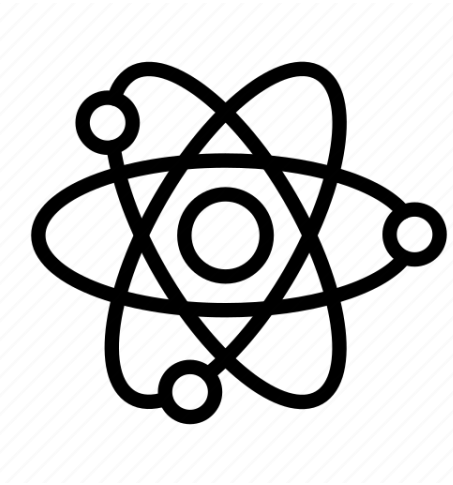
# CAVE: Concurrency-Aware Graph Processing System for SSD

Tarikul Islam Papon  Taishan Chen  Shuo Zhang  Manos Athanassoulis

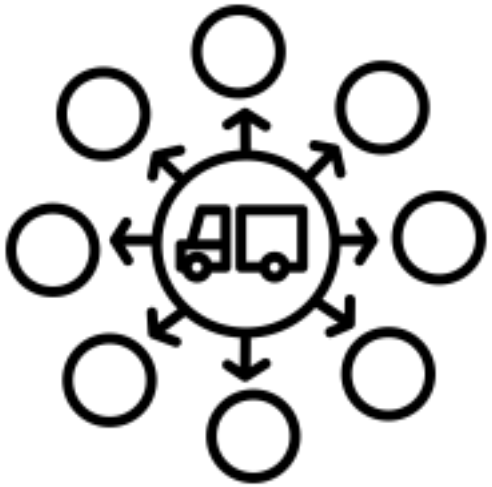
# Rise of Large Graphs

Graphs are everywhere!

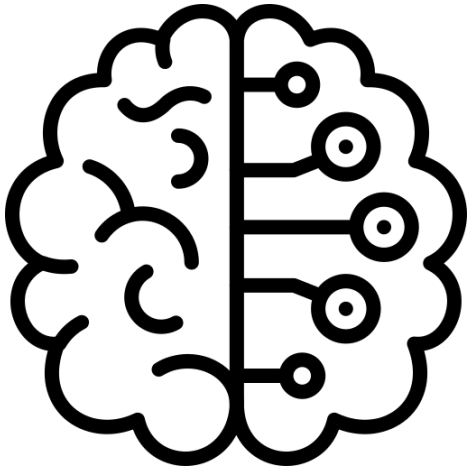
Social Network



Physical Science



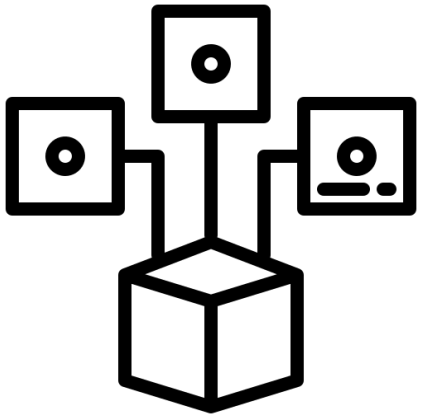
Transportation Network



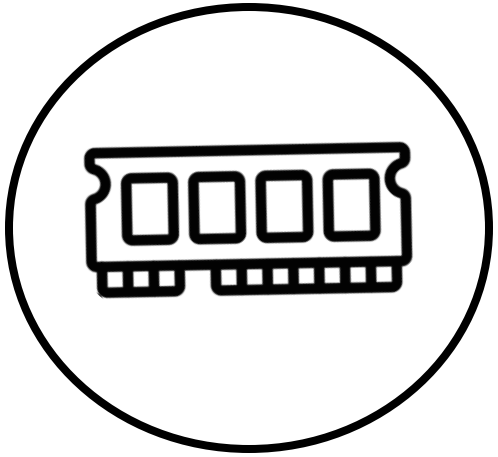
Machine Learning

Real-world graphs often have more than a billion nodes

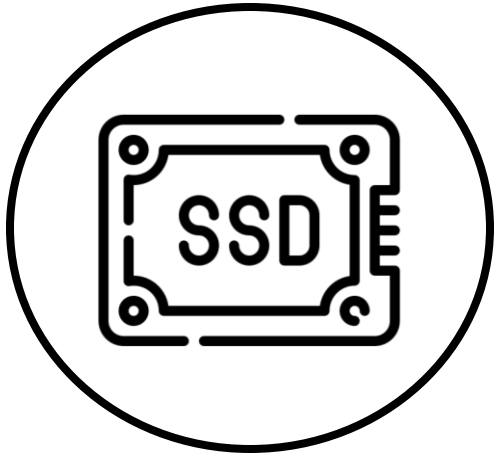
# Processing Large Graphs



Distributed Systems

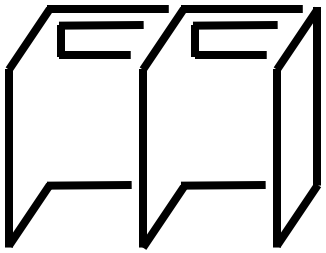


Single-node
in-memory systems

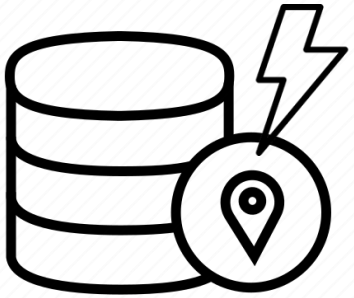



**Single-node**
**out-of-core systems**

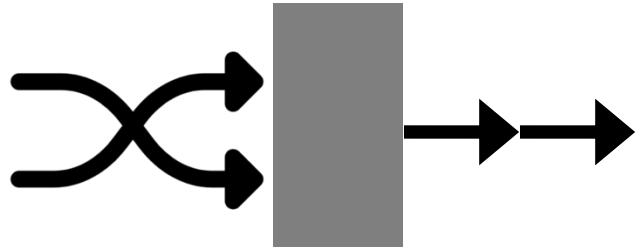
# SOA Out of Core Systems



Data partitioning



Improve memory
& disk locality

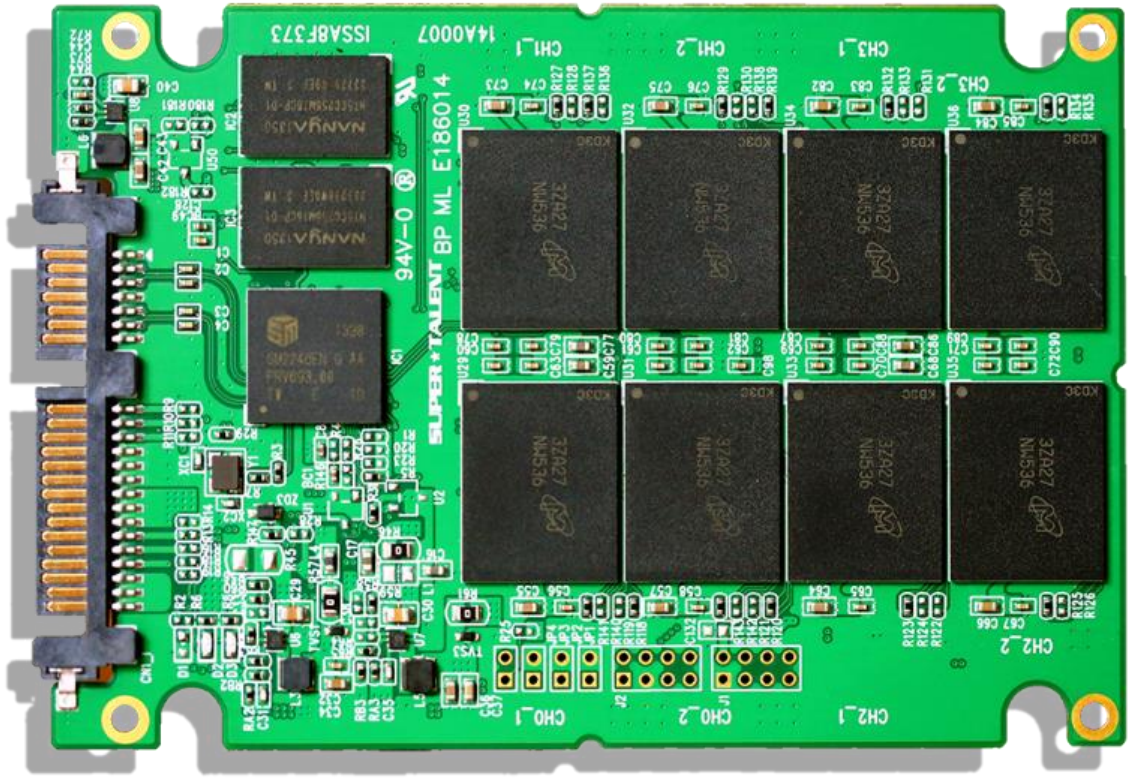


Reduce random I/O

**Designed for HDDs**

**“Tape is Dead. Disk is Tape. Flash is Disk.” - Jim Gray**

# Solid State Drives



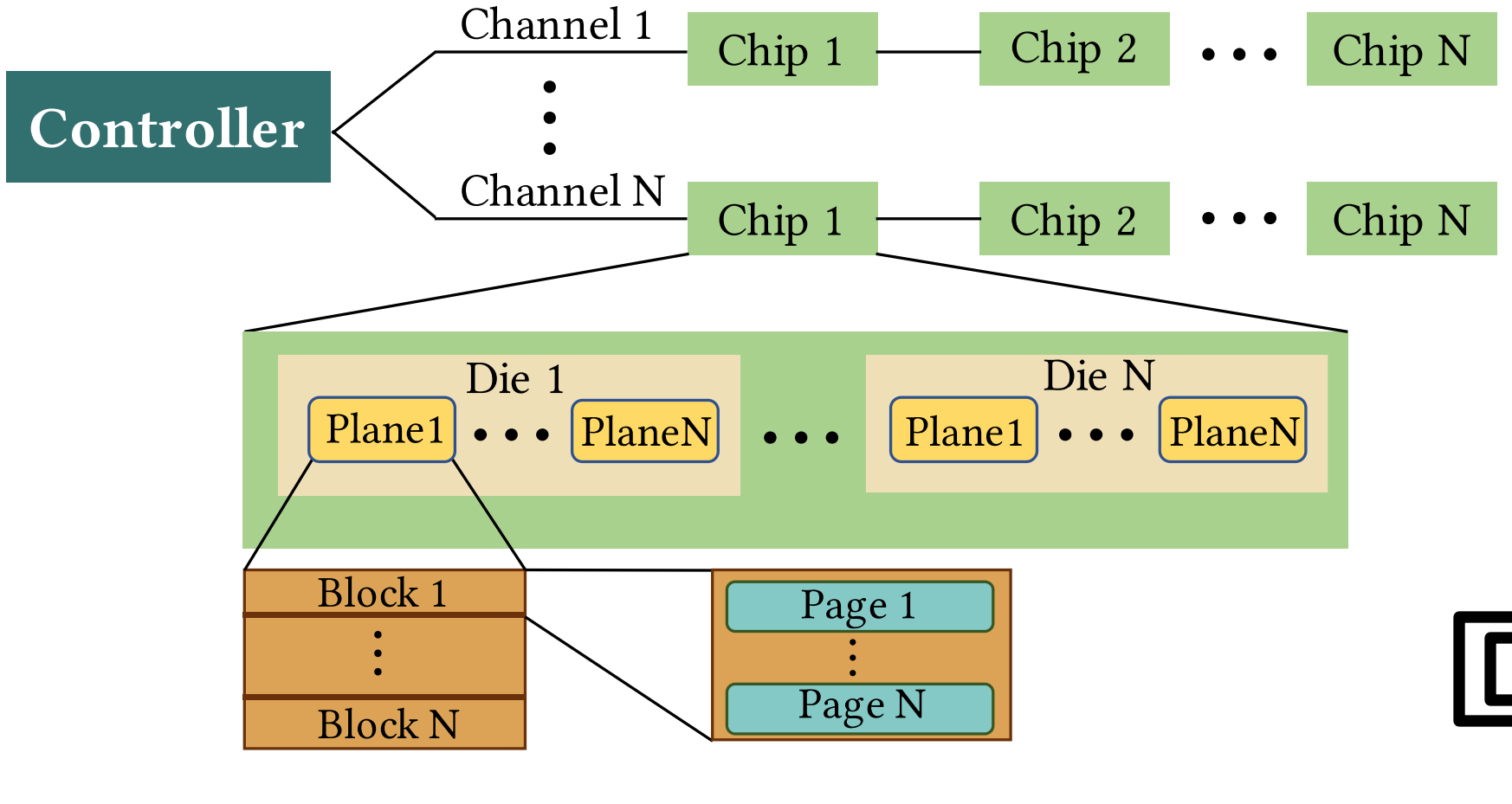

electronic device

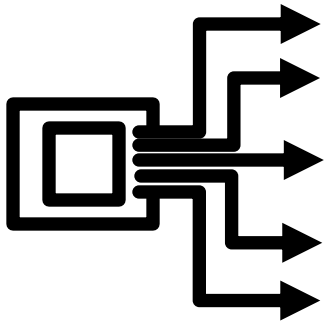
**fast random access**

write latency > read latency

**concurrent I/Os**

# SSD Concurrency



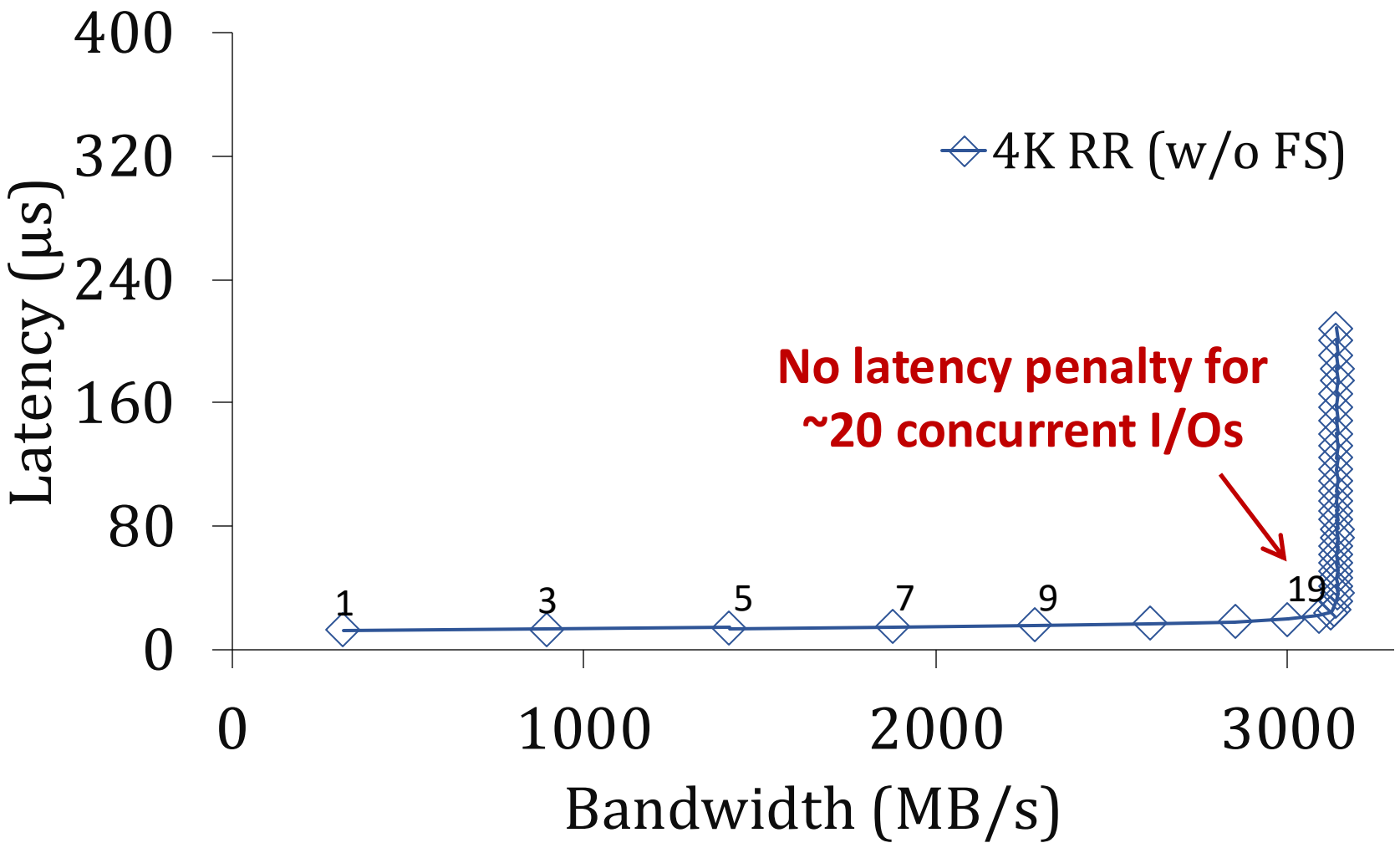
**Parallelism** at different levels

# Impact of Concurrency

**Device**

PCIe SSD - P4510 (1TB)

