Weighted Transaction Precedence Graph

5Q-3

OHMORI, Tadashi and TANAKA, Hidehiko
Dept. Electrical Engineering, The University of Tokyo

1 Introduction

We are studying the concurrency control of Bulk Access Transactions (BAT): the transactions to access bulk of data to the database-disks. We proposed a Weighted Transaction Precedence Graph (WTPG) and a cautious scheduler using it [Ohm88]. By enforcing the serializable order (SR-order) to make the shortest critical path in the graph, a schedule of less data/ resource contention is generated. This paper describes the $O(n^2)$ algorithm to find such the SR-order in a given chain WTPG.

2 The Algorithm

Fig.1 shows the chain WTPG $G(1,N)$. The 'chain' means that the graph without the edges $T0 \rightarrow Ti$ and $Ti \rightarrow Tf$ for all i is a set of chains. As the notations, $G(i,j)$ $(1 \leq i \leq j \leq N)$ is the subgraph of $G(1,N)$, composed of $n0$ and $\{n(i), n(i+1), .., n(j)\}$. The directed edge $n(i) \rightarrow n(j)$ says "$n(i)$ precedes $n(j)$ in the SR order". A pair of directed edges $(n(i), n(i+1))$ says "$n(i)$ conflicts with $n(i+1)$". The weights on the edges are the cost of the transactions to be executed. e.g. $a(k)$ on $n(k-1) \rightarrow n(k)$ is the cost of $n(k)$ to be executed after $n(k-1)$ commits and releases the conflicting locks.

Give a SR-order, then a WTPG becomes a directed acyclyc graph. Its critical path is the earliest possible completion time of the total schedule. We must find the SR-order to make the shortest critical path in it.

We say "the *choice edge* $(n(i), n(i+1))$ is set upwards" when it is resoluted to $n(i+1) \rightarrow n(i)$, and "set downwards" when it is resoluted to $n(i) \rightarrow n(i+1)$.

Definition 1 The parameter $L(k)$ and $R(k)$ are the triplets (curr, crit, rev) defined below:

In $G(k-1,N)$ where $(n(k-1), n(k))$ is set downwards, let $S1(k-1,N)$ be the SR-order of $\{ n(k-1), n(k), ..., nN \}$ to make the shortest critical path in $G(k-1,N)$. Then $L(k)$ is defined on the edge $n(k-1) \rightarrow n(k)$ as follows:

- $L(k).crit$: the length of the shortest critical path in $G(k-1,N)$ under $S1(k-1,N)$.
- $L(k).rev$: the smallest label among the label i such that $(n(i), n(i+1))$ is set upwards under $S1(k-1,N)$.
- $L(k).curr$: the length of the path $n0 \rightarrow n(k-1) \rightarrow n(k) \rightarrow .. \rightarrow n(L(k).rev)$.

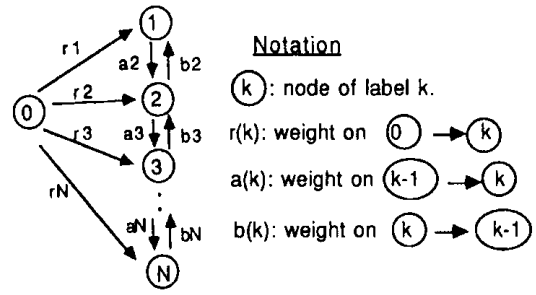

Figure 1: chain WTPG

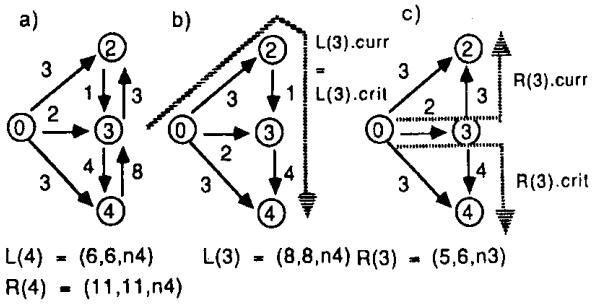

Figure 2: example of L(k) and R(k)

