CAVE: Concurrency-Aware Graph Processing on SSDs

BOSTON UNIVERSITY

Tarikul Islam Papon Taishan Chen Shuo Zhang Manos Athanassoulis

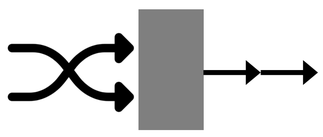
SOA Out-of-core Systems for Large Graphs

Real-world graphs often have more than a billion nodes!

SOA Out-of-core Systems Rely on

Data partitioning

Improve memory & disk locality

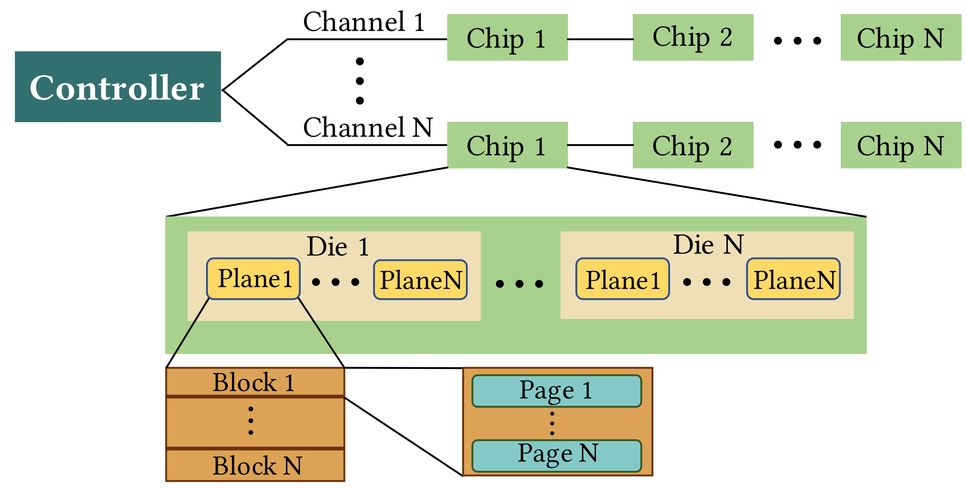


Reduce random I/O

They are Designed for HDDs

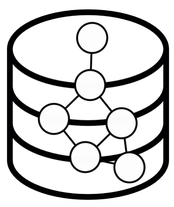
"Tape is Dead. Disk is Tape. Flash is Disk." - Jim Gray

SSD Concurrency



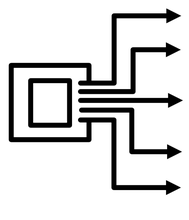
Parallelism at different levels (channel, chip, die, plane block, page)

