In-Memory Computing SUMMIT VIRTUAL EVENT 2020

How to Perfect Your Legacy Strategy: A Mainframe Modernization Case Study

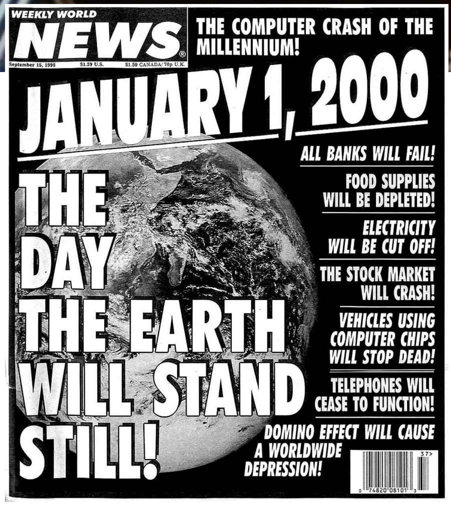
Galen Silvestri, Senior Solutions Engineer
GigaSpaces

October 29th, 2020

GIGASPACES

Mainframes Are Here to Stay

Assembly of IBM 1401, Circa 1960



WEEKLY WORLD NEWS®

September 15, 1998 $1.39 U.S. $1.59 CANADA/70p U.K.

THE COMPUTER CRASH OF THE MILLENNIUM!

JANUARY 1, 2000

THE DAY THE EARTH WILL STAND STILL!

ALL BANKS WILL FAIL!

FOOD SUPPLIES WILL BE DEPLETED!

ELECTRICITY WILL BE CUT OFF!

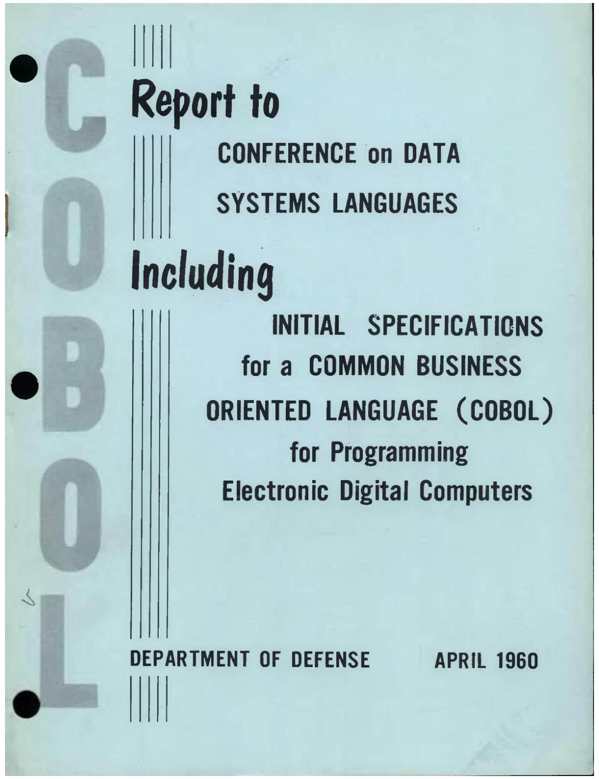
THE STOCK MARKET WILL CRASH!

VEHICLES USING COMPUTER CHIPS WILL STOP DEAD!

TELEPHONES WILL CEASE TO FUNCTION!

DOMINO EFFECT WILL CAUSE A WORLDWIDE DEPRESSION!

