Ad Hoc Cloud Computing From Concept to Realization

Lectures Collections

Prepared by

Prof. Fatma Omara

Ad Hoc Cloud Computing

- Current Cloud Computing is primarily based on proprietary data centers, where hundreds of thousands of dedicated servers are setup to host the cloud services.
- There are ***billions*** of ***underutilized Personal Computers*** (PCs), usually ***used*** only for a ***few hours*** per day, owned by ***individuals*** and ***organizations*** worldwide.
- The **unused compute** and **storage capacities** of the ***underutilized PCs*** owned by individuals and organizations can be **consolidated** as ***alternative*** cloud fabrics to provision broad cloud services, primarily ***infrastructure as a Service***.
- This approach referred to as ***"no data center"*** approach, complements the data center based cloud provision model.

Ad hoc Cloud Computing-Cont..

- The concept of ***ad hoc Cloud*** within an enterprise is that ***harness unused resources*** to ***improve*** overall ***utilization***, ***reduce energy consumption*** , and ***allow*** organizations to take advantage of ***operating their own in-house cloud***.
- An ***ad hoc Cloud*** harvests resources from existing **nonexclusive** available hosts used by ***host users*** (e.g. company employees) and exposes these resources to ***cloud jobs submitted by cloud users***.
  - **Example**, aggregating unused processing and storage capacity of an University of ten thousand machines in offices and labs to represent a major untapped computing resource.

Ad Hoc Cloud Computing-Cont..

- An ***ad hoc Cloud Computing*** platform is developed to transform ***spare resource capacity*** from an infrastructure owner's locally available, but non-exclusive and unreliable infrastructure, into an ***overlay Cloud platform.***
- ***Cloud jobs*** are submitted to ***ad hoc guests***, or **VMs**, running on each of the hosts within ***ad hoc Cloud***
- Applications require ***extremely high levels of security***, may not be suited to the ad hoc cloud, or any Cloud implementation.
- Each host has ***an ad hoc client*** installed to ***control*** the guest, ***monitor*** resources and ***provide state information*** to the server.
- Within an ad hoc cloud, a ***set of cloudlets*** are created; a set of ***connected ad hoc guests*** that offer a particular **service** or **environment**.

Ad Hoc Computing Terminologies

Non-exclusive Infrastructure

An infrastructure whose hosts are reserved for ***some primary purpose***,

- e.g., employee workstations running company applications.

Ad Hoc Host

A physical machine whose resources are donated to the ***ad hoc Cloud,*** but is used for some other ***primary purpose***.

Host Processes

Processes which execute by **Ad hoc host** (i.e., ***host*** and ***guest processes)***

Ad Hoc Computing Terminologies- Cont..

Cloudlet

- A set of connected ***ad hoc guests*** that provide a particular service or execution environment.

Note That

Cloudlet members could be distributed across an infrastructure

- a **host** may either **dedicate** to a **single cloudlet** or **attached** to **many cloudlets** to allow applications from different domains to run concurrently upon the same host.

