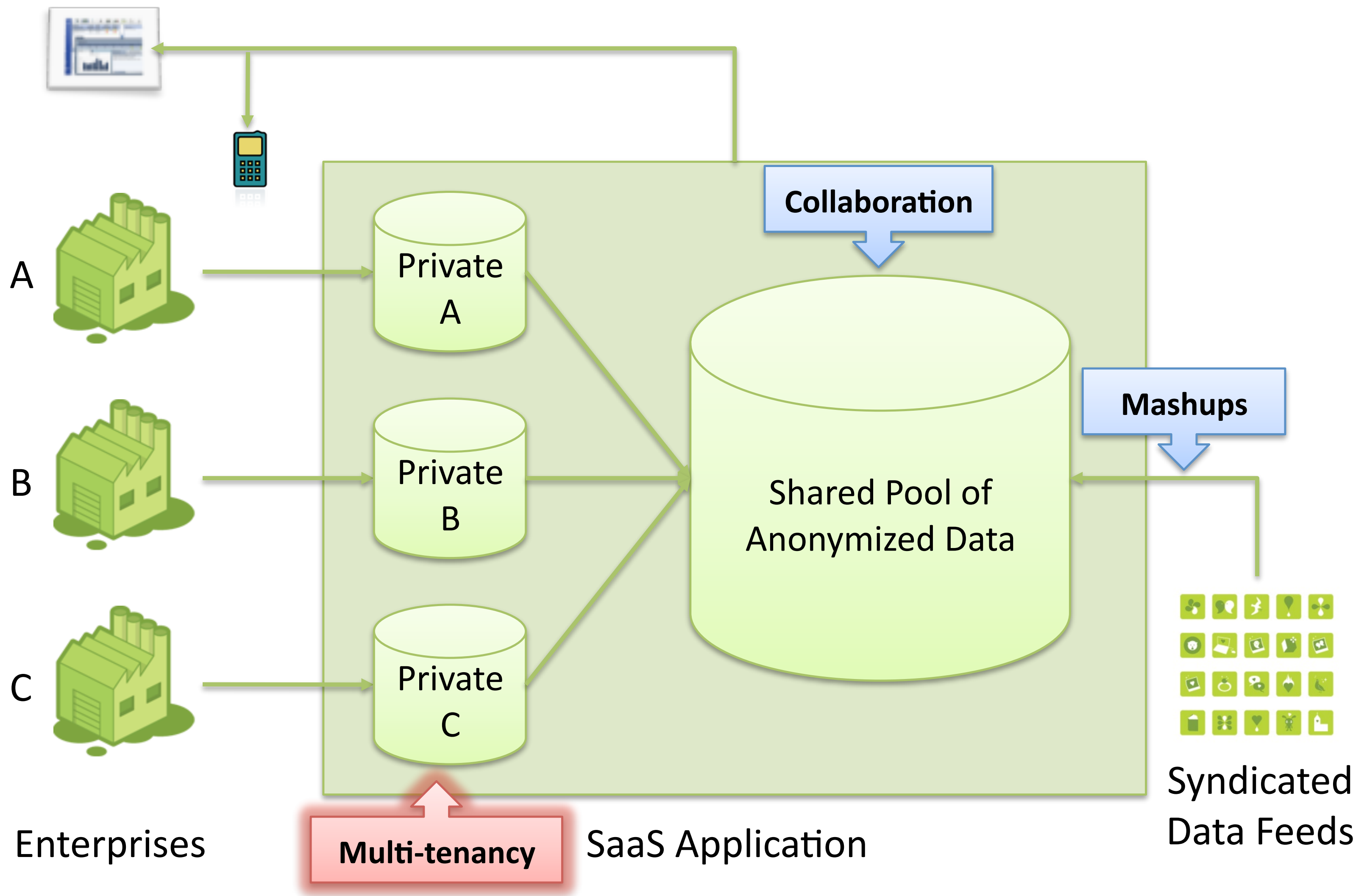
Memory-Based Cloud Architectures

(Or: Technical Challenges for OnDemand Business Software)

Jan Schaffner

Enterprise Platform and Integration Concepts Group

Example: Enterprise Benchmarking



Definition

Cloud Computing
=
Data Center + API

What to take home from this talk?

Answers to four questions:

- Why are memory based architectures great for cloud computing?
- How predictable is the behavior of an in-memory column database?
- Does virtualization have a negative impact on in-memory databases?
- How do I assign tenants to servers in order to manage fault-tolerance and scalability?

First question

Hasso Plattner Institut

Why are memory based architectures great for cloud computing?

Numbers everyone should know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns (in 2008)
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns (in 2008)
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA → Netherlands → CA	150,000,000 ns

Source: Jeff Dean

Memory should be the system of record

- Typically disks have been the system of record
  - Slow → wrap them in complicated caching and distributed file systems to make them perform
  - Memory used as cache all over the place but it can be invalidated when something changes on disk
- Bandwidth:
  - Disk: 120 MB/s/controller
  - DRAM (x86 + FSB): 10.4 GB/s/board
  - DRAM (Nehalem): 25.6 GB/s/socket
- Latency:
  - Disk: 13 milliseconds (up to seconds when queuing)
  - InfiniBand: 1-2 microseconds
  - DRAM: 5 nanoseconds

High-end networks vs. disks

Maximum bandwidths:

Hard Disk	100-120 MB/s
SSD	250 MB/s
Serial ATA II	600 MB/s
10 GB Ethernet	**1204 MB/s**
InfiniBand	**1250 MB/s (4 channels)**
PCIe Flash Storage	1400 MB/s
PCIe 3.0	32 GB/s
DDR3-1600	25.6 GB/s (dual channel)

Even more numbers...

