T U M

I N S T I T U T F Ü R I N F O R M A T I K

Performing Permuting on a Multiprocessor Architecture Using Packet Communication

Riko Jacob Michael Schnupp

TUM-I0817
Juni 08

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-06-I0817-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Performing Permuting on a Multiprocessor Architecture Using Packet Communication

Riko Jacob Michael Schnupp

Institut für Informatik, Technische Universität München
D-80290 München

Abstract

We extend matching upper and lower bounds for permuting from the I/O-model to a parallel model.

Contents

1 Introduction and Model

In [1] Aggarwal and Vitter showed matching upper and lower bounds for sorting and related problems in the I/O-model.

Here we use use those insights to extends the results to the parallel world. To this end we define a family of models of computation and show matching upper and lower bounds in them for the central problem of permutation.

In our model we have an arbitrary number of processors. Each of them has a memory of size M.

All data is stored only in the memory of the processors.

In each round the processors can do as much computation as they like. They then have to send their data in packages to (other) processors. Then all memory is cleared and the packages are received.

We are interested in the total number of packages we have to send and the number of rounds needed.

We furthermore generalize the model by introducing several parameters: A bound on the total number of processors P, a bound on the number of elements send in each packet and a bound on the number of packets D a processor can send in each round.

2 Permutation

Input: An input vector of size N stored on the first $\lceil N/M \rceil$ processors and an appropriate permutation matrix implicitly known to all processors.

Output: An output vector containing the same N elements in permuted order.

2.1 Algorithms

We present three algorithms, which are optimal under certain circumstances as show later.

	R	ℓ	P	$D \geq 2$	B
s-qs	$\log_s \frac{N}{M}$	$12s\frac{N}{M}\log_s \frac{N}{M}$	$\frac{N}{M}$	$8s$	M/s
depose	$\frac{M}{D}$	$N\frac{D-1}{D+1} \leq 3N$	$\frac{N}{M}$	D	1
split	$2\log_D M$	$4\frac{N}{D} + N \leq 2N$	N/D	D	M/D

Table 1: Overview over the three Algorithms. R denotes the number of rounds, P the number of used processors, D the degree, i.e. the number of packages a processor is able to send (or receive) at each round, and ℓ the number of sent packets.

2.1.1 s-Quick-Sort

This algorithm works by dividing the memory in s sections, placing the smallest N/s elements in the first section, etc., and recursively sorting the sections. After that, the elements are obviously sorted and the actual problem is therefore only the placing of the elements in the correct section in each round.

To do this placing, each processor first determines the correct segment for each of its elements, and sends to each section the appropriate elements.

Since all processors have to send to all sections, we have to specify which elements go to which part of the segment. Each processor has its number, and a processor with higher number puts its data after the data of the processors with lower number. This is possible, because all processors know the permutation and therefore know how much data is put to each section by all the lower-numbered processors.

Since the memory allocated to the sections is on different processors, we use the following rule: We try to attach the new data directly after the already present data. If the corresponding processor has already received $8s$ packets we move to the next processor. If the packet does not fit in the remaining space on the processor we split the packet, sending the left over data to the next processor.

Lemma 1. *The number of processors needed is at most* $3\frac{N}{M}$.

Proof. We use induction, showing that given that bound one one round it will also be true for the following round. At the beginning there are $\frac{N}{M}$ processors. So the lemma holds.

At a following round we need at most $1 + V + L$ processors, where V is the number of processors which had to be started, because the previous one was full, and L is the number of processors which had to be started, because the previous one already received $8s$ packets.

The value of V has to be at most $\frac{N}{M}$, because each time a processor is full, M elements were placed, and there are only N elements.

The value of L is at most $\frac{3}{2}\frac{N}{M}$, because there where only $3\frac{N}{M}$ processor and each of them send at most $4s$ packets. Since each target processor is able to receive $8s$ packets, this can only happen $\frac{3\frac{N}{M}4s}{8s} = \frac{3}{2}\frac{N}{M}$ times.

Hence the total number of processors in the next round is at most $1 + V + L \leq 1 + \frac{N}{M} + \frac{3}{2}\frac{N}{M} \leq 3\frac{N}{M}$, assuming $\frac{N}{M} \geq 2$. □

Because each processor sends only $4s$ packets, and the number of rounds is $\log_s \frac{N}{M}$, the total number of sent packages is at most $12s\frac{N}{M}\log_s \frac{N}{M}$ and we made sure that each processor sends and receives at most $8s$ packets.

2.1.2 Depose

This algorithm uses $\frac{N}{M}$ target processors additional to the $\frac{N}{M}$ source processors. In each round it then sends $D-1$ elements (in packets of size one) to the corresponding target processor and uses the last packet to remember the remaining data. To put all the elements it needs $\frac{M}{D-1}$ rounds. In each round $(D+1)\frac{N}{M}$ are send. Therefore, $N\frac{D+1}{D-1} \leq 3N$ packets will be sent.

