Labeled RISC-V: A New Perspective on Software-Defined Architecture

Zihao Yu, Jiuyue Ma, Bowen Huang, Xin Jin, Huizhe Wang, Yaoyang Zhou, Zihao Chang, Yan Cao, Sa Wang, Yungang Bao

May 9th, 2017 @ ShangHai

Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS)
Software-Defined Architecture

Agenda

• Why?
• How?
• What Effect?
Tradeoff in Data Center: Util. vs. QoS

- LOW utilization[1,2]: 6%~12%
- Sharing causes interference
- Sacrifice utilization to guarantee QoS


More Hardware Support Needed

Modern challenges in CPU design

- Isolating programs from each other on a shared server is hard
- As an industry, we do it poorly
  - Shared CPU scheduling
  - Shared caches
  - Shared network links
  - Shared disks

- More hardware support needed
- More innovation needed
QoS-unaware Hardware

- Unmanaged Sharing
- No architectural support for QoS

Unable to distinguish shared resources
Software-Defined Architecture

Agenda

• Why?
• How?
• What Effect?
Labeled Networking

- **Fine-grain**: every packet has a label
- **Semantic association**: correlate labels with users’ demand
- **Propagation**: propagate labels in a whole network
- **DiffServ**: process packets differentiately based on labels

MPLS is widely used for VPN and QoS
The Computer as a Network

- Hardware components communicate via internal packets, e.g., PCIe packets, NoC packets, QPI packets
Labeled von Neumann Architecture (LvNA)

4. Software-defined control logic

1. Fine-grained object

2. Semantic association

3. Propagation

Add label registers to req. sources
Semantic association
Propagation

VM\textsubscript{0} -> Core -> Shared Last Level Cache -> I/O Chipset

CPU request

VM\textsubscript{1} -> Core -> Shared Last Level Cache -> I/O Chipset

DMA

VM\textsubscript{n} -> Core -> Shared Last Level Cache -> I/O Chipset
Software-defined label-based control logic

Diagram showing a system with multiple cores (VM0, VM1, ..., VMn) connected to a shared last level cache (CL) through chipset and memory controllers. Each core and peripheral device (disk, NIC) has a DS-id associated with it.
Programmable Architecture for Resourcing-on-Demand

PARD

Leverage LvNA to perform fine-grained control w/o loss of QoS

Ma et. al, Supporting Differentiated Services in Computers via Programmable Architecture for Resourcing-on-Demand (PARD), ASPLOS, 2015
Tables as CL

Cache Controller

Memory Controller
Closed-loop Control

Trigger => Action

e.g. miss_rate > 30%

Trigger Table

<table>
<thead>
<tr>
<th>DS-id</th>
<th>Cond</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS-id1</td>
<td>Cond-1</td>
<td>Action-1</td>
</tr>
<tr>
<td>DS-id1</td>
<td>Cond-2</td>
<td>Action-2</td>
</tr>
<tr>
<td>DS-id2</td>
<td>Cond-3</td>
<td>Action-3</td>
</tr>
</tbody>
</table>

Statistics Table

| DS-id | Stat1 | Stat2 | ...
|-------|-------|-------|-----
| DS-id1 |       |       |     |
| DS-id2 |       |       |     |
| DS-id3 |       |       |     |

Parameter Table

| DS-id | Param1 | Param2 | ...
|-------|--------|--------|-----
| DS-id1 |       |       |     |
| DS-id2 |       |       |     |
| DS-id3 |       |       |     |

+ firmware action script

e.g. adjust way mask

e.g. adjust way mask

Op = > threshold
Platform Resource Manager (PRM)

- Augmented IPMI
- Connect all control logics
- Run linux-based firmware
- Abstract CLs as files
Access Control Logics

**Query Control Logic Info**
- `cat /sys/cpa/cpa0/ident`
- `cat /sys/cpa/cpa0/type`

**Query Parameters**
- `cat /sys/cpa/cpa0/.../parameter/param1`

**Setting Parameters**
- `echo 10 > /sys/cpa/cpa0/.../parameter/param2`
Software-Defined Architecture

Agenda

- Why?
- How?
- What Effect?
Implementation

- Full-system cycle-accurate simulator [Open Sourced *
- FPGA prototype on Xilinx VC709 evaluation board
- MicroBlaze version [Deprecated]
- RISC-V version [Open Sourced +

* http://github.com/fsg-ict/PARD-gem5
+ http://github.com/fsg-ict/labeled-RISC-V
Evaluation - Performance Isolation

• 4 Ldoms: 1 X 429.mcf + 3 X Attacker
• Allocate different LLC capacities
• Perf. degradation: 7%(w/ PARD) vs. 48%(w/o PARD)
LvNA + RISC-V = Labeled RISC-V