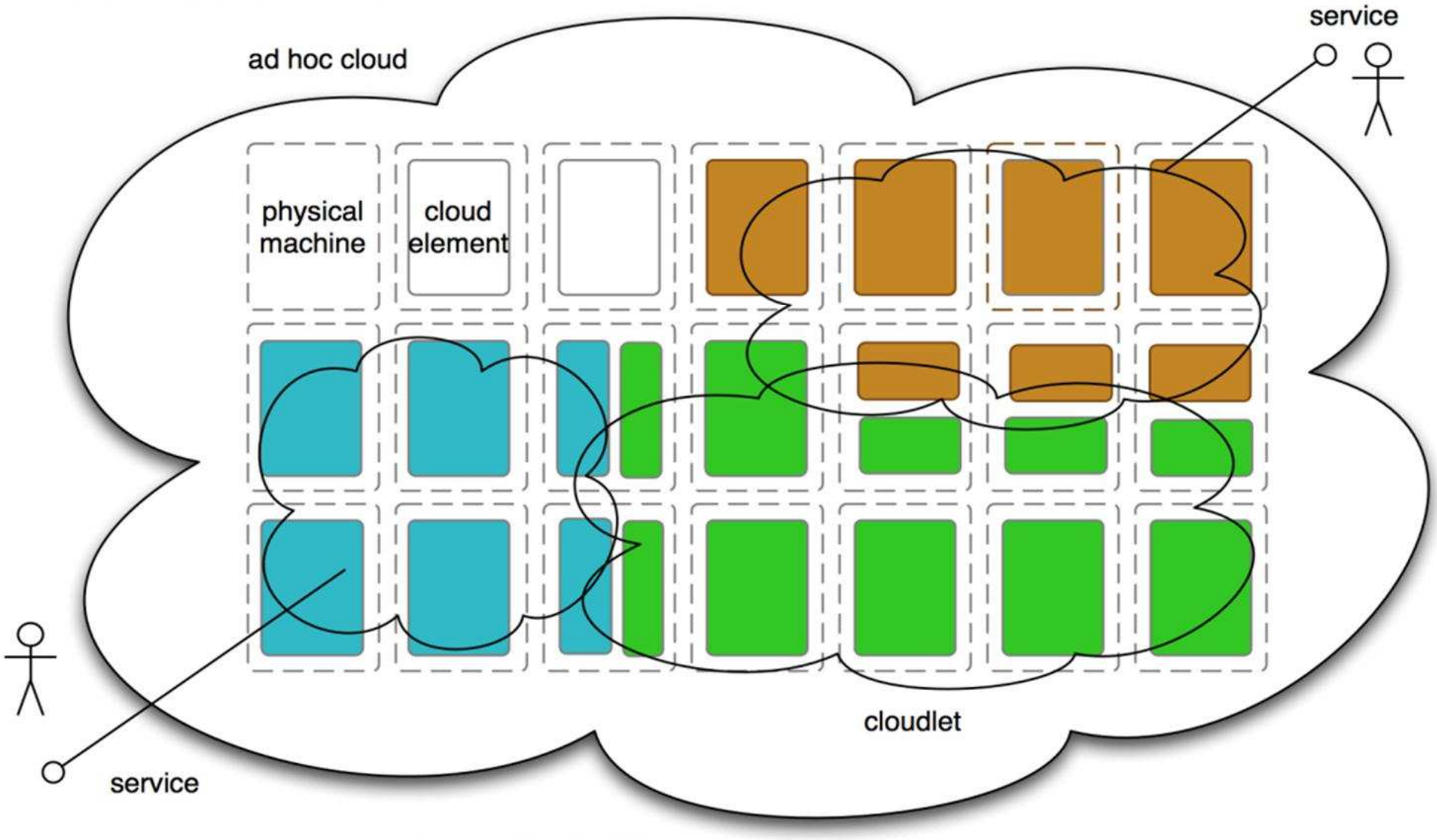
Volunteer User

A user of a volunteer computing infrastructure.

Volunteer Host

a host whose resources are donated to a volunteer computing infrastructure under the instruction of a volunteer user.

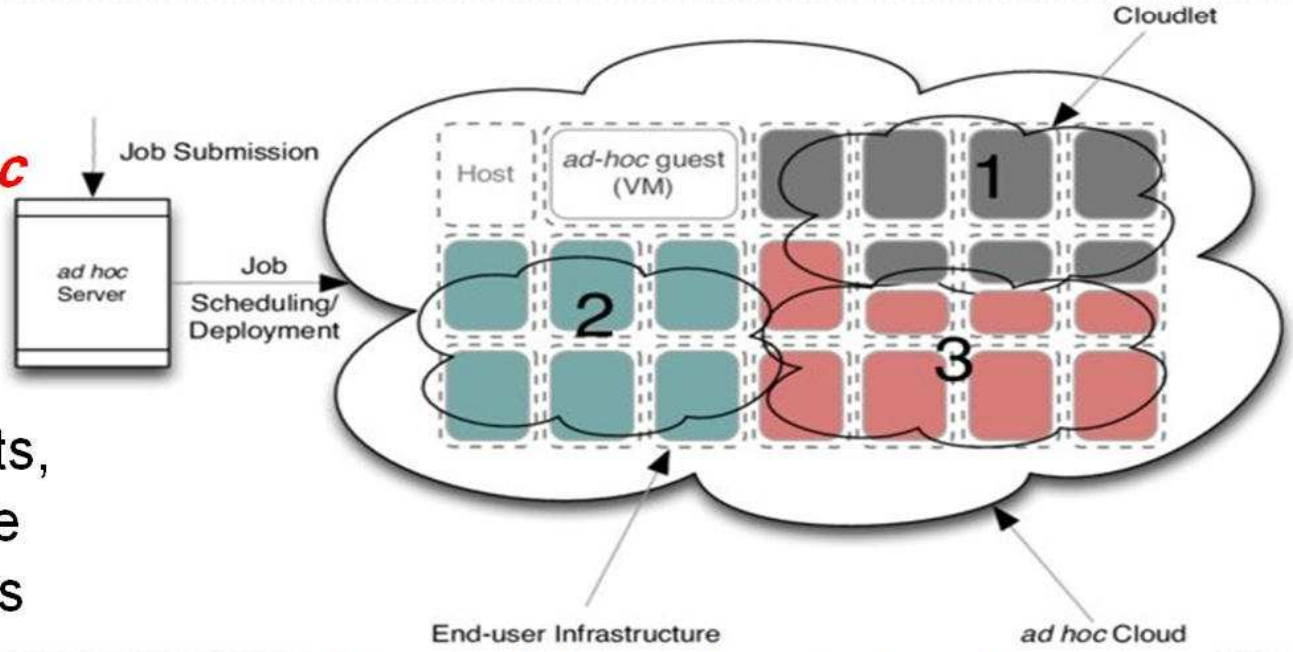
Cloudlets and Cloud Elements of an Ad Hoc Cloud



