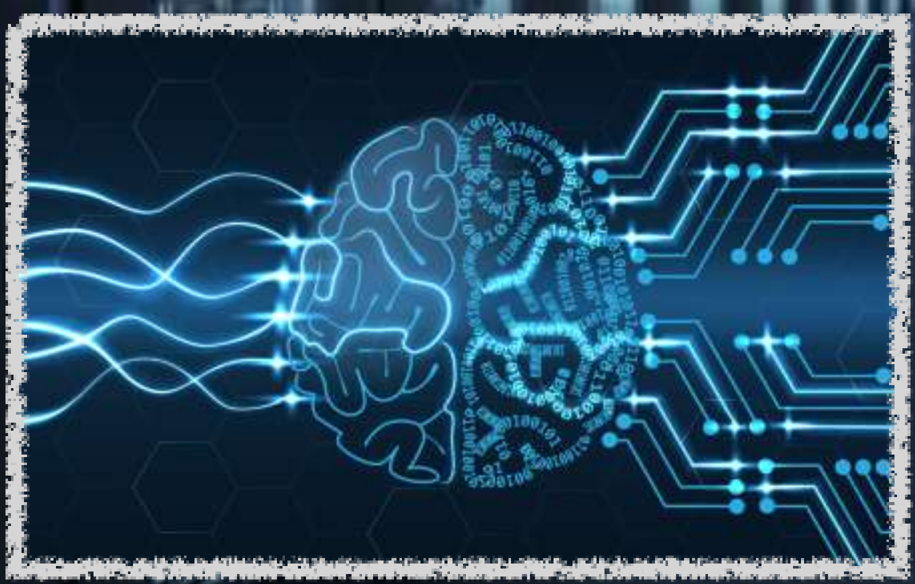
Farewell to Servers:
Hardware, Software, and Network Approaches towards Datacenter Resource Disaggregation

Yiying Zhang

Datacenter

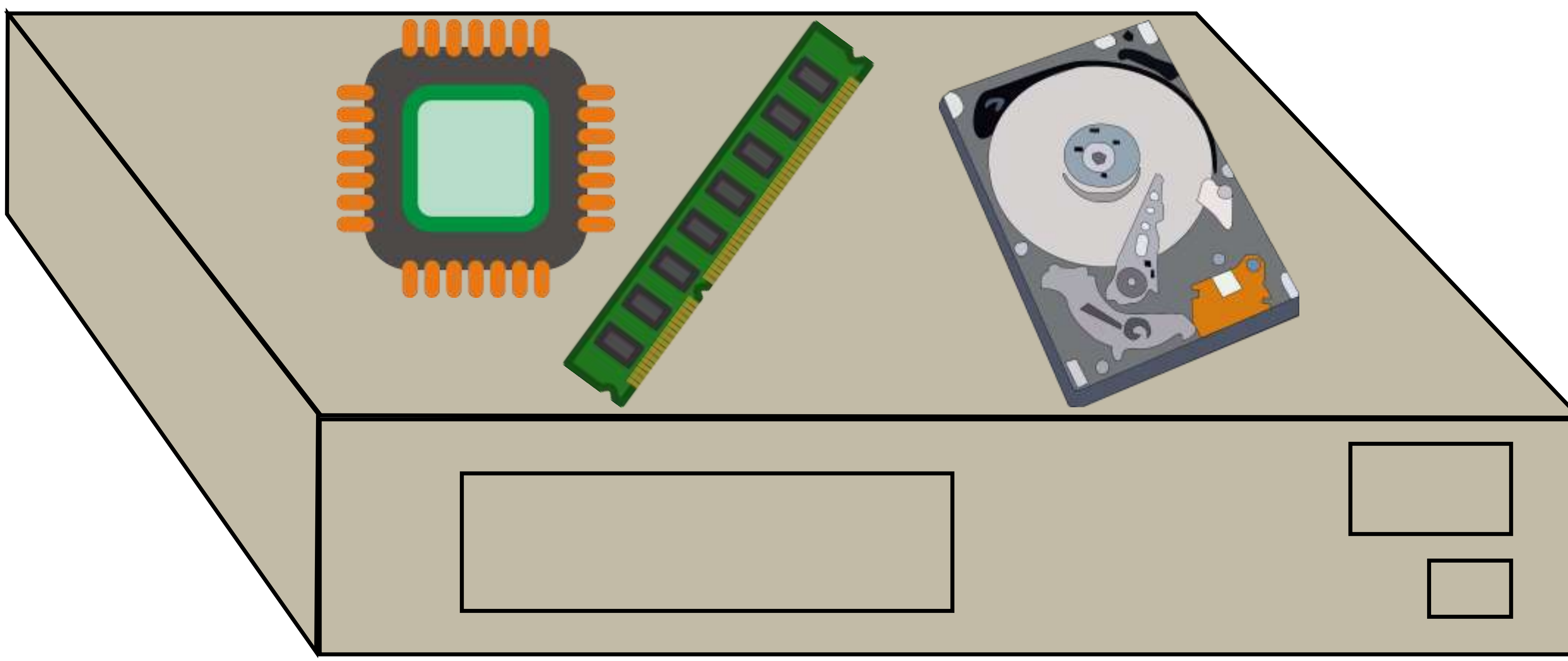

MUSIC
VIDEOS
SHARE
CLOUD STORAGE
DOCUMENTS
CONTACTS
FILES
PICTURES




Big Data Analytics

Monolithic Computer

OS / Hypervisor



Can monolithic servers continue to meet datacenter needs?

Application

Hardware

Heterogeneity

Flexibility

Perf / $


RoHS

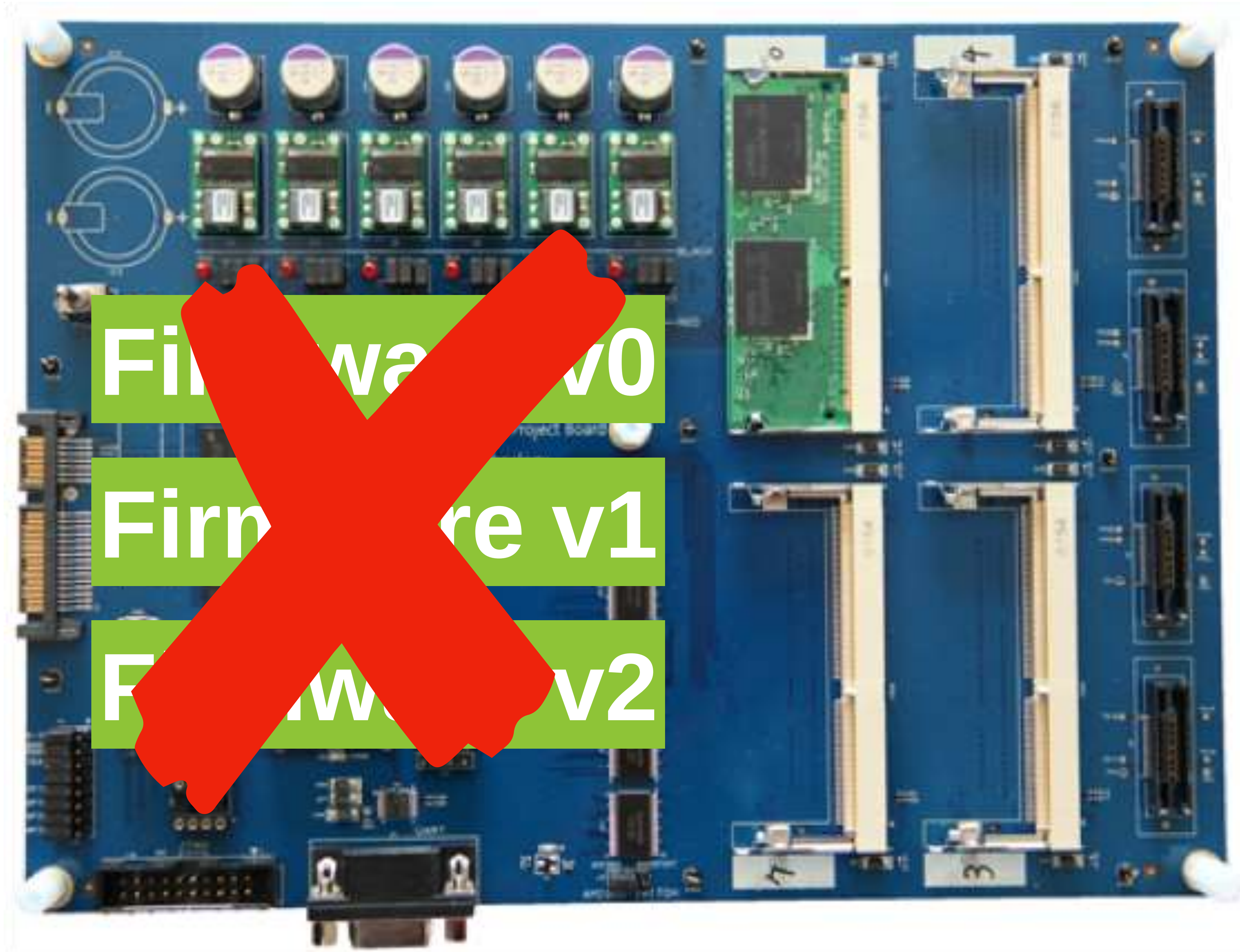
Making new hardware work with existing servers is like fitting puzzles

Tons of Issues

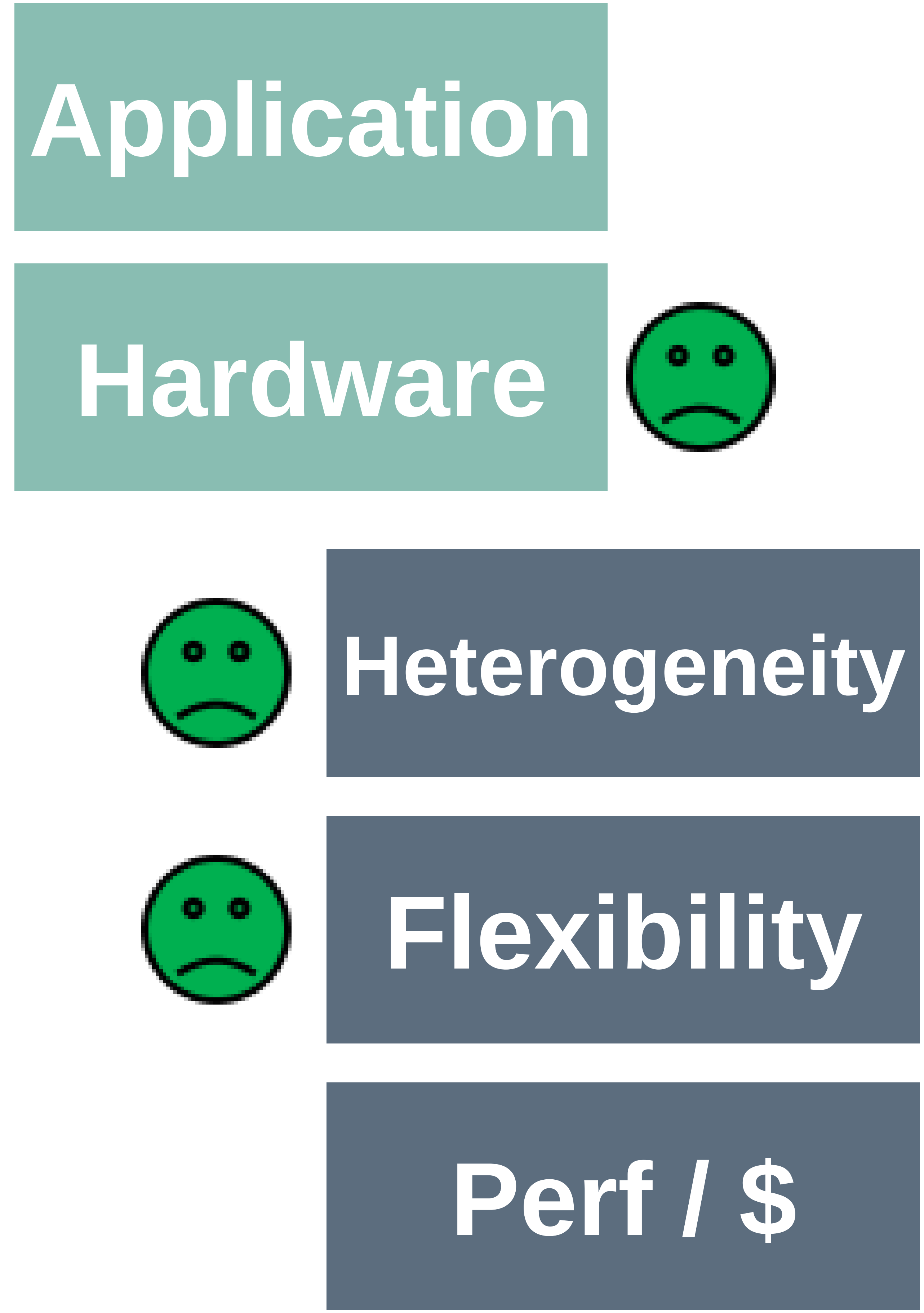
- Fitting bus standards
- Slots for new devices?
- Power for new devices?
- Changing bus?
- Software for new devices?

6 months of my life went to

SATA

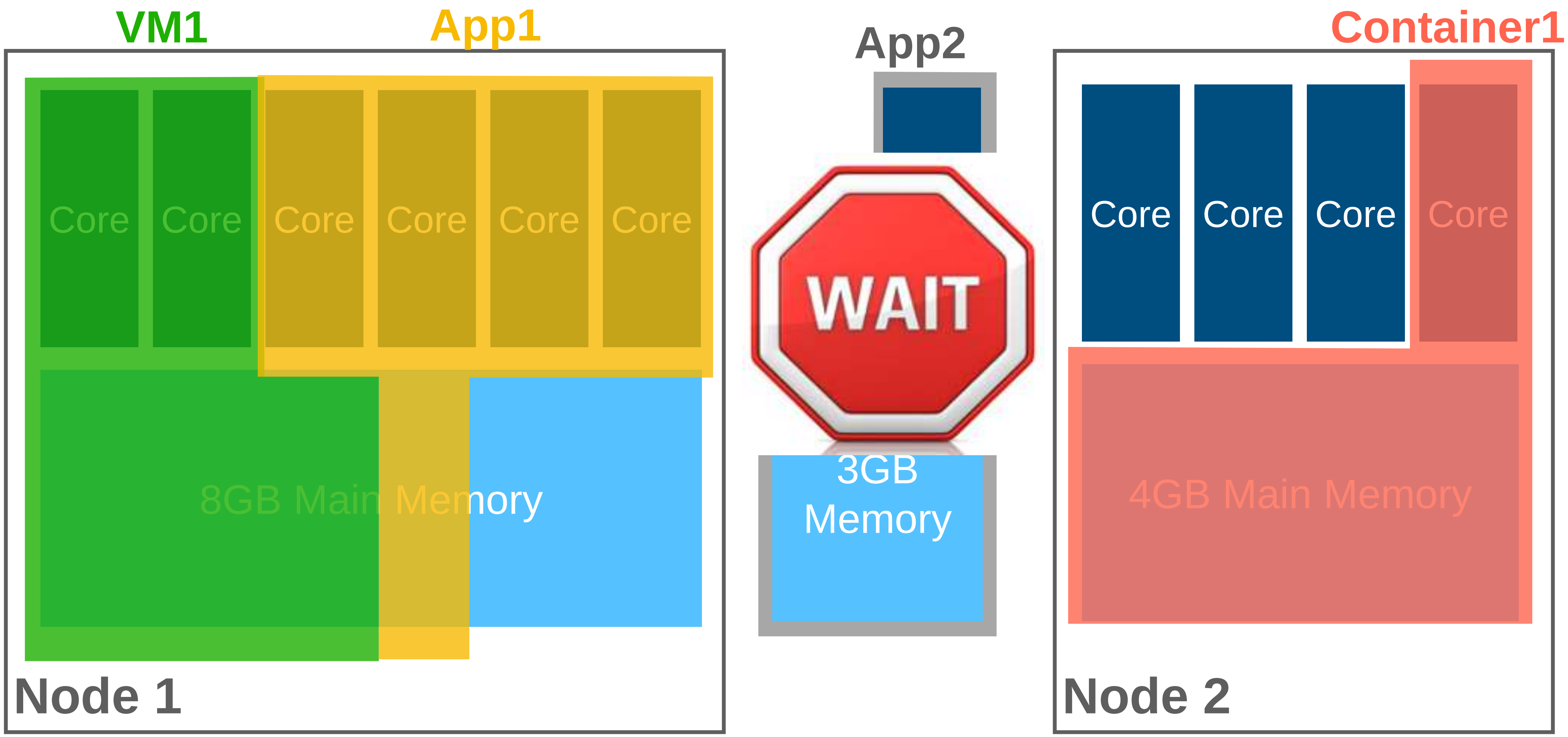


Can monolithic servers continue to meet datacenter needs?

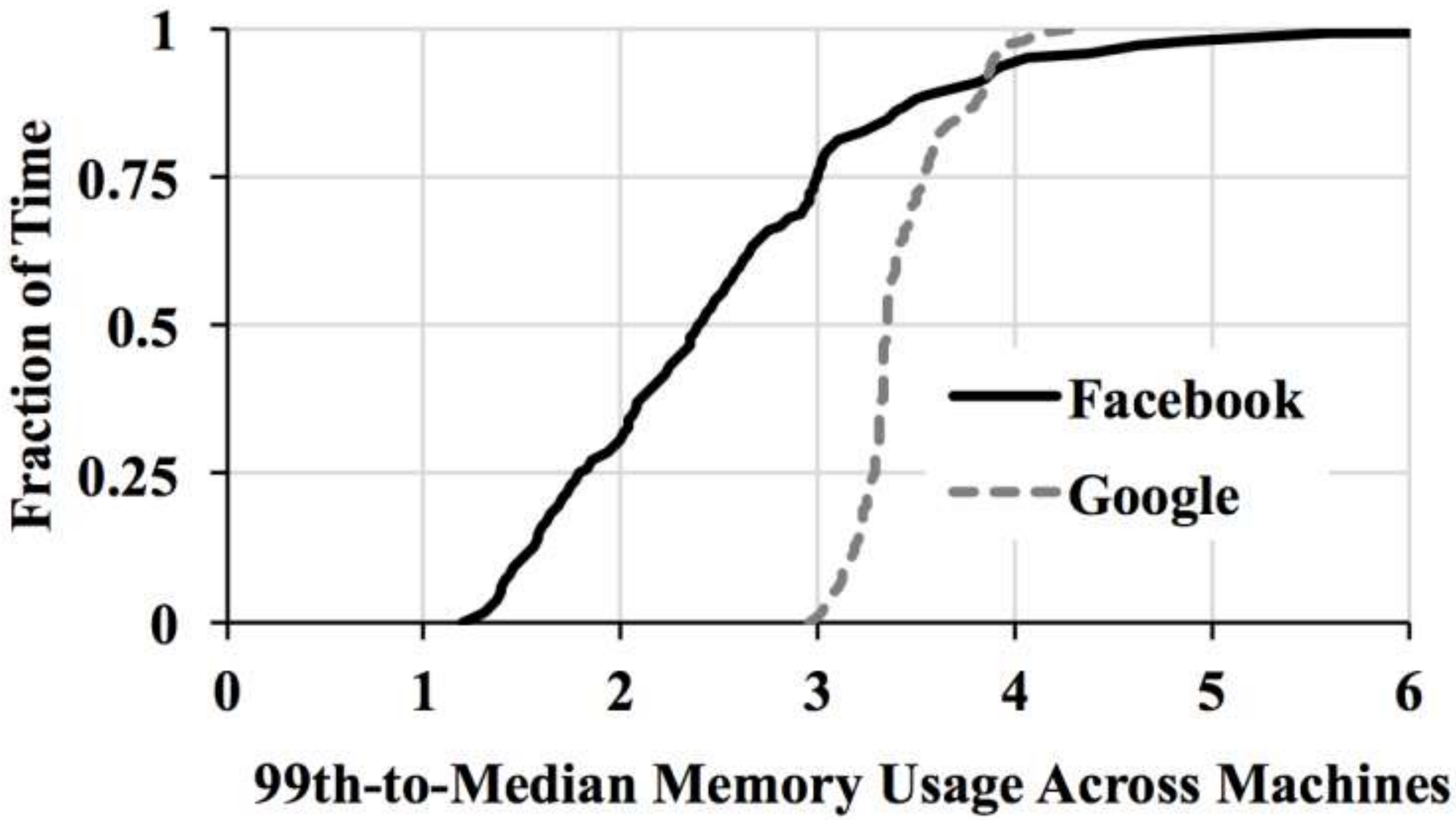


Application Elasticity and Perf / $

- Whole VM/container has to run on **one physical machine**
  - Move current applications to make room for new ones



