PIZZA DELIVERED QUICKLY

FACTORY LOCATOR SUBSYSTEM

KOD Inc

OUTLINE

- ✓ Problem Description
- ✓ Project Management
- ✓ The Solution
- ✓ Features
- ✓ Demo

Problem Description

Problem Description

Pizza Factory Locator Subsystem

The first is a software subsystem to find pizza factory locations. It is not known how many such factories will be needed nor where they should be located. The software subsystem will have to determine that. Clearly this subsystem is a very complex application. The goal can be clearly defined, but even then the solution will not be at all obvious. This subsystem will have to use a very sophisticated modeling tool. The requirements, functionality, and features are not at all obvious. Some of the solution can probably be envisioned, but clearly the whole solution is elusive at this early stage. Exactly how to model it is not known at the outset. It will have to be discovered as the development project is underway.

Project Description

- Determine the optimal location for a new factory
- Return closest factory to customers' order
- Provide location of existing factories

Problem Description (contd.)

Considerations

- ✓ Big Data Processing
- ✓ Fault-Tolerance
- ✓ Scalability

Project Management

Initiating Processes

- Develop project charter
- Identify stakeholders

Planning Processes

- Collect requirements
- Define scope
- Create WBS
- Plan communication management
- Plan risk management

Executing Processes

- Acquire resources
- Manage communication
- Direct and manage project work

Monitoring and Controlling Processes

- Monitor and control project work
- Perform integrated change control
- Valid scope
- Control scope
- Monitor communication

Closing Processes

- Close project

The Solution

The Solution

Erlang-Powered, multithreaded, big-data processing application

https://github.com/otboss/PDQ-Pizza-Factory-Locator-Subsystem

The Solution (contd.)

Erlang Introduction

- Created in 1986 by Ericsson
- Erlang is widely used in the telecommunications industry
- Massively scalable applications
- Built upon concurrency, processes inside the VM are isolated

The Solution (contd.)

Big-Data processing can be tricky..

The Solution (contd.)

Data Processing Strategy

1. The total number of relevant records in the database are considered
2. The total number of CPU threads are considered
3. Ranges for each CPU thread are created by dividing the number of relevant records by the number of CPU threads

The Solution (contd.)

4. Relevant records are read from the database in chunks by each thread and processed
5. During processing each thread changes a large, random numerical value in shared memory. If this value does not change after (x) seconds then all threads are finished
6. The result from each thread is combined and the final solution is calculated on a single thread

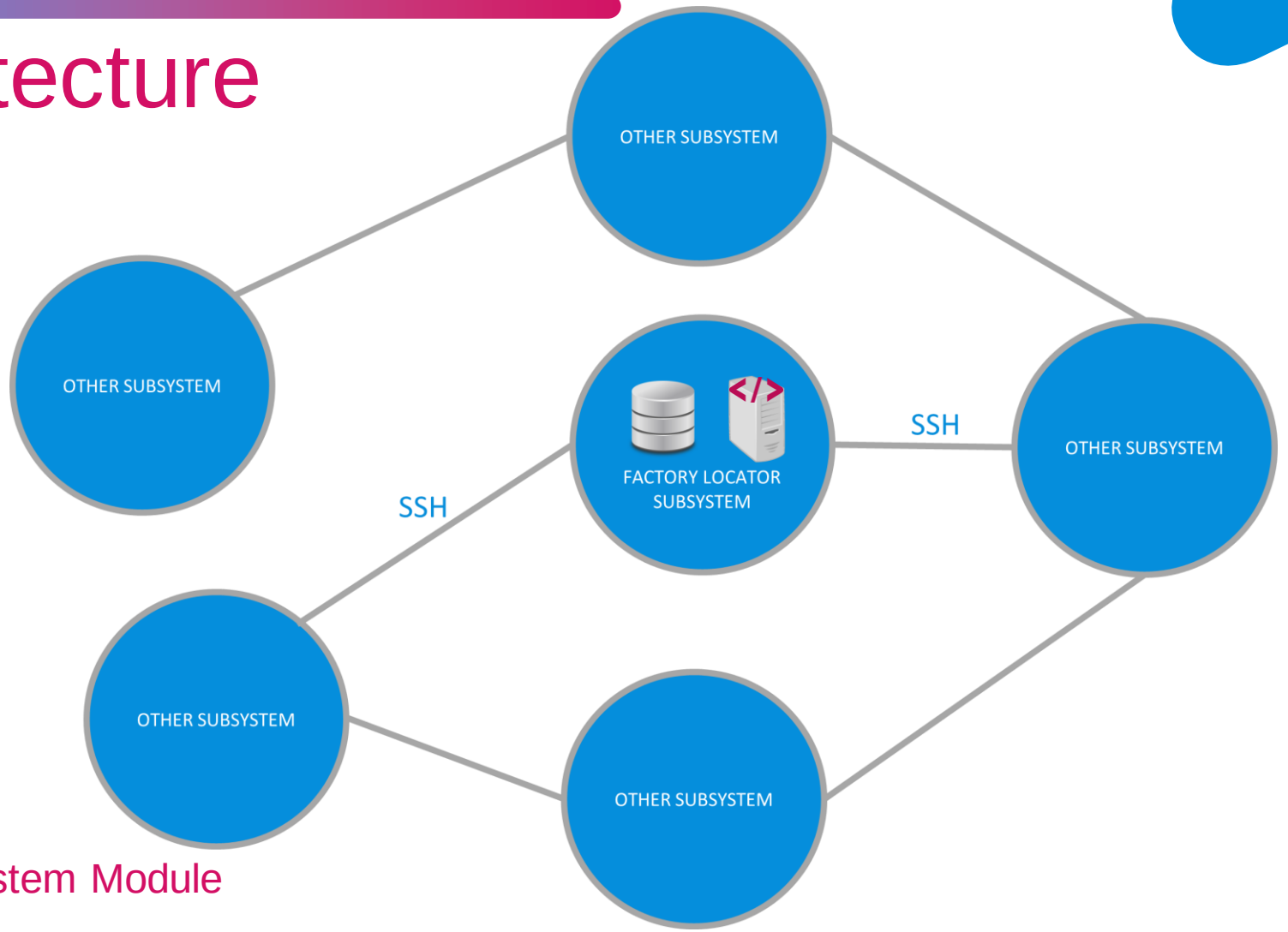
The Solution (contd.)

