SYSTEM1

Data Science infrastruktúra skálázása a felhőben

Budapest Data Forum 2020

attila.nagy@system1.com

A System1-ról

Alapítás:

2013

Venice, CA, USA

Alkalmazottak száma:

300+

Irodák száma:

7,01*

Venice, CA (HQ), Bellevue, WA, Denver, CO, San Diego, CA, Guelph, ON (Kanada), Portsmouth (Egyesült Királyság), Hága (Hollandia)

Publishing

Tartalomgyártás és -szolgáltatás széles spektrummal (egészség, wellness, autók, pénzügyek, szórakozás és kvízek)

Search

Google, Bing és Yahoo partnerségre épülő privát keresés (Startpage), info.com, Webcrawler

Applications & Services

MapQuest, Waterfox böngésző

The System1 Platform

“Our platform offers top publishers and our proprietary products a full funnel solution to attracting users and expanding audiences. We have buying strength across multiple networks and have unique search and ad partnerships”

*: Magyarország

SYSTEM1

Open Source Software

DATABASE TOOLS

SECURE DB FOR R
Convenient & secure database connector from R

SNOWFLAKE JDBC FOR R
Snowflake JDBC driver for R

ATHENA JDBC FOR R
Amazon Athena JDBC driver for R

Redshift JDBC driver for R
Amazon Redshift JDBC driver for R

JOB TOOLS

JOBS FOR REDIS
Job input/output control with Redis

JENKINS TO AWS BATCH
Jenkins plugin to submit tasks to AWS Batch

MODELING

FILTERNET: NEURAL NEWORK FOR TIME SERIES
PyTorch implementation of a neural network for time series modeling

UTILITIES

CLOUDPERF
Measuring the relative performance of cloud resources

LOGGING FOR R
A modern and flexible logging utility for R

RETICULATE ON BOTO
Reticulate wrapper around boto3: an AWS client for R

CONSUL.IO CLIENT FOR R
Consul.io distributed server mesh client for R

JUPYTER NOTEBOOK TO EMAIL
nb2mail: Sending Jupyter notebooks as email

SHELL FOR R
Shell runner for R scripts, like Python Fire

SLACK SCRUM BOT
Python script to generate scrum standup report for Slack

GRPC FOR R
gRPC client and server implementation in R

SYSTEM1

Data Science csapat

- 10 fő data scientist/engineer
- Python és R
- Amazon AWS infrastruktúra
- Ad-hoc és erőforrásigényes, hossza(ba)n futó jobok (pld. modellek tanítása)
- Gépi tanulás
- Ad networkök kezelése
- Kísérletezés, számításigényes feladatok (GPU)

SYSTEM1

Data Science jobok ütemezése, futtatása

- Jenkins egy EC2 gépen
- Jobok definíciója groovy fájlokban (job DSL pluginnel), gitből
  - követhető, auditálható, peer review kikényszeríthető (CODEOWNERS, merge policy)
  - szintaktikailag ellenőrizhető (GitHub PR builder, hiba esetén komment, illetve merge megakadályozása)
- Build logok
  - Jenkins UI (színes kimenet, timestampek)
  - AWS CloudWatch Logs
- Értesítés a futások eredményéről
  - Jenkins UI
  - Slack
  - e-mail
- A jobok konténerekben futnak, image-ek kezdetben saját repóban, aztán AWS ECR
- Adattárolás/betöltés: AWS S3/Snowflake

Erőforrások

Kevés és/vagy nem erőforrásintenzív jobokhoz elég egy gép. De milyen lehetőségeink vannak ha már nem? (számtalan)

Vertikális skálázás
+ a legegyszerűbb
+ az AWS-ben mindig* választhatunk nagyobb gépet, típuson belül kb. azonos teljesítmény/ár aránnyal
- a kihasználatlan időre is fizetjük a költségeket

AWS Batch
± AWS menedzselt infrastruktúra/nincs ráhatásunk
± System1 plugin
- sok üresjárat, amit mi fizetünk
- job konzol kimenet/debugging problémák (CloudWatch Logs API limit)

Elosztott Jenkins (statikus/dinamikus node-okkal)
+ EC2 plugin
+ AWS Auto Scaling Group
- az eltérő jobok eltérő igényeit kézzel kell kiszolgálni, beállítani
- aszimmetrikus jobok (CPU/RAM)

Kubernetes
+ Jenkins plugin
± Bin Packing (több job egy gépen)
- pod-ok skálázása/dinamikus igények gyors lekövetése
- eltérő típusú instance-ok kezelése
- aszimmetrikus jobok

- Hasonló gondok
- Mindegyikkel “dolgozni kell”
- Nem egy megadott instance-típusból akarunk scale-out clustert építeni, és/vagy kézzel jobokhoz rendelni ezeket
- Nem akarunk az üresjáratért fizetni

*a jelenlegi korlát: u-24tb1.metal: 224 core, 24 TiB RAM, ~15M Ft/hónap

SYSTEM1

Feljesszünk saját megoldást!

