Predicting MPI Buffer Addresses*

Felix Freitag, Montse Farreras, Toni Cortes, and Jesus Labarta

Computer Architecture Department (DAC)
European Center for Parallelism of Barcelona (CEPBA)
Politechnic University of Catalonia (UPC)
{felix,mfarrera,toni,jesus}@ac.upc.es

Abstract. Communication latencies have been identified as one of the performance limiting factors of message passing applications in clusters of workstations/multiprocessors. On the receiver side, message-copying operations contribute to these communication latencies. Recently, prediction of MPI messages has been proposed as part of the design of a zero message-copying mechanism. Until now, prediction was only evaluated for the next message. Predicting only the next message, however, may not be enough for real implementations, since messages do not arrive in the same order as they are requested. In this paper, we explore long-term prediction of MPI messages for the design of a zero message-copying mechanism. To achieve long-term prediction we evaluate two prediction schemes, the first based on graphs, and the second based on periodicity detection. Our experiments indicate that with both prediction schemes the buffer addresses and message sizes of several future MPI messages (up to +10) can be predicted successfully.

1 Introduction

MPI (Message Passing Interface) is a specification for a standard library to address the message-passing model of parallel computation [11]. In this model, applications are divided into different tasks (or processes) that communicate by sending and receiving messages among them. A number of implementations of MPI are available like MPICH from Argonne National Laboratory [12], CHIMP from Edinburgh Parallel Computing Center (EPCC) [4], and LAM from Ohio Supercomputer Center [10].

Communication latencies have been identified as one of the performance limiting factors of message passing applications in clusters of workstations/multiprocessors [1]. On the receiver side, message-copying operations contribute to these communication latencies. In a standard implementation, there is at least one copy of the message from the buffer of the MPI implementation to the user space.

Zero message-copying on the receiver side of MPI has been indicated as a technique to reduce this communication latency [1]. One of the identified requirements to achieve zero message-copying is to predict the characteristics of the

* This work was supported in part by the Spanish Ministry of Education and Science under TIC2001-0995-C02-03, CEPBA and CIRI fellowship grants.

next MPI message. The prediction of only the next message, however, may not be enough to implement a zero message-copying mechanisms in real implementations, since messages do not arrive in the order in which they are requested. If the receiver could predict several future messages, then changes in the message arrival order could be handled, which enables the design of a zero message-copying mechanism.

The goals of this paper are the followings: We first describe the design of a zero message-copying mechanism, which requires several future messages to be predicted. Then we show that with the proposed prediction mechanisms this long-term message prediction can be achieved with high prediction rates.

The remainder or this paper is structured as follows: In section 2 we describe related work done on MPI message characterization and MPI message prediction. In section 3 we explain the design of a zero message-copying mechanism and the predictors we evaluate. Section 4 shows the achieved message prediction rates and compares both predictors. In section 5 we conclude the paper.

2 Related Work

Communication locality of MPI messages was studied by Kim and Lilja in [9]. The result of their work is important since if communication locality exits, then the future values of a stream will belong to the observed data set, and will allow prediction. In their experiments it was observed that the processes communicate only with a small number of partners (message-destination locality). Also, it was observed that the MPI applications usually have only 2-3 distinct message sizes (message size locality). The results showed the locality of MPI communication patterns in the spatial domain. The characteristics of the temporal patterns in MPI messages, however, were not reported. In our work, we examine the existence of temporal patterns in MPI communications, which is an important requirement to allow long-term prediction.

The prediction of MPI messages is proposed in [1]. In their work, predictors are based on heuristics and detect cycles in an observed data stream. The prediction heuristics predict the next value of the given data stream. It was shown that the predictors obtained very high hit rates for the studied benchmarks. The benchmarks were the NAS Bt, Cg, and Sp, and the PSTSWM benchmark. In [7], the prediction of MPI messages is explored with the goal to achieve a better scalability of MPI applications.

