MANAGING DATA ON THE CLOUD

GENOVEVA VARGAS SOLAR

FRENCH COUNCIL OF SCIENTIFIC RESEARCH, LIG-LAFMIA, FRANCE

Genoveva.Vargas@imag.fr

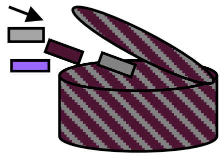
http://www.vargas-solar.com/data-management-services-cloud

DATA MANAGEMENT IN LARGE-SCALE ENVIRONMENTS

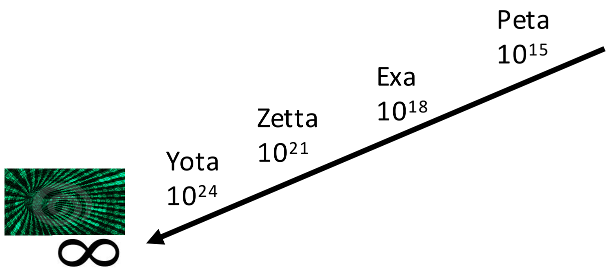
http://news.cnet.com/2300-10805_3-10001679-10.html?tag=mncol

Data management in large-scale environments

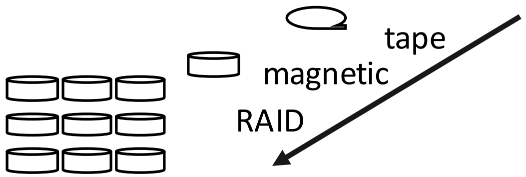
- Definition
- Querying and exploiting
- Manipulation



- Storage (persistency)
- Efficient retrieval (indexing, caching)
- Fault tolerance (recovery, replication)
- Maintenance

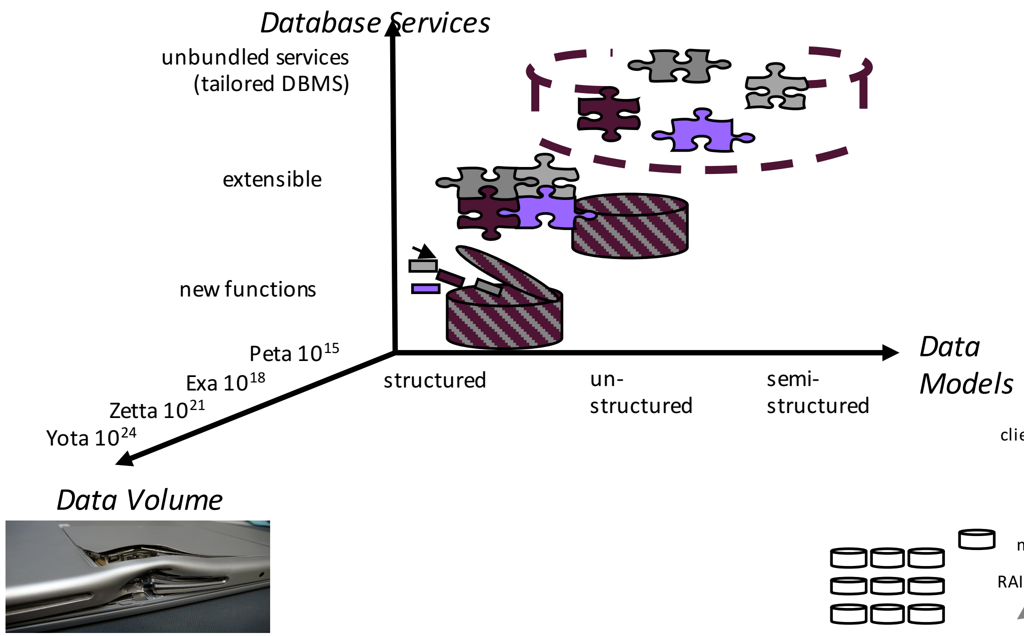


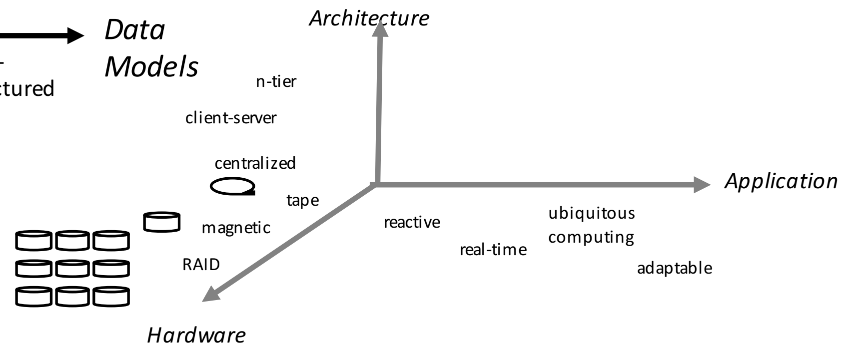
Data Volume



Hardware

DATA MANAGEMENT: STATE OF THE ART AND SOME CHALLENGES



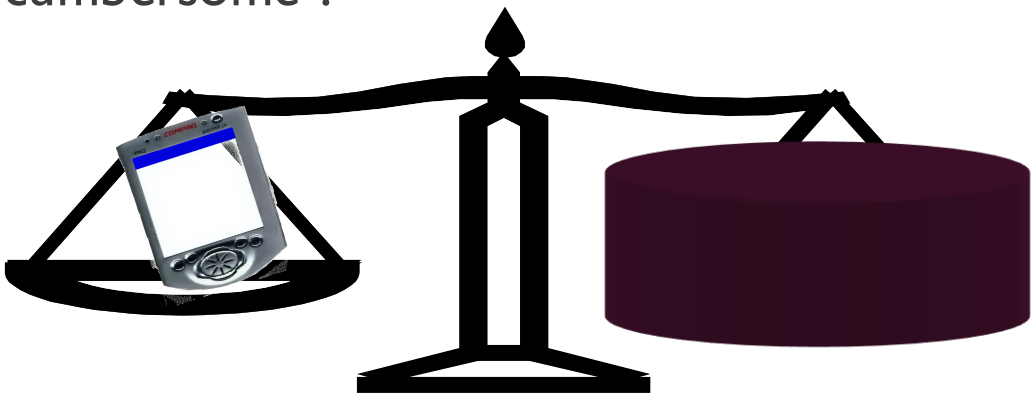


- Use of memory and computing capacities of all computers and servers distributed in the world communicated by a network (e.g. Internet)

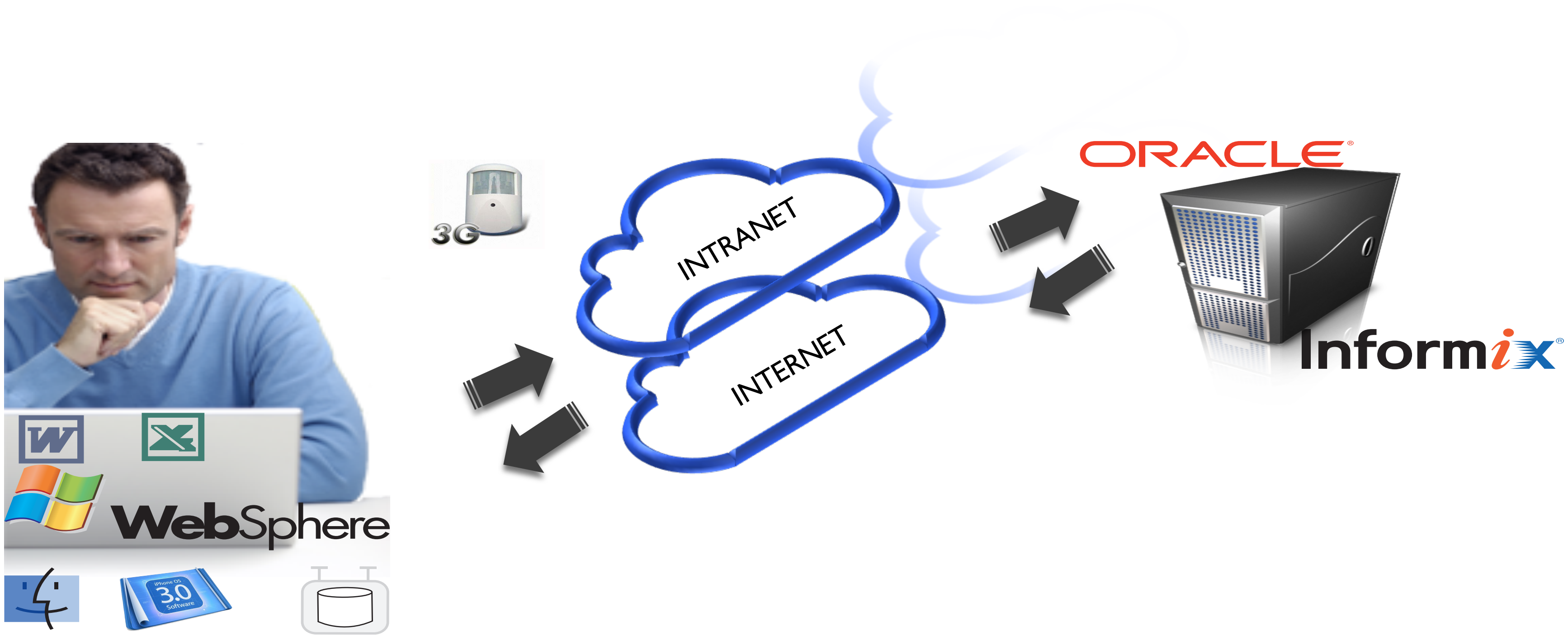
DATA MANAGEMENT IN SMALL-SCALE ENVIRONMENTS

Nowadays, monolithic software:

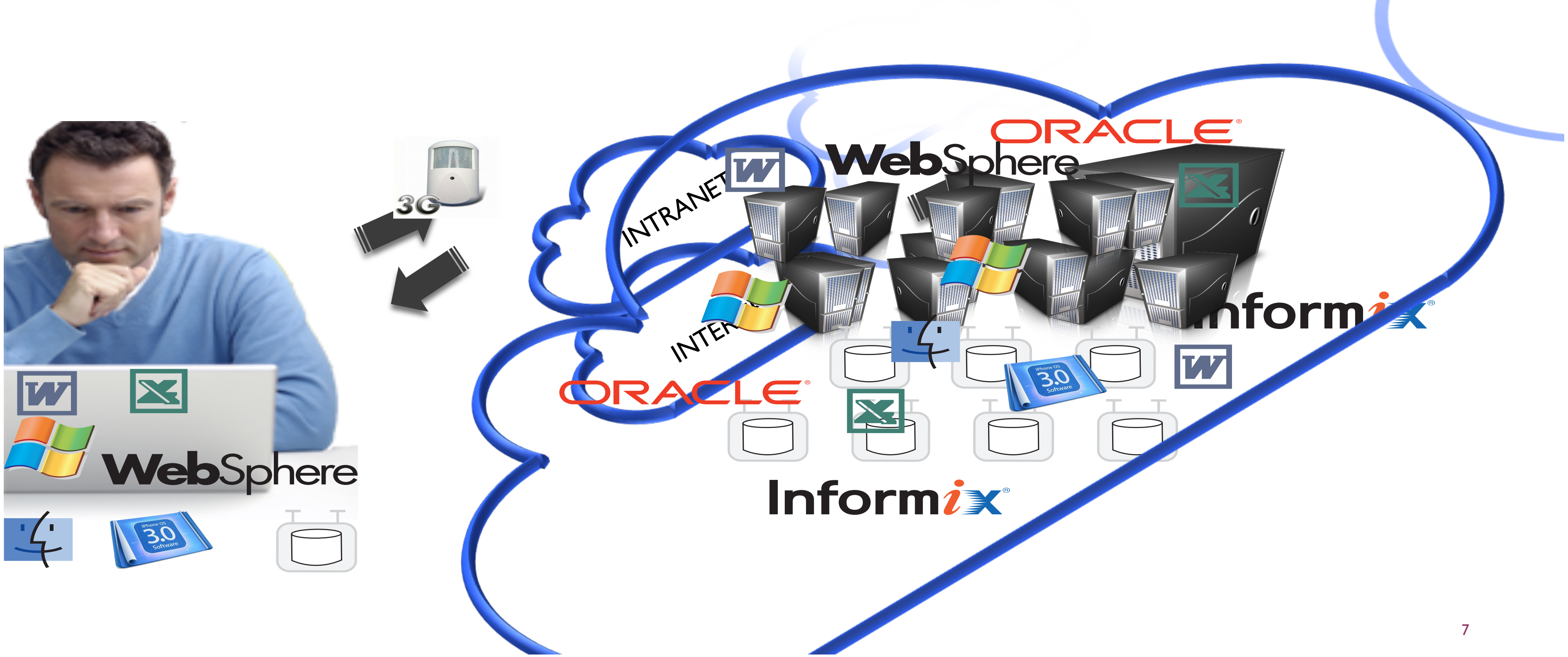
- Adding/removing functions is difficult !
- Full-fledged DBMS can be cumbersome !



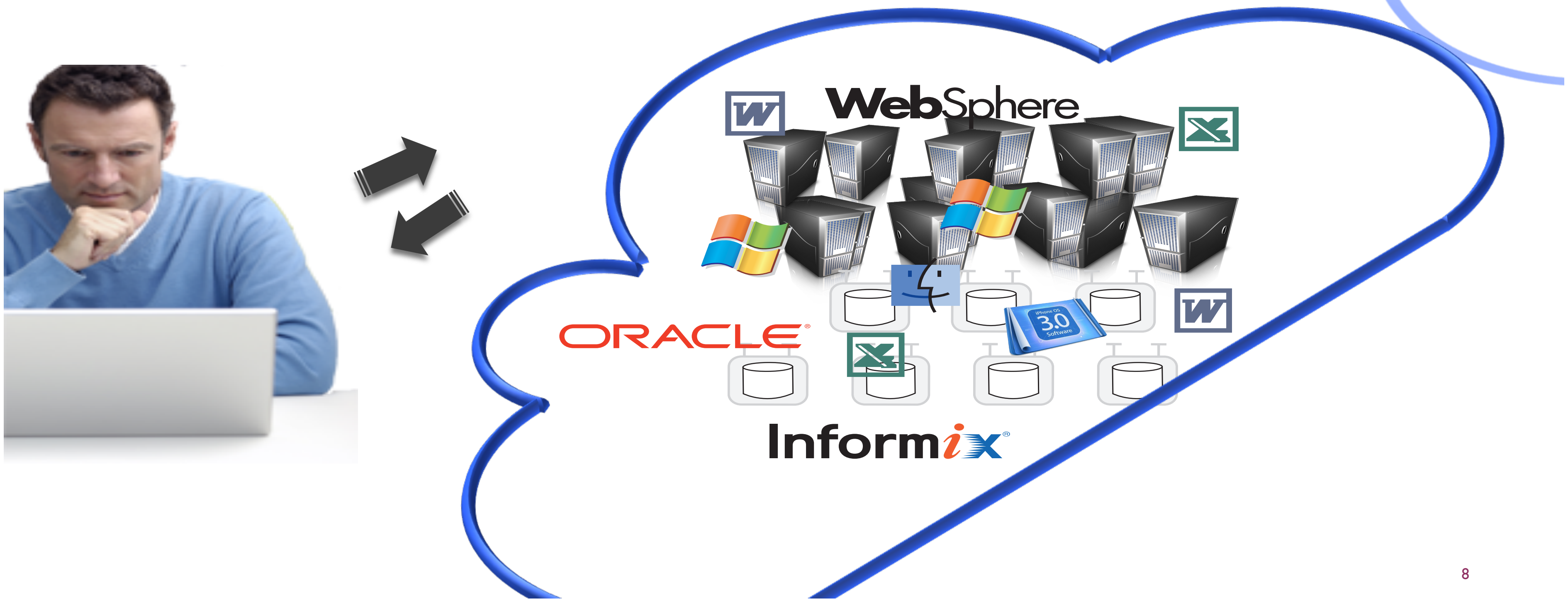
CURRENT SCENARIO



CLOUD PRINCIPLE

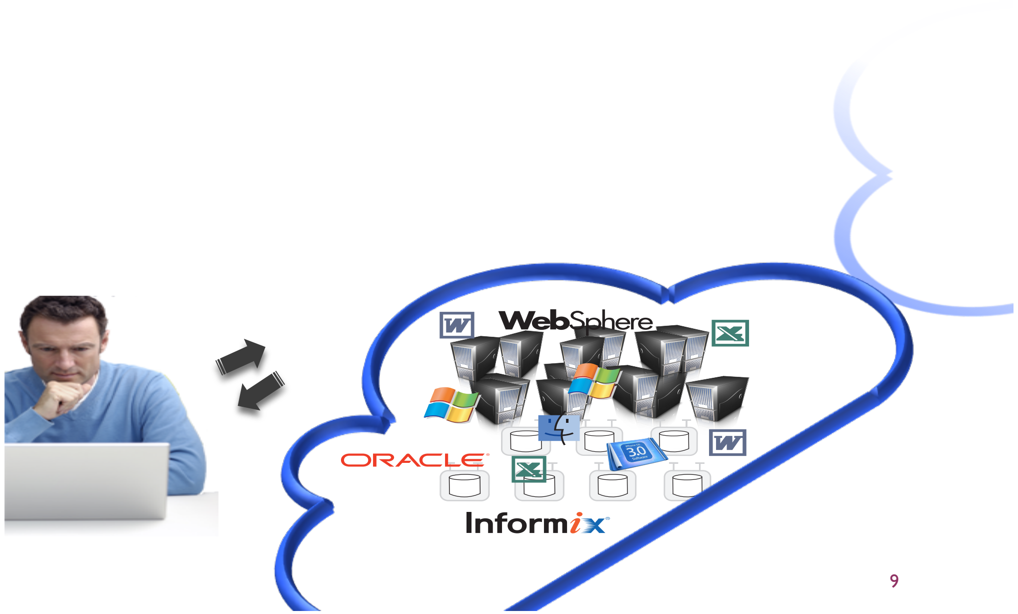


CLOUD PRINCIPLE



CLOUD AIMS

- Ability to use applications on the Internet that store and protect data while providing a service
- Ability to hold application, business and personal data
- Ability to use a handful of Web services to integrate photos, maps and GPS information to create a mashup in a costumer Web Browser



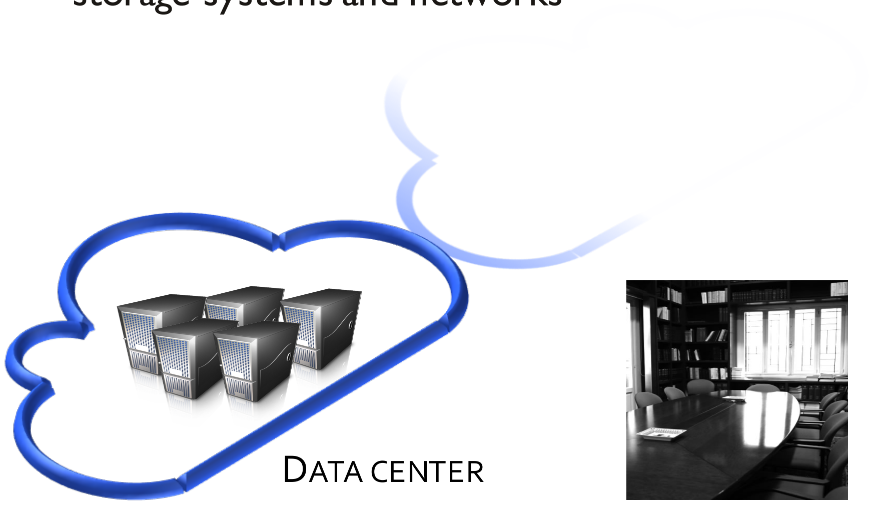
ON DEMAND, SELF-SERVICE, PAY AS U GO MODEL

- Clouds work on a pay as U go model where an application may exist to
  - run a job for a few minutes or hours
  - provide services to costumers on a long-term basis
- Billing is based on resource consumption: CPU hours, volumes of data moved, or gigabytes of data stored

