UiO : USIT

Using the HPC resources effectively - 2

Using HPC resources effectively - Performance analysis

Ole W. Saastad, UiO/USIT/ITF/FI
March 2022

- Running programs
- Understanding basics of program execution
- Compile and optimise programs
- Using performance libraries

- Running programs
  - Scaling
  - Profiling
  - Memory
  - Storage

- Understanding basics of program execution

- Compile and optimise programs

- Using performance libraries

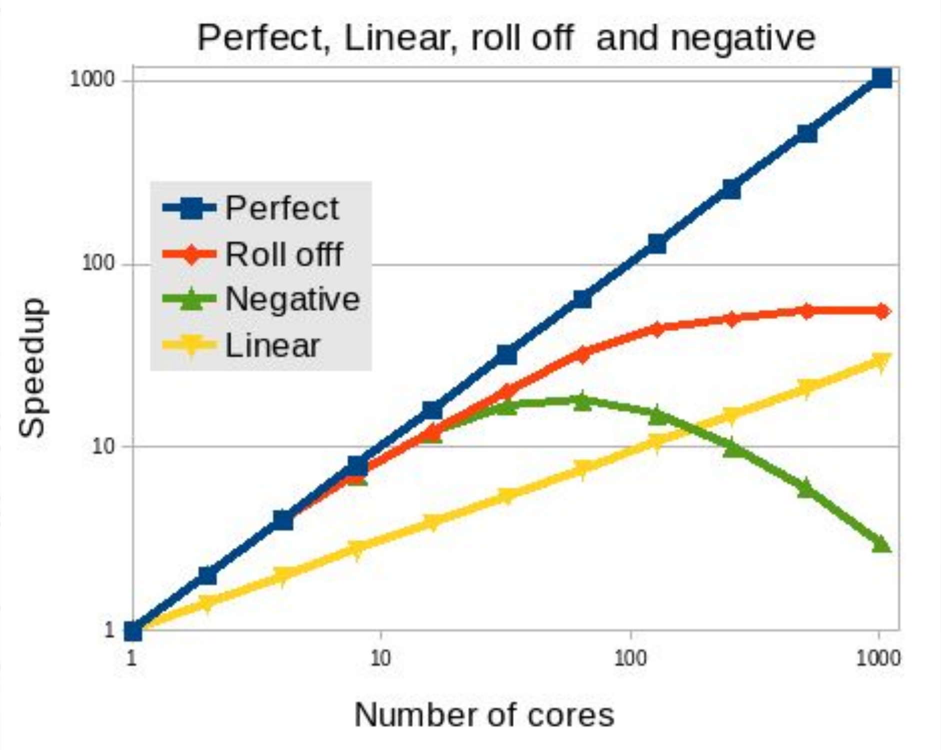
Scaling

SLURM:

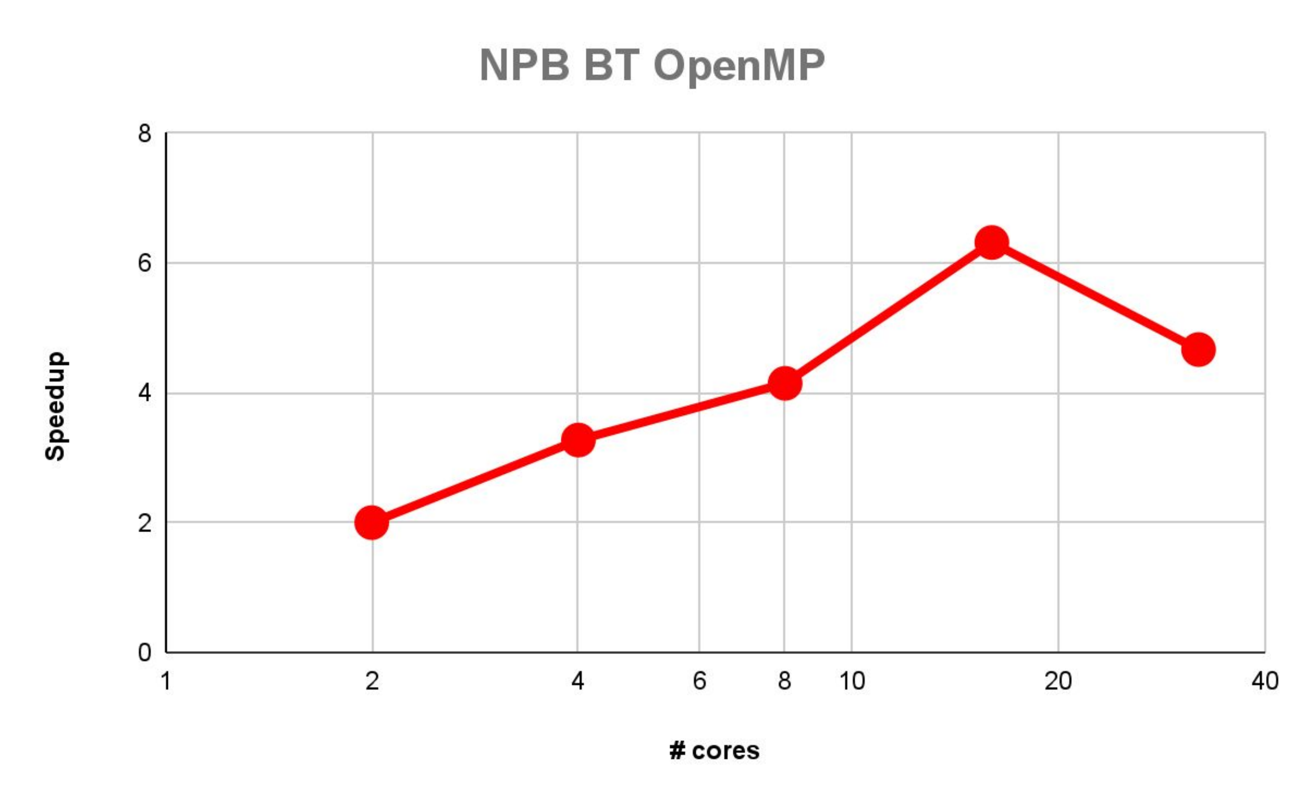
--nodes=N

--tasks-per-node=M

--cpus-per-task=K



#cores	Perf	Speedup
2	12744,8	2
4	20856,1	3,3
8	26422,6	4,1
16	40250,3	6,3
32	29738,0	4,7



Check scaling !

