Engenharia de Sistemas de Computação
Performance Optimisation

Class 4 – Analyzing complex systems

Vítor Oliveira (vitor.s.p.oliveira@gmail.com)

May 11, 2021

1 Analyzing complex systems

Analyzing the performance of large applications is complex, as they are usually composed of many sub-components that impact the behavior of the system.

Analyzing complex systems

Large and complex applications are difficult to analyze

- Large application usually have many moving parts, with a large number of sub-systems that can compete for resources;
- The code base is usually large (in the several million lines of code), with multiple independently-developed modules, which makes knowing how it works hard;
- These applications are usually designed to take over the complete underlying machine, and in many cases multiple machines, and may require special tuning in deployment;
- They have a large impact on the local computing resources, but also on the network, storage systems, etc.
- And they are impacted by other application in the same infrastructure, which may cause fluctuations and behaviour changes.

Analyzing complex systems

Measuring the performance is difficult

- Artificial benchmark workloads can be used to measure the performance of a system but they must be meaningful and represent the particular features that are important in a particular context, in a hardware configuration the matches what is to be used in the real system;
- But the results of artificial benchmarks is a poor replacement for proper testing with actual workloads in real production environment;
- And the results in a particular benchmark is no assurance that other workloads will behave the same, which means that exhaustive testing is important, so it’s also time/money consuming.

Analyzing complex systems

Measuring the performance in production is very difficult

- Benchmarking a system reduces the throughput available for other workloads, so measuring performance in production must be done with a lot of care;
- The running workloads also affect the benchmarks, so the results are less reliable than would be if the system was dedicated to the benchmark alone;
- Testing can be done in dedicated resources in a common infrastructure, but replicating the workload may be hard, so actual user requests must be used;
- But in that case user-data may be lost if the system under test is not yet stable enough;
- It is also not easy to divert the running workloads to a test system without impacting other parts of a large mesh of services;

Analyzing complex systems

Optimizing the performance is difficult

- Optimizations in one part of the code may be crippled by other parts of the code;
  - Work may need to be done in several areas before the benefits of each improvements is seen
- Improvements in some use-cases may degrade other use-cases, which is a problem for applications that are used in many contexts;
- The interference between sub-systems may trigger behavioral changes that may seem improvement, but are circumstantial and not real improvements

Analyzing complex systems

Profiling a large system is difficult

- It is hard to know who is doing what then there are many threads with different goals running in the same process;
- Matching external operations such as system calls to different internals tasks is challenging;

Debugging a large system is difficult

- High-levels of concurrency can introduce non-deterministic behaviour that can make bugs hard to reproduce;
- Operations using large code paths are hard to follow;

2 Analyzing complex systems: hands-on

The goal of this assignment is to analyze a large application and to gather relevant information about it, the underlying system, during class.

Analyzing complex systems: hands-on

Goals:

- Analyze a test system that is running three instances of a MySQL database, connected with three different storage devices: memory, local SSD and network storage;
- Connect to the system using:

  ssh esc0@search7.di.uminho.pt (diEscUM21)

  ssh compute-641-10
- Part 1:
  - Execute the sysbench benchmarks on each of the MySQL instances and check the behavior of each instance;
- Part 2:
  - Profile the running system and gather the information needed.

Analyzing complex systems: hands-on

Execute sysbench:

```
/usr/local/share/sysbench/oltp_read_write.lua
--mysql-host=127.0.0.1 --mysql-user=root --mysql-password=rmysql
--report-interval=1  --percentile=99 --histogram --time=60
--table-size=1000000 --tables=8 --threads=1
--mysql-port=33071 or
--mysql-port=33072 or
--mysql-port=33073
run
```

Analyzing complex systems: hands-on

Investigate:

- How many transactions per second, and what latency, is mysql able to achieve in RAM (33071), local disk (33072) and network disk (33073)?
- Repeat for different benchmarks: oltp_read_only.lua and oltp_update_index
- What is the performance per thread obtained by varying the number of threads: 1 8 64 512 threads

/usr/local/share/sysbench/<test.lua> --mysql-host=127.0.0.1 --mysql-port=<server> --mysql-user=root --mysql-password=rmysql --table-size=1000000 --tables=8 --report-interval=1 --percentile=99 --histogram --time=120 --threads=1 run

Analyzing complex systems: hands-on

Investigate:

- What is the hardware configuration of the underlying server?
- What files are being used by MySQL?
- What is the storage access pattern?
- How much of the data is actually read and written to disk?
- What is the actual disk latency?

Analyzing complex systems: hands-on

Investigate:

- Which functions are the main cpu consumers?
- What are the most commonly used system calls?

3 Capacity planning

In the previous classes we discussed using benchmarking to help planning the necessary capacity of a system.

What follows highlights some examples on how that is used.

Capacity planning

- Capacity planning tries to determine the capacity required by a system to meet the changing workload demand, minimizing underutilized resources and unsatisfied stakeholders;

- Insufficient capacity can lead to the loss of customers, while excess capacity can drain the company's resources and limit future investments;

- Capacity planning can be performed prior to system deployment, but it is also a long-term, cyclical activity.

Capacity planning

- Capacity planning builds on the architecture of the system, and the performance of the applications, to create a plan for the infrastructure sizing;

- Software vendors sometimes provide guidelines regarding the sizing for the infrastructure for their applications: https://www.microsoft.com/en-us/download/details.aspx?id=102123

- And some hardware venders provide server sizing tools that use their hardware: https://www.dell.com/solutions/systembuilder/us/en/g_5/SQLServer2012

Cloud Infrastructures

Cloud Infrastructures are large-scale distributed infrastructures that provide remote compute resources to clients. While they share many characteristics of other large distributed systems, most of them share a few special characteristics of relevance.

Cloud Infrastructures

What is special about Cloud Infrastructures:

- Globally distributed resources;
- Multi-tenant, shared resources;
- Pay for the resources used only;
- A large set of readily-available distributed services;
- A multitude of computing shapes;
- …

Cloud Infrastructures

Globally distributed resources:

- Datacenters distributed in many regions, typically up to 50 regions globally;
- Each region is usually composed of multiple datacenters, physically separated as independent availability domains, or zones, to avoid single-points of failure;
- Each availability domain can have multiple, more dependent, faults domains to further separate the resources for availability and management ease;
- Services may be global, regional or ad-local, with different latency implications;
- The latency between regions can range from 10ms to 250ms, with intercontinental links over 70ms+,.

Cloud Infrastructures

Multi-tenant:

- Infrastructures are shared by many different costumers, which may be a problem for clients that expect predictable behavior from the infrastructure, when there are noisy neighbors in the same physical hosts or network;
- This is also a problem for security, in particular related to information leakage between VMs running in the same physical host;
- To avoid this problems some vendors allow the user to reserve full machines for their VMs, allocating the VMs dynamically only in the physical hosts reserved;
- Nevertheless, it can be expected that the infrastructure can have significant fluctuation over time due to a variety of reasons.

Cloud Infrastructures

A myriad of services:

- Cloud service providers provide many distributed services without additional installation/operating effort for the clients that use them;
- There services can be very elaborate, providing unique features whose development can only be justified at very large scales;
- Different vendors try to distinguish from the competition by providing different services, or at least different service implementations;

Cloud Infrastructures

Pay for what you use:

- Cloud costumers are billed according to resources requested from the infrastructure, with a small granularity;
- That means that resources are not acquired in advance and using a full server each time, which means that reducing a single core required for an application converts to savings in the final bill;
- This is different from what used to happen on-premises, where a full server was always acquired and using it fully was not a necessity;
- It also means that applications that optimize for a large number of cores may find that the use of more cloud VMs with less cores is more effective.

Cloud Infrastructures

All kinds of shapes:

- The VMs provided by cloud vendors runs in specific hardware, aside from having some number of cpu cores, memory, storage and network resources;
- Over time the hardware changes, in order to match the correct hardware the VMs run in specific shapes and shape generations;
- In addition to the common general purpose VMs, some venders also provide bare-metal shapes that provide a full node without an hypervisor in the middle, even allowing the user to install his own hypervisors such as VMWare;
- In additional, there are shapes with special hardware, such as local storage, GPUs, RoCE networks, etc.

Cloud Infrastructures

Some of the services:

- Computing
  - Bare-metal, VMs, containers
- Networking
  - SDN, isolation, security, load-balancing
- Storage
  - Block, Object, File, Backup
- Services
  - Identity
  - Databases
  - Workflow
  - Monitoring
  - …

Cloud Infrastructures

Control-plane and Data-plane

- Cloud services are usually deployed in a split infrastructure:
  - Data-plane interfaces with the user;
  - Control-plane sets up and manages the data-plane instances;

- The infrastructure is asynchronous:
  - Synchronicity in large networks, which always have failures in some part or another, is a bad idead;
  - User requests are translated to asynchronous jobs that follow the workflow that the control plane stipulated to execute the tasks

5 Performance in the Cloud (MySQL focused)

While large SMP machines with fast storage were making us forget the importance of latency, here comes the Cloud to remember that.

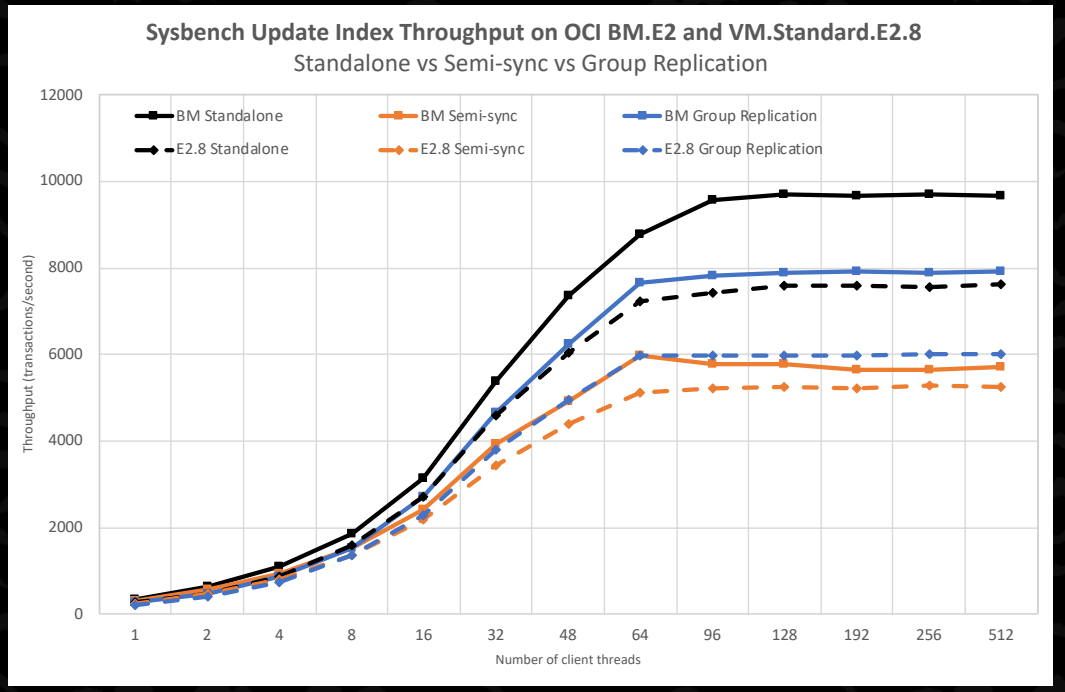
Benchmarking replication

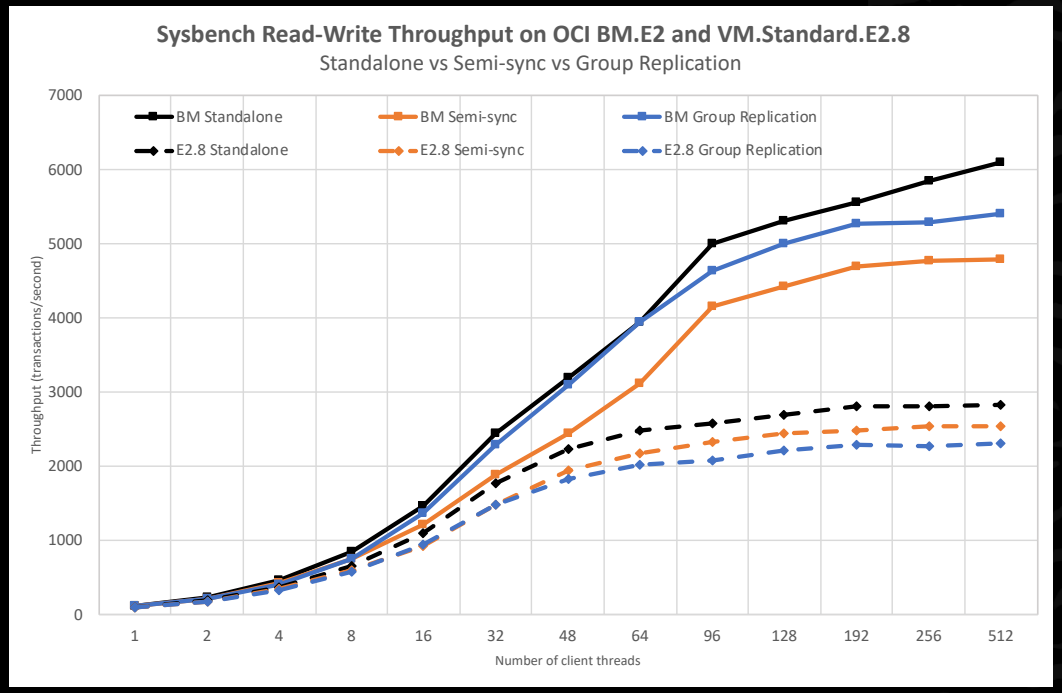
- We will start by presenting a performance evaluation of the MySQL Replication technologies running on the OCI, varying the shape type and some configuration parameters.

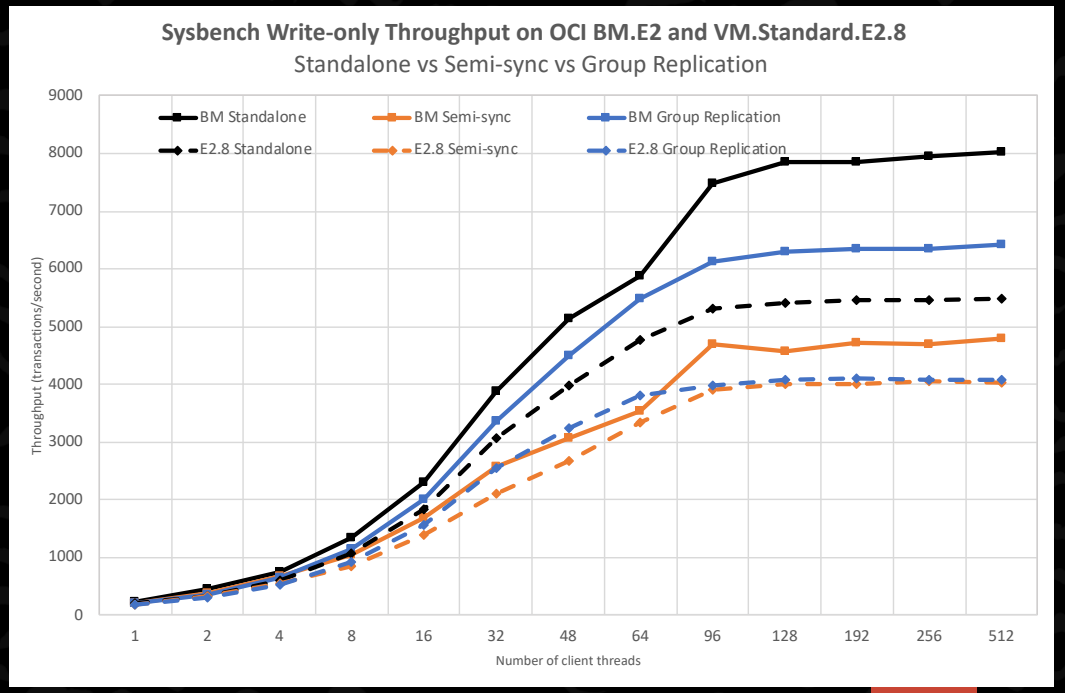
- The following testing setup was built:
  - Topologies with 3 servers on OCI, one in each AD, and a single client machine in the same AD as the primary server;
  - The benchmark selected was the usual Sysbench, in particular OLTP RW, OLTP Write-only and Update Index;
  - The machines selected where both a set of three OCI's BM.Standard.E2.64 (64 core, 128 HW threads), and a set of VM.Standard.E2.8 (8 core, 128 HW threads).

