

# CPN Application and Code Review

---

Review Committee: A.Norman, M.Mengel, R.Illingworth, A.Romero

## Overview

The central disk services provided by the Bluearc systems, serve out in increasing volume of high performance disk storage to the current Intensity Frontier (IF) experiments including g-2, Minerva, Minos, Mu2e and NOvA. The Bluearc systems have the capability to aggregate large amounts of physical storage to present large logical volumes as NFS exported volumes. This system has been used successfully to make experiment's analysis code distributions and user specific working areas visible to both interactive login nodes and Fermigrid worker nodes. For some IF experiments, data files are also made available through the Bluearc system which permits distribution of the data without using a formal data handling service (such as SAM).

Under the current computing models used by the experiments this can result in 100's to 1000's of simultaneous requires for data from the central disk server to be transferred to the Fermigrid worker nodes. This load must be controlled and throttled to maintain optimal performance of the disk servers and balance out the bandwidth available to the experiments.

A utility called CPN was developed to perform the task of creating a file access-locking scheme that could be used to limit the number of concurrent I/O tasks that were performed on the Bluearc system. The CPN utility performs two basic functions:

1. It wrappers an I/O related command in a framework for establishing a lock indicator on the bluearc system.
2. It checks the current state of the system through the lock indicator framework to ensure that a predetermined number of I/O processes are not exceeded.

The CPN utility, its infrastructure, algorithm choices and implementation decisions have been reviewed to determine potential operational risks associated with the utility as well as opportunities to enhance the CPN utility to provide more robust guards against affecting the performance of the bluearc system.

## Bluearc Performance and Limitations

Any system for throttling the concurrent I/O operations that are performed on the Bluearc system must be capable of being tuned to match the operational limitations that the system provides.

The Bluearc titan 3200 has a theoretical performance of 12 Gb/s data transfer and 200k input/outputs per second (IOPS) per cluster node. This performance is achieved only when using high performance disks on the back end of the system. The observed performance using the low cost sata disks that are installed on the current Bluearc backend, as measured by the real IF workloads have achieved 4.5 Gb/s and 45k IOPS per cluster node.

Degradations in Bluearc performance have been observed at transfer rates below 4.5 Gb/s and 45k IOPS. These degradations have been linked to access to relatively small disk pools which can become overloaded at much lower levels. When this small disk pool is overloaded for a sustained amount of time, it has the potential to affect larger operation of the system. In this failure mode the NAS has the potential to become “clogged” and backup I/O traffic beyond the storage elements being accessed. This behavior makes the determination of overload metrics difficult to determine since they are specifically tied to individual disk pools and to the interaction of that disk pool under overload to the activity on other small disk pools. As a result, dynamic determination of the health of the system and a feedback mechanism that can act on this information is the only reliable method of preventing system overloads. (I.e. there is no simple “run less than N concurrent accesses to the Bluearc type of requirement)

The Bluearc has performance counters, which can serve as indicators and predictors of performance trouble. These counters can be published to monitoring software and locking scripts to serve as dynamic feedback regarding the current state of the Bluearc system.

## Findings

In reviewing the current CPN/LOCK utilities, the committee identified the following points as significant finding regarding the structure or operations of the system.

1. The CPN utilities are currently being distributed through a MINOS specific location on the Bluearc disk. There are multiple versions of the utility, each with different feature sets and levels of lock protection.

The current implementation of CPN is distributed through the /grid/fermiapp/minos/scripts/ area of bluearc. This deployment requires that non-minos users know of the location (which is documented on the Minos experiment specific webpages) and then have an appropriate setup that matches their experiment group setting. The way that the experiment group is determined is

based on the primary gid that is registered to a user. This is problematic since for many users this does not match an established lock area on the bluearc and can cause CPN to silently fail or create locks in the wrong area of the disk.

The CPN utilities also need to be specifically setup through a set of use defined aliases and environment variables, which can be prone to misconfiguration due to the base documentation (which is Minos specific).

More importantly, there are not one but many different copies of CPN floating around on the Minos areas of Bluearc. These differ in the functionality that implement and can cause conflicts in the locks they establish.

