Analysis of Parallel Garbage Collection with Multiple List Processors and Garbage Collectors

SHINSUKE TERAMURA*

This paper presents the performance analysis of a list processing system with multiple Lisp Processors and Garbage Collectors. The analysis is intended to be of great use when we design a high performance list processing system with multiple processors, in which several list processes can be executed simultaneously without interruption for garbage collection.

Hickey et al. have described a detailed analysis of a system which has one List Processor and one Garbage Collector. Our analysis is based on Hickey's analysis. We introduce two parameters and evaluate performance of the parallel garbage collector by comparing it with a sequential garbage collector. We also extend the result to the system with l Lisp Processors and g Garbage Collectors. It is shown that the rate of reduced execution time due to the parallel garbage collection to the total execution time in the sequential garbage collector gets near to $g/(g+2l)$. Moreover, the optimal number of the Garbage Collectors according to the given application can be estimated.

We also show the number of cells created by List Processor without waiting is not constant but those of each cycle form a kind of sequence by taking the time to append the collected cell into account. As a result, we can determine the condition where Lisp Processors never run out of cells.

1. Introduction

One of the most serious problems of the list processing system such as Lisp is the long interruption caused by Garbage Collection (abbreviated to G.C.), especially in the case where real time response is required. To solve this problem, real-time G.C. was designed. An approach to the real-time G.C. is parallel G.C.

Dijkstra [1], Kung [3] have described parallel G.C. algorithms. Both algorithms use one list processor (abbreviated to LP) and one garbage collector (abbreviated to GC).

Lamport [4] extended Dijkstra's algorithm to incorporate multiple LPs and multiple GCs. Newman [7] described an algorithm using stacking, chaining [6] and Lamport method together.

For a parallel G.C. algorithm to be efficient, not only its correctness should be proved, but the condition that the LPs never run out of cells should be satisfied. In this sense, it is very important to analyze the performance of the parallel garbage collection. Detailed analysis of Dijkstra's algorithm and Kung's algorithm have been done by Hickey [2].

The system with multiple LPs and GCs can be analyzed similarly except that the access conflict should be considered. In addition, time to append the collected cells to the free list, which is disregarded in Hickey's analysis, produces a nontrivial effect especially when multiple GCs append the collected cells exclusively, that is, appending procedure contains the critical region, or the collecting process is not equally distributed among multiple GCs.

In these days, there are many computers adopting multi-processor architecture, and the research on the parallel list processing has been in progress. In addition, these computers may be used as multi-user list processing systems where several independent processes run. From viewpoint of cell consumption, these 2 usages can be regarded as identical. Multiple LPs running simultaneously produce more garbage cells than can be reclaimed by one GC effectively. It is clear that the use of multiple processors for G.C. is one solution to improve collecting performance. We should choose the number of GCs carefully, since too many GCs disturb the list processing by the access conflict rather than performance advantage.

In section 2, Hickey's analysis is surveyed briefly, as our analysis is based on it. In section 3, two new parameters are introduced to analyze the system with one LP and one GC. We extend the result for multiple LPs and GCs with and without access conflict in section 4. We also discuss the method to determine the optimal number of GCs. We report the result of experiment carried out on real machine in section 5. Finally, conclusion is discussed in section 6.

*R & D Laboratory, RICOH Ltd., 16-1 Shin-ei cho, Kohoku, Yokohama 223, Japan.

2. Hickey's Analysis

2.1 The Algorithm

Hickey analyzed Dijkstra's algorithm (called three color collector) and Kung's algorithm (called four color collector). Here we outline Kung's algorithm because our analysis is essentially for the four color collector.

Kung's algorithm consists of the root insertion phase, the marking phase and the collecting phase. In the root insertion phase, all the roots are inserted into the deque by which cells are marked. In the marking phase, the GC takes a cell and marks (blacken) all the cells accessible from the cell using recursive trace. If the LP modifies the list structure during this phase, it inserts the corresponding cell since the cell may have to be marked. To avoid access conflict between the LP and the GC, they use different end of the deque. Free cells are not marked since they have special color (off-white).

In the collecting phase, the GC scans the whole cell space sequentially to find unmarked (white) cells and appends them to the free list. GC also unmarks (whiten) the marked cells unless they are free cells. There are two types of garbage cells existing at the end of marking phase: one is white garbage and another is non-white garbage. The former, called white (unmarked) garbage, will be collected during the collecting phase of the current G.C. cycle. The latter, called black (marked) garbage, will not be collected until the collecting phase of the next cycle.

Kung's algorithm has better marking performance than other parallel G.C. algorithm. Unfortunately, it is difficult to implement this algorithm because the deque cannot be effectively dealt with, and the roots are treated as fixed cells in this algorithm while they are not in the real system. We have proposed a parallel G.C. algorithm including less overhead than Kung's one, and implemented a real system using the algorithm to evaluate its performance [5].

2.2 The Model

Table 1 shows the parameters used in Hickey's analysis. The assumptions required to analyze are shown below:

- While the free list is not empty, the LP creates cells at a constant rate of one cell every r time units and produces garbage cells at the same constant rate. Therefore, the number of active cells and the proportion of active cells are always constant.
- The GC marks and scans cells at a constant rate of one cell every m time units to mark and every s time units to scan.
- Every newly created garbage is marked. By this assumption, the worst-case analysis can be given.

The parameter F_i, B_i and W_i are used to describe the state of the memory at the end of the marking phase. These parameters determine the state of memory of the next cycle. Therefore, a G.C. cycle is defined to begin with the collecting phase and end with the marking phase.

Table 1 The parameters in Hickey's analysis.

