

Tier-2 Optimisation for Computational Density/Diversity and Big Data

R B Fay¹ and J Bland

Department of Physics, University of Liverpool, Liverpool, L69 7ZE, UK

fay@hep.ph.liv.ac.uk

Abstract. As the number of cores on chip continues to trend upwards and new CPU architectures emerge, increasing CPU density and diversity presents multiple challenges to site administrators. These include scheduling for massively multi-core systems (potentially including Graphical Processing Units (GPU), integrated and dedicated) and Many Integrated Core (MIC)) to ensure a balanced throughput of jobs while preserving overall cluster throughput, as well as the increasing complexity of developing for these heterogeneous platforms, and the challenge in managing this more complex mix of resources. In addition, meeting data demands as both dataset sizes increase and as the rate of demand scales with increased computational power requires additional performance from the associated storage elements. In this report, we evaluate one emerging technology, Solid State Drive (SSD) caching for RAID controllers, with consideration to its potential to assist in meeting evolving demand. We also briefly consider the broader developing trends outlined above in order to identify issues that may develop and assess what actions should be taken in the immediate term to address those.

1. Introduction

With limited resources available, sites have to be as efficient as possible in the use of their hardware and to choose the correct technologies to invest in for maximum performance. This will continue to drive the need to optimise storage and networking solutions, taking advantage of emerging technology solutions where beneficial. Additionally, as General-Purpose computing on GPUs (GPGPU) and MIC architectures become an increasingly significant contributor to High Performance Computing (HPC), increasing computational diversity presents another challenge for High Energy Physics (HEP), particularly in choosing when and what to invest in.

One of the biggest storage technology changes in recent years is the introduction of SSDs. Traditional magnetic disks still have a large advantage in capacity and cost but, while optimisation of topology and TCP/IP settings can offer significant improvements, performance has been a significant bottleneck for many years with only minor improvements. As disk capacities increase and data access demands are accelerated by faster processors disk speed becomes a serious limiting factor in data analysis.

In many sectors the two technologies are being combined with the use of SSD caches of various types, theoretically leveraging the latency advantages of SSDs while still retaining the large capacity and affordability of standard drives. While this is very successful in some areas it does depend on the

¹ To whom any correspondence should be addressed.



access patterns and relative size of data to SSD capacity. Typically HEP storage servers already utilise caching with volatile RAM, using Linux page cache and RAID controller RAM caches. With RAM speeds being even greater than SSDs these can significantly increase the performance of storage but they are typically quite small. With this in mind, Liverpool has assessed one SSD-caching solution with regard to HEP usage.

2. Storage Caching

While there are an increasing number of SSD caching technologies, a convenient one for Tier2 sites is a solution that combines this with Serial Attached SCSI (SAS) RAID controllers. With all of the caching technology and SSDs integrated into a single adapter much of the complexity and configuration of the cache is hidden from both the operating system and administrator. One example of this type of integrated SSD caching is the LSI Nytro range of RAID controllers [1]. We have tested one model (the Nytro MegaRAID NMR8100-4i) with 100GB of SSD accelerated cache with generic storage benchmarks and a more typical HEP data analysis model.

2.1. SSD caching benchmark configuration

- IOzone [2] used for sequential and random read/write tests
- 64bit Scientific Linux 6 install on 32GB 6-core server, “out of the box” setup
- LSI Nytro MegaRAID controller with 100GB SSD cache (Accel)
- Also tested 2xEnterprise SSD in RAID1 configuration for comparison
- File sizes chosen to reflect typical HEP data files plus large files
- Benchmarks performed with Linux page cache on/off and SSD cache on/off
- Sequential and random read/write tested for small and large chunk sizes

2.2. SSD caching benchmark results

In these benchmarks it is seen, as in Figure 1, that the Linux page cache significantly improves read performance in all cases up to the point that the file being cached is bigger than available system RAM. Normal on-board RAID RAM cache also increases performance for direct access to any files smaller than the cache size (1GB in this case) but not as effectively as Linux page cache. The SSD RAID1 array shows comparable throughput for read operations, which is impressive given its relatively small number of drives.

Sequential write throughput, shown in Figure 2, with or without page cache enabled is generally comparable for larger files. The throughput of the array itself is more of a limiting factor during writes. The 1GB RAID RAM cache again provides a significant boost to writes, for small files this can be greater than the increase from Linux page cache. The SSD RAID1 array shows significantly lower throughput for write operations, both direct and when using Linux page cache, even though the file sizes are below the available system RAM. This highlights background disk write operations by the Linux kernel as it flushes any unwritten blocks in the page cache.

Linux kernel parameters e.g. `dirty_ratio` can affect these background operations. This parameter sets the limit at which the kernel blocks further IO operations until the excess has been written to the disk. The default of 20% for this parameter produces a big drop in performance when writing larger files particularly on slow devices, e.g. single SSD, which are unable to clear data in the background fast enough. Raising this to a much higher value e.g. 80% produces a flatter throughput profile in these benchmarks but care must be taken when modifying these parameters as the optimal values depend greatly on the application IO access patterns.

