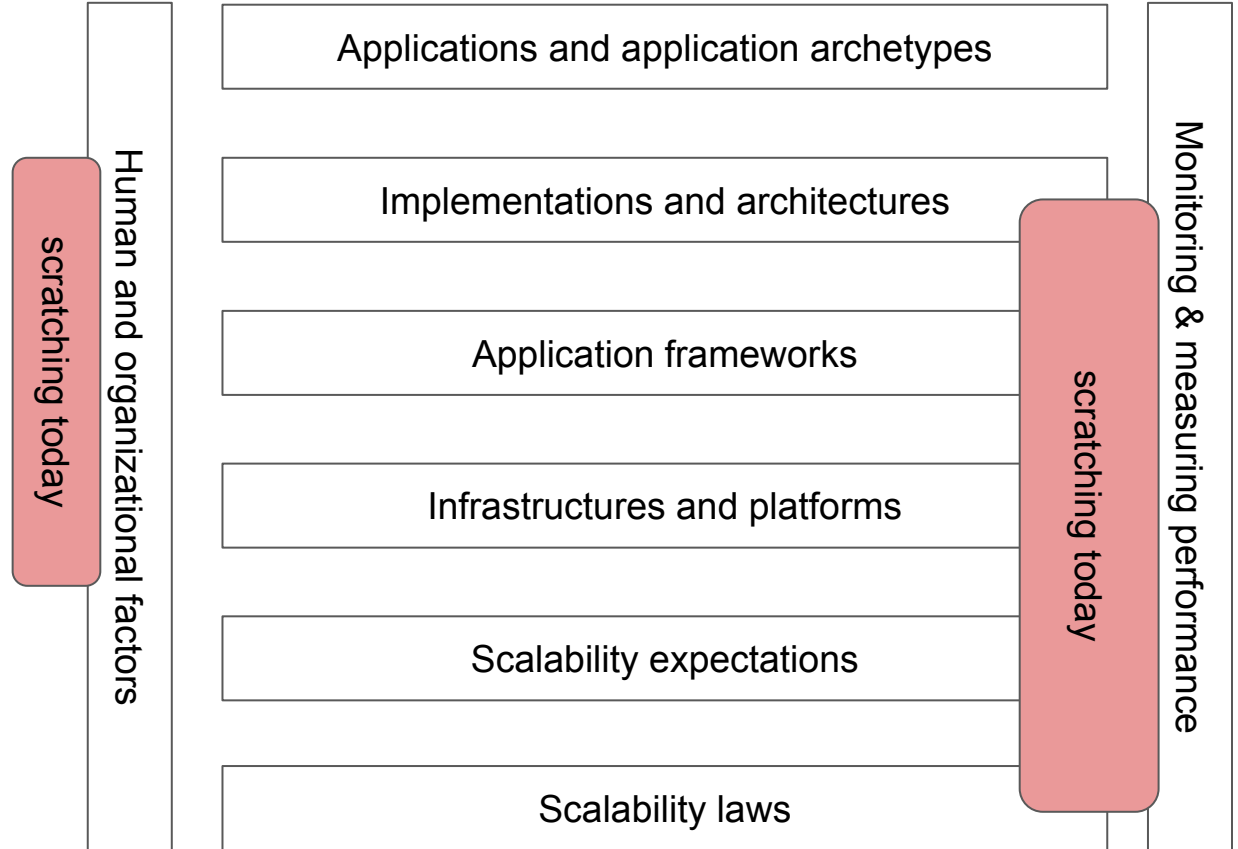


Designing and Building Scalable Web Applications

Lecture 6 / 28.11.2022

The Big Picture



Agenda

- Scalability Laws
- Kubernetes and PostgreSQL
- Scalability and the Cloud
- Example: Pokémon GO
- Scalability and the Cloud: Serverless
- Serverless and Knative
- Beyond software scalability
- Third course project

In distributed systems, linear scalability is not possible in practice due to bottlenecks

Scalability Laws

Scalability Laws

- Amdahl's Law
- Gunther's Universal Scalability Law

In distributed systems, linear scalability is not possible in practice due to bottlenecks

Scalability Laws: Amdahl's Law

Scalability Laws: Amdahl's Law

- A program has parts that can be parallelized and parts that cannot be parallelized

Scalability Laws: Amdahl's Law

program:

Can parallelize

Cannot parallelize

- A program has parts that can be parallelized and parts that cannot be parallelized

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:



- A program has parts that can be parallelized and parts that cannot be parallelized

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:



- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:



- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$\textit{Speedup} = \frac{\text{Time without improvement}}{\text{Time with improvement}}$$

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:

Can parallelize

Cannot parallelize

- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$Speedup = \frac{1}{(1 - p) + p/N}$$

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:



- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$Speedup = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:

Can parallelize

Cannot parallelize

- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$Speedup = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

The number of workers to which the parallel work can be divided to

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

Scalability Laws: Amdahl's Law

program:



- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

5% of work can be parallelized, 10 workers
 $\Rightarrow 1 / ((1 - 0.05) + (0.05 / 10))$
 $\Rightarrow 1 / (0.95 + 0.005)$
 $\Rightarrow 1 / 0.955$
 $\Rightarrow \sim 1.047$

$$Speedup = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

The number of workers to which the parallel work can be divided to

Scalability Laws: Amdahl's Law

- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$Speedup = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

The number of workers to which the parallel work can be divided to

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

program:

Can parallelize

Cannot parallelize

5% of work can be parallelized, 10 workers
 $\Rightarrow 1 / ((1 - 0.05) + (0.05 / 10))$
 $\Rightarrow 1 / (0.95 + 0.005)$
 $\Rightarrow 1 / 0.955$
 $\Rightarrow \sim 1.047$

Up to 4.7% improvement in total time it takes to complete the program

Scalability Laws: Amdahl's Law

- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$\text{Speedup} = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

The number of workers to which the parallel work can be divided to

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

program:

Can parallelize

Cannot parallelize

5% of work can be parallelized, 10 workers
 $\Rightarrow 1 / ((1 - 0.05) + (0.05 / 10))$
 $\Rightarrow 1 / (0.95 + 0.005)$
 $\Rightarrow 1 / 0.955$
 $\Rightarrow \sim 1.047$

Up to 4.7% improvement in total time it takes to complete the program

80% of work can be parallelized, 5 workers
 $\Rightarrow 1 / ((1 - 0.8) + (0.8 / 5))$
 $\Rightarrow 1 / (0.2 + 0.16)$
 $\Rightarrow 1 / 0.316$
 $\Rightarrow \sim 3.165$

Scalability Laws: Amdahl's Law

- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$\text{Speedup} = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

The number of workers to which the parallel work can be divided to

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

program:

Can parallelize

Cannot parallelize

5% of work can be parallelized, 10 workers
 $\Rightarrow 1 / ((1 - 0.05) + (0.05 / 10))$
 $\Rightarrow 1 / (0.95 + 0.005)$
 $\Rightarrow 1 / 0.955$
 $\Rightarrow \sim 1.047$