2.1.3 Split

This algorithm needs many processors. Each processor splits its elements equally and arbitrarily to D new processors until each processor has at most D elements. Then each processor can either put its elements directly to the corresponding target processor, or, if the in-degree is also important, merge in the same way.

The split process takes, like the merge process, at most $\log_D M$ rounds. Sending the elements to the final (merge) processor needs another round, giving a total of $1 + 2\log_D M$ rounds. Because the split process stops if the number of elements reaches D, at most N/D processors will be used.

Because each processor is sent exactly one packet and the number of packets send during the split phase is equal to the sum of processors over the split rounds. At the end there are $\frac{N}{D}$ processors and in each round before the number decreases by a factor D. Therefore the number of packets send during the split phase is at most $\sum_{i=0}^{\infty} \frac{N}{D}\frac{1}{D}^i = 2\frac{N}{D}\frac{1}{D-1} \leq 2\frac{N}{D}$. The number of packets during the merge processes works exactly the same and sending the element to the final (merge) processor needs N packets in the worst case. Hence, at most $N + 4\frac{N}{D}$ packets will be sent.

2.2 Lower Bounds For Packets

2.2.1 No Restrictions

Lemma 2. *If there is a program computing a permutation using at least $2\frac{N}{M}$ processors, then there is also a program computing the same permutation using fewer than $2\frac{N}{M}$ processors, while sending no more packages. [This holds as long we have no limit on D.]*

Proof. If, at some point in the computation, at least $2\frac{N}{M}$ processors are used, then we group the first $2\frac{N}{M}$ processors in pairs.

If each of the $\frac{N}{M}$ pairs holds more than M elements, the total number of elements would be larger than N.

Since we have only N elements, there has to be a pair holding at most M elements and we can use a single processor to hold the whole data and simulate both processors.

We can do so until the number of used processors is smaller than $2\frac{N}{M}$. □

Lemma 3. *There is a permutation π so, that each program performing π has to send at least* $\frac{N \log \frac{N}{Me}}{2+\log \frac{N}{M}+M} \geq \frac{1}{3}\min\{N, \frac{N}{M}\log\frac{N}{Me}\}$ *packets.*

Proof. Since we need a different program for each permutation the number of programs we can specify has to be at least the number of different tasks (i.e. permutations) we need to compute.

In the following we count the information needed to specify a program.

Permuting N elements results in $N!$ different tasks, which can be expressed in $\log N! \geq N \log \frac{N}{e}$ bits.

For specifying a program, which is sending ℓ packets, we know we have to send each element, and need no copies. Therefore we can specify the packages for each processor, switching to the next processor when all data is sent.

For each packet we have to specify which elements are contained and to which processor they are sent. For specifying the elements we have to state for each of the up to M elements if it is contained or not. This can clearly be done using M bits.

By Lemma 2 we can assume that at most $2\frac{N}{M}$ processors will be used. Therefore, for specifying the target processor of a packet we have at most $2\frac{N}{M}$ targets and need at most $1 + \log \frac{N}{M}$ bits.

Since we care about the final order of the elements in memory, we additionally have to specify the order of the elements in final memory. Since there are $M!$ possibilities for each processor, we need at most $\frac{N}{M}\log M! \leq N \log M$ bits.

To be able to solve all possible tasks it must hold:

$N \log N - N \log e \leq \ell(M + 1 + \log \frac{N}{M}) + N \log M$ which is equivalent to

$\ell \geq \frac{N \log \frac{N}{Me}}{1+\log \frac{N}{M}+M} = \frac{N \log \frac{N}{Me}}{1+\log e+\log \frac{N}{Me}+M}$

Now we have different cases, depending which part of the denominator is largest:

1. $M \geq \log \frac{N}{Me}$ and $M \geq 1 + \log e$

$\frac{N \log \frac{N}{Me}}{1+\log e+\log \frac{N}{Me}+M} \geq \frac{N \log \frac{N}{Me}}{3M} = \frac{1}{3}\frac{N}{M}\log\frac{N}{Me} \geq \frac{1}{3}\min\{N, \frac{N}{M}\log\frac{N}{Me}\}$

2. $M < \log \frac{N}{Me}$ and $M \geq 1 + \log e$

$\frac{N \log \frac{N}{Me}}{1+\log e+\log \frac{N}{Me}+M} \geq \frac{N \log \frac{N}{Me}}{3 \log \frac{N}{Me}} = \frac{1}{3}N \geq \frac{1}{3}\min\{N, \frac{N}{M}\log\frac{N}{Me}\}$

3. $1 + \log e > M$

$M < 1 + \log e < 3$ this leaves only $M = 1$ and $M = 2$.

