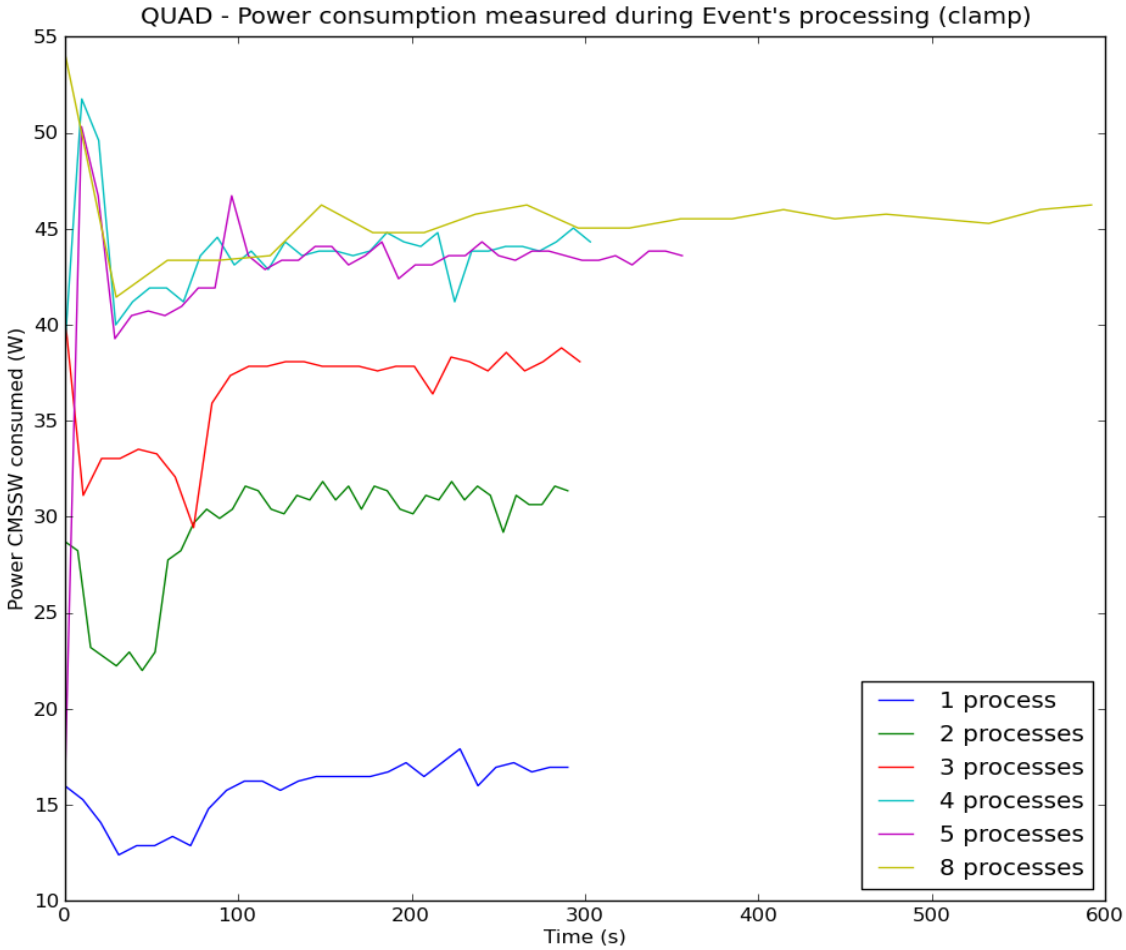
CMSSW - ARM vs Intel

on power consumption

1. ARM vs Intel
    - power consumed in event processing
    - comparison

2. Preliminary RAPL measurements


QUAD - Power consumption measured during Event's processing (clamp)
Power CMSSW consumed (W)
Time (s)
55
50
45
40
35
30
25
20
15
10
0
100
200
300
400
500
600
1 process
2 processes
3 processes
4 processes
5 processes
8 processes

