BigStorage

STORAGE-BASED CONVERGENCE BETWEEN HPC AND CLOUD TO HANDLE BIG DATA

Deliverable number	D4.2
Deliverable title	Final Report M48
	WP4 Storage Solutions
Editor	Angelos Bilas (FORTH)
Main Authors	Alvaro Brandon (UPM), Rizkallah Touma (BSC), Fotis Nikolaidis (CEA), M. Umar Hameed (Mainz), Georgios Koloventzos (BSC), and Michał Zasadziński (CA)

Grant Agreement number	642963
Project ref. no	MSCA-ITN-2014-ETN-642963
Project acronym	BigStorage
Project full name	BigStorage: Storage-based convergence between HPC and Cloud to handle Big Data
Starting date (dur.)	1/1/2015 (48 months)
Ending date	31/12/2018
Project website	http://www.bigstorage-project.eu

Coordinator	María S. Pérez
Address	Campus de Montegancedo sn. 28660 Boadilla del Monte, Madrid, Spain
Reply to	mperez@fi.upm.es
Phone	34- 910672857

Executive Summary

This document provides an overview of the progress of the work done until M24 of the Project BigStorage (from 01-01-2015 until 31-12-2016) in WP4 Storage Solutions.

The tasks in WP4 cover problems related to device technology and how these ones affect storage architecture and the storage stack. WP4 categorizes issues and tasks as:

T4.1 Storage acceleration: How modern storage systems can take advantage of heterogeneity and locality to improve performance of applications.

T4.2 Storage convergence: How storage systems can support different types of applications, especially where big data is involved, converging different uses of storage over the same platforms.

T4.3 Storage isolation: How we can infer system behavior in mixed environments, where there is heterogeneity in terms of storage technologies but also the applications that use storage.

The work carried out by ESRs and in relation to WP4 covers these topics as follows:

- Prefetching mechanisms and polices that have the potential to deal with heterogeneity.
- File systems for heterogeneous storage systems.
- Support for big data – big storage applications.
- Cloud blobs as a basis for supporting diverse applications.
- Multi-criteria optimization for dealing with performance issues.
- Monitoring and inference for understanding the impact of co-location.
- Root cause analysis for detecting causality relationships in complex environments.

This report presents an overview of each of these topics, including a brief analysis of the state of the art in each case and a preliminary plan of the specific problems to be addressed by each ESR.

Document Information

IST Project Number	MSCA-ITN-2014-ETN-642963
Acronym	BigStorage
Title	Storage-based convergence between HPC and Cloud to handle Big Data
Project URL	http://www.bigstorage-project.eu
Deliverable	D4.2 Final Report on WP4
Workpackage	WP4 Storage Solutions
Date of Delivery	Planned: 31.12.2016 Actual: 20.12.2016
Status	Version 1.0 final ■ draft □
Nature	prototype □ report ■ dissemination □
Dissemination level	public □ consortium ■
Distribution List	Consortium Partners
Responsible Editor	Angelos Bilas (FORTH), bilas@ics.forth.gr, Tel: +302810391669
Authors (Partner)	Alvaro Brandon (UPM), Rizkallah Touma (BSC), Fotis Nikolaidis (CEA), M. Umar Hameed (Mainz), Georgios Koloventzos (BSC), and Michał Zasadziński (CA)
Reviewers	BigStorage Advisors
Abstract (for dissemination)	Executive Summary
Keywords	Storage acceleration, storage convergence, storage isolation, heterogeneity, Flash, NVM, monitoring, inference

Version	Modification(s)	Date	Author(s)
0.1	Initial template and structure	20.07.2018	Angelos Bilas, FORTH
0.2	Sections from all authors	30.11.2018	All authors
0.3	Internal version for review	17.12.2018	Angelos Bilas, FORTH
0.4	Comments from internal reviewers	21.12.2018	Internal reviewers
0.5	Comments from internal reviewers	26.12.2018	Internal reviewers
0.6	Comments from internal reviewers	28.12.2018	Internal reviewers
0.7	Final version to commission	31.12.2018	Angelos Bilas, FORTH

Project Consortium Information

Participants		Contact
Universidad Politécnica de Madrid (UPM), Spain		María S. Pérez Email: mperez@fi.upm.es
Barcelona Supercomputing Center (BSC), Spain		Toni Cortes Email: toni.cortes@bsc.es
Johannes Gutenberg University (JGU) Mainz, Germany		André Brinkmann Email: brinkman@uni-mainz.de
Inria, France		Gabriel Antoniu Email: gabriel.antoniu@inria.fr Adrian Lebre Email: adrien.lebre@inria.fr
Foundation for Research and Technology - Hellas (FORTH), Greece		Angelos Bilas Email: bilas@ics.forth.gr
Seagate, UK		Sai Narasimhamurthy Email:sai.narasimhamurthy@seagate.com
DKRZ, Germany		Thomas Ludwig Email: ludwig@dkrz.de
CA Technologies Development Spain (CA), Spain		Victor Muntes Email: Victor.Muntes@ca.com
CEA, France		Jacque Charles Lafoucriere Email: Charles.LAFOUCRIERE@CEA.FR
Fujitsu Technology Solutions GMBH, Germany		Sepp Stieger Email: sepp.stieger@ts.fujitsu.com

Table of Contents

1 Introduction

1.1 *WP4 Overview*

This subsection discusses an overview of WP4 as was presented in the proposal.

Storage systems are undergoing fundamental changes to keep up with the requirements of new applications and services. At the infrastructure level the problems we face today can be categorized in the areas storage acceleration, storage convergence, and storage isolation:

T4.1 Storage acceleration received much attention, largely due to the advent of Flash-based storage device technologies. Future disruptive change to the I/O hierarchy will be from the use of emerging, byte addressable Non-volatile Memories (NVM), which are candidates for bridging the gap between non-persistent byte-addressable DRAM and persistent, block-addressable storage.