COBOL is Still Among Us



COBOL

Report to

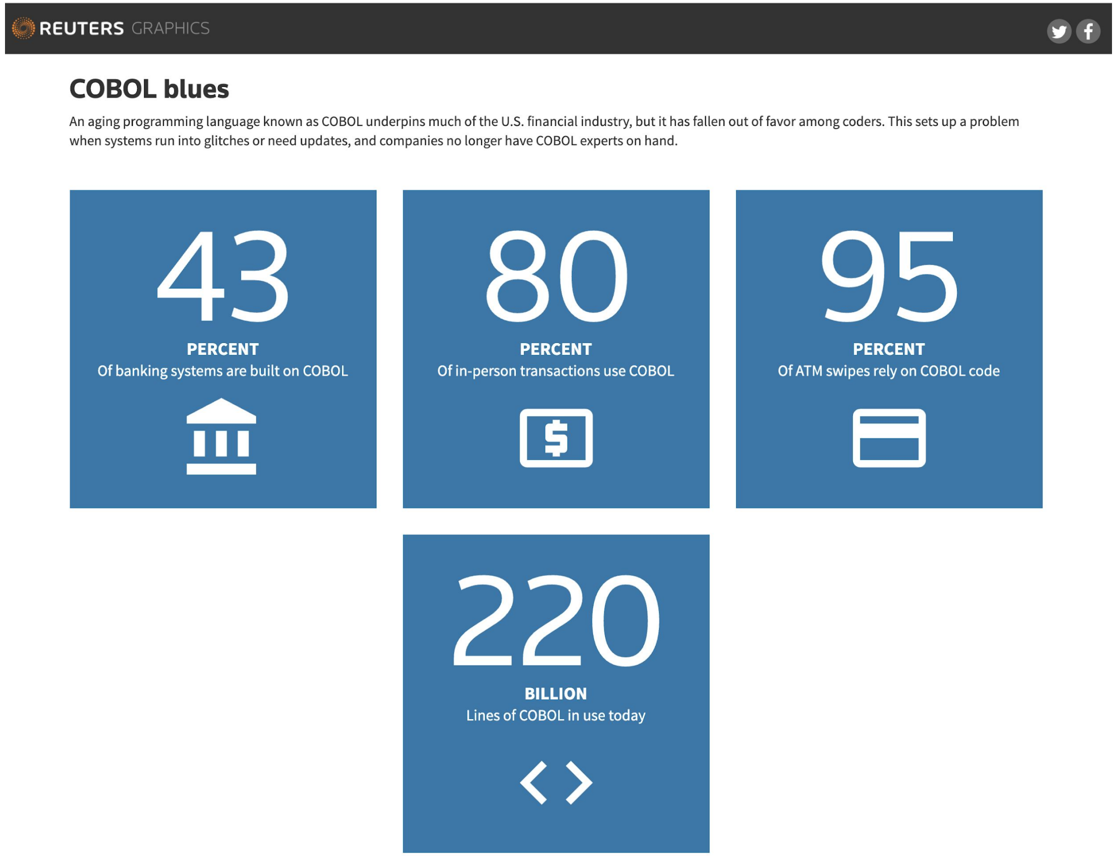
CONFERENCE on DATA

SYSTEMS LANGUAGES

Including

INITIAL SPECIFICATIONS
for a COMMON BUSINESS
ORIENTED LANGUAGE (COBOL)
for Programming
Electronic Digital Computers

DEPARTMENT OF DEFENSE APRIL 1960



REUTERS GRAPHICS

COBOL blues

An aging programming language known as COBOL underpins much of the U.S. financial industry, but it has fallen out of favor among coders. This sets up a problem when systems run into glitches or need updates, and companies no longer have COBOL experts on hand.

43
PERCENT
Of banking systems are built on COBOL

80
PERCENT
Of in-person transactions use COBOL

95
PERCENT
Of ATM swipes rely on COBOL code

220
BILLION
Lines of COBOL in use today

Reuters Report: COBOL Underpins Financial Industry

Mainframe Use is Growing

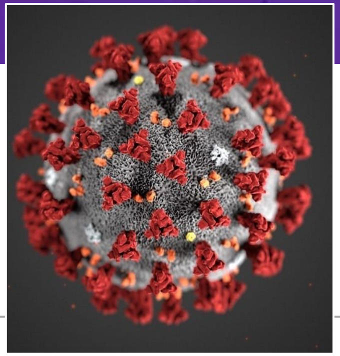
56% of infrastructure decision makers
at these enterprises **use the mainframe**

46% predict an i**ncreased investment**
over the next two years

2019: Forrester's "Tackling the Unsexy Challenge of Mainframe Modernization the Cloud Era Demands Connection to Modern DevOps Practices"

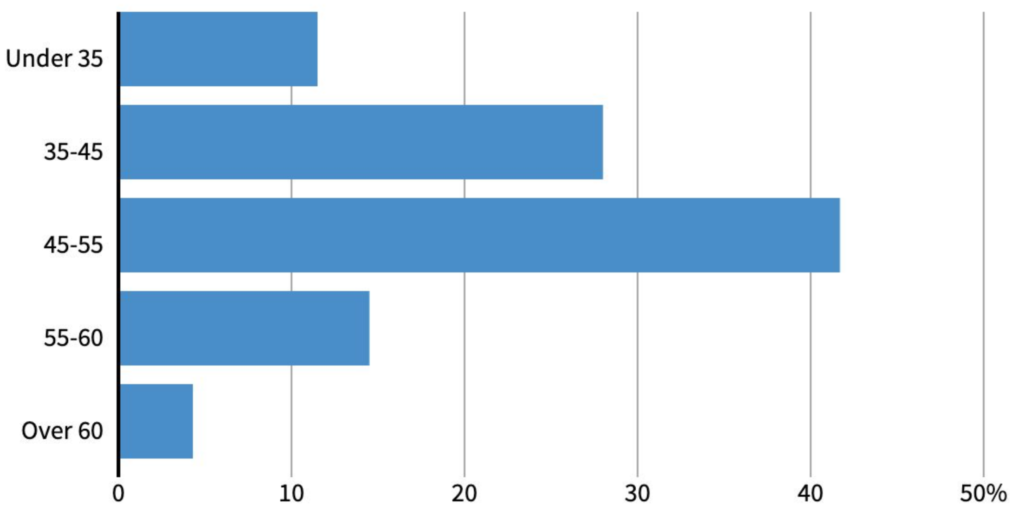
Challenges

- **Missing Capacity. Extend mainframes?** $$$
- **Procure COBOL DEV's?** Is retraining this possible?
- **Rip & Replace Mainframes?** High Risk



AVERAGE AGE OF DEVELOPERS

On average, COBOL programmers are most likely to be between 45-55 years old.



