**CH6: Ad Hoc Cloud Computing**

﻿Current Cloud Computing is based on data centers. There are billions of underutilized PCs, used only for a few hours per day, owned by individuals and organizations worldwide. The unused Pcs can be consolidated as alternative cloud fabrics to provision broad cloud services, primarily Iaas.

This is called "no data center" approach, complements data center-based cloud provision model.

**ad hoc Cloud within an enterprise**: harness unused resources to improve overall utilization, reduce energy consumption, allow organizations to operate 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 University machines to represent computing resource.

-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.

-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.

﻿**Non-exclusive Infrastructure:** infrastructure whose hosts are reserved for some primary purpose.

**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)**.**

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

**ad hoc server:** accepts Cloud jobs, schedules jobs to hosts, maintains system state and ensures cloud jobs successfully complete.

**Ad hoc Cloud Processes**

1. A cloud user submits a job to the ad hoc cloud server via a web interface.

2. First, hosts which donate their machines 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.

﻿5. 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.

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

7. Periodically. ad hoc client captures VM snapshots and distributes these in a Peer-to-Peer fashion to other available hosts within the platform. Only the most recent snapshot is stored.

**Virtualized Berkeley Open Infrastructure for Network Computing (BOINC):** An open-source client-server middleware platform

**A diagram of a serverless process

Description automatically generatedEvolution of serverless**

**Bare metal VM containers Functions**

**﻿Serverless Computing**

type of cloud computing model, allows users to write and deploy code, build web applications, and perform other tasks, without need to provision or manage any physical servers.

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.

-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.

-provides backend services using a serverless platform.

**Serverless platform**: an interface through which developers can build, deploy, and run applications, without having to worry about the underlying infrastructure that's powering them.

-The serverless platform offers features and tools such as: **Functions as a Service (FaaS)**

-The user of a serverless computing service is charged based on their computation only.

-Serverless is like 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**: a phone service that allows you to make calls over the internet.

﻿**Function as a service (FaaS)**: 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.

-providers 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, Microsoft Azure Functions, Oracle Cloud Functions.

**How Serverless Computing Work**

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.

﻿**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.

**Advantages of Serverless Computing**

**Lower Costs:** charged on an event-based model. Developers are charged for code execution time.

Eliminating needs of provision, manage, and maintain physical servers, (so, save a lot of money).

**Increase Productivity:** No need to devote time to maintaining hardware, allows developers to focus on developing. Server-side applications rely on functions, decided by provider's infrastructure, So, Developers need to upload a few pieces of code and run the program.

﻿**Greater Scalability**

**Greener Computing:** Resource utilization is improved as resources are only used when needed.

﻿**Serverless Computing Disadvantages**

**Possible Performance Issues:** If an app is reactivated after a long period of non-use, it can experience a 'Cold Start'.It arises from starting up slowly, and the infrastructure requires time to process the server request. This can lead to slower performance.

-To limit cold starts, keep strings of code short However, this has the adverse effect of creating a higher number of smaller functions to manage, which can be inconvenient.

**Increased Complexity:** 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 they charge based on the amount of time that code is being run, applications with lengthy processes may be more costly.

﻿**Complicated Testing & Debugging**

-Serverless computing results in having a lack of backend visibility, which can make testing and debugging quite challenging. Extra steps taken in order to recognize, predict, and plan for faults.

﻿**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**

**Provisioning Time**: Measured in milliseconds for serverless, vs. mins 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 (updating and managing OS).

-Serverless architecture works best on use cases around microservices, mobile backends and data and event stream processing.

-Serverless gained significant momentum given its attributes around small bits of code, automatic scaling, rapid provisioning, and a pricing model.