中国科学院计算技术研究所
Institute Of Computing Technology Chinese Academy Of Sciences

The Case for Labeled Computer Architecture

Yungang Bao

ICT, CAS

2018-9-19

@ ARM Research Summit

A Brief Intro of ICT

- **Institute of Computing Technology (ICT)** founded in 1956
- ICT made many **“The 1st” in China’s** computer history
  - The 1st **digital computer, 103** (1958)
  - The 1st **vector supercomputer, 757** (1983)
  - The 1st **SMP server**, **Dawning-1** (1993)
  - The 1st **general-purpose CPU**, **Loongson** (2001)
  - The 1st **DNN accelerator**, **DianNao** (2014)

Lenovo & Dawning

- ICT founded **Lenovo** in 1984 (**$65B**)
- ICT founded **Dawning** in 1995 (**$4B**)

Dawning-I
China's 1st
SMP server:
1993

Dawning-1000
China's fastest
SC
1995

Dawning-2000
China's 1st
Cluster-based SC
1998

Dawning-3000
Large-scale
cluster-based SC
2001

Dawning-4000
#10 of TOP500
2004

Dawning-5000
China's fastest SC,
#10 of TOP500
2008

Dawning-6000
#2 of TOP500
2010

Recent Startups

- **Cambricon,** focusing on AI accelerators, founded in 2016 (Unicorn, **$2.5B**)

BlueWhale蓝鲸 中科蓝鲸

SeetaTech 中 科 视 拓

龙芯中科 LOONGSON TECHNOLOGY

Labeled Computer Architecture

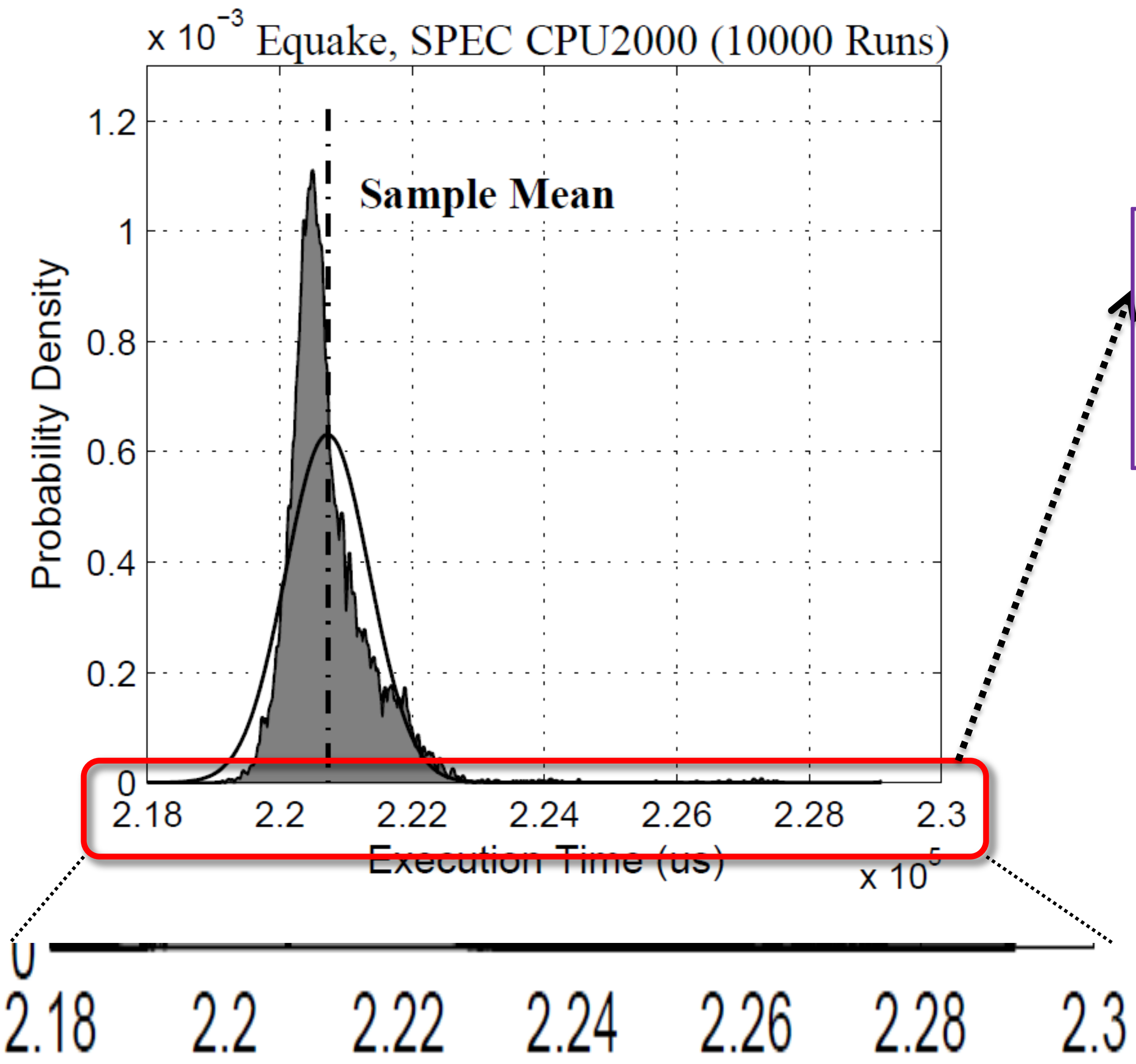
New Interfaces of
Conveying High-level Information to
the Hardware

Virtualization
Quality of Service
Security

……

Performance Variation (1)

- Run a program for **10000 times**



Parameters: avg = 2.16, max=2.3, min=2.18

Variations (%): $\frac{2.3-2.18}{2.16} * 100\% =$ **0.9%**

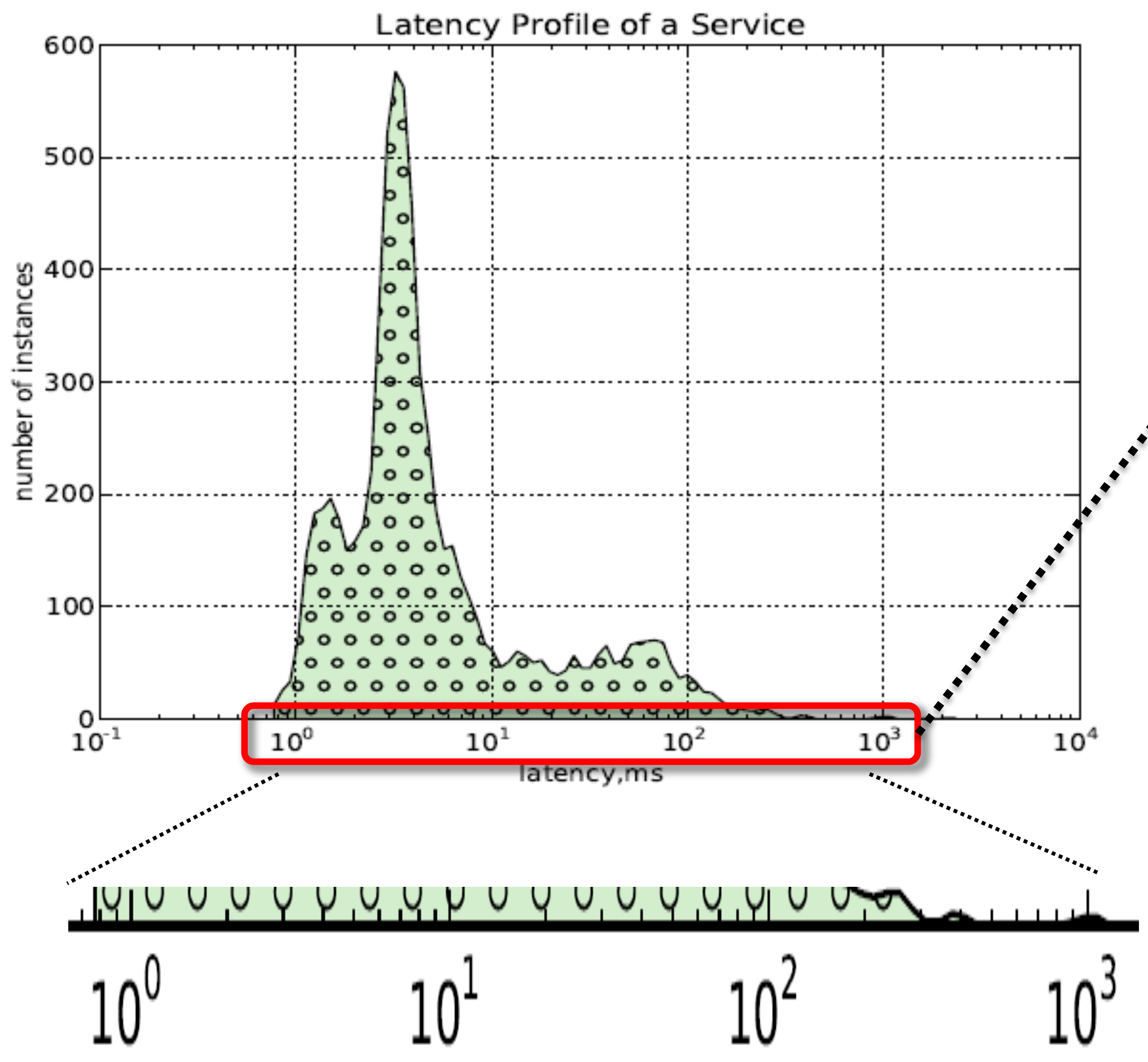
Performance variation on a single server : 1%

T. Chen et al., Statistical Performance Comparisons of Computers, HPCA 2012.

Performance Variation (2)

- Run a service on a Google's datacenter
  - Process **10000 requests**

Dist. of Res. Time



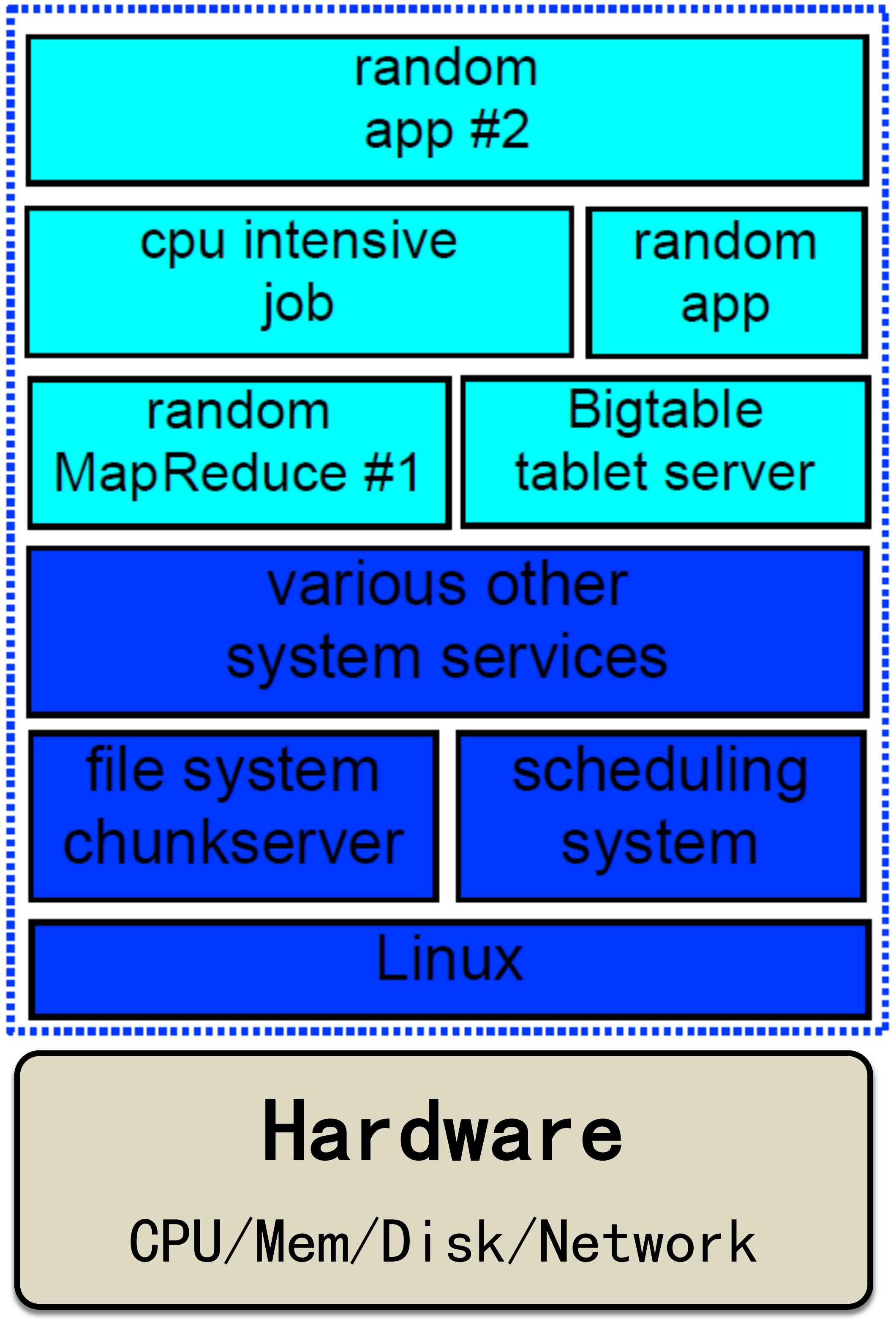
Parameters: avg ≈ 5, max > 300, min < 1

Variations (%): $\frac{300-1}{5} * 100\% = 600\%$

Performance variation in a datacenter: ~10x

D. Krushevskaja and M. Sandler, Understanding Latency Variations of Black Box Services, WWW 2013.

A Typical Google's Server



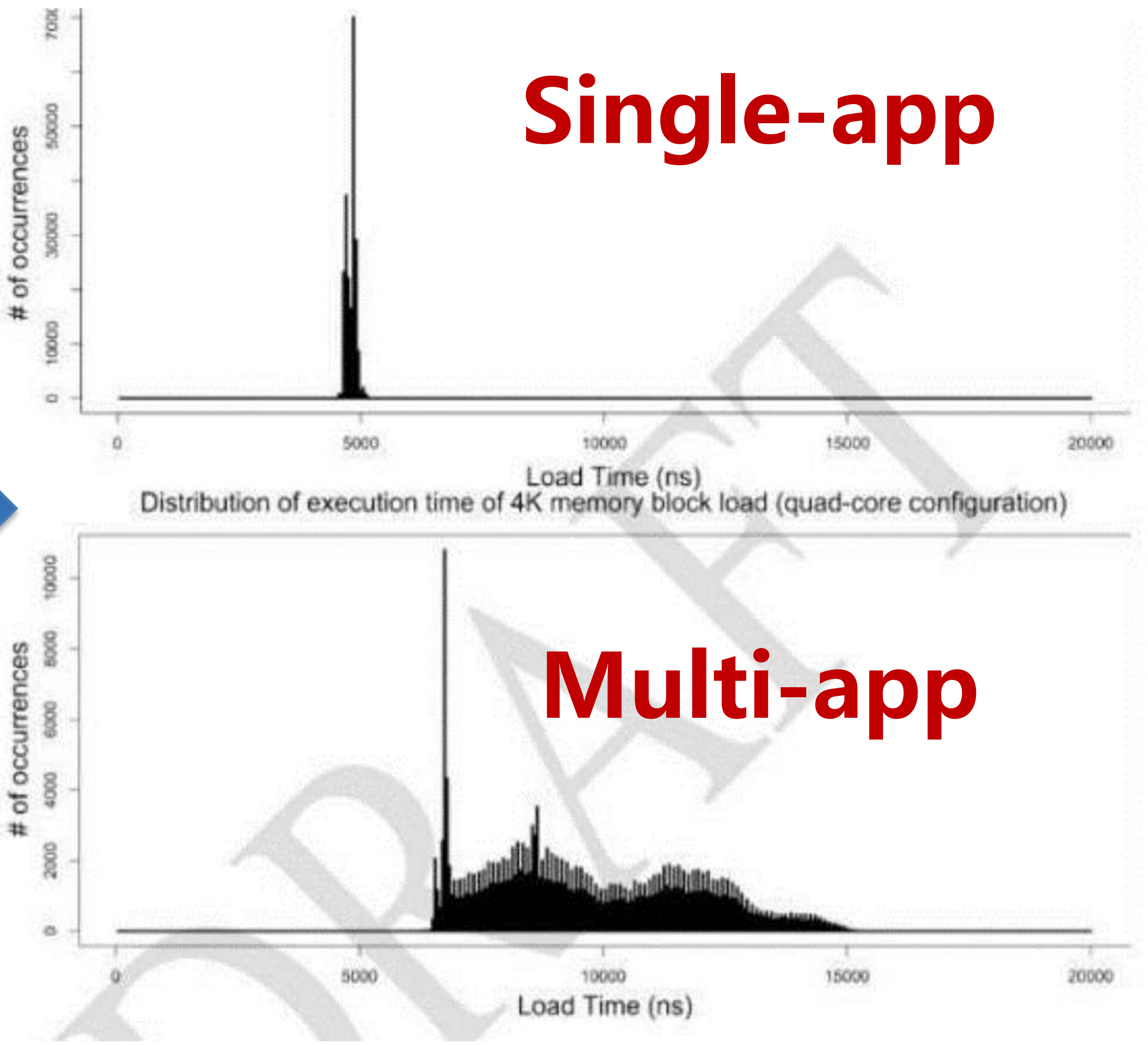
Sharing!

J. Dean, Achieving Rapid Response Times in Large Online Services, talk at Berkeley, 2012.

Unmanaged Sharing

Typical unmanaged sharing

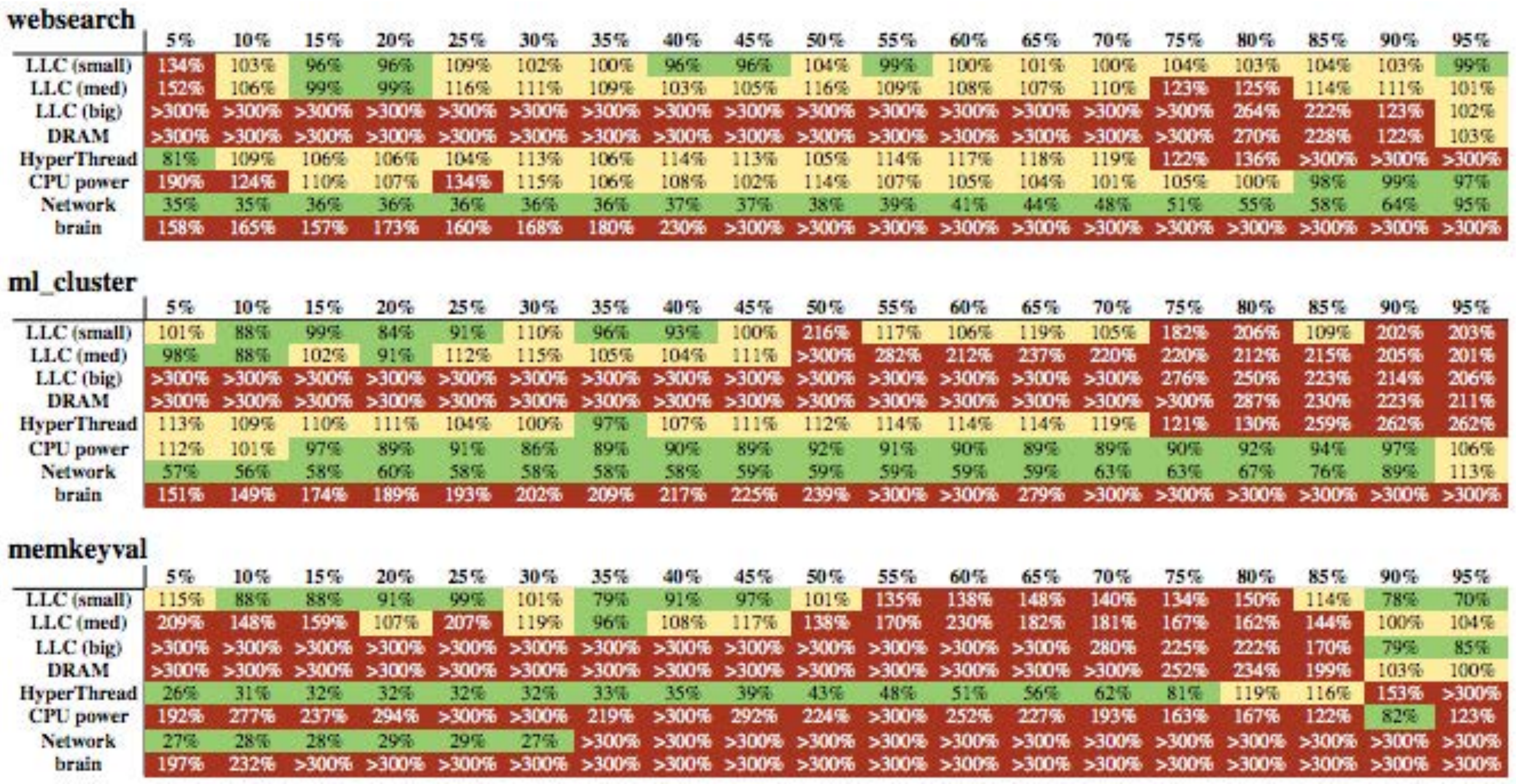
Read 4KB data from DRAM



FAA, White Paper on Issues Associated with Interference Applied to Multicore Processors, 2016

It’s everywhere

SMT、LLC、DRAM、Network etc.



websearch

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
LLC (small)	134%	103%	96%	96%	109%	102%	100%	96%	96%	104%	99%	100%	101%	100%	104%	103%	104%	103%	99%
LLC (med)	152%	106%	99%	99%	116%	111%	109%	103%	105%	116%	109%	108%	107%	110%	123%	125%	114%	111%	101%
LLC (big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	264%	222%	123%	102%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	270%	228%	122%	103%
HyperThread	81%	109%	106%	106%	104%	113%	106%	114%	113%	105%	114%	117%	118%	119%	122%	136%	>300%	>300%	>300%
CPU power	190%	124%	110%	107%	134%	115%	106%	108%	102%	114%	107%	105%	104%	101%	105%	100%	98%	99%	97%
Network	35%	35%	36%	36%	36%	36%	36%	37%	37%	38%	39%	41%	44%	48%	51%	55%	58%	64%	95%
brain	158%	165%	157%	173%	160%	168%	180%	230%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%

ml_cluster

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
LLC (small)	101%	88%	99%	84%	91%	110%	96%	93%	100%	216%	117%	106%	119%	105%	182%	206%	109%	202%	203%
LLC (med)	98%	88%	102%	91%	112%	115%	105%	104%	111%	>300%	282%	212%	237%	220%	220%	212%	215%	205%	201%
LLC (big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	276%	250%	223%	214%	206%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	287%	230%	223%	211%
HyperThread	113%	109%	110%	111%	104%	100%	97%	107%	111%	112%	114%	114%	114%	119%	121%	130%	259%	262%	262%
CPU power	112%	101%	97%	89%	91%	86%	89%	90%	89%	92%	91%	90%	89%	89%	90%	92%	94%	97%	106%
Network	57%	56%	58%	60%	58%	58%	58%	58%	59%	59%	59%	59%	59%	63%	63%	67%	76%	89%	113%
brain	151%	149%	174%	189%	193%	202%	209%	217%	225%	239%	>300%	>300%	279%	>300%	>300%	>300%	>300%	>300%	>300%

memkeyval

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
LLC (small)	115%	88%	88%	91%	99%	101%	79%	91%	97%	101%	135%	138%	148%	140%	134%	150%	114%	78%	70%
LLC (med)	209%	148%	159%	107%	207%	119%	96%	108%	117%	138%	170%	230%	182%	181%	167%	162%	144%	100%	104%
LLC (big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	280%	225%	222%	170%	79%	85%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	252%	234%	199%	103%	100%
HyperThread	26%	31%	32%	32%	32%	32%	33%	35%	39%	43%	48%	51%	56%	62%	81%	119%	116%	153%	>300%
CPU power	192%	277%	237%	294%	>300%	>300%	219%	>300%	292%	224%	>300%	252%	227%	193%	163%	167%	122%	82%	123%
Network	27%	28%	28%	29%	29%	27%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%
brain	197%	232%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%	>300%

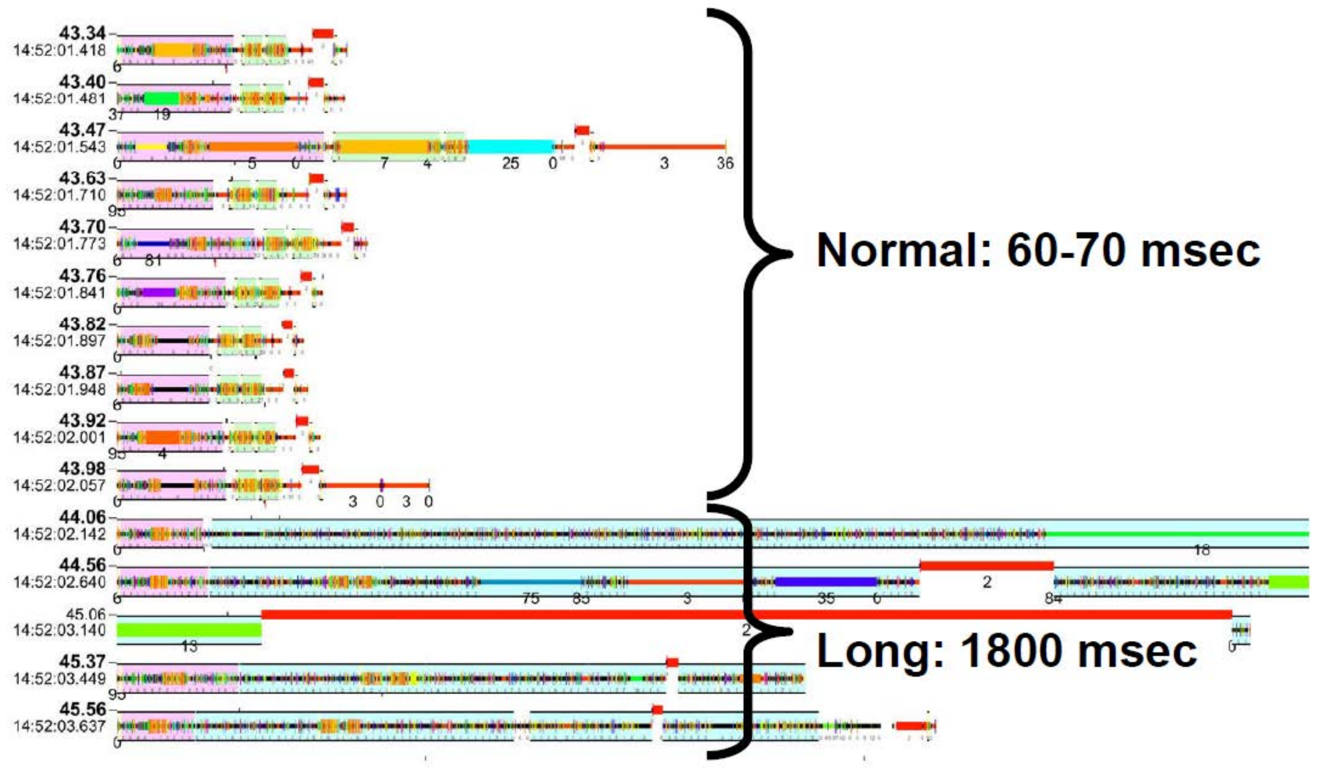
Lo et al. Heracles: Improving Resource Efficiency at Scale, ISCA, 2015.

