A PostgreSQL fork for horizontal scalability: YugabyteDB

Franck Pachot, Developer Advocate



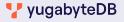


Developer Advocate at Yugabyte

Past:

20+ years in databases, dev and ops, consulting Oracle ACE Director, AWS Data Hero Oracle Certified Master, AWS Database Specialty











PostgreSQL is one of the most popular Open-Source database

- can run high throughput applications (if well-tuned)
- can run with good availability (failover with sync standbys)

Other are many extensions and forks to make it distributed:

- bi-directional replication: BDR/EDB PGD, pgactive, pgEdge
- single writer on distributed storage: Aurora, AlloyDB, Neon
- sharding: PostgresXL, Citus, Aurora Limitless
- distributed SQL: Spanner, CockroachDB, YugabyteDB, YDB

Why do they want to break the monolith?

3

Cloud can provides multiple data centers, multiple regions

- can run clusters with all nodes active (availability, latency, data governance)

Cloud resources are expensive if not used with elasticity

- want to scale CPU / RAM / IO independently, without downtime

Managed services are responsible for operations

- must operate without downtime (resilience to failures, rolling upgrade online)

On-premises cloud-native (Virtualization, Kubernetes) - infrastructure can scale horizontally if all pods are equal

4

You connect your application (ACID reads and writes) to one node You read and write from shared buffers (shared memory) The writes are protected from memory loss by one WAL (sync to disk) Database files can scale (NAS, SAN, EBS) but are not a bottleneck (written asynchronously)

What is monolithic:

- connection is handled by one stateful process that does everything
- RAM can be shared between processors of only one server (not though common network)
- WAL is only one sequential stream because of ACID transactions

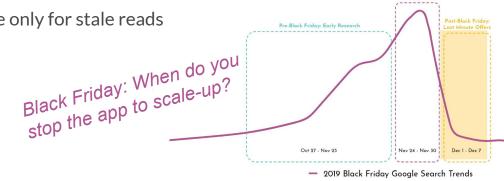
In case of failure of the single read/write node:

- all current sessions receive an error and must re-connect
- failover takes time (split-brain detection, cold cache, re-connections)
 Can be fast but still longer than the application timeout

In case of workload increase in the single read/write node



- scale-out to additional servers possible only for stale reads



Purchase Spike

To scale horizontally, we must split the database

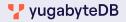
On top of multiple databases:

- by the application (more code, more tests)
- by a coordinator (Citus, Aurora limitless)

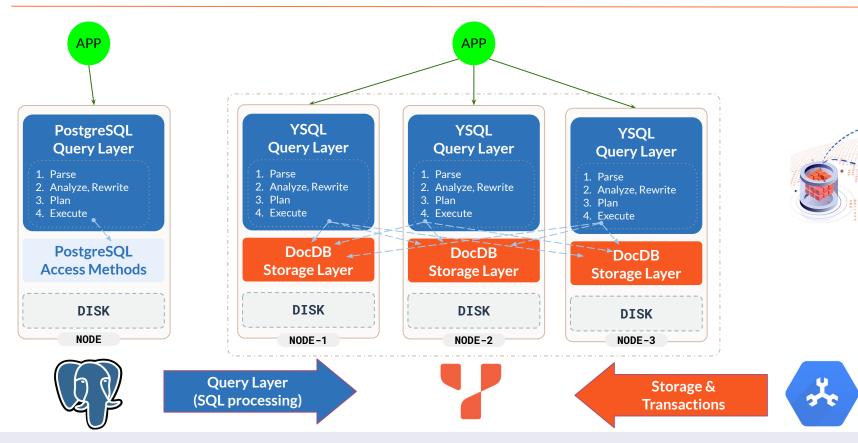
Local transactions (single-shard) are ACID but not global ones (Lack of read consistency, global constraints, unique keys, foreign keys)

Improves scalability and availability but still not fully resilient, and hard to re-shard when scaling-out

Alternative: sharding within the transactional storage (rows, index entries, transaction intents) **Clistributed SQL**



Monolithic PostgreSQL M Distributed YugabyteDB



Y yugabyteDB

A fork of PostgreSQL for the query layer

Started with PostgreSQL 10 Currently, based on PostgreSQL 11

psql (16.0, server 11.2-YB-2.20.1.1-b0)
yugabyte=# select * from version();

version

PostgreSQL 11.2-YB-2.20.1.1-b0 on aarch64-unknown-linux-gnu, compiled by clang version 16.0.6

Work In Progress: merging from PostgreSQL 15 https://github.com/yugabyte/yugabyte-db/tree/pg15/src/postgres

Goal: follow the latest PostgreSQL versions but with new features controlled by flags (to allow rolling upgrades)

