Real-time Communication for Distributed Vision Processing based on Imprecise Computation Model

Yoshimoto, Hiromasa
Department of Intelligent Systems, Kyushu University

Arita, Daisaku
Department of Intelligent Systems, Kyushu University

Taniguchi, Rin-ichiro
Department of Intelligent Systems, Kyushu University

出版情報：Proceedings of International Parallel and Distributed Processing Symposium, IPDPS 2002, pp.128-133, 2002-04. IEEE
バージョン：

Real-time Communication for Distributed Vision Processing based on Imprecise Computation Model

Hiromasa Yoshimoto, Daisaku Arita and Rin-ichiro Taniguchi
Department of Intelligent Systems, Kyushu University,
6-1 Kasuga-Koen, Kasuga, Fukuoka 816-8580 Japan.
{yosimoto, arita, rin}@limu.is.kyushu-u.ac.jp
http://limu.is.kyushu-u.ac.jp

Abstract

In this paper we propose an efficient real-time communication mechanism for distributed vision processing. One of the biggest problems of distributed vision processing, as is the same as in other distributed systems, is how to reduce the overhead of communication among computation nodes. In vision processing, we have to deal with a lot of time varying variables, some of which are large in size, and, therefore, the efficiency of sending and receiving of those variables is essential. To solve the problem, we propose Accuracy-driven Memory architecture, whose key idea is based on imprecise computation model and predictive coding. Here, we will present the basic framework of Accuracy-driven Memory architecture and show its efficiency based on some simulation results.

1 Introduction

Recently, especially in computer vision community, image analysis using multiple cameras has been extensively researched. When we use multiple cameras, distributed systems are indispensable because a single or centralized system can not handle a large amount of image data[1, 2, 3, 4]. When we require real-time analysis of those images, the problem becomes much more serious because of their large bandwidth. To solve the problem, we have developed a distributed real-time vision system on a PC-cluster, which consists of multiple off-the-shelf PCs connected via very high speed network[1]. The key issues of such distributed real-time vision systems are synchronization among distributed PCs, the performance of network, the end-to-end latency at user level, the programming framework[5]. The biggest problem of our current system is that to realize real-time processing we have to prepare sufficient computation resources and to set the deadline for real-time processing with enough margins, which decreases the execution efficiency of the system, i.e., throughput and latency of the system.

In this paper, we propose a framework of efficient real-time communication for distributed real-time vision system to solve this problem, or to increase the system efficiency. The key idea is based on the facts that data handled by real-time vision processing such as image sequences and time varying variables related to them have a lot of redundancies and that the reduction of the redundancies can lead to decrease of the communication overhead. For example, when estimating the position of a target object, the estimated position in the previous frame can be a good cue for the estimation in the current frame, i.e., search region of the object is restricted in the neighborhood of the previous position. If we know more detailed characteristics of the object, e.g., the motion of the object is uniform velocity, we can further narrow down the search region. Thus, we can drastically reduce the amount of data to be actually processed and transferred based on this kind of estimation.

The proposed framework is Accuracy-driven Memory architecture, where the most important point is that the accuracy of processed result required by an application is explicitly considered. When required accuracy is high, relatively dense communication is invoked, and, as a result, fast computation is required. On the other hand, required accuracy is low, relatively sparse communication is invoked, and relatively slow computation is permitted. Thus, according to the required accuracy, network bandwidth and allocable time slot for computation is controlled, and, as a result, higher execution efficiency can be achieved.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

2 Related Works

Here, to clarify our motivation, we discuss several related works which provide frameworks of real-time processing on distributed systems.

