CS546 Parallel and Distributed Processing
Program Assignment 4

Exercise 1: Please list all available CPU counters with the perf list command

- The perf list Command lists all the available CPU counters,
- List of pre-defined events (to be used in -e):

```
branch-instructions OR branches                    [Hardware event]
  branch-misses                                    [Hardware event]
  bus-cycles                                       [Hardware event]
  cache-misses                                     [Hardware event]
  cache-references                                 [Hardware event]
  cpu-cycles OR cycles                             [Hardware event]
  instructions                                     [Hardware event]
  ref-cycles                                       [Hardware event]

  alignment-faults                                 [Software event]
  bpf-output                                       [Software event]
  context-switches OR cs                           [Software event]
  cpu-clock                                        [Software event]
  cpu-migrations OR migrations                     [Software event]
  dummy                                            [Software event]
  emulation-faults                                 [Software event]
  major-faults                                     [Software event]
  minor-faults                                     [Software event]
  page-faults OR faults                            [Software event]
  task-clock                                       [Software event]

  L1-dcache-load-misses                            [Hardware cache event]
  L1-dcache-loads                                  [Hardware cache event]
  L1-dcache-stores                                 [Hardware cache event]
  L1-icache-load-misses                            [Hardware cache event]
  LLC-load-misses                                  [Hardware cache event]
  LLC-loads                                        [Hardware cache event]
  LLC-store-misses                                 [Hardware cache event]
  LLC-stores                                       [Hardware cache event]
  branch-load-misses                               [Hardware cache event]
  branch-loads                                     [Hardware cache event]
  dTLB-load-misses                                 [Hardware cache event]
  dTLB-loads                                       [Hardware cache event]
  dTLB-store-misses                                [Hardware cache event]
  dTLB-stores                                      [Hardware cache event]
  iTLB-load-misses                                 [Hardware cache event]
  iTLB-loads                                       [Hardware cache event]
```

node-load-misses [Hardware cache event]
node-loads [Hardware cache event]
node-store-misses [Hardware cache event]
node-stores [Hardware cache event]

branch-instructions OR cpu/branch-instructions/ [Kernel PMU event]
branch-misses OR cpu/branch-misses/ [Kernel PMU event]
bus-cycles OR cpu/bus-cycles/ [Kernel PMU event]
cache-misses OR cpu/cache-misses/ [Kernel PMU event]
cache-references OR cpu/cache-references/ [Kernel PMU event]
cpu-cycles OR cpu/cpu-cycles/ [Kernel PMU event]
cstate_core/c3-residency/ [Kernel PMU event]
cstate_core/c6-residency/ [Kernel PMU event]
cstate_core/c7-residency/ [Kernel PMU event]
cstate_pkg/c2-residency/ [Kernel PMU event]
cstate_pkg/c3-residency/ [Kernel PMU event]
cstate_pkg/c6-residency/ [Kernel PMU event]
cstate_pkg/c7-residency/ [Kernel PMU event]
cycles-ct OR cpu/cycles-ct/ [Kernel PMU event]
cycles-t OR cpu/cycles-t/ [Kernel PMU event]
el-abort OR cpu/el-abort/ [Kernel PMU event]
el-capacity OR cpu/el-capacity/ [Kernel PMU event]
el-commit OR cpu/el-commit/ [Kernel PMU event]
el-conflict OR cpu/el-conflict/ [Kernel PMU event]
el-start OR cpu/el-start/ [Kernel PMU event]
instructions OR cpu/instructions/ [Kernel PMU event]
intel_bts// [Kernel PMU event]
intel_cqm/llc_occupancy/ [Kernel PMU event]
intel_cqm/local_bytes/ [Kernel PMU event]
intel_cqm/total_bytes/ [Kernel PMU event]
intel_pt// [Kernel PMU event]
mem-loads OR cpu/mem-loads/ [Kernel PMU event]
mem-stores OR cpu/mem-stores/ [Kernel PMU event]
msr/aperf/ [Kernel PMU event]
msr/mperf/ [Kernel PMU event]
msr/smi/ [Kernel PMU event]
msr/tsc/ [Kernel PMU event]
tx-abort OR cpu/tx-abort/ [Kernel PMU event]
tx-capacity OR cpu/tx-capacity/ [Kernel PMU event]
tx-commit OR cpu/tx-commit/ [Kernel PMU event]
tx-conflict OR cpu/tx-conflict/ [Kernel PMU event]
tx-start OR cpu/tx-start/ [Kernel PMU event]
uncore_imc_0/cas_count_read/ [Kernel PMU event]
uncore_imc_0/cas_count_write/ [Kernel PMU event]
uncore_imc_0/clockticks/ [Kernel PMU event]
uncore_imc_1/cas_count_read/ [Kernel PMU event]

```
  uncore_imc_1/cas_count_write/                [Kernel PMU event]
  uncore_imc_1/clockticks/                     [Kernel PMU event]
  uncore_imc_2/cas_count_read/                 [Kernel PMU event]
  uncore_imc_2/cas_count_write/                [Kernel PMU event]
  uncore_imc_2/clockticks/                     [Kernel PMU event]
  uncore_imc_3/cas_count_read/                 [Kernel PMU event]
  uncore_imc_3/cas_count_write/                [Kernel PMU event]
  uncore_imc_3/clockticks/                     [Kernel PMU event]
  uncore_imc_4/cas_count_read/                 [Kernel PMU event]
  uncore_imc_4/cas_count_write/                [Kernel PMU event]
  uncore_imc_4/clockticks/                     [Kernel PMU event]

  rNNN                                         [Raw hardware event descriptor]
  cpu/t1=v1[,t2=v2,t3 ...]/modifier            [Raw hardware event descriptor]
   (see 'man perf-list' on how to encode it)

  mem:<addr>[/len][:access]                    [Hardware breakpoint]
```

