# AMEEDA: A General-Purpose Mapping Tool for Parallel Applications on Dedicated Clusters*

X. Yuan[1], C. Roig[2], A. Ripoll[1], M.A. Senar[1], F. Guirado[2], and E. Luque[1]

[1] Universitat Autònoma de Barcelona, Dept. of CS
xiaoyuan@aows10.uab.es, a.ripoll@cc.uab.es,
miquelangel.senar@uab.es, e.luque@cc.uab.es
[2] Universitat de Lleida, Dept. of CS
roig@eup.udl.es, fernando@eup.udl.es

**Abstract.** The mapping of parallel applications constitutes a difficult problem for which very few practical tools are available. AMEEDA has been developed in order to overcome the lack of a general-purpose mapping tool. The automatic services provided in AMEEDA include instrumentation facilities, parameter extraction modules and mapping strategies. With all these services, and a novel graph formalism called TTIG, users can apply different mapping strategies to the corresponding application through an easy-to-use GUI, and run the application on a PVM cluster using the desired mapping.

## 1 Introduction

Several applications from scientific computing, e.g. from numerical analysis, image processing and multidisciplinary codes, contain different kinds of potential parallelism: task parallelism and data parallelism [1]. Both data and task parallelism can be expressed using parallel libraries such as PVM and MPI. However, these libraries are not particularly efficient in exploiting the potential parallelism of applications. In both cases, the user is required to choose the number of processors before computation begins, and the processor mapping mechanism is based on very simple heuristics that take decisions independently of the relationship exhibited by tasks. However, smart allocations should take these relationships into account in order to guarantee that good value for the running time is achieved.

In general, static mapping strategies make use of synthetic models to represent the application. Two distinct kinds of graph models have been extensively used in the literature [2]. The first is the TPG (Task Precedence Graph), which models parallel programs as a directed acyclic graph with nodes representing tasks and arcs representing dependencies and communication requirements. The second is the TIG (Task Interaction Graph) model, in which the parallel application is modeled as an undirected graph, where vertices represent the tasks and

* This work was supported by the MCyT under contract 2001-2592 and partially sponsored by the Generalitat de Catalunya (G. de Rec. Consolidat 2001SGR-00218).

B. Monien and R. Feldmann (Eds.): Euro-Par 2002, LNCS 2400, pp. 248–252.

edges denote intertask interactions. Additionally, the authors have proposed a new model, TTIG (Temporal Task Interaction Graph) [3], which represents a parallel application as a directed graph, where nodes are tasks and arcs denote the interactions between tasks. The TTIG arcs include a new parameter, called degree of parallelism, which indicates the maximum ability of concurrency of communicating tasks. This means that the TTIG is a generalized model that includes both the TPG and the TI G.

In this work, we present a new tool called AMEEDA (Automatic Mapping for Efficient Execution of Distributed Applications). AMEEDA is an automatic general-purpose mapping tool that provides a unified environment for the efficient execution of parallel applications on dedicated cluster environments. In contrast to the tools existing in the literature [4] [5], AMEEDA is not tied to a particular synthetical graph model.

## 2 Overview of AMEEDA

The AMEEDA tool provides a user-friendly environment that performs the automatic mapping of tasks to processors in a PVM platform. First, the user supplies AMEEDA with a C+PVM program whose behavior is synthesized by means of a tracing mechanism. This synthesized behavior is used to derive the task graph model corresponding to the program, which will be used later to automatically allocate tasks to processors, in order to subsequently run the application. Figure 1 shows AMEEDA's overall organization and its main modules, together with the utility services that it is connected with, whose functionalities are described below.

### 2.1 Program Instrumentation

Starting with a C+PVM application, the source code is instrumented using the TapePVM tool *(ftp://ftp.imag.fr/pub/APACHE/TAPE)*. We have adopted this technique, in which instructions or functions that correspond to instrumentation probes are inserted in users' code before compilation, because of its simplicity. Using a representative data set, the instrumented application is executed in the PVM platform, where a program execution trace is obtained with TapePVM and is recorded onto a trace file.

### 2.2 Synthesized Behaviour

For each task, the trace file is processed to obtain the computation phases where the task performs sequential computation of sets of instructions, and the communication and synchronization events with their adjacent tasks. This information is captured in a synthetic graph called the Temporal Flow Graph (TFG).