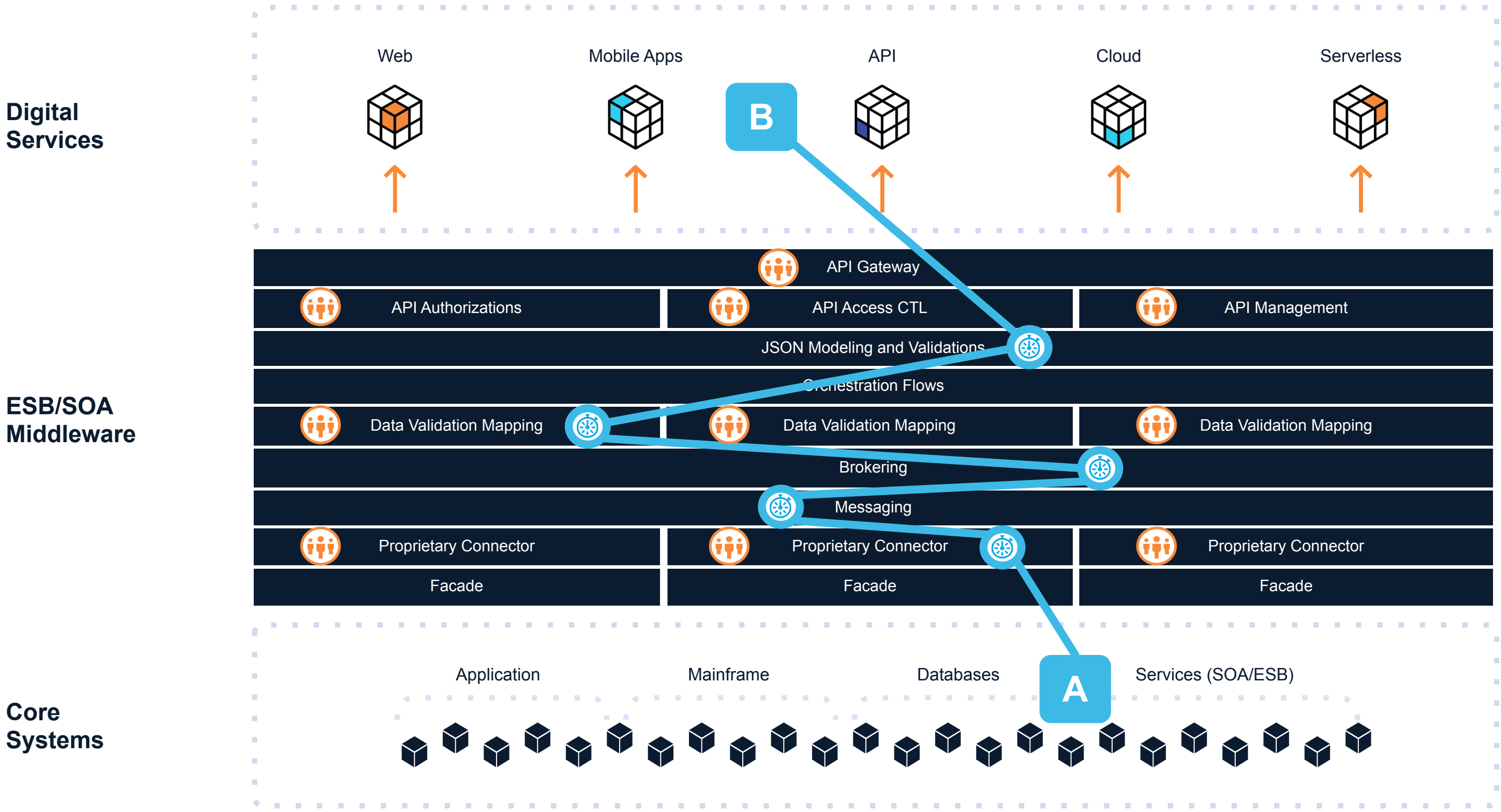
Typical Modernization Approaches

#	Approach	Time	Effort	Cost	Overall Risk	Business Support
1	Convert/ migrate and retire existing					
	A. 3rd Party solution	Medium	Medium	High	High	Low
	B. in-house solution	High	High	High	High	Low
2	Upgrade Existing Mainframe	Medium	Medium	Medium	Medium	Medium
3	**Augmenting technology on existing**	Low	Low	Low	Low	High

Benefits of #3:

- *Path of least resistance – augment existing client solution*
- *Least disruption to existing Business and IT stakeholders – at all levels*
- *Reduces TCO and provides business value the quickest*