Benchmarking replication
Overview of replication performance

- Comparison between a single server and three servers in different ADs, using semi-sync and group replication.
- Two different type of servers: 1 VM with 8 cores and a full bare metal machine with 64 cores.

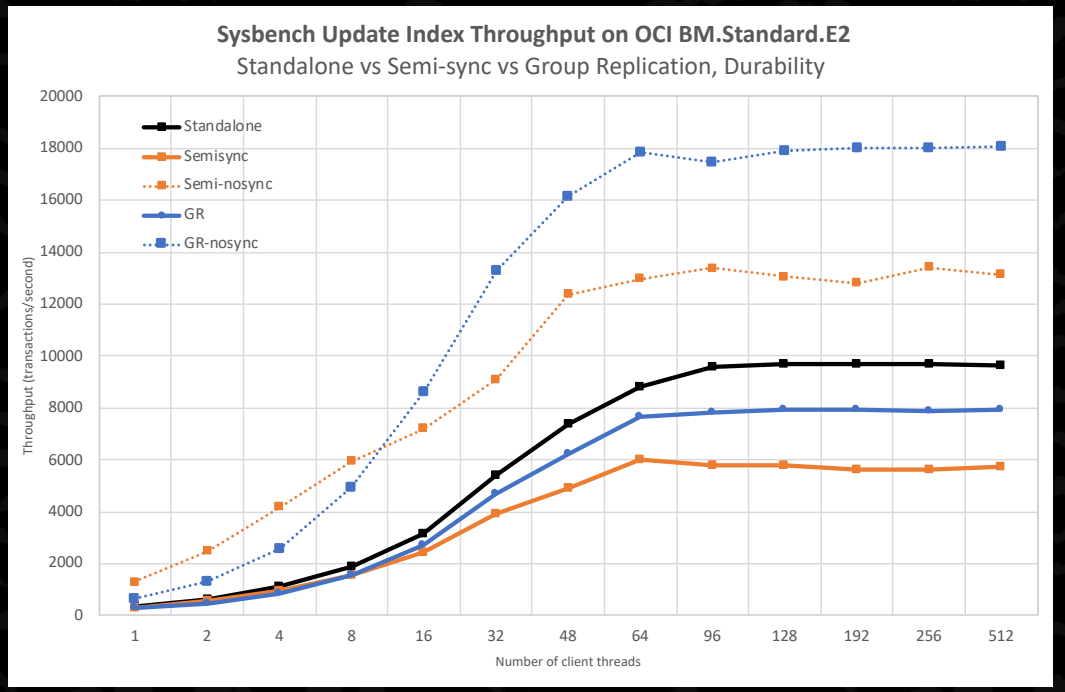


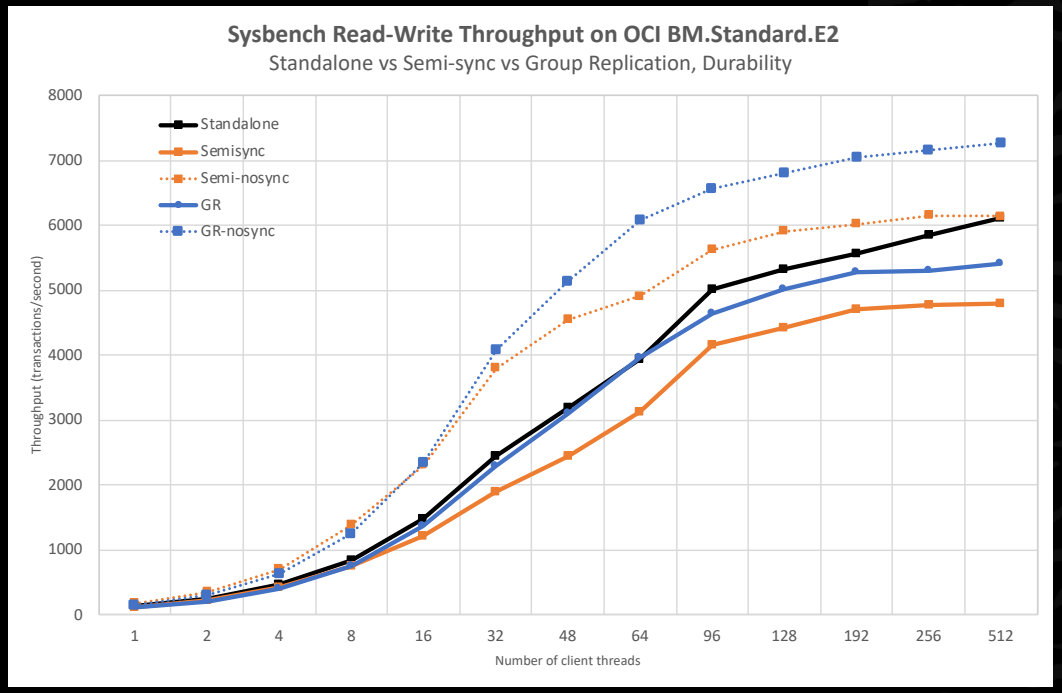


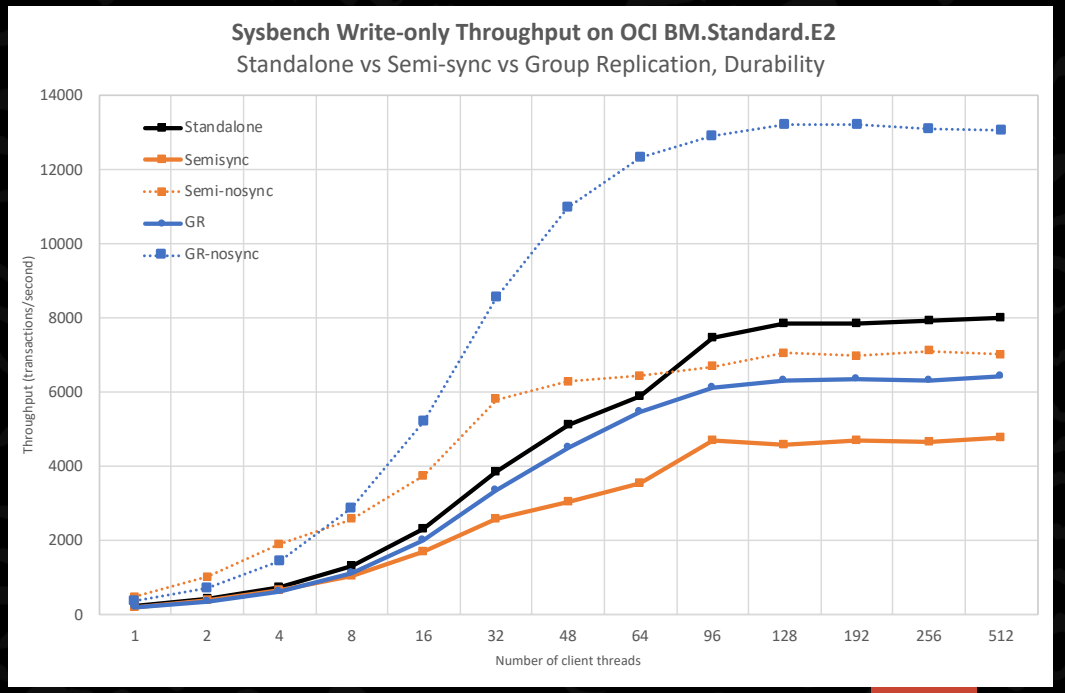


Benchmarking replication
Impact of durability on throughput

- Impact of relaxing the durability on the disks, transferring it to the network.
- The gain is significant, as the distributed storage system is more complex and with a higher latency than the Paxos from Group Replication.







Benchmarking replication
What do we gain with these performance evaluation results?

- There benchmarks are representative of a particular workload in a particular testing scenario.

- When evaluating such benchmarks many questions pop up:
  - What explains those results?
  - Can I expect the same behavior for my workload in the same setup?
  - How does the performance compare other cloud providers?
  - How does the performance compare to on-premises deployments?
  - Would other parameters be more effective for a particular purpose?
  - Were the measurements properly taken?

  - **What can I reliably estimate using these results?**

Can we estimate the behavior?

- The proper way to benchmark a system is to deploy it in a real system, and apply to it the actual user workload, even if that is hard to do in many cases.

- Benchmarks can be use capture three different things (often hard to distinguish):
  1. The performance of the software implementation of certain features tested by a benchmark in a particular way;
  2. The noise in the environment and other test setup limitations;
  3. The fundamental architectural and configuration details of the computing system used in the tests.

- Wouldn't it be great if we had a model that represents different deployment scenarios and calculate a rough estimate on the throughput to expect?

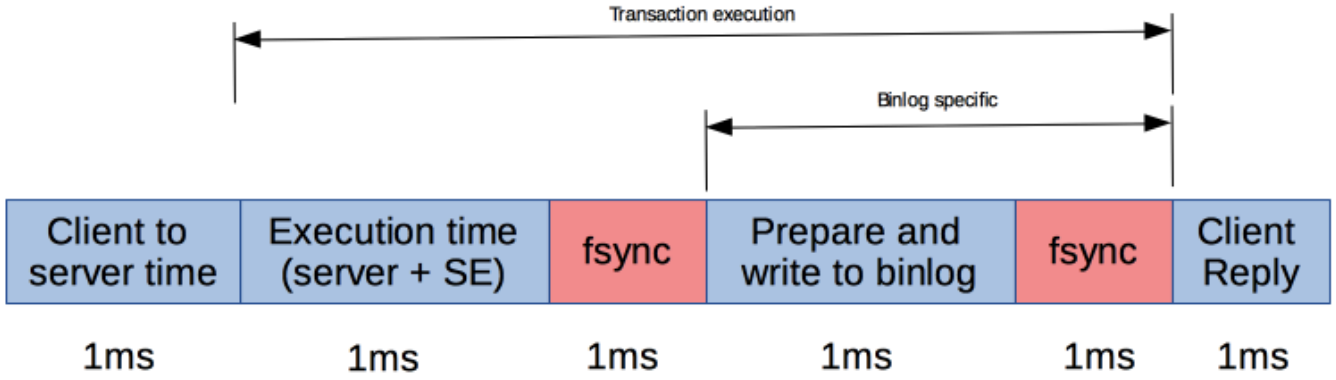
Can we estimate the behavior?
Building a simple model

- While the software implementation is hard to model, the fundamental aspects of the computing systems that are used are not, at least in a very simplified way.

- This model can be build on information such as:
  - The architecture of MySQL and how it uses the computing resources;
  - The characteristics of the workload: number of queries on the workload, size of the active dataset compared to the memory size;
  - And the computing system characteristics: network latency and bandwidth, computing resources, number of network jumps.

Can we estimate the behavior? **Building a simple model**

- On the right is a really simple model for the execution of a transaction.
- It demonstrates the goal, but we need more details…



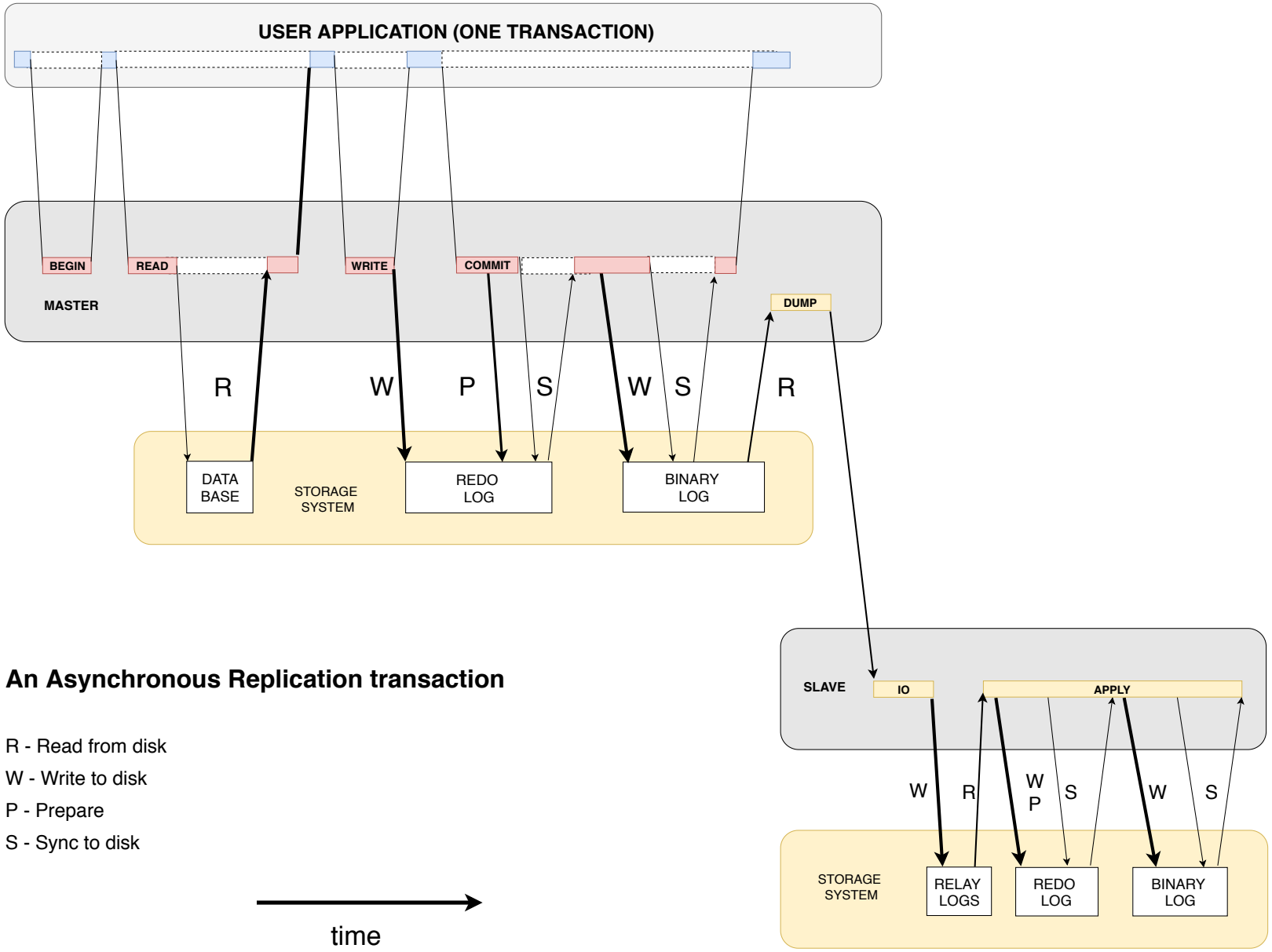
Using 1ms as an example, in each step we have:

Max TPS without binlog = 1000 / 4 = 250

Max TPS with binlog = 1000 / 6 = 166
Max TPS with perfect binlog = 100 / 5 = 200

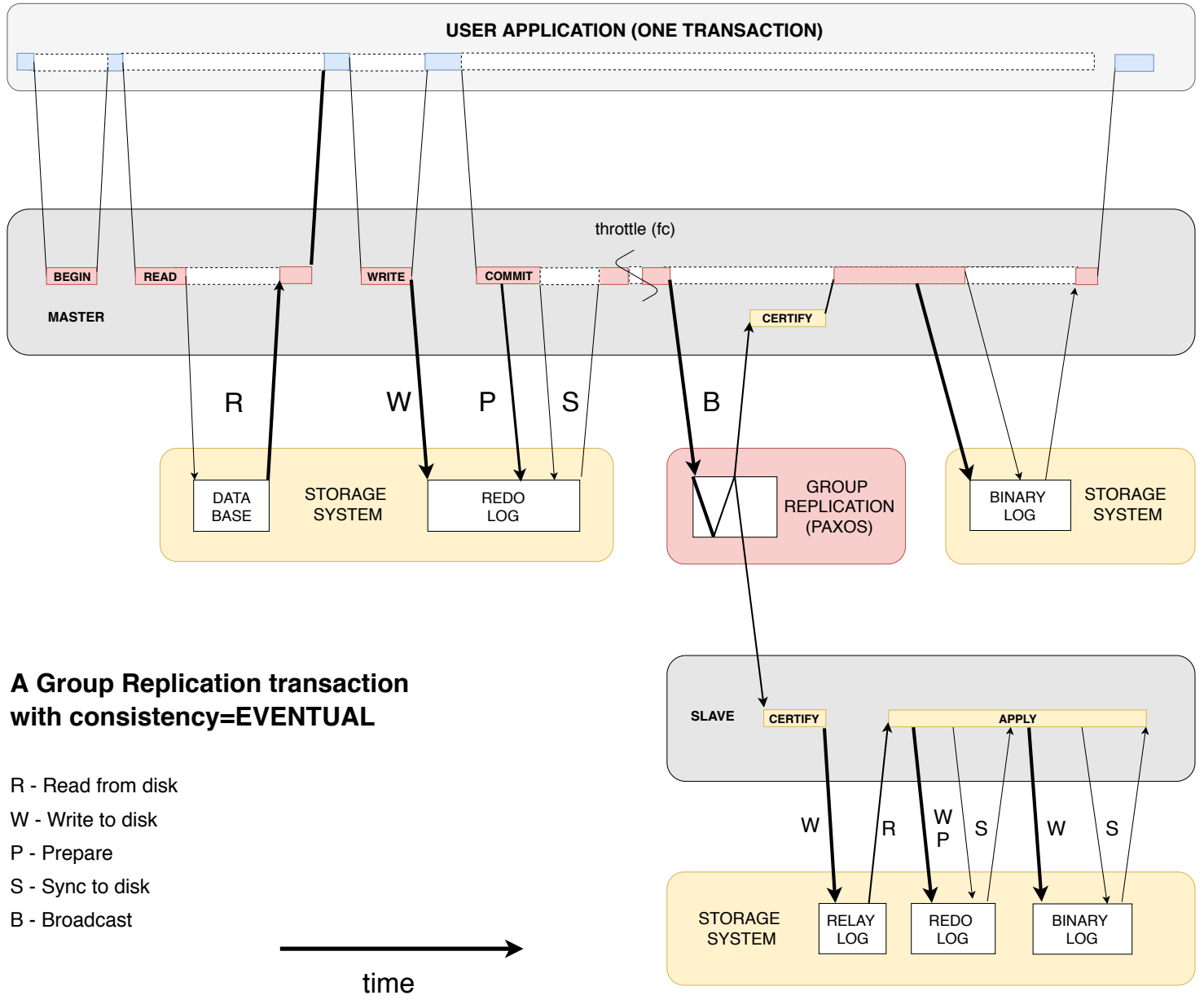
MySQL
Asynchronous Replication

- Lifecycle of a transaction:
  - Execute the queries as they are sent, read any data from the database that isn't already in the buffer pool, write the changes to the redo log, prepare the transaction, **fsync** (phase 1);
  - Write the full transaction to the binary log, **fsync** (phase 2);
  - Send the binary log to the slave and apply the same changes there.



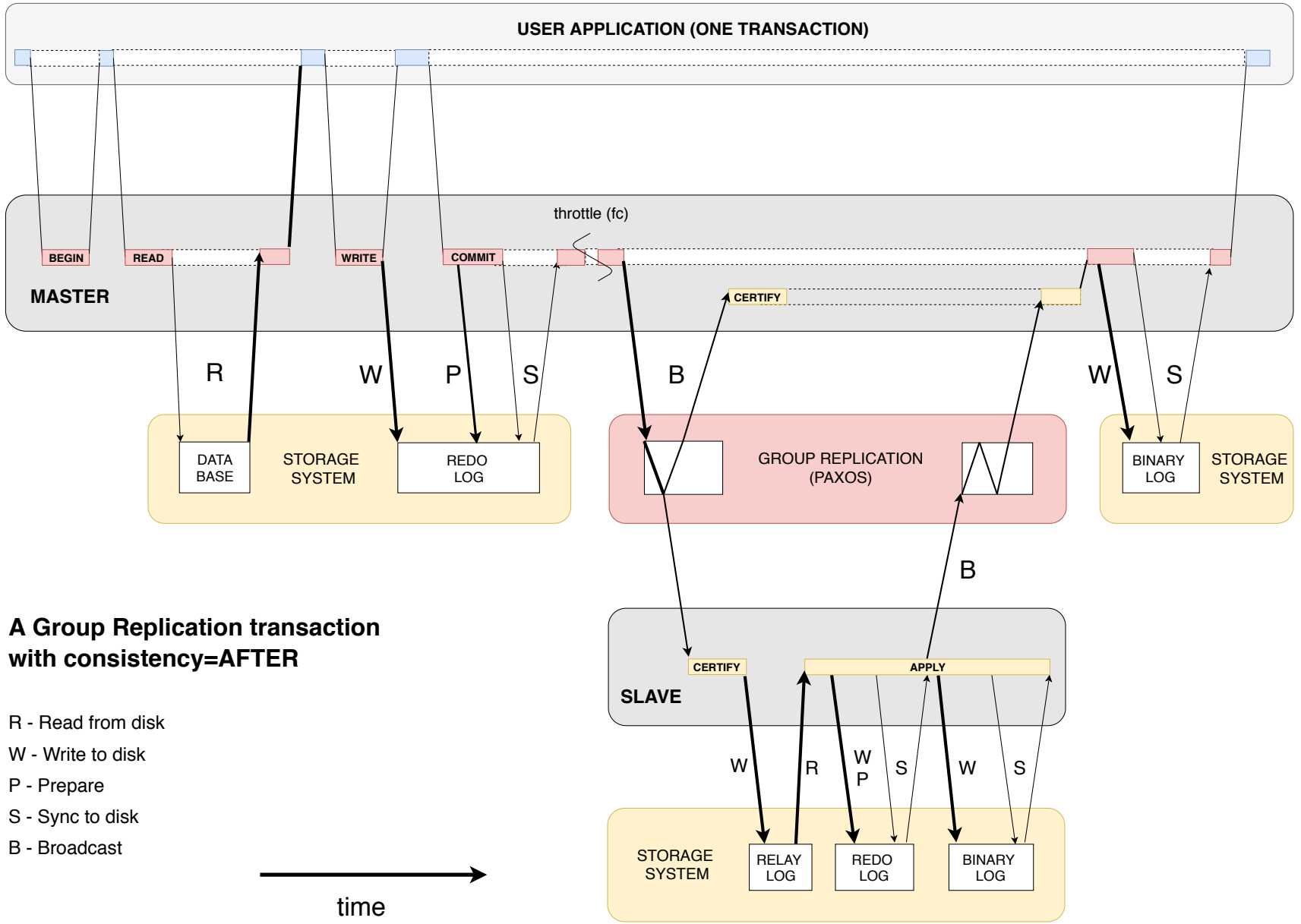
MySQL
Group Replication

- Changes to the lifecycle:
  - Group replication intercepts the transactions before they are committed and broadcasts them to the network;
  - The Paxos protocol in GR requires at least 1 or 1.5 RTT to propagate the messages, although 2 RTT is more common;
  - After that the certification takes over and the rest of the pipeline is similar to asynchronous replication.



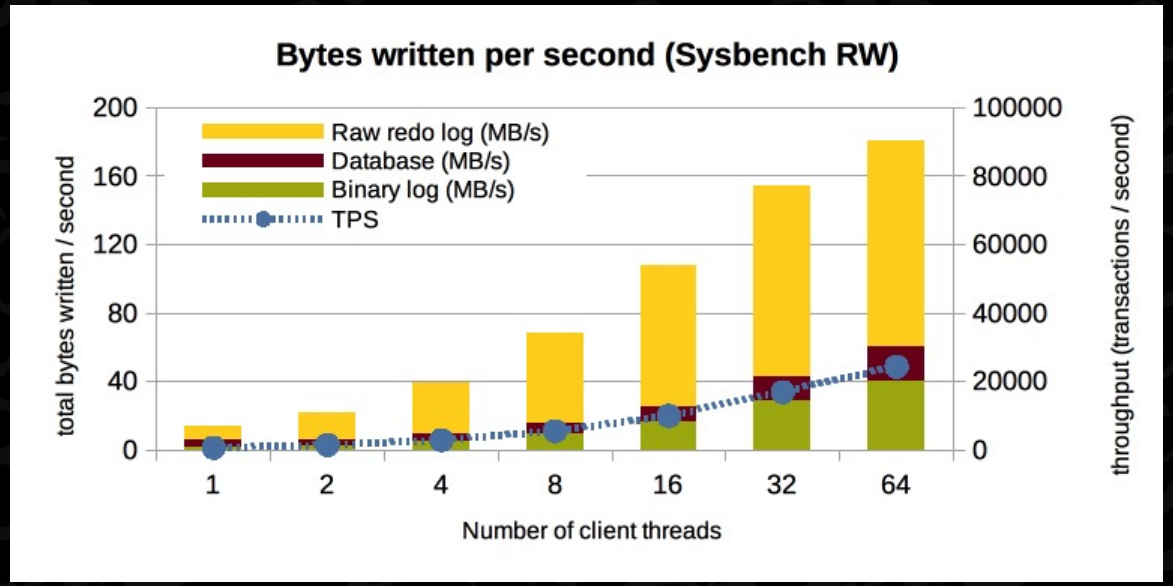
MySQL
GR + Synchronous writes

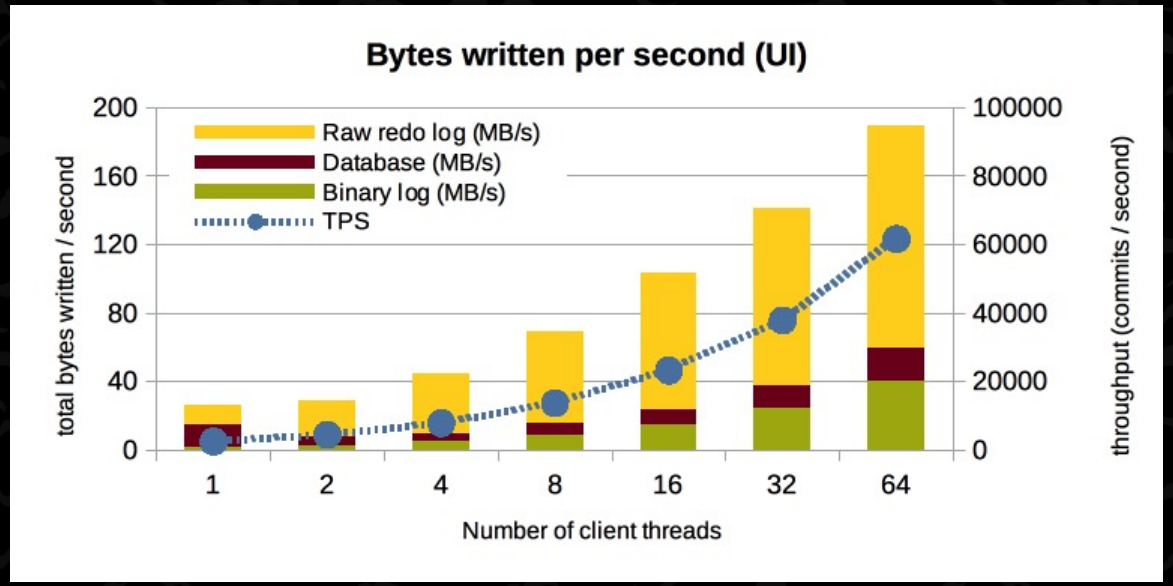
- Changes to the lifecycle:
  - With transaction consistency = AFTER, the commit is only finished after all members have signaled that they have the transaction prepared on disk;
  - That extends the transaction latency, but the apply is really synchronous.



Sysbench workloads
A sample write pattern

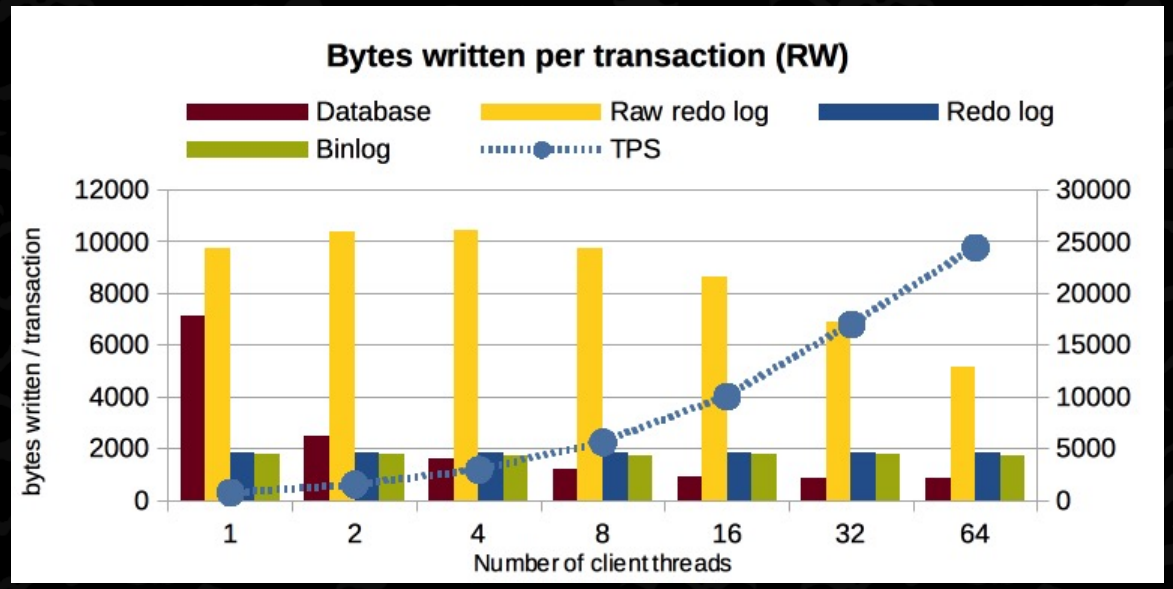
- MySQL disk throughput while running Sysbench RW and Update Index in a bare metal machine with local SSD.
- Main observations:
  - At load, the redo log takes 69% of disk write bandwidth, the binary log takes 21% and the database takes 10%;
  - The total consumed write bandwidth is very similar in both, at 180MB/s (RW) and 190MB/s (UI).

