Lecture 10:
Database Configuration

Logistics

- Point Solutions App
  - Session ID: **database**
- Programming assignment 2 due on **Sep 24** (Gradescope)
- Exercise sheet 1 due on **Sep 24** (Gradescope)

Recap

- Multi-Threading
- Synchronization
- Fine-Grained Locking
- Debugging

Lecture Overview

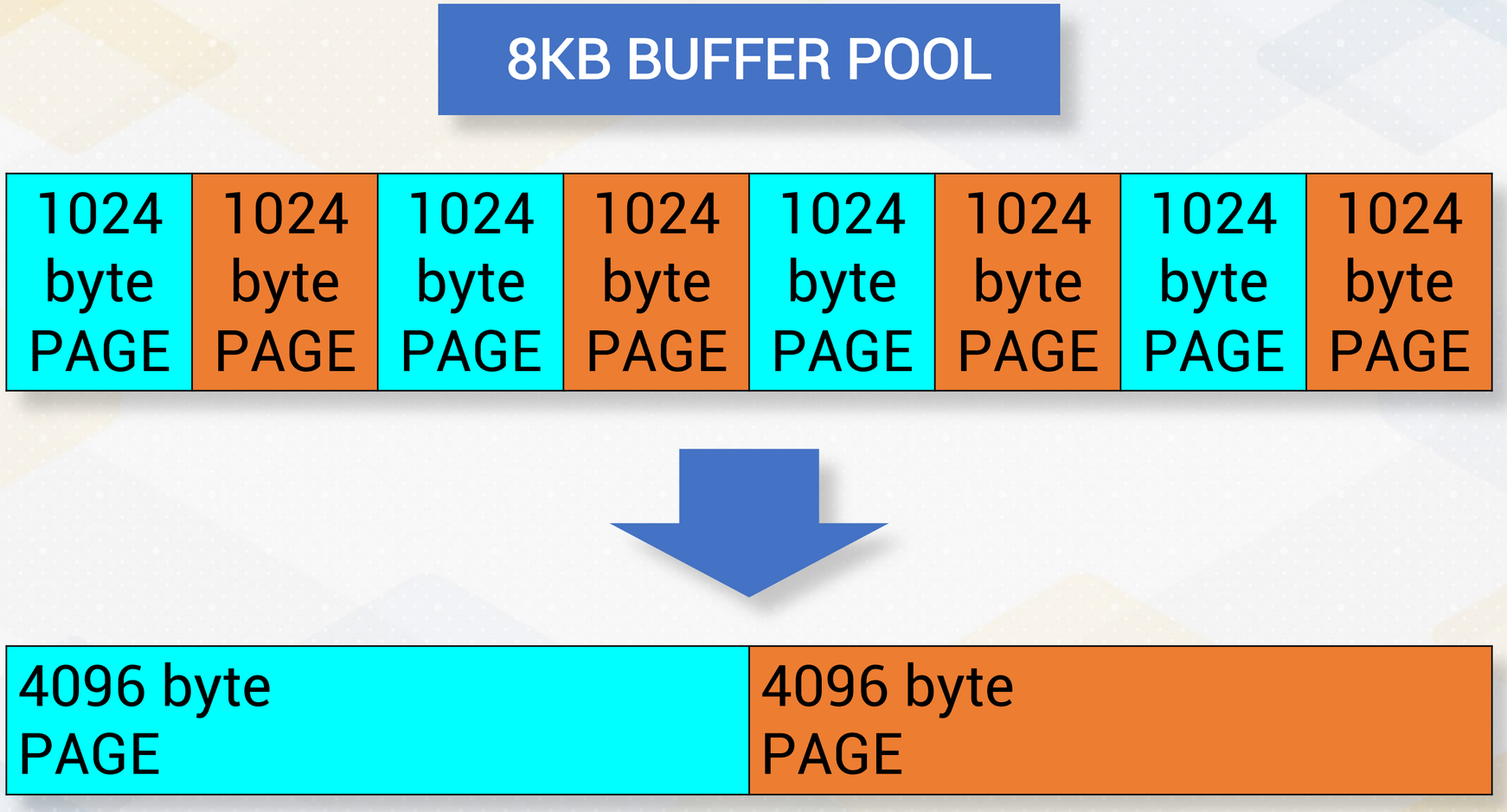
- Database Configuration
- Indexing in C++

