

# CASTOR: A Distributed Storage Resource Facility for High Performance Data Processing at CERN

Giuseppe Lo Presti, Olof Barring, Alasdair Earl, Rosa María García Rioja, Sébastien Ponce, Giulia Taurelli, Dennis Waldron, Miguel Coelho Dos Santos  
CERN, European Laboratory for Particle Physics, CH-1211 Geneva 23, Switzerland  
{giuseppe.lopresti, olof.barring, alasdair.earl, rosa.garcia, sebastien.ponce, giulia.taurelli, dennis.waldron, miguel.coelho.santos}@cern.ch

## Abstract

Mass storage systems at CERN have evolved over time to meet growing requirements, in terms of both scalability and fault resiliency. The CERN Advanced STORAge system (CASTOR) and its new disk cache management layer (CASTOR2) have been developed to meet the challenges raised by the experiments using the new accelerator that CERN is building: the Large Hadron Collider (LHC) [4]. This system must be able to cope with hundreds of millions of files, tens of petabytes of storage and handle a constant throughput of several gigabytes per second. In this paper, we detail CASTOR's architecture and implementation and present some operational aspects. We finally list the performance levels achieved by the current version both in a production environment and during internal tests.

## 1. Introduction

The Cern Advanced STORAge manager (CASTOR) is a modular Hierarchical Storage Management (HSM) system designed to handle tens of millions of files (with sizes in the megabyte to gigabyte range) which result in an aggregated storage capacity of tens of petabytes of tape archive and petabytes of disk storage.

CASTOR builds on SHIFT (Scalable Heterogeneous Integrated FaciliTy) system, which provided users with access to the CERN tape system but had limited scalability, up to approximately 10,000 files and 10MB/sec of data throughput. The first version of CASTOR had a greater scalability than SHIFT by several orders of magnitude, handling several million files on tape and 200,000 files in its disk cache. Although this proved adequate for LEP era computing it could not provide the performance or scalability necessary for LHC era computing requirements [6].

The CASTOR2 project started in 2003 and aimed at pro-

viding the data storage capacity and performance for managing the data produced by the LHC experiments. These data will be processed globally using the LHC Computing Grid [8].

The main data store, called Tier 0, needs to store all data coming from the LHC experiments, i.e. ATLAS, ALICE, CMS and LHCb, and to run the initial data reconstruction. CASTOR provides a *Central Data Recording* facility (CDR) and storage for the associated *Reconstruction* facilities. CASTOR handles data transfers to the Tier 1 sites, where the data are replicated and further reconstruction and analysis take place. Finally, many Tier 2 sites are linked to each Tier 1 and act as data customers for the physics analysis data. Figure 1 gives a graphical view of the different concurrent activities CASTOR is handling.

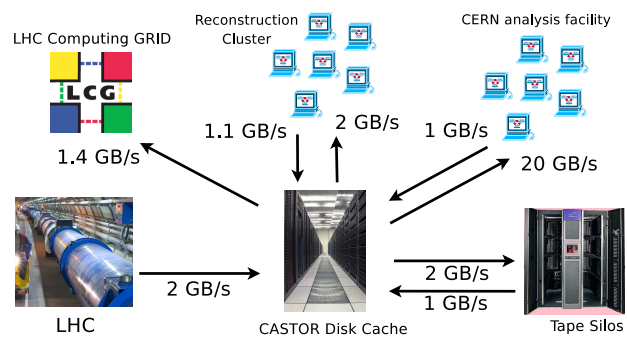
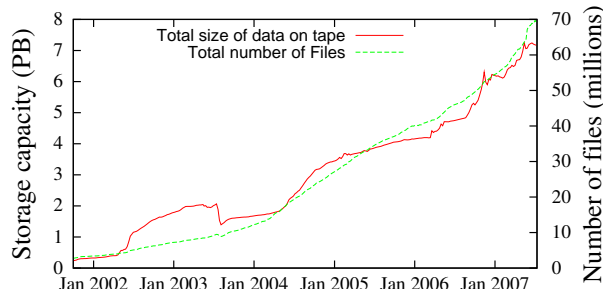


Figure 1. CASTOR data rates

CASTOR 2 was deployed in production at CERN in 2006. It currently (July 2007) manages 73.43 million files, requiring over 7.5 PB of tape and a disk cache of over 1.5 PB. Figure 2 shows how the CASTOR namespace has grown over time at CERN. CASTOR is used as well at several other high energy physics institutes around the world,

notably ASGC<sup>1</sup>, INFN<sup>2</sup>, RAL<sup>3</sup> and IHEP<sup>4</sup>. These institutes have deployed or are in the process of deploying CASTOR 2.



