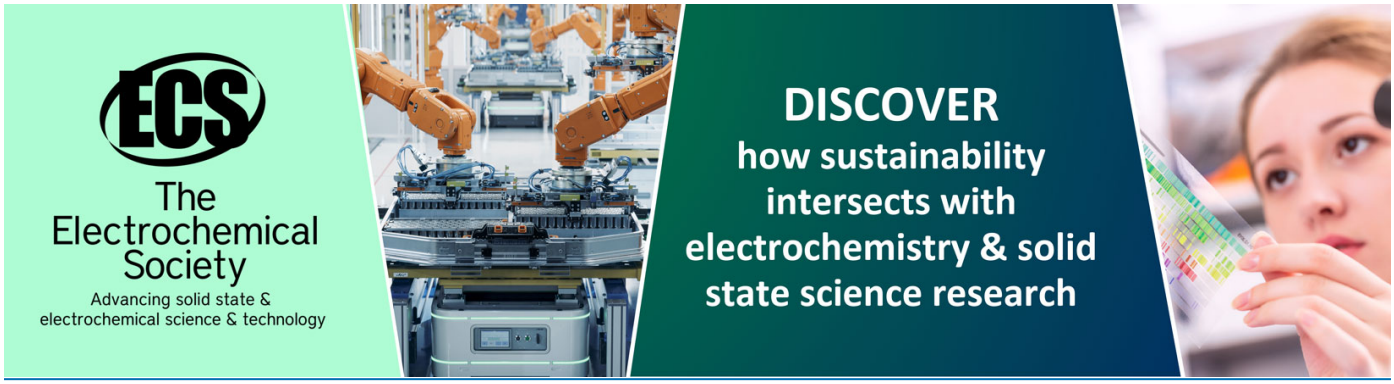
STAR load balancing and tiered-storage infrastructure strategy for ultimate db access



Journal of Physics: Conference Series **331** (2011) 042028 doi:10.1088/1742-6596/331/4/042028

STAR load balancing and tiered-storage infrastructure strategy for ultimate db access

D Arkhipkin, J Lauret and W Betts

Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000 USA

E-mail: arkhipkin@bnl.gov, jlauret@bnl.gov, wbetts@bnl.gov

L Didenko and G Van Buren

Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000 USA

Abstract. In recent years, the STAR experiment's database demands have grown in accord not only with simple facility growth, but also with a growing physics program. In addition to the accumulated metadata from a decade of operations, refinements to detector calibrations force user analysis to access database information post data production. Users may access any year's data at any point in time, causing a near random access of the metadata queried, contrary to time-organized production cycles. Moreover, complex online event selection algorithms created a query scarcity ("sparsity") scenario for offline production further impacting performance. Fundamental changes in our hardware approach were hence necessary to improve query speed. Initial strategic improvements were focused on developing fault-tolerant, load-balanced access to a multi-slave infrastructure. Beyond that, we explored, tested and quantified the benefits of introducing a Tiered storage architecture composed of conventional drives, solid-state disks, and memory-resident databases as well as leveraging the use of smaller database services fitting in memory. The results of our extensive testing in real life usage are presented.

1. Introduction

The RHIC Facility is a National User Facility at Brookhaven National Laboratory [1] that delivers the world's highest energy heavy ion collisions and colliding polarized proton beams for experimental research in nuclear physics. The Solenoidal Tracker at RHIC (STAR, [2]) is one of the large detector systems at RHIC, and it accumulates petabytes of raw data a year at DAQ rate of 1000 Hz. This implies several hundred gigabytes of associated metadata per RHIC Run, being collected and used in physics data production. Metadata are subsequently converted into physics-usable quantities suitable for data production. This process may be iterative: calibration quality converges at different speeds depending on the detector sub-system.

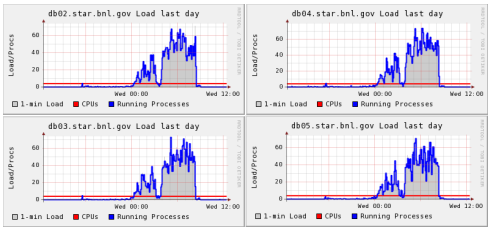


Figure 1. STAR Offline DB server load typical for stream data production. Load Balancing works, but it is simply not enough to deal with the load.