Google's Efforts in Software Stack

Optimization in Software Stack

- Borg
- Linux Container
- Cgroups
- Backup Requests
- Priority
- Sync-back-tasks
- ……

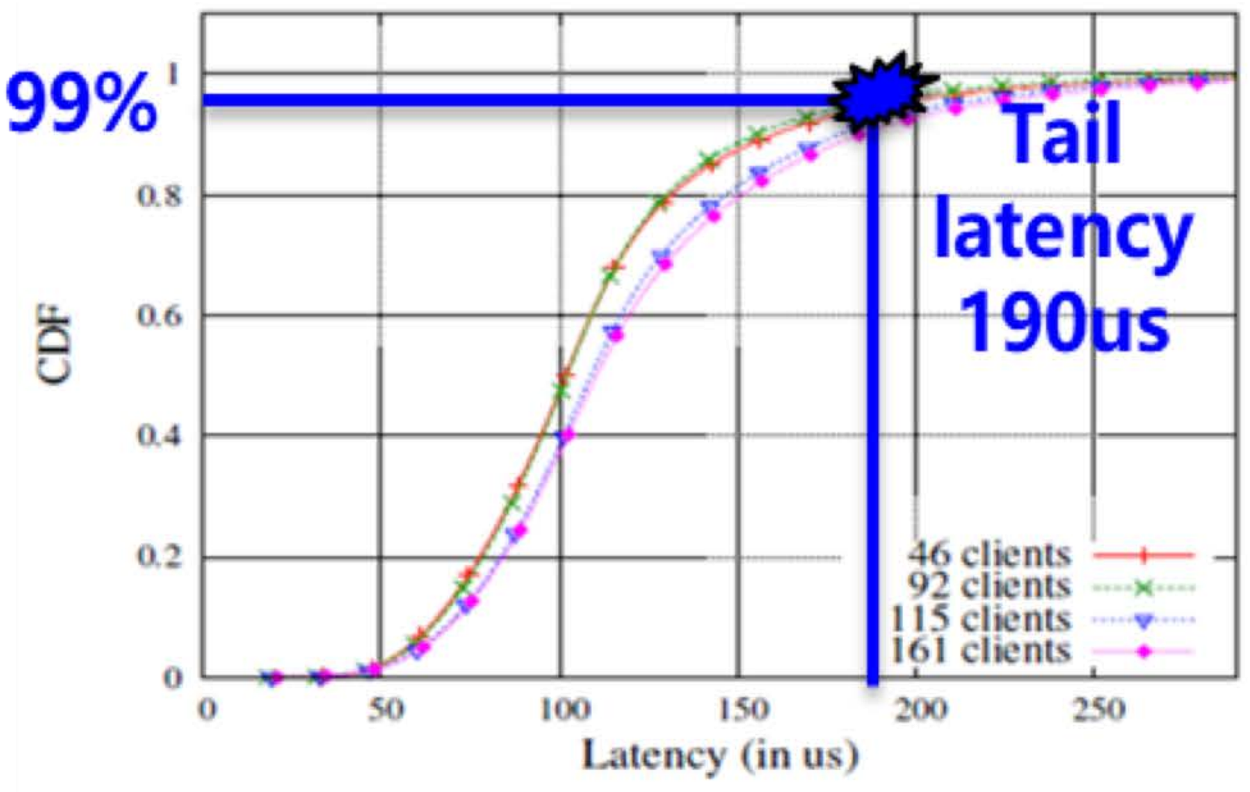
Long Tail Latency



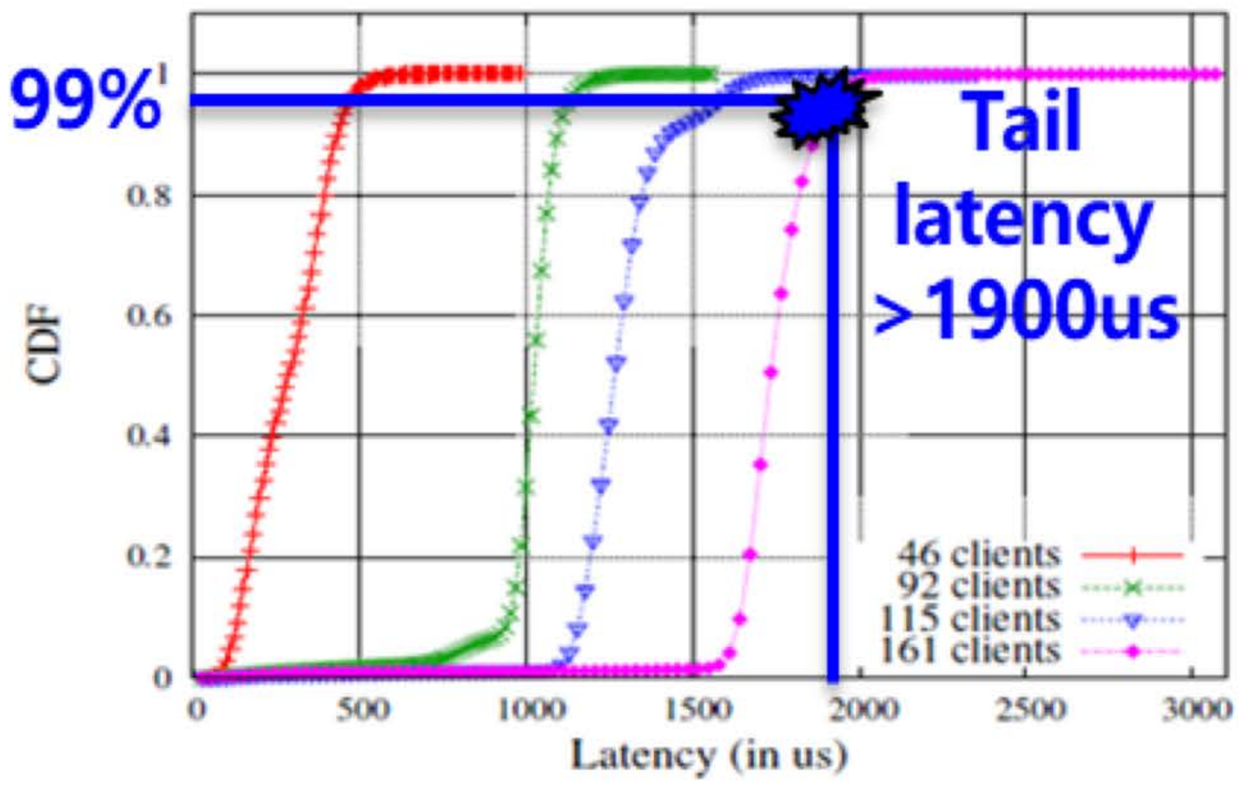
[1] J. Dean, L. Barroso, "The tail at scale", Communication of the ACM, Feb. 2013.
[2] J. Dean, "Achieving Rapid Response Times in Large Online Services", talk at Berkeley, 2012.
[3] Abhishek Verma et al., Large-scale cluster management at Google with Borg, EuroSys, 2015.

Utilization v.s. User Experience

- Utilization: **30%→70%**
- 99th %ile tail latency increases: **10x**



Utilization: 30%

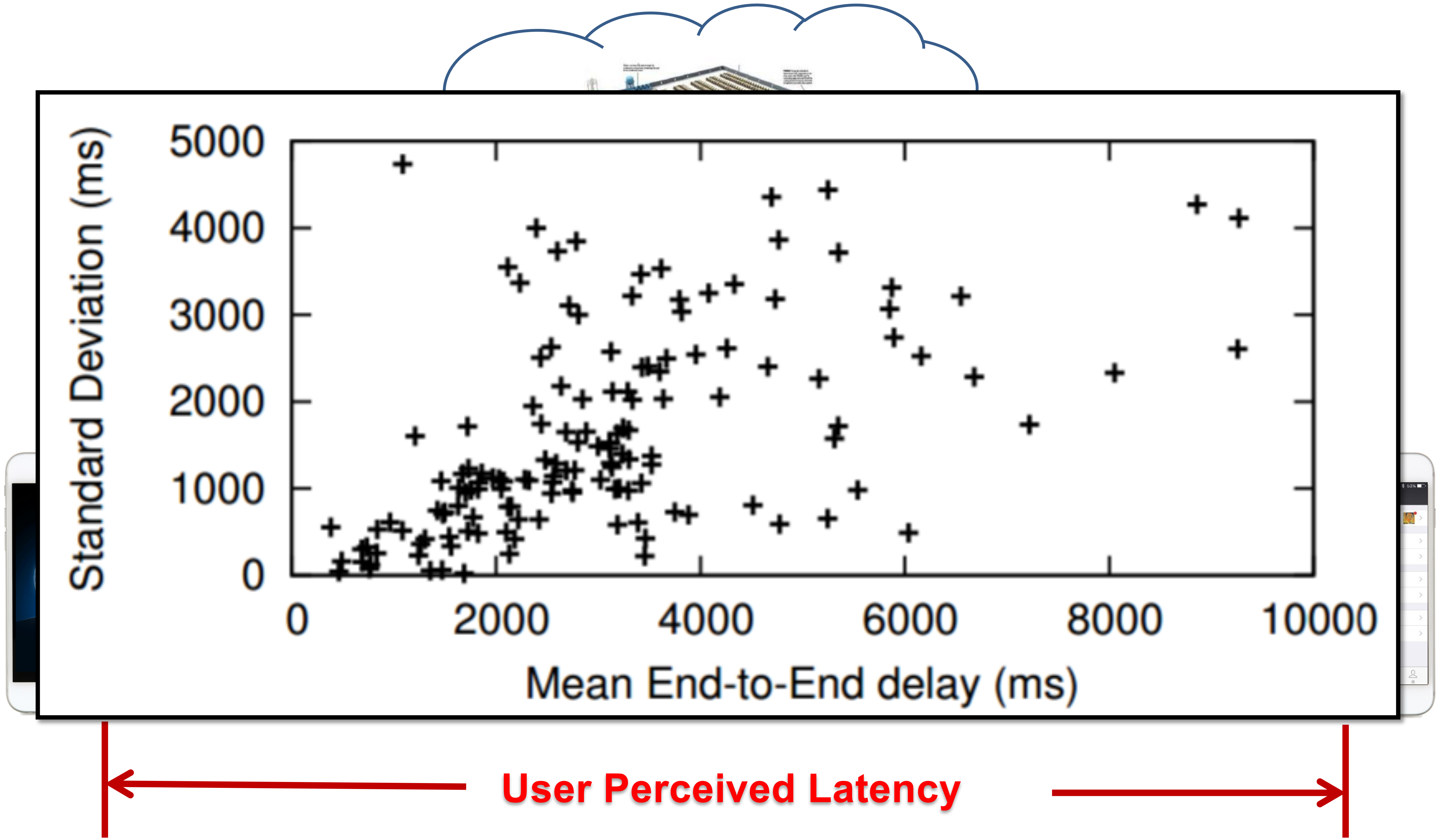


Utilization: 70%

Memcached

R. Kapoor et al. Chronos: Predictable Low Latency for Data Center Applications, SOCC, 2012.

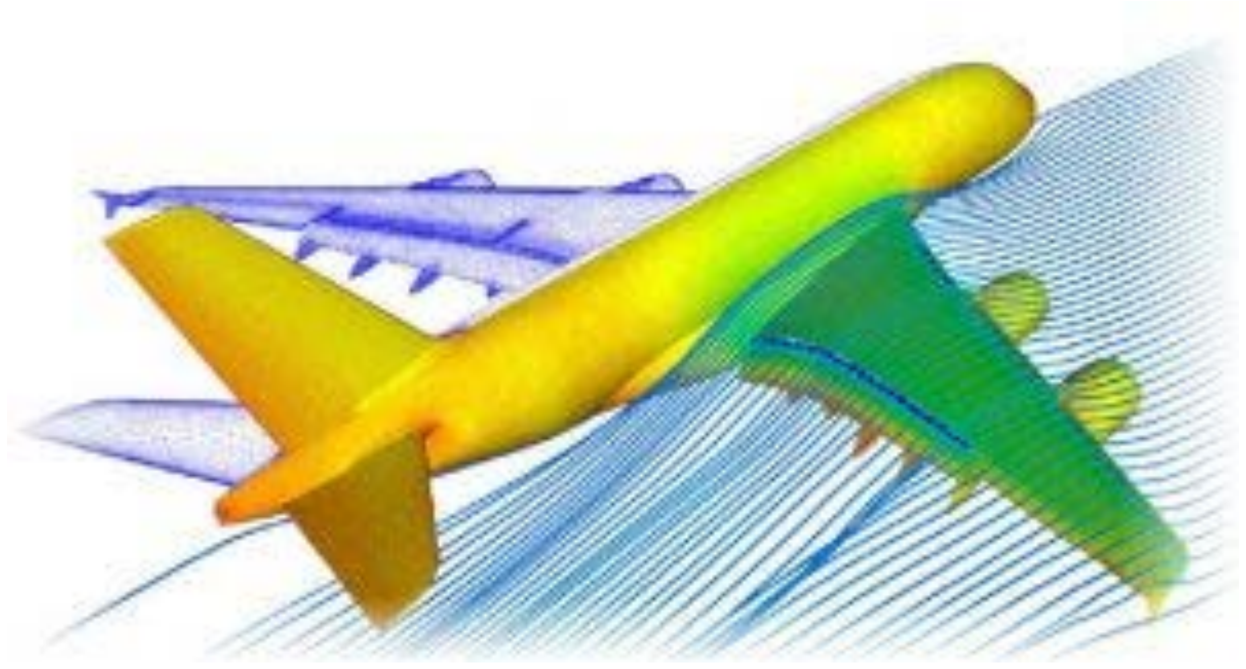
Variable End-to-End Latency



L. Ravindranath et al., Timecard: Controlling User-Perceived Delays in Server-Based Mobile Applications, SOSP, 2013.

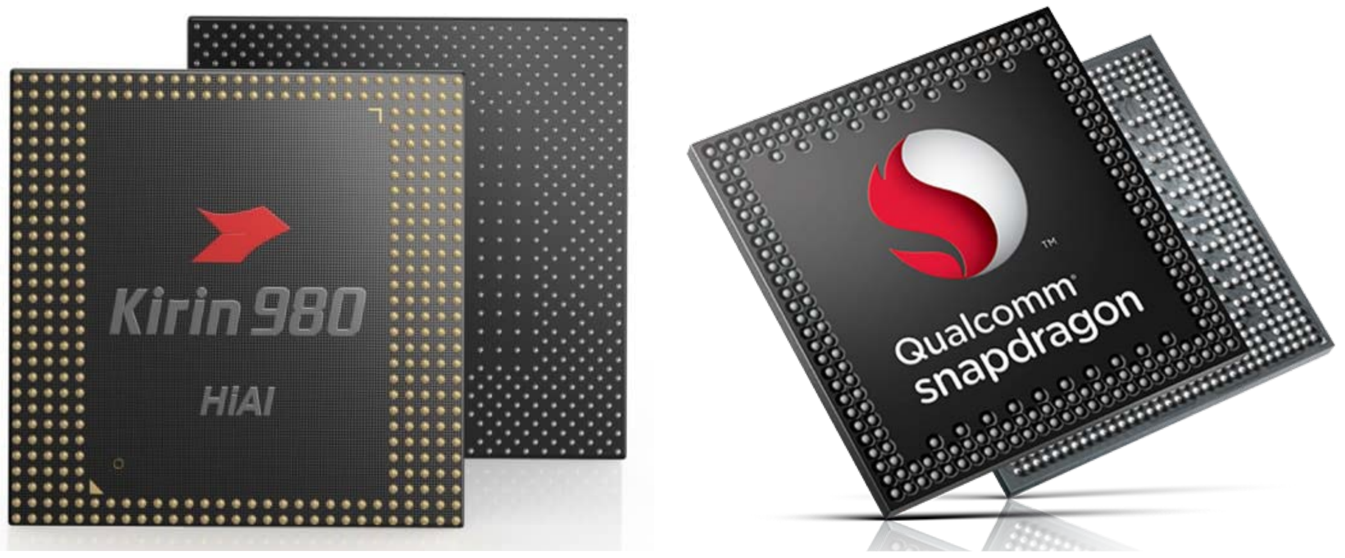
Negative Impact on Real-time Systems

- **Aviation:** leave one active core, turn off other cores

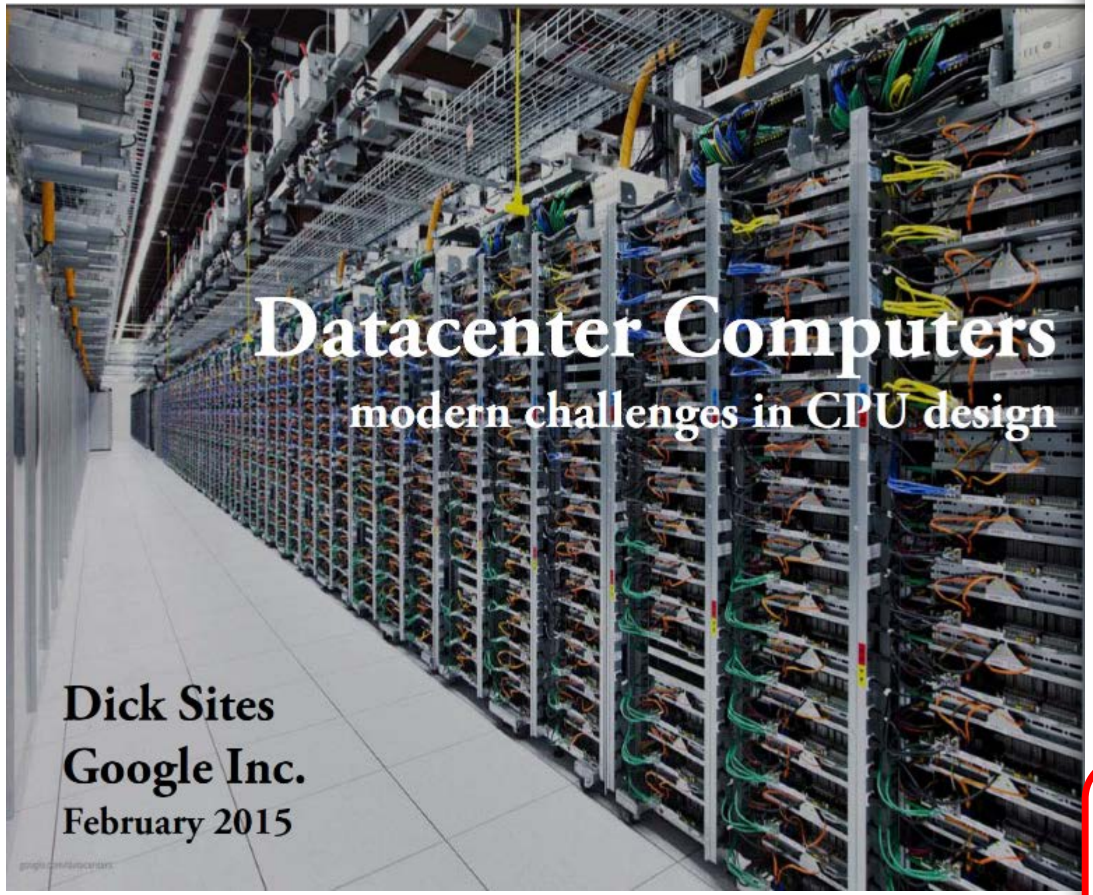


http://ukaerodynamics.co.uk/the-aerodynamics-basics/

- **Smartphone:** Overprovision resources



More Hardware Support Needed

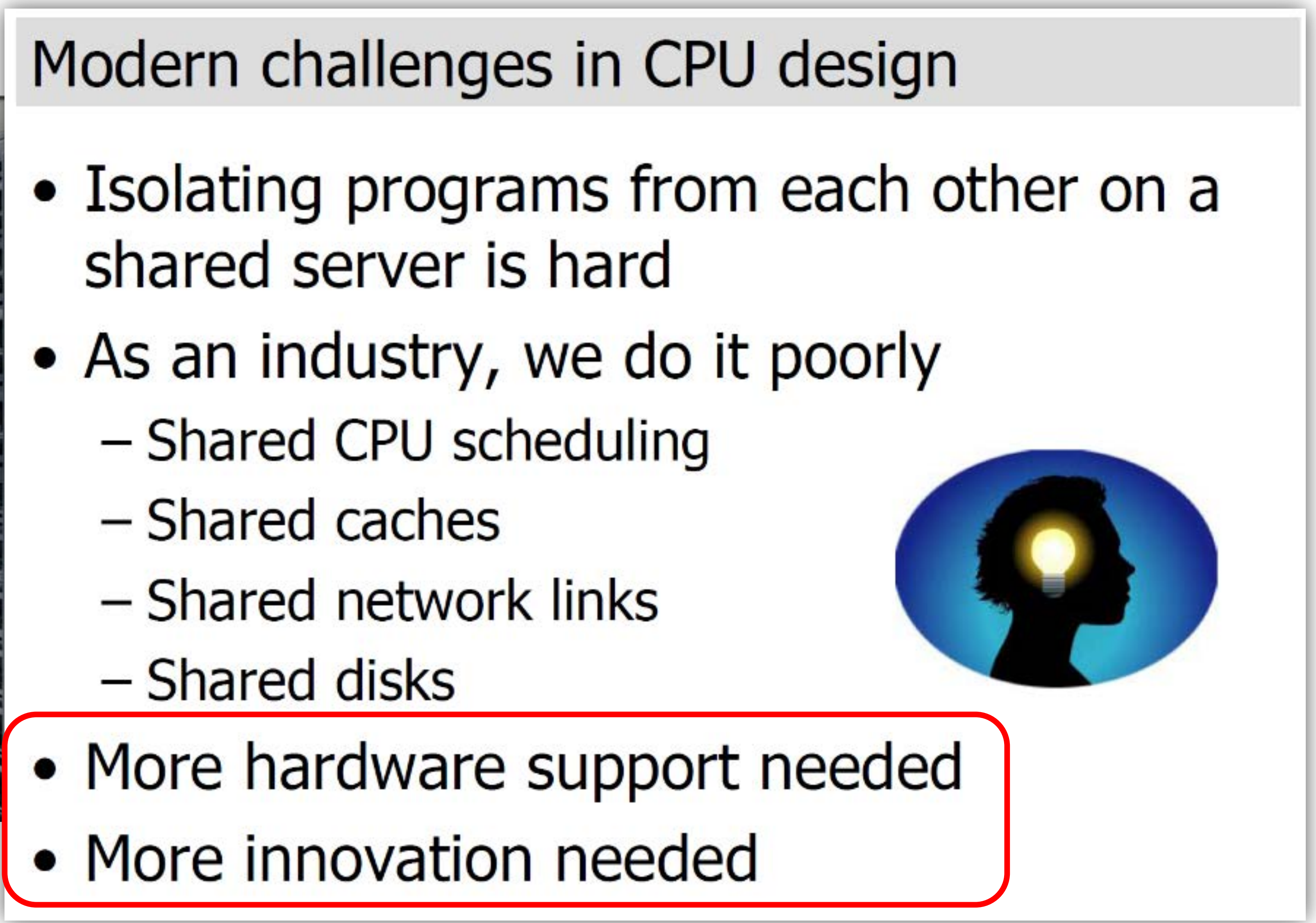

Datacenter Computers

modern challenges in CPU design

Dick Sites

Google Inc.

February 2015


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

Dick Sites, Datacenter Computers: modern challenges in CPU design, 2015.

