Towards Modern Development of Cloud Applications

Sanjay Ghemawat, Robert Grandl, Srdjan Petrovic, Michael Whittaker, Parveen Patel, Ivan Posva, Amin Vahdat
Microservices

Plethora of technologies that help to grow the microservices ecosystem

Opinionated discussions

Breaking a Monolithic API into Microservices at Uber

How and Why Etsy Moved to an API-First Architecture

API Strategies at eBay

Migrating Applications from Monolithic to Microservice on AWS

Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%

The move from a distributed microservices architecture to a monolith application helped achieve higher scale, resilience, and reduce costs.
Microservices Pros and Cons

+ Performance 😊
+ Abstraction 😊
+ Fault Tolerance 😊

- Performance 😞
- Abstraction 😞
- Fault Tolerance 😞
Coupling of Logical and Physical Boundaries

```
// Python
print("Python")

// C++
#include <iostream>
int main()
{
    std::cout << "C++";
    return 0;
}
```

```
// Go
package main
func main()
{
    fmt.Println("go")
}
```
decide to split
add protos
add grpc

update configs
update all callers
remove grpc
remove protos

deploy

update configs
update all callers
remove protos
remove grpc

deploy
decide to merge
Versioning Woes

???

v1

???

v2

???
“About two thirds of update failures are caused by interaction between two software versions that hold incompatible data syntax or semantics assumption.”

*Understanding and Detecting Software Upgrade Failures in Distributed Systems [SOSP’21]*
OUR PROPOSAL

1. Decoupling of Logical and Physical Boundaries
2. Isolated Rollouts
Decoupling of Logical and Physical Boundaries

Write as a monolith.

Programming Framework

Deploy as a set of microservices.

Runtime
PROGRAMMING FRAMEWORK
Programming Framework

Components are...

- the key abstraction.
- like actors.
- long-lived.
- possibly replicated.
- soft-state.
- written using native language constructs.
// Component interface.
type Reverser interface {
    Reverse(string) string
}

Programming Framework
// Component implementation.
type reverser struct {
    weaver.Implements[Reverser]}

func (r *reverser) Reverse(s string) string {
    runes := []rune(s)
    n := len(runes)
    for i := 0; i < n/2; i++ {
        runes[i], runes[n-i-1] = runes[n-i-1], runes[i]
    }
    return string(runes)
}
func main() {
    app := weaver.Init()
    r := weaver.Get[Reverser](app)
    reversed := r.Reverse("!dlroW,olleH")
    fmt.Println(reversed)
}
RUNTIME
Runtime

- type A
- type B
- type C
- type D

Diagram showing relationships between types A, B, C, and D.
2 Isolated Rollouts
Blue Green Rollouts

v1

Load Balancer
Blue Green Rollouts

v1

Load Balancer

v2
Blue Green Rollouts

v1

Load Balancer

v2
Blue Green Rollouts
INNOVATIONS
Innovations: Serialization

```go
type pair struct {
    x, y int32
}

pair {
    0xAAAAAAAA,
    0xAAAAAAAA,
}
```

Protobuf serialization:
```
08 8A D5 AA D5 2A
10 8A D5 AA D5 2A
```

Custom Serialization:
```
AA AA AA AA
AA AA AA AA
```
Innovations: Smart Scaling and Placement

Diagram showing the concept of smart scaling and placement.
Innovation: Testing

```go
type A
func main() {...}
```

```go
type B
func main() {...}
```

```go
type C
func main() {...}
```

test???

---

- unit tests
- failure tests
Innovations: Routing

Fast key-value stores: An idea whose time has come and gone [HotOS’19]
- **Online Boutique** application
- 11 microservices
- C2-medium-8 nodes on GKE
- 10,800 QPS load

<table>
<thead>
<tr>
<th>Metric</th>
<th>Baseline</th>
<th>Prototype (split)</th>
<th>Prototype (merged)</th>
<th>Gains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go code</td>
<td>2647 lines</td>
<td>2117 lines</td>
<td>2117 lines</td>
<td>up to 1.25x</td>
</tr>
<tr>
<td>Autoscaled to</td>
<td>77.7 CPU</td>
<td>27.7 CPU</td>
<td>9.11 CPU</td>
<td>up to 8x</td>
</tr>
<tr>
<td>Median latency</td>
<td>5.47 ms</td>
<td>2.66 ms</td>
<td>0.38 ms</td>
<td>up to 14x</td>
</tr>
<tr>
<td>99p latency</td>
<td>18.87 ms</td>
<td>9.24 ms</td>
<td>2.47 ms</td>
<td>up to 7x</td>
</tr>
</tbody>
</table>
RELATED WORK
Related Work (Actor Frameworks)

- Orleans
- Akka
- Cloudflare Durable Objects
- Ray
- Erlang
- C++ Actor Framework
- ...

Bigger focus on rollouts, versioning, portability, and simplicity.
Related Work (Serverless Functions)

- AWS Lambda
- Cloud Run
- Cloud Functions
- App Engine
- Azure Functions

Easier to integrate multiple services together.
Related Work (Physical and Logical Decoupling)

- Databases
- Data processing systems
- Dataflow systems
- ML training systems
- ...

Same idea, but for serving systems.