INFRASTRUCTURE MODELS

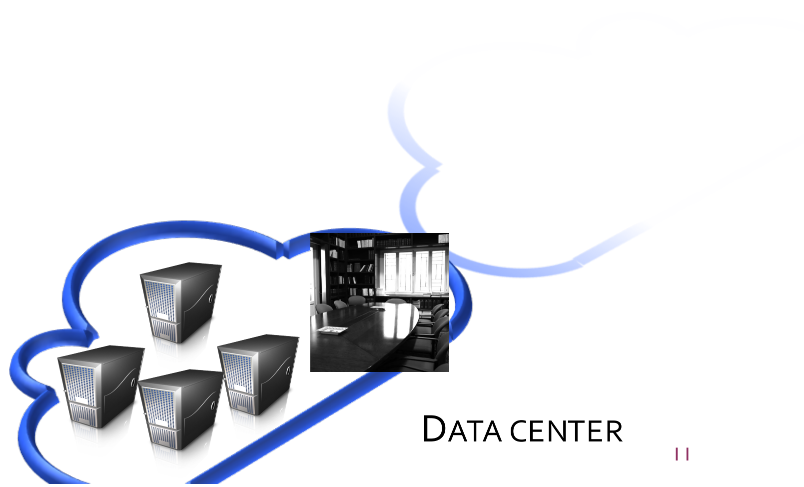
PUBLIC CLOUD

- Run by third parties and applications from different clients mixed together on the clouds' servers, storage systems and networks

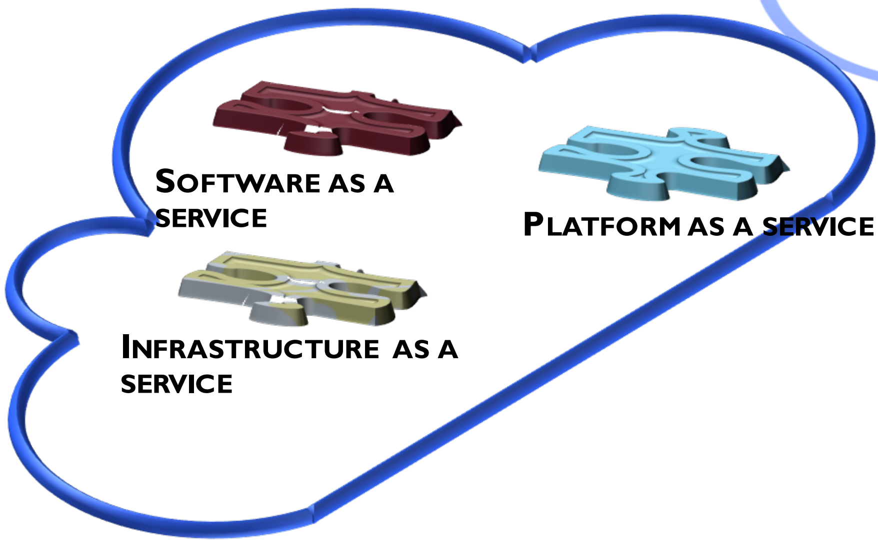


PRIVATE CLOUD

- Used by an exclusive client providing utmost control over data, security and QoS
- The company owns and controls the infrastructure

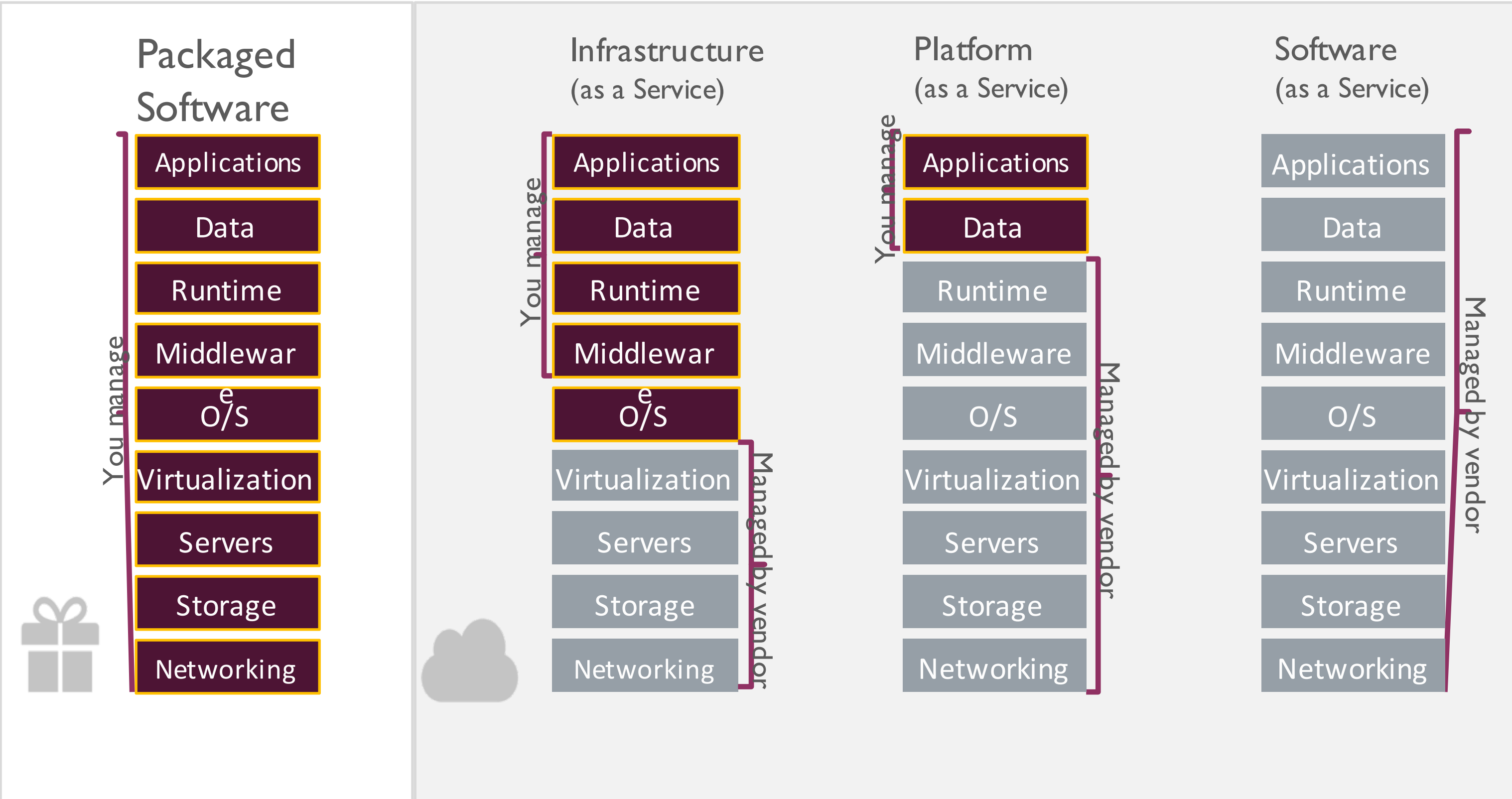


ARCHITECTURE



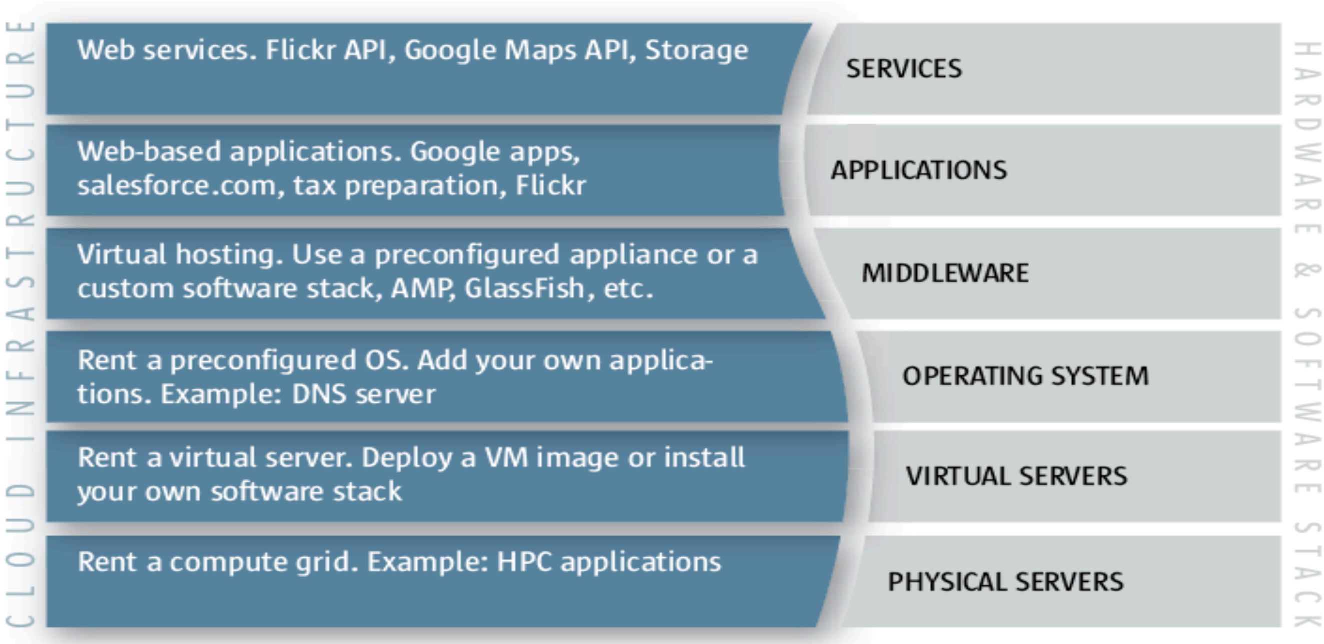
- SOFTWARE AS A SERVICE: applications accessible through the network (Web services, REST/SOAP)
  - Salesforce.com (CRM) and Google (Gmail, Google Apps)
- PLATFORM AS A SERVICE: provide services for transparently managing hardware resources
  - SalesForce.com (Force.com), Google (Google App Engine), Microsoft (Windows Azure), Facebook (Facebook Platform)
- INSFRASTRUCTURE AS A SERVICE: provide Data centers resources and others like CPU, storage and memory
  - Amazon (EC2/S3) and IBM (Bluehouse)

