High-level description of the algorithm

In the following sections, we discuss how the abstract version algorithm works and explain its requirements for implementation. The general idea is to have processes agree on the linearization and then respond fast from the history.

Tree specs

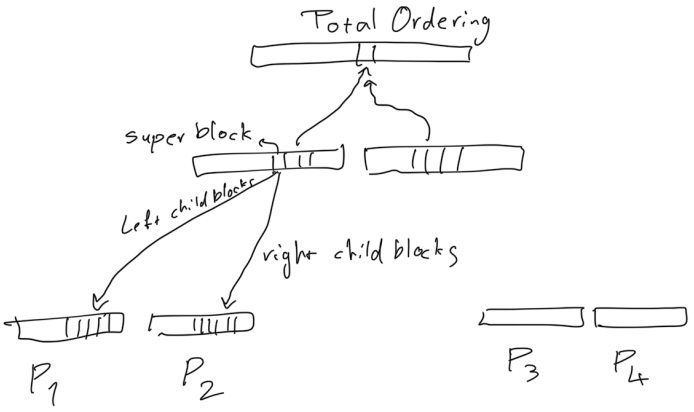

Figure 1: An example of a linearizable execution. Both could take effect first since ENQ(A), and ENQ(B) are concurrent.

A block is an ordering on some operations. Note that it does not necessarily contain the operations. Each node contains an ordering of blocks appended to it. Process p_i adds *op* to a block in its node and propagates it to the root, which stores the total ordering.

Propagate(node n):

1. Create a block from n's children's blocks which are not already in n
2. Append the block to n
3. If 2 was not successful, do 1-2 one more time

Requirements Creating a block takes less than $O(\log p)$ steps. If some processes try to append their block into node n simultaneously, at least one of them succeeds.

Linearization Operations are linearized when they are added to the root. The operations in a block are linearized as the block's ordering. Methods $Get(i)$ and $Index(op)$ give us information about the linearization.

Get(i): gets the ith operation in the total ordering

1. B= find the block containing ith in the root — $\log \#ops$ steps
2. Get(B, i-pre(B), root)

Get(B,i,n): gets the ith operation in block B in node n — pre(B) is #operations before B

1. SB= find the subblock of B containing ith op — $\log p$ steps
2. Get(SB, i-pre(SB), B's node)

Requirements Each level of Get() takes $\log p$ steps, and there are $\log p$ levels. So Get may take $O(\log \#ops + \log^2 p)$. In the following, we discuss that Get is going to be called on the current values of the Queue, so it is going to be $\log Q$.

Index(op): Where is the position of propagated operation op? It will be done by recursively calling Index(n,b,i) in the path from a leaf up to the root.

Index(n,B,i):

1. SB= find the block in n's parent that contains ith op in block B of node n — $\log p$ steps
2. Index(n,B,i+pre(SB))

Requirements find the superblock of a block in $\log p$ steps

Transform to a Queue

We are going to augment the root blocks to find out which index is the head of Queue while a Dequeue is linearized.

Enqueue(v): Add ENQ(v) to the tree

Dequeue():

1. Add DEQ() to the tree
2. Index the DEQ() order
3. Compute the index of the ENQ() which is the response of DEQ() might be null
4. Get(i) and return the argument

Requirements Augment root blocks so that 3 is fast. Get(i) takes $\log Q + \log^2 p$.