Accelerate Stage-out in Single Shared Files from Node-local Burst-buffers

Kohei Sugihara[1], Osamu Tatebe[2]

sugihara@hpcs.cs.tsukuba.ac.jp

[1] Department of Computer Science, University of Tsukuba, Japan

[2] Center for Computational Sciences, University of Tsukuba, Japan

Context: Node-local Burst Buffer and SSF

- **Our approach:** Convert SSF to FPP by Sparse Segments [Sugihara HPS 2020]

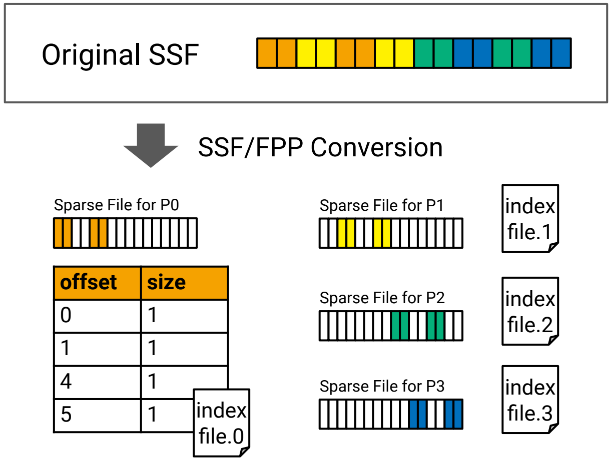


Fig. 1: Sparse Segments
(pairs of a sparse file and an index table)

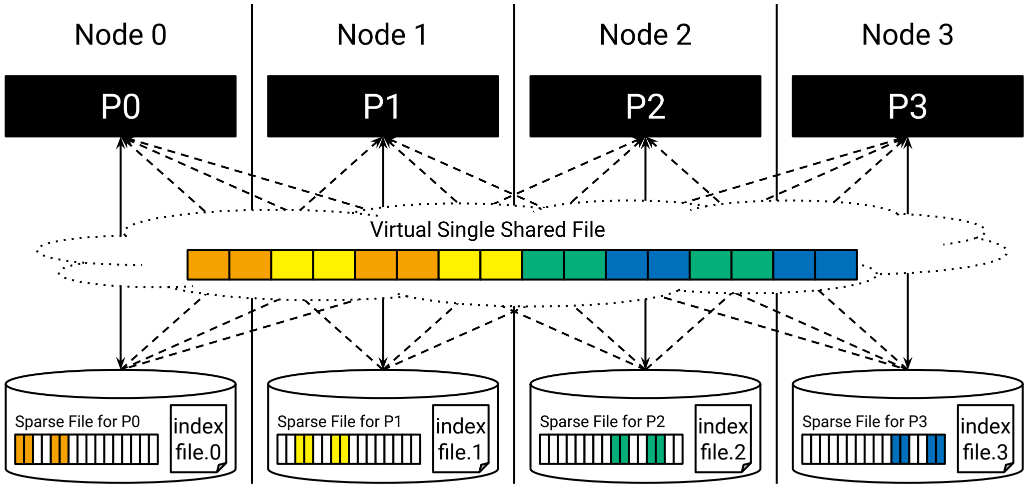


Fig. 2: Write and Read operations for Node-local Burst Buffer
with Sparse Segments

Flushing Sparse Segments

- Flushing sparse segments will merge all sparse files
- Naive merge will induce small writes into the destination SSF