Up to 4.7% improvement in total time it takes to complete the program

80% of work can be parallelized, 5 workers
 $\Rightarrow 1 / ((1 - 0.8) + (0.8 / 5))$
 $\Rightarrow 1 / (0.2 + 0.16)$
 $\Rightarrow 1 / 0.316$
 $\Rightarrow \sim 3.165$

Up to 216.5% improvement in total time it takes to complete the program

Scalability Laws: Amdahl's Law

- A program has parts that can be parallelized and parts that cannot be parallelized
- Amdahl's Law outlines a maximum theoretical improvement due to parallelization of work

$$\text{Speedup} = \frac{1}{(1 - p) + p/N}$$

Proportion of the program that can be parallelized

The number of workers to which the parallel work can be divided to

Total execution time = execution time of part that can be parallelized + execution time of part that cannot be parallelized

program:

Can parallelize

Cannot parallelize

5% of work can be parallelized, 10 workers
 $\Rightarrow 1 / ((1 - 0.05) + (0.05 / 10))$
 $\Rightarrow 1 / (0.95 + 0.005)$
 $\Rightarrow 1 / 0.955$
 $\Rightarrow \sim 1.047$

Up to 4.7% improvement in total time it takes to complete the program

80% of work can be parallelized, 5 workers
 $\Rightarrow 1 / ((1 - 0.8) + (0.8 / 5))$
 $\Rightarrow 1 / (0.2 + 0.16)$
 $\Rightarrow 1 / 0.316$
 $\Rightarrow \sim 3.165$

Up to 216.5% improvement in total time it takes to complete the program

With an infinite number of workers, maximum speedup: $1 / (1 - p)$

Scalability Laws: Amdahl's Law

- Outlines the theoretical maximum improvement through parallelization of tasks that can be parallelized
- But, does not account for communication between the workers → with more workers, the time needed for communication increases

Scalability Laws: Amdahl's Law

- Outlines the theoretical maximum improvement through parallelization of tasks that can be parallelized
- But, does not account for communication between the workers → with more workers, the time needed for communication increases

*E.g. synchronization
of tasks*



Scalability Laws: Gunther's Universal Scalability Law

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.

$$C(N) = \frac{N}{1 + \sigma(N - 1)}$$

Workers (arrow pointing to N)

Proportion of work that cannot be parallelized. (arrow pointing to σ)

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.

95% of work cannot be parallelized, 10 workers
 $\Rightarrow 10 / (1 + 0.95 * (10 - 1))$
 $\Rightarrow \sim 1.047$

$$C(N) = \frac{N}{1 + \sigma(N - 1)}$$

Workers \leftarrow N

\square

*Proportion of work
that cannot be
parallelized.*

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.
- Gunther's Universal Scalability Law adds communication overhead.

$$C(N) = \frac{N}{1 + \sigma(N - 1)}$$

Workers (arrow pointing to N)

Proportion of work that cannot be parallelized. (arrow pointing to σ)

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.
- Gunther's Universal Scalability Law adds communication overhead.

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \sigma\lambda N(N - 1)}$$

Workers

*Proportion of work
that cannot be
parallelized.*

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.
- Gunther's Universal Scalability Law adds communication overhead.

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \sigma\lambda N(N - 1)}$$

Workers (arrow pointing to N)

Proportion of work that cannot be parallelized. (arrow pointing to $\sigma(N - 1)$)

Communication overhead (e.g. using shared data, invalidating caches, ...) (arrow pointing to $\sigma\lambda N(N - 1)$)

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.
- Gunther's Universal Scalability Law adds communication overhead.

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \sigma\lambda N(N - 1)}$$

Workers

Proportion of work that cannot be parallelized.

Communication overhead (e.g. using shared data, invalidating caches, ...)

Workers passing messages to each others

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.
- Gunther's Universal Scalability Law adds communication overhead.

95% of work cannot be parallelized, 10 workers, 5% communication overhead
 $\Rightarrow 10 / (1 + 0.95 * (10 - 1) + 0.95 * 0.05 * (10 * 9))$
 $\Rightarrow \sim 0.723$
 \Rightarrow leads to a decrease in performance

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \sigma\lambda N(N - 1)}$$

Workers (points to N)

Proportion of work that cannot be parallelized. (points to σ)

Communication overhead (e.g. using shared data, invalidating caches, ...) (points to λ)

Workers passing messages to each others (points to $N(N - 1)$)

Gunther, Neil J. A simple capacity model of massively parallel transaction systems. Int. CMG Conference. 1993.

Scalability Laws: Gunther's Universal Scalability Law

- A peek at Amdahl's Law with different notation.
- Gunther's Universal Scalability Law adds communication overhead.

95% of work cannot be parallelized, 10 workers, 5% communication overhead
 $\Rightarrow 10 / (1 + 0.95 * (10 - 1) + 0.95 * 0.05 * (10 * 9))$
 $\Rightarrow \sim 0.723$
 \Rightarrow leads to a decrease in performance

50% of work cannot be parallelized, 2 workers, 5% communication overhead
 $\Rightarrow 2 / (1 + 0.5 * (2 - 1) + 0.5 * 0.05 * (2 * 1))$
 $\Rightarrow \sim 1.29$
 \Rightarrow leads to an increase in performance

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \sigma\lambda N(N - 1)}$$

Workers

Proportion of work that cannot be parallelized.

Communication overhead (e.g. using shared data, invalidating caches, ...)

Workers passing messages to each others

Scalability Laws

Scalability Laws

- Reminding that scalability is not linear
 - There are parts that cannot be parallelized
 - There is communication overhead

Scalability Laws

- Reminding that scalability is not linear
 - There are parts that cannot be parallelized
 - There is communication overhead
- Reminding that parallelization is not always an answer
 - Parallelization of a system with high communication overhead and high proportion of parts that cannot be parallelized can lead to decrease in performance when compared to a single worker setup

Scalability Laws

- Reminding that scalability is not linear
 - There are parts that cannot be parallelized
 - There is communication overhead
- Reminding that parallelization is not always an answer
 - Parallelization of a system with high communication overhead and high proportion of parts that cannot be parallelized can lead to decrease in performance when compared to a single worker setup

Should not always assume that distributing work will make the work faster (although, in practice, this is most often the case)

Scalability Laws

- Reminding that scalability is not linear
 - There are parts that cannot be parallelized
 - There is communication overhead
- Reminding that parallelization is not always an answer
 - Parallelization of a system with high communication overhead and high proportion of parts that cannot be parallelized can lead to decrease in performance when compared to a single worker setup

Should not always assume that distributing work will make the work faster (although, in practice, this is most often the case)

Disk IO often bottleneck

Scalability Laws

- Reminding that scalability is not linear
 - There are parts that cannot be parallelized
 - There is communication overhead
- Reminding that parallelization is not always an answer
 - Parallelization of a system with high communication overhead and high proportion of parts that cannot be parallelized can lead to decrease in performance when compared to a single worker setup

Should not always assume that distributing work will make the work faster (although, in practice, this is most often the case)

Disk IO often bottleneck

But, even that can be parallelized to some extent

Scalability Laws

- Reminding that scalability is not linear
 - There are parts that cannot be parallelized
 - There is communication overhead
- Reminding that parallelization is not always an answer
 - Parallelization of a system with high communication overhead and high proportion of parts that cannot be parallelized can lead to decrease in performance when compared to a single worker setup

