PortHadoop: Support Direct HPC Data Processing in Hadoop

XI YANG, NING LIU, BO FENG, XIAN-HE SUN AND SHUJIA ZHOU

Illinois Institute of Technology

INTRODUCTION

Big computing needs the power of both computational science and data analytic ecosystems.

Typical tasks of an HPC storage system:

1) Input/Output from computing system (major concern)
2) Query and data processing requests from external users

The query and data processing tasks often take a large portion of the performance time and impair the I/O performance of running applications.

To offload the onerous data intensive tasks to a cheaper and yet powerful system like Hadoop MapReduce system.

Many data processing and data analytic tools/software are developed under Hadoop MapReduce environment.

PRACTICAL NEEDS

Data diagnosis and visualization

Climate and weather simulations can generate a few Terabyte simulation data set.

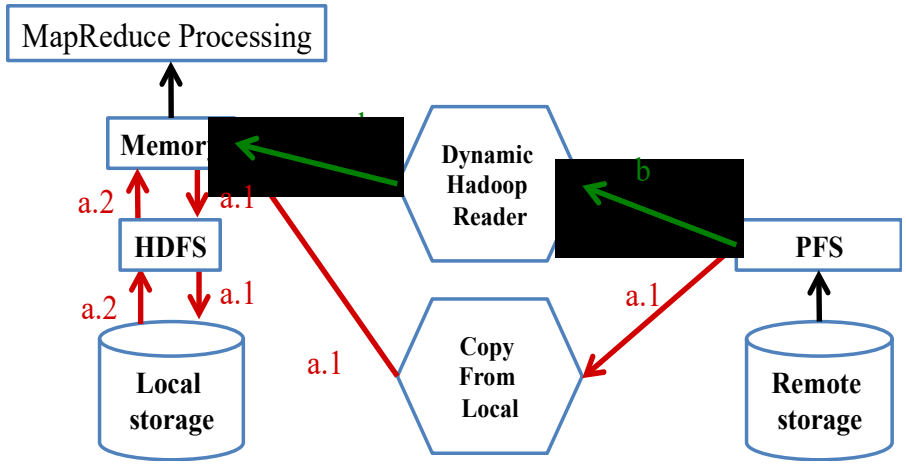
Not all the data is needed to be further analyzed under Hadoop platform.

- Only the data items with interesting events, such as hurricane center and thunderstorms, which could be one order of magnitude smaller than the original data size, are further analyzed.

DATA PATH

Data paths for supporting data processing in Hadoop.

The data path of the naive approach is labeled as a, and colored in red while the data path of PortHadoop is labeled as b, and colored in green.



CHALLENGES

To make data access ***efficient***

- Efficient on each stage of the data path
  - Remote disk I/O
  - Memory of Hadoop client side

Transparently integrate new features into Hadoop

- Enhance Hadoop, but not ask users to do more
  - Bypass HDFS, reading data directly from PFS
  - Parallelism
  - Fault-tolerance support

