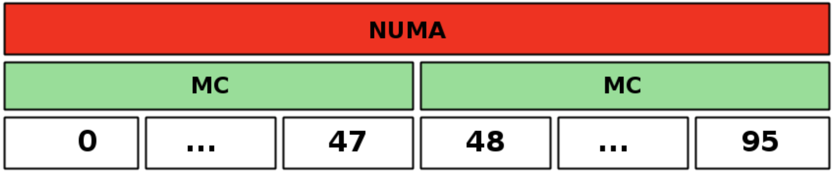
Fragmenting sched domains

Valentin Schneider <valentin.schneider@arm.com>

09/09/2019

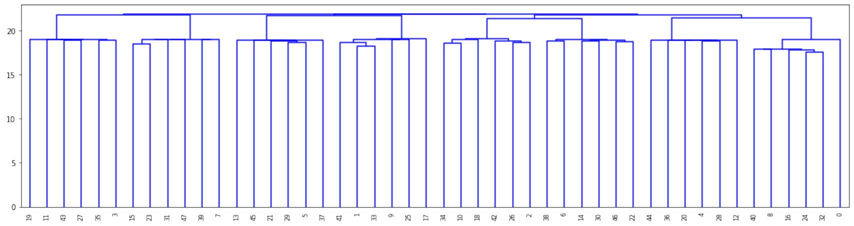
Some motivation

- Cavium ThunderX (first of its name)
- 2 sockets of 48 Armv8 CPUs
- Topology reported by devicetree



Some motivation (cont.)

- Topology quirk hinted at by some paper
- Verified with LMbench (memory bandwidth) + hierarchical clustering



- 8 ”socklets” of 6 CPUs per socket, each sharing memory bandwidth
  - [0, 8, 16, 24, 32, 40], [1, 9, 17, 25, 33, 41], …
  - `[[i + 8 * j for j in range(6)] for i in range(8)]`

”Improving” this specific scheduler topology

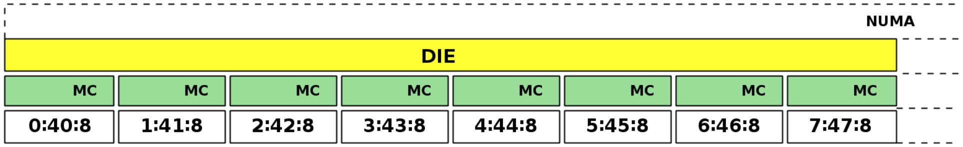


Figure: (a:b:s) == a to b (inclusive) with a stride of s

- Better hackbench results (can reach -25% runtime)
- Should give better memory bandwidth results when not overloaded

Food for thought

- Not necessarily reserved to ~~broken~~ special hardware
  - Smaller SD's -> faster wakeup scan (Re: latency-nice)
  - Does it really make sense to cram so many CPUs in the same domain mask?
  - Distances not always identical from one CPU to any other
  - (NoC / NUCA relevance?)

- Does the hardcoded (SMT)/MC/DIE distinction still make sense?

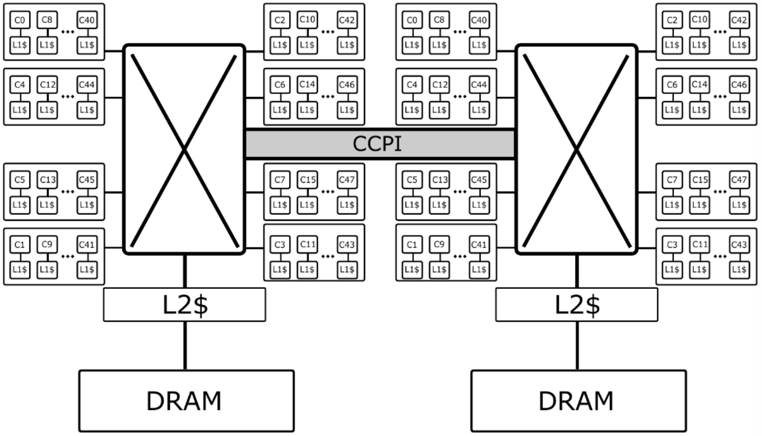
- NUMA domains are built dynamically, could we extend this to the lower domains?
  - Based on different metrics/distances
  - Doesn't have to be limited to 3 levels
  - Could improve some properties (e.g. `imbalance_pct` is currently hardcoded)

arm

Thank you!

Paper snippet

- Diagram from a paper by folks from PRACE (Partnership for Advanced Computing in Europe)
- Absolutely zero information about how they came up with it



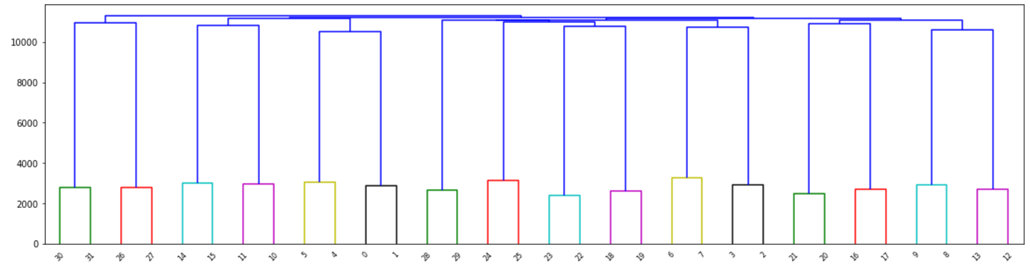
Profiling flow

- Leverage LMbench to get some memory bandwidth measurement
- Make sure the data doesn’t fit in the L1, so we stress the L2

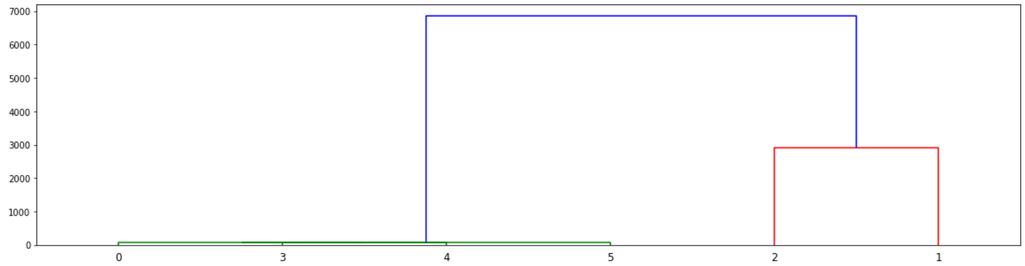
- Pin an lmbench thread on a CPU, get the score: that’s the baseline
- For each CPU base
  - For each CPU other != base
    - Pin an lmbench thread on base
    - Pin an lmbench thread on other

- If the bandwidth score of thread on base is lower than the baseline, then it was affected by other
- Add in some iterations for statistical relevance

Ampere Emag


10000
8000
6000
4000
2000
0
30
31
26
27
14
15
11
10
5
4
0
1
28
29
24
25
23
22
18
19
6
7
3
2
21
20
16
17
9
8
13
12

Juno R0


7000
6000
5000
4000
3000
2000
1000
0
0
3
4
5
2
1