Ad hoc Cloud Computing-Cont..

- For instance, ***cloudlet '1'*** and ***cloudlet '2'*** may hold the necessary environments for ***Matlab*** and ***BLAST*** applications to execute respectively.
- ***Guest*** may either be dedicated to a ***single cloudlet***, or part of many if ***multiple jobs*** are ***scheduled*** onto the ***same guest*** and ***require different operating environments***.

- ***Cloudlets***, ***hosts*** and ***guests*** are all centrally managed via the ***ad hoc server***.
- The ***ad hoc*** server ***accepts*** Cloud jobs, ***schedules*** jobs to hosts, ***maintains*** system state and ***ensures*** cloud jobs successfully complete.



The *Ad hoc* Cloud Architecture

Ad hoc Cloud Processes

1. **A cloud user** submits a job (application and/or data) to the **ad hoc cloud server** via a web interface.
2. First, **hosts** which are instructed to ***donate*** their machines to **ad hoc platform** connects to **ad hoc server**. A **VM** is sent to hosts' machines and they await further instructions.
3. **Ad hoc server** schedules a **job** to a **host** for execution based on host reliability.
4. Upon a job arriving, the **host** starts **VM** and job execution begins.

Ad hoc Cloud Processes- Cont..

4) The installed ***ad hoc client*** software in each host dynamically ***controls*** **VM** resources to minimize cloud job interference with executing host processes. Furthermore the ***ad hoc client monitors*** the resource loads of both **host** and **guest** and ***passes*** this information to the **ad hoc cloud server** on a regular basis.

5) Periodically, ***ad hoc client*** informs ***ad hoc server*** of **host** and **VM** availability to indicate that they have not failed.

6) Periodically. ***ad hoc client*** captures VM snapshots and distributes these in a Peer-to-Peer (P2P) fashion to other reliable and available hosts within the platform;

- Only the most recent snapshot is stored to minimize disk consumption.

Volunteer Computing Middleware

- Integrating ***Volunteer Computing*** and ***virtualization*** is the ***backbone*** of the ***ad hoc cloud***.
- An open source client-server middleware platform could be used
  - Virtualized **B**erkeley **O**pen **I**nfrastructure for **N**etwork **C**omputing (**BOINC**) as an example

