White Paper

BlueData Storage Options

BlueData EPIC™ Software Platform

This technical white paper describes the various storage options and means by which the BlueData EPIC platform makes datasets available to containerized clusters for Hadoop, Spark, and other distributed Big Data platforms. It also describes how EPIC can use a persistent external storage pool to statefully migrate containers.

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

Table of Contents

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

1. Data Storage Considerations

We must begin our discussion about the various options for storing and accessing Big Data by considering the following common challenges:

- The access to data must be fast.
- The cost of storing data must be reasonable.

These two needs are often in conflict because fast storage is often expensive. Additionally:

- Data may need to be shared between clusters or persist beyond the lifespan of a single cluster.
- Multiple data access protocols must be supported.
- Compute and storage resources must be able to scale independently of each other.

Each of these challenges places different requirements on the storage architecture.

Data Locality

In the context of Big Data (and Hadoop in particular), the term *data locality* refers to storing the data to be analyzed on persistent storage "near" where the software doing the analysis is executing. In a deployment with physical servers on-premises, this means storing the data to be analyzed on disk drives located in the same physical server that is running the data analysis software. The typical reasons for co-locating data and compute resources are:

- Local disks are faster than networks at delivering data to the CPU.
- Big Data-related tasks spend most of their time reading data.

This deployment model achieves scalability by running many physical servers in parallel, where each server runs the same software and processes the portion of the dataset stored on that individual server. Results are aggregated and reported once the data analysis is complete. The traditional Hadoop infrastructure architecture is designed around the goal of data locality, and Hadoop is often said to "bring the compute to the data."

Figure 1 illustrates how the Hadoop Distributed File System (HDFS) and Hadoop YARN Node Manager service run a Big Data job by breaking it into tasks and then running each task on the server where the data for that task is located.