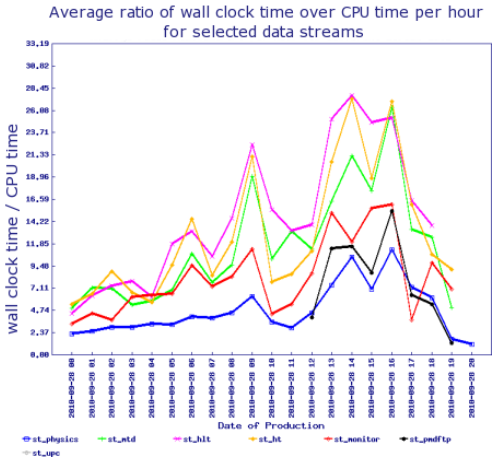


Figure 2. Average wall clock time / CPU time per hour: "stream" data production before optimizations.

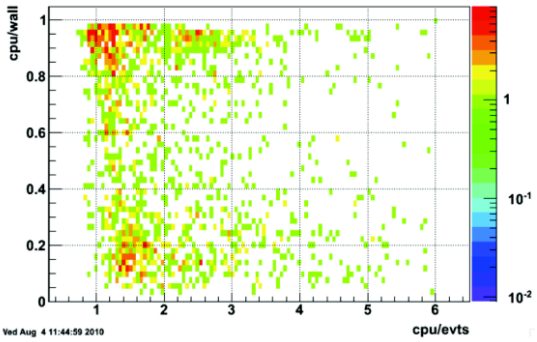


Figure 3. CPU time / Wall clock time ratio: "stream" data production. Two distinctive groups of events could be clearly identified.

Typical bottlenecks associated with data production in experiments comparable to STAR generally include track reconstruction, IO and possibly database access. Our work mainly focuses on database access optimizations so it becomes an insignificant part of the workflow.

Our first level improvement was load balancing (as published in [3]), and as illustrated on Fig. 1, our strategy seem to work well. Plots represent the load of the servers as a function of time - based on Ganglia monitoring framework [4]. We can clearly see that at equivalent power, all servers are equally used and at any point in time as explained in our past publication. This is not the case when DNS round-robin alone is used. However, in the performance topic, there are much room for further improvement as usage becomes more specific.

In 2010, moving toward rare event selection up-stream of the data taking process, the STAR experiment created a special set of data (known as "stream" data) believed to allow faster access to key Physics topics. However, these data being sparse in time, re-usability of cached and queried set of calibration constants from many concurrent processes accessing our service is not only unlikely but is not occurring. As a result, the data processing is slowed down due to the database access depending on the kind of "stream" data (as illustrated in Figs. 2 and 3). Problem analysis reveals that event sparsity (physics events not time-ordered and/or very separate in time) cause nearly random access to the database. In essence, the events in a given file being processed are separated in average by time far greater than the granularity of the calibration data validity. This implies that each event requires a fresh set of data to perform its calibration correction. One may consider re-ordering of events, but over a workflow of 1000 jobs, synchronization is non-trivial and not immediate. Such a database bottleneck needs to be addressed using different techniques then load balancing alone. For example, random data access performance needs to be addressed at the core. Before designing new solution, schema or new mechanisms to deliver the metadata to the jobs, the database infrastructure and layers need to be analyzed and understood.

2. Initial schema

The data reconstruction process essentially makes use of what is named "Conditions" metadata. The "Condition" database contains calibration constants: detector states, gas pressure, temperature and other environmental conditions. To organize the Conditions metadata service, STAR utilizes a typical data warehousing schema as illustrated at Fig. 4. It is composed of an

operational layer, a data access layer and an informational access layer (user and API).

- The operational Database Layer resides in the "online" domain, stores comprehensive amount of experimental data monitoring, alarms, beam and subsystem parameters. It may contain non-structured data;
- The Data Access Layer: converts raw data stored in "online" domain to stable "offline" domain format, sometimes, this process involves non-trivial transformations. In other words, the data access layer may proceed with data reduction and aggregation, conditional selections - a few environmental values combined may lead to a physical value if some criteria is met. Its information is no longer "raw", but preprocessed;
- The Metadata Layer is a database schema definitions, required to support data structures evolution;
- The Informational Access Layer, or "offline" domain, contains structured, packed data optimized for fast retrieval and processing by production workflows. A high level end-user API is available at this level, hiding the complexity of data retrieval in regards of time the event occurred, or in regards of incremental updates to the database, preserving and ensuring reproducibility of past results;