References

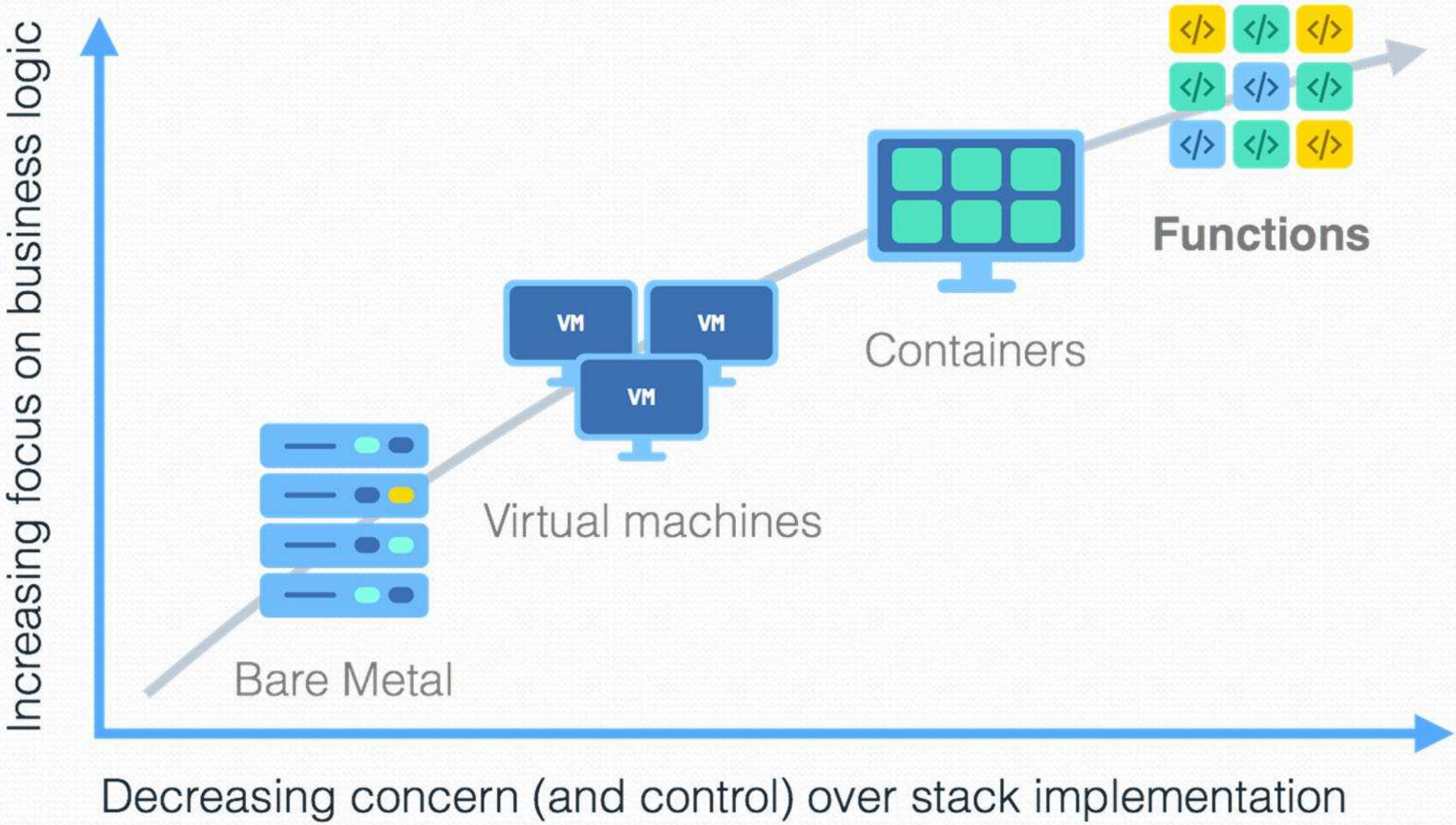
1) https://arxiv.org/abs/1505.08097
2) https://www.researchgate.net/publication/308868099_Ad_Hoc_Cloud_Computing
3) https://ieeexplore.ieee.org/document/7214163
4) http://research.nesc.ac.uk/files/Gary_McGilvary_Thesis.pdf
5) https://arxiv.org/abs/1002.4738

