High-Performance Data Management, Access, and Storage Techniques for Scientific Applications

PI: *Alok Choudhary*

Team: *M. Kandemir, W. Liao, G. Memik, S. More*
A. Singh, X. Shen, V. Taylor, G. Thiruvathukal

Collaborators (ANL): *M. Papka, R. Stevens, R. Thakur*

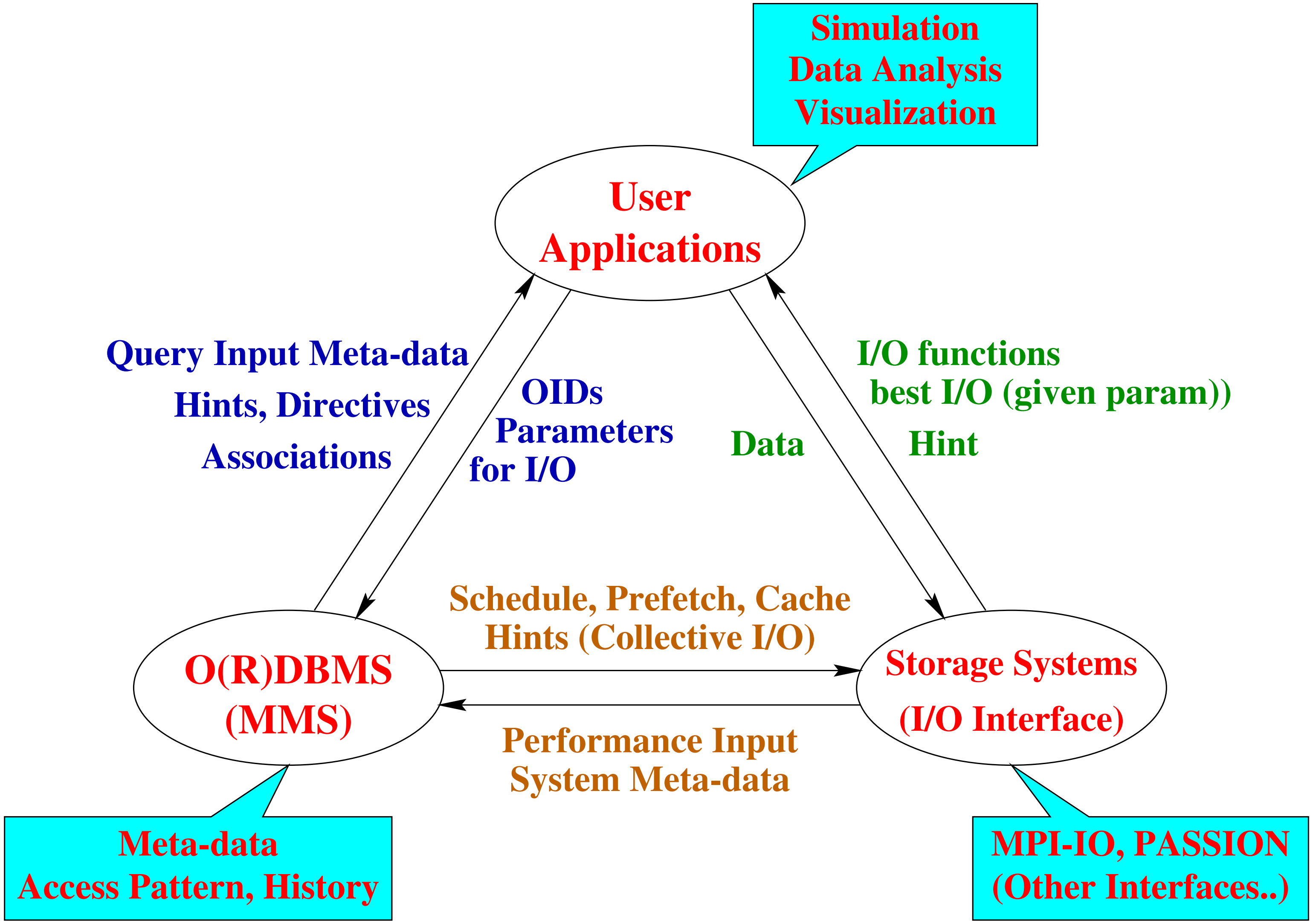
Sponsored by DOE's ASCI Program

Department of ECE, Northwestern University

Goals and Objectives

- **Massive data (100s of GB to 100s of TB) management for large-scale scientific computations**
  - **Record metadata of data storage pattern**
  - **Record data access pattern by parallel applications**
  - **Retrieve historical data usage**
- **High-performance I/O**
  - **Access storage system via high-performance I/O techniques** *(eg. MPI-IO, Parallel File System, PASSION)*
  - **Support hierarchical storage system (HSS)**
  - **Transparent use of prefetching, caching, layout, inter-object relationship, striping, etc.**
  - **Performance modeling and analysis of HSS which is used for performance prediction**

