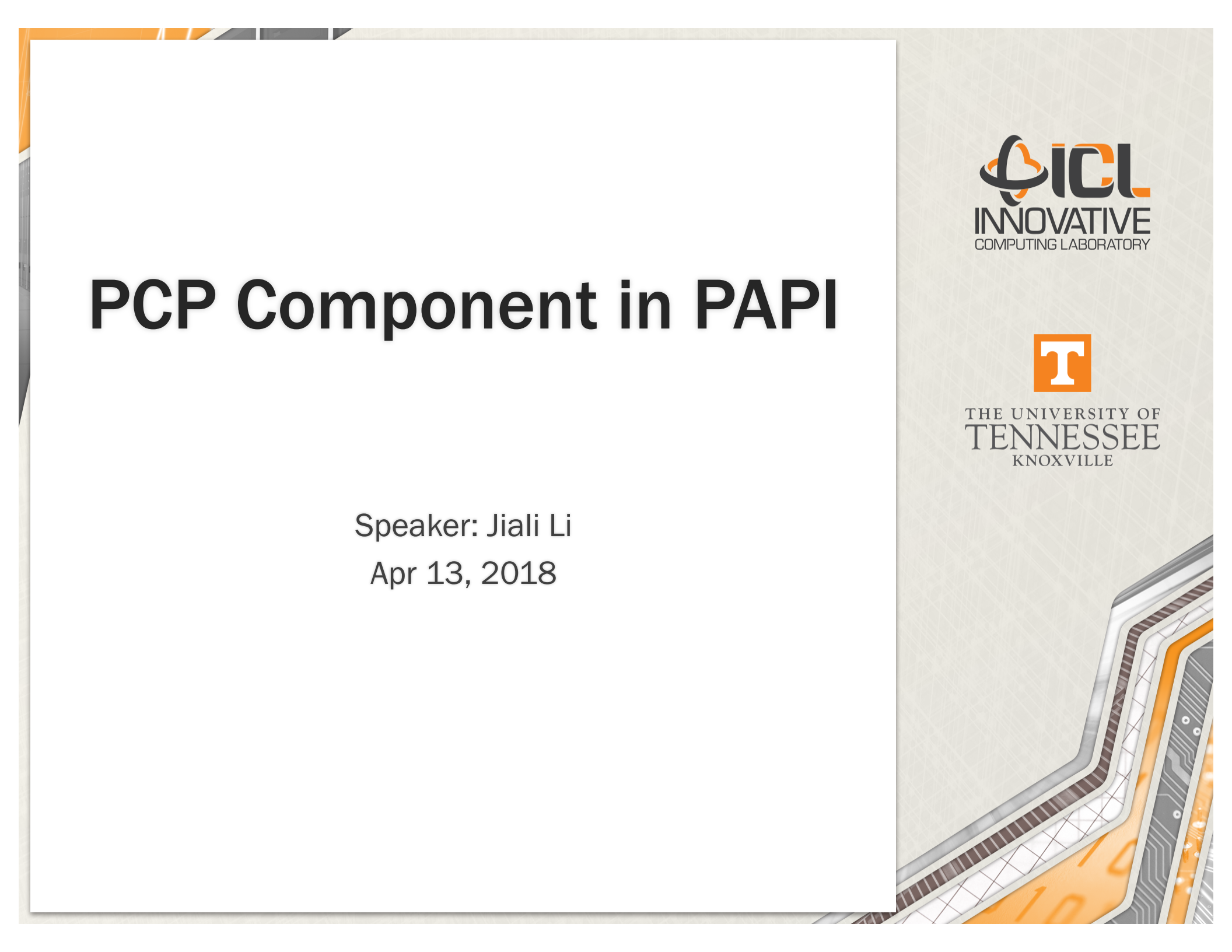
PCP Component in PAPI

Speaker: Jiali Li

Apr 13, 2018



- Content
  - Background and motivation
  - Introduction
  - Methodology
  - Results
  - Future work

Background and motivation

- Modern processors have many processing cores in one socket and have **core** and uncore hardware performance **events.**
  - **Core** events: which are specific for each core
  - **Uncore** events: for resources which are shared between cores on the node.

- Names and terminology for these counters vary by vendor: e.g. Intel calls these shared events "uncore" events, AMD calls them "Northbridge" events, and IBM started calling them "NEST" events for the Power series.

- Access to uncore events needs elevated privileges. IBM's official route to provide (higher privilege) access to NEST events will be through the Performance Co-Pilot (PCP) for non-root users. In a nutshell, that's why we are developing a PCP component for PAPI.

