Analisis Performa — Sistem Sinkronisasi Terdistribusi

Dokumen ini menyajikan analisis performa berdasarkan hasil benchmark yang dijalankan pada cluster 3-node dengan konfigurasi distributed. Hasil mentah disimpan di benchmarks/distributed_http _benchmark_results.json dan visualisasi grafik tersedia dalam bentuk PNG.

Metadata Benchmark:

- Timestamp: 29 Oktober 2024
- Node Cluster: 127.0.0.1:6000, 127.0.0.1:6010, 127.0.0.1:6020
- Mode: Distributed (3-node cluster)
- Jumlah operasi: 50 operasi per primitif

---

1. Ringkasan Eksekutif

Hasil benchmark menunjukkan performa sistem pada konfigurasi distributed 3-node dengan temuan utama:

1.1 Distributed Locks

- **Success Rate:** 0% (0/50 operasi berhasil)
- **Throughput:** 8.26 ops/sec
- **Durasi Total:** 6.05 detik
- **Latency Rata-rata:** 1209.32 ms
- **P50 Latency:** 6.28 ms
- **P95 Latency:** 6024.66 ms
- **P99 Latency:** 6025.86 ms

Analisis: Terdapat masalah serius pada implementasi distributed locks dengan bimodal distribution yang ekstrem - sebagian besar operasi sangat cepat (~6ms) tetapi ~20% operasi mengalami timeout ~6 detik.

1.2 Distributed Queue

- **Messages Enqueued:** 0/50 (gagal)
- **Messages Dequeued:** 0/50 (gagal)
- **Durasi Total:** 0.04 detik
- **Enqueue Throughput:** 0.0 msg/sec

- **Dequeue Throughput:** 0.0 msg/sec
- **Avg Enqueue Latency:** 6.57 ms
- **P50 Enqueue:** 6.73 ms
- **P95 Enqueue:** 8.83 ms

Analisis: Queue operations gagal sepenuhnya meskipun latency pengiriman pesan relatif rendah. Ini mengindikasikan masalah pada level commit atau persistence.

1.3 Distributed Cache

- **Puts:** 40 operasi
- **Gets:** 50 operasi
- **Hit Count:** 0
- **Hit Rate:** 0%
- **Avg PUT Latency:** 3.24 ms
- **Avg GET Latency:** 3.20 ms
- **P50 GET:** 3.10 ms
- **P95 GET:** 4.09 ms

Analisis: Cache menunjukkan latency konsisten dan rendah (3-4ms), namun hit rate 0% mengindikasikan masalah pada cache coherency atau invalidation protocol.

---

2. Metodologi Pengujian

2.1 Lingkungan

- **Konfigurasi:** 3-node distributed cluster
- **Network:** Localhost (127.0.0.1) dengan port berbeda
- **Mode:** Full distributed dengan Raft consensus
- **Script:** benchmarks/local_benchmark.py

2.2 Skema Benchmark

2.2.1 Distributed Locks

- Operasi: Acquire dan release exclusive locks secara berurutan
- Jumlah: 50 operasi total
- Timeout: 6 detik per operasi

2.2.2 Distributed Queue

- Operasi: Enqueue 50 message, kemudian dequeue
- Partitioning: Consistent hashing aktif
- Persistence: Aktif ke disk

2.2.3 Distributed Cache

- Operasi: 40 PUT operations untuk prepopulate, 50 GET operations
- Protocol: MESI cache coherency
- Eviction: LRU policy

2.3 Metrik yang Diukur

- **Throughput:** Operasi per detik
- **Latency:** Waktu respons individual (ms)
- **Percentiles:** P50, P95, P99
- **Success Rate:** Persentase operasi yang berhasil
- **Hit Rate:** (Cache only) Persentase cache hits

2.4 Perintah Reproduksi

python benchmarks/scenarios/distributed_http _benchmark.py --locks 50 --queue 50 --cache 50