Database Configuration

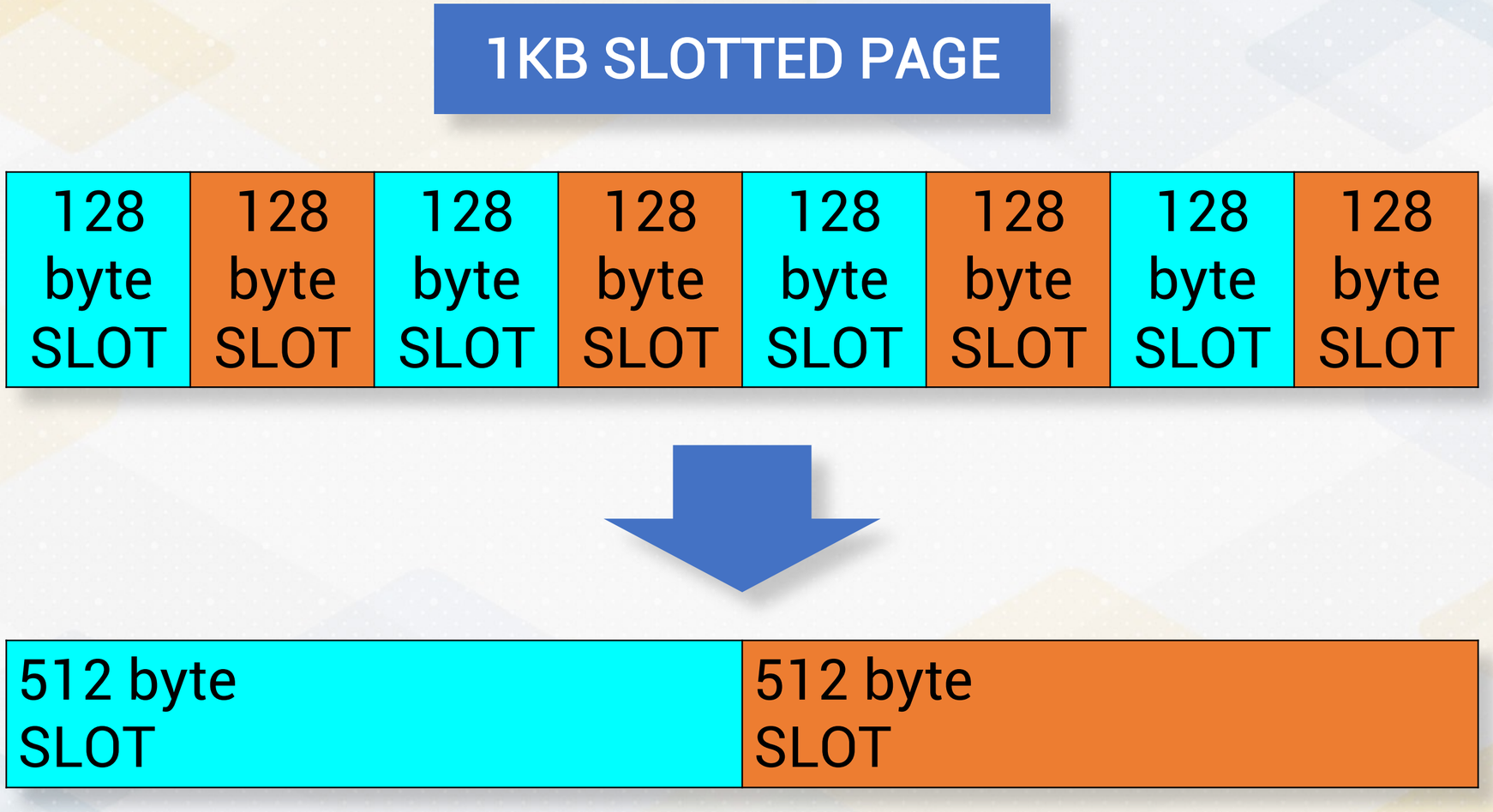
Database Configuration

Parameter	Value
PAGE_SIZE	4096 bytes
MAX_SLOTS	512 slots per page
MAX_PAGES_IN_MEMORY	10 pages
...	...