3 Our Approach

3.1 Requirement for a Zero Message-Copying Mechanism

Previous research [1] has proposed the idea of predicting the user buffer, where the data will be read, and to place it directly in its final location. This would avoid copying first in an MPI buffer and then copy it again to the user buffer. This idea, although interesting in its concept, it is not implementable. Predicting the destination buffer to use it by the MPI library is too dangerous because an error in the prediction may modify a memory location that has useful data for the application. It is also important to notice that previous work only predicted the location of the buffer for the next message, and this is not enough. Messages do not arrive in the same order they are requested and thus, the system has to be ready to receive messages out of order and still be able to implement the zero-copy mechanism with them. Otherwise, the applicability of the improvement will be very restricted.

Le us assume now that we are able to predict the buffers for the future messages (which we will show along this work), the challenge is how we can use this knowledge to avoid extra copies. As we have said, using the exact buffer is not a possibility because of miss predictions (among others), but the MPI library can place the data aligned in the same way as in the final destination buffer. If we hit in the prediction, then we can change the mapping of logical pages to physical pages to move the data to the user address space without a copy.

This proposal could avoid copies of full pages, but not the portions of the message that do not fill a full page. In the latter case, we will have to copy this information, but hopefully, long and costly messages will avoid the long copies. All messages longer than 2 pages, will at least avoid the copy of one page. Regarding miss predictions, they do not cause any problem because the message will be copied as if no prediction had been done. A miss prediction will mean that the improvement will not be possible but no miss function will appear.

The only requirement we have is a system call that allows us to switch the binding of logical pages to their physical pages (of course only among the pages of the process). Once proven its necessity, this implementation is simple and could be easily incorporated in new versions of the OS.

3.2 Graph-Based Predictor

Our first solution for long-term prediction is a graph-based predictor as described in [5]. This predictor is similar to the prediction heuristics used in [1]. The second prediction mechanism, which we evaluate for this task, is a periodicity-based predictor [6].

Graph-based predictors describe an observed sequence through a number of states with transitions between them. Each state represents an observation. A probability or counter values is associated with the transition between the states. The graphs are trained (and build) on the observed data. The number of states increases with the increase of different symbols in the observed sequence. The observations contribute to form the transition probability from one state to another. Cyclic behavior in a data sequence, for instance, can be easily represented with such graphs, as demonstrated in [1].

In order to evaluate long-term MPI message buffer address prediction, we implement the graph-based predictor following the description in [5]. Each state represents a sequence of three observations. The value of the transition between states is computed according to the observed sequence. Prediction can be achieved by selecting the most likely successor of a current state. We predict several future values by repeating this process on the predicted states.

3.3 Periodicity-Based Predictor

The approach of the periodicity-based predictor is different to the graph-based predictor, since prediction is based on the detection of iterative patterns in the temporal order of the sequence.

Previous results from [7] show that MPI messages contain repetitive sender and message size patterns. The knowledge of the periodic patterns allows predicting future values. We use the dynamic periodicity detector (DPD) from [6] to capture the periodicity of the data stream and modify it to enable the prediction of data streams.

The algorithm used by the periodicity detector is based on the distance metric given in equation (1).

$$d(m) = sign \sum_{i=0}^{N-1} | x(i) - x(i-m) | \quad (1)$$

In equation (1) N is the size of the data window, m is the delay (0<m<M), M<=N, x[i] is the current value of the data stream, and d(m) is the value computed to detect the periodicity. It can be seen that equation (1) compares the data sequence with the data sequence shifted m samples. Equation (1) computes the distance between two vectors of size N by summing the magnitudes of the L1-metric distance of N vector elements. The sign function is used to set the values d(m) to 1 if the distance is not zero. The value d(m) becomes zero if the data window contains an identical periodic pattern with periodicity m.