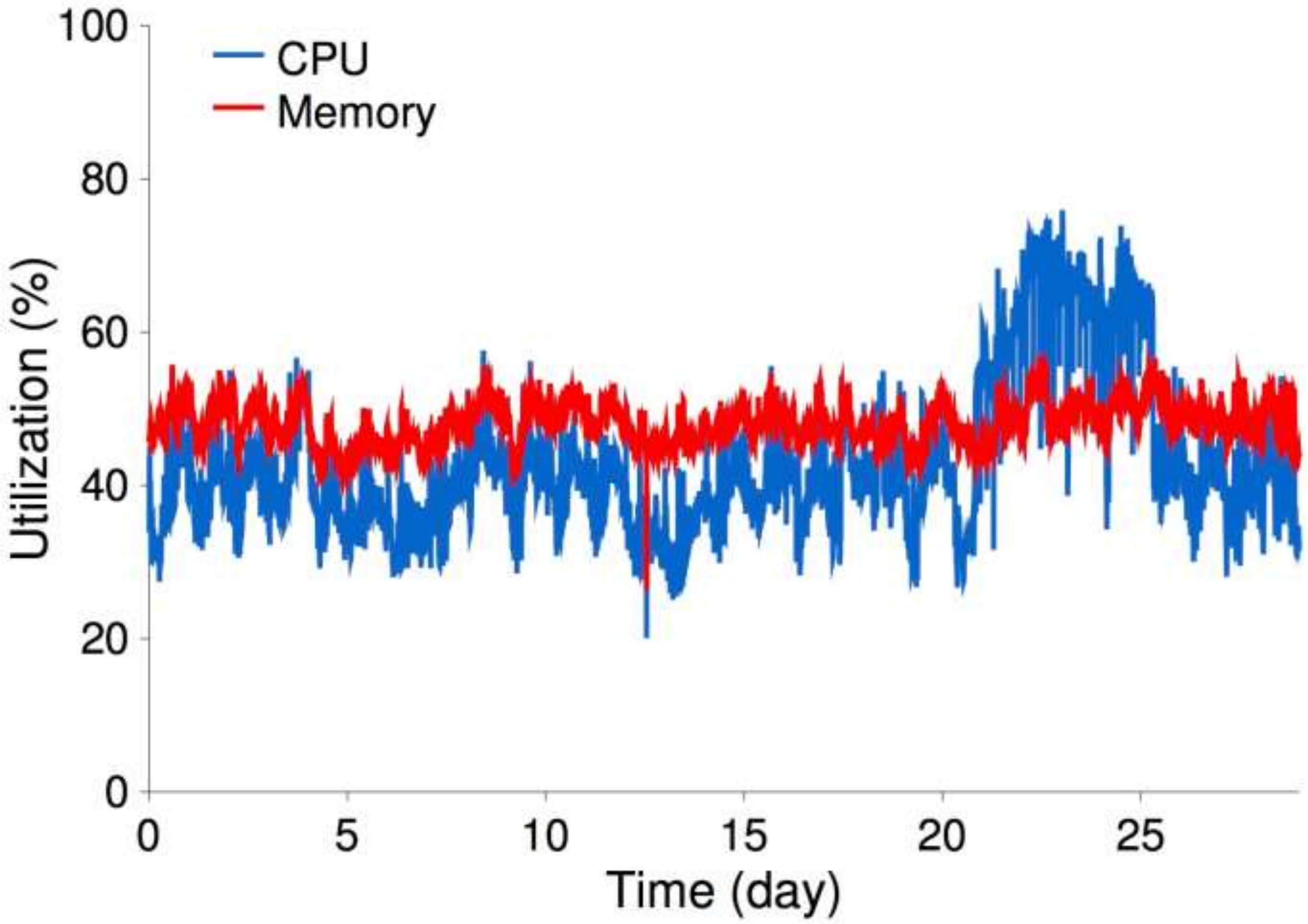
CPU/Memory Usages across Machines and across Jobs

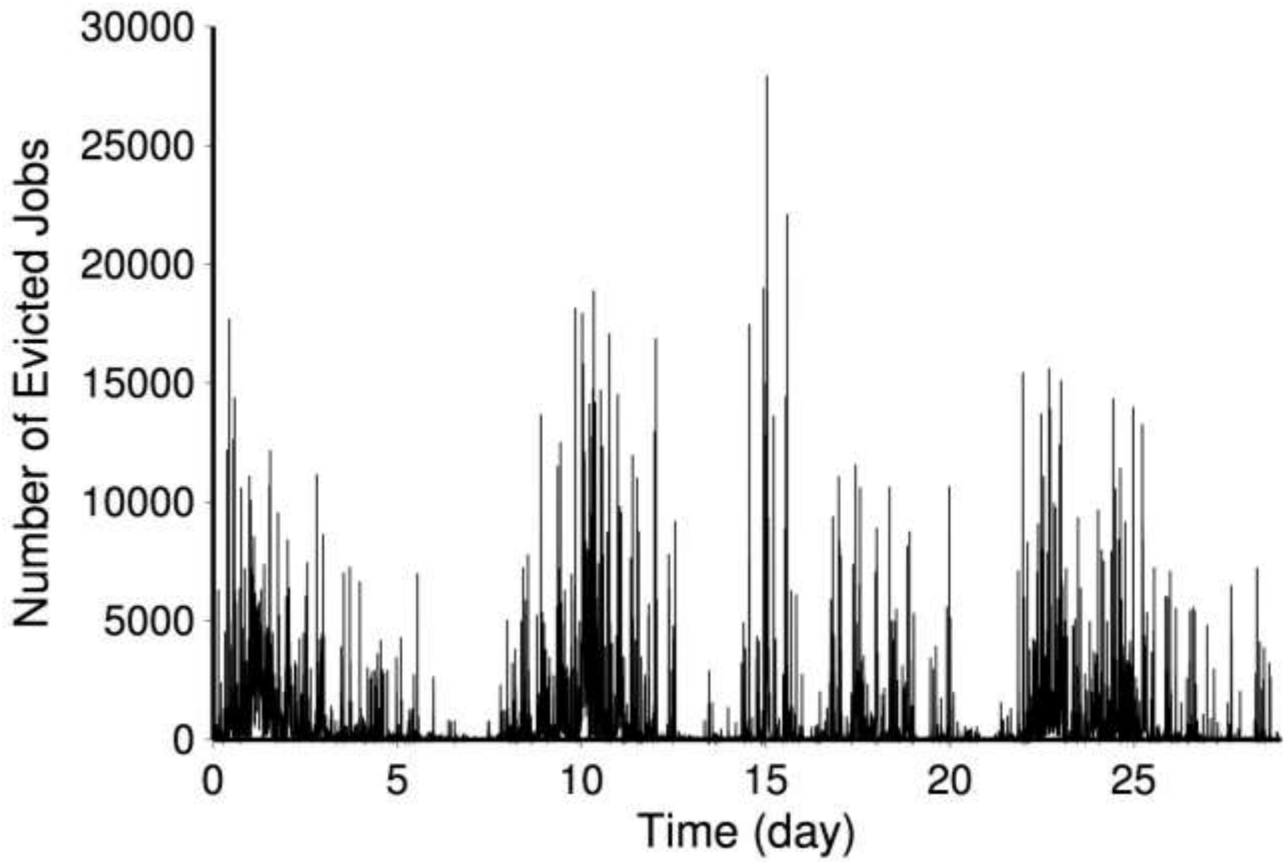


Source: Gu et al. “Efficient Memory Disaggregation with Infiniswap” NSDI’17

Memory usage in datacenter are highly imbalanced

Resource Utilization in Production Clusters





* Google Production Cluster Trace Data. “https://github.com/google/cluster-data”

Unused Resource + Waiting/Killed Jobs Because of Physical-Node Constraints

No fine-grained failure handling

Can monolithic servers continue to meet datacenter needs?

Application

Hardware

Heterogeneity

Flexibility

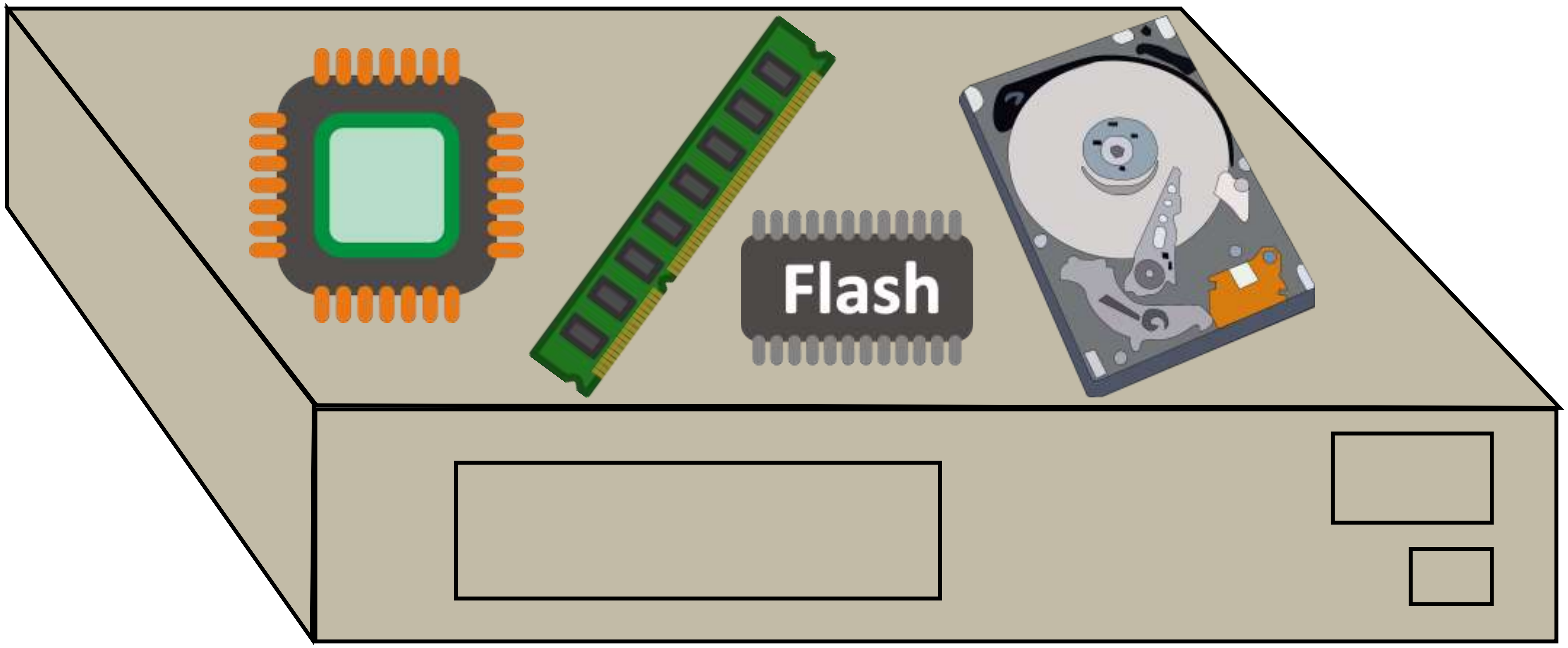
Perf / $

How to achieve better heterogeneity, flexibility, and perf/$?

Go beyond physical node boundary

Resource Disaggregation:

Breaking monolithic servers into network-attached, independent hardware components


Flash

Berkeley
UNIVERSITY OF CALIFORNIA

Hewlett Packard Enterprise

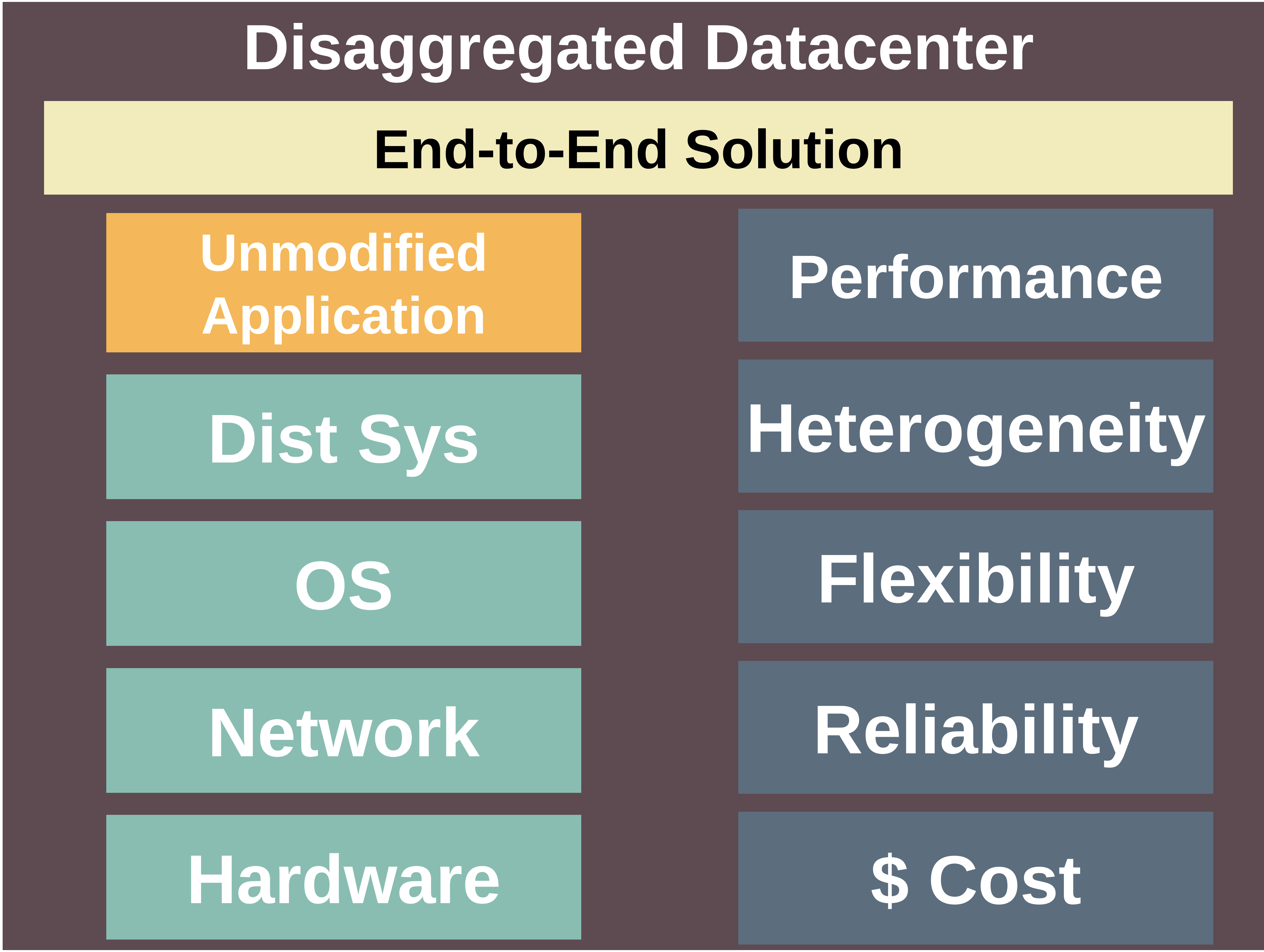
Hardware

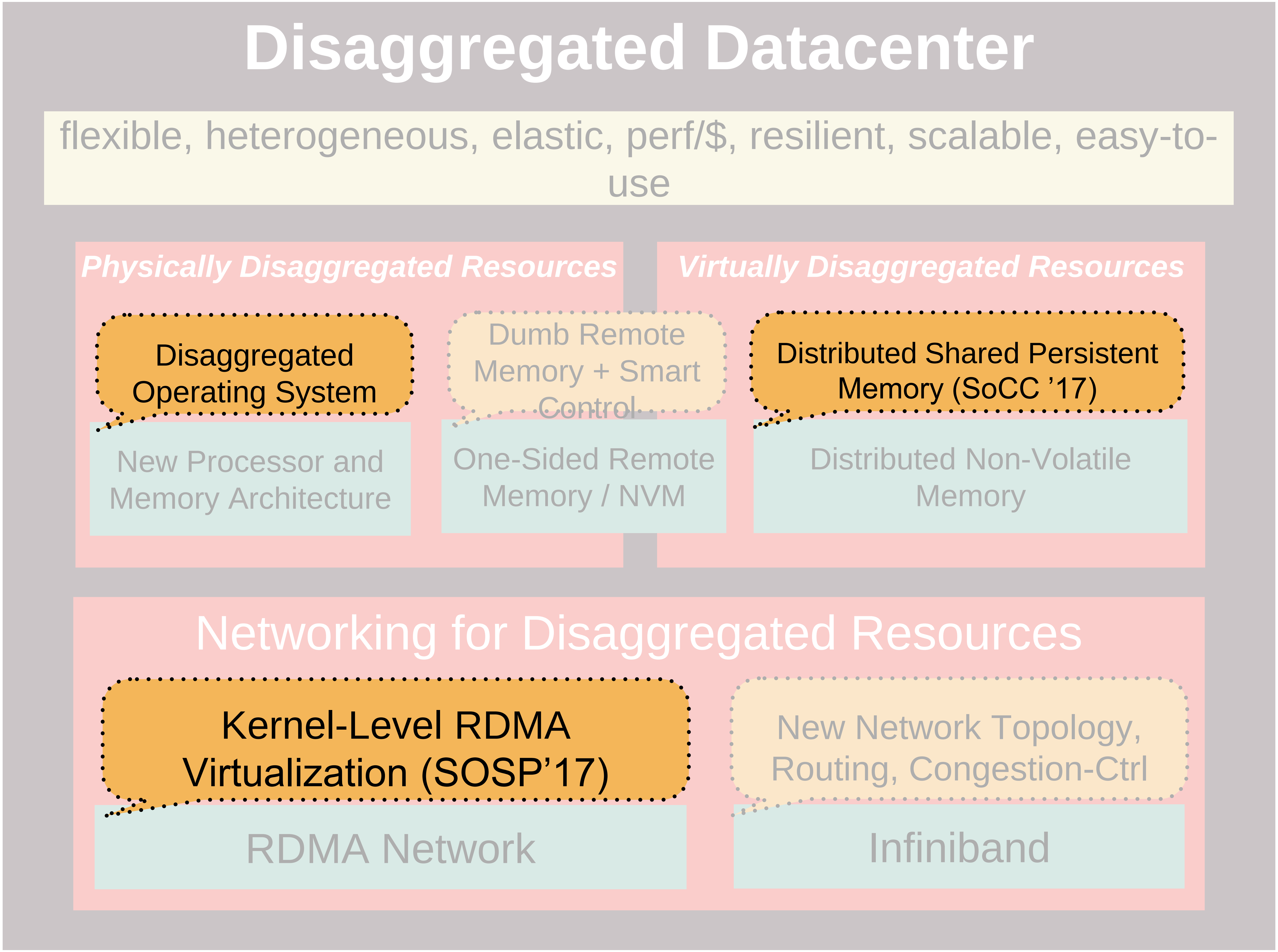
Microsoft

Network

vmware

Distributed Memory


Disaggregated Datacenter
End-to-End Solution
Unmodified Application
Dist Sys
OS
Network
Hardware
Performance
Heterogeneity
Flexibility
Reliability
$ Cost



Disaggregated Datacenter
flexible, heterogeneous, elastic, perf/$, resilient, scalable, easy-to-use
Physically Disaggregated Resources
Virtually Disaggregated Resources
Disaggregated Operating System
Dumb Remote Memory + Smart Control
Distributed Shared Persistent Memory (SoCC ’17)
New Processor and Memory Architecture
One-Sided Remote Memory / NVM
Distributed Non-Volatile Memory
Networking for Disaggregated Resources
Kernel-Level RDMA Virtualization (SOSP’17)
New Network Topology, Routing, Congestion-Ctrl
RDMA Network
Infiniband

Key Challenge of Resource Disaggregation: Cost of Crossing Network

	Bandwidth	Latency
Mem Bus	50-100 GB/s	~50ns
PCIe 3.0 (x16)	16 GB/s	~700ns
InfiniBand (EDR)	12.5 GB/s	500ns
InfiniBand (HDR)	25 GB/s	<500ns
GenZ	32-400 GB/s	<100ns