Exercise 2: (35 points) Please use the perf stat command instrument and summarize selected CPU counters:

Usage: **perf stat** [<options>] [<command>]

```
    -a, --all-cpus                        system-wide collection from all CPUs
    -A, --no-aggr                         disable CPU count aggregation
    -B, --big-num                         print large numbers with thousands' separators
    -C, --cpu <cpu>                       list of cpus to monitor in system-wide
    -c, --scale                           scale/normalize counters
    -D, --delay <n>                       ms to wait before starting measurement after program start
    -d, --detailed                        detailed run - start a lot of events
    -e, --event <event>                   event selector. use 'perf list' to list available events
    -G, --cgroup <name>                   monitor event in cgroup name only
    -g, --group                           put the counters into a counter group
    -I, --interval-print <n>              print counts at regular interval in ms (>= 10)
    -i, --no-inherit                      child tasks do not inherit counters
    -n, --null                            null run - dont start any counters
    -o, --output <file>                   output file name
    -p, --pid <pid>                       stat events on existing process id
    -r, --repeat <n>                      repeat command and print average + stddev (max: 100, forever:
    -S, --sync                            call sync() before starting a run
    -t, --tid <tid>                       stat events on existing thread id
    -T, --transaction                     hardware transaction statistics
    -v, --verbose                         be more verbose (show counter open errors, etc)
    -x, --field-separator <separator>     print counts with custom separator
        --append                          append to the output file
        --filter <filter>                 event filter
```

--log-fd <n>	log output to fd, instead of stderr
--per-core	aggregate counts per physical processor core
--per-socket	aggregate counts per processor socket
--per-thread	aggregate counts per thread
--post <command>	command to run after to the measured command
--pre <command>	command to run prior to the measured command

1. **Compile the code: (the naive approach of performing matrix-matrix multiplication with i,j,k order).**

```
gbairi@fusion1:~$  g++ -g -fno-omit-frame-pointer -O3 -DNAIVE matmul_2D.cpp -o mm_naive.out
```

2. **Run perf stat**:

```
gbairi@fusion1:~$ perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-stores ./mm_naive.out 500
using matrix size:500
```

3. **Record the numbers**

```
gbairi@fusion1:~$ perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-stores ./mm_naive.out 500
using matrix size:500

 Performance counter stats for './mm_naive.out 500':

     5,786,320,904      cpu-cycles
    10,056,646,808      instructions              #    1.74  insns per cycle
     3,765,438,022      L1-dcache-loads
     1,506,606,244      L1-dcache-load-misses     #   40.01% of all L1-dcache hits
     1,258,142,279      L1-dcache-stores

       2.136062973 seconds time elapsed
```

4. **Compile the code: (the interchange approach of performing matrix-matri multiplication with i,k,j order**

```
gbairi@fusion1:~$ g++ -g -fno-omit-frame-pointer -O3 -DINTERCHANGE matmul_2D.cpp -o mm_interchange.out
```