4 Evaluation

4.1 Benchmarks Used

In order to evaluate the long-term predictability of MPI buffer addresses we run several experiments with the NAS benchmark suite programmed with MPI. We use the NAS Bt, Cg, Ft, Is, Lu, and Sp benchmarks [2]. The MPI applications are executed on an IBM RS/6000 in a dedicated environment. We use the class A problem size of the NAS benchmarks. The MPI implementation we used is MPICH [12]. The applications are run with different numbers of processes, where the number of processes is from 4 to 32 processes.

In order to obtain the communication behavior of the applications, we instrument the MPICH implementation. To obtain the communication data we trace the MPI calls from the application code to the top level of the MPI library. We trace point-to-point and collective calls. Collective calls are represented in the trace as point-to-point calls from the different senders. The traces we extract correspond to the receiver side. We extract the buffer address, message size, and sender processes.

In Table 1 we summarize the characteristics we observed in these traces. Column 2 indicates the number of processes the application is executed with. Column 3 gives the number of messages received per process. Column 4 indicates the number of different buffer addresses in the pattern. Column 5 shows the number of different message sizes in the pattern. Column 6 indicates the size of the observed temporal pattern. The pattern size refers to the pattern formed by point-to-point and collective calls. We have obtained the size of the periodic patterns using the DPD of [6]. We can see that the results given in columns 4 and 5 confirm the data locality in MPI messages described in [9]. Column 6 indicates the existence and size of temporal patterns in MPI messages.

Table 1. Evaluated benchmarks and communication characteristics

Benchmark	# of processes	#of messages received	# of different buffer addresses in the pattern	#different message sizes in the pattern	detected pattern size
NAS BT	4	2425	7	4	18
	9	3630	7	3	12
	16	4835	7	3	24
	25	6034	7	5	30
NAS CG	4	1679	2	2	4
	8	2942	3	2	7
	16	2942	3	2	7
	32	4204	3	2	10
NAS FT	4	998	6	3	12
	8	70	8	1	8
	16	8992	6	7	24
	32	262	32	32	32
NAS IS	4	100	9	6	9
	8	188	17	9	17
	16	364	33	17	33
	32	716	65	34	65
NAS LU	4	31490	2	2	126
	8	31492	2	4	126
	16	31492	2	2	126
	32	47229	2	4	126
NAS SP	4	4823	6	3	12
	9	7226	6	6	18
	16	9000	6	7	24
	25	12000	6	12	30

4.2 Long-Term Buffer Address Predictability

The zero message-copying mechanism designed in section 3.1 requires the prediction of several messages. Therefore, we are interested in evaluating the long-term predictability of the buffer address together with the message size. The task is to predict the next (+1) and the tenth (+10) future buffer address and message size. The tenth message in the future (+10) is chosen as upper bound. The chosen mechanism for zero message-copying may require to advance less. In our experiments, the input stream of both predictors is a linear combination of the buffer address, messages size and sender process, such as used in [1] and [7].

In Figure 1 the prediction results are shown. In the graphics, we denote the prediction of the future values with +1, and +10. The letter "D" indicates the periodicity-based predictor and the letter "G" the graph-based predictor.

The prediction accuracy for messages larger than 8k is shown. As described in section 3.1, the zero-copying mechanism is effective for messages larger than 8k. It can be observed that the prediction accuracy is generally very high (many times > 90%) with both predictors and in the +10 scenario. We can see very similar performance of both predictors in the Bt, Cg, Ft, Is, and Sp benchmarks. An exception is the Bt executed with 25 processes. In this case the performance of the graph-based predictor decreases when performing the +10 prediction task.

A special behavior can be observed in the Lu benchmark for the graph-based predictor, when predicting messages larger than 8k on +10. Here, correct prediction is not achieved. The reason for failing in this prediction is discussed in detail in the next section.

