Paper 3351-2019

# Engineering CAS Performance Hardware Network, and Storage Considerations for CAS Servers

Tony Brown, SAS Institute Inc.

## ABSTRACT

SAS Viya, using SAS Cloud Analytical Services (CAS), offers a highly performant and scalable computing environment. For the best performance, the CAS Server can benefit from performant hardware architecture solutions to support implementations with high software scalability and large data models. This paper offers best practices for provisioning hardware and network options to create a performant SAS Viya environment: from CPU and Memory to CAS_Disk_Cache configuration and inter-server network bandwidth requirements.

## INTRODUCTION

SAS Viya provides scale out data manipulation steps and statistical procedures that can parallelize large data workloads efficiently. This allows very long-running or high-data-volume jobs to be completed very quickly via task parallelization. We have seen drastic reductions in job run time, benefitting data management, and modeling workloads. Scaling tasks through parallelization is enabled by several things to be successful:

- Optimizing any existing inefficient code structures in terms of IO, memory usage, and inefficient task activity
- A scale-out hardware infrastructure that can support the concurrent load and instantaneous resource demand of parallelizing a large task set
- Software that can manage the parallel workload
- Proper hardware sizing, planning, and configuration to support the application workload performance goals

This addresses the above topics for running SAS Viya on SAS Cloud Analytic Services to successfully achieve the expected performance goals of task parallelization.

It begins with practical goals to make the process tasks and underlying system efficient. This is followed by discussions on CAS resource requirements, CAS backing store performance prerequisites, and finally container and virtualized processing.

## PRACTICAL GOALS

### OPTIMIZING APPLICATION WORKLOAD TASKS – CODE AND PROCESS

Everything starts with good code. Software performance engineers are aware that optimizing an inefficiently coded process yields the largest return for the effort in job performance chains. This can figuratively supersede all other aspects of the performance chain, IO, Memory, and CPU. There is an incredible amount of customer SAS 9 legacy code that "works," but could benefit from recoding for much more efficient single or low-thread-count operations. This code is rarely changed, often resulting in additional hardware being thrown at it to try to overcome the inefficient process instead. This approach is frequently

perceived as easier and less-risky, especially when the author of the code is no longer available. The code or process might be horribly inefficient, but it still works somehow, so it won't be touched.

When the execution time of a business-critical, long-running process needs to be shortened, parallel scale-out job architecture is the obvious answer. CAS offers the underlying technology to assist accomplishing this with SAS Viya. You might want to re-engineer some of your code to optimize this in SAS Viya. The world is standing on the cusp of another leap in analytics, such as machine learning and artificial intelligence (AI) to enhance current analysis and modeling techniques. This requires a highly performant processing scale to work. To leverage parallel scale out technology, coding and process structures need to be modified to efficiently take advantage of them. Significant effort is put into fabricating the physical scale-out infrastructure, and it is best optimized by running efficient process and code.

A good example of bringing efficiency to processing data in SAS Viya has to do with sorting the data that will be lifted into memory for analytical processing. If you are a SAS 9 user, you will think the best way to do this is to use the SORT procedure. In SAS Viya, we use DATA step code and BY GROUP processing to replace memory intensive PROC SORTS, because of two factors:

- Memory Speed - SAS Viya processes are in-memory processes, and in-memory processes need to stay within memory to perform well. The SORT procedure is a memory hog in SAS 9, so we replaced it with DATA step coding in SAS Viya. DATA step coding can more efficiently perform the same sorting tasks with minor work-arounds while also requiring less memory.
- Parallel Workload Management - DATA step processing can take advantage of very highly parallelized resources in CAS (the parallel DATA step). The data can be divided, sent to a large number of scaled-out servers, and processed on many CPUs and in different host memory pools simultaneously. This drastically reduces the entire job workload time to a mere fraction of the original task execution time.

Performing a code review is both efficient and recommended since ***many SAS 9 procedures are Not Available to run in the in-memory environment of CAS in SAS Viya yet***. However, those procedures will still run in the SAS Workspace node. Please review the SAS Viya documentation and consult with our SAS Account Team to have your code evaluated by SAS (we have a program that does this). This can help spot where code conversion is needed for SAS Viya in-memory processing, as well as indicate areas where coding efficiencies could optimize high-parallel processing and the speed-up of jobs.

## SYSTEM RESOURCE OPTIMIZATION CHAIN – DATA ACQUISITION THROUGH DELIVERY

SAS 9 and CAS have some system performance pressure points in common, and some that are very different. They both use high IO, CPU, and Memory. Because CAS performs in-memory processing, versus on-device processing, and is highly parallel, capacity and bandwidth of these resources demands close attention.

