The ASSIST High Performance Programming Environment for Parallel and Grid Applications

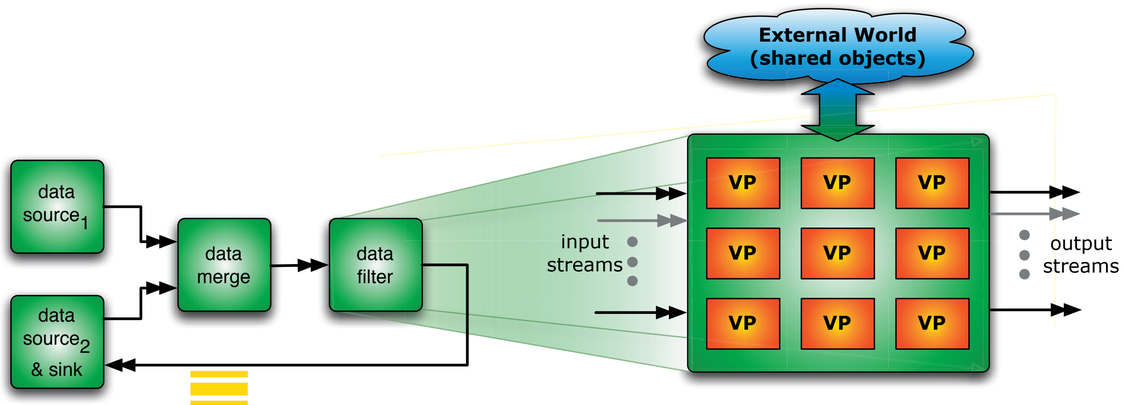
Dipartimento di Informatica
Università di Pisa

Consiglio Nazionale delle Ricerche
ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

ASSIST is a programming environment aimed at providing parallel programmers with user-friendly, efficient, portable, fast ways of implementing parallel applications. It includes a skeleton based parallel programming language (ASSISTcl, cl stands for coordination language) and a set of compiling tools and run time libraries. The ensemble allows parallel programs written using ASSISTcl to be seamlessly run on top of workstation networks supporting POSIX and ACE (the Adaptive Communication Environment) and computational grids.

