Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

Dokumentasi Teknis — Sistem Sinkronisasi Terdistribusi

Dokumen ini menyajikan dokumentasi teknis dalam Bahasa Indonesia yang menjelaskan desain, implementasi, antarmuka, alur data, konfigurasi, dan panduan operasional proyek yang terdapat di folder Tugas-individu/src/. Tujuan dokumen ini adalah memenuhi kebutuhan pengembang, asisten pengajar, dan penilai untuk memahami perilaku sistem, cara menjalankan, serta tempat memeriksa bagian kode yang relevan.

Catatan: Analisis performa akan disiapkan terpisah di docs/performance_analysis.md setelah benchmark dijalankan.

---

1. Ringkasan Singkat

Sistem menyediakan tiga primitif sinkronisasi utama:

- **Distributed Lock Manager** - kunci eksklusif dan bersama, deteksi deadlock
- **Distributed Queue** - partisi, persistensi, jaminan at-least-once
- **Distributed Cache** - protokol MESI, penggantian LRU

Konsensus dicapai menggunakan algoritma Raft (pemilihan leader, replikasi log, commit). Komunikasi antar-node memakai modul message passing berbasis TCP (asyncio streams). Deteksi kegagalan node menggunakan varian phi-accrual.

Kode sumber utama berada di folder src/.

---

2. Tujuan Dokumen

Dokumen ini menjelaskan hal-hal berikut dengan jelas dan terukur:

1. Struktur proyek dan lokasi berkas penting
2. Deskripsi teknis tiap komponen dan kontrak antarlapis
3. API publik (HTTP) serta tipe pesan internal
4. Alur data end-to-end untuk operasi lock, queue, dan cache
5. Instruksi konfigurasi dan deployment (singkat)
6. Troubleshooting umum dan lokasi log/health checks
7. Rekomendasi pengembangan lanjutan

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

3. Struktur Proyek & Berkas Penting

3.1 Komponen Utama

- src/consensus/raft.py — Implementasi Raft: RaftNode, LogEntry, election, append_entries, commit
- src/nodes/base_node.py — BaseNode yang menyatukan MessagePassing, FailureDetector, Raft, dan HTTP API opsional
- src/nodes/lock_manager.py — DistributedLockManager: logika pengelolaan kunci dan deteksi deadlock
- src/nodes/queue_node.py — DistributedQueue: consistent hashing, partisi, persistensi message
- src/nodes/cache_node.py — DistributedCache: MESI, LRU cache, invalidasi/update

3.2 Komponen Komunikasi

- src/communication/message_passing.py — Message dataclass dan kelas MessagePassing
- src/communication/failure_detector.py — Failure detector (phi-accrual)
- src/api/http_server.py — Server HTTP opsional (dipakai bila diaktifkan)

3.3 Komponen Pendukung

- benchmarks/ — Skrip demo dan benchmark
- docker/ — Dockerfile dan docker-compose untuk cluster 3-node + Redis (opsional)
- tests/ — Unit, integration, dan performance tests (pytest)

---

4. Deskripsi Komponen & Kontrak

4.1 Raft Consensus

Lokasi: src/consensus/raft.py

4.1.1 Tipe Utama

- **RaftState:** FOLLOWER, CANDIDATE, LEADER
- **LogEntry:** term, index, command, data, timestamp
- **RaftNode:** implementasi inti Raft

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

4.1.2 Tanggung Jawab

- Timer election dan inisiasi pemilihan
- RPC handlers: handle_request_vote, handle_append_entries
- Leader actions: kirim heartbeat / append_entries
- Commit & apply entries: _update_commit_index, _apply_committed_entries

4.1.3 Titik Integrasi

- message_sender (callback) untuk mengirim payload RPC ke peer
- state_change_callback & commit_callback untuk notifikasi state/commit

4.1.4 Catatan Penting

Pada mode standalone (cluster_nodes = []), append_log meng-commit segera. Ini nyaman untuk demo, tetapi bukan perilaku cluster multi-node.