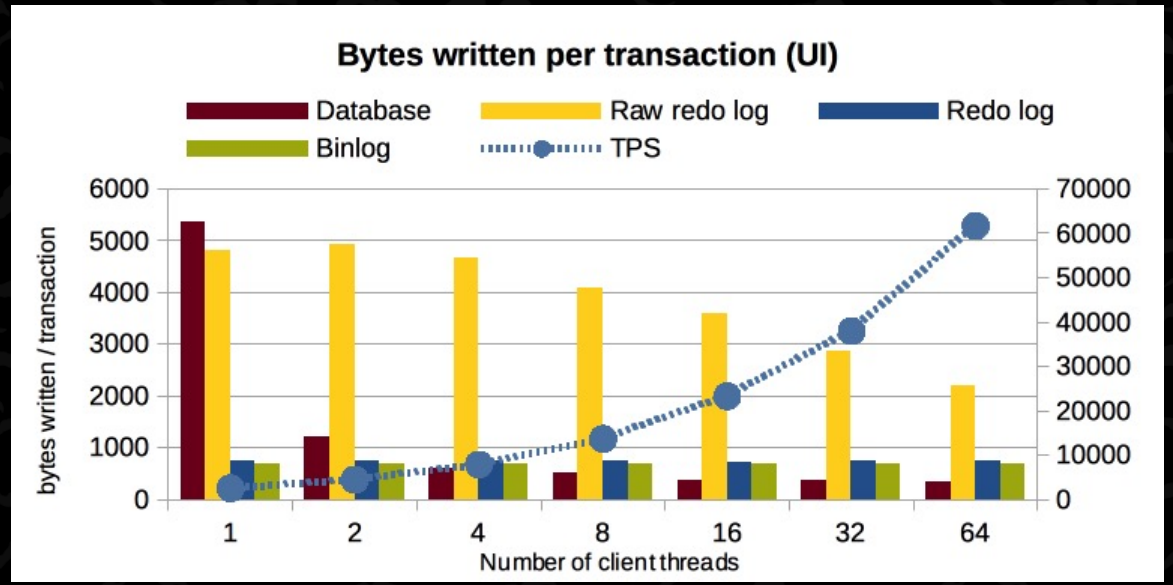




Sysbench workloads
Data written per transaction

- Main observations:
  - The transaction size on the redo log and on the binary log is similar, with the redo log being larger by 7% to 10%;
  - The additional write bandwidth for the redo log is for rewriting blocks as additional data is logged, but only part of these writes reach the disk;
  - However, fsync'ing frequently forces the intermediate writes to be sent to disk, and that can happen multiple times per transaction.





Cloud Infrastructure

- Deploying a database system in any Cloud infrastructure introduces a few performance problems one must be aware.

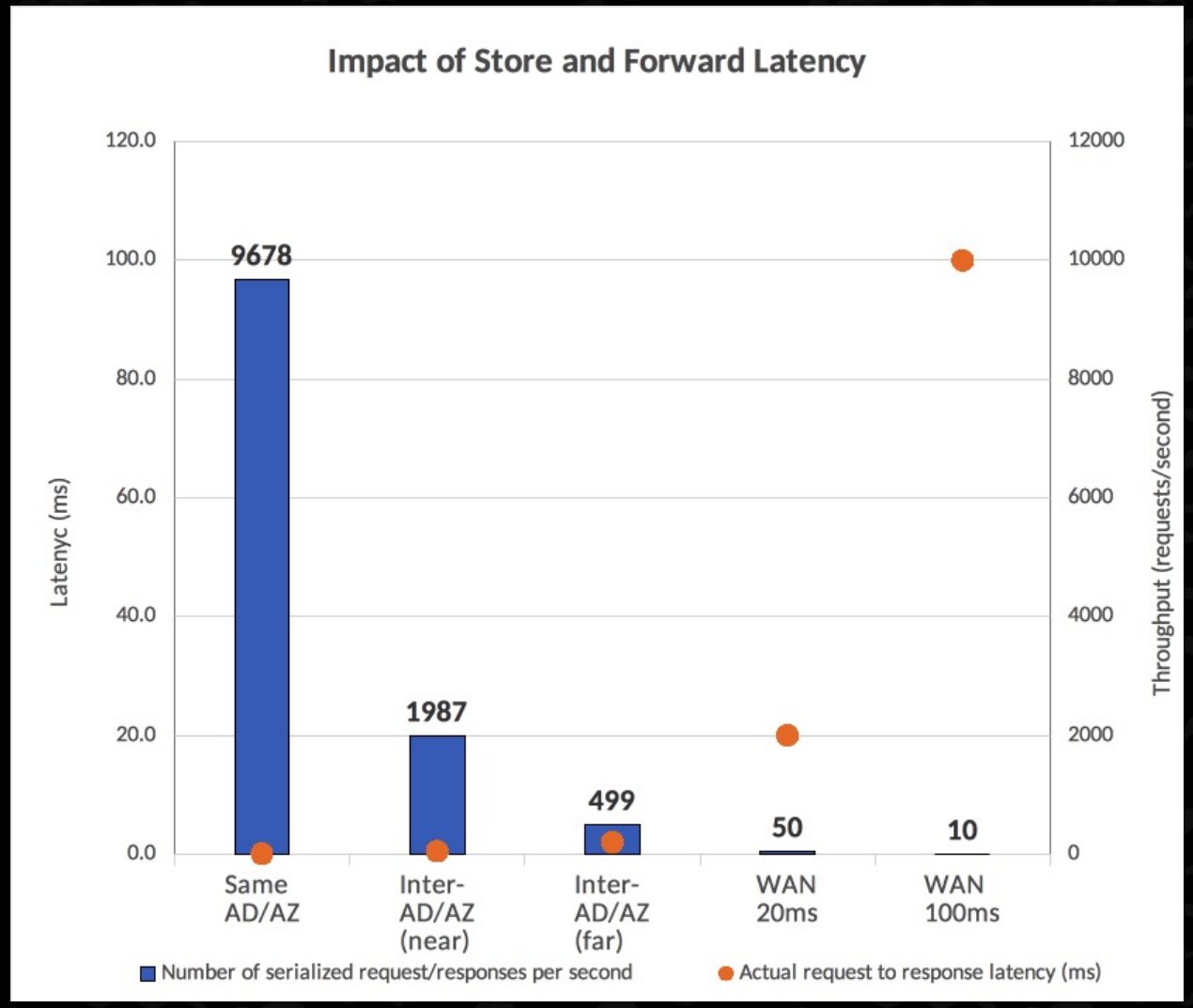
- Some are actually new incarnation of old problems, that the availability of cheap highly multi-core machines with flash storage was making us forget:
- storage is slow - network storage must always pay the network latency penalty;
- computing power is expensive - every core used is now billed;
- network bandwidth is limited - it is split between the VMs in the server and usually even storage needs to go through it.

- But here we will focus more on network latency.

Cloud Infrastructure

Latency is very important

- **The main factor affecting the performance of MySQL in a Cloud deployment is latency.**
- That latency impacts in the connections between:
  - the client and the server,
  - the server and the storage ,
  - the server and its replicas in a replication topology.
- Transaction execution is mostly affected by the latency to fsync the redo and binlog to disk.

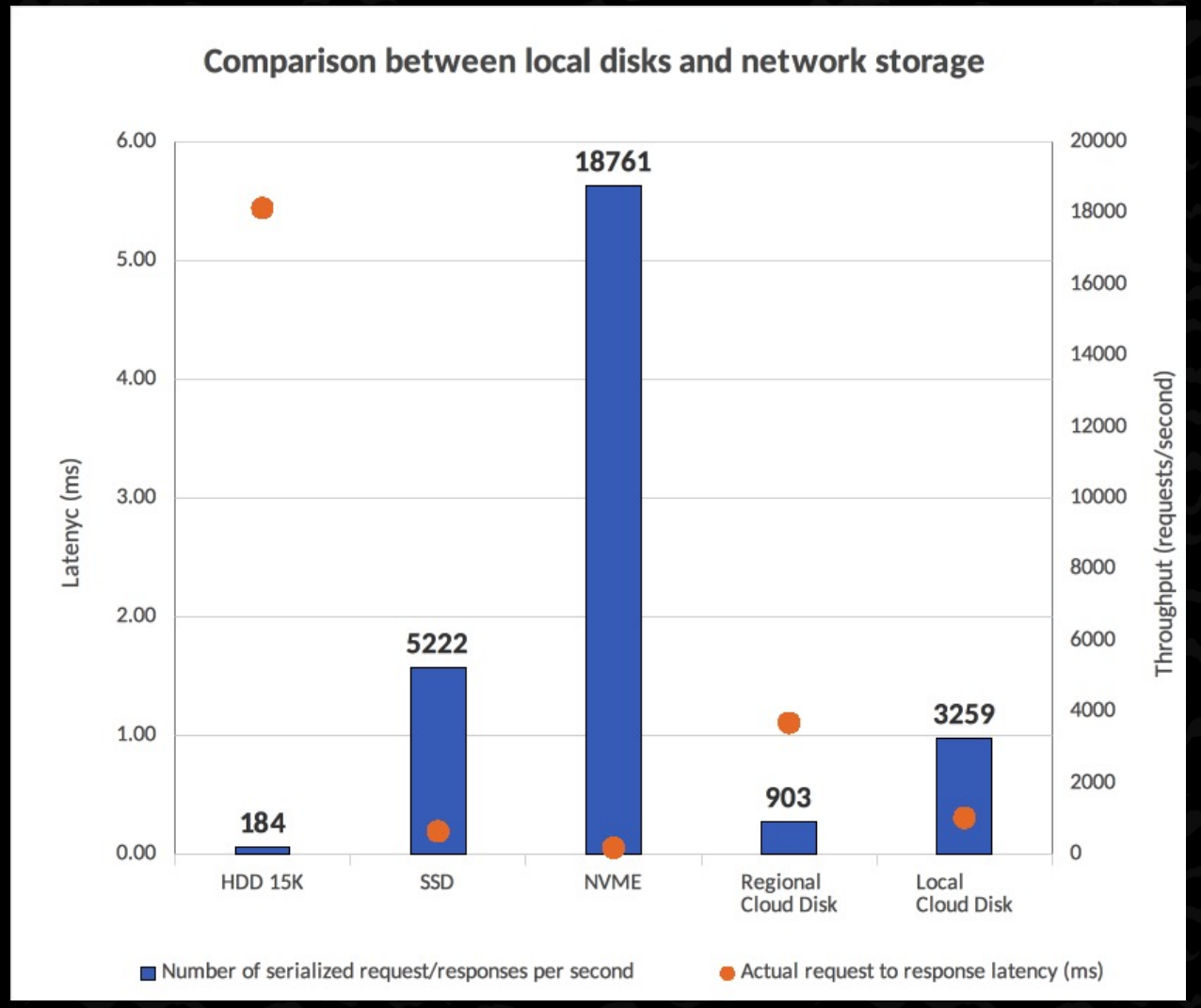


	Same AD/AZ	Inter-AD/AZ (near)	Inter-AD/AZ (far)	WAN 20ms	WAN 100ms
Bandwidth (Mbytes/s, full duplex)	1250	1250	1250	1250	1250
Request size (bytes)	4096	4096	4096	4096	4096
Response size (bytes)	64	64	64	64	64
Network hops	1.00	1.00	1.00	1.00	1.00
Client latency (RTT)	0.10	0.50	2.00	20.00	100.00

Cloud Infrastructure
Storage is also network

- Clouds usually support nodes with networked storage and/or nodes with local storage.
  - Local storage can provide very low latency, but it is expensive and has a number of limitations.
  - Networked storage is less expensive, has more features and is easier to manage both to users and to the infrastructure management.

- Networked storage is much more common, and it allows the deployments to be simpler, there will be only one interface in and out of computing nodes – the network.

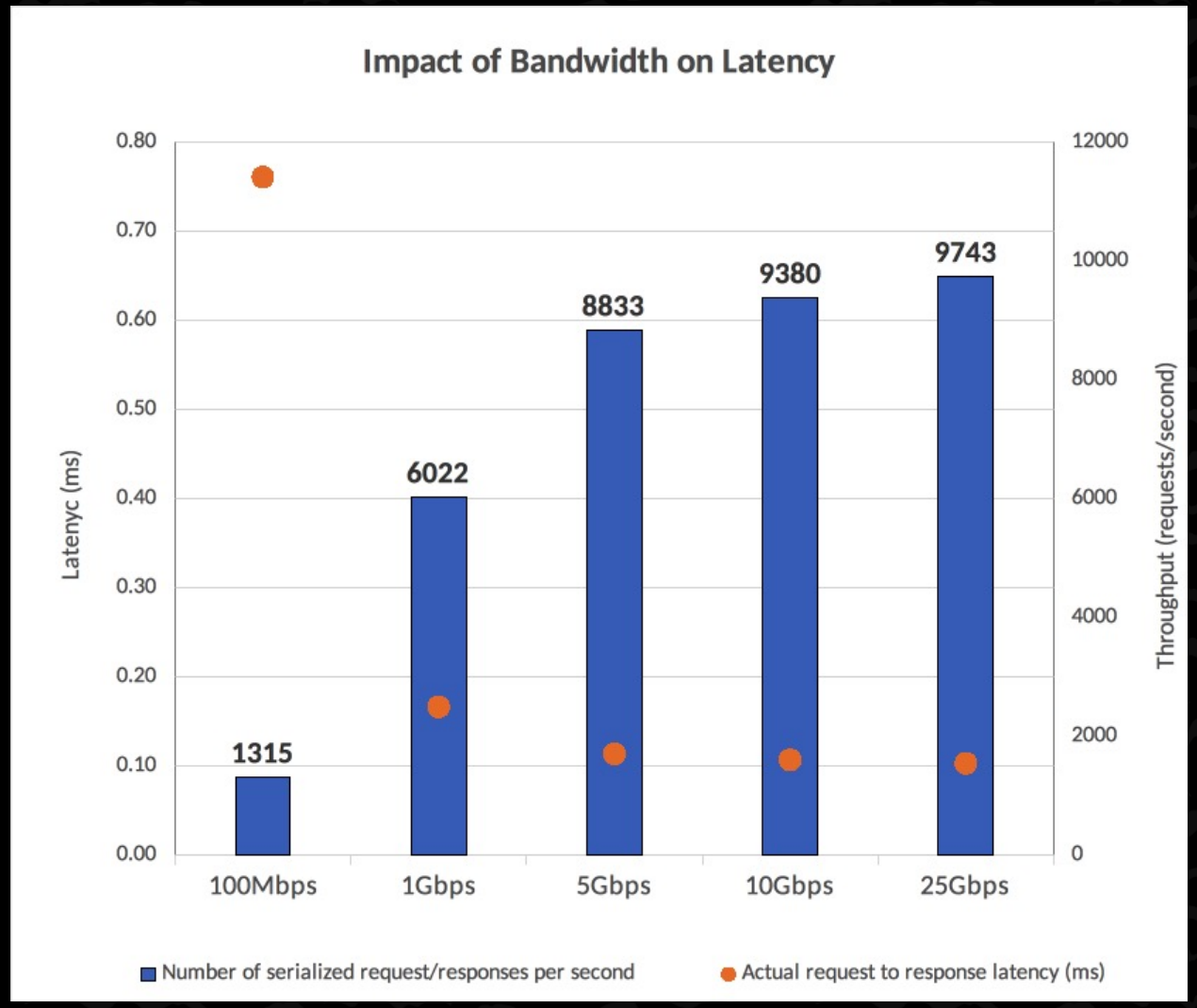


	HDD 15K	SSD	NVME	Regional Cloud Disk	Local Cloud Disk
Bandwidth (Mbytes/s, full	200	500	2500	1200	1200
Request size (bytes)	64	64	64	64	64
Response size (bytes)	8192	8192	8192	8192	8192
Network hops	1.00	1.00	1.00	1.00	1.00
Internal latency	0.00	0.00	0.00	2.00	0.40
Client latency (RTT)	5.40	0.18	0.05	0.10	0.10

Cloud Infrastructure
Bandwidth is also latency

- The available network bandwidth is also relevant to the actual latency of a request:
  - a reply can only leave the remote node when the request message is fully received, so the size of the packets must be considered;
  - there is nothing similar to cut-through routing for services.