2.1 Dynamic Memory Architecture[6]

Dynamic Memory is a kind of distributed shared memory which provides a mechanism of real-time asynchronous Read and Write operations of time varying variables among multiple parallel processes. Read operation has a mechanism of interpolation and prediction of a value at any timing referring to previously written values of a time varying variable, and this mechanism provides complete asynchronous data exchange. Since, in principle, synchronization process on Dynamic Memory causes no delay, users can acquire a communication path with very wide bandwidth. Also they can describe programs without explicit synchronization, which simplifies the program description. The problem of Dynamic Memory is that it cannot deal with the accuracy of read values, because a function used to interpolation and prediction function can calculate a value at any timing, which has no responsibility how the value accurate is, especially when future values, or values not yet written, are predicted.

2.2 Dead Reckoning

Using very high-speed network, we can acquire high throughput. However, the latency problem still remains, and we have to introduce a software mechanism to reduce the latency. A typical example is "Dead Reckoning," which is widely used in applications of networked virtual reality system[7]. Dead Reckoning is that when object state is shared in multiple nodes, each node except for the home node of the object estimates the object state based on its previous states. When the actual object state is received, the error of the object state is compensated. It is a kind of predictive coding and can reduce the communication bandwidth and the latency.

2.3 Imprecise Computation[8]

In general, valid computation results are only ones produced by a task whose execution is completed. In other words, results of a suspended task are meaningless, and can not be used at all. In imprecise computation model, on the other hand, results of a suspended task can be used for further computation. Of course, the results have some errors in some sense, but they can be still used by other tasks. The feature of the imprecise computation model that the computation can be suspended gives us flexibility of task scheduling. For example, we can establish a processing scheme in which the accuracy of the computation can be changed according to the restriction of allowed computation time, which is suitable for real-time scheduling.

3 Accuracy-driven Memory

3.1 Basic Idea

When computation is started, a processor has to wait for all the data required for the computation. When distributed processing on processors which do not share a common bus is executed, the overhead of communication among processors for data synchronization can not be ignored. For example, a required data for computation of a processor is on another processor, or a sender, the processor, or a receiver, should always wait for arrival of the data to guarantee the data synchronization, and it causes the delay in starting the computation. Therefore, we can not realize real-time systems without wastefully redundant computation resources including allocable time slot.

To solve the problem, or to establish more efficient and flexible framework for real-time processing, we propose a new idea of data synchronization for distributed processing. The key idea is that in many cases image or vision bears redundant information and, therefore, strictly precise computation is not always required. In other words, imprecise computation model can be introduced to some extent. The important point is that it can be imprecise but still some accuracy is required, which should be specified by a target application.

In our architecture, when reading a remote variable which is time varying, we specify two kinds of additional parameters: required accuracy and deadline for the read operation. When the remote data at specified time is not ready on the receiver, the memory system estimates the value, based on a given estimation function, referring to previously received data. The important point on the estimation mechanism is that an accuracy evaluation function, which evaluates the accuracy of the estimation, is linked to the estimation function, and that if the evaluated accuracy is less than the given accuracy of read instruction the read operation is blocked until the estimation accuracy is larger than the parameter. If the deadline arrives and if the evaluated accuracy is still below the threshold, a value estimated at this timing is returned.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

3.2 Framework of Accuracy-driven Memory

In our Accuracy-driven Memory architecture, for each memory, we define two functions: $F_x(t)$ for estimating a required value at time t and $F_a(t)$ for estimating the accuracy of the estimated value at time t. Of course, F_x and F_a implicitly refer to the previously written data. These functions, which are specified by a user programmer, have the following characteristics[1]:

- If t is before the latest time of data arrival, in F_x a required value is interpolated from previously arrived data. Otherwise, it is predicted.
- $F_a(t)$ is ranging from 0 to 1, and the higher the value is, the more accurate, i.e., the more confident, the estimation is. The accuracy 1 means that the required time t is older than or equal to the time of the latest data, and it can be precisely interpolated from previously received data. In case that a value is predicted in $F_x(t)$, F_a is monotonically decreasing according to the difference between t and the latest time of data arrival.

