Decoupled I/O for Data-Intensive High Performance Computing

Chao Chen [1] **Yong Chen** [1] Kun Feng [2] Yanlong Yin [2]
Hassan Eslami [3] Rajeev Thakur [4] Xian-He Sun [2]
William D. Gropp [3]

[1]Department of Computer Science, Texas Tech University

[2]Department of Computer Science, Illinoise Institude of Technology

[3]Department of Computer Science, University of Illinois Urbana-Champaign

[4]Mathematics and Computer Science Division, Argonne National Laboratory

Sep 12th, 2014

Scientific Computing and Workload

- High performance computing is a strategic tool for scientific discovery and innovation
  - Climate Change: Community Earth System Model (CESM)
  - Astronomy: Supernova, Sloan Digital Sky Survey
  - etc..
- Utilizing HPC system to simulate events and analyze the output to get insights

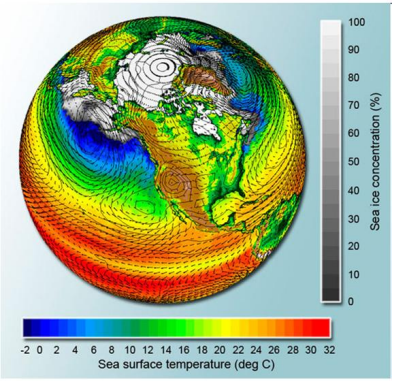

Figure 1: Climate modeling and analysis

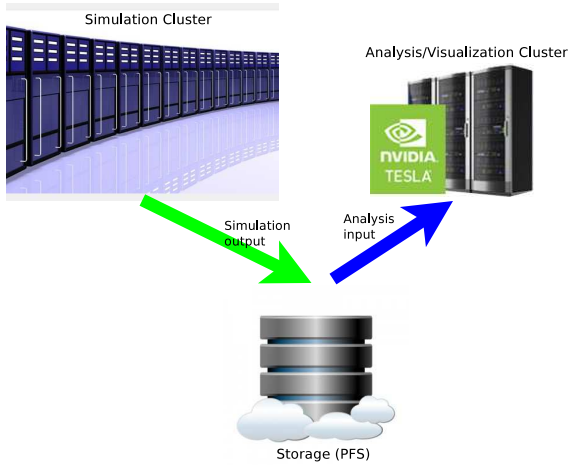

Figure 2: Typical scientific workload

Big Data Problem

- Many scientific simulations become highly data intensive
- Simulation resolution desires finer granularity both spacial and temporal
  - e.x. climate model, 250KM ⇒ 20KM; 6 hours ⇒ 30 minutes
- The output data volume reaches tens of terabytes in a single simulation, the entire system deals with petabytes of data
- The pressure on the I/O system capability substantially increases

PI	Project	On-Line Data	Off-Line Data
Lamb, Don	FLASH: Buoyancy-Driven Turbulent Nuclear Burning	75TB	300TB
Fischer, Paul	Reactor Core Hydrodynamics	2TB	5TB
Dean, David	Computational Nuclear Structure	4TB	40TB
Baker, David	Computational Protein Structure	1TB	2TB
Worley, Patrick H.	Performance Evaluation and Analysis	1TB	1TB
Wolverton, Christopher	Kinetics and Thermodynamics of Metal and Complex Hydride Nanoparticles	5TB	100TB
Washington, Warren	Climate Science	10TB	345TB
Tsigelny, Igor	Parkinson's Disease	2.5TB	50TB
Tang, William	Plasma Microturbulence	2TB	10TB
Sugar, Robert	Lattice QCD	1TB	44TB
Siegel, Andrew	Thermal Striping in Sodium Cooled Reactors	4TB	8TB
Roux, Benoit	Gating Mechanisms of Membrane Proteins	10TB	10TB

