A Brief Review of Microservices Architecture

Pros and Cons in BSS Environment

May 2023

INDEX

dnext-technology.com
May 2023

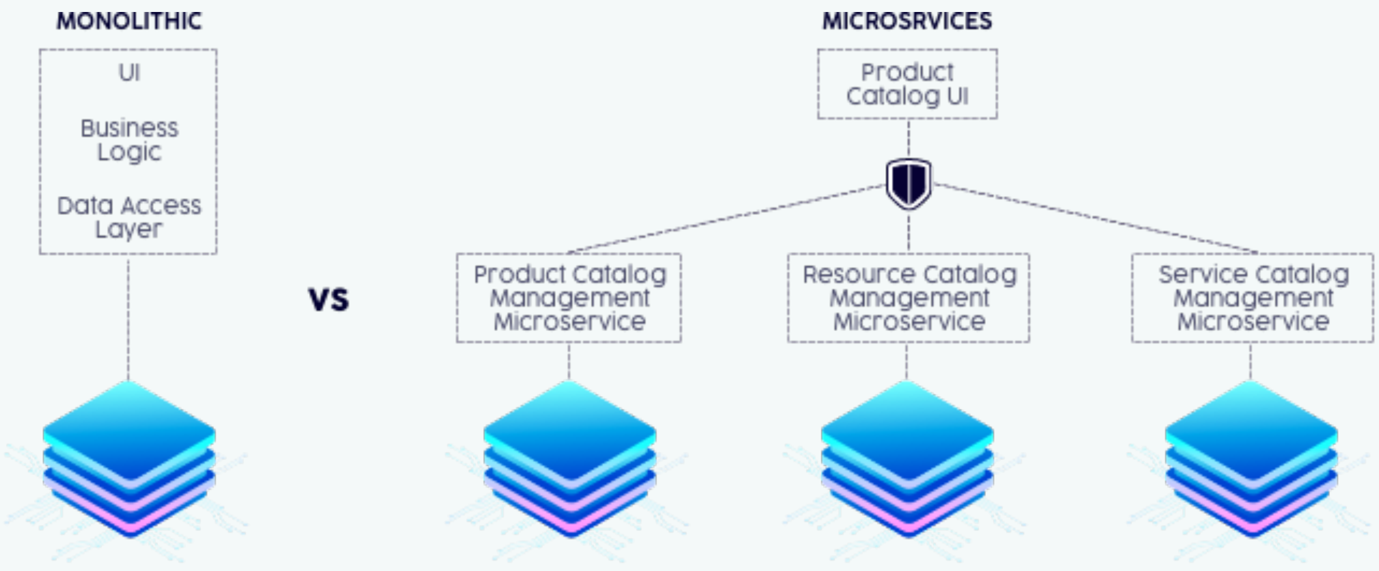
Microservice Usage in BSS

Microservices are used in the emerging World to build bigger and more dynamic systems that are best designed and operated as a mix of smaller services that work together coherently.

Microservices are self-contained, independent deployment modules.

They present a modular structure and bring many advantages compared to monolithic architecture. Microservices break complex structures into smaller pieces. Instead of using a single server, microservices also use separate servers. Thus, developers can work more efficiently. Also, it
provides deployment and technology flexibility while increasing scalability.

DNext harnesses the advantages of using reusable components. Reusable components introduce a new approach and allow fast customization of each module.


MONOLITHIC
UI
Business Logic
Data Access Layer
VS
MICROSRVICES
Product Catalog UI
Product Catalog Management Microservice
Resource Catalog Management Microservice
Service Catalog Management Microservice

Monolithic vs Microservice structures

Monolithic vs Microservices
Structures

The main benefits of using DNext microservices in BSS

- Allows meeting customer needs and your business priorities via rapid changes.
- Improves scalability, flexibility, and efficiency.
- Provides building unique solutions that can easily be integrated with existing applications
- Eases working on unique or complex products, sales, or check-out processes that require considerable customization.
- Reduces vendor/technology lock-in.
- Simplifies understanding and managing source code
- Allows smoother and faster deployments/updates while providing improved
- fault isolation.
- Easeas integration.
- Provides flexible data storage
- Increase Fault Tolerance and Fault Localisation
- Increases resilience and reduces the security risks due to independent microservices creating isolated domains
- Eases testing.
- Provides a faster release cycle.
- Allows partial service upgrades/changes instead of upgrading the entire application.
- The cost of scaling is comparatively less than the monolithic architecture.

