CHINA LINUX KERNEL
中国Linux内核开发者大会

華中科技大學
网络空间安全学院
School of Cyber Science and Engineering, HUST

第19届中国
Linux内核开发者大会

AMD平台Core&&Uncore PMU虚拟化技术实践

郑翔 陈培鸿

CHINA LINUX KERNEL
中国Linux内核开发者大会

華中科技大學
网络空间安全学院
School of Cyber Science and Engineering, HUST

Content

目录

01 vPMU背景及现状概述

02 AMD vPMU技术探索

03 AMD vPMU应用实践

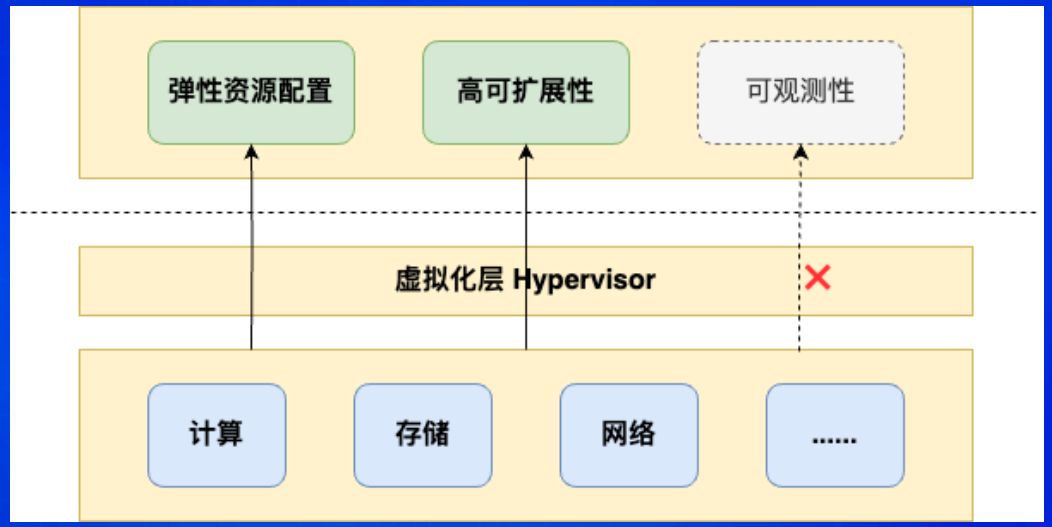
04 总结与未来演进

Part 01

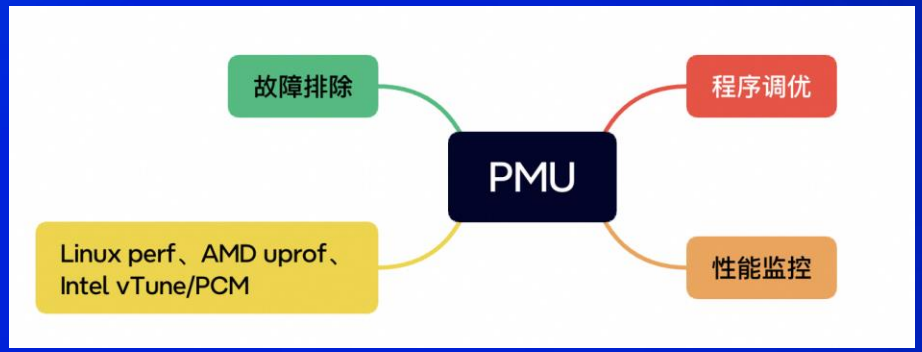
vPMU背景及现状概述

云环境虚拟机带内监控的诉求

- 云平台客户离在线业务调度、性能监控和调优越来越依赖底层PMU能力。虚拟化层的引入，对虚拟机内部用户屏蔽了大部分PMU能力，业务监控和优化工作变得复杂。



- 硬件性能监控单元（PMU）
  - 处理器内置的硬件组件，高精度、低开销。
  - 测量处理器工作期间各个硬件器件的运行情况和性能数据，辅助工程师排查分析软硬件上的性能问题。



