System Design Considerations of Big Data Processing

Xian-He Sun

Illinois Institute of Technology

Chicago, Illinois

sun@iit.edu

Addressing the Big Data Challenges

Trends indicate that the "data tsunami" and "memory-wall" will continue, waiting for miracle is not a answer

Big-Data problem is a HPC problem:

- **Data access & Interface**

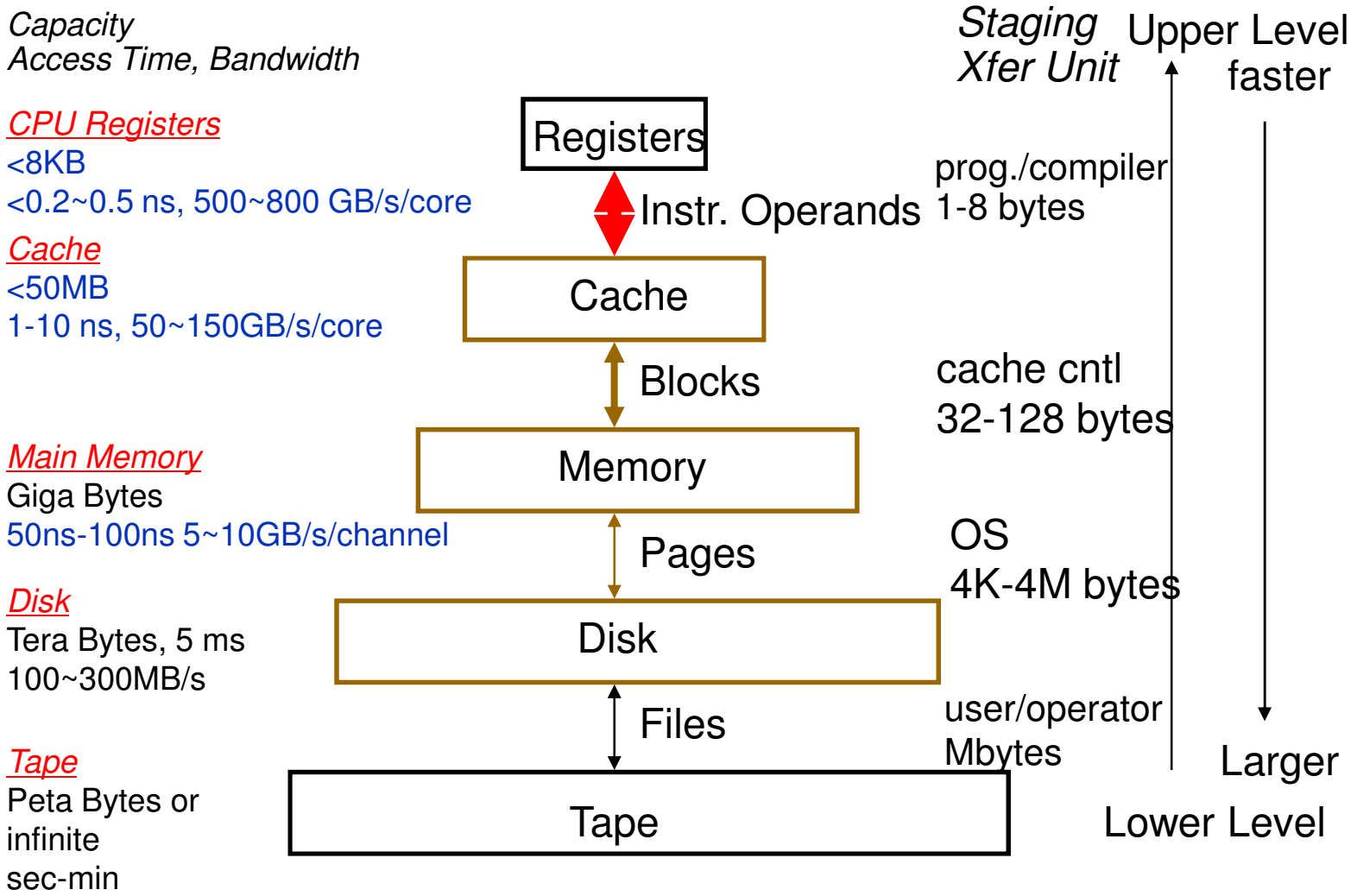
Need **rethinking** from the **data-centric** point-of-view in:

- **Understanding** the system, application, and algorithm relevant to data access
- **Optimizing** current systems
- **Developing** new system architectures
- Developing **integrated solutions**
  - Algorithm, programming model, system, architecture, co-design
  - In situ application-aware data access optimization

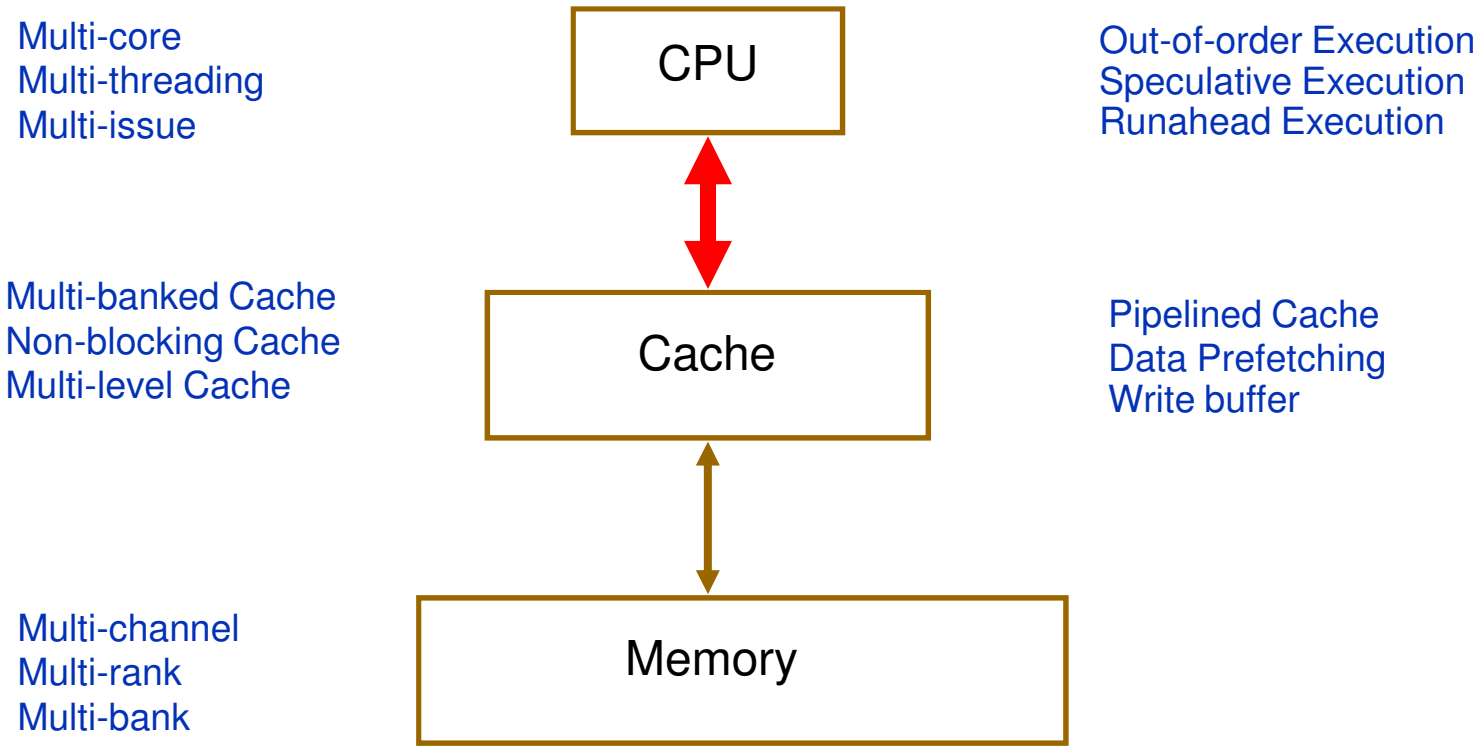
Understanding

MEMORY SYSTEM BEHAVIOR

Improve via Memory Hierarchy



Solution: Memory Hierarchy & Parallelism

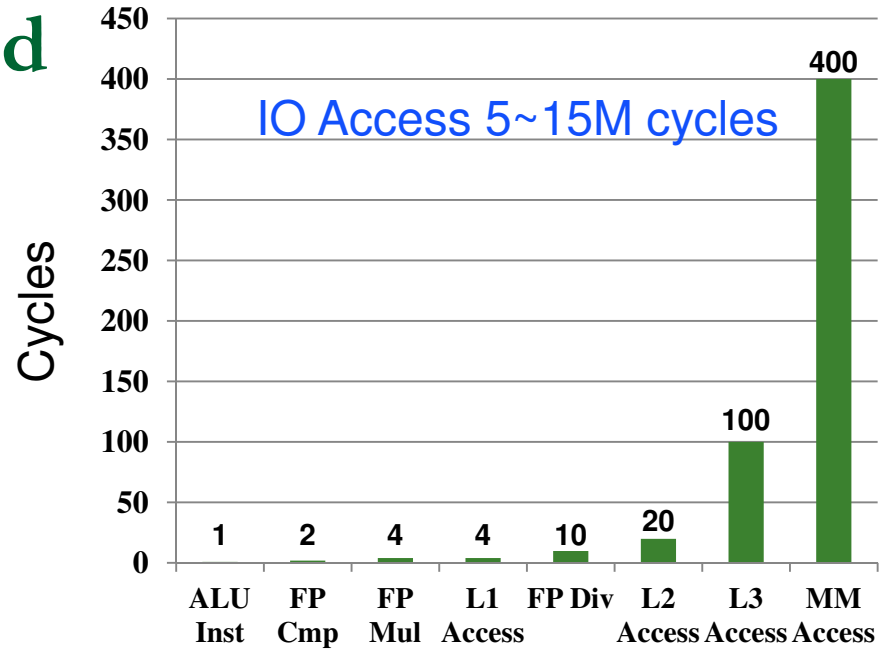


Assumption of Current Solutions

- Memory Hierarchy: Locality
- Concurrence: Data access pattern
  - Data stream

Extremely Unbalanced Operation Latency

Performances vary largely



Existing Memory Metrics

- Miss Rate(MR)
  - {the number of miss memory accesses} over {the number of total memory accesses}
- Misses Per Kilo-Instructions(MPKI)
  - {the number of miss memory accesses} over {the number of total committed Instructions × 1000}
- Average Miss Penalty(AMP)
  - {the summary of single miss latency} over {the number of miss memory accesses}
- Average Memory Access Time (AMAT)
  - AMAT = Hit time + MR×AMP
- Flaw of Existing Metrics
  - Focus on a single component or
  - A single memory access

Missing memory parallelism/concurrency

The Introduction of APC

- Access Per Cycle (APC)
  - APC = A/T
- APC is measured as the number of memory accesses per cycle
  - Measures the overall memory system performance
  - Each memory level has its own APC value
- Benefits of APC
  - Separate memory evaluation from CPU evaluation
  - A better understanding of memory system as a whole
  - A better understanding of the match between computing capacity and memory system performance

