

Why Uber Was (Mostly) Wrong.

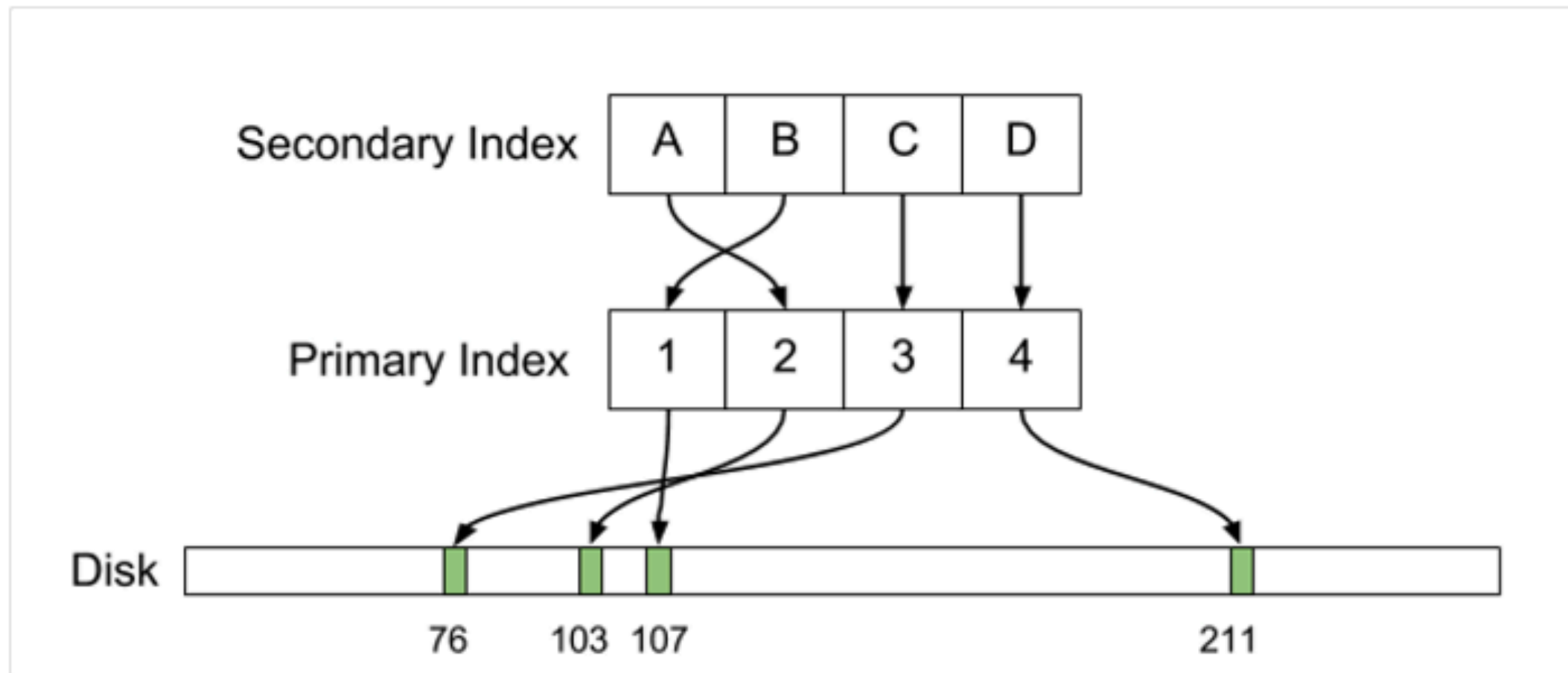
Christophe Pettus
PostgreSQL Experts, Inc.
PGDay Nordic 2017



WHY UBER ENGINEERING SWITCHED FROM POSTGRES TO MYSQL

JULY 26, 2016

BY EVAN KLITZKE

[f Facebook 9.5k](#)[t Twitter 562](#)[Y HN 402](#)[G+ Google+ 201](#)[in LinkedIn 2.4k](#)



You insulted my elephant.
Prepare to die.

What were the complaints?