**Figure 2. Time evolution of CASTOR data storage at CERN**

Outside the particle physics domain, CASTOR is used at INFN to store data from astrophysics projects, and at CERN for satellite data from UNOSAT and data from various computer science projects.

In this paper we present the operational aspects of running a CASTOR instance and the performances achieved by the current version, both in a production environment and during internal tests. We conclude with a brief outline of our roadmap for the continued development of the system and the enhancements we plan to provide.

### 1.1. Related Work

Of other storage solutions [1], the High Performance Storage System [2] is the most comparable to CASTOR.

HPSS provides HSM and archive services. It is a joint development by IBM and five US Government laboratories. Its design follows the IEEE Mass Storage Reference Model and allows data to be moved from an intelligent disk or tape controller to the client. All metadata information is kept on a RDBMS-based metadata engine, while movers control the raw data transfer from heterogeneous hardware.

HPSS was used at CERN in late 1990s. It was not adopted for LHC computing because at the time the system was optimized for serial access to large files, but it did not have the performances required for random access. Because it is a collaborative effort, it was not possible to guarantee development would meet our objectives, and an in-house solution was considered preferable.

<sup>1</sup> Academia Sinica Grid Computing, Taipei, Taiwan

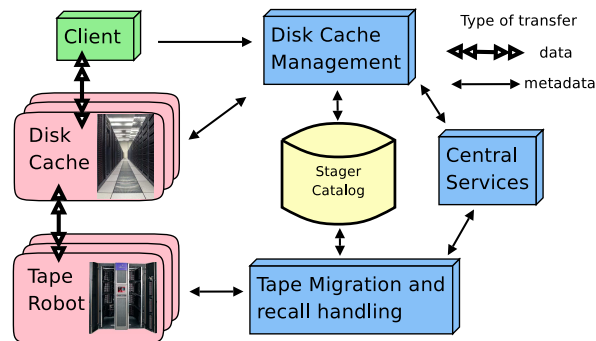
<sup>2</sup> Istituto Nazionale di Fisica Nucleare, Bologna, Italy

<sup>3</sup> Rutherford Appleton Laboratory, Didcot, United Kingdom

<sup>4</sup> Institute for High Energy Physics, Moscow, Russia

## 2. Architecture

The CASTOR system exposes a global hierarchical namespace that allows users to name files in a UNIX like manner. It provides transparent tape storage and automated disk cache management to ensure performance and reliability. Hence, given the requirements depicted in Figure 1, the scalability and reliability needs have a large impact on the architecture. Reliability imposes strong constraints on the consistency of the system state (e.g. the disk cache state), especially in case of crashes. Scalability adds constraints on the size. CASTOR 2 architecture is thus database centric, with a number of stateless daemons, as shown in Figure 3.



**Figure 3. The CASTOR system architecture**

A relational database contains the system state, as well as all requests and their status. All daemons continuously query the database for the next operation to perform, e.g. schedule the next transfer for a client, issue a tape recall or garbage collect some filesystem. This design allows for a number of key features, including easy scalability and better fault tolerance by replicating the daemons for the same component on different machines, and simplified operation by allowing the updating and restarting of daemons while another instance is supporting the load.

With clusters of hundreds of disk servers using thousands of commodity disks, the management of storage resources increasingly faces the same issues and challenges as the management of standard computing resources. Therefore, the CASTOR system is viewed as a Storage Resource Sharing Facility, where load has to be properly scheduled on the available resources. CASTOR benefits from the existing tools by externalizing scheduling activities, as well as most of the decision making processes, where policies can be easily plugged in a variety of scripting languages. Typical examples are migration/recall policies and file replication policy.

A large project like CASTOR needs to be developed according to a robust software process. The UML methodol-

ogy is used at all levels, from describing the workflow to designing the different components. UML modelling has also allowed to use automated code generation for services like a database abstraction layer, and streaming of objects for the interprocess data exchange. Standard design patterns like *Abstract interfaces*, *Pluggable services* and *Factories* are heavily used, allowing for easy code evolution.