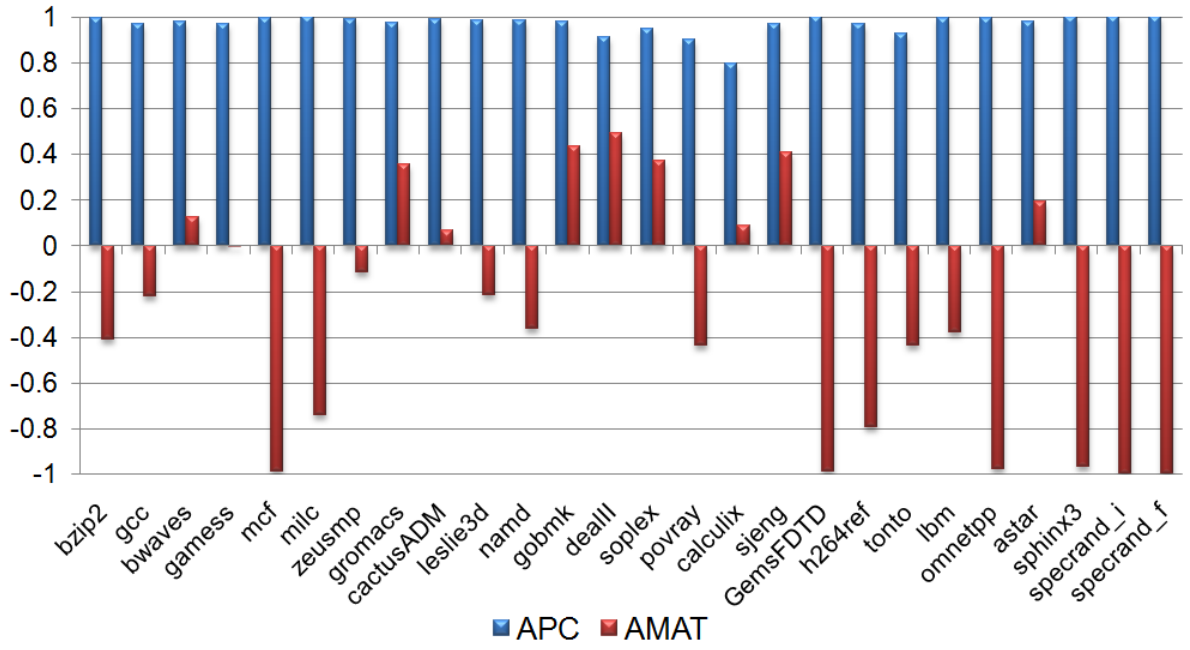
X.-H. Sun and D. Wang, "APC: A Performance Metric of Memory Systems", ACM SIGMETRICS Performance Evaluation Review, Volume 40 , Issue 2, 2012.

APC Measurement

- The difficulty is measuring the total cycle T
  - Hundreds of memory accesses co-exist the memory system

- Measure T based on the ***overlapping mode***
  - When there are several memory accesses co-existing during the same clock cycle, *T* only increases by one
  - Measure the concurrence
  - Measure the concurrence at each level

APC & IPC: Changing Cache Parallelism



- Changing the number of MSHR entries (1→2→10→16)
- APC still has the dominant correlation, with average value of 0.9656
- AMAT does not correlate with IPC for most applications
  - APC record the CPU blocked cycles by MSHR cycles
  - AMAT cannot records block cycles, it only measure the issued memory requests

Exhausted Testing

- With different benchmarks, and with different configurations
- With advanced cache technologies
  - Non-block cache
  - Pipelined cache
  - Multi-port cache
  - Hardware prefetcher
- With single core or multicore
- **APC always has the highest CC values among all the memory metrics**

D. Wang, X.-H. Sun "Memory Access Cycle and the Measurement of Memory Systems", IEEE Transactions on Computers, 2013

APC Applications

- Identify the contribution of concurrence
- Quantitatively **define data intensiveness**
- Find the lowest level that has a dominating correlation with IPC
- Provide a mean to study the matching between memory organization and microprocessor architecture,
- Provide a mean to study the matching between memory organization and a given application
- Co-Design
- **C-AMAT= 1/APC (Architecture Analysis)**

D. Wang and X.-H. Sun, "Concurrent Average Memory Access Time", Illinois Institute of Technology Technical Report (IIT/CS-SCS-2012-05), 2012

