STORMATICS

Clustering in PostgreSQL

Because one database server is never enough (and neither is two)

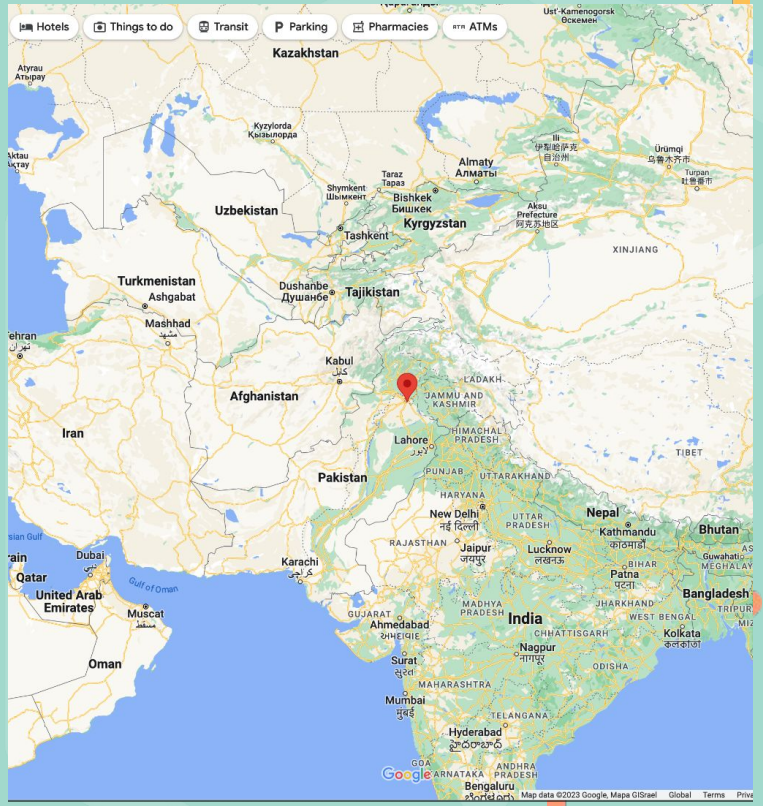
FOSDEM 2024

Brussels / 3 & 4 February 2024

STORMATICS

```
postgres=# select * from umair;

-[ RECORD 1 ]-------------------------------
name        | Umair Shahid
description | 20+ year PostgreSQL veteran
company     | Stormatics
designation | Founder
location    | Islamabad, Pakistan
family      | Mom, Wife & 2 kids
kid1        | Son, 17 year old
kid2        | Daughter, 14 year old
```



EDB POWER TO POSTGRES™ | OpenSCG PostgreSQL, Java & Linux Experts™ | 2ndQuadrant PostgreSQL® | PERCONA

Professional Services for PostgreSQL

Our mission is to help businesses scale PostgreSQL reliably for their mission-critical data

STORMATICS

On to the topic now!

STORMATICS

What is High Availability?

- Remain operational even in the face of hardware or software failure
- Minimize downtime
- Essential for mission-critical applications that require 24/7 availability
- Measured in ‘Nines of Availability’

STORMATICS

Nines of Availability

Availability	Downtime per year
90% (one nine)	36.53 days
99% (two nines)	3.65 days
99.9% (three nines)	8.77 hours
99.99% (four nines)	52.60 minutes
99.999% (five nines)	5.26 minutes

STORMATICS

But my database resides in the cloud, and the cloud is always available

Right?

STORMATICS

Wrong!

STORMATICS

Amazon RDS Service Level Agreement

Multi-AZ configurations for MySQL, MariaDB, Oracle, and PostgreSQL are covered by the Amazon RDS Service Level Agreement ("SLA"). The RDS SLA affirms that AWS will use commercially reasonable efforts to make Multi-AZ instances of Amazon RDS available with a Monthly Uptime Percentage of at least 99.95% during any monthly billing cycle. In the event Amazon RDS does not meet the Monthly Uptime Percentage commitment, affected customers will be eligible to receive a service credit.*

99.95% = 4.38 hours of downtime per year!

22 minutes of downtime per month!

