SOFTWARE AGENTS IN DATA AND WORKFLOW MANAGEMENT

T.A. Barrass (University of Bristol, UK)

O. Maroney, S. Metson, D. Newbold (University of Bristol, UK)
W. Jank (CERN, Geneva, Switzerland)
P. Garcia-Abia, J. M. Hernández (CIEMAT, Spain)
A. Afaq, M. Ernst, I. Fisk, Y. Wu (FNAL, Chicago, USA)
C. Charlot, I. Semeniouk (IN2P3 LLR, France)
D. Bonacorsi, A. Fanfani, C. Grandi, N. DeFilippis (INFN CNAF, Bologna, Bari, Italy)
K. Rabbertz, J. Rehn (University of Karlsruhe, Germany)
L. Tuura (Northeastern University, USA)
T. Wildish (Princeton, USA)

Abstract

CMS currently uses a number of tools to transfer data which, taken together, form the basis of a heterogeneous datagrid. The range of tools used, and the directed, rather than optimized nature of CMS recent large scale data challenge required the creation of a simple infrastructure that allowed a range of tools to operate in a complementary way.

The system created comprises a hierarchy of simple processes (named 'agents') that propagate files through a number of transfer states. File locations and some application metadata were stored in POOL file catalogues, with LCG LRC or MySQL back-ends. Agents were assigned limited responsibilities, and were restricted to communicating state in a well-defined, indirect fashion through a central transfer management database. In this way, the task of distributing data was easily divided between different groups for implementation.

The prototype system was developed rapidly, and achieved the required sustained transfer rate of ~10 MBps, with O(10^6) files distributed to 6 sites from CERN. Experience with the system during the data challenge raised issues with underlying technology (MSS write/read, stability of the LRC, maintenance of file catalogues, synchronization of filespaces), all of which have been successfully identified and handled. The development of this prototype infrastructure allows us to plan the evolution of backbone CMS data distribution from a simple hierarchy to a more autonomous, scalable model drawing on emerging agent and grid technology.

DATA DISTRIBUTION FOR CMS

The Compact Muon Solenoid (CMS) experiment at the LHC will produce Petabytes of data a year [1]. This data is then to be distributed to multiple sites which form a hierarchical structure based on available resources: the detector is associated with a Tier 0 site; Tier 1 sites are typically large national computing centres; and Tier 2 sites are Institutes with a more restricted availability of resources and/or services. A core set of Tier 1 sites with large tape, disk and network resources will receive raw and reconstructed data to safeguard against data loss at CERN. Smaller sites, associated with certain analysis groups or Universities, will also subscribe to certain parts of the data. Sites at all levels will be involved in producing Monte Carlo data for comparison with detector data.

At the Tier 0 the raw experiment data undergoes a process called reconstruction in which it is restructured to represent physics objects. This data will be grouped hierarchically by stream and dataset based on physics content, then further subdivided by finer granularity metadata.

There are therefore three main use cases for distribution in CMS. The first can be described as a push with high priority, in which raw data is replicated to tape at Tier 1s. The second is a subscription pull, where a site subscribes to all data in a given set and data is transferred as it is produced. This use case corresponds to a site registering an interest in the data produced by an ongoing Monte Carlo simulation. The third is a random pull, where a site or individual physicist just wishes to replicate an extant dataset in a one-off transfer.

Although these use cases are here discussed in terms of push and pull these can be slightly misleading descriptions. The key point is the effective handover of responsibility for replication between distribution components; for example, it is necessary to determine whether a replica has been created safely in a Tier 1 tape store before being able to delete it from a buffer at the source. This handover is enabled with well-defined handshakes or exchanges of state messages between distribution components.

The conceptual basis of data distribution for CMS is then distribution through a hierarchy of sites, with smaller sites associating themselves to larger by subscribing to some subset of the data stored at the larger site.

The management of this data poses two overall problems. The first problem is that sustained transfers at the 100+ MBps estimated for CMS alone are currently only approached by existing experiments. The second problem is one of managing the logisitics of subscription transfer based on metadata at granularities between high level stream location and low level bulk staging requests across multiple mass storage platforms.

Managing large data flows

The grid tools currently in use by CMS do not scale to the management of data-flows of this size. At present point-to-point transfers require significant manual triggering, intervention and failure recovery. Managing the transfer of many millions of files from and to multiple sites is currently close to impossible without significant manpower cost. It is a solution to the second of these problems that we discuss here.