5. **Run perf stat:**

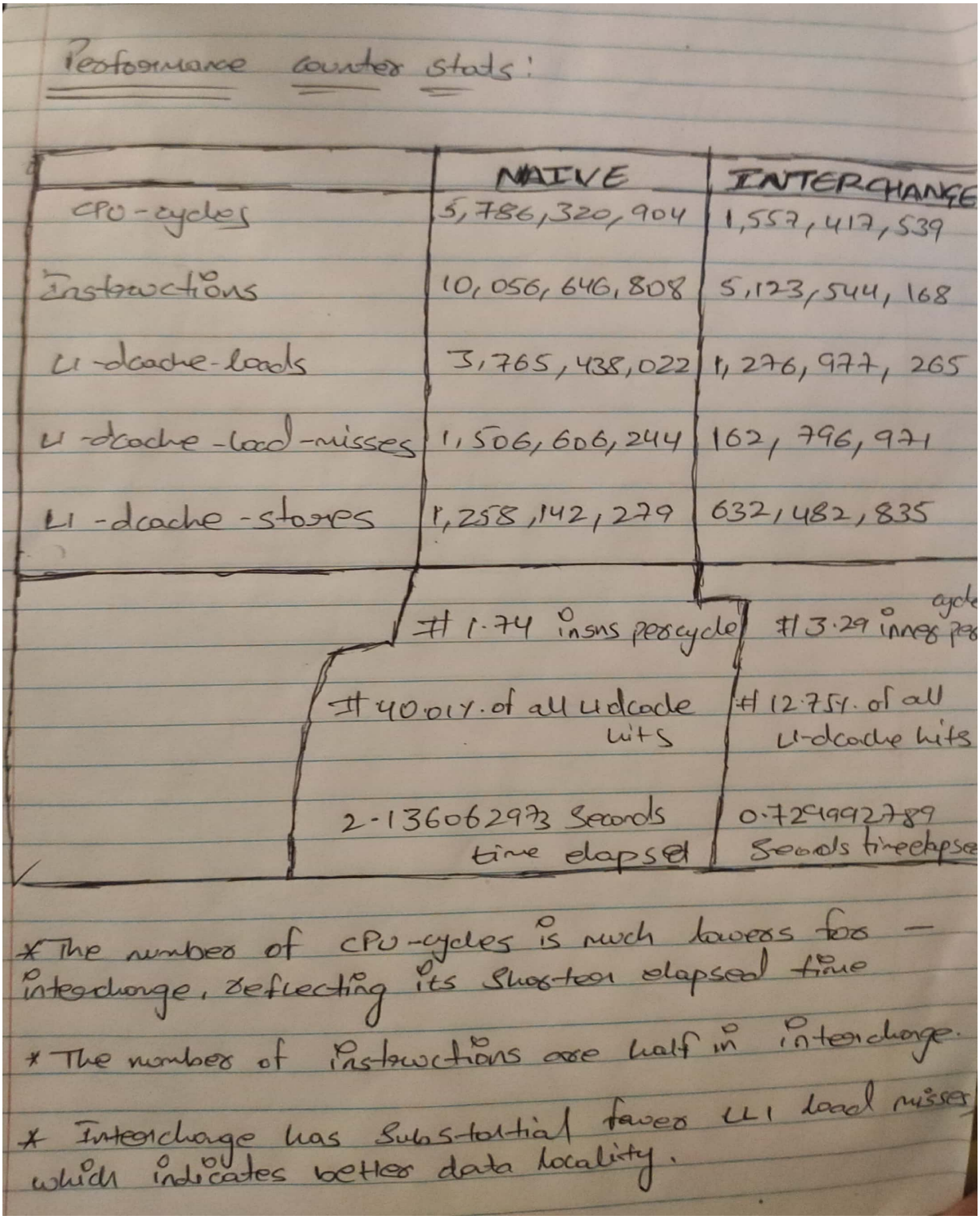
```
gbairi@fusion1:~$ perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-stores ./mm_interchange.out 500
using matrix size:500

 Performance counter stats for './mm_interchange.out 500':

     1,557,417,539      cpu-cycles
     5,123,544,168      instructions              #    3.29  insns per cycle
     1,276,977,265      L1-dcache-loads
       162,796,971      L1-dcache-load-misses     #   12.75% of all L1-dcache hits
       632,482,835      L1-dcache-stores

       0.729992789 seconds time elapsed
```

6. **Compare the numbers for both cases. The number of CPU cycles should much lower for interchange version, please list the CPU counts for each version, analyze the reason based on the code and CPU counters you get.**



Performance counter stats:

	NAIVE	INTERCHANGE
CPU-cycles	5,786,320,904	1,557,417,539
Instructions	10,056,646,808	5,123,544,168
L1-dcache-loads	3,765,438,022	1,276,977,265
L1-dcache-load-misses	1,506,606,244	162,796,971
L1-dcache-stores	1,258,142,279	632,482,835

1.74 insns per cycle

3.29 insns per cycle

40.01% of all L1dcache hits

12.75% of all L1-dcache hits

2.13606293 Seconds time elapsed

0.72499289 Seconds time elapsed

* The number of CPU-cycles is much lower for interchange, reflecting its shorter elapsed time
* The number of instructions are half in interchange.
* Interchange has substantial lower L1 load misses which indicates better data locality.

