Journal of Network and Innovative Computing
ISSN 2160-2174 Volume 5 (2017) pp. 020-024
© MIR Labs, www.mirlabs.net/jnic/index.html

# From Monolith to Microservice Architecture

**PoojaParnami[1], AmanJain[2], Navneet Sharma[3]**

[1]CS & IT Department, IIS University
Gurukul Marg, SFS, Mansarovar, , Jaipur, India
*pooja.parnami@yahoo.com*

[2]CS Department Maharishi Arvind Institute of Science and Management
Ambabari Circle, Ambabari, Jaipur, India
*amanjain.jpr@gmail.com*

[3]CS & IT Department, IIS University
Gurukul Marg, SFS, Mansarovar, , Jaipur, India
*navneet.sharma@iisuniv.ac.in*

*Abstract:* **Invention of cloud change the complete scenario of IT world. With the help of cloud, the developers can access infrastructure ubiquitously, cheaply and can scale up infinitely. The demand of modern business is agility and high availability, this results in the rise of micro-service based application. With the help of microservice platform, developers can create applications of properties like: scalable, independent lifecycle management high performance, high availability, cost effective and can run across public clouds and private clouds. Microservices are an application revolution powered by the cloud. This paper explains the journey from monolith to microservice based architecture, with the benefits and limitations of each architecture style.**

*Keywords:* Monolith, Component based, Layered, Object Oriented, Service Oriented, Microservice based Architecture, Cloud Computing

## I. Introduction

A monolith application is generally considered as a single unit. Generally enterprise applications are built in three parts : A client site user interface, a database and a server side application. The client side user interface is used for user interaction, the database is for data storage and server side application will handle the request sent by user, execute domain logic, retrieve or update data from the database and send back results to the user. This sever side application is monolith. To handle a request the logic is created in the form of a process. If any changes are required, we need to create a new server-side application. In case of horizontal scaling many instances of monolith application can run behind the load balancer but scaling of parts of application require greater amount of resources. Monolith applications are successful but as more applications are deployed on cloud it creates frustration among people. The reason behind this is tight coupling. If a small change is required in a part of application, it leads to rebuild and deploy of entire application. These frustrations have led to the microservice architectural style i.e. building applications as collection of services. Service is a module with a firm boundary, which could be written in different programming language, independently deployable and scalable. Each service could be managed by a different team. The IT industry travelled from the era of monolith to microservice to achieve different pinnacles of software development. This paper describes the complete journey from a monolith generation to microservice era.

## II. Monolith Architecture

A monolith application can be viewed as a single-tiered software application in which the user interface and data access code are combined into a single program. Monolith application is non-modular, self-contained and independent in nature.

### A. *Advantages:*

1. Simple to develop and deploy: Currently available all IDE tools supports Monolith development.
2. Simple to scale - multiple copies of the application can run behind a load balancer
3. Easy to implement
4. Easy to optimize performance.
5. No Context switching or compatibility issues.
6. Less development and management cost.

B. *Disadvantages:* Generally an application becomes large over time and the team size grows, the monolith designing approach starts showing significant drawbacks.

1. A large monolithic application is difficult to understand and modify, therefore development process slows down.
2. Continuous deployment is difficult –Update of one process or component result in redeploy of complete application.
3. Monolith architecture does not supports remote or distributed access of data resources.
4. Maintenance of Monolithic code is high in cost.
5. The cost of central mainframe used for running monolith applications is high.

## III. Client Server Architecture or 2 Tier Architecture

The two tiers of Client Server Architecture are a Graphical User Interface (GUI) application and a Database Server. The business logic is stored in the database in the form of stored procedure. The GUI application sends request to the database which returns the result in the form of different views. The example of client/server architecture is any web browser based application running on internet / intranet [1].

### A. *Advantages:*

1. Simple to deploy.
2. Higher security: All data is stored on the server, this enforce security.
3. Centralized Database :The data is stored in centralized manner, this results in ease of access and update of data.
4. High Performance: There are only two layers, database server and business logic, which enhance performance of the application.

