**Pros and cons Monolithic architecture based Applicaiton Development**

**=====**

**pros (advantages)**

**=> Provides simple and easy env.. to develop Projects/Apps as the Layered Apps having**

**layers like presnetation layer, controller layer, service Layer, Persistence Layer(DAO), Integration layer and etc..**

**note: Most of the current maintainance projects are Monolithic projects**

**=> Easy to deploy (all togather single jar/war file), easy to manage (single war/jar file), easy to scale (increasting or decresing no.of instances)**

**=> Performing unit Testing is very easy becoz all layers and all services are avaiable togaher**

**=> Less possiblity of getting network related issues becoz all services /layers mostly there togather.**

**=> Performing logging and dubegging operations is very easy.. (Even Adutiing activities are easy)**

**=> Getting complete grip on the Project is possible**

cons (DisAdvantages)

====

**=> For small or tiny changes in one or two services /modules/layers we need redeploy the application by alterting the endusers when App in the production. (App will not for these many hours)**

**=> one bug or issue in one module/service of Project/App may effect other services /modules ..this indicates these Apps are not reliable. (Sometimes Apps will shutdown providing no services to clients)**

**.Project**

**=> Adding new technologies/concepts/ frameworks in the existing as part of enhacement is very complex sometime we will be forced to redesign the project..**

**=>Logging keeps track of Code flow**

**=> Auditing keeps track of User activities**

**=> all the services /modules of the Project must be developed in the same programming language i.e more programmers of certain language is required**

**=> if no.of modules/services are incresed in the Project then their maintainace towards bug fixing and redeployment also takes lots of time while working with webserver and application server .. (This may cause increasing App/Project downtime )**

**=> we can not sell certain services/modules of one Project to clients as they need...**

**=> Since we can create service instances only for entire App .. not for certain services/modules of then the Project lots of memory and CPU time wastage will be there, as shown below**

**and etc..**

**Application means Project service means module**

**Monolithic Project/App (Flipkart)**

**Monolith Monolithic**

**Service Instances created for the Project/Application**

**(Horizontal scaling)**

**request**

**capacity of each instance 250**

**Flipkart (120MB) (London) (instance1)**

**50 req:: cartService 50 req: search Service**

**war**

file

**(current load request/ max requests)**

**search servide (20MB)**

**cart service (20MB)**

**Order Service (20MB)**

Trading

**Return & Replace**

**mentService**

**(20MB)**

**Rewards (20MB)**

**Service**

**Total Project/App size is :: 120 MB**

**note:: since we are creating service instances for total project/App**

**we are wasting memory in eaach service instance though**

**we are not using some services of each service instance**

**Let us assume we are able to create seperate project/App for every service/Module**

**search service**

**(war file) 20MB**

**(indirectly MicroService architectecture)**

**Flipkart (120 MB) (Dubai) (instance2)**

**(instance3) Flipkart (120MB) (singapore) Flipkart (120MB) (Blore) (instance4)**

**100 req:: search 100 req:: orders**

**50 req: cartService 50 req: search service**

**100 req request cart service 50 req tracking service 40 req search service**

**(Horizontal scaling by taking**

**modules/sericces as the Projects/Apps)**

**50 req**

**Total Project Info (All instance togather)**

**Size :: 480 MB (4*120MB)**

**currently serving : 590 requests max capacity :: 1000 request s (4*250=1000)**

**Load Factor is :: 0.59**

**(Here instances are created for the projects)**

**Cart service**

**(war file) 20MB**

**request capacity of each instance is 250**

**Search Service Instance1 (20MB)**

**100 req**

**Order service (war file) 20MB**

**Return & Replacement**

**service**

**Tracking service**

**(war file) 20MB**

**Search Serviqe 200 req Instance2 (20MB) Search Service Instance3 (20MB)**

**200 req**

**Reward search service**

**(war file) 20MB**

**(war file) 20MB**

**Order Service 50req Instance1 (20MB)**

**RR Service Instance1 20 req (20MB)**

**Cart Service Instance(20MB)**

**Cart Service 70 requests Instance2(20MB)**

**Tracking instance1 50 request (20MB)**

**Total Project Info**

(All service Instance togather).

