A Visual Model-Driven Rapid Development Toolsuite for Parallel Applications

Xu Zhen, Sun Jizhou, Yu Ce∗, Wu Huabei, Meng Xiaojing, Tang Shanjiang
School of Computer Science and Technology, Tianjin University, Tianjin 300072, China
xuzhen@tju.edu.cn

∗corresponding author.
Email Address: yuce@tju.edu.cn

Abstract
Modeling and programming parallel applications is becoming unavoidable for next generation of software architects and programmers, owing to the popularization of multi-core processors. ParDT is a graphical model-driven development toolsuite which supports not only modeling of parallel programs on high abstraction level but the translation of the constructed models into source code skeletons according to the specific runtime environments and libraries. The process of translation, which involves the parsing of graphical models and the generation of source code skeletons aimed at different parallel platforms, are explained in detail. ParDT is implemented based on Eclipse and compatible with its plug-in architecture. The toolsuite manages to help programmers relieve the burden of building parallel applications.

1. Introduction
The development of methodology and integrated development environments for designing serial computing programs have reached quite maturity period. Similar work is also encouraged for programming parallel applications since the chips with multi-core and many-core processors[1] have been widely applied in most personal computers and workstations, to make full use of computing capability and markedly reduce the running time on such parallel computing platforms.

However, parallel programming, which involves task division and low level interaction between the subtasks, is usually difficult for most programmers who are only familiar with serial programming. Although the current parallel programming libraries such as MPI, PVM, OpenMP and Pthread helps programmers to develop parallel applications in a higher abstraction level, the process of parallel programming is still inefficient and error-prone because of the unavoidable specific programming details.

A rapid development toolsuite called ParDT (Parallel Development Toolsuite) is introduced in this paper, which greatly reduces the complexity of developing parallel programs. ParDT is designed to work with a graphical editor by which high level model is at first visually constructed to describe the parallel application. ParDT provides a suite of components for mainly translating graphical models into source code files and then deploying them conveniently, according to different parallel runtime environment and libraries. The process of translation is divided into two important phases- parsing graphs and code generation, which are connected by a shared data structure.

2. Related Work
A number of programming models, libraries, languages, and tools for developing parallel programs have been brought forward and implemented, such as MPI, OpenMP, HPF, etc. On the other hand, the research work on the general methodology, such as programming template, parallel design patterns, architectural skeleton[2] and algorithm skeleton[3], is ongoing. However, these approaches can not manage to help programmers avoid dealing with the low level parallel programming details.

Several development environment have been implemented. PTP project[4] is based on the Eclipse platform, aimed at integrating the programming, compiling, debugging functions to develop MPI programs, without any concerns on graphical modeling. P-GRADE[5] is an integrated graphical environment.
which supports the whole lifecycle of parallel program development for supercomputers and heterogeneous workstations clusters based on PVM, but it only runs under UNIX systems. P-GRADE can describe elements of the program graphically in the graphical editor named GRED[6] and then parse the program graph to generate source codes used for P_GRADE only.

Paradigms mentioned above try to simplify the development of parallel applications via providing graphical user interfaces or weakly supporting to parse graphs. By contrasting with them, ParDT has got the stronger functions of parsing graphical models as well as translating models into source code skeletons, which essentially reduce the complexity of parallel programming.

3. Features of ParDT

As a rapid development tool suite, ParDT works with a graphical editor which provides the definition of a set of graphical symbols, which can be extended in the future, to help programmers design and describe parallel applications and algorithms. The constructed models can be automatically translated into the source code skeletons according to different parallel runtime environment.

3.1 Architecture of ParDT

ParDT is a key component of an ongoing project named EasyPAB [7], which is an extensible IDE framework designed for both message-passing-based and shared-memory-based parallel computing infrastructures, and supports different parallel programming model. Architecture of ParDT, as an important part of EasyPAB architecture, is shown in figure 1.

EasyPAB_UI provides a graphical editor to construct models visually and the editor is designed based on MVC architecture, where the models are simplified description of parallel applications. A model[8] is displayed in the form of diagrams, each of which can be considered to be the specific projection of the View.

Construction Blocks(CBs) are the essential elements to represent diagrams which can be extended in the future. Different instance of CBs are connected and organized to build a new diagram, with the restriction of the constraints set aimed at the diagram.

A set of graphical symbols is designed to represent CBs graphically.

In EasyPAB architecture, ParDT covers the middle layer of EasyPAB and collaborates with the graphical editor in the top layer.

Graphical Model manager interacts with the graphical editor, at the same time stores and manages the constructed model. Constraints Validator checks if the connections of CBs in those diagrams follow the pre-defined constraint conditions. Model Translating Engine is to translate the information about the graphical models into different source code files automatically. Executable Code Deployer deploys the binaries of compiled applications to physical computing nodes.

