NoSQL Systems Project

CS265 Spring 2025

DASlab
@ Harvard SEAS

Aims & Scope

- Designing and implementing a Log-Structured-Merge-tree, i.e., **LSM-tree**, as a key-value store
- Designing a system
- C/C++ implementation
- Low-level systems issues
  - Parallel processing, read/write trade-offs, etc.

- **Main idea:** *Buffered writes at expense of reads*

Main Memory

Secondary Storage

Buffer (L0)

L1 Run #1 Run #2 ... Run #N1

L2 Run #1 Run #2 ... Run #N2

⋮

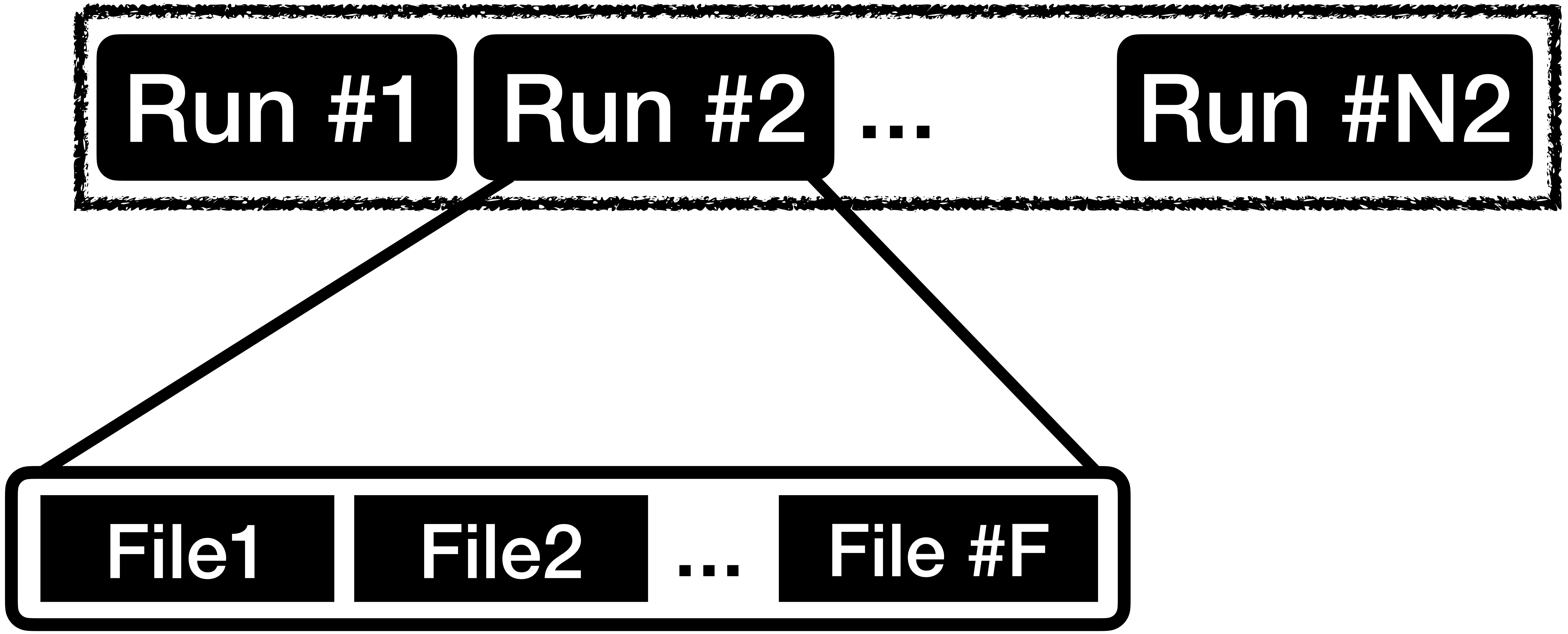
LM Run #1 Run #2 ... Run #N3

Main Memory

Buffer (L0)

