Experience with PAPI performance analysis tool

Rui Silva

3 December 2012
Universidade do Minho

CENTROALGORITMI

CENTROALGORITMI

Outline

- Introduction to PAPI
- My experience with PAPI

CENTROALGORITMI

Introduction to PAPI

CENTROALGORITMI

PAPI

- Uniform access to hardware performance counters
- My usage
  - Justification of gains of optimisations
  - Identify side effects
  - Identify the executed code (Hotspot)

CENTROALGORITMI

Code executed

```
..B1.14:
        movaps    %xmm0, %xmm3
        paddd     %xmm1, %xmm0
        pslld     $2, %xmm3
        movdqa    %xmm3, (%rax,%rcx,4)
        addq      $4, %rcx
        paddd     %xmm3, %xmm2
        cmpq      %rdx, %rcx
        jb        ..B1.14
```

```
..B2.12:
        movdqa    (%rcx), %xmm0
        addq      $4, %rax
        pslld     $2, %xmm0
        movdqa    %xmm0, (%rcx)
        addq      $16, %rcx
        cmpq      %r8, %rax
        jb        ..B2.12
```

The results

What is its significance?

- The number is **good** or **bad**?

- The question is: **it is possible to improve?**
  - Knowledge of the problem
    - Define the bottleneck of the code (Other tools. Gprof...)
    - When possible identify theorical limit (Gflops/Misses)

Example (-O0 vs -O3)

	?	?
CPI	**0.6**	**1.3**

Convolve 3x1

CENTROALGORITMI

Example (-O0 vs -O3)

	?	?
#I	3.1×10^6	1.0×10^6
#CC	2.0×10^6	1.4×10^6
CPI	**0.6**	**1.3**

Convolve 3x1

CENTROALGORITMI

Example (-O0 vs -O3)

	-O0	-O3
#I	3.1×10^6	1.0×10^6
#CC	2.0×10^6	1.4×10^6
CPI	**0.6**	**1.3**

Convolve 3x1

CENTROALGORITMI

Improve performance

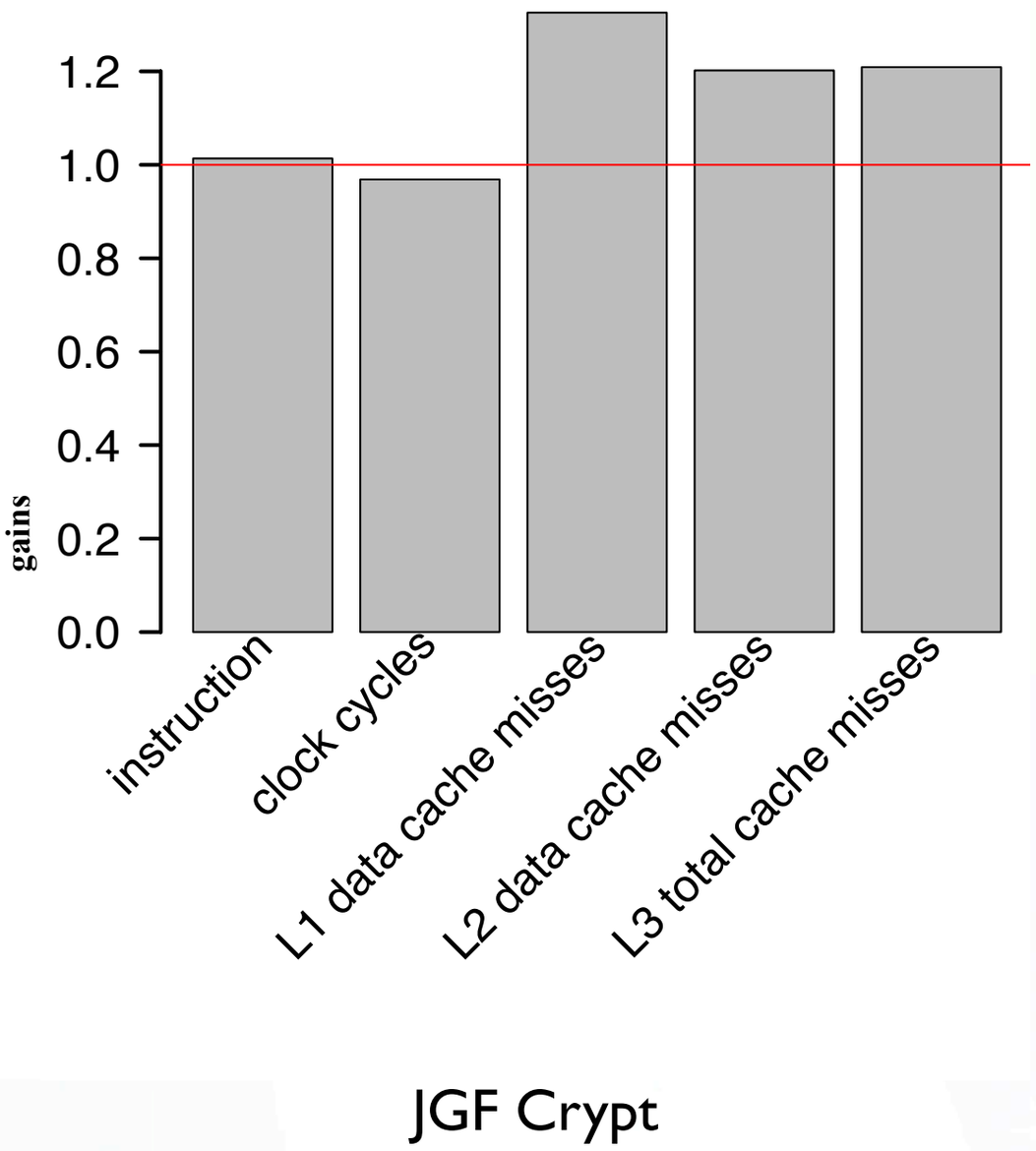
- Knowledge of the problem
- Knowledge and implementation of the optimisations
- Analyse and compare performance counters
  - Justify improvements with the new values
  - Analyse side effects
  - It may be necessary to analyse more counters (using guessing/intuition)

Example (Loop Fusion)

```
for i = 1 to N
    M[ i ] += C1
for i = 1 to N
    M[ i ] *= C2
```

```
for i = 1 to N
    M[ i ] += C1
    M[ i ] *= C2
```

