ECX 2.0
Best Practices for Deployment and Cataloging

Table of Contents

Audience

This Best Practices Guide is intended primarily for administrators of ECX 2.0 specifically with respect to VMware and NetApp Catalog Data policies.

Concepts and Terms

ECX stores data in two volumes described herein as the Configuration volume and the Catalog volume.

The data stored by ECX can be characterized as high-level and low-level objects. NetApp Snapshots and VMware VMs are examples of high-level objects. File metadata obtained from a NetApp storage system is an example of a low-level object.

Objects are stored in catalogs. A catalog is populated by running an ECX catalog policy. (Note that Copy Data and Use Data policies will also add data to the catalog.)

Each of the two volumes contains a MongoDB database. An ECX catalog is a set of related objects that is physically realized as one or more MongoDB database collections.

Please note that the generic term "ECX catalog" denotes the *form* in which ECX stores data. It should not be confused with the ECX "Catalog volume". In fact, both the Catalog and Configuration volumes contain catalogs. We distinguish between these two volumes because of the difference in their *content*.

ECX creates and maintains search indices used by the management console's search functionality.

Best Practices for Deployment

Install the ECX Virtual Machine

Install the ECX virtual machine into a dedicated Resource Pool, using the default settings for no reservation and no limits. This can be helpful with either reserving resources for the ECX appliance or limiting its resources to prevent conflicts with other virtual machines.

Select a Datastore Type

If you are using ECX solely for search and reports, then write optimized storage is recommended for optimizing cataloging, searching, and reporting performance. However, if you are using ECX for data or application protection and recovery, you must deploy the product with Appliance Protection in mind. For details about protecting the ECX appliance itself, see Catalogic Software Knowledge Base article 47034.

What is the best virtual machine disk format?

Thick provision, Lazy Zeroed is the preferred virtual machine disk format selection. This reserves the necessary disk space with fast initialization. Eager Zeroed can be used if purposefully clearing out the space is a requirement, however, this option increases the initialization time. Thin Provision is not suggested as it imposes performance degradation on the application as new space is used.

Select Networking Options

ECX collects data from other nodes on the network. It is important for ECX to have fast access to the network where multiple parallel streams of data can be accepted for cataloging. Avoid using slow links and highly congested vSwitches and interfaces. Try to avoid aggregating ECX traffic on virtual switches and interfaces shared with other applications that have high performance demands, e.g., email, file, and other application servers.

The recommended practice is to use a dedicated IP address and DNS name for your ECX host. Using DHCP is not recommended as it may result in the appliance obtaining a new IP address that users on the network might not have access to. Most high performance and mission critical servers in an Enterprise are expected to exist at fixed, known IP addresses.

Should I start the virtual machine immediately or is further tuning required?

Deploy the virtual machine powered off. The ECX administrator should review the new VM configuration to spot-check for any concerns.

Increasing memory is acceptable. Avoid decreasing memory unless the default requirements in your environment are not acceptable.

Increasing CPU capacity is acceptable. Avoid decreasing CPU capacity.

Adding a CD-ROM/DVD virtual device is suggested and helps with future needs to update or troubleshoot the virtual machine.

Best Practices for Disk Storage

How many objects does ECX support?

As delivered, the ECX appliance supports a steady state of roughly 250 million objects. This total represents the sum of high-level and low-level objects. This estimate is based on the size of the ECX Catalog volume and the mean sizes of the objects.

Increase the Size of an LVM Volume

The Configuration volume is 100 GB in size and the Catalog volume is 250 GB. These defaults sizes were selected as a reasonable starting point for most deployments. If necessary, increase the size of these volumes at any time with the assistance of Catalogic Software Data Protection Technical Support.

The volumes are managed by the Linux Logical Volume Manager, therefore their sizes can be increased. Increasing the size of disks should only be performed with the assistance of Catalogic Software Data Protection Technical Support.