Deployment Architectures

- Central
- Partially Central
- Distributed

The Solution (contd.)

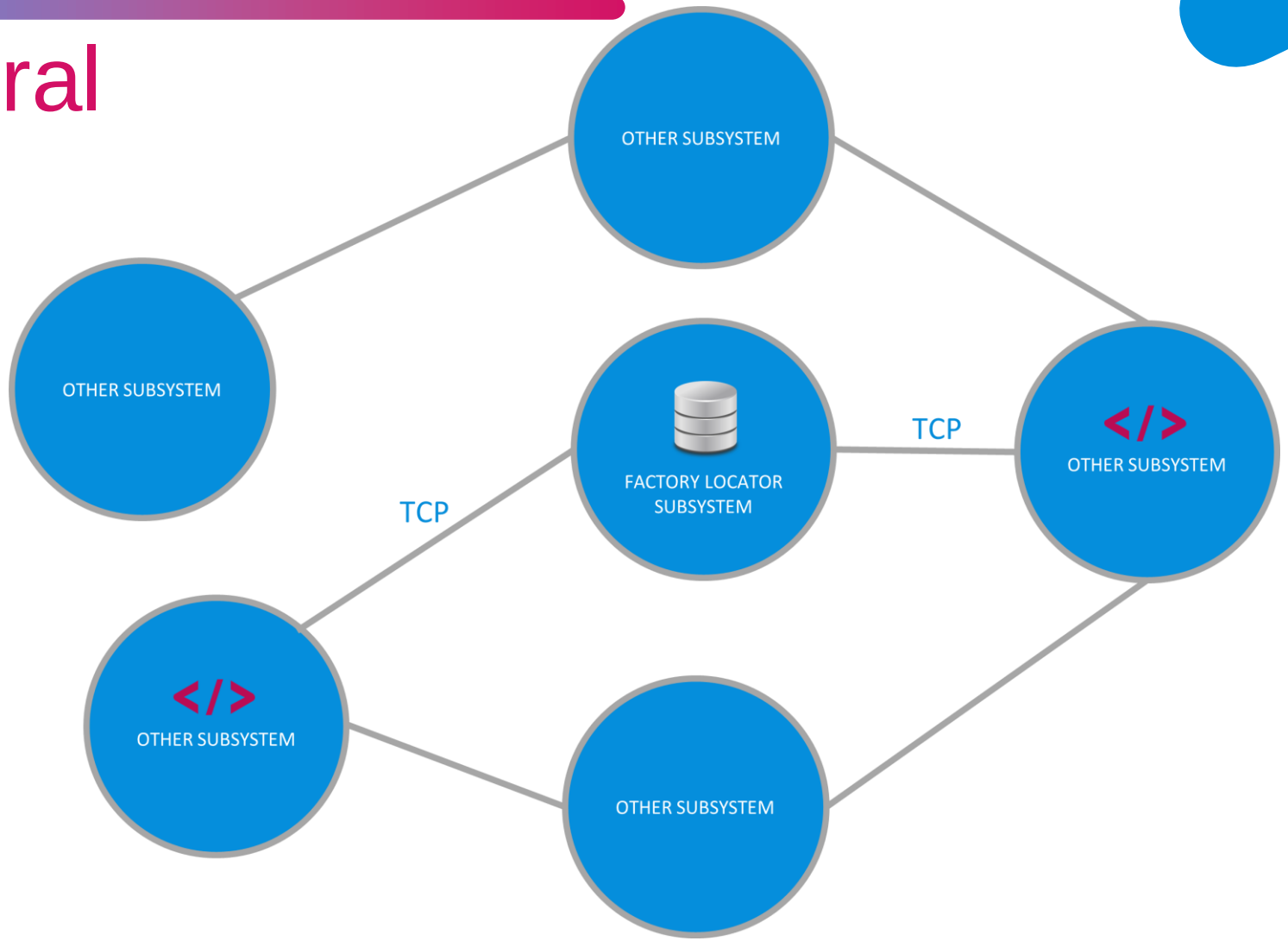
Central Architecture



</> - Factory Locator Subsystem Module

The Solution (contd.)