4.2 Message Passing

Lokasi: src/communication/message_passing.py

4.2.1 Struktur Message

Message memiliki field: msg_type, sender_id, receiver_id, term, payload, timestamp, message_id

4.2.2 Kelas MessagePassing

- Server TCP menggunakan asyncio.start_server
- Koneksi disimpan keyed by sender_id
- Alamat node diharapkan dalam format node-id:host:port
- Register handler per msg_type (mis. request_vote, append_entries, heartbeat, cache_invalidate)

4.2.3 API Publik

- start() - memulai server
- stop() - menghentikan server
- send_message(target_node, message) - kirim pesan ke target
- broadcast_message() - broadcast ke semua node
- register_handler() - daftarkan handler untuk tipe pesan

4.2.4 Detail Implementasi

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

Saat koneksi masuk, pesan pertama berisi sender_id dipakai untuk mengikat writer ke sender sehingga subsequent messages dikenali.

4.3 Failure Detector

Lokasi: src/communication/failure_detector.py

4.3.1 Mekanisme

Menggunakan sliding-window statistik interval heartbeat untuk menghitung nilai phi.

4.3.2 Node State

- ALIVE
- SUSPECTED
- FAILED

4.3.3 Konfigurasi

Callback tersedia untuk failure & recovery. Parameter yang bisa dikonfigurasi: heartbeat_interval, phi_threshold, window_size, min_samples, dsb.

4.4 BaseNode

Lokasi: src/nodes/base_node.py

4.4.1 Fungsi Utama

- Menyatukan MessagePassing, FailureDetector, dan Raft
- Mendaftarkan handlers yang memanggil Raft RPC handlers saat menerima pesan
- Menyediakan submit_command(command, data) — hanya berfungsi bila node adalah leader

4.4.2 Callback System

Callback commit akan memicu process_committed_entry pada subclass (lock/queue/cache).

4.4.3 HTTP API

HTTP API opsional diinisialisasi bila enable_http_api true. Port default di-deduce dari node-id (konvensi node-1, node-2, ...).

4.5 Distributed Lock Manager

Lokasi: src/nodes/lock_manager.py

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

4.5.1 Struktur State

- locks - resource_id → Lock
- held_locks - client → set resources
- wait_for_graph - graph untuk deteksi deadlock

4.5.2 Alur Operasi

Acquire Lock:

1. acquire_lock(...) membuat LockRequest
2. Submit acquire_lock ke Raft
3. Menunggu grant via perubahan state yang diterapkan pada commit

Release Lock:

- Submit release_lock ke Raft

4.5.3 Deteksi Deadlock

Wait-for graph di-scan periodik menggunakan DFS untuk menemukan siklus. Resolusi memilih abort salah satu partisipan (heuristik youngest).

4.6 Distributed Queue

Lokasi: src/nodes/queue_node.py

4.6.1 Partitioning

partition = hash(queue_name) % partition_count

Konsistent hashing mengalokasikan partition ke node (class ConsistentHash).

4.6.2 Durability

Periodic persistence ke disk (pickle) di persistence_path per node.

4.6.3 Lifecycle Message

Enqueue:

- Submit 'enqueue' command ke Raft
- On commit append ke partition queue

Dequeue:

- Dilakukan oleh owner partition
- mark_delivered & acknowledge mengikuti untuk at-least-once semantics

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

4.7 Distributed Cache

Lokasi: src/nodes/cache_node.py

4.7.1 MESI States

- MODIFIED
- EXCLUSIVE
- SHARED
- INVALID

4.7.2 LRU Cache

Memakai OrderedDict untuk eviction.

4.7.3 Operasi Cache

Get:

- Jika miss, coba fetch dari peer via message passing
- Jika ditemukan, simpan sebagai SHARED

Put:

- Broadcast invalidation ke peer
- Set local line MODIFIED
- Submit cache_put ke Raft untuk durability

4.7.4 Commit Processing

process_committed_entry menerapkan cache_put dan cache_delete setelah commit.

---

5. API REST (Opsional)

Spesifikasi: docs/api_spec.yaml

5.1 Endpoint Umum

Endpoint berikut tersedia apabila HTTP API diaktifkan:

5.1.1 Lock Management

- POST /locks/{lock_id} — mengambil lock
    - Body: { lock_type: "exclusive"|"shared", timeout: ms }
- DELETE /locks/{lock_id} — melepas lock

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

5.1.2 Queue Operations

- POST /queue/{queue_id}/enqueue — enqueue pesan
- POST /queue/{queue_id}/dequeue — dequeue pesan

5.1.3 Cache Operations

- GET /cache/{key} — membaca cache
- PUT /cache/{key} — menulis cache

5.1.4 Health & Status

- GET /health — status node (is_leader, term)

5.2 Pattern Client

Client yang tidak langsung terhubung ke leader harus menemukan leader dahulu (mis. coba /status pada semua node) atau menggunakan client library yang otomatis memilih leader.