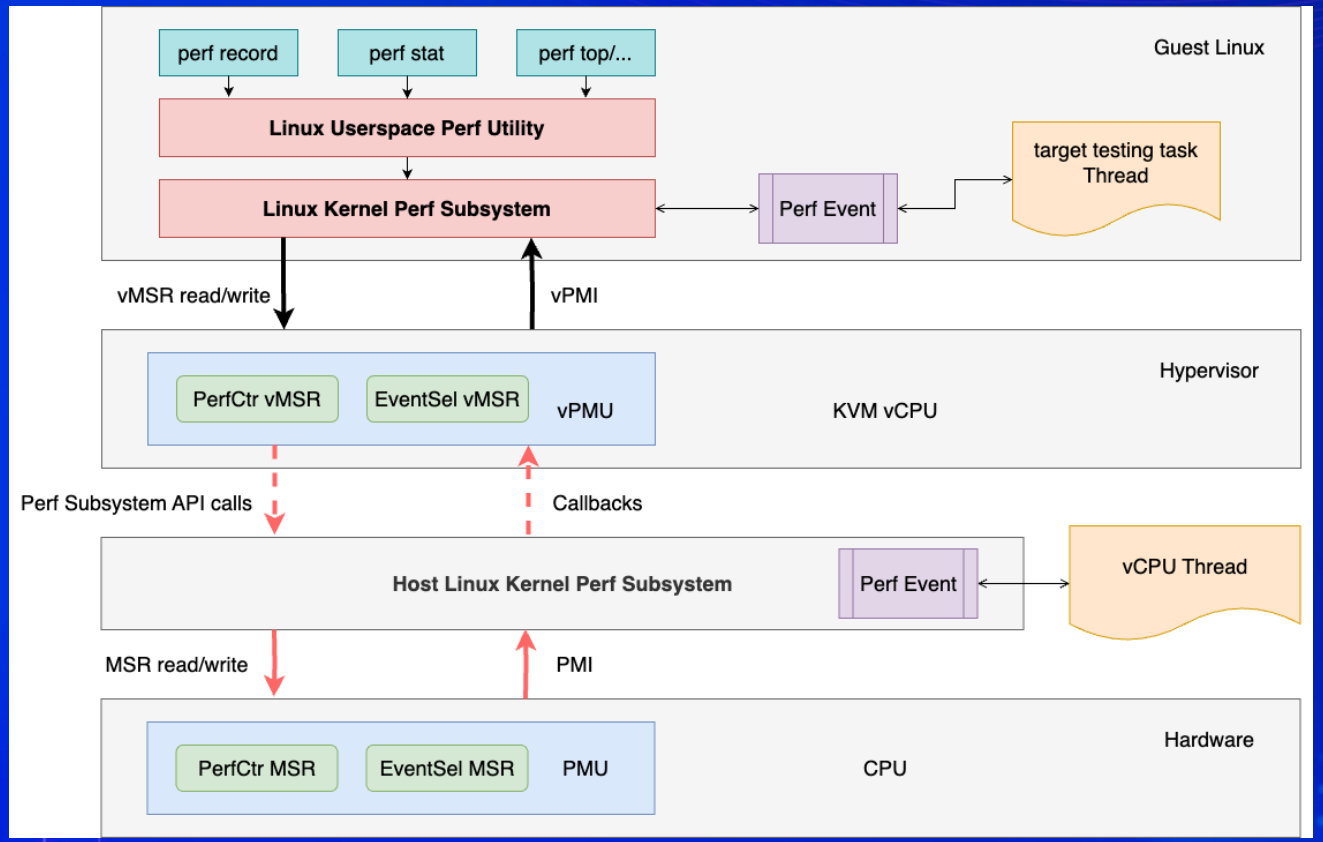
通过将PMU硬件资源虚拟化后呈现给虚拟机，允许Guest用户在VM里利用PMU实现带内性能监控。

QEMU/KVM PMU虚拟化方案（Emulated vPMU）

- 当前QEMU/KVM vPMU是一种基于Linux perf子系统模拟的虚拟化方案。

  - 硬件资源模拟：各类MSRs寄存器PerfCtr/EventSel/GlobalCtrl/GlobalStatus等，Guest无法直接访问。

  - 硬件工作机制模拟：拦截Guest对vPMU MSRs的访问，KVM通过创建perf event实现Guest对PMU硬件的间接访问，并为Guest注入vPMI中断。



阿里云在AMD vPMU领域的探索

- QEMU/KVM vPMU现状与不足：
  - 只具备最基础的Core PMU能力，缺少LBR/IBS等高级扩展特性。
  - 不支持Uncore PMU，无法采集监控Uncore组件性能指标（DMA访存带宽/内存带宽/LLC访问指标等）。

- 阿里云探索和补齐了QEMU/KVM上述缺失的AMD vPMU特性：
  - LBR/IBS：提供功能强大、高精度、低开销的Core PMU扩展。
  - Uncore PMU：业界首次尝试虚拟化，Guest带内性能监控成为可能。
  - 支持热迁移能力，持续优化vPMU的性能开销、精度以及稳定性。

vPMU	Intel	AMD	
	QEMU/KVM	QEMU/KVM	阿里云自研
Arch vPMU	✅ (v2)	✅ (v2)	✅ (v2)
vLBR	✅ Non-arch LBR ❌ Arch LBR	✅ LBR(v1) ❌ LBR Stack (v2)	✅ LBR(v1) ✅ LBR Stack (v2)
vPEBS/vIBS	✅	❌	✅
Uncore vPMU	❌	❌	✅

