procedure $I$-update-ctb($i, S, L$)
  $h := \lceil \log L \rceil$;
  for $j := 1$ to $h$ do
    $I[j] := CBT[L \div 2^{h-j+1}]$;
  Build up array $D$ from $I$ without duplicates;
  for $j := 1$ to $|D|$ do
    $a := D[j]$;
    $e := \text{SELECT}(\text{Prio}[a] \cup S, n)$;
    $\text{Prio}[a] := \{ z \mid z \in (\text{Prio}[a] \cup S) \text{ and } z \leq e \}$;
    $S := \{ z \mid z \in (\text{Prio}[a] \cup S) \text{ and } z > e \}$;
  endfor
  $\text{Prio}[i] := S$;
end

Figure 3: Insert $b$-update-ctb operation for parallel PQs.

is the new item selected to be stored in $L$. During an $extract-min$ and after setting Leaf[$i$] = $L$ and
$CBT[L]$ = $i$, the Extract-min $b$-update-ctb($L$) operation executes the steps shown in figure 4. As a
consequence of SELECT, the update operations on the distributed array Prio take $O(h)$ time [3].
Two calls to MAX and one to SELECT are made in each update step. However, the calls to MAX
are made at the same point and thereby they may be reduced to one in terms of computation,
communication and synchronisation costs. The update operation takes $h$ steps to finish, which
leads to the above presented performance bound for BSP $extract-min$.

Note that the algorithms proposed in [3] are designed for a $d$-ary implicit heap. During an
$extract-min$ on this implicit heap, it is necessary to perform $d$ calls to SELECT for every node
located between the root and the target insertion leaf. On the other hand, a $d$-ary implementation
of the CBT requires a similar procedure. In this case it is necessary to determine the child with
less extended priority and compare it with the current winner (item $a$ in figure 4). However $d - 1$
calls to SELECT are needed in this $d$-ary CBT.

6 Final Comments

We have presented parallel priority queue algorithms on a tournament based complete binary tree.
Although we build up on ideas similar to previous solutions, in most cases the data structure we
use enables a more efficient implementation of the $extract-min$ and $insert$ operations. In particular
we reduce the amount of communication and synchronisation among processors by reducing the
number of calls to cost-dominating primitive parallel operations such as MERGE or SELECT.
From a theoretical point of view we only improve the asymptotic bounds of $extract-min$ and $insert$
on constant factors. However, these operations can be more than twice faster using much simpler
parallel algorithms upon the CBT than the usual data structure, the implicit heap.
procedure \textit{E-bsp-update-cbt}(L) 

\begin{align*}
  h &:= \lceil \log L \rceil; \\
  \text{for } j := h \text{ downto } 1 \text{ do} & \\
  a &:= 2\left(L \div 2^{h-j+1}\right); \\
  b &:= a + 1; \\
  x &:= \text{MAX}(\text{Prio}[\text{CBT}[a]]); \\
  y &:= \text{MAX}(\text{Prio}[\text{CBT}[b]]); \\
  \text{if } (x > y) \text{ then swap}(a, b); \\
  \text{CBT}[L \div 2^{h-j+1}] &:= a; \\
  \text{endfor} \\
\end{align*}

Figure 4: Extract-min \textit{bsp-update-cbt} operation for parallel PQs.

References


