peaCS - Performance and Efficiency Analysis for Cloud Storage

Josef Spillner, Maximilian Quellmalz, Martin Friedrich, Alexander Schill

Technische Universität Dresden,
Faculty of Computer Science,
{josef.spillner,alexander.schill}@tu-dresden.de,
{maximilian.quellmalz,martin.friedrich}@mailbox.tu-dresden.de

Abstract. Those who need to store larger amounts of data either for burst periods or for convenient synchronisation between devices are currently looking at new ways of how to integrate cloud storage services into the data processing applications. The benefits (on-demand access and pricing, elasticity) and the drawbacks (reduced control, increased dependencies on providers) need to be well balanced. Recently, a new class of applications called cloud storage controllers or integrators appeared with a high potential to become a standard feature of operating systems and network gateways. They seamlessly integrate single and bundled storage services into the users' digital environment. However, it is not clear which controllers are better than others. Some are open source, some commercial, some perform better than others, but some of the others provide more data pre-processing and routing features. We solve this problem by introducing peaCS as a test framework to analyse, compare and optimise the functional and non-functional properties of such client-side storage service integration solutions.

1 Motivation

Access to the cloud is shifting from an ad-hoc style to planned and systematic integration. Often, either a hardware appliance (e.g. for cryptographic key management) or a software gateway (e.g. for enforcing a policy of not using compute clouds during expensive periods) is used to manage the controlled access. For cloud storage services in particular, differently named service-independent client-side storage controllers, integrators, gateways, "cloud filesystems" or access applications are becoming more popular [24]. The popularity causes an increasing number of prototypes to become available and thus ensures healthy competition [18, 8]. On the downside, it becomes more difficult to evaluate and assess the best solution for this task. Especially non-functional properties such as performance or efficiency, while becoming more commonplace as standard metrics in service descriptions [14], are hard to determine for client-side integration tools. The difficulty aggravates when considering coordinated distributed storage integration across multiple independent services in which the individual service metrics shall be encapsulated as much as possible [19].

This trend motivates us to present a test framework to systematically determine these metrics. Fig. 1 positions the testing methodology as a special case of Service Bundle Integration Testing (SBIT) in which a number of independent reference services, real or simulated, are used in combination during the test execution. SBIT is distinct from Service Testing (ST) [7, 11] which focuses more on invocation and protocols and less on the eventual effect on the client. It is also unlike most Service Integration Testing (SIT) methods which focus on a single service interface with one or multiple applications during one test run [3]. SBIT is applicable to all XaaS service classes, including Cloud Storage on the IaaS and PaaS levels for which the combination technique is data splitting, multiplexing and dispersion.

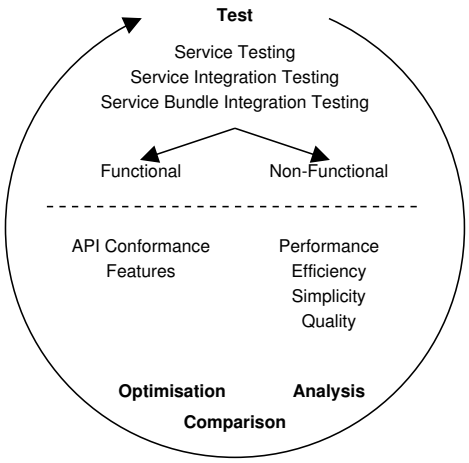

Fig. 1. General overview about Service Bundle Integration Testing

The next section gives an overview about testable cloud storage controllers and storage service integrators. Then, the *peaCS* test framework is introduced and explained with multiple experiments in which we analyse, compare and optimise storage controllers. In the fourth section, its functionality is compared to related work, before concluding the paper in the fifth section.

