925	
156255750	



Counter Reading	
2322	CPU
2453	
10744	???
21592	???
101477746	Memory
155967195	

Initial Experiments And First Insights

- WE need to observe how PMC behave when apply different on the system
- Is PMCs grouping possible? Is it unique? What is the system status thereby?
- *sytemStatus = f(MPCs)*
- Clustering? ML helps, specifically Unsupervised Learning

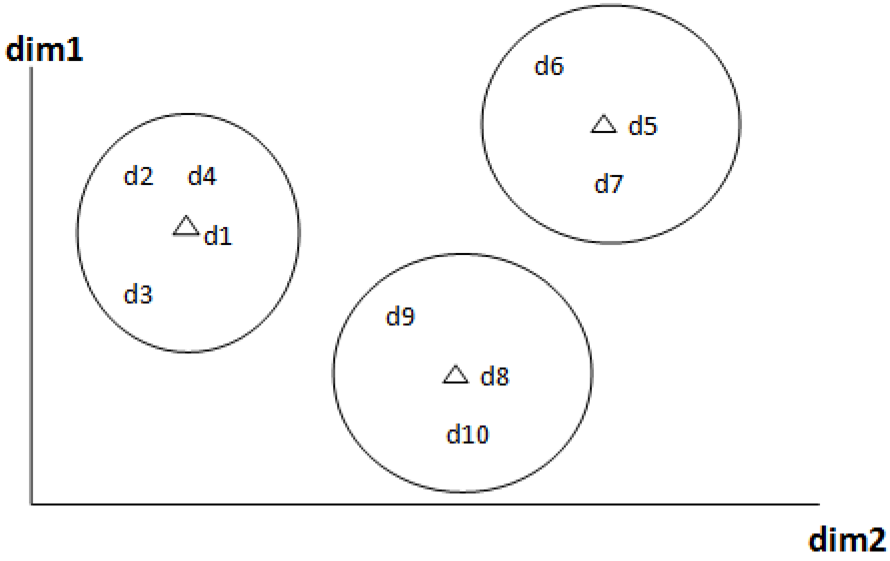

IDEA

IDEA APPLICABLE?

IDEA REALIZABLE?

Initial Experiments And First Insights
Unsupervised Learning

- Contrary to Supervised Learning we do not need trained labeled dataset
- In unsupervised learning we are trying to draw inferences from unlabeled dataset
- SL → Classification, USL → Clustering (KNN: K Nearest Neighbors)
-
- D1= [a1, b1, c1]
  D2= [a2, b2, c2]
  …..
  Dn=[an, bn, cn]



Initial Experiments And First Insights
Unsupervised Learning

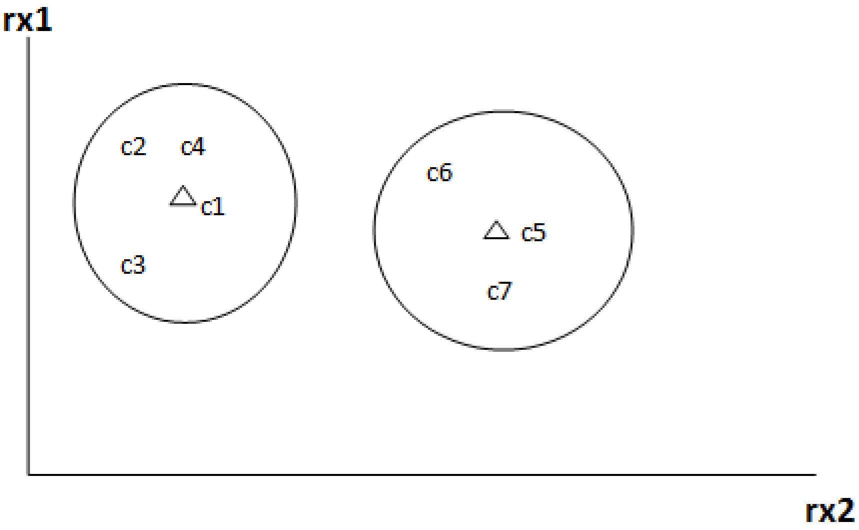
- How this would look like? An overview (rather a very simplified one in 2D)
  We consider the case when the system is lightly unloaded

- N-dataset of PMCs readings

- c1 = [r11, r12]
- c2 = [r21, r22]
  …..
- cn = [rn1, rn2]

- rx1 = counter’s reading per million cycles when running CPU bound application.

- rx2 = counter’s reading per million cycles when running memory bound application



Next Steps

- We continue developing the approach:
  - adding energy measurements to the existing set of experiments.
  - using more complex benchmarks with known but fluctuating behavior.
  - developing ML model

- Evaluation and comparison to related works

- Eventually, we will be glad to present the results in “Herbsttreffen 2017”!

- Beyond this step, if results are found promising we will delve into sophisticated techniques like “Reinforcement learning”.