Serb. Astron. J. № 179 (2009), 75 - 86 **UDC 523.44–17**
DOI: 10.2298/SAJ0979075N Original scientific paper

COMPUTATION OF ASTEROID PROPER ELEMENTS ON THE GRID

B. Novaković[1], A. Balaž[2], Z. Knežević[1] and M. Potočnik[3,4]

[1]*Astronomical Observatory, Volgina 7, 11060 Belgrade 38, Serbia*
E–mail: *bojan@matf.bg.ac.rs*

[2]*Scientific Computing Laboratory, Institute of Physics Belgrade, Pregrevica 118, 11080 Belgrade, Serbia*

[3]*Faculty of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, 11120 Belgrade, Serbia*

[4]*Belgrade University Computing Center, Kumanovska bb, 11000 Belgrade, Serbia*

(Received: July 20, 2009; Accepted: August 21, 2009)

SUMMARY: A procedure of gridification of the computation of asteroid proper orbital elements is described. The need to speed up the time consuming computations and make them more efficient is justified by the large increase of observational data expected from the next generation all sky surveys. We give the basic notion of proper elements and of the contemporary theories and methods used to compute them for different populations of objects. Proper elements for nearly 70,000 asteroids are derived since the beginning of use of the Grid infrastructure for the purpose. The average time for the catalogs update is significantly shortened with respect to the time needed with stand-alone workstations. We also present basics of the Grid computing, the concepts of Grid middleware and its Workload management system. The practical steps we undertook to efficiently gridify our application are described in full detail. We present the results of a comprehensive testing of the performance of different Grid sites, and offer some practical conclusions based on the benchmark results and on our experience. Finally, we propose some possibilities for the future work.

Key words. Minor planets, asteroids – Methods: data analysis

1. INTRODUCTION

The methods of observing Solar System bodies have changed dramatically in the recent years and will be changing even faster in the next future due to advances in digital astrometry. The new powerful facilities and the improved strategies of data collection give rise to an unprecedented improvement of the accuracy and efficiency of astronomical observations in general and of the astrometric observations in particular, and we are now facing a task of handling huge amounts of raw observational data in terms of their storage, reduction and post processing.

The obvious question arises, what needs to be improved in terms of the computing power and in the existing algorithms to cope with the expected rate of data from the next generation all-sky surveys? The issue is not with the reduction of the raw observations, as available computational resources grow at essentially the same rate as the capability of generating astrometric data. An exponential growth with time of the number of elements on a chip or, as of recently, of a number of chips on a platform, affects the number of pixels in a CCD camera and the performance of the computers used to process astrometric

data much in the same way. Reliability of the data reduction is the main problem when handling large sets of raw observations (Milani et al. 2008).

Once produced, the output of observations reduction must be processed in such a way to extract as much information contained in the data as possible, and it is at this stage that the significant increase of the data can cause more serious problems with the necessary resources. The algorithms often contain code of non linear complexity, which can affect the performance to the point of their failure to meet the requirements of the specific task (timely production of the results, accuracy, reliability, etc.) In addition, there is an ever growing need to improve upon the existing solutions, and this, as a rule, involves more complex and extensive computations and requires more and more powerful and efficient resources to cope with such demands. This is the challenge which puts forward new ideas and approaches, that become necessary even in the case of the well known problems and their widely used and well tested solutions.

In the present paper we describe one such problem to become critical in the immediate future, that of computing asteroid proper elements for tens, even hundreds of thousands of newly discovered asteroids, to be performed in the real time (essentially between two successive lunations). The data have to be collected, processed and retrieved, stored in catalogs and made available on-line, and all this must be performed in a highly automated manner, with as little human interactive involvement as possible. We would also like to extend the time span covered by the numerical integrations of orbits in order to better account for the long term variations of orbital elements which could have hitherto been only partially dealt with or had to be entirely neglected due to the short integrations limited by the available computing power. This in turn means to recalculate the proper elements for all the asteroids in the catalogs from the scratch in order to get a homogeneous data set. Such a formidable task cannot be completed without resort to parallel computing, and an obvious, cost effective way to achieve these goals is to resort to the use of the existing Grid infrastructures. A considerable effort is already devoted in the astronomical community towards adopting and using the Grid computing (e.g. ESAGrid Planck project[1], UK AstroGrid Virtual Observatory[2], German Astronomy Community Grid[3], ESA Gaia Grid[4], US National Virtual Observatory[5], EGEE Grid Observatory[6]). The gridification of already existing applications is the next step in using of the new and powerful e-Science tools and of the available infrastructures in this field of science.

The paper is organized as follows: in Section 2. we give a brief review of basic definitions and methods to compute asteroid proper elements, of their usage in the Solar System research, in Section 3. we discuss the Grid computation, its infrastructure, middleware and workload management, in Section 4. we describe in full detail the PROPEL project within which we implemented our asteroid proper elements computation application on the Grid, we discuss specific software solutions, Grid interface development and improvement (gridification and automation), and we give some basic results of the software performance testing and benchmarks. In Section 5., we present our conclusions.

2. ASTEROID PROPER ELEMENTS

The proper orbital elements are derived from the instantaneous osculating elements by removing the short and long periodic perturbations. By definition they represent the integrals of motion and are thus supposed to be a sort of average characteristics of motion and constant in time. It is, however, well known that a full N-body problem is non integrable, and that therefore it does not poses such integrals. The proper elements can only be computed as quasi integrals of motion, that is, as more or less good approximations of the real dynamics, or as the true integrals of motion, but of a significantly simplified dynamics. It is just the deviation of proper elements from constancy (over the time spans of interest) that is used as a measure of the goodness of the approximation and of the accuracy of the theories and procedures used to compute them.

The classical proper orbital elements are: the proper semimajor axis ($a_{\rm p}$), the proper eccentricity ($e_{\rm p}$), the proper inclination ($I_{\rm p}$), the proper longitude of perihelion ($\varpi_{\rm p}$), and the proper longitude of node ($\Omega_{\rm p}$). Let us note that the term "proper" is in practice sometimes used in the sense of the constancy of a certain quantity in time. Thus, different parameters sharing this property have been used for the same or similar purposes, like the resonant proper parameters (Milani 1993, Morbidelli 1993), the proper fundamental frequencies (Carruba and Michtchenko 2007), even the simple long term averages of the instantaneous values, etc.

