Distributed DB for IoT Systems

Project Presentation

Nicola Ruaro, Atefeh Mohseni

Overview

1. Introduction
2. System Architecture
3. Security Considerations
4. Profiling and Performance Considerations

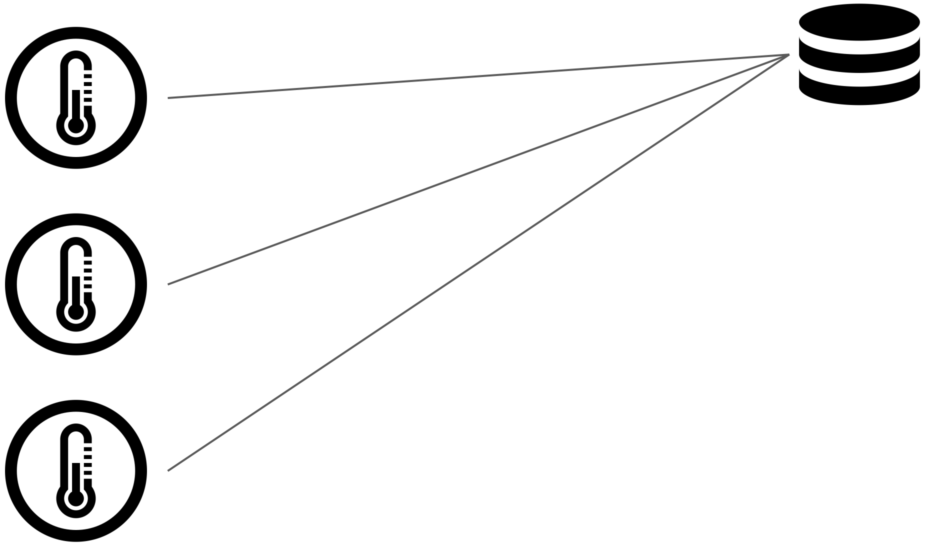
Introduction

What is a distributed DB?

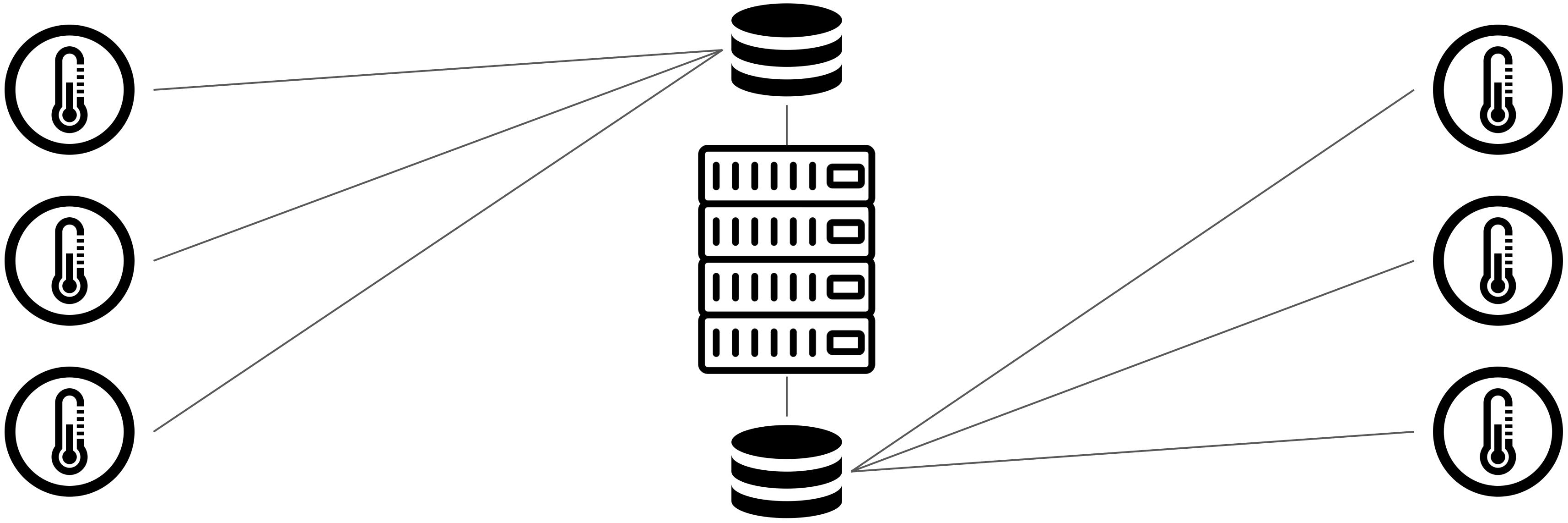
What is a Distributed DB?

