4/10/2024 @ European LLVM Developers' Meeting

Enabling HW-based PGO

for both Windows and Linux

Wei Xiao (wei3.xiao@intel.com)

Contributors: Timothy Creech, Haohai Wen, Rakesh Krishnaiyer
Mike Chynoweth, Ahmad Yasin, Tianqing Wang

intel®

Agenda

1. Motivation
2. New Feedback Capabilities
3. Windows Support
4. Demo
5. Challenges & Solutions
6. Upstreaming
7. Summary
8. Q&A

Motivation

- Sampled profiling periodically interrupts program execution to grab a HW event count or machine state. Most CPUs can do this purely in HW or can emulate it in SW (by using a timer).
- Modern CPUs support more advanced forms of HW profiling:

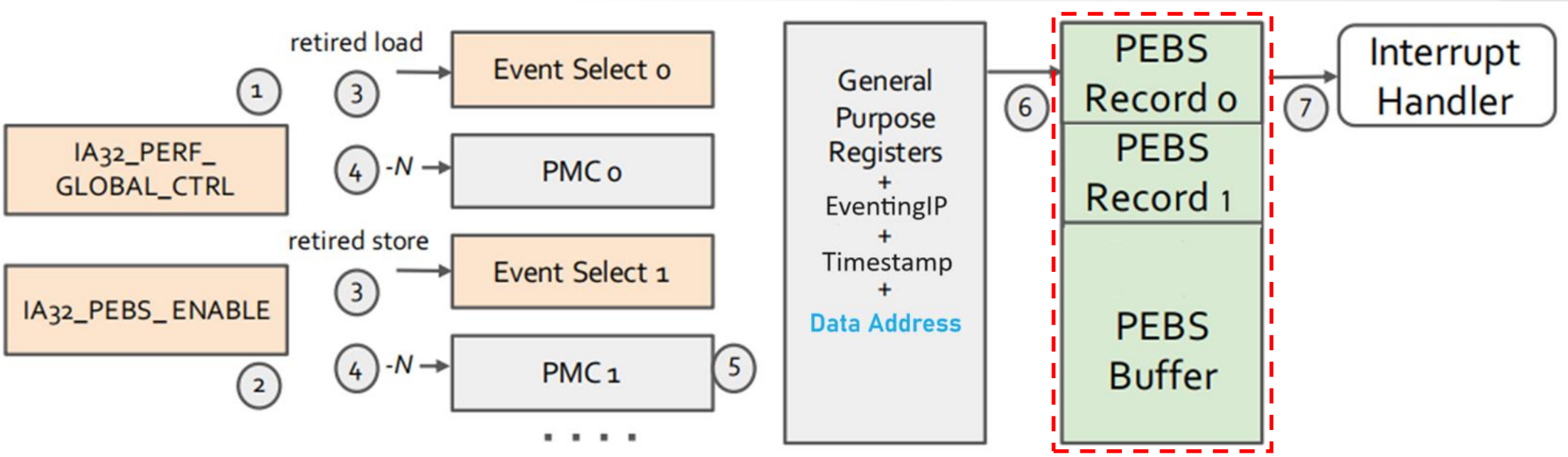
Intel x86_64	AMD x86_64	ARM	RISC-V	HW Profiling Capabilities
PEBS	IBS	SPE		Event-based Sampling
LBR	LbrExtV2	BRBE	CTR	Short trace of branches
PT		CoreSight		Full trace of executed instructions

These allow gathering samples in HW, possibly multiple at a time, with lower overhead and provide other benefits, such as reduced-skid, precise distribution and Data Address.

Intel PEBS Overview

Processor Event-Based Sampling (PEBS)

- Low-overhead sampling (an order of magnitude reduction)
- Reduced-skid or Precise-Distribution



Intel LBR Overview

Last Branch Record (LBR)

- CPU collects data for taken branches
  - Source Address
  - Target Address
  - INFO
- Low overhead
- Recent CPUs offer Architectural LBRs
  - Consistent across processor generations and in virtualized environments

Register Address (Hex)	Architectural MSR Name and bit fields	MSR/Bit Description
1500-151FH	**IA32_LBR_x_FROM_IP** [63:0]	**FROM_IP:** The source IP of the recorded branch or event, in canonical form.
1600-161FH	**IA32_LBR_x_TO_IP** [63:0]	**TO_IP:** The destination IP of the recorded branch or event, in canonical form.
1200-121FH	**IA32_LBR_x_INFO**	Last Branch Record entry X info register (R/W). An attempt to read or write IA32_LBR_x_INFO such that x >= IA32_LBR_DEPTH.DEPTH will #GP.
	15:0	**CYC_CNT:** The elapsed CPU cycles (saturating) since the last LBR was recorded.
	55:16	Undefined, may be zero or non-zero. Writes of non-zero values do not fault, but reads may return a different value.
	59:56	**BR_TYPE:** The branch type recorded by this LBR. Encodings: 0000B: JCC 0001B: JMP Indirect 0010B: JMP Direct 0011B: CALL Indirect 0100B: CALL Direct 0101B: RET 011xB: Reserved 1xxxB: Other Branch
	60	**CYC_CNT_VALID:** CYC_CNT value is valid.
	61	**TSX_ABORT:** This LBR record is a TSX abort. On processors that do not support Intel® TSX (CPUID.07H.EBX.HLE[bit 4]=0 and CPUID.07H.EBX.RTM[bit 11]=0), this bit is undefined.
	62	**IN_TSX:** This LBR record records a branch that retired during a TSX transaction. On processors that do not support Intel® TSX (CPUID.07H.EBX.HLE[bit 4]=0 and CPUID.07H.EBX.RTM[bit 11]=0), this bit is undefined.
	63	**MISPRED:** The recorded branch direction (Jcc) or target (indirect branch) was mispredicted.