Catatan: Implementasi forwarding pada follower belum tersedia secara default.

---

6. Alur Data End-to-End

6.1 Acquire Lock (Exclusive)

1. Client mengirim request ke node A (HTTP API atau client library)
2. Jika node A adalah leader:
    - submit_command('acquire_lock', request)
    - Raft append_log
    - Replikasi ke followers
3. Setelah mayoritas commit:
    - commit_callback memicu DistributedLockManager._process_acquire_lock
    - State lock diperbarui (grant atau antrian)
4. Client yang menunggu akan melihat grant melalui API atau polling state

Catatan: Jika node A bukan leader, client harus menemukan leader dan mengirim request ke leader.

6.2 Enqueue Message

1. Client mengirim enqueue ke node X
2. Node X mengirim enqueue command ke leader Raft (atau jika X leader langsung append)
3. Setelah commit:
    - _process_enqueue menambahkan QueueMessage ke partition queue lokal
    - Persist ke disk periodik
4. Consumer melakukan dequeue pada owner partition
5. Delivery dan ack mengikuti

6.3 Cache Put

1. Client mengirim PUT /cache/{key} ke leader
2. Leader broadcast CACHE_INVALIDATE ke semua peer via MessagePassing
3. Leader memasukkan entry lokal state MODIFIED dan submit cache_put ke Raft
4. Setelah commit, followers menerapkan update melalui process_committed_entry

---

7. Konfigurasi & Environment Variables

7.1 Environment Variables Penting

CLUSTER_NODES

- Format: comma-separated daftar node
- Pattern: node-id:host:port
- Contoh: node-1:localhost:6000,node-2:localhost:6010,node-3:localhost:6020
- **Sangat penting untuk konsistensi alamat**

7.2 Raft Configuration

Lihat docker/docker-compose.yml untuk referensi:

- RAFT_HEARTBEAT_INTERVAL (ms) - interval heartbeat
- RAFT_ELECTION_TIMEOUT_MIN (ms) - timeout minimum election
- RAFT_ELECTION_TIMEOUT_MAX (ms) - timeout maksimum election

7.3 Failure Detector Configuration

- FAILURE_DETECTOR_PHI_THRESHOLD - threshold phi untuk deteksi failure
- FAILURE_DETECTOR_WINDOW_SIZE - ukuran window untuk statistik
- Parameter lainnya sesuai kebutuhan

7.4 HTTP API Configuration

ENABLE_HTTP_API

- Jika true, BaseNode akan menyiapkan HTTP API server
- Port HTTP diturunkan berdasarkan pola node-<n>
- Jika memakai ID berbeda, periksa log HTTP initialization

---

8. Cara Menjalankan

8.1 Standalone (Demo Cepat)

python -m venv venv

source venv/bin/activate # Windows: venv\Scripts\activate.bat

pip install -r requirements.txt

python benchmarks/demo_standalone.py

Pilihan demo:

- 1 = Lock
- 2 = Queue
- 3 = Cache

Catatan: Standalone menggunakan cluster_nodes = [] sehingga node menjadi leader segera.

8.2 Docker (3-Node Cluster)

cd docker

docker-compose build

docker-compose up -d

Monitoring:

Cek logs

docker-compose logs -f

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

Cek health

docker-compose ps

8.3 Manual 3-Proses (Tanpa Docker)

Terminal 1:

```
python -m src.nodes.base_node --node-id node-1 --host 0.0.0.0 --port 6000 \
 --cluster-nodes node-1:localhost:6000,node-2:localhost:6010,node-3:localhost:6020
```

Terminal 2:

```
python -m src.nodes.base_node --node-id node-2 --host 0.0.0.0 --port 6010 \
 --cluster-nodes node-1:localhost:6000,node-2:localhost:6010,node-3:localhost:6020
```

