BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Dalam melakukan penelitian Optimalisasi Kinerja Server Database PostgreSQL Melalui Cluster Patroni terdapat beberapa sumber bacaan atau referensi dengan pembahasan utama adalah *Cluster Database* PostgreSQL. Berikut ini adalah beberapa sumber penelitian yang didalamnya memuat topik pembahasan dengan penelitian yang dilakukan pada skripsi ini.

Suryanto (2015), melakukan implementasi *Clustering* pada *database server* untuk melakukan optimalisasi kinerja sistem basis data. *Service database* yang digunakan pada penelitian yang dilakukan oleh Suryanto adalah PostgreSQL dan untuk implementasi Clusternya menggunakan teknologi PGCluster. Pada sumber pertama metode pegujian yang dilakukan melalui web *server* dan menguji seberapa *availabilitas* dari web *server* tersebut. Hasil pengujian dua buah pengguna melakukan koneksi dan mengeksekusi sebuah operasi query pada masing-masing sistem secara bersamaan, menunjukkan bahwa waktu yang diperlukan dari sistem *Cluster* lebih sedikit.

Sebastian, Angga, (2015), melakukan implementasi *Clusterisasi database* menggunakan *service database* PostgreSQL dan untuk teknologi *Clustering* yang digunakan adalah PGPool. Dalam penelitian tersebut *Cluster database* digunakan untuk melakukan perbandingan penyimpanan log antara log yang disimpan kedalam

database atau disimpan didalam *file*. Dari hasil pengujian penyimpanan log kedalam *database* memiliki ukuran yang lebih besar dibandingkan dengan disimpan kedalam *file*. Dan untuk *availabilitas* dari *Cluster database* lebih baik dibandingkan tanpa *Cluster database*.

Sugiyatno (2019), dalam jurnalnya yang berjudul *Perancangan Clustering Database Server Untuk Meningkatkan Unjuk Kerja Server Dan Menjamin Ketersediaan Layanan* melakukan penelitian dengan membangun *Cluster database* menggunakan *service* MySQL. Dalam *Cluster* MySQL yang dibangun dilakukan pengujian menggunakan *Response Time* dan *Throughput* melalui *tools* aplikasi *sysbens*. Dari hasil pengujian yang dilakukan diperoleh hasil pengukuran *response time* dan *throughput* menunjukkan performa MySQL Cluster cukup baik dibandingkan dengan MySQL *server default*.

Susanto, Andi, (2021), melakukan penelitian pada Cluster database PostgreSQL melalui *Cluster* Patroni, untuk metode pengujian yang dilakukan melalui *response time* dan *throughput*. Hasil yang didapatkan dari pengujian tersebut hasil pengukuran *response time* naik sebesar 23.13% dibandingkan dengan *single instance* dan *throughput* naik sebesar 27.09% dibandingkan *single instance*.

Sulistiawan, Muhammad Ridho, (2023), melakukan penelitian pada *Cluster database* menggunakan *service database* PostgreSQL dan untuk metode *Clusternya* menggunakan Patroni. Untuk proses pengujian yang dilakukan dari sumber penelitian yang dilakukan adalah melakukan uji *availabilitas* dari *Cluster database* yang diakses

oleh *Cluster* aplikasi *kubernetes*. Melalui pengujian tersebut *availabilitas Cluster database* lebih baik karena sudah terdapat *autofailover* baik dari *Cluster kubernete*s maupun *Cluster database* yang diimplementasikan. Dari beberapa sumber referensi diatas yang berkaitan dengan topik penelitian ini maka dapat diambil perbandingan pada tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