Traditional Architecture



Digital Services

Web

Mobile Apps

API

Cloud

Serverless

B

ESB/SOA Middleware

API Gateway

API Authorizations

API Access CTL

API Management

JSON Modeling and Validations

Orchestration Flows

Data Validation Mapping

Data Validation Mapping

Data Validation Mapping

Brokering

Messaging

Proprietary Connector

Proprietary Connector

Proprietary Connector

Facade

Facade

Facade

Core Systems

Application

Mainframe

Databases

A

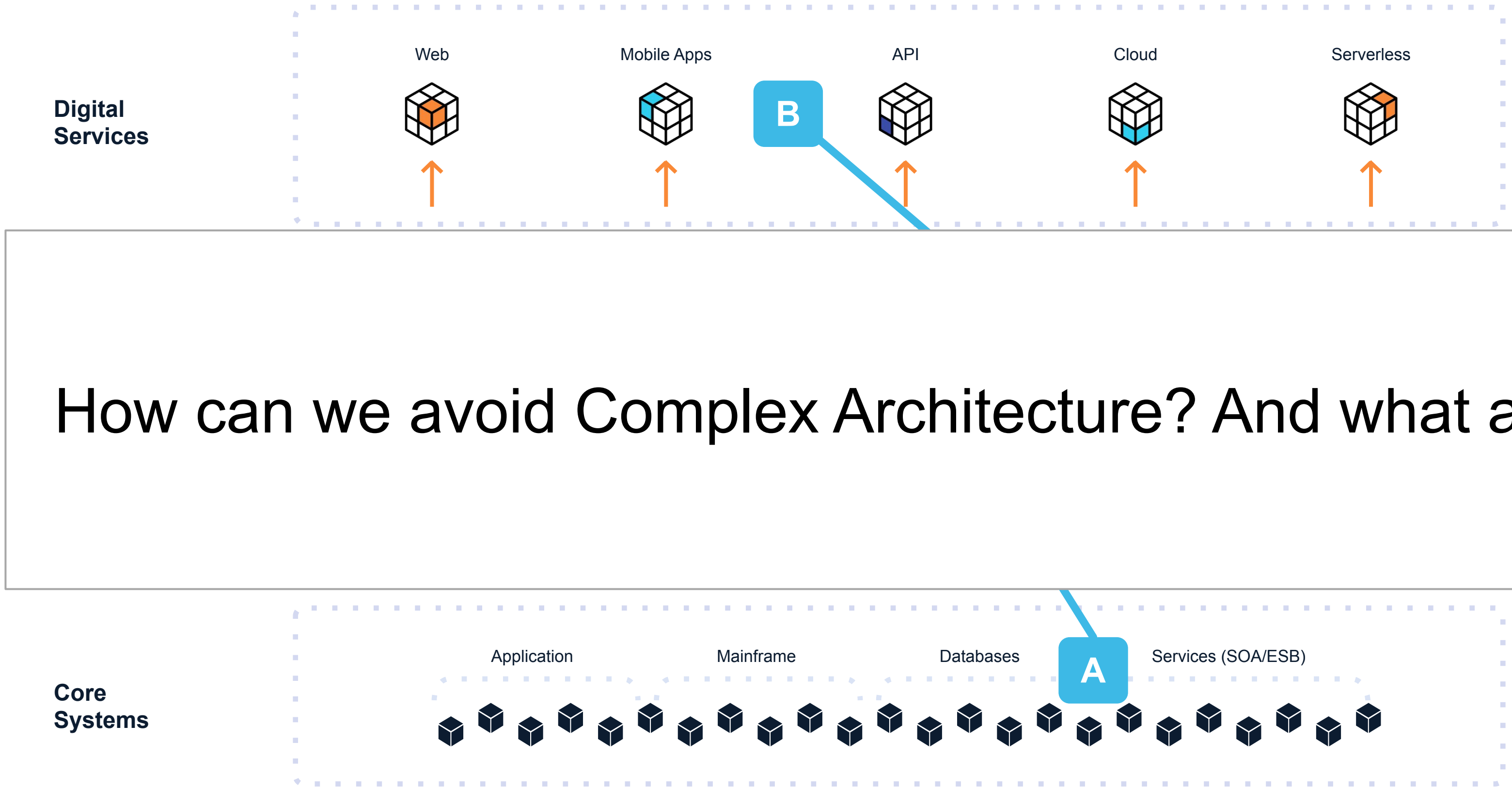
Services (SOA/ESB)

Result is that digital services pass thru these complex layers which impacts API performance.

Layers of ESB/SOA add complexity, requiring more people and specific skill sets.

Legacy systems of all types require special skills and a dwindling workforce.

Traditional Architecture



How can we avoid Complex Architecture? And what are the benefits?