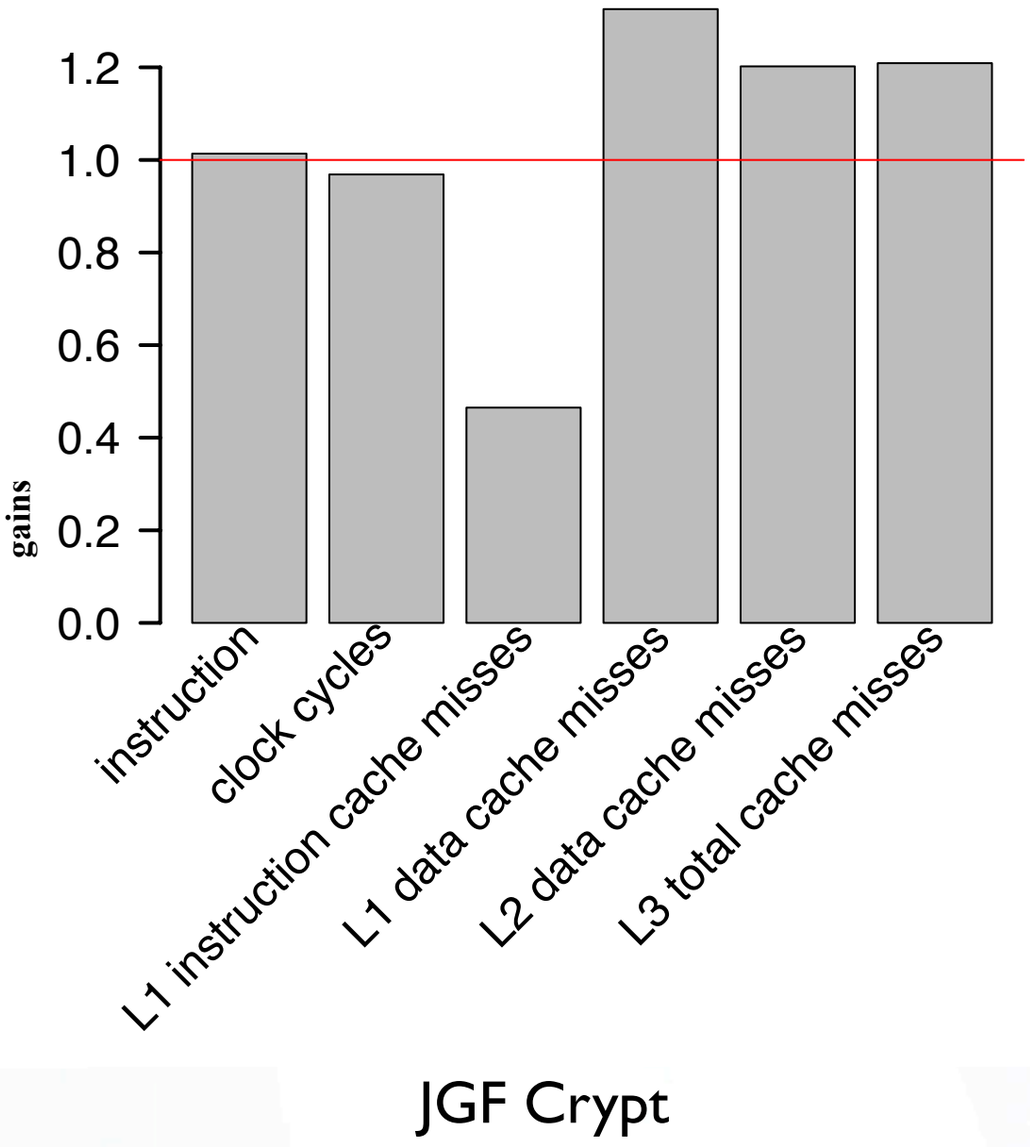
Example (Loop Fusion)



JGF Crypt

- What is the problem?
  - Less instructions
  - Better locality in access data
  - But more clock cycles

Example (Loop Fusion)



JGF Crypt

- More misses on access the instruction

CENTROALGORITMI

Performance metrics

- CPI and miss rate
  - Hides increases in instructions and access
- Cycles/Misses/Instructions per element
  - Different problems, different values
- It is not direct to compare the performance of different problems

CENTROALGORITMI

What to measure?

- All code
  - Hide local improvements
- Part of the code that was optimised
  - The size of input
    - Attention to precision of PAPI (papi_cost)
    - Data size (example optimisation access data, the problem do not fit in cache levels)

How to use PAPI
(EXAMPLE 1)

```
(...)
string nEvents[NUMEVENTS] = { "PAPI_TOT_INS", "PAPI_TOT_CYC" };
int events[NUMEVENTS] = {PAPI_TOT_INS, PAPI_TOT_CYC};
long long values[NUMEVENTS];
int errorcode;
  char errorstring[PAPI_MAX_STR_LEN+1];

  // Initialize Papi and its events
startPAPI();
errorcode = PAPI_start_counters(events, NUMEVENTS);

  convolve3x1 (res_img->buf, img->buf, img->width, img->height);

  errorcode = PAPI_stop_counters(values, NUMEVENTS);
for (int w=0; w<NUMEVENTS; w++)
  cout << nEvents[w] << ":" << values[w] << endl;
(...)
```

Problem in this approach

- The number of counters is limited
- Solution
  - Run several times

How to use PAPI
(EXAMPLE 1)

```
int main(int argc, char **argv) {

    events_define_by_user();

    for (int i = 0; i < papi_profiler_length_events; i++) {


        init_program();
        papi_profiler_i = i;

        main2(argc, argv); //original main

        PAPI_shutdown();
    }

    print_cache();


    exit(0);
}
```

```
(...)
// Initialize Papi and its events
papi_profiler_start();

 convolve3x1 (res_img->buf, img->buf, img->width, img->height);

 papi_profiler_stop();
(...)
```

CENTROALGORITMI

PAPI commands

- papi_avail
- papi_error_codes
- papi_cost
- papi_mem_info
- papi_native_avail

CENTROALGORITMI

My experience with PAPI

CENTROALGORITMI

Counters

- Total of instructions completed
- Total cycles
- Cache accesses (L1, L2 and L3)
- Cache misses (L1, L2 and L3)

