Performance Evaluation

Computadors

Grau en Ciència i Enginyeria de Dades

Xavier Verdú, Xavier Martorell

Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC)

2019-2020 Q2

Creative Commons License

Guia Docent

- Tecniques bàsiques d'anàlisi del rendiment
- Rendiment de les aplicacions, mètriques, obtenció de la informació, performance counters, rellotges d'alta precisió. Càlcul del rendiment, GFlops, bandwidth

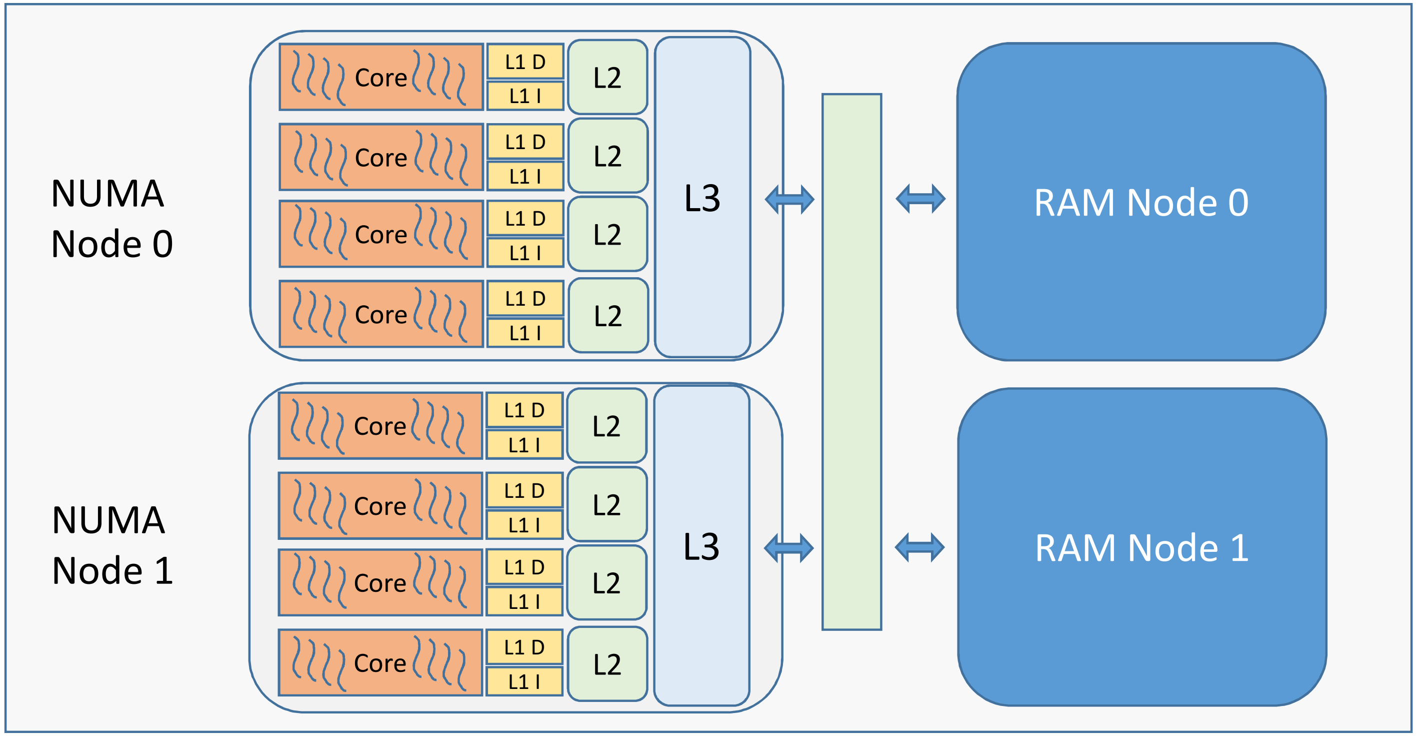
- Conèixer i saber utilitzar les tècniques bàsiques d'anàlisi del rendiment
    - Related competences: CT5, CG2, CB2,
    - Subcompetences:
        - Saber analitzar el rendiment del computador: processador, memòria, comunicacions i subsistema d'emmagatzematge

Table of Contents

- Introduction
- Performance metrics
- OS support services
- System tools
  - System calls
  - Commands
  - Global system information
  - Detailed hardware information

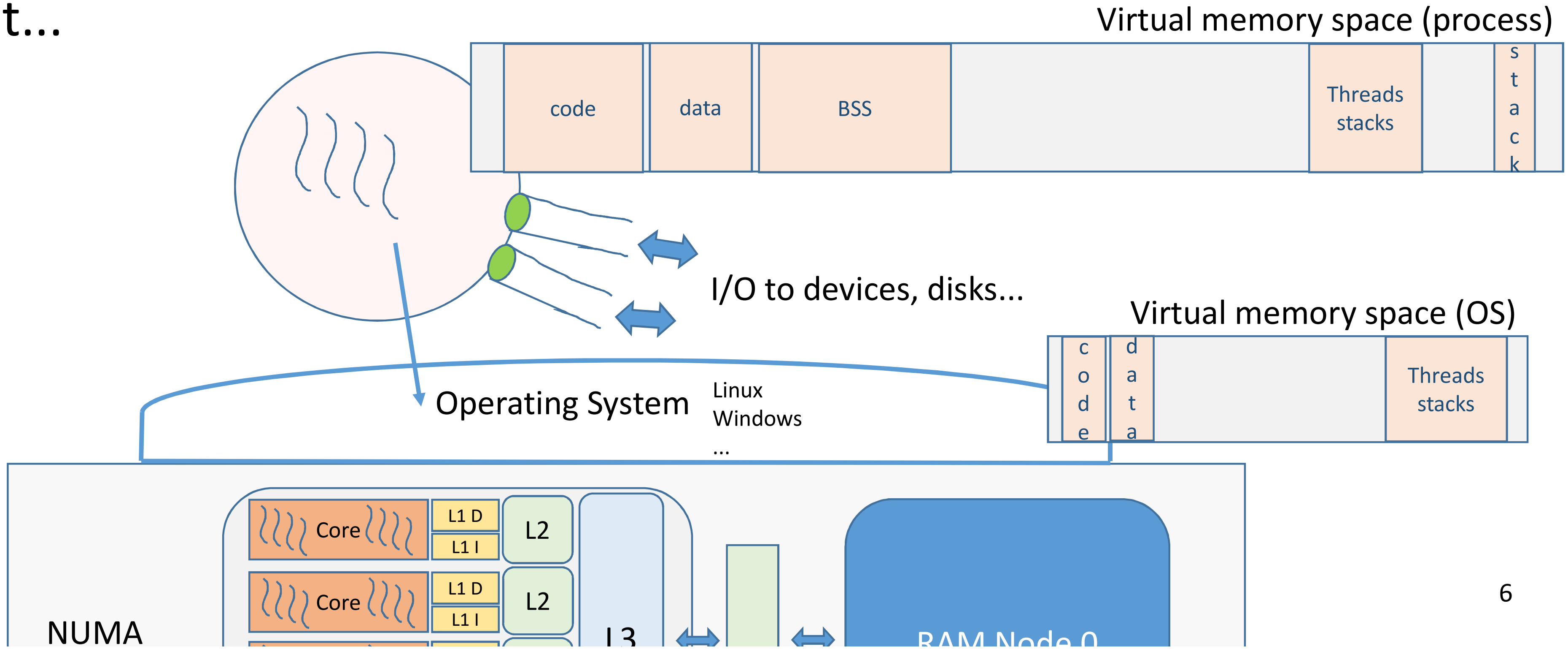
Performance evaluation, why?