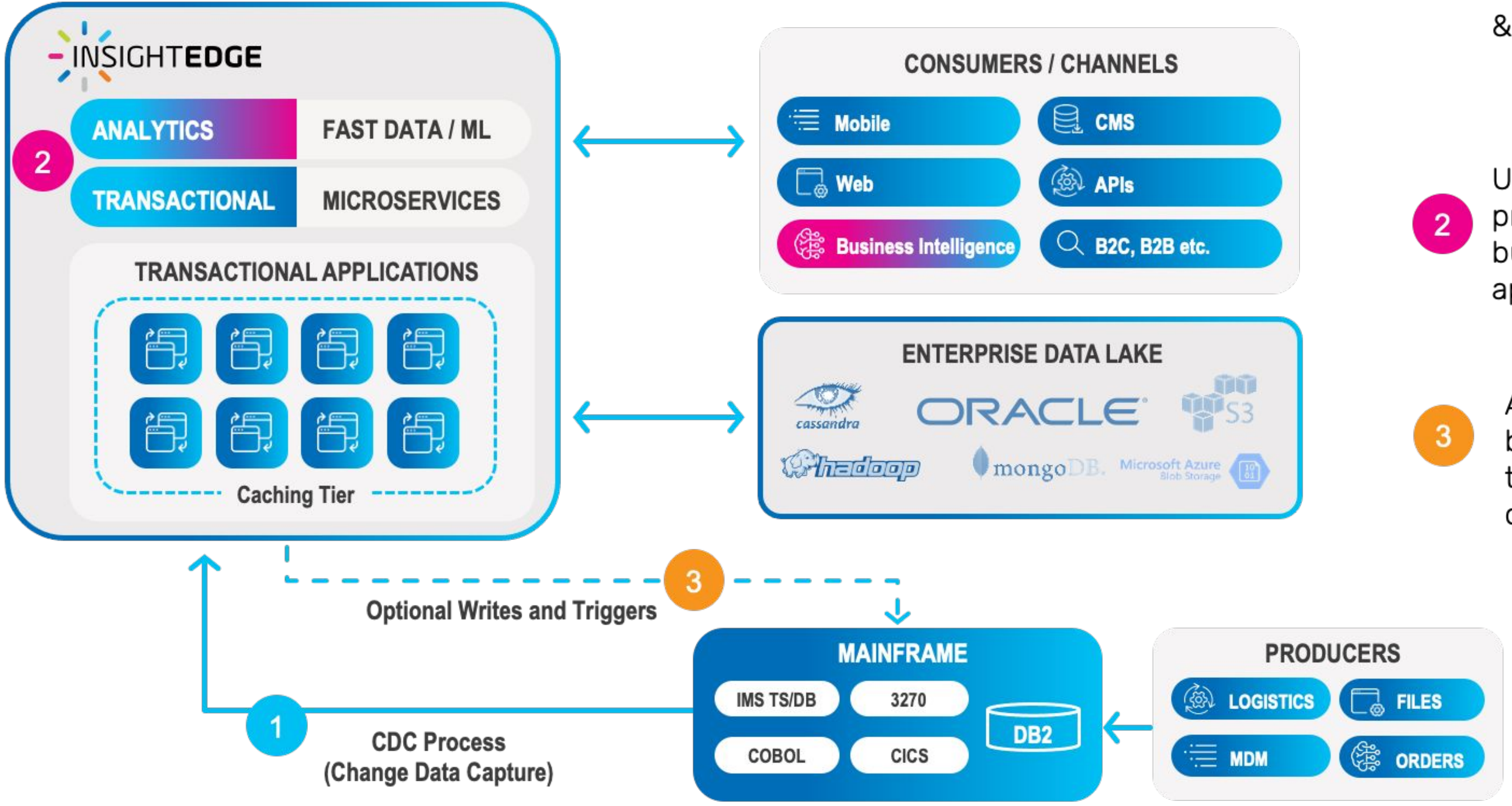
Result is that digital services pass thru these complex layers which impacts API performance.

Legacy systems of all types require special skills and a dwindling workforce.

So What Does a Mainframe Modernization Strategy Look Like?

- **Capture your mainframe data**
  - Ex. DB2
  - Via: Change Data Capture (CDC)
  - Transfer to distributed in-memory data fabric
- **Leverage Extreme Transaction Processing**
  - ACID compliant
  - Fast analytics for business intelligence
  - MIPs offloading with Minimal latency
- **Optional Bi-Direction**
  - Expose COBOL applications or CICS API
  - Via microservice-based APIs
  - Integrate modernized applications

Proposed Mainframe Modernization Architecture



INSIGHTEDGE

2

ANALYTICS | FAST DATA / ML

TRANSACTIONAL | MICROSERVICES

TRANSACTIONAL APPLICATIONS

Caching Tier

CONSUMERS / CHANNELS

Mobile, CMS, Web, APIs, Business Intelligence, B2C, B2B etc.

ENTERPRISE DATA LAKE

cassandra, ORACLE, S3, hadoop, mongoDB., Microsoft Azure Blob Storage

3 Optional Writes and Triggers

MAINFRAME

IMS TS/DB, 3270, COBOL, CICS, DB2

PRODUCERS

LOGISTICS, FILES, MDM, ORDERS

1 CDC Process (Change Data Capture)

1 Continuously integrate and capture the data from a Mainframe (DB2) to GigaSpaces InsightEdge, a front end scalable & distributed data fabric

2 Unifying extreme transaction processing and fast analytics for business intelligence and digital applications with minimal latency

3 Automation of microservice-based APIs with modern SDKs triggering legacy COBOL or CICS applications

Required Functionality

- **Mainframe offload**
  - eZ & quick access
  - Enrich via data lakes / warehouses

- **Microservices**
  - APIs
  - Event-driven analytics - Spark and BI

- **Translytics** Hybrid Transactional and Analytical Processing

- **Subsecond Data format agnostic ingestion**. IOPS M / sec with minimal ETL

- **Automatic Management:** Elastic Scale, DR, and Data tiers movement, indexing.