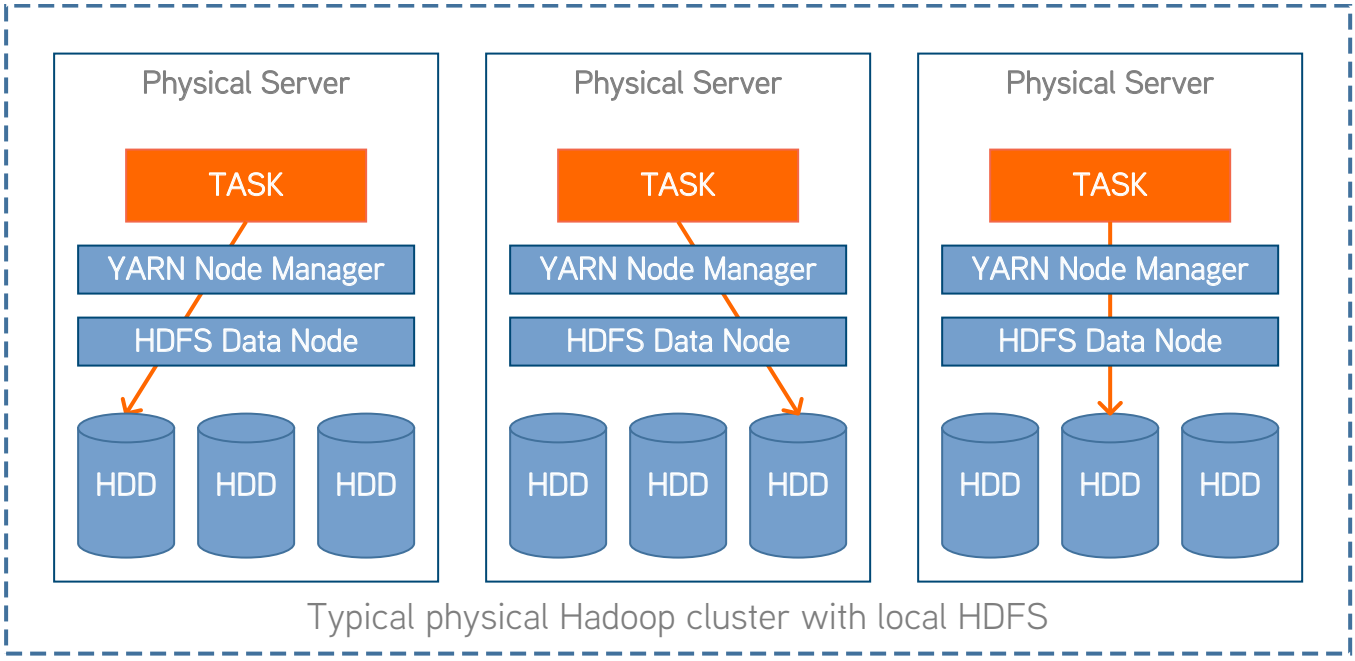

Figure 1: Physical Hadoop cluster

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

2. Local Data Storage

The BlueData EPIC software platform supports multiple storage options for running Big Data workloads such as Hadoop and Spark in containerized environments, including the data access model described in "Data Locality" on page 1. For example, HDFS can be provisioned within the Docker containers that comprise a virtual Hadoop cluster, as shown in Figure 2.

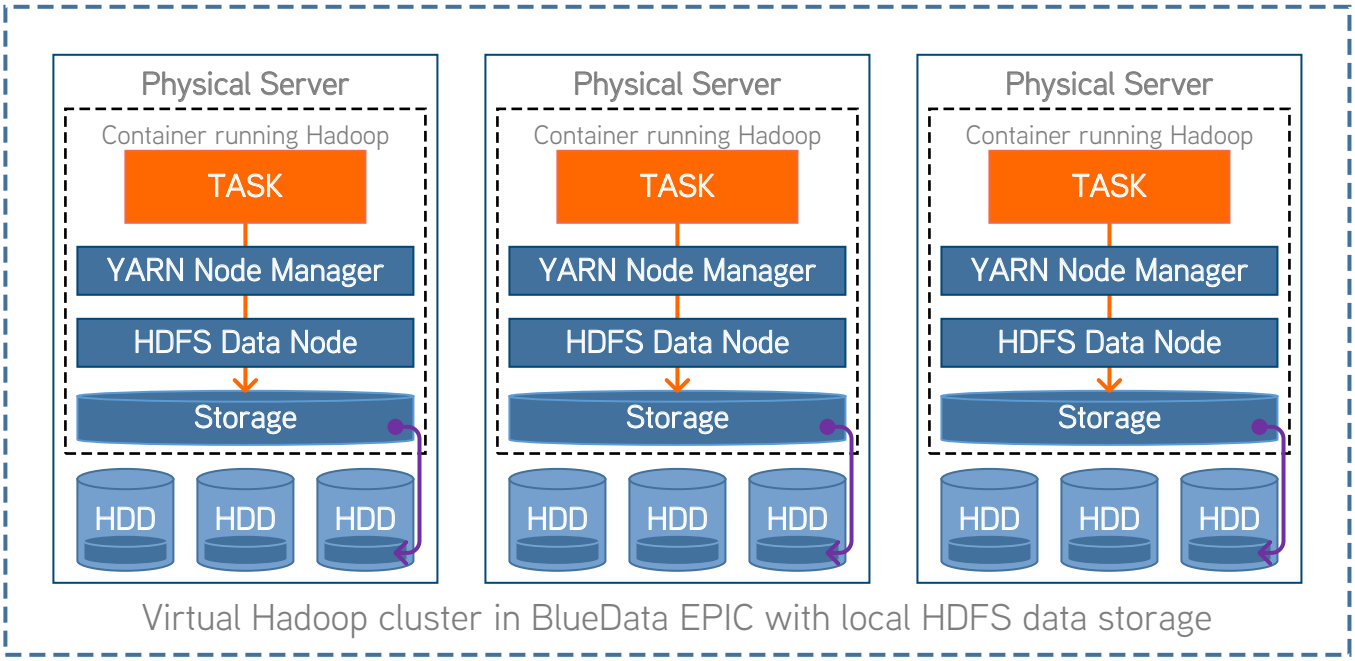

Figure 2: HDFS running inside a virtual cluster with BlueData EPIC

This method of deploying a Big Data cluster on the EPIC platform replicates that of bare metal. The underlying storage for the HDFS Data Nodes in the containers resides on local disks in the physical servers hosting those containers. BlueData EPIC refers to the set of local disk drives used for this purpose as *Node Storage*.