***Optimal* read concurrency ($k_r$) = 20**



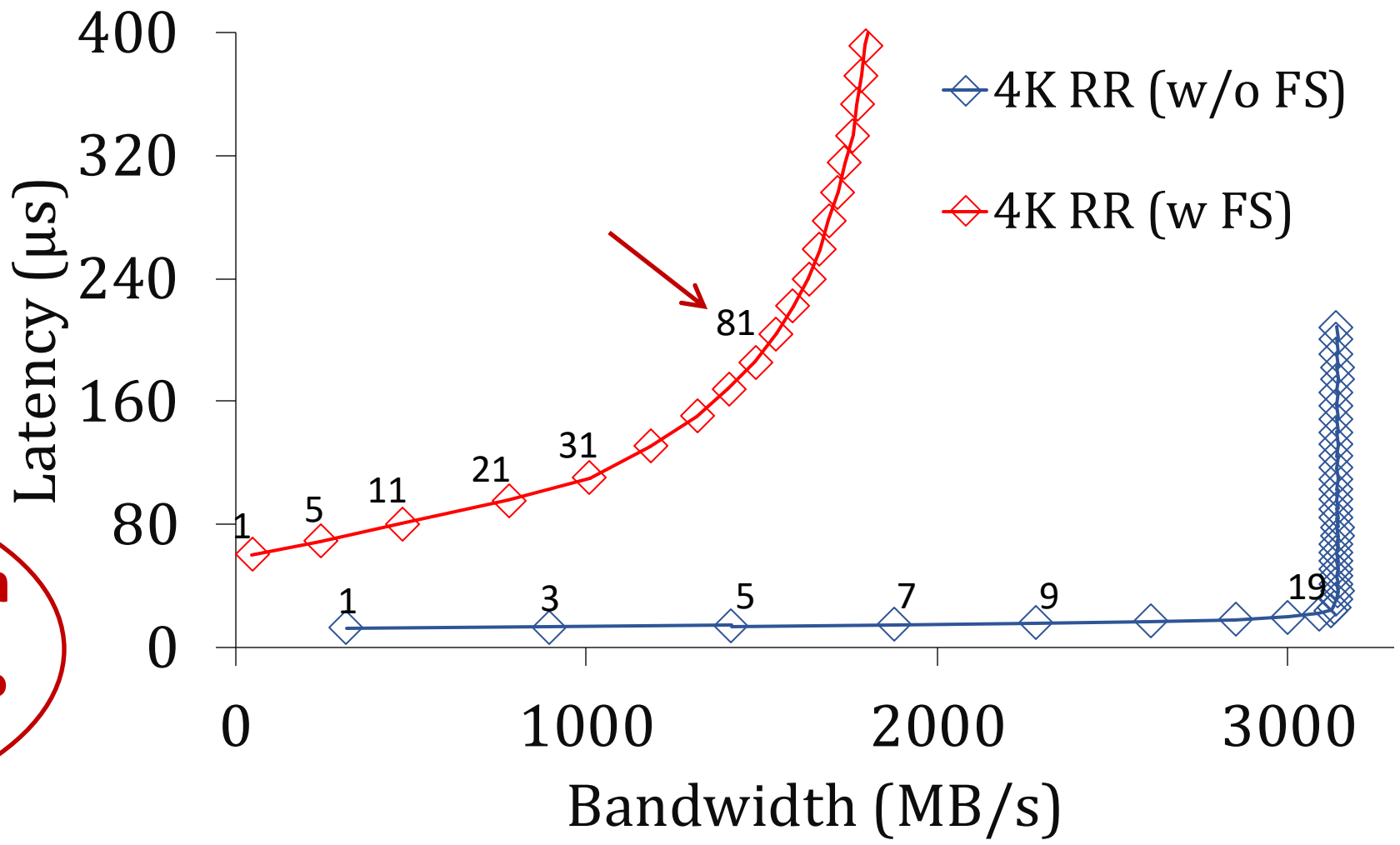
# Impact of Concurrency

**Device**

PCIe SSD - P4510 (1TB)

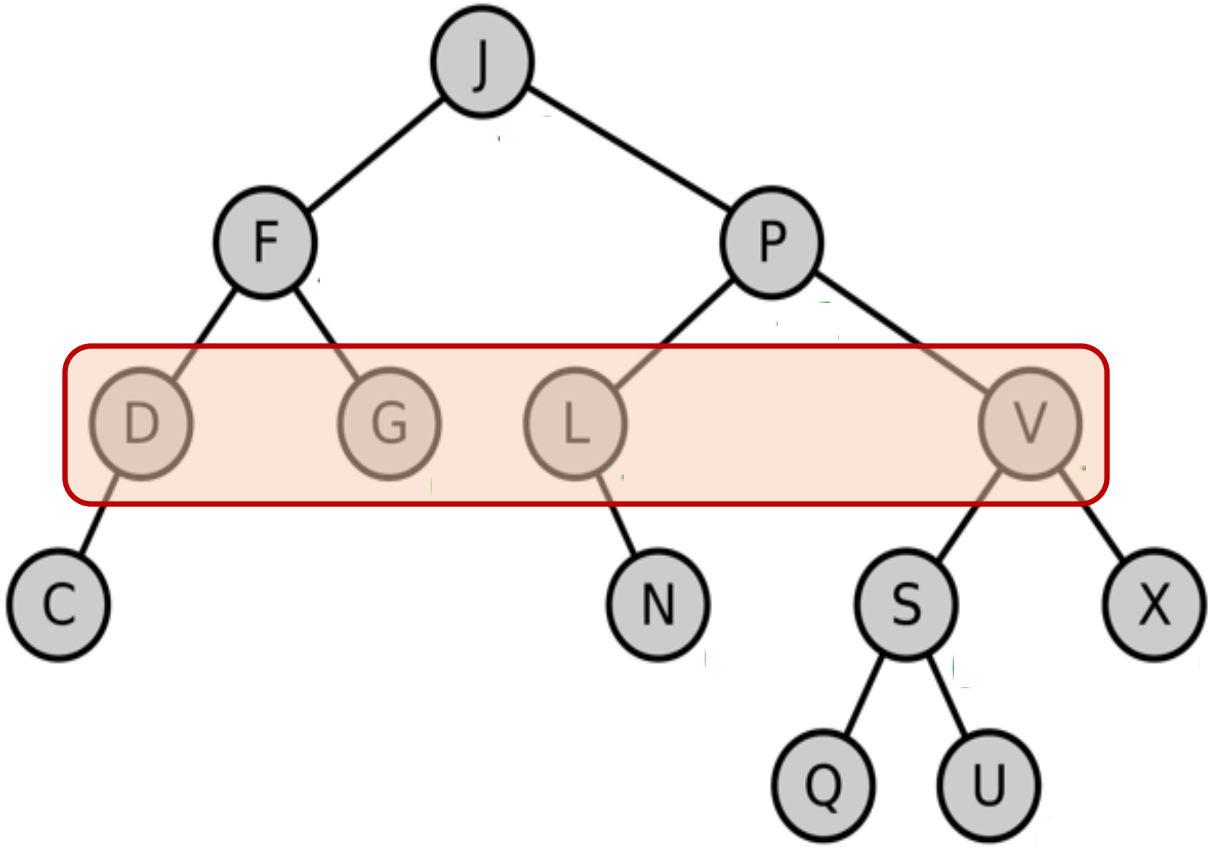
***Optimal* read concurrency ($k_r$) = 80**

