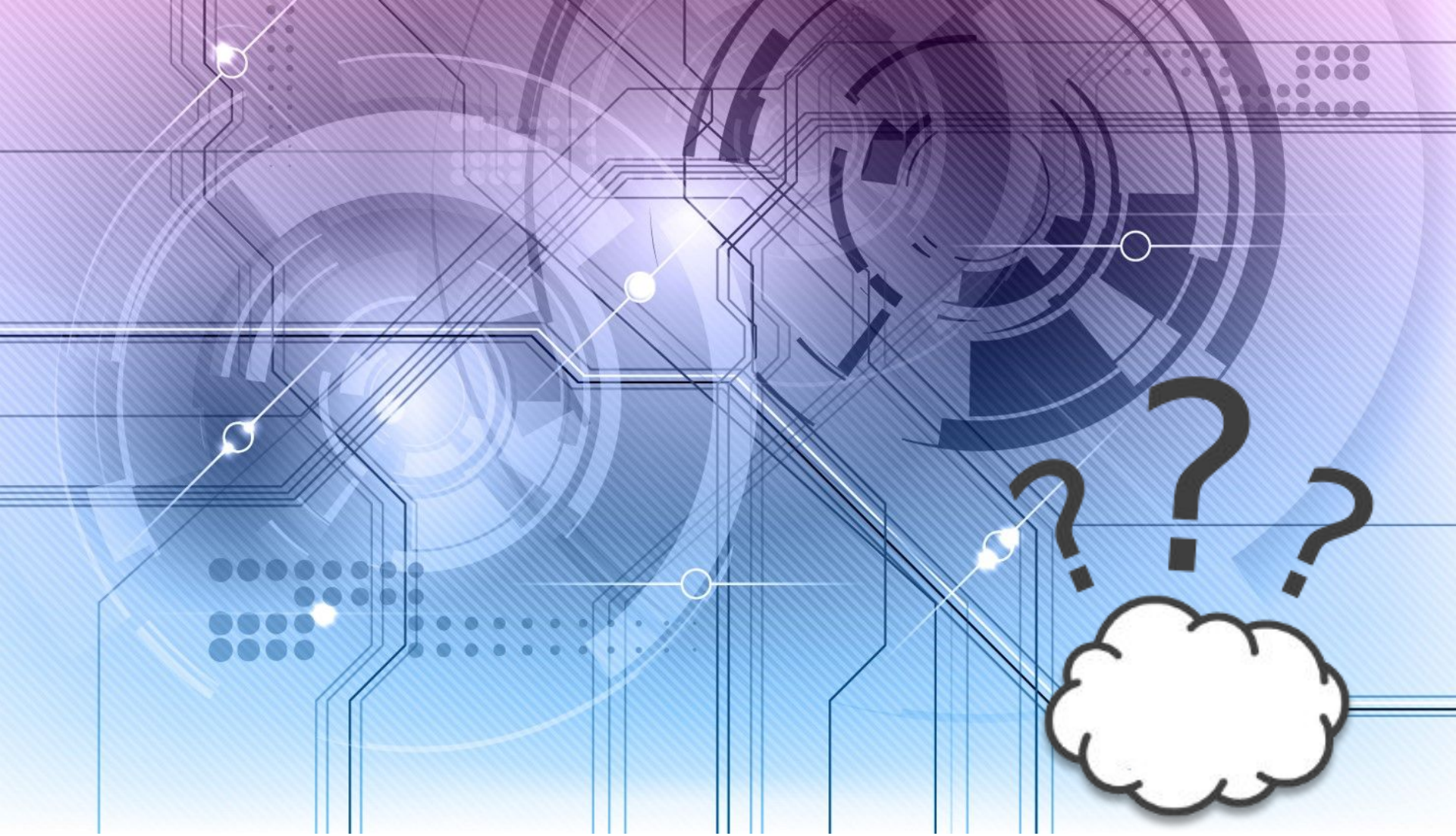
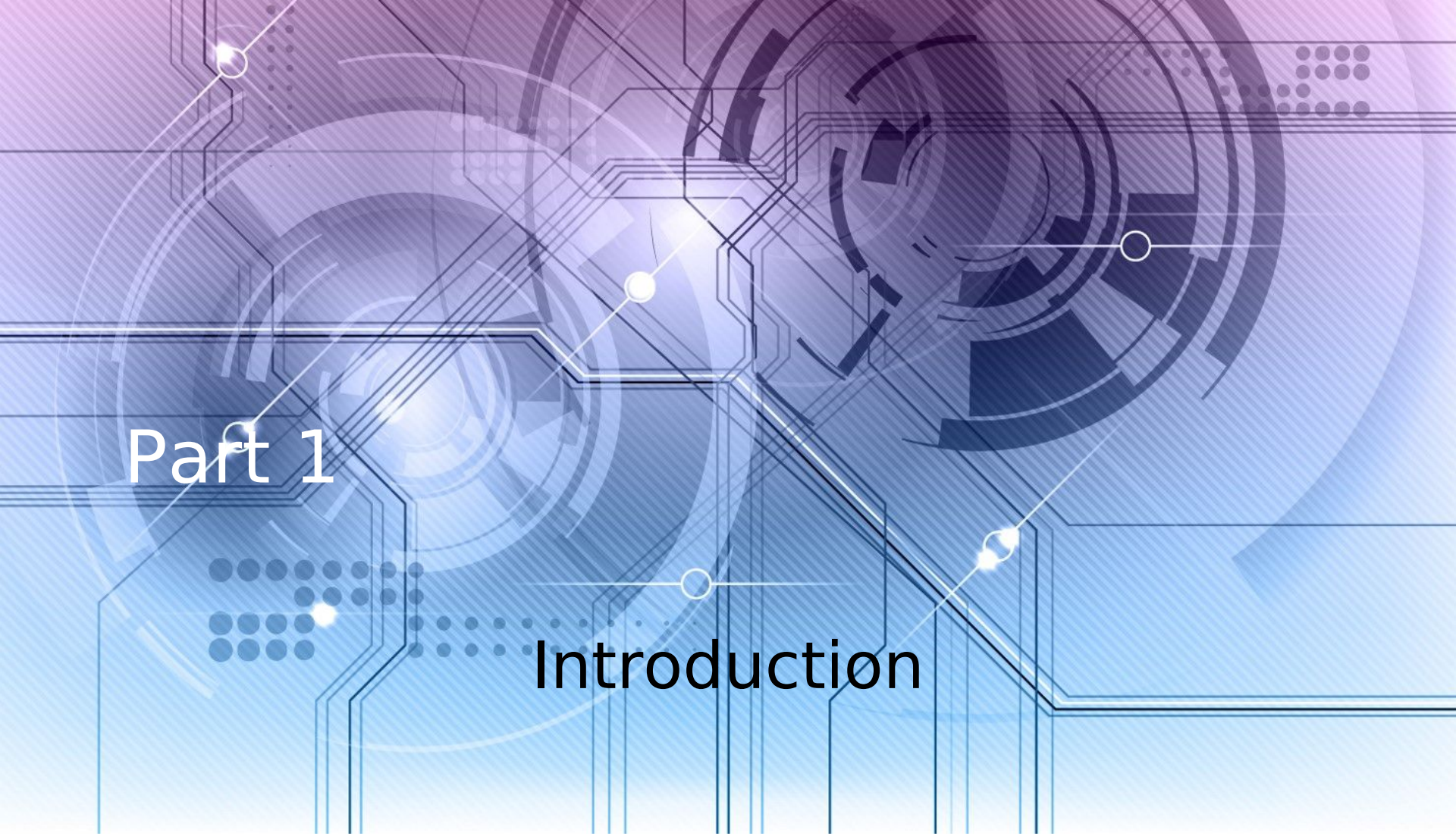


Cloud Computing

By
Nirmallya Mukherjee



Part 1

Introduction

A few stories

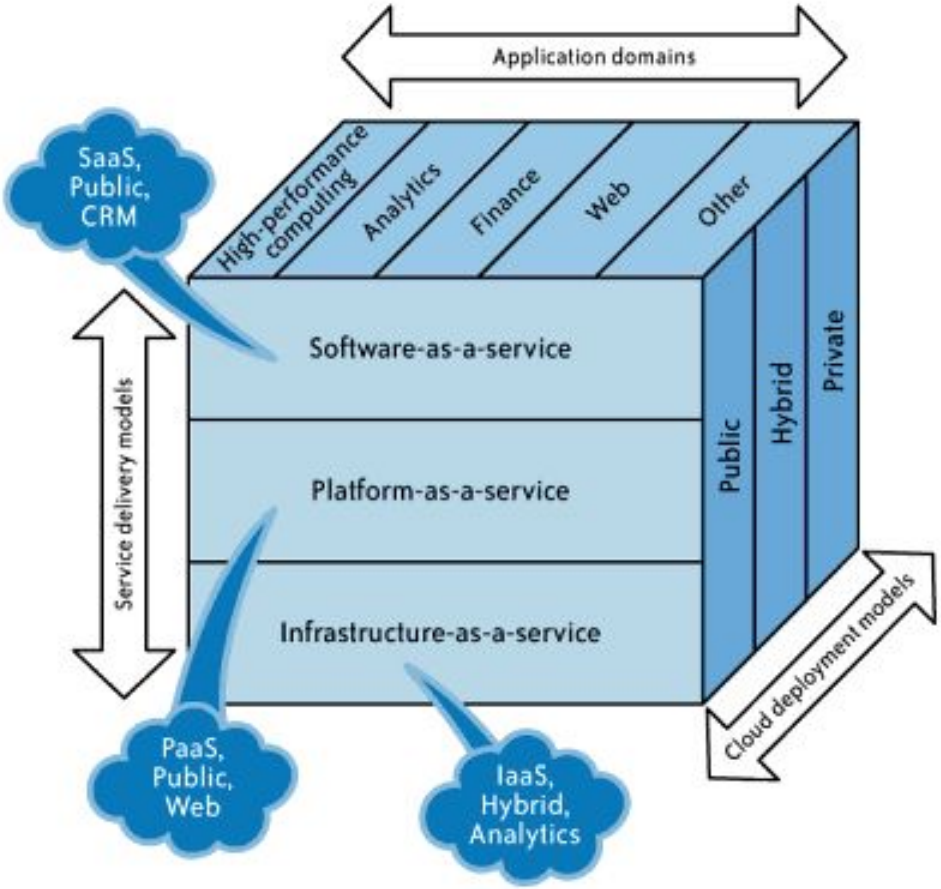
- Instagram launch
- Flipkart billion dollar sale day
- Did you hear about ASP? (this was early 2000)
- How many of us want to build or have already installed a nuclear power plant at home?
- One final thing … familiar with SETI@HOME?

Myths of cloud computing

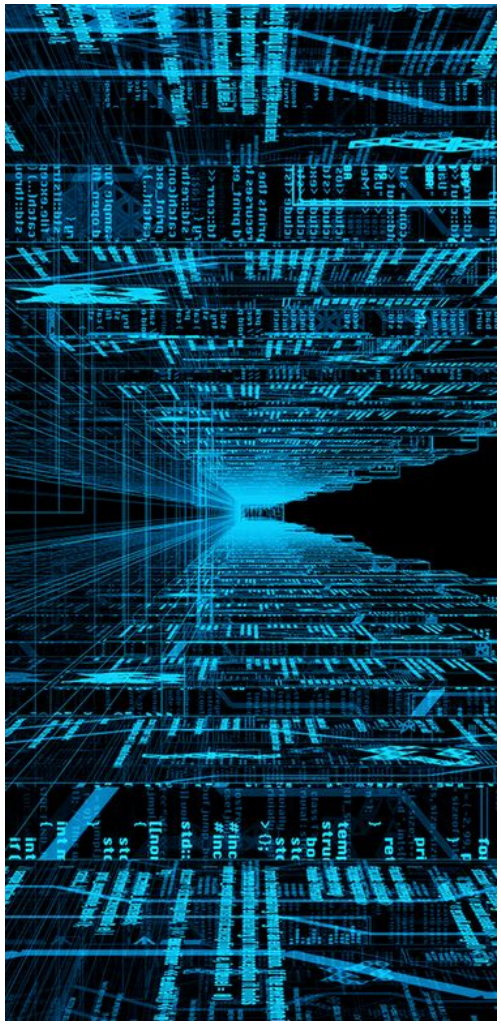
- There's one single "Cloud"
- All you need is your credit card
- The cloud reduces your workload
- Integration (two versions)
  - You can seamlessly blend your private "Cloud" (your virtualized datacenters) with public providers
  - You won't ever be able to seamlessly blend your public and private clouds
- The cloud always saves you money
- A cloud provider can guarantee security
- If you are using virtualization, you are doing cloud computing
- Cloud computing is about technology

SPIDER

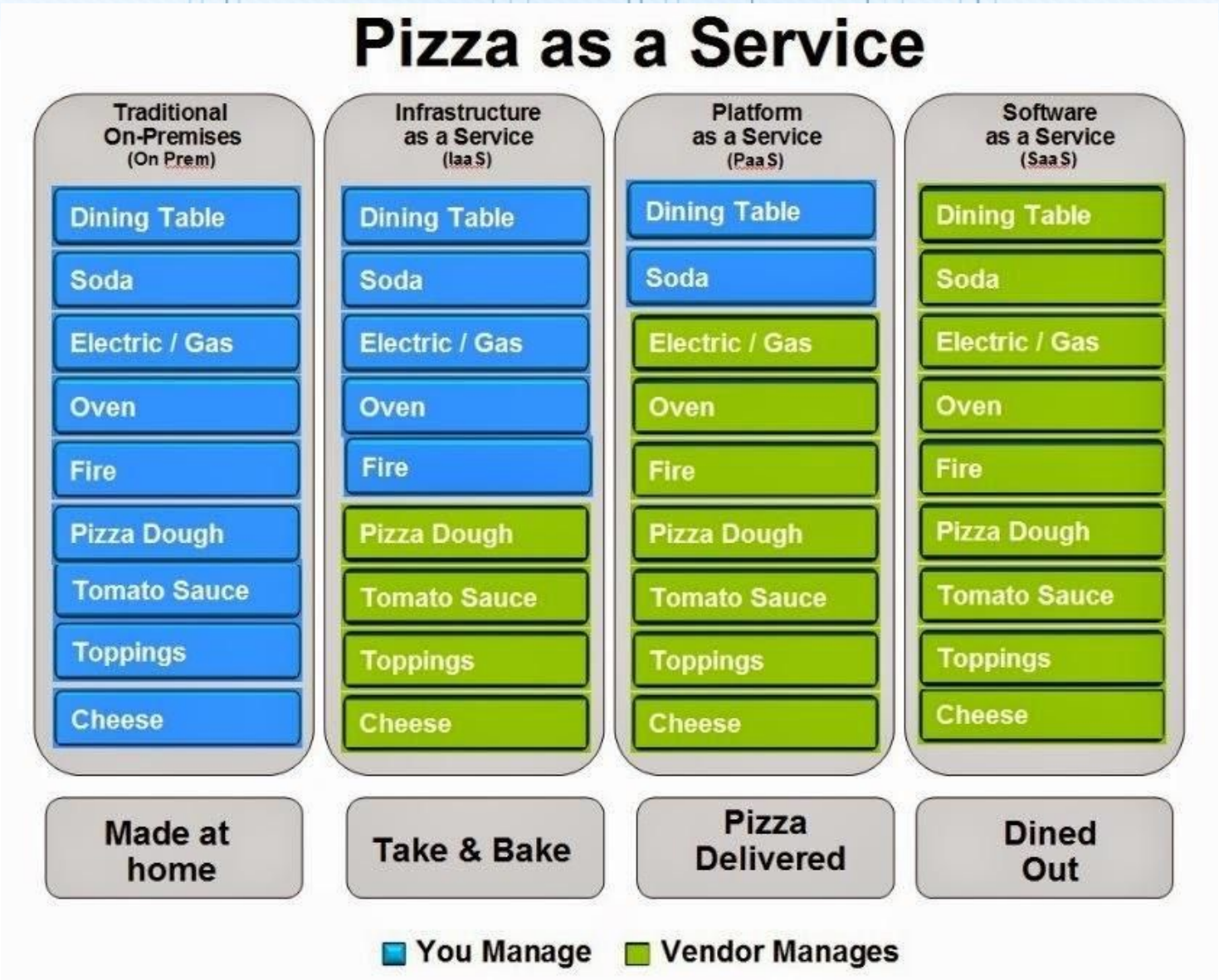
SPIDER stands for "**S**aaS", "**P**aaS", "**I**aaS", "big**D**ata", "**E**lastic" & "**R**esilient"



