# Global and Local Page Replacement Algorithms on Virtual Memory Systems for Image Processing

Ben Tsutom WADA

**ABSTRACT**

Three virtual memory systems for image processing, different one another in frame allocation algorithms and page replacement algorithms, were examined experimentally upon their page-fault characteristics. The hypothesis, that global page replacement algorithms are susceptible to thrashing, held in the raster scan experiment, while it did not in another non raster-scan experiment. The results of the experiments may be useful also in making parallel image processors more efficient, while they are of completely serial basis.

**Keywords and Phrases**

virtual memory, image processing, picture processing, page replacement, frame allocation, page fault frequency, raster scan, virtual array, special-purpose memory hierarchy, parallel processor, image processor.

## 1. Introduction

In the previous work, I made a software-implemented virtual memory system, RH2, that is aimed specifically at image processing (or picture processing; I treat them as synonyms) [7]. Although I evaluated it with some experiments as described in the paper, it was done by itself; no comparative experiments were not made.

In that paper, I assumed that global page-replacement algorithms are susceptible to the thrashing according to Denning [2,3], and hence used a local page replacement algorithm. But there was no experimental proof specifically for that system.

In this paper, I show the results of the experiments that compare local algorithms similar to that used in RH2 with a global one, for confirming the hypothesis noticed above.

A part of contents of this paper has been reported by the author at the 30th National Conference of Information Processing Society of Japan, Tokyo, March, 1985, and inserted in its proceedings. A part of the auther's previous paper[7], including figures, was reused for this paper. Acknowledgement is paid to the Information Processing Society of Japan and the Association for Computing Machinery.

Author's Present Address: Ben T. WADA, Nagano University, Shimonogo, Ueda-shi, NAGANO 386-12 JAPAN .

This paper is so self-contained as possible. But it is desirable for readers to see the original paper on the Raster Handler 2 (RH2) [7] together.

## 2. The Virtual Memory Systems

In this section, the systems I examined, or used for comparing purpose, are described. Basically, they are similar to RH2.

They are virtual memory systems of paging method (not segmentation method.) They handle square black-and-white image data only. Programs or non-image data are not handled at all. As they are implemented in software, they can also be said to be simulations of virtual memory systems. They can handle images of 256×256 pixels only. And only two images are handled simultaneously. (Because they are made for exprerimental purpose. The original RH2 can handle square images of various size, and many images simultaneously.) An image is divided into 256 pieces of 16×16 square blocks. Each of them is a page, that is, an unit for replacement. Page replacement is performed in demand basis. (RH 2 uses also asynchronous page replacement.)

I implemented, estimated, and compared three different systems. The features noticed above are common to the three. They differ in the frame allocation and the page replacement algorithm. Following subsections describe on each of them.

### 2.1. Purized RH2 algorithm

This algorithm is basically the same as RH2's one. In RH2, the asynchronous page replacement and the secondary allocation 'disorder' the page replacement pattern. (Secondary allocation is the feature that keeps some amount of frames unused for a while at the beginning of the execution, and allocates them later.) In this experiment, a slightly 'pure' algorithm was used.

#### Algorithm

#### Frame allocation

* Choose *a priori* the image that is to be accessed more frequently than the other.

* Allocate all available frames to each image as follows in advance of the execution.

| No. of available frames | No. of frames allocated to the image more accessed. | No. of frames allocated to the image less accessed. |
|---|---|---|
| 64≦nofr | Halved | |
| 50≦nofr<64 | 32 | 31～18 |
| 36≦nofr<50 | 31～18 | 18 |
| 22≦nofr<36 | 18 | 17～4 |
| 8≦nofr<22 | 17～4 | 4 |
| nofr<8 | Out of this algorithm | |

* Frames do not move between images during the execution, that is, the allocation is static.

**Page Replacement**

FINUFO (or second chance) algorithm is applied locally to each image. FINUFO algorithm is such one as follows:

It contains a pointer that points to every frame cyclically, and each frame has a flag or usage bit that is set whenever the page in the frame is accessed. When it is time to discard a frame to make room, the frame that the pointer has in view at the time is looked at first. If its usage bit is not set, the frame is discarded; if set, the bit is then reset and the pointer advances to the next frame, and so on. The usage bit of a frame is set when any page enters the frame.

