THE PAPI PERFORMANCE ANALYSIS TOOL

Rui Silva

20 November 2012
Universidade do Minho

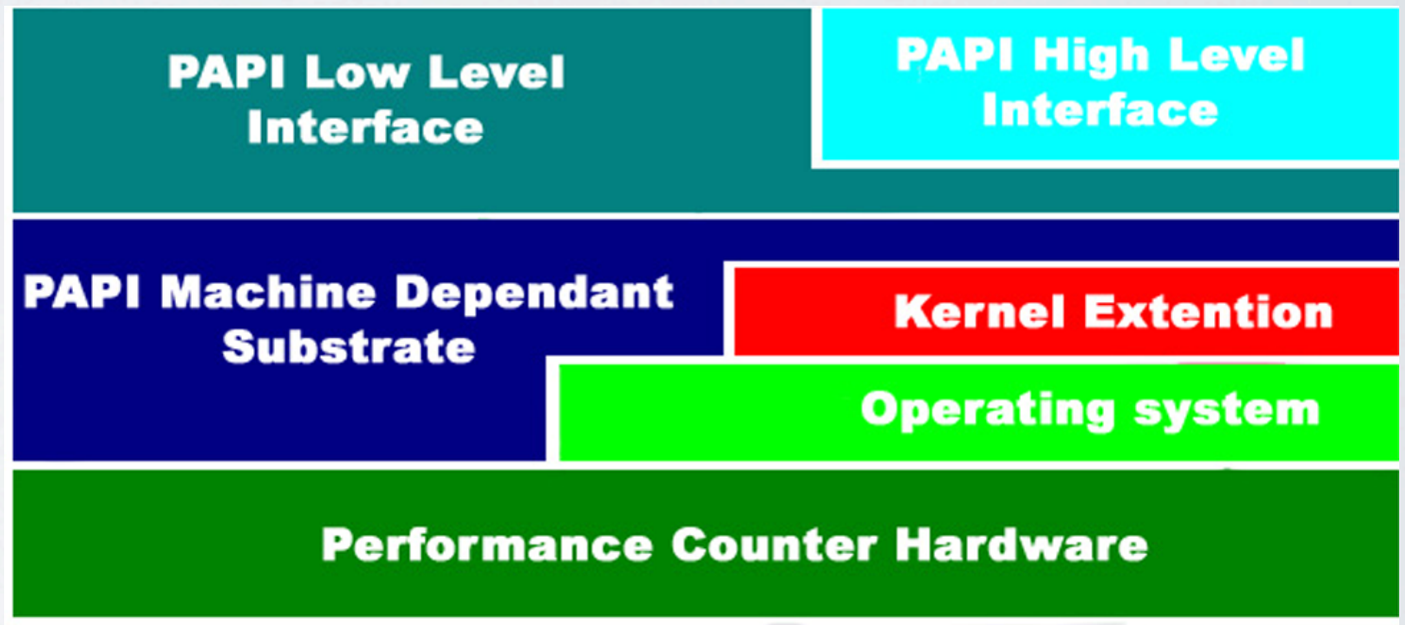
OUTLINE

- Introduction to PAPI
- My experience with PAPI

INTRODUCTION TO PAPI

PAPI

- Access to hardware performance counters
- Justification of gains of optimisations
- Identify side effects



THE RESULTS. WHAT IS IT 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...)

IMPROVE PERFORMANCE

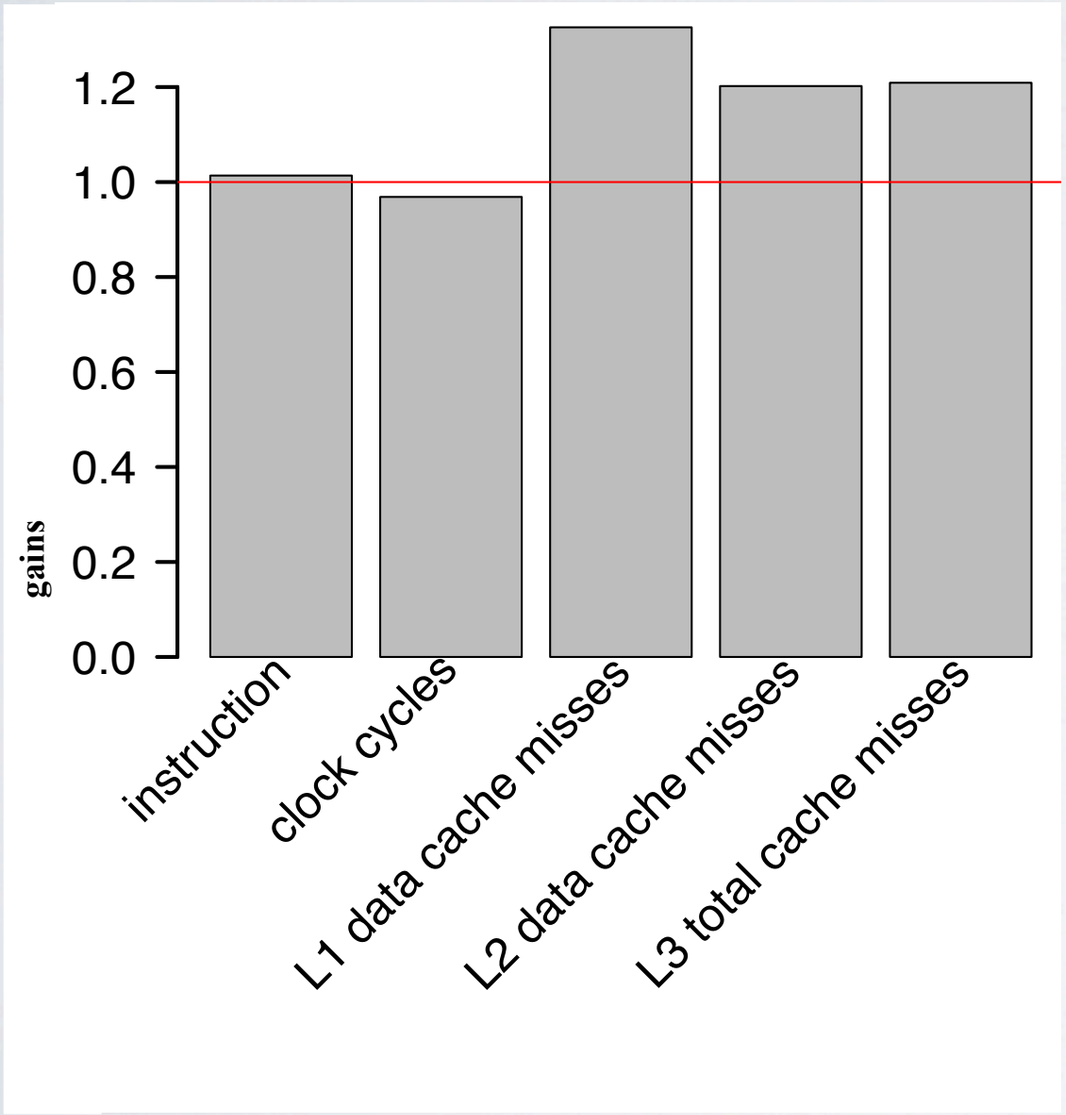
- Knowledge of the problem
- Knowledge and implementation the optimizations
- Analyse and compare performance counters
  - Justify improvements with the new values
  - Analyse side effects
  - It may be necessary to analyse more counters (using guessing/ intuition)

EXAMPLE (LOOP FUSION)

