Turbo Boost Up, AVX Clock Down: Complications for Scaling Tests

Steve Lantz
What Is CPU Turbo? (Sandy Bridge)

- **P-state**: a voltage/frequency pair (ACPI terminology)
  - CPU and PG simultaneous heavy load at worst case conditions
  - Actual power has high dynamic range

- **P0** is max possible frequency

- **Pn** is the energy efficient state
  - OS control Pn-P1 range

- **P1-P0** has significant frequency range (GHz)
  - P1 to P0 range is fully H/W controlled
  - User preferences and policies
  - Single thread or lightly loaded applications

Factors Affecting Turbo Boost 2.0 (SNB)

“The processor’s rated frequency assumes that all execution cores are active and are at the sustained thermal design power (TDP). However, under typical operation not all cores are active or at executing a high power workload...Intel Turbo Boost Technology takes advantage of the available TDP headroom and active cores are able to increase their operating frequency.

“To determine the highest performance frequency amongst active cores, the processor takes the following into consideration to recalculate turbo frequency during runtime:

• The number of cores operating in the C0 state.
• The estimated core current consumption.
• The estimated package prior and present power consumption.
• The package temperature.

“Any of these factors can affect the maximum frequency for a given workload. If the power, current, or thermal limit is reached, the processor will automatically reduce the frequency to stay with its TDP limit.”

How Long Can SNB Sustain Turbo Boost?

After idle periods, the system accumulates “energy budget” and can accommodate high power/performance for up to a minute.

In Steady State conditions the power stabilizes on TDP, possibly at higher then nominal frequency.

Use accumulated energy budget to enhance user experience.
Does It Affect Tests on phiphi?

• Yes, in principle, for 1-4 threads
  – Faster clock, even in steady state?
  – More likely if a test is < 2 minutes
  – More likely if it has a cold(ish) start

• Why do we care?
  – It throws off parallel scaling tests
  – Baseline with 1 thread is too high

• We can try to measure the effect
  – Results in a few slides...

```bash
slantz@phiphi ~$ sudo cpupower frequency-info
analyzing CPU 0:
...
boost state support:
  Supported: yes
  Active: yes
  2400 MHz max turbo 4 active cores
  2400 MHz max turbo 3 active cores
  2500 MHz max turbo 2 active cores
  2500 MHz max turbo 1 active cores

slantz@phiphi ~$ grep MHz /proc/cpuinfo
cpu MHz : 2000.205
...

slantz@phiphi ~$ grep -c ida /proc/cpuinfo
24
```
Does It Affect Tests on mic0 or phi2?

- No, Turbo Boost is disabled
  - No IDA flags in /proc/cpuinfo
  - No Intel Dynamic Acceleration

- IDA test works best
  - Based on CPUID
  - Root isn’t needed

- Others do use IDA!
  - Stampede2 KNL
  - Stampede2 Skylake

```
slantz@phiphi ~$ ssh mic0 grep -c ida /proc/cpuinfo
0

slantz@phiphi ~$ micsmc --turbo status mic0
mic0 (turbo):
  Turbo mode is disabled

slantz@phi2 ~$ grep -c ida /proc/cpuinfo
0

slantz@phi2 ~$ sudo cpupower frequency-info
analyzing CPU 0:
...  
  boost state support:
    Supported: no
    Active: no

slantz@phi2 ~$ cat /sys/devices/system/cpu/intel_pstate/no_turbo
1
```
Strategy for Measuring Turbo Boost*

• Write a loop that takes a known, fixed number of cycles
  – Must always execute sequentially (no reordering by CPU)
  – Must not involve any memory operations
  – Can use SSE operations, but not AVX (we’ll look at that later)

• Run copies of it on some number of threads with OpenMP

• Time the loop on all threads; do this for many trials
  – Take minimum across threads and trials to remove OpenMP overhead

• Calculate max frequency observed (for nthreads ≤ ncores)

*https://stackoverflow.com/questions/11706563/how-can-i-programmatically-find-the-cpu-frequency-with-c/25400230#25400230
Inner Loop: SpinALot128

