Mi

eting in "General"

)6 06:28 UTC

	Organized by	Channel
e Kalpesh	Hemangee Kalpesh Kapoor	General

Global (Barrier) Event Synchronisation

- Centralised Software Barrier
- Shared counter
  - To maintain number of processes arrived at Barrier
  - Incremented by each arriving process
  - Increment must be mutually exclusive
- Process increments count
  - If count == p then it is last process
  - Else busy-wait till Barrier flag is ON
- The last process sets Barrier flag and releases all (p-1) waiting processes

Simple Centralised Barrier algo

```
struct bar_type {
   int counter;
   struct lock_type lock;
   int flag = 0;
} bar_name;
BARINIT (bar_name) {
   LOCKINIT(bar_name.lock);
   bar_name.counter = 0;
}
BARRIER (bar_name, P) {
   int my_count;
   LOCK (bar_name.lock);

   if (bar_name.counter == 0) {
      bar_name.flag = 0; /* first one */     }

   my_count = bar_name.counter++ ;

   UNLOCK (bar_name.lock);

   if (my_count == P) {    // last one to arrive
      bar_name.counter = 0;  //reset count,
      bar_name.flag = 1;  }  // set flag
   else {
       while (bar_name.flag == 0);    }} // busy wait
```

Centralised Barrier with Sense Reversal

- Problem in above barrier, if barrier operation is use consecutively using same bar variable. EX:

```
some computation
BARRIER(bar1, p);
Some more computation
BARRIER(bar1, p);
```

- Sample execution leading to the problem. Steps =

1. P1 || P2 || P3 || P4
2. BARRIER (1st)
3. (P1 last, so releases all. P1 makes counter=0; flag=1;) || (P2, P3, P4 wait for flag to be 1)
4. P2, P3 see flag=1 ..... [[P4 has not seen flag=1 as it may be in waiting queue, i.e. not scheduled by OS]]
5. therefore P1, P2, P3 do more computation and again wait for BARRIER
6. P1, P2, P3 has made flag=0 (Reset) but P4 waits for flag=1 (its old instance)
7. when P1, P2, P3 are done they wait for flag to be ‘1’ which will be done by the last processes (here P4)

=> BUT P4 is waiting for flag to be 1 (from earlier instance of P1)

Therefore BARRIER will never be resolved

Solution

- (1) Prevent processes to enter new instance of barrier until all have exited the previous of same barrier
- (2) Use another counter and do not reset flag in new instance until counter has turned to p
  - This new counter counts processes that leave barrier
  - But having more counters incurs latency and contention
  - Current setup requires to reset flag when count=p
- (3) Better solution: Do not reset flag
  - Make processes to wait for a new value of flag for every instance
  - e.g. wait for flag=1 then wait for flag to become ‘0’ then again ‘1’, etc.
  - Maximum we can have processes in two barriers:
    - One old pending and one new on-going
    - We need flag in 2 senses only: ‘0’ and ‘1’
    - Therefore called Toggle ==> called sense reversal