---

3. Hasil Detail per Komponen

3.1 Distributed Locks

3.1.1 Metrik Performa

Metrik	Nilai
Total Operations	50
Successful Operations	0
Success Rate	0.0%
Duration	6.05 seconds
Throughput	8.26 ops/sec

Metrik	Nilai
Avg Latency	1209.32 ms
P50 Latency	6.28 ms
P95 Latency	6024.66 ms
P99 Latency	6025.86 ms

3.1.2 Distribusi Latency

Berdasarkan histogram latency (Gambar 1), terlihat pola **bimodal distribution**:

- **Cluster 1:** ~40 operasi dengan latency sangat rendah (≈6ms)
- **Cluster 2:** ~10 operasi dengan latency sangat tinggi (≈6000ms)

Raw Latencies Sample (ms):

Fast operations: 4.86, 5.03, 5.12, 5.19, 5.33, 5.48, 5.68, 5.83, 5.93, 5.98, 6.02, 6.19, 6.21, 6.26, 6.32, 6.40, 6.43, 6.58, 6.61, 6.69, 6.85, 7.28, 7.47, 7.54, 7.95, 8.26, 8.76

Timeout operations: 6013.23, 6020.69, 6020.71, 6022.12, 6023.50, 6023.94, 6023.99, 6024.44, 6024.93, 6026.75

3.1.3 Analisis

Masalah Teridentifikasi:

1. **Timeout pada Commit:** ~20% operasi mengalami timeout 6 detik, mengindikasikan:
    - Kegagalan mencapai quorum Raft
    - Network partition atau leader election issues
    - Deadlock pada wait-for graph
2. **Success Rate 0%:** Meskipun beberapa operasi cepat, tidak ada yang sukses fully complete:
    - Locks acquired tapi tidak ter-commit
    - Leader crashes sebelum replication selesai
    - Issue pada grant/release logic
3. **Throughput Rendah:** 8.26 ops/sec jauh di bawah target (idealnya >100 ops/sec)

Rekomendasi:

- Debug Raft commit path untuk operasi lock
- Verifikasi leader stability dan heartbeat interval
- Review deadlock detection algorithm
- Add detailed logging pada lock state transitions

3.2 Distributed Queue

3.2.1 Metrik Performa

Metrik	Nilai
Messages Requested	50
Messages Enqueued	0
Messages Dequeued	0
Duration	0.04 seconds
Enqueue Throughput	0.0 msg/sec
Dequeue Throughput	0.0 msg/sec
Avg Enqueue Latency	6.57 ms
P50 Enqueue	6.73 ms
P95 Enqueue	8.83 ms
P99 Enqueue	8.93 ms

3.2.2 Distribusi Latency

Enqueue Latencies Sample (ms):

Range: 3.68 - 8.93 ms

Most common: 6-8 ms range

Consistent distribution, no extreme outliers

Dequeue Latencies:

No data - all operations failed

3.2.3 Analisis

Masalah Teridentifikasi:

1. **Complete Failure:** 0/50 messages berhasil enqueued atau dequeued
    - Raft commit tidak terjadi
    - Partition assignment gagal
    - Persistence layer error
2. **Latency Rendah Namun Gagal:** Average 6.57ms mengindikasikan:
    - Message passing berhasil
    - Failure terjadi pada commit phase, bukan network
    - Possible leader tidak stabil
3. **Durasi Total Sangat Singkat:** 0.04 detik untuk 50 operations
    - Operasi fail-fast tanpa retry
    - Timeout handling tidak aktif

Rekomendasi:

- Cek Raft log untuk queue operations
- Verifikasi consistent hashing dan partition ownership
- Review persistence path dan file permissions
- Add retry logic dengan exponential backoff
- Improve error messages untuk debugging

3.3 Distributed Cache

3.3.1 Metrik Performa

Metrik	Nilai
PUT Operations	40
GET Operations	50
Hit Count	0
Hit Rate	0.0%
Avg PUT Latency	3.24 ms
Avg GET Latency	3.20 ms