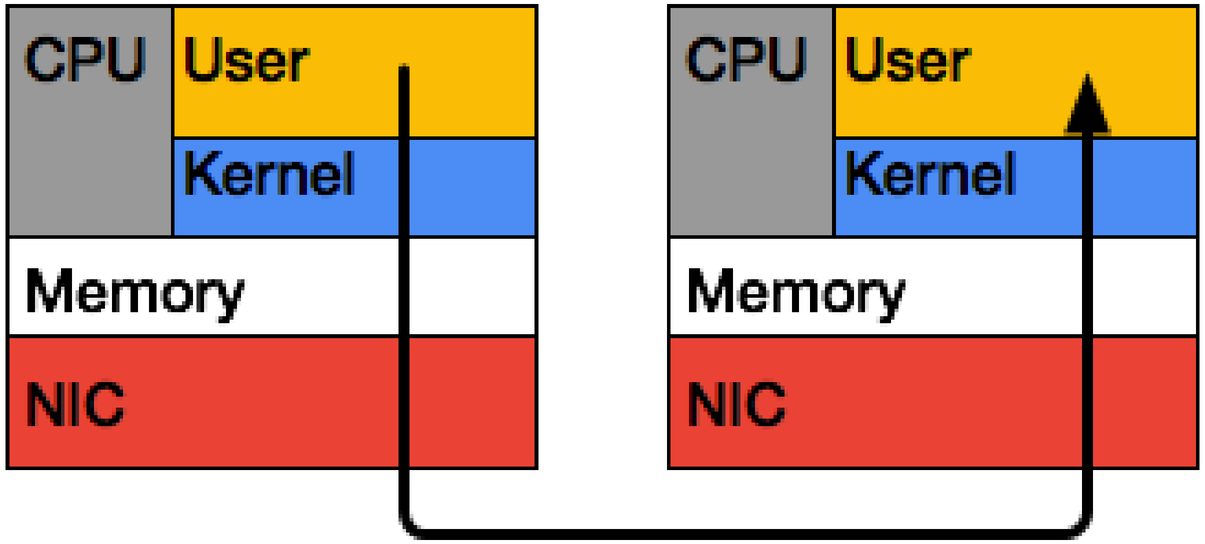
- Network **hardware** is much faster than before
- Current network still slower than local memory bus

Network Requirements for Resource Disaggregation

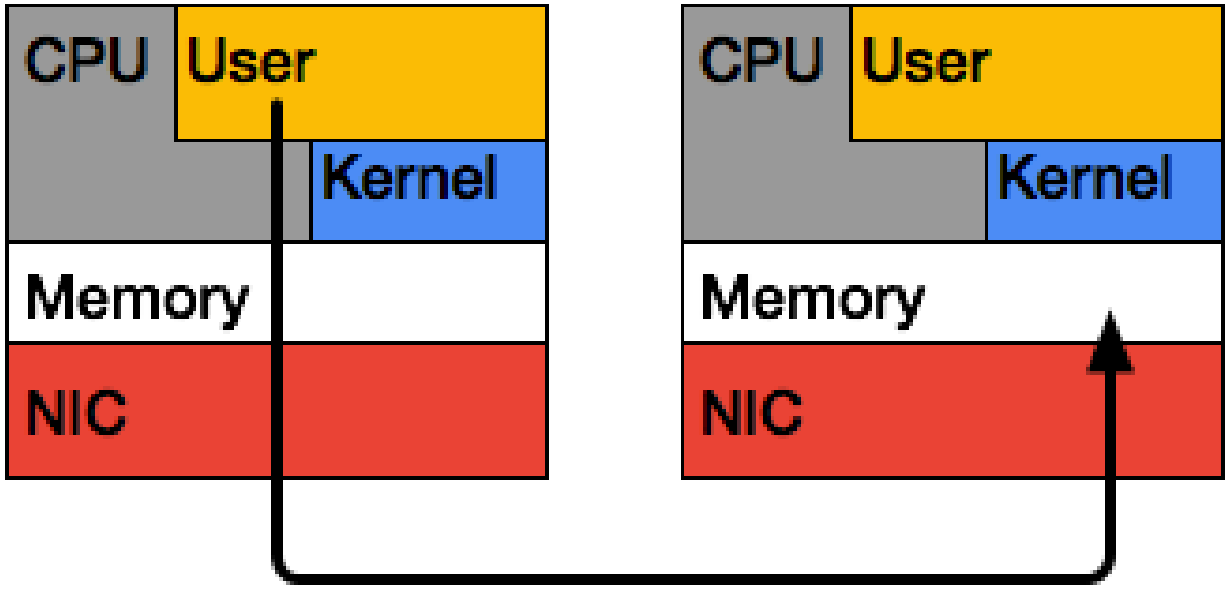
- Low latency
- High bandwidth
- Scale
- Reliable

RDMA

RDMA (Remote Direct Memory Access)



Socket over Ethernet



RDMA

- Directly read/write remote memory
- Bypass kernel
- Memory zero copy

Benefits:

- Low latency
- High throughput
- Low CPU utilization

Things have worked well in HPC

- Special hardware
- Few applications
- Cheaper developer

RDMA-Based Datacenter Applications

Pilaf
[ATC '13]

HERD-RPC
[ATC '16]

Cell
[ATC '16]

FaRM
[NSDI '14]

Wukong
[OSDI '16]

FaSST
[OSDI '16]

HERD
[SIGCOMM '14]

Hotpot
[SoCC '17]

NAM-DB
[VLDB '17]

RSI
[VLDB '16]

DrTM
[SOSP '15]

APUS
[SoCC '17]

Octopus
[ATC '17]

DrTM+R
[EuroSys '16]

FaRM+Xact
[SOSP '15]

Mojim
[ASPLOS '15]

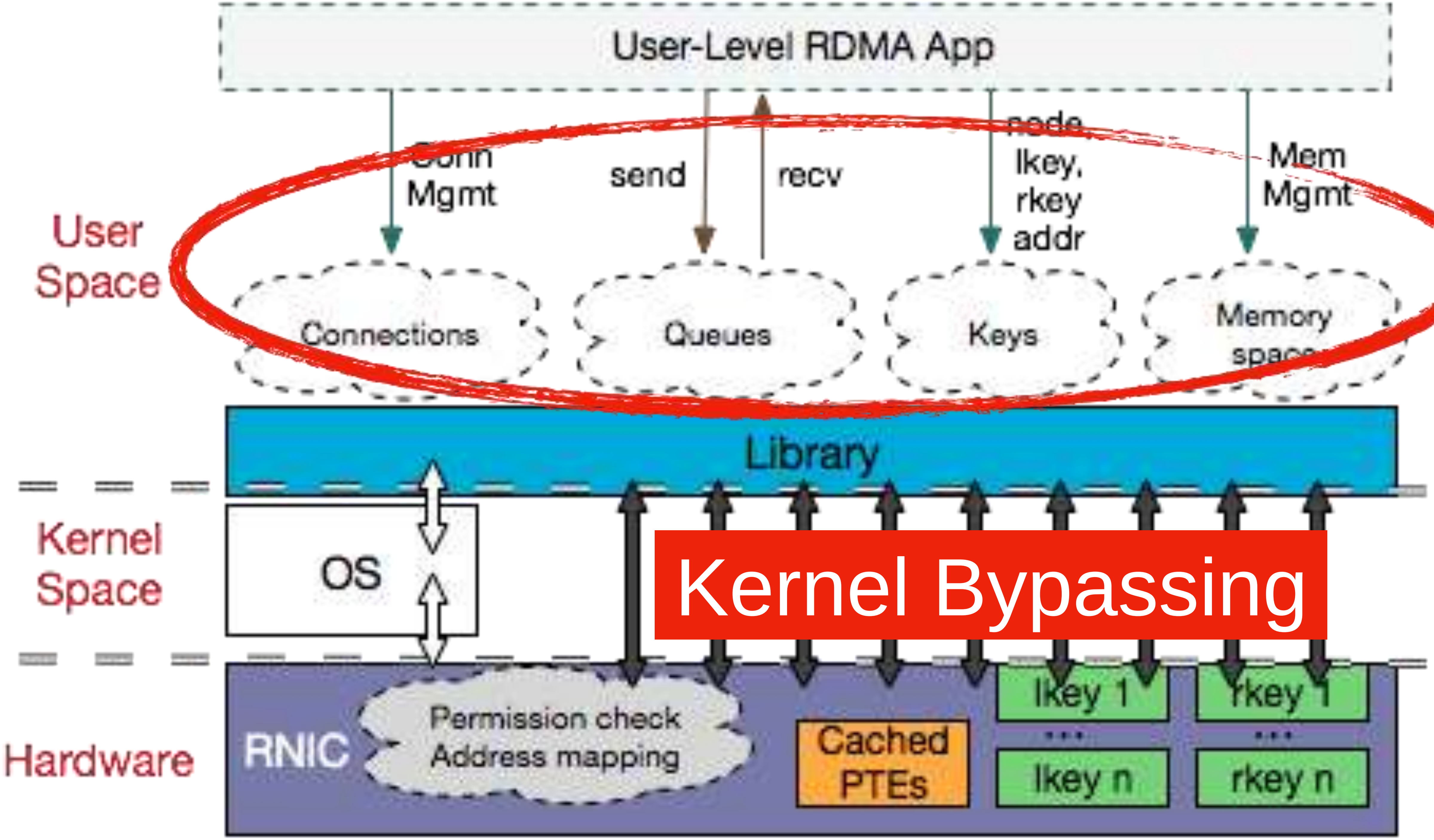
Things have worked well in HPC

- Special hardware
- Few applications
- Cheaper developer

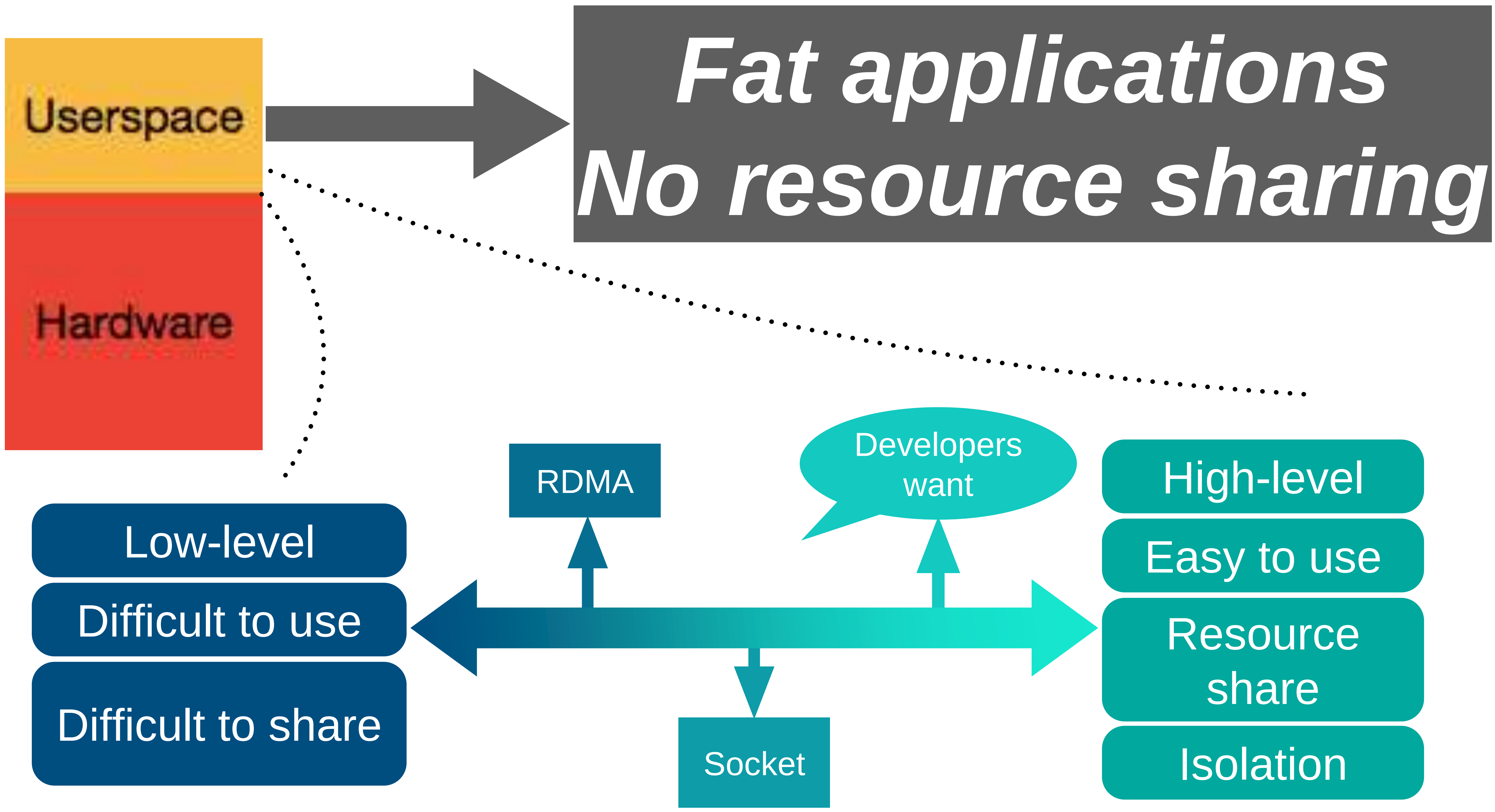
What about datacenters?

- Commodity, cheaper hardware
- Many (changing) applications
- Resource sharing and isolation

Native RDMA



User-Level RDMA App
Conn Mgmt
send
recv
node, lkey, rkey addr
Mem Mgmt
User Space
Connections
Queues
Keys
Memory space
Library
Kernel Space
OS
Kernel Bypassing
Hardware
RNIC
Permission check
Address mapping
Cached PTEs
lkey 1
rkey 1
lkey n
rkey n

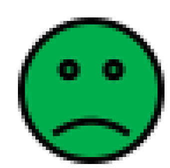
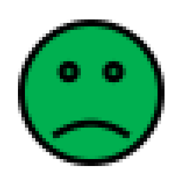


Abstraction Mismatch

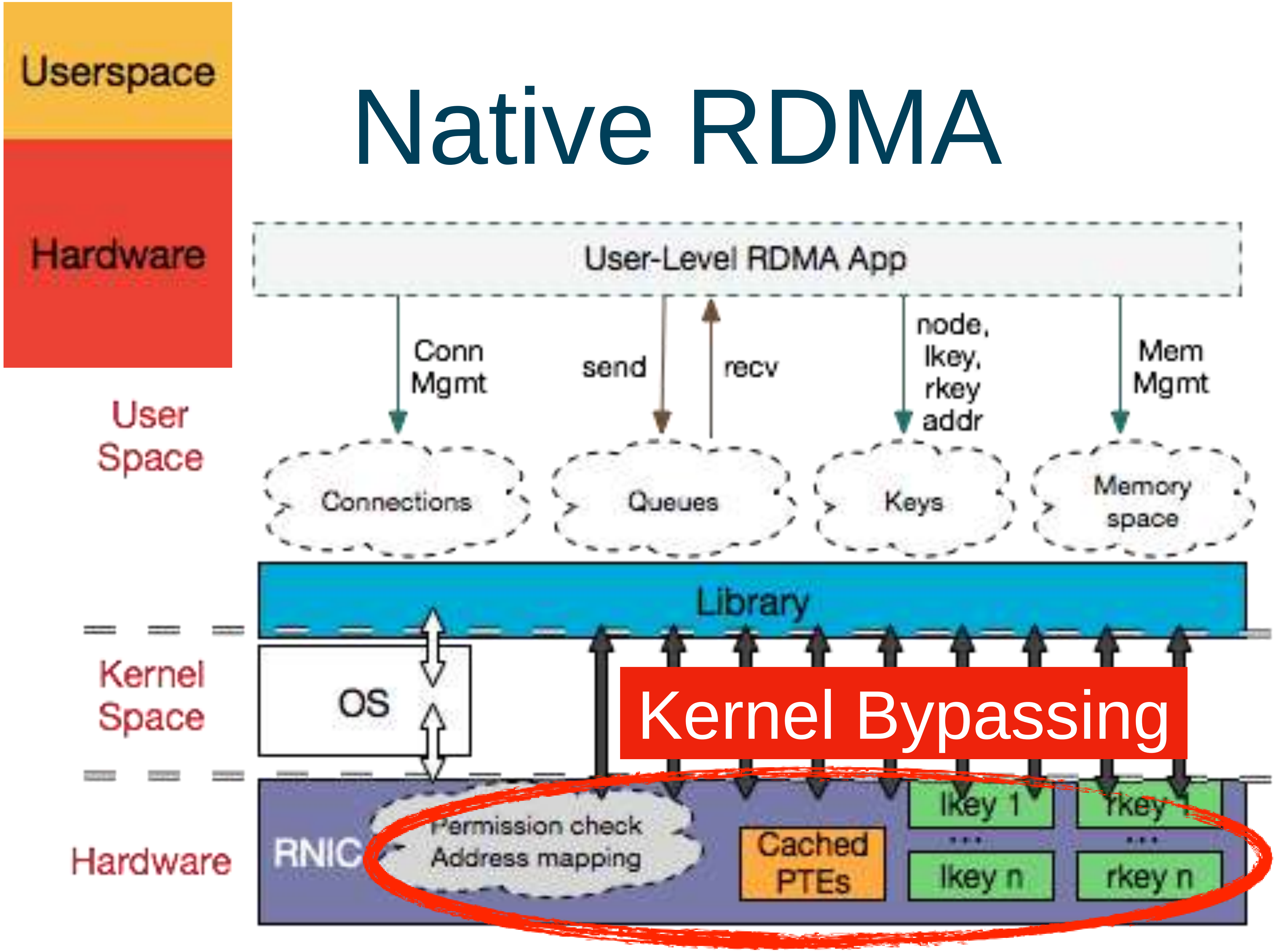
Things have worked well in HPC

- Special hardware
- Few applications 
- Cheaper developer 

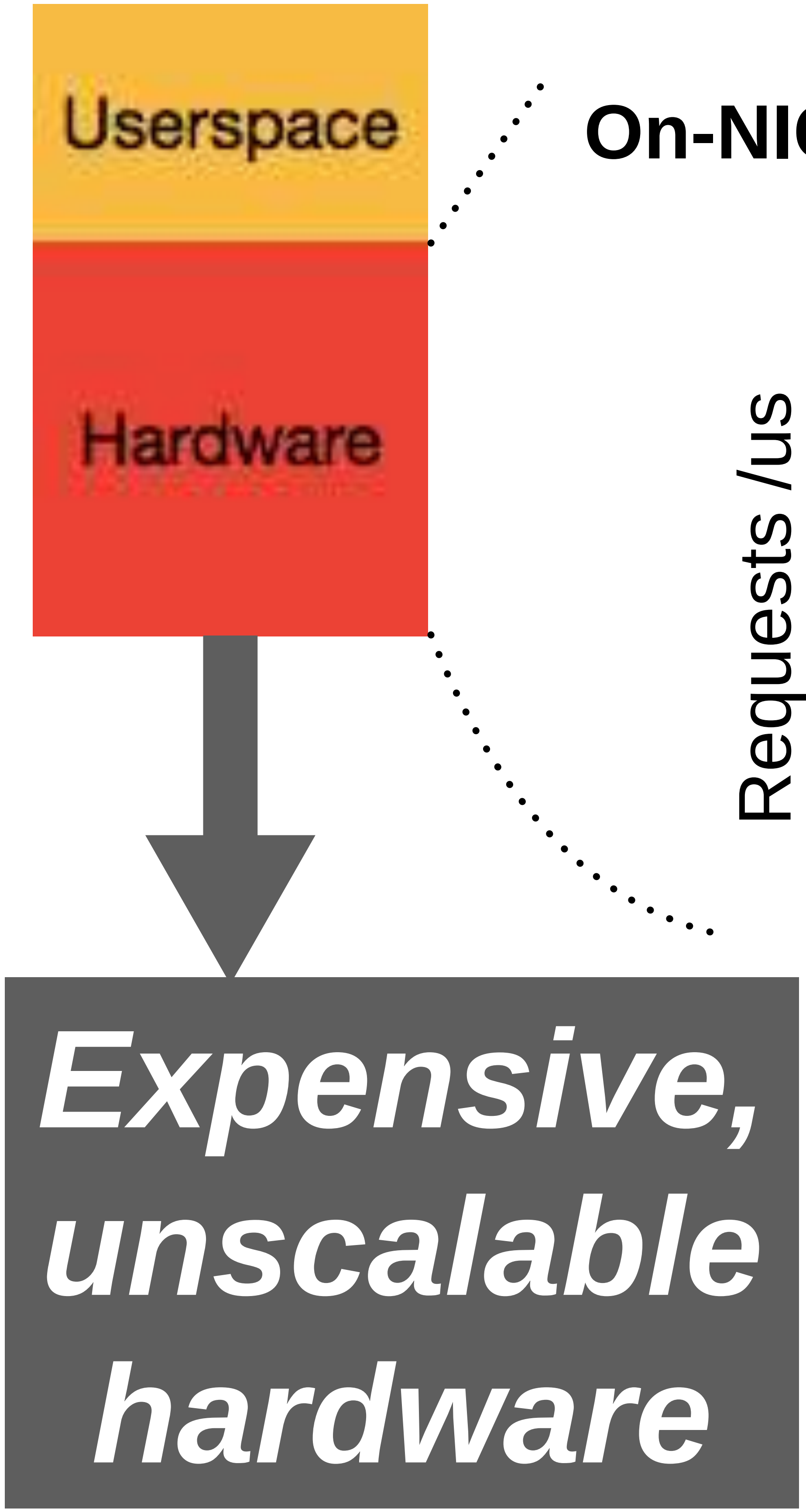
What about datacenters?