Part 02

AMD vPMU技术探索

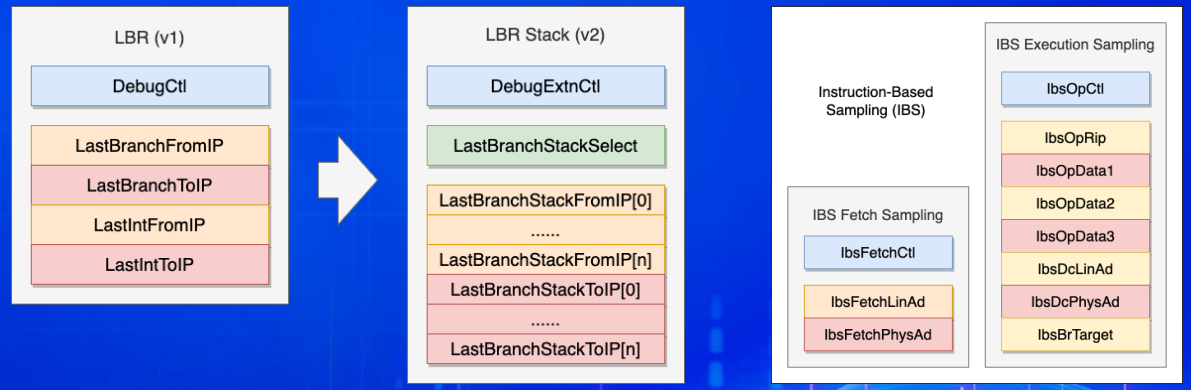
AMD PMU特性概览

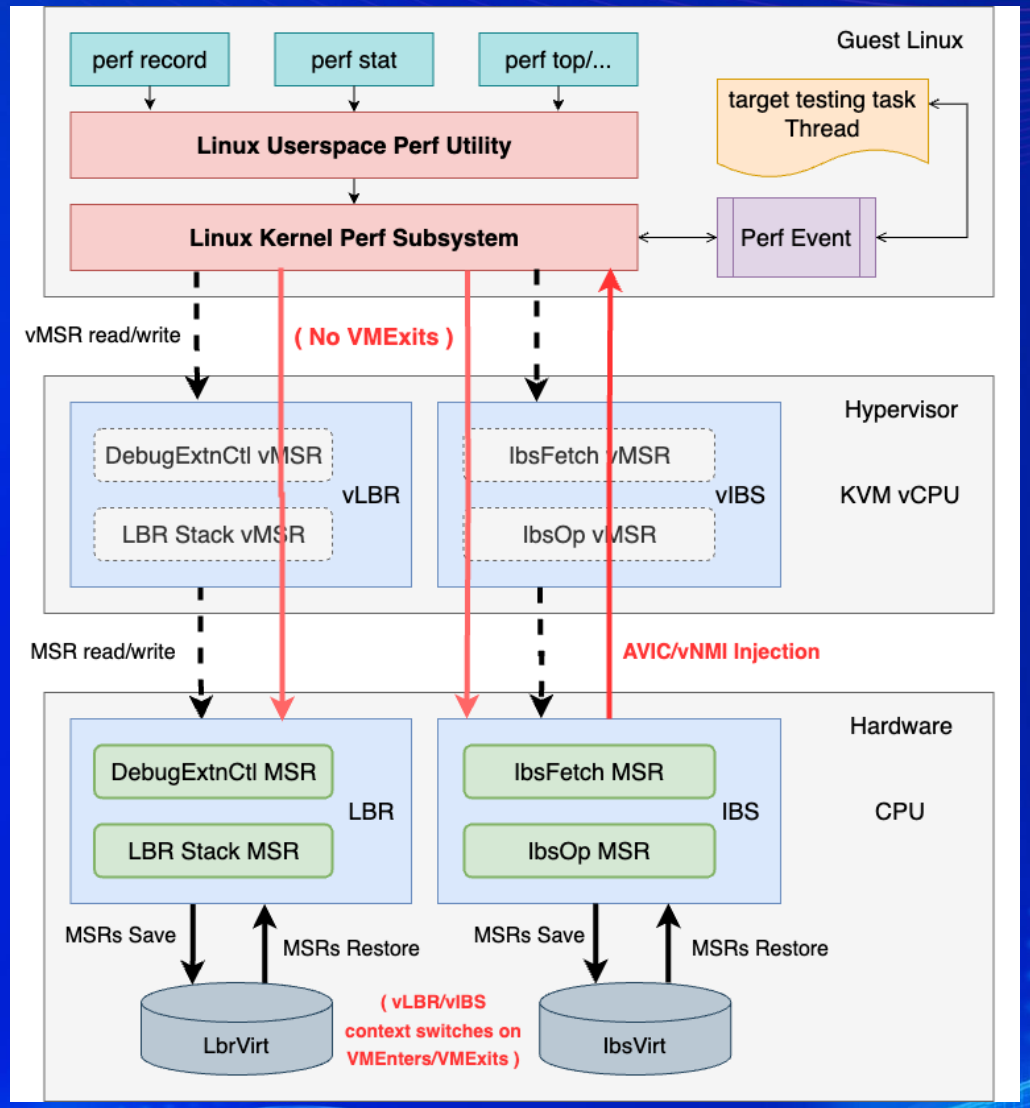
特性		功能描述	共享粒度	硬件辅助虚拟化	Patches/RFC
Core	Arch PMU	监控Core内部事件，例如cpu周期数（cycles）、分支指令数（branches）以及缓存访问等。	Per-HT	Zen5支持	[1]
	LBR (v1)	提供了一组MSRs用于保存最近分支跳转记录。	Per-HT	✅	[2]
	LBR Stack (v2)	LBR扩展，支持更多的LBR MSRs，用于跟踪程序的执行控制流以分析程序热点路径，并且提供统一的访问接口和代际兼容能力。	Per-HT	✅	N/A
	IBS	提供更高精度（Zero Skid）、性能开销更小的Core事件采样。	Per-HT	✅	[3][4]
Uncore	Data Fabric (DF)	采集处理器、内存控制器、NBIO等组件之间数据流量的性能指标，例如DMA带宽、内存带宽等。	Per-Socket	❌	N/A
	Last Level Cache (LLC)	采集L3 Cache访问的性能指标，例如L3 Cache Accesses/Misses、L3 Cache Miss Latency等。	Per-CCD	❌	
	Unified Memory Controller (UMC)	采集内存控制器上DRAM Channel事件的性能指标。	Per-Socket	❌	

[1] https://lore.kernel.org/linux-kernel//20230603011058.1038821-1-seanjc@google.com/T/
[2] https://lkml.iu.edu/hypermail/linux/kernel/0803.3/3052.html
[3] https://lore.kernel.org/all/20230908133114.GK19320@noisy.programming.kicks-ass.net/T/
[4] https://github.com/Kullu14/qemu/tree/qemu_vibs_branch