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

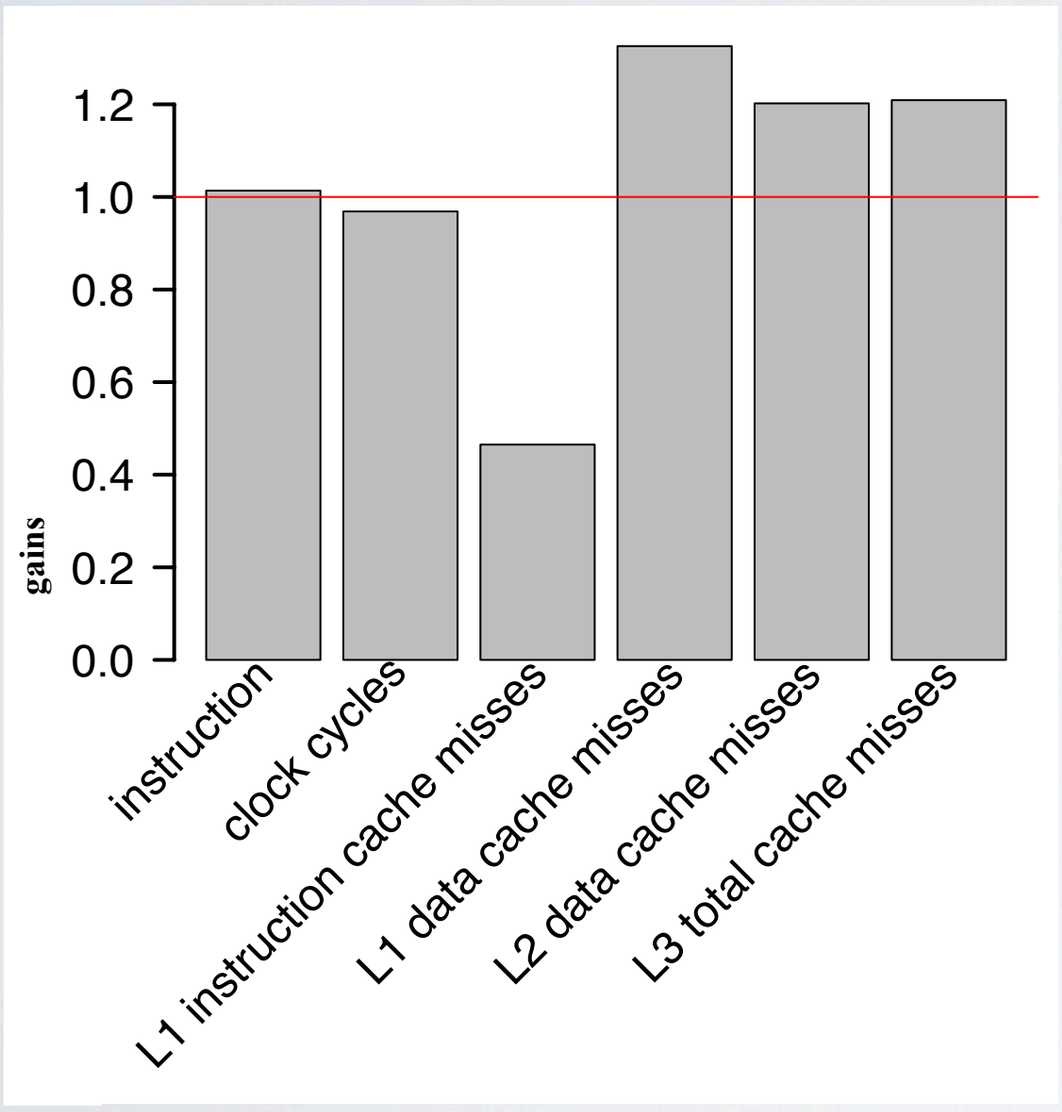
```
for i = 1 to N
    M[ i ] += C1
    M[ i ] *= C2
```

EXAMPLE (LOOP FUSION)



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

EXAMPLE (LOOP FUSION)



- More misses on access the instruction

PERFORMANCE METRICS

- CPI and misse rate
  - Hides increases in instructions and access
- CPE and Misses per element
  - Different problems, different values
- It is not possible to compare the performance of different problems

EXAMPLE (-O0 VS -O3)

	-O0	**-O3**
#I	$3.1x10^6$	$1.0x10^6$
#CC	$2.0x10^6$	$1.4x10^6$
CPI	**0.6**	**1.3**

Convolve 3x1

WHAT TO MEASURE?

- All code
  - Hide 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 II)

