Databricks Best Practises

Deploying Applications on ADB: Guidelines for Selecting, Sizing, and Optimizing Clusters Performance

These are some of the design questions we will address:

- *What type of clusters should we use?*
- *How many drivers and how many workers?*
- *Which Azure VMs should we select?*

When it comes to taxonomy, ADB clusters are divided along the notions of **"type"**, and **"mode".**

Type of clusters:

1. **Interactive Clusters -** Clusters created using UI and Clusters API
2. **Job Clusters -** Clusters created using Jobs API

Mode of clusters:

1. Standard
2. High Concurrency

Regardless of types or mode, all clusters in Azure Databricks can automatically scale to match the workload, using a feature known as Autoscaling.

Table below showcases which cluster mode is suitable for what purposes, the recommended cluster mode for Data Scientists is High Concurrency Mode as it's more suitable for Data Exploration:

	Standard Mode	High Concurrency Mode
Targeted User	Data Engineers	Data Scientists, Business Analysts
Languages	Scala, Java, SQL, Python, R	SQL, Python, R
Best Use	Batch Jobs for ETL	Data Exploration
Security Model	Single User	Multi User
Isolation	Medium	High
Table-level security	No	Yes
Query Preemption	No	Yes
AAD Passthrough	No	Yes

Batch vs. Interactive workloads

High Concurrency Clusters

There are three steps for supporting Interactive workloads on ADB:

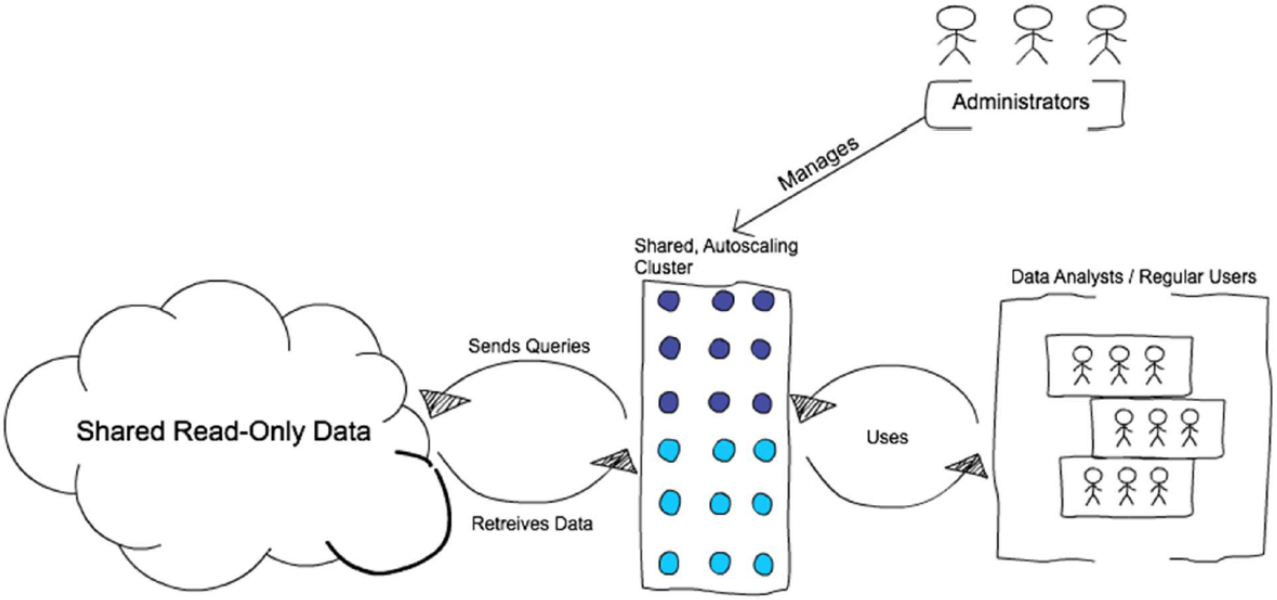
1. Deploy a shared cluster instead of letting each user create their own cluster.
2. Create the shared cluster in High Concurrency mode instead of Standard mode.
3. Configure security on the shared High Concurrency cluster, using **one** of the following options:
    - Turn on AAD Credential Passthrough if you're using ADLS
    - Turn on Table Access Control for all other stores

