METHODS AND MEANS OF PREDICTING THE RUN TIME OF SERIAL PROGRAMS

D. V. Kalinichenko, A. P. Kapitonova, and N. V. Yushchenko

We study methods of predicting the run time of serial programs on modern computers. To solve this problem we propose a combined static-dynamic approach to program analysis. Along with methods of prediction we study means of describing the computer architecture. Particular attention is paid to the specifics of the problem of prediction for the class of RISC computers.

Three figures. Bibliography: 16 titles.

The present article is devoted to the problem of estimating the run time of programs. The possibility of solving this problem is of interest both for ordinary users and for developers of new architectural solutions in the area of computing systems. The choice of a suitable computing system for implementing a set of algorithms, problems of effective parallelization of programs, and the study of the behavior of programs, development and evaluation of new architectures—all these problems are connected with estimating the run time.

Particular interest in the problem of estimating the time has arisen in the past decade in connection with the widespread application of real-time computing systems. Such systems, as a rule, participate in controlling a certain real process and must possess one extremely important property: the computer must not only carry out the computation correctly, but must also do so within a prescribed time interval. A system that fulfills a task later than the established time boundary is regarded as malfunctioning. Therefore the possibility of obtaining data on the run time of programs has become not merely a desirable condition but a necessary condition for developers of such systems at the design stage.

An estimate of run time is usually taken to mean one of the following:

• estimate of the time by an accurate measurement of certain functional parameters of the computing system;
• prediction of the run time of programs without performing them on the computer being studied.

The first approach usually entails observing the execution of the program, keeping a trace, and constructing program and equipment emulators [1, 2, 3]. Thus it always involves the study of the dynamics of the program. The merit of dynamic methods is the precision of the estimates obtained. However it is not always possible to apply them for the following reasons:

• the absence of a real computer;
• the implementation of an emulator of the computer being studied is an expensive and laborious process;
• a significant retarding (10–1000 times) of run time of the program due to the high cost of measurement.

Prediction (or prognosis) of the run time of a program usually presumes the use of various static methods and also simulation modeling of the functioning of the equipment of the computing system being studied.

In the present paper the estimation of the run time will be understood in the sense of prognosis.

More specifically, in this paper we propose a version of solving the following problem: from the text of a program, information on the behavior of the program, and a description of the architecture of the computer, determine the run time of the program on this computer.

To solve this problem the method of complex static-dynamic prognosis was developed. The proposed
methodology of prognosis was implemented as an instrumental system for estimating the run time of a program in the context of the DYANA project [4].

1. The problem of estimating time. When solving the problem of time estimation, it is necessary to specify the following:
   - the programs being evaluated;
   - the class of computers for which the prognosis is being performed;
   - the time being computed.

   The methodology discussed in this paper is intended for analyzing serial programs (or serial fragments of computations). The resulting value of time is the time spent by the computing system in executing the commands of the program being studied. This means that the actions of the operating system are not evaluated, nor are input/output operations.

   The class of computers studied has a significant effect on the method of estimating time. The method of solving the prognosis problem in the class of serial computers of von Neumann type was studied by us earlier [5, 6]. In these papers we introduced the model of a serial register-flow computer and discussed a method of prognosis of execution time for serial fragments of computations by constructing model code of minimal length, and we described the architecture of an instrumental system for estimating the execution time of programs. In the context of the same study we developed a language for describing the architecture of a serial computer [4], making it possible to fix the structure of the computer through the set of parameters and prescribe the time values for executing commands. Such a method of describing the architecture made it possible to make the instrumental system for estimating the time multipurpose, that is, attuned to a specific computer of the class in question.

   In the present paper the problem of estimating time is studied for the class of RISC processors. We first discuss the ideas of the static/dynamic approach applied. We then study the characteristics of prognosis of the execution time of programs on RISC computers connected via pipelines and the use of cache-memory. Finally we give a brief description of an instrumental system for estimating the time multipurpose, that is, attuned to a specific computer of the class in question.

   In the present paper the problem of estimating time is studied for the class of RISC processors. We first discuss the ideas of the static/dynamic approach applied. We then study the characteristics of prognosis of the execution time of programs on RISC computers connected via pipelines and the use of cache-memory. Finally we give a brief description of an instrumental system for estimating the time multipurpose, that is, attuned to a specific computer of the class in question.

2. Methods of time prognosis. There exist various methods of prognosis of the run time of programs. The dynamic method consists of running the program to be evaluated on an emulator of the computer being studied. The estimate obtained in this way is the most precise, but requires large expenditures of time and memory, and is therefore inapplicable in the majority of cases. Prediction (or prognosis) of the run time of a program usually entails the use of various static methods, as well as simulation of the functioning of the devices in the computer system being studied.

   Static methods make possible a prognosis of the run time of a program on the basis of analysis of the text of the program without a full simulation of the work of the processor. Depending on the purpose of the prognosis various estimates may be required: best case execution time, worst case execution time, time for an arbitrary history of execution of the program. In the first two cases the critical point of static prognosis is the development of an algorithm for finding the best (or worst) path for executing the program with a subsequent mapping of the actions from this path to the computer devices. It should be noted that this is the approach that is at present being intensively studied in application to the software for real-time systems [7, 8, 9], since in this case the most important thing is a guaranteed upper limit on the time. In the present paper we shall describe a combined static-dynamic method of prognosis designed for predicting the run time of a program on a specific path, that is, in the case when the set of initial data is fixed.

2.1. Characteristics of the static-dynamic approach. For the static-dynamic approach the use of a two-step method of estimation is characteristic. The first, static, stage of the estimate begins by partitioning the program being analyzed into linear parts. A linear part will be defined as an "indivisible" fragment of a program with one input or one output, that is, a set of commands that are always performed together. In distinguishing the linear parts we perform a mark-up of the program being analyzed, consisting of the addition of a special-