4x Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz

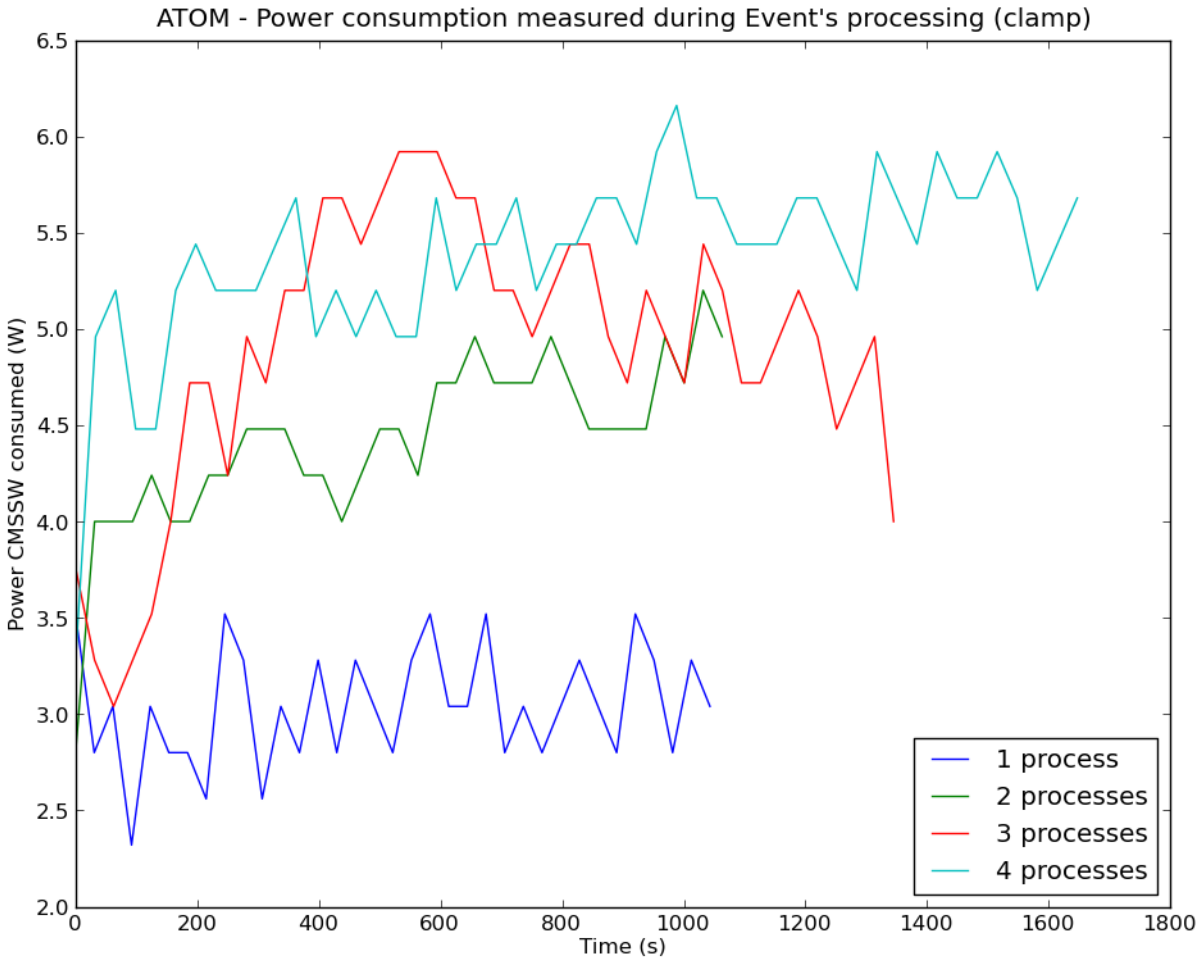
- the plots account for power consumption (W) during the Event processing stage.
- 1, 2, 3 and 4 processes take roughly the same time to process the events, as expected

- when there is a physical cpu overcommitment, the time difference is:

    5 processes: 22.75% > 1 process

    8 processes: 204% > 1 process

4x Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz
no hyperthreading
speed step technology
http://ark.intel.com/products/49490/Intel-Atom-processor-D525-1M-Cache-1_80-GHz


ATOM - Power consumption measured during Event's processing (clamp)
Power CMSSW consumed (W)
Time (s)
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
0
200
400
600
800
1000
1200
1400
1600
1800
1 process
2 processes
3 processes
4 processes

2x Intel Atom CPU D525 @ 1.8GHz

- the plots account for power consumption (W) during the Event processing stage.
- 1 and 2 processes take roughly the same time to process the events, as expected

- when there is a physical cpu overcommitment, the time difference is:

    3 processes: 29% > 1 process

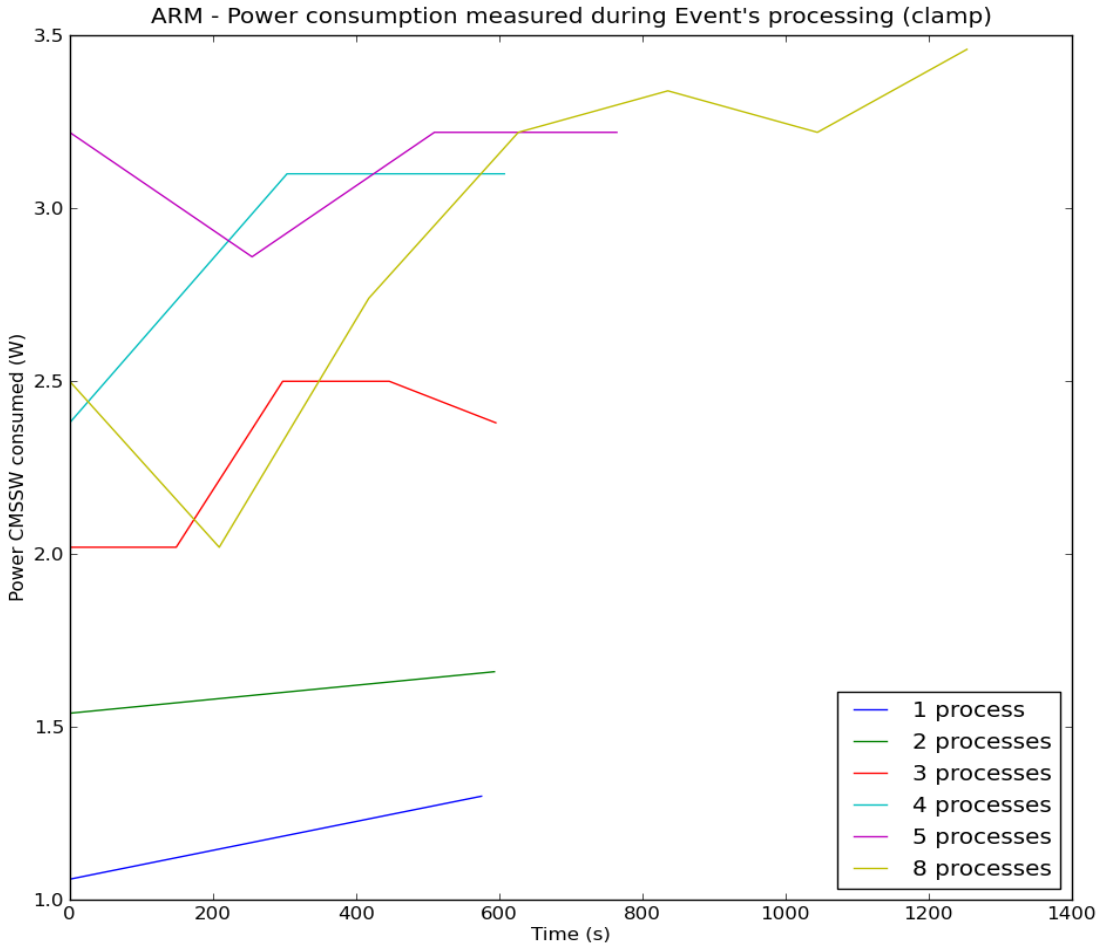
    4 processes: 58% > 1 process

- under cpu overcommit, the increase of time is relatively smaller than what happens in quad and ARM (maybe due to hyperthreading)

2x Intel Atom CPU D525 @ 1.8GHz

hyper-threading (2 processing threads per physical core)

http://ark.intel.com/products/49490/Intel-Atom-processor-D525-1M-Cache-1_80-GHz


ARM - Power consumption measured during Event's processing (clamp)
Power CMSSW consumed (W)
Time (s)
3.5
3.0
2.5
2.0
1.5
1.0
0
200
400
600
800
1000
1200
1400
1 process
2 processes
3 processes
4 processes
5 processes
8 processes

4x ARMv7-A Cortex 2.50 DMIPS/MHz

- the plots account for power consumption (W) during the Event processing stage.
- 1, 2, 3 and 4 processes take roughly the same time to process the events, as expected

- when there is a physical cpu overcommitment, the time difference is:

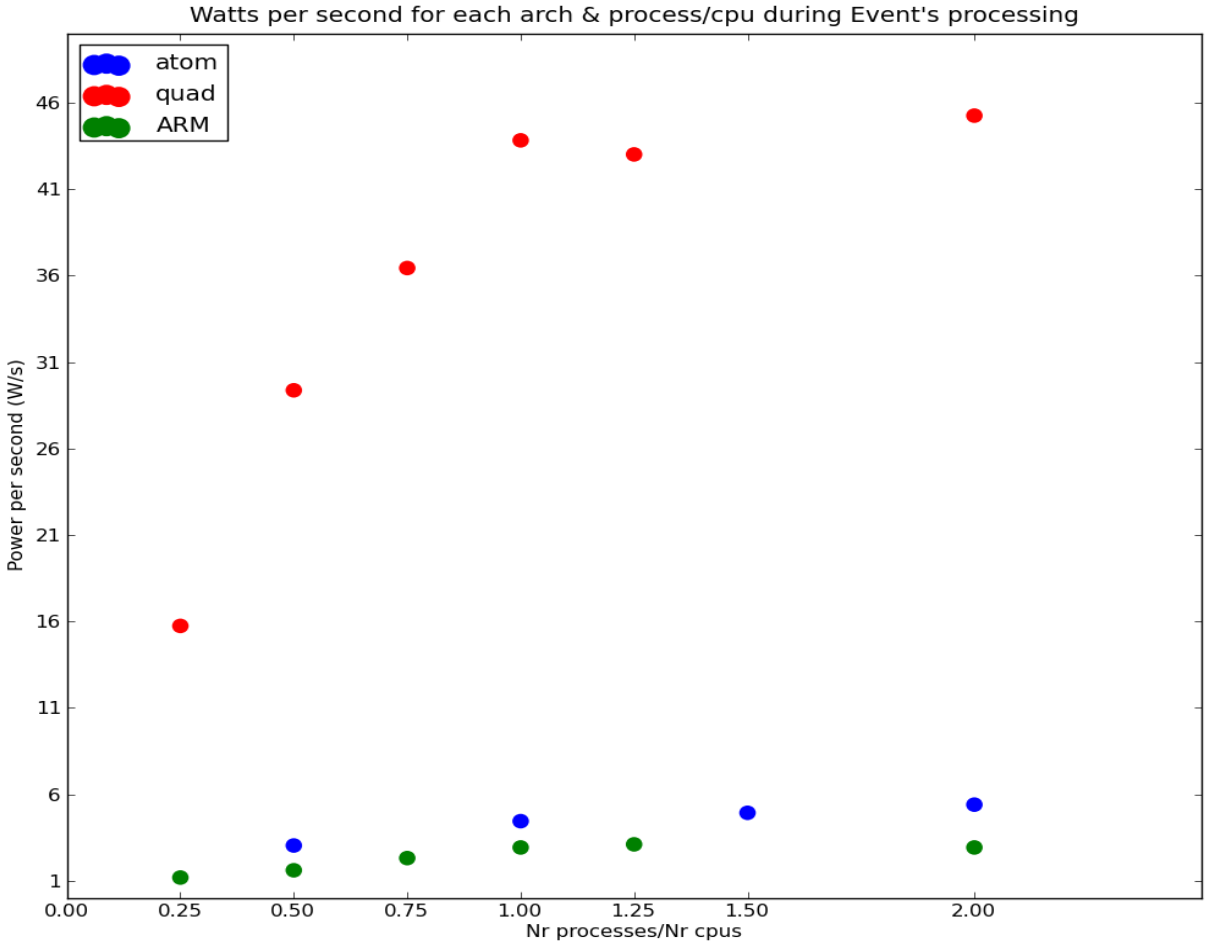
    5 processes: 132% > 1 process

    8 processes: 218% > 1 process

- plots don’t have many samples, since the Event’s processing stage is relatively small when compared with the startup phase.

4x ARMv7-A Cortex 2.50 DMIPS/MHz

http://www.arm.com/products/processors/cortex-a/cortex-a9.php


Watts per second for each arch & process/cpu during Event's processing
atom
quad
ARM
Power per second (W/s)
46
41
36
31
26
21
16
11
6
1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
2.00
Nr processes/Nr cpus

2. RAPL

running average power limit driver

- high precision power consumed in different domains: **socket**, **DRAM**, **processor**

- power consumption → better picture of how CMSSW behaves

machine specs *caesrv1003.cern.ch*

- Sandy Bridge machine, 4 CPUs total of 32 physical cores [2.7GHz, 20MB cache]

- Hyper-threading disabled

- 0.5TB of RAM

2. RAPL

bad performances if NUMA architecture is not taken into consideration at application level

→ What about energy efficiency ?