Figure 1 shows a protocol of reading and writing a value at time t with the accuracy p and the maximum blocking time is w. In this basic protocol, the write protocol is the same as the other memory architecture[2]. For comparison, Figure 3.2 (a) shows a read protocol of Dynamic Memory and (b) shows one of ordinary memory system. If we set p=0, the architecture is equivalent to Dynamic Memory, and if we replace $F_x(t)$ by a FIFO queue, it is almost equivalent to buffered read mechanism provided by an ordinary operating system.

3.3 Programming Interface for Imprecise Computation Model

In the imprecise computation model, the writer task may terminate the processing according to the required accuracy in the reader task. The required accuracy depends on applications and runtime situations, but it can be obtained from the `read()` operation's parameter. The `write()` operation returns either `SATISFIED` or `NOT_SATISFIED` which means whether the written value with the specified accuracy satisfies the `read()` operation. With this `write()` can be used as shown in Figure 3.3:

[1]We assume that computation costs of these functions are small enough.

[2]We have a plan to extend the write protocol.

```
write(Memory m, Value x, Accuracy a,
Time t)
 {
  insert(m, x, a, t);
  send_signal(m);
  if ( a > m.required_accuracy )
   return SATISFIED;
  else
   return NOT_SATISFIED;
 }
```

(a) Write Operation

```
read(Memory m, Time t, Accuracy a,
Deadline w)
 {
  m.required_accuracy = a;
  while( Fa(m,t) < a ){
   wait_for_a_signal(m, w);
   if ( timeout(w) ) break;
  }
  return {Fx(m,t), Fa(m,t)};
 }
```

(b) Read Operation

Figure 1. Memory Access Protocol of Accuracy-driven Memory

3.4 Barrier Synchronization

`poll()` is the barrier synchronous mechanism that waits for two or more memories to become accurate enough. Like the "poll" system call of UNIX, it can wait for some accuracy-specified values without busy-wait. Specifically, `poll()` is called with a set of N independent memories and a set of N required accuracies. `poll()` will wait for M values out of N memories to become available for reading. On exit, the calculated N accuracies of the memories are returned. According to the returned accuracies, for example, an application program can read the most accurate memory.

3.5 Node-to-node Communication over the Accuracy-driven Memory

The communication by transmitting packets in a distributed environment such as PC cluster can be re-

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

```
read(Memory m, Time t)
{
  return Fx(m,t);
}
```

(a) Dynamic Memory

```
read(Memory m, Time t)
{
  while ( !data_exist(m,t) );
  return receive(m,t);
}
```

(b) Ordinary Read Operation

Figure 2. Read Protocol of Other Method

```
Memory m;
time   t;
value  x;

x = make_a_rough_estimation(t);
if ( write(m, x, 0.3, t) == SATISFIED )
  goto end;
x = calculate_more_accurately(x,t);
if ( write(m, x, 0.6, t) == SATISFIED )
  goto end;
x = calculate_precisely(x, t);
write(m, x, 1.0, t);
```

Figure 3. Programming Interface for Imprecise Computation Model

placed with Accuracy-driven Memory. Figure 4 shows an overview of node-to-node communication over the Accuracy-driven Memory. It should be noted that all the packet are not transmitted actually, but some of them are transmitted by which a required accuracy can be satisfied. An application with Accuracy-driven Memory works with predicted data, and, therefore, there is no need for the application to calculate and transmit data while the accuracy of predicating data is enough accurate for the application.

