Teaching perf to show processor hazards

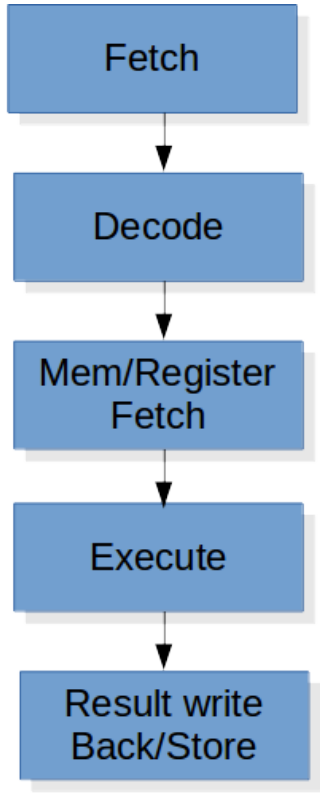
Madhavan Srinivasan
maddy@linux.vnet.ibm.com
Linux Technology Centre - IBM

Agenda

- Processor Pipeline
- Pipeline issues/Hazard
- IBM Processor Sampling support
- Perf API - arch neutral interface
- Perf tool – options and enhancements
- Screenshot

Instruction cycle

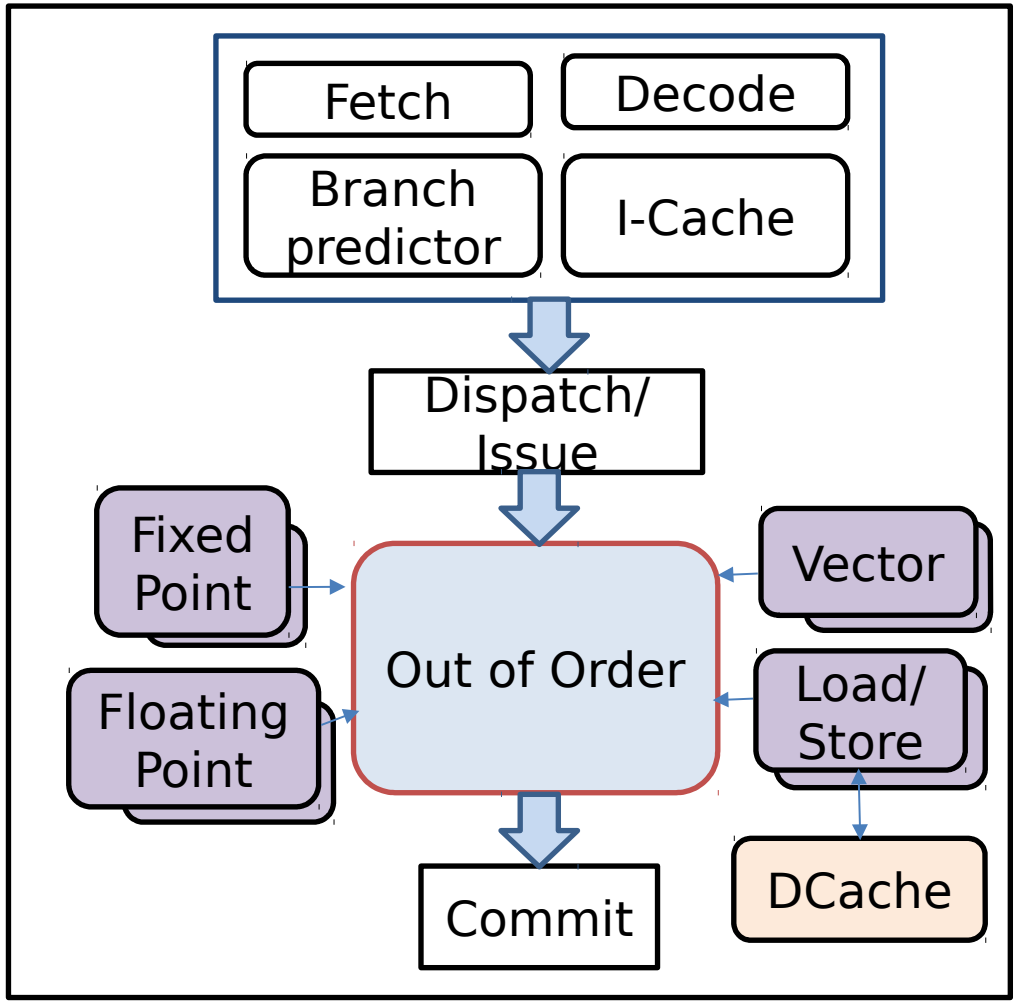
- Processing instruction includes these steps
  - Fetch opcode
  - Decode stage
  - register/memory fetch based on opcode type
  - Execute instruction
  - write-back/store the result



