P2

Sunday, July 12, 2020 10:40 AM

2.1 The Simple, yet Sophisticated CoreMark

1. Briefly describe the theory of operation of the benchmark algorithm. Make sure you add a short description of its three main algorithms:
   (a) Linked List
   (b) Matrix Multiply
   (c) State Machine

https://www.eembc.org/techlit/articles/coremark-whitepaper.pdf

CoreMark provides a performance indicator for embedded processors using basic data structures and algorithms that are common in real applications.

(a) Linked List
It consists of reversing, searching or sorting a list according to different parameters. Each list item can either contain a pre-computed value or a directive to call a specific algorithm to provide a value during sorting. The data space is partitioned into two blocks, one with the list itself and the other with the data items.

(b) Matrix Multiply
CoreMark performs multiplication on the input matrices with a constant, a vector, or another matrix. To validate that all results have been performed, CoreMark computes a CRC on the results from the matrix test.

(c) State Machine
CoreMark uses switch and if statements to exercise the CPU control structure. The state machine receives a string (stream of bytes) that is mapped to a number that can indicate a transition to 9 different states. The test is configured to ensure passing to all available states.

2. How does the CoreMark benchmark try to deal with compiler optimization to come up with a standardized result? Make sure you include the next concepts in your description:
   (a) Compile time vs run time
   (b) Volatile variables
   (c) Input-dependent results by using time based, scanf and command line parameters.

https://github.com/eembc/coremark

CoreMark follows certain strategies to ensure the desired data for the benchmark is calculated at runtime (by the processor to benchmark) and not by the compiler. For this they would rely on system functions (like scanf), command line parameters or volatile variables, whose values cannot be pre-determined by the compiler. They will be used/invoked before the portion of code to time/benchmark. The values these methods would provide are standardized since the intend is not randomizing, but that they can't be determined in compile time.

3. What is the difference between the "core_portme" and the "core" files? Are we allowed to modify all of them?

https://github.com/eembc/coremark

The "core_portme" are intended to be used for cross-compilation. This can be modified to indicate the specific platform to be used before compiling. The "core" files are not expected to be modified since these are generic for the benchmark.

2.1.1. Running CoreMark successfully on RPI4 for at least 20 seconds

```
root@raspberrypi4-64:~# ./coremark.exe 0x0 0x0 0x66 75000 7 1 2000
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 23018
Total time (secs): 23.018000
Iterations/Sec   : 3258.319576
Iterations       : 75000
Compiler version : GCC9.2.0
Compiler flags   : -O2 -DPERFORMANCE_RUN=1-lrt
Memory location  : Please put data memory location here
                        (e.g. code in flash, data on heap etc)
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0x382f
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 3258.319576 / GCC9.2.0 -O2 -DPERFORMANCE_RUN=1-lrt / Heap
```

2.1.2 Results

Changes applied in core_portme.h. Changing MULTHIREAD variable from 1 to 8. USE_PTHREAD changed to 1.
Compile with pthread flag:

$CC -O2 -Ilinux64 -I. -DFLAGS_STR=\\\\""-O2 -DPERFORMANCE_RUN=1-lrt"\\\\" -pthread -DITERATIONS=50000 -DPERFORMANCE_RUN=1 core_list_join.c core_main.c core_matrix.c core_state.c core_util.c linux64/core_portme.c -o ./coremark.exe

MULTHIREAD value	Total Time (sec)	Iterations/sec	Iterations
1	23.1	3246.7532	75000
2	23.106	6491.8203	150000
3	23.157	9716.2845	225000
4	23.201	12930.4771	300000
5	30.116	12451.8528	375000
6	38.385	11723.3294	450000
7	41.105	12772.1688	525000
8	46.190	12989.8246	600000

With fork:

MULTHIREAD value	Total Time (sec)	Iterations/sec	Iterations
4	23.127	12971.8511	300000
8	46.246	12974.0951	600000

2.1.3 Results

Keeping MULTHIREAD to 1. Disabling both: USE_PTHREAD and USE_FORK

Compiler Optimization	Total Time (sec)	Iterations/sec	Iterations
O0	105.770	709.0858	75000
O1	29.298	2559.9017	75000
Os	45.445	1650.3466	75000
O2	22.999	3261.0113	75000
O3	22.079	3396.8930	75000
Ofast	22.105	3392.8975	75000

Best Results

Compiler Optimization	MULTHIREAD value	Type	Iterations/sec
O3	4	PTHREADS	13485.5704
O3	8	PTHREADS	13539.1281
O3	4	FORK	13505.6048
O3	**8**	**FORK**	**13547.9938**

2.2 CoreMark-Pro

2.2.1 Understanding the Concept of the Improved Algorithm

1. How does the algorithm differ from the original one? What has improved?

From https://www.eembc.org/coremark-pro/

While CoreMark stresses the CPU pipeline, CoreMark-Pro tests the entire processor, adding comprehensive support for multicore technology, a combination of integer and floating-point workloads, and data sets for utilizing larger memory subsystems.

2. Overview its integer and floating-point workloads without explaining in detail the 24 FORTRAN kernels.

From: https://www.eembc.org/coremark-pro/

The integer workloads include JPEG compression, ZIP compression, an XML parser, the SHA-256 Secure Hash Algorithm, and a more memory-intensive version of the original CoreMark. The floating-point workloads include a fast Fourier transform (FFT), a linear algebra routine derived from LINPACK, a greatly improved version of the Livermore loops benchmark, and a neural net algorithm to evaluate patterns.