Do a scaling check of your application with relevant input.

Do not use more cores than needed.

Going from 128 to 256 cores with just a 10% speedup is a waste of 128 cores.

Check it again with new input.

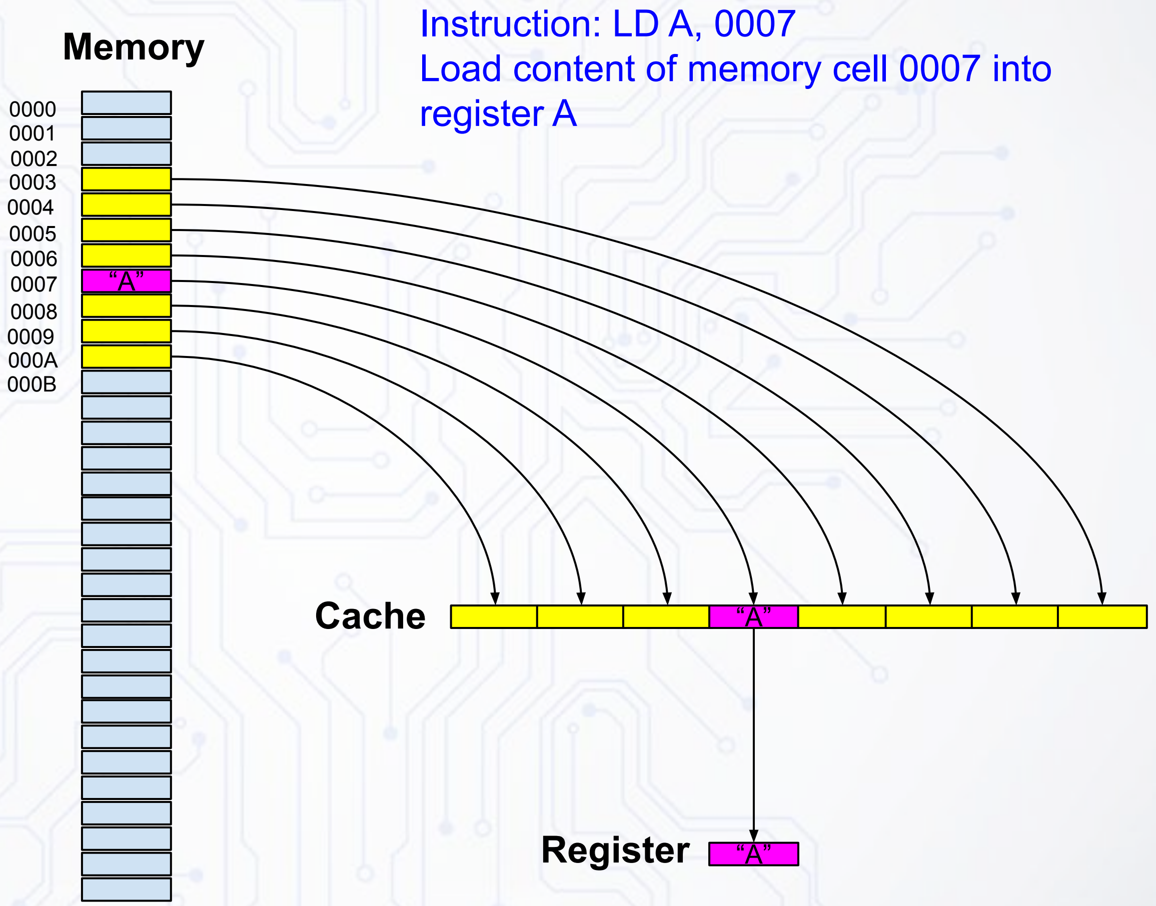
- Running programs
- Understanding basics of program execution
  - Memory, cache and cache coherence
  - Vectorisation
  - Ranks and threads
  - Storage
- Compile and optimise programs
- Using performance libraries

What is the smallest amount of data you can access from memory ?

Cache line is 64 byte wide
Requesting a short/char waste 63/64 (98%) of the memory bandwidth.

Requesting a double waste only ⅞ (87%).