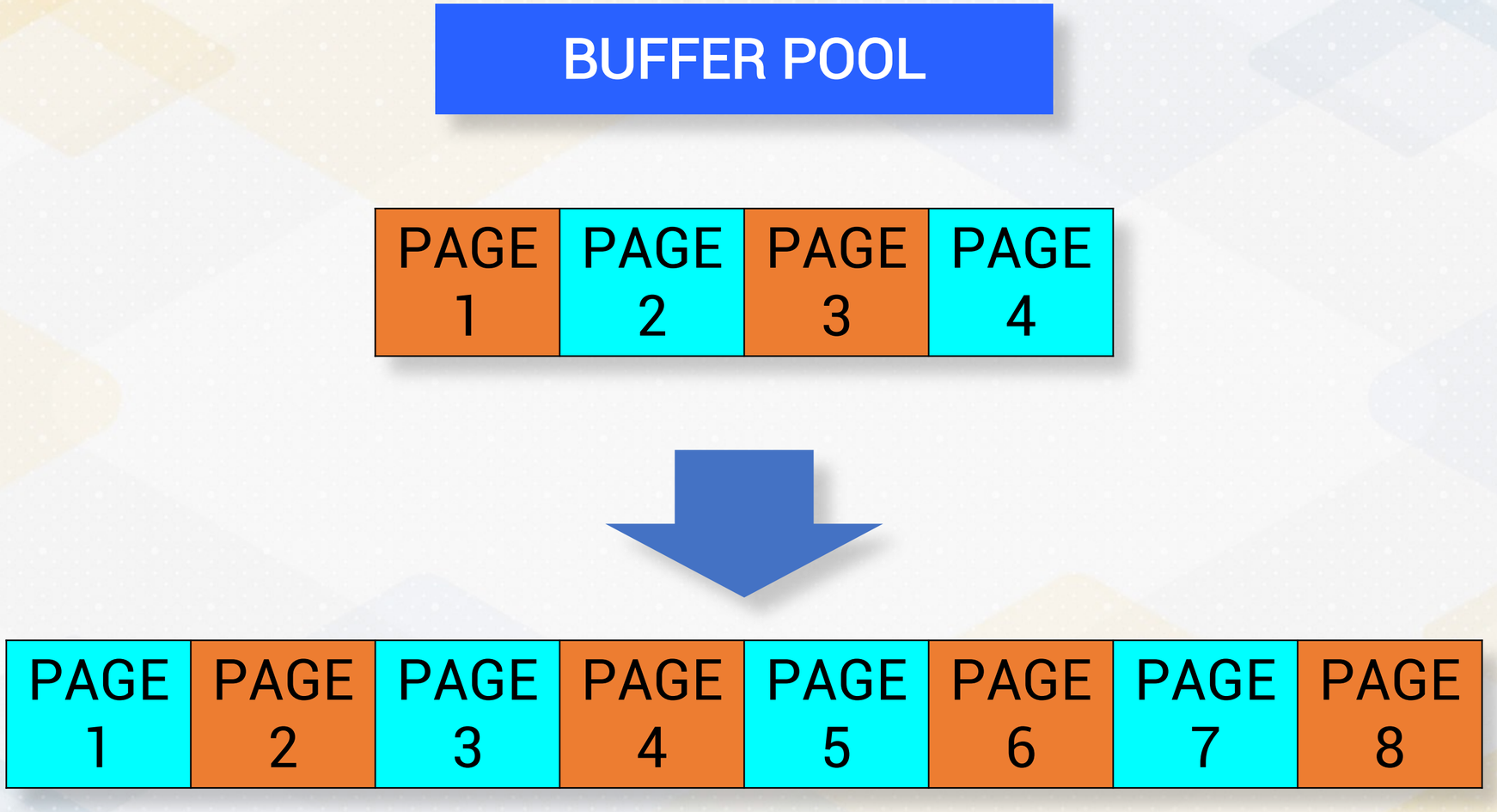
PAGE_SIZE


8KB BUFFER POOL
1024 byte PAGE
1024 byte PAGE
1024 byte PAGE
1024 byte PAGE
1024 byte PAGE
1024 byte PAGE
1024 byte PAGE
1024 byte PAGE
4096 byte PAGE
4096 byte PAGE

