



RISC-V Perf Tool Status

Alan Kao
Andes Technology

2019 RISC-V Workshop Taiwan

Outline

- **Big picture: perf record**
- **RISC-V perf**
 - Current status, and
 - ◆ Its weaknesses
 - Solutions in AndeStar™ V5 series
- **Future work**

Sampling a Program Using Perf

```
$ perf record \
  -c 1000000 \
  # every 1000000 times \
  -e cycles:u \
  # #cycles in userspace \
  ls -R /usr && \
  # the target program \
perf report
```

```
Samples: 969 of event 'cycles:u', Event count (approx.): 96
Overhead Command Shared Object Symbol
11.46% ls ls [.] 0x000000000000090bd
5.99% ls ls [.] 0x000000000000090b2
5.47% ls libc-2.28.so [.] __strlen_avx2
3.30% ls libc-2.28.so [.] __strcmp_avx2
2.89% ls libc-2.28.so [.] _IO_file_xsputn@@GL
2.79% ls ls [.] 0x000000000000679b
2.58% ls ls [.] 0x0000000000006790
2.06% ls ls [.] 0x0000000000008da9
1.86% ls libc-2.28.so [.] _int_malloc
1.75% ls libc-2.28.so [.] __strcoll_l
1.55% ls libc-2.28.so [.] malloc
1.55% ls libc-2.28.so [.] malloc_consolidate
1.55% ls ls [.] 0x00000000000090d5
1.44% ls ls [.] 0x00000000000090f5
1.14% ls libc-2.28.so [.] __memmove_avx_unali
1.14% ls libc-2.28.so [.] _int_free
1.03% ls ls [.] 0x000000000001286f
0.83% ls libc-2.28.so [.] _IO_file_write@@GLT
```

Life Cycle of Sampling

Initialization:

```
$pmc = MAX-1000000
```

```
perf_irq: // when $pmc overflows  
stop $pmc  
sampling/recording  
$pmc = MAX-1000000  
start $pmc
```

```
Main loop: // pure HW things  
while running  
    increase $pmc by  
        #cycles in user space  
endwhile
```

Interrupt !!!



Current State of RISC-V Perf



■ RISC-V base PMU (Performance Monitoring Unit)

- Upstream since April 2018
 - Which should be able to run on all RISC-V systems
- Supports common counters (cycle & instret) only
 - Counting (using perf stat) is OK, but
 - Sampling is not possible due to following reasons

■ Weaknesses

- No writable counters in S-mode
- Always-free-running counters
- No mode selections
- No counter-triggered interrupts

No Writable Counters in S-mode

Initialization:

```
$pmc = MAX-1000000
```

```
perf_irq: // when $pmc overflows  
stop $pmc  
sampling/recording  
$pmc = MAX-1000000  
start $pmc
```

```
Main loop: // pure HW things  
while running  
    increase $pmc by  
        #cycles in user space  
endwhile
```

Interrupt !!!

Always-free-running Counters

Initialization:

```
$pmc = MAX-1000000
```

```
perf_irq: // when $pmc overflows  
stop $pmc  
sampling/recording  
$pmc = MAX-1000000-$pmc  
start $pmc
```

```
Main loop: // pure HW things  
while running  
    increase $pmc by  
        #cycles in user space  
endwhile
```

Interrupt !!!

No Mode Selections

Initialization:

```
$pmc = MAX-1000000
```

```
perf_irq: // when $pmc overflows  
stop $pmc  
sampling/recording  
$pmc = MAX-1000000-$pmc  
start $pmc
```

```
Main loop: // pure HW things  
while running  
    increase $pmc by  
        #cycles in user space  
endwhile
```

Interrupt !!!

No Counter-triggered Interrupts

Initialization:

```
$pmc = MAX-1000000
```

```
perf_irq: // when $pmc overflows  
stop $pmc  
sampling/recording  
$pmc = MAX-1000000-$pmc  
start $pmc
```

```
Main loop: // pure HW things  
while running  
    increase $pmc by  
        #cycles in user space  
endwhile
```

~~Interrupt !!!~~

No sampling in RISC-V Perf !

Initialization:

```
$pmc = MAX-1000000
```

```
perf_irq: // when $pmc overflows  
stop $pmc  
sampling/recording  
$pmc = MAX-1000000-$pmc  
start $pmc
```

```
Main loop: // pure HW things  
while running  
    increase $pmc by  
        #cycles in user space  
endwhile
```

~~Interrupt !!!~~

The slide features a dark blue background. In the top left corner, there is a small image of an AndeStar V5 chip. In the top right corner, there is a stylized graphic of a human head profile composed of a grid of blue lines. The main title 'AndeStar™ V5 Solutions' is centered at the top in a large, white, sans-serif font. Below the title, the content is organized into three main sections, each starting with a white square bullet point. Each section contains a list of features or updates, with some items using circular and diamond-shaped bullet points for sub-points. The text is white and clearly legible against the dark background.

AndeStar™ V5 Solutions

■ AndeStar™ PMU

- Solves them all

■ Writable counters in S-mode

- Counters are read-only shadows in S-mode
 - ◆ They can be written through SBI calls to M-mode, but
 - ◆ The performance penalty is huge
- We introduce counterwen CSRs
- Let counters be writable in S-mode instead of just shadows

■ Pausing/resuming Counters

- We proposed mcounterinhibit CSR
- Already accepted in the privileged spec since Nov. 2018



AndeStar™ V5 Solutions (Cont.)

■ Mode selections

- We introduce `countermask_[u|k|m]` CSRs
- Mode-aware counting/sampling

■ Interrupts from counter overflows

- Introduce `counterinen` and `counterovf` CSRs

■ Now perf sampling just works!

- For RV64-UP targets for now



Future Work



■ Performance Monitoring Working Group

- We are preparing for this!

■ We need more tests

- All configurations
- Working with distributions (e.g. Fedora has perf package)

■ Support more features of perf

- With libraries, other kernel features



Thank you