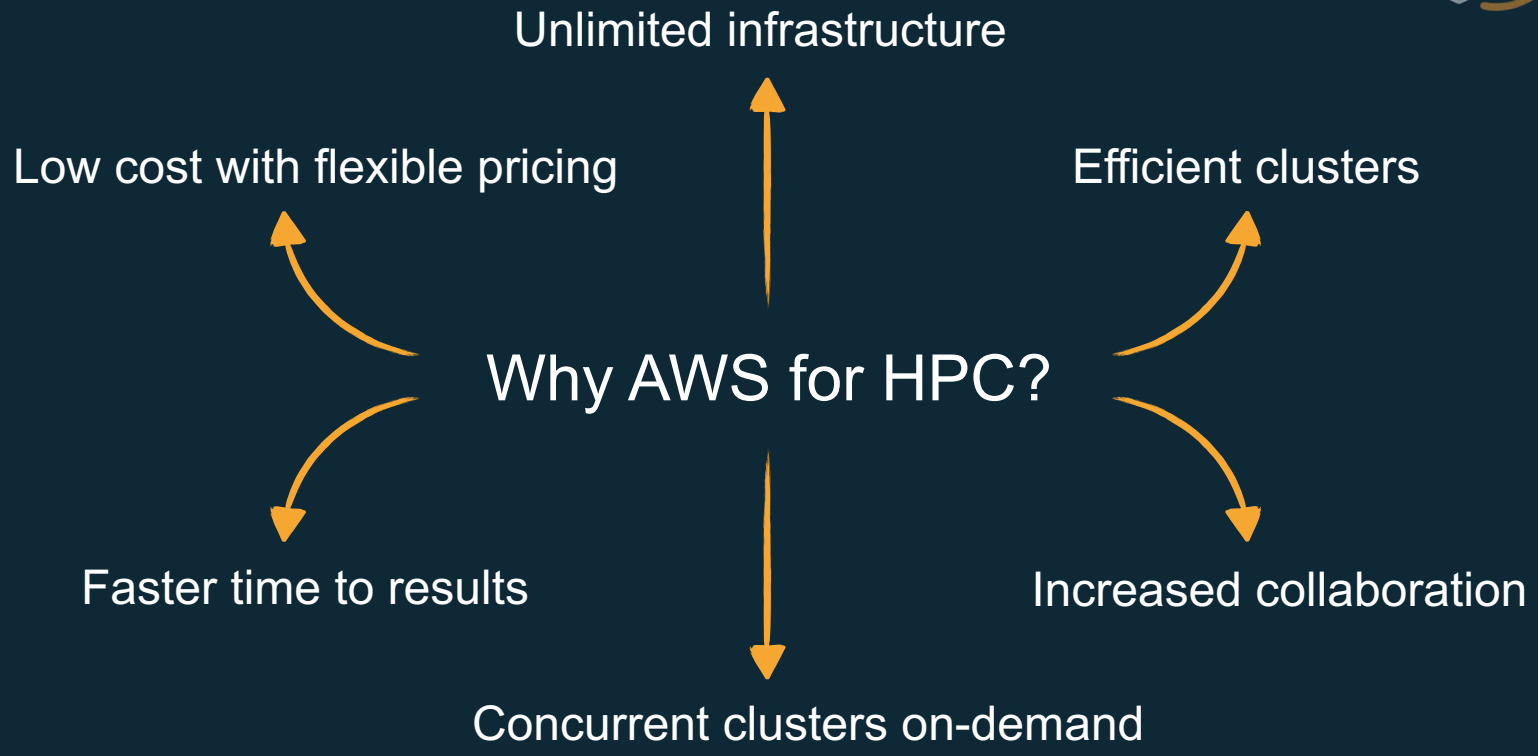