Intel's RDT

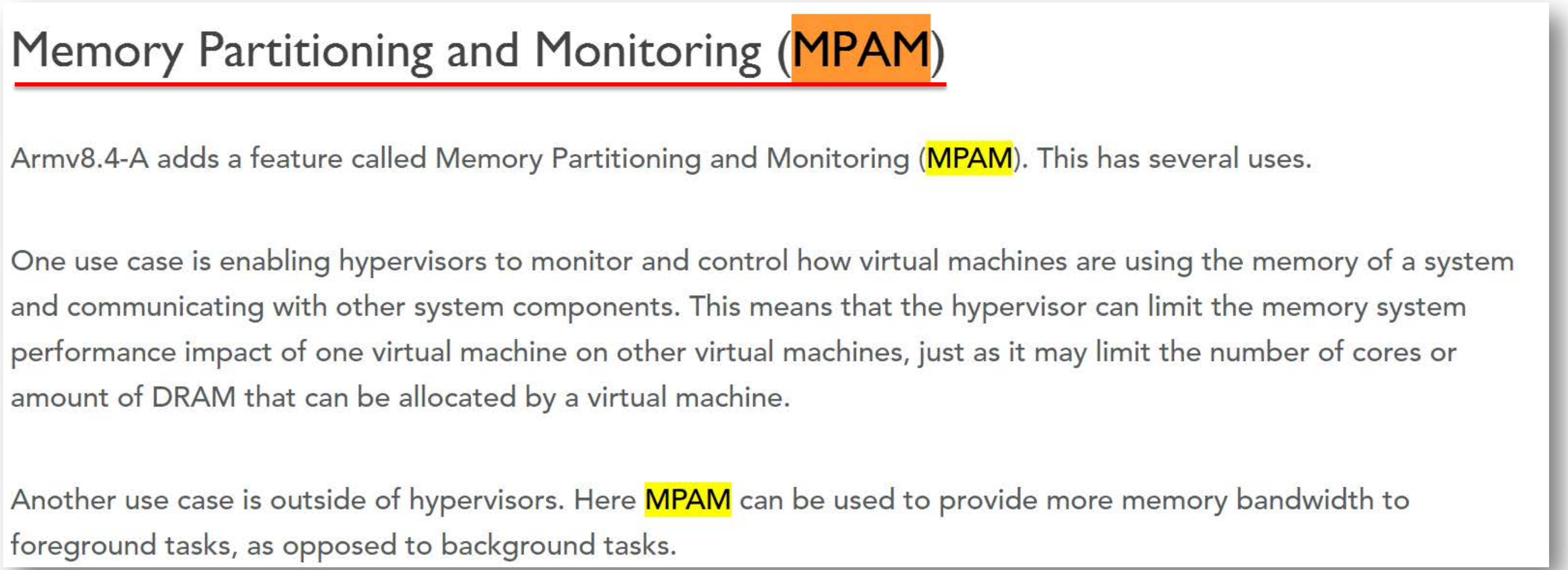
- In April 2016, Intel **released Resource Director Technology (RDT)**
  - Cache Monitoring Technology (CMT)
  - Cache Allocation Technology (CAT)
  - Memory Bandwidth Monitoring(MBM)
  - Code and Data Prioritization (CDP)

intel
XEON
inside
INTEL® XEON® PROCESSOR
E5-2600 v4 PRODUCT FAMILY
Intel® Resource Director Technology

[1] https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html
[2] Improving Real-Time Performance by Utilizing Cache Allocation Technology, Intel Corporation, Apr 2015.

ARM's MPAM

- ARM is catching up: **Memory Partition and Monitoring (MPAM)**



Memory Partitioning and Monitoring (MPAM)

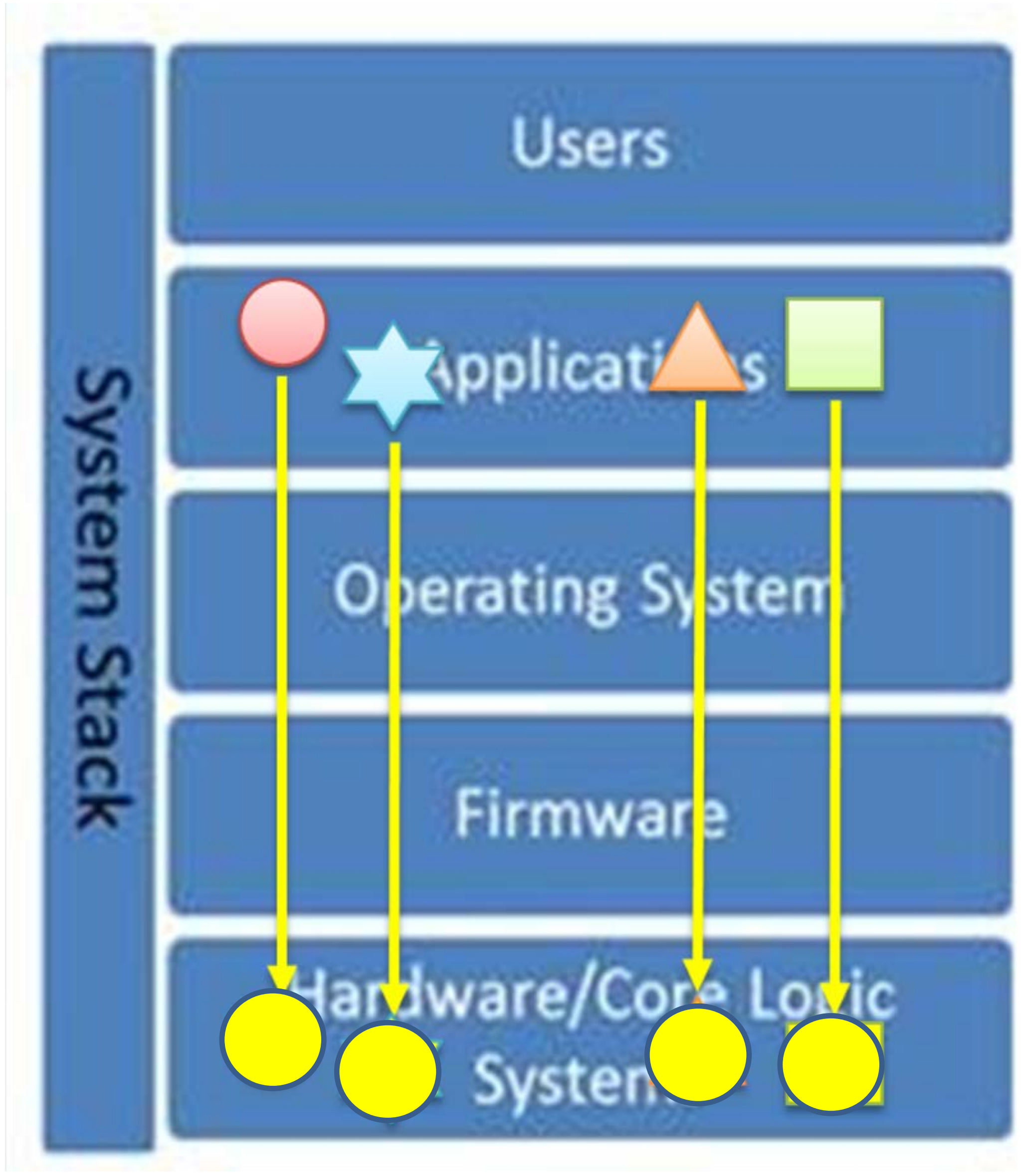
Armv8.4-A adds a feature called Memory Partitioning and Monitoring (MPAM). This has several uses.

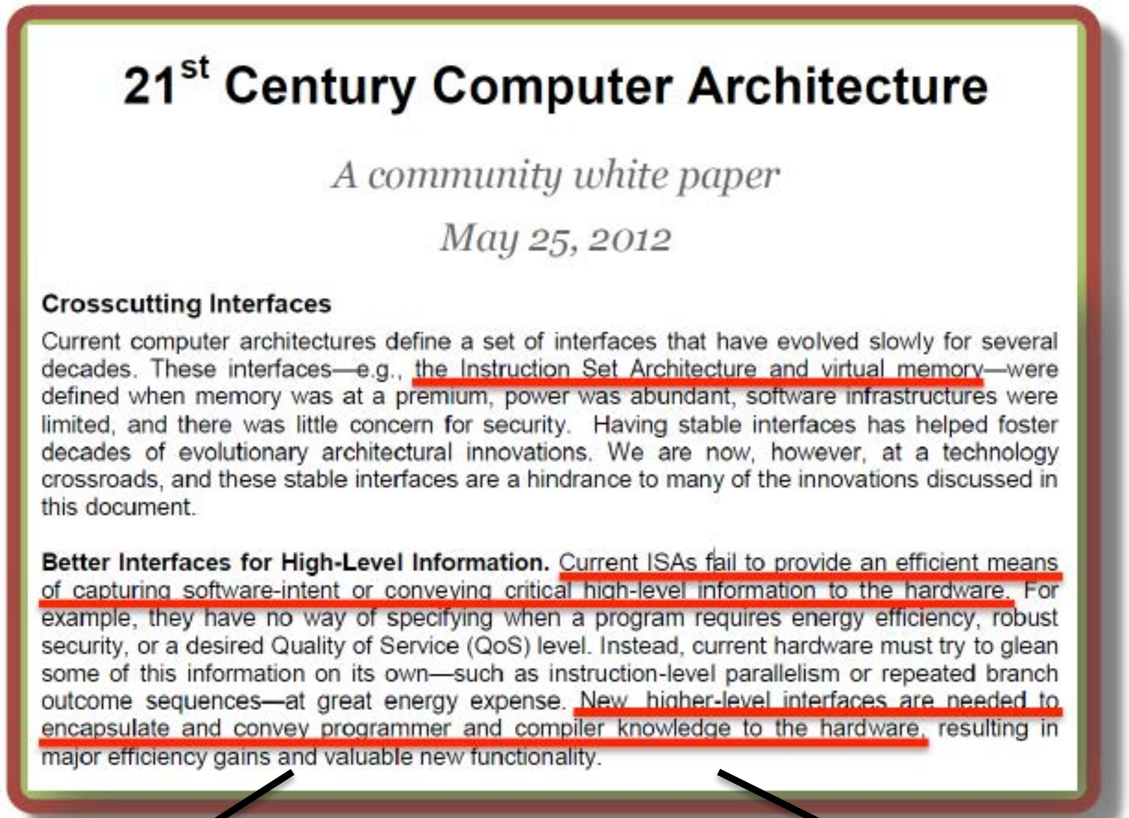
One use case is enabling hypervisors to monitor and control how virtual machines are using the memory of a system and communicating with other system components. This means that the hypervisor can limit the memory system performance impact of one virtual machine on other virtual machines, just as it may limit the number of cores or amount of DRAM that can be allocated by a virtual machine.

Another use case is outside of hypervisors. Here MPAM can be used to provide more memory bandwidth to foreground tasks, as opposed to background tasks.

https://community.arm.com/processors/b/blog/posts/introducing-2017s-extensions-to-the-arm-architecture

Academy's Efforts





21st Century Computer Architecture

A community white paper

May 25, 2012

Crosscutting Interfaces

Current computer architectures define a set of interfaces that have evolved slowly for several decades. These interfaces—e.g., the Instruction Set Architecture and virtual memory—were defined when memory was at a premium, power was abundant, software infrastructures were limited, and there was little concern for security. Having stable interfaces has helped foster decades of evolutionary architectural innovations. We are now, however, at a technology crossroads, and these stable interfaces are a hindrance to many of the innovations discussed in this document.

Better Interfaces for High-Level Information. Current ISAs fail to provide an efficient means of capturing software-intent or conveying critical high-level information to the hardware. For example, they have no way of specifying when a program requires energy efficiency, robust security, or a desired Quality of Service (QoS) level. Instead, current hardware must try to glean some of this information on its own—such as instruction-level parallelism or repeated branch outcome sequences—at great energy expense. New, higher-level interfaces are needed to encapsulate and convey programmer and compiler knowledge to the hardware, resulting in major efficiency gains and valuable new functionality.

"New, high-level interfaces are required to convey programmer and compiler knowledge to the hardware."

21st Century Computer Architecture

My Story at Princeton

PARSEC
['pär-''sek] A unit of measure

SDN Research @ Princeton

**Prof.
Jenifer Rexford**

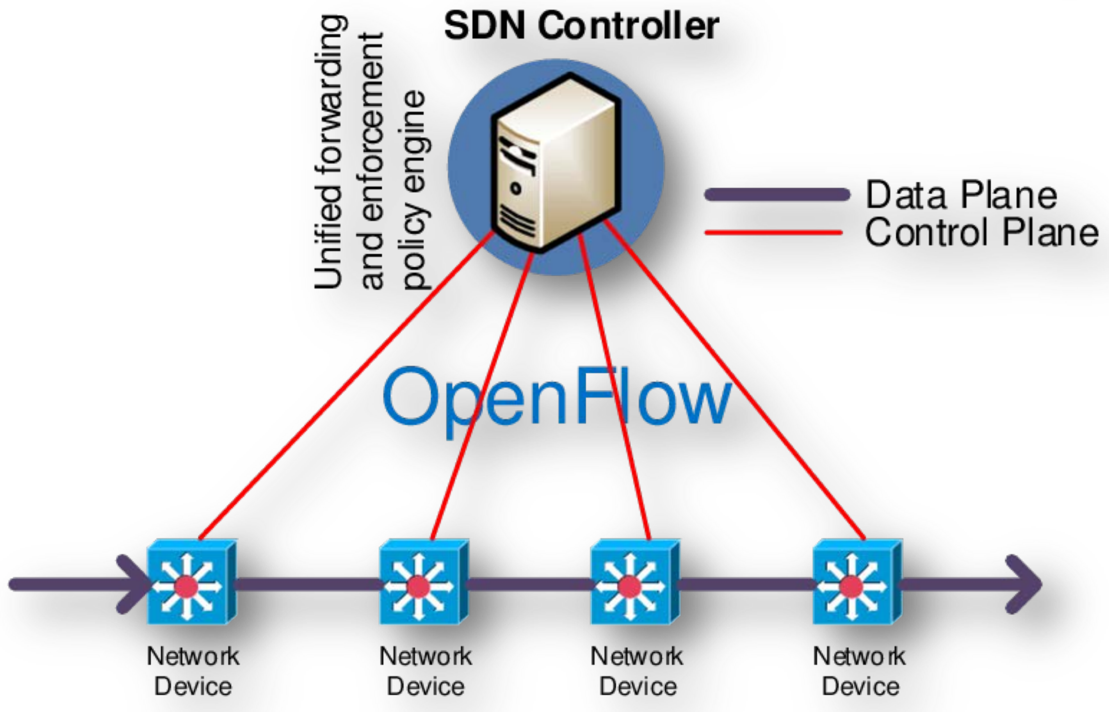
**Prof.
David Walker**

Xin Jin

frenetic >>

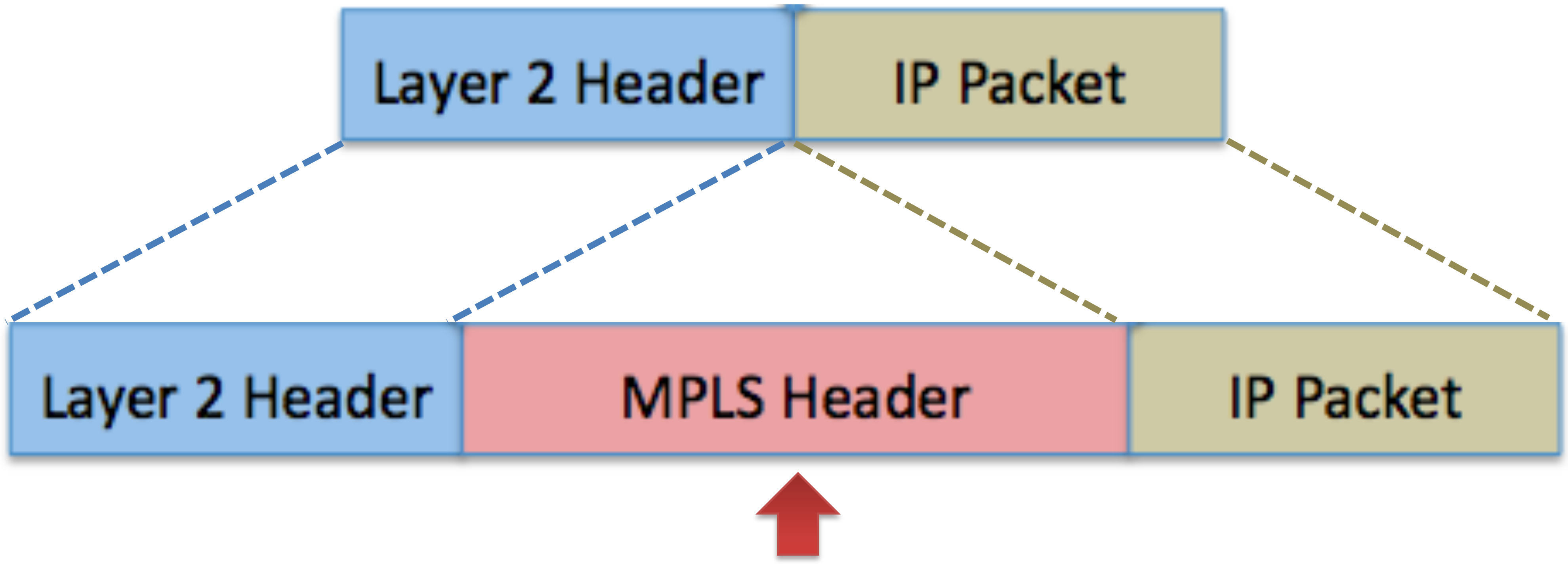
A Network Programming Language

P4



Labeled Network

- **Fine-grain** : every packet has a label
- **Semantics** : correlate labels with users' demand
- **Propagation** : propagate labels in a whole network
- **DiffServ** : process packets differentiately based on labels; SDN further provides programmability

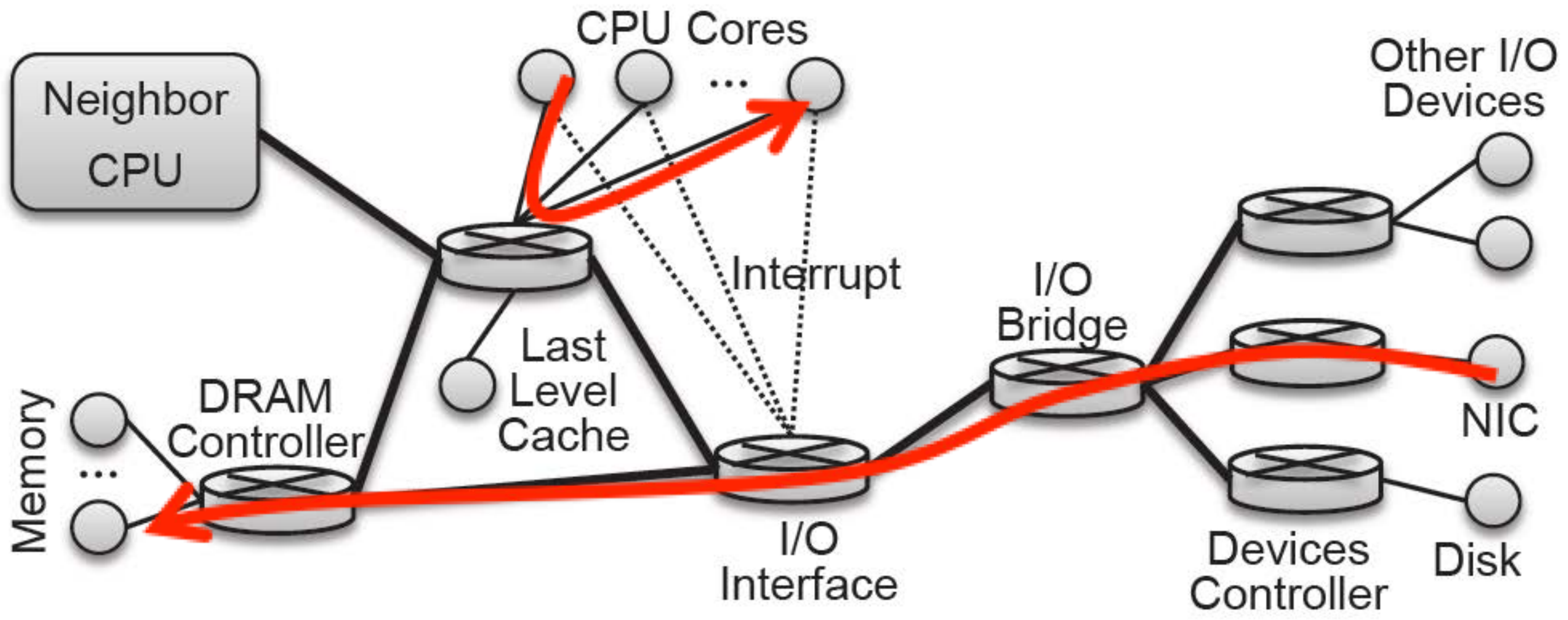


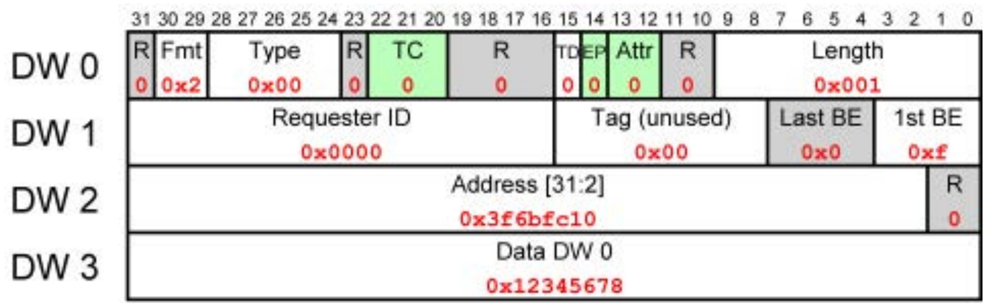
MPLS is widely used for VPN and QoS

Reconstruct a computer as an SDN network?

Yes!

- Hardware components communicate via internal packets, e.g., PCIe packets, NoC packets, QPI packets





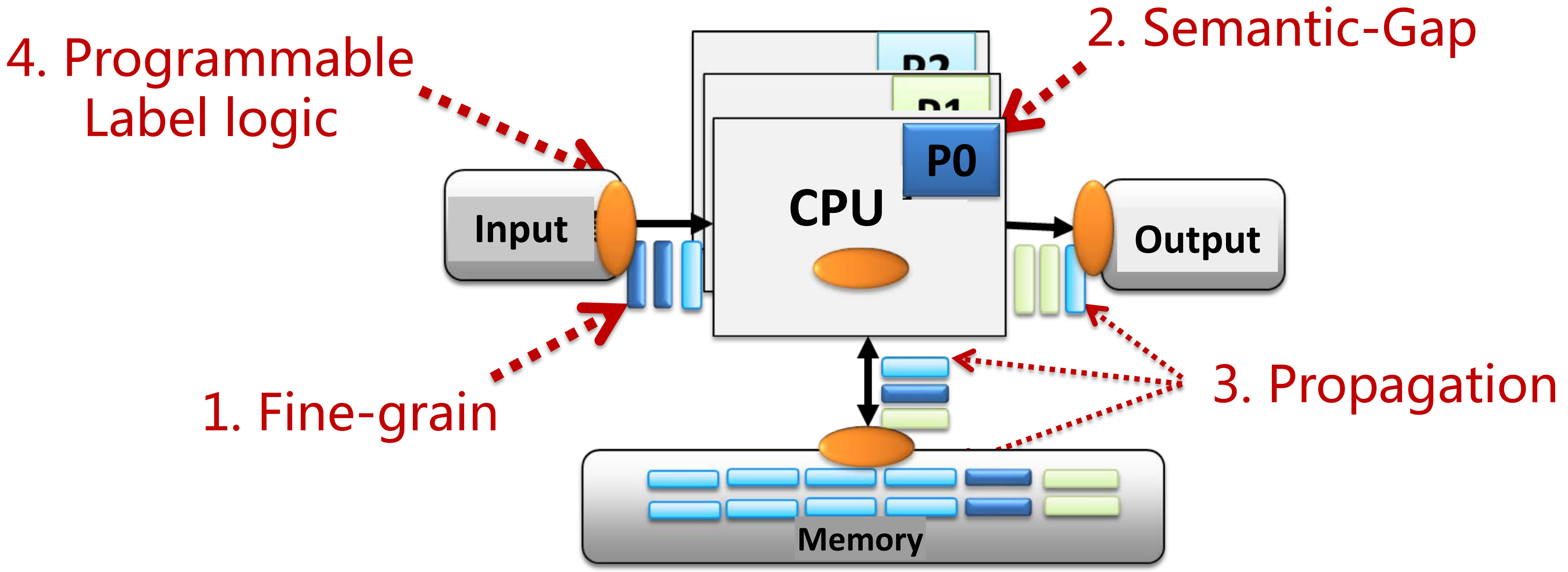
	31	30 29	28 27 26 25 24	23	22 21 20	19 18 17 16	15	14	13 12	11 10	9 8 7 6 5 4 3 2 1 0
DW 0	R	Fmt	Type	R	TC	R	TD	EP	Attr	R	Length
	0	0x2	0x00	0	0	0	0	0	0	0	0x001

	31–16	15–8	7–4	3–0
DW 1	Requester ID	Tag (unused)	Last BE	1st BE
	0x0000	0x00	0x0	0xf

	31–2	1–0
DW 2	Address [31:2]	R
	0x3f6bfc10	0
DW 3	Data DW 0	
	0x12345678	

Labeled von Neumann Architecture (LvNA)

- **Fine-grain :** attach a label to each memory and I/O request
- **Semantics** : correlate labels with VM/Proc/Thread/Var
- **Propagation** : propagate labels in a whole machine
- **Programmable label control logic (CL):** : provide differentiated services based on different label-indexed rules



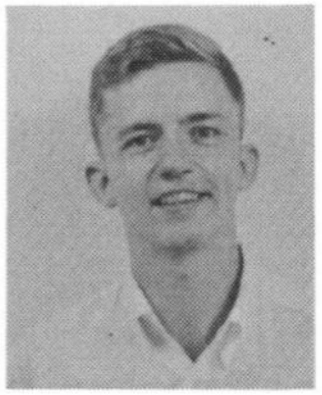
Bao and Wang, Labeled von Neumann Architecture for Software-Defined Cloud, Journal of Computer Science and Technology, 2017 Vol. 32 (2): 219-223.

