Even Bigger Data: Preparing for the LHC/ATLAS Upgrade

V. Oliveira[1], A. Pina[1], N. Castro[2], F. Veloso[3], A. Onofre[2]

[1] Departamento de Informática, Universidade do Minho,
Campus de Gualtar, Braga, Portugal
[2] Laboratório de Instrumentação e Física Experimental de Partículas,
Departamento de Física, Campus de Gualtar, Braga, Portugal
[3] Laboratório de Instrumentação e Física Experimental de Partículas,
Departamento de Física, Universidade de Coimbra, 3003-516 Coimbra, Portugal

{vspo, pina}@di.uminho.pt, {nuno.castro, filipe.veloso, antonio.onofre}@cern.ch

Abstract. The Large Hadron Collider's (LHC) experiments' data volume is expected to grow one order of magnitude following the machine operation conditions upgrade in 2013-2014. The challenge to the scientific results of our team is: i) how to deal with a 10-fold increase in the data volume that must be processed for each analysis, while ii) supporting the increase in the complexity of the analysis applications, iii) reduce the turnover time of the results and iv) these issues must be addressed with limited additional resources given Europe's present political and economic panorama. In this paper we take a position in this challenge and on the research directions to be explored. A systematic analysis of the analysis applications is presented to study optimization opportunities of the application and of the underlying running system. Than a new local system architecture is proposed to increase resource usage efficiency and to provide a gradual upgrade route from current systems.

Keywords: Big Data, IO profile, performance analysis, ATLAS collaboration

1 Introduction

The Large Hadron Collider's (LHC) experiments at CERN generate an overwhelming amount of information. They build data stores that are part of a growing family of "big data" repositories [1, 2] expected to have large impact on society [3]. The size of the ATLAS [4] data set that our team is working with starts in the multi-petabyte range and after several pre-processing stages the data volume that reaches the analysis applications is in the tens of terabytes range, which is barely manageable. However, this data volume is expected to grow one order of magnitude following the machine operation conditions upgrade after the shutdown in 2013-2014.

In order to produce useful science in useful time with this dataset there must be a trade-off between the complexity of the analysis for each event and the number of events that are selected to analyse given the large volume of data that must be stored and processed. In fact, the number and complexity of the correct solutions that best describe the collected data at the LHC depends crucially on the efficiency of the available computational resources.

The ability to fully explore the physics potential of the LHC analysis program developed by the team [5, 6, 7, 8, 9, 23] depends on the efficient use of local resources, with direct impact on the scientific results expected by the team. In order to remain competitive in the ATLAS collaboration we would like to improve the quality of our analysis and, for that, one or even two orders of magnitude increase in total processing resources may also be required. But a large increase in the processing time is also unacceptable; the time currently spent is already limiting our scientific results.

A grand challenge is now presented: i) how to deal with a 10-fold increase in the data volume of each analysis, while ii) supporting the increase in the complexity of the analysis applications and, iii) reduce the turnover time of the results. Given Europe's present political and economic panorama, or just to make it difficult, we must also confront these issues with very limited additional resources.

The base requirements demand a significant joint effort of the physics and computational sciences communities must be devoted to finding adequate approaches to computing. In this paper we take a position in this challenge and explore possible research directions. Next section starts by presenting an analysis of our applications to evaluate the optimization opportunities from the computational points of view given the measure profile. In section 3 the optimizations of the underlying local system are studied particularly for those applications. In section 4 focus turns to an adaptive system architecture that, over time, can deliver the hardware resources required in a sustainable way, increasing the efficiency of how local resources are used and providing a gradual upgrade route that is adjusted to the scarce funding available.

2 Computing at the LIP/ATLAS group

The ATLAS Event Data Model [10] defines how data taken by the ATLAS detector is stored and analysed. Data is first delivered to the CERN Tier-0 computing clusters and from there is distributed to the 10 Tier-1 data centres, spread by different countries, which are used for central processing and reconstruction of data events and simulation of Monte Carlo events. Tier-2 sites are dedicated to further processing and reconstruction of data and Monte Carlo events, while Tier-3 sites are used to perform data analysis and simulation.

