Using Solid State Disk Array as a Cache for LHC ATLAS Data Analysis

Wei Yang Andrew B. Hanushevsky Richard P. Mount
SLAC National Accelerator Laboratory, USA

Abstract

User data analysis in high energy physics presents a challenge to spinning-disk based storage systems. The analysis is data intense, yet reads are small, sparse and cover a large volume of data files. It is also unpredictable due to users' response to storage performance. We describe here a system with an array of Solid State Disk as a non-conventional, standalone file level cache in front of the spinning disk storage to help improve the performance of LHC ATLAS user analysis at SLAC. The system uses a long period of data access records to make caching decisions. It can also use information from other sources such as a work-flow management system. We evaluate the performance of the system both in terms of caching and its impact on user analysis jobs. The system currently uses Xrootd technology, but the technique can be applied to any storage system.

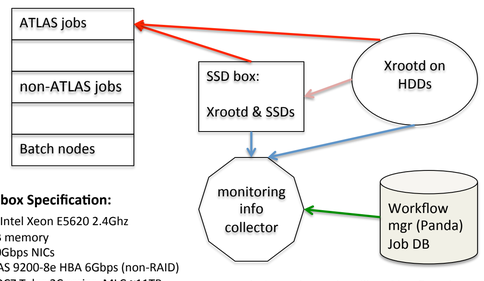
Storage Architecture

Architecture consideration:

SLAC ATLAS Tier 2 runs varied ATLAS analysis jobs via the Grid. Some of them are repeated analysis on the same input data, often within days. So we need long term historical file-accessing info in order to analysis the accessing pattern.

A central SSD storage box is much easier to setup and manage than distributing the SSDs among machines along with HDDs

We cache files at whole file level. Subfile level caching technology using Xrootd is still under development



SSD box Specification:

Dual Intel Xeon E5620 2.4Ghz
24GB memory
2x 10Gbps NICs
LSI SAS 9200-8e HBA 6Gbps (non-RAID)
12x OCZ Talos 2C series, MLC ~11TB
RHEL6 and Xrootd
We use non-RAID HBA and RHEL6 order to pass TRIM to SSDs

Jobs input data, Jobs will try SSDs first, then HDDs if needed
SSD stage-in data from HDD based on caching algorithm
SSD and HDD boxes send Xrootd monitoring info to collector
Fetching a list of input data files of the upcoming Panda jobs

Both SSD and HDD storage systems run Xrootd. ATLAS jobs will read SSDs first, and will read from HDDs if and only if data is not available on SSDs

The monitoring collector host will use Xrootd monitoring information and a list of input data files of the upcoming Panda jobs (from Panda DB) to determine which files will be stage-in from HDDs to SSDs.

To free up space, SSD box will purge old files based on last recent access time stamp.

Xrootd Monitoring Info

An example record of the monitoring info:

unique_id=xrd-1381343744000000
file_lfn=/atlas/xrootd/atlasdatadisk/rucio/mc12_8TeV/ea/1a/NTUP_SUSY.01271227._000009.root.1
file_size=1041320520
start_time=1381343564
end_time=1381343744
…
read_single_average=0.000000
…
read_vector_average=0.000000
…
write_average=0.000000
read_bytes_at_close=24768903
write_bytes_at_close=0
client_host=134.79.128.10
server_host=atlxrd001

These info can be collected but we don't

All storage servers (SSD and HDDs servers) will send monitoring info to an UCSD Collector (developed by the CMS AAA project). The collector will assemble the info and build a record like the above for each access of a file. The collector can send out the assembled info via UDP packet or HTTP text for real time consumption. We save the info as TTree to ROOT files.

A process will periodically look at these ROOT files. By combining these access pattern with the list of input data files of the upcoming jobs it gets from Panda DB, this process determines which files should be cached in the SSDs, and will then ask the SSD box to stage-in the files from HDDs

Other Consideration:

The monitor collector receives file access info from both ATLAS production jobs and user analysis jobs, as well as jobs submitted by local users. All ATLAS production jobs, and some user analysis jobs, copy their input files to scratch space on batch node, while other user analysis jobs read directly from the storage. For sequential file copying, SSDs have very little advantage over HDDs. **Unless otherwise specified, plots in the article have those sequential file copying access filtered out.**

Caching Algorithms & Cache Performance

We used three caching algorithms during this studying:

1. Make caching decision based on file accessing frequency
2. Make caching decision based on bytes read from files
3. Make caching decision based on bytes read from files and list of input files of upcoming Panda jobs

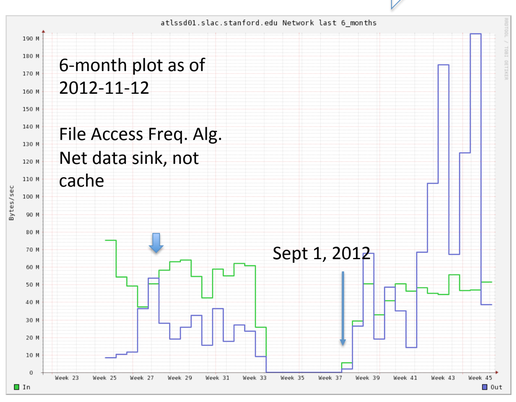
The OCZ Talos 2C series we used in this studying are MLC SSDs. They have limited rewrite cycles. In the above algorithms we make sure that no more than 2% of the total SSDs space will be rewritten in a one hour period.