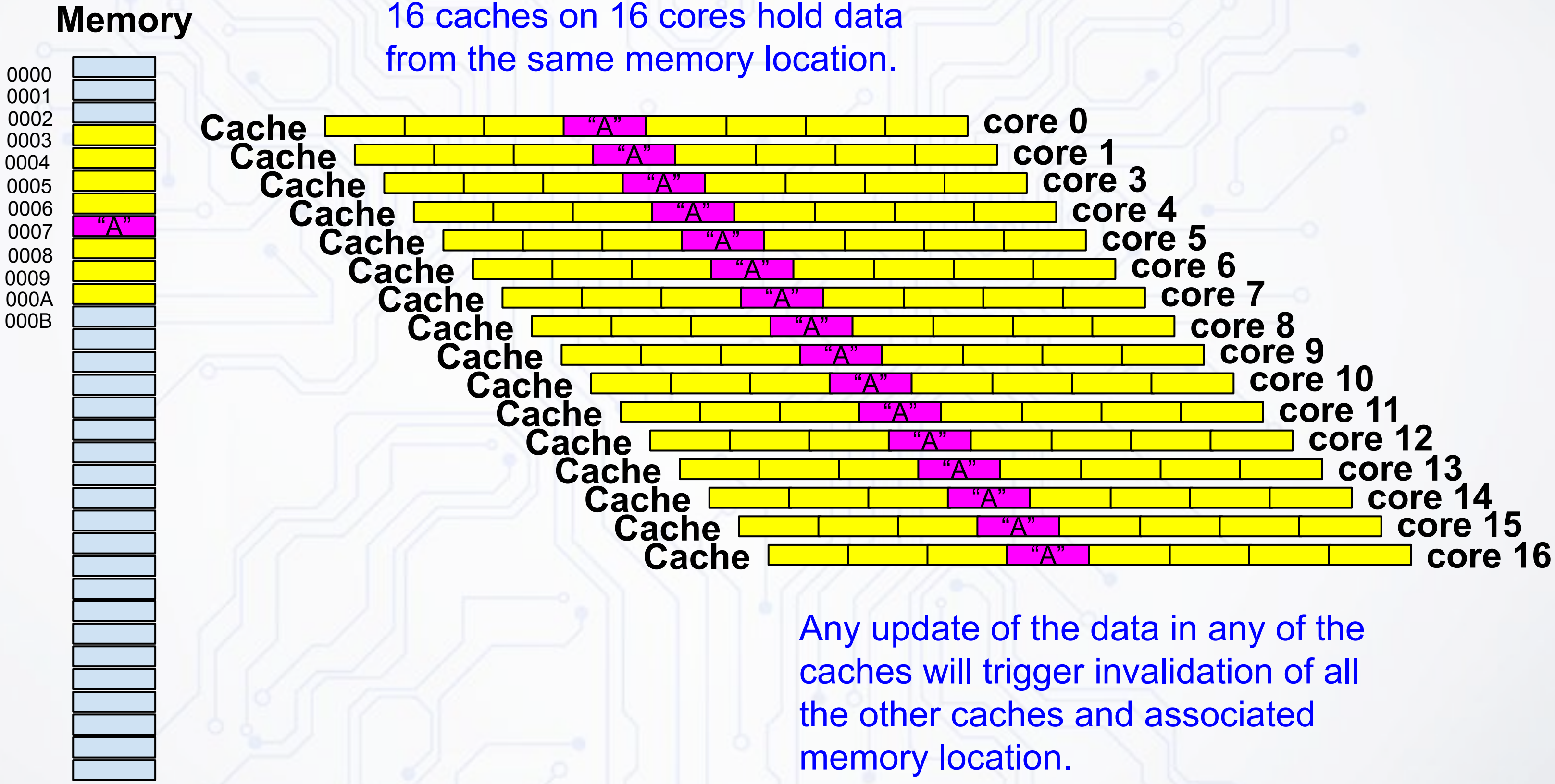
Using only a single byte of this memory request waste ⅞ of the memory bandwidth.



Cache Coherence (cc)

User can be sure that all memory locations are «updated» with the same data regardless of any other core or thread updates the location. Accessing a memory location will always yield the latest update to that memory location.

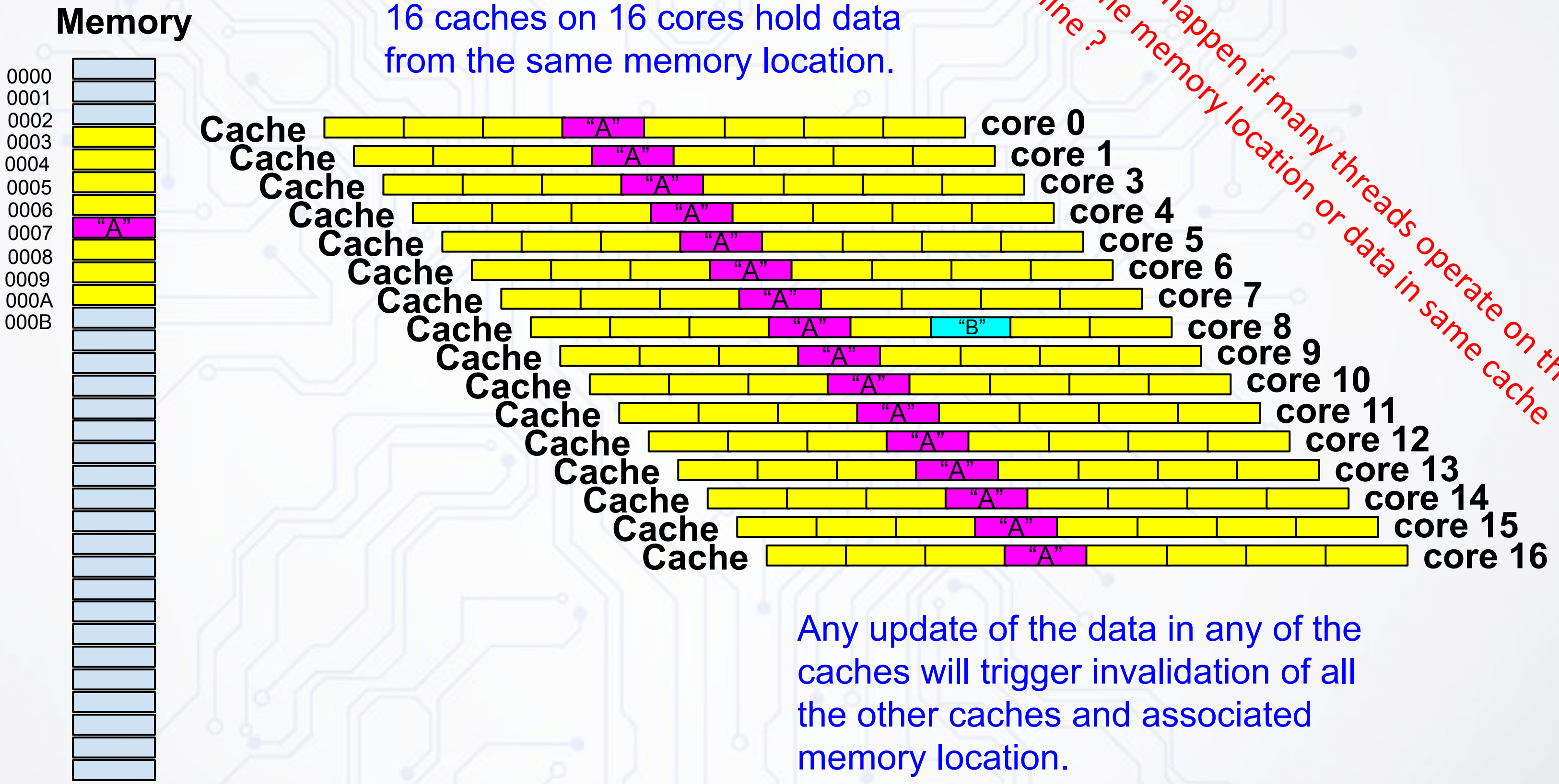
16 caches on 16 cores hold data from the same memory location.