There are numerous white papers for performance, tuning, and workload management for SAS 9: http://support.sas.com/kb/42/197.html

These guides were produced to help SAS Administrators understand the end-to-end performance requirements of SAS. The same concepts of allotting enough IO bandwidth, memory for in-memory processing, and adequate CPU resources will need to be even more carefully considered for SAS Viya processing on CAS. Because the highest performance (and job run-time reduction) in Viya/CAS requires the job to stay within the bounds of the

in-memory processing, the sizing and hardware capacity and performance requirements are crucial. The minimal guidelines and heuristics are included in the ***Viya® Resource Requirements and Sizing*** section below. The provisioning should include the entire job chain, from data acquisition to CAS provisioning and management, to final data delivery.

## WORKLOAD PLANNING

CAS and SAS 9 are very different in terms of their operations and performance considerations. There are common workload elements to consider with both for sizing and provisioning:

- Number of concurrent jobs
- Data IO Memory and CPU Required for the mix of Job Types
- Desired Execution Time or Time to Solution
- Types of Data Acquisition – Serial, Parallel
- SMP versus MPP Architectures
- Workload Resource Management – Restriction or priority of resources by users or groups

The above elements are calculated and used to help size and configure the following infrastructure and system management resource requirements:

- Number of Cores, Speed of CPUs, Hyper-threading use
- Size and speed of the Memory
- SMP/MPP - Bandwidth and Speed of the Network (between nodes and between storage and nodes)
- Bandwidth and Speed of any direct-attached storage
- Storage capacity, bandwidth, and throughput for:
    - Source Data Acquisition
    - CAS_Controller_Temp space
    - CAS_Disk_Cache
- Host System or Application implemented workload restrictions:
    - Host workload managers, cgroups, ulimits, and so on
    - *CAS Priority Policies (who gets how much, of what)

CAS is an in-memory processing system: application processing, including scratch work, is performed in-memory at high memory speeds, replacing memory maps for the slower hard-paging of data from and to disk via a traditional file system. Memory capacity and performance is crucial. The highly parallelized CAS job structures provide a reduced time-to-solution, due to parallelism, and in-memory processing speeds. Because so many job tasks are launched in parallel, significant resources are needed to cover the instantaneous load demand.

The SAS Enterprise Excellence Center (EEC) offers free sizing guidance for new Viya/CAS systems. They use detailed questionnaires that collect required metrics to calculate hardware system resource capacities and performance parameters. Please work with your SAS Account team to have a sizing done for your planned SAS Viya usage.

These sizing exercises use the questionnaire data to assess workloads and provide two crucial elements: the number and speed of CPUs needed, and the amount of Memory

required for a sized workload. The third element of the performance triad, IO bandwidth, is provided as a heuristic of Megabytes per second, per core (MB/sec/core). This metric includes all movement of IO, from first data acquisition, to host processing, CAS_Disk_Cache, to persistent storage or delivery of results.

The above results apply to both SMP (symmetric or single machine processing) as well as MPP systems (massive parallel processing – using multiple host nodes).

## SAS VIYA RESOURCE REQUIREMENTS AND CAPACITY

### SAS VIYA NODE ARCHITECTURES

Your recommended EEC sizing will suggest a capacity configuration for your SMP or MPP SAS Viya system. Please follow that configuration, including server separation of node identities and tasks. For example, in an MPP configuration you might be assigned a CAS Master Node, CAS Worker Nodes, a Services Node, and so on. It is typically wise to provide a separate Services Node for the Microservices. Microservices instantiate 60 GB of RAM usage when they are started and consume even more later. They also use CPU resources to accomplish their work. These resources are best provisioned on a stand-alone node. If you try to combine the CAS Microservices with another node, you would have to provision adequate CPU and Memory resources plus any additional IO capacity and bandwidth requirements. It is wise to separate it, with the suggested capacity of your EEC Sizing.

You can learn more about CAS node architectures in the SAS Viya 3.4 Administration: Orientation link in the Additional Reading section below.

### CPU

CPU recommendations for CAS Servers are discussed below. Use the fastest and newest generation CPU you can afford for your SAS Viya applications. SAS Viya VA, VS, and VDMML applications can be very CPU-intensive. CPU cycles are required to drive the very fast in-memory processing of data, as well as handle any backing-store data IO to the CAS_Disk_Cache. It is recommended to use at least the number of CPUs suggested by your EEC Sizing.

The following recommendations should be carefully considered:

- Use x86_64 processors only (64-bit ARM processors are supported for ESP on the Edge only)
- AMD processors are technically supported but Intel Xeon are strongly recommended
- The Intel E3-E7 series and Intel Xeon Scalable processors have been tested for CAS nodes
- The minimum CPU core count for CAS licensing is 4 cores
- Enabling hyper-threading is highly recommended
- Since processing is at in-memory speeds, the fastest CPU you can afford is a better choice
- There are many flavors of Xeon processors
    - Examples of different 10 core processors
        - Intel Xeon E5-2689 v4 @ 3.10GHz - CPU Mark Benchmark* 19,708

- Intel Xeon E5-1681 v3 @ 2.90GHz - CPU Mark Benchmark* 18,367
- Intel Xeon E5-2680 v2 @ 2.80GHz - CPU Mark Benchmark 16,341