Commits			
listory for yugabyte	db / src / postgres on pg15		
-o- Commits or	Feb 17, 2024		
[pg15] fix: aml	iguous column reference in REFRES and 3 days ago	H MV CONCURRENTLY and get	TestPgRegressMatview to pass
[pg15] fix: Pric	itise wholerow var of rel_type_id va red 3 days ago	type over RECORDOID vartype	in target list 🚥
-o- Commits of	Feb 15, 2024		
[pg15] style: re	move unused function 📼		
-o- Commits of	Feb 14, 2024		
[pg15] test: ge	TestPgRegressPublication to pass last week		
-> Commits or	Feb 13, 2024		
[pg15] fix: CRE	ATE/ALTER PUBLICATION grammar	ules m	
-o- Commits of	Feb 6, 2024		
Merge commi	'340212f084b2493b3ffa18911c5285 2 weeks ago	id860b3c6e' into pg15-merge	
-o- Commits o	Feb 2, 2024		
(pg15) fix: Incl	de non-updated columns in UPDA red 3 weeks ago	Es with secondary indexes or B	R triggers 🚥
	769e2a5bb1e574a6fe96cd5d001cf	86d8d2a7a7' into pg15 📟	

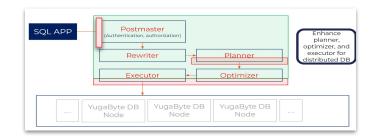
🏆 yugabyteDB

PostgreSQL extensibility is great:

- Foreign Data Wrapper
- Table/Index Access Methods
- Some hooks in the code for extensions

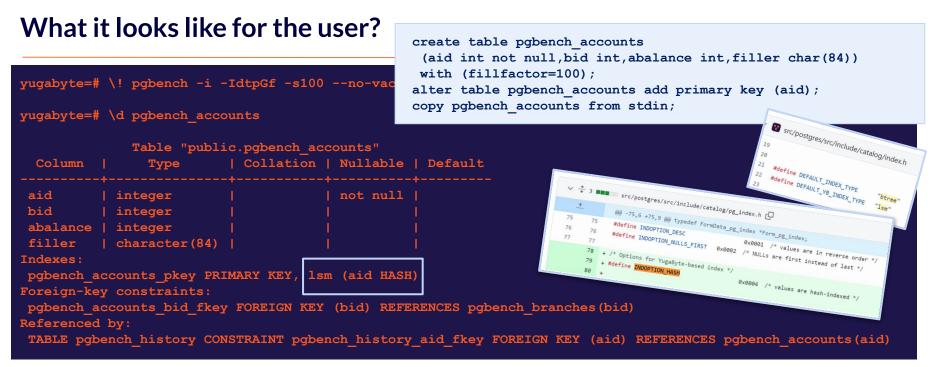
But also some limitations:

- expects Heap Tables
- expects tuples with TID, XID
- expects shared buffers (blocks)
- no hooks for WAL, syntax
- no threaded connections



YugabyteDB:

- pushdowns (FDW?), transaction intents (TableAM?)
- lot of batching (ex: Batched Nested Loop)
- additional syntax (ex: hash sharding)
- different cost model, different info in catalog
- threaded connections (connection manager)
- independent of system libraries (GLIBC)



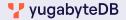
Primary key and Index use HASH for hash sharding, or ASC/DESC for range, Auto-split on size, or add split into 8 tablets, split at ((...),(...))

What it looks like for the system?

yugabyte=# \! pgbench -n -N -c 10 -T 60

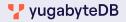
verview Tables	Nodes Backups Performance	Activity Mainter	nance Settings	Connect Actions ~
Nodes				
Primary Cluster	~ All Regions	~		
Name 🌲	Region 🔶	Zone 🌲	Memory Used 🌲	Read Write ops/sec 🌲
✓ zone-n1 🗐	IE Ireland (eu-west-1)	eu-west-1b	6 GB	80.23 41.01
✓ zone-n2 🗐	IE Ireland (eu-west-1)	eu-west-1a	6 GB	71.07 35.04
✓ zone-n3 🕞	IE Ireland (eu-west-1)	eu-west-1c	6 GB	74.06 37.63

Can connect to all nodes, Read and writes are balanced over all nodes



What it looks like for the application?

