Software Architecture

» Skan.ai - chief Architect

» Ai.robotics - chief Architect

» Genpact - solution Architect

» Welldoc - chief Architect

» Microsoft

» Mercedes

» Siemens

» Honeywell

Mubarak

1. Arch Requirements
2. Arch Design & Doc
3. Arch Eval

Architecture vs Design

Quality	Metrics	Approach
• Performance	• Response time	• Caching
• Scalability	• Throughput	• NIO
• Availability	• tps	• Compression
• Maintainability	• Latency	• JIT (Lazy loading)
• Security (Trust)	• Resource usage	• Extensibility
• Reliability (Trust)		• Testability
• Robustness		• KISS
• Portability		• Doc
• Interoperability		• Observability
• Cost, TTM		• ACID
		•

Which Quality &
How Much ?
Collect
Decide
Approaches
(Architectural
Decisions)

5 Communicate

Pre

Engineering vs Tuning

Post

Performance engineering

Performance Tuning

Threat Modeling

Pen Testing/ Ethical Hacking

Architecture
(Decisions)
Understand
Functional
Requirement
Understand
Create
Skeleton for Code
<<code design>>
Knows
OO, functional, proc,
Code Design Pattern,
Code Anti Patterns,
Code Design Principles,
Idioms

Architecture vs Design

System quality

Architecture Design
High Level Design
Blue Print
System Design
...

Code Maintainability

Detail Design
Module Design
Class Design
Low Level Design
Code design <—
Implementation Design
...

TOGAF,
DODAF,
Zachman

Security
Infra
SEO
Data
Cloud
Technology
ui
network
...

Address quality of application

Address quality of process

Architectural Requirements

 # Context View
 # Functional View
 # Constraints
 # Quality View *
 # Assumptions

Context View

Self Driving Car Rental

..
Seller
Payment Gateway
System
(to be built)
Buyer
Shipping Service
Admins
Notification Service

Guest
Payment Gateway
System
(to be built)
Host
Analyst
Admin
Customer Support

Functional view

Self Driving Car Rental

Manage Booking
Search Property
Rank Property
Manage Property
Guest
Host
Customer Support

Constraints view

Must support popular Desktop OS and Mobile OS

Should work on a low bandwidth network

Use Opensource only

Use Postgres database version xyz

Use GCP cloud

Yearly Cloud Billing should not exceed 1 million dollars - cost

Should be in production on April 1st 2023 - (next financial year)

Data should be kept for 8 years

Self Driving Car Rental

Should work on Azure Cloud

GDPR Compliance

Quality View

Usability
Portability
Interoperability
..

Seed Quality Scenarios

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a user	I want to upload a 2 MB file	From Mobile Client	During Peak Load	The File is uploaded	In < 3 secs
Maintainability	Developer	Want to replace the storage	In the cloud API	During maintenance	The storage is replaced	In < 20 man days
Security	unknown identity	requests to download a file	In the portal	During Normal load.	block access to the data and record the access attempts	100% probability of detecting the attack, 100% probability of denying access
Availability	The Database	Failed	In the Data Centre	During Operational Hours	Secondary is made Primary	In < 2 minutes

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Availability	Processor	Stops working	Central System	Peak hours	Notify Provide degraded mode	< 5 min degraded mode
Reliability	Consumer WebSite	Send purchase order request	To Xyz App	Duplicate Request	the consumer is not double-charged and one POS is placed.	99% of the time
Maintainability	New Display System	Is Installed	Int the Bus Stop	Off Peak hours	The Display system starts showing bus arrival information	within 30 minutes

Assumptions View

Will use open source Technologies only

For offline scenarios in native app or any of services going down in backend last available timeline will be visible.

System will be eventually consistent.

QAW

- Stake holders, dev, ops, QA, BA, …
- half day
- Prepare Seed scenarios(lots example scenarios *)
- Collect scenarios
-

Drivers

Context View
Functional View
Constraints
Quality View *
Assumptions

Design

Logical View *<—
Security View
Deployment View
Infrastructure View
…

Eval

Justification

Logical View

System
Application type
App 1
App 2
Apply Styles
M1
M2
App 2.1
App 2.2
App 2.3
Apply patterns
M3
M4

Choose System Decomposition Patterns

Invoke logic()
Layer 1
Data1
Layer 2
Data2
Layer 3
Data3
Layer 4
Data4
Data5

Process Data()
Filter 1
Data1
Pipe
Data2
Filter 2
Data3
Filter 3
Data4
Filter 4
Data5

Eclipse IDE. Downloading the basic Eclipse product provides you little more than a fancy editor. However, once you start adding plug-ins, it becomes a highly customizable and useful product.

An application is required to perform a **variety of tasks** of **varying complexity** on the information that it processes. The processing tasks performed by each module, or the deployment requirements for each task, **could change** as business requirements are updated. Also, **additional processing** might be required in the future, **or the order** in which the tasks performed by the processing could change. A solution is required that addresses these issues, and increases the possibilities for code reuse.

A workflow implementation. The implementation of a workflow contains concepts like the order of the different steps, evaluating the results of steps, deciding what the next step is, etc.

A task scheduler. A scheduler contains all the logic for scheduling and triggering tasks

Internet browses plug-ins add additional capabilities that are not otherwise found in the basic browser

Networking engineering is a complicated task, which involves software, firmware, chip level engineering, hardware, and electric pulses. To ease network engineering, the whole networking concept is decomposed into more manageable parts.

Presentation Layer
Component
Component
Component
Business Layer
Component
Component
Component
Persistence Layer
Component
Component
Component
Database Layer

Rest Api
Message
Controller/facade
Timer
Logic
UI
Mail
API Proxy
Data access layer

UI
Update order
Rest Api
Domain Logic
Entity
Data Access Layer
API Proxy
Boundary

Boundary
UI
Update order
Schedule
Rest Api
Message
Delegate
Control Flow
Commons
Data Access Layer
API Proxy
Domain Logic
<<POJO>>
Domain Flow
Steps (actions)
Domain rules
Entity
Hexagonal
Boundary

Layered Architecture

Pipes and Filters

Microkernel Architecture

View
Input
*
1
Controller
Model
MVC
View
Input
1
1
Presenter
Model
MVP
View
Input
*
1
View Model
Model
MVVM

	Layered	Pipes and Filter	Micro Kernel
SOC	layer	Filter	common/Variance
Low Coupling	- -	+	+ +
Does not need Abstraction	+ +	-	- -
Ease of Design	+ +	-	- -
Agility (Shuffle, Rearrange, Add)	- -	+	+ +
Unit Testability	- -	+	+ +
Needs less domain knowledge	+ +	-	- -

Distributed Communication Patterns

SIMFORM

Peer-to-Peer architecture

SIMFORM

Client
Broker
Server 1
Server 2
Server 3

https://www.enterpriseintegrationpatterns.com/patterns/messaging/

Microservice B
Microservice A
Microservice D
Users
Orchestrator
Microservice C
Microservice E
Microservice F

UI
Request
Service A
State Z
Response
State X,Y,Z
Request
Service B
Response
State X
Response
State Y
Request
Service C

Users
Event Broker
Microservice A
Microservice D
Microservice B
Microservice E
Microservice C
Microservice F

Hybrid

Microservice C
Event Tracker
Microservice B
Users
Microservice A
Orchestrator
Microservice D
Microservice F
Microservice E

Event
Stream

Apply Architecture Principles

Microservice

1
1980

Ship one binary

One Large exe file

Compile time monolithic

2
1980

Ship one binary

One Large exe file

Link time monolithic

3
1990

Ship multiple binary

COM

Process

Runtime monolithic

In process comp

Ship multiple binary & host in multiple process

4
1998

DCOM
CORBA
RMI
EJB
...

Process

Distributed comp/module

Process

Distributed comp/ module

Distributed Application

5 (4++)
2015

Process

Application1

Process

Application2

Distributed Application (Micro Application)

Service Oriented Architecture

ToDo Portal
(Single Page Application)
ToDo API Service
(Api Application)
ToDo Calendar Service
(Background Application)
ToDo Prediction Service
(Background Application)

SOA relates to ***enterprise*** *service exposure*
Application
Application
Micro-service
Micro-service
Micro-service
Micro-service
Microservice application
Application
Microservices relate to **application** architecture

View
Input
*
1
Controller
Model
MVC
View
Input
1
1
Presenter
Model
MVP
View
Input
*
1
View Model
Model
MVVM

Break an app into smaller manageable piece

	2 Modules	**2 Applications**	**W**	**Score**
Share Database / Storage	Yes	No *	2	
Share same Virtual Infra	Yes	No	3	
Share Sorce Control	Yes	No	2	
Share CI/CD (Build pipeline)	Yes	No	3	
Share domain code	No	No	3	
Fun Requirements	Application level	It owns	1	
SCRUM Team / Sprint	Application Scope	Its owns	1	
Acceptance Test Cases	Application Scope	Its owns	1	
Architecture	Application Scope	Its owns	1	
Technology Stack / Fwks	Application Scope	Its owns	1	

3 or 4

Application
Small App
Small App
Small App
Modules
Modules
Modules
Modules
Modules
Modules

App
Inventory code
Binary
Accounting Code
Binary
App
Inventory code
Binary
App
Accounting Code
Binary

Inventory App
Accounting App

Data loss
Data Correction
Data …
Event data

Inventory Application

Inventory App
Accounting App

VM
Inventory App
Accounting App

Types of Microservice

UI App (SPA)
1
Web Server
API Layer
Domain Layer
<<feature1>>
Data Layer
Angular (SPA)
Browser
2
Web Server
UI Layer
(server page)
Domain Layer
<<feature2>>
Data Layer
JSP, Servlet, JSF (Server pages)
3
Desktop
UI Layer
(native)
Domain Layer
<<feature3>>
Data Layer
Swing, Android, Ios (Native)

4
Message Layer
Domain Layer
<<feature4>>
Data Layer
5
Schedule Layer
Domain Layer
<<feature5>>
Data Layer
Rest
6
Web Server
API Layer
Domain Layer
<<feature6>>
Data Layer

API
Rest, Grpc
* Message
Queue, Topic, Hub
Microservice
Schedule
UI

	Pros/	Solution
Development time	- -	
Micro-service practices Learning Curve	- -	
Resource Performance (CPU, Memory, I/O)	- -	Grpc, protobuf, ..
Db Transaction Management	- -	SAGA
Views / Report / Dash board/ join	- -	Materialized View
Infra Cost	- -	Containerization
First time Deployment effort	- -	Automation
Debugging, Error Handling (End to End)	- -	Distributed Tracing (Jaeger)
Integration Test	- -	
Log Mgmt (debug/ error)	- -	Centeralize EFK, ELK, Splunk
Config Mgmt	- -	Centralize Consul, Zookeeper, ..
Authentication (who)	- -	Centralize (Oauth2, OpenID connect, SAML, …)
Authorization (what can they do)	- -	Centeralize (Claims)
Audit Log mgmt (who accessed what)	- -	Centeralize (Event Store, …)
Monitoring / Alerting	- -	Centeralize
Data Security and Privacy (transit, storage)	- -	
Build Pipeline (CI/CD)	- -	Automation
Agile Architecture (Agility to change)	+ + +	fwk, technology, platform, …
Feature Shipping (Agility to ship)/ CD	+++	Timing of when to goto production
Scalability (volume - request, data, compute)	+	
Availability	+	
Ability to do Polygot	+	
Fault Isolation	+	

Inventory Application

1 phase commit

db.Begin()
Cmd
Cmd
Cmd
db.Commit()

2 phase commit

rm.Begin()
Cmd on db1
Cmd on db2
Cmd on db 1
...
rm.Commit()
-> get votes from all
-> commit all

Decentrailze Domain

Centrailze Tech

Pyramid test

Manual Test - 75%
UI Automation Test - 15 %
API Test - 10%
Integration Test - 15 %
Unit Test - 70%

1. Performance

Operation	Cpu Cycles
• 10 + 12	3
• Calling a in-memory Method	10
• Create Thread	2,00,000
• Destroy Thread	1,00,000
• Database Call	40,00,000
• Distributed Fun Call	20,00,000
Write to disk	10,00,000

Process
Mod1
Mod2
fun(){
....
}
10 cpu cycles

10,00,000 cpu cycles
Process
Process
Mod1
Mod2
fun(){
....
}

2. Db transaction

```
do(…)
{
   begin_tran()
   Update 1
   Update 2
   commit_tran()
}
```

2. Db transaction

App
App
App

App
App
App

App
Mod
Mod
Mod

App
App
App

2. Db transaction

2. Db transaction

2 phase commit
JTX , MSDTC, …

msdtc.begin_tran()
Update 1
Update 2
mstdc.commit_tran()
-> take votes
-> commit

3. Db query

3. Query

4. Development time

5. Learning Curve

6. Infra Cost

Micro App 1
Micro App 2
Micro App 5
Micro App 3
Micro App 4
Micro App 6
Micro App 7
Micro App 8
Micro App 9

7. Debugging

Dev & Ops Teams
Log Data
Metrics Data
APM Data
Uptime Data
Web Logs
App Logs
Database Logs
Container Logs
Container Metrics
Host Metrics
Database Metics
Network Metrics
Storage Metrics
Real User Monitoring
Txn Perf Monitoring
Distributed Tracing
Uptime
Response Time

Host (Linux)
Service Container (Docker)
Web Server (Nginx) + PHP
Application Request (eg: order)
SQL Query 1
SQL Query 2
Call another services request (Eg: customer, cashflow...)
OS log
Syslog
Access, Error log
API Request, Audit...
Application Log
SQL Query log
Distributed tracing

System
Decompose system into applications
*
Applications
Decompose application into small application
*
Micro Applications
Decompose application into modules
*
Modules

Service Oriented Architecture

Choosing Technology ?