2. The method used for determining the file size of the files that are being operated on will be incorrectly determined when the source is a directory instead of a regular file.

The CPN system uses an ls command to extract the size of the source input that is passed to the utility. If the name of the source is not a regular file, but a directory, then the size that CPN determines is wrong. This causes a failure mode where the utility will decide that the input falls under the minimum threshold for locking (but the input is actually much larger) and hence not register a lock but then perform the intensive I/O operation. An example use case where a user will trigger this behavior is in the recursive copying of a directory from one location to another. This would result in CPN choosing to not use the lock system but perform the copy. This usage pattern has been found to be in use in multiple Nova jobs.

3. The CPN utility establishes separate lock areas and configuration sets for each experiment that uses it. Each installation of CPN is unaware of the others.

The current design of CPN uses a separate lock areas and sets of configuration files for each experiment. These areas are currently hosted under the /grid/data/ area. The areas are all independent and there is no global locking system that attempts to aggregate the locks across experiments or implement global concurrency limits. Monitoring and logging is kept local to each of these areas.

4. The CPN utilities and lock directories used by CPN are configurable by the user via environment variables.

It is also possible for a user to not configure their work to use CPN. The default behavior is that the user does not have CPN configure and hence all major I/O operations are by default unprotected.

It is possible for a user to configure CPN to bypass the locking mechanisms and to set different limits on concurrency. It is also possible for a user to misconfigure CPN by overriding environment variables. This has the potential to cause failure of the

locking mechanism and to create situations where the concurrency limits are exceeded.

5. The LOCK utility uses the Unix “find” command to count files

The CPN utilities attempt to maintain and count locks in order to limit the number of concurrent file operations that are performed on the bluearc system. To count the number of active locks, CPN issues a “find” command on the lock directory and then parses the results to determine the number of active locks. The use of the “find” command when used over a network file system can become an expensive operation and can incur significant. The command is also susceptible to race conditions (multiple processes trying to count the active locks), which can be more prevalent during periods of high network latency or disruption. This can cause the find operation to return an incorrect or stale result and can result in incorrect issuing of locks, especially during periods of burst activity. This can result in the catastrophic failure of the lock system as in the following scenario:

A large number of processes (potentially 100’s or 1000’s) start up and all perform an initial “find” based check to determine the number of active locks. Due to network latency or disruption, all of the processes determine it is safe to take out a lock prior to the first lock in the group being issued. All of the processes then take out locks (there is no additional checking) and perform their I/O. The concurrency limit is greatly exceeded with the potential of catastrophically affecting the performance of the Bluearc and causing a cascading failure mode.

6. The CPN system is susceptible to overbooking during periods of high NFS latency or network disruption

Due to the single pass, non-deterministic nature of the locking system that CPN uses, it is possible for the lock limit to be exceeded through arbitrary delays in the time between when the lock scripts check for the availability of locks and when the locks are formally established. Because of the current order of these operations and the inherent delays in the NFS file system, it is possible for burst activity to cause a group of processes to all obtain a lock and overload the system. (See the scenario outlined in the previous section) This situation is more apt to occur during periods of high latency writing to the Bluearc and has the effect of further increasing the latency of the Bluearc (cascading failure mode).

7. The CPN system is susceptible to overloading the Bluearc due to the lack of global concurrency checks.

The CPN system does not currently attempt to limit concurrent operations on a global level (i.e. across experiments). As a result it is possible for the total number of concurrent I/O operations to exceed the maximum operational limits for the Bluearc system, when multiple experiments each reach their maximum limits (e.g. Experiment-A has a limit of 40 concurrent operations, Experiment-B has a limit of

40, Experiment-C has a limit of 20. If all three experiments are utilizing their quotas, the sum can reach 100 concurrent operations, which would exceed the bluearc operational parameters)

## Recommendations

The following recommendations have been suggested to reduce risk of overloading the Bluearc and to allow CPN to reduce other risks associated with lock mechanisms across network file systems. These recommendations are also targeted at reducing the overhead associated with the CPN locking and improving overall performance of the copy procedures. The recommendations are presented in a prioritized list corresponding to a weighting of their importance and ability to impact the stability of the Bluearc system. Notations have been made regarding the estimated effort to implement a solution

1. There should be a single definitive copy of the CPN scripts/utilities, which is available through a common (non-experiment specific) area of the blue arc file system.