Should not always assume that distributing work will make the work faster (although, in practice, this is most often the case)

Disk IO often bottleneck

But, even that can be parallelized to some extent

Kubernetes and PostgreSQL

Kubernetes and PostgreSQL

- When using Kubernetes and PostgreSQL, hoping to scale the database, we'd use an operator such as Kubegres.
 - There are many operators to choose from; see e.g. <https://blog.palark.com/cloudnativepg-and-other-kubernetes-operators-for-postgresql/>

Kubernetes and PostgreSQL: Example with CloudNative PG

CloudNativePG example

- Install operator:
 - `kubectl apply -f`
<https://raw.githubusercontent.com/cloudnative-pg/cloudnative-pg/release-1.18/releases/cnpg-1.18.0.yaml>
- Deploy a cluster (using slightly modified sample yml from CloudNativePG quickstart – name: pg-cluster-example instead of cluster-example)
 - `kubectl apply -f pg-cluster-example.yaml`
- Login to cluster
 - `kubectl exec -ti pg-cluster-example-1 -- /bin/bash`
- Launch psql
 - `psql`
- Adjust password
 - `ALTER USER postgres WITH password 'postgres';`
- Create a table “names” and add a name
 - `CREATE TABLE names (name VARCHAR(255));`
 - `INSERT INTO names VALUES ('Mickey Mouse');`
- Quit using `\q`, then exit
- Connect to another pod
 - `kubectl exec -ti pg-cluster-example-2 -- /bin/bash`
- List names
 - `SELECT * FROM names;`
- Quit using `\q`, then exit

Note that in reality we'd store passwords etc in a secret file

...and use Flyway or similar for versioning

<https://cloudnative-pg.io/documentation/1.18/quickstart/>

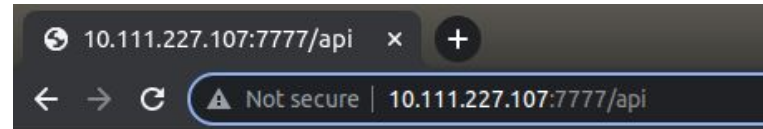
CloudNativePG example

- Adjust application to use the CloudNativePG service
 - apps in same cluster have environment variable `PG_CLUSTER_EXAMPLE_RW_SERVICE_HOST`
- Build image
 - `minikube image build -t names-api .`
- Create deployment (and remember to adjust the image pull policy – alternatively, create using a config file)
 - `kubectl create deployment names-api-app --image=names-api`
- Open tunnel (as root)
 - `minikube tunnel`
- Create a load balancer
 - `kubectl expose deployment names-api-app --type=LoadBalancer --port=7777`
- Find load balancer (external) IP:
 - `kubectl get svc`
- Access server at port

```
import { postgres } from "./deps.js";
const sql = postgres({
  host: Deno.env.get("PG_CLUSTER_EXAMPLE_RW_SERVICE_HOST"),
  database: "postgres",
  username: "postgres",
  password: "postgres"
});
```

Again, in reality we'd store passwords etc in the environment

<https://cloudnative-pg.io/documentation/1.18/quickstart/>



```
[{"name": "Mickey Mouse"}]
```