Workload → Attribute ↓	Interactive	Batch
Optimization Metric: What matters to end users?	Low execution time: low individual query latency.	Maximizing jobs executed over some time period: high throughput.
Submission Pattern: How is the work submitted to ADB?	By users manually. Either executing Notebook queries or	Automatically submitted by a scheduler or external workflow tool without user input.
	exploring data in a connected BI tool.	
Cost: Are the workload's demands predictable?	No. Understanding data via interactive exploration requires multitude of queries impossible to predict ahead of time.	Yes, because a Job's logic is fixed and doesn't change with each run.

Advantages of High Concurrency clusters with Table access controls or AAD Passthrough turned on (in case of ADLS):

1. **Minimizing Cost:** By forcing users to share an autoscaling cluster you have configured with maximum node count, rather than say, asking them to create a new one for their use each time they log in, you can control the total cost easily. The max cost of shared cluster can be calculated by assuming it is running X hours at maximum size with the particular VMs. It is difficult to achieve this if each user is given free reign over creating clusters of arbitrary size and VMs.
2. **Optimizing for Latency:** Only High Concurrency clusters have features which allow queries from different users share cluster resources in a fair, secure manner. HC clusters come with Query Watchdog, a process which keeps disruptive queries in check by automatically pre-empting rogue queries, limiting the maximum size of output rows returned, etc.
3. **Security:** Table Access control feature is only available in High Concurrency mode and needs to be turned on so that users can limit access to their database objects (tables, views, functions, etc.) created on the shared cluster. In case of ADLS, we recommend restricting access using the AAD Credential Passthrough feature instead of Table Access Controls.

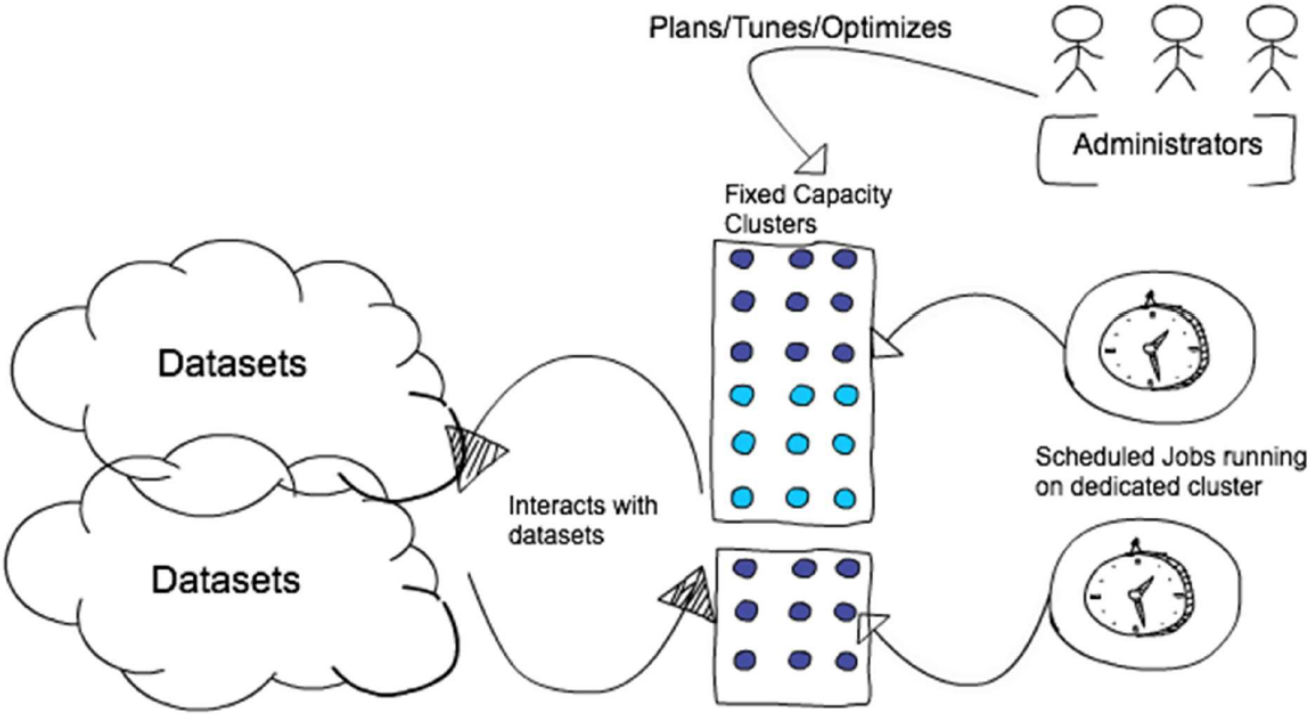
However if ADLS is being used, credential passthrough is the preferable choice over Table Access controls.



