RAMCloud Overview

John Ousterhout

Stanford University

Introduction

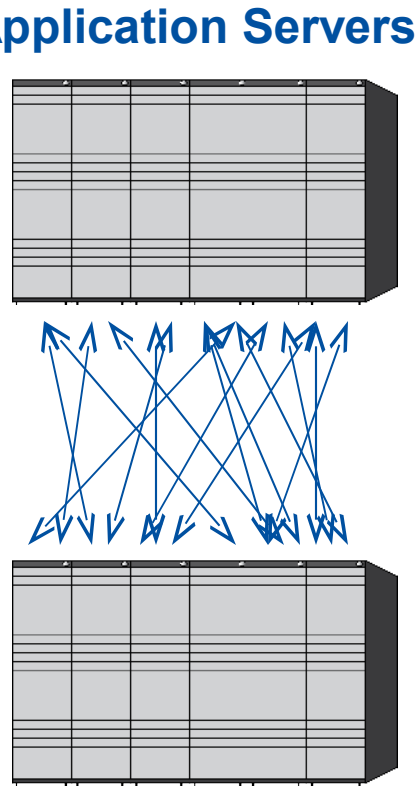
- **Large-scale storage system entirely in DRAM**
- **Interesting combination: scale, low latency**
- **Enable new applications?**
- **The future of datacenter storage?**

Outline

- **Overview of RAMCloud**
- **Motivation**
- **Research challenges**
- **Basic cluster structure and data model**

The Basic Idea

- **Storage for datacenters**
- **1000-10000 commodity servers**
- **32-64 GB DRAM/server**
- **All data always in RAM**
- **Durable and available**
- **Performance goals:**
  - High throughput:
    1M ops/sec/server
  - Low-latency access:
    5-10µs RPC

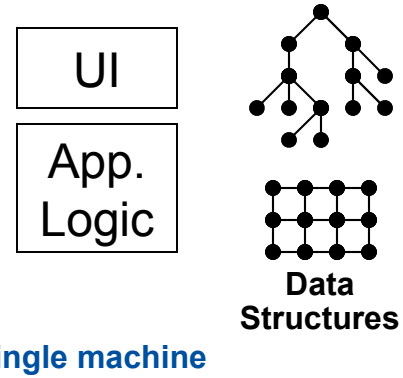


Example Configurations

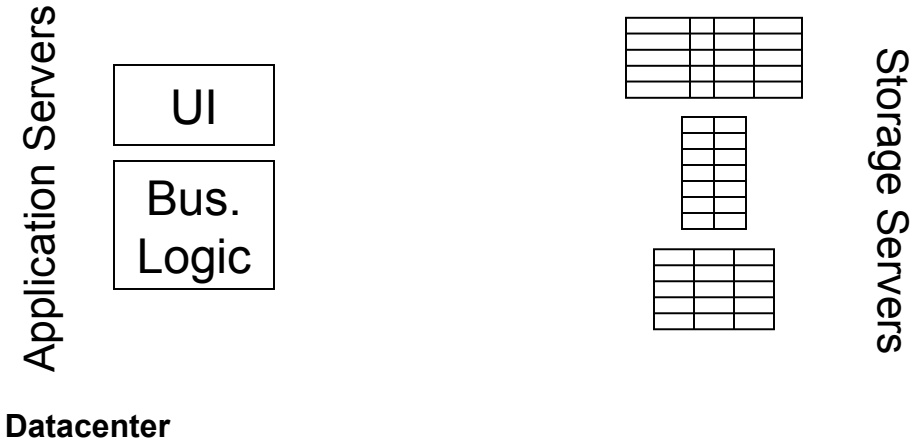
	Today	5-10 years
# servers	2000	4000
GB/server	24GB	256GB
Total capacity	48TB	1PB
Total server cost	$3.1M	$6M
$/GB	$65	$6