Note: Kubernetes configuration

- Typically, as separate folder in our project, e.g. kubernetes
 - Separate configuration files
 - Can still (and often do) use docker-compose for local development
- For example, for our names-api-app, could have `names-api-example.yaml` in a folder called `kubernetes`
- Now, deployment using
 - `kubectl apply -f kubernetes/names-api-example.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: names-api-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: names-api-app
  template:
    metadata:
      labels:
        app: names-api-app
    spec:
      containers:
        - name: names-api-app
          image: names-api:latest
          imagePullPolicy: Never
          ports:
            - containerPort: 7777
```

If the application does not need ACID properties (i.e. BASE is fine), sharding with e.g. MongoDB comes out of the box

(mongo is a document database – has document-level ACID)

<https://www.mongodb.com/docs/kubernetes-operator/master/>

Developing with Kubernetes?

Developing with Kubernetes?

- Often, separate docker-compose setup for local development and a separate Kubernetes setup for staging and production
 - For kubernetes setup, see e.g. <https://kustomize.io/>
- Some work on tools that help development
 - See e.g. <https://skaffold.dev/>
- Perhaps still not yet in a situation where all development would happen with Kubernetes – use docker compose (or similar) for local development and Kubernetes for staging and production?

Scalability and the Cloud

Scalability and the Cloud

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Cloud platform as a service

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Cloud application as a service

Cloud platform as a service

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards

Cloud clients

Cloud application as a service

Cloud platform as a service

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards
- E.g. Kubernetes offered as a service by cloud providers
 - Amazon Elastic Kubernetes Service – <https://aws.amazon.com/eks/>
 - Google Kubernetes Engine – <https://cloud.google.com/kubernetes-engine>

Cloud clients

Cloud application as a service

Cloud platform as a service

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards
- E.g. Kubernetes offered as a service by cloud providers
 - Amazon Elastic Kubernetes Service – <https://aws.amazon.com/eks/>
 - Google Kubernetes Engine – <https://cloud.google.com/kubernetes-engine>
- Similarly, cloud providers offer managed databases etc..
 - Amazon databases – <https://aws.amazon.com/products/databases/>
 - Google Cloud databases – <https://cloud.google.com/products/databases>

Cloud clients

Cloud application as a service

Cloud platform as a service

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

- So far, worked with local deployments or deployments on virtual machines.
- Cloud computing services abstract away “layers” leading to infrastructure, platform, and applications as a service.
 - Can e.g. have multi-region application deployments with the click of a button
 - Some services have explicit APIs that one must build on, while others leverage open standards