CENTROALGORITMI

Case studies

- Molecular dynamics simulation
- Matrix Multiplication
- Others

CENTROALGORITMI

PAPI
Sequential versions

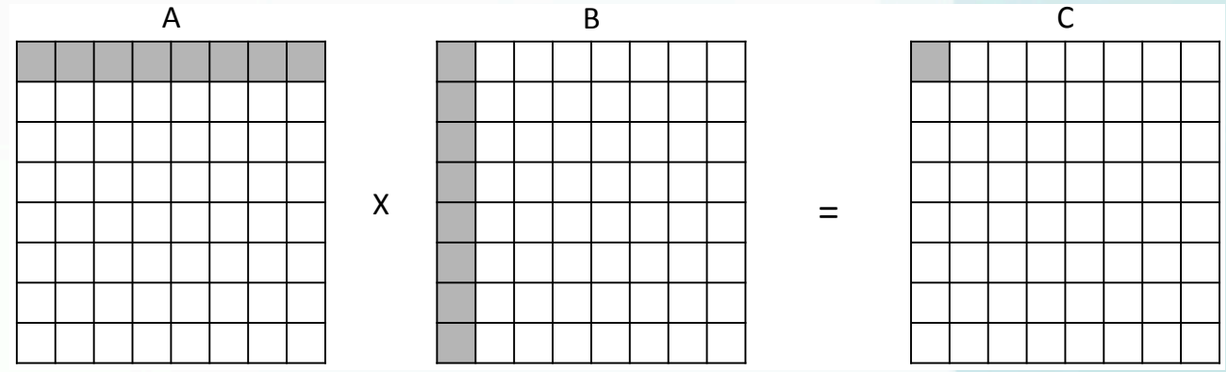
Loop Reorder

CENTROALGORITMI

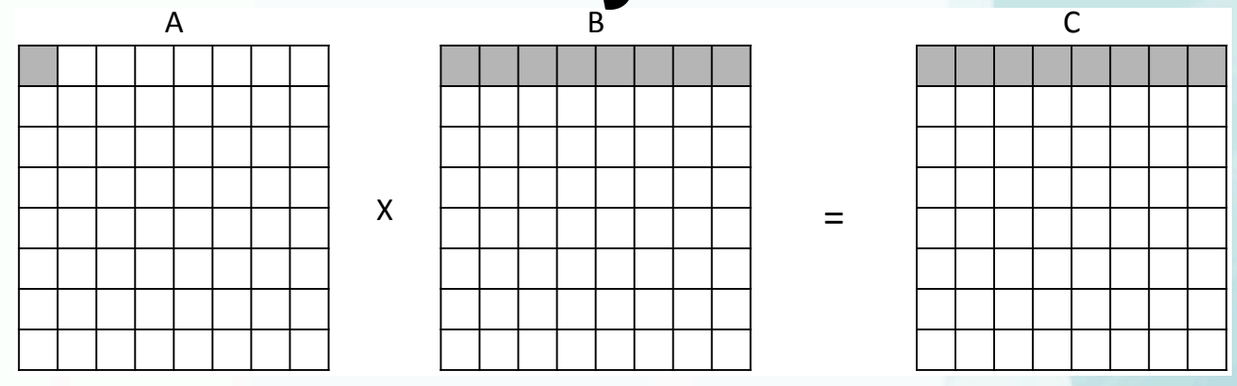
Loop Reorder

```
for i = 1 to RA
    for j = 1 to CB
        for k = 1 to RB
            C[i][j] += A[i][k]*B[k][j]
```

IJK

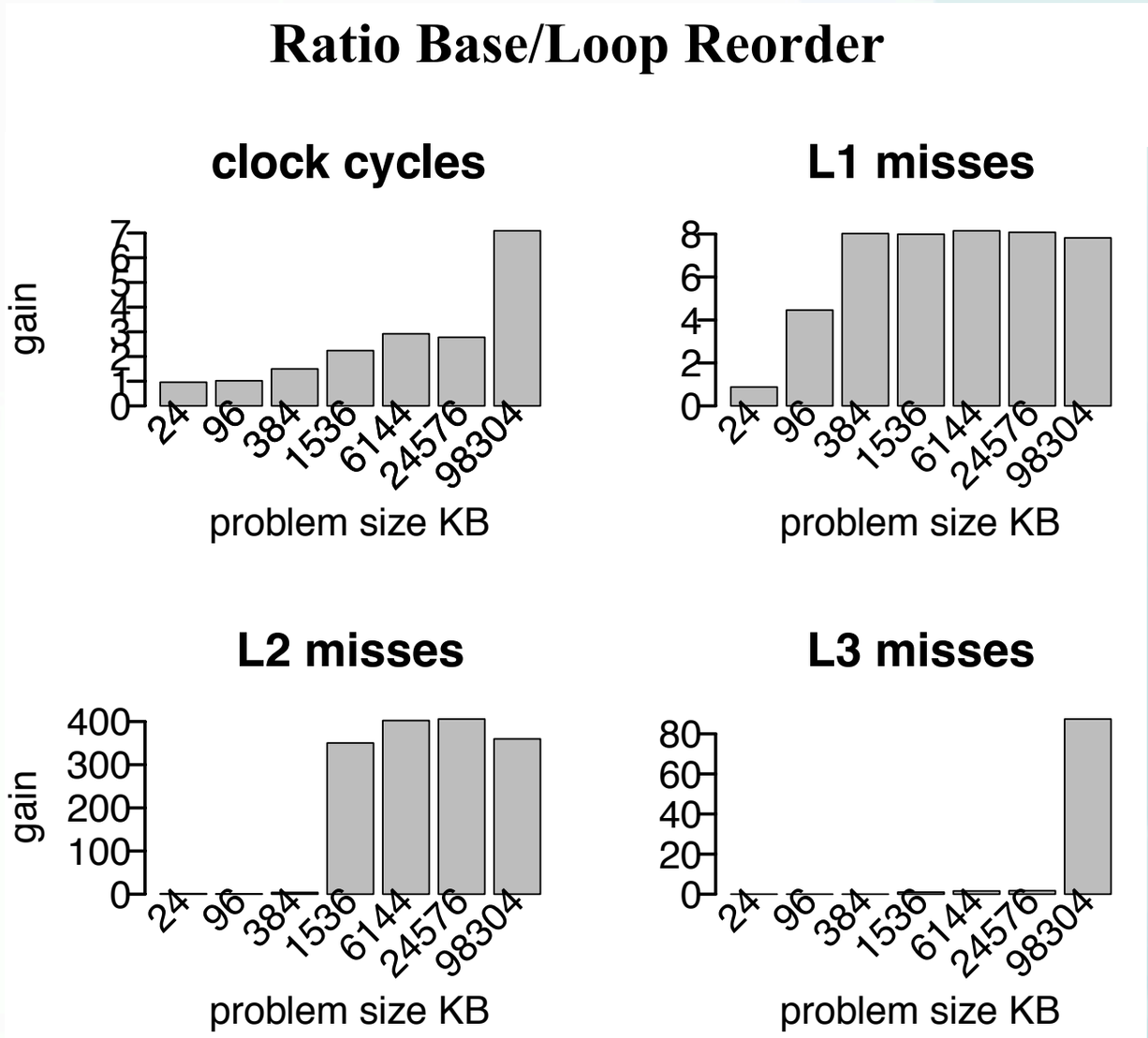


IKJ



Loop Reorder (MM)

- No gains on small problems
  - Fit in L1/L2 cache
- Better locality => Increased performance
  - Better usage of L2/L3
- Instruction count remains constant

