ECE297 Project

Storage Server for Grocery Store Inventory Management

Xiangkun Li
Tanzim Mokammel
Wina Ng

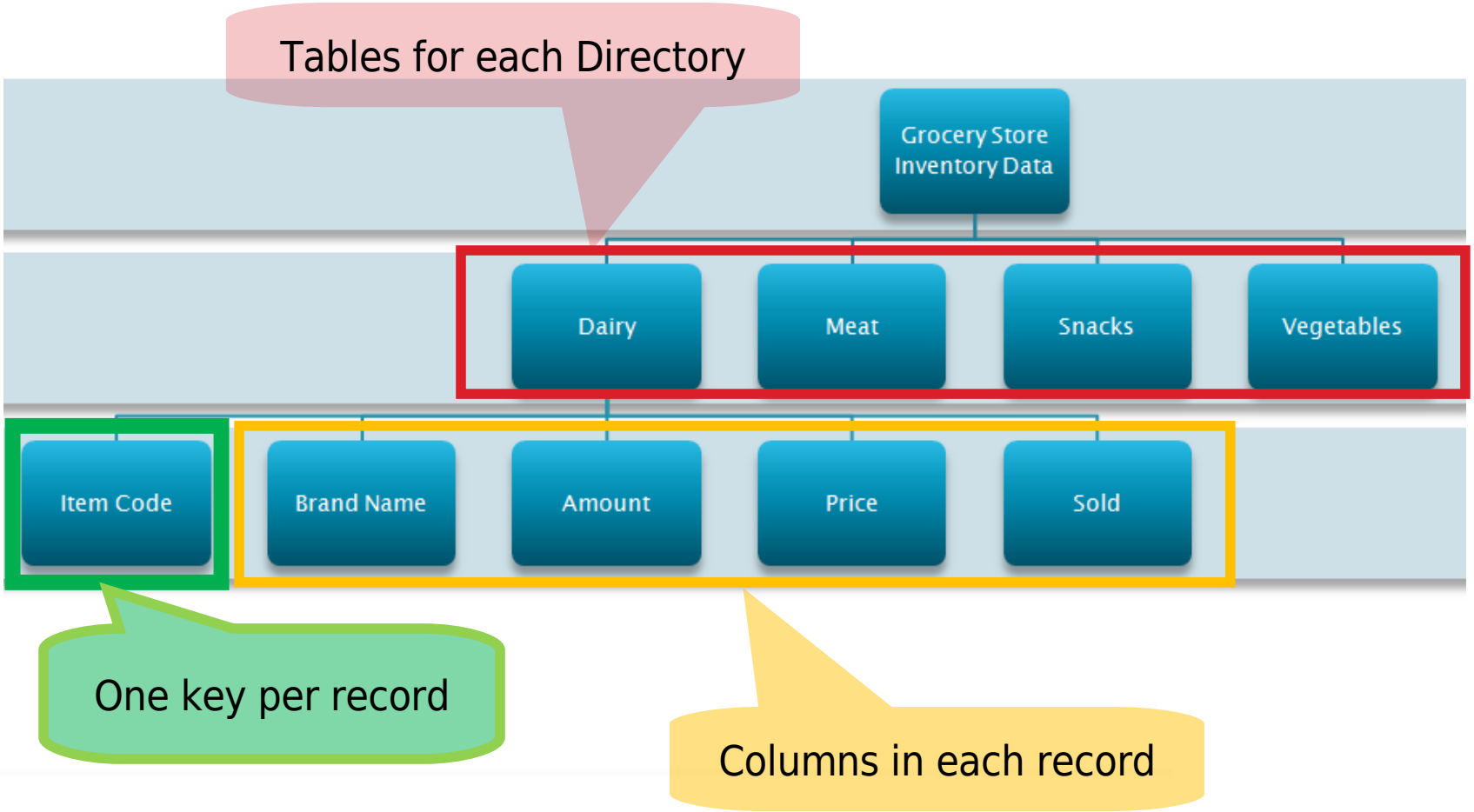
Motivation: Importance of Keeping Inventory

- It is a common practice in businesses to organize data i.e. sales, revenues, stock information
- It helps to identify patterns in demands, price changes to help make business decisions e.g. Reordering
- Grocery store stocks in small companies
  - Organize inventories by product type, barcodes, brand names, stock amount, prices etc.

