Cloud Programming Simplified

Electrical Engineering and Computer Sciences
University of California at Berkeley

- **Presented by Kirtan**

Overview

18 Slides

Introduction - Berkeley Paper

*** In 2009, to help explain the excitement around cloud computing, “The Berkeley View on Cloud Computing” identified six potential advantages.**

1. The appearance of infinite computing resources on demand.

2. The elimination of an up-front commitment by cloud users.

3. The ability to pay for use of computing resources on a short-term basis as needed.

4. Economies of scale that significantly reduced cost due to many, very large data centers.

Introduction - Berkeley Paper

5. Simplifying operation and increasing utilization via resource virtualization.

6. Higher hardware utilization by multiplexing workloads from different organizations.

However, cloud users continue to bear a burden from complex operations and many workloads still do not benefit from efficient multiplexing.

This is mainly due to the failure to realize the last two potential advantages.

Introduction - Two different Cloud Approaches

Two approaches to cloud: Amazon EC2 vs Google App Engine.

Users embraced Amazon EC2 approach.

But developers had to manage virtual machines themselves.

Introduction - Issue in setting up an environment.

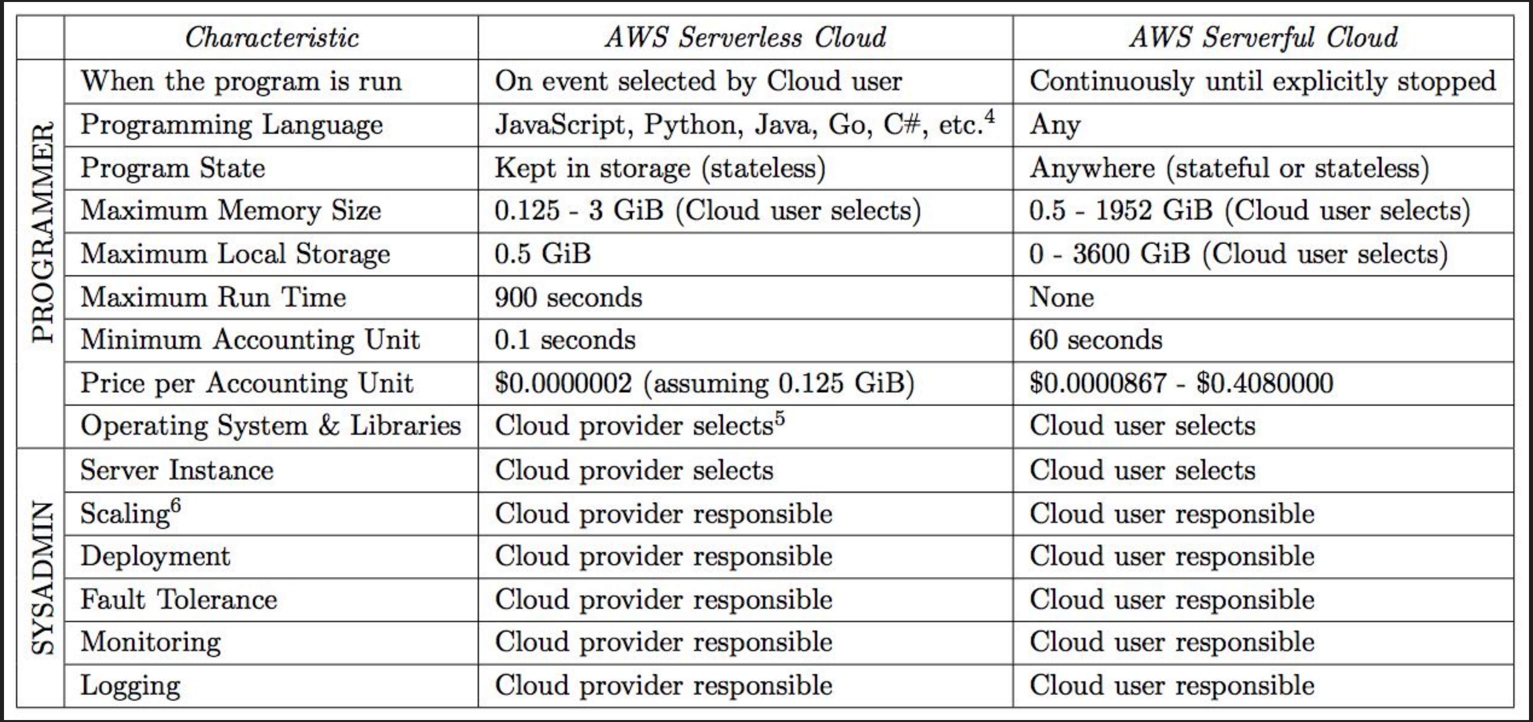
1. Redundancy for availability, so that a single machine failure doesn’t take down the service.
2. Geographic distribution of redundant copies to preserve the service in case of disaster.
3. Load balancing and request routing to efficiently utilize resources.
4. Autoscaling in response to changes in load to scale up or down the system.
5. Monitoring to make sure the service is still running well.
6. Logging to record messages needed for debugging or performance tuning.
7. System upgrades, including security patching.
8. Migration to new instances as they become available.