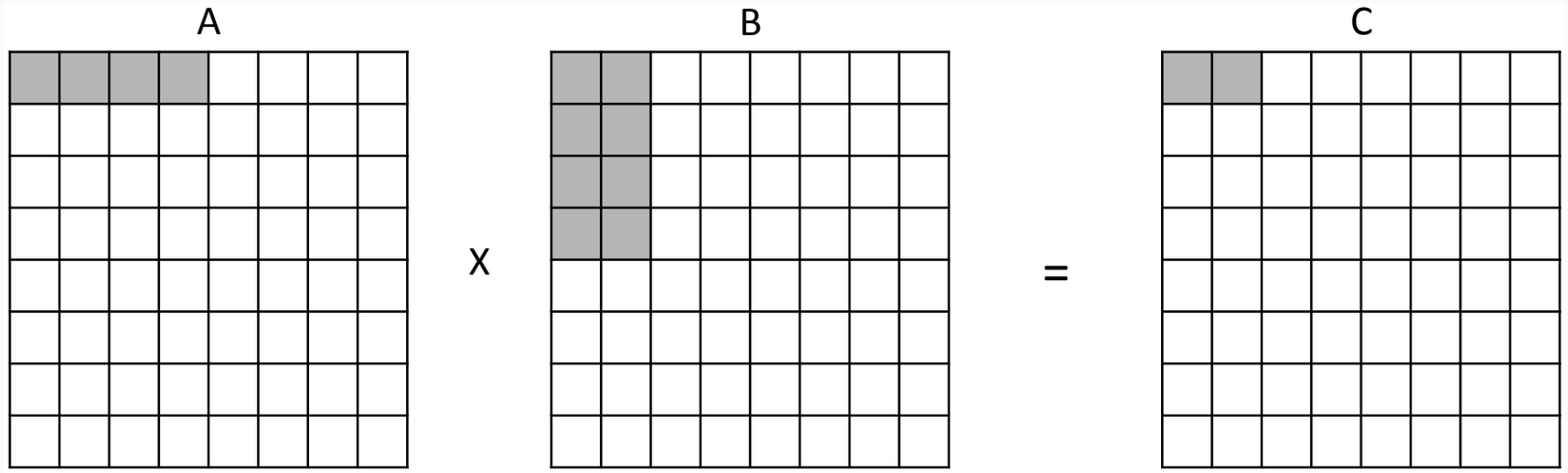


CENTROALGORITMI

Loop Tiling

CENTROALGORITMI

Matrix Multiplication

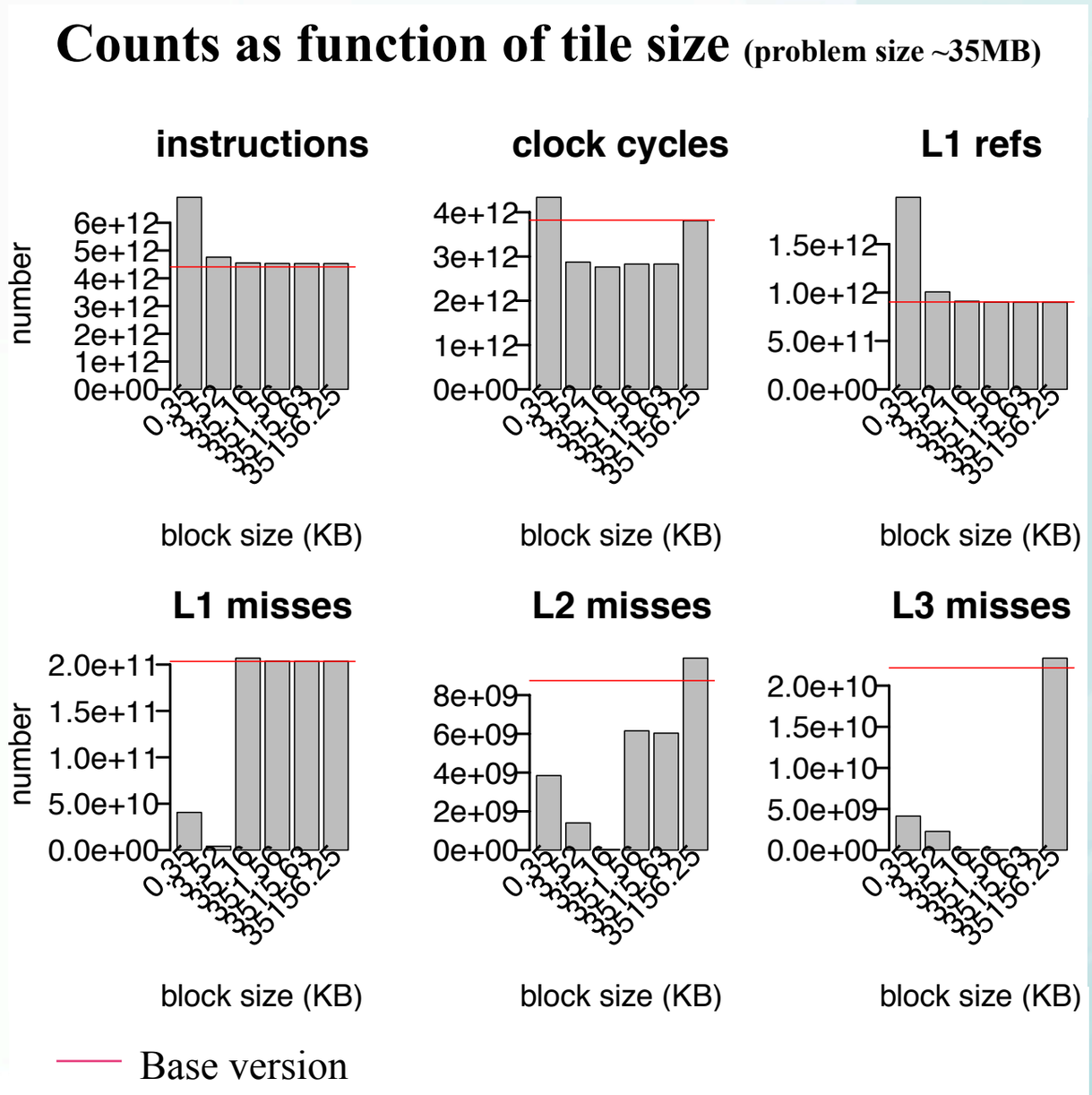


- Matrix computations by blocks
  - Block size is adjusted to different levels of the cache

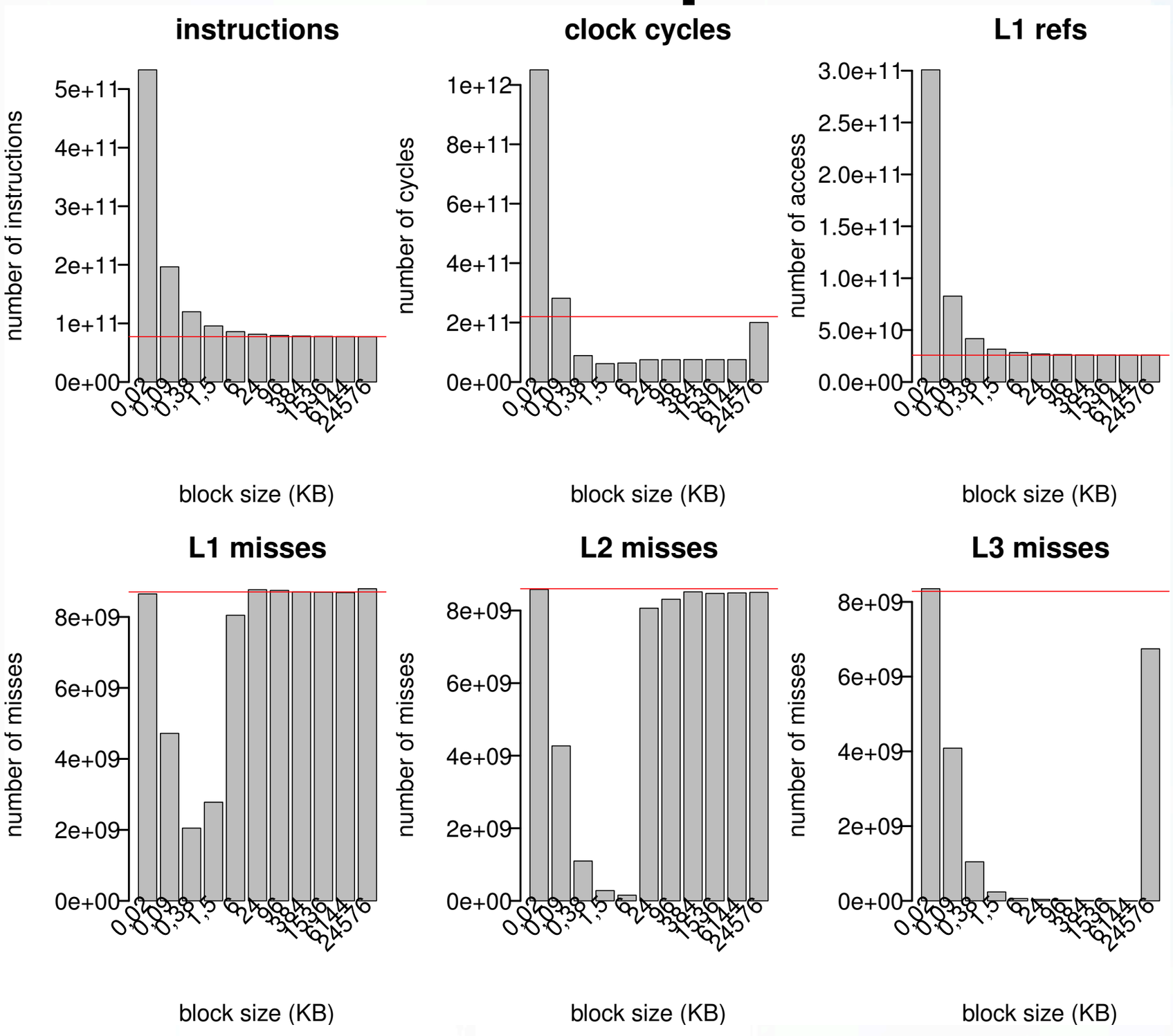
Loop Tiling (MD)