RAMCloud Motivation: Latency

Traditional Application



Web Application

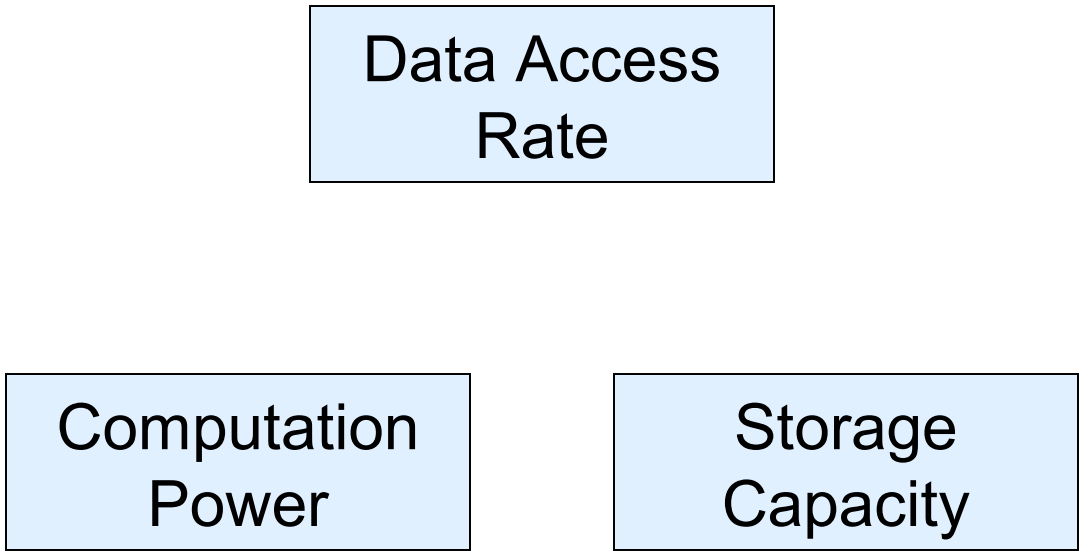


<< 1µs latency

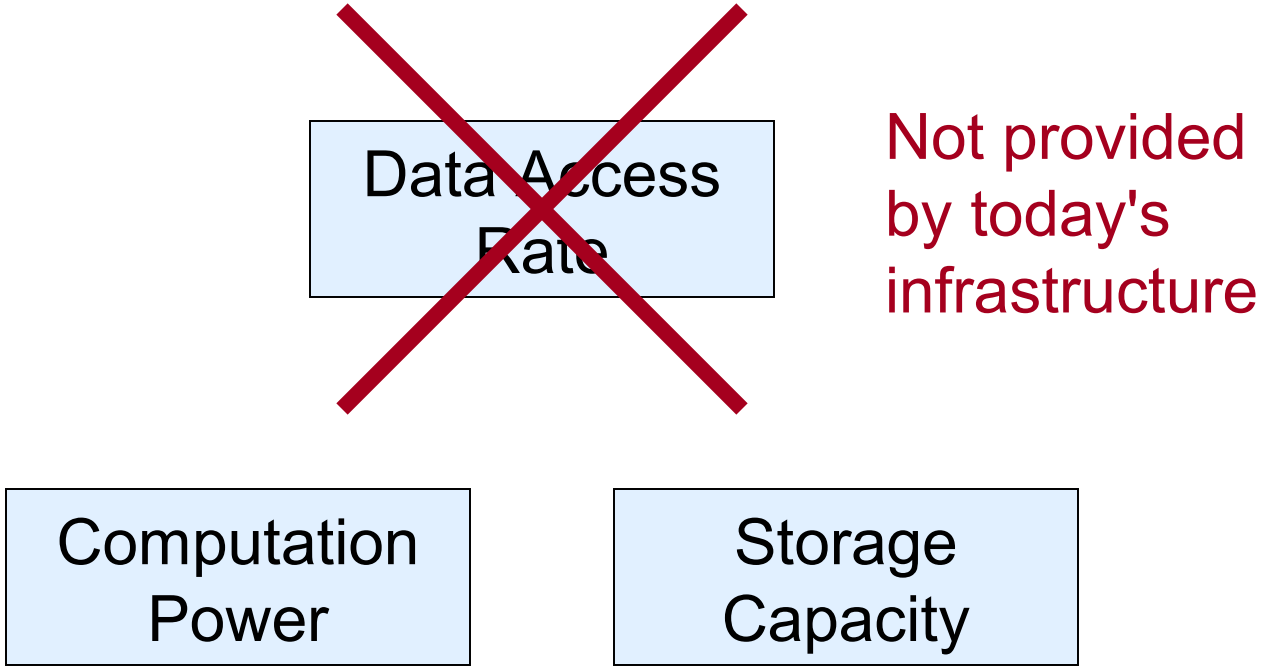
0.5-10ms latency

- **Large-scale apps struggle with high latency**
  - Facebook: can only make 100-150 internal requests per page