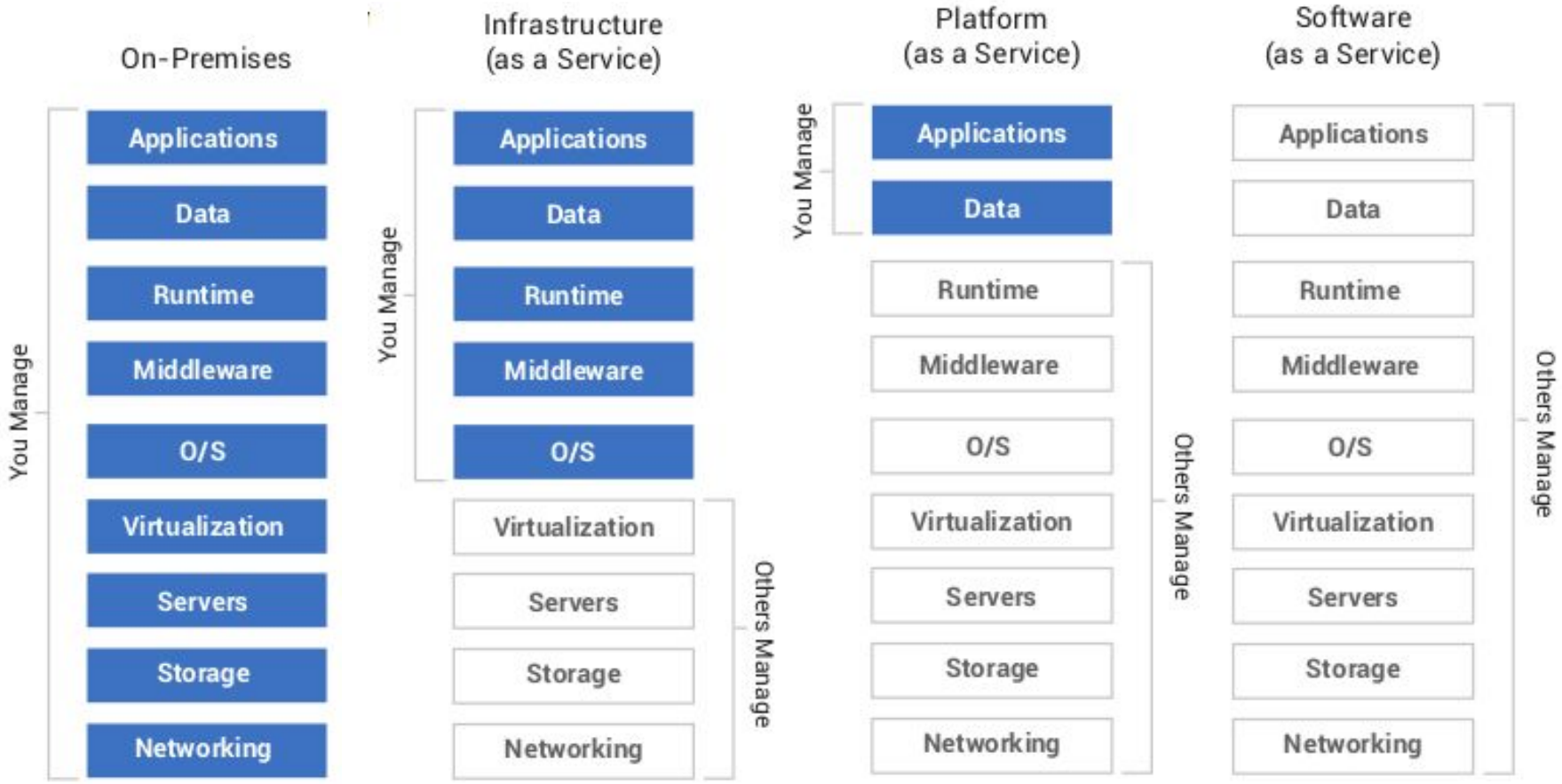
+



A perspective



A perspective - continued



Cloud computing attributes

- Choice of provider
  - Based on business need and partnership models
  - Market competition is good for consumers as it avoids monopoly
- Agility
  - Procurement latency
- Open standards
  - Open source
  - Global consortium
- Provisioning
  - Ability to package and deploy to any provider
- Monitoring

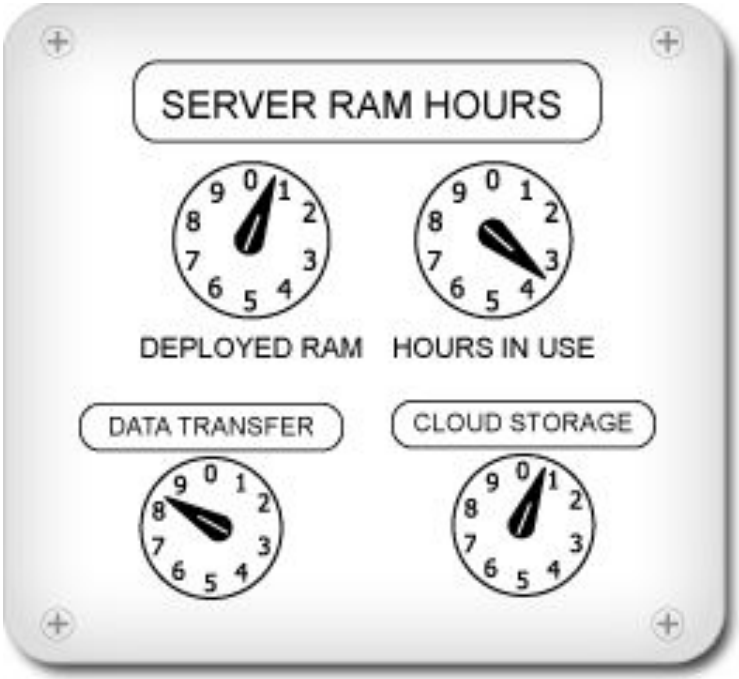
Debate

Analyze these statements -

- From tomorrow everyone needs to pay 2,000/month as electricity charges no matter what.
- Make your CFO spend cash from bottom line

Cloud computing attributes

- Service subscription
  - Customers avoid large upfront capital expense.
  - Pay as an ongoing operational expense.
  - Easily and quickly scale up or down based on business demand and only pay for what is needed (Economies of scale)
  - Better matches today's financial drivers



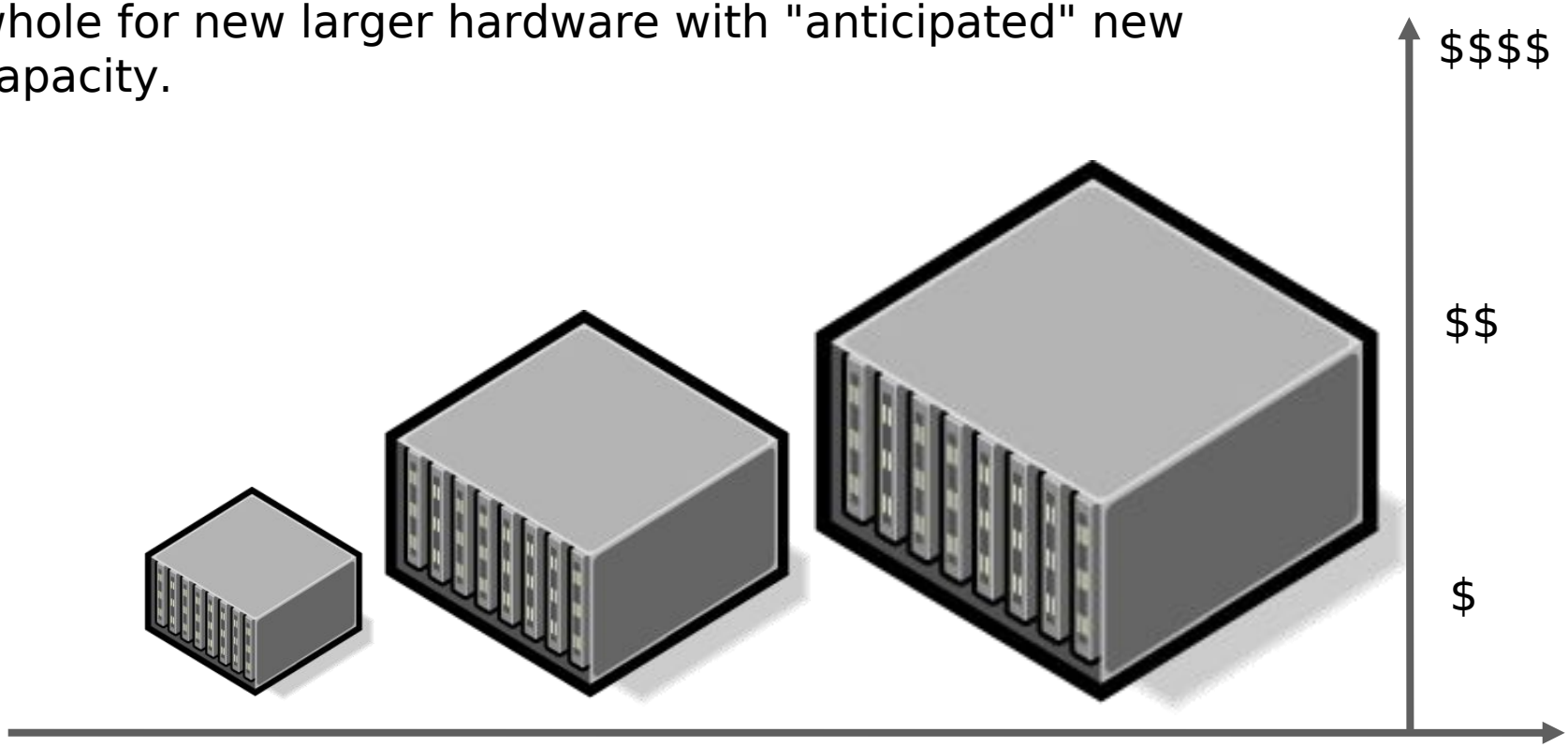
Cloud deployment models

- Public
  - A model where a service provider makes resources, such as applications and storage, available to the general public over the Internet
  - It is hosted and managed by a 3rd party from one or more data centers
  - What does this mean about a customer's data?
- Private (A service provider making computing resources available as a dedicated service to the subscriber)
  - Dedicated - Hosted within a customer's organization
  - Managed - Owned by an organization but managed by a 3rd party
  - Community - Hosted at the data center of a 3rd party
  - Private networking - on public cloud machine instances
- Hybrid
  - A combination of public and private deployment model

Classic scaling model

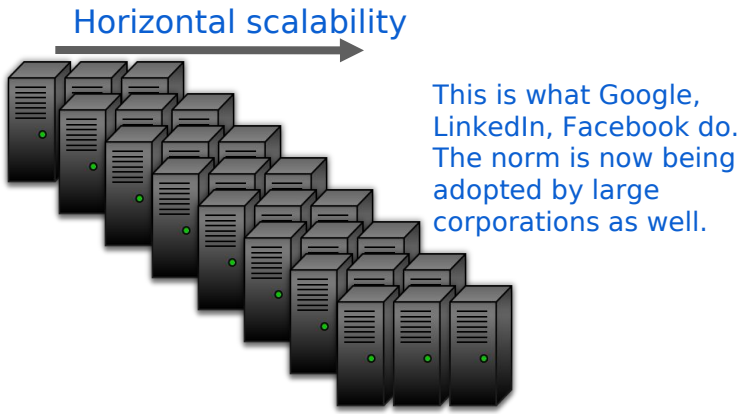
Previous generation of application architectures required larger and larger servers to handle capacity needs.

As usage would grow, previous hardware was swapped out whole for new larger hardware with "anticipated" new capacity.



Commodity vs Specialized hardware

vs



1. Large CAPEX
2. Wasted/Idle resource
3. Failure takes out a large chunk
4. Expensive redundancy model
5. One shoe fitting all model
6. Too much co-existence

1. Low CAPEX (rent on IaaS)
2. Maximum resource utilization
3. Failure takes out a small chunk
4. Inexpensive redundancy
5. Specific h/w for specific tasks
6. Very less to no co-existence

“Just in time” expansion; stay in tune with the load. No need to build ahead in time in anticipation

Designing applications for cloud

- Evaluate enterprise architecture impact
- Choice of the right service provider
- Defining the business process and integration touch points
- Security fabric cutting across all providers
- Think stateless, adapters, foundation
- Micro service approach

Challenges with Distributed Computing

- Heterogeneity
- Fault handling
- Consistency
- Global concurrency
- Upgrades and maintenance
- Local resources - file system
- Application sessions & transient data
- Clock synchronization

Data storage compliance

- Driven by compliance regulations such as
  - HIPAA in USA
  - Data Protection Directive in EU
- Data sensitivity - Financial / Medical
- Driven by a sense of "Complete control" as realized by on-premise
- Legal issues with data residency depending on where the data is physically stored off-premise
- Data archival policy impact

Cloud computing concerns & recommendations

- Progressive Architecture Development is where the overall architecture vision is broken into piece meal and developed over a period of time. The cloud approach should follow a similar approach.

- This is particularly important because the cloud applications space is still maturing barring a few select players. The following are the typical impediments perceived by organizations when it comes to cloud application implementations.

- Current enterprise apps can't be migrated conveniently
- Lock in with proprietary architecture
- Risks – Legal, regulatory, and business
- Difficulty of managing cloud applications
- Lack of clarity of SLA ownership
- Unclear ROI

The pinnacle of cloud computing

National Security Administration (NSA) of USA

- https://www.nsa.gov/ia/_files/factsheets/i43v_slick_sheets/slicksheet_cloudsecurityconsiderations_web.pdf
- https://www.nsa.gov/research/_files/publications/cloud_computing_overview.pdf
- Probably the pinnacle of private cloud is NSA http://www.networkworld.com/article/2687084/security0/exclusive-inside-the-nsa-s-private-cloud.html
- Openstack is used by NSA http://fedscoop.com/inside-nsas-data-protection-cloud-strategy
- http://www.androidauthority.com/nsa-creates-android-based-spy-phones-59330/

Closing thoughts

"*Cloud computing as a concept glues together several other, often independent, concepts in a complimentary way to open up new operating models and opportunities for various industries.*

Cloud Computing has evolved from, and extends, several concepts that have been around for some time, such as SaaS, utility Computing, Grid Computing, Virtualization, Real-Time Infrastructure, Web Platforms, and SOA."