**Size of project :: 10* 20 MB 200 MB current count of requests :: 870 fequests max cap of requests: 2500 requests. (10 * 250 =2500)**

**are**

**Differnt services of Flikart developed as**

**different Projects (war files)**

**=>This kind of Application development in Monolithic architecture is possible but very complex to implement and manage .. To overcome this problem we have MicroServices Archicture..**

go for

**SOA (Service Oriented Architecture)**

**Order Service Instance2 (20MB)**

**100 req**

**RewardService instance1 20MB)**

**30 reqs**

**(Here the instances are created for the modules by treating them projects)**

**SOAP based web service fall under SOA Restfull webservices fall under Producer- Consumer mechanism**

**of**

**note:: MicrorServices Archicture is enhacment Producer-consumer mechanism where we develop each**

**microservice as restfull App What is the link between webServices and MicroServices?**

**Microservices are extension of webServices (Restfull webservices)**

**In fact each micro service will be developed as one Restfull webservice adding other facilities.**

**=> MicroService architecture says develop every service as one restful service/API and intergrate them using various tools and design patterns of MicroServices.**

**Microservices App= mutiple restful services + design patterns + third party tools**

**Monothilic vs SOA vs MicroServices archicture**

**SOA (Service Oriented Architecture)**

==========

is

**=> This architecture given to get communication between two applications that**

**are developed in two different technologies and running from two different machines using Register and Discovery Concepts..**

**(non-java)**

**=>SOAP based webServices (like Jax-ws, axis,apche cfx and etc..), CGI-Links,webmethods, TIBCO and etc.. given as the implmenentation models of SOA Design..**

**=>SOA is design that provides certain archicture with principles to get communication between two incompitable Apps..**

of

**3 comps SOA**

**Service Registry**

**a) Service Provider**

**b) Service Consumer**

**1**

**c) Service Registry**

**Publish**

Service details

**2 Find**

**=>SOAP based web services are given based on**

**SOA (Service Oriented Architecture)**

**=> Restful web services are given to develop Micro**

**services Architecture based apps (or) we can develop the independent APIs (webservice APIs)**

**SErvice Provider**

**Ervice Consumer**

**[Skelton -Code]**

**3 request Bind response**

[ Stub -Code]

**Xml**

**1.Service Provide develops the prodiver App and exposes it details to serviceRegistry in the form of xml docs (Publish Operation)**

(wsdl doc)

**2. TheService consumer contacts the ServiceRegistry and get the details about provider or service (Find Operation) in the form of wsdl doc(xml)**

**3. The service consumer prepare stub code to consume services of the Provider using request-response model (Bind Operation)**

**Advatntages of SOA**

d

**1) Interoperability :: we can develop the Provier and consumer Apps either in same technology or in different technologies**

**2) Easy Maintainace :: Any change in provider App, we just need to update to**

**Service registry and need not inform to all consumers..becoz Service cosumer can collect the changes from Service Registry**

**=>The Integration layer code of Service Provider is called Skelton logics**

**=>The Integration layer code of Service Consumer is called Stub logics**

**note:: The stub and skelton logics actually interacts with each other in App to App interaction.**

3) Quality Code

is guaranteed ::

**Since we can develop provider App in our choice technology So we can give quality provider App..**

**4) Scalable ::**

**5) realiable::**

**DisAdvantages ::**

**we can add more parellel servers or instanstaces for provider App for giving better service to more consumers and their requests**

**Since the SOA machanism is involving the Service Registry.. we can maintain our Distributed App in reliable and stable state..**

**HighCost:: It needs lots of man power and technology/infrastrucure support for developing SOA applications**

**Xml utilization :: Supports only XML based Data exchange which is fading out day to day**

**HighBandwidth**

**of internet required :: Since service registry,Service Providers will be placed in different locations we definelty need good bandwith of internet connections.**

**note: To overcome all these problems prefer using MicroService architecture**

**MicroServices Architecture**

==========

το

**(Extesion Restufull web service model)**

**=>In this architecture every service /module will be developed as seperate project/App**

**having connection with other service/module..**

**=> It is Decoupled Archicture of a single Project becoz multiple services/modules will be devleoped**

**as multiple projects and they will be registered/Integrated in a common place.**

**=> Micro Service is small service or small application**

**micro = small**

**service = project/ Application..**

**Every small service/module**

**is developed as the separate project**

**Monolithic architecture**

O

module#1

**module 2**

**module#3**

**module #4**

**single project as war /jar file**

**MS#1 (war1/jar1)**

**module#1**

**MicroService Architecture**

**=> In restful web servies no registration of services becoz they**

**are identified and invoked through end point details (urls)**

**=> In microservice archicture Apps, the services will be developed as restful web services but they will be registered in the common Registry to make them discoverable for the microservices**

**MS#2 (war2/jar2)**

**module#2**

**That common registry is nothing but Eureka server)**

Eureka sever

UI

**DB s/w**

**DB s/w**

**MS#3 Module#3**

**MS ::: MiroService (Restful Service)**

**MS# 4 (war3/jar3)**

**war4/jar4**

**DB s/w**

Module#4

**DB s/w**

**note:: UI can be web application or mobile App or destktop app or IOT App and etc..**

**Every Micro service will be developed as Restfull App /RestController and will be place in common registry/server to make it avaiable for other micro services or Client Apps.**