In the case of asteroids, the proper orbital elements are used for two main purposes: as parameters for the classification of asteroids into families, and as tools to study their long term dynamical evolution (see Knežević et al. 2003), and the references there-

[1] http://www.rssd.esa.int/index.php?project=Planck

[2] http://www.astrogrid.ac.uk/

[3] http://www.gac-grid.de/

[4] http://gaiagrid.esa.int/

[5] http://www.us-vo.org/

[6] http://technical.eu-egee.org/index.php?id=393

in for a comprehensive review of the classical and contemporary methods to compute the asteroid proper elements and for the discussion on their application). The proper elements for the majority of asteroids of low to moderate eccentricities and/or inclinations are computed by means of the analytical theories based on the series development of the perturbing Hamiltonian. These theories are extremely complex and cannot be pursued too far. They require handling of complicated, cumbersome relations and are subject to problems of the convergence of solutions. The most advanced available theory of the kind is the theory by Milani and Knežević (Milani and Knežević 1990, 1994), based on the Lie series canonical transformations (Hori 1966, Yuasa 1973). It takes into account terms in the expansion of the perturbing Hamiltonian up to the second order in perturbing mass and up to degree four in eccentricity and inclination. Once developed, the procedure to compute proper elements by means of the analytical theory is very efficient and suitable for the computation of large catalogs of proper elements for hundreds of thousands of asteroids; the results, however, are supplied without error estimates (because these cannot be computed analytically), and are known to be of limited accuracy.

Let us note that several specially adapted theories exist for dynamically specific asteroid populations, as for the high eccentricity and inclination asteroids (Lemaitre and Morbidelli 1993), for Trojans (Milani 1993, Beauge and Roig 2001), Hildas (Schubart 1982), etc.

Knežević and Milani (2000) have recently developed a new method for computation of the so called synthetic proper elements of asteroids, which consists of a set of purely numerical procedures, collectively called the synthetic theory. The procedure includes: (i) numerical integration of asteroid orbits in the framework of a realistic dynamical model; (ii) online digital filtering of the short periodic perturbations to compute the mean elements and the proper semimajor axis; (iii) Fourier analysis of the output to remove main forced terms and extract proper eccentricity, proper inclination, and the corresponding fundamental frequencies; (iv) check of the accuracy of the results by means of running box tests. The accuracy of the synthetic proper elements is better by a factor of more than 3 on the average with respect to the results derived by means of the above mentioned most advanced version of the analytical theory (Milani and Knežević 1994).

The synthetic proper elements of asteroids in the outer part of the main belt (between 2.5 and 4.0 AU) are computed by using the dynamical model including the four outer major planets as perturbing bodies. To account for the indirect effect of the inner planets, the barycentric correction is applied to the initial conditions (Milani and Knežević 1992). The orbits are integrated by means of the ORBIT9 integrator, which employs as starter a symplectic single step method (implicit Runge-Kutta-Gauss), while a multi-step predictor performs most of the propagation (Milani and Nobili 1988). The integration initially covers a span of 2 Myr for all the bodies, but it is subsequently extended to 10 Myr for those asteroids for which the results of the short run prove to be insufficiently accurate.

To compute accurate synthetic proper elements for asteroids in the inner main belt (between 2.0 and 2.5 AU) the orbits of asteroids in this region are integrated in the framework of a dynamical model including the direct perturbations by 7 major planets, from Venus to Neptune. The indirect effect of Mercury is taken into account by applying a barycentric correction to the initial conditions. Orbits of all the included asteroids are integrated for 2 Myr (1 Myr forward and 1 Myr backward in time), and the proper values with their errors in terms of the standard deviations and maximum excursions are computed by joining data from both runs.

The setup similar to the one employed for the asteroids in the outer main belt is used also in the case of the transneptunian objects (TNO). The integrations are performed only forwards in time, but using a much longer time step; thus the interval of time of 100 Myr is covered in the case of the numbered TNOs, and of 10 Myr for the multiopposition ones. Finally, an adjusted synthetic theory is used to compute proper elements for Trojan asteroids. In this case the integrations cover 5Myr and 50Myr.

The computation of the synthetic proper elements is a time consuming procedure. It currently takes hundreds of hours of CPU time on a standard workstation for each monthly update (when, typically, the proper elements are computed for several thousands of newly numbered asteroids). It is also not a fully automated procedure, with significant interactive user intervention in data preparation, input and output. A 100-fold increase of the new discoveries is expected from the next generation observational surveys like Pan-STARRS (Jedicke et al. 2007) and LSST (Ivezić et al. 2007). Thus, it is absolutely necessary to resort to a new approach, both in terms of the resources (like making use of the distributed computation on the Grid) and of the software (parallelization of the code, automation of the procedures of data handling, etc.)

What is the most important is the availability of both analytic and synthetic proper elements for all discovered asteroids, with frequent updates, as this allows new insights in the dynamics and evolution of the entire asteroid population.

3. GRID COMPUTING

Many science experiments generate enormous amounts of data. The processing of these data requires huge computational and storage resources, as well as human resources for operation and support. Scientists also face problems requiring vast computing power, i.e. number crunching problems. We can roughly categorize these tasks into: tasks with large amounts of distributed data; number crunching tasks; tasks which require simultaneous work of a group of researchers/developers, accessing the same resources at the same time. Note that typical prob-

lems may consist of overlapping tasks from different categories, i.e. they may contain computing-intensive analysis of a large amount of distributed data etc. Often a single computer, a cluster of computers or even a special-purpose supercomputer is not enough for solving challenging science or development problems of today.

In order to avoid these obstacles, middleware concept is introduced - layer of software that is able to interconnect distributed computing and storage resources, and make them interoperate, providing users with the unified access to all resources, even if the underlying software (e.g. batch system on individual clusters) or hardware (e.g. different types of storage elements, ranging from tape robots to generic PCs with several HDDs attached) is different. Of course, this middleware layer is built on top of the existing network infrastructure, which is essential for the proper functioning of Grids.

This approach is in some way similar to the World Wide Web (WWW), and people expect that what WWW has done for the information exchange and sharing, the Grids will do for computing resources sharing. However, there are some substantial differences between WWW and Grids: while on the Internet the basic idea is to provide information and we usually have client-server interaction, in Grids the resources are valuable assets and their use should be governed according to the policies of resource providers. Moreover, in order to have most efficient use of available computing resources, complex algorithms and internal information system need to be developed and deployed, and a set of new services that will allow simple usage by the end users provided.