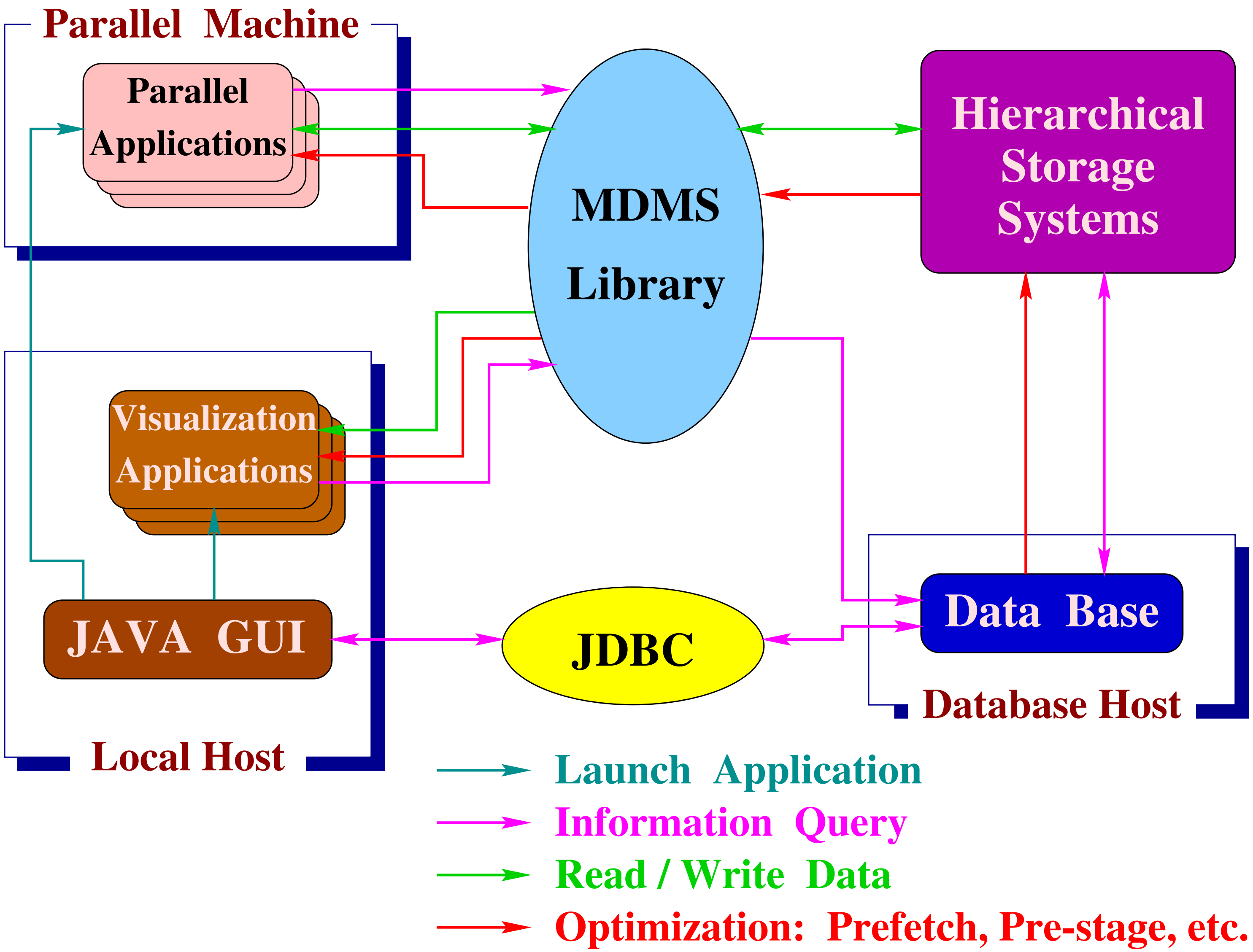
Architecture



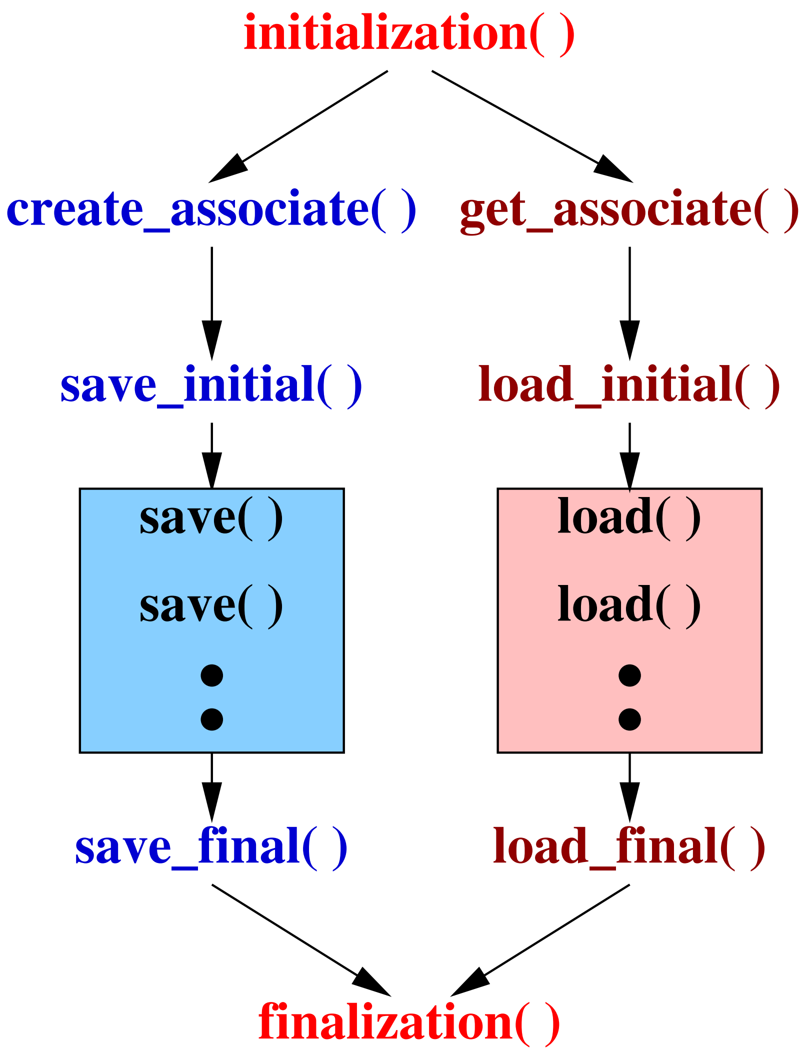
Approach

- **Design and development on OR-DBMS**
  - Metadata: experimental runs, data access/storage patterns, relationships among multiple objects, performance factors, etc.
  - Query Interface: standard SQL
- **Storage and access techniques for HSS**
  - Data storage: layout, striping, etc. using sub-filing strategy
  - Access method: collective I/O, asynchronous I/O
- **Optimizations**
  - Determine optimal parallel I/O calls
  - Pre-fetching and pre-staging: overlapping the latency of future accesses to HSS
  - Pooled striping: pool related objects together as a unit for striping to achieve better parallelism

Data and Control Flow



MDMS Library Interface



- **Initialization**: register application, record arguments for each run
- **Data association**: build relationship between multiple data sets
- **Load**: find data location in the storage system from previous runs, determine best I/O calls by comparing access and storage pattern
- **Save**: choose file names, set file views, and provide hints of optimal I/O calls
- **Finalization**: close files and connection to the database

Meta Data

- **Application Level**
  - Date, parameters, comments, result summary
  - Object id, key values
- **Program Level**
  - Data types, structures
  - Association of data structures, files
- **Performance Level**
  - I/O functions *(eg. collective I/O parameters)*
  - Access pattern, access hints
  - Storage pattern
  - Striping, Pooled striping, storage association
  - Prefetching, staging, migration, caching hints