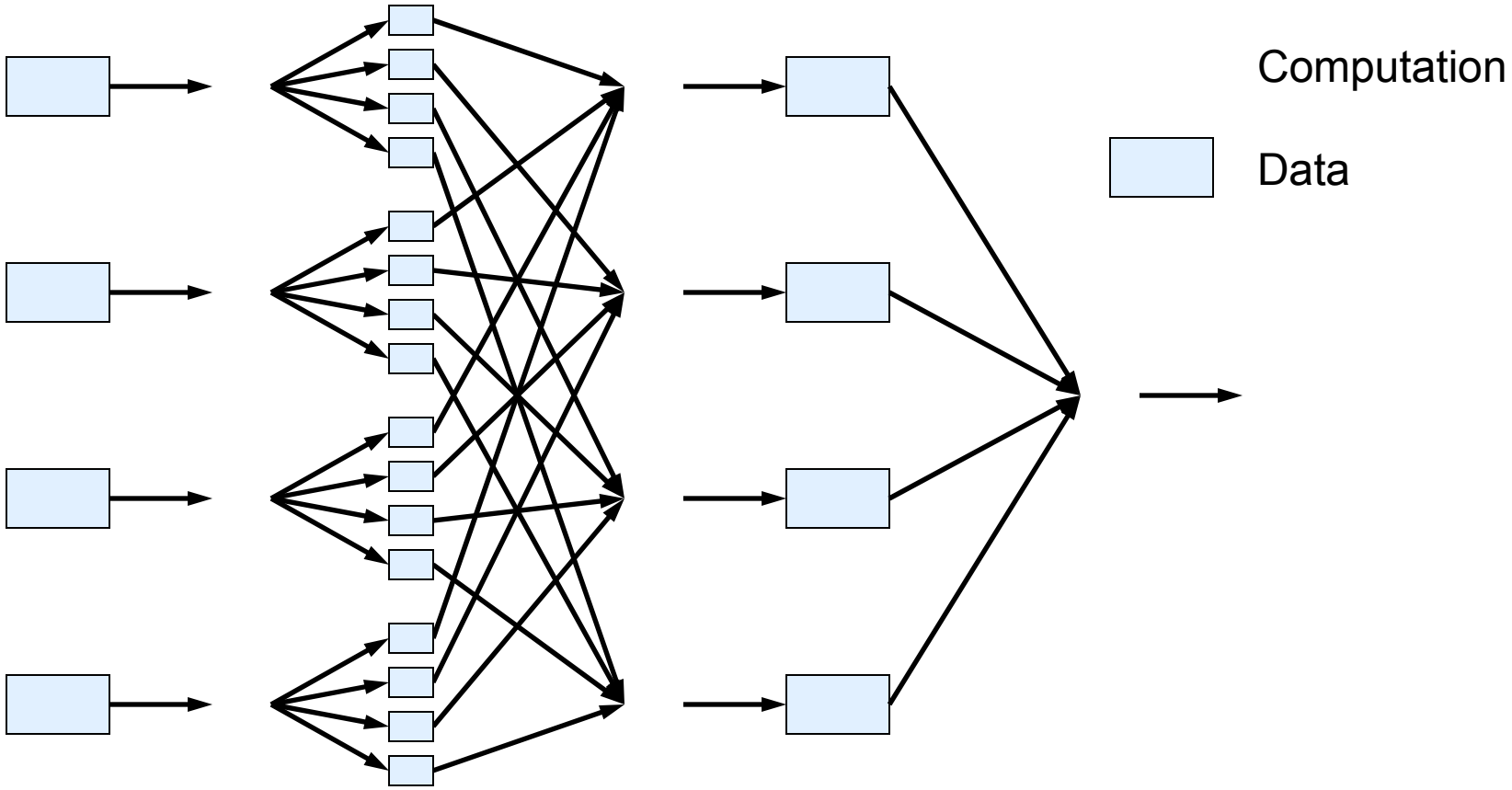
Dimensions of Scalability



Dimensions of Scalability



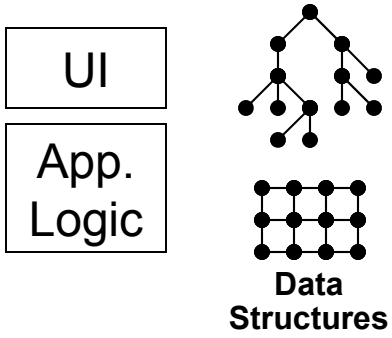
MapReduce



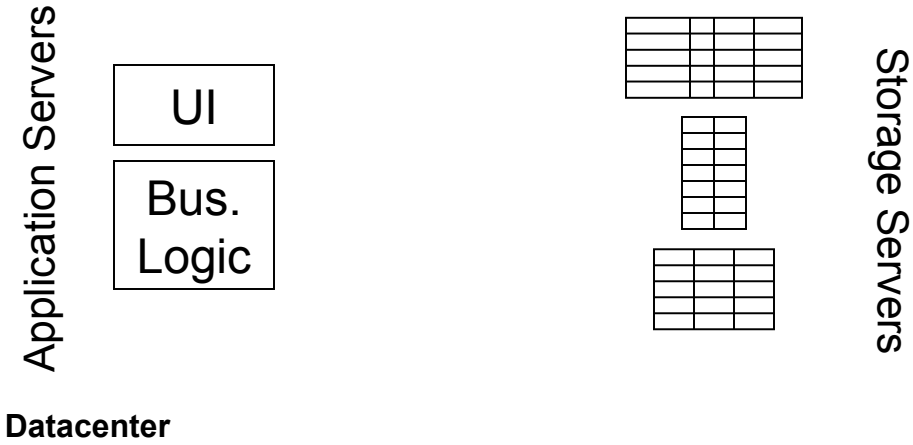
- ✓ **Sequential data access → high data access rate**
- ✘ **Not all applications fit this model**

RAMCloud Motivation: Latency

Traditional Application



Web Application



<< 1µs latency

~~0.5-10ms~~ latency
5-10µs

- **RAMCloud goal: large scale and low latency**
- **Enable a new breed of information-intensive applications**

RAMCloud Motivation: Scalability

- **Relational databases don't scale**
- **Every large-scale Web application has problems:**
  - Facebook: 4000 MySQL instances + 2000 memcached servers
- **Major system redesign for every 10x increase in scale**
- **New forms of storage appearing:**
  - Bigtable
  - Dynamo
  - PNUTS
  - Sinfonia
  - H-store
  - memcached

RAMCloud Motivation: Technology

Disk access rate not keeping up with capacity:

	Mid-1980's	2009	Change
Disk capacity	30 MB	500 GB	16667x
Max. transfer rate	2 MB/s	100 MB/s	50x