4.3 Comparison of Predictors: Accuracy and Overhead

We observed very high prediction rates including for long-term prediction (+10) both with the graph and the periodicity-based predictor. In many benchmarks, the rates of both predictors are similar. These benchmarks include the Bt, Cg, Ft, Is, and SP, which all showed regular patterns of a rather small temporal size (see section 4.1). Furthermore, the sequences have many different elements, which is beneficial for the performance of the graph-based predictor.

Differently, the Lu benchmark showed a large pattern, of size 122 and 126, respectively, within which smaller (nested) pattern are repeated. In the Lu, a large message appears after observing a long sequence of small messages with identical values. In terms of prediction rates, the prediction of +10 in the Lu with the graph-based predictor goes down to zero, as it can be seen in Figure 1. The periodicity-based predictor, however, could predict such a message, even after having observed identical messages during a long time. The reason for this capability is that it captures the periodicity of 126 in the message stream.

We found that achieving the knowledge of such long periodicities with the periodicity-based predictor is computationally more expensive than using the graph. In our current implementation, the graph-based predictor is much faster than the periodicity-based predictor. In the periodicity-based predictor, the length of the history, which enables to compute the periodicity, strongly affects the execution time. In our study, we have used the default value of the periodicity-based predictor, which is a history of 256 samples (which allows to capture periodicities up to 256).

Although graphs are usually not used to predict several future values, we saw that predicting them by walking along the built graph provided accurate results for long-term prediction. We found that predictions of this type of sequences by statistical models such the graphs are computationally efficient combined with high prediction rates. On the other hand, the periodicity-based predictor showed its strength capturing large patterns such as observed in the Lu. This achievement, however, also involved a higher computational cost.