Revisiting Tagged Architecture

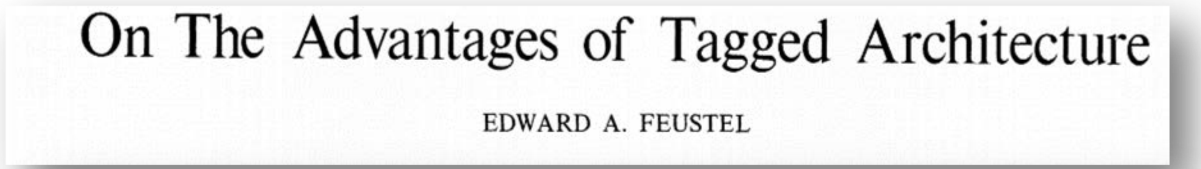
In 1973, Feustel proposed **Tagged Architecture** to replace von Neumann architecture

- All data elements are assigned with a type tag
- Tags are stored in memory
- Tags trigger trap to process

Too complicated to impl

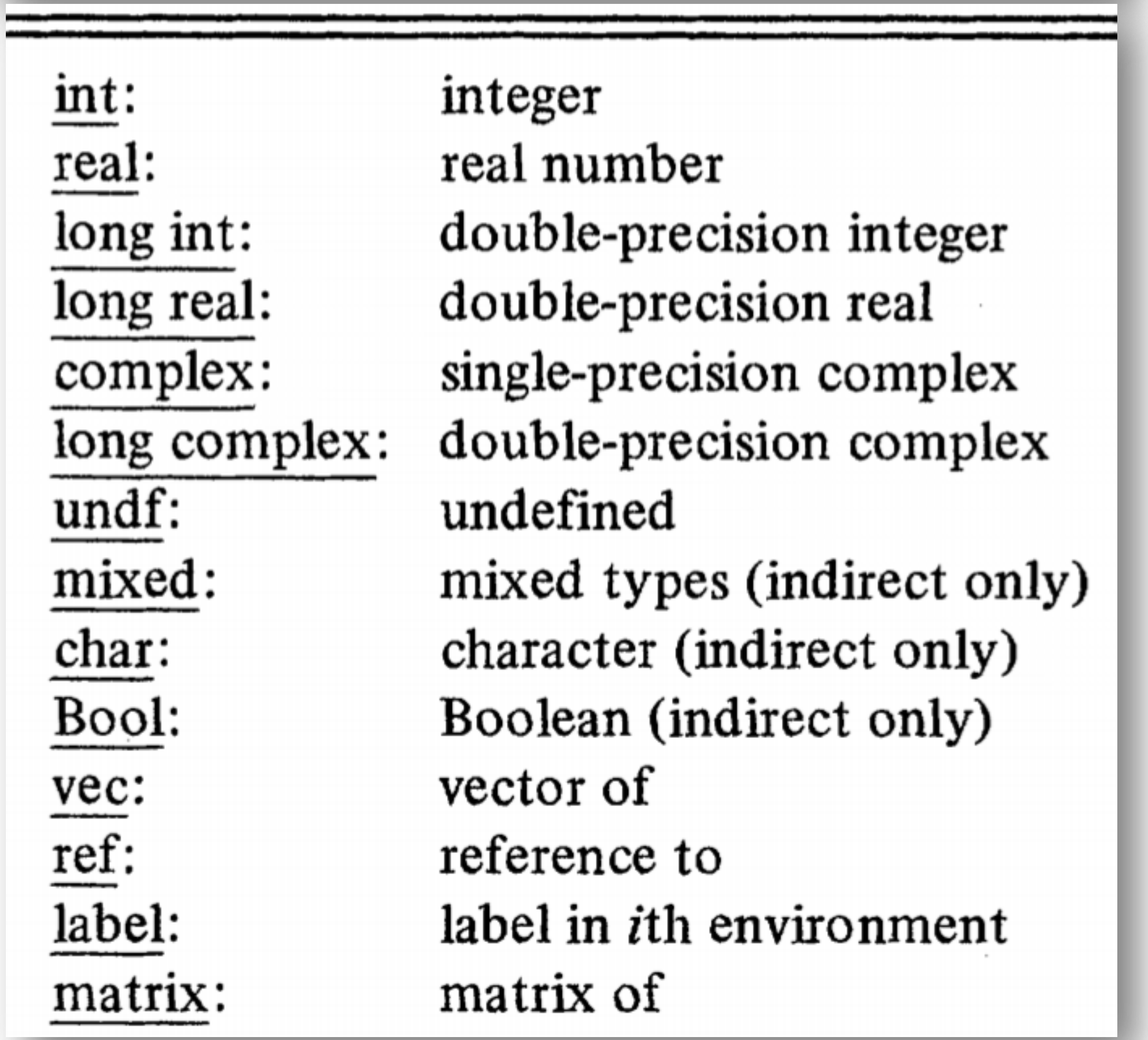


- Edward A. Feustel
- Rice Univ.



On The Advantages of Tagged Architecture

EDWARD A. FEUSTEL

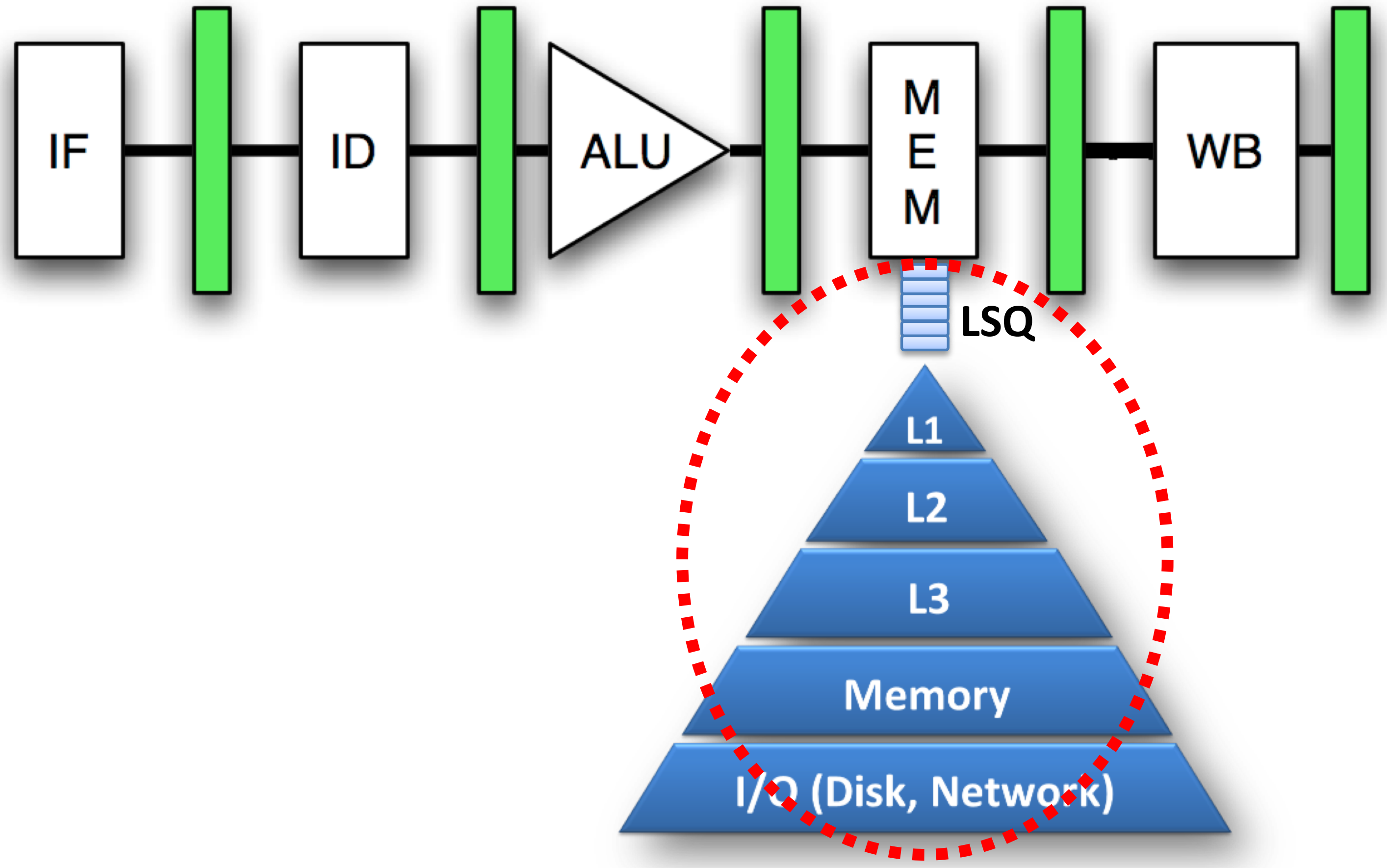


int:	integer
real:	real number
long int:	double-precision integer
long real:	double-precision real
complex:	single-precision complex
long complex:	double-precision complex
undf:	undefined
mixed:	mixed types (indirect only)
char:	character (indirect only)
Bool:	Boolean (indirect only)
vec:	vector of
ref:	reference to
label:	label in *i*th environment
matrix:	matrix of

LvNA's Principles

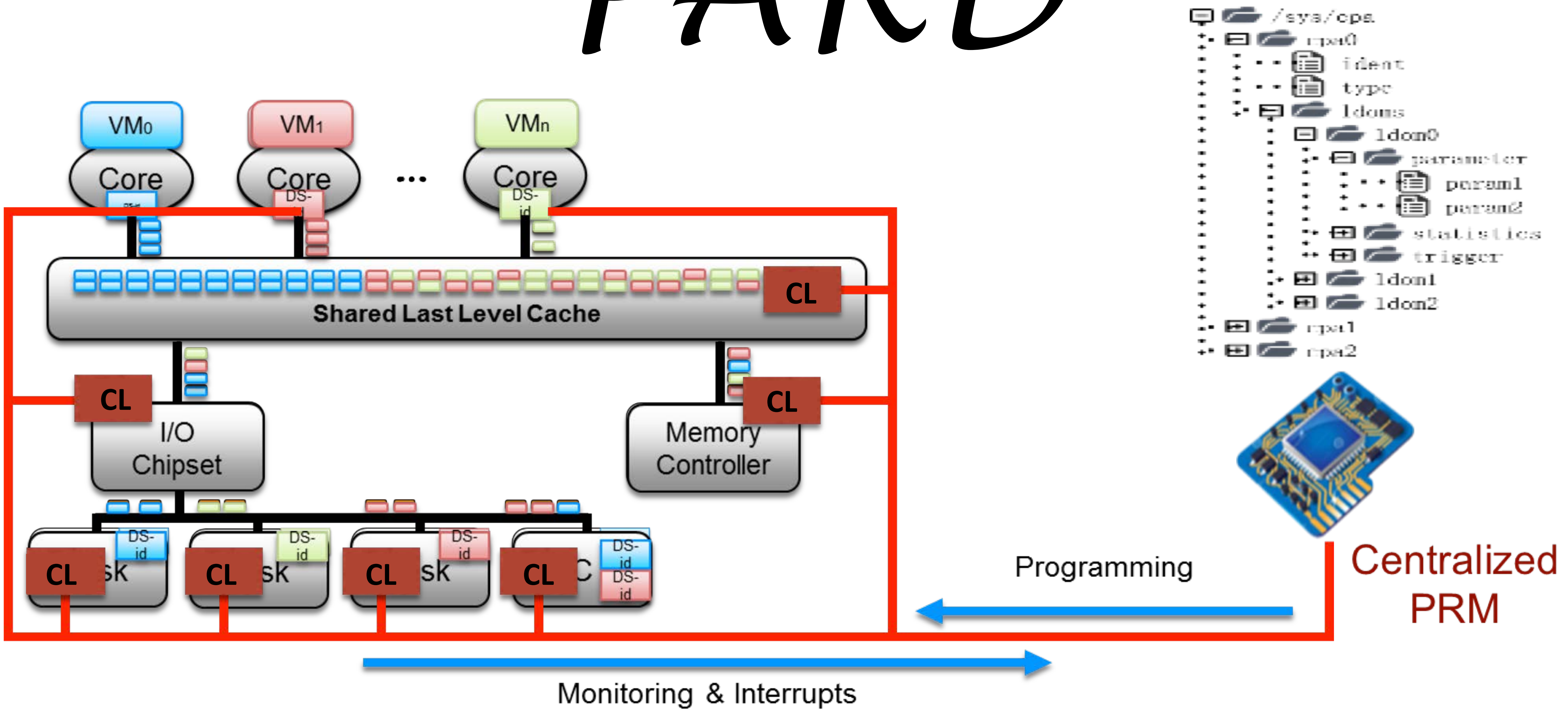
Minimal intrusion to the existing architecture

- **#1:** Labels should be as simple as possible
- **#2:** Labels are stored in requests rather than in memory
- **#3:** Control logics (CLs) reside in datapaths



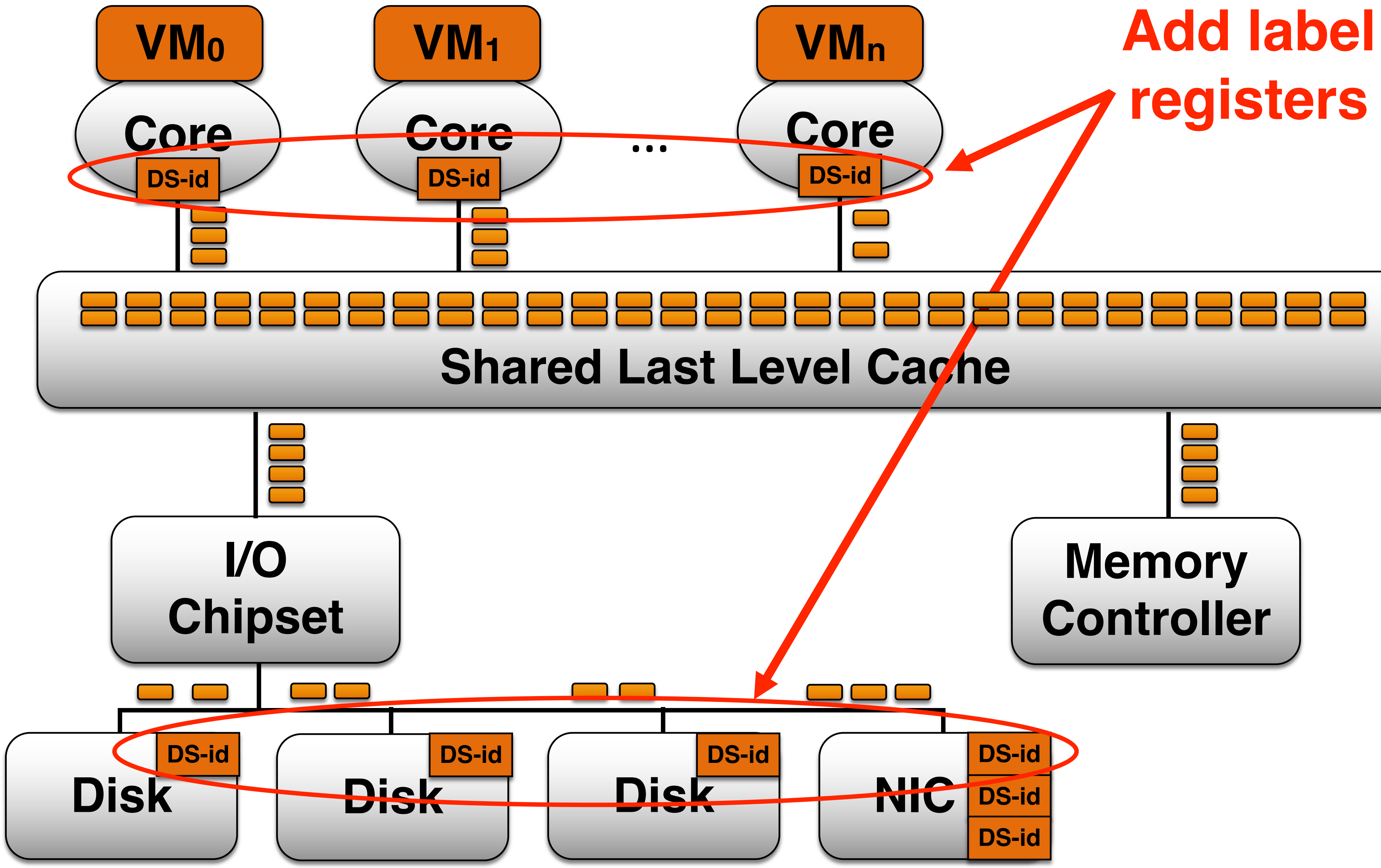
Programmable **A**rchitecture for **R**esourcing-on-**D**emand

PARD

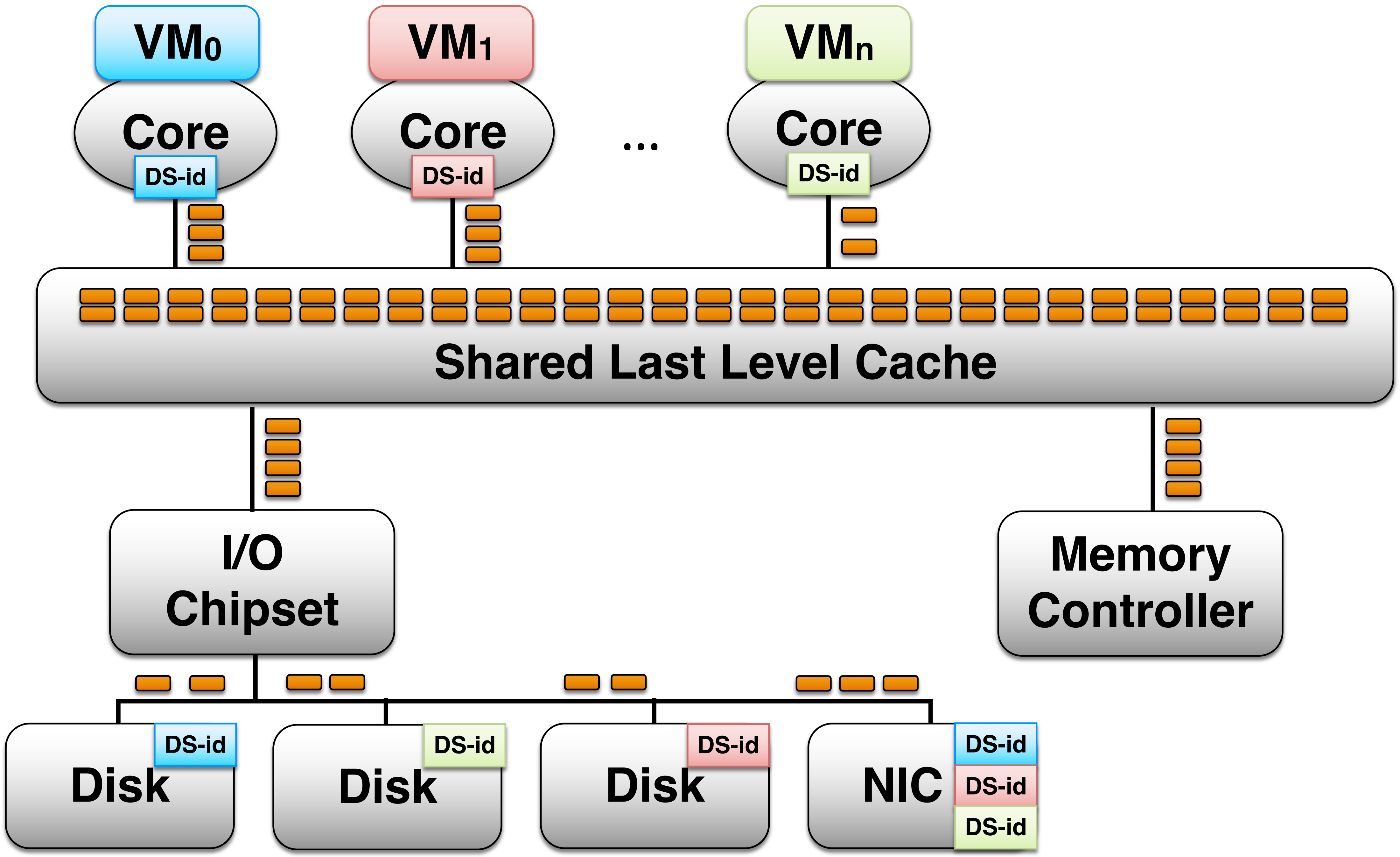


Ma et. al, Supporting Differentiated Services in Computers via Programmable Architecture for Resourcing-on-Demand (PARD), ***ASPLOS***, *2015*

