

Caching and Prefetching for Fast Storage Data Accesses

Xiaodong Zhang

Ohio State University

Song Jiang

Wayne State University

“Disk Wall” is a Critical Issue

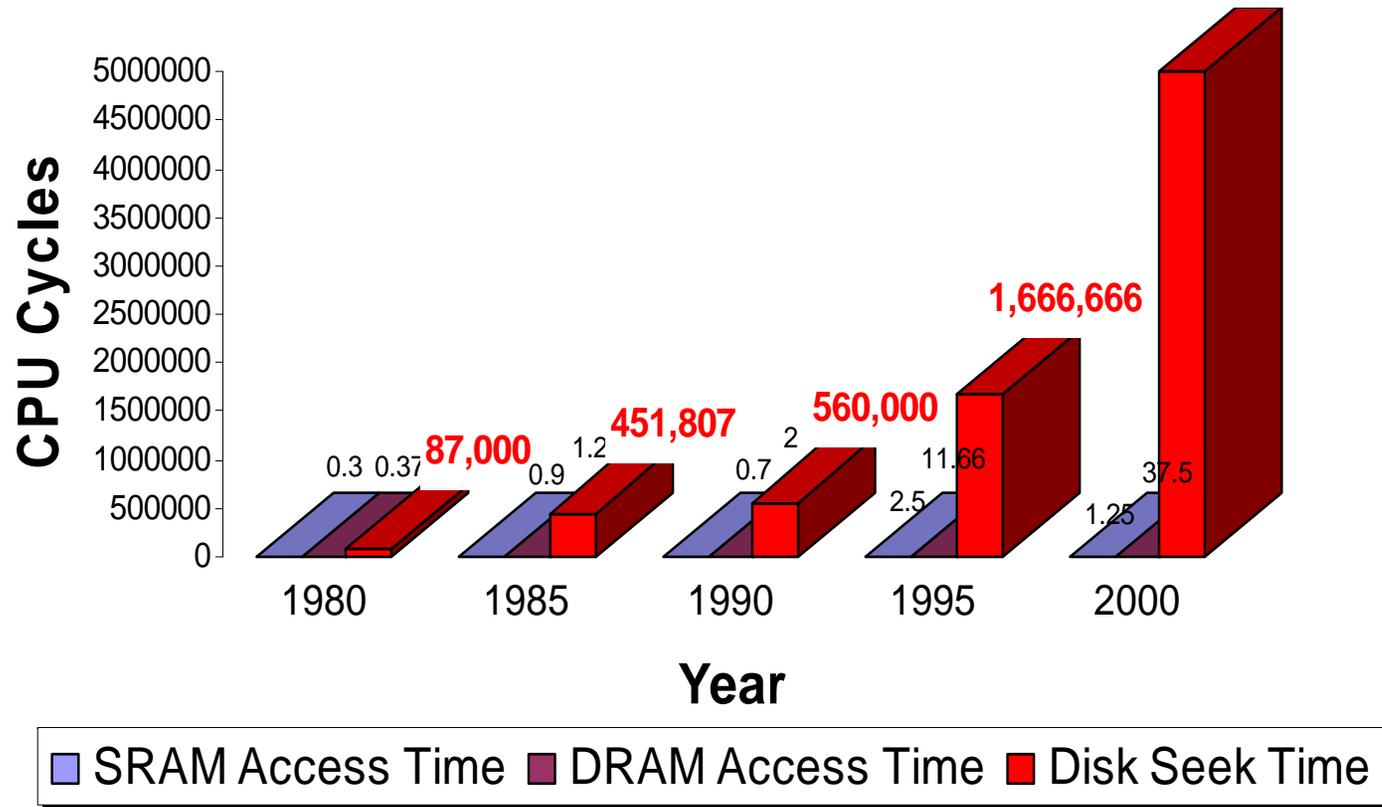
Many data-intensive applications generate huge data sets in disks world wide in very fast speed.