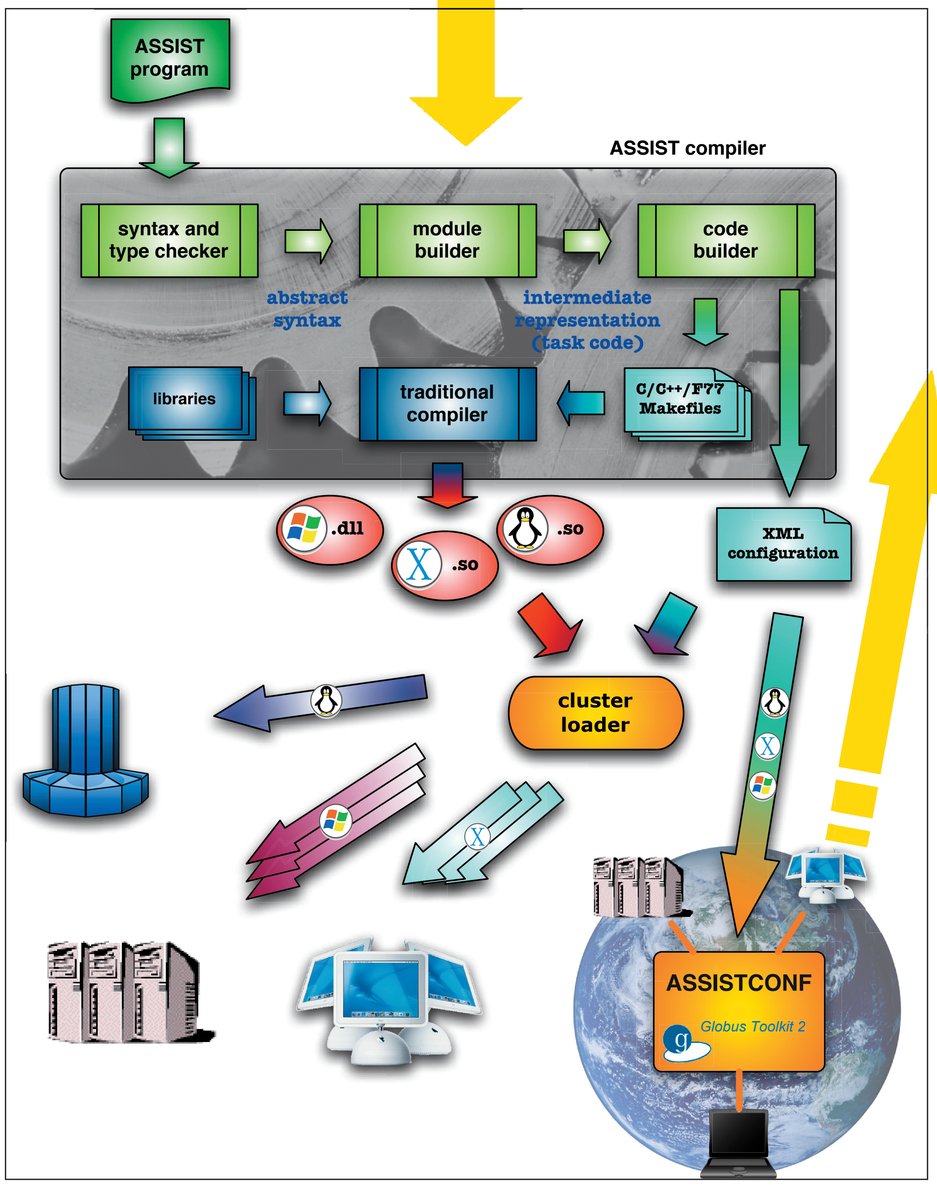
ASSISTCONF is a graphical user interface designed to configure and execute ASSIST applications on Globus-based grids. It hides the programmer the structure of the grid used and provides the interaction between the ASSIST Run Time Support and the Globus middleware (Globus Toolkit 2.4).
The programming environment, the coordination language and the graphical user interface have been designed in a joint project ASI/CNR (Italian National Space Agency and Italian National Research Council), by people of the Dept. of Computer Science of Pisa and of the CNR Information Science and Technologies Institute. Recently, this program terminated and the development of ASSIST has been moved to other italian national research projects (Strategic projects "Legge 449/97" No. 02-00470-ST97 02-00640-ST97 and a FIRB project No. RNBNE01KNFP "GRID.it").

Using ASSISTcl the programmer may structure parallel application as generic graphs of either sequential processes or parallel modules. Nodes in the graphs (i.e. the processes or the parallel modules) are connected by means of data streams. Non-deterministic control is provided to accept inputs from different streams and explicit commands are provided to output items on the output streams. Sequential portions of code can be written using C, C++ or FORTRAN77.



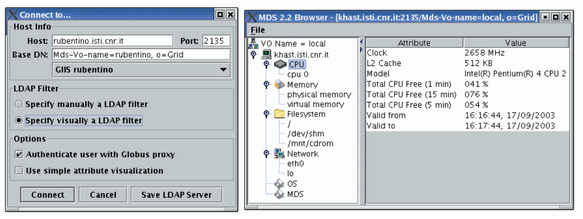
The parmod (the parallel module skeleton) allows to define a set of Virtual Processors, to assign task (to all of them or one task per Virtual processor or one task per partition of Virtual processors), to handle concurrent accesses to state variables, to manage zero or more input stream and zero or more output streams, and to interact with external world accessing (possibly shared) objects via standard object access methods (e.g. CORBA).
A parmod can be specialized to behave as the most common parallelism exploitation pattern/skeleton/design patterns.
Therefore parmods can be used to express farms, pipelines as well as geometric and data parallel computation patterns.

