shaping tomorrow with you

PostgreSQL Development of shared disk scale-out

November 15, 2019
Fujitsu Limited
Data Management Division
Takayuki Tsunakawa

Self-introduction

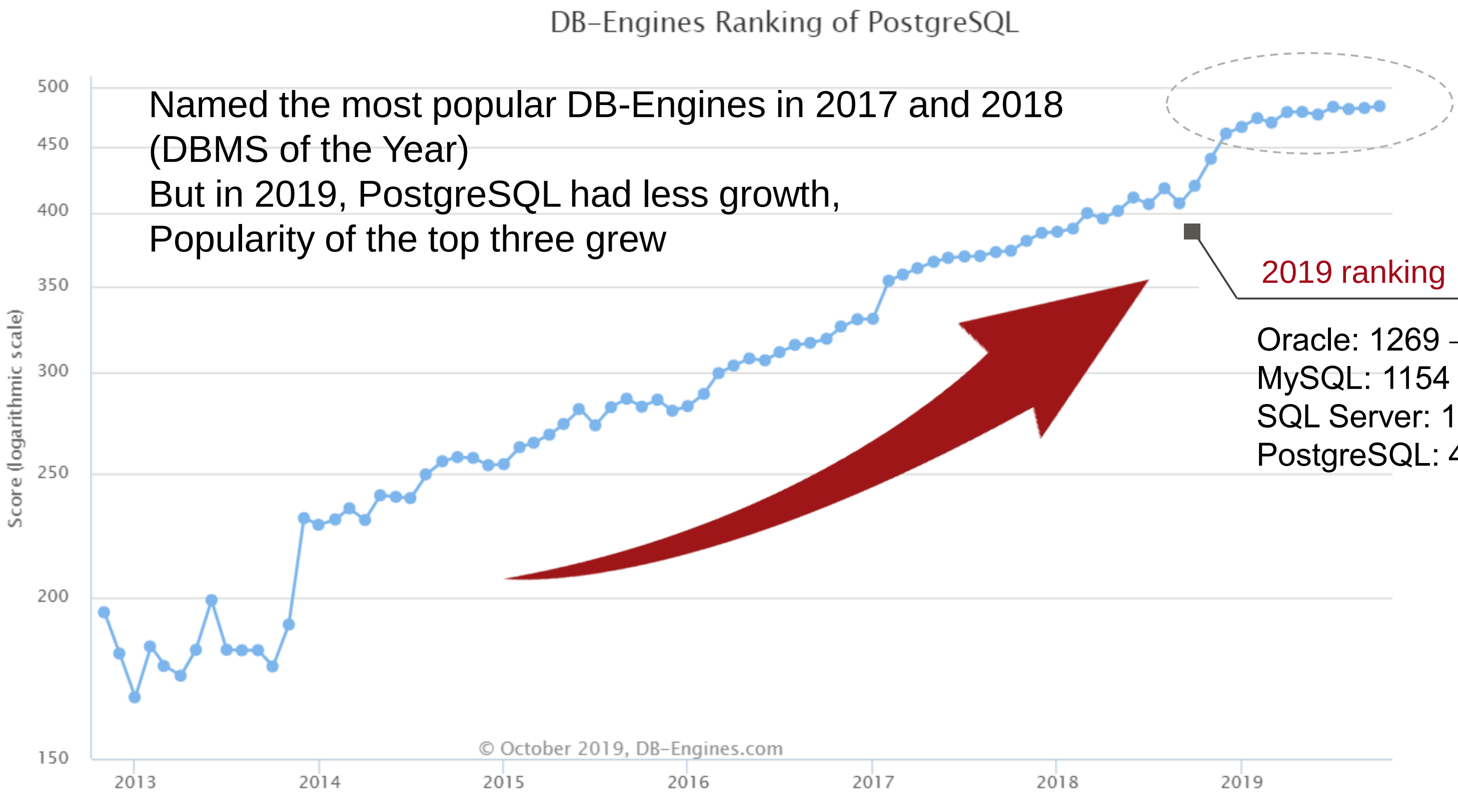
- Community citizen
  - PostgreSQL contributor
  - PostgreSQL Ecosystem Wiki creation & maintainer
  - Member of the CR division of the PostgreSQL Enterprise Consortium

- Corporate citizen
  - Community Activities Team Leader
  - FUJITSU Software Enterprise Postgres (FEP) Developer

Contents of this deck

- Triggers for the development of shared disk scale-out
- Compare scalability choices
- Shared disk scale-out architecture and design
- Is shared disk scale-out needed?