Node Storage

Installing an EPIC host reserves a subset of the local disks on that host for Node Storage. EPIC creates Linux physical volumes on those disks, and then uses those physical volumes to create a Linux volume group called `XqnDFUEUvqtg`. A Linux Logical Volume named `vjkprqqn` is then created from this Linux volume group. This Linux logical volume is assigned to the Linux Docker subsystem, which then uses the Linux device mapper functionality to allocate portions of the `vjkprqqn` logical volume to the containers running on that host for use as local storage within those containers.

Figure 3 shows the Linux physical volumes on an EPIC host. In this example, the single local hard disk `1fgx1ufd3` is allocated as a physical volume to the `XqnDFUEUvqtg` volume group.

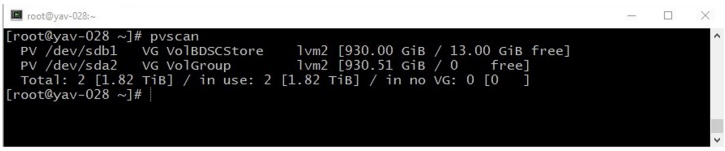

Figure 3: Linux physical volumes on a BlueData EPIC host

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

Figure 4 shows the `vjkprqqn` logical volume created from the `XqnDFUEUvqtg` volume group.

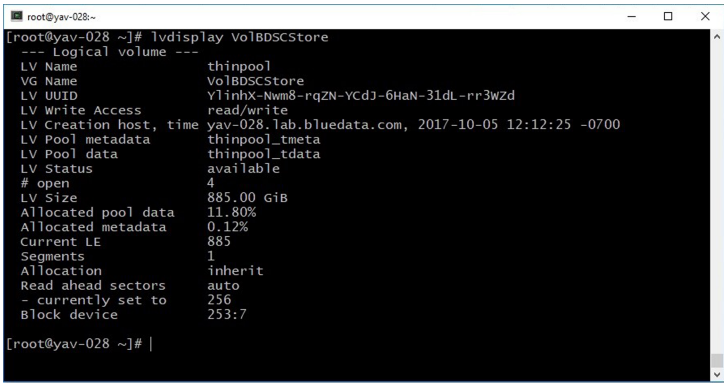

Figure 4: Thinpool logic volume created from VolBDSCtore

Figure 5 shows the device mapper devices created by the Docker system from the `vjkprqqn` logical volume for use by the containers. In this example, there are three active containers running on the EPIC host.

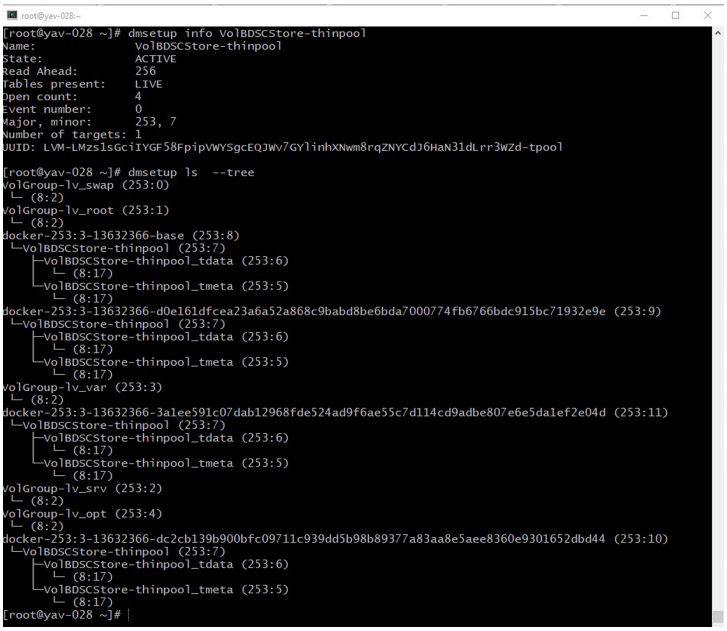

Figure 5: Device mapper devices created by the Docker system

Containerized local HDFS storage provides the flexibility and resource management power that comes from running a Big Data application in containers without sacrificing performance. However, the cost of this approach is the same as that incurred when running Big Data on a bare-metal cluster: Data must be copied into HDFS running in the cluster prior to beginning a data analysis job. The copying process can be time-consuming, depending on dataset size, which can lengthen time to insight.