- Architecture evolves, and we need to keep track of the improvements
- What are the benefits that applications get out of the architecture



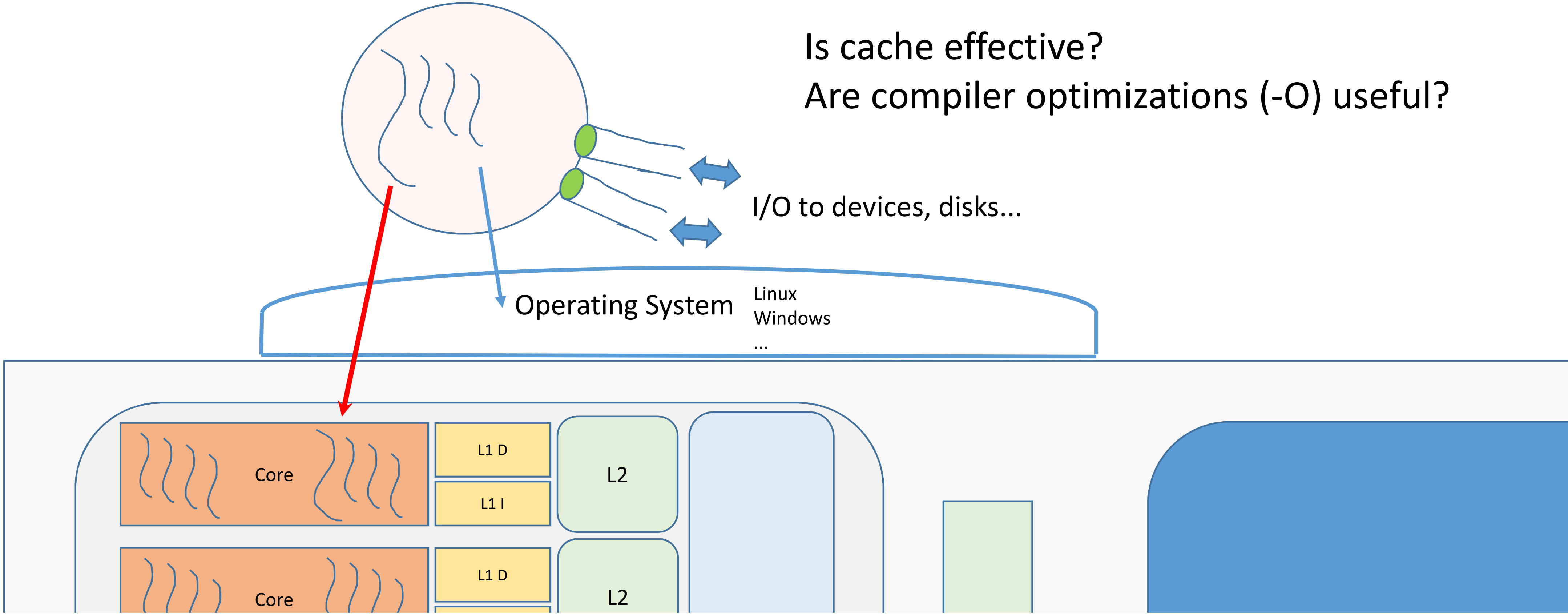
Performance evaluation, why?

- Operating system, software, applications, algorithms & compilers evolve, and we are interested in knowing if they are improving or not...



Thread performance

- What's the rate of instructions executed per second?



Thread performance sample

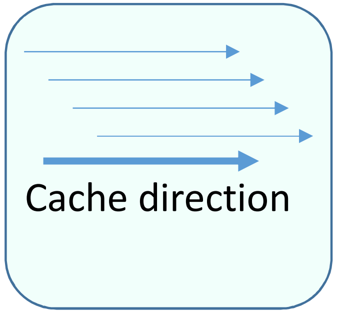
- Matrix initialization

```
#include <stdio.h>
#include <sys/time.h>
#include <malloc.h>
#include <cmath>

#define SIZE 10240
#define TYPE float

int main(int argc, char * argv [])
{
   TYPE * mat = (TYPE *)
       malloc(SIZE*SIZE*sizeof(TYPE));
   int loop, res, i, j;
   struct timeval tv0, tv1;
   if (mat == NULL) {
      perror ("malloc"); return 1;
   }
```

```
// Regular access
#define ACCESS(i,j) (((i)*SIZE)+(j))
// Transposed matrix
//#define ACCESS(i,j) (((j)*SIZE)+(i))

 for (loop = 0; loop < 3; loop++) {
   res = gettimeofday(&tv0, NULL);
   if (res < 0) perror("gettimeofday");
   for (i=0; i < SIZE; i++) {
     for (j=0; j < SIZE; j++) {
       mat[ACCESS(i,j)] = (TYPE) (loop+i*j);
     }
   }
   res = gettimeofday(&tv1, NULL);
   if (res < 0) perror("gettimeofday");
   fprintf (stderr, "ini: %lf us\n",
      tv1.tv_sec*1000000.0 + tv1.tv_usec -
      tv0.tv_sec*1000000.0 - tv0.tv_usec);
 }
 return 0;
}
```

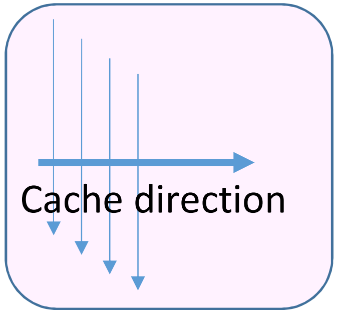
Thread performance sample

- ACCESS(i,j)



Row-wise access

ACCESS(j,i)