→ What about CMSSW ?

example

what happens if multithreading are scaled to more than 8 threads per socket ?

probably threads from same process running in different NUMA sockets will decrease significantly energy efficiency (due to latency, and others)

Some results

1. 16 processes

i) bound to socket #2 and #3 *(using numactl -N ..)*

ii) start processes normally

2. 32 processes

i) bound to socket #2 and #3 *(using numactl -N ..)*

ii) start processes normally

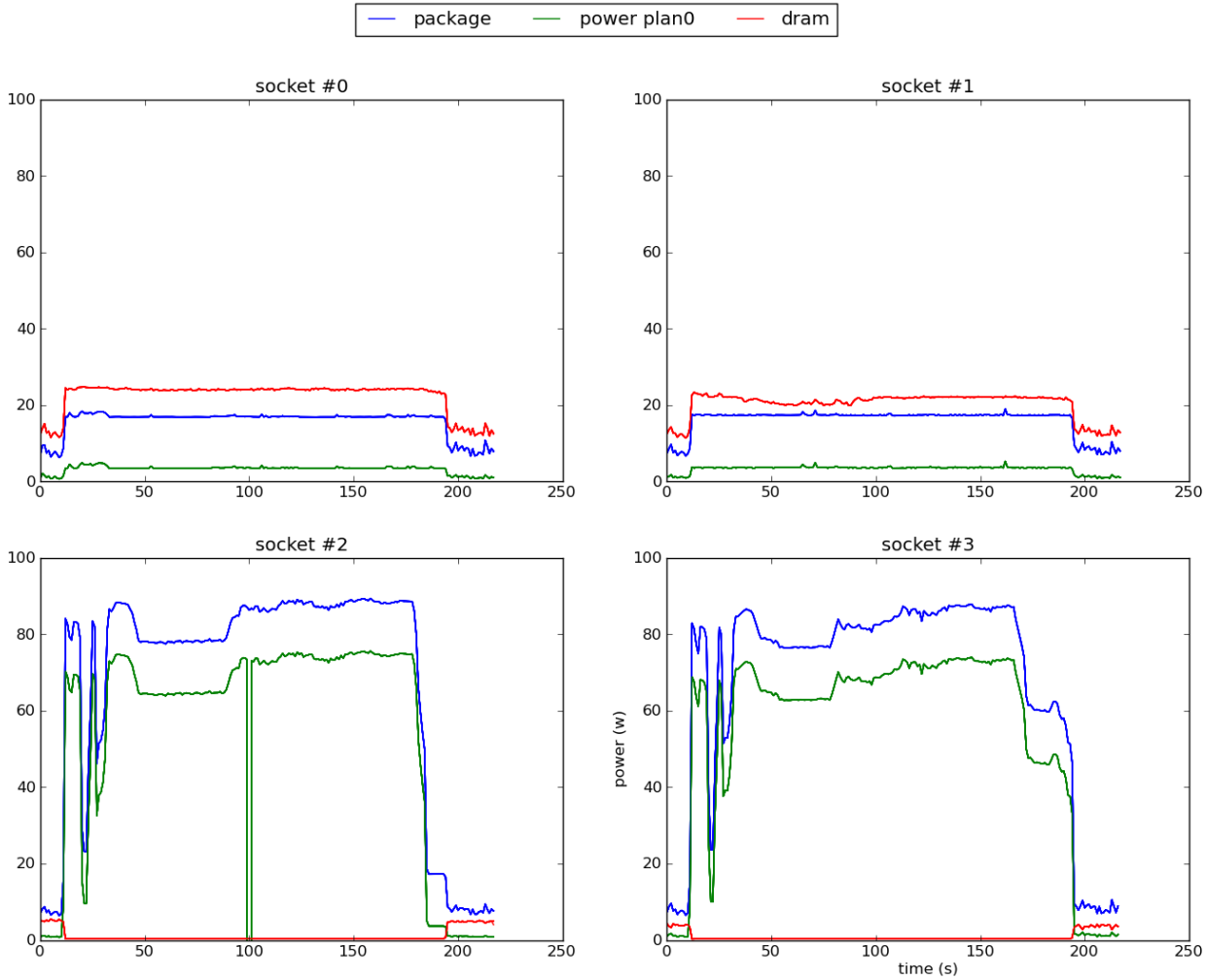
workflow: MinBias_8TeV_pythia8_cff_GEN_SIM.py, 10 Events, output off