* https://aws.amazon.com/rds/ha/

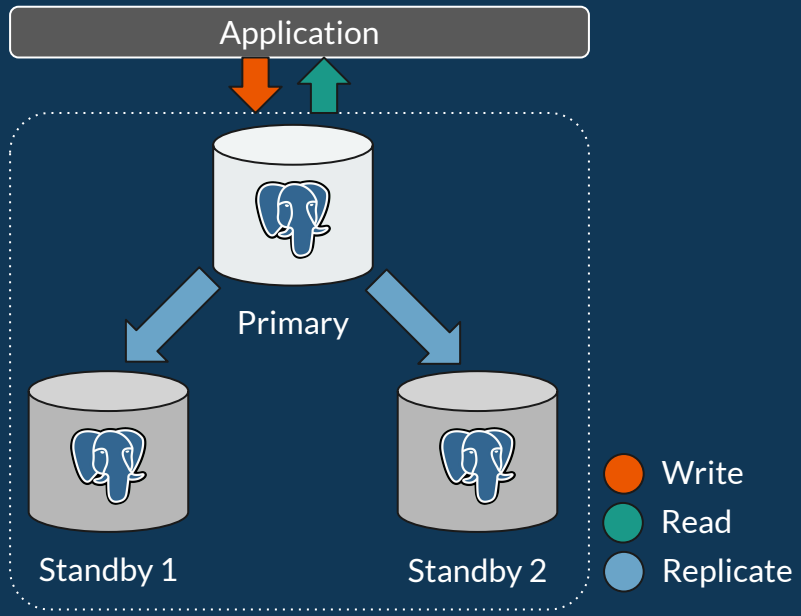
STORMATICS

So - what do I do if I want better reliability for my mission-critical data?

Clustering!

STORMATICS

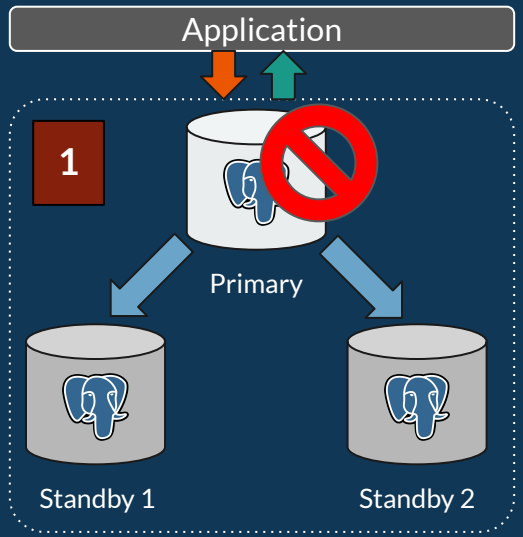
What is clustering?



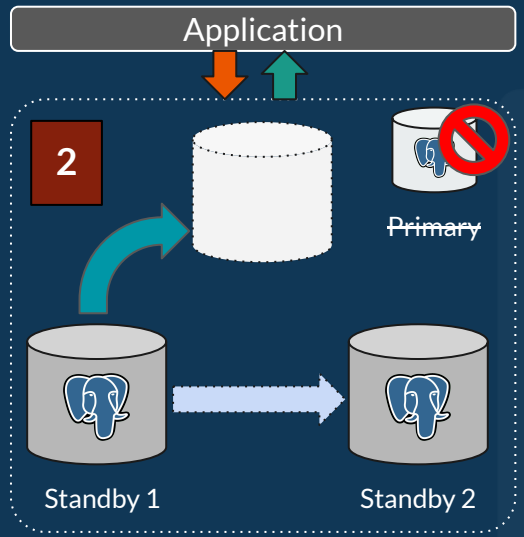
- Multiple database servers work together to provide redundancy
- Gives the appearance of a single database server
- Application communicates with the primary PostgreSQL instance
- Data is replicated to standby instances
- Auto failover in case the primary node goes down