- SAS has tested the above CPU models for SAS Viya performance. The newest generation processors (for example, v4) are typically recommended for higher floating point and memory-handling performance, as well as an improved overall CPU Mark score using PASSMARK software

*The CPU Mark Benchmark above is score composed of many elements giving the CPU a relative numeric score. The higher the score, the better. You can learn more about the PASSMARK benchmark software and how to interpret scores here: https://www.passmark.com/support/performancetest/interpreting_test_results.htm

## MEMORY

Since SAS Viya is an in-memory processing system, the CAS controller and worker nodes require not only an ample amount of memory, but balanced memory configured with speed to uphold the highest performance. Please consider the following points when choosing and allocating memory:

- You will see recommendations in SAS Viya documentation to allocate at least 2 times the amount of RAM for the size of the total concurrent incoming files in your workload. This is a good starting place for memory capacity planning. Be aware that this amount can and will vary depending on your workload actions. If you are performing significant data manipulation (new data creation, deletion, re-ordering, and so on.), the memory required to support this activity can push your Memory needs higher, perhaps up to 3 times the size of the incoming data or more. Please perform a detailed workload analysis to best estimate your memory capacity, especially if you are performing heavy data manipulation.
- For optimal performance we typically recommend a minimum memory speed of 1600 MHz.
- When populating a server with memory, very careful attention should be paid to:
  - Maintaining the maximum clock frequency of the memory. In many systems, fully populating the DIMMs on the bus board can result in a saturation of the bus, dramatically lowering the resulting memory speed. Don't fill every slot! Instead, scale out to more nodes and divide the workload further.
  - Maximize interleaving (access) by maintaining a **balanced configuration of DIMMS across the bus and channels.**
- Manufacturers typically publish a population chart for adding DIMMs and related frequency degradation as banks are populated. Please refer to this!
- This advice applies to E5 and E7 Intel chipsets (4 channels) and the latest Intel Platinum chipsets (6 channels). See the illustration in Figure 1 below.

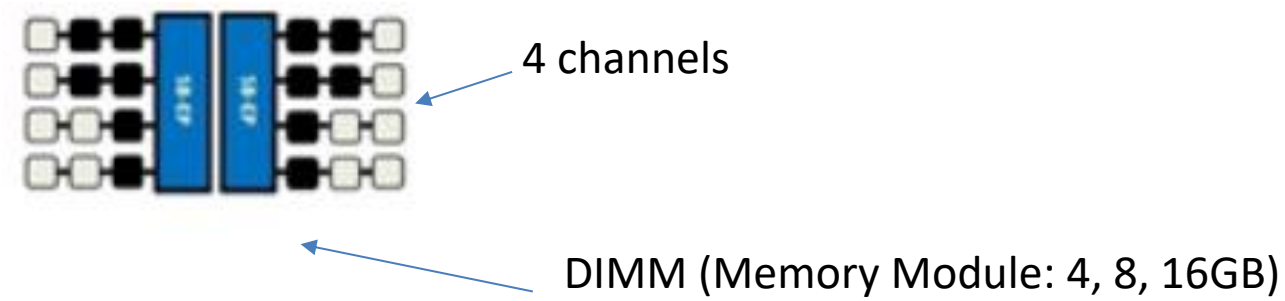


**Figure 1. Use the manufacturers population chart to place DIMMs in a balanced configuration. The example above is unbalanced.**

## NETWORK

There are several aspects of network connectivity that will need to be considered for SAS Viya. These include:

- Any network paths to get source data to the CAS system for SAS Viya processing
- Network connectivity between the CAS nodes themselves:
  - Between CAS Controller Node and CAS Worker Nodes to distribute data in either serial or parallel fashion
  - Between the MicroServices nodes and the CAS controller
- Between the CAS Controller Node and upstream nodes for Stateful Services (examples…)
- Any network paths required to handle request and results transfer to and from SAS Viya processing to Viya applications such as VA, VS, or VDMML, and so on.
- Any network paths required to persistently store data files created by the SAS Viya processing

Data loading into, and resulting from, SAS Viya processing (SMP or MPP CAS architectures) can involve very large data sources read in and large result sets written out. This part of the individual customer architecture needs to be measured and provisioned with enough bandwidth to achieve the response times that will meet your business needs and budget. This typically involves allocating enough network bandwidth to provide the equivalent of 100 Megabytes per second per host core for SMP systems. This generally guarantees you can feed the cores fast enough for consumption.

The rate for transferring data to CAS MPP nodes will require approximately the same equivalent, per node. This means each CAS MPP node will need high bandwidth for incoming and outgoing persistently stored data.

If the incoming and outgoing data sources are network attached, you might need multiple bonded network adapters in each node. We typically recommend a multiple of 10 Gbit-minimum adapters per CAS host node to achieve the bandwidth desired. For example, if you use 16 Cores in a CAS worker node, you would typically need a minimum bandwidth of 100 Megabytes per second per core or greater, or 1.6 Gigabytes per second total to/from the node. This would require the bandwidth of two bonded 10 Gbit network connections and

commensurate underlying LAN fabric and switching bandwidth. These guidelines will help you achieve the best performance available, but actual engineering choices and ultimate performance characteristics should be driven by your business need and budget.

