Postgres 9.4: Major Features

Bruce Momjian

October, 2014

PostgreSQL is an open-source, full-featured relational database. This presentation gives an overview of the Postgres 9.4 release.

Creative Commons Attribution License *http://momjian.us/presentations*

9.4 Feature Outline

1. Allow materialized views to be refreshed without blocking reads
2. Logical decoding allows database changes to be streamed out in a customizable format
3. Allow background workers to be dynamically registered, started and terminated
4. Add structured (non-text) data type (JSONB) for storing JSON data
5. Add SQL-level ALTER SYSTEM command to edit the postgresql.conf configuration file
6. GIN indexes are 5x smaller and faster for multi-key lookups
7. Improve WAL performance
8. User-configurable delay of streaming replication

Released 2014

1. Allow Non-Blocking Materialized View Refresh

- Allows materialized view refresh without blocking reads
- Enabled with CONCURRENTLY keyword
- Requires a unique index

```
REFRESH MATERIALIZED VIEW CONCURRENTLY matview_test_view;
```

2.Logical Decoding

Logical decoding allows database changes to be optionally streamed in a configurable format. The data is read from the WAL and transformed into the desired target format, which can be easily processed by external tools. In previous releases, only binary changes were recorded in the WAL. This will be used for multi-master replication, zero-downtime major upgrades, write auditing, cache invalidation, and finer-grained replication control.

2.Logical Decoding Example

In postgresql.conf:

```
wal_level = logical
max_replication_slots = 1
```

From SQL:

```
SELECT * FROM pg_create_logical_replication_slot('regression_slot',
                                                 'test_decoding');
    slot_name    | xlog_position
-----------------+---------------
 regression_slot | 0/16E63E0
```

2.Logical Decoding Example

```
CREATE TABLE data(id serial primary key, data text);
BEGIN;
INSERT INTO data(data) VALUES('1');
INSERT INTO data(data) VALUES('2');
COMMIT;
SELECT * FROM pg_logical_slot_get_changes('regression_slot', NULL, NULL);
 location  | xid |                              data
-----------+-----+-----------------------------------------------------
 0/16E6410 | 688 | BEGIN 688
 0/16F7F70 | 688 | COMMIT 688
 0/16F8080 | 689 | BEGIN 689
 0/16F8080 | 689 | table public.data: INSERT: id[integer]:1 data[text]:'1'
 0/16F8148 | 689 | table public.data: INSERT: id[integer]:2 data[text]:'2'
 0/16F8218 | 689 | COMMIT 689
```

3. Dynamic Control of Background Workers

- Registered
- Started
- Terminated
- Also shared memory dynamic control and message queue support

This will be used to support parallel query.

4. Binary JSON Support (JSONB)

- Allows rapid access to JSON keys and values
- Flexible indexing options, including indexing of all keys and values (the default), or all key/value combinations (jsonb_path_ops)

```
CREATE TABLE json_demo(data JSONB);
CREATE INDEX i_json_demo ON json_demo USING gin (data);
```

5. SQL Control of postgresql.conf

This allows postgresql.conf values to be modified via SQL.

```
SHOW work_mem;
 work_mem
----------
 4MB
ALTER SYSTEM SET work_mem = '30MB';
SELECT pg_reload_conf();
-- postgresql.conf settings are pushed to running sessions
SHOW work_mem;
 work_mem
----------
 30MB
```

6. GIN Indexes Are Smaller and Faster

- Typical index shrank from 58MB to 11MB using varbyte encoding of item pointers
- Advantages of bitmap indexes, without the limitations
- When doing AND index matches, rare items are looked up first, reducing lookups of more common values

7. Improve WAL Performance

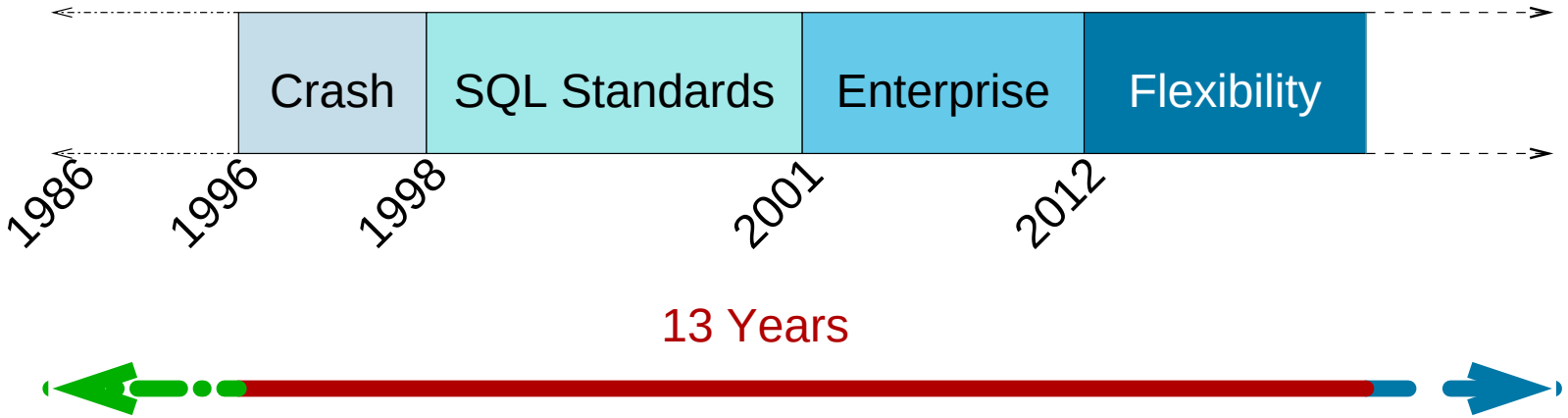
- Allow parallel inserts into WAL
- Only write modified fields into WAL, rather than entire rows