	Type	Device / Medium	Latency	Throughput
Same core	Storage	L1 cache read (local)	1.3 ns	364.8 Gbps
	Storage	L2 cache read (local)	3.4 ns	248.8 Gbps
	Storage	L3 cache read (local)	13 ns	209.6 Gbps
Same die	Storage	L1 cache read (remote, same die)	13 - 28.3 ns	75.2 - 154.4 Gbps
	Storage	L2 cache read (remote, same die)	13 - 25.5 ns	105.6 - 157.6 Gbps
	Storage	L3 cache read (remote, same die)	13 - 22.2 ns	157.6 - 209.6 Gbps
Same board	Storage	L1 cache read (remote, via QPI)	58 - 109 ns	44.8 - 72 Gbps
	Storage	L2 cache read (remote, via QPI)	58 - 109 ns	44.8 - 73.6 Gbps
	Storage	L3 cache read (remote, via QPI)	58 - 109 ns	44.8 - 73.6 Gbps
Same machine	Storage	DRAM (Nehalem)	65 - 106 ns	160 - 256 Gbps / socket
	Interconnect	SATA 3.0	at least 1 µs	6 Gbps
	Interconnect	Serial Attached SCSI	at least 1 µs	6 Gbps
	Interconnect	PCI Express	3.8 - 5 µs	4 Gbps x number of lanes
	Storage	Magnetical disk read / write	3.2 - 13 ms	0.96 - 1.12 Gbps
	Storage	Solid State Disk read	65 µs	1.9 Gbps
Network	Interconnect	RDMA over InfiniBand	1 - 3 µs	2.5 - 10 Gbps x number of channels
	Interconnect	RDMA over iWARP	6 µs	10 Gbps / link
	Interconnect	10Gb Ethernet	20 µs	10 Gbps / link
	Interconnect	Fibre channel	3 - 10 µs (add 1 ms per 100 km)	8 Gbps / channel

Designing a database for the cloud

- Disks are the limiting factor in contemporary database systems
  - Sharing a high performance disk on a machine/cluster/cloud is fine/troublesome/miserable
  - While one guy is fetching 100 MB/s, everyone else is waiting

- **Claim:** Two machines + network is better than one machine + disk
  - Log to disk on a single node:
    > 10,000 μs (not predictable)
  - Transactions only in memory but on two nodes:
    < 600 μs (more predictable)

- Concept: Design to the strengths of cloud (redundancy) rather than their weaknesses (shared anything)

Design choices for a cloud database

- No disks (in-memory delta tables + async snapshots)
- Multi-master replication
  - Two copies of the data
  - Load balancing both reads and (monotonic) writes
  - (Eventual) consistency achieved via MVCC (+ Paxos, later)
- High-end hardware
  - Nehalem for high memory bandwidth
  - Fast interconnect
- Virtualization
  - Ease of deployment/administration
  - Consolidation/multi-tenancy

Why consolidation?

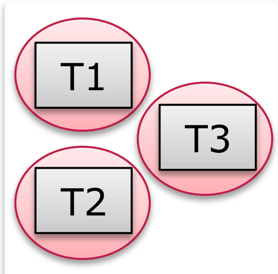
- In-memory column databases are ideal for mixed workload processing
- **But:** In a SaaS environment it seems costly to give everybody their private NewDB box

- How much consolidation is possible?
  - 3 years worth of sales records from our favorite Fortune 500 retail company
  - 360 million records
  - Less than 3 GB in compressed columns in memory
  - Next door is a machine with 2 TB of DRAM
  - (Beware of overhead)

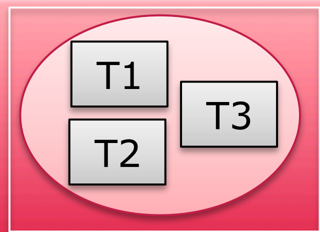
Multi-tenancy in the database – four different options

Hasso Plattner Institut

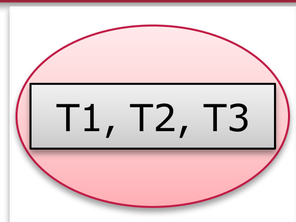
- **No multi-tenancy** – one VM per tenant
  - **Ex.:** RightNow has 3000 tenants in 200 databases (2007): 3000 vs. 200 Amazon VMs cost $2,628,000 vs. $175,200/year
  - Very strong isolation
- **Shared machine** – one database process per tenant
  - Scheduler, session manager and transaction manager need live inside the individual DB processes: IPC for synchronization
  - Good for custom extensions, good isolation