Insure ***data semantic consistency***

- HDFS data are un-typed, byte-oriented
- Cross boundary reading

HOW TO DO IT: VIRTUAL BLOCK

Introduced the ***virtual block*** concept for PFS data

With techniques in

- Namespace management
- Split alignment
- Task scheduling support

PortHadoop also supports

- Prefetch
- Inherent fault-tolerance

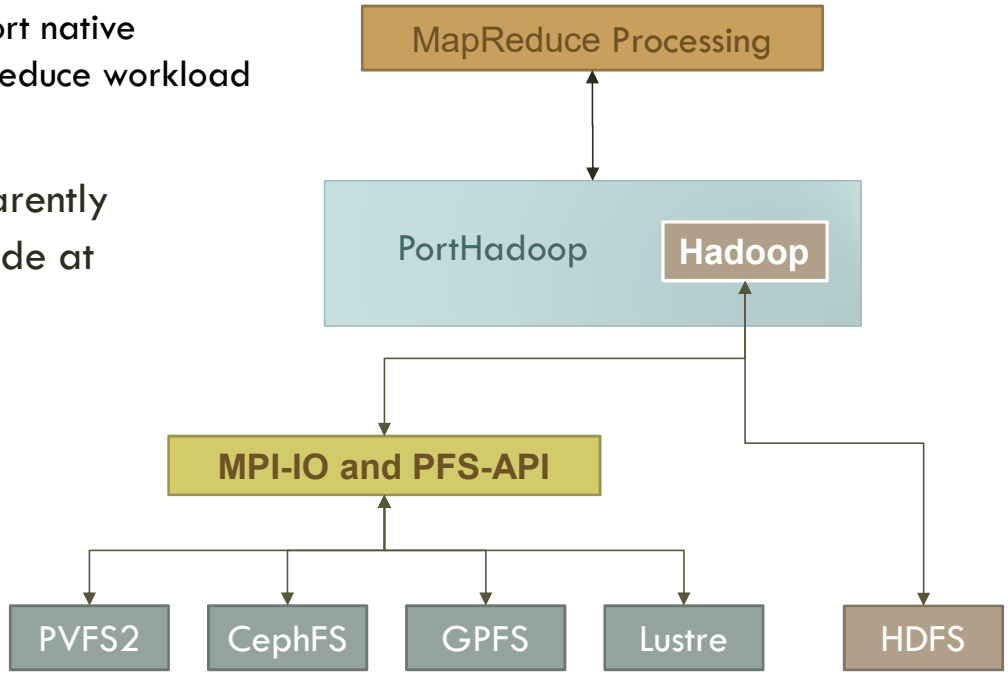
PORTABLE HADOOP

Support native MapReduce workload

More than data transfer, it transparently processes the target data that reside at remote site (not in HDFS).

Communication and interaction

Target dataset has already been kept in in-production PFS



TASK SCHEDULING SUPPORT

PortHadoop intercepts the input split and redirect the data block retrieval from HDFS or local storage system to memory.

Avoid communication to DataNode for input data accessing.

Metadata retrieval before task scheduling.

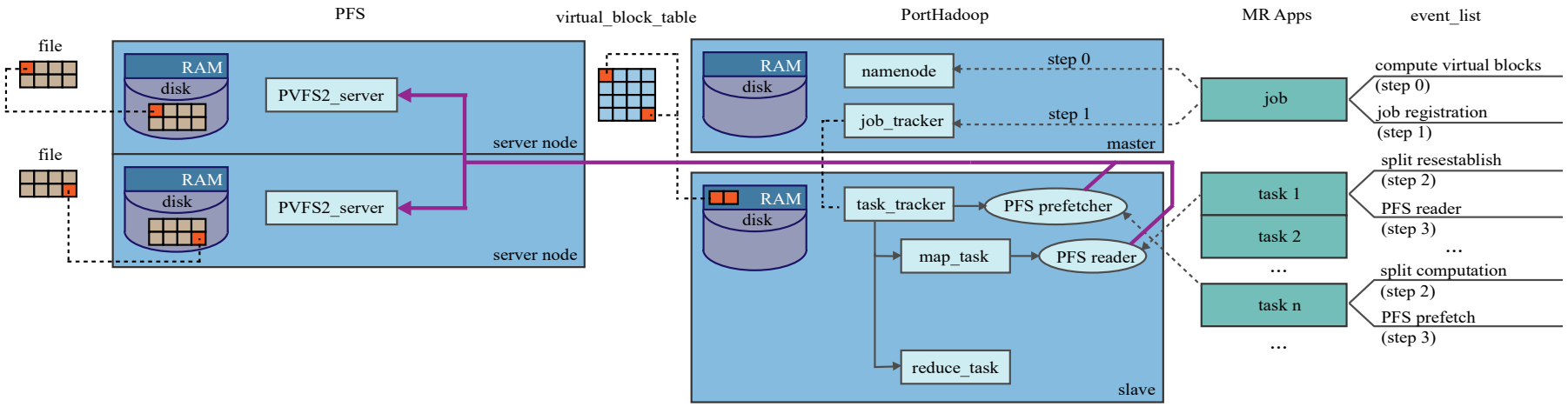
Transparently recognizing remote data

- PFS prefix indicating the data reside at PFS (e.g., pvfs2://)
- addInputPath method in FileInputFormat

The hint of PFS will be reserved in the path, to inform the consequent map task to process a remote data split at a specific PFS.