When increasing the size of one volume, increase the second volume by the same factor. For example, if you double the size of the Catalog volume, double the size of the Configuration volume.

Should I keep multiple instances of the NetApp Files Catalog?

The NetApp File Catalog, which records low-level objects in the form of storage system metadata, can grow to be quite large. It is customary for this catalog to contain hundreds of millions of objects with each object consuming roughly 1000 bytes of disk storage (100 million objects require 100 GB of disk storage). If you keep more than one instance of this Catalog, consider the effect on the consumption of

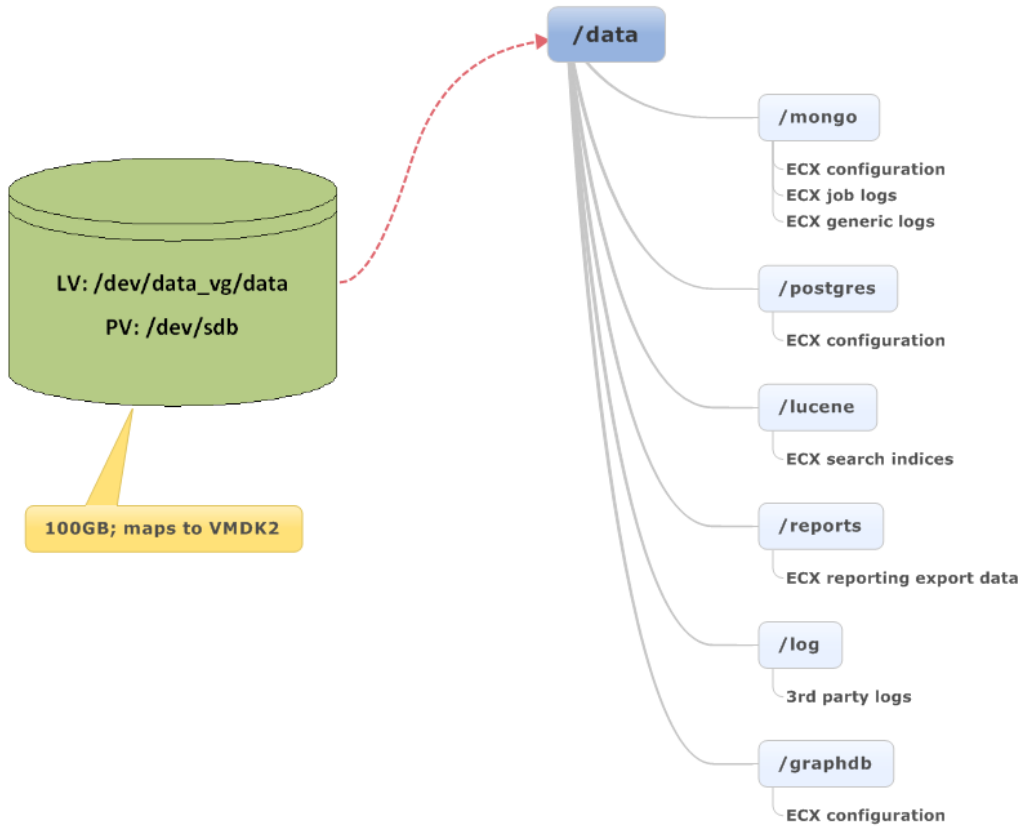
disk storage. By default, ECX keeps one instance of the NetApp File Catalog. However, because ECX deletes the old instance only *after* the creation of a new instance, *twice the size of the Catalog is temporarily required.*

Disk Storage Layout

The ECX VMware appliance is configured with three virtual disks (VMDK files). VMDK1 (30GB) is the system disk. VMDK2 (100GB) and VMDK3 (250GB) contain all of the data used by ECX.