SPLIT AT VALUES is optional (auto-splitting when table grows) DDL is not (yet) transactional, but does not lock concurrent DML (optimistic locking for no downtime migrations)



What it looks like for developer?

yugabyte=# explain (analyze, dist, costs off, summary on)
 select from pgbench_accounts
 order by abalance asc fetch first 1000 rows only;

QUERY PLAN	
Limit (actual time=0.7973.102 rows=1000 loops=1) -> Index Only Scan using acc_bal on pgbench_accounts (a Heap Fetches: 0	actual time=0.7963.035 rows=1000 loops=1)
Storage Index Read Requests: 2 Storage Index Read Execution Time: 2.717 ms Planning Time: 0.067 ms	True Index Only Scan (no need to vacuum)
Execution Time: 3.173 ms Storage Read Requests: 2 Storage Read Execution Time: 2.717 ms Storage Write Requests: 0 Catalog Read Requests: 0 Catalog Write Requests: 0	Read / Write requests are - batched - parallelized between YSQL (PostgreSQL backend)
Storage Flush Requests: 0 Storage Execution Time: 2.717 ms Peak Memory Usage: 14 kB (15 rows) catalog is on yb-master (cluster metadata)	and DocDB (Distributed Storage and Transaction) Raft leaders in yb-tserver nodes

Y yugabyteDB

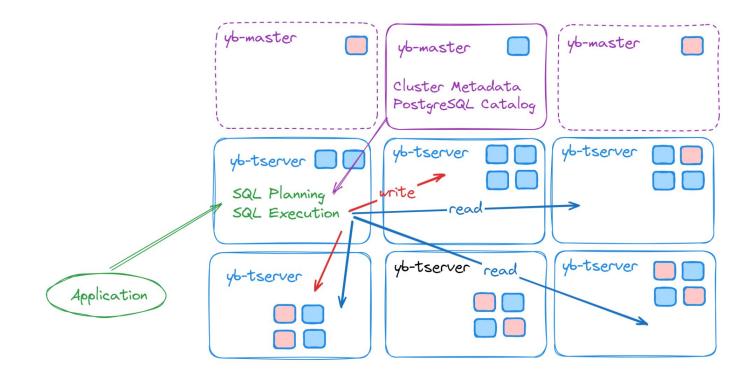
Joins can scale in Distributed SQL

yugabyte=# explain (analyze, dist, costs off, summary on) select count(aid) from pgbench history join pgbench accounts using (aid) where delta>0; OUERY PLAN Aggregate (actual time=27.323..27.323 rows=1 loops=1) YB Batched Nested Loop Join (actual time=17.209..27.205 rows=1657 loops=1) Join Filter: (pgbench history.aid = pgbench accounts.aid) -> Seq Scan on pgbench history (actual time=4.375..4.515 rows=1657 loops=1) Remote Filter: (delta > 0) Storage Table Read Requests: 1 Storage Table Read Execution Time: 4.281 ms -> Index Scan using pgbench_accounts_pkey on pgbench_accounts (actual time=10.269..10.565 rows=828 loops=2 Index Cond: (aid = ANY (ARRAY [pgbench history.aid, \$1, \$2, ..., \$1023])) Storage Table Read Requests: 1 Storage Table Read Execution Time: 9.653 ms Joining 10k rows 176ms Planning Time: 0.495 ms Execution Time: 27.604 ms **f** 50k rows joins second Storage Read Requests: 3 Storage Read Execution Time: 23.587 ms

Shards are LSM-Trees (RocksDB with read optimizations)

yugabyte=#	<pre>explain (analyze, dist, debug, co select distinct aid from pgbench_accounts order by aid</pre>	osts off, summary off)
;	QUERY PLAN	
-> Index Onl Heap Fe Storage Storage Metric Metric Metric	<pre>time=1.08873.485 rows=100000 loops=1) y Scan using pgbench_account_abal on pgbench_accounts ttches: 0 a Index Read Requests: 98 a Index Read Execution Time: 4.545 ms rocksdb_number_db_seek: 98.000 rocksdb_number_db_next: 100097.000 rocksdb_number_db_seek_found: 98.000 rocksdb_number_db_next_found: 100096.000</pre>	(actual time=1.08731.300 rows=100000 loops=1) RocksDBLSM-Tree - seek() to key (or key prefix) - next() to read row/column versions
Metric	rocksdb_iter_bytes_read: 4249346.000 docdb_keys_found: 100097.000 ql_read_latency: sum: 38190.000, count: 98.000	Rpc (network calls)

