Amazon Web Services (AWS)

Programování zařízení Apple (IZA)

Martin Hrubý, FIT VUT, 2022/23

Motivace

- Moje profesní zaměření: simulace, kombinatorické optimalizace.
- Nasazení výpočtů do podoby služby.

Vztah cloudu k mobilním aplikacím:

- hostování výpočtů,
- serverové databázové systémy - sdílení dat.
- multi-mediální služby, zpracování multi-medií.
- testování aplikací.

Apple nikdy nevynikal ve serverových službách.

Úvod

Co je cloud?

Cloud poskytuje:

- služby,
- zdroje (výpočetní, storage, komunikační, analytické),
- správu projektů (typicky financování).

Cloudová aplikace = distribuovaný systém. Nový zdroj problémů.

Historické pozadí

- Servery pro konkrétní služby (FileServer, Mail, DB, …).
- Pak se objevilo slovo *cloud*.
- Poskytování prostoru pro virtuální počítače.
- Docker.

Containerizace uvolnila široké spektrum možných architektur.

Docker - image & container.

Mikro-virtualizace.

- Jako VM, ale není VM. Je napojena na hostitelské jádro.
- Systémový démon.
- Správa images. Běh containerů. Volumes. Networks.
- Kompozice containerů do vyšších celků - Docker Compose.

Dále pak:

- Kubernetes a spol.
- Rozsáhlé systémy služeb (AWS, GCP).

K čemu je dockerizace dobrá?

- Je to zapouzdření aplikace do containeru. Primární motivací **není** bezpečnost.
- Předání aplikace uživateli. Snadná de-instalace aplikace (pokusy, služby, apod.).
- Standardizace - pracuju s containerem.
- Serverové nasazení (mikro-služby, výpočty, servery).

Architektura přehledově

Lokální aplikace `docker`.

- Lokální repozitář pro images.
- Serverový repozitář (pull, push).
- Názvosloví images: <repo><nazev><:tag>

Operace:

- `docker build -f skript .`
- `docker run <args> -t repo [args]`
- `docker tag ...`
- `docker push ...`

Tvorba docker-image

Dockerfile

```
FROM repo [AS nazev]

ARG prom1
ENV prom2=cosi

WORKDIR /neconeco
COPY ./ ./

RUN ...

CMD | ENTRYPOINT ...
```

```
docker build –t mujimage:1 .
```

Tvorba docker-image

- Z lokálního adresáře - kopie souborů.
- Z dalších repozitářů (Github) - pozor na credentials.
- Dvou-fázový build.

```
FROM cosi as BUILD
RUN ... make; build; ... cokoli

FROM alpine
COPY --from=BUILD /odkud/binarka /na/binarka
CMD ["/na/binarka"]
```

Spuštění docker-container

`docker run args image:tag ARGS`

- mapování svazků `-v from:to`
- systémové proměnné ENV
- sít (network), porty `-p FROM:TO`
- `-it`, `--rm`

Správa images

Repozitář. Export-import (tar).

- Docker Hub - private, public.
- AWS ECR.
- Google GCP.
- DigitalOcean.

Serverové služby potenciálně stahují (pull) image z repozitáře při každém spuštění.

- minimalizace velikosti image.
- credentials.

Docker Compose

```
services:
  db:
    image: postgres
    volumes:
      - ./data/db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    environment:
      - POSTGRES_NAME=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    depends_on:
      - db
```

`docker-compose up` (`down`)

Kubernetes

Platforma pro spouštění/správu docker-containers.

Koncepce:

- vytvoříme cluster (sada systémových démonů).
- do clusteru vkládáme zdroje (YAML definice).

Zdroje:

- nodes - výpočetní uzly (1-node cluster). Škálování.
- Pod, Service, Job, Deployment, StatefulSet.

Aplikace v Kubernetes versus v AWS/GCP.

Kubernetes - prakticky

- docker poskytuje implementaci v rámci distribuce.
- AWS - $0.1/hod za cluster :(
- GCP - cluster je zdarma (! jeden).
- DigitalOcean.
- vlastní pokusy o správu clusteru (minikube).

`kubectl ...`

- napojit na cluster (lze přepínat kontext),
- příkazy (create, apply, ...).

Kubernetes: Pod

Každý prvek v clusteru je virtuálně počítač, tj

- má IP adresu (lokální).
- cluster je systém potenciálně spolupracujících podů.
- je dán docker containerem, porty.
- má přístup na volumes.

Pod = jedna virtuálka s obecně více containery.