CENTROALGORITMI

- Instruction count overhead on small block size
  - Inner loop is small
- Tile size can be tuned to L1/L2/L3
  - More impact of L2/L3 misses
- Similar results in MM



Matrix Multiplication



CENTROALGORITMI

PAPI
Parallel versions

Code in parallel versions (1)

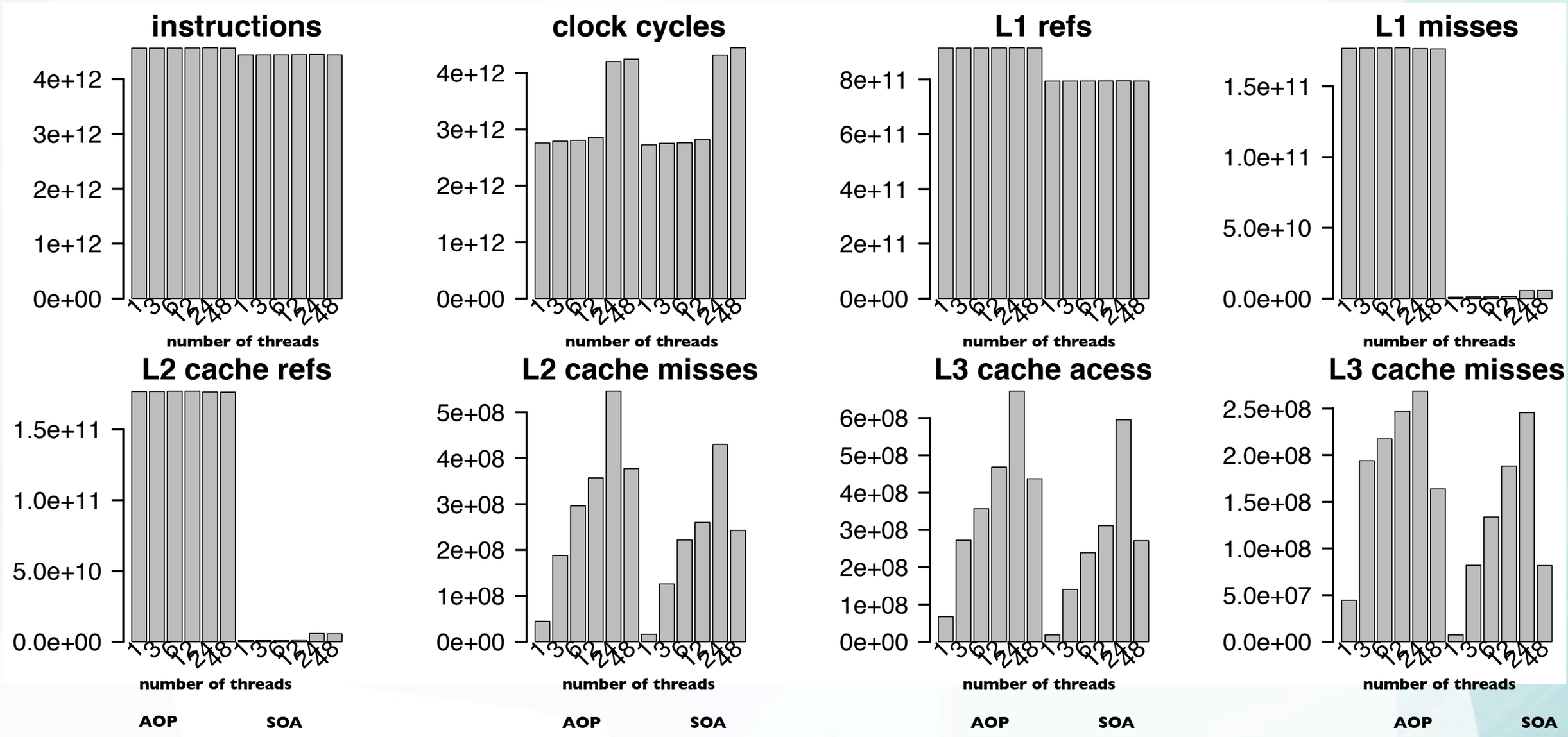
```
void runiters(MD *md, Particles *particulas) {

    Reduction vars[md->threads];
    create_newtowsArrays(vars, md);
    md->move = 0;
#pragma omp parallel
    {
        papi_profiler_start();

        for (; md->move < md->movemx;) {

#pragma omp master
            cicleDoMove(md, particulas); // Calcular o movimento

            cicleForces(md, particulas, vars); // Calcular a força

#pragma omp master
            {
                cicleMkekin(md, particulas); // Scale forces, update velocities
                cicleVelavg(md, particulas); // calcular a velocidade
                scale_temperature(md, particulas); // temperature scale if required
                get_full_potential_energy(md); // sum to get full potential energy and virial
            }

#pragma omp master
            md->move++;
#pragma omp barrier
        }

        papi_profiler_stop();
    }
}
```

Code in parallel versions (2)

```
int main(int argc, char **argv) {

    events_define_by_user();

    for (int i = 0; i < papi_profiler_length_events; i++) {

        init_program();
        papi_profiler_i = i;

        main2(argc, argv); //original main

        PAPI_shutdown();
    }

    print_cache();


    exit(0);

}
```

