Douglas Pereira Luiz

Relatório de Avaliação Experimental: Biblioteca Repart-KV e a Técnica de Hard Partitioning

1 de fevereiro de 2026

Douglas Pereira Luiz

Relatório de Avaliação Experimental: Biblioteca Repart-KV e a Técnica de Hard Partitioning

1 de fevereiro de 2026

1 Introdução

Sistemas de armazenamento distribuídos e paralelos frequentemente utilizam o particionamento de estado para escalar o desempenho e a capacidade de armazenamento. Entretanto, o particionamento estático pode levar a desequilíbrios de carga quando os padrões de acesso mudam ao longo do tempo. Para mitigar esse problema, técnicas de reparticionamento dinâmico buscam reconfigurar o esquema de partições durante a execução, visando manter o equilíbrio e a eficiência do sistema.

A biblioteca `repart-kv` foi desenvolvida para consolidar e estender as técnicas de reparticionamento dinâmico de baixo impacto (*stall-free*) propostas em trabalhos anteriores [Luiz e Mendizabal 2024, Luiz e Mendizabal 2024, Luiz e Mendizabal 2025]. Diferente de protótipos anteriores, a `repart-kv` oferece uma interface padronizada semelhante a bancos de dados embarcados (*embedded databases*) e introduz a estratégia de *hard partitioning*, que permite o particionamento físico dos dados em múltiplas instâncias de motores de armazenamento.

Este relatório apresenta uma avaliação experimental inicial da biblioteca `repart-kv`, na forma de *microbenchmaking* com foco na análise de desempenho da técnica de *hard partitioning* sob diferentes cargas de trabalho e configurações.

O restante deste documento está organizado da seguinte forma: a Seção 2 descreve os objetivos da avaliação; a Seção 3 apresenta a metodologia adotada; a Seção 4 detalha a configuração dos experimentos; a Seção 5 apresenta e analisa os resultados obtidos; e, por fim, a Seção 6 discute as conclusões e trabalhos futuros.

2 Objetivos

O principal objetivo desta avaliação experimental é realizar uma verificação inicial do desempenho e da viabilidade da técnica de *hard partitioning* implementada na biblioteca `repart-kv`. Busca-se observar como a biblioteca se comporta em termos de vazão (*throughput*) quando submetida a diferentes cargas de trabalho do benchmark YCSB, comparando o impacto do uso de múltiplas partições físicas e do intervalo de reparticionamento em relação ao uso direto do motor de armazenamento (*baseline*).

3 Metodologia

Para a realização deste trabalho, a metodologia seguiu as seguintes etapas:

1. **Seleção de Soluções**: Foram selecionadas soluções comerciais de *key-value store* local (LMDB e Tkrzw) para servirem como motores de armazenamento subjacentes.

2. **Definição de Cargas de Trabalho**: Utilizou-se o benchmark YCSB (Yahoo! Cloud Serving Benchmark) com as cargas A e D, que representam diferentes padrões de acesso comuns em sistemas reais.

3. **Definição de Parâmetros**: Foram estabelecidos parâmetros de teste, como o número de trabalhadores (*workers*), número de partições e intervalos de reparticionamento, para verificar a sensibilidade da solução.

4. **Execução dos Experimentos**: Os experimentos foram automatizados através de scripts Bash (`experiment_set.sh`), garantindo a repetibilidade dos testes (5 repetições para cada configuração).

5. **Compilação e Análise**: Os resultados coletados em formato CSV foram processados para a geração de gráficos de *throughput*, seguidos de uma análise dos dados.

4 Configuração dos Experimentos

Nesta seção, detalhamos o ambiente experimental e os parâmetros utilizados para a avaliação da biblioteca `repart-kv`. A configuração abrange a escolha dos motores de armazenamento subjacentes, a caracterização das cargas de trabalho sintéticas e a definição dos parâmetros operacionais que guiaram a execução dos testes.

4.1 Motores de Armazenamento

Foram utilizadas duas soluções de armazenamento persistente amplamente reconhecidas:

- **LMDB (Lightning Memory-Mapped Database)**: Um banco de dados baseado em árvores B+ que utiliza mapeamento de memória (*mmap*), oferecendo transações ACID e alta eficiência em leituras. Nos experimentos, o LMDB foi utilizado em sua configuração padrão, sem compressão.

- **Tkrzw**: Uma biblioteca moderna que sucede o Kyoto Cabinet, oferecendo diversas implementações de estruturas de dados. Para este trabalho, utilizou-se a implementação `TreeDBM` (árvore B+), configurada especificamente para utilizar compressão *zlib*, visando observar o impacto da compressão no desempenho e uso de disco.

