Parallel Sorting - the Zamfir Way

D.C. van Leijenhorst

Computing Science Institute/

CSI-R9804 January 1998

Computing Science Institute Nijmegen
Faculty of Mathematics and Informatics
Catholic University of Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands

Parallel Sorting - the Zamfir Way.

D.C. van Leijenhorst

March 23, 1998

Department of Mathematics

and Computer Science

University of Nijmegen

The Netherlands

Abstract

A simple heuristic is given for Batcher's famous algorithm, which makes this parallel sorting method easy to understand for (practically) anybody, panflautists included.

Key notions: parallel algorithms, sorting, heuristics, education.

1 Introduction.

Parallel algorithms are fast and often ingenious and beautiful, but a lot of them appear difficult to understand and explain in an intuitive sense (as opposed to "mechanical" verification by induction). In this note, a heuristic is given for

a notorious example, namely Batcher's "bitonic" sorting algorithm. It will be seen that this algorithm can be explained to anyone but the dullest; almost on an elementary school level. Indeed, it is great fun to perform the algorithm with cardboard, scissors and paste in front of the lecture room.

An old problem in computer science is, how to sort quickly a large number of objects (numbers, books, archive cards,$\cdots$). Many efficient algorithms are known for "ordinary" (sequential) computers [Knu]. While performing such an algorithm, the computer imitates a single person able to compare and interchange any two objects. After a certain time (measured in comparisons and interchanges) the objects are ordered.

For example, on might try to move the largest element to the right, and sort the remaining object thereafter in the same way. This is called "bubblesort"; it may cost about n^2 time to sort n objects in this way. For arbitrary methods, it can be shown [AHU] that there exist rows of 2^k objects needing time $\geq C.k.2^k$ to sort, where C is a certain constant not depending on k. Algorithms reaching such a bound are known as well, e.g. the well-known heapsort method.

Things are different on a parallel computer. Such a machine (at least in the model considered here) can be compared with a large office containing a number of clerks performing simultaneously subtasks of a large algorithm. Some of the persons in the office are connected by telephone lines on which they may interchange information in a "clocked" way; say, every minute. Since many subtasks are done at the same time, problems in general can be solved much

faster on a parallel computer than on a classical one. For that reason, in all modern computers a lot of parallellism is built in.

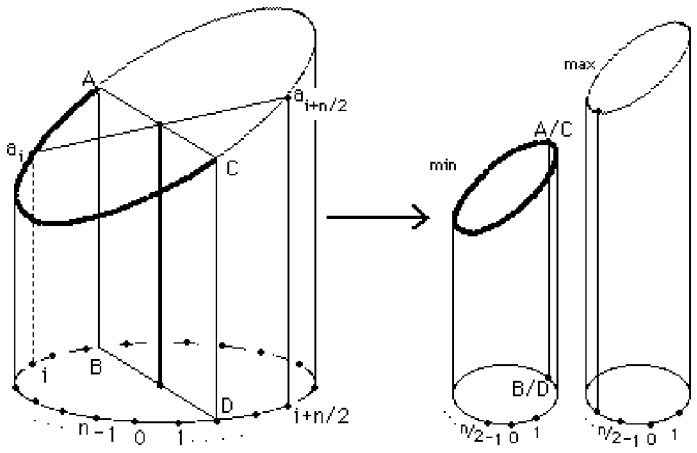
Of course, an office needs special work schedules ("algorithms", "programs") to split up a problem in subtasks for the clerks (who will also be called "processors" from now on). These algorithms are not found as easily as those for non-parallel machines, since the (conscious) human mind seems to prefer (or at least is used to) sequential thinking. Earliest attempts to write parallel programs stem from the fifties.

The sorting algorithm considered here is from 1968 and was invented by Batcher. The simple way it is told here is based on the presentation in Quinn's book [Qui]. The idea occurred to me while I was building a telescope from pieces of PVC tubing.

2 Bitonic merge by cutting tubes.

A sequence $a_0, a_1, ...a_n$ of objects is called bitonic whenever it increases and decreases cyclically, as illustrated in the left part of fig. 1.

(Fig. 1)



That's to say, a_i, $a_{i+1}, \cdots, a_j$ is a non-decreasing sequence; while a_{j+1}, $a_{j+2}, \cdots, a_{i-1}$ does not increase. Here, one is supposed to read a_0 for a_n; a_1 for a_{n+1} etc.; so the indices are taken "modulo n". This (mental) picture looks a bit like an obliquely sawn-off cardboard tube. Let us stick to this intuitive interpretation.

Now cut the tube lengthwise along AB and CD in such a way, that the "lower half" of the tube (fat line) and the "higher half" get separated. Each of the two pieces can be made into a thinner, obliquely sawed-off tube by the judicious application of some glue (right part of fig. 1). Put the smaller one left, the longer one right. Repeating this with the smaller tubes, one finds a sort of panflute consisting of four tubes of increasing lengths.