CLOUD COMPUTING: AZURE POINT OF VIEW



SOFTWARE AS SERVICES

SERVICE ON DEMAND

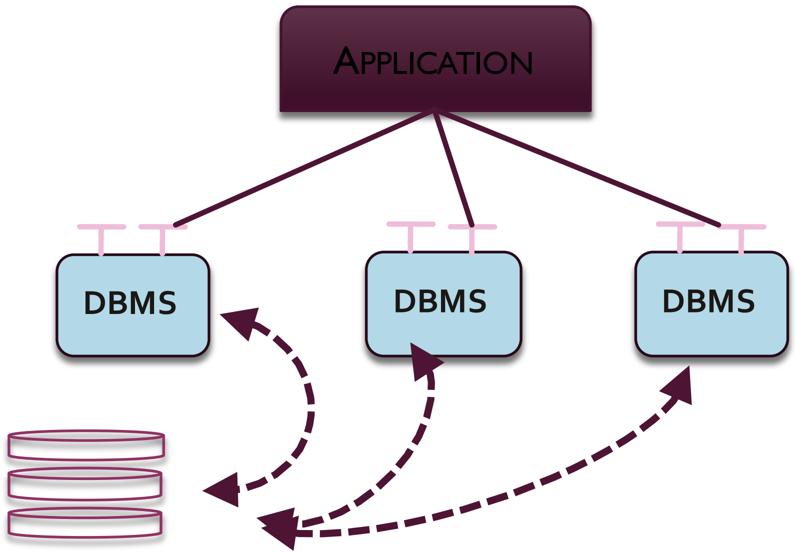


INFRASTRUCTURE AS SERVICE

- Delivers basic storage and computing capabilities as standardized services over the network
- Servers, storage systems, switches, routers are pooled and made available to handle workloads that range from application components to high-performance computing applications
  - e.g., Joyent (http://www.joyent.com/), virtualized servers – performance on-demand infrastructure
- The infrastructure is programmable: developers specify how to configure and interconnect virtual components, how virtual machine and application data are stored and retrieved from a storage cloud

INFRASTRUCTURE AS A SERVICE

- How to deploy components on separate servers to optimize non-functional requirements including scalability, availability, manageability and security
- Dynamic datacenter enable the deployment of virtual application architectures
  - Horizontal scaling
  - Run several database servers, create several virtual machines with integration and parallelization tools
  - e.g., hadoop for map reduce hadoop.apache.org



VIRTUALIZATION

- Virtual machines and virtual appliances become standard deployment object
- Abstract the hardware to the point where software stacks can be deployed and redeployed without being tied to a specific physical server
  - Servers provide a pool of resources that are harnessed as needed
  - The relationship of applications to compute, storage, and network resources changes dynamically to meet workload and business demands
  - Applications can be deployed and scaled rapidly without having to produce physical servers

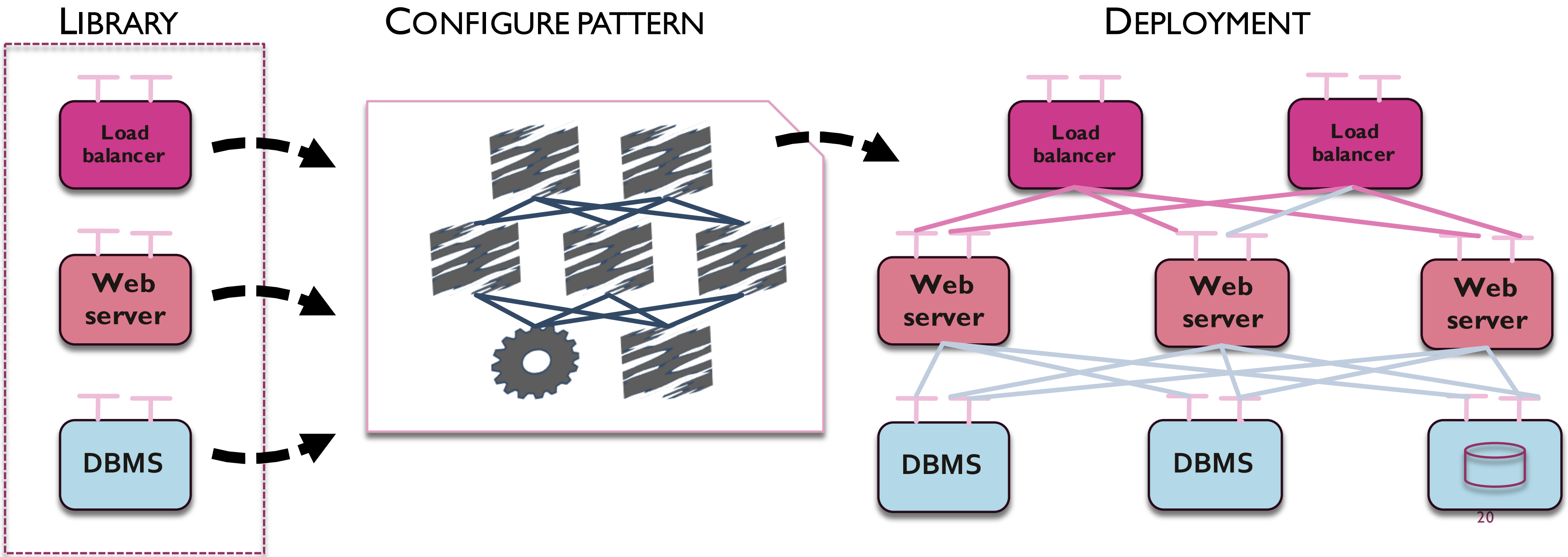
VIRTUALIZATION

- Full virtualization is a technique in which a complete installation of one machine is run on another
  - A system where all software running on the server is within a virtual machine
  - Applications and operating systems
  - Means of accessing services on the cloud
- A compute cloud is a self-service proposition where a credit card can purchase compute cycles, and a Web interface or API is used to create virtual machines and establish network relationships between them

PROGRAMMABLE INFRASTRUCTURE

- Cloud provider API
  - to create an application initial composition onto virtual machines
  - To define how it should scale and evolve to accommodate workload changes
    - Self monitoring and self expanding applications
- Applications must be assembled by assembling and configuring appliances and software
- Cloud services must be composable so they can be consumed easily

PROGRAMMABLE INFRASTRUCTURE



HIGH PERFORMANCE UNDERLYING SUPPORT

- Hardware support: clusters of networked parallel computers
- Well supported by programming models languages and tools
- Concurrency, parallelism, distribution and availability
  - Refine existing programming solutions and investigate new approaches for constructing robust, reliable software

HIGH PERFORMANCE UNDERLYING SUPPORT

- Concurrency
  - Inherent concurrency of cloud computing where autonomous processes interact by exchanging messages
  - Provides control flow to respond to unordered events
  - Supports processing of independent streams of requests
- Parallelism
  - Cloud computing runs on parallel computers on client and server side
  - Higher level programming models such as transactional memory and deterministic execution
- Message passing
  - Primary parallel programming model for cloud computing
  - Inherent performance, isolation with points of interaction
  - Requires adequate interfaces between asynchronous communication of messages and synchronous control flow of procedure calls
  - Erlang (www.erlang.org) integrates message passing constructs to existing languages

HIGH PERFORMANCE UNDERLYING SUPPORT

- Distribution
  - Integrate replication, concurrency, and quorum solutions on a mainstream programming model
  - Libraries or languages with well suited runtimes
- High availability: fault tolerance and efficient exception handling for ensuring services availability at every level

- Performance
  - Shared resources running across large number of computers and complex networks
  - Make performance a first class programming abstraction
- Application partitioning
  - Go beyond client server partitioning that does not support computations migration
  - Virtual machines move an entire image from the OS between computers
  - http://labs.live.com/volta

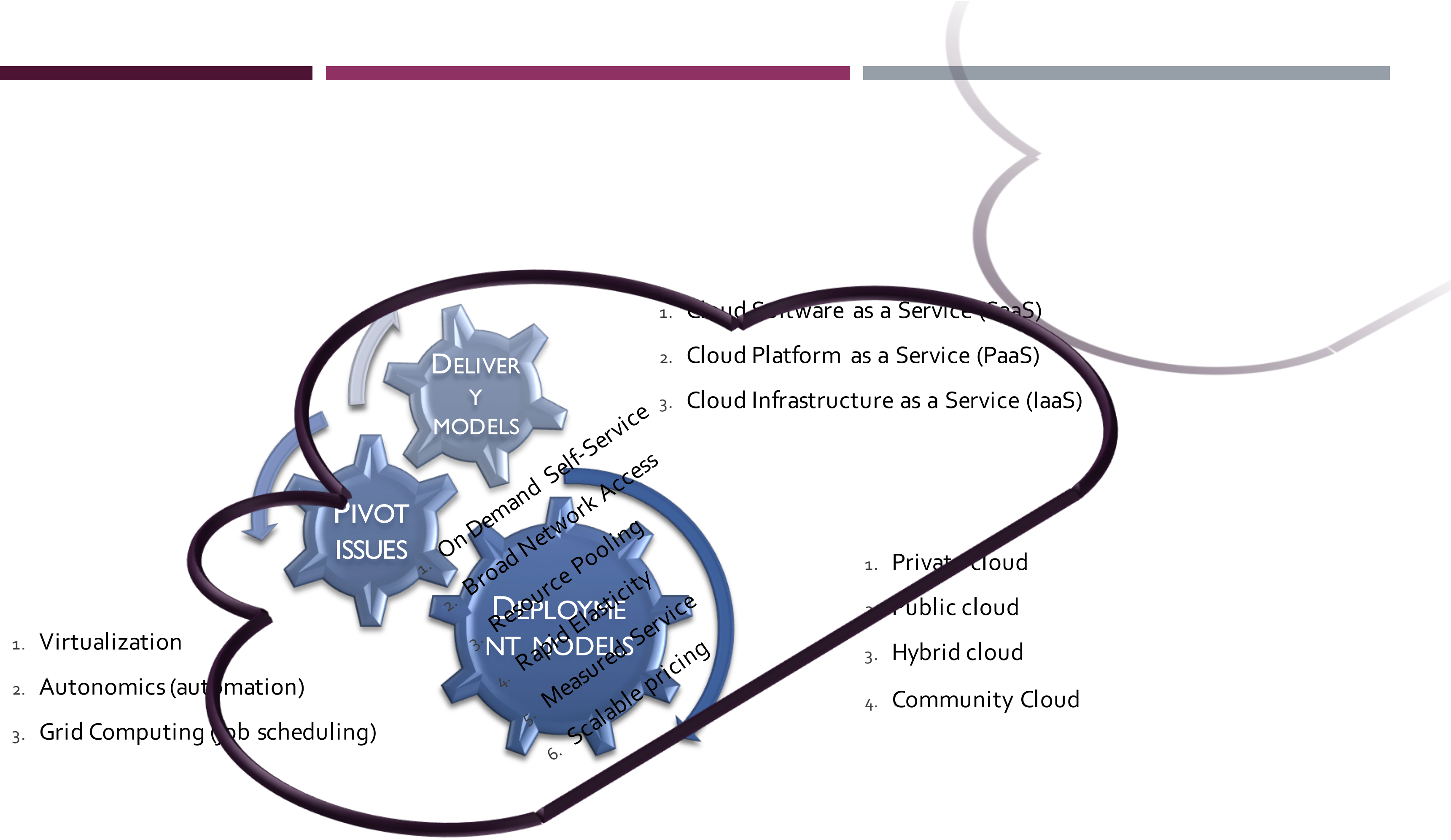
HIGH PERFORMANCE UNDERLYING SUPPORT

- Defect detection
  - A system is resilient it can tolerate failures of its components
  - In the cloud: computers, communication network, other services and the data center in which it turns
  - Detect failures, respond to them minimizing the effect, restore the service when possible and resume execution

