High Performance Co*mputing* / *ding* (HPC)

Alberto Dassatti - 2023

Performance

an hard definition... for us a measurable quanty of interest that we can exploit for comparisons and that has some connection with a figure of merit of a system under test.

Many Performances

- Development Time
- Resources requested
- Resources utilization
- Market Shares
- Maintainability
- ...

end very generally applicable

- Cost

HPC Interesting performances

All these performances are Computer Architecture metrics used to compare different machines, here we will use them in a different way: given one machine, we will compare different codes solving the same problem and we will derive from these numbers knowledge and optimization opportunities.

- Instruction Level Parallelism
- Cache behaviour
- Latency
- Throughput
- Power

Warning: they are not independent! Trade-offs will became your daily job.

Instruction Level Parallelism

Theory: is a measure of how many of the operations in a computer program can be performed simultaneously
Practice: is a measure of how many of the operations in a computer program are performed simultaneously

```
  2.066862   task-clock (msec)         #  0.746 CPUs utilized
         7   context-switches          #  0.003 M/sec
         0   cpu-migrations            #  0.000 K/sec
       105   page-faults               #  0.051 M/sec
 1,825,561   cycles                    #  0.883 GHz
 1,167,832   stalled-cycles-frontend   #  63.97% frontend cycles idle
         0   stalled-cycles-backend    #  0.00% backend  cycles idle
 1,424,315   instructions              #  0.78  insns per cycle
                                       #  0.82  stalled cycles per insn
   294,760   branches                  #  142.612 M/sec
    13,046   branch-misses             #  4.43% of all branches
```

Instruction Level Parallelism

Theory: is a measure of how many of the operations in a computer program can be performed simultaneously
Practice: is a measure of how many of the operations in a computer program are performed simultaneously

```
  2.066862   task-clock (msec)         #  0.746 CPUs utilized
         7   context-switches          #  0.003 M/sec
         0   cpu-migrations            #  0.000 K/sec
       105   page-faults               #  0.051 M/sec
 1,825,561   cycles                    #  0.883 GHz
 1,167,832   stalled-cycles-frontend   #  63.97% frontend cycles idle
         0   stalled-cycles-backend    #  0.00% backend  cycles idle
 1,424,315   instructions              #  0.78  insns per cycle
                                       #  0.82  stalled cycles per insn
   294,760   branches                  #  142.612 M/sec
    13,046   branch-misses             #  4.43% of all branches
```

Cache

The hit/miss ratio is very interesting.
We still have no information about what kind of miss we are facing.

```
    28,973  L1-dcache-load-misses
     9,457  L1-dcache-store-misses
     6,106  L1-dcache-prefetch-misses
    31,855  L1-icache-load-misses
 1,387,200  instructions              # 0.81  insns per cycle
    27,840  cache-references
     7,740  cache-misses              # 27.802 % of all cache refs
 1,720,732  cycles
```

Cache

The hit/miss ratio is very interesting.
We still have no information about what kind of miss we are facing.

```
    28,973  L1-dcache-load-misses
     9,457  L1-dcache-store-misses
     6,106  L1-dcache-prefetch-misses
    31,855  L1-icache-load-misses
 1,387,200  instructions              # 0.81  insns per cycle
    27,840  cache-references
     7,740  cache-misses              # 27.802 % of all cache refs
 1,720,732  cycles
```

Latency

Time needed to complete a task.
How to reduce it:

- change architecture
- change algorithm/implementation

Throughput

Number ot task done in parallel.
How to increase it:

- parallel execution (Pipeline, ILP, Multi-core, etc)
- it has specific obstacles we will address later in this class