Time Variables	
m	time to MARK a cell
s	time to SCAN the next list cell
r	time to RELEASE a garbage cell or RETRIEVE a new cell
tmark_i	length of the ith marking stage
tscan_i	length of the ith scanning stage
tcycle_i	length of the ith cycle
Space Variables	
N	number of list cells in the memory
L	number of accessible cells
π	proportion of list cells that are accessible (L/N)
F_i	number of cells in the free list
W_i	number of unmarked garbage cells
B_i	number of marked garbage cells
B^{crit}	largest possible value of B before a non-critical cycle
G^{crit}	amount of marked garbage produced during a stable cycle
Parameters	
productivity ρ	average proportion of execution time where LP is not waiting
throughput τ	average rate at which cells are taken from the free list
wait time	length of garbage collection interruption
productivity speedup σ	ratio of productivity of parallel collector to productivity of sequential collector
relative speed λ	speed of LP relative to marking speed

If the LP never has to wait, the amount of newly created cells during a G.C. cycle is constant and called G^{crit}. Let Av_i be the amount of available cells during the ith cycle. There's a relation $Av_i = F_{i-1} + W_{i-1} = N - L - B_{i-1}$. Hickey also defined B^{crit} as B_{i-1} where Av_i equals to G^{crit}. B^{crit} and G^{crit} is

$$G^{\text{crit}} = \frac{sN + mL}{r - m}$$

$$B^{\text{crit}} = \frac{r(N-L) - (m+s)N}{r - m}$$

Hickey describes three types of performance of parallel G.C. according to the value of the parameters:

1. stable: the LP never needs to wait

$$r > m \quad \text{and} \quad B^{\text{crht}} \geq (N-L)/2$$

2. alternating: the LP must wait every other G.C. cycle

$$r > m \quad \text{and} \quad B^{\text{crit}} < (N-L)/2$$

and B_0 is not in I^{crit}

3. critical: the LP is forced to wait every cycle

$$r \leq m \quad \text{or} \quad B^{\text{crit}} < (N-L)/2 \quad \text{and} \quad B_0 \text{ is in } I^{\text{crit}}$$

where I^{crit} is the interval $B^{crit} < B < N-L-B^{crit}$

Figure 1 shows these three types. In the figure, horizontal axis and vertical axis correspond to G.C. cycle and the number of cells respectively.

In his analysis, next three parameters are used to examine the efficiency of the system:

1. productivity ρ: the average proportion of execution time where the LP is active
2. throughput τ: the average rate at which cells are taken from the free list by the LP
3. wait time: the length of the G.C. interruption

The ratio of the productivity of the parallel collector ρ to that of a sequential collector ρ_{seq}[1] is called productivity speedup, and the ratio of r to m is called relative LP speed. They are denoted as σ and λ respectively. Hickey described how the value of σ is affected according to the value of λ and π, and showed that productivity of the parallel collector is at most 150 percent of the sequential one. σ can be represented as a function of λ and τ:

$$\sigma = \min\left(\frac{(r+s)-(r-m)\pi}{2s+(1+\pi)m}, \frac{(r+s)-(r-m)\pi}{r(1-\pi)}\right)$$

$$= \min\left(\frac{\pi\lambda+1-\pi+\frac{s}{r}}{(1+\pi)\lambda+2\frac{s}{r}}, \frac{\pi\lambda+1-\pi+\frac{s}{r}}{1-\pi}\right)$$

Both the relation between σ and λ when π is fixed and the relation between σ and π when λ is fixed are fractional expressions. So they are depicted as hyperbolas. Hickey showed these figures by giving the values to each parameter.

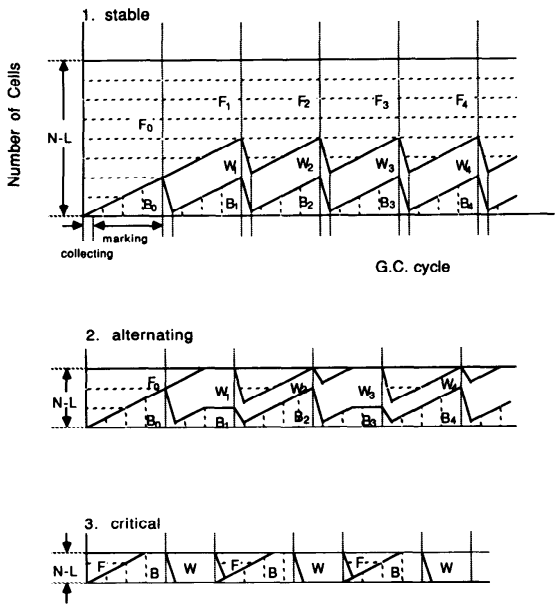


Fig. 1 The three types of the parallel G.C. performance.

[1]It is called ρ_{serial} in Hickey's analysis.

3. GC Ratio and Improvement Ratio

It is not so easy to depict the relations described above because they are fractional expressions and the value of several parameters are needed even if we'd like to know the outline of the figure.

We introduce two parameters to compare the parallel G.C. system (from now on, we call parallel system) with the sequential G.C. system (sequential system):

- GC ratio G: the ratio of the time spent for G.C. to the total execution time of the sequential system

$$G = \frac{T_{seq.gc}}{T_{seq.total}}$$

- Improvement ratio I: the ratio showing the improvement of the parallel system over the sequential system

$$I = \frac{T_{seq.total} - T_{para}}{T_{seq.total}}$$

where $T_{seq.total}$ and $T_{seq.gc}$ are the total execution time and G.C. time of the sequential system respectively, and T_{para} is the total execution time of the parallel system.

