Research in Middleware Systems For In-Situ Data Analytics and Instrument Data Analysis

Gagan Agrawal
The Ohio State University
(Joint work with Yi Wang, Yu Su, Tekin Bicer and others)

Outline

- Middleware Systems
  - Work on In Situ Analysis
  - Analysis of Instrument Data
- Compression/Summarization of Streaming Data
  - Post analysis using just summary

In Situ Analysis – Simulation Data

- In-Situ Algorithms
  - No disk I/O
  - Indexing, compression, visualization, statistical a

Algorithm/Application Level

Seamlessly Connected?

- In-Situ Resource Scheduling Systems
  - Enhance resource utilization
  - Simplify the management of analytics code
  - GoldRush, Glean, DataSpaces, FlexIO, etc.

Platform/System Level

Opportunity

- Explore the **Programming Model Level** in In-Situ Environment
  - Between application level and system level
  - Hides all the parallelization complexities by simplified API
  - A prominent example: MapReduce

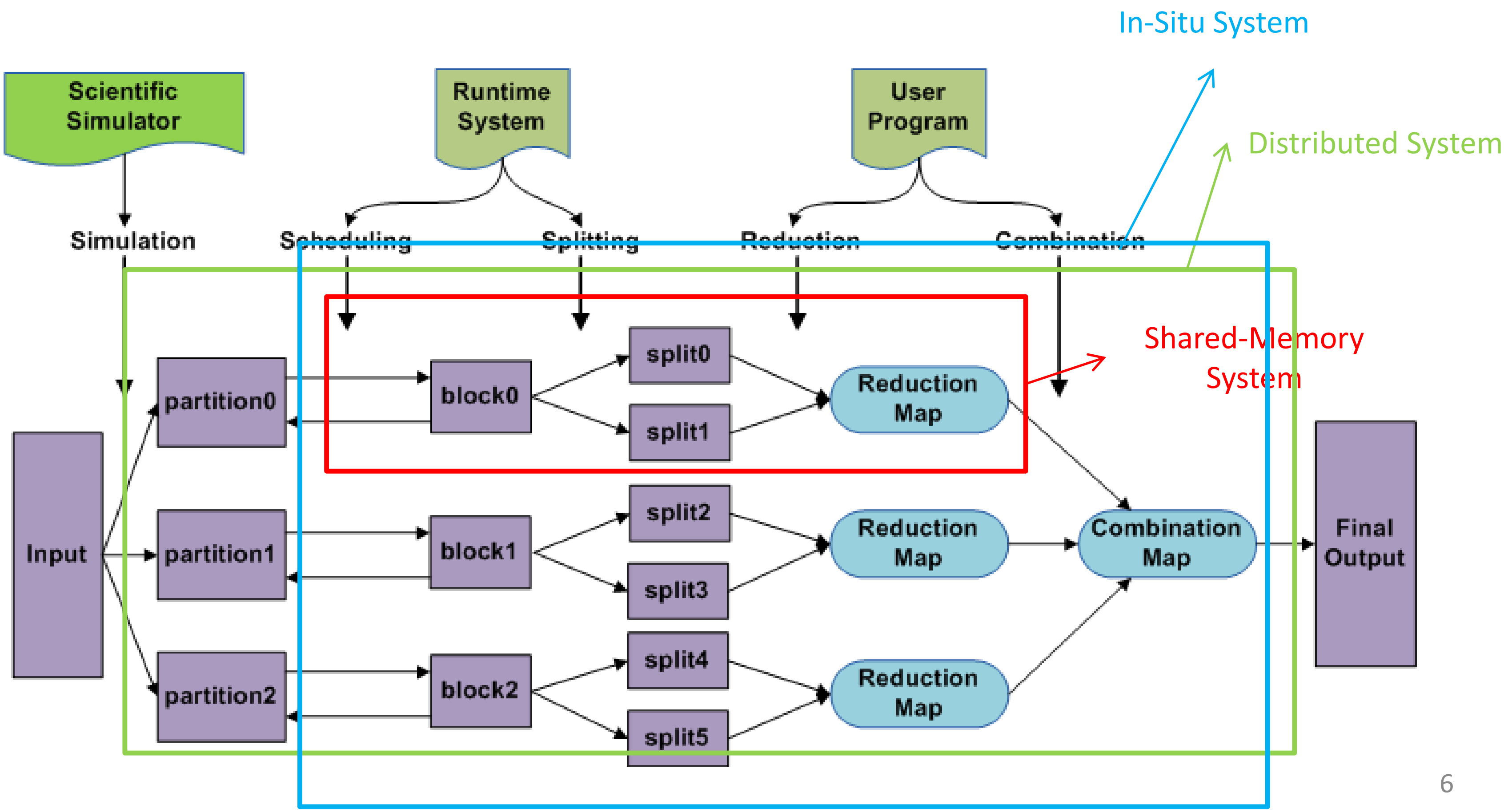
IN SITU + hadoop MapReduce

Challenges

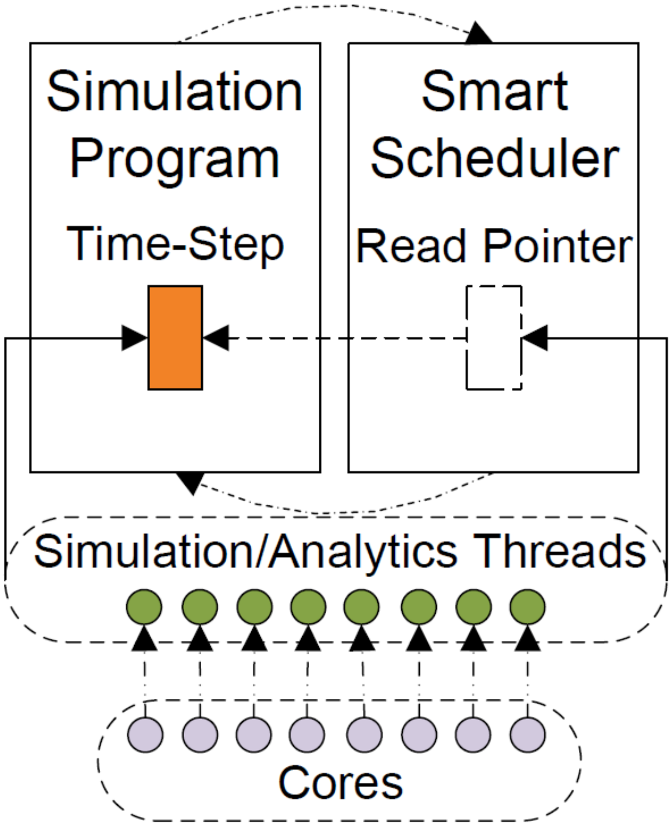
- Hard to Adapt MR to In-Situ Environment
  - MR is not designed for in-situ analytics
- 4 Mismatches
  - Data Loading Mismatch
  - Programming View Mismatch
  - Memory Constraint Mismatch
  - Programming Language Mismatch