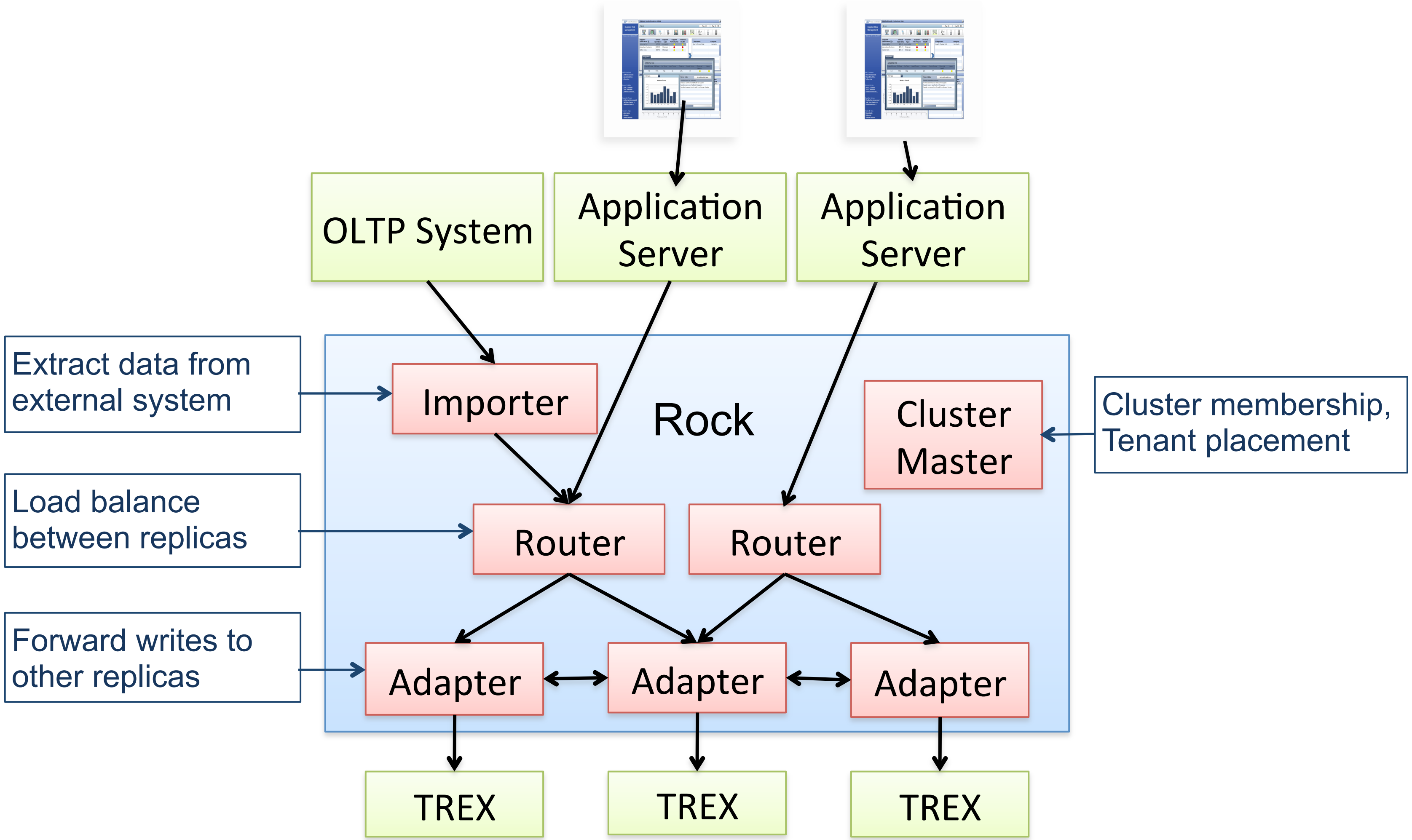
- **Shared instance** – one schema instance per tenant
  - Must support large numbers of tables
  - Must support online schema extension and evolution



- **Shared table** – use a `tenant_id` column and partitioning
  - Bad for custom extensions, bad isolation
  - Hard to backup/restore/migrate individual tenants



Putting it all together: Rock cluster architecture



Second question

How predictable is the behavior of an in-memory column database?

What does "predictable" mean?

