Optimizing Apache Spark

Designing Clusters for High Performance - Condensed

Designing Clusters for High Performance
WWWWHW

Before designing the cluster, we need to answer 6 questions:

- Who will be using the cluster?
- What will the cluster be used for?
- Where will the cluster [and data] reside?
- When are the results needed?
- How do I control/predict the costs?
- Why do I care about all these details?

Who will be using the cluster?

At one level, we can split on personas...

- Data Analyst
- SQL Analyst
- Data Scientist
- Data Engineers
- ...and everyone else (intentionally oversimplified)

Designing Clusters for High Performance
Who - Groups

Besides personas, we also have to consider groups:

- Different data restrictions for Group A vs Group B
- Groups with heavy cluster demands (e.g. engineers)
- Groups with light cluster demands (e.g. SQL Analyst)
- Groups that will share a cluster

What will the cluster be used for?

- Ad Hoc Data Analysis
- Reporting Generation
- Training ML & Deep Learning Models
- Structured Streaming Jobs
- Batch ETL
- Data Pipelines

How a cluster is used often follows the persona of the person using it

Designing Clusters for High Performance

Where will the cluster [and data] reside?

Where is often dictated to us, but consider these options...

- Personal PC or Laptop
- On-Prem
- Cloud (MSA, AWS, GCP, Other)
- Gov-Cloud
- Various Cloud Regions

Designing Clusters for High Performance
When are the results needed?

A Spark job's SLA generally refers to how long it takes to "deliver" data

- Real-Time
  - The data needs to be "processed" as soon as it arrives

- Near Real-Time
  - The data needs to be "processed" faster than it arrives

- ...and everything else kind of depends

Designing Clusters for High Performance
When – "It Depends" #1

Example #1: Yesterday's Sales

- Data arrives at midnight
- The report must be ready by 9 AM the following morning
- We have up to 9 hours to "deliver" the data
- Given the hours of execution, cluster stability might be a concern
- Multiple executors will help mitigate this, but we may want to limit ourselves to 4 hours of execution in case it has to be reran
- In this case a job-specific cluster sized and tuned to 4 hours of execution would be enough to support retrying the job
- There is no need/harm to tune to 1 hour of execution

Designing Clusters for High Performance
When – “It Depends” #2

Example #2: Last Month’s Sales

- Data is collected over the course of the month (1st to ~31st)
- The report must be ready by the 7th of the following month
- We have up to 7 days to “deliver” the data
- Our untuned implementation takes 24 hours to complete
- A commodity or even a shared cluster would suffice for this job
- If performance is impacted by low memory (e.g. spill) or other jobs, there is still plenty of time. A job-specific cluster may be unwarranted.
- Prudence would dictate that one not tune this job
- The cost of tuning this job is not justifiable given its SLA and possible labor

How do I control/predict the costs?

The price between a **Level-N** VM and a **Level-N+1** VM is 2x the cost, with 2x the resources

Level	Cores	Size	Cloud-A		Cloud-B	
1	4	Small	30.5 GB	$0.266 / hour	28 GB	$0.299 / hour
2	8	Medium	61.0 GB	$0.532 / hour	56 GB	$0.598 / hour
3	16	Large	122.0 GB	$1.064 / hour	112 GB	$1.196 / hour

Designing Clusters for High Performance
Costs - Actual Consumption Cost

It's all about compute-time!

Assume you have a job with 256 partitions and that each partition takes 2 minute to process.

Level	Cores	VMs	Max Compute (cores * VMs)	Iterations (max/part)	Actual Durations (iterations * min)	~Price/Hour (level $ * VMs)	VM Costs (VMs * dur * price / 60)
1	4	1	4	64	128 minutes	$0.283	60¢
1	4	64	256	1	2 minutes	$0.283	60¢
2	8	1	8	32	6[illegible] minutes	$0.565	60¢
3	16	1	16	[illegible]	[illegible]utes	$1.130	60¢
3	16	8	12[illegible]	[illegible]	[illegible]es	$1.130	60¢

And it's always 512 minutes

Designing Clusters for High Performance
Costs - Developer Costs

Another factor to consider is the cost of the developers:

- What are they doing when the job is running?
- How much time does it take to tune?