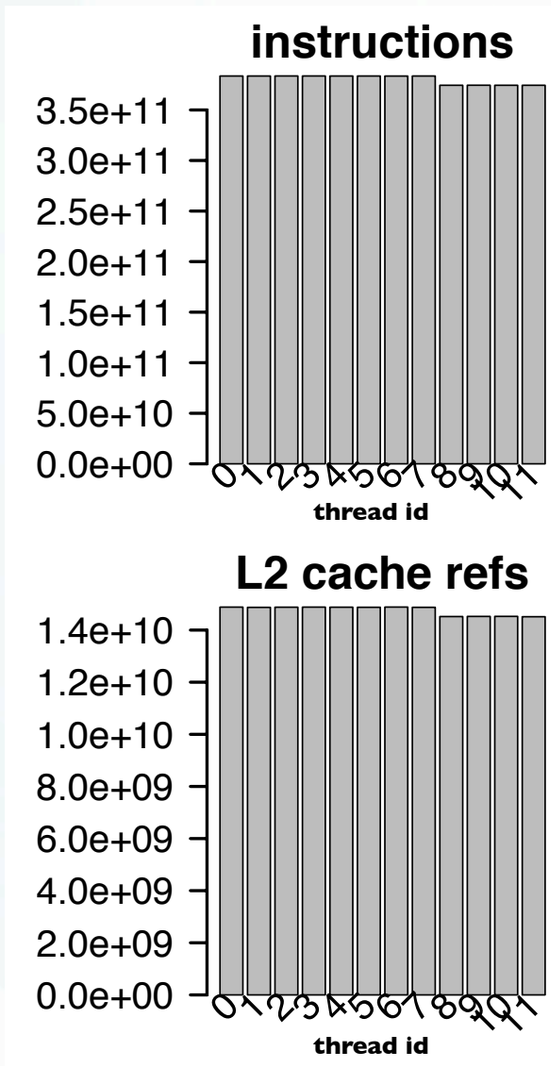
CENTROALGORITMI

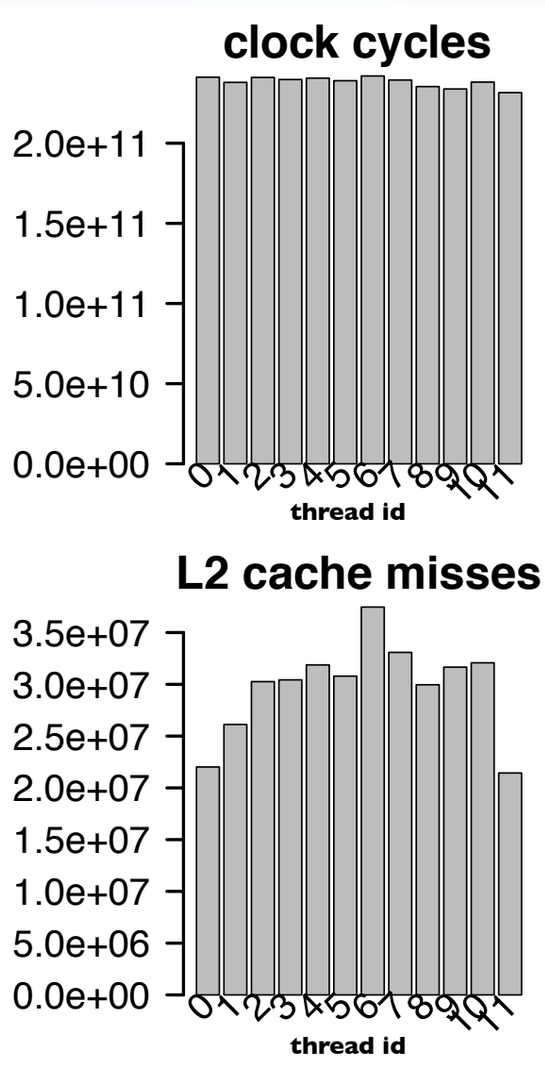
MD (AOP, SOA)