- Data is stored across different physical locations
  - Management of data with different level of transparency
  - Increased Reliability and availability
  - Easier Expansion
  - Improved Performance

Scenario



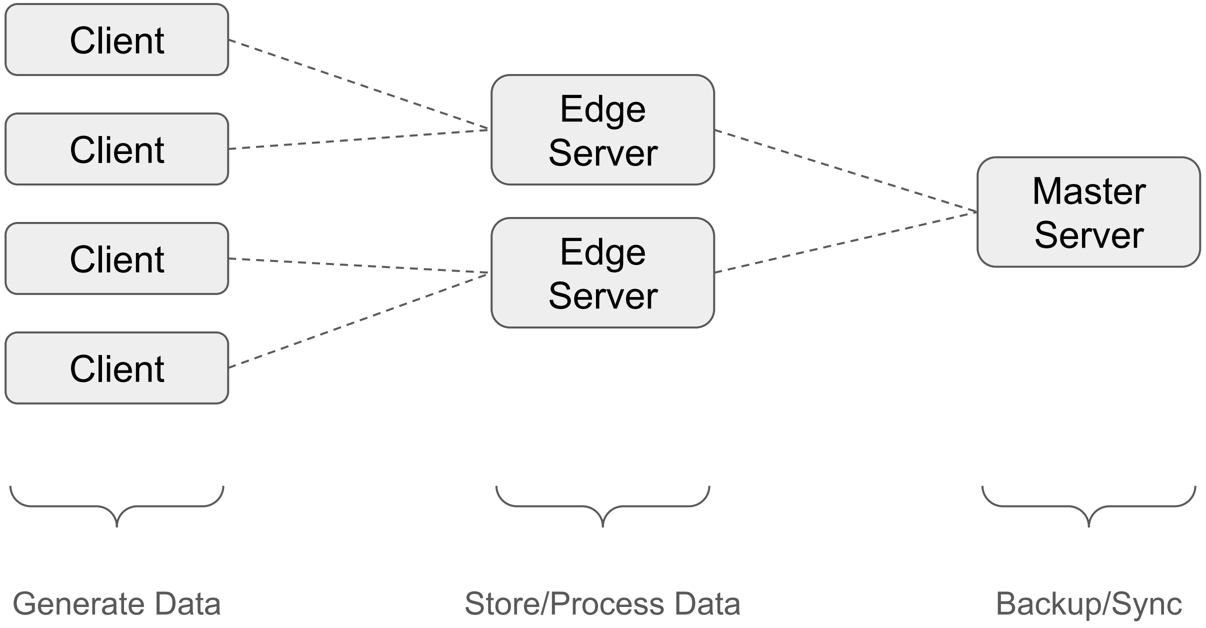
Scenario



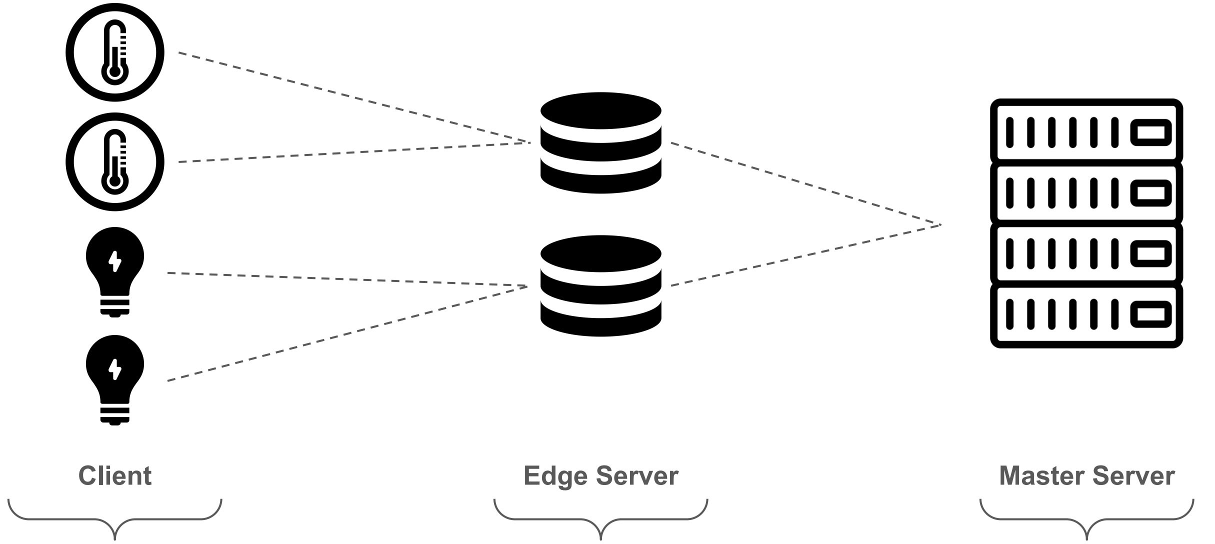
System Architecture

How did we design this system?

Architecture Overview



Scenario - Smart / Sustainable Home

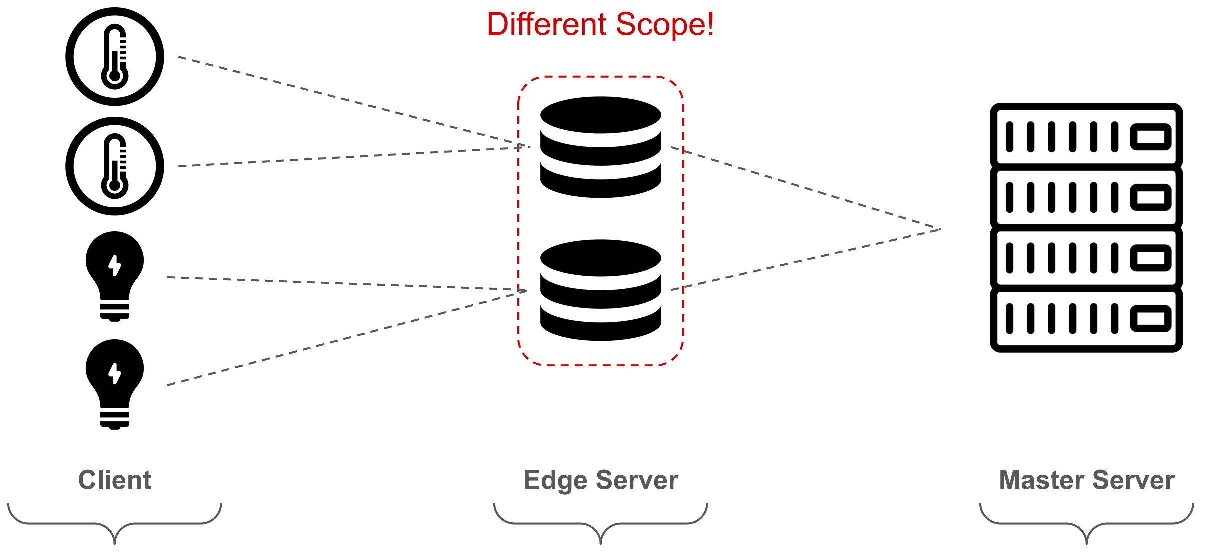


Generate Data

Store/Process Data

Backup/Sync

Scenario - Smart / Sustainable Home



Generate Data

Store/Process Data

Backup/Sync

System Architecture

Implementation Details

Client - Local Data Store

- Queue for PUT and DELETE requests
- Guarantees reliability
  - Network failure
  - Program failure
- The main thread is not locked during the network communication
- Enables some optimization
  - Create batches for different requests
  - Ignore requests if they are immediately overwritten

Edge Servers

- Implement a database with key-value format
    - Support read/write/delete queries
- REST API for connection
    - edge-server to client
    - edge server to master-server
- Runs queries received from client
    - Put (for write query), Get (for read query), Delete (for delete query)
- Data persistence
    - Backup database with master server

Master Server

- Stores edge server backup files
- Checks the status of edge servers (passively)

Security Considerations

Where do we guarantee more security?