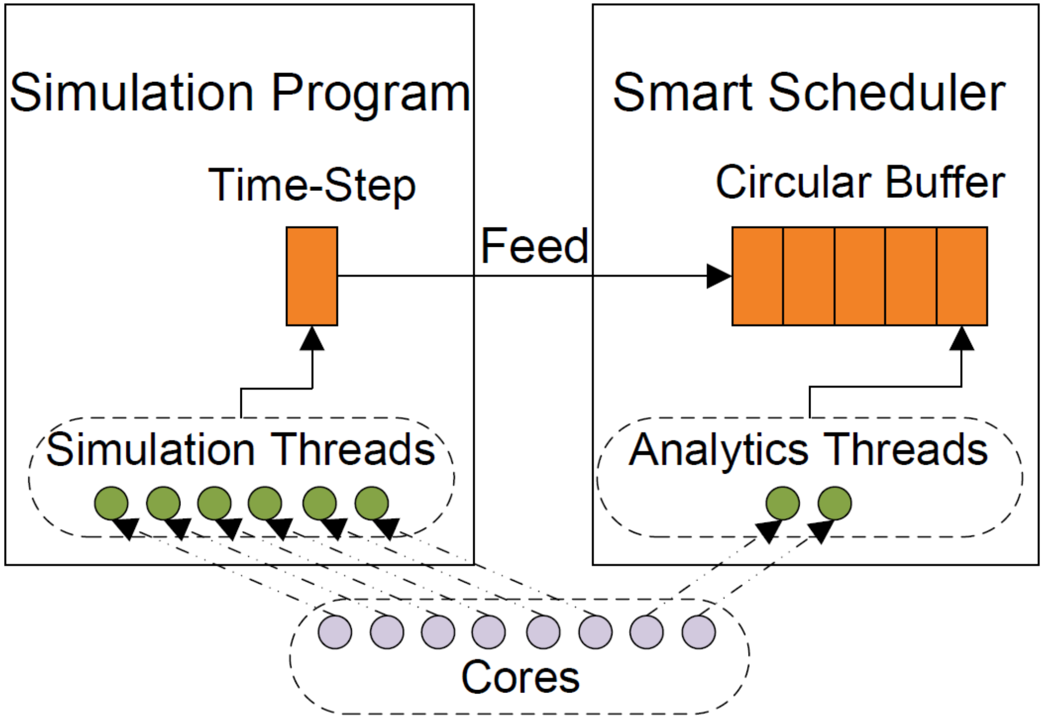
System Overview

In-Situ System = Shared-Memory System + Combination
= Distributed System – Partitioning



Two In-Situ Modes



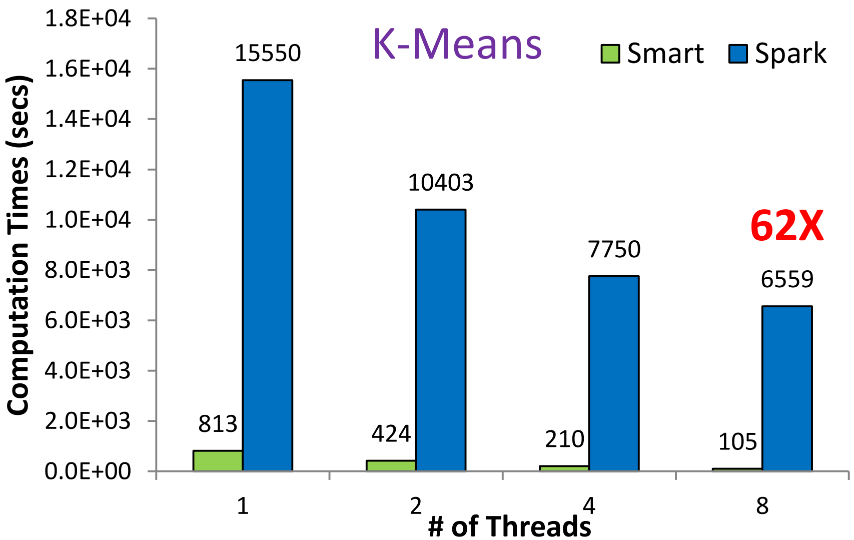


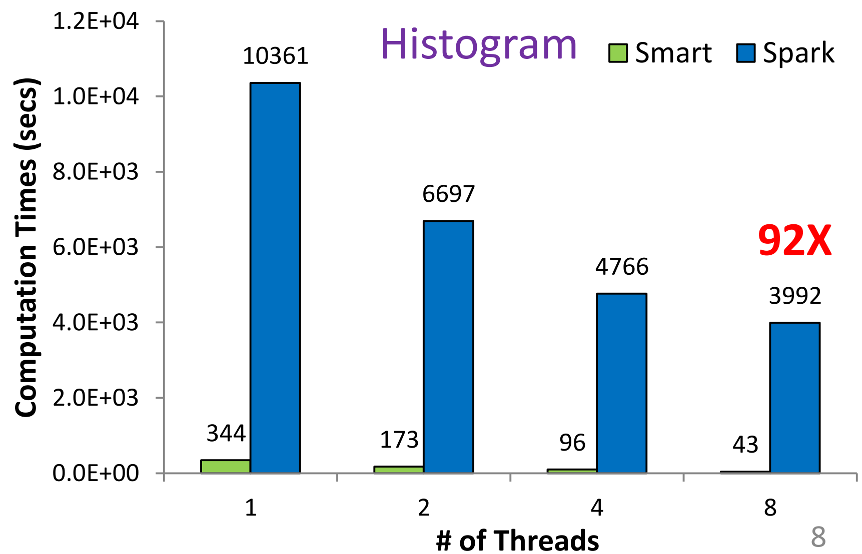
Time Sharing Mode:
Minimizes memory consumption

