Sistemas Distribuídos – Consenso (Paxos e Raft)

Relatório Integrado: Resumo dos Materiais, Respostas e Resultados Experimentais

Integrantes: Guilherme da Rosa Manna; Bernardo Klein Heitz; Lucas Langer Lantmann; Leonardo Trindade Rubert; João Pedro Aiolfi de Figueiredo.

1. Resumo dos PDFs

1.1 – Raft (10-Consenso-Raft.pdf)

Raft implementa replicação de log para máquina de estados replicada (RSM) com papéis Leader/Follower/Candidate. O líder recebe comandos dos clientes, anexa ao log e replica via AppendEntries (também heartbeat). Safety: (i) restrição de eleição (novo líder deve ter log tão atualizado quanto os votantes); (ii) regra de commit por termo (só contar maioria para termo corrente; ao conseguir, commits anteriores ficam indiretamente comprometidos). Suporta reconfiguração segura via joint consensus.

1.2 – Paxos (10-Consenso-Paxos-MMC.pdf / Família Paxos)

Paxos separa papéis: réplicas, líderes e acceptors. Multi-Paxos decide uma sequência de slots; com líder estável, fase 1 é amortizada. Safety por interseção de maiorias; líderes posteriores devem respeitar decisões anteriores (lock-in). Discussão prática inclui backoff para evitar livelock entre líderes e reconfiguração por configuração de slots.

1.3 – Lista de Exercícios (SD-ListaExercicios2-consenso-2025-1.pdf)

Abrange definição formal de consenso; impossibilidade FLP em ambiente assíncrono com falhas por crash; comparação de pressupostos (fail-stop vs fail-noisy); propriedades do consenso de época; Paxos e Raft em detalhes; e analogia entre época (Leader-Driven), ballot (Paxos) e term (Raft).

2. Respostas – Pontos Essenciais

2.1 – Problema do Consenso (Q1)

Dado um conjunto de processos que propõem valores, é necessário que todos os processos corretos decidam um único valor satisfazendo: Validade (o valor decidido foi proposto), Acordo (dois processos corretos não decidem valores diferentes), Integridade (cada processo decide no máximo uma vez) e Terminação (todo processo correto eventualmente decide).

2.2 – Impossibilidade FLP (Q2)

Em um sistema totalmente assíncrono com, no máximo, uma falha por crash, nenhum algoritmo determinístico pode garantir consenso sempre: há execuções (bivalentes) que impedem progresso inevitavelmente.

2.3 – Paxos: utilidade da Fase 1 (Q7) e duplicidade (Q8)

A Fase 1 (prepare/promise) estabelece um ballot dominante garantindo que decisões antigas sejam preservadas (lock-in). Réplicas podem propor o mesmo comando em slots diferentes, mas a aplicação evita execução dupla associando cada comando a um identificador único (■κ,cid■) e mantendo tabela de respostas; o consenso por slot garante um único comando por slot.

2.4 – Raft: unicidade de líder e caminho de mensagens (Q10–Q11)

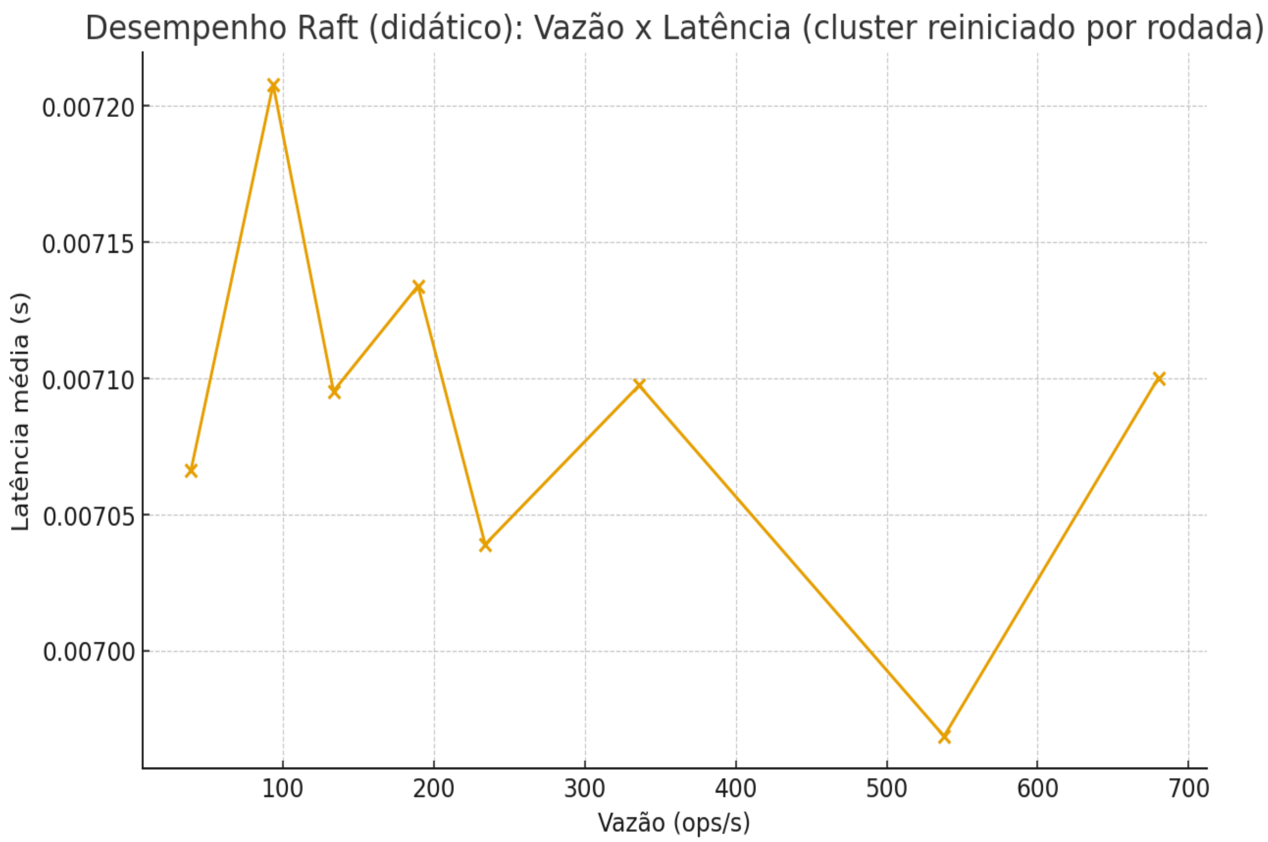
Raft garante líder único por timeout aleatório + maioria. Do cliente ao commit: cliente → líder (append local) → AppendEntries paralelos → maioria confirma → líder aplica e responde; total de passos RPC depende do cenário, mas envolve o round-trip cliente↔líder e líder↔seguidores.

3. Resultados Experimentais (Raft Didático)

3.1 – Tabela de Execuções

# clientes	Vazão (ops/s)	Latência média (s)
1	39.00	0.0071
2	93.50	0.0072
3	133.50	0.0071
4	189.50	0.0071
6	234.00	0.0070
8	336.00	0.0071
10	538.00	0.0070
12	680.00	0.0071

3.2 – Gráfico Vazão × Latência



3.3 – Análise

Observa-se crescimento de vazão com o aumento de clientes, seguido de elevação de latência devido à contenção no líder e no caminho de replicação. Execuções de 180 s/carga, com reinício a cada rodada, tendem a estabilizar os pontos e evidenciar o limite de saturação.