Figure 3: Data volume of current simulations

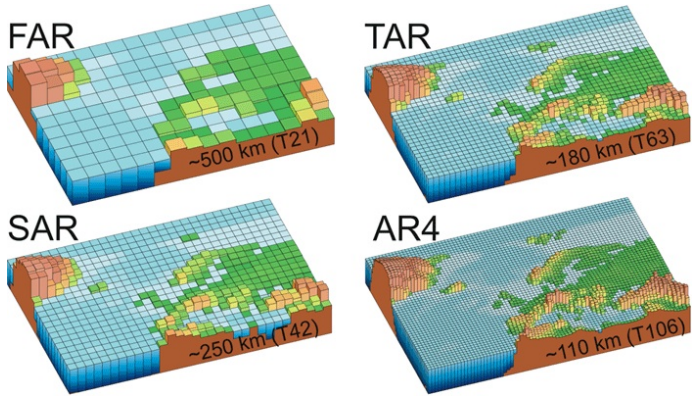

Figure 4: Climate Model Evolution: FAR (1990), SAR (1996), TAR (2001), AR4 (2007)

Gap between Applications' Demand and I/O System Capability

- Gyrokinetic Toroidal Code (GTC) code
  - Outputs particle data that consists of two 2D arrays for electrons and ions, respectively
  - Two arrays distributed among all cores, particles can move across cores in a random manner as the simulation evolves
- A production run with the scale of 16,384 cores
  - Each core outputs roughly two million particles, 260GB in total
  - Desires $O(100MB/s)$ for efficient output
- The average I/O throughput of Jaguar (now Titan) is around 4.7MB/s per node
- Large and growing gap between the application's requirement and system capability

Decoupled I/O

A new way of moving computations near to data to minimize the data movement and address the I/O bottleneck issue

- A runtime system design for our Decoupled Execution Paradigm
- Providing a set of interface for users to decouple their applications, and map into different sets of nodes

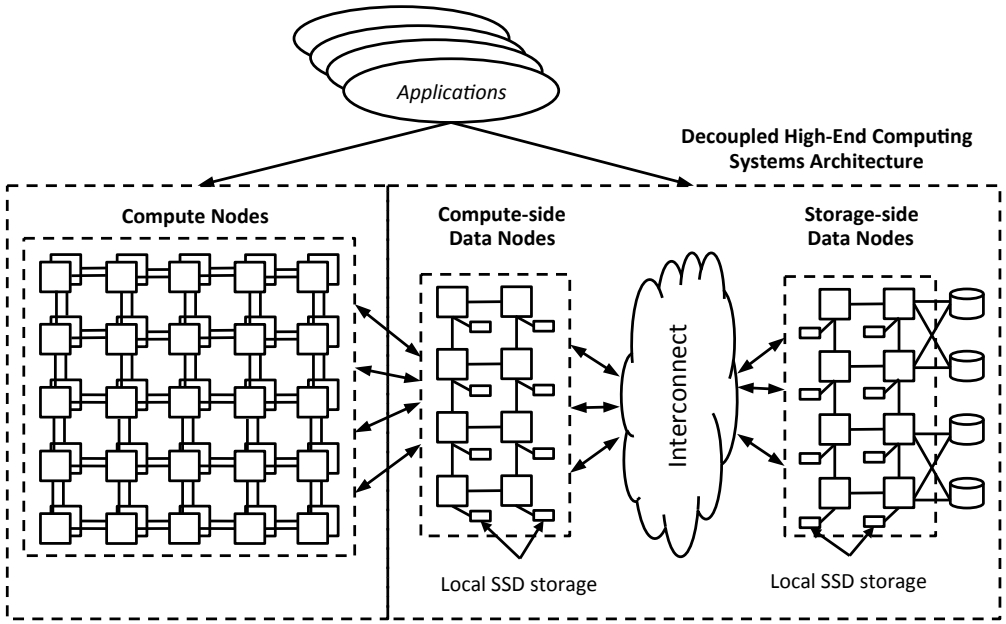

Figure 5: Decoupled Execution Paradigm and System Architecture

Overview of Decoupled I/O

- An extension to MPI library, managing both Compute nodes and Data nodes in the DEP architecture.
- Internally splits them into compute group and data group for normal applications and data-intensive operations respectively.