Serverless Computing

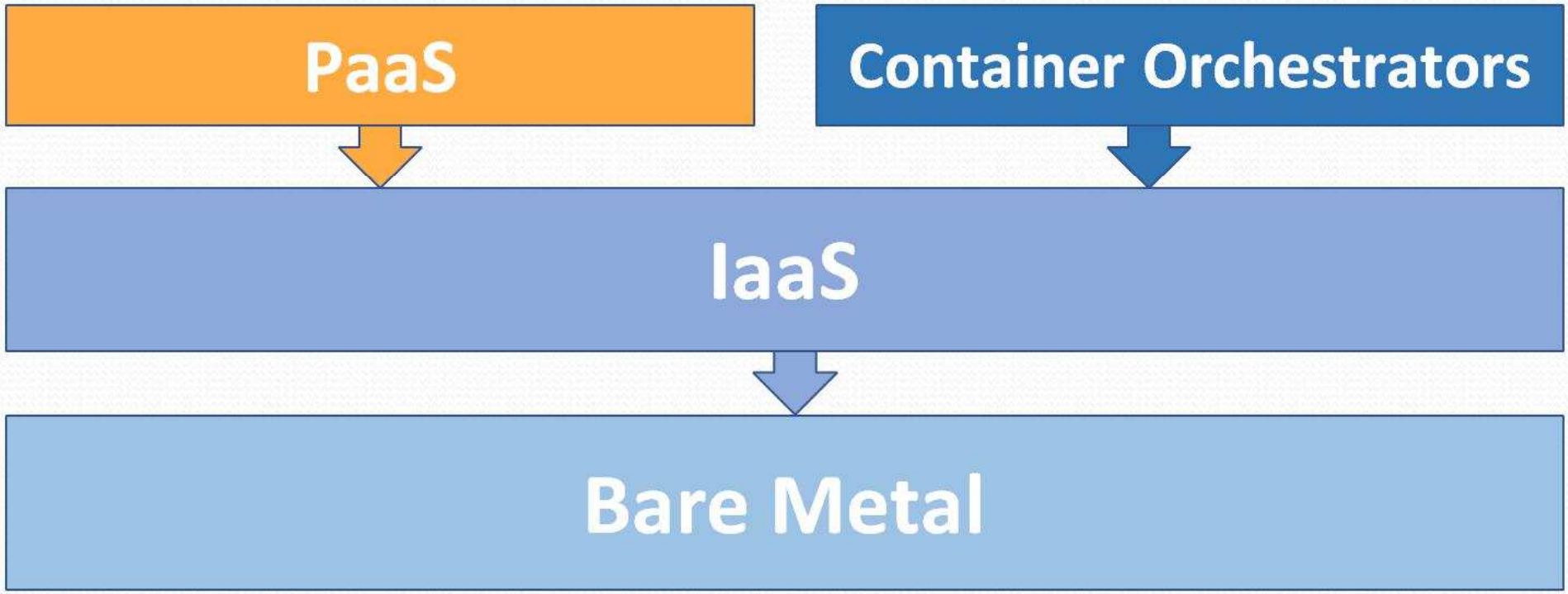
Prepared By: Prof. Fatma Omara

Materials from different sites and papers

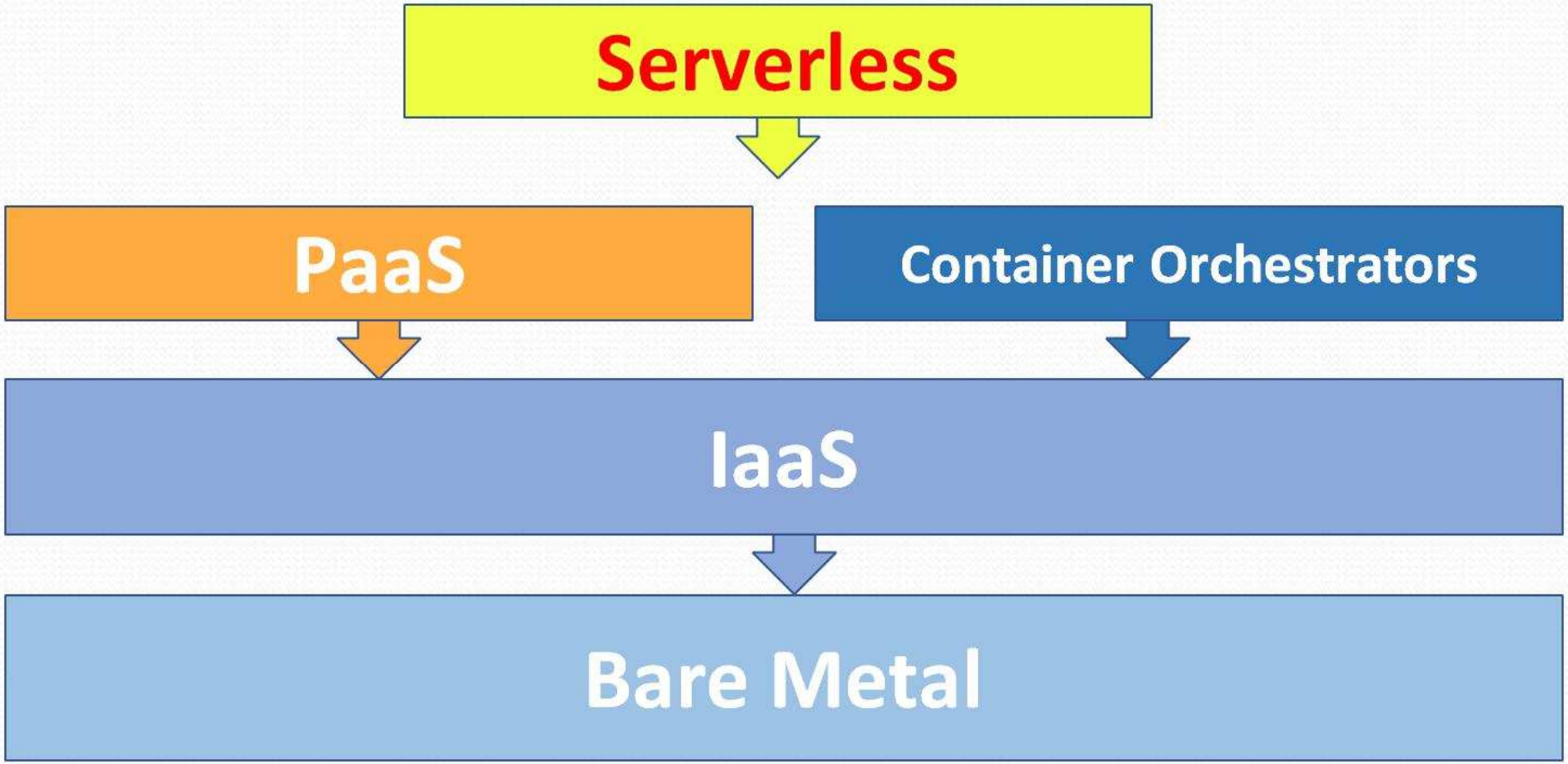
Evolution of Serverless



Evolution of Serverless



Evolution of Serverless

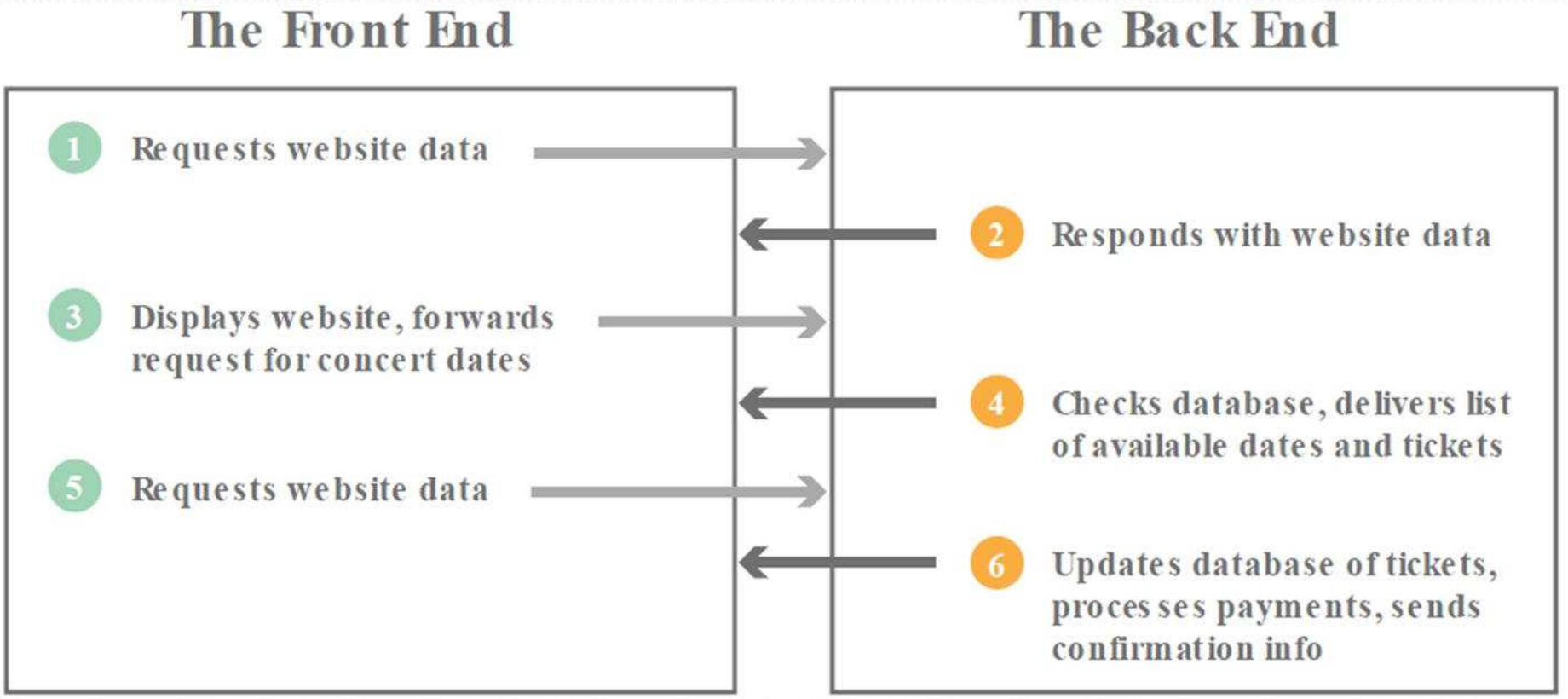


Serverless Computing

- Serverless computing is a ***type*** of ***cloud computing model,*** allows ***users*** to ***write*** and ***deploy code***, ***build web applications***, and ***perform*** a ***range*** of ***other tasks***, ***without*** the ***need*** to ***provision*** or ***manage*** any ***physical servers***.
- Serverless Computing **does not imply** that **servers** are not **involved**, but **servers** are utilize with **administered less, i.e.,**
  - **Servers operate** and **maintain** by ***third-party providers***.
    - It **enables developers** to **deploy** and **execute** their **code** in the Cloud **without needing** to **worry** about **underlying infrastructure** they're working on, which is completely fine.
- The ***service*** is ***auto-scaling***
  - No need to reserve and pay for a fixed number of servers or amount of bandwidth. Instead,
    - **Service provider *allocates backend services*** as they're required.

Serverless Computing-Cont..

- Usually, Application Development is typically split into two main sections; ***frontend*** and ***backend***.
- **Ex:** Web Based Application



Serverless Computing-Cont..

- ***Serverless computing*** provides ***backend services*** using a ***serverless platform***.
