CSC4005/MDS6108: Parallel Programming
Tutorial 4: Project 2

Sergei Kudria

October 11th, 2024

Grading

- Code (all parts together, fail/pass) - 60%
- Code efficiency - 20%
- Profiling (using perf) - 10%
- Report - 10%

For UG students the coding part consists of 4 tasks.

Caching Effect

Suppose, one can cache $2n$ elements of matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$.
How many memory queries do we need if

- ... we compute elements of $\mathbf{C}$ in a random order?
- ... we compute elements of $\mathbf{C}$ rowwise-columnwise?
- ... we partition $\mathbf{A}, \mathbf{B}$ into 4 blocks, then multiply blockwise?

Now, one can adjust these judgments to a general case.

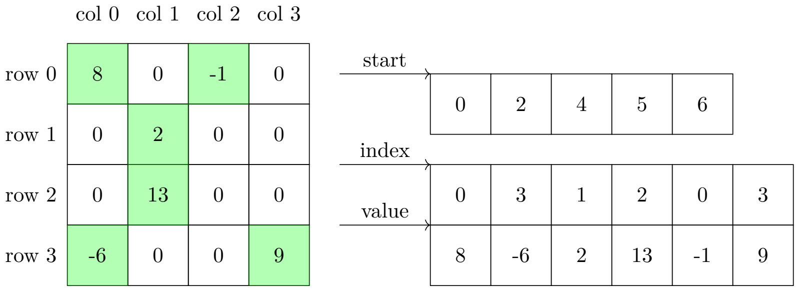
Tiled Matrix Multiplication

Advantages of tiled matrix multiplication:

- less number of memory quiries (task 1)
- parallelization (tasks 2-4)

Partition matrices $\mathbf{A}, \mathbf{B}$ into blocks of the same dimension.
Multiplication of blocks is performed by using nested loops (order?)
Be careful with partition, it must be a disjoint one!

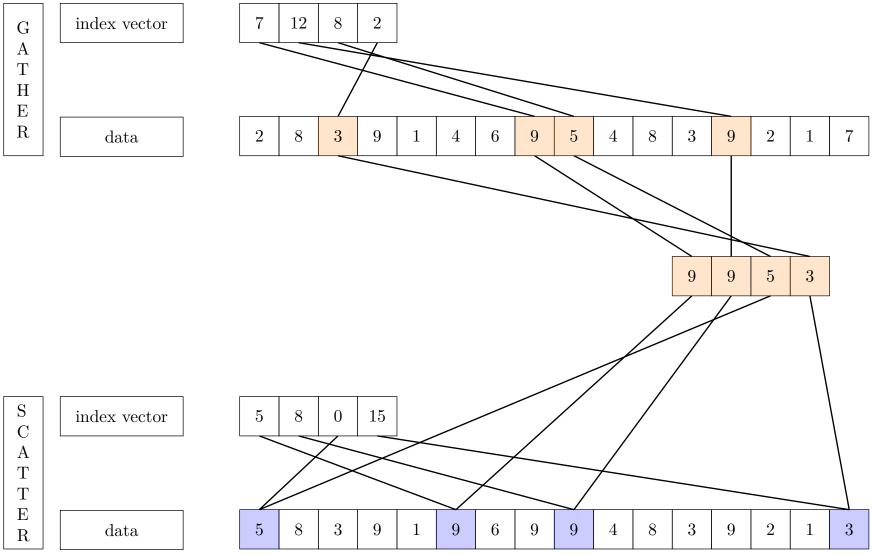
Sparse Matrix



	col 0	col 1	col 2	col 3
row 0	8	0	-1	0
row 1	0	2	0	0
row 2	0	13	0	0
row 3	-6	0	0	9

start	0	2	4	5	6	
index	0	3	1	2	0	3
value	8	-6	2	13	-1	9

Sparse Matrix Multiplication



Perf

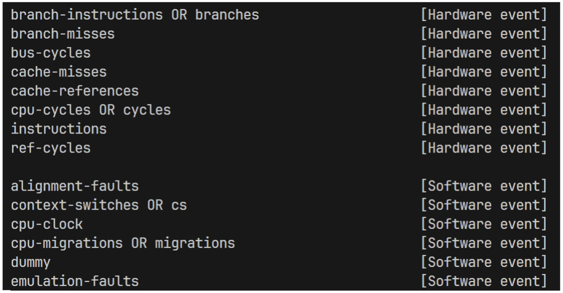
Manual: `perf.wiki.kernel.org`. We need the following commands:

- perf record - record events for later reporting
- perf report - break down events by process, function, etc.
- perf stat - obtain event counts

To look up: perf list, displays the symbolic event types which can be selected in the various perf commands with the -e option.

