CAPTools - semiautomatic parallelisation of mesh based computational mechanics codes

M.Cross, C.S.Ierotheou, S.P.Johnson and P.F.Leggett

Parallel Processing Research Group, University of Greenwich, London SE18 6PF, UK

Abstract. An overview of a suite of Computer Aided Parallelisation Tools (CAPTools) is presented for the parallelisation of FORTRAN based structured mesh computational mechanics codes. In recognition of the limitations of automatic compilers, CAPTools is a semiautomatic, interactive system which embeds user supplied knowledge into its detailed dependence analysis. Applying the decomposition paradigm, then the addition of masks and communication calls yields generic parallel source code for a DMS.

1 Introduction

Despite the attractive performance measures of distributed memory systems (DMS) they have not had the impact on the HPC community that might have been anticipated. The main reason for the slow take-up of parallel technology has been the difficulty of implementing existing codes on such systems efficiently. This effort has been fraught with difficulty and the task of implementing and testing a parallelisation strategy was a tedious and error prone task.

In an ideal world the implementation of software on a parallel system would require no more effort than is currently required on a conventional scalar system. In the past decade a number of research programmes have pursued the concept of parallelisation compilers; the most notable examples include - Paraphrase, a multi-pass paralleliser developed by Kuck et al [1]; The KAP paralleliser [2] using the theory of dependence analysis and its application to parallel code generation; Parallel Fortran Converter (PFC) developed by Kennedy et al [3] and work on the SUPERB compiler by Zima et al [4].

Parallel compilers based upon the above research are very powerful in some respects and extremely restricted in others. The core of any parallelising compiler is the dependence analysis and many decisions in the analysis involve a knowledge of program variables, about which no information is explicitly available. As such, this results in the conservative assumption of a data dependence which could influence adversely the quality of the consequent parallel version. New parallel languages may smooth the route to parallel software; unfortunately, it does not relieve the programmer from many of the tasks involved in the development and implementation of parallelisation strategies. Moreover, new languages do not help the vast scientific/engineering community with a huge array of existing FORTRAN software.

The alternative to full parallelising compilers or new languages is to use a set of
tools to assist and, as far as is practical, to automate the parallelisation process. Recently, the FORGE90 parallelisation analysis system has been launched. The effectiveness of FORGE90 is based on the power of its analysis. Its current version is constrained in a number of ways. For example, its DMS parallelisation is based on localised loops with independent data decompositions, and little communications optimisations. CAPTools (Computer Aided Parallelisation Tools) is also based on the fact that the process of code parallelisation can not, in general, be fully automated [5,6,7,8,9,10]. Information from the programmer is vital during the process of parallelisation and, therefore, a dialogue is an essential component. At the heart of CAPTools is a very sophisticated interprocedural dependence analysis based upon advanced symbolic algebra techniques, combined with a powerful inference engine to elicit and embed user supplied information in the knowledge base. It is the dependence analysis which enables the generation of high quality parallel codes which are almost as efficient as any hand parallelisation.

2 Overview of CAPTools Structure

The main aims of CAPTools are the following:-

- No end-user knowledge of parallel execution.
- Parallelise the entire code, not just the solver.
- There should be no loss in efficiency in using CAPTools as opposed to a manual parallelisation.
- There is a need for essential user input to assist in the parallelisation.
- The generated parallel code should be generic and similar to the serial code.

Figure 1 shows an overview of the structure of CAPTools. The main components of the tools comprise:-

- Detailed control and dependence analysis.
- User specification of a parallelisation strategy.
- Automatic insertion of parallel control and dataflow structures.
- Automatic adaption of serial code to produce parallel code.
- Code optimisations e.g. loop splitting, loop interchanging, etc.

Although CAPTools is designed for any application, it is currently used in the field of computational mechanics. The current version uses a mesh decomposition for both structured and unstructured meshes, and the code generated is for a DMS.

2.1 Summary of Dependence Analysis

The dependence analysis can be as basic or as extensive as the paralleliser desires. The options available are:- To perform symbolic variable equality tests; Perform tests to set the exact dependence attribute to dependencies; A full interprocedural analysis to transfer information across routine boundaries; Use of the symbolic inequality disproof algorithm together with knowledge provided from the user, extracted from the control flow and assumed as part of the extended Banerjee test; Use the inference engine to exploit more complex knowledge from the control flow etc; Trace variables through conditional assignments.