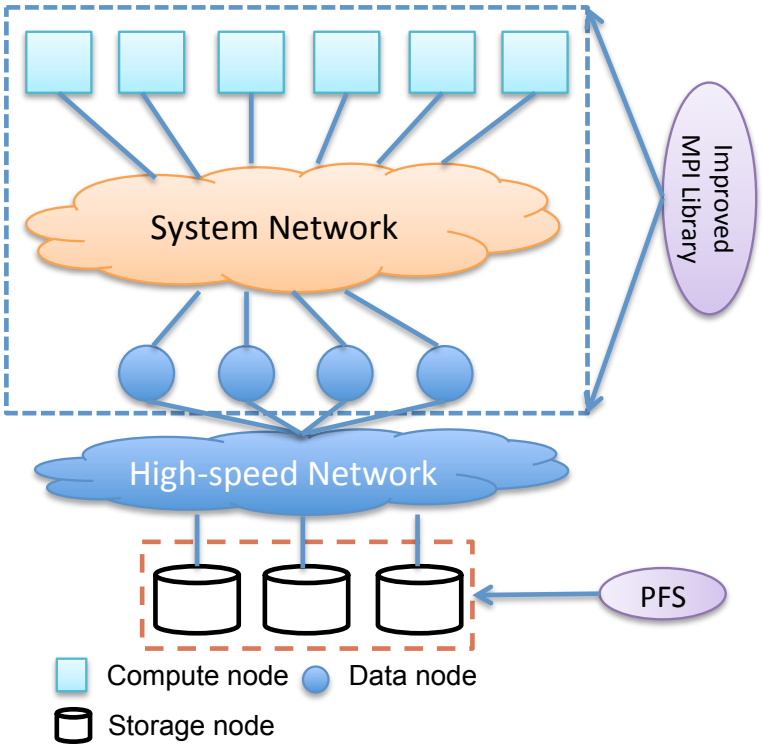

Figure 6: Overview of Decoupled I/O

Overview of Decoupled I/O

Involves 3 improvements to existing MPI library:

- Decoupled I/O APIs
- Improved MPI compiler (mpicc)
- Improved MPI process manager (hydra)

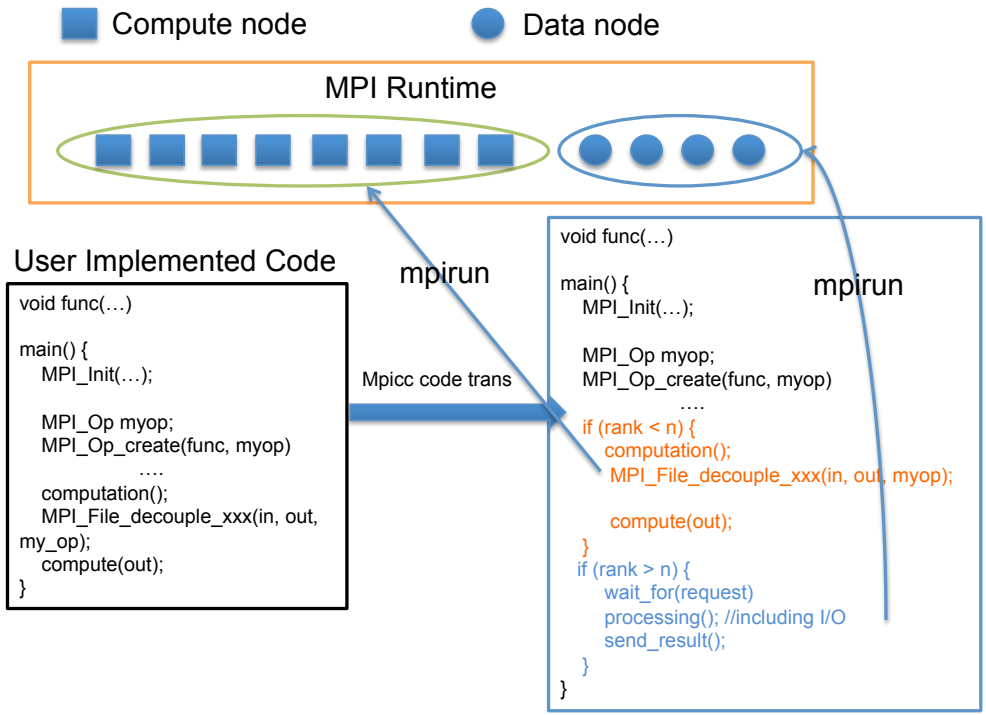

Figure 7: Decoupled I/O at runtime

Decoupled I/O API

- Abstracting each data-intensive operation with two phases: traditional I/O and data processing
- Providing APIs to treat them as an ensemble with different file handler design, and *data_op* argument