4.2 Cargas de Trabalho

As cargas de trabalho foram derivadas do YCSB [Cooper et al. 2010], conforme detalhado em [Luiz e Mendizabal 2025]:

- **Carga A (Update-heavy)**: Composta por 50% de leituras e 50% de atualizações. Representa cenários como o armazenamento de sessões de usuários, onde as informações são frequentemente lidas e atualizadas.

- **Carga D (Read-latest)**: Composta por 95% de leituras e 5% de inserções. Simula aplicações de redes sociais onde os usuários acessam predominantemente os dados mais recentemente inseridos (ex: *trending topics*).

4.3 Parâmetros de Teste

Os experimentos foram configurados com os seguintes parâmetros, extraídos de `experiment_set.sh` e `src/repart_kv.cpp`:

- **Trabalhadores (*workers*)**: 1 e 2 threads. Cada *worker* é uma *thread* dedicada que executa todas as operações definidas no arquivo de carga de trabalho;

- **Partições**: 1 e 8 partições.
- **Intervalo de Reparticionamento**: 0 (desabilitado) e 1 segundo.
- **Repetições**: 5 execuções para cada cenário.

4.4 Ambiente de Testes

Os experimentos foram conduzidos em uma workstation com a seguinte configuração:

- **Processador**: Intel Xeon E5-2683 v4 (16 núcleos físicos (*Hyper-Threading foi desabilitado para os experimentos*), 2,10 GHz, 40 MB de cache);
- **Memória**: 64 GB RAM DDR4 2400 MHz ECC RDIMM;
- **Sistema Operacional**: Ubuntu 24.04 LTS (kernel Linux 6.14.0-37-generic);
- **Compilador**: GCC 13.3.0 (padrão C++20).

5 Resultados

Nesta seção, apresentamos os resultados de vazão (*throughput*) obtidos para as diferentes configurações, analisando primeiro a carga A e, em seguida, a carga D. Os resultados foram compilados em gráficos que apresentam a vazão no eixo Y (em operações por segundo) e o número de *workers* no eixo X.

5.1 Análise do Throughput com a carga A (Update-heavy)

A carga A apresenta um desafio para os motores de armazenamento devido à alta frequência de atualizações (50%). Os resultados agregados mostram comportamentos distintos entre os motores Tkrzw e LMDB.

5.1.1 Tkrzw com carga A

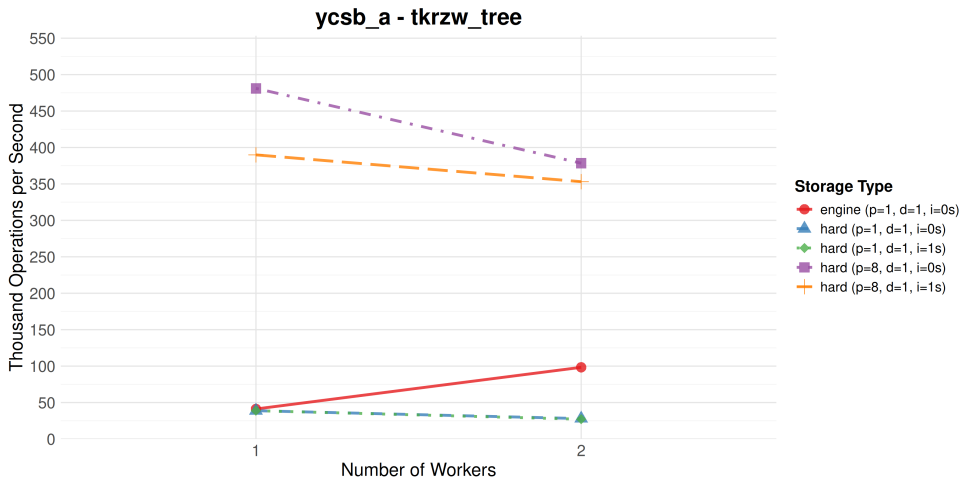

Figura 1 – Vazão para a carga A utilizando o motor Tkrzw (Eixo X: número de *workers*; Eixo Y: vazão em ops/s). Valores maiores de vazão indicam melhor desempenho.

Para o motor Tkrzw (Figura 1), observa-se que o uso de *hard partitioning* com 8 partições elevou significativamente a vazão em comparação ao *baseline* (`engine`). Com 1 trabalhador, a vazão saltou de aproximadamente 41 mil ops/s no *baseline* para mais de 481 mil ops/s com 8 partições físicas. Mesmo com 2 trabalhadores, a configuração de 8 partições manteve uma vazão superior (378 mil ops/s) à do motor direto (98 mil ops/s). O intervalo de reparticionamento de 1000ms causou uma redução aceitável na vazão (de 481 mil para 389 mil ops/s com 1 *worker*), refletindo o custo do rastreamento em uma carga intensiva de escrita.