T4.2 Storage convergence refers to bringing storage and computation closer both in the data centre and HPC. Traditionally, persistent storage has been placed behind a storage area network (SAN) for scaling and management purposes. With current technology trends, it becomes important that storage (and NVM) moves closer to the compute nodes to further reduce energy footprint. This is a significant architectural shift and requires fundamentally different approaches to storing, caching, replicating, and moving data.

T4.3 Storage isolation is emerging as a central problem for storage infrastructures in heterogeneous and multi-tenant environments. Access to storage (and infrastructure in general) leads to contention that is induced even when independent applications access different parts of the data on the same storage devices, resulting in a loss of efficiency at the device and I/O path level. As a consequence, providers prefer to reduce shared access to storage resources by over-provisioning, leading to increased costs. This problem is particularly acute in the era of Big Data and Cloud Storage where data access is at the heart of applications and services.

WP4 will examine Big Data issues associated with deep storage hierarchies, how storage can be architected closer to computation, and how we can ensure application isolation over shared infrastructures.

1.2 *ESR Participation and contributions*

The full list of ESRs in the current form of the project is:

ESR	Name	Institution	Main advisor
1	Ovidiu Marcu	INRIA	Gabriel Antoniu/Alexandru Costan
2	*Alvaro Brandon*	*UPM*	*Maria S. Perez*
3	Pierre Matri	UPM	Maria S. Perez
4	*Nafiseh Moti*	*Mainz*	*Andre Brinkmann*
5	*Rizkallah Touma*	*BSC*	*Anna Queralt/Toni Cortes*
6	Fotios Papaodyssefs	Seagate	Malcolm Muggeridge

7	Linh Thuy Nguyen	INRIA	Adrien Lebre
8	*Athanasios Kiatipis*	*Fujitsu*	*Sepp Stieger*
9	*Fotis Nikolaidis*	*CEA*	*Philippe Deniel*
10	*Dimitrios Ganosis*	*FORTH*	*Angelos Bilas/Manolis Marazakis*
11	Muhammad Umar Hameed	Mainz	Andre Brinkmann
12	*Georgios Koloventzos*	*BSC*	*Ramon Nou/Toni Cortes*
13	Mohammed-Yacine Taleb	INRIA	Shadi Ibrahim
14	Yevhen Alforov	DKRZ	Thomas Ludwig / Michael Kuhn
15	*Michał Zasadziński*	*CA*	*Victor Muntes*

The ESRs that continued to contribute to WP4, in the second half of of the project, based on how their work evolved and the respective tasks are:

- ESR9: T4.2 T4.3
- ESR11: T4.1 T4.2 T4.3
- ESR12: T4.1 T4.2 T4.3

In addition, the work of certain other ESRs has overlap with WP4 so we include that work in the report as well. These ESRs are: ESR2, ESR5, and ESR15. At the current stage of work the research by individual ESRs contributes to each task as follows:

T4.1 Storage acceleration: This work shows how modern storage systems can take advantage of heterogeneity and locality to improve performance of applications.

- ESR5: Prefetching
- ESR11: Garbage collection in NAND-Flash SSDs
- ESR12: Heterogeneous file systems

T4.2 Storage convergence: This work shows how storage systems can support different types of applications, especially where big data is involved, converging different uses of storage over the same platforms.

- ESR9: Cloud blobs

T4.3 Storage isolation: This work shows how we can infer system behavior in mixed environments, where there is heterogeneity in terms of storage technologies but also the applications that use storage.

- ESR2: Multi-criteria optimization
- ESR15: Root cause analysis

Next, we discuss each of these categories and the associated ESR work in mode detail.

2 Storage acceleration (T4.1)

2.1 *ESR5: Prefetching and Data Placement via Static Code Analysis*

Contributor: *Rizkallah Touma, BSC*

Overview

Data prefetching aims to improve access times to data storage systems by predicting data records that are likely to be accessed by subsequent requests and retrieving them into a memory cache before they are needed. In the case of Persistent Object Stores, previous approaches to prefetching have been based on predictions made through analysis of the store's schema, which generates rigid predictions, or monitoring of access to the store while applications are executed, which introduces overhead.

Persistent Object Stores (POSs) are data storage systems that record and retrieve persistent data in the form of complete objects (Brown, 1992). They are especially used with Object-Oriented programming languages to avoid the impedance mismatch that occurs when developing OO applications on top of other types of databases, such as Relational Database Management Systems (RDBMSs). This is because POSs make it easier to access persistent data without worrying about database access and query details, which can amount to 30% of the total code of an application (Chen T.-H. a., 2014).

Examples of POSs include object-oriented databases (e.g. Caché and Actian NoSQL) and Object-Relational Mapping (ORM) systems (e.g. Hibernate, Apache OpenJPA and DataNucleus). The rise of NoSQL databases has also led to the development of mapping systems for non-relational databases such as Neo4J's Object-Graph Mapping (OGM). Moreover, several POSs that support data distribution have been developed to accommodate the needs of parallel and distributed programming (e.g. Mneme, Nexus, Thor and dataClay).

Previous approaches to prefetching in POSs can be split into three broad categories: (1) schema-based, (2) data-based, and, (3) code-based. An example of a schema-based approach is the *Referenced-Objects Predictor (ROP)*, which uses the following heuristic: each time an object is accessed, all the objects referenced from it are likely to be accessed as well. While this type of approaches gives rigid predictions that do not take into account how the persistent objects are accessed by different applications, the ROP is widely used in commercial POSs because it does not involve a costly prediction process.

