A report about producer/consumer pattern in parallel computing

Because of the end of Moore's law, the frequency of CPU could not double every 18 months as before, we changed our focus on the multicore processor.

Nowadays, most CPUs have more than one processor, and many computers, like sever, mainframe, datacenter, supercomputer, have more than one CPU.

This is the coming of parallel computing. In parallel computing, we have to solve many problems which we never faced before, such as synchronization.

A producer-consumer pattern is a problem in synchronization. The problem is asking how could we provide a queue to serve producer(writes data) and consumer(reads data) together, and satisfy the following conditions:

1, no lost or duplicated data;

2, the order the consumer reads the data is that the producer writes it;

3, prevent the producer from writing the data when the queue is full, the consumer from reading the data when it is empty.[1]

So it has three properties, mutual exclusion, starvation-freedom, producer-consumer.[2]

This problem was first introduced by Edsger W. Dijkstra. And then he came up with a multiple version, he also proposed the idea of semaphores to solve this problem.[3][4]

There are several ways to solve this problem, such as semaphores and monitors.

The following pseudocode implemented semaphores,

```
void *producer(void *arg) {
             int i;
            for (i = 0; i < loops; i++) {
            sem_wait(&empty); // Line P1
            sem_wait(&mutex); // Line P1.5 (MUTEX HERE)
            put(i); // Line P2
            sem_post(&mutex); // Line P2.5 (AND HERE)
            sem_post(&full); // Line P3
                        }
             }


             void *consumer(void *arg) {
```

```
int i;

for (i = 0; i < loops; i++) {

sem_wait(&full); // Line C1

sem_wait(&mutex); // Line C1.5 (MUTEX HERE)

int tmp = get(); // Line C2

sem_post(&mutex); // Line C2.5 (AND HERE)

sem_post(&empty); // Line C3

printf("%d\n", tmp);

        }

}
```

Current research hotspot

The recent paper showed that they focused on the speedup of the producer/consumer problem on DSM(Distributed Sharing Memory) systems. They have some communication ways from producers to consumers, which can be divided by consumer-initiated or producer-initiated. Consumer initiated has invalidate-based coherence and prefetch. Producer initiated has data forwarding, message passing, update-based coherence, cache-based locks, and selective update. They did some simulation experiments to test their best performance and compare their advantages and disadvantages.[6]

In paper[7], they proposed a new RMA(Remote Memory Access) paradigm, called Notified Access, which is beneficial for asynchronous small latency-limited messaging.

[1] Wikipedia, https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem#cite_note-ostep1-1
[2]Herlihy M, Shavit N, Luchangco V, et al. The art of multiprocessor programming[M]. Newnes, 2020.
[3] Dijkstra, E. W. "Cooperating Sequential Processes", unpublished manuscript EWD123 (1965)
[4] Dijkstra, E. W. "Information streams sharing a finite buffer." Information Processing Letters 1.5 (1972): 179-180.
[5]Arpaci-Dusseau R H, Arpaci-Dusseau A C. Operating systems: Three easy pieces[M]. Arpaci-Dusseau Books LLC, 2018.
[6]G. T. Byrd and M. J. Flynn, "Producer-consumer communication in distributed shared memory multiprocessors," in *Proceedings of the IEEE*, vol. 87, no. 3, pp. 456-466, March 1999, doi: 10.1109/5.747866.
[7]R. Belli and T. Hoefler, "Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization," *2015 IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 871-881, doi: 10.1109/IPDPS.2015.30.