- **Deployment Anywhere** cloud, onPrem, hybrid, multi-cloud

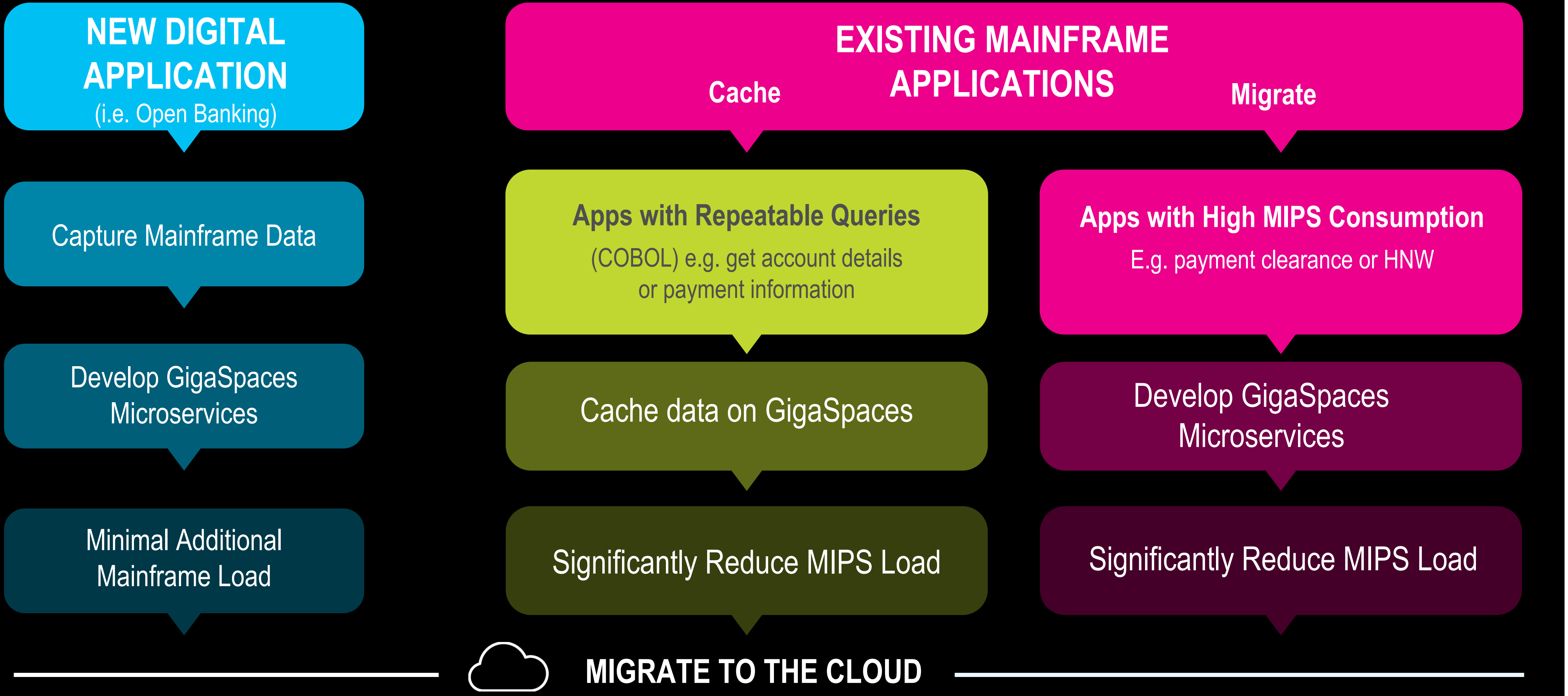
- **Agility.** Support DevOps and modern data management

What Are The Benefits ?

- **Reduce costs**
  - Smart Caching tier: offload mainframe MIPS
  - Infrastructure agnostic: Commodity Hardware and/or Cloud
- **Meet Availability SLAs**
  - Elastic scaling for peaks
  - Reduce overprovision
- **Modernize**
  - Innovate with modern apps
  - Future migration Journey
  - High-throughput, low-latency transactions
  - Fast data analytics & ML

- **Eliminate bottleneck** via on back-end applications
- **Faster time2market** for new modern services
  - Microservices architecture
  - Modern coding languages and frameworks (ie. Java and Spark)
  - Avoid new apps / use-cases addicted to on legacy
- **Easily migrate to Cloud**
  - Cloud native software that supports continuous migration
- **GigaSpaces WAN.** Efficient replication solution between remote sites,
  - Hybrid (MF + Gigaspace onPrem) ⇄ (GigaSpaces Cloud)
  - Reducing network overhead
  - Enforcing privacy regulations

Enterprises Require Flexible Implementation Paths



NEW DIGITAL APPLICATION
(i.e. Open Banking)

Capture Mainframe Data

Develop GigaSpaces Microservices

Minimal Additional Mainframe Load

EXISTING MAINFRAME APPLICATIONS

Cache

Apps with Repeatable Queries
(COBOL) e.g. get account details or payment information

Cache data on GigaSpaces

Significantly Reduce MIPS Load

Migrate

Apps with High MIPS Consumption
E.g. payment clearance or HNW

Develop GigaSpaces Microservices

Significantly Reduce MIPS Load

MIGRATE TO THE CLOUD

Real Life Case Study: PSA

Real-Time Pricing Engine

PSA GROUPE

ABOUT PSA GROUPE

Groupe PSA is the second largest car manufacturer in Europe. PSA sold 3.5 million vehicles worldwide in 2019.

BEFORE GIGASPACES

- WLTP regulation requires to calculate CO2 emission for every priced car. Compliance issues may lead to significant fines Many car configurations are unique, but not all parts are significant for CO2 calculation
- The mainframe pricing engine max capacity is 200 calculations per second. Demand is expected to reach 3000

WHY DID REDIS FAIL

PSA tried to use Redis cache to offload queries from their mainframe. It failed because PSA needed multi-criteria queries but Redis was designed for a single index. Redis workarounds required replicating data footprint by 6X, with a major performance hit

SOLUTION

- GigaSpaces Smart Cache was implemented with secondary indexes that allow multiple key queries
- Digital applications query Smart Cache for CO2 calculation. Only if the result is not in the cache, Smart Cache will query the mainframe one time. After which future queries will be served from the cache.

