High throughput at massive scale deployment pattern

A PEGA WHITEPAPER

Introduction

Business use cases often require that Data Capture is handled at very high throughput and massive scale – often in "bursty" traffic patterns.

Examples include:

- Survey or Registration applications that capture information from communities, groups,or population subsets for subsequent analysis or processing.
- Order Capture for popular products or events where demand scales massively around the launch/availability dates.
- IOT processing where sensor farms produce a stream of events – most of which require minimal immediate action – and can instantly scale to a deluge in the event of failure or incident

All use cases share certain characteristics:

- The front-end processing is typically quite light with limited requirement to draw on data or parameters from a dynamic central database.
- The front-end must scale up rapidly in response to potential viral load demand and scale down equally rapidly to avoid wasted capacity expense.
- Response at the front-end must be very fast to provide a superb user experience.
- Back-end processing – fulfilment, analysis, reporting – may be more intense and complex.
- Back-end processing has more relaxed throughput expectations, as the customer is no longer waiting.

Pega's Deployment Pattern uses the flexibility of the microservice architecture of the Infinity Platform to leverage Service Isolation to distribute microservices to achieve extreme throughput:

Front-end Services

Lightweight Data Capture and Validation services are built as a front-end that can operate on a variety of infrastructure platforms – cloud based servers, cloud based serverless functions, mobile devices or laptops, and MEC (Mobile Edge Computing) services that can be located with high adjacency to the user or sensor population, minimizing network overhead and latency.

Front-end services are designed to be deployed leveraging K8S (Kubernetes) with flexible scalability implemented at the cluster/pod parameter level – adjusting capacity as demand shifts.

Isolated, distributed, front-end services facilitate massive parallelism – and eliminate points of contention.

Back-end Services

Heavier, less time-sensitive processing is delegated to a central set of back-end services – typically aggregating data into a centralized data lake. These services would include such capabilities as analysis, fulfilment and follow-up, case creation and resolution, and reporting.

These services are typically deployed on redundant cloud infrastructure with appropriate capacity for compute, storage, and persistence. Auto-scaling here is less instantaneous as it responds less to response/demand parameters, rather to throughput SLAs, and cost-containment.

Joining the Front and Back Ends

Transactions are streamed in a "lazy load" fashion from the front-ends to the back-end. Typically Apache Kafka (or a cloud-specific service such as AWS Kinesis) is chosen for this – offering several advantages:

- Asynchronous operation – such that issues or overloads in any component do not export problems to other components.
- Increased security at the front-end when network isolation separates the front-end from the back-end. There is no connectivity that permits an attacker with access to the front-ends to target the back-end infrastructure

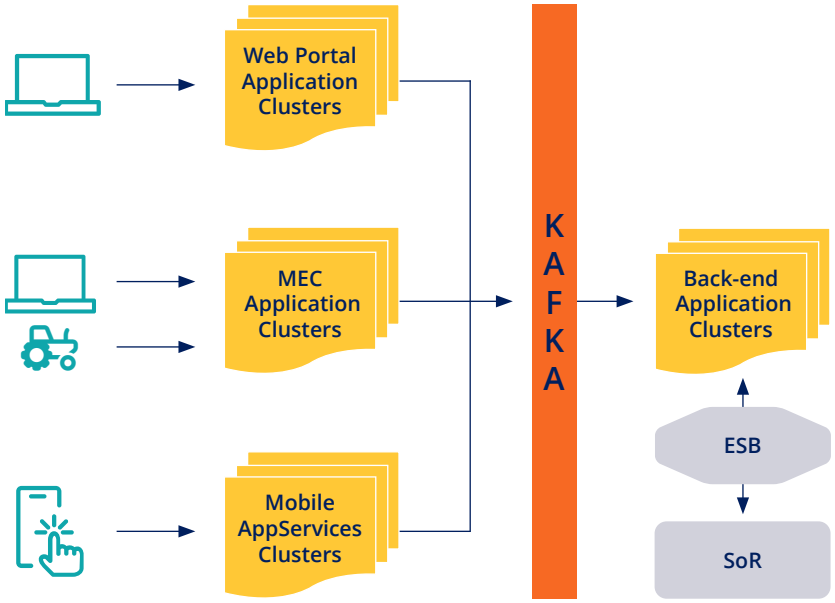
Conceptual Deployment Pattern – putting it all together

The key to optimized scale and performance is to match resource capacity to demand – where key resources are compute, storage, and network.

Even with the availability of massive capacity, we are still constrained by the speed of light if we create monolithic sequential processes with interdependencies.

Best practice requires an architecture that facilitates parallelism in processing, making a given process "wide" rather than "long" by creating compact microservices that can be deployed and invoked independently on appropriately-scaled infrastructure.

Distributed Adjacent Services Example



Registration Application

- Web Channel Served by Lightweight Data Capture Application Deployed in-Region
- IOT Sensors and 5G Laptop Users Served by local MEC servers
- Mobile App Leverages Disconnected Processing capability to handle data capture processing locally on the device
- Data capture isolated from full scale Back-end Application and Data Lake which is deployed in multiple regions and availability zones behind the firewall
- Kafka streaming transmits data asynchronously to back-end

Example of a multi-channel application

As illustrated by the schematic describing a multi-channel application, both the design of the services and their deployment topology can significantly improve ability to scale and ensure an optimal user experience.

The illustrated application involves rudimentary data capture, which involves basic validation and a workflow that is dependent on the data provided – so it needs processing intelligence (the rules) and data constraints (which are either static or infrequently changed).

Once data has been captured, more significant and expensive processing and transformation are required before the processing is completed and safe-stored within the data lake (system of record), and follow-up fulfilment, analysis, and reporting is completed.

This design separates data capture into a very lightweight, front-end service specific to each channel. This eliminates dependency on potentially slow database operations while creating a very fast, stateless service that can be deployed multiple times to meet fluctuating demand; in the web or MEC channels we have automation to add or remove Kubernetes (K8S) clusters.

The services can be deployed locally – at the edge on the web, in MEC cloud servers, or on local in-region cloud infrastructure – minimizing network latency.

Similarly, the mobile data capture app is designed to operate with or without a mobile connection – leveraging the compute power of the mobile device. Data capture operates asynchronously with lazy update being passed to the app service clusters when the connection is good and computational demand on the device is low.

About Pegasystems

Pega delivers innovative software that crushes business complexity. From maximizing customer lifetime value to streamlining service to boosting efficiency, we help the world's leading brands solve problems fast and transform for tomorrow. Pega clients make better decisions and get work done with real-time AI and intelligent automation. And, since 1983, we've built our scalable architecture and low-code platform to stay ahead of rapid change. Our solutions save people time, so our clients' employees and customers can get back to what matters most.

For more information, please visit us at **pega.com**