1. Caching Algorithm using File Accessing Frequency Info

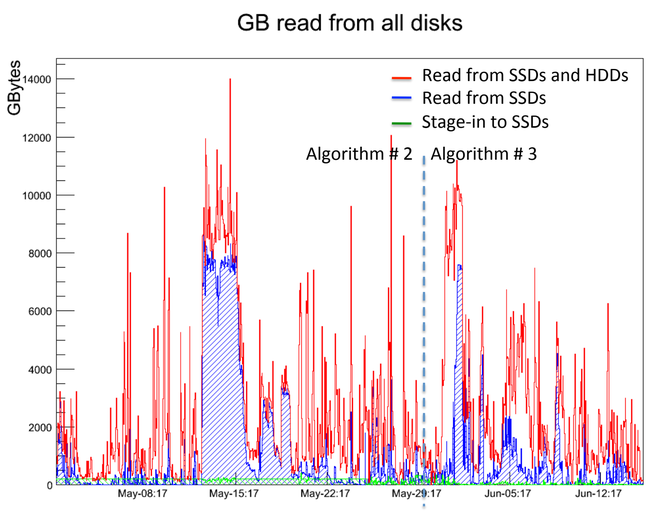
Using the Xrootd monitoring info, we built a table below. We record the number times each file is accessed in each period in the table. A period is 12 housr long. Period 1 is the most recent period and we right shift the table every 12 hours.

	Period 1	Period 2	…	Period 10
File 1	4	2		0
…	1	3		1
File N	2	0		7

Right shift every 12 hours



- The blue line in the plot represent data delivered by the SSDs.
- The green line is represent data staged in by the SSDs.
- We can not consistently use the SSD arrays as an effective cache.



Cache performance during a 45-day period in May and June, 2013

- Y-axis is GBytes/hour
- Each bin is one hour
- The two NICs on the SSD box can deliver up to 2.5GB/s (9000GB/hour)
  When this limit is reach, it may actually slow down the analysis jobs
- Algorithm 3 stages-in fewer data than Algorithm 2.
  There is a stage-in cap of 2% of total SSDs per hour

2. Caching Algorithm using read byte info

Every hour the algorithm builds a table (below) using all file records in the last 5 days' monitoring data, sorts the (right most) column according to average percentage the file is read. It then builds a list by filtering out those files that haven't been read frequently enough (for example, less than 5 times during the last 5 days), or are already in SSDs. The algorithm then triggers the stage-in up to 2% of the total SSD storage.

	Number of reads in 5 days	Average bytes read/ file size
File A	3	0.73
File B	17	0.20
…		
File X	4	0.01

sort

3. Caching Algorithm using read byte Info and Jobs Info

Every hour the algorithm builds the blue cells of the following table using the algorithm above. Every 20 minutes the algorithm inserts two green columns according to the upcoming Panda jobs. For files that do not have records in the last 5 days' monitoring data, the algorithm assumes that 10% will be read each time. It then decides which files are worth to be cached, and stage them into SSDs.

	Number of reads in 5 days	Average bytes read/file size	# of read by upcoming jobs	Total read/file size by upcoming jobs
File A	3	0.73	0	0
File B	17	0.20	2	0.4
File C	0	0.1 (assume)	5	0.5
…				
File X	4	0.01	3	0.03

sort

Impact on Analysis Jobs

Since SSDs don't move mechanical heads in order to read data, the file seek latency is much shorter. One goal of this studying is to learn whether these SSDs in front of the HDDs will speed up or slow down analysis jobs.

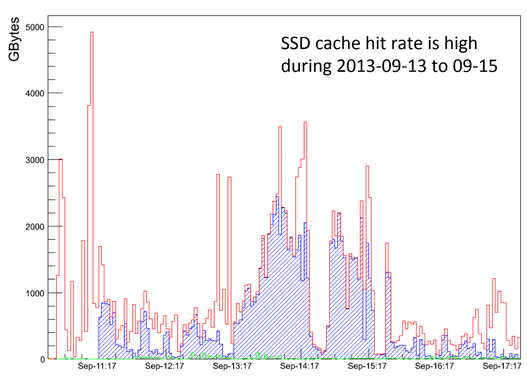
From all 4600 batch slots available to ATLAS jobs during the period of studying, we chose two subsets of them. Each group has 216 slots (on 18 machines). All of them are on identical hardware. By manipulating the firewall rules on Group A, we let jobs running on Group A to skip the SSDs and go directly to HDDs, while jobs on Group B will try SSD first, just like all other ATLAS jobs.

With this setting, we expect that over a long period all jobs, ATLAS and non-ATLAS, will be evenly distributed to these two groups of slots if SSDs and HDDs give the same performance. Since each ATLAS batch jobs will sequentially run as many user analysis tasks as possible, until it reaches certain time limit, we will compare the total CPU time and wall clock time contribution of these two groups in a given period.

2013/09/04 to 2013/10/07	CPU hours	Wall clock hours	CPU/Wall clock
Group A (skip SSDs)	25776.9	34143.3	0.75
Group B	28508.6	35258.7	0.81

The above table show that Group B, which do not skip SSDs, contributed more CPU time, and used the CPU more efficiently during the above 33 days.

Sept 13-15 is a periods when SSD caching hit rate is very high, as shown in the plot below. During the period, we expect input data for jobs running on group B will mostly coming from SSDs, while jobs running on Group A will exclusively read from HDDs.



2013/09/13 to 2013/09/15	CPU hours	Wall clock hours	CPU/Wall clock
Group A (skip SSDs)	49.5	240.2	0.21
Group B	59.8	166.3	0.36

Conclusions and Issues

The SSD cache at ATLAS Tier 2 at SLAC demonstrated that it can help caching data, and thus reduce the load on HDDs. It can further speed up user analysis jobs due low file seek time and high sustained IO per second.

Due to the nature of of Panda based user analysis jobs running at Tier 2, it is difficult to archive high cache hit rate all the time. The setup we have is at R&D stage. The operational complexity makes it vulnerable to mistakes. Some of the software is not of production quality and requires constant manual checking.