RESULTS

- Redis **footprint reduced by 6X** by leveraging multiple indexes and avoiding unnecessary replication.
- Smart Cache **response time reduced to 15-19 milliseconds vs 300 milliseconds** mainframe response time
- **More than 95% of queries are served by a super fast cache**, avoiding overloading the mainframe beyond its capacity

Reduce Redis footprint **6X**

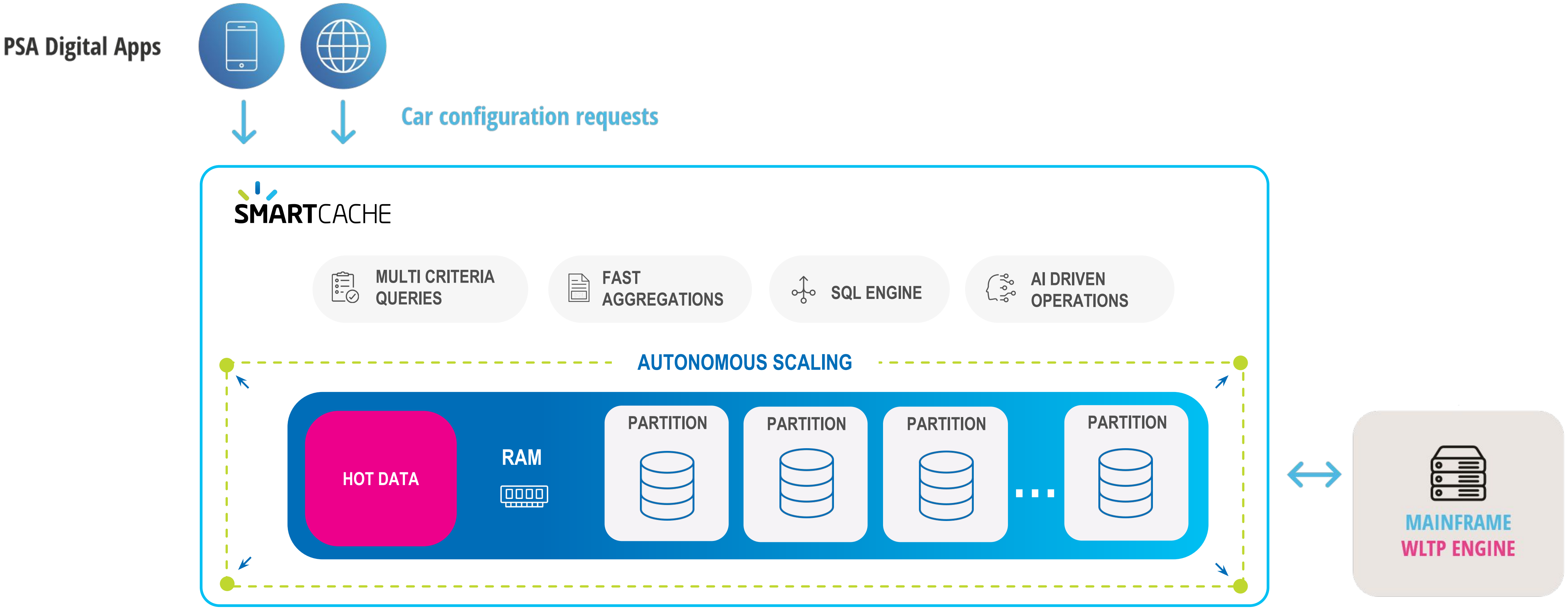
Scaling up by **20x**

15-19 millisecond Response time (down from 300)

