And now for a brief paws

CSE 250
Lecture 34

Patterns in Data Science

Data Science Is Everywhere

- The Corporate World (e.g. MANGA)
- Open Data → Civic Computing
- Science!
- Internet of Things

Data Science Data is Big

- O(f(n)): The behavior of f(n) as n gets really really big
- Data Science works with 100MBs, TBs of data
  - n gets really really big

Today's Lecture

- Examples of a data science pattern
- Algorithms for the pattern (← useful for PA4)
- Twists on the pattern (← advanced ideas, not covered on Final)

Usage Pattern 1: MANGA

- Dataset: **Sales**
  - **productID**: Int
  - **date**: Date
  - **volume**: Int
- Objective
  - Find the 100 most purchased products from in the last month (by ID)

Usage Pattern 1: Open Data

- Dataset: **TrafficViolations**
  - **blockID**: Int
  - **infraction**: InfractionType
  - **date**: Date
- Objective
  - Find the fraction of parking tickets that were issued in each block (by the block's ID)

Usage Pattern 1: Science!

- Dataset: **Patient**
  - **patientId**: Int
  - **doseVolume**: Double
  - **contractedCOVID**: Boolean
- Objective
  - What is the dosage that minimizes the rate of contracting COVID.

Usage Pattern 1: Internet of Things

- Dataset: **EngineDailyLog**
  - **engineID**: Int
  - **date**: Date
  - **kmTraveledToday**: Double
- Objective:
  - A train engine needs to be serviced every 30,000km. Which engines need service?

Usage Pattern 1: Aggregation

- Examples:
  - “sum up __, for each__”
  - “average __, by __”
  - “number of __, for __”
  - “biggest __, for each __”
- Pattern
  - (Optionally) Group records by a “Group By” key
  - For each group, compute a statistic
    - e.g., sum, count, average, min, max

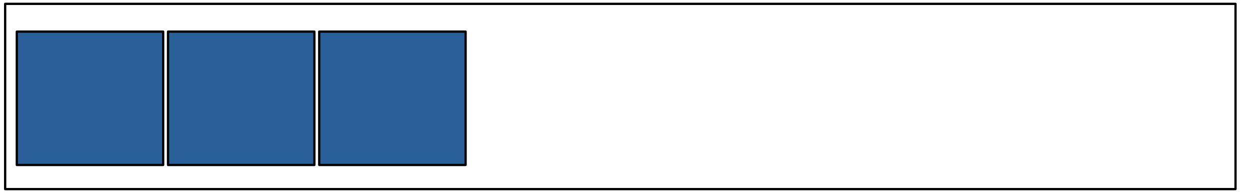
Aggregation

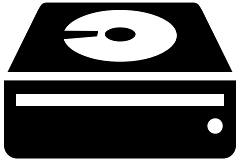
Code Example

Aggregation

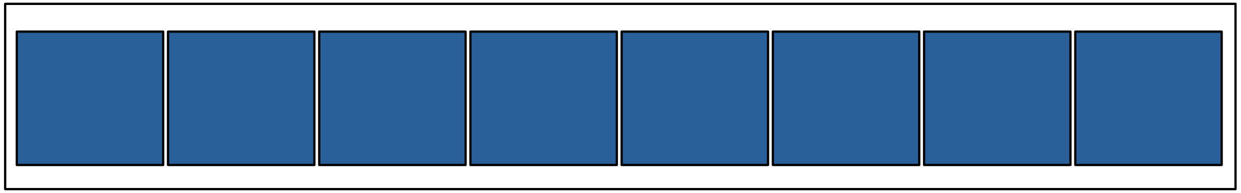
- **Twist 1**: Not enough memory for all of the groups
  - e.g., All of Amazon, Google’s users; LHC particles
  - **Idea**: Use disk for storage
    - **Problem**: Group-by keys not in any specific order, most accesses will be random (slow).
    - **Idea**: O(n) pass to organize the data