阿里云自研AMD vPMU能力矩阵（1）

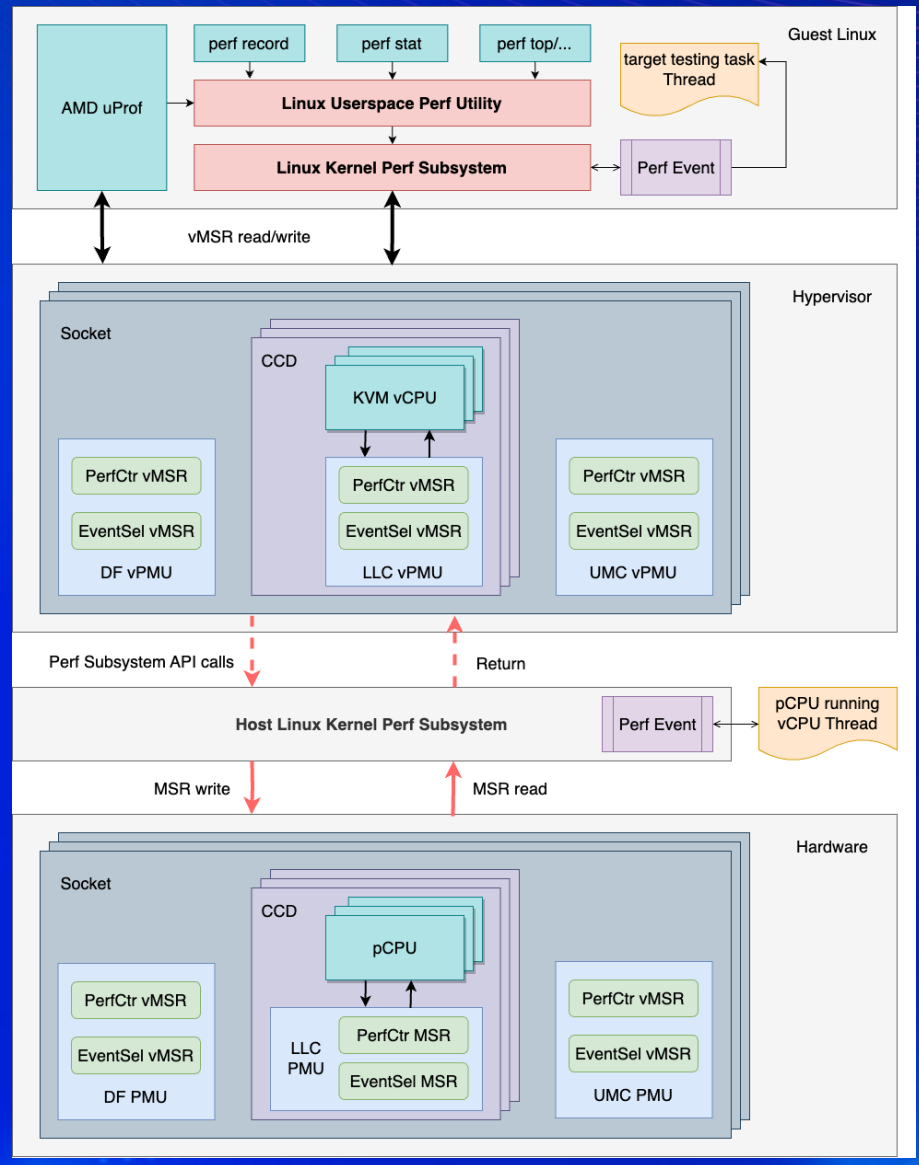
- Arch vPMU
  - 继承upstream QEMU/KVM vPMU全部能力，支持**PerfMonV2**。
  - KVM PMU通用路径优化，提升MSR Emulation的执行效率。
  - PMU MSRs save&&load，支持状态热迁移。
- vLBR / vIBS
  - 基于LbrVirt/IbsVirt硬件辅助虚拟化，实现LBR/IBS硬件直通，降低MSRs的访问开销和上下文切换开销。支持热迁移。
  - 通过AVIC注入IBS vPMI中断，无需KVM干预。
  - vLBR呈现LbrExtV2，支持更多LBR MSRs，Guest LBR精度更高。





阿里云自研AMD vPMU能力矩阵（2）

- Uncore vPMU（DF/LLC/UMC）
  - 无硬件辅助虚拟化，采用模拟方式实现，支持热迁移。
  - 共享资源，仅支持独占pCPU的特定vCPU规格虚拟机。
  - KVM给Guest模拟呈现的PMU MSRs共享视图与物理硬件一致。
  - Guest访问PMU MSRs会trap到KVM，KVM根据PMU type（DF/UMC/LLC）和事件配置参数，调用perf subsystem api创建perf event，并绑定到vCPU独占的pCPU上，实现对目标Socket/CCD上Uncore事件的采集。
  - 无PMI中断，性能损耗<1.5%。
- AMD uProf in Guest
  - 对比Linux perf，功能更强大，使用更简单便捷。
  - Prerequisite: APERF/MPERF/IRPERF MSRs直通。
  - DMA带宽、内存带宽、L3 Cache Hit/Miss Ratio、Miss Latency等。