Part 2

Architecture for the cloud

IaaS - key aspects

- Enterprise infrastructure
- Cloud hosting
- Virtual Data Centers (VDC)
- Scalability
- No investment in hardware
- Utility style costing
- Location independence
- Physical security of data center locations
- No single point of failure

PaaS - key aspects

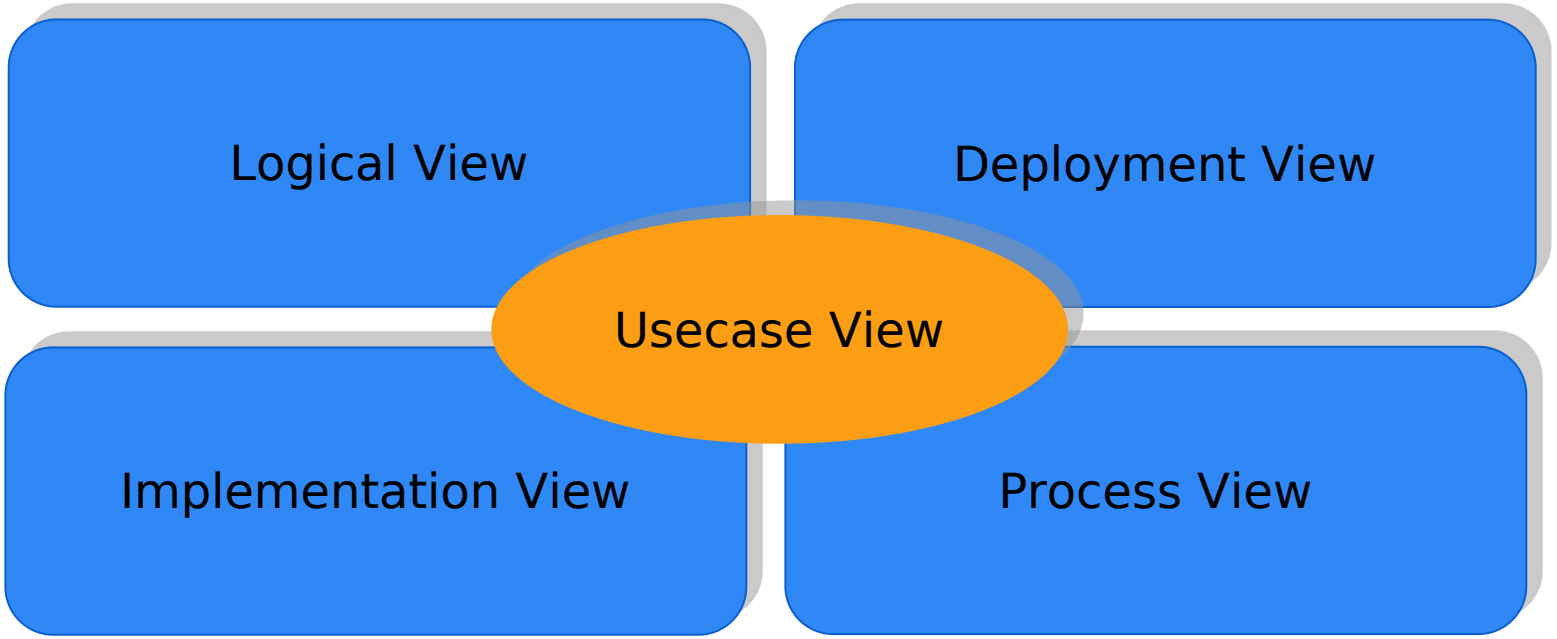
- Managed services, foundation, APIs
- Managed runtime, no need to start, stop
- Managed infrastructure, agnostic on the nature of hardware the application runs on, though the user can get some control of picking the type of machine
- Extensive monitoring & Dashboards, allows for monitoring and also identifying application bottlenecks and helps in debugging
- Auto scalability, can specify some limits to protect from malitious access
- Billing thresholds to keep a tab on the expenditure
- Alerts, always good to get informed "before" an outage
- Choice of programming language, customers have the ability to create teams as per organizational strengths

SaaS - key aspects

- Allows for hosting and management of applications to a 3rd party provider
- Usually a complete business process that can be used in isolation or in conjuntion with another business process to create composite macro business processes
- 3rd party provider is responsible for delivery SLA, L1 to L4 support etc
- Does not require any installation at the client location on any machine on any device
- Pricing can be per seat or fixed price depending on the provider
- The provider typically has a multitenant architecture for one instance to many client delivery models. This allows for economy of scale and reduces cost of usage
- These are usually online applications, however there can be offline applications too using contenporary browser capabilities