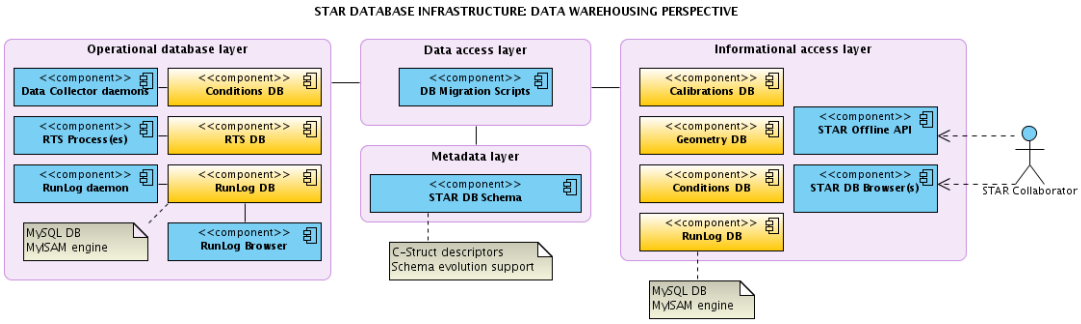


Figure 4. STAR database warehousing overview.

In our studies, we focused at the "last mile" issues: metadata transfer from Offline database to production jobs. Effectively this means that we need to find a proper way to allow several thousand of concurrent processes to retrieve detector state information and apply it to raw data stream, which is a very typical task in our workflow. Let us partition the problematic areas into the following categories: a) database storage performance, b) caching options, c) alternative ways to access metadata. Each of those stages could lead to different improvements and applied techniques. Improving database storage performance may however be a complex process, involving a choice and evaluation of the underlined technologies used for storage. Often, simplistic reasoning ("faster is better", for example) may not apply as the problem may very well be more dependent on random access to storage media rather than linear read speed. We investigated the following storage types: Serial Attached SCSI drives, Solid State Disk drives, and Dynamic RAM drives. Test setup included the following:

- Hardware: Quad-Core Intel Xeon E5520 @ 2.27 GHz, RAM: 12 GB - constrained to 2 GB for certain tests to reduce filesystem caching effects;
- OS: Scientific Linux 5.3 (Boron), x86_64, 2.6.18-164.15.1.el5, all drives were mounted with *noatime* filesystem flag to reduce the amount of "write" IO operations per request;
- Database type: MySQL 5.0.77 [5], x86_64 version, MyISAM table type, Query Cache size: 128MB, Key Buffer size: 256MB

Additionally, we explored several time-series data caching strategies, and various types of relational and noSQL databases.

h!

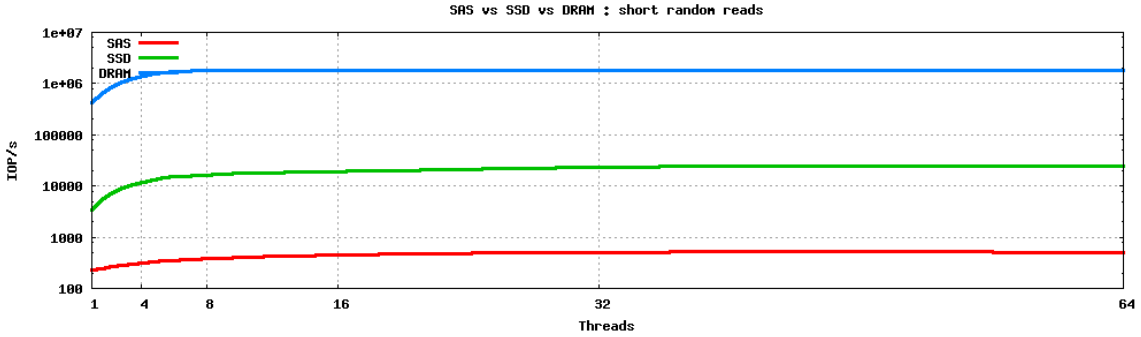


Figure 5. SAS/SSD/DRAM measured IOP/S vs Number of concurrent threads. Our results show perfect match with manufacturers' specs.