HWPGO Overview

HW-based PGO is an extension of existing Sampling-based PGO

- HWPGO is a kind of Sampling-based PGO for efficient profiling on optimized binaries in production environments.
- HWPGO enables new types of feedback capabilities provided by HW for new compiler optimizations. HW counters can track a wide range of events, including:
  - Instructions retired
  - Branch mispredictions
  - Cache misses
  - Memory accesses and Data Address
  - Floating-point operations
  - Architectural LBR Inserts (in next-gen CPUs)
  - …

New Feedback Capabilities

Hardware can provide accurate frequency and profiles of other events:

Capture LBR

frequency with precise-distribution

Additional event for all Mispredictions

- $ perf record -b -e BR_INST_RETIRED.NEAR_TAKEN:uppp,BR_MISP_RETIRED.ALL_BRANCHES:upp

```
11  BR_INST_RETIRED.NEAR_TAKEN:uppp:              4016b0 0x4016b0/0x40116d/P/-/-/8  0x401168/0x4016b0/P/-/-/
   1193/0x401160/P/-/-/6  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/5
   b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/6  0x4016b0/0x40116d/P/-/-/1  0
   /0x4016b0/P/-/-/11  0x401193/0x401160/M/-/-/3  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1
12 BR_MISP_RETIRED.ALL_BRANCHES:upp:             401193 0x4016b0/0x40116d/P/-/-/7  0x401168/0x4016b0/P/-/-/
   01193/0x401160/M/-/-/3  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/5
   1b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/5  0x4011b5/0x401170/P/-/-/1
   8/0x4016b0/P/-/-/14  0x401193/0x401160/M/-/-/3  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1
```

New Feedback Capabilities

Branch Mispredict Feedback Example

```
1  for (int i = 0; i < N; i++) {
2    int *p;
3    if(s1[i] > 8000) {
4      p = &s2[i];
5      int z = i * i * i * i * i * i * i;
6      nop(p, z);
7    } else {
8      p =  &s3[i];
9      nop(p, 3);
10   }
11   dst[i] = *p;
12 }
```

Poorly predicted according to HW events:

- **BR_INST_RETIRED.NEAR_TAKEN**
- **BR_MISP_RETIRED.ALL_BRANCHES**

Attach ‘unpredictable’ Metadata to BR instruction

HWPGO

```
1  for (int i = 0; i < N; i++) {
2    int *p;
3
4    int c = (s1[i] > 8000);
5    p = c ? (&s2[i]) : (&s3[i]);
6    int z = i * i * i * i * i * i * i;
7    int arg2 = c ? z : 3;
8    nop(p, arg2);
9
10   dst[i] = *p;
11 }
```

New Feedback Capabilities

Before HWPGO:

```
        jmp     .LBB0_1
        .p2align        4, 0x90
.LBB0_3:
        movq    %rsi, %rcx
        movl    $3, %edx
        callq   nop
        movq    %rsi, %r13
.LBB0_4:
        movl    (%r13), %eax
        movl    %eax, (%r14,%r12,4)
        incq    %r12
        addq    $4, %rsi
        addq    $4, %r15
        cmpq    $20000, %r12
        je      .LBB0_5
.LBB0_1:
        cmpl    $8001, (%rbx,%r12,4)
        jl      .LBB0_3
# %bb.2:
        leaq    (%rdi,%r12,4), %r13
        movl    %r12d, %eax
        imull   %r12d, %eax
        imull   %r12d, %eax
        movl    %eax, %edx
        imull   %r12d, %edx
        imull   %eax, %edx
        movq    %r15, %rcx
        callq   nop
        jmp     .LBB0_4
```

Opt. mispredicted conditional branch to conditional move

HWPGO

After HWPGO:

```
.LBB0_1:
        movl    %r12d, %eax
        imull   %r12d, %eax
        imull   %r12d, %eax
        movl    %eax, %edx
        imull   %r12d, %edx
        imull   %eax, %edx
        cmpl    $8001, (%rbx,%r15)
        movq    %rsi, %r13
        cmovgeq %rdi, %r13
        cmovll  %ebp, %edx
        leaq    (%r15,%r13), %rcx
        callq   nop
        movl    (%r13,%r15), %eax
        movl    %eax, (%r14,%r15)
        incq    %r12
        addq    $4, %r15
        cmpq    $20000, %r12
        jne     .LBB0_1
```