- **Heap**
- **Skiplist**
- **Btree**

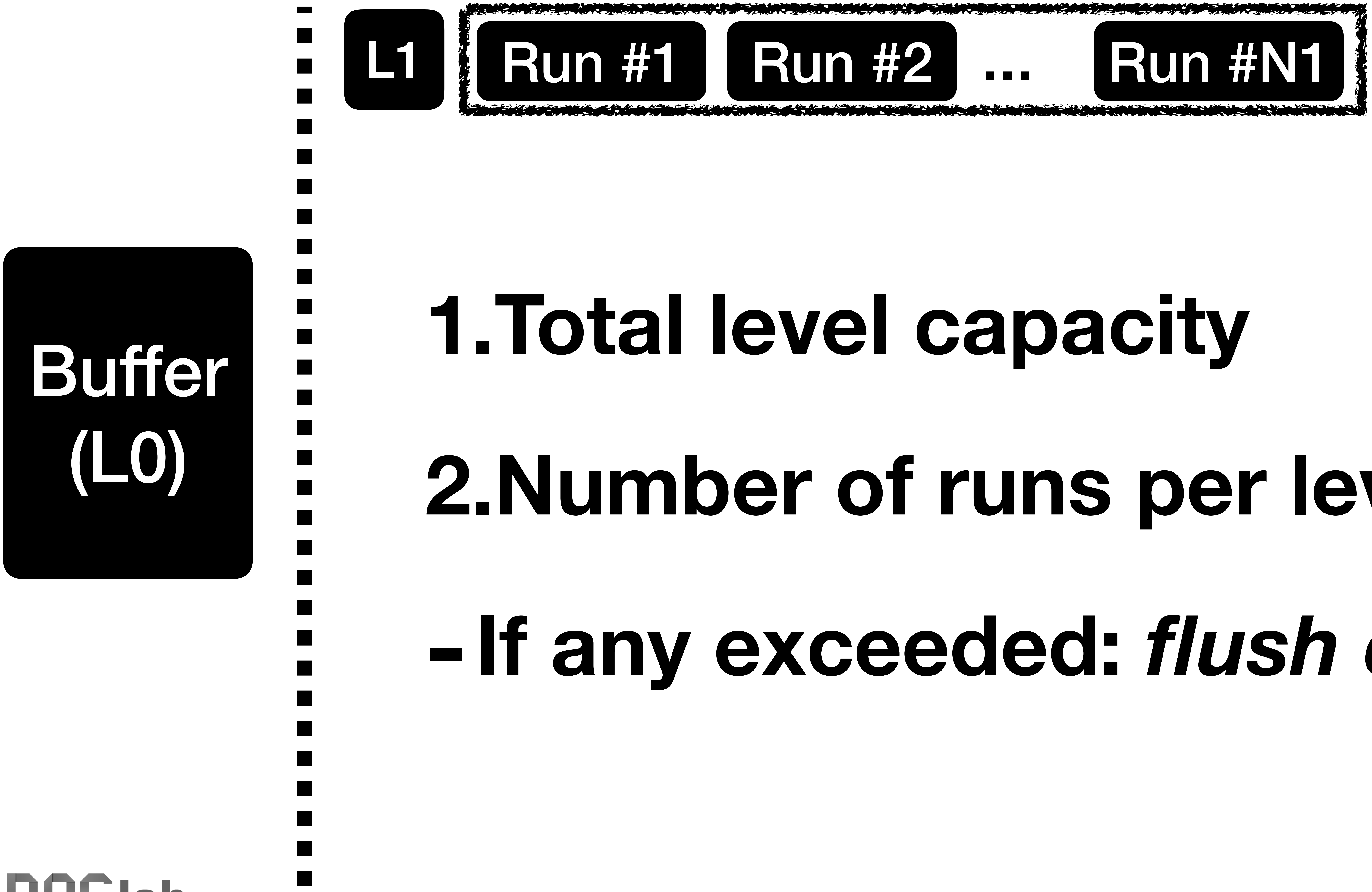
Fast data ingestion



Run #1 Run #2 ... Run #N2

File1 File2 ... File #F

A set of sorted files with non-overlapping key ranges



L1

Run #1

Run #2

...

Run #N1

Buffer (L0)