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

PAPI COMMAND

- papi_avail
- papi_error_codes
- papi_cost
- papi_mem_info
- papi_native_avail

MY EXPERIENCE WITH PAPI

COUNTERS

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

CASE STUDIES

- Molecular dynamics simulation
- Matrix Multiplication
- Others

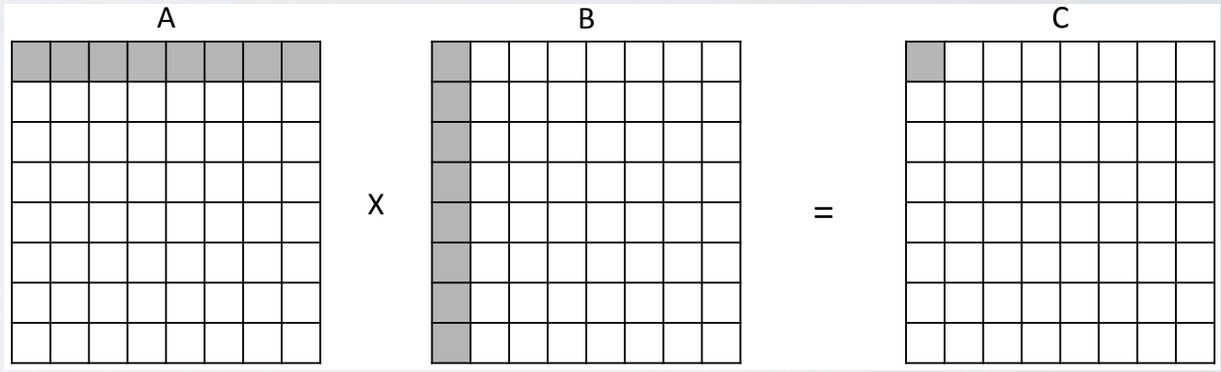
PAPI
SEQUENTIAL VERSIONS

LOOP REORDER

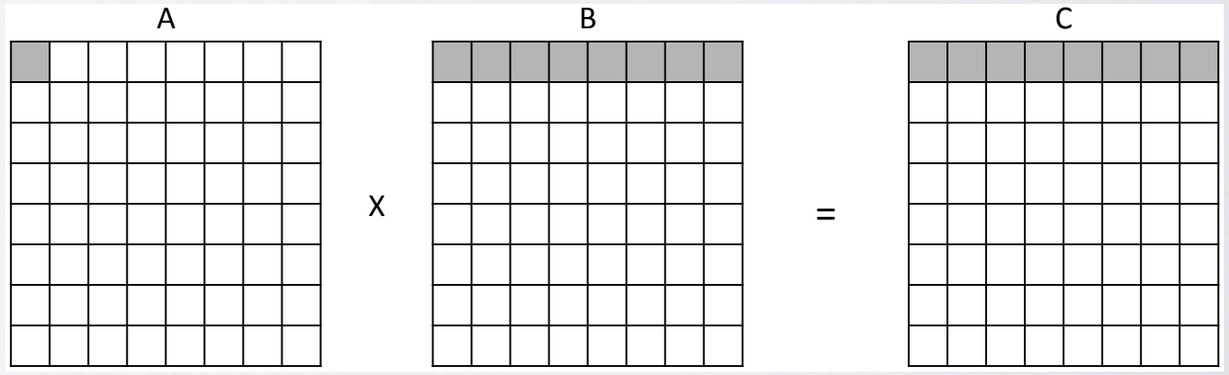
LOOP REORDER

```
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

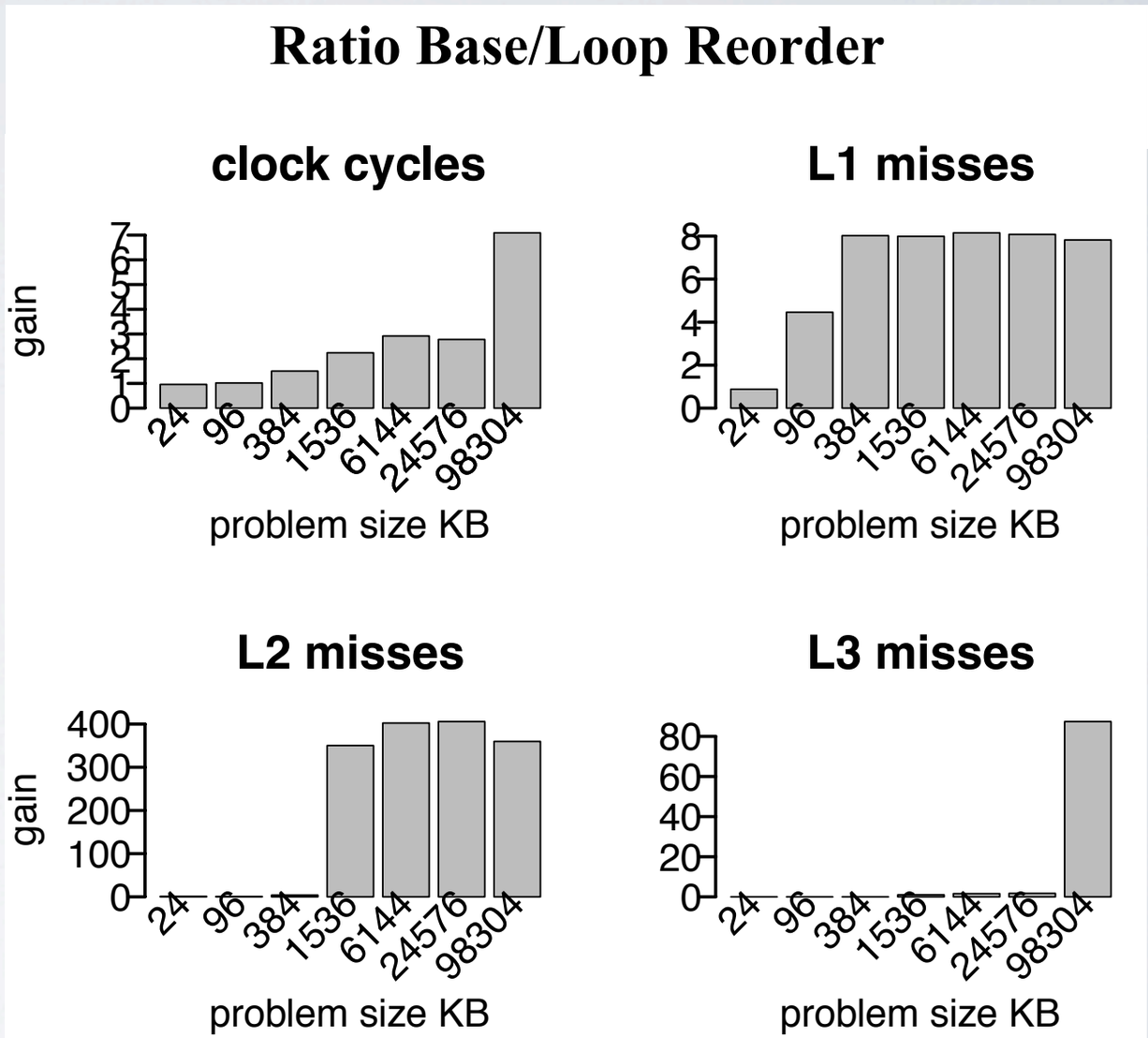


IKJ



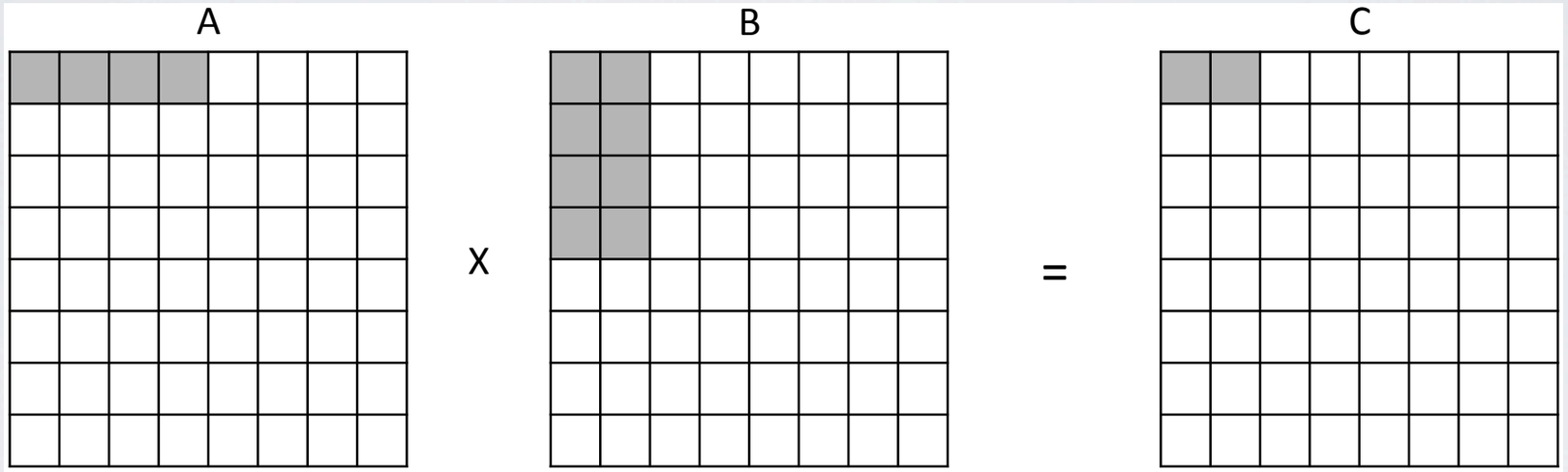