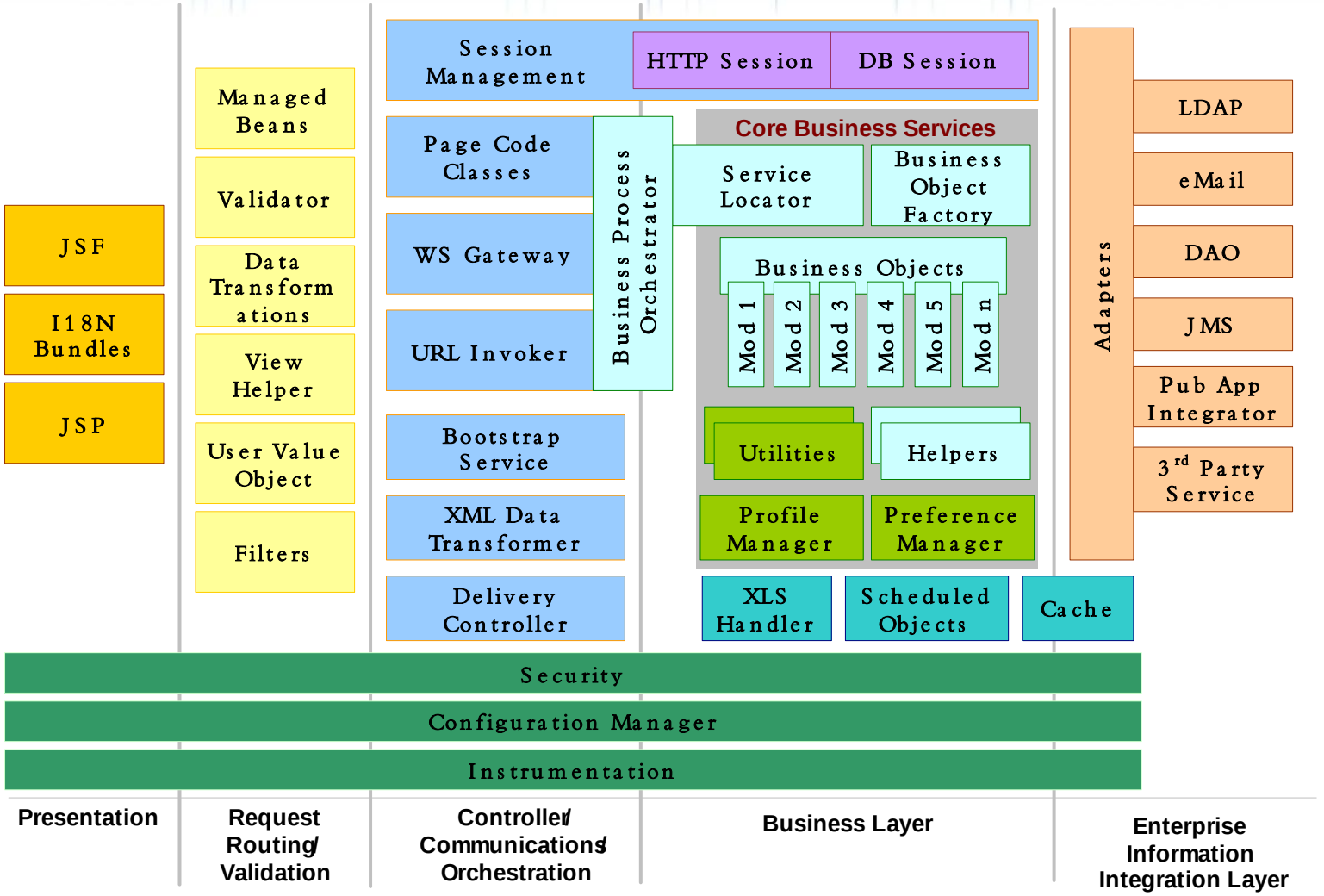
The basic formula for architecture

4 + 1 View of architecture

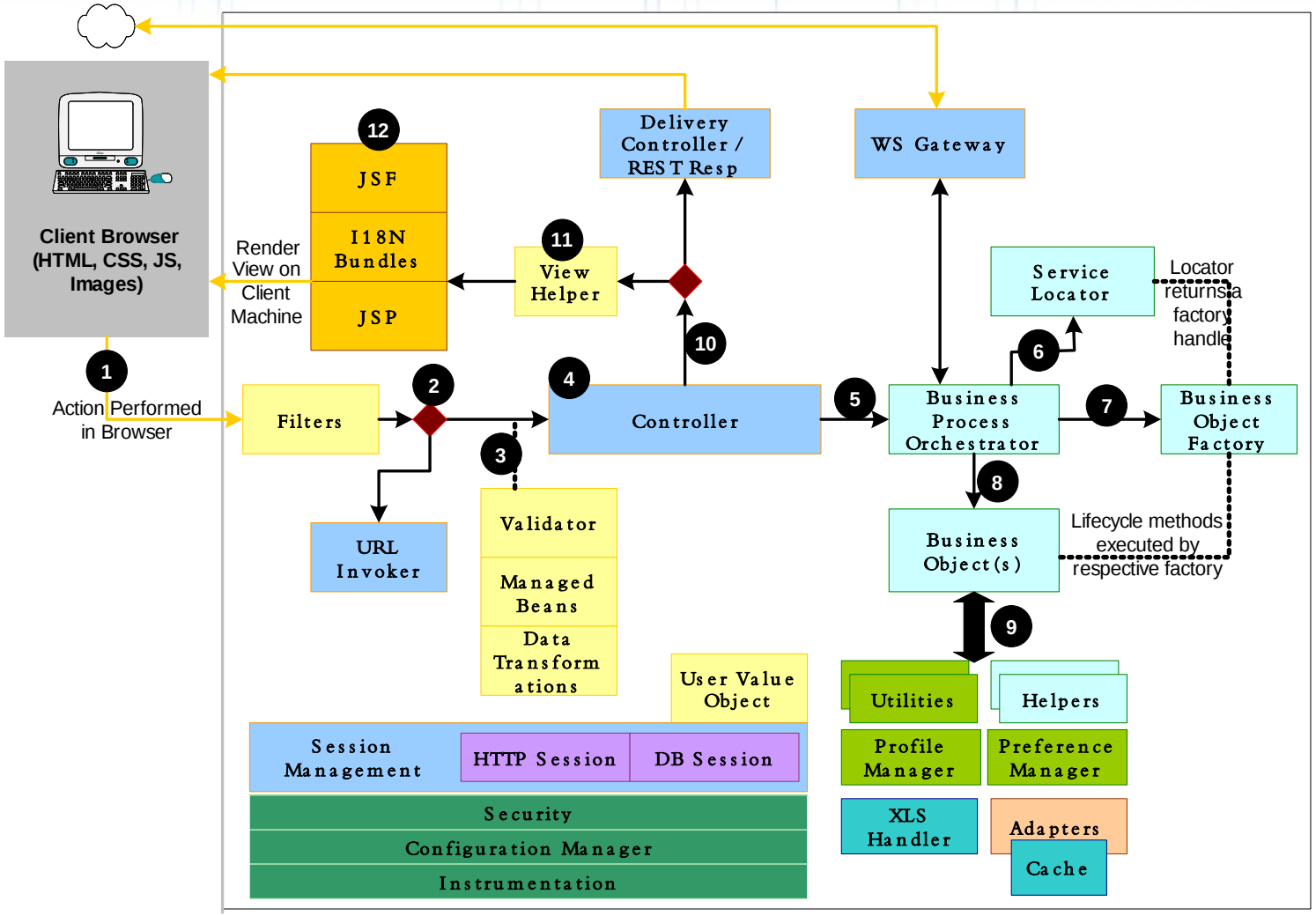


A basic blue print is advisable even in agile development process

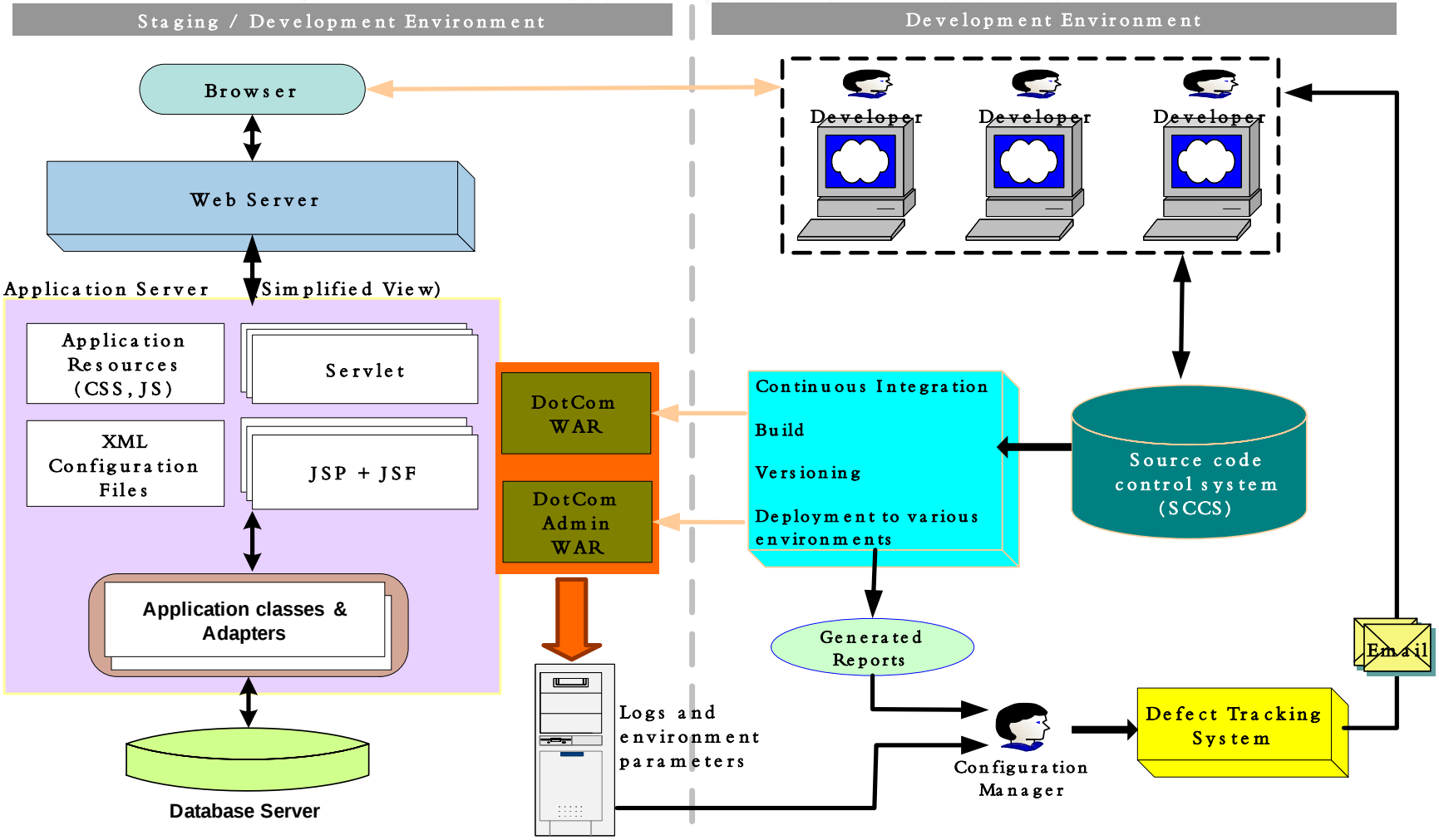
Logical view



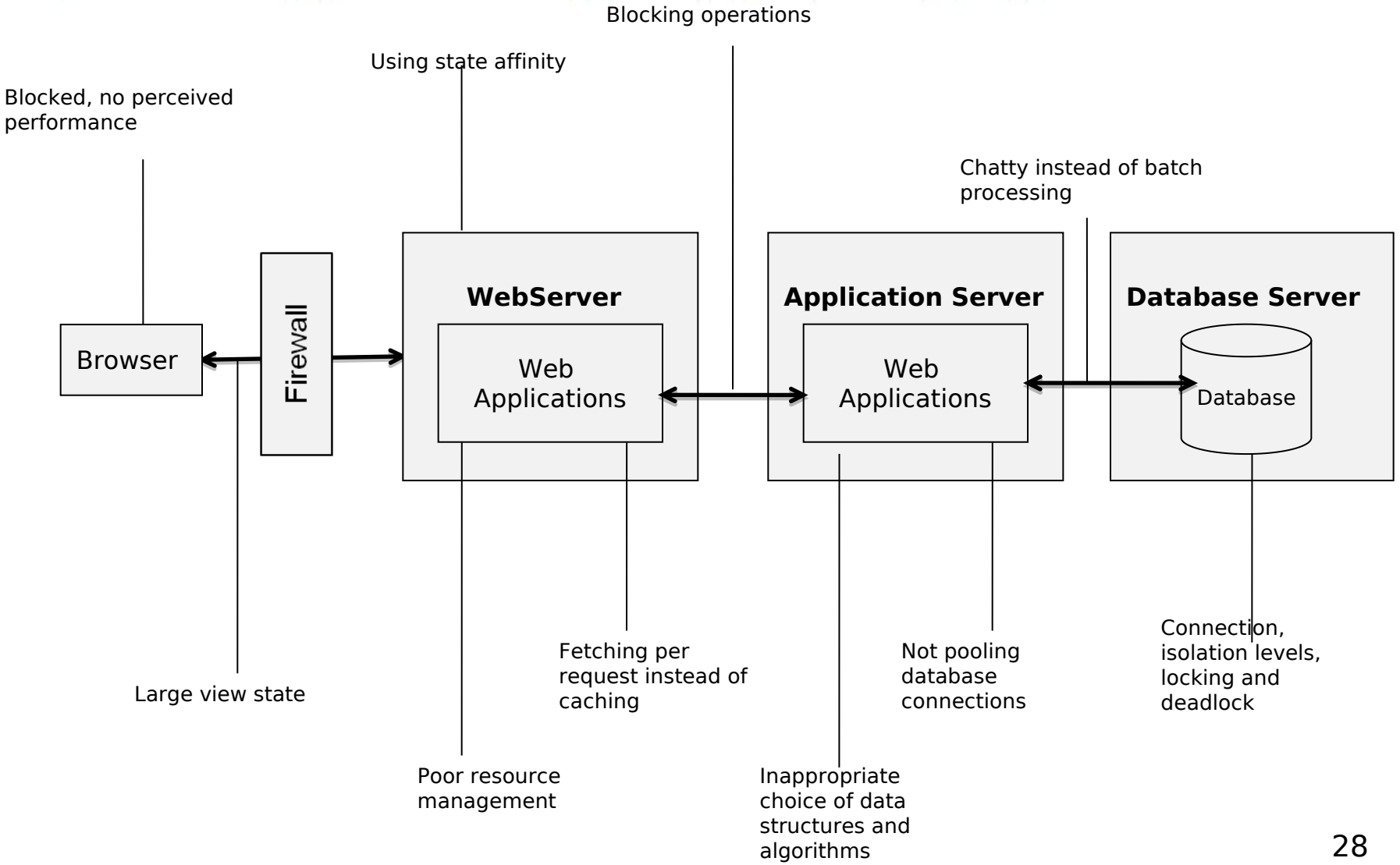
Process view



Implementation view



Web Application - Typical issue areas



Impact to enterprise architecture

Enterprise Architecture

Business Architecture

defines the business strategy, governance, organization, and key business processes.

Data Architecture

describes the structure of an organization's logical and physical data assets and data management resources.

Application Architecture

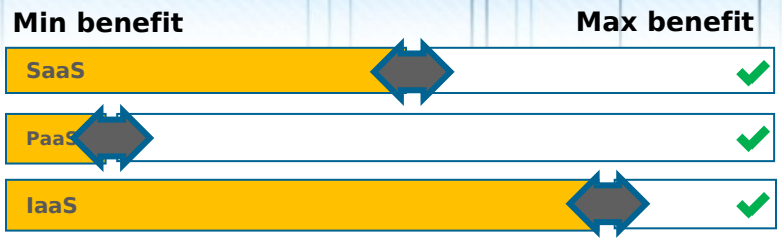
provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization.

Technology Architecture

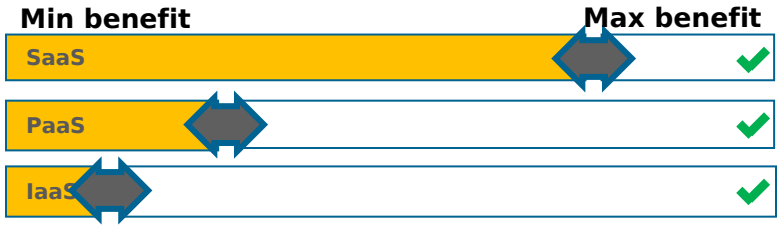
describes the logical software and hardware capabilities that are required to support the deployment of business, data, and application services. This includes IT infrastructure, middleware, networks, communications, processing, standards, etc

Business architecture

- Cost

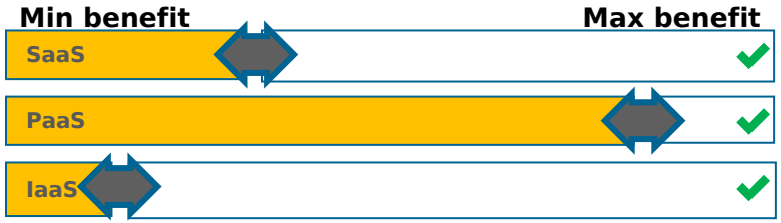


- Business Process Re-engineering



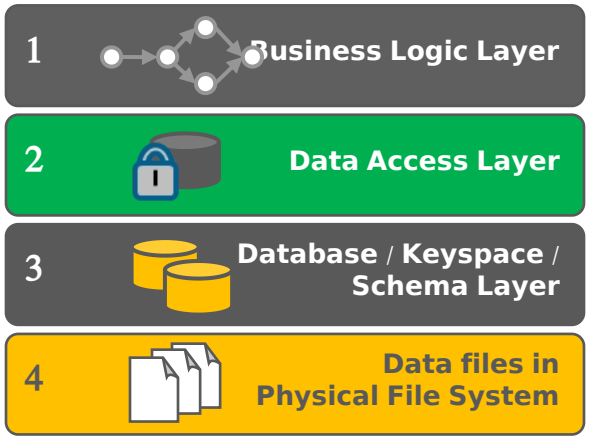
- Time to Market

- Functionality responsiveness

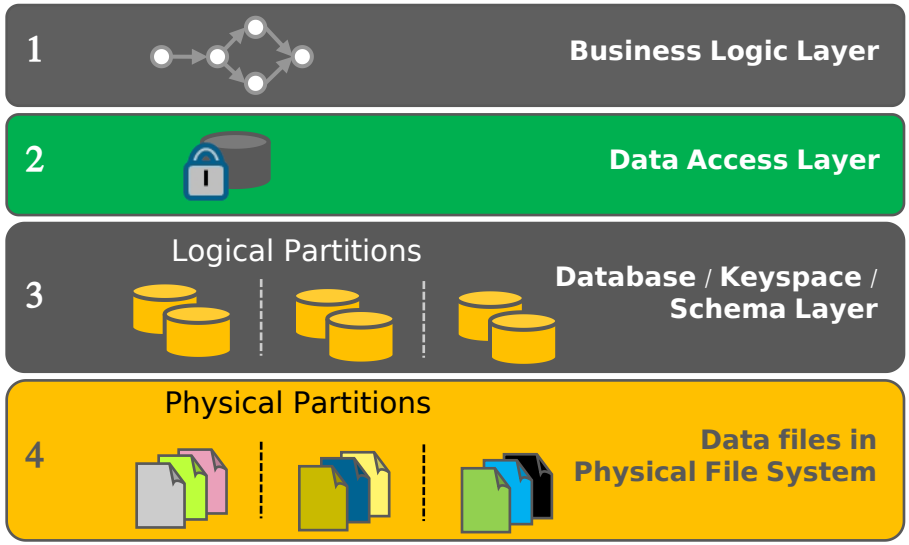


Data architecture

- Applications on the cloud can be classified as single tenant or multi tenant. While single tenant applications have a natural partition of data from one customer to another; multitenant applications require a deeper architectural consideration



Single Instance



Multi/ Tenant

- Applications that consume data from external sources need to consider appropriate security and customer partitioning handshake protocols.
- The application on the cloud needs to consider various country specific regulations that restrict data movements beyond the physical boundaries of that country. This is an important aspect to consider for applications which may have an impact to the deployment model or even the overall ROI.

Security/Service assurance - A few questions

- Within the same data center, there are some cases in which information belonging to more than one customer resides on the same computer. In such a case, will such different sets of information be appropriately isolated?
- Should we be concerned that operations in a data center might lead to information leakage or data corruption caused, for example, by one customer’s information being mistaken for another’s?
- Since the system platform of a Cloud services provider is shared by a wide variety of customer environments, couldn’t reliability be a problem? For example, if a malicious program such as a virus were to penetrate the service, can all the environments using that service be affected?
- When multiple Cloud services are used at the same time to perform work involving the linking of tasks between those services, can service reliability be assured?

Security architecture considerations

- Cloud computing architectural framework - enterprise security architecture
- Governance and enterprise risk management
- Legal and electronic discovery
- Compliance and audit Information life cycle management
- Portability and interoperability
  - Among multiple security providers
- Traditional security, business continuity, and disaster recovery
  - Particularly in case of life and shift model
- Data center operations
  - Physical security, personnel access policy
- Incident response, notification, and remediation
- Application security
- Encryption and key management
- Identity and access management

App architecture - current scenario

Application Client

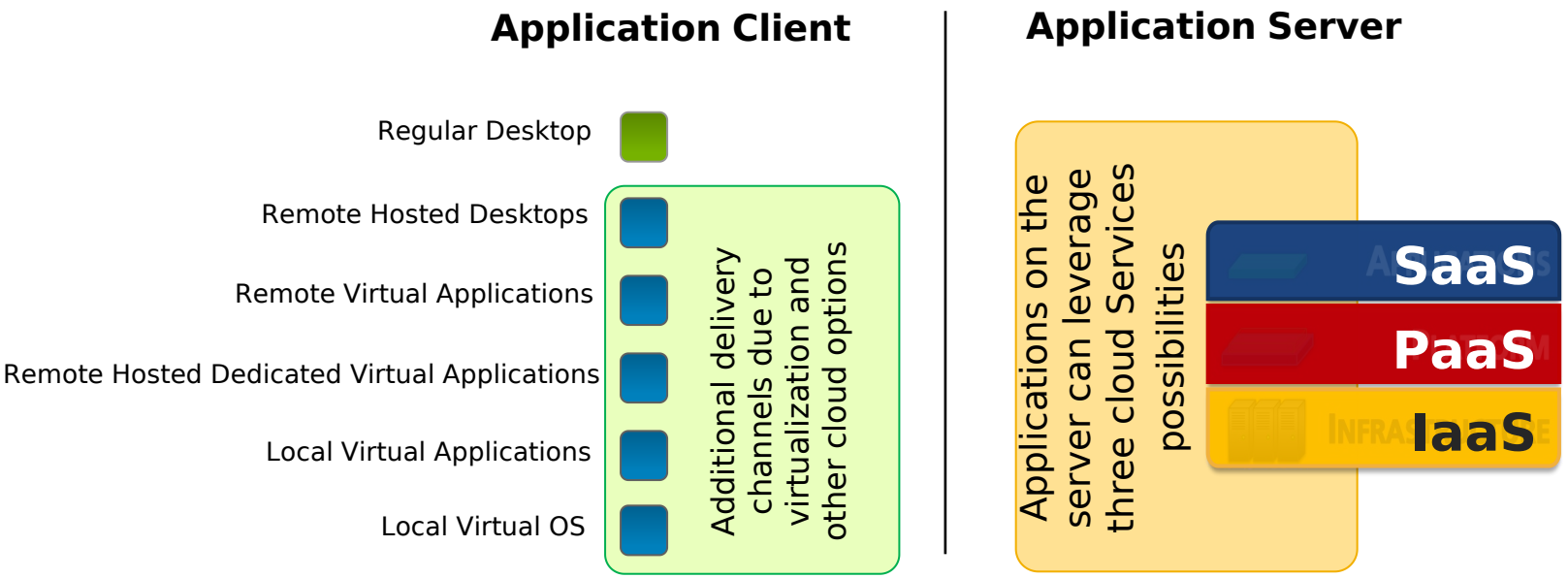
Browser based consumption

Other clients like tablet, mobile, etc

Application Server

Any technology landscape Including middleware, Database, content management and other Information management systems. Also includes systems
Like ERP, CRM and others.

App architecture - Cloud impact



The bigger impact is on the server side

Business applications and social applications

- Within an enterprise
- Has users in hundred of thousands
- Usually single security domain
- Data produced in TB
- Established biz product vendors
- Usage of databases
- Infrastructure with vertical scalability
- SDLC process
- Monolithic design
- Usually homogeneous environment (eg Microsoft, Java, SAP etc)

- Consumer oriented applications
- Users in millions
- Complex multiple security providers
- Data produced in hundred/thousand of TB
- Many open source products/libraries
- Usage of datastores
- Infrastructure designed for horizontal scalability
- Agile/iterative development process
- Componentized design
- Usually heterogeneous environment (many languages, libraries etc)

Social applications?

- Do you think only FB, LinkedIn etc are the social applications?
- Hint - social app is a "concept"

Choice of database or datastore

1. Database, Relational, strict models
2. Data in rows, pre-defined schema, sql supports join
3. Vertically scalable
4. Random access pattern support
5. Good fit for online transactional systems
6. Master slave model
7. Periodic data replication as read only copies in slave
8. Availability model includes a slight downtime in case of outages

1. Data store, distributed & Non relational, flexible schema
2. Data in key value pairs, no joins
3. Horizontally scalable
4. Designed for access patterns
5. Good for optimized read based system or high availability write systems
6. Can have masterless model
7. Seamless data replication
8. Masterless allows for no downtime

CAP Theorem

- ACID
  - **A**tomicity - Atomicity requires that each transaction be "all or nothing"
  - **C**onsistency - The consistency property ensures that any transaction will bring the database from one valid state to another
  - **I**solation - The isolation property ensures that the concurrent execution of transactions result in a system state that would be obtained if transactions were executed serially
  - **D**urability - Durability means that once a transaction has been committed, it will remain so under all circumstances

- What is it? Can I have all?
  - **C**onsistency - all nodes have the same data at all times
  - **A**vailability - Request must receive a response
  - **P**artition tolerance - Should run even if there is a part failure

- CAP leads to BASE theory
  - **B**asically **A**vailable, **S**oft state, **E**ventual consistency

Scalability of database and datastore

- Database being an ACID oriented storage has to do the following to accept data
  - Get a connection
  - Prepare a SQL and Bind the data
  - Check FK constraints
  - Update index (if any)
  - Null column checks (schema compliance)
  - Disk write (random disk I/O)
  - Store null for columns for which data is not there

- On the other hand a datastore only does the following
  - Get connection
  - Prepare and bind sql
  - Disk write (append only)
  - Update index (secondary index in certain cases only)
  - Store only the columns for which the value is provided

