AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

CMP326-R2

Large-scale batch processing with AWS Batch

Ala Abunijem (he/him)

Principal HPC Go to Market Specialist
AWS

Pierre-Yves Aquilanti, Ph.D.

Principal Specialist Solutions Architect, HPC
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

I. What is AWS Batch?

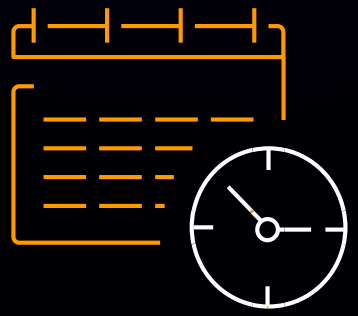
II. What's new with AWS Batch

III. Introducing fair-share scheduling

IV. Tips to run at scale using AWS Batch

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is AWS Batch?



Job scheduler



Orchestrator

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

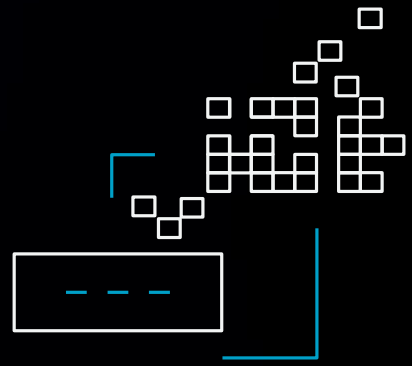
Who uses AWS Batch?



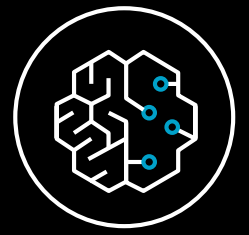
Autonomous vehicle ML and simulation



Gene sequencing & drug discovery



Big data



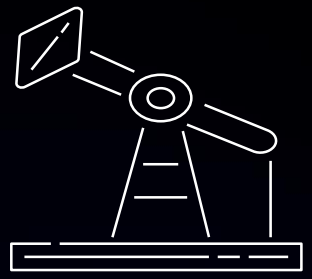
Machine learning



Financial risk analysis



EDA, CAD, FDC



Renewable energy, oil, and gas exploration



Weather simulation

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why AWS Batch?



Fully managed



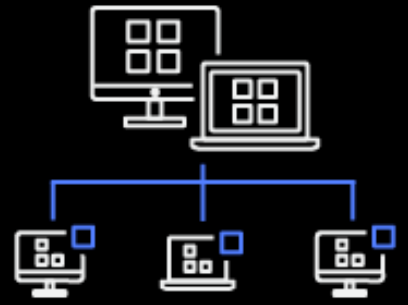
Integrates with AWS services



Cloud native, optimizes resource provisioning



Cost efficient



High scalability

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What’s new with AWS Batch

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

New features that improve scalability

1. Job definition
   - **Amazon EFS per-job support**
2. Job
   - **Higher RunTask TPS**
   - **Enhanced ListJob API**
3. Job queue (JQ)
   - **Increased JQs from 20 to 50 JQ**
   - **Fair-share scheduling**
4. Compute environment (CE)
   - **Faster scaling cycles**
   - **Fargate performance**
   - **Enhanced update CEs**



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless vs. Amazon EC2



JQ1

Large scale workload
More knobs to control

Purchasing models

EC2 On-Demand

EC2 Spot

Compute options

Arm/AMD/Intel

FPGAs GPUs

Amazon Elastic Container Service (Amazon ECS)