Latency (seek & rotate)	20 ms	10 ms	2x
Capacity/bandwidth (large blocks)	15 s	5000 s	333x
Capacity/bandwidth (1KB blocks)	600 s	58 days	8333x
Jim Gray's rule	5 min	30 hrs	360x

- **Disks must become more archival**
- **More information must move to memory**

Why Not a Caching Approach?

- **Lost performance:**
  - 1% misses → 10x performance degradation
  - Hard to approach 1% misses (Facebook ~ 5-7% misses)
- **Won't save much money:**
  - Already have to keep information in memory
  - Example: Facebook caches ~75% of data size
- **Changes disk management issues:**
  - Optimize for reads, vs. writes & recovery

Why not Flash Memory?

- **Many candidate technologies besides DRAM**
  - Flash (NAND, NOR)
  - PC RAM
  - …
- **DRAM enables lowest latency today:**
  - 5-10x faster than flash
- **Most RAMCloud techniques will apply to other technologies**

Is RAMCloud Capacity Sufficient?

- **Facebook: 200 TB of (non-image) data in 2009**
- **Amazon:**

Revenues/year:	$16B
Orders/year:	400M? ($40/order?)
Bytes/order:	1000-10000?
Order data/year:	0.4-4.0 TB?
RAMCloud cost:	$26-260K?