APP
Storage
Compute
Communication
binary/append only
Random read/write
Files
Database
Run business (frequent update)
Manage business
Operational (OLTP)
Analytical (OLAP)
Run business
Manage Business
LOB
Analytical
Scale, Complex
Message
APi

Operational Application/
Analytical Application
Storage
Compute
Communication
binary/append only
Random read/write
Run
business
Manage
Business
Scale, Complex
Files
Database
LOB
Analytical
Message
APi
Object Storage
File Storage
Block Storage (Managed Disk)
Unmanaged Disk
Operational
(OLTP)
Analytical
(OLAP)
Queue
Topic
Hub
Rest
Ws
Grpc
RDBMS
key value
Document
Graph
Columnar db (column store)
Text Database
Wide Column db (column-family)
Time Series
Immutable Logger
geo Spatial

File Storage

* AWS DataSync subscription is required to provide support for Server Message Block (SMB) protocol

Files
JSON
PARQUET
CSV
XML
AVRO
RAW
Databases (Cloud / On Premises)
My
SQL
ORACLE
SQL
SAP DATABASE
Cloud Sources
AWS
Azure
Data Lake
Audio/ Video/Documents
PDF
DOC
WebApps, Sensors, IoT
Social Feeds

Data Source Tier
Structured
Database,
Data warehouse,
Lob App logs
unstructured
Images,
Audio,
Word doc,
Spread Sheet
semi-structured
XMI,
Html,
Json,
Emails
Landing Zone
(immutable)
handling missing values,
filtering outliers,
& Validated data
Highly refined views
of the data
Raw
Stage X
Stage Y
Curated
Data Lake
Azure Blob
AWS S3
Azure Data Lake
Data WareHouse / Mart
Aws Redshift
Azure Synapse Analytics
Insight/Analytics Tier
BI Tools
Applications
Automations
QuickSight
Power BI

Data Warehouse
BI
Reports
Data Warehouses
ETL
Structured Data
Data Lake
BI
Reports
Data Science
Machine Learning
Data Warehouses
ETL
Data Lake
Structured, Semi-structured and Unstructured Data
Data Lakehouse
BI
Reports
Data Science
Machine Learning
Metadata and Governance Layer
ETL
Data Lake
Structured, Semi-structured and Unstructured Data

Relational

Key Value

Text

Time Series

Columnar

Document

File Storage

Geo Spatial

Wide Column

Graph

Datalake

Immutable Logger

Skan.
CONFIDENTIAL AND PROPRIETARY

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total
InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity
Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice
Album
AlbumId
Title
ArtistId
Artist
ArtistId
Name
Genre
GenreId
Name
MediaType
MediaTypeId
Name

Dim_track
PK
Track_Id
Track_name
Track_Album_id
Track_Album_title
Track_Album_artist_id
Track_Album_artist_name
Track_media_type_id
Track_media_type_name
track_genre_id
track_genre_name
track_composer
track_milliseconds
track_bytes
track_unit_price
Fact_track_transaction
PK
Track_id
PK
Invoice_Id
PK
Customer_Id
Invoice_Line_Id
Sold_UnitPrice
Sold_Quantity
Dim_Customer
PK
Cumtomer_id
customer_first_name
customer_last_name
customer_company
customer_address
customer_city
customer_state
customer_country
customer_postal_code
customer_phone
customer_fax
customer_email
customer_support_rep_id
customer_support_rep_full_name
customer_support_rep_title
Dim_Invoice
PK
invoice_id
invoice_date
invoice_billing_address
invoice_billing_city
invoice_billing_state
invoice_billing_country
invoice_billing_post_code
invoice_total_amount

Data Cube
Date
1Qtr
2Qtr
3Qtr
4Qtr
sum
Product
TV
PC
VCR
sum
India
Nepal
China
sum
Country
All,All,All

dicing
drill-down
roll-up
slicing
pivot

One-to-One	User works for a single company (embedding)	*User*
		{
		"_id": "ObjectId('AAA')",
		"name": "Joe Karlsson",
		"company": "MongoDB"
		}

One-to-Few	User has a few addresses (embedding)	*User*
		{
		"_id": "ObjectId('AAA')",
		"name": "Joe Karlsson",
		"company": "MongoDB",
		"addresses": [
		{ "street": "45 Sesame St", "city": "Los Angeles" },
		{ "street": "123 Avenue Q", "city": "New York" }
		]
		}

One-to-Many	Product has many parts (child referencing)	*Product*
		{
		"name": "left-handed smoke shifter",
		"manufacturer": "Acme Corp",
		"catalog_number": "1234",
		"parts": [
		"ObjectID('AAAA')",
		"ObjectID('BBBB')",
		"ObjectID('CCCC')"
		]
		}
		
		Part
		{
		"_id" : "ObjectID('AAAA')",
		"partno" : "123-aff-456",
		"name" : "#4 grommet",
		"qty": "94",
		"cost": "0.94",
		"price": "3.99"
		}

One-to-Huge	Host server has millions of log messages (parent referencing)	*Host*
		{
		"_id": ObjectID("AAAB"),
		"name": "goofy.example.com",
		"ipaddr": "127.66.66.66"
		}
		
		Log
		{
		"time": ISODate("2014-03-28T09:42:41.382Z"),
		"message": "cpu is on fire!",
		"host": ObjectID("AAAB")
		}

Many-to-Many	User has many tasks and a task can be assigned to many users (mutual referencing)	*User*
		{
		"_id": ObjectID("AAF1"),
		"name": "Kate Monster",
		"tasks": [
		ObjectID("ADF9"),
		ObjectID("AE02"),
		ObjectID("AE73")
		]
		}
		
		Task
		{
		"_id": ObjectID("ADF9"),
		"description": "Write blog post about schema design",
		"due_date": ISODate("2014-04-01"),
		"owners": [ObjectID("AAF1"), ObjectID("BB3G")]
		}

Product Catalog

```
{
  type: "Audio Album",
  title: "A Love Supreme",
  description: "by John Coltrane",

  shipping: {
    weight: 6,
    dimensions: {
      width: 10,
      height: 10,
      depth: 1
    },
  },

  pricing: {
    list: 1200,
    retail: 1100,
    savings: 100,
    pct_savings: 8
  },

  details: {
    title: "A Love Supreme [Original Recording Reissued]",
    artist: "John Coltrane",
    genre: [ "Jazz", "General" ],
    tracks: [
      "A Love Supreme Part I: Acknowledgement",
      "A Love Supreme Part II - Resolution",
      "A Love Supreme, Part III: Pursuance",
      "A Love Supreme, Part IV-Psalm"
    ],
  },
}
```

Product (Book)

key:
products::books::chasm_city

document:

```
{
  "type": "book",
  "title": "Chasm City",
  "author": "Alastair Reynolds",
  "skus": [
    {
      "sku": 123456,
      "isbn-13": "9780441010646",
      "format": "paperback",
      "pages": 704
      "published": 2003
      "price": 9.99},
    {
      "sku": 234567,
      "isbn-13": "9780575068773",
      "format": "hardcover",
      "pages": 528
      "published": 2001
      "price": 19.99}]
  "reviews": {
    "average": 4.4,
    "count": 26},
  "categories": [
    "Science Fiction and Fantasy",
    "Science Fiction",
    "Space Opera"]
}
```


Product (Movie)

key:
products::movies::2001_a_space_odyssey

document:

```
{
  "type": "movie",
  "title": "2001: A Space Odyssey",
  "director": "Stanley Kubrick",
  "released": 1968
  "skus": [
    {
      "sku": 789012,
      "format": "blu-ray",
      "duration": 149,
      "price": 9.99,
      "reviews": {
        "average": 4.7,
        "count": 150}},
    {
      "sku": 890123,
      "format": "laserdisc",
      "duration": 149,
      "price": 124.99,
      "reviews": {
        "average": 4.5,
        "count": 130}}]
  "categories": [
    "Science Fiction"]
}
```

Project
title
startDate
endDate
departmentID (FK)
Department
ID (PK)
name
parentDepartmentID (FK)
leadPersonID (FK)
Project_Members
projectID (FK)
personID (FK)
role
Department_Members
departmentID (FK)
personID (FK)
Person
ID (PK)
entityID (FK)
dayOfBirth
Entity
ID (PK)
name
Organization
ID (PK)
entityID (FK)
departmentID (FK)
taxId

:Project
title
startDate
endDate
:BELONGS_TO
:Dept
name
:IS_PART_OF
:LEAD_BY
:WORKS_AT
:Person, :Entity
name
dayOfBirth
:Org, :Dept, :Entity
name
taxId

Recommendation model

```
[
    {
        "name": "device.status",
        "tags: {
            "deviceId" : "aliyun-tablestore",
            "version" : "2.0",
            "model" : "unify"
        },
        "location": {
            "lat" : 30.1276799186,
            "lon": 120.0848161441
        },
        "timestamp" : 1531918377012,
        "values" : {
            "status" : "active",
            "temperature" : 30,
            "cpu" : 50
        }
    }
]
```

Who

Where

When

What

Who At what time What Where Related to what?

WHO

WHEN

WHERE

WHAT

metric	timestamp	cluster	hostname	metric value
cpu	2015-04-28T17:50:00Z	Cluster-A	host-a	10
cpu	2015-04-28T17:50:10Z	Cluster-A	host-b	20
cpu	2015-04-28T17:50:20Z	Cluster-A	host-a	5
iops	2015-04-28T17:50:00Z	Cluster-A	host-a	10
iops	2015-04-28T17:50:10Z	Cluster-A	host-b	30
iops	2015-04-28T17:50:20Z	Cluster-A	host-a	8

Metrics **Timestamp** **Subject dimension** **Measurements**

Skan.
© SKAN.AI 2019 CONFIDE

Shard id, row id, column family id, column

Key Factors that Drive the Decision

	Mongo	Cassandra	HBase
	Document	Wide column	Wide column
Rich Object Model	++		
Secondary Indexes	++		
Replication	Master slave	Multi master	Master slave
High Availability		++	
Write Scalability		++	
Rich Query Language		++	
Schema	++		
CAP	CP	AP	CP
Performance			
Aggregation	++		

https://benchant.com/blog/mongodb-vs-cassandra

Consistency
All clients see the same data at the same time
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
CA
CP
AP
MariaDB
Microsoft SQL Server
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak

Availability
Rdbms
Cassandra
Consistency
P (Distributed)
Mongo

	MongoDB	Cassandra
Expressive object model	Yes	No
Secondary indexes	Yes	No
No downtime on node failure	No	Yes
High write throughput	No	Yes

https://benchant.com/blog/mongodb-vs-cassandra

If you need query language support, Cassandra is the better fit for you.

Use Cases

Content Management System

Fraud Detection

- We extract entities from a large database of business activities, make relationships between them.

- We can then use a technique called “Entity Link Analysis” to identify suspicious links between types of entities that may indicate fraudulent behaviour.

- We will use entities from people such as names and dates of birth, as well as contextual entities such as IP addresses, device identifiers and access times.

- We can then analyse the links between these entities and mark up those that have previously been marked as fraudulent. When entities that are marked as fraudulent start to intersect those that are not marked as fraudulent, suspicions begin to arise. For example, use of multiple bank accounts on a single device that has been previously used to access fraudulent accounts.

Product Catalogue

- 3 different systems to maintain: RDBMS, Search engine, Caching layer
- RDBMS schema is complex and static
- Applications needs to talk many languages

- With the API layer, the application just issues a single call, thus reducing complexity and latency
- No need for a caching layer since MongoDB's read performance is close to Memcache and the likes
- The schema is dynamic and maps well to the API calls

An inventory system that might want full ACID. I was very unhappy when I bought a product and they said later they were out of stock. I did not want a compensated transaction. I wanted my item!