Batch ETL Workloads

Unlike Interactive workloads, logic in batch Jobs is well defined and their cluster resource requirements are known *a priori*. Hence to minimize cost, there's no reason to follow the shared cluster model and **we recommend letting each job create a separate cluster for its execution**. Thus, instead of submitting batch ETL jobs to a cluster already created from ADB's UI, submit them using the Jobs APIs. These APIs automatically create new clusters to run Jobs and also terminate them after running it. We call this the **Ephemeral Job Cluster** pattern for running jobs because the clusters short life is tied to the job lifecycle.

Azure Data Factory uses this pattern as well - each job ends up creating a separate cluster since the underlying call is made using the Runs-Submit Jobs API.



This pattern will achieve general goals of minimizing cost, improving the target metric (throughput), and enhancing security by:

1. **Enhanced Security:** ephemeral clusters run only one job at a time, so each executor's JVM runs code from only one user. This makes ephemeral clusters more secure than shared clusters for Java and Scala code.
2. **Lower Cost:** if you run jobs on a cluster created from ADB's UI, you will be charged at the higher Interactive DBU rate. The lower Data Engineering DBUs are only available when the lifecycle of job and cluster are same. This is only achievable using the Jobs APIs to launch jobs on ephemeral clusters.
3. **Better Throughput:** cluster's resources are dedicated to one job only, making the job finish faster than while running in a shared environment.

For very short duration jobs (< 10 min) the cluster launch time (~ 7 min) adds a significant overhead to total execution time. Historically this forced users to run short jobs on existing clusters created by UI -- a costlier and less secure alternative. To fix this, ADB is coming out with a new feature called Instance Pools in Q3 2019 bringing down cluster launch time to 30 seconds or less.

Correct Cluster Size by Iterative Performance Testing

It is impossible to predict the correct cluster size without developing the application because Spark and Azure Databricks use numerous techniques to improve cluster utilization. The broad approach you should follow for sizing is:

1. Develop on a medium sized cluster of 2-3 nodes, with VMs matched to workload class as explained earlier.
2. After meeting functional requirements, run end to end test on larger representative data while measuring CPU, memory and I/O used by the cluster at an aggregate level.
3. Optimize cluster to remove bottlenecks found in step 2
    - **CPU bound**: add more cores by adding more nodes
    - **Network bound**: use fewer, bigger SSD backed machines to reduce network size and improve remote read performance
    - **Disk I/O bound**: if jobs are spilling to disk, use VMs with more memory.

Repeat steps 2 and 3 by adding nodes and/or evaluating different VMs until all obvious bottlenecks have been addressed.

Azure Databricks Billing

First, it is important to understand the different workloads and tiers available with Azure Databricks. Azure Databricks is available in 2 tiers – Standard and Premium. Premium Tier offers additional features on top of what is available in Standard tier. These include Role-based access control for notebooks, jobs, and tables, Audit logs, Azure AD conditional pass-through, conditional authentication and many more.