Introduction - Need for FaaS

1. Is it worth maintaining cloud infra for simple applications/extensions? For example; a simple mobile application to upload image files.
2. AWS recognised this need and developed Lamda (Cloud Function or FaaS).
3. Also, cloud platforms provide specialized serverless frameworks that cater to specific application requirements as BaaS. (S3)

Serverless Computing = FaaS + BaaS

Emergence of Serverless Computing



	Characteristic	*AWS Serverless Cloud*	*AWS Serverful Cloud*
PROGRAMMER	When the program is run	On event selected by Cloud user	Continuously until explicitly stopped
	Programming Language	JavaScript, Python, Java, Go, C#, etc.[4]	Any
	Program State	Kept in storage (stateless)	Anywhere (stateful or stateless)
	Maximum Memory Size	0.125 - 3 GiB (Cloud user selects)	0.5 - 1952 GiB (Cloud user selects)
	Maximum Local Storage	0.5 GiB	0 - 3600 GiB (Cloud user selects)
	Maximum Run Time	900 seconds	None
	Minimum Accounting Unit	0.1 seconds	60 seconds
	Price per Accounting Unit	$0.0000002 (assuming 0.125 GiB)	$0.0000867 - $0.4080000
	Operating System & Libraries	Cloud provider selects[5]	Cloud user selects
SYSADMIN	Server Instance	Cloud provider selects	Cloud user selects
	Scaling[6]	Cloud provider responsible	Cloud user responsible
	Deployment	Cloud provider responsible	Cloud user responsible
	Fault Tolerance	Cloud provider responsible	Cloud user responsible
	Monitoring	Cloud provider responsible	Cloud user responsible
	Logging	Cloud provider responsible	Cloud user responsible

