The Cloud Without the Fluff:
Rethinking Resource Disaggregation

Ryan Stutsman
University of Utah

Utah Scalable Computer
Systems Lab

There are smart people behind this work

Tian (Candy) Zhang

Chinmay Kulkarni

Mazhar Naqvi

Ankit Bhardwaj

Jacob Barzee

Robert Ricci

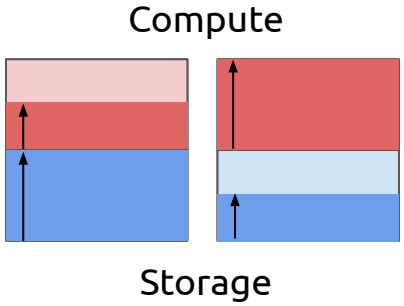
Sara Moore

Outline

1. Basically, a rant on RDMA
2. Splinter, some real work we've done
3. Raving aspirations & fawning over other people's work

RDMA Rant

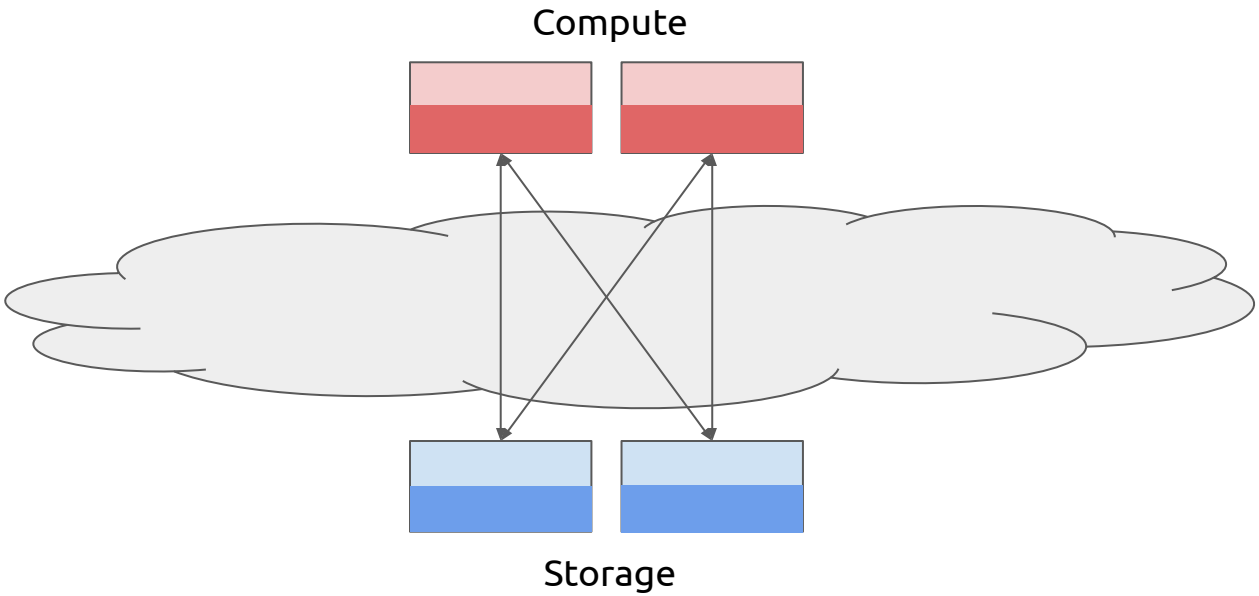
The Bad Old Days



Tight Coupling → Poor Utilization

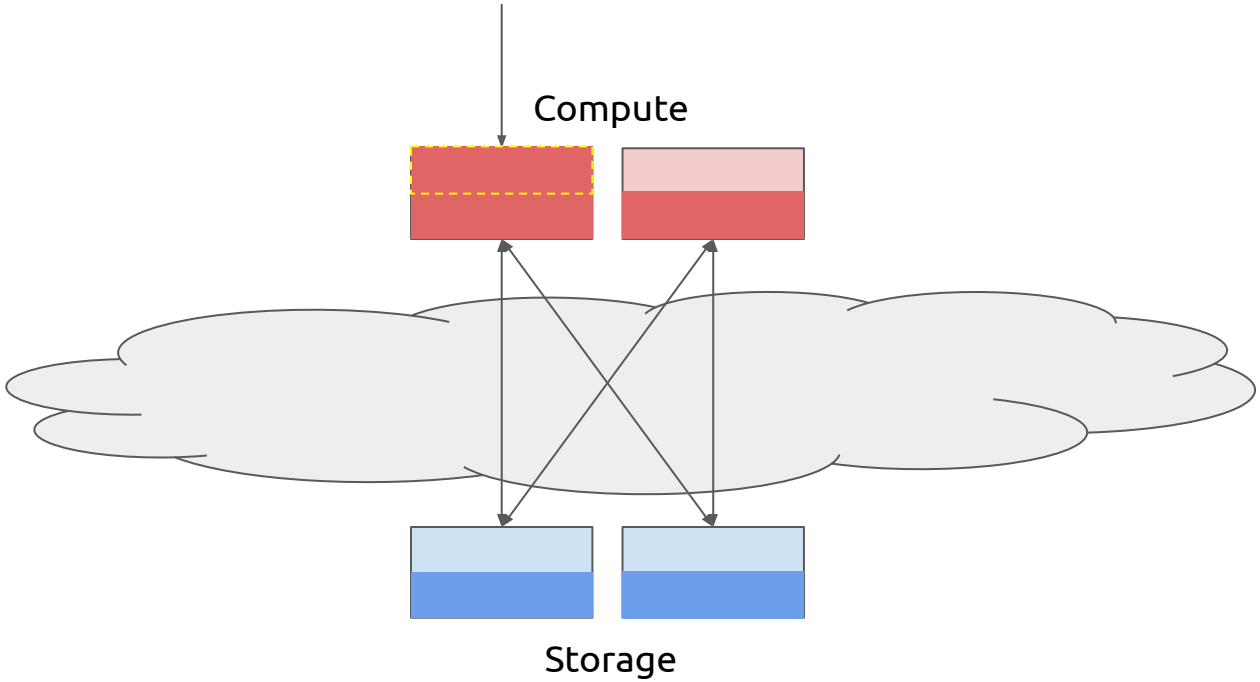
Complicated Fault-tolerance

Disaggregation to the Rescue



Decouple Compute & Storage using Network

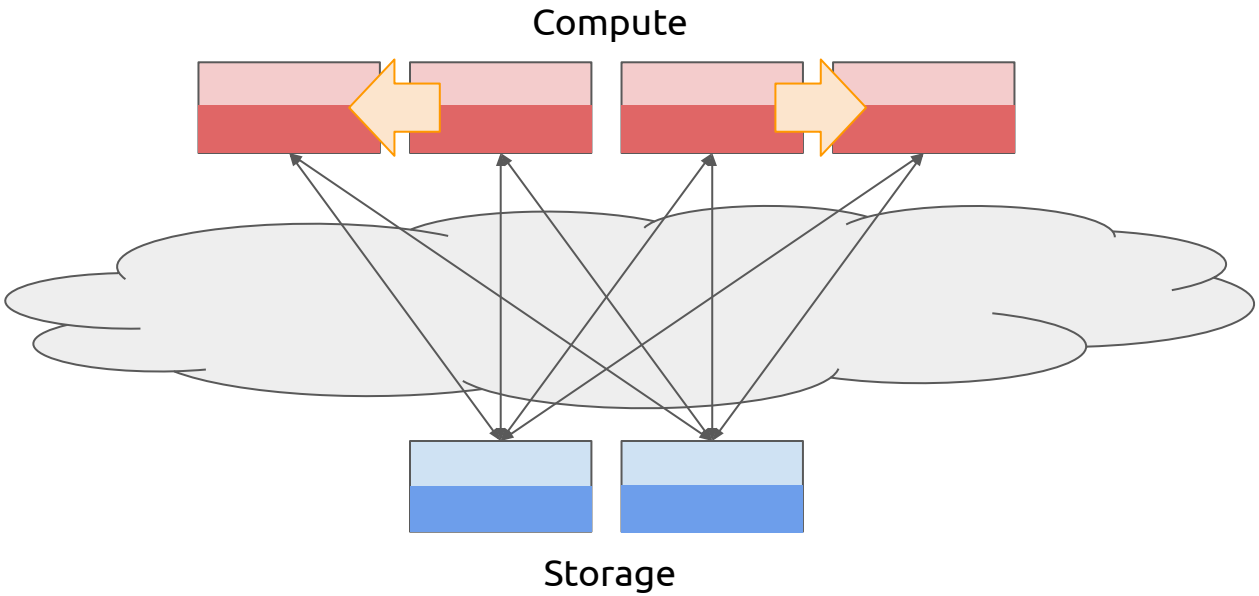
Disaggregation to the Rescue



Decouple Compute & Storage using Network

Provision at Idle Capacity

Disaggregation to the Rescue

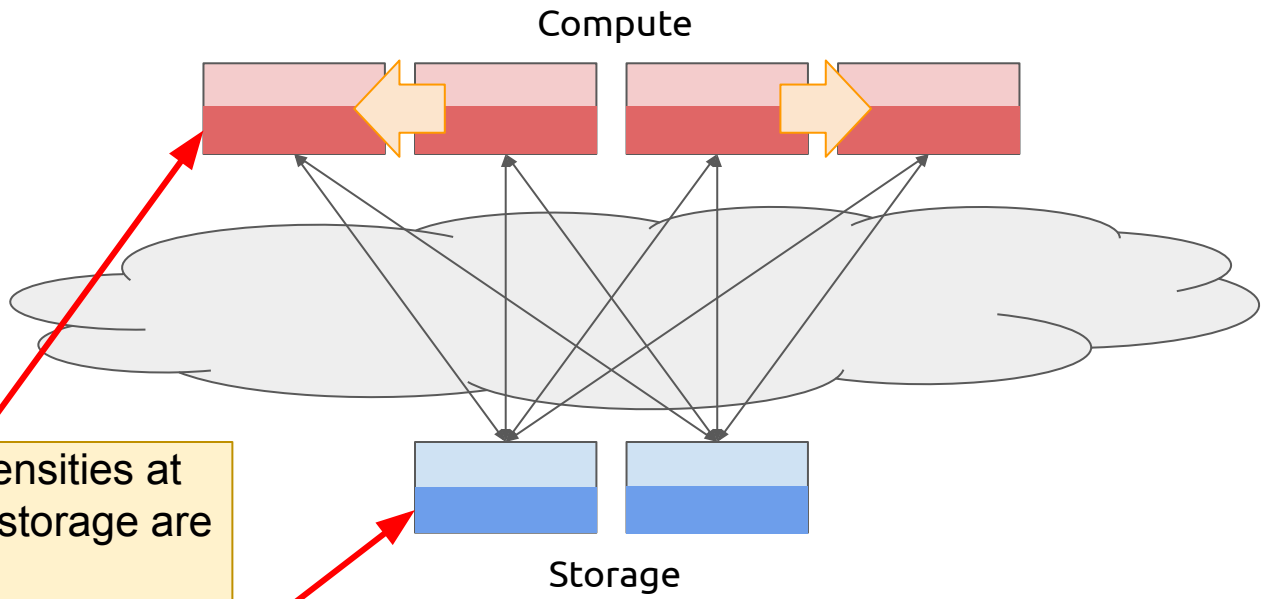


Decouple Compute & Storage using Network

Provision at Idle Capacity

Scale Independently

Disaggregation to the Rescue



Key: tenant densities at compute and storage are **very** different.

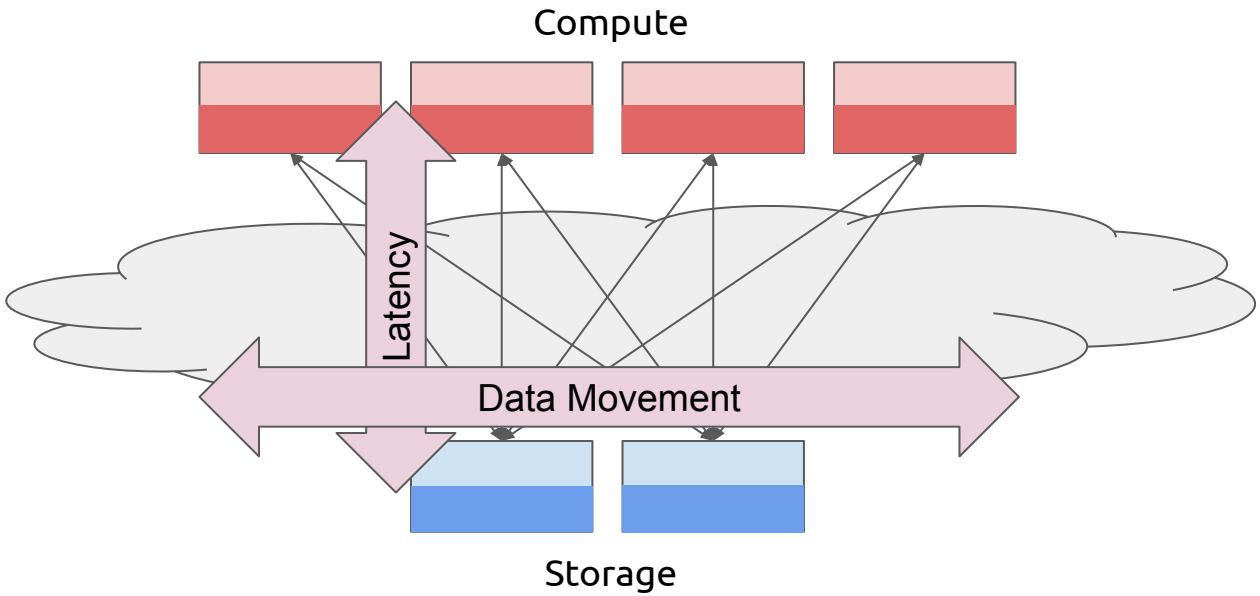
10s of VMs
vs
1000s of storage tenants per machine

Compute & Storage using Network

Provision at Idle Capacity

Scale Independently

The Fly in the Ointment



- High-delay Network Stacks Leave Compute Stalled
- Massive Data Movementment Destroys Efficiency
- Can Run Tenant Logic to Avoid Some Movement

1 µs @ 200 Gbps!

The End of a Myth: Distributed Transactions Can Scale

Erfan Zamanian
Brown University
erfanz@cs.brown.edu

Carsten Binnig
Brown University
carsten.binnig@brown.edu

Tim Harris
Oracle Labs
timothy.l.harris@oracle.com

Tim Kraska
Brown University
tim_kraska@brown.edu

ABSTRACT

The common wisdom is that distributed transactions do not scale. But what if distributed transactions could be made scalable using the next generation of networks and a redesign of distributed databases? There would no longer be a need for developers to worry about co-partitioning schemes to achieve decent performance. Application development would become easier as data placement would no longer determine how scalable an application is. Hardware provisioning would be simplified as the system administrator can expect a linear scale-out when adding more machines rather than some complex sub-linear function, which is highly application specific.

In this paper, we present the design of our novel scalable database system NAM-DB and show that distributed transactions with the very common Snapshot Isolation guarantee can indeed scale using the next generation of RDMA-enabled network technology without any inherent bottlenecks. Our experiments with the TPC-C benchmark show that our system scales linearly to over 6.5 million new-order (14.5 million total) distributed transactions per second on 56 machines.