There are many kinds of Grids with different purposes, such as national Grid infrastructures (aiming to couple high-end resources across a nation, e.g. Academic and Educational Grid Initiative of Serbia - AEGIS[7], or the UK e-Science program), project Grids (funded by certain funding agencies), goodwill Grid infrastructures provided by individuals aiming to help in solving important common problems (e.g. in finding drugs for diseases), consumer Grids established by commercial companies, etc.

Project Grids is currently the main provider of different middleware distributions, some of which are freely available, thus enabling general public to join the Grid, or to adapt it for their own needs. Project Grids is created to meet the needs of a variety of multi-institutional research groups and multi-company "virtual teams", to pursue short- or medium-term projects (scientific collaborations, engineering projects). Such a project is World wide LHC Computing Grid Project[8] (WLCG), which was created to prepare the computing infrastructure for the simulation, processing and analysis of the data of the Large Hadron Collider (LHC) experiments.

The WLCG project shares a large part of its infrastructure and works in conjunction with the Enabling Grids for E-Science project[9] (EGEE), large European series of e-Infrastructure projects with the main goal to provide researchers with access to a geographically distributed computing Grid infrastructure, available 24 hours a day. SEE-GRID is the regional series of projects aiming to provide Grid infrastructure in the South East Europe region, incubate new regional communities, and stimulate development of new Grid-aware applications [10].

3.1. Grid middleware

The essence of the Grid is the software that enables the user to access computers distributed over the network. This software is called "middleware", because it is distinct from the operating systems software that makes the computers run (e.g. Linux) and also different from the applications software that solves a particular problem for a user (e.g. a computer visualization program). The term "middleware" refers to the fact that it is conceptually in between these two types of software. The middleware's task is to organize and integrate the distributed computational resources of the Grid into a coherent structure. This means the objective of the middleware is to get the applications running on the appropriate computers, wherever they may be on the Grid, in an efficient and reliable way. It also provides users with a single interface to the Grid.

Different distributions of middleware exist today - Globus, LCG, gLite, UNICORE, GAT. The gLite middleware (gLite 2007) is successor of the LCG-2 (Peris et al. 2005) middleware, and is the one used by the EGEE, SEE-GRID and several other e-Infrastructures. The EGEE series of projects, among other aims, focuses on maintaining the gLite middleware and on operating a large computing infrastructure for the benefit of a vast and diverse research community. The gLite middleware hides much of the complexity of this environment from the user, giving the impression that all of these resources are available in a coherent virtual computer center.

In the following we briefly describe basic entities ("building blocks") and available interfaces which allow user to run jobs and manage data (Burke et al. 2007).

(i) The access point to the WLCG/EGEE/SEE-GRID Grid is the User Interface (UI). This can be any machine where users have a personal account and where their user digital certificate is installed. From a UI, user can be authenticated and authorized to use the WLCG/EGEE/SEE-GRID resources, and can access the functionalities offered by the Information, Workload and Data management systems.

[7] `http://aegis.phy.bg.ac.yu/`

[8] `http://lcg.web.cern.ch/LCG/`

[9] `http://www.eu-egee.org/`

[10] `http://www.see-grid.eu/` and `http://www.see-grid-sci.eu/`

(ii) A Computing Element (CE) is a set of computing resources localized at a site (often referred to as a cluster, or a computing farm)
(iii) A Storage Element (SE) provides uniform access to storage resources at a certain site. The Storage Element may control simple disk servers, large disk arrays or tape-based Mass Storage Systems (MSS). Most WLCG/EGEE/SEE-GRID sites provide at least one SE. Storage Elements can support different data access protocols and interfaces.
(iv) The Information Service (IS) provides information about the Grid resources and their status.
(v) In a Grid environment, files can have replicas at many different sites. Ideally, the users do not need to know where a file is located, as they use logical names for the files that the Data Management services will use to locate and access them
(vi) The Workload Management System (WMS) accepts user jobs, assigns them to the most appropriate Computing Element, records their status and retrieves their output (Pacini 2005)
(vii) Finally, the Logging and Bookkeeping service (LB) tracks jobs managed by the WMS. It collects events from many WMS components and records the status and history of the job.

3.2. Workload management system

Central element in any Grid infrastructure is WMS. The purpose of WMS is to accept requests for job submission and management coming from its clients and take the appropriate actions to satisfy them. The complexity of the management of applications and resources in the Grid is hidden to the users by the WMS. Their interaction with the WMS is limited to the description of the characteristics and requirements of the request via a high-level, user-oriented specification language, the Job Description Language (JDL), and to the submission of the request through the provided interfaces. The WMS is responsible for translation of these abstract resource requirements into a set of actual resources, taken from the overall Grid resource pool, to which the user has access permission.

The JDL allows the description of the following request types supported by the WMS:
(i) Job: a simple application
(ii) DAG: a direct acyclic graph of dependent jobs
(iii) Collection/Bulk: a set of independent jobs

There is a set of client tools, referred to as WMS-UI, which allows the user to access the main services (job management services). These client tools include a command line interface, a graphical interface and an API, providing both C++ and Java bindings, which allow the requests to be submitted and managed programmatically. Through the WMS UI user can find the list of resources suitable to run a specific job, submit a job/DAG for execution on a remote Computing Element, check the status of a submitted job/DAG, cancel one or more submitted jobs/DAGs, retrieve the output files of a completed job/DAG (output sandbox), retrieve and display logging and bookkeeping information about submitted jobs/DAGs.

After submission, the request passes through several components of the WMS, before it completes its execution. The internal architecture of the WMS is given in Fig. 1. There are two approaches for accepting the incoming requests: one is based on a generic daemon, and the other on the Web Services based interface.

