The Case for Labeled von Neumann Architecture (LvNA)

Yungang Bao

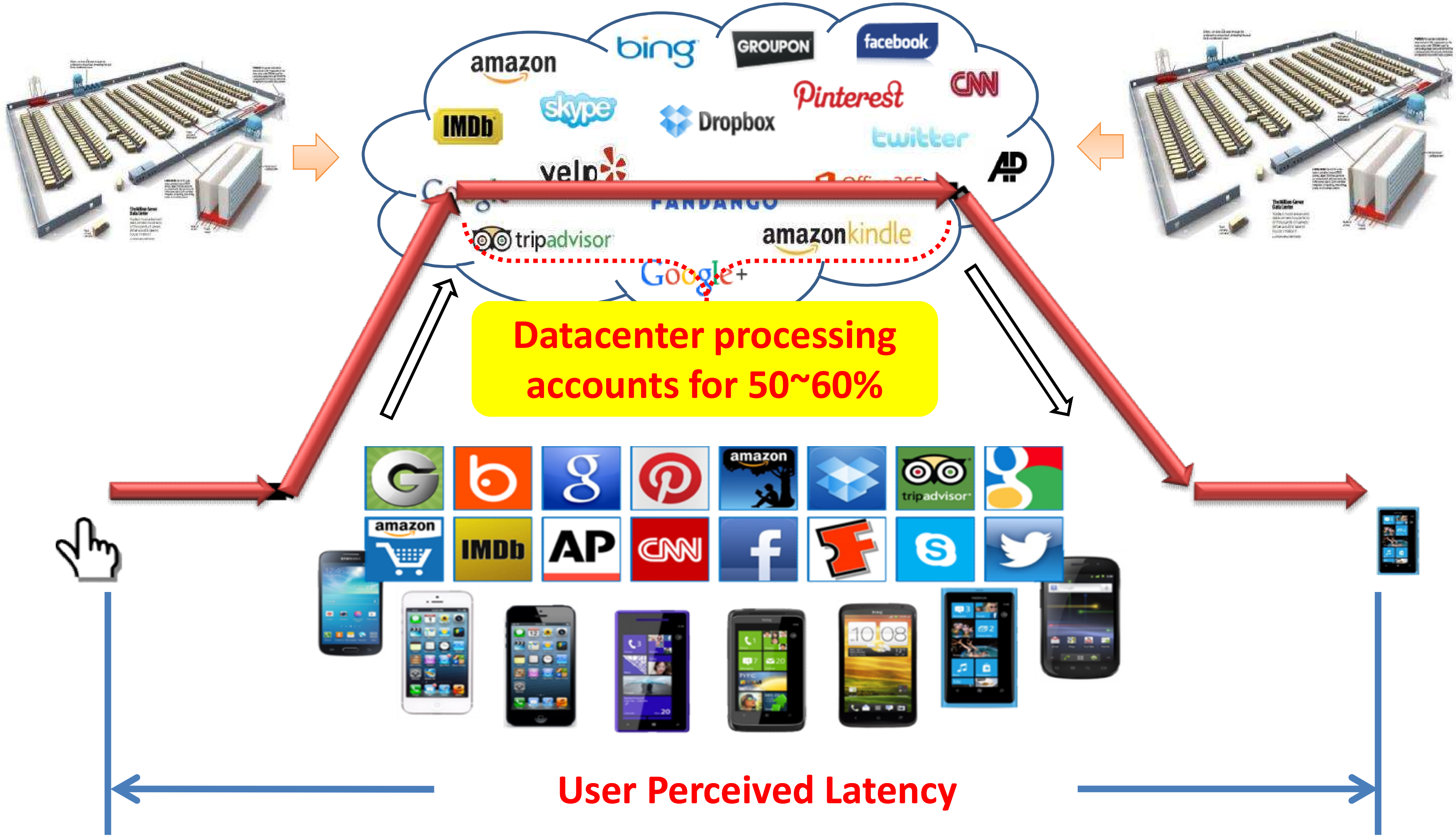
April, 2017

Institute of Computing Technology (ICT),
Chinese Academy of Sciences (CAS)

Agenda

- Background
- Challenges
- Opportunities
- Our Efforts
- Summary

We are in the Cloud Era



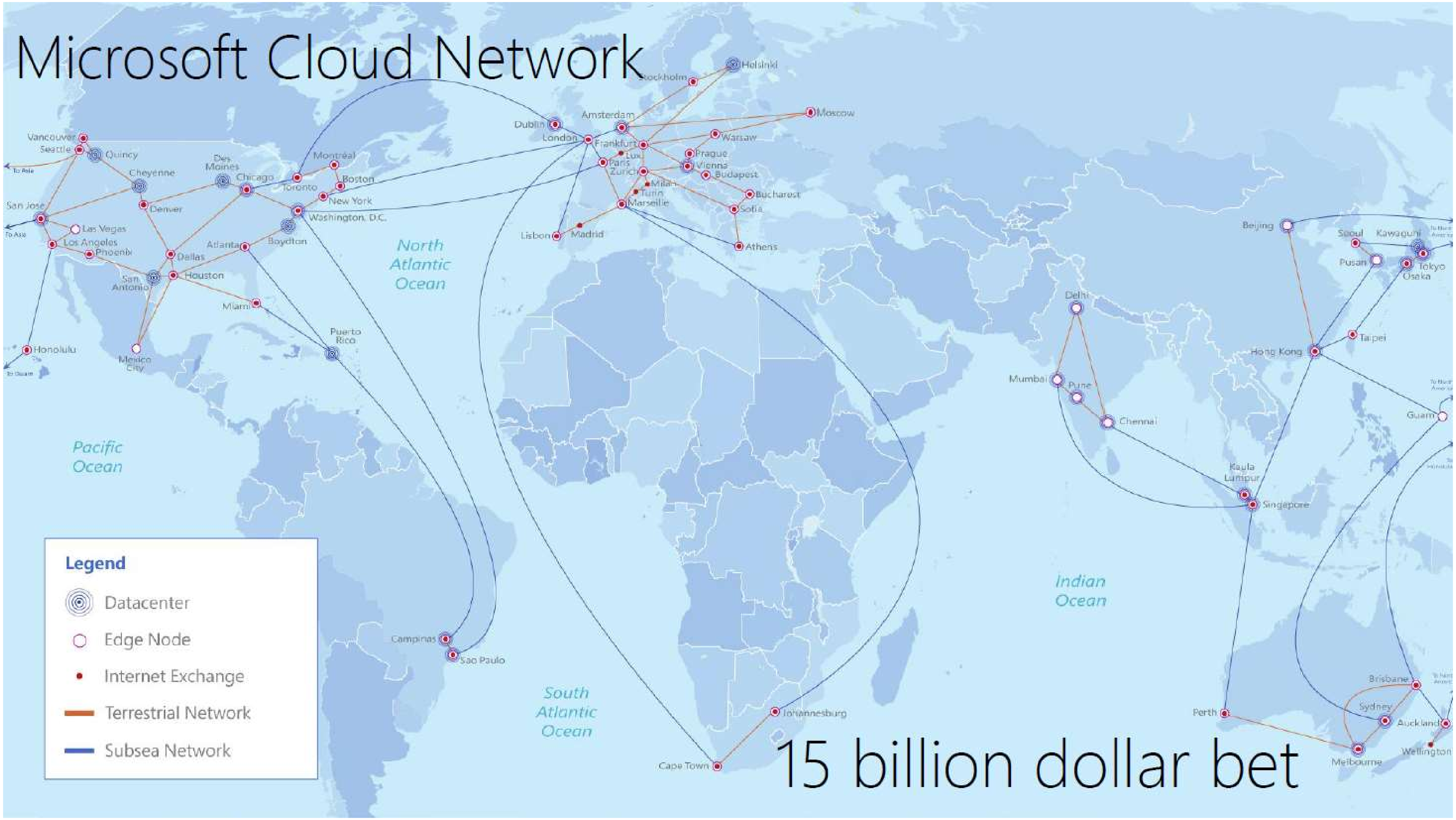
[1] L. Ravindranath et al., Timecard: Controlling User-Perceived Delays in Server-Based Mobile Applications, SOSP, 2013.

Datacenters: The Giant Game

- I claim there really are almost no companies in the world, just a handful, that are really investing in scaled public cloud infrastructure.
- We have something over a million servers in our data center infrastructure. Google is bigger than we are. Amazon is a little bit smaller. ... So the number of companies that really understand the network topology, the data center construction, the server requirements to build this public cloud infrastructure is very, very small.

—Steve Ballmer，Microsoft's former CEO，2013

Microsoft's 15B USD Bet



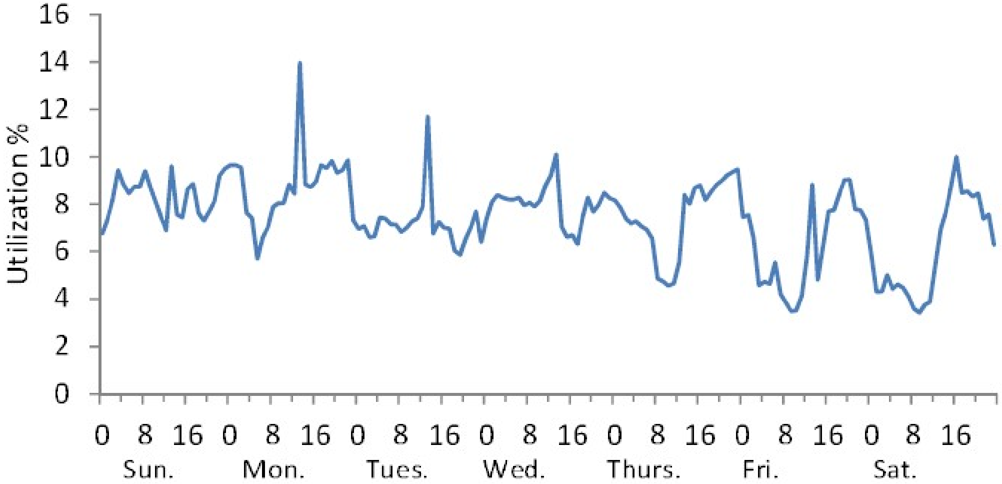
[1] L. Albert Greenberg, SDN for the Cloud, SIGCOMM Keynote, 2015.

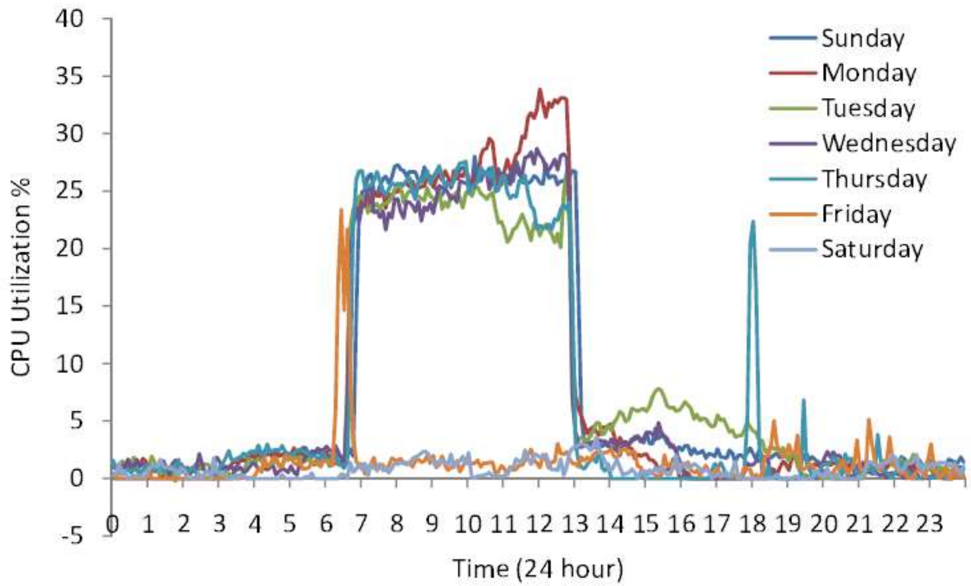
Alibaba's 3B USD Datacenter

[1] "阿里绿色智能数据中心落户张北 将成北方数据心脏", 阿里云资讯, 2016.

Utilization is LOW

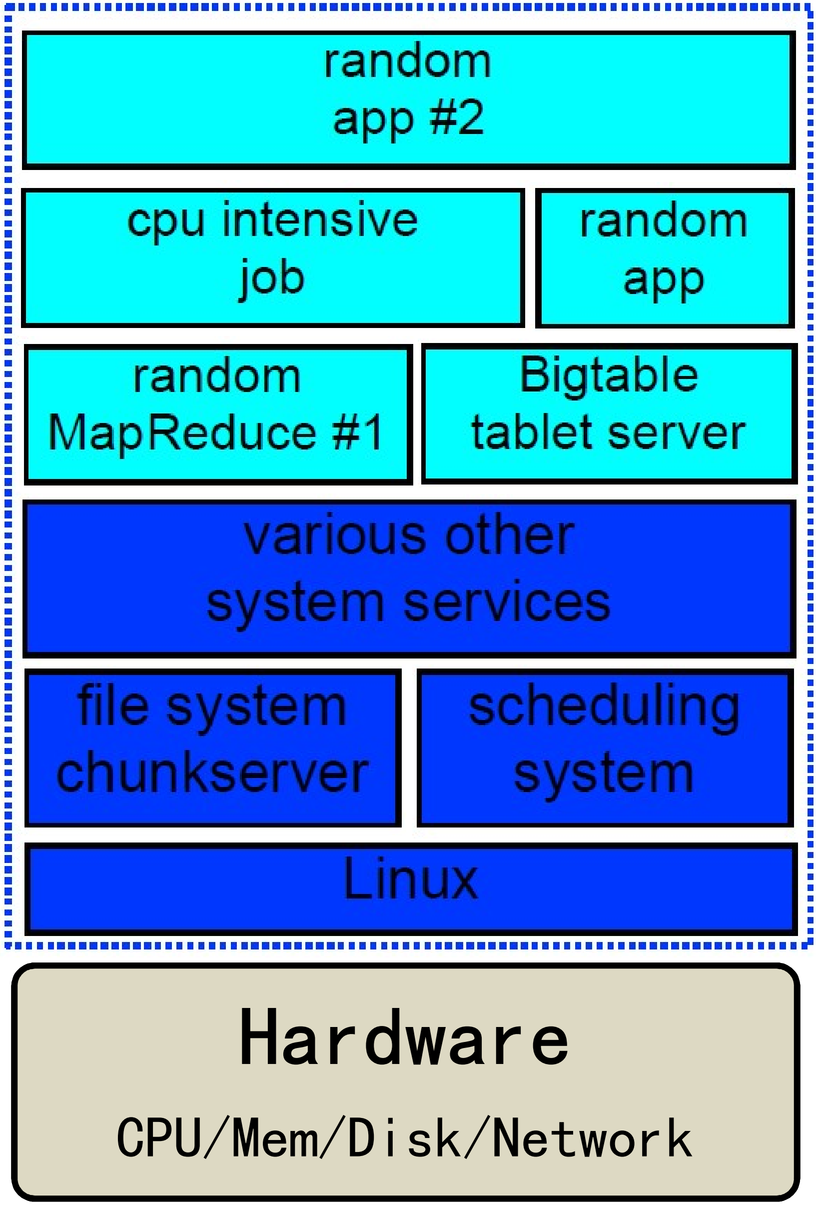
- Survey of Gartner/McKinsey[1,2]: **6%~12%**
- Amazon AWS Average CPU Utilization[3] : **7%~17%**

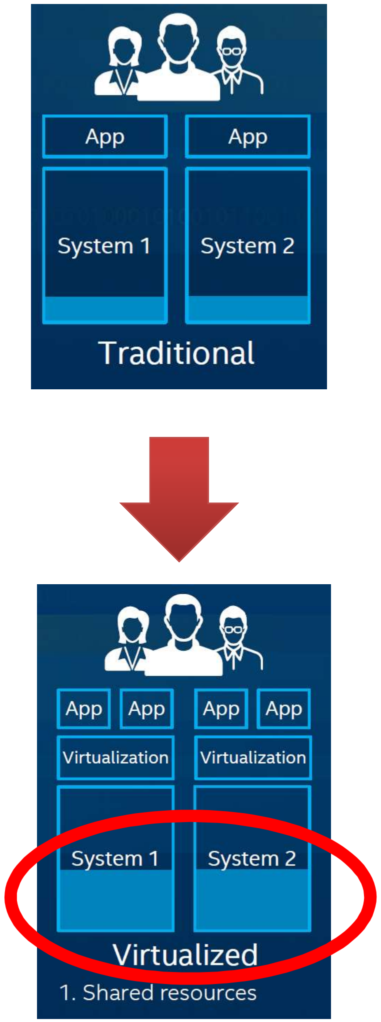




[1] http://www.gartner.com/newsroom/id/1472714.
[2] J. M. Kaplan, W. Forrest, and N. Kindler. Revolutionizing data center energy efficiency. McKinsey & Company, 2008.
[3] Huan Liu, A Measurement Study of Server Utilization in Public Clouds, 2011.

Sharing improves utilizations





Google's Solutions

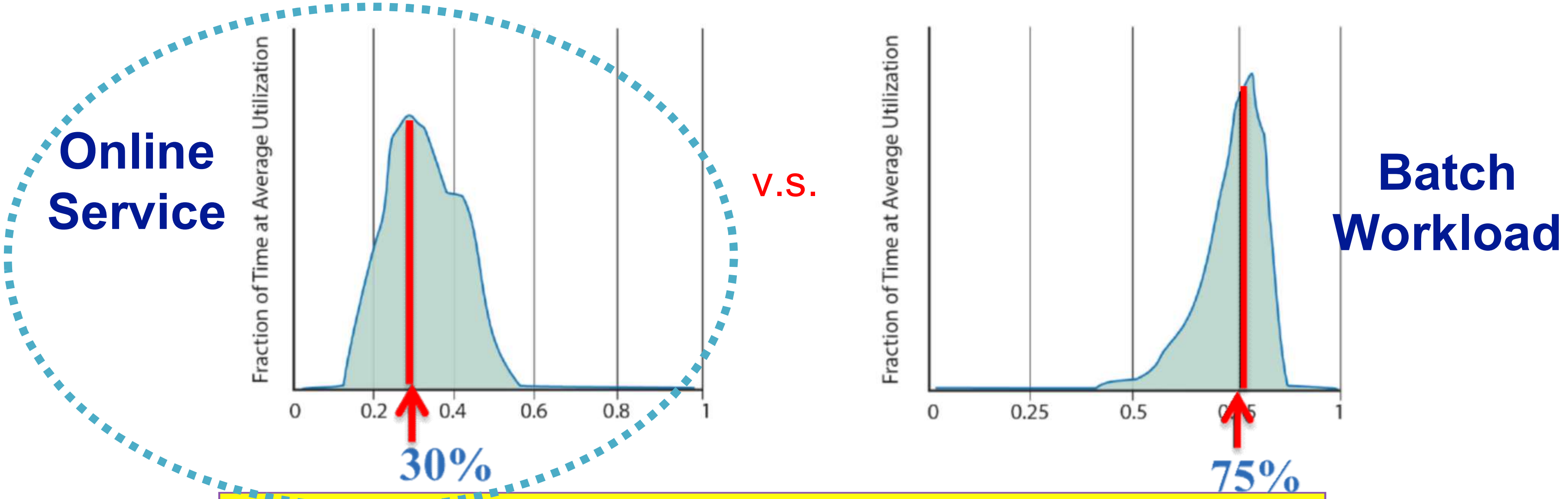
- Batch-Workload Data Center
  - Highly shared

\+

Software Optimization

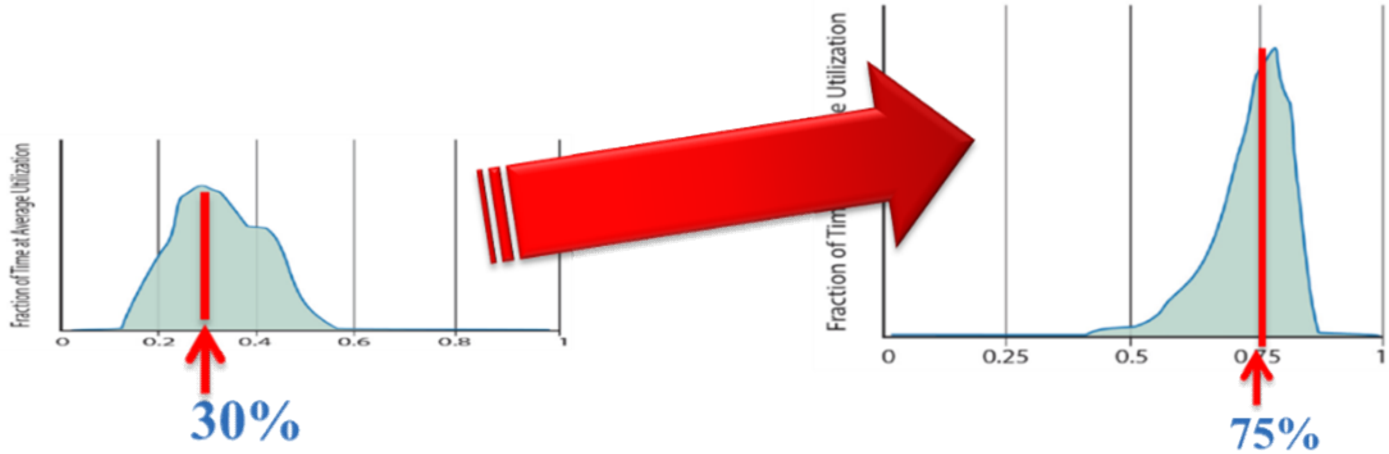
Borg, cgroup,
backup request
LXC, priority,
sync-backup-tasks

Google Datacenters Utilization: (Jan-Mar, 2013)[1]



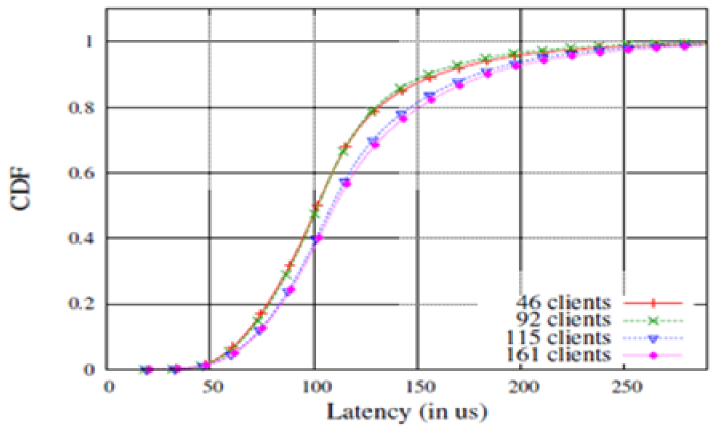
[1] L. Barrosa, J. Clidaras, U. Holzle, The Datacenter as a Computer (2nd Edition), July, 2013.

Why not increase to 75%?

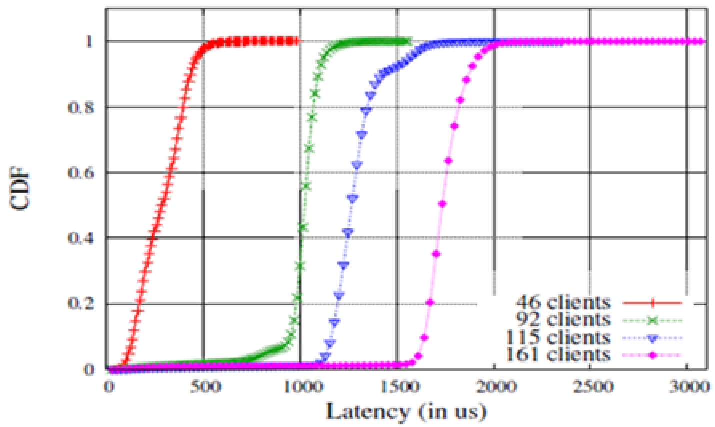


An example: Memcached

- CPU: 30% ➔ 70%
- Response time **>10X** , user experience ↓



CPU: 30%



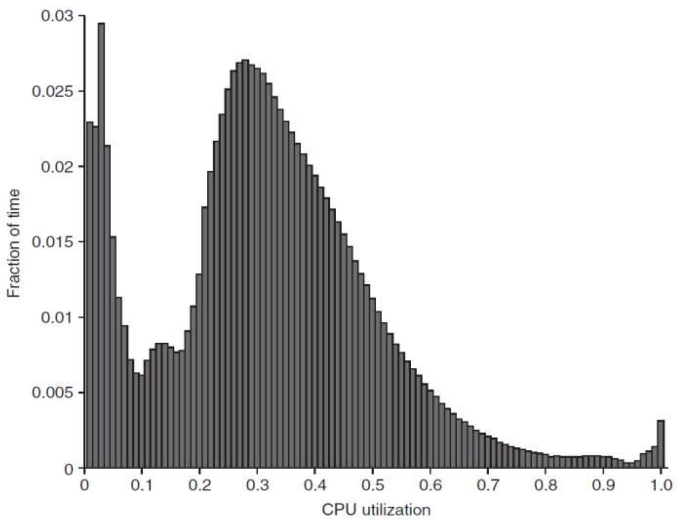
CPU: 70%

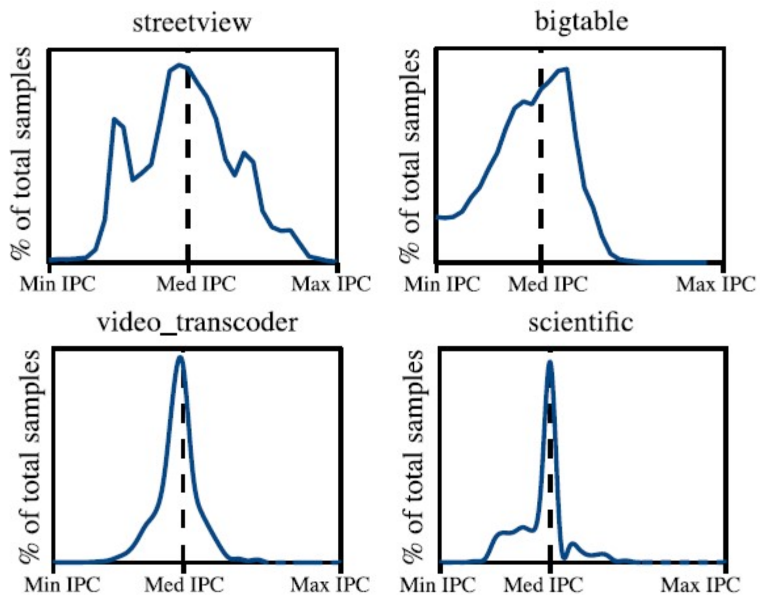
Challenges

- A Tradeoff between

Resource Utilization

User Experience





Response Time is Money

- Search Response time

0.4s→0.9s

- Ad revenue

reduces by 20%

Google's Marissa Mayer: Speed wins

Summary: *Marissa Mayer of Google gave a testimonial to speed. Her key insight for the crowd at the Web 2.*

By Dan Farber for Between the Lines | November 9, 2006 -- 15:01 GMT (07:01 PST)

Follow @ZDNet

Marissa Mayer of Google gave a testimonial to speed. Her key insight for the crowd at the Web 2.0 Summit is that "slow and steady doesn't win the race." Speed is a huge component and big market driver of Web 2.0, she said.