A scalable architecture

A combined blackboard and multi-agent architecture [2] allows CMS to automate a succession of point-to-point transfers throughout its distribution network. CMS' agents are focused, persistent processes like daemons. Although algorithmically they are quite sophisticated, incorporating robust backoff, failover and recovery mechanisms, they are all intended to handle only specific tasks, for example copying a file from A to B or allocating new files to a set of subscribed destinations (see Fig. 1.).

At present these agents do not operate within an advanced agent framework like Diamonds [3], for example, although they draw heavily on such systems. Instead, agents are typically coded in Perl, which allows rapid prototyping of system logic and workflow testing. Note that the use of a sophisticated agent framework is seen as a reasonable future step.

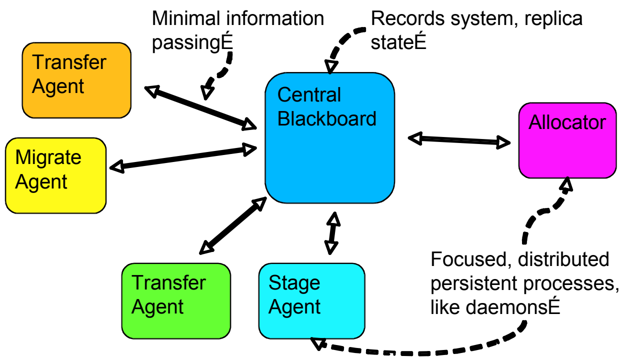
Minimal information passingÉ
Records system, replica stateÉ
Transfer Agent
Central Blackboard
Allocator
Migrate Agent
Transfer Agent
Stage Agent
Focused, distributed persistent processes, like daemonsÉ

Fig. 1. CMS distribution relies on a scalable architecture of distributed processes communicating through a shared blackboard.

There is no intra-agent communication. Instead, all agents post information to a central blackboard service, which acts as a message board containing global system state. Traditional AI blackboard systems include control structures that trigger certain external component behaviours. In CMS' case there is no such control structure- instead, each agent is autonomous and has the responsibility of reading and posting relevant information entirely asynchronously. Agent functionality can be defined solely in terms of messages passed to and read from the blackboard, allowing multiple developers to produce prototypes, in parallel if necessary to manage site-specific issues. We have found that this is not a waste of resources as agent code is typically only several hundred lines of script.

Grid context

The CMS distribution architecture complements current grid tools, forming a layer at which large scale data-flows can be managed. It separates high-level replica management components from low-level replication tools. It gives the replica managers a more abstract interface to the experiment's dataset, allowing them to manage transfers of data at any granularity. It also handles issues of staging and space reservation that are closely coupled to the underlying replication tools.

This approach is in contrast to some existing grid tools, for example the LCG-2 replica manager tools [4], where an overall replica management component couples point-to-point transfer to catalogue operations without any sense of data clustering.

DATA DISTRIBUTION DURING A RECENT DATA CHALLENGE

In the first quarter of 2004 CMS undertook a large-scale data challenge with the aim of simulating the production and distribution of reconstructed data at 25% of the full start-up rate, or 25 events per second [5].

An Oracle database located at CERN and named the Transfer Management Database (TMDB) was used as the central blackboard. The operation of a number of agents was defined in advance as simple SQL interactions with the blackboard, with the aggregate of these operations forming the distribution workflow. The agents were then implemented in a variety of languages, including Bash script, Perl, C and Java, determined by local experience. In this way very lightweight but effective prototypes were rapidly built and customised for site-specific operations.

A distribution chain with a star shaped topology was used to propagate replicas to 6 Tier 1s and multiple associated Tier 2s in the USA, France, UK, Germany, Spain and Italy. Three different transfer tools were used: SRB [6], SRM [7] and the LCG-2 Replica Manager [4]. The data challenge is discussed in detail by Bonacorsi et al [8], and Fanfani et al [9].

A series of "export buffers" at CERN were used as staging posts to inject data into the domain of each transfer tool. Tier 1 sites then replicated files, migrated them to tape and made them available to associated Tier 2s.

Distribution workflow

Distribution for CMS consists of a series of handshakes by distribution components. During each handshake each component completes a task and notifies the next component in distribution of the chain (see Fig. 2).