PostgreSQL's popularity trends



Named the most popular DB-Engines in 2017 and 2018 (DBMS of the Year)
But in 2019, PostgreSQL had less growth,
Popularity of the top three grew

2019 ranking

Oracle: 1269 → 1355, + 86
MySQL: 1154 → 1283, + 129
SQL Server: 1040 → 1095, + 55
PostgreSQL: 466 → 484, + 18

https://db-engines.com/en/ranking_trend/system/PostgreSQL

The Popularity of Linux

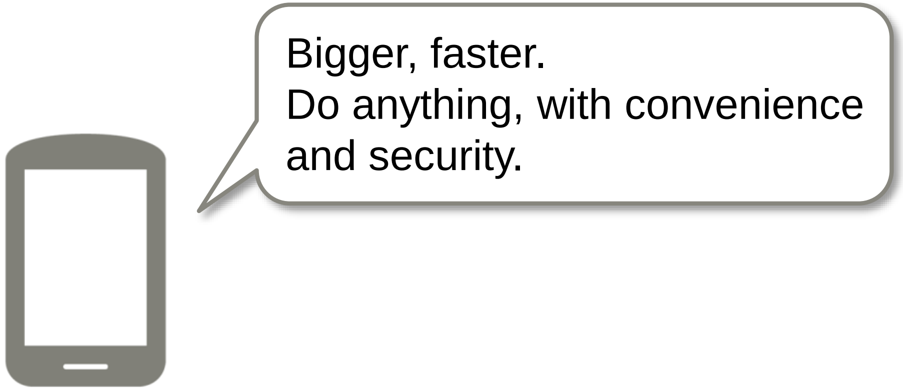
- About half of Azure's VMs are running Linux
- 40% of top 1 million sites run Linux, 33% Windows (W3Techs.com, Oct 2019)
- # 1 Growth in Server OS Market, Second in market sales share

What can we do to make it more popular?

- Scale-out: Handle more data and requests
- Hardware-accelerated: DRAM, non-volatile memory, GPU, FPGA
- Multiple models: working with data in different formats

 (Key Value, Document, Graph, Time Series...)

- Higher security: encryption, privilegs, SQL firewall
- Improved migration from other DBMSs



Customer voice for shared disk scale-out

- "We want Oracle RAC capabilities"

1. Want to exceed the processing power of a single server
   - Satisfied with Exadata performance and scalability
2. Need cost-effective high availability
   - Want to read and write data utilising standby server
3. Don't want to change the application
   - It is difficult to partition and position data

Long history of shared disk scale-out

FUJITSU

- More than 25 years since mainframe and minicomputer era
  - Oracle Parallel Server (OPS) for VAX/VMS — early 1990s
    ->Oracle Real Application Clusters (RAC) — 2001
  - IBM DB2 for z/OS - early 1990s
    ->IBM DB2 pureScale for AIX, Linux - 2009
  - Sybase Adaptive Server Enterprise Cluster Edition: 2010

- Is it so easy to incorporate this advanced technology into PostgreSQL?
- Is it useful?　Will it lead to PostgreSQL taking a leap?

