Dynamic Compilation for Reducing Energy Consumption of I/O-Intensive Applications

Seung Woo Son, Guangyu Chen,
Mahmut Kandemir
Dept. of CSE
Pennsylvania State University
{sson,gchen,kandemir}@cse.psu.edu

Alok Choudhary
Dept. of ECE
Northwestern University
choudhar@ece.northwestern.edu

The 18th International Workshop on Languages and Compilers for Parallel Computing (LCPC 05)
October 20~22, 2005

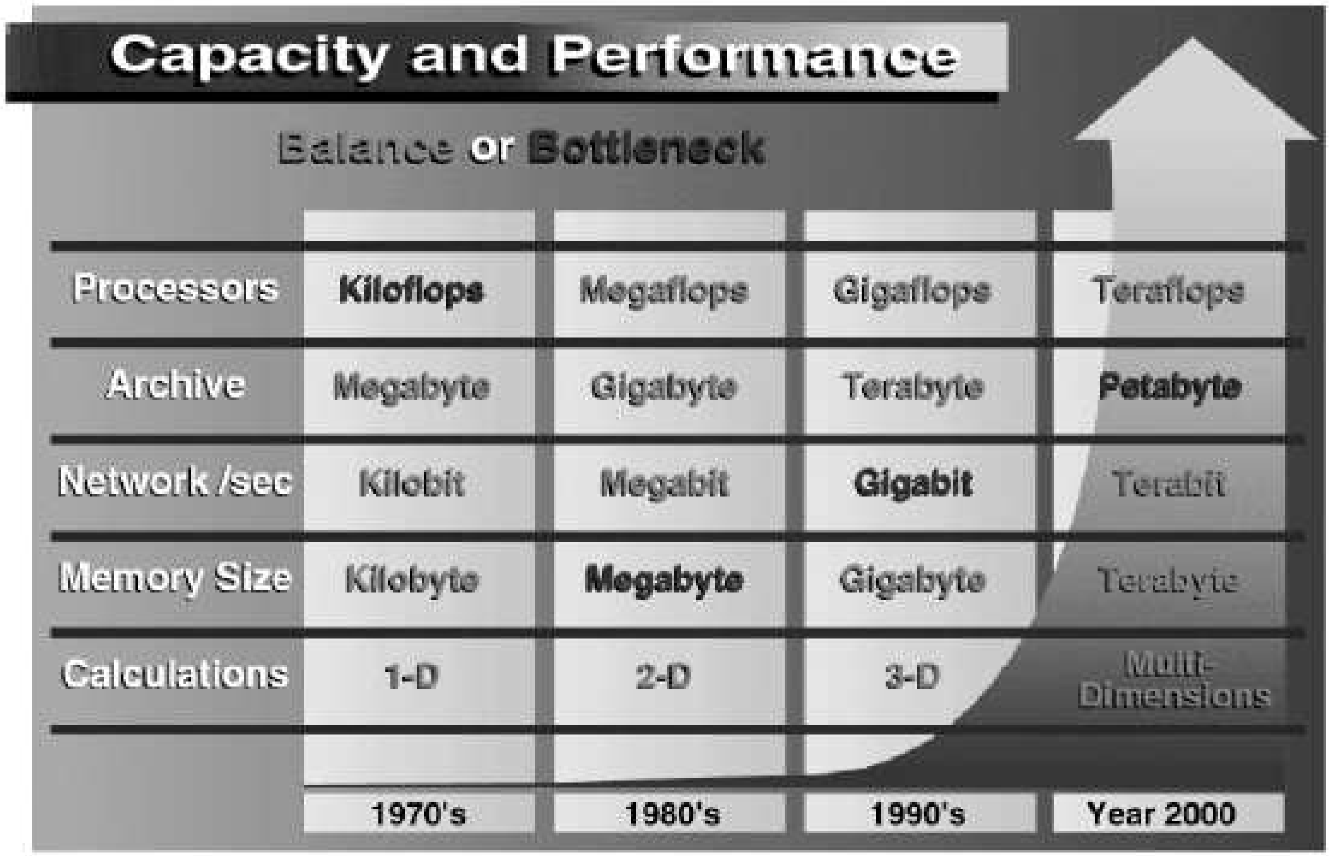
Outline

- Motivation
- Dynamic Compilation
- Our Dynamic Compilation Framework
  - Dynamic compiler/linker
  - Metadata manager
  - Layout manager
  - High-level I/O library
- Experimental Results
- Conclusion