During DC04, distribution of a given file was initiated when a reconstruction task dropped summary information into a dropbox managed by the first in a chain of agents. The summary information comprised a small XML catalogue fragment containing local PFNs, and file attributes like checksum and filesize information.

The injection agents then issued a stage request for the file and published information to a global LCG-2 LRC and RMC [7], and to the TMDB.

A global allocator agent noticed that new files had been made available, and allocated those files to destination sites based on predefined mappings. The allocations took the form of an advert for the destination site, indicating the file state as “available”.

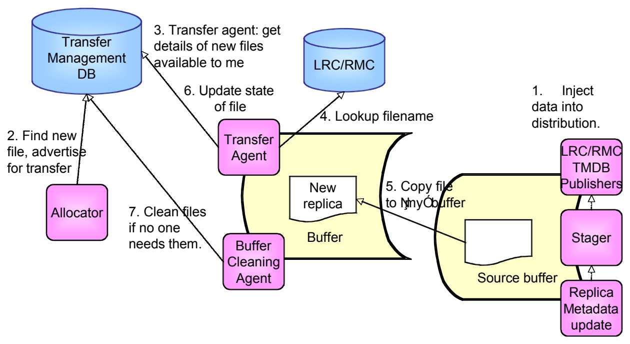

Fig. 2. A distribution chain is built from the action of a series of agents that transfer responsibility for sections of a given transfer by handshaking.

Agents at each of the export buffers were aware of which Tier 1 sites were associated with them. They scanned the TMDB, looking for files advertised as available for “their” Tier 1s. On finding files they initiated a transfer from stage disk at CERN to their local buffer. They then updated the state of the file, indicating a successful transfer to the export buffer, thus effectively advertising it to agents running at the Tier 1 sites.

The Tier 1 transfer agents initiated transfers of available files from an export buffer, and then updated the file state in the TMDB advert to indicate their presence at the Tier 1. Migration agents at the Tier 1 then handled the migration of the files to tape before updating the file state to “safe”.

The final update to a “safe” state was a critical step for the system. It enabled the system manager to see how many files had been successfully “pushed” to tape at the Tier 1, and also enabled the cleaning of unnecessary replicas further up the chain, meaning that export buffers could be purged.

In some cases further transfers to associated Tier 2 sites were also facilitated with a similar chain of agents and a local, independent TMDB.

Experience gained during DC04

During DC04 ~0.5 million files were generated. Each file was tagged with 9 items of replica metadata and was typically replicated between 5 and 10 times. The final number of replicas in the LRC at the end of DC04 was ~3.5 million, ~6 TB of data.

It was found that the information posted in the TMDB was sufficient to trigger automatic analysis of data on local Grid resources on arrival. The PIC Tier 1 was able to have the results of a simple analysis available after a median period of only 20 minutes after the source data was made available for distribution.

There were two key problems experienced during DC04: overload of the central tape stager at CERN and access to single-point file catalogues. In both cases problems were compounded by the fact that the reconstruction farm produced very small files: 0.5 MB in size on average.

Catalogue access during DC04 meant access to a global single instance of an EDG LRC and RMC coupled to form an LCG POOL catalogue, and a single instance of an SRB V2 MCat. The performance of both catalogues was unsatisfactory. In the first case the catalogue stopped responding as requests ramped up, reaching ~100,000 queries a day to the LRC, and 3 times that to the RMC [9,10]. Catalogue query performance was also poor: a query for a PFN based on guid metadata typically took several seconds. In contrast it was possible to dump the entire catalogue from the Oracle backend directly to a file in ~ 2 minutes. The LRC/RMC problems stemmed from deployment issues, principally the implementation as a Java webservice and the necessity of joining across two separate databases. During DC04 the backend database was indexed and CMS worked closely with the EDG to develop a more performant catalogue, although the development was only complete as DC04 finished.

Access to the SRB MCat was also problematic: although the backend database was available, some unresolved problem related to local configuration, thought to be causing a fragmentation of memory under high loads, meant that connectivity between the database and clients was low. As a result the majority of SRB transfers under high load timed out.