Keep in mind the figures above pertain to both large incoming files to process, and large result files to deliver, from SAS Viya. The smaller result sets of jobs, studies, analyses, and so on. from consuming applications such as VA, VS, VDMML, and so on. will likely not require this bandwidth back to the end-user delivery layer.

In MPP configurations, coordination of CAS actions and results generate cross-node communication where speed is of equally high, or higher, importance to that of incoming or outgoing data. If a high availability (HA) mechanism used by CAS is used to back up stored CAS_Disk_Cache blocks from one CAS worker node to others (See CAS_Disk_Cache below), then IO traffic between the MPP worker nodes can represent a considerable volume (consider 1.5 to 3x the size of each worker node's RAM being stored across the interconnecting fabric on other nodes for safe keeping of backed-data, in CAS_Disk_Cache, in case of failure).

### PERSISTENT UPSTREAM OR DOWNSTREAM STORAGE

In the Network section above, we described the minimum 100 Megabytes per second, per physical core, bandwidth requirements of getting data into and out of CAS nodes. Those requirements also apply to your upstream and downstream IO, to and from permanent storage that is not network attached. If you are using fiber-channel, direct attached storage, or storage within an HCI cluster, the same minimum rate applies.

## CAS_CONTROLLER_TEMP

### WHAT IS CAS_CONTROLLER_TEMP?

For serial data loads to MPP CAS, the CAS Controller stages the incoming data on local disk. Once all the data is on board, the Controller distributes the data to the CAS Workers. Similar actions apply when saving data back out to some cloud providers, like Amazon S3. The CAS_CONTROLLER_TEMP environment variable defines the location on the Controller's local disk where that data is temporarily staged. The suggested bandwidth for this staging area is 100 – 150 Megabytes per second, per node core serviced for the CAS worker nodes.

## CAS_DISK_CACHE PERFORMANCE AND ARCHITECTURE

### WHAT IS CAS_DISK_CACHE?

SAS 9 jobs use scratch working space in the SAS WORK file system stored on a persistent device. These file systems were provisioned with a lot of space so that SAS procedures had ample room to store their work when operating within host memory became a constraint. This produced a lot of traditional file system paging to/from a storage system.

Sizing memory capacity is especially important to CAS since we want data to stay in-memory and run at memory speed as much as possible. But what happens when you exceed memory space? How do you continue? CAS uses a memory-mapped file system to provide extra space for backing stores of its in-memory data. A memory-mapped file system maps virtual memory to another source (that is, a disk storage device, shared memory object, and so on.). For the sake of our discussion, the source will be HDD, SSD, or NVMe storage devices, inside or outside (but very close to) the CAS worker nodes.

This memory-mapped space is treated and behaves like memory, even though it might be mapped to a persistent disk store. This memory map can act as a "backing store" for data that is operated on in-memory.

## WHICH FILES, WORKING SETS, REPLICATION BLOCKS GET INSTANTIATED TO CAS_DISK_CACHE?

CAS_Disk_Cache is a location that holds memory-accessible blocks of data coming into CAS, or being processed by CAS, that need a backing store or replication copy for safety:

- Incoming Source data that does not already have its own backing store that CAS can use (replication copies in case data needs to be re-retrieved). Backing rules for these sources vary based on SMP and MPP architectures, source library paths, and in some cases, whether data is co-located (on the same host)
- Files being processed in-memory that add, change, or create new data for a file being worked on. The default size of the CAS_Disk_Cache blocks is 16MB. This can be increased up to 512 MB for more IO efficient operation of very large files
- Copies of data blocks of other files from other CAS Worker Hosts that are stored on this node for high availability replication and retrieval

Figure 2 below is a table of different scenarios where data is loaded into CAS for in-memory processing, and which of these gets established to the CAS_Disk_Cache location as a backing store. The only incoming file types, source locations, and load mechanism types that do not automatically get established to the CAS_Disk_Cache are:

- SMP Nodes (Single CAS nodes) – Serial Load Mechanism Only
  - Any SASHDAT file types
- MPP Nodes (multiple parallel nodes) – Serial or Parallel Load Mechanism
  - Co-located Hadoop File System (HDFS) source SASHDAT files, or Distributed Network File System (DNFS) source SASHDAT files

| Case | CAS SMP/MPP | Serial/parallel load | CAS_DISK_ CACHE used | COPIES=1 (default) enforced? | Valid for |
|---|---|---|---|---|---|
| 1 | SMP | Serial | **YES** | **NO** (SMP) | PATH SAS7BDAT, PATH CSV, LASR, DBMS |
| 2 | SMP | Serial | **NO** | **NO** (SMP) | PATH SASHDAT |
| 3 | MPP | Serial | **YES** | **YES** – Fault tolerance | PATH SAS7BDAT*, PATH CSV, PATH SASHDAT, LASR, DBMS |
| 4 | MPP | Parallel | **NO** | **NO** – Fault tolerance (data redundancy relies on HDFS/DNFS) | CO-LOCATED HDFS SASHDAT, DNFS SASHDAT |
| 5 | MPP | Parallel | **YES** | **YES** – Fault tolerance | CO-LOCATED HDFS CSV, REMOTE HDFS CSV, DNFS CSV, LASR, DBMS EP, S3 CSV, S3 SASHDAT |
| 6 | MPP | Parallel | **YES** | **NO** (not by default**) – Fault tolerance (not enabled by default**) | REMOTE HDFS SASHDAT |