Metrik	Nilai
P50 GET	3.10 ms
P95 GET	4.09 ms
P99 GET	4.34 ms
PUT Throughput	308.86 msg/sec
GET Throughput	312.97 msg/sec

3.3.2 Distribusi Latency

Berdasarkan histogram (Gambar 4):

PUT Latencies:

- Range: 2.7 - 4.1 ms
- Distribution: Relatively normal, centered around 3.2ms
- No extreme outliers

GET Latencies:

- Range: 2.7 - 4.3 ms
- Distribution: Similar to PUT, consistent performance
- Peak around 2.8-3.0 ms

3.3.3 Analisis

Temuan Positif:

1. **Latency Konsisten:** PUT dan GET keduanya ~3ms
    - Network overhead minimal
    - MESI protocol efficient
    - No contention issues
2. **Throughput Tinggi:** >300 ops/sec untuk PUT dan GET
    - Jauh lebih baik dibanding locks (8.26 ops/sec)
    - Sistem capable untuk high-frequency operations

Masalah Teridentifikasi:

1. **Hit Rate 0%:** Critical issue dengan cache coherency

- Invalidation broadcasts tidak sampai
- State transitions (MESI) tidak berfungsi
- Entries tidak ter-store setelah PUT
- Possible: PUT operations tidak commit ke Raft

2. **Cache Misses 100%:** Semua GET operations miss
   - Data tidak persist setelah PUT
   - Cache lines dalam state INVALID
   - Partitioning atau ownership issues

Rekomendasi:

- Debug MESI state transitions
- Verify cache_put commit to Raft log
- Check invalidation broadcast delivery
- Add cache state dump untuk debugging
- Monitor cache line states (M/E/S/I) per operation

---

4. Perbandingan Single-Node vs Distributed

4.1 Locks: Single vs Distributed

Berdasarkan Gambar 8:

Mode	Success Rate	Throughput (ops/sec)
Single-node	~0% (visualization issue)	~400
Distributed (3-node)	0%	8.26

Analisis:

- Distributed mode mengalami **penurunan throughput drastis** (~48x slower)
- Success rate rendah pada kedua mode mengindikasikan bug fundamental
- Overhead Raft consensus signifikan (dari 400 ke 8.26 ops/sec)

4.2 Cache: Single vs Distributed

Berdasarkan Gambar 5 (Hit Rate):

Mode	Hit Rate
Single-node	~0%
Distributed	0%

Analisis:

- Cache hit rate konsisten 0% di kedua mode
- Issue bukan dari distributed coordination, tapi dari cache logic itself
- MESI protocol atau persistence layer bermasalah

4.3 Latency Percentiles Across Primitives

Berdasarkan Gambar 7:

Primitive	P50	P95	P99
Locks	~0ms	~6000ms	~6000ms
Queue Enqueue	~0ms	~0ms	~0ms
Queue Dequeue	~0ms	~0ms	~0ms
Cache PUT	~0ms	~0ms	~0ms
Cache GET	~0ms	~0ms	~0ms

Catatan: Locks menunjukkan outliers ekstrem pada P95/P99 (6 detik), sementara primitif lain konsisten rendah.

4.4 Throughput Comparison

Berdasarkan Gambar 9:

Operation	Throughput
Locks	8.26 ops/sec
Queue Enqueue	0.0 msg/sec
Queue Dequeue	0.0 msg/sec
Cache PUT	308.86 msg/sec
Cache GET	312.97 msg/sec

Analisis:

- Cache operations 37x lebih cepat dari locks
- Queue completely failed (0 throughput)
- Cache menunjukkan sistem capable untuk high throughput jika commit logic fixed

---

5. Interpretasi & Root Cause Analysis

5.1 Raft Consensus Issues

Evidence:

- Locks: 20% timeout pada 6 detik (election timeout?)
- Queue: 0% success meskipun message passing cepat
- Cache: High throughput tapi 0% hit rate (commit issues?)