Any update of the data in any of the caches will trigger invalidation of all the other caches and associated memory location.

16 caches on 16 cores hold data from the same memory location.

What happen if many threads operate on the same memory location or data in same cache line ?



Any update of the data in any of the caches will trigger invalidation of all the other caches and associated memory location.

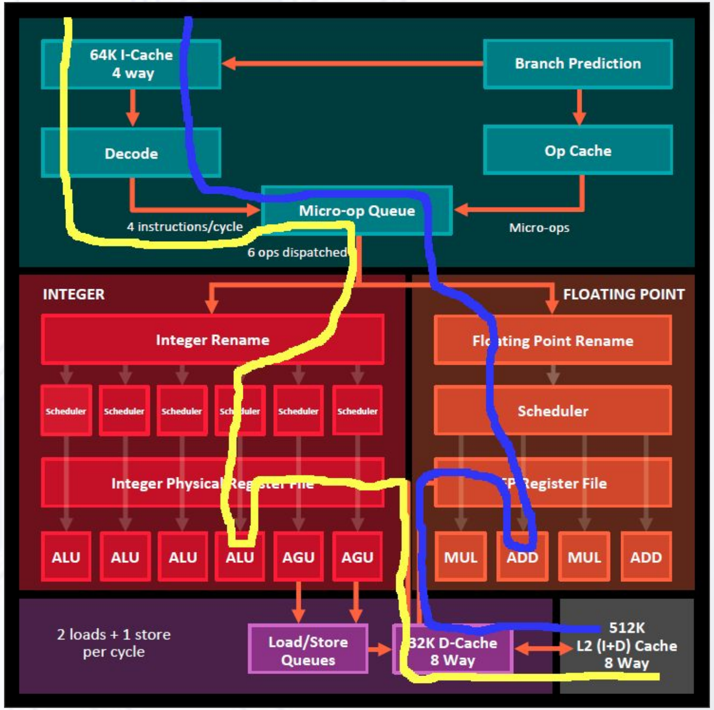
Cores, logical/virtual/physical

Each core contain many executional units, decoding,integer, floating, vector, load, store...

Each core can simultaneously run several instruction streams, x86-64 normally two (SMT2).

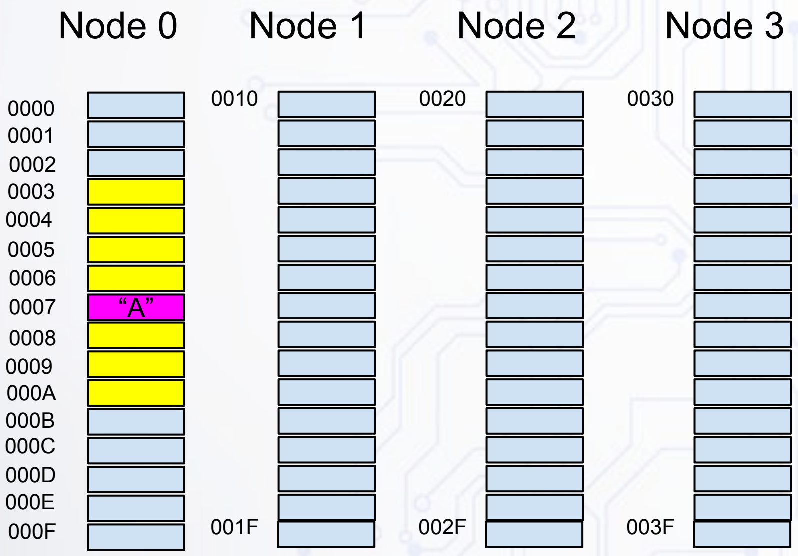
Seen from the OS it looks like two cores/cpus. Two instruction streams scheduled on the same core.

