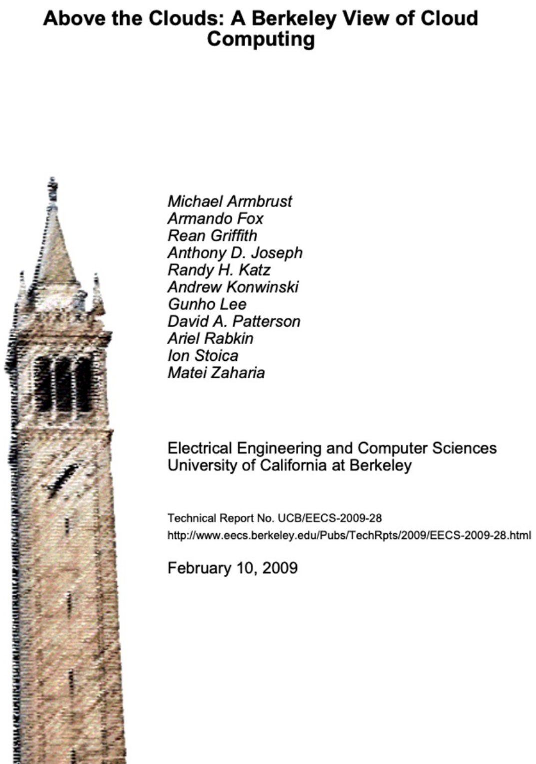
Cloud Programming Simplified: A Berkeley View on Serverless Computing

Varan Shukla

Background



Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust
Armando Fox
Rean Griffith
Anthony D. Joseph
Randy H. Katz
Andrew Konwinski
Gunho Lee
David A. Patterson
Ariel Rabkin
Ion Stoica
Matei Zaharia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-28
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

February 10, 2009

- Obstacles and Research Opportunities
- Advantages:
  - Infinite Compute Resources
  - Eliminate up-front commitment
  - Pay for use
  - Economy of scale
  - Simplify Operations
  - Increase utilization by multiplexing

➔ Proliferation of virtual resources to manage

Introduction

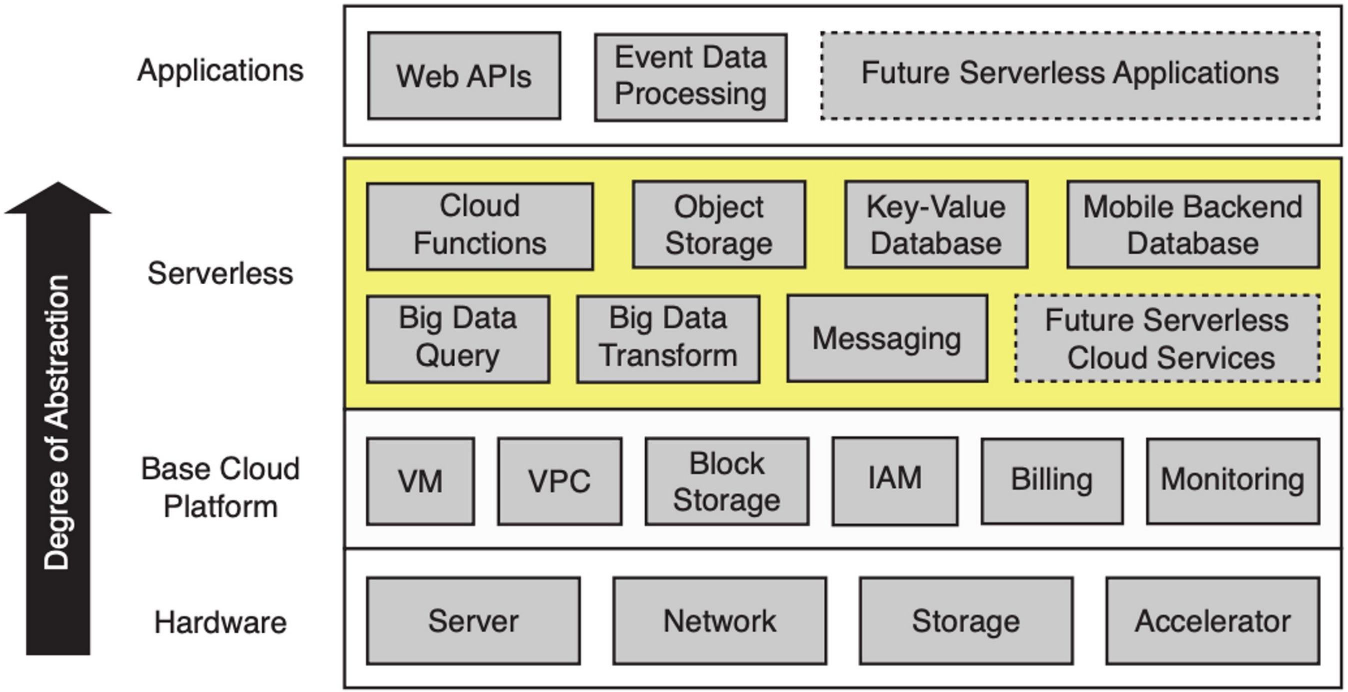
- 2 competing approaches: EC2 vs App Engine
- Why EC2 won?
    - Simple to port
- Customers still need to manage the virtual resources themselves.
    - Is there an easier path?
- AWS Lambda: cloud functions and serverless(an oxymoron)
- BaaS: Backend as a Service
    - specialized frameworks for app specific requirements
        - Serverless = Faas + Baas
        - Autoscale, billed on actual usage