STORMATICS

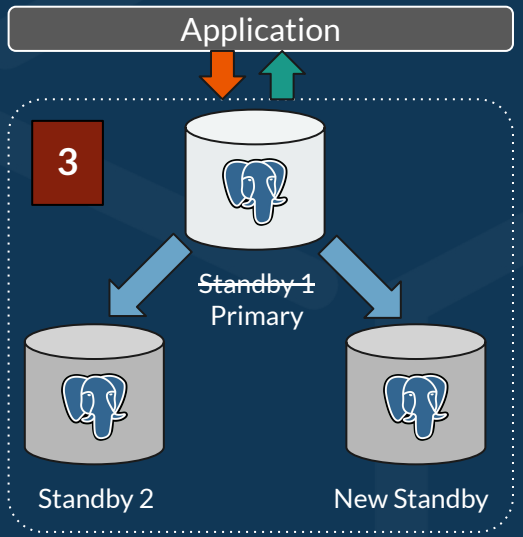
What is auto failover?



* Primary node goes down



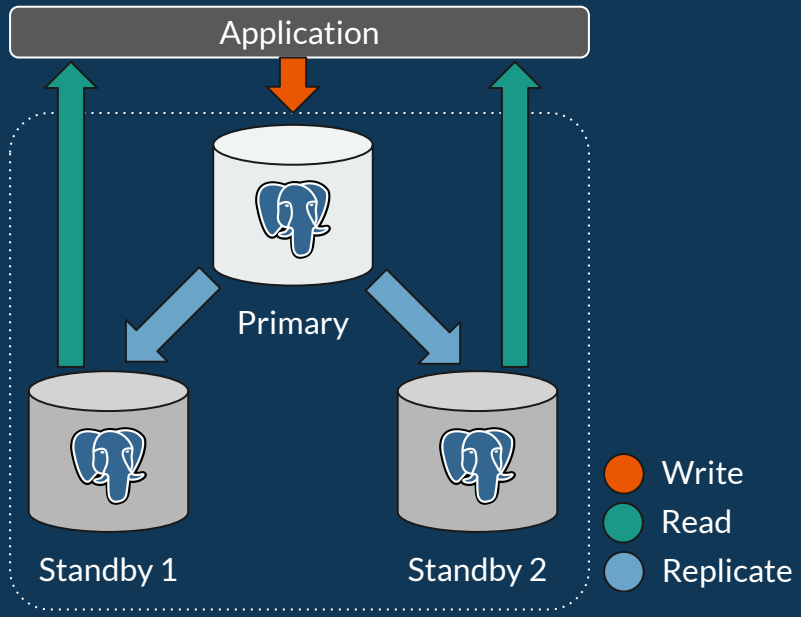
* Standby 1 gets promoted to Primary
* Standby 2 becomes subscriber to Standby 1



* New Standby is added to the cluster
* Application talks to the new Primary

STORMATICS

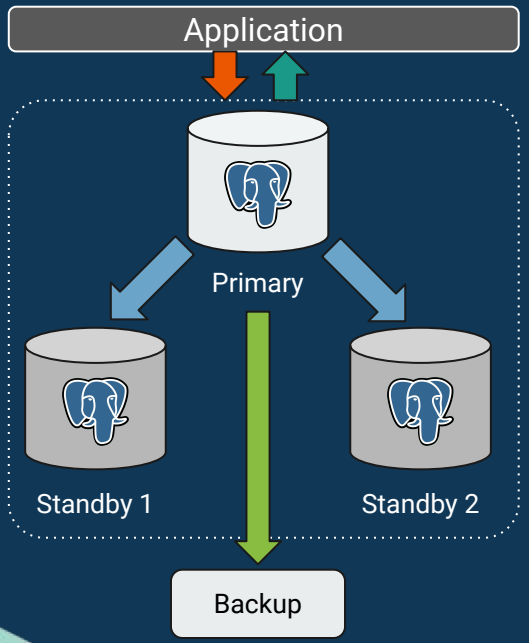
Clusters with load balancing



- Write to the primary PostgreSQL instance and read from standbys
- Data redundancy through replication to two standbys
- Auto failover in case the primary node goes down