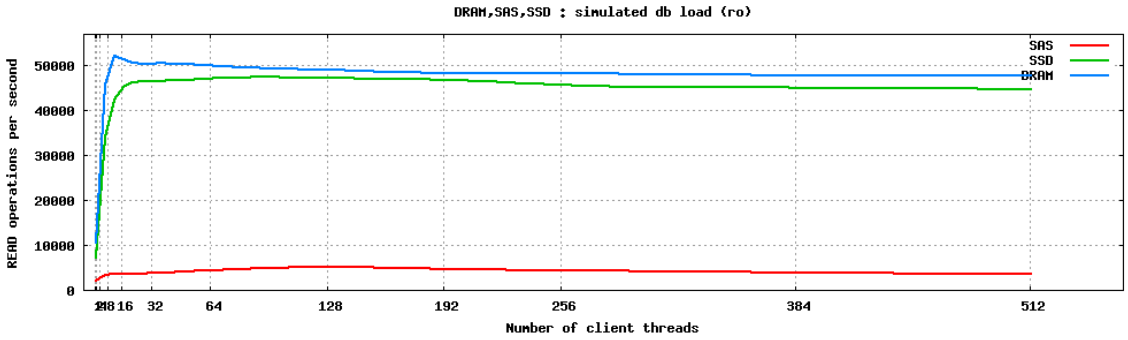


Figure 6. SysBench results for simulated load tests of MySQL 5.0 number of transactions per second vs. number of concurrent threads. Similar performance results for SSD and DRAM are likely due to CPU limit reached.

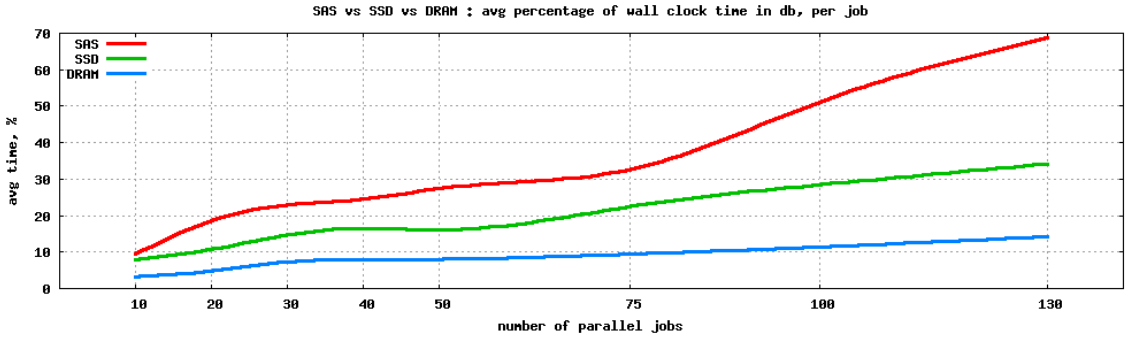


Figure 7. Real STAR job performance results: total time spent in db access code (smaller is better) vs number of concurrent jobs accessing same database. Results indicate quite moderate performance improvement for SSD/DRAM over SAS.

3. Storage performance testing

In order to evaluate database storage performance, we chose the following process: first, we measured the basic filesystem performance for all three storage types, then we measured performance of the system under simulated stress test, and, finally, we run a real STAR production job stress tests. Reported performance testing results are split in three groups:

- Filesystem IO Testing: reference point, with no databases involved at all (Fig. 5).
- Simulated DB Load Testing: core MySQL engine testing with different storage types, with no STAR-specific details. Provides ideal reference point for Real Jobs testing (Fig. 6).

- Real Jobs Load Testing: quantification of the benefit for STAR database cluster performance. SysBench is selected to be the tool of choice for FileSystem and Simulated Load testing (Fig. 7).

3.1. Summary of hardware testing

Database index search or sparse data read emulation: SSD performs 50 times better than SAS HDD in short random reads (64 threads, 4kb blocks, random read) : 24000 IOP/s vs 500 IOP/s; Sequential table scan emulation: SSD performs 2 times better than SAS HDD in flat sequential reads (64 threads, 256kb blocks, sequential read) : 190 MB/s vs 90 MB/s; Database insert speed: SSD performs 10 times better than SAS HDD in short random writes (64 threads, 4kb blocks, random write) : 6800 IOP/s vs 750 IOP/s; DRAM outperforms every other storage type by factor of x100. Obvious drawbacks: DRAM is non-persistent, and size-limited, compared to SAS/SSD disks. SysBench indicates similar performance for SSD and DRAM storage types most likely, a CPU limit has been reached. Read data production tests indicate moderate performance improvement in case of SSD/DRAM compared to SAS, with DRAM being most tolerant to number of concurrent users.