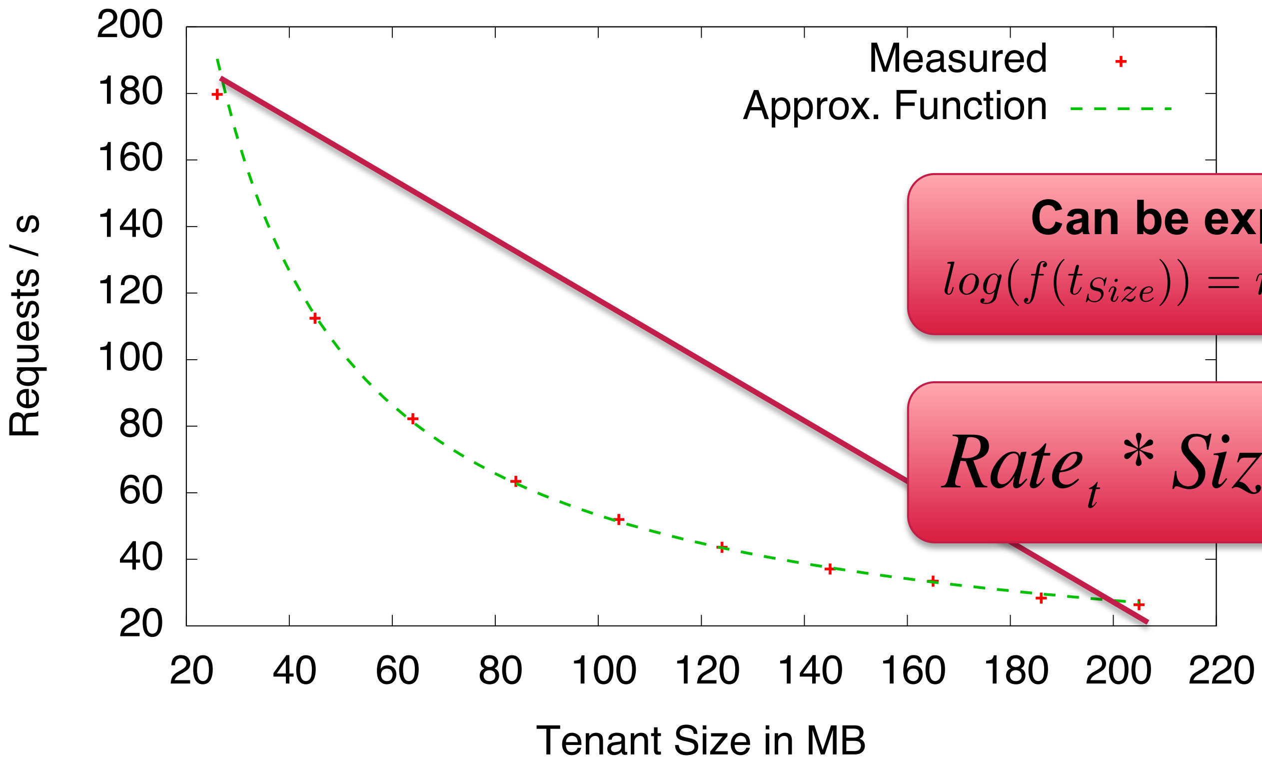
- Traditionally, database people are concerned with the questions of type "how do I make a query faster?"

- In a SaaS environment, the question is "how do I get a fixed (low) response time as cheap as possible?"
  - Look at throughput
  - Look at quantiles (e.g. 99-th percentile)

- Example formulation of desired performance:
  - Response time goal "1 second in the 99-th percentile"
  - Average response time around 200 ms
  - Less than 1% of all queries exceed 1,000 ms
  - Results in a maximum number of concurrent queries before response time goal is violated

System capacity

- Fixed amount of data split equally among all tenants



Can be expressed as:

$$log(f(t_{Size})) = m \cdot log(t_{Size}) + n$$

$$Rate_t * Size_t^{0.95} = 4144$$

- Capacity ≈ bytes scanned per second
  (there is a small overhead when processing more requests)
- In-memory databases behave very linearly!

Workload

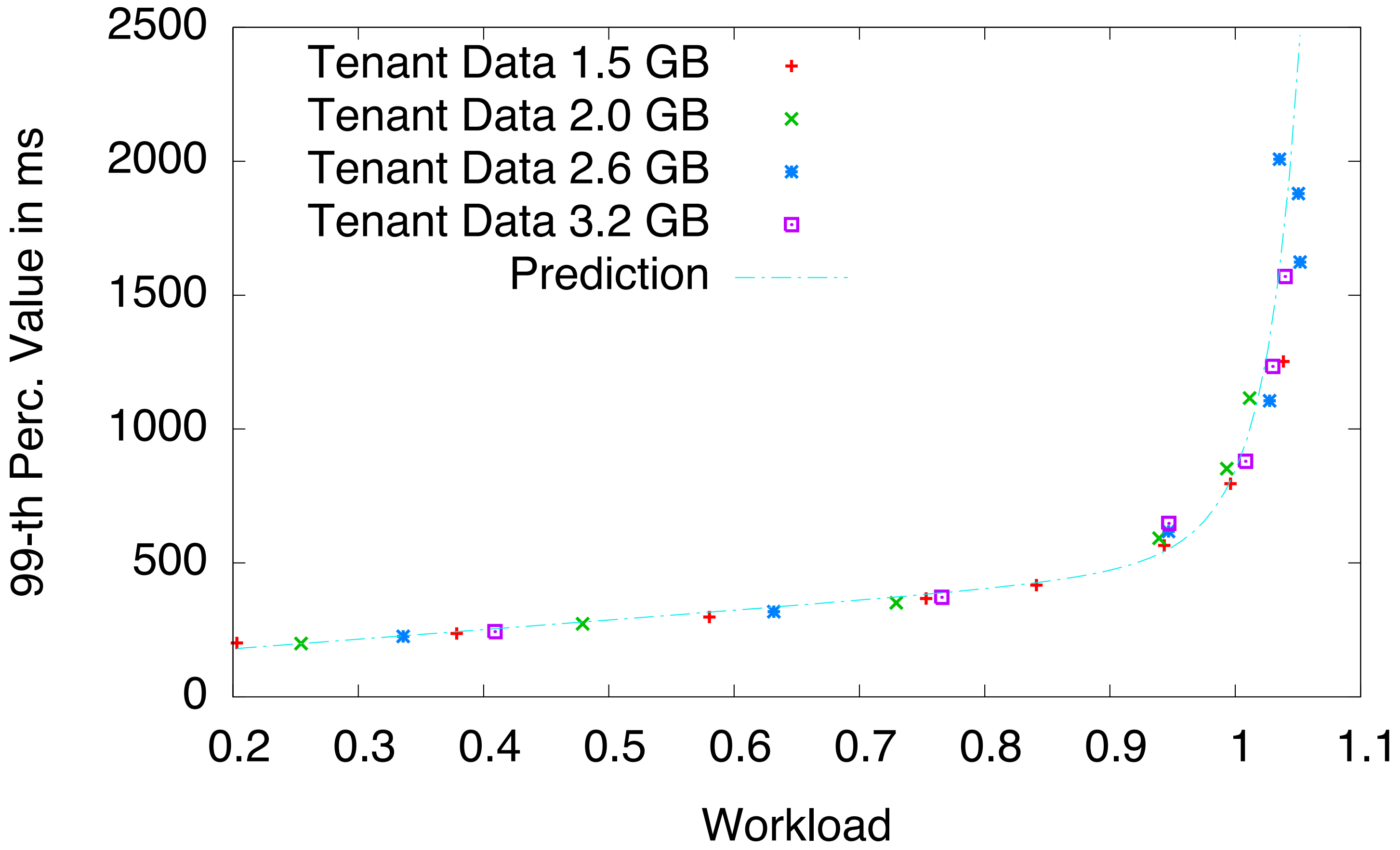
- Tenants generally have different rates and sizes
- For a given set of T tenants (on one server) define