New Feedback Capabilities

Branch Mispredict Feedback Example

Before HWPGO:

```
$ perf stat -e
cycles:u,instructions,br_inst_retired.all_branches:u,br_misp_retired.all_branches
-- ./unpredictable

 Performance counter stats for './unpredictable':

     3,243,043,047      cycles:u
     3,619,535,187      instructions:u       #   1.12  insn per cycle
       917,083,309      br_inst_retired.all_branches:u
        85,966,707      br_misp_retired.all_branches:u

       1.021617622 seconds time elapsed
```

After HWPGO:

```
$ perf stat -e
cycles:u,instructions,br_inst_retired.all_branches:u,br_misp_retired.all_branches
-- ./unpredictable.hwpgo

 Performance counter stats for './unpredictable.hwpgo':

     1,715,030,113      cycles:u
     4,000,954,710      instructions:u       #   2.33  insn per cycle
       600,132,829      br_inst_retired.all_branches:u
            13,322      br_misp_retired.all_branches:u

       0.541091594 seconds time elapsed
```

+10% retired instructions

2X IPC

-35% (retired branches)

1.8X improvement in overall performance

Visit: https... de

Call community collaboration on HWPGO to:

- Add infrastructure support for more feedback/profile types besides frequency (i.e., "-fprofile-sample-use=code.freq.prof")
- Add optimizations for new profile types

SPGO/HWPGO Compiler Support on Windows

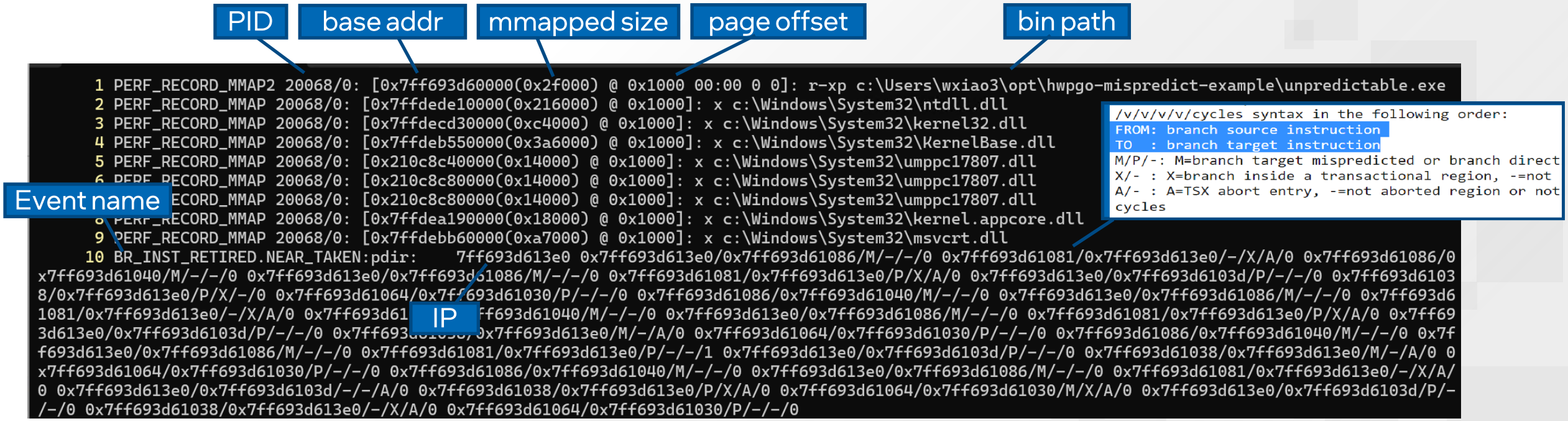
- Windows (and Linux) HWPGO feature supported since Intel® oneAPI DPC++/C++ Compiler 2024.0 release
  - LLVM-based Intel proprietary compiler released in Nov 2023
  - https://www.intel.com/content/www/us/en/docs/dpcpp-cpp-compiler/developer-guide-reference/current/hardware-profile-guided-optimization.html
  - https://www.intel.com/content/www/us/en/developer/articles/technical/hwpgo.html
- Basic Windows (and Linux) SPGO/HWPGO features now available in LLVM trunk – as of Mar 2024
  - https://clang.llvm.org/docs/UsersManual.html#id50
  - Features mostly contributed by Intel - ported from the Intel proprietary codebase above
  - Requires use of Intel VTune SEP from oneAPI 2024.0

- **These are the first Windows compilers to support SPGO/HWPGO**

Windows Support: Profiling Tool

Intel® VTune™ SEP supports Linux perf script output format since oneAPI 20

- $ sep -perf-script event,ip,brstack –ec BR_INST_RETIRED.NEAR_TAKEN …



Windows Support: How Symbolization is Handled

Use DWARF Instead of PDB

PDB encode de-mangled (display) names

DWARF encode mangled (linkage) names

```
29  namespace MyNameSpace2 {
30      template <typename T1, typename T2>
31      void init(T1 c,
32                T2 *p)
33      {
38      void
39      print(unsigned short *p, int i) {
44      }
45  }

64  int main (int argc, char *argv[]) {
65    g = argc;
66    MyNameSpace2::init(argc, M);
67    MyNameSpace1::MyClass<unsigned short, int> Obj;
73    MyNameSpace2::print(M, result % argc);
74    return 0;
75  }
```

```
*** INLINEE LINES

** Module: "C:\Users\wxiao3\AppData

InlineeId  FileId  StaringLine
     1003       1          635
     1210       0           31
     1211       0           23
     1212       0            6
     1213       0           16
     120C       0           39
```