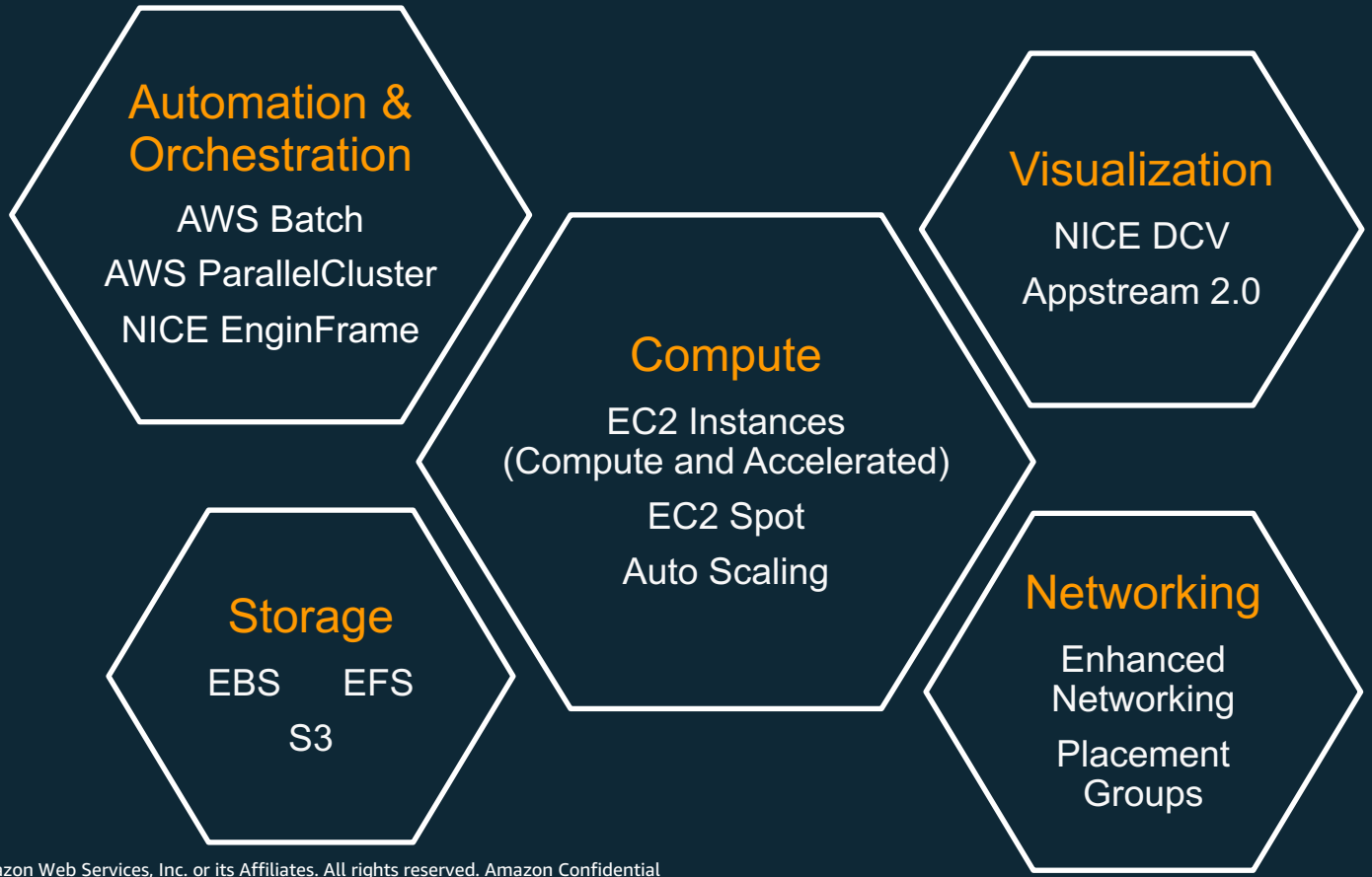
AWS Batch and AWS ParallelCluster