JQ2

Small scale workload
Easier to deploy

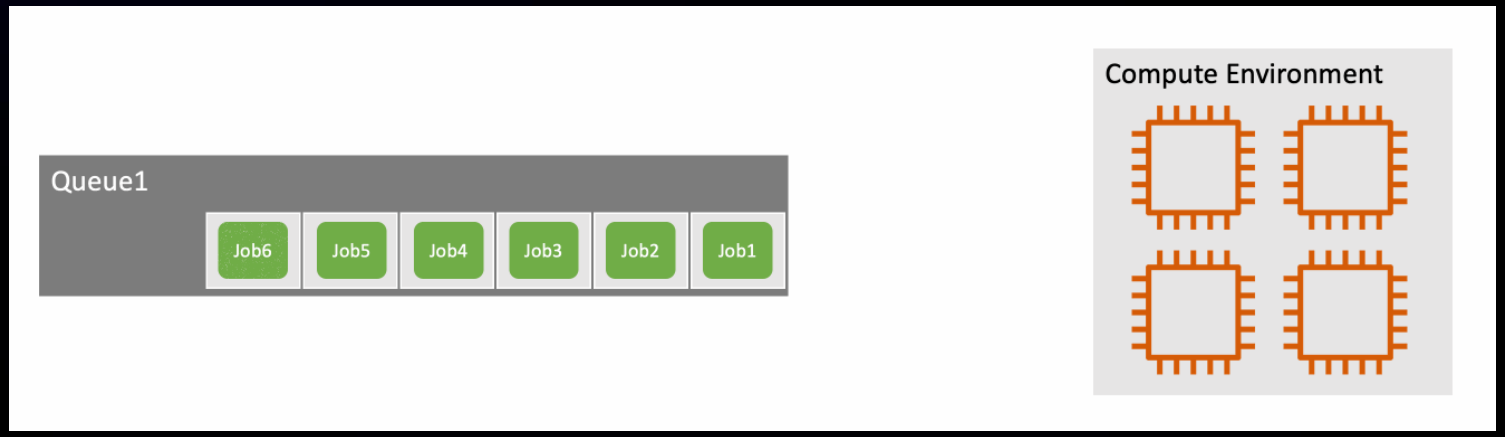
Fargate

aws

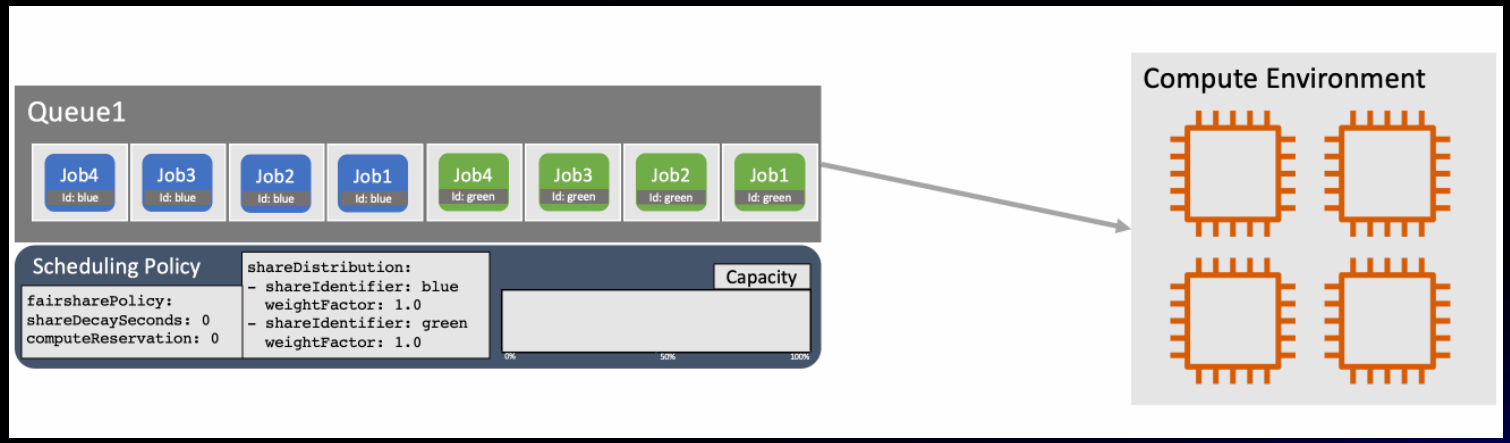
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Introducing Fair-Share Scheduling (FSS)