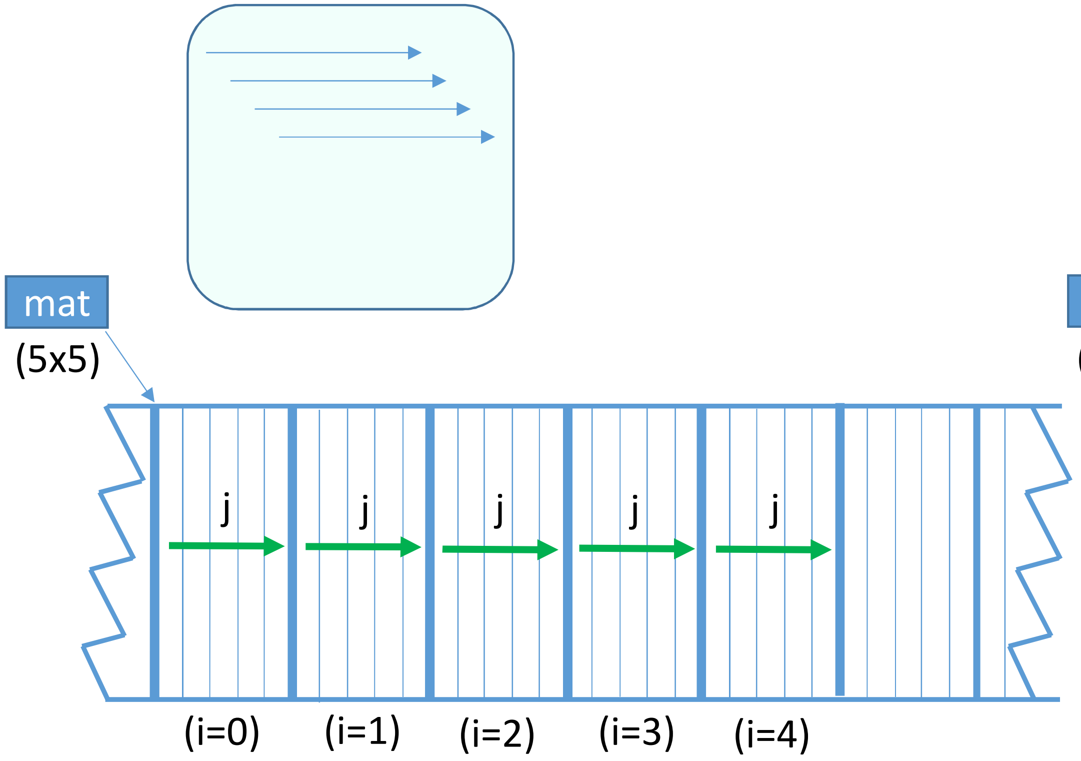


Column-wise access

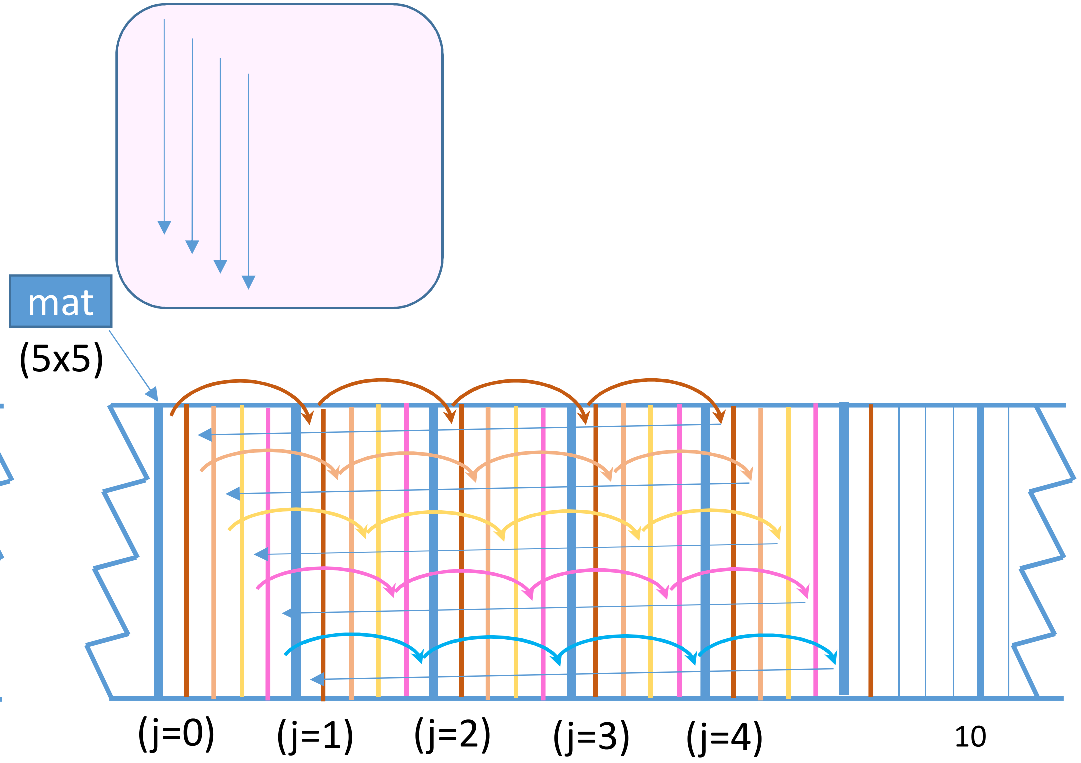
Matrix transposed

Thread performance sample

- ACCESS(i,j)