Some ideas for inspiration

Luca Carrogu
Software Dev Engineer – carrogu@amazon.com



AWS HPC Solution Components



AWS Batch

Introducing AWS Batch

Fully Managed

No software to install or servers to manage. AWS Batch provisions, manages, and scales your infrastructure

Integrated with AWS

Natively integrated with the AWS Platform, AWS Batch jobs can easily and securely interact with services such as Amazon S3 and DynamoDB

Cost-optimized Resource Provisioning

AWS Batch automatically provisions compute resources tailored to the needs of your jobs using Amazon EC2 and EC2 Spot

AWS Batch Concepts

- Job Queue
- Compute Environments
- Job Definitions
- Jobs
    - Single jobs vs Array jobs vs Multi-node Parallel jobs
- Scheduler

aws

Job Queues

Jobs are submitted to a **Job Queue**, where they reside until they are able to be scheduled to a compute resource. Information related to completed jobs persists in the queue for 24 hours.

```
$ aws batch create-job-queue --job-queue-name genomics
--priority 500 --compute-environment-order ...
```

Compute Environments

Job queues are mapped to one or more **Compute Environments** containing the EC2 instances used to run containerized batch jobs.

Managed compute environments enable you to describe your business requirements (instance types, min/max/desired vCPUs, and **EC2 Spot** bid as a % of **On-Demand**) and we launch and scale resources on your behalf.

Alternatively, you can launch and manage your own resources within an **Unmanaged** compute environment. Your instances need to include the ECS agent and run supported versions of Linux and Docker.

Job Definitions

Similar to ECS Task Definitions, AWS Batch **Job Definitions** specify how jobs are to be run. While each job must reference a job definition, many parameters can be overridden.

Some of the attributes specified in a job definition:

- IAM role associated with the job
- vCPU and memory requirements
- Retry strategy
- Mount points
- Container properties
- Environment variables

```
$ aws batch register-job-definition --job-definition-name gatk
--container-properties ...
```


aws

Jobs

Jobs are the unit of work executed by AWS Batch as containerized applications running on Amazon EC2.

Containerized jobs can reference a container image, command, and parameters.

```
$ aws batch submit-job --job-name variant-calling
--job-definition gatk --job-queue genomics
```

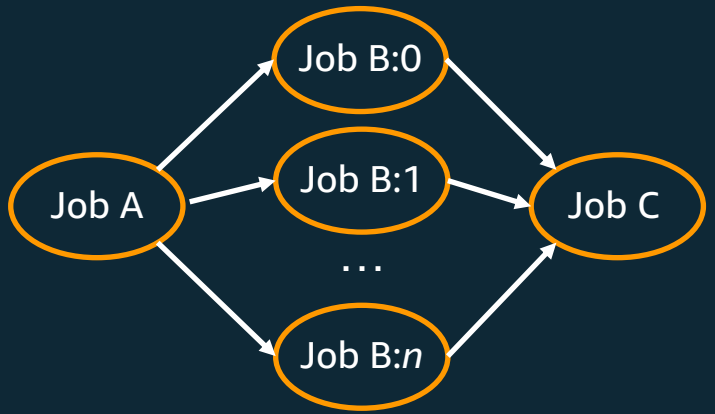

aws

Easily run massively parallel jobs

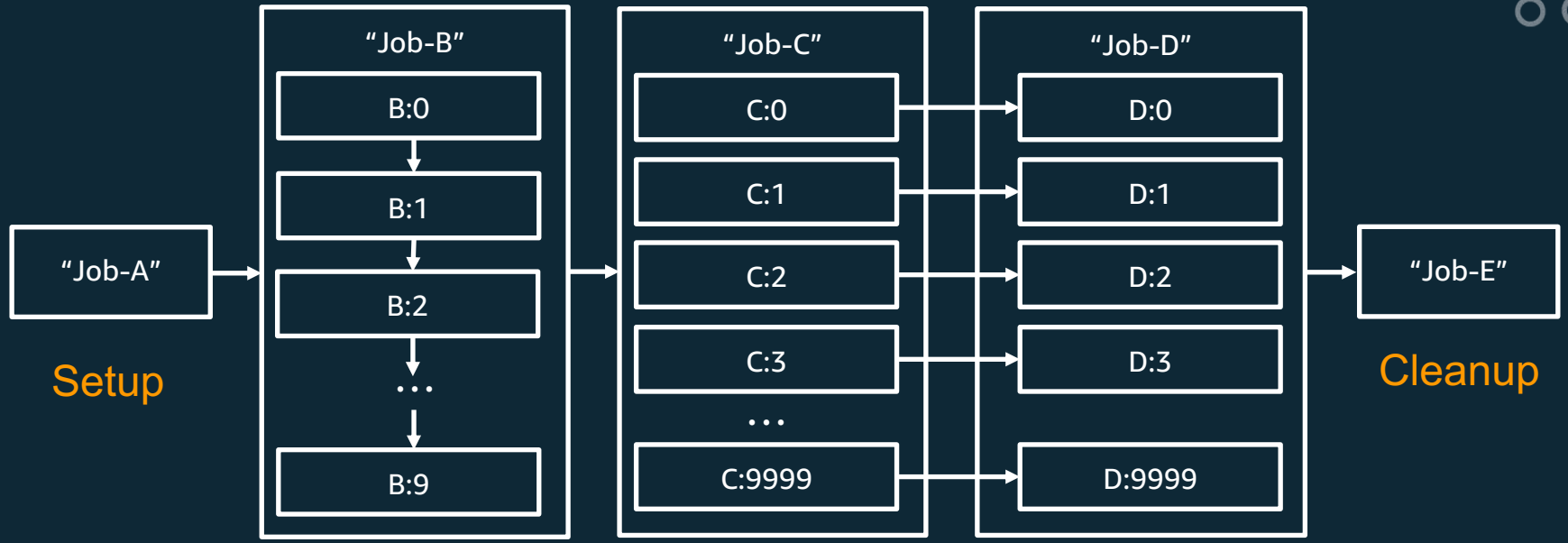
Instead of submitting a large number of independent "**simple jobs**", we also support "**array jobs**" that run many copies of an application against an array of elements.