Mount Point	Logical Volume	Volume Group	Device	VMDK	Default Size (GB)	Description
/			/dev/sda	VMDK1	30	System disk
/data	data	data_vg	/dev/sdb	VMDK2	100	Configuration
/data2	data2	data2_vg	/dev/sdc	VMDK3	250	Catalog

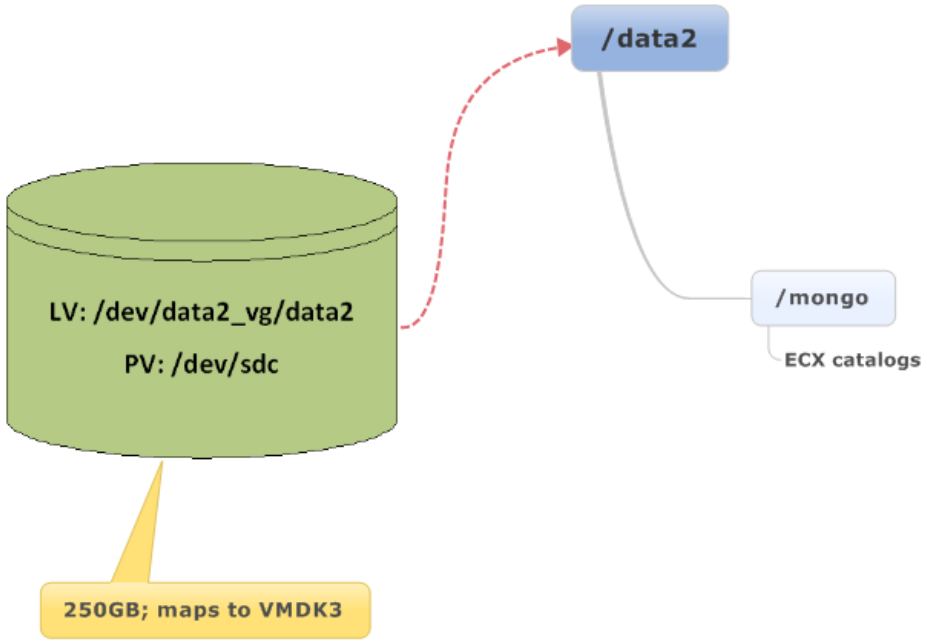
The contents of physical volume /dev/sdb, the ECX configuration volume (VMDK2), are depicted below:



Contents of the LVM ECX Configuration Volume at Mount Point **/data**

In addition to configuration data, ECX stores both search indices and exported report data on this volume. The **/lucene** directory contains search indices that are used by the ECX search facility. These indices are created by running Catalog policies. The **/reports** directory contains output created by running a report policy that specifies exporting the report data.

The contents of physical volume /dev/sdc, the ECX Catalog volume (VMDK3), are depicted below:



Contents of the LVM ECX Catalog Volume at Mount Point **/data2**

Configuration and Catalog Volume Content Considerations

The catalog policies for NetApp Storage, NetApp Files, and VMware along with Copy and Use Data generate most of the data stored by ECX. However, ECX also needs and maintains a variety of configuration information. Some of this configuration information is generated in response to user actions in ECX, for example, configuring NetApp and VMware nodes. Other configuration information is generated by ECX, for example, the creation of metadata used to keep track of its Catalogs, or log files produced by running a policy.

Almost all Catalog volume or Configuration volume information is kept in MongoDB, a scalable, NoSQL document database. A small amount is kept in PostgreSQL.

Catalog and Configuration volume information have different workload characteristics. Catalog volume information, which includes high-level and low-level objects, is read and written more frequently, and in greater quantities, than Configuration volume information. Consequently, each is kept in its own database. This separation results in reduced ECX latency when accessing Configuration volume information while cataloging policies are active.