Table 1: Decoupled I/O APIs

MPI_File_decouple_open(MPI_Decoupled_File fh, char * filename, MPI_Comm comm);
MPI_File_decouple_close(MPI_Decoupled_File fh, MPI_Comm comm);
MPI_File_decouple_read (MPI_Decoupled_File fh, void *buf, int count, MPI_Datatype data_type, MPI_Op data_op, MPI_Comm comm);
MPI_File_decouple_write(MPI_Decoupled_File fh, void *buf, int count, MPI_Datatype data_type, MPI_Op data_op, MPI_Comm comm);
MPI_File_decouple_set_view(MPI_Decoupled_File fh, MPI_Offset disp, MPI_Datatype etype, MPI_Datatype filetype, char * datarep, MPI_Info info, MPI_Comm comm);
MPI_File_decouple_seek(MPI_Decoupled_File fh, MPI_Offset offset, int whence, MPI_Comm comm);

Decoupled I/O API Example

Traditional Code

```
int buf;
MPI_File_read(fh, buf, ...);
for(i = 0; i < bufsize; i++) {
  sum += buf[i];
} ...
```

Decoupled I/O Code

```
/* define operation */
int sum_op(buf, bufsize) {
    for (i = 0; i < bufsize; i++ )
        sum += buf[i];
}
....
MPI_op myop;
MPI_Op_create(myop, sum_op);
MPI_File_decoupled_read(fh, sum, myop, ....);
```

Process/Node management

- Data nodes and compute nodes are at the same level belonging to two groups
- "mpirun -np *n* -dp *m* -f hostfile ./app" to run an application with *n* compute processes and *m* data processes
- All of them belong to the MPI_COMM_WORLD communicator with distinguished rank
- Each group has its own group communicator MPI_COMM_LOCAL as an intra-communicator,
- MPI_COMM_INTER communicator as a group-to-group inter-communicator between the compute processes group and data processes group.

Code Decoupling & Compiler Improvement

- Identify the process type, compute process or data process, with its *rank* in MPI_COMM_WORLD to execute different codes
- Data process code is automatically generated by mpicc with hints defined by macros MPI_DECOUPLE_START and MPI_DECOUPLE_END
- MPI_Op for defining offloaded operations that have to be registered at the before MPI_DECOUPLE_START.

Decoupled I/O Implementation and Prototyping

- Completely based on MPI library
- Gather the tasks from compute processes, and scatter them to data process.