SQL execution, read/write to Tablets

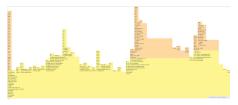




A fork of PostgreSQL for the query layer

	1159		
	1140 /* ess / 1141 * heap access method interface 1142 * 1143 */		
👷 jaki Merge commit '340212f084b2493b3ffa18911c5285dd860b3c6e' into pg15-merge	1144		
This branch is 159 commits ahead of, 868 commits behind master .	1147 heap_beginscan(Relation relation, Snapshot snapshot, 1148 1148 int nkeys, ScanKey key, 1149 1149 ParallelTableScanDesc parallel_scan, 1150		
Name	1151 (1152 HeapScanDesc scan;		
B	1153 1154 /* YB scan methods should only be used for tables that are handled by YugaByte. */ 1155 if (IsYBRelation(relation))		
🗅 Makefile	1156 { 1157 return ybc_heap_beginscan(relation, snapshot, nkeys, key, flags); 1158 } 1159 1160 /*		
🗅 yb_lsm.c			
b_pg_inherits_scan.c			
b_yb_scan.c	* 3. IndexScan(UserTable, Index)		
/*	* - Both target and bind descriptors are specifed by the IndexTable.		
	* - For this scan, YugaByte returns an index-tuple, which has a ybctid (ROWID) to be used for		
* Set up scan plan.	<pre>* querving data from the UserTable.</pre>		
* This function sets up target and bind columns for each type of scans.	 - TODO(neil) By batching ybctid and processing it on YugaByte for all index-scans, the target 		
* SELECT <target_columns> FROM <table> WHERE <binds></binds></table></target_columns>	 for index-scan on regular table should also be the table itself (relation). 		
*	* Tor Index-scan on regular table should also be the table itself (relation).		
* 1. SequentialScan(Table) and PrimaryIndexScan(Table): index = 0	* 4. IndexOnlyScan(Table, Index)		
 * - Table can be systable or usertable. 			
 YugaByte doesn't have a separate PrimaryIndexTable. It's a special case 	* - Table can be systable or usertable.		
 * - Both target and bind descriptors are specified by the <table></table> 	* - Both target and bind descriptors are specified by the indexiable.		
*	 For this scan, YugaByte ALWAYS return index-tuple, which is expected by Postgres layer. 		

YSQL (Query Layer)



https://share.firefox.dev/3I5HS3e

```
create table demo (
  k bigserial primary key
, v int default 0);
insert into demo
  select from generate_series
  (1,1000000);
\watch 0.001
```

Total (s			
	samples)	Self	
100%	53	_	_start /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	▼
100%	53	_	▼ main /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	Imain /nome/opc/yugabyte-2.19.3.0/postgres/bin/postgres PostgreSQL
100%	53	_	PostmasterMain /home/opc/yugapyte-2.19.3.0/postgres/bin/postgres
100%	53	-	ServerLoop /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	EackendRun /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	-	👿 🦲 PostgresMain /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	-	y_exec_query_wrapper_one_attempt /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	yb_exec_simple_query_impl /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	PortalRun /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	PortalRunMulti /home/opc/yuqabyte-2.19.3.0/postgres/bin/postgres
100%	53	_	ProcessQuery /home/opc/yugabyte 2.19.3.0/postgres/bin/postgres YugabyteDB
100%	53	_	Texplain ExecutorBun /home/onc/sugabute 2:10:2.0/postgras/lib/oute.evalain.co
100%	53	_	 Appling Execution and interformer support of the supp
100%	53	-	▼ pqss ExecutorRun /home/opc/yuqabyte-2.19.3.0/postgres/lib/pq_stat_statements.so
100%	53	_	standard ExecutorRun /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
100%	53	-	ExecModifyTable /home/opc/yugabyte-2.19.3.0/postgres/bin/postgres
53%	28	_	Fuerbreat / heme/anc/susabite 2.10.2.0/nestaves/hip/nestaves
47%	25	1	► VBCExecuteInsertInternal /home/opc/yugabyte=2.19.3.0/postgres/bin/postgres
3.8%	2	_	EverConstraints /home/one/sugabute 2 19 2 0/nestares/bin/nestares
1.9%	1	_	► Execcloscalaria / home/opc/ydgabyte-2.19.3.0/postgres/bin/postgres/
47%	25	_	ExecScan /home/opc/yugabyte-2.19.3.0/postares/bin/post
45%	24	_	ExecInterpExpr / home/op/yugab/te-2/19.30/ors/bin/postgres
45%	24	_	Inextval internal /home/opc/yugaby/ 19.3.0/postgres/bin/postgres
43%	23	_	VBCFetchSequenceTuple /nome/opc/yuqabyte-2.19.3.0/iib/yb/libyb_pqqate.so
43%	23	_	▼ b::pqqate::pqApiimpb::FetchSequenceTuple / home/opc/yqabte-2.19.3.0/lib/yb/libyb_pqqate.so
43%	23	_	V by by by a gate: PaClient: FetchSequenceTuple / home/opc/yugabyte-2.19.3.0/ib/vb/ib/by.bgqate.so
43%	23	1	Terror Structure: PgClientServiceProxy::FetchSequenceTuple //home/opc/yugabyte-2.19.3.0/lib/yb/libpg_client_proto.
42%	22	_	yb::rpc::Proxy::SyncRequest /home/opc/yuqabyte-2.19.3.0/lib/yb/libyrpc.so
40%	21	_	✓ yourpear tooyyretequeet /home/opec/yogabyte/so.ion//joynet.co.
26%	14	_	 ✓ ysalpediate / home/opc/ydapte / home/opc/ydapte/ei/selse ✓ ψscCountDownLatch:Wait / home/opc/ydapte/ei/selse
26%	14	_	▼ pthread_cond_wait@@GLIBC_2.17 /usr/lib64/libpthread-2.28.so
26%	14	-	▼ el0t.64 sync [kernelkallsyms]
26%	14	_	▼ elot_64_sync_handler [kernel.kallsyms] YugabyteDB
26%	14	_	V elo svc [kernel kallsyms]
25%	13	_	► eos (reinskalsyms) -> tserver client
1.9%	1	_	
13%	7	_	▼ yb::rp::/Poxy:DoAsyncRequest /home/opc/yugabyte-2.19.3.0/lib/yb/libyrpc.so
13%	7	_	▼ ■ yu.pu.rtox.rtox.boxsyncequest / mineropolyagab/e=1 = 32/miny/miny/miny/pcso
11%	6	_	▼ yu-pu-rtox-xsyntemotecaii / none/opb/ yugatytez-1/so/nib/yb/ib/pcs0
1.9%	1	1	 yb_apc_wresenger_cgeuecoutoducal /none/opc/yugapte_13.30/ib/yb/lib/b_
1.9%	1	1	yu.nacewayobearvia.cewpir /mone/op/yagayte2.15.3.0/moy/b/mb/yu.ims/ b/b/cautbownLate/pir /mone/opc/yagayte2.15.3.0/mb/yb/libyrp.so
			yb.countbownLatencountbownLaten@ptc=/none/opc/yugabyte=2.19.5.0/lb/yb/lbytpc.so