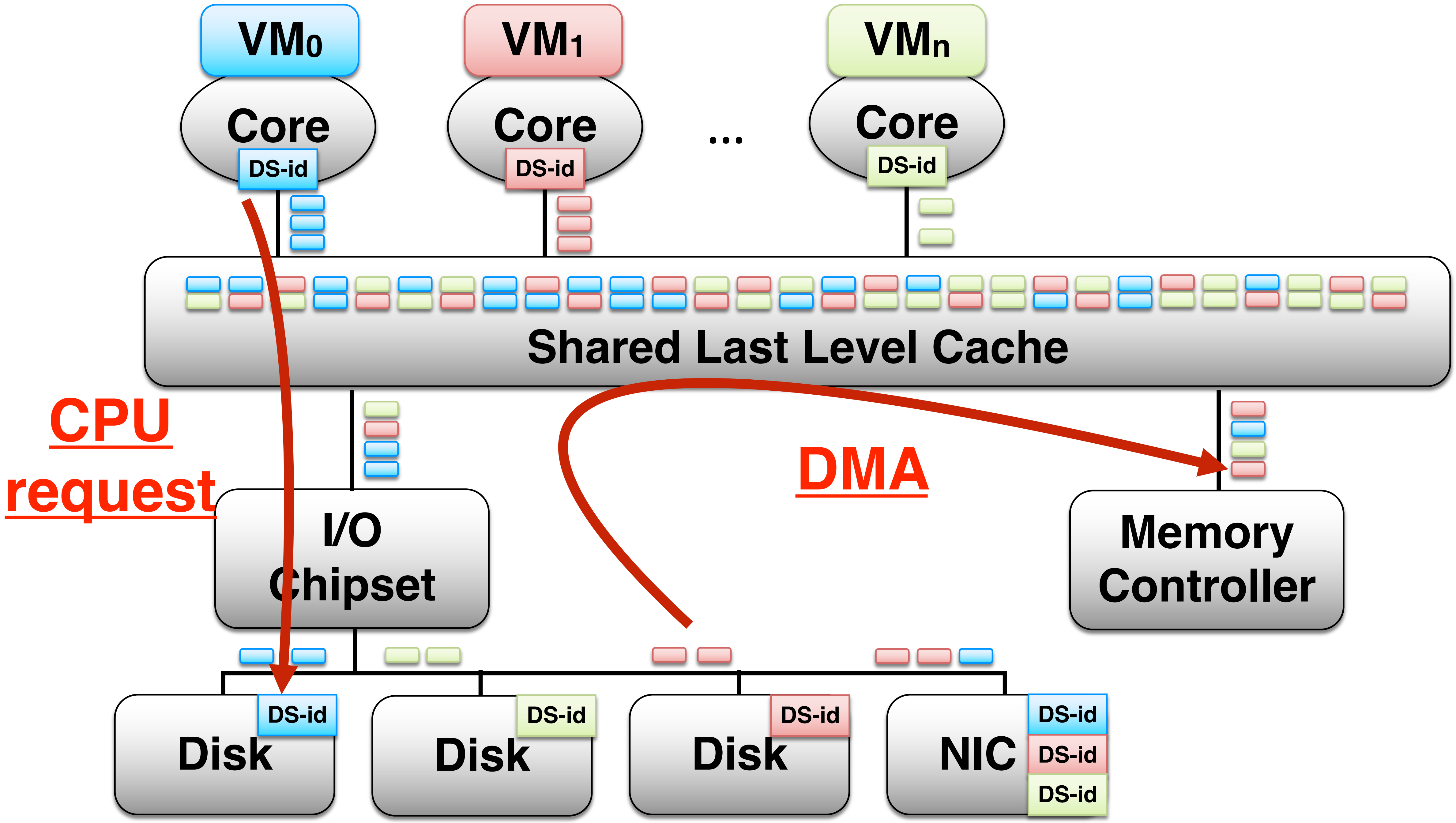
Registers to store labels



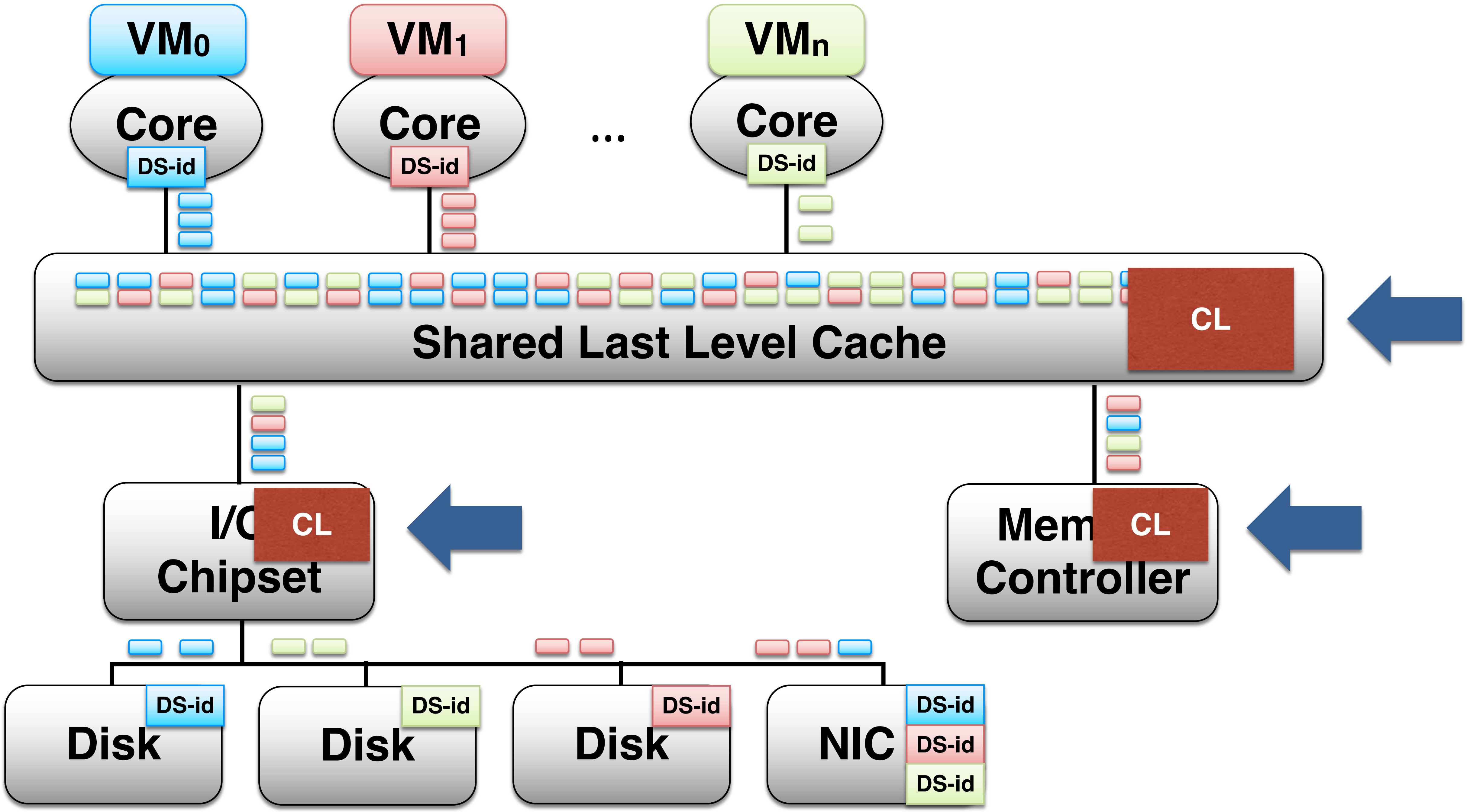
Assign Entities w/ Labels



Label Propagation



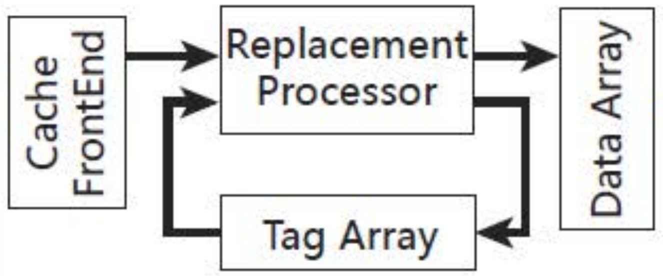
Programmable Control Logic



Attempt 1: MCU-based CL

Replacement Processor

```
rbget s0, rb0    # get request
   ...     # extra <set,tag> to <r2,r3>
mtrb s1, r2, 0   # build tag array req
rbput rb1, s1    # send req to tag array
rbget s1, rb1    # wait resp
   ...     # do pseudo-LRU Replacement
           # r4 <- victim way
mtrb s0, r4, 0   # build replacement req
rbput rb0, s0    # send out request
```

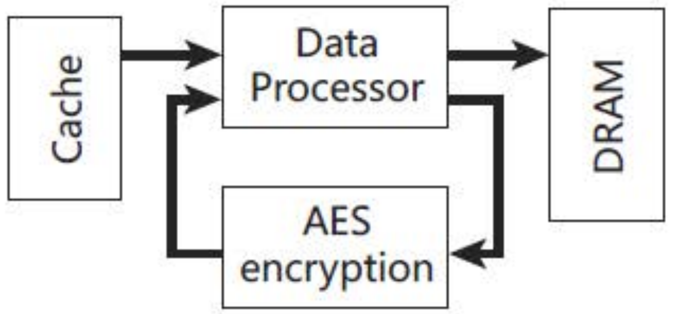


Cache Replacement

Memory Access Encryption

Data Processor

```
rbget s0, rb0    # get data request
rbput rb1, s0    # send data to AES
rbget s0, rb1    # wait AES response
rbput rb0, s0    # send out request
```

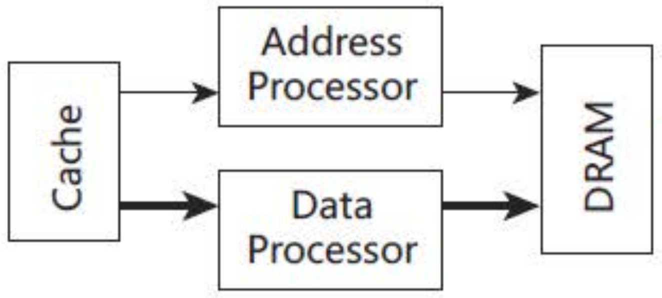


Address Processor

```
rbget s0, rb0    # get request
mfrb r1, s0, 1   # get address
mfrb r2, s0, 0   # get DSid (low-16bit)
andi r2, r2, 0xFFFF
bsrl r2, r2, 2   # offset: DSid*4
# translate base addr stored in r5
lwx r3, r5, r2   # r3 <- base[DSid]
add r1, r1, r3   # addr += base[DSid]
mtrb s0, r1, 1   # write address
rbput rb0, s0    # send out request
```

Data Processor

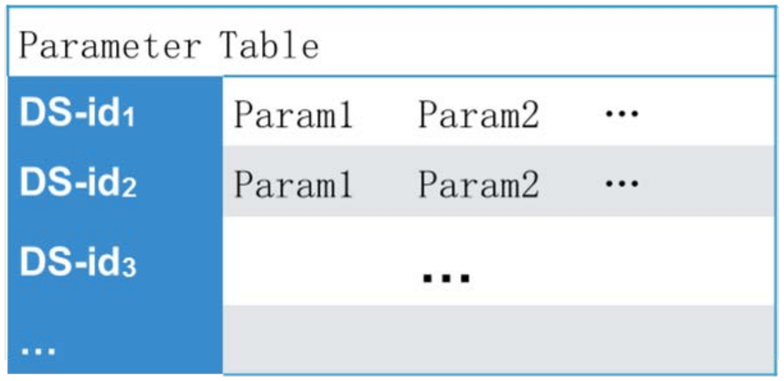
```
rbget s0, rb0    # get request
rbput rb0, s0    # send out request
```



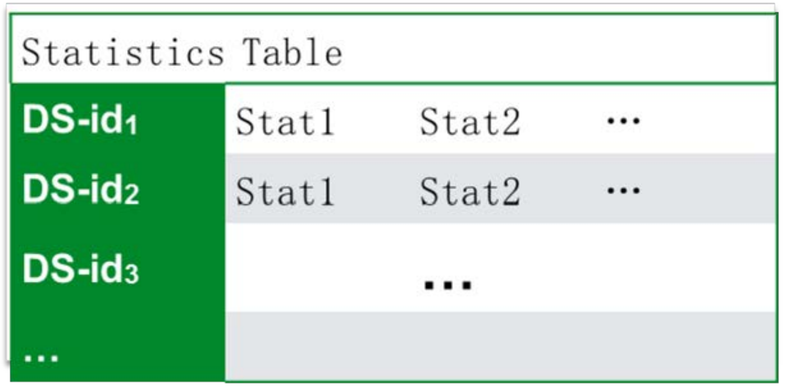
Address Mapping for Virtualization

Ma et. al, A Programmable Data Plane Design in Computer Architecture, *Journal of Computer Research and Development*, 2017

Attempt 2: Tables as CL

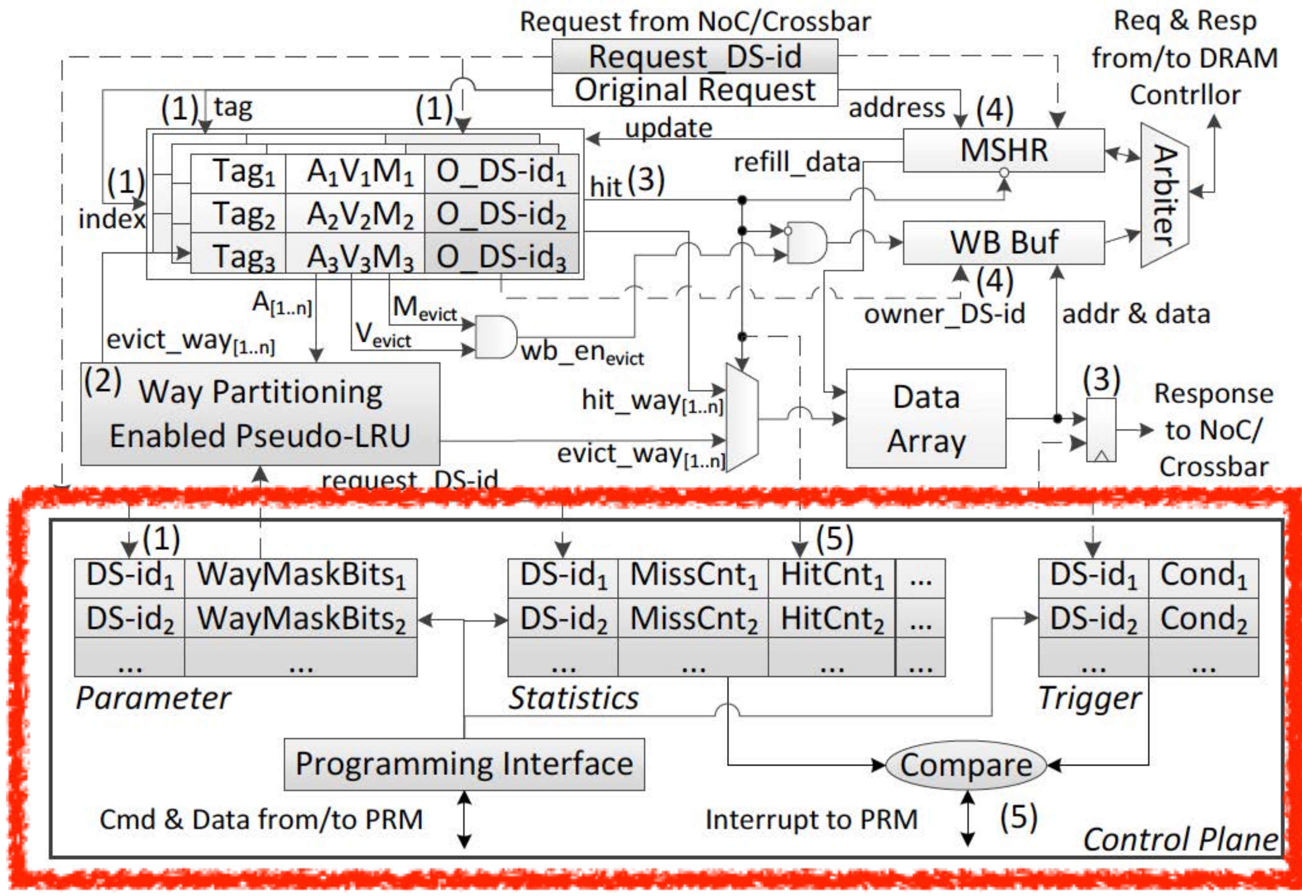


Parameter Table			
DS-id1	Param1	Param2	…
DS-id2	Param1	Param2	…
DS-id3		…	
…			

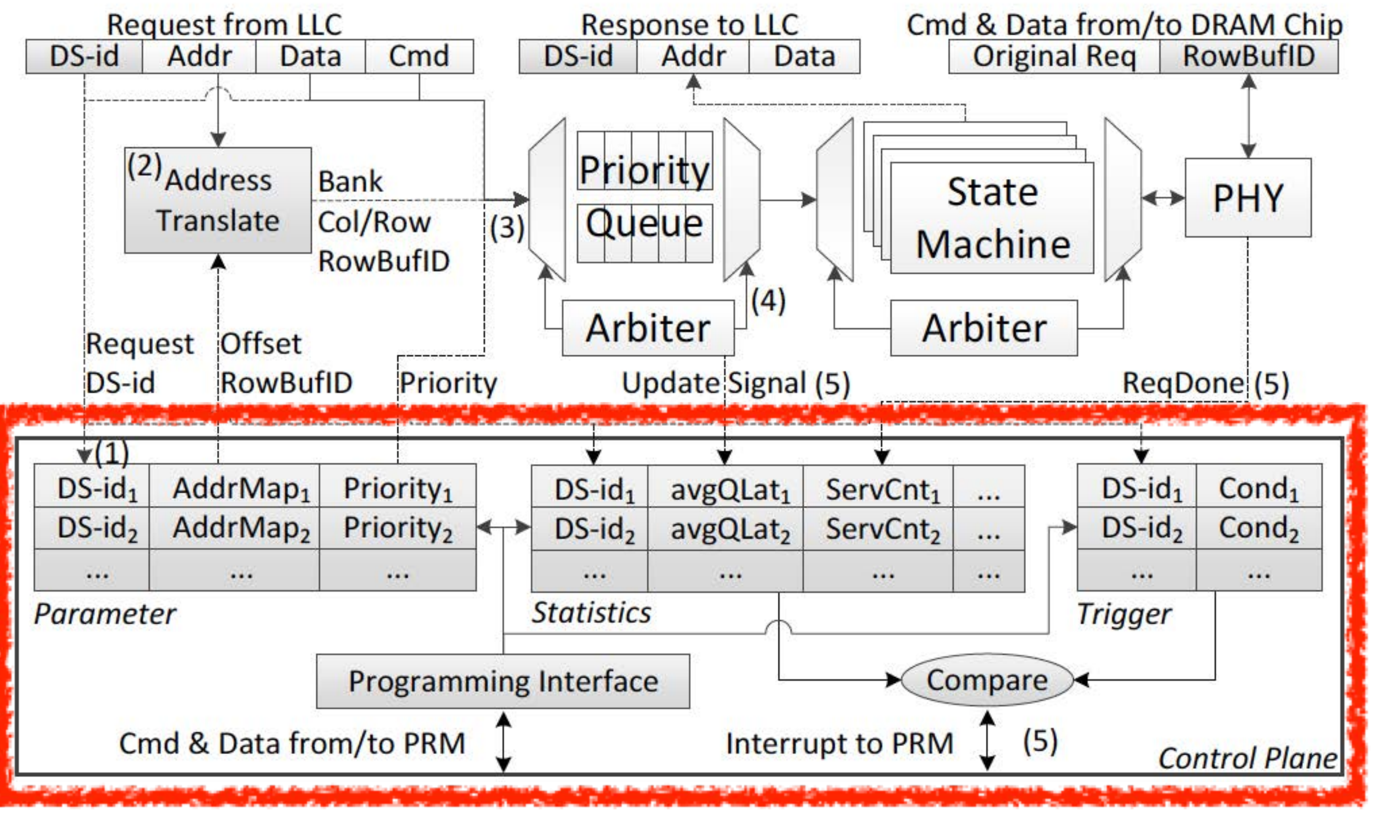


Statistics Table			
DS-id1	Stat1	Stat2	…
DS-id2	Stat1	Stat2	…
DS-id3		…	
…			

Cache Controller

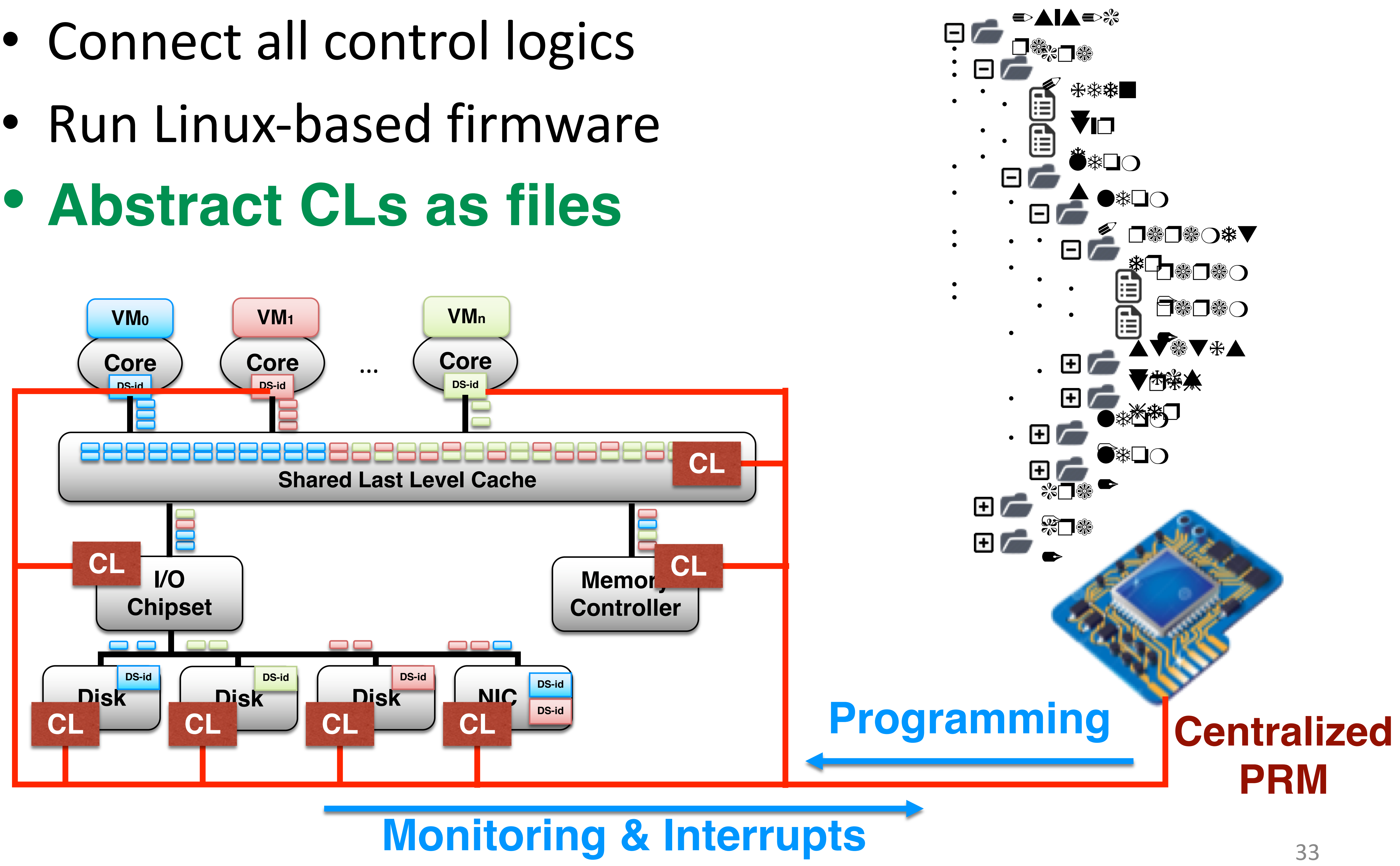


Memory Controller



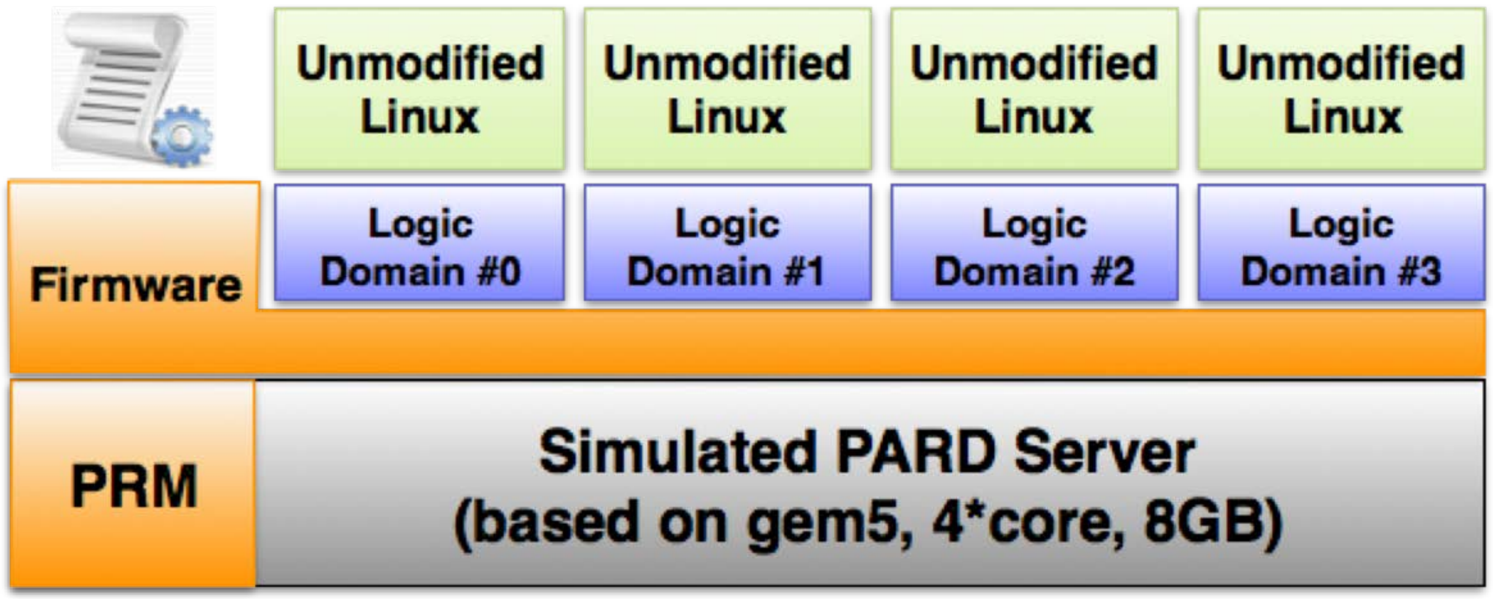
Platform Resource Manager (PRM)

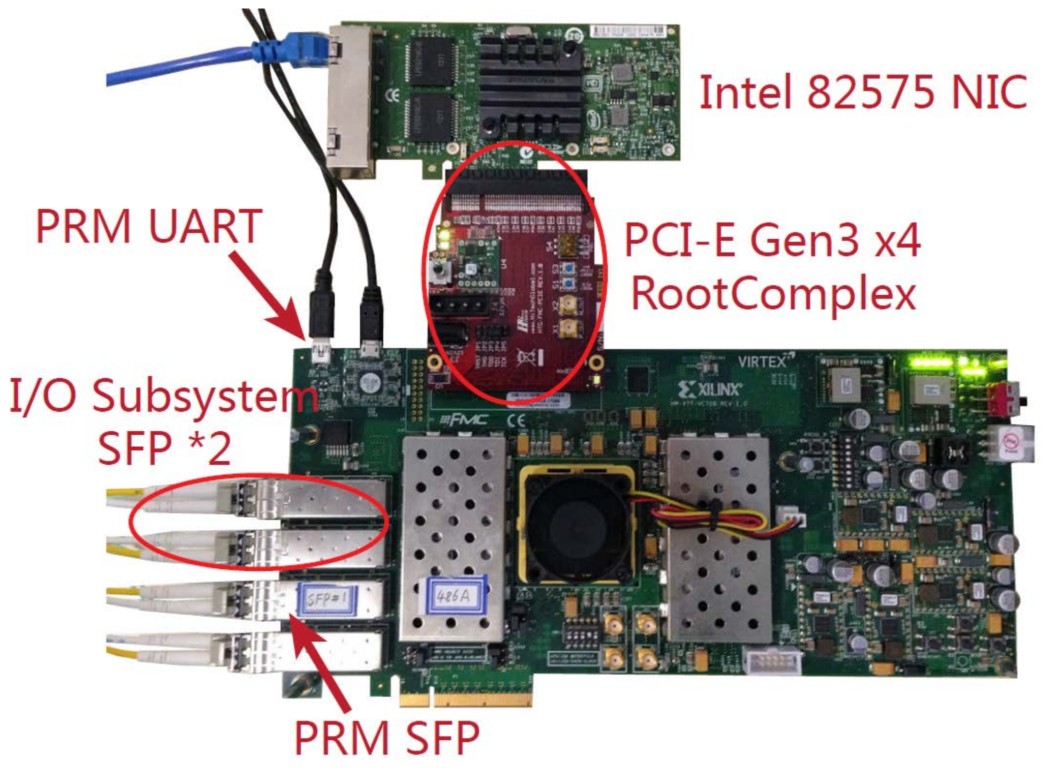
- Connect all control logics
- Run Linux-based firmware
- **Abstract CLs as files**



