Overview of Cloud Computing large-scale processing technologies

Adam Westerski
Universidad Politecnica de Madrid
westerski@dit.upm.es

In this paper we present a state of the art overview in the contemporary cloud computing research area. By doing so, we focus on the most important technologies that are the backbone behind the modern Internet services and point out some interesting trends. Additionally, we point of various examples of employing cloud computing technologies used in different domains and for achieving different goals.

I. Introduction

In the modern Internet the quality of services measured by performance and efficiency has became a increasingly important feature. The business concept and functionality of the product are sill very important, however even the best web applications will not create any revenue if they are unavailable due to traffic problems.

As Internet popularity grows and as it becomes available for more people the contemporary web applications need to handle huge concurrent processing demands. The high competition on the market of Internet services and development of new application design paradigms (i.e. Web 2.0) has caused a burst of new types of services that let users to shift processing from their local machines to remote servers. Additionally, following the trend to store all kinds of data on-line rather then on a local hard drive, the server-side solutions need to cope with a problem of handling large portions of data and processing very selected subsets on mass scale. The example of a video hosting service called YouTube shows that selected website popularity can grow a lot faster then the average companies capabilities to support reliable and high-end server hardware infrastructure[1].

In the response to such needs a number of companies have developed software solutions and application design paradigms that are known under a common name of cloud computing(ref?) technologies.

In the following article we do not tackle neither analyze the topic of cloud computing services architecture, components etc. Our intent is to gather information about the technologies in the background that make large-scale processing possible and allow to deliver efficient products for end users. Furthermore within such solutions we focus on those referred to as data-intensive scalable computing(DISC)[2]. Namely, we put most emphasis on proposals by Google (see Sec.3) and their implementations prepared by Yahoo and the open-source communities. They are the ones best documented and therefore prove as a good example to depict the mechanisms used in data-intensive processing over the Internet. Additionally, we conclude the state of the art description with information about various interesting platforms delivered by other companies (see Sec. 4).

TABLE I
Large-scale computer systems (based on Ref. [2])

No.	Type	Characteristic
1	Current Supercomputers	compute all sort of complex and time consuming calculation problems; very big multiprocessor machines; superior arithmetic performance per CPU; high-end CPU interconnection technologies; applications written in a very low level language, heavily optimized code
2	Transaction Processing Systems	maintained for the needs of financial institutions, airlines online retailers etc.; main goal is data processing and analysis; big data consistency requirements, strong security and reliability constraints (i.e. airline systems failure can be matter of big financial loss or even involve human life hazard situations).
3	Grid Systems	a form of distributed computing; set of loosely coupled machines perform (usually) large data processing tasks; single nodes are very independent – the data exchange between nodes is minimal due to high time constraints
4	Data-intensive Scalable Computing	similar to grid computing- a set of machines perform data-centric computing tasks;single nodes are better communicated then in a grid, data exchange is still small but more interaction is possible;

