Scientific Programming 14 (2006) 141–150
IOS Press

Regular Paper

On-the-fly XMM-Newton spacecraft data reduction on the Grid[1]

A. Ibarra[a,*], D. Tapiador[a], E. Huedo[b], R.S. Montero[b], C. Gabriel[a], C. Arviset[a] and I.M. Llorente[b]
[a]*European Space Astronomy Centre, ESAC-ESA, Apartado 50727, E-28080 Madrid, Spain*
[b]*Departamento de Arquitectura de Computadores y Automática*
Universidad Complutense, Madrid, Spain

Abstract. We present the results of the first prototype of a XMM-Newton pipeline processing task, parallelized at a CCD level, which can be run in a Grid system. By using the Grid*W*ay application and the XMM-Newton Science Archive system, the processing of the XMM-Newton data is distributed across the Virtual Organization (VO) constituted by three different research centres: ESAC (European Space Astronomy Centre), ESTEC (the European Space research and TEchnology Centre) and UCM (Complutense University of Madrid). The proposed application workflow adjusts well to the Grid environment, making use of the massive parallel resources in a flexible and adaptive fashion.

1. Introduction

The advent of Virtual Observatories [11] will allow the astronomers around the world access to an unprecedented amount of scientific data. The processing of these data will be carried out, in most cases, by using the computational resources supplied by the Data Provider to the scientific community.

Due to the large amount of data available, it is crucial that the Data Providers manage their hardware resources in an optimal way, if they do not want the data processing to slow down. In this context, Grid technology offers the capability of managing not only the user queries retrieving data from the archive, but also the online processing of that data.

Some other efforts about coupling Grid technology to data archives have been performed. For example, the distributed generation of NASA Earth Science data products [1] provides an exceptionally rich environment for the development of custom data products that have more value to users than the original data has. Also, Virtual Data systems [12] are being developed to allow users to describe data product derivations in declarative terms, discover such definitions and assemble workflows based on them, and execute the resulting workflows on distributed Grid resources. Even more, the application of Grid technology to build Virtual Observatories is not new [10].

This work is focused on the XMM-Newton [7] data. In particular, we have worked with the data collected by one of the cameras (EPIC-pn [8] camera) on-board the satellite. XMM-Newton is a major X-ray observatory of the

*Corresponding author. Tel.: +34 91 813 13 23; Fax: +34 91 813 13 22; E-mail: Aitor.Ibarra@sciops.esa.int.
[1]This research was supported by ESA Member States, by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through the research grant TIC2003-01321.

ISSN 1058-9244/06/$17.00

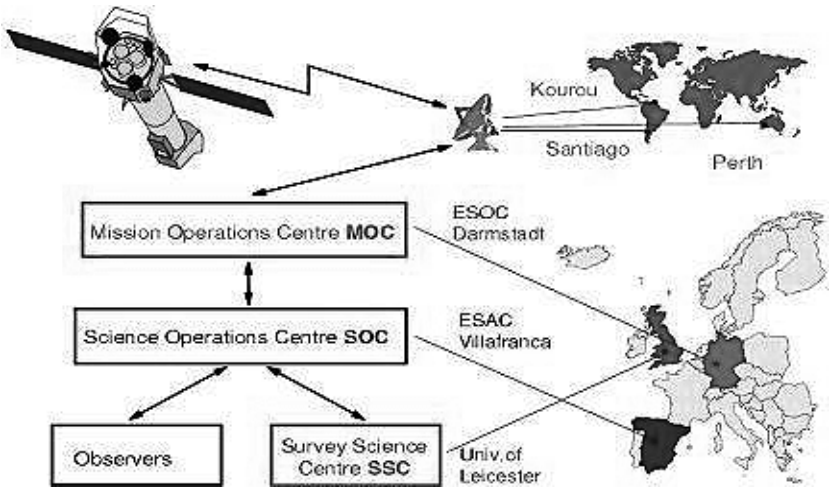


Fig. 1. The XMM-Newton Ground Segment.

European Space Agency (ESA). All the data taken by this satellite are kept in the XMM-Newton Science Archive (XSA). The scientific data corresponding to a particular observation belong, for a certain period of time, to the astronomer who requested that observation. After this period (as a rule one year) the data become public and the whole scientific community can use it through the XSA.

In this paper we propose a Grid architecture where the scientific community can process the XMM-Newton data on-the-fly[1], by using the XSA facilities and the latest version of the XMM-Newton Science Analysis Software, *SAS* [3]. Beneath the XSA, the Grid*W*ay tool controls the available computational resources in order to optimize the data retrieval carried out by the users.