- Switching to RISC-V
  - Add labels in the core easier
  - riscv-go to run container
- Goal - establish Labeled RISC-V branch

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf
LvNA + RISC-V = Labeled RISC-V

- Switching to RISC-V
  - Add labels in the core easier
  - riscv-go to run container
- Goal - establish Labeled RISC-V branch

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf
LvNA + RISC-V = Labeled RISC-V

- Switching to RISC-V
  - Add labels in the core easier
  - riscv-go to run container
- Goal - establish Labeled RISC-V branch

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf
LvNA + RISC-V = Labeled RISC-V

- Switching to RISC-V
  - Add labels in the core easier
  - riscv-go to run container
- Goal - establish Labeled RISC-V branch

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf
LvNA + RISC-V = Labeled RISC-V

- Switching to RISC-V
  - Add labels in the core easier
  - riscv-go to run container
- Goal - establish Labeled RISC-V branch

Address mapping

<table>
<thead>
<tr>
<th>DS-id</th>
<th>Base</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0000</td>
<td>0x4000</td>
</tr>
<tr>
<td>2</td>
<td>0x8000</td>
<td>0x8000</td>
</tr>
</tbody>
</table>

Labeled token bucket

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf
LvNA + RISC-V = Labeled RISC-V

- Switching to RISC-V
  - Add labels in the core easier
  - riscv-go to run container
- Goal - establish Labeled RISC-V branch

Address mapping

<table>
<thead>
<tr>
<th>DS-id</th>
<th>Base</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0000</td>
<td>0x4000</td>
</tr>
<tr>
<td>2</td>
<td>0x8000</td>
<td>0x8000</td>
</tr>
</tbody>
</table>
Overheads

• **16 lines of chisel code to add labels into RocketChip**
  – Add dsid member in the Bundle of TileLink2
  – Attach labels at the TileLink2 masters of core tiles
• **< 5% resource overheads for CLs**
  – Much more less with complex cores, e.g. BOOM
Demo 1 - NoHype

- Push the software hypervisor down to LvNA
  - Isolate the resources (address space, device) by CLs
  - Partitioned into several sub-machines
Demo 1 - NoHype
Demo 2 – Memory Bandwidth Control

- Use labeled token buckets to isolate the bandwidth attacker
Hardware - LvNA

- Labels everywhere in the hardware
- Plan to tape out with 40nm TSMC!
• Push the software hypervisor down to LvNA
  • Remove run-time overhead
    • VMentry, VMexit…

Isolated by NoHype

LVNA

VM1 VM2 VM3 software VMM

VM4 ... VMn

Application
Schedule Framework
Compiler
Runtime Library
Operating System
Hypervisor
Hardware
• Add fine-grained label as context resource
  • **Process** (integrated into Cgroup)
    • Process/container-level
    • Thread-level
  • **Address space**
    • Function-level
    • Object-level
• Provide libraries
  • `pthread_create_with_dsid()`
  • `malloc_with_dsid()`
Compiler - collect QoS info. from prog

- Express QoS info. from source files
- Additional compilation results
  - Address space labeling info
  - Extra ELF sections for loader
- Resource requirement
  - QoS desc. file for schedule framework

<table>
<thead>
<tr>
<th>dsid</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x8000</td>
<td>0xffff</td>
</tr>
<tr>
<td>3</td>
<td>0x2000</td>
<td>0x27ff</td>
</tr>
</tbody>
</table>

```
#include <stdio.h>

int main() {
    call sort
    ... 
    \#pragma qos(10s)
    sort();
    SLA = 10s
    working set = 64KB
    ... 
```
Sche. Framework - QoS resource schedule

- Expose QoS resources to schedule frameworks
- Integrate QoS resources into OpenStack [Finished]
A lot to explore!

- **Theory**: How does LvNA impact on RAM, PRAM, LogP models?
- **Hardware/Arch**: How to implement LvNA at CPU pipeline/SMT, memory, storage, networking? How to correlate LvNA and SDN by labels?
- **OS/Hypervisor**: How to correlate labels with VMs, containers, processes, threads? How to abstract programming interfaces for labels?
- **Programing Model and Compilers**: How to express users’ requirements and propagate to the hardware via labels? How to make compilers support labels?
- **Distributed systems**: How to correlate labels with distributed resources? How to manage distributed systems with label mechanisms?
- **Measurement/Audit**: How to leverage labels to gauge and audit resource usages?

- Finished
- On-going
- Have ideas
- Feature work
Summary

- **LvNA**: a model of software-defined architecture
- **PARD**: a proof of concept of LvNA
- **Labeled RISC-V**: an implementation of LvNA
Labeled RISC-V: A New Perspective on Software-Defined Architecture

Thanks

Q & A

Scan the QR code to join the discussion group with WeChat

Valid until 5/16 and will update upon joining group