DWARF: ??$init@HG@MyNameSpace2@@YAXHPEAG@Z

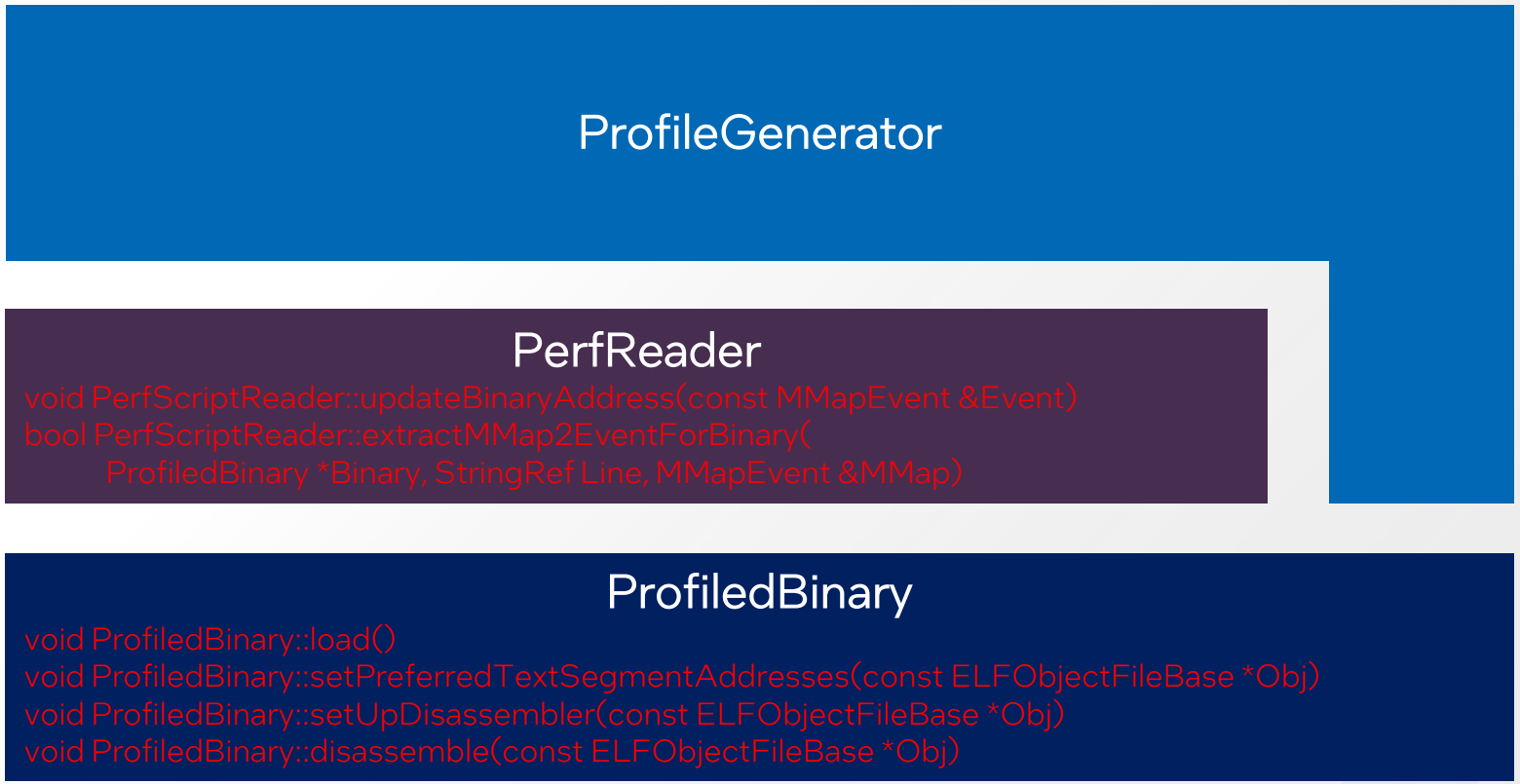
```
0x1210 : Length = 18, Leaf = 0x1601 LF_FUNC_ID
         Type = 0x1013              Scope = 0x1000  init
```

DWARF: ?print@MyNameSpace2@@YAXPEAGH@Z

```
(0000BC) S_GPROC32: [0001:000003B0], Cb: 00000030, Type:          0x1001, MyNameSpace2::print
         Parent: 00000000, End: 00000150, Next: 00000000
         Debug start: 00000000, Debug end: 00000000
```

Windows Support: Changes made to llvm-profgen

Understand COFF/PE with DWARF by enhancing:



https://github.com/tcreech-intel/hwpgo-mispredict-example

Microsoft Teams

HW-based PGO Demo

2024-04-08 03:42 UTC

Recorded by
Xiao, Wei3

Organized by
Xiao, Wei3

Challenges & Solutions

Usability

- Many flags (**-fdebug-info-for-profiling, -funique-internal-linkage-names, -gdwarf, /debug:dwarf, …**) needed to produce good debug info. Need to consolidate/simplify.
  - OneAPI compilers have "**-fprofile-sample-generate**"
- Multiple profile types (frequency, branch mispredicts, etc.) will become difficult to produce, manage, and pass to the compiler.
  - Considering profile "bundles" and higher-level tools to drive creation of PMU profiles.