$$Workload = \sum_{t \in T} \frac{Rate_t * Size_t^{0.95}}{4144}$$

- When Workload = 1
  - System runs at it's maximum throughput level
  - Further increase of workload will result in violation of response time goal

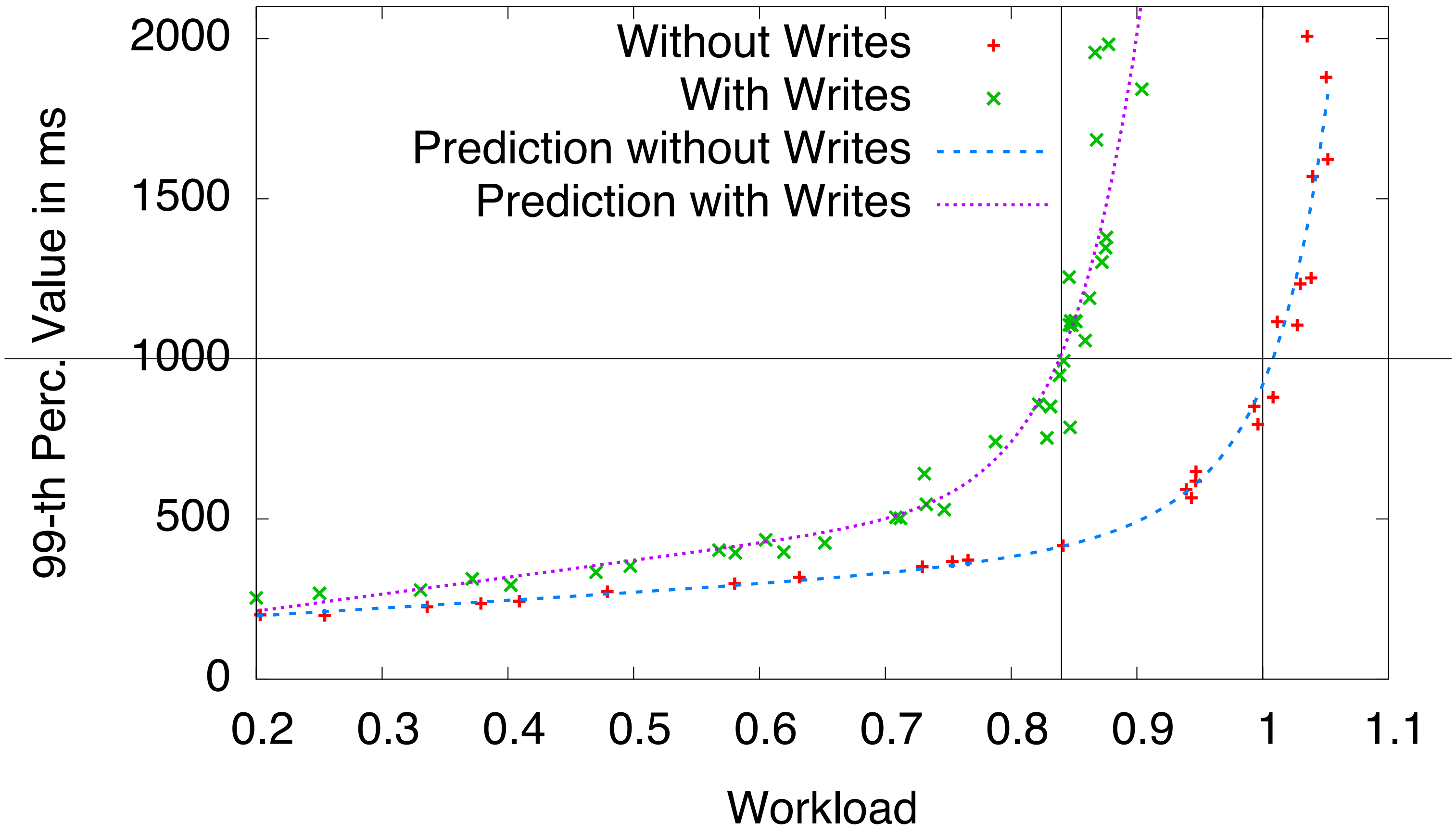
Response time

- Different amounts of data and different request rates ("assorted mix")
- Workload is varied by scaling the request rates