Tape staging problems with Castor [11] at CERN stemmed from Castor’s current lack of internal throttling mechanism. The large number of unmanaged requests made of the stager decreased stager performance dramatically, so that transfers typically timed out. In addition CMS experienced many stage misses due to downtime of other components: although a single file transfer chain could be complete in seconds, problems with either catalogues, disk servers or software meant that there could be a latency of days between files becoming available and the first transfer. In this case the files were migrated before they could be transferred. CMS developed a sophisticated pinning mechanism that allowed the prioritisation of file migrations based on experiment policy.

PHEDEX

PhEDEx [12] (for Physics Experiment Data Export) is a project born of CMS' experience during DC04. It retains the same architecture, relying on a central blackboard to enable the exchange of information between a series of distributed agents. The principal current aim of PhEDEx is to incorporate the second CMS distribution use case, that of subscription pull of Monte Carlo data.

Subscription pull of Monte Carlo for CMS requires that many more sources of data must be accommodated within the system. To enable the addition of multiple data sources we borrow from established internet technology: where routes from single source to multiple destinations were hard coded into agents before, in PhEDEx nodes in the distribution chain act as routers which share route information using an implementation of the RIP2 [13] algorithm. Any node in the network can act as a source of data, and a route from any node to any other node in the chain can be determined.

More sophisticated resource management

Agent handshaking has become more sophisticated in light of experience during DC04. Staging is no longer part of the data injection process. Instead dedicated agents manage stage and migration at each distribution node. The PhEDEx transfer workflow is then slightly modified from DC04 distribution:

In addition to a single allocator agent PhEDEx has a single file routing agent, whose task is to manage the propagation of files from source to destination. It continually scans currently allocated guid-destination pairs. For each pair it determines the closest replica to the destination. It then posts a pending transfer between the closest-replica node and the next hop node toward the destination.

A transfer agent at the next hop node manages a pool of pending transfers. Each guid in this pool is marked wanted by posting information in the TMDB. The transition to a wanted state indirectly triggers a stage request at the source node, managed by a migration agent. When the file is staged the migration agent marks the file as available in the TMDB, and the transfer agent completes the transfer.

By making the handshake between agents more sophisticated we are able to more effectively manage the disparate resources available at CMS Tier 1 sites.

Current work

The PhEDEx project is currently deploying software to manage transfers between CERN and Tier 1 sites in Spain, Italy, the USA, Germany, France, Taiwan and the UK. In addition a number of Tier 2 sites are also being added, each associated with a Tier 1. Its current task is to help supply 10 M Monte Carlo events per month to physicists around the world in preparation for the CMS Physics TDR in 2005.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the experience and support of the CERN IT Database group in refining the TMDB structure and collaborating to improve the behaviour of replica management components.

REFERENCES

[1] The CMS Technical Proposal,
http://cmsinfo.cern.ch/TP/TP.html

[2] D. Corkhill, "Collaborating Software: Blackboard and Multiagent systems & the future" Proceedings of the International Lisp Conference, October 2003

[3] M. Aamir Shafi, M. Riaz, S. Kiani, A. Shehzad, U. Farooq, A. Ali, I. C. LeGrand, H. B. Newman, "Diamonds- Distributed Agents for Mobile and Dynamic Services", CHEP'03, La Jolla, California, USA March 2003

[4] The LHC Computing Grid,
http://lcg.web.cern.ch/lcg/

[5] C. Grandi, on behalf of the CMS Data Challenge group, "CMS Data Challenge 2004", CMS NOTE, in preparation

[6] A. Rajasekar, et al, "Storage Resource Broker - Managing Distributed Data in a Grid", Computer Society of India Journal, Special Issue on SAN, Vol. 33, No. 4, pp. 42-54 Oct 2003.

[7] The Storage Resource Management Working Group,
http://sdm.lbl.gov/srm-wg/

[8] D. Bonacorsi et al, "Role of Tier-0, Tier-1 and Tier-2 Regional Centers during CMS DC04", CHEP'04, Interlaken, Switzerland, September 2004

[9] A. Fanfani et al, "Distributed Computing Grid Experiences in CMS DC04", CHEP'04, Interlaken, Switzerland, September 2004

[10] M. Girone et al, "Experience with POOL from the First Three Data Challenges using the LCG"

[11] Castor Mass Storage,
http://www.cern.ch/castor

[12] CMS PhEDEx,
http://www.cern.ch/cms-project-phedex

[13] G. Malkin, Bay Networks, "The Routing Internet Protocol version 2", Internet RFC2453