MetaData Representation

APPLICATION

app id	host	exe file	owner	parameters	time
1	*aixdev*	*astro3d*	*my id*	*int,int,int*	*t1*

RUN TABLE

run id	nproc	dimZ	dimY	dimX	time
1	*4*	*128*	*16*	*32*	*t2*

DATASET TABLE

name	run id	acc id
temperature	*1*	*5*
pressure	*1*	*5*
rho	*1*	*6*

association

ACCESS PATTERN TABLE

acc id	num	ndims	dimension	type	pattern	amode	run id
5	*2*	*3*	*128 16 32*	*float*	*BBB*	*create*	*1*
6	*1*	*3*	*128 16 32*	*double*	*B****	*append*	*1*

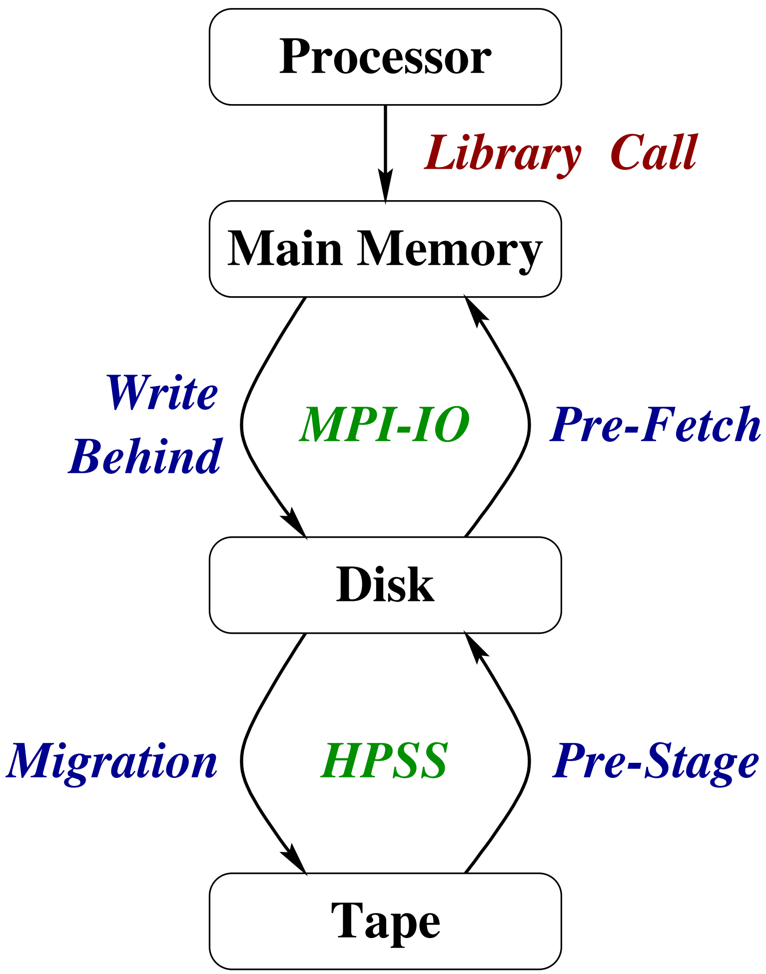
EXECUTION TABLE

run id	dataset	iter #	stg id	offset	max	min
1	*temperature*	*2*	*1*	*0*	*123.4*	*-56.7*
1	*pressure*	*2*	*1*	*262144*	*456.7*	*-11.2*
1	*temperature*	*6*	*1*	*524288*	*89.1*	*-2.3*
1	*rho*	*4*	*2*	*0*	*12.3*	*-567.8*

STORAGE PATTERN TABLE

stg id	run id	nproc	filename	pattern
1	*1*	*4*	*/scratch/astro3d.001.002*	*B***
2	*1*	*4*	*/scratch/astro3d.001.003*	*B***

Sub-filing in Hierarchical Storage System



- **Sub-filing**: tape-resident data sets are divided into chunks and each stored in a separate sub-file
- **Data movement**: is made in sub-files between the hierarchical storage device
- **Pre-stage**: related sub-files are defined by the storage access pattern and moved from tapes to disks
- **Pre-fetch**: extract required data from the relevant sub-files from disk to memory buffer
- **I/O Calls**: asynchronous MPI-IO is used to overlap multiple I/O operations