Implementation

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



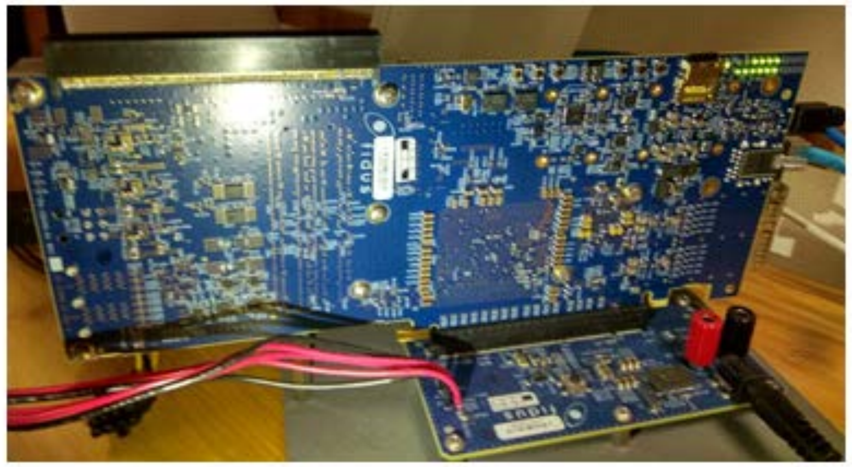


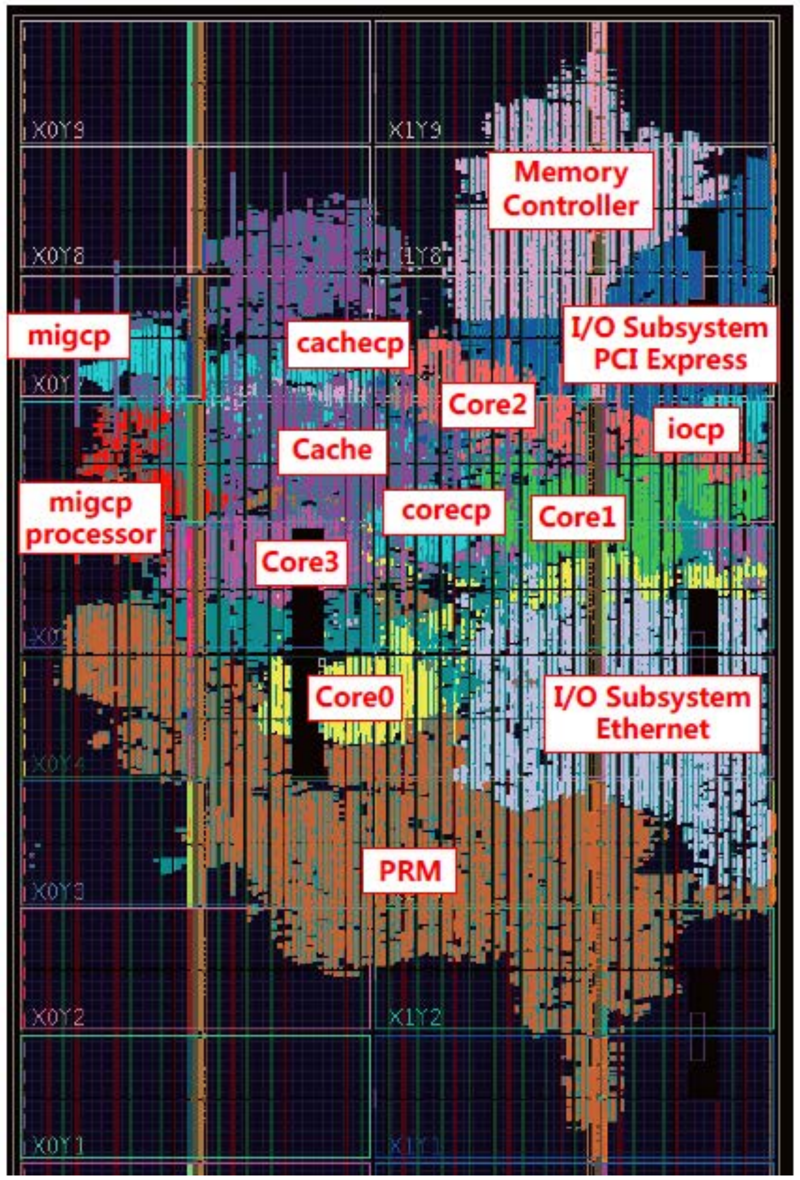
Labeled RISC-V

- 4 core/16-way L2$/DRAMCtrler/1GbE
- Run on different FPGAs

Digilent Zedboard

Xilinx ZCU102

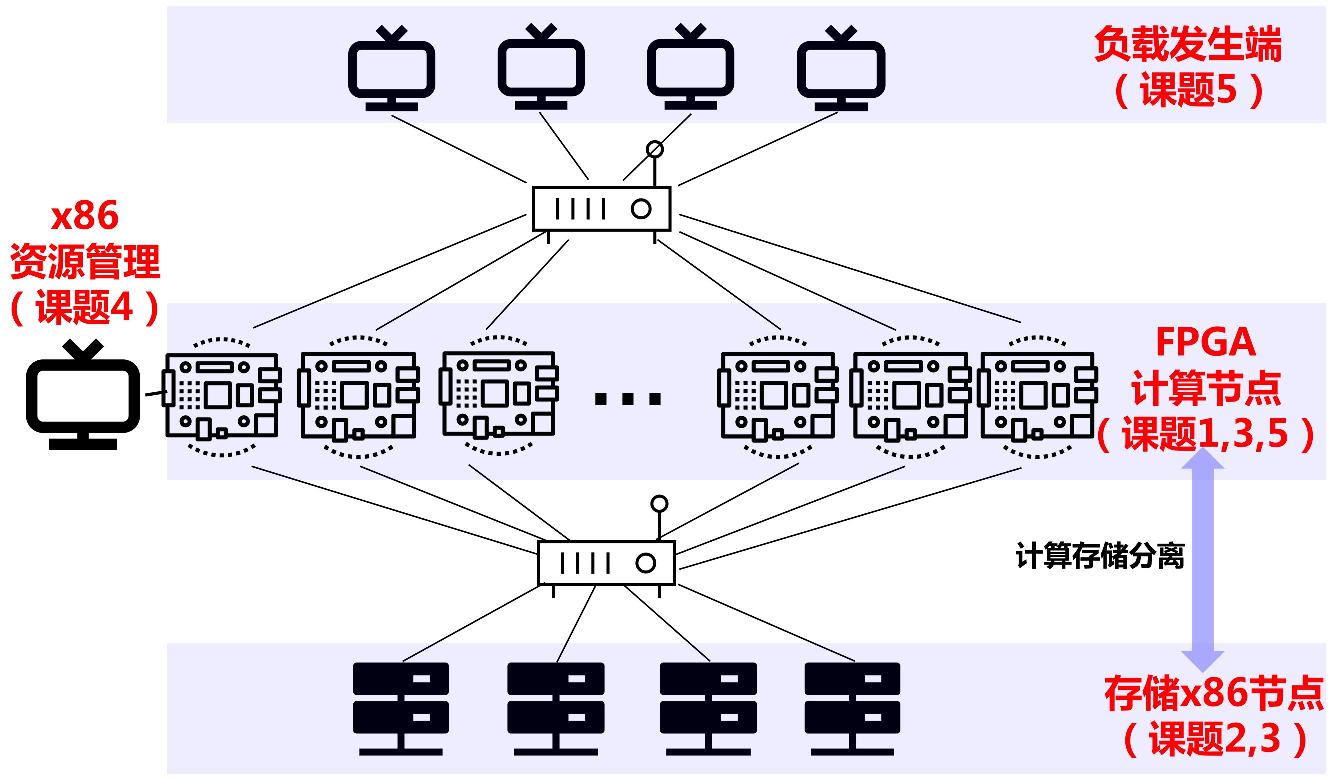

Fidus Sidewinder-100



Yu et al.,, **Labeled RISC-V: A New Perspective on Software-Defined Architecture**. *Workshop on Computer Architecture Research with RISC-V (**CARRV**), 2017*

8-node FlameCluster Prototype

- Each node: Labeled RISC-V
- Run Linux, Ceph Client, labeled TCP/IP Stack



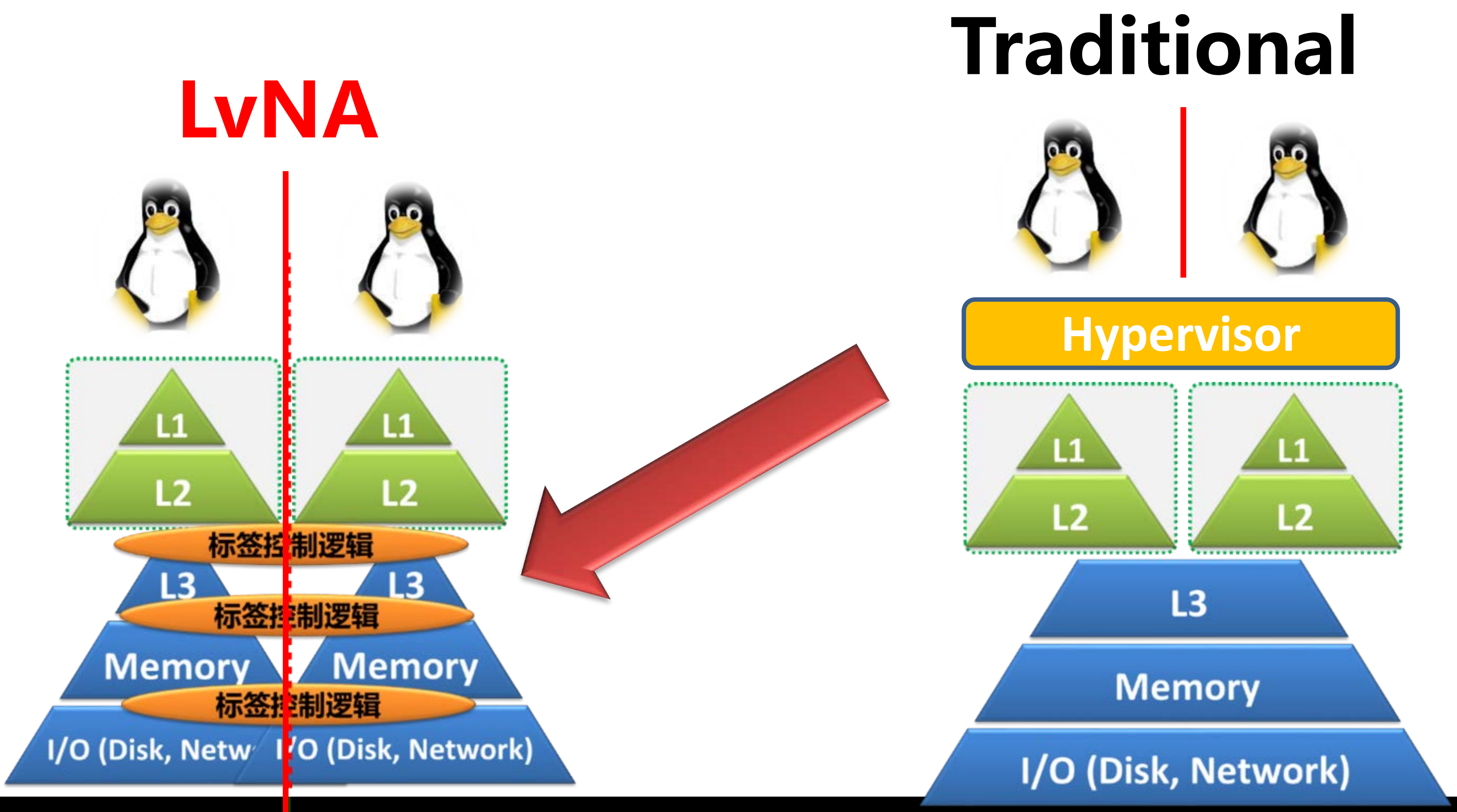
Berkeley's FireSim

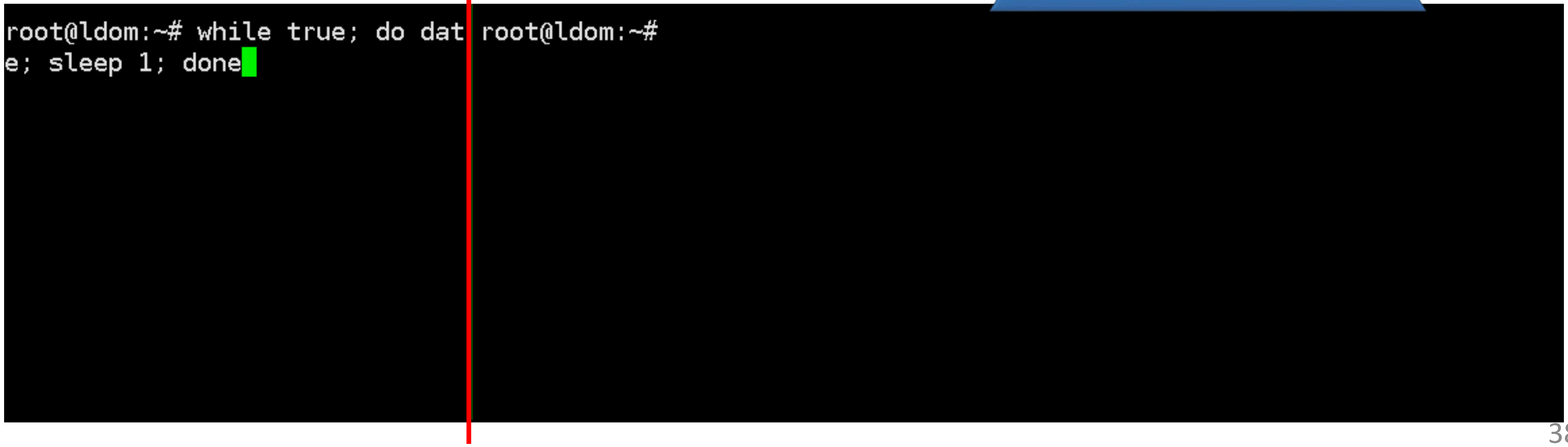
FireSim

Easy-to-use FPGA-accelerated
Cycle-accurate Hardware Simulation

	FireSim	FlameCluster
ISA	RISC-V	**RISC-V**
CPU Core	Rocket/Boom	**Rocket + Labled**
Frequency	Core：150MHz System：3.4Mhz	**Core：100MHz** **System：100MHz**
Platform	Amazon F1 FPGA Cloud	**FPGA Box**
Scale	1-4096 cores	**1-32 cores**
OS	Linux	**Linux**

Demo 1: Label-based Virtualization

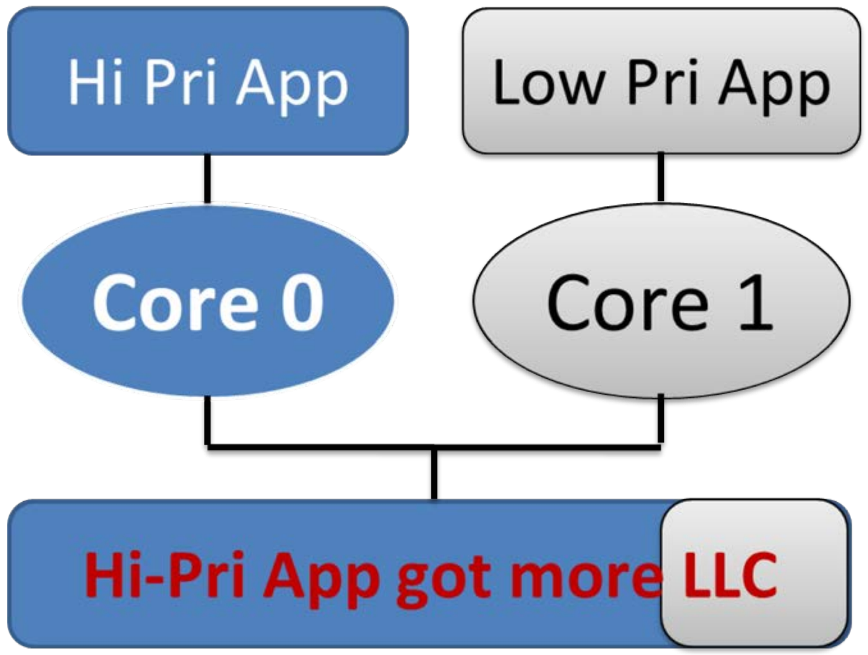




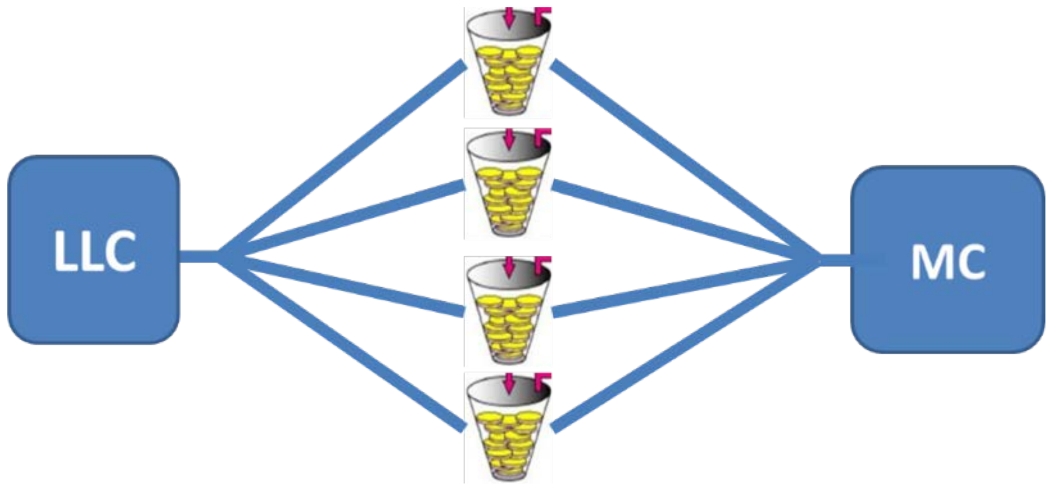
```
root@ldom:~# while true; do dat    root@ldom:~#
e; sleep 1; done
```

Demo 2：Performance Isolation

- **Cache partitioning** based on labels



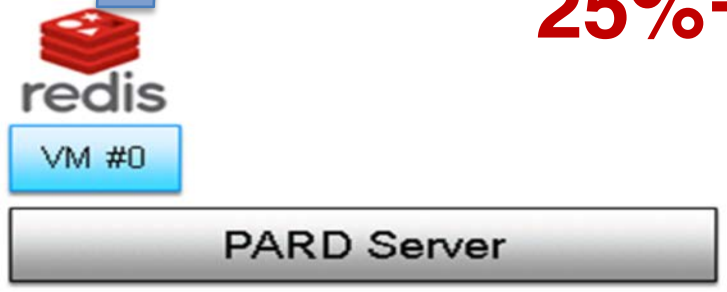
- Control **memory BW** based on labeled token buckets



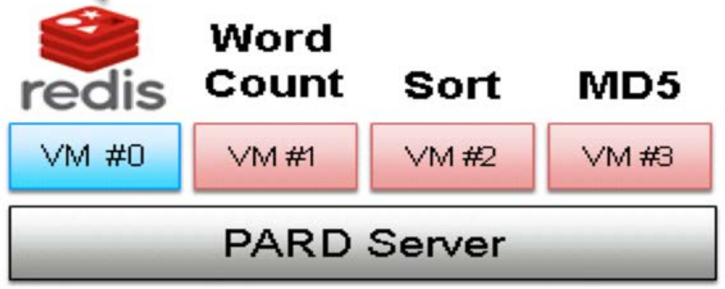
Performance Evaluation

CPU Util: Increase 4X

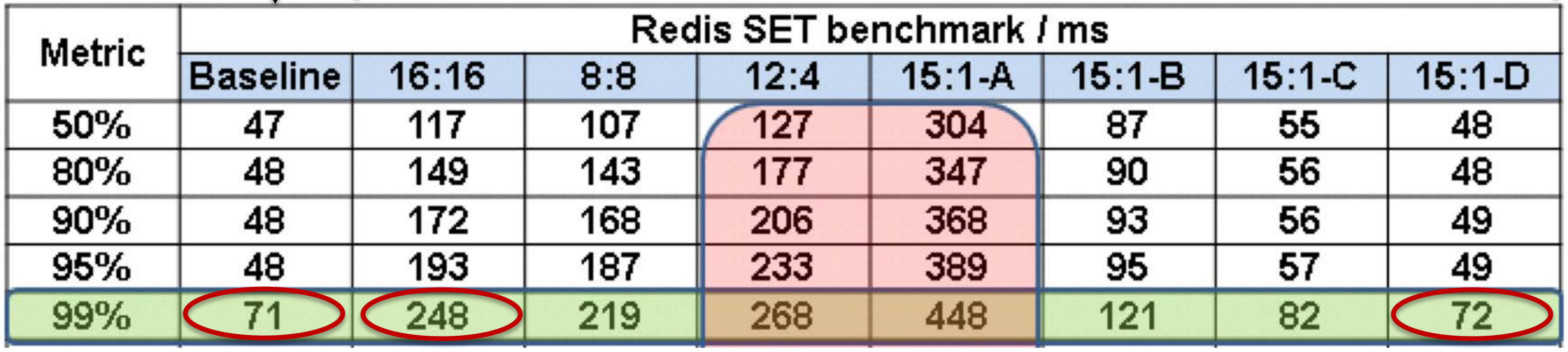
25%→100%



Solo , CPU-25%



Mix , CPU-100%



Metric	Redis SET benchmark / ms							
	Baseline	16:16	8:8	12:4	15:1-A	15:1-B	15:1-C	15:1-D
50%	47	117	107	127	304	87	55	48
80%	48	149	143	177	347	90	56	48
90%	48	172	168	206	368	93	56	49
95%	48	193	187	233	389	95	57	49
99%	71	248	219	268	448	121	82	72