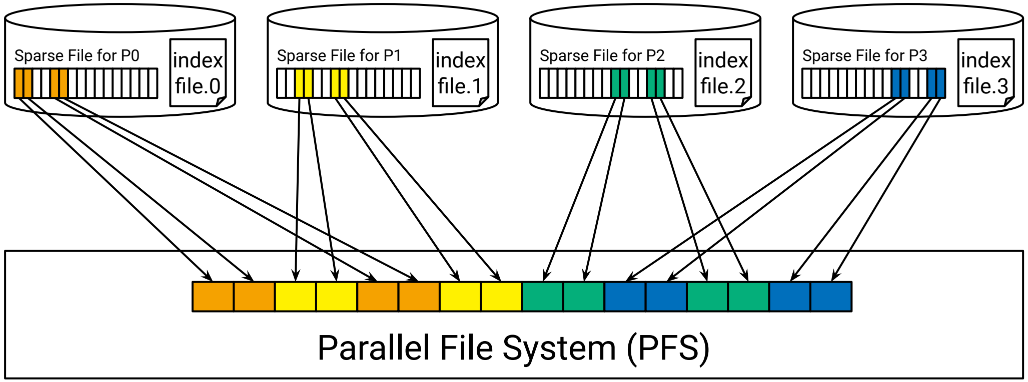


Fig. 3: Small writes against an SSF on flushing Sparse Segments

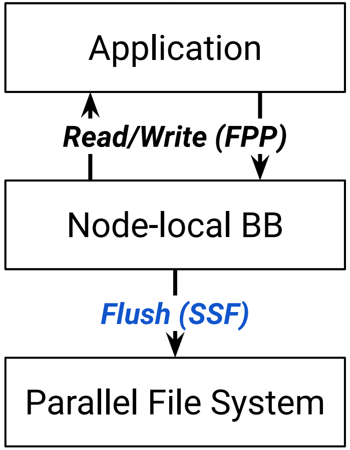


Fig. 4: Lifecycle of Sparse Segments

Optimizations for Flushing Sparse Segments

- **Reconstruct small I/O chunks before flashing to parallel file system**
  - Large contiguous I/O is faster than small-strided I/O
  - Approach #1: I/O aggregators like Two-phase I/O (for sparse files !!)
- **Reduce resources on I/O aggregation as flushing is a background task**
  - Reduce memory footprint and communication
  - Approach #2: I/O aggregation using node-local storage as well
  - Approach #3: Locality-aware process mapping (next slide)

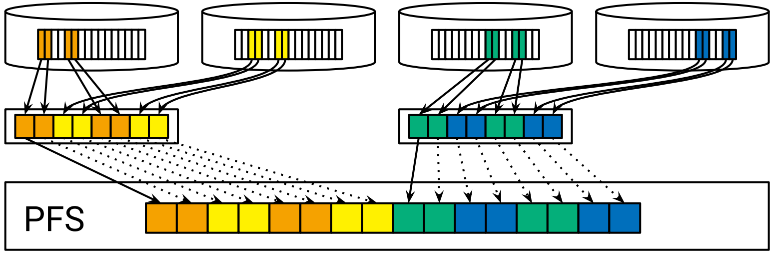


Fig. 5: I/O Aggregation (approach #1)

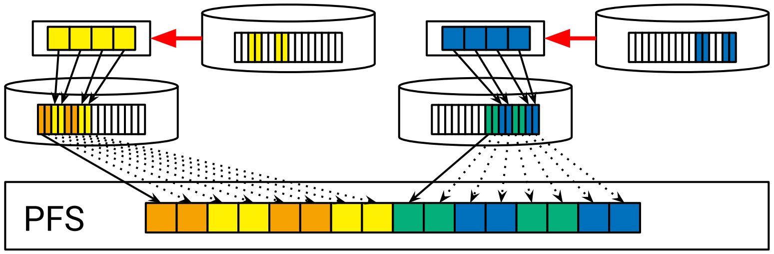


Fig. 6: I/O Aggregation via disk (approach #1 & #2)

Locality-aware Mapping for I/O Aggregators

How do we assign I/O aggregators for an efficient aggregation?