**** It is possible to load legacy SAS7BDAT files in parallel in certain conditions. If the SAS7BDAT file is accessible to all CAS workers at the same location (for instance a SAS7BDAT file on a NFS share mounted on every CAS worker), then you can use the dataTransferMode="parallel"option. The CASLIB must be a PATH CASLIB, not a DNFS CASLIB. Syntax example:***

```
proc casutil ;
  load casdata="big_prdsale.sas7bdat" incaslib="caspath" casout="big_prdsale"
outcaslib="caspath"
  importoptions=(filetype="basesas" dataTransferMode="parallel") ;
quit ;
```

**** Copies=1 and fault tolerance can be implemented by using the cas.SUBSETSESSIONCOPIES option***

**Figure 2. Types of data backed by the CAS_Disk_Cache**

Figure 3 below shows data that CAS is operating on in-memory (XYZ blocks pictured below), also gets instantiated to CAS_Disk_Cache. Depending on the type of job STEP activity used, this changed data can be quite voluminous and can increase the job footprint in CAS_Disk_Cache. This is discussed more below.

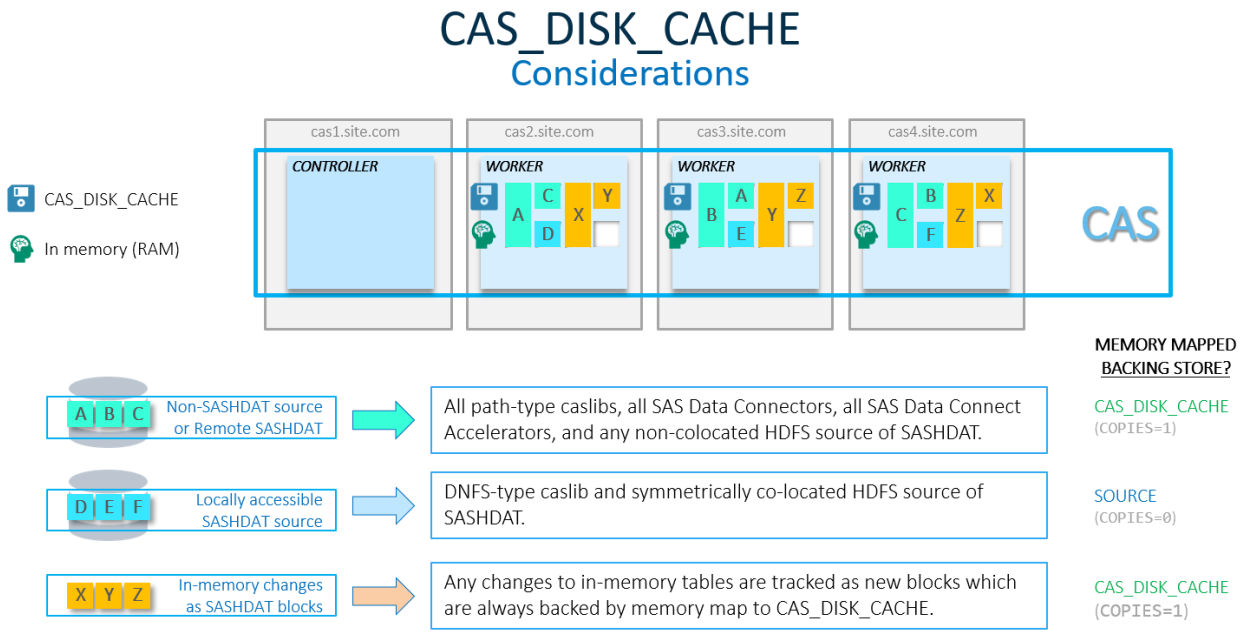


**Figure 3. Types of data backed by the CAS_Disk_Cache**

## HIGH AVAILABILITY – NON-MEMORY MAPPED BLOCKS FOR CAS_DISK_CACHE BACKUP BLOCKS

CAS keeps open files handles in CAS_DISK_CACHE for both active and backup blocks. Active blocks are mapped into memory, while backup blocks are not. In the case of a node failure, some backup blocks will be mapped on the remaining nodes to replace data that was lost.

See figure 4 below.