In testing out the user interface for Google search, Mayer said that with more results for a query, users were spending less time on the site. It turned out that the cause wasn't just the paradox of choice--paralyzed by too many choices--but the fact that a page with 10 results was half a second faster that the page with 30 results. So, Google set about making the page with more results faster, and the rest is history.

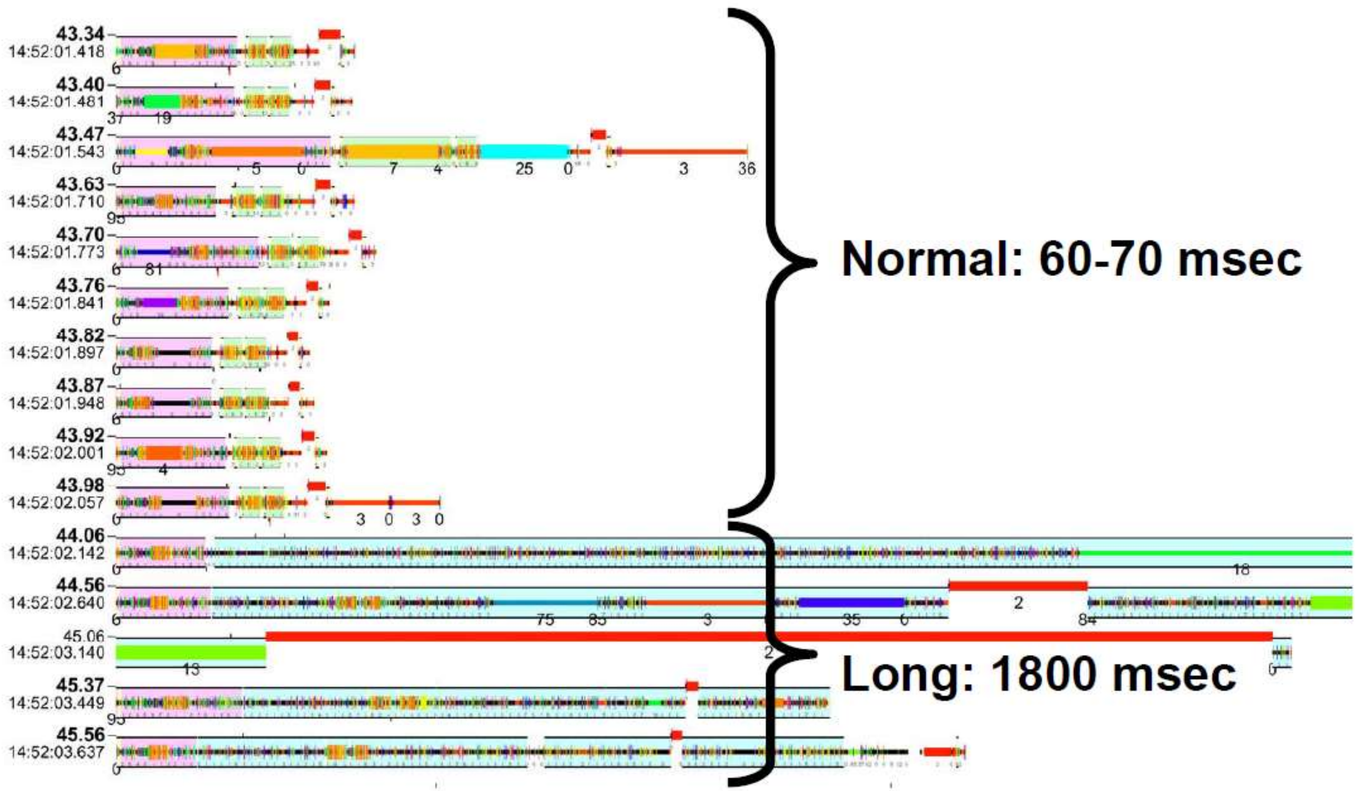
Google's Efforts in Software Stack

Borg,
Linux Container,
Cgroups,
Backup Requests,
Priority,
Sync-back-tasks,
......

[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.

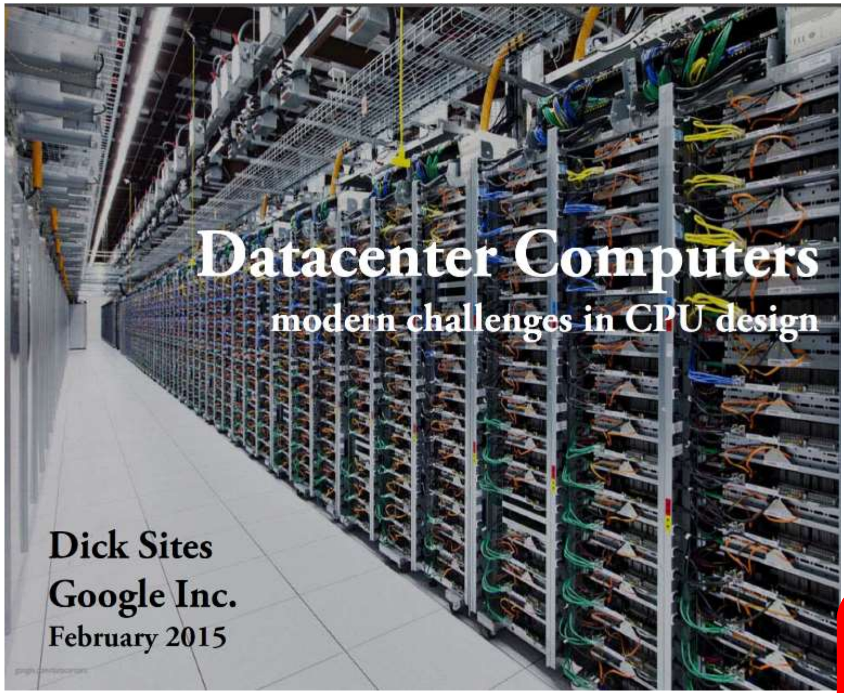
Long Tail Latency

- **Average latency of most requests is 60-70ms, but the tail latency can be 1800ms (~30X)**



D. Sites, "Datacenter Computers modern challenges in CPU design", Google Inc, Feb. 2015.

More Hardware Support Needed



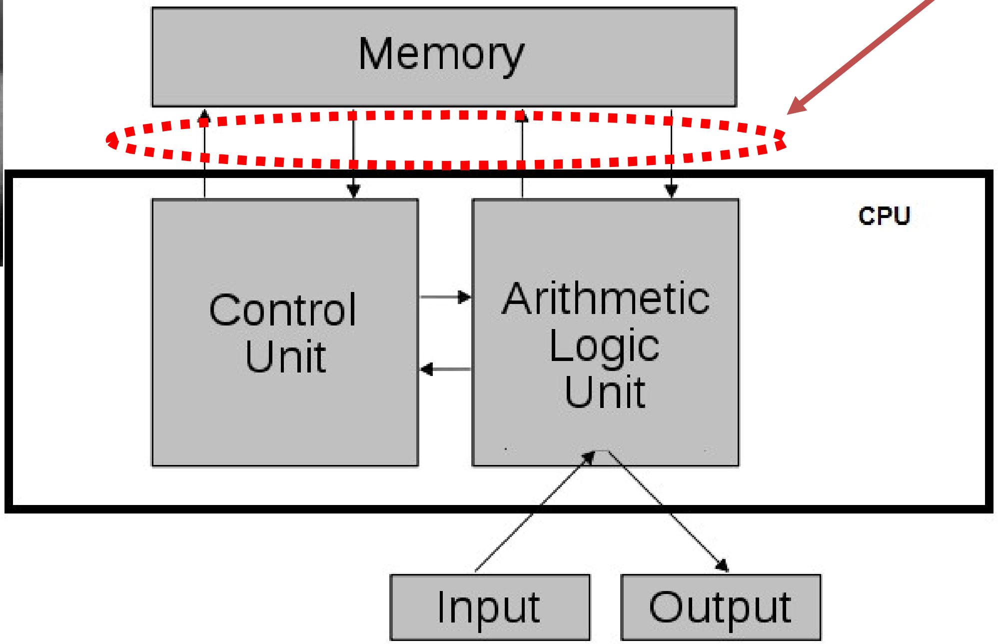
Modern challenges in CPU design

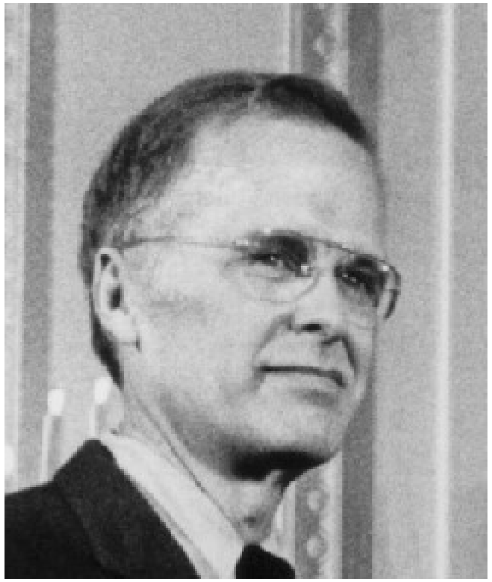
- Isolating programs from each other on a shared server is hard
- As an industry, we do it poorly
  - Shared CPU scheduling
  - Shared caches
  - Shared network links
  - Shared disks
- More hardware support needed
- More innovation needed

von Neumann Bottleneck

John
von Neumann

Von Neumann
Bottleneck

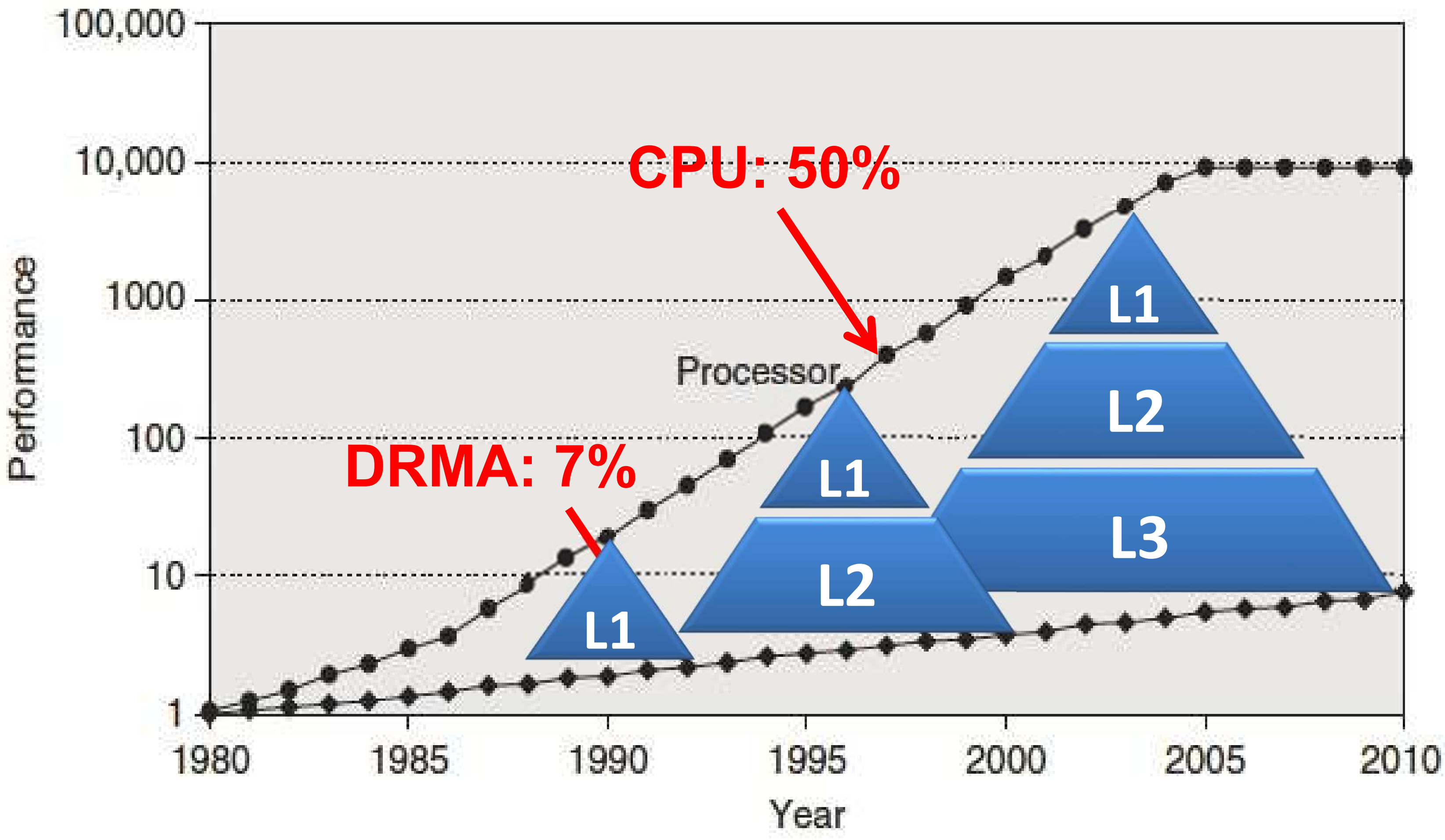




John Backus

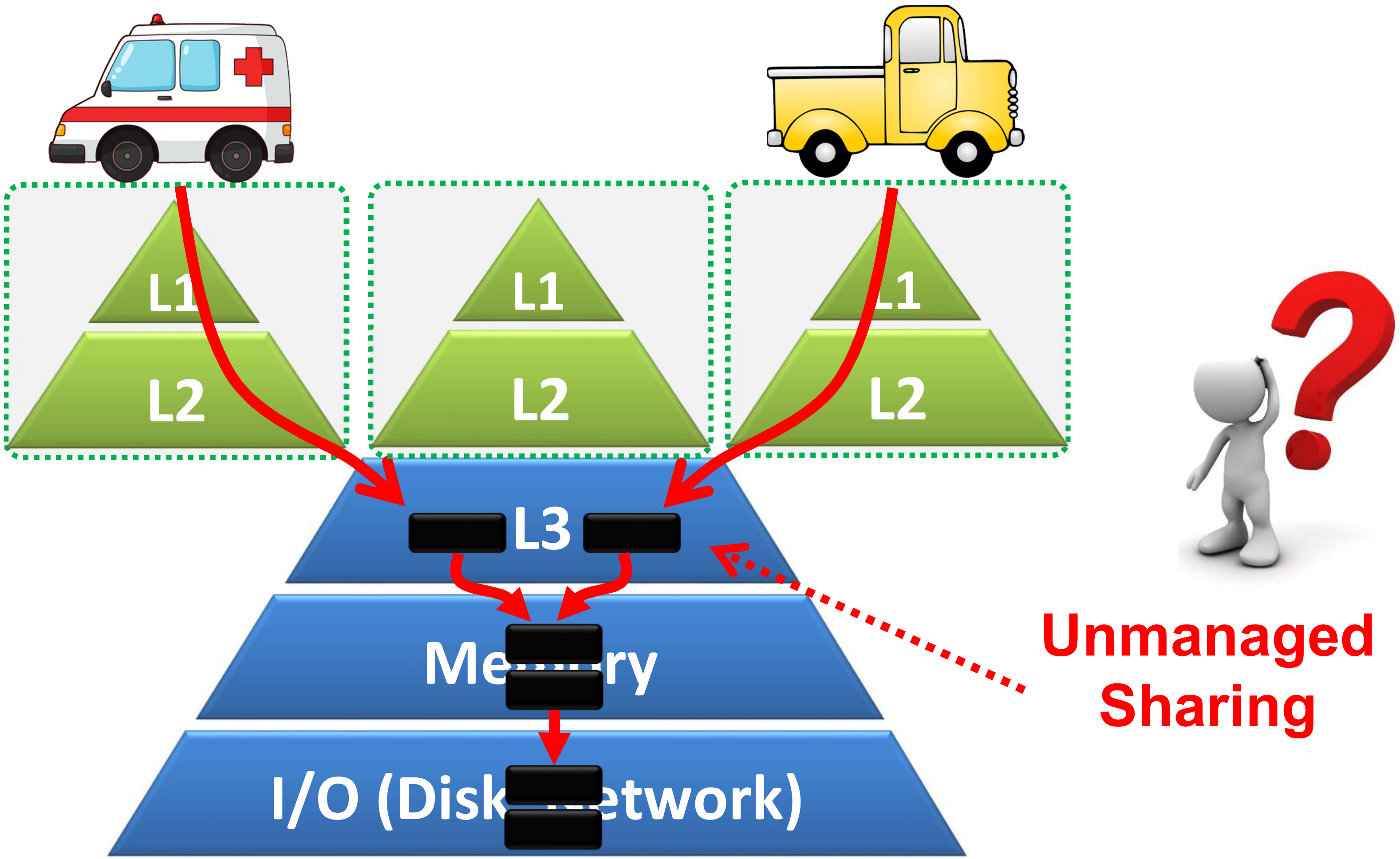
CPU-Memory Gap

- Memory Wall
- Increase memory hierarchy



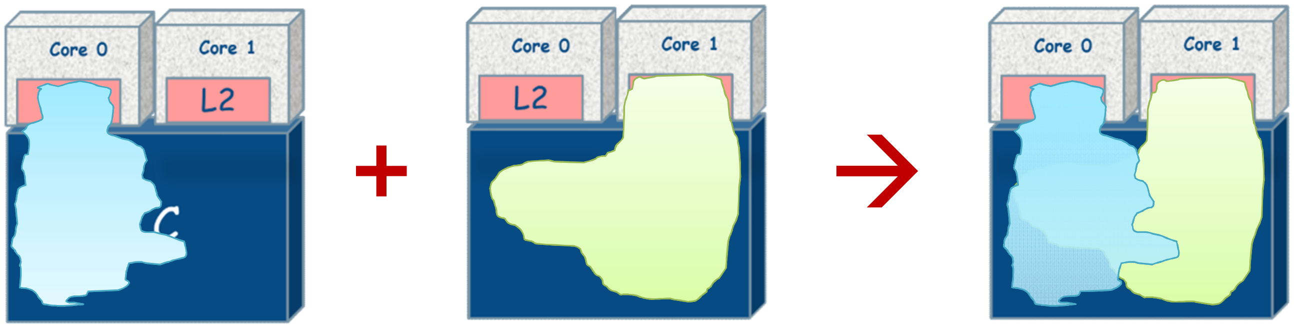
Memory Hierarchy

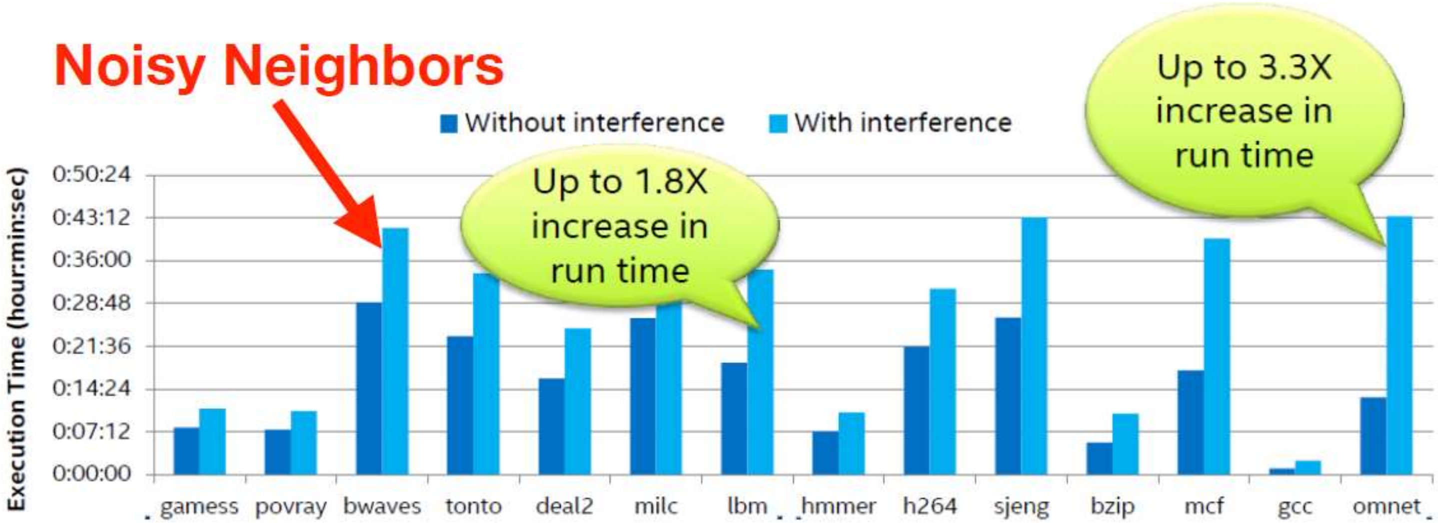
- On-Core vs. Un-Core



Sharing -> Interference

- Cache sharing causes performance degradation

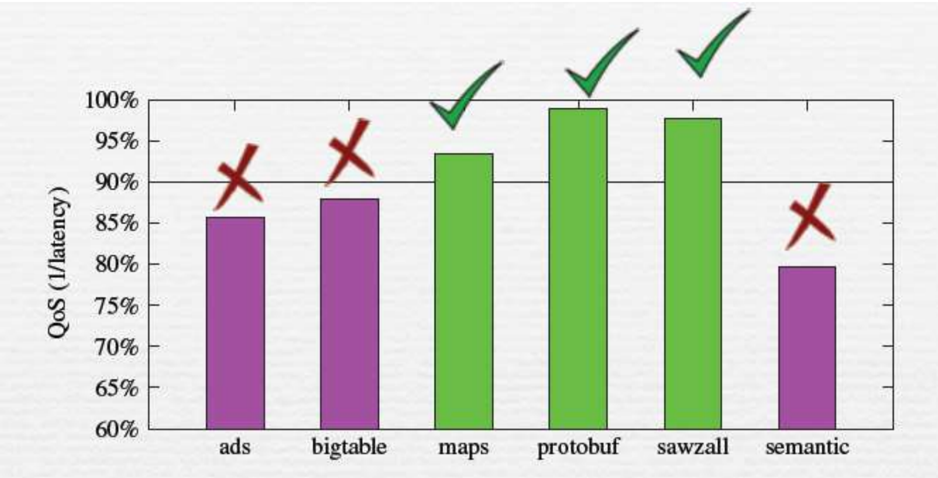




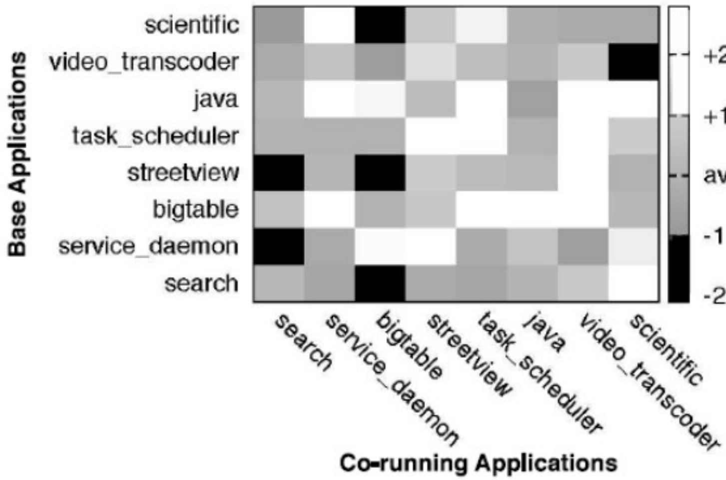
Christine Wang. Intel® Xeon® Processor E5-2600 v3 Product Family Performance & Platform Solutions. 2014.