Motivation

- Tera-scale high-performance computing has enabled scientists to tackle very large and computationally challenging problems
  - Data-intensive, I/O-intensive, and energy consuming
- To cope with larger problems and data sizes, models and applications need to be dynamic in nature

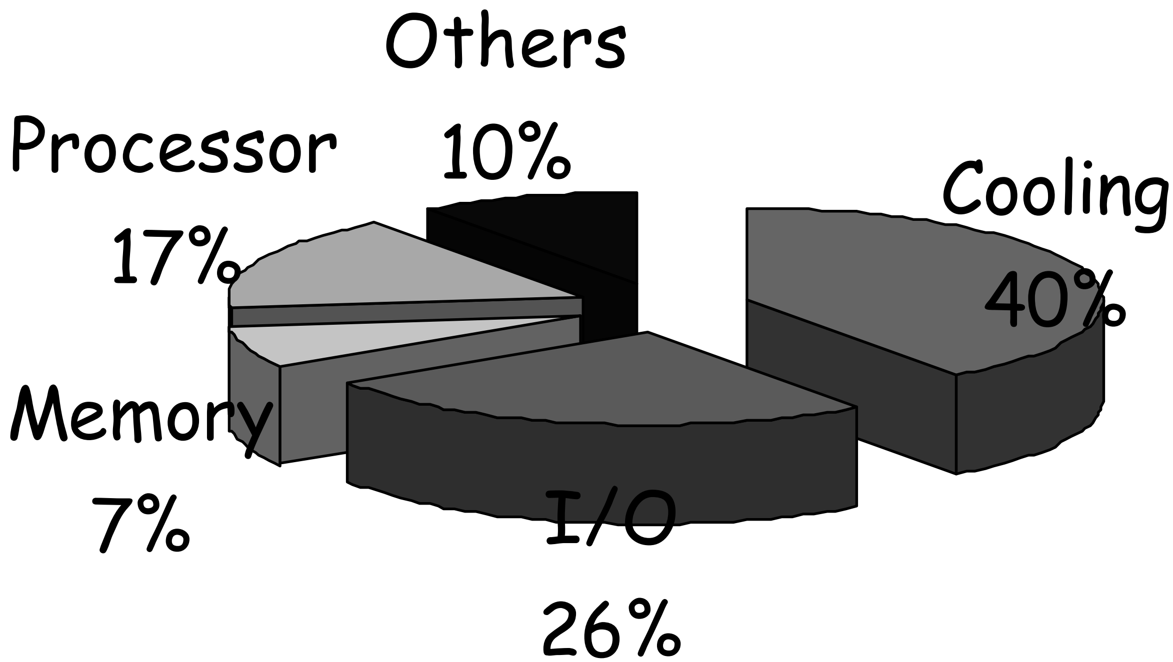
I/O is bottleneck



Increased capacity enables applications to scale accordingly in complexity, throughput, and resultant data.

*Source: Terascale Data Management, LLNL.

Energy Consumption?



*Source: Mike Rosenfield, ACEED, February 2003.

Related Work

- Academic/industry-based dynamic compilers
  - Dynamo, DAISY, PIN, DyC, …
- All efforts focused on enhancing the performance, i.e., their goal is to reduce the execution cycles
- Recently, dynamic voltage/frequency scaling technique proposed using dynamic compilation -> focused on reducing processor's energy consumption [MICRO-38]

Our Goal

- To capture high-level dynamic behaviors in the I/O-intensive applications using dynamic compilers
- Propose a dynamic compilation framework for I/O-intensive applications
  - Dynamic compiler/linker, metadata manager, high-level I/O library, and layout manager