4 Preliminary Experiment and Evaluation

Here, we show the efficiency of Accuracy-driven Memory architecture by preliminary experiments, in which a time varying variable having sinusoidal value is transferred. In this experiment, linear interpolation

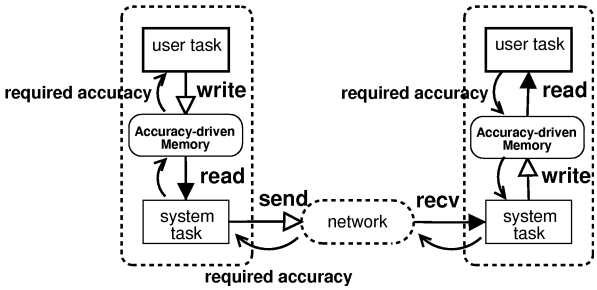

Figure 4. Node-to-node Communication

and prediction is used for $F_x(t)$. Figure 5 shows time chart of reading and writing the variable where

- a dotted sinusoidal line shows continuous real data, which is represented in

$$f(t) = 10sin(\frac{\pi}{1800}t + \frac{\pi}{4})$$

- $F_a(t)$ is defined as follows:

$$F_a(t) = \begin{cases} 1 & t < T \\ (1 - \frac{\|t-T\|}{100})p(T) & T \le t < T + 100 \\ 0 & T + 100 \le t \end{cases}$$

  where T is the latest time of data arrival.

- required accuracy p is 0.8.
- x's show values written to the memory (write commands are issued every 330 msec).
- +'s show values read from the memory (read commands are issued every 30 msec).
- downward arrows show correspondence between read command issues and their completion where the tails indicate when read commands are issued and the heads indicate when they are completed. When a read command is completed a value indicated at the tail of the arrow is returned.

If an arrow is vertical, there is no delay. Otherwise a certain delay occurs[3]. The important point shown here is that even if the write cycle is large, i.e., throughput of actual data transfer is not large, its following computation can be done eagerly based on the estimation and the computation efficiency in the following stage can be kept high. When the reliability of the estimation seems to be low, such eager computation is blocked.

[3]Since read commands issued during this delay are automatically canceled here, read transactions completed in the figure become rather sparse.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

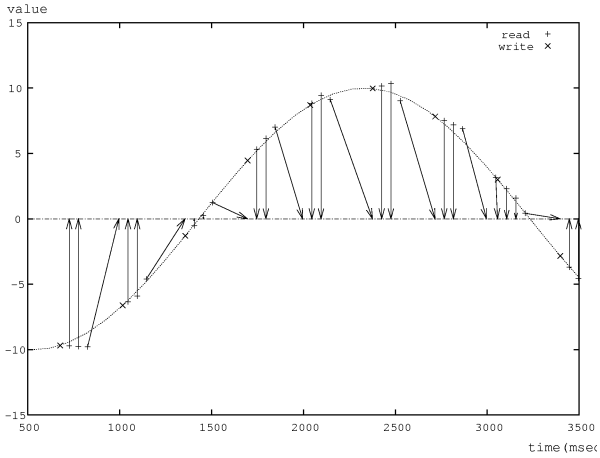

Figure 5. Timing of Read and Write

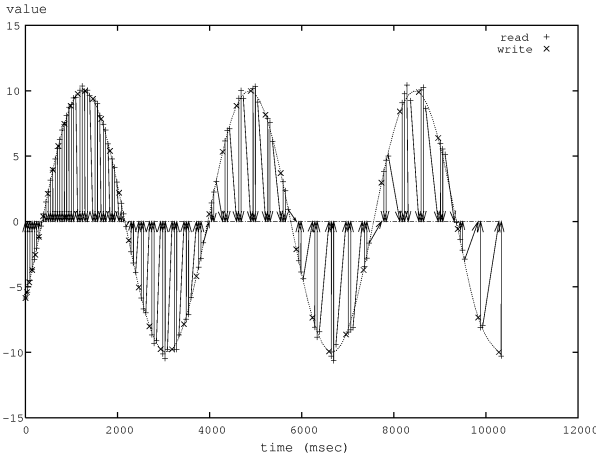

Figure 6. Influence of Change of Write Cycle

Figure 6 shows relation between write cycle and read cycle, where the write cycle gradually becomes longer. When the write cycle is short, good estimation can be done and, as a result, the throughput of computation issuing the read commands becomes high. When the write cycle becomes long, the throughput becomes low, but, compared with an ordinary remote memory mechanism, the throughput is still higher.