The sharing problem in Google

- **Dynamicity**: Different mixtures cause different performance degradation
- **Poor QoS**: Latency-critical workloads suffer from longer response time



[Yang *et.al.* ISCA '13] Bubble-Flux: Precise Online QoS Management for Increased Utilization in Warehouse Scale Computer



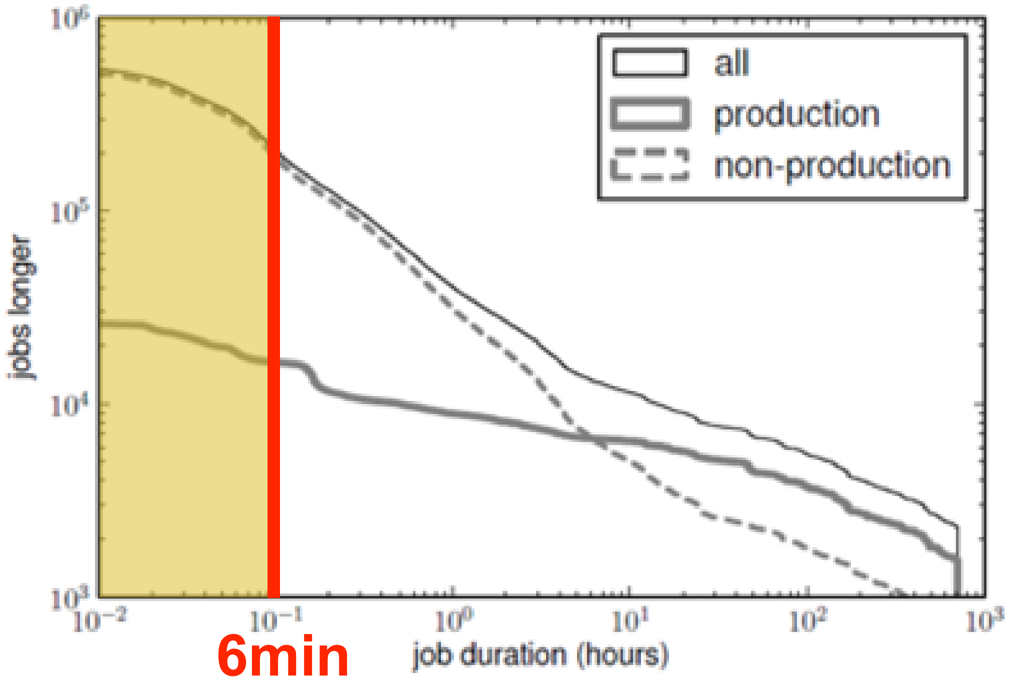
[Kambadur et.al SC'12] Measuring Interference Between Live Datacenter Applications

Hard to Predict

Google

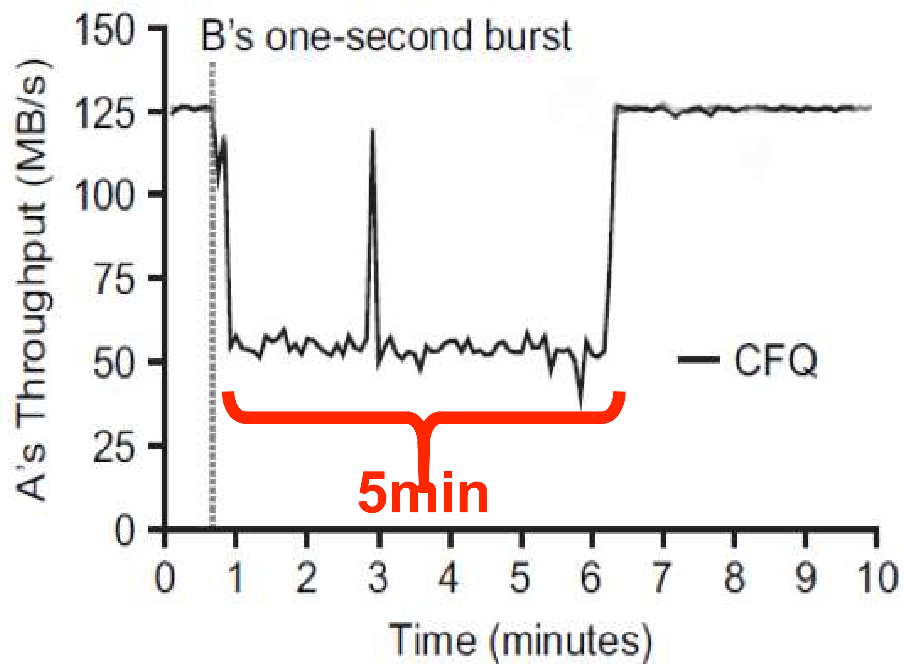
Tens of millions of jobs co-run 12,000 servers in a month

- Unpredictable short jobs
  - Test-and-debug



C. Reiss et al. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis, SOCC, 2012.

- B's one-second burst cause A's five-min degradation



S.Yang et al. Split-Level I/O Scheduling, SOSP, 2015.

Agenda

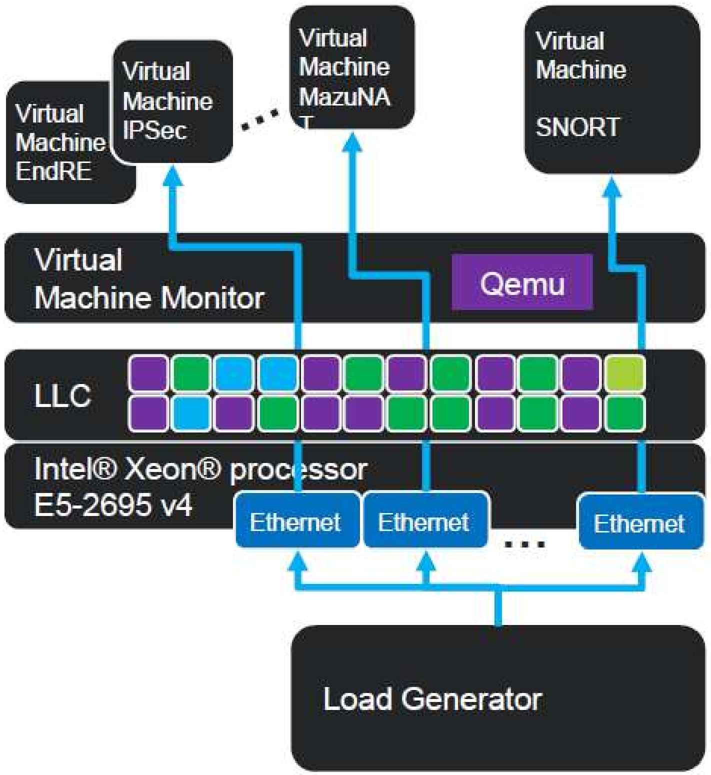
- Background
- Challenges
- **Opportunities**
- Our Work
- Summary

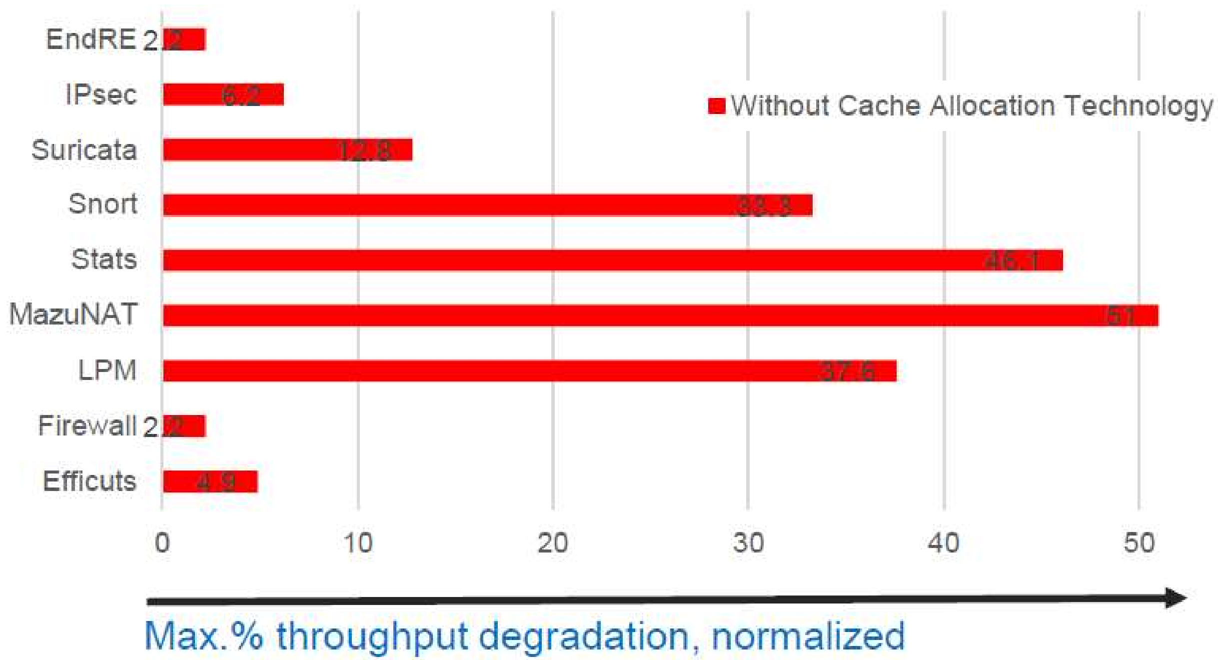
Intel Resource Director Technology

- In April 2016, Intel released Resource Director Technology （RDT） that support QoS
  - Cache Monitoring Technology (CMT)
  - Cache Allocation Technology (CAT) [HPCA’16]
  - Memory Bandwidth Monitoring(MBM)

NFV w/o CAT

- UC Berkeley’s Experimental results of CAT for network function virtualization (NFV)
- **w/o CAT**：throughput degrades by **51%**

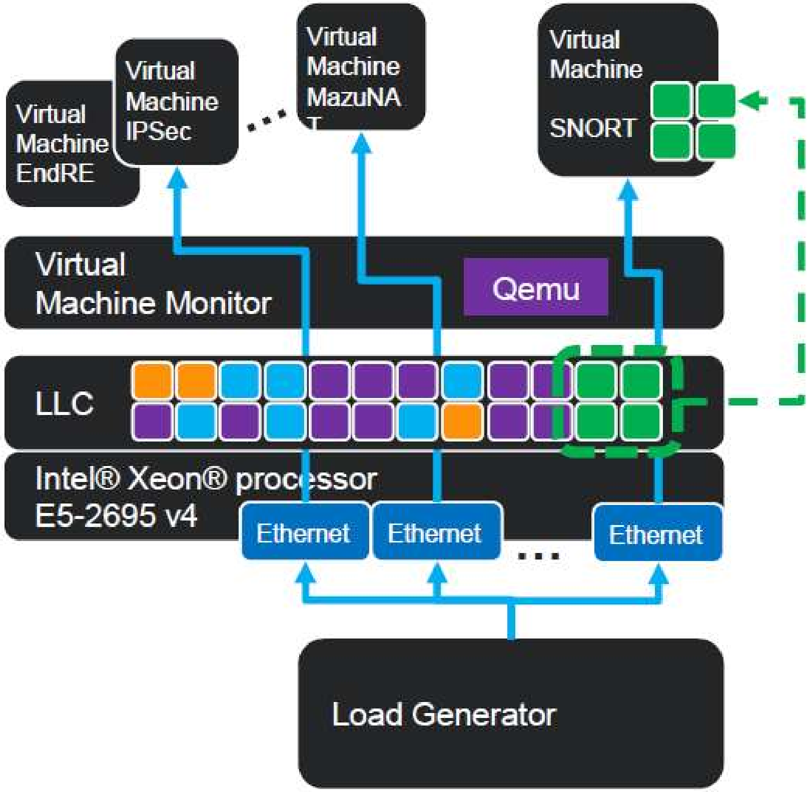


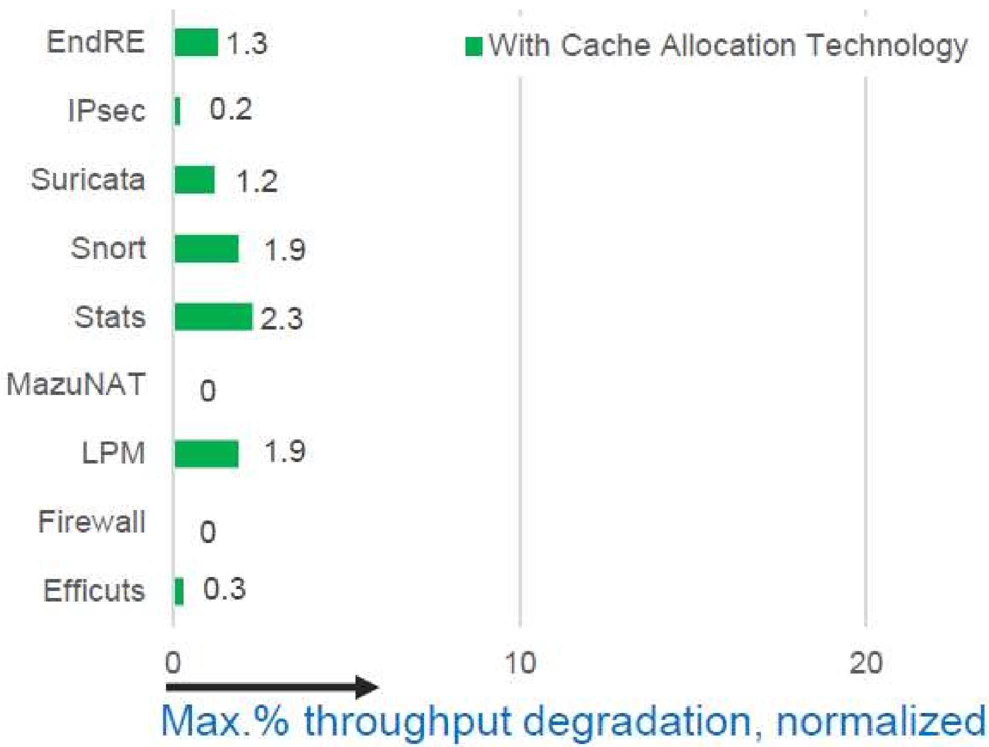


http://span.cs.berkeley.edu

NFV w/ CAT

- **w/ CAT**：Throughput degrades by **<2%** when dedicating two ways to a specific NF.





http://span.cs.berkeley.edu

Contention is Everywhere

- HyperThread、 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.

Data Center Era 2010s

- Search, On-line shopping, Cloud computing,…
- Priority, Throughput, Latency, …
- QoS v.s. Utilization

Separate Online/Offline Service

Internet Era 1990s

- HTTP, FTP, VoIP, Stream Media, Game, …
- VoIP, Game, …: Latency-critical
- FTP, VoD,…: Bandwidth-sensitive
- Email: Best Effort
- QoS

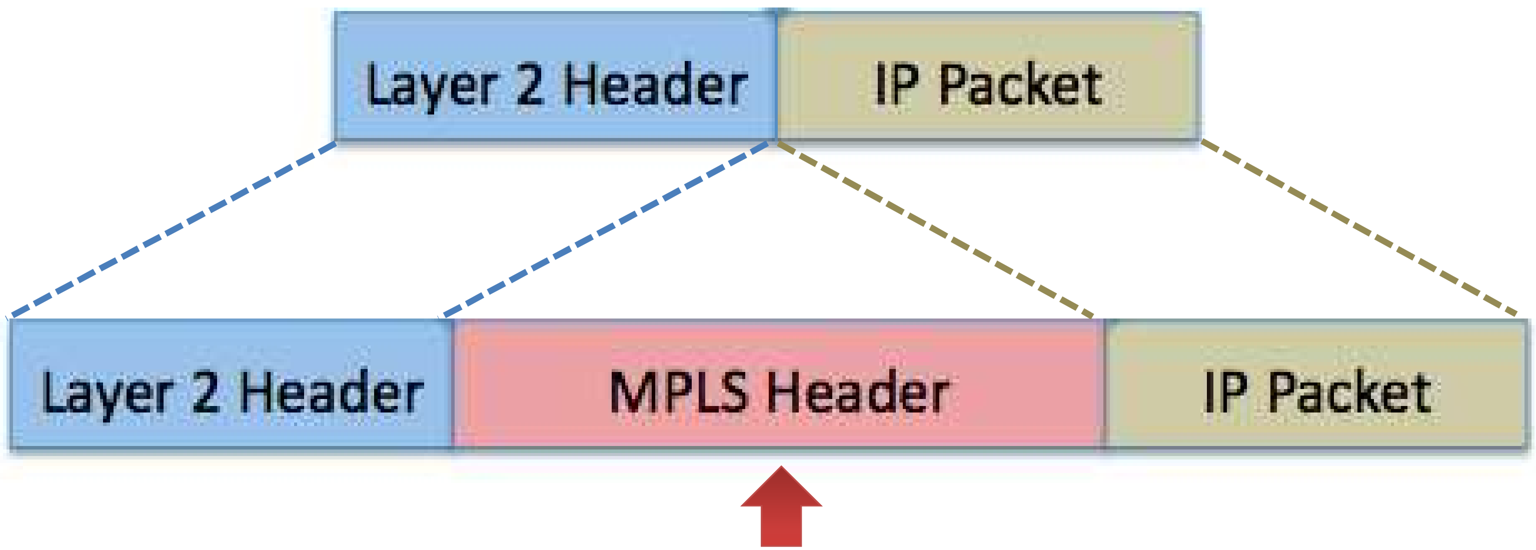
Applications sharing infrastructure → **Different Requirements** → **QoS Problem**

1994, Integrated services
1998, Differentiated Services
2001, MPLS

Fine-grain solution
Labeling each packet

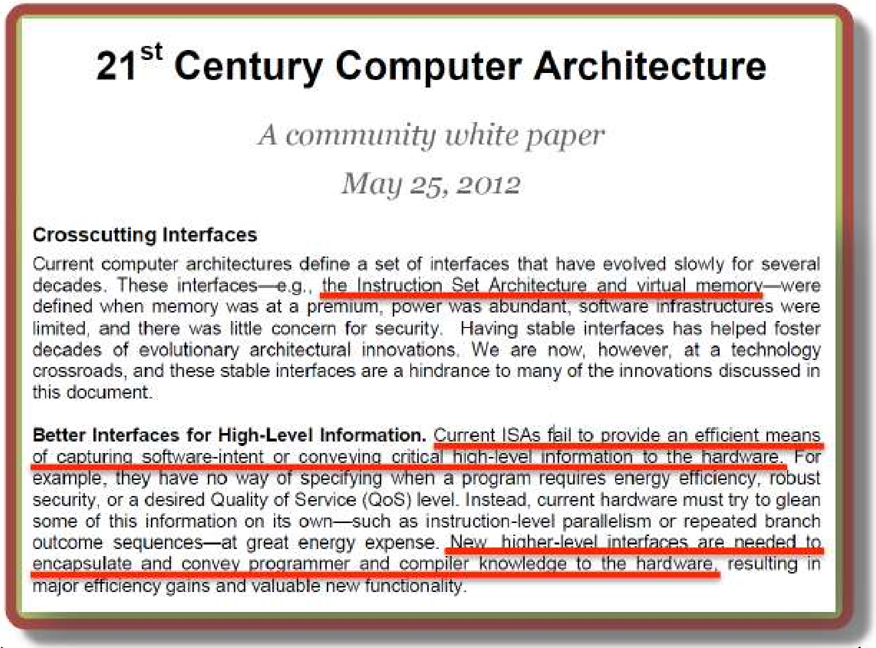
Labeled Networking

- **Fine-grain**：every packet has a label
- **Semantic Gap**：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

Arch requires new interfaces



21st Century Computer Architecture

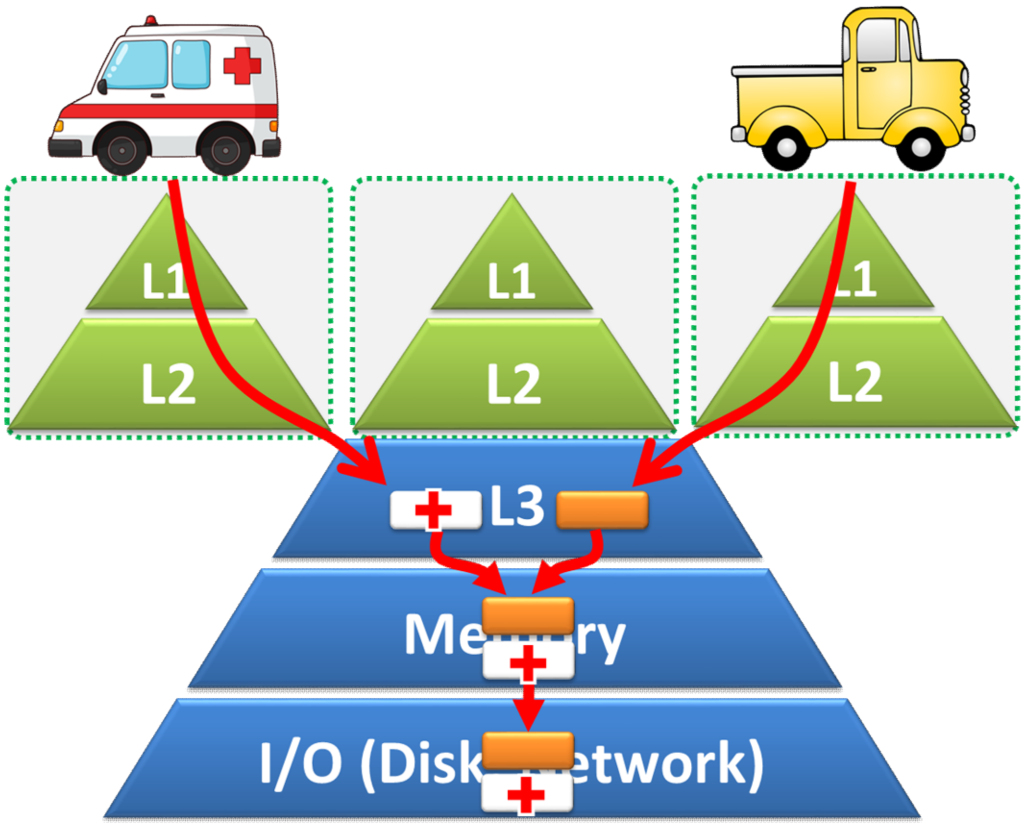
A community white paper

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.



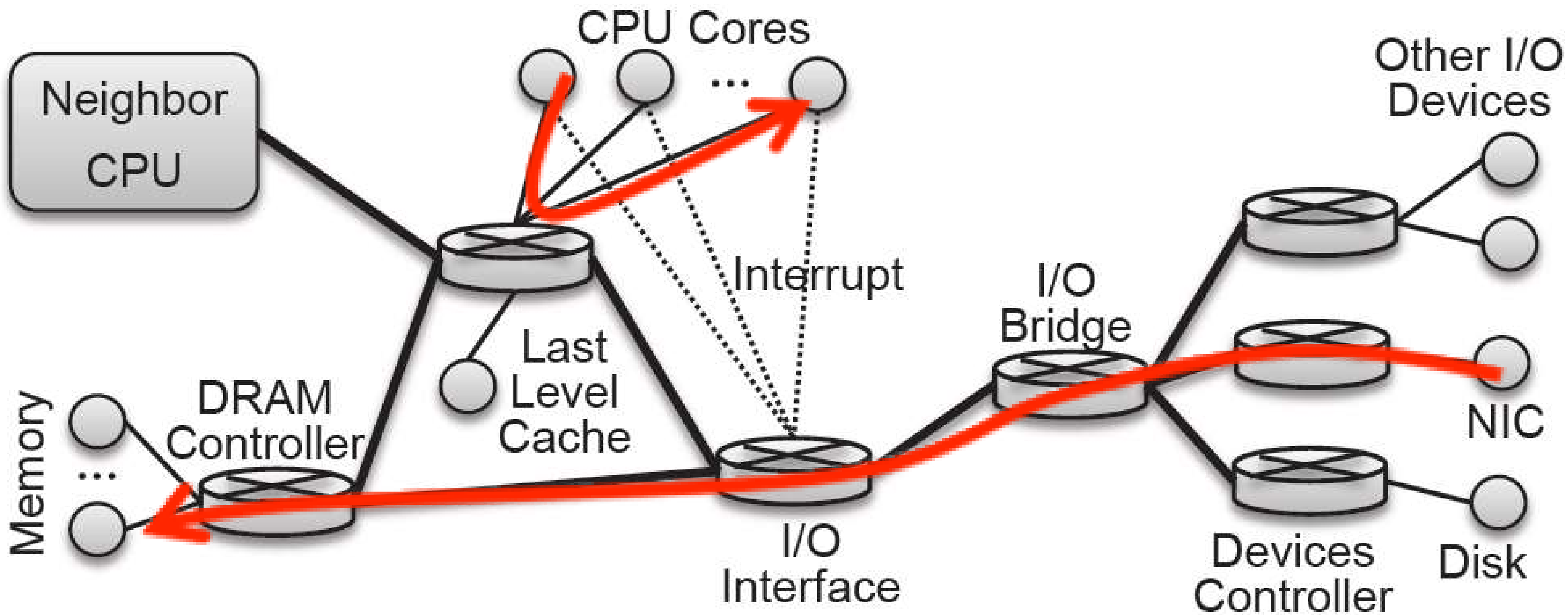
New, high-level interfaces are required to convey programmer and compiler knowledge to the hardware.

21st Century Computer Architecture

Labeled Architecture?

The Computer as a Network

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



DW 0	R 0	Fmt 0x2	Type 0x00	R 0	TC 0	R 0	TD 0	EP 0	Attr 0	R 0	Length 0x001
DW 1	Requester ID 0x0000						Tag (unused) 0x00			Last BE 0x0	1st BE 0xf
DW 2	Address [31:2] 0x3f6bfc10										R 0
DW 3	Data DW 0 0x12345678										

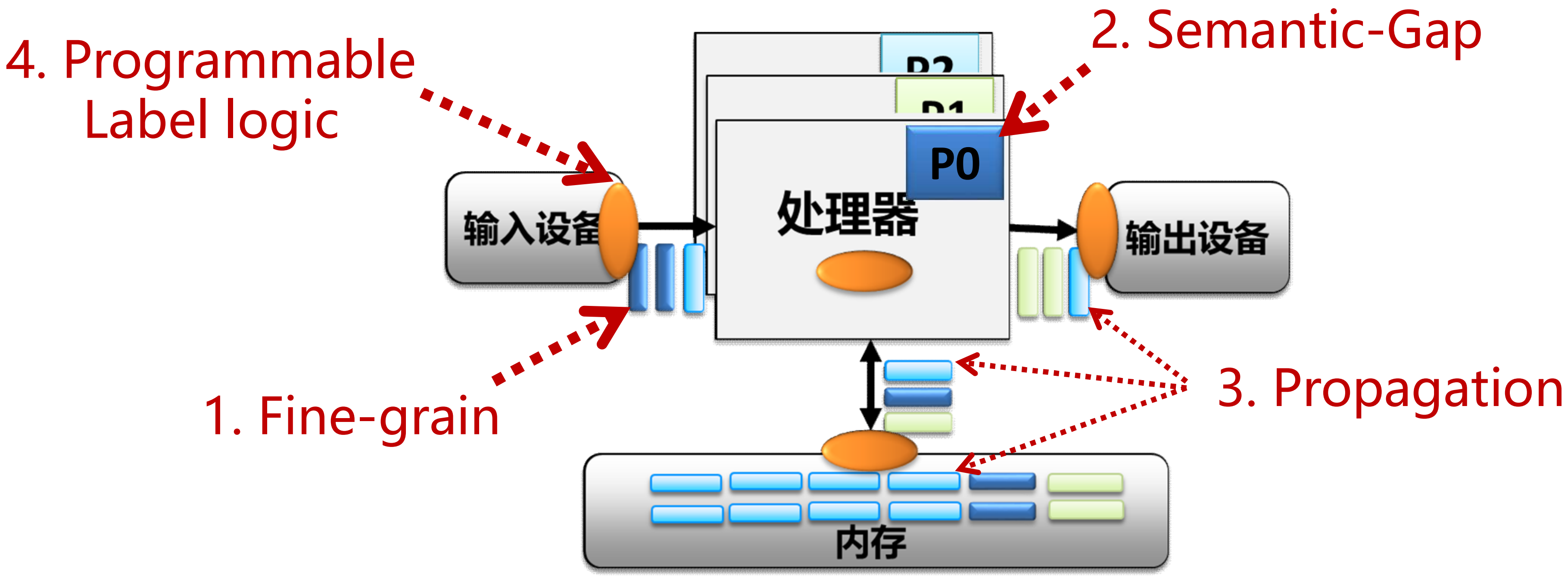
Yes!