1 Introduction

The common wisdom is that distributed transactions do not scale [40, 22, 39, 12, 37]. As a result, many techniques have been proposed to avoid distributed transactions ranging from locality-aware partitioning [35, 33, 12, 43] and speculative execution [32] to new consistency levels [24] and the relaxation of durability guarantees [25]. Even worse, most of these techniques are not transparent to the developer. Instead, the developer not only has to understand all the implications of these techniques, but also must carefully design the application to take advantage of them. For example, Oracle requires the user to carefully specify the co-location of data using special SQL constructs [15]. A similar feature was also recently introduced in Azure SQL Server [2]. This works well as long as all queries are able to respect the partitioning scheme. However, transactions crossing partitions usually observe a much higher abort rate and relatively unpredictable performance [9]. For other applications (e.g., social apps), a developer might not even be able to design a proper sharding scheme since those applications are notoriously hard to partition.

But what if distributed transactions could be made scalable using the next generation of networks and we could rethink the distributed database design? What if we would treat every transaction as a distributed transaction? The performance of the system would become more predictable. The developer would no longer need to worry about co-partitioning schemes in order to achieve scalability and decent performance. The system would scale out linearly when adding more machines rather than sub-linearly because of partitioning effects, making it much easier to provision how much hardware is needed.

Would this make co-partitioning obsolete? Probably not, but its importance would significantly change. Instead of being a necessity to achieve a scalable system, it becomes a second-class design consideration in order to improve the performance of a few selected queries, similar to how creating an index can help a selected class of queries.

In this paper, we will show that distributed transactions with the common Snapshot Isolation scheme [8] can indeed scale using the next generation of RDMA-enabled networking technology without an inherent bottleneck other than the workload itself. With Remote-Direct-Memory-Access (RDMA), it is possible to bypass the CPU when transferring data from one machine to another. Moreover, as our previous work [10] showed, the current generation of RDMA-capable networks, such as InfiniBand FDR 4×, is already able to provide a bandwidth similar to the aggregated memory bandwidth between a CPU socket and its attached RAM. Both of these aspects are key requirements to make distributed transactions truly scalable. However, as we will show, the next generation of networks does not automatically yield scalability without redesigning distributed databases. In fact, when keeping the "old" architecture, the performance can sometimes even de-

Transformative Performance! … Right?

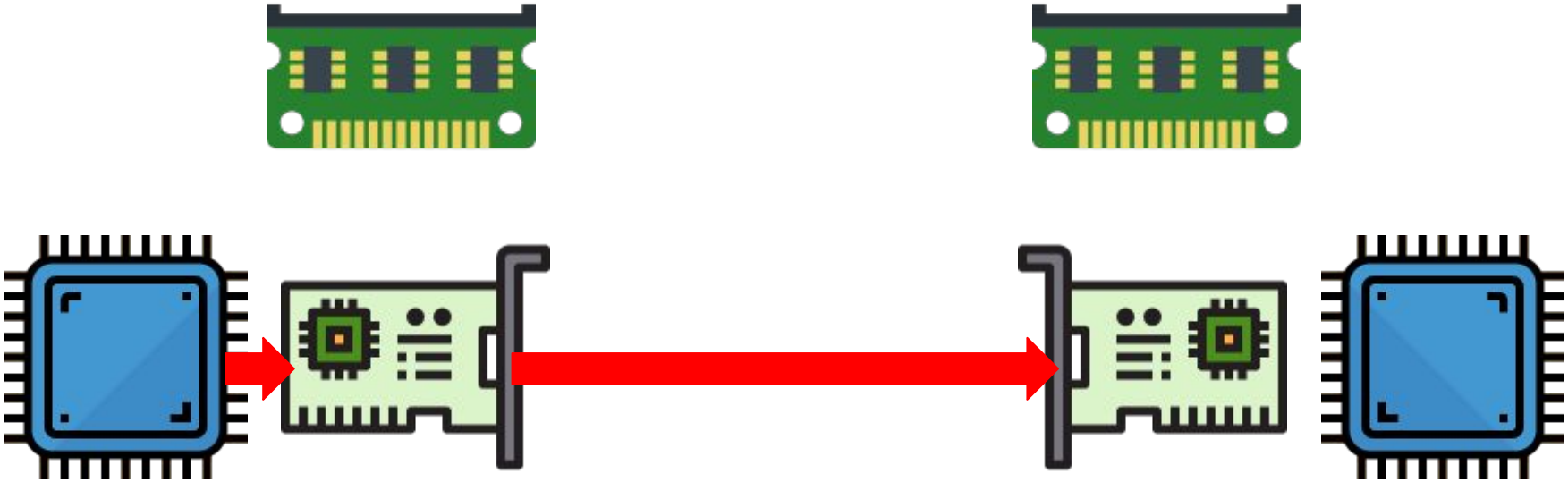
1-sided RDMAs in approximately one slide

READ,
Remote VA,
Local VA

Memory-mapped I/O over PCIe to post RDMA descriptor

Remote virtual address, length, & local address target

1-sided RDMAs in approximately one slide



Local NIC forward RDMA READ operation to remote NIC

1-sided RDMAs in approximately one slide

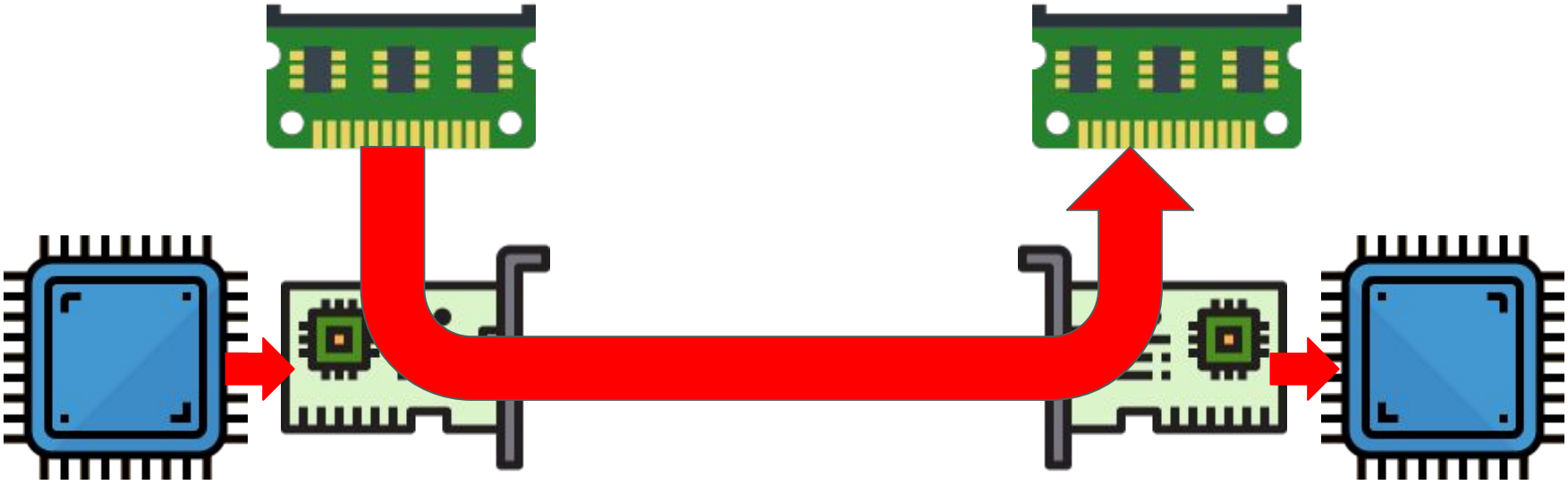
READ,
Remote VA,
Local VA

Remote NIC transfers requested back to client

Local NIC DMAs data from NIC into chosen destination

no CPU involvement at the remote machine

2-sided RDMAs (for completeness)