**Can SSD-based Graph Systems Exploit This?**

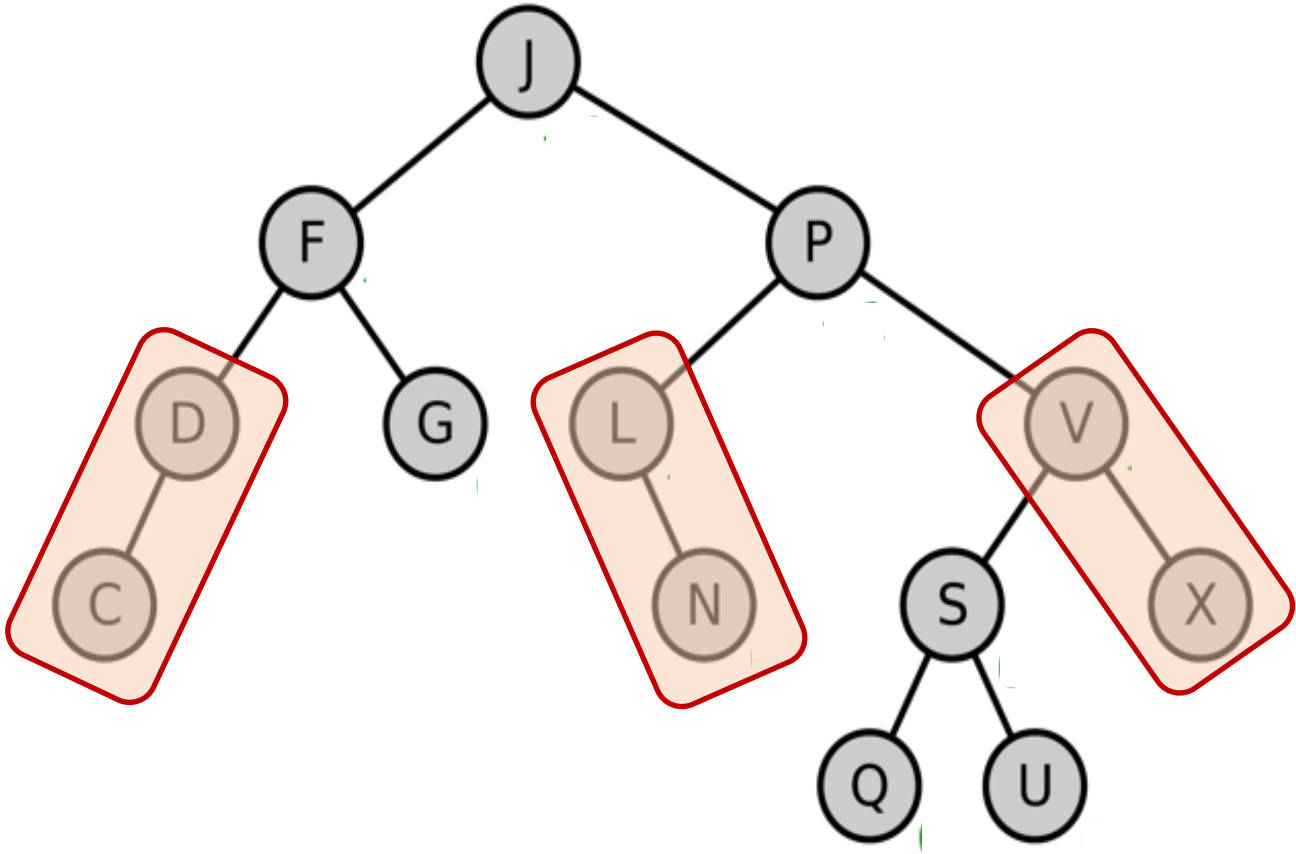


# Parallelizing Graph Traversal

**Intra-Subgraph Parallelization**

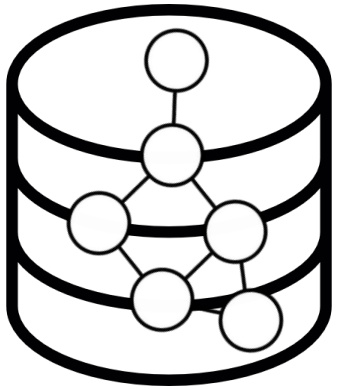



**Inter-Subgraph Parallelization**



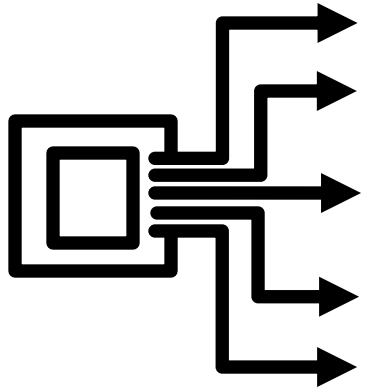

**process in parallel up to $k_r$ nodes/subgraphs**

# Our Goal

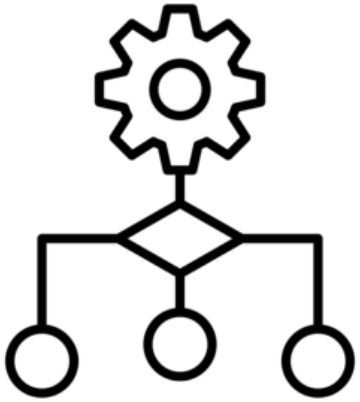


Optimize for **storage-based** graph workloads

Focus on **traversal** operations



Utilize **SSD Concurrency**

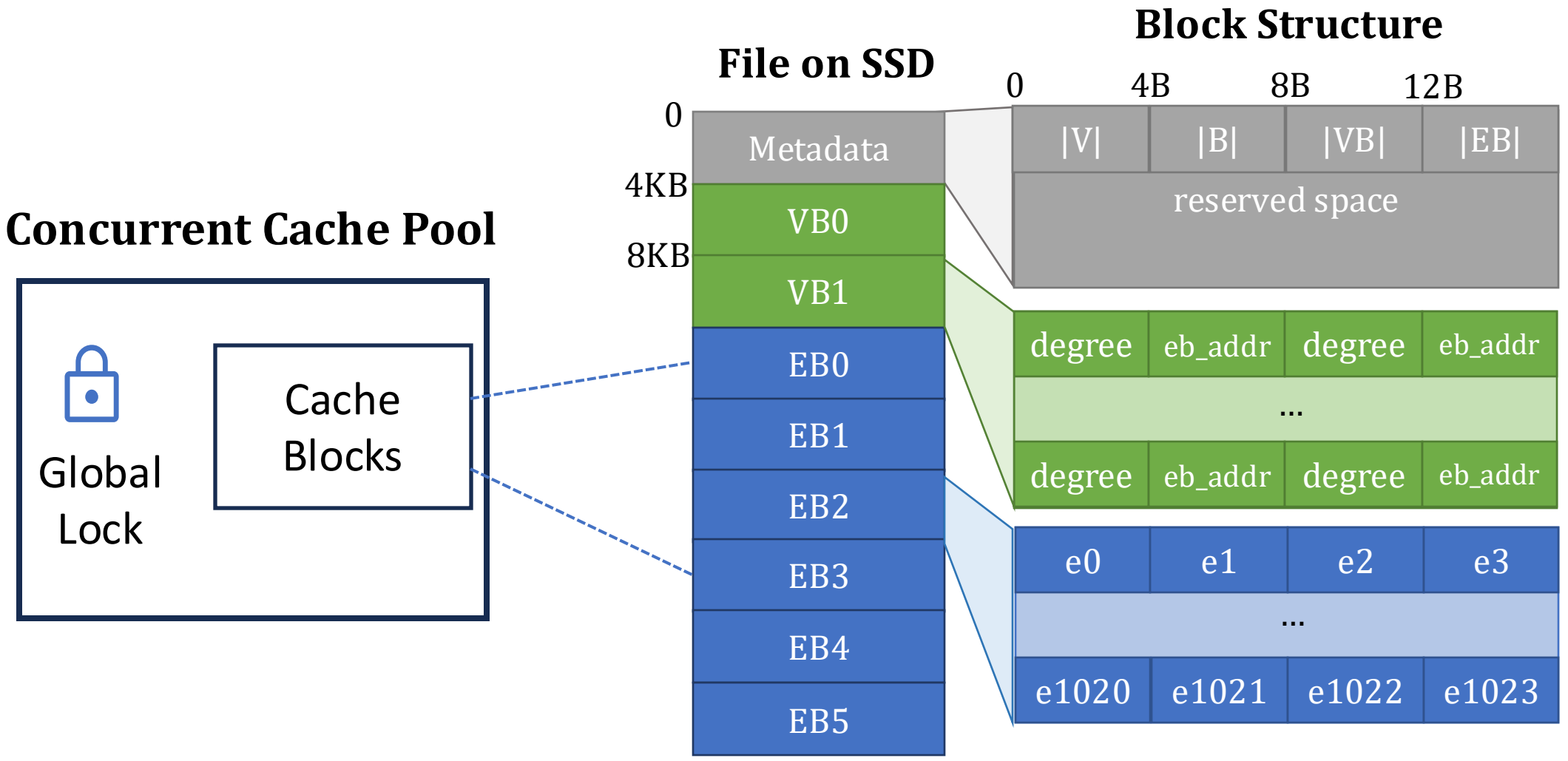


Maintain core **algorithm properties**

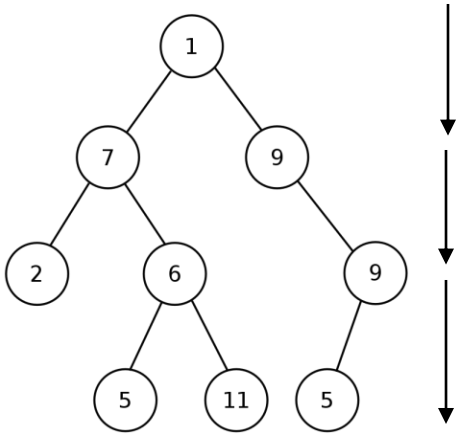
**Concurrency-Aware Graph (V, E) Manager**

**CAVE**

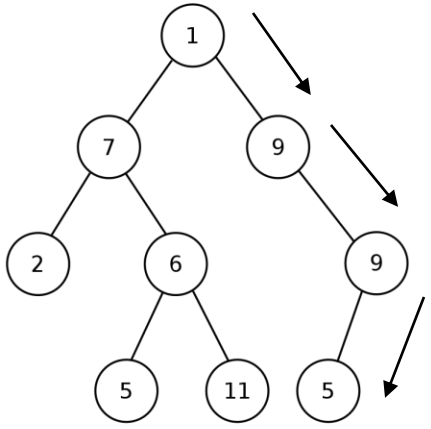
# CAVE Architecture



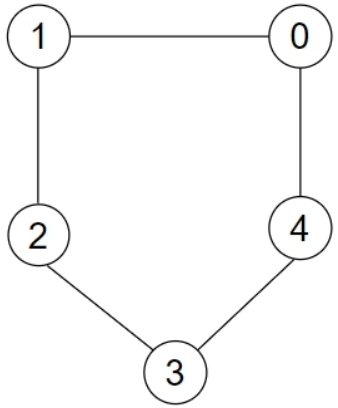
# Concurrent Graph Algorithms



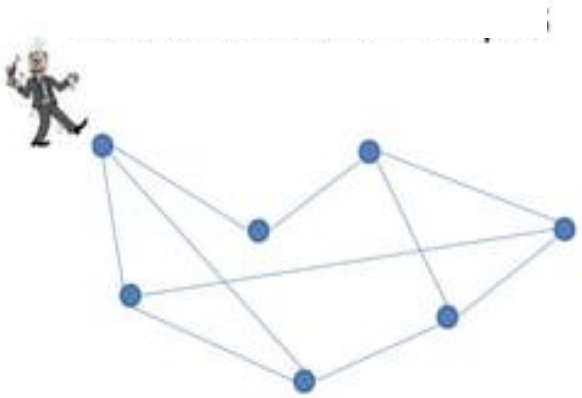

Parallel BFS



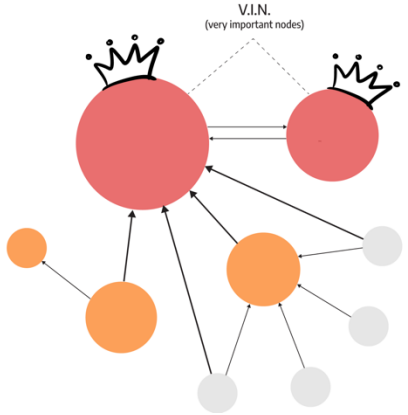

Parallel DFS




Parallel WCC



Parallel Random Walk



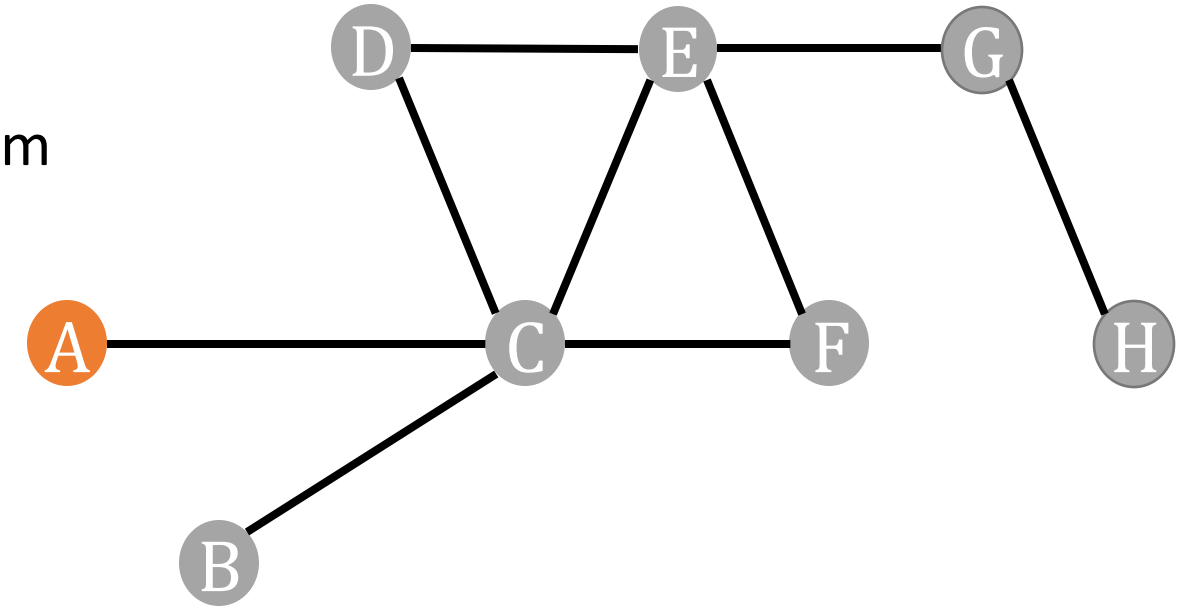

Parallel PageRank

# Parallel BFS

processed nodes  processing in progress  yet to be processed

Each iteration involves

1. processing $k_r$ vertices concurrently from a list of vertices (**frontier)**
2. accessing neighbors of each vertex
3. updating vertex values
4. determining next frontier

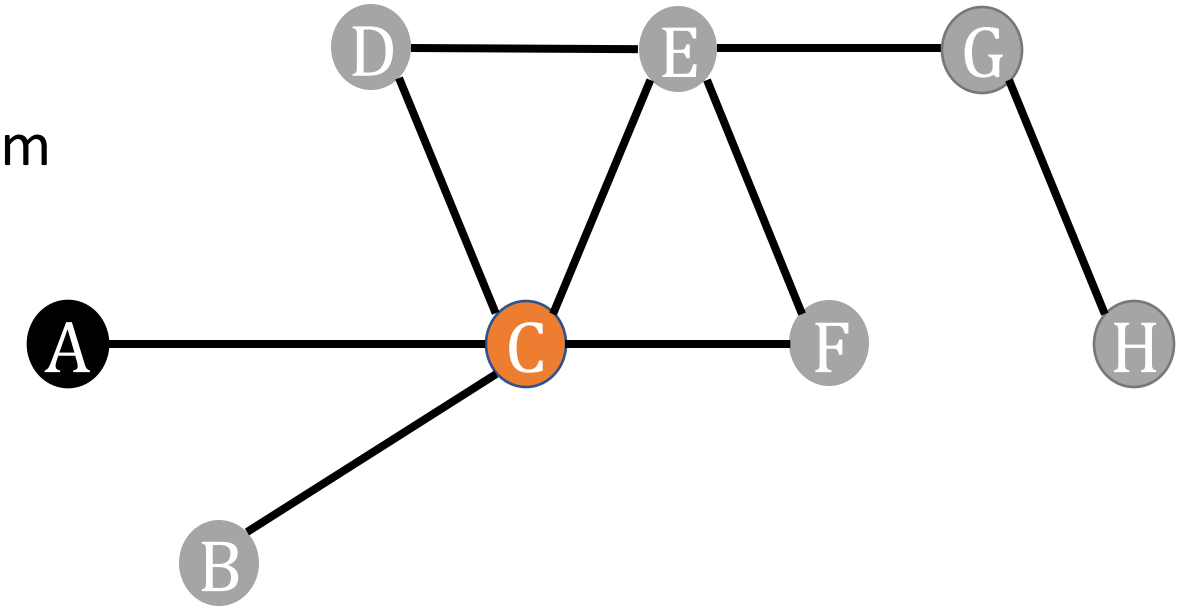


# Parallel BFS

processed nodes  processing in progress  yet to be processed

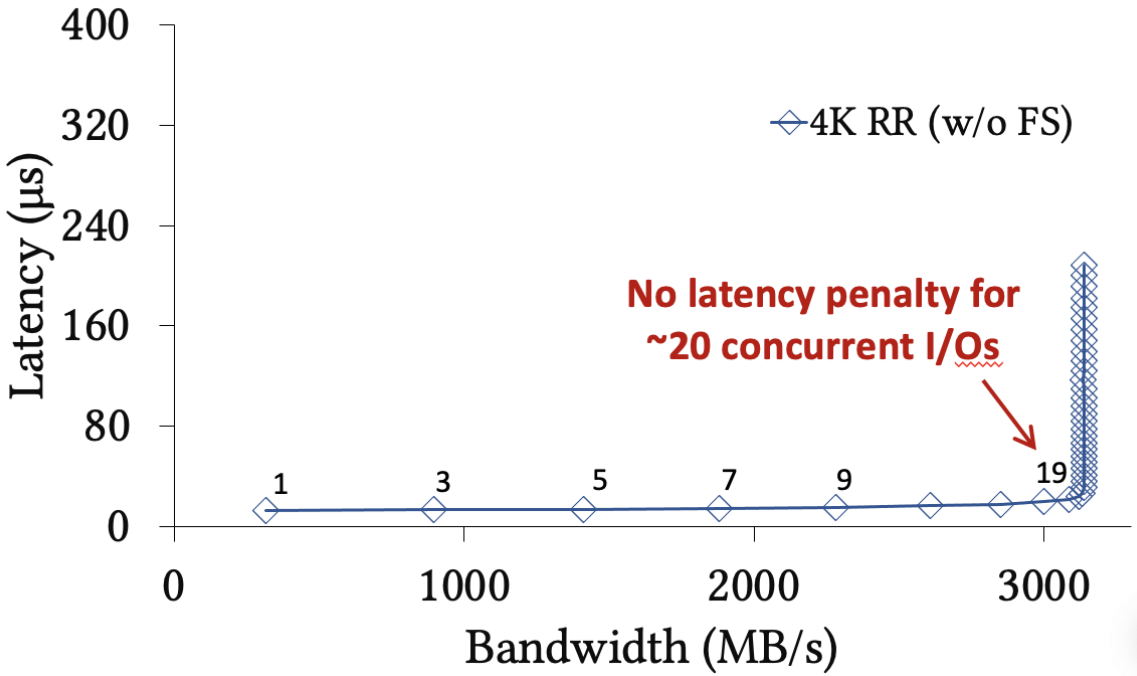
Each iteration involves

1. processing $k_r$ vertices concurrently from a list of vertices (**frontier)**
2. accessing neighbors of each vertex
3. updating vertex values
4. determining next frontier