The rest of this section deals with the most important CASTOR components, the disk cache layer and the tape archive layer.

## 2.1. Disk Cache Management

In the management of large disk caches, the primary challenge is to define efficient garbage collection policies in order to optimize cache usage and avoid inefficient recalls from tape. However, handling user requests, and especially scheduling their accesses to the disks, is also of primary importance if good bandwidth is to be achieved. This implies proper monitoring of the disk status which must be fed to the scheduling system. All these activities are implemented in CASTOR 2 as separate components.

The disk cache management layer is also responsible for interaction with clients. Clients first interact with the **request handler**, a very light weight gateway that only stores requests in the central database. CASTOR has demonstrated the ability to manage peaks of more than 100 requests per second without service degradation.

In the context of Grid applications, clients may not interact directly with the request handler but rather go through the more generic SRM interface. The SRM specifications have evolved over time as a worldwide effort. As of July 2007 SRM version 1.1 is the production version. The CASTOR implementation has been proved to scale up to 1.7 million requests/day. The new version 2.2 of the SRM interface has been implemented for CASTOR and is being tested. For further details, please refer to [15].

The CASTOR system supports a number of file transfer protocols. These can either be tightly or loosely integrated with CASTOR. They are respectively called internal and external protocols. Internal protocols deal with CASTOR transfer URLs (e.g. `protocol://disk-server:port//castor/...`) and will internally connect to the CASTOR core services to gather metadata information. The transfer will then take place using the native protocol in the CASTOR disk cache. CASTOR internal protocols are RFIO, ROOT [5], XROOT [7], and GridFTP v2 [12]. External protocols do not support CASTOR URLs, nor contact the CASTOR core services directly. For these protocols the client contacts a gateway machine running a modified version of the external protocol daemon. This will request the data from CASTOR using one of the internal protocols (usually RFIO). GridFTPv1

falls into this category.

The **stager** is the daemon responsible for handling user requests. Like all CASTOR daemons, it is stateless and relies on the central database. Different services are defined for handling different types of requests. They are implemented using separate thread pools in order to decouple the different loads. The stager is also responsible for the management of file replication within the disk cache.

The **resource monitoring** daemon collects monitoring information from the available disk servers e.g. CPU load and number of I/O streams. This information is used as input to the I/O scheduling system. An external scheduler is responsible for implementing the resource sharing and scheduling [9] in order to optimize hardware usage and avoid overloading. CASTOR currently supports two different schedulers: MAUI [10], an open source solution for small sized clusters, and LSF [13], a commercial scheduling facility usually used for large CPU farms. At CERN, the size of our resources demands the use of LSF.

The **Nameserver** provides the CASTOR namespace, which presents CASTOR files under the form of a hierarchical filesystem rooted with “castor” followed by the domain, e.g. `/castor/cern.ch` or `/castor/cnaf.infn.it`. The next level in the hierarchy usually identifies the hostname (or alias) of the node with a running instance of the CASTOR Name Server. The naming of all directories below the third level hierarchy is entirely up to the service administrator managing the subtrees.

The **Distributed Logging Facility** (DLF) provides a centralized recording of logging and accounting information from multiple machines and an intuitive and easy-to-use web interface to browse the logging data. This is essential within a distributed architecture to diagnose and understand problems as well as to compute statistics about the system. The DLF framework consists of a central server, which can process thousands of messages a second, and a client-side API, widely used in all CASTOR components. The server stores all the data collected in a relational database.

Garbage collection in the disk cache is mostly implemented as a database job which selects, according to dynamic policies, the files to be removed from cache. An external daemon can then query the database for files to delete and effectively remove them. As an example, typical policies used at CERN includes deletion of all migrated files (Data recording mode) and deletion of old and unused files (analysis facility).

## 2.2. Tape Archive Management

The CASTOR tape archive provides long term availability of data with maximum transparency and flexibility for

the end user. In other words, users should not realize that their data are stored on tape and the data should be automatically migrated to newer media when necessary without any impact on their availability. CASTOR's tape system addresses these issues via a set of components.

The **tape daemon** is the interface to the tape hardware. It interacts with the tape robots to transfers data to/from tape. The tape daemon supports a number of types of tape drives and related robots. The current status of CERN's tape robots and drives is described in Table 1.