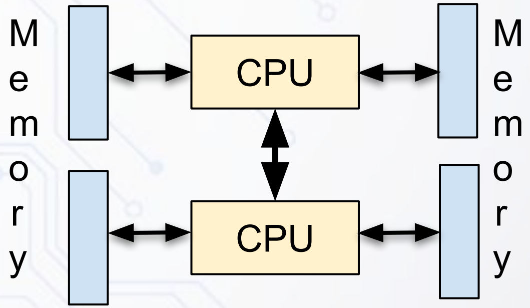
This feature can be enabled or disabled, for HPC it often disabled for performance reasons.



Non Uniform Memory Access (NUMA)

Memory divided into banks (nodes)





```
lscpu --all --extended=CPU,CORE,SOCKET,NODE
CPU CORE SOCKET NODE
0   0       0     0
1   1       0     0
..
64  64      1     4
65  65      1     4
..
128 0 0     0
129 1 0     0
..
192 64      1     4
193 65      1     4
..
255 127     1     7
```

```
numactl -H | grep cpus

node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

node 1 cpus: 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159

node 2 cpus: 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174
175

node 3 cpus: 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
191

node 4 cpus: 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
207

node 5 cpus: 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222
223

node 6 cpus: 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 224 225 226 227 228 229 230 231 232 233 234 235
236 237 238 239

node 7 cpus: 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 240 241 242 243 244 245 246 247 248 249 250
251 252 253 254 255
```

Socket 0 : NUMA node 0-3, socket 1 NUMA node 4-7

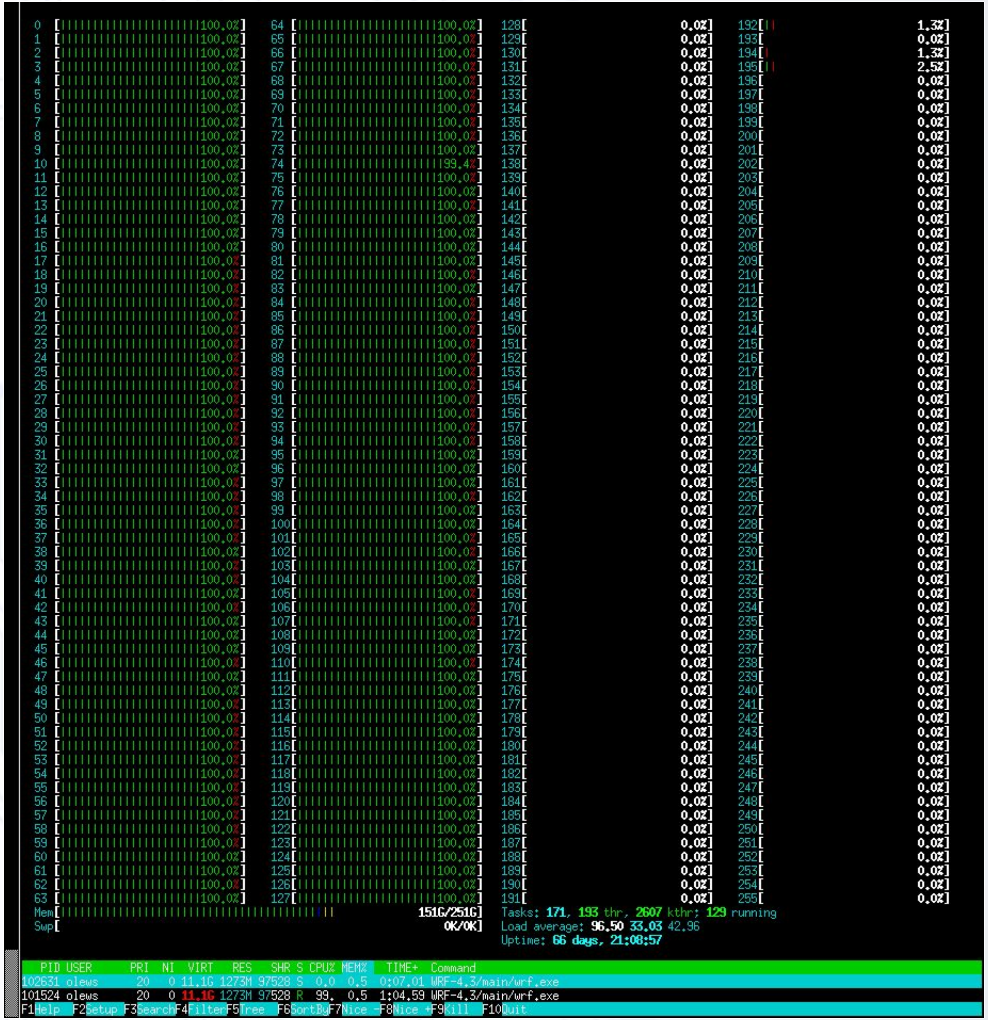
We observe that the 128 cores 1-127 includes 64 from socket 0 and 64 from socket 1.

Distribution of ranks on cores

Ranks distributed on both sockets, socket 0: 0-63,128-192 while socket 1:64-127,192-255.

This placement utilise both sockets as 0-63 is socket 0 and 64-127 is socket 1.