- LANL Turbulence Simulation: processing **100+ TB**.
- Google searches and accesses over **10 billion** web pages and **tens of TB data** in Internet.
- Internet traffic is expected to increase from **1 to 16 million TB/month** due to multimedia data.
- Stanford Linear Accelerator’s BaBar detector generates **1 TB/day** in PEP-II Storage Ring.
- We carry very large digital data, films, photos, ...

□ Data home is the cost-effective & reliable Disks

Slow disk data access is the major bottleneck

The disks in 2000 are 57 times “SLOWER” than their ancestors in 1980 --- increasingly widen the Speed Gap between Peta-Scale computing and Peta-Byte acesses.



Bryant and O'Hallaron, "Computer Systems: A Programmer's Perspective",
Prentice Hall, 2003

Principles of **Locality**

During an interval of execution, a set of data/instructions is repeatedly accessed (working set). (Denning, 70)

- **temporal locality**: data will be re-accessed timely.
- **spatial locality**: data stored nearby will be accessed.

□ Similar locality observations in many other areas:

- **Law of scattering**: significant papers hit in core journals.
- **Zipf's law**: frequently used words concentrate on 7%.
- **80-20 rule** for wealth distribution: 20% own 80% total asset.

□ Exploiting locality: identify/place data set in fast caches

- Large caches would never eliminate misses (Kung, 86)
- **What can we do after misses?**

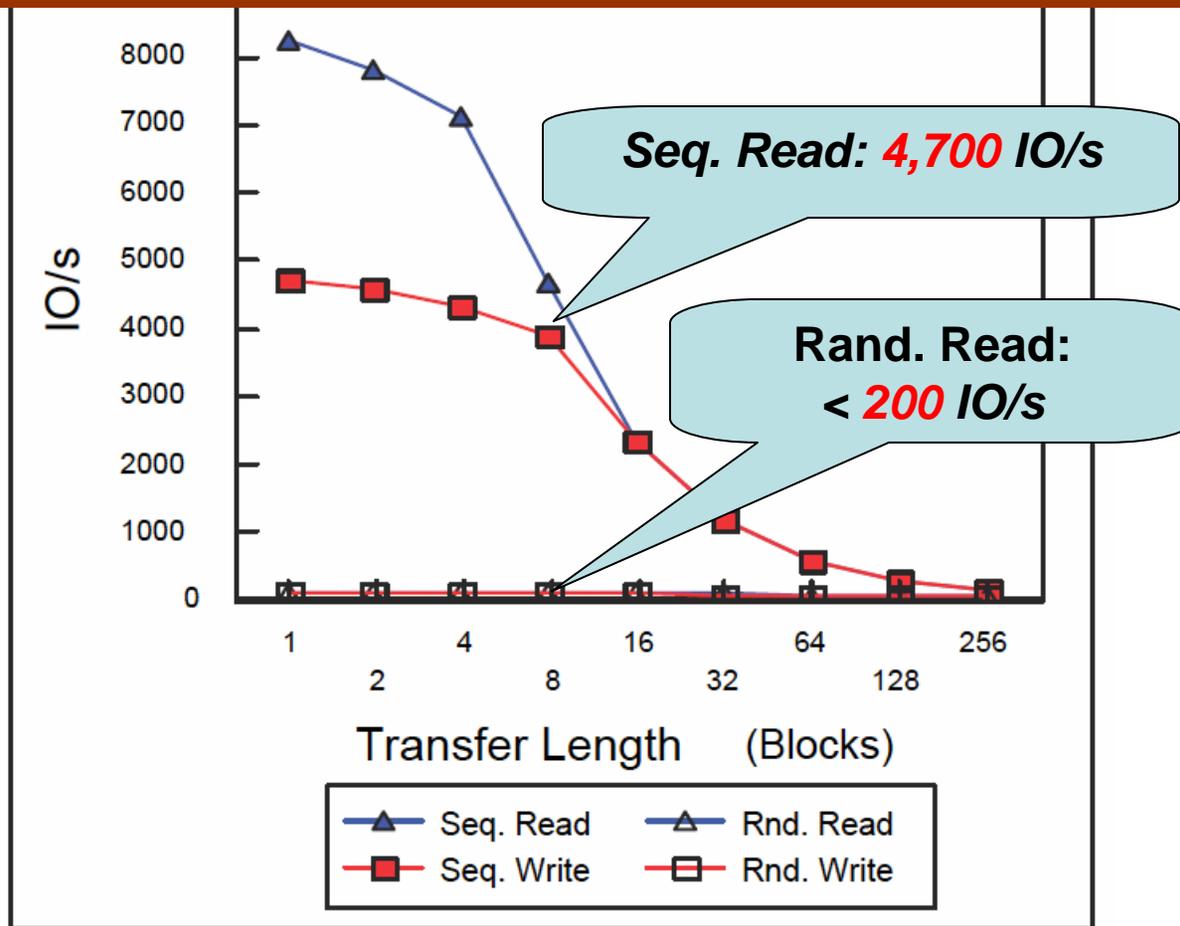
Sequential Locality in Disks has been Ignored