Data events are reconstructed with the ATLAS software framework ATHENA and are then stored in the ESD format (Event Summary Data, 1 MB/event), including among other information, the trigger, tracks, calorimeter clusters, and combined reconstruction objects. The information stored in the ESD is combined into final state reconstructed objects, namely photons, electrons, muons, jet, etc., and stored in the AOD format (Analysis Object Data; about 200 kB/event), which are them skimmed, slimmed and/or thinned into the Derived Physics Data format (DPD; about 10 kB/event) used in the physics analyses. Monte Carlo simulated events are used to compare data with different theoretical expectations. The reconstruction of final state particles of Monte Carlo events is based on the simulated signals of each ATLAS sub-detector, done with GEANT4 and using ATHENA. The signals of each sub-detector are digitized and stored in the ATLAS RAW format (about 1.6 MB/event). Monte Carlo events are then reconstructed using the same procedures used with data. The typical time for completely simulating an event is about 15 minutes.

The final step of the data analysis is performed by the group at the portuguese Tier-3 and on local clusters. This step, which includes the data calibration and evaluation of systematic uncertainties, is performed using LipCbrAnalysis, an analysis framework based on ROOT [11]. Each analysis performed by the group requires reading the full data and simulation events, stored locally in the DPD format. This corresponds presently to about 7 TB of disk space usage. Typically each analysis has to be run around 50 times before converging to a final result.

3 Optimizing the software architecture

The data analysis applications presented above constitute the state-of-the-art in our physics analysis methods. They are major assets of this team that took a significant amount of time to develop and test during the course of several years. The software development methods used in these applications allowed them to be swiftly updated over time to accommodate new analysis and rapidly changing system parameters, essential to produce useful results in the ATLAS collaboration. Next sections present an effort to identify the performance bottlenecks and the optimization opportunities, both of which must be addressed in order to withstand large increases in data volume and processing time while, hopefully, reducing turnaround time.

3.1 Application profilers

There are many profiling tools to allow applications to be analysed [12], focusing mostly on the time that is spent by the CPU on the program functions, over all the program time. There are also utilities that take advantage of hardware counter support to build profiles with instruction and memory access patterns, valuable to identify cache and memory access problems [13]. While these are essential views, applications that are highly dependent on the data flow also demand the less readily available IO profile to show how the underlying devices are used and to increase control over those devices. There is also the need to provide not only full execution profiles, but also window-oriented profiles that show how the resources are used over time.

The **tomograph** profiler, still in development, provided most of the information for the following section. It uses a combination of system call intersection, stack-trace unwinding and system /proc monitoring to present an integrated perspective of the aforementioned profiles by integrating along with the typical call-stack view the application memory usage and IO patterns over time, without changing or recompiling the applications. The IO patterns include the specific files that are used and the data volume that is read or written to them, and the ratio of random to sequential accesses for files and devices, with accounting to measure the time each IO function takes and the bandwidth of each device.

3.2 The ttbar_dilep execution profile

In this section we present the profile of the **ttbar_dilep** application. This application was elected from several IO bound analysis applications that deal with ROOT in a similar manner. Fig. 1 presents the main application indicators, including CPU usage, memory usage and IO usage over time, obtained running it in Xeon X5600 based nodes configured with Scientific Linux 5.7 (kernel 2.6.18-274) accessing a dedicated DELL MD1000 storage system with 15 2TB SATA disks.

The application consumes a little above 80% of one core of the CPU, wasting the remaining 12% to 20%. It uses 500MB of RAM memory, an amount that is kept stable throughout the execution. It reads around 5GB of data, in blocks of 9K at a constant rate of about 70MB/s, but writes are not significant.