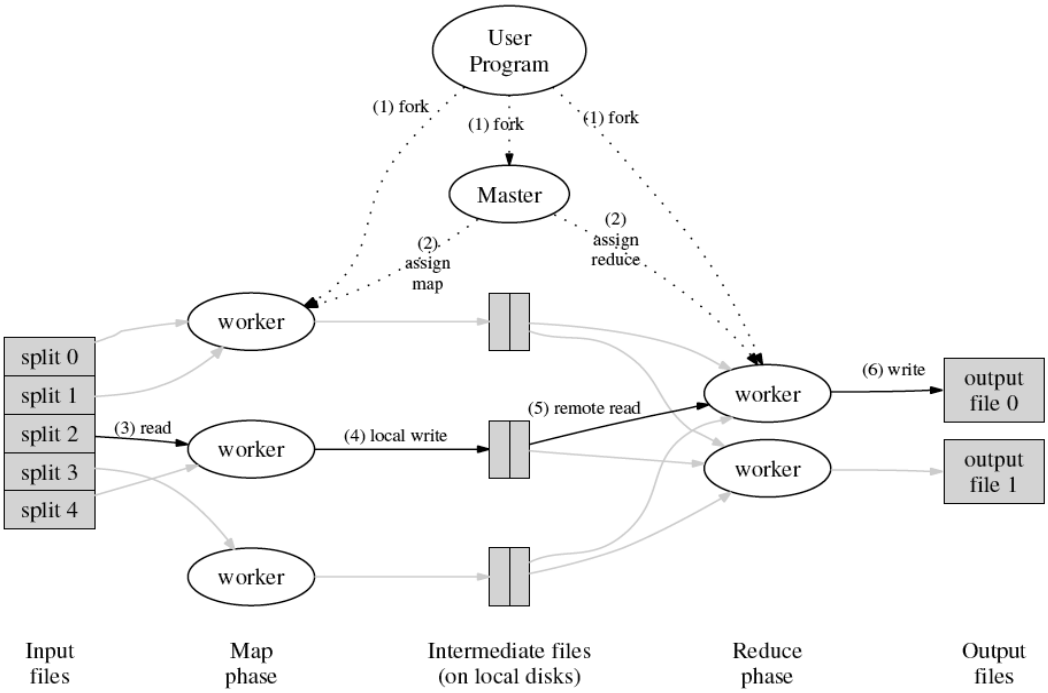

Fig. 1. The MapReduce execution workflow overview[5].

II. Overview

The large-scale processing model characteristic for the aforementioned DISC systems is strictly tied to the hardware architecture that those solutions utilize. In his proposal for constructing a multi-purpose DISC center *Bryant*[2] makes a comparison of large-scale processing systems(see Table 1) and defines some of the characteristics of DISC. Regarding the hardware design he points out most similarities to Grid computing. On the hardware level DISC system consists of a large a number of independent nodes, each with its own (single or multiple) processors and local memory – much like in a grid. The key difference to grid computing is the location of all DISC nodes is a single facility. Whereas Grid systems in the context of Internet network try to take advantage of the existing vast distributed architecture (i.e. SETI[3]) and put it into use, DISC is about creating large computational centers from scratch to respond a very selected area of data processing problems.

The systems that we describe further(see Sec. 3) were originally created to support processing needs of web search engines. Although currently such architectures find many other appliances the trend was initiated by the Inktomi company that created a 300-processor system back in 1998 for their search engine infrastructure[4]. The model was later adopted by the current top players in the search engine market and also other companies that deliver solutions classified as could computing(see Sec. 4). The general computation paradigm is driven by the distribution of data across many hardware nodes. Each node supplied with its own processing unit performs certain operations on local data. The output is later aggregated by a master scheduler and a number of slave scheduler tasks. Regardless how simple that does sound in theory, in practice, in a distributed environment it proves to be a very difficult task. In the next section we present how Google, currently being one of the most often explored case studies in DISC, coped with the task by proposing a number of abstraction layers to simplify application development for their distributed data centers. Namely: MapReduce[5] – a programming model, BigTable[6] – a non-relational distributed database, GFS[7] - distributed file system, being the base for BigTable.

III. MapReduce, Bigtable and their implementations

A. MapReduce

MapReduce is a programming model designed to simplify the task of processing large datasets across hundreds of machines. The abstraction layer assumes that every task can be accomplished with two basic steps: mapping data and aggregating the results of map functions across all nodes(see Fig. 1). In practice the developer needs only to design two functions for each of those operations. Such approach enables to forget about problems of parallel computing, data distribution, failure handling and many others. The developer only needs to focus on solving his problem with those simple tools:

$$map\ (k1, v1) \rightarrow list\ (k2, v2) \quad (1)$$

$$reduce\ (k2, list(v2)) \rightarrow list\ (v2) \quad (2)$$

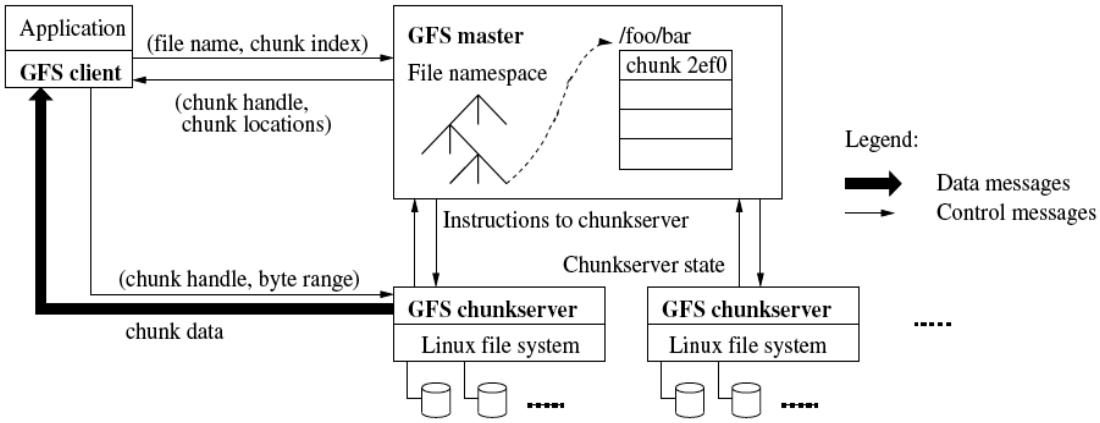

Fig. 2. The GFS Architecture[7].

The map function (1) takes a set of input key/value pairs and produces output key/value pairs. Next, the MapReduce logic aggregates all values from outputs of *map* functions with the same key and passes them to the *reduce* function (2). The job for the reduce function is to merge all values into a smaller subset -typically just a zero/one output. To explain this better we present a simple example provided by *Dean et al.*[5].

```
map(String key, String value) :                    (3)
// key: document name
// value: document contents
for each word w in value:
  EmitIntermediate(w, "1");

reduce (String key, Iterator values) :             (4)
// key: a word
// value: a list of word counts in a document
int result = 0;
for each v in values:
   result += ParseInt(v);
Emit(AsString(result));
```

In the above example the *map* function (3) emits occurrence count for each word it finds in a document. The *reduce* function (4) will sum all word counts for each word across many documents. The full C++ code for this example has been presented in Ref. [5].

Although the developer only needs to implement the two aforementioned functions it is also good to know how the execution flow proceeds to get a better understanding of the solution architecture(see Fig. 1):

1. Input files are split into a number of pieces, assigned to machines and map functions are started on the corresponding hosts.
2. One of the machines is given master status. It is responsible for assigning the tasks(map or reduce) to other idle machines – the workers.
3. The worker machine reads the assigned data, extracts key/value pairs and passes them to the map function.
4. Periodically the output of map functions are written from the memory buffer to local disks and partitioned for the reduce workers. The locations along with notifications of write operations are passed to the master machine which is responsible for alerting reduce workers.
5. Reduce workers use remote calls to read data of the map workers disks. Before doing their tasks the reduce workers sort the read data according to keys and next group all values for the same key.
6. The reduce workers iterate over their set of unique keys and pass the parameters to reduce function. The output is written to the output file for the corresponding data partition. The final output is available as a set of files(their number is equal to the amount of reduce workers).

The presented workflow is also employed by Hadoop[8] – an implementation of MapReduce. In 2004 Google did release information about research on data-intensive processing but the papers only describe the general idea and some evaluation results. Due to obvious commercial reasons no working software or implementation details were made available. However, Google's direct competitor – Yahoo went a step further. They took the knowledge delivered by Google and produced implementations that exactly comply to the described model. Furthermore the basic versions of the implementations were disclosed and are managed as open-source projects. The benefits of such actions can be seen both in the commercial and research environments by viewing the number and the characteristics of most successful deployments[9]. Apart of the informative value the list can be a good source of success stories that tell where technologies like Hadoop and MapReduce apply. Additionally analysis of vendors and centers that use Hadoop exposes the vast scaling capabilities.

The biggest reported deployment of Hadoop by Yahoo operates on a cluster with 10,000 cores in total[10]. At the same time, successful deployments in other companies and some of the universities show that even with a small number of nodes (i.e. 10-50) the technology can be useful.

B. Google File System (GFS)

The Google File System (GFS) among other is a technology that lets MapReduce (and also further described BigTable) read/write data in a distributed environment like it was a single virtual hard drive. The characteristics of this system are very much determined by the hardware that Google uses for its web search engine and other applications.

The clusters are constructed of commodity parts, therefore there are very prone to failure. In case of a system constructed from hundreds or thousands of machines, permanent equipment failures are a normal part of the process rather then an exception. Issues such as fault detection, monitoring and recovery are a very important part of the software.

The entire system is supposed to run only for a particular type of tasks. With respect to data read and write operations applications that run on Google clusters do not act like normal desktop software. The size of data accessed data is different and the frequency of requests for selected subsets as well. In GFS single files are a lot bigger then on a casual system, ranging up to gigabytes of data. Also, for efficiency reasons, when writing files, the new data is appended rather then overwritten. Therefore, in practice, once written files are only read.