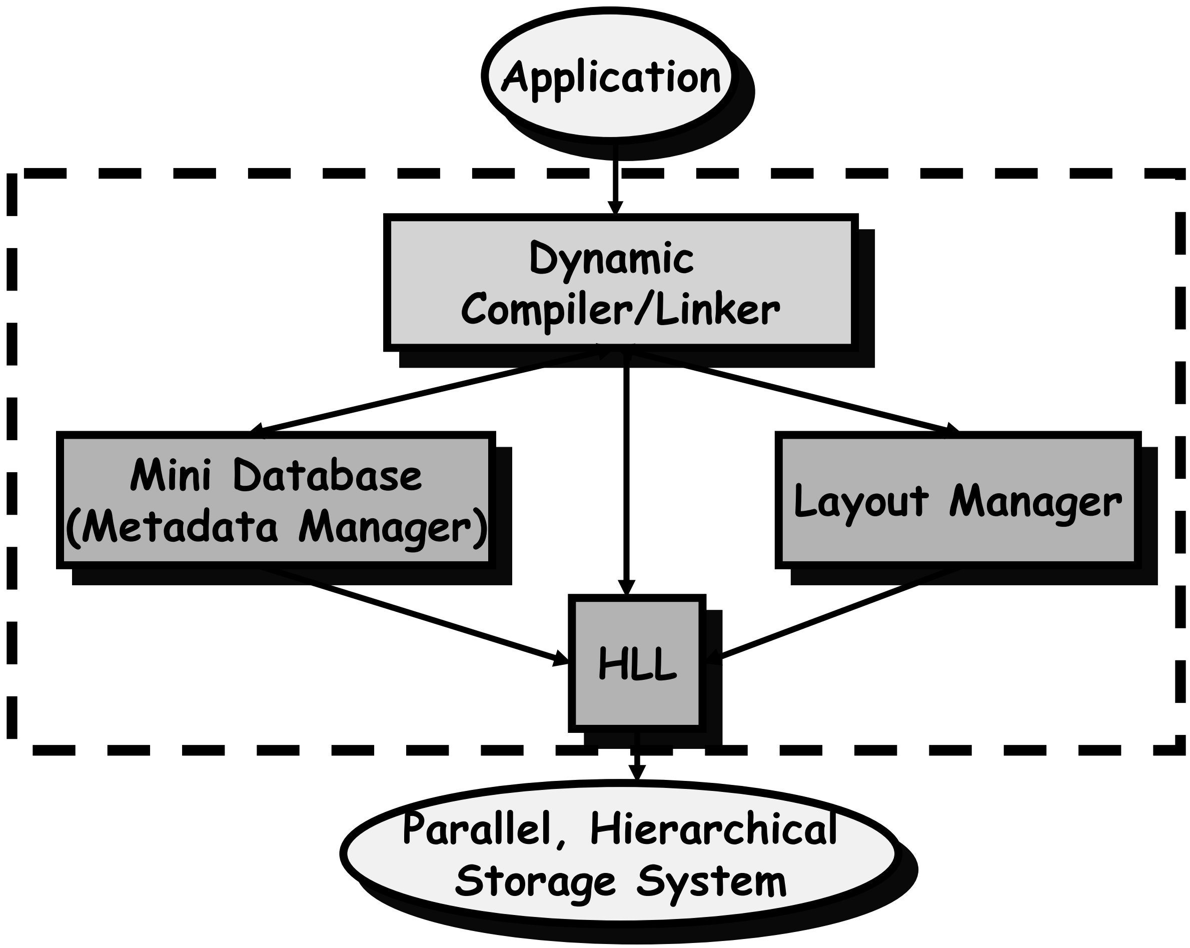
Why Dynamic Compilation?

- Dynamic compilation exploits run-time state to generate code that is specific to run-time behavior
- Large-scale scientific applications exhibit the changes in data access patterns
  - Simulation runs, post-processing, and analysis
  - Large quantities of data are generated and frequent data layout changes occur