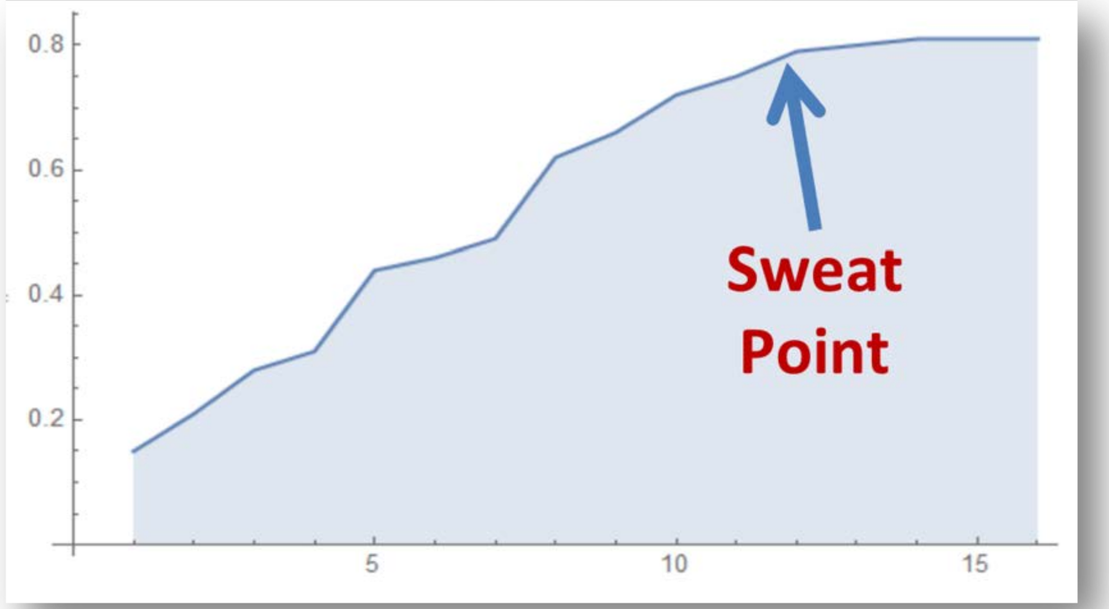
No-Label

Label+ Control

Reduce 3.4X

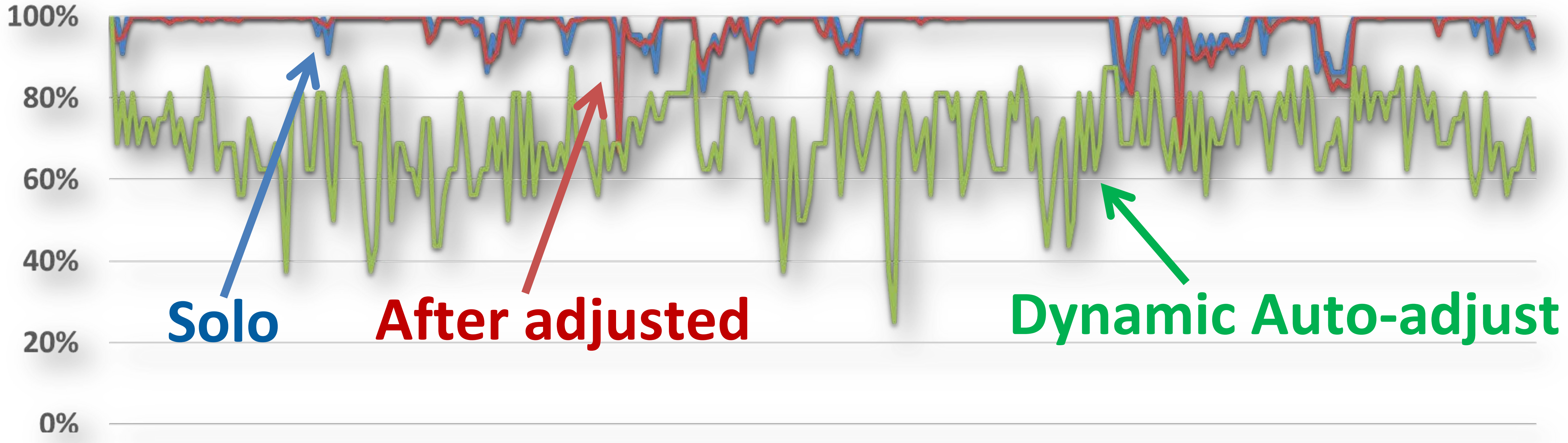
Automated LLC Management

LLC Hit Rate Curve



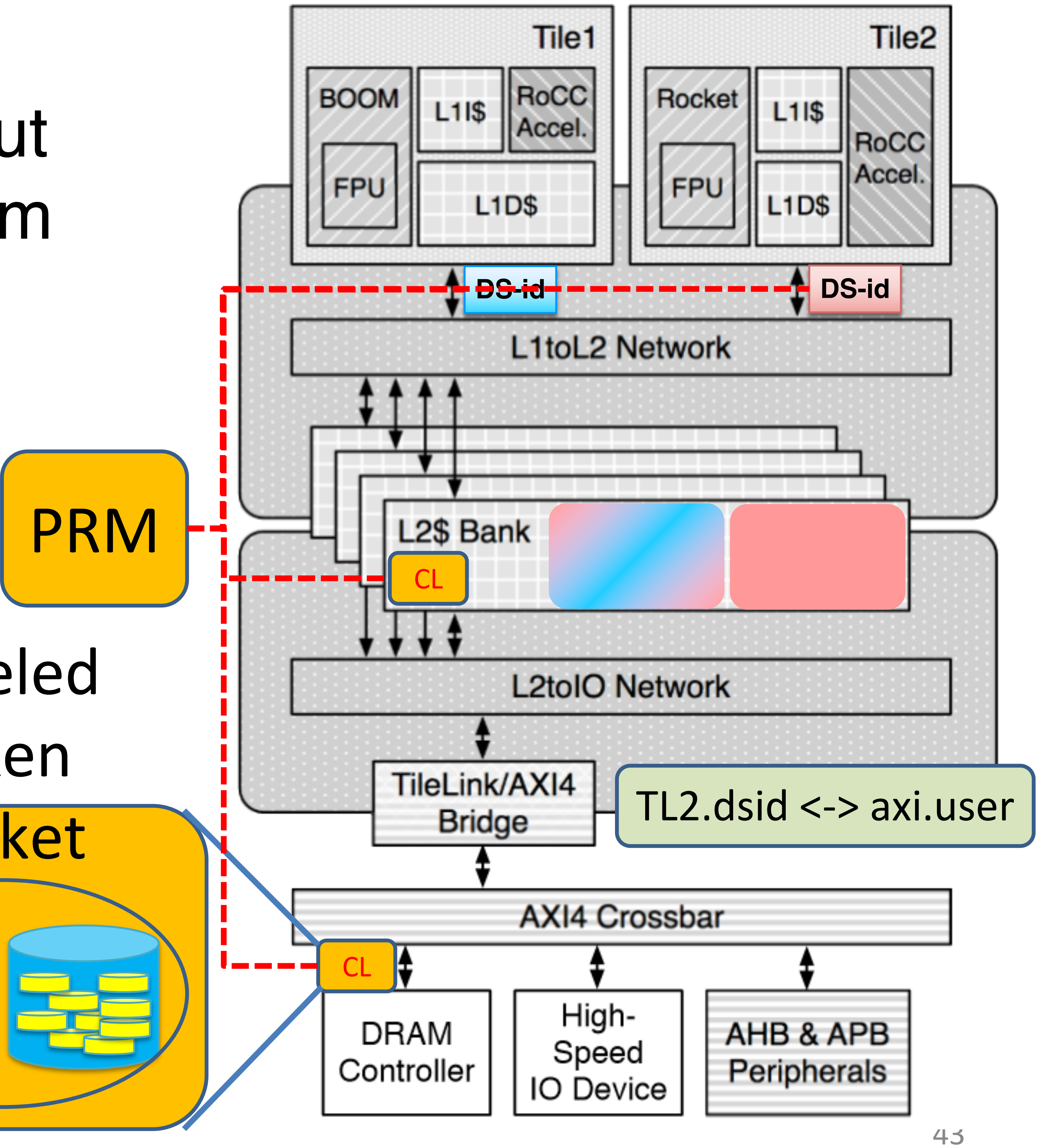
- $\Delta HitRate = \mathbf{0.4\%}$
- Save **~30%** of LLC for Bzip2

Bzip2



Taping out Labeled RISC-V

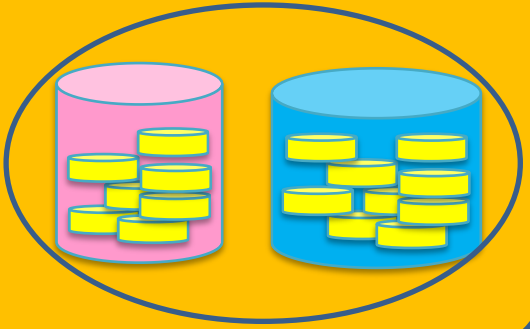
- We have been taping out an SoC with TSMC 40nm technology



Address mapping

Labeled token bucket

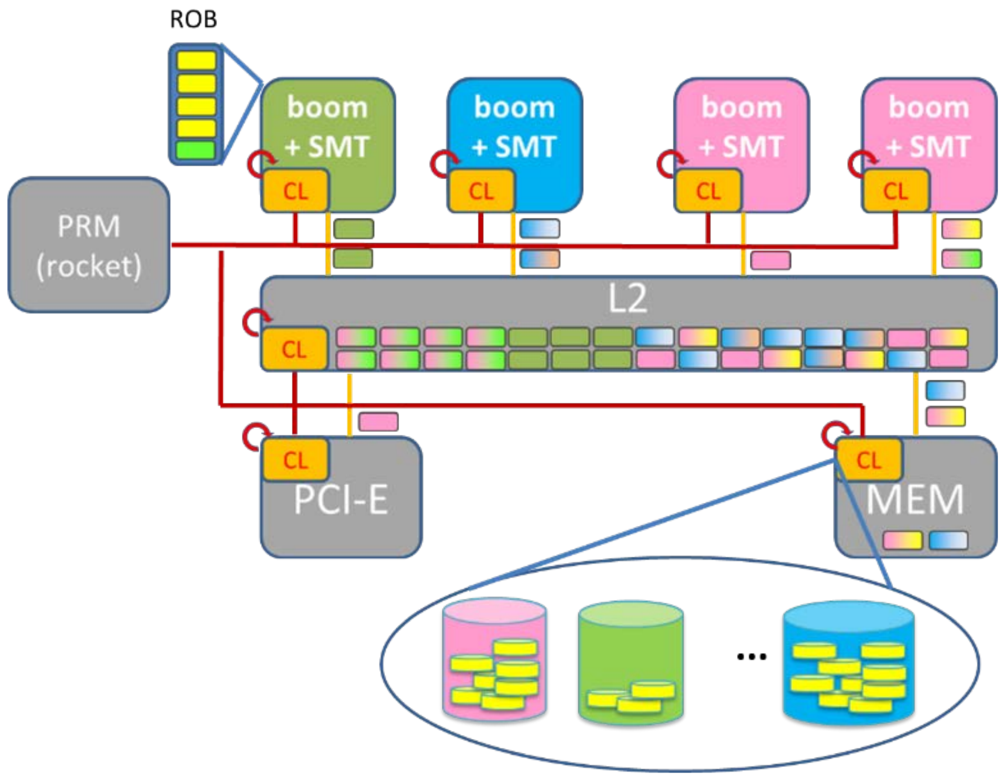
DS-id	Base	Len
1	0x0000	0x4000
2	0x8000	0x8000

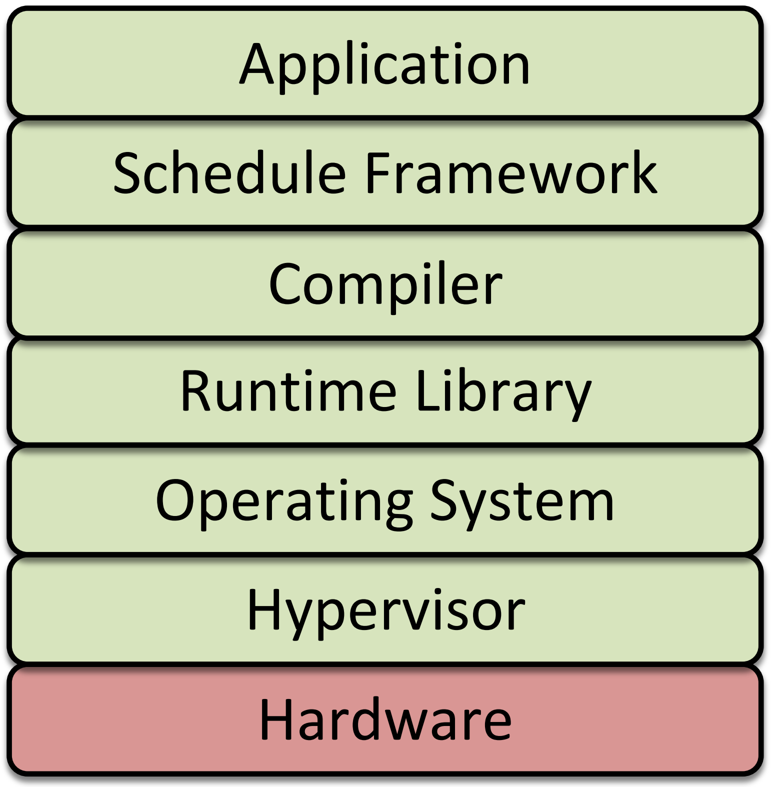


https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf

HW: New Functionality

- **Add new functionality into CL**
  - Encryption/Decryption, Compression, Virtualization, Monitoring, …

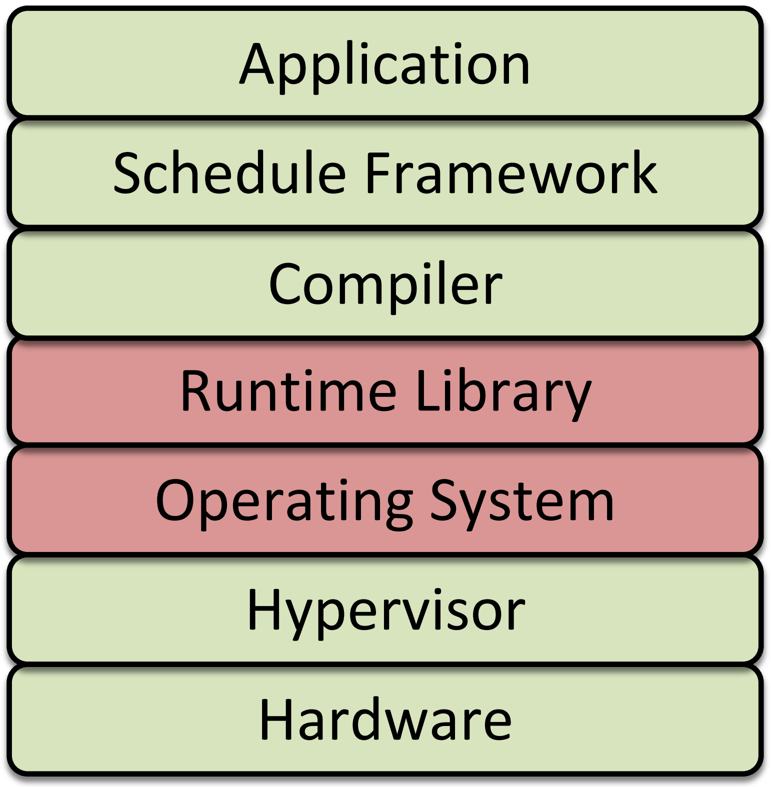


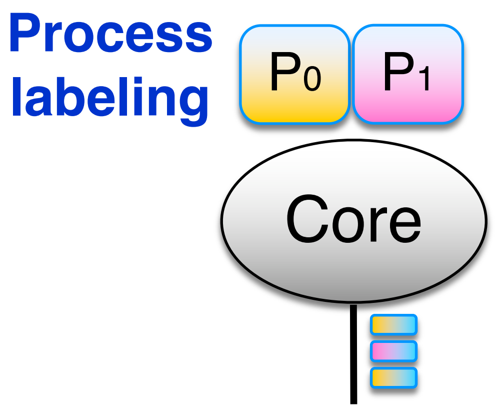


- Prof. Sally McKee’s group is working on security based LvNA

Fine-grained labeling

- Process (w/ Cgroup)
  - Process/container-level **Finished**
  - Thread-level
- Address space
  - Function-level
  - Object-level





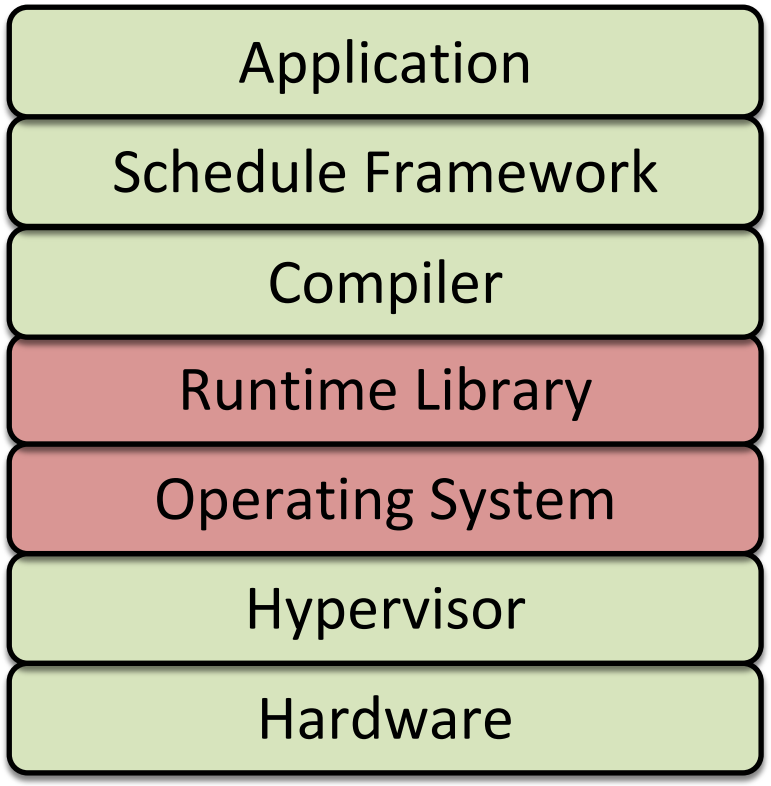
dsid	start	end
1	0x8000	0xffff
3	0x2000	0x27ff

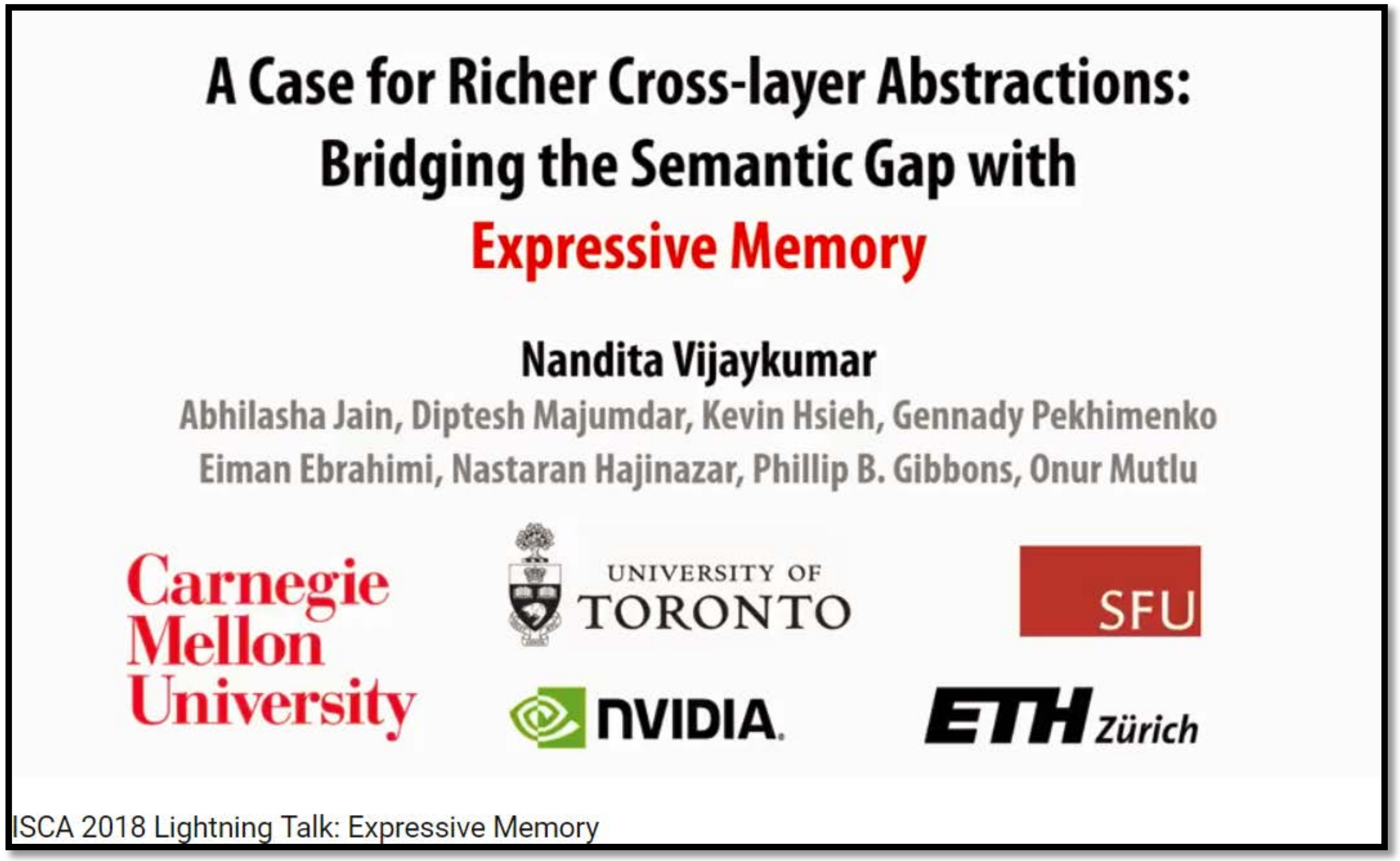
Address space labeling

Fine-grained labeling

- Address space labeling
  - Function-level
  - Object-level

dsid	start	end
1	0x8000	0xffff
3	0x2000	0x27ff





A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory

Nandita Vijaykumar

Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko

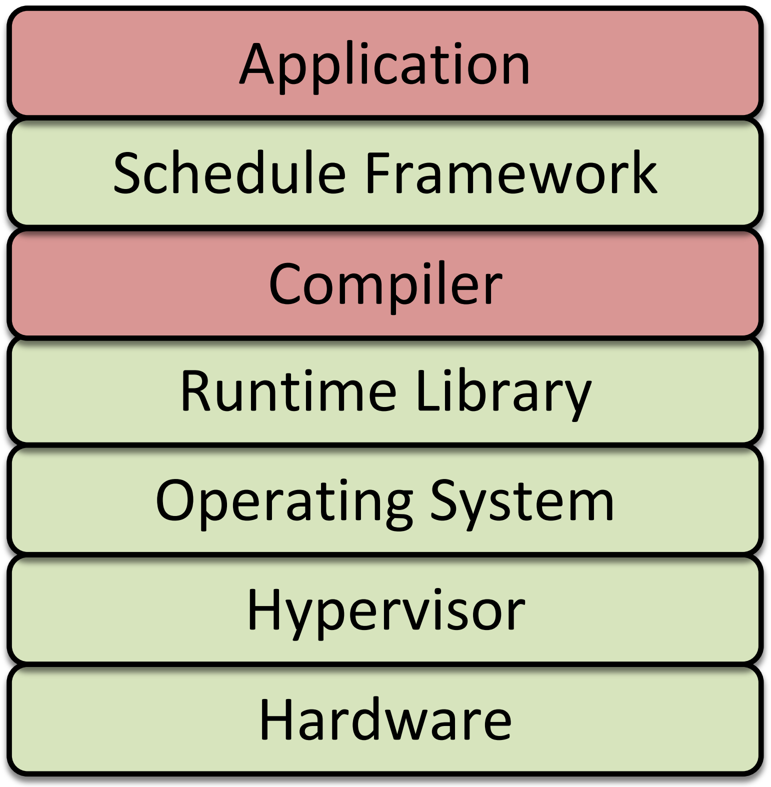
Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons, Onur Mutlu

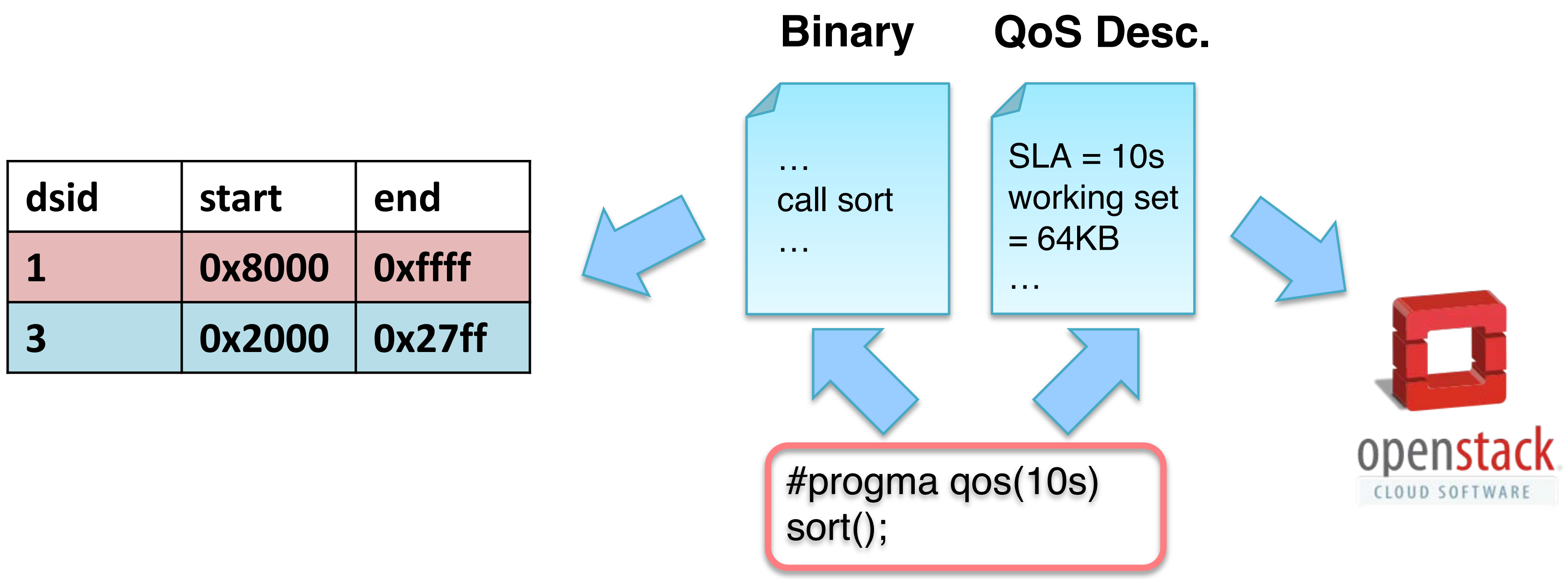
Carnegie Mellon University, University of Toronto, SFU, NVIDIA, ETH Zürich

ISCA 2018 Lightning Talk: Expressive Memory

Programming

- **New programming model for expressing QoS/timing/security requirements**
  - Compilers translate requirements into label-based rules.