Space Sharing Mode:
Enhances resource utilization when simulation reaches its scalability bottleneck

Smart vs. Spark

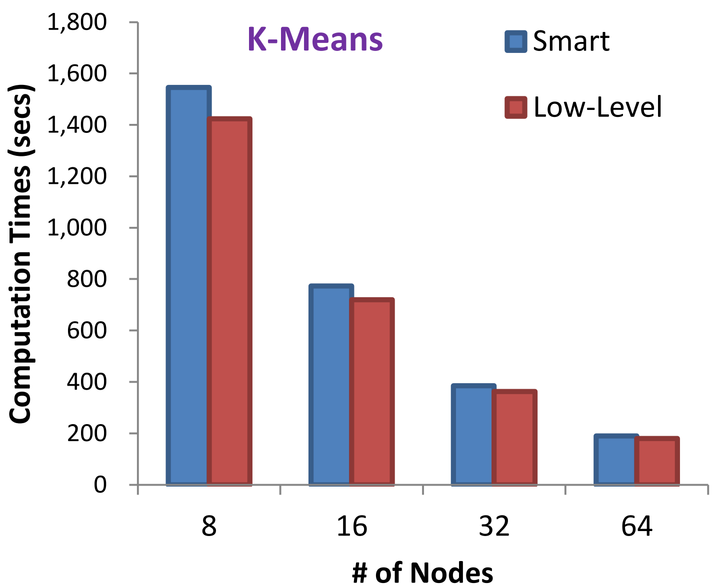
- To Make a Fair Comparison
  - Bypass programming view mismatch
    - Run on an 8-core node: multi-threaded but not distributed
  - Bypass memory constraint mismatch
    - Use a simulation emulator that consumes little memory
  - Bypass programming language mismatch
    - Rewrite the simulation in Java and only compare computation time
- 40 GB input and 0.5 GB per time-step