Security - Confidentiality

- Access Control
  - Check client’s credentials before executing the query
    - HTTP Basic Authentication
    - bcrypt
- Data Encryption
  - Client-Edge Server and Edge Server-Master Server
    - SSL (HTTPS)
    - Certificates

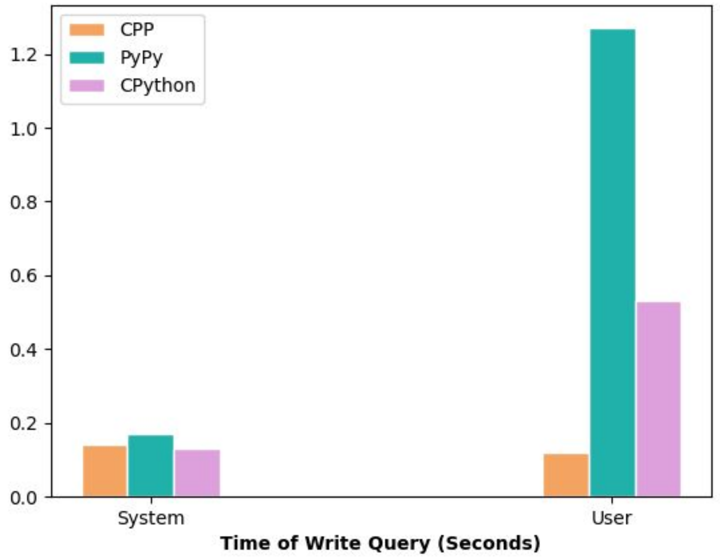
Profiling

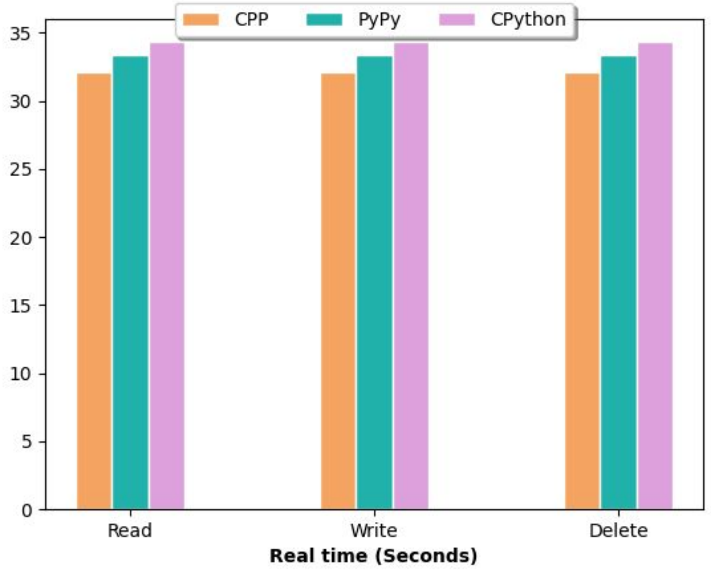
C++ / Python Comparison

Profiling

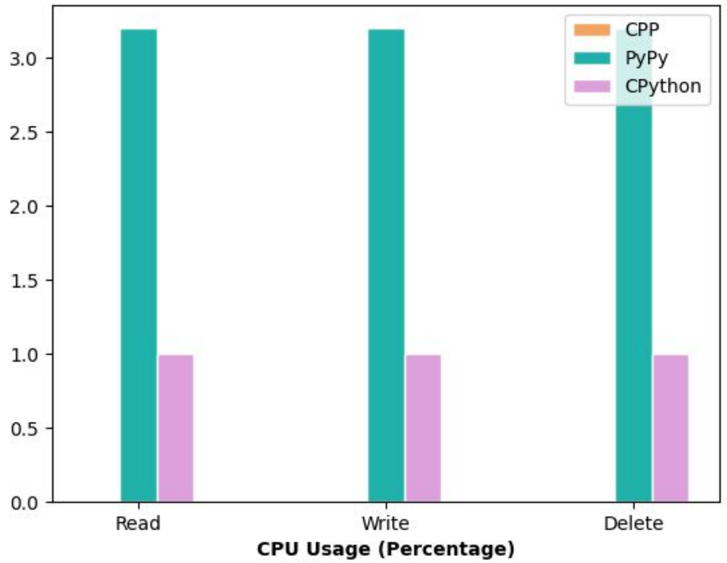
- Offline profiling of resource usage
    - Event Monitoring with wait4 syscall (linux `time` utility with custom format string)
    - Functions: read, write, delete