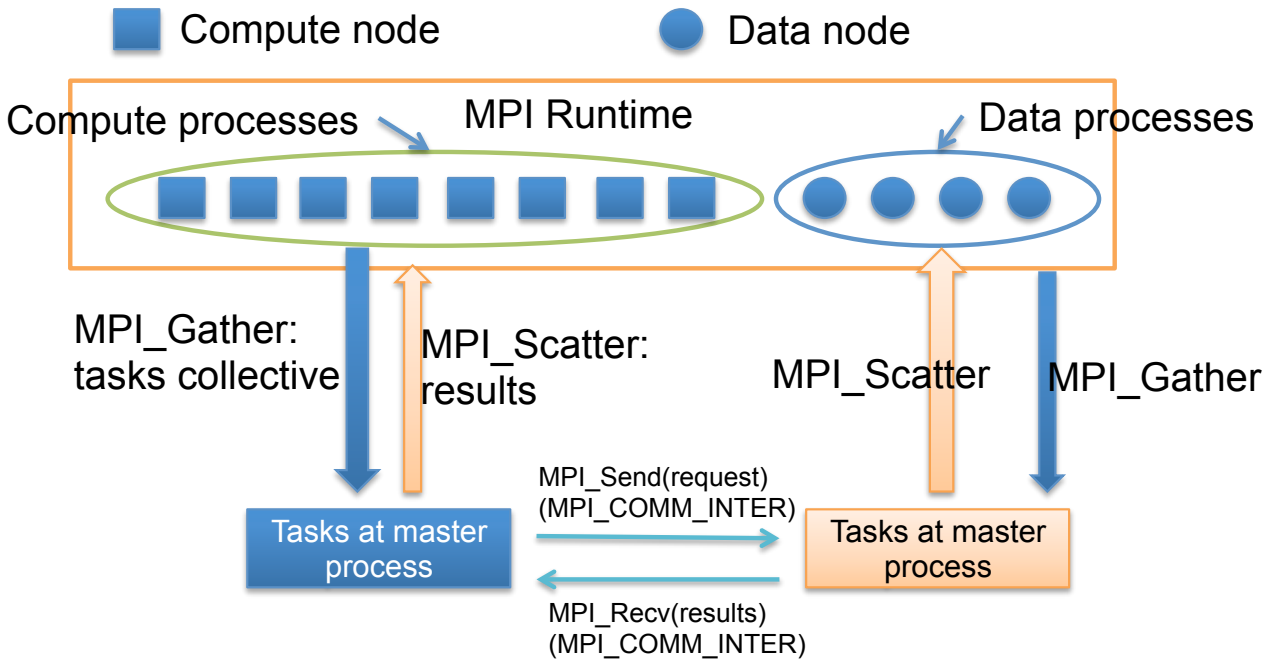

Figure 8: Decoupled I/O prototype

Platform and Setup

Platform:

Name	DISCFarm Cluster(small), Hrothgar Cluster(large)
Num. of nodes	DISCFarm: 16 nodes, Hrothgar: 640 nodes
CPU	DISCFarm: Xeon 2.6GHz, 8 cores, Hrothgar: Westmere 2.8GHz, 12 cores
Memory	DISCFarm: 4GB/node, Hrothgar: 24GB/node

Evaluated Operations:

Data assimilation (ENKF)	read the data, and apply EnKF algorithm (including 6 matrix multiplications, 1 matrix addition, and 1 matrix substitution), then write almost the same size data
Flow-routing	compute the direction where fluids flow to
Summation	calculates the total value of all specified data elements
Lookup	searches for and returns all elements that meet given criteria

Results and Analysis

- Compared against Active Storage (AS)
- 2 storage nodes, 4 data nodes and 8 compute nodes for DEPIO, 12 compute nodes for AS
- Around 13% improvements