- **United Airlines:**

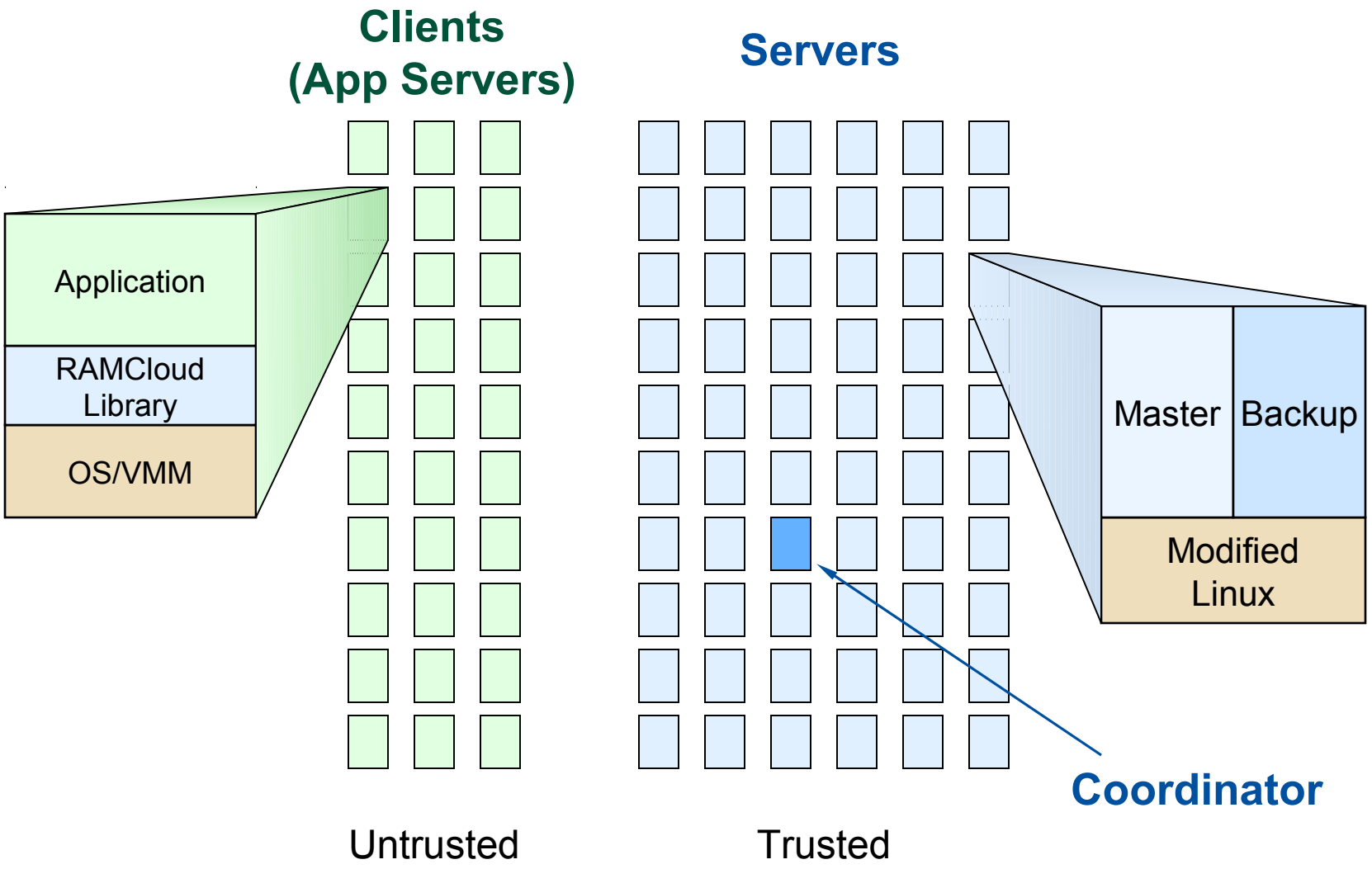
Total flights/day:	4000? (30,000 for all airlines in U.S.)
Passenger flights/year:	200M?
Bytes/passenger-flight:	1000-10000?
Order data/year:	0.2-2.0 TB?
RAMCloud cost:	$13-130K?

- **Ready today for almost all online data; media soon**

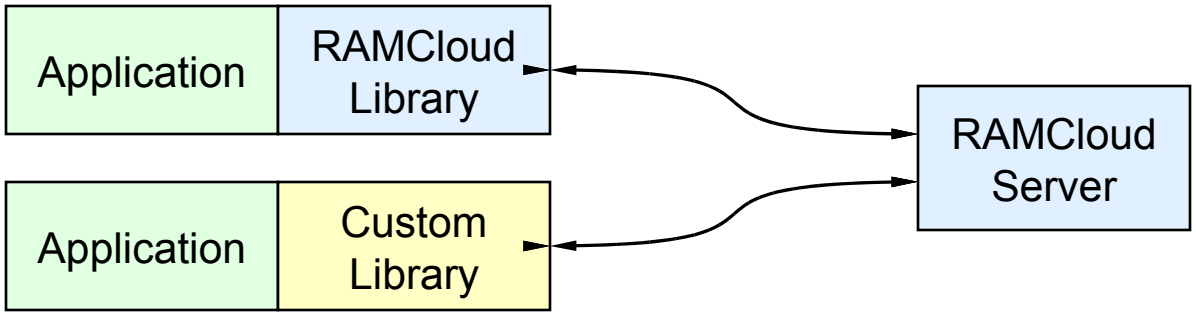
RAMCloud Research Issues

- **Data durability/availability**
- **Fast RPCs**
- **Data model, concurrency/consistency model**
- **Data distribution, scaling**
- **Automated management**
- **Multi-tenancy**
- **Client-server functional distribution**
- **Node architecture**