In order to optimize the XMM-Newton data processing, we have also parallelized, at a CCD level (see Fig. 3), the SAS task in charge of the data reduction of the EPIC-pn camera. This X-ray camera is composed of 12 CCD, and thanks to this new parallelized SAS task, the data reduction of each CCD can be performed in distinct Grid resources.

In the next section the XMM-Newton satellite observatory and its associated ground segment is briefly described. In Sections 3, 4 and 5, a detailed description of the infrastructure used during this work is presented. In Section 6 the results obtained from the XMM-Newton data reduction is shown. Finally, in Section 7 the results are analyzed and discussed, and a critical analysis of the advantages and disadvantages of our approach is given.

2. The XMM-Newton Observatory and Ground Segment

XMM-Newton is the most sensitive X-ray satellite ever built and, with a total length of 10 metres, the largest satellite ever launched by ESA. It has been operating as an open observatory since the beginning of 2000, providing X-ray scientific data through three imaging cameras and two spectrometers, as well as visible and UV images through an optical telescope. The large quantity of data collected by XMM-Newton derives from its unprecedented effective area in the X-ray domain in combination with the simultaneous operation of all its instruments.

XMM-Newton carries three X-ray imaging CCD cameras: the European Photon Imaging Cameras (EPIC) [8], each of them in the focal plane of one of the X-ray telescopes. The cameras are of two different types, one of them using a new type of CCD (pn) especially developed for X-ray missions. Since all three work in single-photon register mode, and can register also energy and arrival time of each incoming X-ray photon, they provide simultaneously moderate spectroscopic and timing capabilities.

[1] Archive Inter-Operability Tool for XMM-Newton: http:// xsa.vilspa.esa.es:8080/aio/doc/index.html.

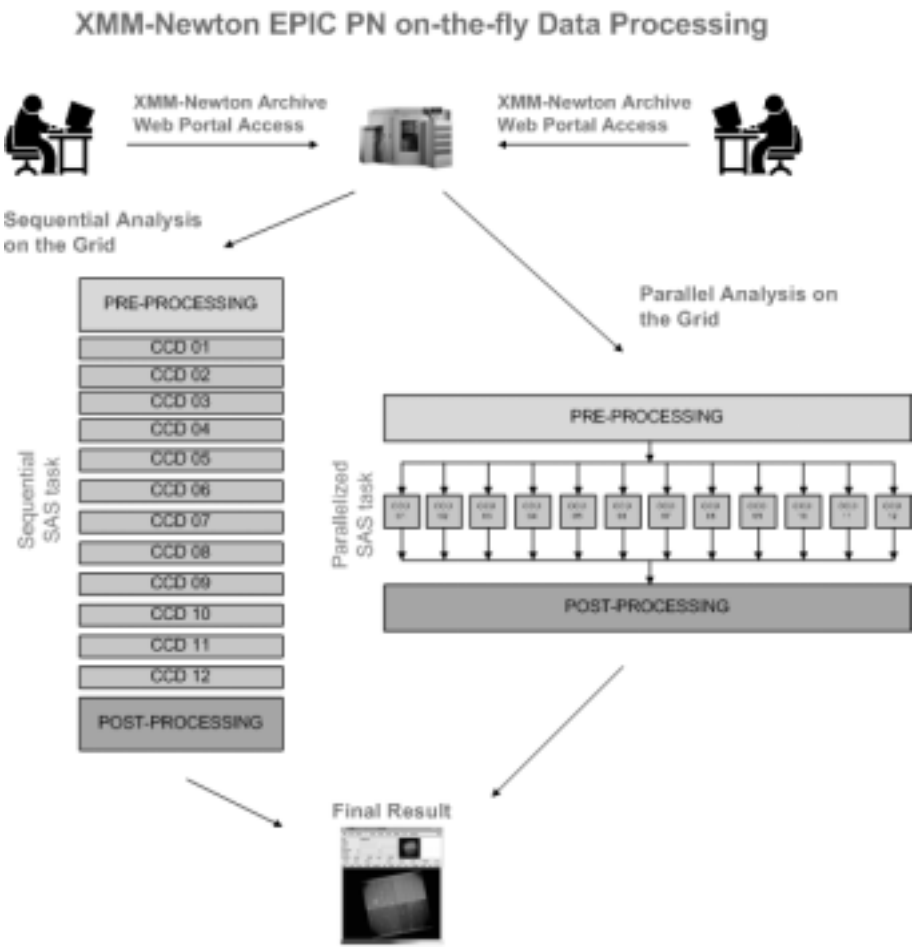


Fig. 2. Testbed Grid infrastructure.