Challenges & Solutions (2)

Debug Info Accuracy

- HWPGO uses debug information (such as DWARF) to associate profile data from the optimized binary to source code and compiler IR.
  - Pro: neither prevent any optimizations nor add run-time overhead to the profiling binary.
  - Con: suffer from inaccurate correlation with aggressive optimizations.
- Solutions:
  - Enhance Debug Info.
  - Turn off aggressive optimizations.
  - PSEUDO-INSTRUMENTATION.

```
 1 unpredictable:12711006836:0
 2  0: 0
 3  3.1: 200031858
 4  3.2: 200031858
 5  5: 200031858
 6  6: 116870734
 7  7: 116870734
 8  8: 121580402 nop:121580402
 9  11: 84838540 nop:84838540
10  13: 200031858
11  15: 0
12  65517: 116870734
13 nop:192386712:206418942
14  1: 192386712
```

```
Initial selection DAG: %bb.14 'foo:entry'
SelectionDAG has 5 nodes:
    t0: ch,glue = EntryToken
  t2: i64,ch = CopyFromReg t0, Register:i64 %1, jump_table.c:4:3
  t4: ch = br_jt t2:1, JumpTable:i64<0>, t2, jump_table.c:4:3
```

https://github.com/llvm/llvm-project/pull/71021

Challenges & Solutions (3)

Profile Maintenance

- After each optimization, profile (probability) needs to be adjusted to reflect control flow graph changes if any.
- Example below shows one of the bug-fixes made recently:

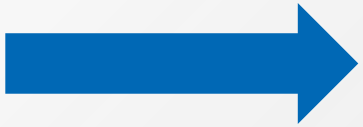
```
define i32 @invert_cond(ptr %p) {
entry:
  %cond = icmp ne ptr %p, null
  br i1 %cond, label %bb1, label %bb2, !prof !0

bb1:
  ret i32 0

bb2:
  ret i32 1
}

!0 = !{!"branch_weights", i32 1, i32 20}
```

InstCombine



```
define i32 @invert_cond(ptr %p) {
entry:
  %cond.not = icmp eq ptr %p, null
  br i1 %cond.not, label %bb2, label %bb1, !prof !0

bb1:                                              ;
  ret i32 0

bb2:                                              ;
  ret i32 1
}

!0 = !{!"branch_weights", i32 20, i32 1}
```

https://github.com/llvm/llvm-project/pull/86470

Challenges & Solutions (4)

Value Profiling

- If both PEBS and LBR records are captured, we can sample both function call counts and function limited arguments

  $ perf record --user-regs -b -e xxx

```
 BR_INST_RETIRED.NEAR_TAKEN:uppp:            401168 ABI:2    AX:0x1    BX:0x1d0a03c    CX:0x1d023c0
 DX:0x1ceeb30    SI:0x3    DI:0x1d0a03c    BP:0x1f1f    SP:0x7ffde51a1868    IP:0x401168 FLAGS:0x8
CS:0x33    SS:0x2b    R8:0x1d15c50    R9:0x1    R10:0xfff    R11:0x1d23000    R12:0x1d15c50    R13:0x1cf67
ac    R14:0x1d0a038    R15:0x1cdb2a0  0x401168/0x4016b0/P/-/-/18  0x401193/0x401160/M/-/-/3  0x4016b0/0x
40116d/P/-/-/8  0x401168/0x4016b0/P/-/-/18  0x401193/0x401160/M/-/-/3  0x4016b0/0x40116d/P/-/-/8  0x40
1168/0x4016b0/P/-/-/15  0x401193/0x401160/M/-/-/3  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/
1  0x4011b0/0x4016b0/P/-/-/10  0x4016b0/0x40116d/P/-/-/7  0x401168/0x4016b0/P/-/-/14  0x401193/0x40116
0/M/-/-/3  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/29  0x4011b5/
0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/29  0x4011b5/0x401170/P/-/-/1  0x
4016b0/0x4011b5/P/-/-/1  0x4011b0/0x4016b0/P/-/-/28  0x4011b5/0x401170/P/-/-/1  0x4016b0/0x4011b5/P/-/
-/1  0x4011b0/0x4016b0/P/-/-/37  0x4016b0/0x40116d/P/-/-/7  0x401168/0x4016b0/P/-/-/18  0x401193/0x401
160/M/-/-/6  0x4016b0/0x40116d/P/-/-/7  0x401168/0x4016b0/P/-/-/4  0x401193/0x401160/M/-/-/3
```