Both Premium and Standard tier come with 3 types of workload:

- Jobs Compute (previously called Data Engineering)
- Jobs Light Compute (previously called Data Engineering Light)
- All-purpose Compute (previously called Data Analytics) The Jobs Compute and Jobs Light Compute make it easy for data engineers to build and execute jobs, and All-purpose make it easy for data scientists to explore, visualize, manipulate, and share data and insights interactively. Depending upon the use-case, one can also use All-purpose Compute for data engineering or automated scenarios especially if the incoming job rate is higher.

When you create an Azure Databricks workspace and spin up a cluster, below resources are consumed

- **DBUs –** A DBU is a unit of processing capability, billed on a per-second usage
- **Virtual Machines –** These represent your Databricks clusters that run the Databricks Runtime
- **Blob Storage –** Each workspace comes with a default storage
- **Bandwidth –** Bandwidth charges for any data transfer
- **Managed Disk**

In addition, if you use additional services as part of your end-2-end solution, such as Azure CosmosDB, or Azure Event Hub, then they are charged per their pricing plan.

There are 2 pricing plans for Azure Databricks DBUs:

1. **Pay as you go –** Pay for the DBUs as you use: Refer to the pricing page for the DBU prices based on the SKU. The DBU per hour price for different SKUs differs across Azure public cloud, Azure Gov and Azure China region.
2. **Pre-purchase or Reservations –** You can get up to 37% savings over pay-as-you-go DBU when you pre-purchase Azure Databricks Units (DBU) as Databricks Commit Units (DBCU) for either 1 or 3 years. A Databricks Commit Unit (DBCU) normalizes usage from Azure Databricks workloads and tiers into to a single purchase. Your DBU usage across those workloads and tiers will draw down from the Databricks Commit Units (DBCU) until they are exhausted, or the purchase term expires. The draw down rate will be equivalent to the price of the DBU, as per the table above.

Since, you are also billed for the VMs, you have both the above options for VMs as well:

1. Pay as you go
2. Reservations

Please see few examples of a billing for Azure Databricks with Pay as you go:

Depending on the type of workload your cluster runs, you will either be charged for Jobs Compute, Jobs Light Compute, or All-purpose Compute workload. For example, if the cluster runs workloads triggered by the Databricks jobs scheduler, you will be charged for the Jobs Compute workload. If your cluster runs interactive features such as ad-hoc commands, you will be billed for All-purpose Compute workload.

Accordingly, the pricing will be dependent on below components

1. DBU SKU – DBU price based on the workload and tier
2. VM SKU – VM price based on the VM SKU
3. DBU Count – Each VM SKU has an associated DBU count. Example – D3v2 has DBU count of 0.75
4. Region
5. Duration

Example 1: If you run Premium tier cluster for 100 hours in West EU 2 with 10 DS13v2 instances, the billing would be the following for All-purpose Compute:

- VM cost for 10 DS13v2 instances —100 hours x 10 instances x $0.598/hour = $598
- DBU cost for All-purpose Compute workload for 10 DS13v2 instances —100 hours x 10 instances x 2 DBU per node x $0.55/DBU = $1,100
- The total cost would therefore be $598 (VM Cost) + $1,100 (DBU Cost) = $1,698.

Example 2: If you run Premium tier cluster for 100 hours in West EU 2 with 10 DS13v2 instances, the billing would be the following for Jobs Compute workload:

- VM cost for 10 DS13v2 instances —100 hours x 10 instances x $0.598/hour = $598
- DBU cost for Jobs Compute workload for 10 DS13v2 instances —100 hours x 10 instances x 2 DBU per node x $0.30/DBU = $600
- The total cost would therefore be $598 (VM Cost) + $600 (DBU Cost) = $1,198.

In addition to VM and DBU charges, there will be additional charges for managed disks, public IP address, bandwidth, or any other resource such as Azure Storage, Azure Cosmos DB depending on your application.