Célok:

- Docker image-ek igény szerint helyben (Jenkins), vagy távoli instance-okon történő futtatása
- A költségek felezése (AWS Batchhez képest)
- Job kimenetek realtime begyűjtése, CloudWatch Logs-ban tárolása
- AWS EC2 Spot instance-ok használata, paraméterezhetőség
- Jobok használati adatainak gyűjtése (CPU, memória)
- A jobnak leginkább megfelelő instance típus kiválasztása
- GPU-támogatás (AWS GPU instance-ok, CUDA)
- Timeout, retry és job leállítás támogatás
- Instance-ok tagelése, költségek nyilvántartása
- Hiba esetén riasztás

Alap használat

Az alábbi parancs a helyi gépen kéri le az instance ID-jét egy konténerben:

```
$ docker-run.py bash -c 'curl -s http://169.254.169.254/latest/meta-data/instance-id | sha1sum'

=> Date: Tue Sep 15 09:30:18 UTC 2020
=> Container timeout: 6h
=> Starting Docker container: 11c7fddd-f736-11ea-940c-021189b30dec
=> Docker image (hash): ***.dkr.ecr.us-west-2.amazonaws.com/om-datasci:latest
(sha256:6a3c4a3c222394d47099b898c79f5c7c63f1da8dcfab777481d0a52dafa67fc2)
=> Docker run script hash (size): e683f8d38a6b05e7adfc313fa6b20c61 (81849)
=> Extra Docker parameters:
=> Command to run in Docker: bash -c curl -s http://169.254.169.254/latest/meta-data/instance-id | sha1sum
=> Actual Docker command: docker run --rm --init --name=11c7fddd-f736-11ea-940c-021189b30dec -e
BUILD_TAG=11c7fddd-f736-11ea-940c-021189b30dec ***.dkr.ecr.us-west-2.amazonaws.com/om-datasci:latest bash -c curl -s
http://169.254.169.254/latest/meta-data/instance-id | sha1sum
================================================================================
75c317dcc81762565c04bd2ba0e8fbf0a85b4107  -
Exit code: 0
```

SYSTEM1

Futtatás másik instance-on

```
$ docker-run.py --remote bash -c 'curl -s http://169.254.169.254/latest/meta-data/instance-id | sha1sum'

=> Starting m3.medium spot:True price:0.05/0.07 perf/price:747 1c score:35 vcpu:1 mem:3.75 AZ:us-west-2c AMI:***
=> Waiting for instance i-0d8de92f895400857 to be ready (10.150.136.253)
=> Initializing remote machine
=> Date: Tue Sep 15 09:22:54 UTC 2020
=> Container timeout: 6h
=> Starting Docker container: 08de866a-f735-11ea-8929-0a0f76ad7a81
=> Docker image (hash): ***.dkr.ecr.us-west-2.amazonaws.com/om-datasci:latest
(sha256:6a3c4a3c222394d47099b898c79f5c7c63f1da8dcfab777481d0a52dafa67fc2)
=> Docker run script hash (size): e683f8d38a6b05e7adfc313fa6b20c61 (81849)
=> Extra Docker parameters:
=> Command to run in Docker: bash -c curl -s http://169.254.169.254/latest/meta-data/instance-id | sha1sum
=> Actual Docker command: docker run --rm --init --name=08de866a-f735-11ea-8929-0a0f76ad7a81 -e
BUILD_TAG=08de866a-f735-11ea-8929-0a0f76ad7a81 ***.dkr.ecr.us-west-2.amazonaws.com/om-datasci:latest bash -c curl -s
http://169.254.169.254/latest/meta-data/instance-id | sha1sum
================================================================================
b049486d1602a78876c4f4bbd713d9dfd10862e5  -
Exit code: 0
Maximum memory usage: 19.6MiB
Average memory usage: 19.6MiB
Average CPU usage: 0%
Max CPU usage: 0%
Estimated EC2 cost: $0.00
```

SYSTEM1

Távoli futtatás - lehetőségek

- Automatikus instance típus választás legjobb ár szerint
- Filterek (pld. vcpu>=8, memory>=16)
- Spot instance-ok (blocked 1-6 óra, vagy nélküle)
- Spot instance kill signal kezelés
- Tetszőleges AMI indítható (ha teljesíti a futási paramétereket). A saját git repónkkal “előmelegített” image-eket publikálunk, így gyorsabb az elindulás
- Egy adott jobra beállítható, hogy legfeljebb hány példányban futhasson
- Auto tuning lehetőség: a job előélete alapján kitalálja, hogy milyen instance-on érdemes futtatni azt