- “Write Amplification.”
- “Replication.”
- “Bug in 9.2”
- “Replica MVCC”
- “Upgrades.”
- “Buffers.”
- “Connections.”

Ground Rules.

- There was plenty of speculation about "real" motives.
- We confine ourselves to what the technical paper actually said.
- We take them at their word that they experienced what they say they did.

**INSERT
MVCC
LECTURE
HERE**

Marshall

Marshall

Marshall

Marshall

Marshall

Write

Amplification

Marshall

Marshall

Marshall

Marshall

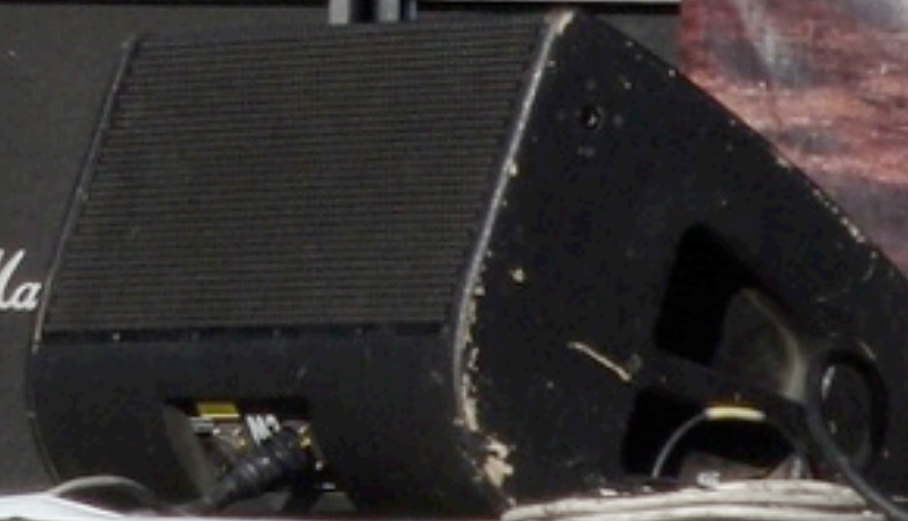
Marshall

Marshall

Marshall

Marshall

Ma



The Complaint.

- PostgreSQL's index implementation points directly at tuples on disk.
- Any change to a tuple means all indexes have to have a new entry added.
- One tuple write is then turned into many page writes, to update the indexes.

MySQL is better because...

- It uses two-level indexes for non-primary-keys.
- Key value -> primary key -> row.
- Updating a row only writes that one row.
- Indexes only need to be rewritten on primary key changes, and those are infrequent.

A close-up, high-contrast photograph of an elephant's face, focusing on the eye and the surrounding skin. The skin is heavily wrinkled and textured, with deep creases and ridges. The eye is partially visible, showing a dark, textured surface. The overall tone is dark and moody.

Elefact Says:

Half-True, Half-False

True:

- PostgreSQL must update every index if a change to the row updates an index.
- PostgreSQL keeps each version of the tuple on disk until it is vacuumed.
- Each page changed here must be pushed down the binary replication link.

But:

- Changes to non-indexed columns do not require an index update (HOT).
- “The Postgres autovacuum process has to do full table scans to identify deleted rows.”
- Not for years and years.

Missing:

- MySQL's design requires a special rollback area.
- Concurrency is hurt by having to reconstruct "old" database state.
- All non-PK index lookups require two separate index operations.
- Walking a large b-tree is not free.



Replication

The Complaint:

- PostgreSQL pushes every single page change down the binary replication link.
- This means that index changes, etc. are included in the replication stream.
- This creates very large bandwidth demands, especially over WAN links.

MySQL is better because...

- It only sends down logical changes.
- Index changes don't need to be pushed down.
- This is significantly more compact.

A close-up, grayscale photograph of an elephant's face, focusing on the eye and the surrounding wrinkled skin. The image is dark and has a high-contrast, almost monochromatic appearance. The text is overlaid in white, bold, sans-serif font.