- restarty Podu.

```
kubectl get pods
kubectl describe pod/nazev-podu
```

Kubernetes: Pod (demo)

```
apiVersion: v1
kind: Pod
metadata:
  name: moje-echo
spec:
  containers:
  - name: moje-echo-cont
    image: mhafan/echo1
    ports:
    - containerPort: 8080
  imagePullSecrets:
  - name: regcred
```

Na portu 8080 v cluster naběhne služba. Co dál?

```
kubectl exec --stdin --tty moje-echo -- /bin/bash
```

Kubernetes: Pod (co dál?)

Napojení na service - doménové jméno, vnější port.

- service - fakticky jenom pojmenování podu.
- loadbalancer - mapování interního portu podu na vnější port clusteru.

Správa podu:

- job.
- deployment.

Životní cyklus podu:

- obnovování, restartPolicy.

Kubernetes: Job

Pod postavený do role zadaného (jednorázového) výpočtu.

- completions/parallelism
- completionMode: Indexed

Jediný prvek Kubernetes s charakterem fronty.

- cluster vlastně chce spustit veškeré Pody.
- … až po naplnění své kapacity (vCPU, memory).
- … ale nezavádí pořadí, tj frontu (priority, uživatelé apod).

Kubernetes: Job (demo)

```
apiVersion: batch/v1
kind: Job
metadata:
  name: cpuload1
spec:
  ttlSecondsAfterFinished: 3
  completions: 3
  parallelism: 3
  template:
    spec:
      containers:
      - name: pi
        image: jfusterm/stress
        command: ["stress", "-t", "10", "-c", "1"]
        resources:
          requests:
            cpu: "1000m"
          limits:
            cpu: "1000m"
      restartPolicy: Never
```

Kubernetes: vsuvka (Sun Grid Engine)

SGE. Správa distribuovaných zdrojů (uzlů, CPU).

- fronty
- priority, uživatelé.
- qsub -q fronta skript.sh

Kubernetes rozhodně není něco jako SGE.

- systém služeb (stavových/bez-stavových),
- popsán sadou YAML konfigurací (včetně volumes).
- rychlá migrace (volumes, images v repozitáři, pár YAML skriptů).

Kubernetes: Deployment

Pod s rozšířenou správou.

- replicas - kolik instancí se má spustit.
- service - pak principem LB na repliky.
- přechod na novou/starou verzi (image:tag).

Smyslem je hladký přechod na novou verzi služby.

- aktualizace verze v produkčním prostředí.
- TOTO je nejsilnější stránka Kubernetes.

StatefulSet:

- součástí deployment má být i perzistentní paměť.

Kubernetes: Deployment (demo)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: moje-echo
spec:
  selector:
    matchLabels:
      app: echo1
  replicas: 3 # !!!!
  template:
    metadata:
      labels:
        app: echo1
    spec:
      containers:
      - name: echo1
        image: mhafan/echo1
        ports:
        - containerPort: 8080
```

Kubernetes: Secrets

Obecně problematická a otravná část serverových aplikací.

- správné uložení credentias.
- obecná pravidla.
- nový druh počítačové kriminality.

Amazon Web Services (AWS)

Amazon Web Services (AWS)

Serverová aplikace stavěná prostředky AWS.

- srovnání s Kubernetes.

Opět úzce zaměřeno na simulační výpočty.

- akcent na výpočetní služby.
- cloud = velmi rozsáhlý pool výpočetních zdrojů.

Zřízení přístupu

Koncepce:

- účet - vlastník účtu, účetnictví, organizace.
- uživatelé - každému lze definovat přístupová práva (IAm).
- regiony - jsou separovány (soft/hard constraint). Přetoky.

Účtované položky (kategorie):

- zatížení vCPU - různé formy. (co je 0.25 vCPU)
- zatížení RAM + storage - různé formy.
- síťové toky - snaha minimalizovat.

Free tier. Amazon se snaží vést uživatele k úspornosti.

Přehled služeb rámcově

- Analytika.
- Compute. Container.
- Database.
- Developer tools.
- Web.
- Storage.
- Servisní činnosti - IAM, účetnictví.

IAM - uživatelé, role, policies.

- uživatel obdrží přístupové kódy - login/heslo (AppKey). AWS CLI.
- role - virtuální uživatel (zastupuje uživatele v úkonech).
- policy - soubor oprávnění. Značně košaté.

Sestavování souboru těch správných policies pro role je bolestivé.

- automatické generování,
- ruční vytváření a znovupoužití.

AWS - na každý konkrétní úkon extra role. Granularita.

- Srovnání GCP - projekty, service accounts.

Developer tools

Svět "dev-tools" kolem serverových aplikací.

- Nástroje a jejich prezentace.
- Continuous všechmožné (Integration/Development).

Příšerný prales vývojářských nástrojů.

CodeCommit:

- GIT private repozitář. Pochybné účtování přístupu.
- AWS nechce z CodeCommit dělat veřejný GitHub.
- (ukázka)

Developer tools - CodeBuild:

Nejen pro `docker build`.

- vcelku pěkná služba. Cena za 1min běhu.
- tajemné parametry (ukázka).
- webhook, role (univerzál, odškrtnout "Allow"), vCPU.
- `buildspec.yaml`
- Dockerfile: FROM xy, preferováno z Amazon public ECR
  - docker-hub odmítá množství pull requestů ;)
  - docker-hub, můj účet - předkompilované public repo s instalacemi g++/golang/knihovny...

Code Pipeline/Deploy.

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region eu-central-1 \
       | docker login --username AWS \
       --password-stdin <ACC-ID>
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t XY:dev -f Dockerfile.xy .
      - docker tag XY:dev <ECR-REPO>
      - docker push <ECR-REPO>
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
```