Remote machine gets a notification of recv'd message

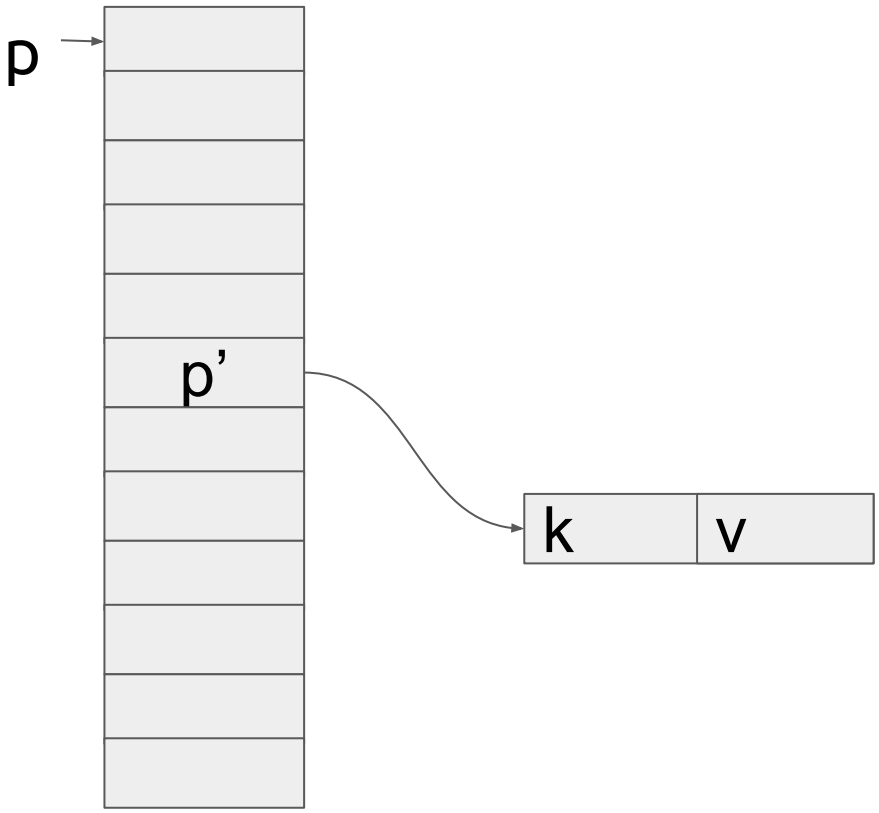
Still uses fast DMA, but "activates" remote CPU
(which would usually be polling for the message)

The Tricky Part of 1-sided RDMA

Local

get(k)?

Remote



The Tricky Part of 1-sided RDMA

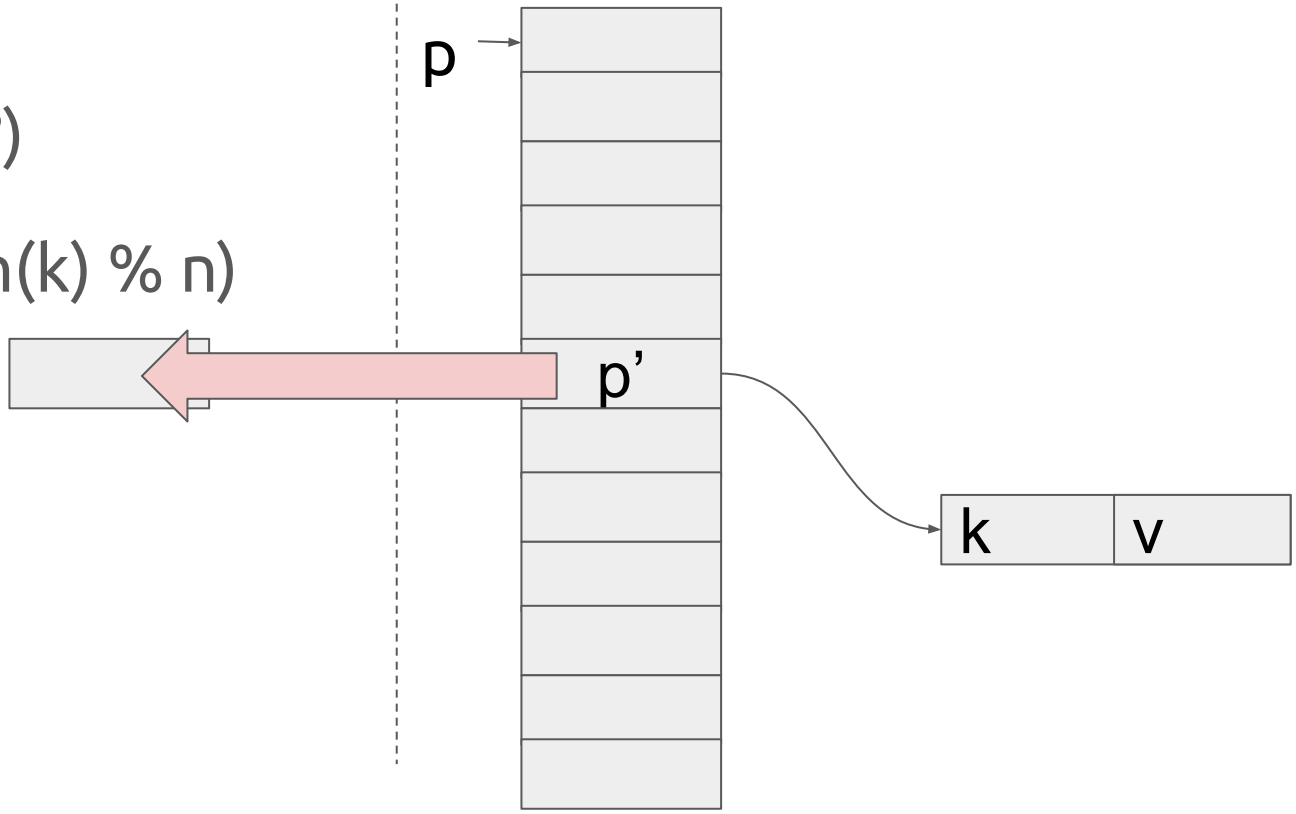
Local

get(k)?

rdma_read(p + ?)

rdma_read(p + h(k) % n)

Remote



The Tricky Part of 1-sided RDMA

Local

get(k)?