As a typical example of international cooperation characteristic of ESA's approach, the XMM-Newton Ground Segment is distributed among different sites across Europe, as shown in Fig 1. The operations are centrally conducted from ESA's European Space Astronomy Centre (ESAC) of ESA, in Villafranca del Castillo, Spain, where the XMM-Newton Science Operations Centre (SOC) is located. Operational control and commanding of the on-board instruments is maintained from this site. Spacecraft control and overall commanding is performed from ESA's European Space Operations Centre (ESOC) in Darmstadt, Germany, through antennas located in Kourou (French Guyana), Perth (Australia) and Santiago (Chile), which ensure permanent communication with the satellite during the scientifically useful part of its orbit.

The Survey Science Centre (SSC), is responsible for the pipeline processing of all the XMM-Newton data. The XMM-Newton SOC is responsible for the planning and execution of the observations proposed by the observers. In addition to these instrument operational tasks the SOC is deeply committed to maintaining and developing the software used by the observer for the XMM-Newton data reduction.

The Scientific Analysis System (SAS) [3] is a software suite for the interactive analysis of all the XMM-Newton data, making possible the tailoring of the data reduction to the scientific goal. In addition it makes possible a re-calibration of the data whenever new calibration files have been released.

SAS is freely distributed and it is 100% based on free software. In order to cover a user community as broad as possible, SAS is distributed and maintained (including full user support) to run on many different operating systems (different flavors of Solaris, Linux, DEC and MacOS). Although SAS is considered a fully mature package, a large number of people at the SOC and in the SSC Consortium are still working to improve it.

Table 1
Characteristics of the machines in the testbed

Name	Site	Nodes	Processors	Speed	Mem.	DRMS
xgrid	ESAC	4	Intel P4	2.8 GHz	1 GB	SGE6
scigrid	ESAC	18	Intel Xeon	3 GHz	2 GB	SGE6
draco	UCM	1	Intel P4	3.2 GHz	512 MB	fork
ursa	UCM	1	Intel P4	3.2 GHz	512 MB	fork
hydrus	UCM	4	Intel P4	2.5 GHz	512 MB	PBS
esagrid11	ESTEC	10	Intel PII	500 MHz	128 MB	SGE5.3