- Commodity, cheaper hardware
- Many (changing) applications 
- Resource sharing and isolation 

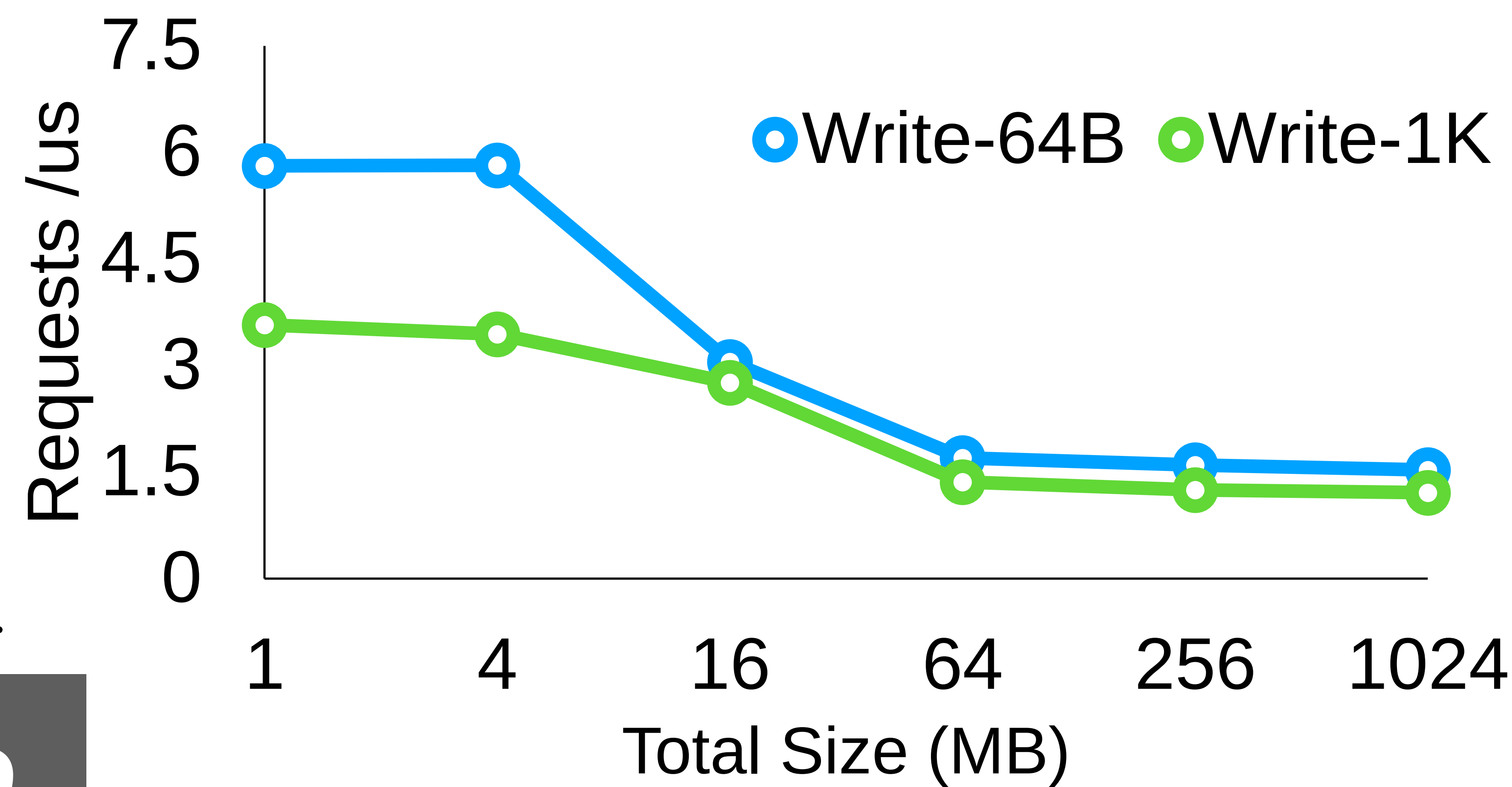
Native RDMA



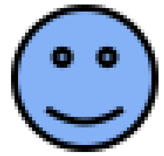
Userspace
Hardware
User-Level RDMA App
Conn Mgmt
send
recv
node, lkey, rkey addr
Mem Mgmt
User Space
Connections
Queues
Keys
Memory space
Library
Kernel Space
OS
Kernel Bypassing
Hardware
RNIC
Permission check
Address mapping
Cached PTEs
lkey 1
...
lkey n
rkey 1
...
rkey n



On-NIC SRAM stores and caches metadata



Things have worked well in HPC

- Special hardware 
- Few applications 
- Cheaper developer 

What about datacenters?

- Commodity, cheaper hardware 
- Many (changing) applications 
- Resource sharing and isolation 

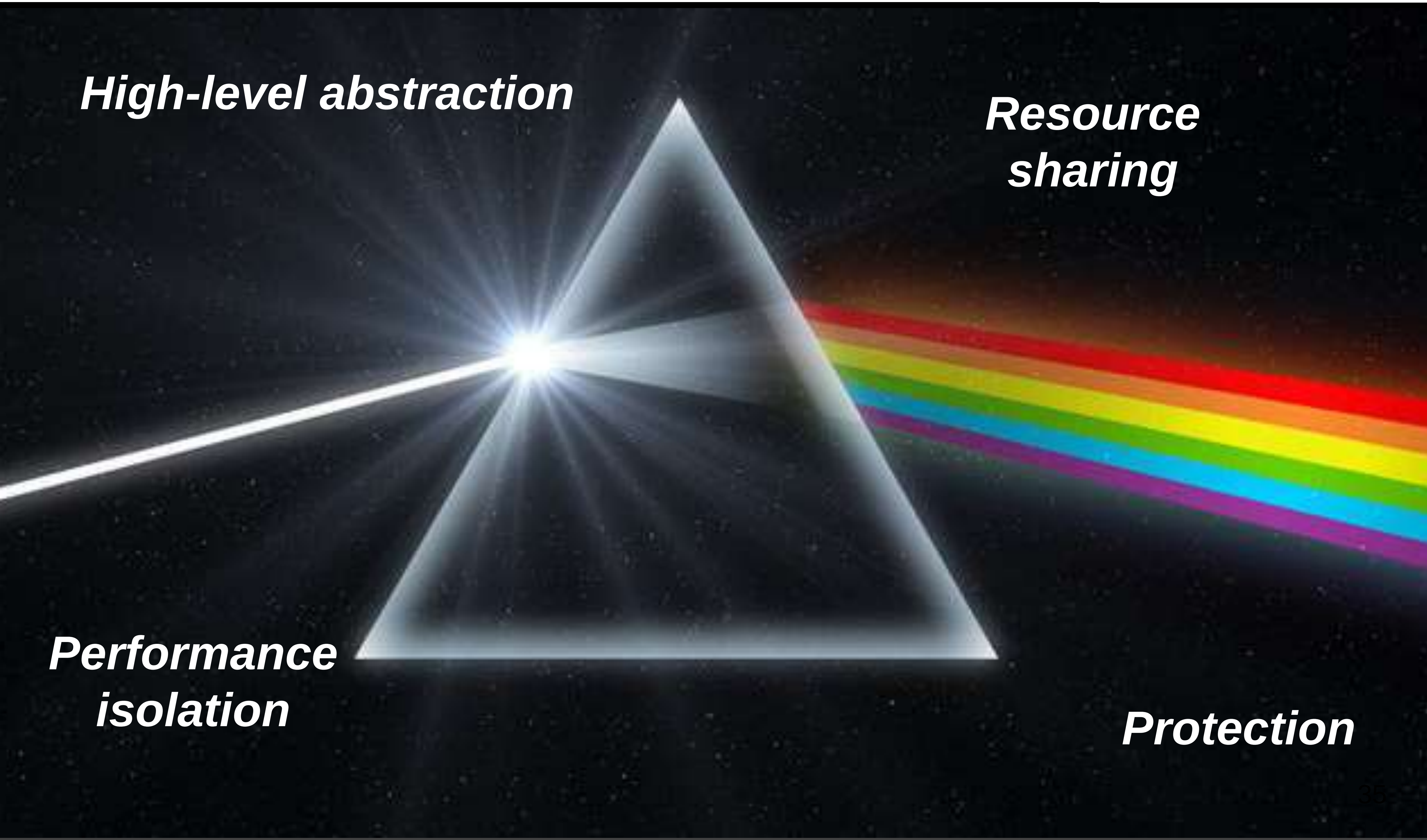
Fat applications No resource sharing

Expensive, unscalable hardware

Are we removing too much from kernel?

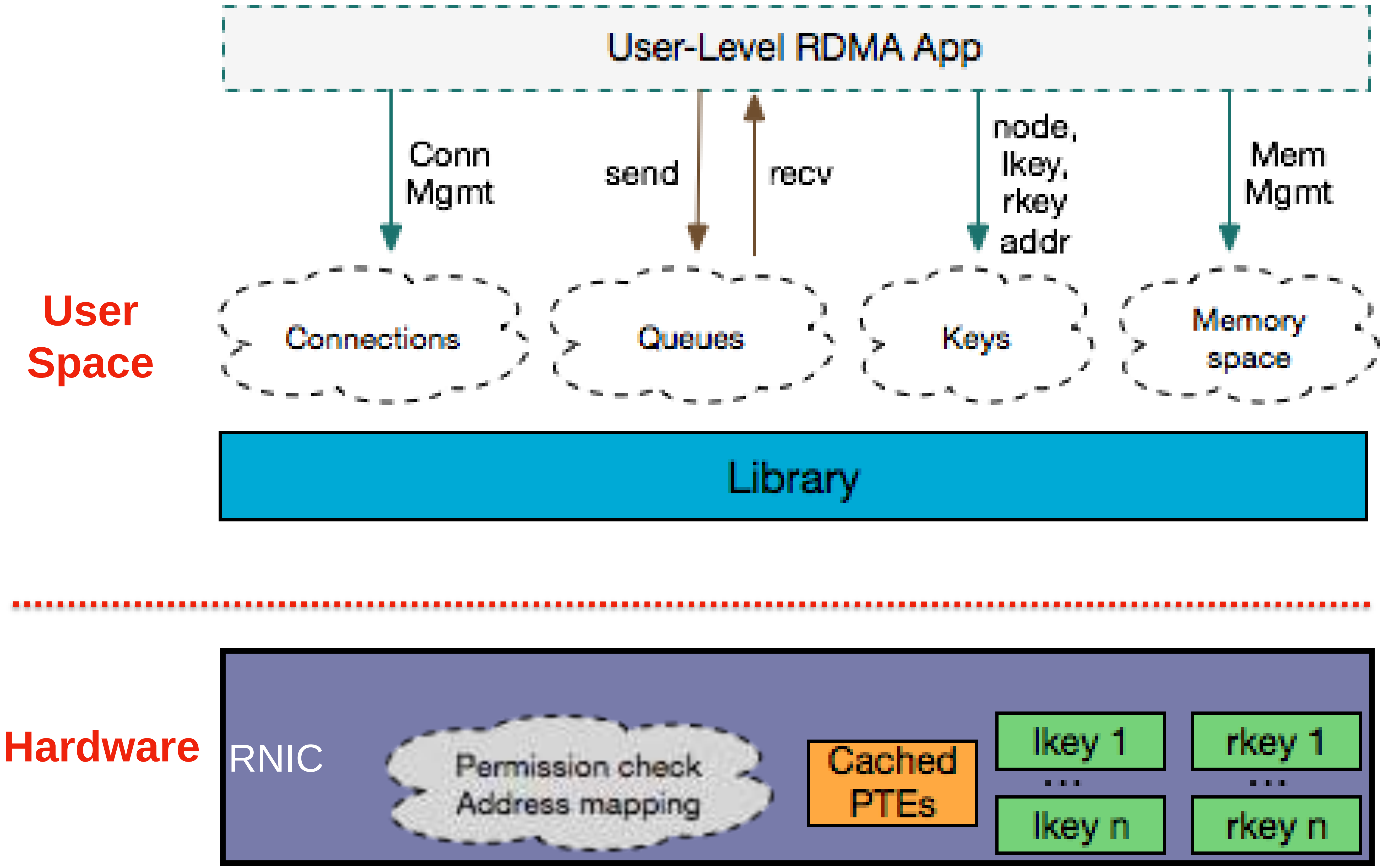
LITE - Local Indirection TiEr

Without Kernel

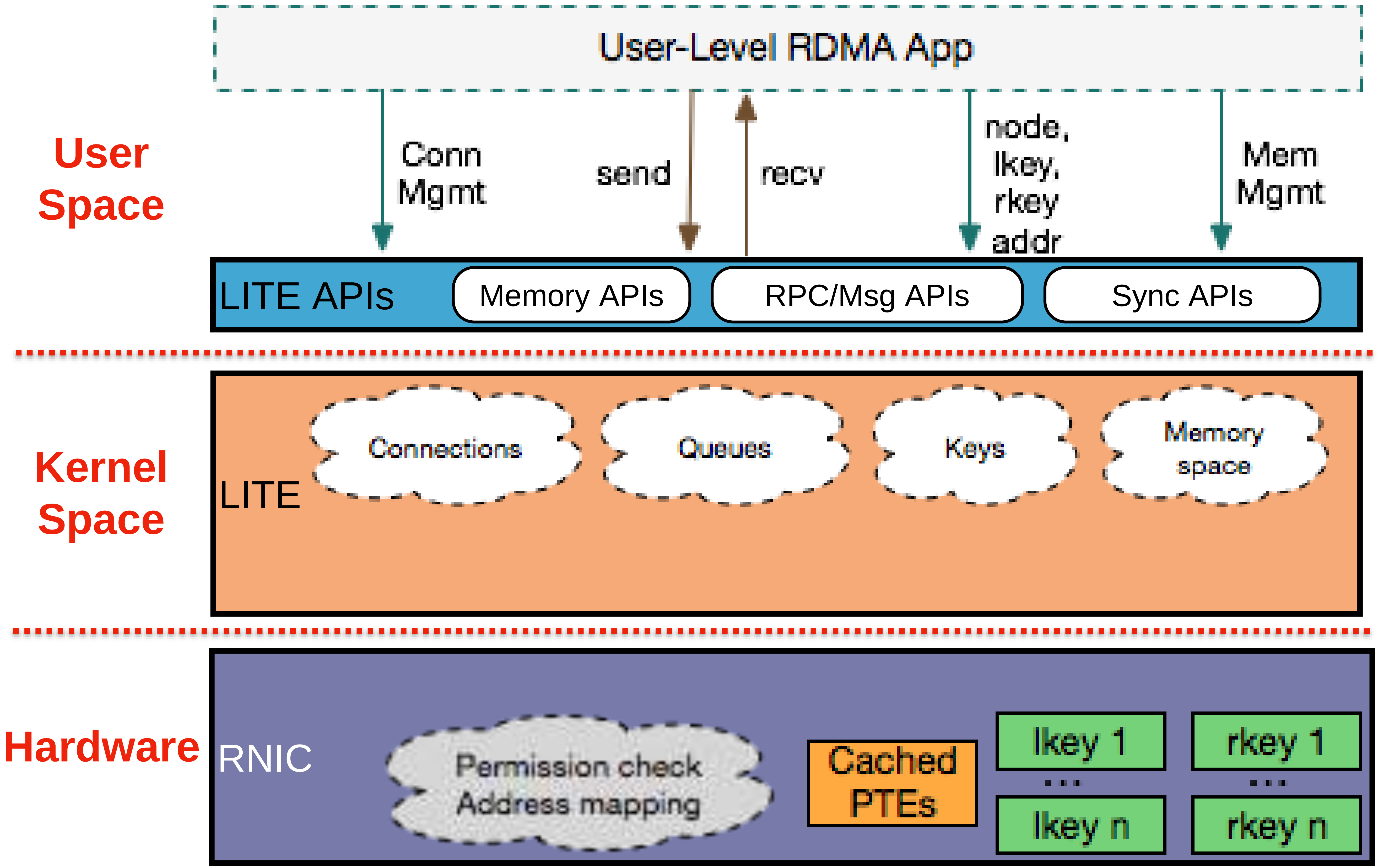


All problems in computer science can be solved by another level of indirection

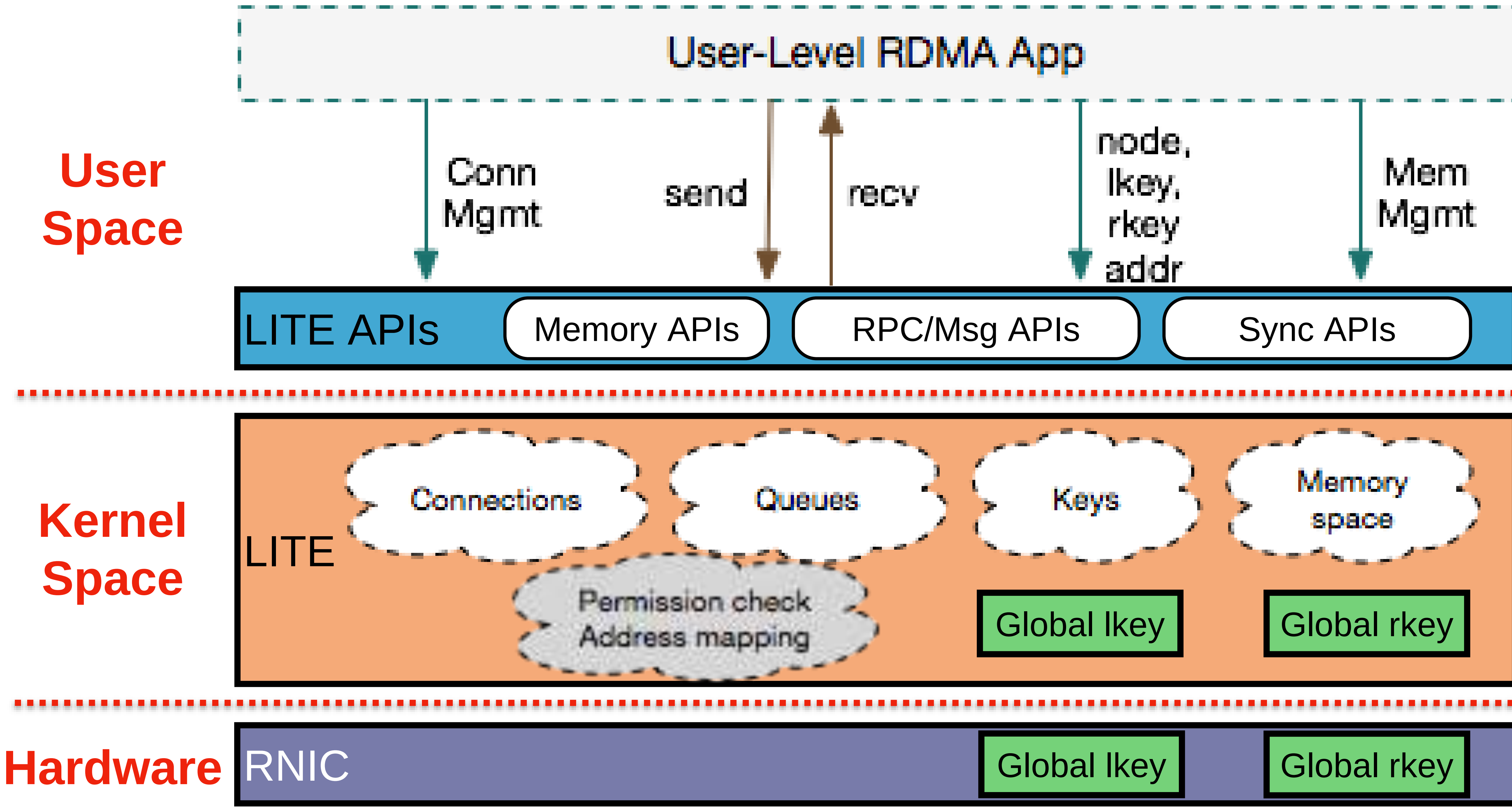
Butler Lampson


User-Level RDMA App
Conn Mgmt
send
recv
node, lkey, rkey addr
Mem Mgmt
User Space
Connections
Queues
Keys
Memory space
Library
Hardware
RNIC
Permission check
Address mapping
Cached PTEs
lkey 1
...
lkey n
rkey 1
...
rkey n