- ❑ **Sequential Locality:** data accesses in sequence very fast.
- ❑ Disk speed is limited by mechanical constraints.
 - seek/rotation** (high latency and power consumption)
- ❑ **Access of sequential blocks is fastest.**
- ❑ OS is not disk-layout aware
 - Little **sequential locality** info for caching and prefetching



IBM Ultrastar 18ZX Specification *

Our goal: to maximize opportunities of sequential accesses for high speed and high I/O throughput



. Ultrastar 18ZX with Write Cache on and Read Cache on

* Taken from IBM "ULTRASTAR 9LZX/18ZX Hardware/Functional Specification" Version 2.4

Existing Approaches and Limits

□ Programming for Disk Performance

- Hiding disk latency by overlapping computing
- Sorting large data sets (SIGMOD'97)
- Application dependent and programming burden

□ Transparent and Informed Prefetching (TIP)

- Applications issue hints on their future I/O patterns to guide prefetching/caching (SOSP'99)
- Not a general enough to cover all applications

□ Collective I/O: gather multiple I/O requests

- make contiguous disk accesses for parallel programs

Our Objectives

- **Exploiting sequential locality in disks**
 - by minimizing random disk accesses
 - making disk-aware caching and prefetching

- **Application independent approach**
 - putting disk access information on OS map
 - Exploiting DUal LOcalities (DULO):
 - Temporal locality of program execution
 - Sequential locality of disk accesses

Outline

- ❑ What is missing in buffer cache management?
- ❑ Managing disk layout information in OS
- ❑ **DULO-caching**
- ❑ **DULO-prefetching**
- ❑ Performance results
- ❑ Summary.

What is Buffer Cache Aware and Unaware?

❑ Buffer is an agent between I/O requests and disks.

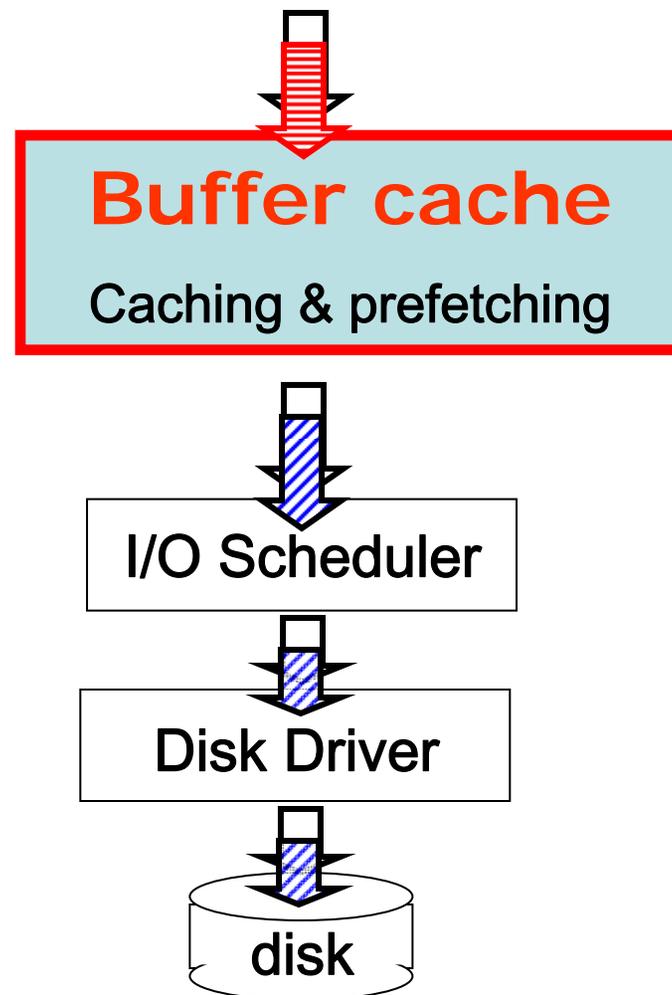
- aware access patterns in time sequence (in a **good position** to exploit **temporal locality**)
- unaware physical layout (**no effort** to exploit **sequential locality** in disks)