16 processes bound to #2 and #3

- roughly 190s

- used DRAM is in #0 and #1

- kernel in #2 *(not confirmed)*



note: all processores are represented. though, they overlap per package

16 processes (no binding)

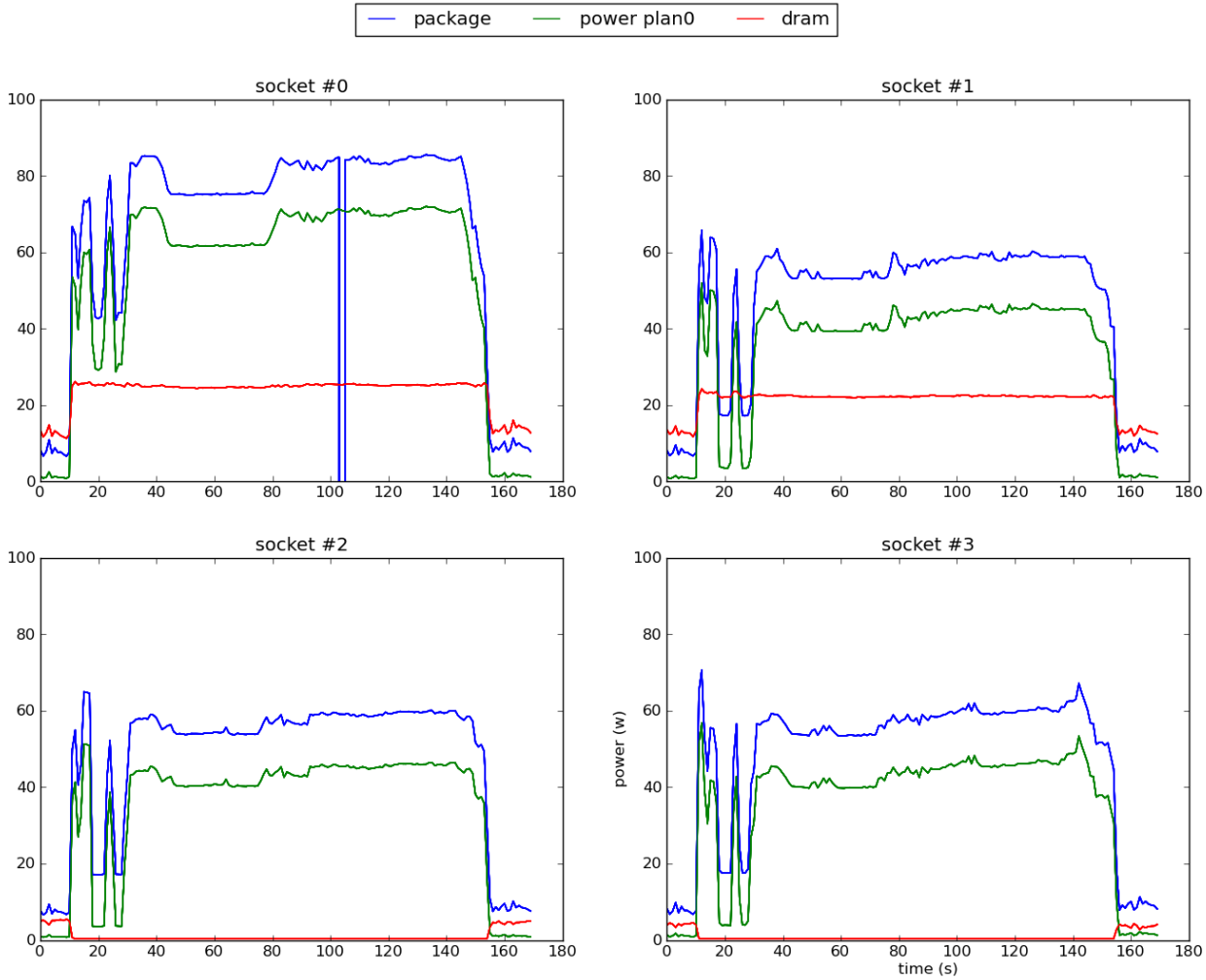
- roughly 150s

- used DRAM is in #0 and #1

- kernel in #0 *(not confirmed)*

- processes distributed evenly

- seems that when there is no explicit binding, the overall power efficiency is poorer
(probably a too far fetched conclusion, will confirm it this week)