FIFO



Fair-share

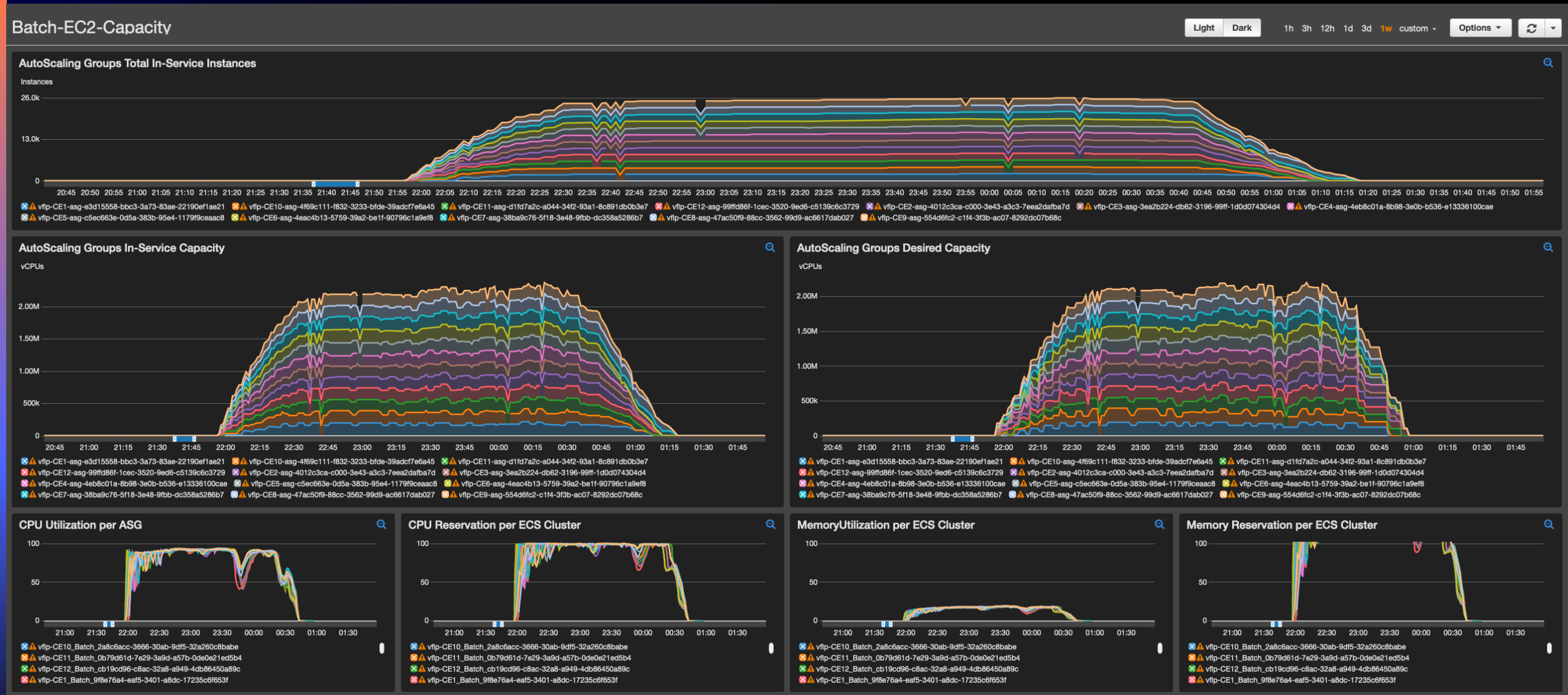


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tips to run at scale using AWS Batch