Array jobs are an efficient way to run:

- Parametric sweeps
- Monte Carlo simulations
- Processing a large collection of objects



Array Job Dependency Models



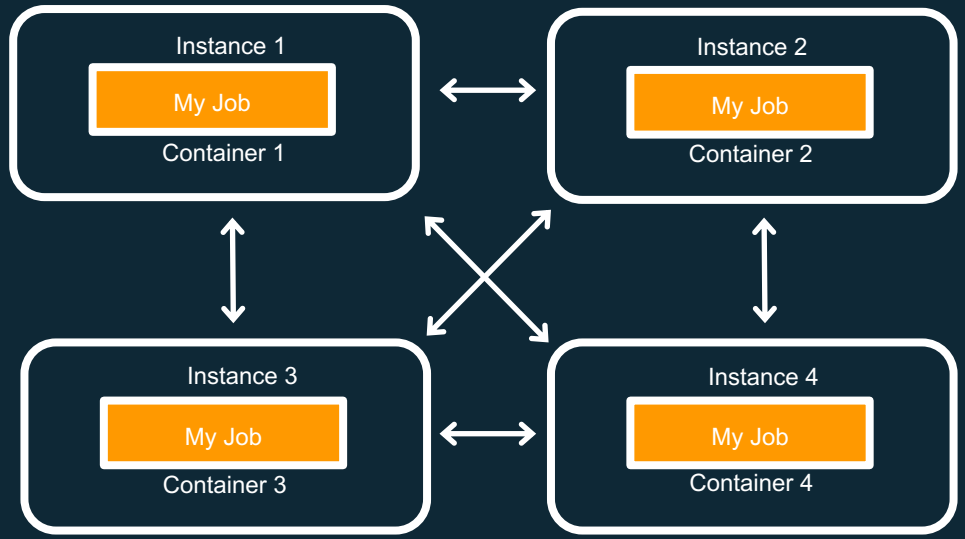

```
$ aws batch submit-job –depends-on 606b3ad1-aa31-48d8-92ec-f154bfc8215f ...
```

Multi-node Parallel Jobs

NEW

- Scale jobs across multiple instances with AWS Batch support for Multi-node Parallel (MNP) jobs

Use AWS Batch to efficiently run larger-scale tightly coupled High Performance Computing (HPC) applications and distributed GPU model training without the need to launch, configure, and manage EC2 resources directly



aws

Multi-node Parallel Jobs

NEW

```
$ aws batch register-job-definition --job-definition-name gatk
--cli-input-json file://simple-mnp.json
```

simple-mnp.json

```
{
    "type": "multinode",
    "nodeProperties": {
          "numNodes": 10,
          "mainNode": 0,
          "nodeRangeProperties": [
            {
              "targetNodes" : "0:4",
              "container": {
                "command": [
                  "whoami"
                ],
                "environment": [],
                "image": "amazonlinux",
                "memory": 128,
                "vcpus": 1
              }
            },
[…]
```

```
[…]
            {
              "targetNodes" : "5:9",
              "container": {
                "command": [
                  "whoami"
                ],
                "environment": [],
                "image": "amazonlinux",
                "memory": 256,
                "vcpus": 2
              }
            }
          ]
        }
}
```


aws

Scheduler

The **Scheduler** evaluates when, where, and how to run jobs that have been submitted to a job queue.

Jobs run in approximately the order in which they are submitted as long as all dependencies on other jobs have been met.