Emergence of Serverless Computing

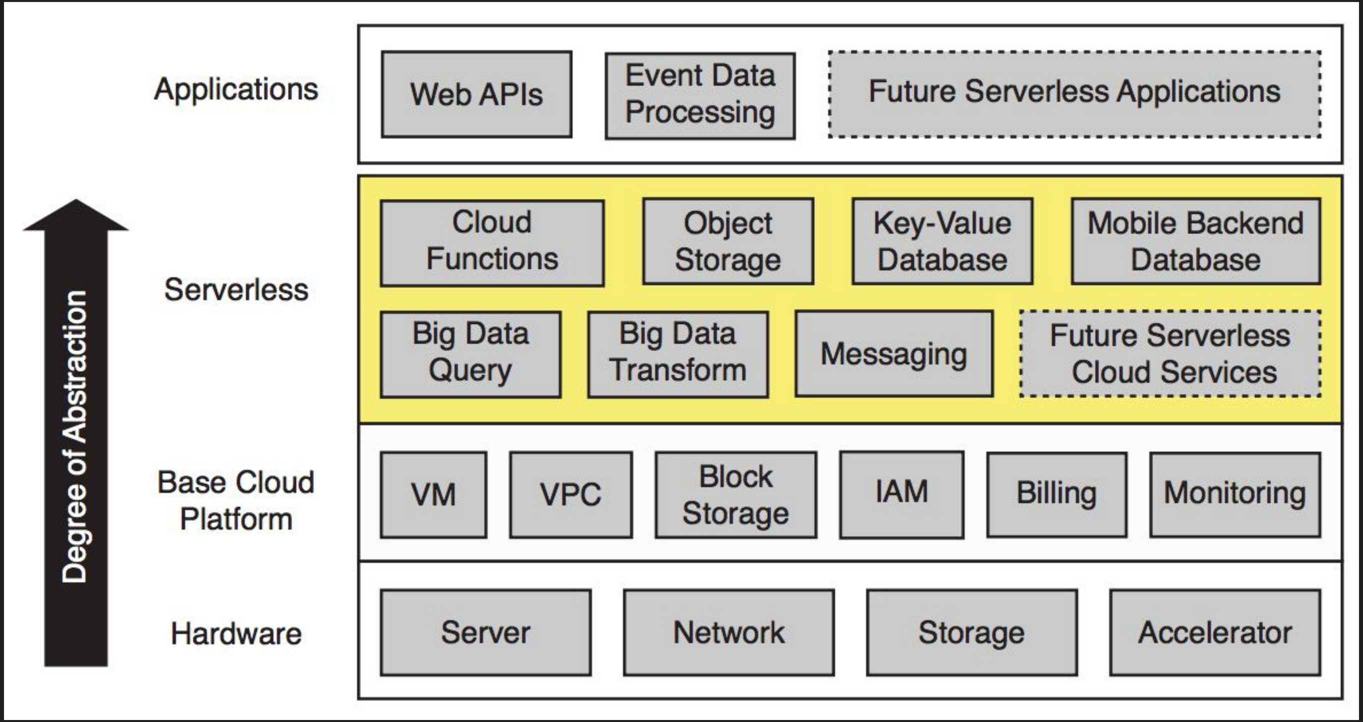
Higher level language : Lower level language

Serverless Computing : Serverful Computing

The three important distinctions between the two,

1. Decoupled computation and storage.
2. Executing code without managing resource allocation.
3. Paying in proportion to resources used instead of for resources allocated.

Emergence of Serverless Computing



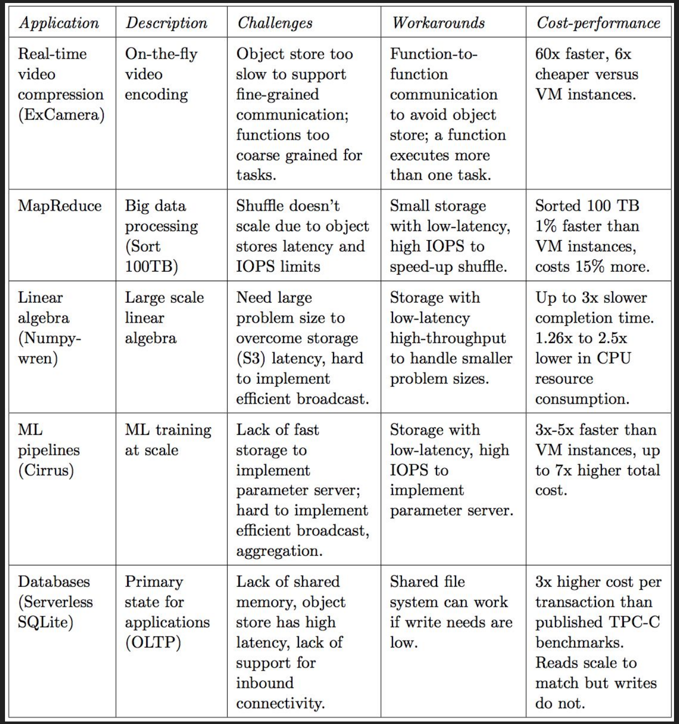
Emergence - Contextualizing Serverless

1. Rebranding of Platform as a Service (PaaS) like Heroku, Firebase, or Parse.
2. Better autoscaling, strong isolation, platform flexibility, and service ecosystem support like AWS Lambda.
3. Strong performance and security isolation to make multi-tenant hardware sharing possible using VM-like isolation.
4. Speed up the creation of function execution environments using “warm pool” of VM instances, unikernels, library OSes, or language VMs.