Our Goal

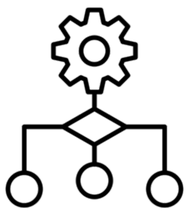


Optimize for **storage-based** graph workloads

Focus on **traversal** operations



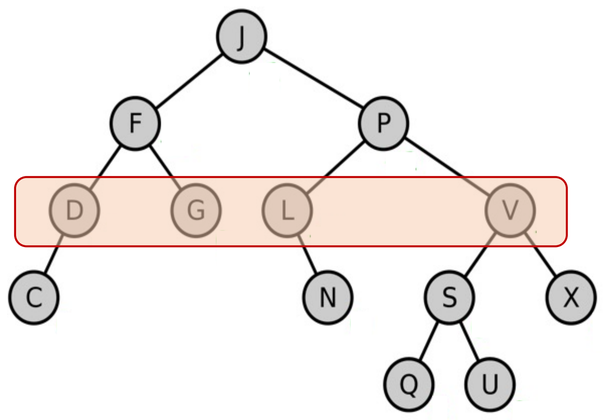
Utilize *optimal* **SSD Concurrency**



Maintain core **algorithm properties**

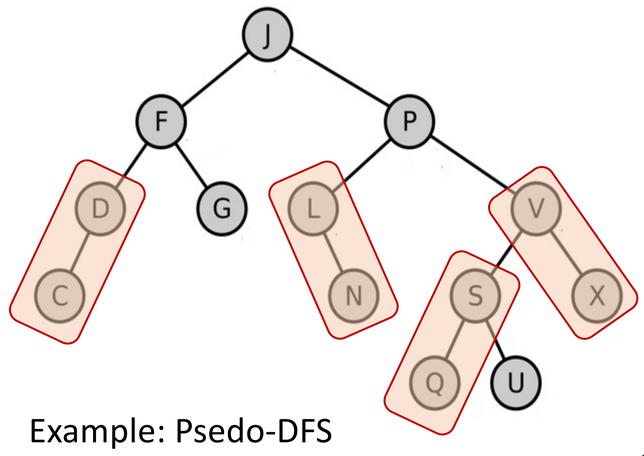
Parallelizing Graph Traversal Operations