Basic Instruction pipeline

Instruction cycle

- Most modern microprocessors employ complex instruction execution (pipelined superscalar)
  - Multiple instruction in parallel
    - more execution units
    - Execution unit divided in different stages
  - Speculation/OOO Execution
  - Multiple different pipelines/Sub-pipelines



Example:IBM Power9

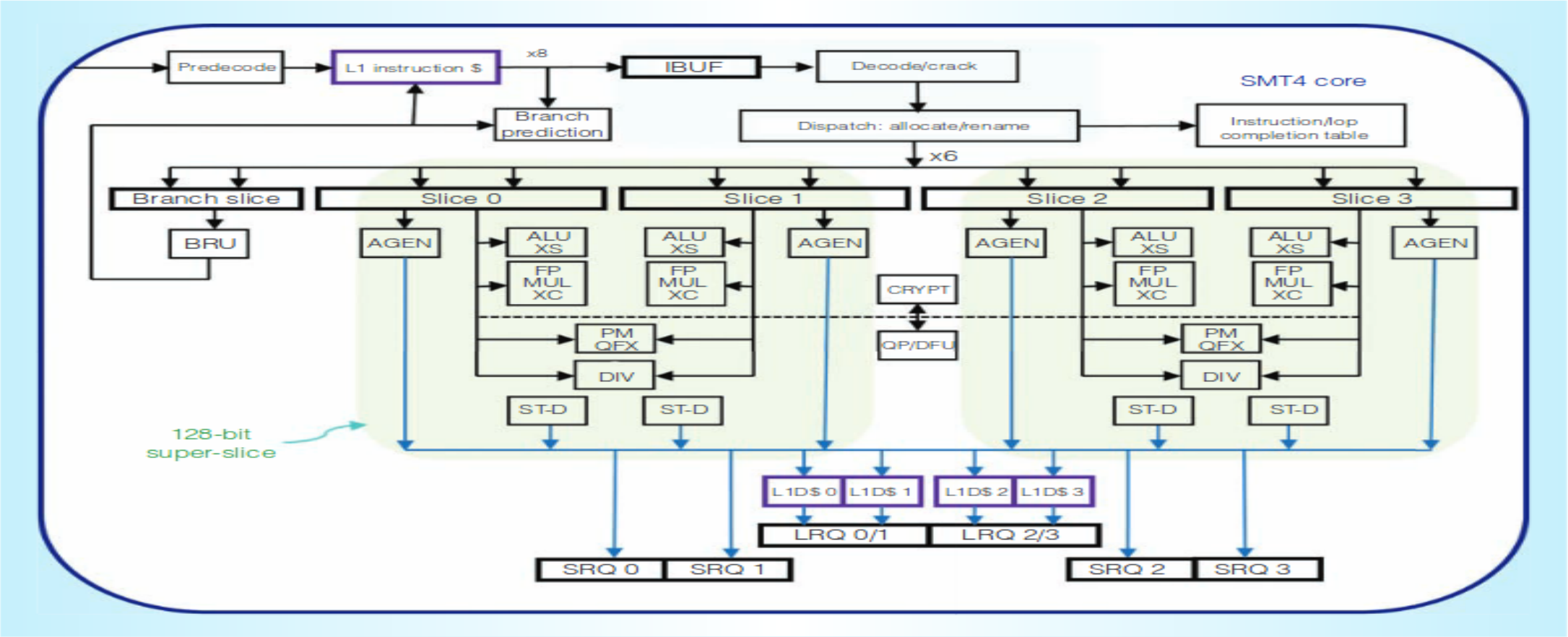


Figure 5. Power9 SMT4 core. The detailed core block diagram shows all the key components of the Power9 core.