List of PRs checked into LLVM Trunk so far

- Refer to: https://clang.llvm.org/docs/UsersManual.html#using-sampling-profilers
- [llvm-profgen] Support COFF binary: 83972
- [LLD] [COFF] Port -lto-sample-profile to COFF version of LLD: 85701
- Update documentation and release notes for llvm-profgen COFF support: 84864
- Profile Maintenance:
  - LoopRotate: 86496
- DebugInfo Fix:
  - JumpTable: 71018, 72075 , 72082, 72118, 71021
  - CodeGen: 72192
- Support –gsplit-dwarf for COFF (RFC: 71276):
  - MC: D151793, D152119, D152229, D152340
  - Clang & MC: D152785, 82dff24bde112984314568e7d581379fd0ea48e6
  - [LLD][COFF]: D154070 (to support /dwodir for LTO)
  - Clang: D154176, D154295
- Emit symbol-table for COFF:
  - [LLD][COFF]: D149235
- Fix HW-based PGO/Sampling-based PGO gap with Instrumentation-based PGO:
  - InlineCost: 66457
  - InstCombine: 68474, 68502

Summary

- HW-based PGO is an extension of existing Sampling-based PGO for:
  - Lower overhead
  - Higher accuracy
  - New feedback capabilities for higher performance gains

- Call community collaboration on HW-based PGO to:
  - Add infrastructure to support more feedback/profile types besides frequency
  - Add optimizations for new feedback/profile types
  - Enhance Debug Info Accuracy
  - Enhance Profile Maintenance
  - Support Value Profiling


Q
&
A

Legal Disclaimer & Optimization Notice

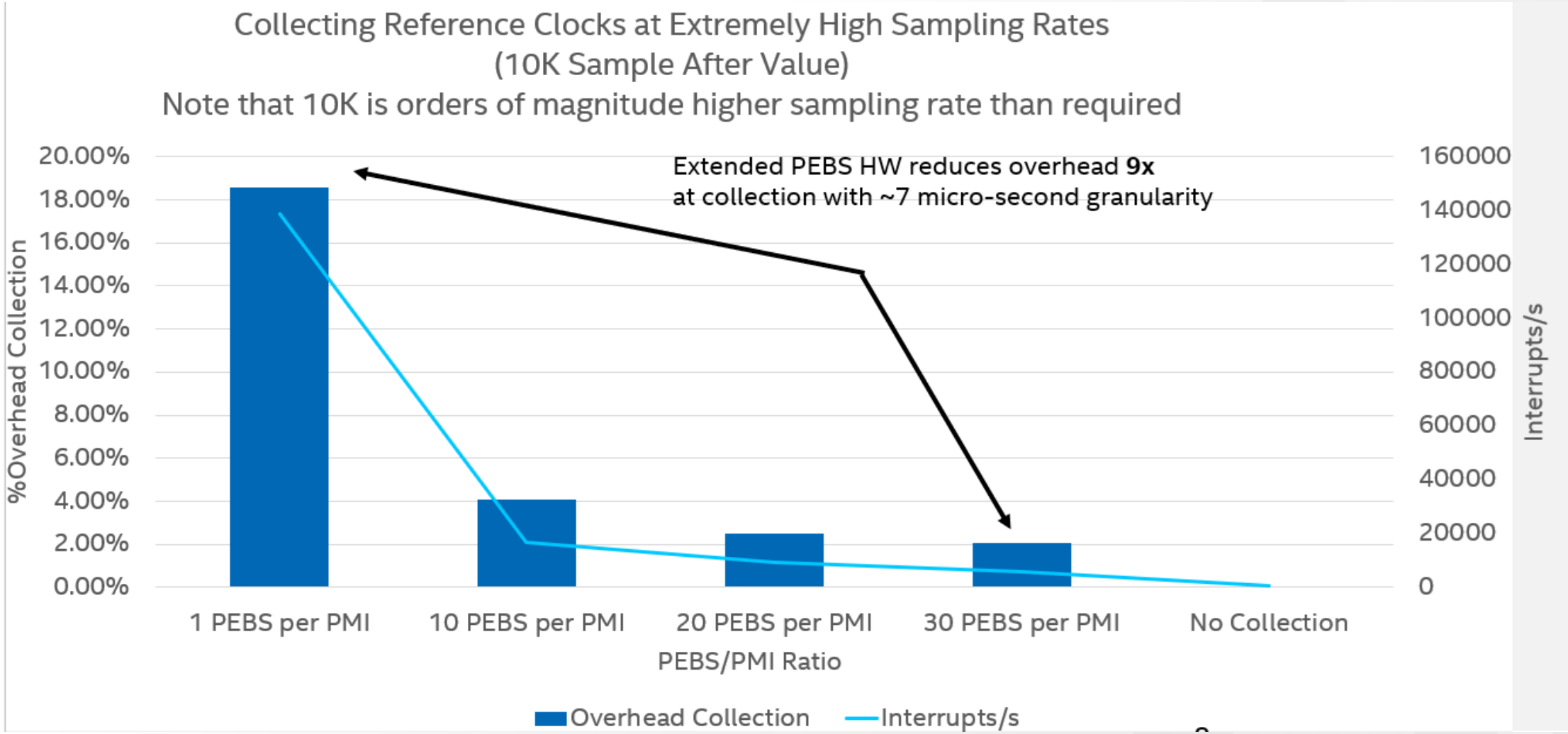
- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
-