No	Penulis	Teknologi	Hasil
1	Suryanto, (2015)	PostgreSQL, PGCluster	Hasil pengujian dua buah pengguna melakukan koneksi dan mengeksekusi sebuah operasi *query* pada masing-masing sistem secara bersamaan, menunjukkan bahwa waktu yang diperlukan untuk menjalankan *query* pada sistem *Cluster* lebih sedikit dibandingkan dengan *query* yang dijalankan di *standalone* server.
2	Sebastian, Angga, (2015)	PostgreSQL, PGPool	Dari hasil pengujian penyimpanan log kedalam *database* memiliki ukuran yang lebih besar dibandingkan dengan disimpan kedalam *file*. Dan untuk *availabilitas* dari *Cluster database* lebih baik dibandingkan tanpa *Cluster database*.
3	Sugiyatno, (2015)	MySQL	Dari hasil pengujian yang dilakukan diperoleh hasil pengukuran *response time* dan *throughput* menunjukkan performa MySQL *Cluster* cukup baik dibandingkan dengan MySQL *server default*.
4	Susanto, Andi, (2021)	Ansibel, PostgreSQL, Patroni	Hasil yang didapatkan dari pengujian tersebut hasil pengukuran *response time* naik sebesar 23.13% dibandingkan dengan *single instance* dan *throughput* naik sebesar 27.09% dibandingkan *single instance*.
5	Sulistiawan, Muhammad Ridho, (2023)	Kubernetes, PostgreSQL, Patroni	Hasil pengujian yang didapatkan *availabilitas Cluster database* lebih baik karena sudah terdapat *autofailover* baik dari *Cluster kubernete*s maupun *Cluster database* yang diimplementasikan.

No	Penulis	Teknologi	Hasil
6	Apriandi, Rizqi, (2024)	NGINX, PostgreSQL, Patroni	Melalui proses pengujian akan dicari seberapa mampu Cluster Patroni mengelola proses CRUD pada database jika terjadi *downtime* pada server baik *Leader* maupun *Replica*.

Dari beberapa tinjauan pustaka diatas penulis akan mengambil topik teknologi *Clusterisasi* pada pengelolaan sistem database.

2.2. Dasar Teori

2.2.1. Pengertian Database

Database atau dalam bahasa Indonesia disebuat basis data merupakan kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Dengan pengelolaan pada *database* pengguna dapat memperoleh kemudahan dalam menemukan informasi yang sedang dibutuhkan. *Database* juga memiliki pengertian lain yakni sistem yang berfungsi sebagai mengumpulkan file, tabel, atau arsip yang terhubung dan disimpan dalam berbagai media elektronik.

2.2.2. Jenis-Jenis Database

Jenis *database* sendiri dapat dikelompokan menjadi tiga jenis, yaitu:

1. *Relational Database*

   Relational database adalah jenis *database* yang paling umum dan banyak digunakan. *Database* ini mengimplementasikan konsep *relational model*, yang berarti data disimpan dalam tabel-tabel yang saling terkait melalui *primary key* dan

foreign key. *Database relational* menawarkan fleksibilitas dan kemampuan query yang baik, dan sering digunakan untuk aplikasi bisnis yang membutuhkan integritas data yang tinggi. Contoh database relational yang populer adalah MySQL, PostgreSQL, dan Oracle.

2. *NoSQL Database*

NoSQL merupakan singkatan dari *"Not only SQL",* yang menunjukkan bahwa *database* ini tidak hanya dapat mengeksekusi *query SQL*. *NoSQL* mengadaptasi pendekatan *non-relational,* yang berarti data disimpan dalam bentuk tidak terstruktur, seperti *document, graph,* atau *key-value*. Keuntungan dari *NoSQL* adalah skalabilitas dan performa yang baik untuk menangani data yang sangat besar dan tidak terstruktur. Contoh *database NoSQL* yang populer adalah MongoDB, Cassandra, dan Couchbase.

3. *NewSQL Database*

NewSQL adalah jenis database baru yang mencoba untuk menyatukan keunggulan *relational* dan *NoSQL*. *NewSQL* menawarkan performa yang tinggi dan skalabilitas seperti *NoSQL*, serta integritas data dan kemampuan *query* yang baik seperti *relational. NewSQL* sering digunakan untuk aplikasi bisnis yang membutuhkan skalabilitas dan performa tinggi, serta integritas data yang baik. Contoh *database NewSQL* yang populer adalah Amazon Aurora, Google Cloud Spanner, dan CockroachDB.