Agenda

- Background
- Challenges
- Opportunities
- **Our Work**
- Summary

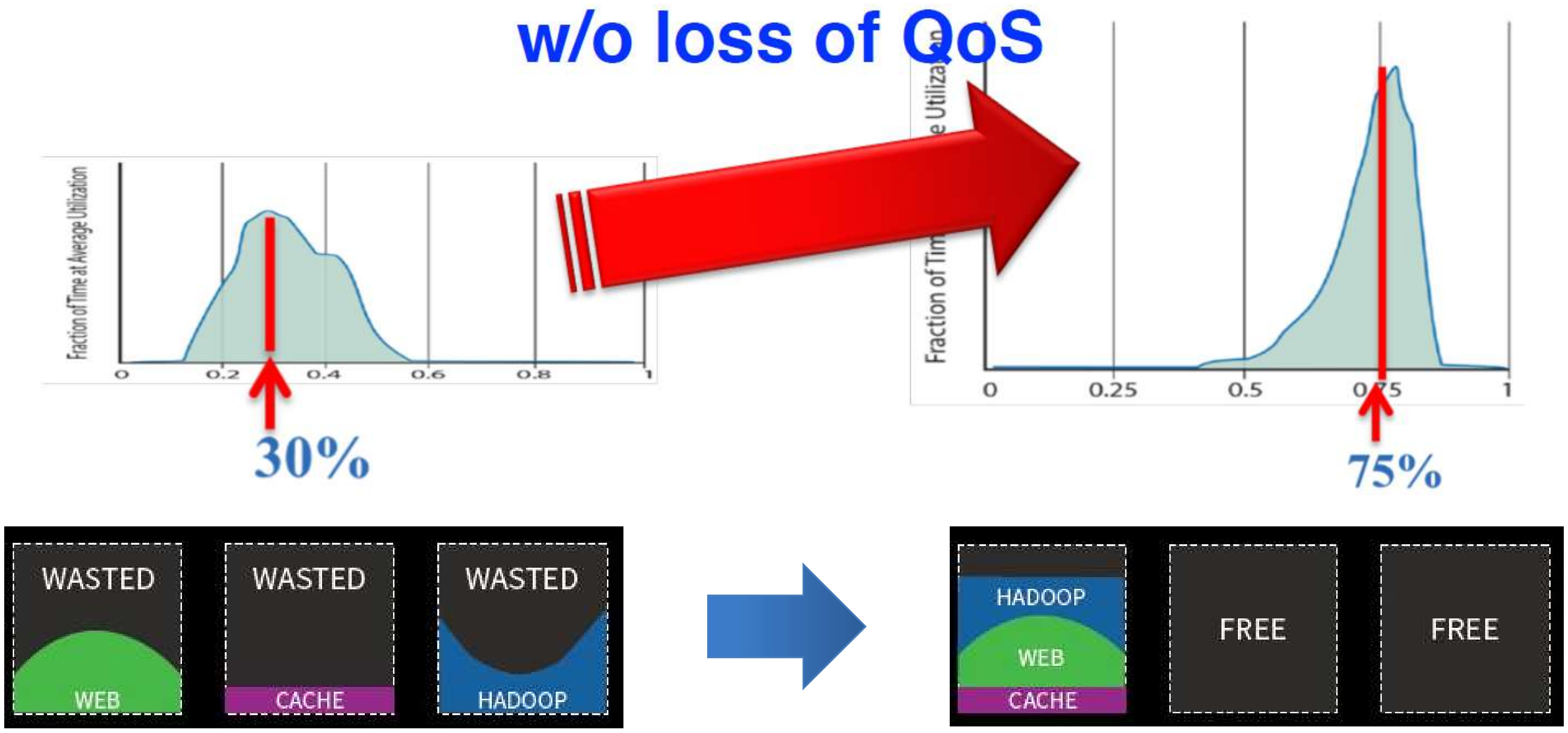
Labeled von Neumann Architecture (LvNA)

- **Fine-grain :** attach a label to each memory and I/O request
- **Semantic-Gap** : 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



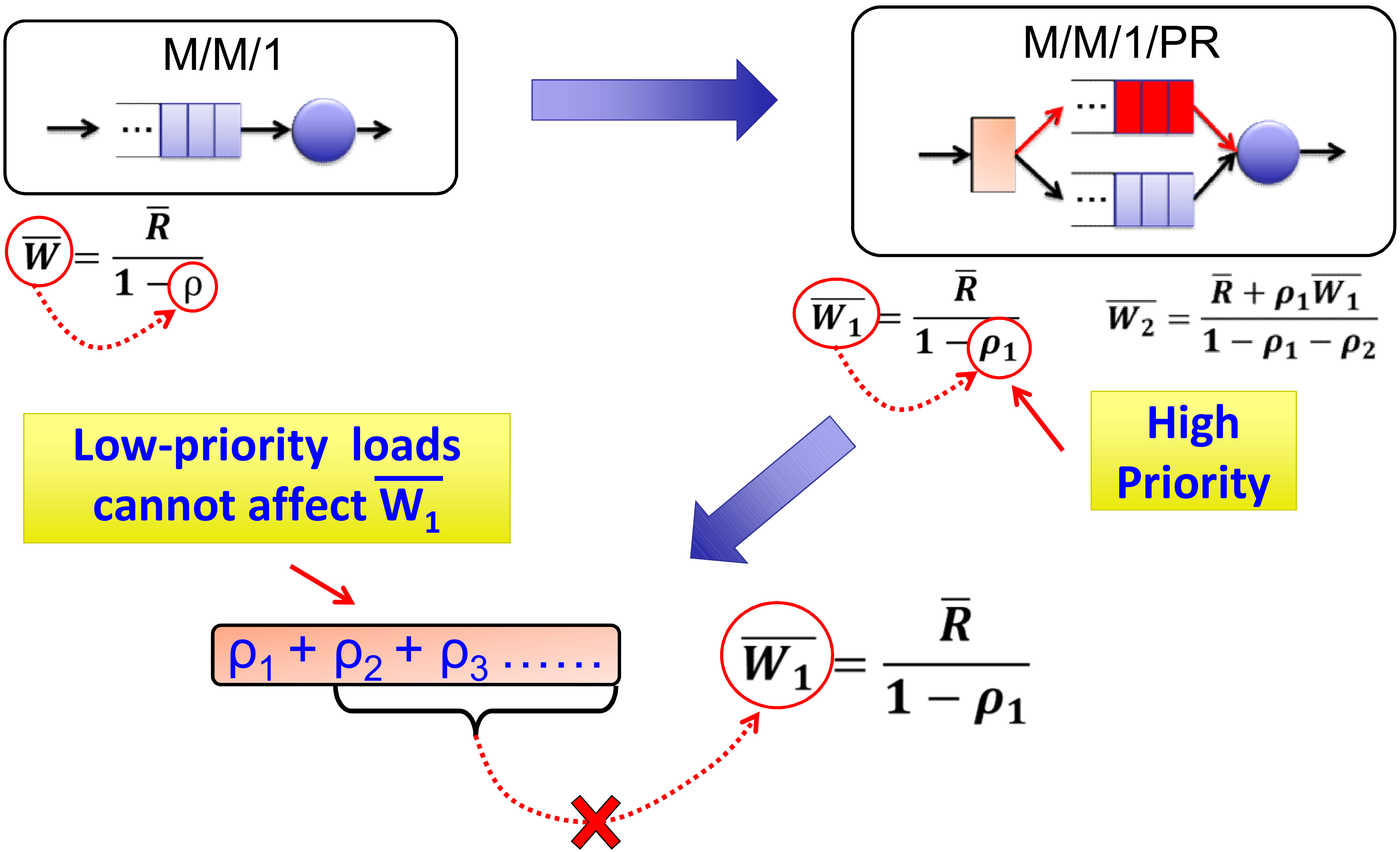
Bao and Wang, Labeled von Neumann Architecture for Software-Defined Cloud, Journal of Computer Science and Technology, 2017 Vol. 32 (2): 219-223.

Goal


w/o loss of QoS
Fraction of Time at Average Utilization
0
0.2
0.4
0.6
0.8
1
30%
0
0.25
0.5
0.75
1
75%
WASTED
WEB
WASTED
CACHE
WASTED
HADOOP
HADOOP
WEB
CACHE
FREE
FREE

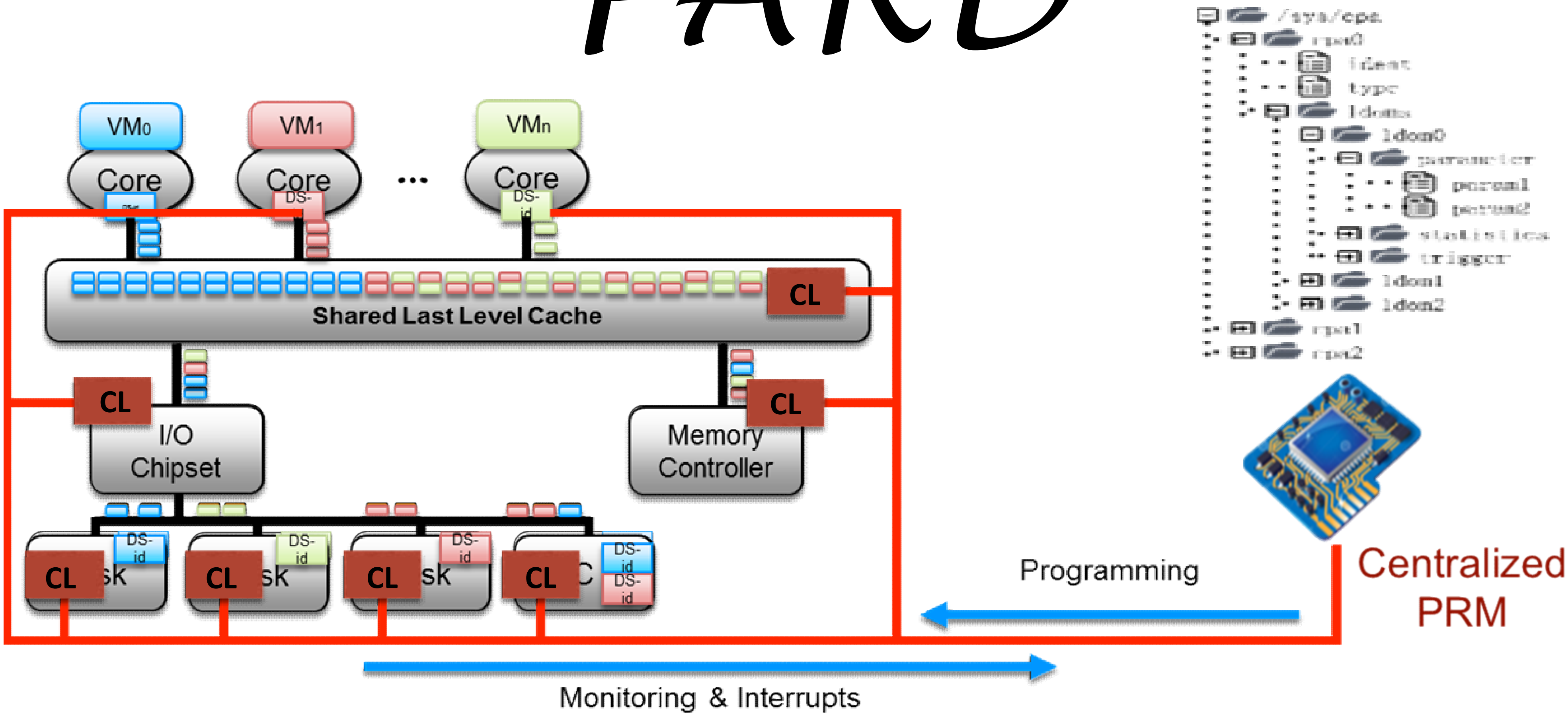
Labeling + CP → Priority Queues

- PriQ can achieve both utilization and QoS



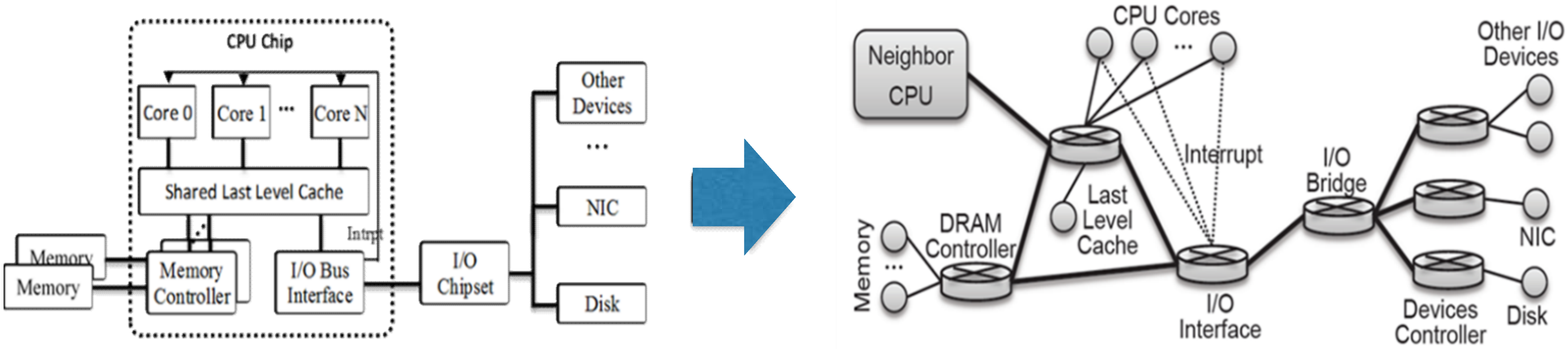
Programmable **A**rchitecture for **R**esourcing-on-**D**emand

PARD



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

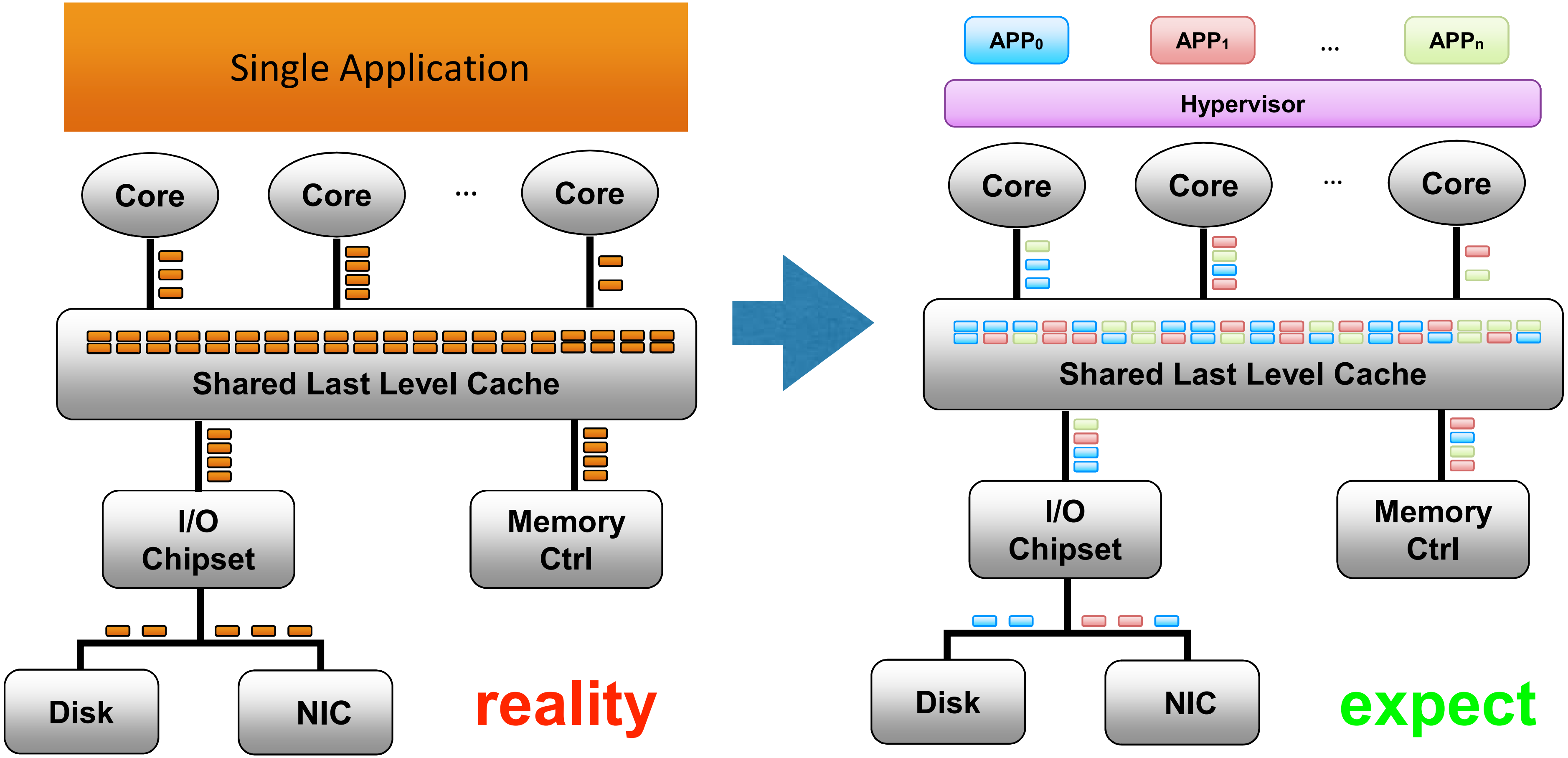
Challenges in Reconstruction



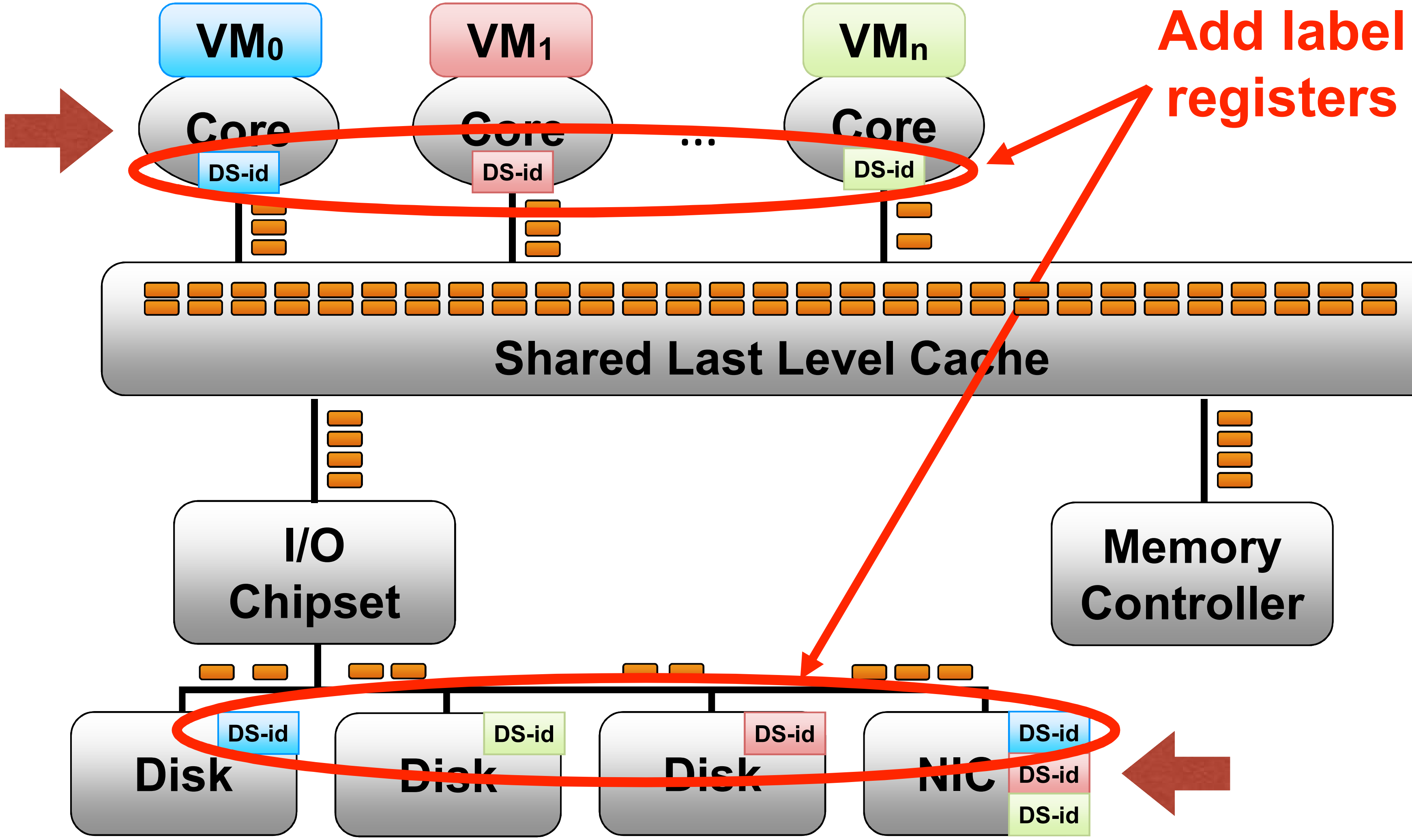
1. How to enforce labeling mechanism?
2. How to design control logics?
3. How to design programming interface?

Challenge #1

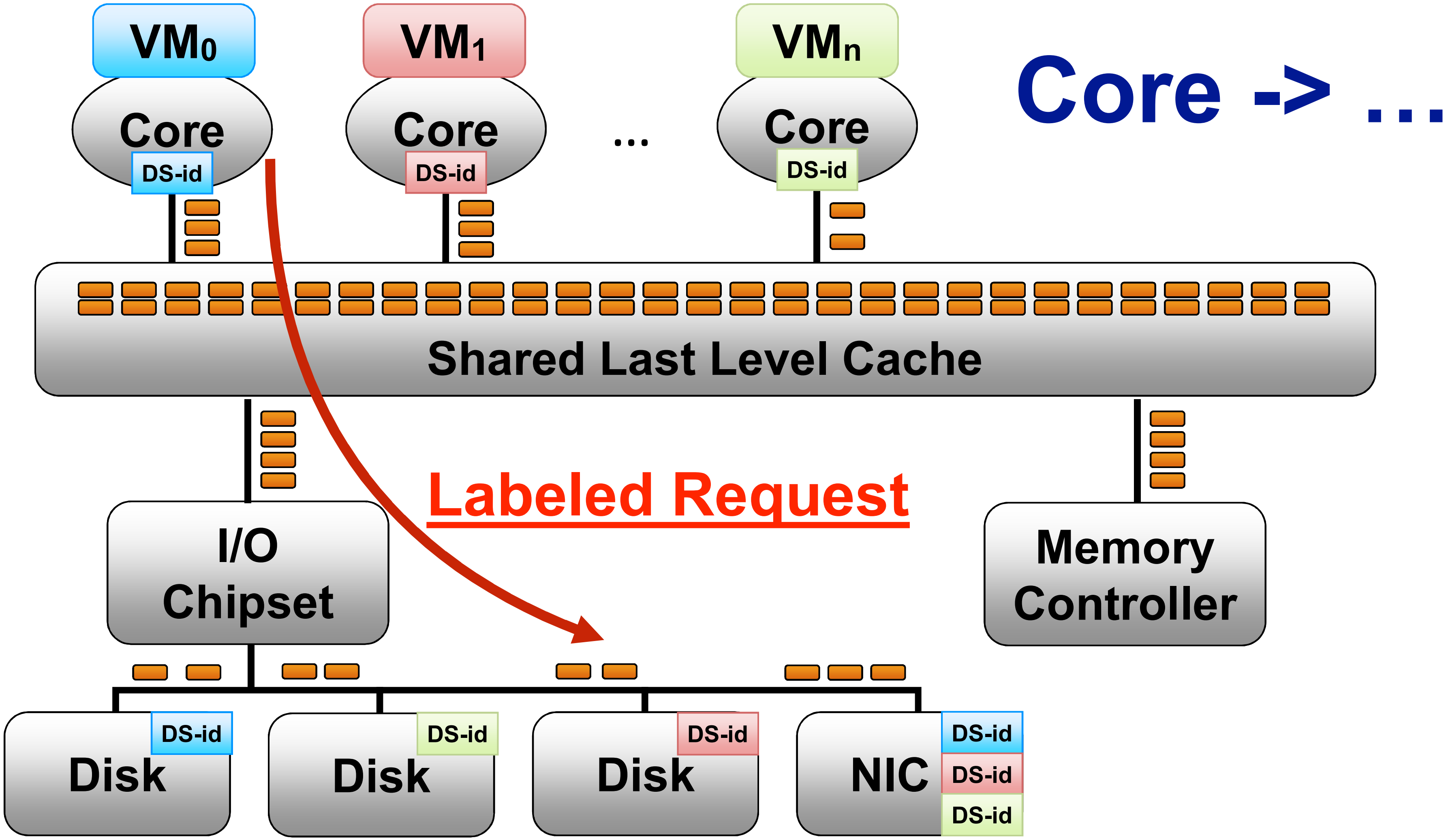
How to enable computer hardware to distinguish different applications?