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel

Collection Overhead Reduced by Extended PEBS



New Feedback Capabilities

Branch Mispredict Feedback CoreMark®-PRO Example:

coremark-pro/benchmarks/consumer_v2/cjpeg/jcdctmgr.c

Poorly predicted according to HW events:
- **BR_INST_RETIRED.NEAR_TAKEN**
- **BR_MISP_RETIRED.ALL_BRANCHES**

```
223     #define DIVIDE_BY(a,b)    if (a >= b) a /= b; else a = 0
224     #endif
225         if (temp < 0) {
226           temp = -temp;
227           temp += qval>>1;    /* for rounding */
228           DIVIDE_BY(temp, qval);
229           temp = -temp;
230         } else {
231           temp += qval>>1;    /* for rounding */
232           DIVIDE_BY(temp, qval);
233         }
```

HWPGO

```
t12 = -temp + qval >> 1
if (t12 >= qval)
  t15 = t12 / (temp < 0 ? qval : 1)
else
  t15 = 0
t16 = -t15

t21 = temp + qval >> 1
if (t21 >= qval)
  t24 = t21 / (temp >= 0 ? qval : 1)
else
  t24 = 0

temp = temp < 0 ? t16 : t24
```

New Feedback Capabilities

Branch Mispredict Feedback CoreMark®-PRO Example:

Before HWPGO:

```
Performance counter stats for './cjpeg-rose7-preset.exe.Vanilla -v0 -i1500':

      4,255.40 msec task-clock                       #    0.998 CPUs utilized
            18      context-switches                 #    4.230 /sec
             1      cpu-migrations                   #    0.235 /sec
           178      page-faults                      #   41.829 /sec
22,051,463,005      cpu_core/cycles/                 #    5.182 G/sec
   <not counted>    cpu_atom/cycles/                                        (0.00%)
48,356,977,972      cpu_core/instructions/           #   11.364 G/sec
   <not counted>    cpu_atom/instructions/                                  (0.00%)
 4,727,869,197      cpu_core/branches/               #    1.111 G/sec
   <not counted>    cpu_atom/branches/                                      (0.00%)
   436,828,793      cpu_core/branch-misses/          #  102.653 M/sec
   <not counted>    cpu_atom/branch-misses/                                 (0.00%)
132,308,528,448     cpu_core/slots/                  #   31.092 G/sec
 44,620,028,215     cpu_core/topdown-retiring/       #     33.6% Retiring
 36,836,285,578     cpu_core/topdown-bad-spec/       #     27.7% Bad Speculation
 36,839,837,672     cpu_core/topdown-fe-bound/       #     27.7% Frontend Bound
 14,531,129,481     cpu_core/topdown-be-bound/       #     10.9% Backend Bound
    519,327,103     cpu_core/topdown-heavy-ops/      #      0.4% Heavy Operations       #     33.2% Light Operations
 36,836,233,341     cpu_core/topdown-br-mispredict/  #     27.7% Branch Mispredict      #      0.0% Machine Clears
 22,311,685,681     cpu_core/topdown-fetch-lat/      #     16.8% Fetch Latency          #     10.9% Fetch Bandwidth
  2,077,047,233     cpu_core/topdown-mem-bound/      #      1.6% Memory Bound           #      9.4% Core Bound

    4.265031168 seconds time elapsed

    4.252556000 seconds user
    0.004000000 seconds sys
```

-33%

+17%

After HWPGO:

```
Performance counter stats for './cjpeg-rose7-preset.exe.New -v0 -i1500':

      3,118.57 msec task-clock                       #    0.997 CPUs utilized
            10      context-switches                 #    3.207 /sec
             1      cpu-migrations                   #    0.321 /sec
           177      page-faults                      #   56.757 /sec
16,174,859,445      cpu_core/cycles/                 #    5.187 G/sec
   <not counted>    cpu_atom/cycles/                                        (0.00%)
56,997,553,072      cpu_core/instructions/           #   18.277 G/sec
   <not counted>    cpu_atom/instructions/                                  (0.00%)
 4,751,693,034      cpu_core/branches/               #    1.524 G/sec
   <not counted>    cpu_atom/branches/                                      (0.00%)
   186,405,909      cpu_core/branch-misses/          #   59.773 M/sec
   <not counted>    cpu_atom/branch-misses/                                 (0.00%)
97,048,835,412      cpu_core/slots/                  #   31.120 G/sec
53,277,581,186      cpu_core/topdown-retiring/       #     54.9% Retiring
23,213,761,622      cpu_core/topdown-bad-spec/       #     23.9% Bad Speculation
 9,898,710,296      cpu_core/topdown-fe-bound/       #     10.2% Frontend Bound
10,658,832,092      cpu_core/topdown-be-bound/       #     11.0% Backend Bound
   381,031,756      cpu_core/topdown-heavy-ops/      #      0.4% Heavy Operations       #     54.5% Light Operations
23,213,662,047      cpu_core/topdown-br-mispredict/  #     23.9% Branch Mispredict      #      0.0% Machine Clears
 4,950,077,068      cpu_core/topdown-fetch-lat/      #      5.1% Fetch Latency          #      5.1% Fetch Bandwidth
   381,678,995      cpu_core/topdown-mem-bound/      #      0.4% Memory Bound           #     10.6% Core Bound

    3.128461260 seconds time elapsed
```