The relation between I and σ is

$$I = 1 - \frac{1}{\sigma}$$

because σ is defined as $T_{para}/T_{seq.total}$. The throughput of the sequential system, denoted as τ_{seq}, is

$$\tau_{seq} = \frac{1-\pi}{(r+s)-(r-m)\pi}$$

Let C be the number of cells required to achieve some given process, then $T_{seq.total}$ is

$$T_{seq.total} = \frac{C}{\tau_{seq}} = \frac{r+s-(r-m)\pi}{1-\pi} C$$

In the parallel system, τ_{para} is equal to $1/r$ when the system is stable, i.e., LP never waits for cells to be reclaimed. If the system is not stable, τ_{para} is calculated as follow: the amount of created cells during two consecutive G.C. cycles is $N-L$, so the time needed for creating these cells is $r(N-L)$, and the total length of two G.C. cycles is $2sN+m(N+L)$. Hence, the productivity in non-stable state is

$$\rho = \frac{r(1-\pi)}{2s+(1+\pi)m}$$

So the throughput and total execution time are

$$\tau_{\text{para}}=\min\left(\frac{1}{r},\frac{1-\pi}{2s+(1+\pi)m}\right)$$

$$T_{\text{para}}=\max\left(rC,\frac{2s+(1+\pi)m}{1-\pi}C\right)$$

The improvement ratio is

$$I=1-\frac{T_{\text{para}}}{T_{\text{seq. total}}}=\min\left(\frac{s+m\pi}{r+s-(r-m)\pi},\frac{r-s-m-r\pi}{r+s-(r-m)\pi}\right)$$

$$=\min\left(G,\frac{r-s-m-r\pi}{r+s-(r-m)\pi}\right) \quad (1)$$

r and π can be represented as the function of G

$$r=\frac{(s+m\pi)(1-G)}{G(1-\pi)}$$

$$\pi=\frac{(r+s)G-s}{(r-m)G+m}$$

The formula (1), together with the equations above takes the form below:

$$I=\min\left(G,\left(1-\frac{m}{r}\right)-\left(2-\frac{m}{r}\right)G\right) \quad (2)$$

$$=\min\left(G,1-\left(2+\frac{(1-\pi)m}{s+m\pi}\right)G\right) \quad (3)$$

The graphic representation of the right expression of minimum operator in (2) is a line through the point (1, -1) with a coefficient of $(-2+m/r)$. (2) is the relation between I and G where the value of G is varied by means of changing the value of π. If $\lambda=m/r=0$, then the line also goes through the point (0, 1). G monotonously increases being regarded as a function of π. When $0\leq\pi\leq 1$, $s/(r+s)\leq G\leq 1$. Figure 2 shows the graph of this relation. When λ gets larger, I-intercept and the absolute value of the coefficient get smaller. Therefore, the maximum improvement ratio $I_{\max}$ and the corresponding value of G also get smaller.

When $G=s/(r+s)$,

$$I=\min\left(\frac{s}{r+s},\frac{r-s-m}{r+s}\right)$$

If $s/(r+s)<(r-s-m)/(r+s)$, i.e., $r>2s+m$, then the intersecting point of the two lines is where the improvement ratio has the maximum value. If $r\leq 2s+m$, then

$$I=\left(1-\frac{m}{r}\right)-\left(2-\frac{m}{r}\right)G \quad \text{for} \quad \frac{s}{r+s}\leq G\leq 1$$

$$I_{\max}=\frac{r-s-m}{r+s}$$

When $r-s-m<0$ in addition to the condition above, $I<0$ independent of the value of π. So $r\geq s+m$ is the necessary condition where the parallel system is not worse than the sequential system. Although $\lambda\to 0$ when $m\to 0$ or $r\to\infty$, the discussion above is meaningless in the latter case because $G=0$ regardless of the values of other parameters.

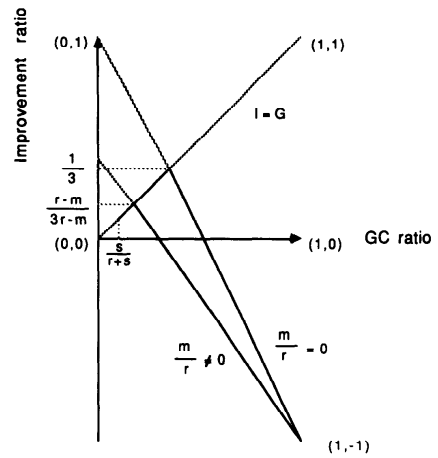


Fig. 2 GC ratio and Improvement ratio.

The right expression in (3) is a line through the point (0, 1). In this case, G is considered as a function of r decreasing monotonously. Note that $0<r\leq\infty$, $1>G\geq 0$. Let α be the coefficient of this expression, when π varies from 0 to 1, the range of α is

$$-2-\frac{m}{s}\leq\alpha\leq -2$$

This implies that G can be chosen so that $I\geq 0$ whatever value π has. Let G^0 be the maximum value of G that satisfies the condition above:

$$1-\left(2+\frac{(1-\pi)m}{s+m\pi}\right)G^0=0$$

$$G^0=\frac{s+m\pi}{2s+m\pi+m}$$

When $G\leq G^0$, $I\geq 0$, so

$$G=\frac{s+m\pi}{r+s-(r-m)\pi}\leq\frac{s+m\pi}{2s+m\pi+m} \quad r(1-\pi)\geq s+m$$