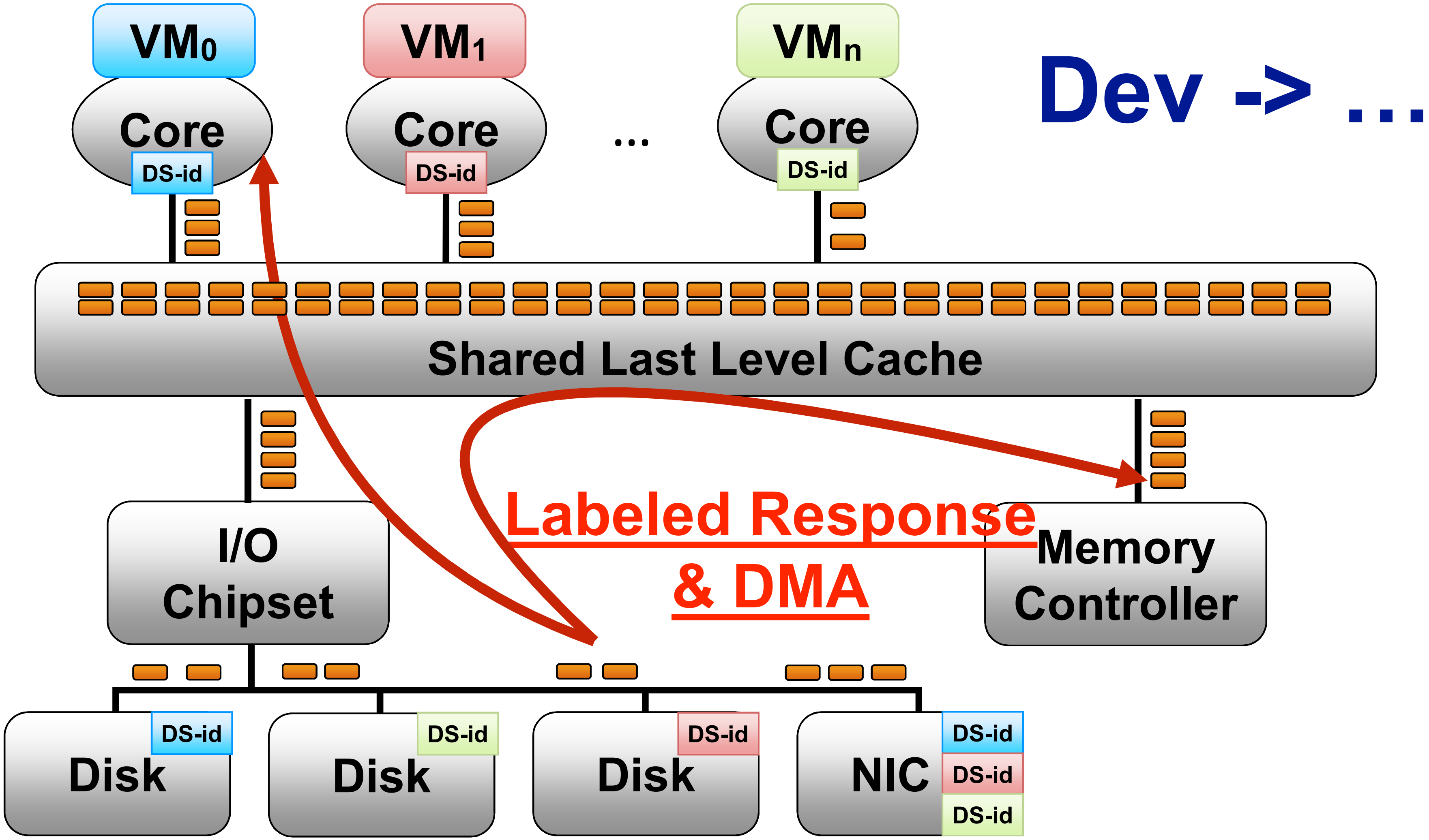
Label Sources



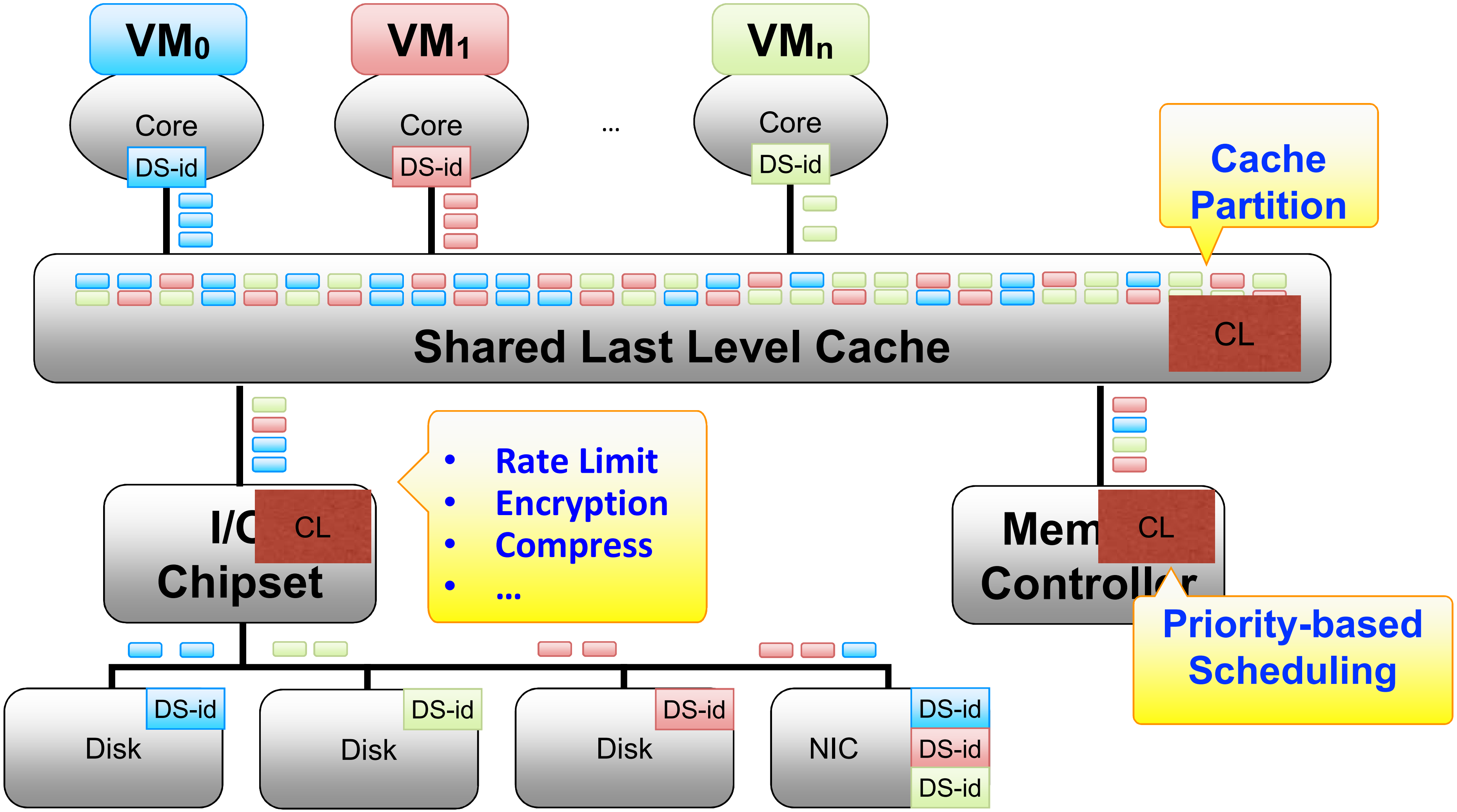
Propagate Labels in Datapath



Propagate Labels in Datapath

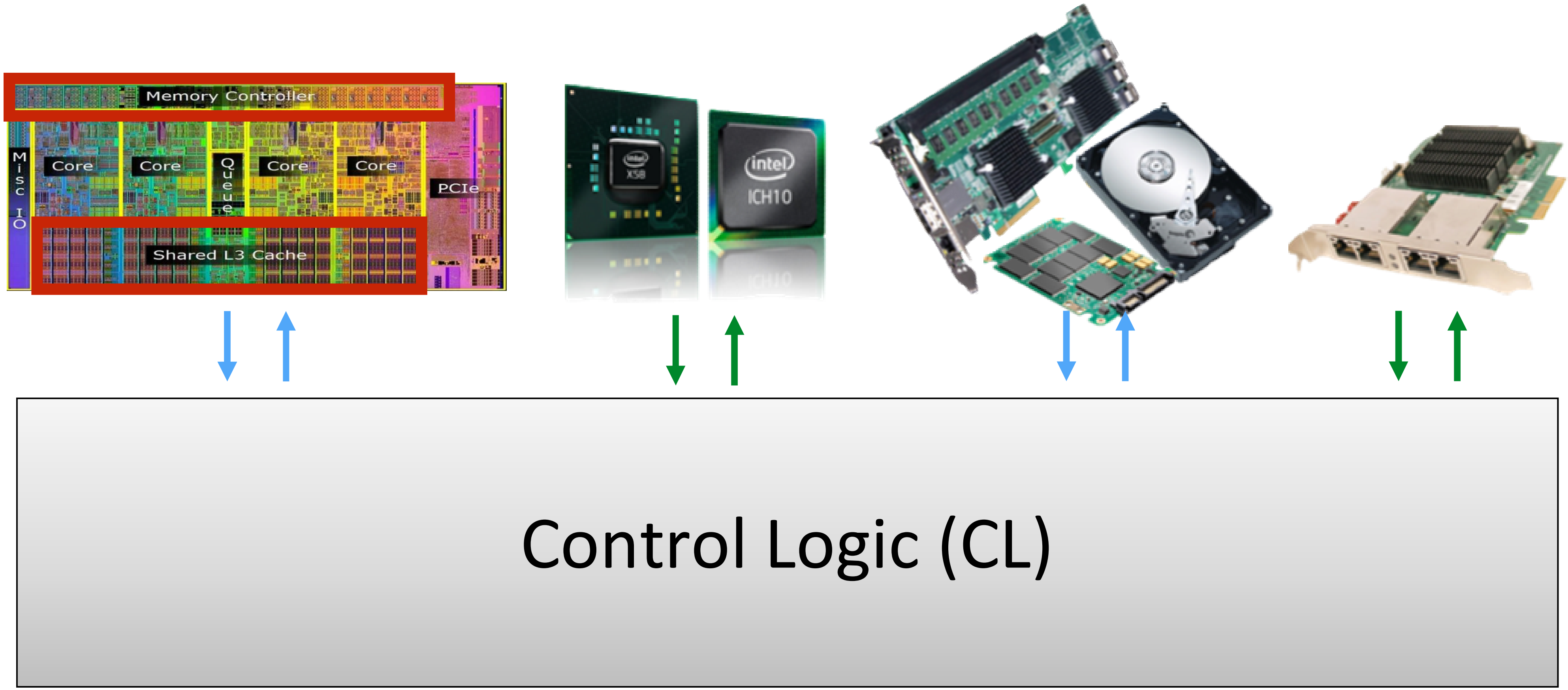


How to User Labels



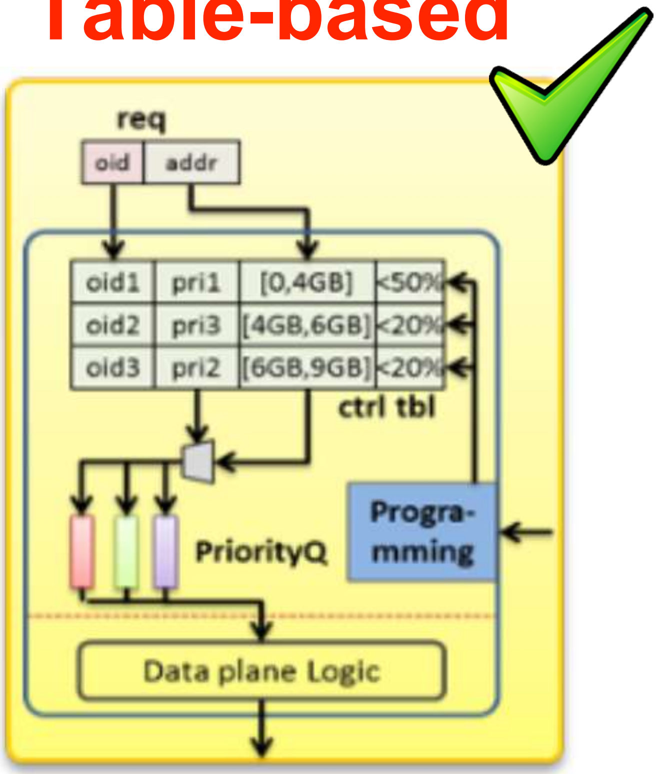
Challenge #2

How to design control logics for a diversity of hardware?



CL Design Choices

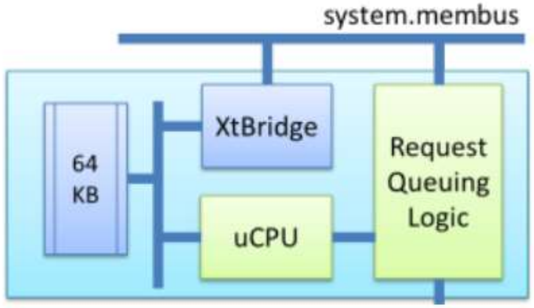
Table-based



- Simple to implement, Fast
- Inflexible

v.s.

Processor-based

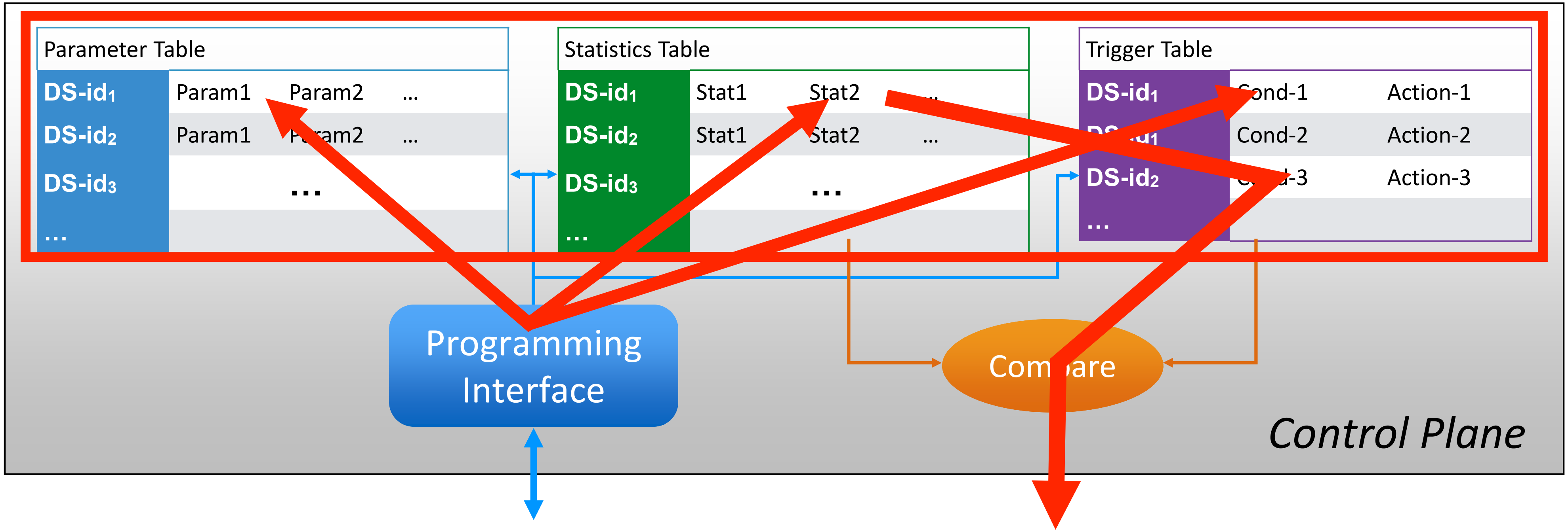



```
loop:
   rbld
   rbrd r1, <req-type offset>
   cmp r1, REQUEST
       be .request
   cmp r1, RESPONSE
       be .response
.dispatch:
   rbst
       b .loop
.request:
   call encrypt
       b .dispatch
.response:
   call decrypt
       b. dispatch
```

- Support advanced functionalities
- Complicated, slow

Table-based CL Design

Three Tables + Programming Interface + Interrupt Line

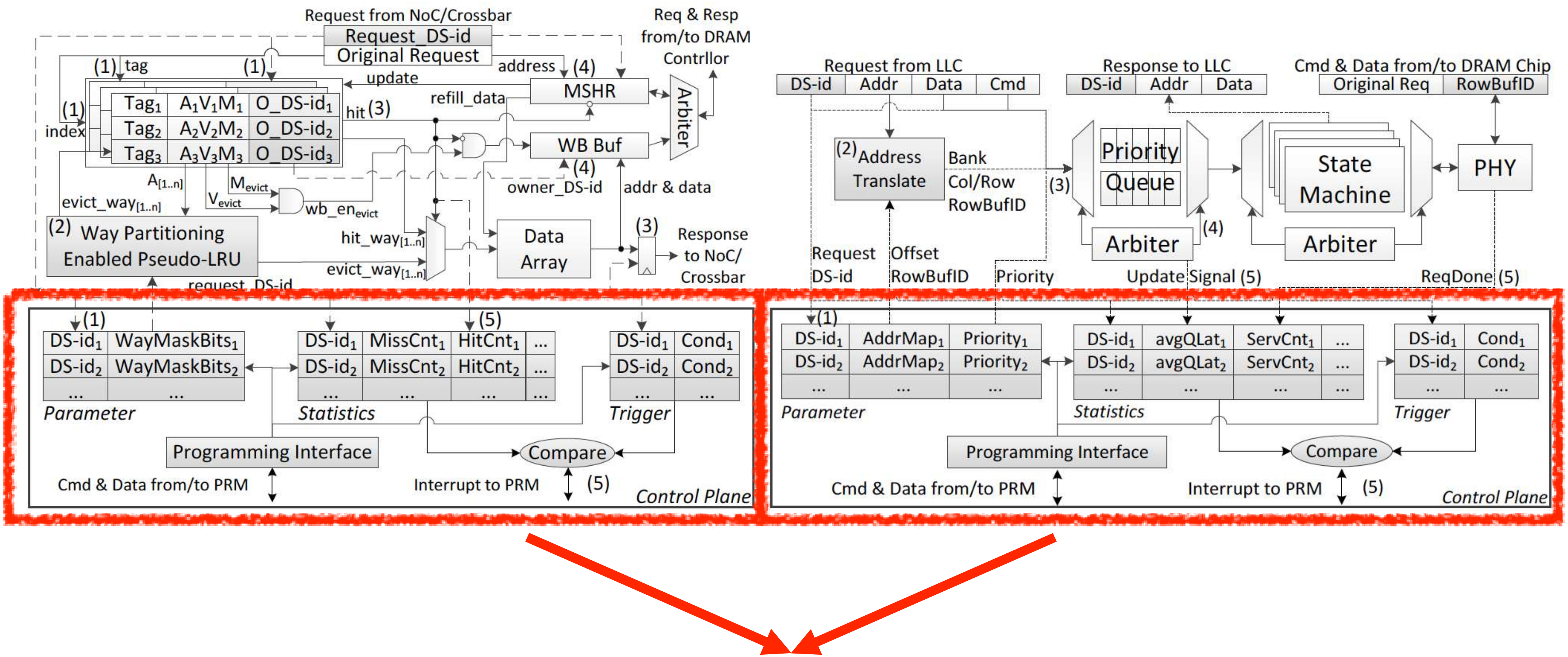


- **Three Control Table**: Parameter / Statistics / Trigger
- **A Programming Interface**: Control Tables R/W
- **A Interrupt Logic**: Send Interrupt when trigger condition meet

Integrate into HW Components

Cache Controller

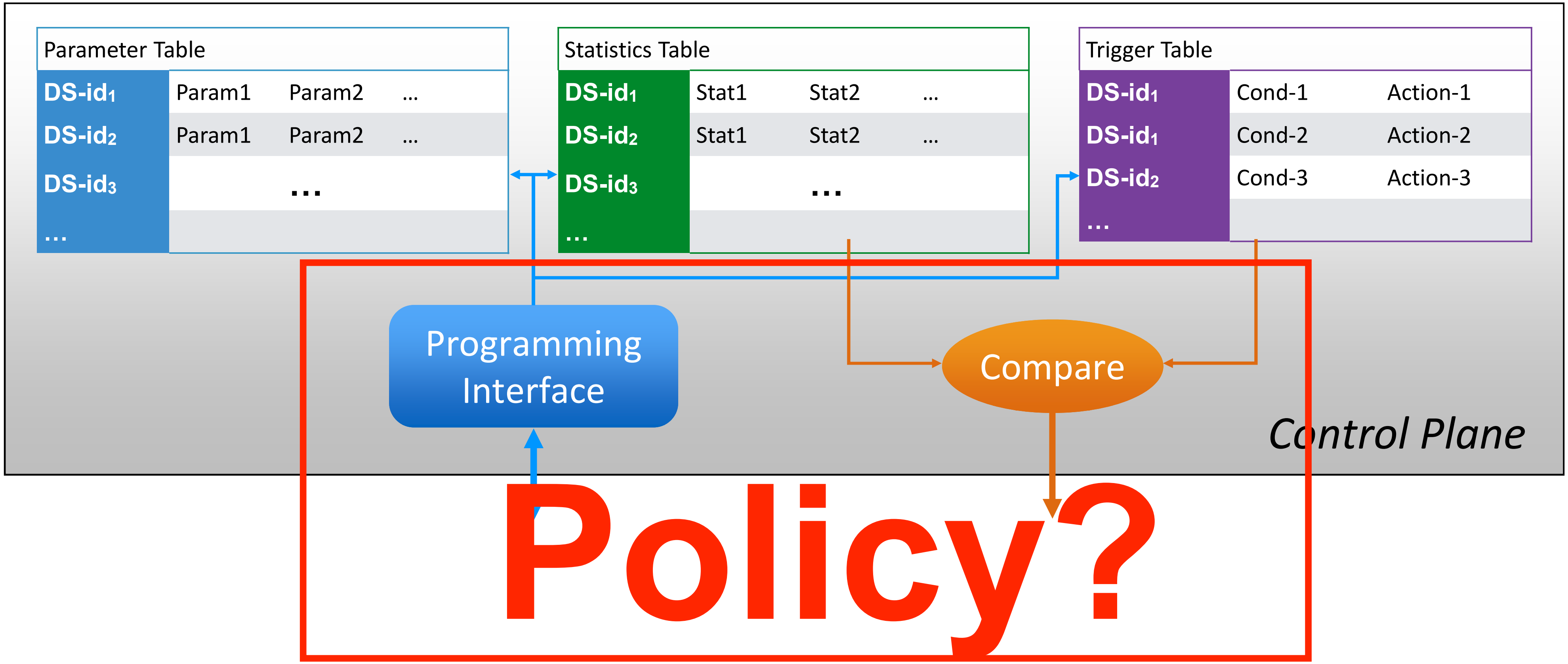
Memory Controller



Common Control Logic Structure

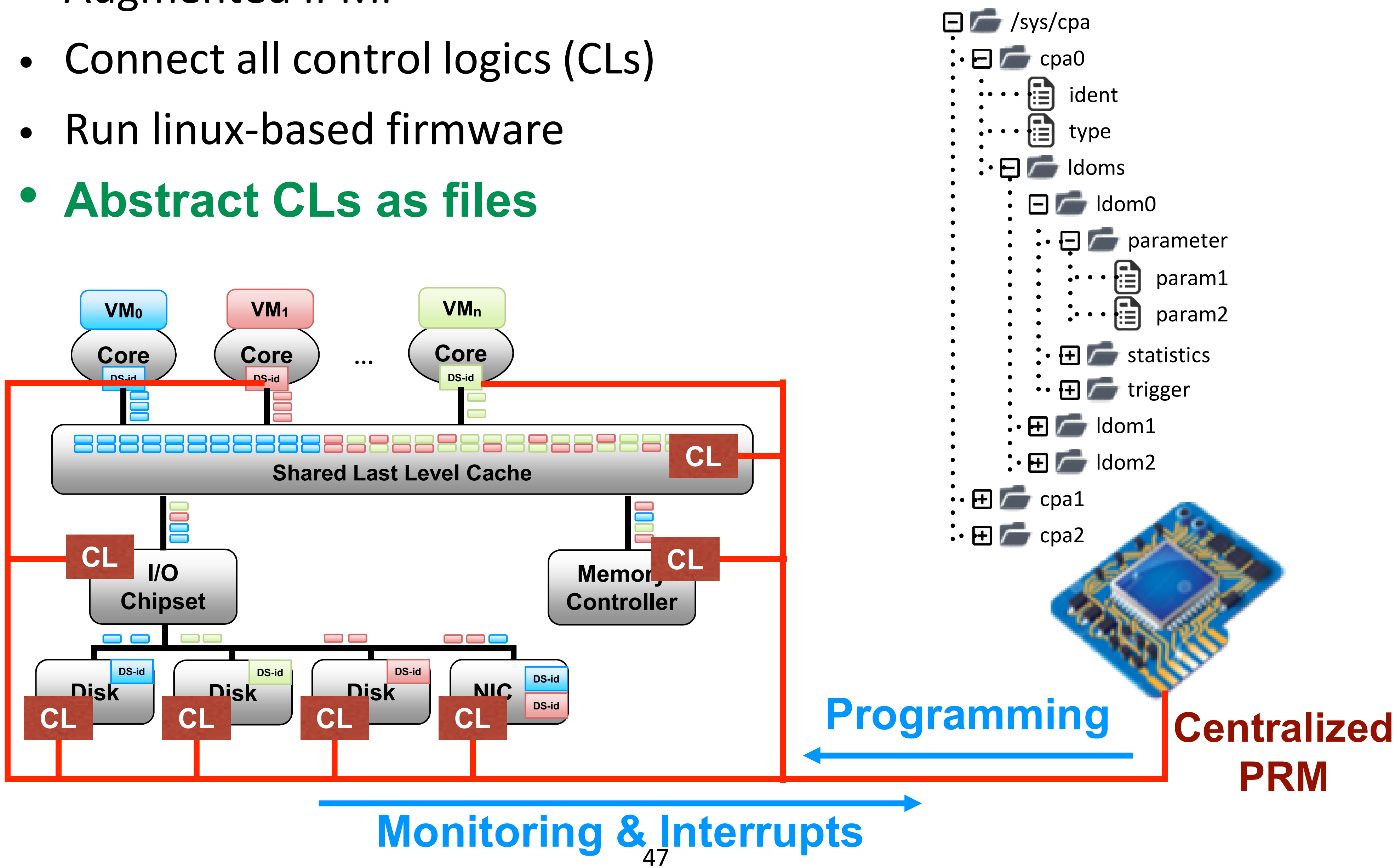
Challenge #3

How to define/program resourcing-on-demand policy into hardware



Platform Resource Manager (PRM)