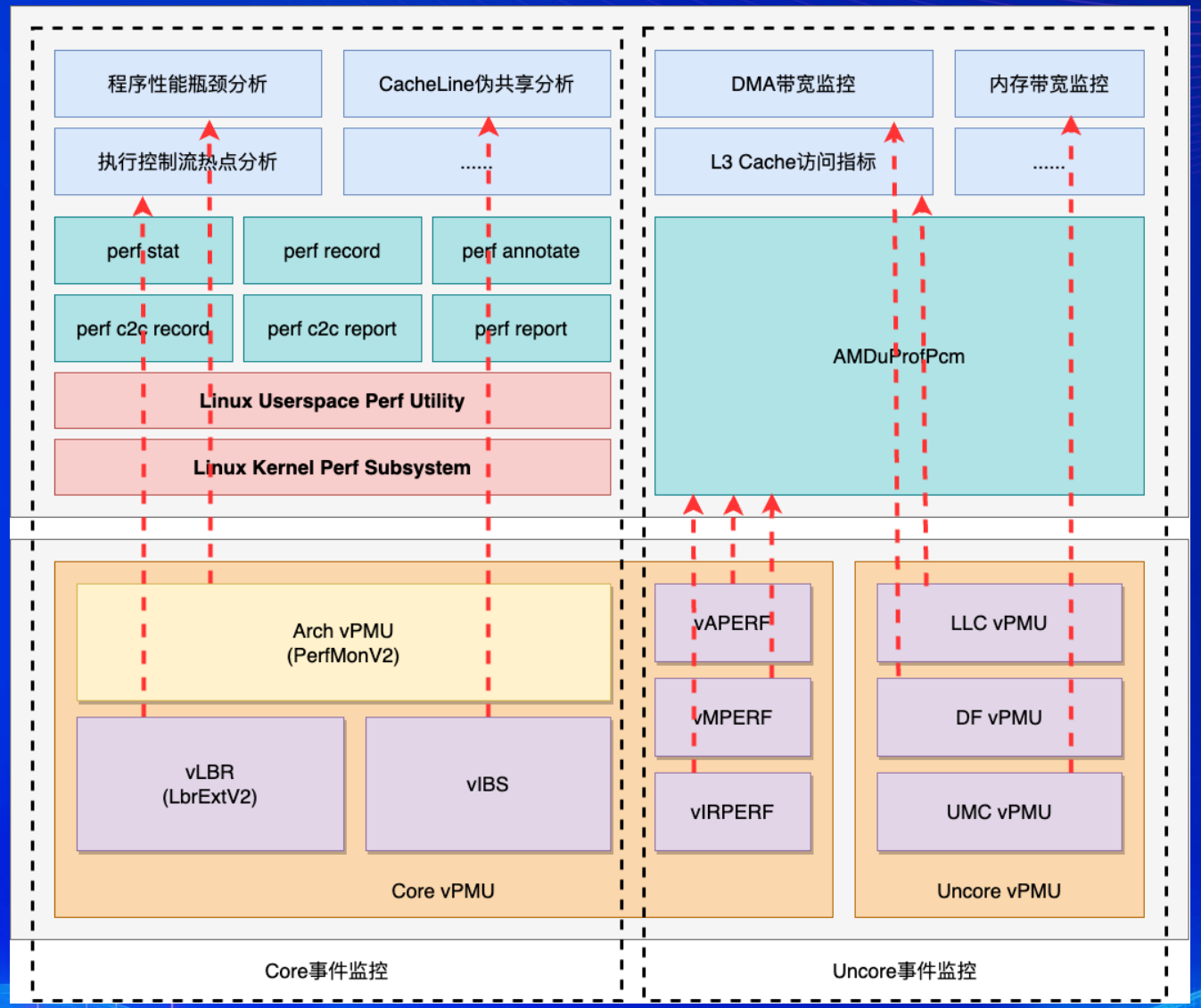


Part 03

AMD vPMU应用实践

AMD vPMU应用实践

- 阿里云通过自研补齐AMD vPMU能力矩阵，支撑了VM带内性能分析监控的各类场景：
  - Arch vPMU => Topdown程序性能瓶颈分析。
  - vLBR => 执行控制流热点分析。
  - vIBS => CacheLine伪共享分析。
  - Uncore vPMU => 借助AMDuProf在Guest高效便捷地监控DMA带宽、内存带宽、L3 Cache访问等性能指标。
  - 其他不限于上述的应用场景，拥有接近物理机PMU功能的使用体验。



AMD Arch vPMU应用实践 —— 程序性能瓶颈分析（1）

- 自顶向下微架构分析方法：Top-down Microarchitecture Analysis Method （TMAM）

```
static int foo(char *arr, int n) {
        int res = 0;
        for (int i = 0; i < 100; i++)
                res += n * i;
        return res + arr[arr[n]];
}

const int _200MB = 1024*1024*200;

int main() {                          1. 申请远大于L3的内存
        char *a = (char *)malloc(_200MB);
        memset(a, 0, _200MB);
        srand(time(NULL));
        for (int i = 0; i < 10000000; i++)
                foo(a, rand() % _200MB);
        free(a);
        return 0;          2. 随机访问该内存，触发大量L3 Misses
}
```

4. 优化程序性能

```
      |    ↓ jmp    29
      |    res += n * i;
15.19 |1b:   mov    -0x1c(%rbp),%eax
 0.15 |      imul   -0x8(%rbp),%eax
 1.95 |      add    %eax,-0x4(%rbp)
      |    for (int i = 0; i < 100; i++)
44.41 |      addl   $0x1,-0x8(%rbp)
 1.81 |29:   cmpl   $0x63,-0x8(%rbp)
21.75 |    ↑ jle    1b
      |    res += arr[arr[n]];
 0.01 |      mov    -0x1c(%rbp),%eax
 1.59 |      movslq %eax,%rdx
 0.07 |      mov    -0x18(%rbp),%rax
      |      add    %rdx,%rax          高频访存操作
 0.07 |      movzbl (%rax),%eax
12.64 |      movsbq %al,%rdx
 0.01 |      mov    -0x18(%rbp),%rax
      |      add    %rdx,%rax
```

