Static Communications in Parallel Scientific Programs

Damien Gautier de Lahaut, Cécile Germain

LRI ura 410 du CNRS
Université Paris XI, 91405 Orsay cedex

Abstract. Most massively parallel architectures exhibit a large gap between hardware capacities and actual communication performance. Dynamic routing is the major cause of this loss of efficiency, because the interconnection network and the processing elements must be quite loosely coupled. The first part of this paper presents experimental analysis of communications in parallel scientific programs, showing that most communication patterns of application programs are determined at compile-time. In the second part, we sketch an execution model intended to exploit this knowledge, under very general assumptions about the underlying interconnection network, and we prove that the model gracefully degrades with the growing complexity and dynamicity of the communication patterns.

1 Introduction

From the very beginning of parallel architectures, a recurrent problem is the large gap between peak and actual performance. Despite the progress in hardware and software, most recent experimental studies [1, 2, 3] show that the actual performance usually remains between one and two orders of magnitude beyond the peak. One major cause of this sobering fact is data transfer and especially the interconnection network. For instance, considering data from [3], the best performance figures are reached by programs that have the lowest remote data access to floating point operations ratio. Although communications seem to be the bottleneck of parallel architectures, not much is known about the characteristics of the communications created by parallel programs. The first purpose of this paper is to give some experimental results about the statistical distribution of communication patterns. The communications that can be known at compile-time will be called static in the following, and those that can only be determined at run-time will be called dynamic. The point being statistics, a significant benchmark set has been studied; this set amounts to around 25,000 lines, is written in various dialects of parallel FORTRAN, and is composed of two parts: the first is a set of scientific parallel codes, partially hand-written and partially generated by automatic parallelization; the second part is a subset of library routines from LAPACK. The dynamic (run-time) occurrences of both static and dynamic communication schemes have been gathered. The main result is that static communications are nearly exclusive in parallelized codes and dominant in user programs, whereas the situation is much more complex in library routines.
We are interested in this taxonomy (static/dynamic) not for classification purpose, but because a considerable speedup in parallel computations can be achieved by a careful exploitation of the compile-time information about static communications. In fact, a parallel execution model where the communications are computed at compile-time can reach the hardware raw performance for the most frequently used static communication schemes. This contrasts with the actual communication performance of most parallel architecture, which is dominated by the communication protocol overhead. However, the overall speedup must take into account the contributions of all communication types, both static and dynamic (Amdhal's law). The point is then to assess the penalty of compiling the dynamic communications. This is a very difficult task, because many factors are involved, and it is almost impossible to quantify their respective impacts and cross-effects. Nevertheless, meaningful results can be derived by evaluating, for broad classes of communication schemes, the speedup achieved on each of them by the static execution model. As a testbed, we compare the CM-5 communication figures with the expected performance of the static model. The speedup is two orders of magnitude in the best case, while the performance is divided only by two in the worst case.

The rest of this paper is organised as follows. The first part discusses dynamic routing, the basic communication mechanism of almost all parallel architectures, and the background in compiled communications. The second part presents a classification of communication schemes. The third part is devoted to the experiments, methodology and results. Finally, we assess the cost of emulating dynamic communications in the static model and we present the expected performance.

2 Background

2.1 Dynamic routing

Almost all massively parallel architectures do use asynchronous dynamic routing. Dynamic routing means that the routing circuits in each network node determine the path of each message at run-time. This costs extra hardware, the routing circuits, and network bandwidth, the address header carried by each message. The routing is asynchronous in the following sense: the latency of the messages depends on the network load, thus is not known; a processor/network interface is necessary to adapt the message threads. The overhead of this interface is enormous: for instance, it costs more than 90% of the routing time of the Paragon machine [4], and it is announced around 90μs for the CM-5 [5, 6].

One could object that, for large data transfers, this overhead would ultimately vanish. In fact, a significant part of the effort in practical parallel programming is careful data organisation in order to pack the data such that the transfers reach the appropriate size; a lot of research is devoted to sophisticated compilation techniques, such as vectorization of data transfers, with the same goal. However, the startup penalty is so high that effective use of the network is extremely difficult. For instance, to use half of the peak bandwidth of the network, the message size must be more than 1KBytes for the CM-5; to reach full use of the bandwidth, the message size must be more than 8KBytes [6]. Moreover, parallel scientific programs are highly synchronous, because