The compiler works on three basic steps: first syntax is parsed and an abstract syntax form is produced. Then a task code (architecture independent parallel abstract code) is produced out of the abstract syntax tree. In this step optimizations are performed aimed at improving program performance and efficiency. Last, POSIX/ACE object code is generated out of the task code, which is suitable to be run onto either a COW through the ASSIST CLAM (Coordination Language Abstract Machine) or a grid through ASSISTCONF interfaces for Globus Toolkit 2 services (GSI, MDS, GRAM, GridFTP). The object code is actually produced using standard C++ compilers. Along with the object code, an XML configuration file is generated, holding all the information needed to run the parallel code. Such information include parallelism degree, mapping of specialized code to processing nodes and the alike.



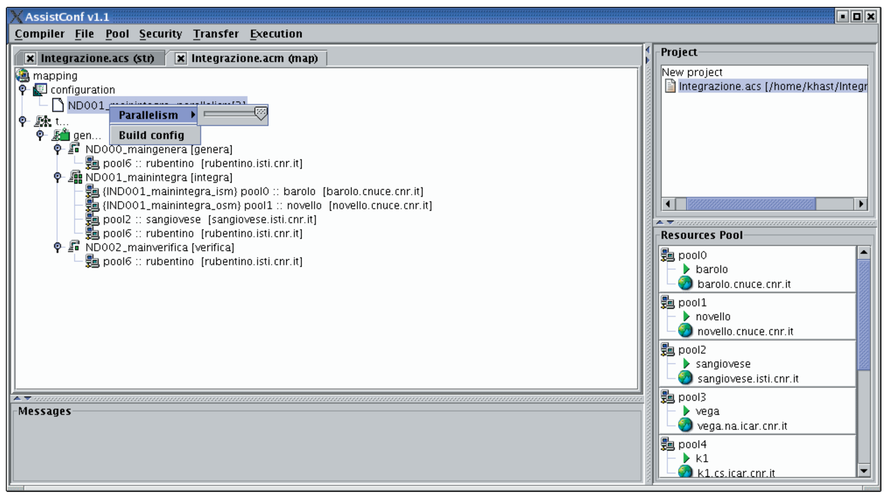
Resource Discovery

In order to carry out a mapping for an application it is needed to select the suitable machines by accessing a list of available machines.
ASSISTCONF can obtain the list and the characteristics of the available machines by accessing the Grid Information System, implemented by the Monitoring and Discovery System (MDS).



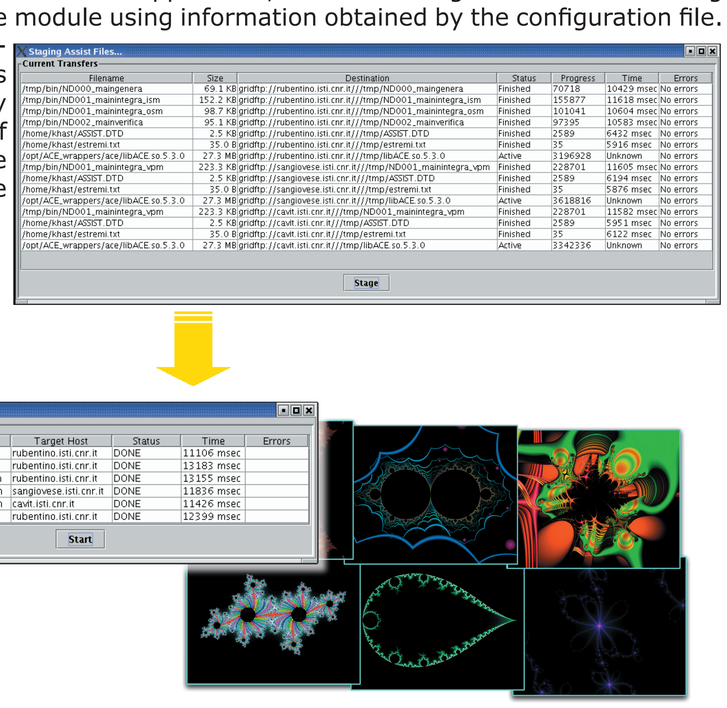
Configuration and Mapping

By accessing the context menus of ASSISTCONF it is possible to specify the parallelism degree of a parallel module and the number of instances of a replicated module. The final step of the configuration process is to establish a mapping between the program modules and the machines in the computational grid.