Simpler applications



Simpler applications



Cheaper hardware

Scalable performance

Implementing Remote *memset*

Native RDMA

```
struct pingpong_context *ctx;
ctx = calloc(1, sizeof *ctx);

ctx->size = size;
ctx->rx_depth = rx_depth;
ctx->buf = malloc(roundup(size, page_size));
memset(ctx->buf, 0x7b + is_server, size);
ctx->context = ibv_open_device(ib_dev);
ctx->channel = NULL;
ctx->pd = ibv_alloc_pd(ctx->context);
ctx->mr = ibv_reg_mr(ctx->pd, ctx->buf, size,IBV_ACCESS_LOCAL_WRITE|IBV_ACCESS_REMOTE_WRITE);
ctx->cq = ibv_create_cq(ctx->context, rx_depth + 1, NULL, ctx->channel, 0);

/* bunch of QP setup .... 50 LOCs */
ctx->qp = ibv_create_qp(ctx->pd, &attr);
ibv_modify_qp(ctx->qp, &attr,IBV_QP_STATE|IBV_QP_PKEY_INDEX|IBV_QP_PORT|IBV_QP_ACCESS_FLAGS);
/* build connections .... 100 LOCs */
/* exchange all required information, qpns, psns, and keys */

/* start doing write request */
struct ibv_sge sg;
struct ibv_send_wr wr;
struct ibv_send_wr *bad_wr;
memset(&wr, 0, sizeof(wr));
memset(&sg, 0, sizeof(sg));

/* setup all required metadata for a write request*/
sg.addr = (uintptr_t)buf_addr;
sg.length = buf_size;
sg.lkey = ctx->mr->lkey;

wr.wr_id = 0;
wr.sg_list = &sg;
wr.num_sge = 1;
wr.opcode  = IBV_WR_RDMA_READ;
wr.send_flags = IBV_SEND_SIGNALED;
wr.wr.rdma.remote_addr = remote_address
wr.wr.rdma.rkey = remote_key;

/* send out the request */
ibv_post_send(qp, &wr, &bad_wr);
struct ibv_wc wc[2];
ibv_poll_cq(ctx->cq, 2, wc); /* busy poll until getting completion */

return ctx;
```

LITE

```
LITE_join(IP);
uin64_t lh = LITE_malloc(node, size);
LITE_memset(lh, 0, offset, size);
```

All problems in computer science can be solved by another level of indirection

Butler Lampson

David Wheeler

except for the problem of too many layers of indirection

– David Wheeler

Main Challenge:
How to preserve the performance benefit of RDMA?

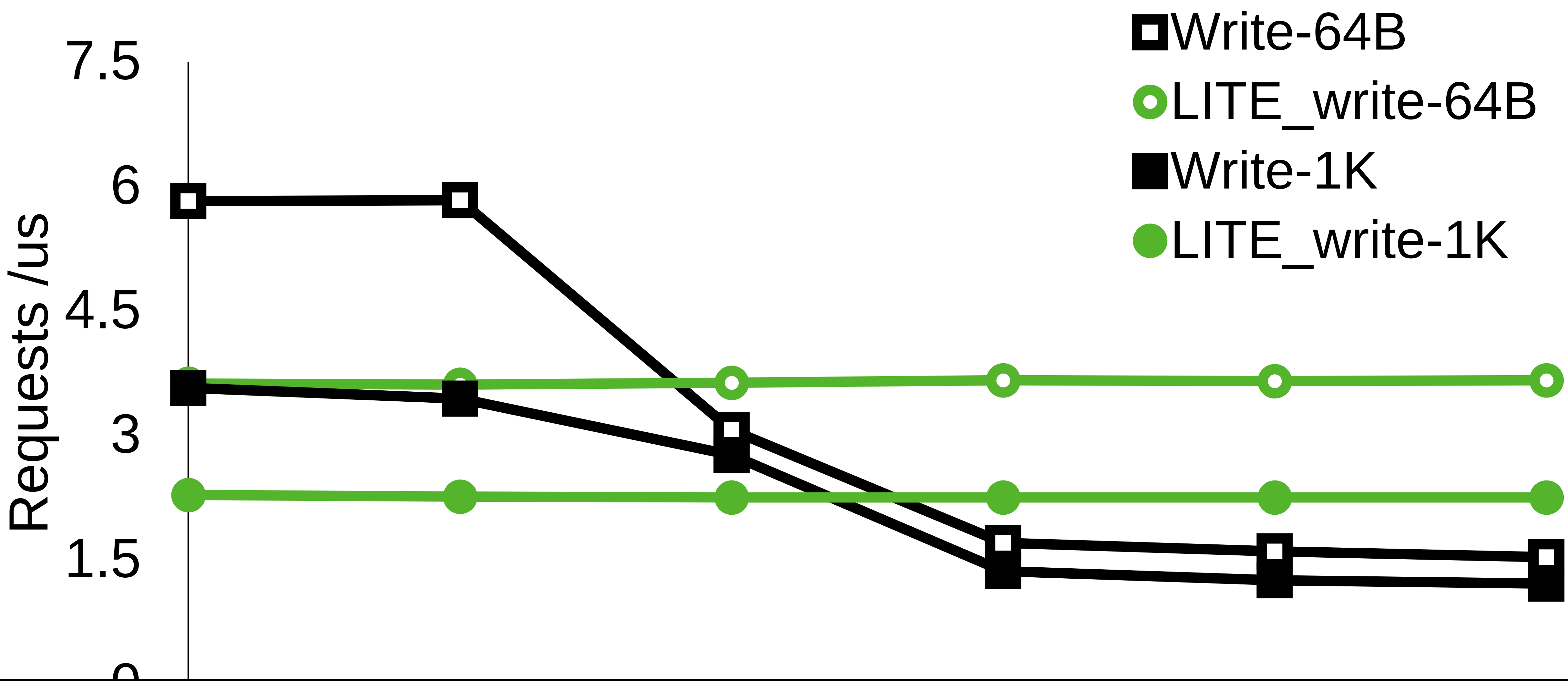
LITE Design Principles

1. Indirection only at local node
2. Avoid hardware-level indirection
3. Hide kernel-space crossing cost

~~except for the problem of too many layers of indirection – David Wheeler~~

Great Performance and Scalability

LITE RDMA:Size of MR Scalability



LITE scales much better than native RDMA wrt MR size and numbers

LITE Application Effort

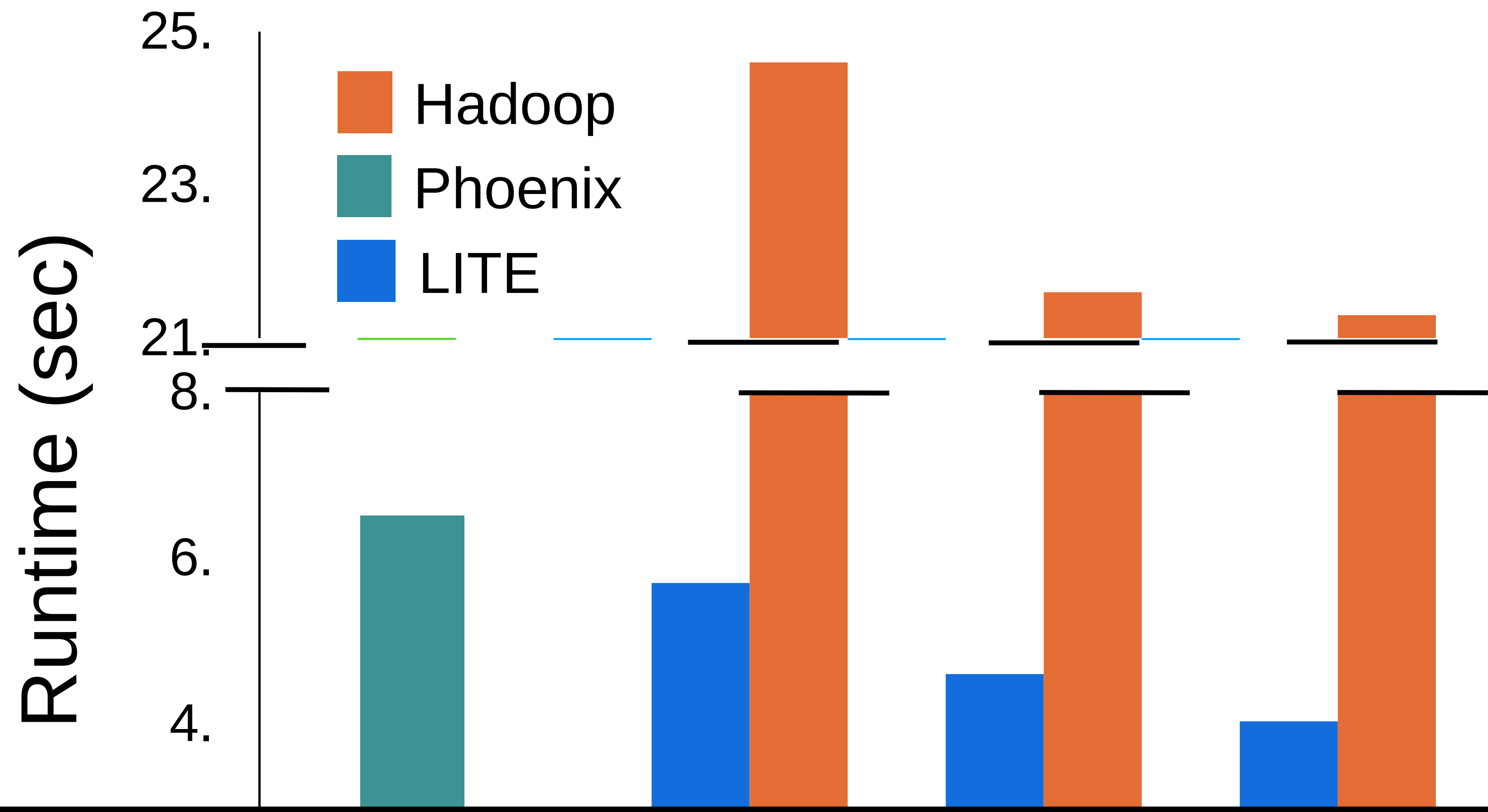
Application	LOC	LOC using LITE	Student Days
LITE-Log	330	36	1
LITE-MapReduce	600*	49	4
LITE-Graph	1400	20	7
LITE-Kernel-DSM	3000	45	26

- Simple to use
- Needs no expert knowledge
- Flexible, powerful abstraction
- Easy to achieve optimized performance

* LITE-MapReduce ports from the 3000-LOC Phoenix with 600 lines of change or addition

MapReduce Results

- LITE-MapReduce adapted from Phoenix [1]

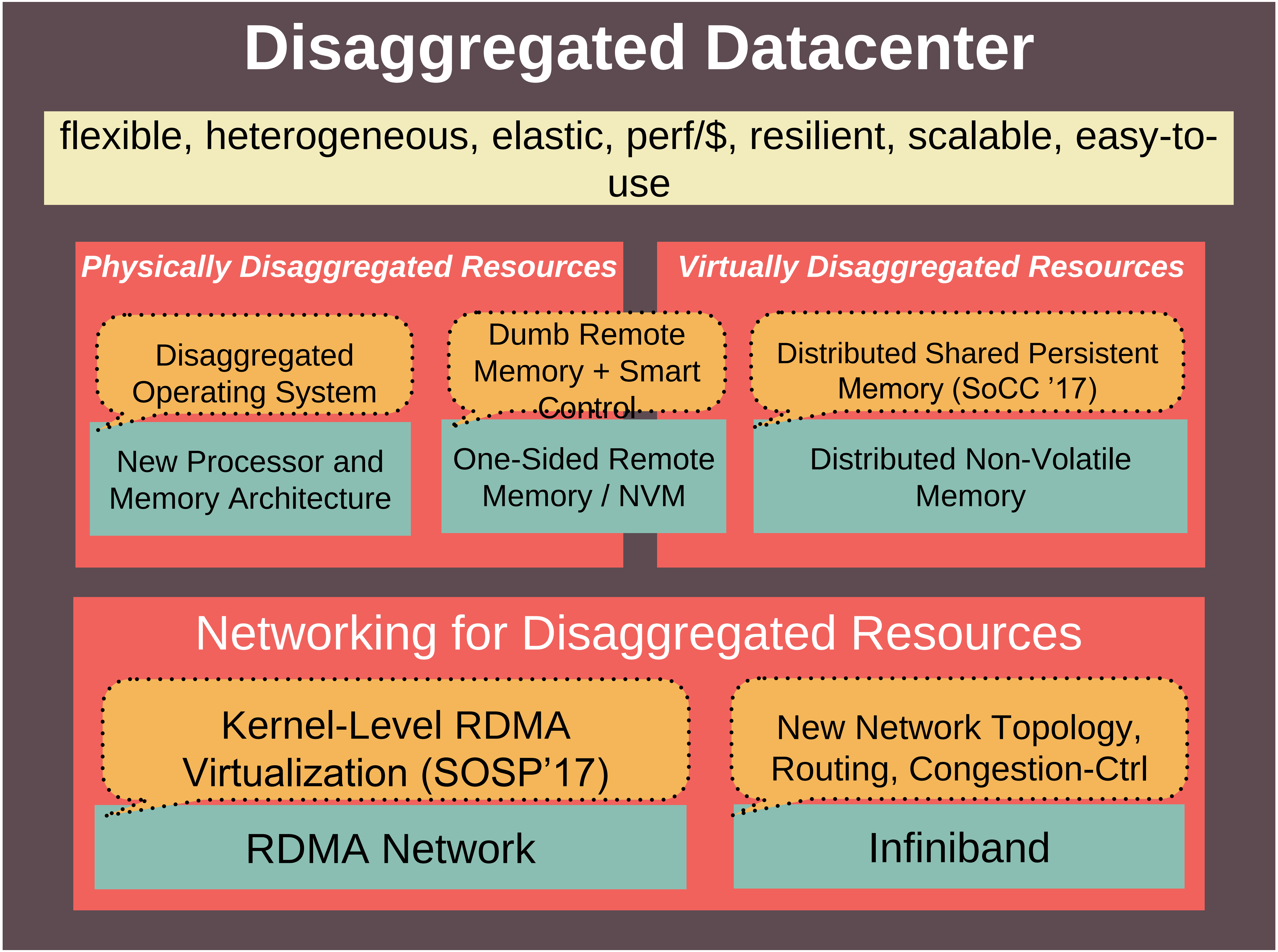


LITE-MapReduce outperforms Hadoop by 4.3x to 5.3x

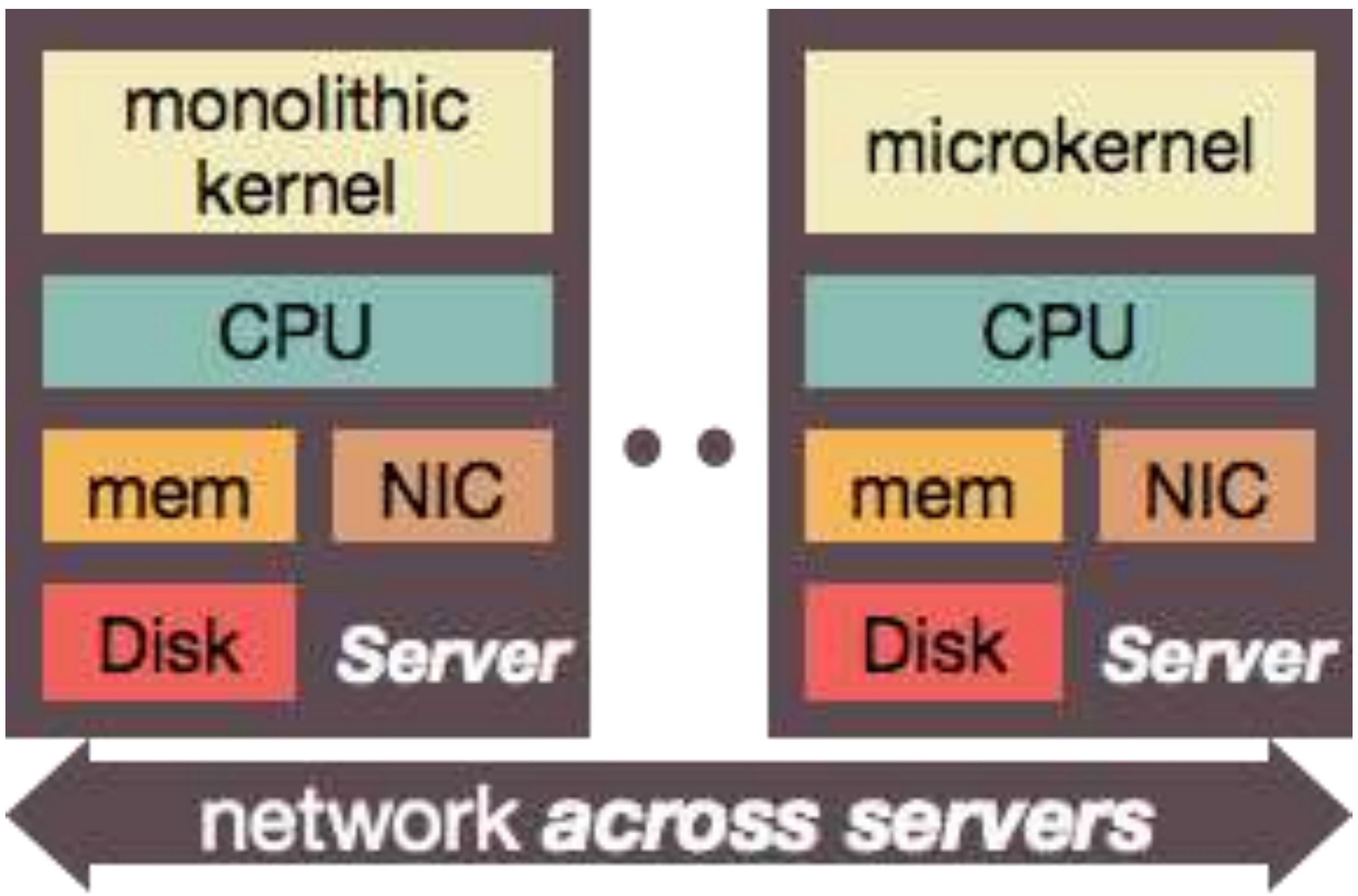
[1]: "Ranger etal., Evaluating MapReduce for Multi-core and Multiprocessor Systems. (HPCA 07)"