1. **Total level capacity**

2. **Number of runs per level**

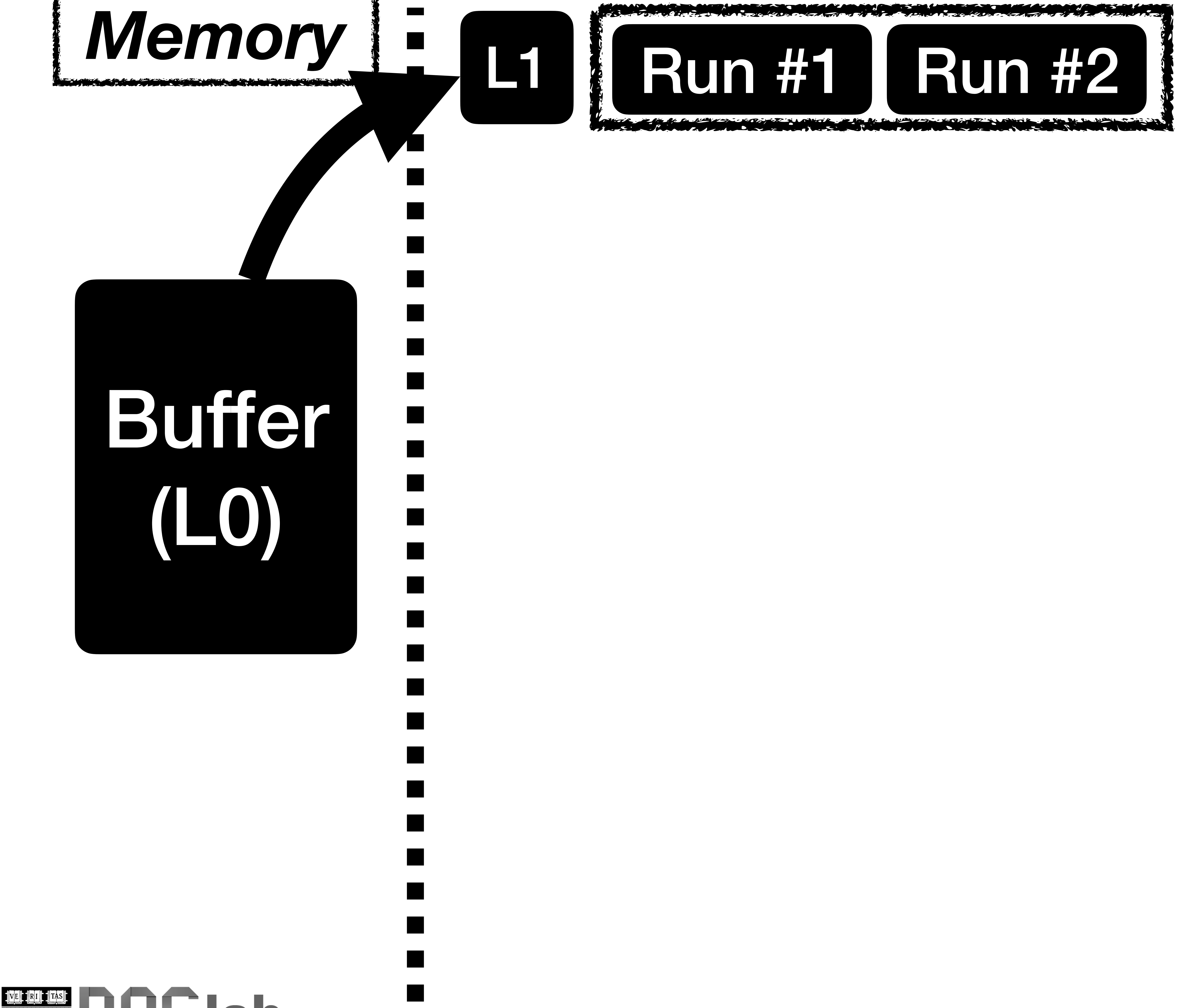
- **If any exceeded: *flush down***

Main Memory
L1
Run #1
Secondary Storage
Buffer (L0)

