Evaluation of Distributed Garbage Collection System for PIE64

4W-4

Lu Xu, Yasuo Hidaka, Hanpei Koike, and Hidehiko Tanaka *

Abstract

In this paper, we will evaluate an algorithm for garbage collection of distributed heap memories. We propose that a garbage collector should be comprised by several layers of garbage collections. We refer such a garbage collector to as Garbage Collection System (GCS). Our GCS is mainly based on static analysis and a combination of Reference Counting on memory pages and Mark-Scan Scheme. Just as illustrated later, our algorithm, which deals with variable size objects, is very time-efficient, partly real-time and can be implemented with very little space overhead.

1 Introduction

There must be many kinds of objects in a system. These kinds of objects can be divided into several categories according to their consuming rates. According to the characteristics of each one of these categories, we choose a garbage collection scheme appropriate for it. All of these garbage collections make up what we call the Garbage Collection System (GCS). The main differences between the concept of GCS and the basic idea of the works before ours are:

1. We recommend that for different layers of a GCS, different schemes should be adopted if it is necessary and feasible.
2. In a GCS, objects are divided by consuming rates rather than lifetimes of them.

We need not be afraid that some kinds of objects of low consuming rates leak out from our real-time garbage collection, even we would leak them out from our real-time garbage collection deliberately, as long as we can collect them eventually by another layer of our GCS. This is our basic idea.

2 Object-Management System

If we can allocate according to the lifetimes of objects as well as according to their consuming speeds, we can reuse memory efficiently. This makes it possible to count references to pages instead of to objects and combine Reference Counting with Mark-Scan easily. Therefore, the objects in our system are classified into several categories according to their consuming speeds. Under one category for which we want to collect the garbage by Reference Counting Scheme, objects are furthermore divided into several kinds according to their lifetimes and we allocate these kinds of objects in different places. Each kind of objects may be stored in several pages.

3 GCS of PIE64

Our garbage collection system consists of three stages:

1. Real-time garbage collection based on Paging Reference Counting,
2. Local garbage collection of goal frame areas, which is accomplished independently by each IU (Inference Unit) and much more quickly than the conventional Mark-Scan method,
3. Global garbage collection by the Mark-Scan method.

In fact, there are only two categories of objects in PIE64. The first and second stages are provided for the same one category: SRO (Single Reference Objects). The third stage is provided for the other category: UNSRO.

4 Performance Evaluation

4.1 Space Overhead

At first, we would like to analyze the spatial overhead of our GCS. In our GCS, there are two kinds of spatial overhead. One is that used for the real-time garbage collection. The other part is that used for the global garbage collection. Therefore, the total amount of space overhead is $2N + o(N)$.

4.2 Real-Time Nature

Because we only offer the real-time garbage collection for SRO, therefore, the real-time nature of our GCS relies on

- the percentage of the memories consumed by SRO over that consumed totally,
- the percentage of the pages collected by real-time garbage collection over that collected our GCS.

Intuitively, the first percentage represents the nature of FLENG language. The second percentage shows the

*Tanaka Lab., Dept. of Electrical Engineering, Univ. of Tokyo

Program	Total Pages Consumed	SRO			UNSRO		
		SRO Pages Percentage	Allocation Times	Average Size	UNSRO Pages Percentage	Allocation Times	Average Size
6-Queen	237.14	96.67	56859	4.03	3.33	6433	1.23
10-Append	1.91	95.20	566	3.21	4.80	81	1.13
13-Nreverse	1.81	89.87	522	3.12	10.13	133	1.38
10-Merge	2.13	93.92	627	3.20	6.08	107	1.21

Table 1: Consuming Rates

Programs	IU Number and LMS[a]	Percent (RGC[b])	Percent (LGC[c])	Percent (GGC[d])
6-Queen	4 IUs, LMS = 23	54.65	25.29	20.07
7-Queen	4 IUs, LMS = 50	55.78	31.66	12.56
13-Append	f4 IUs, LMS = 13	100.0	0.00	0.00
10-Merge	4 IUs, LMS = 13	100.0	0.00	0.00

[a]means Local Memory Size and the unit is page.
[b]Real-Time Garbage Collector
[c]Local Garbage Collector
[d]Global Garbage Collector

Table 2: Performance of the GCS

IU Number and LMS	The Clocks Taken by One GGC	The Ratio to 1 IU
1 IU, LMS = 20	122743.13	1.00
4 IUs, LMS = 20	118211.38	0.96
16 IUs, LMS = 20	119997.48	0.98
64 IUs, LMS = 20	124390.21	1.01

Table 3: Performance of the Cooperations between Processors

nature of our GCS.

From Table 1, we can know immediately that the storage consumed by SRO is always above ninety percent. This nature is the base of our collector.

From Table 2, we can know that the garbage collected by the real-time garbage collector is over fifty percent. If we include the local garbage collector, we can see the garbage collected "in time" is about eighty percent. From the results, we can imagine that the real-time nature of our collector is quite good.

4.3 The Performance of Co-operations between Processors

We designed some cooperations between processers and expect this will alleviate the impact of remote references to some extent. The results are list in Table 3.

When the number of IUs increases to sixty-four, the time taken by the global garbage collection only increases about one percent compared to the case of uniprocessor. This implies the co-operations designed to alleviate the effect of remote references on the global garbage collection work very well.

From the evaluations above, we can claim our GCS is able to satisfy the aims proposed in the first section. We think it is the cheapest and is sufficient for PIE64.

5 Comparisons with other Works

5.1 Weighted Reference Counting

This algorithm is proposed on the base of Reference Counting. Its advantages are real-time, simplicity and high efficiency of memory reuse.

But its real-time nature is gained at the cost of time and space. Our purpose is to get high both efficiency and real-time property, so we must make a compromise. For the environment of PIE64, the local memories are mainly consumed by SRO objects, especially by goalframes, so if we reclaim the areas of SRO in time, we can obtain real-time nature to a great extent.

5.2 Copying Algorithm

When this kind of method is implemented for the environment of PIE64, indirection tables are used for remote references in processing. Another problem is that the amount of floating garbage will be increased rapidly. The main problem, we think, is its high spatial overhead.

5.3 Mark-Compaction Algorithm

This method is of very good efficiency both in time and in space. But it loses real-time property completely. In fact, our global garbage collection is based on it.

References

[1] Xu, L., Shimada, K., Shimizu, T., Koike, H. and Tanaka, H. The Distributed Garbage Collection for Parallel Inference Machine PIE64. *Proc. 38th Annual Convention IPS, Japan*, 1988.