On the other hand, data-based approaches predict which objects to prefetch by detecting data access patterns while monitoring application execution, which causes overhead that can amount to roughly 10% of the application execution. Finally, few approaches have based the predictions on analyzing the source code of the OO applications that access the POS, and these have been largely theoretical without any in-depth analysis of the prediction accuracy or the performance improvement that they can achieve.

Our work presents an approach to predict access to persistent objects through static code analysis of object-oriented applications. The approach includes a complex inter-procedural analysis and takes nondeterministic program behavior into consideration. We evaluated the prediction accuracy of the approach and concluded that it can be used to apply various optimization techniques, including data prefetching.

We have also implemented *CAPre*: a prefetching system that uses this prediction approach to prefetch objects from a POS. *CAPre* performs the prediction at compile-time without adding any overhead to application execution time. It then uses source code generation and injection to modify the application's original code to activate automatic prefetching of the predicted objects when the application is executed. *CAPre* also includes a further optimization by automatically prefetching data in parallel whenever possible, in order to maximize the benefits obtained from prefetching when using distributed POSs.

We integrated *CAPre* into a distributed POS called and ran a series of experiments to measure the improvement in application performance that it can achieve. The experimental results indicate that, apart from a limited number of exceptions, using *CAPre* to prefetch objects from a POS can indeed reduce execution times of applications, with the most significant gains observed in applications with complex data models and / or many collections of persistent objects.

Results

We tested the effect that *CAPre* has on application performance by calculating the execution times of four benchmarks using *dataClay* without any prefetching, and with both the sequential and parallel versions of *CAPre*. We also compared *CAPre* with the *Referenced-Objects Predictor (ROP)* using different fetch depths, which indicate the levels of related objects that the ROP should prefetch. The benchmarks we chose were OO7, the standard benchmark used to evaluate POSs and object-oriented databases, and Wordcount, K-Means and the Princeton Graph Algorithms, which are commonly used as Big Data benchmarks.

The results obtained from our experiments indicate that *CAPre* offers the highest improvement in execution time when used with applications with a complex data model, such as OO7. Moreover, prefetching data in parallel proved extremely beneficial, especially with simple data models that contain many collection associations, such as the case with the Wordcount and K-Means benchmarks. We also encountered one limitation of *CAPre*, with the Bellman-Ford shortest path algorithm, where it could not offer significant improvement because the algorithm accesses persistent objects in a random order that is difficult to predict.

In terms of data size, the experiments indicate that *CAPre* provides the same level of improvement regardless of the number or size of persistent objects manipulated by each benchmark. When compared with the ROP, *CAPre* achieves at least the same improvement and, in cases where prefetching is not needed, has less of a negative effect on application performance. When considering the parallel version, *CAPre* always reduces the execution times of the benchmarks by a higher percentage than the ROP.

2.2 *ESR11: Coordinated Garbage Collection to improve the RAIN model*

Contributor: M.Umar Hameed

Overview

NAND flash-based devices for example Solid State Drives (SSDs) has gained much attention in storage technology. Flash is being in use from small hand-held devices to big vendor service equipments. The need of high performance is always a consumer demand and service provider is responsible to provide according to Service Level Agreement (SLA). Despite a lot of improvement, it is difficult to achieve those requirements. Flash has improved its performance in terms of latency

and throughput, but still it has some serious performance drawback. There are three major operations of Flash; Read, Write on the granularity of page and Erase operation on the granularity of block. Erase is the most cost-effective operation because it executes on the level of block; includes reading, copying and writing the valid pages from one block to another and in the end cleaning the previous block. Read and Write operations are external as they initiate from users, whereas erase operation is an internal operation i.e it invokes to claim free space within the flash package.

Error-Correction codes(ECC) have been a traditional approach to overcome the fading nature of flash cells. The redundant array of independent Nand (RAIN) approach is combined to enable the reliability as well as to reduce the tail latency. RAIN provides an effective approach in comparison to high bit error rates by the traditional ECC approach. Due to its high failure rate, SSD vendors are shifting towards parity based scheme. Another advantage of this approach is that it aids in reducing the long tail latencies caused by time consuming garbage collection(GC). But the core problem of ensuring a redundant SSD model remains at its place when observed at a fine grain level. For example, in a stripe when a block is being GCed which removes a part of the stripe, leaving the other stripe contributors invaluable for the aim of redundancy. One way is to recalculate the parity for the residual stripe contributors which increases the internal writes.

The purpose of the RAIN is to provide the data integrity within the flash in order to ensure reliability by adding some extra parities. Error Correction Code (ECC) have been used for the intention of data protection in flash, but further RAIN facilitates extra features in addition to the solution of primary problem. For example, using RAIN the parities are distributed across the flash to further provide reliability on component level i.e in case of channel failure. The RAIN architecture is also helpful in improving read performance when a GC is in process in a particular die.

Although the RAIN model have been helpful in particular cases, but the RAIN model suffers from high internal writes due to out of place update in flash. For example, a single page update also requires the parity page to be updated, hence increasing the write amplification. Consider three data pages D0, D1 and D2 from three different channels having the same offset in the block, creates a parity P0 in separate channel within the flash at time period T 0. The user write in this case is three and a single write makes the write amplification of 1.33. Meanwhile at time T 10, a data page D0 is updated and the previous one is marked as invalidated as well as its corresponding parity page P0 is also invalidated. A new parity page P0' within the same channel created at T 11 with the updated data page D0' with D11' and D21' data pages. Therefore, a single update in a data page increases the write amplification from 1.33 to 2, which leads to early wear-out of the device. In addition to above problem, this traditional scheme still does not provide redundancy to two of the data pages D1 and D2 as their two counterparts are invalidated which further leads to removal after garbage collection.