2.2.3. Pengertian PostgreSQL

PostgreSQL adalah sistem manajemen *database* relasional *(RDBMS)* yang bersifat *open source*. Manajemen *database* ini dapat mengolah data dalam tabel yang memiliki relasi satu sama lain dan dapat digunakan secara gratis serta bebas dimodifikasi.

PostgreSQL dikembangkan oleh Berkeley Computer Science Department, PostgreSQL telah menjadi *database* yang andal dalam 30 tahun terakhir ini. Keunggulan dari sistem manajemen database ini memiliki performa stabil, keamanan tinggi, serta fitur melimpah.

Aplikasi web POStgreSQL memiliki beberapa fungsi untuk mengelola *database*. Berikut ini beberapa fungsi yang dimiliki oleh PostgreSQL, diantaranya :

1. Mengelola Transaksi

   Pertama, fungsi PostgreSQL dapat membantu untuk mengelola transaksi dalam *database*. Fungsi ini dijalankan menggunakan *Data Control Language* (DCL) dengan *query* seperti GRANT, COMMIT, dan REVOKE.
2. Memanipulasi *Value* Data

   Cara kerja PostgreSQL untuk fungsi ini menggunakan *Data Manipulation Language* dengan *query* seperti UPDATE, INSERT, dan DELETE.
3. Membuat dan Memanipulasi Tabel

   Untuk melakukan fungsi ini PostgreSQL menggunakan *Data Definition Language* (DDL) dengan *query* berupa DROP, ALTER, dan CREATE.

2.2.4. Pengertian Patroni

Patroni merupakan *package* python dan *open source* yang dapat digunakan untuk mengelola konfigurasi PostgreSQL. Dalam penelitian ini penulis akan menggunakan Patroni untuk mengelola dan mengoptimalkan kinerja dari server PostrgreSQL,

Dalam menjalankan *service* Patroni, terdapat beberapa *package* pendukung yang harus dijalankan juga, diantaranaya adalah :

1. etcd

   etcd adalah penyimpanan *key value* terdistribusi yang sangat konsisten yang menyediakan cara andal untuk menyimpan data yang perlu diakses oleh sistem terdistribusi atau *Cluster* mesin. etcd merupakan salah satu *package* aplikasi yang dibutuhkan dalam melakukan implementasi pada *Cluster* Patroni.

2. HAProxy

   HAProxy adalah singkatan dari *High Availability Proxy*, yaitu perangkat lunak yang berfungsi sebagai penyeimbang *traffic* untuk aplikasi yang berbasis HTTP atau TCP. Sistem Haproxy bekerja dengan membagi permintaan atau request yang masuk ke beberapa server *backend* yang tersedia. Dengan demikian, beban server akan didistribusikan secara merata diantara server-server tersebut. HAProxy dirancang khusus untuk digunakan dalam lingkungan jaringan yang sibuk dan kompleks. Tujuannya adalah memastikan bahwa setiap permintaan dari klien diproses dengan efisien dan diarahkan ke server *backend* yang paling cocok,

sehingga menjaga ketersediaan aplikasi atau layanan dan memastikan akses yang cepat. HAProxy adalah proyek *open source* yang berada di bawah lisensi GPLv2, yang memungkinkan kita menjalankannya pada berbagai sistem operasi seperti Linux, FreeBSD, dan Solaris. Sama halnya dengan ETCD, HAProxy juga menjadi salah satu *service* aplikasi yang dibutuhkan dalam menjalankan *Cluster* Patroni.

3. Keepalived

   Keepalive adalah *service* yang digunakan untuk menjaga koneksi tetap terbuka untuk sejumlah *request* ke server tertentu atau hingga periode batas waktu permintaan berakhir.