PortHadoop-R: The Merging of HPC and Big Data

Kun Feng
Illinois Institute of Technology
PortHadoop-R: The Merging of HPC and Big Data

Outline

• Introduction
• Background
• Design
• Evaluation
• Conclusion
Introduction

• The merging of HPC and big data analytics is inevitable.
  • High Performance Computing (HPC) is becoming data intensive.
  • Big data applications are requiring more and more computing power.
  • Two ecosystems are designed for different applications and with different design principles.

• Two ecosystems will co-exist.
  • Neither can have all the merits of the other.
  • The best we can have is a hybrid system which can provide the functionalities and merits of both ecosystems.
Background -- Motivation

• Native data access exists
  • HPC applications read from/write to PFS (POSIX I/O and MPI-IO)
  • Big Data applications read from/write to HDFS (HDFS API)

• Data dependency in-between
  • Big data applications analyze data generated by simulations
  • Analysis results guide simulation runs
  • Use explicit **serial** copy
  • Redundant data store in HDFS
Background -- Use case

- **Iterative process**
  - Simulation generates large data
  - Analyze and visualize iteratively
  - Analysis results help steering simulations

- **Simulation**
  - MPI-based
  - HPC platform
  - Use PFS as data storage

- **Visualization**
  - Coarse visual analysis
  - Quick identification
  - Animated images

- **Data analysis**
  - Statistical analysis
  - SQL query
  - Use HDFS as data storage
Our solution

• We propose PortHadoop-R to support the merging at the data access level
  • Allow reading data directly from PFS to the memory of Hadoop nodes
  • Integrate the data transfer with R data analysis and visualization
  • Optimized to utilize the merits of PFS and MapReduce to achieve concurrent data transfer and latency hiding
  • Tested with real data of NASA climate modeling applications
Design -- Architecture

Conventional solution
- Serial data copy
- Fat node design
- Serial visualization and analysis

PortHadoop-R
- Copy free design
- Parallel data access
- Parallel visualization and analysis
Design -- PortHadoop (cont.)

• Virtual block
  • Create HDFS data block virtually
  • Map HDFS block to file segment in PFS (filename, offset, size)
  • Stored in NameNode

• On-demand data access
  • MPI-based PFS file reader
  • Triggered only when the block is accessed
  • Minimal data transfer

• Portability
  • MPI-IO guarantees the PFS compatibility
Design -- Parallel Image Plotting

• Multiple concurrent image plotting tasks
  • Key is defined based on dimension of the dataset (e.g. along time or altitude)
  • Each map task is assigned to plot one image
  • Parallelism of the Hadoop cluster is utilized
  • The target file segment info can be obtained from virtual block
  • Multiple read requests are issued to read data from PFS concurrently
  • PFS bandwidth is fully utilized as well
Design -- Parallel Data Analysis

• Data analysis can be carried out alongside image plotting
  • Statistical information can be extracted (e.g. min, max and avg)
  • SQL queries can be executed upon the data (e.g. select top 10 records)
  • Results are packed with plotted images
Design -- Animation

• Animated image sequence is a very good way to analyze the HPC data
  • Images and analysis results are shuffled according to the key
  • Reduce tasks combine images together to make animation
  • Based on the `convert` command in ImageMagick
  • Not parallelizable
  • Animations are saved in HDFS
Experiment Setup

• Hardware
  • Chameleon at TACC
  • Baremetal
  • 4, 8 and 16 Hadoop nodes, 8 OrangeFS nodes

• Software
  • CentOS 7.3
  • Java 1.7.0_79
  • PortHadoop-R (modified from Hadoop-2.5.0-CDH5.3.3)
Experiment Setup (cont.)

- **Configuration**
  - Input data: 48, 96, 192 and 384 single-level data files from a real NU-WRF 1250x1250x50 run (up to 23.14GB). Each file is ~61.7MB.
  - All data are read from 8-node OrangeFS. Each Map task visualizes one file and creates one image. A SQL query is also applied in Map task for Anlys workload. A Reduce task merges all images into an animation.
  - The elapsed time, measured on the master node.

- **Workload**

<table>
<thead>
<tr>
<th>Name</th>
<th>Image Plotting</th>
<th>Animation</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vis-only</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Img-only</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Anlys</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
PortHadoop-R: The Merging of HPC and Big Data

PortHadoop-R VS Conventional Methods: Performance (Vis-only)

Serial Copy: one copy command issued on one Hadoop node
Parallel Copy: 4 copy commands issued on each Hadoop node (32 in total)
Serial Copy: one copy command issued on one Hadoop node
Parallel Copy: 4 copy commands issued on each Hadoop node (32 in total)
Serial Vis: read all data into one fat node, visualize
Parallel Vis: visualize all files on all Hadoop nodes
PortHadoop-R Vis: visualize all files into images
Serial Animation: merge all images into one animation

Good speedup: Total time is reduced by up to 15x and 1.47x compared to serial and parallel solution.

Serial solution
Manually parallelized solution

\[
\text{Speedup}_{\text{total}} = 15
\]

\[
\text{Speedup}_{\text{PortHadoop-R}} = 1.47
\]

Good speedup: Total time is reduced by up to 15x and 1.47x compared to serial and parallel solution.
PortHadoop-R VS Conventional Methods: Speedup (Img-only)

Good speedup: Image plotting time is reduced by up to 24x and 1.77x compared to serial and parallel solution.