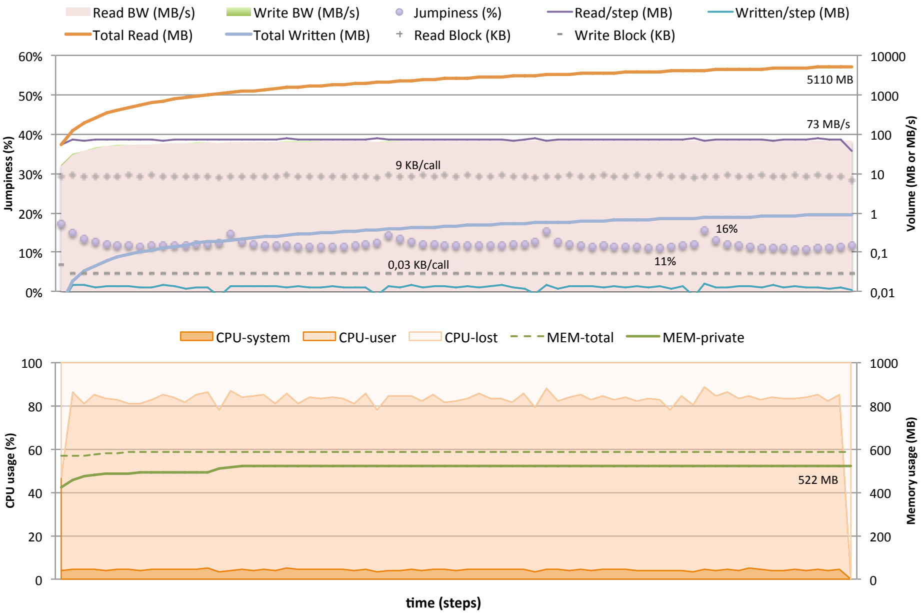

Fig. 1. Overview of the execution: top) input/output usage, bottom) CPU and memory usage.

Access pattern is not sequential, as there is a large number of disk seeks (represented by the jumpiness factor) that varies from 12% to 16%. Note, that jumpiness peaks coincide with processor usage dropping below the 80% mark. The large volume of IO seen in Fig. 1 is the result of the file access patterns. The observed file profile (not presented) shows that, as expected, the application files responsible for most of the IO volume are ROOT's files and that the jumpiness factor comes precisely from these files, with a 12% overall average. Fig. 2 depicts the function call stack, which includes the CPU time of each function and sub-function that consumes more than 3% of the total execution time. Execution time is divided between the main analysis functions (6%), the event reading process (50%), the data calibration operations (42%), the data decompression (18%) and some orphan functions (3%).

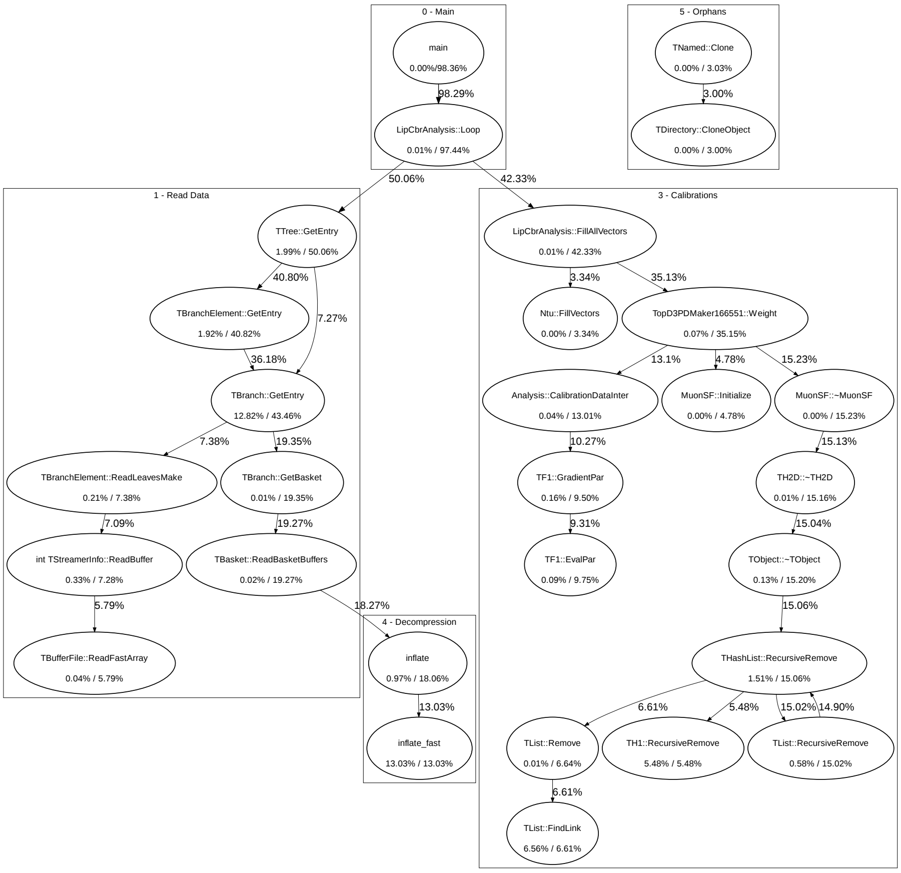

Fig. 2. The ttbar_dilep call stack.