Types of NoSQL

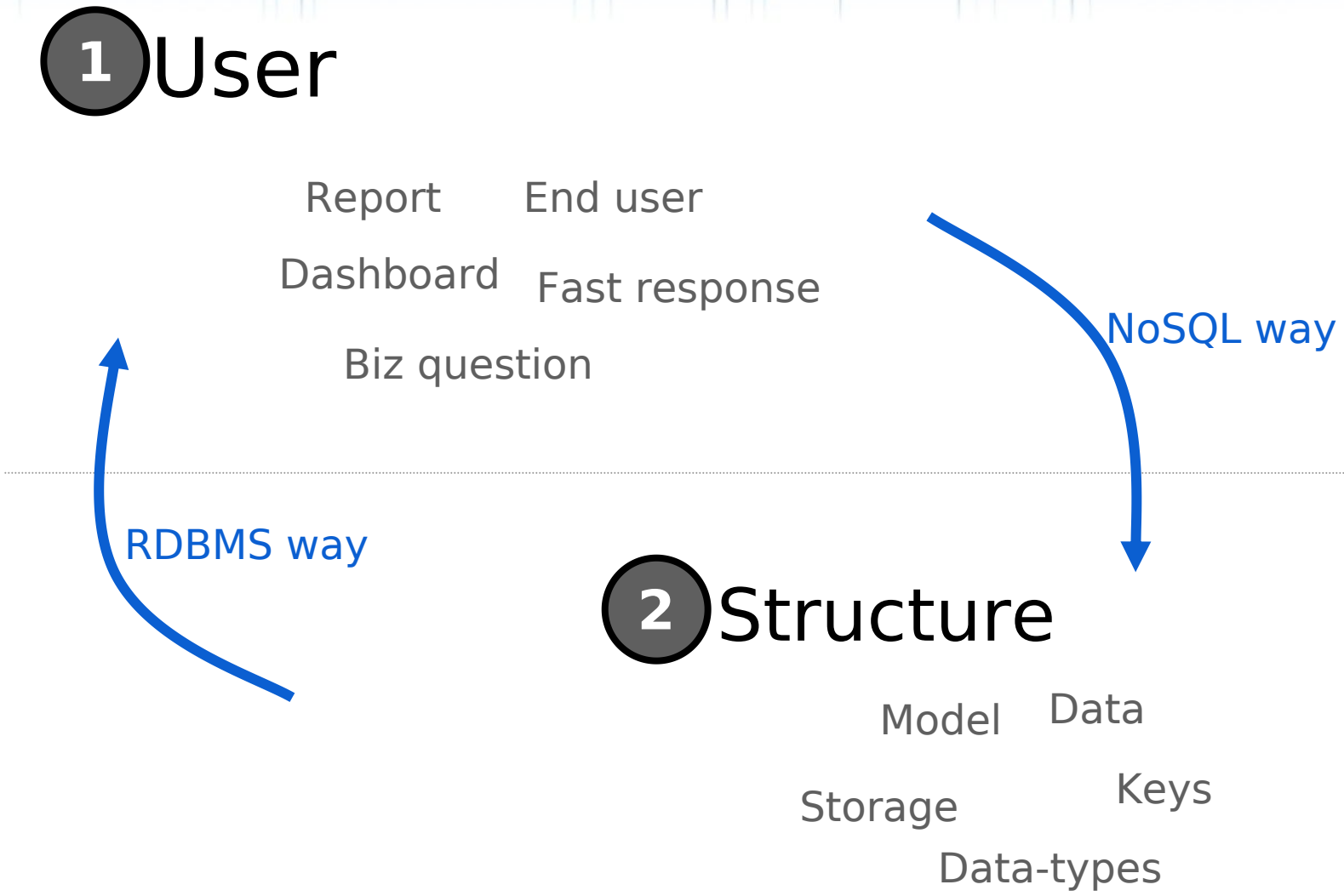
- **Wide Row Store** - Also known as wide-column stores, these databases store data in rows and users are able to perform some query operations via column-based access. A wide-row store offers very high performance and a highly scalable architecture. Examples include: Cassandra, HBase, and Google BigTable.

- **Key/Value** - These NoSQL databases are some of the least complex as all of the data within consists of an indexed key and a value. Examples include Amazon DynamoDB, Riak, and Oracle NoSQL database

- **Document** - Expands on the basic idea of key-value stores where "documents" are more complex, in that they contain data and each document is assigned a unique key, which is used to retrieve the document. These are designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. Examples include MongoDB and CouchDB

- **Graph** - Designed for data whose relationships are well represented as a graph structure and has elements that are interconnected; with an undetermined number of relationships between them. Examples include: Neo4J and TitanDB

Approach to modeling



Denormalization

- Unlike RDBMS you cannot use any foreign key relationships
- FK and joins are not supported anyway!
- Embed details, it will cause duplication but that is alright
- Helps in getting the complete data in a single read
- More efficient, less random I/O
- May have to create more than one denormalized view to serve different queries

Data replication

- Typical NoSQL and DFS based engines will provide replication across nodes
- Nodes can be in diferent
  - Racks
  - Zones
  - Regions

Distributed File System (DFS)

- Database striping, table partitioning
  - how do these work?

Here are the most well known options

- GFS - Google File System
- IBM GPFS -General Parallel File System
- Hadoop DFS

Map-reduce fundamentals

- Debate - What is the reason that allows for the stored procedures to execute faster on high data volume than developing an equivalent application logic?

- Data comes to the compute OR Compute goes to data?

- What is a mapper?
- What is a reducer?
- How do they work together?

A/B testing of applications

- Incrementally rollout features
  - Region based (IP)
  - User type based (Cookie)
- Data compatability
  - Need for flexible schema
  - Data generated by new features may be different than the older version
- Overall application versioning
- Need to quick rollback in case of trouble and trouble does happen!

How can you - the architect contribute in reducing costs?

- What factors add to cost?
  - CPU
  - Memory
  - Storage
  - Network
  - 3rd party API

Stateless design

- What is stateless design?
- Is too much stateless good?
- What's the optimum balance?
  - Split between framework components and business components
  - Biz components are BO and Helpers
  - Data access layer
- File system
  - Using local file system for storage
- Http Session
  - In distributed systems where to create the session?
  - Session stickiness and impact

Thinking adapters

- Applications in the cloud can potentially leverage services/APIs from multiple providers
- Each provider will have their own
  - release cycles
  - API service versioning
  - feature enhancements
  - payload data structure upgrades
- How do we protect our application from such changes?
- Abstraction techniques - data format transformations
- Caching
- API calling techniques - Web services or REST?

Microservices

- Monolythic server code
  - It's great because anyone can do anything and may help in being "polyglot"
  - In practice, however, there’s a balancing act where developers then need to understand how everything actually works in order to make change
  - Generic on shoe fit all approach
  - Overall testing can become pretty intense as the codebase grows
- You typically do not have to change everything at the same time!
- Think microservices
  - Also allows a way to create specialized teams that understands one/or more service very deeply
  - Specific service signature is defined and agreed across teams
  - Each service gets what it truly needs - language, infrastructure, scale etc
  - Its like a combination of multiple small code bases each corresponding to a service

Business service layer pattern

- Can be easily powered by Microservices
- Access
  - Consider using REST / URL APIs to access business features from outside
  - Within an application consider using direct API calls and object exchange
- Service versioning is critical
- Data exchange
  - JSON (compact)
  - XML (verbose)
- This allows for the capability of "Integration ready" where other applications can leverage various features
  - Other applications can be inside OR outside the enterprise
  - Since the services are atomic the usage can be monitored very closely
  - Usage monitoring can lead to very interesting and flexible pricing models in case these are exposed to outside the enterprise
  - Example http://api.acme.com/sc/customer/update/123

Debate - Mobile apps!

- "Download our mobile app and your next ride will be Rs 100 only"
- "Buy using our mobile app and you will get an additional 10% off"
- "This discount coupon is available on purchases from our mobile app only"

Q: What's going on? What are your thoughts?

Assemble UI on devices

- There are two models of rendering the user interface
  - Assemble on server side
  - Assemble on user's device
- Server side
  - Fully created HTML is served
  - Server needs to be aware of user's device
  - Server utilization is more
  - Not possible to do responsive design
  - Page refresh may be time consuming
- User's device
  - Server serves the page skeleton
  - UI calls various services and builds the UI depending on the device characteristics
  - Server utilization is low
  - Responsive design
  - Partial refresh creates engaging UI and gives a sense of speed
- Assembly on user's device should be the choice because
  - Reduces OPEX
  - Leverage the capabilities of very powerful handheld devices

Native device applications or ...

- There are two kinds of applications that can run on a mobile device (iPads included)
  - Native applications
  - Browser based applications
- Let's look at the Native application properties ...

- Is a different app for each type of device - iOS, Android, WinOS etc
- Since these are installed on the device, they have access to the device a lot more (phone book, camera, local disk etc)
- Requires specialized development skills
- Each app is like a project that needs to exist and get development by a team
- Hard to synchronize app features across the board
- Push updates are required to ensure latest version is available
- Can work in offline mode if designed
- Degree of customization and application features are limited to whatever is supported by the device OS

... browser based?

- Let's look at the browser based app properties ...

- As long as there is a compatible browser with a JS engine, the site will run
- No installation required but may not have access to device features (address book etc)
- It is a web applications and the skills are confined to web UI development
- There is one web app that renders on a multitude of devices
- No feature synchronization issues
- If the site is updated then all devices will serve the latest
- Cannot work offline very effectively (though there are offline embedded DB in JS)
- Degree of customization is limited to the browser and JS engine

- Regardless of the type chosen the server side services should remain the same

Content

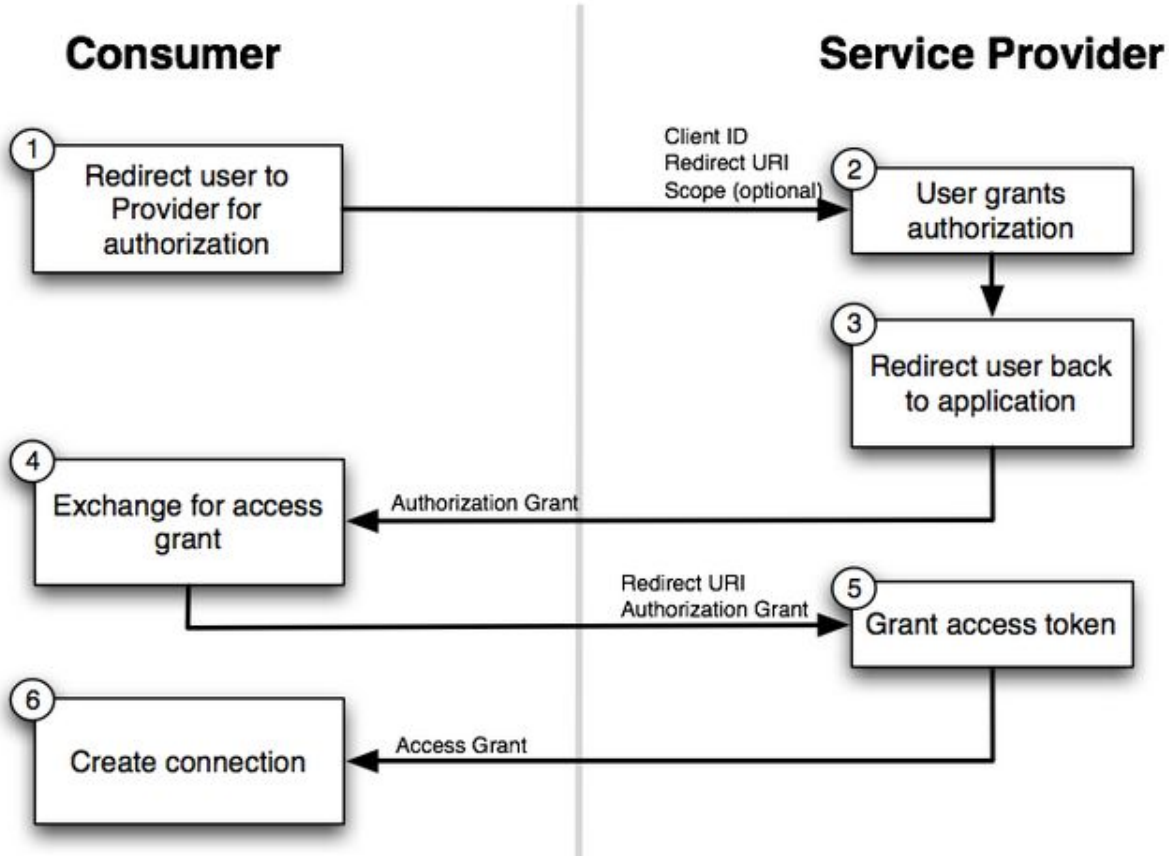
- Content can be of two kinds
  - Static
  - Dynamic
- Static
  - Typical examples are media, images, audio etc
  - Other include JS, CSS, HTML etc
- Dynamic
  - Generated by server side programs such as JSP, ASPX, JS, PHP etc
- Evaluate the split ratio static:dynamic
- Have different serving strategies for both
- Attempt should be made not to bundle too many static content in the web application
- Consider using CDN for static content
- Enable cache parameters so that the calling device can store them locally thereby reducing server load
- Have a refresh policy in case of content update (eg file name update) - this may have an application impact

Leverage cloud APIs and services

- The cloud offers a plethora of services and APIs
- Business functions
  - User voice
  - Survey
  - Checkout / payment
  - Email dispatch (eg SendGrid)
- Social features
  - Social provider login
  - Likes / comments
- Reduces time to develop and market
- Creates a dependency on 3rd party
- How should the application behave if the service is unavailable?
- How about data?
  - Need to write certain programs to extract data
  - Leverage 3rd party provider dashboards
  - What if the data size is in TB?
- Creates de-centralized administration
  - How to give CxO an unified view of operations?

Security fabric for the architecture with multiple providers

- The usual model in the cloud is oAuth
- The current version is 2.0
- Here is a typical flow to integrate with Facebook authentication

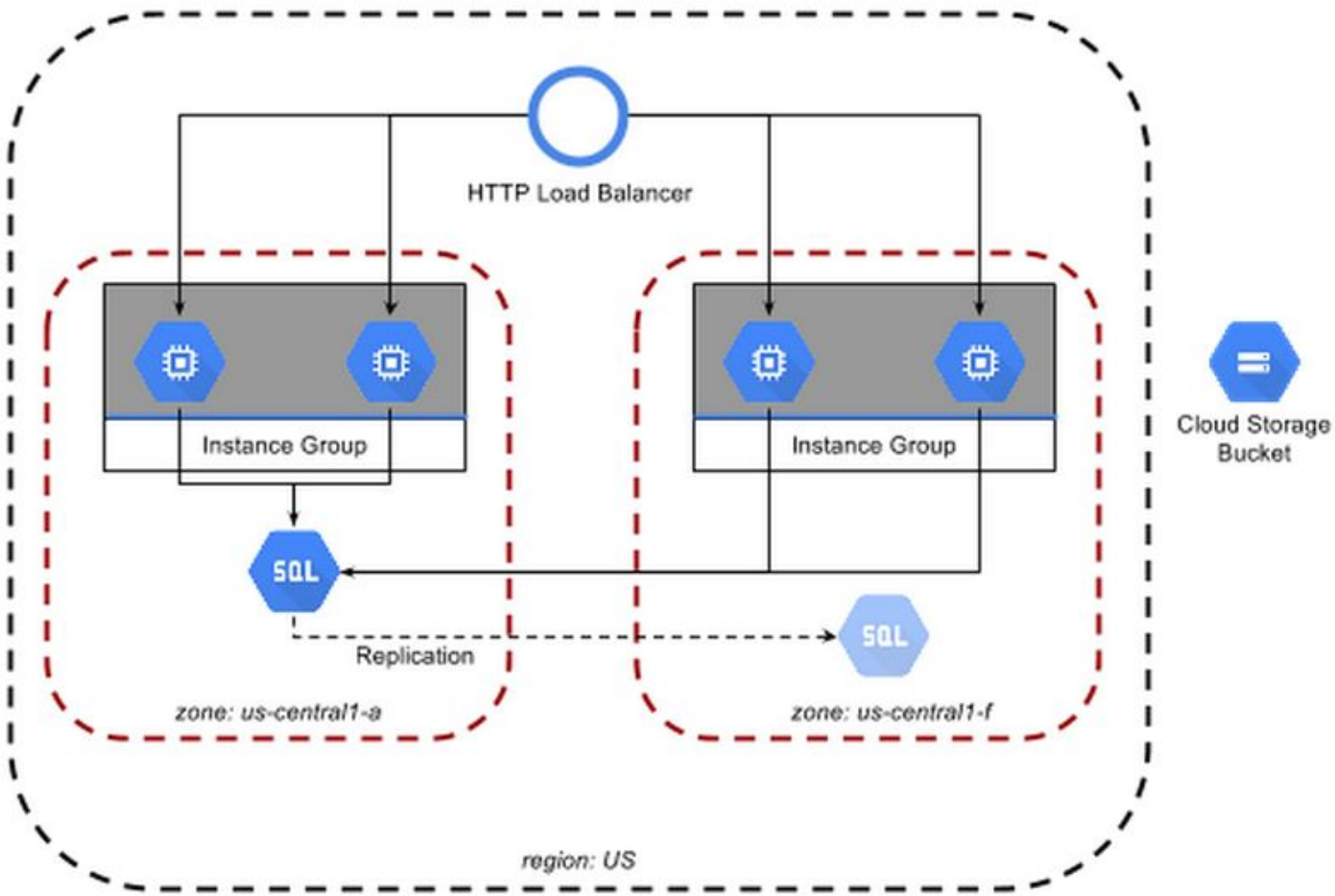


Contemporary architectural styles

- Building scalable resilient applications
- Define application zones
- Think foundation and common service layer
- Lambda architecture for analytics
- Some other techniques such as vector clocks, mutex models needed in distributed computing