3. Perf record + perf annotate 找到导致L3 Misses的高频访存操作：

res += arr[arr[n]]

```
[root@iZbp12117z8jvdz55oicw6Z topdown]# perf stat -e l3_cache_accesses,l3_misses
 -C 1 -- taskset -c 1 ./test

 Performance counter stats for 'CPU(s) 1':

         30,734,308      l3_cache_accesses

         22,674,612      l3_misses

                                  L3 Miss Ratio = 22,674,612 / 30,734,308
                                                ~= 73.78%
       2.094748446 seconds time elapsed

[root@iZbp12117z8jvdz55oicw6Z topdown]#
```

1. 使用topdown分析程序，发现性能瓶颈在Backend_Bound.Memory阶段，即访存耗时高。

```
[root@iZbp12117z8jvdz55oicw6Z topdown]# gcc -ggdb ./test.c -o test
[root@iZbp12117z8jvdz55oicw6Z topdown]# AMDuProfPcm -m topdown -c core=1 -o /tmp/td.csv -- taskset -c 1 ./test
[root@iZbp12117z8jvdz55oicw6Z topdown]# column -s, -t /tmp/td.csv | tail -n 2
Total_Dispatch_Slots                    SMT_Disp_contention                 Frontend_Bound  Bad_Speculation  Backend_Bound  Retiring  IPC   Frontend_Bound.Latency
 Frontend_Bound.BW  Bad_Speculation.Mispredicts  Bad_Speculation.Pipeline_Restarts  Backend_Bound.Memory  Backend_Bound.CPU  Retiring.Fastpath  Retiring.Microcode
22068544662.00                          0.11                                5.07                  3.06               72.28              19.47      0.89  3.96
 1.11               1.82                        1.24                                57.63                 14.66              19.32                 0.14
[root@iZbp12117z8jvdz55oicw6Z topdown]#
```

2. Perf分析发现L3 Miss Ratio高达约73.78%

AMD vLBR应用实践 —— 程序性能瓶颈分析（2）

- LBR：用于跟踪程序的**执行控制流**，根据调用栈记录以分析程序热点路径。

```
int func_3(int i) {
        volatile int result = 0;
        for (int j = 0; j < 100; ++j)
        result += (i - j);
        return result;
}

int func_2(int i) {
        volatile int result = 0;
        for (int j = 0; j < 100; ++j)
        result += (i - j);
        return result;
}

int func_1(int i) {
        volatile int result = 0;
        for (int j = 0; j < 100; ++j)
        result += (i - j);
        return result;
}
                func_1 : func_2 : func_3 ~= 1 : 3 : 6
int main() {
        volatile int result = 0;

    // func_1 : func_2 : func_3 ~= 1 : 3 : 6
        for (int i = 0; i < 100000000; ++i) {
        int tmp = i % 10;

                if (tmp < 1)
                        result += func_1(i);
        else if (tmp < 4)
            result += func_2(i);
        else
                result += func_3(i);
        }

        return !result;
}
```

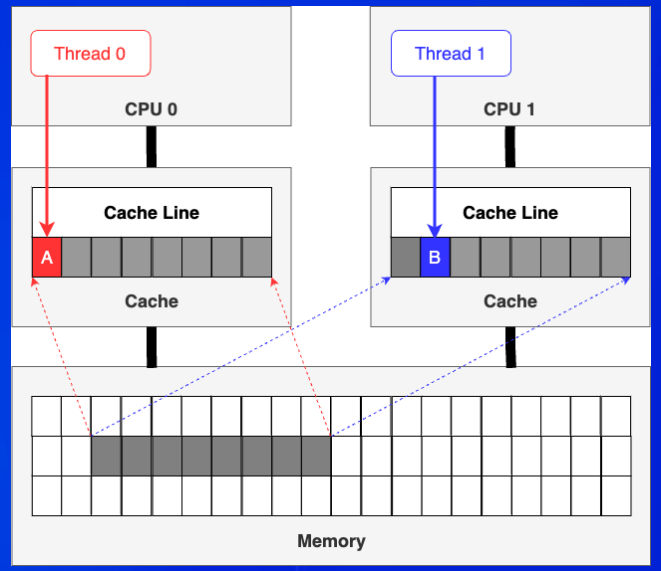
```
perf record -b -e cycles:u -C 1 -- taskset -c 1 ./lbr-test
perf report --sort symbol_from,symbol_to
```

```
Samples: 281K of event 'cycles:u', Event count (approx.): 281041
Overhead  Source Symbol                 Target Symbol                 IPC   [IPC Coverage]
 56.16%  [.] func_3                     [.] func_3                    0.00  [  0.0%]
 29.75%  [.] func_2                     [.] func_2                    0.00  [  0.0%]
  8.72%  [.] func_1                     [.] func_1                    0.00  [  0.0%]
  3.14%  [.] main                       [.] main                      0.00  [  0.0%]
  0.68%  [.] func_3                     [.] main                      0.00  [  0.0%]
  0.52%  [.] main                       [.] func_3                    0.00  [  0.0%]
  0.47%  [.] func_2                     [.] main                      0.00  [  0.0%]
  0.37%  [.] main                       [.] func_2                    0.00  [  0.0%]
  0.10%  [.] main                       [.] func_1                    0.00  [  0.0%]
  0.07%  [.] func_1                     [.] main                      0.00  [  0.0%]
  0.01%  [.] _dl_map_object_deps        [.] _dl_map_object_deps       0.00  [  0.0%]
  0.00%  [.] perf_evsel__enable_cpu     [.] perf_evsel__enable_cpu    0.00  [  0.0%]
  0.00%  [.] _dl_map_object_deps        [.] memmove                   0.00  [  0.0%]
  0.00%  [.] memmove                    [.] _dl_map_object_deps       0.00  [  0.0%]
  0.00%  [.] __GI___ioctl               [.] perf_evsel__enable_cpu    0.00  [  0.0%]
  0.00%  [.] entry_SYSRETQ_unsafe_stack [.] __GI___ioctl              0.00  [  0.0%]
  0.00%  [.] _dl_map_object_deps        [.] rtld_malloc               0.00  [  0.0%]
  0.00%  [.] rtld_malloc                [.] _dl_map_object_deps       0.00  [  0.0%]
  0.00%  [.] perf_evsel__enable_cpu     [.] evlist__enable            0.00  [  0.0%]
  0.00%  [.] native_irq_return_iret     [.] perf_evsel__enable_cpu    0.00  [  0.0%]
  0.00%  [.] _dl_map_object_deps        [.] _dl_sort_maps             0.00  [  0.0%]
  0.00%  [.] _dl_sort_maps              [.] _dl_sort_maps             0.00  [  0.0%]
  0.00%  [.] _dl_sort_maps              [.] memset                    0.00  [  0.0%]
  0.00%  [.] evlist__enable             [.] evlist__enable            0.00  [  0.0%]
  0.00%  [.] native_irq_return_iret     [.] _dl_map_object_deps       0.00  [  0.0%]
  0.00%  [.] native_irq_return_iret     [.] rtld_malloc               0.00  [  0.0%]
  0.00%  [.] native_irq_return_iret     [.] evlist__enable            0.00  [  0.0%]
```