- **Serial Copy**: one copy command issued on one Hadoop node
- **Parallel Copy**: 4 copy commands issued on each Hadoop node (32 in total)
- **Serial Vis**: read all data into one fat node, visualize
- **Parallel Vis**: visualize all files on all Hadoop nodes

PortHadoop-R: The Merging of HPC and Big Data
PortHadoop-R Scale-out Performance (Vis-only)

Good scale-out performance: The image plotting time reduces proportional to number of nodes.

Image Plotting: visualize all files into images
Serial Animation: merge all images into one animation
PortHadoop-R: The Merging of HPC and Big Data

PortHadoop-R Scale-up Performance (Vis-only)

Image Plotting: visualize all files into images
Serial Animation: merge all images into one animation

Good scale-up performance: The image plotting time reduces proportional to number of cores per node.
Conclusion

• PortHadoop-R supports the merging of HPC and big data in data access level

• Leverage merits of both systems

• Great performance (speedup, scalability)
  • Speedup in visualization
    • 15x and 24x: compared with serial solution for Vis-only and Img-only
    • 1.47x and 1.77x: compared with manually parallelized solution for Vis-only and Img-only
  • Good scalability (scale-out and scale-up)

• Applicable to other frameworks
Chameleon toolkit

• A small tool created to facilitate our experiments on Chameleon
  • Forked from the official cc-snapshot repo
  • Helper scripts to wire up and authenticate instances
  • Helper commands to keep files on multiple nodes synchronized
  • [https://github.com/fkengun/Chameleon-toolkit](https://github.com/fkengun/Chameleon-toolkit)
Thank You

Q&A
Design -- PortHadoop (backup)
Related work (backup)

• In-situ analysis in HPC
  • E.g. VisIt, ParaView
  • Compete with simulations for resources

• Big data frameworks
  • E.g. Hadoop
  • Require explicit serial data copy to load data

• Improved big data frameworks
  • E.g. SciHadoop
  • Only support processing data on HDFS
Design -- R Interface (backup)

- Leverage existing R packages
  - Easy to use
  - Rich capabilities in analysis and graphics
  - Rely on RHadoop for distributed computation

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>3.2.2</td>
<td></td>
</tr>
<tr>
<td>rhdfs</td>
<td>1.0.8</td>
<td>HDFS connector</td>
</tr>
<tr>
<td>rmr2</td>
<td>3.3.1</td>
<td>R interface for Hadoop MapReduce</td>
</tr>
<tr>
<td>Cairo</td>
<td>1.5-9</td>
<td>Graphics library to plot image</td>
</tr>
<tr>
<td>plot3D</td>
<td>1.1</td>
<td>Provides 3D plotting functions</td>
</tr>
<tr>
<td>sqldf</td>
<td>0.4.10</td>
<td>Run SQL upon dataframe in R</td>
</tr>
</tbody>
</table>
PortHadoop-R VS Conventional Methods: Speedup (Vis-only) (backup)

- **Parallel Copy+Parallel Vis vs Serial Copy+Serial Vis**
- **PortHadoop-R vs Parallel Copy+Parallel Vis**
- **PortHadoop-R vs Serial Copy+Serial Vis**

**Good speedup:** Total time is reduced by up to **15x** and **1.47x** compared to serial and parallel solution.

Serial Copy: one copy command issued on one Hadoop node
Parallel Copy: 4 copy commands issued on each Hadoop node (32 in total)
Serial Vis: read all data into one fat node, visualize and animate
Parallel Vis: visualize all files on all Hadoop nodes and generate animation

PortHadoop-R: visualize all files into images and animate
**PortHadoop-R VS Conventional Methods: Performance (backup)**

Serial Copy: one copy command issued on one Hadoop node
Parallel Copy: 4 copy commands issued on each Hadoop node (32 in total)
Serial Vis: read all data into one fat node, visualize and animate
Parallel Vis: visualize all files on all Hadoop nodes and animate on the master node
PortHadoop-R Vis: visualize all files into images
Serial Animation: merge all images into one animation

\[
\text{Speedup}_{\text{total}} = \text{Speedup}_{\text{parallel}} \times \text{Speedup}_{\text{PortHadoop}}
\]

\[
\text{Speedup}_{\text{parallel}} = \frac{1}{(1 - P_{\text{vis}} - P_{\text{copy}}) + \frac{P_{\text{vis}}}{\text{Speedup}_{\text{vis}}} + \frac{P_{\text{copy}}}{\text{Speedup}_{\text{copy}}}}
\]

\[
\text{Speedup}_{\text{vis}} = 12.1
\]

\[
\text{Speedup}_{\text{copy}} = 16
\]

\[
\text{Speedup}_{\text{PortHadoop}} = 1.47
\]

\[
\text{Speedup}_{\text{total}} = 15
\]
PortHadoop-R: The Merging of HPC and Big Data

PortHadoop-R Data Analysis Performance (Anlys) (backup)

SQL query time is proportional to result size.

Simple analysis takes same time as image plotting only (no analysis).

no analysis: image plotting and animation only
average: plot image and calculate average of the data set
stdev: plot image and calculate the standard deviation of the data set
highlight: plot image and highlight the top 1% area
top 0.5%: plot image and subset the top 0.5% data of the data set
top 1%: plot image and subset the top 1% data of the data set