Asset Management

Network Mapping

- when mapping relationships between connected physical/virtual hardware and the services that they support. An enterprise will use CMDBs (configuration management databases) and/or service catalogs to keep inventory of their systems. They are used to keep track of components, their purpose, software versions and the interdependencies between them.
- relationships between infrastructure components not only enables interactive visualisations of the network estate but also network tracing algorithms to walk the graph. For example, algorithms for:

Dependency Management: Identify single points of failure and simulate the impact of their failure on services, to identify cascading failures before they happen.

Bottleneck Identification: Find weak links in network routing that could cause bottlenecks at times of high network utilisation.

Latency Evaluation: Estimate latency across paths in the network, and the impact on services accessed from various geographic regions.

- Real-time recommendations and advertising **can quickly access and present** new recommendations or ads as a web visitor moves throughout a site.

- build social relationship to enhance content personalization

- Store user session details and preference. All the information lend themselves to fast reads and writes.

transactions logging in Banking and Finance, events logging in web analytics or messaging systems, time-series data, tracking of inventory, IoT (Internet of Things) data, weather tracking,

Terms
Document
brown
dog
fat
fox
jump
lazi
over
quick
{
"_id": 1
"phrase": “the quick brown fox jumps over the lazy dog”
{
"_id": 2
"phrase": "The fat brown dog"
}

beats
logstash
Syslog
logstash
elasticsearch
kibana
ClientApp
logstash

Logs
Scales out
APACHE
HTTP SERVER
Logs
Ingested
Logs
Search
Aggregate
Logstash
Elasticsearch
Logs
Scales out

ELK
EFK
Splunk

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Row Store

Row 1	India
	Chocolate
	1000
Row 2	India
	Ice-cream
	2000
Row 3	Germany
	Chocolate
	4000
Row 4	US
	Noodle
	500

Column Store

Country	India
	India
	Germany
	US
Product	Chocolate
	Ice-cream
	Chocolate
	Noodle
Sales	1000
	2000
	4000
	500

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

{Ordered, cust name, amount} , {itemcode, qty,price}

(Row partition id (shard key), column family id, row id)

multidimensional map (map of maps)

```
{
  "rowkey1": {"cf11": {"column111": {"version1111": value1111,
                                     "version1112": value1112},
                       "column112": {"version1121": value1121,
                                     "version1122": value1122,
                                     "version1123": value1123,
                                     "version1124": value1124},
                       "column113": {"version1131": value1131}
                      },
             "cf12": {"column121": {"version1211": value1211},
                       "column122": {"version1221": value1221},
                                     "version1222": value1222}
                      }
             },
  "rowkey2": {"cf11": {"column111": {"version2111": value2111,
                                     "version2112": value2112},
                       "column112": {"version2121": value2121,
                                     "version2122": value2122,
                                     "version2123": value 2123,
                                     "version2124": value2124}
                      },
             "cf12": {"column121": {"version2211": value2211},
                       "column122": {"version2221": value2221}
                      }
             }
}
```

Row partition id,
Col partition id

Column Family 1
Column Family 2
cf1:col-A
cf1:col-B
cf2:col-Foo
cf2:col-XYZ
cf2:foobar
Region 1
row-1
row-10
row-18
A18 - v1
B18 - v3
Foo18 - v1
XYZ18 - v2
foobar18 - v1
Region 2
row-2
row-5
row-6
row-7
Physical Coordinates for a Cell: Region Directory → Column Family Directory
→ Row Key → Column Family Name → Column Qualifier → Version

	CF1:colA	CF1:colB	CF1:colC
Row1	@time7: value3		
Row10	@time2: value1	@time2: value1	
Row11	@time6: value2		
Row2	@time4: value1		@time4: value1

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

column-family	Columnar
sparse data model	
multi-dimensional map	
column-families are not independently accessible.	every column is stored separately
NoSQL	SQL interface
Reads that use the partition key are incredibly fast	optimized for read-mostly analytical workloads
high throughput writes	very slow writes
	data warehouses

	MongoDB	Cassandra
Expressive object model	Yes	No
Secondary indexes	Yes	No
No downtime on node failure	No	Yes
High write throughput	No	Yes

- Cassandra is optimized for really high throughput on writes. **If your use case is read-heavy (like cache) then Cassandra might not be an ideal choice.**
- It does not support complete transaction management across the tables. Not ACID compliant system.
- Secondary Index not supported. Have to rely on Elastic search /Solr for Secondary index and the custom sync component has to be written.

not possible to aggregate data in a query. Sums and averages like information have to be done by the client-side application.

Use Cassandra as the primary data store for capturing information as it arrives into the system; but then build “query-optimised views” of subsets of that data in other databases specialised to the kinds of access users require. Use an indexing engine such as SOLR to provide full-text search across records, or a graph database such as Neo4j to store metadata in a way that supports “traversal-heavy” queries that join together many different kinds of entity. Use an RDBMS when you need the full power and flexibility of SQL to express ad hoc queries that explore the data in multiple dimensions.

docker
Logfiles
JMX Metrics
Notary
docker
Logfiles
JMX Metrics
Party A
docker
Logfiles
JMX Metrics
Party B
docker
beats
FileBeat Logfile Collector
docker
Prometheus
JMX Collector
docker
Stack
Elasticsearch - Logstash
Kibana Monitoring
http://localhost:5601
docker
Grafana
Grafana Monitoring
http://localhost:3000

Classic Relational Databases

Name	Age	Nickname	Employee
Giacomo Guilizzoni Founder & CEO	40	Heidi	◉
Marco Botton Tuttofare	38		☑
Mariah Maclachlan Better Half	41	Potato	⊟
Valerie Liberty Head Chef	16	Val	☑

Data are multidimensional

Time Series Databases

Sensor Temperature	Time
39.6	12/04/19 @ 14:12
41.2	12/04/19 @ 14:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

Case : retention policy = 1 hour

DBMS & TSDB Difference

Tools that 'produce' TSDB data

Tools that 'consume' TSDB

*Non-exhaustive list

Car ID	Departure			Arrival		
	Date-Time	Latitude	Longitude	Date-Time	Latitude	Longitude
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...

Topic Data
Msg 1
Msg 2
Msg t
Nonce
HMAC
HMAC
HMAC
Checkpoint
Stamp: 1
Proof:
Checkpoint
Stamp: 2
Proof:
Checkpoint
Stamp: t
Proof:

Rdbms (Referential Integrity)

tblPerson			
ID	Name	Email	GenderID
1	Jade	j@j.com	2
2	Mary	m@m.com	3
3	Martin	ma@ma.com	1
4	Rob	r@r.com	NULL
5	May	may@may.com	2
6	Kristy	k@k.com	NULL

tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

key	value
123	123 Main St.
126	(805) 477-3900

Wide Column

- Spotify uses Cassandra to store user profile attributes and metadata about artists, songs, etc. for better personalization
- Facebook initially built its revamped Messages on top of HBase, but is now also used for other Facebook services like the Nearby Friends feature and search indexing
- Outbrain uses Cassandra to serve over 190 billion personalized content recommendations each month

UPDATE
DELETE
INSERT
Extract
Transform
Load
1. OLTP
2. ETL
3. OLAP

A fact table has two types of columns: those that include fact (purely numerical facts) and those that are foreign keys to the dimension table.

Dimension tables store the descriptive attributes of entities across the business enterprise and function as common lookup tables

n.
© SKAN.AI 2019 CONFIDE

DimSalesTerritory
SalesTerritoryKey
SalesTerritoryAltern...
SalesTerritoryRegion
SalesTerritoryCountry
SalesTerritoryGroup
DimDate
DateKey
FullDateAlternateKey
DayNumberOfWeek
EnglishDayNameOf...
SpanishDayNameO...
FrenchDayNameOf...
DayNumberOfMonth
DayNumberOfYear
WeekNumberOfYear
EnglishMonthName
DimPromotion
PromotionKey
PromotionAlternate...
EnglishPromotionN...
SpanishPromotionN...
FrenchPromotionNa...
DiscountPct
EnglishPromotionTy...
SpanishPromotionT...
FrenchPromotionType
EnglishPromotionCa...
FactInternetSales
ProductKey
OrderDateKey
DueDateKey
ShipDateKey
CustomerKey
PromotionKey
CurrencyKey
SalesTerritoryKey
SalesOrderNumber
SalesOrderLineNumber
RevisionNumber
OrderQuantity
UnitPrice
ExtendedAmount
UnitPriceDiscountPct
DiscountAmount
ProductStandardCost
TotalProductCost
DimCurrency
CurrencyKey
CurrencyAlternateKey
CurrencyName
DimProduct
ProductKey
ProductAlternateKey
ProductSubcategory...
WeightUnitMeasure...
SizeUnitMeasureCo...
EnglishProductName
SpanishProductNa...
FrenchProductName
StandardCost
FinishedGoodsFlag
DimCustomer
CustomerKey
GeographyKey
CustomerAlternate...
Title
FirstName
MiddleName
LastName
NameStyle
BirthDate
MaritalStatus
FactInternetSalesReaso
SalesOrderNumber
SalesOrderLineNu...
SalesReasonKey

Relational

Product
Product_ID INT
Product_Name VARCHAR(45)
Description TINYTEXT
Category INT
Weight_Class INT
Warranty_Period INT
Supplier_ID INT
Status VARCHAR(15)
List_Price DECIMAL
Minimum_Price DECIMAL
Price_Currency VARCHAR(5)
Indexes
Inventory
Inventory_ID INT
Product_ID INT
Warehouse_ID INT
Quantity_on_Hand INT
Quantity_Available INT
Indexes
Warehouse
Warehouse_ID INT
Location_ID INT
Warehouse_Name VARCHAR(45)
Indexes
Order_Item
Order_Item_ID INT
Order_ID INT
Product_ID INT
Unit_Price DECIMAL
Quantity DOUBLE
Indexes
Location
Location_ID INT
Country_ID INT
Address_Line_1 VARCHAR(45)
Address_Line_2 VARCHAR(45)
City VARCHAR(45)
State VARCHAR(24)
District VARCHAR(24)
Postal_code VARCHAR(20)
Location_type_code INT
Description VARCHAR(128)
Shipping_notes VARCHAR(256)
Countries_Country_ID INT
Indexes
Customer_Company
Company_ID INT
Company_Name VARCHAR(45)
Company_Credit_Limit INT
Credit_Limit_Currency VARCHAR(5)
Indexes
Orders
Order_ID INT
Customer_ID INT
Sales_Rep_ID INT
Order_Date DATE
Order_Code INT
Order_Status VARCHAR(15)
Order_Total INT
Order_Currency VARCHAR(5)
Promotion_Code VARCHAR(45)
Indexes
Phone_Number
Phone_number_ID INT
Persons_Person_ID INT
Locations_Location_ID I...
Phone_number INT
Country_code INT
Phone_Type_ID INT
Indexes
Person_Location
Persons_Person_ID INT
Locations_Location_ID INT
Sub_Address VARCHAR(45)
Location_Usage VARCHAR(4...
Notes TINYTEXT
Indexes
Customer_Employee
Customer_Employee_ID INT
Company_ID INT
Badge_Number VARCHAR(20)
Job_Title VARCHAR(45)
Department VARCHAR(45)
Credit_Limit INT
Credit_Limit_Currency VARCHAR(5)
Indexes
Person
Person_ID INT
First_name VARCHAR(20)
Last_name VARCHAR(20)
Middle_names VARCHAR(45)
Nickname VARCHAR(20)
Nat_lang_code INT
Culture_code INT
Gender VARCHAR(12)
Indexes
Country
Country_ID INT
Country_Name VARCHAR(24)
Country_Code VARCHAR(3)
Nat_Lang_Code INT
Currency_Code VARCHAR(10)
Indexes
Customer
Customer_ID INT
Person_ID INT
Customer_Employee_ID INT
Account_Mgr_ID INT
Income_level INT
Indexes
Employment
Employee_ID INT
Person_ID INT
HR_Job_ID INT
Manager_Employee_I...
Start_Date DATE
End_Date DATE
Salary INT
Commission_Percent D...
Employment_Status VARCH...
Indexes
Restricted_Info
Person_ID INT
Date_of_Birth DATE
Date_of_Death DATE
Government_ID VARCHAR(24)
Passport_ID VARCHAR(24)
Hire_Date DATE
Seniority_Code INT
Indexes
Employment_Jobs
HR_Job_ID INT
Countries_Country_ID INT
Job_Title VARCHAR(45)
Min_Salary INT
Max_Salary INT
Indexes

Skan.

© SKAN.AI 2019 CONFIDE

Star schema

A star schema has a single fact table in the center, containing business "facts" (like transaction amounts and quantities). The fact table connects to multiple other dimension tables along "dimensions" like time, or product. Star schemas enable users to slice and dice the data

Skan.