2 Overview about Multiplexing Cloud Storage Controllers and Libraries

The client-side integration of single and multiple cloud storage services happens on multiple layers. The lowest layer encompasses the algorithmic treatment and pre-processing of data, which includes dispersion, replication, de-duplication and encryption. A middle layer offers an interface to applications, e.g. through a service interface or a virtual file system, to let data in to and out of the lowest

layer. Higher layers offer storage service selection, attachment and configuration as well as user and permission management. Such multi-layer integration systems for cloud storage services are usually called storage controllers or integrators. We will first present a number of file system-based prototypes which appear to be the least common denominator among all types and are typically a superset of all service-based unification interfaces like DeltaCloud or jClouds. Afterwards, we extend the presentation to promising libraries for data treatment which may influence future storage controllers and hence motivate the need for systematic testing and comparison.

2.1 Controllers, Integrators, Gateways & Filesystems

NubiSave [19] is an Optimal Cloud Storage Controller which takes storage service properties into account to achieve an optimal data distribution through replication and dispersion. It offers a FUSE file system interface and hence is a valid candidate for file-based testing on all operating systems supported by FUSE, such as Linux, BSD and Mac OS X. A strong characteristic of NubiSave concerning testing is the hot-plugging of storage service providers by means of writing configuration files into a special directory. ChironFS[1] is a RAIFS controller, essentially functioning like a RAID controller on a file system level. Compared to NubiSave, it only replicates files and doesn't allow for more fine-grained configuration. The dispersing Secure Cloud Storage Integrator for Enterprises (SecCSIE) [16] focuses on Intranet integration and offers a CIFS interface to network clients. Most operating systems can natively import CIFS drives as local directories. In a similar way, the Least Authority File System (Tahoe-LAFS) operates as a gateway with HTTP(S) and (S)FTP interfaces which can be mapped to a local directory [25]. Hence, SecCSIE and Tahoe-LAFS are also valid candidates for file system-based testing. One design difference between them is that SecCSIE integrates arbitrary storage services whereas the Tahoe-LAFS gateway assumes Tahoe-LAFS storage backends. This difference is well hidden behind a unifying file system interface. RACS, a Redundant Array of Cloud Storage proxy [1], exposes an S3 bucket interface for which FUSE file system translators exist. FUSE pass-through adapters typically add an overhead of about 8-15% in the worst case so it remains feasible to test RACS. The Iris cloud file system, in contrast, already offers a remote file system natively [20]. More FUSE modules exist for individual online and cloud storage services, e.g. CloudFusion[2] for DropBox and SugarSync, S3QL for Amazon S3, Google Storage and OpenStack Swift, FuseDav for WebDAV and CurlFtpFs for FTP. In general, single-service integration offers less parametrisation space compared to multi-service integration, but otherwise the former is a subset of the latter and hence the same test methods are applicable.

Additional storage integration tools exist without an appropriate data management interface which can be accessed automatically as part of a SIT or SBIT

[1] ChironFS: http://www.furquim.org/chironfs/index.en.html
[2] CloudFusion: https://github.com/joe42/CloudFusion

test execution. Trust Store [10], for instance, requires the drag and drop interaction with a GUI to initiate the storage and retrieval of files. Such tools are out of scope for peaCS and require GUI testing frameworks like Sikuli. Similarly, Dependable Sky [4] only offers a library interface, the commercial product Trusted Safe a Windows plugin interface and another unnamed platform [15] a web interface which all require additional test agents with user interface interaction intelligence. For Cloud Shredder [23], the interface is not specified. A survey on distributed cloud storage integration compares the security characteristics which are hard to measure and hard to assess in an automated way [18]. Characteristics determined in such a manual process complement the expected results of automated test frameworks. Therefore, our work aims at a subset of storage controllers (those with a file system interface) and a subset of metrics (which can be measured or calculated).

2.2 Data Pre-Processing Libraries

The transformation and distribution of files and file parts is often captured in specific libraries which are used by some of the systems presented in the previous paragraph. A well-known dispersion and secret sharing library is Jerasure in version 1.2 and 2.0 [12]. It performs erasure coding with optional encryption of the resulting file fragments. Each fragment is then distributed to a storage service by the surrounding controller. Both erasure codes and implementations thereof are still subject to research, for instance, to adapt them to efficient SIMD processor instructions and hard-wired cryptographic routines. Therefore, a test framework is useful to track the progress of systems over time even when only internal parts like a pre-processing library change.