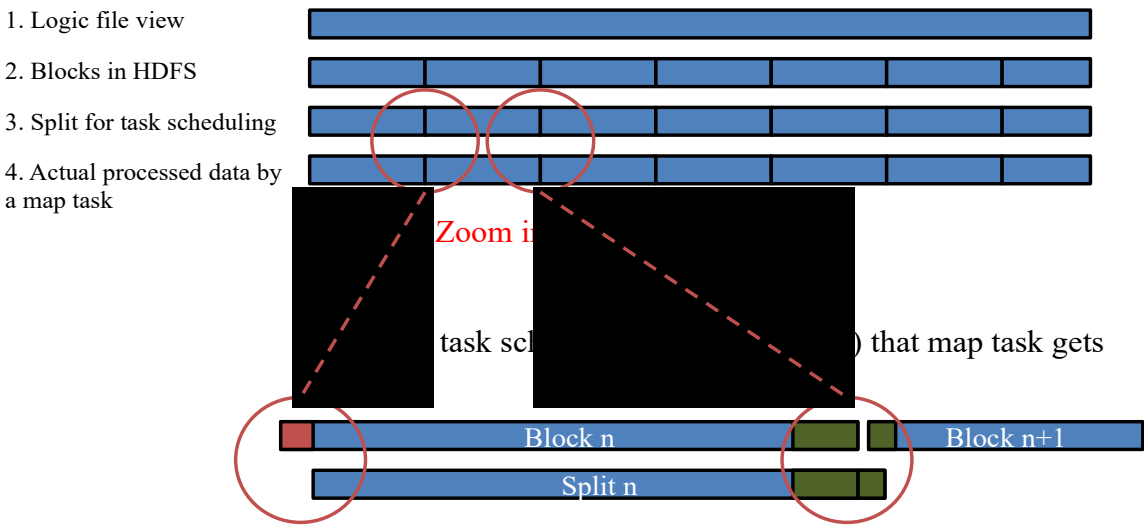
RPC between the client and NameNode to establish virtual blocks according to the input data.

PORTHADOOP SYSTEM ARCHITECTURE



We take PVFS2 as an example for PFS, PortHadoop job_tracker and task_tracker runs on master node, and map task run on PortHadoop slave node. (There are many slaves, take one as example here.) Virtual blocks are managed in the virtual block table in namenode. Task tracker initiates the PFS prefetcher thread, which is responsible for prefetching data directly from remote PFS. Map task initiate the PFS reader thread, which reads data directly from remote PFS.

TRANSPARENT CROSS-BLOCK-BOUNDARY RECORD READ



4. Actual processed data by a map task

HOW TO ALIGN SPLIT?

The splits are categorized into two classes

1. The split with predictable size and pattern (e.g., fixed size record)

   Solution:

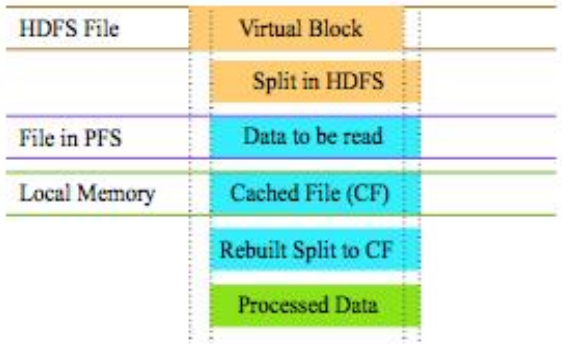
   1) perfectly-aligned read with predictable split size (to assign splits according to the predictable boundary)

2. The split with unpredictable size

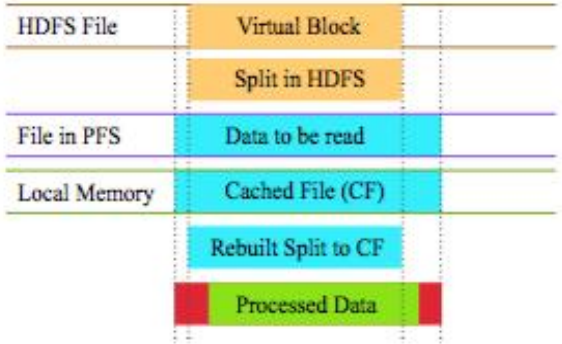
   Solutions:

   2) redundant read
   3) perfectly-aligned read with dynamic split boundary detection

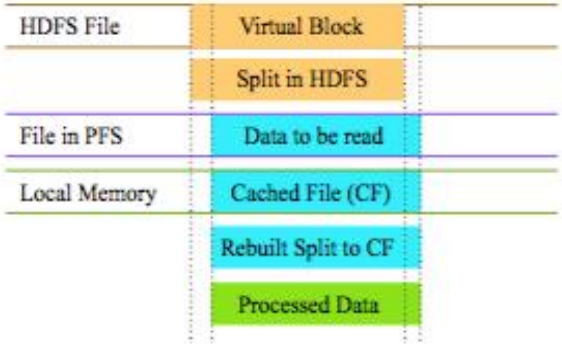
READ STRATEGIES AND ACTUAL DATA PROCESSING IN PORTHADOOP



(a) Perfectly-aligned read strategy with predicatable split sizes



(b) Redundant read strategy



(c) Perfectly-aligned read strategy with dynamic split boundary detection

PREFETCHING

Data transfer-then-processing, a blocking procedure.

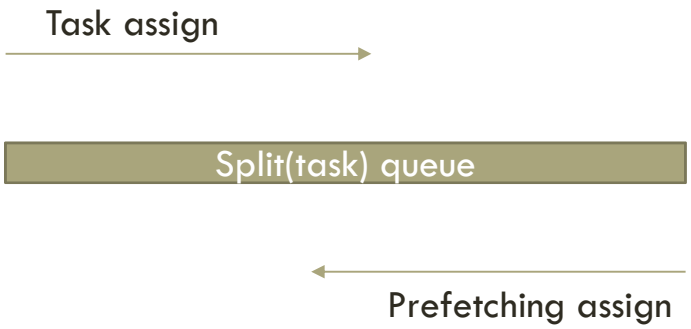
Considering

- Large memory
- Multiple map task waves

Prefetching helps

- To hide data access latency.
- To utilize bandwidth.

When bandwidth is underutilization.



Implementation:

- Poll-based task assignment, pursue data locality
- PRC call, action
- Synchronized and memory consumption control

FAULT-TOLERANCE

As PortHadoop is an extended Hadoop, its map tasks will store their intermediate results onto local file system as the default Hadoop does.

- Reschedule the failure task and re-fetch the corresponding input data.

PortHadoop optimizes data transfer by pipelining the transfer and processing prior to alleviating map task skewness and stragglers.

- We observed the skewnesses in multiple waves of map task processing.

To optimizing task processing, PortHadoop also supports backup tasks, as that in the conventional Hadoop.