After all, $r(1-\pi)\geq s+m$ is the necessary and sufficient condition where the parallel system shows better or equal performance to the sequential system. Under this condition, $I_{\max}$ is given when both expressions in (1) are equal, so next equation can be calculated:

$$2s+(1+\pi)m=r(1-\pi) \quad (4)$$

By combining equations (1) and (4), we can express $I_{\max}$ by means of r and m:

$$I_{\max}=\frac{r-m}{3r-m}=\frac{1-\frac{m}{r}}{3-\frac{m}{r}} \quad (5)$$

Or by means of s, m and π:

$$I_{\max}=\frac{s+m\pi}{3s+2m\pi+m}=\frac{1}{3+\dfrac{(1-\pi)m}{s+m\pi}} \tag{6}$$

When $m\to 0$ in (5) and (6), $I_{\max}\to 1/3$. Note that $I_{\max}\neq 1/3$ when $r\to\infty$ or $\pi=1$ in (6) because $G=0$ or $G=1$ respectively. We can see that $I_{\max}=1/3$ from the figure also. This corresponds to the result of Hickey's analysis that $\sigma_{\max}=150\%$.

4. Multiple LPs and GCs

4.1 Analysis of the Case Where Access Conflict is Ignored

We discussed the system with an LP and a GC in the previous section. In this section, the system with the multiple LPs and GCs will be considered.

A deque cannot avoid the access conflict in the marking phase when there are multiple GCs. Therefore, we use the stack the access of which is the indivisible operation. Each GC pops a cell from the stack and marks the linked list from the cell. In the collecting phase, the whole cell space is divided into subspaces sequentially to be assigned to each GC. They scan each subspace and reclaim the garbage cells of the space. Note that every GC should synchronously change their phase. Consider that a GC enter the collecting phase while others are still in the marking phase. Those cells scanned by the GC are not necessarily marked yet, and possibly reclaimed even if they are alive. All the GC should find the marking stack empty to exit the marking phase. Similarly all of them should complete scanning to exit the collecting phase.

Firstly, we ignore the delay caused by the access conflict. Let l and g be the number of the LP and the GC respectively. We observe the improvement ratio over the sequential system. Special care should be taken that the more LP is, not only the smaller r is but the larger L is. π can be greater than 1 and all the LPs are forced to cease if N is the fixed value.Therefore we should keep π constant by changing N according to the number of the LPs when the parallel system with multiple LPs is compared with sequential system.

Each LP creates a cell every r time units and produces a garbage cell at the same rate. Each GC marks and scans a cell every m and every s time units respectively. The number of cells contained in each subspace is N and the number of active cells possessed by each LP is L. The parameters of the total values of the system are denoted by adding apostrophe (') to the parameters described in Table 1:

$$r'=r/l$$
$$m'=m/g$$
$$s'=s/g$$
$$L'=lL$$
$$N'=lN$$

The formulas shown in the previous section can be expressed as functions of l and g.

The improvement ratio can be calculated the same as in the previous section:

$$I(l,g)=\min\left(1-\frac{r(1-\pi)}{r+s-(r-m)\pi},\ 1-\frac{l\{2s+(1+\pi)m\}}{g\{r+s-(r-m)\pi\}}\right) \tag{7}$$

$$=\min\left(G,\left(1-\frac{l}{g}\cdot\frac{m}{r}\right)-\frac{l}{g}\left(2-\frac{m}{r}\right)G\right) \tag{8}$$

$$=\min\left(G,1-\frac{l}{g}\left(2+\frac{m(1-\pi)}{s+m\pi}\right)G\right) \tag{9}$$

(8) and (9) are the lines through the point $(1, 1-2l/g)$ and $(0, 1)$ respectively. $I_{\max}$ is given when both expressions in (7) are equal.

$$I_{\max}=\frac{gr-lm}{gr-lm+2lr}=\frac{g-l\dfrac{m}{r}}{g+2l-l\dfrac{m}{r}} \tag{10}$$

Or

$$I_{\max}=\frac{g(s+m\pi)}{(g+2l)(s+m\pi)+(1-\pi)lm}=\frac{g}{g+2l+\dfrac{(1-\pi)lm}{s+m\pi}} \tag{11}$$

When $m\to 0$, $I_{\max}\to g/(g+2l)$ in (10) and (11). The dependence of the GC ratio and the improvement ratio is illustrated in Figure 3, where $(l, g)=(1, 1)$, $(1, 2)$ and $(1, 3)$. $I_{\max}$ is $1/3$, $1/2$ and $3/5$ in each case.

4.2 Analysis of the Case with Access Conflict

As we described, when $m\to 0$, $I_{\max}\to g/(g+2l)$. We can see that any number of the LPs never have to wait as far as the number of the GCs is large enough from the fact that $I_{\max}\to 1$ when $g\to\infty$. But $I_{\max}$ is never equal to 1 because m cannot be reduced to 0 in the real system. Moreover, the more the number of processors, the worse the improvement ratio because of the delay caused by the access conflict.