Call community collaboration on HWPGO to:

- Add infrastructure support for more feedback/profile types besides frequency (i.e., "-fprofile-sample-use=code.freq.prof")
- Add optimizations for new profile types

Windows Support: llvm-profgen

Canonicalize WINDOWS Virtual Address for COFF/PE

```
// Canonicalize to use preferred load address as base address.
uint64_t canonicalizeVirtualAddress(uint64_t Address) {
  return Address - BaseAddress + getPreferredBaseAddress();
}
```

```
5 PERF_RECORD_MMAP2 1764/0: [0x00007ff9f3271000(0x294000) @ 0x1000 ]: x C:\Windows\System32\KernelBase.dll
6 PERF_RECORD_MMAP2 1764/0: [0x00007ff9f2661000(0xfa000) @ 0x1000 ]: x C:\Windows\System32\ucrtbase.dll
7 PERF_RECORD_MMAP2 1764/0: [0x00007ff9f2531000(0x11000) @ 0x1000 ]: x C:\Windows\System32\kernel.appcore.dll
8 PERF_RECORD_MMAP2 1764/0: [0x00007ff9ef461000(0x1a000) @ 0x1000 ]: x C:\Windows\System32\vcruntime140.dll
9 PERF_RECORD_MMAP2 1764/0: [0x00007ff9ef451000(0xc000) @ 0x1000 ]: x C:\Windows\System32\vcruntime140_1.dll
10 PERF_RECORD_MMAP2 1764/0: [0x00007ff9cc161000(0x8d000) @ 0x1000 ]: x C:\Windows\System32\msvcp140.dll
11 PERF_RECORD_MMAP2 1764/0: [0x00007ff6ca0d0000(0x5a000) @ 0x1000 ]: x D:\wxiao3\opt\spgo\sort\sort.exe
```

Binary->getTextSegmentOffset()

Event.Offset

Disasm | General | DOS Hdr | File Hdr | Optional Hdr | Section Hdrs | Imports

Offset	Name	Value	Value
90	Magic	20B	NT64
92	Linker Ver. (Major)	E	
93	Linker Ver. (Minor)	0	
94	Size of Code	3A00	
98	Size of Initialized Data	3000	
9C	Size of Uninitialized Data	0	
A0	Entry Point	4068	
A4	Base of Code	1000	
A8	Image Base	140000000	
B0	Section Alignment	1000	
B4	File Alignment	200	
B8	OS Ver. (Major)	6	Windows Vista / Server 2008
BA	OS Ver. (Minor)	0	
BC	Image Ver. (Major)	0	

Disasm | General | DOS Hdr | File Hdr | Optional Hdr | Section Hdrs | Imports | Exception | BaseReloc.

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
.text	400	3A00	1000	3926	60000020	0	0	0
.rdata	3E00	2200	5000	20C4	40000040	0	0	0
.data	6000	400	8000	4EB50	C0000040	0	0	0
.pdata	6400	600	57000	444	40000040	0	0	0
.00cfg	6A00	200	58000	28	40000040	0	0	0
.voltbl	6C00	200	59000	18	0	0	0	0
.reloc	6E00	200	5A000	A0	42000040	0	0	0

HWPGO Documentation/Links

Intel® oneAPI DPC++/C++ Compiler:

- https://www.intel.com/content/www/us/en/docs/dpcpp-cpp-compiler/developer-guide-reference/current/hardware-profile-guided-optimization.html
- https://www.intel.com/content/www/us/en/developer/articles/technical/hwpgo.html
- https://github.com/tcreech-intel/hwpgo-mispredict-example

LLVM:

- https://clang.llvm.org/docs/UsersManual.html#using-sampling-profilers
- Unmerged branch mispredict feedback features:
  - https://github.com/tcreech-intel/llvm-project/tree/ip_profiles
  - https://github.com/tcreech-intel/llvm-project/tree/unpredictable_loader
  - https://github.com/tcreech-intel/llvm-project/tree/aggressive_speculation

SPEC CPU2017 Performance on IceLake Windows Server

llvm-trunk 20240408	Default (Normalized Performance)	HW-based PGO	Instrumentation-based PGO
500.perlbench_r	**100%**	**110.19%**	**113.63%**
502.gcc_r	**100%**	**103.13%**	**105.28%**
511.povray_r	**100%**	**106.37%**	**110.41%**

HW-based PGO

- 1st build: /clang:-fdebug-info-for-profiling /clang:-funique-internal-linkage-names -gdwarf -gline-tables-only -fuse-ld=lld
- 2nd build: /clang:-fprofile-sample-use=default.profdata -gline-tables-only -fuse-ld=lld