Yandex.Mail success story

Vladimir Borodin, DBA
About Yandex

› One of the largest internet companies in Europe
› 57+% of all search traffic in Russia
› Ukraine, Kazakhstan, Belarus and Turkey
› [https://yandex.com/company/technologies](https://yandex.com/company/technologies)
› About 6000 employees all over the world
About Yandex.Mail

› Launched in 2000
› 10+ million users daily
› 200,000 RPS to web/mobile/imap backends
› 150+ million incoming letters daily
› 20+ PB of data
About this talk

› Migration from Oracle to PostgreSQL
› 300+ TB of metadata without redundancy
› 250k requests per second
› OLTP with 80% reads, 20% writes

Previous attempts

› MySQL
› Self-written DBMS
What is mail metadata?
[HACKERS] what to revert

Noah Misch  noah@leadboat.com
To you and:  Kevin Grittner
Cc:  Tom Lane  Tomas Vondra  Andres Freund  pgsqi-hackers@postgresql.org

I discourage focusing on the statistical significance, because the hypothesis in question ("Applying revert.patch to 4bbc1a7e decreases 'pgbench -S -M prepared -j N -c N' tps by 0.46%.") is already an unreliable proxy for anything we care about. PostgreSQL performance variation due to incidental, ephemeral binary layout motion is roughly +/-5%. Assuming perfect confidence that 4bbc1a7e+revert.patch is 0.46% slower than 4bbc1a7e, the long-term effect of revert.patch could be anywhere from -5% to +4%.

If one wishes to make benchmark-driven decisions about single-digit performance changes, one must control for binary layout effects:
http://www.postgresql.org/message-id/87ybitb2zp.fsf@news-spur.riddles.org.uk
http://www.postgresql.org/message-id/20160416204452.GA1910190@tornado.leadboat.com

nm
Back in 2012
Yandex.Mail metadata

- Everything stored in Oracle
- Lots of PL/SQL logic
- Efficient hardware usage
  - 10+ TB per shard
    - Working LA 100
- Lots of manual operations
- Warm (SSD) and cold (SATA) databases for different users
  - 75% SSD, 25% SATA
Sharding and fault tolerance

BlackBox

Auth info

UserID, ShardName

Backend

OCCI
/etc/tnsnames.ora

DC1

Shard1 Standby
Shard1 Primary
Shard2 Primary

DC2

Shard1 Primary
Shard2 Standby
Inside the backend

C++ backend → macs → mcs_ora → dbpool → OCCI
Reality

Java backend → C++ backend → C++ backend → C++ backend → macs → macs_ora → dbpool → OCCl
Most common problems

› PL/SQL deploy
  
  Library cache

› Lots of manual operations
  
  Switchover, new DB setup, data transfer between shards

› Only synchronous interface in OCCI

› Problems with development environments

› Not very responsive support
The main reason

shop.oracle.com
Timeline
Experiments

› Oct 2012 — the willful decision
  Get rid of Oracle in 3 years

› Apr 2013 — first experiments with different DBMS
  PostgreSQL
  Lots on NoSQL stores
  Self-written solution on base of search backend

› Jul 2013 — Jun 2014 — collectors experiment
About collectors

Get all your mail instantly

Read all your messages from other accounts in Yandex.Mail. You can reply to messages using the same address to which they were sent, so your contacts won't even notice the difference.

Email

Password

Copy messages along with folders

Connect mailbox  Back to list of mailboxes

All information entered here will be securely encrypted.
Experiment with collectors

- https://simply.name/video-pg-meetup-yandex.html
- Our first experience with PostgreSQL
  - Monitoring/graphs/deploy
  - PL/Proxy for sharding
  - Self-written tools for switchovers and read-only degradation
  - Plenty of initial problems
- 2 TB of metadata (15+ billion records)
- 40k RPS
Full mail prototype

› Aug 2014 — Dec 2014

› Storing all production stream of letters to PostgreSQL
  Asynchronously

› Initial schema decisions
  Important for abstraction library

› Load testing with our workload
  Choosing hardware

› Lots of other PostgreSQL related experience

https://simply.name/postgresql-and-systemtap.html
Main work

- Jan 2015 — Jan 2016 — development
- Jun 2015 — dog fooding
  - Accelerated development
- Sep 2015 — start of inactive users migration
  - Fixing bugs of transfer code
  - Reverse transfer (plan B)
- Jan 2016 — Apr 2016 — migration
Time to rewrite all software to support Oracle and PostgreSQL