Staging & Execution

In order to execute an ASSIST application on a grid we exploit the Globus staging and execution mechanisms (GSI, GridFTP, RSL and GRAM APIs). To do this, ASSISTCONF provides functionalities to create and manage a proxy of a valid X.509 certificate. The input files and libraries to be staged can be selected from a list of local files; the executable files are selected by directly accessing the configuration file. To execute the application, ASSISTCONF generates a RSL string for each executable module using information obtained by the configuration file. The execution parallel activation is synchronized by the integration of the GRAM and the ASSIST Run Time Support.



ASSIST features

Programmability

Skeleton and coordination technology are exploited in the ASSIST environment in such a way the programmer is not required to handle most of the error prone details he is usually concerned with (processes and communications setup, scheduling, mapping, etc.). The skeletons included in ASSISTcl are far more powerful that the traditional ones. The programmer can therefore implement parallel applications with complex parallelism exploitation patterns.

Code reuse

Sequential portions of code embedded in ASSISTcl programs can be written in C, C++ and FORTRAN, thus enhancing the possibility to reuse existing code.

Rapid prototyping

Programmer can experiment different parallelization strategies just changing a few lines of code and recompiling.

Interoperability

ASSISTcl programs can access external objects via CORBA. A whole ASSISTcl program can be automatically exported (i.e. standard IDL can be automatically generated and proper skeleton code is generated) as a CORBA object to the external world. Furthermore, facilities are present in the language that allow to use external libraries from within the sequential portions of code in the ASSISTcl programs.

Portability

ASSIST has been currently implemented on POSIX/ACE Linux workstation networks. We are currently testing the compiler part needed to target heterogeneous architectures.

Performance

ASSISTcl benchmarks demonstrated good (close to ideal) scalability and efficiency on Linux clusters. Efficiency close to 99% has been achieved using medium to coarse grain parallel code.

Grid Targeting

The ASSISTCONF main functionalities are aimed to:

- *select the computational resources needed to run an application*
- *configure an ASSIST application*
- *assist the user to establish a mapping of the various modules on the selected computational resources*
- *stage on the selected computational resources the libraries, executable modules and input files neede to run the ASSIST application*
- *activate the modules execution*
- *transfer the output files to the user machine and delete, if required, all the files used to run the application from the grid resources.*

ASSIST has been mostly designed by people of the Dept. of Computer Science, University of Pisa, Italy. Among the others, the following people contributed in either ASSIST(cl) design or implementation: M. Aldinucci, S. Campa, P. Ciullo, M. Coppola, M. Danelutto (project leader), D. Guerri, D. Laforenza, M. Lettere, S. Magini, S. Orlando, A. Paternesi, R. Perego, P. Pesciullesi, A. Petrocelli, E. Pistoletti, L. Potiti, R. Ravazzolo, M. Torquati, L. Vaglini, P. Vitale, M. Vanneschi (group leader), G. Virdis, C. Zoccolo.
ASSISTCONF has been mostly designed by people of the Information Science and Technologies Institute of the Italian National Research Council (ISTI-CNR). Among the others, the following people contributed: R. Baraglia, D. Laforenza (group leader), T. Fagni, R. Ferrini, F. Furfari, P. Ciullo, S. Orlando, R. Perego, F. Silvestri, P. Pesciullesi, N. Tonellotto, M. Vanneschi.