It may be acceptable to pay this time and cost penalty in order to populate the local HDFS for a physical deployment of a Big Data cluster in situations where resource flexibility, agility, and maximizing the use of the available hardware are not primary concerns. It is not acceptable in a containerized environment where speed, flexibility, and agility are key. Further:

- Data ingested into the cluster HDFS file system will be lost when the cluster is destroyed.
- The compute resources used by the virtual cluster cannot be repurposed without losing the stored data.
- Compute and storage resources cannot be scaled independently. Adding storage capacity requires adding compute resources, and vice-versa.

3. Remote Data Storage

Getting the maximum flexibility from a container-based platform such as BlueData EPIC requires being able to independently scale compute and storage resources. It is also essential to be able to support the persistence of Big Data datasets beyond the lifespan of a Big Data compute cluster.

Let us reexamine the arguments for co-locating compute and storage in Big Data clusters in light of recent hardware infrastructure improvements:

- **Local disks are faster than networks at delivering data to the CPU:** Recent studies reveal that reading data from local disks is not much faster than reading that data from remote disks over the network, thanks to network infrastructure improvements. For example, this paper on disk locality from U.C. Berkeley's AMPLab shows that network speeds are increasing while disk storage speed has leveled off.
- **Big Data-related tasks spend most of their time reading data:** There is a practical limit to how many disks can be used, and technologies such as data compression and deduplication are increasingly being used to pack more data onto existing hard disks. This means that Big Data tasks are spending less time waiting for disk I/O requests to complete and more time uncompressing and then processing data. This phenomenon negates the second argument for co-locating compute and storage resources.

The fewer bytes that must be read across the network, the less time required to transfer data. The same amount of uncompressed data can be delivered to the CPU more quickly via the network.

BlueData capitalized on this by developing a container-based infrastructure software platform to support compute and storage separation for Big Data applications. The BlueData DataTap and IOBoost technologies allow Big Data clusters running on the EPIC platform to access remote data, regardless of location or format.

DataTap creates a logical data lake overlay that enables access to shared data in enterprise storage devices. This allows users to run Big Data jobs using existing enterprise storage, without needing to make time-consuming copies or transfers of data to local HDFS. IOBoost augments DataTap flexibility by adding an application-aware data caching and tiering service to ensure high-speed remote data delivery.

Figure 6 shows how DataTap and IOBoost allow the BlueData EPIC platform and remote storage resources to scale independently. Big Data datasets persist beyond the lifespan of the compute clusters, and hardware utilization is maximized.