STORMATICS

Clusters with backups and disaster recovery



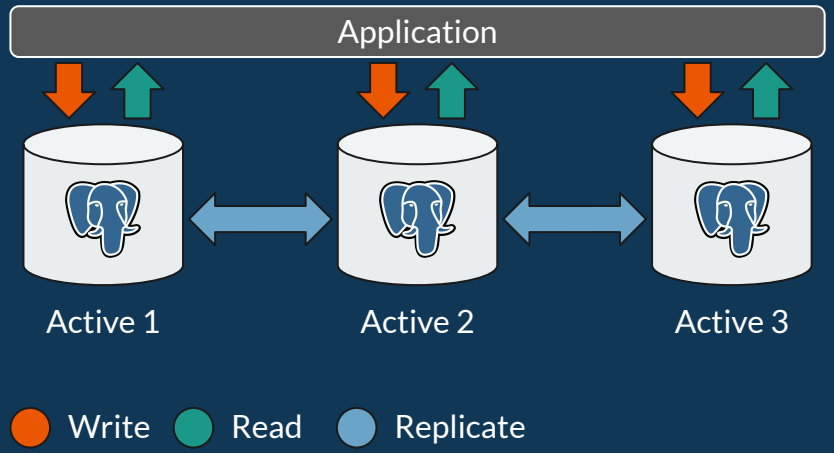
Write
Read
Replicate
Backup

- Off-site backups
- RTO and RPO requirements dictate configuration
- Point-in-time recovery

It is extremely important to periodically test your backups

STORMATICS

Multi-node clusters with Active-Active configuration*

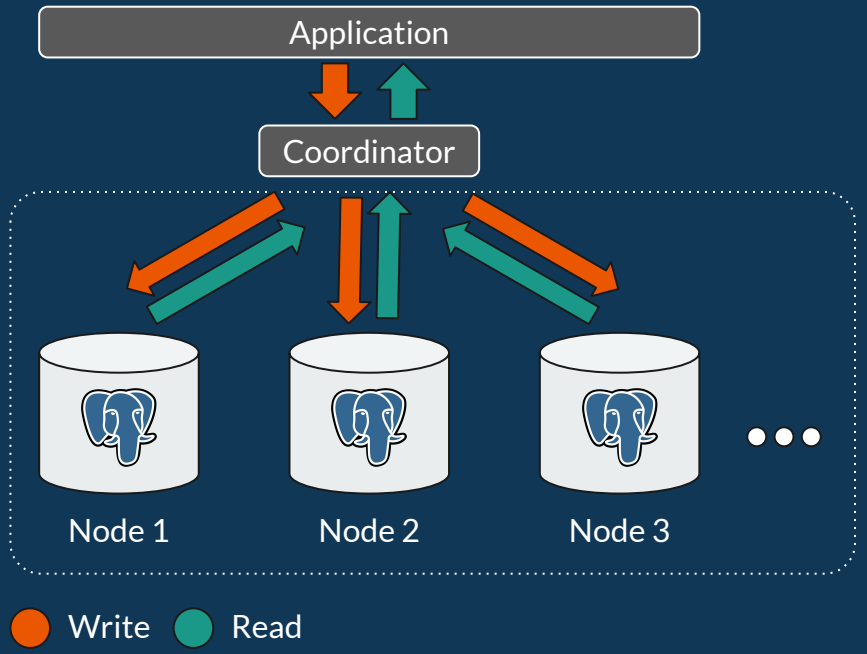


- Shared-Everything architecture
- Load balancing for read as well as write operations
- Database redundancy to achieve high availability
- Asynchronous replication between nodes for better efficiency

* *with conflict resolution at the application layer*

STORMATICS

Multi-node clusters with data sharding and horizontal scaling



- Shared-Nothing architecture
- Automatic data sharding based on defined criteria
- Read and write operations are auto directed to the relevant node
- Each node can have its own standbys for high availability

STORMATICS

Globally distributed clusters

- Spin up clusters on the cloud, on-prem, bare metal, VMs, or a hybrid of the above
- Geo fencing for regulatory compliance and better local performance
- High availability across data centers and geographies