❑ Existing functions

- send unsatisfied requests to disks
- LRU replacement by temporal locality
- generate prefetching requests to disks.

❑ I/O scheduler is not effective: **sequential locality in disks is not open to buffer cache.**

Application I/O Requests



Limits of Buffer Caching Unaware of Disk Layout

- Minimizing cache miss ratio by only exploiting temporal locality

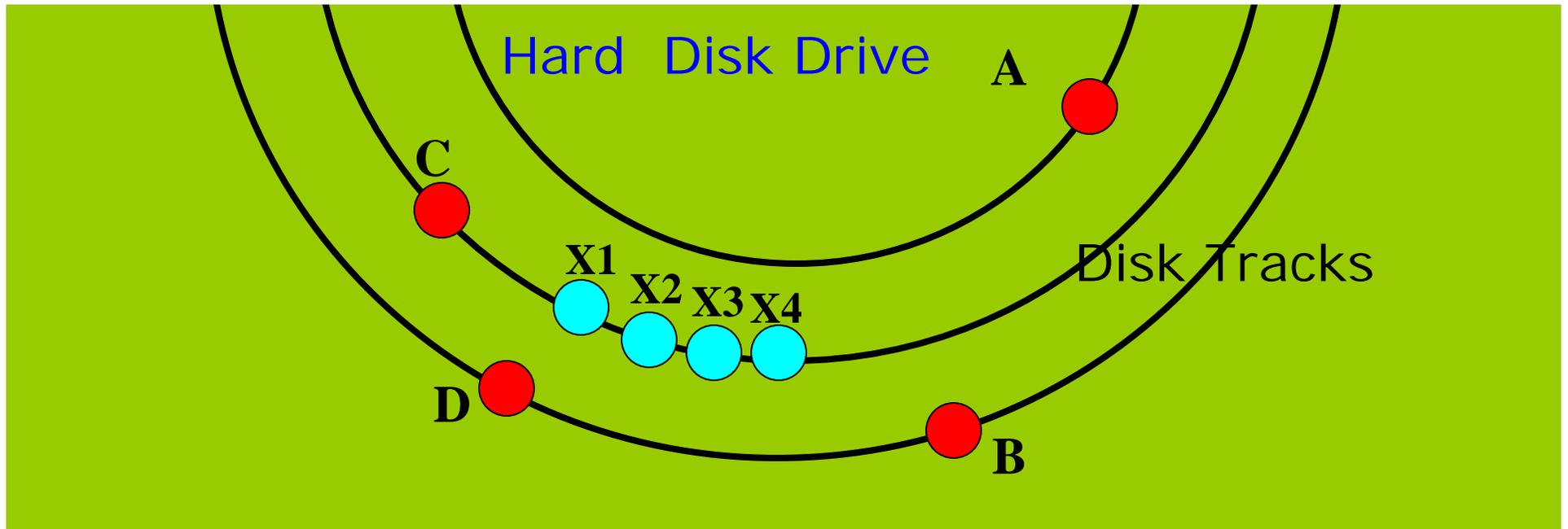
Average access time

$$= \underbrace{\text{Hit rate} \times \text{Hit time}}_{\text{Temporal locality}} + \underbrace{\text{Miss rate} \times \text{Miss penalty}}_{\text{Sequential locality}}$$