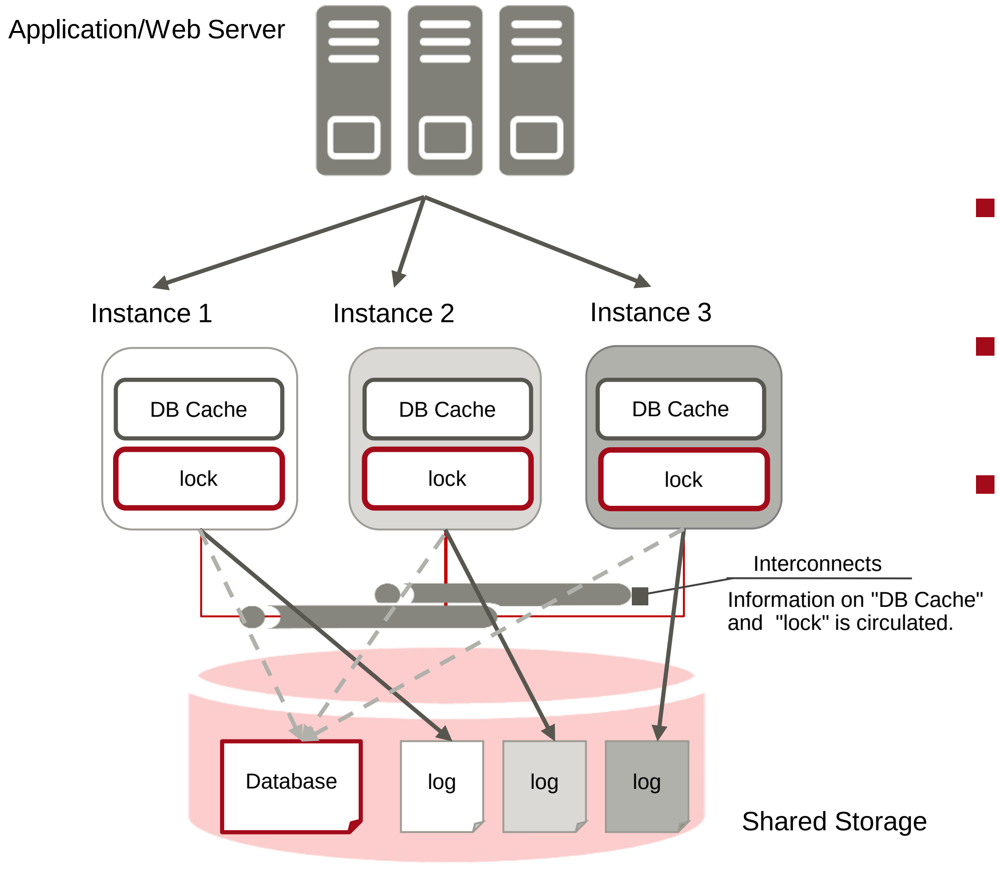
Compare scalability choices

1. shared disk scale-out (hereinafter SD)
   - Oracle RAC, IBM Db2 pureScale

2. shared nothing scale-out (hereinafter SN)
   - Oracle Sharding, IBM Db2
   - Google Cloud Spanner, CockroachDB, MySQL Cluster
   - Azure Database for PostgreSQL - Hyperscale (Citus)
   - Greenplum, Postgres-XL