Motivation: Economic Viability

- Grocery stores sell daily needs
  - In 2003, the average US family spent $3,305 per year at grocery stores
  - There is a lot of transactions between grocery stores and the population
- According to Statistics Canada, 90% of the operating costs for grocery stores involve stock management

Applying the Server to Grocery Store Inventory Management



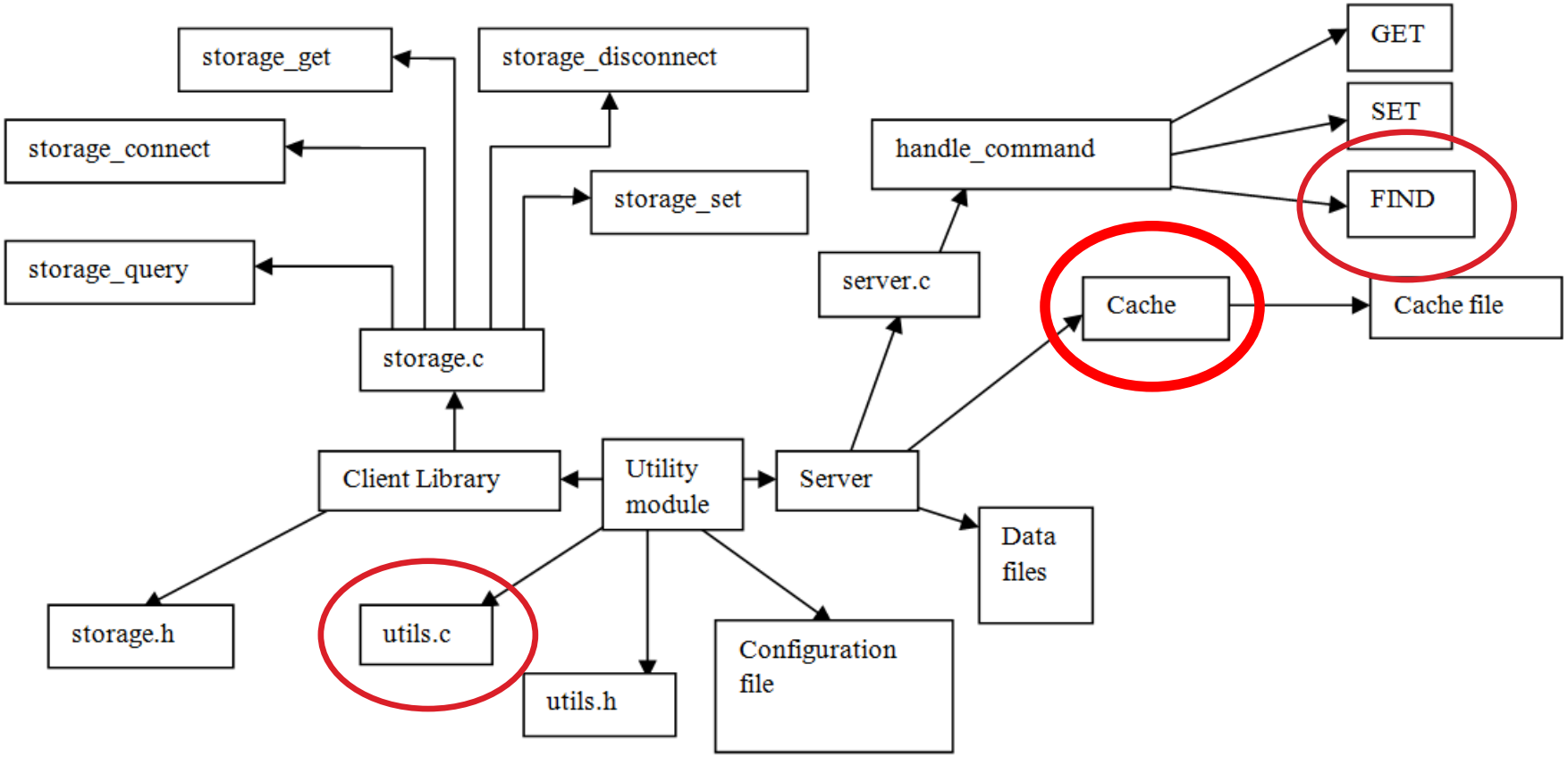
Example: Snacks Directory

Item Code	Stock Information			
	Brand Name	Amount	Price per Unit	Sold
172639	Doritos	5382	$1.49	346
273649	Cheetos	4276	$3.00	273
621983	Humpty Dumpty	3740	$2.39	298
273648	Lays	3846	$4.99	698