Temporal
locality

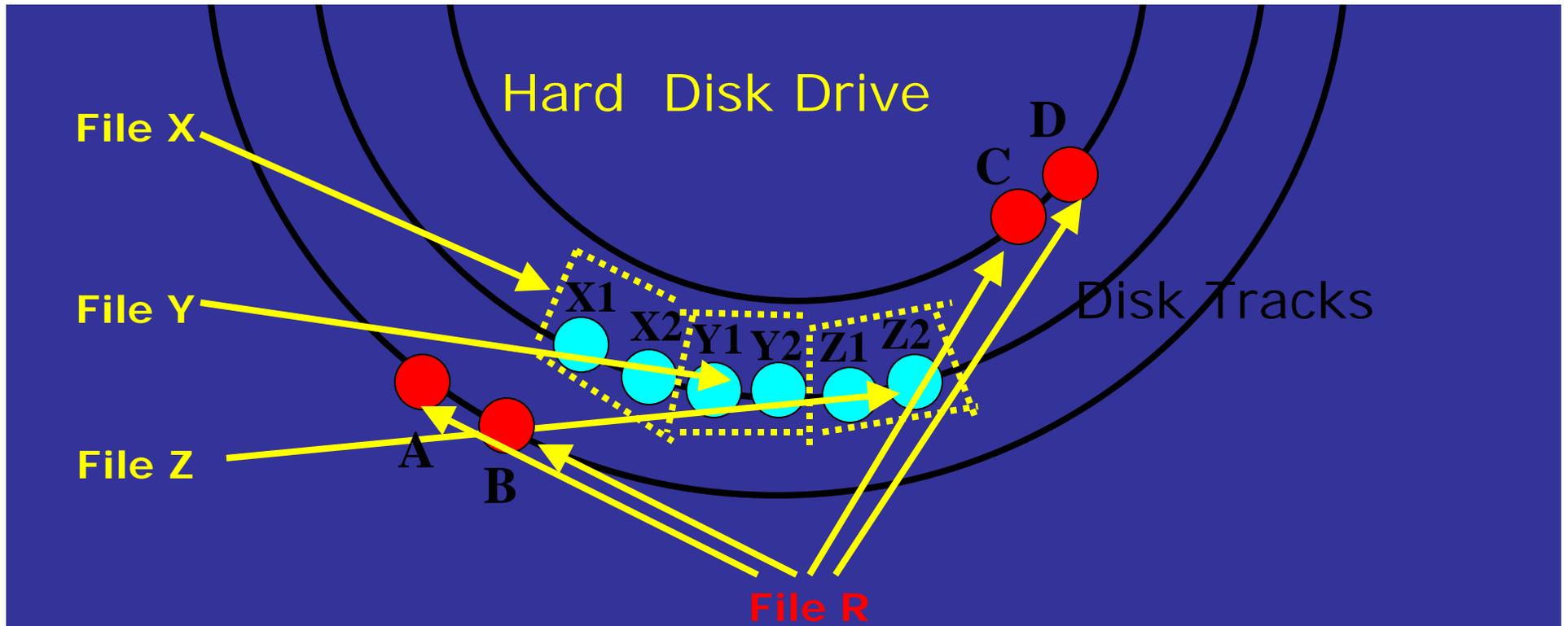
Sequential
locality

- **Sequentially** accessed blocks → **small** miss penalty
- **Randomly** accessed blocks → **large** miss penalty



❑ Unique and critical roles of buffer cache

- ❑ Buffer cache can influence request stream patterns in disks
- ❑ If buffer cache is disk-layout-aware, OS is able to
 - Distinguish sequentially and randomly accessed blocks
 - Give “expensive” random blocks a high caching priority
 - replace long sequential data blocks timely to disks
 - Disk accesses become more sequential.