Drive type	Nb drives	Speed
STK 9940B	44	30 MB/s
IBM 3592 E05	50	100 MB/s
STK T10000	50	120 MB/s
Library	Nb tapes	Capacity
STK Powderhorn	10,000	2 PB
IBM 3584	5,500	3.85 PB
STK SL8500	13,000	6.5 PB

**Table 1. Status of the tape hardware at CERN**

The **Volume Manager** (VMGR) and the Volume and Drive Queue Manager (VDQM) respectively handle the status of the tapes and tape drives. They handle queues of requests from tapes and drives and act as resource allocators.

The **remote tape copy** (rtcopy) software interacts with the disk layer. It manages a set of migrators and recalls respectively writing and reading data from/to tape. Both migrations and recalls are user driven actions that depend on the disk cache status and are controlled by external policies. They use high speed streams and allow for multiplexing of streams from different disk servers in order to fill the tape buffers. rtbody also takes care of computing a checksum of the files going to, and coming from, tape in order to detect possible tape errors.

Finally, the **repack** component is responsible for moving data from a set of tapes to another set. This allows both to migrate data from one generation of tapes to another and to recuperate space on tapes where a lot of files were deleted.

### 3. Monitoring

Monitoring is an important component of CASTOR. It is used by the operations team, by the user community, and to aid in planning future resource allocation. This section presents a brief overview of the monitoring system, a detailed explanation is available in [11].

The CASTOR monitoring system queries the state of the system, in terms of performance and errors, in a variety of

ways. We make use of DLF to centrally analyze logs from all machines within the system. SQL procedures are used on each instance database to provide status information to a variety of presentations systems.

In addition to the CASTOR specific software, we make use of Computer Centre monitoring tools such as Lemon [3]. For example, we make use of *Lemon actuators* which act as triggers for specific monitoring metrics. When specific conditions are met, actions are automatically taken. These can range from restarting a daemon, to moving a disk server out of production, to sending a GSM SMS to a service manager.

The monitoring of resource utilization provides accounting information that is used for planning the allocation of resources. We record information about the number of requests per second to the stager, the number of files in the system and their status, disk and tape space usage and a cornucopia of other information.

To make ensure system consistency, checking mechanisms are used for configuration and for data consistency. The **castorReconcile** package has been developed to present information from the CASTOR database, the scheduling subsystem, the state management system, and others. It provides a summary of each instance so that all information about a problem machine can be seen in a single place, and the state of all machines in an instance can be seen at a glance. In addition, we have software which periodically scans all the file systems and compare the file metadata on the disk servers with the status described by the stager database to detect deviations. This detects problems, such as orphaned files, missing disk copies, size mismatches and misplaced files. It is planned to add checksums to this to verify the integrity of files.

#### 3.1. Monitoring Evolution

Initially CASTOR monitoring was fully integrated with the monitoring system used at CERN. However, to aid deployment at other sites, since 2006 we have made the software more independent of the monitoring tool used for aggregation, transport and display of the information. To achieve this we run as much monitoring as possible at the database layer, where SQL procedures aggregate information and make it available to various display systems. This feature has increased the monitoring systems flexibility and efficiency, and has also resulted in our sensors becoming predominantly thin clients. These thin clients can easily be developed for other monitoring systems.

### 4. Performance Tests

CASTOR has several performance requirements: they are related to storage capacity, provided bandwidth and ac-

cess pattern. To be able to store all raw data coming from the experiment CASTOR should provide 10 PB of tape storage during the first year and is expected to increase to 30 PB/year.

The disk cache will require 6 PB in the first year and eventually provide 11 PB/year.

From the bandwidth viewpoint, as shown in Figure 1, CASTOR should handle concurrently several streams of data coming from different kind of sources up to a total of 27 GB/s. Additionally, it has to support an extra 2 GB/s out and 1 GB/s in of transfers between tapes and disk cache.

For data recording and export to Tier 1 centers, CASTOR must deal with several tens of 60-100 MB/s streams, being able to simultaneously store this data on disk and tape, but without latency concerns.