- E.g. Kubernetes offered as a service by cloud providers
 - Amazon Elastic Kubernetes Service – <https://aws.amazon.com/eks/>
 - Google Kubernetes Engine – <https://cloud.google.com/kubernetes-engine>
- Similarly, cloud providers offer managed databases etc..
 - Amazon databases – <https://aws.amazon.com/products/databases/>
 - Google Cloud databases – <https://cloud.google.com/products/databases>
- Provided services often meeting application-specific demands, e.g. Google Cloud Bigtable can process more than 5 billion requests per second (per Google’s documentation)

Cloud clients

Cloud application as a service

Cloud platform as a service

Cloud infrastructure as a service

Infrastructure as a service

Servers at multiple locations

Scalability and the Cloud

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years
 - Pretty much all software development companies rely on cloud services for deployment

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years
 - Pretty much all software development companies rely on cloud services for deployment
 - Hardware maintained by dedicated service providers, e.g. Amazon, Google, UpCloud, ...

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years
 - Pretty much all software development companies rely on cloud services for deployment
 - Hardware maintained by dedicated service providers, e.g. Amazon, Google, UpCloud, ...
- Cloud platforms have been classically billed using a hourly rate, where the cost depends on the resources

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years
 - Pretty much all software development companies rely on cloud services for deployment
 - Hardware maintained by dedicated service providers, e.g. Amazon, Google, UpCloud, ...
- Cloud platforms have been classically billed using a hourly rate, where the cost depends on the resources
 - Amazon EC2 t4g.nano instance costs 0.0042\$ per hour

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years
 - Pretty much all software development companies rely on cloud services for deployment
 - Hardware maintained by dedicated service providers, e.g. Amazon, Google, UpCloud, ...
- Cloud platforms have been classically billed using a hourly rate, where the cost depends on the resources
 - Amazon EC2 t4g.nano instance costs 0.0042\$ per hour
 - Often need for different services, e.g. computing, database, traffic, ...

Scalability and the Cloud

- Deployment on the cloud is the de-facto standard for software development
 - E.g. Heroku has been around for some 15 years
 - Pretty much all software development companies rely on cloud services for deployment
 - Hardware maintained by dedicated service providers, e.g. Amazon, Google, UpCloud, ...
- Cloud platforms have been classically billed using a hourly rate, where the cost depends on the resources
 - Amazon EC2 t4g.nano instance costs 0.0042\$ per hour
 - Often need for different services, e.g. computing, database, traffic, ...
 - Platforms offer tools for cost estimates, see e.g.
 - <https://cloud.google.com/products/calculator/>
 - <https://calculator.aws/>

Pokémon GO