### B. *Disadvantages:*

1. The business logic is stored in the database, therefore this architecture is not suitable for applications with rapidly changing business rules
2. Less reliable, because of total dependency on central server.
3. Scalability: Supports less number of users[2].
4. Modification overhead: To apply any changes in an application, it should be redeployed on all clients, this increases extensive administrative overhead. [2]

## IV. Component Based Architecture

A component could be defined as a exchangeable software unit with clearly defined interface. A component encapsulate related functions and has a clearly define interface. Component-based architecture focuses on the decomposition of the design into individual functional or logical components. Each component can communicate through methods, events, and properties. The main property of a component is abstraction i.e. each component hides its implementation and it is modular in nature. A component is easily extendable. Not much modifications is needed to make changes in the internal code or design of a component[3].

Components interaction could take place in the form of message passing, method invocations, asynchronous calls, broadcasting, data stream communications, and other protocol specific interactions. In short the component based architecture provides an infrastructure which uses mechanism like persistence, message-exchange, security and versioning. The examples of component platforms provided by different manufacturers are DCOM, JavaBeans, Enterprise Java Beans and CORBA.

### A. *Advantages:*

1. Ease of deployment – A new version of component can easily replace the previous version, without affecting other components and the system as a whole.

### B. *Disadvantages:[4]*

1. Increase in time and effort required for development of a component: Components are build for reuse. A reusable unit requires 4 to 5 times of time and effort as compare to other software unit.
2. Unclear and ambiguous requirement:-. A reusable component by definition, to be used in different applications. Most of the requirements may yet be unknown and cannot be predicted.
3. High maintenance costs:- maintenance costs of a component can be very high since the component must respond to the different requirements, different applications, different environments and different reliability requirements.
4. Difficult to find : Finding a suitable components which fit the architectural design of the software, may be difficult because of gaps between the software requirements and component's features.

## V. N-Tier / 3-Tier Architectural Style

N-tier and 3-tier architecture divided into layers and each layer is a separate tier, could be located on a physically separate computer. In N-tier application architecture the functionality of the application is decomposed in the form of service components, deployed in distributed way. This distributed property increases scalability, availability, manageability and resource utilization. Each layer is completely independent from other layers, except for those immediately above and below it. Communication between tiers is typically asynchronous.
N-tier application generally refers to 3 or more tiered application, but 3-tier architecture is most common. A 3-tier application divided into three layers: presentation layer, business rules layer and data layer. The presentation layer contains the user interface of the application. It does not make any application decisions. Business rule layer contains the application logic and also stores and retrieve data from the data layer. The data layer stores data and also provides security, transaction management and data mining facilities.

### A. *Advantages:*

1. Improved Scalability: Due to the distributed nature, scalability of the system increases[5]
2. Enhanced Re-usage: A similar logic can be sustained in many clients or applications. In appliance of object standards like COM/DCOM or CORBA, the language in the business-logic tier can be made transparent.[5]
3. Improved Data Integrity and Security: The client cannot directly interact with the database, the middle tier between User interface layer and database layer ensures data validations. The placement of the business logic on a centralized server improves the security of the data [6]
4. Reduced Distribution: In the case of modification in business logic, due to layered architecture updations are required to be done only at the application servers, not all distributed clients[6]
5. Reliability :It is possible to recover the system from network or server failures, due to availability of redundant servers.[6]

6. Database transparency :The schema design of the database remains hidden from users, which enables any change of the database to be transparent. [6]
7. n tier architecture is extension of 3-teir model, therefore includes all the advantages of 3-tier model. But the performance increases due to off-load from the database tier[5]

*B. Disadvantages*

1. Complexity of Communication: Due to presence of more layers, the communication become complex between client and server [5].
2. Fewer Tools: In comparison of 2-tier model, less automation tools are available in 3-tier model[5]
3. Componentization into tiers, makes the structure complex, which is difficult to implement and maintain[6].

## VI. Object-Oriented Architectural Style

Object Oriented Architecture is an important concept for developing the software. It is a design model based on dividing the responsibilities for an application or system into reusable and self-contained objects. Object oriented is based on modelling real-world objects. Each object encapsulates data and functionality relevant to itself. Each object is a discrete, reusable, loosely coupled unit. These units communicate through message passing, interfaces, calling methods or accessing properties of other object.