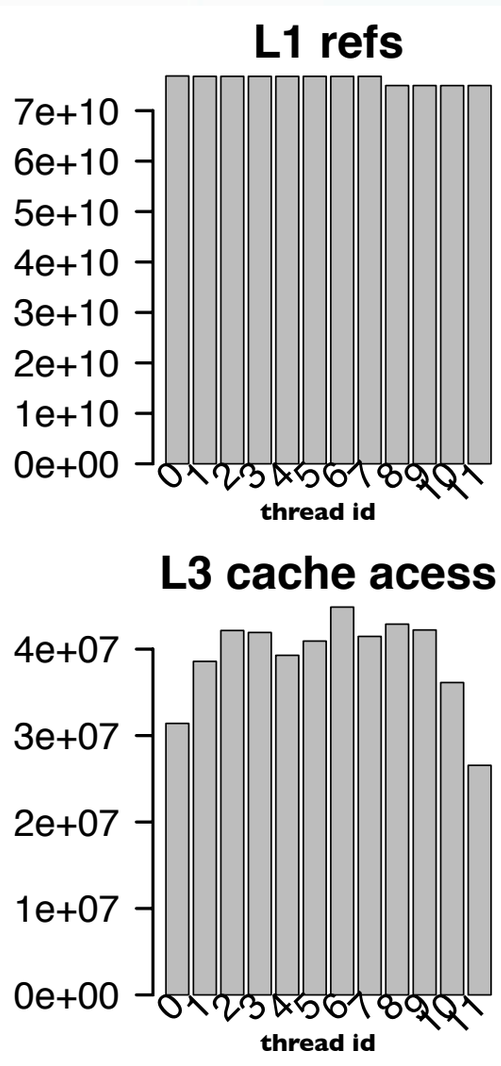


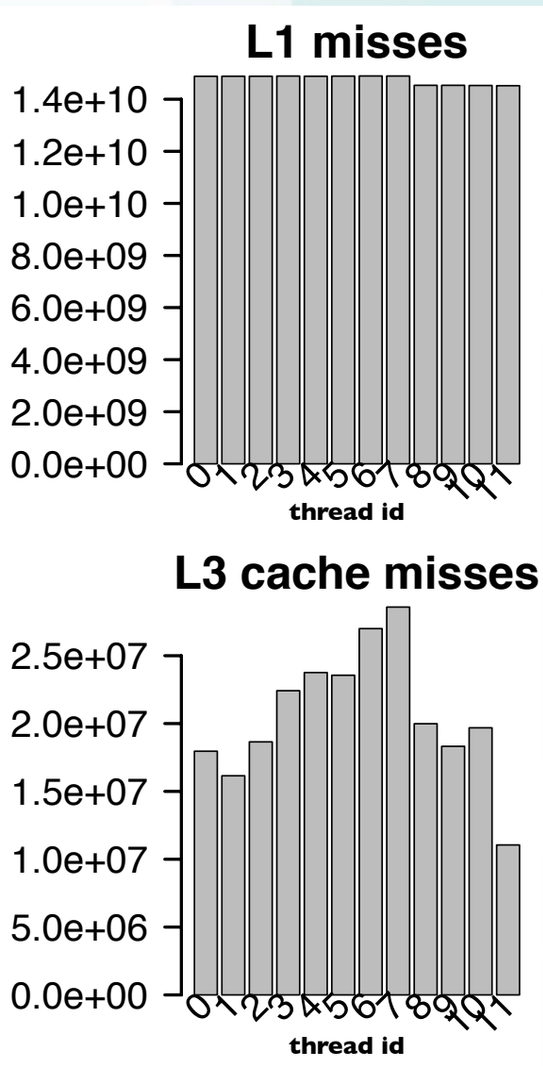
CENTROALGORITMI

AOP12 threads



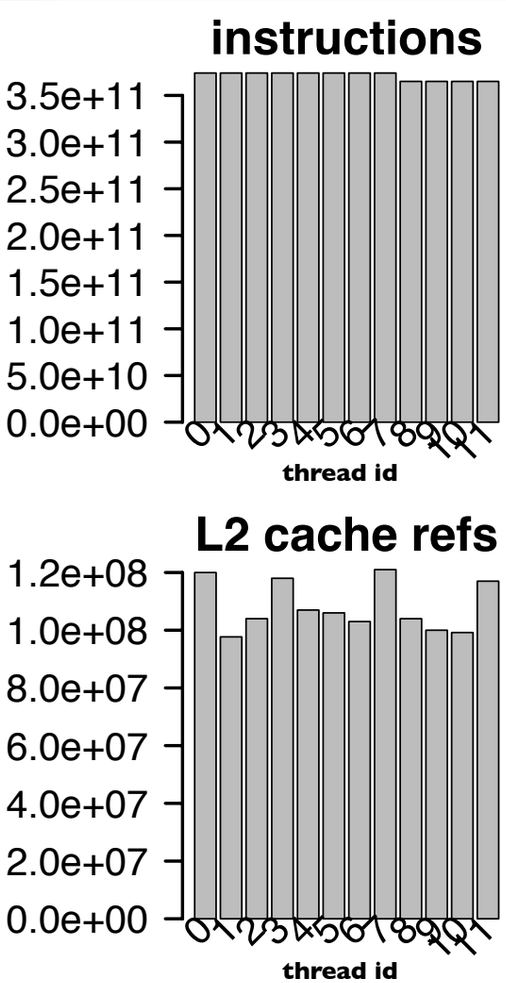


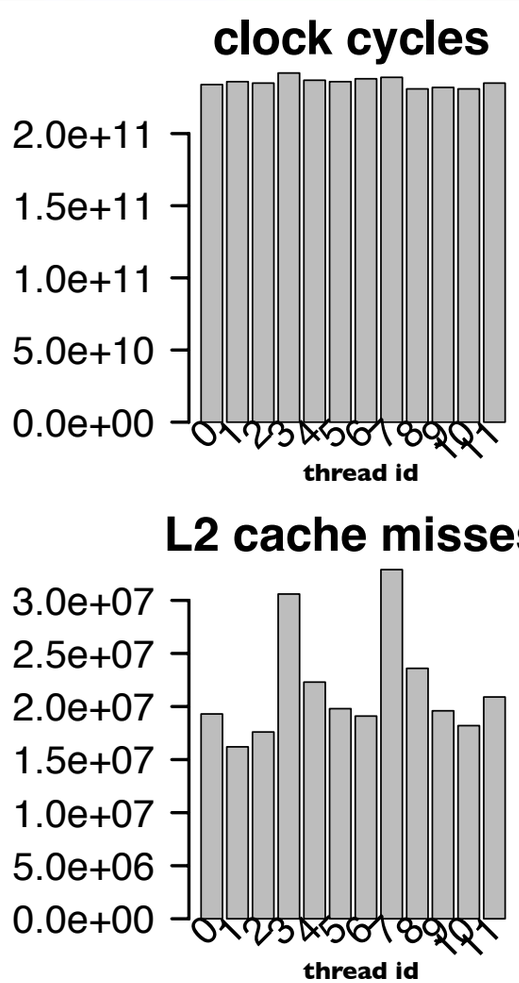


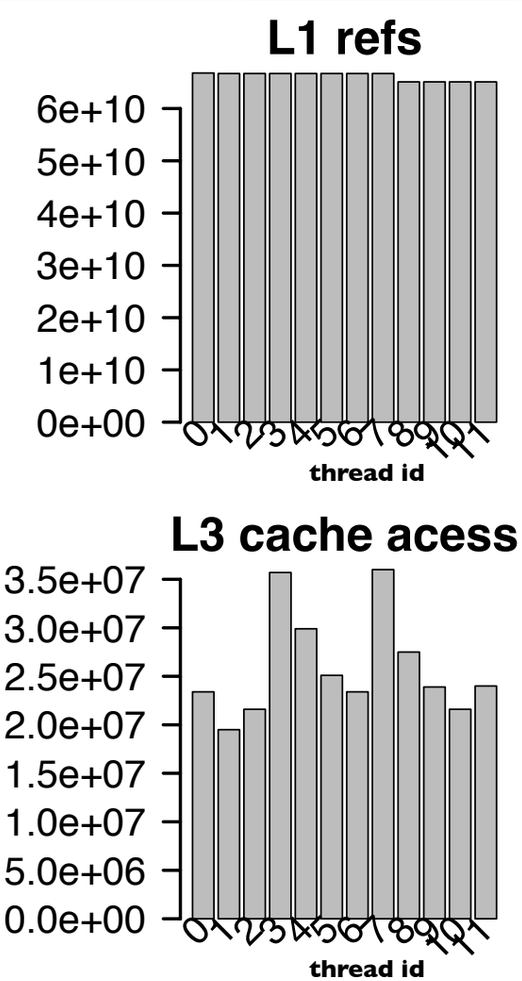


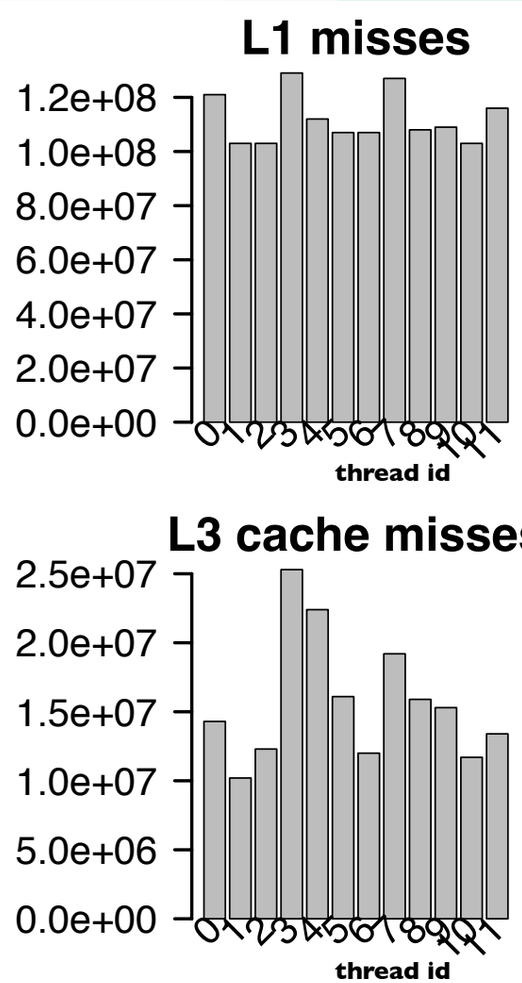