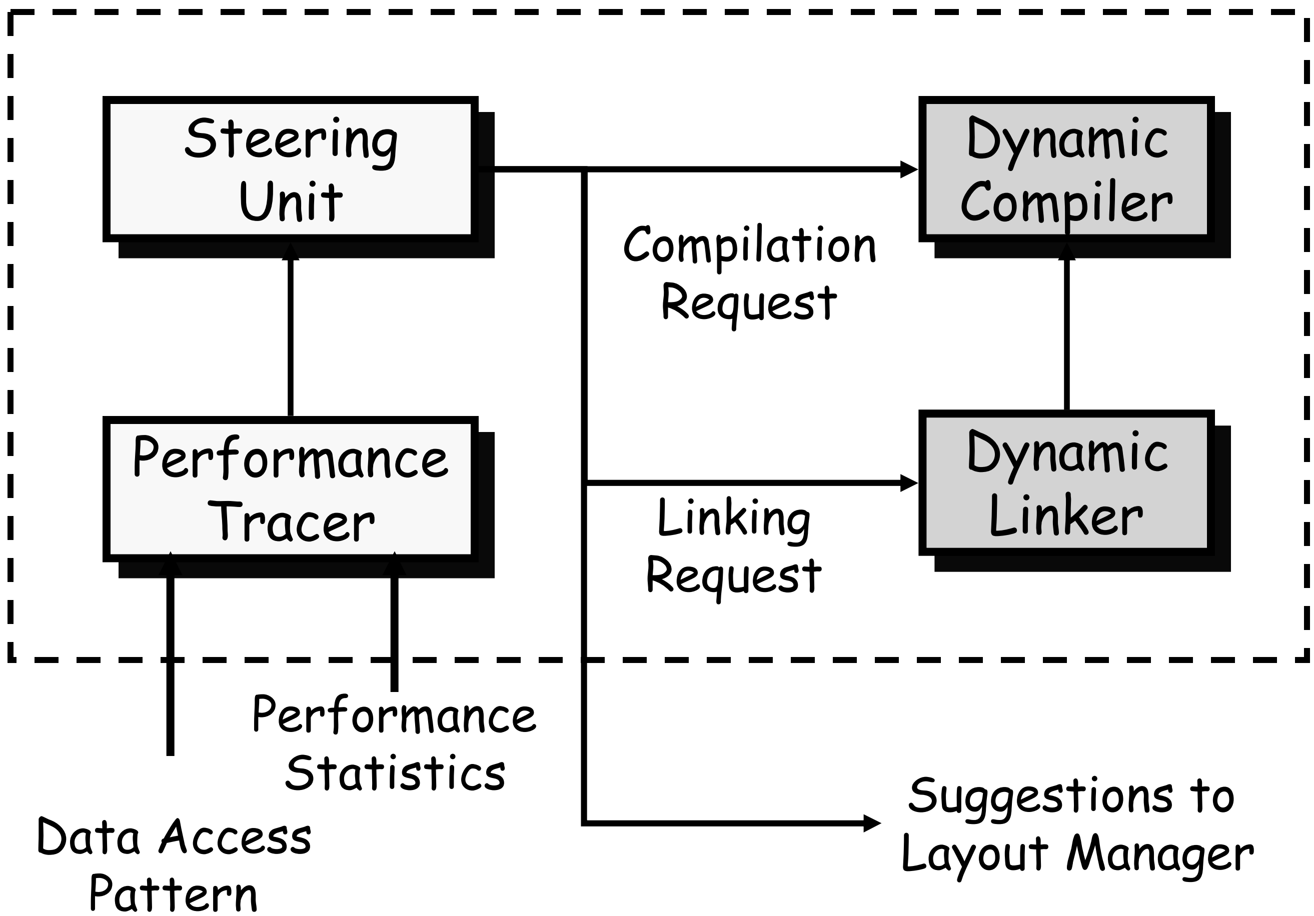
Application Codes

Application Name	Description	Data Size	Number of Phase Changes	Energy consumed (J)
AST	Astrophysics	153.3GB	38	57,322
FFT	Fast Fourier Transform	96.6GB	19	39,451
Cholesky	Sparse Cholesky Factorization	87.4GB	27	36,076
Visuo	3D Visualization	95.5GB	31	42,905
SCF 3.0	Quantum Chemistry	106.1GB	11	49,518
RSense 2.0	Remote Sensing Database	104.0GB	46	51,114

