EnterpriseDB®
The Enterprise PostgreSQL Company

Maintaining Very Large Databases

Devrim GÜNDÜZ
@DevrimGunduz
Principal Systems Engineer
EnterpriseDB
devrim.gunduz@EnterpriseDB.com

This guy...

- Who is this guy?
  - I have been contributing to PostgreSQL over the last 10 years, and using it over the last 13 years.
  - I'm not a hacker, I work on PostgreSQL Community RPMs and rarely on website.
  - I rarely break RPMs, but break website more often.
  - Working at EnterpriseDB right now. Dream job, dream company.
  - Live in Istanbul, Turkey.
  - Have a 5-year old son, who likes Slipknot.

Warning!

50-min talk, cannot talk about everything in deep details.

Before we start

- Please note that:
  - I will not claim that PostgreSQL can handle VLDs without any problems.
  - PostgreSQL can do some wonderful stuff, but there are also some stuff that needs improvements.
  - OTOH, there are people out there who are running VLDs.

Agenda

- Defining Very Large Database (VLD) Concept
- Which version to use? (Obvious, isn't it?)
- Designing VLDs
- Choosing the right hardware
- Backups
- Maintenance
- Performance (load balancing, replication, clustering, HA, multimaster, etc)
- Upgrading
- Monitoring
- Questions

Agenda

Defining Very Large Database (VLD) Concept

Defining Very Large Database (VLD) Concept

- No common unit .
- Usually databases > 1 TB, or databases having billions and billions of rows are called VLD.
    - Sometimes it is 100 TB, 250 TB, 500 TB, or 1 PB.
- Please note that everyone has a big data, and/or everyone's data is critical data.

Agenda

Which PostgreSQL version to use?

Which PostgreSQL version to use

- 9.1 is a solid version, but it has some known scaling problems.
- 9.2 development period, and performance patches
  - A credit to Robert Haas and several other hackers goes here.
- Is 9.2 production ready?
  - Answers may vary, but it is really the most suitable version from the performance point of view, depending on the architecture.
- So use 9.1 or above.

Agenda

Designing VLDs

Designing VLDs – General overview

- Key to **the** success
- If you know that your database will be a VLD, start designing in day 0.
- A common mistake: “Let's start with a basic design first, we can scale it later on”
- No one will want to wear a 10XL t-shirt, but on the other hand make sure that you can extend your t-shirt to 10XL, when needed.
- Naming standards in the objects.
- Documentation!

Designing VLDs - Normalization

- Normalization
  - Quote: “SQL was created to work with normalized data structures”
  - A rule of thumb: "Normalize until it hurts, denormalize till it works"
  - Start the normalization during the design state
  - More will come later

Designing VLDs - Partitioning

- Partitioning!
  - Using a single table for Big Data is just insane.
  - Finding the right key
  - Increased availability
  - Integrates with hardware well

Designing VLDs - Partitioning

- Partitioning!
  - Number of partitions are important!
  - We need better partitioning syntax, and handling (should I say “advanced”?).
  - Of course, there are some caveats: http://www.postgresql.org/docs/devel/static/ddl-pa
  - PL/Proxy is also an option.