3.3 The ROOT of some evil

The interface between the analysis applications and the storage system is implemented using ROOT, which is also at the core of the execution time; the data reading process takes half of the total application time, reduced to a third after excluding the data decompression time, but the calibration process consumes most of the remaining time.

In ROOT version 5.28 used in the tests there were some issues with the library code itself, including a function that in **ttbar_dilep** is responsible for the opening and closing a file around 35 thousand times, each time writing 29 and reading 29 bytes; or the destructor for TH2D which, unjustifiably, consumed 15% of the overall time.

Next sub-sections present optimizations that may provide a performance benefit without sacrificing existing applications and partnerships.

Eliminating repeated work. An effective way to reduce application time is to avoid repeating calculations whenever possible. Fig. 2 shows that calibrating data with the latest sensor parameters imposes a significant overhead to execution time (42%), but this operation should only be done once for each event, as long as the calibration parameters do not change and the calibration data is saved for latter reuse.

The dataflow sequence determines that each event is pre-processed once for initial selection and then stored locally and read many times for analysis. The pre-processing stage could save data already calibrated, thus avoiding repeated calibration. But when the calibration parameters do change data must be reprocessed directly from the source site. An efficient alternative is to have the data calibrated a single time locally and use cache files with the calibrated data for future reuse. The compatibility requirements on these cache files can be relaxed in favour of high-performance alternatives, given that they are, like the calibrations, already application specific. It then becomes possible to use formats other than ROOT's in this specific setting without losing interoperability.

Data compression and decompression. The size of the ATLAS data files is significantly reduced by the compression used in ROOT, but as Fig. 2 shows decompression consumes 18% of the application CPU time of **ttbar_dilep**.

The time it takes to compress and decompress data may be offset by the reduced time associated with writing and reading less data to disk, so compression by itself may provide a significant time gain in slower storage systems. Users can adjust ROOT file compression at file creation time, presently choosing between LZ77 (ZLIB) and LZMA. Both schemes allows multiple compression levels and are asymmetrical – compression takes more time than decompression – which is adequate for applications where files are written once but are read and processed many times.

However, when the storage bandwidth exceeds the compression/decompression bandwidth the process the compressor becomes the bottleneck. Given a single application with a file pattern mostly sequential the system can be optimized per file to read-ahead the required data at almost the available bandwidth. Since a modern spinning-disk bandwidth can exceed 150 MB/s (see Fig. 3) it can be seen from Table 1 that reading a file in a common local disk already exceeds the decompression capacity of a single processor core. This is further aggravated by the fact the ZLIB, the LZ77 library used in ROOT, does not support multi-threading. Compression schemes that trade a small percentage of disk space for a significantly higher compression and decompression bandwidth have become attractive in recent years, and several high-performance libraries such as LZ4 [14] appeared.

Table 1 presents a comparison between ZLIB and LZ4 on the files being used by our applications. The base LZ4 compressor is more than an order of magnitude quicker at compressing than ZLIB, even at it's faster setting, but it has a 10% cost at the compression level, leading to files 25% larger than ZLIB's at the default compression (N6). The LZ4 HC compressor, on the other hand, compresses almost at the same speed as ZLIB N6 and produces files that are 8% larger, but decompresses 5 times faster. While not a panacea, the inclusion of the LZ4HC compressor as an option in ROOT can reduce the decompression time from 18% to 3% of the total application time, a feature to be explored, mainly in CPU-bound applications.

Table 1. Bandwidth comparison between the ZLIB compressor used in ROOT and the high-performance LZ4 compressors (running single threaded on an Intel Xeon E5620, 2.4 GHz).

Compressor	Compression		Compr.	Decompression	
	Rate	Time	Ratio	Rate	Time
LZ4	313 MB/s	1,0 x	50,7%	529 MB/s	1,2 x
LZ4 HC	20 MB/s	15,9 x	57,5%	629 MB/s	1,0 x
ZLIB N1	33 MB/s	9,4 x	57,3%	104 MB/s	6,1 x
ZLIB N6	19 MB/s	16,2 x	60,7%	119 MB/s	5,3 x
ZLIB N9	4 MB/s	87,0 x	61,4%	128 MB/s	4,9 x

Data access patterns. In a spinning-disk storage system, the data bandwidth is largely determined by three factors: i) the number and speed of the disks that compose them, ii) the mix of data access patterns used by the applications and iii) the amount of memory that is available for caching information at various levels. In a setting where there is little data reuse, such as when there are continuous reads or writes, iii) becomes less relevant and simplifies the way performance can be evaluated, leaving it dependent to the raw disks and on the patterns exerted over them.