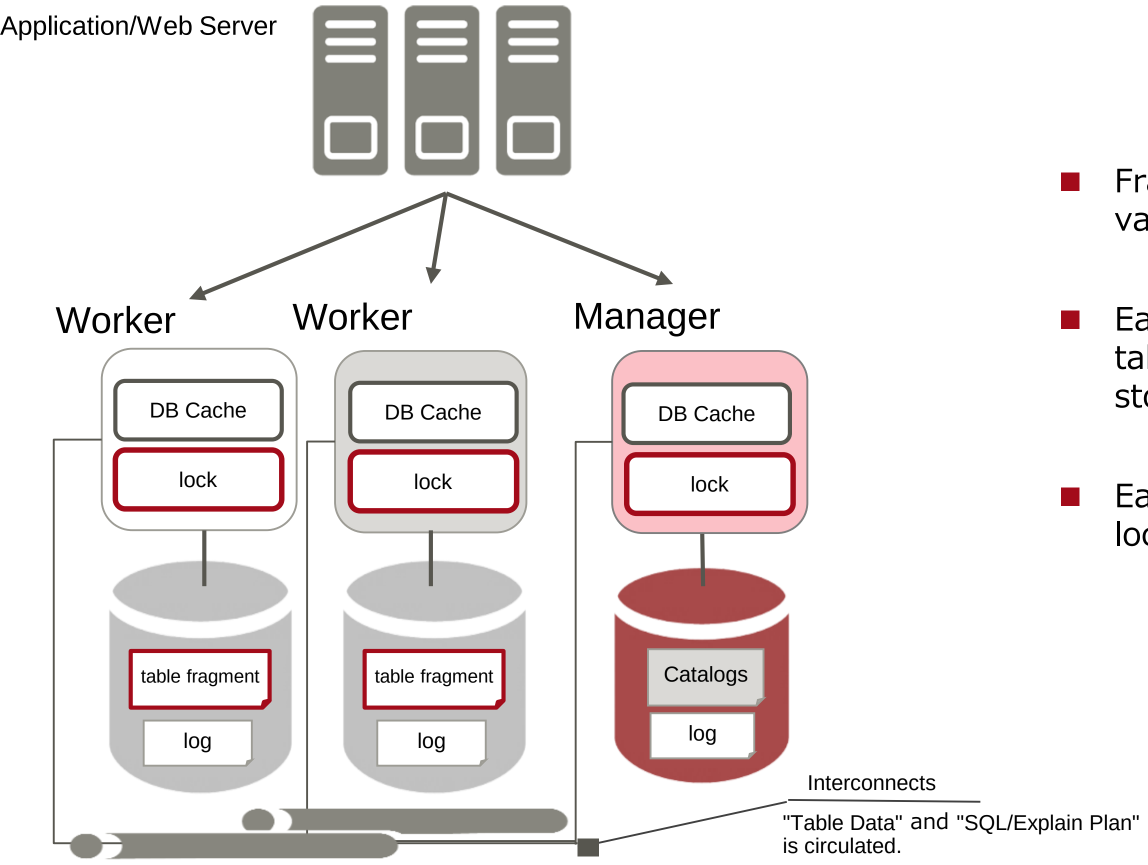
3. Scale up

Shared disk method overview



- Multiple DB instances accessing database on shared storage
- DB cache and locks are managed in a distributed way across coordinated DB instances
- Each DB instance writes to a separate transaction log

Overview of the shared nothing method



- Fragments a table into row groups by column values and stores the fragments in DB instances
- Each DB instance reads and writes only its own table fragments and transaction logs on local storage
- Each DB instance manages the DB cache and locks for its own table fragments

Method comparison - Scalability and performance (1/7)

- Scalability of processing capability: SN > SD > Scaleup
  - Is scale up enough for OLTP for users in a single organization?
    - x 86 commodity servers: 2 sockets/56 cores, hundreds of GB RAM
    - X 86 high-end servers (HPE Superdome): 32 sockets/896 cores, 48 TB RAM
    - Cloud VM (Azure M 208 ms v2): 208 vCPU, 5700 GB RAM
  - SD method scales RAC to 100 nodes and pureScale to 128 nodes
    - pureScale is used for mostly read Web commerce workloads
    - High scalability of 95% on 64 nodes and 84% on 128 nodes (Source: IBM)
  - The SN method has the highest scalability because it has few competing shared resources.

Method Comparison - Scalability and performance (2/7)

- Performance comparison
  - TPC-C delivers high SN and SD performance
    - 1 = SN 60.88 million tpmC (Alibaba OceanBase, 2019)
    - 2 = SD: 30.24 million tpmC (Oracle RAC, 2010)
    - 3 = SN: 10.36 million tpmC (IBM DB2, 2010)

Method Comparison - Scalability and performance (3/7)

- Easiness of scaling: Scaleup > SD > SN
  - Scaleup: Add CPU and memory to server, change VM instance
  - SD: Add Server. Performance may require changes to the physical data structure
  - SN: Add servers and redistribute data or change where apps connect

- Distance between DB servers: SN > SD
  - SD – all DB servers are close together to share the entire dataset
  - SN — A globally distributed database divides data sets by country or region.
    Data owned by DB servers that are near local users of each location

- Load balancing: SD > SN
  - SN: To balance the load on the DB server, distribute data so that access frequency is equal

Method Comparison - Scalability and performance (4/7)

- Cache validity: SN > SD > Scaleup
  - SD: Each DB server must cache the entire dataset
  - SN: Each DB server only needs to cache its own data.