Although the file system constructed by Google has totally different constraints then the regular operating file systems, it presents more or less the same interface to the user. The files are identified by paths and organized hierarchically in directories. Furthermore the system does support the usual operations like: create, delete, open, close, read, write.

The general architecture of GFS is rather simple (see Fig. 2) - the system is composed of a single master and multiple slave machines (called chunkservers) that store the actual files. Similarly as in typical file systems the files are split into chunks, however in GFS the chunk size is a lot bigger (64 MB). Due to the aforementioned equipment failure problems each chunk is replicated. The GFS master stores metadata about the filenames, file location in the abstract directory hierarchy and location of the file chunks and their replicas on the chunkservers. The single master architecture simplifies the entire system, however it also exposes the danger of master becoming the bottleneck of the system. Therefore all the read/ write data operations are exchanged directly between chunkservers and the application client.

Similarly like with MapReduce, GFS also has its counterpart within Yahoo's products. It is called Hadoop File System(HDFS)[11].

C. Bigtable

Bigtable is a distributed storage system based on the Google File System. Its suppose to leverage from the file system in order to store structured information much like a database. However, contrary to the relational databases, it provides a lot less storage and data organization tools and at the same time delivers a lot better performance for huge datasets.

It has to be noted, that similarly as MapReduce and GFS, Bigtable is not meant for everyone. The range of appliances is very selected. The user can only read and write data to a single map. There is no sophisticated query language(i.e. like SQL) nor data split to tables with relations.

(row:string, column:string, time:int64) → string

Fig. 4. Row indexing in Bigtable[6].

In the Bigtable data model, the rows in the map are indexed by a row key, a column key and a timestamp. Unlike relational databases there are no datatypes – each row is just an array of bytes (see Fig. 4).

Additionally, column keys are grouped into *column families* common for each row. The concept of families extends the regular concept of a column in a relational table to bit to make the map model more flexible. Each family usually contains the same types of data. The continues columns in the family are addressed by adding a qualifier to the family name(see Fig. 5). Finally the third type of indexing in the Bigtable is the timestamp. The data model allows to store many versions of the same data in a single row. Each is indexed by either server time of the moment data was stored (measured in milliseconds) or is explicitly assigned by the client application.

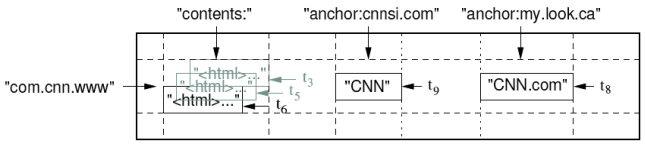

Fig. 5. Sample row from Bigtable with inverted page URL address as row index and “contents” and “anchor” column families. The t3, t5, t6, t8, t9 values symbolize the third type of indexing in Bigtable -timestamp [6].

The open-source implementation of BigTable originally delivered by Yahoo is called Hbase[12].

D. Implementations

Apart of the most notable, earlier mentioned Google and Yahoo implementations the discussed technologies find use in other companies as well. Most often other vendors adopt the already available code of Hadoop and extend it for their particular needs[9]. However there is a number of original contributions as well. In this subsection we describe some of the interesting implementations while later (see Sec. 4) we mention particular services that employ DISC to deliver some business value.

Greenplum[13] and Aster Data Systems[14] both provide their individual implementations of MapReduce paradigm with focus on database processing and large-scale data analysis.

Interestingly both of those companies merge MapReduce principals with regular SQL processing rather then solutions similar to BigTable. Contrary to the JAVA based Hadoop both solutions support a variety of other languages like C/C++, Perl or Python.

Similar as Hadoop, the GridGain[15] is an open-source MapReduce implementation. From the technical point of view the biggest difference is in the initial process of *Map* tasks assignment to the nodes. In the MapReduce algorithm the task is split into subtasks and workers pull the split parts as soon as they have free processor time. In GridGain the subtasks are pushed to the nodes. Authors claim that this proves to be an advantage since it gives more load balancing capabilities. It practice it should be noted that this benefit is rather situational and depends on users needs. Apart of extra functionality it introduces some additional complexity – the developer has to plan ahead so that no worker does stay needlessly idle. Although GridGain seems to be far less popular then Hadoop, it shows to be better documented and is more welcoming for beginners.