There should be only one copy of the CPN and lock scripts. Currently there are multiple copies that are potentially in use and implement different features of the CPN lock mechanism. This has the potential of causing inconsistencies and preventing new protection mechanisms from being used by all users. Instead, the CPN scripts and utilities should be deployed either as a versioned UPS product or as a static installation in the applications area of the IF experiments on the blue arc.

Effort Required: Standard repacking as UPS product

Time Required: Minimal (1-2 hrs)

2. The CPN infrastructure should correctly take out locks when working with groups of files and recursive file operations. The CPN infrastructure should also work properly with other data handling and transfer methods, including but not limited to rsync and gridftp.

The CPN scripts and locking code are currently design to be used to wrapper basic file I/O commands including cp, mv and rm because each of these commands has the potential to affect performance of the Bluearc if to many simultaneous operations occur. The CPN scripts no not correctly handle specific invocations of these commands that deal with recursive operations. This behavior should be changed to default to locking the entire operation regardless of what the operation is (i.e. lock out an entire “cp -R src dest” command with a single lock.) Since there are other replication and data transfer commands that have the same potential to affect the

Bluearc and these commands should be wrapped to provide the same concurrency guards and operate on the same lock pool as the baseline CPN utility. In particular the rsync utility, which is commonly used for copying large directory structures and the gridftp utility which is used for transferring files to/from grid worker nodes should be include in this protection scheme.

Effort Required: Disable existing code that bypasses locking

Time Required: Minimal (1-2 hrs)

3. There should be a single global limit on concurrent file operations that applies across all experiments that are attempting to access the blue arc

Currently each experiment has a limit on the number of concurrent file operations that that experiment is allowed to perform. When summed over all active experiments these individual limits have the potential to exceed the actual performance capabilities of the blue arc headnodes or backend disk arrays. To avoid this, there should be a single global concurrency limit that is shared across all experiments/users who utilized the central disk services.

Effort Required: Modification of existing infrastructure

Time Required: Minor (1-2 days)

4. The locking mechanism used by the CPN/Lock utilities should be changed to use a standard pattern that is know to work on NFS filesystems which avoids race conditions and is resilient to network latencies, system slow downs and network disruptions.

The current locking mechanism is highly susceptible to a cascading failure condition under situations where the write to the network disk is delayed or disrupted. Under this situation it is possible for the locking system to fail to limit the number of concurrent operations that the disk is subjected to. We would suggest using an industry standard pattern that utilizes a "token" passing scheme to eliminate race conditions and can be easily implemented on an NFS file system. We also suggest the use of a modified lock checking system that does not rely on the "find" utility and that is immune to race conditions. An example design of a system based on these suggestions is outline in the appendix to this document.

Effort Required: Modification of existing infrastructure

Time Required: Minor (2-3 days)

5. Technical Design Required: Yes There should be a method of balancing the amount of bandwidth/locks that experiments receive from the global lock pool, so that usage patterns of experiments do not adversely affect the global performance of the system

With a single global limit on concurrent bluearc file operations there is a potential that a highly asymmetric access patterns between experiments could affect the “share” of the bluearc bandwidth that jobs receive. This would occur in many schemes when one experiment had a large number of jobs running that were trying to obtain file locks, while another experiment only had a few jobs. If each job had an equal probability of obtaining a lock slot, then the experiment with more jobs would effectively receive a higher share of the bandwidth. To avoid this type of situation, a method of weighting the ability of a job to obtain a lock, or a “fair-share” like algorithm should be used.

Technical Design Required: Yes

6. The CPN lock mechanism should be expanded to support the use of a “heartbeat” mechanism to detect locks associated with defunct processes and to prevent locks associated with large/long copy operations from being cleared by the timeout settings.