A fundamental consideration concerning data storage is how many objects can be stored on the two ECX data volumes. While deriving an accurate estimate, you must consider several factors, some of which can be controlled by the user. Sizing factors that apply to both high-level and low-level objects include:

- Number of objects
- Storage space required by an object
- Policy run frequency
- Catalog instance retention level

Additionally, some sizing factors are unique to low-level objects:

- The file change rate of the indexed NetApp volume
- The quality of the these changes, for example, the distribution of file creation, modification, and deletion events
- The frequency of volume snapshots. This is relevant only if the policy specifies the option “Catalog all available snapshots”.
- The retention period of volume snapshots. This is relevant only if the policy specifies the option “Catalog all available snapshots”.

All of these factors must be considered in the context of a catalog steady state. This state occurs after the user has:

1. Defined and scheduled catalog policies for all nodes (NetApp and VMware) of interest, including NetApp high-level and low-level objects and VMware high-level objects
2. Run those policies for a duration sufficient to allow the indicated number of Catalog instances to be created

Each object cataloged by ECX also creates an index entry in the Lucene search indices. These search indices, which are used by the ECX search facility, live on the Configuration volume. The 100 GB default size of this volume is sufficient to support roughly 1 billion Lucene index entries. However, as noted above, this volume stores more than just the Lucene indices. The largest storage consumer on this volume after the Lucene indices is usually the configuration information kept in the MongoDB database. In the unlikely event the MongoDB consumes 50% of the volume, there is still room for roughly 500 million Lucene index entries.

Retention of Catalog Instances

Each execution of a policy creates an instance of a Catalog. The most recently run policy creates the current instance. ECX also keeps previously generated instances called “versions”.

ECX retains Catalogs based not on an explicit duration, but on the number of existing Catalog instances. For each of the three types of Catalogs, you can specify the maximum number of instances that may exist at any time. By default, ECX keeps three instances of the NetApp and VMware Catalogs, or high-level objects, and one instance of the NetApp File Catalog, or low-level objects. You can change the number of Catalog instances to retain.

Best Practices for Performance

NetApp Storage and VMware Catalogs

You should not experience performance problems when running ECX policies that catalog high-level objects. Their numbers are typically small so that throughput, or objects per hour, and elapsed time are robust.

NetApp Files Catalog Performance

Before you define any production policies for low-level objects, test ECX performance in your installation. Create a simple NetApp Files policy that catalogs at least 10 million objects. Monitor the policy's execution in ECX. Run the policy under the load conditions that are common in a production environment. Anticipate run time of 3 to 5 hours for 10 million objects.

When the job completes, messages in the job log display the throughput. This is a useful metric for estimating a policy's elapsed time.

If you intend to run NetApp Files Catalog policies on several NetApp storage systems, repeat this experiment for each storage system.

Working at a derived throughput of 2-3 million objects per hour, ECX catalogs 100 million objects in no less than 33 hours. This rate, obtained on commodity hardware, scales linearly up to a certain point. This rate has been observed for as many as 200 million objects when the ECX Catalog creation phase is write-bound, for example, when MongoDB, and not the NetApp storage system, is the bottleneck. If the NetApp storage system delivers data slowly, then the ECX File Catalog policy performance degrades accordingly. In such cases, the observed throughput is in the range of 400,000 to 1 million objects per hour. See the "**Throughput and Elapsed Time**" section for further details on ECX file cataloging performance.

As the number of concurrently cataloged objects increases, performance begins to degrade. This occurs when one policy, or several concurrently running policies, catalogs hundreds of millions of objects.

The number of operations in progress on the Monitor tab varies depending on the number of jobs currently running. ECX controls the number of jobs allowed to run. When the number of jobs exceeds the value defined by ECX, jobs marked with "Waiting" indicators display on the Monitor tab. ECX also controls the number of job tasks to run simultaneously for a given job when multiple jobs are running.

MongoDB Journaling

While it is possible to disable MongoDB journaling, there are consequences in doing so. Without journaling, MongoDB has no way to ensure data integrity after a crash. If the ECX appliance or MongoDB crashes, then you must assume that your database is corrupt. If you continue to run ECX after such a crash, your database may stop working. Depending on your business needs, this approach *may* be acceptable, but it not recommended.