On the other hand, **microservices bring some disadvantages as outlined below.** Actually, in this paper, we have decided to spend more time on disadvantages so that investment in favor of microservices needs to be considered in greater dept looking at these issues.

Disadvantage I : Microservices Are More Complex

Microservices increase communication, but the complexity comes with a certain amount of overhead. Microservices introduces a lot more distributed moving parts than traditional applications, requiring increased efforts, careful planning, automation to ensure communication, monitoring, testing, and deployment processes to run smoothly.

Disadvantage I : Microservices Are More Complex

- Developers need to solve problems that can arise due to distributed systems, such as network latency, load balancing, and communication between different services. Moreover, managing many services might pose difficulties.
- There is a higher chance of failure during communication between different services
- Each service requires individualized testing and monitoring, which means organizations also need to account for automation tools.
- The initial refactoring of a monolithic application is a massive undertaking for large enterprise applications.
- The number of processes can grow exponentially when load balancing and messaging middleware are considered.
- There are several microservices patterns to choose from, making it challenging to find the one that makes sense for your organization.

As it can be imagined, the more services included in applications, the more complex the entire development lifecycle becomes.

In DNext, we have done extensive research and development in the last five-six years in order to overcome most of the issues above. An optimized microservice design of DNext products makes effective communication between the services that ease the complexity while decreasing efforts for debugging and testing.

Disadvantage II : Data Inconsistency

Since microservices have independent data stores, transaction control can be an issue that could bring data inconsistency. To get over this problem DNext uses a supporting tool called ICC (Intelligent Consistency Checker). As a result, upon carefully architecting and ssing automation in DNext, weaknesses are converted into advantages by strengthening the controls on data. ICC as an integral part of DNext provides consistency, reliability, and accuracy while managing the data.

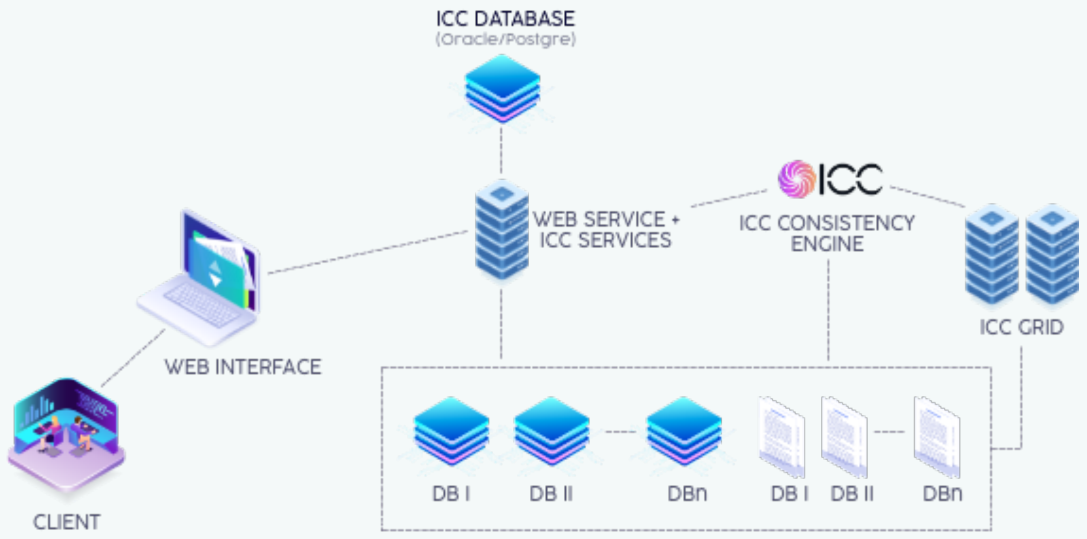
DNext ICC capabilities are outlined below

- Create jobs by using ready-made Capability templates
- Create rules /scenarios from jobs
- Schedule jobs/rules
- Process/monitor jobs/rules on GRID
- Provide actionable reports Alerts
- (SMS, email)

Web application is used to create consistency controls and define thresholds and alerts. Consistency controls are created by jobs and rules combinations. Job Editor is used to create jobs that contain algorithms and parameters that are necessary to create a consistency checks.

ICC Architecture Overview

ICC services will delegate rule execution to the ICC consistency engine. The engine will share workload between the ICC grid servers.