In all tests the throughput with SSD caching (Accel) enabled is often lower than disabled, although any differences are small and can be masked by the page cache. Direct file access, or when files are bigger than the available RAM, shows the greatest difference e.g. in Figure 1.

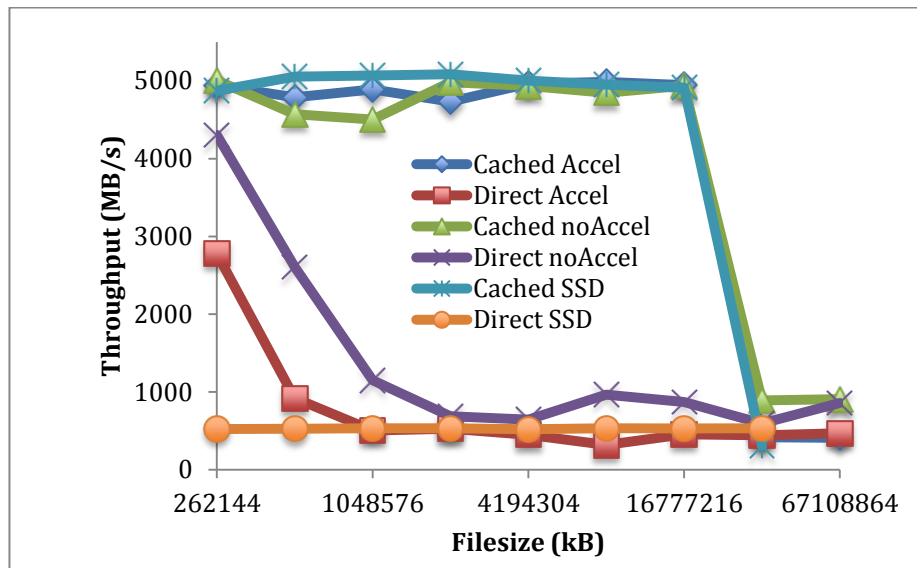


Figure 1. IOzone benchmark results for sequential reads, using a 16MB record size. Results are shown for a RAID6 array with SSD acceleration enabled (Accel) or disabled (noAccel), and with Linux page cache enabled (Cached) or disabled (Direct). A simple two drive RAID1 SSD array is shown for comparison.

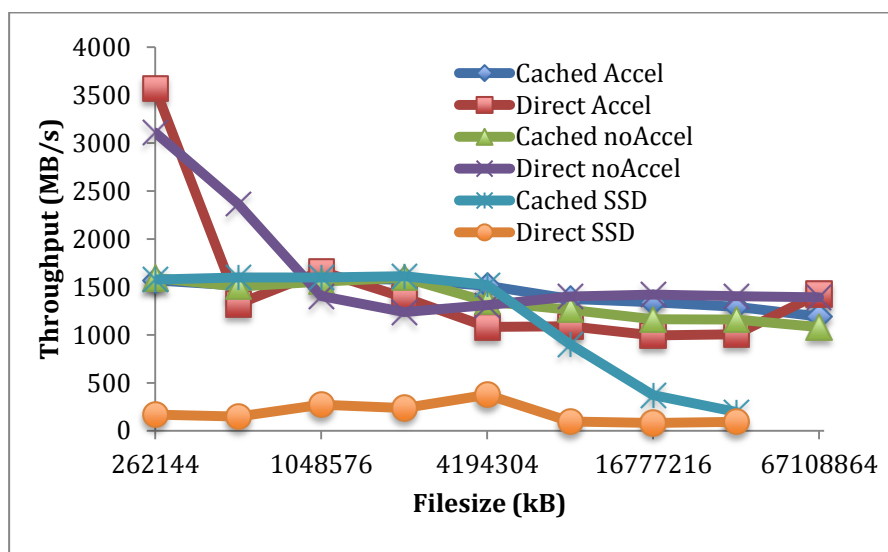


Figure 2. IOzone benchmark results for sequential writes, using a 16MB record size. Results are shown for a RAID6 array with SSD acceleration enabled (Accel) or disabled (noAccel), and with Linux page cache enabled (Cached) or disabled (Direct). A simple two drive RAID1 SSD array is shown for comparison.

2.3. Test Case

While IOzone tests are useful for stress testing storage systems they do not reflect more typical HEP access patterns. They also perform their operations on a freshly written file. We would expect most caching algorithms to be more effective for files that are read more than once. To investigate this we used an example ROOT-based [3] data set stored on an array with the SSD caching RAID controller.

- ROOT tasks accessing files directly from a filesystem on the array
- 12 concurrent tasks accessing different sections of a 0.5TB dataset
- Each task reads in approximately 40GB of data
- Tasks were run multiple times with SSD cache disabled, page cache dropped after each run
- The SSD cache was then enabled
- The tasks were then run multiple times, page cache cleared after each run

2.4. Test case results

The CPU efficiency for each of the task runs is shown in Figure 3. Performance without SSD caching was consistently at ~79% CPU efficiency. On first enabling the SSD cache the performance is degraded significantly. Successive runs show a slight increase in efficiency eventually reaching a maximum, the controller appears to be caching more of the data, but CPU efficiency is always lower than with no SSD caching.