Managing the Server's Internal Operations

- The use of arrays of structures throughout the server to manage and manipulate data
- 3 main areas of implementations:
  - Used to hold configuration parameters
  - Used to store and operate on predicates
  - **Used to set up cache structure**

Managing the Server's Internal Operations



Arrays of Structures:

- Pros:
  - Easy to implement and debug
  - No memory management
  - Quick access by index
  - Relatively fast to loop through
- Cons:
  - Initial size must be known (static)
  - May waste memory

Cache Implementation with Nested Array of Structures

(Array for all tables)

allCaches [0]	allCaches [1]	allCaches[2]	allCaches [3]	...
Dairy	Meat	Snacks	Vegetables	

Table Name: Dairy.txt
Number of Items: 50
CacheEntry [0]
CacheEntry[1]
⋮

(Structure for a table)

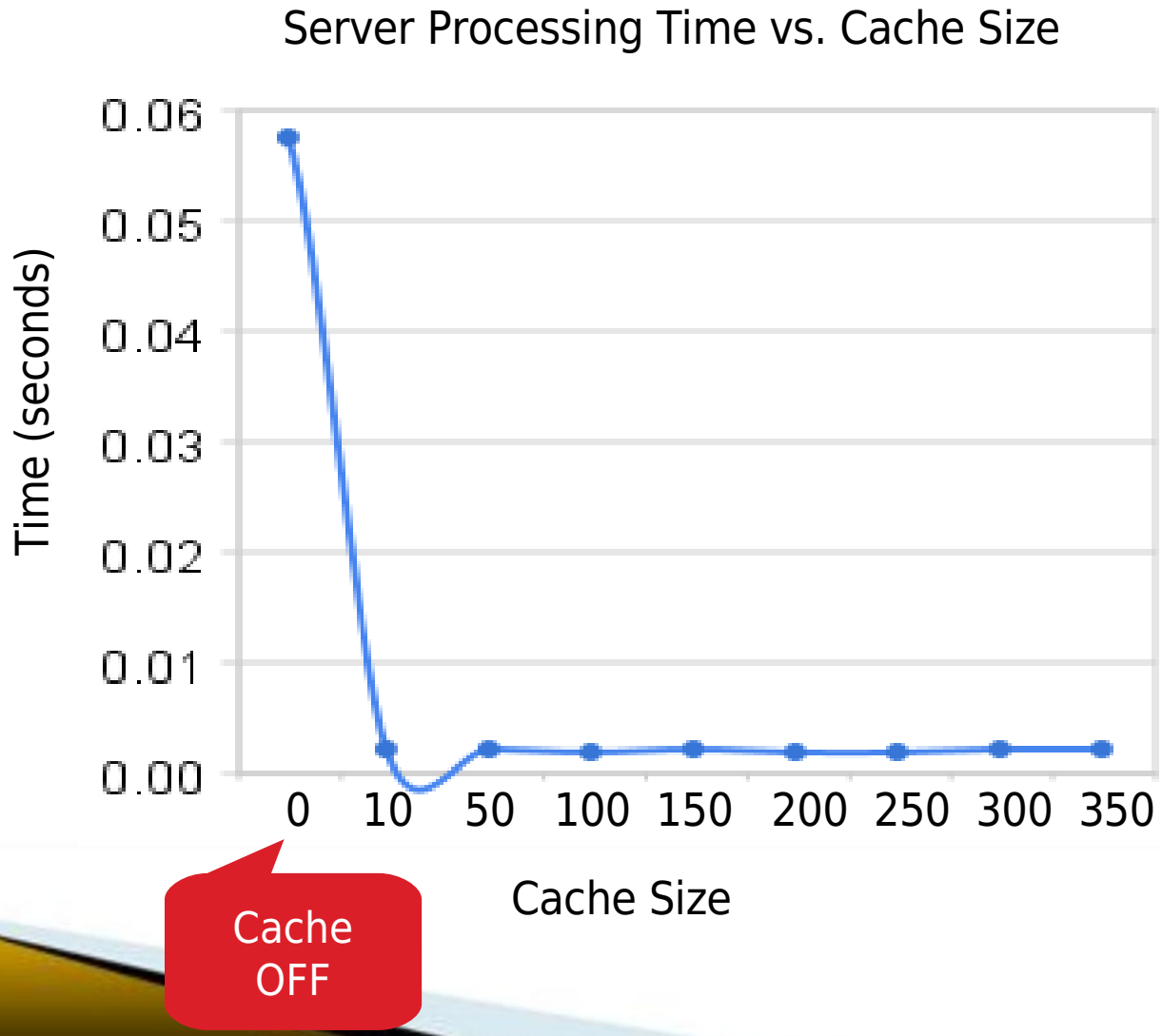
Key: 99876234112
Record: Brand Name: Neilson, Amount: 300, Price: 3.99, Sold: 231
Version: 5
Access Frequency: 12