#run per level ths: 3

Main Memory

Secondary Storage



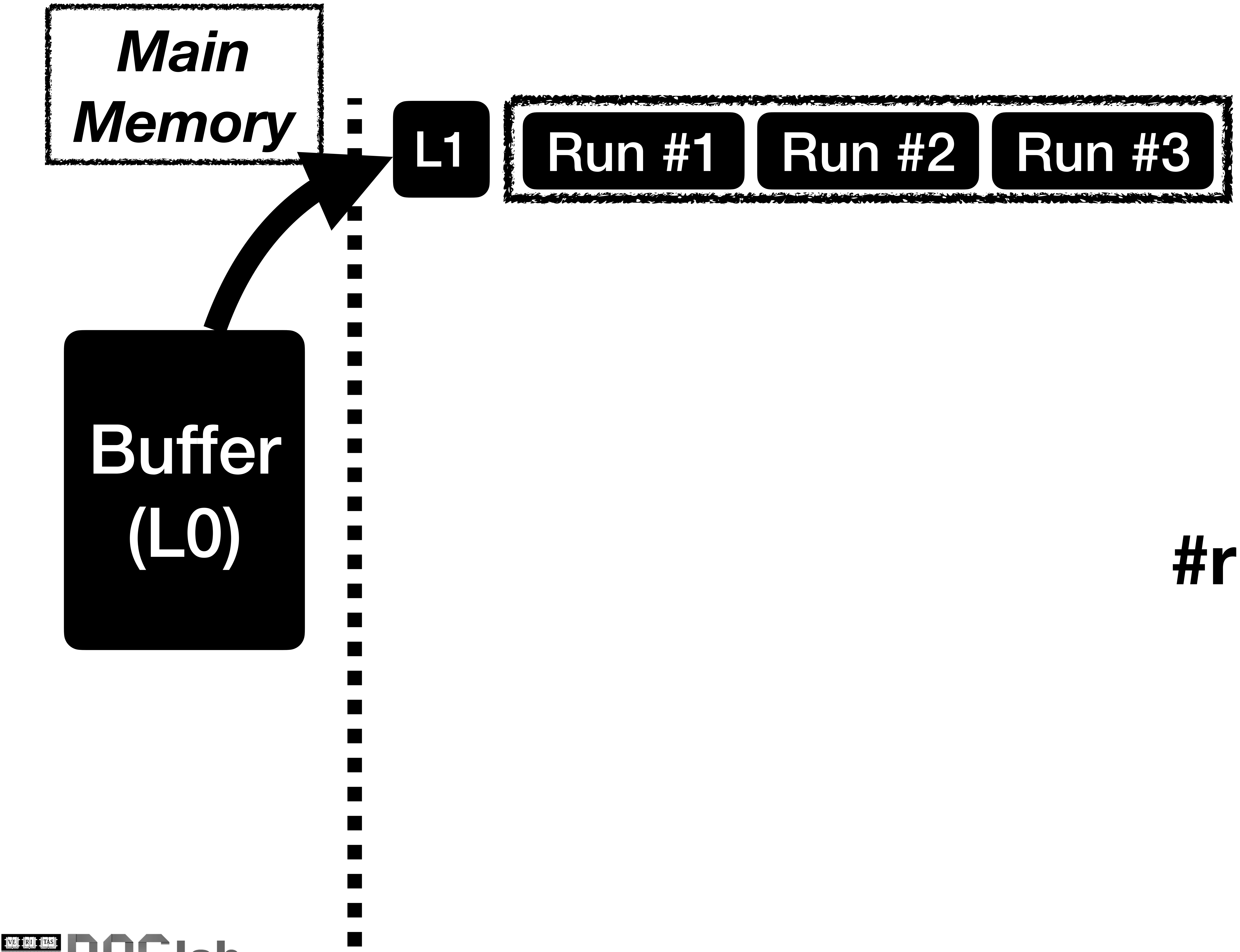
L1

Run #1

Run #2

Buffer (L0)

#run per level ths: 3



Main Memory

Buffer (L0)

L1

Run #1

Run #2

Run #3

Secondary Storage

#run per level ths: 3

Main Memory
Secondary Storage
un #1
Run #2
Run #3
Buffer (L0)

Max #runs/level threshold reached!

Flush down!

#run per level ths: 3

Main Memory

Secondary Storage

Buffer (L0)

L1

Run #1

Run #2

Run #3

Merge

L2

Run #1

#run per level ths: 3

Main Memory
Secondary Storage
Buffer (L0)
L1
L2
Run #1

#run per level ths: 3

DASlab @ Harvard SEAS

Main Memory
Secondary Storage
L1
Run #1
L2
Run #1
Buffer (L0)

#run per level ths: 3

Cascading merges

Main Memory
Bloom filters
Fence pointers
Secondary Storage
Buffer (L0)
L1
L2
LM

Balances look-up time

The Project

- **Two parts**
  1. Designing the basic structure of an LSM tree for reads and writes
  2. Same functionality in a parallel way so we can support multiple concurrent reads and writes.
- **Open ended; e.g.:**
  - Each level may be designed in its own way
  - Each level may be a complex or simple data structure
    - Tree vs. simple array

The Project

- **Minimum design**
  - Align with Monkey or Dostoevsky paper
    - Merge policies
    - One bloom filter and fence pointer per level
    - …
- **Additional design considerations**
  - At least three optimizations: size ratio between levels, buffer data structure, etc.
- **See:** http://daslab.seas.harvard.edu/classes/cs265/project.html

Midway checkin

- **Three deliverables**
    1. Design document, describing in detail the first phase of the project
    2. 45 minute presentation that describes the intended design for the whole project
    3. At least two performance experiments that demonstrate an unoptimized variant of a get and a put operation.