Proceeding in the same way, finally one obtains "tubes" of diameter one; that is, if n is a power 2^k of 2, which we shall suppose. For example, if $n = 1024 = 2^{10}$, this stage is reached after 10 steps (a step being the cutting and glueing of all tubes extant at that time). The result is a row of 1024 tubes of increasing

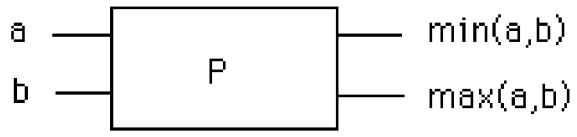
length: we have sorted $a_0, a_1, \cdots, a_{1023}$!

This special type of sorting, which works only for bitonic sequences, is also called "merging". This is because such a sequence can also be regarded as a pair of sequences, one increasing and the other decreasing, which as-it-were are mixed together.

3 How to do it in parallel.

Let us now consider how to do this in parallel in an "office". A processor (office clerk) considered here is rather stupid: he can do only one thing. Every now and then he gets two objects a and b (by telephone). All he does is: look if a is larger than b. If and only if this is so, a and b are interchanged. Thereupon they are sent on to another processor. Such a processor can be compared with the box of fig. 2 below.

(Fig. 2)



In the "tube algorithm" we need only $\frac{n}{2}$ office clerks: in step 1 the first clerk is given a_0 and $a_{\frac{n}{2}}$; in the second a_1 and $a_{\frac{n}{2}+1}$; $\cdots$; clerk $\frac{n}{2}-1$ is given $a_{\frac{n}{2}-1}$ and a_n. Each clerk determines $min(a_i, a_{i+\frac{n}{2}})$ (the i^{th} point on the left tube) and $max(a_i, a_{i+\frac{n}{2}})$ (the i^{th} point on the right tube) as can be seen clearly in fig. 1.

This is done simultaneously. Then, the first $\frac{n}{4}$ clerks proceed analogously with the left tube and the second half at the same time with the right one. As we saw before, ordering 2^k objects in this way takes time k only.

Remark. To prove rigourously that the sequences $min(a_i, a_{i+\frac{n}{2}})$, $0 \leq i \leq \frac{n}{2} - 1$ and $max(a_i, a_{i+\frac{n}{2}})$, $0 \leq i \leq \frac{n}{2} - 1$ are bitonic again, note, that this problem is invariant under various (bitonicity-preserving) transformations like index-shift ($a_i \rightarrow a_{i+1}$); index flip ($a_i \rightarrow a_{n-i}$), and order reversal on the underlying set of the a_i's. Also, the problem is trivial if the a_i's are constant.

Hence, w.l.o.g., we may assume that a_0 is a smallest element; that $a_0, a_1, \cdots, a_j$ is the non-decreasing part of the sequence; while $a_{j+1}, a_{j+2}, \cdots, a_{n-1}$ does not increase; and furthermore, that $j \geq \frac{n}{2}$. Then $min(a_i, a_{i+\frac{n}{2}}) = a_i$ and this increases for $i = 0, 1, \cdots$ until for some i, $min(a_i, a_{i+\frac{n}{2}}) = a_{i+\frac{n}{2}}$ which then decreases until $i = n$. So the "min"-sequence is bitonic. The same holds for $max(a_i, a_{i+\frac{n}{2}})$.

4 Arbitrary sequences.

The funny thing is that we are now able to order arbitrary, non-bitonic rows as well! For assume, that we are already able to sort an arbitrary row $a_0, \cdots, a_{\frac{n}{2}-1}$. This is certainly true if $n = 1 = 2^0$: our clerks just do nothing. Then if we are given any longer sequence $a_0, \cdots, a_{n-1}$, we split it into two equal parts, and sort the first half a_0 to $a_{\frac{n}{2}-1}$ and the second half $a_{\frac{n}{2}}$ to a_{n-1} simultaneously (as we are supposed to be able to).

If we write the first half in increasing order and append the second half written decreasingly $\cdots$ there results a bitonic sequence! But we know that, finally, that one can be ordered by the "tube algorithm". We are done.

Another way to formulate this is: Given any collection of $2m$ tubes, take pairs of them. For any pair, apply the "tube algorithm" to each tube and write the resulting ordered sequences increasingly and decreasingly, respectively. Join the two sequences to a larger tube. One obtains a collection m tubes, each twice as thick. Proceed in the same way until there is one tube left and order this tube by a final application of the tube algorithm. One starts with n small tubes corresponding to the single elements a_i.

5 Time and space.

What time does all this take? Obviously, ordering n objects takes the time to order $\frac{n}{2}$ objects plus the time of the tube algorithm, that is, k. Hence the total time is $1 + \cdots + k = \frac{1}{2}k(k+1) \leq k^2$. The number of clerks needed still is $\frac{n}{2} = 2^{k-1}$ (however, it may be more convenient to take an array consisting of new processors for each parallel step; a circuit size of order $n.k^2$).

References.

[AHU] A.V. Aho, J.E. Hopcroft and J.D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, 1974.

[Knu] D.E. Knuth, "The Art of Computer Programming", Vol. 3, Addison-Wesley.

[Qui] M.J. Quinn, "Designing Efficient Algorithms for Parallel Computers", McGraw-Hill, 1987, pp. 82 ff.