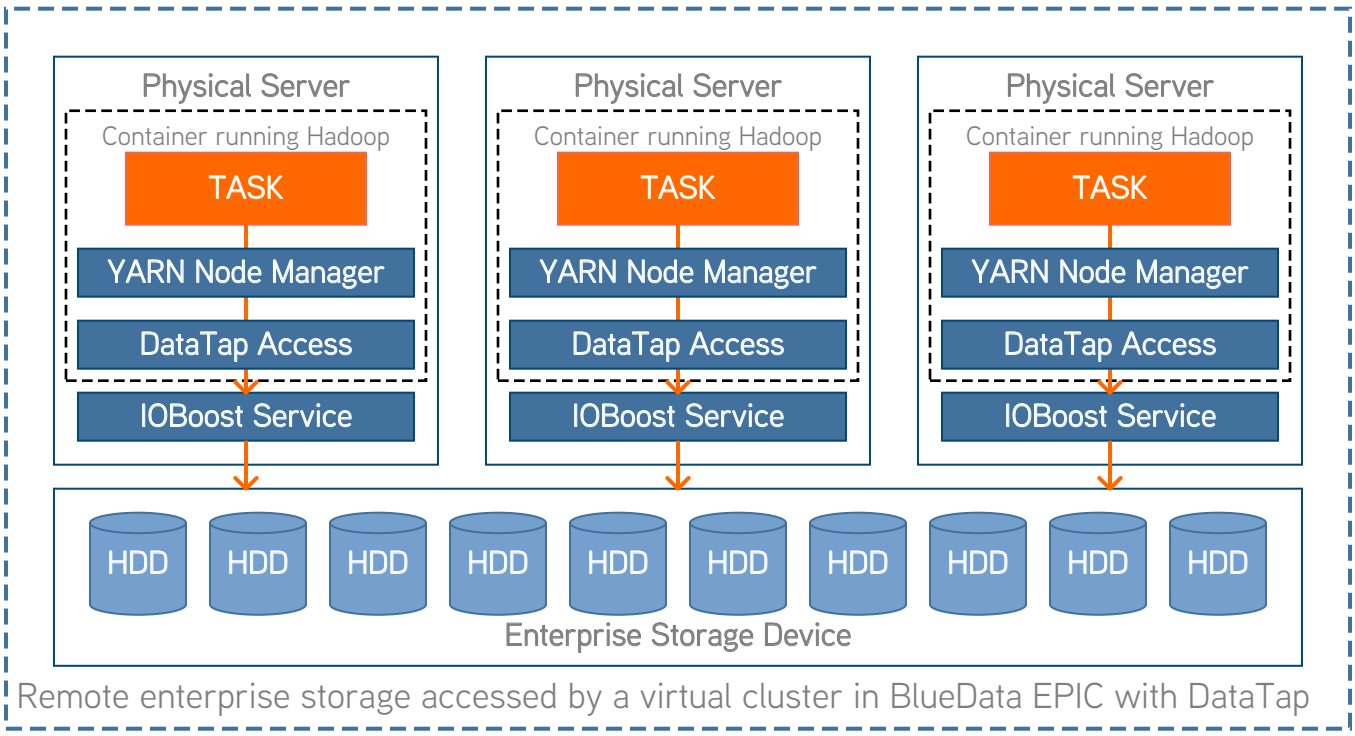

Figure 6: DataTap connection to remote enterprise storage for a virtual cluster in BlueData EPIC

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

4. Tenant Storage for Local Data Access

The unique DataTap and IOBoost technologies included in BlueData EPIC enable remote data access to achieve compute and storage separation for Big Data workloads. In early on-premises BlueData deployments, this required a high-performance network infrastructure.

Later versions of BlueData EPIC removed this requirement by developing an alternative on-premises deployment pattern that leverages the fact that EPIC is typically deployed on servers with considerable local storage capacity. This pattern avoids the problems of allocating this storage solely to the local physical cluster (e.g. local HDFS) by using an alternate storage configuration that allows customers to:

- Achieve the flexibility of compute and storage separation.
- Remove the requirement for a high-performance network between the physical hosts.
- Maintain data persistence beyond the lifespan of a virtual cluster.

To do this, EPIC deploys HDFS on the local disks within the servers running the EPIC services. This local storage allocation is referred to as *Tenant Storage*. The DataTap interface then surfaces the physical locations of the Tenant Storage data blocks to the containers that make up the virtual cluster. This allows the Big Data task scheduling software running within the containers to route Big Data tasks to the containers running on the physical servers where copies of the needed HDFS data blocks reside.

Data stored in Tenant Storage cannot be shared between tenants, in order to maintain tenant data security. This is unique with respect to the use of DataTap and Tenant Storage. Other uses of DataTap (e.g. for remote data access) allow data to be shared between tenants.

This behavior mimics bare-metal Big Data deployments, thereby preserving the performance advantages of data locality without losing the flexibility and agility of a container-based virtualized compute platform. It also allows Big Data datasets to persist beyond the lifespan of a given Big Data cluster. Figure 7 shows how this configuration works.