Immediately after the required page has been brought up with the FINUFO algorithm as above, the frame that should be used next is sought using the FINUFO algorithm again, and the pointer remains there until the next page fault. Therefore the frame the pointer gets in view will be discarded at the next time, unless it will have been accessed by that time.

## 2.2. Best-of-localFINUFO algorithm

The second algorithm, Best-of-localFINUFO, involves a frame allocation algorithm entirely 'unrealizable.' It was experimented for proving the limitation of the local page replacement algorithm. The involved page replacement algorithm is identical to that of the purized RH2 algorithm.

The allocation algorithm is as follows. It distributes all available frame to each image so as to minimize the sum of page faults under the page replacement algorithm. That is, every possible static allocation is tested and the best one is chosen as the final result. For example. in the case that 40 frames are available, 33 to 7, 32 to 8,⋯⋯8 to 32, 7 to 33 are all tested. (The cases that an image owns more than 33 frames were not experimented, expecting that there would be no unnegligible changes in the result. Similarly, the cases that an image owns less than four frames are also not experimented.)

**Note.** As each image is independent in local page replacement environment, the page fault frequency in the purized RH2 algorithm and that in the best-of-localFINUFO algorithm can be estimated at the same time. At first, the page fault frequencies of each image under various number of owning frames are estimated; and the frequencies of the entire system under each allocation algorithm were calculated from them.

## 2.3. Global FINUFO Algorithm

The third algorithm is the global FINUFO algorithm. That is, the same page replacement algorithm as those in two subsections above is applied to the whole system with no consideration as to which page belongs to which image. Therefore, frames are not allocated statically to images, but images would give and take frames to／from the other during the execution.

## 3. Experiments and Results

This section presents the results of two experiments with different application programs. In both experiments, each image processing program is executed repeatedly under each of three algorithms described in the previous section, varying the number of frames available to the whole system. The image processing programs used are the same ones as used in the previous paper [7]. One experiment is a raster scan case, whereas the other is not.

### 3.1. Raster Scan Case

In the raster scan experiment, the approximate gradient of each pixel is computed using two images at the same time; one of the images contains the input data, and the other will contain the computed result. The program computes the differences between each pixel in an image and each of the surrounding eight pixels; the maximum one is put onto the corresponding position of the output image.

Each of the eight computations of differences involves an access to the central pixel. Therefore, 16 accesses are made on the input image in processing one particular pixel; in total, $16 \times 256 \times 256$ or about a million accesses are made. On the other hand, the output image is accessed only once during the processing of each pixel, that is, when the result is entered.

The operation moves rowwise, from top to bottom, and from left to right in each row (Figure 1.) The order of the 16 accesses is as shown in Figure 2.

```
1 2.......256
257.......512
.............
.............
.............
.............
........65536
```