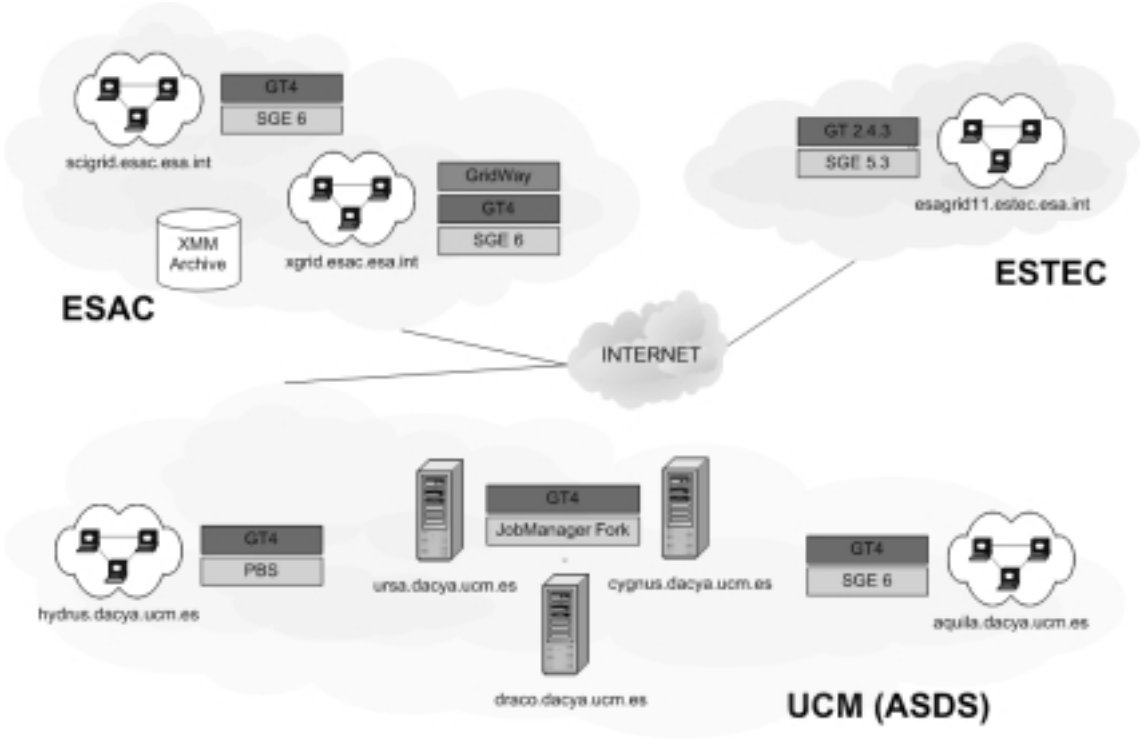


Fig. 3. XMM-Newton EPIC-pn on-the-fly data processing workflows.

3. Grid architecture

The layered architecture used, is the most common in Grid environments. At the bottom of these layers, we have the local resources (local schedulers or job managers). The Grid middleware sits over these local schedulers and provides uniform access to these heterogeneous job managers. This middleware can be used to move data across the Grid, to get information about its state and configuration and to send tasks to a specific node (with a general syntax and without having to care about the underlying scheduler). The Grid middleware chosen has been the Globus Toolkit (the "de facto" standard in Grid computing). As explained above, it is rather complex to use the services provided directly by Globus Toolkit, so Grid*W*ay meta-scheduler has been used to take care of data transfer and task execution along the Grid.

The Virtual Organization infrastructure is shown in Fig. 2. We can see in this picture that different organizations collaborate by sharing some of their resources whose characteristics are specified in Table 1.

The results from each job submitted to the Grid are retrieved to the host where Grid*W*ay is running, and if these data are needed by later tasks, they will be transferred again to the remote node. One possible optimization could be to keep the result data on the host where they were produced and return a reference to the host where Grid*W*ay is running, so later tasks will get the data needed by using this reference.

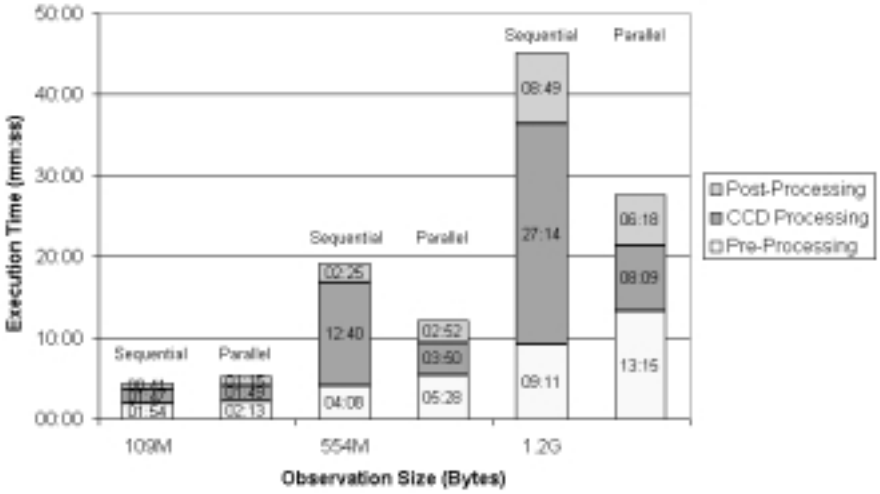


Fig. 4. Sequential and parallel execution on the Grid.

When running simple tests, the hosts chosen by Grid*W*ay meta-scheduler are those belonging to both ESAC and UCM because they are the best resources when taking into account variables such as CPU power and network bandwidth. However, ESTEC nodes remains as infrastructure that can be useful to cope with load peaks that may appear.

