Embracing the Event-Driven Microservices Architecture

The Microservices Architecture is about breaking down complex system-level problems into independent modular entities so that they become easier to manage and deploy. Each Microservice runs a unique process and communicates through a well-defined, lightweight mechanism, such as a container, to serve a business goal. One major drawback with Microservices is its increased complexity which directly proportional to the number of services involved, which in turn scales up operational costs as well. The answer to this is **Event Driven Microservices (EDM)** which is an architectural pattern from the MSA family that overcomes the shortcomings of other micro-service architectural (MSA) patterns.

The Event-Driven Architecture enables a way to make Microservices manage transactions, concurrency, isolation, durability, materialized views, indexes, etc. This article emphasizes on the key benefits of an Event-driven Architecture for Microservices that is missing in other MSA patterns.

As the leading EDA implementation provider for Microservices, we can help accelerate the performance of your business applications. To learn more, call us today or visit our website.

Evolution of Enterprise Integration

The task of collaborating different software applications and forming new software solutions is referred to as 'enterprise integration'.

Stage 1- In the early stages of enterprise integration evolution, most organizations implemented in-house integration middleware using **Point-to-point integration**. This method led to a conundrum of spaghetti connections between several applications and systems because of a missing dedicated integration middleware framework that can centralize and facilitate all integration requirements of the infrastructure.

Stage 2- This stage saw the evolution of the **Enterprise Service Bus** integration that addressed issues like single point of failure and hub bottlenecks. The Bus architecture solves the scalability issues of the hub-spoke architecture as the centralized messaging bus can be scaled horizontally. However, the distributed integration tasks complicate and increase maintenance and troubleshooting overheads along with proprietary issues.

Stage 3- This led to the advent of the **MSA** which overcomes the drawback of Monolith ESBs by providing agility and flexibility, but also takes us back more towards point-to-point smaller functionalities that posed the challenges mentioned below.

Stage 4- This has finally evolved into the **EDM** architecture which inherits the key benefits of both Microservices and enterprise service bus architectures as a architecture, namely,

1. Scalability
2. Availability
3. Resiliency

4. Independent, autonomous
5. Decentralized governance
6. Failure isolation
7. Auto-Provisioning
8. Continuous delivery through DevOps

Point-to-Point Integration– The Challenges

This architecture is being used predominantly across enterprises these days, but, it quickly becomes unmanageable, expensive, brittle, and damaging to both the IT budget and the organization's ability to meet current and changing business demands because of the following reasons:

- **Exponential Increase in Complexity**
- **Single Points of Failure**
- **Security Issues**
- **Multi-Point Dependency**
- **Loss of Agility**

Why Event-Driven Integration Architecture?

Event-driven integration architecture (EDA) is a software architecture pattern that offers flexibility, scalability, and potential for vastly improved performance within High Performance Computing (HPC) and High Throughput Computing (HTC) clusters compared to the traditional point-to-point architecture. It consists of event emitters, event consumers, and event channels.

Event-Driven Integration Architecture is useful in the cases where,

- Multiple subsystems must process the same events
- Real-time processing is required with minimum time lag
- Complex event processing, such as pattern matching or aggregation over time windows is involved
- High volume and high velocity of data need to be processed

The Key Benefits of this approach are:

- It offers a potential for a fully-asynchronous non-blocking solution
- Messages/Events are small and represent very specific state changes within each service and/or task
- Messages/Events can be collected at any level and relayed (emitted individually or together in batch)
- Highly scalable and distributed when using high-performance channels
- Highly flexible workflow orchestration
- Highly optimizable. How events are collected, emitted, consumed can be tuned
- Producers and consumers are decoupled
- No point-to point-integrations. It's easy to add new consumers to the system
- Any number of consumers can respond to events immediately as they arrive
- Subsystems have independent views of the event stream