- Hotspot tolerance: SN > Scaleup > SD

How can I reduce the concentration of reads and writes to particular data and transaction logs?

  - SN: Divide data across DB servers and distributes access
  - Scaleup: More CPU, more storage, less waiting at hotspots
  - SD: Read hotspots are mitigated by adding more DB servers and storage

    Writes cause increased latency in competition on the same block due to cache integrity

Method Comparison - Scalability and performance (5/7)

- Data injection and analysis, batch processing: SN > SD > Scaleup
  - SN is ideal for handling large amounts of data
  - To reduce read/write response time, divide data and processing to run in parallel
  - Reduce bottlenecked shared resources to increase throughput
  - SN occupies the top rank of TPC-H.

Method Comparison - Scalability and performance (6/7)

- Multi-tenant OLTP: SN > SD > Scaleup

  Apps that allow users, devices, organizations and stores to divide data and processing

  - SN: Distribute data on DB server with key containing tenant ID.

    The app uses the tenant ID to select a connection from the connection pool.

    If it cannot be partitioned, such as read-only data, it must be on the management server.

    Otherwise, replicate on all DB servers.
  - SD: Like SN, divide data and send a tenant's request to a specific DB server

    Shared storage is the bottleneck. Advantages of not having to duplicate read-only data
  - Sequence is the bottleneck for both SN and SD
  - Access using a secondary index in SN is likely to be slow across multiple DB servers

Method Comparison - Scalability and performance (7/7)

- Single-tenant OLTP: SD > Scaleup > SN

  Difficult to divide data for use by users in one organization

  - Scaleup: The simplest and most cost-effective way if handled on a commodity server
  - SD — Keep data as is, add servers (CPU and Memory) to increase throughput

    Response times are higher than single servers due to distributed buffer and lock management

    When reads and writes are concentrated on the same block, data transfer increase and performance degradate due to cache integrity,

    Example:Row insertion in ascending order of key to index, numbering from sequence
  - SN: Increased response time due to data transfer between DB servers and distributed 2PC

Method Comparison - Availability

- Resilience to server failures: SD > SN = Scaleup
  - SN – only failed server data is inaccessible

    Increasing the number of servers increases the probability that one of them will fail, reducing the overall availability of the cluster

    Single point of failure is admin node
  - SD: Regardless of the failed DB server, surviving server can access all data

- Resilience to storage failures: SN = SD = Scaleup

  All methods require hardware or software redundancy

- Failover impact: SD > SN = Scaleup
  - Scaleup, SN – During DB server recovery, only the data it holds is inaccessible
  - SD – Other DB servers can access data other than what was updated by the failed DB server

    RAC freezes the activity of the entire cluster during remaster and identification of recovery set

Method Comparison - Application transparency (1/3)

- Data placement: SD = Scaleup > SN
  - SN: Data segmentation and placement designed for inter-node transfer and 2PC avoidance and load balancing

    Partition tables by user or store IDs and distribute across DB servers

    Place copies of read-only data on all DB servers

- Workload management: Scaleup > SD > SN
  - SN: The app issues a transaction request to the DB server with the required data.
  - SD: All data is shared so you don't have to worry about where you put it

    However, to reduce read/write competition for the same block,

    it is better to split workloads across DB servers

Method Comparison - Application transparency (2/3)

- Application changes: Scaleup > SD > SN
  - DML statements do not need to be modified on any method
  - SN: Change DDL statements based on data placement design

    Partition tables by hash or range.

