CS540 Assignment 3

Dustin Ingram

December 5, 2011

1 Introduction

For this assignment, various methods of mini- matrix multiplication optimization are compared, including SSE2 vectorization and multithreading using OpenMP. The results are compared based on the performance.

2 Summary of Results

The results can be organized as follows:

2.1 SSE2 vs. OpenMP vs. Original

It was found that threadnig the vectorized matrix multiplication offered the highest performance levels.

3 Description of Computing Platform

All tests were run on `float.cs.drexel.edu`. Relevant system architecture information follows.

3.1 System & Kernel Information

```
$ uname -a
Linux float.cs.drexel.edu 2.6.35-28-generic #50-Ubuntu SMP Fri Mar 18 18:42:20
    UTC 2011 x86_64 GNU/Linux
```

3.2 GCC Version Information

```
$ gcc --version
gcc (Ubuntu/Linaro 4.4.4-14ubuntu5) 4.4.5
```

3.3 CPU Information

```
$ cat /proc/cpuinfo
processor       : 15
vendor_id       : GenuineIntel
cpu family      : 6
model           : 44
model name      : Intel(R) Xeon(R) CPU           L5630  @ 2.13GHz
stepping        : 2
cpu MHz         : 1600.000
cache size      : 12288 KB
physical id     : 1
siblings        : 8
core id         : 10
cpu cores       : 4
apicid          : 53
initial apicid  : 53
fpu             : yes
fpu_exception   : yes
cpuid level     : 11
wp              : yes
```

```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
    cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
    pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology
    nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2
    ssse3 cx16 xtpr pdcm dca sse4_1 sse4_2 popcnt aes lahf_lm ida arat dts
    tpr_shadow vnmi flexpriority ept vpid
bogomips        : 4266.84
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:
```

3.4 Memory Information

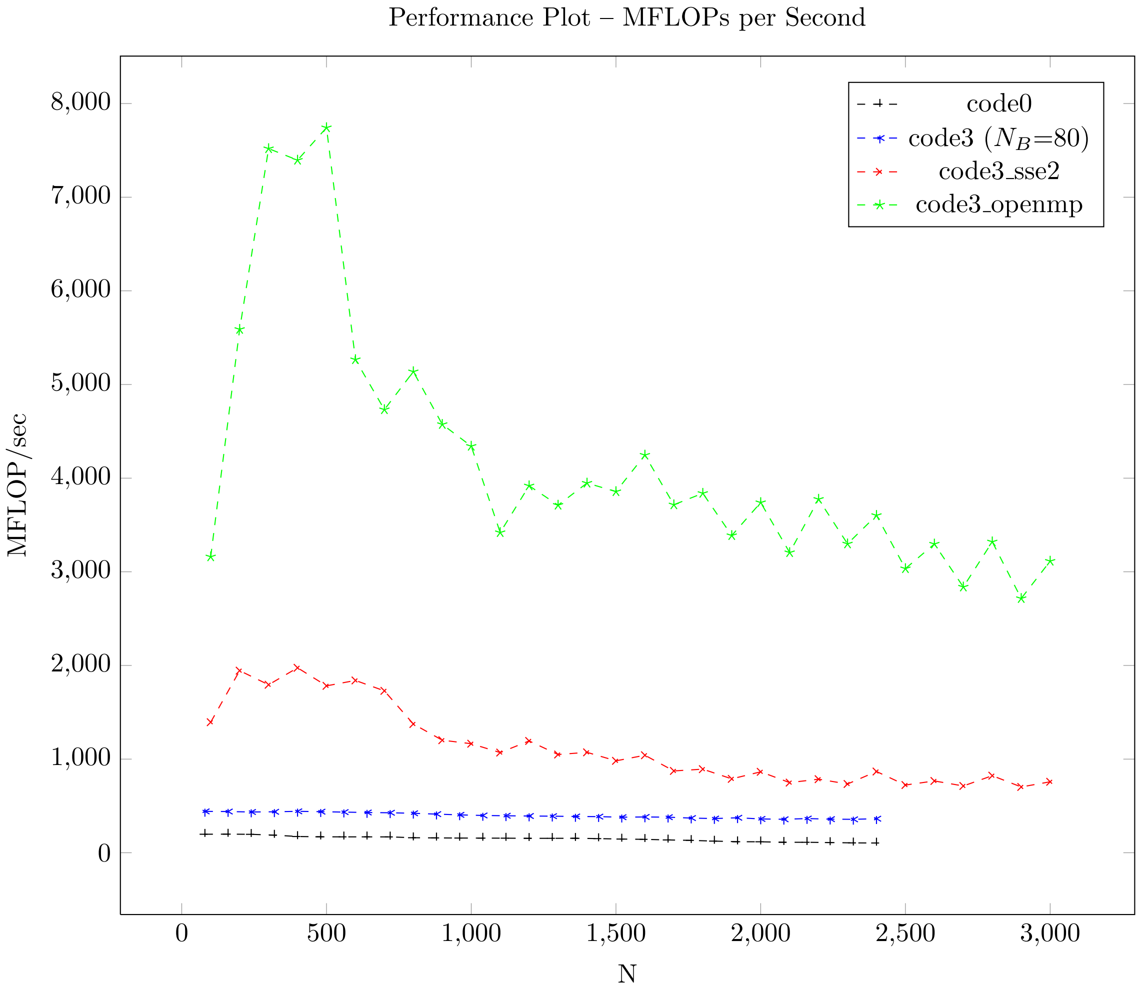
```
$ papi_mem_info
Memory Cache and TLB Hierarchy Information.
--------------------------------------------------------------------------
TLB Information.
  There may be multiple descriptors for each level of TLB
  if multiple page sizes are supported.

L1 Instruction TLB:
  Page Size:            2048 KB
  Number of Entries:       7
  Associativity:        Full

L1 Instruction TLB:
  Page Size:            4096 KB
  Number of Entries:       7
  Associativity:        Full

L1 Data TLB:
  Page Size:               4 KB
  Number of Entries:      64
  Associativity:           4

L1 Data TLB:
  Page Size:            2048 KB
  Number of Entries:      32
  Associativity:           4

L1 Data TLB:
  Page Size:            4096 KB
  Number of Entries:      32
  Associativity:           4

L1 Instruction TLB:
  Page Size:               4 KB
  Number of Entries:      64
  Associativity:           4
```

```
Cache Information.

L1 Data Cache:
  Total size:             32 KB
  Line size:              64 B
  Number of Lines:       512
  Associativity:           8

L1 Instruction Cache:
  Total size:             32 KB
  Line size:              64 B
  Number of Lines:       512
  Associativity:           4

L2 Unified Cache:
  Total size:            256 KB
  Line size:              64 B
  Number of Lines:      4096
  Associativity:           8

L3 Unified Cache:
  Total size:          12288 KB
  Line size:              64 B
  Number of Lines:    196608
  Associativity:          16

mem_info.c                                          PASSED
```

4 Experiments Performed & Results

4.1 SSE2 vs. OpenMP vs. Original

This is relatively self-explanatory. Matrix-multiplication was implemented using just SSE2, and then parallelized using OpenMP. The results are as follows, comparing them with the original triply-nested loop (I also included the best blocked-matrix multiply results for additional comparison).



This reveals, as expected, that vectorization with SSE2 greatly improves speed, and that matrix multiplication is embarrassingly parallelizable when used with a library such as OpenMP for multi-threading.

4.2 Source Code & Data

An explanation of the source used to generate these graphs as well as the resulting data is contained in the included tarball in the `README` file.

5 Conclusion

To conclude, a number of observations can be made:

- Vectorization and multithreading both greatly improve performance.