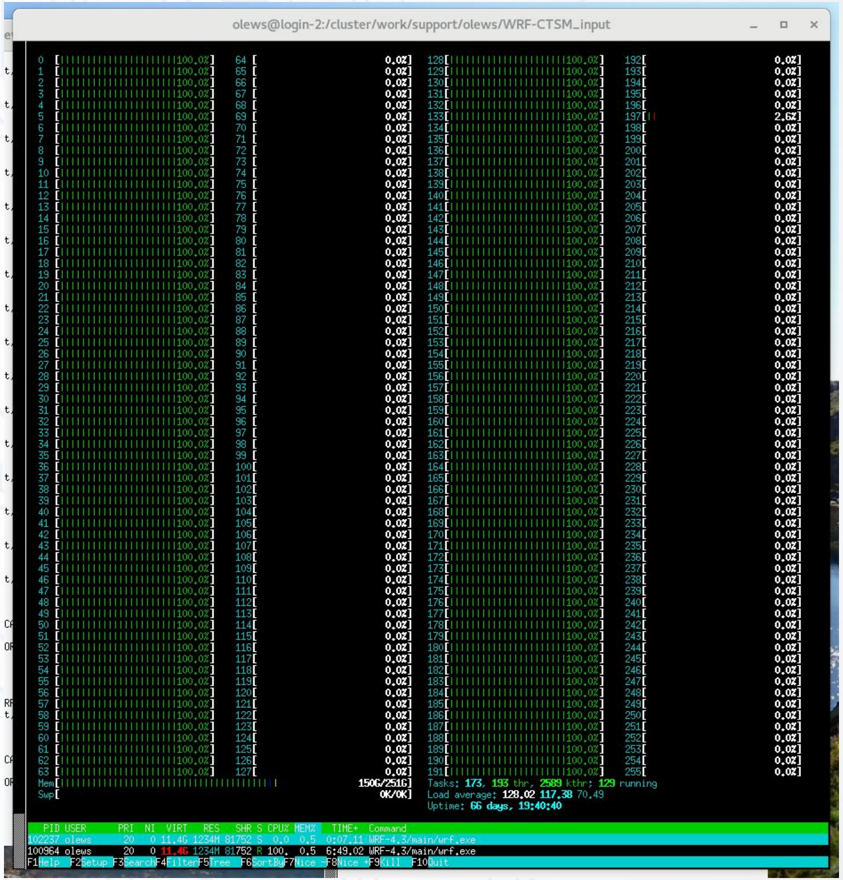
Full performance is achieved.



Distribution of ranks on cores

All ranks on a single socket, 0-63 and 128-192. Effectively using only one processor. Only ½ of the cores used as 2 threads per core is used.

Inferior performance is experienced



numactl -H| grep free
node 0 free: 405 MB
node 1 free: 280 MB
node 2 free: 301 MB
node 3 free: 282 MB
node 4 free: 24628 MB
node 5 free: 25787 MB
node 6 free: 25410 MB
node 7 free: 25191 MB

numactl -H| grep free
node 0 free: 290 MB
node 1 free: 13876 MB
node 2 free: 14053 MB
node 3 free: 13704 MB
node 4 free: 13754 MB
node 5 free: 13740 MB
node 6 free: 13784 MB
node 7 free: 13830 MB

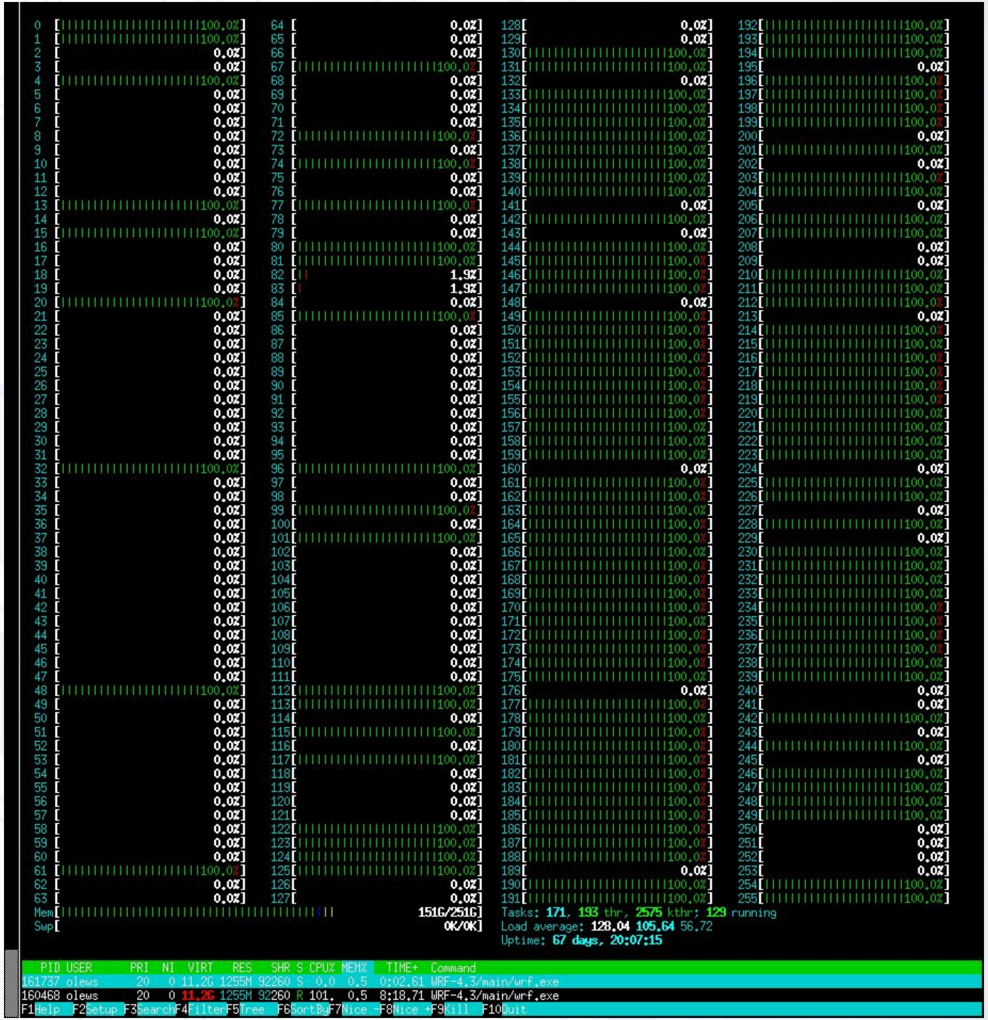
Default OS distribution of ranks on cores

Ranks distributed on both sockets, socket 0: 0-63,128-192 while socket 1:64-127,192-255.