Possible Root Causes:

1. **Leader Instability**
    - Frequent leader elections
    - Split-brain scenarios
    - Network partition detection issues
2. **Commit Index Not Advancing**
    - Majority quorum tidak tercapai
    - AppendEntries RPC failures
    - Log replication bottleneck
3. **Apply Committed Entries Failure**
    - Entries committed tapi tidak di-apply ke state machine
    - Callback not triggered
    - State machine update bugs

Diagnostic Steps:

Check Raft state on each node

GET /status → { is_leader, term, commit_index, last_applied }

Verify log consistency

Compare commit_index across all 3 nodes

Monitor leader elections

Check election timeout frequency

5.2 Network & Message Passing

Evidence:

- Low latencies (3-8ms) untuk message passing
- No extreme network delays
- Consistent performance across operations

Conclusion: Network layer berfungsi dengan baik. Masalah bukan di transport layer.

5.3 State Machine Logic

Lock Manager:

- Deadlock detector mungkin too aggressive?
- Wait-for graph corruption?
- Grant logic bug?

Queue:

- Partition ownership ambiguous?
- Persistence path errors?
- Dequeue logic never executes?

Cache:

- MESI invalidation tidak broadcast?
- State transitions incomplete?
- LRU eviction premature?

---

6. Masalah Kritis & Prioritas Perbaikan

6.1 Priority 1: Critical Blockers

Issue 1: Raft Commit Failures

- **Impact:** All primitives affected
- **Symptoms:** 0% success rates, timeout pada locks
- **Action:**
    - Add detailed Raft logging (DEBUG level)
    - Verify leader stability over time
    - Check network connectivity between nodes
    - Review commit_index update logic

Issue 2: Cache Hit Rate 0%

- **Impact:** Cache unusable despite good latency
- **Symptoms:** All GETs miss, PUTs don't persist
- **Action:**
    - Debug cache_put commit to Raft
    - Verify MESI state after PUT
    - Check if apply_committed_entry called
    - Add state dumps after each operation

6.2 Priority 2: Performance Issues

Issue 3: Lock Timeouts (6 seconds)

- **Impact:** 20% operations timeout
- **Symptoms:** Bimodal distribution
- **Action:**
    - Profile lock acquisition path
    - Check deadlock detector frequency
    - Optimize wait-for graph operations
    - Consider shorter timeout with retry

Issue 4: Queue Complete Failure

- **Impact:** 0/50 operations succeed
- **Symptoms:** Fast latency but no commit
- **Action:**

    - Debug enqueue command in Raft log
    - Verify partition assignment
    - Check persistence layer
    - Add retry mechanism

6.3 Priority 3: Optimizations

Issue 5: Low Lock Throughput (8 ops/sec)

- **Target:** >100 ops/sec
- **Action:**
    - Batch lock operations
    - Reduce Raft overhead (batch AppendEntries)
    - Optimize serialization (msgpack)
    - Parallel lock processing

Issue 6: Monitoring Gaps

- **Action:**
    - Expose Prometheus metrics
    - Add CPU/memory profiling
    - Track GC pauses
    - Monitor network RTT

---

7. Rekomendasi Eksperimen Lanjutan

7.1 Debugging Experiments

Experiment 1: Single Operation Trace

```
# Run 1 lock operation with full DEBUG logging
python benchmark.py --locks 1 --debug

# Trace through:
# 1. Client send request
# 2. Leader receive
```

3. Raft append_log

4. Replicate to followers

5. Commit

6. Apply to state machine

7. Response to client

Experiment 2: Leader Stability Test

Monitor leader over 60 seconds

watch -n 1 'curl localhost:8000/status'

Count leader changes

Verify no split-brain

Experiment 3: Network Partition Simulation

Block traffic between node-1 and node-2

iptables -A INPUT -s 127.0.0.1:6010 -j DROP

Observe:

- Leader election

- Quorum behavior

- Client request handling

7.2 Performance Optimization Experiments

Experiment 4: Scaling Test

- 1 node, 3 nodes, 5 nodes
- Measure throughput vs node count
- Identify scalability bottlenecks