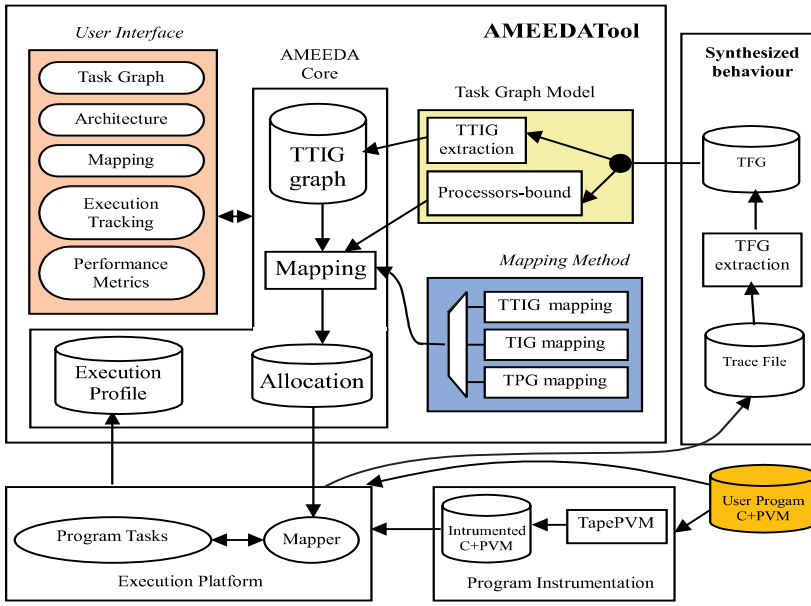


**Fig. 1.** Block diagram of AMEEDA.

### 2.3 AMEEDA Tool

With the synthesized behavior captured in the TFG graph, the AMEEDA tool executes the application using a specific task allocation. The necessary steps to physically execute the application tasks using the derived allocation are carried out by the following AMEEDA modules.

**1. Task Graph Model**

Starting from the TFG graph, the TTIG model corresponding to the application is calculated. Note that, although different traces may be collected if an application is executed with different sets of data, only one TTIG is finally obtained, which captures the application's most representative behavior.

The *Processors-bound* sub-module estimates the minimum number of processors to be used in the execution that allows the potential parallelism of application tasks to be exploited. This is calculated using the methodology proposed in [6] for TPGs, adapted to the temporal information summarized in the TFG graph.

**2. Mapping Method**

Currently, there are three kinds of mapping policies integrated within AMEEDA that can be applied to the information captured in the TTIG graph of an application.

- *(a) TTIG mapping.* This option contains the MATE (Mapping Algorithm based on Task Dependencies) algorithm, based on the TTIG model [3]. The assignment of tasks to processors is carried out with the main goal of joining the most dependent tasks to the same processor, while the least-dependent tasks are assigned to different processors in order to exploit their ability for concurrency.
- *(b) TIG mapping.* In this case, allocation is carried out through using the CREMA heuristic [7]. This heuristic is based on a two-stage approach that first merges the tasks into as many clusters as number of processors, and then assigns clusters to processors. The merging stage is carried out with the goal of achieving load balancing and minimization of communication cost.
- *(c) TPG mapping.* Allocation is based on the TPG model. In particular, we have integrated the ETF heuristic (Earliest Task First) [8], which assigns tasks to processors with the goal of minimizing the starting time for each task, and has obtained good results at the expense of relatively high computational complexity.

**3. User Interface**

This module provides several options through a window interface that facilitates the use of the tool. The *Task Graph* sub-module allows the information from the TTIG graph to be visualized. The *Architecture* sub-module shows the current configuration of the PVM virtual machine. The execution of the application, with a specific allocation chosen in the *Mapping* option, can be visualized by using the *Execution tracking* submodule that graphically shows the execution state for the application. The *Mapping* can also be used to plug-in other mapping methods. Finally, the *Performance* option gives the final execution time and speedup of a specific run. It can also show historical data recorded in previous executions in a graphical way, so that performance analysis studies are simplified. Figure 2 corresponds to the AMEEDA window, showing the TTIG graph for a real application in image processing, together with the speedup graphic generated with the Performance sub-module, obtained when this application was executed using the PVM default allocation and the three different mapping strategies under evaluation.