    Distribute to different DB servers

    Do not increase the sequence cache or enforce ordering
  - SD: Change DDL statements to reduce read/write competition for the same block between DB servers

Method Comparison - Application transparency (3/3)

[Recommendations for RAC]

- Reduce the number of indexes
- Do not increase the sequence cache or enforce ordering
- Generate node-specific sequence range values (scalable sequence)
- Choose a smaller block size or set free space for blocks
- Partition the primary key index
- Hash partitioning tables to create local indexes
- Use a reverse key index with key bits inverted

Method Comparison - Cost (1/2)

- Servers: Scaleup = SD > SN
  - Scaleup, SD: need only DB server to access data
  - SN – Admin server and its standby required

    Manages cluster membership, DB catalogs, sequences, and transactions

- Standby server capacity: SD > SN = Scaleup
  - Scaleup, SN: Standby servers can perform read queries and backups. Cannot write.
  - SD – No standby-only server, all servers can read and write

Method Comparison - Cost (2/2)

- Storage: Scaleup > SN > SD
  - Scaleup provides direct-attached storage (DAS).

    Get the most out of your storage capacity and performance
  - SN: Same as scaleup, but if data and processing cannot be distributed well, capacity and performance will be insufficient
  - SD: No DAS available, lower price/performance ratio

    Access storage over network (FC, NVMet-oF, iSCSI)

    If using a distributed file system, then also through the storage server (NFS, Ceph)

    Persistent memory in server memory slots is not used for its true-value

Method Comparison Summary (1/2)

FUJITSU

- SN has higher scalability and price/performance ratio than SD

Scalability and performance

Item	SN	SD	Scaleup
Scalability of processing capability	3	2	1
Easiness of scaling	1	2	3
Distance between DB servers	3	1	1
Load balancing	1	3	1
Cache validity	3	2	1
Hotspot tolerance	3	1	2
Data injection and analysis, batch processing	3	2	1
Multi-tenant OLTP	3	2	1
Single-tenant OLTP	1	3	2
Subtotal	21	18	13

Cost

Item	SN	SD	Scaleup
Number of servers	2	3	3
Storage	2	1	3
Standby server capacity	2	3	2
Subtotal	6	7	8

Method Comparison Summary (2/2)