(Structure for a record)

Main Alternative: Linked Lists

- Pros:
  - Dynamic size
  - No wasted memory
- Cons:
  - Slow to traverse (required by query and cache searches)
  - A lot of pointer manipulation and memory allocation
    - Higher rate of crashes

Evaluating Cache Performance



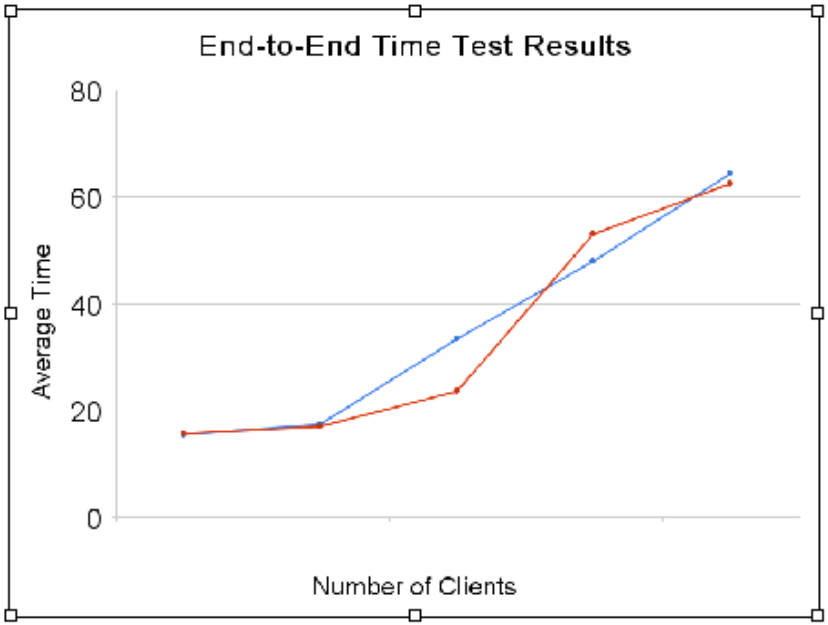
- Enabling the cache significantly improves the server processing time

- Conditions:
  - 100 calls per test
  - Clustered calls with 10 records

Evaluating the Cache With Multiple Clients

- End-to-End execution time
  - Allows us to average the time among multiple clients
  - Calculates actual execution time the client experiences
- The main variables in the tests are:
  - The cache policy
  - The number of clients
  - The operations on the data table