STORMATICS

Replication - Synchronous vs Asynchronous

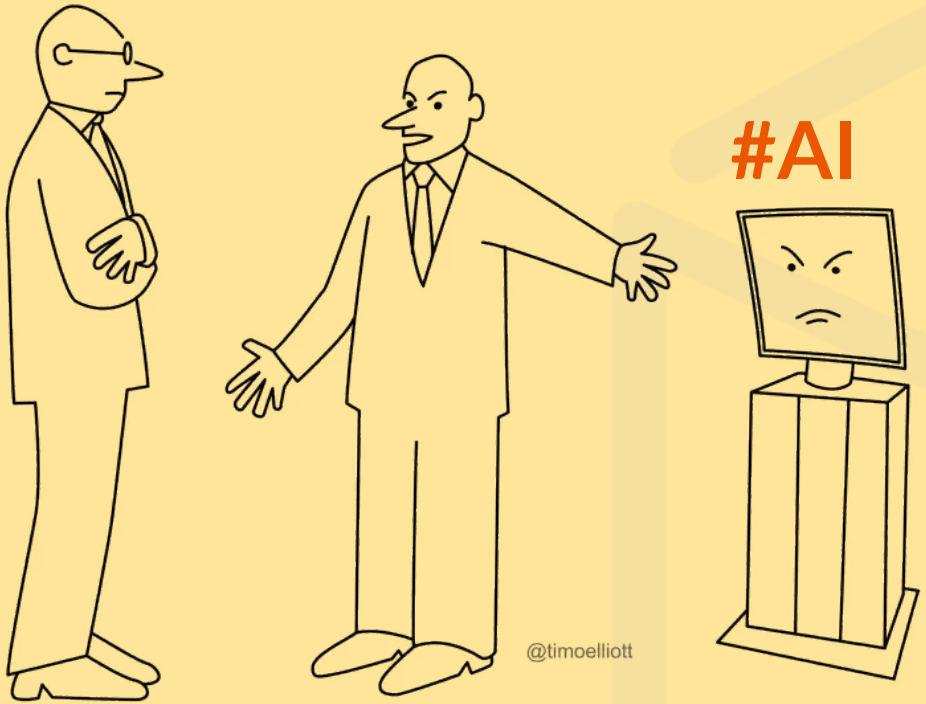
Synchronous

- Data is transferred immediately
- Transaction waits for confirmation from replica before it commits
- Ensures data consistency across all nodes
- Performance overhead caused by latency
- Used where data accuracy is critical, even at the expense of performance

Asynchronous

- Data may not be transferred immediately
- Transaction commits without waiting for confirmation from replica
- Data may be inconsistent across nodes
- Faster and more scalable
- Used where performance matters more than data accuracy

STORMATICS



His decisions aren't any better than yours
— but they're WAY faster...