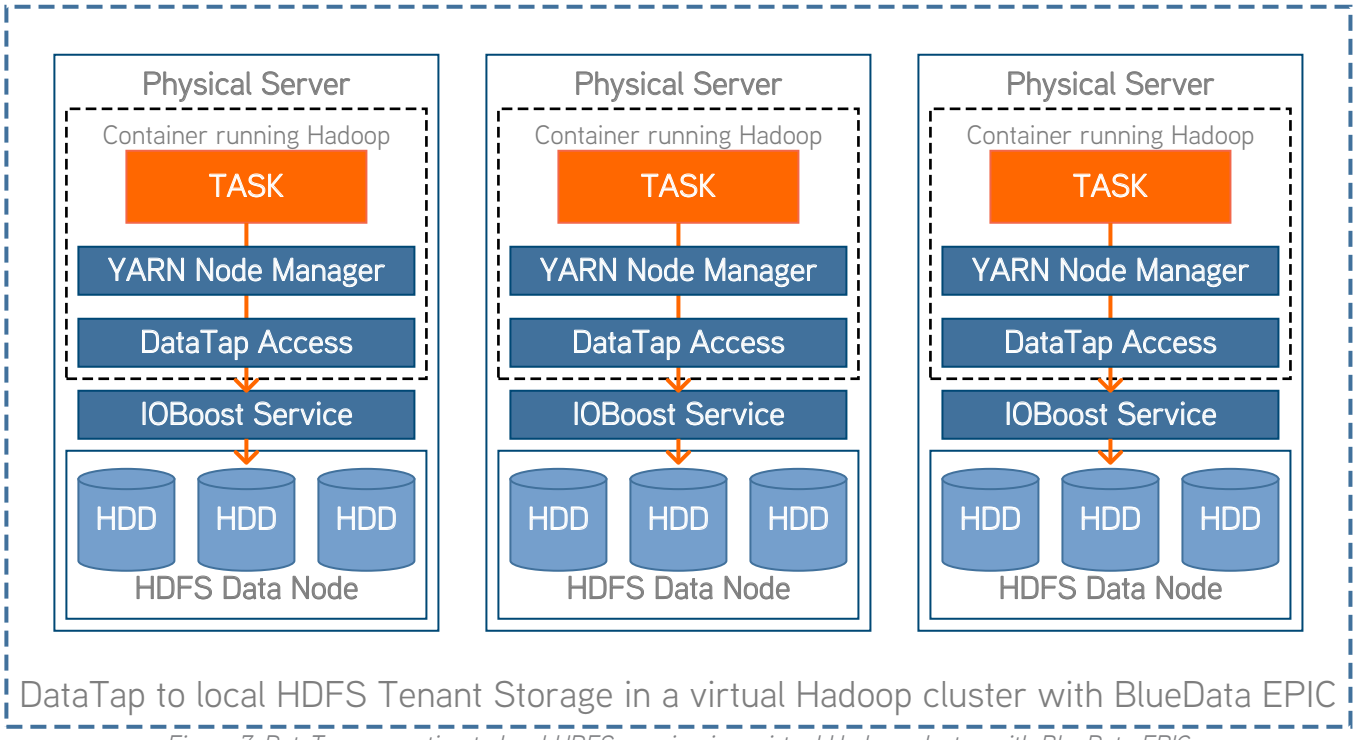

Figure 7: DataTap connection to local HDFS running in a virtual Hadoop cluster with BlueData EPIC

Intel conducted a comprehensive performance benchmark comparison of Big Data workloads running on bare metal vs. running on the BlueData EPIC platform with this configuration. This comparison determined that there was no performance loss.

5. Comparing Data Access Modes

The BlueData EPIC software platform supports three modes of data access for virtual Big Data clusters (such as Hadoop and HDFS) in an on-premises deployment:

Data access mode	Advantages and considerations
In-cluster HDFS access using local compute server storage	**Advantage:** High performance due to co-location of compute and storage resources. **Consideration:** Data not persisted beyond the life of the virtual cluster.
Remote data access using DataTap and IOBoost acceleration	**Advantages:** • Permits remote access to shared enterprise storage systems, regardless of data access protocol. • Data persisted beyond the life of the virtual cluster. • Independent scalability of compute and storage resources. **Consideration:** Requires a high-speed network connection between the compute servers and remote storage.
DataTap and IOBoost access to Tenant Storage	**Advantages:** • High performance due to co-location of compute and storage resources. • Data persisted beyond the life of the virtual cluster. **Consideration:** Local storage is required on the EPIC compute servers, meaning that compute and storage resources cannot scale independently.

These three methods of accessing data give enterprises the freedom to achieve their Big Data access goals regardless of their performance, cost, resource scalability, or protocol support constraints.

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

6. Stateful Container Migration

BlueData EPIC allows you to specify an external persistent storage pool using the **Persistent Storage** tab of the **System Settings** screen. Persistent storage exists on a remote storage resource that is pointed to but not managed by EPIC. You can create, expand, and shrink storage capacity just as you would any other resource. This feature allows you to migrate containers between hosts by (default) preserving the following critical container folders for ongoing use:

- /usr
- /opt
- /var
- /etc
- /home

The contents of other folders can be preserved during container migration by specifying their names in the metadata JSON file when creating a new application image generated using the App Workbench.