Another problem arises when a potential blocks from different dies is garbage collected and disturbs the stripe like formation. The RAIN approach among the SSD dies create stripes to improve the read performance and reliability. Each single page in a die can be recovered by the pages with the same offset present in rest of the available dies. When a garbage collection is initiated, the data block is moved to another location to be a part of new stripe and hence not be a part of previous stripe.

We proposed a coordinated GC (CGC) approach within the flash dies, which ensures a redundant RAIN model. In our CGC, the parity plane aids to select a potential block for the GC of its corresponding data blocks. A stripe is selected for the GC via parity plane, making the whole stripe data to be moved to new location at different time intervals. Furthermore, CGC also don't need to trace the parity pages in a mapping table, as each parity page is an offset of their corresponding data pages.

Summary

We measured the Write Amplification(WA) for the RAIN model and compared it with the traditional flash model without the RAIN layout. In the results, we observed that initially, the WA of the RAIN model which includes the parity writes have almost a WA of 2, which means the GC in parity plane starts earlier than the data plane. Whereas, without RAIN, WA starts with almost '1' as it is the first GC within the data plane.

Moreover, the difference of WA between the following two models, remain distant throughout the simulation. But at the end, the both lines tend to converge and achieves respective WA of almost 10 for RAIN model and almost 9 without RAIN model.

We have also tested our approach with various trace files and analyzed the WA after running the trace files. The WA with our approach is reduced by a factor of 0.7 for uniform random writes and 0.5 with TPCC trace. We also concluded that the skewed access pattern of the trace with overwrite operations are more favorable with our approach.

2.3 *ESR12: Heterogeneous File Systems*

Contributor: *Georgios Koloventzos, BSC*

Overview

Storage devices have been diversified in recent years. From traditional hard disk drives (HDDs) to solid state drives (SSDs) and newly Non-Volatile Memory (NVM) drives, a plethora of different devices has emerged each one having different attributes to favor. Storage media have three major characteristics: speed, capacity and price. Trying to optimize them is nearly an impossible task for both individuals and companies. New media are giving a tremendous boost in speed but lack in both price and capacities. A popular solution and an enormous amount of research have set focus on creating a storage stack of the devices to incorporate the faster and smaller in capacity storage devices as caching tiers for HDD-based storage systems.

While trying to optimize storage, a tier infrastructure was created. Bulk and slow media are being placed at the bottom with each higher layer having a faster and smaller-capacity medium, due to expensive components. Such infrastructure introduced the cache to the storage tier, a technique that had proven its usefulness on the CPU memory tier. Cache architecture as it is application agnostic is an easy and simple solution. SSDs were a major breakthrough regarding their response time. The difference in some characteristic with HDDs are enormous, HDDs cannot compare with the random request response time SSDs have introduced. Thus, the usage of SSDs had focused on cache or as a main storage tier for crucial applications. Nowadays, NVMs are getting the same treatment. They are introduced as either a bridge between the RAM and the main storage tier (either SSDs or HDDs) or as main storage devices with a huge advantage on response time. Companies in storage pools use this architecture in order to offer solutions to all price ranges. This blind architecture does not take into consideration any drawbacks of the underlining storage media. The most crucial drawback a storage system has, is the wearing out of the SSDs. HDDs may be slow but their response time does not degrade as much as of a worn out SSD. Because this drawback is rooted at the physical layer of the medium, it is really difficult to overcome.

Files have also evolved over the years. Data structures that help programmers to seek or better visualize the file are also stored inside the file. The notion of the simple file as a stream is far from what modern applications are using (Harter T. et al, 2011). Files can be compartmentalized in various regions. Metadata, metablocks, tables, bitmaps and the actual data are some of the regions, a file can have. Files may contain more than one of these data structures. Such regions are chosen arbitrarily to fit the needs of each application and the developer. Even though a tendency is emerging for the files containing data for Big Data applications to specify a protocol of self-describing stream files. In addition, programmers now use files in a different way. As some file functionality is considered as atomic operations, developers are using them to guarantee the consistency of the data and the reliability of their application. Using such mechanisms, which imply writing to disk, is not ideal for storage media especially for SSDs.

CADMUS identifies regions that are accessed more and stores them permanently in a faster media. It tries to break the storage nedia stack and split the file in important regions, which will be permanently stored at SSDs and rest of a file at a slower medium. Such approach will not have the same response time as if a cache was used or with an SSD as main storage, but it will prolong the lifespan of the SSD with a non-negligible amount of speedup for the majority of the underlying workload.

Results

We tested our hypothesis using 4 workloads in 3 different settings. We first placed all data only to an SSD to measure the best latency the workshop can have. Then all data were placed on HDD to measure the worst-case scenario. Then the top 20% of the most accessed blocks were transferred to SSD and run again the experiments. For workloads we picked a relational database (MySQL), a non-relational database (RocksDB), the boot sequence of a Virtual Machine (VM) with 2 flavors of disk images (raw and qcow2) and last the Ferret application that manipulates and depicts weather data.