Framework overview



Dynamic Compiler/linker



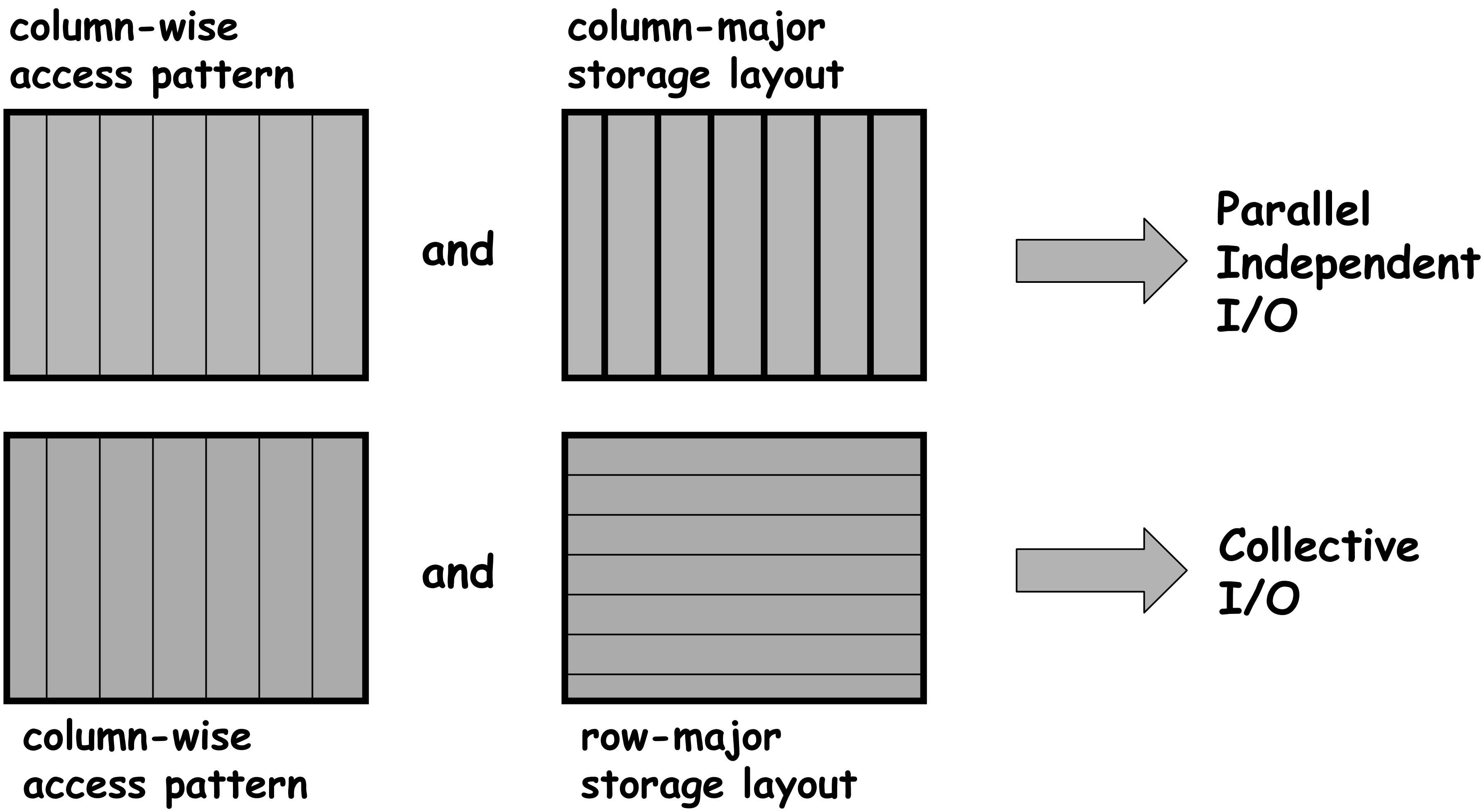
Optimization Rules

Opt rule	Optimization
CIO	Collective I/O
MCIO	Multi-collective I/O
SP	Sequential Prefetching
STD	Strided Prefetching
POL	Replacement Policy Selection
SSU	Setting Striping Unit
DM	Data Migration
DP	Data Purging
PRE	Prestaging
SUB	Subfiling

Optimization Rules

- Collective I/O (CIO)
  - Invoked if access pattern of the data is different from its storage pattern, and multiple processors are used to access the data
- Subfiling (SUB)
  - Invoked if a small subregion of a file is accessed with high temporal locality

Example: CIO


column-wise
access pattern
and
column-major
storage layout
Parallel
Independent
I/O
and
Collective
I/O
column-wise
access pattern
row-major
storage layout