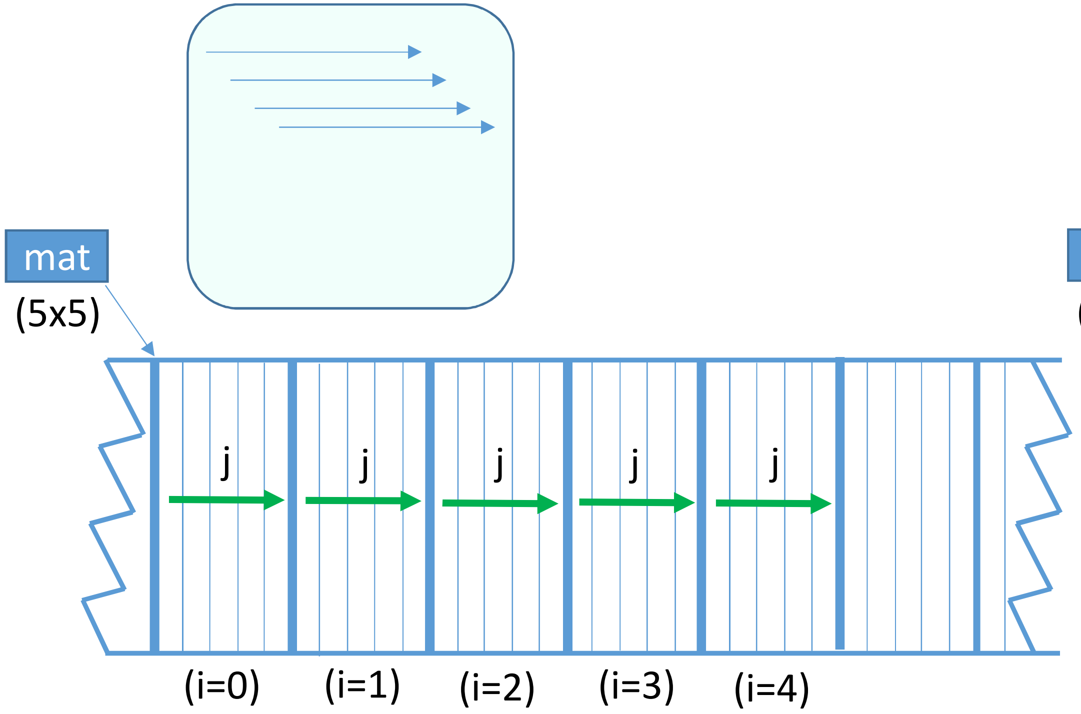


ACCESS(j,i)



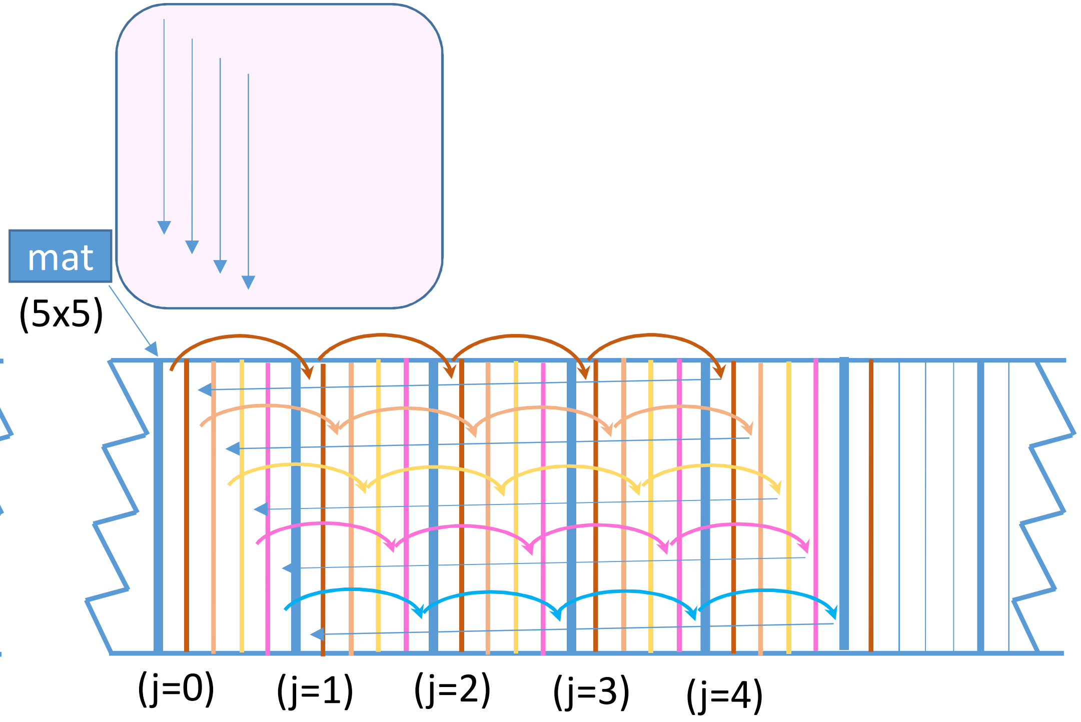
Thread performance sample

- ACCESS(i,j)



~5 misses

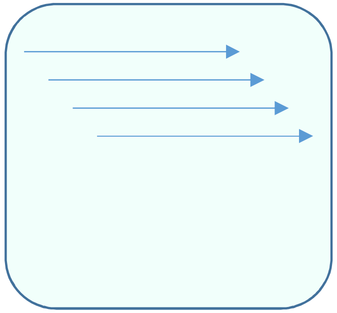
ACCESS(j,i)

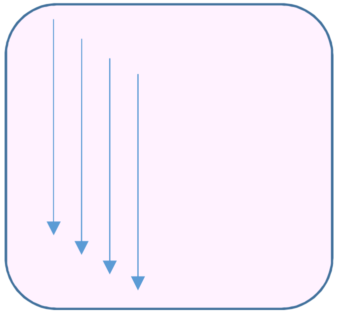


>5 misses (capacity)

Thread performance sample

- ACCESS(i,j) ACCESS(j,i)





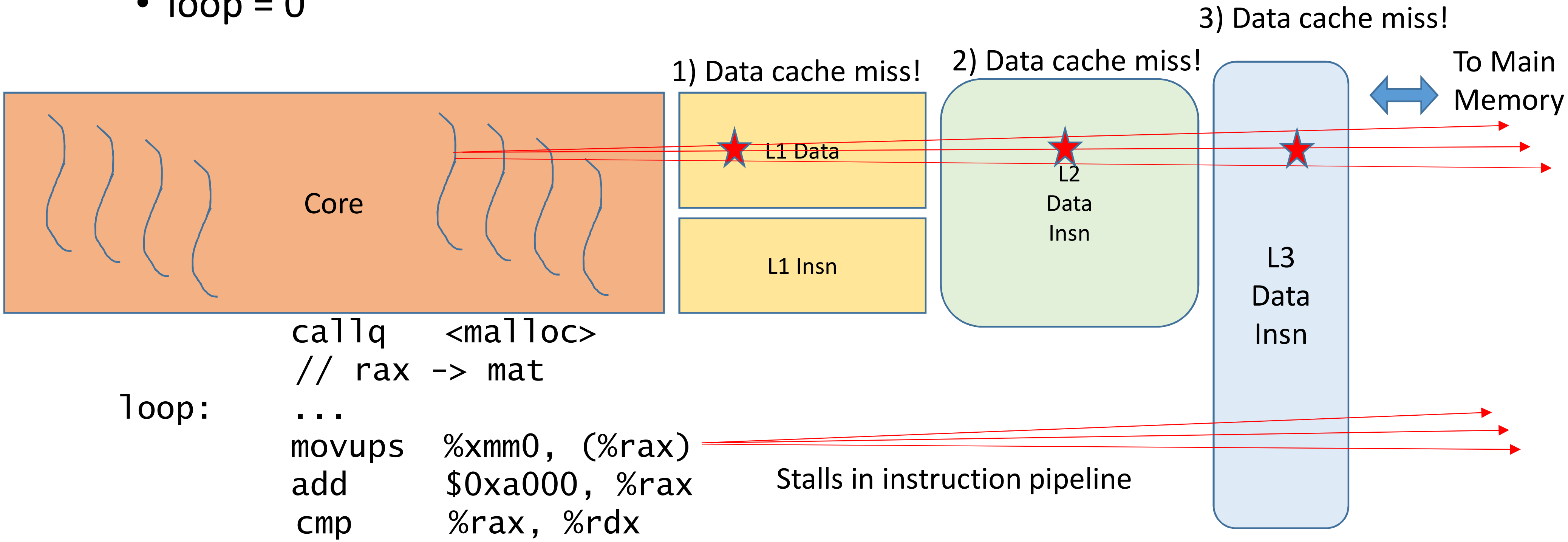
  - Matrix 10240 x 10240, single precision floating point values
  - O3 compiler optimization level
  - MN4 processor, Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz
  - Initialization times
    - loop =0: 148 ms loop =0: 465 ms
    - loop >0:   51ms loop >0: 361ms