The current CPN system creates a lock file whose creation time is check against a timeout value to determine if the lock is still active. This system is prone to failure when a valid I/O operation takes more time to complete than the timeout value. In this case the lock system fails and can cascade into a catastrophic failure event of the storage system. To prevent this type of failure, we recommend updating the modification time of the lock file on a regular basis while the I/O operation is in progress. Under this model, the lock file would provide a positive indicator that the I/O operation was still in progress and the timeout code that clears stales locks would not get invoked on a lock corresponding to an operation in progress. A standard pattern exists for this operation and can be easily integrated into the existing CPN utility.

Technical Design Required: Yes

7. Monitoring of the CPN lock system should be redesigned to publish both global and experiment specific metrics. These metrics should be incorporated into the established tools for monitoring of the IF Cluster and resources.

The current version of CPN is capable of recording limited statistics on the locks that it establishes. This monitoring information should be retained and additional information on the global state of the system and each experiment’s share of the lock bandwidth should be added to the accumulated metrics. Where possible this information should be published to the IF monitoring pages and/or to the monitoring pages of groups responsible for the support of this system.

Technical Design Required: Yes

## Appendix A: Example Locking Pattern

The following example demonstrates a potential redesign of the CPN system using a standard locking pattern that is known to work properly over NFS filesystems. This example assumes the need for global locking as well as the need for experiment based balancing of lock allocation.

1. The locking mechanism should be changed to use a standard pattern that is known to work on NFS filesystems
  - a. According to research, to obtain an exclusive lock on a file the script should make a unique file and then hard link the file to a common token file in the lock directory
  - b. The client then must check the link count on the unique file
  - c. If it has obtained the token then the ref count on the file will have incremented
  - d. This allows the client to then move their file into the lock directory
2. The overhead associated with the find command should be reduced
3. Should work properly with other transfer methods (sync and gridftp)

Our current, whiteboard-draft system assumes the following setup:

1. A global lock directory named “locks/active” is established on the bluearc system
2. A queue directory for each experiment is created on the same bluearc system with a name “locks/queue/*experiment*” (i.e. locks/queue/nova or locks/queue/minos, etc...)
3. A completed locks directory is established on the same bluearc file system with a name “locks/done”

When the CPN script wishes to obtain an I/O lock it uses the following pattern:

1. Create/touch a unique directory and file locks/queue/*experiment*/*host-pid*/f
2. Try to hard link the above file f to locks/active/exclusive
3. Check the link count on locks/queue/*experiment*/*host-pid*/f, if it is 2, we got the exclusive lock, and can continue; if not, we sleep a little, and go back to step 2.
4. Check the link count on locks/queue/active. The link count is the number of subdirectories plus 2. So we can check our global limit. If the number of subdirectories is at our limit, unlink locks/active/exclusive, sleep an amount determined by our fair-is-share algorithm below, and go to step 1.

5. Rename our `locks/queue/experiment/host-pid` directory to `locks/active/host-pid` to claim our slot, unlink `locks/active/exclusive`, and declare victory.

CPN should periodically touch the file “f” in the directory above to update its timestamp while the actual copy is going on. This will be a heartbeat for stale lock detection.

Returning a lock when done consists of moving the `locks/active/host-pid` directory to the `locks/done` area, after obtaining the `locks/active/exclusive` link to it; then removing `locks/active/exclusive` when done.

A simple old-file cleaner on `locks/active` and `locks/experiment` cleans out bookkeeping for cpn processes that died for any reason. (This is why step 4, above goes back to step 1 on failure, if it has been sleeping a long time, its queue entry might have disappeared).

The model currently being considered to give locks out fairly is to affect the amount of time a given cpn process sleeps when there are no slots available in step 4, above.

Basically, the idea is that if you have a longer queue, you try less often to get a lock.

So you multiply the length of your queue (which you can get by checking the link count of your queue directory) by some scaling factor  $f$ , and pick a random integer in the range 1 to that product; then divide by 10; and that gives your sleep time. If the scaling factor is 10; a cpn for an experiment with 2 items in the queue will check every second on average; one with 200 in the queue will check every 100 seconds on average; and the experiments will have approximately equal chances at any open slot that appears – each experiment will check about twice a second.