Typical AWS Batch Job Architecture

Input Files

S3 Events Trigger Lambda Function or CloudWatch Event to Submit Batch Job

Job Definition

IAM Role for Batch Job

Application Image

Job Resource Requirements and other parameters

AWS Batch Execution

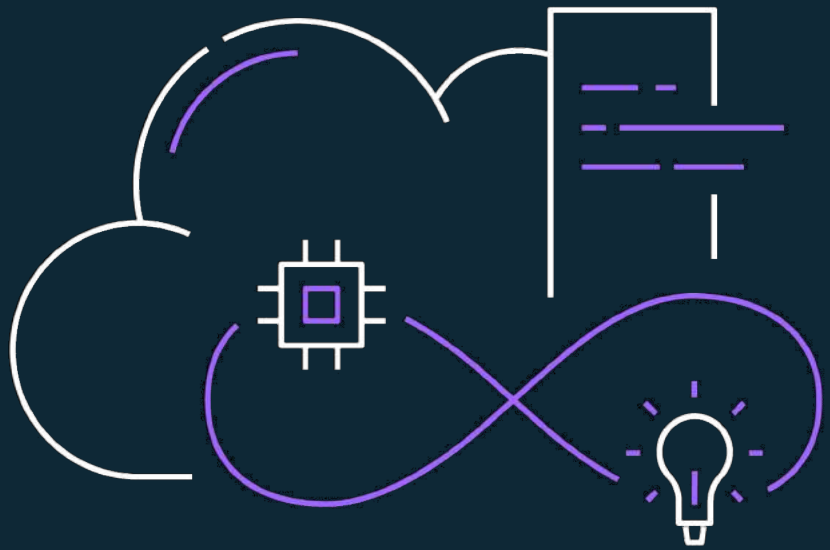
Queue of Runnable Jobs

AWS Batch Compute Environments

AWS Batch Scheduler

AWS Batch Job Output

aws



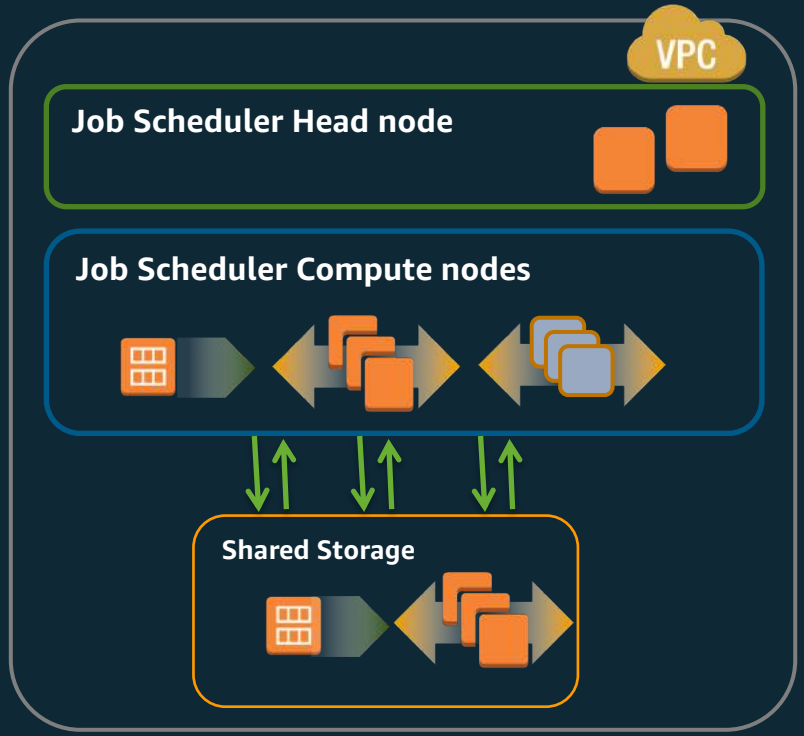
AWS ParallelCluster

aws

HPC Cluster

A high performance computing cluster is a collection of tightly coupled compute, storage, and networking resources that enable customers to run large scale scientific and engineering workloads.

- Job scheduler
- Head node
- Compute nodes
- Shared storage



Simplified HPC Cluster Management

- AWS ParallelCluster, formally known as CfnCluster, simplifies deployment of HPC in the cloud.
- Creates a HPC environment: EC2 cluster + popular HPC schedulers + Shared file system.
- Built on AWS CloudFormation, easy to modify to meet specific application or project requirements.