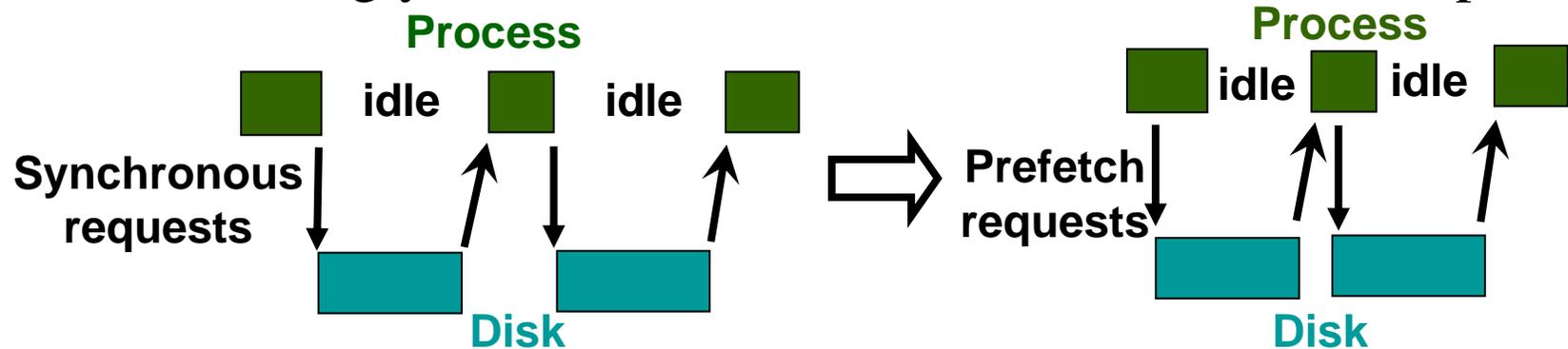


❑ Performance is low by logic file abstraction based prefetch.

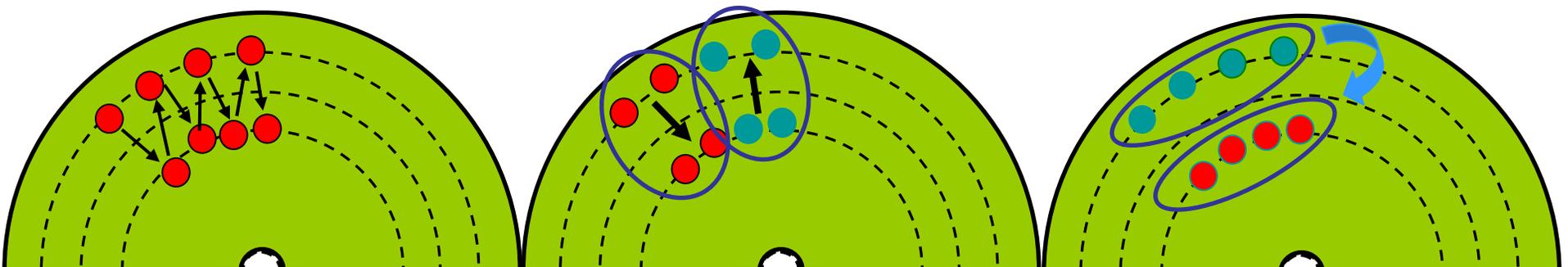
- Sequential accesses spanning multiple small files cannot be detected.
- Sequential layout in logic abstraction may not be sequential layout on physical disk.
- Detected sequences of blocks are not remembered.

