

Tools/perf/example_dgemm

Example perf: dgemm

- Prepare environment

```
module purge
module add \
    compiler/intel/2022 \
    numlib/mkl/2022
```

- Build dgemm benchmark

```
icc -O2 -xHost -ipo -qopenmp \
    -DUSE_MKL \
    timing.c stats.c matrix_common.c dgemm.multithread.c -o dgemm \
    -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
```

- Set up perftool, OpenMP and MKL environment

```
module add devel/perf

# Run with 76 threads
export OMP_NUM_THREADS=76
export KMP_AFFINITY="granularity=core,respect,scatter"
export MKL_NUM_THREADS=76
export MKL_DYNAMIC=false
```

- List available performance counters

```
perf list

...
fp_arith_inst_retired.256b_packed_double
[Counts number of SSE/AVX computational 256-bit packed double precision
floating-point instructions retired; some instructions will count twice as
noted below. Each count represents 4 computation operations, one for each
element. Applies to SSE* and AVX* packed double precision floating-point
instructions: ADD SUB HADD HSUB SUBADD MUL DIV MIN MAX SQRT FM(N)ADD/SUB.
FM(N)ADD/SUB instructions count twice as they perform 2 calculations per
element]
```

```
...
fp_arith_inst_retired.512b_packed_double
    [Counts number of SSE/AVX computational 512-bit packed double precision
    floating-point instructions retired; some instructions will count twice as
    noted below. Each count represents 8 computation operations, one for each
    element. Applies to SSE* and AVX* packed double precision floating-point
    instructions: ADD SUB MUL DIV MIN MAX SQRT RSQRT14 RCP14 FM(N)ADD/SUB.
    FM(N)ADD/SUB instructions count twice as they perform 2 calculations per
    element]
...
```

- Get performance statistics for benchmark dgemm

```
perf stat \
    --event=fp_arith_inst_retired.256b_packed_double \
    --event=fp_arith_inst_retired.512b_packed_double \
    --event=inst_retired.any \
    --event=cpu-clock \
    --event=cpu-cycles \
    --event=cpu-migrations \
    ./dgemm -m 30 -n 8000
```

```
Matrix size: 8000
Repeat multiply 30 times.
Alpha = 1.000000
Beta = 1.000000
Allocating Matrices...
Allocation complete, populating with values...
Performing multiplication...
Calculating matrix check...
```

```
=====
|| E ||_∞: 0.000000E+00
-> Solution check PASSED successfully.
Memory for Matrices: 1464.843750 MB
Multiply time: 6.276484 seconds
FLOPs computed: 30723840000000.000000
Min GFLOP/s: 4333.699080 GF/s
Max GFLOP/s: 5010.903317 GF/s
Average GFLOP/s: 4899.584788 GF/s
Std. dev. GFLOP/s: 782.431868 GF/s
Median GFLOP/s: 4958.747911 GF/s
MAD GFLOP/s: 17.685723 GF/s
=====
```

Performance counter stats for 'dgemm -m 30 -n 8000':

```
32,063,458      fp_arith_inst_retired.256b_packed_double:u # 67.156 K/sec
```

```

3,845,040,000,000      fp_arith_inst_retired.512b_packed_double:u #      8.053 G/sec
3,075,749,991,671      inst_retired.any:u                        #      6.442 G/sec
      477,444.54 msec  cpu-clock:u                                #      75.374 CPUs utilized
1,182,007,006,273      cpu-cycles:u                              #      2.476 GHz
      0                cpu-migrations:u                          #      0.000 /sec

```

6.334354962 seconds time elapsed

470.671232000 seconds user

4.735849000 seconds sys

- fp_arith_inst_retired.512b_packed_double / cpu-cycles ~ 3,25

- inst_retired.any / cpu-cycles ~ 2,60

- Comparison

* Num FLOP: (size³ + (size-1)³) * repetitions = 3,071424072e+13

* Num FLOP measured: fp_arith_inst_retired.512b_packed_double

* 8 (double per Register) = 3,076032e+13

- Record performance data of benchmark `dgemm` for use with `perf report` and `perf annotate`

```

perf record \
./dgemm -m 30 -n 8000

```

Matrix size: 8000

Repeat multiply 30 times.

Alpha = 1.000000

Beta = 1.000000

Allocating Matrices...

Allocation complete, populating with values...

Performing multiplication...

Calculating matrix check...

```

=====
|| E ||_∞:          0.000000E+00
-> Solution check PASSED successfully.

```

Memory for Matrices: 1464.843750 MB

Multiply time: 6.406716 seconds

FLOPs computed: 3072384000000.000000

Min GFLOP/s: 4231.797562 GF/s

Max GFLOP/s: 4919.832463 GF/s

Average GFLOP/s: 4800.559548 GF/s

Std. dev. GFLOP/s: 817.892980 GF/s

Median GFLOP/s: 4860.577063 GF/s

MAD GFLOP/s: 42.335237 GF/s

```

=====
[ perf record: Woken up 8 times to write data ]

```

```
[ perf record: Captured and wrote 73.874 MB perf.data (1936040 samples) ]
```

- Reduce recording overhead

```
perf record --freq=100 \  
./dgemm -m 30 -n 8000
```

```
Matrix size: 8000  
Repeat multiply 30 times.  
Alpha = 1.000000  
Beta = 1.000000  
Allocating Matrices...  
Allocation complete, populating with values...  
Performing multiplication...  
Calculating matrix check...
```

```
=====
```

E _∞:	0.000000E+00
-> Solution check PASSED successfully.	
Memory for Matrices:	1464.843750 MB
Multiply time:	6.323440 seconds
FLOPs computed:	3072384000000.000000
Min GFLOP/s:	4343.950293 GF/s
Max GFLOP/s:	4978.869913 GF/s
Average GFLOP/s:	4864.018914 GF/s
Std. dev. GFLOP/s:	847.409492 GF/s
Median GFLOP/s:	4922.905454 GF/s
MAD GFLOP/s:	23.178705 GF/s

```
=====
```

```
[ perf record: Woken up 1 times to write data ]
```

```
[ perf record: Captured and wrote 1.838 MB perf.data (47896 samples) ]
```

- Create performance report

```
perf report
```

87.76%	dgemm	libmkl_avx512.so.2	[.] mkl_blas_avx512_dgemm_kernel_0
5.21%	dgemm	libmkl_avx512.so.2	[.] mkl_blas_avx512_dgemm_dcopy_down24_ea
3.91%	dgemm	libiomp5.so	[.] __kmp_wait_4
2.30%	dgemm	libmkl_avx512.so.2	[.] mkl_blas_avx512_dgemm_dcopy_right8_ea
...			

– Interactive navigation in performance report:

- * h: get help
- * a: jump to annotated assembler code

- -> Most of the time is spent in only one MKL function
- -> `perf annotate` can only show MKL assembler code