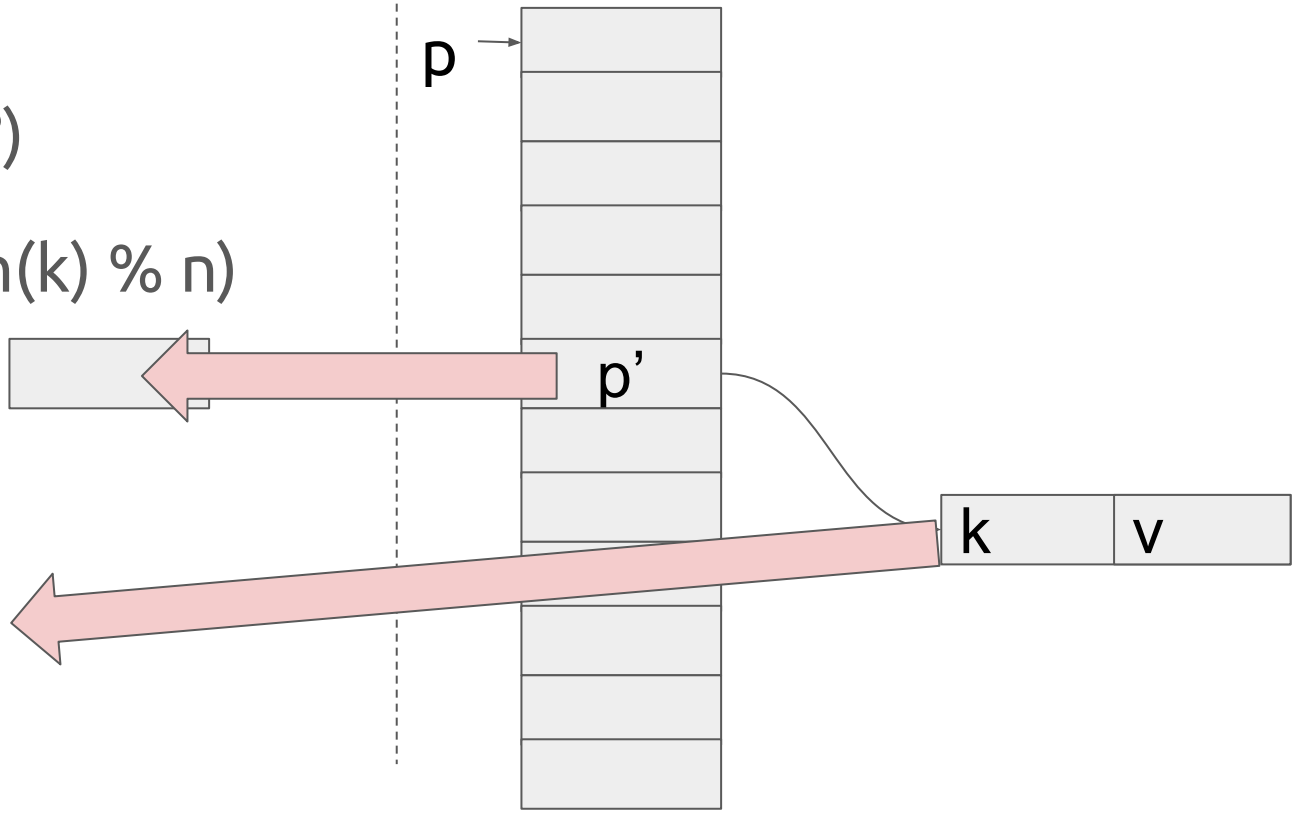
rdma_read(p + ?)

rdma_read(p + h(k) % n)

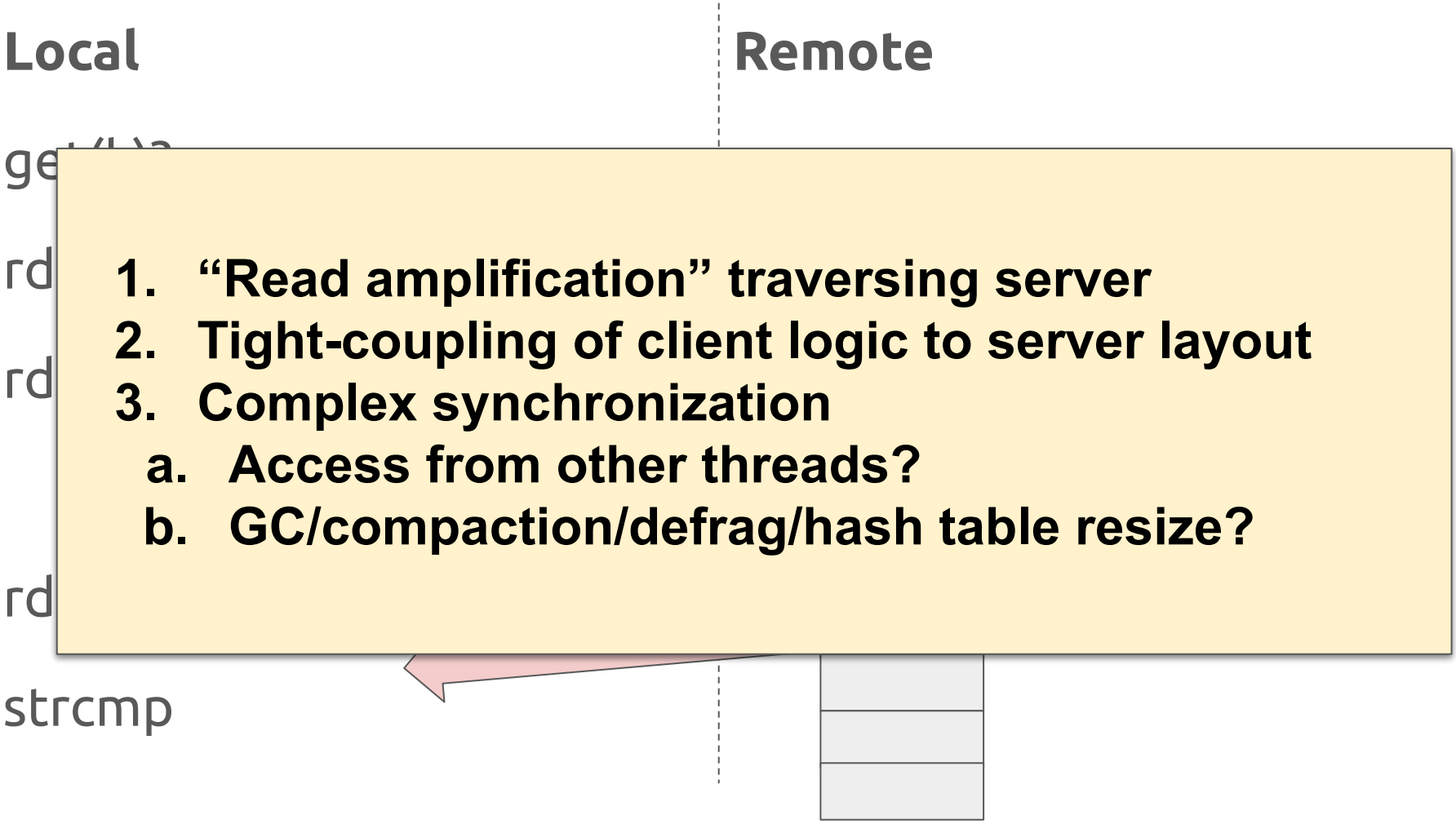
rdma_read(p')