Background and motivation

- Modern processors have many processing cores in one socket and have **core** and uncore hardware performance **events.**
  - **Core** events: which are specific for each core
  - **Uncore** events: for resources which are shared between cores on the node.

- Names and terminology for these counters vary by vendor: e.g. Intel calls these shared events "uncore" events, AMD calls them "Northbridge" events, and IBM started calling them "NEST" events for the Power series.

- Access to uncore events needs elevated privileges. IBM's official route to provide (higher privilege) access to NEST events will be through the Performance Co-Pilot (PCP) for non-root users. In a nutshell, that's why we are developing a PCP component for PAPI.

Background and motivation

- Modern processors have many processing cores in one socket and have **core** and uncore hardware performance **events**.
  - **Core** events: which are specific for each core
  - **Uncore** events: for resources which are shared between cores on the node.

- Names and terminology for these counters vary by vendor: e.g. Intel calls these shared events "uncore" events, AMD calls them "Northbridge" events, and IBM started calling them "NEST" events for the Power series.

- Access to uncore events needs elevated privileges. IBM's official route to provide (higher privilege) access to NEST events will be through the Performance Co-Pilot (PCP) for non-root users. In a nutshell, that's why we are developing a PCP component for PAPI.

- Introduction

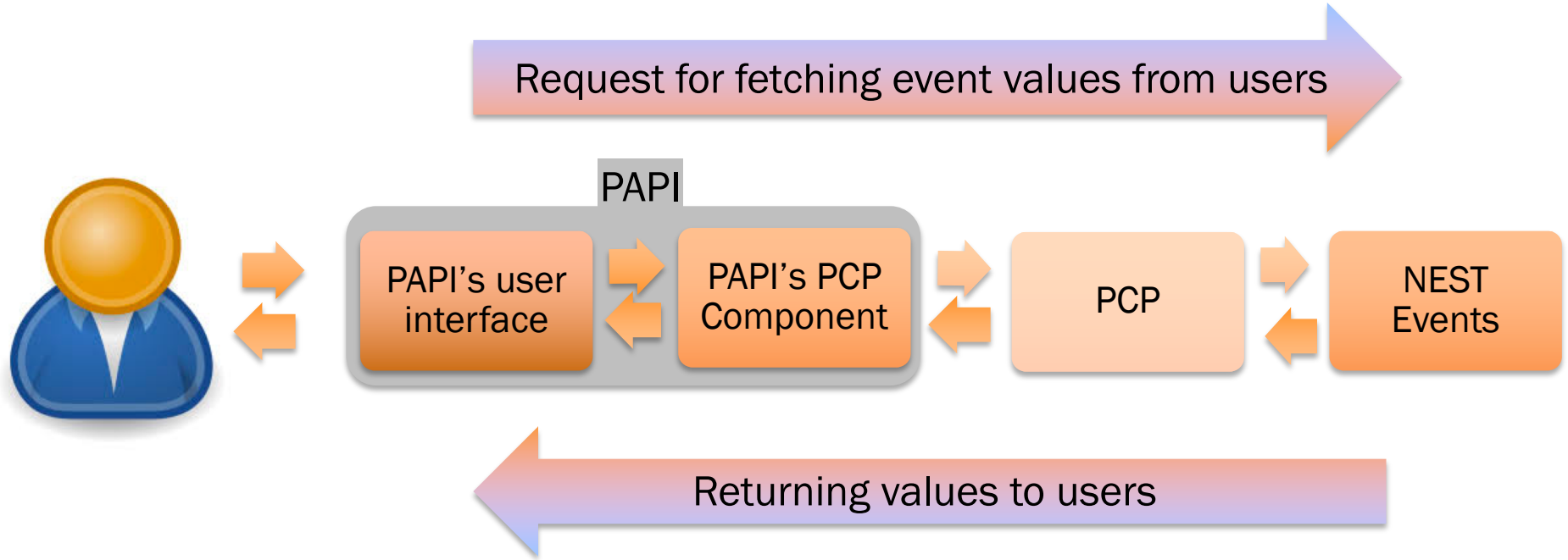
Performance Co-Pilot (PCP)

Performance Co-Pilot (PCP) provides a framework and services to support system-level performance monitoring and management. It presents a unifying abstraction for all of the performance data in a system, and many tools for interrogating, retrieving and processing that data.