In $G(k-1,N)$ where $(n(k-1), n(k))$ is set upwards, let $S2(k-1,N)$ be the SR-order of $\{ n(k-1), n(k), ..., nN \}$ to make the shortest critical path in $G(k-1,N)$. Then $R(k)$ is defined on the edge $n(k) \rightarrow n(k-1)$ as follows:

- $R(k).crit$: the length of the shortest critical path in $G(k-1,N)$ under $S2(k-1,N)$.
- $R(k).rev$: the smallest label among the label i such that $(n(i), n(i+1))$ is set downwards under $S2(k-1,N)$.
- $R(k).curr$: the length of the critical path from $n0$ to $n(k-1)$ in $G(k-1, R(k).rev)$ under $S2(k-1,N)$.

□

$L(k)$ and $R(k)$ are notated by the triplet ($curr$, $crit$, $n(rev)$).

Example1: Fig.2-a illustrates the chain WTPG $G(2,4)$, $L(4)$, and $R(4)$. For computing $L(3)$, the edge $(n2, n3)$ is set downwards (in Fig.2-b) at first. Then $S1(2,4) = \{n2 \rightarrow n3 \rightarrow n4\}$ makes the shortest critical path of length 8. Hence $L(3)$ is (8,8,n4). For

```
main()
{    /* procedure L1(k) */
     temp = L(k+1).curr − r(k) + r(k-1) + a(k) ;
     if  ( temp =< L(k+1).crit )  then
           L1(k).crit = L(k+1).crit;
         L1(k).curr = temp;
         L1(k).rev  = L(k+1).rev;
     else {   /* temp > L(k+1).crit */
         L1(k).crit = min( max( V(h), R(h+1).crit ) );---(1)
                      h   s.t.  h = k+1 to L(k+1).rev.
   /*  V(h) = max( r(h),V(h-1)+a(h) );  ( k =< h )
       V(k-1) = r(k-1);
    */
         L1(k).rev = h0; s..t h=h0 takes the minimun in (1).
         L1(k).curr = the length of the path
                  n0→n(k-1)→ n(k)→n(k+1)-->n( L1(k).rev )
     endif /* end of L1(k) */
  /* procedure L2(k) */
     L2(k).curr = r(k-1) + a(k);
     L2(k).crit = max( L2(k).curr, R(k+1).crit );
     L2(k).rev = k;  /* end of L2(k) */

     L(k).crit = min( L1(k).crit, L2(k).crit );
     if  ( L1(k).crit =< L2(k).crit ) then
           L(k).curr = L1(k).curr;
         L(k).rev  = L1(k).rev;
     else
           L(k).curr = L2(k).curr;
         L(k).rev  = L2(k).rev;
     endif
}
```

Figure 3: the algorithm for $L(k)$

$R(3)$ in Fig.2-c, $S2(2,4)$ is $\{n3 \rightarrow n2, n3 \rightarrow n4\}$, not $\{n4 \rightarrow n3 \rightarrow n2\}$. Consequently $R(3)$ is (5,6,n3).[]

Theorem 1 Suppose that $L(i)$ and $R(i)$ for all $i = k+1$ to N are given in $G(k,N)$. Then the SR-order $S1(k,N)$, $S2(k,N)$ in Definition1 and the SR-order $S(k,N)$ to make the shortest critical path P in $G(k,N)$ are computed by the formulae:

$$S1(k,N) = \{k \rightarrow (k+1) \rightarrow .. \rightarrow L(k+1).rev\} \cup S2(L(k+1).rev, N)$$

$$S2(k,N) = \{k \leftarrow (k+1) \leftarrow .. \leftarrow R(k+1).rev\} \cup S1(R(k+1).rev, N)$$

such that $S1(k,k) = S2(k,k) = \phi$ for all k.
The length of $P = min(L(k+1).crit, R(k+1).crit)$.
$S(k,N) = S1(k,N)$; if $L(k+1).crit \leq R(k+1).crit$
$S(k,N) = S2(k,N)$; otherwise
[]

Example2: In the Example1, $L(3).crit = 8 > R(3).crit = 6$. Hence $S(2,4) = S2(2,4) = \{n2 \leftarrow n3\} \cup S1(3,4) = \{n2 \leftarrow n3 \rightarrow n4\} \cup S2(4,4) = \{n2 \leftarrow n3 \rightarrow n4\}$.