A completely different access pattern is the one related to Tier 1 data analysis, where the concurrent streams can be thousands, each reading or writing randomly and at a random speed. In this usage pattern, the average throughput for each stream can be low but, latency should ideally be < 100 ms. To insure that these performance requirements are met and to validate the software, CASTOR has been extensively tested using simulations of real life scenario which are called Data Challenges. Two kinds of simulations can be distinguished: the ones driven by user communities (User Data Challenges) and the internal ones (Internal Data Challenges).

#### 4.1. User Data Challenges

User driven data challenges are scheduled to validate the computing infrastructure for a specific scenario. The scope of these challenges ranges from CASTOR to the experiment specific framework, to the grid layers and to the network infrastructure. Typical scenarios are data taking, event reconstruction and analysis, and export of data to Tier 1 institutes. In all cases, the challenge lasts several weeks and runs on a regular CASTOR production instance with no specific configuration. Usually, this is run in parallel to normal production activities.

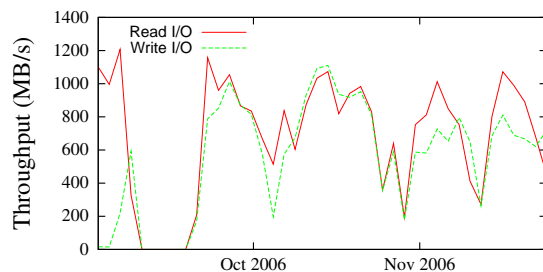


Figure 4. ALICE data challenge

Figure 4 shows an example of the network activity for

the Alice CASTOR production instance between September and December 2006. During this period the ALICE experiment ran an extended data challenge where CASTOR demonstrated that it can sustain an aggregated traffic of the order of 1 GB/sec for more than 2 months. Figure 4 measures inbound and outbound traffic to the disk cache. Inbound traffic includes both new user data and files recalled from tape. Outbound traffic includes both the retrieval of data by the users and the migration of new data to tape.

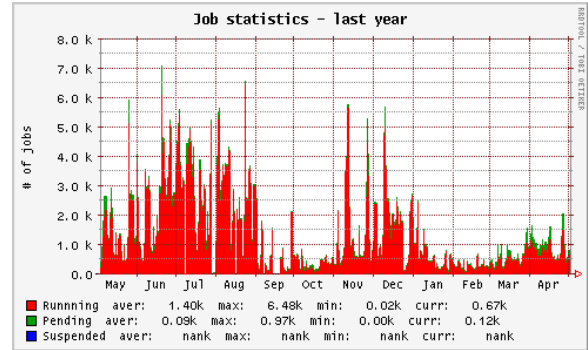


Figure 5. CMS number of running and pending transfers during one year

Figure 5 shows a graph of the number of concurrent I/O transfers taking place on the system for a full year for the CMS experiments production CASTOR setup. When the number of transfers goes higher than the total number of available slots on the disk servers, exceeding requests were properly queued in the scheduling system. In this instance and during other challenges, CASTOR proved able to sustain more than 5000 concurrent transfers and 30 000 pending ones.

#### 4.2. Internal Data Challenges

Internal data challenges are performed on a dedicated CASTOR instance and target specific usage patterns, features, and boundary conditions of the CASTOR software. They are run under the control of the CASTOR development team and allow observation of the system under heavy load so that it can be optimized by tuning its various parameters. The incoming traffic is generated by scripts, which are able to simulate the different activities that are foreseen in the Tier 0. As already introduced in Figure 1, four different concurrent activities are taking place: the raw data transfer from the DAQ (Data acquisition) buffers, the transfer from the Tier 0 buffer to the reconstruction farm, the export to the Tier 1 sites through the Grid, and the tape migration.

Figure 6 shows the outcome of one week of data acquisition, including the migration to tape. For this challenge,

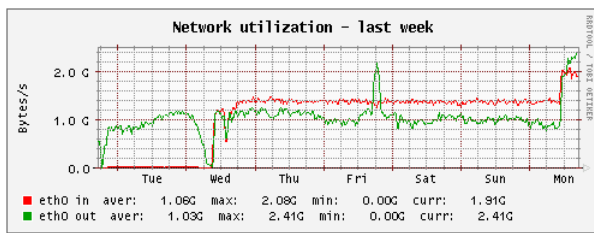


Figure 6. Tier 0 data-taking simulation

100 CPU nodes have been used as sources, and the CASTOR system has been configured with 48 disk servers with a total of 240 TB and 28 tape drives. In this configuration the system sustained an average throughput of 1 GB/s and demonstrated that it could sustain over 2 GB/s for hours, as shown by the last part of the plot.