- SD outperforms SN for availability and application transparency

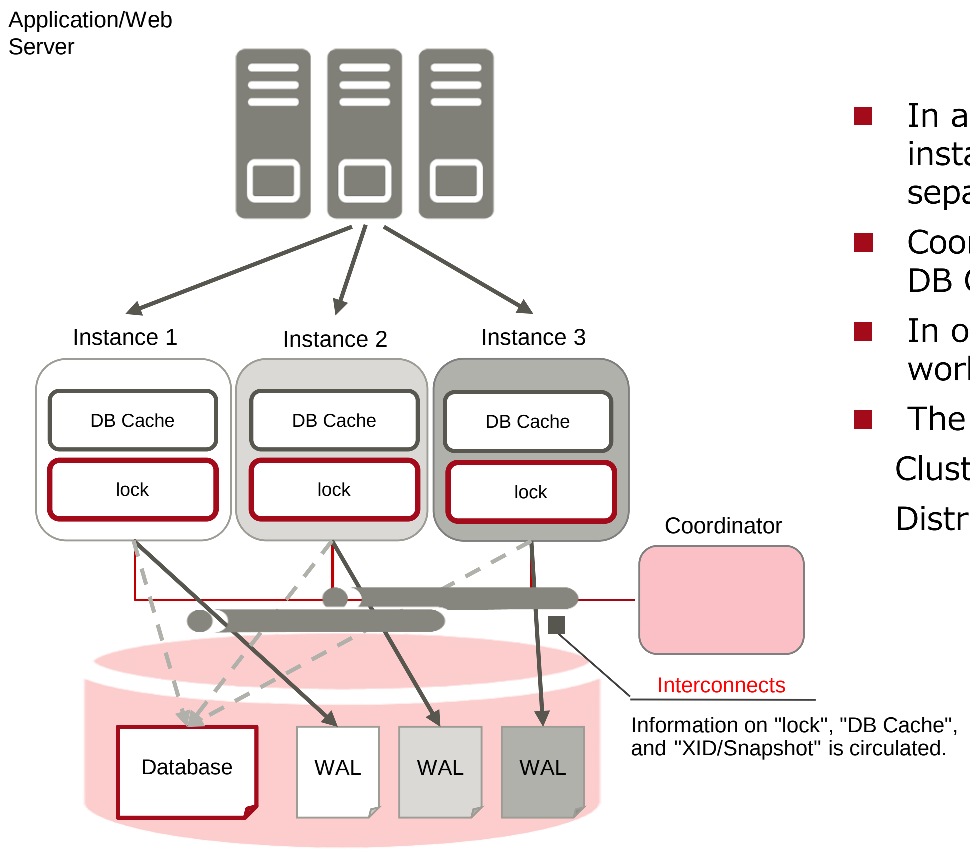
Availability

Item	SN	SD	Scaleup
Resilience to server failures	1	3	1
Resilience to storage failures	1	1	1
Failover impact	1	2	1
Subtotal	3	6	3

Application transparency

Item	SN	SD	Scaleup
Data placement	1	3	3
Workload management	1	2	3
Application changes	1	2	3
Subtotal	3	7	9

Overall configuration of the shared disk method for PostgreSQL

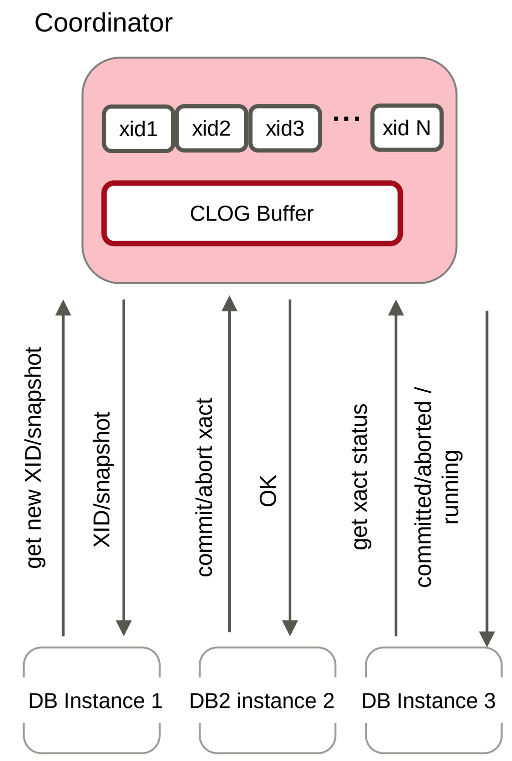


- In addition to a maximum of 128 DB instances, Coordinator is running on a separate server
- Coordinator centrally manages transactions, DB Cache, and locks
- In order to eliminate SPoF, Coordinator works in a master-standby configuration
- The file system is either

  Cluster FS (Red Hat GFS2, GPFS) or

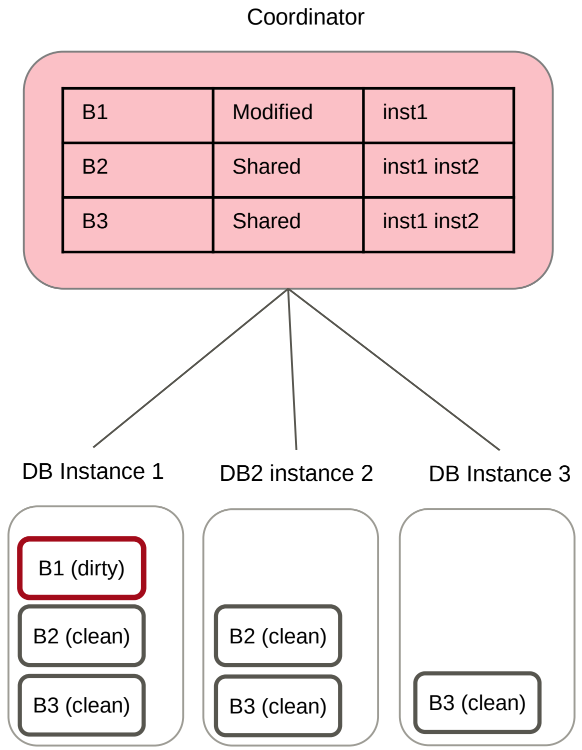
  Distributed FS (Ceph, NFS)