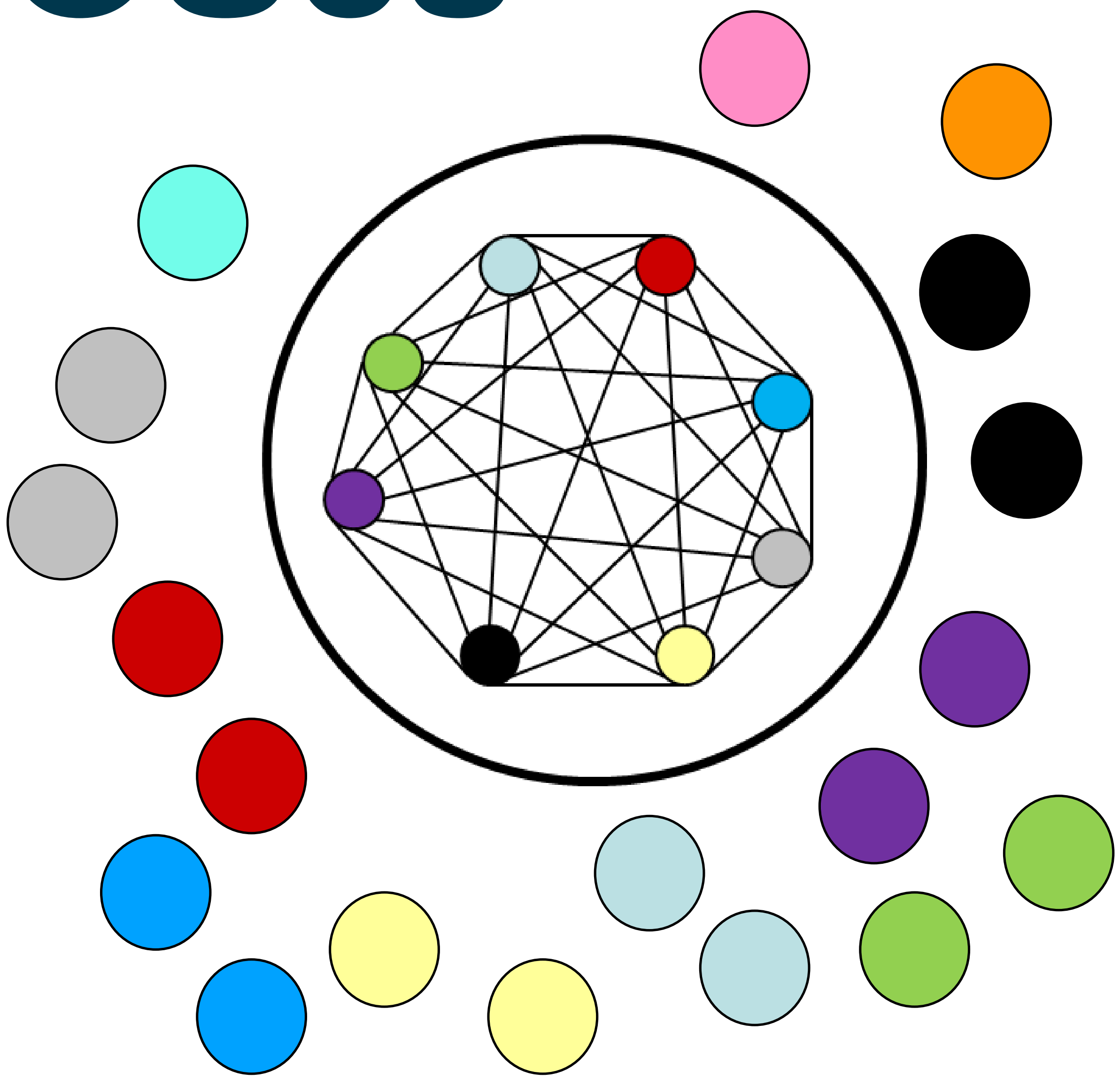
LITE Summary

- Virtualizes RDMA into flexible, easy-to-use abstraction
- Preserves RDMA’s performance benefits
- ***Indirection*** not always degrade performance!
- Division across user space, kernel, and hardware


Disaggregated Datacenter
flexible, heterogeneous, elastic, perf/$, resilient, scalable, easy-to-use
Physically Disaggregated Resources
Virtually Disaggregated Resources
Disaggregated Operating System
Dumb Remote Memory + Smart Control
Distributed Shared Persistent Memory (SoCC ’17)
New Processor and Memory Architecture
One-Sided Remote Memory / NVM
Distributed Non-Volatile Memory
Networking for Disaggregated Resources
Kernel-Level RDMA Virtualization (SOSP’17)
New Network Topology, Routing, Congestion-Ctrl
RDMA Network
Infiniband

Traditional OSes

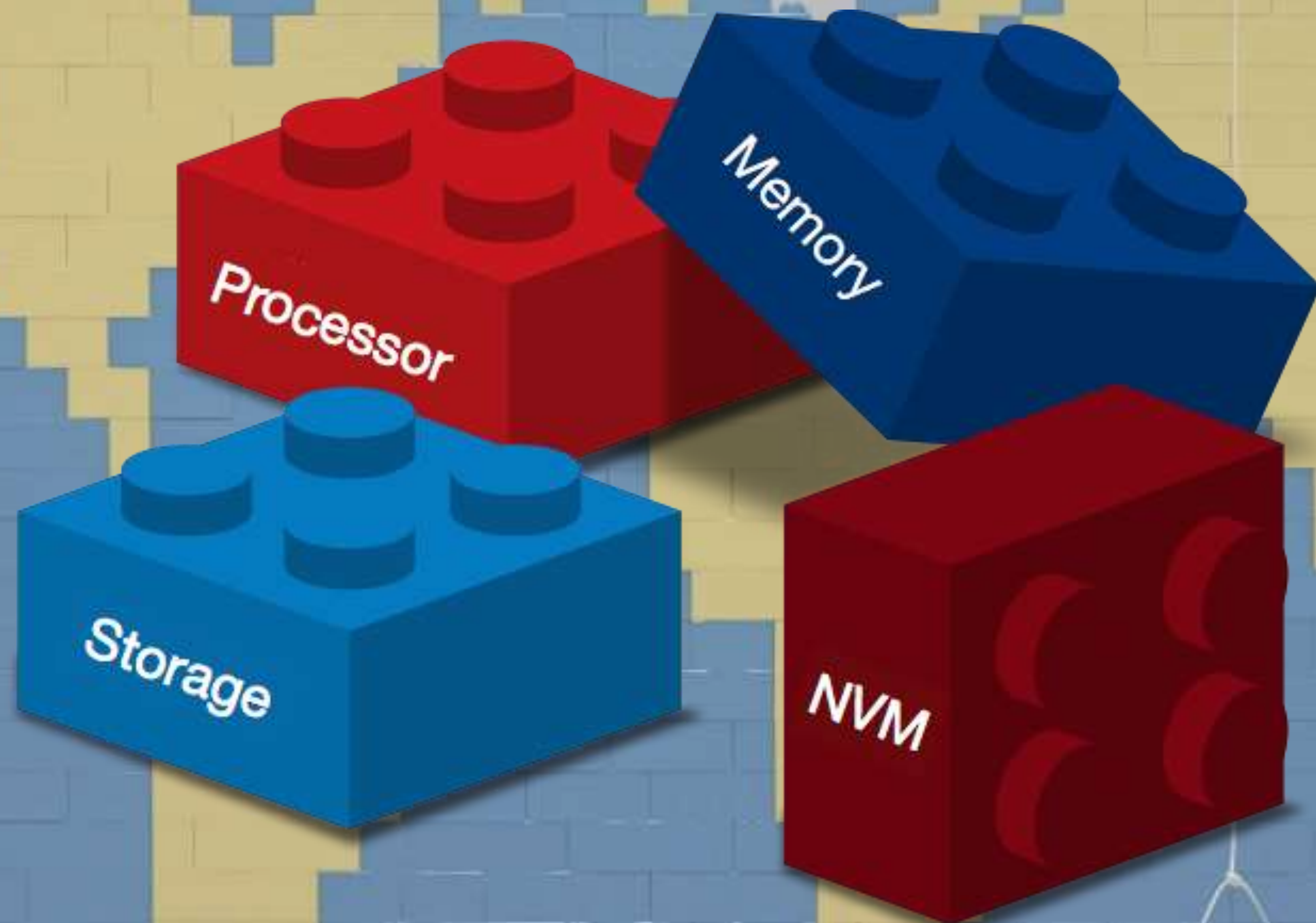




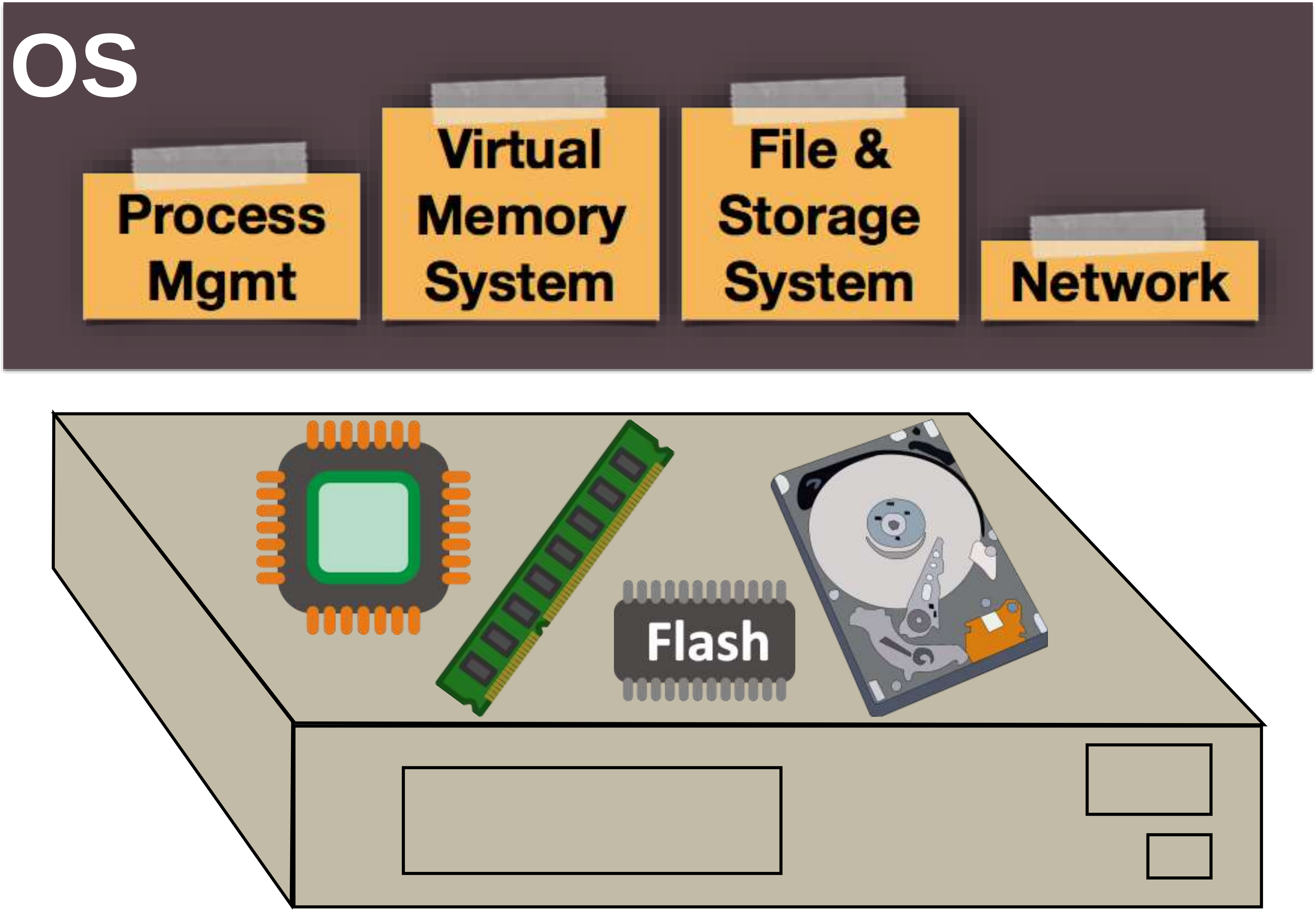
- Manages single node and all hardware resources in it
- Bad for hardware heterogeneity and hotplug
- Does not handle component failure

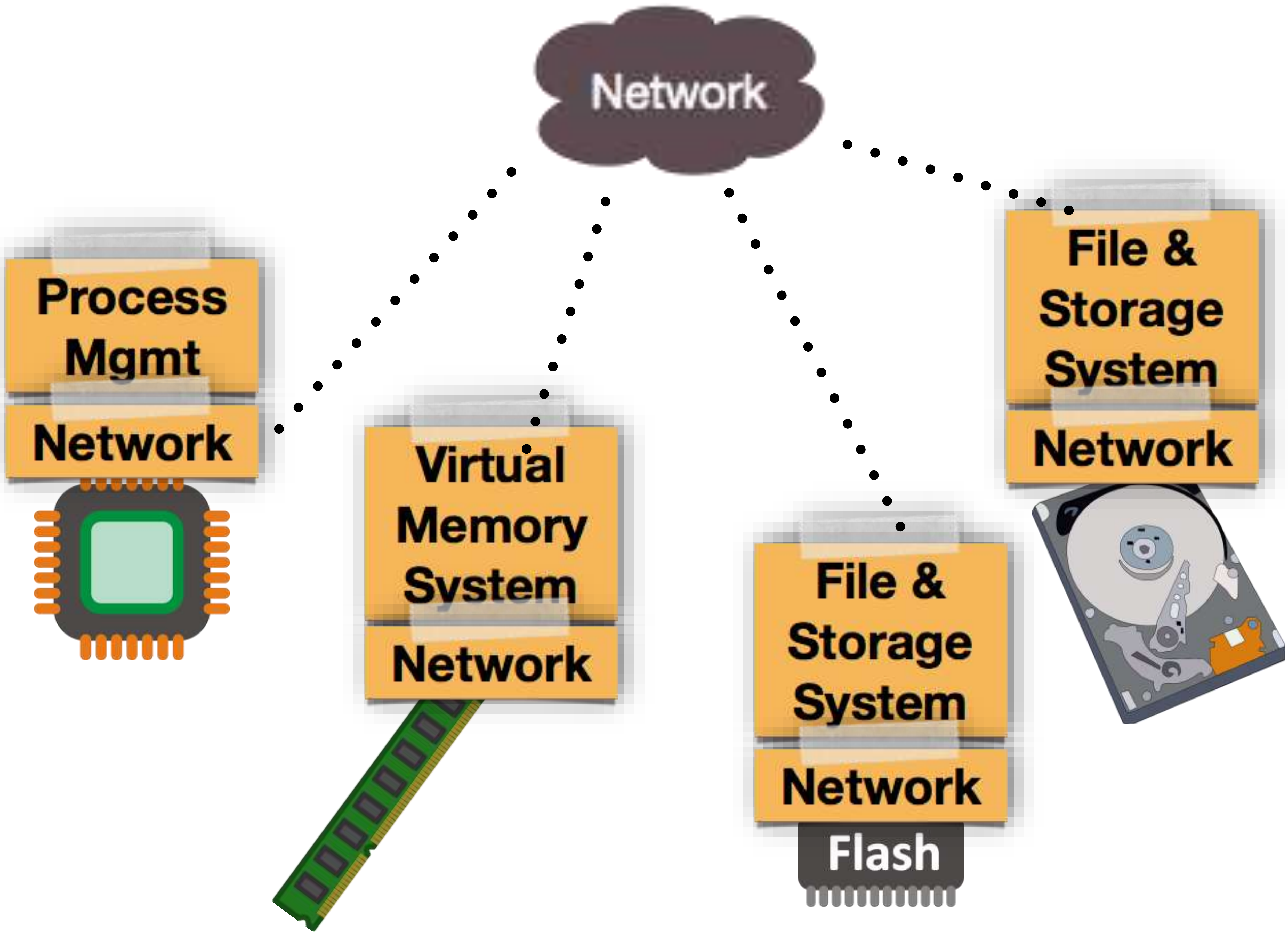

Operating
System
hardware
systems
machine
game
large
request
responsible
interact
gooey
like
computer
sharing
acts
typing
makes
Almost
activities
command
type
operation
portable
media
offer
generally
Unix
applications
appliances
OS
implemented
programming
embedded
computing
use
programs
calls
data
host
details
kind
using
however
supercomputers
management
pass
commands
well
pronounced
handheld
included
purposes
run
CLI
disk
also
device
machines
consoles
graphical
considered
easier
robots
contention
parameters
number
players
relieves
washing
including
interfaces
one
may
manage
invoking
APIs
runs
multi-user
GUI

Lego: the *First* Disaggregated OS



When hardware is disaggregated, the OS should be also!


OS
Process Mgmt
Virtual Memory System
File & Storage System
Network
Flash

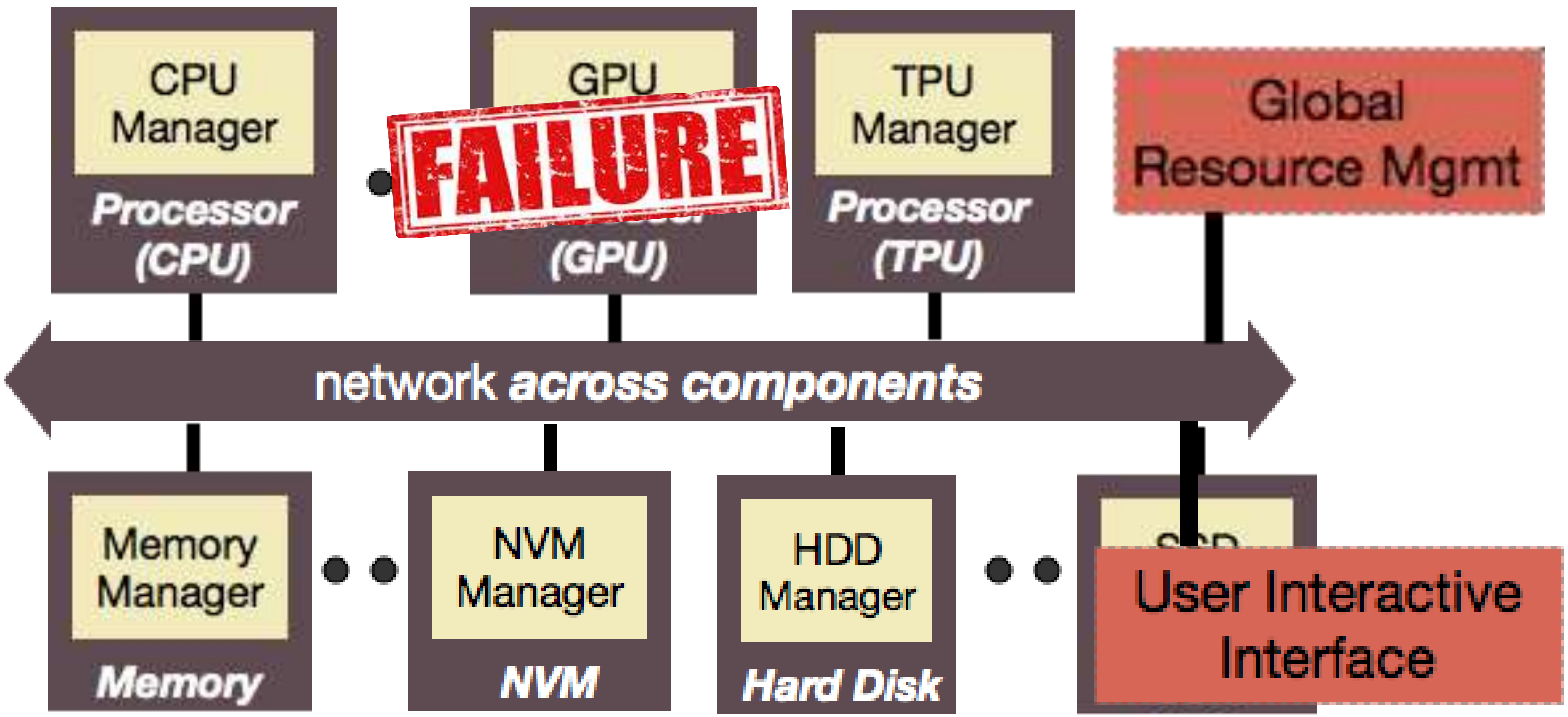

Network
Process Mgmt
Network
Virtual Memory System
Network
File & Storage System
Network
Flash
File & Storage System
Network

Micro-OS Service

- Manages hardware resource of a component
- Virtualizes the hardware
- Communicates with other micro-OS services
- Runs in hardware controller (kernel space)
- Only processors have user space

Lego Architecture

- ✓ Heterogeneity
- ✓ Flexibility
- ✓ Elasticity
- ✓ Resource util
- ✓ Failure isolation