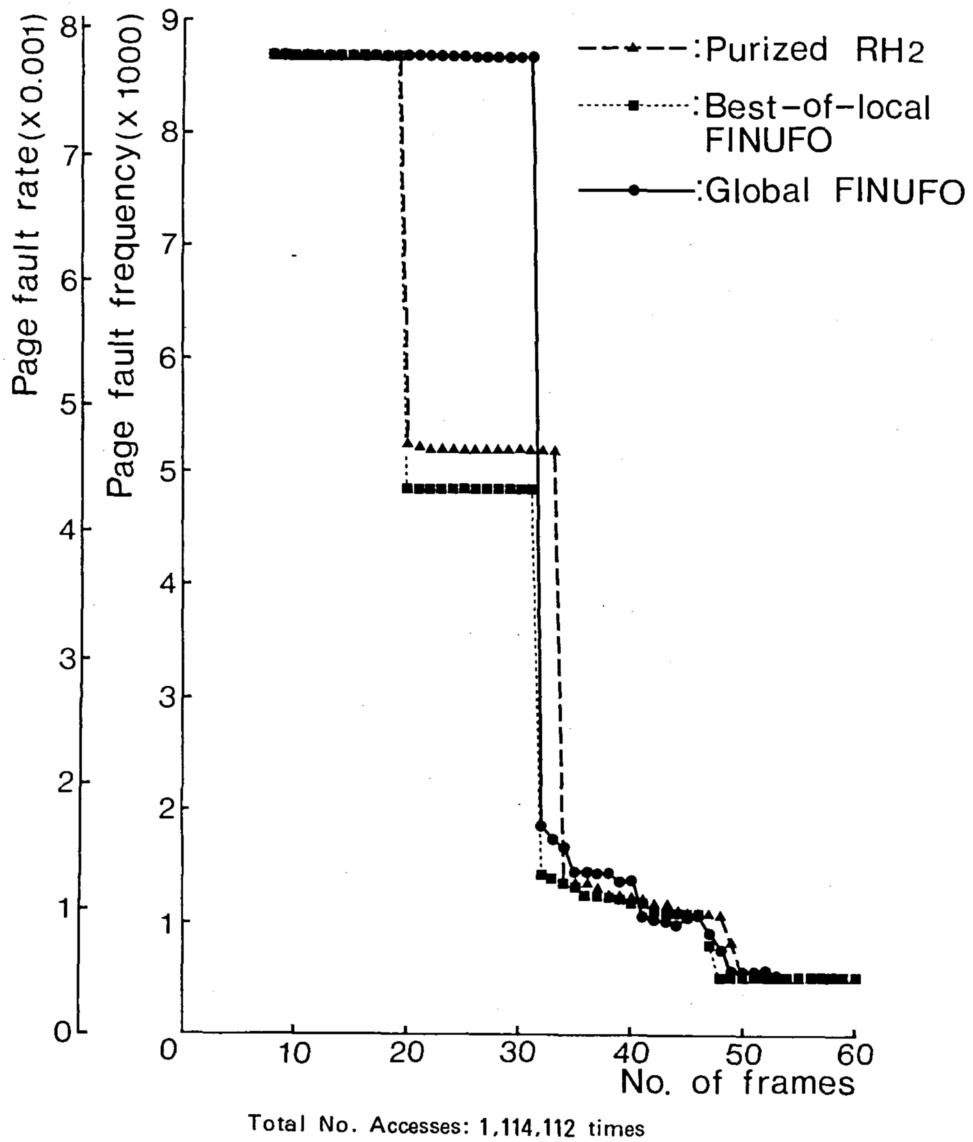
Figure 1. **The Order of Operation**

| | | |
|---|---|---|
| 1 | 3 | 5 |
| 7 | * | 9 |
| 11 | 13 | 15 |

The central pixel. The 2, 4, 6, . . . , 16th access are made on this pixel.

Figure 2. **Accessing Order Within the Operation on a Pixel**

The result of this experiment is shown in Figure 3. There the page fault frequencies of the whole system under each page replacement algorithm with various number of available frames are shown. The result almost follows the hypothesis introduced in ***introduction***, that global page replacement algorithms are susceptible to thrashing; local algorithms hold out against the decrease of frames to the limit, while the global one breaks down much earlier. It is natural that the purized RH2 algorithm works as nearly well as the best-of-local-FINUFO algorithm, because it was designed so as to fit such applications as this example.



Total No. Accesses: 1,114,112 times

Figure 3. Page Fault Frequency - Raster Scan Operation (Computation of the Gradient)

## 3.2 Non Raster-scan Case

Here I have tested a kind of 'region painting' as an example of non raster-scan operation. In this operation, depending upon whether or not each of the pixel couples adjacent horizontally or vertically in the image is linked, the entire image is divided up into several uniquely-numbered regions each of which is made up of pixels linked together. As a result, a pseudo-image is output in which each pixel possesses a number identifying the region to which it belongs. In actually, the program receives an image—not the linking information—and regards two adjacent pixels as linked if their values in the input image differ by less than the given threshold. The actual algorithm is shown in Figure 4. It extends a region as far as possible beginning with a single pixel that belongs to none of the existing regions.