- Augmented IPMI
- Connect all control logics (CLs)
- Run linux-based firmware
- **Abstract CLs as files**



Access Control Logics

Query Control Logic Info

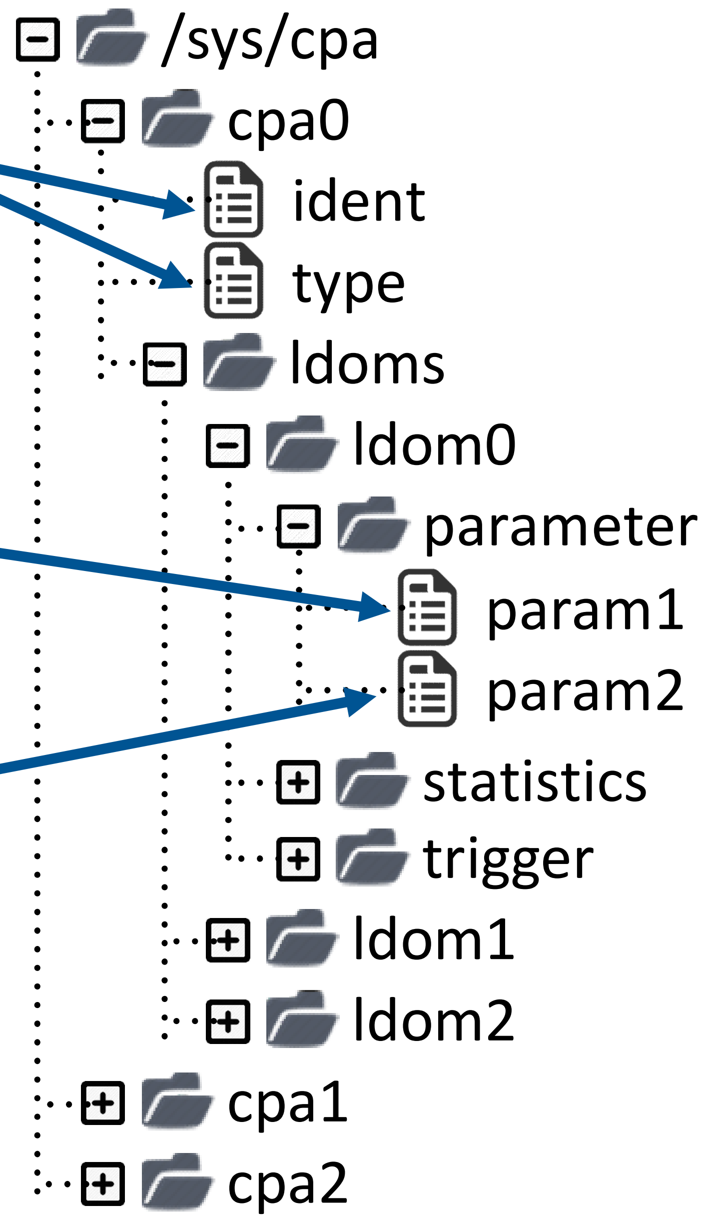
cat /sys/cpa/cpa0/ident
cat /sys/cpa/cpa0/type

Query Parameters

cat /sys/cpa/cpa0/…/parameter/param1

Setting Parameters

echo 10 > /sys/cpa/cpa0/…/parameter/param2



Trigger->Action

1. Register trigger

pardtrigger /dev/cpa0
-ldom=0 -action=0
-stats=miss_rate -cond=gt,30

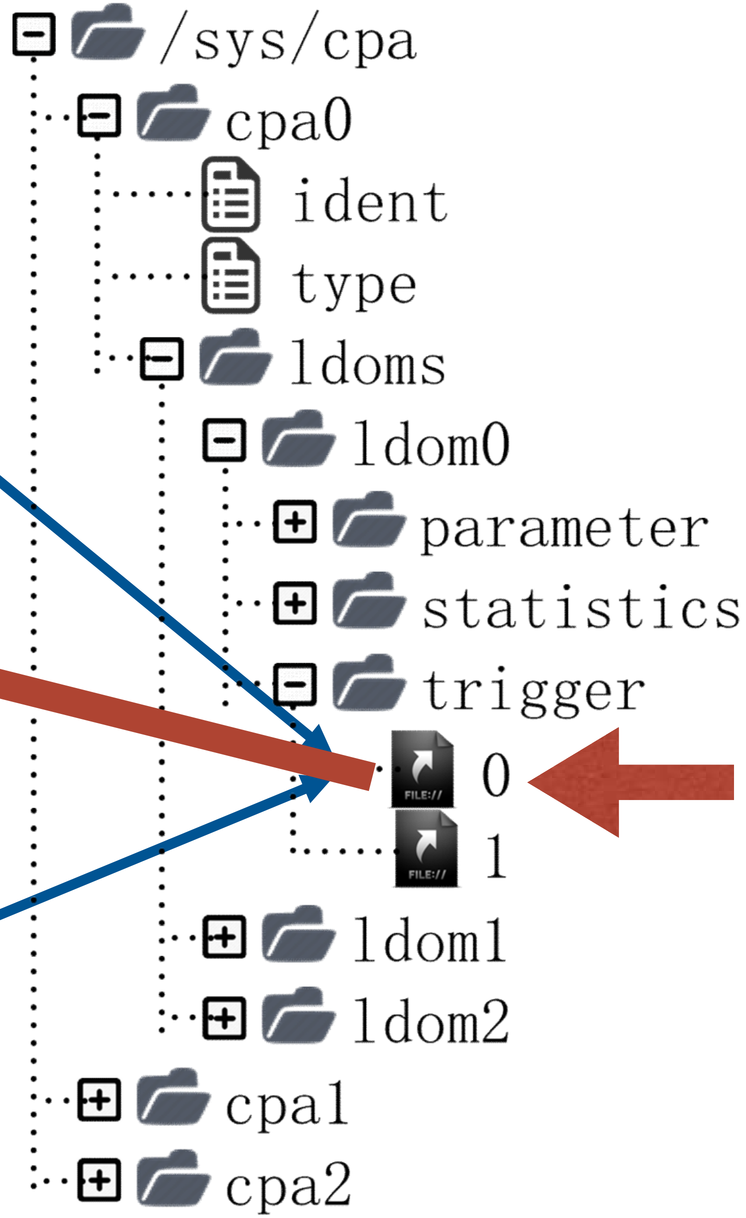
2. Prepare action scripts

Example 2: /cpa0_ldom0_t0.sh

```
# !/bin/sh
echo "<log message>" > /log/triggers.log
cur_mask=`cat /sys/cpa/.../waymask`
miss_rate=`cat /sys/cpa/.../miss_rate`
capacity=`cat /sys/cpa/.../capacity`
target=update_mask(
        $cur_mask, $miss_rate, $capacity)
echo $targe > /sys/cpa/…/waymask
```

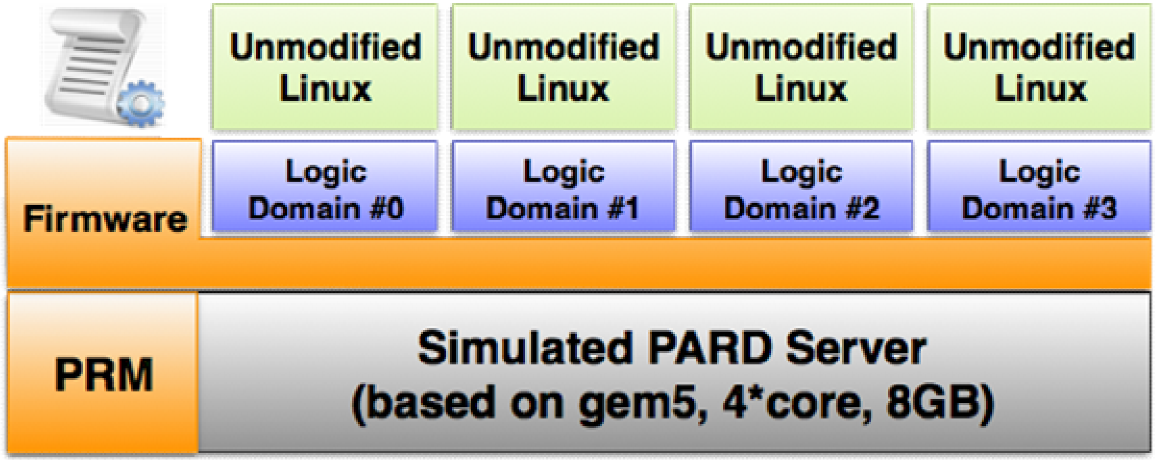
3. Install trigger action script

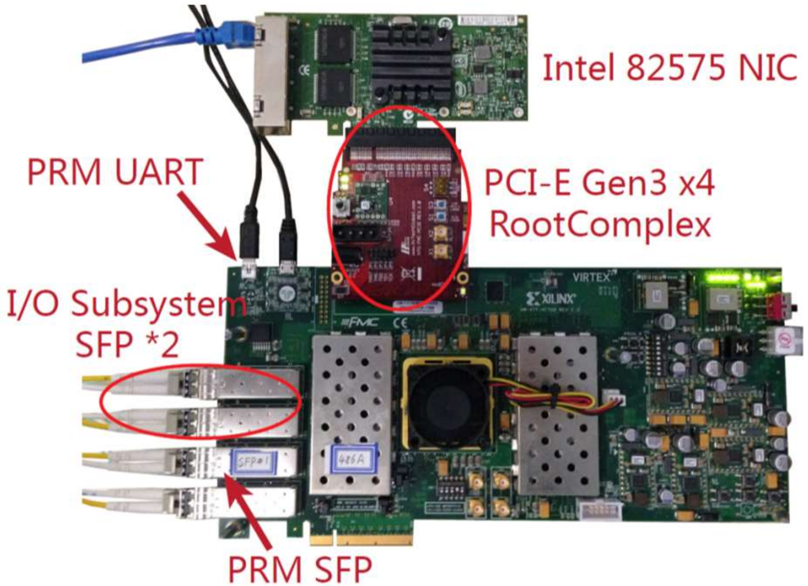
echo "/cpa0_ldom0_t0.sh" >
/sys/cpa/cpa0/ldoms/ldom0/triggers/0



Implementation

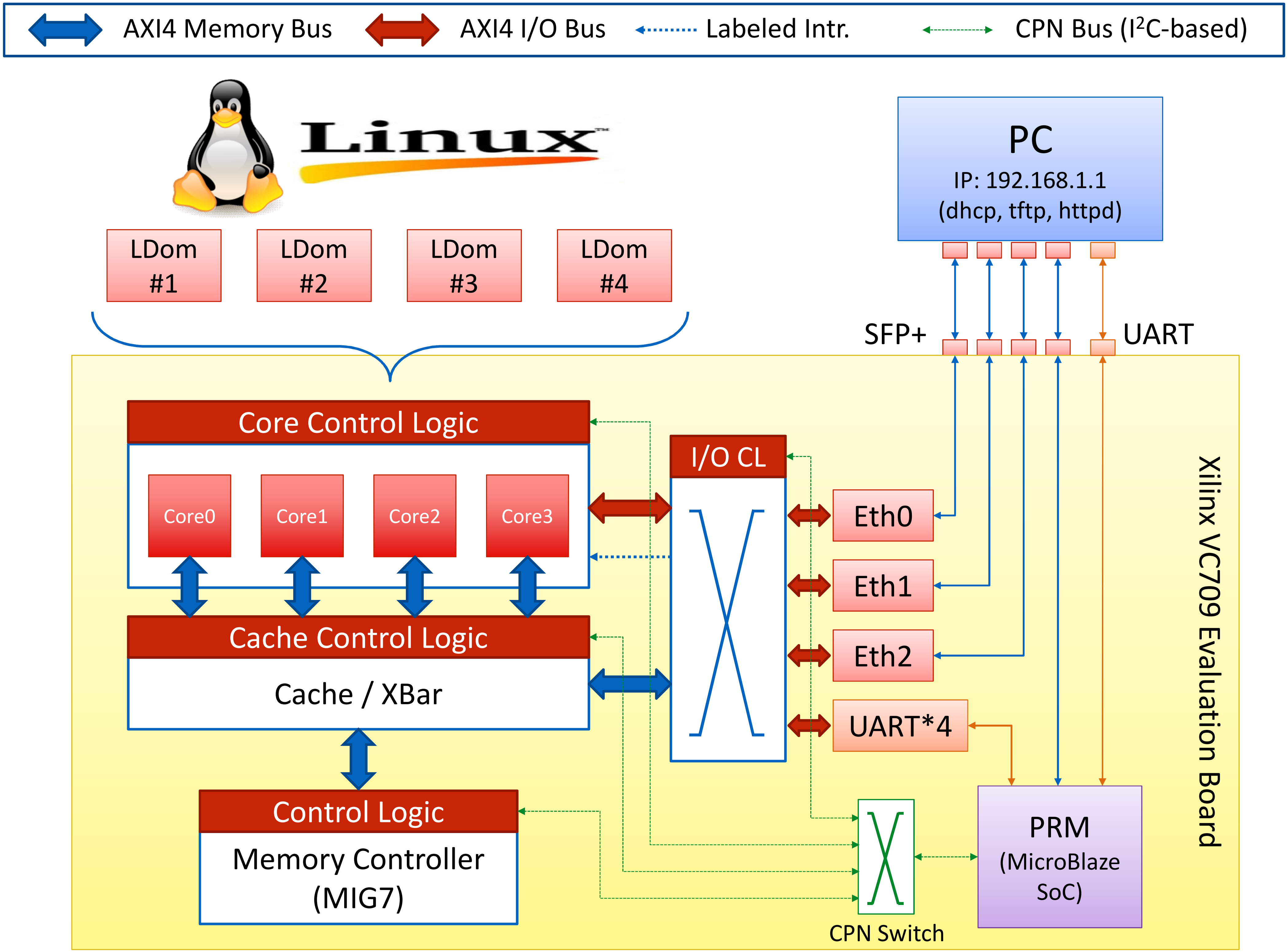
- Full-system cycle-accurate simulator **Open Sourced ***
- FPGA prototype on Xilinx VC709 evaluation board
- Microblaze version **Deprecated**
- RISC-V version **Coming soon ⁺**





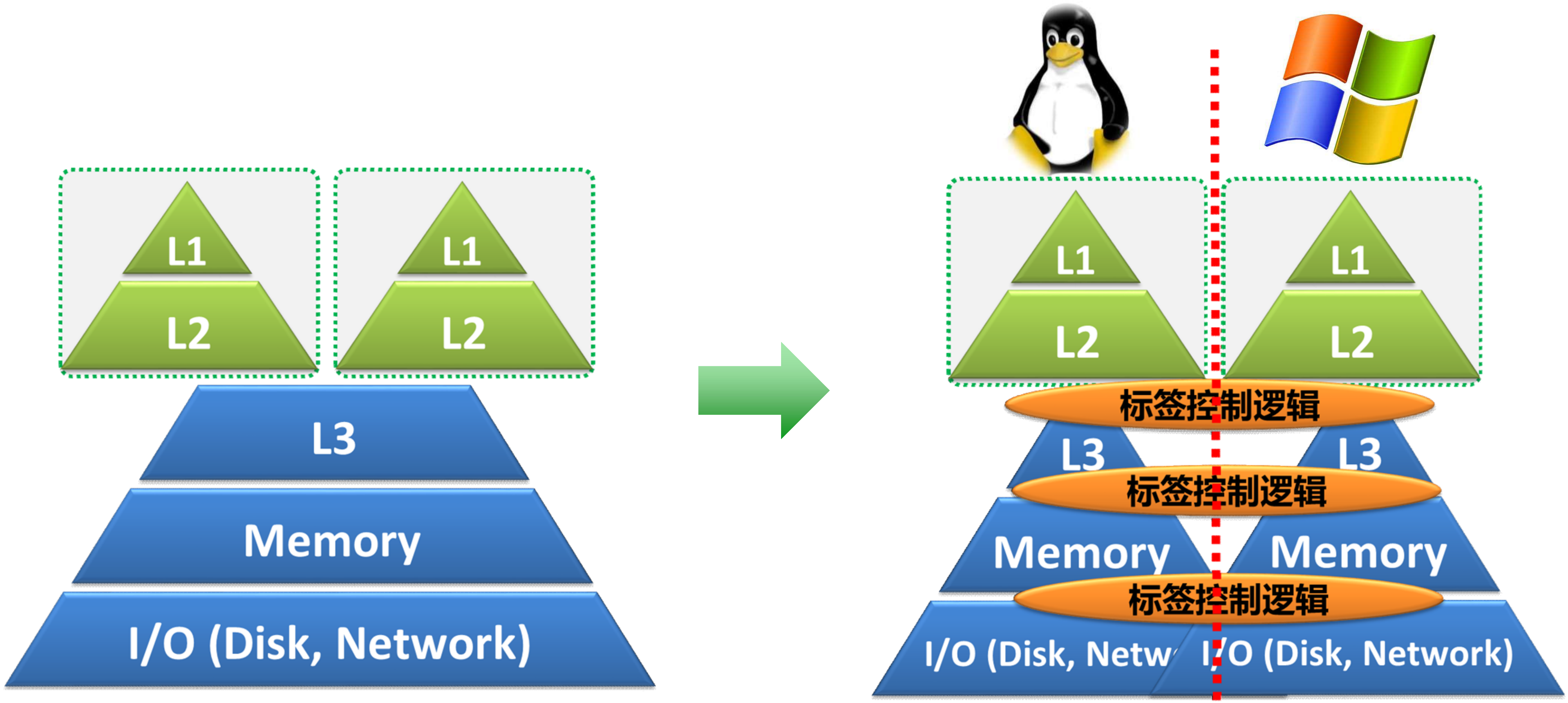
* available at http://github.com/fsg-ict/PARD-gem5

⁺ check http://github.com/fsg-ict/PARD-fpga

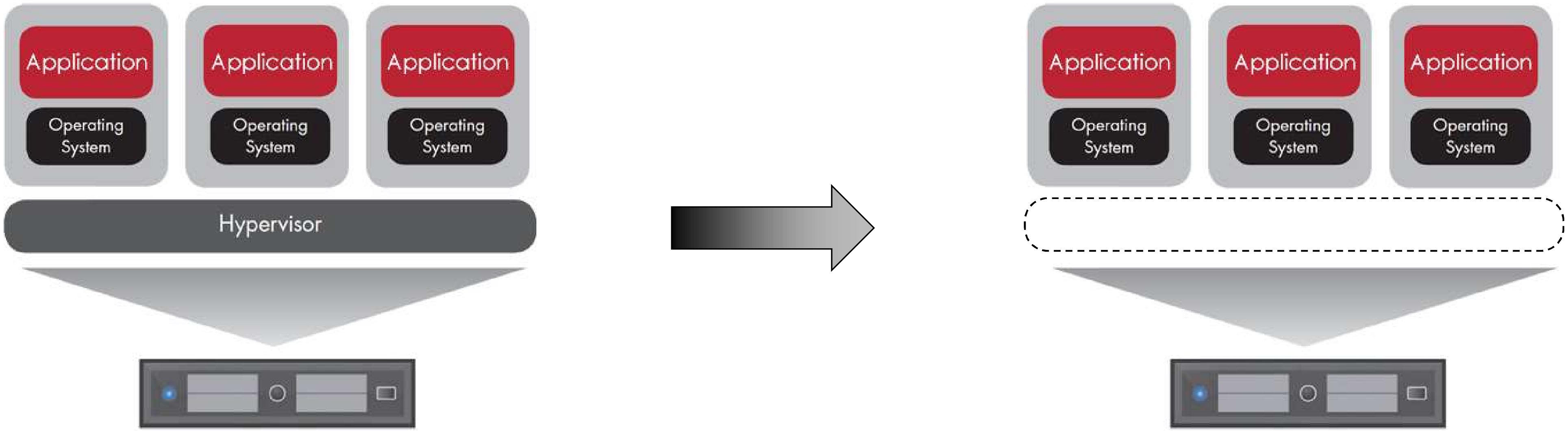

AXI4 Memory Bus
AXI4 I/O Bus
Labeled Intr.
CPN Bus (I2C-based)
PC
IP: 192.168.1.1
(dhcp, tftp, httpd)
LDom #1
LDom #2
LDom #3
LDom #4
SFP+
UART
Core Control Logic
Core0
Core1
Core2
Core3
I/O CL
Eth0
Eth1
Eth2
UART*4
Cache Control Logic
Cache / XBar
Control Logic
Memory Controller (MIG7)
CPN Switch
PRM (MicroBlaze SoC)
Xilinx VC709 Evaluation Board

Case 1: Add address mapping into CLs

- The whole server is partitioned into several sub-macines



Bare Metal Virtualization w/o Hypervisor



Web application (non-virtualized)

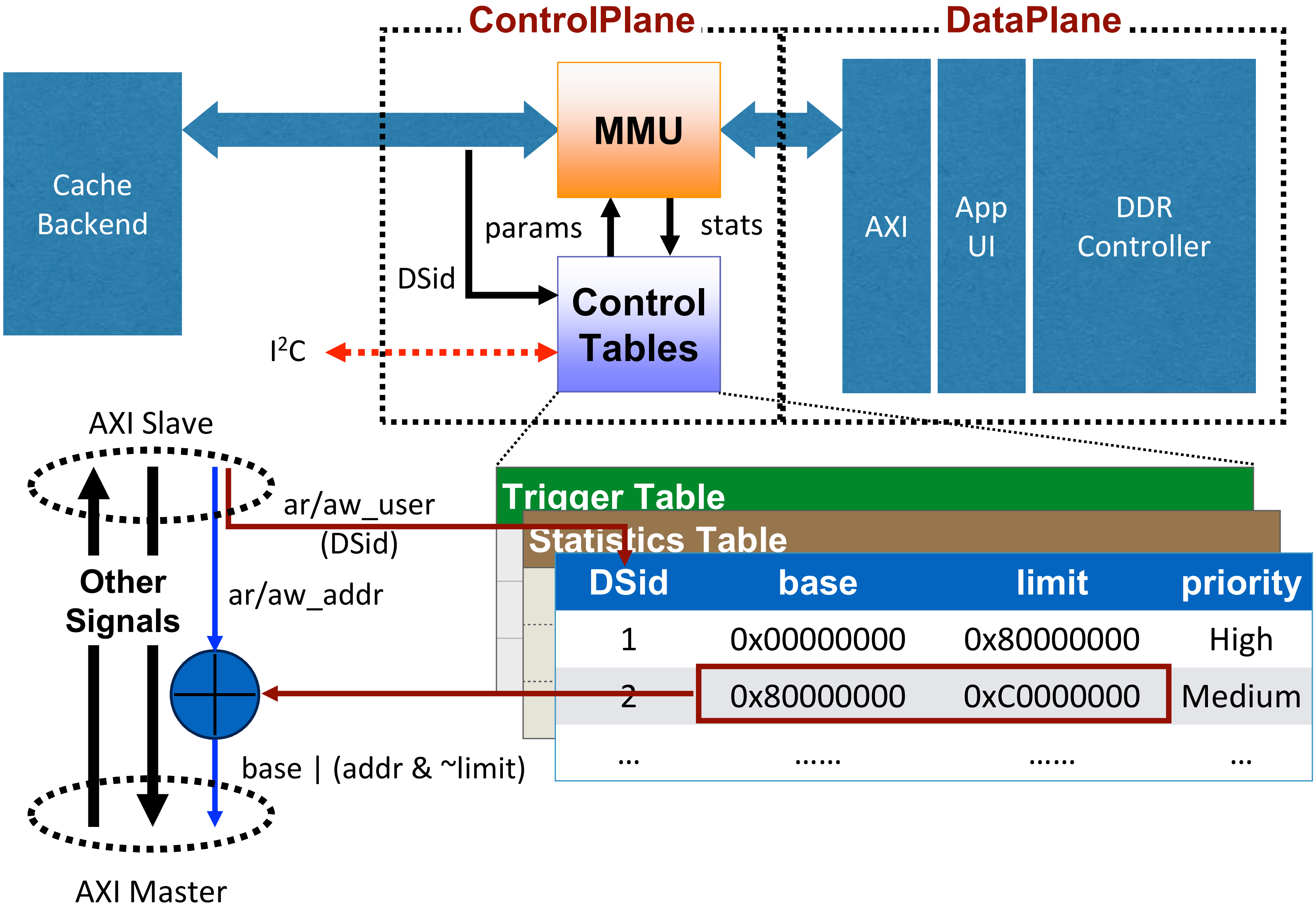
Price-Performance		
Performance component	**IBM SoftLayer**	**AWS**
Maximum requests per second (RPS)	21,765 RPS	16,079 RPS
Average requests per second	3,628 RPS	2,680 RPS
Cost per unit of work (RPS)	$46/average RPS	$119/average RPS

Messaging (network-intensive)

