# POLYPHASE MERGE SORTING -- AN ADVANCED TECHNIQUE

R. L. Gilstad
Minneapolis-Honeywell Regulator Company
Electronic Data Processing Division
Wellesley Hills, Massachusetts

## The Challenge

Designers of generalized library sort packages for the current and future generations of computers are faced with the challenge of developing new techniques that provide more effective use of these computers. The major concern in developing efficient sorting routines in the past has been the internal sorting techniques, that is, the methods of manipulating the data within the memory of the computer. Precise methods must, of course, be devised for each new computer design but, due to the extensive effort in this area in the past, few new internal sorting techniques have been introduced for, what are in computer terms, generations.

Emphasis is now being given toward more effective use of the tape drives used by a sort routine. Progress toward this end was reported in the paper "New Merge Sorting Techniques", presented by B. K. Betz at the September 1959, ACM Conference. The paper then presented described in theory an advanced merging technique, originally called the "N-1" technique, now a proven method better known as the Cascade sorting technique.

The intention of this paper is to introduce a new merging technique, polyphase sorting. The following section describing the application of the Cascade sorting technique is included to aid in the understanding of the evolution of the polyphase sorting technique and to prepare for certain comparisons later in this paper between the various sorting techniques. A complete study of the changes in merge sorting would, of course, include a description of the process that is referred to in this paper as normal merge sorting and that has such names as two-way merge sorting and three-way merge sorting. Because of the extensive use of normal merge sorting techniques, this paper assumes a general understanding of them by those interested in this subject.

## Cascade Sorting

The Cascade Merge Sort, available exclusively in the Honeywell 800 automatic programming packages, is a two-segment program, the first part of which is an internal sort that creates strings of ordered items. The internal sorting method that has proven to be most advantageous to Honeywell for generalized sort generators uses the "tag bin" concept which transfers internally only a tag representing each item stored in memory, instead of transferring the entire item. Further, the "replacement" sorting method is added, which creates strings of ordered items substantially longer than the number of items stored in memory. This method, in fact, provides strings averaging twice the number of items stored in memory for randomly ordered input data and longer strings if any pre-ordering exists in the input file.

The only real difference between the internal sort, hereafter called the pre-sort, for Cascade sorting and normal sorting is the manner of distributing the strings onto the work tapes used by the sort. Normal merge sorts require that the strings be distributed alternately on two work tapes for a two-way merge sort, or on three tapes for a three-way merge sort. The pre-sort for a Cascade sort distributes the strings of sorted records onto all but one of the work tapes available to the sort. The ideal distribution at the completion of the pre-sort is such that there are fewer strings on each succeeding tape. The exact distribution is based on one of several sequences, depending on the number of tape drives being used. The sequence for the three-tape sort is the Fibonacci sequence:

$$1,1,2,3,5,8,13,21.....,$$

while the sequence for a four-tape Cascade sort is the sequence:

$$1,1,1,2,3,5,6,11,14,25,31..... .$$

If the pre-sort for a four-tape Cascade sort creates 14 strings, the distribution of strings on the three work tapes would be six strings, five strings, and three strings.

The determination of the distribution of strings by the pre-sort can be in the form of a pair of counters for each tape being used as an output tape. One counter for each of the tapes contains the ideal distribution for one merge pass, while the second counter contains the total number of actual strings written on each tape. Strings are written onto each tape until the pair of counters for that tape are equal. When all of the counters are equal for one ideal distribution, the ideal distribution counters are updated to the values for an additional merge pass. The flow of the distribution process and the use of the counters is shown for the pre-sort for a four-tape Cascade sort in Figure I.

The second segment of the Cascade sort is a merge sort, during which each pass over the file begins with an N-1 way merge (where N is the number of tapes available to the sort) that continues until the work tape with the least number

of strings is depleted. As each tape is depleted, the way-merge is decreased by one until the pass is concluded by a one-way merge (copying), depleting what was the longest work tape.