Scalable & resilient apps approach

https://cloud.google.com/solutions/scalable-and-resilient-apps



Define application zones

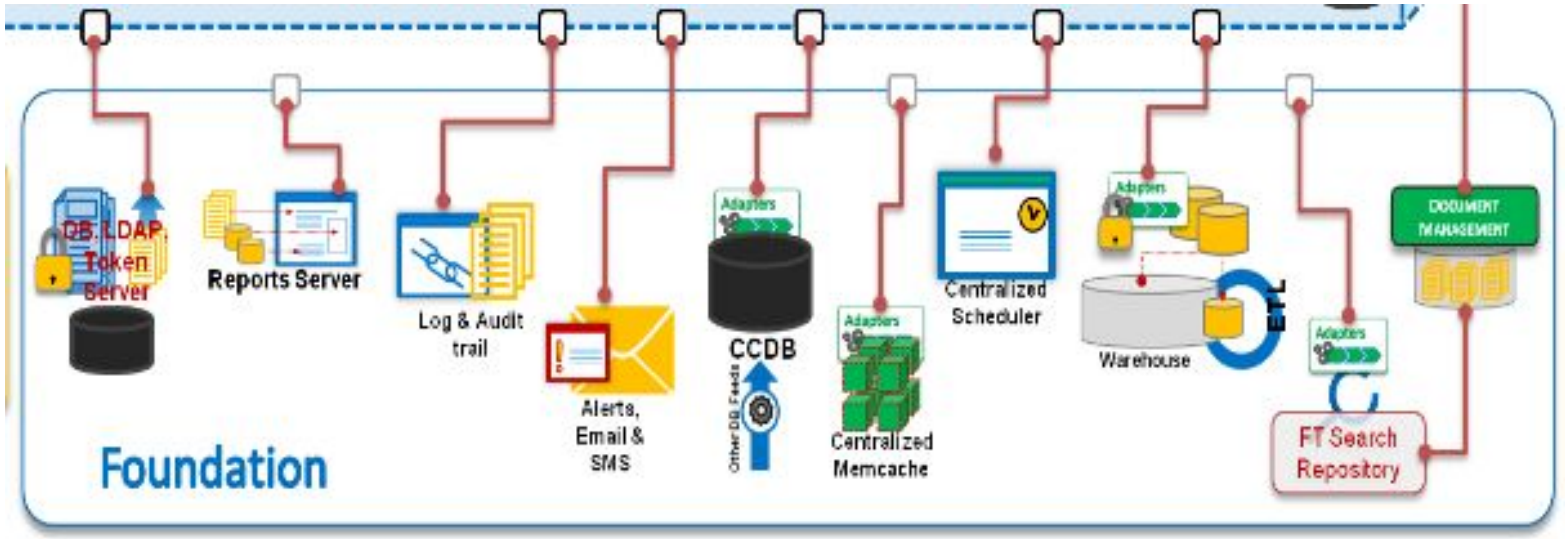
- Various components of the application tiers require different attention
  - Choice of programming language
  - Choice of hardware
  - Specialized libraries
  - Scaling requirements
  - Backup and recovery mechanisms
  - Monitoring requirements

- Can "One shoe fit all?" How do we ensure "Availability of customer facing services?"

- Application zones are an effective way to isolate different areas of the application and provide an environment that is optimized and tuned for that function
  - Distributed data layer
  - Microservice layer
  - Analytics layer
  - Online web application layer

Thinking "Foundation"

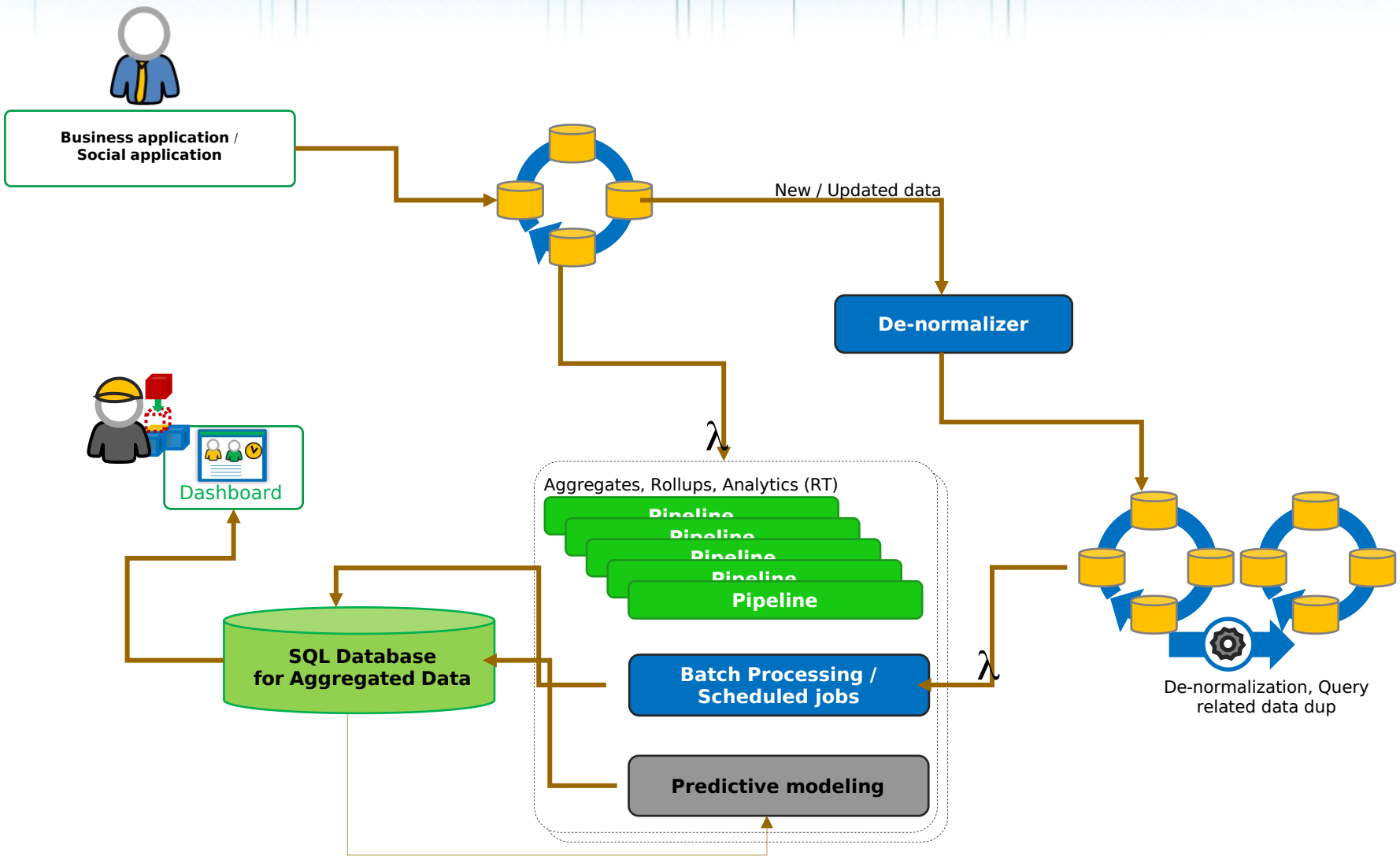
- When setting up an application landscape consider having a fundation containing common components that can be installed as a service
- Reduces redundancy
- Optimizes cost of operations
- Eases upgrades and release management



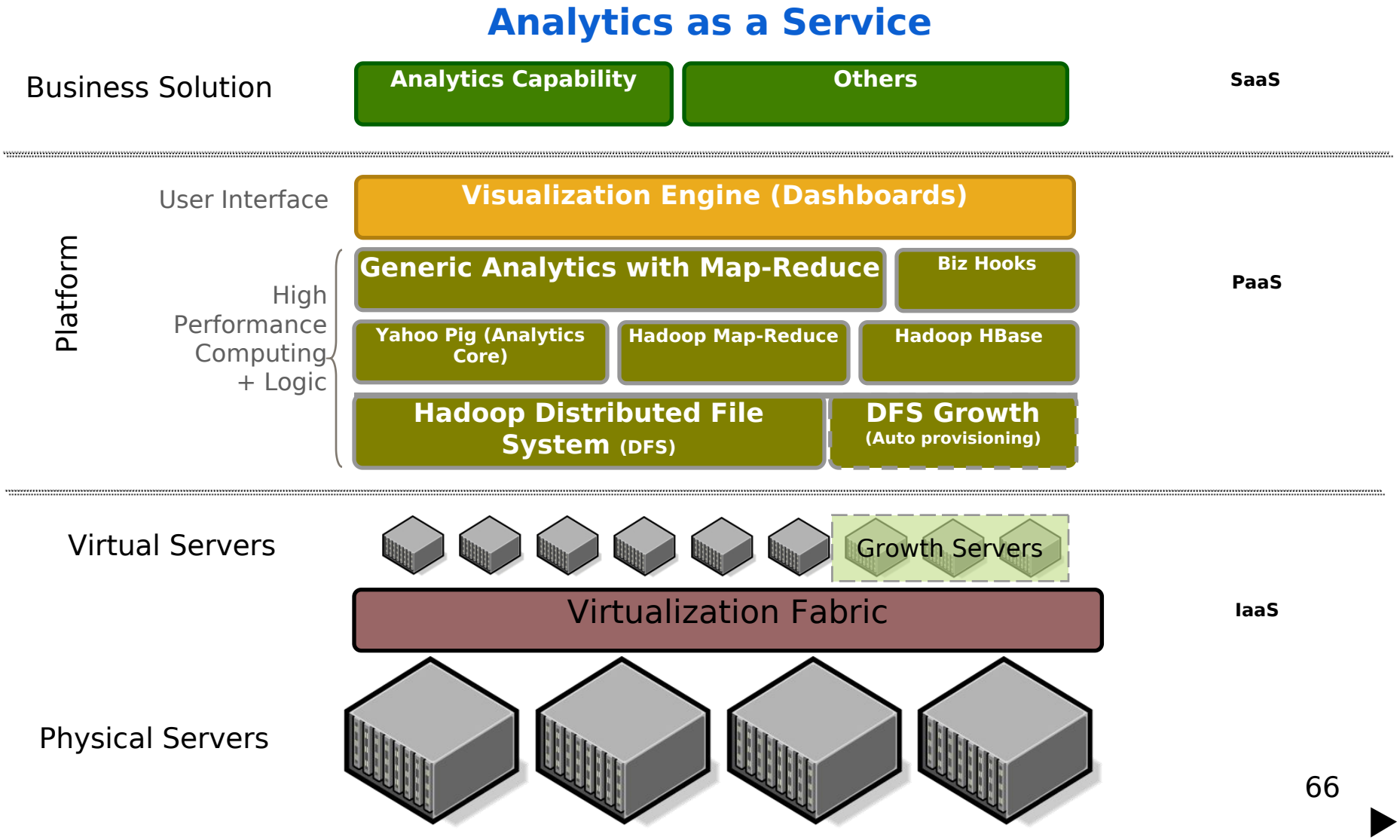
Global counters & locks

- Let's discuss some real world challanges in distributed computing
  - Vector clocks
  - Memcache Mutex model

Lambda architecture example



Example of an analytics stack



Closing thoughts

- Modularize
- Microservices
- User interface assembly on devices
- Create application zones
- It is ok to have a mix of technologies
- No joins is not necessarily bad
- Denormalization is fine
- Being frugal is awesome!

