Benchmark Report for memFS (CL Project)

Gana Jayant Sigadam
Roll No: 24CS60R12

November 17, 2024

1 Benchmarking Setup

The benchmarking tests for MemFS were conducted using the following workloads:

- **Workload 100**: 100 operations each of Create, Write, Read, Delete.
- **Workload 1000**: 1000 operations each of Create, Write, Read, Delete.
- **Workload 10000**: 10000 operations each of Create, Write, Read, Delete.

Tests were run with thread counts of 1, 2, 4, 8, and 16, and the following metrics were measured:

- Time per operation (Create, Write, Read, Delete)
- CPU Utilization
- Memory Usage

2 Measuring Performance

So for measuring the latency of the operations, I have used the `std::chrono` library in C++ to measure the time taken by each operation. The CPU Utilization was measured using `/proc/stat` file in Linux which gives user, nice, system, idle, iowait, irq, softirq, steal jiffies. The CPU Utilization is calculated as follows: we need to calculate the total jiffies and work jiffies for the start and end of the operation. The CPU Utilization is calculated as follows:

$$\text{CPU Utilization} = \frac{\text{work jiffies}_{\text{end}} - \text{work jiffies}_{\text{start}}}{\text{total jiffies}_{\text{end}} - \text{total jiffies}_{\text{start}}} \times 100$$

where work jiffies is calculated as follows:

$$\text{work jiffies} = \text{user} + \text{nice} + \text{system}$$

and total jiffies is calculated as follows:

$$\text{total jiffies} = \text{user} + \text{nice} + \text{system} + \text{idle} + \text{iowait} + \text{irq} + \text{softirq} + \text{steal}$$

I have referred to the following link for the calculation of CPU Utilization StackOverflow Link

3 Performance Benchmarking Results

Table 1: Performance Metrics Across Thread Counts and Workloads

Threads	Workload	Create (ms)	Write (ms)	Read (ms)	Delete (ms)	CPU Util (%)	Memory (KB)
1	100	15	10	30	6	18.03	6732
	1000	22	10	33	24	15.91	7172
	10000	22	14	34	6	18.42	7300
2	100	29	18	35	10	17.78	7804
	1000	31	19	39	10	14.74	7932
	10000	29	20	33	10	17.05	7932
4	100	38	23	31	16	21.15	8572
	1000	36	24	33	16	20.39	8572
	10000	37	23	34	16	18.18	8572
8	100	42	25	31	18	27.45	9084
	1000	42	26	32	30	25.86	9084
	10000	41	26	33	19	30.19	9084
16	100	43	26	32	20	39.81	9724
	1000	37	27	32	20	38.68	9724
	10000	45	25	34	20	40.18	9724

This benchmarking is done in a system which contains 10 hardware threads.

4 Performance Analysis

- Create operation takes the most time among all the multi-threaded operations because it needs to allocate File object in the hashmap tree. note read is not a multi-threaded operation.
- Multi-threading is really helpful for larger workloads as it reduces the time taken for the operations. Example for 16 threads, the time taken for 10000 write operations is 25ms which is less compared to 1000 write operations which is 26ms. But it is very small difference.
- Read operation takes same time across different workloads because it is not a multi-threaded operation.
- CPU Utilization is increasing with the increase in the number of threads because more threads are running in parallel.