Elefact Says:

Apples and Oranges

True:

- Until recently, PostgreSQL did not have logical replication in core.
- Existing logical replication tools (Slony, Bucardo, etc.) are somewhat fiddly to set up and manage.
- But... c'mon. *Uber?*

But:

- This compares MySQL logical replication to PostgreSQL's binary replication.
- PostgreSQL has had logical replication tools since pretty much ever.
- PostgreSQL 9.4+ has logical replication as a core feature.

A photograph of a road with a large sinkhole, with the text "Database Corruption" overlaid in white. The road is asphalt and has a white dashed line down the center. The sinkhole is a large, irregularly shaped hole in the road surface, revealing a dark, rocky interior. The surrounding area is a dry, hilly landscape with sparse, brownish vegetation. In the background, there are orange traffic cones and a fence line, suggesting a construction or maintenance site. The text "Database Corruption" is written in a large, bold, white sans-serif font, centered over the sinkhole.

Database Corruption

The Complaint:

- 9.2 had a data corruption bug around streaming replication.
- It was very unpleasant.
- "PostgreSQL had a bug, so we're switching to MySQL."

עַל (שֵׁשׁ) לְעַל

How to put this?

- Those bugs were very promptly fixed by the PostgreSQL project.
- I have used MySQL.
- I would not call MySQL bug free.
- Let's just leave it at that.



“Replica MVCC”

The Complaint:

- “Postgres does not have true replica MVCC support.”
- Incoming changes on the replication stream can either:
 - Delay replication.
 - Cancel queries.

MySQL is better because...

- It only sends down logical changes.
- Those changes are transactional just like any SQL operations.
- Queries are not blocked by incoming changes.

A close-up, high-contrast photograph of an elephant's face, focusing on the eye and the surrounding wrinkled skin. The skin is dark and heavily textured with deep, wavy ridges and grooves. The eye is partially visible, showing a dark, textured iris. The overall tone is dark and moody.

Elefact Says:

Apples and Oranges

True:

- Incoming streaming replication activity can be blocked by queries, or queries can be cancelled.
- Naïve users can be surprised by query cancellation messages.

But:

- This is configurable.
- You can have a “close” replica for failover and a “delayed” replica for queries.
- Again, we’re comparing logical replication to binary replication.
- Uber had a lot of long-running transactions because...

“While it’s always bad form to let your code hold open database transactions while performing unrelated blocking I/O, the reality is that most engineers are not database experts and may not always understand this problem, especially when using an ORM that obscures low-level details like open transactions.”

עַל (שֵׁי) עַל

And...

- Incoming SQL-level operations will take locks.
- Long-running transactions can block other sessions by holding these locks.
- Is this better or worse? Why? Uber doesn't say.

A close-up photograph of a computer keyboard. The central focus is a large, rectangular key with a vibrant red surface and rounded corners. The word "Upgrade" is printed on this key in a clean, white, sans-serif font, oriented diagonally from the bottom-left to the top-right. To the left of the red key is a white key with the letter "P" printed on it. Above the red key is a white key with an asterisk "*" and a plus sign "+" printed on it. The background shows other keys of the keyboard, which are slightly out of focus, creating a shallow depth of field. The lighting is soft and even, highlighting the texture of the keys.

The Complaint:

- PostgreSQL upgrades can require a lot of downtime.
- This is made worse if you have a large fleet of secondaries.
- pg_dump/pg_restore-style upgrades aren't practical for large databases.

MySQL is better because...

- You can use logical replication to upgrade one machine, replicate to it, and then fail over to it.
- The switchover is very fast.
- This sounds like a great idea! PostgreSQL should do it! Why don't we?

A close-up, high-contrast photograph of an elephant's face, focusing on the eye and the surrounding wrinkled skin. The image is in grayscale with a dark, moody tone. The skin shows deep, horizontal wrinkles on the left side and more intricate, circular patterns around the eye. The eye itself is partially visible, showing a dark, textured surface.

Elefact Says:

Half-True, Half-False.

True:

- PostgreSQL does not have in-place major version upgrade.
- You have to do some kind of process to get low-downtime upgrades.
- `pg_upgrade`, while a big improvement, is not a panacea.
- PostGIS, for example, is a huge pain.