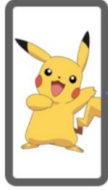
Pokémon Go!



How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!



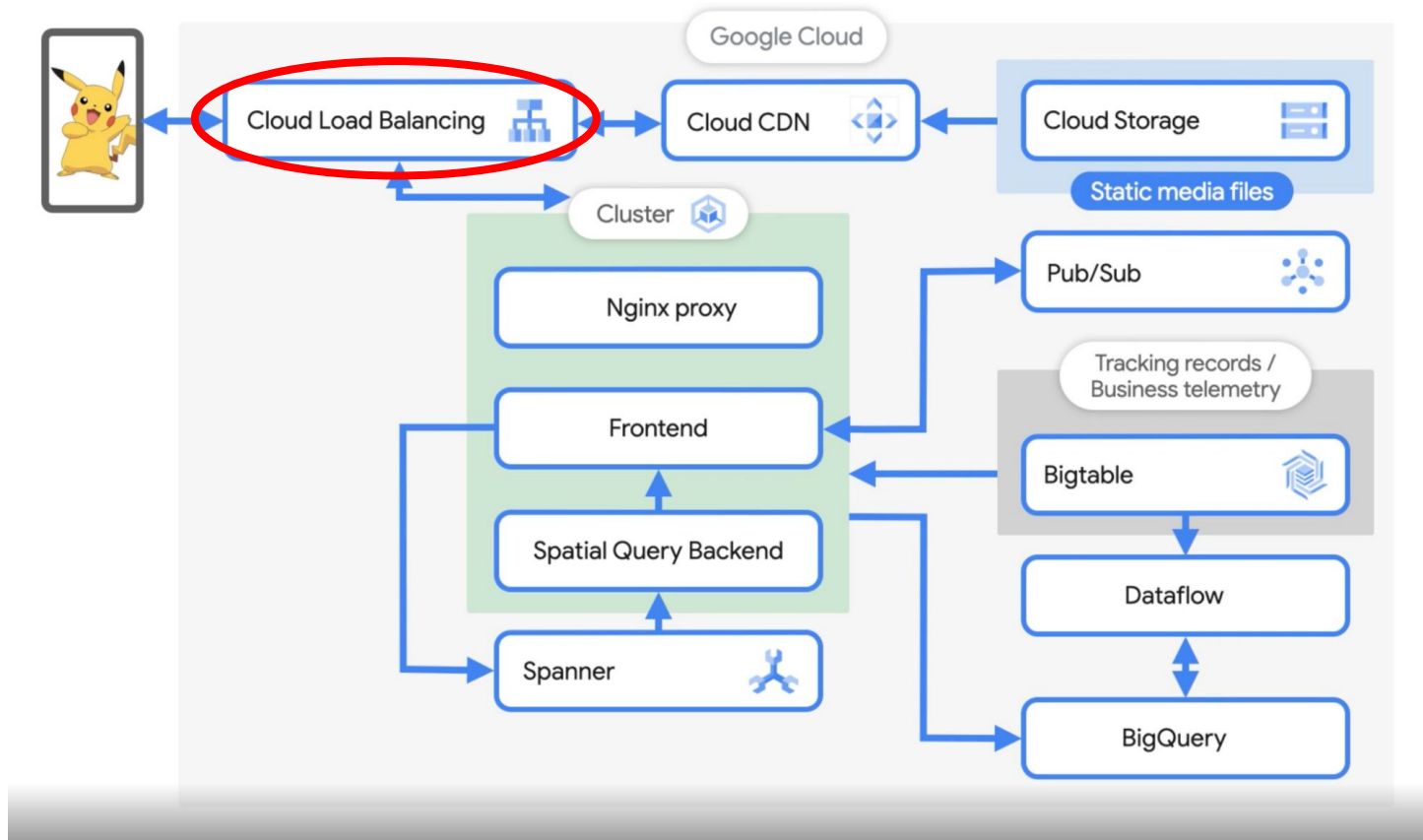
400k requests per
second to up to 1
million requests per
second

How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!

400k requests per second to up to 1 million requests per second

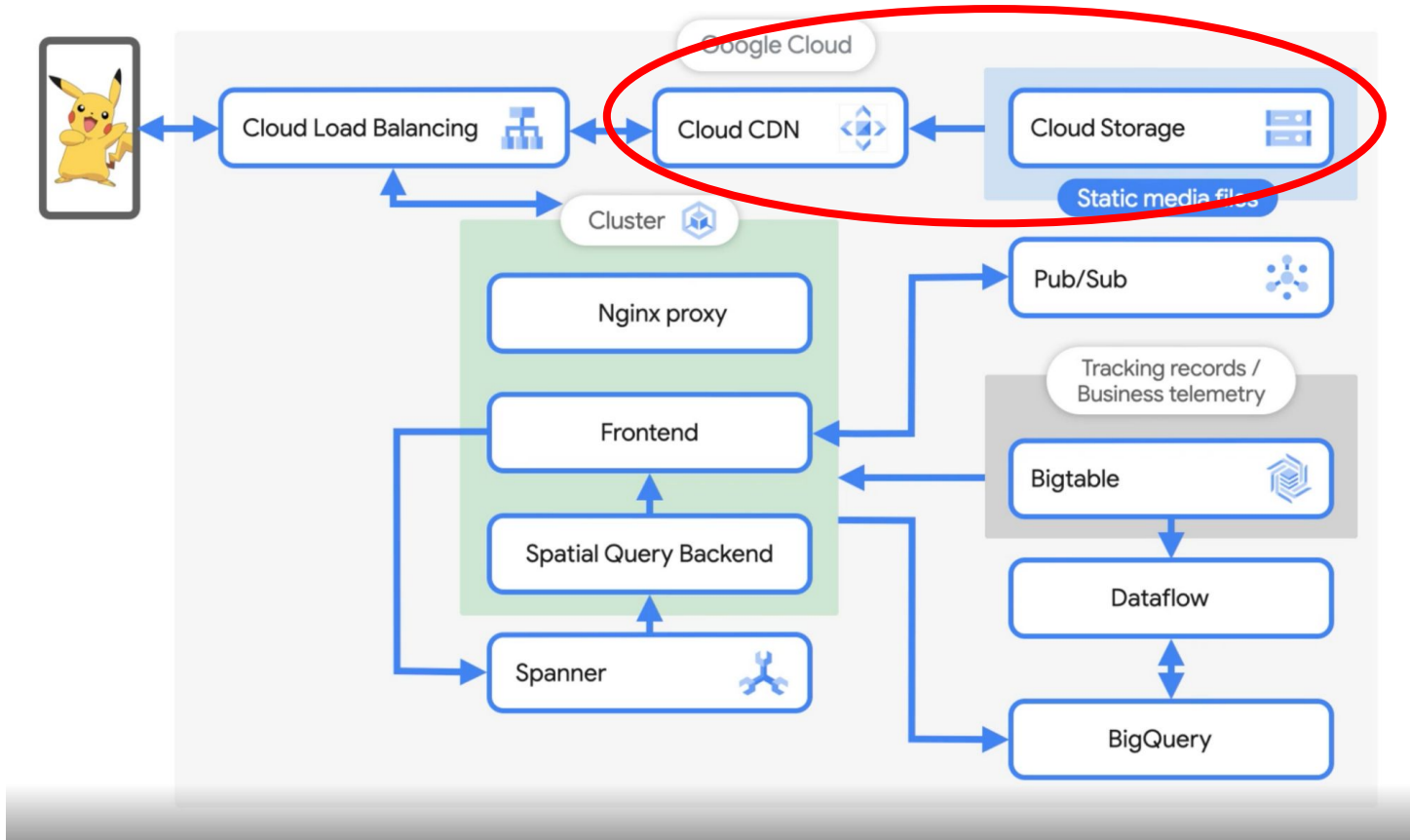


How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!

400k requests per second to up to 1 million requests per second

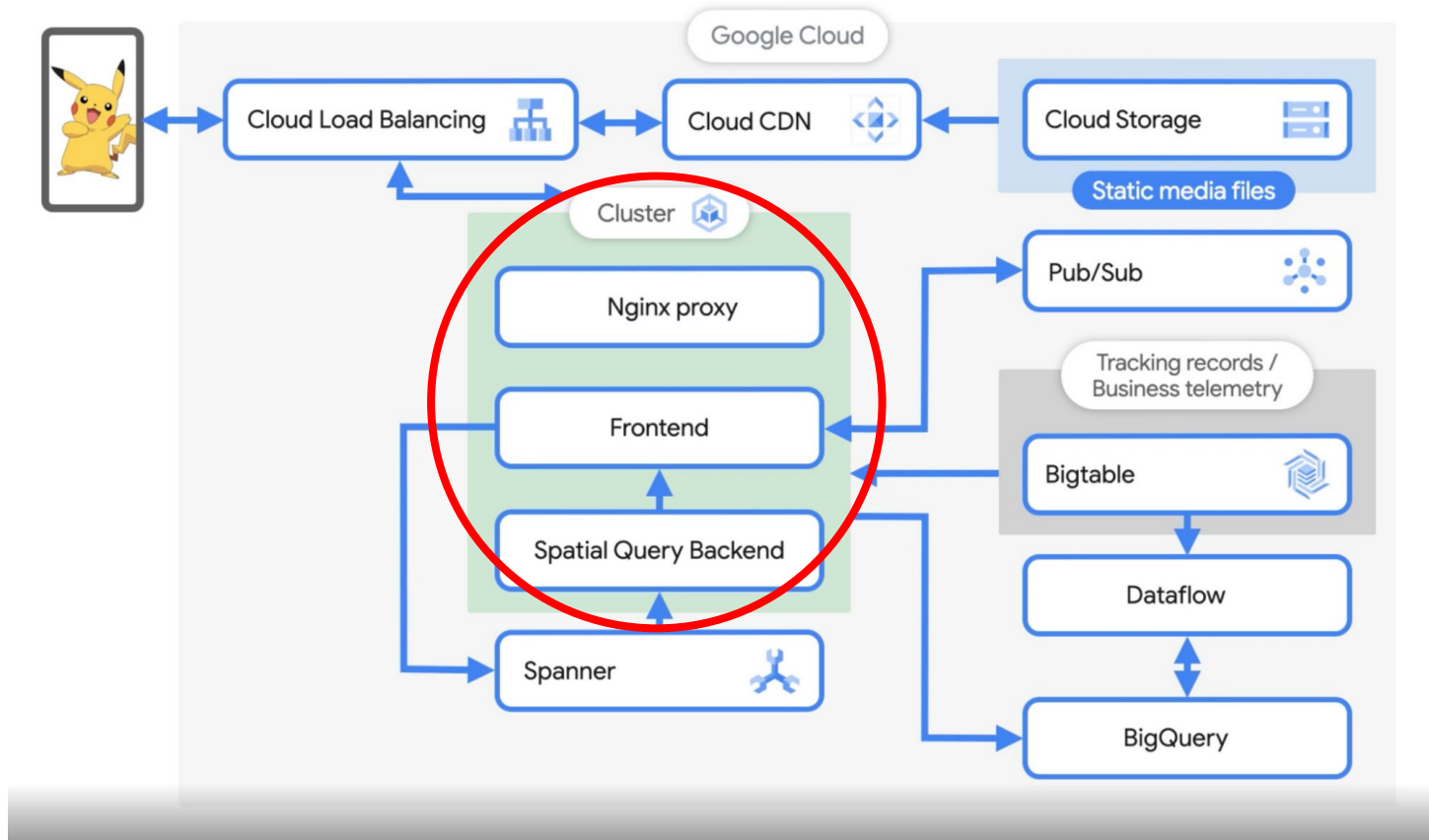


How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!

400k requests per second to up to 1 million requests per second

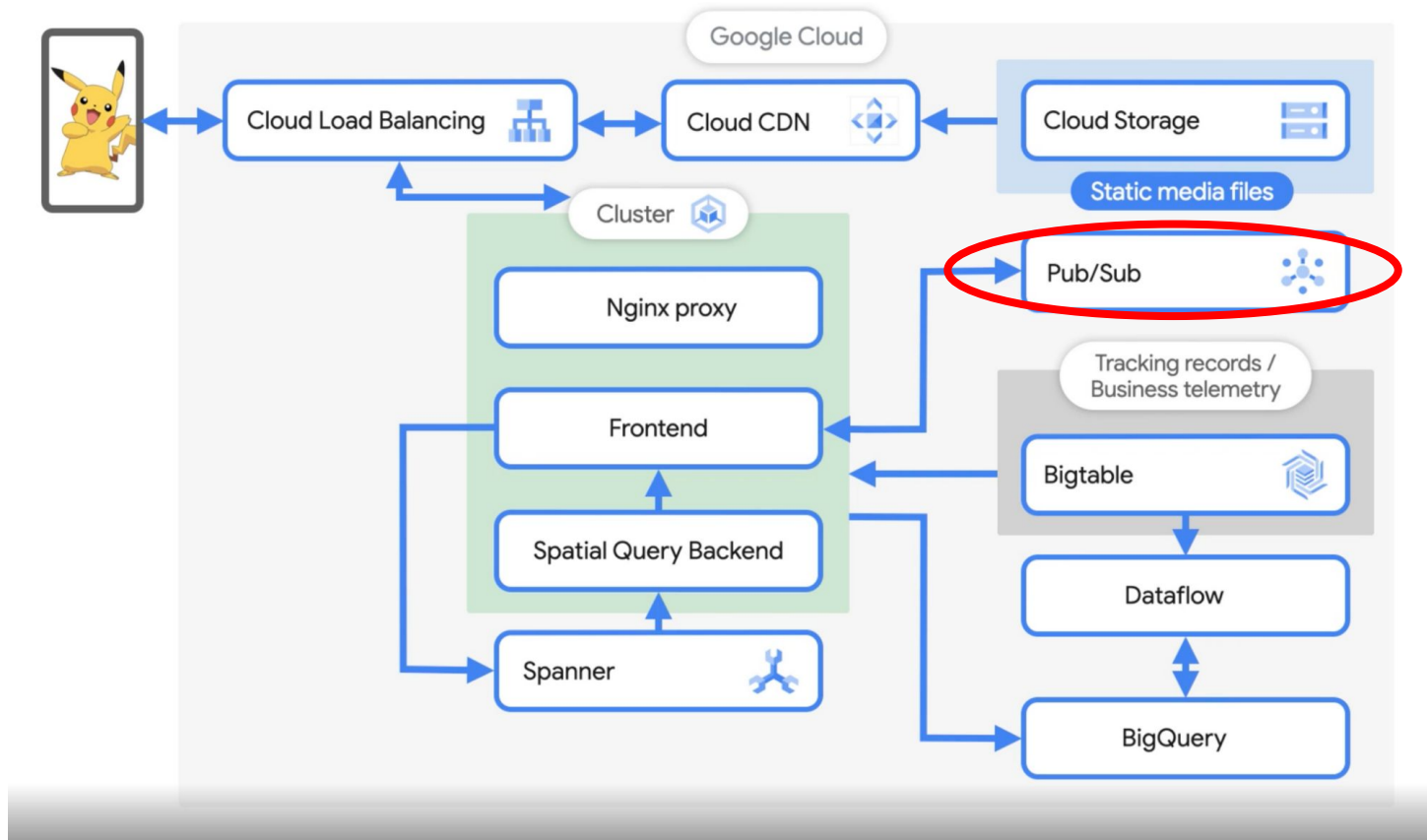


How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!

400k requests per second to up to 1 million requests per second

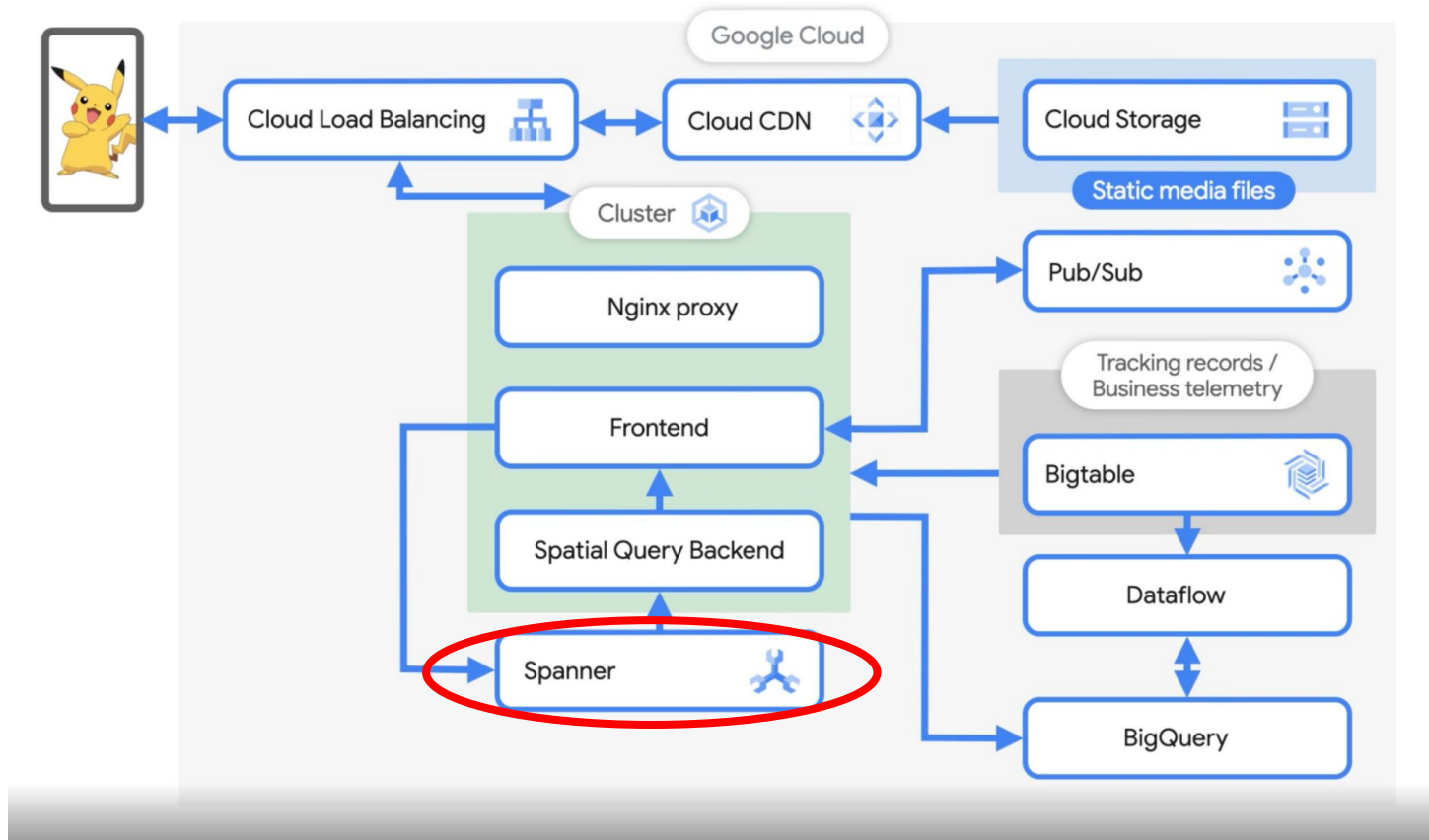


How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!

400k requests per second to up to 1 million requests per second

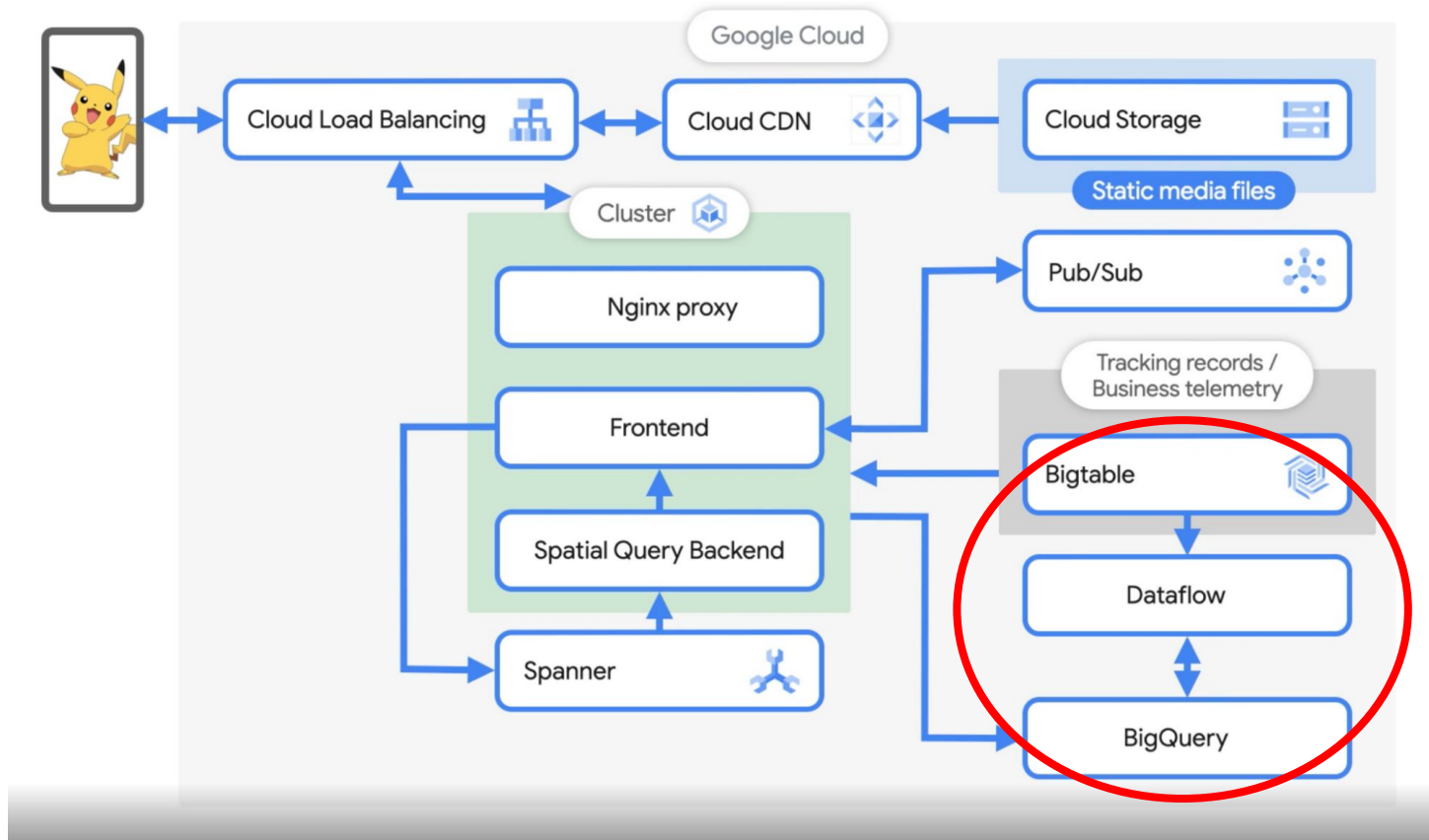


How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Pokémon Go!

400k requests per second to up to 1 million requests per second



How Pokémon GO scales to millions of requests?

<https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests>

Scalability and the Cloud: Serverless

Scalability and the Cloud: Serverless

- In serverless computing, the platform provider takes care of resource allocation (i.e. servers allocated on demand, provider takes care of scaling)
 - AWS Lambda
 - Google Cloud Functions
 - ...

Scalability and the Cloud: Serverless

- In serverless computing, the platform provider takes care of resource allocation (i.e. servers allocated on demand, provider takes care of scaling)
 - AWS Lambda
 - Google Cloud Functions
 - ...
- Billing based on use, e.g. cost per calculation second multiplied by amount of ram consumed, cost per request, cost per storage second

Scalability and the Cloud: Serverless

- In serverless computing, the platform provider takes care of resource allocation (i.e. servers allocated on demand, provider takes care of scaling)
 - AWS Lambda
 - Google Cloud Functions
 - ...
- Billing based on use, e.g. cost per calculation second multiplied by amount of ram consumed, cost per request, cost per storage second
 - E.g. 0.000017\$ per cost per calculation second multiplied by ram consumed, 0.20\$ per 1 million requests (AWS Lambda)

Scalability and the Cloud: Serverless

- In serverless computing, the platform provider takes care of resource allocation (i.e. servers allocated on demand, provider takes care of scaling)
 - AWS Lambda
 - Google Cloud Functions
 - ...
- Billing based on use, e.g. cost per calculation second multiplied by amount of ram consumed, cost per request, cost per storage second
 - E.g. 0.000017\$ per cost per calculation second multiplied by ram consumed, 0.20\$ per 1 million requests (AWS Lambda)
 - Services often include a free tier (e.g. AWS Lambda comes with one million free requests per month)

Scalability and the Cloud: Serverless