Source: IBM Power9 Processor Architecture. Satish Kumar Sadasivam ; Brian W. Thompto; Ron Kalla ; William J. Starke. IEEE Micro, Volume 37, Issue 2, 2017. https://ieeexplore.ieee.org/document/7924241?section=abstract

Performance (Instruction per Cycles)

- Increases Stall cycles
- Reduce workload performance
- Lowers Instruction per cycles
  - Hazard

Hazards

- Prevent the next instruction in the instruction stream from being executing during its designated clock cycle
  - Performance hit
- Classes of Hazards:
  - Structural
    - part of the processor's hardware is needed by two or more instructions at the same time
  - Control
    - conditional branches interfere with instruction fetches in a pipeline
  - Data
    - instructions that exhibit data dependence modify data in different stages of a pipeline
    - Read after Write (RAW) hazards, also known as true dependences
    - Write after Write (WAW) hazards, also known as output dependences
    - Write after Read (WAR) hazards, also known as antidependences

IBM Power Processor Sampling Support

- Why sample
  - Identificatiion of hotspots in code/data and performance-sensitive areas
- Mark (sample) an instruction and collection details
  - 64bit register records details about marked instruction during its lifetime in pipeline
- Power ISA provides three special purpose registers
  - Sampled Instruction Address Register (SIAR)
  - Sampled Data Address Register (SDAR)
  - Sample Instruction Event Register (SIER)

- Source: https://www-355.ibm.com/systems/power/openpower/tgcmDocumentRepository.xhtml?aliasId=POWER9_Sforza#

Why perf?

- Performance tool used by eco-system
- Provides access to Performance Monitoring Unit (PMU)
  - Allows to closer look at hardware behaviour
- Capability to generate reports out of data collected
- It is fast, lightweight and precise

Why add hazard information in perf

- perf today support exporting memory sampling information
  - PERF_SAMPLE_DATA_SRC and PERF_SAMPLE_WEIGHT
- Based on hardware support, it expose
  - Instruction class (load, store ....)
  - where the data came from (memory hierarchy, hit, hitm, miss)
  - how long did it took for the reload (time in cycles)
  - Data translation (TLB), snoop

```
Samples: 7  of event 'cpu/mem-loads/pp', Event count (approx.): 66
Overhead  Symbol                  Shared Object      Memory access  Weight  Locked  TLB access     Snoop  Data Symbol
  37.88%  [k] queue_work_on       [kernel.kallsyms]  L1 hit         25      Yes     L1 or L2 hit   None   [k] 0xffff88021f94a408
  10.61%  [.] __overflow@plt      ls                 L1 hit         7       No      L1 or L2 hit   None   [.] 0x000000000061e150
  10.61%  [k] account_entity_enqueue [kernel.kallsyms] L1 hit      7       No      L1 or L2 hit   None   [k] 0xffff88022dc97380
  10.61%  [k] attach_entity_load_avg [kernel.kallsyms] L1 hit      7       No      L1 or L2 hit   None   [k] 0xffff8802207ebae0
  10.61%  [k] nmi_cpu_backtrace   [kernel.kallsyms]  L1 hit         7       No      L1 or L2 hit   None   [k] 0xffff88022dc48e20
  10.61%  [k] unmap_page_range    [kernel.kallsyms]  L1 hit         7       No      L1 or L2 hit   None   [k] 0xffff88018f7f3ad0
   9.09%  [.] 0x00000000000ddc1   ls                 L1 hit         6       No      L1 or L2 hit   None   [.] 0x00007ffc2baec668
```

Challenges extending -- *perf_mem_data_src*

- *perf_mem_data_src* intended for memory sampling
- Not enough bits to expose
  - pipeline stage
  - hazard reason
  - Stall reason
  - Other instruction class

```
#if defined(__LITTLE_ENDIAN_BITFIELD)
union perf_mem_data_src {
        __u64 val;
        struct {
                __u64   mem_op:5,       /* type of opcode */
                        mem_lvl:14,     /* memory hierarchy level */
                        mem_snoop:5,    /* snoop mode */
                        mem_lock:2,     /* lock instr */
                        mem_dtlb:7,     /* tlb access */
                        mem_lvl_num:4,  /* memory hierarchy level number */
                        mem_remote:1,   /* remote */
                        mem_snoopx:2,   /* snoop mode, ext */
                        mem_rsvd:24;
        };
};
#elif defined(__BIG_ENDIAN_BITFIELD)
union perf_mem_data_src {
        __u64 val;
        struct {
                __u64   mem_rsvd:24,
                        mem_snoopx:2,   /* snoop mode, ext */
                        mem_remote:1,   /* remote */
                        mem_lvl_num:4,  /* memory hierarchy level number */
                        mem_dtlb:7,     /* tlb access */
                        mem_lock:2,     /* lock instr */
                        mem_snoop:5,    /* snoop mode */
                        mem_lvl:14,     /* memory hierarchy level */
                        mem_op:5;       /* type of opcode */
        };
};
#else
#error "Unknown endianness"
#endif
```