Contact Catalogic Software Data Protection Technical Support for assistance in disabling journaling.

Place the MongoDB Journal File on a Separate Disk

Because there are two MongoDB databases, there are two distinct journal files. Create a symbolic link to place the journal directory on the ECX Catalog volume on a different disk. Consider the use of a solid state disk for the journal file.

Contact Catalogic Software Data Protection Technical Support for assistance in isolating the MongoDB journal file.

Select a Traversal Method When Creating a NetApp Files Catalog Policy

This option indicates the methodology to employ when cataloging Snapshots. ECX honors your preference if it is supported for the particular system configuration. If the selected preference is not supported for your system configuration, the operation fails.

Automatic. This default is not really a traversal method at all. Rather, it simply instructs ECX to choose an actual traversal method as described below, either "SnapDiff" or "Generic". For NetApp 7-Mode and Infinite Volume, ECX selects SnapDiff. For Cluster mode, ECX selects SnapDiff with Clustered Data ONTAP version 8.2p3. Otherwise, ECX selects Generic for storage controllers in Cluster mode. See System Requirements for supported clustered Data ONTAP versions. We recommend that you leave the traversal method setting at "automatic".

SnapDiff. ECX uses the NetApp Snapshot Differencing API to collect and catalog differences between volume snapshots. If SnapDiff is selected, the default number of file system differences requested in each query is 256 and the default maximum number of volumes that are simultaneously cataloged is 8.

Generic. ECX performs cataloging by scanning your file system and examining file and directory metadata. This option consumes more resources during the catalog operation. If Generic is selected, the default number of files requested in each query is 4096.

Limit and Stagger NetApp Files Catalog Policies

Consider creating NetApp Files Catalog policies that constrain the number of file system objects to catalog by limiting the number of storage systems and volumes processed by a single policy. You can also stagger the scheduled run times of the policies to prevent them from running concurrently. Please note that for the SnapDiff traversal method, ECX will never use more than 8 threads *per policy*. For example, if a NetApp Files policy selects a single storage system with 9 volumes, then 1 of those volumes will not be processed until one of the initial 8 volumes finishes. The same constraint applies to a policy that selects multiple storage systems and multiple volumes. We recommend that your NetApp Files policies select only one storage system.

Search Index Creation Occurs After Cataloging Completes

The search index creation phase of a cataloging policy is a separate step that runs after the Catalog creation phase completes. As search index creation progresses, entries are added to the indices. Consequently, two identical search queries performed 10 minutes apart may produce different results.

The search index creation phase is relatively fast. On commodity hardware, a rate of 20 million objects per hour is achievable.

Note that as the search indices are being built they are updated about 24 times per hour. Consequently, you can begin using the ECX search function before the search indices are built in their entirety.

Also, the ECX reporting facility does *not* depend on the search indices. Its functionality is fully available when the Catalog creation phase completes.

Performance Expectations

Although ECX is not a database product, it depends heavily on the NoSQL database MongoDB. It is commonplace in our industry that production, line of business databases run on big, fast, dedicated computers. Moreover, databases usually run on physical rather than virtual computers. Finally, they are typically connected to high-speed disk storage.

Despite its reliance on database software, ECX runs as a VMware virtual machine. Consequently, it competes with other VMs on the ESXi host for processor and memory. For catalog, search, and report functions, ECX has no requirements concerning the type, capabilities, and configuration of the disk storage on which its database resides. These factors inform the expectations of ECX performance. (Note that for ECX appliance protection, storage specifications are more limited; see Catalogic Software Knowledge Base article 47034.)

Another important factor is the scale of the objects ECX catalogs. There are generally very few performance problems concerning the collecting and storing of high-level objects. This is chiefly because at any moment their numbers are relatively small. But, given multiple NetApp storage systems, volumes, and snapshots, the number of low-level objects can reach billions.