LOOP REORDER (MM)

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



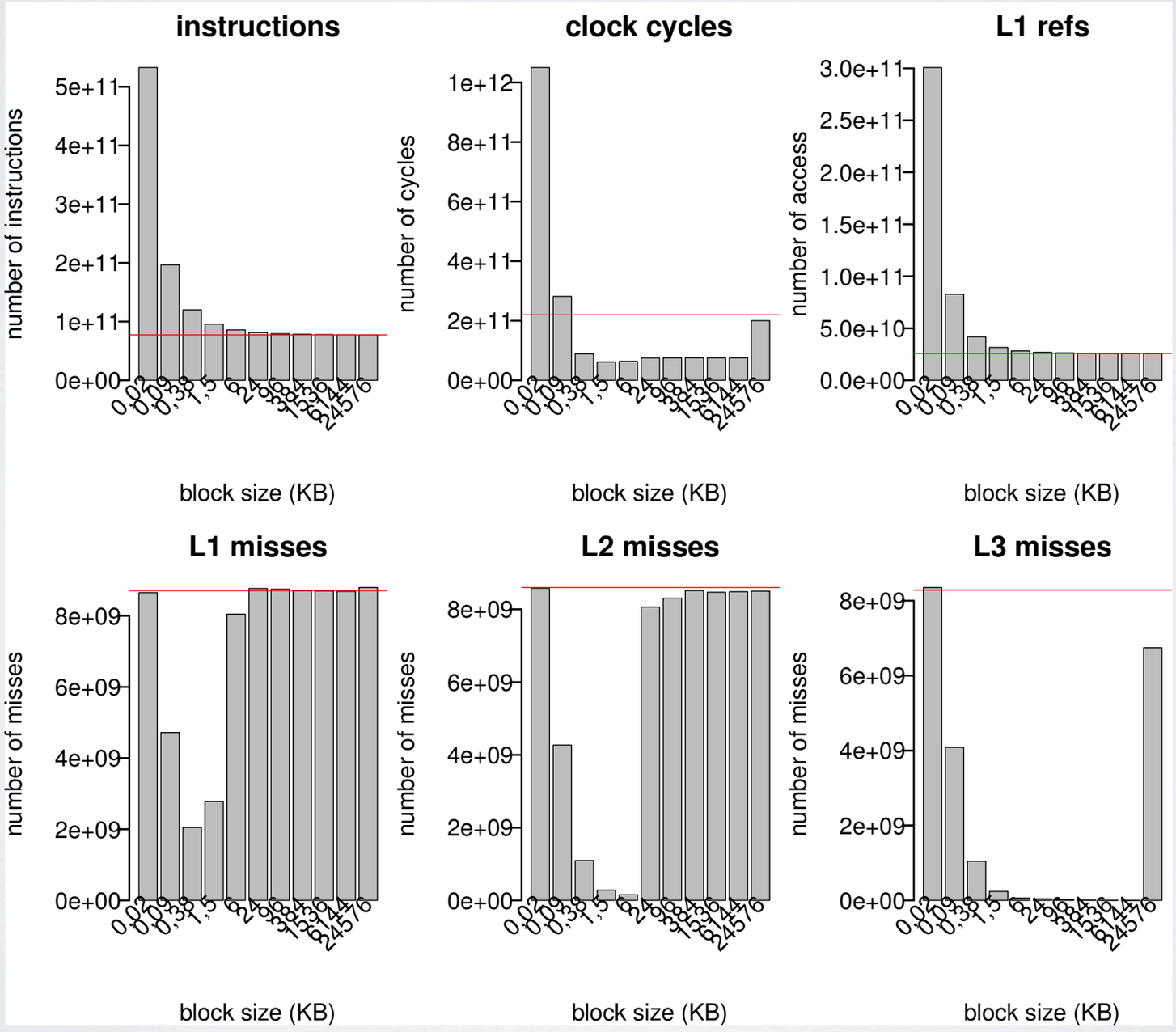
LOOP TILING

MATRIX MULTIPLICATION



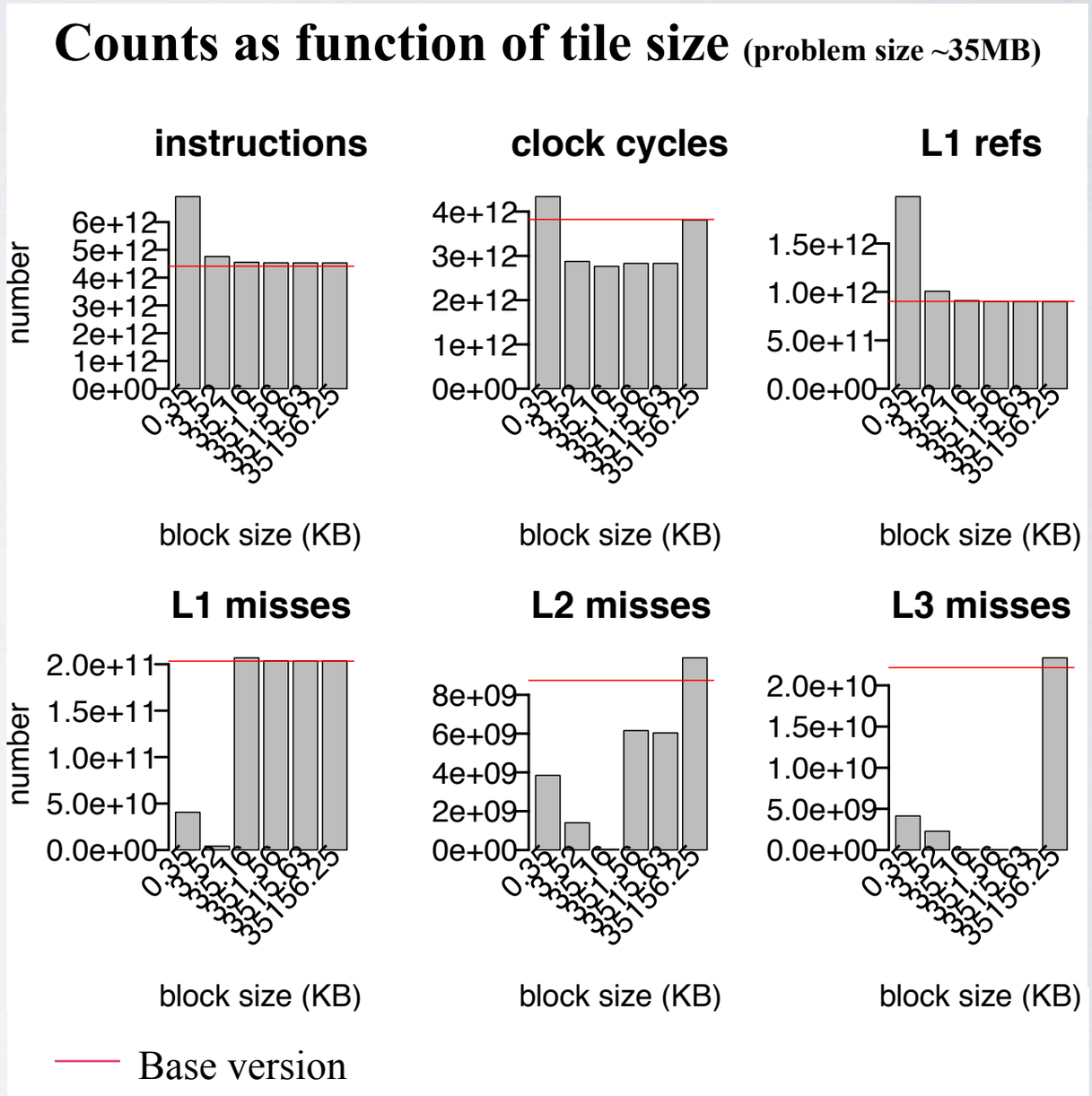
- Matrix computations by blocks
  - Block size is adjusted to different levels of the cache

MATRIX MULTIPLICATION



LOOP TILING (MD)

- 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



PAPI
PARALLEL VERSIONS

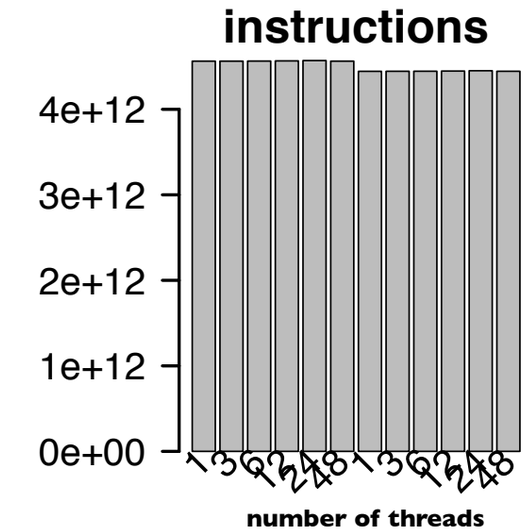
CODE IN PARALLEL VERSIONS (I)

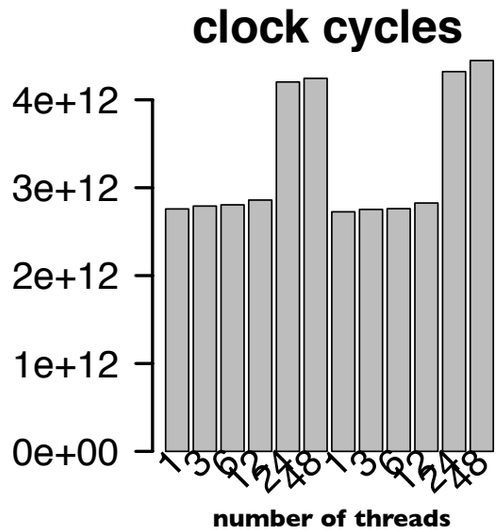
```
void runiters(MD *md, Particles *particulas) {