4. Grid*W*ay meta-scheduler

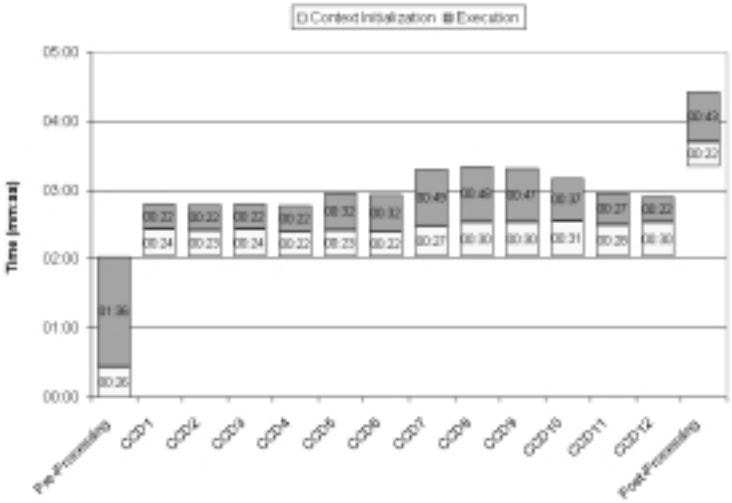
The Grid*W*ay meta-scheduler [5] provides the following techniques to allow the user to efficiently execute of jobs in, Globus-based, heterogeneous and dynamic grids:

- Given the dynamic characteristics of a grid, Grid*W*ay periodically adapts the schedule to the available resources and their characteristics [4]. Grid*W*ay incorporates a *resource selector* that reflects the applications demands, in terms of requirements and preferences, and the dynamic characteristics of Grid resources.
- Grid*W*ay also provides adaptive job execution to migrate running applications to more suitable resources [6]. Hence it improves the application performance by adapting it to the dynamic availability, capacity and cost of Grid resources. Moreover, an application can migrate to a new resource to satisfy new requirements or preferences.
- Grid*W*ay also provides the application with fault tolerance capabilities by capturing Globus Resource Allocation Manager (GRAM) callbacks, by periodically probing the GRAM job-manager, and by inspecting the exit codes of each job.

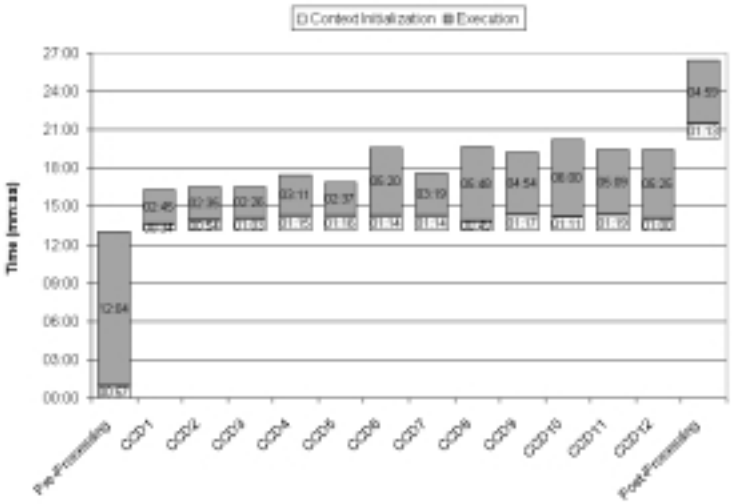
Once a resource is selected to run the job, its execution is performed in three steps:

1. *prolog*: Preparation of the remote system by creating an experiment directory and transferring the input files from the client.
2. *wrapper*: Execution of the actual job, and propagations of the exit status code.
3. *epilog*: Finalization of the remote system by transferring output files back to the client and cleaning the experiment directory.

This way, Grid*W*ay doesn't rely on the underlying middleware to perform context initialization tasks. Moreover, since both *prolog* and *epilog* are submitted to the front-end node of a cluster and *wrapper* is submitted to a compute node, Grid*W*ay doesn't require any middleware installation nor external network connectivity in the compute nodes, and only Globus basic services in the front-end node.



(a) Small observation



(b) Large observation

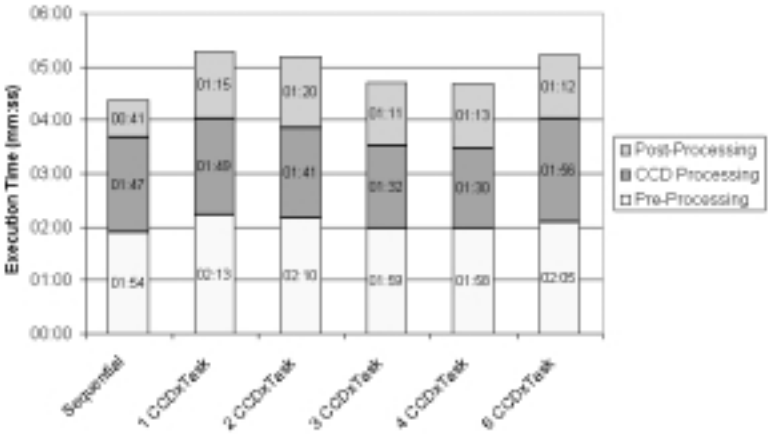
Fig. 5. Context and execution time comparison.