*A. Advantages:*

1. Understandable. Maps the application to the real world objects.
2. Reusable. Provides reusability through polymorphism and abstraction.
3. Testable. Provides testability through encapsulation.
4. Extensible. Encapsulation, polymorphism, and abstraction ensures data independence.
5. Highly Cohesive. The encapsulation helps in keep only related methods and features in an object, and creating different objects for different sets of features, this results in high level of cohesion.
6. Robustness: The errors could be managed during execution.

*B. Disadvantages:*

1. Not all problems could be broke-down in classes and objects.
2. Due to inheritance, the superclass and subclasses imposes strong coupling.
3. Not the a good choice for small and complex projects.

## VII. Service-Oriented Architectural Style

The principle of service-oriented architecture is to build real world distributed application. It is not a technology but set of architectural principles. The service-oriented architecture design the functionality of a software with services. A service is a self-contained, loosely coupled and un-associated unit. In Service-oriented architecture (SOA) application functionality is provided as a set of services, and the application is crated with the help of software services.

In SOA style business processes are packaged into interoperable services. These services implement variety of protocols and data formats to communicate information.

*A. Advantages*[7],[8]:

1. In comparison to large applications services provides free flow of information between and within enterprises.
2. Increase in business agility due to seamless connectivity of applications and interoperability.
3. Alignment of IT around the needs of the business – Service based software is easy to change according to the need of business.
4. Using existing software modules help in reduction of cost, development time and time to market
5. The conversion of data from one format to other is achieved through automated filed mapping.
6. Data confidentiality and integrity achieved through encryption.
7. Parallel and independent applications development – Due to the reuse of services it is possible to develop parallel and independent applications.
8. Reduced vendor lock-in
9. Ability to develop new function combinations with the help and reuse of existing functions.
10. Reuse of existing assets.
11. Smooth migration from old architecture to new one.
12. Virtualization helps in scaling.

*B. Disadvantages:*

Following are the downsides of SOA[9],[10]:

1. Each time when a service invokes other services, every input validation is required. This increases the machine load and response time. Resultantly, reduction in overall performance of the system.
2. Services work on message passing mechanism. Every message must be acknowledged. Therefore sometimes the number of messages can reach up to a million. It is a challenge to manage huge population of services.
3. The implementation cost in terms of technology, development and human resource is quite high.
4. SOA is not suitable for following type of applications:
   a. Standalone
   b. Short lived
   c. Applications with asynchronous communication
   d. Homogeneous
   e. Applications with GUI based functionality

## VIII. MicroService

In Microservice architecture, an application is developed with a collection of small services called microservice. Each microservice has its own light weight process and communicate through a well-defined Interface[11].

A Microservice is:

*A. Independent*

1. Autonomous in implementing a minimal unit of work,

2. Autonomous in persisting its data,
3. Autonomous in its implementing programming language,
4. Autonomous in its deploy.

*B. Integrated*

1. Communicate with other services exchanging messages over the network,
2. Its work (and so implemented functionality) may depend on one or more other services.

*C. Lightweight*

1. Its process has not a relatively big footprint.

*D. Minimal*

1. normally a single service does not provide a considerable complex functionality, to get that various services have to be orchestrated.

*E. Pipeline-ready*

1. Naturally a Microservice is designed to be plugged into a processing Pipeline.

*F. Stateless*

1. Its request processing does not depend for the history of the previous requests.

*G. Independent*

## IX. Characteristics of Microservice:

*A. Easily deployable[11] :* In microservice architecture, the application is divided into services. But is any changes are made in a service, we need not to deploy the complete application, only modified service need to be redeployed.

B. *Organized around Business Capabilities[11] :* Before microservice, to divide the application into parts, the method opted was to divide it into layers. The microservice divide the application into services and each service organized around business capability, resulting the teams are cross-functional.

C. *Independent code base*[12]: Each service has its own software repository. The small size of code makes it fast in development, testing and refactoring. Startup time is very low and dependency among code is negligible.

D. *Independent Technology Stack*[12] : Each service can be made on different technology stack based on their specific requirements. It means there is no system-wise standardized technology stack, in which your code need to fit in.

E. *Independent Scalability* [12]: Each service can identify the bottlenecks and scale accordingly and if not required could be made non-scalable.