    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();
    }
}
```

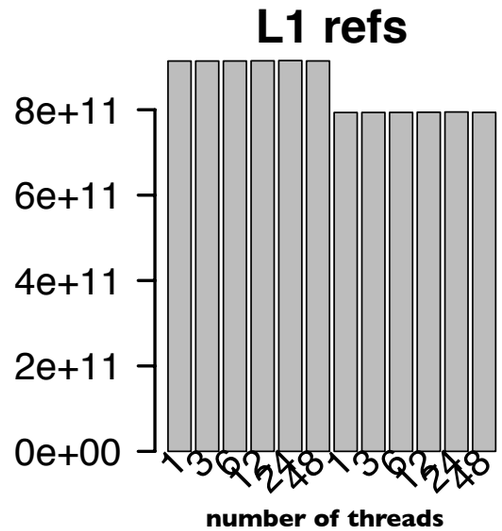
CODE IN PARALLEL VERSIONS (2)

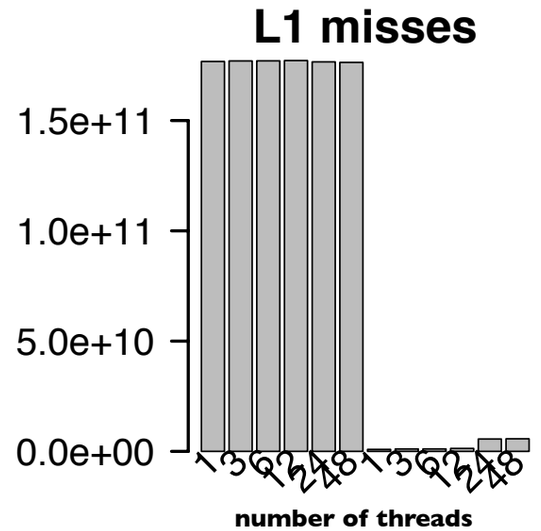
```
int main(int argc, char **argv) {

    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);