Understanding

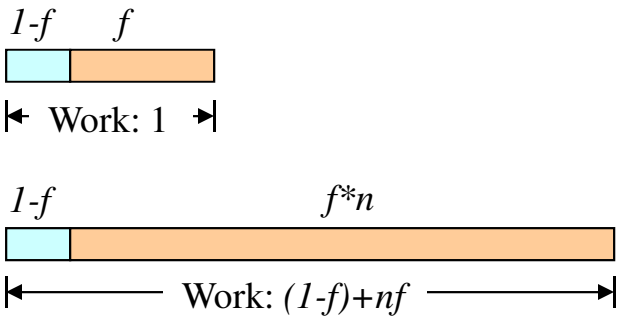
APPLICATION BEHAVIOR

Data Access is Application Dependent

- Conventional algorithm analysis
  - Floating point operation
- Data-centric algorithm analysis
  - Floating point operation
  - Memory requirement
  - Data reuse rate
  - Data access/movement pattern

The Memory-bounded Speedup

- Tacit assumption in Amdahl's law
  - The problem size is fixed
  - The speedup emphasizes time reduction



- Gustafson's Law, 1988
  - Fixed-time speedup model

$$Speedup_{fixed-time} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload}$$
$$= (1-f)+nf$$

- Sun and Ni's law, 1990
  - Memory-bounded speedup model

$$Speedup_{memory-bounded} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload}$$
$$= \frac{(1-f)+fG(n)}{(1-f)+fG(n)/n}$$

