**Design Document: Parallel Line Processing with Pthreads**

**Group Number 10**

**Overview**

The objective of this project is to parallelize the task of analyzing a large Wikipedia dump file (~1.7GB, ~1 million lines) by finding the maximum ASCII character value on each line. This is achieved using POSIX threads (pthreads) in C.

The input file is read into memory, then the lines are distributed across multiple threads. Each thread computes the maximum ASCII value for its assigned lines. The result is stored in a global array and printed in order.

**Software Architecture**

* **Language & Libraries:** C with pthreads
* **Major Components:**

1. read\_file() — Reads and stores all lines into a dynamically resized array.
2. process\_lines() — Thread function that computes max ASCII per line.
   1. main() — Sets up threading, joins threads, and handles output.

* **Thread Partitioning Strategy:** Each thread processes approximately total\_lines / NUM\_THREADS lines. The final thread handles any remaining lines due to integer division.
* **Global Structures:**
  1. char \*\*lines — Stores all file lines

1. int \*results — Stores max ASCII per line

o

**Performance Evaluation**

The implementation was tested with 1 million lines using /usr/bin/time ./pthread ~dan/625/wiki\_dump.txt. The output below summarizes the results.

**Threads Elapsed Time (s) CPU % User Time System Time**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 10.33 | 98% | 4.86 | 5.29 |
| 2 | 7.82 | 118% | 4.88 | 4.38 |
| 4 | 9.59 | 117% | 5.60 | 5.65 |
| 8 | 8.63 | 123% | 6.04 | 4.63 |
| 12 | 10.08 | 119% | 5.90 | 6.12 |
| 16 | 8.58 | 121% | 5.70 | 4.72 |
| 20 | 12.84 | 104% | 6.20 | 7.17 |

**Analysis:**

* Best performance was achieved with 2–8 threads, where elapsed time was lowest, and CPU utilization was high.
* Performance degraded at 12 and 20 threads, indicating oversubscription and increased overhead.
* CPU usage >100% validates use of multiple cores.
* Slight irregularity at 4 threads may indicate imbalanced workload or thread scheduling artifacts.

**Next Steps**

* Collect additional data points for intermediate thread counts (e.g., 3, 6, 10).
* Average results in multiple trials to reduce noise.
* Extend performance testing to OpenMP and MPI implementations.

**Conclusion**

The pthreads implementation successfully demonstrates parallel processing of large text data. The preliminary performance analysis shows promising results with clear trends that can guide further optimization and comparisons with other parallel paradigms.