DocDB (Storage and Transactions)

All frames	Instants	C. Martin		Film make O rockedb	
All frames	JavaScript	Native	invert call stack	Filter stacks: Q rocksdb	
mplete "rpc_tp	_TabletSe"				
Total (sar	mples)	Self			
100%	11	_	Thread start /usr/lib64/libc-2.28.so		
100%	11	-	▼ start thread /usr/lib64/libbthread-2.28.so	tserver tablet	
100%	11	_	✓ dur_thread:/bu/how/serviseThread /home/opc/yugabyte-2.19.3.0/bin/yb-server	tservertablet	
100%	11	-	▼ yourpe://anonymous.namespace/Worker:Secute / home/opc/yourpe://anonymous.namespace/Worker:Secute / home/opc/yourpe://anonymous.namespace/Worker:		
100%	11	_	V bp:pc:/hboundCall:InboundCallTask:Run /home/opc/yugabyte-2.19.3.0/bin/yb-server		
100%	11	_	Vicroc:ServicePoolImpl:Handle /home/opc/yugabyte-2.19.3.0/bin/yb-server		
100%	11	-	V vb:tserver:TabletServerServicelf:Handle /home/opc/yuqabyte-2.19.3.0/bin/yb-server		
100%	11	-	🗸 dt:1:function:func.vp::tserver::TabletServerServicelf::InitMethods(scoped_refptr <yb::metricentity> const&)::\$_0, std:1::allocator<yb::tserver::tabletserverservice< td=""><td>elf-InitMethods(</td><td>(std:: 1::sh</td></yb::tserver::tabletserverservice<></yb::metricentity>	elf-InitMethods((std:: 1::sh
100%	11	-	▼ ↓b::tserver:TabletServiceImpl:Write /home/opc//uqabyte-2.19.3.0/bin/yb-server		(***** <u>-</u>
100%	11	-	▼ yb:tablet::TabletPeer:WriteAsync /home/opc/yugabyte-2.19.3.0/bin/yb-server	DocDB write	
100%	11	-	yb::tablet:Tablet:Applict.acduireLocksAndPerformDocOperations /home/opc/yugabyte-2.19.3.0/bin/yb-server	DOCDB write	
100%	11	-	▼ yb::docdb::(anonymous namespace)::WaitOnConflictResolver::Run /home/opc/yugabyte-2.19.3.0/bin/yb-server		
100%	11	_	b::docdb::(anonymous namespace)::ConflictResolver::Resolve /home/opc/yugabyte-2.19.3.0/bin/yb-server		
55%	6	-	V upp://www.execution.com/www.execution.com/interestiver://www.callback //home/opc/yugabyte-3-3-0-0-yup-server		
55%	6	_	Isocation and the second se	pc/vugabyte-2.19.3.0/bin/vb-server	
55%	6		▼ Up:tablet:WriteQuery::DoCompleteExecute Anome/opc/yugabyte-2.19.3.0/bin/yb-server		
55%	6	_	▼ yb::do:db::PqsqlWriteOperation::Apply /home/opc/yuqabyte-2.19.3.0/bin/yb-server		
55%	6	_	yb::docdb:PqsqlWriteOperation::ReadColumns /home/opc/yugabyte-2.19.3.0/bin/yb-server		
36%	4	-	▼ bb:docdb::http://www.etterator::Sec.//home/opc/yugabyte-2.19.3//bin/yb-server		
36%	4	-	Jb::docdb::PerformRocksDBSeek /home/opc/yugabyte-2.19.3.0/bin/yb-server		
36%	4	_	yb:docdb::BoundedRocksDbiterator:Seek //home/opc/yugabyte-2.19.3.0/bin/yb-server		
36%	4	-	▼ focksdb::DBIter:Seek /home/opc/yuqabyte-19J3/bin/yb-server		