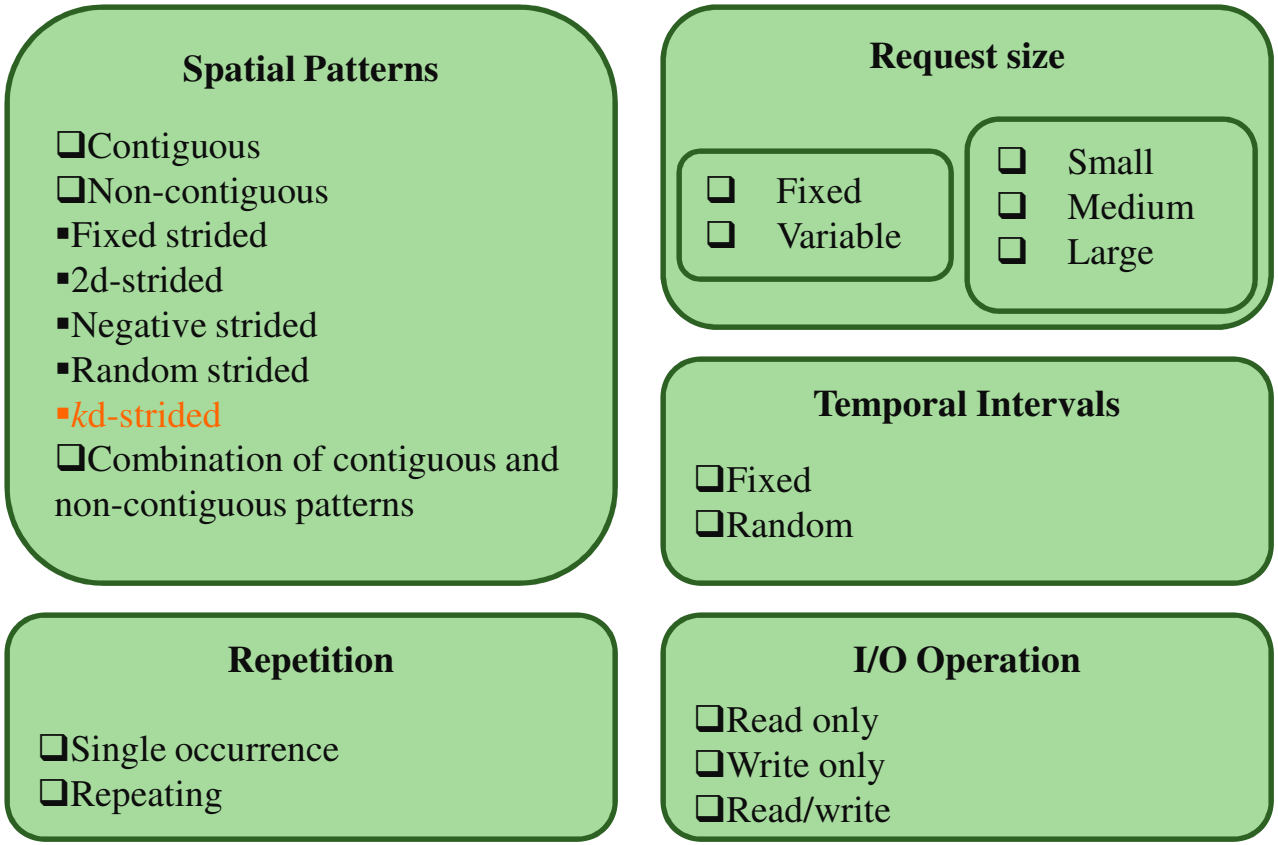
Contribution of Memory-bounded Speedup

- Data-centric thinking
- Where the memory-bound function **W = G(M)** provide
  - W, the work in floating point operation
  - M, the memory requirement
  - G, the data reuse rate
  - Enough for memory hierarchy, but not parallelism
- Need to find the data access/movement patterns for data access concurrency

- Dense Linear Algebra, M memory, $M^{3/2}$ work
- FFT, M memory, O(M *log(*M)) work
- G(pM) > pW, can lead to large increase in execution time
  - (ex) 10K x 10K matrix factorization: 800MB, 1 hr in uniprocessor with 1024 processors, 320K x 320K matrix, 32 hrs

Data Access Signature: Patterns and Notation

- Comprehensive access pattern classification
- Implemented for I/O and memory access (MPI datatype)



S. Byna, X.-H. Sun, et. al, "Parallel I/O Prefetching Using MPI File Caching and I/O Signatures," SC'08

I/O Trace Signature

- Description of a sequence of I/O accesses in a pattern
- Form: *{I/O operation, init position, dimension, ([{offset pattern}, {request size pattern}, {pattern of number of repetitions}], […]), # of repetitions}*

Pattern Signature

- provides a simple description that explains the nature of a pattern
- Form: *{I/O operation, <Spatial pattern, Dimension>, <Repetitive behavior>, <Request size>, <Temporal Intervals>}*

I/O Pattern Detection

- Developed a pattern detection tool
- Five pattern detectors for finding patterns among initial positions, offsets, request sizes, temporality, and repetitions
- Outputs I/O Signature that can be used for prefetching, data layout, and data reorganization

IOSIG: An I/O Characterization Tool

Website: www.cs.iit.edu/~scs/iosig/

Goal: To provide a better understanding of parallel I/O accesses and information to be used for optimization techniques.

Two steps:

Trace collection

- Collects parallel I/O calls of an application
- Does not require any code modification

Trace analysis

- Analyzes the collected information to give a clear understanding of I/O behavior of the application
- Handles trace files in any text-based format

Y. Yin, et.al, "Boosting Application-Specific Parallel I/O Optimization Using IOSIG", in Proc. of IEEE/ACM CCGrid, 2012.

More about IOSIG

- Additional contributions
  - Data access pattern categories
  - Local and global I/O signatures
- Applications

Prefetching	**SC'08**	S. Byna, Y. Chen, X.-H. Sun, R. Thakur, and W. Gropp *Parallel I/O Prefetching Using MPI File Caching and I/O Signatures*
Data Layout	**HPDC11**	H. Song, Y. Yin, Y. Chen, X.-H. Sun, *A Cost-intelligent Application-specific Data layout Scheme for Parallel File Systems*
Data Orga-nization	**PDSW11**	J.He, H. Song, X.-H. Sun, Y. Yin, and R. Thakur Pattern-aware File Reorganization in MPI-IO
Data Replication	**IPDPS13**	Y. Yin, J. Li, J. He, X.-H. Sun, and R. Thakur, *Pattern-Direct and Layout-Aware Replication Scheme for Parallel I/O Systems*