## Appendix: Network bandwidth

Concerns have been raised regarding the limitations on the network bandwidth that is available between different stations and the bluearc storage system. The following section covers some of these concerns and gives examples of how these bandwidth limits could be addressed. This follows directly from notes provided by K.Chadwick on 25OCT2012.

The bandwidth to perform any file transfer will be limited by one or more of the following factors:

1. Available bandwidth between the client system and the immediate upstream network device. In the case of the currently deployed Grid cluster worker

nodes, this is typically a 1 Gbit/second network connection.

With the advent of many core Grid work nodes, it is important to recognize that a single file transfer may have to compete for the available upstream bandwidth. At the present time, this would be bounded by:

Network interface - 1 Gbit/second

The number of file transfers currently "in flight" on the specific Grid worker node. This may be difficult to determine directly, so as an alternative, a scaled value based on the number of jobs or cpus may be used as an approximation.

The number of jobs (this can be determined via batch system queries or can also be inferred from the number of processors on a particular system):

```
let Ncpu=`/bin/cat /proc/cpuinfo | grep -c 'processor'`
```

So, the instantaneous "fair share" available bandwidth is:

$$\frac{1 \text{ Gbit/second}}{\text{Number of simultaneous transfers}}$$

The an instantaneous lower bound of "fair share" available bandwidth is:

$$\frac{1 \text{ Gbit/second}}{\text{Number of jobs}}$$

This can change fairly frequently if the Grid worker node is not in a "steady state" processing of jobs.

An approximation that is easier to calculate is:

$$\frac{1 \text{ Gbit/second}}{0.25 * Ncpu}$$

The choice of 0.25 in the above formula is somewhat arbitrary, I can see arguments that might bound this value between [0.1,1.0].

It is worthwhile keeping in mind that future Grid cluster worker nodes may be deployed with "bonded" multiple 1 Gbit/second network connections or may even have "native" 10 Gbit/second network connections once the cost

of network interfaces decreases to "commodity" price levels. So there will eventually be a requirement to take the varying network connection speeds into account in the above calculations.

2. Available bandwidth across the backplane of the immediate upstream network device.

This bandwidth is usually one or two orders of magnitude above the individual Grid cluster worker network connections, and for the majority of the time it is extremely unlikely that this will be a bandwidth limitation.

3. Available bandwidth between the immediate upstream network device and the Fermilab network "core".

At the present time, the typical upstream network device is a 6500 series switch that has one or more 10 Gbit/second interconnections to the Fermilab network core. This may be sufficient to occasionally present a bandwidth limitation, but probably not in the majority of the time.

4. Available bandwidth between the Fermilab network core and any network devices that "front end" the storage service.

At the present time, the typical bandwidth across the Fermilab network core is 80 Gbit/second, and the storage service "front end" network devices typically have multiple 10 Gbit/second interface connections to their connection point to the network core.

It is worthwhile to note that network services is planning on enhancing the available "cross core" bandwidth, so this should be parameterized.

5. Available bandwidth between the network devices that "front end" the storage service and the network interfaces of the storage service.

At the present time, the BlueArc heads that service the scientific community have quantity  $4 \times 10$  Gbit/second network interfaces that are "bonded" together. Any single transfer may be limited to a single 10 Gbit/second bandwidth, but multiple simultaneous transfers are possible.

6. Available bandwidth between the BlueArc heads and the back end storage.

I believe that the CPN application should be enhanced to automatically determine the likely available bandwidth when determining the CPN timeout parameters.

If the CPN application keeps track of the number of global simultaneous file

transfers, then estimates of the globally available network bandwidth across the network infrastructure can also be performed.

In any event, the CPN application should calculate the available bandwidth via the several models discussed above, and choose appropriate timeout or file transfer progress inquiry parameters based on the values that are determined and the estimated file transfer size rather than fixed values read from a parameter file.