EVALUATION

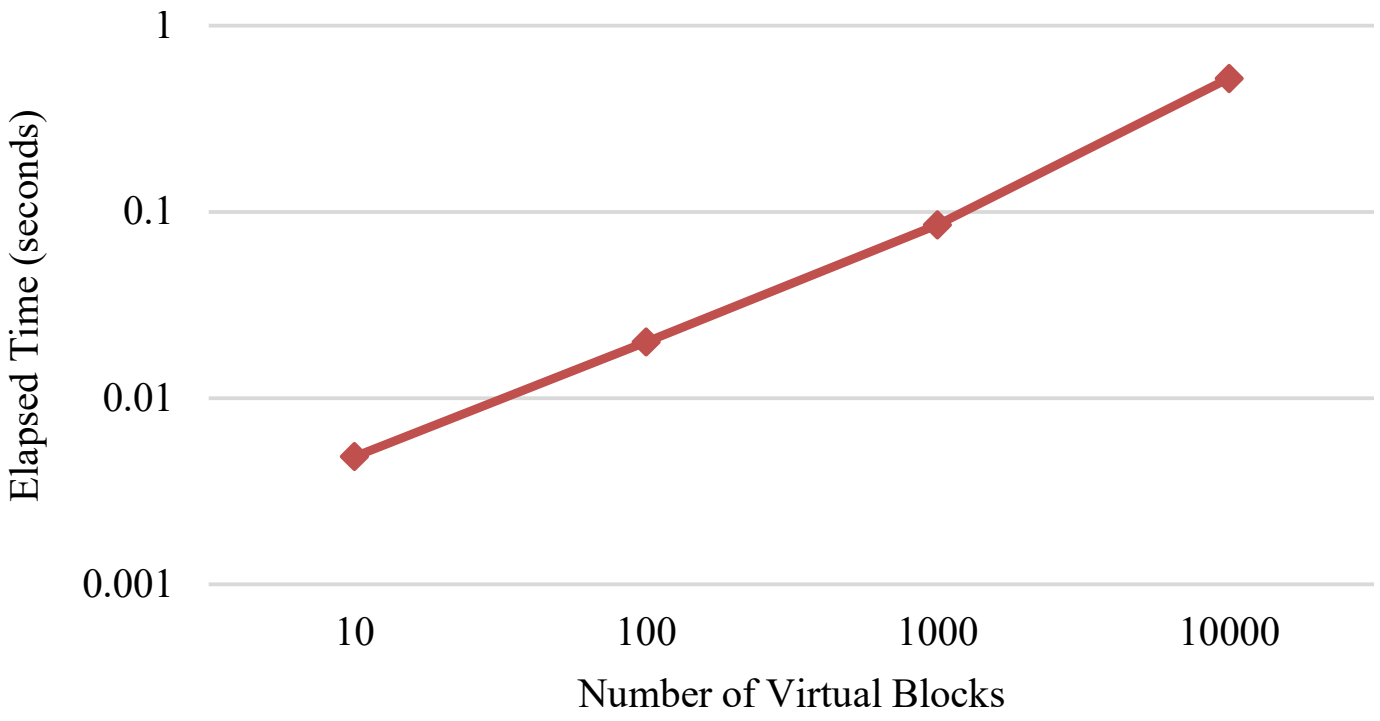
We use 17 nodes in Craysun and 25 nodes in HEC, for CephFS and PVFS2, respectively.

Craysun: All slave nodes are Dell PowerEdge SC 1425 Server, each of which has an Intel Xeon CPU 3.40GHz processor, 1GB of memory, and a 36GB (15,000 rpm) SCSI hard drive.

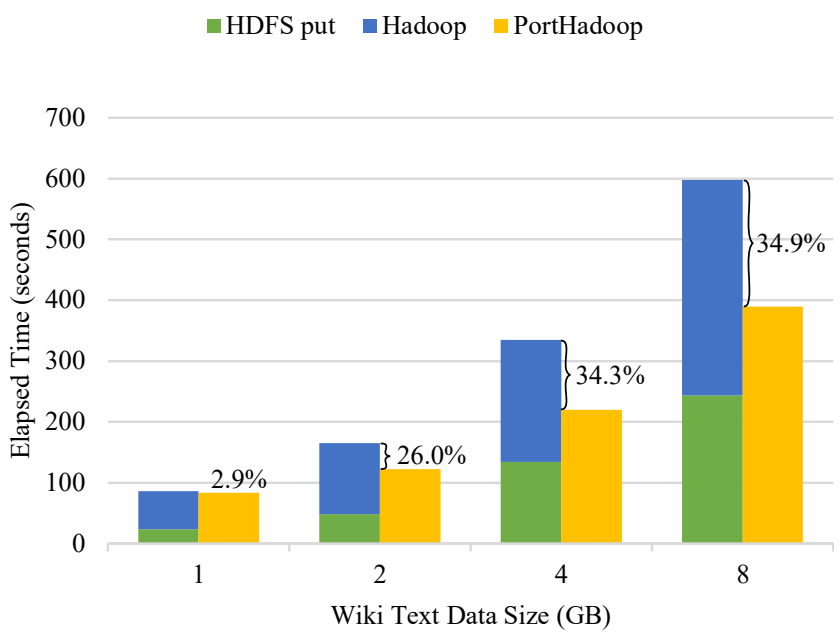
HEC is an SUN Fire Linux cluster, in which each node has two AMD Opteron processors, 8GB memory, and a 250GB SEAGATE HDD. All nodes run Ubuntu 14.04 with the latest Linux kernel.