3. Is the simple CoreMark included into the CoreMark-Pro?

4. How are the multiple workloads combined to summarize results in one single score?

Source: https://github.com/eembc/coremark-pro

The CoreMark-PRO score is a weighted geometric mean of each workload

2.2.2 Running the CoreMark-Pro in the Raspberry Pi 4

For simplicity and to experience with a different approach, we decide to transfer and compile CoreMark-Pro directly on the RPI4. We compile and run with the command:
>> make TARGET=linux64 XCMD='-c4' certify-all

Results:

```
WORKLOAD RESULTS TABLE

                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            97.09      25.13       3.86
core                                           0.93       0.23       4.04
linear_alg-mid-100x100-sp                     93.28      24.25       3.85
loops-all-mid-10k-sp                           1.73       0.79       2.19
nnet_test                                      3.86       1.15       3.36
parser-125k                                    5.96       3.80       1.57
radix2-big-64k                                94.51      84.67       1.12
sha-test                                     156.25      47.17       3.31
zip-test                                      56.34      15.62       3.61

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                2150.79     779.78       2.76
```

-c1 results:

```
WORKLOAD RESULTS TABLE

                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            25.25      25.06       1.01
core                                           0.23       0.23       1.00
linear_alg-mid-100x100-sp                     24.34      24.15       1.01
loops-all-mid-10k-sp                           0.79       0.79       1.00
nnet_test                                      1.12       1.13       0.99
parser-125k                                    3.82       3.80       1.01
radix2-big-64k                                84.85      83.68       1.01
sha-test                                      47.17      47.17       1.00
zip-test                                      15.62      15.87       0.98

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                 778.86     778.02       1.00
```

-c2 results:

```
WORKLOAD RESULTS TABLE

                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            50.00      25.13       1.99
core                                           0.47       0.23       2.04
linear_alg-mid-100x100-sp                     48.59      24.13       2.01
loops-all-mid-10k-sp                           1.27       0.79       1.61
nnet_test                                      2.32       1.15       2.02
parser-125k                                    7.25       3.80       1.91
radix2-big-64k                                94.41      82.90       1.14
sha-test                                      94.34      47.17       2.00
zip-test                                      30.77      15.62       1.97

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                1420.88     777.52       1.83
```

-c3 results

```
WORKLOAD RESULTS TABLE

                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            73.53      25.19       2.92
core                                           0.70       0.23       3.04
linear_alg-mid-100x100-sp                     71.33      24.24       2.94
loops-all-mid-10k-sp                           1.61       0.79       2.04
nnet_test                                      2.90       1.12       2.59
parser-125k                                    7.43       3.79       1.96
radix2-big-64k                                96.76      84.19       1.15
sha-test                                     117.65      47.17       2.49
zip-test                                      44.12      15.62       2.82

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                1826.19     776.95       2.35
```

-c5 results:

```
WORKLOAD RESULTS TABLE

                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            96.15      25.19       3.82
core                                           0.88       0.23       3.83
linear_alg-mid-100x100-sp                     93.81      24.49       3.83
loops-all-mid-10k-sp                           1.81       0.79       2.29
nnet_test                                      4.15       1.16       3.58
parser-125k                                    6.41       3.79       1.69
radix2-big-64k                                92.03      83.31       1.10
sha-test                                     156.25      47.17       3.31
zip-test                                      47.17      15.62       3.02

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                2133.38     779.96       2.74
```

-c6 results

```
                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            97.09      25.25       3.85
core                                           0.84       0.23       3.65
linear_alg-mid-100x100-sp                     93.81      24.35       3.85
loops-all-mid-10k-sp                           1.78       0.79       2.25
nnet_test                                      4.26       1.14       3.74
parser-125k                                    6.23       3.80       1.64
radix2-big-64k                                91.18      84.77       1.08
sha-test                                     156.25      47.17       3.31
zip-test                                      51.28      15.62       3.28

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                2137.75     779.90       2.74
```

-c7 results:

```
                                          MultiCore SingleCore
Workload Name                              (iter/s)   (iter/s)    Scaling
------------------------------------------ ---------- ---------- ----------
cjpeg-rose7-preset                            96.15      25.13       3.83
core                                           0.91       0.23       3.96
linear_alg-mid-100x100-sp                     93.28      24.55       3.80
loops-all-mid-10k-sp                           1.78       0.79       2.25
nnet_test                                      4.21       1.14       3.69
parser-125k                                    6.04       3.80       1.59
radix2-big-64k                                90.47      83.92       1.08
sha-test                                     153.85      47.17       3.26
zip-test                                      51.09      15.87       3.22

MARK RESULTS TABLE

Mark Name                                  MultiCore SingleCore    Scaling
------------------------------------------ ---------- ---------- ----------
CoreMark-PRO                                2136.53     780.69       2.74
```

-c8 results:

Workload Name	MultiCore (iter/s)	SingleCore (iter/s)	Scaling
cjpeg-rose7-preset	95.24	25.13	3.79
core	0.93	0.23	4.04
linear_alg-mid-100x100-sp	93.11	24.35	3.82
loops-all-mid-10k-sp	1.67	0.79	2.11
nnet_test	3.88	1.14	3.40
parser-125k	5.80	3.79	1.53
radix2-big-64k	90.94	83.72	1.09
sha-test	172.41	47.17	3.66
zip-test	50.96	15.87	3.21

MARK RESULTS TABLE

Mark Name	MultiCore	SingleCore	Scaling
CoreMark-PRO	2122.57	779.55	2.72