OS actually distribute nicely on both cores.

Message to take home is that the defaults for Intel MPI are OK.

For OpenMP see earlier slides.



Storage during execution

Are all your input data available ?

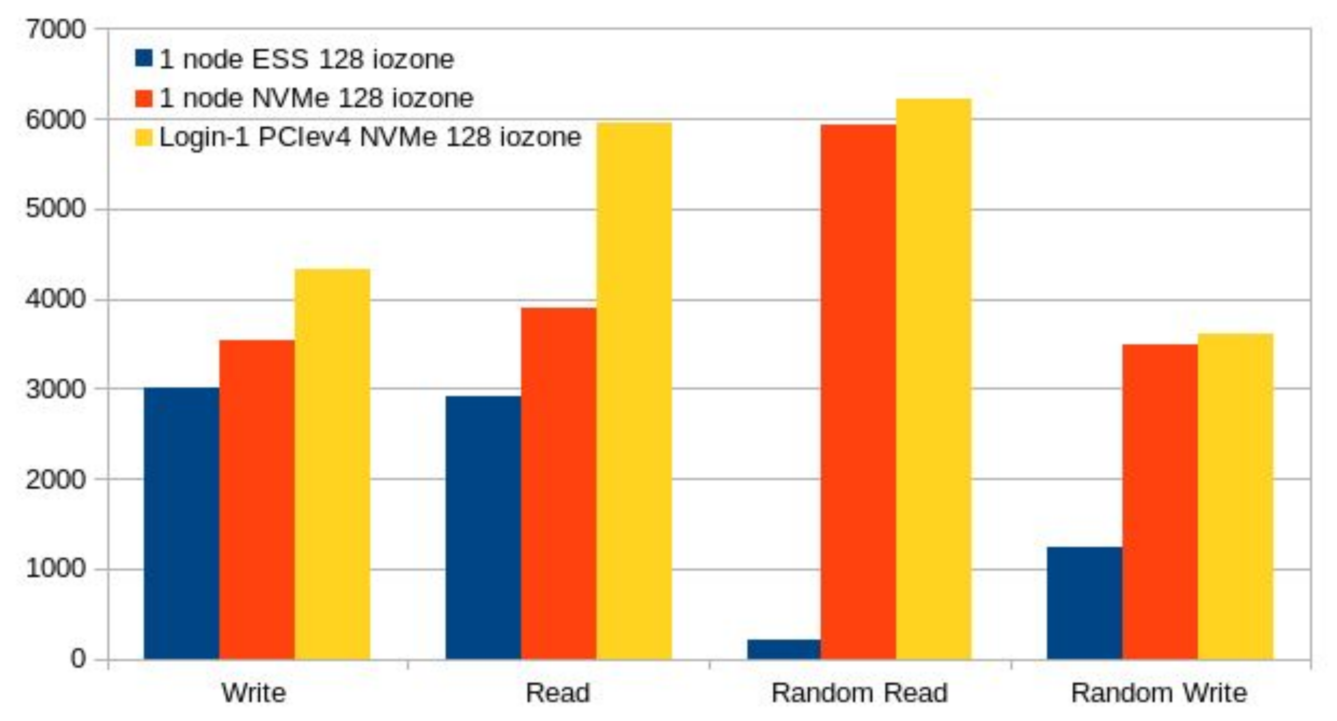
How do you store the output data ?

What about scratch data during a run ?

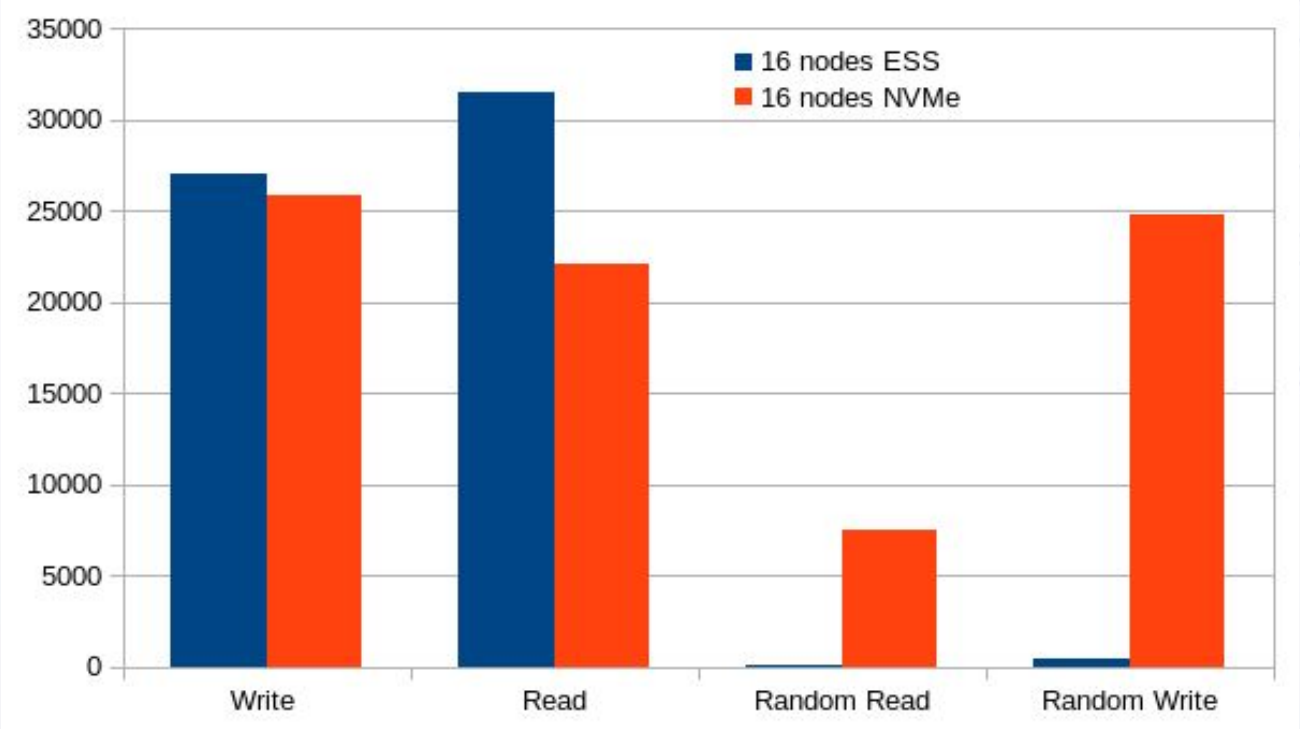
Have you consider the access of data during a run ?