For a nontrivial permutation we need to send at least one package to each processor. Since there are $\frac{N}{M}$ processors, $\frac{N}{M}$ packages are needed. Since $M < 3$ we have:

$\frac{N}{M} > \frac{1}{3}N \geq \frac{1}{3}\min\{N, \frac{N}{M}\log\frac{N}{Me}\}$ □

The dispose and split algorithm when used with $D = M$ result in the same algorithm. This, and the s-qs algorithm will match our result.

2.2.2 Maximal packet size of B

Now we use an additional parameter B restricting the size of the packets.

Lemma 4. *A permutation algorithm which is only allowed to send packets up to size $B \leq \frac{M}{2e}$ has to send at least $\ell \geq \frac{1}{3}\min\{N, \frac{1}{2}\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{Me}\}$ packets.*

Proof. Lemma 2 still holds. The only difference to the above proof is, that the bits for description of the content of a packet changes. While with arbitrary packet size, we could specify the content with M bit. We have now smaller packet size and hence fewer possibilities. Since a packet can hold up to B elements, we have at most $\binom{M+B}{B}$ different possibilities, which will result in a bit size of

$\log\binom{M+B}{B} \leq B\log\frac{e(M+B)}{B} = B\log e + B\log(\frac{M}{B}+1) \leq B\log e + B\log(2\frac{M}{B}) \leq B\log(2e) + B\log\frac{M}{B} \leq 2B\log\frac{M}{B}$

This assumes $\frac{M}{B} \geq 2e$ and results in a lower bound of

$\ell \geq \frac{N\log\frac{N}{Me}}{(1+\log e)+\log\frac{N}{Me}+2B\log\frac{M}{B}}$

Again we have different cases, depending on which part of the denominator is largest:

1. $\log\frac{N}{Me}$ is largest:

$\ell \geq \frac{N\log\frac{N}{Me}}{3\log\frac{N}{Me}} = \frac{1}{3}N$

2. $2B\log\frac{M}{B}$ is largest:

$\ell \geq \frac{N\log\frac{N}{Me}}{6B\log\frac{M}{B}} = \frac{1}{6}\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{Me}$

3. $1 + \log e$ is largest:

Then $N < 15M$ and $M < 3B$. This cannot happen, since we before already assumed $\frac{M}{B} \geq 2e \geq 3$

Therefore we need always at least $\ell \geq \frac{1}{3} \min\{N, \frac{1}{2}\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{Me}\}$ packets. □

As before the direct algorithm (split/depose with $D = M$) matches the one case. The second case is matched by an s-qs with $s = \frac{M}{B}$.

2.2.3 Restricting the number of sent packets D

Now each processor can only send D packets in each round.

In this case the argument for Lemma 2 does not hold anymore. One processor simulating two processors has to send all the packets the two processors would have send, which could be more than D and therefore too much for a single processor.

Nevertheless, an algorithm that has a restriction on the number of send packets is still an algorithm, so the old bound is still valid.

If $D \geq 4$ we can use s-qs with $s = 2$ and the dispose algorithm, respectively, to match the bound.

With $D < 2$ no useful computation is possible.

For $D = 2$ or $D = 3$ the depose algorithm still works, but the qs has to be slightly modified to use depose like steps to send its data to the 4 target processors. This will only result in a constant factor, so it still matches.

2.3 Lower Bounds For Rounds

2.3.1 No Restrictions or only B

If no restrictions are involved it is always possible to use only a single round, in which each element is send in a packet of its own to the corresponding processor.

Because only single elements are send, a restriction on B does not change things.

This matches the split/depose algorithm for $D = M$, i.e. no restriction on D.

2.3.2 Restriction on D

Now we put a restriction on D. We generally cannot send to all target processors at once. The number of target processors is $\min M, \frac{N}{M}$ so we need $\log_D \min M, \frac{N}{M}$ rounds, to be able to send the data from one processor to all the target processors.

This matches the split algorithm.

2.3.3 Restricting D and P

Additionally restricting the number of available processors P results in a stronger bound.

For each round and each packet of each processor we have to specify the destination of that packet. We also have to specify for each element in each round in which of the D packets it is sent. As before all possible configurations have to be reachable.

$rPD \log P + rN \log D \geq N \log \frac{N}{Me}$

$r \geq \frac{N \log \frac{N}{Me}}{PD \log P + N \log D}$

Depending which summand of the denominator is largest, we have again two cases:

1. $\log_D P \geq \frac{N}{PD}$

$r \geq \frac{N}{PD} \log_P \frac{N}{Me} \geq \frac{N}{PD}$ (we assume $N > Me$ and we know $P \geq \frac{N}{M} > \frac{N}{Me}$)

If we assume $P = c\frac{N}{M}$ we get $r \geq \frac{M}{cD}$ which matches the bound of the dispose algorithm.

2. $\log_D P < \frac{N}{PD}$

$r \geq \log_D \frac{N}{Me}$

This matches the bound of the $D/2$-quicksort.

References

[1] Alok Aggarwal and S. Vitter Jeffrey. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.