Co-Perf: Industry Sheet

For source code of this test: http://code.google.com/p/perftestharness/

Uses: SVN Repo Version # 47

History:

CPUs contain automatic hardware pre-fetching capabilities. According to the September version of the September 08 Intel 64 and IA-32 Architectures Optimization Reference Manual:

*Page: 64 Section 2.4.2 Data Prefetching*

*“This mechanism is similar to that of the Pentium M processor, but can track 16 forward data streams and 4 backward streams.”*

*Page: 134 Section 3.6.5 Data Layout Optimizations*

*“The prefetcher can recognize up to eight concurrent streams.”*

**Hypothesis:** Using varying numbers of data streams on a fixed amount of data, we should be able to show a performance cliff after 8 or 16 forward data streams.

Systems Spec:

While we have observed similar readings from multiple Intel processors, the test machine measured in this paper includes the following specifications:

Windows XP Pro SP3 Build 2600

Intel Mobile Core 2 Duo T7100 @ 1.80GHz

Cache L1 D 2 x 32 Kbytes

Cache L1 I 2 x 32 Kbytes

Level 2 2048 KBytes

Package: Socket P (478)

Technology: 65nm

Core VID 1.288 V

Memory 2048 Mbytes DDR2

Control

Our control case uses the following code:

void Test()

{

for(int k=0;k<10;k++ )

for(int j=0; j<DATA\_SIZE\_ROW; j++)

for(int i=0; i<1; i++)

{

gResult += gStaticData[i][j];

}

}

The following conditions exist:

* The test function will run 30 times.
* Time samples recorded before and after test use QueryPerformanceCounter timer
* Before the timer, a cache “scrambling” function sets cache to a reliable state.
* The program runs in release mode (MS Visual Studio)
* gStaticData contains 20 x DATA\_SIZE\_ROW
* DATA\_SIZE\_ROW = 10,000 integers

Figure The control test shows a fairly reliable min and average.

Test 1: IV: numStream, DV: m\_readNum

void test()

{

for(int k=0;k<10;k++ )

for(int j=0; j<m\_readNum; j++)

for(int i=0; i<numStream; i++)

{

gResult += gStaticData[i][j];

}

}

The following conditions exist:

* The test function will run 30 times.
* Time samples recorded before and after test use QueryPerformanceCounter timer
* Before the timer, a cache “scrambling” function sets cache to a reliable state.
* The program runs in release mode (MS Visual Studio)
* gStaticData contains 20 x DATA\_SIZE\_ROW
* DATA\_SIZE\_ROW = 10,000 integers
* numStream increments from [1..20].
* m\_readNum = DATA\_SIZE\_ROW/ numStreams

Figure Test 1 - linear reads of one stream is slower than linear reads of multiple streams.

Test 2: IV: numStream, DV: m\_readNum

In test 2, we re-run test 1 except we iterate from [20..1]. Doing so rules out compulsory cache misses as an explanation for slow linear reads of one stream.

Figure In an attempt to rule out cache biased results, we perform the test from [20..1] streams.

Summary

Unfortunately, we have with more questions than answers. We were unable to show data representative of a set number of hardware pre-fetch streams.