End-to-End Tests

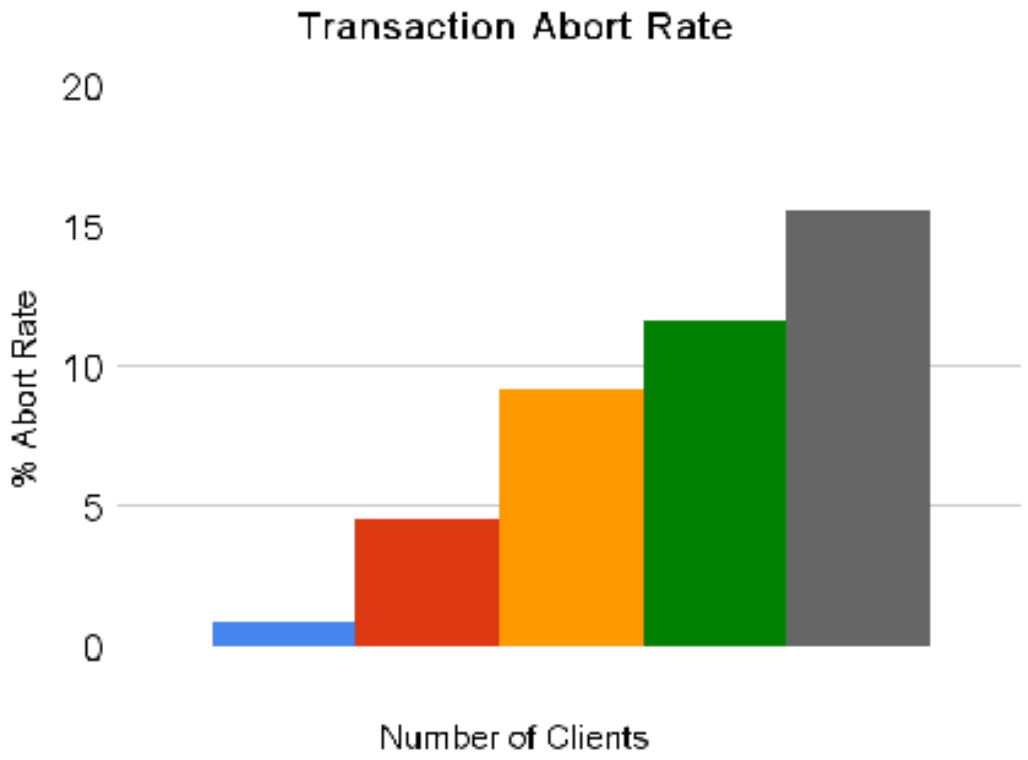


- Number of clients connected: 1 to 5
- The average execution time increases with more connections
- On average, caching improves execution times with multiple clients
- GET and SET calls increase more than 50% with two or more connections

Transaction Abort Rate Tests

- To ensure accuracy of inventory records, we implemented transactions.
- We tested this feature varying the number of clients to calculate the average abortion rate.

Transaction Abort Rate Tests



- Number of clients connected: 1 to 5
- The average abortion rate increases linearly with more connections

Lessons Learned: Professional Writing and Documentation

- Active voice
- UML Diagrams
- IEEE Format
- Software design document format

Lessons Learned: Self-Teaching

- Debugging tool: DDD
- Parsing tools: Awk and Sed
- Select system call to support multiple clients
- Built-in functions
  - Check.h - for unit tests
  - Strtok() - for parsing

Lessons Learned: The Design Process

- Incremental design
  - Frequent debugging and testing (Check)
- Identifying metrics
- Splitting up coding tasks
  - Based on each member's expertise and experience
- Reintegrating individual parts into a final working storage server

Attribution Table

Milestone	Xiangkun Li	Tanzim Mokammel	Wina Ng
1 (Code)	1. Server	1. Client Library 2. Configuration File	1. Design Decisions
1 (Document)	1. Software Architecture 2. Design Decisions	1. Software Architecture 2. Design Decisions	1. Design Decisions 2. Test Plan 3. Software Architecture 4. Editing and Formatting
2 (Code)	1. Server	1. Configuration File 2. Client Library	1. Client Library 2. Unit Tests
2 (Document)	1. Software Architecture 2. Design Decisions	1. Software Architecture 2. Design Decisions	1. Use Cases 2. Test Plan 3. Editing and Formatting
3 (Code)	1. Server 2. Performance Metrics	1. Configuration file 2. Performance Metrics 3. Workload	1. Server Configurations to Evaluate 2. Performance Metrics
3 (Document)	1. Software Architecture 2. Design Decisions 3. Performance Evaluation Report	1. Executive Summary 2. Software Architecture	1. UML Diagrams 2. Use Case Motivation 3. Use Cases 4. Performance Evaluation Plan 5. Performance Evaluation Report 6. Editing and Formatting
4 (Code)	1. Transactions	1. Multiple Clients 2. Performance Evaluation (Transaction Abort Rate)	3. Performance Evaluation (End-to-End Delay)
4 (Document)	1. Software Architecture 2. Design Decisions	1. Executive Summary 2. Assignment 4 Addenda: Design Considerations 3. Performance Evaluation Report	1. Use Cases 2. Performance Evaluation Plan 3. Performance Evaluation Report 4. Editing and Formatting