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

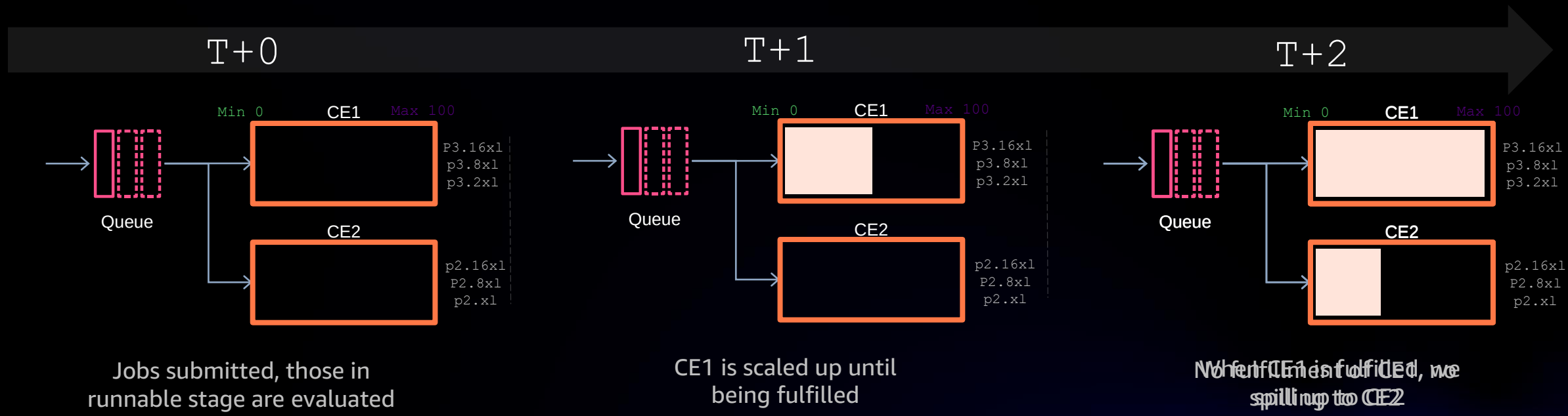
High scalability



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CEs in-order scaling example

- CEs are scaled in order by the JQ
- Switching to the next CE occurs when *maxvCPU* is met in the current CE
- CEs can be scaled concurrently to meet capacity requirements



Jobs submitted, those in runnable stage are evaluated

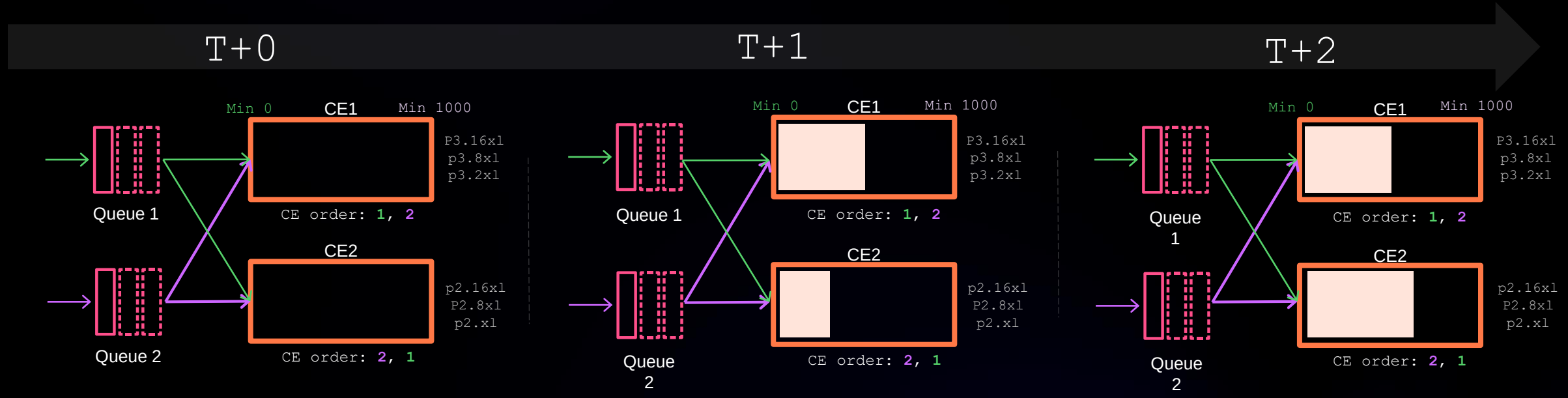
CE1 is scaled up until being fulfilled

When CE1 is fulfilled, we spill up to CE2

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CE/JQ interleaving technique