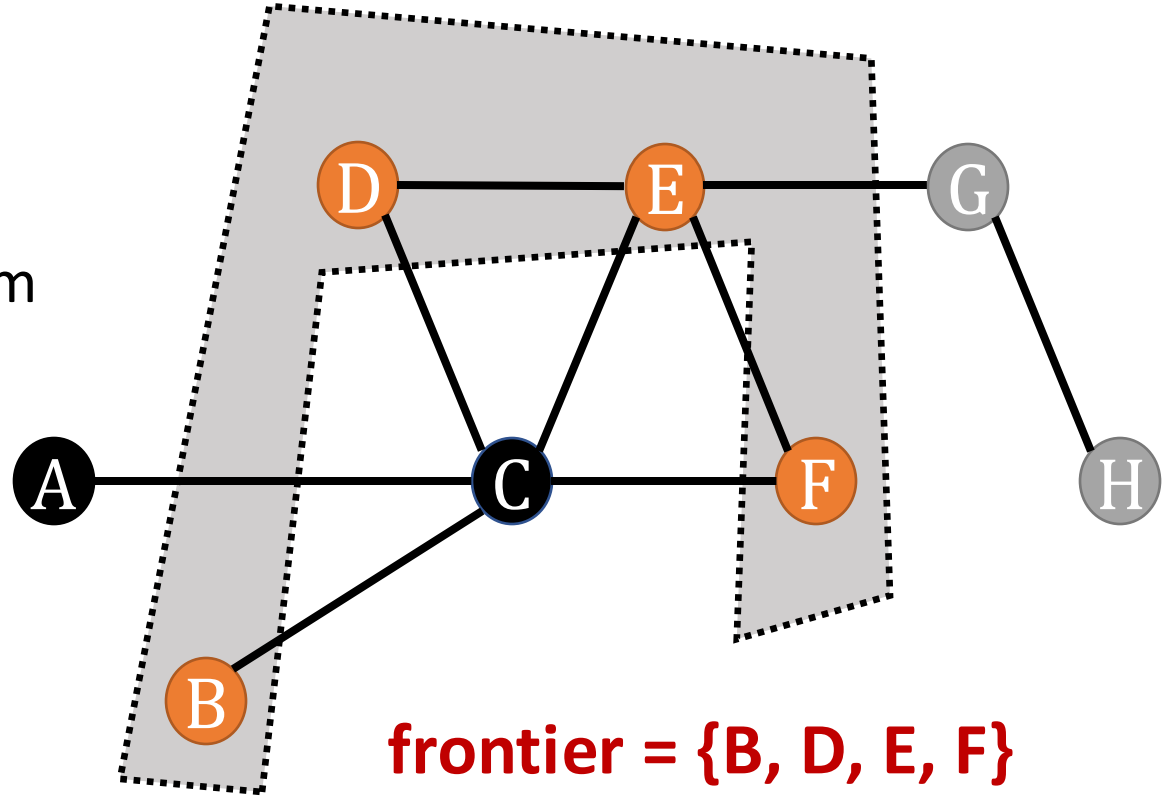
# Parallel BFS

processed nodes  processing in progress  yet to be processed




4. determining next frontier

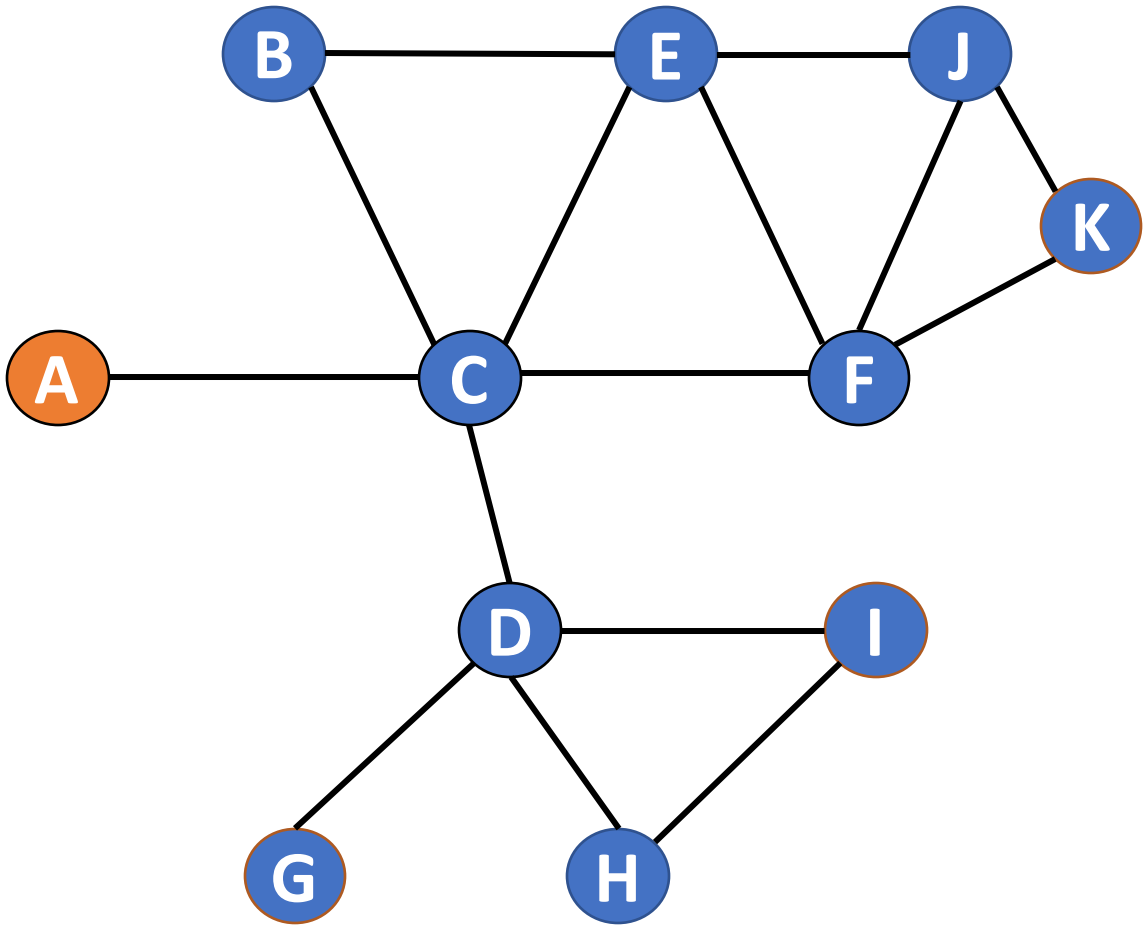
m




frontier = {B, D, E, F}

# Parallel pseudo DFS

processed nodes  processing in progress  yet to be processed



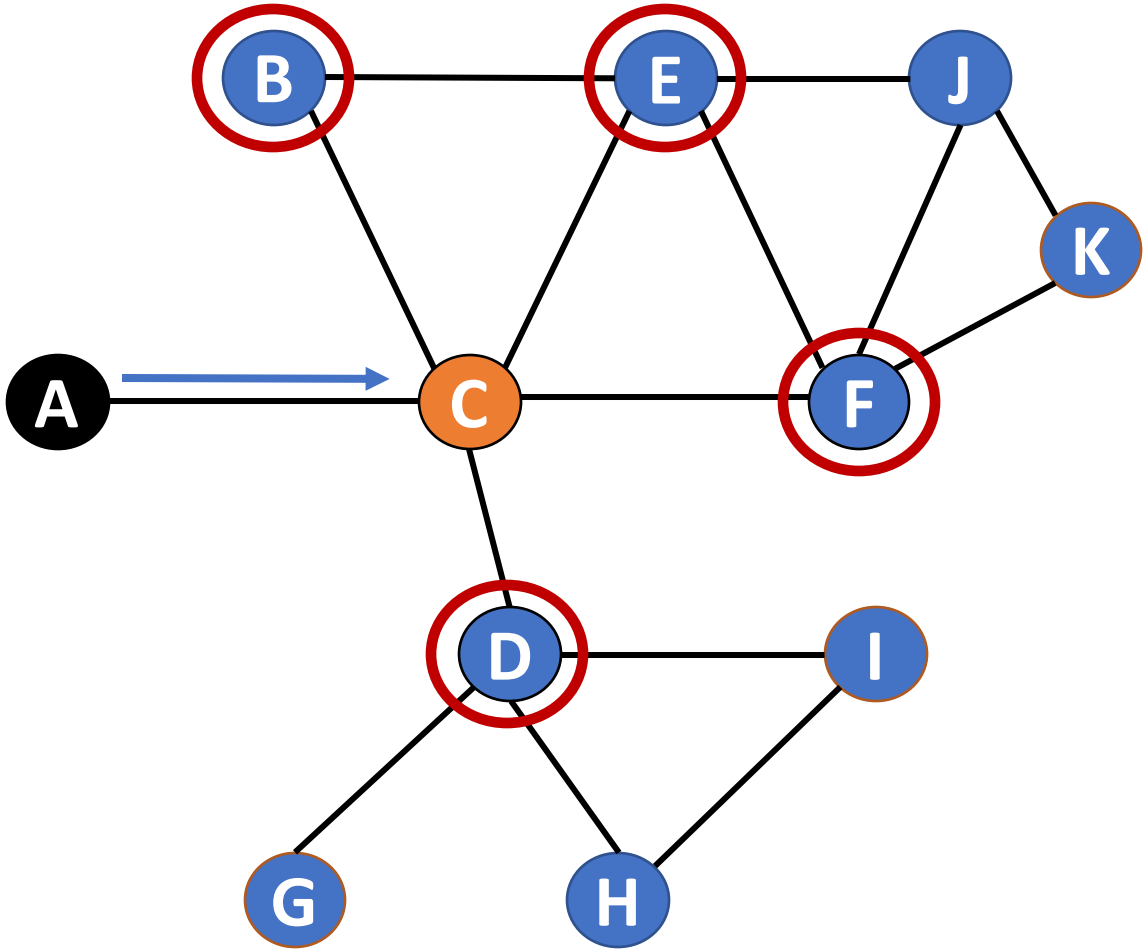

**Time**

**1**

A **Thread #1**

# Parallel pseudo DFS

processed nodes  processing in progress  yet to be processed



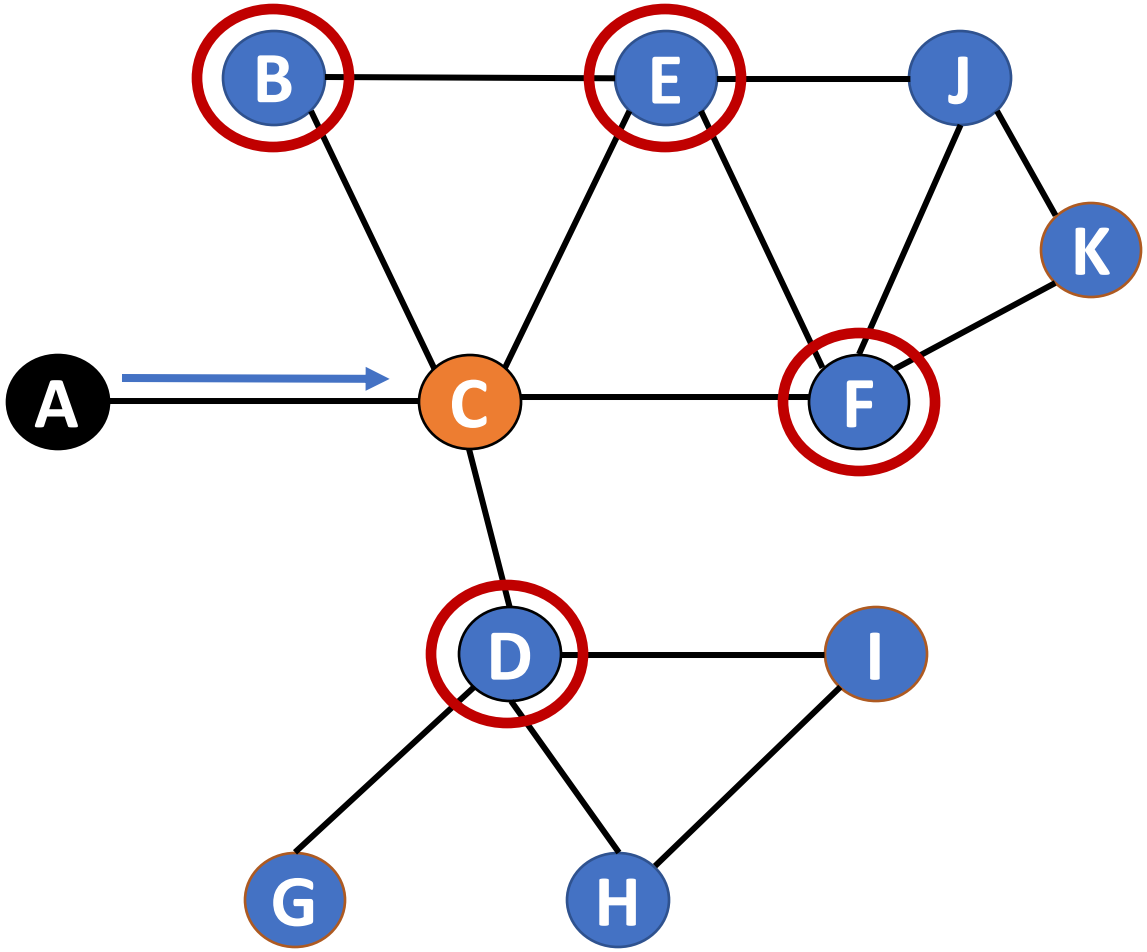

**Time**

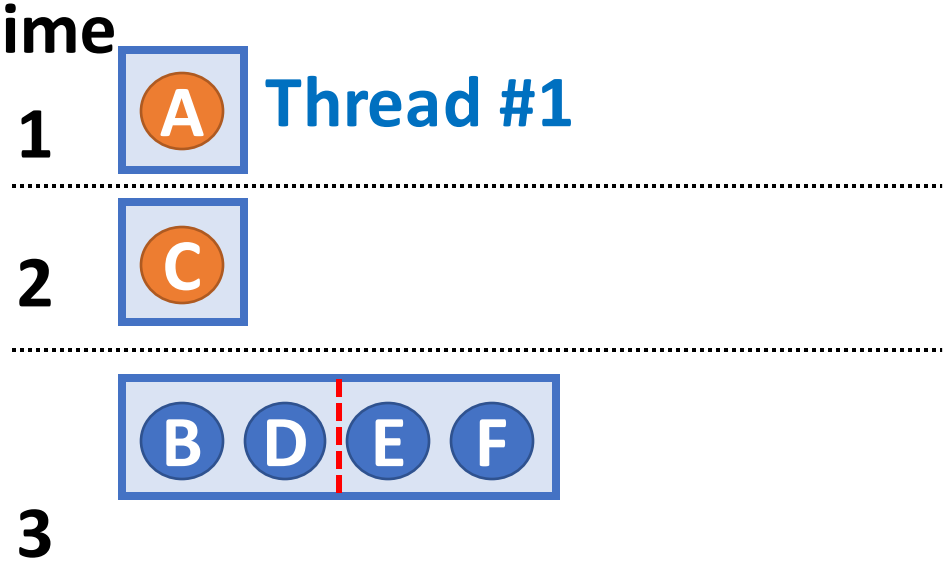
**1** A **Thread #1**

**2** C

# Parallel pseudo DFS

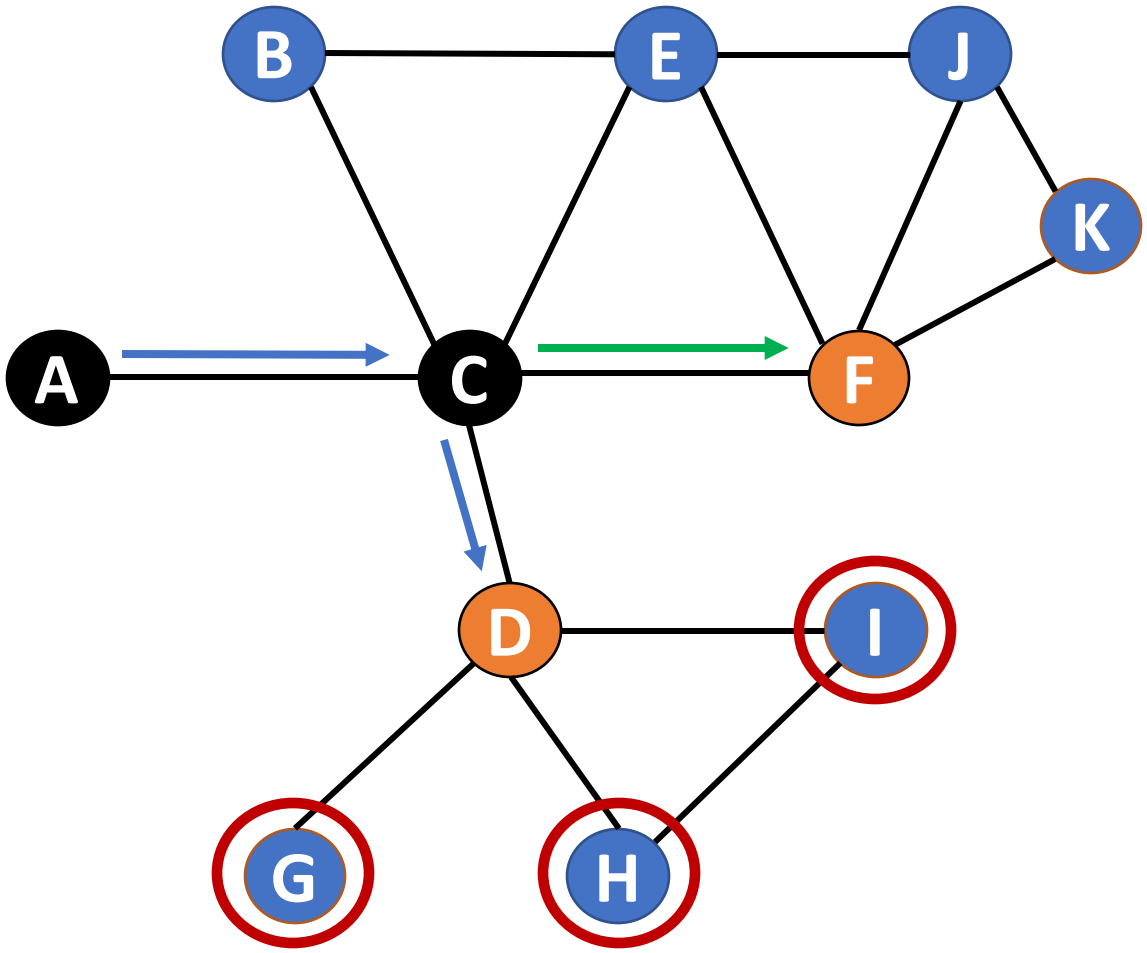
processed nodes  processing in progress  yet to be processed