Buffered Writer

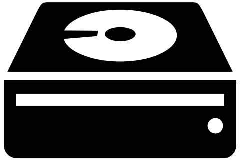




Buffered Writer

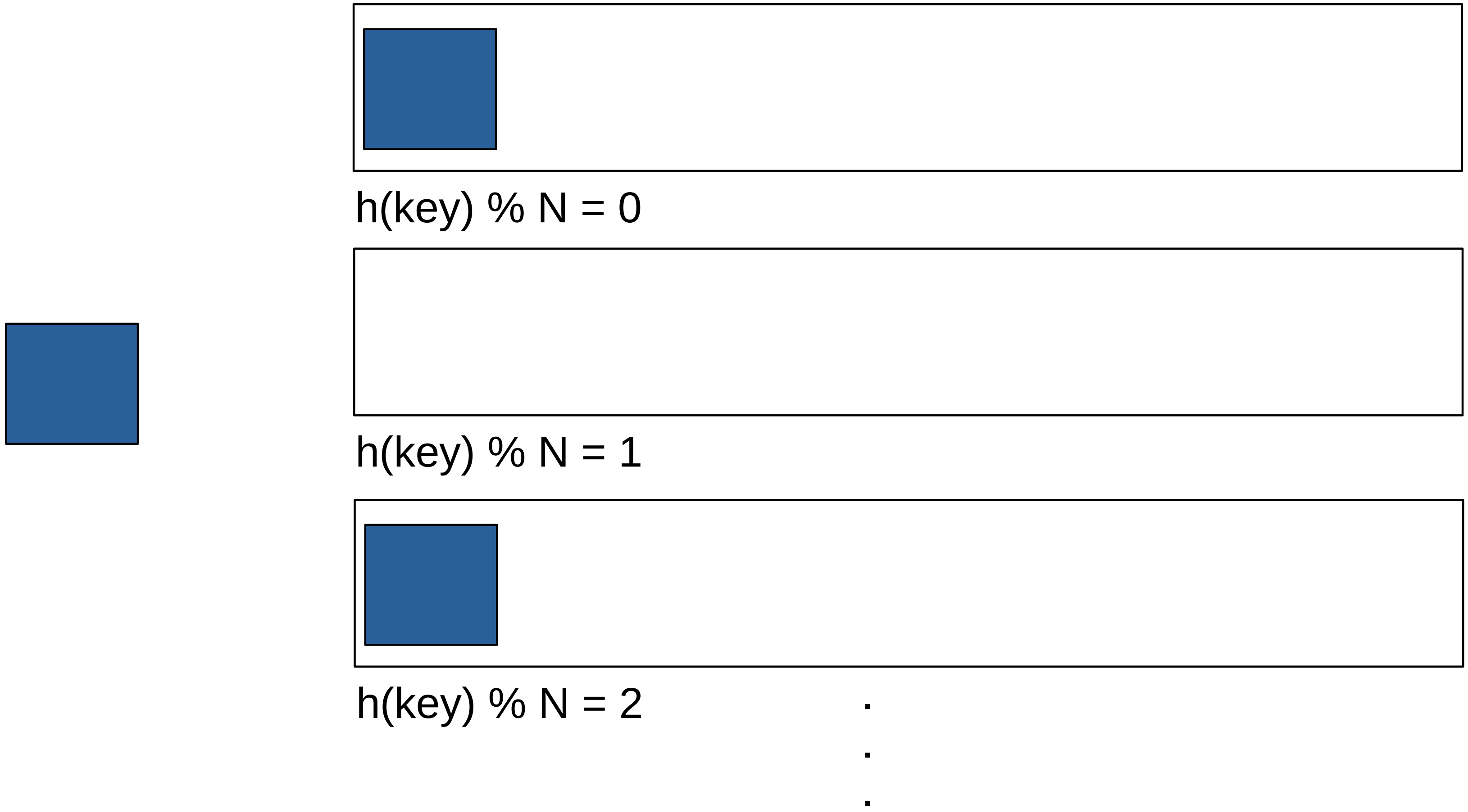


Buffer

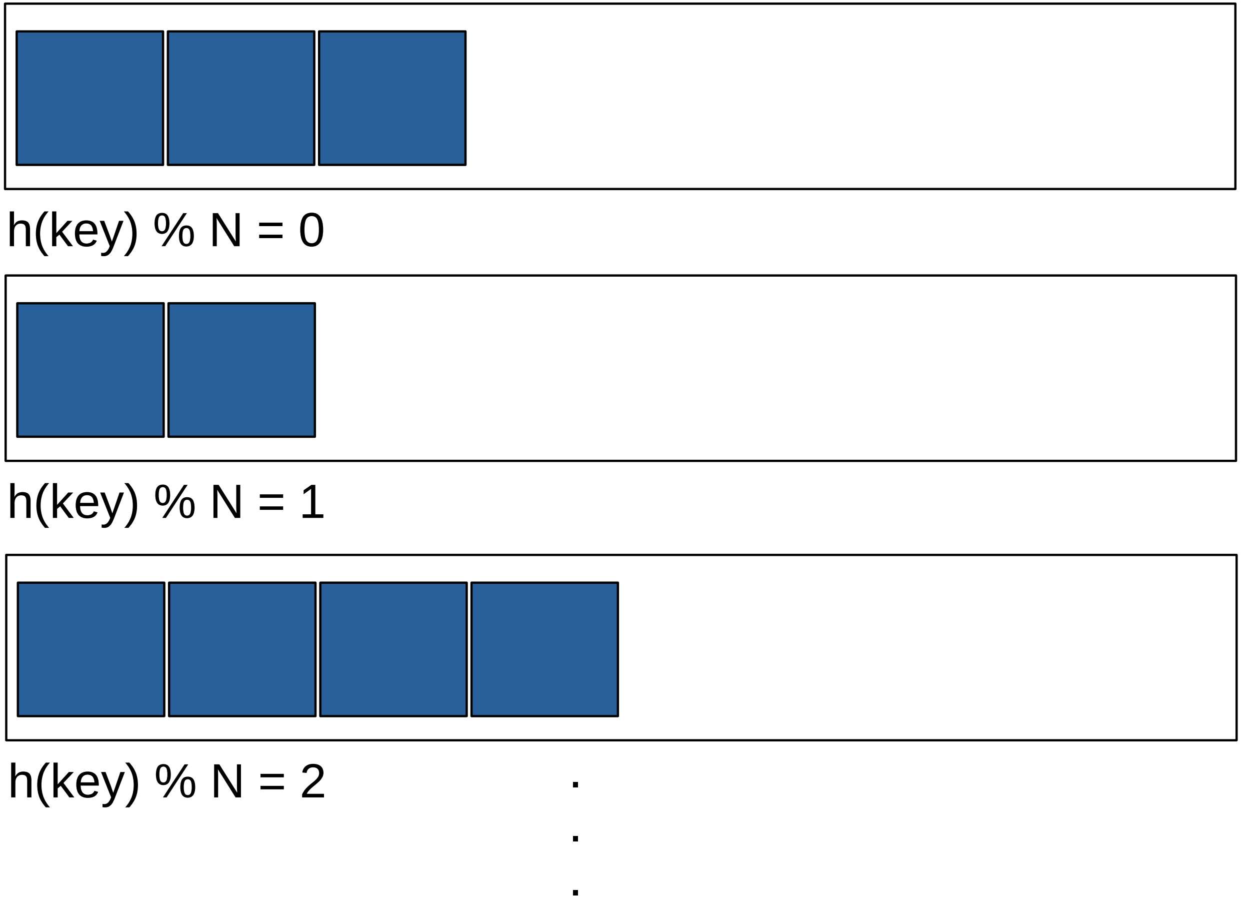


Disk

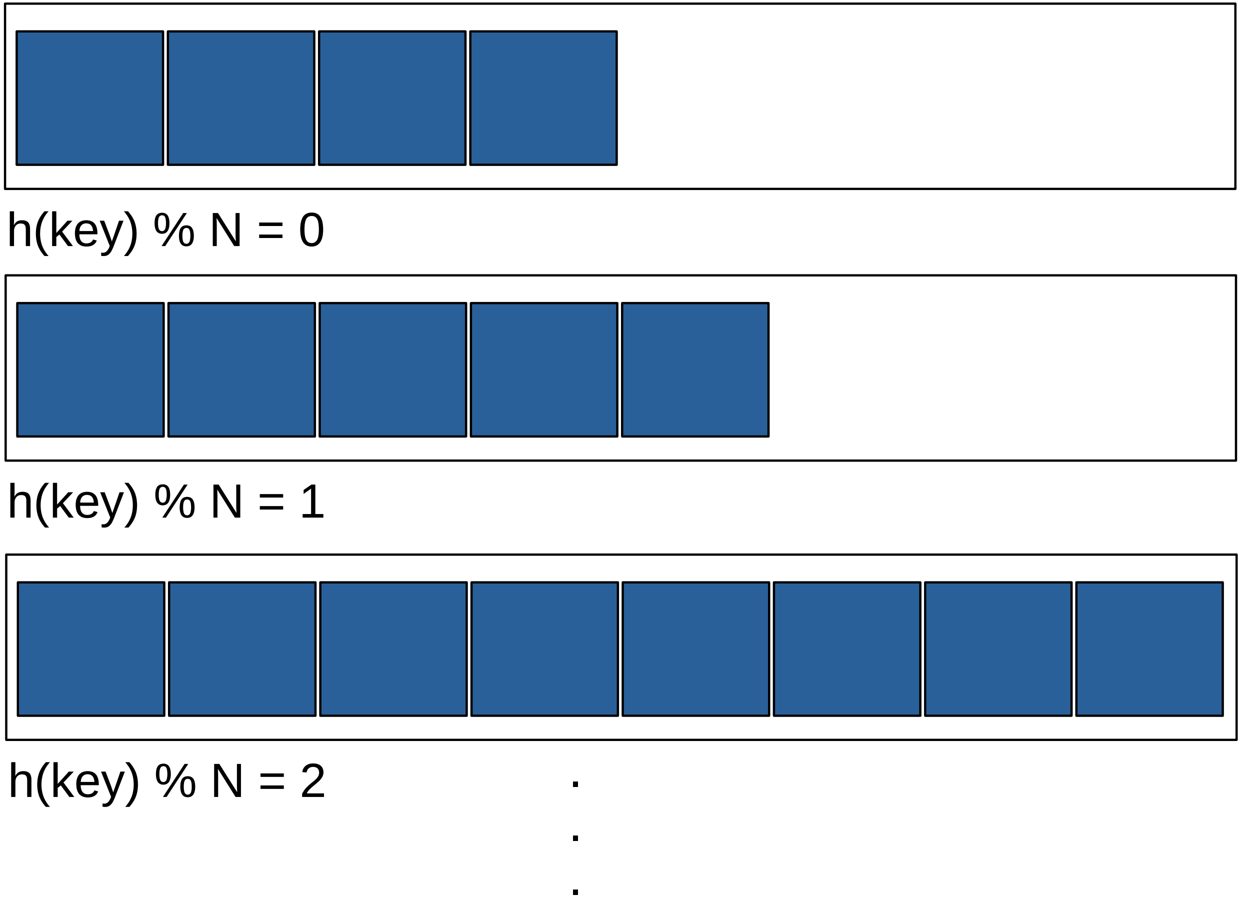
Hash Partitioning



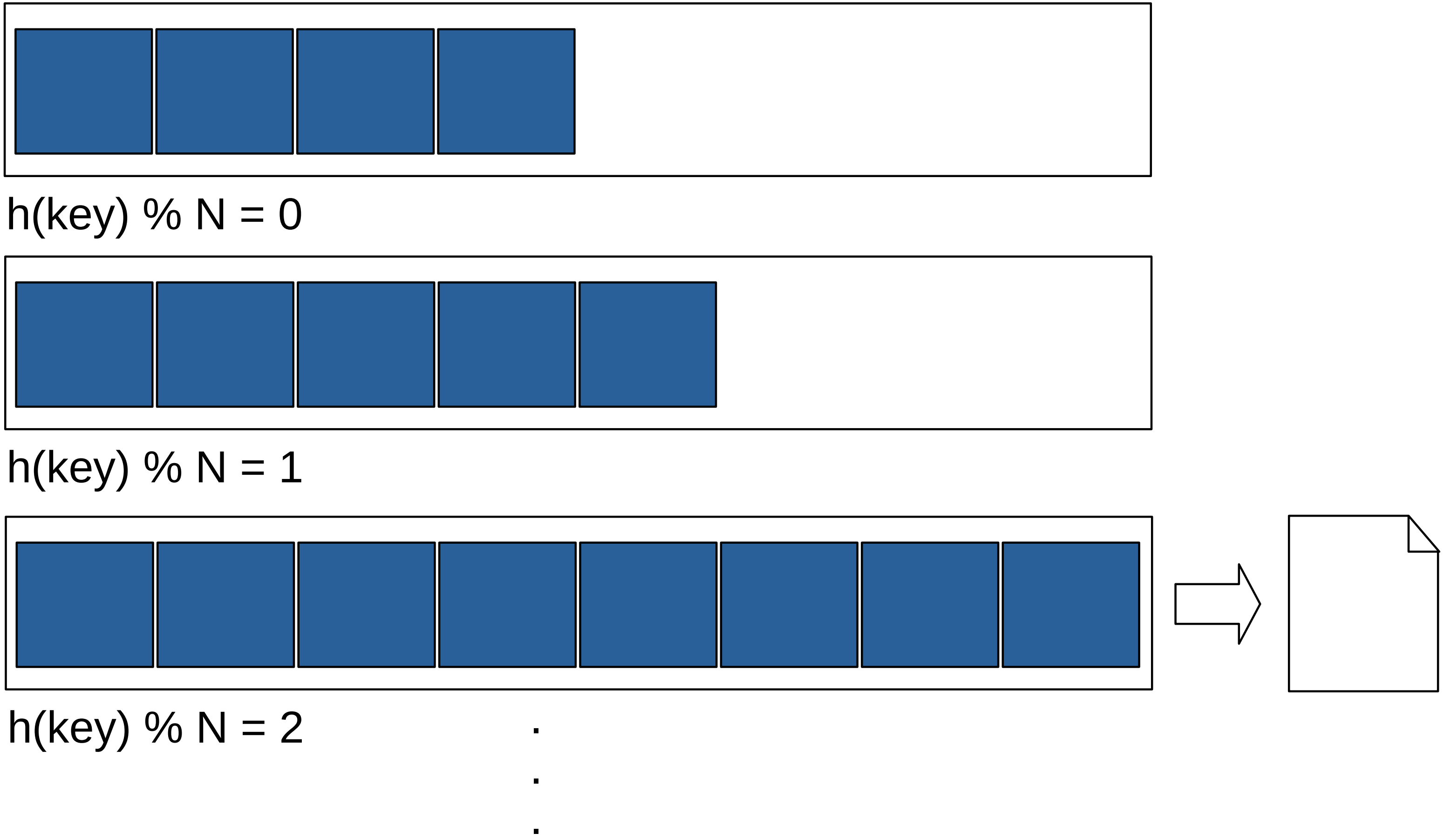
Hash Partitioning



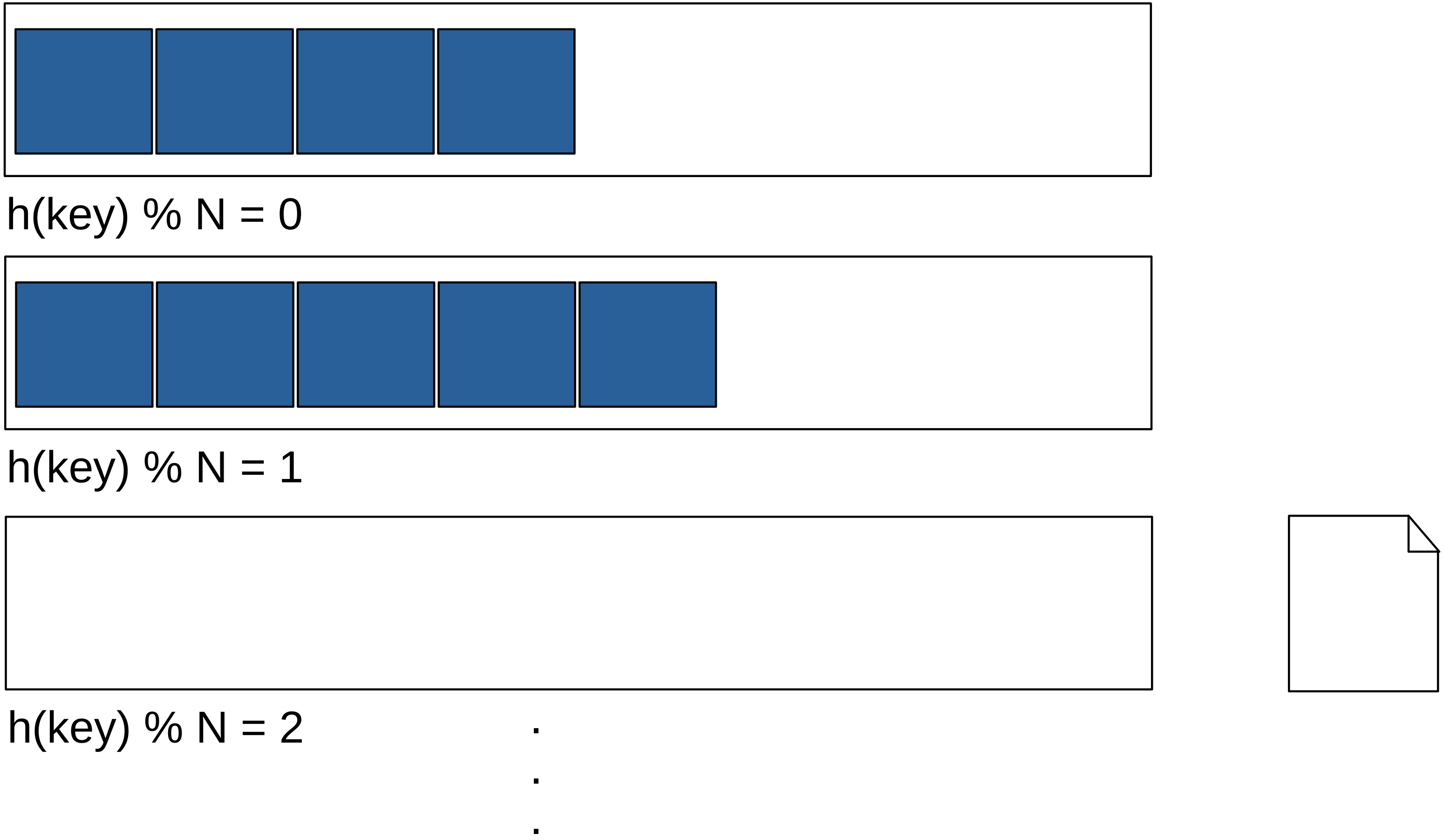
Hash Partitioning



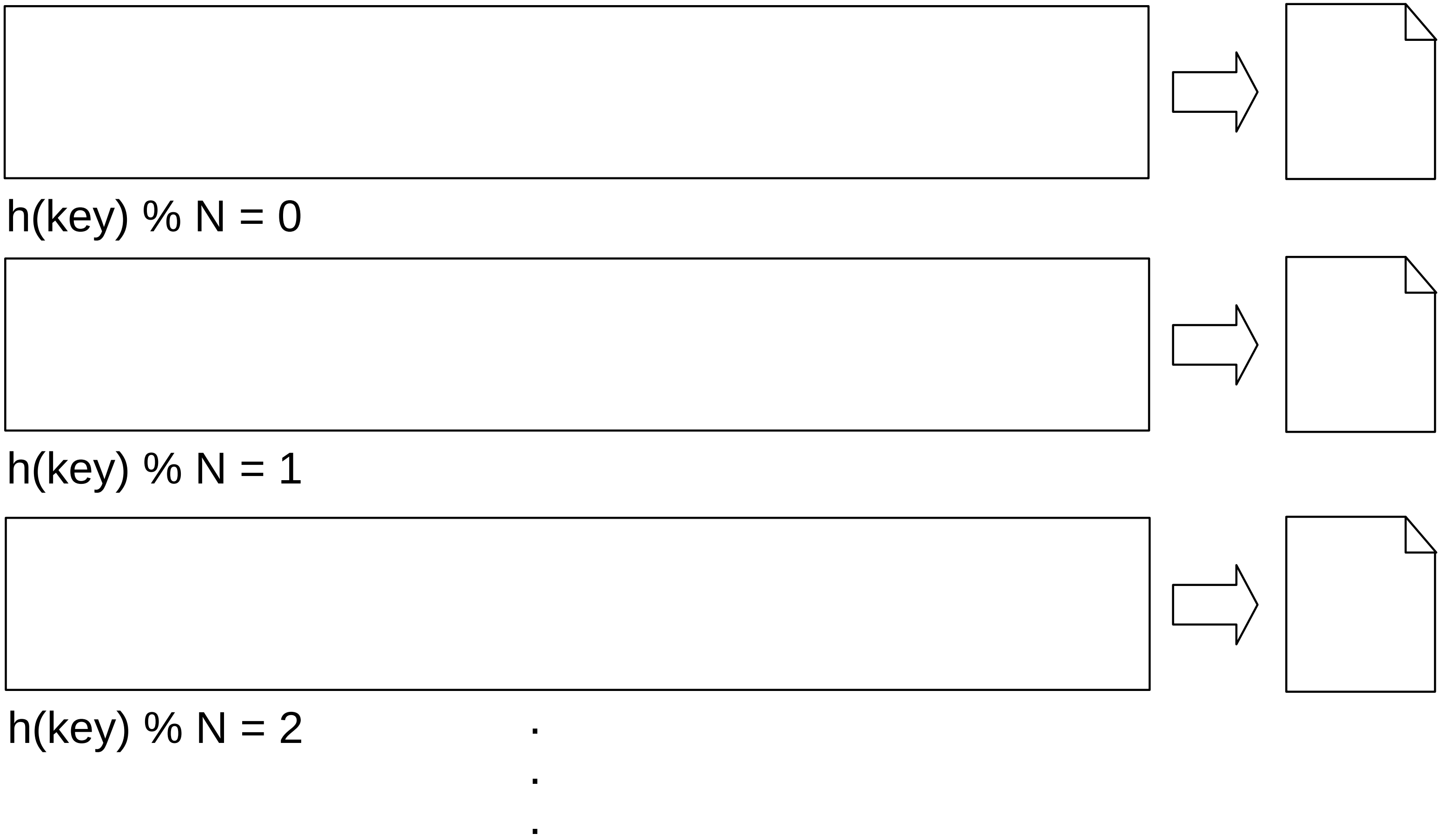
Hash Partitioning



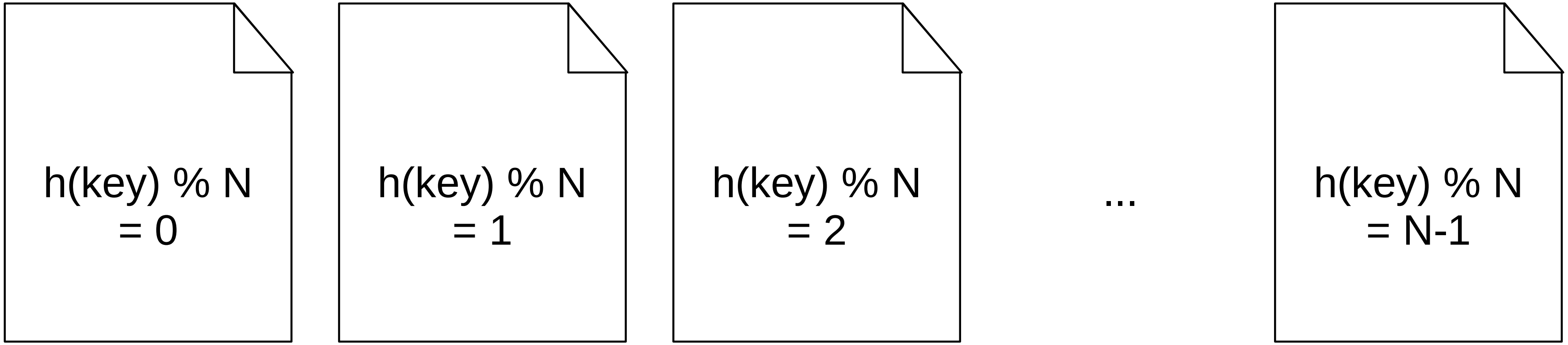