- In serverless computing, the platform provider takes care of resource allocation (i.e. servers allocated on demand, provider takes care of scaling)
 - AWS Lambda
 - Google Cloud Functions
 - ...
- Billing based on use, e.g. cost per calculation second multiplied by amount of ram consumed, cost per request, cost per storage second
 - E.g. 0.000017\$ per cost per calculation second multiplied by ram consumed, 0.20\$ per 1 million requests (AWS Lambda)
 - Services often include a free tier (e.g. AWS Lambda comes with one million free requests per month)

When no activity, resources scaled to zero; can take a while to start up - the cold start problem

Scalability and the Cloud: Serverless

- In serverless computing, the platform provider takes care of resource allocation (i.e. servers allocated on demand, provider takes care of scaling)
 - AWS Lambda
 - Google Cloud Functions
 - ...
- Billing based on use, e.g. cost per calculation second multiplied by amount of ram consumed, cost per request, cost per storage second
 - E.g. 0.000017\$ per cost per calculation second multiplied by ram consumed, 0.20\$ per 1 million requests (AWS Lambda)
 - Services often include a free tier (e.g. AWS Lambda comes with one million free requests per month)

When no activity, resources scaled to zero; can take a while to start up - the cold start problem

Still quite a few vendor-specific APIs - vendor lock-in

Scalability and the Cloud: Serverless

Scalability and the Cloud: Serverless

- Starting to also see serverless message queues, serverless databases, ...
 - IronMQ <https://www.iron.io/mq>
 - Amazon SQS <https://aws.amazon.com/sqs/>
 - Google Cloud Pub/Sub <https://cloud.google.com/pubsub>
 - Amazon Aurora Serverless <https://aws.amazon.com/rds/aurora/serverless/>
 - DataStax AstraDB – <https://www.datastax.com/>
 - ...

Serverless and Knative

Serverless and Knative

- Knative <https://knative.dev/> provides an abstraction over Kubernetes, allowing Serverless applications with Kubernetes
- Two parts:
 - Knative serving – set of objects used for running services, automated scaling, etc
 - Knative eventing – a set of APIs for event-driven architecture
- Getting started tutorial at <https://knative.dev/docs/getting-started/>

Serverless and Knative: Example

- Install kn, kn quickstart, kn func
 - <https://knative.dev/docs/getting-started/quickstart-install/>
 - <https://knative.dev/docs/getting-started/install-func/>
- Run quickstart for minikube
 - kn quickstart minikube
- Create tunnel (as root) when prompted by quickstart, then continue quickstart
 - minikube tunnel --profile knative
- Check for existence of knative cluster
 - minikube profile list
- Create a function (here, Python)
 - kn func create -l python hello-python
- Build function (need a registry name)
 - cd hello-python
 - kn func build
- Run function
 - kn func run
- Create a service based on function
 - kubectl apply -f hello-python.yaml
- Find serverless functions
 - kn service list

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello-python
spec:
  template:
    spec:
      containers:
        - image: docker.io/username/hello-python:latest
          ports:
            - containerPort: 8080
```

