Author: Jatin Bhateja

Employee ID: 11846689

AVX 512 Frequency Throttling on CLX

===============================

AVX512 heavy instruction operates at higher licensing level (LVL2). These heavy operations mainly comprise of FMA, INT MUL, DIV instructions which consume more cycles. Thus, to operate under prescribed TDP limits X86 CPUs reduces the CORE frequency. Occasional execution of AVX512 heavy instruction on few cores does not impact the CORE frequency if there is sufficient POWER head room to absorb the power spikes. Its important to note here that licensing levels translations are function of vector size and not the instruction encoding, thus an AVX512 instruction operating over narrow 256-bit vectors never impact CORE frequency.

Please refer to following micro workload and its performance data over Cascadelake Server.

**#include "stdio.h"**

**#include "pthread.h"**

**#include "stdlib.h"**

**#define CORES 112**

**#define ITERATIONS 500000000**

**typedef struct {**

**int init;**

**int res;**

**} packet;**

**void\* micro(void\* val) {**

**packet \* data = (packet\*)(val);**

**int init = data->init;**

**int ret = 0;**

**for(int i = 0; i < ITERATIONS; i++) {**

**int res[16];**

**asm volatile(**

**"vpbroadcastd %1 , %%zmm1 \n\t"**

**"vpmulld %%zmm1, %%zmm1, %%zmm1 \n\t"**

**"vfmadd231ps %%zmm1, %%zmm1, %%zmm1 \n\t"**

**"vfmadd231ps %%zmm1, %%zmm1, %%zmm1 \n\t"**

**"vmovdqu64 %%zmm1 , %0 \n\t"**

**: "=m"(\*res)**

**: "r"(init)**

**: "%zmm1"**

**);**

**ret += res[i & 7];**

**}**

**data->res = ret;**

**return data;**

**}**

**int main() {**

**int res = 0;**

**pthread\_t workers[CORES];**

**char \* mem = (char\*)malloc(sizeof(packet)\*CORES);**

**for(int i = 0; i < CORES; i++) {**

**packet \* workitem = (packet\*)(mem + sizeof(packet)\*i);**

**workitem->init = i\*1000;**

**}**

**for(int i = 0; i < CORES; i++) {**

**pthread\_create(&workers[i], NULL, micro, (void\*)(mem + sizeof(packet)\*i));**

**}**

**for(int i = 0; i < CORES; i++) {**

**pthread\_join(workers[i], NULL);**

**}**

**// Accumulate the results;**

**for(int i = 0; i < CORES; i++) {**

**packet \* workitem = (packet\*)(mem + sizeof(packet)\*i);**

**res += workitem->res;**

**}**

**printf("[RES] %d\n", res);**

**return res;**

**}**

AVX2 workload was created by replacing ZMM vectors with YMM vectors in above thread kernel.