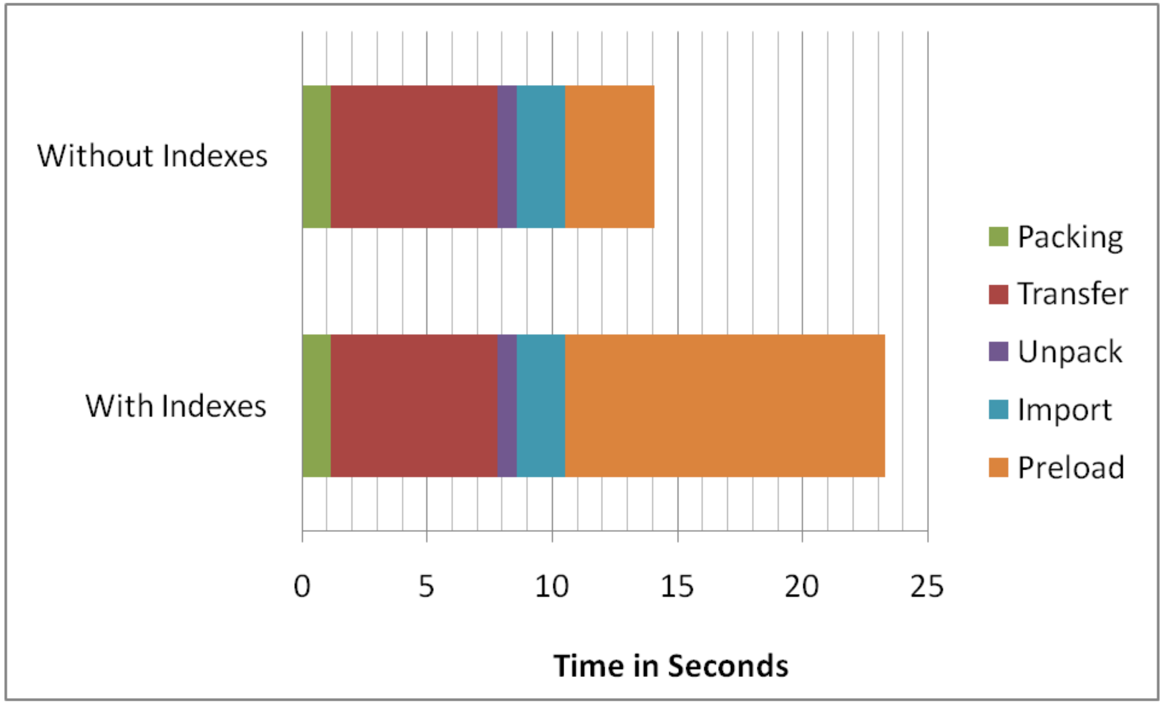
Impact of writes

- Added periodic batch writes (fact table grows by 0.5% every 5 minutes)



Why is predictability good?

- Ability to plan and perform resource intensive tasks during normal operations:
  - Upgrades
  - Merges
  - Migrations of tenants in the cluster (e.g. to dynamically re-balance the load situation in the cluster)