8. User-Configurable Delay of Streaming Replication

- Controlled by `recovery_min_apply_delay`
- Useful for recovering from mistakes

PostgreSQL Future

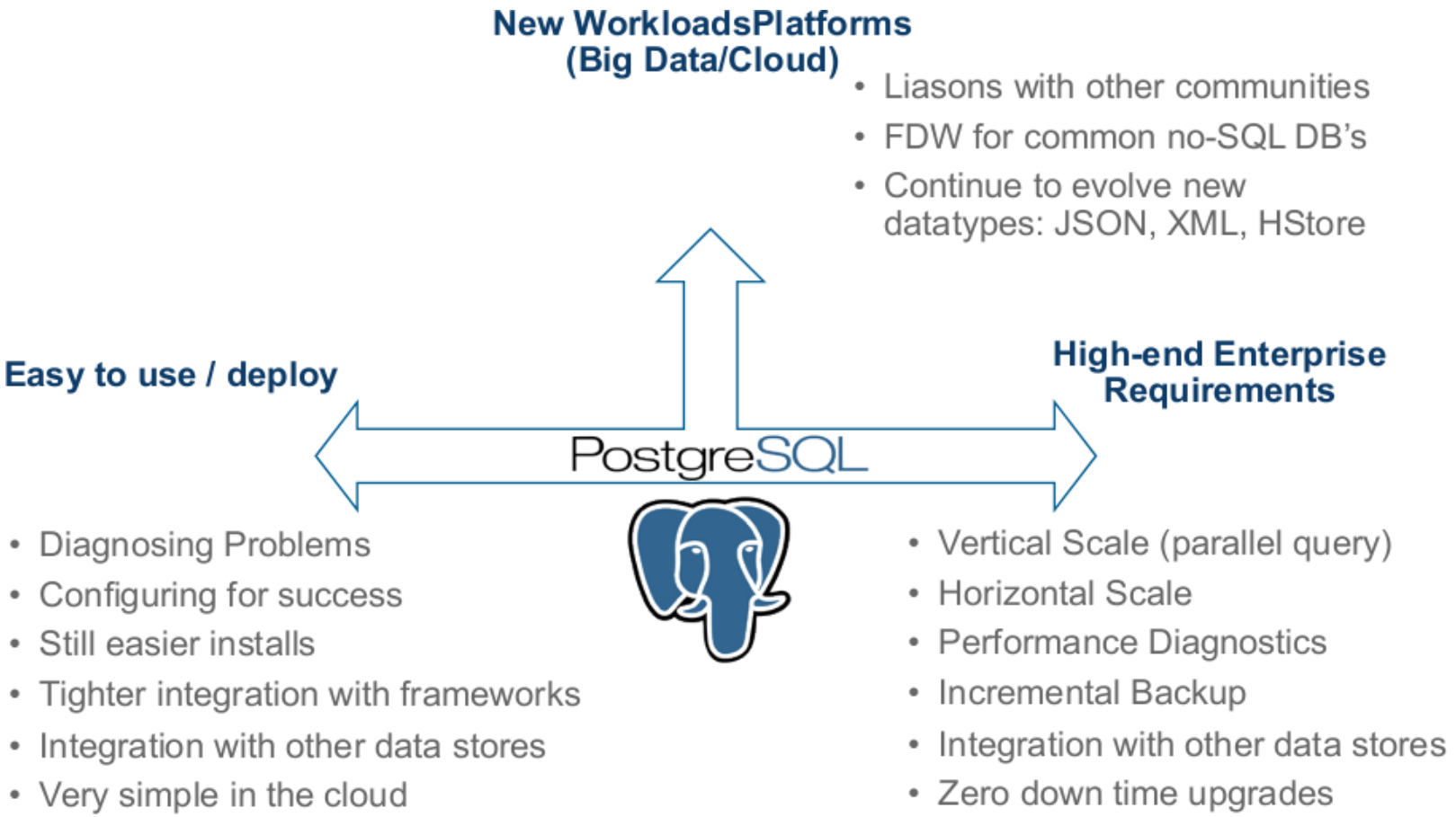
PostgreSQL Evolution



Flexibility includes:

- Application-specific data types, e.g. JSON, PostGIS, range types
- Advanced index types, e.g. GIN, SP-GiST
- Single and multi-node scalability

Three Focuses



Keith Alsheimer, EnterpriseDB

Conclusion

http://momjian.us/presentations

http://www.flickr.com/photos/timlawrenz/