Serverful vs Serverless

Parallels Low-level assembly vs High-level Programming

	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

Serverful vs Serverless



Key Distinctions

- Decoupled Computation and Storage
  - Stateless Computation and independent scaling
- Execute without managing resource allocation
  - Auto provisioning
- Pay in proportion to actual usage of resource rather than for allocation

Merely a re-branding of previous offerings like PaaS? Nope

- Better autoscaling, strong isolation, platform flexibility & ecosystem support

Autoscaling How?

- Need Strong performance & security isolation
- Warm pool of VM instances
- Leverage Unikernels, library OSes, language based VMs, microVMs: Firecracker

How this relates to Kubernetes?

K8S lies somewhere in between - perfect match to hybrid solutions

Serverless is a paradigm shift - fully offloading operational responsibilities

Why is serverless attractive?

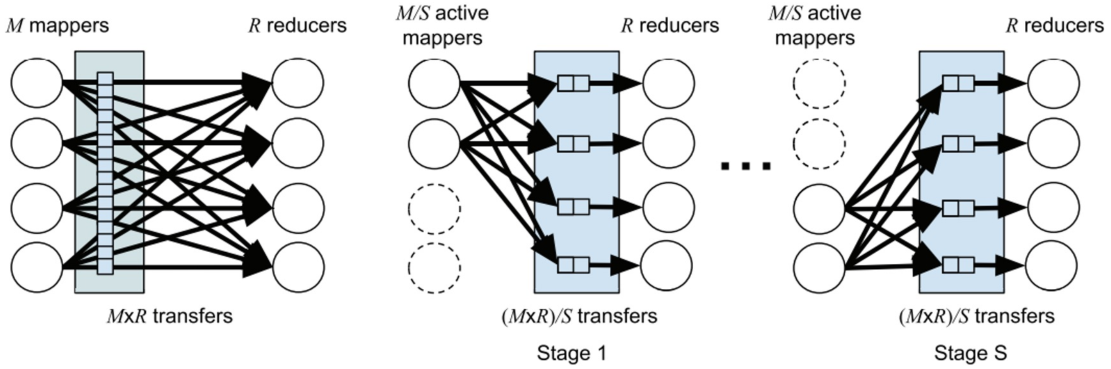
- Draw in new customers - makes cloud approachable & easier
- Utilize unused resources
- Increased programming productivity
- Opportunities for software/hardware optimizations & research
- Fine grained accounting (~100ms)

Limitations

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.