Additionally there is a number of minor MapReduce implementations that did not go into main stream far various reasons. Some are done for specific hardware platforms, including: Phoenix [16] for shared-memory systems like Solaris, MapReduce implementation on Cell[17] or Mars[18] an implementation for graphic processors(GPU). Others are written for use with less popular programming languages like: Skynet[19] for Ruby, Disco[20] written in Erlang and Python or Holumbus[21] a library for Haskell language. Finally some just employ MapReduce ideas to achieve very specific tasks: FileMap[22] a lightweight implementation meant to be used alongside with Unix command line file processing tools or CouchDB[23] a document-oriented database that uses some of MapReduce principals to achieve better scalability.

IV. Cloud computing technology appliances

In the previous sections we have given an overview of the large-scale processing technologies and have shown how they work. In this section we introduce some of the actual cloud computing products that take advantage of the distributed computing paradigms.

As it might be expected, both Google and Yahoo use the technologies very extensively in their products. With respect to Google it discloses several examples and evaluation results in the papers about BigTable[6] and MapReduce[5]. Regarding MapReduce the examples are quite general and among others include problems such as: machine learning and data generation for web search service, clustering information for Google News, extraction of data used for reports in popular queries. The BigTable is used in products like Google Analytics[1], Google Earth[2] or Google Personalized Search[3]. Depending on the solution the use of BigTable is different. For example, in Google Analytics the tables are used to store large quantities of statistical information about websites (i.e. user visits per day). Interestingly this example also exposes use of MapReduce. Among others, Google Analytics uses two tables: one that has a row for each single user visit (identified with website name and date of the visit – table size around 200TB), second table with summaries for each website (size up to 20TB). The second table is generated using MapReduce from the contents of the first table.

Apart of the products that use large scale processing in a transparent way for the end user it is possible to for a casual developer to take advantage of Google's resources through Google AppEngine[24]. The company claims to share the very same tools and resources that their own developers use. Therefore the product offers web application serving along with persistent storage both based on Google large-scale processing technologies. However, applications have to be implemented in Python and are a subject of limitations and quotas for storage space and bandwidth. A huge benefit is the availability of the entire service for free with some basic limitations (i.e.500 MB storage space and 5 million views per month) that can be extended by enabling the paid version.

Other interesting vendor of large scale services is Amazon[4]. The company delivers both online storage services and processing power for rent. The synergy between the services allows to use them as complementary products to run a fully functional large scale commercial web application. Nevertheless Amazon services are also known to be used for research and private purposes due to relatively low costs in comparison to constructing a data center from scratch. Thanks to the relatively fast and simple availability of computational time new possibilities arise. With reference to earlier mentioned MapReduce it is possible to run Hadoop instances on Amazon EC2[25] server that will operate on S3 storage[26]. Therefore both services can be used for some occasional needs to process large quantities of data or in some particular research projects that need to cope with big amount of evaluation data to test some of the research results. In comparison to Google Apps, Amazon service has similar principals but offers a lot more broad choice of technologies for development of applications (a number different server side technologies both for application development and data storage). An interesting use of Amazon technology is TimeMachine implemented by New York times to browse publicly available newspaper scans[26].

Recently released Microsoft Azure[28] is a service similar to the aforementioned Google AppEngine and Amazon EC2. Due to the early stage it is still hard to say how popular the service will become. Nevertheless the example of Microsoft shows the how the leading companies in various Information Technology branches invest in infrastructures for such type of web application development. In comparison to its competitors Microsoft has an advantage of wide range of products and long history of developer tools market. Along with the release of the

1 http://www.google.com/analytics/
2 http://earth.google.com/
3 http://www.google.com/psearch
4 http://www.amazon.com/

platform Microsoft provided support for applications developers within the Visual Studio framework.

Other interesting use of cloud computing in business is connected to software-as-an-service model (SaaS). The leading provider in the area Salesforce.com[29], contrary to Google and other vendors of cloud computing infrastructures does not deliver a platform for application developers. Its target group are companies that need read-to-use solutions. Salesforce.com offers to develop and maintain highly scalable applications. Although currently the company became so successful that is invests in new types of products that enable increasing involvement of external developers, the core services remain in the Customer Relationship Management (CRM) area.