- High level abstractions
  - Google's Map Reduce or MS Dryad are higher level programming models that hide the complexity of writing a server-side analytic application
  - Hide complexity of data distribution, failure detection, notification, communication and scheduling
  - Optimization

PLATFORM AS A SERVICE

- Encapsulates a layer of software and provides it as as a service for building higher level services
- Platform integrating an OS, middleware, application software and development environments
  - xVM hypervisor virtual machines including netBeans, Sun GlassFish Web stack and support to languages like Perl and Ruby
- Encapsulated service exporting an API able to manage and scale itself to provide a given level of service
  - Google Apps Engine serving applications on Google's infrastructure



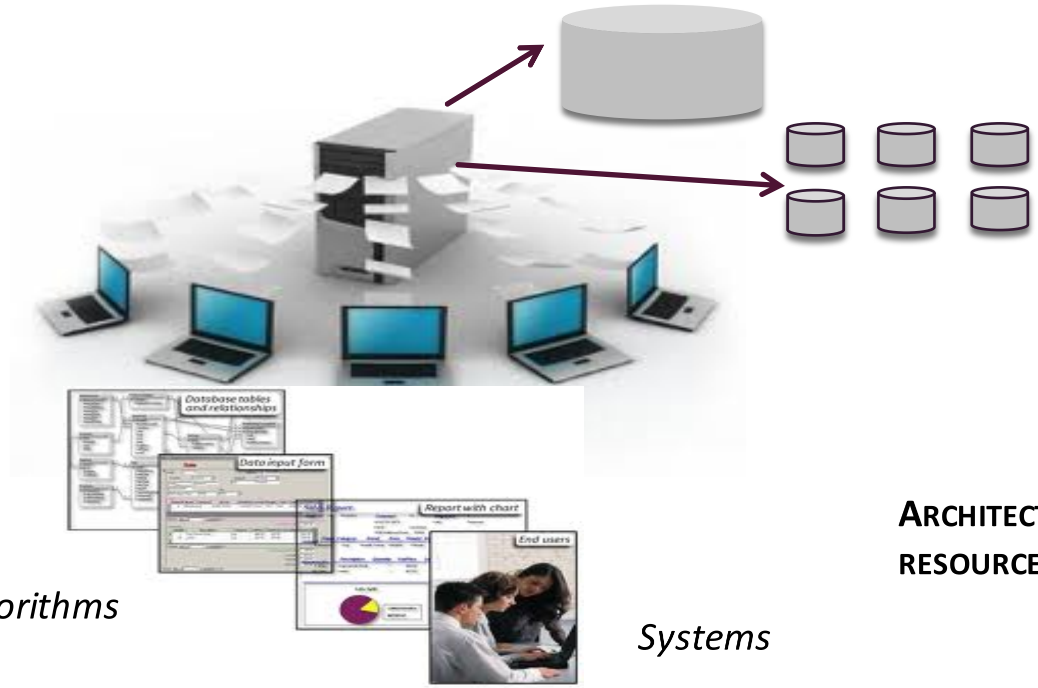
DELIVERY MODELS
PIVOT ISSUES
DEPLOYMENT MODELS
1. Cloud Software as a Service (SaaS)
2. Cloud Platform as a Service (PaaS)
3. Cloud Infrastructure as a Service (IaaS)
1. On Demand Self-Service
2. Broad Network Access
3. Resource Pooling
4. Rapid Elasticity
5. Measured Service
6. Scalable pricing
1. Private cloud
2. Public cloud
3. Hybrid cloud
4. Community Cloud
1. Virtualization
2. Autonomics (automation)
3. Grid Computing (job scheduling)

MANAGING DATA AS A SERVICE

DATA MANAGEMENT WITH RESOURCES CONSTRAINTS

STORAGE SUPPORT

RAM



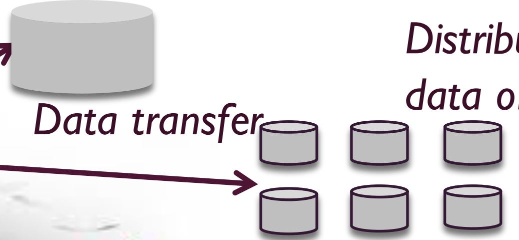
ARCHITECTURE & RESOURCES AWARE

Algorithms

Systems

Efficiently manage and exploit data sets according to given specific storage, memory and computation resources

… WITH RESOURCES CONSTRAINTS

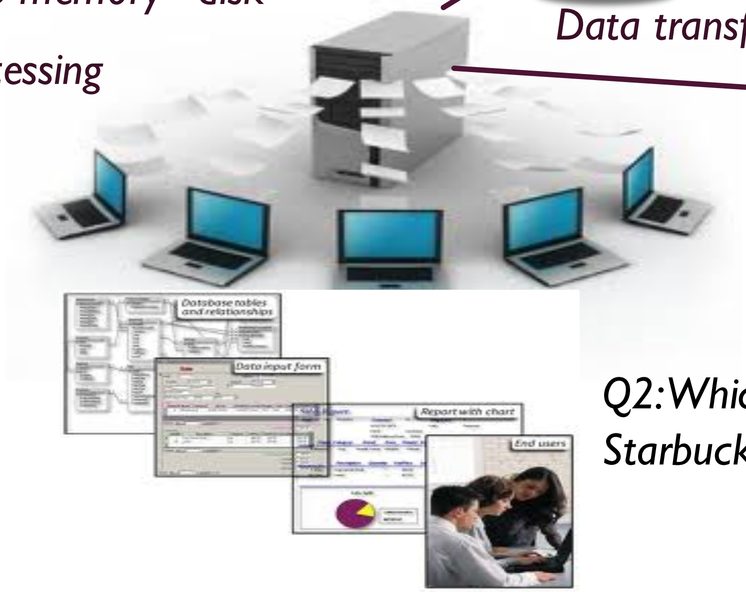


Distribution and organization of data on disk

Swap memory– disk

Data transfer

Query and data processing on server

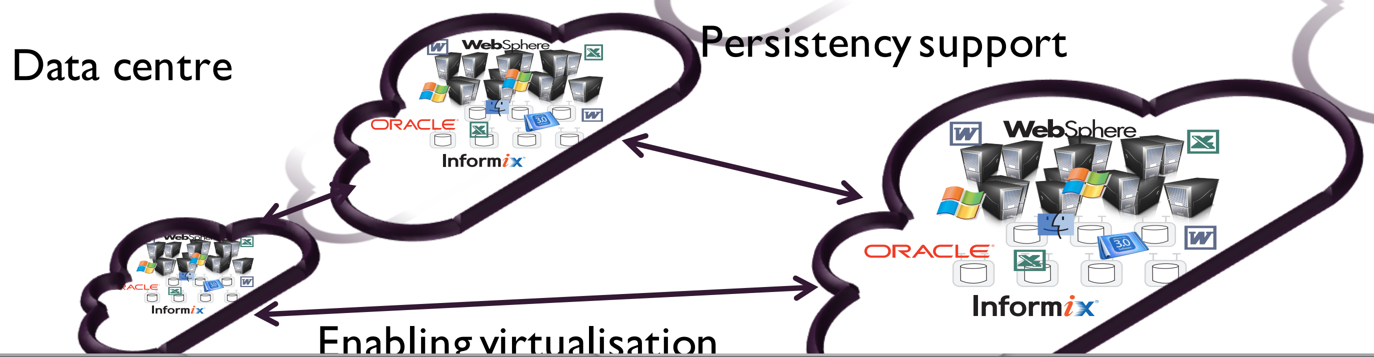


- Efficiency => time cost
- Optimizing memory and computing cost

Q2:Which are the consumption rules of Starbucks clients ?