Many Challenges

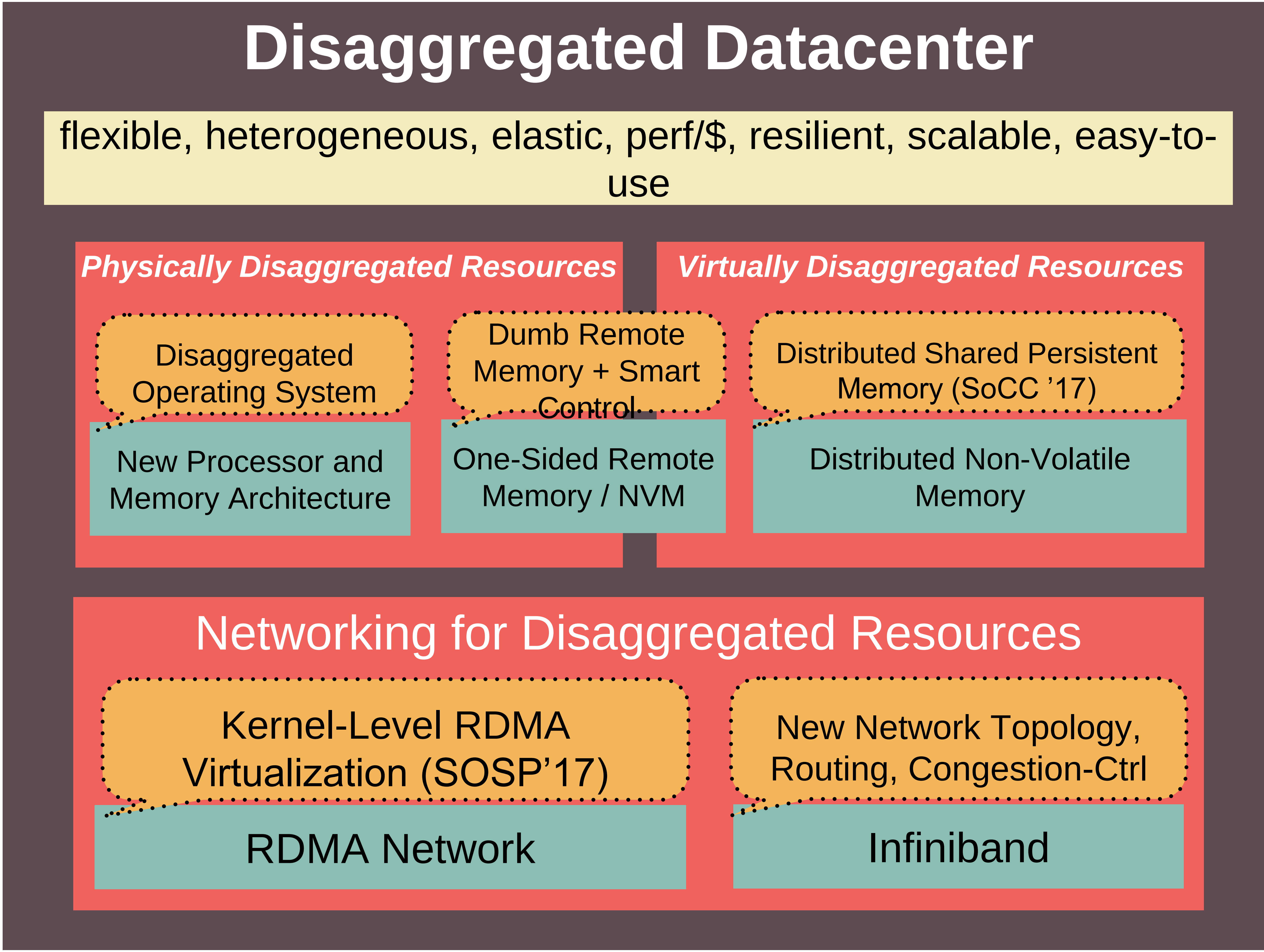
- Handling component failure
- Manage distributed, heterogeneous resources
- Fitting micro-OS services in hardware controller
- Implementing Lego on current servers

Lego Implementation

- Built from scratch, >200K LOC and growing
- Runs all Linux ABIs and unmodified binaries
- Three micro-OS services: processor, memory, storage
- Global resource manager
- Emulates disaggregated hardware with regular servers

Lego Summary

- Resource disaggregation calls for new system
- ***Lego***: new OS designed and built from scratch for datacenter resource disaggregation
- Split OS into distributed micro-OS services, running at device
- Many challenges and many potentials


Disaggregated Datacenter
flexible, heterogeneous, elastic, perf/$, resilient, scalable, easy-to-use
Physically Disaggregated Resources
Virtually Disaggregated Resources
Disaggregated Operating System
New Processor and Memory Architecture
Dumb Remote Memory + Smart Control
One-Sided Remote Memory / NVM
Distributed Shared Persistent Memory (SoCC ’17)
Distributed Non-Volatile Memory
Networking for Disaggregated Resources
Kernel-Level RDMA Virtualization (SOSP’17)
RDMA Network
New Network Topology, Routing, Congestion-Ctrl
Infiniband

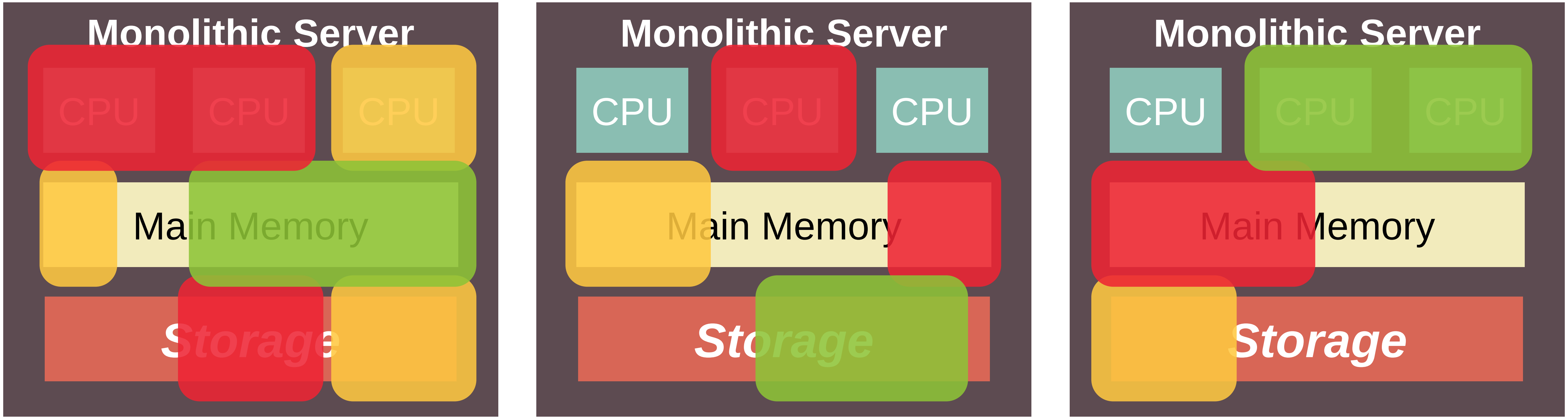
Physical Resource Disaggregation

- Great support of heterogeneity
- Very flexible in resource management
- But needs hardware, network, and OS changes

Is there any less disruptive way to achieve better resource utilization and elasticity?

Virtually Disaggregated Datacenter

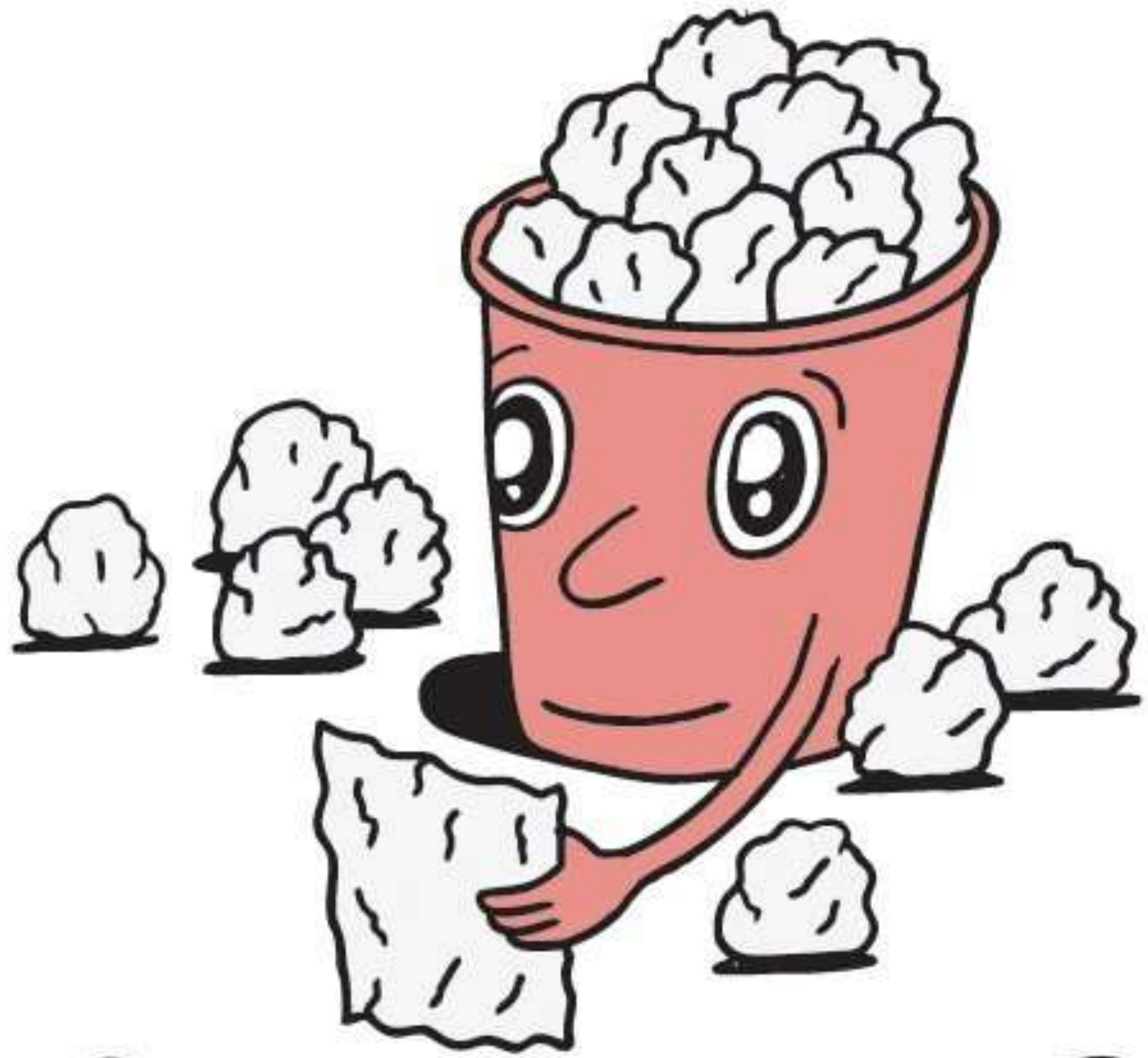
- Use resources on remote (distributed) machines



Using Remote/Distributed Resources

- Was a popular idea in 90s
  - Remote memory/paging/swap
  - Network block device
  - Distributed shared memory (DSM)

- No production-scale adoption
  - Cost of network communication
  - Coherence traffic



REVISIT YOUR
OLD IDEAS
WITH NEW EYES.

Remote/Distributed Memory in Modern Times

- New and heterogeneous applications
  - Large parallelism
  - New computation and memory requirements
  - New programming models
- Network is 10x-100x faster
  - InfiniBand: 200Gbps, <500ns
  - GenZ: 32-400GB/s, <100ns
- New types of memory
  - NVM, HBM

Recent New Attempts

- Distributed Shared Memory
  - Grappa
- Network swapping
  - InfiniSwap
- Non-coherent distributed memory
  - VMware

Our View

- Design new remote/distributed memory systems that leverage new hardware and network properties
- Design for modern datacenter applications
- Tradeoff of performance and programability
- First project: distributed Non-Volatile Memory (Persistent Memory)

DSM

Distributed Shared Persistent Memory (DSPM)

a significant step towards using PM in datacenters

DSPM

- Native memory load/store interface
  – Local or remote (transparent)
  – Pointers and in-memory data structures

- Supports memory read/write sharing

DSM

Distributed Shared Persistent Memory (DSPM)

a significant step towards using PM in datacenters

DSPM

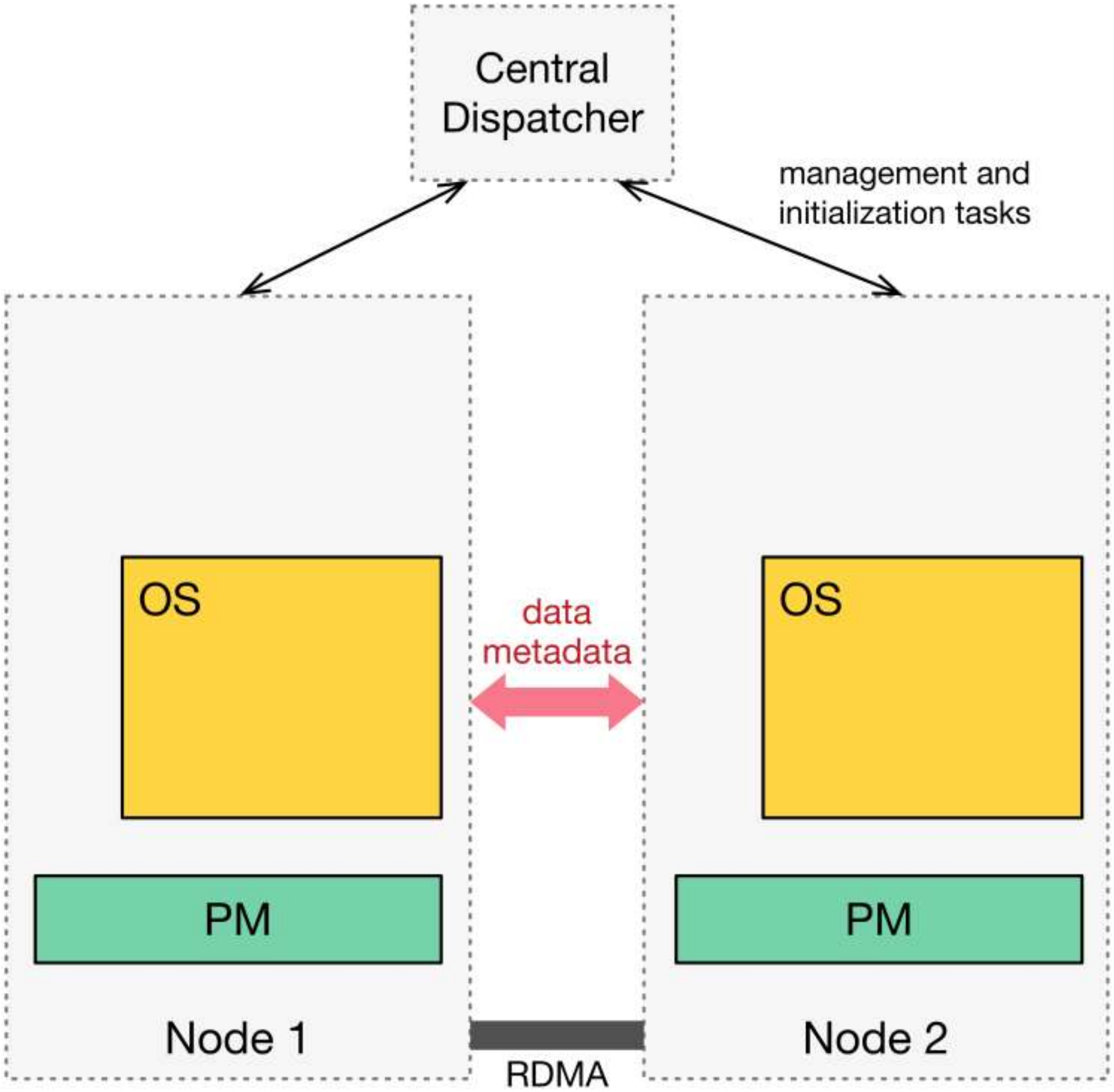
DSPM: One Layer Approach

Benefits of both memory and storage
No redundant layers
No data marshaling/unmarshalling

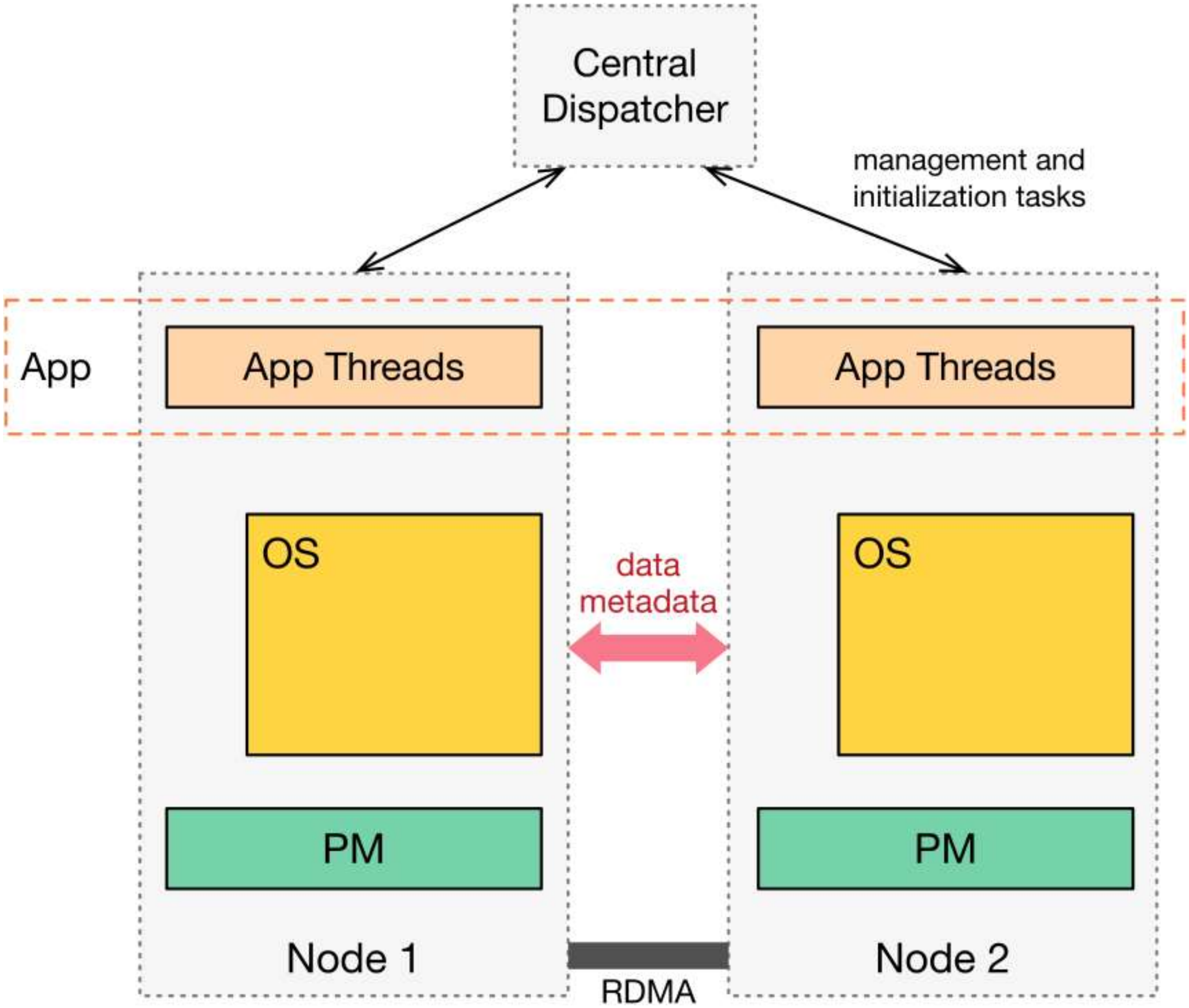
Hotpot: A Kernel-Level RDMA-Based DSPM System

- Easy to use
- Native memory interface
- Fast, scalable
- Flexible consistency levels
- Data durability & reliability