}
```

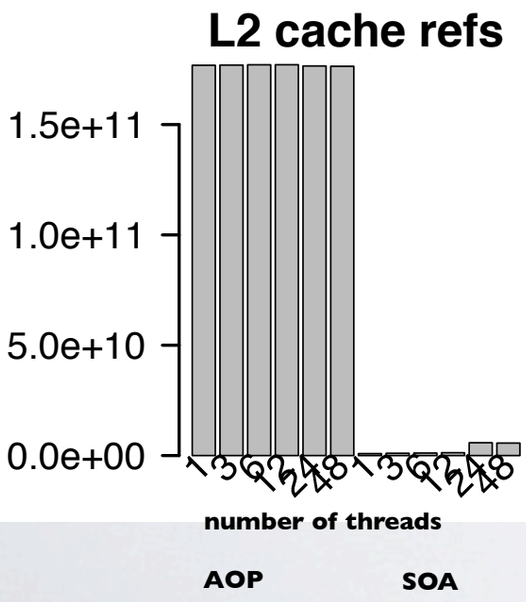
MD (AOP, SOA)

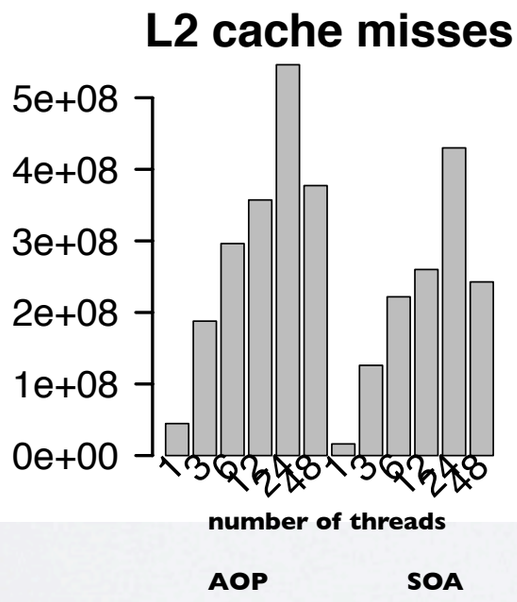

instructions
4e+12
3e+12
2e+12
1e+12
0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads

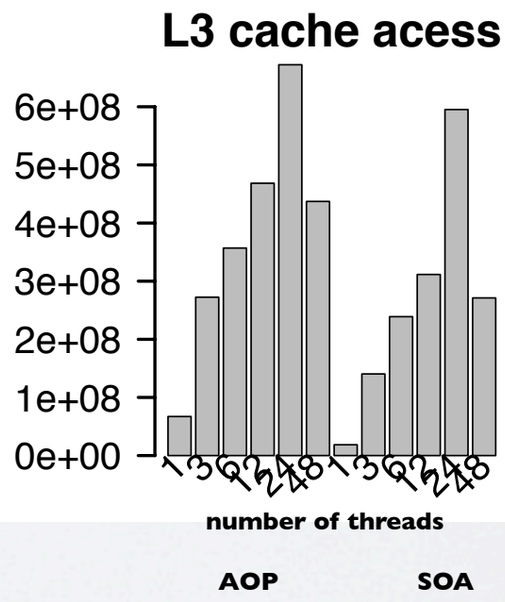

clock cycles
4e+12
3e+12
2e+12
1e+12
0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads

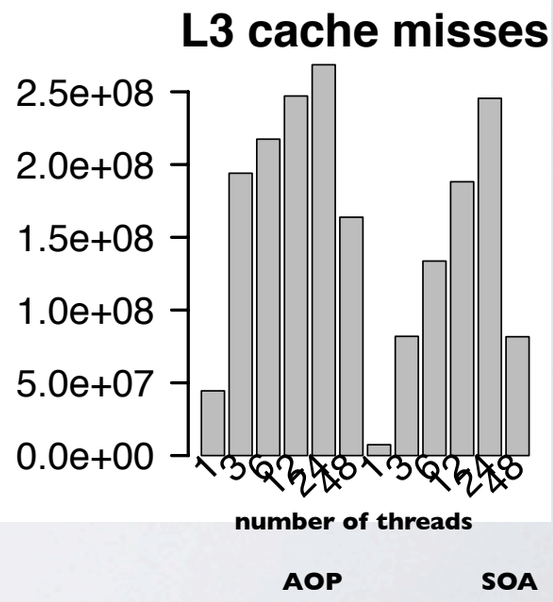

L1 refs
8e+11
6e+11
4e+11
2e+11
0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads


L1 misses
1.5e+11
1.0e+11
5.0e+10
0.0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads

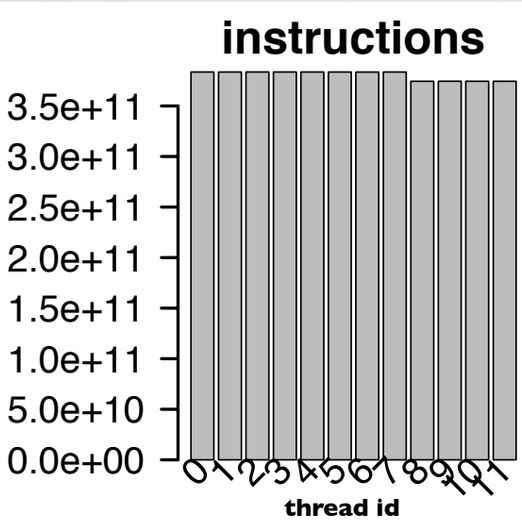

L2 cache refs
1.5e+11
1.0e+11
5.0e+10
0.0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads
AOP
SOA

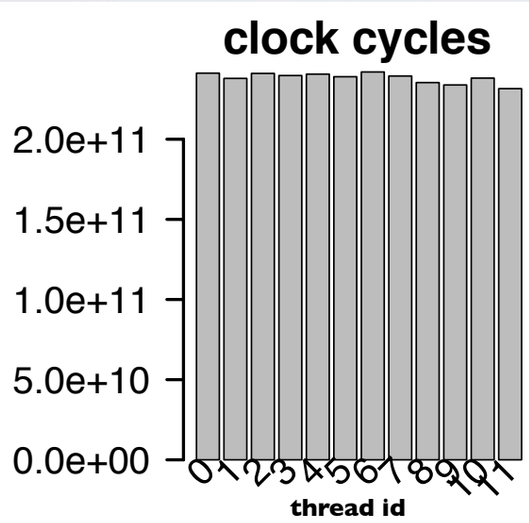

L2 cache misses
5e+08
4e+08
3e+08
2e+08
1e+08