  - ***Serverless platform*** is an interface through which **developers** can **build**, **deploy**, and **run applications**, **without having to worry** about the **underlying infrastructure** that's powering them.
    - ***Serverless platform*** offers **features** and **tools** such as **Functions as a Service (FaaS)**, which allow for code to be triggered in response to specific, predetermined events.
  - The ***user*** of a ***serverless computing service*** is ***charged*** based on ***their computation only***.
  - In principle, **Serverless** is similar to a **cloud based phone system**, where most of the associated hardware is stored off-site and maintained by a ***third-party***.

Cloud Based Phone System

- **Cloud Based Phone System** is a phone service that **allows you to make calls over the internet**
- **Cloud Phones** are hosted in one or more **off-site secure data centers**.

Functions as a service (FaaS)

- ***Function as a service (FaaS)*** is a ***cloud computing model*** that runs code in *small modular pieces* ,or *microservices*, in response to events.
- ***FaaS*** enables developers to ***create*** and ***run*** **a single function** in the ***cloud*** using a ***serverless compute model***.
- ***Cloud providers*** then ***manage*** physical hardware, virtual machines, and web server software management.
- ***FaaS*** lets ***developers write*** and ***update*** a ***piece*** of ***code*** on the fly, which can be ***executed*** in response to an ***event***, as a user ***clicking*** on an ***element*** in a web application
