Communication among processes is generating considerable interest in the scientific computing community due to the increasing use of distributed memory systems. In the field of high energy physics (HEP), however, little research has been addressed on this topic. More precisely in ROOT I/O [1], the de facto standard for data persistence in HEP applications, no such feature is provided. In order to perform efficient and robust inter-process communications, we introduce TMPIFile into ROOT where Message Passing Interface (MPI) is used to pass data across the entire distributed system.

### Message Passing Interface

- **MPI:**
  - A specification of message passing libraries
  - Communications among processes with separate address spaces
  - Explicit parallelism using distributed memory model

- **Reasons for Using MPI [2]:**
  - Standardization - The only standard that widely supported on all HPC platforms
  - Portability - No code modification required for a different platform that supports the MPI standard
  - Performance Opportunities - Vendor implementations will exploit native hardware features to optimize performance
  - Functionality - Rich set of features
  - Availability - Both vendor and public domain are available:
    - Open-source implementations: MPICH and OpenMPI
    - Vendors: Intel MPI, IBM Blue Gene MPI, Cray MPI, etc.

### Why TMPIFile

- Uniformed I/O solution: Due to the lack of cross-process communication in ROOT, HEP simulations tend to create solutions individually for each use case which results in unnecessary duplicates
- Consistent data handling: Extra procedures beyond ROOT are often required to post-process data which negatively affect the I/O consistency and introduce undesired difficulties for code debugging and maintenance

### Overview

- Class derived from ROOT::TMemFile
- Use MPI to perform cross-process communications
- Write data into disk in TFile as output
- Minimum work unit: one Sender and one Collector
- Each Collector or Sender has its own unique MPI rank (address space)

### Features

- Multi-communicator to improve merge concurrency:
  - Number of Collectors set by users
  - Data collections are local for each communicator group
  - Multiple Collectors (local rank 0) allow concurrent merging

### Performance

- System setup:
  - Theta: 4392 nodes of KNL (64-core); peak of 11.7 PFlops; Cray Aries with Dragonfly.
  - Cori KNL: 9668 nodes of KNL 7250 (68-core); peak of 29.5 PFlops; Cray Aries with Dragonfly.

- Simulation-like test case:
  - ~ 2 MB per event
  - 10 events per sync/message
  - Sleep ~ 5 s between two sends to fake real-case workload

- Peak performance: about 1 message per second per collector which includes receiving the message and merging it with the final output
- Better & more stable performance on Theta due to smaller cluster size and less I/O-intensive workloads
- Scale up to ~ 130 Senders per Collector

### Reference