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

Rob Fay (fay@hep.ph.liv.ac.uk), John Bland (jbland@hep.ph.liv.ac.uk)

Overview

Increasing computing density and data set sizes lead to rising challenges in meeting demands, and increasing computing diversity (GPGPUS, MIC) presents its own problems.

With limited available resources, 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 GPGPU and MIC architectures become an increasingly significant contributor to HPC, increasing computational diversity presents another challenge for HEP, particularly in choosing when and what to invest in.

One of the biggest storage technology changes in recent years is the introduction of Solid State Disks. 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. In many sectors the two technologies are being combined with 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 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.

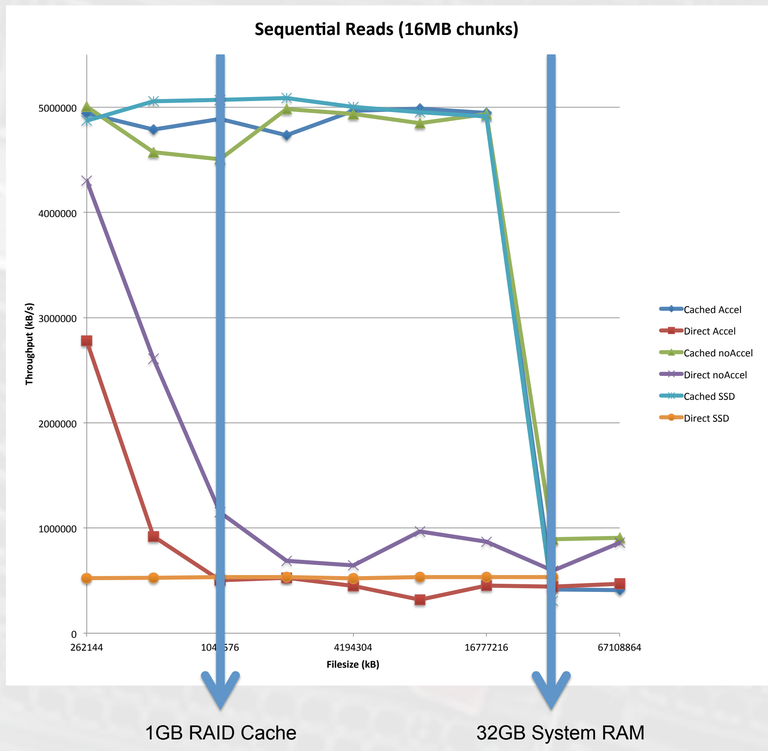
Storage: Caching

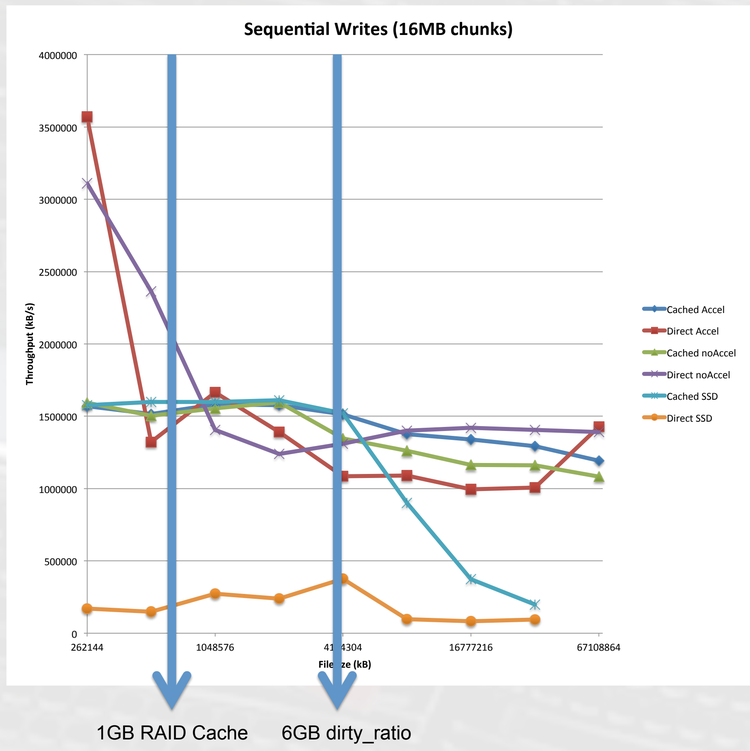
SSD caching benchmarks

- Iozone used for sequential and random read/write tests
- 64bit SL6 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