- Sharing the network bandwidth by multiple services, particularly between serving client requests and accessing storage, amplifies this issue.



	100Mbps	1Gbps	10Gbps	25Gbps	100Gbps
Bandwidth (Mbytes/s, full duplex)	12.5	125	1250	3125	125000
Request size (bytes)	64	64	64	64	64
Response size (bytes)	1024	1024	1024	1024	1024
Network hops	1.00	1.00	1.00	1.00	1.00
Max network utilization	43.80%	7.54%	0.81%	0.33%	0.01%
Client latency (RTT)	0.10	0.10	0.10	0.10	0.10

A transaction simulator
Estimating MySQL throughput

- To help estimate the behavior a small simulator was built as a spreadsheet that includes parameters of the workload and of the underlying computing infrastructure, and generates simulations on how the system could behave.

TPS Simulator	OLTP RW	OLTP Write-only	Update Index
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	0.3	0.1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	6	1
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	2.00	1.30
maximum throughput	30,522	97,990	269,474
maximum cpu effectiveness	95%	92%	84%

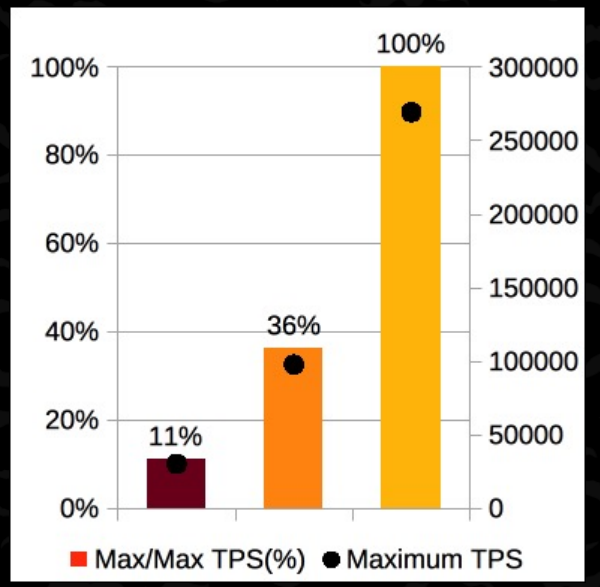
The charts that follow, marked with **SIMULATION** on the title, are an exercise around estimating the upper bound on performance taking into account some characteristics of the underlying computing system.

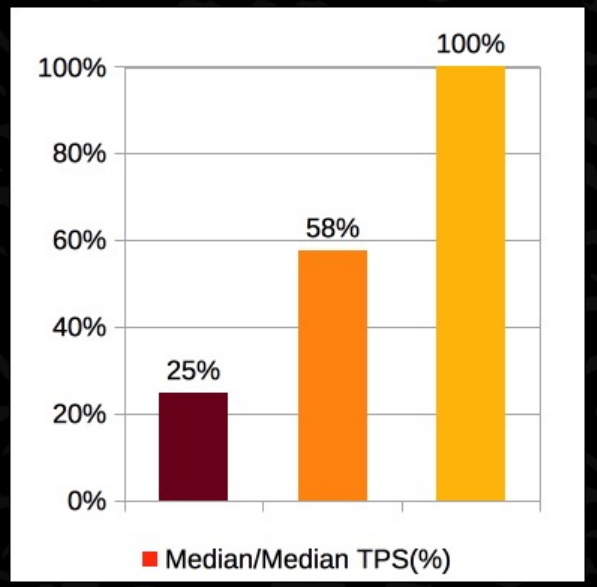
The following charts do not result from actual measurements!

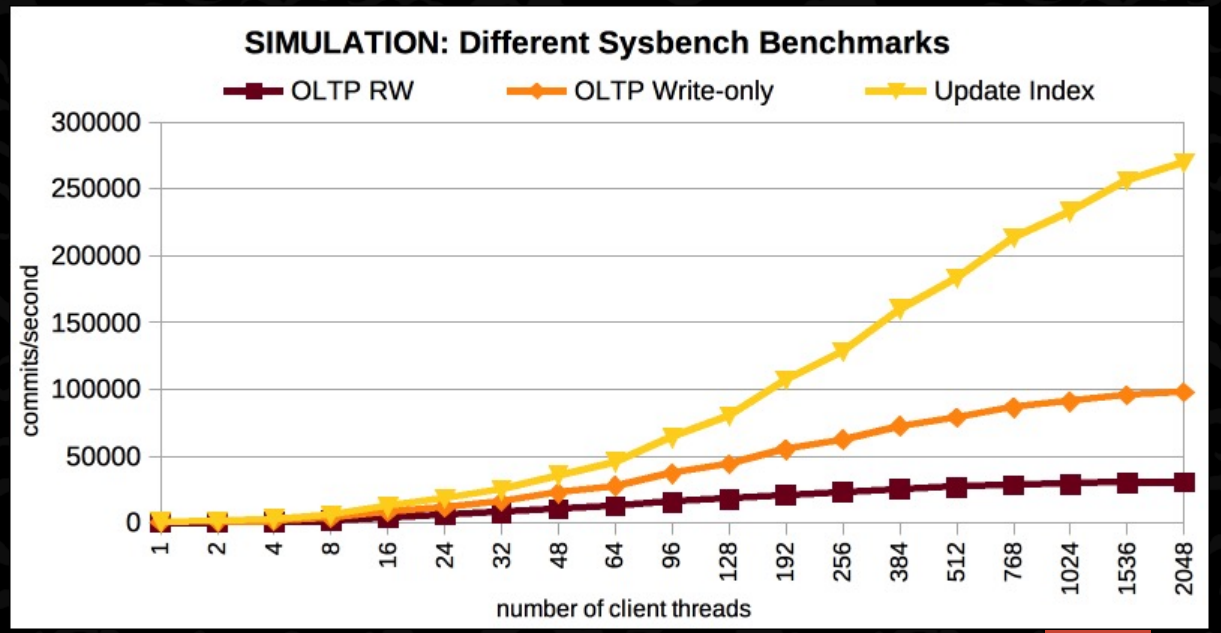
A transaction simulator
Different Sysbench Benchmarks

- Observations:
  - Different workloads imply different behaviors from the system, so it is important to know the characteristics of the workload;
  - The Sysbench benchmarks presented represent a read-write workload, a write-only workload and a single-update, and the shape is similar to the behavior found in practice.

TPS Simulator	OLTP RW	OLTP Write-only	Update Index
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	0.3	0.1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	6	1
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	2.00	1.30
maximum throughput	30,522	97,990	269,474
maximum cpu effectiveness	95%	92%	84%

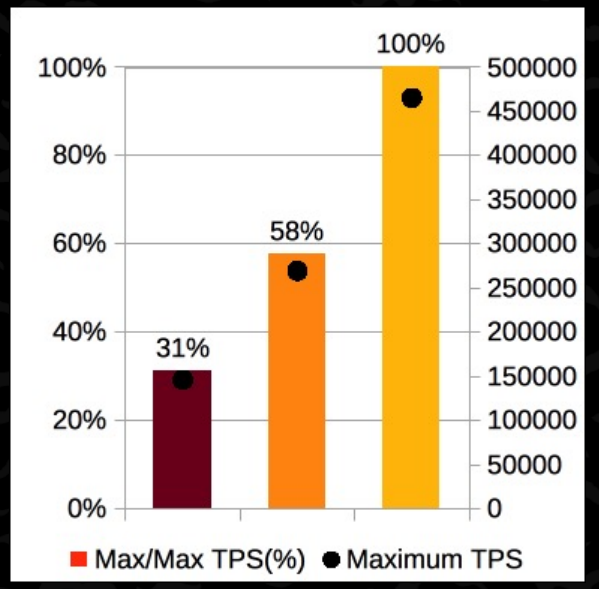


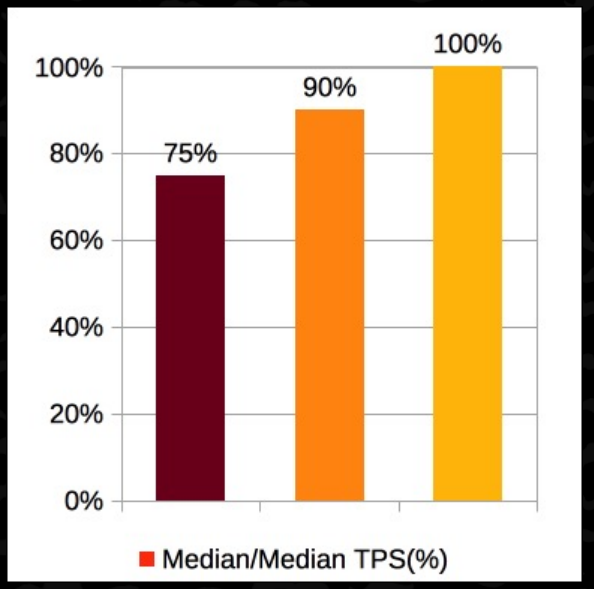




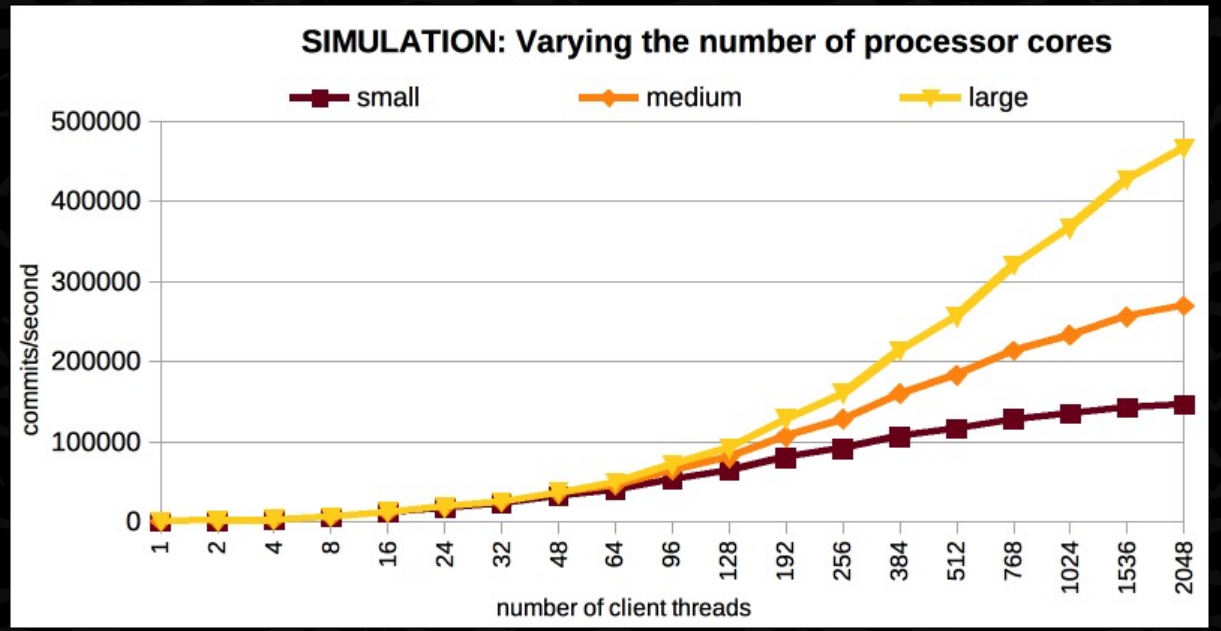
A transaction simulator
Number of processor cores

- Observations:
  - When the computing effort per transaction is low, the gain from going to larger machines becomes smaller;
  - As the latency from other factors dominates, the gain is only realized when the number of threads is very high.





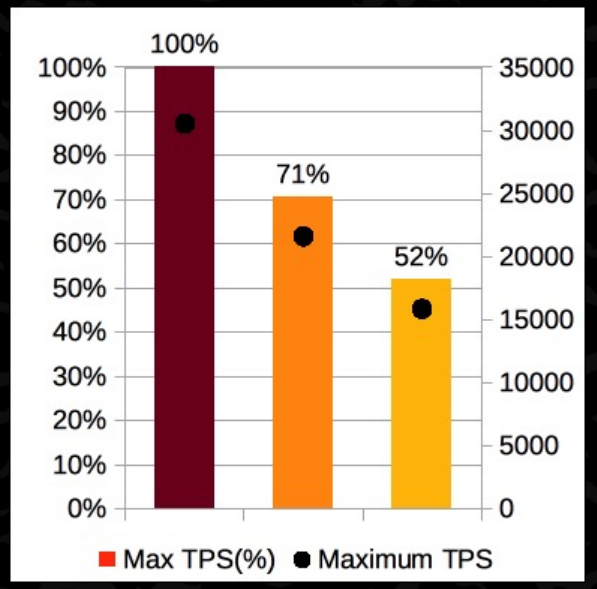
TPS Simulator	small	medium	large
server cores	16	32	64
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	0.1	0.1	0.1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	1	1	1
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	1.30	1.30	1.30
maximum throughput	146,286	269,474	465,455
maximum cpu effectiveness	91%	84%	73%

