Systems Design

### Introduction

This document covers

Introduction

## Projects

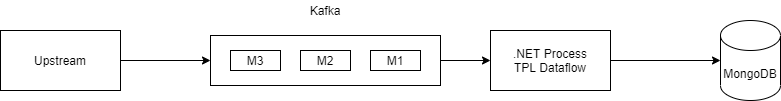
### Project P

Our aim was to use modern technology to create an architecture that could scale to future business needs. The technologies we used included.

* MongoDB
* GraphQL
* Open Fin
* Elastic APM
* React.JS
* TypeScript
* KeyCloak

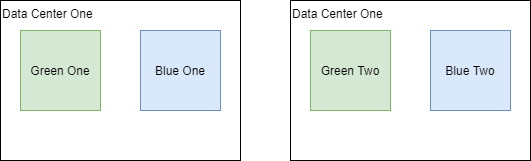
#### Pipe-Line

On the data feed upstream processes write to Kafka. A .NET process takes the messages from Kafka, processes them, and writes them into MongoDB. The .NET process uses TPL Dataflow and overall achieves 20,000 updates per second using MongDB Sharding.



#### .NET Core API

Using GraphQL enables clients to request only fields they require reducing I/O from MongoDB to .NET core process and from .NET core process to Web client. We deploy our process using active/active. By taking down one host from each data center we can deploy new versions with 100% uptime.



#### Open Fin

Lets web apps run as desktop apps. Provides client side bus.

#### .NET Health Checks

Used by netscaler to determine which nodes are healthy.

#### KeyCloak

Open Source identity management supporting roles. Using OIDC and JWT

### Sharding MongoDB

By using sharding we obtained 20,000 position updates per second.

### Hekaton

Batch update 450,000 rows per second

140,000 transactions per second 24 cores.

Slowest part of SQL Server is persisting/reading to disk. In memory OLTP does not have this problem. Memory though put. We need to optimize DataFlow TPL task scheduler does not have basic configuration that deals with disk write expectations that deal with the speed of In Memory OLTP.

## 