Alternative dispersion libraries with comparable functionality are JigDFS, Schifra, dsNet, Crypto++, IDA-Java, the Tahoe-LAFS library ZFec, JSharing and StorageCORE. Varying run-time characteristics result from design and implementation differences between them, last but not least due to different programming languages – C/C++, Java, Python, Haskell and VHDL for the mentioned libraries. Further pre-processing, which includes compression, encryption, de-duplication, steganography and versioning, is realised either by FUSE modules or by additional libraries. For some algorithms, like AONT-RS secret sharing, no publicly available implementation exists.

It is important to understand that the performance and efficiency of distributed storage depends not only on the algorithms and implementations, but also heavily on configurable parameters such as redundancy, degree of parallelism, distribution scheduling, streaming support and file system buffers. For cloud storage service bundles, service selection preferences influence the results further. For arbitrary combinations of these parameters, a test framework must support multi-dimensional result sets.

3 The peaCS Test Framework

We introduce the test framework *peaCS*, which stands for Performance and Efficiency Analysis for Cloud Storage, in order to allow for systematic testing, analysis and comparison of file-based cloud storage controllers and integrators. The strength of peaCS is the coordinated, controlled and repeatable test of variegated storage service combinations by instrumenting a target controller as the subject of testing.

3.1 peaCS Architecture

Driven by the desired features, the design of peaCS mandates flexibility concerning the definition of test executions, extensibility of test cases and reproducibility of test results. The resulting test suite architecture is shown in Fig. 2. The main application of peaCS is implemented as a shell script. It can be extended by plugins (realised as include scripts) which perform the controller configuration and subsequently the actual test runs. The behaviour of peaCS is driven by both a configuration file and overriding command-line switches and interactive commands. Test results can be compared against previously measured and determined gold standards. All output, including temporary scratch files, informative logs, numerical results and visualisations derived from them are stored in appropriately versioned locations. On the implementation level, Git, Gnuplot and various measurement tools like Iozone are used in this workflow.

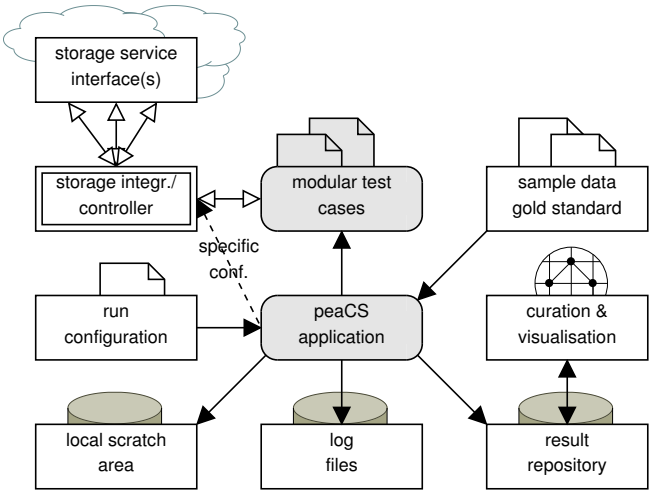

Fig. 2. Architecture of peaCS for coordinated storage integration testing

Within peaCS, variability is achieved by combining server-side parameters such as storage service selection and configuration with client-side parameters

such as weights, redundancy and scheduling methods. Storage services and datasets can be simulated through mass-generated local directories and files, but can also be picked up from an existing configuration. Remote storage services can be integrated semi-automatically by supplying a file with a list of accounts.

The variability is captured in the modular run configuration. Each step of the test sequence can be switched on and off. Listing 1.1 conveys the structure of the peaCS configuration file with portions of the key-value parameter pairs.