The MPI installed in both systems is MPICH 3.0.4.

VIRTUAL BLOCK OVERHEAD

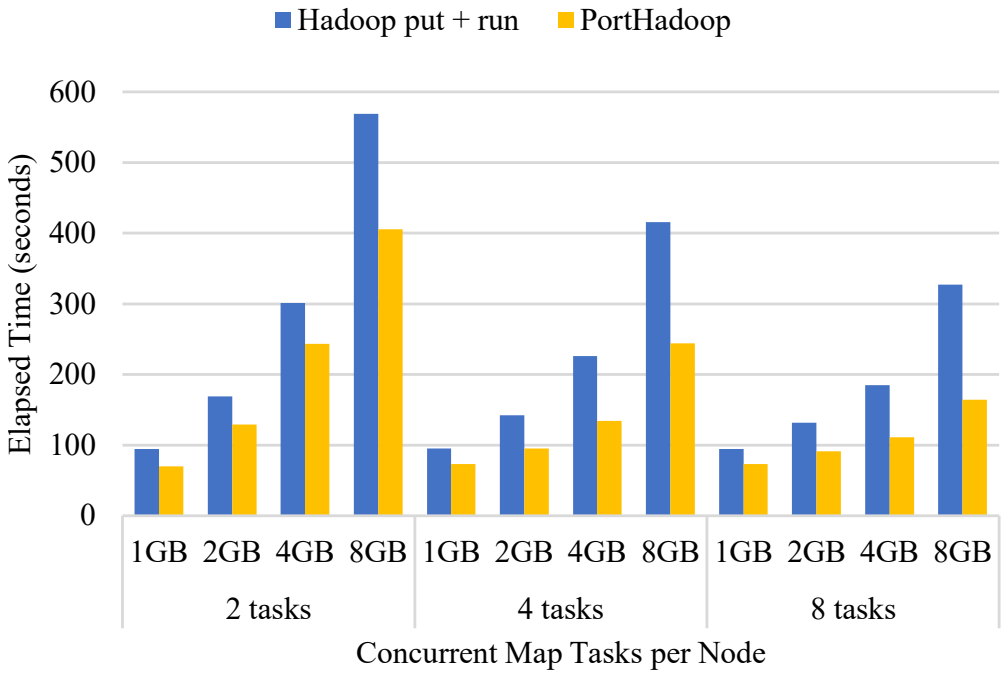


EXPERIMENTAL RESULTS (W/CEPH FS)