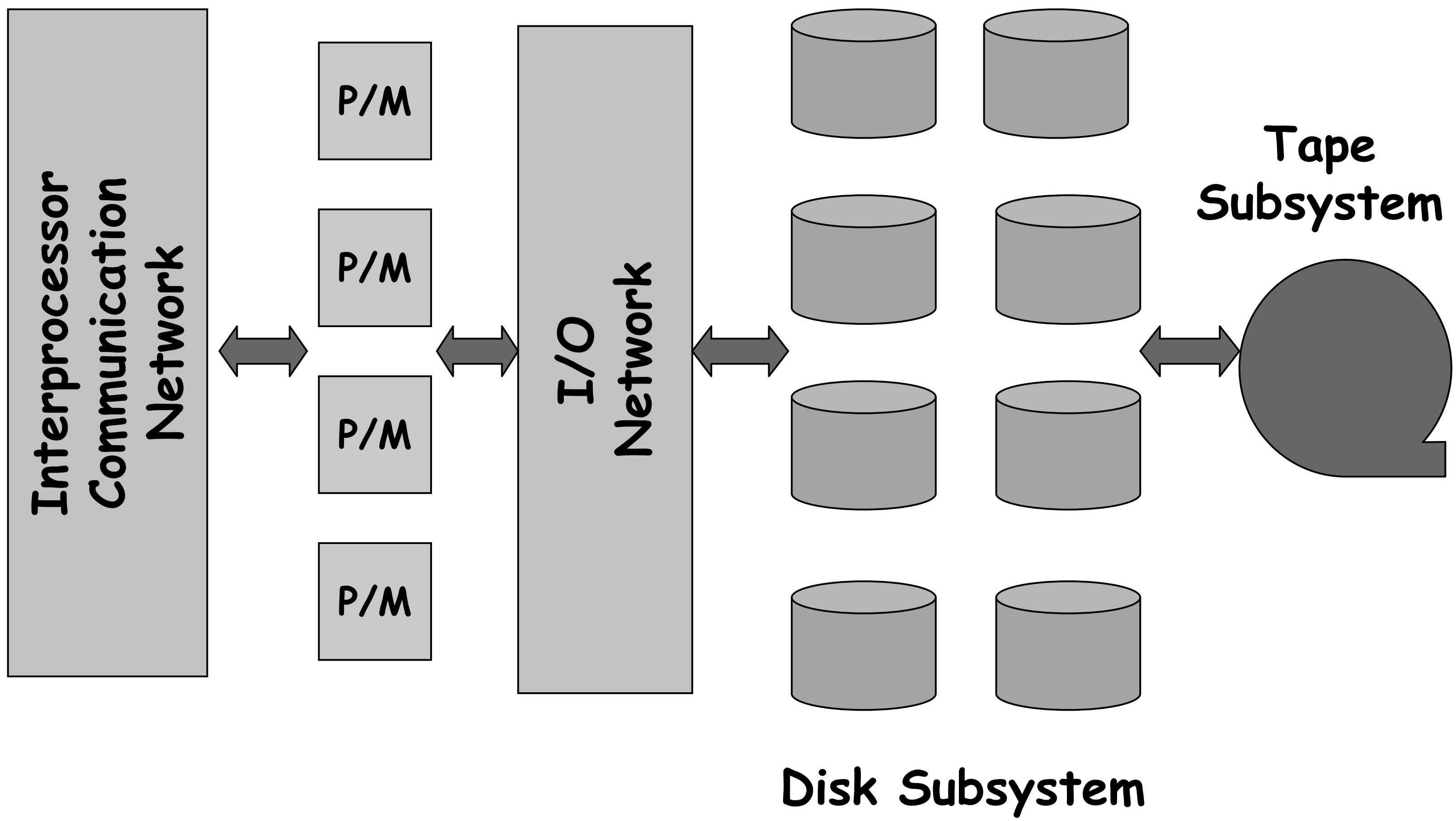
Experiment – application codes

Application Name	Description	Data Size	Number of Phase Changes	Energy consumed (J)
AST	Astrophysics	153.3GB	38	57,322
FFT	Fast Fourier Transform	96.6GB	19	39,451
Cholesky	Sparse Cholesky Factorization	87.4GB	27	36,076
Visuo	3D Visualization	95.5GB	31	42,905
SCF 3.0	Quantum Chemistry	106.1GB	11	49,518
RSense 2.0	Remote Sensing Database	104.0GB	46	51,114