Cost breakdown for migration of tenants

Definition

Cloud Computing

=

Data Center + API

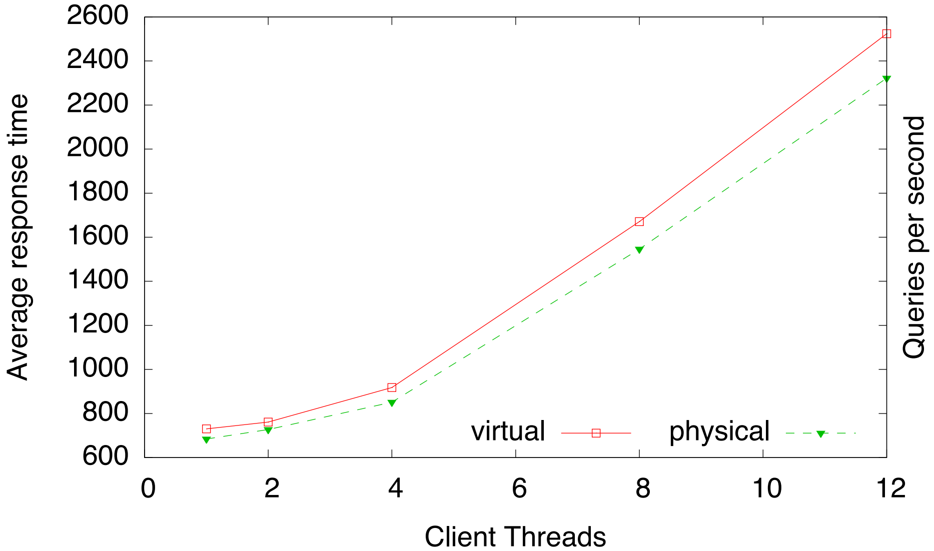
Third question

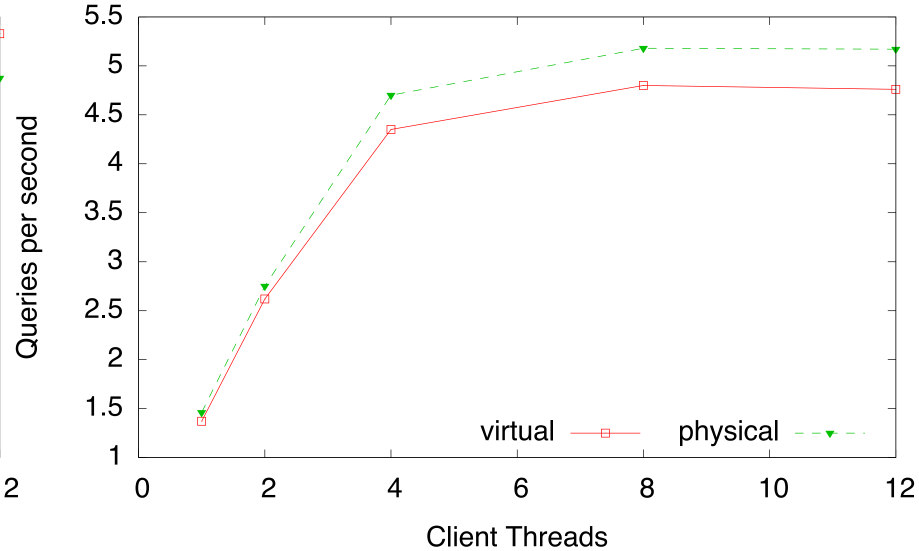
Does virtualization have a negative impact on in-memory databases?

Impact of virtualization

- Run multi-tenant OLAP benchmark on either:
  - one TREX instance directly on the physical host vs.
  - one TREX instance inside VM on the physical host

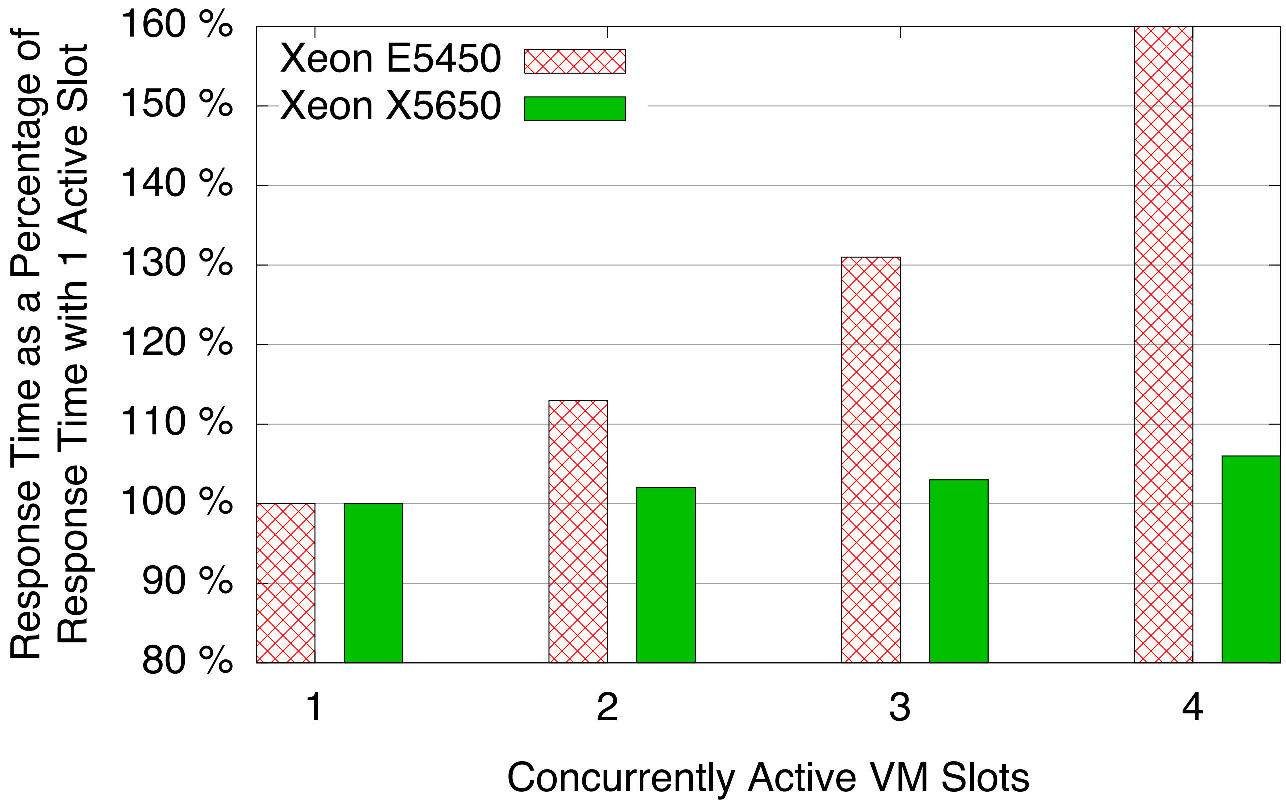
- Overhead is approximately 7% (both in response time and throughput)





Impact of virtualization (contd.)

- Virtualization is often used to get "better" system utilization
  - What happens when a physical machine is split into multiple VMs?
  - Burning CPU cycles does not hurt → memory bandwidth is the limiting factor



Fourth question

How do I assign tenants to servers in order to manage fault-tolerance and scalability?

Why is it good to have multiple copies of the data?