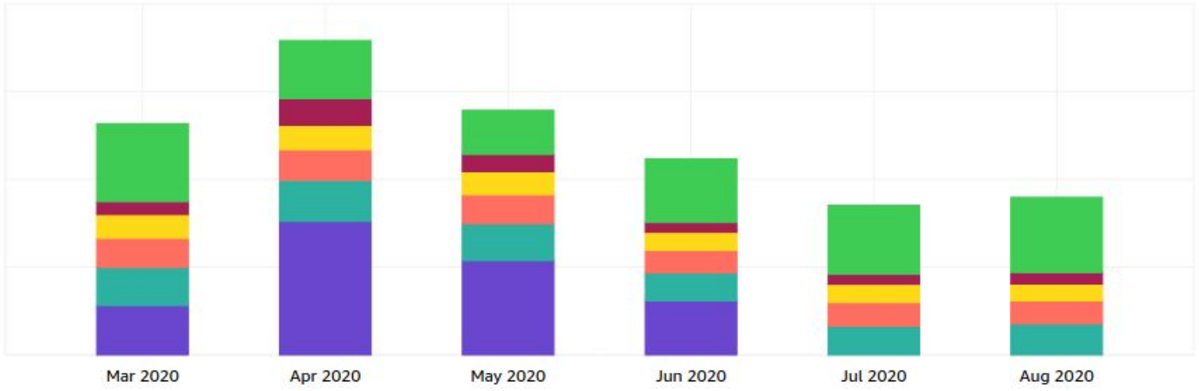
Futási metrikák

- Minden jobról (egyedi kulcs: JOB_NAME, BUILD_NUMBER, vagy GUID) metrikákat gyűjtünk, amiket a futás végén S3-ba írunk, onnan pedig Snowpipe-pal Snowflake táblákba kerülnek, és SQL-lel lekérdezhetők
- Könnyen megkereshető pld. a legtöbb memóriát fogyasztó, vagy a “legdrágább” job:

```
330 SELECT job_name, build_number, "start", duration, instance_type, cost, mem/POWER(1024, 3) FROM data_science.infra.job where cost is not null ORDER BY cost DESC LIMIT 1;
```

Row	JOB_NAME	BUILD_NUMBER	start	DURATION	INSTANCE_TYPE	COST	MEM/POWER(1024, 3)
1	SEM Keyword NGram Embe...	54	2019-08-14 19:19:37.000	45014	p3.2xlarge	38.363641535	0.01578521729

- Az EC2 tageknek köszönhetően könnyen nyomonkövethető az AWS Cost Explorerben a jobok költségeinek megoszlása:

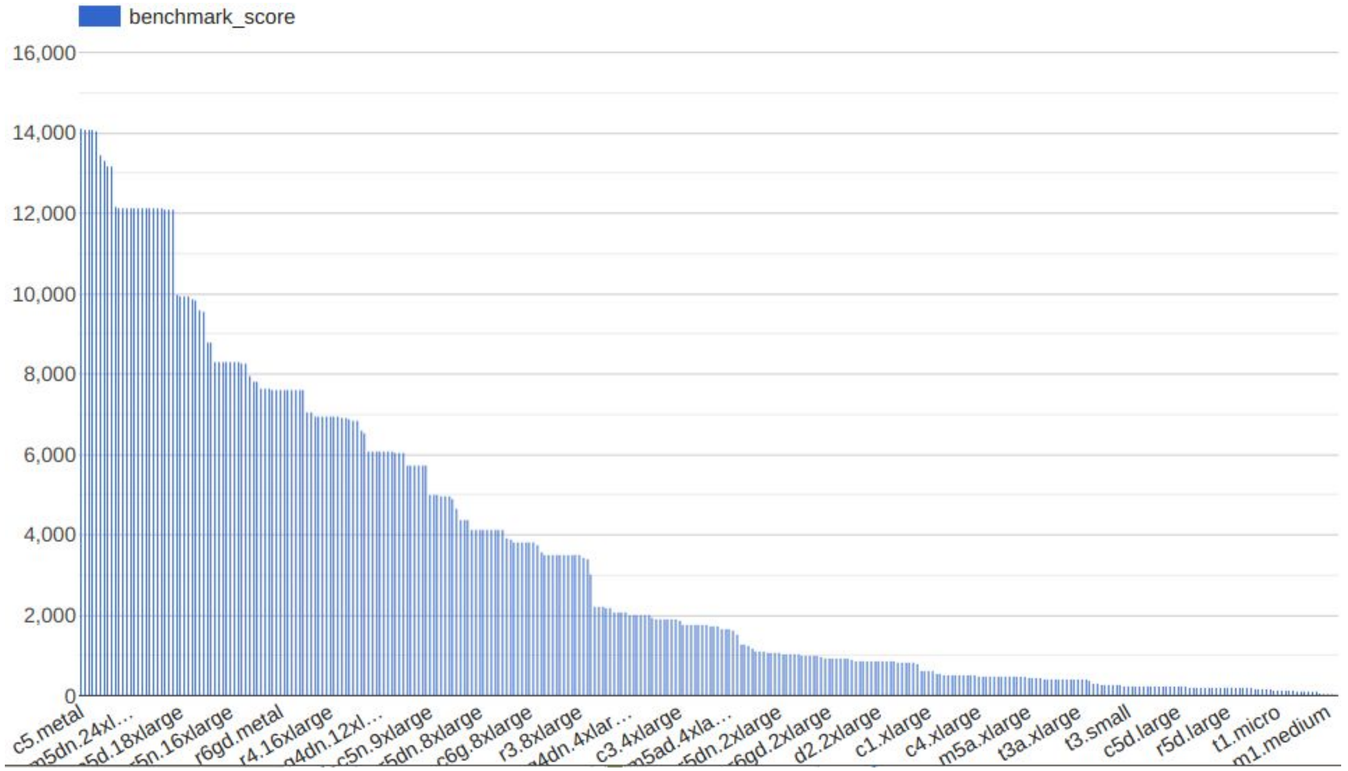


- A letárolt metrikák alapján akár meg is tudnánk jósolni, hogy egy job mennyi ideig fog futni egy adott instance-on, ha tudnánk azok relatív teljesítményét...

SYSTEM1

Cloudperf

Open source projekt, amely a publikus felhős erőforrások (jelenleg AWS EC2) relatív teljesítményét méri



https://bra-fsn.github.io/cloudperf/benchmarks/index.html

SYSTEM1

Automatikus instance-választás

- A futásidejű adatoknak köszönhetően minden jobról tudjuk, hogy milyen gépen mennyi ideig futott, mennyi erőforrást fogyasztott és tud-e több CPU-n futni
- A cloudperf-mérések alapján van egy relatív teljesítményindexünk minden EC2 instance típusra
- A kettő kombinációja segít abban, hogy a jobnak megfelelő instance-ot válasszunk olyan módon, hogy a futás a legolcsóbb legyen, és teljesítse a timeout elvárásokat:

```
+ docker-run.py --timeout 2h --remote --remote-filter-autotune Rscript model/monetization_rpc/rpsearch_tbats.R
INFO:root:Loaded maximum memory usage (7.66 GiBs)
INFO:root:Loaded maximum CPU usage (200.00%/2 vcpu)
=> Run-time statistics for max CPU usage and duration over different instances
  instanceType  used_cpus  duration_max  benchmark_score  vcpu
0     r5.large          2          5719         220.6000     2
2    r5d.large          2          5705         213.7900     2
=> Removed outliers:
  instanceType  used_cpus  duration_max  benchmark_score  vcpu
0     r5.large          2          6130         220.6000     2
2    r5d.large          2          5821         213.7900     2
=> Autotuned vcpu max/tested: 2/2, minimum memory: 10 GiBs
=> Minimum score needed for runtime target of 5760s is: 211
=> Minimum score is lower than the mean of 2 vCPU instances: 283, job could scale to 2/2 CPUs
=> Starting r5.large spot:True price:0.07/0.13 perf/price:2941 2c score:221 vcpu:2 mem:16.00 AZ:us-west-2c AMI
=> Waiting for instance i-03b9ca99d74a03a05 to be ready (10.150.144.11)
```

SYSTEM1

Továbbiak

- Több régió engedélyezése
- Multi cloud (GCP, Azure..)
- Szofisztikáltabb CPU mérés
- A következő evolúciós lépcső egy service lehetne, a futtatással kapcsolatos API-jal (instance indítás, job outputok menedzselése, background job, időzítés stb)
- Többféle policy (minél gyorsabban, minél olcsóbban, most fusson, vagy akármikor (amikor olcsó) fusson a job stb)
- Nagyobb igény esetén dedikált fizikai szerverek használata

Köszönöm a figyelmet!

Data Science infrastruktúra skálázása a felhőben

Budapest Data Forum 2020
attila.nagy@system1.com

SYSTEM1