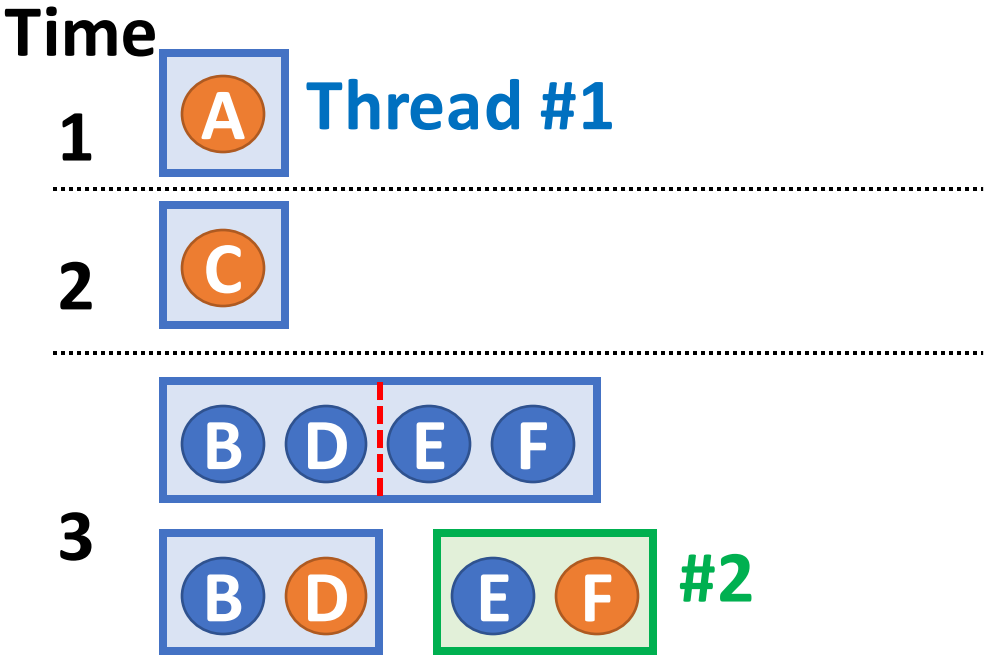


# Parallel pseudo DFS

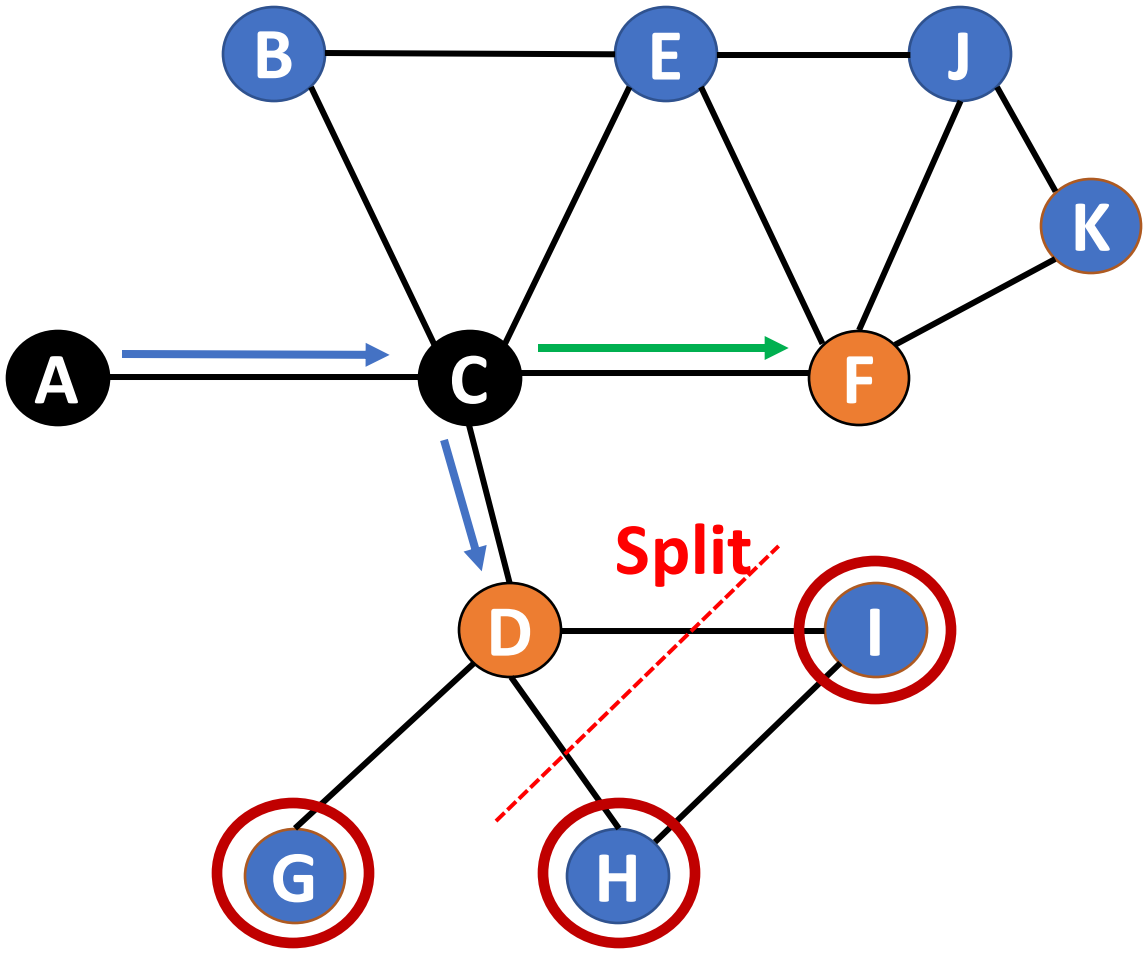
processed nodes  processing in progress  yet to be processed

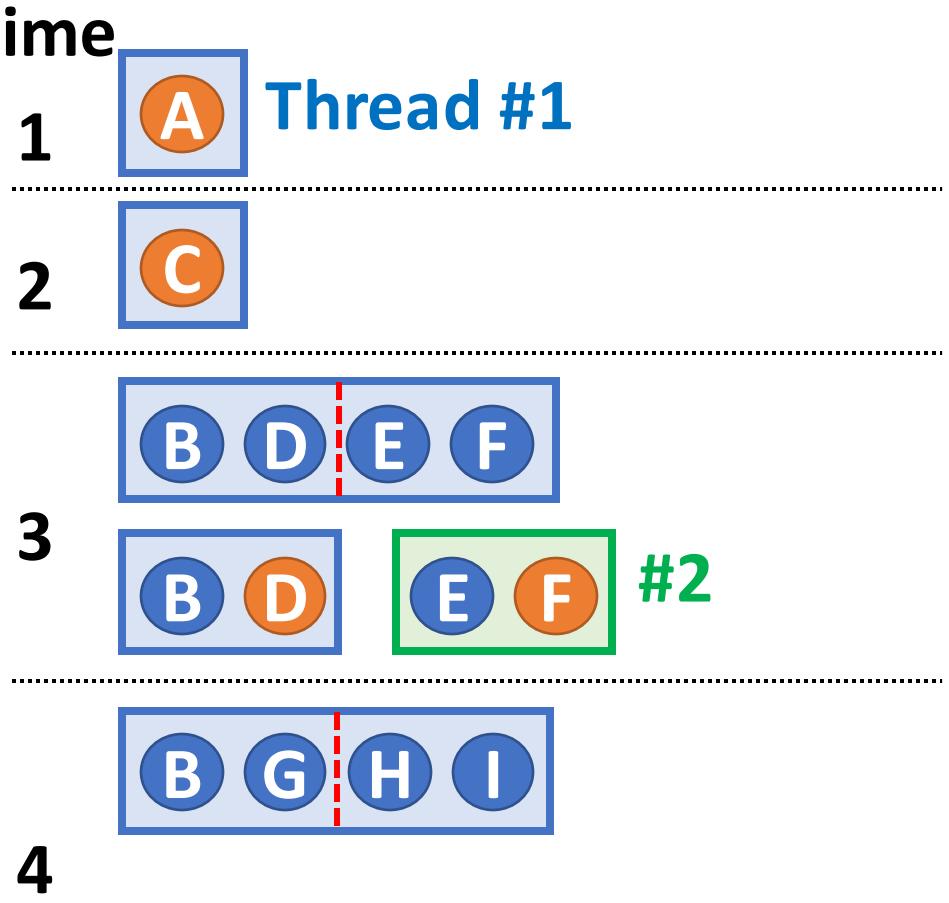


# Parallel pseudo DFS

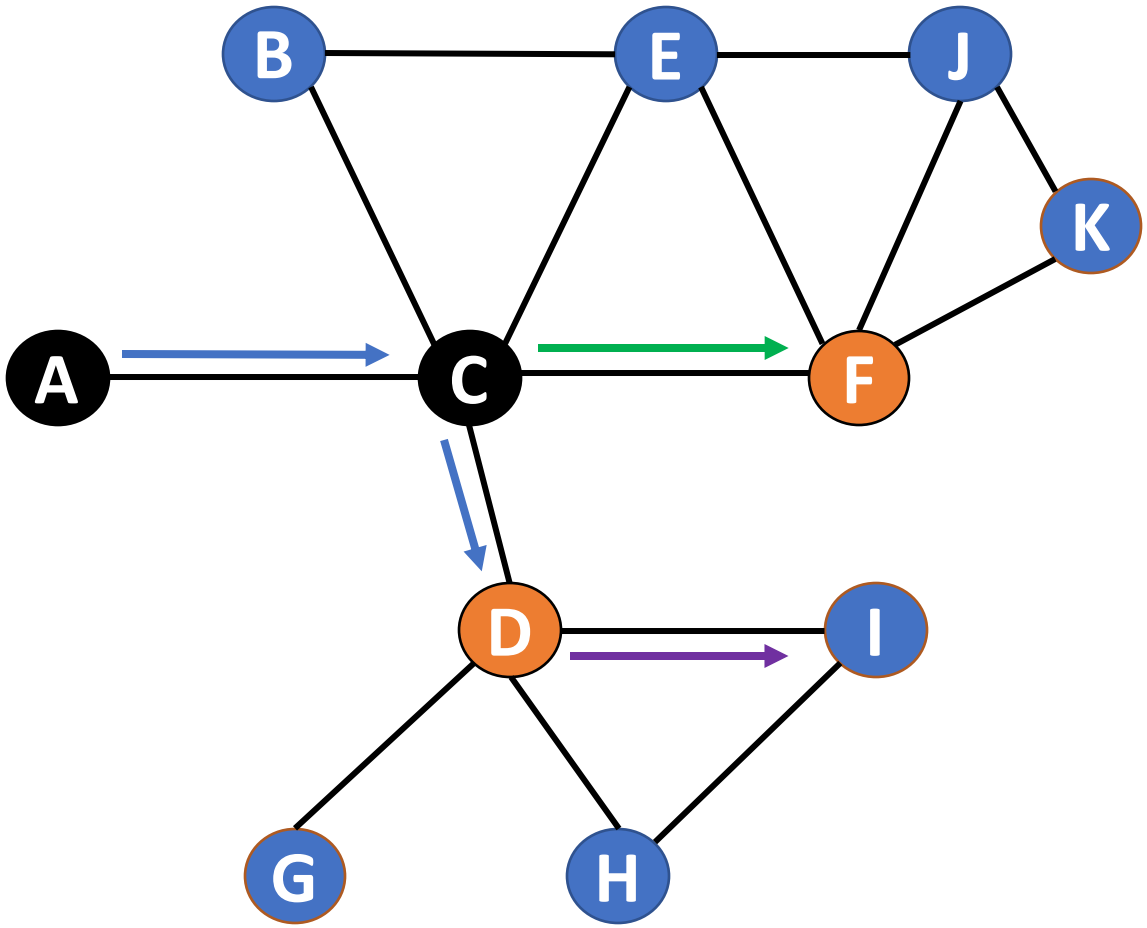
processed nodes  processing in progress  yet to be processed