Thread performance sample

• Row-wise access	Column-wise access
• Initialization times	
• loop =0: 148 ms	loop =0: 465 ms
• loop >0: 51 ms	loop >0: 361 ms
• Cache friendly	Cache unfriendly
• Spatial and temporal locality	Temporal locality, no spatial locality
• O2 optimization level	
• loop =0: 192 ms	loop =0: **1.5 s**
• loop >0: 100 ms	loop >0: **1.4 s**

Thread performance sample

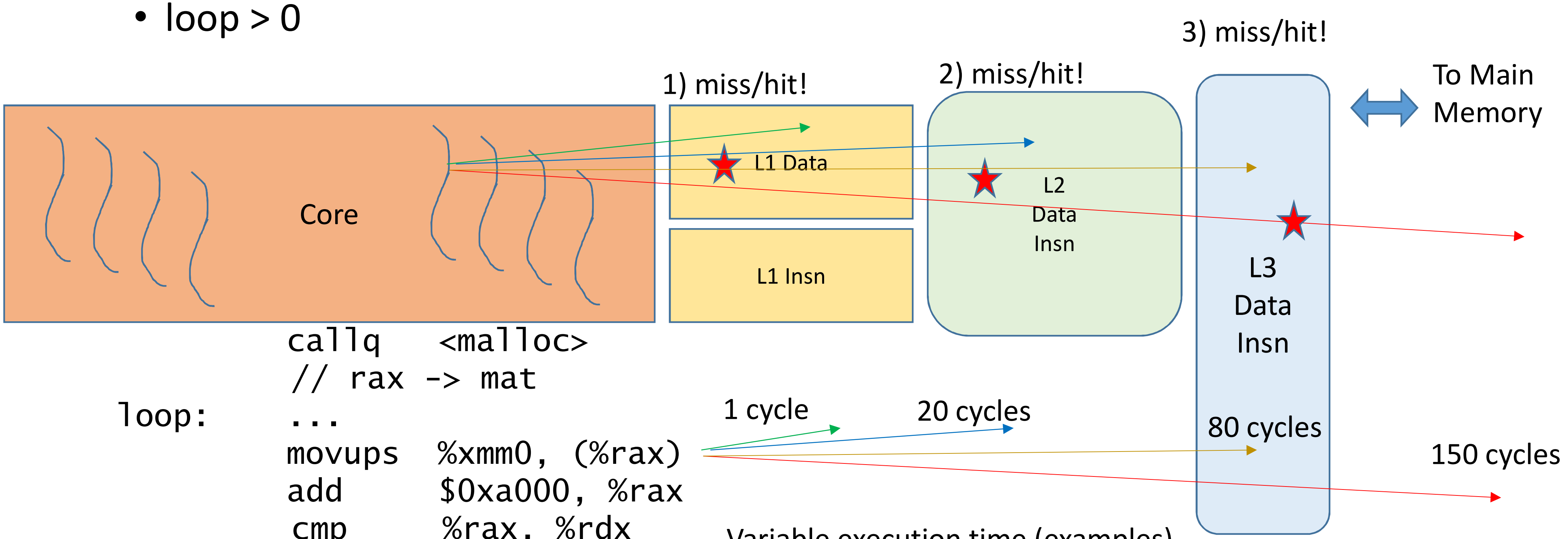
- Cache accesses
  - loop = 0



```
           callq    <malloc>
           // rax -> mat
loop:      ...
           movups   %xmm0, (%rax)
           add      $0xa000, %rax
           cmp      %rax, %rdx
           jne      loop
           ...
```

Thread performance sample

- Cache accesses
  - loop > 0



```
           callq    <malloc>
           // rax -> mat
loop:      ...
           movups   %xmm0, (%rax)
           add      $0xa000, %rax
            cmp      %rax, %rdx
           jne      loop
           ...
```

Variable execution time (examples)

Matrix multiplication

```
#include <iostream>
#include <stdlib.h>

template <class T, int d0, int d1>
class matrix {

 T data[d0][d1];

public:

 ...

 // matrix product, no blocking
     void product(matrix * A, matrix * B)
     {
        struct timeval tv0, tv1;
        int i, j, k;
        int res;
```

```
        res = gettimeofday(&tv0, NULL);
        if (res < 0) perror("gettimeofday");
#pragma omp parallel for \
        private(i, j, k) schedule(static)
        for (i=0; i < d0; i++) {
           for (j=0; j < d1; j++) {
#pragma omp simd // not successful (k access)
              for (k=0; k < d1; k++) {
                 data[i][j] +=
                      A->data[i][k] *
                      B->data[k][j];
        }  }  }
        res = gettimeofday(&tv1, NULL);
        if (res < 0) perror("gettimeofday");
        tv1.tv_sec -= tv0.tv_sec;
        tv0.tv_sec =  0;
        double t = tv1.tv_sec*1000000.0
                + tv1.tv_usec - tv0.tv_usec;
        fprintf (stderr,
              "matmul: %lf us %lf GFlops\n",
                 t, 2.0*d0*d1*d1 / t / 1e3);
     }
```