Many methods for demonstrating the flow of information through a merge sort have been developed and used, none of them to the satisfaction of this writer. A picture does, however, replace a thousand words, so a Cascade sort is pictured in Figure II using numbers representing the number of strings on each tape at each step of the sort.

The power of this sort technique is derived from the fact that a larger percentage of the file is merged during the most powerful way-merge than is merged during the later phases of the pass; more specifically, for a four-tape sort, 52% of the file is merged during the three-way merge, 36% is merged during the two-way merge and only 12% of the file is involved in the copy portion of the pass. A normal four-tape merge sort is continually performing a two-way merge, giving it a merging power of 2. The Cascade sort for a like number of tape drives has a merging power of 2.3. That is, it reduces the number of total strings by a factor of 2.3 each pass.

A further advantage of the Cascade sort, which is implied above, is that an odd as well as an even number of tape drives, and as few as three, can be used to full advantage.

The sort routines described above assume a read-backward merge sort, which eliminates the need for rewinding during the sort, but which requires the copy portion of each pass in order to reverse the order of the remaining strings on the long work tape. (A read-backward sort must switch the strings from ascending order to descending order on successive passes.) The same Cascade method is available for a read-forward merge, which must, of course, rewind certain tapes during and between passes. This allows the elimination of the copy portion of each pass, as all of the strings are always in ascending order. In order for a read-forward Cascade sort to retain a time advantage over normal read-forward merging, tape rewinding must be a faster process than the tape reading process. All of the comparisons between normal merging and Cascade sorting involving rewinding are based on the Honeywell 400, which has a 3 to 1 ratio of rewind speed over reading speed.

## Polyphase Sorting

Further advancement in the efficient use of the tape drives used by a sort has developed from the Cascade sort. This new sorting technique is called the polyphase sorting.

The polyphase sort, like most merge sorting methods, is a two-segment program, a pre-sort segment and a merge segment. Again the pre-sort distributes the ordered strings onto all but one of the available tapes, based upon one of several sequences. The sequences for the various tape drive configurations are as follows:

3-tape 1,1,2,3,5,8,13,21.....(It is noted that this is the same sequence as for a 3-tape Cascade sort. Indeed, a 3-tape read-forward Cascade sort is the same as a 3-tape polyphase sort.)

4-tape 1,1,1,2,2,3,4,6,7,11,13,20,24.....

5-tape 1,1,1,1,2,2,2,3,4,4,6,7,8,12,14,15.....

6-tape 1,1,1,1,1,2,2,2,2,3,4,4,4,6,7,8,8,12,14, 15,16.....

During the merge segment of a polyphase sort, a continuous N-1 way merge is performed. At the beginning of the polyphase merge segment, the N-1 way merge is performed until the tape with the least number of strings is depleted. At this point, instead of switching to an N-2 way merge as in the Cascade sort, an N-1 way merge is continued, merging additional strings from the tapes not yet depleted, with strings from the tape just created. Because this process is continued throughout the merge, there is no point that can be called a complete pass over the file. Instead, there are a series of phases, wherein some strings from a number of previous phases are merged together.

The four-tape polyphase sort example in Figure III shows the effect of this technique through an entire sort. The numbers given in the example represent the number of strings at each of the various phases of the sort.

## Comparing the Merges

The power of a polyphase sort is not easily discernible or readily comparable to other sorting techniques, due to the fact that the phases described above cannot be compared directly with the passes of the other techniques. A normal two-way merge sort, for example, processes the entire file being sorted during each merge pass and in so doing, reduces the number of strings by one-half. Another way of stating this is that during each merge pass the length of each string is doubled. This is abbreviated by stating that a two-way merge pass has a power of two. Each step shown in the polyphase sort example processes only a portion of the file. Therefore, while it can be determined from the tables in Figure IV that a four-tape polyphase sort has a power of 1.88 per step as compared with a power of 2.0 for a normal sort using four tape drives, the polyphase is significantly faster because it processes only 62% of the file during each step as compared to the 100% processed during each step of a normal sort.