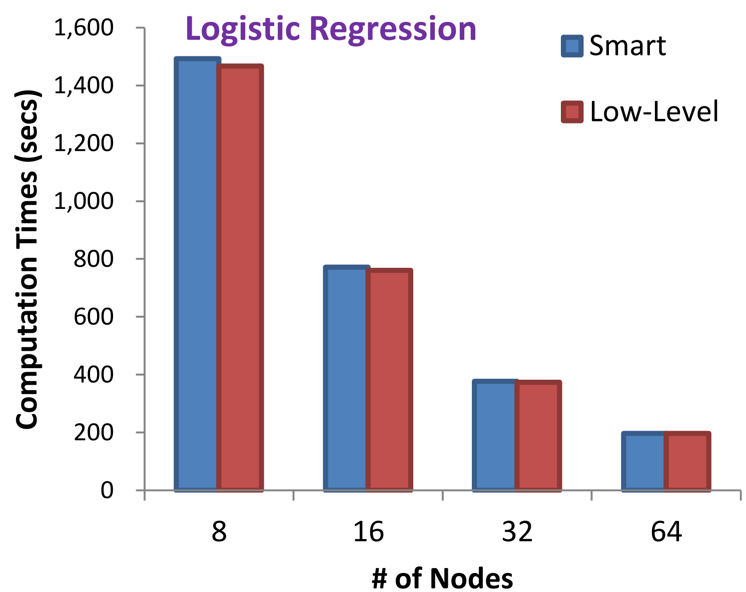




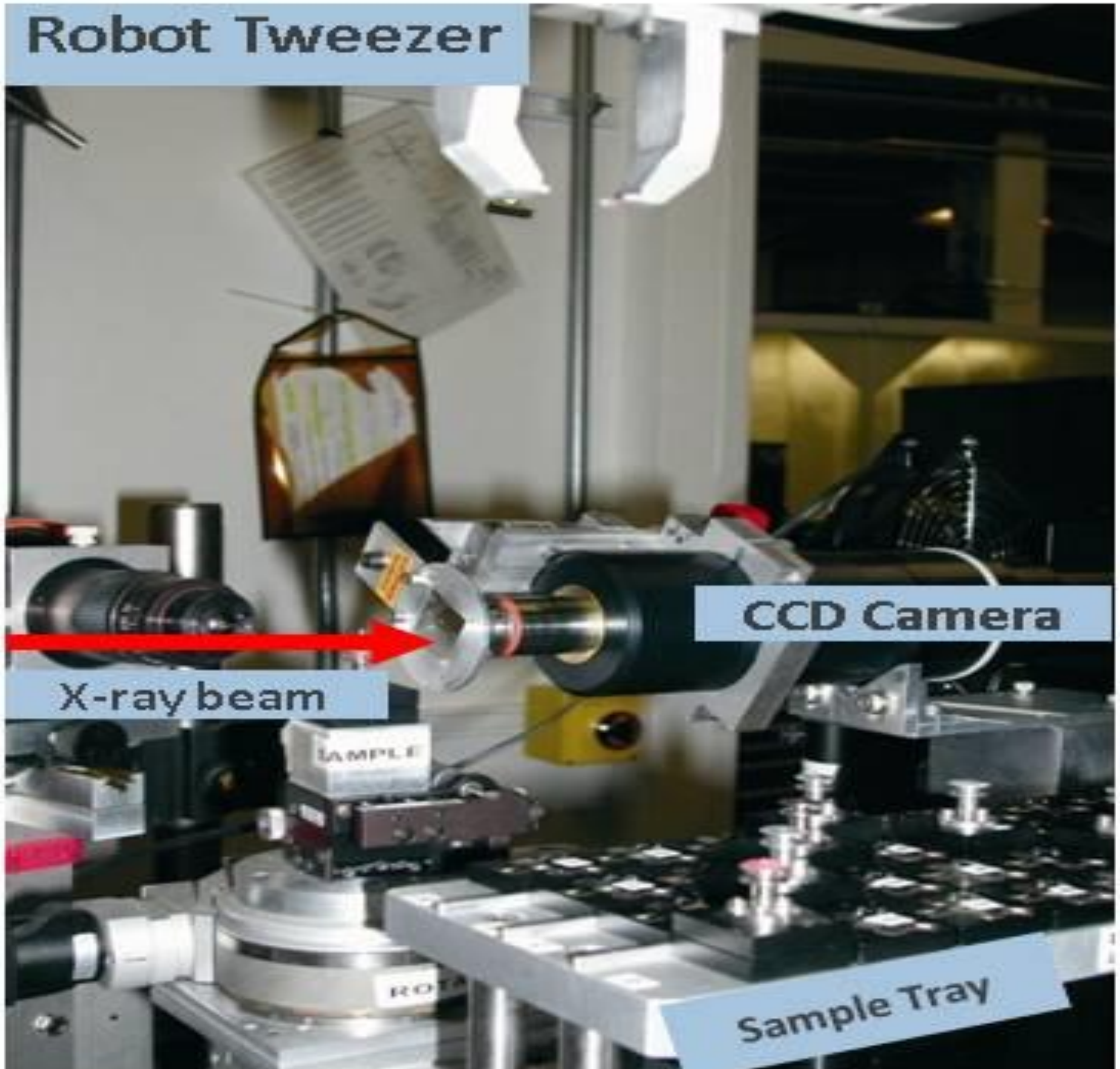
Smart vs. Low-Level Implementations

- Setup
  - Smart: time sharing mode; Low-Level: OpenMP + MPI
  - Apps: K-means and logistic regression
  - 1 TB input on 8–64 nodes
- Programmability
  - 55% and 69% parallel codes are either eliminated or converted into sequential code
- Performance
  - Up to 9% extra overheads for k-means
  - Nearly unnoticeable overheads for logistic regression