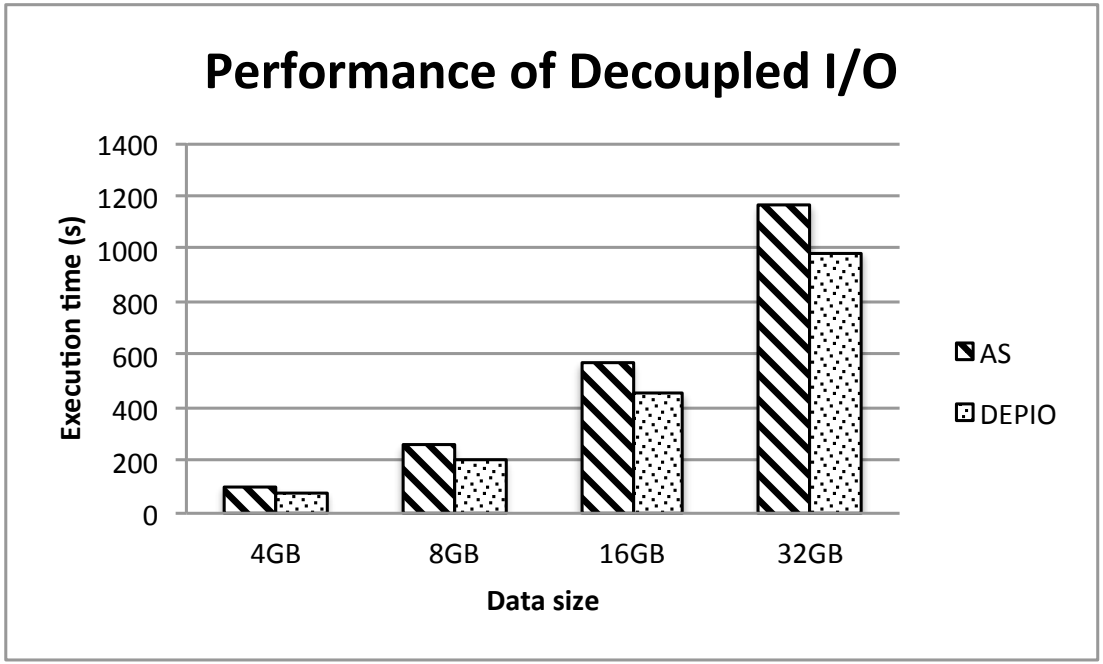

Figure 9: Performance Comparison of Decoupled I/O and Active Storage I/O (Observed CPU usage on storage nodes: 1.3%)

Results and Analysis (with resource contention)

- Workload on storage nodes has great impact on Active Storage performance
- DEPIO keeps better performance than AS with less impact from workload on storage nodes

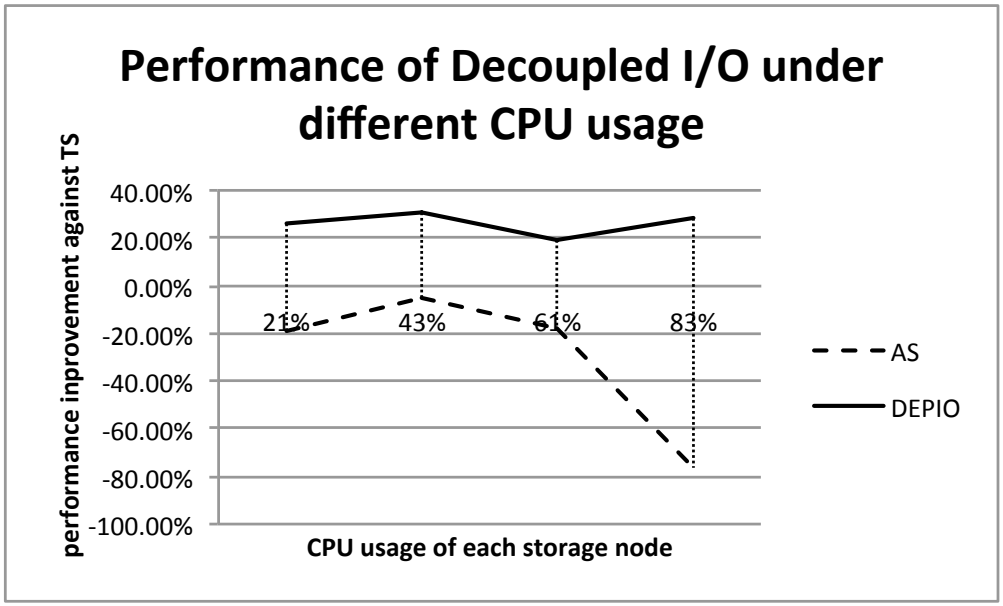

Figure 10: Performance of Decoupled I/O under Different CPU Usages on storage nodes

Results and Analysis

- Up to 60 nodes in the Hrothagar cluster
- Compared against traditional storage I/O (TS)
- Observed 25% performance improvements