It has to be noted that apart of services that we categorize as cloud computing a number of large and popular internet portals also use the very same technologies to cope with their efficiency problems. Apart of the aforementioned YouTube(see Sec. 1), MySpace is disclosed to be a customer of Aster Data Systems and is using their large scale data warehousing product with MapReduce implementation called nCluster[14].

V. Conclusions

Analyzing the software models presented a question arises. When to use those solutions? Where is the border line for regular multiprocessor or multicore machines becoming insufficient in terms of efficiency and DISC systems starting to have advantage? Observing the given examples one can notice that the response to that question is very individual. With some specific types of data and operations to perform even small clusters can prove to be a good solution. Nevertheless, when creating a cluster for a DISC system it has to be kept in mind that the benefits come at a price. Maintaining such a system and even the initial configuration are sophisticated tasks. Information about those technologies is shared by companies but only to a certain extent. Even the described open-source implementations are only a small part of a bigger picture that developers let out to the public. Therefore setting up a system that would, in practice, successfully use those tools is hard task.

References

[1] K. Cordes: YouTube scalability Talk (July 14, 2007), http://ylecordes.com/2007/07/12/youtube-scalability/

[2] R. E. Bryant, Data-Intensive Supercomputing: The case for DISC. Carnegie Mellon University School of Computer Science Tech Report CMU-CS-07-128. May 10, 2007.

[3] D. Anderson, D. Werthimer, J. Cobb, E. Korpela, M. Lebofsky, D. Gedye, and W. Sullivan, ``SETI@home-Internet Distributed Computing for SETI," Bioastronomy '99 - A New Era in the Search for Life in the Universe, ed. G. Lemarchand and K. Meech, ASP Conf. Series 213, 511, 2000.

[4] E. A. Brewer. “Delivering high availability for Inktomi search engines.” In L. M. Haas and A. Tiwary, editors, ACM SIGMOD International Conference on Management of Data, page 538. ACM, 1998.

[5] J. Dean and S. Ghemawat. “MapReduce: Simplified data processing on large clusters.” In Operating Systems Design and Implementation, 2004.

[6] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. BigTable: A distributed storage system for structured data. In Operating Systems Design and Implementation, 2006.

[7] S. Ghemawat, H. Gobioff, and S. T. Leung. The Google file system. In Symposium on Operating Systems Principles, pages 29–43. ACM, 2003.

[8] Apache Hadoop homepage, http://hadoop.apache.org/

[9] List of companies and research units that use Hadoop, http://wiki.apache.org/hadoop/PoweredBy

[10] Yahoo! Developer Network Blog: Yahoo! Launches World's Largest Hadoop Production Application,February 19, 2008), http://developer.yahoo.net/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html

[11] HDFS architecture overview, http://hadoop.apache.org/core/docs/current/hdfs_design.html

[12] HBase homepage, http://hadoop.apache.org/hbase/

[13] Greenplum MapReduce, http://www.greenplum.com/resources/mapreduce/

[14] Aster Data homepage, http://www.asterdata.com/

[15] GridGain homepage, http://www.gridgain.com/

[16] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, "Evaluating MapReduce for Multi-core and Multiprocessor Systems", Proceedings of the 13th Intl. Symposium on High-Performance Computer Architecture (HPCA), Phoenix, AZ, February 2007

[17] MapReduce on Cell, http://sourceforge.net/projects/mapreduce-cell

[18] Mars: A MapReduce Framework on Graphics Processors, http://www.cse.ust.hk/gpuqp/Mars.html

[19] Skynet: A Ruby MapReduce Framework homepage, http://skynet.rubyforge.org/

[20] Disco project homepage, http://discoproject.org/

[21] Holumbus project homepage, http://holumbus.fh-wedel.de/trac

[22] FileMap: File-Based Map-Reduce, http://mfisk.github.com/filemap/

[23] CouchDB Project homepage, http://couchdb.apache.org/

[24] Google AppEngine homepage, http://code.google.com/appengine/

[25] Amazon Elastic Compute Cloud homepage, http://aws.amazon.com/ec2/

[26] Amazon Simple Storage Service homepage, http://aws.amazon.com/s3/

[27] New York Times Time Machine, http://open.blogs.nytimes.com/2008/05/21/the-new-york-times-archives-amazon-web-services-timesmachine/

[28] Windows Azure homepage, http://www.microsoft.com/azure/

[29] Salesforce.com main homepage, http://www.salesforce.com/