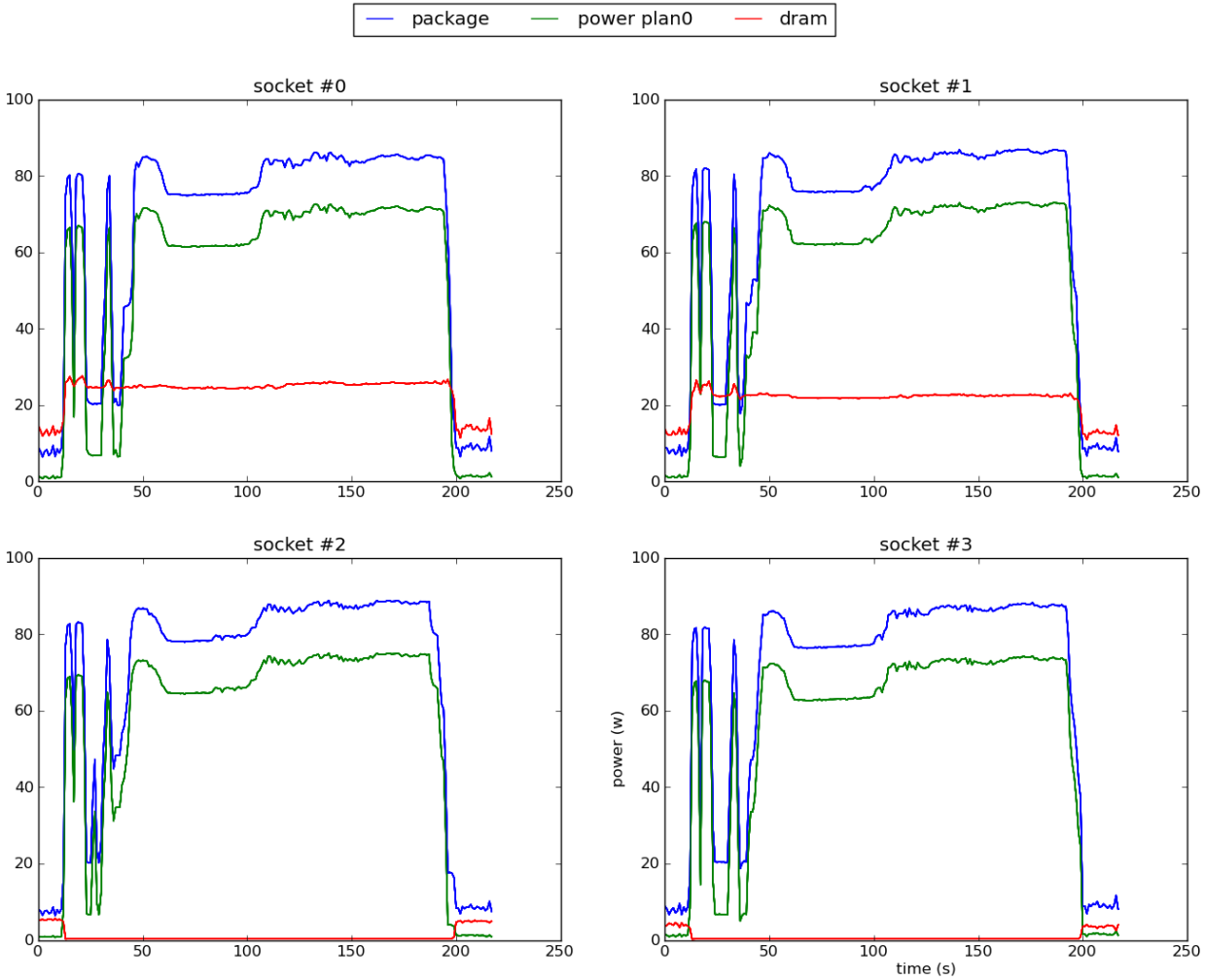
note: all processores are represented. though, they overlap per package