27%	3	_	▼ rocksdb://demTable/terator:/Seek -/bome/opc/vuab/tre-2.19.3.0/bin/yb-server		
27%	3	_	Incksdb:(anonymous namespace):SkipLister	"Iterator: Seek /home/onc/vugabyte-2 19 3 0/bin/vb-server	
27%	3	3	rocksdb::MemTable::KeyComparator::operator() //e/opc/yugabyte-2.19.3.0/bin/yb-server	interationiseen //forme/op-cycgas/te-Erisis/orbit/js-serier	
9.1%	1	1	rocksdb:StatisticsMetricImpErecordTick /home/opc/yugabyte-z0/bin/yb-server		
18%	2	_	vb:docdb:IntentAwareIterator:SeekForward /home/opc/yugabyte-2.19.3.0/bin/yb-sec.		
18%	2	-	▼ yb:docdb:intentAwareIterator:intentSeekForward /home/opc/yugabyte 2:193.0/bin/yb-serve-		
18%	2	_	V Systacdb::(anonymous namespace):SeekPossiblyUsingNext /home/opc/yugabyte-2.13.0/bin/yb-serv		
18%	2	_	▼ ybs/docdb::BoundedRocks/Dbitrator.Seek /home/opc/yugabyte-2.19.30/bin/yb-server		
18%	2	_	▼ rocksdb::DBiter::Seek /home/opc/yugabyte=2.19.30/bin/yb-server		
18%	2	_	▼ rocksdb::MemTableIterator:Seek /home/opc/yugabyte-2.19.3.0/bin/yb-server		
18%	2	_	Tocksdb: (anonymous namespace): SkipListRep <rocksdb: const&:<="" memtablerep:keycomparator="" p="" rocksdb:="" singlewriterinlineskiplist<=""></rocksdb:>	>::Itera. /bome/opc/yugabyte-2.19.3.0/bin/yb-serve	ar
18%	2	1	▼ Tocksdo: denotingen to the start of th	anieroperjuguojee e. 15.5.0/bill/yb/serve	
9.1%	1	1	 rocksdb:(anonymous namespace):SyltewiseComparatorinpl:Compare /home/opc/yugabyte-2.19.3.0/bin/yb-server 		
45%	5	_	v bybdocdb:(anonymous) namespace/j transactionConflictResolverContext:ReadConflicts /home/ocy/ugabyte=1:53.0bin/g/server		
45%	5		 y Succession and y most micespace): and succession an	RocksDB	
45%	5	-	 gu.docdu.gunonymous namespace.commic.teesover.ceesomer.ceeso ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceeesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.ceesomer.	RUCKSDD	
45%	5		 by update and you can be an an appaced access of solar group of you good and you can be an appaced access of your group of you good and you can be an appaced access of the appaced acces		
45%	5		▼ rocksdb::DBIter:Seek /home/opc/yugubte-2:19:3.0/bit/v5-server		
45%	5	_	▼ Tocksdb://wmTabletractor:Seek / home/opc/yugabyte=2:19:30/bin/yb-Server		
45%	5	3	▼ Incksdb://anonymous.memspace/SkiplisRep-rocksdb:SingleWriterInlineSkipList <rocksdb: wemtablerep:keycomparator.const&=""> >::terator</rocksdb:>	Esterior de la construcción de l	
4070	-	3	 Tocksub. (anonymous namespace)skipcisticp stocksubsinglewriterinineskipcist stocksubweimablekepkeycomparator constact >terator 	index /none/opc/yagabyte-2.13.5.0/blit/yb*server	