Names of Events	Descriptions of Events
PM_FLOP_CMPL	Floating Point Operation Finished
PM_MATH_FLOP_CMPL	Math flop instruction completed
PM_SCALAR_FLOP_CMPL	Scalar flop operation completed
PM_1FLOP_CMPL	one flop (fadd, fmul, fsub, fcmp, fsel, fabs, fnabs, fres, fsqrte, fneg) operation completed
PM_2FLOP_CMPL	DP vector version of fmul, fsub, fcmp, fsel, fabs, fnabs, fres ,fsqrte, fneg
PM_4FLOP_CMPL	4 FLOP instruction completed
PM_8FLOP_CMPL	8 FLOP instruction completed

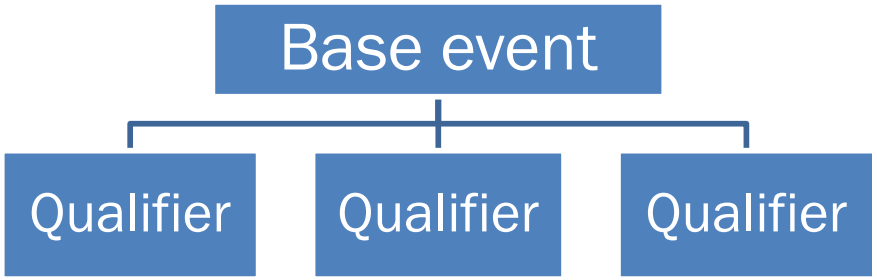
Events that are inquired by users on Power 9 machine, will be exposed from PCP component in PAPI

- Methodology



- All operations are hidden behind PAPI_start()/PAPI_stop()
- Users do not need to learn anything about PCP

- Methodology
  - PCP Data Structure

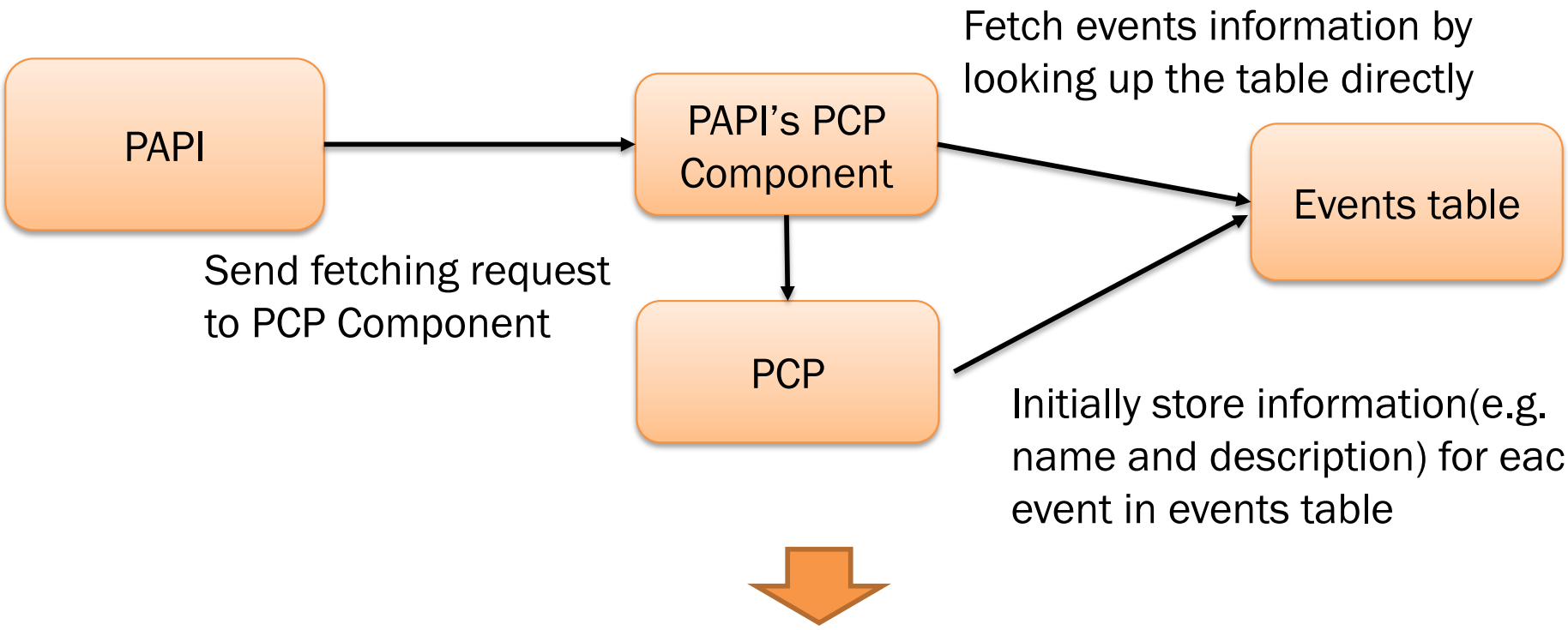



```
xfs.perdev.read_bytes
Help:
This is the number of bytes read via read(2) system calls to files in
XFS file systems. It can be used in conjunction with the read_calls
count to calculate the average size of the read operations to files in
XFS file systems.
jli111@saturn ~> pminfo -f xfs.perdev.read_bytes

xfs.perdev.read_bytes
    inst [0 or "/dev/mapper/sl-root"] value 1420817752763
    inst [1 or "/dev/mapper/sl-var_log"] value 16624557174
    inst [2 or "/dev/sda1"] value 26076327
jli111@saturn ~> pminfo -f xfs.perdev.read_bytes

xfs.perdev.read_bytes
    inst [0 or "/dev/mapper/sl-root"] value 1420831203428
    inst [1 or "/dev/mapper/sl-var_log"] value 16624557174
    inst [2 or "/dev/sda1"] value 26076327
```