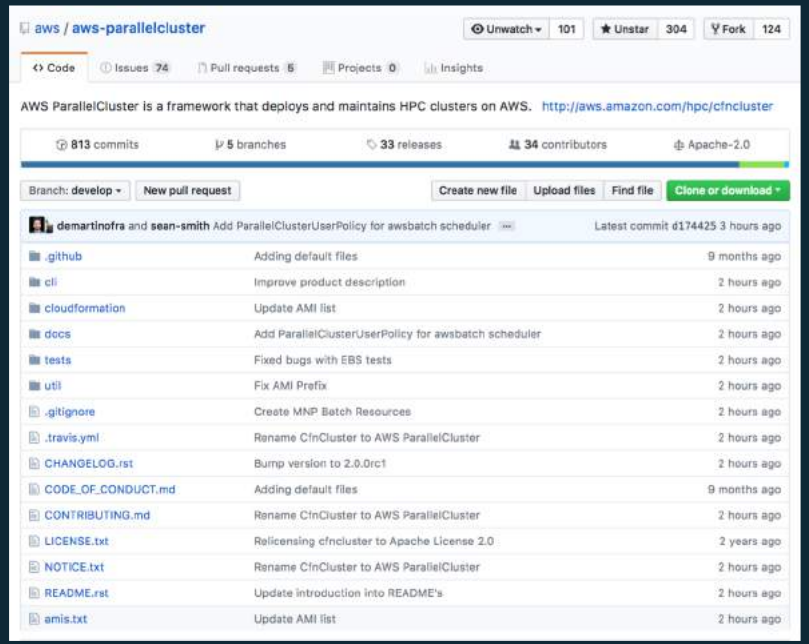
aws

Introduction to ParallelCluster

- Built on top of AWS Services
- Simple to install, easy to manage
- Everything you need to get a cluster up and running in minutes
  - Head node with scheduler
  - Shared NFS Storage from the head node to compute nodes
    - `/home`
    - `/shared`
  - OpenMPI
- Compute nodes that grow and shrink on demand

aws

Install AWS ParallelCluster



https://github.com/aws/aws-parallelcluster

Open Source!

Apache 2.0

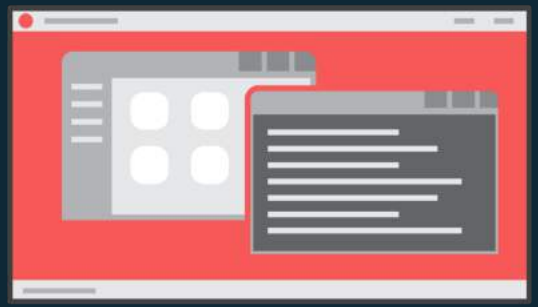
PyPi

```
$ pip install aws-parallelcluster
```

Configuration Options

- Operating System
  - Amazon Linux
  - Centos 6 – 7
  - Ubuntu 14.04 – 16.04
- Scheduler
  - SGE
  - Torque
  - SLURM
  - AWS Batch

- Storage Size & IOPS
- Multiple EBS
- Snapshots
- Scaling Limits
- Provisioning Scripts
- Custom AMI



aws

AWS ParallelCluster CLI – Create

```
$ cat .parallelcluster/configure
[...]
scheduler = sge
compute_instance_type = c5.large
initial_queue_size = 1
max_queue_size = 10
maintain_initial_size = false
base_os = alinux
[...]

$ time pcluster create myCluster
Beginning cluster creation for cluster: myCluster
Creating stack named: parallelcluster-myCluster
Status: parallelcluster-myCluster - CREATE_COMPLETE
MasterPublicIP: 52.20.70.167
ClusterUser: ec2-user
MasterPrivateIP: 172.31.11.104

real  8m44.119s
user  0m2.947s
sys   0m0.284s
```