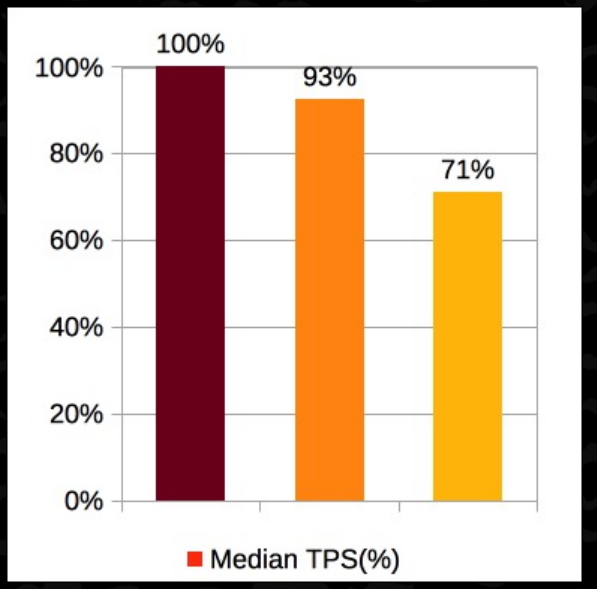


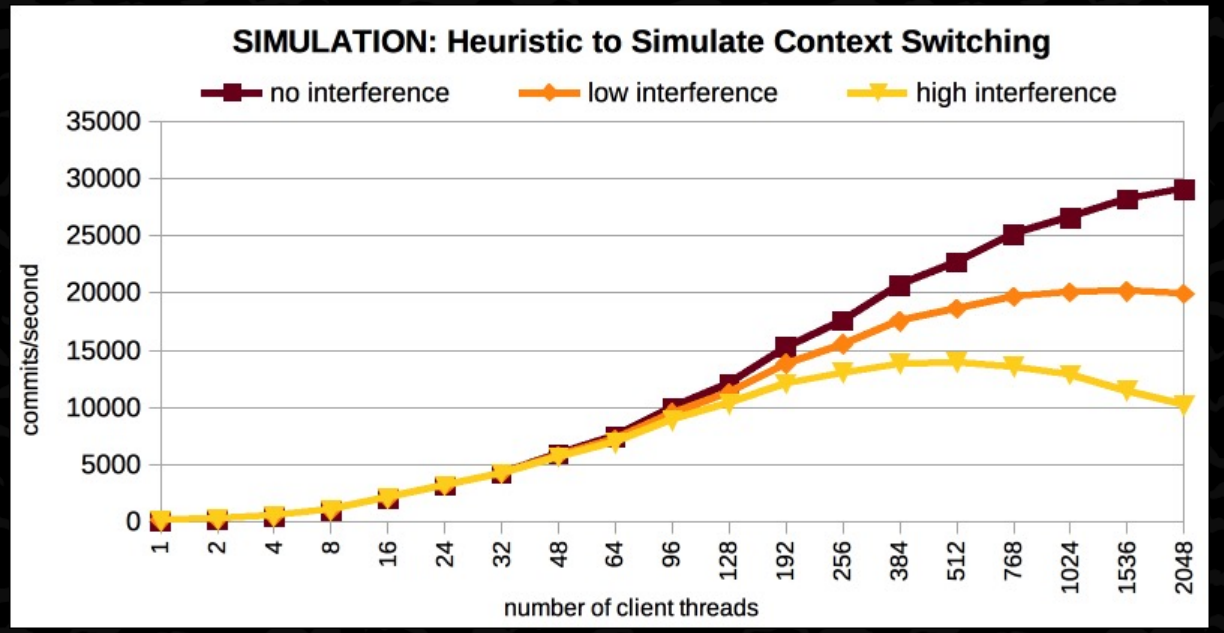
A transaction simulator
Context switching impact

- Observations:
  - The context switching between threads is expensive, and many threads become less effective if the workload is CPU bound;
  - As the number of threads grows the performance may drop, and there is a parameter to mimic some interference between threads.

TPS Simulator	no interference	low interference	high interference
server cores	32	32	32
context switching loss estimate	0%	5%	10%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	4.10	4.10
maximum throughput	30,522	21,603	15,854
maximum cpu effectiveness	95%	68%	50%

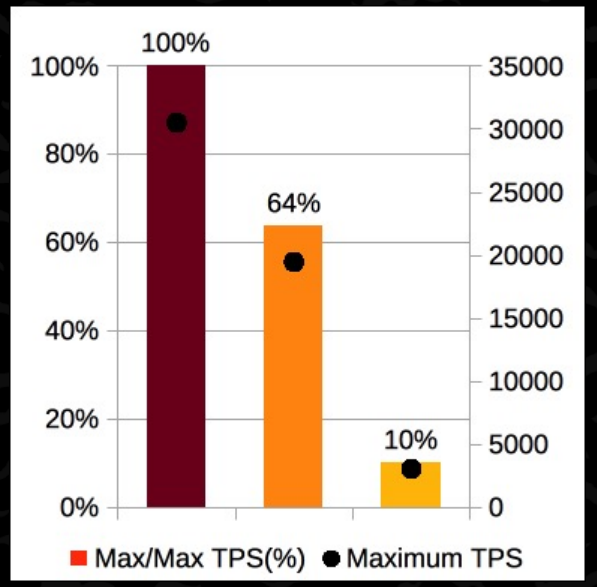


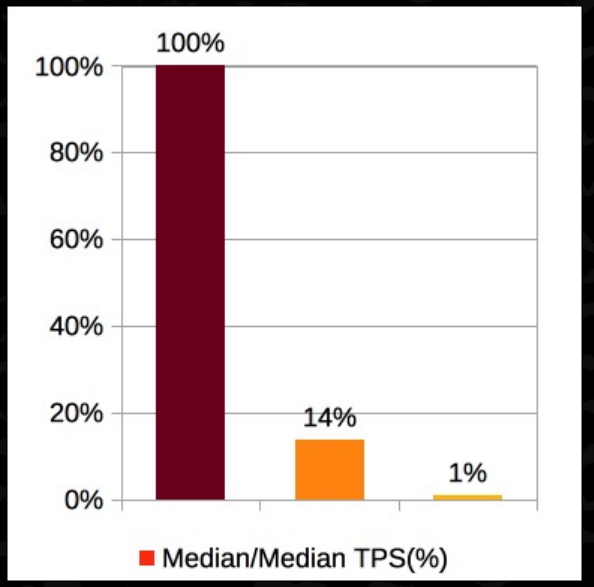




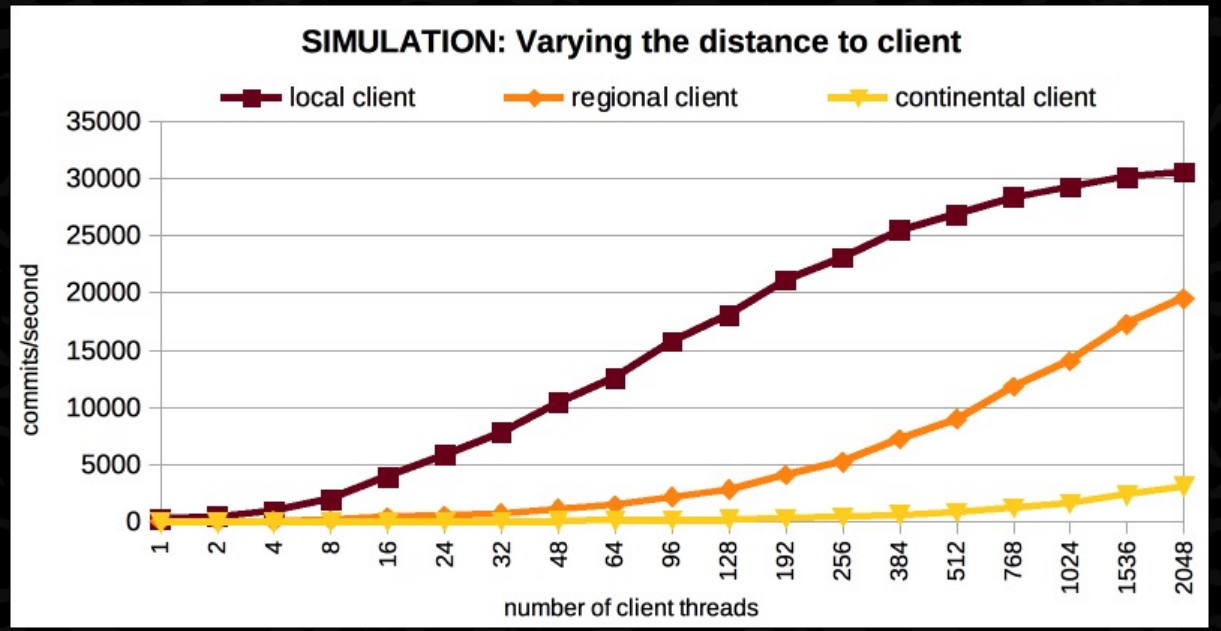
A transaction simulator
Impact of the distance to client

- Observations:
  - The distance between the clients and the server can have a dramatic impact on the throughput;
  - That is mainly due to the synchronous request/response model of the MySQL client protocol.



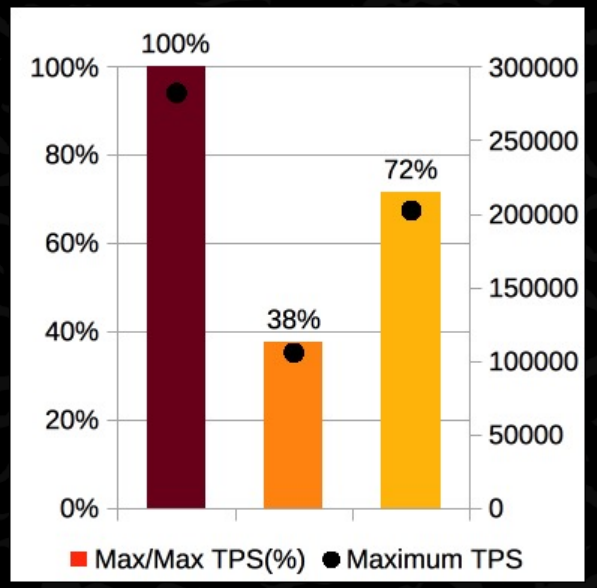


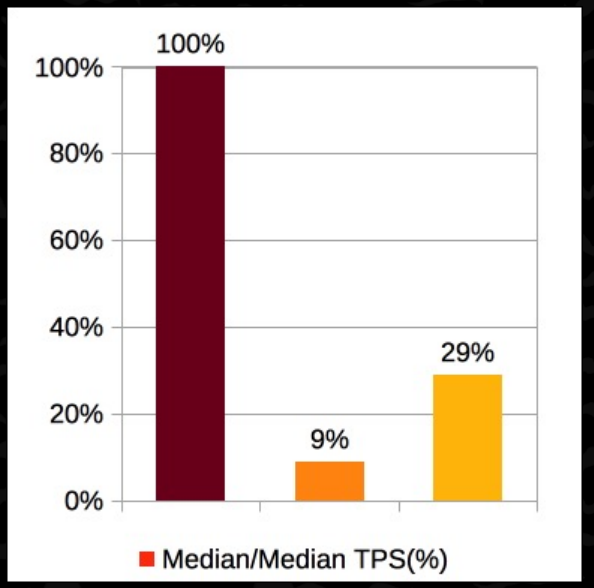
TPS Simulator	local client	regional client	continental clien
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	2	30
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	42.10	602.10
maximum throughput	30,522	19,486	3,079
maximum cpu effectiveness	95%	61%	10%



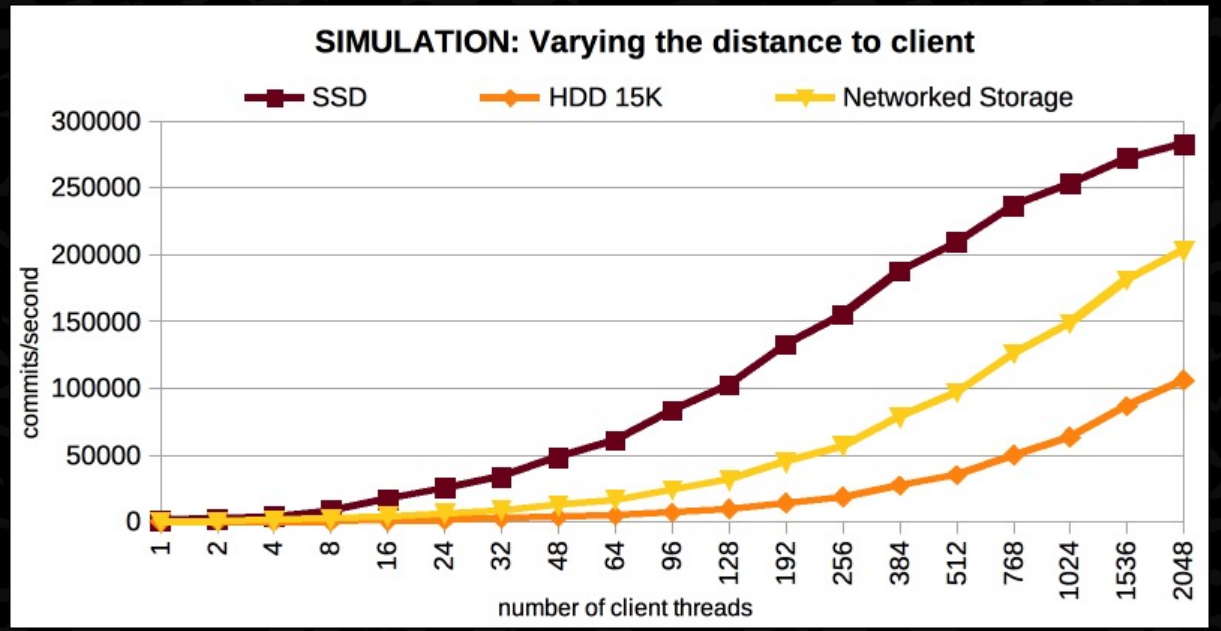
A transaction simulator
Impact of the storage technology

- Observations:
  - The performance of the storage system is critical to the throughput of MySQL;
  - While the throughput can grow to the point where the difference is less visible, at lower thread counts the latency, in particular, is a major factor impacting.





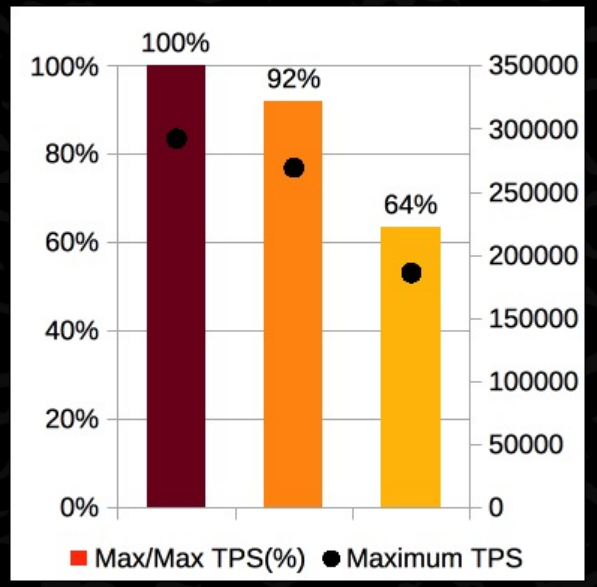
TPS Simulator	SSD	HDD 15K	etworked Storag
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	0.1	0.1	0.1
3. data read time (ms)	0.175	5.4	1
4. engine log write time (ms)	0.175	1	0.6
5. engine log sync time (ms)	0.2	5.4	1.2
6. group replication time (ms)	0	0	0
7. write to binlog time (ms)	0.175	1	0.6
8. binlog sync time (ms)	0.2	5.4	1.2
number of queries per transaction	1	1	1
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	0.95	13.00	3.80
maximum throughput	282,483	106,114	202,772
maximum cpu effectiveness	88%	33%	63%