- This technique helps for large capacity acquisition and fast scale-up
- JQ to CE order defined in a rolling fashion, each JQ scales a CE
- All JQs attached to each CE (jobs can be scheduled on each CE)



Jobs are distributed to both queues at the same time

Each queue scales the CEs in order, order varies by queue

Jobs can be scheduled on both CEs to be executed

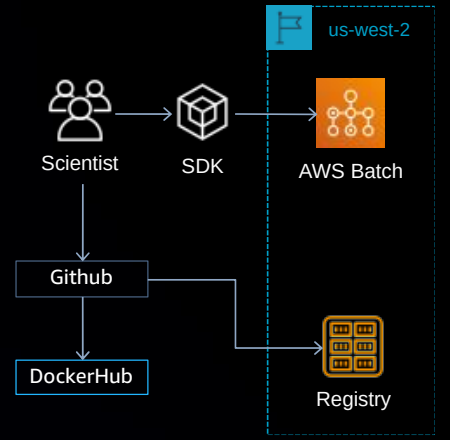
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Containers structure and storage

```
ubuntu@ip-172-31-79-173:~$ docker images
REPOSITORY                                  TAG                 IMAGE ID            CREATED             SIZE
nvc                                         1.15.1              6ddfe8ed            12 days ago         32.9GB
ubuntu@ip-172-31-79-173:~$ docker history 6ddfe8edbeb3
IMAGE               CREATED             CREATED BY                                      SIZE
6ddfe8edbeb3        12 days ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENTRYPOINT ["/scripts/ent…   0B
<missing>           12 days ago         |3 archive_dir=D                  driv    ve…   77B
<missing>           12 days ago         /bin/sh -c #(nop) WORKDIR /d                    0B
<missing>           12 days ago         /bin/sh -c #(nop)  USER driveconst              0B
<missing>           12 days ago         /bin/sh -c #(nop) COPY file:741dcfa00aff76a0…   2.16kB
<missing>           12 days ago         |3 archive_dir=D                  dri    _ve…   152MB
<missing>           12 days ago         |3 archive_dir=D                  dri    _ve…   205MB
<missing>           12 days ago         |3 archive_dir=D                  dri    _ve…   530MB
<missing>           12 days ago         /bin/sh -c #(nop)  ARG                          0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV DDS_CONFIG=/d            0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV DDS_DOMAIN=1066          0B
<missing>           12 days ago         |2 archive_dir=D                  008 drivesim_ve…   0B
<missing>           12 days ago         |2 archive_dir=D                  008 drivesim_ve…   887B
<missing>           12 days ago         |2 archive_dir=D                  008 drivesim_ve…   0B
<missing>           12 days ago         /bin/sh -c #(nop) COPY dir:c874f713e6155ef38…   26.8GB
<missing>           12 days ago         /bin/sh -c #(nop)  ARG archive_dir              0B
<missing>           12 days ago         |1                  ion=1.15.1-internal /bin/sh …   2.11kB
<missing>           12 days ago         |1                  ion=1.15.1-internal /bin/sh …   784B
<missing>           12 days ago         |1                  ion=1.15.1-internal /bin/sh …   2.33kB
<missing>           12 days ago         |1                  ion=1.15.1-internal /bin/sh …   336kB
<missing>           12 days ago         /bin/sh -c #(nop)  ENV HOME=/c                  0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV GID=1000                 0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV UID=1000                 0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV USER=d                   0B
<missing>           12 days ago         /bin/sh -c #(nop)  LABEL organization=          0B
<missing>           12 days ago         /bin/sh -c #(nop)  ARG                  sion    0B
<missing>           2 months ago        /bin/sh -c echo "                               9.03kB
<missing>           2 months ago        /bin/sh -c echo "PS1=\"\\[\\033[01;31m\\][\…   3.22kB
<missing>           2 months ago        /bin/sh -c echo "PS1=\"\\[\\033[01;33m\\][\…   2.31kB
<missing>           2 months ago        /bin/sh -c #(nop) COPY file:ecccda9e8f2129f1…   139B
<missing>           2 months ago        /bin/sh -c #(nop)  ENV                  CAPABIL…   0B
<missing>           2 months ago        /bin/sh -c #(nop) COPY file:052cbddc3360723a…   114B
<missing>           2 months ago        /bin/sh -c #(nop)  ENV LD_LIBRARY_PATH=/usr/…   0B
<missing>           2 months ago        /bin/sh -c touch /tmp/temp.cpp &&    emcc /t…   7.75MB
<missing>           2 months ago        /bin/sh -c #(nop)  ENV EM_NODE_JS=/3rd/emsdk…   0B
```