In common qith some other projects [2,9], Grid*W*ay addresses adaptive scheduling, adaptive execution and fault tolerance. However, we note the advantages of its modular, decentralized and "end-to-end" architecture for job adaptation to a dynamic Grid environment.

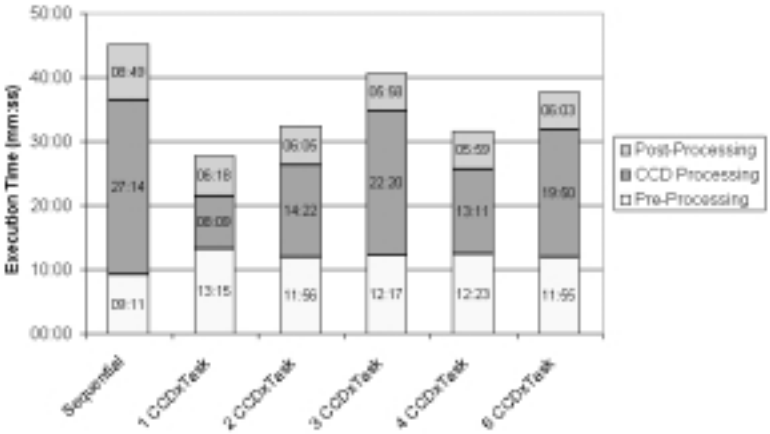
5. Application workflow

The XMM-Newton EPIC-pn data reduction software was designed to analyse the data of each CCD sequentially, see Fig. 3 (left side). Nowadays, users request to the XSA to reduce (data reduction) different observations, in stead of download the observation and analyse the data in their computers (Fig. 3 (right side)). In this context, the Simple Program Multiple Data (SPMD) approach fits perfectly, and consequently the first step taken has been the migration of this sequential task, being then executed on a Grid environment.

The final product of the data processing is a calibrated event file that can be used for the Astronomers to study the properties of the astronomical source (spectral analysis, light curves, timing analysis.)



(a) Small observation



(b) Large observation

Fig. 6. Execution time for different approaches.

The bulk of the analysis of data corresponding to one CCD does not depend on the other CCDs, so we can make our data processing task work in parallel. As shown later, some parts of the task must be run sequentially, and other parts can be run in parallel (both sequential and parallel workflows are shown in Fig. 3).

The new parallel task consists of three steps. The first is called *pre-processing*. It retrieves the observation from the XSA, prepares the dataset for analysis using the latest version of the Current Calibration Files (CCF) and creates the attitude history file with the satellite pointing attitude corrected. In the second step, called *CCD-processing*, the twelve CCD processing tasks are launched in parallel. They perform the data reduction of each CCD (bad pixels correction, energy conversion, etc). It is important to note that only the data needed by each CCD processing task are sent to the node where the CCD is going to be processed, although there are some common files that are needed by all CCD processing tasks. The last step, called *post-processing*, merges the CCD data produced by the second step in a single event file and calculate the good time intervals for the entire observation. Each step has been implemented by a shell script that prepares the environment and executes the appropriate SAS tasks (*cifbuild*, *odfingest*, *atthkgen*, *epproc*, etc).

When developing the task, some issues have come up which have been sorted out as follows:

– In order to get the observations from the XSA, a tool called AIOClient has been used. This client tool uses a socket to connect to the XSA and to send a command where the observation ID is specified. Once the XSA