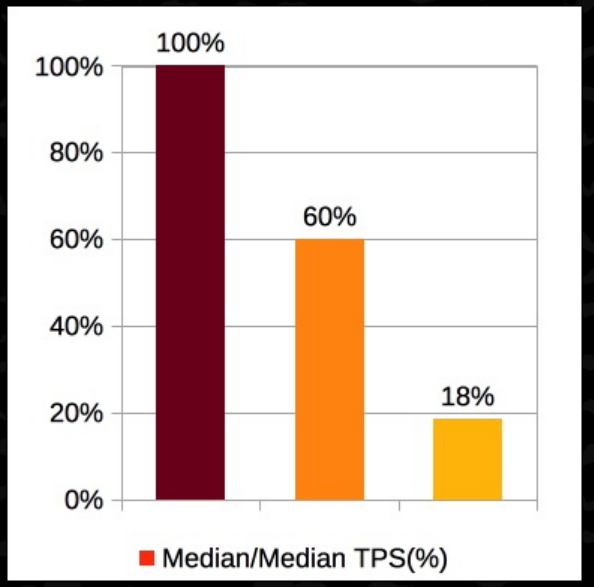


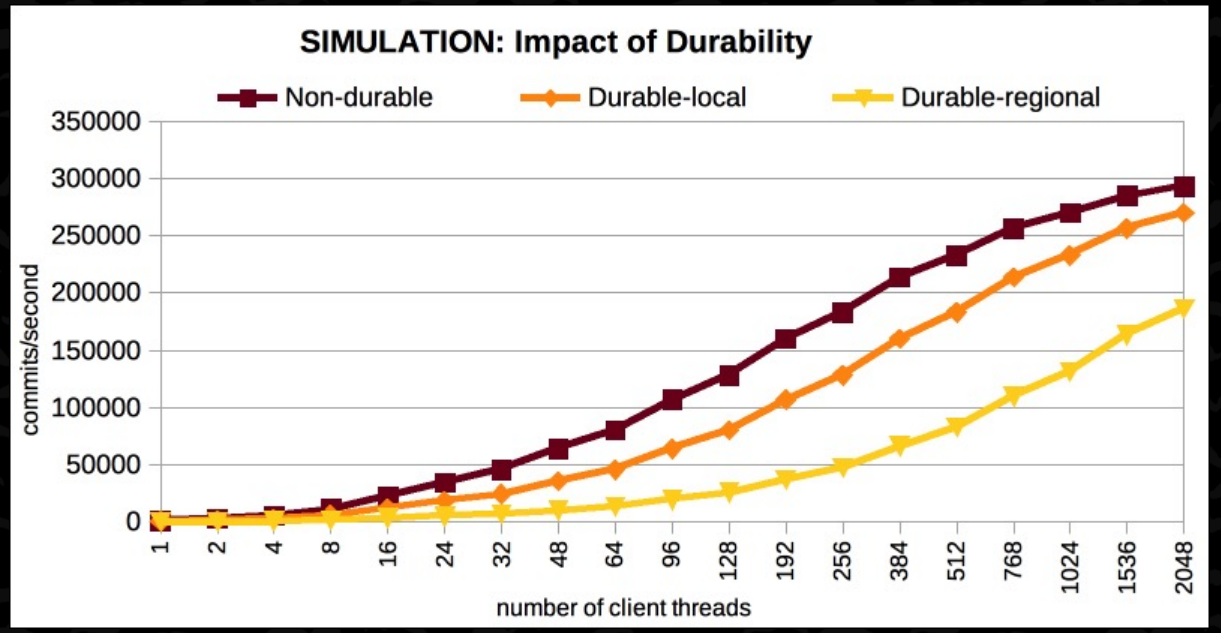
A transaction simulator
Impact of the durability settings

- Observations:
  - The impact of durability is hidden if the number of threads is high;
  - But at lower thread counts, the impact is larger, as the throughput curve is shifted to the right;
  - Again, this is similar to what is observed in real systems.

TPS Simulator	Non-durable	Durable-local	Durable-regional
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	0.1	0.1	0.1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0	0.3	2
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0	0.3	2
number of queries per transaction	1	1	1
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	0.70	1.30	4.70
maximum throughput	292,571	269,474	186,182
maximum cpu effectiveness	91%	84%	58%



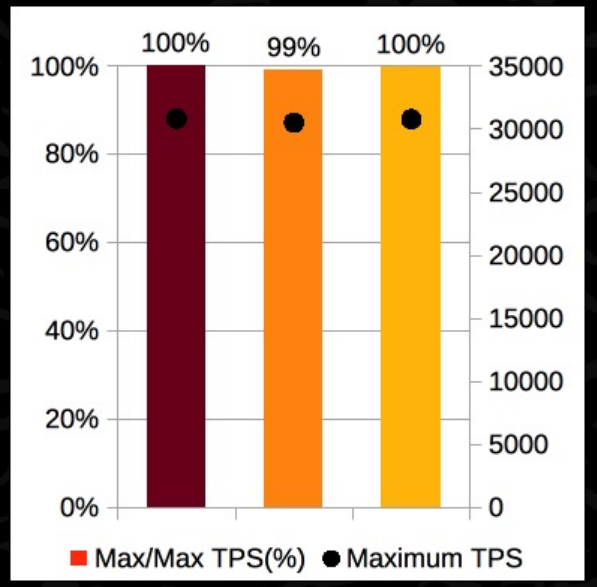


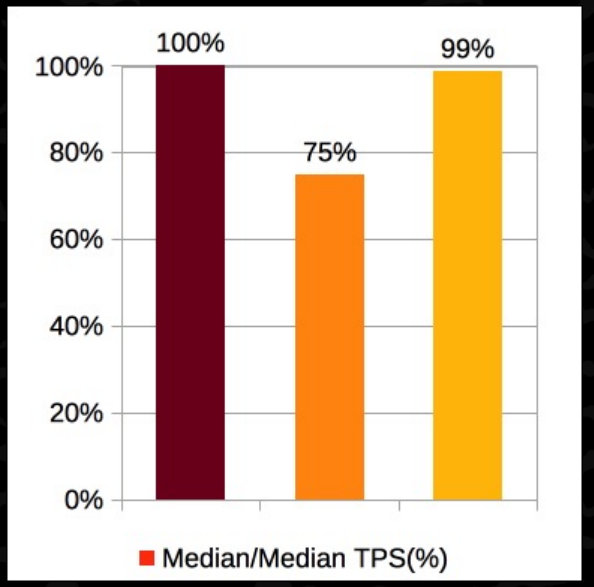


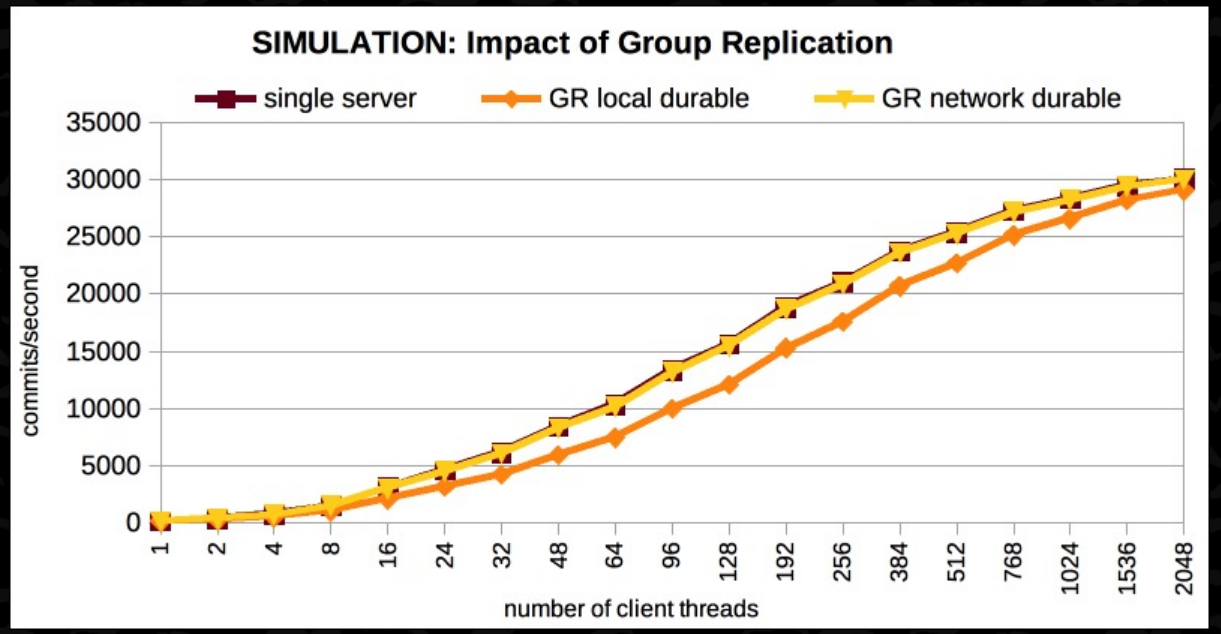
A transaction simulator
Impact of Group Replication

- Observations:
  - Group Replication can be used to avoid using fsyncs in the local storage, in practice replacing local for network durability;
  - Since the GR protocol depends only on two RTT, it may bring better performance than writing to a distributed storage system.

TPS Simulator	single server	GR durable	R network durab
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0
6. group replication time (ms)	0	0.2	0.2
7. write to binlog time (ms)	0	0.2	0.2
8. binlog sync time (ms)	0	0.3	0
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	3.40	4.10	3.50
maximum throughput	30,843	30,522	30,797
maximum cpu effectiveness	96%	95%	96%



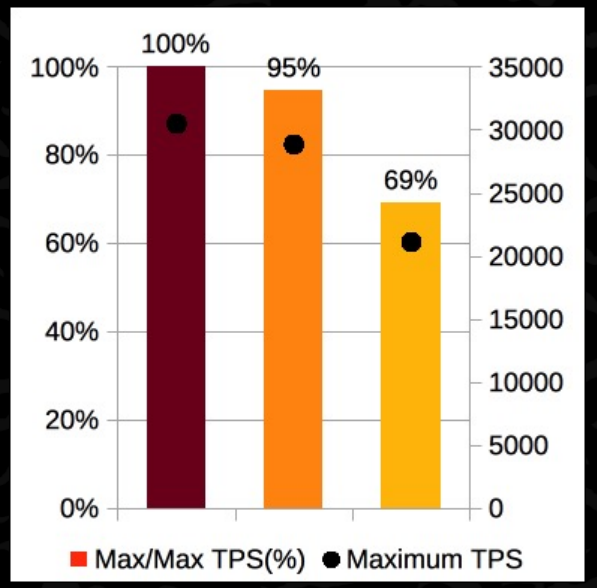


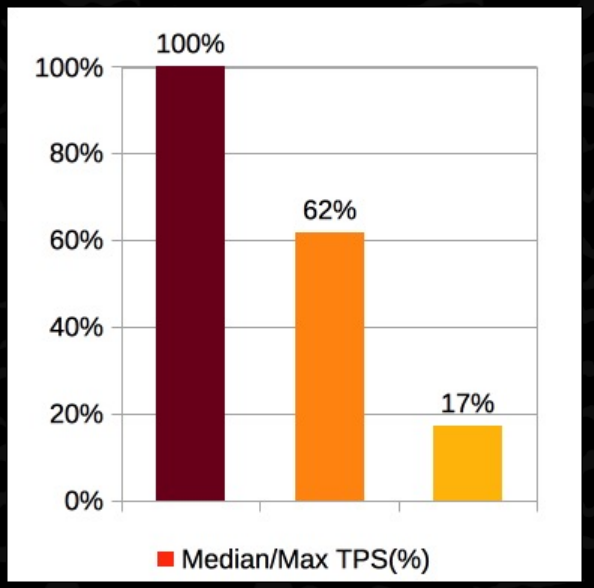


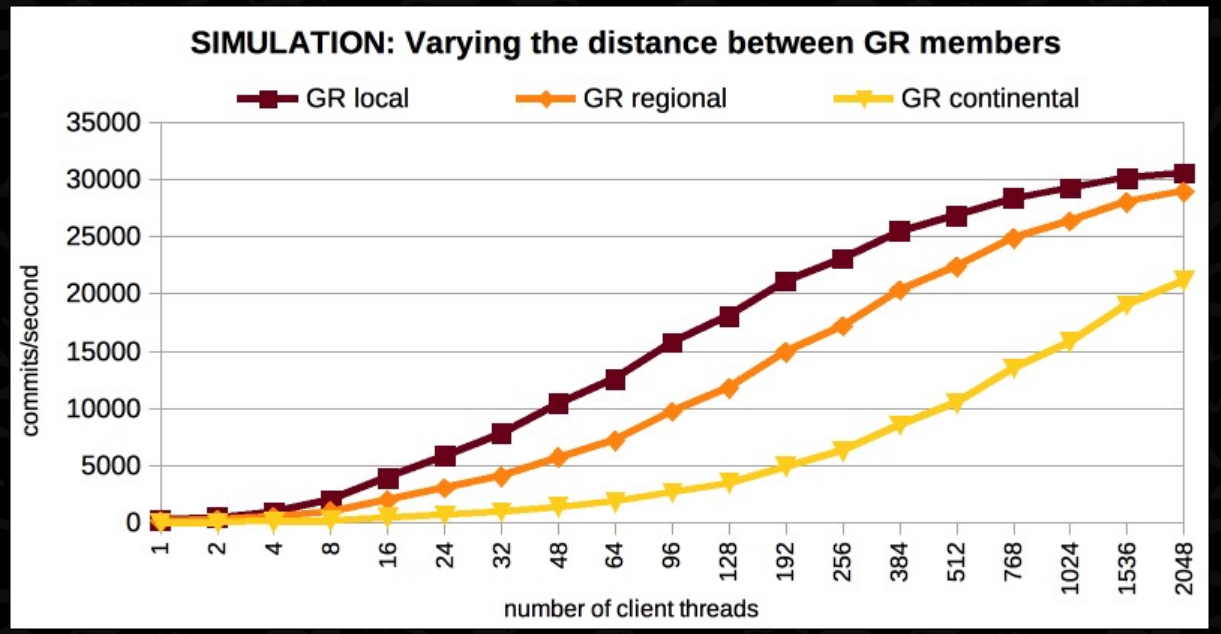
A transaction simulator
Impact of GR member distance

- Observations:
  - The distance between members is very significant;
  - However, it applies once per transaction, and only for those that write to disk, so the impact is less than the client distance.