MAX_SLOTS

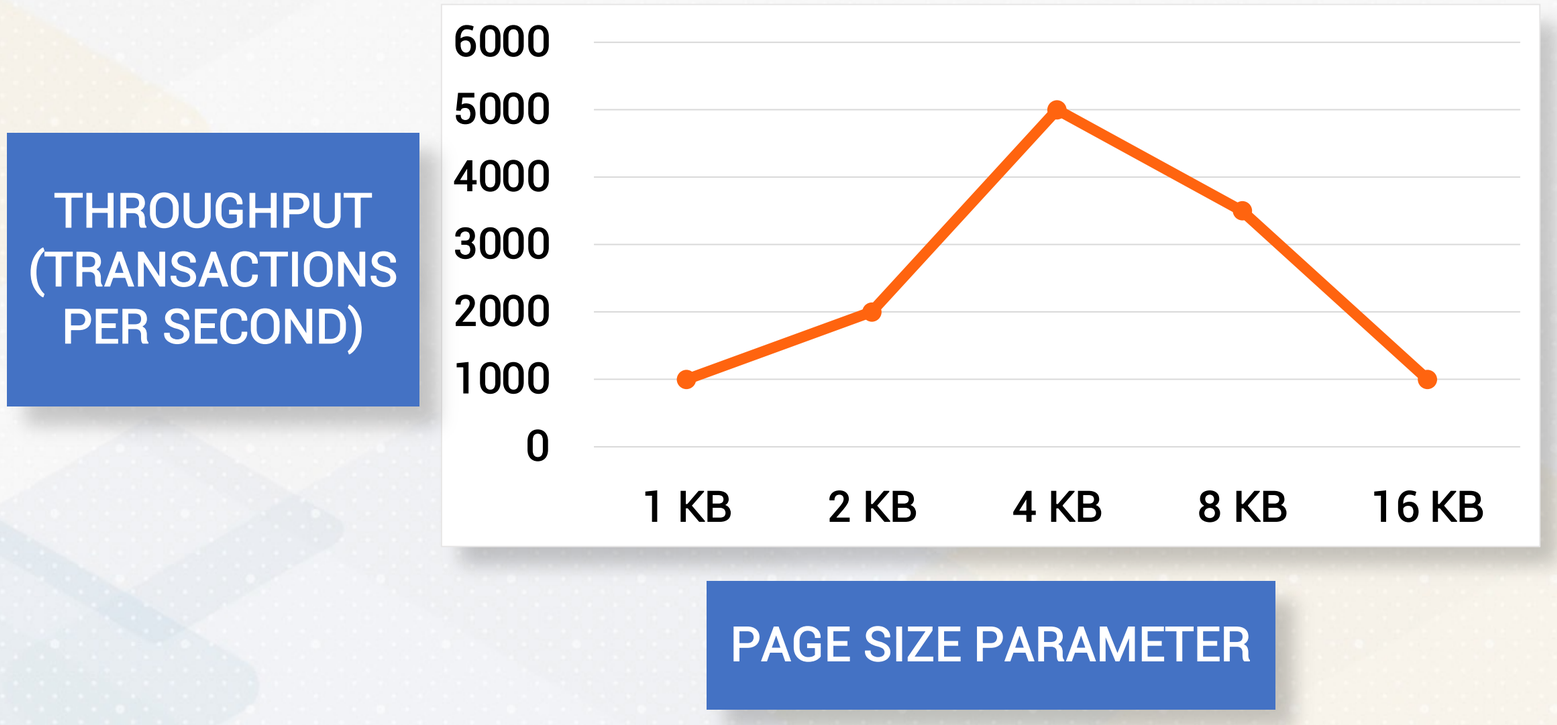

1KB SLOTTED PAGE
128 byte SLOT
128 byte SLOT
128 byte SLOT
128 byte SLOT
128 byte SLOT
128 byte SLOT
128 byte SLOT
128 byte SLOT
512 byte SLOT
512 byte SLOT

MAX_PAGES_IN_MEMORY



Challenges in Configuration Tuning

Most parameters like **PAGE_SIZE** have a “sweet spot”



Challenges in Configuration Tuning

PostgreSQL's default configuration has **low** buffer pool size

System Parameter	Value
BUFFER_POOL_SIZE	128 MB

System Parameter	Value
BUFFER_POOL_SIZE	16 GB

Immutable vs. Mutable Parameters

System Parameter	Mutable or Immutable
PAGE_SIZE	Immutable
BUFFER_POOL_SIZE	Mutable

Cloud Database Tuning

Cost vs Performance

Memory Size	Throughput	Cost	Price-Performance
8 GB	100 transactions per second	$0.05 per hour	$139 per million transactions
16 GB	200 transactions per second	$0.15 per hour	$208 per million transactions

Tools for Configuration Tuning

Automatically suggest configuration based on hardware and application

System Parameter	Value
TOTAL_MEMORY	16 GB
NUMBER_OF_CPUS	16
STORAGE_TYPE	SSD / HDD / SAN
APPLICATION_TYPE	Type 1 / Type 2 / Type 3

Application Type

Application Type	Performance bound by	Database Size Relative to DRAM	Workload Characteristics
Online Transaction Processing (OLTP)	CPU- or I/O-bound	DB slightly larger than DRAM ~ 1 TB	20-40% short transactions, some long transactions and queries
Online Analytical Processing (OLAP)	I/O- or DRAM-bound	DB much larger than DRAM ~ 4 TB	Large complex reporting queries; Also known as data warehouse
Web Application (WEB)	CPU-bound	DB much smaller than DRAM ~ 4 GB	90% or more simple queries

Reinforcement Learning for Configuration Tuning

- Agent interacts with database by trying different configuration settings
- Observes the impact on performance such as throughput

State	Current settings of database parameters and the current workload characteristics
Action	Increase or decrease the value of some parameter
Reward	Increase in throughput or decrease in query response time