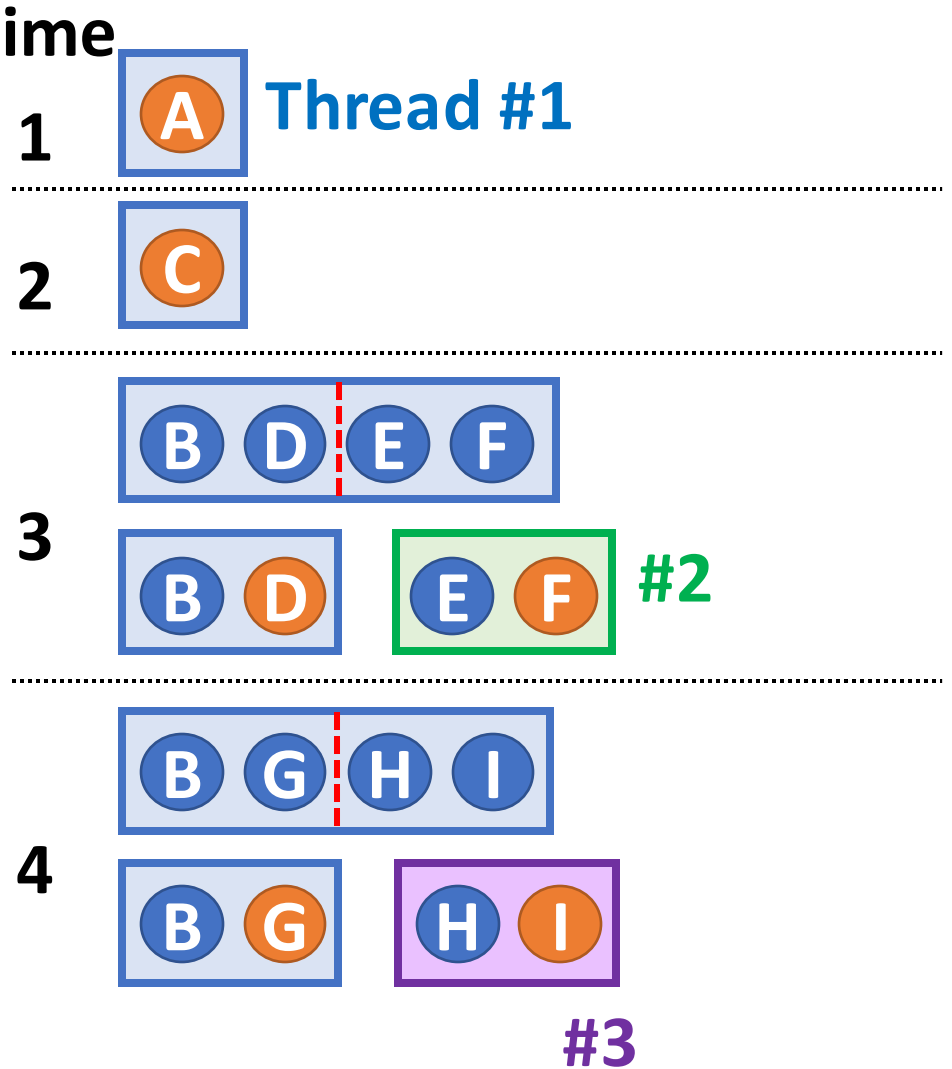


# Parallel pseudo DFS

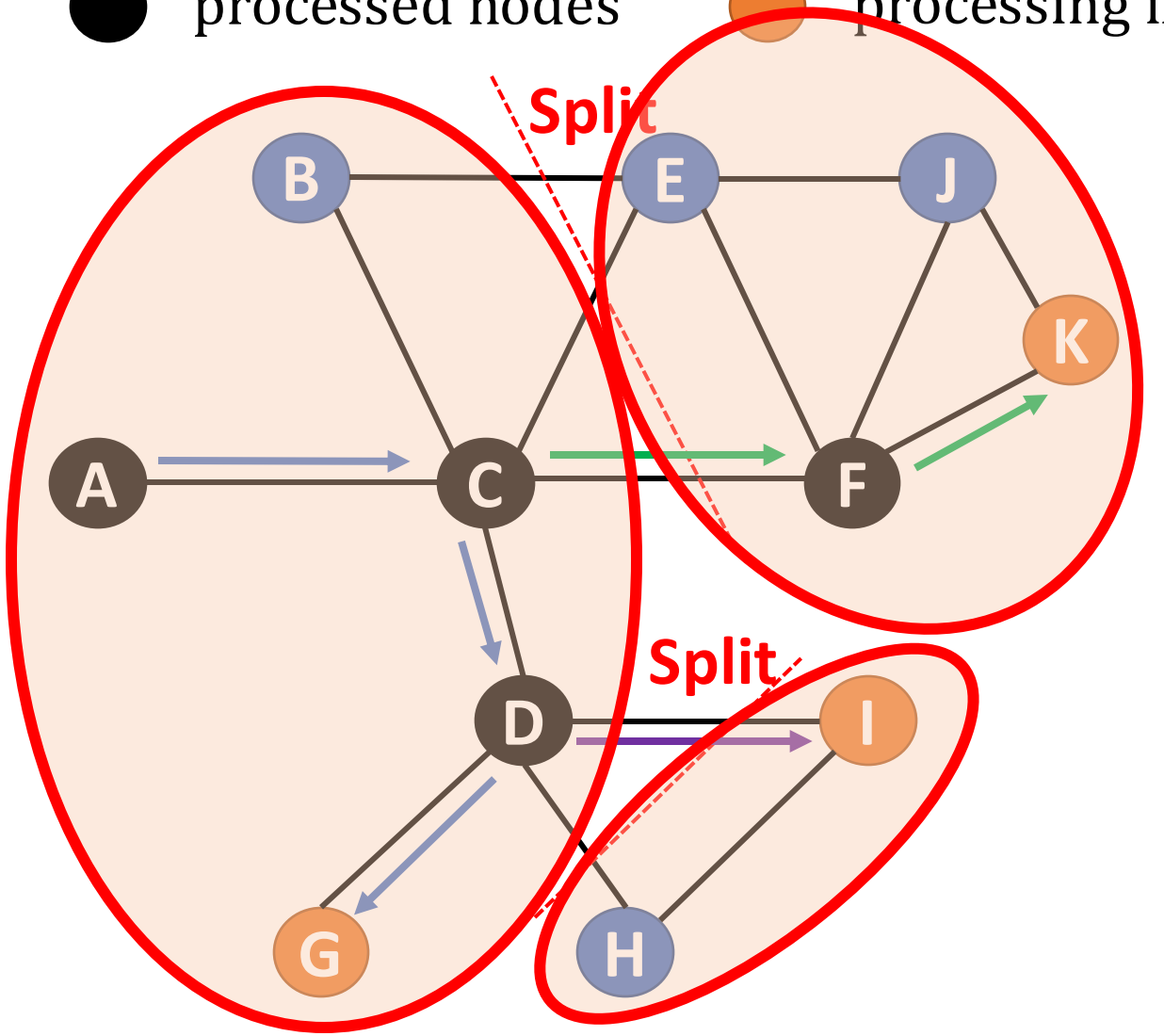
processed nodes  processing in progress  yet to be processed



# Parallel pseudo DFS

processed nodes  processing in progress  yet to be processed




Time

Up to $k_r$ stacks!

1 — A — Thread #1

2 — C

3 — B D E F

B D — E F — #2

4 — B G H I — E J K

B G — H I — E J K

#3

# Experimental Evaluation

6 datasets

| Dataset | Description | #Nodes | #Edges | Diameter | Size |
|---|---|---|---|---|---|
| FS | Friendster Social Network | 65M | 1.8B | 32 | 32 GB |
| TW | Twitter Social Network | 53M | 2B | 18 | 28 GB |
| RN | RoadNet Network of PA | 1M | 1.5M | 786 | 47 MB |
| LJ | LiveJournal Social Network | 5M | 69M | 16 | 1 GB |
| YT | YouTube Social Network | 1.1M | 3M | 20 | 39 MB |
| SD | Synthetic data | 50M | 1.25B | 6 | 20 GB |

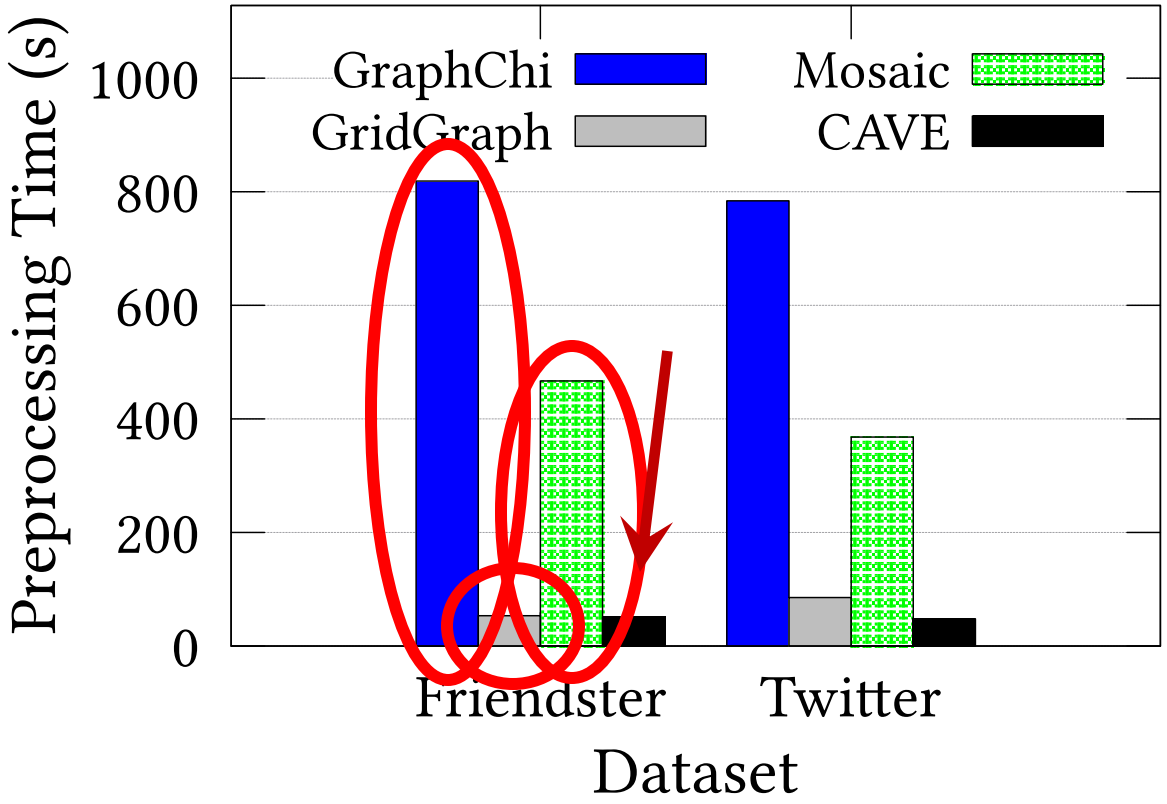
3 devices

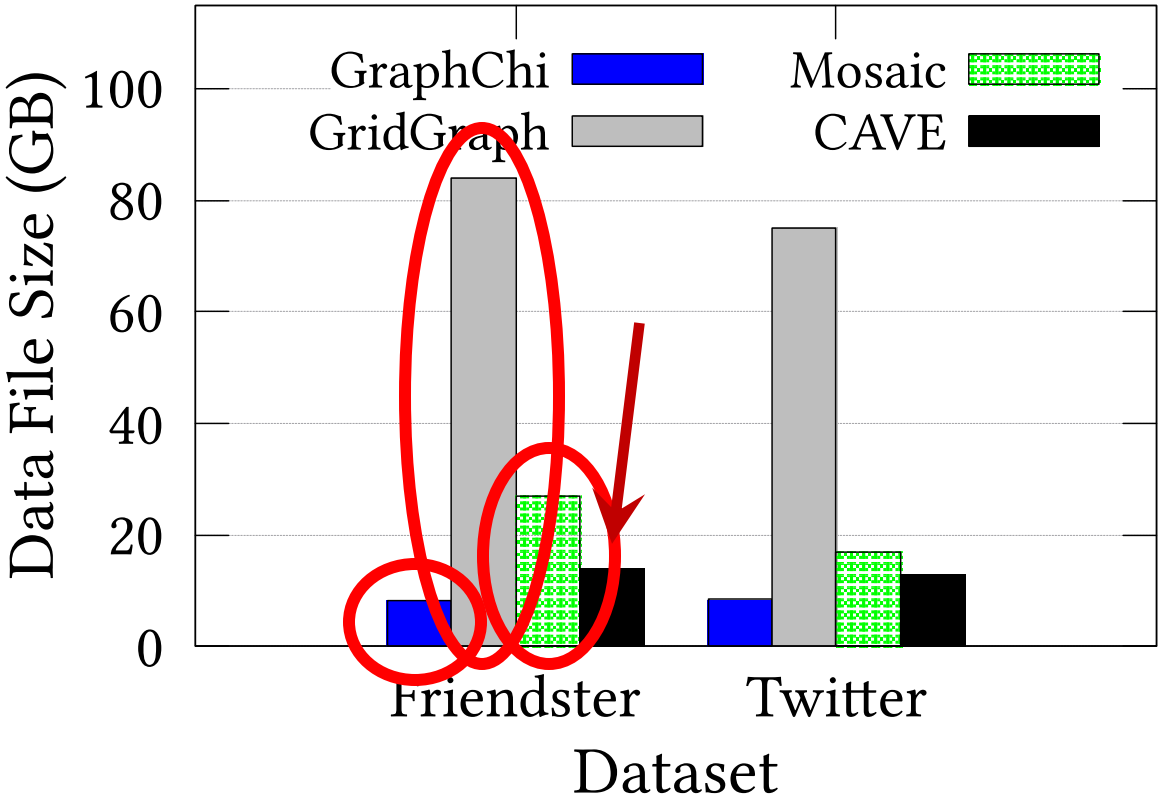
Optane SSD ($k_r = 6$)
PCIe SSD ($k_r = 80$)
SATA SSD ($k_r = 25$)

Approaches Used:

**GraphChi, GridGraph, Mosaic, CAVE, CAVE_blocked**

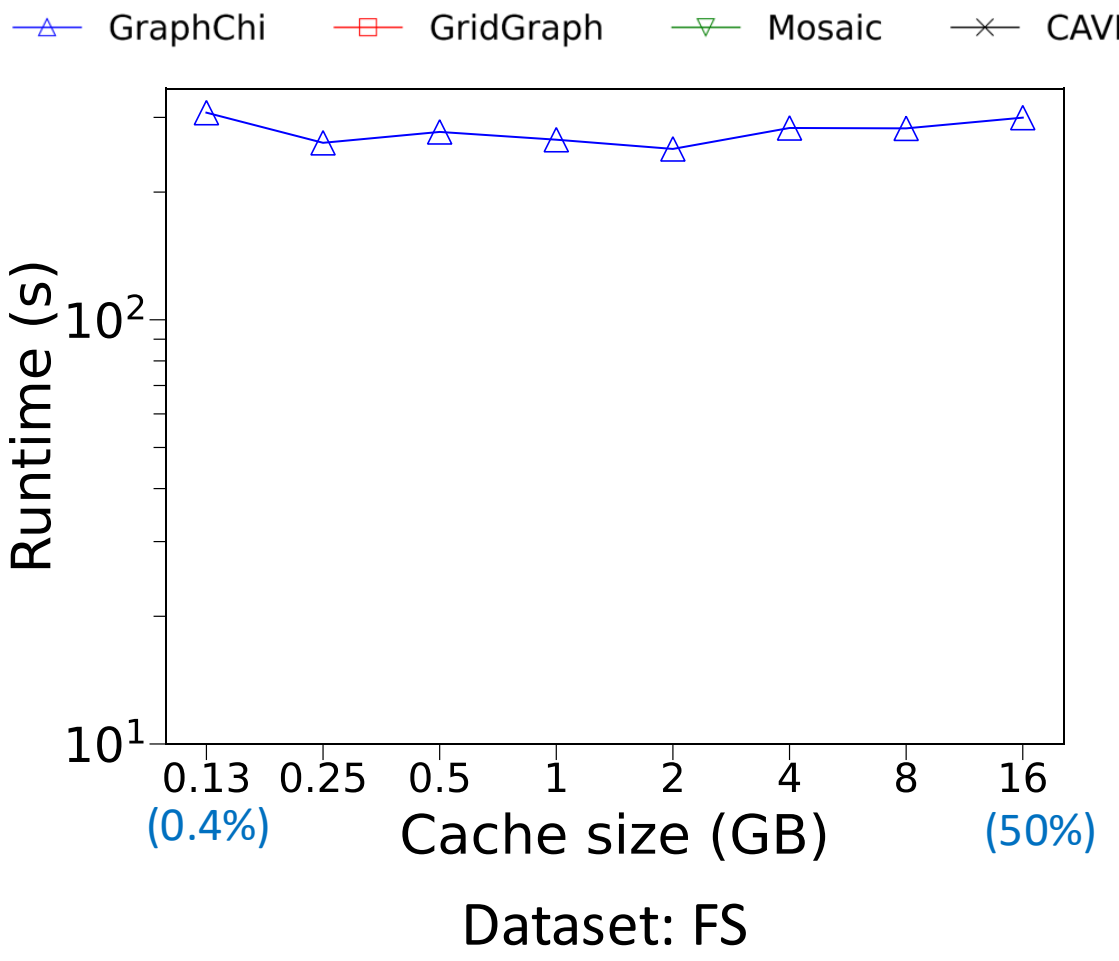
# Preprocessing Time and Space Requirement



