LAPORAN TUGAS 2 - SISTEM TERDISTRIBUSI

Nama: Dhio Anugrah Prakasa Putro

NIM: 11221004

Mata Kuliah: Sistem Paralel dan Terdistribusi

Judul: Implementasi Distributed Synchronization System

1. Deskripsi Singkat

Proyek ini mengimplementasikan sistem sinkronisasi terdistribusi yang mensimulasikan skenario real-world di mana beberapa node saling berkomunikasi dan menjaga konsistensi data bersama. Sistem ini terdiri dari Distributed Lock Manager, Distributed Queue System, dan Distributed Cache Coherence. Tujuan utama adalah memastikan koordinasi antar node tanpa konflik dan menjaga konsistensi data.

2. Arsitektur Sistem

Sistem terdiri dari tiga node utama: node1 sebagai leader (menggunakan algoritma Raft stub), dan dua follower (node2 dan node3). Redis digunakan sebagai shared backend untuk penyimpanan state global. Komunikasi antar-node menggunakan aiohttp dengan pendekatan REST. Setiap node menjalankan API endpoint untuk operasi lock, queue, dan cache.

3. Komponen Utama

Komponen	Fungsi	Algoritma
Distributed Lock Manager	Koordinasi akses resource antar-node	Raft Consensus (Stub)
Distributed Queue System	Distribusi pesan antar producer/consumer	Consistent Hashing + At-Least-Once Delivery
Distributed Cache Coherence	Koherensi cache antar-node	MESI Protocol + LRU

4. Langkah Eksperimen

Langkah-langkah pengujian dilakukan menggunakan PowerShell dan Docker Compose untuk mengelola container multi-node. Setiap node menjalankan endpoint REST yang dapat diuji dengan perintah Invoke-WebRequest.

Contoh uji health check:

``` (Invoke-WebRequest -Uri http://localhost:8081/health).Content ```

Output menunjukkan node1 sebagai leader dan node2/node3 sebagai follower.

Contoh uji Distributed Lock:

``` Invoke-WebRequest -Uri http://localhost:8081/locks/acquire -Method POST ` -Body '{"key":"doc1","client_id":"c1","mode":"S"}' -ContentType 'application/json' ```

5. Hasil dan Analisis

Pengujian menunjukkan sistem berhasil melakukan sinkronisasi antar-node. Leader node mengatur commit log dan follower mengikuti status lock. Deadlock berhasil terdeteksi menggunakan wait-for graph. Cache invalidation bekerja saat PUT dilakukan oleh salah satu node. Metrics berhasil dikumpulkan melalui endpoint /metrics, termasuk data Prometheus untuk Python GC dan custom metrics.

6. Kesimpulan

Implementasi ini memenuhi seluruh spesifikasi core: Distributed Lock Manager, Queue System, Cache Coherence, dan Containerization. Sistem dapat dijalankan pada tiga node, berkomunikasi melalui HTTP REST, dan menampilkan data monitoring via Prometheus. Proyek ini siap untuk dikembangkan ke tahap lanjutan dengan menambahkan PBFT, keamanan TLS, atau adaptive load balancing.

Balikpapan, Oktober 2025

Dhio Anugrah Prakasa Putro