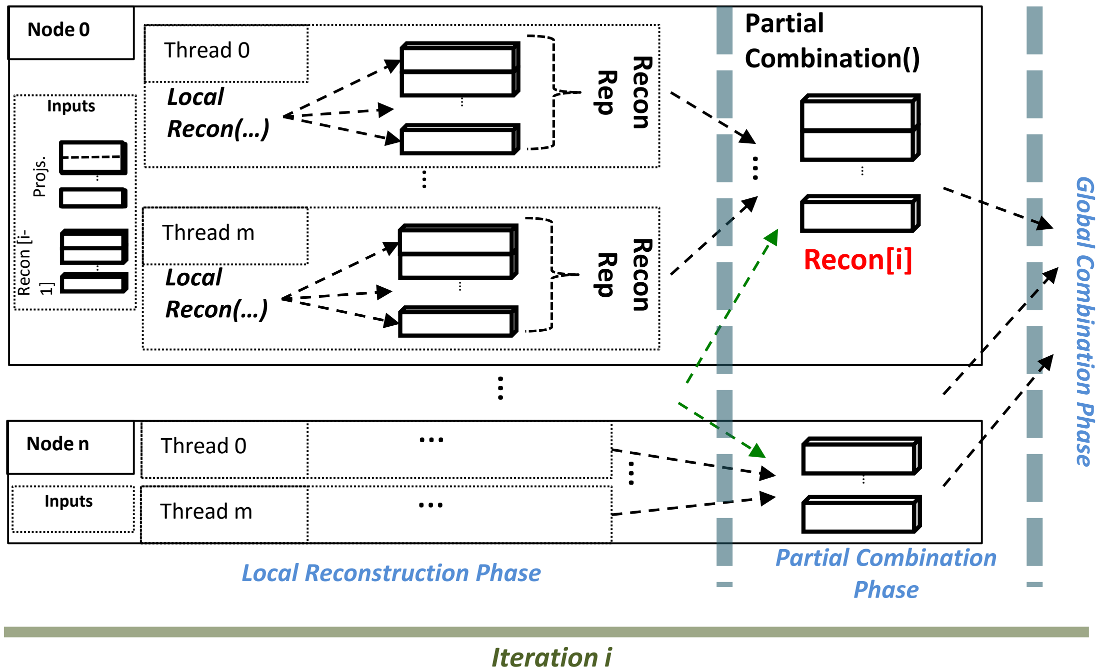
Tomography at Advanced Photon Source



Tomographic Image Reconstruction

- Analysis of tomographic datasets is challenging
- Long image reconstruction/analysis time
  - E.g. 12GB Data, 12 hours with 24 Cores
  - Different reconstruction algorithms
    - Longer computation times
  - Input dataset < Output dataset
    - 73MB vs. 476MB
- Parallelization using MATE+
  - Predecessor of Smart System

Mapping to a MapReduce-like API



Inputs *IS*: Assigned projection slices
Recon: Reconstruction object
dist: Subsetting distance
Output *Recon*: Final reconstruction object

/* (Partial) iteration *i* */
For each assigned projection slice, *is*, in *IS* {
IR = **GetOrderedRaySubset**(*is*, *i*, *dist*);
For each ray, *ir,* in rays *IR* {
(*k*, *off*, *val*) = **LocalRecon**(*ir*, *Recon*(*is*));
ReconRep(*k*) = **Reduce** (*ReconRep*(*k*), *off*, *val*);
}
}
/* Combine updated replicas */
Recon = **PartialCombination**(*ReconRep*)
/* Exchange and update adjacent slices*/
Recon = **GlobalCombination**(*Recon*)