- Examples of FaaS are databases (SQL and NoSQL), and storage (particularly object storage)
- ***Compagnies Offer Function-as-a-service:***
  - Alibaba Cloud Function Compute, AWS Lambda, Google Cloud Functions, IBM Cloud Functions based on Apache OpenWhisk, Microsoft Azure Functions, Oracle Cloud Functions.

How Serverless Computing Work

- In practice, Serverless computing works :

1. Developers write code, and deploy it to their ***cloud provider***.
2. This code is then packaged by the ***Cloud Provider***, and deployed to a **fleet of servers**.
3. Upon a request being made to execute the code, the ***Cloud Provider*** will create a ***new container*** to run the code in, which is then destroyed when the execution has completed.

- Using this system, developers only need pay for the time in which their ***code is executing***.

Generally,

Serverless is a cloud-native platform For:

Short-running, Stateless computation

and

event-driven applications

which

scales up and down instantly and automatically

and

charges for actual usage at a millisecond granularity

Current Platforms for Serverless

AWS
Lambda

OpenLambda

Azure Functions

Kubernetes

IBM Cloud Functions

Red-Hat

Google Functions

Advantages of Serverless Computing

Lower Costs

- Serverless Computing is charged on an event-based model.
  - Service providers charge developers for their code execution time
  - Eliminating needs of **provision**, **manage**, and **maintain physical servers**, (so, save a great deal of money).