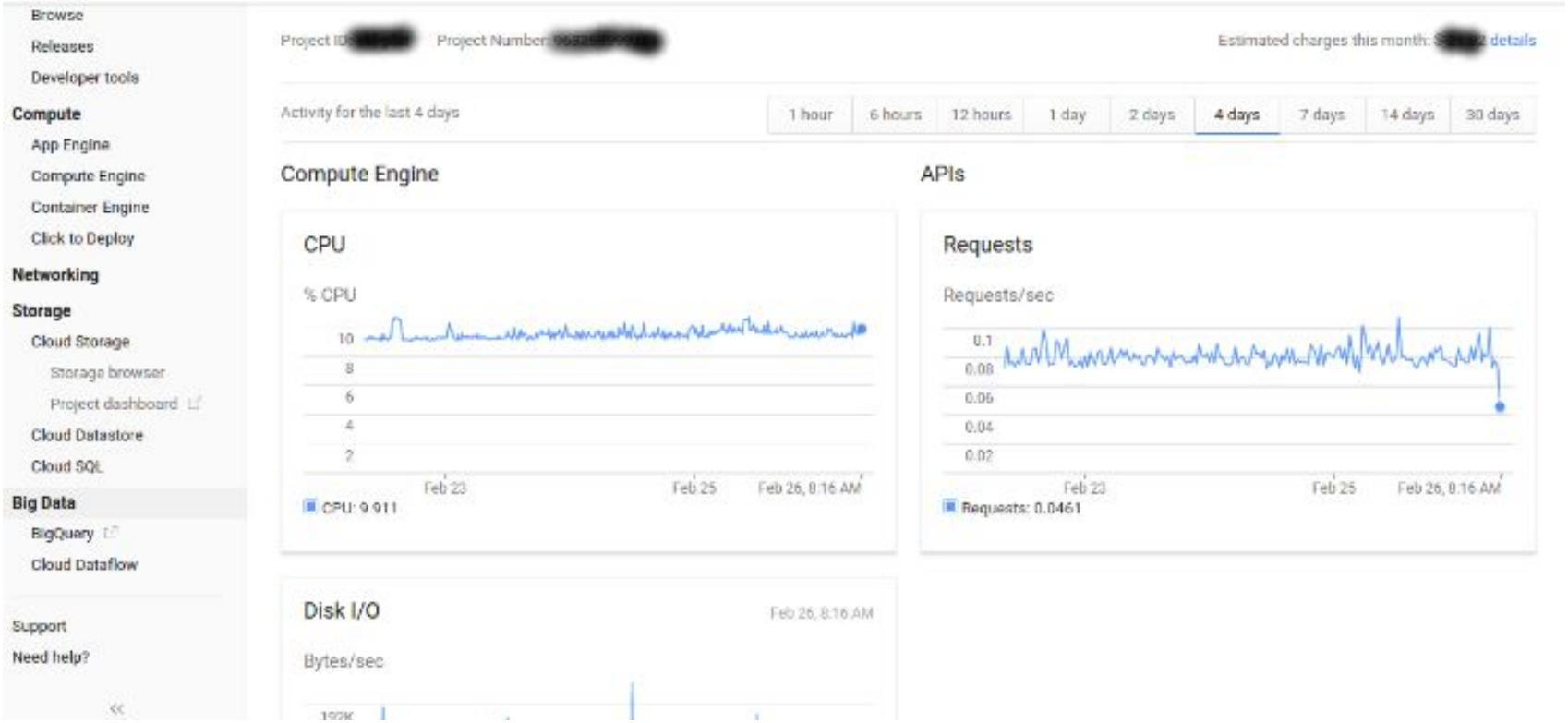
Part 3

Infrastructure as a service

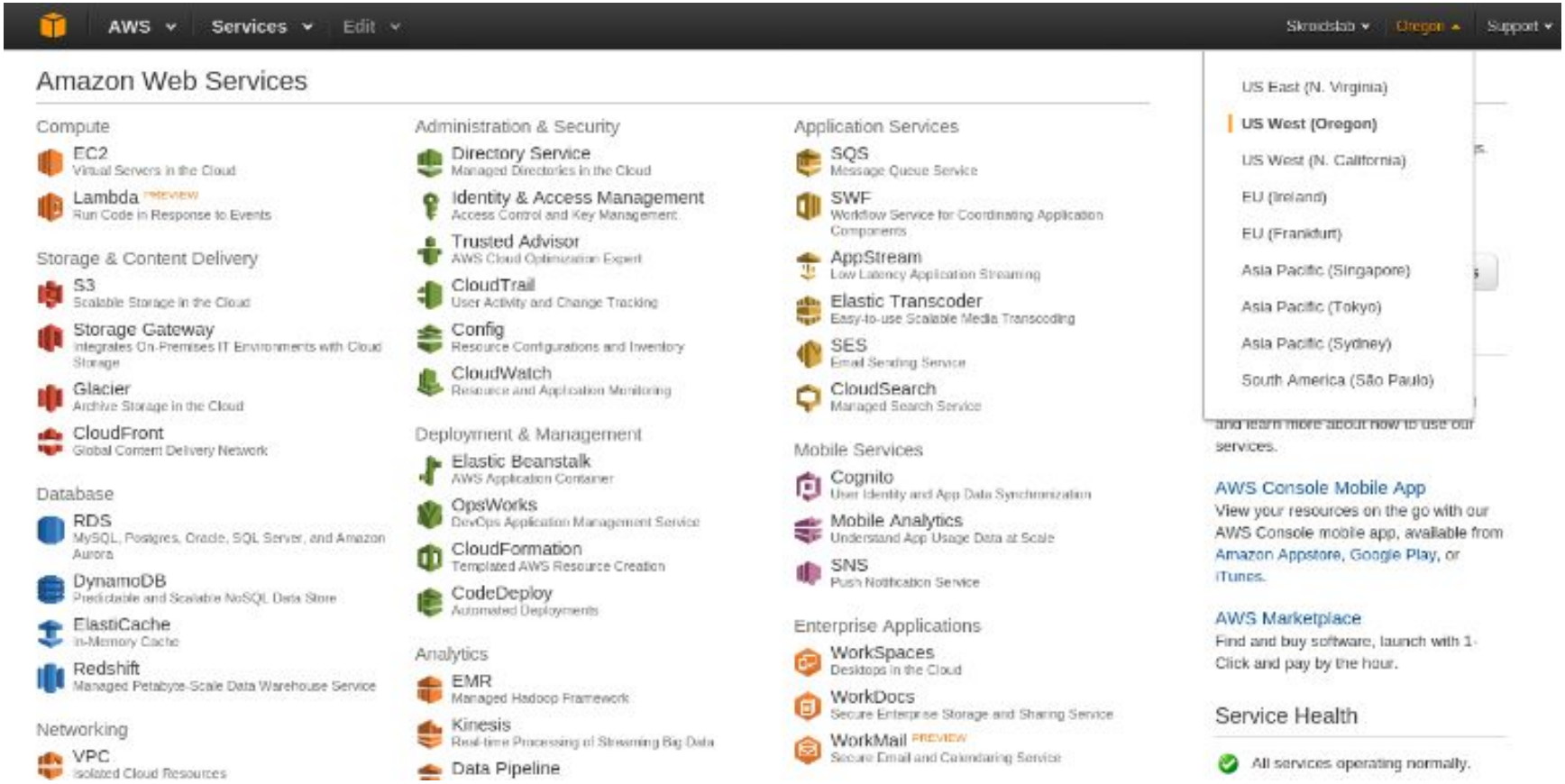
Quick look at the Facebook datacenter

- https://www.youtube.com/watch?v=Y8Rgje94iI0

Demo of GCE dashboard



Demo of AWS


AWS
Services
Edit
Skreidslab
Oregon
Support
Amazon Web Services
Compute
EC2
Virtual Servers in the Cloud
Lambda PREVIEW
Run Code in Response to Events
Storage & Content Delivery
S3
Scalable Storage in the Cloud
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage
Glacier
Archive Storage in the Cloud
CloudFront
Global Content Delivery Network
Database
RDS
MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora
DynamoDB
Predictable and Scalable NoSQL Data Store
ElastiCache
In-Memory Cache
Redshift
Managed Petabyte-Scale Data Warehouse Service
Networking
VPC
Isolated Cloud Resources
Administration & Security
Directory Service
Managed Directories in the Cloud
Identity & Access Management
Access Control and Key Management
Trusted Advisor
AWS Cloud Optimization Expert
CloudTrail
User Activity and Change Tracking
Config
Resource Configurations and Inventory
CloudWatch
Resource and Application Monitoring
Deployment & Management
Elastic Beanstalk
AWS Application Container
OpsWorks
DevOps Application Management Service
CloudFormation
Templated AWS Resource Creation
CodeDeploy
Automated Deployments
Analytics
EMR
Managed Hadoop Framework
Kinesis
Real-time Processing of Streaming Big Data
Data Pipeline
Application Services
SQS
Message Queue Service
SWF
Workflow Service for Coordinating Application Components
AppStream
Low Latency Application Streaming
Elastic Transcoder
Easy-to-use Scalable Media Transcoding
SES
Email Sending Service
CloudSearch
Managed Search Service
Mobile Services
Cognito
User Identity and App Data Synchronization
Mobile Analytics
Understand App Usage Data at Scale
SNS
Push Notification Service
Enterprise Applications
WorkSpaces
Desktops in the Cloud
WorkDocs
Secure Enterprise Storage and Sharing Service
WorkMail PREVIEW
Secure Email and Calendaring Service
US East (N. Virginia)
US West (Oregon)
US West (N. California)
EU (Ireland)
EU (Frankfurt)
Asia Pacific (Singapore)
Asia Pacific (Tokyo)
Asia Pacific (Sydney)
South America (São Paulo)
and learn more about how to use our services.
AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.
AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.
Service Health
All services operating normally.

Install and start HTTP on AWS/GCE

On AWS

Http server

- sudo yum install httpd
- sudo /etc/init.d/httpd start

Tomcat

- sudo yum install tomcat7 tomcat7-webapps
- sudo service tomcat7 start

On GCE

- sudo apt-get update
- sudo apt-get install apache2

Closing thoughts

- Infrastructure on rent
- Allows you to focus on applications
- Ground up management of apps and other containers
- Leverage expertise
- Plug and play services as demand arises
- Pick the right infrastructure as applicable for the use case rather than generic one shoe fit all model

Part 4

Platform as a service with GAE

Objectives

- We will use Java
- We will setup Eclipse
- Install the plugin
- Get the SDK
- Create a model in Google BigTable
- Build an application that does authentication
- A sample page
- Logout page

- Along the way we will also explore GAE in detail

Platform / foundation services

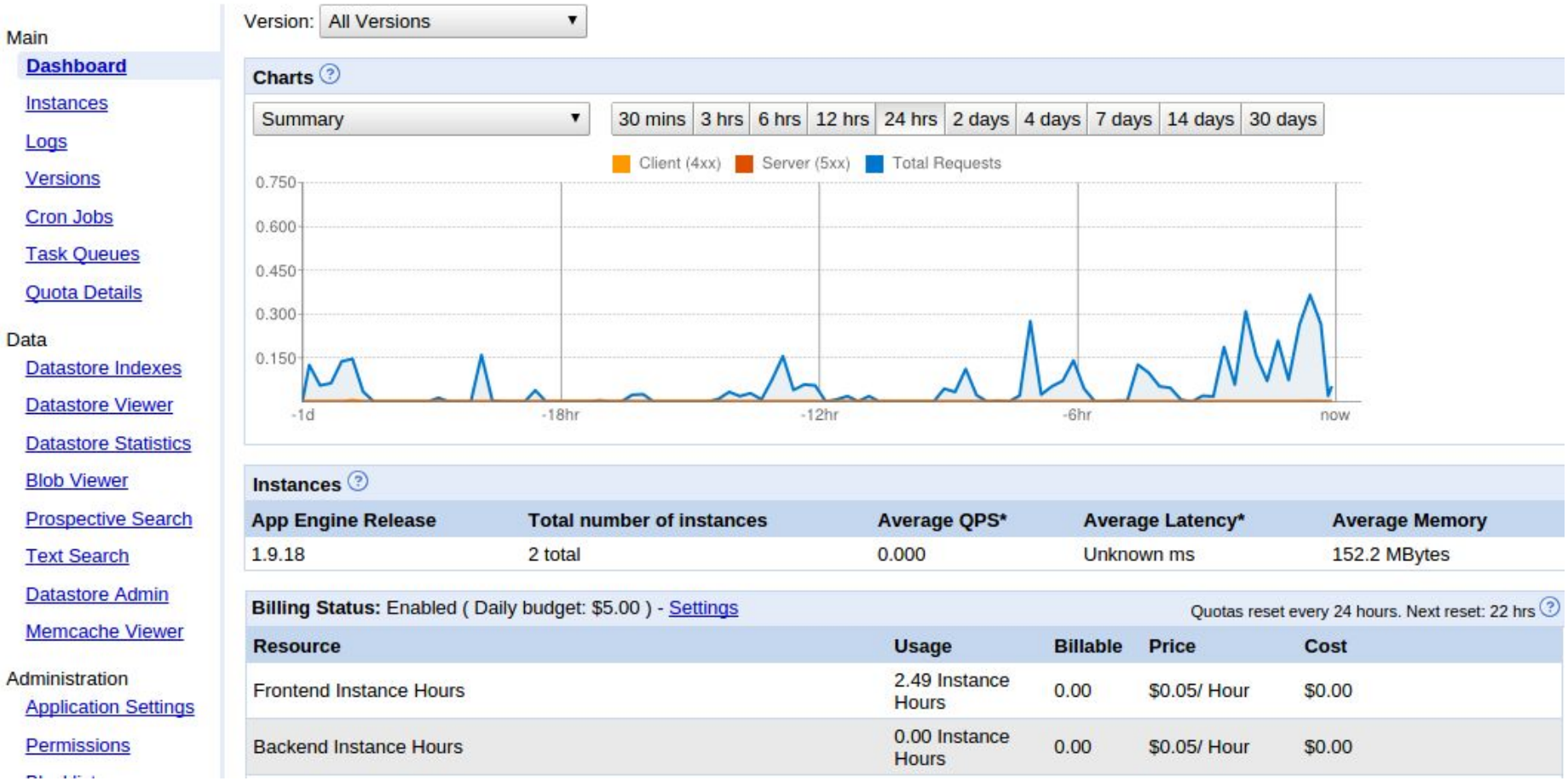
- Runtime
- Data store services

- Users API and Oauth security APIs
- Multitenancy (Global namespace for each tenant)
- Mail service
- Memcache
- Image manipulation
- Task queues and Crons (look at the corresponding XML files)
- URL fetch
- Prospective search
- Search
- XMPP for chat applications
- Loggin service
- Usage monitoring and cost budgeting dashboard
- Quota management
- BigQuery

Data store services

- Datastore (not Database)
  - Google BigTable (same infrastructure used by Google search)
  - How is it different from a traditional database? How does it store data?
  - GQL
  - No joins, no referential integrity, constant retrieval time from millions of records
  - Faster access tips – get by ID
  - Index management and areas to be careful of (exploding index, vacuum index). Take a closer look at data-store.xml contents. Turning the default ASC/DESC index off.
- BigTable management – viewer, statistics, Datastore admin
- Blobstore
  - Storing binary large objects (typically image and video files)
  - How this mechanism works (container intercepts, http payload is updated, servlet gets the request)
- Google Cloud SQL (Database)
  - This is managed MySQL instances with some limitations
- Google Cloud Storage
  - Only place where file API is available (as part of the blobstore API)

Demo of GAE dashboard

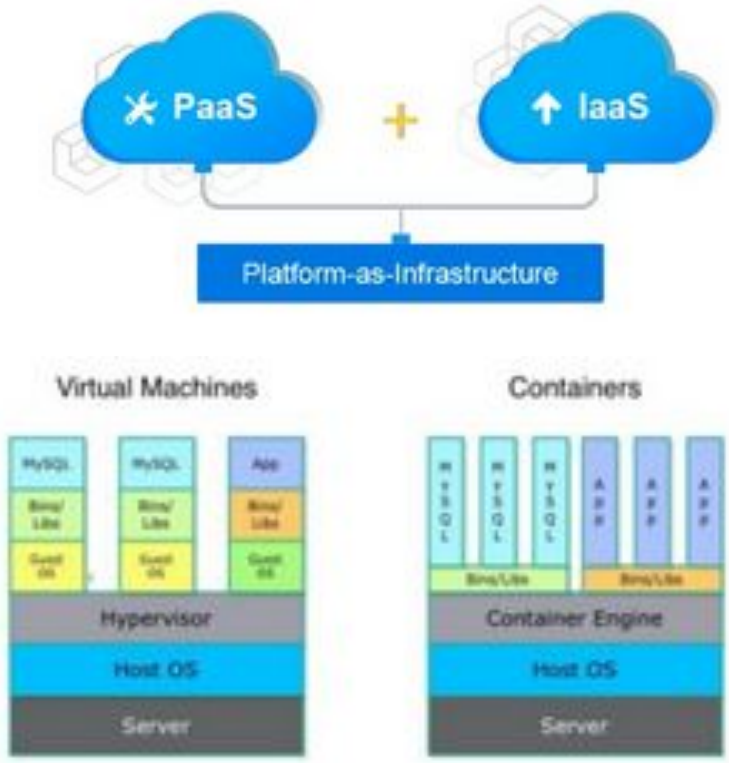


Let's build

- Setup
- Approach and design for GAE
- Code
- Deploy
- Run!

Jelastic = IaaS + PaaS

- Java specific PaaS
- Looks to bring IaaS and PaaS together as an integrated offering
- The virtualization is at the container level
- This VM strategy creates a more dense cluster, further increases utilization



Jelastic = IaaS + PaaS

- Polyglot PaaS, supports multiple languages
- Re-engineering effort is expected to be less, promotes life and shift model very effectively
- Monitoring and notification services are baked into the platform
- Scalability depending on load/demand (upto predefined limits)
- Hot addition of resources
  - Vertical scale of CPU, storage, mem etc
  - Horizontal scale by adding more boxes
- Usual CI is suuported along with IDE integration

Closing thoughts

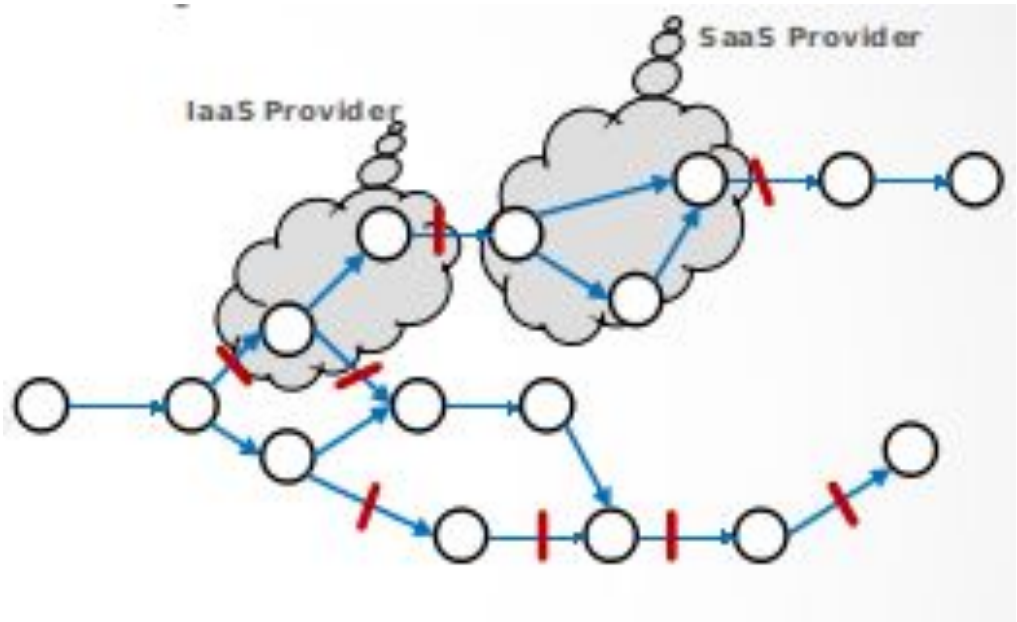
- Focus only on applications
- Hardware and the container layer is abstracted
- Focus on application architecture
- Architectural patterns needs to be followed

Part 5

Software as a service

Enterprise Process Integration

- SaaS is live outsourcing a complete business process to a 3rd party
- Naturally the generated data resides in the DC of that provider
- Usually API based integration is required to pull data out OR batch extraction is also an option
- Issue arises under high data volume - imagine what will happen if you would like to get a list of millions or orders from Amazon webstore!