Elastic Container Registry (ECR)

- Private/Public. To "public" je pochybné. Toky dat.
- URL repozitáře (docker-hub, ECR, GCP): ECR asi nejhorší.
- LifeCycle policy - průběžná likvidace starých verzí.
- účto: $0.1/GB/month mínus FreeTier.

Výpočetní služby AWS vyžadují ECR jako zdroj (GCP/DO ne).

- CodeBuild nebo docker-push.

Elastic Container Service (ECS)

Sdružuje služby spouštění kontejnerů.

- Task - definice parametrů spuštění containeru.
- Cluster - virtuální výpočetní zdroj.
- Spuštění tasku (ukázka).
- AWS Batch - bude dále.

Obecněji spouštění containerů

AWS/GCP/DO podobně stavěné služby. Logování.

Container chci spustit jako:

- službu - URL+port (očekávám REST API). Setrvalý běh. Běh na požadavek.
- jednorázový job - řada možností.

Zajímá nás typicky účtování. Osobně: jednorázové výpočty.

- způsob startu výpočtu/služby.
- studený start, horký start (zřejmě "předstartováno").

AWS Lambda (!!!)

Úplně ze všeho nejvíc nejuniverzálnější služba v Compute.

- Má charakter container-jako-služba. Vstup-Výstup.
- Způsoby invokace Lambda-funkce.
- Nastartovaný container naslouchající na portu.
- Účtuje se za čas strávený dotazy.

Vytvoření:

- malý kus kódu v Go/Node.js/Ruby/Python.
- nebo ECR image.
- execution role.

Lambda: parametry existence funkce

- Memory - GB. Určuje porci přiděleného času (1500MB=1vCPU).
- Lokální `/tmp`.
- Paralelizace (1000). Rezervovaná paralelizace.
- Cena: $0.000016667/GBs ($0.06/hod). 400tis GBs FT.
- Kód funkce musí předpokládat více-vláknový běh.

StateMachine.

Demo: NMTSimulator/bloader.

Lambda: spuštění

Je to předem rozběhnutý container. Globální data = napojení na další AWS služby. Inicializace.

- přímá invokace - URL funkce, z rozhraní AWS (sync/async).
- trigger - GateApi, SQS, další služby AWS.

Concurrency:

- LoadBalancer Lambdy řídí spouštění dalších instancí containeru. ColdStart. 1000 kvóta.
- Dodatečné kontejnery zůstávají aktivní (20 min cca).
- V tomto smyslu je to zařízení s omezenou kapacitou a bez fronty.

Lambda s explicitní frontou: SQS

Simple Queue Service:

- pojmenované fronty. Zápis do fronty. Výběr z fronty.
- Standard/FIFO.

Sémantika výběru z fronty (masivně paralelní přístup):

- dočasně vyberu prvek (je rezervován).
- do časového limitu ho musím smazat, jinak je vrácen do fronty.

Účtování:

- přístupy ($0.4/1m), uložená data (256kb-limit na prvek).

SQS

- je pasivní zdroj - musí se explicitně dotazovat.
- Lambda+SQS - dotazování (počet přístupů za měsíc).
- Alternativně: Rabbit MQ (aktivní prvek, tj deployment).

Zápis do SQS:

- REST API. Max 10 prvků na request.
- masivně paralelní zápis (tisíce prvků za 1s).

Aplikace pro Lambdu

- drobné činnosti - přístup do DB, odeslání něčeho. Milisekundové operace (při 128MB velmi levné).
- servisní činnosti - napojení na DynamoDB/S3/Athena.
- výpočty všeho druhu - 900s max, 2 vCPU max.

Napojení na vnější svět: GateAPI.

Micro-services: server-less platforma.