Transaction management



- Coordinator manages:
  - List of active transaction IDs (XID)
  - Commit log showing transaction status (CLOG)
- Upon request from the backend, the Coordinator
  - Assign XIDs, take snapshots and return to requestor
  - Log transaction completion in XID list and CLOG
  - Notify requester of transaction status by CLOG

DB cache management



B1	Modified	inst1
B2	Shared	inst1 inst2
B3	Shared	inst1 inst2

- DB instances with separate caches read and write the same block ->Cache integrity required
- Cache consistency ≈ Prevent old data from being read
- CPU cache consistency protocol MESI
  - Multiple DB instances can have clean (Shared) copies of the same block
  - When a DB instance writes a block and makes it dirty (Modified), the copies that are held by other DB instances are discarded (Invalidate)
  - Coordinator manages the status of each block with its cached instance. Mediates exchanges.

Lock management

- Coordinator and DB instance share management
  - Coordinator centrally manages lock tables for deadlock detection
  - However, if it manages all locks at all times, Coordinator becomes the bottleneck
  - Fast Path Locking (FPL) since PG 9.2 means DML does not interact with Coordinator
  - Weak locking: SELECT, INSERT, DELETE, UPDATE
  - Strong locking: ALTER/DROP, CLUSTER, REINDEX, etc.

- DB instance
  - Weak locks continue to be fast with FPL
  - Strong locks are requested to and processed by Coordinator

- Coordinator
  - When a strong lock request is received, it instructs the DB instance with the conflicting weak lock to send the lock information.
  - Detects deadlock and instructs DB instance to abort transaction

Other design elements

- Network
  - Use UDP for much of the DB instance-to-instance communication to reduce overhead
  - Even faster with InfiniBand and RoCE (RDMA over Converged Ethernet)

    Mellanox 100Gb Ethernet/IB NICs for $795, 50 GbE NICs for $ 475
    InifiBand and RDMA available for Azure H-Series VMs
  - Do not use multicast

    Oracle RAC uses multicast, so it cannot be used on clouds like AWS or Azure
- Cluster
  - The HA framework such as monitoring, failover, and fencing is left to the clustering software.
- Recovery
  - Coordinator detects DB instance failure and directs recovery to one of the DB instances
  - Apply WAL to keep multiple DB instances' updates to the same page

    To do this, add a page generation number to the page header and to the WAL record
  - PITR merges all DB instance WALs and applies them in their original update order

Performance (Reference)

PG-CALS performance (Source: Yotaro NAKAYAMA, 2007)

UNIADEX (stock), in partnership with Unisys, announces preliminary findings

2 DB servers, pgbench, 100 concurrent connections

How many times (X) is tps per 1 existing PostgreSQL?

Read-only:update ratio 7:3 is 1.4X, 0:10 is 1.8X

Our implementation wants to increase this X factor by the following differences

- Updating buffer does not send data to Coordinator
- Based on FPL, DML statements do not interact with Coordinator
- Low latency network and RDMA

Is shared disk scale-out needed?

Assumptions: SN is required for analysis and Web-scale OLTP

Question: Does OLTP require SD?

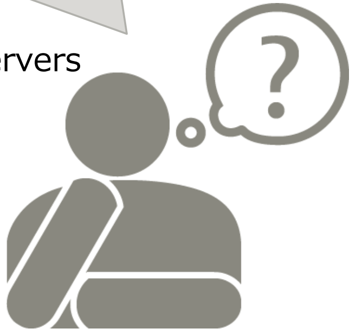
1. Can you handle it with scale up?
   - On commodity servers, up to 56 cores and up to 1.5 TB of DRAM is possible
   - Storage accelerated with NVMe SSD and persistent memory
   - Amazon wipeout nearly 7,500 Oracle DBs (2019/10/15)
   - Migrated to RDS, Aurora, Redshift, DynamoDB, ElastiCache

2. Is SD a cost-effective HA?
   - Reduce utilization of servers in normal operation to take over the processing of failed servers

     ->What's the difference between having a standby server underutilized?

3. Do users want it even with changes to their applications?
   Is it possible to implement those changes?

Seeking for opinion



FUJITSU

shaping tomorrow with you