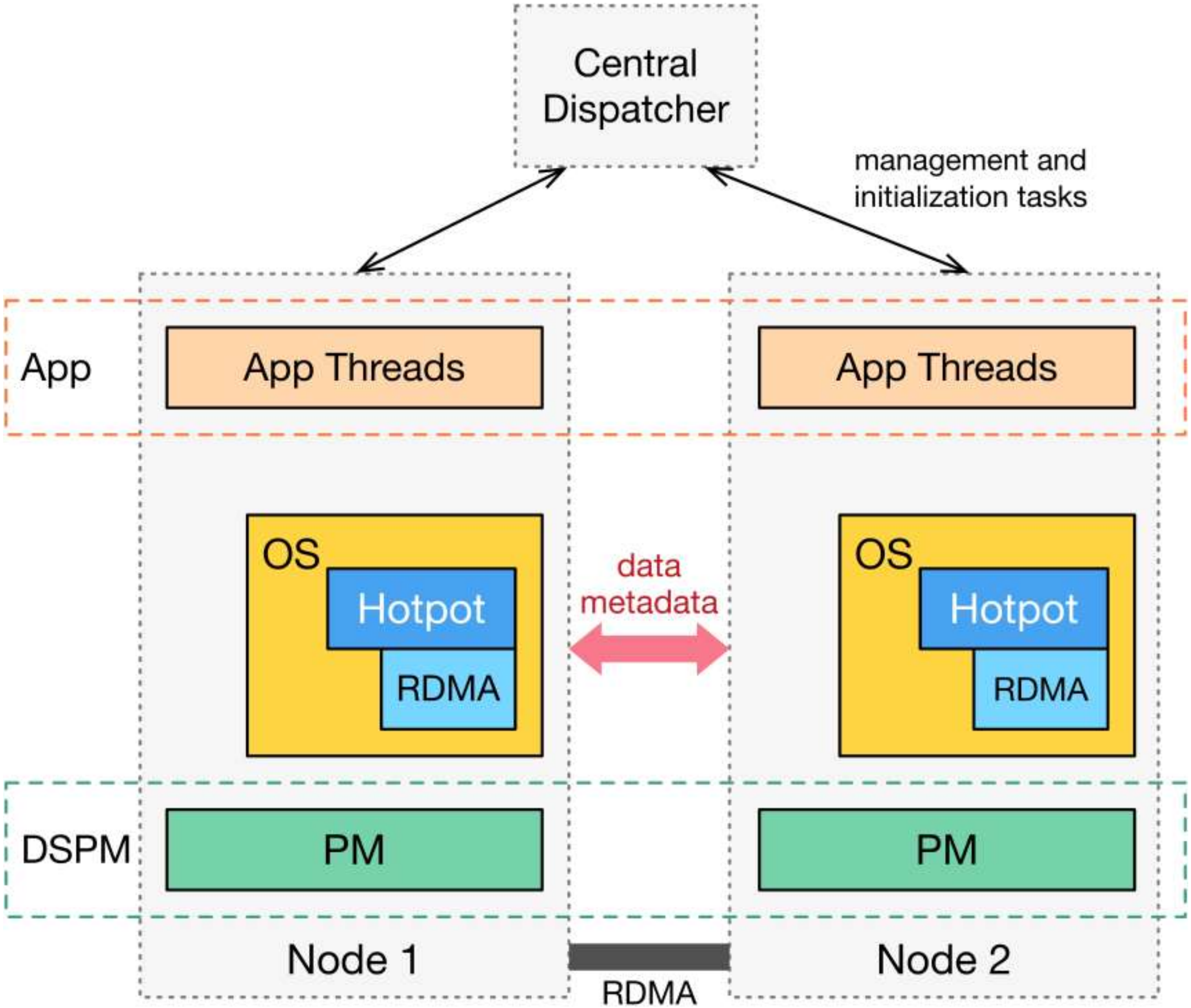
Hotpot Architecture



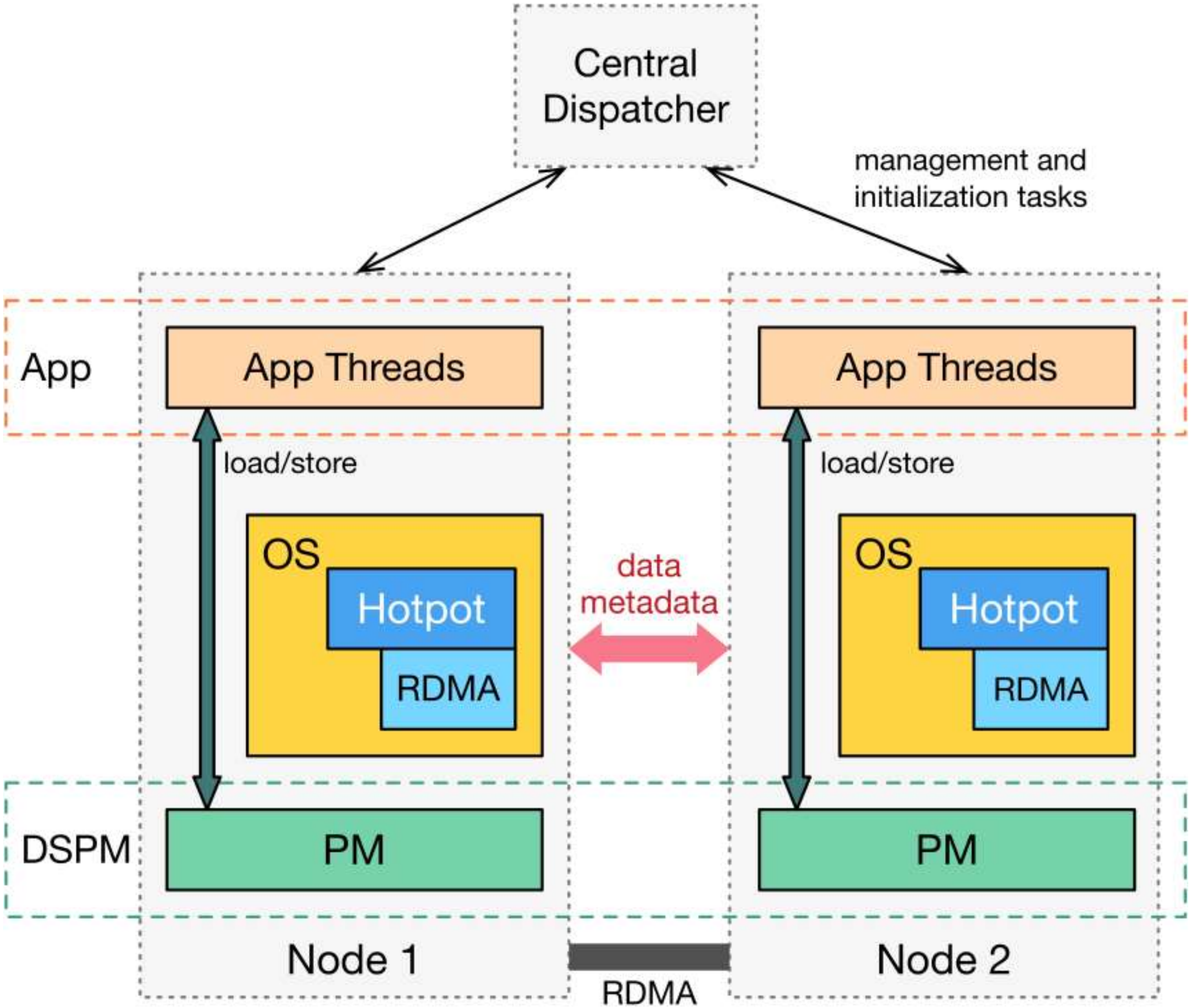
Hotpot Architecture



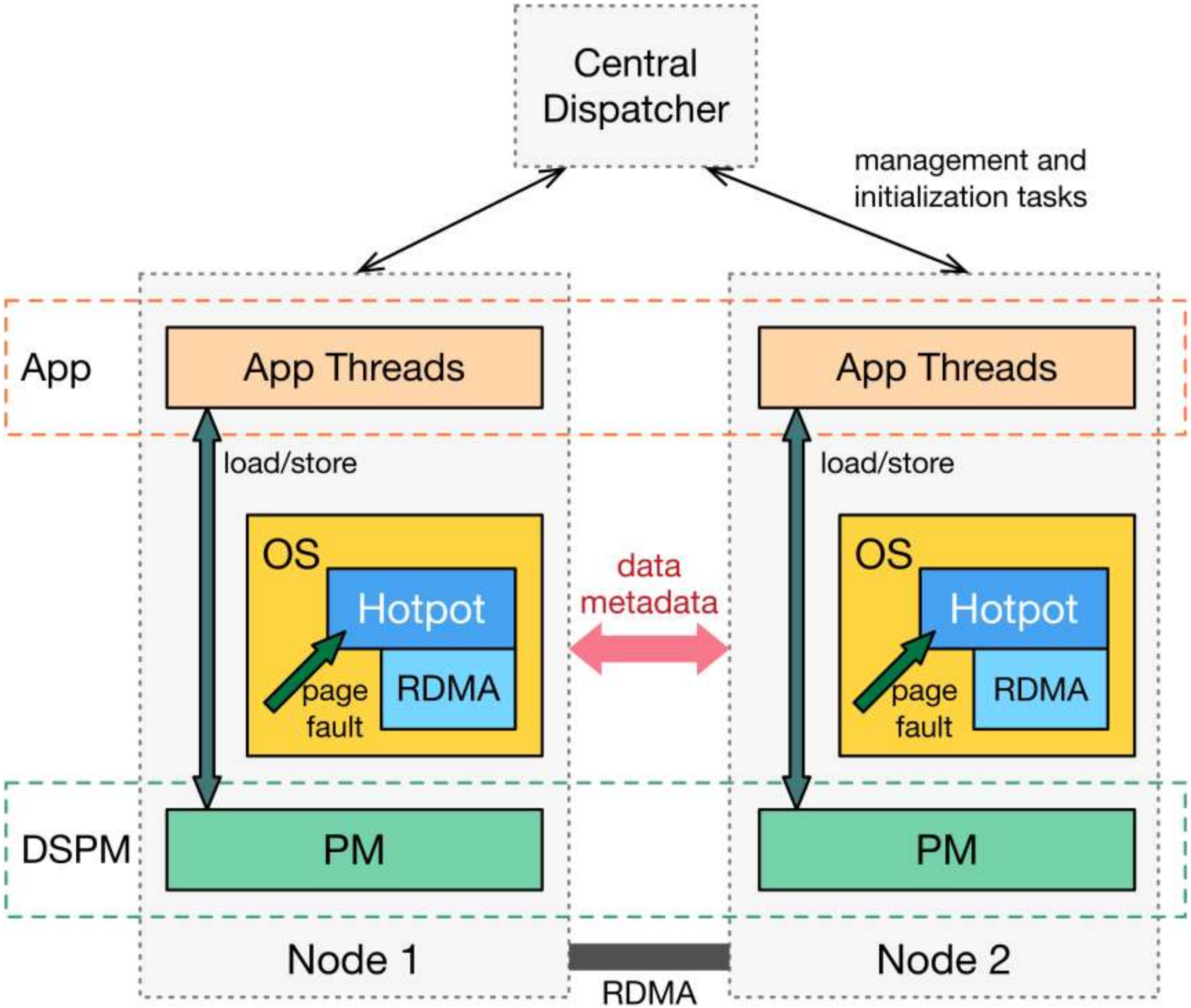
Hotpot Architecture



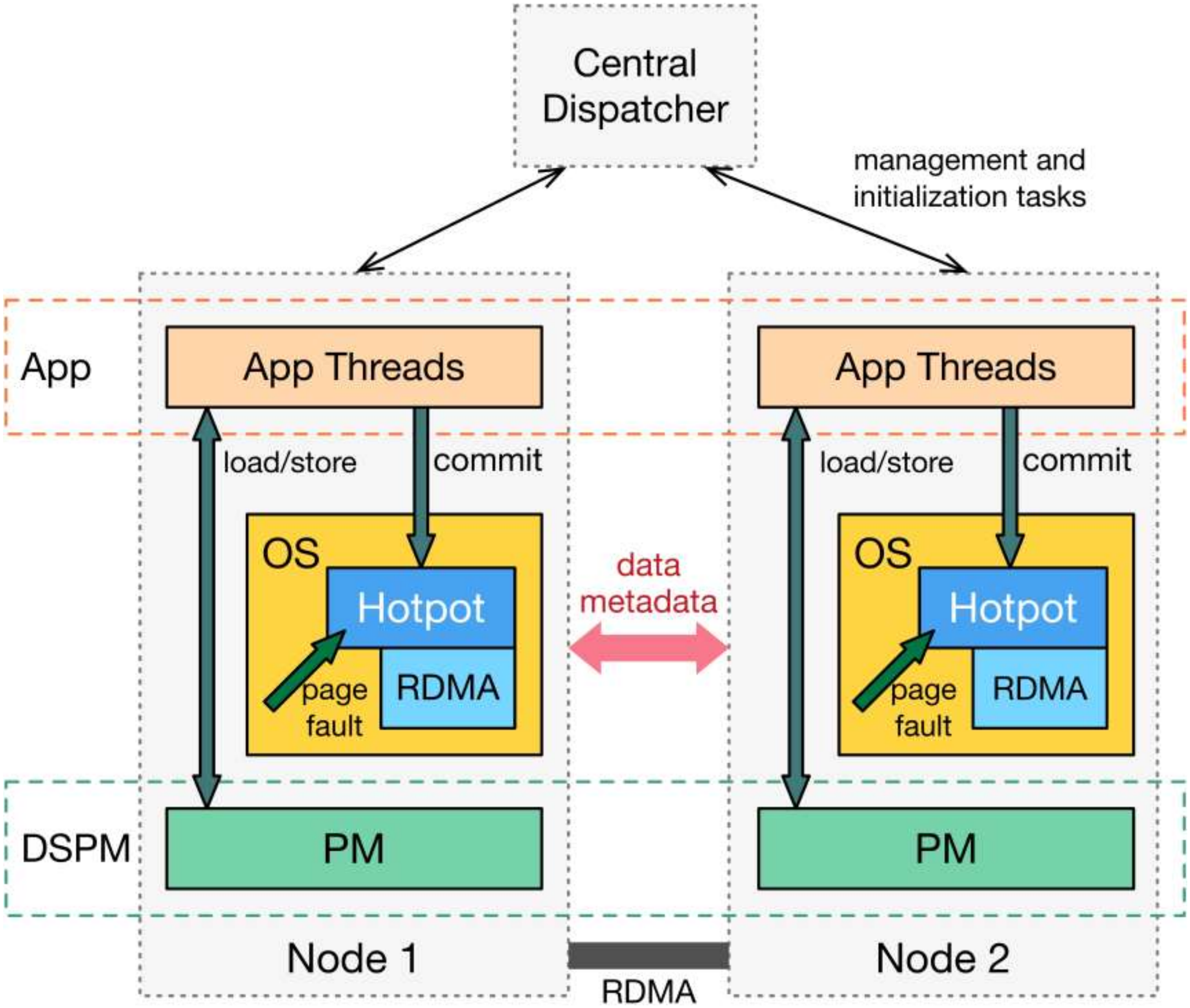
Hotpot Architecture



Hotpot Architecture



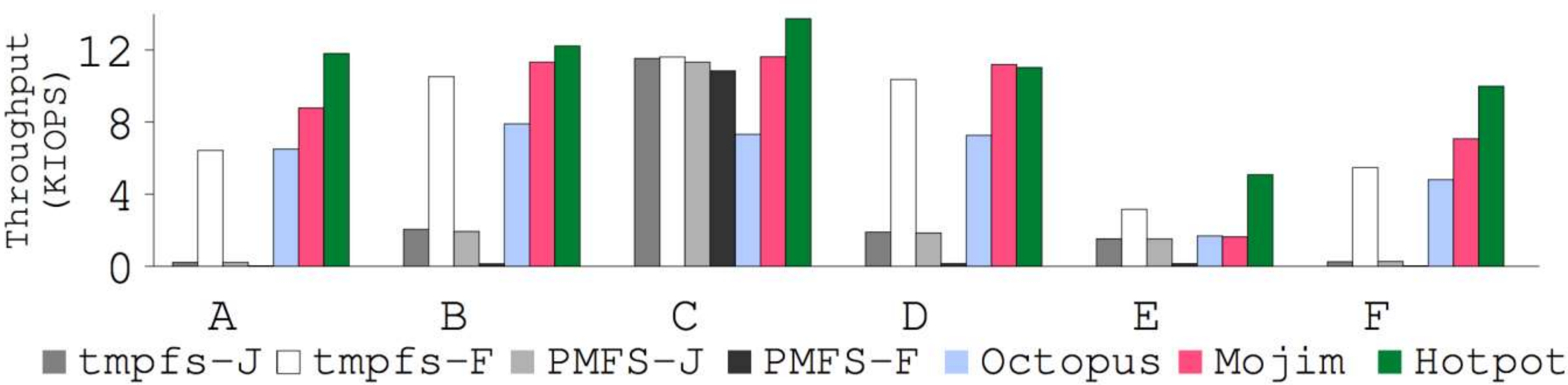
Hotpot Architecture

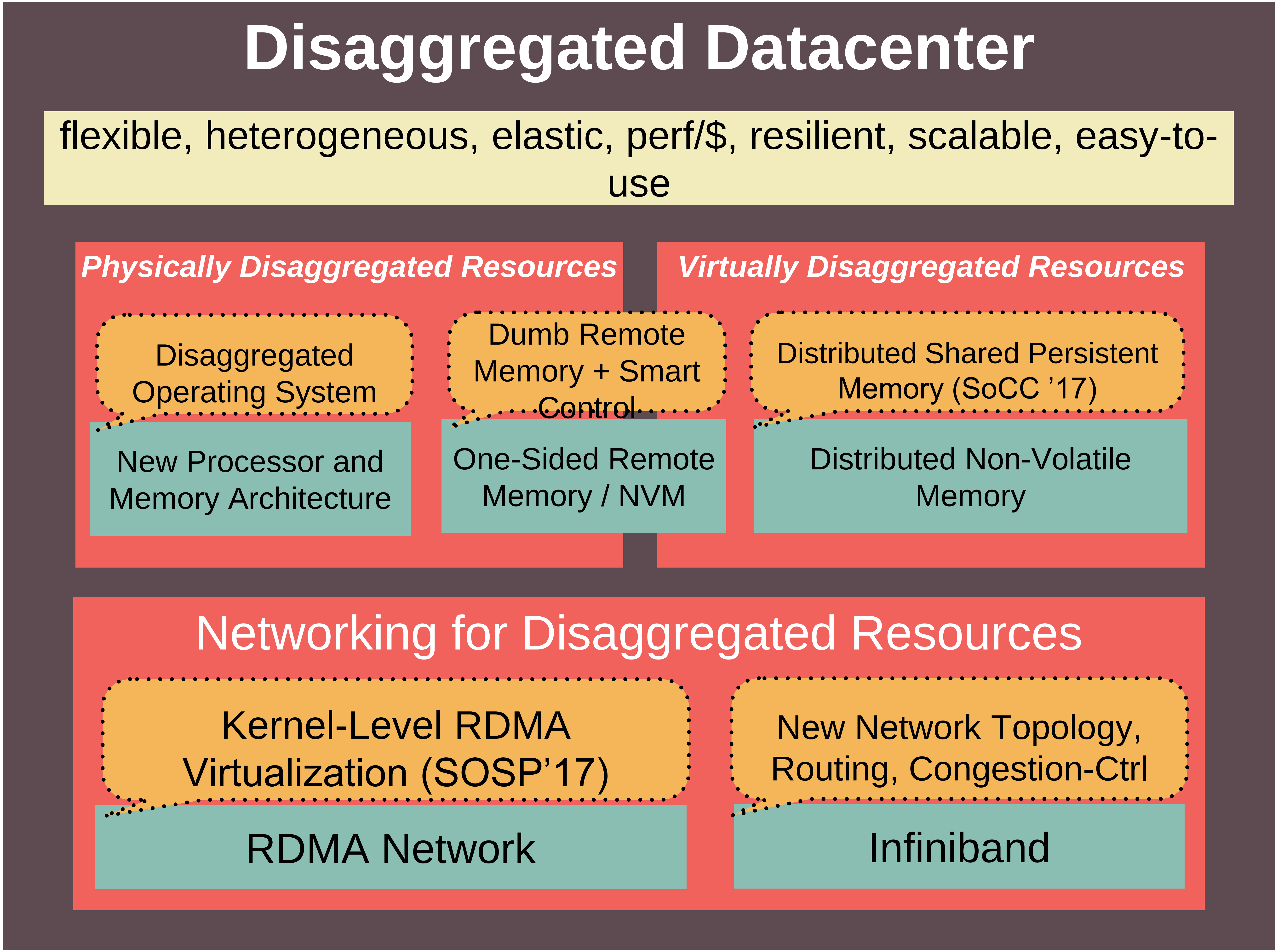


MongoDB Results

- Modify MongoDB with ~120 LOC, use MRMW mode
- Compare with ***tmpfs***, ***PMFS***, ***Mojim***, ***Octopus*** using ***YCSB***

Workload	Read	Update	Scan	Insert	R&U
A	50%	50%	-	-	-
B	95%	5%	-	-	-
C	100%	-	-	-	-
D	95%	-	-	5%	-
E	-	-	95%	5%	-
F	50%	-	-	-	50%




Disaggregated Datacenter
flexible, heterogeneous, elastic, perf/$, resilient, scalable, easy-to-use
Physically Disaggregated Resources
Virtually Disaggregated Resources
Disaggregated Operating System
Dumb Remote Memory + Smart Control
Distributed Shared Persistent Memory (SoCC '17)
New Processor and Memory Architecture
One-Sided Remote Memory / NVM
Distributed Non-Volatile Memory
Networking for Disaggregated Resources
Kernel-Level RDMA Virtualization (SOSP'17)
New Network Topology, Routing, Congestion-Ctrl
RDMA Network
Infiniband

Conclusion

- New hardware and software trends point to resource disaggregation
- WukLab is building an end-to-end solution for disaggregated datacenter
- Opens up new research opportunities in hardware, software, networking, security, and programming language

Thank you
Questions?

wuklab.io