© SKAN.AI 2019 CONFIDE

there are two basic models which are used in dimensional modeling

- Star Model
- Snowflake Model

Skan.
© SKAN.AI 2019 CONFIDE

Sales (StoreID, ItemID, CustID, qty, price)
StoreID (storeid, city, state)
ItemID (itemid, category, brand, color, size)
CustID (custid, name, address)

Skan.
© SKAN.AI 2019 CONFIDE

Slicing & Dicing

OLAP cube operations

OLAP
CUBE
Quaters
Q1
Q2
Q3
Q4
$21,935649
9975
Territory
U.S.
Australia
PC
Book
Shoe
Clothes
Product Category

Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile Modem Phone Security
item(types)
Drill down on time(from quarters to month)
Locations (countries)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (months)
January
February
March
April
May
June
July
August
September
October
November
December
150
100
150
Mobile Modem Phone Security
item(types)

Locations (countries)
USA
Canada
2000
1000
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)
roll-up on location (from cities to countries)
Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
605
825
14
400
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)

Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile Modem Phone Security
item(types)
slice
for time
="Q1"
Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile Modem Phone Security
item(types)

Locations (cities)
Toronto
395
Vancouver
Q1
605
Q2
Time (Quarter)
Mobile
Modem
item (types)
Dice for (location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "Mobile" or "Modem")
Locations (cities)
Chicago
440
New York
1560
Toronto
395
Vancouver
Q1
605
825
14
400
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security

Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
Pivot
Item (types)
Mobile
Modem
Phone
Security
605
825
14
400
Chicago
New York
Toronto
Vancouver
Location (Cities)

Operational

Structured

+ ACID - transaction,
- LIMITED SCALABLE - not distributed
- Fixed Schema
CAP - CA

Xf

- AWS RDS
- Azure SQL
- Google Cloud Sql
- Google Cloud Spanner
- **Postgres**
- **Mysql**

1

Semi Structured

- BASE,
+ SCALE- shareded (distributed),
+ Flexible Schema
CAP - CP | AP

Dictionary (key value)

- AWS Dynamo Db
- Azure Cosmos -> Table
- Azure Table
- Google Big Table

2

Nested Objects

Document / Tabular

- AWS Document DB
- Azure Cosmos -> Document Db
- Firestore
- **Mongodb**, **Couchbase**,Solr

3

Entity Relationship

Graph

- AWS Neptune
- Azure Cosmos -> Graph
- Google Janusgraph
- **Neo4j**
- Graph,orientdb

4

Wide Column db (column-family)

frequently added but rarely updated.

- AWS Keyspaces (unlimited)
- Cosmos
- Google Big Table
- **Hbase, cassandra**
- Scylla

5

Rows vs. Documents

	Table
Row 1	Data
Row 2	Data
Row 3	Data
...	:

Document 1
```
{
 data
}
```

Document 2
```
{
 data
}
```

Document 3
```
{
 data
}
```

...

Columns vs. Properties

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1
```
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

Document 2
```
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

Document 3
```
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

Schema vs. Schema-Free

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1
```
{
 "id": "1",
 "name": "John Smith",
 "isActive": true,
 "dob": "1964-30-08"
}
```

Document 2
```
{
 "id": "2",
 "fullName": "Sarah Jones",
 "isActive": false,
 "dob": "2002-02-18"
}
```

Document 3
```
{
 "id": "3",
 "fullName":
 {
  "first": "Adam",
  "last": "Stark"
 },
 "isActive": true,
 "dob": "2015-04-19"
}
```

Normalized vs. Denormalized

User Table

UserID	Name	Dob
1	John Smith	8/30/1964

Holdings Table

StockID	UserID	Qty	Symbol
1	1	100	MSFT
2	1	75	WMT

Document

```
{
  "id": "1",
  "name": "John Smith",
  "dob": "1964-30-08",
  "holdings": [
    { "qty": 100, "symbol": "MSFT" },
    { "qty": 75, "symbol": "WMT" }
  ]
}
```

Counter
C
BP

Counter
C
BP
Counter
E
BP
Counter
?
BP

General
C
E
?
BP
BP
BP
BP
BP
BP
BP

General
C
E
?
BP
D BP
I BP
BP
D BP
I BP

General
C
E
?
Split
General : BP
D BP
D BP
D BP
Clone
DBP : DBP
D BP
D BP
D BP
Shard
IBP : DBP
I BP
I BP
I BP

Scalability Cube - 50 rules for high Scalability

Scalability Cube - 50 rules for high Scalability

Scalability Cube - 50 rules for high Scalability

Scale-Up vs. Scale-Out

Strong Consistency vs. Eventual Consistency

CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak

North America Data Center
Singapore Data Center
Node A
[NewData]
Replicate
Node B
[NewData]
Node C
[OldData]
Write
[NewData]
Block Reading [NewData]
Block Reading [NewData]
All Clients trying to read are blocked till replication completes

North America Data Center
Singapore Data Center
Node A
[NewData]
Replicate
Node B
[NewData]
Node C
[OldData]
Write
[NewData]
Read
[NewData]
Read
[OldData]

COMMON COMPARISONS BETWEEN

MYSQL & NOSQL

	MySQL	NoSQL
Nature	Relational Database	Non-Relational Database
Design	Based on the concept of tables	Based on the concept of documents
Scalable	Tough to scale due to its relational nature	Easily scalable big data compared to relational
Model	Detailed database model is needed before creation	No need of a detailed database model
Community	Vast community available	Community is growing rapidly, but still smaller compared to MySQL
Standardization	SQL is standard language	Lacks standard query language
Schema	The Schema is rigid	The Schema is dynamic
Flexibility	Not very flexible in terms of design	Very flexible in terms of design
Insertions	Inserting new columns or fields affect the design	No effect on the design with the insertion of new columns or fields

Facebook, Wikipedia, Quora, Flickr	MySQL
Twitter	MySQL for tweets and users their own special kind of graph database, FlockDB, built on top of MySQL their own version of Memcached
LinkedIn	Oracle Database and Voldemort
YouTube	MySQL -> BigTable
Microsoft, Myspace	***SQL Server***
Yahoo	***PostgreSQL***

Key Value	○ Twitter uses Redis to deliver your Twitter timeline
	○ Pinterest uses Redis to store lists of users, followers, unfollowers, boards, and more
	○ Coinbase uses Redis to enforce rate limits and guarantee correctness of Bitcoin transactions
	○ Quora uses Memcached to cache results from slower, persistent databases
Graph	○ Walmart uses Neo4j to provide customers personalized, relevant product recommendations and promotions
	○ Medium uses Neo4j to build their social graph to enhance content personalization
	○ Cisco uses Neo4j to mine customer support cases to anticipate bugs
Document	○ SEGA uses MongoDB for handling 11 million in-game accounts
	○ Cisco moved its VSRM (video session and research manager) platform to Couchbase to achieve greater scalability
	○ Aer Lingus uses MongoDB with Studio 3T to handle ticketing and internal apps
	○ Built on MongoDB, The Weather Channel's iOS and Android apps deliver weather alerts to 40 million users in real-time

Analytical

Read
Columnar db
(column database)
Dimensional modeling
Cloud DWH (OLAP)

- AWS Redshit (max: 8 PB)
- **Azure Synapse**
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house

Read

Text Database

- AWS **Elastic Search**
- AWS cloud Search
- Azure Cognitive Search
- Google Search API
- Elastic Search, **Solr**

Read/write

Wide Column db
(column-family database)

- AWS Keyspaces (unlimited)
- Cosmos
- Google Big Table
- **Hbase, cassandra**
- Scylla
- ⋮

Time Series

- AWS Timestream
- InfluxDB, Scylla
- OpenTSDB,
- KairosDB
-

Audit Trial

Immutable Logger

- AWS Quantum Ledger
- Azure SQL Database Ledger
- Google ?
- Hyper ledger Fabric

geo Spatial

- AWS Keyspaces
- Azure Cosmos
- Big Table, Big Query
- **Solr**, PostGIS
- Mongoldb (GeoJSON)

Choose Communication protocol

Event storming

Step 1: Create domain events

Step 2: Add the commands that caused the domain event

Step 2b: Add the actor that executes the command

Step 3: Add corresponding aggregate

Connected protocol
(REST, WS, GRPC , Thrift)
Communication Services
Connectivity Services
Message protocol
(AMQP, MQTT, …)

	Connected	Msg database (ACID)
Scale	- -	+ +
Coupling	- -	+ +
Reliability	- -	+ + (retry)
Developer effort		
Consistency	Immediate	Eventual - - (SAGA)
Debugging	+	-
Delivery Order	Ordered ++	Unordered * - -
Duplicate	- -	- -
Dev Ops Effort		
Exception Handling	+ + (error response)	- - ?
Result	Response	? (one way)
Infra structure cost	- -	+ +

	REST	WSS	GRPC
Browser Support	Y	Y	N
Format	TEXT (HTTP 1.x)	TEXT (HTTP 1.x)	BINARY, HTTP2
Serialization	JSON	JSON / XML / Any	Proto Buf
Performance	- -	+	+ +
Duplex communication	Pull	Pull & Push	Pull & Push
Use case	North South	Streaming & North South	East West

Portal
North south
East West

Communication Services
Connectivity Services
Orchestration Services
Process Services
Development Services
Development Framework Services
User Interface Services
Administration Services
System Management Services
Data Management Services
Operational Business Intelligence Services
Collaboration Services
Security Services

Communication Services
Connectivity Services
Orchestration Services
Process Services
Development Services
Development Framework Services
User Interface Services
Administration Services
System Management Services
Data Management Services
Operational Business Intelligence Services
Collaboration Services
Security Services

Communication Services
Connectivity Services
Orchestration Services
Process Services
Development Services
Development Framework Services
User Interface Services
Administration Services
System Management Services
Data Management Services
Operational Business Intelligence Services
Collaboration Services
Security Services

Event vs Command

Post **Pre**

	<< API >>	<< Message >>	
Ordering	Ordered (+)	Unordered(-)	
Duplicate	Yes (-)	Yes(-)	Idempotency
Protocol	2 way (+)	One way (-)	
Resilence (recover)	No (-) retry logic	Yes (+)	
Connection	Always connected	Occousionaly connected	
Scalability	- -	+ + +	
Consistency	Immediate (++)	Eventual (- -)	
Load Leveling	- -	+ +	
Low Coupling (maintainability)	- -	+ +	
Distributed Comm Patterns	- -	+ +	SAGA, Materialized View
Internet	Yes	Yes *	
Browser support	Yes	No	
Dev effort	++	- -	
Operational effort	+ +	- -	