STORMATICS

Challenges in Clustering

- Split brain
- Network latency
- False alarms
- Data inconsistency

STORMATICS

Challenges in Clustering

- **Split brain**
- Network latency
- False alarms
- Data inconsistency

STORMATICS

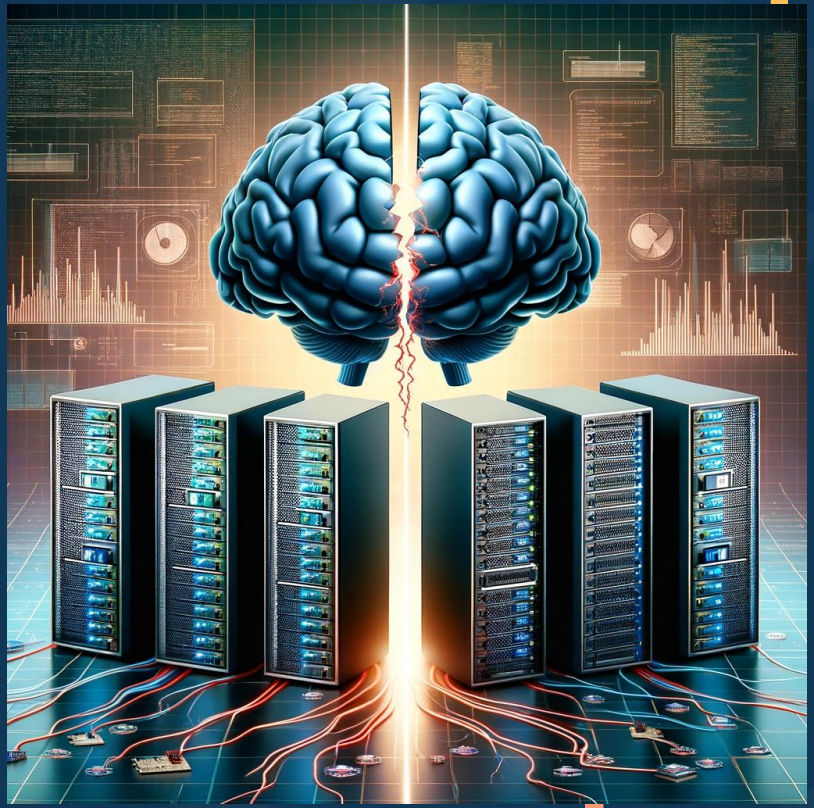
Split Brain

Defined

Node in a highly available cluster lose connectivity with each other but continue to function independently

Challenge

More than one node believes that it is the primary leading to inconsistencies and possible data loss



STORMATICS

Split Brain - Prevention

- Use a reliable cluster manager
  - Algos and heartbeat mechanisms to monitor node availability
  - Make decisions about failovers and promotions
- Quorum-based decision making
  - Majority of nodes must agree on primary node's status
  - Requires odd number of nodes
- Witness server
  - Used to achieve a majority in an even-node cluster
  - Does not store data
- Network reliability and redundancy
  - Minimize the risk of partitions due to connectivity issues
  - Redundant network hardware and paths between nodes
  - Reliable cross datacenter connectivity
- Miscellaneous
  - Monitoring and alerts
  - Regular testing
  - Clear and precise documentation
  - Training

STORMATICS

Split Brain - Resolution

1. Identify the situation
   - Monitoring and alerting is crucial
2. Stop traffic
   - Application will need to pause
3. Determine the most up to date node
   - Compare transaction logs, timestamps, transaction IDs, etc …
4. Isolate the nodes from each other
   - Prevent further replication so outdated data does not overwrite latest one
5. Restore data consistency
   - Apply missed transactions
   - Resolve data conflicts
6. Reconfigure replication
   - Make the most update to date node the primary
   - Reinstate the remaining nodes as replicas
7. Confirm integrity of the cluster
   - Monitor and double-check replication
8. Re-enable traffic
   - Allow read-only traffic, confirm reliability, then allow write operations
9. Run a retrospective
   - Thorough analysis of the incident to prevent future occurrences
   - Update docs and training to capture the cause of split brain

STORMATICS

Challenges in Clustering

- Split brain
- **Network latency**
- False alarms
- Data inconsistency

STORMATICS

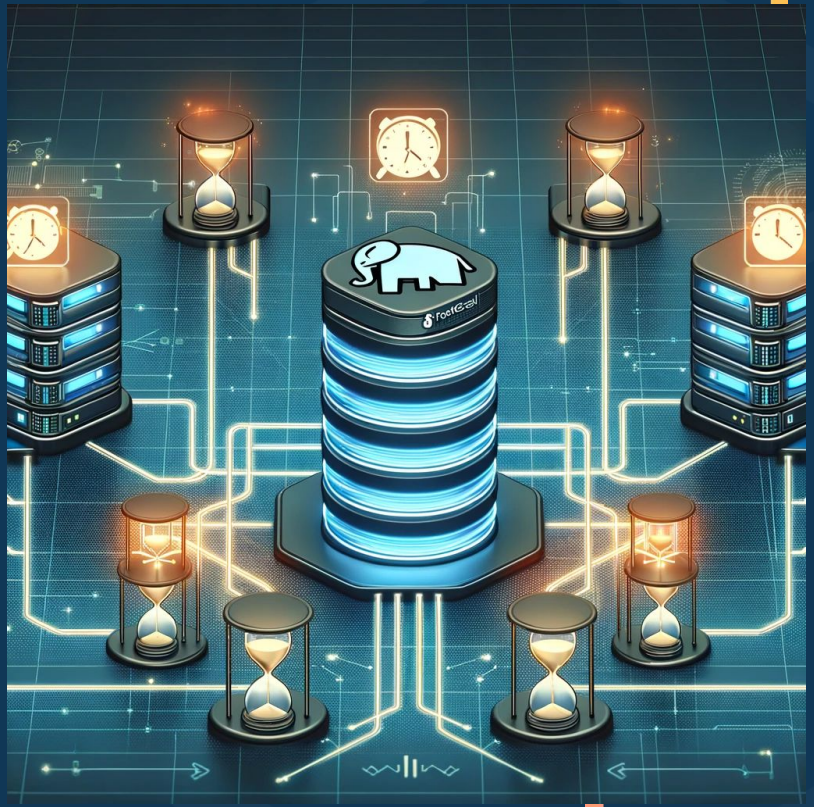
Network Latency

Defined

Time delay when data is transmitted from one point to another

Challenge

Delayed replication can result in data loss. Delayed signals can trigger a false positive for failover.