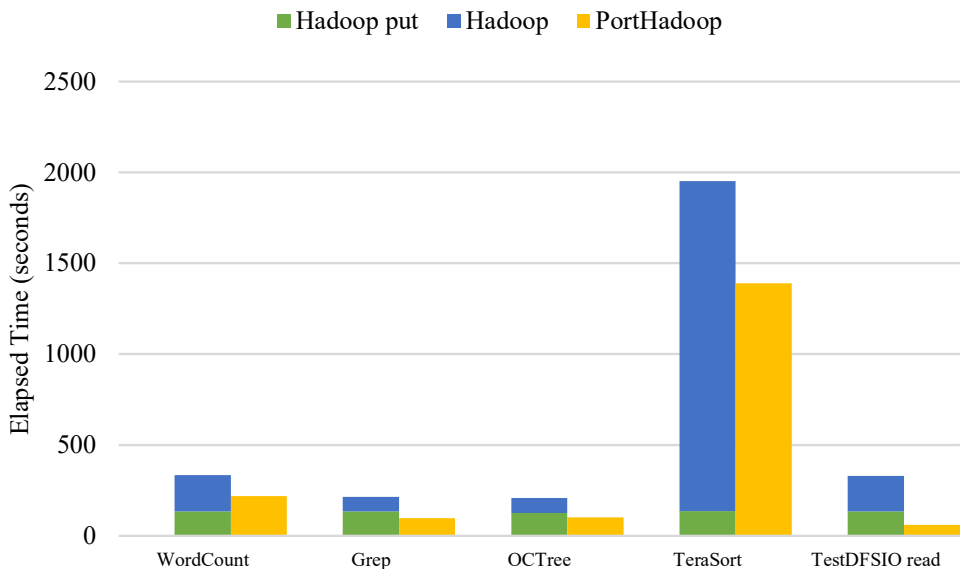


Performance comparison between conventional Hadoop and PortHadoop with WordCount on wiki text dataset. "HDFS put" copies an input data file from CephFS to HDFS and then "Hadoop" processes the dataset. "PortHadoop" directly reads and processes the dataset residing in CephFS.

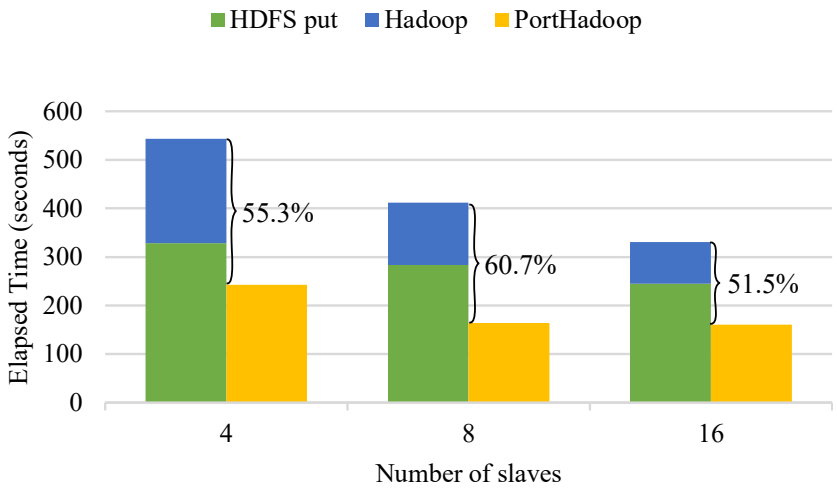
VARY NUMBER OF CONCURRENT MAP TASKS (W/PVFS2)



PERFORMANCE OF MULTIPLE APPLICATIONS (W/CEPHFS)



WHAT IF BANDWIDTH BETWEEN PFS AND HADOOP CLUSTER IS A BOTTLENECK (W/PVFS2)



PortHadoop improves the overall performance but suffers from the limited network bandwidth for further scalability.

However, the copy phase in naive approach is still costly due to the constraint of the network bandwidth.

Performance comparison between Hadoop and PortHadoop for 3 number of slaves. Each WordCount job processes 8 GB wiki text data with 8 concurrent map tasks per slave.

RELATED WORK

1. Integrating MapReduce and HPC data processing power. Such as, MRAP and SciHadoop.
   - Data model mapping. **Copy-based.**
2. Researchers in HPC proposed to extend existing PFS to support its non-native MapReduce workloads. Such as, PVFS-shim-layer, GPFS-FPO, and Lustre.
   - **Plugin needed & Data layout changed.**
3. Improve the response time of applications, adopting pipelining or in-memory processing strategies. Such as MapReduce Online, Themis, M3R.
   - These pipelining strategies are adopted **within MapReduce cluster.**
   - Themis and M3R **trade reliability for performance.**

SUMMARY

Our experimental results show that PortHadoop is effective and compatible with existing PFS such as PVFS2 and CephFS.

In particular, under PortHadoop:

1. MapReduce can read data directly from PFS without data copying. The target blocks processed by map tasks reside in memory rather than on disk.
2. Only the needed data at PFS are taken to Hadoop at runtime.
3. Blocks in a PFS files can be accessed concurrently.
4. According to the amount of data required by map tasks, the data transfer operations from PFS to HDFS can overlap with MapReduce data processing.
5. PortHadoop supports fault tolerance as default Hadoop does.

Q & A

THANK YOU !

xyang34@hawk.iit.edu