```
fun(){
   while(true){
      Msg m = GetMessage();
      Run domain Logic
      m.ack();
   }
}
```

Portal
Create Order
Order Api
Order Service
Order
Order Service
Account Service
Order
Account

Idempotency

Request
Time

Queue-based Load Levelling

source:msdn

p2p
SQS
MSMQ
...
Publisher
Publisher
Queue
Payroll Subscriber 1
Payroll Subscriber 2
pubSub
RabbitMQ
SNS
Redis
MQ Series
Publisher
Publisher
Topic
Tax Subscriber 1
Payroll Subscriber 2
Payroll Subscriber 3
T3 T2 T1
Publisher
Tax Subscriber
State
p2: time
Payroll Subscriber
State
P1:t
Kafka
Event Hub
Kinesis

Publisher
Publisher
D1
D2
D3
Queue
Logic A
Logic A
Logic A

Data Concurrency
Logic A
Logic A
Logic A
D1
D2
D3

Task Concurrency
Obstacle
Traffic
Line
Logic A
Logic B
Logic C
D1
D1
D1

Publisher
Publisher
D1
Topic
Logic A
Logic B

	Kafka	Pulsar	RabbitMQ (Mirrored)
Peak Throughput (MB/s)	605 MB/s	305 MB/s	38 MB/s
p99 Latency (ms)	5 ms (200 MB/s load)	25 ms (200 MB/s load)	1 ms* (reduced 30 MB/s load)
Number of Programming Languages Supported	17	6	22
Stack Overflow Questions	21,233	134	11,430
Pub/sub	Yes	Yes	Yes
Message routing	Medium	Medium	High
Queueing	Low	Medium	High
Event Streaming	High	Medium	No
Mission-critical	High	Low	High
Slack Community Size	23,057	2,332	7,492
Meetups	486	1	1
Message replay, time travel	+++	+++	-
Exactly-once processing	+++	+	-
consumption	Pull	Push	Push
Permanent storage	Yes	Partial	No

	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,…	Py, java, php, .net, js,…	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	

Table 1

	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,…	Py, java, php, .net, js,…	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	
Potential data loss	Yes	Yes	
Community and vendor support	Good	Good	Good
Building an event store system (used as a store)	No	Yes	No
Ordering	not guaranteed	Guaranteed	
Replay events	No	Yes	No
Transactions	No	Yes	No

Streaming

Data stream as an unbounded table

Programming Model for Structured Streaming

nc
cat dog
dog dog
owl cat
dog
owl
Time
1
2
3
Input
Unbounded
table of all input
cat dog
dog dog
data up
to t=1
cat dog
dog dog
owl cat
data up
to t=2
cat dog
dog dog
owl cat
dog
owl
data up
to t=3
word count query
Result
Table of
word counts
cat 1
dog 3
result up
to t=1
cat 2
dog 3
owl 1
result up
to t=2
cat 2
dog 4
owl 2
result up
to t=3
Output
Complete Mode
print all the counts to console

Identification of Domain Events

Time Sequencing of Domain Events

parallel have been stacked up vertically

Identification of Triggers/Commands

- Yellow represents the actors
- Blue represents the commands issued by the actors
- Orange are the events we started with

Identification of Aggregates

1
Portal
Portal
2
Portal
WS
LB
Bid Service
2
Bid Service
Bid Service
1
pub/sub
Redis Cache
(key-value)

Device registration
Device communication
Protocol translation

Choose Operational Compute

Inventory
Tom cat
Accounting
Tom cat
Inventory
Tom cat
Accounting
Tom cat
Inventory
Tom cat
Order
Tom cat

Order
Accounting
Inventory
Tom cat
Order
Accounting
Inventory
Tom cat
Order
Accounting
Inventory
Tom cat

On-Premises
Infrastructure
(as a Service)
Platform
(as a Service)
Software
(as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
Other Manages

Isolated Env.

My app 1
My app 2
Middlewear (tomcat/ Apache/ IIS / Nodejs…)
Os
Cpu
RAM
Disk
I/o
Machine

VM1
VM2
My app 1
My app 3
Middlewear(tomcat/apache/…)
Unix Guest Os
My app 2
Win Guest Os
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Virtualisation layer
Host Os
Cpu
RAM
Disk
I/o
Machine

Container 1
Container 2
Cpu
RAM
Disk
I/o
My app 1
Middlewear
My app 2
Cpu
RAM
Disk
I/o
Containerization layer (Docker)
Cpu
RAM
Disk
I/o
Host Os
Machine

VM1
VM2
Container 1
8080
Container 2
My app 1
My app 2
Middlewear
Middlewear
Cpu
RAM
Disk
I/o
8080
8080
Cpu
RAM
Disk
I/o
Containerization layer (Docker)
Unix Guest Os
Win Guest Os
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Virtualisation layer (Hyperviser)
Host Os (azure)
Cpu
RAM
Disk
I/o
Machine

Managing Container Life cycle

VM1
VM2
Container 1
Container 2
Container 3
My app 1
My app 2
My app 3
Middlewear
Middlewear
Middlewear
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Containerization layer (Docker)
..
Unix Guest Os
…
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Virtualisation layer
Host Os (azure)
Machine

Kubernetes (slave)
Kubernetes (Slave)
Kubernetes (Master)

VM1
Container 1
Container 2
My app 1
Middlewear
Cpu
RAM
Disk
I/o
Kubernetes (slave)
Containerization layer (Docker)
Unix Guest Os
VM2
Container 1
Container 2
My app 1
Middlewear
Cpu
RAM
Disk
I/o
Kubernetes (Slave)
Containerization layer (Docker)
Unix Guest Os
VM3
Kubernetes (Master)
Cmds

Cluster
VM *
Pod *
Container *
Process *
Code
Execute *
Docker Image
Code
cmds
Docker /k8s

API management platform : API gateway plus a whole lot more features to create an entire API marketplace

VM1
8080
Vnet
Container 1
8080
Container 2
todoDB
Todo
app(py)
Flask
Cpu
RAM
Disk
I/o
8080
8080
Disk
Cpu
RAM
Disk
I/o
Containerization layer (Docker)
Disk
Unix Guest Os
VM2
Win Guest Os
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Virtualisation layer
Host Os (azure)
Cpu
RAM
Disk
I/o
Machine
27017:27017
/root/myapi/mydata:/data/db

INFRASTRUCTURE SETUP

.NET framework
MVC 4
IIS
windows
[$0.192/hr for Windows OS]
http://XXXX.redbus.pe/, Throughput was 60.5/Sec.

2–6 EC2s of C4 family
depending on the traffic.

.NET CORE
[$0.1/hr for Linux] 45% of cost

http://XXXX.redbus.pe/ ,
Throughput was 142.9/Sec.

TCP/IP Model		OSI Model
Application Layer	HTTP, HTTPS, SMTP, IMAP, FTP, DNS, NNTP	Application
		Presentation
		Session
Transport	UDP, TCP, SCTP	Transport
Internet		Network
Network Access (Link)		Data Link
		Physical

'Layer 7' Load Balancing

'Layer 4' Load Balancing

Are clients internal or external?
Internal
External
Internal Load Balancing
Is the traffic HTTP(S)?
Yes
No
HTTP(S) Load Balancer
Is the traffic SSL?
Yes
No
SSL Proxy Load Balancer
Is the traffic TCP?
Yes
No
TCP Proxy Load Balancer
Network Load Balancer
Walk down the OSI Stack

Choose Analytical Compute

Data Lake
Data Warehouse
Data Mart

Operational
raw form
cleaned
EL
ETL
ETL
T
T
T

Reference Architecture
Analytical Application

Generation 1: The beginning of Big Data at Uber (Before 2014)

Data was scattered across different OLTP databases.

Generation 1 (2014-2015) - The beginning of Big Data at Uber

Aggregating all of Uber's data in one place. Developed multiple ad hoc ETL jobs that copied data from different sources (i.e. AWS S3, OLTP databases, service logs, etc.) into Vertica.

Data was ingested with no transformation during ingestion.

Generation 2 (2015-2016) - The arrival of Hadoop

only over 100 gb of new data was added every day. Since HDFS and Parquet do not support data updates, each run of the ingestion job had to convert the entire, over 100 tb dataset for that specific table.

Generation 3 (2017-present) - Let's rebuild for long term

Incremental ingestion:

Redbus

700 GB/ week
Full synchronization
Sqoop,
Apache Drill,
AWS Glue (Spark)
NiFi
Database
API
HTTP
Kafka Stream
nifi
data routing
Data Lake
on Amazon S3
AWS Glue
Spark Jobs
batch processing
Data Marts
on Amazon S3
AWS Lambda
Amazon Cloudwatch
triggers
Amazon Athena
SQL
Data
Visualisation
druid
OLAP
Superset
data visualizations
Incremental synchronization
NiFi
Kafka

Event Sequence

Hide >

Elapsed time

0sec 1sec 1sec 2sec 2sec 3sec

0 2 4

Events

Clear Selection (11)

ALLCONFIG

0min 17min 33min 50min 67min 83min

4e29b3a5-222e...

abecd1f0-4cbf...

543bc36e-c9e2...

309e6507-8c90...

c9fb8876-0701...

Align sequences by

1st - + **event**

Event type summary

78 events (427 hidden [84.6%])

- allconfig (14.1%)
- busdetails
- busroutes
- filters
- offers (35.9%)
- offerscovid (23.1%)
- persuasion
- perz (26.9%)
- rides
- seatlayout
- signin

Calendar heat-map

Modern Data Platform Reference Architecture

AWS IoT
Lambda Function
Kinesis Streams / Firehose
Glacier
S3
Glue ETL
aws
EMR
Kinesis Analytics
SageMaker
RedShift
RDS
SageMaker Studio (EDA)
Elastic Search
DynamoDB
Glue Catalog
Kinesis Streams
Lambda Function
QuickSight

	Kinesis Data Streams	Kinesis Firehose
Purpose	Low latency streaming service for ingest at scale	Data transfer service to load streaming data into Amazon S3, Redshift, Elasticsearch & Splunk
Provisioning	Managed service but needs configuration for shards.	Fully managed service, no administration.
Processing	Real Time (~200ms latency for classic, ~70ms for enhanced fan out)	Near real time (depends on buffer size OR buffer time min. 60 secs)
Scaling	Must manage scaling (configure shards)	Automated Scaling – as per the demand
Data Storage	Configurable from 1 to 7 days	Does not provide data storage
Replay capability	Supports replay capability	Does not support replay capability
Producers	Need to write code for producer. Supports SDK, Kinesis Agent, KPL, CloudWatch, IoT	Need to write code for producer. Supports KPL, Kinesis Agent, Data Streams, CloudWatch, IoT
Consumers	Open ended. Supports multiple consumers and destinations. Supports KCL and Spark.	Closed ended. Handled by Firehose. Does not support KCL or Spark.

jayendrapatil.com

Data
1-5 msec
5 mim/ 5 hours
Amazon S3
Amazon S3 Glacier
$0.025 per GB per month
$0.004 per GB per month + extraction cost

	S3	**S3 Glacier**
Access speed	Hi	Low
API availability	Yes	Yes
Data storage cost	Relatively high	Very low
Object size	Up to 5 Tb	40 Tb
Accommodation in regions	Many regions	Average
Static Web Content	Yes	No
Supporting Versioning	Yes	No

	Data Streams	Data Firehose	Data Analytics	Video Streams
Short definition	Scalable and durable real-time data streaming service.	Capture, transform, and deliver streaming data into data lakes, data stores, and analytics services.	Transform and analyze streaming data in real time with Apache Flink.	Stream video from connected devices to AWS for analytics, machine learning, playback, and other processing.
Data sources	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Amazon MSK, Amazon Kinesis Data Streams, servers, mobile devices, IoT devices, etc.	Any streaming device that supports Kinesis Video Streams SDK.
Data consumers	Kinesis Data Analytics, Amazon EMR, Amazon EC2, AWS Lambda	Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, generic HTTP endpoints, Datadog, New Relic, MongoDB, and Splunk	Analysis results can be sent to another Kinesis stream, a Kinesis Data Firehose delivery stream, or a Lambda function	Amazon Rekognition, Amazon SageMaker, MxNet, TensorFlow, HLS-based media playback, custom media processing application
Use cases	– Log and event data collection – Real-time analytics – Mobile data capture – Gaming data feed	– IoT Analytics – Clickstream Analytics – Log Analytics – Security monitoring	– Streaming ETL – Real-time analytics – Stateful event processing	– Smart technologies – Video-related AI/ML – Video processing

S. No.	Amazon DynamoDB	Amazon Redshift
1	It was developed by Amazon in 2012.	It was developed by Amazon in 2012.
2	It is hosted, scalable database service by Amazon with data stored in Amazon cloud.	It is large scale data warehouse service for use with business intelligence tools.
3	It does not support SQL query language.	It supports SQL query language. But it does not fully support an SQL-standard.
4	It does not provide concept of Referential Integrity. Hence, no Foreign Keys.	It provides concept of Referential Integrity. Hence, there are Foreign Keys.
5	Its Primary database models are Document store and Key-value store.	Its primary database model is Relational DBMS.
6	It does not support Server-side scripting.	It supports user-defined functions for Server-side scripting in python.
7	Eventual Consistency and Immediate Consistency are used to ensure consistency in distributed system.	Immediate Consistency is used to ensure consistency in distributed system.
8	It does not offers API for user-defined Map/Reduce methods. But maybe implemented via Amazon Elastic MapReduce.	It does not offer API for user-defined Map/Reduce methods.
9	It supports secondary indexes.	It supports restricted secondary indexes.

Data Sources
S3
EMR
S3
Redshift
QuickSight
Upload data from multiple sources into S3
Use Amazon EMR to transform and cleanse the data (ETL)
Load formatted and cleansed data into S3
Redshift loads data in parallel optimizing it for fast analytics queries
Analyze and visualize data with Amazon Quicksight
Ad-hoc access to data using Athena
Athena can query aggregated datasets as well

Google Cloud
Cloud IoT
Cloud Function
PubSub
Cloud Storage
DataPrep
DataProc
DataFlow
AutoML
BigQuery
Cloud Datastore
Bigtable
Data Catalog
Cloud SQL
Memory-store
PubSub
Colab (EDA)
Datalab
Data Studio
Cloud Function
© 2021 Satish Chandra Gupta
CC BY-NC-ND 4.0 International Licence
creativecommons.org/licenses/by-nc-nd/4.0/
scgupta.me
twitter.com/scgupta
linkedin.com/in/scgupta

Open Source Data analytics Architecture

© Satish Chandra Gupta

@scgupta

Data analytics Architecture adopted by Pinterest

Data Architecture overview

Welldoc

Log Entry
Log Entry
Log Entry
Log Entry
Log Entry
Model
Log Entr

Log Entry
Model

Lambda Architecture

Lambda Architecture

Option 2: Separate serving layers

RAW DATA
DATA STORAGE
BATCH
PROCESSING
PROCESSING DATA
BATCH LAYER
QUERY
DATA
VIEW 1
VIEW 2
VIEW 3
BATCH VIEWS
PARTIAL AGGREGATIONS
MERGED VIEW
REAL-TIME VIEWS
VIEW 1
VIEW 2
VIEW 3
SERVING LAYER
STREAM
DATA INCREMENT
REAL-TIME
PROCESSING
PROCESSED INCREMENT
SPEED LAYER
@luminousmen.com

Lambda Architecture

Kappa Architecture

One pipeline for real-time and batch consumers

REAL-TIME LAYER
SERVING LAYER
Incoming data
Real-time engine
Serving backend
Queries
DATA
Historical data storage
Results storage

Kappa Architecture

SERVING LAYER
PARTIAL AGGREGATES
QUERY
REAL-TIME VIEWS
VIEW 1
VIEW 2
VIEW 3
DATA
STREAM
RAW DATA
IMMUTABLE DATA
LOG
REAL-TIME PROCESSING
PROCESSED DATA
SPEED LAYER
@luminousmen.com

What Architecture to Choose?

Kappa
Real-Time
Data in Motion

VS

Lambda
Batch
Data at Rest

KAI WAEHNER

Streaming Storage
(Kafka - Event Hub)
Spark code
Logic 1 - ICU pattern v1/v2
Data
Batch Storage
(s3 - Object Storage)
Spark code
Logic 2 -Exercise pattern
logic 1 - - ICU pattern v2

Streaming Storage
(Kafka - Event Hub)
Spark code
Data
Logic 1 - ICU pattern v1/v2
Logic 2 -Exercise pattern
HIVE
table

Stream
Data
Batch

2 tx, same employee, in < 5 minutes on same account
Tx by an employee on an acc from a different regions

Welldoc

bg bg bg bg bg bg bg bg bg
Xe Xe Xe Xe
ye
CEP
DROOLS
ne

Kappa architectures enable transactional workloads in addition to analytical workloads too.
A single pipeline for everything. No need for a Lambda architecture! Kappa enables transactional and analytical workloads.

Stream

LOB

Analytics

Stream

Batch

Spark itself doesn't manage these machines. It needs a cluster manager (also sometimes called *scheduler*)

- Standalone: Simple cluster-manager, limited in features, incorporated with Spark.
- Apache Mesos.
- Hadoop YARN
- Kubernetes

Azure Databricks
Azure Databricks
Cloud Pipeline Microservices
Domain Events
DataLakeMessage Publisher
Azure Data Lake Storage Gen2
Raw Zone (BRONZE)
Spark streaming
DELTA LAKE
Working Zone (SILVER)
Events
Date1
(40,00,000 = 8000 * 500 participants)
P1
Events(8000 records per day)
Spark streaming
Power BI

Operational Application/
Analytical Application
Storage
Compute
Communication
binary/append only
Random read/write
Scale, Complex
Files
Database
LOB
Analytical
Message
APi
Run business
(frequent update)
Manage business
Operational
(OLTP)
Analytical
(OLAP)
Batch
Stream
New sql
VoltDB, NuoDB or MemSQL
sql streaming

Operational Application/
Analytical Application
Storage
Compute
Communication
binary/append only
Random read/write
Files
Database
Run business
(frequent update)
Manage business
Operational
(OLTP)
Analytical
(OLAP)
New Sql
VoltDB, NuoDB or MemSQL
Scale, Complex
Message
APi

Operational Application/
Analytical Application
Storage
Compute
Communication
binary/append only
Random read/write
Scale, Complex
Files
Database
APi
Run business
(frequent update)
Manage business
Operational
(OLTP)
Analytical
(OLAP)
Message
New Sql + Streaming = Streaming SQL

AWS Step
Azure Logic App
Biztalk
Webmethods
Mule
ServiceNow
#

Deployment View

K8s Cluster
Node (VM)
Pod
Code
Pod Ip

K8s Cluster
Node (VM)
Node (VM)
Pod
Pod
Pod Ip
Code B
Code A

10.104.68.18
->10.104.68.21
->10.104.68.20
->10.104.68.19

Cluster
Service B
Node port
900
Cluster Ip
900
Node port
Pod
Pod Ip
Code B
Pod
Code A
Node (VM)
Node (VM)

Load Balancer
Cluster
Node port
Node port
Service B
Cluster Ip
900
900
Pod
Pod
Pod
Pod Ip
Pod Ip
Code B
Code B
Code A
Node (VM 1)
Node (VM 2)

Load Balancer
Node port :900
Node port :800
Node port :700
Cluster
Cluster Ip
Service A
Cluster Ip
Service B
Cluster Ip
Service C
Pod
Pod
Pod
Pod
Pod
Pod
Code A
VM1
Code A
VM2
Code B
Code B
Code C
Code C

lb
Cluster
19.12.24.3 :4000
Node
Node
19.12.24.4 : 4000
10.104.68.78
Kubernetes ingress/ Istio Ingress / nginx reverse proxy
route config
#....
10.104.68.18
10.104.68.75
Service for Service A
Service for Service B
10.104.68.21
10.104.68.20
10.104.68.19
Pod
Pod
Pod
Container
Container
Container
Container
Service A
Service A
Service A
Service B

APi Gateway
WAF
lb
Cluster
19.12.24.3 :4000
Node
Node
19.12.24.4 : 4000
10.104.68.78
Kubernetes ingress/ **Istio Ingress** / nginx reverse proxy
route config
#....
10.104.68.18
->10.104.68.21
->10.104.68.20
->10.104.68.19
10.104.68.18
Service for Service A
10.104.68.75
Service for Service B
10.104.68.21
Pod
Container
Service A
10.104.68.20
Pod
Container
Service A
10.104.68.19
Pod
Container
Service A
Container
Service B

APi Management
(developer portal, metering, Payment, …)
APi Gateway
(WAF, …)
LB
Cluster
19.12.24.3 :4000
Node
Node
19.12.24.4 : 4000
10.104.68.78
Kubernetes ingress/ Istio Ingress / nginx reverse proxy
route config
#….
10.104.68.18
->10.104.68.21
->10.104.68.20
->10.104.68.19
10.104.68.18
Service for Service A
10.104.68.75
Service for Service B
10.104.68.21
Pod
Container
Service A
10.104.68.20
Pod
Container
Service A
10.104.68.19
Pod
Container
Service A
Container
Service B

APi Management
(developer portal, metering, Payment, …)
APi Gateway
(WAF, …)
LB
Cluster
19.12.24.3 :4000
Node
Node
19.12.24.4 : 4000
10.104.68.78
reverse proxy
10.104.68.18
Service for Service A
10.104.68.21
Pod
10.104.68.20
Pod
10.104.68.19
Pod
Container
Side Car
Container
Service A
Container
Side Car
Container
Service A
Container
Side Car
Container
Service A
#Service Mesh

Cross cutting concerns

The **crosscutting concern** is a concern which is needed in almost every module of an application.

Service

Cache
Log
Exception
Transaction
Session
Config
Authorization
Authentication
...

Choose Cache

Browser
Forward Proxy
APi Gateway
Load balancer
Reverse proxy
Web Server/ Application Server
Db server
tomcat
IIS
gunicorn
jobs
web sphere
Apache

Web acceleration r

SSL/TLS Processing

Compression

Caching and Prefetching

Using a local private cache with a shared cache.

Speed Comparison

HOW CONTENT DELIVERY NETWORK FUNCTIONS

Forward Proxy vs Reverse Proxy

Logs

Application
Exception
Debug
Audit
Streaming Logs
Jaeger
Elastic APM
Infrastructure
Server Logs
web server
app server
database server
container
os
Metrics
server
storage
network
database
...
APM
Server
Uptime data

Host (Linux)
Service Container (Docker)
Web Server (Nginx) + PHP
Application Request (eg: order)
SQL Query 1
SQL Query 2
Call another services request (Eg: customer, cashflow...)
OS log
Syslog
Access, Error log
API Request, Audit...
Application Log
SQL Query log
Distributed tracing

Dev & Ops Teams
Log Data
Metrics Data
APM Data
Uptime Data
Web Logs
App Logs
Database Logs
Container Logs
Container Metrics
Host Metrics
Database Metics
Network Metrics
Storage Metrics
Real User Monitoring
Txn Perf Monitoring
Distributed Tracing
Uptime
Response Time
Activity
Resource

Config

User
<<runtime>>

preference
profile

Operational
<<deployment time>>

Throttling
keys
connection string
##

Developer
<<compile time>>

State Management

Concurrency View

Engineering for Performance

Performance Objectives
(Response Time, Throughput, Resource Utilization, Workload)

Performance Modeling
(Scenarios, Objectives, Workloads, Requirements, Budgets, Metrics)

Architecture and Design Guidelines
(Principles, Practices and Patterns)

Performance and Scalability Frame

Coupling and Cohesion
Communication
Concurrency
Resource Management
Caching, State Management
Data Structures / Algorithms

Measuring, Testing, Tuning

Measuring
Response Time
Throughput
Resource Utilization
Workload

Testing
Load Testing
Stress Testing
Capacity Testing

Tuning
Network
System
Platform
Application

Roles
(Architects, Developers, Testers, Administrators)

Life Cycle
(Requirements, Design, Develop, Test, Deploy, Maintain)

Performance Modeling

1. Identify Key Scenarios
2. Identify Workloads
3. Identify performance Objectives
4. Identify Budget
5. Identify Processing Steps
6. Allocate Budget
7. Evaluate
8. Validate

Iterate

Allocate Budget

Spread your budget across your processing steps

Performance Patterns
CPU
Memory
I/O
Eager Loading
Caching (Data, Output, Code)
Concurrency(cpu, IO)
Pooling (Object, Memory)
Queuing (Load Leveling)
Lazy Loading
GC
Fixed Size Memory Pool
Virtualization (Data, UI)
Singleton
Fly weight - object cache (singleton for a given state)
Disk I/O
Network I/O
Chunky / Batch..
Async I/O
Cache
Chunky / Batch instead of chatty
Split large chunks
Compression/ Minify/ ..
Async I/O
Pooling (Connection)
Acquire Late and Release Early
Fail Fast - Circuit Breaker
Cache

Reactive System
Concurrency
Compute bound
Thread priority
suspend/resume
TLS
...
Platform/Fwk
decides on number of
threads
You decide on number
of threads
80%
20%
Distributed
Non Distributed
Thread
Task
Parallelism
Data
Parallelism
Logic (d1,d2,d3,...dn)
Actor Pattern
Master Slave
Map Reduce
Imperative
Parallelism
80%
custom partition &
collation
98%
Must complete
(Flush)
2%
Akka
ErLang
Vert.X
projectreactor.io
...
Imperative
Parallelism
Declarative
Parallelism
98%
2%
2
Short running
Long running
1
Thread pool
(Background)
Background
Thread
Foreground
Thread
3
4
5

Global Queue
Local Queue
...
Local Queue
Worker Thread 1
...
Worker Thread p
Task 6
Task 4
Task 5
Task 2
ogram Thread