Layers created on RUN, COPY, and ADD

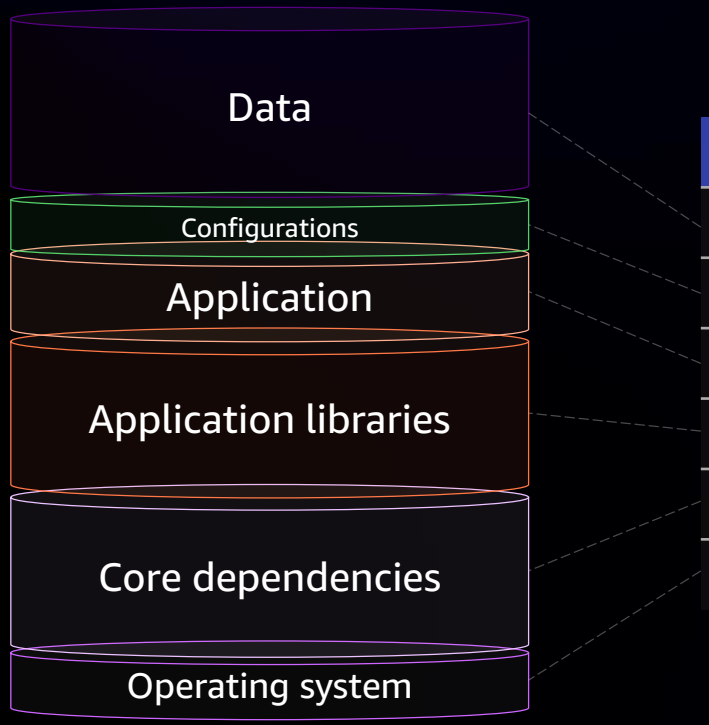
- Prefer Amazon ECR or Amazon S3 for Batch
  - DockerHub throttles under load, private registries can suffer



- Fewer even layers is better
  - Docker requests layers in parallel (1 layer = 1 request)
  - Even layers provide a better load distribution

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using machine images and containers



Data type	Size	Change frequency
Input data	20 GB (r+w)	Runtime — 20 min. per job
Configurations	3 MB	Container — Weekly
Application	1 GB	Container — 5 min.
Application libraries	4 GB	machine images — Weekly
Core dependencies	5 GB	machine images — Biweekly
Operating system	500 MB	machine images — Monthly

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Allocation strategies

- `BEST_FIT` (default in the CLI)

  Pick the least number of instances that can fit the jobs requirements at the lowest cost regardless of the type and size within your selection of instances; will diversify across instance families

- `BEST_FIT_PROGRESSIVE` (default for OD in the console)

  Same as best fit but will select instances of the same family in priority, then look at other families if $/vCPUs and requirements cannot be met

- `SPOT_CAPACITY_OPTIMIZED` (default for Spot in the console)

  Pick instances in chosen families and focus on pools with lower chances of interruptions based on historical data

https://aws.amazon.com/blogs/compute/optimizing-for-cost-availability-and-throughput-by-selecting-your-aws-batch-allocation-strategy/

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Ala Abunijem
aabunij@amazon.com

Pierre-Yves Aquilanti, Ph.D.
pierreya@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.