Exercise 3: What will happen if the matrix dimension is changed to 1000? Please analyze the performance difference between Naive and Interchange.

```
gbairi@fusion1:~$ perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-stores ./mm_naive.out 1000
using matrix size:1000


 Performance counter stats for './mm_naive.out 1000':

    48,441,452,921      cpu-cycles
    80,248,836,686      instructions              #    1.66  insns per cycle
    30,070,067,560      L1-dcache-loads
    12,893,679,014      L1-dcache-load-misses     #   42.88% of all L1-dcache hits
    10,038,998,560      L1-dcache-stores

      16.364624393 seconds time elapsed
```

```
gbairi@fusion1:~$ perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-stores ./mm_interchange.out 1000
using matrix size:1000

 Performance counter stats for './mm_interchange.out 1000':

    12,264,522,023      cpu-cycles
    40,494,421,497      instructions              #    3.30  insns per cycle
    10,109,698,455      L1-dcache-loads
     1,280,677,592      L1-dcache-load-misses     #   12.67% of all L1-dcache hits
     5,031,815,759      L1-dcache-stores

       4.287282230 seconds time elapsed
```

From the following figure you can keenly observe that

- CPU cycles for the matrix size 1000 is much greater than for the matrix size 500.
- Elapsed time is decreased in matrix size of 1000 when compared to matrix size of 500 in both Naive and Interchange.
- Also L1 load misses increases to greater level when matrix size increased to 1000 from 500.
- L1 loads are increased 10 times when matrix size is increased.

500 & 1000

(*) Performance counter stats for Matrix size

	MATRIX SIZE 500		Matrix size 1000	
NAIVE				
CPU-cycles	5,786,320,904	# 1.174 insns per cycle # 40.01% of all L-dcache hits	46,441,452,921	# 1.66 insns per cycle # 42.88% of
Instructions	10,056,646,808		80,248,836,686	
L1-dcache-loads	3,765,438,022		30,070,067,560	
L1-dcache-load-misses	1,506,606,244		12,893,679,014	
L1-dcache Stores	1,258,142,279		10,038,998,560	
	2.13606 Seconds time elapsed		0.72999 Seconds time elapsed	
INTERCHANGE				
CPU-cycles	1,557,417,539	#3.99 insns per cycle # 12.35% of all L-dcache hits	12,264,522,023	# 3.30 insns per cycle # 12.67% of all L1-dcache hits
Instructions	5,123,544,168		40,494,421,497	
L1-dcache-loads	1,276,977,265		10,109,698,455	
L1-dcache-load-misses	162,796,971		1,280,677,592	
L1-dcache Stores	632,482,835		5,031,815,759	
	16.364624 Seconds time elapsed		4.28728 Seconds time elapsed	

Exercise 4: Please find the hotspot using perf.

```
gbairi@fusion1:~$ perf record -g ./mm_interchange.out 500
WARNING: Kernel address maps (/proc/{kallsyms,modules}) are restricted,
check /proc/sys/kernel/kptr_restrict.

Samples in kernel functions may not be resolved if a suitable vmlinux
file is not found in the buildid cache or in the vmlinux path.

Samples in kernel modules won't be resolved at all.

If some relocation was applied (e.g. kexec) symbols may be misresolved
even with a suitable vmlinux or kallsyms file.

Cannot read kernel map
Couldn't record kernel reference relocation symbol
Symbol resolution may be skewed if relocation was used (e.g. kexec).
Check /proc/kallsyms permission or run as root.
using matrix size:500
[ perf record: Woken up 2 times to write data ]
[ perf record: Captured and wrote 0.259 MB perf.data (3059 samples) ]
gbairi@fusion1:~$
```