• Simple C code with intrinsics
  – Based on 128-bit SSE operations
  – Intrinsics prevent compiler trickery
  – Claim: 1 iteration is exactly 3 cycles

• Repeatedly adds 2 vectors of floats
  – Vector x just accumulates a vector of 1’s
  – Little or no cache accesses during computation
  – After spinning is over, a single int is returned

• Has a loop-carried dependency so CPU can’t reorder anything

```c
static int inline SpinALot128(int spinCount)
{
    __m128 x = _mm_setzero_ps();
    for(int i=0; i<spinCount; i++)
    {
        x = _mm_add_ps(x, _mm_set1_ps(1.0f));
    }
    return _mm_cvt_ss2si(x);
}
```

Code lifted from StackOverflow
void sample_frequency(const int nsamples, const int n, float *max, int nthreads) {
    *max = 0;
    volatile int x = 0;
    double min_time = DBL_MAX;

    #pragma omp parallel reduction(+:x) num_threads(nthreads)
    {
        double dtime, min_time_private = DBL_MAX;
        for(int i=0; i<nsamples; i++) {
            #pragma omp barrier
            dtime = omp_get_wtime();
            x += SpinALot128(n);
            dtime = omp_get_wtime() - dtime;
            if(dtime<min_time_private) min_time_private = dtime;
        }
        #pragma omp critical
        {
            if(min_time_private<min_time) min_time = min_time_private;
        }
    }

    *max = 3.0f*n/min_time*1E-9f;
}
What About Other Vector Sizes?

• For some time it has been known that Intel *slows down* the clock speed for the largest vector sizes (AVX, AVX-512)
  – P1 base frequency is only guaranteed for < AVX
  – Different base frequencies can apply to AVX and AVX-512
  – Max turbo frequencies can also be lower for these instructions

• Slowdowns started with Haswell (Xeon E3/5/7 “v3”) and AVX2
  – Sandy Bridge, Ivy Bridge (v1, v2) don’t have separate AVX frequencies

• Publication of AVX frequencies is inconsistent and scattered
  – By contrast, Intel makes non-AVX P1, P0 (max turbo) quite easy to find
Measuring AVX Slowdown Relative to SSE

• Need SpinALot256 and SpinALot512 for AVX vector widths
  – 256 bits for AVX (e.g., SNB)
  – 512 bits for AVX-512, MIC
• Can also supply a SpinALot32 for floats
  – Use “#pragma novector” to preclude vectorization

static int inline SpinALot256(int spinCount)
{
  __m256 x = _mm256_setzero_ps();
  for(int i=0; i<spinCount; i++) {
    x = _mm256_add_ps(x, _mm256_set1_ps(1.0f));
  }
  return (int)_mm256_cvtss_f32(x);
}

static int inline SpinALot512(int spinCount)
{
  __m512 x = _mm512_setzero_ps();
  for(int i=0; i<spinCount; i++) {
    x = _mm512_add_ps(x, _mm512_set1_ps(1.0f));
  }
  return (int)_mm512_cvtss_f32(x);

Intel OpenMP Settings

• Want threads pinned to separate physical cores
  – OMP_NUM_THREADS should not exceed physical core count
  – Careful! The OS will treat hyperthreading slots just like cores

• Set affinity via Intel (or OpenMP 4) environment variables*
  – Approach is similar to what was done in prior testing
  – KMP_AFFINITY=scatter,granularity=fine
  – Granularity clause pins each thread to *exactly* 1 hyperthreading slot
  – John McCalpin advises this if hardware counters are to be read

An Unused Intel OpenMP Variable

• KMP_HW_SUBSET also affects the OpenMP environment*
  – KMP_HW_SUBSET=1t is a different way to ensure 1 thread/core
  – Tricks OpenMP into thinking there truly is 1 hardware thread per core
  – Can also limit apparent cores/socket, e.g., KMP_HW_SUBSET=4c,1t
  – On phiphi, actual hardware corresponds to KMP_HW_SUBSET=6c,2t

• But KMP_HW_SUBSET may be too heavy-handed...
  – It changes the answer returned by omp_get_num_procs()

• This variable was formerly called KMP_PLACE_THREADS

*https://software.intel.com/en-us/node/694293
Results for SNB: No Evidence of Turbo!