Theorem 2 Suppose that $G(k-1,N)$ and all the parameter $L(i)$ and $R(i)$ s.t. $i = k+1$ to N are given. Then the algorithm in Fig.3 computes $L(k)$ in $O(N-k)$. []

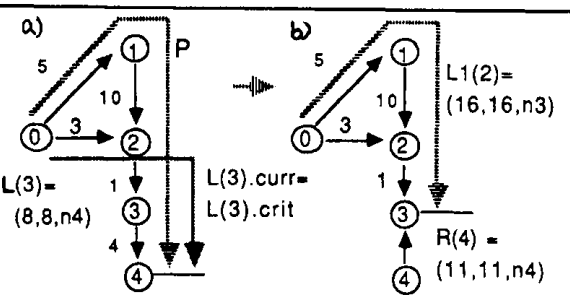

Figure 4: example of the algorithm for $L(k)$

outline of the proof: $L(k).crit$ is the minimum of those of $L1(k)$ and $L2(k)$. $L1(k)$ is the value of $L(k)$ when $(n(k), n(k+1))$ is set downwards. $L2(k)$ is that of $L(k)$ when it is set upwards. $L1(k)$ and $L2(k)$ must only consider the case where $n(k-1) \rightarrow n(k)$ is appended to $G(k,N)$ under $S1(k,N)$ and $S2(k,N)$ respectively. In $L1(k)$, if the new generated path $P0 = n0 \rightarrow n(k-1) \rightarrow n(k) \rightarrow n(k+1) \rightarrow ... \rightarrow n(L(k+1).rev)$ is longer than the critical path in $G(k,N)$, the new critical path $P0$ may get shortened by setting upwards an edge in $P0$. The expression (1) in Fig.3 computes this case. In the expression, $V(h)$ is the length of the critical path in $G(k-1,N)$ s.t. all its choice-edges are set downwards. The other cases are trivial. []

corollary 1 The SR-order of the shortest critical path in a given chain-WTPG is found in $O(n^2)$. s.t. n is the number of nodes in the WTPG.

proof: $R(k)$ is computed by $O(N-k)$ as $L(k)$ is. By computing $L(k)$ and $R(k)$ from $k = N$ to 2, $S(1,N)$ in theorem1 is computed by $O(n^2)$.[]

Example3: Fig.4 shows the case when the edges $n0 \rightarrow n1$ of the weight 5 and $n1 \rightarrow n2$ of the weight 10 are appended to the graph in Fig.2-a. Fig.4-a shows the behaviour of the algorithm to compute $L1(2)$. When adding $n1 \rightarrow n2$ to $G(2,4)$ under $S1(2,4)$, the new path P is $n0 \rightarrow n1 \rightarrow n2 \rightarrow n3 \rightarrow n4$ of length 20, greater than $L(3).crit = 8$. But P can be shortened by setting the edge (n3,n4) upwards as in Fig.4-b. In Fig.4-b, the critical path has the length $max(|n0 \rightarrow n1 \rightarrow n2 \rightarrow n3|, R(4).crit) = 16$. Hence L1(2) is (16,16,n3). []

3 Conclusion

This paper describes the $O(n^2)$ algorithm to find the SR order of the shortest critical path in the given chain form WTPG. In the simulation for the BAT processing, our scheduler outperforms the other locking protocols by 30% to 100% in throughput.

Reference: [Ohm88] IEICE, Tech.Rep., DE-88-19. July,'88.