As seen in section 3.2, ROOT files are the main sources of information, with is 12-16% random access pattern with blocks of around 9K bytes per read, which is almost equivalent to a fully random access patterns with 75K byte blocks. Fig. 3 presents the bandwidth that can be achieved with typical storage systems, where it can be seen that a block size of 75K may be unable to exploit the storage systems' capabilities.

Optimizing the data access pattern becomes essential to explore the full potential of the underlying storage systems. Exploring the ROOT files' options related to how the information is organized physically it is possible to reduce the randomness in the access patterns to increase the block size to take advantage of the raw physical performance.

The use of a system RAM-disk device for temporary storage is already being explored in some analysis applications, copying the files to RAM prior to actually using them in order to reduce the overhead of non-sequential access. However, this requires some administrative effort and can only be used where is enough memory available.

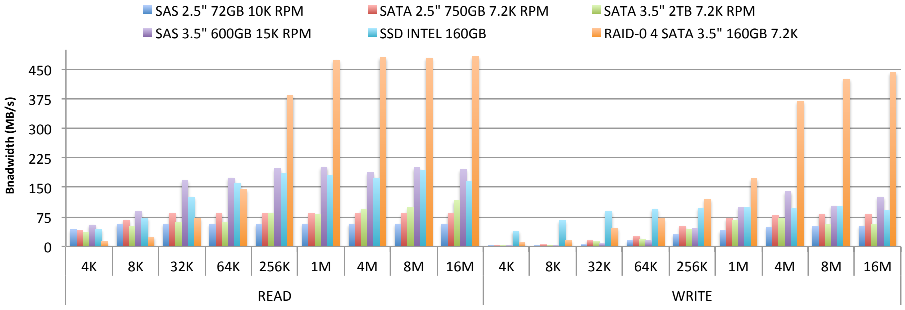

Fig. 3. Bandwidth of a disk subjected to multiple access patterns.

4 Scaling the computing and storage infrastructure

Previous section explored the software architecture aspects that can be improved in order to support more data and more processing in less time. The improvements are important, but insufficient to reach the required increase in more than an order of magnitude in processing speed and in storage space. In this section the main system parameters are evaluated in order to define the system architecture to be used as a cost effective and smooth upgrade path from currently deployed systems, to achieve further increase in computing power and in data storage capacity.

4.1 Technological evolution

Computing systems evolve rapidly, as demonstrated by Moore's law which states that computer performance follows closely the number of transistors per chip, doubling every two years [15]. In fact, the technological capacity of humankind from 1986 to 2007 has been studied in [16] and based on data from that period one may loosely estimate that an order of magnitude improvement in computing performance, in similar hardware, can be obtained five years in the future, while for networking that improvement can take more than 9 years and for storage it can take 11 years (although recently storage has been accelerating in capacity).

Amdahl defined that a balanced computing system [17] must deliver one bit of I/O bandwidth for each instruction it executes in one second. From that perspective systems are getting more unbalanced, as the availability of multi-core systems brought a tremendous increase in processing power to the compute nodes that was unaccompanied by their ability to input and output information.

4.2 The storage conundrum

For many years as the prime motivator of poor storage performance has been the spinning nature of magnetic disks – a moving parts relic in a modern system akin only to the fan blowers. The advent of solid-state disks (SSDs) led to large improvements in storage system, particularly in access latency, but due to bandwidth and capacity current spinning-disks still have some cards to play. Since bandwidth varies mostly with the rotational speed and the aerial data density and, although rotational speeds have not changed much, data density has grown tremendously and current disk drives able to sustain bandwidths between 120MB/s and 200MB/s (see Fig. 3). The bandwidth and a size and a cost, make them still a competitor to the lower latency SSDs. The main problem with spinning disks is that they are very sensitive to the data access patterns, as is demonstrated by the less than 1MB/s bandwidth of a fully random access pattern with small blocks.