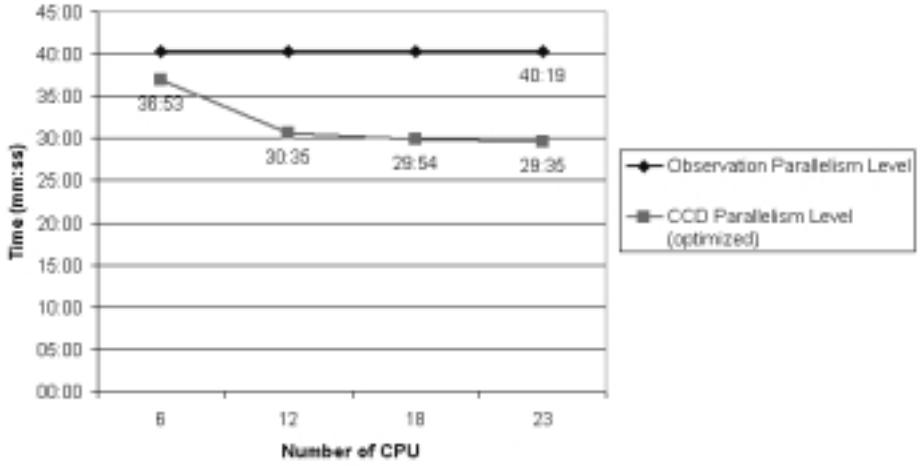


Fig. 7. Observation and CCD parallelism level results (6 observations).

server receives the command, it proceeds to write the observation in the socket for the AIOclient to store it locally.

– When processing an observation, SAS software needs to access the CCF. Because it is not known a priori which CCF files are used when processing a determined observation, all the calibration files must be accessible. There are different approaches that can be followed here, one is to replicate all the data in all resources of the VO. Another more efficient option could be to put replicas that would be managed by a Replica Manager such as Replica Location Service (RLS) provided by Globus. For simplicity purposes, in our experiment, the CCF files have been replicated in all resources of the VO.
– The same problem applies with the XMM-Newton data reduction software. SAS is a large collection of libraries and programs that need to be together when performing the data processing. The same solution of replication has been chosen again.
– It is necessary to create a script to set up the appropriate environment variables on each resource. This is just to make the *pre-processing*, *CCD-processing* and *post-processing* scripts easier to develop without having to worry about the location of SAS software, CCF files and JAVA SDK.

6. Results

The goal of this experiment was to determine if this approach is useful and could be applied to a grid in production, in such a way that many organizations may collaborate to support the needs of the scientific community. To that end some tests have been performed.

The XSA infrastructure offers the astronomer the possibility to analyze a requested observation on-the-fly with the latest version of the SAS software. If this observation is processed on the Grid, apart from the usual benefits, such as: collaboration between loosely coupled organizations, exploiting underutilized resources, resource balancing, reliability. It would be desirable if the observation processing execution time could be reduced significantly.

The first test (see Fig. 4) shows a comparison of execution time when processing small, medium and large observations sequentially (SPMD approach with only one piece of data) and in parallel (CCD processing is performed in different tasks). We can see that every execution consists of three steps (see Section 5) and that only the central job (*CCD-processing* job) can be parallelized (Amdahl's law). Taking this into account, the total execution time is reduced considerably in the medium and large size observations (27 minutes in the parallel approach compare with the 45 minutes for sequential execution for the large observation), although in the small one it takes more time to process it in parallel due to the overheads of data transfer through the nodes, the context initialization time, etc.

If we take a close look at the three step time (see Fig. 5), and divide the whole duration of each phase into the execution time and the context initialization time, we can appreciate that the context initialization time can be rejected in all phases except in the processing of small observations. In that phase, the duration of the CCD execution and the CCD Context initialization stages are comparable, so when processing small observations, data transfer may be even more important than CPU power in the CCD processing phase.

In order to optimize this situation and given that the processing of each CCD needs a common set of files that are sent together with each task, a possible optimization could be to group 2, 3, 4 or 6 CCDs per task so that these common files would be sent only to 6, 4, 3 or 2 nodes respectively, reducing the data transfers needed and consequently the context initialization time. It is important to note that this time (context initialization) represents both the *prolog* and *epilog* execution time (as explained in Section 4).

Figure 6 shows that the best approach in large observations is the one with 12 tasks (one per each CCD) in the CCD processing stage. When processing small observations, we notice that reducing the number of tasks (and in consequence the data transferred) is a good way to obtain better performance than in the 12 tasks approach; it is always faster to launch the sequential task in a single CPU.

When analyzing these type of figures, strange results may appear but they can be easily explained. For instance, if we take a look at Fig. 6, the "4 CCDxTask" entry would be expected to be bigger than the "3 CCDxTask" entry because there are fewer CPUs that are going to process the CCDs. This is caused by the heterogeneity of the resources on the Grid. When reducing the number of tasks, they are scheduled in fewer but better resources which have more network bandwidth and CPU power, so take less time to run. As a conclusion, Grid tests must be analyzed carefully because there are more variables to consider than in cluster computing (the nodes do not have the same CPU power and network bandwidth).