```
foreach(var item in items)
{
  ThreadPool.Do(() => do(item));
}
```

v/s

```
foreach(var item in items)
{
  Task.Factory.StartNew(() => do(item));
}
```

```
Parallel.ForEach<Item>(items, item
=>do(item));
```

v/s

```
foreach(var item in items)
{
  Task.Factory.StartNew(() => do(item));
}
```

System.Threading

PLinq

```
var q = from p in people.AsParallel()
        where p.Name == queryInfo.Name &&
              p.State == queryInfo.State &&
              p.Year >= yearStart &&
              p.Year <= yearEnd
        orderby p.Year ascending
        select p;
```

```
"abcdef".AsParallel().Select (c => char.ToUpper(c)).ToArray()
```

Partitioning Algorithms in PLINQ

- Several partitioning schemes built-in
  - Chunk
    - Works with any IEnumerable<T>
    - Single enumerator shared; chunks handed out on-demand

  - Range
    - Works only with IList<T>
    - Input divided into contiguous regions, one per partition

  - Stripe
    - Works only with IList<T>
    - Elements handed out round-robin to each partition

  - Hash
    - Works with any IEnumerable<T>
    - Elements assigned to partition based on hash code

- Custom partitioning available through Partitioner<T>
  - Partitioner.Create available for tighter control over built-in partitioning schemes

Reactive Programming != Reactive System

The Reactive Manifesto

The Reactive Manifesto (http://www.reactivemanifesto.org/) is a document defining the four reactive principles

Responsive
Scalable
Resilient
Event-Driven

reactive manifesto

Using **reactive programming does not build a reactive system**. Reactive systems, as defined in the reactive manifesto, are an architectural style to *build responsive distributed systems*. Reactive Systems could be seen as distributed systems done right. A reactive system is characterized by four properties:

- **Responsive**: a reactive system needs to handle requests in a reasonable time (I let you define reasonable).
- **Resilient**: a reactive system must stay responsive in the face of failures (crash, timeout, 500 errors…), so it must be designed for failures and deal with them appropriately.
- **Elastic**: a reactive system must stay responsive under various loads. Consequently, it must scale up and down, and be able to handle the load with minimal resources.
- **Message driven**: components from a reactive system interacts using asynchronous message passing.

Despite the simplicity of these fundamental principles of reactive systems, building one of them is tricky. Typically, each node needs to embrace an asynchronous non-blocking development model, a task-based concurrency model and uses non-blocking I/O.

Cricket score,
Stock exchange,
Bidding,
time,

Multiple values (*)

Single value (1)

Array