DocDB (Raft replication)

Call Tree			Stack Chart Marker Chart Marker Table	
All frames	JavaScript	Native	Invert call stack	
Complete "c	onsensus [work*	-		
	(samples)	Self		Update Raft Log
100%	710		thread start /usr/lib64/libc-2.28.so	Opuale Kall Lug
100%	710	_	▼ start_thread /usr/lib64/libpthread-2.28.so	
100%	710	-	vb:Thread::SuperviseThread /home/opc/yugabyte-2.19.3.0/bin/yb-server	
92%	655	-	view with the second se	
63%	448	-	Vb:rpc:InboundCall:InboundCallTask:Run /home/opc/yugabyte-2.19.3.0/bin/yb-server	
63%	446	_	V vb::rpc::ServicePoolImpl::Handle /home/opc/yugabyte-2.19.3.0/bin/yb-server	
63%	444	2	V::consensus:-Consensus:Servicelf::Handle /home/opc/yugabyte-2.19.3.0/bin/yb-server	
62%	440	_	std:1:function:func <yb::consensus:consensus:serviceif::initmethod: _coped_refptr<yb::metricentity=""> const&)::\$_0, std:1</yb::consensus:consensus:serviceif::initmethod:>	
57%	404	4	yb::tserver::ConsensusServiceImpl::UpdateConsensus /home/opc/yuqabyte-2.19.3.0/bin/yb-server	
41%	292	6	Update // home/opc/yugabyte-2.19.3.0/bin/yb-server	
30%	212	4	v::consensus::RaftConsensus::UpdateReplica /home/opc/yugabyte-2.19.3.0/bin/yb-server	
13%	93	-	vb:tablet:TabletPeer::SetPropagatedSafeTime /home/opc/yugabyte-2.19.3.0/bin/yb-server	
13%	90	2	vb:tablet::OperationDriver::ExecuteAsync /home/opc/yugabyte-2.19.3.0/bin/yb-server	
12%	86	1	vb:ThreadPoolToken::SubmitFunc /home/opc/yugabyte-2.19.3.0/bin/yb-server	
12%	85	2	▼ yb::ThreadPool::DoSubmit /home/opc/yugabyte-2.19.3.0/bin/yb-server	
11%	78	_	Thread_cond_signal@@GLIBC_2.17 /usr/lib64/libpthread-2.28.so	
1196	78	-	elot_64_sync [kernel.kallsyms]	
11%	78	-	elot_64_sync_handler [kernel.kallsyms]	
11%	78	-	Verification [kernel.kallsyms]	
8.6%	61	-	▶ do_el0_svc [kernel.kalisyms]	
2.4%	17	_	do_notify_resume [kernel.kallsyms]	
0.7%	5	_	Up::ThreadPool::CreateThreadUnlocked /home/opc/yugabyte-2.19.3.0/bin/yb-server	
0.3%	2	-	Jb::tablet::OperationDriver::PrepareAndStart /home/opc/yugabyte-2.19.3.0/bin/yb-server	
0.3%	2	1	yb:tablet::TabletPeer::NewOperationDriver /home/opc/yugabyte-2.19.3.0/bin/yb-server	
0.1%	1	1	yb:tablet::Table_type /home/opc/yugabyte-2.19.3.0/bin/yb-server	
5.2%	37	-	Junious and AdvanceCommittedOpIdUnlocked /home/opc/yugabyte-2.19.3.0/bin/yb-server	
5.2%	37	_	vb::consensus::ReplicaState::ApplyPendingOperationsUnlocked /home/opc/yugabyte-2.19.3.0/bin/yb-server	
4.9%	35	_	United and the service of the ser	
4.9%	35	1	yb::tablet::OperationDriver::ReplicationFinished /home/opc/yugabyte-2.19.3.0/bin/yb-server	
4.5%	32	1	▼ yb:tablet::Operation::Replicated /home/ope/yagubyte=2/00/07/1/j1	
4.2%	30	_	▼ yb:tablet::WriteOperation::DoReplicated /home/opc/yugabyte-2.19.3.0/bin/yb-server	Commit write
4.1%	29	_	ub:tablet::Tablet::ApplyRowOperations /home/opc/yugabyte-2.19.3.0/bin/yb-server	Commit write
4.1%	29	2	▼ yb:tablet::Tablet::ApplyOperation /home/opc/yugabyte-2.19.3.0/bin/yb-server	
2.8%	20	_	vb:tablet::Tablet::WriteTransactionalBatch /home/opc/yugabyte-2.19.3.0/bin/yb-server	

Tablets are split on range of ASC/DESC primary key or range of HASH.