Listing 1.1. Configuration file sections of peacs.conf

```
[global]
mntpoint = /media/cloud
syslog = /var/log/peacs.log
[samplefiles]
[sampledirectories]
[samplewebdavs]
[redundancytest]
startbyredundancy = 1
[iozonetest]
[availabilitytest]
strategy = AllUseInParallel
[directorytest]
[plot]
```

The goal of peaCS is to determine the performance and the resource utilisation efficiency of storage integration combinations. Additionally, deterministic functional and calculable tests are offered to build regression detection series. These metrics will be discussed in the following three paragraphs before completing the prototype presentation with an example of a test run.

3.2 Performance Determination

In times of increasing big data requirements [5], high performance for data storage, retrieval, search and general processing becomes paramount. Storage controllers should not cause performance penalties through compute-intensive dispersion, encryption or de-duplication tasks, and yet offer these powerful mechanisms with high quality. In peaCS, throughput and performance are measured through Iozone.

3.3 Efficiency Determination

Efficiency and high utilisation of minimal resources are two largely overlapping primary goals in utility and cloud computing. This applies both to energy efficiency [2] and to hardware resource allocation scheduling [9] in addition to efficient time utilisation through high performance. Computational resource consumption, i.e. processor, main memory, disk and network adapter, is measured through Smem, Iozone and further tools in peaCS.

3.4 Functional Testing and Calculations

Functional tests verify the capabilities of the file system interface by invoking all possible functions on it. These functions encompass directory, file and metadata management. As an example, maximum file name lengths and directory nesting limits can be found out this way. Calculations are performed to determine availability values for certain storage service combinations and other repeatable per-configuration results. Higher-level calculations combine measured and calculated values. For example, peaCS can determine the efficiency of required storage capacity per redundancy level.

3.5 Test Run Examples

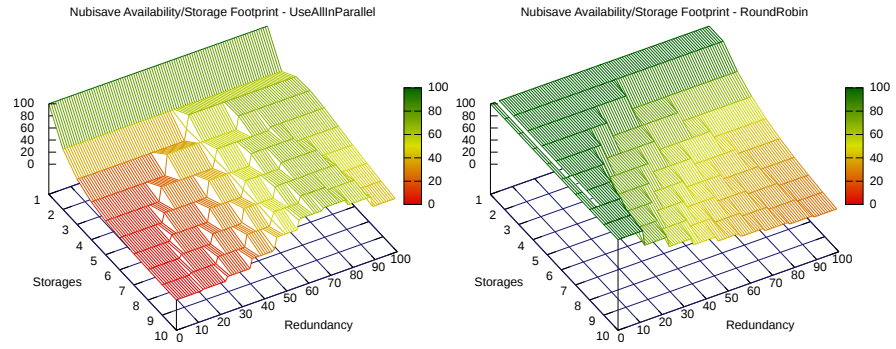

Fig. 3. Comparison of the ratio of file availability and storage requirements, depending on both redundancy and number of storage targets, for both parallel (left) and round-robin (right) scheduling

peaCS allows for multi-dimensional combinatorial variations which result in diagrams with an additional dimension for the calculated or measured target metric, for instance, per-thread performance or RAM utilisation efficiency. The following test results were achieved with the NubiSave cloud storage controller.

Fig. 3 gives an example of a two-dimensional variation. It has been created by 1000 repeated availability calculation calls for each of the two main file part distribution strategies. The results are independent of the hardware and must be reproducible on any machine. In contrast, measured metrics differ depending on the experimentation system and must at least be normalised before a comparison.

Fig. 4 represents a different visualisation of the availability calculation. peaCS generates multiple table and plotting instruction files by default. This increases the chance to spot anomalies and interesting artefacts.

Fig. 5 is an example of a measured result. For both strategies, the read performance for a 1 MB file is determined. Given that the experiment used only one hard disk (notebook HDD) to simulate up to 10 storage providers, it comes at

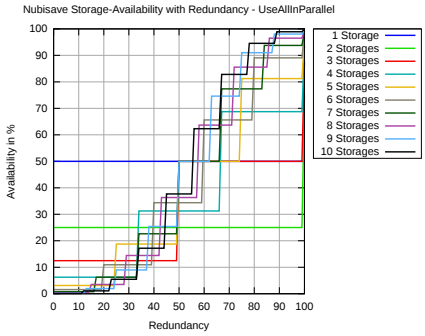

Fig. 4. Alternative visualisation of the parallel availability calculation results