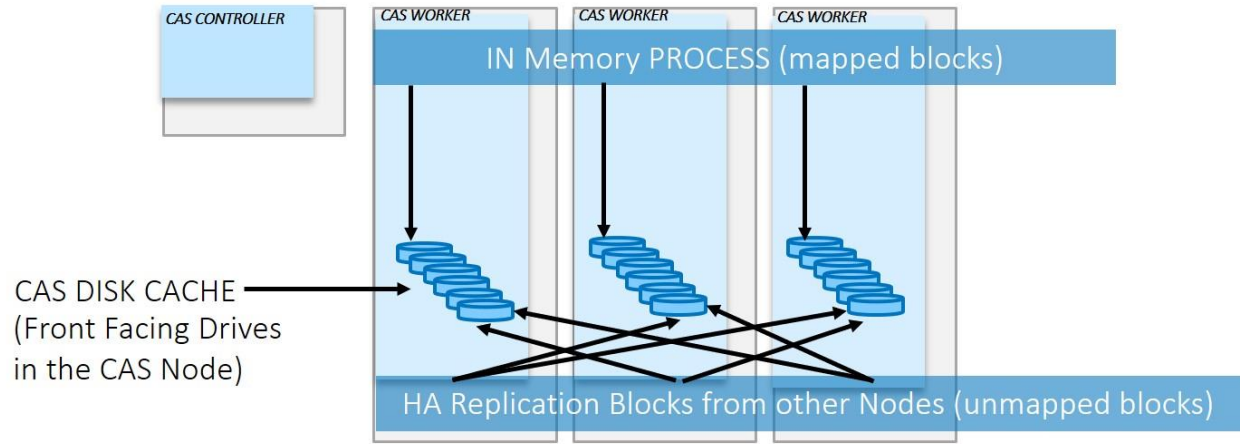


**Figure 4. High availability replication blocks from other nodes can be stored as unmapped blocks in the CAS_Disk_Cache location**

## CAS_DISK_CACHE MEMORY MAP

The default location for CAS_Disk_Cache is /tmp. The only danger with using the /tmp location is when it gets filled up. This stops everything on that node, requiring further use of /tmp space to operate.

If a site is upgrading from LASR, and wants a setup as close to LASR as possible, setting this to /dev/shm will give the best performance, closest to LASR. It requires that all tables fit into memory though.

If it is known that there will be more tables than can fit into memory,
then setting this to local drives becomes the best choice. The bandwidth and latency of the drives primarily into play if the user creates large output tables, or if paging is common.

The method of using the local drives above for the memory map for the memory map, using front-facing drives in the CAS node, is described in Architectural Considerations for CAS_Disk_Cache below.

## CAPACITY CONSIDERATIONS FOR CAS_DISK_CACHE

CAS_Disk_Cache can become a very busy place, used as a store for in-memory process data, changed blocks, and HA replication from other MPP nodes. The data expansion in this location can be considerable. DATA step processing can create new tables, alter tables, reduce tables, and increase tables. These changed data blocks associated with table manipulation are backed in the CAS_Disk_Cache. Therefore, it can grow quite large. I have seen minimum sizing recommendations for the CAS_Disk_Cache to be 1.5 times the size of RAM in a CAS node. I strongly recommend provisioning that capacity at a minimum of 3 times the size of RAM in a CAS node. Depending on your specific job actions and data sizes, your capacity requirements might vary considerably.

In addition to the capacity requirements of CAS_Disk_Cache, IO performance of this location is paramount. We discuss that in the Architectural Considerations for CAS_Disk_Cache Storage section.

## ARCHITECTURAL CONSIDERATIONS FOR CAS_DISK_CACHE STORAGE

CAS_Disk_Cache performance benefits from a high bandwidth, low latency path to storage (that is, performant and fast).

This path bandwidth and latency are key factors in overall CAS performance. When CAS needs to write to or read from the disk cache, it needs to be highly performant with the default 16 MB blocks it manages. Fortunately, the cache can be configured to take advantage of many storage devices at once, parallelizing its work across all of them.

Think of CAS in-memory operations as a boat zooming across the water "on plane." The only part of the boat touching the water is basically the propeller, with the entire hull lifted up and avoiding the friction of the water. This is as fast as a boat can go. When the hull touches the water, it slows the boat, decreasing its speed dramatically. This is similar to what happens when CAS is moving at in-memory speeds performing its processes, then suddenly has to push data blocks to/from the CAS_Disk_Cache. The boat comes "off plane." How much the slowdown is depends on the physical configuration of what underlies the CAS_Disk_Cache .

Consider a rough equivalency:

- If Memory speed = ~0 IO latency
- A drop from Memory speed to locally, bus attached disk/ssd = ~10x latency
- A drop from Memory speed to local disk/ssd via a Node Hardware Raid Controller = ~100x latency
- A drop from Memory speed to storage via a FC/Network storage connection (off-board) =~ 1000x latency
- A drop from Memory speed to storage via a Network NFS storage connection (off-board) =~*Off the Charts* latency

CAS_Disk_Cache can significantly benefit from both very high bandwidth and very low latency per IO transaction. Therefore, the list above becomes important.