Taking all these results into consideration, we have come to the conclusion that the optimal solution would be to launch the medium and large size observations by following the CCD parallelism level approach and the small observations by following the SPMD approach. As a result, we would save time when processing medium and large observations and spend the same time when processing small observations. We would, of course, take advantage of the benefits that Grid computing provides in all cases.

The comparison between the observation and CCD parallelism level approaches can be seen in Fig. 7. In this test, where 6 observations have been processed, it can be appreciated that the time spent on the data processing is reduced by using our approach (large and medium observations at a CCD parallelism level and small ones at SPMD approach). We get a better result even when the number of nodes is the most suitable to the SPMD approach (which is 6 observations and 6 CPUs), and as we increase the number of CPUs the time is reduced until the amount of CPUs is high enough (around 23 CPUs).

7. Conclusions

We have presented the first parallelized SAS task prototype that can be executed in a Grid architecture by using Grid*W*ay and Globus solutions.

As can be seen from the results, the parallelized SAS task together with Grid*W*ay solution speedup the data processing for large observations. Also the fact that we are splitting the large observations (which are CPU intensive) into small jobs makes the heterogeneity of our architecture less relevant.

The CCD level parallelism approach is a good solution because the Grid is thought to be unstable, dynamic and self-managed. Therefore, if a resource goes down, only one piece of our application must be restarted, or if a new better resource is discovered, the task that has been running under a previously defined threshold can be migrated to the new resource. Summarizing, the resources fit better to the Grid approach because it makes use of the massive parallel resources and is more appropriate for adaptability purposes.

Apart from the on-the-fly data processing, where we have seen the benefits of our approach, we can also consider the massive data reduction scenario (pipeline reprocessing, catalogue production. . .). In this case, our Grid technology solution together with the SAS parallel task for large and medium size observations, and SAS sequential task for small size observations, will speedup the process.

References

[1] B.R. Barkstrom, T.H. Hinke, S. Gavali, W. Smith, W.J. Seufzer, C. Hu and D.E. Cordner, Distributed Generation of NASA Earth Science Data Products, *Grid Computing* **1**(2) (2003), 101–116.
[2] F. Berman, R. Wolski, H. Casanova et al., Adaptive Computing on the Grid Using AppLeS, *IEEE Trans. Parallel and Distributed Systems* **14**(4) (2003), 369–382.
[3] C. Gabriel et al., The XMM-Newton SAS – Distributed Development and Maintenance of a Large Science Analysis System: A Critical Analysis, in: *Astronomical Data Analysis Software and Systems XIII*, ASP Conferences Series, 2004, pp. 759–763.
[4] E. Huedo, R.S. Montero and I.M. Llorente, *Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications*, In Proc. 12th Euromicro Conf. Parallel, Distributed and Network-based Processing (PDP 2004), IEEE CS, 2004, 28–33.
[5] E. Huedo, R.S. Montero and I.M. Llorente, A Framework for Adaptive Execution on Grids, *Intl. J. Software – Practice and Experience* **34**(7) (2004), 631–651.
[6] E. Huedo, R.S. Montero and I.M. Llorente, The GridWay Framework for Adaptive Scheduling and Execution in Grids, *Scalable Computing: Practice and Experience* **6**(3) (2005), 1–8.
[7] F. Jansen et al., XMM-Newton Observatory. I. The Spacecraft and Operations, *Astronomy and Astrophysics* **365** (2001), L1–L6.
[8] L. Stüder et al., The European Photon Imaging Camera on XMM-Newton: The pn-CCD camera, *Astronomy and Astrophysics* **365** (2001), L18–L26.
[9] S. Vadhiyar and J. Dongarra, A Performance Oriented Migration Framework for the Grid, in: *Proc. 3rd Intl. Symp. Cluster Computing and the Grid* (*CCGrid 2003*), IEEE CS, 2003, pp. 130–137.
[10] R. Williams, *Grid Computing: Making the Global Infrastructure a Reality*, chapter Grids and the Virtual Observatory. Wiley, March 2003.
[11] R. Williams et al., *Virtual Observatory Architecture Overview*, Version 1.0. Technical Report IVOA Note 2004-06-14, International Virtual Observatory Alliance, 2004, Available at http://www.ivoa.net/Documents/Notes/IVOArch/IVOArch-20040615.html.
[12] Y. Zhao, M. Wilde, I. Foster et al., Virtual Data Grid Middleware Services for Data-Intensive Science, *Concurrency Computation – Practice and Experience* **18**(6) (2006), 595–608.