```
res =
  stocks
  .filter(q => q.symbol == 'FB')
  .map(q => q.quote)

res.forEach(x =>
  ...
```

Observable

```
res =
  stocks
  .filter(q => q.symbol == 'FB')
  .map(q => q.quote)

res.forEach(x =>
  ...
```

Object

```
var y = f(x);
var z = g(y);
```

Promise

```
fAsync(x).then(...);
gAsync(y).then(...);
```

Synchronous

Asynchronous

- Asynchronous data streams
- **Everything** is a stream

XHR polling
SSE
WebSocket
GET /poll
GET /poll
GET /sse
GET /ws
HTTP protocol
EventSource protocol
WebSocket protocol

Master
Slave
Slave
Slave

dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]
dict[Key,value]

Actor Pattern

```
var c = new ActionBlock<int>(i =>
{
    Process(i);
});

for(int i = 0; i < 5; i++)
{
    c.Post(i);
}
```

4

Concurrency
I/O bound
Sync I/O
Async I/O
Kernel supports
Non blocking api's
synchronous
blocking
Blocking
I/O
Non Blocking
I/O
Thread is blocked
while hardware
performs the work.
Thread is not blocked, But
there is still a thread lower
down the stack which is
blocked on the IO
The kernel doesn't block
your thread and returns from
the IO call immediately.

How Windows does Synchronous I/O
.NET
FileStream fs = new FileStream(...);
Int32 bytesRead = fs.Read(...);
Win32
User-Mode
ReadFile(...);
IRP
Windows
Kernel-Mode
(Windows I/O Subsystem Dispatcher code)
1
2
3
4
5
6
7
8
9
Your thread blocks here!
Hardward does I/O;
No threads involved!
NTFS Driver
IRP
Queue
Network
DVD-ROM
RS-232

How Windows does Asynchronous I/O

Syncronization

No Lock Update

Nonblocking synchronization constructs

- volatile
- Atomic.

Wait

Simple blocking methods

- Sleep (bad)
- Join
- Spin

- Sleep (bad)
- Suspend
- Abort
- SetPriority
- Resume

Resource Lock

Locking constructs

- *Exclusive* locking
- nonexclusive locking
  - Semaphore
  - reader/writer locks.

Notify

Signaling constructs

- event wait
- Latch
- Barrier

Scalability Patterns
Distributed Computing
Cache
Principles
Communication
Scale Up (bigger box)
CQRS
Master Slave
Actor pattern
Map Reduce
Scalability Cube
- Cloning
- Splitting y axis (vertical) column
- Sharding z axis (horizontal) rows
Materialized View
Stateless
Remove resource Affinity
CAP
BASE instead of ACID
Reactive Systems
Message Queue
Event Driven Architecture

Performance
Anti Patterns
CPU
Memory
I/O
Single threaded
Lazy Loading ?
Excessive dynamic allocation
GOF Singleton pattern
Chatty
Sync I/O
Extraneous Fetching (fetching more)
Not Pooling shared resource
Acquire early and Release Late

Application Security View

Application Security

Infra Security

AAA

- Authentication (first line of defence)
- Authorization
- Audit (last defence) who did , what
- Input validation (WAF, …)
- Asset Handling (in rest, in transit)
- Exception Handling
- Session Handling
- Key Management (vault)

- Authentication (first line of defence)
- Authorization
- Audit (last defence) who did , what
- Input validation (fwk)
- Asset Handling (in rest, in transit)
- Exception Handling (fwk)
- Session Handling (fwk)
- Key Management

STRIDE

- Spoofing
- Tampering
- Repudation
- Information Disclosure
- Deniel of Service
- Elevation of Privileage

Threat Modeling

Threat Modeling Method	Features
STRIDE	• Helps identify relevant mitigating techniques • Is the most mature • Is easy to use but is time consuming
PASTA	• Helps identify relevant mitigating techniques • Directly contributes to risk management • Encourages collaboration among stakeholders • Contains built-in prioritization of threat mitigation • Is laborious but has rich documentation
LINDDUN	• Helps identify relevant mitigation techniques • Contains built-in prioritization of threat mitigation • Can be labor intensive and time consuming
CVSS	• Contains built-in prioritization of threat mitigation • Has consistent results when repeated • Has automated components • Has score calculations that are not transparent
Attack Trees	• Helps identify relevant mitigation techniques • Has consistent results when repeated • Is easy to use if you already have a thorough understanding of the system
Persona non Grata	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Has consistent results when repeated • Tends to detect only some subsets of threats
Security Cards	• Encourages collaboration among stakeholders • Targets out-of-the-ordinary threats • Leads to many false positives
hTMM	• Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated
Quantitative TMM	• Contains built-in prioritization of threat mitigation • Has automated components • Has consistent results when repeated
Trike	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has automated components • Has vague, insufficient documentation
VAST Modeling	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Has automated components • Is explicitly designed to be scalable • Has little publicly available documentation
OCTAVE	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Is explicitly designed to be scalable • Is time consuming and has vague documentation

PASTA

STAGE I - Definition of the Objectives (DO)	• DO 1.1 - Document the business requirements • DO 1.2 – Define the security/compliance requirements • DO 1.3 – Define the business impact • DO 1.4 – Determine the risk profile
Stage II - Definition of the Technical Scope (DTS)	• DTS 2.1 – Enumerate Software components • DTS 2.2 – Identify Actors & Data Sinks/Source • DTS 2.3 – Enumerate System-Level services • DTS 2.4 – Enumerate 3rd Party infrastructure. • DTS 2.5 – Assert completeness of secure design.
Stage III - Application Decomposition and Analysis (ADA)	• ADA 3.1 – Enumerate all application use cases • ADA 3.2 – Document Data Flow Diagrams (DFDs) • ADA 3.3 – Security functional analysis & the use of trust boundaries
Stage IV - Threat Analysis (TA)	• TA 4.1 – Analyze the overall threat scenario • TA 4.2 – Gather threat information from internal threat sources • TA 4.3 – Gather threat information from External threat sources • TA 4.4 – Update the threat libraries • TA 4.5 – Threat agents to assets mapping. • TA 4.6 – Assignment of the probabilistic values for identified threats
Stage V - Weakness and Vulnerability Analysis (WVA)	• WVA 5.1 – Review/correlate existing vulnerabilities • WVA 5.2 – Identify weak design patterns in the architecture • WVA 5.3 – Map threats to vulnerabilities • WVA 5.4 – Provide Context risk Analysis based upon Threat-Vulnerability • WVA 5.5 – Conduct targeted vulnerability testing
Stage VI - Attack Modeling & Simulation (AMS)	• AMS 6.1 – Analyze the attack scenarios • AMS 6.2 – Update the attack library/vectors and the control framework • AMS 6.3 – Identify the attack surface and enumerate the attack vectors • AMS 6.4 – Assess the probability and impact of each attack scenario. • AMS 6.5 – Derive a set of cases to test existing countermeasures. • AMS 6.6 – Conduct attack driven security tests and simulations
STAGE VII - Risk Analysis & Management (RAM)	• RAM 7.1 – Calculate the risk of each threat • RAM 7.2 – Identify countermeasures and risk mitigations measures • RAM 7.3 – Calculate the residual risks • RAM 7.4 – Recommend strategies to manage risks

STRIDE
Spoofing
Elevation of Privilege
Tampering
Repudation
Information Disclosure
Deniel of Service
Authentication
Session Handling
Authorization
Input Validation
Integrity
Audit
Exception Handling
Confidentiality
Availability

App Concerns

- 3A
  - Authentication (who)
  - Authorization (what)
  - Audit
- Asset
  - Confidentiality (on wire, in rest)
  - Integrity (on wire, in rest)
- Input Validation (70%)
- Exception Handling
- Session Management
- Key Management

Infra Concerns

- Updates/ patches
- Throttling
- vnet/ subnet
- L4 FW
- L7 FW (WAF)
- VPN

Step 1. Authentication

Customer
Token
UI
Service1
Claims
Service2
Claims + event
Claims + event
Service3

App1
App2

OAuth
Server
Pwd
User
Browser
Pwd
App
Server

OAuth
Server
Pwd
User
Browser
App
Server

OAuth
Server
Auth0
Pwd
Token
User
Browser
Token
App
Server

Org
Identity
Server
Pwd
Identity server
Token
Identity server
Token
OAuth
Server
Token
Auth0
User
Browser
Token
App
Server

```
# Something you know - pwd(oauth2) , security question <— 1
# Something you Have - otp, cert, rsa, email,  <— 2
# Where you are - country, office, … <— 3
# Something you are - voice, face, dna, ..
# Something you do - behaviour
```

Centralize

Resource Server
Client Application
APP
Access data
Access service
Issue authz code/
access token
Delegate
authentication
authorization
Grant access
Resource Owner
Authorization Server

Step 2. Authorization

App1: role->claims
App2: role->claims
App3: role->claims
OAuth Server
Pwd, App1
Token + Claims
User
Browser
Token + Claims
App1 Server

Role Based Access Control (RBAC)

Attribute Based Access Control (ABAC) / Claim Based Access Control (CBAC)

- Date
- Time of day
- Location of data
- Location of user
- Device

- Date created
- Date Modified

ABAC is an extension of RBAC

Access Control Mechanism evaluates

a) Rules (RBAC),
b) Subject Attributes,
c) Object Attributes
d) Environment Conditions
to compute a decision