## 3 Conclusions

We have described the AMEEDA tool, a general-purpose mapping tool that has been implemented with the goal of generating efficient allocations of parallel programs on dedicated clusters. AMEEDA provides a unified environment for computing the mapping of long-running applications with relatively stable computational behavior. The tool is based on a set of automatic services that instrumentalize the application and generate the suitable synthetic information. Subsequently, the application will be executed following the allocation computed by AMEEDA, without any user code re-writing. Its graphical user interface constitutes a flexible environment for analyzing various mapping algorithms and

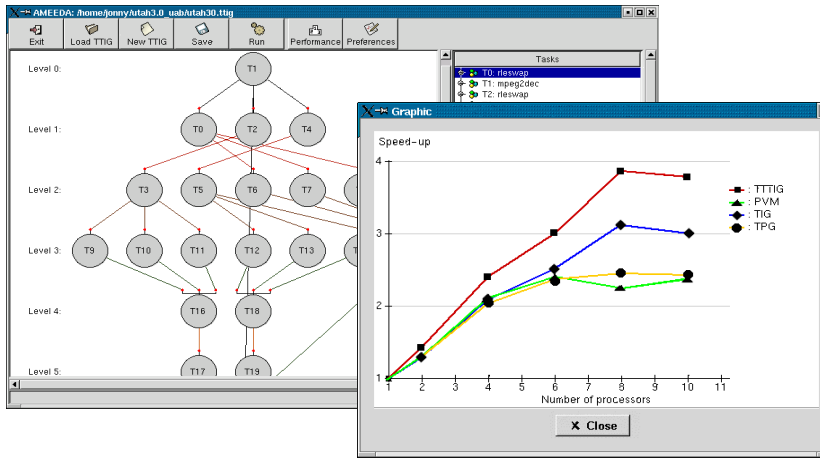


**Fig. 2.** AMEEDA windows showing the TTIG graph and the speedup for a real application.

performance parameters. In its current state of implementation, the graphical tool includes a small set of representative mapping policies. Further strategies are easy to include, which is also a highly desirable characteristic in its use as a teaching and learning aid for understanding mapping algorithms. As future work, AMEEDA will be enhanced in such a way that the most convenient mapping strategy is automatically chosen, according to the characteristics of the application graph, without user intervention.

## References

1. Subhlok J. and Vongran G.: Optimal Use of Mixed Task and Data Parallelism for Pipelined Computations. J. Par. Distr. Computing. vol. 60. pp 297-319. 2000.
2. Norman M.G. and Thanisch P.: Models of Machines and Computation for Mapping in Multicomputers. ACM Computing Surveys, 25(3). pp 263-302. 1993.
3. Roig C., Ripoll A., Senar M.A., Guirado F. and Luque E.: A New Model for Static Mapping of Parallel Applications with Task and Data Parallelism. IEEE Proc. of IPDPS-2002 Conf. ISBN: 0-7695-1573-8. Apr. 2002.
4. Ahmad I. and Kwok Y-K.: CASCH: A Tool for Computer-Aided Scheduling. IEEE Concurrency. pp 21-33. oct-dec. 2000.
5. Decker T. and Diekmann R.: Mapping of Coarse-Grained Applications onto Workstation Clusters. IEEE Proc. of PDP'97. pp 5-12. 1997.
6. Fernandez E.B. and Bussel B.: Bounds on the Number of Processors and Time for Multiprocessor Optimal Schedule. IEEE Tr. on Computers. pp 299-305. Aug. 1973.
7. Senar M. A., Ripoll A., Cortés A. and Luque E.: Clustering and Reassignment-base Mapping Strategy for Message-Passing Architectures. Int. Par. Proc Symp&Sym. On Par. Dist. Proc. (IPPS/SPDP 98) 415-421. IEEE CS Press USA, 1998.
8. Hwang J-J., Chow Y-C., Anger F. and Lee C-Y.: Scheduling Precedence Graphs in Systems with Interprocessor Communication Times. SIAM J. Comput. pp: 244-257, 1989.