Data transfer based integration

- Google offers OLAP service under BigQuery
- This is what Google says about this service
  - Querying massive datasets can be time consuming and expensive without the right hardware and infrastructure.
  - Google BigQuery solves this problem by enabling super-fast, SQL-like queries against append-only tables, using the processing power of Google's infrastructure.
  - Simply move your data into BigQuery and let us handle the hard work. You can control access to both the project and your data based on your business needs, such as giving others the ability to view or query your data.
- Alright, I like this but I have 10TB of data!!

API based integration (Backend)

- Some providers are
  - Facebook
  - LinkedIn
  - YouTube
  - There are many others
- This results in a business process fragment outsourcing to a 3rd party
  - Checkout process
  - Payment gateway

API based integration (Frontend)

- UI layer integration
  - FB comments
  - UserVoice
  - There are many others
- This also results in a business process outsourcing BUT the difference is that it has an user touch point / interface

Workshop

Based on understand of IaaS, PaaS and SaaS along with cloud architecture

1. How do you think Netflix uses AWS? Describe failure handling.

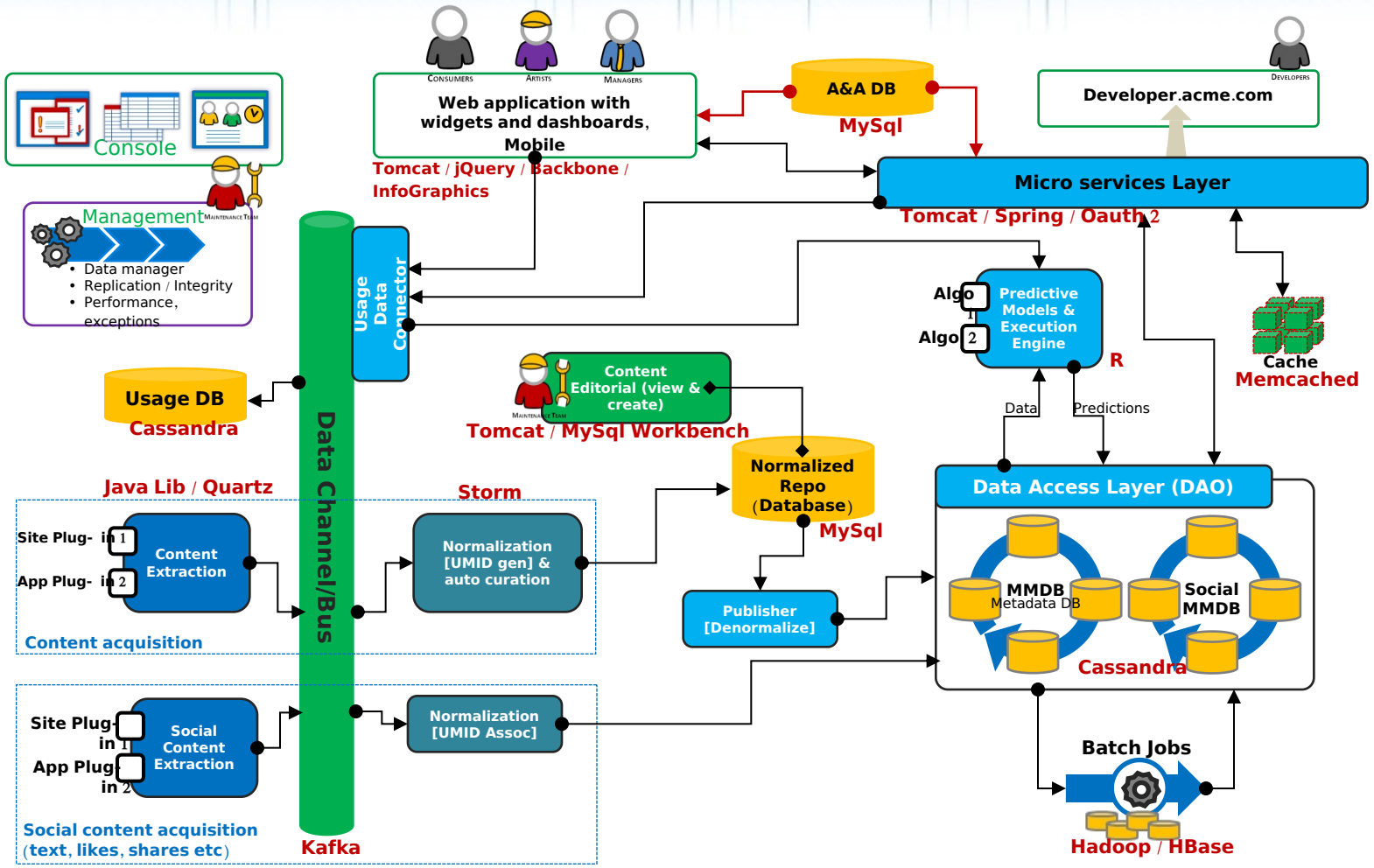
2. How can Timbaktu Electricity Board can create an architecture for 1 million meters communicating every 15 minutes?

3. How can you architect a meta search engine and aggregator? Consider the fashion domain (thefashion.com, lyst.com) or information domain (Flipboard)

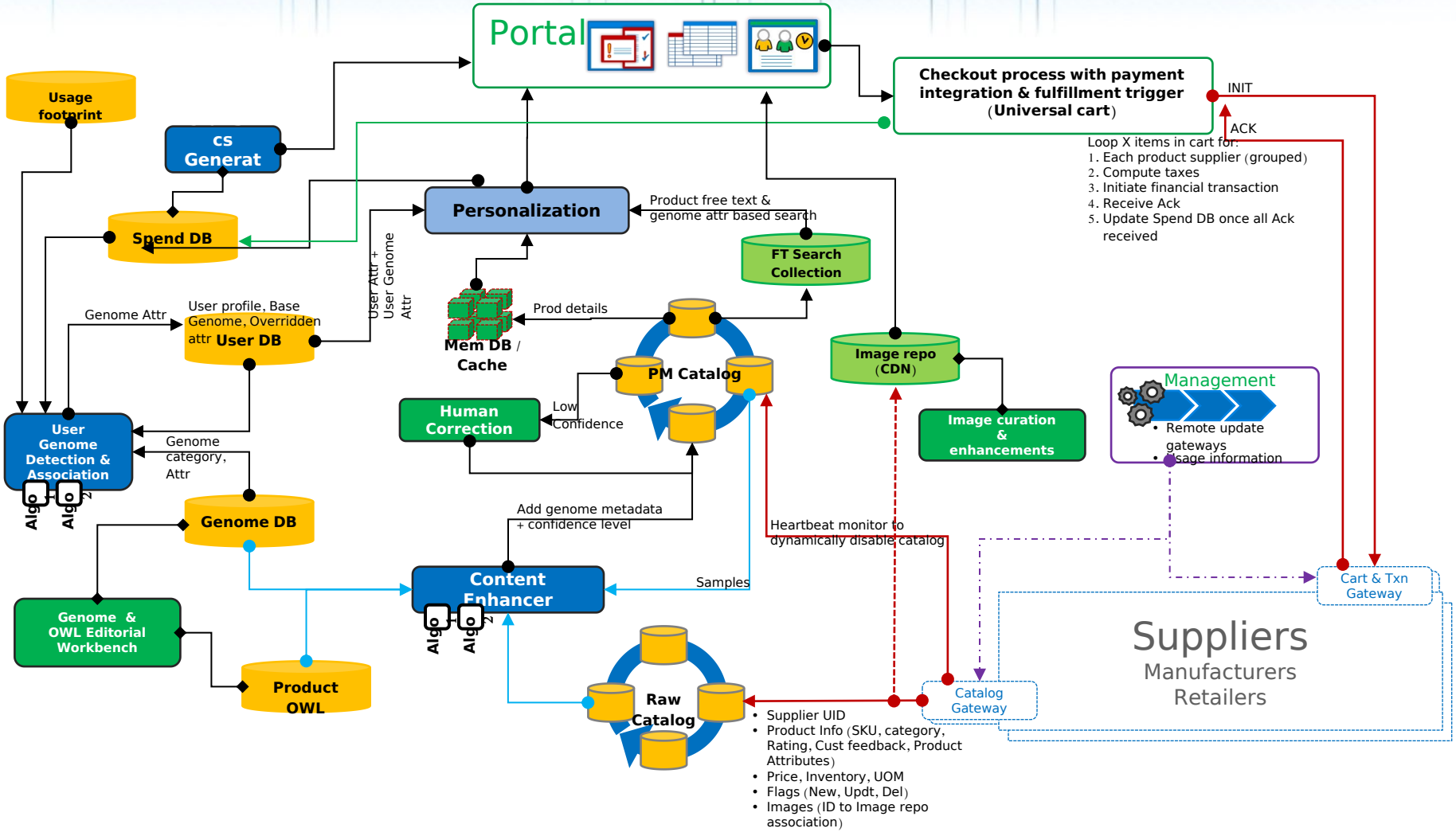
4. What should be the architecture of an application if a country wants the data generated by its users to reside within the country? What issues do you foresee?

5. How startups are architecting for very high transactions per second?

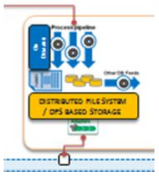
Case: Content acquisition



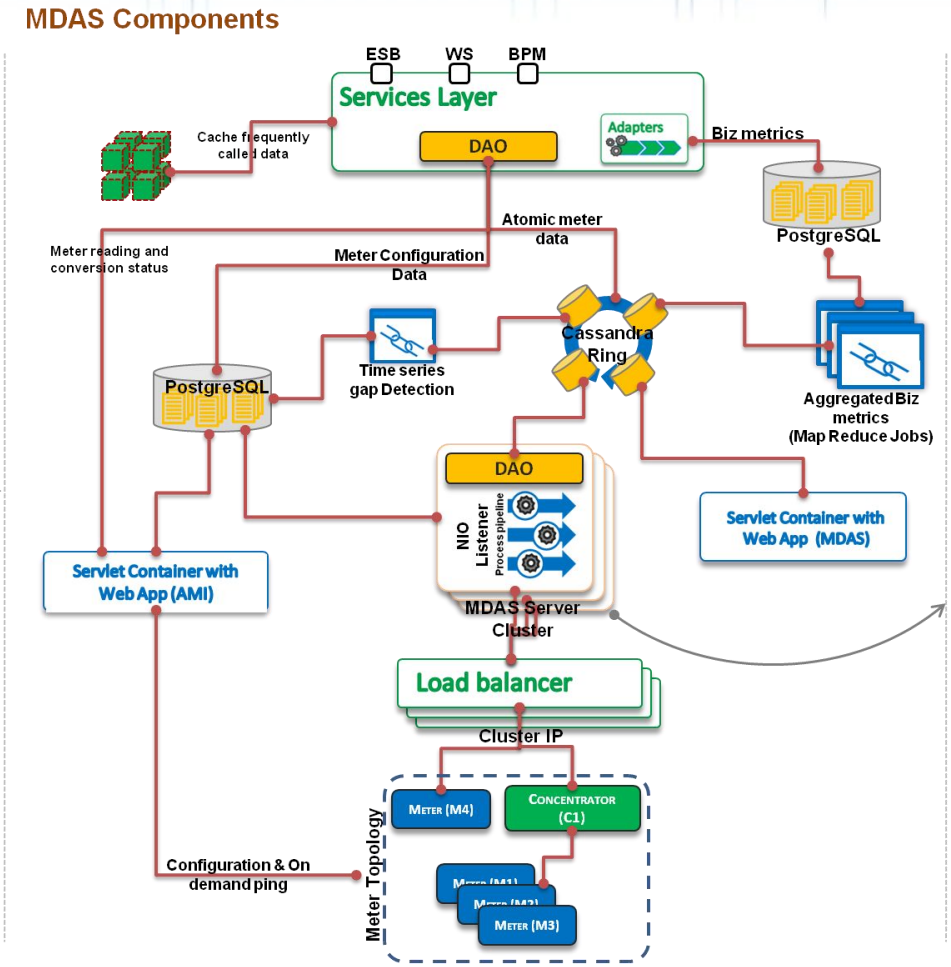
Case: Catalog aggregator and personalization

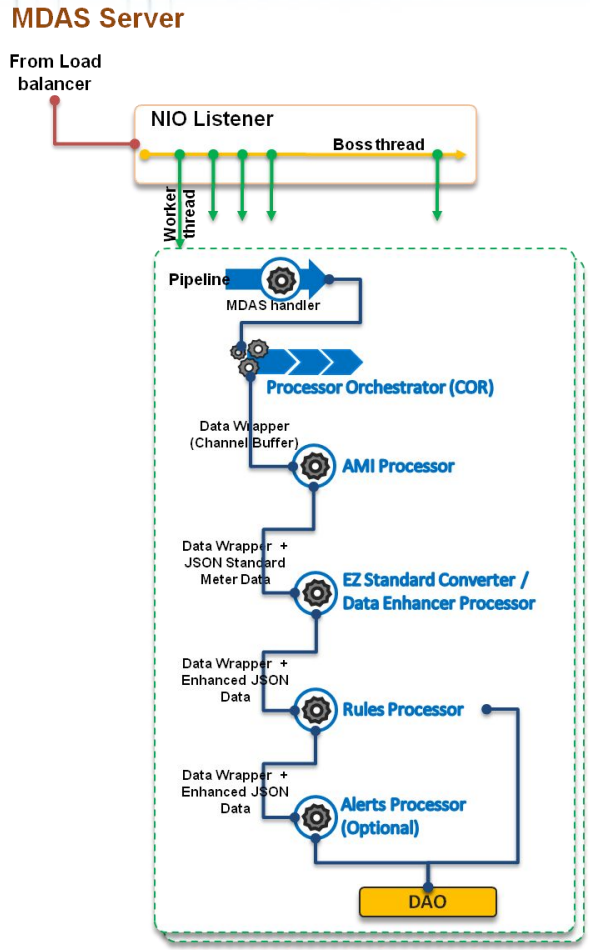


Case: Sensor architecture



MDAS Enterprise Architecture View





About me

Nirmallya Mukherjee

Architect at Independent

Bengaluru Area, India | Information Technology and Services

Current	Independent
Previous	SkroidsLab Technologies Pvt Ltd., Dell Services, Infosys Technologies Limited

www.linkedin.com/in/nirmallya

nirmallya.mukherjee@gmail.com

That's it!