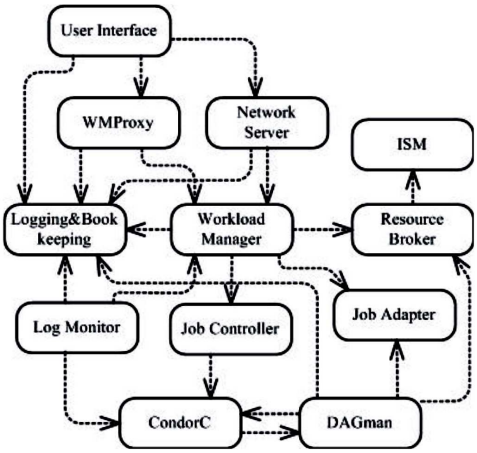

Fig. 1. *Overview of the WMS architecture.*

The Network Server (NS) is a generic network daemon that provides support for the job control functionality. It is responsible for accepting incoming requests from the WMS-UI (e.g. job submission, job removal), which, if valid, are then passed to the Workload Manager. The approach based on NS component has been discontinued in gLite middleware, and is no longer available.

The Workload Manager Proxy (WMProxy) is a service providing access to WMS functionality through a Web Services based interface. Besides being the natural replacement of the NS during the migration to the fully compliant Service Oriented Architecture (SOA) WMS, it provides additional features such as bulk submission and the support for shared and compressed sandboxes for compound jobs.

The Workload Manager (WM) is the core component of the Workload Management System. Given a valid request, it has to take the appropriate actions to satisfy it. It coordinates other modules that provide a matchmaking service (Resource Broker), the actual job management operations (CondorC), preparation of the CondorC submission file and creation of the appropriate execution environment in the CE worker node (Job Adapter).

The Logging and Bookkeeping (LB) service provides support for job monitoring functionality: it stores all information concerning events generated by the various components of the WMS.

For a generic job there are two main types of request: submission and cancellation. The submission request passes the responsibility for the job to

the WM. The WM will then pass the job to an appropriate CE for execution, taking into account the requirements and the job preferences expressed in the job description file. The decision which resource is to be used is the outcome of the matchmaking process between the submission requests and the available resources. The job can also be cancelled by the user at any time after it is submitted using the job ID that uniquely identifies each job.

4. GRIDIFICATION

Gridification can be defined as a process of deploying/integrating an application onto the Grid infrastructure (Temizsoylu and Zengin 2006). It is a sensitive process that must be worked on carefully so that a reasonable efficiency can be obtained from the use of the infrastructure. The application should be analyzed in detail, in a systematic way and the nature of the application should be taken into account to appreciate whether it enables and justifies the use of Grid. When considering whether an application is a good candidate to execute in a Grid environment, one must first understand the basic structure of a Grid (see Section 3.), the services that are and are not provided, and how this can affect the application.

Often people assume that for an application to gain advantage from a Grid environment, it must be highly parallel or otherwise able to take advantage of parallel processing. In fact, some like to think of a Grid as a distributed cluster. Although such parallel applications certainly can take advantage of a Grid, one should not dismiss the use of Grids for other types of applications as well. Even a single threaded batch job could benefit from a Grid environment by runing on any of a set of systems in the Grid, making use of unused cycles.

The upcoming set of applications will be able to solve large scale computational and data-intensive problems, previously limited by constraints of resource availability. Science comes as natural area which can take advantage of working in the Grid environment. In astronomy, applications for solving N-body problems are well known to be computationally complex, CPU intensive and time consuming, what makes them perfectly suitable for Grid. An example, how Grid can be used for N-body simulations, was presented recently by Groen et al. (2008) who applied it for the star cluster simulations.

In this Section we describe the process of gridification of an application for computing asteroid proper elements on the Grid, which we named PROPEL. Among the computational problems arising in science, some of the most difficult to parallelize are N-body problems. Therefore, as the best choice for gridification of PROPEL application we have chosen the parallel batches strategy (Jakob et al. 2003). The parallel batches strategy is based on simple parallelization by grouping similar orbits into independent batches. In this way, many independent batches could be run in parallel on different CPUs or on different Grid sites. A main gridification goals can be summarized as follows:

(i) All batches must be eventually completed and collected,
(ii) The whole process in Grid environment must be fully automated,
(iii) The system must be adaptive to exponentially growing number of newly discovered objects.

The whole process of producing asteroid proper elements is divided in three main steps: preprocessing, processing and post-processing. The preprocessing and post-processing are computationally very cheap and they can be done on a PC. On the other hand processing is a part of PROPEL application which is computationally expensive and should be executed in the Grid environment.

During the pre-processing the set of asteroids (for which proper elements are to be calculated) is extracted from the catalog of asteroid osculating elements. Subsequently, this set of asteroids is divided in small batches of similar orbits in terms of eccentricity and inclination. These batches are input files used by the PROPEL during its execution in the Grid environment. The post-processing is step when outputs produced by PROPEL, for each batch, are joined back together in to single file.

Gridification process of PROPEL application consists of three separate steps. The first step is the software management of an application in the Grid environment, part of which is the installation process of the PROPEL application itself. The second step is the automation of the PROPEL application in the Grid environment, where the goal is to automatize certain operations in the usage of application so that the user can take full advantage of using the Grid environment. This step also includes matchmaking the best available resources. The last step is benchmarking performances of the application. Detailed workflow of PROPEL application is presented in Fig. 2.

4.1. Software Management of PROPEL Application

Software management of the PROPEL application in the Grid environment includes several procedures:

(i) **Installation** - Installation procedure allows the user to install PROPEL application on the desired Grid site.
(ii) **Validation** - Validation procedure allows the user to validate already installed PROPEL application in order to verify that all the steps performed during the installation stage were successful. In case of a successful validation, appropriate information about the installation is submitted to the Information Service (IS). This provides the users, who wish to use PROPEL application, with information on which Grid sites PROPEL application is available.
(iii) **Removal** - Removal procedure allows the user to remove already installed and validated instance of PROPEL application from the destination Grid site. In case of a successful removal, information about the installation of PROPEL application on the destination Grid site is removed from the IS.

(iv) **Running the application** - Successfully installed and validated PROPEL application can be run by the regular users. Users need to submit Job Description File (JDL) in which they specify some of the job parameters (e.g. input and output files, rank and requirements expressions, etc.).

First three procedures (installation, validation and removal) are performed using the Experimental Software Manager (ESM) (Peris et al. 2005) user class. This class has special privileges in the Grid environment which allows to manage software installations. These procedures are performed using specially created scripts, which in turn, use the LCG-ManageSoftware (SEE-GRID Gridification Guide, 2007) utility.