Results

- Linux page cache significantly improves read performance in all cases, particularly with small (eg 4kB) access, until exhausted
- Normal RAID RAM cache increases performance for any files smaller than the cache size, greater than Linux page cache in some cases
- Linux kernel parameters eg dirty_ratio can affect performance; default of 20% produces big drop in performance when writing larger files particularly on slow devices (eg single SSD) which can’t clear data in the background fast enough
- Performance with SSD caching (Accel) enabled is generally lower than disabled, any gains are minor
- Modest SSDs can match much larger disk arrays for read performance but lag significantly in write performance





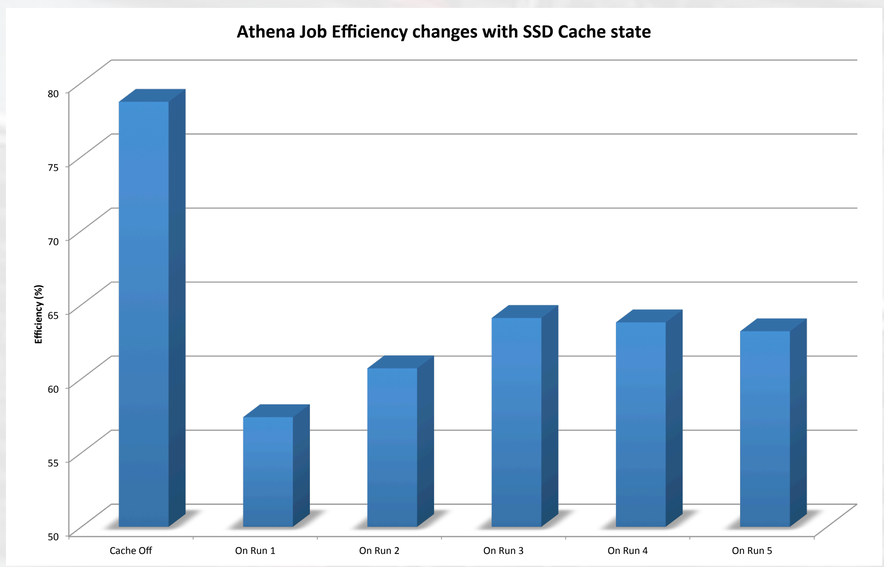
Test Case

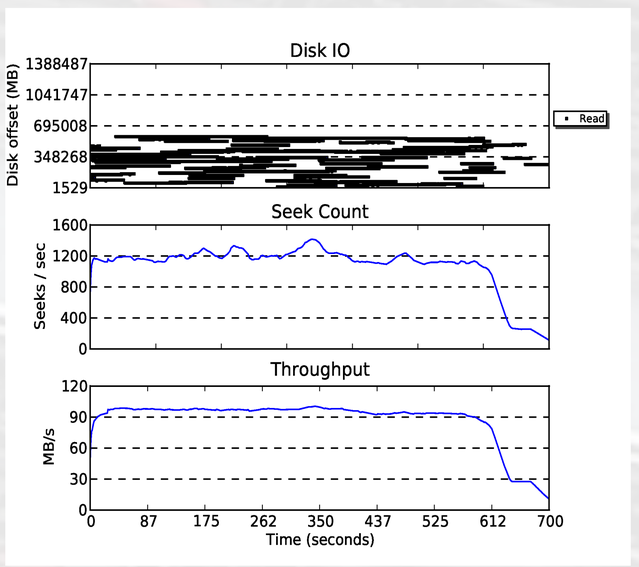
For a test case on more typical HEP data access patterns we used an example ROOT-based data set stored on an array with the SSD caching RAID controller.

- ROOT tasks accessing files directly from filesystem on 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 no SSD cache enabled and page cache dropped after each run
- The SSD cache was then enabled
- Tasks run multiple times, page cache cleared after each run

Performance without SSD caching was consistently at ~79% CPU efficiency. Enabling the SSD cache degrades the performance significantly. Successive runs show a slight increase in efficiency, the controller appears to be caching more of the data, but always lower than no caching.

The access patterns on the array were investigated with blktrace. A single file is read sequentially but multiple files are being accessed concurrently, producing a moderate rate of seeking on the array. Throughput is below the limits of either SSD or array.





On 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 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 differing architectures comes an essential choice: 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 LCG 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 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.

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
- Performance gains from extra 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.

UNIVERSITY OF LIVERPOOL

GridPP
UK Computing for Particle Physics