Fig. 4 presents the **ttbar_dilep** application running on four different storage systems: a system RAM-disk, a local SATA disk and SAN storage system. The chart shows that with the best storage system (RAM-disk) the application takes about half of the time it takes on the SAN, the maximum for a single application, while the system call overhead shows the OS provides a stable response after a few minutes.

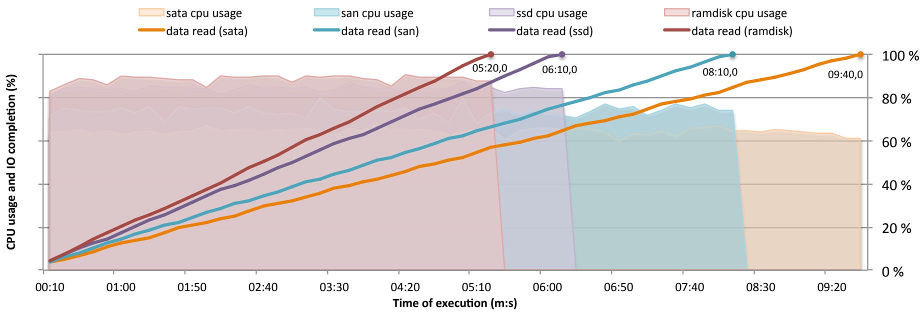

Fig. 4. Application execution time reading from SAN, SATA (3.5 2TB), SSD and RAMDISK.

Managing storage contention. In section 3.3 it was mentioned that the applications should optimize their data access pattern to better fit the underlying computing resources. However, this optimization of a single application is limited by the fact a shared storage system is subjected to many different patterns at the same time, patterns that significantly limits concurrency and that cannot be optimized by a single application without assistance from the job scheduling system. But to aggravate storage matters further, the mix of patterns delivers a random pattern even if individual patterns are sequential and, for many years, operating system and disk controllers had to go to great lengths to optimize multiple simultaneous accesses to disk in order to keep a high percentage of sequential accesses. But without proper isolation between workloads the effective bandwidths cannot match the best performance. High-performance storage systems rely on multiple layers of cache to provide lower latency but maximize transfer bandwidth, but that strategy has fewer benefits when there is a large set of continuous readers or writers with no data reuse.

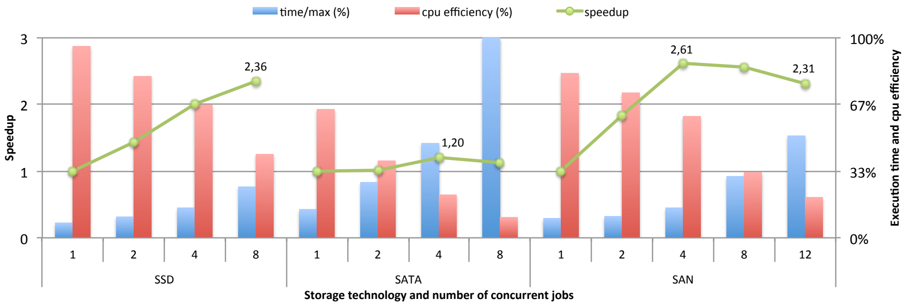

Fig. 5. Concurrent application execution time on multiple storage systems.

Fig. 5 shows the **ttbar_dilep** application running from 1 to 12 concurrent tasks on an SSD disk, on a SATA and on a SAN storage system with 15 SATA disks. The main observation is that the speedup is very small. In a single task a SATA disk has 70% the performance of a storage system with 15 individual disks, but the speedup of that system on 12 tasks is just over 2, suggesting that individual storage systems can bring a large benefit over centralized storage systems with the same number of disks.

4.3 A system architecture proposition

With centralized storage systems data must always be transferred to and from the computing nodes, demanding: i) a high performance central storage system to handle the requests from all nodes and ii) a communications infrastructure that handles the required bandwidth, both of which are potential performance bottlenecks and single points of failure. But a relevant realization from previous sections is that analysis applications process data not as typical databases, for which centralized storage may be important for data consistency, but as data streams, which present important differences in the set of requirements and optimization opportunities.

Given the restraints of the systems currently being used, the big-data requirements of application being deployed and the other challenges that are placed, it is clear that continuing to use centralized, specialized, storage systems are not the most efficient architecture to deal with the LHC upgrade, both in cost and performance. In a recent CERN report it is stated *"research should not be restricted to ROOT data structures and should fully utilise cutting edge industry technologies, such as Hadoop data processing or successors, building on existing exploration activity*" [18].