strcmp

Remote

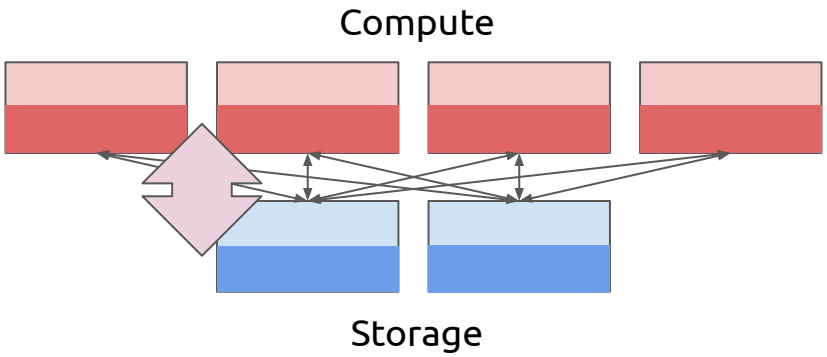


The Tricky Part of 1-sided RDMA



1-sided RDMA

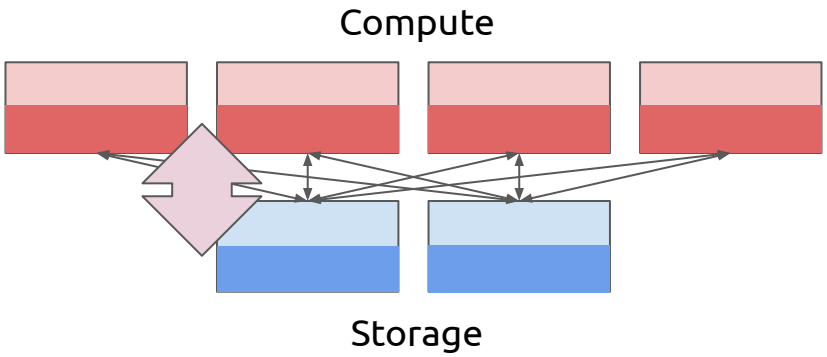
~~ms delays~~
→
µs delays



10-100× latency improvement

1-sided RDMA

~~ms delays~~
→
µs delays

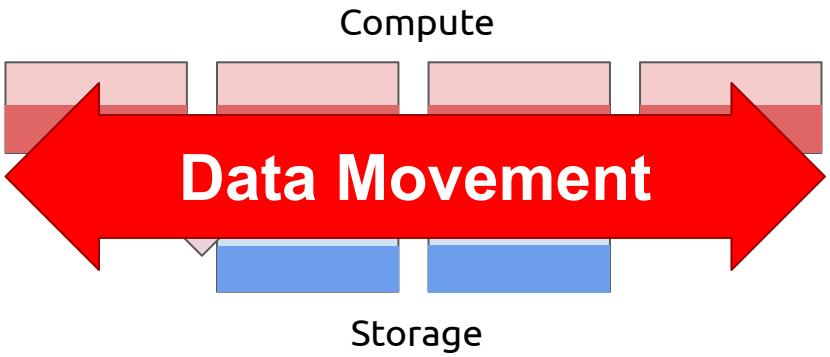


10-100× latency improvement

Restricted Programming Interface

1-sided RDMA

~~ms delays~~
→
µs delays



10-100× latency improvement

Restricted Programming Interface

Explosion in Data Movement

Fast and Dumb?

Catch 22

> 10,000 machines blasting data at 200 Gbps doesn’t make sense

But, **making storage tier programmable**
risks locking compute and storage tier

We can’t go back to poor scaling, utilization, and fault-tolerance!

Splinter [OSDI'18]

The Splinter Vision in a Nutshell

What would it take to solve this?

The Splinter Vision in a Nutshell

What would it take to solve this?

µs-scale Storage-level Functions

10s of Millions of Invocations per Second per Server
(**2-sided ops!**)

Native Code Performance

Strong Isolation for ≥ 10,000 Tenants

Dynamic Placement of Function Execution

The Splinter Vision in a Nutshell

What would it take to solve this?

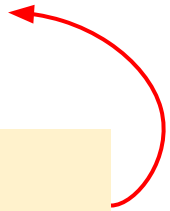
µs-scale Storage-level Functions

10s of Millions of Invocations per Second per Server
(**2-sided ops!**)

