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

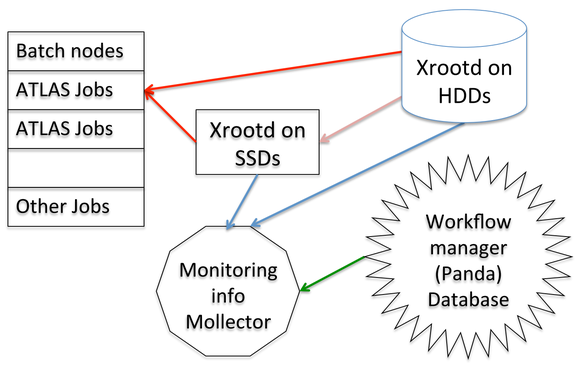
Storage Architecture

Architecture consideration:

A long term historical file accessing information is needed because ATLAS jobs often repeats the access to a same set of files within days.

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

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



- 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

ATLAS jobs read from SSDs first, then HDDs if needed.

The caching algorithms will use Xrootd monitoring information and a list of input data files of the upcoming Panda jobs.

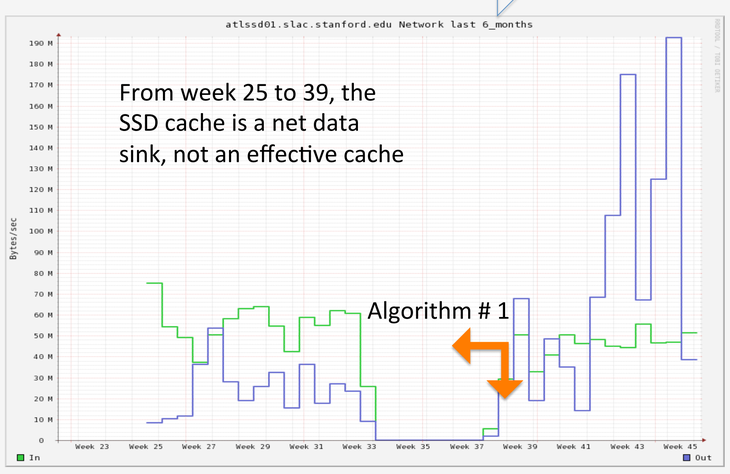
SSD box Specification: Dual Intel Xeon E5620 2.4Ghz, 24GB memory, 2x 10Gbps NICs, LSI SAS 9200-8e HBA 6Gbps (non-RAID), 12x OCZ Talos 2C series MLC (total ~11TB), RHEL6 and Xrootd. We use non-RAID HBA and RHEL6 order to pass TRIM to SSDs

1. Caching Algorithm using File Accessing Frequency Info

Using the Xrootd monitoring info, the algorithm build a table (below) with 10 periods. Each cell is the number times the file is accessed in that period. A period is 12 hours. 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.

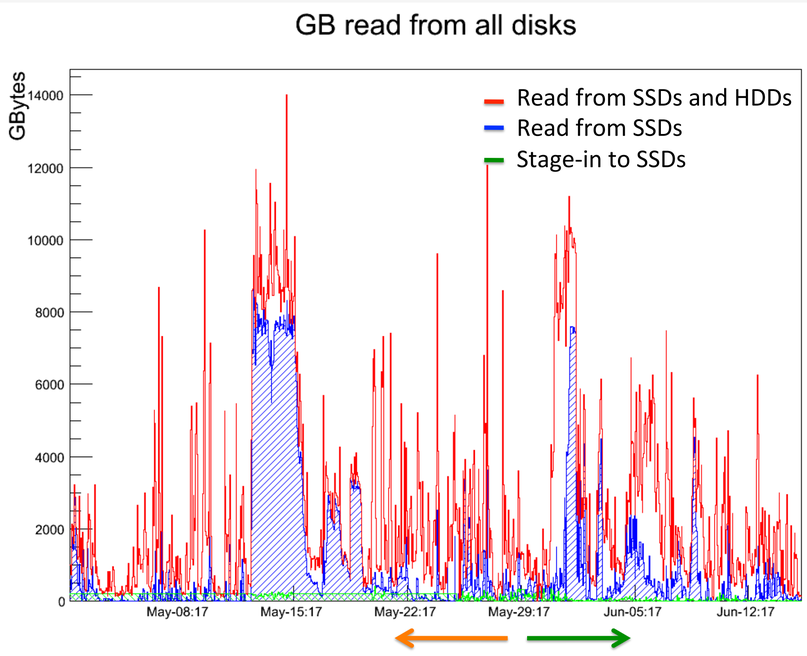
Caching Algorithms & Cache Performance

Three caching algorithms:

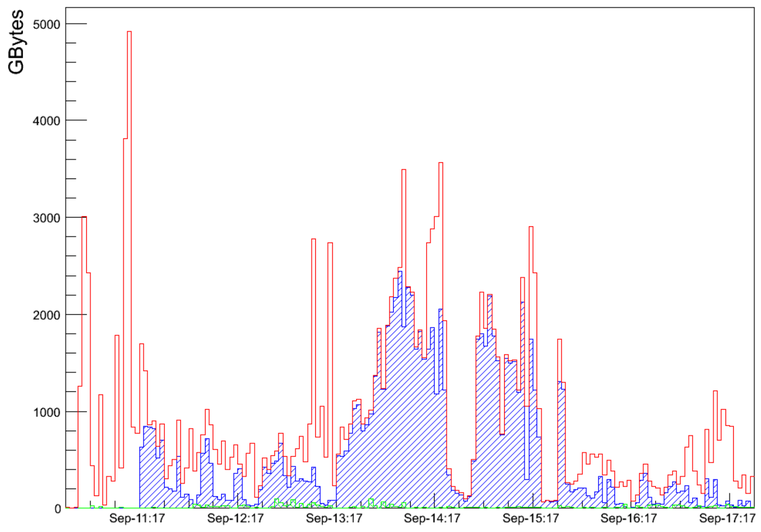
1. Make caching decision based on file accessing frequency
2. Make caching decision based on bytes read from files
3. Make caching decision based on bytes read from files and list of input files of upcoming Panda jobs

The SSDs are MLC, with limited rewrite cycles. Caching algorithms will not rewrite more than 2% of the total SSDs space per hour.

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



Sept 13-15 is a periods when SSD cache hitting rate is very high, as shown in the plot. 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 09/15	CPU hours	Wall hours	CPU/Wall
Group A (SSDs skipper)	49.5	240.2	0.21
Group B	59.8	166.3	0.36

Xrootd Monitoring Info

An example record of the monitoring info:

unique_id=xrd-1381343744000000
file_lfn=/atlas/…/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

we choose not to collect

SSD and HDDs servers will send monitoring info to an UCSD Collector, which build file accessing records like the above. The records are available in real time. We save the info as TTree to ROOT files. The caching algorithms use these records to determine which files should be cached in the SSDs.

Other Consideration:

For sequential file copying, SSDs have very little advantage over HDDs. Caching algorithms filter out those records

2. Caching Algorithm using read byte info

- Every hour the algorithm builds a table (below) from the last 5 days' monitoring data, sorts by the right most column.
- Filter out files that haven't been read frequently enough (e.g. 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	7	0.73
File B	17	0.20
…		
File X	5	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.

	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	7	0.73	0	0
File B	17	0.20	2	0.4
File C	0	0.1 (assume)	5	0.5
…				
File X	5	0.01	3	0.03

sort

Impact on Analysis Jobs

Files access latency on SSDs is very short. One goal of this studying is to learn if SSDs will speed up data intensive analysis jobs.

We choose two subsets of ATLAS jobs 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.

We expect that over a long period, ATLAS jobs will be evenly distributed to these two machine groups if SSDs and HDDs give the same performance. We can not compare at individual user jobs level because each batch job run multiple user jobs. But we can compare the total CPU time and wall clock time contribution of these two groups in a given period.

2013/09/04 to 10/07	CPU hours	Wall hours	CPU/Wall
Group A (SSDs skipped)	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 33-day period.

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 achieve 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.