But:

- Once again, PostgreSQL has had logical replication forever.
- You can do ***exactly the same process*** on PostgreSQL as MySQL.
- I assume the company the size of Uber can figure it out. C'mon.

A photograph of a swimming pool at night. The pool is illuminated, showing ripples in the water. In the background, there is a white fence and some outdoor furniture, including a table and chairs, some of which are covered with blue tarps. The text "Buffer Pools" is overlaid in the center of the image in a large, white, bold font.

Buffer Pools

The Complaint:

- PostgreSQL's buffering system relies heavily on the file system cache.
- Pulling things from file system cache, while faster than from disk, requires a context switch to the OS.
- This is bad.

MySQL is better because...

- It relies more on its own local cache.
- This means it can retrieve more data without context switches.
- This is just the best thing ever.

A close-up, high-contrast photograph of an elephant's eye and the surrounding skin. The skin is heavily wrinkled and textured, with deep creases and ridges. The eye is partially visible, showing the eyelid and the dark, moist surface of the eye. The lighting is dramatic, highlighting the intricate patterns of the skin.

Elefact Says:

Mostly True.

True:

- PostgreSQL's shared buffer management performance peaks at 8-32GB.
- [citation required]
- Larger shared_buffers than that (usually) mean diminishing returns.
- Retrieving things from file system cache is slower than from shared buffers.

But:

- It's not clear what the real-life performance impact of this is.
- (Uber didn't provide any in their paper.)
- General OLTP systems are not super-sensitive to `shared_buffers`.
- While it undoubtedly improves performance, it's just one of many things.

A close-up photograph of a network switch or patch panel. Several red Ethernet cables are plugged into the ports on the right side of the frame. In the background, there are many blue Ethernet cables bundled together. The overall scene is brightly lit with a blue color cast. The text 'Connection Management' is overlaid in the center in a large, white, sans-serif font.

Connection Management

The Complaint:

- PostgreSQL forks a new process for each connection.
- This results in high latency and RAM usage for each new connection.
- It's hard to scale PostgreSQL above a few hundred connections.

MySQL is better because...

- It uses threads instead of processes.
- Each new connection is much lighter-weight.
- This allows it to scale to many more connections.

A close-up, high-contrast photograph of an elephant's eye and the surrounding skin. The skin is heavily wrinkled and textured, with deep creases and ridges. The eye is partially visible, showing the eyelid and the dark, moist surface of the eye. The lighting is dramatic, highlighting the intricate patterns of the skin.

Elefact Says:

Mostly True.

True:

- The PostgreSQL forking model is not efficient for lots of connections, or fast connection cycling.
- While basic RAM statistics can be misleading, each backend does consume a notable amount of memory.

But:

- This conflates connection establishment with connection activity.
- The number of “hot” connections PostgreSQL and MySQL can handle are generally equivalent.
- The putative performance problem of PostgreSQL's context switching is, at best, speculative and not demonstrated.

And:

- pgbouncer exists to mitigate this exact problem.
- Admittedly, pgbouncer is not always a drop-in replacement.
- Uber even tried to use pgbouncer...

“However, we have had occasional application bugs in our backend services that caused them to open more active connections (usually ‘idle in transaction’ connections) than the services ought to be using, and these bugs have caused extended downtimes for us.”

עַל (שֵׁי) עַל



So, the dolphin wins this one.

True:

- Uber identified some real pain points with PostgreSQL.
- Some of the points are valid, and are the subject of active work by the project.
- Unquestionably, they were experiencing some headaches.

But:

- The consistent comparison of logical vs binary replication is maddening.
- Slony or Bucardo are fiddly, but...
- ... it beggars belief that an organization like Uber can't make them go.
- And we've had `pg_logical` for a while now.

A few last notes...

- There was remarkably little quantitative information in how PostgreSQL vs MySQL performed in their environment.
- They had already made a decision to move to a schema-less architecture.
- “MySQL handles our devs’ bugs better.”

עַל (שֵׁי) עַל

Questions?

Christophe Pettus
@xof

thebuild.com
pgexperts.com



Thank you!

Christophe Pettus
@xof

thebuild.com
pgexperts.com