We define access rate to be the proportion of the time used for accessing the shared resource among the total execution time. Let x and y be the access rate of each LP and GC. Consider that l LPs and g GCs access the shared resource simultaneously. We also define P as the LPs' execution time if no access conflict occurs and Q as the delayed execution time by access conflict. The ratio Q/P is the function of x, y, l and g, named ξ:

$$\xi(x,y,l,g)=\frac{(lx+gy)\{1+e(x,l)\cdot e(y,g)\}}{e(x,l)+e(y,g)}$$

where $e(X, N)$ is the total access rate of N processors

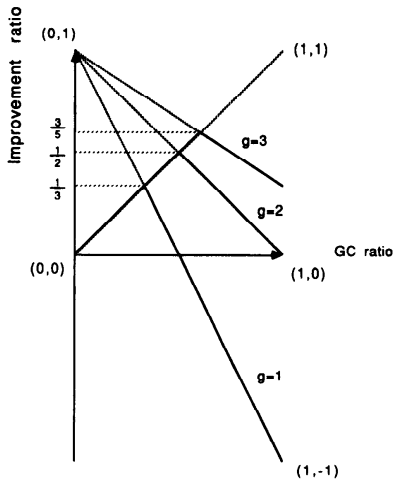


Fig. 3 GC Ratio and Improvement Ratio of Multiple GCs.

whose individual access rate is X, given below:

$$e(X, N)=\frac{(1+X)^N-(1-X)^N}{(1+X)^N+(1+X)^N}$$

m, s and r are multiplied by $\xi=\xi(x, y, l, g)$ times, so the improvement ratio is

$$I=\min\left(1-\frac{r\xi(1-\pi)}{r+s-(r-m)\pi},\ 1-\frac{l\xi}{g}\cdot\frac{2s+(1+\pi)m}{r+s-(r-m)\pi}\right) \tag{12}$$

$$=\min\left(\xi G+1-\xi,\ \left(1-\frac{l\xi}{g}\cdot\frac{m}{r}\right)-\frac{l\xi}{g}\left(2-\frac{m}{r}\right)G\right) \tag{13}$$

$$=\min\left(\xi G+1-\xi,\ 1-\frac{l\xi}{g}\left(2+\frac{m(1-\pi)}{s+m\pi}\right)G\right) \tag{14}$$

Therefore,

$$I_{\max}=\frac{gr-lm+2(1-\xi)lr}{gr-lm+2lr}=\frac{g+\left\{2(1-\xi)-\frac{m}{r}\right\}l}{g+2l-l\frac{m}{r}}$$

Or

$$I_{\max}=\frac{g(s+m\pi)+(1-\xi)l\{2s+(1+\pi)m\}}{(g+2l)(s+m\pi)+(1-\pi)lm}$$

$$=\frac{g+(1-\xi)\left\{2+\frac{(1-\pi)m}{s+m\pi}\right\}l}{g+2l+\frac{(1-\pi)lm}{s+m\pi}}$$

To make $I_{\max}\to g/(g+2l)$, $\xi\to 1$ in addition to $m\to 0$ is required. The left expression in (13) is the line through (1, 1) with a coefficient of $\xi(\geq 1)$. Let G_1 and G_2 denote the GC ratio for which the value of the two expressions in (12) become 0, and G_3 be G for which they intersect each other.

G_1 is

$$G_1=1-\frac{1}{\xi}$$

We can determine the condition of r where $I\geq 0$ by considering the outline of the graph of (13). When $2r-m\neq 0$,

$$G_2=\frac{gr-lm\xi}{l(2r-m)\xi}$$

Formula (13) shows I as π varies from 0 to 1, that is, G varies from $s/(r+s)$ to 1. We assume that $2r-m>0$ because a GC marks a cell faster than an LP consumes a cell in general. (We can analyze the case where $2r-m<0$ or $2r-m=0$ similarly.)

1. if $G_2\geq 1(>G_1)$
   $I\geq 0$ when $G_1\leq G\leq 1$
2. if $G_1\leq G_2<1$
   $I\geq 0$ when $G_1\leq G\leq G_2$
3. if $G_2<G_1$
   $I<0$ when $\frac{s}{r+s}\leq G\leq 1$ (always)

The necessary and sufficient condition where $I\geq 0$ is $G_1\leq G_2$. Figure 4 is the graphic representation of the case of 2.

Similarly, from the right expression in (14), we can see

$$G_2=\frac{g}{l\xi}\cdot\frac{s+m\pi}{2s+m\pi+m}$$

The right expression in (14) is a line with negative coefficient. Hence, the sufficient condition to be $I\geq 0$ is $G_1\leq G_2$. The smaller ξ is, the larger G_2 is, and consequently the range of G where $I\geq 0$ becomes wider.

When the number of GCs increases, not only $I_{\max}$ becomes larger, but G_1, the lower limit of G where the

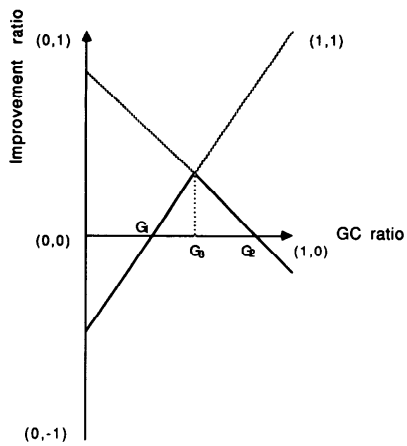


Fig. 4 GC ratio and Improvement ratio.

parallel system has an advantage grows large.

The range of G in general list processing is known to be 10 ~ 30% [9]. If G_1 exceeds 0.3 while running such an application, the improvement ratio is always negative. Especially when $G_2 \geq 1$, it is not desirable to increase the number of GC because the range $(G_1, 1)$ where $I \geq 0$ becomes narrower.