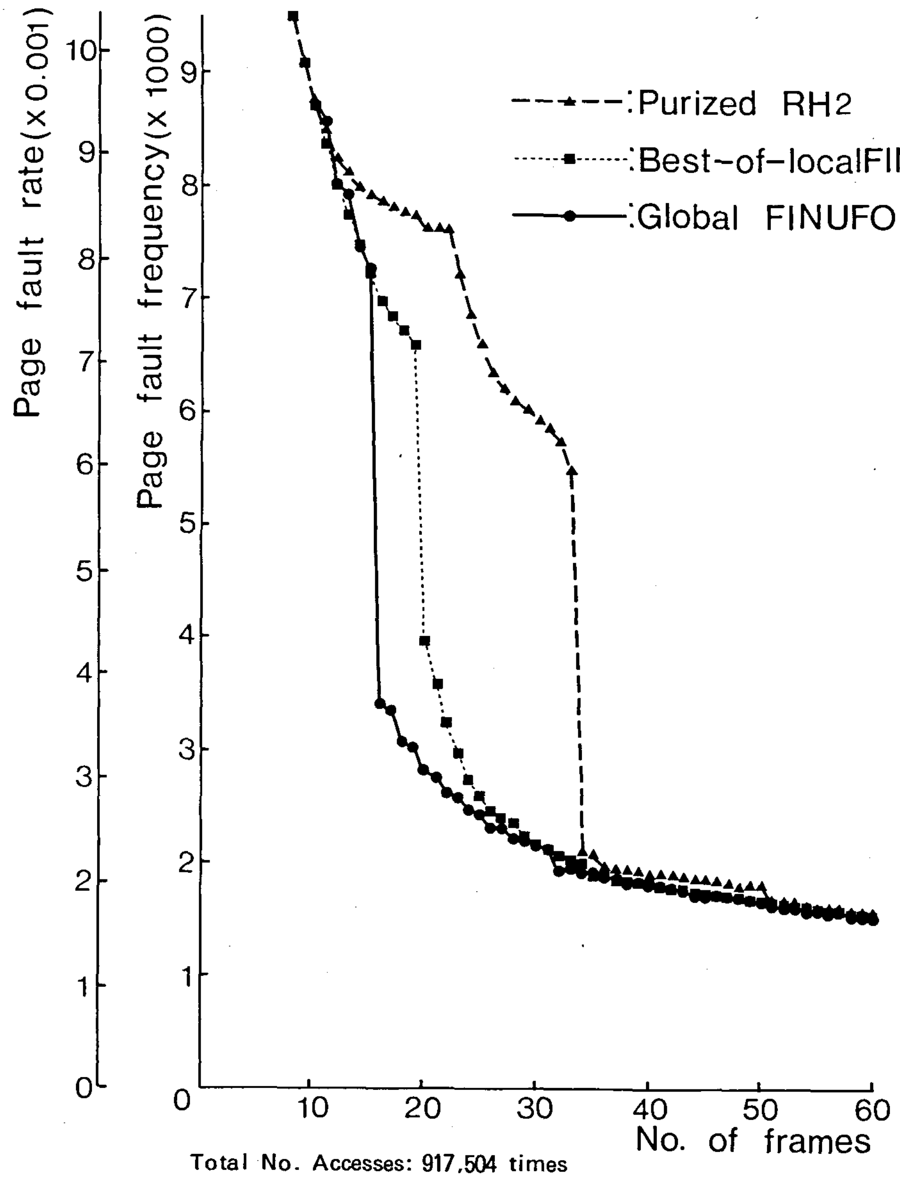
```
Initially, every pixel in the output image is set to zero (i.e., "not
labeled").
begin
label_no:=0;
for y:=1 to bottommost do
  for x:=1 to rightmost do
    if pixel (x,y) in the output image is zero then
      begin
      push x and y into the stack;
      label_no:=label_no+1;
      let pixel (x,y) in the output image be label_no;
      while the stack is not empty do
        begin
        pop the stack and put them into x' and y';
        for x" and y" := each of four adjacent pixels of (x',y')
        do
          if pixel (x",y") in the output image is zero and (x",y")
                is linked to (x',y') then
            begin
            push x" and y" into stack;
            let pixel (x",y") in the output image be label_no;
            end
        end
      end
end
```

Figure 4. **Region Painting Algorithm**

The results of this experiment depend greatly on the contents of the input image. In this case, I digitized the March 1982 Cover of Communications of the ACM and used it for that purpose. It contains only simple patterns, as shown also in [7]. The threshold was so adjusted, that regions visible to human eye would be destinguished, and that signal noises would never divide any true regions falsely.