Matrix multiplication

```
#include <iostream>
#include <stdlib.h>

template <class T, int d0, int d1>
class matrix {

 T data[d0][d1];

public:

 ...

 // matrix product, no blocking
     void product(matrix * A, matrix * B)
     {
        struct timeval tv0, tv1;
        int i, j, k;
        int res;
```

```
        res = gettimeofday(&tv0, NULL);
        if (res < 0) perror("gettimeofday");
#pragma omp parallel for \
        private(i, j, k) schedule(static)
        for (i=0; i < d0; i++) {
              for (k=0; k < d1; k++) {
#pragma omp simd  // successful
           for (j=0; j < d1; j++) {
                 data[i][j] +=
                      A->data[i][k] *
                      B->data[k][j];
        }  }  }
        res = gettimeofday(&tv1, NULL);
        if (res < 0) perror("gettimeofday");
        tv1.tv_sec -= tv0.tv_sec;
        tv0.tv_sec =  0;
        double t = tv1.tv_sec*1000000.0
                + tv1.tv_usec - tv0.tv_usec;
        fprintf (stderr,
              "matmul: %lf us %lf GFlops\n",
                 t, 2.0*d0*d1*d1 / t / 1e3);
     }
```

Performance metrics

- Execution time (s)
  - Actual wall-clock time
  - Affected by hardware events, all of them count as time spent in the app
    - Interrupts
    - Exceptions
    - System calls
  - ... and OS events
    - Multiprogramming level
- CPU time (s)
  - Amount of time spent running on a CPU (hw thread), in user and/or system mode
- Speed-up (no units)
  - Relation between the serial execution time and the parallel execution time
  - Usually applied to wall-clock time

Performance metrics

- Bandwidth (bytes/s)
  - Relation between the amount of data transmitted and the time invested
- Latency (s)
  - Amount of time to start operations or communications
- Throughput (elements/s)
  - Maximum amount of operations, applications, units per second
  - Maximum rate of production / processing / consumption
- Power consumption (W)
  - Work done per unit of time
- And many many more...

Statistics

- Average
  - Provides a more stable and realistic measurement
  - Sum of samples, divided by the number of samples
  - Also possible:
    - Harmonic mean (average of rates)
    - Geometric mean (when comparing different items, with different numeric ranges)
- Standard deviation
  - Indicates how much variability there is among the results
  - $s = \sqrt{\frac{\sum_{i=0}^{N-1} \ (x_i - \bar{x})^2}{N-1}}$

How to use the statistics

- Execution time
    - Average* of the N results obtained
- Speed-up
    - Average sequential execution time over average parallel execution time
- Bandwidth
    - Average* of the bandwidth obtained in N experiments
- Latency
    - Average* of the latencies obtained in N experiments

* Average can be changed by standard deviation

Table of Contents

- Introduction
- Performance metrics
- OS support services
- System tools
  - System calls
  - Commands
  - Global system information
  - Detailed hardware information

OS support services

- System interface

```
int gettimeofday (struct timeval * tv, struct timezone * tz);
```

- Returns seconds and microseconds since Epoch (1970-01-01 00:00:00 +0 (UTC))
  - struct timeval {

```
    time_t         tv_sec;
    suseconds_t    tv_usec;
};
```

Use of timezone is deprecated, use NULL

```
time_t time (time_t * t);
```

- Returns seconds since Epoch (1970-01-01 00:00:00 +0 (UTC))

OS support services

- System interface

```
int clock_gettime (clockid_t clk_id, struct timespec * time);
```

  - Returns seconds and nanoseconds as provided by the clock clk_id
    - struct timespec {

```
    time_t      tv_sec;
    long        tv_nsec;
};
```

```
int clock_getres (clockid_t clk_id, struct timespec * time);
```

  - Returns the resolution of the given clock (in seconds and nanoseconds)

OS support services

- Clocks available (CLOCK_*_ID)
  - REALTIME and MONOTONIC (1 nanosecond)
    - COARSE versions (4 miliseconds)
  - MONOTONIC_RAW (1 nanosecond)
  - BOOTTIME (1 nanosecond)
  - PROCESS_CPUTIME, THREAD_CPUTIME (1 nanosecond)

OS support services

- Thread and process CPU clocks

```
#pragma omp parallel shared(...) firstprivate(...)
private(res)
{
      // some work done in parallel
     {
         ...
     }

     struct timespec thts;
     res = clock_gettime(CLOCK_THREAD_CPUTIME_ID, &thts);
#pragma omp critical
     printf ("%d: %lf ms\n", omp_get_thread_num(),
                             timespec_to_ms(&thts));
}
struct timespec ts;
res = clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &ts);
printf ("Process: %lf ms\n", timespec_to_ms(&ts));
...
```

```
login5:~> OMP_NUM_THREADS=4 ./timth
3: 669.028954 ms
1: 669.088039 ms
0: 673.971984 ms
2: 669.144304 ms
Process: 2681.277546 ms
```

OS support services

- Resource usage

```
int getrusage(int who, struct rusage *usage);
```

  - RUSAGE_SELF current process (including all threads)
  - RUSAGE_CHILDREN all children/granchildren/... terminated and waited for
  - RUSAGE_THREAD calling thread

OS support services

- Resource accounting for processes/threads

```
struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long   ru_maxrss;        /* maximum resident set size / that of the largest child */
...
    long   ru_minflt;        /* page reclaims (soft page faults) */
    long   ru_majflt;        /* page faults (hard page faults) */
...
    long   ru_inblock;       /* block input operations */
    long   ru_oublock;       /* block output operations */
...
    long   ru_nvcsw;         /* voluntary context switches */
    long   ru_nivcsw;        /* involuntary context switches */
};
```

login5:~> man top

Top

Uptime

1min 5min 15min

```
top - 16:44:25 up 126 days,  7:39,  6 users,  load average: 0.03, 0.05, 0.03
Threads: 3114 total,   5 running, 3108 sleeping,   0 stopped,   1 zombie
%Cpu(s):  3.9 us,  0.1 sy,  0.0 ni, 96.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  98621952 total, 36142820 used, 62479136 free,   214448 buffers
KiB Swap:  3905532 total,       508 used,  3905024 free. 28783796 cached Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
199766 bsc15371  20   0   25636   2384   2180 R 95.45 0.002   0:00.63 timth
199767 bsc15371  20   0   25636   2384   2180 R 95.45 0.002   0:00.63 timth
199768 bsc15371  20   0   25636   2384   2180 R 95.45 0.002   0:00.63 timth
199769 bsc15371  20   0   25636   2384   2180 R 95.45 0.002   0:00.63 timth
199677 bsc15371  20   0   16416   4956   1824 R 27.27 0.005   0:00.39 top
     1 root      20   0   37464   5524   3904 S 0.000 0.006 120:28.62 systemd
     2 root      20   0       0      0      0 S 0.000 0.000   0:36.93 kthreadd
     3 root      20   0       0      0      0 S 0.000 0.000   2:49.45 ksoftirq+
     8 root      20   0       0      0      0 S 0.000 0.000  49:33.20 rcu_sched
     9 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    10 root      rt   0       0      0      0 S 0.000 0.000   0:08.30 migratio+
login5:~> man free
```