4.2. Automation of PROPEL application

Some of the most common operations in the automation of any application in the Grid environment include:

(i) Creation of a number of similar jobs with different arguments and input files
(ii) Keeping track and checking status of many similar jobs
(iii) Manipulating many output files of different jobs in an organized manner.

The idea behind automation of PROPEL application is to divide asteroids into segments of optional size, and then process each segment in one partial (sub) job. Partial jobs are identical except for a few different arguments which include asteroid

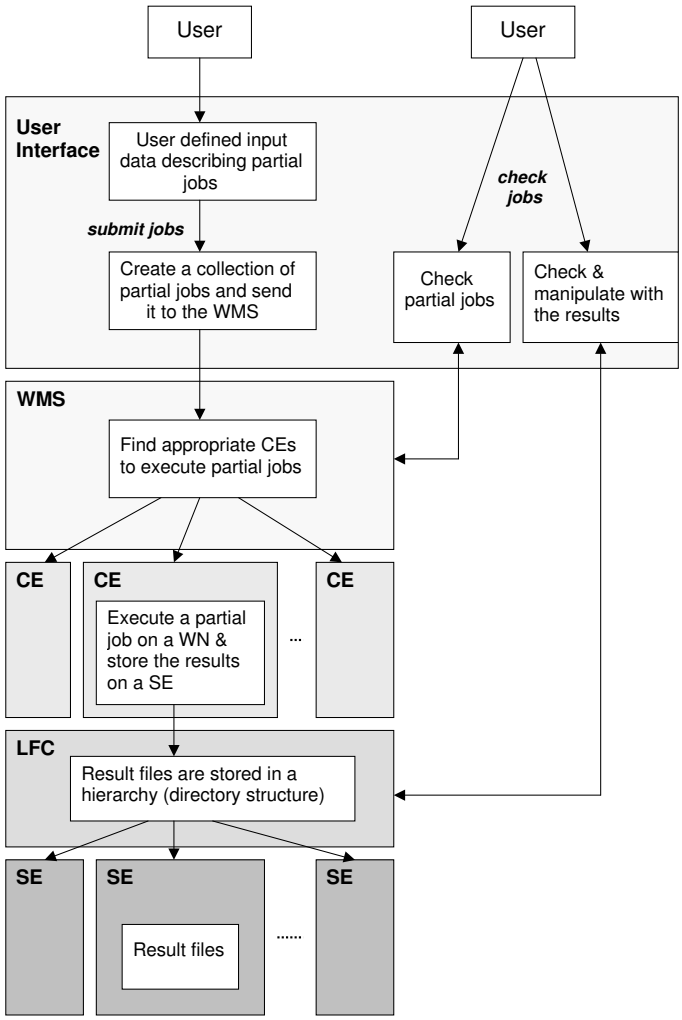

Fig. 2. *Detailed workflow of PROPEL application.*

segment to be processed and Computing Element (CE) where the job will be executed. Because the partial jobs executions are independent, partial jobs are organized in the form of a job collection.

Two scripts are available, *submit jobs* which handles creation and submission of jobs to the Grid and *check jobs* that handles checking jobs status and management of output (results) files. These scripts are designed to work in the gLite middleware (Burke et al. 2007). Script *submit jobs* does the following:

(i) Reads the user defined input where job options and partial jobs parameters are stored
(ii) Creates temporary JDL file containing the collection of partial jobs
(iii) Submits JDL file to the Grid
(iv) Creates LFC (LCG File Catalogue) directories where output files will be stored
(v) Creates run scripts for each partial job
(vi) Creates an output file which is used by *check jobs* script

Check jobs script offers following functionalities:

(i) Checking job results - script checks which of the partial jobs have submitted output (results) files to the LFC and reports the missing output files. In this way user can easily find out what is the problem and why some of the jobs are aborted.
(i) Retrieval of job results - script allows the user to download partial job results to the User Interface (UI) from LFC and Storage Element (SE), user can specify whether he/she wants to download all of the output files or just some.
(iii) Checking job status - script checks the status of partial jobs to see if jobs are still running or if some are aborted or have failed.
(iv) Checking job output information - script allows the user to download output information for partial jobs (console outputs); this can be useful for debugging in case some of the jobs have failed.
(v) Removal of output files - script allows the user to delete all or some of the output files from the remote LFC directory and SE Input file for the *check jobs* script is a file generated by the *submit jobs* script which contains information about submitted partial jobs.

4.3. Requirements and Rank

As the Grid consists of different resources (e.g. CPU clocks, CPU vendors, amount of available memory) an important task in our procedure is matchmaking the best appropriate resource. This is done through the Requirements and Rank expressions which allow the user to specify respectively which are the needs and preferences, in terms of resources, of their applications. The matchmaking process proceeds by evaluating the Rank and Requirements expressions for each pairwise combination of queued jobs and available resources. The pair satisfying the Requirements and having the maximum Rank is selected, and the job is assigned to the appropriate resource.

Following the idea described in Agarwal et. al. (2007) we developed an algorithm for Rank expressions which can be used to cope with two main problems:

(i) to achieve a balanced distribution of jobs to each resource
(ii) to estimate waiting time (WT) and to match a site where the job will start executing the soonest

An appropriate Rank expression which fulfills both conditions mentioned above is based on the following formula:

$$R_i = \max(\frac{A_i - Q_i}{C_i}, 0) + \min(0, \frac{-WT - P * M_i}{T_i^{\text{avg}}}) \quad (1)$$

Here R indicates the Rank of the resource, A is the number of free processors, Q is the number of queued jobs, C indicates the total number of processors, P is a penalty equal to the estimated mean increase in the WT from the currently matched jobs, M is the number of matches in the current matchmaking cycle, T^{avg} is the average running time of the previous jobs, while subscript i indicates i^{th} resource (i.e. SEE-GRID site).

The first term in the right hand side of the Eq. (1), refers to the load balancing while the second term refers to the waiting time. This Rank expression reduces to load balancing when there are free processors, while the second term becomes important when there are no free processors and the first term becomes insignificant. On the other hand one should bear in mind that in a Grid environment it is difficult to balance requests from the application point of view because the Grid is being used by other applications, using other rank expressions, what usually leads to unbalanced situations. Nevertheless, using algorithm to balance requests is still useful in a Grid.

The Rank expression presented above is not the only possible solution but many other algorithms could be used (e.g. Casavant and Kuhl 1994, Xu and Lau 1997, Yagoubi and Slimani 2007). Some of them might be even more efficient. Analysis of other Rank expressions and looking for the best possible solution for our application we intend for future work.