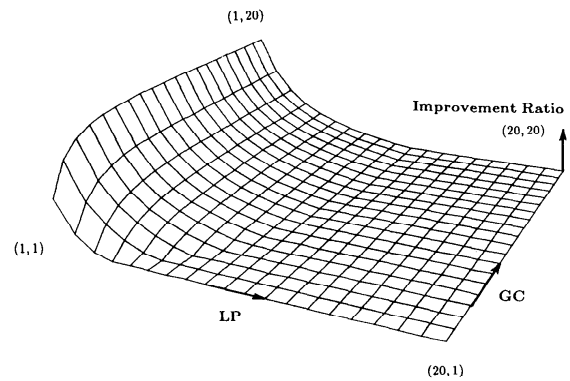
The optimal number of the GCs can be estimated approximately as follows. The GCs ratio depends only on the program to be processed. So we can guess the range of the GC ratio (G_{lo}, G_{hi}) beforehand. $G_1 \leq G_{lo}$ and $G_{hi} \leq G_3$ is required if real-time response is indispensable. But if it does not matter so much and only better performance than sequential system is needed, then $G_1 \leq G_{lo}$ and $G_{hi} \leq G_2$ is sufficient. This condition is not always satisfied because the interval (G_{lo}, G_{hi}) may be wider than (G_1, G_2) in the real system. However, we should not choose g for which there's no (or little, if any) intersection of these two intervals. Consequently, g can be determined so that the relation described above may be satisfied as much as possible.

The dependence of I_{max} on the pair (l, g) are illustrated in Figure 5. Some parameters are supplied their value to draw this graph. The flat plane of the graph is where the sequential system shows better performance. There's little effect by increasing the number of the GCs if current number is more than 4.

4.3 Appending Collected Cells

In this section we analyze the effect of the time to append collected cells, which was ignored in Hickey's analysis. We ignored this factor in 4.1 and 4.2 because this does not affect the results obtained in these sections so much. The number of the cells created in the stable state is not constant, but the set of which is a sequence of difference increasing every other G.C. cycle.

In collecting phase, the whole cell space is scanned to examine the color field of each cell. If the color is white, the cell is appended to the free list. If it is not white, the left pointer field is checked. If its value is f, which indicates it's a free cell, nothing is done while it is whiten otherwise. It takes a longer time to process a garbage cell than to process a non-garbage because of the appending procedure.



Improvement Ratio

Fig. 5 I_{max} as (g, l) vary.

We use the parameters defined by Hickey. The G.C. cycle is assumed to be zero-origin and the LP starts its process at the very beginning of the 0th cycle. As all the black garbage cells existing at the end of the previous cycle are whiten in the collecting phase, we can see that $W_i = B_{i-1}$. The time required for the root insertion phase is negligible.

Let a be the time needed to append a cell to the free list. In the ith $(i \geq 2)$ G.C. cycle,

$$\begin{aligned} \text{tcycle}_i &= \text{tscan}_i + \text{tmark}_i = sN + aW_{i-1} + m(L + B_i) \\ &= sN + mL + aB_{i-2} + mB_i \end{aligned}$$

Hence,

$$\begin{aligned} B_i &= \min(\text{tcycle}_i / r, N - L - B_{i-1}) \\ &= \min\left(\frac{sN + mL + aB_{i-2} + mB_i}{r}, N - L - B_{i-1}\right) \end{aligned} \tag{15}$$

The left expression in (15) shows the stable case while the right is the non stable case. Clearly, $r > m$ is the necessary condition for the LP not to wait. Under this condition, the amount of the ith black garbage is

$$B_i = \frac{sN + mL}{r - m} + \frac{a}{r - m} B_{i-2}$$

Note that there's no reclaimed cell when $i = 0$ or 1,

$$B_0 = B_1 = \frac{sN + mL}{r - m}$$

i.e., $\{B_n\}$ forms a sequence of which value changes every other G.C. cycle. Let $A_n = B_{2n}$, then $B_{2n} = B_{2n-1}$,

$$A_n = B_0 \sum_{k=0}^{n} \left(\frac{a}{r - m}\right)^k$$

If $a < r - m$,

$$\lim_{n \to \infty} \sum_{k=1}^{n} \left(\frac{a}{r - m}\right)^k = \frac{1}{1 - \frac{a}{r - m}} = \frac{r - m}{r - m - a}$$

$$A_\infty = \left(\frac{r - m}{r - m - a}\right) B_0 = \left(1 + \frac{a}{r - m - a}\right) B_0 \tag{16}$$

and the sequence converges to some finite value. If $a \geq r - m$, then A_n diverges, and the LP is forced to wait when B_i exceeds $N - L - B_{i-1}$. It follows that the condition for the system to be stable is

$$0 < a < r - m \quad \text{and} \quad A_\infty \leq \frac{N - L}{2}$$

and the condition to be critical is

$$a \geq r-m \quad \text{or} \quad A_\infty > \frac{N-L}{2}$$

Let γ denote the value of $a/(r-m)$. Suppose that $\gamma<1$ and the number of the LP and the GC is multiplied by n times. The value of r and m can be reduced to $1/n$, while a cannot be always reduced to $1/n$ because the distribution of the garbage is not uniform among every subspaces. Hence γ may exceed 1 where the system would be non stable.

A_∞ shown in (16) is the maximum number of cells created by the LP without waiting and corresponds to G^{crit} described in 2.2. A_∞ can be decreased by reducing a. When a is small, less cell space is required to be stable state. Figure 6 shows how the values of F_i, B_i and W_i change as time varies. The stable state is shown in (a) and non stable state is shown in (b). The system does not always settle into stable state even if the first two cycles are stable. Moreover, if the LP is to wait even once, it would be kept waiting afterwards, that is, the system cannot be alternating.