STORMATICS

Network Latency - Causes

- Network congestion
- Low quality network hardware
- Distance between nodes
- Virtualization overheads
- Bandwidth limitations
- Security devices and policies
- Transmission medium

STORMATICS

Network Latency - Prevention of False Positive

- Adjust heartbeat & timeout settings
  - Fine tune frequency of heartbeat and timeout to match typical network behavior
- High speed & low latency network
  - Investing in high quality networking pays dividends
- Quorum-based decision making
  - Majority of nodes must agree on primary node's status
  - Requires odd number of nodes or a witness node for tie-breaker
- Employ redundancy
  - Network paths as well as health checks
- Best practices
  - Test and simulate various network conditions
  - Monitoring and alerting for early detection of problems
  - Documentation of rationale behind values chosen
  - Periodic training

STORMATICS

Challenges in Clustering

- Split brain
- Network latency
- **False alarms**
- Data inconsistency

STORMATICS

False Alarms

Defined

A problem is reported, but in reality, there is no issue

Challenge

Can trigger a failover when one isn’t required, leading to unnecessary disruptions and impacting performance

STORMATICS

False Alarms - Causes

- Network issues
  - Latency, congestion, misconfiguration
- Configuration errors
  - Thresholds set too low?
- Resource constraints
  - High CPU load, memory pressure, I/O bottleneck
- Human error
  - Misreading information, miscommunication of scheduled maintenance, …
- Database locks
  - Long running queries with exclusive locks

STORMATICS

False Alarms - Prevention

- Optimized thresholds
  - Best practices, past experience, and some hit & trial is required to ensure that the thresholds are configured appropriately
- Regular upgrades and testing
  - Latest version of software and firmware to be used
  - Testing of various use cases can help identify possible misconfigurations
- Resource and performance optimization
  - Regularly monitor resource utilization and tune queries and database for performance
  - Maintenance tasks like vacuum, analyze, …
- Comprehensive monitoring and alerting
  - Monitoring can help with early detection of anomalies
  - Alerts can give early warnings as the database approaches defined thresholds

STORMATICS

Challenges in Clustering

- Split brain
- Network latency
- False alarms
- **Data inconsistency**

STORMATICS

Data Inconsistency

Defined

Situations where data in different nodes of a cluster becomes out of sync, leading to inconsistent results and potential data corruption

Challenge

Inaccurate query results that vary based on which node is queried. Such issues are very hard to debug.

STORMATICS

Data Inconsistency - Causes

- Replication lag
  - Network latency and high workloads can be big contributors
  - Data loss in case of failover
- Split brain
- Incorrect configuration
  - Log shipping configurations
  - Replication slots setup
  - Replication filters

STORMATICS

Data Inconsistency - Prevention

- Closely manage asynchronous replication
  - Closely monitor pg_stat_replication for replication lag
  - Place nodes in close proximity and use high quality network hardware
- Regularly check XID across the cluster
- Monitor replication conflicts and resolve promptly
- Regular maintenance and performance optimization
  - Vacuum, analyze, …
  - XID wraparound

STORMATICS

This all sounds really hard

STORMATICS

Open source clustering tools for PostgreSQL

- Repmgr
  - https://repmgr.org/
  - GPL v3
  - Provides automatic failover
  - Manage and monitor replication
- pgpool-II
  - https://pgpool.net/
  - Similar to BSD & MIT
  - Middleware between PostgreSQL and client applications
  - Connection pooling, load balancing, caching, and automatic failover
- Patroni
  - https://patroni.readthedocs.io/en/latest/
  - MIT
  - Template for PostgreSQL high availability clusters
  - Automatic failover, configuration management, & cluster management

STORMATICS

Questions?

pg_umair