Approach to export hazard data via *perf*

- Struct to collect hazard data
- Sampling type/format
- Tool option to notify hazard data collection
- New reporting mode to present the hazard data
- Optional new built-in tool (wrapper for perf record)
  - Capture and present Hazard data – Usability
  - Similar to "perf mem"

Hazard data – perf screenshot

```
#
# Samples: 41  of event 'r401e0'
# Event count (approx.): 28124
# Sort order   : sym,dso,class,hazard_stage,hazard_reason,stall_stage,stall_reason,icache,type
#
# Overhead  Symbol                          Shared Object     Instruction Class  Hazard Stage  Hazard Reason        Stall Stage  Stall Reason         ICache acc
# ........  ..............................  ................  .................  ............  ....................  ...........  ....................  ..........
#
    10.73%  [.] 0x0000000000b0414           libc-2.26.so      Load               LSU           TLB Miss             -            -                     L2 hit
     7.27%  [k] get_mem_cgroup_from_mm      [kernel.kallsyms] Load               IIU           Resource Collision   LSU          Dcache_miss           L2 hit
     6.01%  [.] 0x0000000000b0400           libc-2.26.so      Floating Point     IIU           Resource Collision   -            -                     L2 hit
     5.44%  [.] 0x0000000000b03fc           libc-2.26.so      Load               LSU           TLB Miss             -            -                     L2 hit
     5.32%  [.] 0x0000000000b0400           libc-2.26.so      Store              -             -                    -            -                     L2 hit
     5.01%  [.] 0x0000000000b040c           libc-2.26.so      Floating Point     IIU           Resource Collision   VX           Others                L2 hit
     4.91%  [.] 0x0000000000b0408           libc-2.26.so      Store              -             -                    -            -                     L2 hit
     4.63%  [.] 0x0000000000b041c           libc-2.26.so      Fixed point        -             -                    -            -                     L2 hit
```

perf_pipeline_haz_data

- Pipeline Stages as u32
  - Arch can decide how many
  - Bit mask or Value as index
- Hazard and stall reasons as separate fields
  - Cleaner implementation
  - Multiple hazard representation
- Instruction cache hierarchy
- Processor version
  - tool side to post process

```
struct perf_pipeline_haz_data {
        /* Class: Load, Store, Branch */
        u32     class;
        /* Type: Multiple/Single cycle/byte */
        u32     type;
        /* Instruction Cache source     */
        u32     icache;
        /* Suffered hazard in pipeline stage */
        u32     hazard_stage;
        /* Hazard reason */
        u32     hazard_reason;
        /* Suffered Stall in pipeline stage */
        u32     stall_stage;
        /* Stall reason */
        u32     stall_reason;
        /* Processor Version information */
        u32     proc_ver;
};
```

perf_pipeline_haz_data – struct to collect hazard data

- Added new perf sample type/format
  - PERF_SAMPLE_PIPELINE_HAZ
- Proposed to be part of *include/uapi/linux/perf_event.h*
- Macros could be part of arch folder (ex.. *arch/powerpc/include/uapi/asm/perf_pipeline_haz.h*

```
/* Macros for the Instruction Class */
enum perf_pipleline_inst_class {
        PERF_PIPELINE_ICLASS_LOAD = 1,
        PERF_PIPELINE_ICLASS_STORE,
        PERF_PIPELINE_ICLASS_BRANCH,
        PERF_PIPELINE_ICLASS_FP,
        PERF_PIPELINE_ICLASS_FX,
        PERF_PIPELINE_ICLASS_IFU_NON_BRANCH,
};

/* Macros for Pipeline units */
enum perf_pipeline_stage {
        PERF_PIPELINE_STAGE_IFU = 1,
        PERF_PIPELINE_STAGE_IDU,
        PERF_PIPELINE_STAGE_IIU,
        PERF_PIPELINE_STAGE_LSU,
        PERF_PIPELINE_STAGE_BR,
        PERF_PIPELINE_STAGE_FX,
        PERF_PIPELINE_STAGE_FP,
        PERF_PIPELINE_STAGE_VX,
};

/* Macros for the Instruction Cache */
#define HAZ_ICACHE_SHIFT        0x2
#define HAZ_ICACHE_VAL(x)       (x) << (HAZ_ICACHE_SHIFT)
#define HAZ_ICACHE_HIT          0x1
#define HAZ_ICACHE_MISS         0x2
enum perf_pipeline_icache {
        PERF_PIPELINE_ICACHE_L1 = 1,
        PERF_PIPELINE_ICACHE_L2,
        PERF_PIPELINE_ICACHE_L3,
};
```