Disk Layout Affects Prefetching Efficiency

It is increasingly difficult to hide disk accesses behind computation

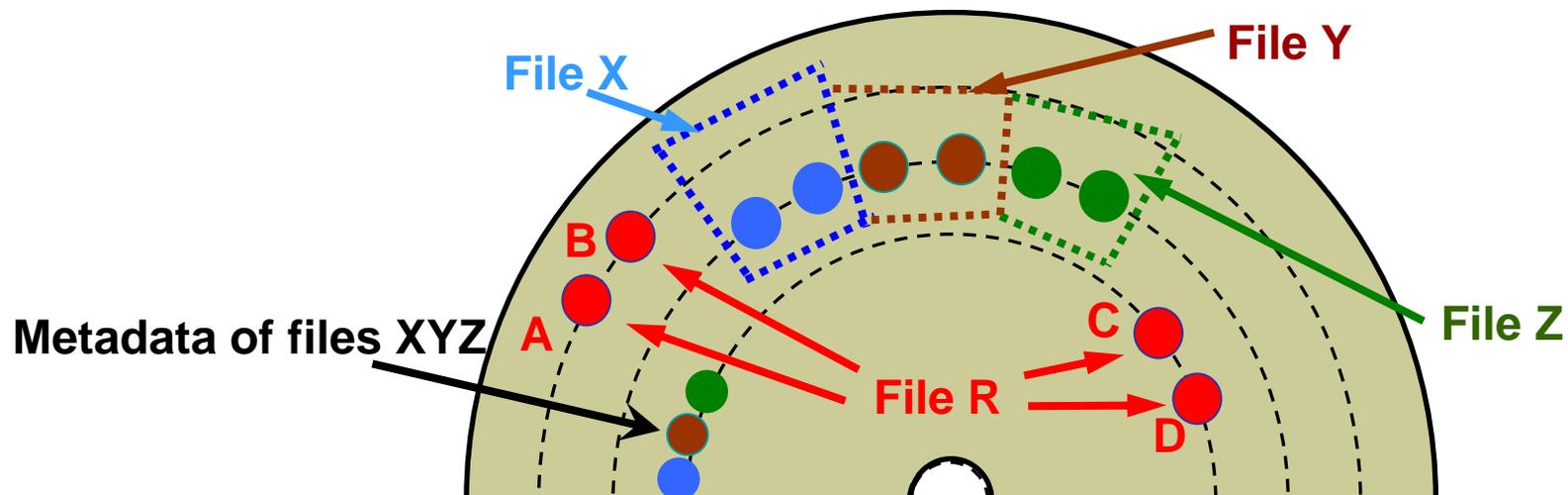


- Prefetching may incur non-sequential disk access
 - Non-sequential accesses are 100+ times slower than sequential ones
 - Disk layout information must be introduced into prefetching policies.