AWS ParallelCluster CLI – Connect

```
$ pcluster ssh myCluster [-i ~/path/to/private-key.pem]

$ qhost
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD   MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------------------
global                  -               -    -    -    -     -       -       -       -       -
ip-172-31-3-43          lx-amd64        2    1    1    2  0.04    3.6G  155.1M     0.0     0.0

$ echo "/usr/lib64/openmpi/bin/mpirun -np 3 hostname >> /shared/output" | qsub -pe mpi 3
Your job 1 ("STDIN") has been submitted

$ qstat
job-ID  prior   name       user         state submit/start at     queue                          slots
---------------------------------------------------------------------------------------------------
     1  0.55500 STDIN      ec2-user     qw    11/17/2018 09:00:24                                3

$ qhost
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD   MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------------------
global                  -               -    -    -    -     -       -       -       -       -
ip-172-31-3-43          lx-amd64        2    1    1    2  0.02    3.6G  156.2M     0.0     0.0
ip-172-31-4-252         lx-amd64        2    1    1    2  0.04    3.6G  156.8M     0.0     0.0

$ cat /shared/output
ip-172-31-3-43
ip-172-31-3-43
ip-172-31-4-252
```

AWS ParallelCluster CLI – AWS Batch


```
$ cat .parallelcluster/config
[…]
scheduler = awsbatch
compute_instance_type = m4

min_vcpus = 2
desired_vcpus = 2
max_vcpus = 20
[…]

$ time pcluster create myBatchCluster
Beginning cluster creation for cluster: myBatchCluster
Creating stack named: parallelcluster-myBatchCluster
Status: parallelcluster-myBatchCluster - CREATE_COMPLETE
MasterPublicIP: 50.16.65.65
ClusterUser: ec2-user
MasterPrivateIP: 172.31.10.41

real  6m19.418s
user  0m2.795s
sys   0m0.249s
```


aws

AWS ParallelCluster CLI – AWS Batch

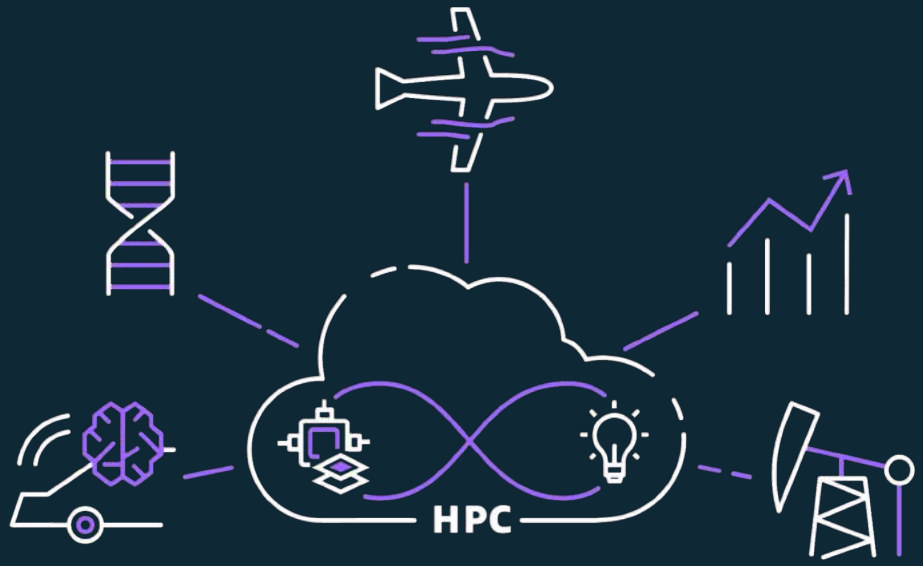

```
$ awsbhosts
ec2InstanceId        instanceType    privateIpAddress    publicIpAddress      runningJobs
-------------------  --------------  ------------------  -----------------  -------------
i-0e2293f8804593ff5  m4.large        172.31.15.248       34.237.136.89                 0

$ echo 'echo "hello $(hostname)"' | awsbsub
Job 81e37c43-1c14-43ea-9540-dcefcdedacf3 (STDIN) has been submitted.

$ awsbstat
jobId                                 jobName    status    startedAt            stoppedAt    exitCode
------------------------------------  ---------  --------  -------------------  -----------  ----------
81e37c43-1c14-43ea-9540-dcefcdedacf3  STDIN      RUNNING   2018-11-17 10:24:01  -            -

$ awsbout 81e37c43-1c14-43ea-9540-dcefcdedacf3
2018-11-17 10:36:23: Starting Job 81e37c43-1c14-43ea-9540-dcefcdedacf3
download: s3://parallelcluster-mybatchcluster-r0pxh4xqm48etnw6/batch/job-STDIN-1542450976407.sh to
tmp/batch/job-STDIN-1542450976407.sh
2018-11-17 10:36:25: hello ip-172-31-15-248
```


