Project 2
SPM course a.a. 24/25

May 15 2025

Distributed out-of-core MergeSort

Design and implement a scalable MergeSort for an N-record file where each record can have a different size in the range [8, PAYLOAD_MAX], where PAYLOAD_MAX is a constant value. A possible C-like representation of one record is as follows:

```
struct Record {
    unsigned long key;  // sorting value 8-bytes
    uint32_t len;       // payload byte length (8 ≤ len ≤ PAYLOAD_MAX)
    char payload[];
};
```

The payload is a blob of bytes arbitrarily initialized. The key field of the record is the only comparison field. You should not assume that the file containing the records to sort can be entirely loaded into the memory of one node. Your solution must work even if the file size exceeds the RAM of a node. Consider the maximum available RAM of a node to be 32 GB.

Tasks

1. Single-node versions (shared-memory)
   Implement two versions, one with FastFlow parallel building blocks (i.e., farm / pipeline/ all-to-all) and one with OpenMP pragmas. Both must produce identical output. Sequential sorting can be implemented leveraging std::sort or equivalent C++ standard library calls.
2. Multi-node hybrid version
   Combine MPI + FastFlow or MPI + OpenMP (your choice, but document it) to produce a distributed version of the MergeSort algorithm. Re-use the single-node code inside each MPI rank.
3. Performance evaluation
   Analyze the performance of your parallel and distributed solutions by varying the number of records N and the payload size distribution (e.g., by considering large N and small PAYLOAD_MAX and vice versa), and the number of FastFlow/OpenMP threads. Report speedup and efficiency varying the number of threads on a single node, and strong and weak scalability curves on the spmcluster up to 8 nodes by changing the number of MPI processes and the number of threads per process.
4. Analysis
   Provide an approximate cost model of your distributed solution. Summarize bottleneck phases, computation-to-communication overlap effectiveness, challenges encountered, and optimizations you adopted.

All parallel versions developed should aim to minimize the parallelization overhead.

Deliverables

Provide all source files, scripts to compile and execute your code on the cluster nodes, and a PDF report (max 15 pages) describing your implementations and the performance analysis conducted. Mention the challenges encountered and the solutions adopted. Submit by email to massimo.torquati@unipi.it your source code and PDF report in a single zip file named 'SPM_project2_<YourName>.zip'**.** Please use the email subject "SPM Project".