MySQL is one of the frequently used relational database management system (RDBMS). From small sites to big industries, MySQL is the first choice for a database engine. Sysbench workload was selected to run the experiments with a big database of 200000 rows. The internal MySQL buffer was capped at 5MB to force more I/O to the underlying media. The rest of MySQL variables are left, as is which means that our environment has full ACID compliance. A variety of numbers of threads were used to emulate a range of usage patterns. For each run (not in the prepare section) Sysbench accesses 5 to 7 files in MySQL including the logs. CADMUS analyzed the patterns in 3 files, ibtmp, ibdata and ibd. The two latter files are structured as pages. Each block in ZFS is two pages. With this in mind, a clear pattern of accessing specific blocks in the file emerged. The first block of the file (header) was accessed a lot, as was also the area of where the first double write buffer is located. CADMUS changed 22 blocks from ibdata file, out of 109 blocks accessed and 608 total. Also 92 blocks changed from the database file, out of 374 were accessed and 442 in total. The last 3 changed blocks came from a temporary file (92 total blocks, 5 accessed). A speedup of 50% to 60% was achieved in comparison with full HDD. CADMUS has stored only 13% of the accessed blocks in SSD. On the contrary, a 70% to 130% slowdown than full SSD was measured. If a measurement is performed on the total block count, then this result is achieved with just 6% of all blocks from all files. CADMUS lowers the LBAs written 54% on average.

The booting time of a virtual machine is crucial as many companies are still using them for serving material. As such machines are spawned and destroyed a lot, shorter booting time may lead to better response times for major vendors. The boot results were taken with the systemd-analyze command. For storage, two options were measured: qcow2 and raw. Both images were of 10GB was created with raw and qcow2 file format. The actual size of qcow2 was 2.2GB as the image is not instantaneously allocating the 10GB Even though raw images are not a standard, it is assumed that such experiment can give a better understanding of underlying file formats and their implications when the same experiment is carried out.

The raw image as it has already allocated the 10GB has better percentages but nonetheless the 47 blocks CADMUS changed was just the % of the accessed blocks and 0.05% of the whole image. The impact of these blocks was a 35% boost in boot time. Raw image requires no additional work from hosts. When a virtual machine writes data to a given offset in its virtual disk, the I/O is written to the same offset on the backing file.

For the qcow2 experiment, CADMUS changed 41 blocks, which is just 0.2% of the whole image and 4.3% of the accessed blocks. Despite the small percentage, a 28% boost was achieved. The respective slowdown is 96% from when the entire file was stored in SSD. As qcow2 format has many table layouts (L1 and L2) these areas are accessed more in the read dominant procedure of booting a virtual machine. These are most of the blocks that CADMUS changed. Even qcow2 image has a smaller imprint in the storage media and now is used as a default for most of the virtual machines; it is observed that in terms of bootup does not have a huge difference with raw images. Both need a tiny part of the both images for booting (946 blocks for qcow2 and 927 for raw), even with less than 50 blocks at a faster medium can have a considerable speedup at boot time. CADMUS lowers the LBAs written 85% on average.

RocksDB was chosen to evaluate if CADMUS can boost key-value applications. RocksDB creates mainly two types of files, sst and log files. SST files containing all the data and tree structure for the RocksDB. RocksDB only analyzes this kind of files because the log files are heavily written. For testing the YCSB suite was used. YCSB has native support for RocksDB. YCSB is organized in 6 experiments, emulating common workloads that can emerge in cloud services. As YCSB runs are mostly deterministic, according to how many records are inserted, the number of sst files (the files that data are stored) are mostly the same. An average number with standard deviation of files and blocks is used, because the number is altered between the workloads. The files are getting merged as a modification of key-value pairs happens or by new insertions. As load workloads are similar and are write intensive, they are not taken into consideration. To create a big enough database, 1000000 records were used and 10000 operations were carried out, to keep the total runtime of the experiments low. In these experiments, the average number of files is 10 (standard deviation of 1) with an average number of 10000 blocks (standard deviation of 550). Mostly all of them are accessed in each workload. As the workloads are going through the data a lot, most of the accesses are concentrated at the first 250 blocks, where the data resides. There are files that are only accessed at the last blocks where the tree structure of this part of the file is stored, but because this file does not contain any pair that the workload needs, it is not accessed more. Thus CADMUS is instructed to store the first 250 blocks in SSD. Having the mean of 10 files with 10000 blocks, an amount of approximately 2500 blocks or 25% of the files are stored in SSD. This gives a speedup of 53% to 88% depending on the workload. CADMUS lowers the LBAs written 46% on average helping the endurance of the SSD.

Ferret is an interactive computer visualization and analysis environment designed to meet the needs of oceanographers and meteorologists analyzing large and complex gridded data sets. For our experiments, we used PyFerret which is a Python module wrapping Ferret. In our first try, we only change the place of blocks of the two data files ferret is accessing, the data of the two variables and the data are needed for drawing the globe in the output. For these runs, we are getting a 5% better result, with just 1.5% of the files or the 3.1% of the accessed blocks. As the installation procedure was all done in the ZFS partition, additional data from accessing the python ferret library were available. Thus in the second experiment of PyFerret the library file was also split into the storage media. CADMUS found that some particular regions are accessed more, every time this workload was executed. For the library itself, CADMUS decided to migrate 238 blocks. The number of the accessed blocks was 298 from 708 of the whole library. For this run, a 14% boost with as low as 2.4% of the total blocks of all the files was observed and 5.1% of the accessed blocks (the low percentage of the accessed blocks is due to the big data file).

3 Storage Convergence (T4.2)

3.1 *ESR9: Storage blobs hosted in the cloud*

Contributor: *Fotis Nikolaidis, CEA*

Overview

The demand for performance, scalability, and workload diversity of modern applications has led to the development of a broad spectrum of storage platforms. Nowadays the storage inventory is highly diversified with parallel filesystems for concurrent access, scalable object storage for capacity and scalability and databases optimized for use in memory, hard-disks, or non-volatile memories

These data-centric platforms assume that data is the primary and permanent asset, and applications come and go. In the data-centric architecture, the data model precedes the implementation of any given application and will be around and valid long after it is gone. Many people may think this is what happens now or what should happen. However, this is very rarely the case. Businesses want functionality, and they purchase or build application systems. Scientists wish to simulate physical phenomena, which may be computationally intensive or I/O intense. Each application system has its data model and inextricably tied code with it.