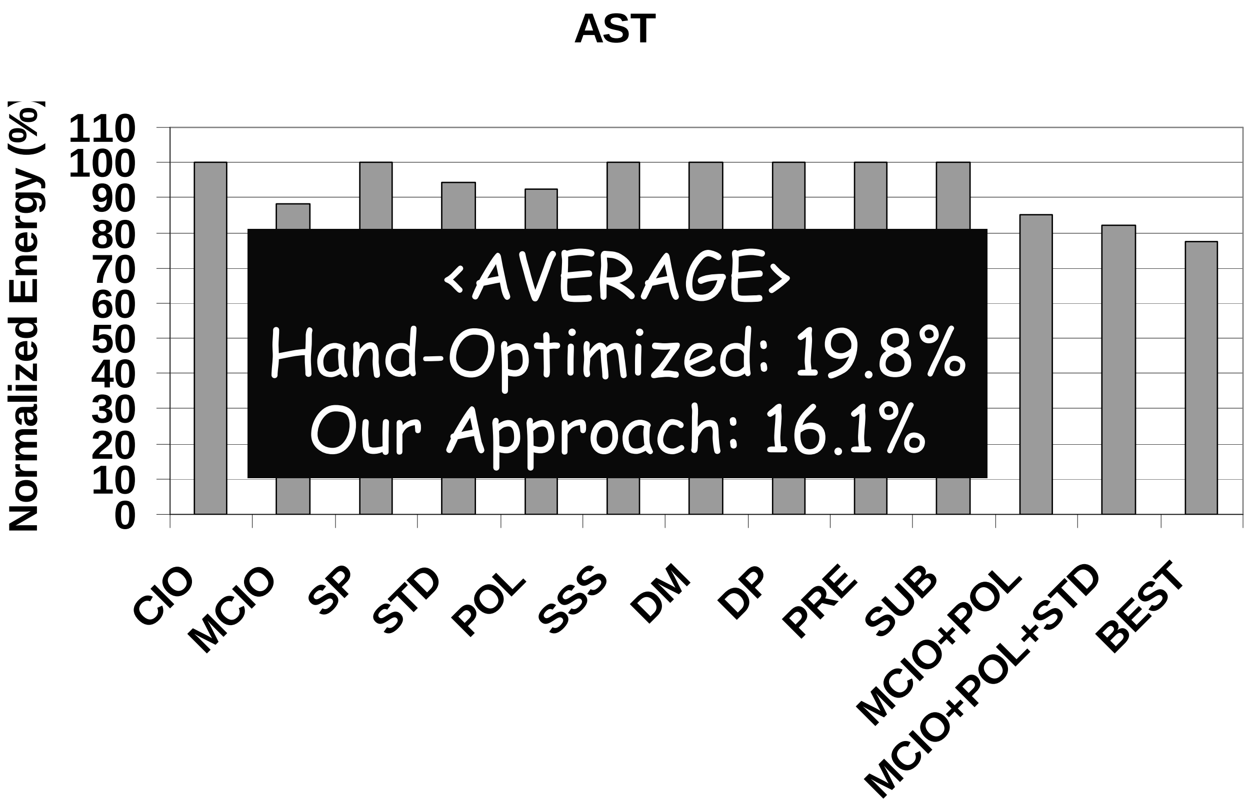
Simulation parameters

- Parallel processors: total 16
  - 1.8 GHz with 2MB 4-way set-associative cache, 1GB main memory
  - Energy consumption measured using Wattch [ISCA'00]
- Parallel disks
  - 8*18GB disks with low-power mode (spin-down)
  - TPM disk power model [ISCA'03]
- Interconnect
  - 2D mesh
  - Infiniband switch/link power model [ISLPED'03]