5.1.2 LMDB com carga A

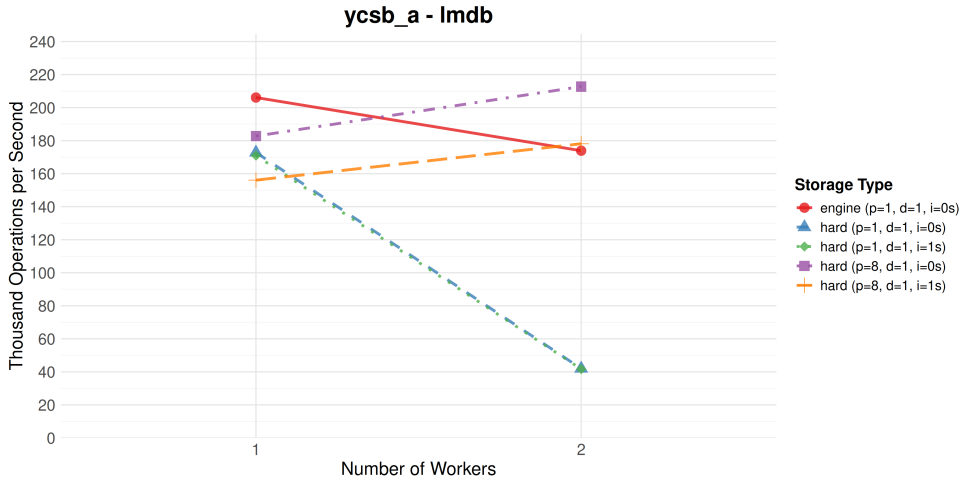

Figura 2 – Vazão para a carga A utilizando o motor LMDB (Eixo X: número de *workers*; Eixo Y: vazão em ops/s). Valores maiores de vazão indicam melhor desempenho.

No caso do LMDB (Figura 2), o motor direto apresentou uma vazão robusta de 206 mil ops/s com 1 trabalhador. A técnica de *hard partitioning* com 8 partições e 2 trabalhadores conseguiu superar o *baseline* de 2 trabalhadores (173 mil ops/s), atingindo 212 mil ops/s. Isso sugere que a segmentação física ajuda a mitigar a contenção de escrita do LMDB quando múltiplos trabalhadores estão ativos.

5.2 Análise do Throughput com a carga D (Read-latest)

A carga D, sendo predominantemente de leitura (95%), permite observar a eficiência do sistema em cenários de baixo conflito de escrita.

5.2.1 Tkrzw com carga D

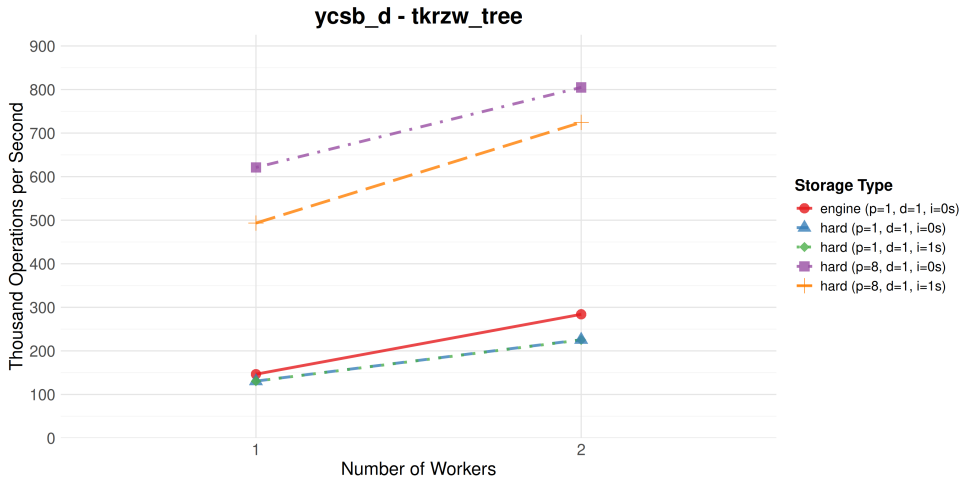

Figura 3 – Vazão para a carga D utilizando o motor Tkrzw (Eixo X: número de *workers*; Eixo Y: vazão em ops/s). Valores maiores de vazão indicam melhor desempenho.

Na Figura 3, o Tkrzw demonstrou um ganho expressivo com o *hard partitioning*. Com 2 trabalhadores, a vazão do motor direto foi de 283 mil ops/s, enquanto a configuração com 8 partições atingiu picos de 804 mil ops/s. O uso do intervalo de reparticionamento de 1000ms manteve o desempenho em patamares elevados (724 mil ops/s), validando a baixa sobrecarga da técnica para cargas de leitura.