The access patterns on the array during the test were investigated with blktrace (a block layer I/O tracing mechanism included in SL6). A single file is read mostly sequentially but as multiple files are being accessed concurrently there is a moderate rate of seeking on the array. Throughput is always below the limits of either SSD cache or array hence we don't believe the drop in efficiency is due to the SSD cache throughput being saturated.

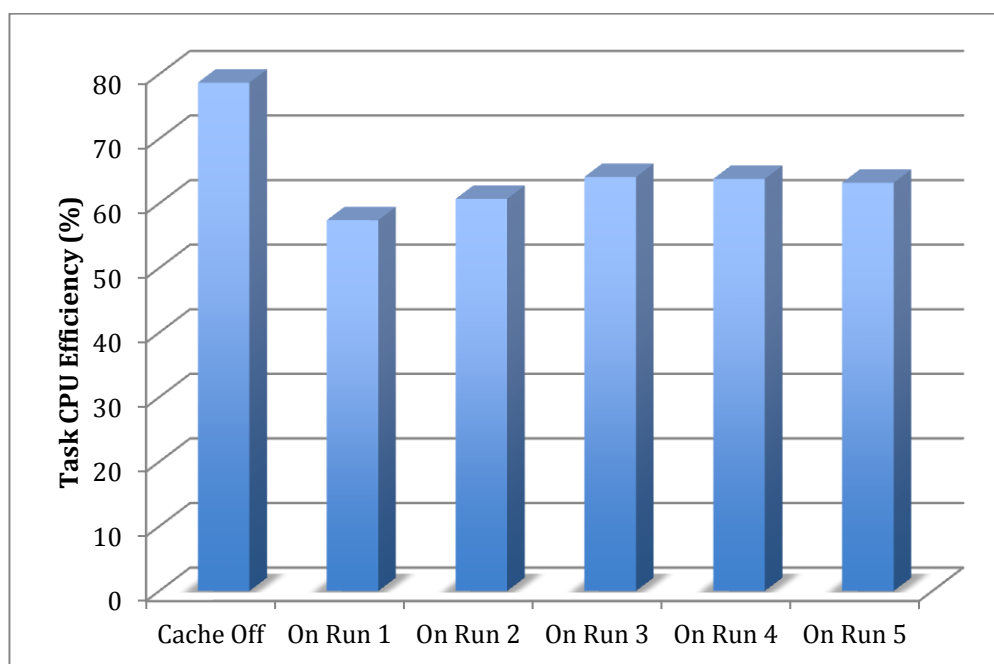


Figure 3. Data analysis CPU efficiency of a ROOT-based data analysis task, with SSD acceleration disabled (Cache Off), then enabled through successive runs.

3. Computing Architecture Diversity

GPGPU/MIC technologies offer attractive potential performance for some applications, and as the technology continues to develop with the prospect of tighter integration between CPU and GPGPU/MIC in the future, it is likely to become increasingly part of the mainstream. But the diversity of technologies comes with its own issues. While developing platform-agnostic code is possible to a limited extent, optimal use of GPGPU/MIC architectures can currently require developing code not only for a specific architecture, but also for specific models of hardware. This presents challenges both

on the development side, with expertise and increased development time required, but also on the site and middleware side in terms of provisioning, information publishing, and scheduling. On the site side, provision of hardware is typically driven by hardware specifications at the time of purchase (e.g. performance per £/\$/€, performance per watt, etc.) but also by user requirements. User requirements are driven partly by technical factors, including potential performance, development support (e.g. ease of coding, extent of pre-existing libraries), but also potentially by availability. This presents something of a “chicken & egg” scenario, in that as adoption drives provision, so provision can drive adoption. With the availability of differing architectures comes an essential choice to the community: standardise on one platform, or provide broad support for multiple platforms. Each approach presents risks, but in the medium to long term, homogeneity across the LHC Computing Grid may not be practical, or desirable at this point in time.

Hence, it becomes necessary to provide cross platform support at this relatively early stage. Ease of development and use for all architectures should hence be prioritised, with consequent requirements for information publishing, job specification and scheduling. Additionally, information flow between developers and sites will be vital to inform both development choices and hardware provisioning.

4. Recommendations

SSD caching doesn’t appear to be effective for HEP data analysis at the moment. While SSDs can give big performance gains in some situations they still have quirks and characteristics that should be taken into account when planning storage configuration. Effective SSD caching will depend on the caching algorithms being tuned for the application, in this case, ROOT analysis access. Performance gains from extra system RAM and tuning parameters can be more cost effective and easy to add. Where appropriate, sites should consider commissioning limited GPGPU/MIC systems as the technology and HEP development proceed to aid and encourage the necessary progress to make optimal use of these technologies.

References

- [1] LSI Nytro RAID controller range <http://www.lsi.com/products/flash-accelerators/>
- [2] IOzone Filesystem Benchmark <http://www.iozone.org/>
- [3] Brun R and Rademakers F 1997 ROOT - an object oriented data analysis framework *Nucl. Inst. & Meth. in Phys. Res. A* **389** 81-86. See also <http://root.cern.ch/>