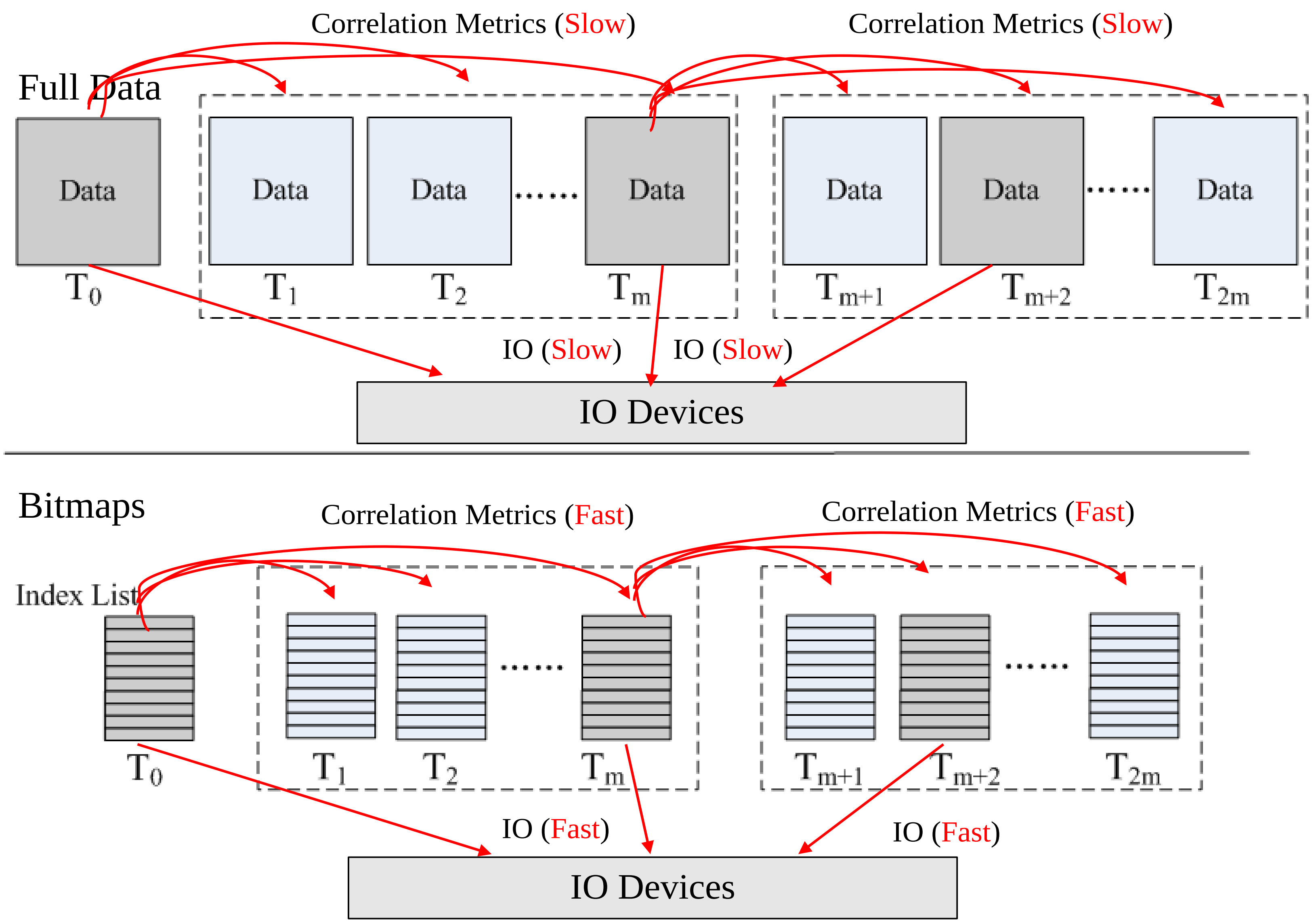
In Situ Analysis

- How do we decide what data to save?
  - This analysis cannot take too much time/memory
  - Simulations already consume most available memory
  - Scientists cannot accept much slowdown for analytics
- How insights can be obtained in-situ?
  - Must be memory and time efficient
- What representation to use for data stored in disks?
  - Effective analysis/visualization
  - Disk/Network Efficient