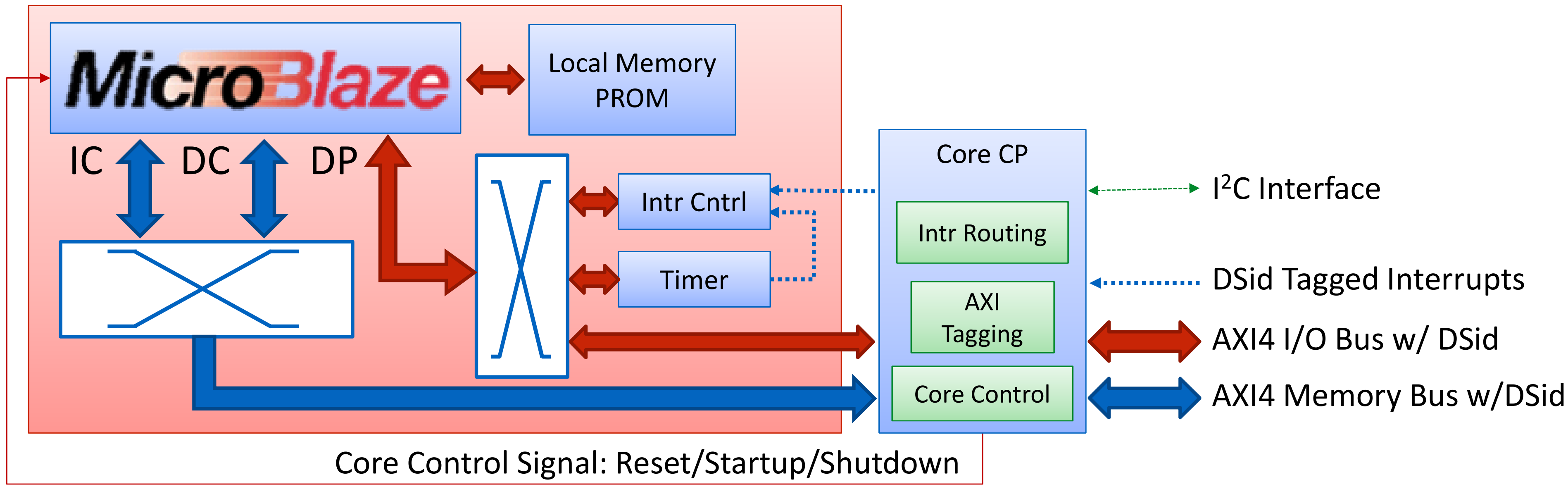
Price-Performance		
Component	**IBM SoftLayer**	**AWS EC2**
Total cost	$128,112	$225,179
Messages per second (MPS)	70,925 MPS	51,995 MPS
Cost per unit of work (MPS)	$1.81/MPS	$4.33/MPS

Bare-metal beats virt. by up to 40%

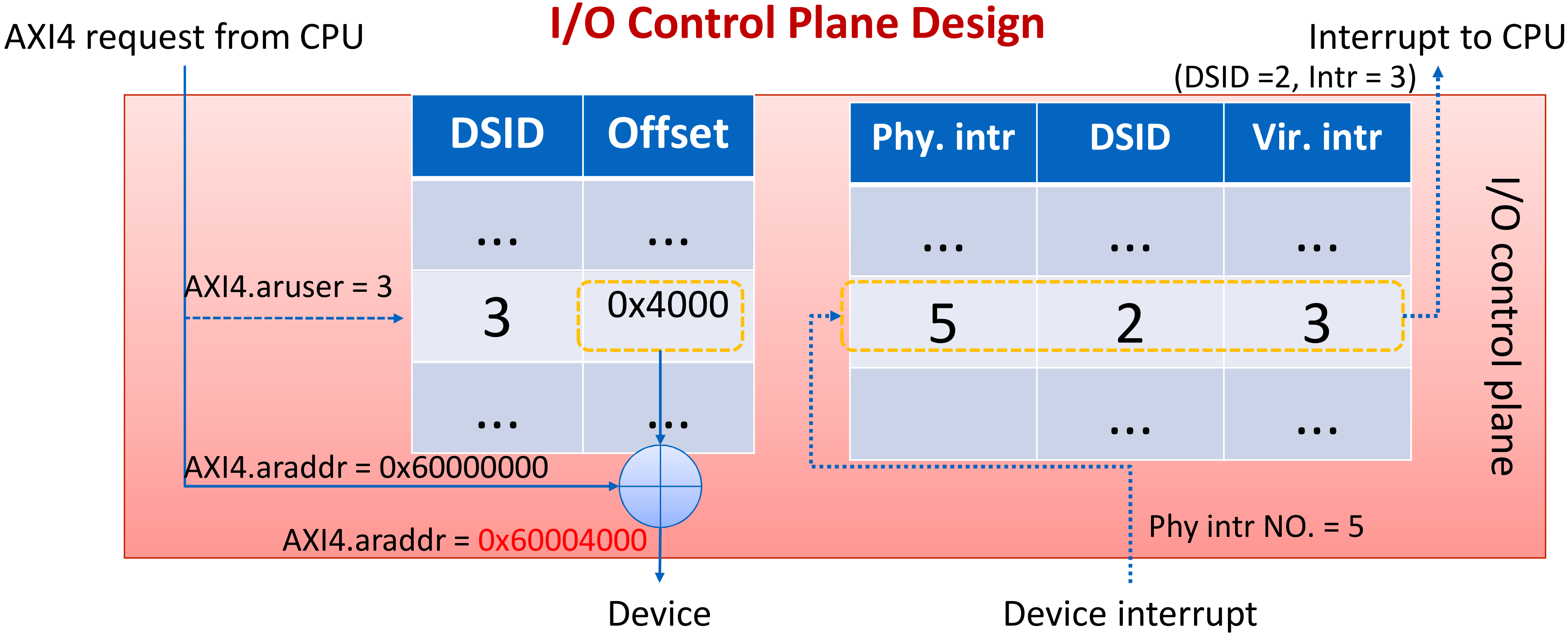
Address Mapping in DRAM CL



CPU核与I/O控制平面设计



I/O Control Plane Design



Bare-Metal Virtualization without Hypervisor

PRM startup LDom#1

```
root@prm_core_bd:~# sh kszh 1
Download required files from server ...
Connecting to 192.168.1.1 (192.168.1.1:80)
u-boot-s.bin         100% |*******************************************************************|   217k  0:00:00 ETA
Connecting to 192.168.1.1 (192.168.1.1:80)
314.ub               100% |*******************************************************************|  5225k  0:00:00 ETA
Connecting to 192.168.1.1 (192.168.1.1:80)
system-mv-eth.dtb    100% |*******************************************************************| 11114   0:00:00 ETA
Configure KLoader for logic domain ...
copying uboot using CDMA ...
1+1 records in
1+1 records out
copying kernel image using CDMA ...
40+1 records in
40+1 records out
copying device tree file using CDMA ...
0+1 records in
0+1 records out
startup ldom0 ...
Run bootm 0x84000000 - 0x90000000 in uboot to startup system
root@prm_core_bd:~# ping 192.168.1.124
PING 192.168.1.124 (192.168.1.124): 56 data bytes
64 bytes from 192.168.1.124: seq=0 ttl=64 time=5.598 ms
64 bytes from 192.168.1.124: seq=1 ttl=64 time=2.506 ms
64 bytes from 192.168.1.124: seq=2 ttl=64 time=2.578 ms
64 bytes from 192.168.1.124: seq=3 ttl=64 time=2.534 ms
64 bytes from 192.168.1.124: seq=4 ttl=64 time=2.502 ms
```

LDom#1 w/ Ethernet
IP: 192.168.1.124

```
Creating /dev/flash/* device nodes
random: dd urandom read with 1 bits of entropy available
starting Busybox inet Daemon: inetd... done.
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
udhcpc (v1.22.1) started
Sending discover...
libphy: 48000000:01 - Link is Up - 1000/Full
Sending discover...
Sending select for 192.168.1.124...
Lease of 192.168.1.124 obtained, lease time 600
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
done.

Built with PetaLinux v2014.4 (Yocto 1.7) fsg_mbcore_bd /dev/ttyUL0
fsg_mbcore_bd login: root
Password:
login[313]: root login on 'ttyUL0'
root@fsg_mbcore_bd:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:A2:01
          inet addr:192.168.1.124  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2408 (2.3 KiB)  TX bytes:1172 (1.1 KiB)

root@fsg_mbcore_bd:~#
```

LDom#2 w/ Ethernet, ip: 192.168.1.125
download file from server

```
INIT: Entering runlevel: 5
Configuring network interfaces... net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
udhcpc (v1.22.1) started
Sending discover...
libphy: 48000000:01 - Link is Up - 1000/Full
Sending discover...
Sending select for 192.168.1.125...
Lease of 192.168.1.125 obtained, lease time 600
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
done.

Built with PetaLinux v2014.4 (Yocto 1.7) fsg_mbcore_bd /dev/ttyUL0
fsg_mbcore_bd login: root
Password:
login[313]: root login on 'ttyUL0'
root@fsg_mbcore_bd:~# wget 192.168.1.1/yzh/test
Connecting to 192.168.1.1 (192.168.1.1:80)
random: nonblocking pool is initialized
test                 100% |********************************| 65536k  0:00:00 ETA
root@fsg_mbcore_bd:~#
```

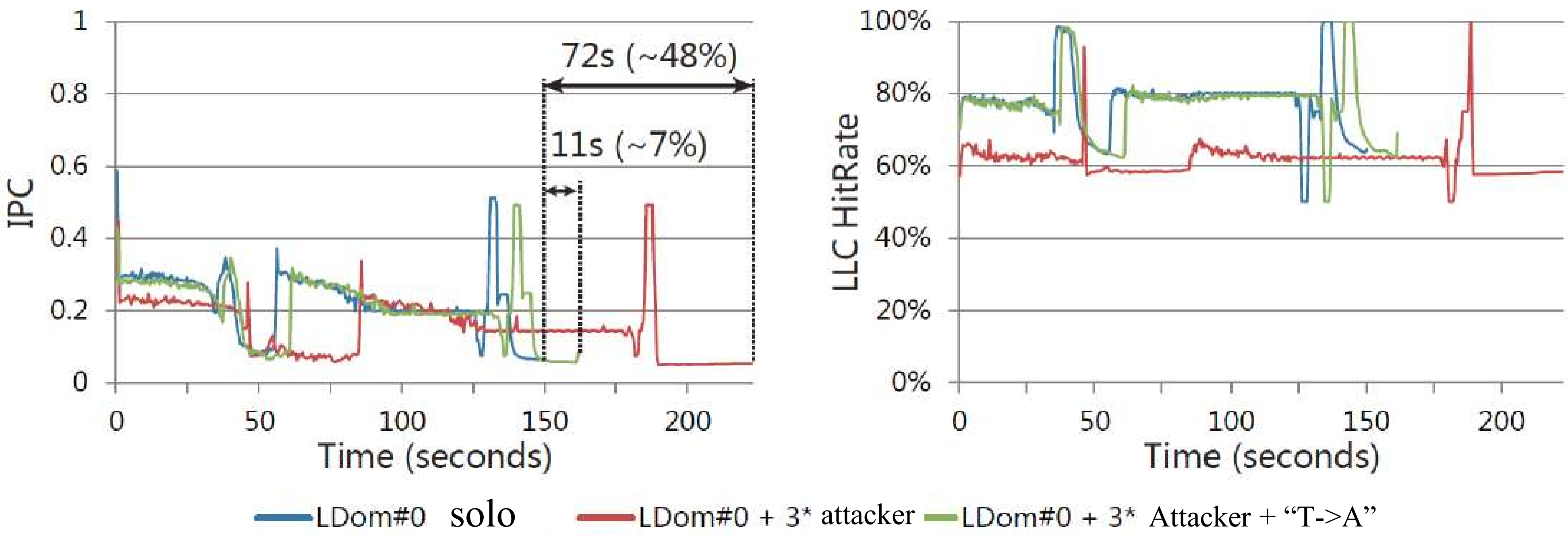
LDom#3 w/o Ethernet
check cpu&memory&kernel

```
root@fsg_mbcore_bd:~# free
             total         used         free       shared      buffers
Mem:        514044        15276       498768            0            0
-/+ buffers:              15276       498768
Swap:
root@fsg_mbcore_bd:~# uname -a
Linux fsg_mbcore_bd 3.14.2 #3 Tue Sep 8 09:54:18 CST 2015 microblaze GNU/Linux
root@fsg_mbcore_bd:~#
```

512MB memory, Linux-3.14.2

Case 2: Cache Partitioning

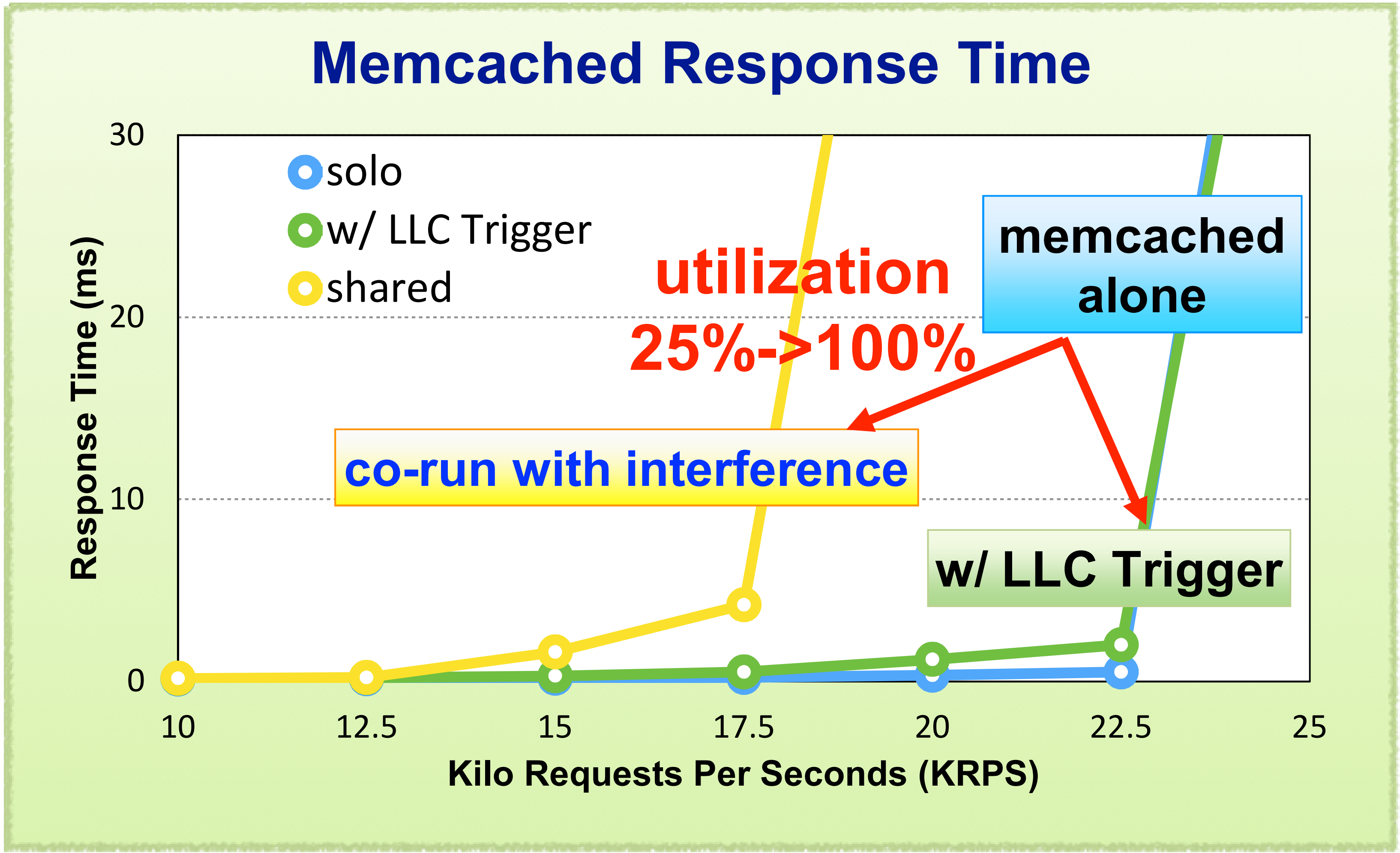
- 4 Ldoms: 1 X 429.mcf + 3 X Attacker
- Allocate different LLC capacities
- Perf. degradation: **7% vs. 48%**



Improve Utilization w/o Loss of QoS

CPU Utilization 4X

- Memcached： Tail Latency <1.5ms

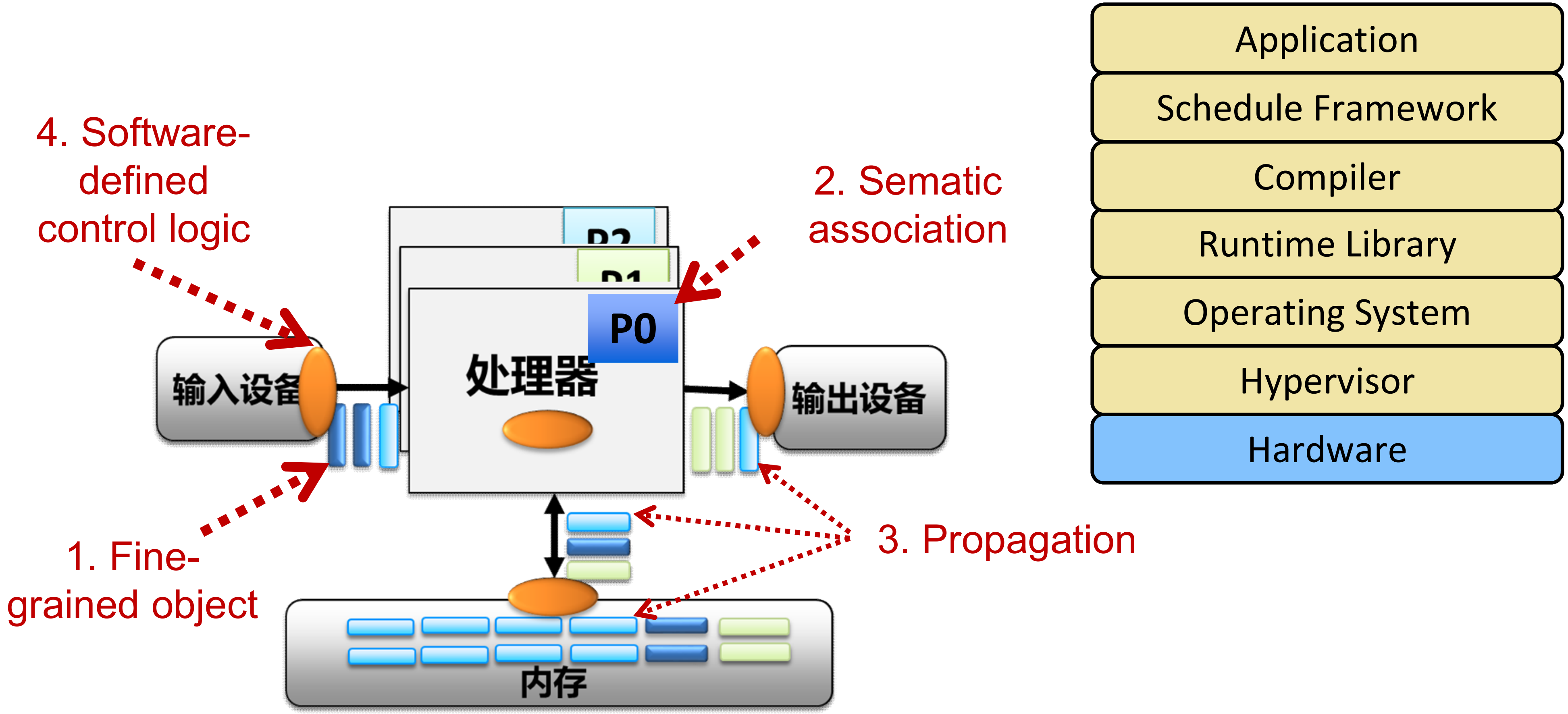


Labeled RISC-V

- Hardware – more exploration
- Software – better ecosystem
- Goal – establish the **labeled RISC-V** branch

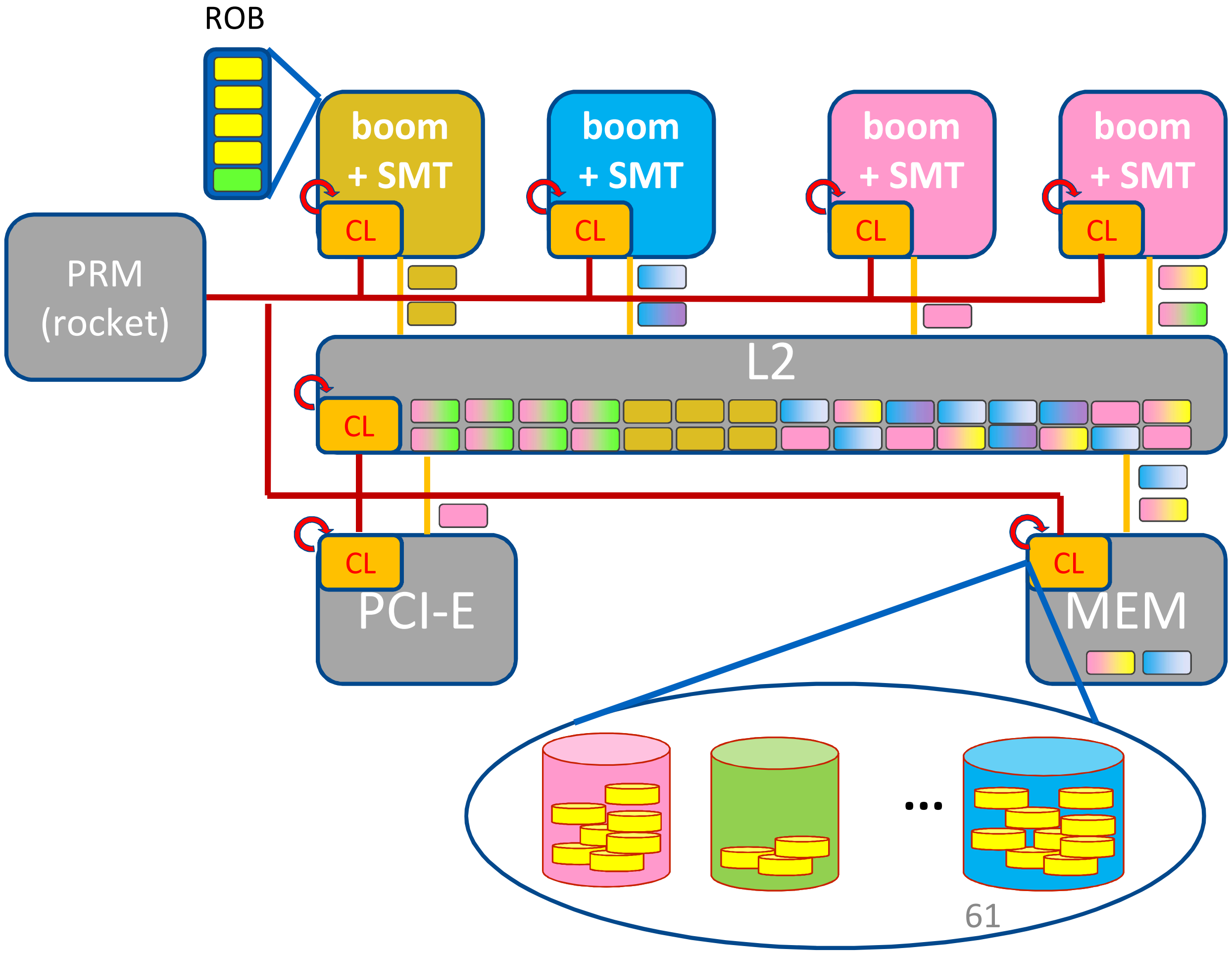
Hardware - LvNA

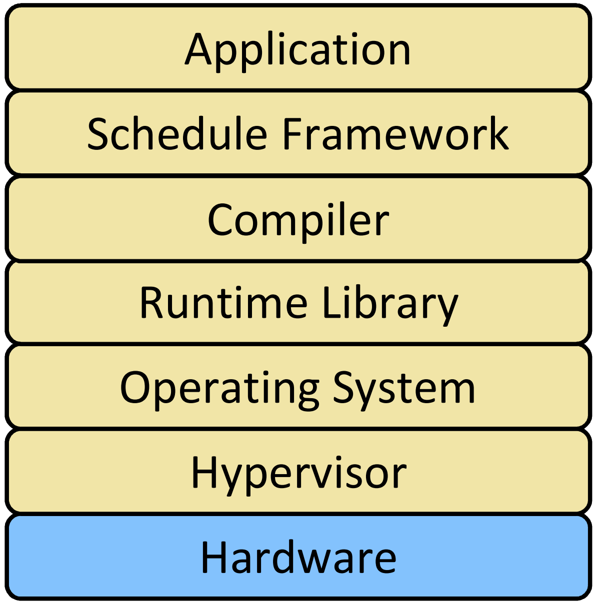
- Labeled von Neumann Architecture
  - Extend PARD to all resources



Hardware - LvNA

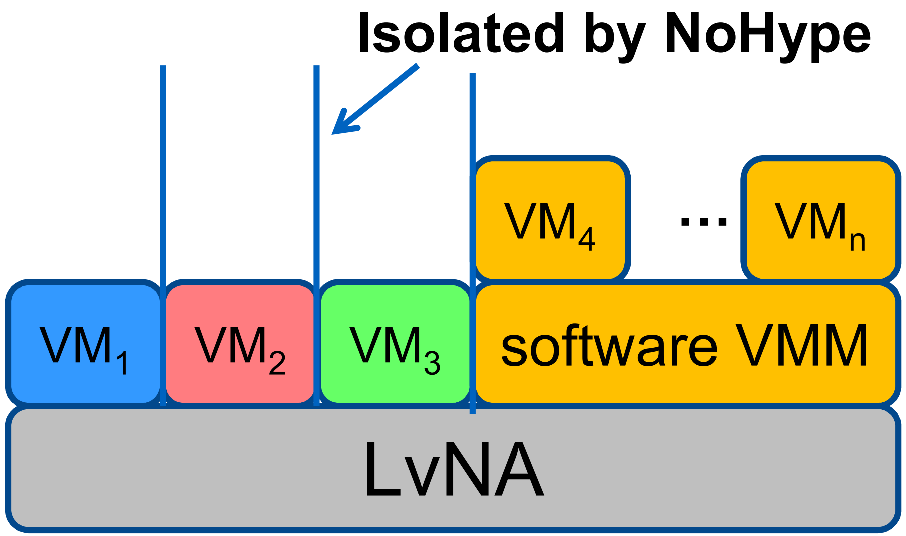
- Labeled von Neumann Architecture
  - Extend PARD to all resources