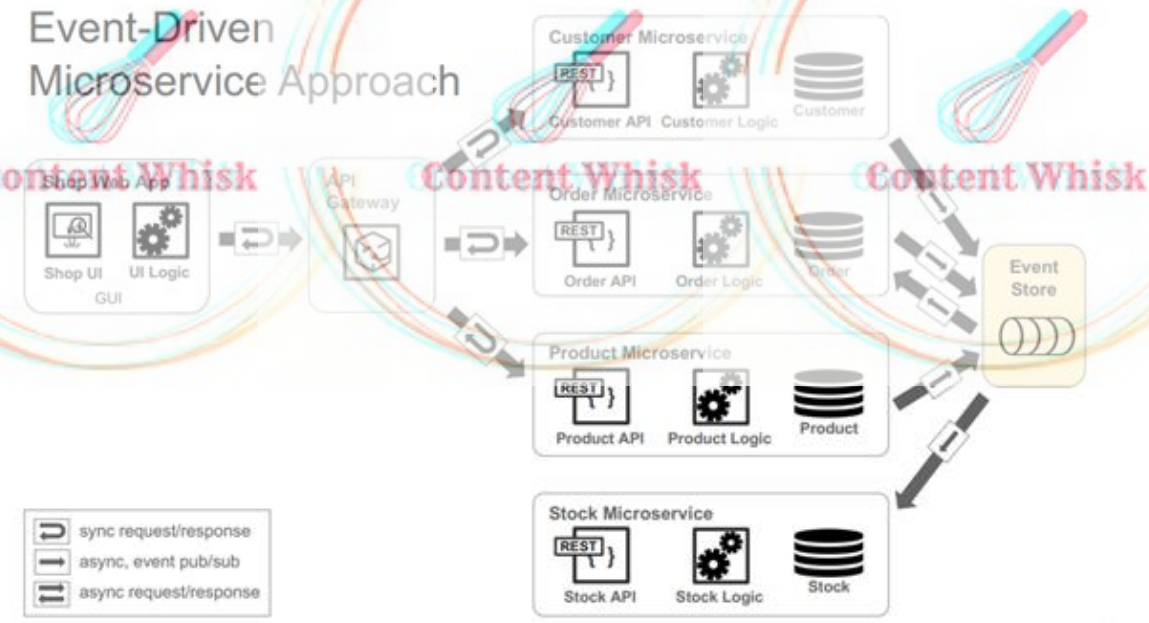

Image Source

How Event-Driven and Micro-services Architectures Collaborate for a Cohesive Event-Driven Microservices Architecture?

Events Do Most of the Application's Heavy Lifting

Methods where the logic has to be written into the ingestion code, relying on inadequate techniques such as log scraping that is implemented by traditional file systems, usually pose portability and scalability issues. By enabling the underlying infrastructure to handle this heavy lifting using events, you can focus on the key business logic of your applications.

Automatic Foreign Key Constraint Update between Services

If there is a user-service and a corresponding search-service which is optimized to perform searches based on the data of the user-service, adding, deleting, or modifying data in user-service will automatically trigger the respective events to concurrently update the search-service and update the function with the dynamic data set using the foreign key.

Distributed Transactions Make Disaster Recovery Easy

When multiple transactions occur through multiple services simultaneously, any issues during execution will be identified by the particular event for the defaulting service, and the rollback will be carried out automatically.

Less Service Coupling

Information exchange between two services is made easy since the two do not need to be updated about each other. The inner complexities of how to reach the other service and get a response are handled by the events triggered for those services.

Improved Scalability

In the traditional Microservices architecture, all the inter-services communication happen using a request-response mechanism. If one service is slow, the caller stack of that service gets slow.

EDM eliminates this through its scalability factor where the capacities of services grow dynamically on a need basis.

Services are smaller/simpler
Each service is not required to have complex error handling for downstream service or network failures.

Enables fine-grained scaling
Each service can be independently scaled up or down based on demand. This ensures good user experience and is less wasteful of computing resources.

Events Optimize the User Experience
User experience is better because events are arriving in real-time, and it will save your business money as bandwidth and computing resources aren't wasted on time-consuming requests.

Additionally, EDM combines the following benefits to the ones mentioned above.

- **Message mediation -** Manipulates the message content, direction, destination, and protocols with message flow configurations
- **Service virtualization –** Wraps existing systems or services with new service interfaces
- **Protocol conversion –** Bridges different protocols, e.g. JMS to HTTP
- Support for enterprise integration patterns (EIP) – EIP is the defector standard for enterprise integration
- **Quality of service –** Applying security, throttling, and caching
- **Connecting to legacy and proprietary systems** using a Business adapter
- **Uses Connectors to cloud services and APIs** like Salesforce, Twitter, PayPal, and more
- **Configuration driven –** Most functionalities are driven by configuration, but not code
- **Extensibility –** There are extension points that can be used to integrate with any custom protocol or a proprietary system

Conclusion

Adopting an Event-driven Architecture (EDA) to the Microservices architecture lowers up-front costs while decreasing time-to-market. EDA extracts value from existing occurrences, limiting invasive refactoring or disrupting existing application development efforts. Implementing Event-driven Microservices yields intelligent, scalable, extensible, and reactive endpoints.

Our Microservices experts can help your business applications scale-up with the Event-Driven Microservices architecture. Call us today!