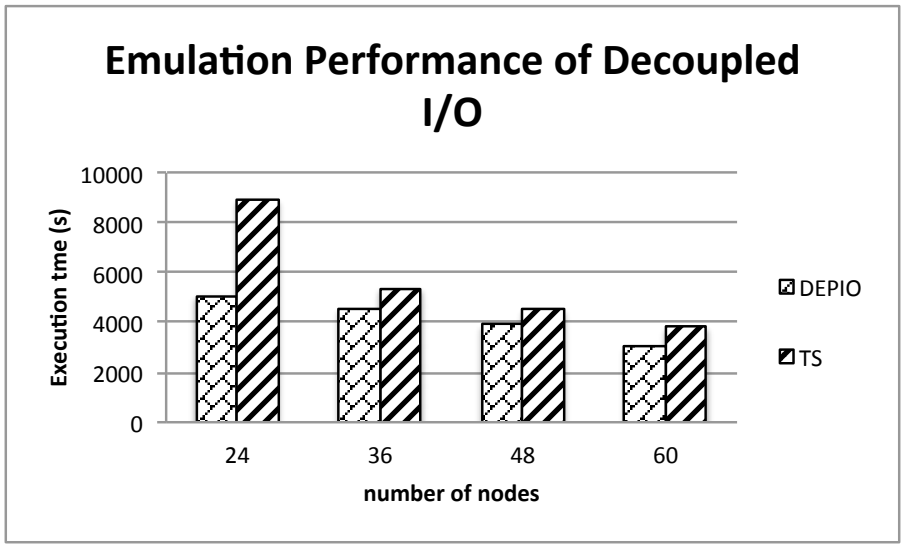

Figure 11: Emulation Performance of the Decoupled I/O

Overhead of the Decoupled I/O

- Primary overhead comes from the communication, Gather, Scatter, etc...
- As the data size of each I/O request increases, this overhead is observed to decrease steadily

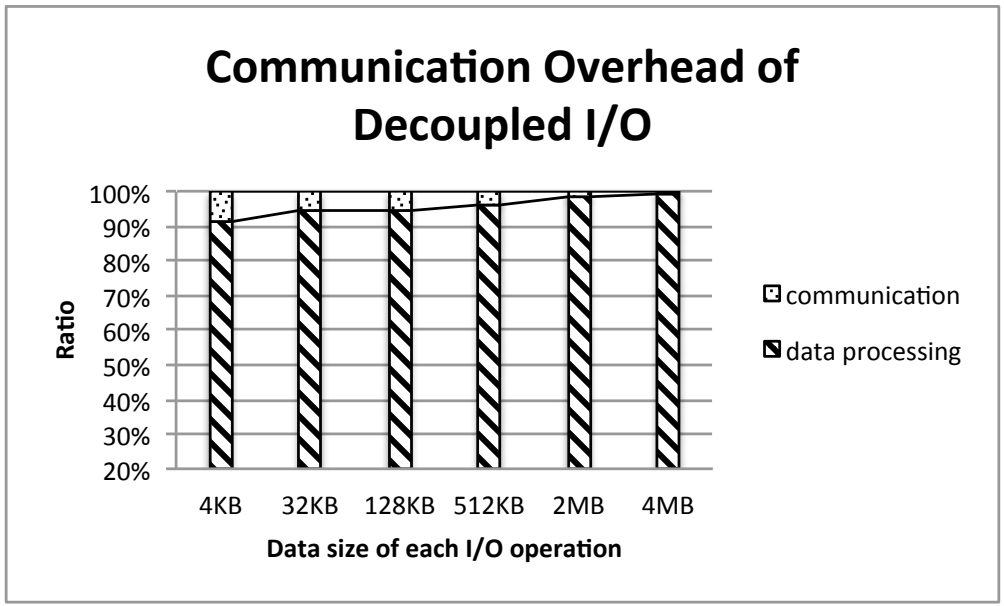

Figure 12: Overhead of the Decoupled I/O Operation

Conclusion and Future Work

- Big data computing brings new opportunities but also poses big challenges
- Dedicating data nodes for data-intensive operations can be helpful and critical for system performance
- An initial investigation of runtime system design to decouple a task into compute-intensive and data-intensive phases
  - Beneficial because of less resource contention, and reduced system wide data movements.
- Prototyping were conducted to evaluate the potential of Decoupled I/O
- Plan to investigate the feasibility of the integration with the MapReduce and in-memory computing model

Thank You

For more information, please visit: `http://discl.cs.ttu.edu`

This research is sponsored in part by the National Science Foundation under the grants CNS-1338078, CNS-1162540, CNS-1162488, and CNS-1161507.

National Science Foundation
WHERE DISCOVERIES BEGIN