- Scalability beyond a certain number of concurrently active users
- High availability during normal operations
- Alternating execution of resource-intensive operations (e.g. merge)
- Rolling upgrades without downtime
- Data migration without downtime

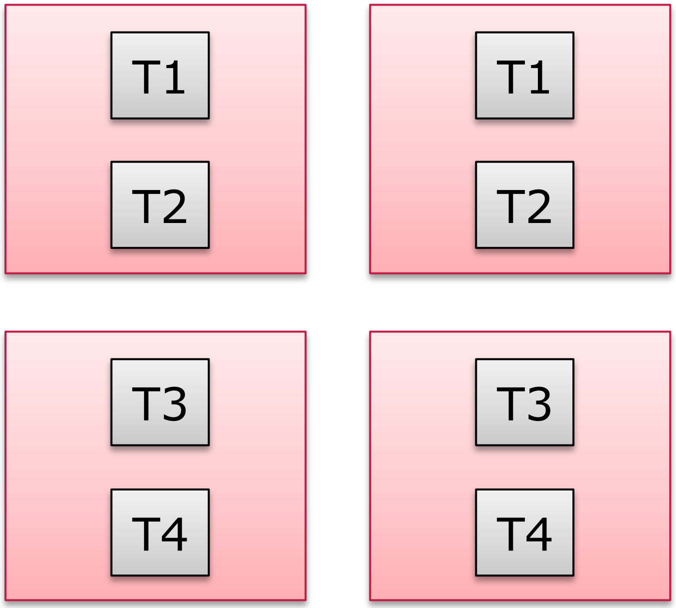
- *Reminder:* Two in-memory copies allow faster writes and are more predictable than one in-memory copy plus disk

- **Downsides:**
  - Response time goal might be violated during recovery
  - You need to plan for twice the capacity

Really?

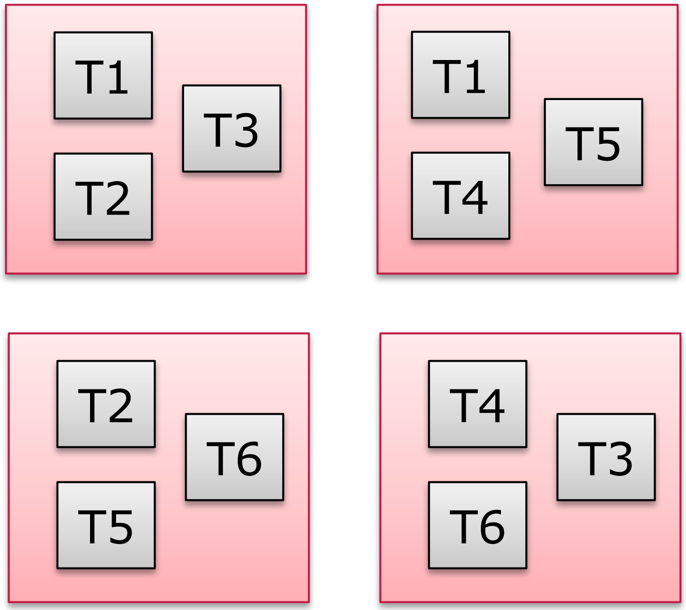
Tenant placement

Conventional
Mirrored Layout



If a node fails, all work moves to one other node. The system must be **100% over-provisioned**.

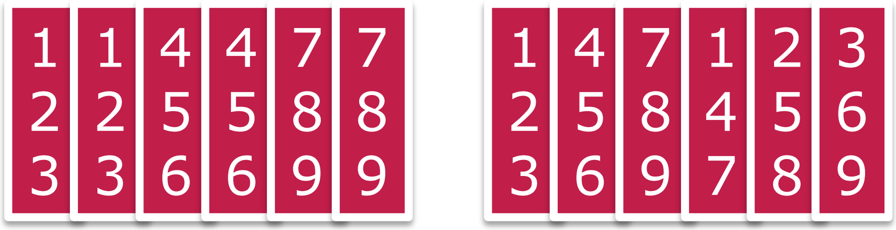
Interleaved
Layout



If a node fails, work moves to many other nodes. Allows **higher utilization** of nodes.

Handcrafted best case

- Perfect placement:
  - 100 tenants
  - 2 copies/tenant
  - All tenants have same size
  - 10 tenants/server



Mirrored Interleaved

- Perfect balancing (same load on all tenants):
  - 6M rows (204 MB compressed) of data per tenant
  - The same (increasing) number of users per tenant
  - No writes

	Mirrored	Interleaved	Improvement
No failures	4218 users	4506 users	7%
Periodic single failures	2265 users	4250 users	88%

Throughput before violating response time goal

Requirements for placement algorithm

- An optimal placement algorithm needs to cope with multiple (conflicting) goals:
  - Balance load across servers
  - Achieve good interleaving

- Use migrations consciously for online layout improvements (no big bang cluster re-organization)

- Take usage patterns into account
  - Request rates double during last week before end of quarter
  - Time-zones, Christmas, etc.

Conclusion

- Answers to four questions:
  - ✓ Why are memory based architectures great for cloud computing?
  - ✓ How predictable is the behavior of an in-memory column database?
  - ✓ Does virtualization have a negative impact on in-memory databases?
  - ✓ How do I assign tenants to servers in order to manage fault-tolerance and scalability?

- **Questions?**