```
{
"UserData": {
"userId":"#{USER_ID}",
"email":'#{USER_EMAIL}",
"first_name':"#{USER_FIRST_NAME}",
"last_name":'#{USER_LAST_NAME}',
"name":"#{USER_NAME}",
"roles": {
"#{ROLE_NAME}":"#{ROLE_ID}"
},
"permissions": {
"#{PERMISSION_NAME}":'#{PERMISSION_ID}",
}
},
"exp':148761231
}
```

Step 3. Audit

who - Identity

what - Action

where - Component/Service/Object/Method

when - Timestamp:

Step 4. Privacy and Confidentiality

Step 5. Cross cutting concerns

Protecting Sensitive data
Secure Configuration
Protecting Sensitive data
Preventing Session Hijacking
Handling Exception
Web App
API Application
Authenticating User
Preventing Parameter Manipulation
Validating Input
Authorizing User
Auditing
Encryption, Hashing Sensitive Data

Public IP address
vnet
WAF
Application Gateway
Network Load Balancer
asg subnet
NGINX
Ingress controller
asg subnet
ToDo Portal
(Single Page Application)
NGINX
ANGULAR
kubernetes
Https(443)
ToDo API Service
(Api Application)
Flask
python
Apache Web Server
ToDo Calendar Service
(Background Application)
python
ToDo Prediction Service
(Background Application)
python
mongoDB
Atlas
mongoDB
Message queue
Azure Queue
Azure Data Lake

DevOps View

Agile Development
Daily Scrum
Product Backlog
Sprint Backlog
Sprint
2 weeks
Potentially Shippable Product
Burndown Chart
User Stories
Bugs/Issues
Continuous Feedback
Continuous Integration
Commit
Build
Unit Test
Static Code Analysis
Dev
Code Repo
CI Server
Artifacts
Monitor
DEV ENV
Ops
Repo Manager
Sprint Review
Daily Scrum
CI
Agile DevOps
CT
Sprint Phases
CD
Continuous Feedback
Continuous Feedback
CI: Continuous Integration
CT: Continuous Testing
CD: Continuous Delivery
Continuous Delivery
Approve
PO, PM
Ops
Monitor
Deploy
PRD ENV
End Users
Deploy
Repo Manager
Continuous Feedback
Continuous Testing
Commit
Build
Dev
Code Repo
Issue Tracking
Report
Ops
QA
Monitor
Test Scripts
Artifacts
CI Server
Repo Manager
STG ENV
Test Suite

Step 1. Containerizing

Step 2. Integrating infrastructure automation with CI/CD

Step 3. test automation and aligning QA with development

Pyramid Test

View Layer
(Manual, Automation)
Controller Layer
(unit test)
Model Layer
REST
API Layer
(API test)
1 to 1
Application Layer/ Facade Layer
(integration test)
(CRUD)
Data Access Layer
Domain Layer
(unit test)

Step 4. Setup Monitoring

Reference

Collaborate
Build
Test
Deploy
Run
Application Lifecycle Mgmt.
JIRA
mingle
Trello
Visual Studio
Team Foundation Server
PivotalTracker
Basecamp
asana
PHABRICATOR
SCM/VCS
git
subversion
mercurial
GitHub
Gogs
Bitbucket
GitBucket
GitLab
Testing
Selenium
Jasmine
NOCHA
GAUNTLT
OWASP
ZAP
Gatling
JUnit
KARMA
FitNesse
N unit
QUnit
Test NG
cucumber
Galen Framework
LOAD IMPACT
JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
Pa11y
specflow
Newman
xUnit.net
Deployment
Octopus Deploy
DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
SSH
JUJU
ElasticBox
Spinnaker
Cloud / IaaS / PaaS
amazon web services
heroku
Dokku
Flynn
Microsoft Azure
CLOUDFOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
applog
OPENSHIFT
Engine Yard
Communication & ChatOps
slack
HipChat
#irc
flowdock
Microsoft Teams
RYVER
Mattermost
ROCKET.CHAT
COG
Nectar
LITA
HUBOT
CI
wercker
snap
TeamCity
Jenkins
Bamboo
crone.io
circleci
Travis CI
go
CODESHIP
Continuous Delivery
Config Mgmt./Provisioning
puppet labs
CHEF
ANSIBLE
CFEngine
SALTSTACK
PowerShell DSC
VAGRANT
TERRAFORM
Orchestration & Scheduling
NESOSPHERE
MARATHON
SWARM
kubernetes
Nomad
RANCHER
MESOS
Knowledge Sharing
jekyll
github pages
Confluence
HUGO
Mark down
Read the Docs
apiblueprint
RAML
FLARUM
OPENAPI
graphviz
Discourse
reddit
Build
sbt
Gradle
GRUNT
docker
Maven
.js
Gulp
Ant
<APACHE ANT>
PACKER
MSBuild
Leiningen
Rake
Database Management
DBmaestro
DBDeploy
Flyway
redgate
Flocker
LIQUIBASE
Artefact Management
QUAY
DockerHub
docker REGISTRY
Bower
JFrog Artifactory
python Package Index
nuget
archiva
npm
Sonatype Nexus
BI / Monitoring / Logging
logstash
elasticsearch
splunk
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic
VIZCERAL
Prometheus
PINPOINT
dynatrace
Runscope
sensu
Atlas
RIEMANN
APImetrics
Grafana
graphite
RAYGUN
STATSD
Airbrake.io
Rollbar
pagerduty
OOC
OpsGenie
Keen IO
http://www.jamesbowman.me

Step 5. SecOps

Arch Risk

- Req1
- Req2
- Req3
- …

- App 1
- App 2
- App3
- …

Performance	As a user	I want to book a home	From Web Portal	During Peak Load	The booking is completed	In < 2 secs

 # Compression
 # Concurrency
 # Cache
 # ...

- ATAM *
- SAAM
- ARID
- HASARD
- …

Evaluate Architecture (ATAM)

identify all Architectural approaches (design)

A1 (CQRS)
A2(Sharding)
A3 (Caching)
...

identify all quality requirements (requirements)

s1 (< 5 sec)
s2 (99.99%)
s3 (...)
...

Utility Tree

analyse Scenario -> Approach

S1 -> A1, A2
S2 -> A6,A8, A9
S3-> ?
S4 -> A6, ?

=> Risk & trade off's

brainstorm for scenarios

s8
s9
s10
...

analyse Scenario -> Approach

S8 -> A1, A2
S9 -> A6,A8, A9
S10-> ?

=> Risk & trade off's

Evaluation Team

Offering mitigation strategies is *not an integral part of* the ATAM.
ATAM is about locating architectural risks.

ATAM
Phase 1
Phase 2
Phase 3
Phase 4
Presentation
Investigation & Analysis
Testing
Reporting
- Present ATAM
- Present business drivers
- Present architecture
- Find architectural approaches
- Build Utility tree
- Analyze approaches
- Prioritize scenarios
- Analyze the architectural approaches
- Present the results
Steps 1 to 3
Steps 4 to 6
Steps 7 to 8
Step 9

Utility
Performance
Data Latency
(M,L) Minimize storage latency on customer DB to 200 ms.
(H,M) Deliver video in real time
Transaction Throughput
Modifiability
New product categories
Change COTS
(L,H) Add CORBA middleware in < 20 person-months
(H,L) Change web user interface in < 4 person weeks
Availability
H/W failure
(L,H) Power outage at Site 1 requires traffic re-direct to Site 2 in < 3 secs
(M,M) Restart after disk failure in < 5 mins
(H,M) Network failure is detected and recovered in < 1.5 mins
COTS S/W failures
Security
Data confidentiality
(L,H) Credit card transactions are secure 99.999% of time
Data integrity
(L,H) Customer database authorization works 99.999% of time

Scenario Utility Tree

- Utility
  - Performance
    - Data latency
      - Minimize storage latency on customer DB to 200 ms
      - Deliver video in real time
    - Transaction throughput
      - Maximize average throughput to authentication server
  - Modifiability
    - New Product Categories
    - Change COTS
      - change web user interface in < 4 person weeks
  - Availability
    - Hardware Failure
      - power output at site 1 requires traffic redirect to site 3 in < 3 s
      - network failure is detected and recovered in < 1,5 min
  - Security
    - Data confidentiality
      - customer database authorisation works 99,999% of time

Scenario Brain Storming

Sc#	Description	Quality Att.	Votes
4	Dynamically replan a dispatched mission within 10 minutes.	Performance	28
27	Split the management of a set of vehicles across multiple control sites.	Performance, Modifiability, Availability	26
10	Change vendor analysis tools after mission has commenced without restarting system.	Integrability	23
12	Retarget a collection of diverse vehicles to handle an emergency situation in less than 10 seconds after commands are issued.	Performance	13
14	Change the data distribution mechanism from CORBA to a new emerging standard with less than six person-months' effort.	Modifiability	12

Business Drivers

Phase - 1

Architec
t Centric

Concentrates on eliciting and analyzing architectural information.

Phase - 2

Stakeholder Centric

Elicits points of view from a more diverse group of stakeholders, and verifies the results of the first phase

Scenario:	E-connector looses connection with SAP
Attribute:	Availability
Stimulus:	Temporary network fault
Response:	The system has an overall availability of 99,25% (max 2 hour down/ month)

Architectural decision	Sensitivity	Trade-off	Risk	Non-risk
DCTM content server runs on a clustered environment with 2 nodes	Common mode failure can not be handled			Probability of common mode failure is low
Integration relationship between SAP and Documentum is 'data consistency‘ and is not protected	Human user must report malfunction		From complaint to resolution > 2 hours	

Scenario:	Invoice poster needs e-document for data entry in SAP/R3
Attribute:	Perfomance-Latency
Stimulus:	Document request to Documentum
Response:	Document is available for processing in less than 10 s

Architectural decision	Sensitivity	Trade-off	Risk	Non-risk
E-documents are scanned in color at 200 dpi	Size of document is sensitive to quality of scanning	Usability vs Performance	Document too large for roundtrip in 10 s.	
E-documents are not cached	Every document must be fetched from DMTM	Development cost vs bandwidth cost	Document roundtrip time exceeds 10 s.	

DropBox
Logical View

View
View Model
Model
Boundary
Meta data
Control
Sync Service
Domain
? God
Chunk Service
Conflict Mgmt Service ?
Upload Download Service
Local Watcher
Cloud Watcher
Metadata Service
Boundary
NetworkService

Share Service
Or
ACL Service
Rdbms
Notification Service
?
File Service
Object
Storage
Ledger Service
Wide Column db
(column-family)
Million of rows
Query on multiple columns
Document
Store
Quota/ metering
Metadata Service
Rdbms
API
Domain
Data Access Layer
User Management
Rdbms

Upload
Download
Service
Chunk
Service
Cloud
Watcher
Quota/ metering ?
Pull during startup (catch up)
Push while running
Grpc
File
Authorization
Service
<<API>>
Rdbms
File Service
<<API>>
Object
Storage
Metadata
Service
<<API>>
Rdbms
User
Management
<<API>>
Rdbms
Notification
Service
Grpc
Should we ?
Audit Service
Wide column

Firewall (TCP)
API Gateway /API Management / WAF (Http)
Load Balancer
Cluster
Nodes
Kubernetes ingress/ Istio Ingress / nginx reverse proxy
Notification Service
File Authorization Service <<API>>
Metadata Service <<API>>
User Management <<API>>
Object Storage
Wide column
Rdbms
Message Queue

Uber reference architecture

(1)
(2)
(3)
(4)
(5)
Rider
Log On; See Drivers on Map
Enter Destination; View Fare
Confirm Pickup Location
Assigned Driver; Get Updates
View Trip
Rate Driver
Current Location
Drivers' Locations
Destination
Fare
Pickup Location
Confirm Driver
Send Rider Location; Refresh Driver Location
Trip Info
Rating Request
Rating
Platform
Find Nearest Drivers
Calculate Fare
Match Driver
Notify Rider
Start Trip
End Trip
Pickup Request
Accept Request
Refresh Rider Location
Confirm Pickup
Trip Info
Confirm Dropoff
Rating Request
Rating
Driver
View Request; Accept Request; Drive Over
Confirm Pickup
View Trip
Confirm Dropoff
Rate Rider
Current Location
Current Location
Current Location
Current Location
(0)
Report Location
Report Location
Report Location
Report Location

Uber/Lyft/Ola System Design

1.collect Telemetry
2.secure
3. Analyse
Customer
Device
Feature engineering
Masking
Anonymoization
Device
Gateway
wss
Internet
wss
mqtt
Message engine (7)
Stream Analytics
Analytics
clustering
sequence modelling
...
Batch Analytics
Csv
Object Storage
Device
APi
Wide column
Tableu
Looker
Power Bi
Reporting server
DWH
Business User

Decompose Applications into Modules

Web Portal
View (UI Layout)
Controller (UI logic & state)
Model (invokes domain)
Report
(power bi, tableu,..)
JSON(document)
Rest
Booking Service
API
Domain
Data
OO
Property Service
API
Domain
Data
Datawarehouse
Normalised
Form (3rd)
Database
Periodic
Insight Service
Filter
Filter
Filter
Formatting, group, removing, ...

Rest
Rest
Rest
Property
Search Service
Property Service
Booking Service
Booking
History Service
API
Domain
Data
API
Domain
Data
API
Domain
Data
API
Domain
Data
?
?
Text
Document
File
storage
Rdbms
Wide Column
?
?
Data Lake
Landing Zone
Staging Zone
Reporting Zone
Analytical Service

Portal
docker
Rest
Rest
Booking Service
docker
Booking
History Service
docker
kubernetes
?
Rdbms
Wide Column

Rest
@ 1
@ 2
Rest
@ 3
Async events (web hooks)
Booking Service
Booking
History Service
@ 4
docker
docker
@ 5
Pull
will create scale issues
Booking
Sync Service
Rdbms
Wide Column
Db log stream
@ 6
Push

UI App
CRUD
Order Command
Service

CQRS pattern
event sourcing
Eventual consistency vs Immediate Consistency

CQRS pattern
event sourcing
event store
Eventual consistency
Immediate Consistency (pessimistic vs optimistic / Async)
Compensatable transaction
SAGA

Reports
UI App
WAF
API Gateway
CUD (rest)
Read (rest)
Current State
Account Query Service
Do
Undo
APACHE STORM
Stream Analytics
Order Request Service
Cmds Events
Cmds Events
Create Order Command
CUD
Domain Event Bus
Cmds Events
Create Order Command
Order Created Event
Out of Stock
Cmds Events
DomainEvent Service
Order Command Service
Order Query Service
Do
Undo
Data Lake
(Object Store/ Distributed file storage)
Data Mart
Columnar
Staging zone
Landing zone
Full State
Current State
Normalised
Write Friendly
ACID
Lots of joins
Not read friendly
Current State
Materialized Views
DeNormalised
Not Write Friendly
No joins
read friendly
Batch Analytics
Spark

Event
Event
Cmd 2
Event 2
Cmd 2
Cmd 1
Cmd 1
Event 1
Event
Event
Loan Request Service

Db
Booking
Sync Service
Booking
Keeping

Booking
Sync Service
Db
Queue
Booking
Sync Scheduler Service
Booking
Keeping

@ Requirements
———————-
Context View
Functional View
Constraints
Quality View *
Assumptions

@ Design
———————-
Logical View
  - System Decomposition (applying patterns)
      * SOC / SRP / High Cohesion /
      * low coupling
  - decide approach for Storage
  - decide approach for Compute Deployment
  - decide approach for communication *
  - decide approach for cross cutting concerns
...

@ Eval
———————-

- Event Sourcing
- DDD
  - ubiquitous language
  - Domain Events
  - Aggregates
  -

Tech	Domain
# api (CUD)	# Domain event