**(Eureka server)**

**Advatnatages of MicroService Architecture**

**=======**

**==========**

**=> Need not stop other services /modules (MicroServices) if problem comes in certain module/service**

**[if Rewards or Feebback sevrice $failed it does not effect other services like**

**to**

**sales, orders, cart and etc..]**

**=> Need not stop all other the services.. while updating/patching certain service for betterment**

**or languages**

**=> We can use different Technologies to develop different services of the Project/application..**

**=> We can do horizontal scaling for each service as needed i.e if certain service like searching,**

**cart and etc.. are having more demand then we can create more instances for them by enabling Load Balancing**

**=> Updgrading/Migrating to new tehchnologies in each service is no complex .. Quite easy compare to monolithic architecture**

**is**

**=> Service Down time (while performing reloading or restart or redployment activities) less becoz we need to perform these actitvies on one service at time.**

**=> Allows to place common things of multiple Services (MicorServices) in a common place like GIT hub env.. instead of placing in every service/module**

**note:: Developing MicroServices using spring boot is very easy becoz most of the common things like DB setup, Schema initializations and etc.. can be cfg to occur through auto Configuration of spring boot (db tables generation)**

**Limitations of MicroServices**

====

**=>Maintainance of the Project is very complex becoz it generates multiple log files, contains multiple communication chains and debugging the code across multiple services is too difficult..**

ur

**=> Needs high end intrastruture to maintain multiple micro services, if going for Cloud**

**we need to purchase /rent muliple things to deploy and manage these projects. (multiple microservices)**

**=> Knowlege on More Tools, technlogies and processes is required to work with MicroServices Projects**

li

**=> Migrating Monothic Project into MicroServices project is very complex**

**=> Testing is bit difficult if one microservice is having dependency with other micro service.**

**=> Transaction Management across the multiple microservices is very complex/difficult to implement.. (Global or distributed Tx Mgmt is required) (or) SAGA PAttern**

**note:: we generally use: Cloud env.. to deploy and execute the modren monolithic, SOA and Micro services**

**projects.. Working with with Cloud is nothing but taking things on Rental basis and using them for our requirements..**

**note:: While developing any architecture based Project/application we need to use MVC Layers..**

**Each MicroService is one @RestController**

**Project (nothing but Restful webservice API)**

**=> TL, developers can not get complete control on the Project as the Project contains multiple microservices implemented in the different technologies**

**=> Bounded context problem**

**(indentifying boundries of each MicroService and deciding the no.of microservcies is very difficult)**

**eg:: AWS, Azure, Goolge clould, PCF (Pivotal Cloud Foundary), RackSpace, salesforce and etc..**

**Clould env.. (AWS/Azure ...)**

(decommissioned)

**Cloud services**

**a) PAAS :: Platform As A service**

**b) IAAS :: Infrastrutur As A service**

**=>BFSI domain projects**

**contain multiple microservceis**

**=> BFSI :: Banking Financial Services and Insurance**

SAAS

**c) SAAS: Software As A service**

**======**

**=>web server/App servers**

**IAAS**

**PAAS**

=====

**RAM HDD network CPU RACKS**

**Linux windows**

**Device**

**Ubentu .MAC Drivers**

**=>DB s/w s =>Mail servers**

**=JDBC drivers =>MQ Services**

**=>Jdk s/w**

**=> Jenkins**

=> Docker

...

a

**=> MicroServices architeture is "Design and whose Apps/Projects can be implemented/ developed in spring env.. with the support of spring boot + spring clould + tools + design Patterns /Spring boot**

**boot**

`boot`

**=> MicroService Architeture App = spring Rest Apps + Desgin Patterns + third party tools**

**=> Spring Cloud module with spring boot provides better env.. to develop and test MicroService Architecture Applications.**

**=> we can develop MS architecture Apps in other domains like .net, python, js and etc.. but Java domain is the best for it becoz it has go spring boot and spring could support**

**to develop MS architecture projects effectively..**

**The MicroService architecture projects can be implemented in mixed technology mode**

**Conclusion:: prefer using monolithic archicture for small and medium scale java applications/projects**

eg:: college websites, university websites, supermarket apps and etc..

**prefer using microService architeture for large scale -complex.**

**applications/projects**

**eg:: e-commerce apps, banking Apps, Financial services Apps and etc..**

**=> Monolithic architecture says develop different services as the different modules by combining them**

**in to a single war file/jar file (It is like united family)**

**=> SOA architecture says develop different services as the different Projects and register them in**

**the common UDDI registry and the The consumer and producer app interacts with each other using the xml global format (It is like Faculty --> Institute ---> Student kind of env..)**

**=>Microservice architecture says develop different services as the diffrent Projects and integrate them using third party tools and lots of Design Patterns. Here the data exchange b/w Apps takes place either using Xml format or using JSON format (becoz the microservices are restfull apps internally) (It is like nuclear families belonging to same surname)**

Load Factor is :: 0.348