热点控制流路路径：func_3、func_2、func_1，
执行频率比率为：56.16:29.75:8.72 ~= 6.39 : 3.4 : 1

AMD vIBS应用实践 —— CacheLine伪共享热点分析（1）

- perf c2c是AMD IBS的一项应用，可用于收集CPU Cache-to-Cache的性能指标信息。
- CacheLine伪共享（CacheLine False Sharing）问题是指多个CPU上的线程同时修改存储在同一Cache Line中不同位置的变量，由于Cache一致性协议，引发频繁的Cache Miss和无效化操作，从而严重降低系统性能。



- 借助perf c2c，能够有效发现**CacheLine伪共享热点代码路径**，并进行性能优化。

AMD vIBS应用实践 —— CacheLine伪共享热点分析（2）

- 测试程序：https://github.com/joemario/perf-c2c-usage-files/blob/master/false_sharing_example.c

```
[root@iZ2ze7njn1asq2wantzlg8Z cacheline_false_sharing]# time -p ./false_sharing > /tmp/false_sharing.log 16
libnuma: Warning: Cannot read node cpumask from sysfs
real 5.42
user 84.95
sys 0.00
[root@iZ2ze7njn1asq2wantzlg8Z cacheline_false_sharing]# time -p ./no_false_sharing > /tmp/no_false_sharing.log 16
libnuma: Warning: Cannot read node cpumask from sysfs
real 4.73
user 74.49
sys 0.00
[root@iZ2ze7njn1asq2wantzlg8Z cacheline_false_sharing]#
```

1. 产生与不产生CacheLine伪共享的情况下，二者的执行效率相差1.7s左右，即约36%。

```
[root@iZ2ze7njn1asq2wantzlg8Z cacheline_false_sharing]# ./run_test
libnuma: Warning: Cannot read node cpumask from sysfs
[ perf record: Woken up 699 times to write data ]
Warning:
Processed 3853526 events and lost 9 chunks!

Check IO/CPU overload!

[ perf record: Captured and wrote 322.590 MB perf.data (3807782 samples) ]
[root@iZ2ze7njn1asq2wantzlg8Z cacheline_false_sharing]#
```

2. perf c2c收集目标程序的性能数据样本：

```
perf c2c record -F 60000 -a
```

4. 进一步分析可知CacheLine伪共享是test.c#read_write_func方法中的代码导致，可以进行针对性优化。

```
160  for(j=0; j<LOOP_CNT; j++) {
161
162    // Check for lock thread.
163    if (*thd_name == *lock_thd_name) {
164      __sync_lock_test_and_set(&buf1.lock0, 1 );
165      buf1.lock0 += 1;
166      buf2.lock1 = 1;
167
168    } else {
169      // Reader threads.
170
171      switch(tix % max_node_num) {
172        volatile long var;
173        case 0:
174          var = *(volatile uint64_t *)&buf1.reader1;
175          var = *(volatile uint64_t *)&buf2.reader1;
176          break;
177        case 1:
178          var = *(volatile uint64_t *)&buf1.reader2;
179          var = *(volatile uint64_t *)&buf2.reader2;
180          break;
181        case 2:
182          var = *(volatile uint64_t *)&buf1.reader3;
183          var = *(volatile uint64_t *)&buf2.reader3;
184          break;
185        case 3:
186          var = *(volatile uint64_t *)&buf1.reader4;
187          var = *(volatile uint64_t *)&buf2.reader4;
188          break;
189      };
190    };
191  } // End of for LOOP_CNT loop
```