Figure I contains two tables which show the total number of strings that are merged together for the given number of steps of the merge sort

for four- and six-tape normal, Cascade, and polyphase sorting. The tables also give the percentage of file processed for each step (phase). This percentage is the average for a number of phases in the case of polyphase sorting, the specific values varying slightly from phase to phase.

Further complications in the comparison of the power of the several sort techniques include the internal machine speeds, the amount of simultaneous operation, whether the sort must include tape rewinding, and if so, the relative rewinding speed of the tape mechanism. Figure V relates normal, Cascade, and polyphase sorting as performed on a computer capable of reading backwards and performing all processing at full tape speed. The figures for the polyphase sort have been equivalenced to the same percentage of file processed as for the normal and Cascade sorts.

Figure VI describes graphically the relationship of the power of the three sorting techniques. The graph shows the number of passes over the file required to merge a given number of strings together into one string with four tape drives.

No attempt has been made to date to implement a read-backward polyphase sort. The delay in doing so is caused by the necessity to alternate ascending and descending strings on each of the work tapes during the pre-sort. This alternation of strings destroys the advantage of a variable length string pre-sort whenever some degree of pre-ordering exists in the input to the sort routine. Pure random input data would be sorted faster with a polyphase sort, but experience indicates pre-ordering to some extent exists on the majority of files to be sorted.

## Read-Forward Merging

Merge sort routines for computers that allow only read-forward tape operations must rewind a certain percentage of the file between succeeding steps of the merge. A normal two-way merge, for example, rewinds the entire file after each pass of the merge, but, because the file is distributed evenly on two tapes that can be rewound simultaneously, the rewind time for each pass is one-half of the time to rewind the entire file.

A read-forward Cascade sort rewinds a larger percentage of the file each pass than does a normal sort, but retains a time advantage because it does not process the entire file each pass. A four-tape, read-backwards Cascade sort performs a three-way merge over 56% of the file, then must rewind the newly created output tape and the input tape that was depleted. The rewind of the depleted input tape, which delays the operation, is over 19% of the file during the first pass. The Cascade sort then performs a two-way merge of 34% of the file and rewinds the same percentage of the file. The remaining 10% of the file on the third tape becomes input to the next pass and is not processed during the current pass. After the three-way merge on the second and all succeeding passes, the input tape to be rewound includes information used during the three-way merge and two-way merge of the previous pass as well as during the three-way merge of the current pass. This amounts to 56% of the file. Normally, therefore, the Cascade sort processes 90% of the file and rewinds 90% of the file during each merge pass.

A read-forward polyphase sort rewinds the same percentage of the file that it processes each phase. A four-tape read-forward polyphase sort processes and rewinds 62% of the file during each phase. As did the Cascade sort, the polyphase sort depends upon a faster rewinding process than merging process to maintain its full advantage over normal sorting.

## Conclusion

The two new merge sorting techniques presented here are now working programs, proven to be the tools for a more efficient computer operation. Cascade sorting provides the efficiency for read-backward operations. Polyphase sorting provides the efficiency for read-forward operations and in the future can provide the efficiency for read-backward operations in cases where the file to be sorted has been determined to be in random order.

There is one hardware prerequisite which should be mentioned, which is necessary to obtain the full advantage of Cascade and polyphase sorting. If reading and writing are to be simultaneous, the sort must be able to read from one tape and write on another in any combination involving the tapes used by the sort.