Our proposal is based on modular building blocks that can both compute and store data and on an effective middleware capable of: i) managing data distribution effectively to deal with the application behaviors and ii) placing jobs in the nodes where the data they require is located to maximize overall system performance.

In the context of our work, a simpler storage system with the same performance and capacity is inherently less expensive, considering that given enough time, data can always be rebuilt from the Tier-0. The cost of the specialized chassis, redundant disk controllers, redundant network controllers, redundant power supplies, etc., can be eliminated or be better diluted in the cost of the computing nodes. The maintenance of a specialized storage system is more complex than that of a number of simpler storage systems. While the amount of labour may be lower in the former, it is more specialized and requires specific training, so it usually adds in maintenance contracts with the storage system supplier.

Modular building block. In this proposal a computing and storage node consists of a typical computer node equipped with an inexpensive local storage system that can handle several local disks, with the number of processing cores and local disks of it building blocks largely determining the capacity of the system. Each node can be used both for computing and for storage to take advantage of data locality when jobs are placed on the nodes where the data is located, in order to: i) maximize the available disk bandwidth, ii) reduce data access latency, iii) reduce network usage and iv) facilitate the isolation of the workloads. A major benefit is the ability to partially upgrade each system component: if the system is bound by computation or lacking in storage capacity, additional nodes can be acquired or throughput additional disks can be to increase the capacity of existent or new nodes. A centralized system has more limited options in terms of upgrades and may force more disruptive transitions between systems. This gradual upgrade approach withstands budgeting restrictions while it facilitates the adoption of new technologies and the exploration of heterogeneous environments, while minimizing prolonged downtime due to complete systems replacement.

Data distribution. The balance in data distribution and the efficient access to it determines the effectiveness of this approach. Data should be distributed between the available nodes, presenting users a unified view of the file structure, replicating whenever fault tolerance or higher performance is required. The Google File System (GFS) and its open source lookalike Hadoop File System (HDFS) have proven their effectiveness in stream-based applications, but get fully explored only when using the associated Map and Reduce computing model [19]. But solutions such as Glusterfs, or even a simpler static distribution and replication of files, may provide the benefits of effective data distribution without any major application changes.

Code placement. To complement data placement a scheduling strategy that allows to places applications near its data may be used as a valuable optimization. Computing models such as Map and Reduce could be valuable, but interoperability with the ATLAS collaboration precludes it. One alternative approach is to improve the local job scheduler with storage related restrictions; a limit on concurrent accesses of each device can be set so that the storage system performance is optimized. Another approach is to develop a fully parallel version of the base application library that is aware of the data placement and distributes work accordingly. It is possible to extend PROOF [19] in that direction, given that at present only a single source of information (the master) is considered. Virtualization is also an alternative approach, one that does not depend on prior knowledge of application structure or on application redesign. It could be used as mechanism to place applications near the resources they effectively require [21, 22] by migrating Virtual Machine to the nodes where data is being sourced from and move on to another when the source changes.

5. Conclusion

This paper presents software and infrastructure optimization opportunities to explore in order to better support the LHC/ATLAS upgrade and the ability to fully explore the physics potential of the analysis programs developed by the team. Further increase in computing power and in data storage capacity can be achieve upgrading the local system following a system architecture proposition based on modular building blocks that can both compute and store data. These can perform better than centralized systems of similar capacity, provide a smooth upgrade from current systems and address the essential scalability, resource efficiency and cost concerns.

This work also highlights the importance of complete execution profiles, both in terms of CPU and IO, that a tool such as **tomograph** can provide. The integrated view of the application CPU and memory usages (fig. 1), a clustered view of the call-stack functions (fig. 2), and the usage of the underlying system and I/O subsystems (fig 5) proved to be a valuable benefit that needs to be further explored.

Acknowledgments. The research presented was partially funded by FCT grants SFRH/BPD/63495/2009 and SFRH/BPD/47928/2008, by the UT Austin | Portugal FCT grant SFRH/BD/47840/2008, and by FCT project PEst-OE/EEI/UI0752/2011.