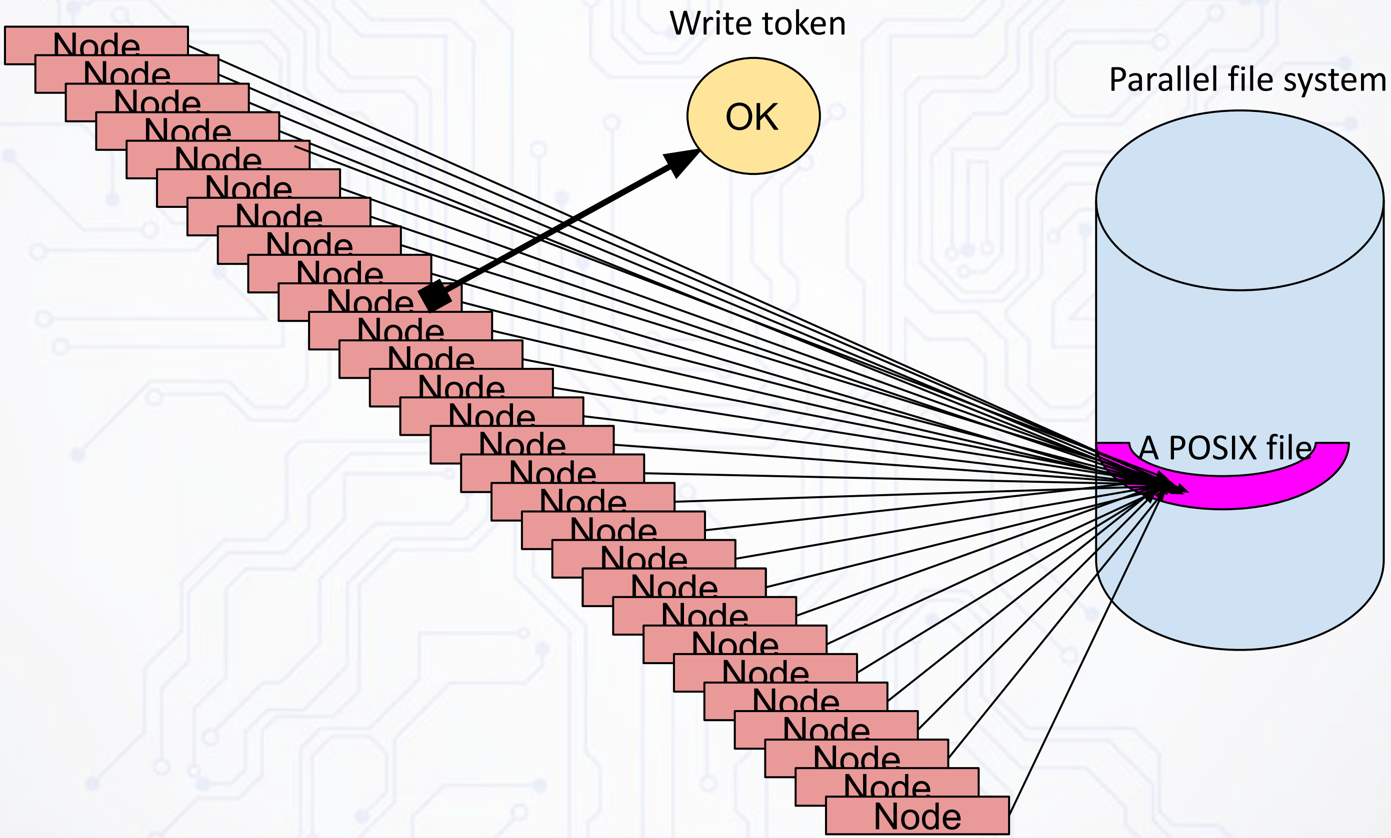
Sequential or random access ? At what size ?

Storage performance, parallel file system vs. local storage



Storage performance, parallel file system vs. local storage




Write token
OK
Parallel file system
A POSIX file
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node

Hands on time!

Scaling
Storage - I/O

Fox - Scaling assessment

Move to an interactive node,

copy the material :

cd to one of these :

NPB/NPB3.4-OMP and NPB/NPB3.4-MPI

Follow instructions in the file «Instructions»

We want to check and calculate scaling

Have fun

Fox - IO assessment

Move to an interactive node,

copy the material :

cd to : IO

Follow instructions in the file «Instructions»

Try out different storages and access patterns.

Have fun

/projects/ec34/inf9380/Exercises-23.Mar/