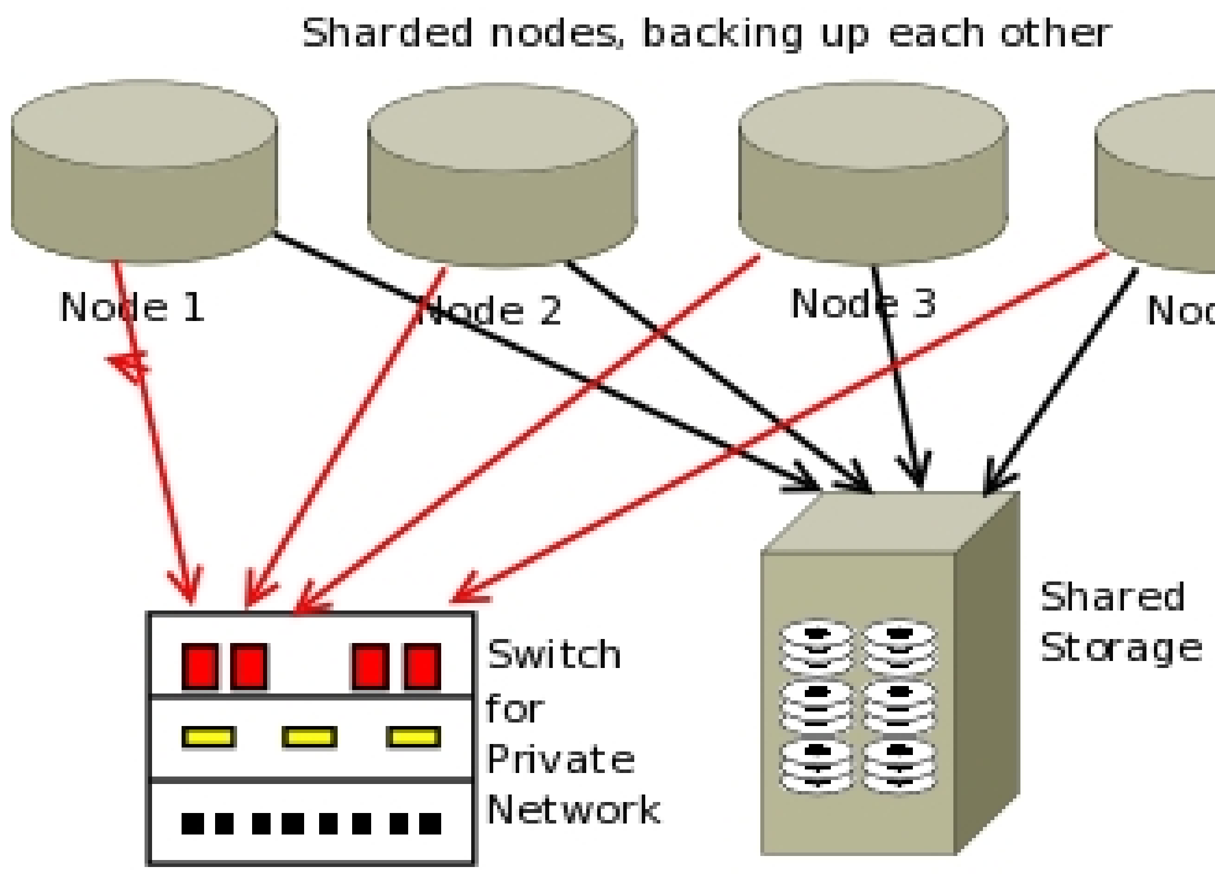
Designing VLDs- Using Stored Procedures

- Using stored procedures
    - This is more a performance tip, but this is what you need for VLDs.
    - Decrease query execution times with PL/pgSQL and its friends.

Designing VLDs - Sharding

- Sharding!
  - One of my favorites!
  - May require a rewrite of application
  - No additional software needed.
  - Helps you to add servers whenever you want.
  - Helps scaling the system horizontally with cheaper servers.
  - Also helps Red Hat Cluster Suite integration, increasing HA.

Designing VLDs - Sharding


Sharded nodes, backing up each other
Node 1
Node 2
Node 3
Node 4
Switch for Private Network
Shared Storage

Agenda

Choosing the right hardware

Choosing the right hardware

- Critical for performance
- I/O is what you need
- CPU: Faster CPUs, more CPUs
- Disks: SSDs vs SAS disks.
- Memory: Fast ones.
- Decent fiber channel network between the machines.
- Check pgsql-performance archives for more details.

Agenda

Backups

Backups

- One of the hardest issues
- Backups will take forever
- No need to mention about testing them
- pg_dump is rarely an option
- xlog archiving is probably what you need
- Did someone talk about disk space requirements?

Agenda

Maintenance

Maintenance

- VACUUM/ANALYZE
  - It will take forever to vacuum or analyze all the database.
    - Who needs manual vacuuming? (rant)
  - Tuning autovacuum is crucial.

Agenda

Performance (load balancing, replication, clustering, HA, multimaster, etc)

Performance

- We need parallel queries.
- Index-only scans in 9.2 may help for many users.
- Unlogged tables?
- COPY improvements in 9.2 (=faster loading of data)
- 9.2 also introduces DROP INDEX CONCURRENTLY (bugs, bugs... Wait for 9.2.3).
- Tablespaces

Performance

- Use replication for distributing read-only workload, and reporting queries.
- <rant>Multimaster!</rant>
- Postgres-XC 1.0.1 is out, and some people use it in production already.
- We need a new load balancer (Josh Berkus) http://www.databasesoup.com/2012/03/postgresql-n

Agenda

Upgrading

Upgrades

- Database upgrades
  - Not a big issue anymore, thanks to pg_upgrade
  - Depending on your environment, Slony and co are also out there, but that will require extra disk space.
  - pg_dump & pg_restore are probably not your friends here.
  - Pre-testing of your application on the new PostgreSQL version is more critical.

Upgrades

- OS upgrades
  - Might be easier
  - Nowadays, popular OSes have packages for all versions, so staying at the same PostgreSQL version should not hurt.

Agenda

Monitoring

Monitoring

- Another topic that needs a few extra person/day.
- Each relation needs separate checks
  - “One autovacuum check for all database” **will** fail at some point
- check_postgres, Nagios, MRTG, PEM and other monitoring tools are available.
- “Work smart, not hard”

Agenda

Questions?

Questions

–

Questions, please.

EnterpriseDB®
The Enterprise PostgreSQL Company

Maintaining Very Large Databases

Devrim GÜNDÜZ
@DevrimGunduz
Principal Systems Engineer
EnterpriseDB
devrim.gunduz@EnterpriseDB.com