Base events are not real events: they are groups of qualifiers which contain real values

PCP doesn't allow to monitor one specific qualifier

- **Methodology**
  - Naïve Method : treat each qualifier as an individual event



Many base events contain 64 (on Saturn) or more qualifiers:
total number of events was 9758
High overhead on storing information and comparing string

- Results
  - Naïve Method
    - Took around 37s in PAPI_init(), the performance is not acceptable;
    - Get all events from papi_native_avail including all qualifiers, the total number of events is 9758 (179 of them are perf events)

```
-------------------------------------------------------------------------------
| pcp:::quota.state.project.enforcement:/dev/mapper/sl-root                   |
|            1 indicates quotas enforced, else 0                              |
-------------------------------------------------------------------------------
| pcp:::quota.state.project.enforcement:/dev/sda1                             |
|            1 indicates quotas enforced, else 0                              |
-------------------------------------------------------------------------------
| pcp:::quota.state.project.enforcement:/dev/mapper/sl-var_log                |
|            1 indicates quotas enforced, else 0                              |
-------------------------------------------------------------------------------

Total events reported: 9937
```

- Results
  - Current Method
    - Takes only couples of microseconds in PAPI_init()

```
-------------------------------------------------------------------------------
| pcp:::quota.state.project.enforcement                                       |
|             1 indicates quotas enforced, else 0                             |
|     :/dev/mapper/sl-root                                                    |
|                                                                             |
|     :/dev/sda1                                                              |
|                                                                             |
|     :/dev/mapper/sl-var_log                                                 |
|                                                                             |
-------------------------------------------------------------------------------

Total events reported: 1110
```

Reduced overhead in initialization from 37s ⟶ microseconds

- **Methodology**
  - Current Method : use PAPI qualifier mapping method

Map each qualifiers and fetch events values

Send fetching request to PCP component

Reduce high overhead on storing events and fetching values
Make events list more readable for users

- Results

```
Stopping measurements, took 18.562s, gathering results...


pcp:::hinv.ncpu
Value: 64
pcp:::hinv.nnode
Value: 4
pcp:::hinv.hugepagesize
Value: 2097152
pcp:::kernel.all.cpu.sys
Value: 640609480
pcp:::kernel.all.nprocs
Value: 1954
pcp:::kernel.all.cpu.nice
Value: 3199200
PASSED
jli111@saturn ~/p/s/c/p/tests>
```

```
Stopping measurements, took 0.006s, gathering results...


pcp:::hinv.ncpu
Value: 64
pcp:::hinv.nnode
Value: 4
pcp:::hinv.hugepagesize
Value: 2097152
pcp:::kernel.all.cpu.sys
Value: 641659270
pcp:::kernel.all.nprocs
Value: 1956
pcp:::kernel.all.cpu.nice
Value: 3199200
PASSED
jli111@saturn ~/p/s/c/p/tests>
```

Reduced fetching overhead form 18.562s ----> 0.006s

- # Future work

  - Re-organize data structure of some base events which are not meaningful to users (shown in the picture)
  - Add more descriptions for some events whose descriptions are not readable

```
----------------------------------------------------------------------------
| pcp:::kernel.percpu.interrupts.line90                                    |
|            IR-PCI-MSI-edge em4-5                                         |
|     :cpu[0...63]                                                         |
|                                                                          |
----------------------------------------------------------------------------
| pcp:::kernel.percpu.interrupts.line89                                    |
|            DMAR_MSI-edge dmar1                                           |
|     :cpu[0...63]                                                         |
|                                                                          |
----------------------------------------------------------------------------
| pcp:::kernel.percpu.interrupts.line88                                    |
|            DMAR_MSI-edge dmar0                                           |
|     :cpu[0...63]                                                         |
|                                                                          |
----------------------------------------------------------------------------
```


Thank You!