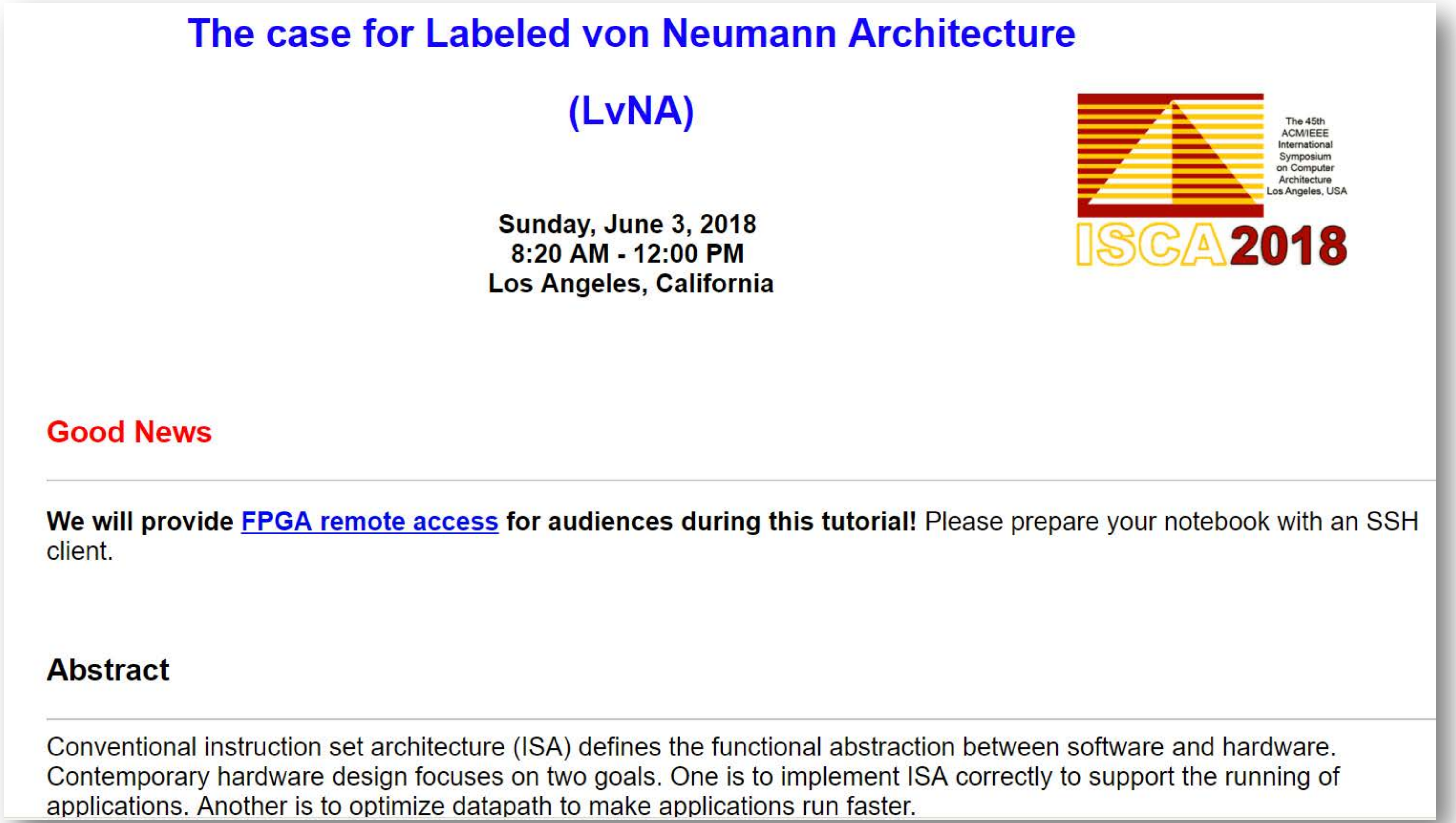


```
#progma qos(10s)
sort();
```

dsid	start	end
1	0x8000	0xffff
3	0x2000	0x27ff

Tutorial @ ISCA 2018

- http://sdc.ict.ac.cn/isca2018-tutorial/



The case for Labeled von Neumann Architecture

(LvNA)

Sunday, June 3, 2018
8:20 AM - 12:00 PM
Los Angeles, California

The 45th ACM/IEEE International Symposium on Computer Architecture Los Angeles, USA

ISCA2018

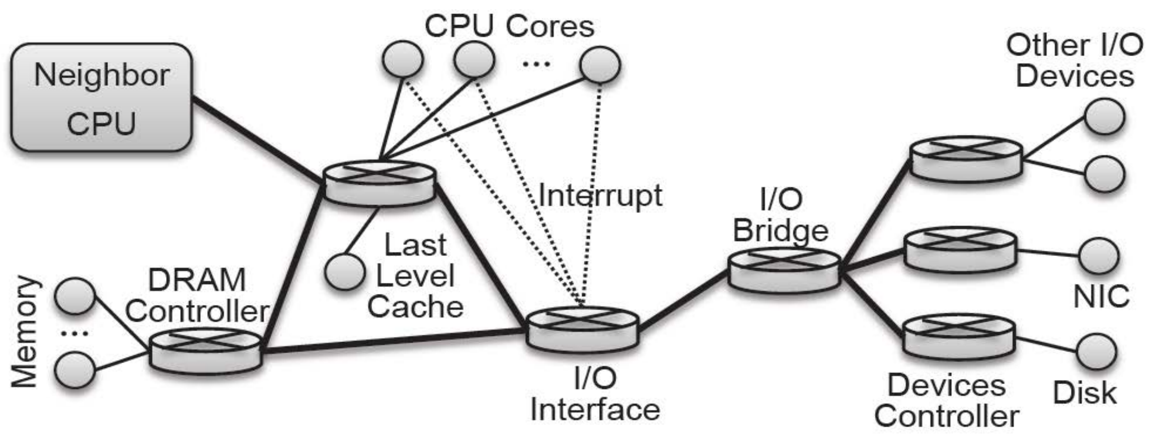
Good News

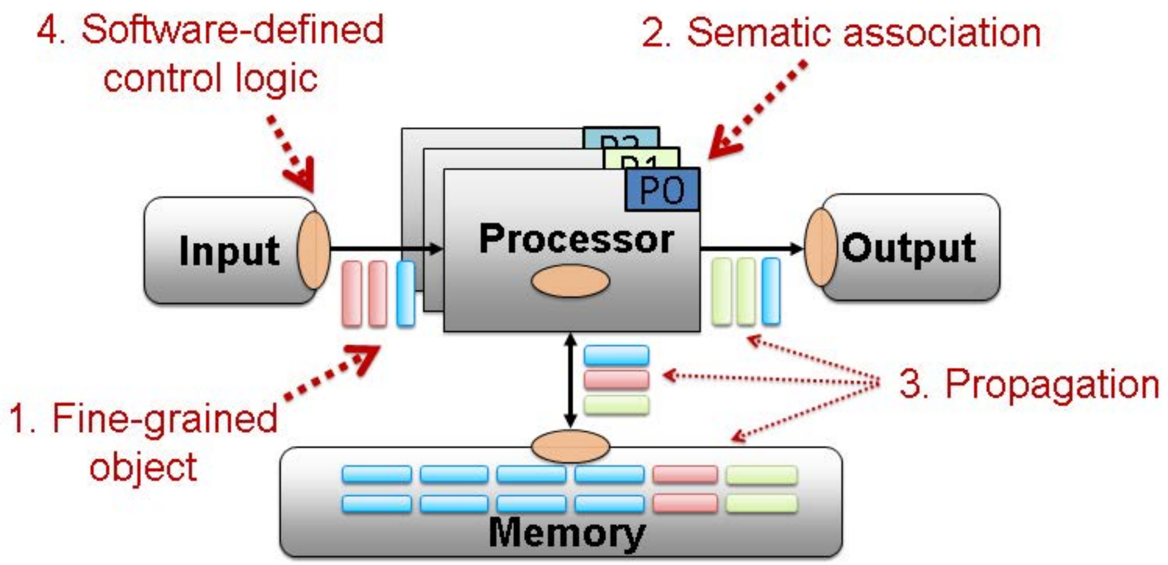
We will provide FPGA remote access for audiences during this tutorial! Please prepare your notebook with an SSH client.

Abstract

Conventional instruction set architecture (ISA) defines the functional abstraction between software and hardware. Contemporary hardware design focuses on two goals. One is to implement ISA correctly to support the running of applications. Another is to optimize datapath to make applications run faster.

Conclusion





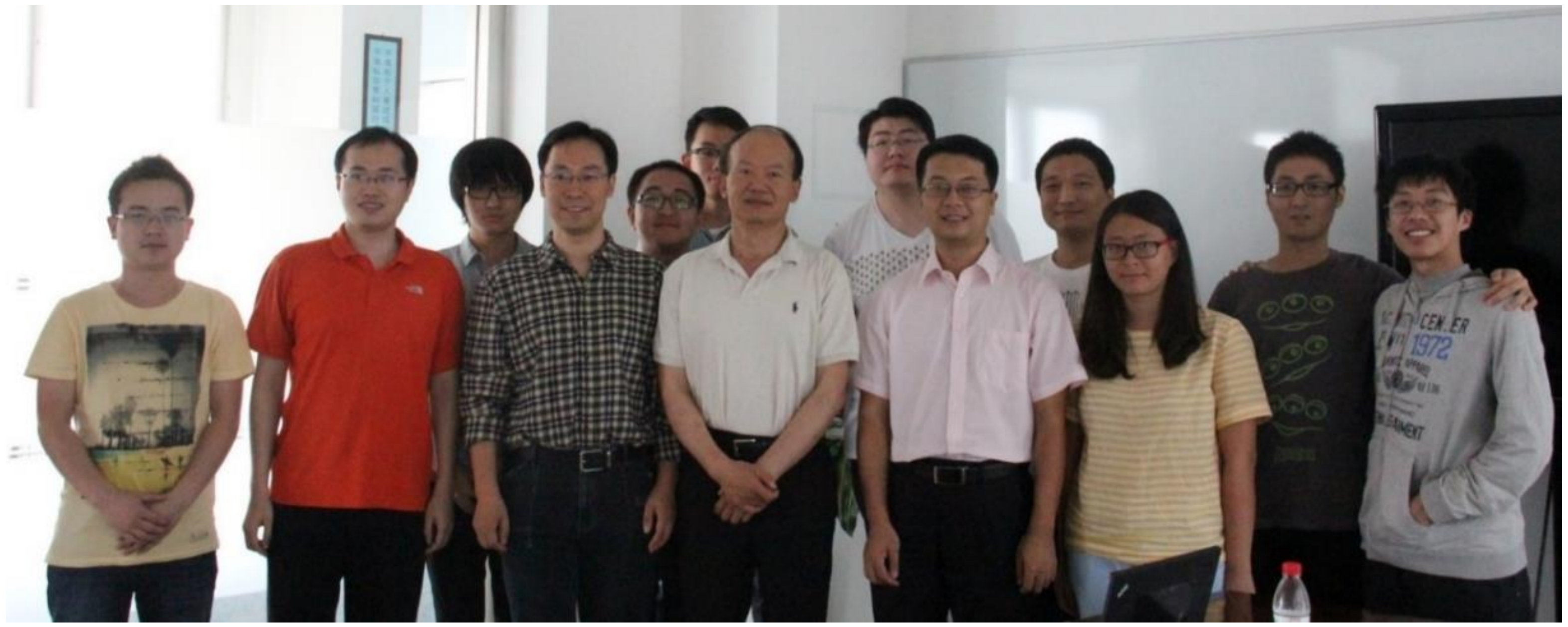
- LvNA: a concept of software-defined architecture
- Labeled RISC-V: an implementation of LvNA

Building Systems is Fun

- **Building systems that will fail**
  "It almost goes without saying that ambitious systems never quite work as expected. Things usually go wrong, sometimes in dramatic ways.“
  -- **Fernando J. Corbató**

- **Building systems that really work**
  "We built an initial prototype, putting in the first 90% of the effort required to create a real system and ... to make INGRES really work.“
  -- **Michael Stonebraker**

Acknowledgement



(2014) from left: Zhicheng Yao, Pin Li, Cong Wang, Yungang Bao, Xusheng Zhan, Jin Xin, Kai Li (visiting), Yupeng Li, Xiufeng Sui, Jiuyue Ma, Tianni Xu, Yupeng Qu, Bowen Huang

(2016) from left: Yungang Bao, Wenjie Li, Zhicheng Yao, Wenbin Lv, Zhiyuan Yan, Han Xue, Tianni Xu, Jin Xin, Yuan Xu, Sa Wang, Zihao Yu, Bowen Huang, Xusheng Zhan

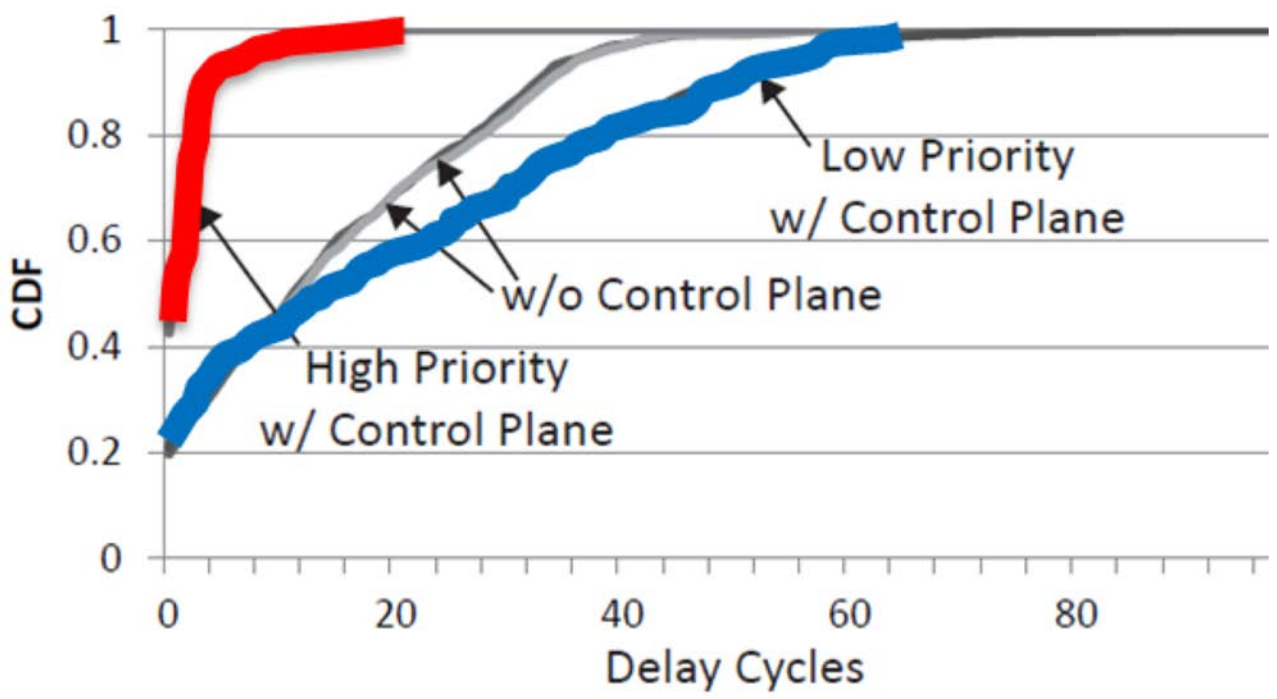
And many others members who are not in the photos, as well as Ninghui Sun, Zhiwei Xu, Lixin Zhang

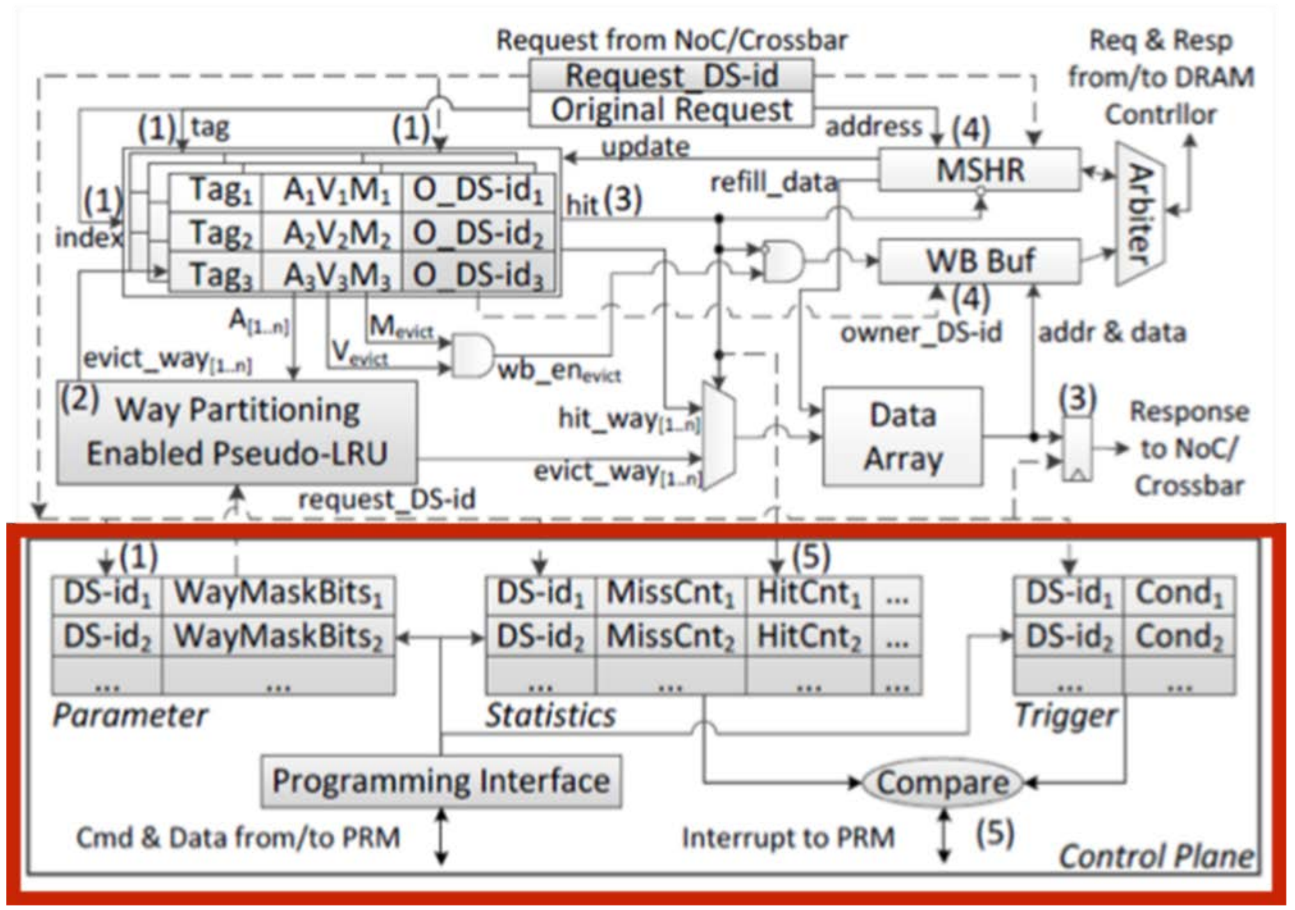
Thanks

Overhead: Latency Analysis

- **Memory controller**: CL reduces queuing delay of high priority requests by **6X**, increases that of low-pri by 33%

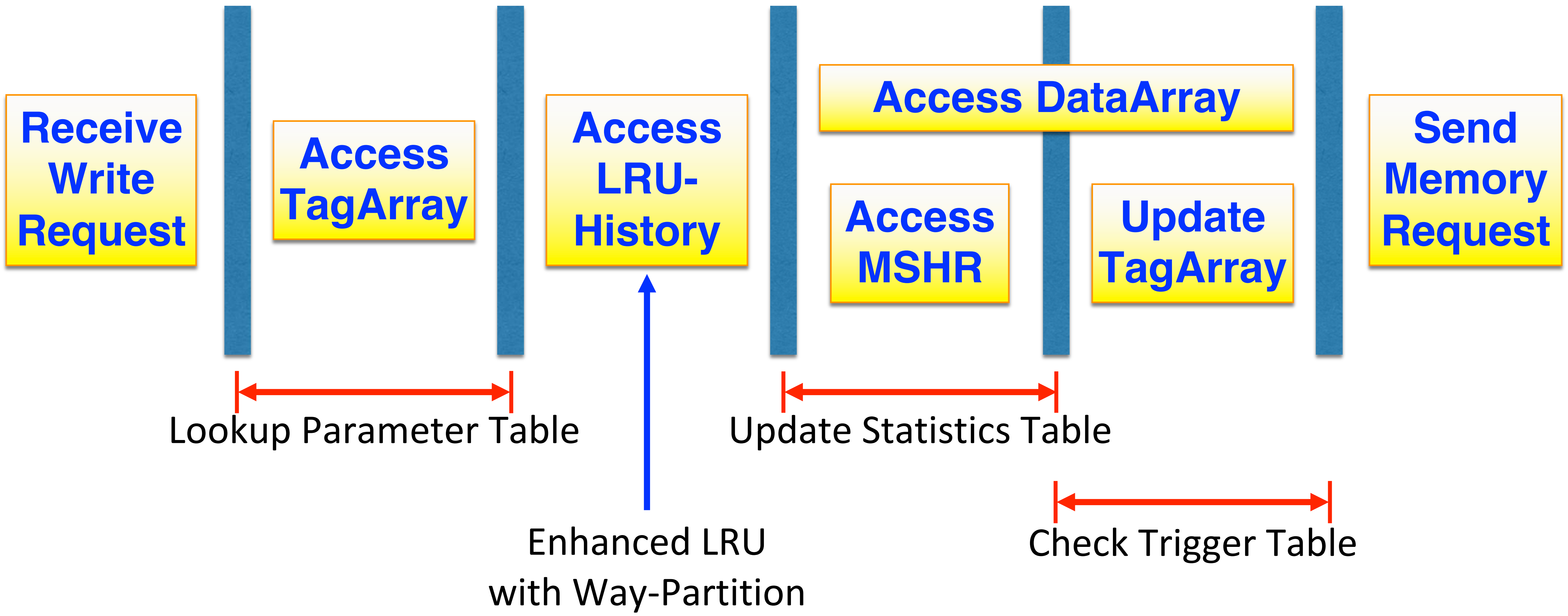
- **Cache**: CL does not add extra latency



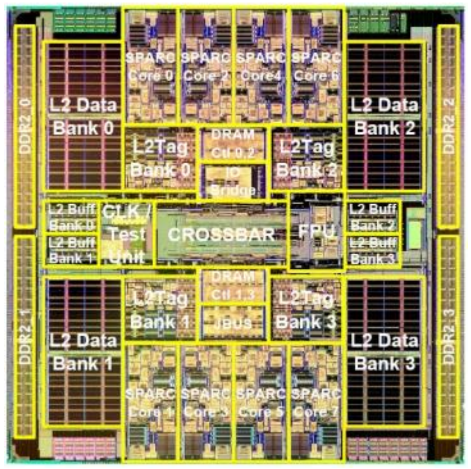


Cache CL latency analysis

- CL operations are hidden in the pipeline



Open Sourced Chips

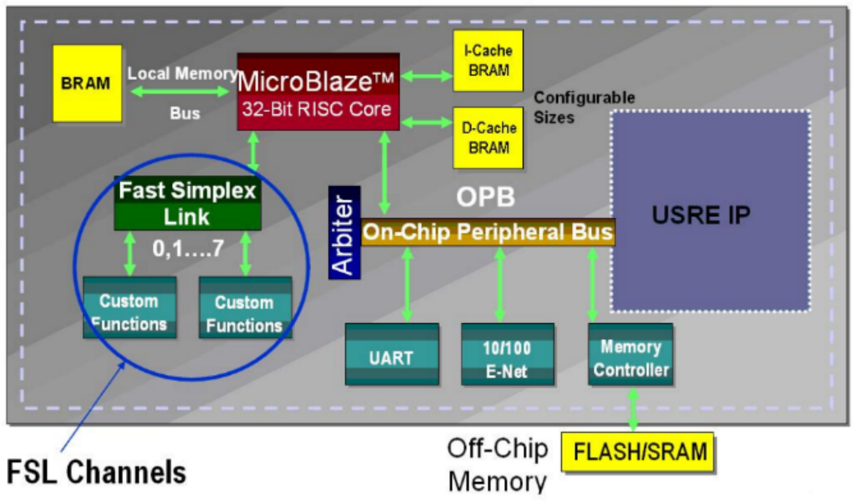


OpenSPARC T1

Open source
Production
COOL

BUT…

Inactive community
Poor SW support
Hard to modify



MicroBlaze

Active community
AXI, IP blocks
HW/SW sloutions
Optimized for Xilinx

BUT…

Not open-sourced
Rely on Xilinx FPGAs

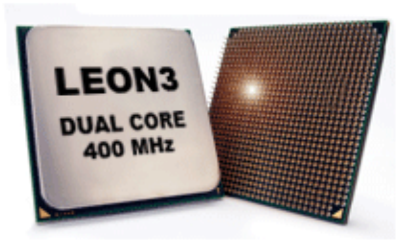
?

OpenPiton

RISC-V

lowRISC

LEON3
Synthesizable
processor



Already Open Sourced

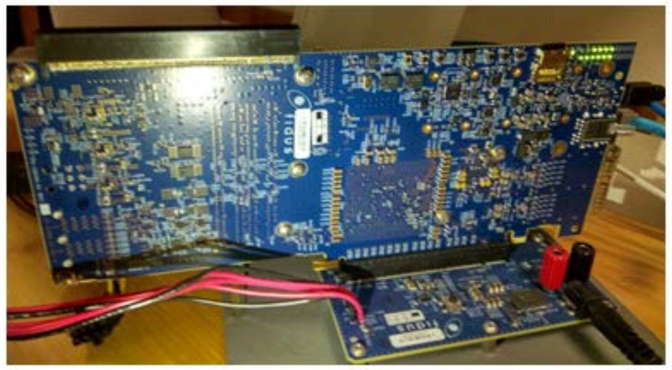
- http://github.com/LvNA-system/labeled-RISC-V

5 Steps to use Labeled RISC-V

1. Get a Xilinx Development Board (VC709, …)
2. Download Labeled RISC-V from github
3. Burn the bit file to FPGA
4. Launch Linux on RISC-V
5. Run applications

Digilent Zedboard

Xilinx ZCU102


Fidus Sidewinder-100

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
```