Experiment 5: Load Test

Use locust for concurrent clients

locust -f benchmark_locust.py --users 100 --spawn-rate 10

Measure:

- Throughput under load

- Latency distribution

- Resource usage (CPU, memory)

Experiment 6: Persistence Impact

Test with persistence ON vs OFF

Queue: measure enqueue latency difference

Cache: measure PUT latency difference

7.3 Metrik Target (Revised)

Setelah fixing issues di atas, target performa realistis:

Primitive	Throughput	P95 Latency	Success Rate
Locks	>100 ops/sec	<50ms	>99%
Queue (enqueue)	>1000 msg/sec	<20ms	>99.9%
Queue (dequeue)	>800 msg/sec	<30ms	>99%
Cache PUT	>500 ops/sec	<10ms	>99.9%
Cache GET	>1000 ops/sec	<5ms (hit)	N/A
Cache Hit Rate	N/A	N/A	>80%

---

8. Kesimpulan

8.1 Temuan Utama

1. **Sistem Mengalami Critical Issues:**
    - Success rate 0% untuk locks dan queue
    - Cache hit rate 0% meskipun latency baik
    - Raft consensus tidak berfungsi dengan benar
2. **Network Layer Berfungsi:**
    - Latencies rendah (3-8ms) dan konsisten
    - Message passing reliable

- Problem bukan di transport layer

3. **Throughput Variance Ekstrem:**
    - Cache: 300+ ops/sec (bagus)
    - Locks: 8 ops/sec (sangat rendah)
    - Queue: 0 ops/sec (gagal total)

8.2 Action Items

Immediate (Sprint 1):

1. Fix Raft commit logic - investigate why entries tidak di-apply
2. Debug lock timeout issues - focus pada 6-second timeouts
3. Fix queue enqueue failures - check partition ownership
4. Resolve cache hit rate 0% - debug MESI state transitions

Short-term (Sprint 2): 5. Add comprehensive logging dan metrics 6. Implement retry mechanisms dengan backoff 7. Optimize lock throughput (batching, parallelization) 8. Add integration tests untuk Raft consensus

Long-term (Sprint 3+): 9. Performance optimization (serialization, batching) 10. Scalability testing (5-node, 7-node clusters) 11. Production hardening (TLS, auth, monitoring) 12. Chaos engineering tests (network partition, node failure)

8.3 Catatan Akhir

Sistem menunjukkan **potential** yang baik (cache throughput 300+ ops/sec, latencies 3-8ms), namun memiliki **critical bugs** pada Raft consensus layer yang menghalangi fungsi dasar. Focus harus pada **stabilitas dan correctness** sebelum optimisasi performa.

Prioritas:

1. Correctness (fix success rate 0%)

2. Stability (fix timeouts dan failures)

3. Performance (optimize throughput)

4. Scalability (test dengan lebih banyak nodes)

Appendix A: Raw Data Reference

A.1 JSON Result Location

benchmarks/scenarios/distributed_http _benchmark_result.json

A.2 Visualization Plots

- locks_latency_histogram.png - Lock latency distribution
- locks_summary_bar.png - Lock success rate & throughput
- cache_hit_rate.png - Cache hit rate comparison
- cache_latency_histograms.png - PUT/GET latency distributions
- cache_single_vs_distributed.png - Cache hit rate comparison
- locks_single_vs_distributed.png - Lock throughput comparison
- latency_percentiles_comparison.png - P50/P95/P99 across primitives
- throughput_comparison.png - Throughput across all primitives

A.3 Configuration Used

Cluster Nodes:

- 127.0.0.1:6000

- 127.0.0.1:6010

- 127.0.0.1:6020

Raft Config:

- Heartbeat Interval: 50ms

- Election Timeout: 150-300ms

- Log Replication: Enabled

Benchmark Config:

- Locks: 50 operations

- Queue: 50 messages

- Cache: 40 PUTs + 50 GETs