top - free

- Physical memory
  - Total – Memory installed
  - Used – Total - free
  - Free – Unused memory
  - Buffers – Memory used in kernel buffers
  - Cached Mem – Memory used by the page cache (files on filesystem)
- Virtual memory (swap)
  - Total
  - Used
  - Free

```
login5:~> free
             total       used       free     shared    buffers     cached
Mem:      98621956   36142200   62479756   22209792     214448   28784456
-/+ buffers/cache:    7143296   91478660
Swap:      3905532        508    3905024
```

ps

- List processes from /proc/
  - Multiple data about processes
    - By default attached to the current terminal (pts)

```
login5:~> ps
   PID TTY          TIME CMD
199294 pts/6    00:00:00 bash
206909 pts/6    00:00:00 ps
login5:~> ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
bsc15371 199294 199292  0 16:43 pts/6    00:00:00 -bash
bsc15371 206910 199294 99 17:24 pts/6    00:00:00 ps -f
login5:~> ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  3003 199294 199292  0  80   0 -  3842 wait   pts/6    00:00:00 bash
0 R  3003 206947 199294 99  80   0 -  8766 -      pts/6    00:00:00 ps
```

ps

- Very flexible
  - -o <field-list>

```
login5:~> ps -u bsc15371 -mo user,pid,tid,psr,stat,comm
USER        PID     TID PSR STAT COMMAND
bsc15371 199294       -   - -    bash
bsc15371      - 199294   7 Ss   -
bsc15371 209303       -   - -    timth
bsc15371      - 209303  24 Rl+  -
bsc15371      - 209304  10 Rl+  -
bsc15371      - 209305   1 Rl+  -
bsc15371      - 209306  38 Rl+  -
bsc15371      - 209307  32 Rl+  -
bsc15371      - 209308  11 Rl+  -
bsc15371      - 209309  20 Rl+  -
bsc15371      - 209310  41 Rl+  -
bsc15371      - 209311   2 Rl+  -
bsc15371      - 209312  21 Rl+  -
bsc15371      - 209313   0 Rl+  -
bsc15371      - 209314  43 Rl+  -
bsc15371      - 209315  40 Rl+  -
bsc15371 209316       -   - -    ps
bsc15371      - 209316  36 R+
```

time

- Different versions – some get partial information from getrusage
  - sh/bash csh/tcsh

```
real    0m6.286s        299.953u 0.003s 0:06.28 4776.2% 0+0k 0+0io 0pf+0w
user    4m59.988s
sys     0m0.008s
```

  - Ksh

```
0m6.26s real     4m59.91s user     0m0.01s system
```

  - /usr/bin/time

```
299.99user 0.01system 0:06.29elapsed 4768%CPU (0avgtext+0avgdata 3288maxresident)k
0inputs+0outputs (0major+284minor)pagefaults 0swaps
```

Global system information

- vmstat [N]   ... display every N seconds

```
login5:~> vmstat 1
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd     free   buff    cache    si    so     bi     bo    in    cs us sy id  wa st
 1  0    508 70910968 214448 20377984     1     1      1      2     0     0  0  0 100  0  0
 0  0    508 70910808 214448 20377984     0     0      0      0  1982   647  0  0 100  0  0
 1  0    508 70910956 214448 20377984     0     0      0      0    93   114  0  0 100  0  0
 1  0    508 70911328 214448 20377984     0     0      0      0   162   156  0  0 100  0  0
 0  0    508 70911484 214448 20377980     0     0      0      0   130   141  0  0 100  0  0
 0  0    508 70911500 214448 20377980     0     0      0      0   412   339  0  0 100  0  0
 2  0    508 70899212 214448 20377980     0     0      0      0  4388  8321  0  0 99  0  0
 0  0    508 70915272 214448 20377972     0     0      0      0 13232 25858  0  1 99  0  0
 0  0    508 70913000 214448 20377972     0     0      0     24   119   100  0  0 100  0  0
 0  0    508 70912848 214448 20377972     0     0      0      0   128   200  0  0 100  0  0
 1  0    508 70912392 214448 20377972     0     0      0      0   181   124  0  0 100  0  0
```

Showing the execution of `dd if=/dev/zero of=myfile2.txt count=32 bs=$((1024*1024*16))`

Global system information

- vmstat... Now with I/O – bi and bo

```
procs -----------memory----------    ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache      si   so    bi    bo   in   cs us sy id wa st
 1  0 3325472 16329064 350944 3172824    0    0     0      0 1143  218  0  8 92  0  0
 0  1 3325472 16010516 350948 3478464    0    0     4 130156 1172  247  0  6 92  2  0
 0  1 3325472 16010752 350956 3478700    0    0     8 229784 1175  322  0  0 92  8  0
 0  1 3325472 16010008 350968 3478716    0    0     8 205312 1186  299  0  0 91  8  0
 0  2 3325472 16010500 350976 3478732    0    0     8 221184 1163 12809  0  1 87 12  0
 1  1 3325472 15907332 350980 3580020    0    0     4 181288 1410 22003  0  3 84 13  0
 0  2 3325472 15508800 350984 3964024    0    0     4 72736 1996  693  2  8 82  8  0
 0  2 3325472 15507312 350988 3964156    0    0     4 148056  892  451  0  0 90  9  0
 0  1 3325472 15507584 350992 3964176    0    0     4 213008 1103  472  0  0 87 12  0
 1  0 3325472 15257352 350996 4207368    0    0     4 155428 1587  383  0  5 90  4  0
 0  1 3325472 15131740 351004 4329432    0    0     8 217816 1385  279  0  3 92  5  0
 0  0 3325472 15130268 351004 4334036    0    0     0 44176  722  274  0  0 98  1  0
 0  0 3325472 15130516 351004 4334036    0    0     0      0  235  271  0  0 100  0  0
```

```
dd if=/dev/zero of=/tmp/myfile.txt bs=1024 count=$((1024*1024*16))
dd if=/dev/zero of=/tmp/myfile.txt bs=16 count=$((1024*1024*16))
```

Global system information

- iostat [N]   [device] …

```
nvblogin2 ~$ iostat  1 /dev/sda
Linux 2.6.32-642.6.2.el6.x86_64 (nvblogin2)     05/22/2018      _x86_64_   (12 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.86    0.00    0.52    0.02    0.00   96.60
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               4.41         2.43        89.49   28382640 1045346806

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.00    0.00    0.00    0.00    0.00  100.00
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.83    0.00    9.08    1.75    0.00   87.33
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda             273.00         0.00    246472.00          0     246472
```

Global system information

- CPU frequency
  - cpufreq-set – change CPUs frequency
    - Governor… powersave, conservative, ondemand, performance
  - cpufreq-aperf – computes the average frequency over time
  - cpufreq-info – reports information about CPU frequencies
  - Also available in /proc/cpuinfo