In our study, we introduced "application-tailored storage systems" as a distributed storage middleware with programmable behavior. The middleware separates I/O logic from the rest of the application logic, thereby separating changes made to application codes by science users from changes made to I/O actions by developers or administrators. Such design helps to defer I/O decisions until the deployment phase, which is the key for portability, workload tuning, and data post-processing transparently to the application. I/O logic design is external to the application, thus allows cleaner code while the developers remain in control. For the middleware development, we introduced Tromos SDK (Software Development Kit): a set of highly compartmentalized modules that provide the building blocks for building customized distributed storage systems. Using its domain-specific language, developers can model and deploy systems with customized distribution logic, consistency, selection algorithms, data layout. Hence, building customized storage environments for the application is only a matter of choosing the appropriate schema.

Results

The evaluation of our SDK emphasizes the component interoperability and feasibility to realize customized storage systems without systems programming. More specifically, we set our goals to compare with GlusterFS ; a well-established storage system powered by RedHat. In GlusterFS terms, a volume is a logical collection of bricks where each brick is an export directory on a server in the storage pool. The GlusterFS volumes for which we set our objectives are:

1) Distributed: distributes files across bricks in the volume,
2) Replicated: replicates files across bricks in the volume,
3) Striped: stripes data across bricks in the volume.
4) Dispersed: bases on erasure codes to providing space-efficient protection against disk or server failures.

Those objectives aim to highlight differences both on control and data path. The first objective aims to compare the Distributed Hash Table (DHT) lookup of GlusterFS with the Decentralized Catalogs of Tromos. The second objective is to compare datapaths with parallel independent streams; the client can read from any of the replicated locations. The third object is to compare datapaths with parallel dependent streams; the must retrieve data from both location in order to reconstruct the original data. The fourth object is to provide resiliency against failures but still retain parallelization. More specifically, Reed-Solomon is an erasure coding algorithm for encoding and splitting that into K+P streams, out of which only K (any K) suffice to reconstruct the original data. The drawback is that processing is CPU-bound.

In general, our conclusions from the experiments summarize to:

1) Without intermediate processing, DHT outperforms Catalog lookup
2) Parallel streams amortize the lookup overhead, and the two methods converge fast as the number of parallel streams increases
3) Tromos performs three times better than GlusterFS on high load – concurrently writing files with multiple parallel streams per file. That holds both for lightweight (e.g., mirror, strip) and CPU-bounded (e.g., erasure coding) in-transit processing.
4) The fuse implementation in which Tromos based enforces serialization on the requests, thus eliminating the gains of large I/O depth. In turn, that also diminishes the gains of data prefetching – although data reside in-memory they are not usable until the previous request is complete.

Briefly, the experience with our prototype proved the viability of the concept and showed that there is fertile ground for future work in storage programmability.

4 Storage Isolation (T4.3)

4.1 *ESR2: Multi-criteria Decision Support Systems for Big Data Analytics*

Contributor: *Alvaro Brandon, UPM*

Overview

Planning big data processes effectively on cloud and HPC platforms can become problematic. They involve complex ecosystems where developers need to discover the main causes of performance degradation in terms of time, cost or energy. However, processing collected logs and metrics can be a tedious and difficult task. In addition, there are several parameters that can be adjusted and have an important impact on application performance. One of the most important challenges is finding the best parallelization strategy for a particular application running on a parallel computing framework. Big data platforms like Spark or Flink use JVM's distributed along the cluster to perform computations. Parallelization policies can be controlled programmatically through APIs or by tuning parameters. These policies are normally left as their default values by the users. However, they control the number of tasks running concurrently on each machine, which constitutes the foundation of big data platforms. This factor can affect both correct execution and application execution time. Nonetheless, we lack concrete methods to optimize the parallelism of an application before its execution. This is specially challenging considering that the concurrent access to disk by these parallel tasks can create a bottleneck if it is not optimised.

We propose a method to recommend optimal parallelization settings to users depending on the type of application. We solve this optimization problem through machine learning, based on system and application metrics collected from previous executions. This way, we can detect and explain the correlation between an application, its level of parallelism and the observed performance. The model keeps learning from the executions in the cluster, becoming more accurate and providing several benefits to the user, without any considerable overhead. In addition, we also consider executions that failed and provide new configurations to avoid these kinds of errors. Finally, bottlenecks related to concurrent access to disk from various tasks are eliminated thanks to the optimal parallelism.

Results

The benefits of an optimal task parallelisation were proven on a series of experiments in the Grid 5000 testbed. We build a benchmark that is a combination of Spark-bench (Li M. T., 2015), Bigdatabench (Wang L. Z., 2014) and some workloads that we implemented on our own. We run all the applications of this unified benchmark with three different file sizes as we want to check how the model reacts when executing the same application with different amounts of data. These executions generate a series of metrics that we use to train and evaluate a machine learning model that has as inputs the metrics, the parallelism configuration and the duration of the workload. The final goal is to be able to predict the effect of different parallelism configurations on the duration of the workload depending on its metrics (the profile of the application). The overhead of applying the model is minimum, and the improvement in the workloads runtime goes up to 50% for graph processing processes (Hernández, 2018). This is a first step towards optimizing the access to data in distributed computing platforms.