perf tool -- Enhancements for hazard capture

- New perf tool option
  - User to indicate hazard data capture
  - Proposing "-H" as option
  - Needed to enable attr_sample_type

```
#./perf record -H -e r401e0 ./ebizzy -t 1 -S 1 -m 4096
24166 records/s
real  1.00 s
user  0.43 s
sys   0.57 s
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.005 MB perf.data (45 samples) ]
#
```

Raw event "*r401e0*" used here is "*PM_MRK_INST_CMPL*" which enables IBM Power processor sampling support to capture hazard/stall data

perf tool – Enhancements for hazard capture

- Support functions to present raw hazard structure data
  - Perf report "–D" support

```
0xdbe [0x50]: event: 9
.
. ... raw event: size 80 bytes
.  0000:  09 00 00 00 01 00 50 00 18 6f 02 00 00 00 00 c0  ......P..o......
.  0010:  e4 66 00 00 e4 66 00 00 f6 c7 01 f6 a1 48 00 00  .f...f.......H..
.  0020:  01 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00  ................
.  0030:  09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
.  0040:  05 00 00 00 02 00 00 00 90 b5 d8 01 00 00 00 c0  ................

79860454246390 0xdbe [0x50]: PERF_RECORD_SAMPLE(IP, 0x1): 26340/26340: 0xc000000000026f18 period: 1 addr: 0
... hazard information:
.... Inst Class 0x6
.... Inst Type 0x0
.... Inst CACHE 0x9
.... Hazard Stage 0x0
.... Hazard Reason 0x0
.... Stall Stage 0x5
.... Stall Reason 0x2
.... Processor Version 0x1d8b590
 ... thread: perf:26340
 ..... dso: [kernel.kallsyms]
```

Screenshot show one PERF_RECORD_SAMPLE data output from "perf report –D" command. Presents all the elements of *perf_pipeline_haz_data* struct

hazard-info – perf report enhancement

- New "--hazard-info" mode
- Support new –sort types
- Focused on hazard data presentation

```
#
# Samples: 41  of event 'r401e0'
# Event count (approx.): 28124
# Sort order   : sym,dso,class,hazard_stage,hazard_reason,stall_stage,stall_reason,icache,type
#
# Overhead  Symbol                          Shared Object     Instruction Class  Hazard Stage  Hazard Reason          Stall Stage   Stall Reason          ICache acc
# ........  ..............................  ................  .................  ............  ......................  ............  ....................  ..........
#
    10.73%  [.] 0x00000000000b0414          libc-2.26.so      Load               LSU           TLB Miss               -             -                     L2 hit
     7.27%  [k] get_mem_cgroup_from_mm      [kernel.kallsyms] Load               IIU           Resource Collision     LSU           Dcache_miss           L2 hit
     6.01%  [.] 0x00000000000b0400          libc-2.26.so      Floating Point     IIU           Resource Collision     -             -                     L2 hit
     5.44%  [.] 0x00000000000b03fc          libc-2.26.so      Load               LSU           TLB Miss               -             -                     L2 hit
     5.32%  [.] 0x00000000000b0400          libc-2.26.so      Store              -             -                      -             -                     L2 hit
     5.01%  [.] 0x00000000000b040c          libc-2.26.so      Floating Point     IIU           Resource Collision     VX            Others                L2 hit
     4.91%  [.] 0x00000000000b0408          libc-2.26.so      Store              -             -                      -             -                     L2 hit
     4.63%  [.] 0x00000000000b041c          libc-2.26.so      Fixed point        -             -                      -             -                     L2 hit
```

Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of the employers (IBM Corporation).
- IBM and IBM (Logo) are trademarks or registered trademarks of International Business Machines in United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product and service names may be trademarks or service marks of others.

Thank you