CENTROALGORITMI

SOA12 threads

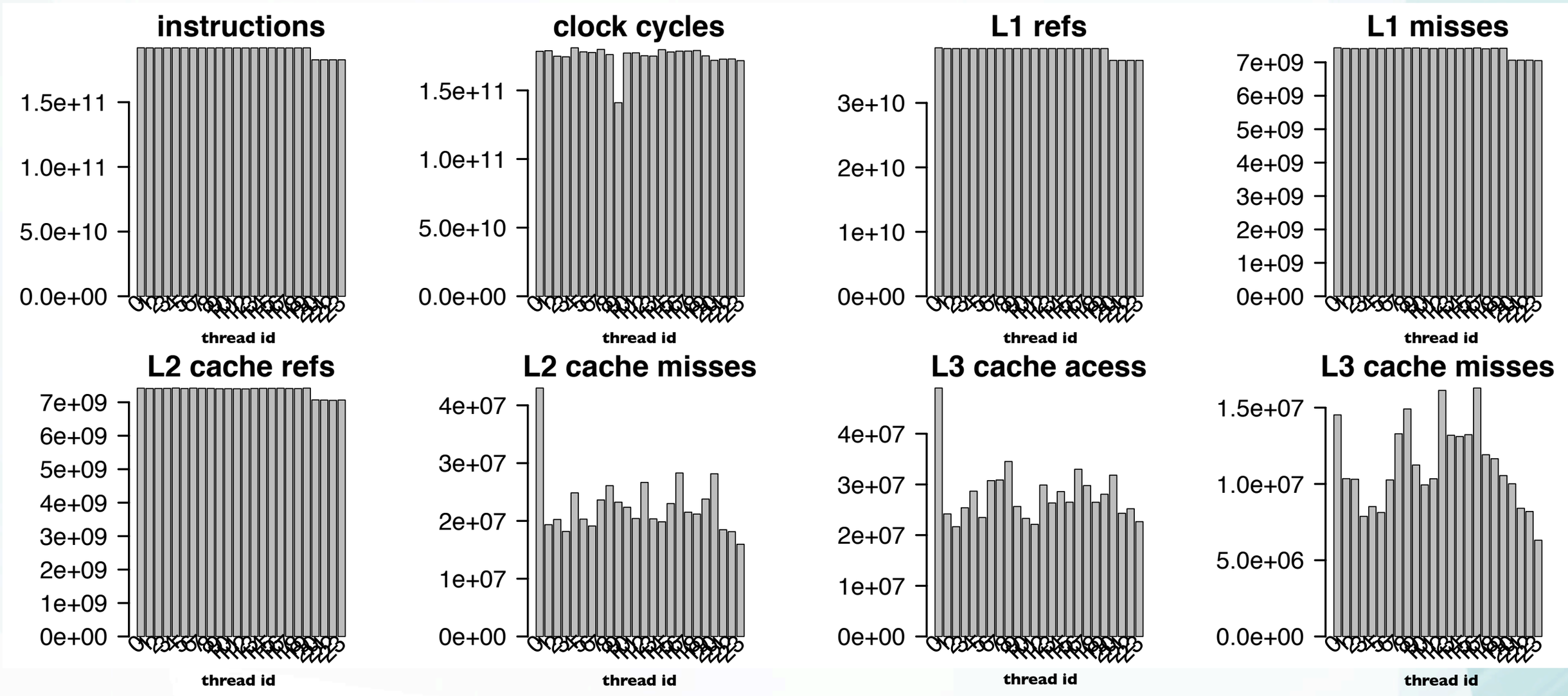






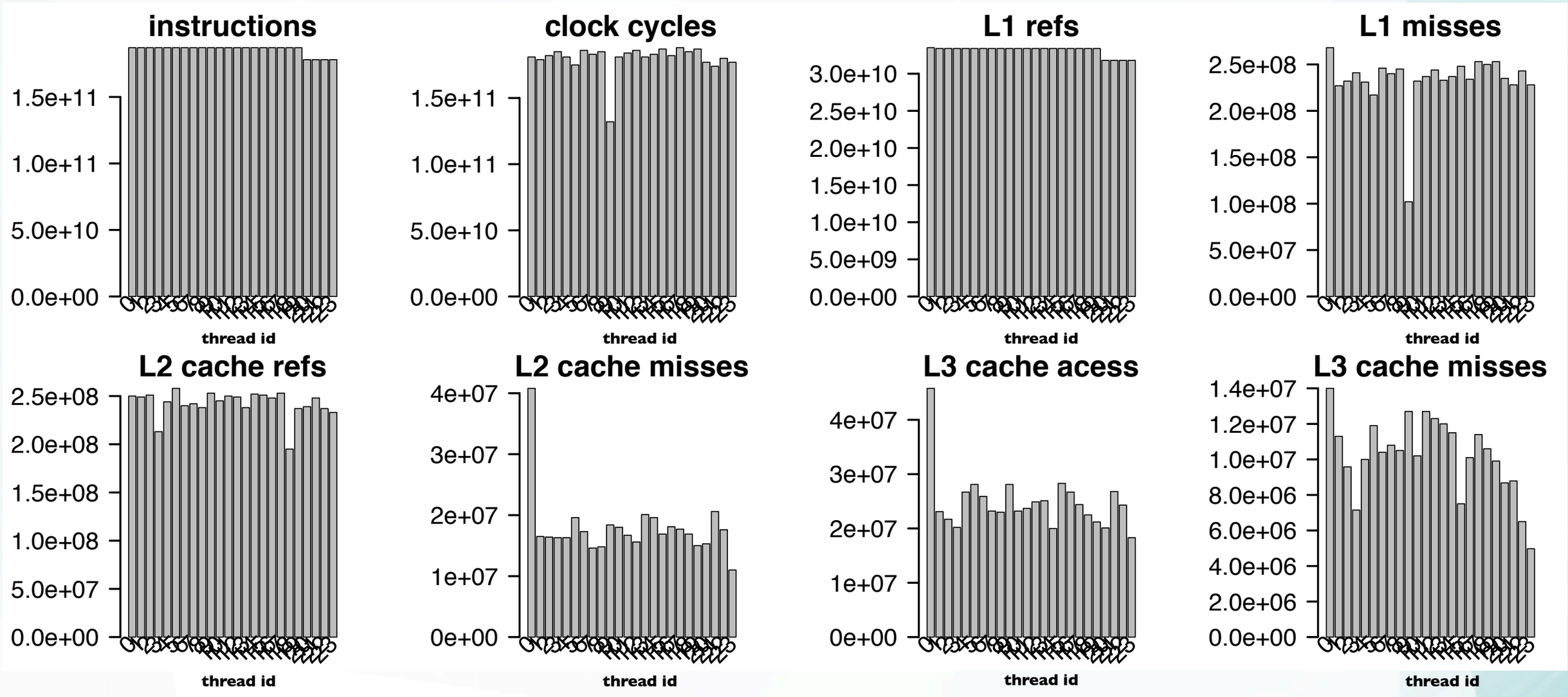


AOP 24 threads



CENTROALGORITMI

SOA 24 threads



CENTROALGORITMI

PAPIJ

CENTROALGORITMI

PAPI in Virtual Machine

CENTROALGORITMI

PAPI in Virtual Machine