4.4. Benchmark

As the final step in our gridification procedure, we performed several benchmarks in order to test application and Grid infrastructure reliability, stability and performance. Benchmarking is a widely accepted method to evaluate the performance of computer architectures. Grid performance evaluation is an important approach to improve the performance of applications intended for use in the Grid environment.

In general, we can identify three aspects that should be assessed when evaluating the Grid and application developed for Grid, namely: functionality, reliability and performance (Montero et al. 2006). Therefore, Grid benchmarks should verify a basic functionality of the environment. A suitable methodology for Grid benchmarking should help to identify

sites with the best performances, to determine the reliability of the Grid and/or application, to test the best choices for application's controls and parameters, and to identify and quantify failure situations.

The first set of benchmarks was performed to identify sites with the best performances. In order to measure computational time at various SEE-GRID sites, the same job has been executed ten times on each site. The average computational time (ACT), in minutes, was determined for each site and obtained results are presented in Table 1. In addition, in Table 1, CPU's models and operating systems for each site are given. All sites operate under one of the versions of Scientific Linux. Scientific Linux (SL) is an open source free Linux distribution, co-developed by Fermi National Accelerator Laboratory and the European Organization for Nuclear Research (CERN), which aims to be 100% compatible with and based on Red Hat Enterprise Linux.

It is known that computational time in the case of N-body problems is growing as $CT(N^2)$. However, to compute the orbits of asteroids it is not necessary to take into account attraction of the asteroids on the major planets (Milani et al. 1990). The masses of the asteroids are so much smaller than the masses of the planets that we can, without any significant loss in accuracy, resort to a heliocentric (N+M)-body problem, that is, we can use the model with N planets attracting each other and attracting each of the M asteroids, but with the asteroids massless and not affecting at all the motion of the planets.

As we were interested to find out what is the optimum number of asteroids per job, that is, how the computational time scales with the number of asteroids per job for PROPEL application, several tests were performed in order to determine this relation. We define a function $P = CT/M$, which allows us to study how the *computational price* for one asteroid depends on the number of asteroids per job. The obtained results are presented in Fig. 3. The two curves in this plot represent results obtained when the determination of Lyapunov characteristic exponent (LCE) is included (open circles), and when it is switched off (filled circles). As expected, the results without calculation of LCE are almost independent of the number of asteroids per job, except for the sharp surge observed for very small M. This surge is due to the fact that for small numbers of asteroids the integration time is dominated by the determination of initial conditions for the propagation of orbits and by the computation of the mutual perturbations of planets which must be done at each step regardless of the number of asteroids involved. Dividing this time with ever increasing number of asteroids produces the sharp drop of the single asteroid's computational price until the propagation of asteroid orbits becomes the most time consuming part of the procedure for large enough number of the included asteroids, thus bringing the ratio P to the observed near constancy. Let us just note that we also verified that the computational price does not depend on the time span covered by the integration.

The LCE indicators of chaotic motion are computed from the variational equations and the procedure involves frequent normalization of the exponentially growing values of the metrics. This is a time consuming process, which affects the computation of asteroid proper elements by increasing P by some 50%, for the small to moderate number of asteroids per job, and by giving rise to an almost linear growth of the value of P with M, although only for large enough M ($M > 250$). The corresponding curve presented in Fig. 3. has a minimum for $M \sim 70$ suggesting that the best choice is to use 70 asteroids per job. However, a steep increase of the curve that begins only beyond some 250 bodies per job, implies that the number of bodies per job up to this latter value does not affect the computational price significantly. On the other hand, one should always take care of a rational use of the available Grid resources and bear in mind that increasing the number of jobs increases the probability to get aborted jobs. Given that, we have chosen to adopt 250 asteroids per job as an optimum choice representing a reasonable compromise between the best performance and a number of jobs to run.

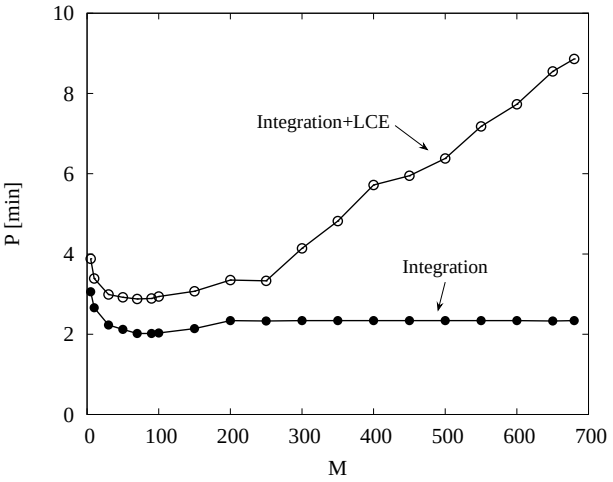

Fig. 3. *The normalized average computational time per asteroid (computational price) as a function of the number of asteroids per job for two different tests, with (filled circles) and without the computation of the LCE (open circles).*

Finally, we study the failure situations and stability of application in the Grid environment. Since we have started working in the Grid environment in November 2005, over 1600 jobs have been submitted and about 1400 of them have been completed successfully. It gives an estimation of about 15% of aborted jobs, but it is important to note here that this rate decreased from about 25% (in the year 2006) to less than 3% (August-December 2007). There are two main reasons for this: firstly, our application became more stable and more suitable for the Grid environment, and secondly, the Grid itself became much more reliable and stable.

These 1400 successfully executed jobs were performed in order to calculate asteroid proper elements for different purposes; out of these several

Table 1. PROPEL benchmark results at various SEE-GRID sites.

