**OLD**

/\* Calculate and insert four-bit IPC value. Shift puts in lo/hi part of a byte \*/

inline void do\_ipc\_calc(u64 \*claim, u64 delta\_cycles,

struct kutrace\_traceblock\* tb, bool shift) {

u64 inst\_ret;

u64 delta\_inst;

u64 ipc;

u8\* ipc\_byte\_addr;

if (!do\_ipc) {return;}

/\* There will be random large differences the first time; we don't care. \*/

#ifdef TEST\_LLC\_MISS

inst\_ret = ku\_get\_llc\_miss();

#else

inst\_ret = ku\_get\_inst\_retired();

#endif

delta\_inst = inst\_ret - tb->prior\_inst\_retired;

tb->prior\_inst\_retired = inst\_ret;

/\* NOTE: pointer arithmetic divides claim by 8, giving the byte offset we want \*/

ipc\_byte\_addr = (u8\*)(tracebase) + (claim - (u64\*)(tracebase));

#ifdef TEST\_LLC\_MISS

ipc = get\_granular\_llc(delta\_inst);

#else

ipc = get\_granular(delta\_inst, delta\_cycles);

#endif

if (shift)

ipc\_byte\_addr[0] |= ipc << 4;

else

ipc\_byte\_addr[0] = ipc;

}

**NEW**

/\* Calculate and insert 4-bit IPC/LLC value. Shift packs in hi part of a byte \*/

/\* If either IPC or LLC is on, store its quantized 4-bit value \*/

/\* If both are on keep just high two bits of each. HTML wrapper displays both \*/

inline void do\_ipcllc\_calc(u64 \*claim, u64 delta\_cycles,

struct kutrace\_traceblock\* tb, bool shift) {

u64 inst\_ret, delta\_inst, ipc;

u64 llc\_miss, delta\_miss, llc;

u64 four\_bits; /\* To store into trace: ipc, llc, or two bits of each \*/

u8\* ipcllc\_byte\_addr;

if (do\_neither) {return;}

/\* There will be random large differences the first time; we don't care. \*/

if (do\_ipc) {

inst\_ret = ku\_get\_inst\_retired();

delta\_inst = inst\_ret - tb->prior\_inst\_retired;

tb->prior\_inst\_retired = inst\_ret;

four\_bits = ipc = get\_granular\_ipc(delta\_inst, delta\_cycles);

}

if (do\_llc) {

llc\_miss = ku\_get\_llc\_miss();

delta\_miss = llc\_miss - tb-> prior\_llc\_misses;

tb->prior\_llc\_misses = llc\_miss;

four\_bits = llc = get\_granular\_llc(delta\_miss);

}

if (do\_both) {

/\* Pack upper two bits only of each counter \*/

four\_bits = (ipc & 0x0c) | (llc >> 2);

}

/\* NOTE: pointer arithmetic divides claim by 8, giving the byte offset \*/

ipcllc\_byte\_addr = (u8\*)(tracebase) + (claim - (u64\*)(tracebase));

/\* Shift saves the value for optimized returns in high 4 bits of byte \*/

if (shift)

ipcllc\_byte\_addr[0] |= four\_bits << 4;

else

ipcllc\_byte\_addr[0] = four\_bits;

}

// sysfoo

function name(d) {return d[9];}

// Create full event name

// short: sysfoo(arg)

// long, simple annot: sysfoo

// long, normal annot: sysfoo(arg)=ret

//

function fullname(d, short) {

// xx.xxus fullname; dur ipc= llc=

function complete\_text(d) {

// short: sysfoo

// simple: sysfoo

// long: xx.xxus fullname; dur ipc= llc=

function annotatespan(d, clss, short, my, rotate) {

// fullname

function annotatesearch2(needle, mindur, maxdur) {

search fullname

if ipc>0, also search ipc= or llc=

gives retval -128..-1 as eagain etc.

xx.xxus sysfoo(arg)=number; dur ipc= llc= annotatespan

complete\_text

sysfoo(arg)=eagain ipc= llc= annotatesearch2 (no times)

sysfoo(arg)=number fullname long

sysfoo(arg) fullname short

sysfoo fullname simple

name

rationalize these...