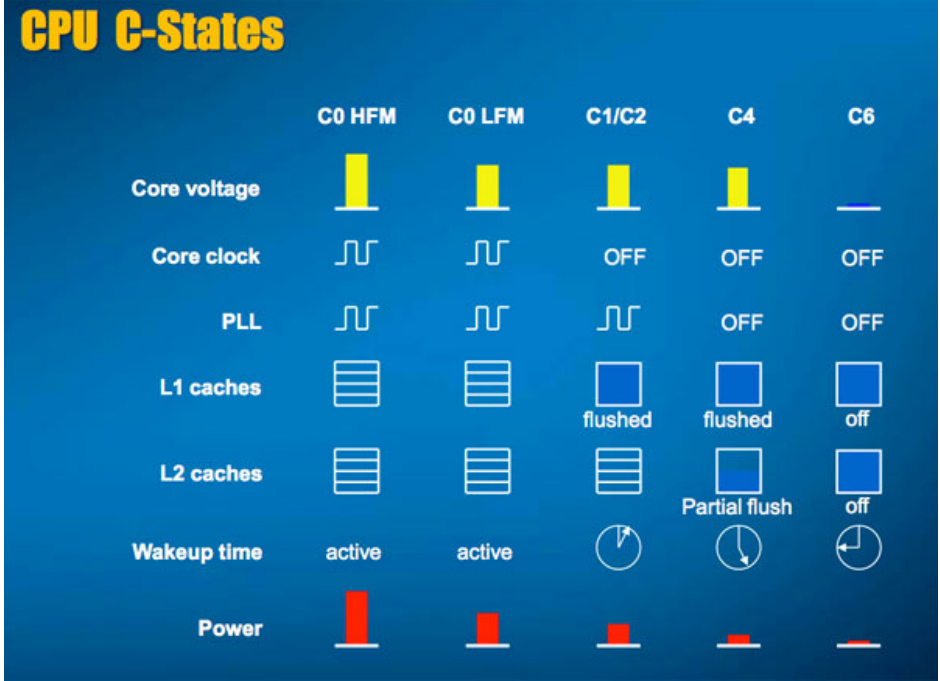
POWER

Reducing power consumtion is the real motivation of HPC. Why?

Power

Reducing power consumtion is the real motivation of HPC. Why?

$$P = \frac{1}{2} C V^2 f + P_{leak}$$


CPU C-States
C0 HFM
C0 LFM
C1/C2
C4
C6
Core voltage
Core clock
OFF
OFF
OFF
PLL
OFF
OFF
L1 caches
flushed
flushed
off
L2 caches
Partial flush
off
Wakeup time
active
active
Power

Where is the limit

memory bound

The task execution is limited by the amount or access time to the memory

WHERE IS THE LIMIT

memory bound

The task execution is limited by the amount or access time to the memory

CPU bound

The task execution is limited by the nomber of instructions the system performs per second

WHERE IS THE LIMIT

memory bound

The task execution is limited by the amount or access time to the memory

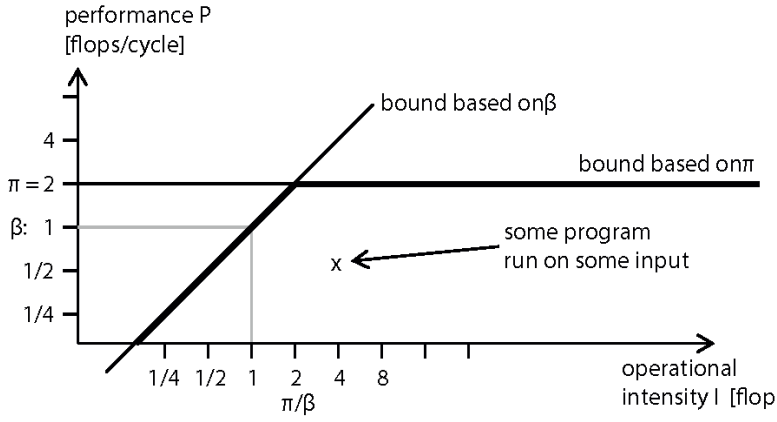
CPU bound

The task execution is limited by the nomber of instructions the system performs per second

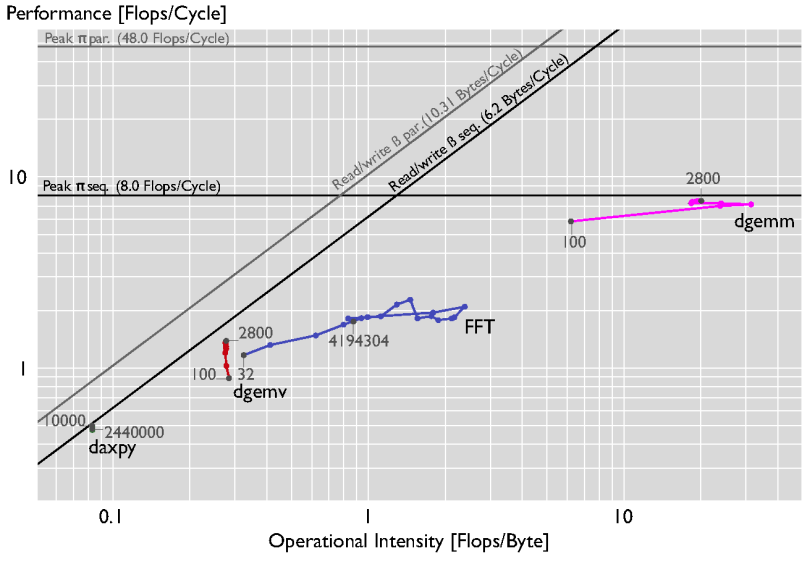
I/O Bound

The task execution is limited by the amount or access time to external devices

Roofline Model



source


Performance [Flops/Cycle]
Peak π par. (48.0 Flops/Cycle)
Read/write β par.(10.31 Bytes/Cycle)
Read/write β seq. (6.2 Bytes/Cycle)
10
Peak π seq. (8.0 Flops/Cycle)
2800
dgemm
100
2800
4194304
FFT
1
100
32
dgemv
10000
2440000
daxpy
0.1
1
10
Operational Intensity [Flops/Byte]

source

Questions