File-level Prefetching is Disk Layout Unaware

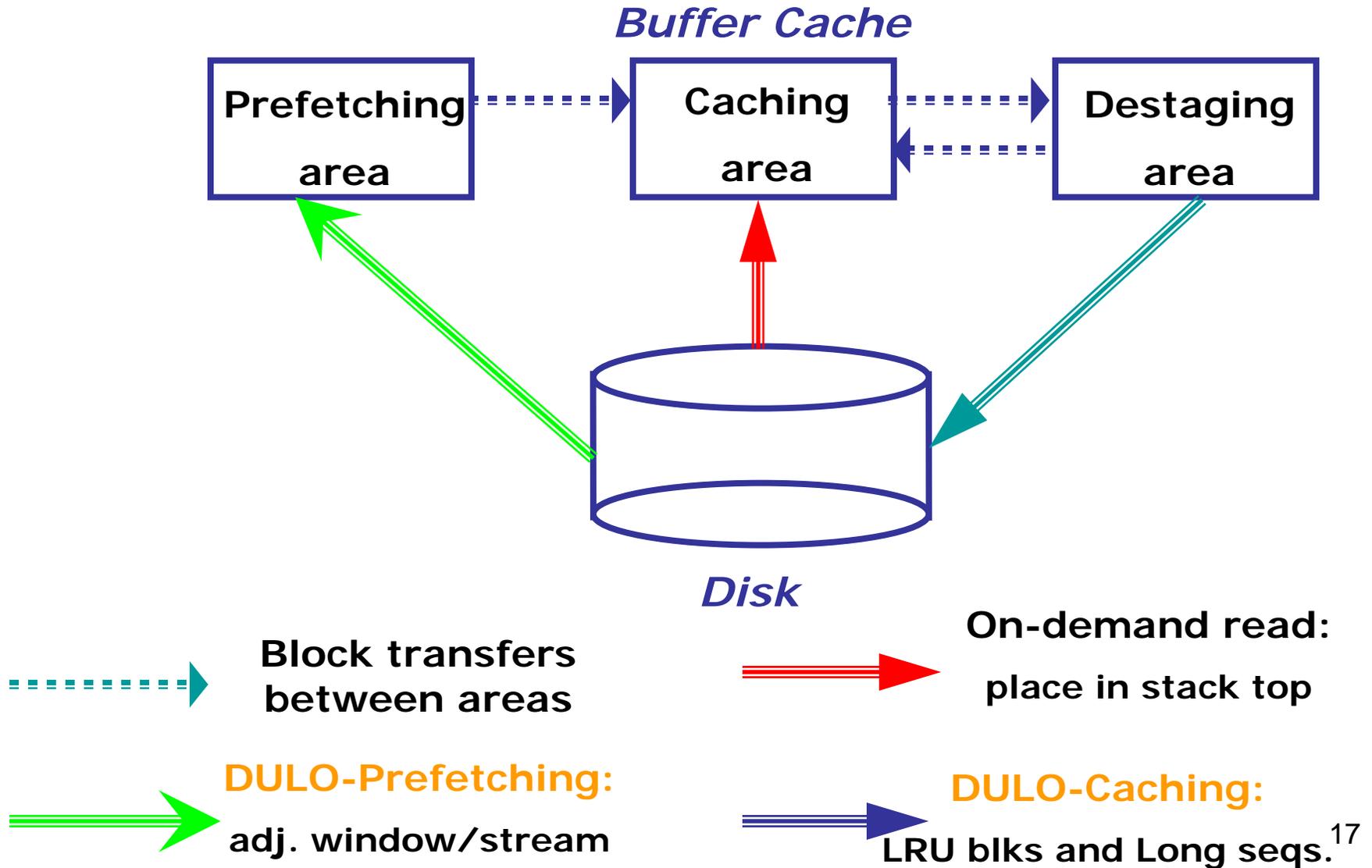
- Multiple files sequentially allocated on disks cannot be prefetched at once.
- Metadata are allocated separately on disks, and cannot be prefetched
- Sequentiality at file abstraction may not translate to sequentiality on physical disk.
- Deep access history information is usually not recorded.



Opportunities and Challenges

- ❑ Opening Disk Sequential Locality (DiskSeen)
 - ❑ Buffer cache exploits Dual Localities (DULO)
 - ❑ Address the limits of both caching and prefetching
- ❑ A DiskSeen System Infrastructure
 - ❑ analyze and utilize disk-layout Information
 - ❑ accurately and timely identify long disk sequences
 - ❑ consider trade-offs of temporal and sequential locality (buffer cache hit ratio vs miss penalty)
 - ❑ manage its data structures with low overhead
 - ❑ Implement it in OS kernel for practical usage

***DiskSeen*: a System Infrastructure to Support DULO-Caching and DULO-Prefetching**



Conclusions

- ❑ **Disk performance is limited by**
 - ❑ Non-uniform accesses: fast sequential, slow random
 - ❑ OS buffer management is Disk-layout unaware.
- ❑ **The buffer cache is a critical component for I/O.**
 - ❑ temporal locality of program execution is used only.
- ❑ **Building a Disk-Seen system infrastructure for**
 - ❑ DULO-Caching
 - ❑ DULO-Prefetching
- ❑ **Enhance the functionality of OS management in storage.**

References

- ❑ **DULO-caching**: a prototype and its results, FAST'05.
- ❑ **SmartSaver**: Saving disk energy by Flash M, ISLPED'06
- ❑ **LAC**: Cooperative buffer caching in clusters, ICDCS'06.
- ❑ **Streaming locality** patterns in Interent, SIGMETRICS'07.
- ❑ **STEP**: caching/prefetch in networked storage sys., ICDCS'07
- ❑ **DULO-prefetching**: OS kernel enhancement, USENIX'07.