Indexing in C++

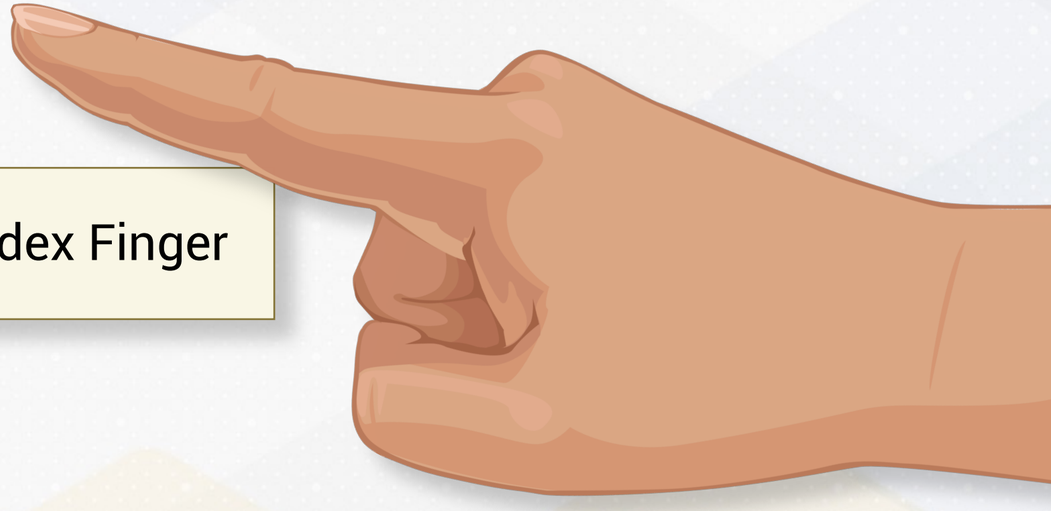
Indexing in Database Systems

Book Index

INDEX

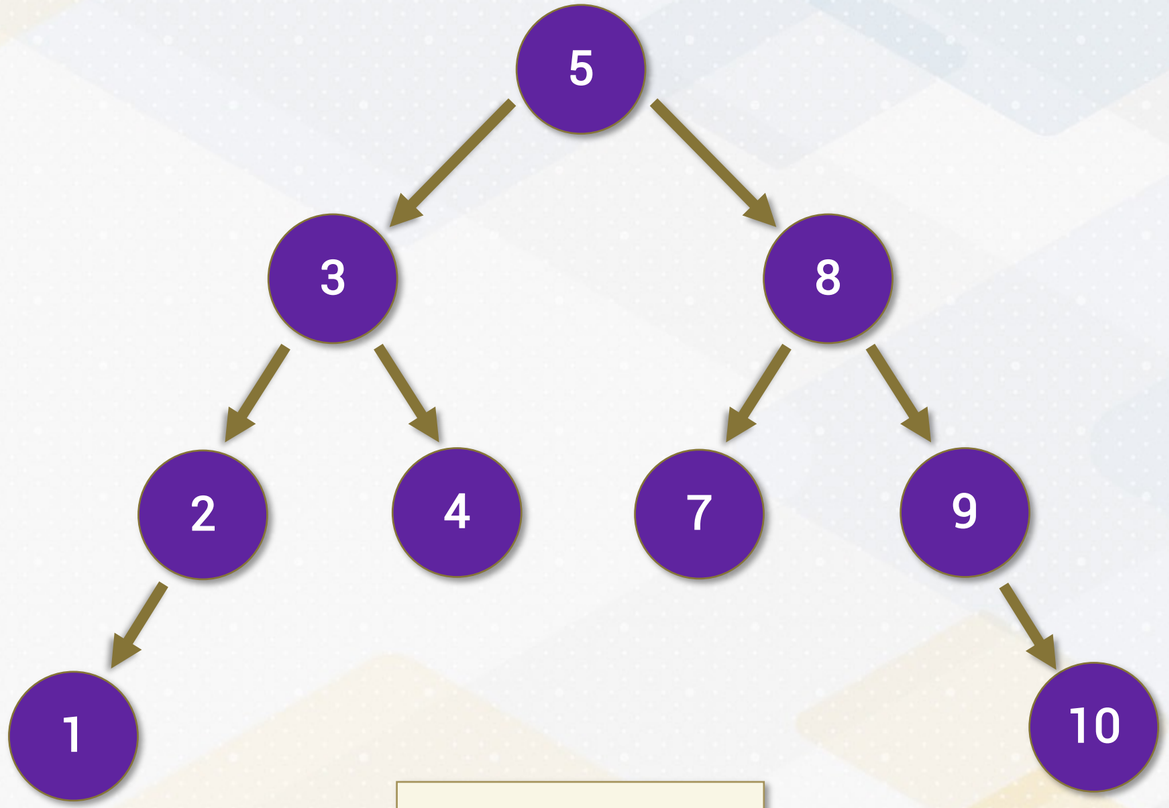
Indiana State Library and Historical Bureau, Public domain, via Wikimedia Commons

Index Finger



Indexing in C++: ordered_map

Key	Value
1	Apple
2	Banana
3	Cherry
4	Date
5	Elderberry
6	Fig
7	Grape
8	Honeydew
9	llama
10	Jackfruit



O (log N)