Data Compression

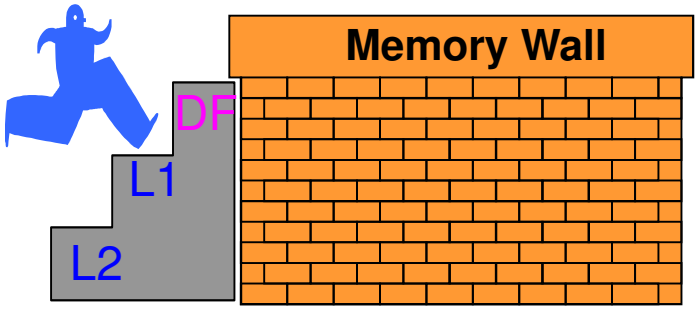
- Website: www.cs.iit.edu/~scs/iosig/

I/O System

OPTIMIZATION *FROM DATA POINT OF VIEW*

In Situ Application-aware Optimization

- Data access is application dependent
- Dynamic, application-aware
- Data access pattern, feed back, control
- Integrating language, memory system, and hardware/software infrastructures
- Understanding design trade-off



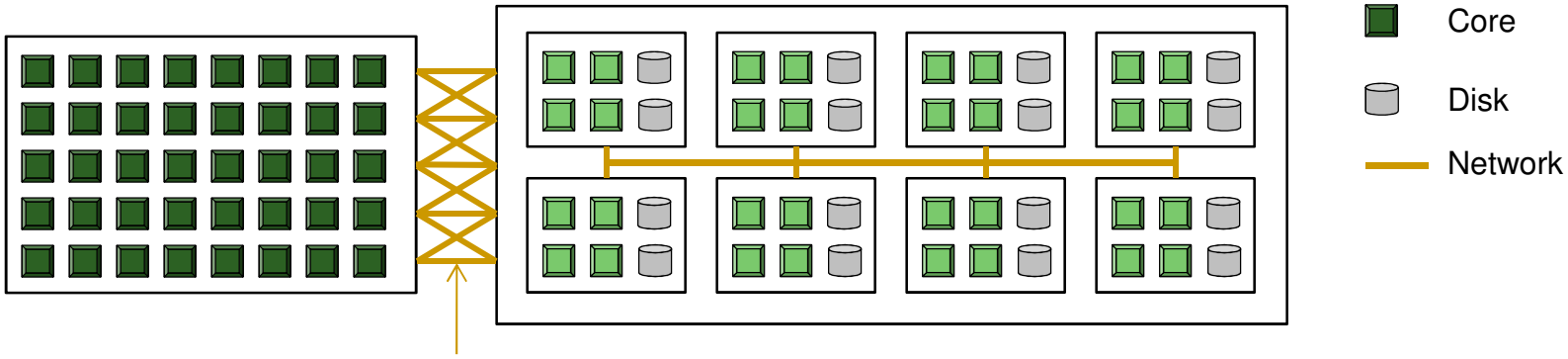
Ours Method: Application-Aware I/O (1)

Decoupled-Execution Paradigm:

- Handle computation- and data- intensive phases separately
- One interface-Two systems, transparent to users
- Integration, scheduling, optimization

Supercomputer or
many-core computing system
for execution of computing intensive part of an application

Data cloud or ***storage cluster***
for execution of data intensive part of an application



Application-Aware I/O Optimization (2)

Interoperability between different file systems

- Enable MPI Apps to access data-intensive file systems
- HPC-Cloud, Data-Cloud

H. Jin, X.-H. Sun, et. al, "CHAIO: Enabling HPC Applications on Data-Intensive File Systems", **ICPP2012**.

Conclusion: Big Data Rethinking

- Power & fault-tolerant depending on data handling
- Big Data requires the rethinking of program model, system, algorithm, and architecture
- Big Data is a community effort
- Understand the memory system and application data behavior is a must
- Integrated data-access system design is the first step