Hash Partitioning



Hash Partitioning

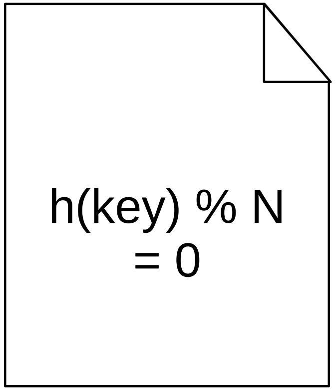


Hash Partitioning



O(n) writes to disk

Hash Aggregation



1. Load file
2. Compute Aggregate In-Memory
3. Repeat for next file

All instances of a key will be in the same file

O(n) reads

Aggregation

- **Twist 2**: Distributed Computation
  - **Idea 1**: Compute Locally, Send Aggregates
  - **Idea 2**: Hash Partition (Shuffle) to each Computer

Usage Pattern 2: MANGA

- Dataset: **Sales**
  - **productID**: Int
  - **date**: Date
  - **volume**: Int
- Dataset: **Pricing**
  - **productID**: Int
  - **price**: Boolean
- Objective
  - Find the 100 products with greatest gross profit (by ID).

Usage Pattern 2: Open Data

- Dataset: **TrafficViolations**
  - **blockID**: Int
  - **infraction**: InfractionType
  - **date**: Date
- Dataset: **PropertyTaxAssessments**
  - **buildingOwner**: String
  - **blockID: Int**
  - **assessment**: Double