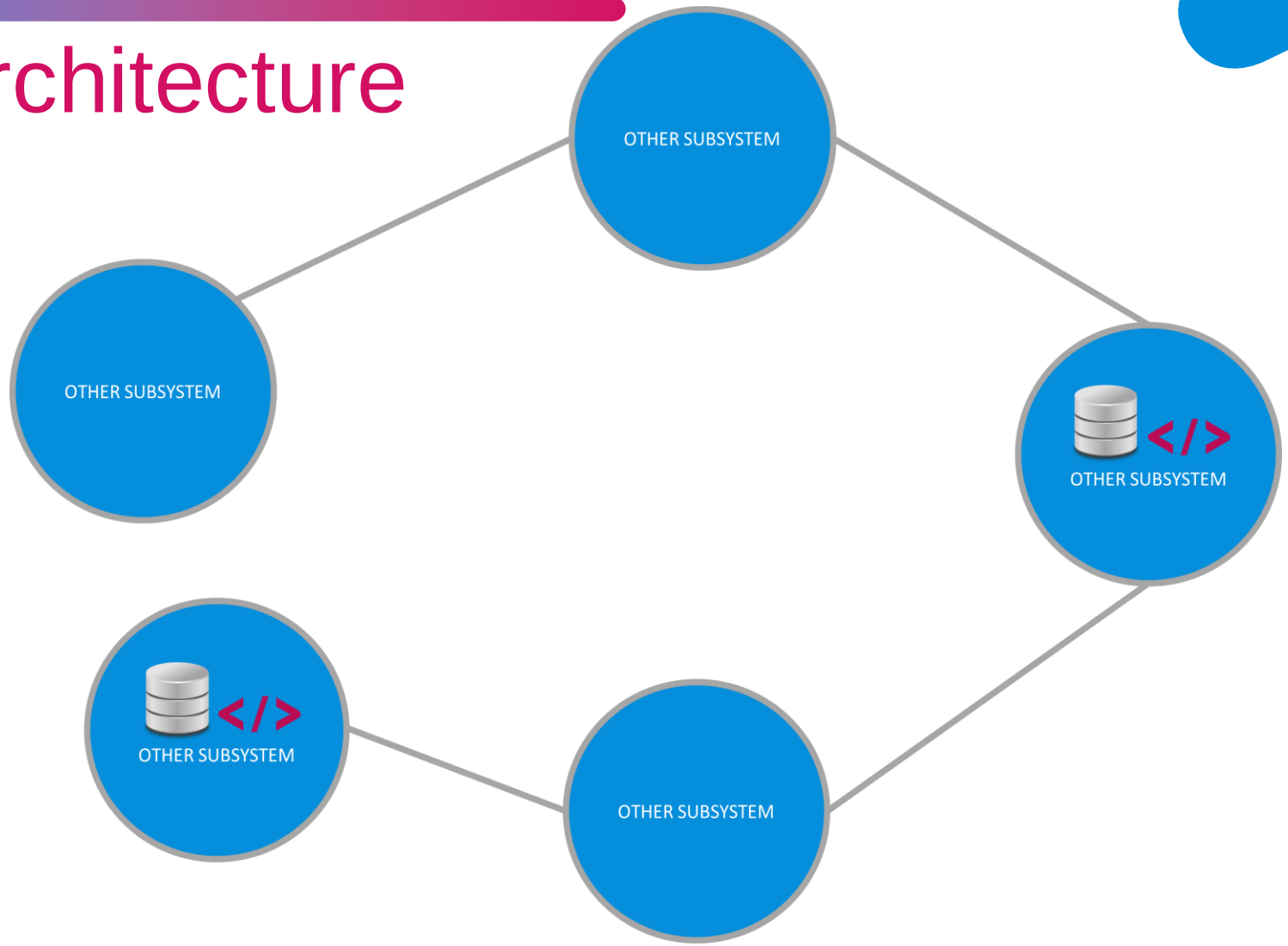
Partially Central Architecture



The Solution (contd.)

Distributed Architecture

(Replica Set)



The Solution (contd.)

CLI Supported Platforms

- Linux
- Mac OS
- Windows

The Solution (contd.)

Other Platforms

- Elixir Interactive Shell (Erlang/OTP)
- Docker (Mounting the Linux CLI Folder)

Features

Features

CLI Features

- Set the configuration of the database (MongoDB)
- Get the database Configuration
- Save order to the database
- Save a factory to the database

Features

CLI Features

- Determine the new factory location
- Get the closest factory to coordinates
- Get factories from the database

For more info see the application's documentation at:

https://github.com/otboss/PDQ-Pizza-Factory-Locator-Subsystem/blob/master/README.md

Demo

Thank You

References

- https://erlang.org/download/armstrong_thesis_2003.pdf
- https://people.uwec.edu/sulzertj/Teaching/is455/Resources/PizzaDeliveryQuickly_Case_Study.pdf