Perf

perf list:

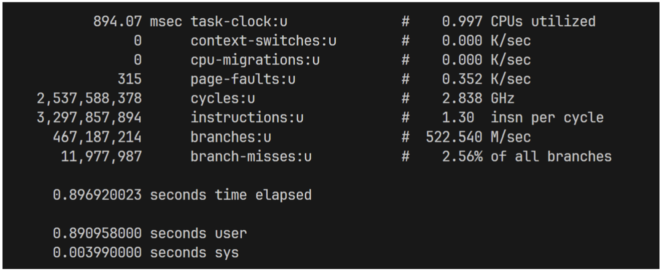


```
branch-instructions OR branches                    [Hardware event]
branch-misses                                      [Hardware event]
bus-cycles                                         [Hardware event]
cache-misses                                       [Hardware event]
cache-references        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

alignment-faults                                   [Software event]
context-switches OR cs                             [Software event]
cpu-clock                                          [Software event]
cpu-migrations OR migrations                       [Software event]
dummy                                              [Software event]
emulation-faults                                   [Software event]
```

Find the desired events.

Perf

perf stat <your_exe>:

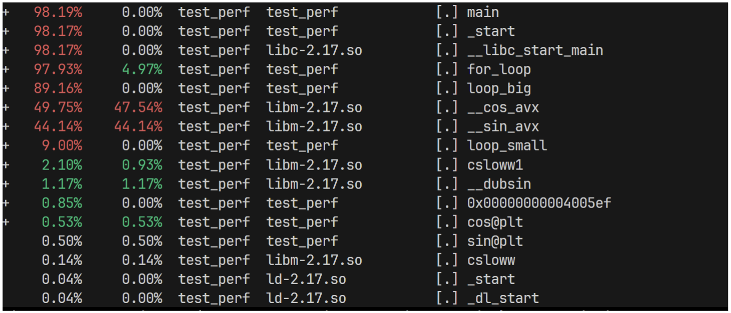


```
         894.07 msec task-clock:u              #    0.997 CPUs utilized
              0      context-switches:u        #    0.000 K/sec
              0      cpu-migrations:u          #    0.000 K/sec
            315      page-faults:u             #    0.352 K/sec
  2,537,588,378      cycles:u                  #    2.838 GHz
  3,297,857,894      instructions:u            #    1.30  insn per cycle
    467,187,214      branches:u                #  522.540 M/sec
     11,977,987      branch-misses:u           #    2.56% of all branches

    0.896920023 seconds time elapsed

    0.890958000 seconds user
    0.003990000 seconds sys
```

For something specific use perf stat –e <your_events> <your_exe>

Perf

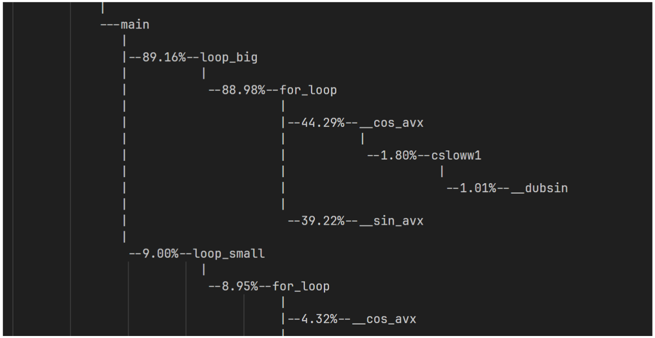
perf record <your_exe>, after that - perf report:



```
+   98.19%     0.00%  test_perf  test_perf          [.] main
+   98.17%     0.00%  test_perf  test_perf          [.] _start
+   98.17%     0.00%  test_perf  libc-2.17.so       [.] __libc_start_main
+   97.93%     4.97%  test_perf  test_perf          [.] for_loop
+   89.16%     0.00%  test_perf  test_perf          [.] loop_big
+   49.75%    47.54%  test_perf  libm-2.17.so       [.] __cos_avx
+   44.14%    44.14%  test_perf  libm-2.17.so       [.] __sin_avx
+    9.00%     0.00%  test_perf  test_perf          [.] loop_small
+    2.10%     0.93%  test_perf  libm-2.17.so       [.] csloww1
+    1.17%     1.17%  test_perf  libm-2.17.so       [.] __dubsin
+    0.85%     0.00%  test_perf  test_perf          [.] 0x00000000004005ef
+    0.53%     0.53%  test_perf  test_perf          [.] cos@plt
     0.50%     0.50%  test_perf  test_perf          [.] sin@plt
     0.14%     0.14%  test_perf  libm-2.17.so       [.] csloww
     0.04%     0.00%  test_perf  ld-2.17.so         [.] _start
     0.04%     0.00%  test_perf  ld-2.17.so         [.] _dl_start
```

Perf

To save in file: perf record –call-graph dwarf <your_exe>, then perf report -i perf.data > your_file.txt



```
       |
       ---main
          |
          |--89.16%--loop_big
          |          |
          |           --88.98%--for_loop
          |                     |
          |                     |--44.29%--__cos_avx
          |                     |          |
          |                     |           --1.80%--csloww1
          |                     |                     |
          |                     |                      --1.01%--__dubsin
          |                     |
          |                      --39.22%--__sin_avx
          |
           --9.00%--loop_small
                     |
                      --8.95%--for_loop
                                |
                                |--4.32%--__cos_avx
                                |
```