SEE-GRID Site	CPU Model	CPU Clock [GHz]	ACT [min]	Operating System
ce.phy.bg.ac.yu	AMD Opteron 285	2.60	90.6	SL 4.5
ce02.grid.acad.bg	AMD Opteron 250	2.40	99.6	SL 4.5
ce01.csa-incas.ro	AMD Athlon 64 3500+	2.20	106.8	-
ce01.info.uvt.ro	AMD Opteron 280	2.40	109.6	SL 4.6
grid01.aob.bg.ac.yu	AMD Opteron 265	1.80	109.8	-
grid01.rcub.bg.ac.yu	AMD Sempron 2800+	2.00	122.3	SL 4.5
ce01.info.uvt.ro	Intel Pentium	3.00	125.4	SL 4.6
cluster1.csk.kg.ac.yu	AMD Athlon 2600+	2.10	125.7	SL 4.5
ce01.isabella.grnet.gr	Intel Xeon	2.80	129.4	SL 3.0.3
ce001.grid.uni-sofia.bg	AMD Opteron 265	1.80	131.4	SL 4.5
grid-ce.feit.ukim.edu.mk	Intel Pentium	3.00	133.8	SL 3.0.9
ce01.info.uvt.ro	Intel Celeron	2.80	135.6	SL 4.6
rti29.etf.bg.ac.yu	AMD Sempron 2600+	1.60	147.6	SL 3.0.8
ce001.grid.uni-sofia.bg	AMD Opteron 242	1.60	148.5	SL 4.5
yildirim.grid.boun.edu.tr	Intel Xeon 5110	1.60	153.4	SL 4.5
cluster1.csk.kg.ac.yu	Intel Pentium	1.70	207.4	SL 4.5
sn0.hpcc.sztaki.hu	Intel Pentium	1.60	229.2	-

tens of jobs were performed to test the application itself and the Grid infrastructure. Over 150,000 asteroid proper elements have been calculated using the Grid, about 70,000 of them as part of regular AstDys[11] database update. The rest have been calculated for fictitious asteroids used in other research. About 450 jobs have been executed as part of the update and the total CPU time was about 4,000 hours. As we performed 12 updates in the period from the beginning of the use of Grid for the purpose, this gives us the estimate that we spent some 333 hours per update on the average. Thus, we would need about two weeks to perform the update if we used the single PC, and this would cause a significant delay in the availability of the proper elements for the newly discovered asteroids. By using Grid, even with all the problems and failures we encounter in practice, the typical update now takes approximately two days.

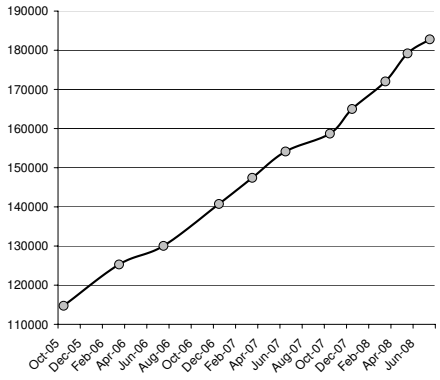

Fig. 4. *The number of asteroids with calculated proper elements since we have started using Grid environment.*

5. CONCLUSIONS

In this paper we described the procedure of gridification of the application to compute asteroid proper elements on the Grid. We explained the rationale for this undertaking by the need to speed up the time consuming computations and make them more efficient. For the time being we preferred to achieve these goals without substantial intervention in the existing, well tested and reliable software (like the parallelization of the code). The improvement we are striving for is particularly important in view of the expected large increase of observational data to be collected by the next generation all sky surveys.

We give the basic notion of asteroid proper elements, of the contemporary theories and methods used to compute them, and we discuss current status-of-the art and setups in use for dynamically distinct populations of the main belt, resonant and distant objects. In Fig. 4. we show the steep increase of the total number of computed proper element sets since the beginning of use of the Grid infrastructure for the purpose. Synthetic proper elements for about 70,000 asteroids are derived in this way, which represents nearly one third of the total number of synthetic proper element sets computed so far. The more important, however, is that the average time for the regular update of the catalogs is significantly shortened with respect to the time needed previously

[11] `http://hamilton.dm.unipi.it/astdys/`

when working with stand-alone workstations. This is in the first place due to the fact that we could use as many as 25 parallel jobs on the SEE-GRID e-Infrastructure available to us; the obvious possibility to run even more jobs in parallel depends on the number of asteroids for which proper elements have to be computed and on the available Grid resources. In any case, the use of Grid proved to be an efficient and cost effective (no significant new investments needed) approach, which guarantees that with some minor additional effort we might be ready to cope with the task of computing asteroid proper elements for many more objects to be discovered by the next generation observational surveys.

We also present basics of the Grid computing, the concepts of Grid middleware and its Workload Management System for the reader to better understand and appreciate its advantages. The practical steps we undertook to efficiently gridify our application are described in full detail, the software management procedures we use and interfaces and scripts we developed to automatize the runs. Finally, we present the results of a comprehensive and thorough testing of the resources we made use of on different Grid sites, coming to the following conclusions based on the obtained benchmark results and our experience:

(i) since our application is number crunching intensive, the CPU speed is, as expected, the most important factor in terms of the efficiency of our application.
(ii) as the outputs from PROPEL application are relatively large files, having enough Grid storage space on SEs is also important
(iii) the network speed between WNs and SEs is relevant as well.
(iv) amount of RAM memory per WN is not a critical parameter (providing there is at least 512MB).
(v) PROPEL application works slightly better on platforms based on AMD processors.
(vi) stability of the Grid has significant influence on the stability of application.

The gridification of the PROPEL application not only allows us to efficiently use the currently available computing resources to compute asteroid proper elements, but also ensures an easy way to access additional resources that may become available in the future on the Grid. The gridified version of PROPEL allows dynamic use of available resources as they are added to the e-Infrastructure, without any special configuration steps. Such approach in using computing resources is flexible and scalable, thus minimizing the work needed to deploy the application. In addition, such gridified version of the application can be easily deployed on other e-Infrastructures and make their use transparent to researchers.

The full automation of the entire procedure, parallelization of the code and deployment of the application to many more sites on different Grids are obvious possibilities for the future work. The practical implementation of these possibilities will certainly be considered, but the timetable and extent of the necessary developments will depend on the availability of the raw data from surveys suitable for this kind of postprocessing.

Acknowledgements – This work was supported in part by the Ministry of Science of the Republic of Serbia under projects No. OI141035 and OI146004. The presented results were obtained on the AEGIS Grid e-infrastructure whose operation is supported in part by EC FP6 and FP7 projects EGEE-II (INFSO-RI-031688), EGEE-III (INFRA-222667), SEE-GRID-2 (INFSO-RI-031775), and SEE-GRID-SCI (INFRA-211338).