MapReduce: their thoughts

- Shuffle operation is a challenge with M x R transfers
- 100TB of data, 3GB blocks, 33k blocks, 2.22 billion IOPS
- $12,000 in S3 alone
- Solution: Use High performance but much expensive storage(ElastiCache)
- Divide in stages to reduce storage size
- 2983s for $144 on 395 VMs v/s 2945s for $163 using AWS Lambda



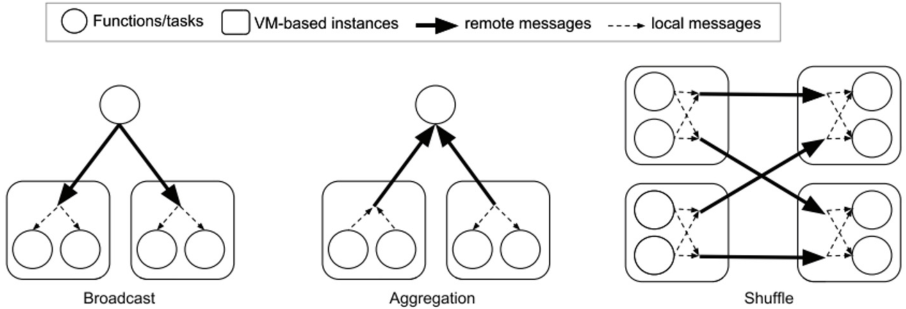
(a) One stage Shuffle (b) Multi-stage Shuffle

Limitations

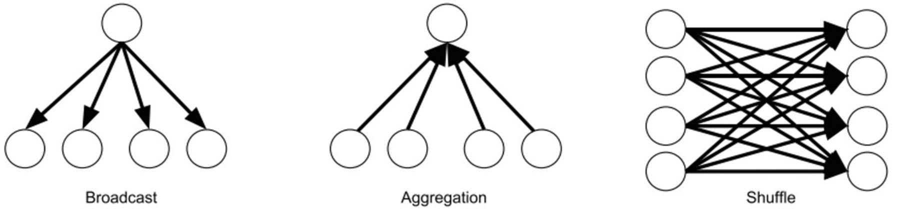
- Inadequate storage for fine-grained operations
  - calls for development of ephemeral and durable storage
- Lack of fine-grained coordination
  - Calls for VM-based rendezvous server/notification systems
  - Name function instances & allow direct addressability to access internal state

Limitations

- Poor performance for standard communication patterns
  - Shuffle is worst with (N x K)^2 messages v/s N^2
  - No control of location
- Predictable performance
  - Cold start latency
  - Variable Hardware Resources



(a) VM-based communication patterns.



(b) Function-based communication patterns.

What Serverless Computing Should Become

Note challenges in 5 areas:

- Abstraction Challenges
  - Resource requirements
    - Explicit control? Against the spirit
    - Instead raise the level of abstraction: infer eg. static code analysis
  - Data Dependencies
    - Suboptimal placement = inefficient communication
    - Specify its computation graph

What Serverless Computing Should Become

- System Challenges
    - High-performance, affordable, transparently provisioned storage
        - Ephemeral Storage - dist. In-memory service leveraging statistical multiplexing
        - Durable Storage - transparently provisioned
    - Coordination/Signaling Service
    - Minimise Startup time
        - Unikernels, warm pools, incremental loading
- Computer Architecture Challenges
    - Hardware Heterogeneity, Pricing, and Ease of Management
        - Hardware-software co-design
        - Domain Specific Architectures(GPUs, TPUs)

What Serverless Computing Should Become

- Networking Challenges
  - K^2 more messages when K functions on a VM
  - Ways to address
    - Combine and share over a single VM
    - Explicit placement
    - Co-locate with computation graph
  - Arguably against the spirit, rescue flexibility for provider
- Security Challenges
  - Scheduling randomization and physical isolation
    - Co-residency attacks are difficult
  - Fine-grained security contexts
  - Oblivious serverless to prevent leaking access patterns

Fallacies and Pitfalls

- More Expensive: could actually end up costing much less
- Unpredictable Costs: bucket based pricing
- Easy to port: not really, need a standard
- Vendor lock in strong: cross-cloud support
- Cannot handle low-latency apps needing predictable performance: not really
- Few “elastic” services are actually as flexible

Predictions

- Serverless skyrockets while serverful although not disappear, will decline
- Expect new BaaS storage services facilitated by serverful
- Will be simpler and more secure
- Billing models will evolve

> - Serverless computing will become the default computing paradigm of the Cloud Era, largely replacing serverful computing and thereby bringing closure to the Client-Server Era.

Thank You

Questions?