Use Cases

Big Data applications such as Hadoop and Spark offer robust high availability capabilities; however, some enterprise customers have operational requirements that require the ability to move virtual nodes/containers from one host to another. These operational requirements include:

- An EPIC host crashing and the containers that were running on that host must be redeployed on other working EPIC hosts with minimal downtime and no additional configuration required.
- One or more EPIC host(s) needs to be replaced for maintenance and/or as part of a server refresh cycle. In this scenario, all of the containers running on those hosts must be seamlessly moved to other EPIC hosts that are not being replaced, with minimal downtime to the applications running in the containers.
- Resolving a condition where there is an inability to meet the SLA for an application running in a containerized clusters (e.g. Spark or Hadoop) due to poor performance (CPU, network, or storage). This is a resource contention (bottleneck) condition that requires a re-balancing of virtual nodes onto EPIC hosts with more available resources.

Enabling Container Migration

Once you have enabled persistent storage, the next step is to create one or more flavor(s) that include at least 20GB of persistent storage. Containers created using a flavor with persistent storage enabled will be preserved as described above. You may also assign a persistent storage quota to EPIC tenants.

Note: Containers created using a flavor that does not have persistent storage enabled will not benefit from this feature, even if you later edit the flavor to enable persistent storage.

Note: You may create a flavor that specifies persistent storage even if no persistent storage has been defined in the Persistent Storage tab; however, EPIC will return an error if you attempt to use this flavor before enabling persistent storage.

Note: The persistent storage resource must have enough free capacity to accommodate the sum of all tenant persistent storage quotas or to accommodate the amount of persistent storage specified in all applicable flavor(s) times the number of containers that use the flavor(s), whichever is greater. Further, if you specify a per-tenant persistent storage quota, then that quota must be large enough to accommodate the flavor-defined persistent storage times the number of containers using the applicable flavor(s).

There are two ways to use persistent storage to migrate a container:

- **Worker Vacate:** If a Worker host goes down and some or all of the containers on that host use persistent storage, then you can click the Worker Vacate button (moving dolly) for the desired host in the **Installation** tab of the EPIC **Installation** screen to perform a Worker vacate function. All jobs running on the affected container(s) will end, but the containers themselves will be recovered as follows:
  - EPIC will not place any new containers on the affected host.
  - The protected containers are removed from the affected host.
  - EPIC automatically migrates those containers to one or more new host(s), provided that the EPIC platform has sufficient available resources, including any applicable placement constraints.

AUTHORED BY ANTHONY HERNANDEZ -- (415)786-2081 -- anthony94122@outlook.com

- **Node Migration:** This use case applies to a scenario where the hosts are functioning properly but are overburdened. In this case, the Tenant Administrator or Platform Administrator can add new hosts to the EPIC platform. Containers can then be migrated to the new host(s) on a container-by-container basis. Placement constraints apply to this type of container migration as well.

The following storage systems are supported for EPIC persistent storage:

- CEPH RBD (on-premises/hybrid only)
- NFS (on-premises/hybrid only)
- ScaleIO (on-premises/hybrid only)
- EBS (AWS only)

Migrating a container (virtual node) has the following effects:

- The cluster to which the container belongs will be stopped and unavailable until the migration completes.
- Any jobs or ActionScripts running on a cluster with one or more migrating container(s) will be lost and must be run again after completing the migration.
- Any data residing in non-persistent storage directories of a container being migrated will be lost.
- Any external host(s) that have access to the EPIC platform will be unable to access the affected container(s) until the migration process completes.
- Migrated containers maintain their configuration and IP addresses.

When migrating containers using EPIC on AWS:

- Volumes are created and tagged with the following tags:
  - / `$Crrnkecvkqp$<$DnwgFcvc"GRKE$`
  - / `$Eqpvtqnngt KF$<$>eqpvtqnngtakf@$`
  - `$Pcog$<$>kocigakf@$` -- EPIC uses this tag to find the volume ID that is used to query and delete volumes.
- EBS volumes are created and attached to the EC2 instance that hosts the migratable container(s). EPIC uses device names similar to `1fgx1ufc`, `1fgx1ufd`, and so on. AWS may internally renames these device names, such as renaming `1fgx1ufc` to `fgx1zdfc`. In these cases, EPIC will attempt to discern the naming scheme and will then use the names and will return an error if this detection fails.
- AWS allows a maximum of 26 block devices (`1fgx1ufc`, `1fgx1ufd`, etc.) to be created and attached to a EC2 instance. EPIC already uses `1fgx1ufc` and `1fgx1ufd` for node storage and tenant storage, meaning that each EC2 host can support up to 24 migratable containers.