Pozor na $-účet! :) Kvóty (concurrency, paměť apod).

Compute: dedikované virtuálky

- LightSail - komplet virtuálka (vCPU/mem, statická IP, EBS).
  - účtuje se až do zrušení (IP, EBS).
  - "neočekává se", že pojede naplno.
- EC2 - pronájem virtuálek (dedikované/SPOT).
- Kubernetes cluster.
- AWS Batch.

Instance EC2

- stav instance - zastavena, běžící, terminated.
- typ instance (typ HW) - lze měnit v zastaveném stavu.
- EBS - $0.1/GB/month.
- EFS - sdílený NFS svazek přes více instancí.

Povolení portů. SSH klíč.

SPOT instance

- Nevyužité zdroje Amazonu za zlomek ceny.
- Jejich dostupnost není garantovaná.
- AWS si je smí vzít kdykoli zpět (2min varování).
- Ideální na krátké výpočty.

Dynamické přidělování SPOT instancí. Request.

Dávkové výpočty: AWS Batch

Dvě provedení: ECS a Fargate.

- Compute Environment - Fargate SPOT. Počet vCPU.
- Queue - napojení na Comp.Env.
- Job Definition (Task) - parametry spuštění z ECR.
- Job - instance Tasku. Jednotlivá/pole.

Fargate

Hned po Lambdě nejvíc nejlepší ;)

- Virtuální výpočetní zdroj - cena za vCPU/hour a GB/hour.
- Frankfurt: cca $0.02/hour za 1vCPU/2GB. Spot price.
- Job definition - specifikuje zdrojové požadavky.
- Instance - joby...

Provedení Jobu:

- Provisioning - cca 25s
- Účtuje se po 1s (minimálně 1min).
- Pozor: Public IP na definici. Log až na konci.

Databáze

Managed versus "na virtuálce".

- DynamoDB.
- PostgreSQL. REDIS.
- TimeStream.

Obecně o no-SQL databázích. Klíč-hodnota. REDIS.

- Pohovořit o koncepci REDISu.
    - noSQL DB, Pub/Sub message broker.

DynamoDB

Tabulka. Primární klíče. Hashování.

- Partition Key - hash. Význam.
- Sort Key - dodatkový klíč pro odlišení záznamů.
- Secondary Index (es).

Účtování přístupu:

- On-Demand.
- Provisioned - pozor, generuje hlášení.

DynamoDB: definice/návrh klíčů

Klíč (Partition/Sort) berme jako libovolně formulovatelný (typicky) string. Pak lze řešit vztahy 1:N apod.

- `atribut1:atr2:atr3`

Články o mapování E-R diagramu na DynamoDB schema.

DynamoDB

Operace:

- Put (insert). Put na stejný klíč (update).
- Query - dotaz na partition/sort Key.
- Scan.

Účtování:

- uložená data.
- přístupy (write/read) - unit, cena za 1M units.

Web a APIs

- AWS Amplify - webová aplikace (React), hostovaná v AWS.
- AWS API Gateway.

API Gateway - napojení např z mobilní aplikace.

- přidělená URL nebo vlastní
- autentizace.

API Gateway

- Routing, metody.
- Napojení do AWS - typicky Lambda.
- Kontrola datového modelu (formát dotazů).
- Usage Plan.
- x-api-key
- Napojení na vlastní domény - Route 53.

Storage

S3:

- S3 - bucket (globálně unikátní název).
- $0.1/GB/month

Dále

- Glacier.
- Backup.

Další speciality

- Event Bridge - rozesílání zpráv, transformace.
- Rabbit MQ - hodně silný message broker.
- Step Functions - skriptování nad službami AWS. Vizuální programování.

Analytika: Athena

- Specifikace datového zdroje - velmi univerzální konektory.
- Pro mě: typicky bucket v S3. CSV/JSON soubory.
- AWS Glue - nad datovým zdrojem (data lake) vytvoří virtuální tabulku.

Athena - SQL-like dotazy nad virtuální DB/tabulkou.

Experimenty -> CSV/JSON výstupy -> S3 -> Athena.

Case Study - NMTSimulator.

Závěr

Bylo to velmi "compute-oriented". Dávkové zpracování.

- Studium služeb AWS/GCP (GCP je velmi podobný).
- Kubernetes: pro studium doporučuji DigitalOcean.

Serverová aplikace: v Kubernetes nebo v AWS?

- Pod/Deployment -> Lambda (1000 replicas).
- Service -> GateWay
- Delší výpočty -> Fargate (fronta, virtuání comp. env.)
- Uložiště (DB/volumes) -> DynamoDB/REDIS/S3.

AWS CloudFormation.