Following the Hadoop model of backing store architectures, the current published advice for the CAS_Disk_Cache underlying storage architecture includes:

- CAS_Disk_Cache drives placed inside the CAS node
- On-host bus, front facing drives (HDD,SSD, and so on.)
- Drives configured in an individual Just-A-Bunch-of-Disks (JBOD) arrangement
- CAS_Disk_Cache Sizing form factors were:
    - 1 memory map per storage drive
    - 1 file system per CPU core
    - 1 directory per file system/drive
    - No drive striping involved

The premise of the above model was to parallelize many worker threads in the OS, servicing CAS_Disk_Cache in parallel across multiple drives, file systems, and directories at once, avoiding IO, metadata, and so on contention at each layer. This model has seemed to work well for testing. To an extent, it does provide decent bandwidth at relatively low latency. It also has some drawbacks when it comes to a physical drive failure: there is no recovery/continuance for the operations touching that JBOD drive.

We have found with some VA testing, that a higher number of directory structures might be needed (not by CAS itself, but due to VA actions). SAS Viya VA can very rapidly create and

destroy large numbers of working files and can overwhelm directory structures if there aren't sufficient numbers of them.

We have also found that striping multiple CAS_Disk_Cache locations using a Hardware Raid Controller yielded very acceptable bandwidth/latency performance, while allowing a RAID level protection. A single SSD can range from 3.5 to 500 times faster than a 15k rpm HDD, so SSD consideration as the storage device is very strong. In addition, mounting the file systems associated with the CAS_Disk_Cache via a striped volume can yield aggregate parallel performance, while taking advantage of the write-cache offered by a Hardware Raid Controller.

We performed a raw IO test on an HP Synergy Converged Infrastructure with the following hardware and test characteristics:

- Hardware
  - HP SYNERGY CI, 16 Cores, 256 GB RAM, On-Board SSD Drives, HDW RAID Controller – Synonymous with Standard Rack Server Provisioning
- Disk Arrangements Tested
  - Independent JBODs, High Speed, Write Intensive SSD
  - Striped JBODs, High Speed, Write Intensive SSD
  - Striped RAID5 Volume, Write Intensive SSD
- Test Characteristics
  - 14 DD Stacks run against Drive Set in Configurations Above, 16 MB Block Transfer Size (to emulate the *CAS.maxtablemem* – default block size for CAS)
  - WRITE Flood
  - READ/COPY/WRITE Flood

In figure 5 below, three RAID 5, 3+P LUNs, configured with a 2 MB LVM stripe on 9 front-facing drives delivered 5000 MB/sec IO bandwidth, while providing RAID level parity and safety should one of the drives physically fail. This is a good alternative to the original JBOD model, which did not support HA. Using write-intensive, performance drives is crucial for drive wear and extended life performance.

| TEST | RUN | 12 JBODS NO STRIPE<br>12 Front Facing Drives | 12 JBODS 2MB Stripe<br>12 Front Facing Drives | 3 – R5 3+1 LUNs – Not Striped*<br>9 Front Facing Drives | 3 – R5 3+1 LUNs – 2 MB LVM Striped*<br>9 Front Facing Drives |
|---|---|---|---|---|---|
| WRITE | RUN1 | 680 MB/sec | 4061 MB/sec | 2019 MB/sec | 3000 MB/sec |
| | RUN2 | 689 MB/sec | 4056 MB/sec | 2016 MB/sec | 3000 MB/sec |
| READ/COPY/ WRITE | RUN1 | 1344 MB/sec | 4053 MB/sec | 4303 MB/sec | 5062 MB/sec |
| | RUN2 | 1392 MB/sec | 4083 MB/sec | 4279 MB/sec | 5050 MB/sec |

***Denotes use of the hardware RAID controller write-down cache benefit***

**Figure 5. 16 MB Block testing against various drive configurations, simulating CAS_Disk_Cache block performance**

Host bus-attached NVMe drives are much faster and offer higher bandwidth than SSDs. If they are used in a JBOD arrangement (which negates recovery if a drive fails), they cannot be beat for low latency or bandwidth on a drive-to-drive comparison. The drawback of NVMe drives is that they cannot be used underneath a Hardware RAID Controller. If you striped them, you would have to use a software RAID stripe. This would not cut out the benefit of the write-down cache in the Hardware RAID Controller with regular SSDs. Our testing has also found that the software stripe of NVMe drives, even in an optimal RAID 10 configuration, would slow the performance of the NVMes back to the equivalent performance of SSDs in a Hardware RAID stripe.

Further testing has also found that the number of directories originally posited in the SAS Viya 3.3 Administration: Orientation guide was based on the testing of SAS Viya VA submitted tests. For non-VA operations, the numbers of directories for the individual file systems in CAS_Disk_Cache can be reduced to around 2 – 4 directories, versus the original recommendation of 4x the number of CPU cores per file system.

