4P-1

Performance Analysis of On-the-fly Garbage Collection

Keiji Hirata, Hidehiko Tanaka
Dept. of Elect rical Eng., Univ. of Tokyo

1 Introduction

This paper proposes the new practical criteria when we estimate the influence of garbage collection, and presents the analysis of the on-the-fly garabage collector with the criteria as an exercise.

2 Model

This section presents a model for analyzing the performance of garbage collection on a parallel machine and identify several system parameters. Using this model, we estimate the efficiency of each garbage collection method.

2.1 Model Organization

We first show a machine model (Fig.1), which may have more than one mutator (Dijkstra called a list processor a *mutator* in [DLM*78]), a collector (a garbage collector) and a common memory, containing a data memory, a mark-bit array, a free-list head register and so on. The three kinds of modules are connected

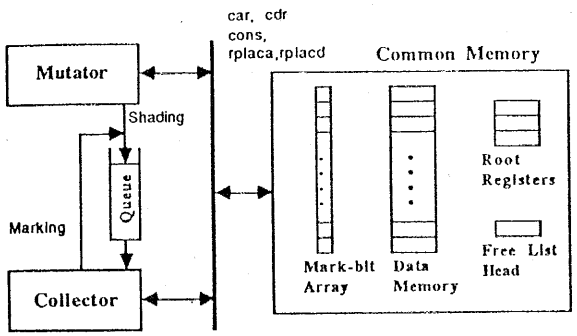

Figure 1: Machine Model for Garbage Collection

by a proper network ; a common memory is not a local memory of the collector. In this machine model, the mutator sends the command packets, e.g. car, cons, rplaca, to the common memory, or directly manipulate structure data in the memory. Moreover, the mutators transmit information for garbage collection, e.g. shading a node, changing a reference. The on-the-fly garbage collector has a packet queue, which receives the marking packets from the mutators or the collector itself.

2.2 System Constants

We calculate several system constants on this model. We take the notations of the system constants and the meanings used in [HC84] in the following discussion. Five system constants are introduced as follows :

r ··· time to RELEASE a garbage node or RETRIEVE a new node,

s ··· time to SCAN the next list node,

m ··· time to MARK an accessible node,

M ··· number of list nodes in the data memory, and

L ··· number of accessible nodes.

In on-the-fly garbage collection, the entire execution of the collector consists of the following two phases :

- marking accessible nodes, and this phase takes $m(L+B)$ (time units).
- appending garbage nodes to the free list while scanning the memory, and this phase takes sM (time units).

From [HC84], we show only the result below, because of the limitation of the space. The entire length of a *stable* cycle is expressed as follows :

$$T_{cycle} = \frac{r(mL + sM)}{r - m}. \qquad (1)$$

3 Performance Analysis

To investigate the on-the-fly garbage collection method, we take the following two criteria : (1) how much the collector interferes in memory access made by the mutators, and (2) load of the collector itself. First, we discuss how many times a collector access a common memory for garbage collection. Second, we estimate the period for which the collector actually works during an entire garbage collection cycle. We make a hypothesis that the data memory stores only list nodes, but the results we obtain are easily applicable when storing vector nodes as well.

3.1 The Number of Times Visiting a Node

We use as the measure of the interference the total number of times a common memory is accessed by the collector. Thus, we consider how many times the collector visits a node.

In the marking phase, the collector visits an accessible node *twice* (Fig.2). At that time, the collector turn on the corresponding mark of the mark-bit array. In the appending phase, for appending a garbage node to the free list, the collector rewrites either cell of the node. Accordingly, the collector visits the node *once* (Fig.3).

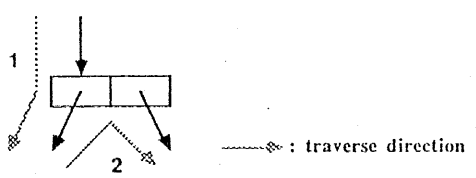

Figure 2: Visiting a Node in Marking Phase

Performance Analysis of On-the-fly Garbage Collection
Keiji HIRATA, Hidehiko TANAKA
Dept. of Elect rical Eng., University of Tokyo
7-3-1 Bukyo-ku, Hongo, Tokyo 113, JAPAN

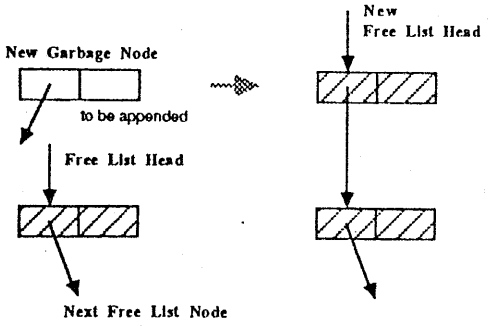

Figure 3: Rewriting a Node in Appending Phase

3.2 Estimation on Interference by Collector

We think of the following two methods :

1. Continually On-the-fly (CO Method)
2. Sometimes On-the-fly (SO Method).

We calculate the average access times per time unit for each method above.

