Homework I

George Bosilca

The FIFO

The main process:
Adding items to the tail while removing items from the head

Problems:
1) Safety = “something bad never happens” (concurrent operations are successfully completed by only one thread)
2) Liveness = “something good eventually happens” (there will be progress)

The trivial solution based on mutexes will be accepted, but cannot lead to more than a B (a mutex is a memory flush operation and cannot give good performance).
The FIFO

The consistency is maintained by cooperative concurrency:

- when a process trying to enqueue a cell detects a pending enqueue operation (i.e. tail is not the last cell on the list), it first tries to complete the pending operation before enqueuing the cell
- The dequeue operation ensure that the tail pointer does not point to the dequeued cell and if necessary, tries to complete any pending operation.

---

The push operation

1) cl->next = ENDFIFO(ff)     # set the cell next pointer to end marker
2) loop                      # try until enqueue is done
3) icount = ff->icount       # read the tail modification count
4) tail = ff->tail           # read the tail cell
5) if CAS(&tail->next, ENDFIFO(ff), cl) # try to link the cell to the tail cell
   break;                     # enqueue is done, exit the loop
7) else # tail was not pointing to the last cell, try to set tail to the next cell
8) CAS2(&ff->tail, tail, icount, tail->next, icount+1)
9) endif                     # enqueue is done, try to set tail to the enqueued cell
The pop operation

1) loop

2) ocount = fl->ocount

3) icount = fl->icount

4) head = fl->head

5) next = head->next

6) if ocount == fl->oc

7) if head == fl->tail

8) if next == END FIFO(fl)

9) return NULL

10) endif

11) CAS2 (&ff->tail, head, icount, next, icount+1)

12) else if next != END FIFO(fl)

13) value = next->value

14) if CAS2 (&ff->head, head, ocount, next, ocount+1)

15) break

16) endif

17) endif

18) endloop

19) head->value = value

20) return head