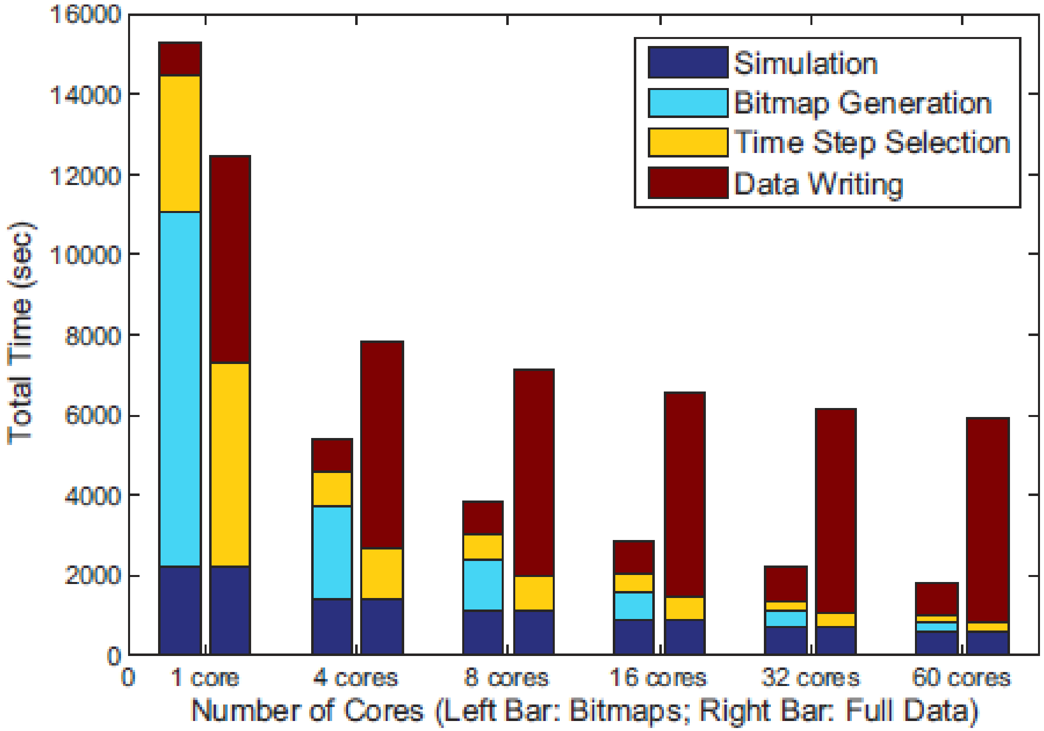
Specific Issues

- Bitmaps as data summarization
  - Utilize extra computer power for data reduction
  - Save memory usage, disk I/O and network transfer time
- In-Situ Data Reduction
  - In-Situ generate bitmaps
    - ✓ Bitmaps generation is time-consuming
    - ✓ Bitmaps before compression has big memory cost
- In-Situ Data Analysis
  - Time steps selection
    - ✓ Can bitmaps support time step selection?
    - ✓ Efficiency of time step selection using bitmaps
- Offline Analysis:
  - Only keep bitmaps instead of data
  - Types of analysis supported by bitmaps

Time-Steps Selection



Efficiency Comparison for In-Situ Analysis - MIC



- Simulation: Heat3D; Processor: MIC
- Time steps: select 25 over 100 time steps
- 1.6 GB per time step (200*1000*1000)
- Metrics: Conditional Entropy

- MIC:
  - More cores
  - Lower bandwidth
- Full Data (original):
  - Huge data writing time
- Bitmaps:
  - Good scalability of both bitmaps generation and time step selection using bitmaps
  - Much smaller data writing time
  - Overall: **0.81x to 3.28x**