0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads
AOP
SOA

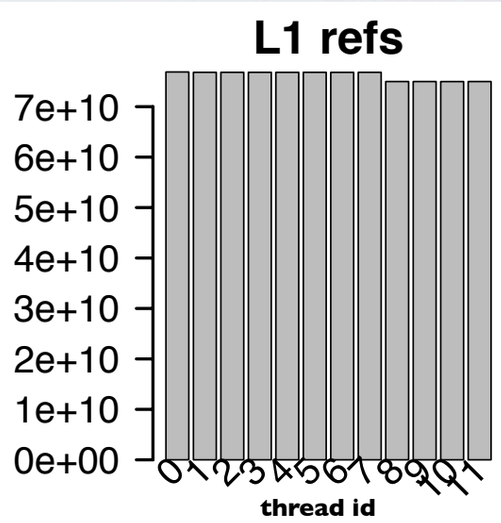

L3 cache acess
6e+08
5e+08
4e+08
3e+08
2e+08
1e+08
0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads
AOP
SOA

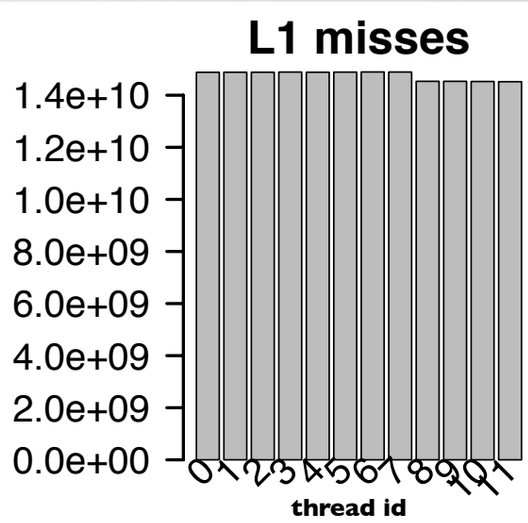

L3 cache misses
2.5e+08
2.0e+08
1.5e+08
1.0e+08
5.0e+07
0.0e+00
1 3 6 12 24 8 1 3 6 12 24 8
number of threads
AOP
SOA

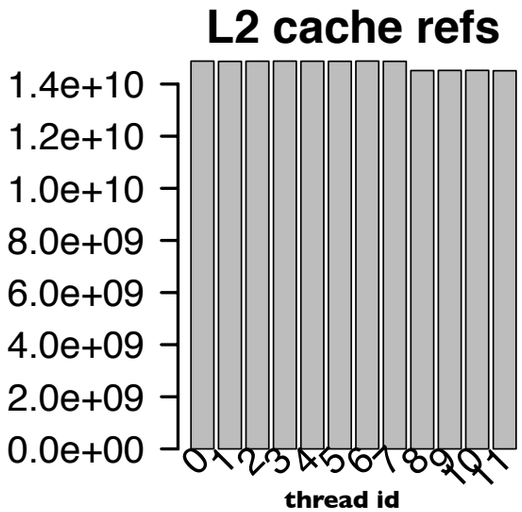
AOP12 THREADS

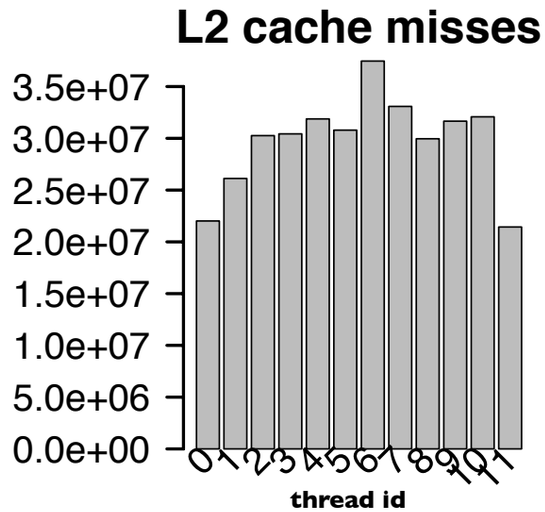


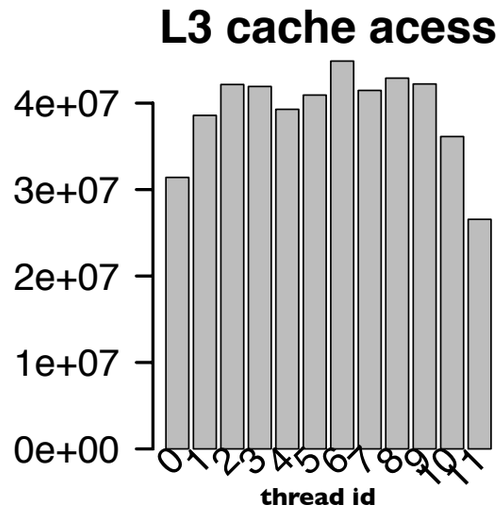


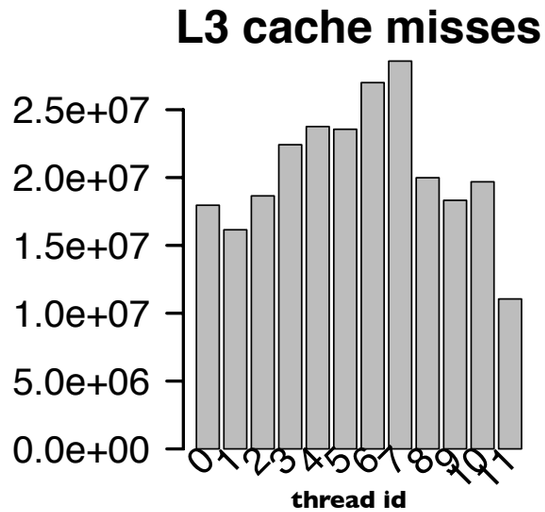




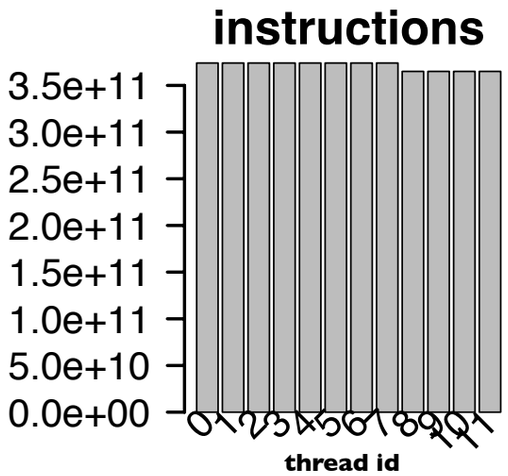


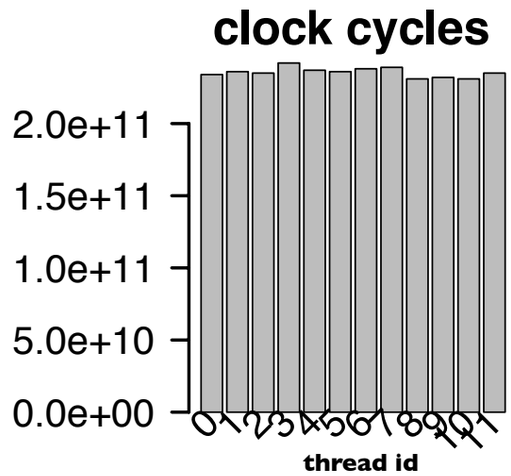


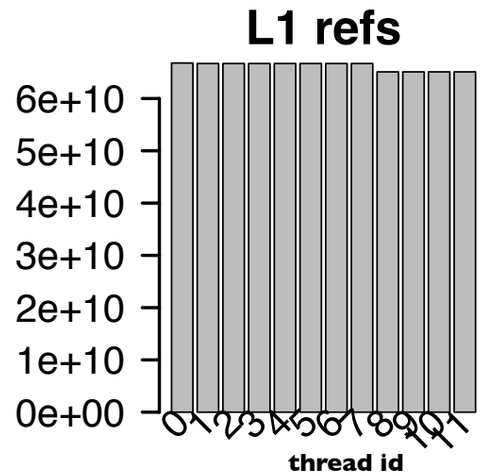


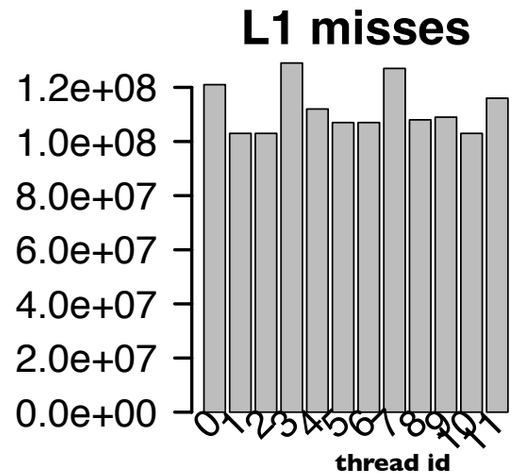


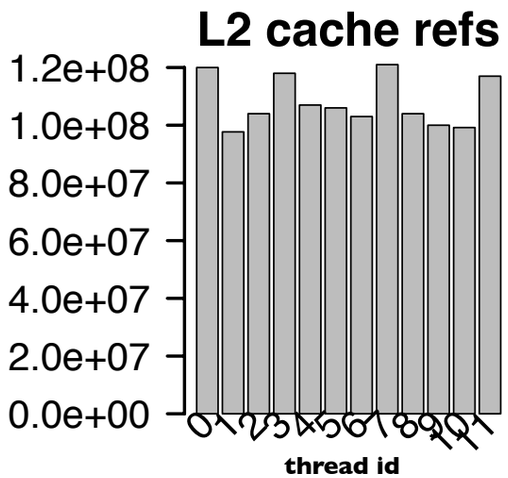
SOA12 THREADS

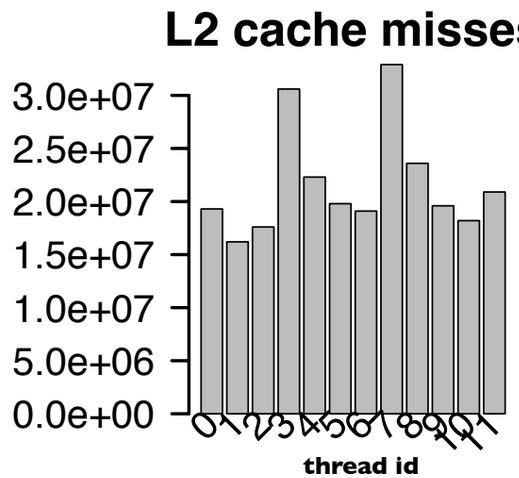