4.2 *ESR15: Root cause analysis system for dynamic large computing environments*

Contributor: *Michał Zasadziński, CA*

Overview

Today's IT systems and storage systems are large, dynamic, complex, and heterogeneous. The current and the future systems will frequently change their architecture and resources according to the business and user demand. Diagnosing them efficiently in satisfactory time (less than minutes) is already not within reach of even the most experienced operators. Because of that, the majority of trends and efforts around the development of troubleshooting and diagnostics of IT systems is driven by NoOps. NoOps stands for No Operations. One of the ways is the software automation. Then, it means a scenario of fully automated and self-manageable IT infrastructure. The shift of conventional operations to NoOps model is achieved by the full automation of maintenance activities, including failure diagnostics. In this model of maintenance, problems occurring in an IT system are solved immediately without any human intervention.

However, to operate successfully in such a business model, the future diagnostic systems should perform precise, automated and fast root cause analysis. Also, these solutions should be able to diagnose problems even in a scenario where there is none or few data about failures and their causes. In many cases, recollecting the data necessary for diagnostics is expensive or even impossible. The use of similar data coming from another system with a different structure is a solution, but it is a considerable challenge. The solutions based on transfer learning can transfer and reuse as much knowledge on the behavior of a system as possible to keep pace with the changing architecture, infrastructure and rapidly growing number of knowledge domains.

We propose a cross-system root cause classification framework based on similarity evaluation of weighted graphs with multi-attribute nodes. The framework uses logs, metrics, configuration and connectivity information to represent the state of a system as a graph. Then, the framework evaluates the similarity between an abnormal state and a collection of previously diagnosed states. By finding the most similar graph in the solution space, we can classify the anomaly and provide a root cause. Moreover, we use automatically calculated weights to highlight the system metrics that better describe a failure. Finally, we use the framework for a cross-system failure classification. By acquiring a collection of diagnosed anomalies for one system architecture, we can establish the root cause for anomalies that occur in a completely different architecture (cross-system diagnostics).

Results

We proposed a framework for finding the nearest failure cause (including problems with storage systems) via similarity evaluation of weighted graphs. The framework is aimed to diagnose one system when the knowledge about failures is acquired from another system with a different structure. An example would be a new system that has just started operating, it fails, and it is hard to diagnose it. Also, the proposed framework aims to facilitate knowledge transfer between systems and operators. Firstly, we described the whole framework and its contributions. The most significant contributions are automatic calculations of metric weights, integration of logs with system topology and metrics into graph representation of a system and leveraging historical metric values for similarity calculations. Then, we evaluated the proposed framework in total with four different systems. We inject common anomalies and failures, such as hardware overload, node crash, and network disconnections. In the first evaluation section, we use Spark and Hadoop clusters.

We confirm the quality of root cause classification that achieves average f1-score of 0.71 for Hadoop and 0.61 for Spark. These results show that the framework outperforms state of the art methods. In the second evaluation, we utilize a cloud environment of containers. We evaluate cross-system diagnostics via knowledge transfer. That means diagnosing a target system when knowledge about failure causes and anomalous states is known only from a source system. We run two scenarios: Kafka acting as the source system and Cassandra as the target one and vice versa. Cross-system diagnostics reaches average f1-score of 0.77. The achieved results confirm that the proposed framework, and in particular its ability of knowledge transfer, allows reaching the state of self-manageable IT systems.

5 Bibliography

Hormann, P., & Campbell, L. (2014). Data Storage Energy Efficiency in the Zettabyte Era. *Australian Journal of Telecommunications and the Digital Economy, 2*(3).

Adnan, M.A., Sugihara, R., Yan Ma, Gupta, R.K. 2013. Energy-Optimized Dynamic Deferral of Workload for Capacity Provisioning in Data Centers. Green Computing Conference (IGCC). 2013 International. 1 – 10. DOI=10.1109/IGCC.2013.6604515 . (n.d.).

Aroca, J., Chatzipapas, A., Anta, A., & Mancuso, V. (2015). A Measurement-Based Characterization of the Energy Consumption in Data Center Servers. *IEEE Journal on Selected Areas in Communications, 33*(12), 2863-2877.

Bash, C., & Forman , G. (2007). *Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center.* HP Laboratories Palo Alto .

Beloglazov, A., & Rajkumar, B. (2010). *Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers.* The University of Melbourne, Australia, CLOUDS Lab, Dept. of Computer Science and Software Engineering, Melbourne.

Beloglazov, A., Buyya, R., Choon Lee, Y., & Zomaya, A. (2011). A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. In *Advances in computers* (pp. 47-111). Elsevier Inc.

Bennacer, L., Amirat, Y., Chibani, A., Mellouk, A., & Ciavaglia, L. (2015, January). Self-Diagnosis Technique for Virtual Private Networks Combining Bayesian Networks and Case-Based Reasoning. *IEEE Transactions on Automation Science and Engineering, 12*(1), 354-366.

Bennacer, L., Ciavaglia, L., Ghamri-Doudane, S., Chibani, A., Amirat, Y., & Mellouk, A. (2013, June). Scalable and fast root cause analysis using inter cluster inference. *IEEE ICC - Next-Generation Networking Symposium*, (pp. 3563-3568).

BigStorage Official Website. (n.d.). Retrieved from http://bigstorage.oeg-upm.net

BigStorage Project Proposal for Horizon 2020 H2020-MSCA-ITN-2014 Call. (n.d.).

BigStorage: Storage-based Convergence between HPC and Cloud to handle Big Data. (n.d.). (European Comission) Retrieved from http://cordis.europa.eu/project/rcn/193971_en.html

Bronevetsky, G., Laguna, I., Bagchi, S., & R. de Supinski, B. (2012). Automatic fault characterization via abnormality-enhanced classification. *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012).*

Brown, A. L. (1992). A Generic Persistent Object Store. *Software Engineering Journal, 7*(2), 161-168.