Intra-Subgraph Parallelization



Example: BFS

Inter-Subgraph Parallelization



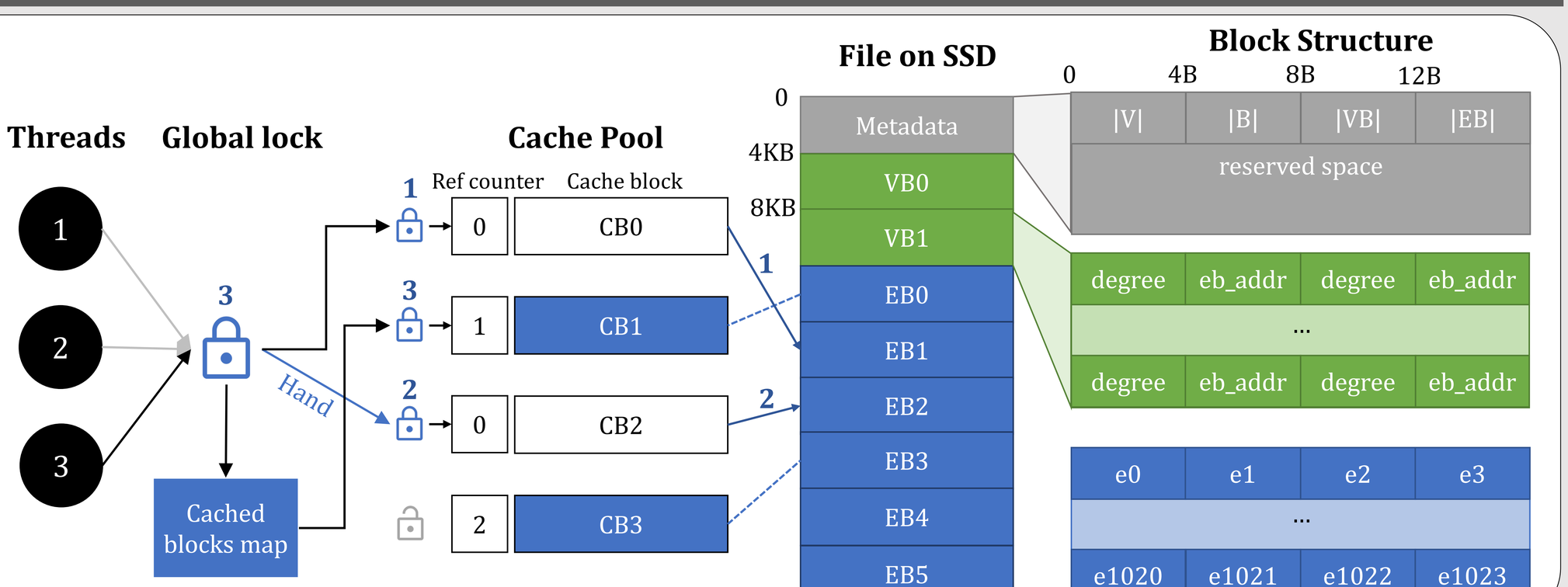
Example: Psedo-DFS

CAVE Architecture

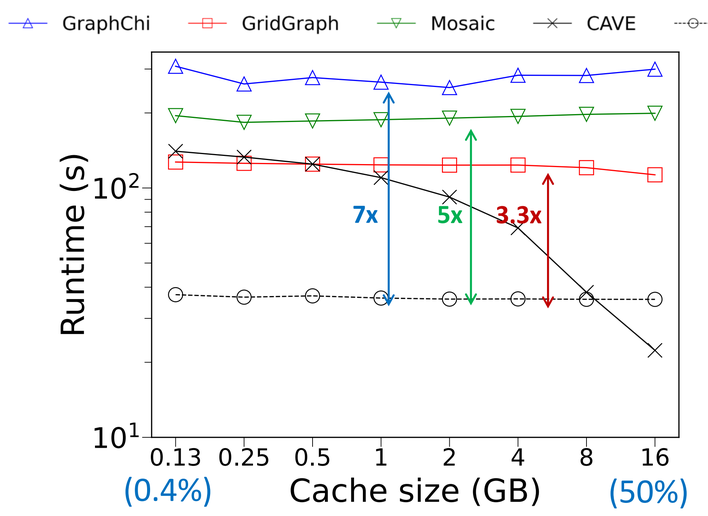
Concurrency-Aware Graph (V, E) Manager

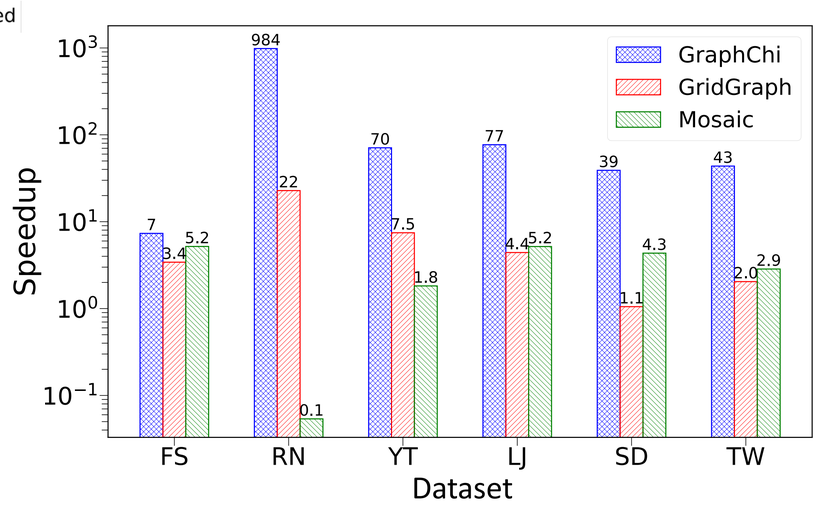
CAVE

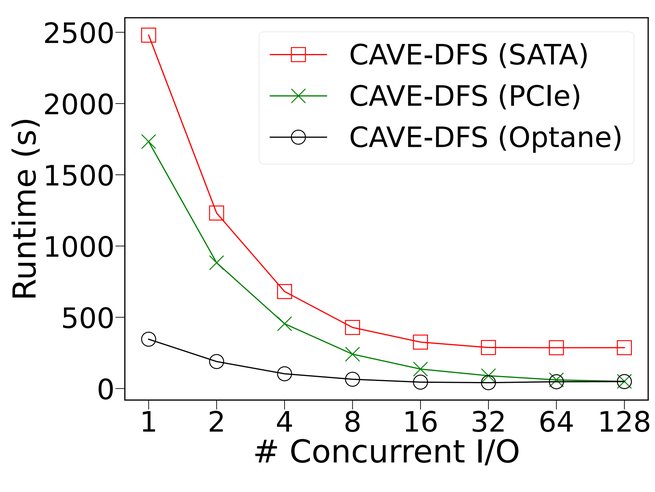
- ✓ Compact data layout
- ✓ Concurrent cache pool for edge blocks
- ✓ Designed to exploit SSD concurrency



CAVE: Performance Evaluation







CAVE outperforms GridGraph, Mosaic and GraphChi. Device gets saturated at *optimal concurrency*.

BOSTON UNIVERSITY
College of Arts & Sciences
Department of Computer Science
disc.bu.edu
DiSC lab