## CONTAINERS AND VIRTUALIZATION WITH SAS VIYA

### SAS VIYA FOR CONTAINERS

SAS for containers provides the ability to deploy SAS components within customized container-enabled infrastructures, including Docker and Kubernetes, which are often run in the cloud. Users can interact with their customized container of SAS software using SAS® Studio (a browser-based interface) or Jupiter Notebook (an open-source notebook-style interface). General information about containers and using SAS Viya within them can be found here: http://support.sas.com/rnd/containers/

You can choose a SAS recipe and create custom containers with specific products or configurations. There is a framework, including recipes, to build SAS Viya Docker images and create deployments using Kubernetes, and you can learn more about it here: https://github.com/sassoftware/sas-container-recipes

You can use the recipe process on the GitHub site above, to create a set of containers for SMP and MPP installations. To be able to process the variety of analytic workloads that our customers expect, the products in the containers may use processing approaches that aren't typically cloud-native, such as holding and processing data in memory for long periods. As such, those containers may not respond perfectly to some typical MPP cloud orchestration events (i.e. scaling, deletion of containers, etc.). So, when we currently employ SAS VIYA in containers, we should think of it more as a typical deployment running in a container/cloud environment, maintaining state over time in some of the containers, and not expected to dynamically change in response to typical orchestration events in cloud environments.

Deploying containers following the recipes, methods, and orchestration in the links above will help ensure the best results currently attainable. Containers are lightweight delivery mechanisms, but still require the same underlying physical resources - speed, bandwidth, and performance - that are addressed throughout this paper. SAS Viya performs a lot of concurrent heavy-weight work and needs the underlying power to do that work quickly.

### SAS VIYA FOR VMWARE VIRTUALIZATION

Our experience has generally been that moving from bare-metal hardware to a virtual machine system (for example, VMWARE) introduces a performance overhead of 4 – 5%. This is due to the resources required for hypervisor services, and hypervisor-level operations for replication, vMotion, and so on.

SAS Viya can subsist in a VMware environment once this overhead is accounted for. The typical problem with VMware ESXi hosts, is that they tend to be fixed definition, commodity servers. When deploying SAS Viya in a thinly provisioned environment, ensure you have enough RAM, on-board CAS_Disk_Cache IO performance, and network bandwidth interconnect between hosts to meet your needs. In a thinly provisioned commodity VM environment, the resources might not match up with the EEC sizing suggestion for your host definitions. Some of the more serious performance field issues with SAS Viya performance on CAS systems have involved insufficient network bandwidth between CAS hosts and CAS_Disk_Cache configuration issues.

If your only available option is to place a SAS Viya system on virtual hardware, please consult your SAS Account Team about optimal setup, configuration, and provisioning for success. In addition, the general underlying performance principles of VMware systems documented previously for SAS Foundation, are still applicable:
http://support.sas.com/resources/papers/proceedings17/SAS0552-2017.pdf

## CONCLUSION

SAS Viya, using Cloud Analytic Services, is a highly performant platform for data management, analytics, modeling, machine learning, and AI delivery. With its massive parallel operations model, you can drastically reduce processes execution time, shortening crucial information delivery cycles. The key for large operations is to highly parallelize a process and run it in-memory at very high speeds.

The instantaneous demand of such a parallel load requires sufficient resources. If configured properly, SAS Viya processes can be scaled to meet very aggressive time-to-solution targets. The key to this is to be aware of the node, CPU, Memory, and IO resource guidelines detailed in this paper. Because scale-out job tasks can range from a few dozen CPUs to hundreds or thousands, close attention must be paid to sizing of capacity, bandwidth, and configuration of resources. Further help is available by working through your SAS Account Team to ensure you have the proper technical assistance in this planning. Pursuing the general guidelines identified in this paper with a detailed plan, best predicates a successful SAS Viya deployment.

## ACKNOWLEDGMENTS

Some foundational SAS Viya material from SAS Global Enablement and Learning, SAS Viya 3.4 Platform Architecture was used in this paper. Please see your SAS Education Representative about this course.

The author would like to thank Ken Gahagan, Margaret Crevar, Brian Bowman, Steve Kreuger, and Rob Collum for editing and content assistance of this paper.

## RECOMMENDED READING

General SAS Viya Deployment and Administration Documentation
http://support.sas.com/documentation/onlinedoc/Viya/index.html

SAS® Cloud Analytic Services 3.4: Fundamentals

https://go.documentation.sas.com/?docsetId=casfun&docsetTarget=titlepage.htm&docsetVersion=3.4&locale=en

SAS Viya 3.4 Administration: Orientation
https://go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calchkcfg&docsetTarget=n00004saschecklist0000config.htm&locale=en

## CONTACT INFORMATION

### YOUR COMMENTS AND QUESTIONS ARE VALUED AND ENCOURAGED. CONTACT THE AUTHOR AT:

Tony Brown
SAS Institute Inc.
Tony.brown@sas.com