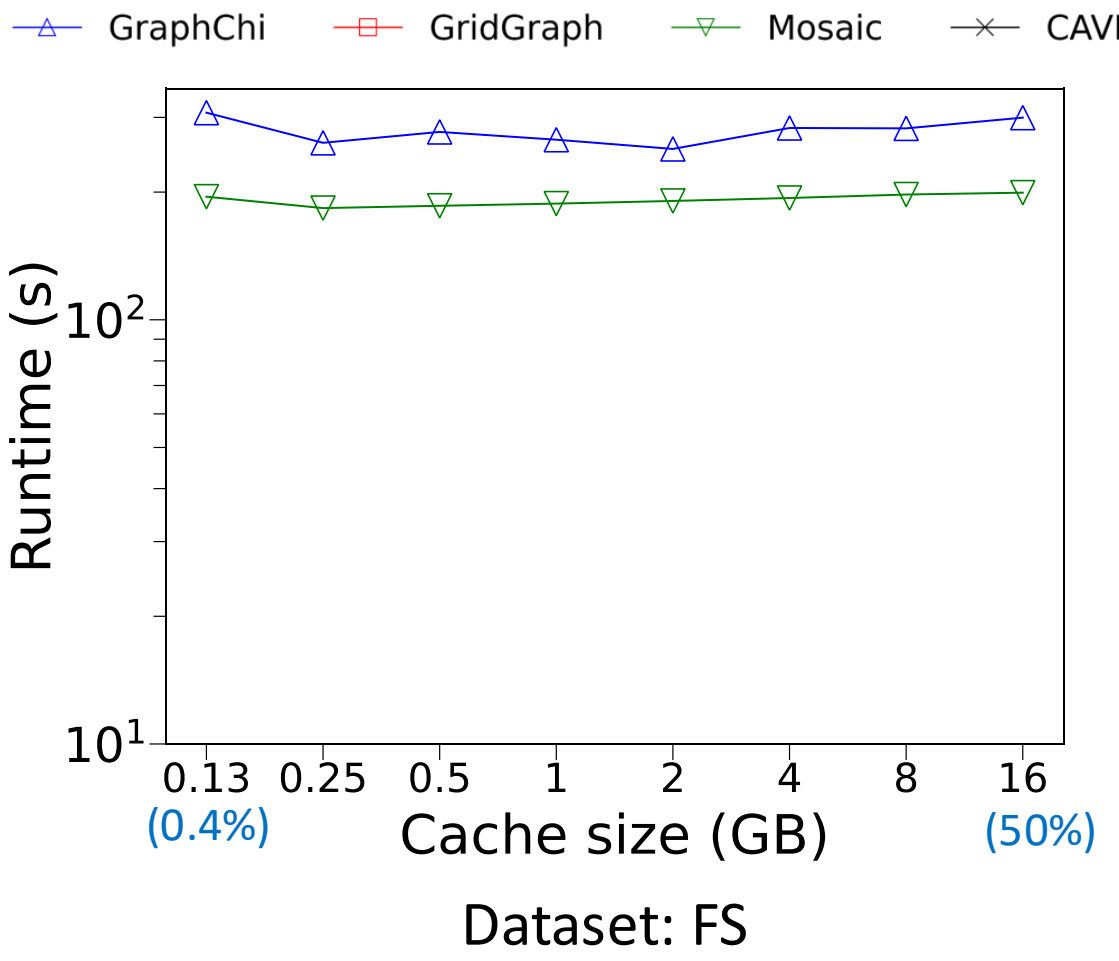

**CAVE** has **low preprocessing time** and **low file size**

# Evaluation: Parallel BFS



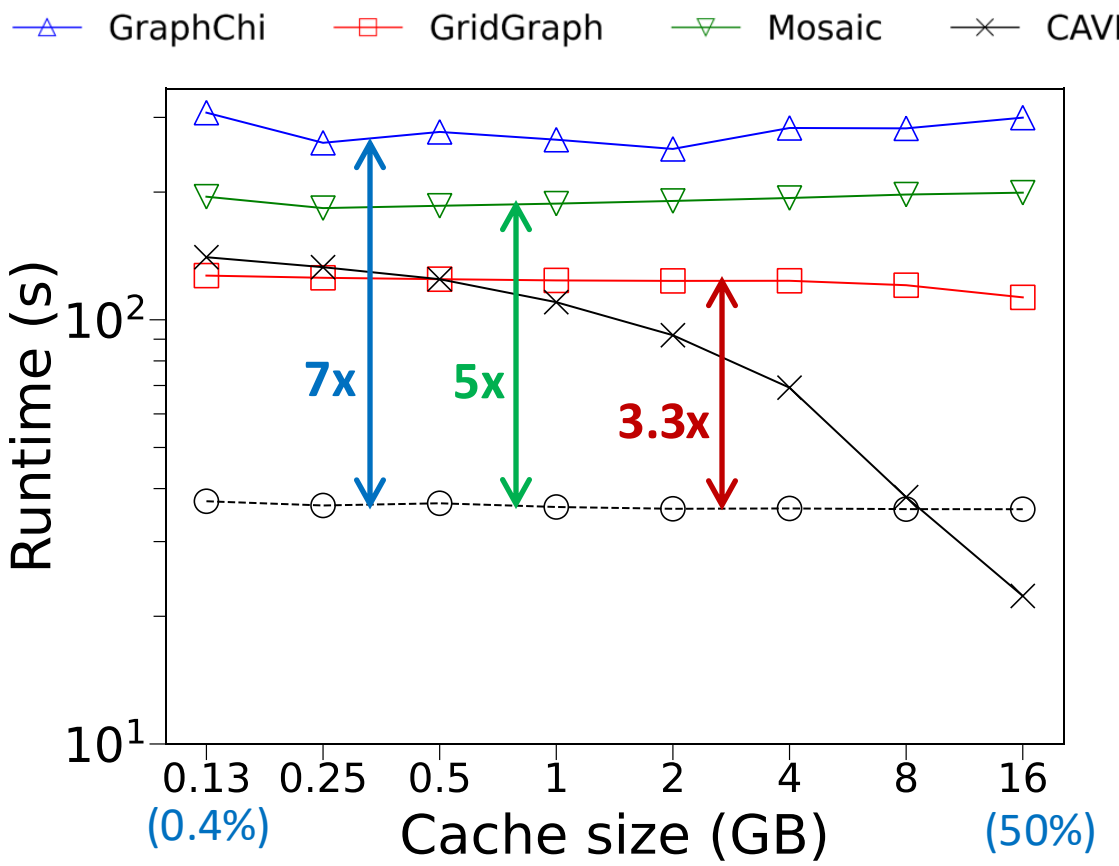

Dataset: FS

# Evaluation: Parallel BFS

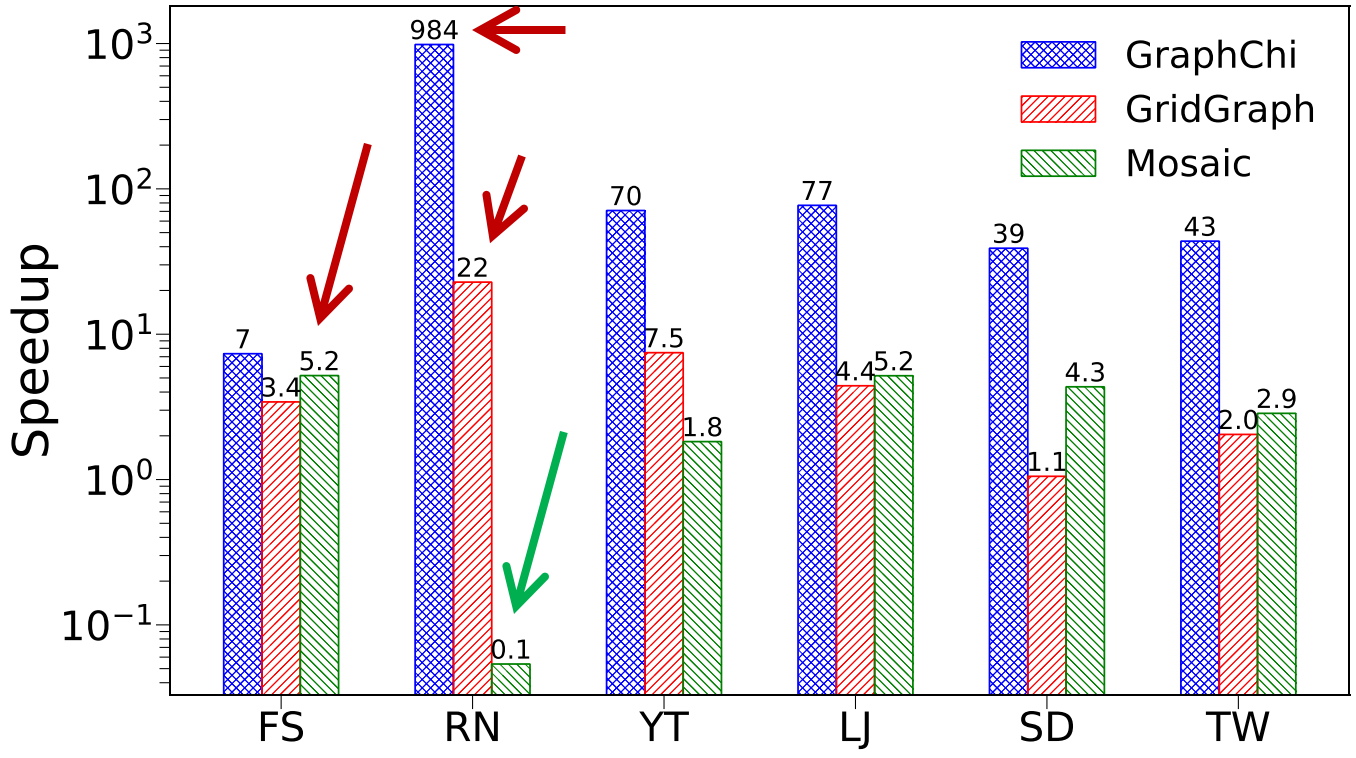



Dataset: FS

# Evaluation: Parallel BFS



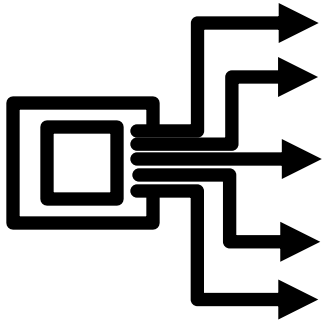

Dataset: FS



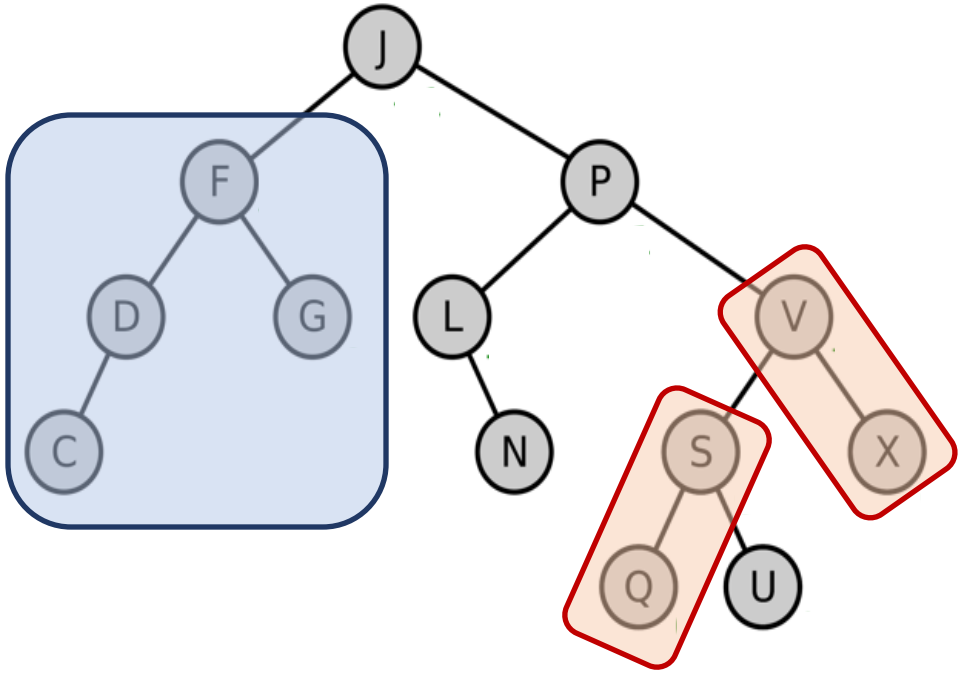

CAVE's Speedup

**Both CAVE implementations outperforms GridGraph, Mosaic and GraphChi**

# Summary



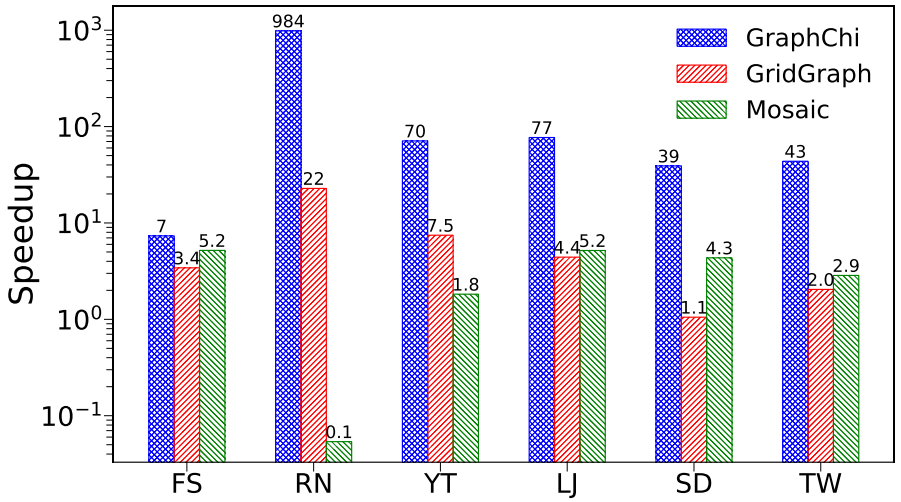
SSD concurrency can accelerate graph traversal