Efficiently manage and exploit data sets according to given specific storage, memory and computation resources

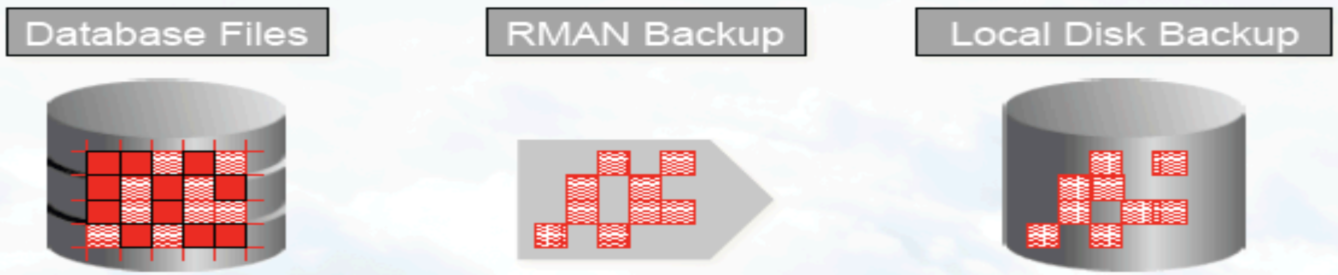
STORING DATA



Amazon Simple Storage Service

- Distributed Data Store
- REST/SOAP web services API
- Simple (Buckets, Objects, Keys)
- Service Level Agreement - 99.9%
- ~70K RPS (scalable, durable, available)
- North America and Europe
- Pay-as-you-go:
  - Tiered storage starts at: $0.15 / GB / month
  - Data Transfer: Tiered $0.17/ GB to $0.10/GB
  - Requests: nominal charges

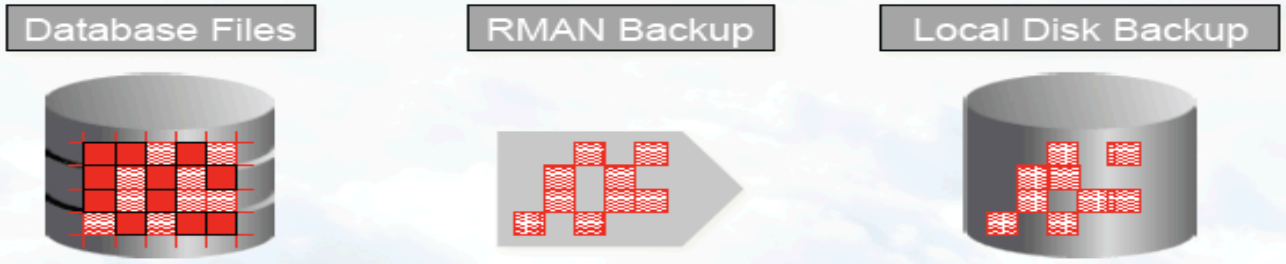
Current Database Backup Best Practice



- Backup to Local Disk
- Migrate backups from disk to tape
- Move tapes to off-site location (e.g. Iron Mountain)

amazon webservices

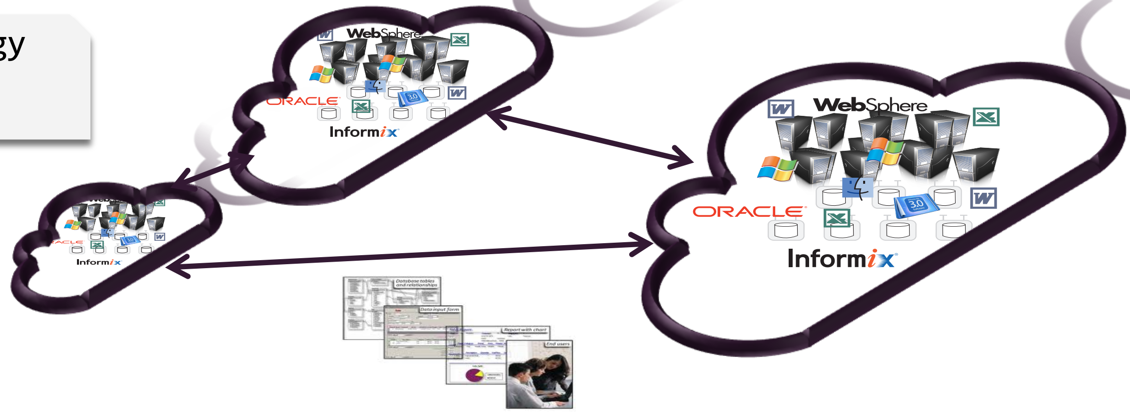
Offsite Backups in the Cloud



- New Oracle Secure Backup module to move database backups to Cloud
  - Works with 9i and higher databases
  - Amazon S3 supported today
- More reliable than tapes, faster restores
- Eliminates tape backup and offsite tape management overhead

WITHOUT RESOURCES CONSTRAINTS …

Costly => minimizing cost, energy consumption



- Query evaluation→ How and under which limits ?
  - Is not longer completely constraint by resources availability: computing, RAM, storage, network services
  - Decision making process determined by resources consumption and consumer requirements
- Data involved in the query, particularly in the result can have different costs: top 5 gratis and the rest available in return to a credit card number
- Results storage and exploitation demands more resources

SCIENTIFIC DATA MANAGEMENT APPLICATIONS

- Old model
  - Query the world"
  - data acquisition coupled to a specific hypothesis
- New model
  - "Download the world"
  - data acquired en masse, in support of many hypotheses
- E-science examples
  - astronomy: high-resolution, high-frequency sky surveys, …
  - oceanography: high-resolution models, cheap sensors, satellites, …
  - biology: lab automation, high-throughput sequencing, ...

HOW TO MAP ARCHITECTURE IN CLOUD ?

How to "map" the components of the reference architecture to (virtual) machines in the cloud.

- How data is collect, transform, integrated, load stored, modeled?
- How to partition data and functions?
  (load balancing)
- How is the consistency of the data maintained (vs availability) ?
- What programming model?
- Whether and how to cache?

SCALABILITY PILLARS

Computing resources
architectures

Programming model: parallelism

Shrek: For your information, there's a lot more to ogres than people think.
Donkey: Example?
Shrek: Example... uh... ogres are like onions!
(holds up an onion, which Donkey sniffs)
Donkey: They stink?
Shrek: Yes... No!
Donkey: Oh, they make you cry?
Shrek: No!
Donkey: Oh, you leave 'em out in the sun, they get all brown, start sproutin' little white hairs...
Shrek: *(peels an onion)* NO! Layers. Onions have layers. Ogres have layers. Onions have layers. You get it? We both have layers.
(walks off)
Donkey: Oh, you both have LAYERS. Oh. You know, not everybody like onions. What about cake? Everybody loves cake!
Shrek: I don't care what everyone else likes! Ogres are not like cakes.
– from the 2001 Dreamworks movie "Shrek" [6]

Going for Ogres, onions or parfaits?

Vinayak borkar, Michael J. Carey, Chen Li, Inside "big data management": ogres, onions, or parfaits?, EDBT, 2012

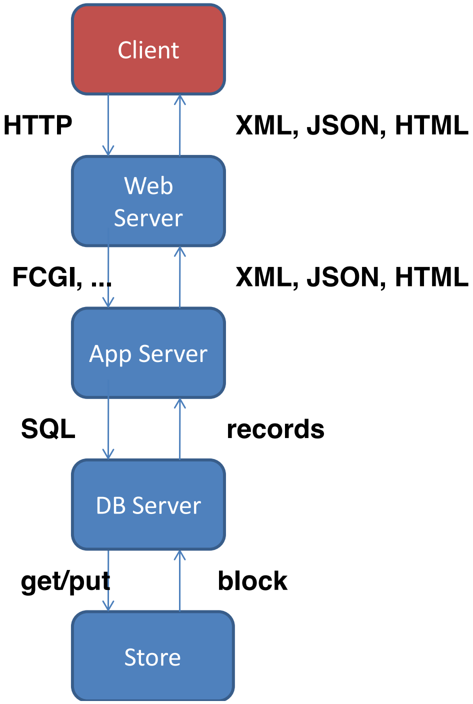
Execution platforms
DM systems

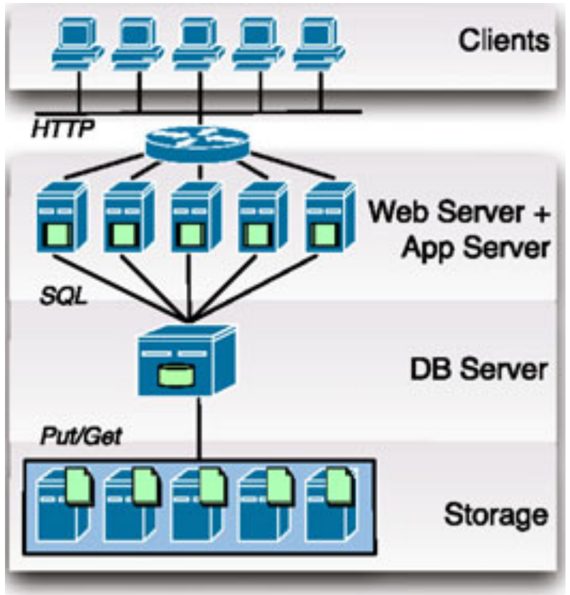
AsteriX DB
more engine less trunk

CLOUD AWARE APPLICATIONS ARCHITECTURES

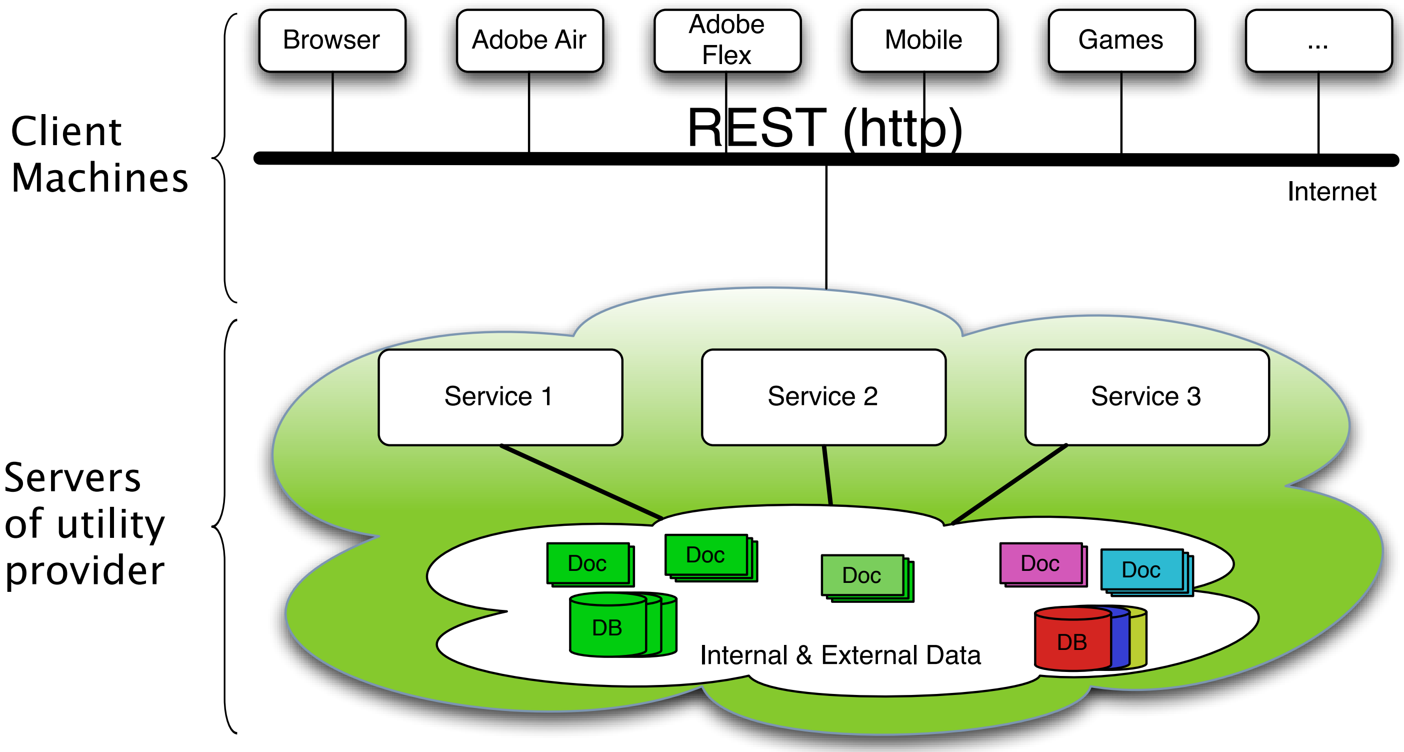
- Good plan for dividing data with tools implementing master/worker or other parallelization patterns
- Data partitioning techniques, real – time analysis
- Data physics: balance between local data processing and data transfer costs
  - Combine data and computing power, e.g. virtual machine location and data storage location