10 man-years
Migration
Completion

Feature complete

95% complete

100%

All users migration

Registration

Dec 2016

Jan

Apr

May

Jul

Time
Main changes
macs

Java backend -> C++ backend -> C++ backend -> C++ backend

macs

macs_ora -> dbpool -> OCCI

macs_pg -> apq

libpq

Oracle

PostgreSQL
Sharding and fault tolerance

**Diagram:**
- **BlackBox**
  - Inputs: Auth info, UserID, ShardName
- **Backend**
  - Inputs: UserID, mode, Connection string(s)
- **Sharpei**
  - Inputs: UserID, ShardName
- **Shards:**
  - DC1: Shard1 Primary, Shard2 Primary, Shard1 Standby
  - DC2: Shard1 Primary, Shard2 Standby
  - DC3: Shard1 Standby, Shard2 Standby

Connections:
- BlackBox to Backend
- Backend to Sharpei
- Sharpei to DC1, DC2, DC3
- Authentication information flow
- UserID, ShardName flow
- UserID, mode flow
- Connection string(s) flow
Hardware

Workload

Oracle

Primary

Standby

PostgreSQL

Primary

Standby

Standby
Hardware

› Warm DBs (SSD) for most active users

› Cold DBs (SATA) for all inactive users

› Hot DBs for super active users
  2% of users generate 50% of workload

› Automation to move users between different shard types

› TBD: moving old letters of one user from SSD to SATA
Identifiers

In Oracle all IDs (mid, fid, lid, tid) were globally unique

  › Sequences ranges for every shard in special DB
  › NUMBER(20, 0) — 20 bytes

In PostgreSQL IDs are unique inside particular user

  › Globally unique mid changed to globally unique (uid, mid)
  › Biginit + bigint — 16 bytes
Schema changes

› Less contention for single index page
  Normal B-Tree instead of reversed indexes

› Revisions for all objects
  Ability to read only actual data from standbys
  Incremental diffs for IMAP and mobile apps

› Denormalized some data
  Arrays and GIN
  Composite types
Example

xdb01g/maildb M # "dS mail.box"

Table "mail.box"

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>bigint</td>
<td>not null</td>
</tr>
<tr>
<td>mid</td>
<td>bigint</td>
<td>not null</td>
</tr>
<tr>
<td>lids</td>
<td>integer[]</td>
<td>not null</td>
</tr>
</tbody>
</table>

Indexes:

- "pk_box" PRIMARY KEY, btree (uid, mid)
- "i_box_uid_lids" gin (mail.ulids(uid, lids)) WITH (fastupdate=off)

xdb01g/maildb M #
Stored logic

- PL/pgSQL is awesome
- Greatly reduced code size
  - Only to ensure data consistency
- Greatly increased test coverage
  - The cost of failure is high
- Easy deploy since no library cache locks
Maintenance approach

- SaltStack
  - Detailed diff between current and desired state
- All schema and code changes through migrations
- All common tasks are automated
- Representative testing environments
Problems
Before main migration

› Problem with ExclusiveLock on inserts
› Checkpoint distribution
› ExclusiveLock on extension of relation with huge shared_buffers
› Hanging startup process on the replica after vacuuming on master
› Replication slots and isolation levels
› Segfault in BackendIdGetTransactionIds
› A lot more solved without community help
In any unclear situation autovacuum is to blame

Oracle DBA
Diagnostics

- https://simply.name/pg-stat-wait.html
- `Wait_event in pg_stat_activity` (9.6)

- https://simply.name/ru/slides-pgday2015.html (RUS)
Backups

› Our retention policy is 7 days

› In Oracle backups (inc0 + 6 * inc1) and archive logs ≈ DB size

› In PostgreSQL with barman ≈ N* DB size, where N > 5

  WALs compressed but backups not

  File-level increments don’t work properly

  All operations are single-threaded and very slow

› For 300 TB we needed ≈ 2 PB for backups

› [https://github.com/2ndquadrant-it/barman/issues/21](https://github.com/2ndquadrant-it/barman/issues/21)
During migration

› Not PostgreSQL problems
› Data problems
  A lot of legacy for 10+ years
  Bugs in transfer code
Conclusion
Our wishlist for PostgreSQL

› Declarative partitioning
› Good recovery manager
   Parallelism/compression/page-level increments
   Partial online recovery (i.e. single table)
› Future development of wait interface
› Huge shared buffers, O_DIRECT and async I/O
› Quorum commit
Summary

› 1 PB with redundancy (100+ billion records)
› 250k TPS
› Three calendar years / 10+ man-years
› Faster deployment / more efficient human time usage
› All backend refactoring
› 3x more hardware
› No major fuckups yet :)
› Linux, nginx, postfix, PostgreSQL
Questions?

Vladimir Borodin
DBA

+7 (495) 739 70 00, ext.: 7255
@man_brain
d0uble@yandex-team.ru
https://simply.name