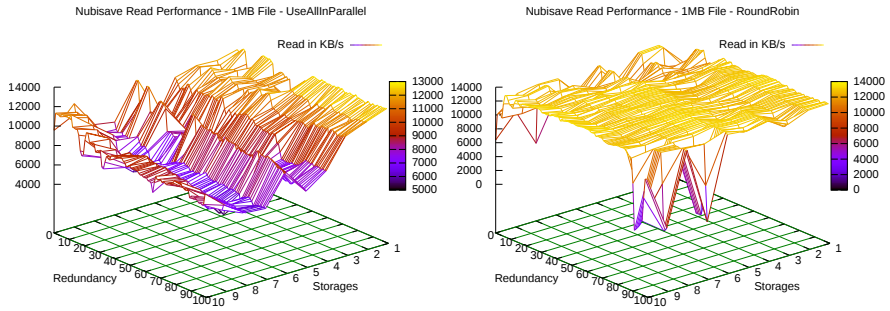

Fig. 5. Comparison of the read performance for both parallel (left) and round-robin (right) scheduling

no surprise that the performance suffers when the file needs to be assembled from independently read file parts (fragments) under the UseAllInParallel strategy, compared to a single large read with nearly constant performance under the RoundRobin strategy.

Fig. 6 contrasts the performance in Fig. 5 with the CPU utilisation for read requests. The optimisation goal is to minimise the CPU utilisation and to maximise the performance. However, the optimum resides within the corner of only one storage service with no redundancy, which is not a desired configuration for practical use due to the lack of safety against unavailability and security against confidentiality.

After performing the measurements, the results of peaCS can be analysed and compared between implementations and parametrisations to derive optimisation targets. Fig. 7 shows an example of a comparison between the ChironFS replication file system and NubiSave with two strategies at 100% redundancy. For all three configurations, a video file of 87 MB was copied 100 times to the integration folder which replicated the file to a number of simulated cloud storage folders, each of which was located on the same disk. The experiments were conducted on a notebook with Intel Core i7 M620 CPU (4 cores) and a 320 GB

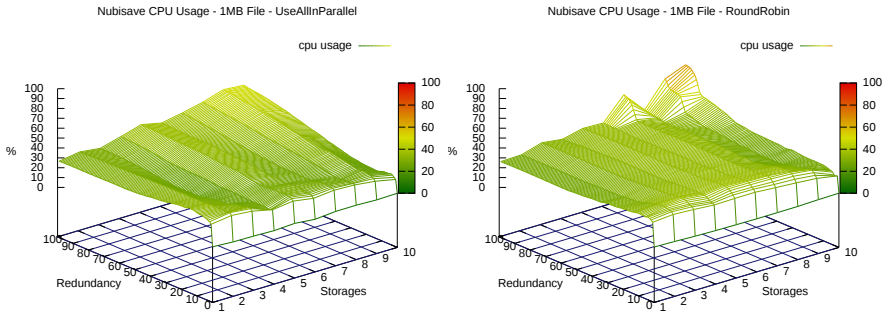

Fig. 6. Comparison of the CPU utilisation for both parallel (left) and round-robin (right) scheduling

Hitachi SATA II disk (5400 RPM, 8 MB cache) running Debian 7.0 for AMD64. The results clearly show an overall optimisation potential for NubiSave, which can be attributed due to its implementation being Java while ChironFS is written in C++, and a particular optimisation potential for the case of 5 or more storages.