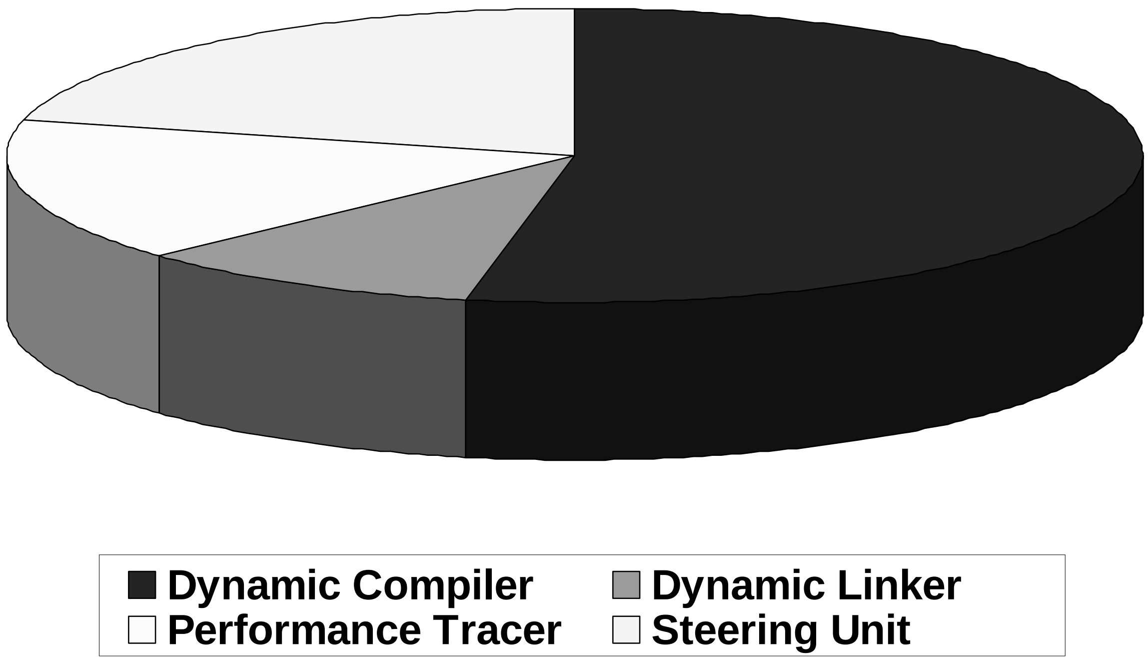
Architecture Considered



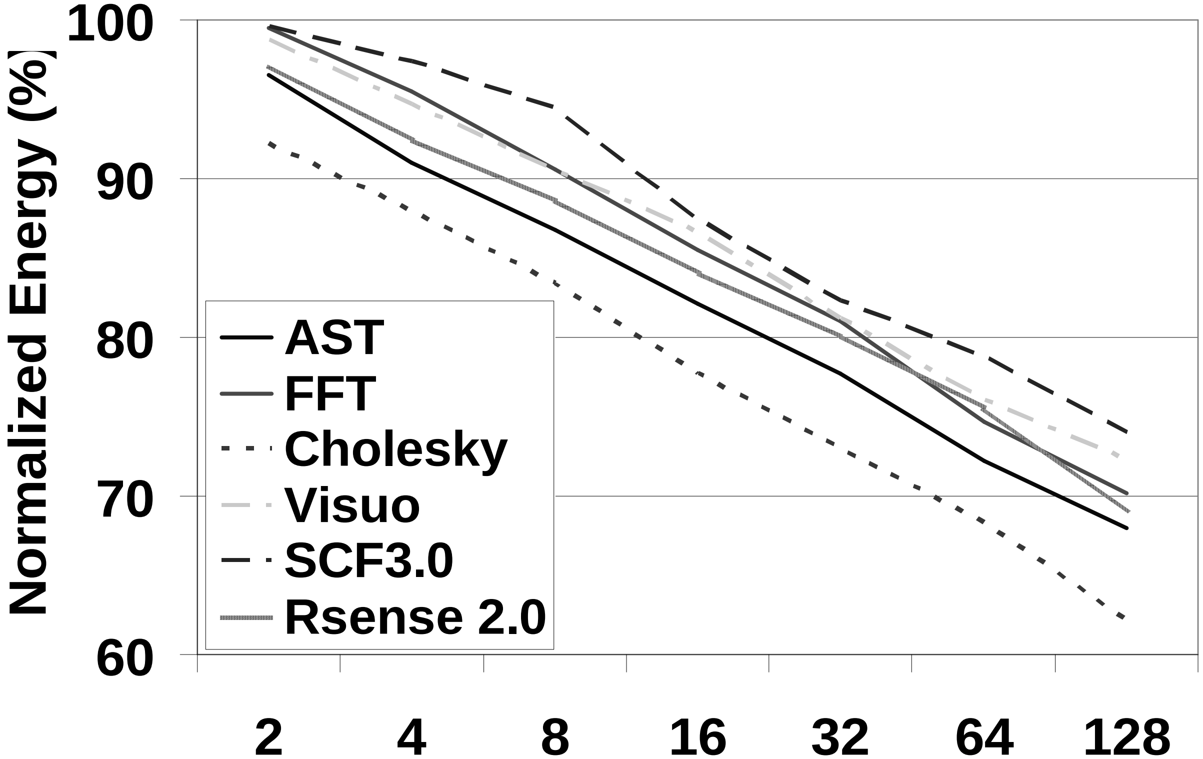
Normalized Energy Consumption



Breakdown of Dynamic Compilation Energy



Sensitivity Analysis – # of processors



Conclusion

- Proposed a dynamic compilation framework for I/O-intensive applications
  - Composed of four components
  - Employ a set of I/O optimizations
- Reduce energy consumption of I/O-intensive applications

Thank you!

sson@cse.psu.edu