Automatically split when growing

Each tablet has IntentsDB (provisional records, committed or not) and RegularDB (committed only) LSM-Trees (RocksDB).

Tablets are Raft group (read/write to leaders)

Each RocksDB has one writable MemTable, flushed to immutable SST Files + Level 0 universal compaction Compacted to reduce space and read amplification

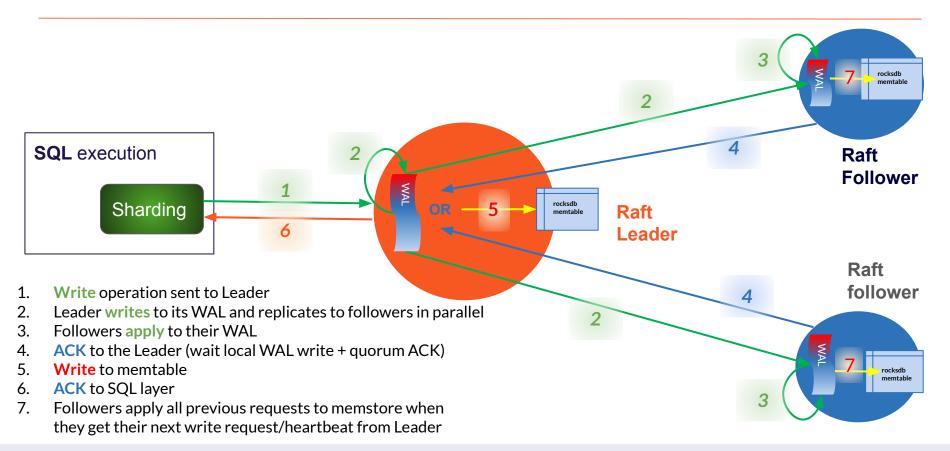


https://dev.to/yugabyte/testing-lsm-tree-merge-for-size-amplification-in-yugabytedb-2kh9



https://dev.to/yugabyte/yugabytedb-auto-sharding-2ahc

Raft replication

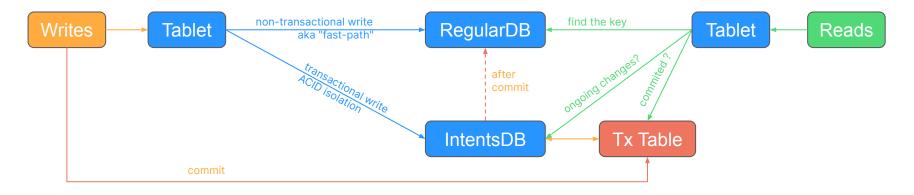


Y yugabyteDB

Tablets and data distribution

The key/value read and writes are per tablet (after sharding):

- Table rows and Index entries are distributed to tablets (by hash or range)
- Tablets are replicated to tablet peers (Raft groups)
- tablet peers have an IntentsDB to support transactional writes
- tablet peers store their committed versions in RegularDB
- compaction removes unnecessary intermediate versions (MVCC)





Advantage of a new storage layer

Beyond horizontal scalability - no vacuum problems 😎

- No bloat: MVCC version are stored per key, removed by RocksDB compaction
- Real Index Only Scans: indexes entries are versioned like tables
- Fast primary key access: index organized tables
- No transaction ID wraparound: clusterwide Hybrid Logical Clock ttimestamp
- transparent compression and encryption

Other advantages 😎

- Connection Manager: a threaded resident connection pool
- pg_hint_plan installed by default
- no downtime upgrades (rolling upgrades)
- rolling restart (online parameter change, key rotation, operating system patching

Reason for Distributed SQL

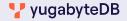
- scale-out for elasticity and resilience
- rolling upgrades, geo-distribution

Reason for PostgreSQL compatibility

- no need to learn a new DB, many SQL features Reason for a **fork**
 - the best compatibility with advanced features

Major difference with PostgreSQL:

- think more about the primary key (hash/range sharding)
- leverage new access patterns
 (primary index, skip scan, index only scan on secondary index)
- understand throughput vs. latency



When it can be an alternative to PostgreSQL:

- 24/7 system of records (OLTP)
- cloud native (Compute instances, kubernetes pods,... all active)
- multitenant (no hardware limitation to scale)
- geo-distribution (data residency, latency)

Not for datawarehouse (only some pushdowns for analytics)



E-mail:

fpachot@yugabyte.com

Blogs: dev.to/FranckPachot blog.yugabyte.com/author/fpachot

Twitter: @FranckPachot

Youtube: youtube.pachot.net

Twitch: www.twitch.tv/franckpachot

LinkedIn:

www.linkedin.com/in/franckpachot