Native Code Performance

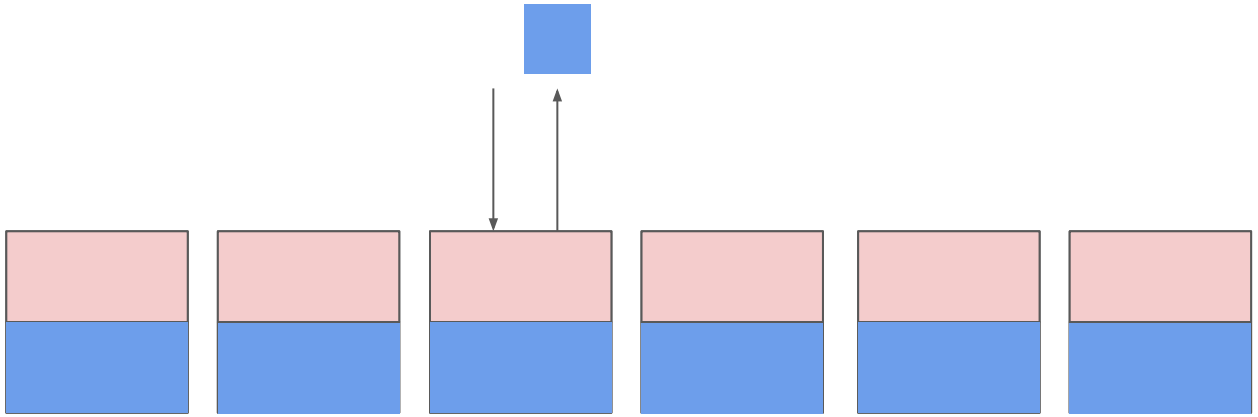
Strong Isolation for ≥ 10,000 Tenants

Dynamic Placement of Function Execution



Avoids data movement
without coupling compute and storage

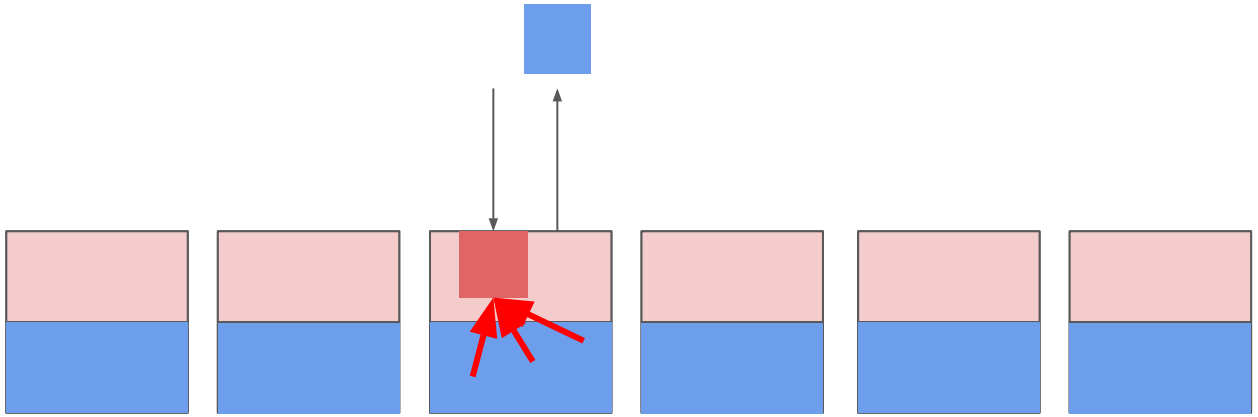
Splinter in Action: Simple CRUD



Kernel-bypass & Zero-copy Networking

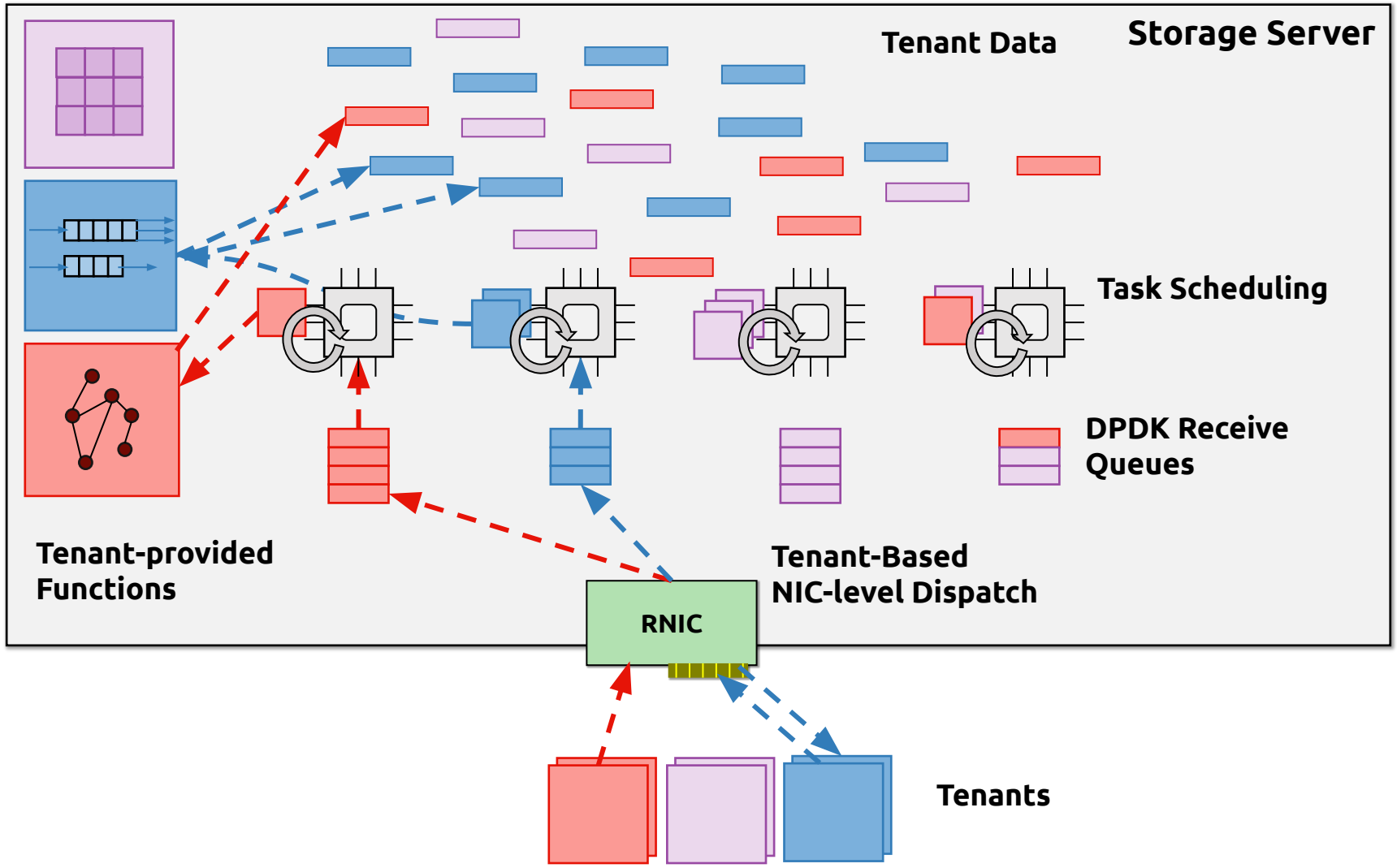
Lookup of Small Data Item → ~10 µs

Splinter in Action: UDF Invocation

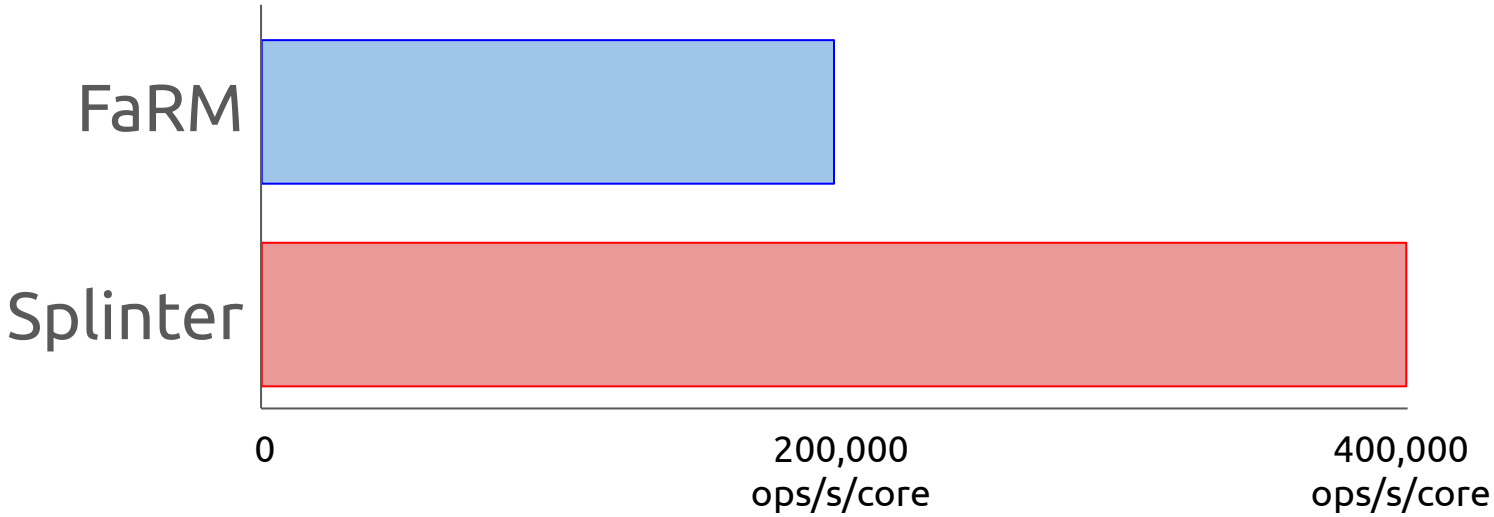


Invocation of Small User-defined Function → ~10 µs

Internal Dispatch and Request Processing



Facebook Tao



"Onloading" **assoc_range** LinkBench ops improves tput

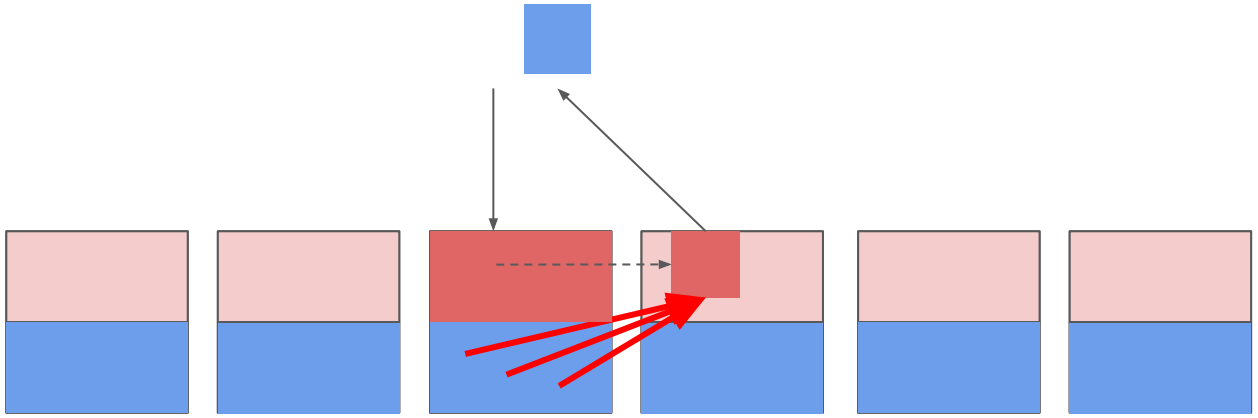
Take with a massive grain of salt

old version of FaRM, different hw, older NICs, etc etc.