Chavira, M., Darwiche, A., & Jaeger, M. (2006). *Compiling relational Bayesian networks for exact inference.* Int. J. Approx. Reasoning 42(1-2).

Chen, T.-H. a. (2014). Detecting Performance Anti-patterns for Applications Developed Using Object-relational Mapping. *Proceedings of the 36th International Conference on Software Engineering* (pp. 1001-1012). ACM.

Chen, Y.-K. (2012). Challenges and Opportunities of Internet of Things. *17th Asia and South Pacific Design Automation Conference*, (pp. 383-388). Sydney.

Darwiche, A. (2003, May). A Differential Approach to Inference in Bayesian Networks. *Journal of the ACM , 50*(3), 280-305.

Díez, F., & Druzdziel, M. (2006). *Canonical probabilistic models for knowledge engineering.* Madrid, Spain.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, *99*, pp. 1300-1309.

Getoor, L., Friedman, N., Koller, D., Pfeffer, A., & Taskar, B. (2007). Probabilistic Relational Models. In *Introduction to Statistical Relational Learning* (pp. 129-174). MIT Press.

Guerra, J., Pucha, H., Glider, J., Belluomini, W., & Rangaswami, R. (2011). Cost Effective Storage using Extent Based Dynamic Tiering. *In Proceedings of the 9th USENIX conference on File and stroage technologies (FAST'11).* Berkeley, CA, USA: USENIX Association.

Guo, Z., Zhou, D., Lin, H., Yang, M., Long, F., Deng, C., . . . Zhou, L. (2011). G2: A Graph Processing System for Diagnosing Distributed Systems. *Proceedings of the 2011 USENIX Annual Technical Conference (USENIX ATC '11).*

Kiciman, E., Maltz, D., & C. Platt, J. (2007). Fast Variational Inference for Large-scale Internet Diagnosis. *Advances in Neural Information Processing Systems 20.*

Kwisthout, J. (2011, December). Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning, 52*(9), 1452–1469.

Li, S., Abdelzaher, T., & Yuan, M. (2011). *Tapa: Temperature Aware Power Allocation in Data Center with Map-Reduce.* Proc. Int'l Green Computing Conf. and Workshops (IGCC).

M. Vaquero, L., & Rodero-Merino, L. (2014, October). Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *ACM SIGCOMM Computer Communication Review, 44*(5), pp. 27-32.

Miyazawa , M., & Nishimura, K. (n.d.). *Scalable Root Cause Analysis Assisted by Classified Alarm Information Model Based Algorithm.* KDDI R&D Laboratories, Inc., Ohara Fujimino City.

Mullen, S., Bostoen, T., & Berbers, Y. (2013, June). Power-Reduction Techniques for Data-Center Storage Systems. *ACM Computing Surveys, 45*.

Nguyen, H., Shen, Z., Tan, Y., & Gu, X. (2013). FChain: Toward black-box online fault localization for cloud systems. *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, (pp. 21-30).

Niggemann, O., Biswas, G., S. Kinnebrew, J., Khorasgani, H., Volgmann, S., & Bunte, A. (2015). Data-Driven Monitoring of Cyber-Physical Systems Leveraging on Big Data and the Internet-of-Things for Diagnosis and Control. *Proceedings of the 26th International Workshop on Principles of Diagnosis.* Lemgo.

Ogawa, S., Kamimura, K., Kato, T., Uehara, T., & Okuda, H. (2001). Performance analysis of hierarchical storage management systems for video retrieval system. *Consumer Electronics, 2001. ICCE. International Conference on*, (pp. 328-329).

Ogawa, S., Kamimura, K., Kato, T., Uehara, T., & Okuda, H. (2001). Performance analysis of hierarchical storage management systems for video retrieval system. *Consumer Electronics, 2001. ICCE. International Conference on*, (pp. 328-329).

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference.* Los Angeles: Morgan Kaufman Publishers.

Shi, H., Arumugam, R. V., Foht, C. H., & Khaing, K. K. (2012). *Optimal disk storage allocation for multi-tier storage system.* Data Storage Institute (DCT Division), Nanyang Technological University.

Shumann, J., Mbaya, T., Mengshoel, O., Pipatsrisawat, K., Srivastava, A., Choi, A., & Darwiche, A. (2013, December). Software Health Management with Bayesian Networks. *Innovations in Systems and Software Engineering, 9*(4), 271-292.

Singh, H., & Reuters, T. (2011). *Data Center Maturity Model.* The Green Grid.

Singh, H., & Reuters, T. (2011). *Data Center Maturity Model.* The Green Grid.

Wang, C., Schwan, K., Laub, B., Kesavan, M., & Gavrilovska, A. (2014). Exploring Graph Analytics for Cloud Troubleshooting. *11th International Conference on Autonomic Computing.* Philadelphia.

Wuillemin, P., & Torti, L. (2011). *Patterns discovery for efficient structured probabilistic inference.* . In Proceedings of the 5th international conference on Scalable uncertainty management (SUM'11), Salem Benferhat and John Grant (Eds.). Springer-Verlag, Berlin, Heidelberg, 247-260.

Yemini, S., Kliger, S., & Mozes, E. (1996). High Speed and Robust Event Correlation. *IEEE Communications Magazine*, 82-90.

Zermani, S., Dezan, C., Euler, R., & Diguet, J. (2014). Online Inference for Adaptive Diagnosis via Arithmetic Circuit Compilation of Bayesian Networks. *Designing with Uncertainty: Opportunities & Challenges workshop.* York, UK.