Intra- and inter-subgraph parallelization

**Concurrency-Aware Graph (V, E) Manager**

**CAVE**



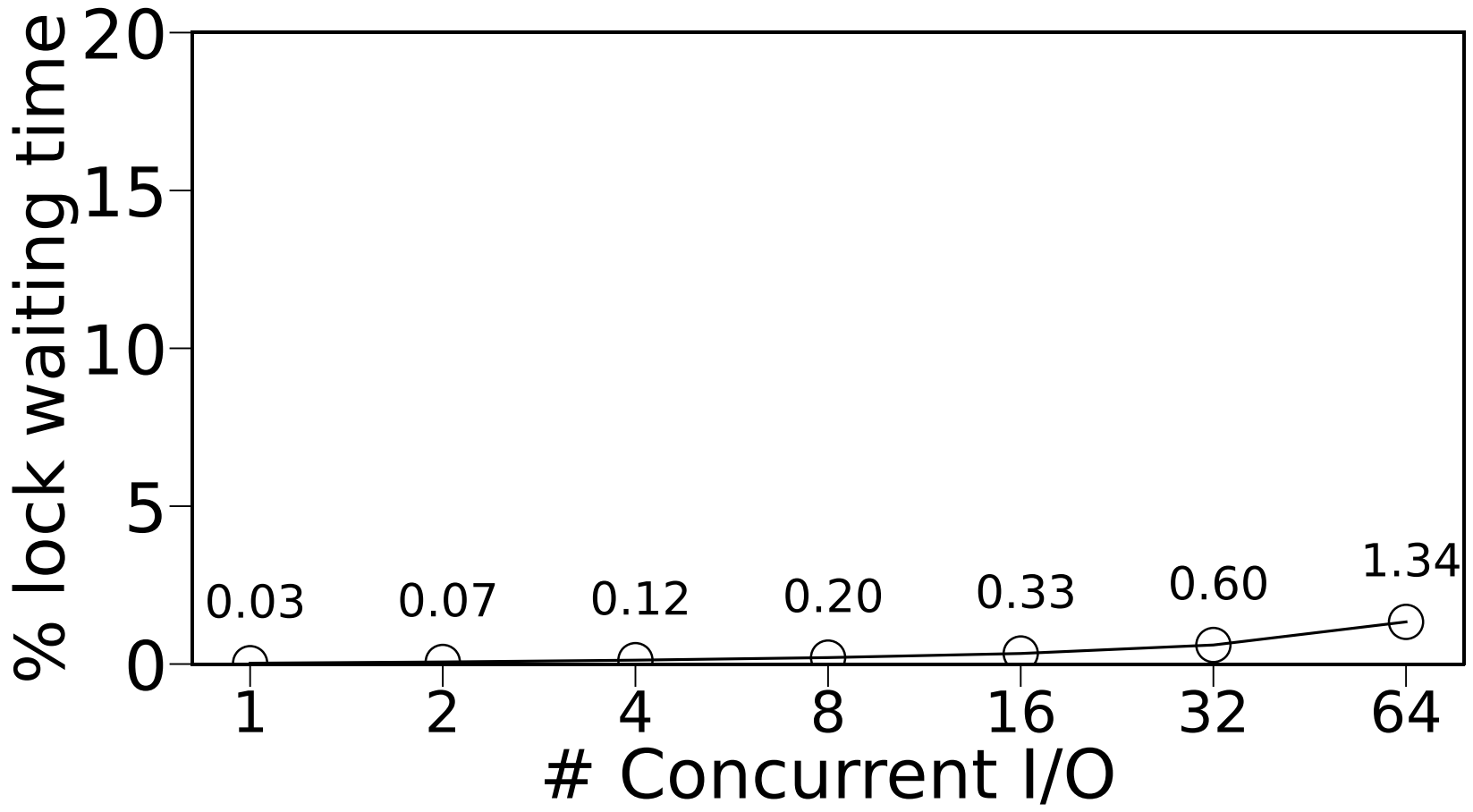

CAVE implementations outperform SOA systems

# Thank You!

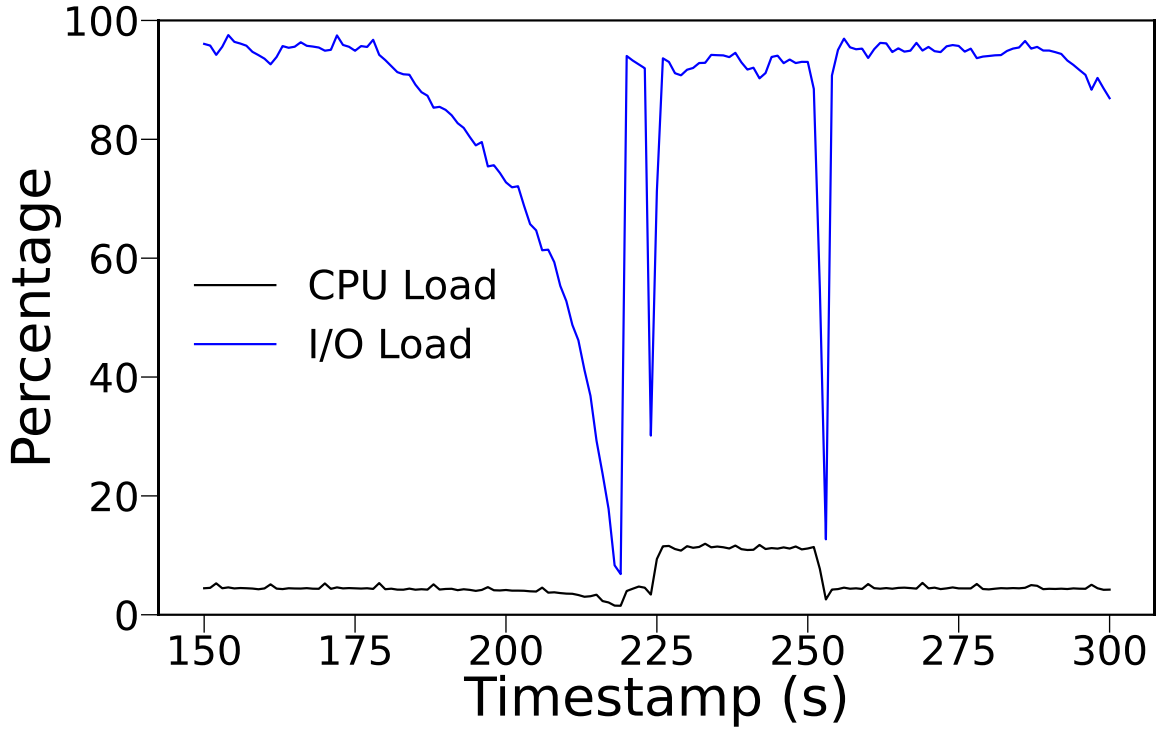
cs-people.bu.edu/papon

disc.bu.edu/papers/sigmod24-cave

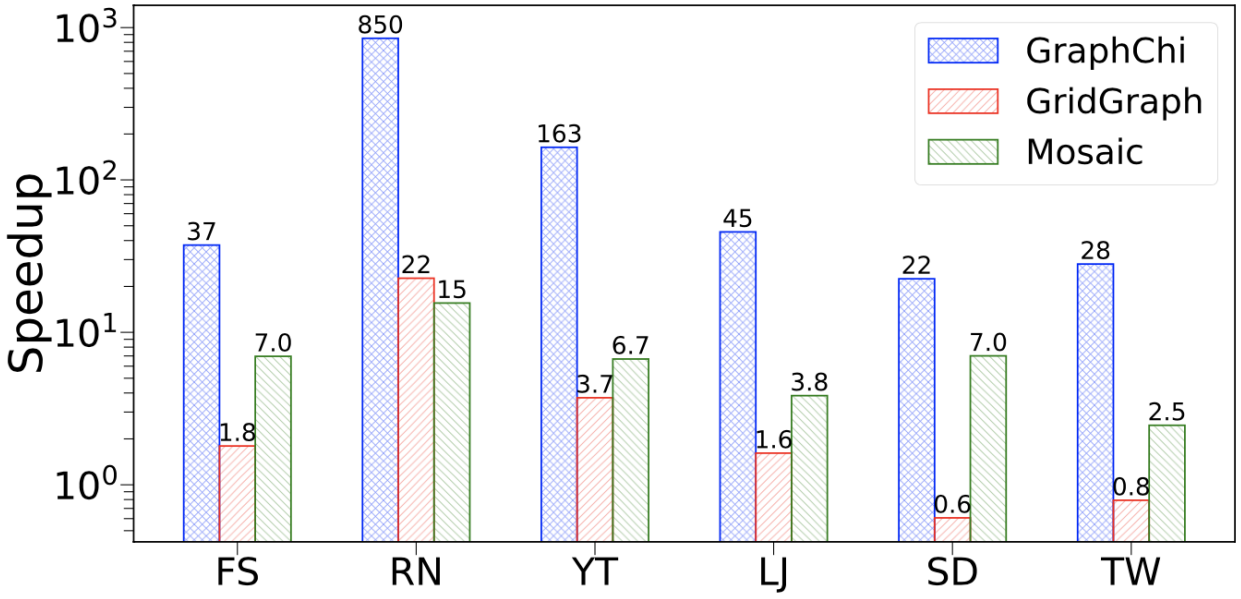
# Lock Waiting Time is Low



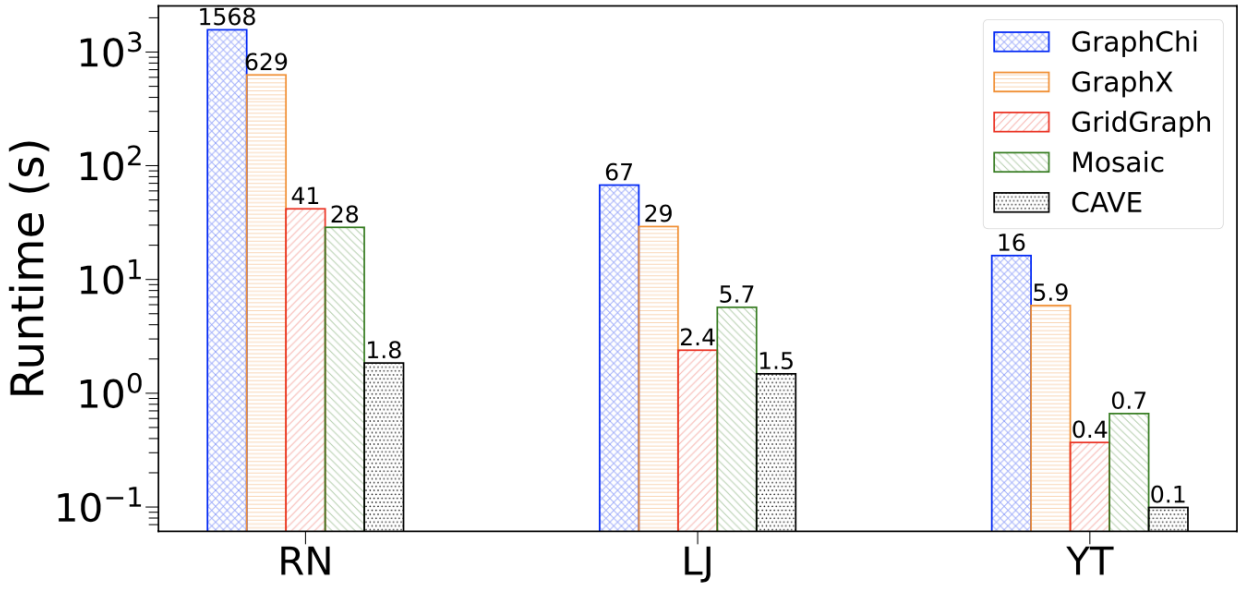
# CAVE is I/O Bound



# CAVE Performs Well for PWCC



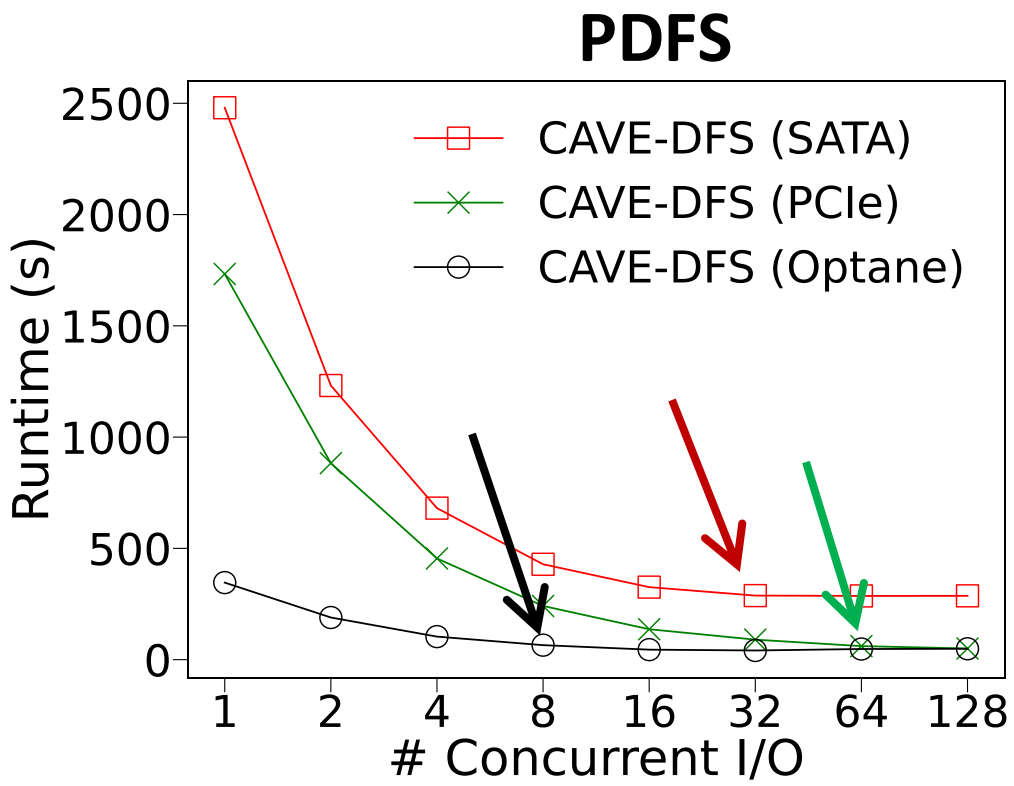

(A) CAVE's speedup for PWCC




(B) Experiment with GraphX and other systems

# CAVE Utilizes Concurrent I/O

Dataset: FS



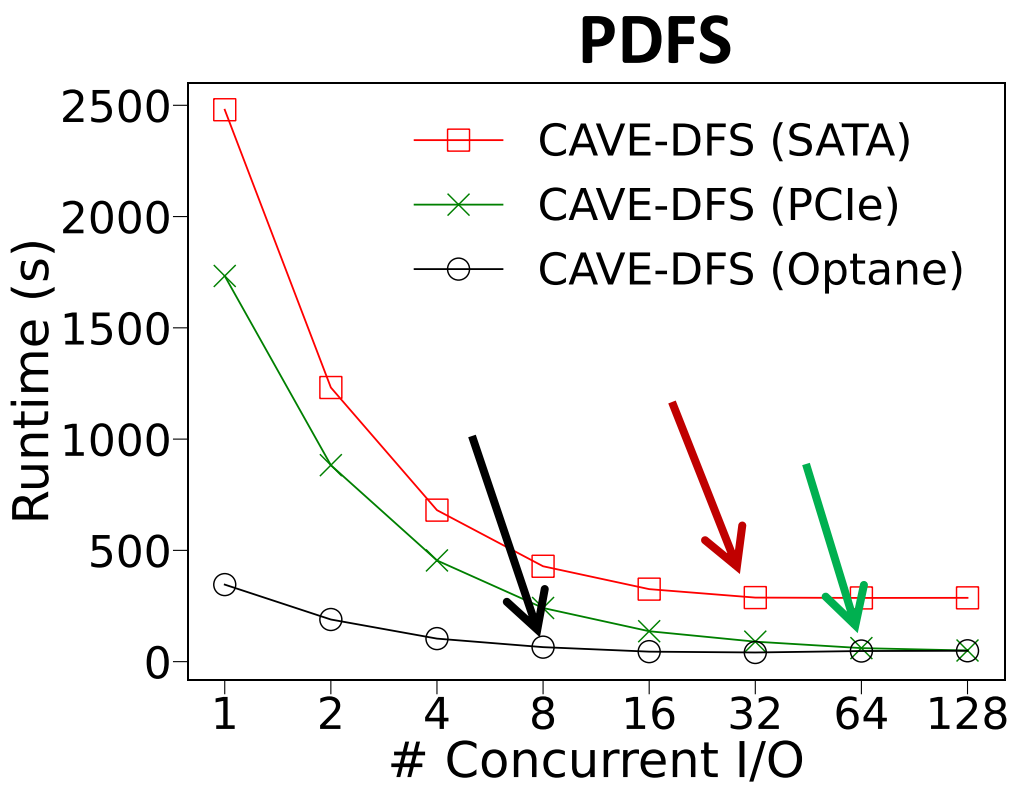

SATA SSD ($k_r = 25$)

PCIe SSD ($k_r = 80$)

Optane SSD ($k_r = 6$)

# CAVE Utilizes Concurrent I/O

Dataset: FS




SATA SSD ($k_r = 25$)

PCIe SSD ($k_r = 80$)

Optane SSD ($k_r = 6$)

**Device gets saturated at *optimal concurrency***