Shows promise - combining is actually best

Problem: only works if not server CPU bottlenecked

Splinter in Action



Dynamically shift invocations to avoid bottlenecks

Exploit idle compute **anywhere**

No data movement in the normal case

No tight coupling of compute and storage

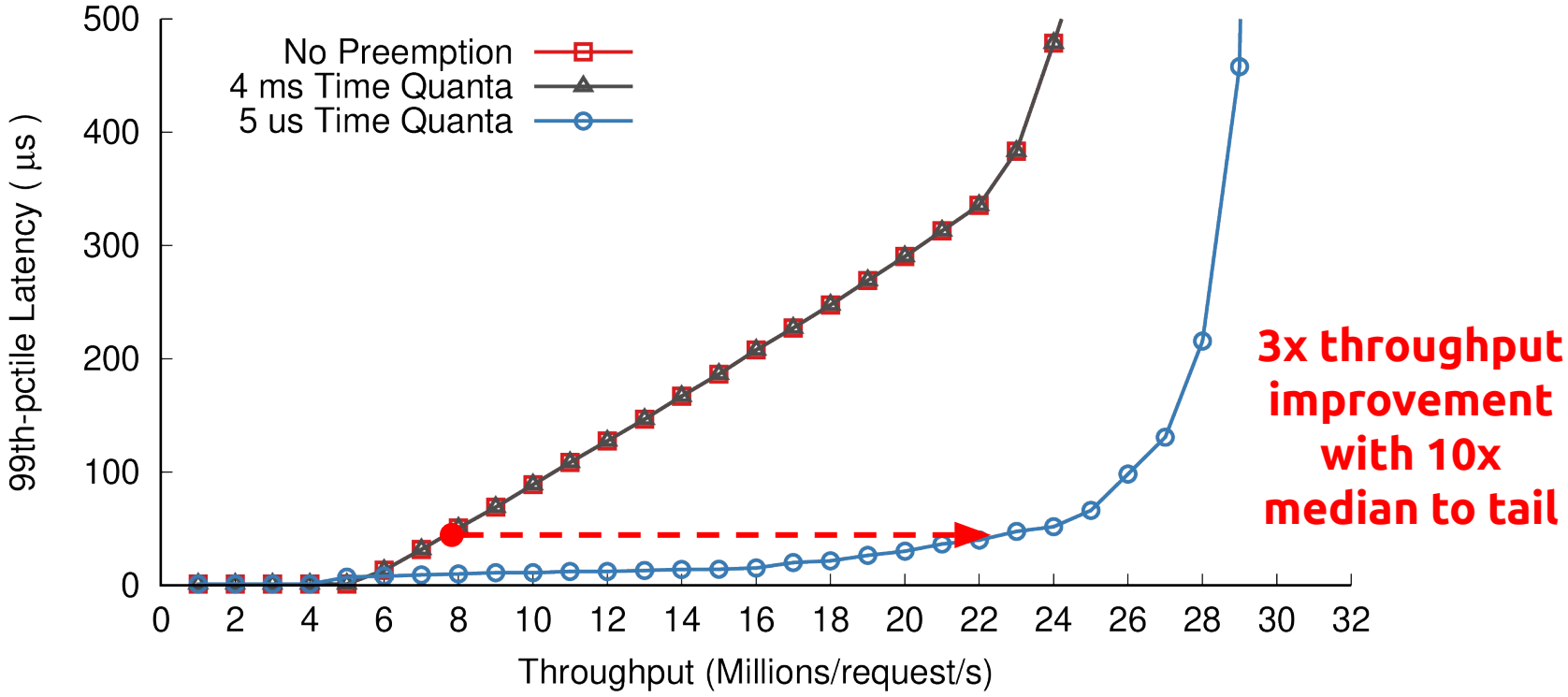
Applying Others’ Amazing Ideas

Two Key Challenges

- Fine-grained Scheduling
  - Head-of-line-blocking → preemption → inefficient kthreads?
  - New ideas virtualizing CPU interrupt controller
  - Now possible to **preempt tasks at 5 µs** granularity
  - Great for tail latency!

- Strong Protection
  - Speed → native code → hardware isolation → slow
  - ≥10,000 functions → context switches → low tput, utilization
  - Fast **ring 3 to ring 3 address space switching** with VMFUNC

The Benefit of Fine-grained Preemption

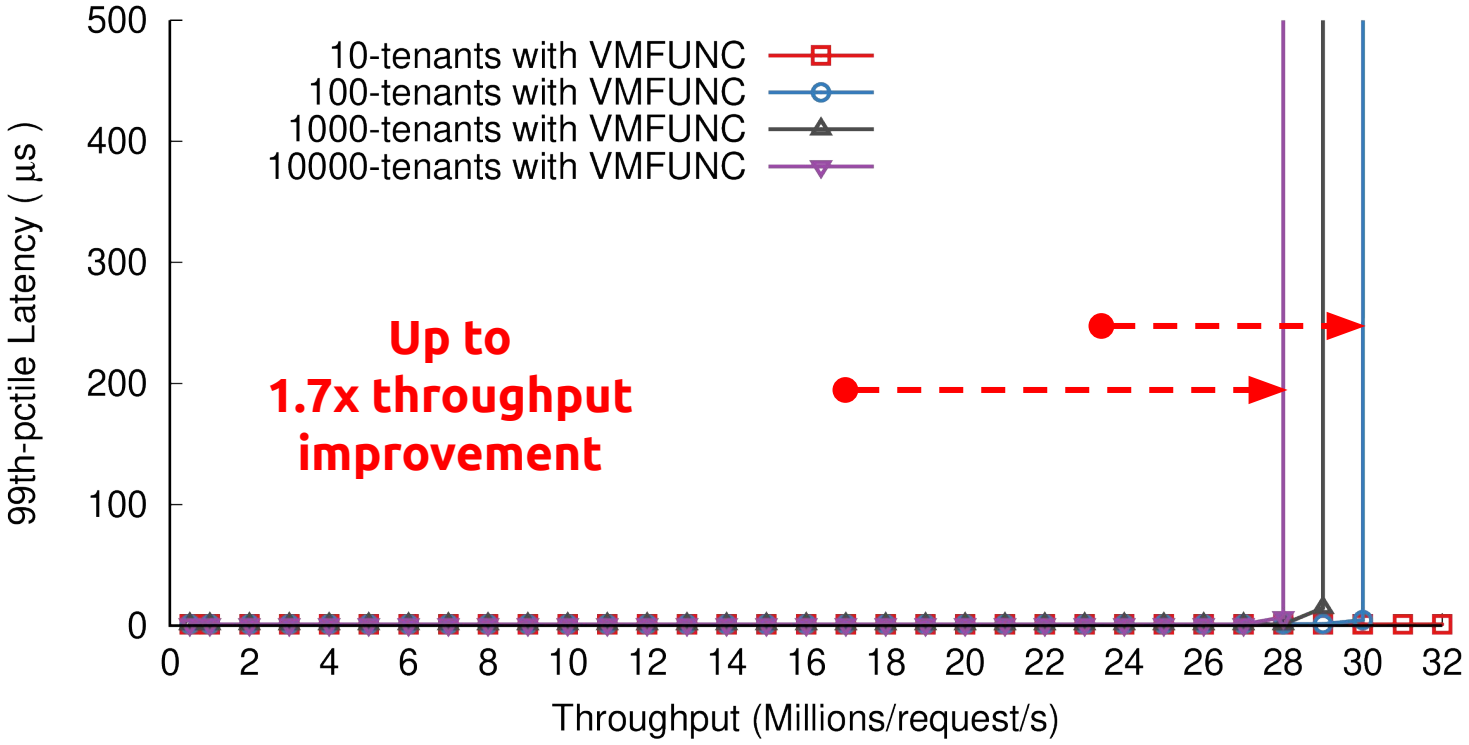


Simulated 32 cores with open-loop load clients

99.9% 1 µs requests with 0.1% 500 µs requests

HOB destroys response times unless **5 µs quanta**

New Hw Protection Scales Better



Conventional **page table switch** cuts to ~**½ throughput**

New hw schemes give good performance & VM isolation

Fast enough that DB/storage can use ubiquitously

Current Status

(Mostly) cooperative + language-based protection

- Works but struggles if functions run long
- Vulnerable to speculative execution attacks

Currently assessing secondary impacts of fine-grained preemption and protection

- I-cache and TLB pressure likely to be significant

Comparison with Morsel Parallelism

Does this matter if a store JITs SQL queries?

- SQL can be compiled to be cooperative
- Can break into granular chunks of work and balance

But,

- This doesn’t help with UDFs
- Only works if JIT’ed coded is trusted
- And still might be less efficient
  - (pre-emption can have 0 cost if it isn’t triggered)

Summary

Fast networks don’t solve disaggregation inefficiency

Delivering µs-scale kernel-bypass performance at scale is **more than a network problem**

Current scheduling and protection approaches breakdown

Storage layer/database will need fine-grained control over scheduling and protection

Image Attributions:

Blue Flame: Public Domain; wikipedia.com
Network card icon: Icon made by srip from www.flaticon.com
CPU icon: Icon made by monkik from www.flaticon.com
Memory icon: Icon made by Smashicons from www.flaticon.com