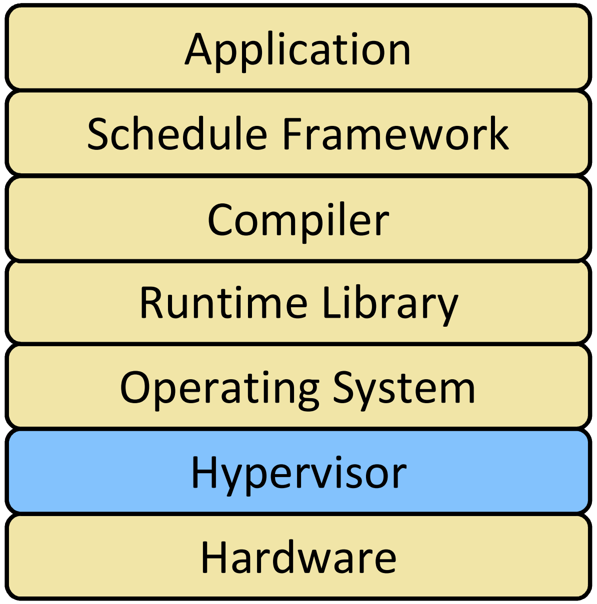




Hypervisor - NoHype **Finished**

- Push the software hypervisor down to LvNA
  - Remove run-time overhead





NoHype Example

Partition #1

```
root@prm_core_bd:~# sh kszh 1
Download required files from server ...
Connecting to 192.168.1.1 (192.168.1.1:80)
u-boot-s.bin         100% |*******************************************************************|
Connecting to 192.168.1.1 (192.168.1.1:80)
314.ub               100% |*******************************************************************|
Connecting to 192.168.1.1 (192.168.1.1:80)
system-mv-eth.dtb    100% |*******************************************************************|
Configure KLoader for logic domain ...
copying uboot using CDMA ...
1+1 records in
1+1 records out
copying kernel image using CDMA ...
40+1 records in
40+1 records out
copying device tree file using CDMA ...
0+1 records in
0+1 records out
startup ldom0 ...
Run bootm 0x84000000 0x00000000 in uboot to startup system
root@prm_core_bd:~# ping 192.168.1.124
PING 192.168.1.124 (192.168.1.124): 56 data bytes
64 bytes from 192.168.1.124: seq=0 ttl=64 time=5.598 ms
64 bytes from 192.168.1.124: seq=1 ttl=64 time=2.506 ms
64 bytes from 192.168.1.124: seq=2 ttl=64 time=2.578 ms
64 bytes from 192.168.1.124: seq=3 ttl=64 time=2.534 ms
64 bytes from 192.168.1.124: seq=4 ttl=64 time=2.502 ms
```

Partition #3

```
Creating /dev/flash/* device nodes
random: dd urandom read with 1 bits of entropy available
starting Busybox inet Daemon: inetd... done.
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
udhcpc (v1.22.1) started
Sending discover...
libphy: 48000000:01 - Link is Up - 1000/Full
Sending discover...
Sending select for 192.168.1.124...
Lease of 192.168.1.124 obtained, lease time 600
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
done.

Built with PetaLinux v2014.4 (Yocto 1.7) fsg_mbcore_bd /dev/ttyUL0
fsg_mbcore_bd login: root
Password:
login[313]: root login on 'ttyUL0'
root@fsg_mbcore_bd:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:A2:01
          inet addr:192.168.1.124  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2408 (2.3 KiB)  TX bytes:1172 (1.1 KiB)

root@fsg_mbcore_bd:~#
```

Linux-3.14.2

Partition #2

```
INIT: Entering runlevel: 5
Configuring network interfaces... net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
net eth0: Promiscuous mode disabled.
udhcpc (v1.22.1) started
Sending discover...
libphy: 48000000:01 - Link is Up - 1000/Full
Sending discover...
Sending select for 192.168.1.125...
Lease of 192.168.1.125 obtained, lease time 600
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
done.

Built with PetaLinux v2014.4 (Yocto 1.7) fsg_mbcore_bd /dev/ttyUL0
fsg_mbcore_bd login: root
Password:
login[313]: root login on 'ttyUL0'
root@fsg_mbcore_bd:~# wget 192.168.1.1/yzh/test
Connecting to 192.168.1.1 (192.168.1.1:80)
random: nonblocking pool is initialized
test                 100% |********************************|  65536k  0:00:00 ETA
root@fsg_mbcore_bd:~#
```

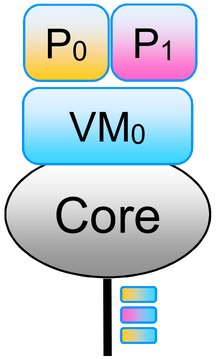
Partition #4

```
root@fsg_mbcore_bd:~# free
              total         used         free       shared      buffers
Mem:         514044        15276       498768            0            0
-/+ buffers:               15276       498768
Swap:             0            0            0
root@fsg_mbcore_bd:~# uname -a
Linux fsg_mbcore_bd 3.14.2 #3 Tue Sep 8 09:54:18 CST 2015 microblaze GNU/Linux
root@fsg_mbcore_bd:~#
```

Operating System - Fine-grained labeling

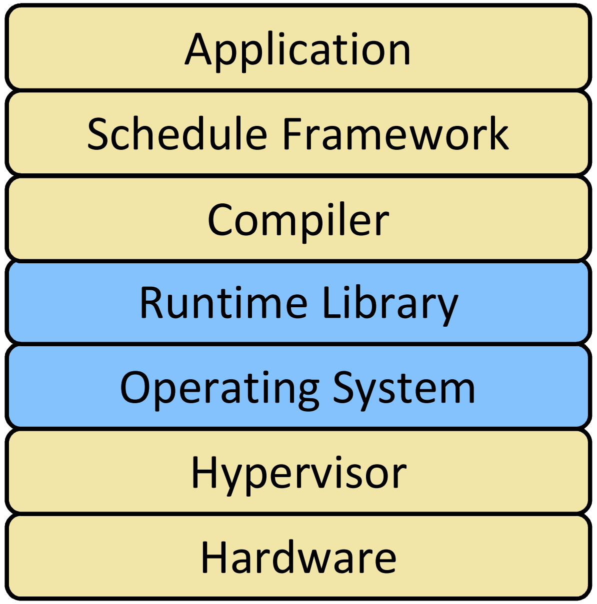
- Add fine-grained label as context resource
  - Process **Finished**
    - Process/container-level
    - Thread-level
  - Address space
    - Function-level
    - Object-level
- Provide libraries
  - pthread_create_with_dsid()
  - malloc_with_dsid()

Process relative labeling



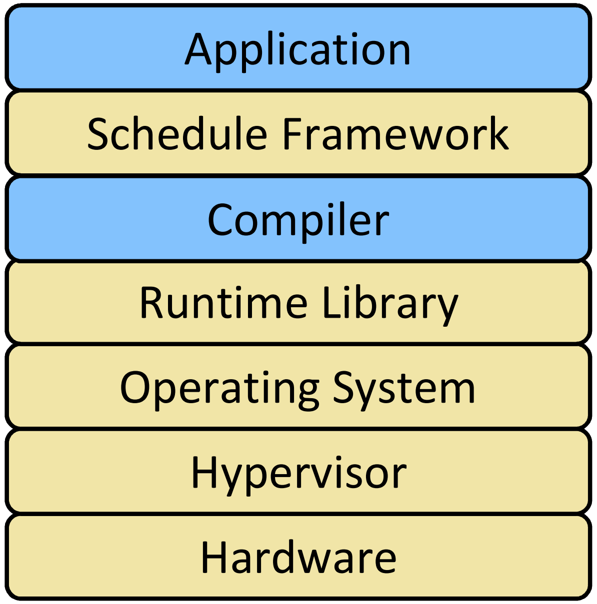
dsid	start	end
1	0x8000	0xffff
3	0x2000	0x27ff

Address space relative labeling

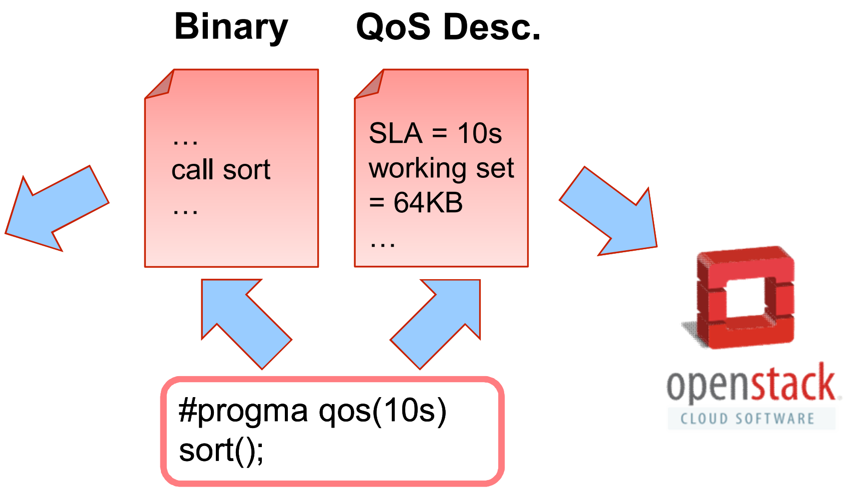


Compiler - collect QoS info. from prog

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

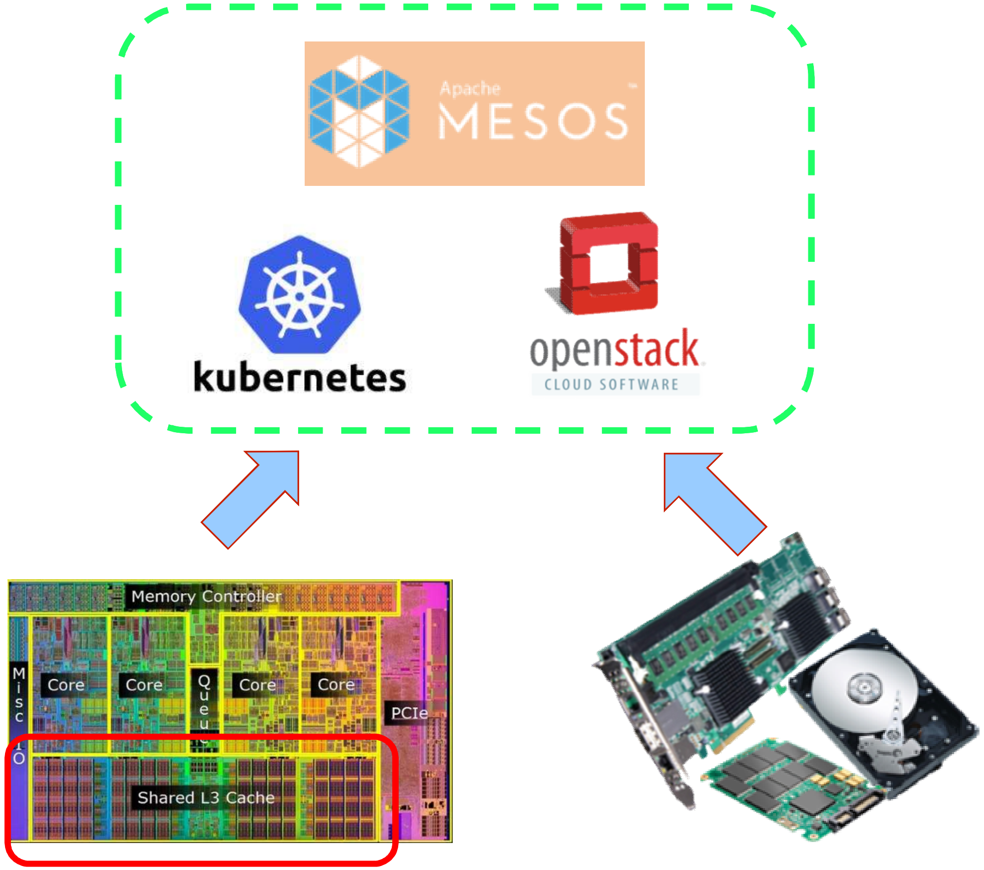


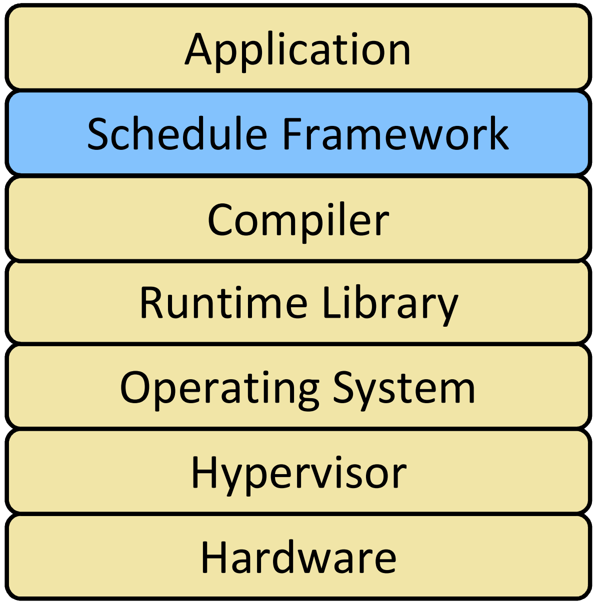
dsid	start	end
1	0x8000	0xffff
3	0x2000	0x27ff



Sche. Framework - QoS resource schedule

- Expose QoS resources to schedule frameworks
  - Integrate QoS resources into OpenStack **Finished**

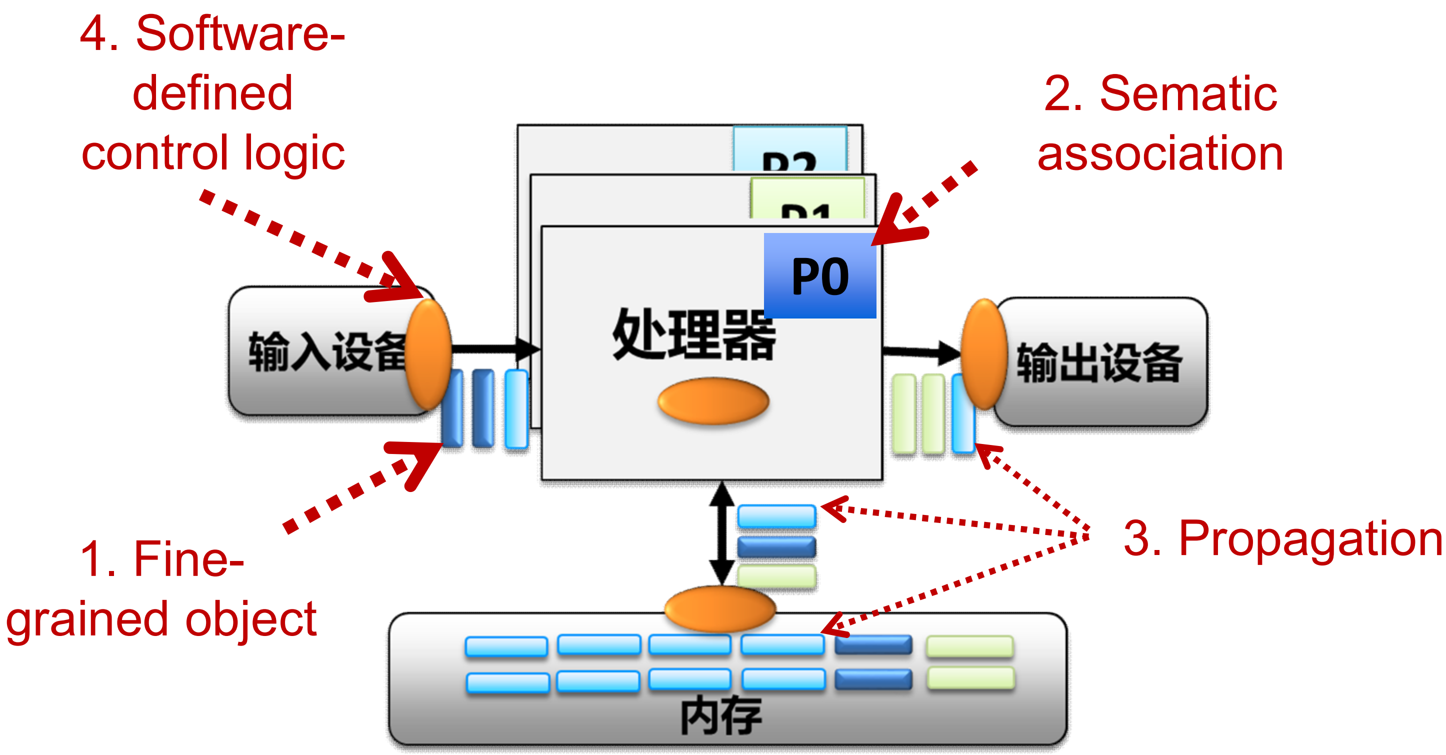




Open Problems

- **Theory**：How does LvNA impact on RAM, PRAM, LogP models?
- **Hardware/Arch:** How to implement LvNA at in CPU, memory, storage, networking?
- **Programing Model and Compilers**：How to express users' requirements and propagate to the hardware via labels? How to make compilers support labels?
- **OS/Hypervisor**：How to correlate labels with VMs, containers, processors, threads? How to abstract programming interfaces for 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?

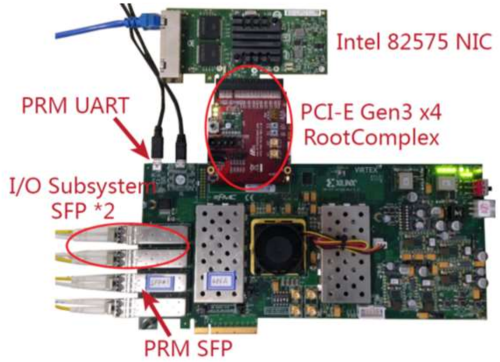
Summary

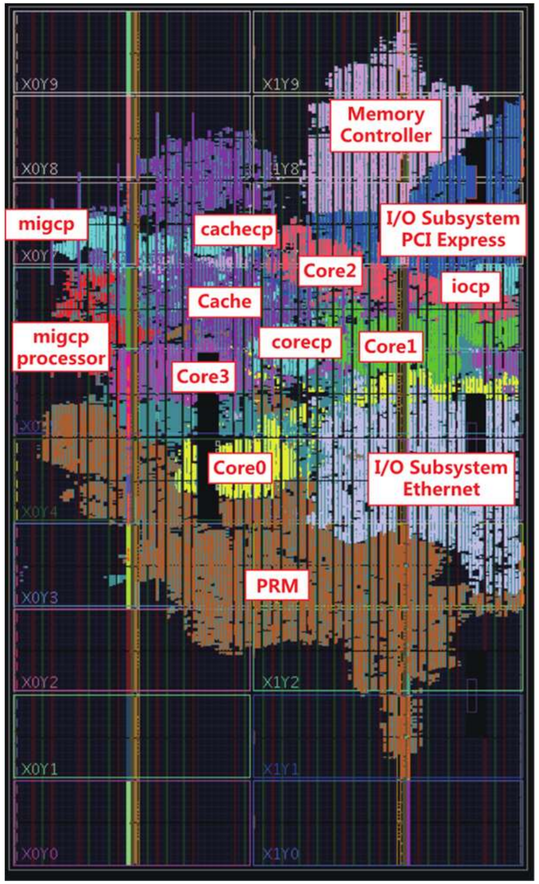


- **QoS:** extremely important for improving utilization
- **LvNA:** a model of software-defined architecture
- **PARD:** a proof of concept of LvNA

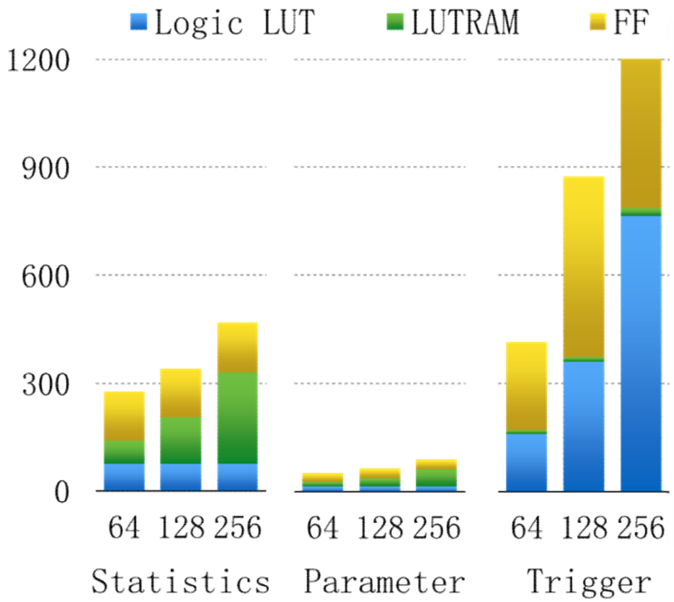
Thanks

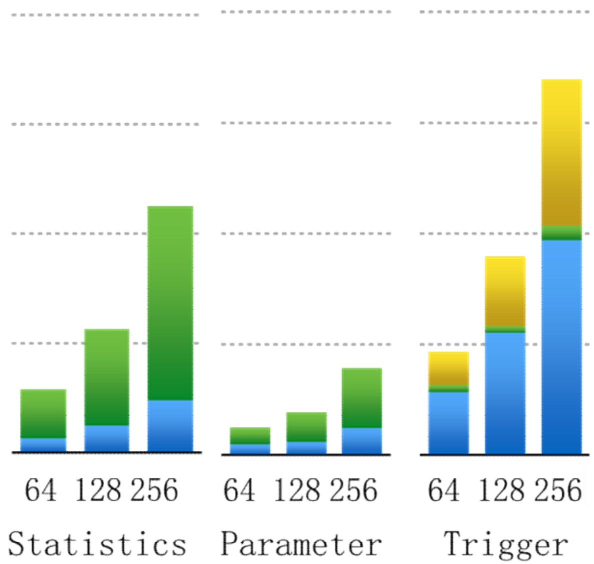
Overhead of Control Logic





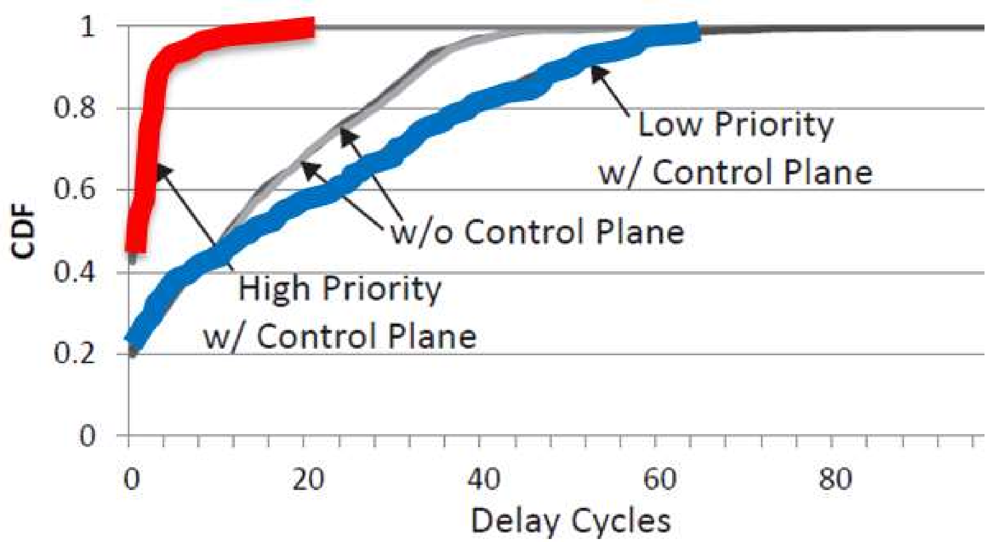
Memory Controller: 10.1% **LLC: 3.5%**



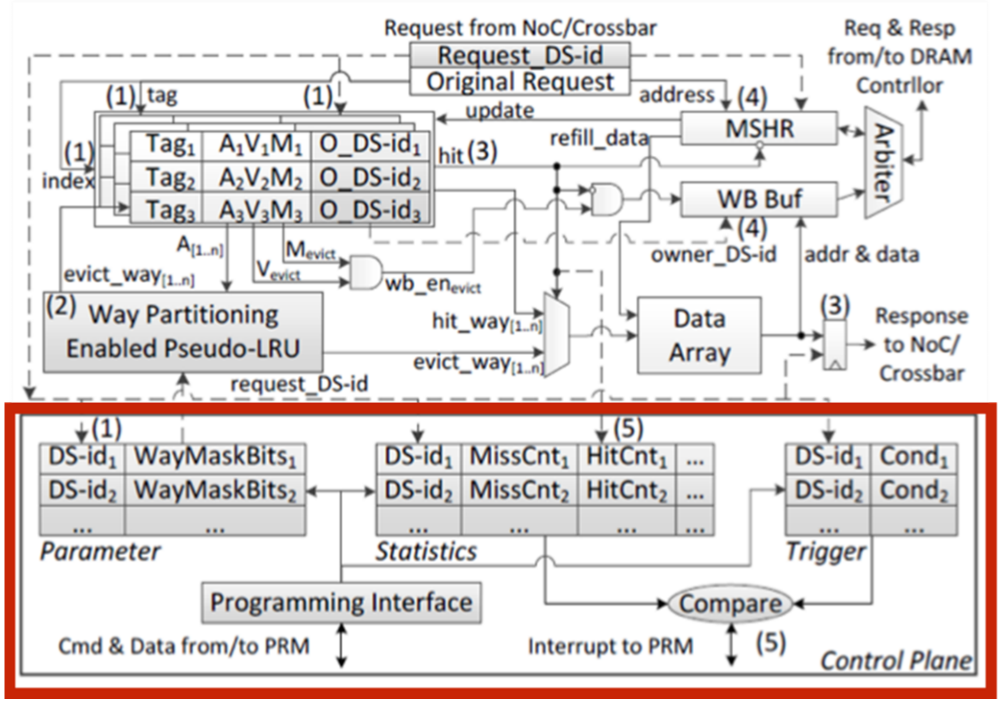


Extra Delay Analysis

- **Memory Controller**: CL significantly reduces queuing delay of high-priority requests by **5.6X**



- **Cache**: CL's logic can be hidden in the pipeline of caches.



Cache CL: No Extra delay

- CL operations are hidden in the pipeline of a write Request