Thus, the improvement ratio and its maximum value are:

$$I=\min\left(G,\ 1-\left(2+\frac{(m-a)(1-\pi)}{a(1-\pi)+s+m\pi}\right)G\right)$$

$$I_{\max}=\frac{r+a-m}{3r+a-m}=\frac{1+\dfrac{a-m}{r}}{3+\dfrac{a-m}{r}}$$

Or

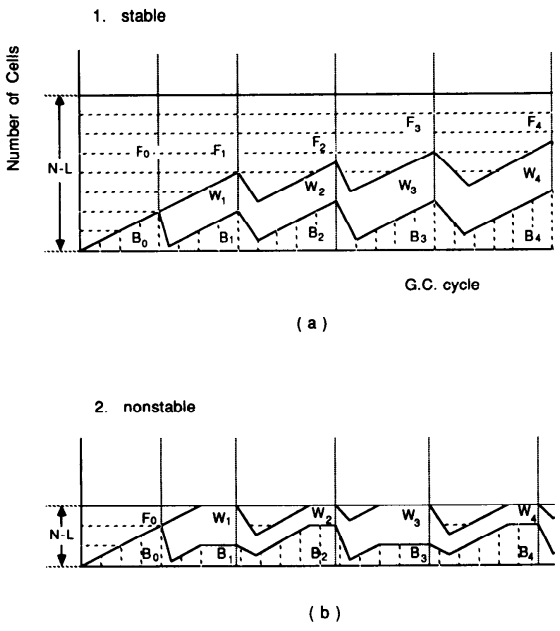


Fig. 6 F_i, B_i and W_i change as time varies.

$$I_{\max}=\frac{s+m\pi+a(1-\pi)}{3s+2m\pi+2a(1-\pi)+m}=\frac{1}{3+\dfrac{(1-\pi)(m-a)}{s+m\pi+a(1-\pi)}}$$

The maximum value of the productivity speedmax $\sigma_{\max}$ is:

$$\sigma_{\max}=\frac{3r-m+a}{2r}=\frac{3}{2}-\frac{m-a}{2r}$$

It follows that when $a-m>0$ then $I_{\max}>1/3$ (i.e., $\sigma_{\max}>3/2$). In other words, the parallel system is more efficient than the sequential system when the time to append a cell is longer than the time to mark a cell. This leads to the fact that parallelizing G.C. would give better performance on such a Lisp machine which can mark a cell faster by special hardware. From the condition $G \leq G^0$ where the parallel system is not worse than the sequential system, $r(1-\pi) \geq s+m$ is acquired. This condition is independent of the value of a.

The analysis of the case of multiple LPs and GCs is similar. The maximum improvement ratio is

$$I_{\max}=\frac{g+\left\{2(1-\xi)-\dfrac{m-a}{r}\right\}l}{g+2l-\dfrac{m-a}{r}l}$$

$$=\frac{g+(1-\xi)\left\{2+\dfrac{(1-\pi)(m-a)}{s+m\pi+a(1-\pi)}\right\}l}{g+2l+\dfrac{(1-\pi)(m-a)l}{s+m\pi+a(1-\pi)}}$$

and we can see that when $m-a \to 0$ and $\xi \to 1$, $I_{\max} \to g/(g+2l)$.

5. The Results of Experiment

We carried out some experiments to confirm the analysis on the multiprocessor Lisp machine SYNAPSE [5]. SYNAPSE has some special hardware to support parallel G.C. One of them is a hardware stack of which access is an indivisible operation. To reduce access conflict, SYNAPSE has 4 memory blocks with their own bus according to their usage. A cell block is one of them and it has 8 Mbytes memory (1 M cells).

We use six versions of Lisp as follow:

1. seq
   The conventional sequential G.C. system
2. para1
   The system with one LP and one GC
3. para2
   The system with one LP and two GCs
4. local
   Reducing access conflict of version 3 by appending the collected cell locally to each GC
5. fast
   Modify the initial structure of the free list of version 4 so that the distribution of the garbage cells may be balanced among each

subspace

6. qcons
   Reducing some overhead of version 5 by dividing the free list into sublists

The modifications in version 4 and 5 are intended to shorten the time to append the collected cells. Compiled function produces no garbage at context switching, so we use an interpreter. Among Lisp contest benchmark programs [8], we choose bit-a [10] and TPU [11], of which G.C. ratio is comparative large. In addition, we define a function named cell-eater, which consumes a lot of cells at a high speed. It is intended to simulate multiple LPs running simultaneously, because we have only one LP.

We measure the value of m, s and a of each version, and the value of r of each program. Some of the results are shown in Table 2 and 3. Table 4 shows the value of B_0, γ and A_∞ of cell-eater when $\pi=0$, where γ is $a/(r-m)$. From the table, the following are shown: as to para1, B_n diverges because $\gamma>1$. So LP is forced to wait. The value of γ of other parallel versions is less than 1, so B_n converge. But the state of para2 will not be stable because $2A_\infty>N$. In fact, para1 takes longer than seq to complete execution, and para2 takes a little longer than local, fast or qcons(about 4%).

Only cell-eater runs faster in the two GCs system than the one GC system when $\pi=0$. Cell-eater consumes cells at least 4 times faster than other benchmark tests. As we described above, this is regarded as multiple LPs running together, or a simulation of some compiled function involving many cons. In other words, 2 GCs can reclaim the garbage cells without interruption if 4 LPs execute these benchmark programs together. From this experiment, we can see that excessive increase of the number of the GCs reduces the performance of the system.

Figure 2 shows the relation between the improvement ratio and the GC ratio as π varies. The function used for evaluation is bit-a. From this figure, we can see that the experiment supports our analysis on the whole. That is, the figure corresponds to the graph of one GC and two GCs in the figure 3 with some execution delay. In version 2~5, I-intercept is about -0.1 and hence $\xi=1.1$ from formula (14). More exactly, ξ is less than 1.1 because there is overhead of mutual exclusion. In version 6, this overhead is reduced to nearly 0, and from the graph $\xi=1.04$. This value is considerably smaller and can be almost ignored in this case.