Final deliverable

- **Two deliverables**
  1. A code deliverable + code review + demo = 50%
  2. A final paper and experimental analysis = 50%

- **See for complete description & templates:**
  - http://daslab.seas.harvard.edu/classes/cs265/project.html

Toolchain

- C/C++
  - Rust also fine
- Any compiler and IDE is fine
  - VS Code is common
- OS: Linux, but Windows is also fine
- Client-server architecture & CS265 DSL

Experimental evaluation example

Performance graph

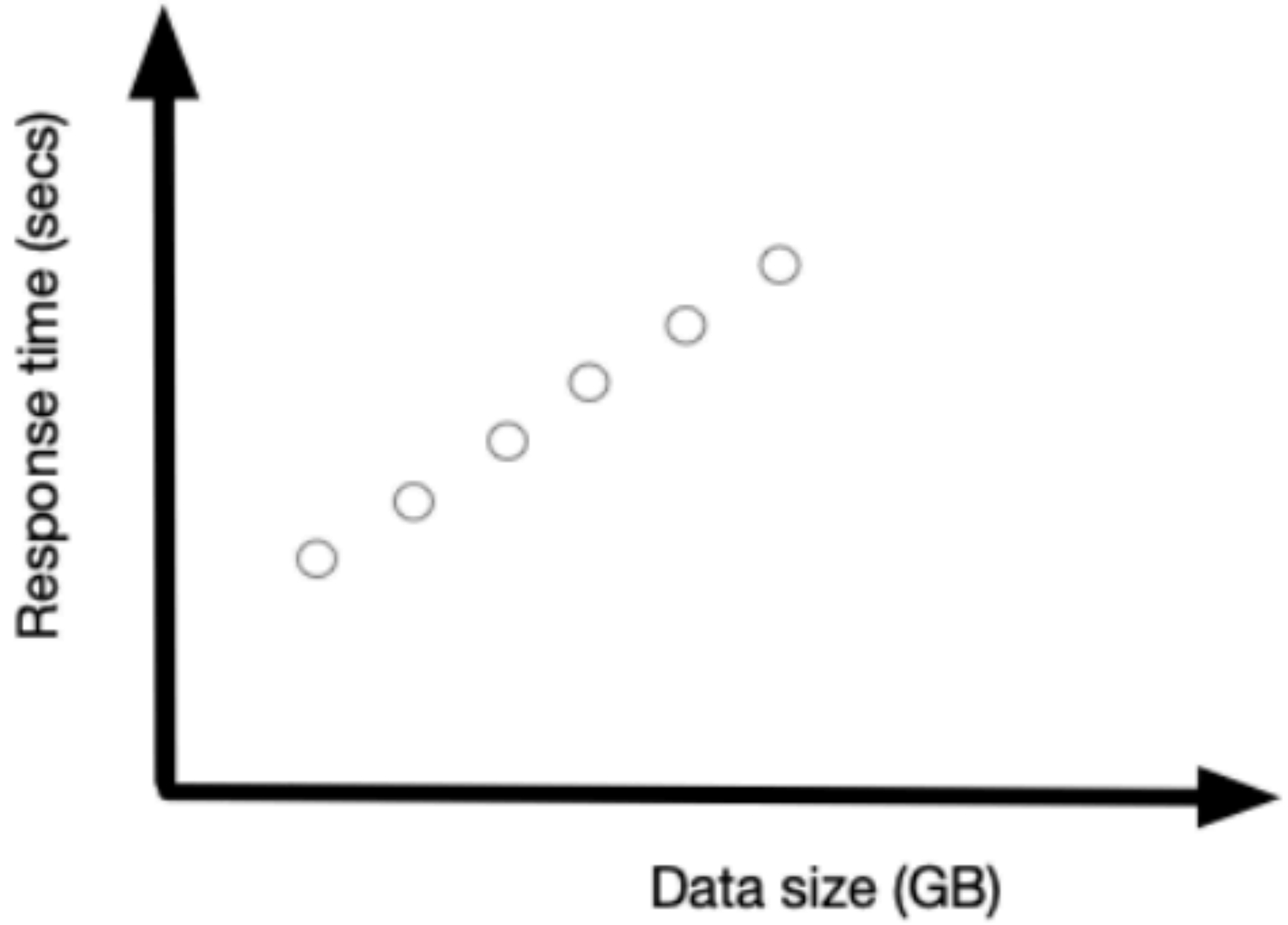


Figure 1. Caption of performance graph

Explanation graph

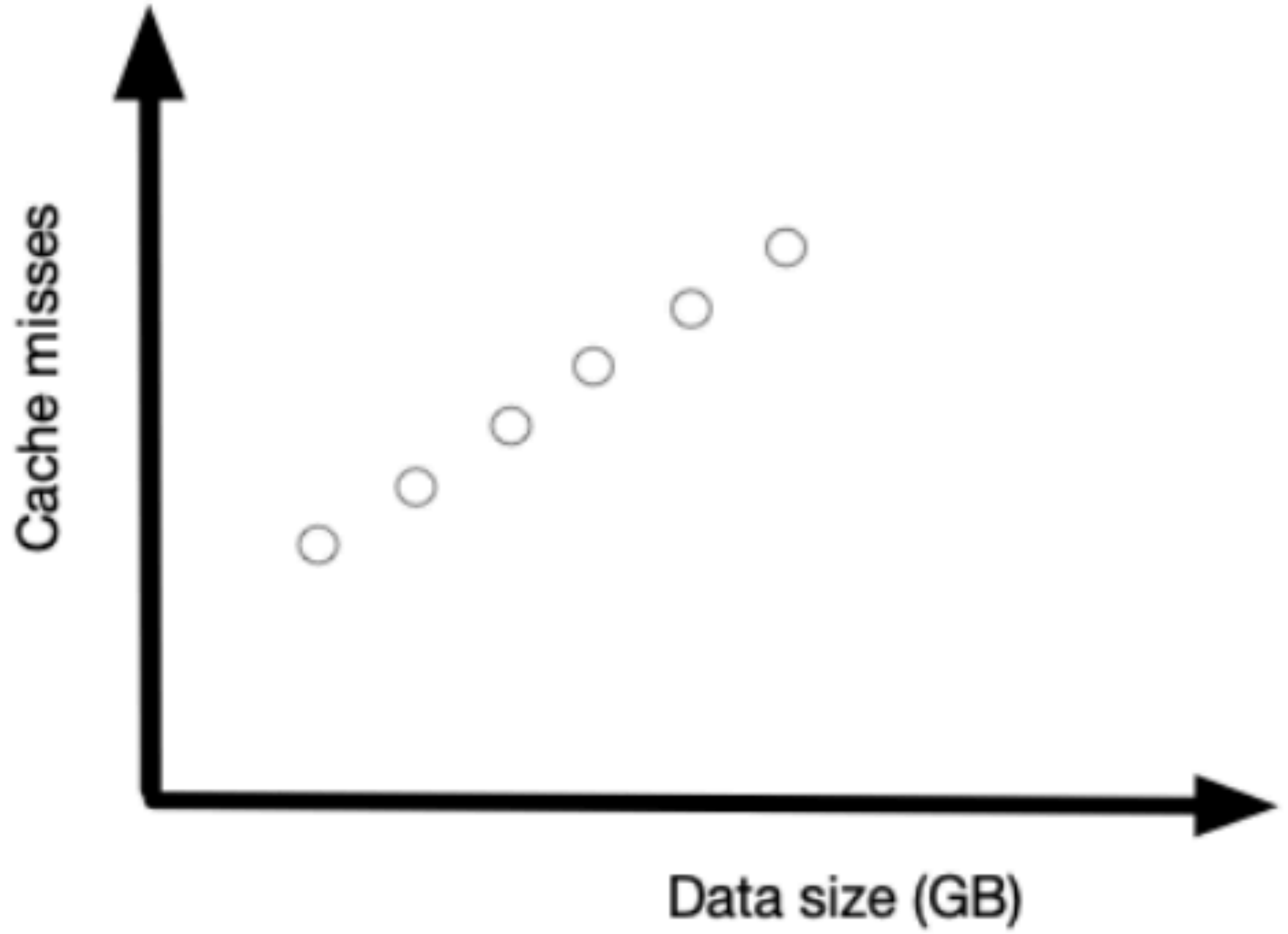


Figure 2. Caption of additional graph

That’s all folks!