3.2.1 CO Method

In CO method, a marking phase follows the previous appending phase immediately, and vice versa. During the marking phase, the collector traverses accessible nodes (L) and floating garbage nodes (B). In the subsequent appending phase, the collector reclaims T_{cycle}/r nodes. Thus,

$$Total\ Access\ Times/Cycle = 2(L+B) + T_{cycle}/r.$$

Therefore, the collector accesses the data memory

$$\frac{3}{r} + \frac{2L(r-m)}{r(mL+sM)}$$

times per time unit.

3.2.2 SO Method

In this method, the collector works for a part of a cycle and suspends for the rest of the cycle. That is, the collector operates during T_{cycle} or more within $q \cdot T_{cycle}$ (Fig.4). Thus, we obtain

$$Total\ Access\ Times/q \cdot Cycles = 2(L+B) + q \cdot T_{cycle}/r.$$

The access times during a marking phase is proportional to the

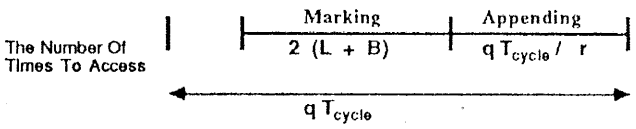

Figure 4: Sometimes On-the-fly Garbage Collection

number of currently accessible nodes, $L+B$, while the access times during an appending phase is proportional to the number of garbage nodes produced during an entire cycle, $q \cdot T_{cycle}/r$, where $q > 1$. Thus, the collector accesses the data memory

$$\frac{1+2/q}{r} + \frac{2L/q(r-m)}{r(mL+sM)}$$

times per time unit.

However, q has an upper limit, because, if q is too large, the free list will have been exhausted. Since the mutator eat up $q \cdot T_{cycle}/r$ nodes during $q \cdot T_{cycle}$, the condition that the mutator does not starve is

$$\frac{q \cdot T_{cycle}}{r} < M - L - B.$$

Thus, we obtain the upper bound of q :

$$q < \frac{r(1-L/M) - m - s}{s + mL/M}. \tag{2}$$

As an example, we calculate the upper bound of q, under the same condition as [HC84] ($r = 20\mu S(microseconds), m = 5\mu S, s = 1\mu S$) :

$$q < 1.14\cdots.$$

This result means that, under such a condition, SO method can work only if $q < 1.14$. Accordingly, since we cannot fix the value of q not so high, the reference-counting garbage collection may be rather preferable in terms of the interference caused by the collector. Further discussion can be found in chapter 6 of [Hir87].

3.3 Estimation on Collector's Load

To estimate how much jobs the collector itself have to perform, we compute the statistical values associated with the collector's load. We use as the load indicator the ratio of term the collector actually operates during an entire cycle.

In particular, we are interested in the ratio of the term for which the collector is actually operational under SO method : The operational ratio of the collector for CO method is, of course, exactly 100 %. In SO method, the collector actually operates during $m(L+B) + qsM$ and, at that time, the length of the entire garbage collection cycle is $q\{m(L+B)+sM\}$. Therefore, the operational ratio is

$$\frac{m(L+B)+qsM}{q\{m(L+B)+sM\}} = \frac{1+K/q}{1+K}, \tag{3}$$

$$where\ K = \frac{m(L+B)}{sM} = \frac{m}{s}\left(\frac{L}{M} + \frac{s+mL/M}{r-m}\right).$$

Let us estimate the value of K under the same condition as above and $L/M = 0.5$. Thus we obtain

$$K = 3.67.$$

Since we, however, notice that q has the upper bound ($q \leq 1.14$), the least value of equation 3 is obtained by $q = 1.14$:

$$equation\ 3 \geq 90\%.$$

Therefore, we can make the operational ratio decrease to 90 %. Moreover, notice that, in fact, this value is derived from the comparative relations among r, m and s, not from the absolute values of the system constants.

4 Conclusion

We have proposed the new practical criteria of performance analysis of garbage collection. We can estimate the performance of the on-the-fly garbage collector, accroding to the criteria. Furthermore, we should analyze the performance in the other machine models, compare on-the-fly with reference counting [Hir87], and consider the difference between the actual effects and the theoretical ones, etc.

References

[DLM*78] E.W. Dijkstra, L. Lamport, A.J. Martin, C.S. Scholten, and E.F.M. Steffens. On-the-fly garbage collection : an exercise in cooperation. *CACM*, 21(11):966-975, Nov. 1978.

[HC84] T. Hickey and J. Cohen. Performance analysis of on-the-fly garbage collection. *CACM*, 27(11), Nov. 1984.

[Hir87] K. Hirata. *A Structure Memory for Parallel Inference Machines.* PhD thesis, Dept. of Information Eng., Univ. of Tokyo, 1987.