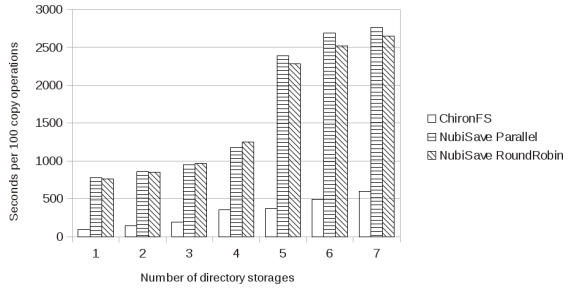

Fig. 7. Comparison between ChironFS and NubiSave n-fold replication performance for $1 \leq n \leq 7$

4 Related Research

Those who define analysis and comparison frameworks should not forget to analyse and compare the framework itself. There are already both experimental and simulation approaches to compare cloud services and file systems, but thus far none to evaluate complementary client-side integration solutions with filesystem or alternative interfaces.

CloudSim [6] uses simulation supports the modelling of distributed cloud environments and simulated experiments inside them. Although it is extensible, it currently does not cover the integration of services on the client. C-Meter [22],

a tool developed from its Grenchmark ancestry, lets the user define artificial workloads which are then executed in the cloud. It is not suitable for storage service bundle integration testing due to its focus on compute clouds. Central storage and filesystem performance comparison, on the other hand, is a well established activity, both for native [21] and for user-space filesystems [13]. Coordinated measurements of multiple file systems for distributed cloud storage service integration are however not covered by these approaches and require the flexible selection and configuration approach taken in peaCS. Some researchers have proposed proof-of-retrievability (PoR) techniques for cloud storage services [17]. These are currently not covered by peaCS but may be added as probes in the future so that the tool will gain usefulness for security-related non-functional properties.

Industrial practice considers cloud storage service testing and certification a necessity[3]. However, as opposed to related industrial domains such as high-performance computing with its omnipresent Linpack benchmark, there is no standard benchmark tool available for storage services and their integration yet. Our intention is that tools like peaCS contribute to the development of a standard way of assessing the client-side combination of cloud storage services.

5 Conclusion

The increasing availability of filesystem-based access to dedicated cloud storage providers or mixed local and cloud storage areas calls for a systematic testing tool. We have motivated the need for such a tool, gathered use cases through an analysis of integration interfaces, and designed the peaCS framework accordingly. Clearly, peaCS is work in progress and warrants large-scale experiments with deduced comparative results. We intend to perform this work in the near future. Its success depends on the availability of storage integration interfaces to the research community.

Acknowledgements

This work has received funding under project number 080949277 by means of the European Regional Development Fund (ERDF), the European Social Fund (ESF) and the German Free State of Saxony.

References

1. Abu-Libdeh, H., Princehouse, L., Weatherspoon, H.: RACS: A Case for Cloud Storage Diversity. In: 1st ACM Symposium on Cloud Computing (SoCC). pp. 229–240 (June 2010), Indianapolis, Indiana, USA

[3] Industrial storage service testing: http://www.nasuni.com/blog/15-testing_the_cloud_storage_providers_part_1-api

