2. Aggregate Event Counts

(a) Are the instruction counts deterministic, or do they vary? How large is the variation?

The instruction counts vary each time, but they are similar. The variation is between 19161534132, 19161534091, it is only 41.

1	19,161,534,119	instructions:u
2	19,161,534,132	instructions:u
3	19,161,534,091	instructions:u
4	19,161,534,108	instructions:u
5	19,161,534,117	instructions:u
Average	19,161,534,113.4	

(b) Are the cycle counts deterministic, or do they vary? How large is the variation?

The cycle counts are not deterministic, they vary. The variation is between 13290466199-12745686736, it is very large.

1	12,745,686,736	cycles:u
2	13,290,466,199	cycles:u
3	12,754,137,562	cycles:u
4	12,769,289,966	cycles:u
5	12,753,437,091	cycles:u
average	12,862,603,510.8	

(c) Does changing the link order change the instructions or cycle metrics?

Changing the link order does not change the instructions or cycle metrics.

The instruction counts vary each time, but they are similar. The variation is between 19161534171, 19161534131, it is only 41.

1	19,161,534,153	instructions:u
2	19,161,534,171	instructions:u
3	19,161,534,152	instructions:u
4	19,161,534,131	instructions:u
5	19,161,534,151	instructions:u
Average	19,161,534,151.6	

The cycle counts are not deterministic, they vary. The variation is between 12756833195-12710649736, it is very large.

1	12,736,370,794	cycles:u
2	12,710,649,736	cycles:u
3	12,746,137,290	cycles:u
4	12,756,833,195	cycles:u

5	12,715,038,219	cycles:u
average	12,733,005,846.8	

3. Sample Results

(a) Did the three different methods of gathering function CPU use return the same results?

No the three methods do not return the same results.

Perf:

Perf sampled data:

real 0m3.424s

user 0m3.352s

sys 0m0.068s

Perf report

56.90% bzip2 bzip2 [.] mainSort

17.56% bzip2 bzip2 [.] BZ2_compressBlock

11.94% bzip2 bzip2 [.] mainGtU.part.0

11.44% bzip2 bzip2 [.] handle_compress.isra.2

0.96% bzip2 bzip2 [.] BZ2_blockSort

0.63% bzip2 bzip2 [.] add_pair_to_block

Perf annotate

3.14 | sub %r9d,%eax

SUB instruction caused the most CPU use.

Valgrind DBI tool:

Sample data:

==10247== Events : Ir

==10247== Collected : 19167425278

==10247==

==10247== I refs: 19,167,425,278

Time taking:

real 1m10.099s

user 1m10.084s

sys 0m0.044s

most used functions:

Ir file:function

11,291,448,187 /opt/ece571/401.bzip2/blocksort.c:mainSort [/opt/ece571/401.bzip2/bzip2]

3,381,835,437 /opt/ece571/401.bzip2/compress.c:BZ2_compressBlock [/opt/ece571/401.bzip2/bzip2]

2,138,813,059 /opt/ece571/401.bzip2/bzlib.c:handle_compress.isra.2 [/opt/ece571/401.bzip2/bzip2]

1,958,107,443 /opt/ece571/401.bzip2/blocksort.c:mainGtU.part.0 [/opt/ece571/401.bzip2/bzip2]

165,396,105 /opt/ece571/401.bzip2/blocksort.c:BZ2_blockSort [/opt/ece571/401 .bzip2/bzip2]

140,068,091 /opt/ece571/401.bzip2/bzlib.c:add_pair_to_block [/opt/ece571/401 .bzip2/bzip2]

Gprof:

Profiling data

real 0m3.518s

user 0m3.516s

sys 0m0.004s

most used funtions:

Index by function name

[88] _init

[24] saveInputFileMetaInfo.constprop.8 (bzip2.c)

[23] fopen_output_safely

[22] applySavedMetaInfoToOutputFile.constprop.7 (bzip2.c)

[21] BZ2_bzCompressEnd

(b) What were the relative speeds of the various methods of gathering the information?

The relative speeds of the three methods is Perf the fastest, Valgrind 50 times slower than Perf, Gprof is slower than Perf, but faster than Valgrind.

4. Skid

Perf record:

real 0m3.498s

user 0m3.432s

sys 0m0.064s

perf annotate:

3.57 | movzbl (%r15,%rax,1),%eax

(a) Which instruction was reported as taking the most time for the two cases?

For these two cases, the movzbl instruction is taking the most time.

(b) Which do you think is more likely?

I think the movslq is more likely taking most time.

(c) What is the cause of this difference?

It is that the instruction before the movslq is the jne (jump not equal) is processed. The processor cannot stop immediately, so the reported instruction might be off by a few instructions.