Increase Productivity

- No need to devote time to **maintaining hardware**,
  - Allows development teams focus on actually developing.
- Server-side applications rely on functions, or series of functions, decided by provider's infrastructure, So,
  - Developers need to upload a few pieces of code and run the program.

Advantages of Serverless Computing

Greater Scalability

- Using serverless computing, developers can quickly scale their operation up or down depending on current demand.
  - If demand for functions increases, the provider's servers adjust, and provide higher capacity to run the functions.
- There's no limitation based on the storage or performance capabilities of the server.

Greener Computing

- Server less computing is considered to be a greener technology
- Resource utilization is improved because resources are only **used when they're needed to execute code**. In addition, energy saving

Serverless Computing Disadvantages

Possible Performance Issues

- Serverless computing can be ***prone to performance issues*** in some scenarios. If an application is reactivated after a prolonged period of non-use, it can experience a **'Cold Start'**.
  - It arises from function **starting up slowly**, and the serverless infrastructure requiring some time to process the server request. This can lead to **slower performance**.
  - To limit cold starts occurrence, keeping *strings of code short*
    - However, this has the adverse effect of creating a higher number of smaller functions to manage, which can be inconvenient.

Serverless Computing Disadvantages

Increased Complexity

- Many serverless architectures operate on a multi-tenancy model, so multiple different software programs for multiple different clients may be running on servers ***simultaneously***.
  - This leads to issues of ***low performance***, or ***security risks*** arising from customers being able to access one another's data.
- Serverless computing is **unsuitable** for **long-running workloads** Because
  - Serverless solutions charge based on the amount of time that code is being run, applications that require lengthy processes may be more costly than if they were run on dedicated servers.

Serverless Computing Disadvantages

Complicated Testing & Debugging

- The nature of serverless computing results in developers having a ***lack of backend visibility***, which can make testing and debugging quite challenging.
- Developers using serverless computing infrastructure should take **additional steps** in order to **recognize**, **predict**, and **plan** for **faults**.
  - This helps to limit the interruptions of services for users.

Serverless is good for

- short-running
- Stateless
- event-driven

- Microservices
- Mobile Backends
- Bots, ML Inferencing
- IoT
- Modest Stream Processing
- Service integration

Serverless is not good for

- long-running
- stateful
- number crunching

- Databases
- Deep Learning Training
- Heavy-Duty Stream Analytics
- Numerical Simulation
- Video Streaming

Serverless Computing Vrs Cloud Computing

- ***Platform as a service*** (***PaaS***), ***containers*** and ***virtual machines*** (***VMs***) all play a **critical role** in the cloud application development and compute ecosystem
- ***Serverless*** compares to ***Cloud*** across some key attributes:
  - **Provisioning Time:** Measured in ***milliseconds*** for serverless, vs. ***minutes*** to ***hours*** for the other models.
  - **Administrative burden:** ***None*** for serverless, compared to a continuum from light to medium to heavy for PaaS, containers and VMs respectively.
  - **Maintenance:** Serverless architectures are ***managed 100%*** by the ***provider***. The ***same*** is ***true*** for ***PaaS***, but ***containers*** and ***VMs*** require significant maintenance including updating and managing operating systems, container images, connections, and so on.

Serverless and Microservices

- Given its unique *combination* of attributes and benefits, **serverless** architecture *works best* on *use cases* around ***microservices***, ***mobile backends*** and ***data*** and ***event stream processing***.
- The ***most common use case*** of ***serverless*** today is ***supporting microservices architectures***.
- Serverless gained significant momentum given its attributes around **small bits of code**, inherent **automatic scaling**, **rapid provisioning** and a **pricing model** that **never charges** for **idle capacity**.

References

1. https://www.codemotion.com/magazine/devops/cloud/serverless-computing-the-advantages-and-disadvantages/
2. https://www.ibm.com/topics/serverless
3. https://www.cloudflare.com/learning/serverless/what-is-serverless/
4. https://www.google.com/search?q=functions+as+a+sfunctions+as+a+service+(FaaS)&rlz=1C1GGRV_enEG751EG755&oq=functions+as+a+sfunctions+as+a+service+(FaaS)+&aqs=chrome..69i57j33i10i160l5.13888j0j7&sourceid=chrome&ie=UTF-8