Indexing in C++: unordered_map

Key	Value
2	Banana
6	Fig
1	Apple
3	Cherry
4	Date
5	Elderberry
8	Honeydew
7	Grape
10	Jackfruit
9	llama

Avg Case	Worst Case
O (1)	O (N)

map vs unordered_map

	MAP	UNORDERED MAP
Ordering	Sorted based on key	Not sorted based on key
Implementation	Binary Search Tree	Hash Table
Average Case Time	O(log N)	O(1)
Worst Case Time	O(log N)	O(N)

HashIndex in BuzzDB

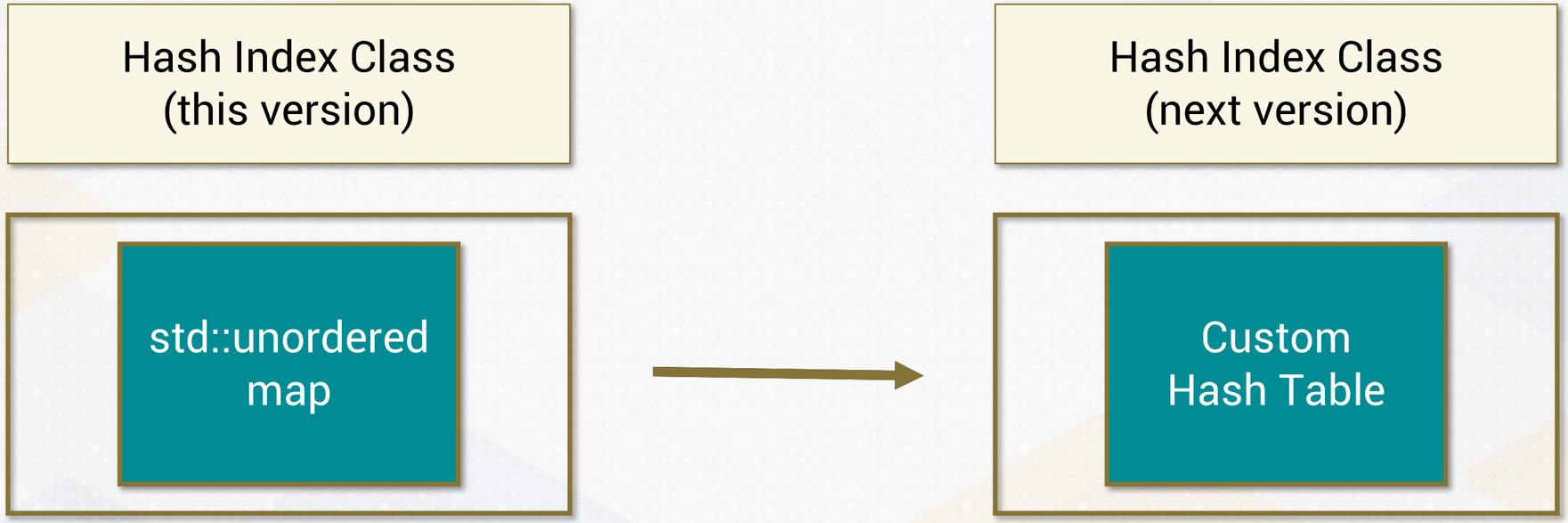
```
class HashIndex {
private:
std::unordered_map<int, int> hash_index;
public:
void insertOrUpdate(int key, int value) {
hash_index[key] = value;
}
int getValue(int key) const {
auto it = hash_index.find(key);
return it != hash_index.end() ? it->second :
-1;
}
};
```

Integration of HashIndex in BuzzDB

During the scanning phase, each tuple's key-value pair is inserted into the HashIndex.

```
void BuzzDB::scanTableToBuildIndex() {
// Iterating over pages and tuples
for (size_t page_itr = 0; page_itr < num_pages; page_itr++) {
// Extract key-value pairs and insert them into the index
int key = loadedTuple->fields[0]->asInt();
int value = loadedTuple->fields[1]->asInt();
index.insertOrUpdate(key, value);
}
}
```

Custom HashTable



Conclusion

- Database Configuration
- Indexing in C++