3.2. The Approach of Building Parallel Programs in ParDT

According to this development tool suite, the process of building parallel programs is mainly divided into following phases — the graphical program modeling, the platform specific configuration, generation of source code skeletons, and deploy of executable codes. Figure 2 shows the process of...
building parallel programs in ParDT tools. Each of the phases is aided by the related component of the tool suite which is loosely coupled with other components. That is to say, ParDT brought forward an approach of building parallel programs:

- **Design the programs in a high abstraction level, namely model level;**
- **The whole process is driven by the constructed graphical models, for program management and code generation;**
- **Little amount of coding work is needed, reduce the workload of programmers.**

3.3 Translating Graphical Models into Source Files

Figure 3 shows the process of translating graphical model into executable source files. Model Translating Engine is the key component of ParDT, which is designed to parse the graphical symbols of diagrams that are the views of the parallel application models in the graphical editor, and then to translate the useful model information that is extracted from the earlier step into different types of code skeletons, according to the specific parallel runtime environment and libraries.

First step of this process is accomplished by the graphical editor. Before Model Translating Engine begins to work, the graphical editor creates a textual editor file that describes the whole application developed by the programmer. This file contains not only descriptions of parallel application models but the information about diagrams, such as location of the diagram and the graphical symbols in it. Such information is necessary to be recorded in that it may be requested by other components of ParDT, for example, Constraints Validator.

The editor file is saved as XML format, and the syntax definition of the file follows specifications of the attributes of the graphical symbols and the constraints of the connections between graphical symbols. It also can be shown in Figure 3 that the editor file is both the source of the finally generated source codes and the input of the following process of translation which can be divided into two phases.

3.3.1 The Phase of Parsing Graphs. The first phase of translation is to parse different kinds of graphs separately to obtain the detailed information about the structure of these graphs, which is then kept in the built internal data structures.

A graph is the simplified abstract representation of a matched diagram. It preserves the complete structural information of the diagram, and discards the inessential information, e.g. displaying details of it in the screen.

In this phase, graph parser firstly traverses the editor file created by the graphical editor and collect the specifications of a species of diagrams from it. Afterwards, an action of abstraction is triggered, that is to say, the representation of the diagrams is simplified to the directed graphs, namely Process Graph, Activity Graph, Message Graph and so on. A graph preserves complete structural information of the diagram. A graphical element in the diagram is represented as a vertex in the directed graph, and the basic attributes of graphical elements are preserved as the member variables of vertexes in the graph.

In the Process Graph, a vertex represents a process and an edge represents the communication between the two processes. Similarly, a vertex in the Activity Graph represents a sequential executing unit and an edge represents the transition of executing states.

An important work of the translation is to design internal data structures that are prepared for the translators aimed at different parallel platforms and libraries. Therefore, the data structures should have the following characteristics: suitable to represent directed graphs, of convenience for traversing a graph, inserting vertexes and merging two graphs.

An improved linked list, with constraints on the nodes of linked list, is defined as the data structure of.
graphs. As is shown in Figure 3, three data structures, namely PGraph, AGraph and MGraph, which differ from each other mainly on the constraints are designed aimed at the three kinds of graphs.

3.3.2 The Phase of Code Generation. The second phase of translation is to generate source code files from the obtained instantiation of the data structure, according to the different runtime environment and libraries.

It is obvious that not all the information that is extracted from the graph data structures contributes to the generation of source code files. For example, the description of the process topology in the Process Graph is not used when MPI programs are generated, and usually it only help programmers to analyze and to understand parallel applications.

Besides, the generation of source code files requires the extracted useful information from more than one graph. As is shown in Figure 3, all the three graph is connected to the translator for PVM programs as the source of information applied to the source code generation.

4. Implementation

ParDT is based on Eclipse[9] plug-in architecture and GEF (Graphical Editing Framework), which provides rich supports for building graphical editor plug-ins.

![Figure 4. Example: matrix multiplication algorithm](image)

After graphical models are constructed, it can be saved in form of XML files as the editor files, or can try to generate source code skeletons for specific runtime environment. Figure 4 shows an example of modeling matrix multiplication algorithm visually in the ParDT graphical editor. Figure 5 shows the C source codes with MPI libraries generated according to the created model.

![Figure 5. The C source codes with MPI libraries](image)

Up to now, ParDT has passed the tests on IBM Cluster1350 (a suit of Linux Cluster) using MPI, and on single servers equipped with Intel/AMD dual-core processors (Windows/Red Hat Linux) using Win32Thread/Pthread.

5. Conclusion and Future Work

ParDT helps to relieve the burden of building parallel programs, by allowing programmers making use of graphical modeling user interface provided by the graphical editor, as well as automatic code skeleton generation mechanism and so on. Developers can focus on the higher level abstract and solution of applications and needn’t care of the complexity of traditional parallel computing technologies.

The implement of ParDT is based on Eclipse and GEF, which are both open source project, so everyone can improve it with new ideas and applied new research conclusions to it.

ParDT is still under development and desiderates more supplements and improvements. In the future work, the definitions of construction blocks, along with the rules of constraints in various diagrams, will be organized into a graphical language – Parallel Modeling Language (PML), and corresponding compilers will be designed and implemented by adding supplemental features to ParDT.

Acknowledgements

The work is supported by Application Fundamentals & Frontier Technologies Key Research Program of Tianjin, China(NO. 08JCZDJC19700)
References