3. perf c2c分析性能数据样本：

```
perf c2c report -NN -c pid,iaddr -full-symbols
```

Hitm(Hit In the Modified)代表CPU load操作命中了Modified状态的Cache Line，Hitm指标明显过高，说明存在严重的CacheLine伪共享现象。

```
Shared Data Cache Line Table     (2 entries, sorted on Total HITMs)
                           ----------- Cacheline ----------      Tot  ------- Load Hitm -------    Total    Total    Total  --------- Stores --------  ----- Core Load Hit -----  - LLC Load Hit --  - RMT Load Hit --  --- Load Dram ----
Index                            Address  Node  PA cnt     Hitm    Total  LclHitm  RmtHitm  records    Loads   Stores     L1Hit   L1Miss      N/A       FB       L1       L2   LclHit  LclHitm   RmtHit  RmtHitm      Lcl      Rmt
    0                          0x404100     0   30577    98.43%   135904   135904        0   797893   781128   512784     12077        0     4688        0    91205   360956   193063   135904        0        0        0        0
    1                          0x404140     0   10446     1.48%     2047     2047        0    26244    20795     9282        13        0     5436        0     8620     4249     5879     2047        0        0        0        0

Cacheline 0x404100
----- HITM -----  ------- Store Refs ------  ------- CL --------                                                 ---------- cycles ----------    Total    cpu                                  Shared
RmtHitm  LclHitm   L1 Hit  L1 Miss     N/A    Off  Node  PA cnt     Pid        Code address  rmt hitm  lcl hitm     load  records    cnt             Symbol          Object  Source:Line  Node{cpu list}
  0.00%   98.93%    0.00%    0.00%    0.00%   0x0     0       1    2149           0x4012de         0       698      398   765158     16  [.] read_write_func  false_sharing  test.c:165  0{0-15}
  0.00%    0.20%    0.00%    0.00%    0.00%   0x0     0       1    2149           0x4012d7         0      2139     1751      397     16  [.] read_write_func  false_sharing  test.c:164  0{0-15}
  0.00%    0.00%  100.00%    0.00%  100.00%   0x0     0       1    2149           0x4012e9         0         0        0    16765     16  [.] read_write_func  false_sharing  test.c:165  0{0-15}
  0.00%    0.86%    0.00%    0.00%    0.00%  0x20     0       1    2149           0x401332         0       382       86    15573     16  [.] read_write_func  false_sharing  test.c:174  0{0-15}
```

AMD Uncore vPMU应用实践 —— AMD uProf性能监控

- 实时监测线上业务运行时性能指标：**DMA带宽/内存带宽/LLC指标**。

1.DMA带宽：AMDuProfPcm top -m dma

DF METRICS

Metric	System	Package-0
Total Upstream DMA Re	0.05	0.05
Local Upstream DMA Re	0.04	0.04
Local Upstream DMA Wr	0.00	0.00
Remote Upstream DMA R	0.00	0.00
Remote Upstream DMA W	0.00	0.00

2.内存带宽：AMDuProfPcm top -m memory

DF METRICS

Metric	System	Package-0
Total Memory Bw (GB/s	2.33	2.33
Local DRAM Read Data	2.18	2.18
Local DRAM Write Data	0.15	0.15
Remote DRAM Read Data	0.00	0.00
Remote DRAM Write Dat	0.00	0.00
Mem Ch-A RdBw (GB/s)	-	0.19
Mem Ch-A WrBw (GB/s)	-	0.01
Mem Ch-B RdBw (GB/s)	-	0.18
Mem Ch-B WrBw (GB/s)	-	0.01
Mem Ch-C RdBw (GB/s)	-	0.19
Mem Ch-C WrBw (GB/s)	-	0.01
Mem Ch-D RdBw (GB/s)	-	0.19
Mem Ch-D WrBw (GB/s)	-	0.02
Mem Ch-E RdBw (GB/s)	-	0.23
Mem Ch-E WrBw (GB/s)	-	0.01
Mem Ch-F RdBw (GB/s)	-	0.22
Mem Ch-F WrBw (GB/s)	-	0.02
Mem Ch-G RdBw (GB/s)	-	0.16
Mem Ch-G WrBw (GB/s)	-	0.01
Mem Ch-H RdBw (GB/s)	-	0.17
Mem Ch-H WrBw (GB/s)	-	0.02
Mem Ch-I RdBw (GB/s)	-	0.16
Mem Ch-I WrBw (GB/s)	-	0.01
Mem Ch-J RdBw (GB/s)	-	0.16
Mem Ch-J WrBw (GB/s)	-	0.01
Mem Ch-K RdBw (GB/s)	-	0.16
Mem Ch-K WrBw (GB/s)	-	0.01
Mem Ch-L RdBw (GB/s)	-	0.16
Mem Ch-L WrBw (GB/s)	-	0.01

3.L3 Cache指标：AMDuProfPcm top -m l3

L3 METRICS

Metric	System	Package-0
L3 Access	2766230.00	2766230.00
L3 Miss	884105.00	884105.00
L3 Miss %	31.96	31.96
Ave L3 Miss Latency	444.49	444.49

Part 04

总结与未来演进

总结与未来演进

- vPMU新特性规划：
  - 支持更多的LBR/IBS增强子特性。
  - 基于AMD PmcVirt硬件辅助虚拟化实现Arch PMU直通，提升精度&&降低开销。
  - 补齐Intel vPMU能力矩阵，包括Arch vLBR、Uncore vPMU[1]。
- 持续提升PMC精度，优化Arch vPMU采样的性能开销。
- Upstreaming

Q&A

[1] https://www.spinics.net/lists/kernel/msg5374461.html

CHINA LINUX KERNEL
中国Linux内核开发者大会

華中科技大學
网络空间安全学院
School of Cyber Science and Engineering, HUST

第19届中国Linux内核开发者大会

THANKS

華中科技大學