The result of this experiment is shown in Figure 5. There the page fault frequencies of the whole system are shown. In contrast to the raster scan case, the result is wholly against the hypothesis described in *introduction*; the global algorithm is the *least* susceptible to thrashing of three.

There are two 'modes' in the pixel-accessing pattern of this painting programs; one is to seek a pixel that belongs to none of existing regions, and the other is to paint the region in sight. In the former mode, the input image is not accessed at all. Under static frame-



Total No. Accesses: 917,504 times

Figure 5. Page Fault Frequency - Non Raster-scan Operation (Region Painting)

allocation strategy, the frames allocated to the input image remain unused at all during the mode; on the other hand, under dynamic allocation strategies, such as global FINUFO, unused frames move to the other image sooner or later. Therefore, local page replacement algorithms do not work well on such applications, unless some frame re-allocation is involved.

## 4. Conclusions

Three virtual memory systems for image processing, with page replacement algorithms different one another, are experimented under two different application programs. In an experiment, raster scan case, the result followed the hypothesis that global page replacement algorithms are susceptible to thrashing. In the other, non raster-scan case, the global one worked best in three; the hypothesis did not hold, as far as compared with two others that allocate frames statically to images.

While these systems, including the original RH2, operate completely serially, these methods are not necessarily incompatible to the parallel processing.

There exist several 'image processors'. The parallelisms used there can be divided into some categories 〔4,5,6〕. Among others, 'Completely-parallel' processors, that allocate a processing element to each of pixels, are very fast. To this kind of parallel procesing, the methods of RH2 or other systems are incompatible. However, mostly due to the cost, there are only a few completely-parallel processors, and most of them can handle only relatively small images at once. Greater images are processed serially in portion by portion basis. (In general, this method is called 'SIMD'〔4〕, 'Plane-parallel'〔5〕, or 'Pixel-parallel'〔6〕 processing; completely-parallel method is a special case.)

In almost cases of parallel image processing, including the method described above, and 'MIMD' 〔4〕 or 'Local-parallel' 〔5,6〕 processing, only a small portion of an image is handled at once. The portion to be processed - maybe a single pixel, or maybe a 128×128 square block (in MPP 〔1〕) - moves probably in similar pattern to the examples experimented in this paper. It means that the results of these experiments and related research works are possibly useful also in making parallel image processors more efficient.

## References

1. Batcher,K.E. MPP: A Supersystem for Satellite Image Processing. In *National Computer Conference,* AFIPS , 1982, pp. 185-191.
2. Denning, P. J. Virtual Memory. *ACM Comput. Surv. 2* , 3(Sep.1970), pp.153-189.
3. Denning, P. J., and Graham, G.S. Multiprogrammed Memory Management. In *Virtual Storage,* C. Boon and C. J. Bunyan, Eds. INFOTECH, International, Maidenhead, U.K.,1976, pp.237-271.
4. Hwang, K., and Fu, K.S. Integrated Computer Architectures for Image Processing and Detabase Management. *IEEE Computer 16* , 1(Jan.1983) , pp. 51-60.
5. Sakaue,K., and Kidode, M. Recent Trend of Image Processor. *Journal of the Institute of Electronics and Communication Engineers of Japan 67* , 1 (Jan. 1984), pp.90-98 (In Japanese.)
6. Temma,T. Processor for Image Processing. *IPSJ Joho Shori 25* , 9 (Sep.1984), pp. 909-917 (In Japanese.)
7. Wada,B.T. A Virtual Memory System for Picture Processing. *Comm. ACM 27* , 5 (May 1984) , pp.444-454.