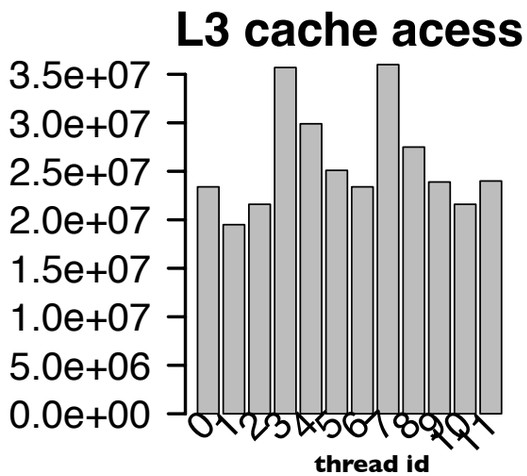


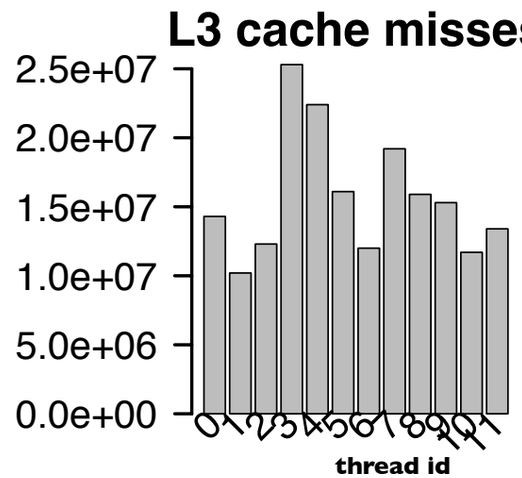




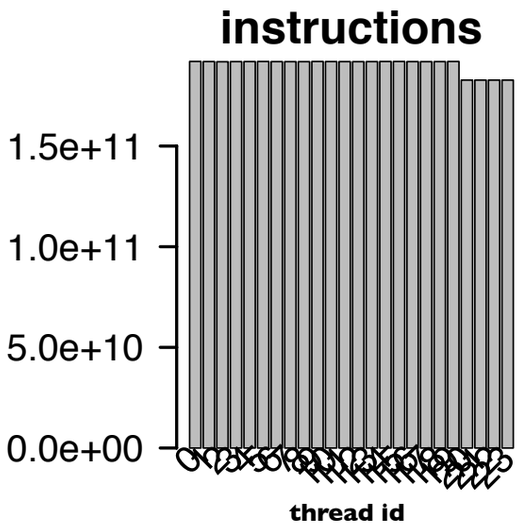


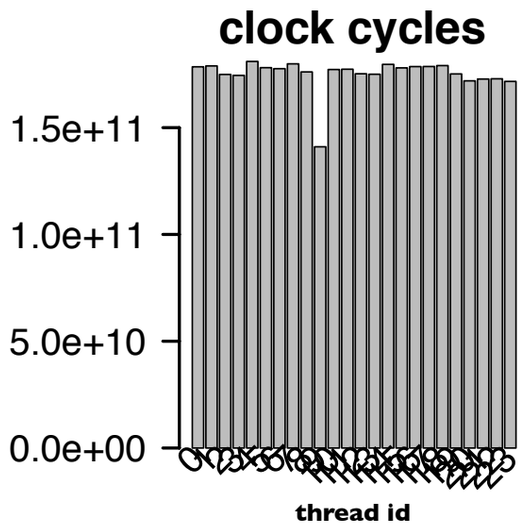


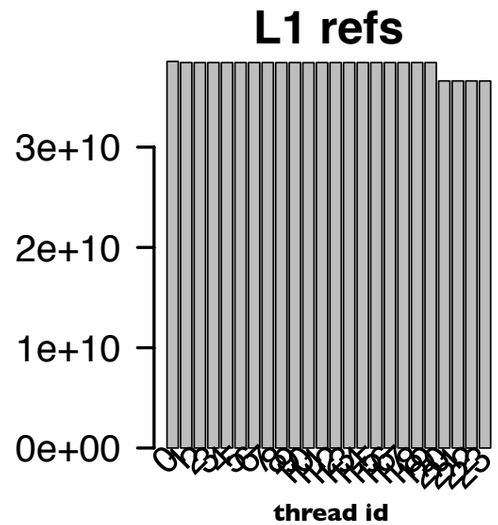


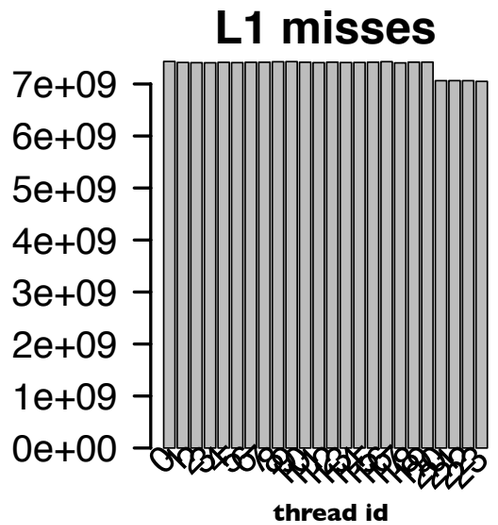


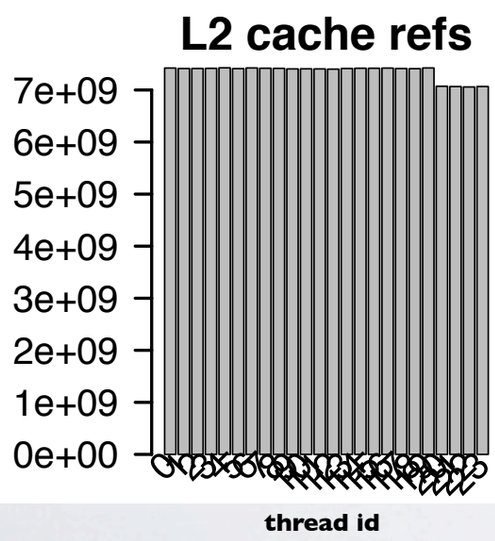
AOP 24 THREADS

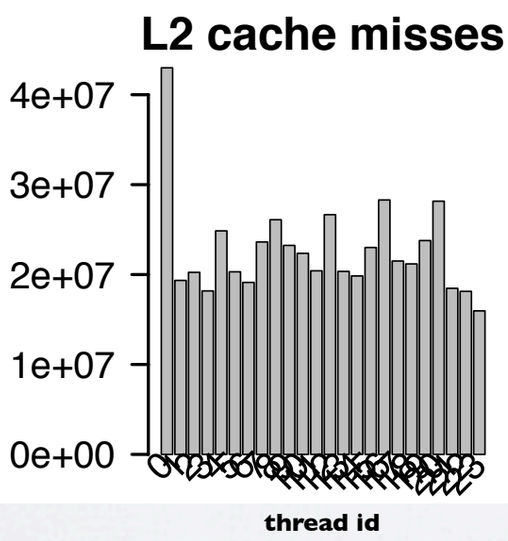


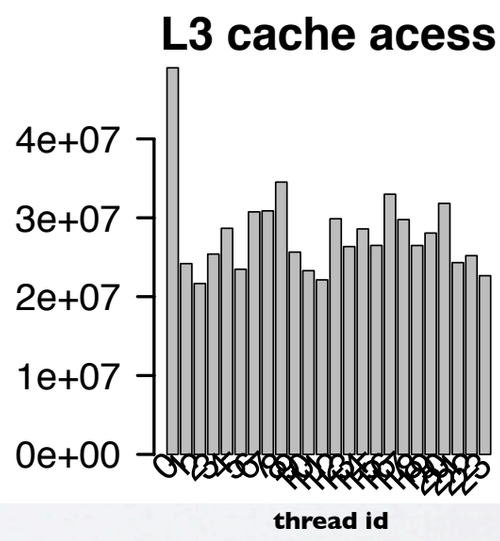


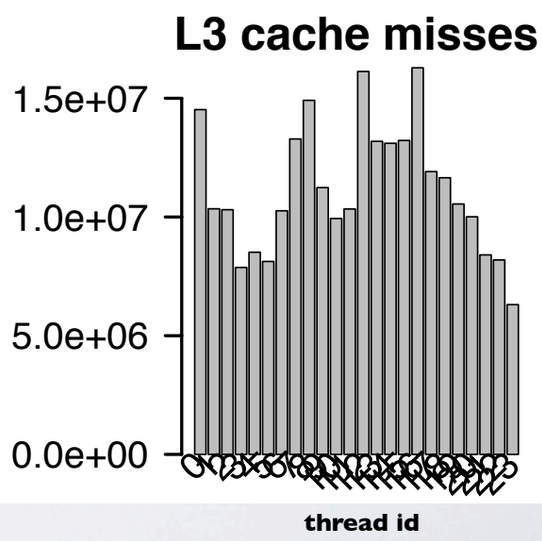




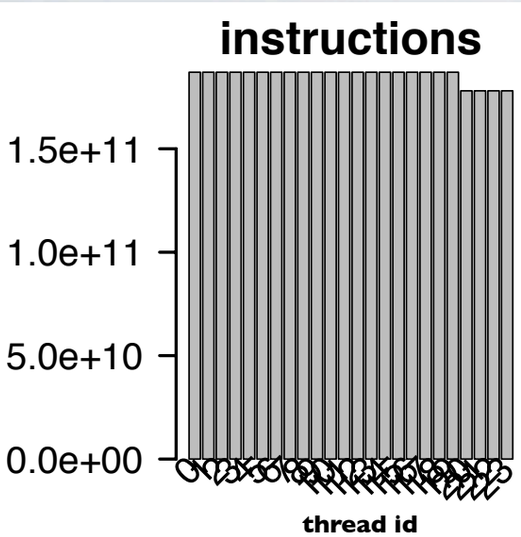


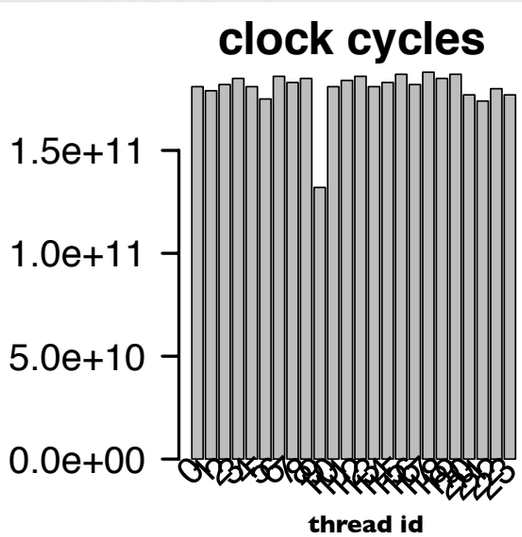


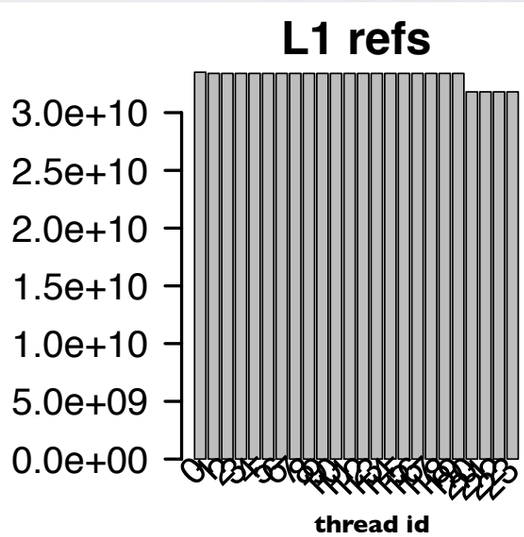


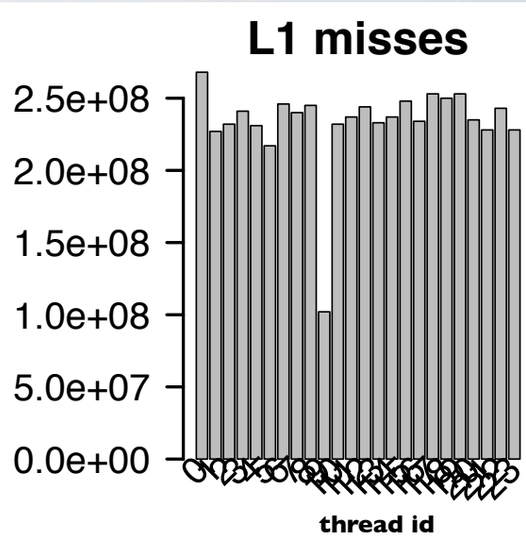


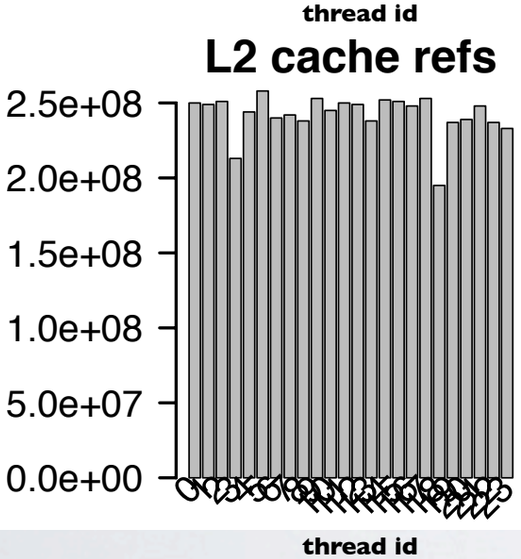
SOA 24 THREADS

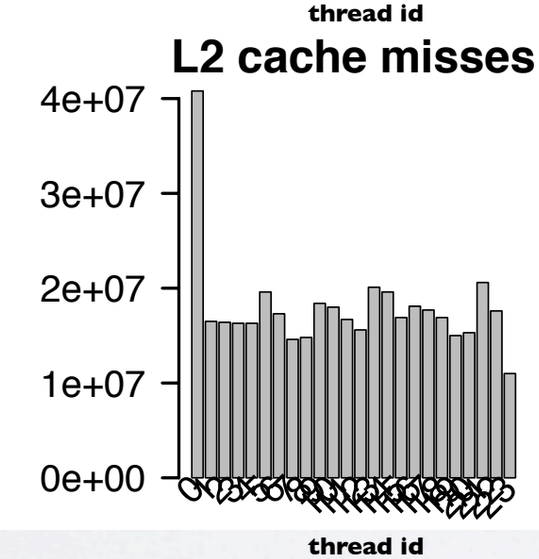


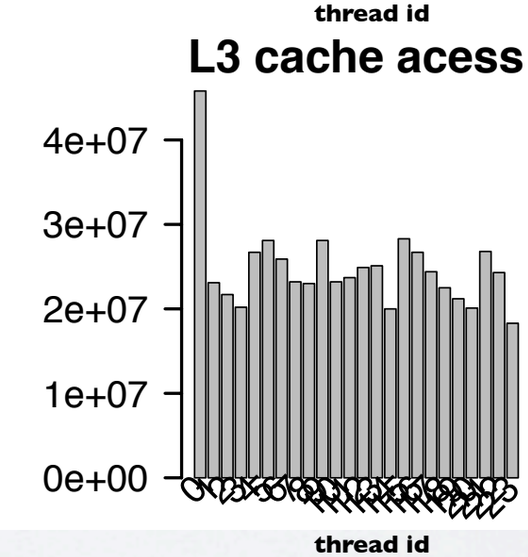


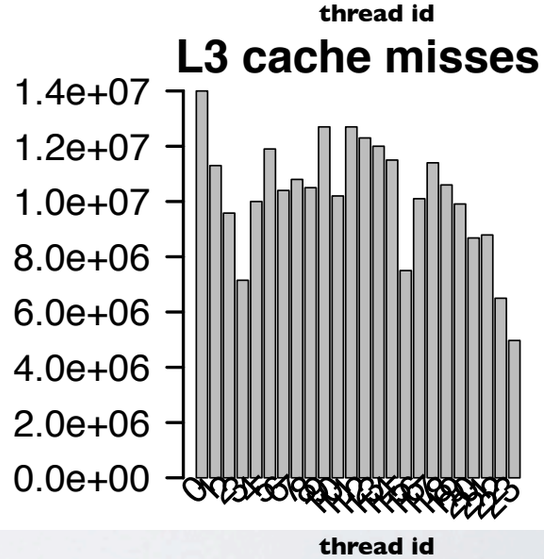












PAPI IN VIRTUAL MACHINE