It is worth noting that improved approaches to the common, everyday computer problems are still being found in an era where the emphasis is on new uses for computers and new designs for computer hardware.

| | Ideal Distribution Counters (Total Number of Strings) | | | Distribution of Strings (Successive String Numbers) | | |
|---|---|---|---|---|---|---|
| | Tape A | Tape B | Tape C | Tape A | Tape B | Tape C |
| One merge pass | 1 | 1 | 1 | 1 | 2 | 3 |
| Two merge passes | 3 | 2 | 1 | 4,5 | 6 | |
| Three merge passes | 6 | 5 | 3 | 7,8,9 | 10,11,12 | 13,14 |
| Four merge passes | 14 | 11 | 6 | 15-22 | 23-28 | 29-31 |

Fig. 1. String Distribution for a Cascade Sort

| Tape A | Tape B | Tape C | Tape D | |
|---|---|---|---|---|
| 14 | 11 | 6 | 0 | Output from pre-sort |
| 8 | 5 | 0 | 6 | After three-way merge of 6 strings |
| 3 | 0 | 5 | (6) | After two-way merge of 5 strings |
| 0 | 3 | (5) | (6) | After copy of 3 strings |
| | | | | End of first pass |
| 3 | 0 | 2 | 3 | After three-way merge of 3 strings |
| (3) | 2 | 0 | 1 | After two-way merge of 2 strings |
| (3) | (2) | 1 | 0 | After copy of 1 string |
| | | | | End of second pass |
| 2 | 1 | 0 | 1 | After three-way merge of 1 string |
| 1 | 0 | 1 | (1) | After two-way merge of 1 string |
| 0 | 1 | (1) | (1) | After copy of 1 string |
| | | | | End of third pass |
| 1 | 0 | 0 | 0 | After three-way merge of 1 string |
| | | | | End of sort |

Note: All numbers represent the number of strings on each tape at each step. The underlined numbers are the output at each step.

Fig. 2. Cascade Sorting.

| Tape A | Tape B | Tape C | Tape D | |
|---|---|---|---|---|
| 13 | 20 | 24 | - | Output of pre-sort |
| - | 7 | 11 | 13 | After three-way merge of 13 strings |
| 7 | - | 4 | 6 | After three-way merge of 7 strings |
| 3 | 4 | - | 2 | After three-way merge of 4 strings |
| 1 | 2 | 2 | - | After three-way merge of 2 strings |
| - | 1 | 1 | 1 | After three-way merge of 1 string |
| 1 | - | - | - | After three-way merge of 1 string |

Fig. 3. Polyphase Sorting.

Table A -- Four-tape Sorts

| Steps | Normal | Cascade | Polyphase |
|---|---|---|---|
| 1 | 2 | 3 | 3 |
| 2 | 4 | 6 | 5 |
| 3 | 8 | 14 | 9 |
| 4 | 16 | 31 | 17 |
| 5 | 32 | 70 | 31 |
| 6 | 64 | 157 | 57 |
| 7 | 128 | 353 | 105 |
| 8 | 256 | 793 | 193 |
| 9 | 512 | 1782 | 355 |
| 10 | 1024 | 4004 | 653 |
| 11 | 2048 | 8997 | 1201 |
| 12 | 4096 | 20,216 | 2209 |
| % of file processed per step | 100% | 100% | 62% |

Table B -- Six-tape Sorts

| Steps | Normal | Cascade | Polyphase |
|---|---|---|---|
| 1 | 3 | 5 | 5 |
| 2 | 9 | 15 | 9 |
| 3 | 27 | 55 | 17 |
| 4 | 81 | 190 | 33 |
| 5 | 243 | 671 | 65 |
| 6 | 729 | 2353 | 129 |
| 7 | 2187 | 8272 | 253 |
| 8 | 6561 | 29,056 | 497 |
| % of file processed per step | 100% | 100% | 55% |

Fig. 4. Number of Strings Merged.

Table of power of read-backward sorts with a tape limited operation.

| Number of Tapes Used by Merge | Power of Normal Sorting | Power of Cascade Sorting | Power of Polyphase Sorting |
|---|---|---|---|
| 3 tapes | 1.5 | 1.61 | 1.80 |
| 4 tapes | 2 | 2.30 | 2.79 |
| 5 tapes | 2.5 | 2.94 | 3.44 |
| 6 tapes | 3 | 3.62 | 3.86 |

Note: The power of a sort, as used here, is the factor by which the number of strings decreases for each full read time of the file being sorted.

Fig. 5. Power of Three Sorting Techniques.