RAMCloud Cluster Structure



Client Library vs. Server



- **Move functionality to library?**
  - Flexibility: enable different implementations
  - Throughput: offload servers
  - May improve performance (e.g., aggregation)
- **Concentrate functionality in servers?**
  - May improve performance (e.g., faster synchronization)
  - Can't depend on proper client behavior:
    - Security/access control
    - Consistency/crash recovery

Data Model Rationale

Lower-level APIs
Less server functionality

Key-value store

Higher-level APIs
More server functionality

Distributed shared memory :

- ✓ **Server implementation easy**
- ✓ **Low-level performance good**
- ✘ **APIs not convenient for applications**
- ✘ **Lose performance in application-level synchronization**

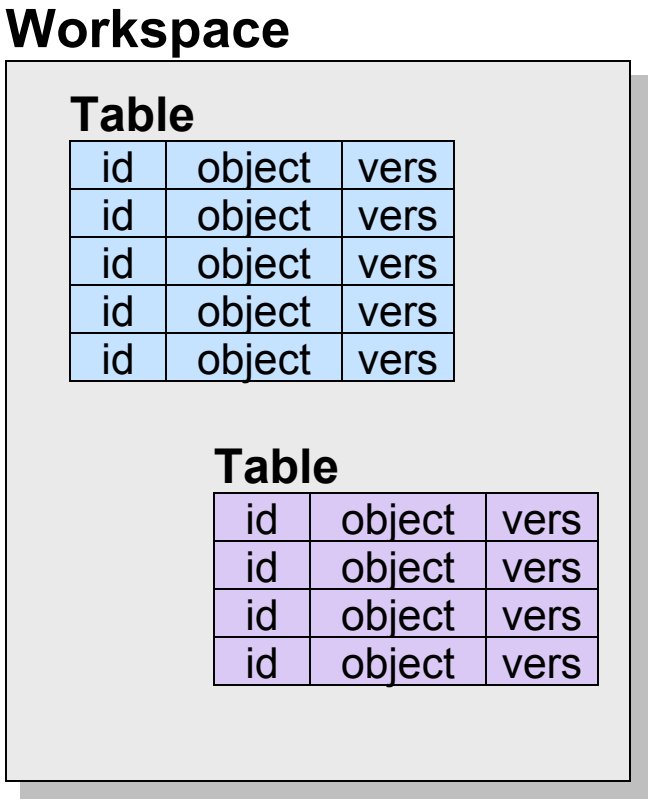
Relational database :

- ✓ **Powerful facilities for apps**
- ✓ **Best RDBMS performance**
- ✘ **Simple cases pay RDBMS performance**
- ✘ **More complexity in servers**

How to get best application-level performance?

Data Model Basics

- **Workspace:**
  - All data for one or more apps
  - Unit of access control
- **Table:**
  - Related collection of objects
- **Object:**
  - Variable-length up to 1MB
  - Contents opaque to servers
- **Id:**
  - 64 bits, unique within table
  - Chosen explicitly by client or implicitly by server (0,1,2,...)
- **Version:**
  - 64 bits
  - Guaranteed increasing, even across deletes



Basic Operations

```
get(tableId, objId)              → (blob, version)
put(tableId, blob)               → (objId, version)
put(tableId, objId, blob)        → (version)
delete(tableId, objId)
```

Other facilities (discussed in later talks)

- **Conditional updates**
- **Mini-transactions**
- **Indexes**

Other Design Goals

- **Data distributed automatically by RAMCloud:**
  - Tables can be split across multiple servers
  - Indexes can be split across multiple servers
  - Distribution transparent to applications
- **Multi-tenancy for cloud computing:**
  - Support multiple (potentially hostile) applications
  - Cost proportional to application size

Conclusion

- **Interesting combination of scale and latency**
- **Enable more powerful uses of information at scale:**
  - 1000-10000 clients
  - 100TB - 1PB
  - 5-10 µs latency

Questions/Comments