    - Runtimes: PyPy, CPython, and C++
- Runtimes Systems
    - PyPy:
        - just-in-time compiler
        - Incminimark (incremental, generational moving collector)
    - CPython:
        - Mixed mode (interpreter with profile guided optimization)
        - Reference counting GC

Profiling - Execution Time

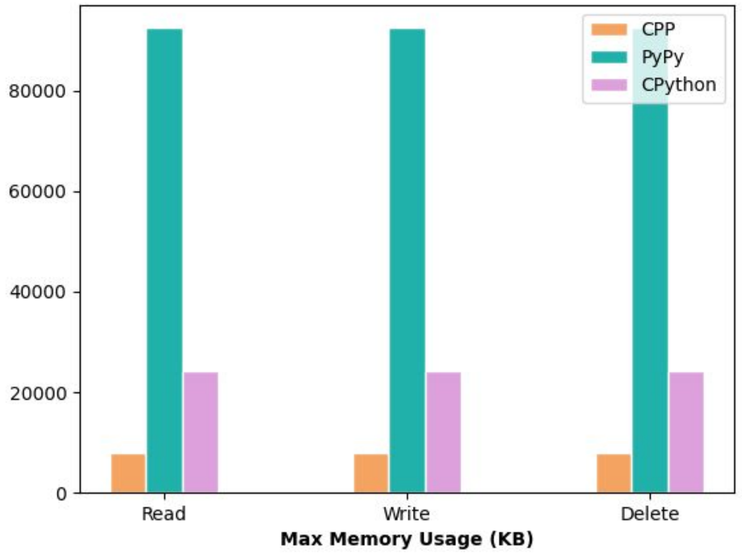


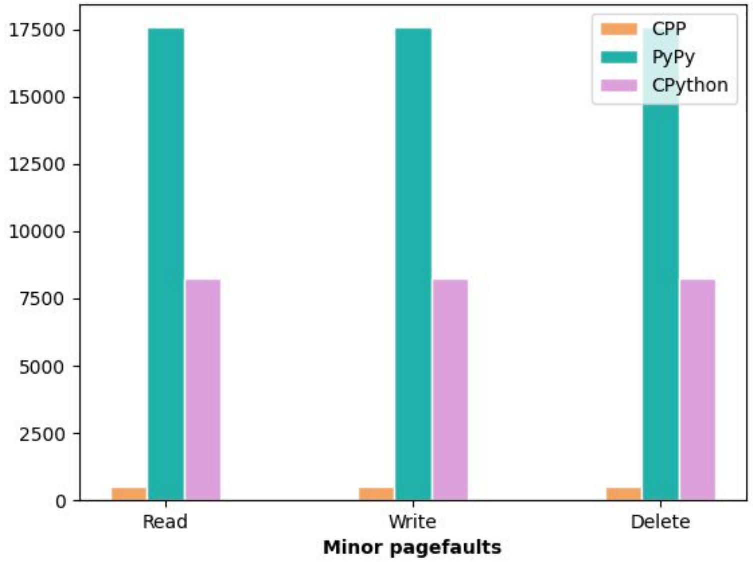


Profiling - CPU



Profiling - Memory





Future Work

- Network requests and request batching is obviously the first thing that should be optimized
- Master server can send regular “heartbeat” message to edge-servers

Thank You!