APPLICATION REFERENCE ARCHITECTURE





BUT IS IT VALUABLE ? AND HOW ?



SOME COMMENTS …

- Data management applications are potential candidates for deployment in the cloud
  - industry: enterprise database system have significant up-front cost that includes both hardware and software costs
  - academia: manage, process and share mass-produced data in the cloud

- Many "Cloud Killer Apps" are in fact data-intensive
  - Batch Processing as with map/reduce
  - On line Transaction Processing(OLTP) as in automated business applications
  - Offline Analytical Processing(OLAP) as in data mining or machine learning

COLLECTING DATA

DATA ACQUISITION

- **Traditional sensing and measurement**: installing sensors dedicated to some applications
- **Passive crowd sensing**
- **Participatory sensing**

DATA ACQUISITION

- **Traditional sensing and measurement**
- **Passive crowd sensing**: wireless cellular networks are built for mobile communication between individuals to sense city dynamics (e.g., predict traffic conditions and improve urban planning)
  - Sensing City Dynamics with GPS-Equipped Vehicles: mobile sensors continually probing the traffic flow on road surfaces processed by infrastructures that produce data representing **city-wide human mobility patterns**
  - *Ticketing Systems of Public Transportation* (e.g., model the city-wide human mobility using **transaction records** of RFID-based cards swiping)
  - *Wireless Communication Systems* (e.g., call detailed records CDR)
  - *Social Networking Services (*e.g., geo-tagged posts/photos, posts on natural disasters analysed for **detecting anomalous events** and **mobility patterns** in the city)
- **Participatory sensing**

DATA ACQUISITION

- **Traditional sensing and measurement**
- **Passive crowd sensing**
- **Participatory sensing:** people obtain information around them and contribute to formulate collective knowledge to solve a problem (i.e., **human as a sensor**)
  - *Human crowd-sensing*: users willingly sense information gathered from sensors embedded in their own devices (e.g., GPS data from a user's mobile phone used to estimate real- time bus arrivals)
  - *Human crowd-sourcing:* users are proactively engaged in the act of generating data: reports on accidents, police traps, or any other road hazard (e.g. Waze), citizens turning into cartographers, to create open maps of their cities

DATA INTEGRATION IN THE CLOUD

- **Resource consuming model** focussing on the **technical** and **economic** conditions to be fulfilled to access potentially unlimited resources
- **Integrating and processing** heterogeneous data collections, calls for **efficient methods** for
  - correlating, associating, and filtering them considering their variety (i.e., different formats and data models)
  - quality, e.g., trust, freshness, provenance, partial or total consistency.

DATA INTEGRATION IN THE CLOUD

- **Quality of service (QoS)** requirements expressed by data consumers and **Service Level Agreement (SLA) contracts** exported by data services
- **Cloud providers** that host these collections and deliver resources for executing data processing and integration processes
- **SLA- based data integration** for better meeting user requirements related
  - to the conditions in which data is delivered and integrated
  - on the quality of the data provided by services

MOOC SCENARIO

- **Producers** characterized by `location`, provided `content type and topic`, access conditions (e.g. cost, inscription, or exchange unit), and content production `time window`
- **Consumers** characterized by `location`, interests during a `time interval`, maximum cost of the consumed content, or resources to get the service, and QoS requirements (availability and how critical it is to consume a given type of content)
- Producers and consumers
  - Have subscriptions **to different cloud providers** for dealing with content storage, processing and exchange
  - Can ask to minimize the transfer of personal data when they share/consume content

MOOC SCENARIO

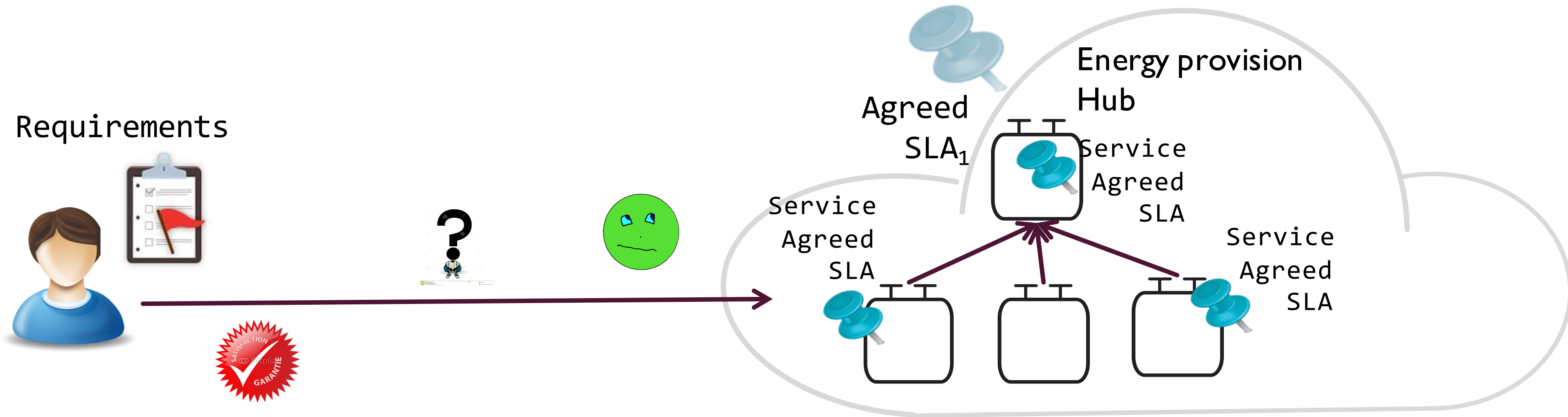
- **MOOC**
  - Aims at being privacy respectful of the producers and consumers participating in courses
  - Uses privacy preserving techniques to let users share content anonymously according to the level of trust associated to data providers
- Data providers can also wish to give restricted data access credentials w.r.t. to their trust level, when their data are used within an integration process

PROBLEM STATEMENT

- How can the user efficiently obtain results for her queries such that they meet her QoS requirements
  - they respect her subscribed contracts with the involved cloud provider(s)
  - they do not neglect services contracts
- Particularly, for queries that call several services deployed on different clouds

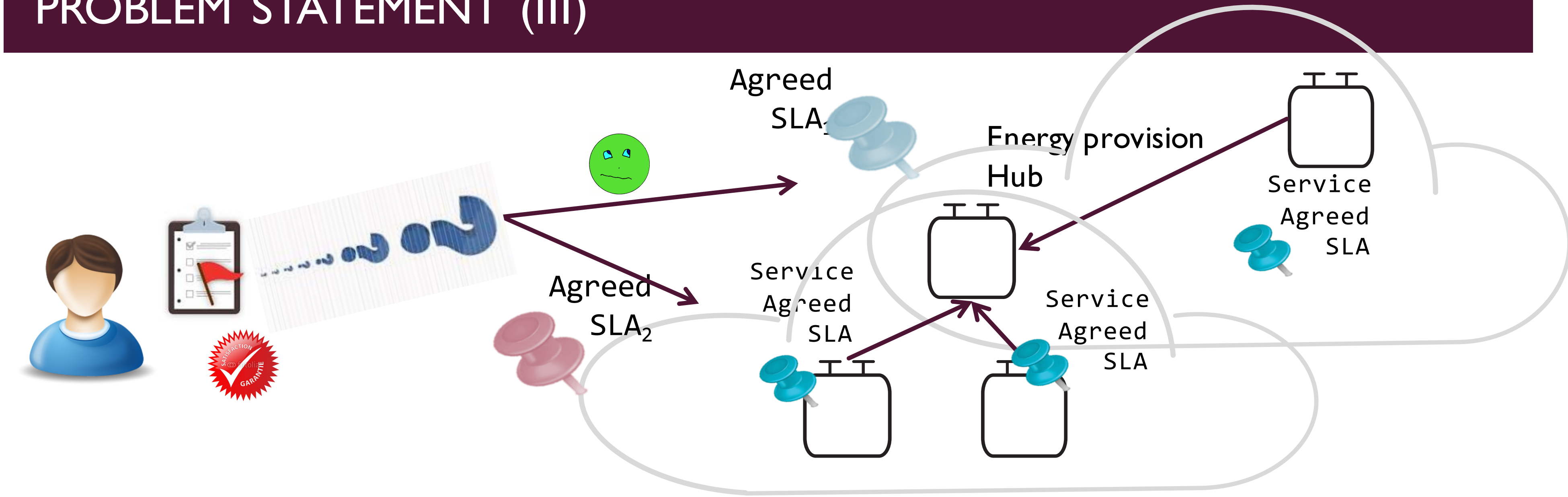
Integration can be done **enforcing all/some** specified **conditions**
Matching data providers with requests and QoS preferences with SLA's can be computationally **costly**
→ **results should be capitalized** for further integration requests

PROBLEM STATEMENT (II)



How to be sure that all the agreed SLAs are respected while satisfying the user?