F. *Stable interfaces*[12] *:* Communication between services is standardized, using HTTP(S) , REST , JSON.

G. *Decentralized Data Management*[11] *:* In monolithic applications there is a single logical database for persistent data. In Microservice architecture each service manages its own database

## X. Limitations of Microservice[13]:

A. Developers must deal with the additional complexity of creating a distributed system.
B. Deployment complexity. The independence of services increases the complexity of deployment.
C. Handling multiple database and transaction management is a challenging job.
D. Testing and deployment of microservice based application is cumbersome. It needs to test a deploy each microservice as compare to single WAR file.
E. Developers need to create a mechanism for inter-service communication
F. Implementing use cases that span multiple services across teams, is difficult.
G. Absence of knowledgebase.
H. Not sufficient developer tools/IDEs are available for developing distributed applications.

## XI. Conclusion

Microservice is not a final and ubiquitous solution, but is a dominant approach for developing light weight services, delivered on web or mobile devices. In the era of cloud computing and agile development, the first need is scalability. With the help of microservices horizontal scaling could be achieved at run time. The transition from a traditional layered architecture to a cloud-based software, the microservice based approach can offer significant benefits for organizations. The microservices architecture enables companies to be much more agile and cut costs at the same time. The industry specialists suggest the IT companies and developers for smart use of microservice with an understanding of benefits and pitfalls.

## References

[1] J. Fong , R. Hui "Application of middleware in the three tier client/server database design methodology" *Journal of the Brazilian Computer Society*, 6(1), pp.50-64, 1999
[2] Thota, " Advantages and disadvantages of 2-Tier Architecture" http://www.dotnetspider.com/forum/32148-advantages-disadvantages-two-tier-architec.aspx. Accessed 21 December 2017
[3] M. Panunzio, T. Vardanega "A component model for on-board software applications". In Proceedings of the *36th Euromicro Conference on Software Engineering and Advanced Applications, IEEE Computer Society*, pp. 57–64, 2010
[4] I. Verma "W model of component based software development." *International Journal of Advanced Studies in Computers, Science and Engineering,* 3(7) pp37,2014
[5] Sarma "3-Tier Architecture " http://www.dotnetspider.com/forum/32148-advantages-disadvantages-two-tier-architec.aspx. Accessed 21 December 2017
[6] Woodger Computing Inc. (2010) Multi-Tier Architectures. http://www.woodger.ca/archmult.html. Accessed 21 December 2017
[7] B. Johnson(2003)The benefits of service oriented architecture. http:// objectsharp.com/ cs/ blogs/ bruce/ pages/ 235.aspx. Accessed 21 December 2017

[8] Z. Mahmood "The promise and limitations of service oriented architecture" *International journal of Computers*. 1(3) pp.74-78, 2007

[9] G. Hohpe "Developing Software in a Service-Oriented World". In BTW , pp 476-484, 2005

[10] D Overall, Have we been there before, *Opinions, Computer Weekly*, UK., 2006

[11] M. Fowler Microservices https://martinfowler.com/articles/microservices.html. Accessed 21 December 2017

[12] A. Schroeder (2014) Microservice Architectures https://www.pst.ifi.lmu.de/Lehre/wise-14-15/mse/microservice-architectures.pdf. Accessed 21 October 2017

[13] C. Richardson (2016) Microservice Architecture http://microservices.io/patterns/microservices.html. Accessed 21 December 2017

## Author Biographies

**Pooja Parnami** is an assistant professor in Amity University, Rajasthan. She is working as a research scholar at IIS University, Jaipur, India. Her main research area is cloud computing with strong focus on cloud migration. She has published a book on database management system. She severed in various institutes and software industries as a faculty, developer and researcher.

**Aman Jain** is a Professor in Maharishi Arvind Institute of Science and Management, Jaipur, India. He completed his PhD from University of Rajasthan. His main research areas are in Databases, ERP and Network with a strong focus on Securities. He served various institutes and guided numerous scholars in their research and projects

**Navneet Sharma** completed his PhD degree in Computer Science from Suresh GyanVihar University, Jaipur, India. His general research interests are Banking, E Commerce and ATM Securities. He has published extensively in areas of Banking and Securities. He was awarded with Bright Researcher Award from NITTR Chandigarh and received various other awards in the field of research and teaching