Global system information

```
nvblogin2 ~$ cpufreq-info
cpufrequtils 007: cpufreq-info (C) Dominik Brodowski 2004-2009
Report errors and bugs to cpufreq@vger.kernel.org, please.
analyzing CPU 0:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0 1 2 6 7 8
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 10.0 us.
  hardware limits: 1.60 GHz - 2.53 GHz
  available frequency steps:  2.53 GHz, 2.39 GHz, 2.26 GHz, 2.13 GHz,
                              2.00 GHz, 1.86 GHz, 1.73 GHz, 1.60 GHz
  available cpufreq governors: userspace, performance
  current policy: frequency should be within 1.60 GHz and 2.53 GHz.
                  The governor "userspace" may decide which speed to use
                  within this range.
  current CPU frequency is 2.53 GHz.
analyzing CPU 1:
  ...
  CPUs which run at the same hardware frequency: 0 1 2 6 7 8
  CPUs which need to have their frequency coordinated by software: 1
  ...
  current CPU frequency is 2.53 GHz.
```

Global system information

- cpupower
  - frequency-info
  - idle-info

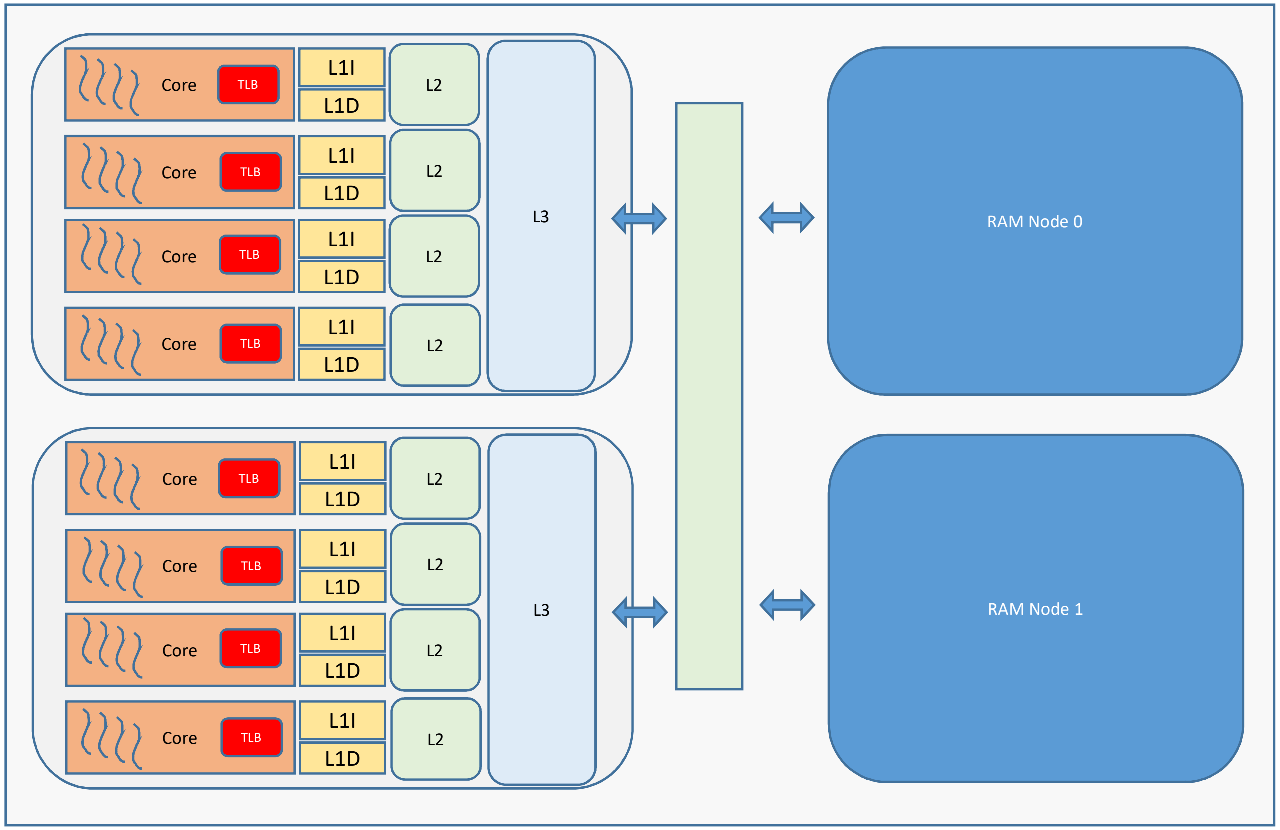
```
login5:~> cpupower -c 2 idle-info
CPUidle driver: intel_idle
CPUidle governor: menu
analyzing CPU 2:

Number of idle states: 4
Available idle states: POLL C1-SKX C1E-SKX C6-SKX
POLL:
Flags/Description: CPUIDLE CORE POLL IDLE
Latency: 0
Usage: 16998351
Duration: 9758705762
C1-SKX:
Flags/Description: MWAIT 0x00
Latency: 2
Usage: 36475538
Duration: 17301919769
C1E-SKX:
Flags/Description: MWAIT 0x01
Latency: 10
Usage: 72152895
Duration: 30924032072
C6-SKX:
Flags/Description: MWAIT 0x20
Latency: 133
Usage: 168982812
Duration: 10769469931715
```

  - frequency-set
  - idle-set

Detailed hardware information

- Dependent on the particular processor model

Fetched instr.
    arithmetic
    jump/branches
    floating point
        add/sub
        mul/div
        …
Executed instr.
        …
Retired instr.
        …
TLB hits/misses
L1I hits/misses
L1D hits/misses
L1D invalidations



L2 events
L3 events
Accesses to RAM
        local
        remote
External interventions
…

Hardware
Performance
Counters

Detailed hardware information

- Software provides access to the internal CPU counters
  - OS
  - Libraries – e.g. libpapi - performance API
- PAPI
  - List counters available – different in each architecture/processor

```
 Name         Code     Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes    No    Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes    No    Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes    Yes   Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes    No    Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No     No    Level 3 data cache misses
PAPI_L3_ICM  0x80000005  No     No    Level 3 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes    Yes   Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes    No    Level 2 cache misses
PAPI_L3_TCM  0x80000008  Yes    No    Level 3 cache misses
```

Detailed hardware information

- PAPI
  - Hardware can count several (4 – 8) counters at a time
  - Multiplex is possible (time slicing across counters)
  - System level / User / Kernel / Interrupt / per process / per thread
  - Can deliver overflow interrupts to software
    - Implement counters-based profiling

Next steps

- Data management
  - Input/output
  - File systems