REFERENCES

Agarwal et. al.: 2007, *Future Generation Computer Systems*, **23**, 680.
Beauge, C., Roig, F.: 2001, *Icarus*, **153**, 391.
Burke, S., Campana, S., Peris, A. D., Donno, F., Lorenzo, P. M., Santinelli, R., Sciaba, A.: 2007, *CERN, GLITE 3 User Guide*, Avialable online at: https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf
Carruba, V., Michtchenko, T. A.: 2007, *Astron. Astrophys.*, **475**, 1145.
Casavant, T. L., Kuhl, J. G.: 1994, A taxonomy of scheduling in general purpose distributed computing systems. *IEEE Transactions on Software Engineering*, **14(2)**, pp.141–153.
gLite middleware: 2007, http://glite.web.cern.ch/glite/
Groen, D., Zwart, S. P., McMillan, S., Makino, J.: 2008, *New Astronomy*, **13**, 348.
Hori: 1966, *Publ. Astron. Soc. Japan*, **18**, 287.
Ivezić, Ž., J. A. Tyson, M. Jurić, J. Kubica, A. Connolly, F. Pierfederici, A. W. Harris, E. Bowell, and the LSST Collaboration: 2007, In Near Earth Objects, our Celestial Neighbors: Opportunity and Risk, ed. A. Milani, G. B. Valsecchi and D. Vokrouhlický, (Cambridge University Press), pp. 353–362.
Jakob, B., Ferreira, L., Bieberstein, N., Gilzean, C., Girard, J., Strachowski, R., Yu, S.: 2003, Enabling Applications for Grid Computing with Globus, IBM, (www.ibm.com/redbooks).
Jedicke, R. E. A. Magnier, Kaiser, N. and Chambers, K. C.: 2007, in *Near Earth Objects, our Celestial Neighbors: Opportunity and Risk*, A. Milani, G.B. Valsecchi and D. Vokrouhlický eds., (Cambridge University Press), pp.341–352.
Knežević, Z., Milani, A.: 2000, *Cel. Mech. Dyn. Astron.*, **78**, 17.
Knežević, Z., Lemaitre, A., Milani, A.: 2003, In Asteroids III, ed. W.F. Bottke et al. (Tucson: Arizona Univ. Press & LPI), pp.603–612.
Lemaitre, A., Morbidelli, A.: 1994, *Cel. Mech. Dyn. Astron.*, **60**, 29.
Milani, A.: 1993, *Cel. Mech. Dyn. Astron.*, **57**, 59.
Milani, A., Knežević, Z.: 1990, *Cel. Mech. Dyn. Astron.*, **49**, 347.
Milani, A., Knežević, Z.: 1992, *Icarus*, **98**, 211.
Milani, A., Knežević, Z.: 1994, *Icarus*, **107**, 219.

Milani, A., Nobili, A. M.: 1988, *Cel. Mech. Dyn. Astron.*, **43**, 1.
Milani, A., Carpino, M., Logan, D.: 1990, *Advances in Parallel Computing*, **1**, 39.
Milani, A., Gronchi, G. F., Farnocchia, D., Knežević, Z., Jedicke, R., Denneau, L., Pierfederici, F.: 2008, *Icarus*, **195**, 474.
Montero, R. S., Huedo, E., Llorente, I. M.: 2006, *Parallel Computing*, **32**, 267.
Morbidelli, A.: 1993, *Icarus*, **105**, 48.
Pacini, F.: 2005, WMS Guide, https://edms.cern.ch /document/572489/
Peris, A. D., Lorenzo, P. M., Sciaba, A., Campana, S., Santinelli, R.: 2005, CERN, LCG-2-UserGuide, Aviaeble online at: https://edms.cern.ch/file/454439//LCG-2-UserGuide.pdf
Schubart, J.: 1982, *Astron. Astrophys.*, **114**, 200.
SEE-GRID Gridification Guide: 2007, Software Installation Management Guide, Aviaeble online at: http://wiki.egee-see.org/index.php/ Software_Installation_Management_Guide
Temizsoylu, T., Zengin, A.: 2006, SEE-GRID-2 Gridification Guideline, Aviaeble online at: http://www.see-grid.eu/ content/modules/downloads/ SEEGRID2-WP4-TR-008-Gridification_Guideline-b-2006-10-19.doc
Xu, C. Z., Lau, F. C. M.: 1997, Load Balancing in Parallel Computers: Theory and Practice. Kluwer, Boston, MA.
Yagoubi, B., Slimani, Y.: 2007, *Journal of Computer Science*, **3(3)**, 186.
Yuasa, M.: 1973, *Publ. Astron. Soc. Japan*, **25**, 399.

ИЗРАЧУНАВАЊЕ СОПСТВЕНИХ ЕЛЕМЕНАТА АСТЕРОИДА НА ГРИД-у

B. Novaković[1], A. Balaž[2], Z. Knežević[1] and M. Potočnik[3,4]

[1]*Astronomical Observatory, Volgina 7, 11060 Belgrade 38, Serbia*
E–mail: *bojan@matf.bg.ac.rs*

[2]*Scientific Computing Laboratory, Institute of Physics Belgrade, Pregrevica 118, 11080 Belgrade, Serbia*

[3]*Faculty of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, 11120 Belgrade, Serbia*

[4]*Belgrade University Computing Center, Kumanovska bb, 11000 Belgrade, Serbia*

УДК 523.44–17
Оригинални научни рад

У раду је описана процедура рачунања сопствених елемената кретања астероида на Гриду. Потреба за убрзањем временски захтевних израчунавања и унапређења њихове ефикасности објашњена је драматичним увећањем количине посматрачких података које се очекује од прегледа неба следеће генерације. Објашњена је природа сопствених елемената и описане су актуелне теорије и методе њиховог израчунавања за различите групе објеката. Сопствени елементи за око 70 000 астероида одређени су од почетка употребе Грид инфраструктуре у ту сврху. Просечно време потребно за ажурирање каталога значајно је скраћено у односу на време потребно за исти посао на једној радној станици. Такође, објашњене су основе функционисања Грида, концепт мидлвера и елементи управљачког и контролног система. Практични кораци, које смо предузели у циљу ефикасне гридификације наше апликације, детаљно су описани. Дати су резултати свеобухватног тестирања перформанси различитих Грид сајтова, и изведени су практични закључци, базирани на резултатима наших тестова, као и на нашем искуству. На крају, предложено је неколико могућности за будући рад.