Let the money/costs decide !!!

Level	Cores	VMs	Max Compute (cores * VMs)	Iterations (max/part)	Actual Durations (iterations * min)	~Price/Hour (level $ * VMs)	Costs	ur (dur / 60)
1	4	1	4	64	128 min	$0.283	60¢	$106.66
1	4	64	256	1	2 min	$0.283	60¢	$1.66
2	8	1	8	32	64 min	$0.565	60¢	$53.33
3	16	1	16	16	32 min	$1.130	60¢	$26.66
3	16	8	128	2	4 min	$1.130	60¢	$3.33

Optimizing Apache Spark

Designing Clusters for High Performance

VM Selection

Designing Clusters for High Performance

VM Selection: Effect on Shuffles

If cost is not a primary factor, what about the effect on performance?

Level	Cores	VMs	Max Compute (cores * VMs)	Iterations (max/part)	Actual Durations (iterations * min)	Notes
1	4	1	4	64	128 minutes	No network IO
1	**4**	**64**	**256**	**1**	**2 minutes**	**Heavy network IO between 64 VMs**
2	8	1	8	32	64 minutes	No network IO
3	16	1	16	16	32 minutes	No network IO
3	16	8	128	2	4 minutes	Reasonable(?) network IO
7	**256**	**1**	**256**	**1**	**2 minutes**	**Most optimal shuffle experience**

Designing Clusters for High Performance
VM Selection: Categories

So which VM should we use? Start by breaking them down by category:

Categorization	Amazon	GBs	Cores	MS Azure	GBs	Cores
Memory Optimized	r4.xlarge	30.5	4	DS12_v2	28.0	4
Compute Optimized	c5.xlarge	8.0	4	F4s	8.0	4
Storage Optimized **Delta Cache**	i3.xlarge**	30.5	4	L4s**	32.0	4
GPU Optimized	p2.xlarge	61.0	1	NC6s_v3	112.0	1
General Purpose	m5.xlarge	16.0	4	DS3_v2	14.0	4

Only a sample of VMs are shown here. Each type is represented by N different levels of memory and cores. Availability varies by cloud.

Designing Clusters for High Performance
VM Categories

Memory Optimized
- ML workload with data caching
- If shuffle-spill remains a problem (no other mitigation strategy)
- When spark-caching is a requirement

Compute Optimized
- ETL with full file scans and no data reuse
- Structured Streaming Jobs

General Purpose
- Used in absence of specific requirements

Storage Optimized
- Optimized with Delta IO Caching !!
- ML & DL workloads with data caching
- Data Analysis / Analytics
- If shuffle-spill remains a problem (no other mitigation strategy)
  - Solid State Drives
  - Non-Volatile Memory Express (NVME)
- When spark-caching is a requirement

GPU Optimized
- ML & DL workloads with exceptionally large memory and compute requirements (presumes caching)

Designing Clusters for High Performance

Guessing Compute Level

Experimentation is easy...

- Make a guess
- If you are spilling, assume you need more RAM (unless you have skew)
- If you shuffles are slow, increase VM Level while decreasing the number of VMs
- How many iterations did it take? Increasing the VM Level or number of VMs for more cores
- Is your cluster underutilized? Reduce the VM level or number of VMs for fewer cores
- Expect this processes to take a fair amount of trial and error (aka time, aka money)

Designing Clusters for High Performance

Estimate Compute Level

1. Calculate the data's size on disk
2. **spark.sql.files.maxPartitionBytes?**
3. Compute the number of partitions or cheat and call **df.rdd.getNumPartitions( )**
4. Decide which category of VM you want
5. Based on the SLA, quota, and budget, select the type and level of VM
6. Select the number of iterations
7. Compute the number of VMs

How?

1. Assume we have 100 GB or **102,400 MB**
2. Assume **maxPartitionBytes** is **128 MB**
3. **102,400 MB** / **128 MB** = **800 partitions**
4. **Compute Optimized**
5. **Level 5**, **144 GB**, **72 cores** each
6. **2 iterations**
7. **800 par / 72 cores / 2 iterations** = **6 VMs**

8. Adjust, experiment and retest - at least time (and money) is saved by starting with a semi-reasonable configuration

databricks