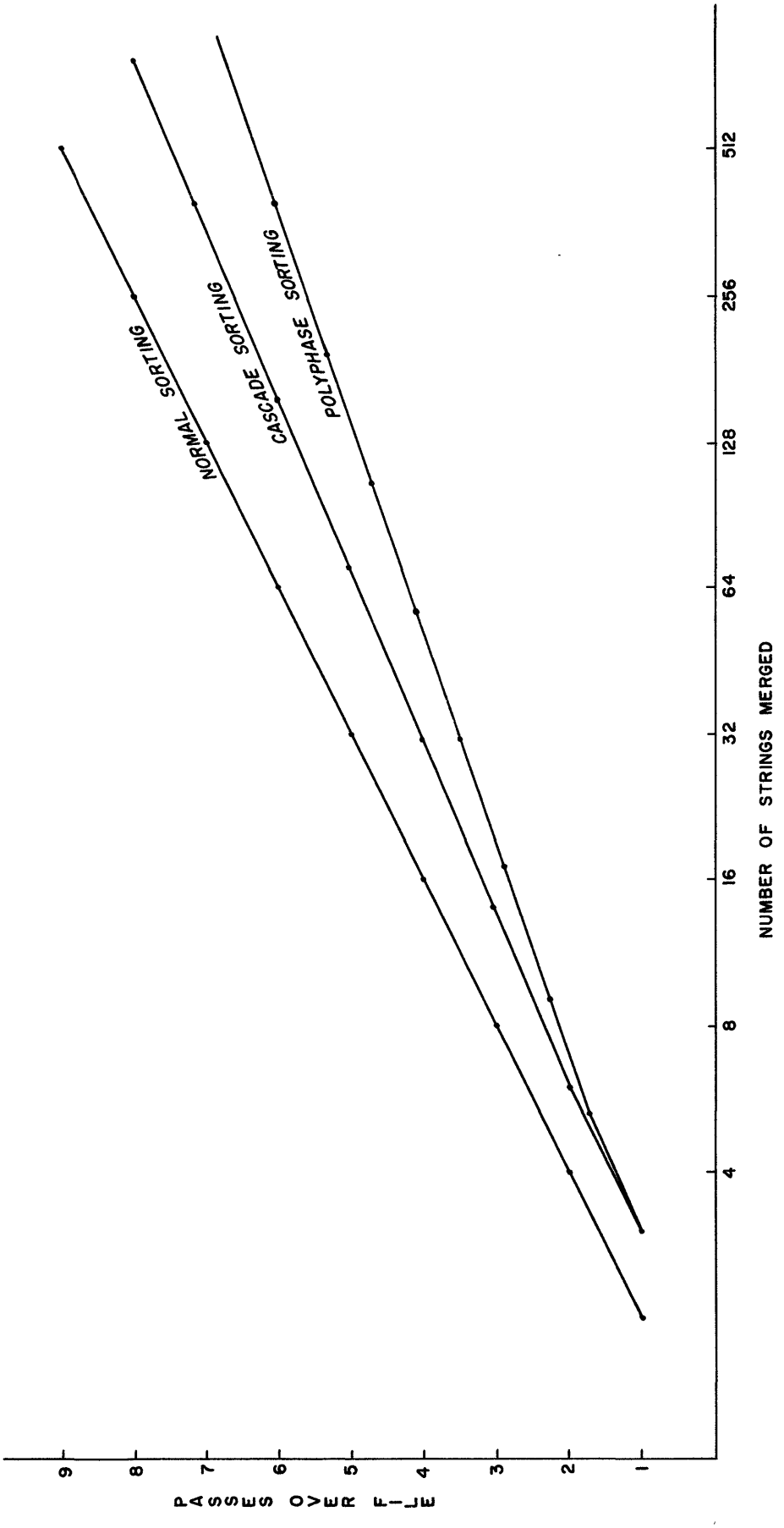


Fig. 6. Comparison of Three Sorting Techniques.