2. Beloglazov, A., Buyya, R.: Energy Efficient Resource Management in Virtualized Cloud Data Centers. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid). pp. 826–831 (May 2010), Melbourne, Australia
3. Bertolino, A., Polini, A.: SOA Test Governance: Enabling Service Integration Testing across Organization and Technology Borders. In: International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 277–286 (April 2009), Denver, Colorado, USA
4. Bessani, A., Correia, M., Quaresma, B., André, F., Sousa, P.: DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds. In: Proceedings of EuroSys (April 2011), Salzburg, Austria
5. Boyd, D., Crawford, K.: Six Provocations for Big Data. In: A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society (September 2011), Oxford Internet Institute
6. Calheiros, R.N., Ranjan, R., Rose, C.A.F.D., Buyya, R.: CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. Tech. Rep. GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia (March 2009)
7. Canfora, G., Penta, M.D.: Testing Services and Service-Centric Systems: Challenges and Opportunities. IT Professional 8(2), 10–17 (March-April 2006)
8. Livenson, I., Laure, E.: Towards Transparent Integration of Heterogeneous Cloud Storage Platforms. In: Proceedings of the Fourth International Workshop on Data-Intensive Distributed Computing (DIDC). pp. 27–34 (June 2011), San Jose, California, USA
9. Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., Pendarakis, D.: Efficient Resource Provisioning in Compute Clouds via VM Multiplexing. In: Proceedings of the 7th IEEE/ACM International Conference on Autonomic Computing (ICAC). pp. 11–20 (June 2010), Washington, DC, USA
10. Nepal, S., Friedrich, C., Henry, L., Chen, S.: A Secure Storage Service in the Hybrid Cloud. In: Fourth IEEE/ACM International Conference on Utility and Cloud Computing (UCC). pp. 334–335 (December 2011), Melbourne, Australia
11. Noikajana, S., Suwannasart, T.: An Improved Test Case Generation Method for Web Service Testing from WSDL-S and OCL with Pair-Wise Testing Technique. In: 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC). pp. 115–123 (July 2009), Seattle, Washington, USA
12. Plank, J.S., Simmerman, S., Schuman, C.D.: Jerasure: A Library in C/C++ Facilitating Erasure Coding for Storage Applications. Tech. Rep. UT-CS-08-627, University of Tennessee, Knoxville, Tennessee, USA (August 2008)
13. Rajgarhia, A., Gehani, A.: Performance and Extension of User Space File Systems (March 2010), Sierre, Switzerland
14. Reiff-Marganiec, S., Yu, H.Q., Tilly, M.: Service Selection Based on Non-functional Properties. In: Service-Oriented Computing - ICSOC Workshops, Revised Selected Papers. Lecture Notes in Computer Science (LNCS), vol. 4907, pp. 128–138 (September 2009), Vienna, Austria
15. Schnjakin, M., Meinel, C.: Plattform zur Bereitstellung sicherer und hochverfügbarer Speicherressourcen in der Cloud. In: Sicher in die digitale Welt von morgen – 12. Dt. IT-Sicherheitskongress des BSI. SecuMedia Verlag (May 2011), Bonn, Germany
16. Seiger, R., Groß, S., Schill, A.: SecCSIE: A Secure Cloud Storage Integrator for Enterprises. In: 13th IEEE Conference on Commerce and Enterprise Computing,

Workshop on Clouds for Enterprises. pp. 252–255 (September 2011), Luxembourg, Luxembourg
17. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT). pp. 90–107 (February 2008), Melbourne, Australia
18. Slamanig, D., Hanser, C.: On cloud storage and the cloud of clouds approach. In: The 7th International Conference for Internet Technology and Secured Transactions (ICITST). pp. 649–655 (December 2012), London, United Kingdom
19. Spillner, J., Müller, J., Schill, A.: Creating Optimal Cloud Storage Systems. Future Generation Computer Systems 29(4), 1062–1072 (June 2013), DOI: http://dx.doi.org/10.1016/j.future.2012.06.004
20. Stefanov, E., van Dijk, M., Oprea, A., Juels, A.: Iris: A Scalable Cloud File System with Efficient Integrity Checks. In: 28th Annual Computer Security Applications Conference (ACSAC). pp. 1–33 (December 2012), Orlando, Florida
21. Vanninen, M., Wang, J.Z.: On Benchmarking Popular File Systems. Clemson University Study (2009)
22. Yigitbasi, N., Iosup, A., Epema, D., Ostermann, S.: C-Meter: A Framework for Performance Analysis of Computing Clouds. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID). pp. 472–477 (May 2009), Shanghai, China
23. Zhang, N., Jing, J., Liu, P.: Removing the Laptop On-Road Data Disclosure Threat in the Cloud Computing Era. In: Proceedings of the 6th International Conference on Frontier of Computer Science and Technology (FCST) (November 2011), IEEE Digital Library
24. Zhao, G., Rong, C., Li, J., Zhang, F., Tang, Y.: Trusted Data Sharing over Untrusted Cloud Storage Providers. In: IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). pp. 97–103 (December 2010), Indianapolis, Indiana, USA
25. Zooko, Warner, B., Hopwood, D., Secor, P., Deppierraz, F., McDonald, P., Marti, F., Tooley, M., Carstensen, K.: Tahoe-LAFS: The Least Authority File System. Open source project (2013), `http://tahoe-lafs.org/`