Then, to see the list of the costliest functions, you just have to use perf report:
Displayed a list of the costliest functions ordered by cost.

```
Samples: 2K of event 'cycles:pp', Event count (approx.): 1604549661
  Children      Self  Command          Shared Object       Symbol
+   99.30%     0.00%  mm_interchange.  mm_interchange.out  [.] main
+   99.30%     0.00%  mm_interchange.  libc-2.23.so        [.] __libc_start_main
+   99.30%     0.00%  mm_interchange.  [unknown]           [.] 0x09f6258d4c544155
+   98.35%    98.33%  mm_interchange.  mm_interchange.out  [.] compute_interchange
+    0.38%     0.00%  mm_interchange.  libc-2.23.so        [.] 0xffff80bb246e0786
+    0.28%     0.28%  mm_interchange.  libc-2.23.so        [.] random
+    0.23%     0.23%  mm_interchange.  mm_interchange.out  [.] init_matrix_2D
+    0.19%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff8106edd4
+    0.19%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff8106f062
+    0.19%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff81867b78
+    0.19%     0.19%  mm_interchange.  libc-2.23.so        [.] random_r
+    0.15%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff81864f9b
+    0.06%     0.00%  mm_interchange.  libc-2.23.so        [.] 0xffff80bb247d0ab3
+    0.06%     0.06%  mm_interchange.  libc-2.23.so        [.] 0x0000000000172ab3
+    0.06%     0.00%  mm_interchange.  libc-2.23.so        [.] brk
+    0.06%     0.00%  mm_interchange.  ld-2.23.so         [.] 0xffff80bb23f2b8ad
+    0.06%     0.06%  mm_interchange.  libc-2.23.so        [.] rand
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff81210fa9
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811cdf02
+    0.05%     0.05%  mm_interchange.  [unknown]           [k] 0x00007f44dba24786
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811a42b9
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811f14fa
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811cdee2
+    0.05%     0.05%  mm_interchange.  [unknown]           [k] 0xffffffff81419807
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811cdff1
+    0.05%     0.05%  mm_interchange.  [unknown]           [.] 0xffffffff81865ca7
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811cbb2d
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811cc611
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811d289c
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811d4cbd
+    0.05%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff811d537a
+    0.05%     0.05%  mm_interchange.  [unknown]           [k] 0xffffffff811cb2c3
+    0.05%     0.00%  mm_interchange.  [unknown]           [k] 0xffffffff811caf91
+    0.05%     0.00%  mm_interchange.  [unknown]           [k] 0xffffffff811d62f1
+    0.05%     0.00%  mm_interchange.  [unknown]           [k] 0xffffffff810831e7
+    0.05%     0.00%  mm_interchange.  [unknown]           [k] 0xffffffff8108953d
+    0.05%     0.00%  mm_interchange.  [unknown]           [k] 0xffffffff81089ed7
+    0.05%     0.00%  mm_interchange.  [unknown]           [k] 0xffffffff81089f54
+    0.05%     0.05%  mm_interchange.  [unknown]           [k] 0xffffffff811da432
+    0.05%     0.00%  mm_interchange.  ld-2.23.so         [.] 0xffff80bb23f160db
+    0.04%     0.04%  mm_interchange.  [unknown]           [k] 0xffffffff81865c3d
+    0.04%     0.04%  mm_interchange.  [unknown]           [k] 0xffffffff8120c9e9
+    0.04%     0.00%  mm_interchange.  [unknown]           [.] 0xffffffff81865bf7
+    0.04%     0.04%  mm_interchange.  [unknown]           [k] 0xffffffff81003c23
+    0.04%     0.04%  mm_interchange.  [unknown]           [k] 0xffffffff81867ddc
+    0.04%     0.04%  mm_interchange.  [unknown]           [k] 0x00007f44dba233d2
+    0.04%     0.00%  mm_interchange.  libc-2.23.so        [.] 0xffff80bb246df3d2
+    0.04%     0.04%  mm_interchange.  [unknown]           [k] 0xffffffff811ab1ee
+    0.03%     0.00%  mm_interchange.  libc-2.23.so        [.] 0xffff80bb247d0ab7
+    0.03%     0.03%  mm_interchange.  libc-2.23.so        [.] 0x0000000000172ab7
+    0.03%     0.00%  mm_interchange.  libc-2.23.so        [.] 0xffff80bb247d0ac6
+    0.03%     0.03%  mm_interchange.  libc-2.23.so        [.] 0x0000000000172ac6
+    0.03%     0.00%  mm_interchange.  ld-2.23.so         [.] 0xffff80bb23f1c5ed
For a higher level overview, try: perf report --sort comm,dso
```