5.2.2 LMDB com carga D

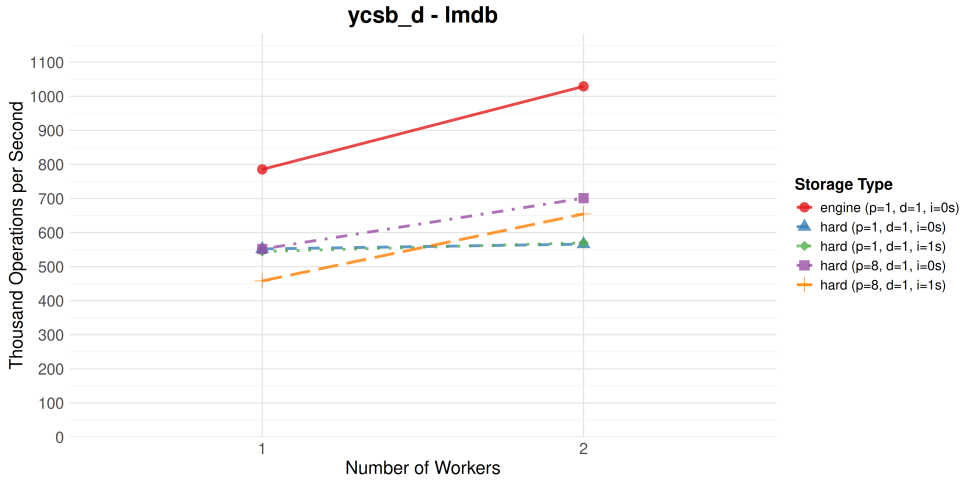

Figura 4 – Vazão para a carga D utilizando o motor LMDB (Eixo X: número de *workers*; Eixo Y: vazão em ops/s). Valores maiores de vazão indicam melhor desempenho.

Para o LMDB (Figura 4), o motor direto apresentou o melhor desempenho absoluto, ultrapassando 1 milhão de ops/s com 2 trabalhadores. Embora o *hard partitioning* com 8 partições tenha apresentado uma vazão menor que o *baseline* (701 mil ops/s), ele superou a configuração

de partição única (566 mil ops/s), indicando que a distribuição física dos dados ainda oferece benefícios de escalabilidade interna frente ao particionamento lógico simples.

6 Discussão

A avaliação experimental inicial da biblioteca `repart-kv` demonstrou que a técnica de *hard partitioning* é funcional e apresenta um desempenho promissor frente aos motores de armazenamento convencionais. A separação física das partições em múltiplas instâncias de motores permitiu manter a vazão competitiva, mesmo com a introdução de camadas de abstração e mecanismos de rastreamento de carga.

Os resultados indicam que, para cargas intensivas de atualização (carga A), o *hard partitioning* pode oferecer benefícios ao reduzir a contenção em motores que utilizam travas globais ou de granularidade grossa para escrita. Para cargas de leitura (carga D), a sobrecarga do sistema de reparticionamento mostrou-se negligenciável.

Como trabalhos futuros, pretende-se:

- Estender a avaliação experimental utilizando uma gama maior de cargas de trabalho do YCSB (B, C, E e F).
- Avaliar outras soluções comerciais como LevelDB ou WiredTiger.
- Explorar o potencial de E/S paralelo da técnica *hard partitioning* ao distribuir as instâncias de armazenamento em diferentes dispositivos físicos de armazenamento.
- Investigar o impacto de diferentes algoritmos de particionamento além do METIS.

Referências

[Cooper et al. 2010]COOPER, B. F. et al. *Benchmarking Cloud Serving Systems with YCSB*. 2010. ACM Symposium on Cloud Computing (SoCC).

[Luiz e Mendizabal 2024]LUIZ, D. P.; MENDIZABAL, O. M. Balanceamento de carga com reparticionamento contínuo em sistemas com estado particionado. In: *Proceedings of the Encontro Regional de Computação (ERAD IC) 2024*. [S.l.: s.n.], 2024. p. 31–38.

[Luiz e Mendizabal 2024]LUIZ, D. P.; MENDIZABAL, O. M. Lightweight asynchronous repartitioning for local state partitioned systems. In: *Proceedings of the 2024 Symposium on System-Level Performance and Availability (SSCAD)*. [S.l.: s.n.], 2024. p. 1–8.

[Luiz e Mendizabal 2025]LUIZ, D. P.; MENDIZABAL, O. M. Stall-free asynchronous state repartitioning with a proactive workload tracking window. *Computers and Concurrent Engineering*, v. 11, n. 2, p. 134–147, 2025.