32 processes explicitly distributed with numactl (8 per socket)

- roughly 190s

- used DRAM is in #0 and #1

- not visually understandable where kernel is running



note: all processores are represented. though, they overlap per package

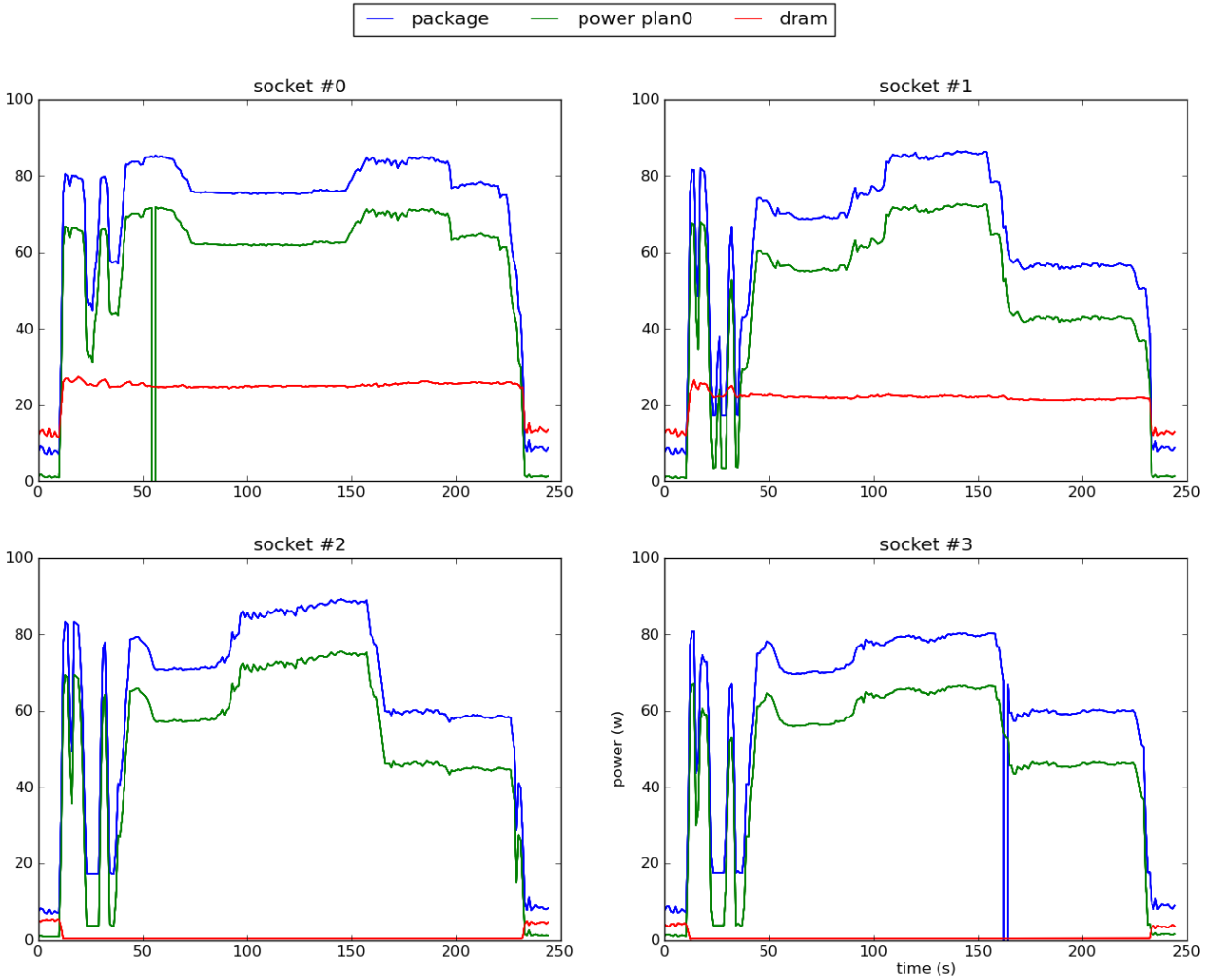
32 processes (no binding)

- roughly 240s

- used DRAM is in #0 and #1

- not visually understandable where kernel is running

- once again, different behaviour. will check power efficiency of both approaches and reasons



note: all processores are represented. though, they overlap per package