Serverless and Knative: Example

- Install kn, kn quickstart, kn func
 - <https://knative.dev/docs/getting-started/quickstart-install/>
 - <https://knative.dev/docs/getting-started/install-func/>
- Run quickstart for minikube
 - kn quickstart minikube
- Create tunnel (as root) when prompted by quickstart, then continue quickstart
 - minikube tunnel --profile knative
- Check for existence of knative cluster
 - minikube profile list
- Create a function (here, Python)
 - kn func create -l python hello-python
- Build function (need a registry name)
 - cd hello-python
 - kn func build
- Run function
 - kn func run
- Create a service based on function
 - kubectl apply -f hello-python.yaml
- Find serverless functions
 - kn service list

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello-python
spec:
  template:
    spec:
      containers:
        - image: docker.io/username/hello-python:latest
          ports:
            - containerPort: 8080
```

To cloud or not to cloud?

<https://aws.amazon.com/economics/>

<https://www.cloudcomputing-news.net/news/2022/aug/18/almost-half-of-businesses-struggle-to-control-cloud-costs/>

To cloud or not to cloud?

<https://blog.back4app.com/reduce-cloud-costs/>

<https://www.capitalone.com/software/blog/cloud-cost-optimization/>

<https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/cloud-cost-optimization-simulator>

<https://aws.amazon.com/economics/>

<https://www.cloudcomputing-news.net/news/2022/aug/18/almost-half-of-businesses-struggle-to-control-cloud-costs/>

To cloud or not to cloud?

<https://blog.back4app.com/reduce-cloud-costs/>

Reliance on cloud services at a scale requires
FinOps — <https://www.finops.org/>

<https://www.capitalone.com/software/blog/cloud-cost-optimization/>

<https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/cloud-cost-optimization-simulator>

Beyond software scalability

Beyond software scalability

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Choosing a technology - mature and battle-tested tech for reduced maintenance costs?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Implementations matter - optimization of slow parts is meaningful.

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Choosing a technology - mature and battle-tested tech for reduced maintenance costs?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Implementations matter - optimization of slow parts is meaningful.

Implementations matter - premature optimization is the root of all evil.

Choosing a technology - mature and battle-tested tech for reduced maintenance costs?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Implementations matter - optimization of slow parts is meaningful.

Implementations matter - premature optimization is the root of all evil.

Choosing a technology - mature and battle-tested tech for reduced maintenance costs?

GitOps?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Implementations matter - optimization of slow parts is meaningful.

Implementations matter - premature optimization is the root of all evil.

Choosing a technology - mature and battle-tested tech for reduced maintenance costs?

GitOps?

Security?

Beyond software scalability

- Measuring performance, CDN, Static site generation, API First, Caching, Load testing, Performance testing, Event-driven architecture, Message queues, Database scaling, Docker, Kubernetes, Serverless, ...
- What about...
 - Practices?
 - Teams and technologies?
 - Implementation-level specifics?

Design for failure? Run an extra instance of each service?

Backups!

Building a team - what sort of development and communication practices scale?

Choosing a technology - new tech for increased hiring opportunities?

Implementations matter - optimization of slow parts is meaningful.

Implementations matter - premature optimization is the root of all evil.

Choosing a technology - mature and battle-tested tech for reduced maintenance costs?

GitOps?

Security?

Monitoring?

Third Course Project

Third Course Project

- In the third course project, your task is to create a Jodel-like messaging application. The application should feature:
 - A main page with a list of twenty most recent messages sorted by their posting time, a textarea into which a new message can be written, and a button that can be used to add the message.
 - Similar to the second course project, there is no registration functionality. The user is identified through a random user token that is generated on opening the application for the first time. The user token is stored in localStorage and is used to identify the user in the future.
 - In the list of messages, each message has a text and the time when the message was posted.
 - Clicking on a message in the message list opens the message. Opening a message shows replies to the message and allows writing a reply to the message.
 - A database for storing messages and replies to the messages.
 - A set of Kubernetes configuration files with autoscaling and a database operator that can be used to deploy the application to Kubernetes.

Third Course Project - Passing Requirements

- A working Jamstack-like implementation returned in a format that allows running it easily locally on Windows, Linux and Mac (i.e. a docker-compose configuration or similar for running the application).
 - Recommended: Separate docker services for client and server. Can have more services (and should have as e.g. a database is needed).
- Kubernetes configuration files that allow deploying the application into a Kubernetes cluster. The application needs to have functionality that scales application servers on demand. The database configuration also needs to scale (use e.g. CloudNative PG).
- Performance tests (e.g. with K6) for the application testing the main page and the message page with and without adding a message or a reply to a message. In the tests, record the average requests per second and the median, 95th percentile, and 99th percentile HTTP request duration. Run the tests with a sensible number of concurrent users for 10 seconds.
- Lighthouse Performance score of at least 70/100 for the pages.
- Summary report.

Third Course Project - Passing Requirements / Report

- A markdown-formatted document (no binary content) with:
 - Brief guidelines for running the application (and performance tests if they have been ran with scripts).
 - Guidelines for deploying the application on Kubernetes (e.g. minikube); guidelines can include also e.g. steps needed to create a database table and to set database credentials.
 - *Guidelines must not assume that the user uses the kubernetes dashboard!*
 - Lighthouse Performance results.
 - A brief reflection (5-10 sentences) on the present performance of the application.
 - A brief list of suggestions (5-10 sentences) for improving the performance of the application.

Third Course Project - Passing With Merits

- In addition to fulfilling the passing requirements:
 - Scrolling down on the main page retrieves more messages, twenty at a time.
 - Each message has a score and the possibility to upvote or downvote the message.
 - Upvoting or downvoting the message changes the score.
 - Votes are stored in the database.
 - In case of new messages, replies, or up or downvotes, shown content is updated.
 - If the user is on the main page, new messages are added to the shown list of messages.
 - If the user is on the main page, incoming up or downvotes change the score of the specific message.
 - If the user is on a message page, new replies to that message are added to the shown list of replies.
 - **Note!** This must work also in the situation where there are multiple application server pods. Consider using a separate messaging service to achieve the desired outcome.
 - Lighthouse Performance score at least 80/100 for the pages.

Future.. WebAssembly?