TPS Simulator	GR local	GR regional	GR continental
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	4	30
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	7.90	33.90
maximum throughput	30,522	28,886	21,135
maximum cpu effectiveness	95%	90%	66%



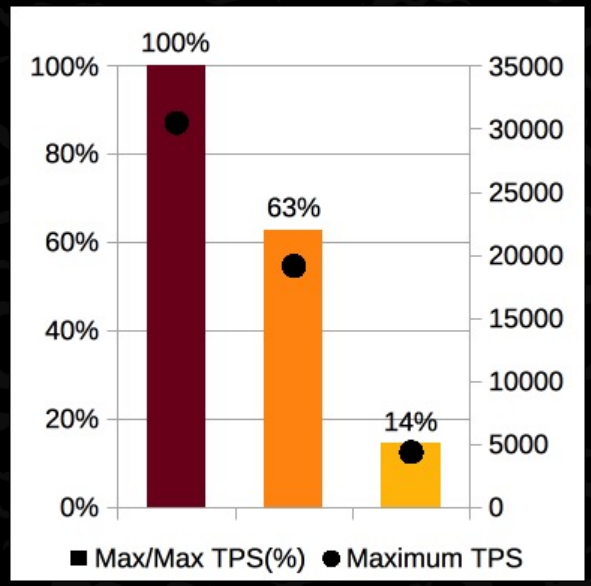


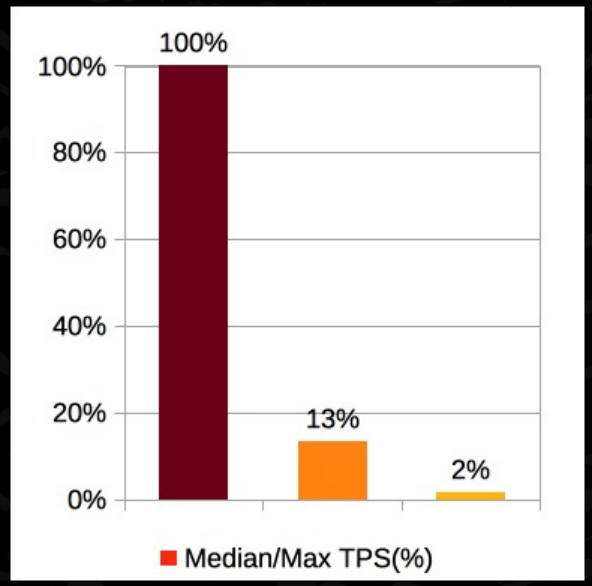


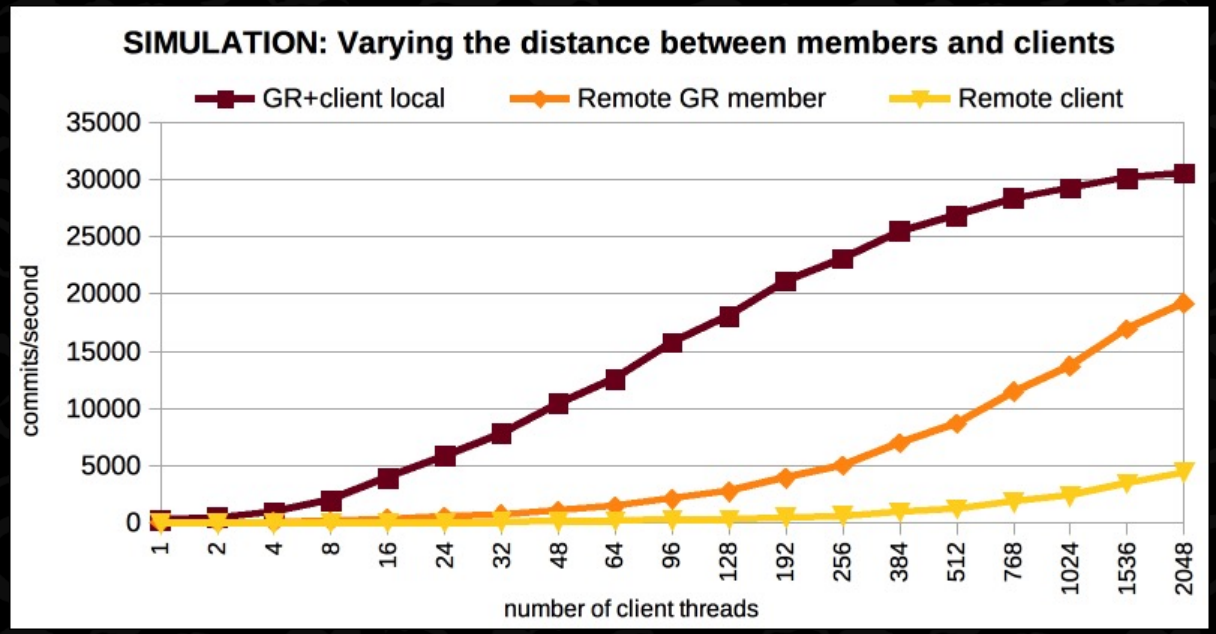
A transaction simulator
WAN latency: GR vs client

- Observations:
  - Comparing the impact of the latency in client to server connection to the impact of having group replication members with similar latency, one can see that it is better to use GR;
  - GR only adds latency as a single message at commit time, instead of having the latency impact all queries.

TPS Simulator	GR+client local	Remote GR member	Remote client
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	20
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	40	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	43.90	402.10
maximum throughput	30,522	19,158	4,403
maximum cpu effectiveness	95%	60%	14%



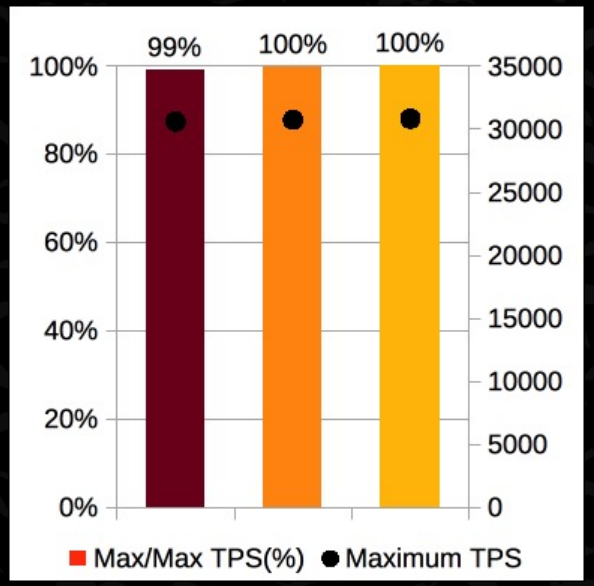


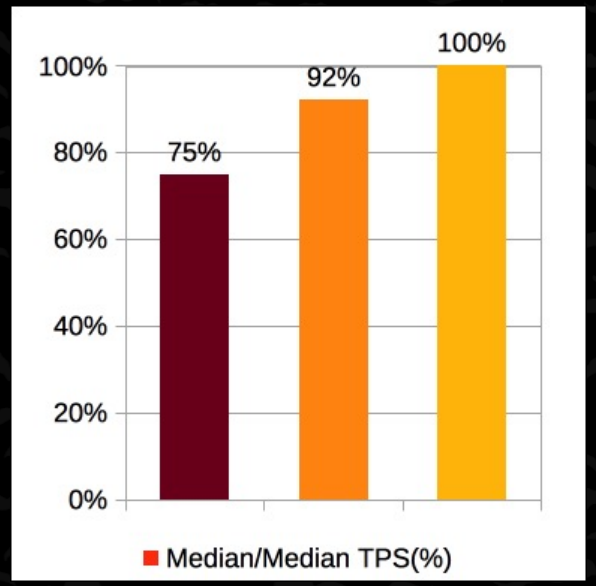


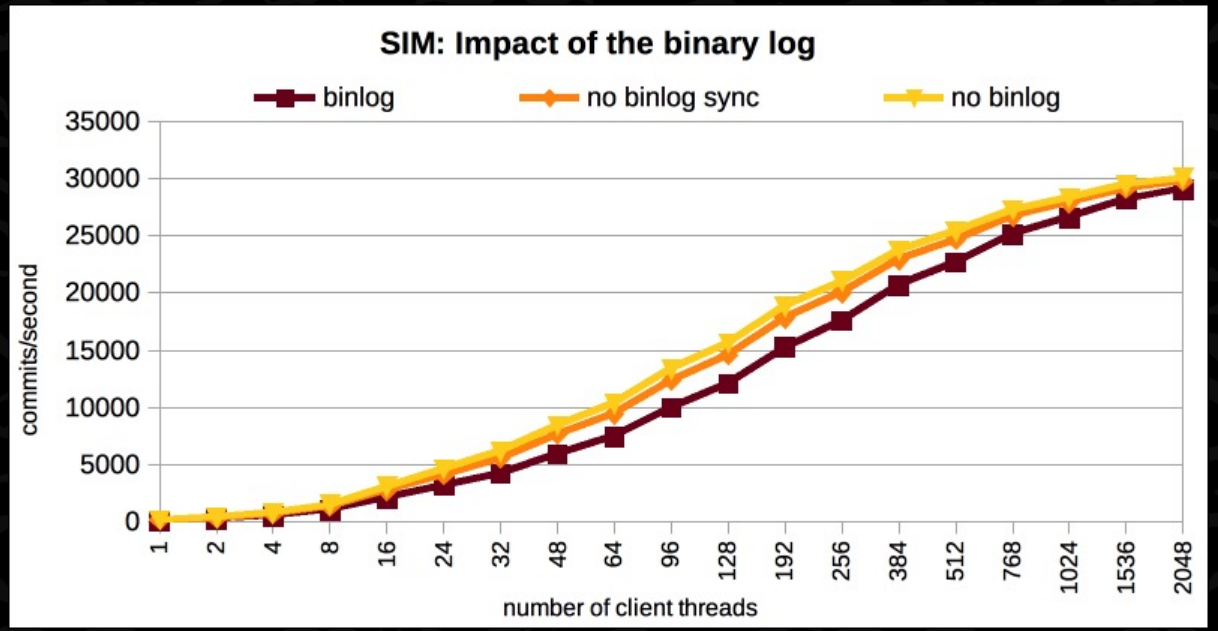
A transaction simulator
Impact of the binary log

- Observations:
  - The impact of the binary log is two-fold: it adds latency due to the write to disk and due do the fsync to disk;
  - Having lower durability allows the throughput to follow the performance of the server closer, something that is even more relevant in the Cloud, as sync time can be much larger.

TPS Simulator	binlog	no binlog sync	no binlog
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0	0	0
7. write to binlog time (ms)	**0.2**	**0.2**	**0**
8. binlog sync time (ms)	**0.3**	**0**	**0**
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	3.90	3.60	3.40
maximum throughput	30,613	30,751	30,843
maximum cpu effectiveness	96%	96%	96%



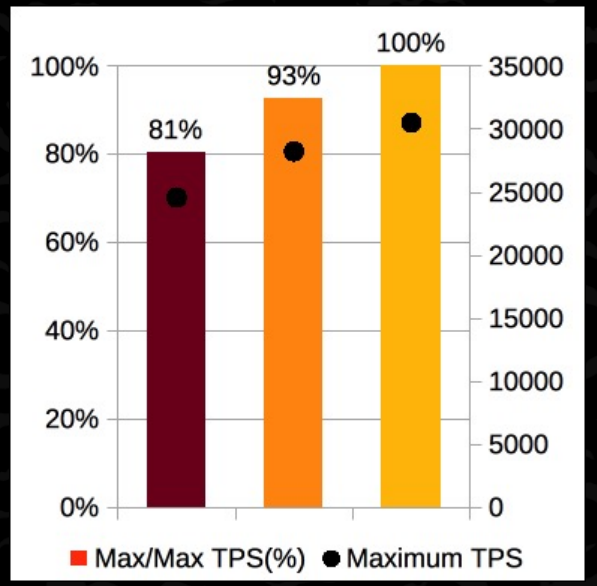


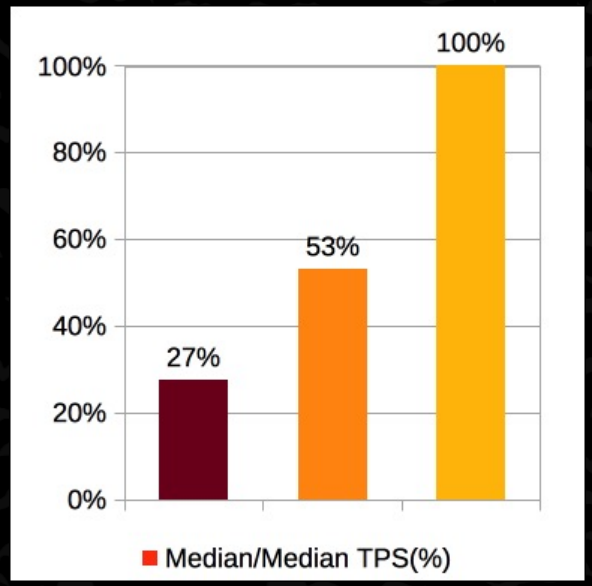


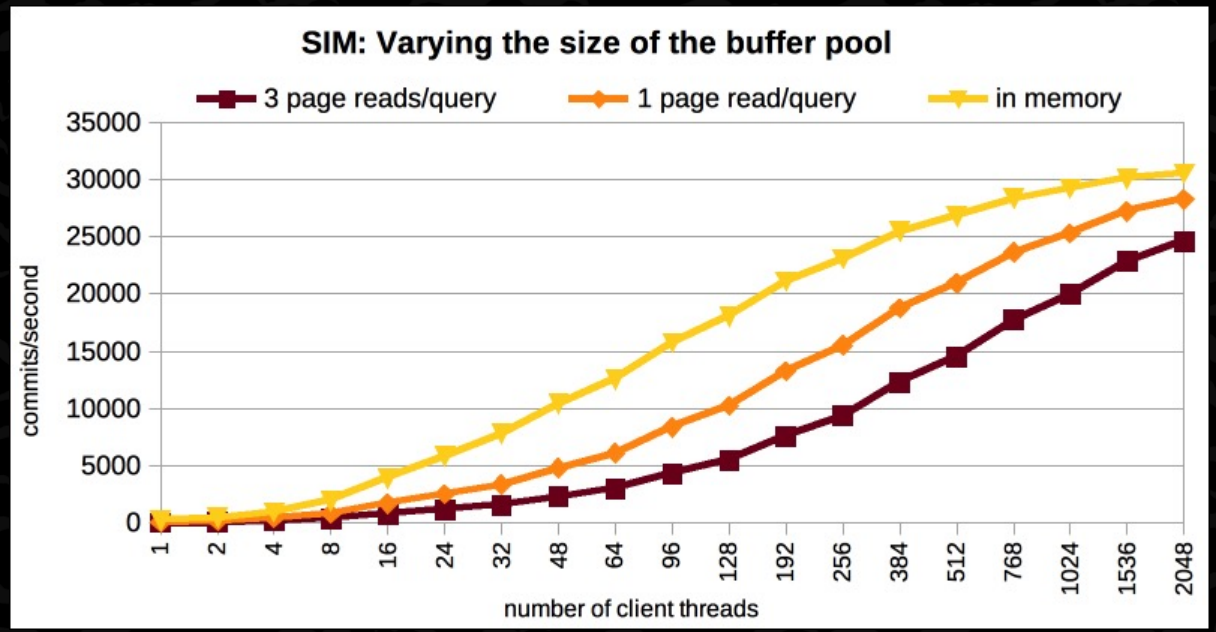
A transaction simulator
Impact of the buffer pool size

- Observations:
  - The read operations cannot be hidden in the transaction execution;
  - If the size of the buffer pool in smaller than the active data set, the probability of needing to pay the penalty becomes larger;
  - With a small buffer pool the effective use of the computing resources is impaired.

TPS Simulator	page reads/quer	page read/quer	in memory
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.1	0.1
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	20	20
number of non-local reads per transaction	54	18	0
minimum latency	20.30	9.50	4.10
maximum throughput	24,586	28,248	30,522
maximum cpu effectiveness	77%	88%	95%







A transaction simulator
Impact of an intermediate router

- Observations:
  - Using an intermediate router adds latency to all queries, so the effect is the same as having a farther away client;
  - While that impact can be reduced at higher thread counts, the impact is significant if there are not enough threads to hide the latency.

TPS Simulator	direct	local router	regional router
server cores	32	32	32
context switching loss estimate	0%	0%	0%
1. client/server latency (rtt, ms)	0.1	0.2	0.6
2. transaction execution time (ms)	1	1	1
3. data read time (ms)	0.3	0.3	0.3
4. engine log write time (ms)	0.1	0.1	0.1
5. engine log sync time (ms)	0.3	0.3	0.3
6. group replication time (ms)	0.2	0.2	0.2
7. write to binlog time (ms)	0.2	0.2	0.2
8. binlog sync time (ms)	0.3	0.3	0.3
number of queries per transaction	20	20	20
number of non-local reads per transaction	0.00	0.00	0.00
minimum latency	4.10	6.10	14.10
maximum throughput	30,522	29,638	26,563
maximum cpu effectiveness	95%	93%	83%