Table 2 The value of m, s and a.

	m	s	a
seq	46	18	15
para1	51	20	26
para2	28	10	38
local	28	10	18
fast	28	10	10
qcons	28	10	9

(μsec.)

Table 3 The value of r.

	cell-eater	bit-a
seq	55	240
par1 ~ fast	77	260
qcons	59	241

(μsec.)

Table 4 The value of B_0, γ and A_∞.

	B_0	γ	A_∞
para1	0.79N	1.39	∞
para2	0.21N	0.77	0.91N
local	0.21N	0.35	0.32N
fast	0.21N	0.20	0.26N
qcons	0.33N	0.28	0.46N

6. Conclusion

We have analyzed the parallel garbage collection by introducing two parameters, the improvement ratio and the GC ratio. The conclusions derived from our discussion are:

1. The optimal number of GC varies according to the application. To increase the number of GC without

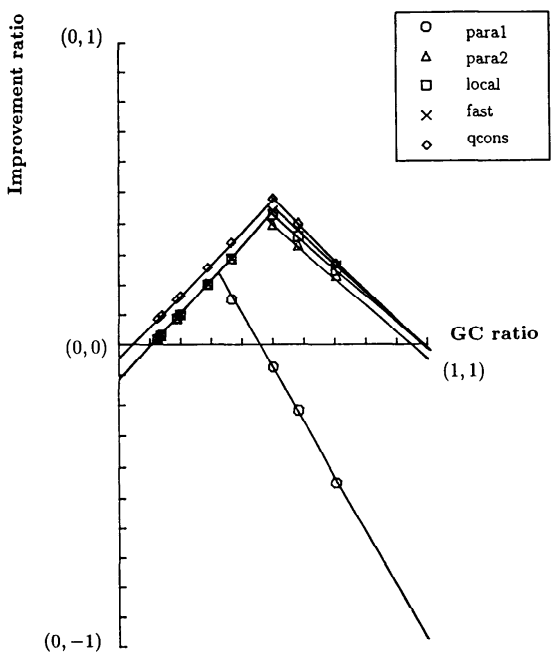


Fig. 7 The result of our experiment.

considering the application is not only meaningless but causes access conflict to be longer processing.

2. Optimal number of GC can be estimated approximately if the value of π and r are provided.

3. Total execution time can be improved by shortening collecting phase. In particular, it is important for collecting process to be distributed equally among multiple GCs and minimize the time to append the collected cells.

4. The improvement ratio in the system with l LPs and g GCs gets near to $g/(g+2l)$ when $m-a\to 0$ and $\xi\to 1$.

5. The condition for parallel G.C. system to be stable is

$$0<a<r-m \quad \text{and} \quad A_\infty \leq \frac{N-L}{2}$$

and the condition to be critical is

$$a \geq r-m \quad \text{or} \quad A_\infty > \frac{N-L}{2}$$

Though our analysis is correct as to some experiments, it is not true in every case. This is due to next two facts: one is that the value of some parameters in sequential G.C. version differ from those in parallel system. Another, this one is considered the main reason, is that the assumption "every newly created garbage is marked garbage" may not be true if the lifetime of the cell is very short. Consequently, this analysis is meaningful in the sense that the worst case of the parallel G.C. can be given as Hickey described.

We conclude that it is not only very effective but necessary to use multiple processors for GC in the parallel G.C. when the amount of cell consumption is large. Ultimately, parallel G.C. is one of the best way of G.C. in the high performance multi-processor list processing system.

Acknowledgement

The author wishes to thank Prof. Masakazu Nakanishi for his comment.

References

1. Dijkstra, E. W. et al. On-the-Fly Garbage Collection: An Exercise in Cooperation, in Lecture note in Computer Science, No. 46. Springer-Verlag, New York (1976): *Comm. ACM.* **21**(11) (1978), 966–975.
2. Hickey, T. et al. Performance Analysis of On-the-Fly Garbage Collection, *Comm. ACM* **27**(11) (1984), 1143–1154.
3. Kung, H. T. et al. An efficient parallel garbage collection system and its correctness proof, Tech. Note. Dep. Computer Sci., Carnegie-Mellon Univ., Pittsburge (1977).
4. Lamport, L. Garbage Collection with Multiple Processes: An Exercise in Parallelism, Proceedings of the International Conference on 'Parallel Processing', Walden Woods (1976), 50–54.
5. Matsui, S. et al. SYNAPSE: A Multi-microprocessor Lisp Machine with Parallel Garbage Collector, Proceedings of the International Workshop on Parallel Algorithms and Architectures (1987), 131–137.
6. Newman, I. A. et al. Alternative Approach to Multiprocessor Garbage Collection, Proceedings of the International Conference on 'Parallel Processing' (IEEE Computer Society) (1982), 205–210.
7. Newman, I. A. et al. A Hybrid Multiple Processor Garbage Collection Algorithm, The Computer Journal, **30**, 2 (1987), 119–127.
8. Okuno, H., G. The Report of The Third Lisp Contest and The Prolog Contest, **33**, 4, *IPS Japan* (1985).
9. Wadler, P. L. Analysis of an Algorithm for Real Time Garbage Collection, *Comm. ACM* **19**(9) (1976), 491–500.
10. Nakanishi, M. Lisp nyuumon-SYSTEM TO PROGRAM (Japanese), Kindai-kagakusha (1986).
11. Chang, C. et al. Symbolic Logic and Mechanical Theorem proving, Academic Press (1973).

(Received December 21, 1987; revised January 19, 1989)