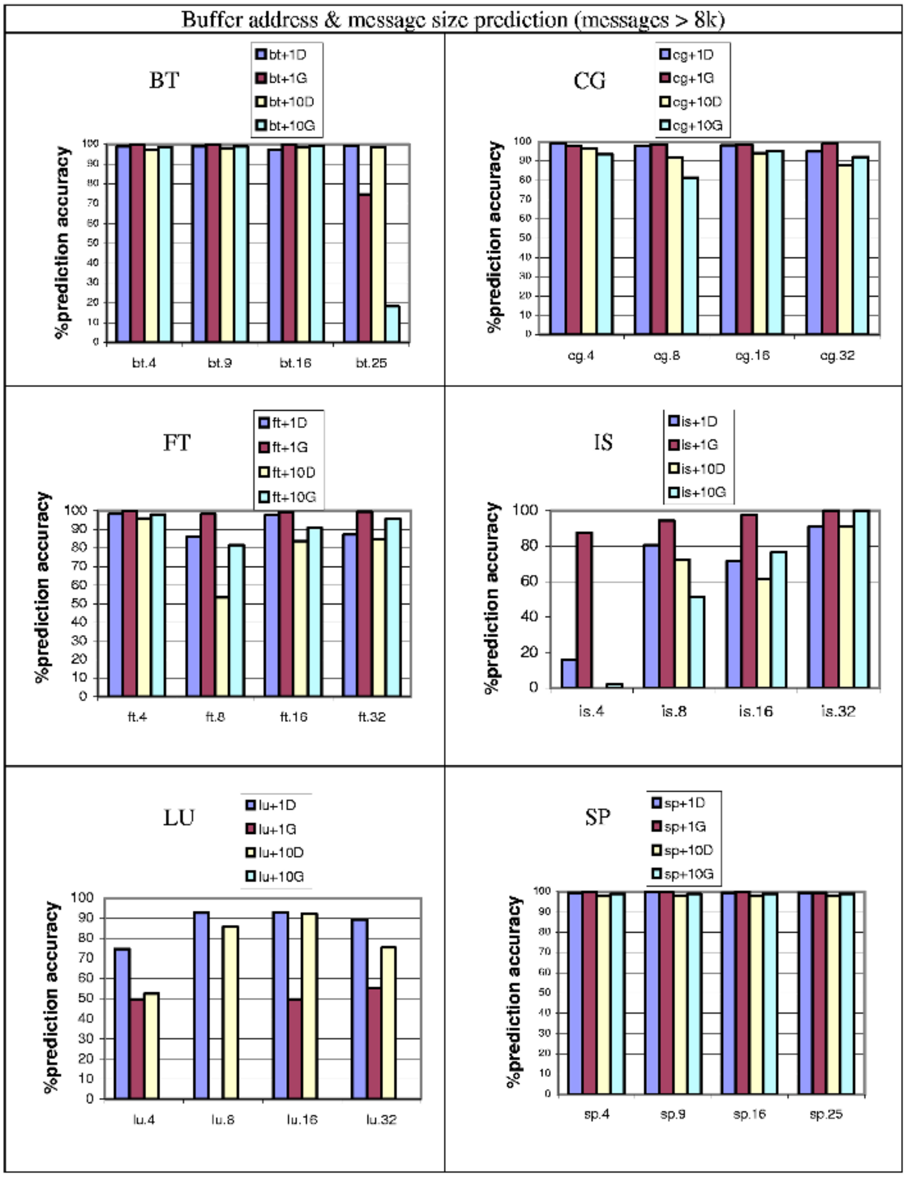

Fig. 1. Long-term buffer address and message size prediction

5 Conclusions

We indicated the need to predict several messages in order to implement a zero message-copying mechanism, which can cope with changes in the arrival order of messages. We described how this zero message-copying could be achieved. In traces of MPI communication data the existence of temporal patterns in the buffer addresses was observed. We evaluated two prediction schemes for prediction, one based on graphs, and the second based on periodicity detection. The predictors were used to predict the buffer addresses of future messages (+1, and +10). Our results indicate that the accuracy of long-term prediction is very high with both predictors. We identified an advantage of the periodicity-based predictor in capturing large patterns as in the Lu, but observed also a larger computational cost than in the graph-based predictor. If patterns are small and consist of different elements, the graph-based predictor showed to be computationally efficient combined with high prediction rates.

References

1. A. Afsahi, N. J. Dimopoulos. Efficient Communication Using Message Prediction for Cluster of Multiprocessors. *Concurrency and Computation: Practice and Experience* 2002; 14:859-883.
2. D. H. Bailey, E. Barszcz, L. Dagum, and H. D. Simon, NAS Parallel Benchmark Results, *Proceedings of the Scalable High-Performance Computing Conference*, 1994, pp. 111-120.
3. BlueGene home page: `http://www.rsearch.ibm.com/bluegene/`
4. CHIMP/MOI Project: `http://www.epcc.ed.ac.uk/epcc-projects/CHIMP`
5. K. M. Curewitz, P. Krishnan, and J. S. Vitter. ``Practical Prefetching via Data Compression," *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93), Washington, DC,* May 1993, 257-266.
6. F. Freitag, J. Corbalan, J. Labarta. A dynamic periodicity detector: Application to Speedup Computation. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS 2001),* April 2001.
7. F. Freitag, J. Caubet, M. Farrara, T. Cortes, J. Labarta. Exploring the Predictability of MPI Messages. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS 2003),* April 2003.
8. W. Gropp, E. Lusk, N. Doss and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. In *Journal of Parallel Computing*, 22(6), pp. 789-828, September 1996.
9. J. Kim, D. J. Lilja. Characterization of Communication Patterns in Message-Passing Parallel Scientific Application Programs. In *Proceedings of the Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing*, pp. 202-216, February 1998.
10. LAM/MPI home page: `http://www.lam-mpi.org/mpi`
11. MPI Forum. MPI: A message-passing interface standard. `http://www.mpi-forum.org`
12. MPICH home page. `http://www-unix.mcs.anl.gov/mpi/mpich`
13. Y. Sazeides, J. E. Smith. The Predictability of Data Values. In *International Symposium on Microarchitecture* (MICRO-30). 1997.