aws

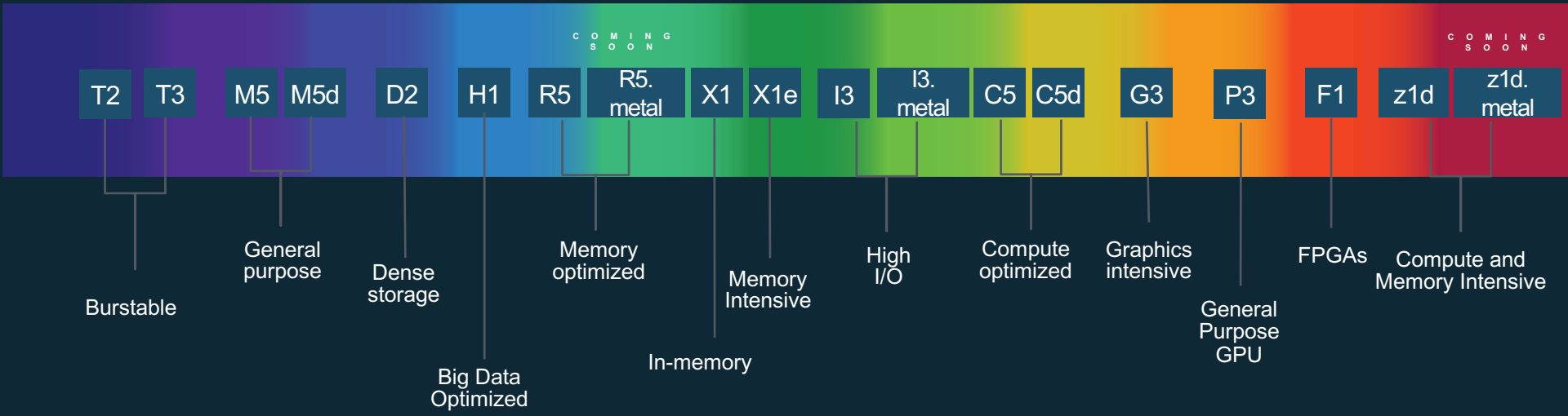


Optimizing Workloads

aws

Amazon EC2 Instances

- Select compute that best fits the workload profile
  - Match the architecture of the job, not vice versa
- Optimize price/performance of your HPC Workloads with widest range of compute instances
- Benefit from the AWS pace of innovation



Enhanced Networking for HPC

AWS Proprietary Network, 10Gbps & 25Gb

- Highest performance in largest EC2 instance sizes
- Full bi-section bandwidth in Placement Groups, with no network oversubscription

Enhanced Networking

- Over 1M PPS performance, reduced instance-to-instance latencies, more consistent network performance

EC2 to S3

- Traffic to and from S3 can now take advantage of up to 25 Gbps of bandwidth

aws

AWS Storage Options

EFS

Highly available, multi-AZ, fully managed network-attached elastic file system.

For near-line, highly-available storage of files in a traditional NFS format (NFSv4).

Use for read-often, temporary working storage

EC2+EBS

Create a single-AZ shared file system using EC2 and EBS, with third-party or open source software (e.g., ZFS, Intel Lustre, etc).

For near-line storage of files optimized for high I/O performance.

Use for high-IOPs, temporary working storage

Amazon S3

Secure, durable, highly-scalable object storage. Fast access, low cost.

For long-term durable storage of data, in a readily accessible get/put access format.

Primary durable and scalable storage for HPC data

Amazon Glacier

Secure, durable, long term, highly cost-effective object storage.

For long-term storage and archival of data that is infrequently accessed.

Use for long-term, lower-cost archival of HPC data

aws

Recommendations

- Use Placement Groups
- Choose an instance type that supports Enhanced Networking
  - C4, C5, M5, R5 are the best choices today – but always test with the latest EC2 instances
  - Consider that the bandwidth scales with instance size
- Placement Group + ENA + Instance Type size = up to 25GBps

- Use a Recent AMI/OS: Amazon Linux or Centos 7.x is Recommended
  - Use updated 3.10+ kernel

- Test with Hyper-threading (HT) on and off – usually off is best, but not always
- Use CPU affinity to pin threads to CPU cores when HT is off
- Use Processor-states to reduce processor variability

aws

Thank you