Figure 7 shows the change of the latency and the change of the errors between real values and read values, or estimated values when a specified accuracy changes. The experimental data is the same as in Figure 6. The result indicates that if the specified accuracy is high, the error becomes low but the latency becomes large. On the other hand, the specified accuracy is low, the error becomes large but the latency becomes low. This means that the system performance can be well controlled by the accuracy.

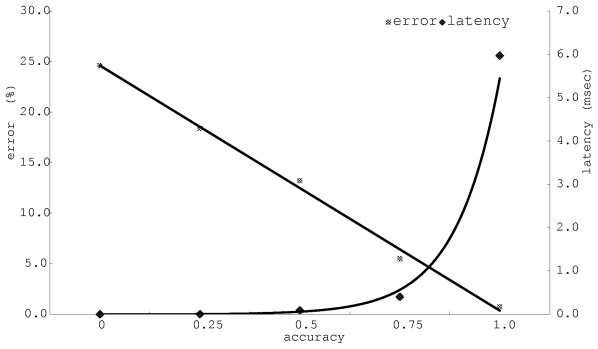

Figure 7. Influence of Accuracy

5 Conclusion and Future Works

In this paper, we propose Accuracy-driven Memory architecture, an efficient real-time communication mechanism for distributed vision processing. The architecture has similar features to Imprecise Computation Model and Dynamic Memory. The important point is that it provides a mechanism of synchronization with respect to accuracy, which can increase the system execution efficiency in terms of throughput and latency. Because it is a practice technique to use predicated values, we believe, the application which will get a benefit is in large numbers more. By using this accuracy driven memory architecture, the programmer can describe real-time computer vision applications more briefly, without taking care of the problem of deadlock, the time required for computation, and latency of communication.

One of important future works is investigation into $F_x()$ and $F_a()$. Since the effectiveness was shown in simulation results, we are evaluating its effectiveness based on actual vision applications.

Acknowledgment

This work has been partly supported by R&D activities in the info-communication area, Telecommunication Advancement Organization of Japan, No.10080.

References

[1] D. Arita, S. Yonemoto and R. Taniguchi. real-time computer vision on PC-cluster and its application to real-time motion capture. *Proc. of IEEE Workshop on Computer Architectures for Machine Perception*, pp.205–214, 2000.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 **IEEE**

[2] T. Kanade, H. Saito and S. Vedula. The 3D room: digitizing time-varying 3D events by synchronized multiple video streams. *Technical Report*, CMU-RI-TR-98-34, Carnegie Mellon University, Dec. 1998.

[3] T. Wada, X. Wu, S. Tokai, and T. Matsuyama. Homography based parallel volume intersection. Toward real-time volume reconstruction using active cameras. *Proc. of IEEE Workshop on Computer Architectures for Machine Perception*, pp.331–339, 2000.

[4] E. Borovikov, and L. Davis. A distributed system for real-time volume reconstruction. *Proc. of IEEE Workshop on Computer Architectures for Machine Perception*, pp.183–189, 2000.

[5] D. Arita, Y. Hamada, S. Yonemoto and R. Taniguchi. RPV: a programming environment for real-time parallel vision – specification and programming methodology. *Proc. of 15 IPDPS 2000 Workshops*, pp.218–225, 2000.

[6] T. Matsuyama, S, Hiura, T. Wada, K. Murase and A. Yoshioka. Dynamic Memory: Architecture for Real Time Integration of Visual Perception, Camera Action, and Network Communication, *Proc. of Computer Vision and Pattern Recognition*, pp.728-735, 2000.

[7] S. Singhal and M. Zyda. *Networked Virtual Environments*, Addison Wesley, 1999.

[8] J. Liu, W. Shih, K. Lin, R. Bettati and J. Chung. Imprecise Computation, *Proc. IEEE*, Vol.82, No.1, pp.83-94, 1994.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 **IEEE**