Emergence - Attractiveness of Serverless

1. Serverless computing promotes business growth.
2. Language oriented optimizations and domain specific architectures.
3. Optimal use of old machines.
4. Customers benefit from increased programming productivity.
5. Cloud users like serverless computing because novices can deploy functions

Limitations of Serverless Computing



Application	*Description*	*Challenges*	*Workarounds*	*Cost-performance*
Real-time video compression (ExCamera)	On-the-fly video encoding	Object store too slow to support fine-grained communication; functions too coarse grained for tasks.	Function-to-function communication to avoid object store; a function executes more than one task.	60x faster, 6x cheaper versus VM instances.
MapReduce	Big data processing (Sort 100TB)	Shuffle doesn't scale due to object stores latency and IOPS limits	Small storage with low-latency, high IOPS to speed-up shuffle.	Sorted 100 TB 1% faster than VM instances, costs 15% more.
Linear algebra (Numpy-wren)	Large scale linear algebra	Need large problem size to overcome storage (S3) latency, hard to implement efficient broadcast.	Storage with low-latency high-throughput to handle smaller problem sizes.	Up to 3x slower completion time. 1.26x to 2.5x lower in CPU resource consumption.
ML pipelines (Cirrus)	ML training at scale	Lack of fast storage to implement parameter server; hard to implement efficient broadcast, aggregation.	Storage with low-latency, high IOPS to implement parameter server.	3x-5x faster than VM instances, up to 7x higher total cost.
Databases (Serverless SQLite)	Primary state for applications (OLTP)	Lack of shared memory, object store has high latency, lack of support for inbound connectivity.	Shared file system can work if write needs are low.	3x higher cost per transaction than published TPC-C benchmarks. Reads scale to match but writes do not.

Limitations of Serverless Computing

1. Inadequate storage for fine-grained operations (state sharing)
2. Lack of fine-grained coordination (event based)
3. Poor performance for standard communication patterns (sharing data)
4. Predictable Performance (load all application libraries)

What Serverless Computing Should Become

1. Abstraction challenges
   a. Resource requirements (number of CPUs, GPUs)
   b. Data dependencies (computation graph)
2. System challenges
   a. High-performance, affordable, transparently provisioned storage (distributed in-memory service)
   b. Coordination/signaling service (Cloud functions are not individually addressable)
   c. Minimize startup time (cloud function resources, application software environment, application-specific startup)
3. Networking challenges (larger number of cores, computation graphs)
4. Security challenges
   a. Scheduling randomization and physical isolation (due to physical co-residency)
   b. Fine-grained security contexts (information flow control)
   c. Oblivious serverless computing (oblivious algorithms)
5. Computer architecture challenges
   a. Hardware Heterogeneity, Pricing, and Ease of Management (Lang Specific, Domain Specific)

Fallacies and Pitfalls

Fallacy- Serverless cloud computing is more expensive than serverful cloud computing.

Pitfall- Serverless computing can have unpredictable costs.

Fallacy- It’s easy to port applications between serverless computing providers.

Pitfall- Vendor lock-in may be stronger with serverless computing than for serverful computing.

Fallacy- Cloud functions cannot handle very low latency applications needing predictable performance. (pre-warm cloud functions)

Pitfall- Few so called “elastic” services match the real flexibility demands of serverless computing. (scale to zero)

Predictions

1. New BaaS storage services to be created that expand the types of applications that run well on serverless computing.
2. Serverless computing will become simpler to program securely than serverful computing.
3. Almost any application, running at almost any scale, will cost no more and perhaps much less with serverless computing.
4. The future of serverful computing will be to facilitate BaaS. (OLTP, Queues)
5. Serverful cloud computing won’t disappear, but the relative importance of that portion of the cloud will decline.
6. Serverless computing will become the default computing paradigm of the Cloud Era.

End.

Thank you.