ICC provides following capabilities;

- Creating discipline for consistency with a centralized web-based user-friendly easy-to-maintain automation instead of writing SQL scripts.
- Mitigating risk of complaints from business units about consistency between reports.
- Saving test efforts for mitigation projects or new developments' impact on the existing platforms.
- Providing early warnings. Thus, IT can act proactively, and business knows the potential problems and act accordingly.

Disadvantage III : Microservices Require Cultural Changes

Any microservice initiative requires organizations to make changes to their internal culture. Before they can start the migration process, there should already be a mature Agile and DevOps culture in place.

Each microservice team operates as an independent business. The idea is that this structure allows
teams to work in small groups, work independently, and offer more control and increased productivity.

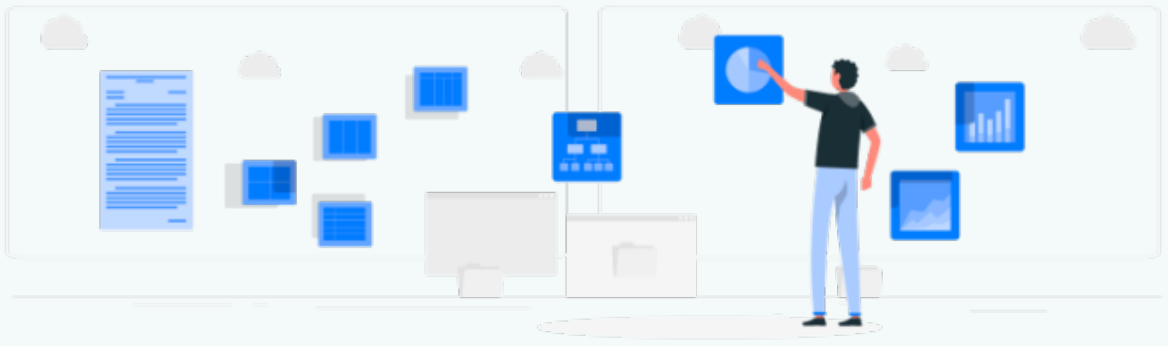
The challenge is that each development team must be able to manage the entire life cycle of the service, including maintaining a reliable API and managing an independent test suite.

On top of this, experts with DevOps and release automation skills are needed in each team. This is a prerequisite that will become increasingly important as you scale.

The first step toward successful adoption is to secure buy-in from the senior management. This requires educating leaders on how to carry out the initiative, rather than prevent their people from being productive.

And finally, organizations will need to consider the communications challenges associated with this undertaking. The independent microservices offer means that employees will not always have the big-picture visibility they would find in a monolith.

Individual services must work together to create a working application, and the inherent separation between teams could make it challenging to create a cohesive end product.



Disadvantage IV : Microservices are More Expensive Than Monoliths

Another potential disadvantage of a microservices architecture is the cost.

Services will need to communicate with each other, which results in a high volume of remote calls. This can increase network latency and processing costs, beyond what you might expect to pay when using traditional architectures.

Developers will need to account for this problem, creating solutions that aim to reduce the number of calls to avoid disruption.

Microservices also demand more resources, as each service is isolated and requires its CPU and runtime environment. You'll be bringing more tools, servers, and APIs into the fold, due o the lack of uniformity.

With each service using a separate programming language and technology stack, the demand for resources increases. Organizations must account for everything needed for the development, management, and maintenance of each module.

Disadvantage V : Microservices Can Presents Security Threats
Compared to monolithic applications, microservices come with some significant security challenges due to the sharp increase in the volume of data exchanged between modules.

Because you're working with multiple small containers, you're exposing more of your system to the network, which in turn means you're exposing more of your order to potential attackers. It's also worth noting that because containers are highly replicable, one weak spot in one module can turn into a bigger problem. Often, the source code is used across several applications, presenting easy access for hackers.

As a result, without the right tools and training, microservices can quickly turn into a security issue.

To sum up;
Although Microservices bring many advantages they may not be suitable for all applications. Breaking down a monolithic structure may not be worth the hassle if you have applications that are not mission critical or having low strategic value. However, in several cases, event in these circumstances, microservices architecture can be worthwhile.

When should you use Microservices?
- *When you thoroughly comprehend your company's issues and customer requirements.*
- *When it is possible to divide your application into smaller services.*
- *When your organization grows and the demands on their applications increase.*
- *When there are various technical specifications for each module.*
- *When you want to increase delivery speed, availability, and maintainability.*