There are also numerous external factors that can affect performance:

1. The capacity of the ESXi host, for example, the number and speed of CPU cores; configured physical memory; nominal throughput of controllers and member disks for datastores; RAID configuration; number and speed of NICs
2. The load on the ESXi host, for example, the number of VMs and their types of applications competing for CPU, memory and disk storage; network load
3. The capacity of the NetApp storage system, for example, the number and speed of CPU cores; configured physical memory; nominal throughput of controllers and member disks; RAID configuration; number and speed of NICs
4. The load on the NetApp storage system, for example, the number and type of operations competing for CPU, memory, and disk storage (NFS, CIFS, SnapVault, SnapMirror); network load

Cataloging Phases

The cataloging of high-level and low-level objects consists of the following phases:

- Catalog creation - read data from a resource domain and write it to MongoDB
- Search index creation - read data from MongoDB and write it to Lucene

During the Catalog creation phase, ECX reads data from a NetApp storage system or VMware host. The data obtained is written to the ECX Catalog volume.

During the search index creation phase ECX reads data from its Catalogs and writes it to the search indices on the ECX Configuration volume.

Throughput and Elapsed Time

The following discussion, which focuses on throughput and elapsed time, concerns the cataloging of low-level objects by the NetApp Files Catalog policy. As just noted, the cataloging of high-level objects is largely immune to significant performance problems.

File cataloging performance depends on the *number of files* ECX is asked to catalog. Performance is not affected by the number or size of the NetApp volumes in terms of terabytes or petabytes. Nor do the sizes of the files cataloged have any effect on performance.

The behavior of the NetApp Snapshot Differencing API (SnapDiff) is a very important aspect of ECX performance. Specifically, prior to Data ONTAP 8.2, SnapDiff performance can degrade *severely* over time. This problematic behavior is most acute if the source volume contains many millions of files spread over a "deep" (many levels) directory structure. For example, SnapDiff running on a FAS 2240 with Data ONTAP 8.1 will at the outset deliver ~1.6 million diffs per hour. But within 5 hours that rate will fall below 800,000 diffs per hour; and 24 hours into the job it will have fallen to ~400,000 diffs per hour. The volume under test contained 100 million files with many directories 5-10 levels deep.

Please note that starting with Data ONTAP 8.2 – *and regardless of FAS physical architecture (Clustered Data ONTAP or 7-mode)* – SnapDiff's performance was greatly improved. This version of SnapDiff maintains a very steady throughput which varies with FAS model, and is largely immune to volume characteristics. For example, SnapDiff running Data ONTAP 8.2 on either a 7-mode or Clustered Data ONTAP FAS 2240 maintains a nearly *constant* throughput of ~1.9 million diffs per hour.

Every run of a NetApp Files Catalog policy indexes all of the metadata on its volumes. A significant consequence of this behavior is that the ECX process virtual size may grow quite large. For example, a policy that indexes 100 million objects generates a process virtual size of at least 100 GB. Because MongoDB memory maps its database, a key performance consideration is the amount of RAM available to ECX.

As delivered, the ECX appliance is configured with 32 GB of allocated RAM. Although this is less than one-third of the 100 GB of RAM that appears to be required by the 100 million object case, it is usually sufficient in practice. This is because the process working set and not its total virtual size governs its performance characteristics. The MongoDB working set consists of the repeatedly referenced data and index pages. Because the working set size is generally smaller than the VM's 32 GB, performance is generally good.

However, if you are running a policy or concurrent policies that catalog hundreds of millions of objects, the working set size may approach and exceed the appliance's 32 GB limit. As this limit is approached, ECX performance continues to degrade and the elapsed time required to complete cataloging increases significantly in a non-linear manner.