Disk servers have been tuned at the kernel and the filesystem level to increase the throughput of their RAID (Redundant Array of Inexpensive Disks) subsystems. It has been demonstrated that a standard diskserver, with 24 physical disks mounted on 3 RAID controllers using the XFS filesystems, can sustain concurrent inbound and outbound throughput of 70 MB/s using the CASTOR RFIO protocol.

Modern tape drives can deliver between 100 and 120 MB/s of throughput, according to their specifications. Taking into account mount times and tape marks, this results in a 70 to 90 MB/s available data rate, depending on the average file size. The CASTOR tape system is able to use all this bandwidth. However, within a global context, the disk cache scheduling mechanism imposes further constraints on the data rates. The typical average data writing rate experienced in a CASTOR production environment is in the range of 40 to 60 MB/s. This still provides a safe margin to sustain the expected data taking requirements of the LHC.

## 5. Conclusion

The CASTOR 2 software is the latest evolution of the CERN hierarchical mass storage system for high availability data. CASTOR 2 was brought into production in early 2006 and has already successfully passed several data challenges, showing its ability to sustain constant loads of over 4 GB/s, to store tens of millions of files using more than 7 PB of total space and to serve concurrently more than 5000 file transfers. CASTOR 2 comes with extensive monitoring tools in an extensible framework allowing service managers to easily monitor the system and improve operations.

CASTOR 2 integrates with the Grid framework by implementing an SRM interface. Specification 1.1 has been implemented and is in production, specification 2.2 is currently undergoing tests prior to its deployment [15]. Future

evolutions of the specification will be supported.

Other future developments include improvements of the authentication and authorization scheme via the use of the VOMS (Virtual Organization Membership Service) Grid standard [14]. Better support for disk only data pools (where no tape backup is used), improvements in efficiency of the disk and tape usage and extension of monitoring functionalities to support Grid accounting are also planned.

## Acknowledgment

We would like to thank the following people for their advice and support: Jan Van Eldik, German Cancio Melia, Hugo Monteiro Cacote, Bernd Panzer-Steindel, Arne Wiebalck.

## References

- [1] EDG Review of Grid data systems. <https://edms.cern.ch/document/336677/1.5>.
- [2] IBM High Performance Storage System. <http://www.hpss-collaboration.org/>.
- [3] LEMON - LHC Era Monitoring. <http://cern.ch/lemon>.
- [4] LHC - The Large Hadron Collider. <http://cern.ch/lhc>.
- [5] ROOT, an Object-Oriented Data Analysis Framework. <http://root.cern.ch>.
- [6] The CASTOR project. <http://cern.ch/castor>.
- [7] The Scalla Software Suite: xrootd/olbd. <http://xrootd.slac.stanford.edu>.
- [8] The Worldwide LHC Computing Grid. <http://cern.ch/LCG>.
- [9] O. Bärring, B. Couturier, J.-D. Durand, E. Knezo, and S. Ponce. Storage Resource sharing with CASTOR. In *NASA/IEEE MSST Conference, Proceedings*. IEEE Computer society, April 2004. <http://cern.ch/castor-presentations/2004>.
- [10] Cluster Resources Inc. Maui Cluster Scheduler<sup>TM</sup>. <http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php>.
- [11] A. Earl and M. Coelho Dos Santos. Portable Monitoring for Castor2. In *CHEP Proceedings*, September 2007.
- [12] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. In *Intl J. Supercomputer Applications*, volume 11(2), pages 115–128, 1997.
- [13] Platform Computing. <http://www.platform.com>.
- [14] R. Alfieri, R. Cecchini et al. VOMS, an authorization system for virtual organizations. In F. F. Rivera, M. Bubak, A. G. Tato, and R. Doallo, editors, *European Across Grids Conference*, volume 2970 of *Lecture Notes in Computer Science*, pages 33–40. Springer, 2003.
- [15] SRM collaboration. Storage resource managers: recent international experience on requirements and multiple cooperating implementations. In *IEEE-NASA MSST Conference, Proceedings*. IEEE Computer society, September 2007.