GIGASPACES

In-Memory Computing SUMMIT VIRTUAL EVENT 2020

PSA SmartCache & MF Architecture



PSA Digital Apps

Car configuration requests

SMARTCACHE

MULTI CRITERIA QUERIES

FAST AGGREGATIONS

SQL ENGINE

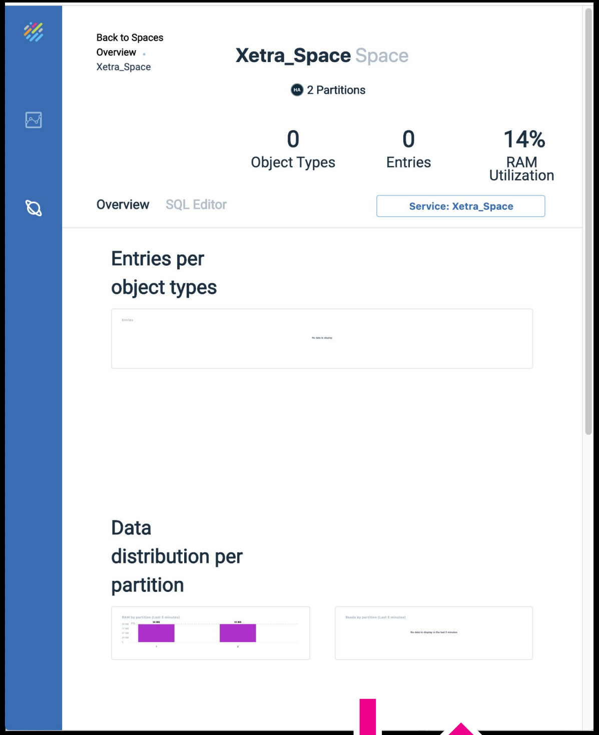
AI DRIVEN OPERATIONS

AUTONOMOUS SCALING

Mainframe Modernization Simple Workflow

Loader

IBM DB2



Back to Spaces
Overview
Xetra_Space

Xetra_Space Space

2 Partitions

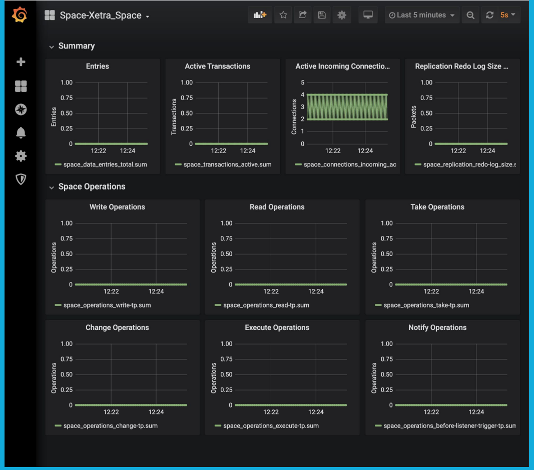
0	0	14%
Object Types	Entries	RAM Utilization

Overview SQL Editor

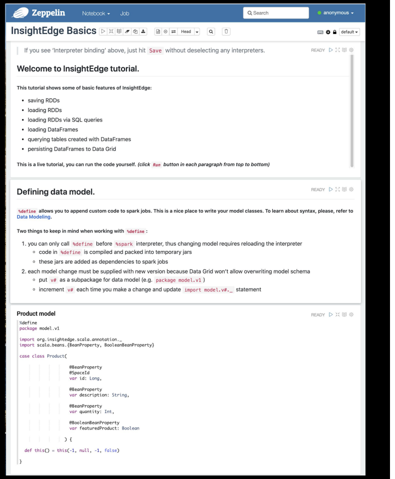
Service: Xetra_Space

Entries per object types

Data distribution per partition



Space-Xetra_Space
Last 5 minutes
5s
Summary
Entries
Active Transactions
Active Incoming Connectio...
Replication Redo Log Size ...
Entries
Transactions
Connections
Packets
1.00
0.75
0.50
0.25
0
12:22
12:24
space_data_entries_total.sum
space_transactions_active.sum
space_connections_incoming_ac
space_replication_redo-log_size.s
Space Operations
Write Operations
Read Operations
Take Operations
Change Operations
Execute Operations
Notify Operations
Operations
space_operations_write-tp.sum
space_operations_read-tp.sum
space_operations_take-tp.sum
space_operations_change-tp.sum
space_operations_execute-tp.sum
space_operations_before-listener-trigger-tp.sum



InsightEdge Basics

If you see 'Interpreter binding' above, just hit Save without deselecting any interpreters.

Welcome to InsightEdge tutorial.

This tutorial shows some of basic features of InsightEdge:

- saving RDDs
- loading RDDs
- loading RDDs via SQL queries
- loading DataFrames
- querying tables created with DataFrames
- persisting DataFrames to Data Grid

This is a live tutorial, you can run the code yourself. ***(click*** `Run` ***button in each paragraph from top to bottom)***

Defining data model.

`%define` allows you to append custom code to spark jobs. This is a nice place to write your model classes. To learn about syntax, please, refer to Data Modeling.

Two things to keep in mind when working with `%define`:

1. you can only call `%define` before `%spark` interpreter, thus changing model requires reloading the interpreter
   - code in `%define` is compiled and packed into temporary jars
   - these jars are added as dependencies to spark jobs
2. each model change must be supplied with new version because Data Grid won't allow overwriting model schema
   - put `v#` as a subpackage for data model (e.g. `package model.v1`)
   - increment `v#` each time you make a change and update `import model.v#._` statement

Product model

```
%define
package model.v1

import org.insightedge.scala.annotation._
import scala.beans.{BeanProperty, BooleanBeanProperty}

case class Product(

                @BeanProperty
                @SpaceId
                var id: Long,

                @BeanProperty
                var description: String,

                @BeanProperty
                var quantity: Int,

                @BooleanBeanProperty
                var featuredProduct: Boolean

              ) {

  def this() = this(-1, null, -1, false)

}
```

Mainframe is here to stay but you can now...

- **Reduce costs**: Smart Caching tier: offload mainframe MIPS
- **Meet Availability SLAs**: Elastic scaling
- **Modernize**

- **Eliminate bottleneck** via on back-end applications
- **Faster time2market** for new modern services
- **Easily migrate to Cloud**
- **GigaSpaces WAN.** Efficient multi cluster

Thank you!

For any questions, don't hesitate to contact me:
galen.silvestri@gigaspaces.com

GIGASPACES

In-Memory Computing SUMMIT
VIRTUAL EVENT 2020