3.2. Concurrent ways to improve database performance

Metadata access improvements also include optimized client-side caching for time-series data and enhanced data access via Web Service interface to utilize remote resources. We note that caching would work to the extent that one job J landing at a later point in time on the same node than job I (reusing some of the cached data) will benefit from this mechanism. Since there are many possible bottlenecks involved, we developed StHyperCache package, which hides local FileSystem-based caching, local RAM caching and distributed RAM caching (memcached) implementations behind a simple interface tailored to STAR needs. StHyperCache is built upon Virtual Factory pattern, and allows easy configuration via XSD-aware XML configuration file. This allows STAR to use the best-suited caching method completely transparently to client code. Local file system caching also allows use of "database snapshots", to be shipped along with jobs to GRID/Cloud resources (as an example of use where it is not un-common to have worker processing nodes isolated from WAN). The most significant difference between conventional caching libraries and StHyperCache is that the latter one enables efficient caching for **time-series** data, not just plain key/value streams.

4. Recommendations

The following set of recommendations, based on aforementioned hardware testing results and software tuning exercises, was developed as a final solution:

- Use MySQL master-slave replication + real-time load balancing to allow fault-tolerant data processing;
- Regroup production jobs in time-ordered chunks random access significantly decreases overall performance no matter what hardware is used;
- Arrange yearly database "snapshots": smaller database chunks that fit in RAM allows to completely avoid performance drops associated with random access;
- Tables, that require special attention could be placed on SSD or RAM drives utilizing MySQL symlinks feature. MySQL Partitioning, featured in MySQL 5.1+, helps to ease processing of large tables containing millions of records;
- Use SAS for large volumes of data, primarily accessed in sequential manner;
- Employ optimized caching mechanisms on a client, to reduce number of data requests to acceptable level;

- Future perspective: evaluate Web Service approach to enhance scalability and load distribution (ongoing R&D in STAR).

After proceeding with optimization including a smaller snapshot, performance tuning and event time ordering for a sample data, the performance was re-assessed. Figure 8 shows the resulting improvement, an overall factor of x5 over a non-optimized database as seen on figure 2, considering the same data samples and job distribution. More results may be found in reference [6].

5. Summary

In this paper, we presented our work in the area of metadata storage and access optimizations. We explored, tested and quantified the benefits of introducing a Tiered storage architecture composed of conventional drives, solid-state disks, and memory-resident databases as well as leveraging the use of smaller database services fitting in memory. The results of our extensive testing in real life usage are presented. Our solution regards metadata storage and access as a complex problem within a multi-dimensional space involving production requests rearrangements to optimize database queueing, replicated load-balanced clustering of database nodes and combined Tiered approach for efficient data storage, augmented by alternative ways to access data (caching strategies, web service approach). While there are many generic ways to improve database access performance, we believe that our work clearly outlines techniques highly recommended for modern Heavy Ion experiments.

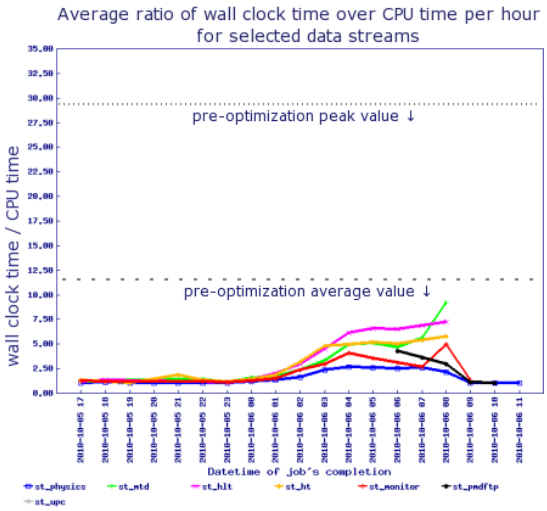


Figure 8. Average wall clock time / CPU time per hour: stream data production **after** optimizations. Please compare to Fig. 2

Acknowledgments

This work was supported by the Office of Nuclear Physics within the U.S. Department of Energy's Office of Science.

References

[1] Brookhaven National Laboratory is operated and managed for DOE's Office of Science [Online] URL `http://www.bnl.gov/`
[2] The Solenoid Tracker At Rhic (STAR) experiment [Online] URL `http://www.star.bnl.gov/`
[3] M DePhillips *et al* 2008 *J. Phys: Conf. Ser.* **119** 042009
[4] Ganglia Monitoring System [Online] URL `http://ganglia.sourceforge.net`
[5] MySQL [Online] URL `http://www.mysql.com`
[6] Effect of stream data on database performance, 2010 study [Online] URL `http://drupal.star.bnl.gov/STAR/blog/jeromel/2010/aug/04/stream-effect-database-summer-2010`