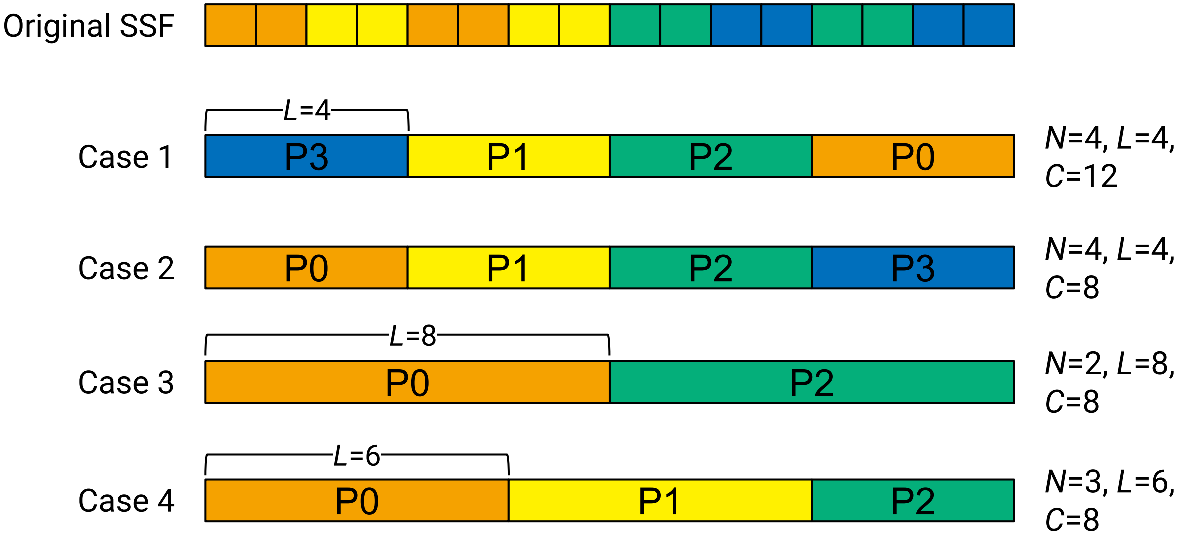


Fig. 7: Process Mapping Candidates on I/O Aggregation

N: # of aggregators
L: Length of the regions
C: # of communications for aggregation

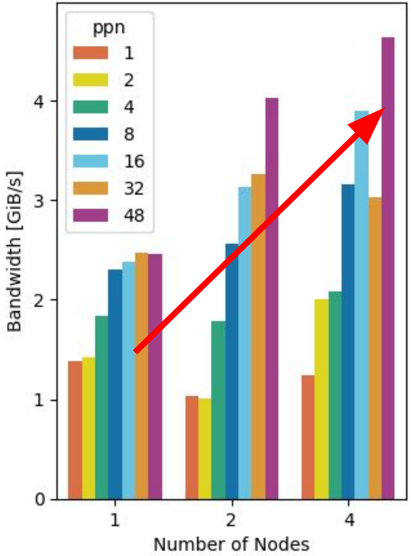
Poor performance (as *C* is greater)

Expected an efficient performance (as *N* & *L* are greater)

Preliminary Experiments

LES-IO: https://github.com/tsukuba-ccs/les-io

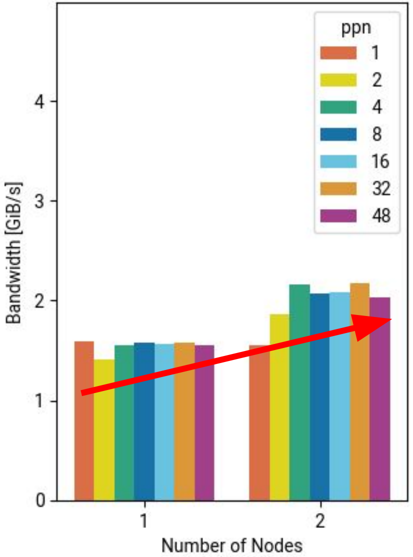
- LES-IO, a Large Eddy Simulation benchmark on Pegasus supercomputer



Better scale for small cases

Q. When the scale ends?

Fig. 8: Results on Flushing Sparse Files into a SSF in the Naive Way



Poor than baseline

Q. Where is the crosspoint?
Q. How does process mapping improve performance?

Fig. 9: Results after Introducing I/O Aggregators (#1 & #2)

Future Work

- Implementation
  - Process mapping method and parameter search (use index tables)
  - Merging in less resource: using node-local storage as well
- Evaluation
  - Find a crosspoint between optimization on process mapping and performance gain
  - At a large scale (> 1k processes)

Acknowledgements

This work was partially supported by:

- JSPS KAKENHI Grant Number JP22H00509
- "Research and Development Project of the Enhanced Infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO)
- "Feasibility studies for the next-generation computing infrastructure", MEXT
- The Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba
- Fujitsu Limited.