Measured Freqs. (turbotest2) vs. Threads for Sandy Bridge E5-2620 (phi phi)

- Scalar freq., GHz, meas.
- SSE freq., GHz, meas.
- AVX freq., GHz, meas.
- (AVX-512 disallowed)
- Turbo freq., GHz
- Base freq., GHz
What Does Intel Say About Later Models?

• Intel provides full documentation for Haswell Xeons (v3)*
  – A Specification Update was first issued in Sept. 2014
  – Table 2 has “Turbo bins” giving turbo frequencies vs. core count
  – Table 3 gives AVX turbo and base frequencies vs. core count

• Nothing equivalent seems to exist for Broadwell (v4)

• Also difficult to find the facts for Xeon Phi KNC, KNL

• Intel has been more forthcoming with data on Skylake (v5)*
  – Let’s start there, then go back and re-examine older processors

Results for Skylake on Stampede2

Measured Frequencies (turbotest2) vs. Thread Count for Dual Skylake 8160

Results Are Insensitive to Length of Tests

- Table shows durations that were tried on Skylake 8160 (~2.5 GHz)
- Results in graph on previous slide are not significantly affected by duration, thus all results shown in slides are based on 1000 samples
- Note, test code is more likely to run at turbo than base, due to only 1 vector flop every 3 cycles

<table>
<thead>
<tr>
<th>Number of Samples</th>
<th>Number of Iterations</th>
<th>Approx. Test Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1,000,000</td>
<td>0.004 sec.</td>
</tr>
<tr>
<td>100</td>
<td>1,000,000</td>
<td>0.04 sec.</td>
</tr>
<tr>
<td>1000 (default)</td>
<td>1,000,000</td>
<td>0.4 sec.</td>
</tr>
<tr>
<td>10,000</td>
<td>1,000,000</td>
<td>4 sec.</td>
</tr>
<tr>
<td>40,000</td>
<td>1,000,000</td>
<td>16 sec.</td>
</tr>
</tbody>
</table>
Other Sources for Turbo, Base Frequencies

• Full Haswell and Skylake specifications are published by Intel
  – Note, Skylake has 3 sets of turbo/base frequencies: SSE, AVX, AVX-512

• Harder to find the equivalent for intermediate processors
  – Quote from John McCalpin in an Intel Forum post on 2/1/2017* -
    “Unfortunately, Intel only published the full set of frequency values for
    the Xeon E5 v3 [Haswell] family... the formula [for KNL] was published
    as a footnote in the Xeon Phi x200 product brief.”

• Next few slides summarize what I could find online
  – Includes data about Haswell from sources besides Intel

Anandtech’s Take on Haswell Frequencies

“To cope with the huge difference between the power consumption of Integer and AVX code [especially the new FMA operation in AVX2], Intel is introducing new [AVX base/turbo] frequencies...”

What Does This Do To Peak FLOP/s?

Intel: “To calculate theoretical peak FLOPS, Intel recommends using the marked TDP [Thermal Design Power] frequency as it best represents the frequency the processor could operate at over a range of workloads that utilize Intel AVX instructions. Customers may choose to use the AVX base frequency to calculate FLOPS if they think it better represents their system behavior.

“For example, Linpack runs closer to the AVX base frequency on the Intel Xeon processor E5 v3 family, so using the Intel AVX base frequency to calculate theoretical peak FLOPS will more accurately represent Linpack efficiency.”

“Haswell Conundrum: AVX or not AVX?”

• Presentation by Vincenzo Innocente at CERN, June 2014*

• Main takeaways from his benchmarks:
  – Quote: “Haswell is a great new Architecture: Not because of AVX”
  – Power management cannot be ignored in Haswell (Xeon E3/5/7 “v3”)
  – Unlike Sandy Bridge, individual cores can run at different clock speeds
  – AVX becomes a bit of a tradeoff, as wide vectors cause clock to drop
  – Memory bandwidth becomes even more constraining
  – Quote: “On modern architecture, extrapolation based on synthetic benchmarks is mission impossible”

*https://indico.cern.ch/event/327306/contributions/760669/attachments/635800/875267/HaswellConundrum.pdf
Microway’s Detailed HSW Specifications

“When examining the differences between AVX and Non-AVX instructions, notice that Non-AVX instructions typically result in no more than a 100MHz to 200MHz increase in the highest clock speed. However, AVX instructions may cause clock speeds to drop by 300MHz to 400MHz if they are particularly intensive.”