But these ECX VM RAM considerations begin to matter only if the SnapDiff input phase can deliver data quickly enough to stress the Linux paging subsystem that supports MongoDB's memory mapping. Versions of SnapDiff prior to Data ONTAP 8.2 generally *cannot* deliver data quickly enough to put any stress on the appliance's RAM.

Another approach to reduce the elapsed time required by cataloging is, where possible, to limit the number of NetApp file policies processed and stagger their scheduling. This can be accomplished during policy definition by distributing volumes across policies such that no one policy or multiple concurrently run policies index too many files relative to either the ingest rate (SnapDiff throughput) or the appliance's allocated memory.

The criteria by which volumes are assigned to policies could also take into account the qualitative aspects of the data on the volumes. For example, some volumes may have greater business value than others. These volumes could be assigned to the same policy, again subject to the quantitative limit on the number of files. By considering the quantitative and qualitative aspects of the files to catalog, policies run faster. Such policies can, consequently, be run more frequently yielding timely data for ECX's searching and reporting functions. The same considerations can also produce policies that catalog more data of less business value. Such policies, which take longer to run, can be scheduled to run less frequently.

This approach requires understanding of ECX performance in your installation under usual runtime conditions. Test ECX performance in your installation. Create a simple test policy that catalogs at least 10 million objects. Monitor the policy's execution in ECX. Run the policy under the load conditions that are common in a production environment.

NetApp Storage System Load

Users might also be concerned with the load that ECX places on a NetApp storage system. ECX's preferred means of obtaining a volume's file system metadata is the NetApp Snapshot Differencing API, or SnapDiff. SnapDiff is a low priority process on a NetApp storage system. Processes that support file sharing, such as NFS and CIFS, as well as replication functions like SnapVault and SnapMirror all run at a higher priority.

If load on a storage system becomes a problem, there are several ways of minimizing the load. You can define policies that limit the number of volumes selected from a single storage system and stagger their execution. In this context, note that ECX has its own throttling rules for NetApp Files policies. Please refer to the previous discussion of traversal methods.

Finally, consider running the policy against a secondary NetApp storage system; for example a SnapMirror replica of the primary source volume. Before implementing this, consider the latency in accessing the secondary system. Also, the target volume on the secondary system may have a snapshot schedule different from the primary source volume. If the Catalog All Available Snapshot option is enabled in a policy definition then, because of differences between the two sites in snapshot schedules, retention, and data change rates, the set of files cataloged across these snapshots might not match what the same policy produces were it run against the primary system.

Search Index Creation

The preceding discussion applies only to the Catalog creation phase of a cataloging policy. After this phase completes, the second phase, search index creation, begins. This phase reads the newly created Catalogs and writes the Lucene search indices. There are several important aspects of this process:

- When the management console's monitoring panel indicates that the policy completes, it only indicates that the first phase, Catalog creation, is complete.
- In a separate step, the search index creation phase begins. This step is not visible in ECX.
- The management console's search facility is not fully available until the search index creation phase completes.

External Performance Factors

The following discussion concerns external factors that can improve the performance of ECX NetApp File Catalog policies:

The ECX appliance defines two LVM volumes, Configuration and Catalog, that are composed respectively of /dev/sdb and /dev/sdc. These are mapped to VMDK2 and VMDK3. By default, these two VMDKs reside on the same VMware datastore. Consequently, while there is apparent I/O concurrency from the Linux perspective, there is no real concurrency. Although I/O requests to the two devices do not queue within Linux, they queue within VMware at the datastore.

Given the current ECX architecture, this lack of real concurrency at the physical layer does not significantly impair performance. Consequently, it is not recommended to define the ECX configuration and Catalog volumes on separate datastores.

More important considerations concern the type of datastore and its underlying RAID configuration. Write optimized storage is recommended.

Because ECX is presently a single node application, its use of MongoDB requires the use of journaling. More accurately, because ECX does not presently use MongoDB replication, best practice requires the use of its journaling facility.