References

1. “Challenges and Opportunities”, Special Online Collection: Dealing with Data, Science 11 February 2011: Vol. 331 no. 6018 pp. 692-693.
2. Lohr, S., “The Age of Big Data”, The New York Times, Feb 11, 2012, page SR1, online: http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html.
3. Tatevossian, A. R., "Big Data, Big Impact: New Possibilities for International Development", Annual Meeting in Davos, the World Economic Forum, Jan 25, 2012.
4. ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider", JINST no. 3 S08003, 2008, doi:10.1088/1748-0221/3/08/S08003
5. ATLAS Collaboration, "Measurement of W boson polarization in top quark decays with the ATLAS detector", JHEP 1206 088, 2012, doi: 10.1007/JHEP06(2012)088.
6. The ATLAS Collaboration, "A search for flavour changing neutral currents in top-quark decays in pp collision data collected with the ATLAS detector at sqrt(s) = 7 TeV", arXiv:1206.0257 [hep-ex] (submitted to JHEP), Jun. 2012
7. ATLAS Collaboration, "Search for exotic same-sign dilepton signatures (b' quark, T5/3 and four top quarks production) in 4.7 fb^{-1} of pp collisions at $\sqrt{s}=7$~TeV with the ATLAS detector", CERN, ATLAS note ATLAS-CONF-2012-130, Sep. 2012.
8. ATLAS Collaboration, "Search for the Standard Model Higgs boson produced in association with top quarks in proton-proton collisions at sqrt(s)=7 TeV using the ATLAS detector", CERN, ATLAS note ATLAS-CONF-2012-135, Sep. 2012.
9. ATLAS Collaboration, "Search for pair production of heavy top-like quarks decaying to a high-p_{T} W boson and a b quark in the lepton plus jets final state at sqrt(s)=7 TeV with the ATLAS detector", CERN, ATLAS paper EXOT-2012-07, Sep. 2012.
10. Jones R and Barberis D, J. Phys.: Conf. Ser. 119 072020 doi:10.1088/1742-6596/119/7/072020
11. R. Brun and F. Rademakers, “ROOT - An Object Oriented Data Analysis Framework”, Nucl. Instrum. Meth. A 389 (1997) 81. See also http://root.cern.ch/.
12. Susan L. Graham, Peter B. Kessler, and Marshall K. Mckusick. gprof: a Call Graph Execution Profiler, Proc. SIGPLAN '82 Symposium on Compiler Construction, SIGPLAN Notices, Vol. 17, No 6, pp. 120-126; doi: 10.1145/800230.806987.
13. Intel VTune, http://software.intel.com/en-us/intel-vtune-amplifier-xe, retrieved Aug. 2012.
14. Collet, Y., LZ4, http://code.google.com/p/lz4/, retrieved Aug. 2012.
15. Moore, G.E., “Moore's law measures technological progress…”, Proc. of SPIE, 2-17 (1995).
16. Hilbert, M, Lopez, P, “The World’s Technological Capacity to Store, Communicate, and Compute Information”, Science 332, 60-65 (2011).
17. Gordon Bell, Jim Gray, Alex Szalay, "Petascale Computational Systems", Computer, vol. 39, no. 1, pp. 110-112, Jan. 2006, doi:10.1109/MC.2006.29
18. Barberis, D., Dykstra, D., et al., Report of the WLCG Database Technical Evolution Group, WLCG Document Repository, March 2012.
19. Ghemawat, S., Gobioff, H., and Leung, ST,. The Google file system. In 19th Symposium on Operating Systems Principles, pages 29–43, Lake George, New York, 2003.
20. M Ballintijn, G Roland, R Brun, F Rademakers. “The PROOF Distributed Parallel Analysis Framework based on ROOT”, Proc. CHEP 2003 Conference, La Jolla, California.
21. Oliveira, V., Pina, A., Rocha, A., 2012, “Running User-provided Virtual Machines in Batch-oriented Computing Clusters”, 20th Euromicro Int. Conf. on Parallel, Distributed and Network-based Computing (PDP’12), Garching, Germany.
22. Oliveira, V., Pina, A., Sá, T., 2012, “Simple Peer Messaging for Remote User Domains Interconnection”, The 2012 Int. Conf. on High Perf. Comp. & Sim. (HPCS), Madrid, Spain.
23. The ATLAS Collaboration, “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC“ Phys. Lett. B 716 (2012) 1-29.