AVX base frequency is ~20% lower than rated. Dell adjusts their stated peak Gflop/s accordingly.

**Base Frequencies of Intel Broadwell processors**

<table>
<thead>
<tr>
<th>Processors</th>
<th>Rated base frequency (GHz)</th>
<th>AVX base frequency (GHz)</th>
<th>Theoretical Maximum Performance (GFLOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5-2699 v4, 22 core, 145W</td>
<td>2.2</td>
<td>1.8</td>
<td>634</td>
</tr>
<tr>
<td>E5-2698 v4, 20 core, 135W</td>
<td>2.2</td>
<td>1.8</td>
<td>576</td>
</tr>
<tr>
<td>E5-2697A v4, 16 core, 145W</td>
<td>2.6</td>
<td>2.2</td>
<td>563</td>
</tr>
<tr>
<td>E5-2690 v4, 14 core, 135W</td>
<td>2.6</td>
<td>2.1</td>
<td>470</td>
</tr>
<tr>
<td>E5-2650 v4, 12 core, 105W</td>
<td>2.2</td>
<td>1.8</td>
<td>346</td>
</tr>
</tbody>
</table>

“A CPU core will return to Non-AVX mode 1 millisecond after AVX instructions complete...

“Within a given CPU, some cores may be operating in AVX mode while others are operating in Non-AVX mode. In the previous generation, AVX instructions running on a single core would cause all cores to run in AVX mode.”

(This article also has a figure providing turbo and base frequencies as a function of core count for various Broadwell model numbers)
Xeon Phi KNL and KNC

• KNL specs were published by Intel as a footnote:
  – “Frequency listed is nominal (non-AVX) TDP frequency. For all-tile turbo frequency, add 100 MHz. For single-tile turbo frequency, add 200 MHz. For high-AVX instruction frequency, subtract 200 MHz.”*
  – Turbo is disabled on phi2 (though not on Stampede2’s KNL nodes)

• KNC 7120 models have 1.333 GHz turbo vs. 1.238 GHz base
  – Other models lack turbo, and KNC has no separate AVX turbo/base
  – Only the base matters to us, since turbo is disabled on mic0 (7120P)

• Let’s finish by trying some experiments on KNL and KNC...

Results for KNL on Stampede2

![Graph: Measured Frequencies (turbotest2) vs. Thread Count for Knights Landing 7250]
Results for KNL on phi2

Measured Freqs. (turbotest2) vs. Threads for KNL 7210 (phi2 - NO TURBO)
Results for KNC on phiphi-mic0
Platform Dependence of Cycles/Iteration

• Latency of the simple “SpinALot” loop turns out to vary quite a lot depending on processor family
  – Answer is rounded up if within 15% of the next integer, in case of turbo

<table>
<thead>
<tr>
<th>Xeon Family</th>
<th>Cycles per Iteration</th>
<th>Turbo Enabled?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandy Bridge</td>
<td>3</td>
<td>yes</td>
</tr>
<tr>
<td>Skylake</td>
<td>3</td>
<td>yes</td>
</tr>
<tr>
<td>KNL (Stampede2)</td>
<td>6</td>
<td>yes</td>
</tr>
<tr>
<td>KNL (phi2)</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>KNC</td>
<td>4</td>
<td>no</td>
</tr>
</tbody>
</table>

12/15/2017
Conclusions

• To an increasing extent, Intel processors adjust their frequency according to workload
  – Turbo Boost gives an extra burst of speed at low thread counts
  – Highly threaded, vectorized code may run in a lower frequency range

• This behavior can confuse scaling studies, and it may reduce the benefit of AVX and AVX-512 vectorization
  – Turbo Boost can be disabled; AVX or AVX-512 effects cannot be

• Our previous test results on older machines are not affected
  – Still need high-flop test to confirm absence of AVX slowdown on phi2