PROBLEM STATEMENT (III)



Can my constraints be satisfyed? Which services shall I ask for?

How can ressources be saved for next query?

OBJECTIVES

Propose an SLA guided continuous data integration and provision system as a DaaS

- Integrated SLA computation out of the Data agreed SLA
- Optimized and adaptable data collection, query rewriting and integration according to user preferences
- Learning based data integration mechanisms

HOW TO EXPLOIT SLA FOR INTEGRATING DATA

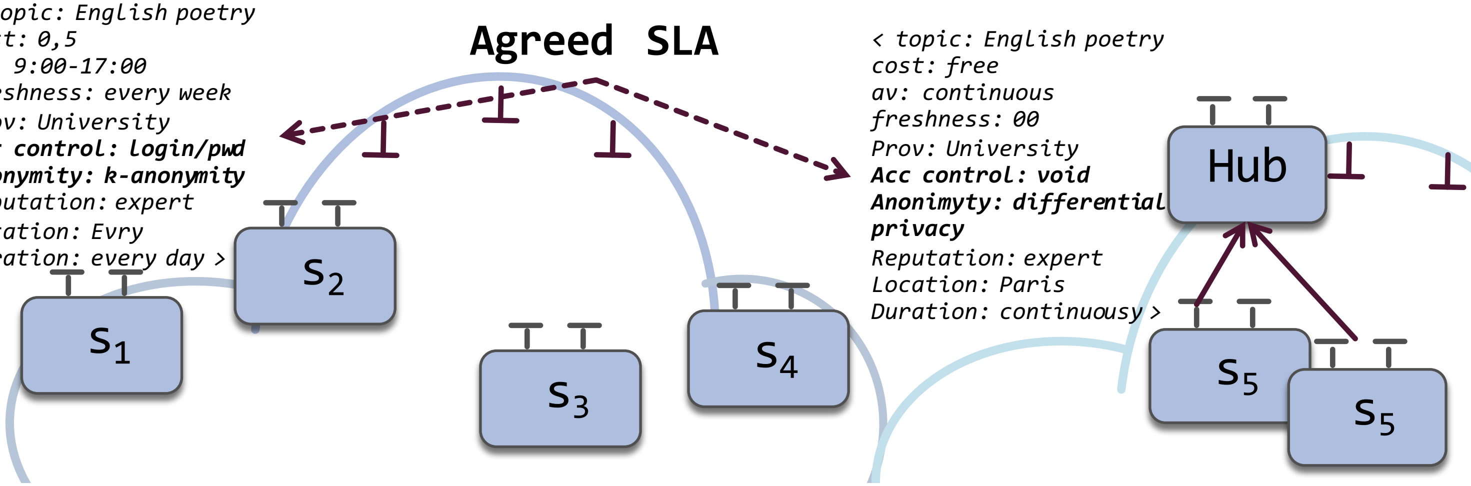
List of English poetry content providers that can provide commented Emily Dickinson poems

QoS preferencesuser : ⟨cost ≤ \$1, freshness = “any”, provenance = “certified”, location = “close”, duration = 7 days, **privacy- preserving=“reputation-based”**⟩.

estim-cost$_i$(contents, req size, cost, prov size, loc), agreedSLA$_i$

engage$_i$(contents, req size, payment), agreedSLA$_i$

< *topic: English poetry*
cost: 0,5
av: 9:00-17:00
freshness: every week
Prov: University
Acc control: login/pwd
Anonymity: k-anonymity
Reputation: expert
Location: Evry
Duration: every day >

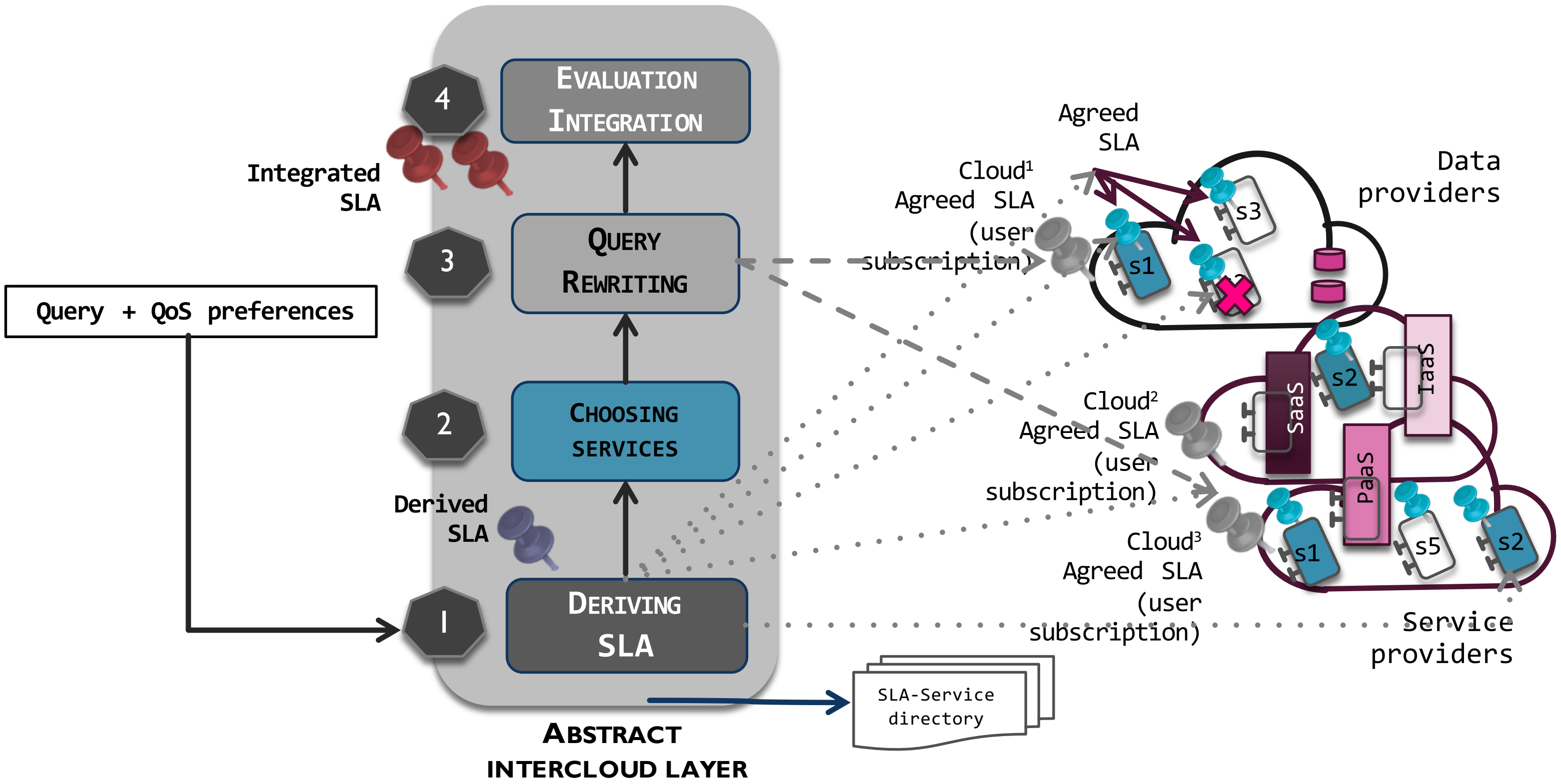


< *topic: English poetry*
cost: free
av: continuous
freshness: 00
Prov: University
Acc control: void
Anonimyty: differential privacy
Reputation: expert
Location: Paris
Duration: continuousy >

Cloud SLA

⟨0,05 cents/call,
8 GB I/O volume/month,
free, 1 GB storage⟩.

HOW TO EXPLOIT SLA FOR INTEGRATING DATA



CHALLENGES BEHIND AN SLA-GUIDED APPROACH

- Agreed-SLA:
  - Its content should allow to match user preferences wrt to service features
  - a service-centric monitoring for service static and dynamic deployment conditions
  - **Challenge: How to compute coarse grained measures with fine grained ones?**
- Derived-SLA
  - Guides the way the query will be evaluated, and the way results will be computed and delivered
  - Helps learning for further data integration operations
  - **Challenge : How to consider in real time the Agreed SLA clauses in the rewriting algorithm, especially for dynamic clauses?**

DERIVED SLA

- Set of measures that correspond to the user preferences computed as a function of different static, computed as a function of different measures
- Inequations that have to be solved during the execution of a service composition.
- Guides the way the query will be evaluated, and the way results will be computed and delivered
- User preferences statement measures are used for defining a derived SLA

```
• total cost: Σi=1...n cost(si) + data transfer + encryption cost ≤ $1;
• availability: (of services involved) ≥ 90%;
• freshness: non;
• provenance: all services involved must be expert;
• duration: 7 days;
• I/O volume/month: 8GB;
• reputation level: ≥ threshold;
• storageSpace: 1GB
```

REWRITTING QUERIES

- The query to be rewritten is seen as **an abstract service composition** to be expressed in terms of concrete services.
- Generating translations of an abstract query into several compositions over concrete services
- **PRINCIPLE**: given a Query denoting abstract services and its Preferences→ generate the service compositions that express
  - The query in terms of a composition of **concrete services**
  - The preferences as **a set of constraints** representing the **derived SLA**

QUERY EXPRESSION

```
Q(myIPaddress, “E.Dickinson”, 1MB, “expert”, $1, myCreditCard) ≡
myLoc = loc(myIPaddress),
estim cost(“E.Dickinson”, 1MB, cost, size, theirLocation),
query total cost + cost ≤ $1,
availability ≥ 90%, freshness = any,
provenance = “expert”, duration = 7 days,
storageSpace ≤ 1GB, I/O volume/month ≤ 8GB,
reputation level ≥ λ (the threshold)
engage(“E.Dickinson”, size, myCreditCard).
```

- Query to be rewritten with respect to the available concrete services

Genoveva Vargas-Solar
Senior Scientist, CNRS
LIG-LAFMIA, France

TECHNICAL SUPPORT

Javier Espinosa
LAFMIA (UMI 3175)
France

Genoveva.Vargas@imag.fr
http://www.vargas-solar.com/data-management-services-cloud