Terminal 3:

```
python -m src.nodes.base_node --node-id node-3 --host 0.0.0.0 --port 6020 \
 --cluster-nodes node-1:localhost:6000,node-2:localhost:6010,node-3:localhost:6020
```

---

9. Monitoring & Logging

9.1 Logging

- Menggunakan modul logging Python
- Output ke stdout
- Terlihat di docker-compose logs bila dijalankan lewat Docker

9.2 Metrics

- Terdapat utilitas metrics di src/utils/metrics.py (counter/gauge)
- Untuk produksi: ekspos endpoint Prometheus di HTTP API
- Konfigurasi Prometheus/Grafana untuk visualisasi

9.3 Health Checks

docker-compose.yml memanfaatkan /status atau endpoint HTTP API untuk healthcheck.

---

10. Troubleshooting & Permasalahan Umum

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

10.1 "Not leader, cannot submit command"

Penyebab: Memanggil submit_command pada follower

Solusi:

- Temukan leader (cek /status)
- Kirim request ke leader
- Atau gunakan demo standalone

10.2 Konflik Port / Proses Python Lama

Solusi:

- Hentikan proses Python yang berjalan pada port terkait
- Lihat QUICK_START.md untuk detail

10.3 Cluster Gagal Elect Leader

Periksa:

- Format CLUSTER_NODES sudah benar
- Konektivitas TCP antar node
- Parameter election timeout

10.4 File Persistensi Queue Tidak Ditemukan

Solusi:

- Pastikan path persistence_path dapat ditulis oleh proses
- Periksa permission dan working directory

---

11. Testing & Quality Assurance

11.1 Unit Tests

pytest tests/unit -v

11.2 Integration Tests

pytest tests/integration -v

11.3 Performance Tests

Menggunakan pytest

pytest tests/performance -v

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

Atau menggunakan benchmark runner

python benchmarks/benchmark_runner.py

Atau menggunakan locust untuk skenario beban kustom

Perhatian: Pastikan tidak ada layanan lain menempati port yang digunakan oleh tes cluster/integrasi.

---

12. Rekomendasi Pengembangan & Peningkatan

12.1 Request Forwarding

Implementasikan request forwarding pada follower sehingga client dapat mengirim request ke follower dan otomatis diteruskan ke leader.

12.2 Dynamic Membership

Tambahkan support dynamic membership (Raft config change) untuk menambah/mengurangi node secara aman.

12.3 Optimisasi Throughput

- Batch append_entries
- Serialisasi cepat (mis. msgpack)
- Persistence asinkron/batched

12.4 Monitoring & Observability

- Ekspos metric Prometheus
- Sediakan template dashboard Grafana
- Monitor p95/p99, commit latency, queue depth

12.5 Keamanan

Tambahkan TLS/auth untuk MessagePassing dan HTTP API demi keamanan produksi.

---

13. Batasan yang Diketahui

13.1 Dynamic Membership

Dynamic membership belum didukung. Membership bersifat statis pada startup.

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050

13.2 Follower Request Handling

submit_command pada follower mengembalikan False. Tidak ada forwarding otomatis.

13.3 Mode Standalone

Mode standalone melakukan commit instan. Ini adalah shortcut demo, bukan perilaku cluster multi-node.

13.4 Keamanan

TLS dan otentikasi tidak diimplementasikan.

---

14. Anchor Kode: Tempat Memeriksa Implementasi

14.1 Raft Core

Lokasi: src/consensus/raft.py

- Logika election: _start_election
- Append entries: _send_append_entries

14.2 Communication Layer

Lokasi: src/communication/message_passing.py

- Transport & handler implementation

14.3 Lock Manager

Lokasi: src/nodes/lock_manager.py

- Alur lock: acquire_lock, _process_acquire_lock

14.4 Queue Management

Lokasi: src/nodes/queue_node.py

- Persistensi queue & consistent hash

14.5 Cache Protocol

Lokasi: src/nodes/cache_node.py

- MESI handlers: _handle_cache_get, _handle_cache_put, _handle_cache_invalidate

---

YOUTUBE: https://youtu.be/c8eUoynUQbI

Nama: Zaky Dio Akbar Pangestu
NIM: 11221050