- Objective
  - Plot the total taxes collected for a given block against the number of parking tickets issued on that block.

Usage Pattern 2: Science!

- Dataset: **Trials**
  - **patientId**: Int
  - **doseVolume**: Double
- Dataset: **Infections**
  - **patientId**: Int
  - **date**: Date
- Objective
  - What is the dosage that minimizes the rate of contracting COVID.

Usage Pattern 2: Internet of Things

- Dataset: **EngineDailyLog**
  - **engineID**: Int
  - **date**: Date
  - **kmTraveledToday**: Double
  - **locationID**: Int
- Dataset: **Locations**
  - **locationID**: Int
  - **shopSpacesAvailable**: Int
- Objective:
  - A train engine needs to be serviced every 30,000km. Are there more engines that need service at a location than cab be serviced there?

Usage Pattern 2: Joins

- Examples:
  - “combine these datasets”
  - “look up __ for each __”
  - “join __ and __ on __”
- Pattern
  - For each record in one dataset...
  - ... find the corresponding record(s) in the other set
  - Output each pair of matched records

Joins

Code Example

Joins

- **Twist 1**: Too much data to build a hash table in memory
  - **Idea**: Hash-partition both datasets on the join key
- **Twist 2**: Distributed Computation
  - **Idea**: Hash-partition both datasets on the join key
  - **Idea**: Send only relevant data
    - Create a Bloom Filter from the join keys of each dataset

For more...

- If you're interested...
  - **CSE-305**: How to build compilers for languages that can be used to express common data science patterns
  - **CSE-460**: How to organize data to make it easier to find and apply tricks for common data science patterns
  - **CSE-462**: How to build systems that automatically pick the best data structure/algorithm for each data science pattern
  - **CSE-486**: How to build systems that do these sorts of computations "at scale"