EE 382C/361C: Multicore Computing **Fall 2016**

Lecture 19: November 1

Lecturer: Vijay Garg *Scribe: Yue Zhao*

19.1 Wait Free Consensus Hierachy

Consensus number of a shared object is the maximum number of processes that can use that object to solve consensus problem.

Operation: R/W Register: Consensus number 1;
TestAndSet: Consensus number 2;
GetAndIncrement: Consensus number 2;
Swap: Consensus number 2;
CAS: Consensus number infinity;

19.1.1 Read-Modify-Write Objects

An Read-Modify-Write Object has the following structure:

```
private int value;
public int synchronized getAndModify(int x) {
        int prev = value;
        value = f(value, x);
        return prev;
}
```

Function f sets a new value and returns the old value:

- Test&Set: f(value, x) = 1;
- Swap: f(value, x) = x;
- Get&Increment: f(value, x) = value+1;

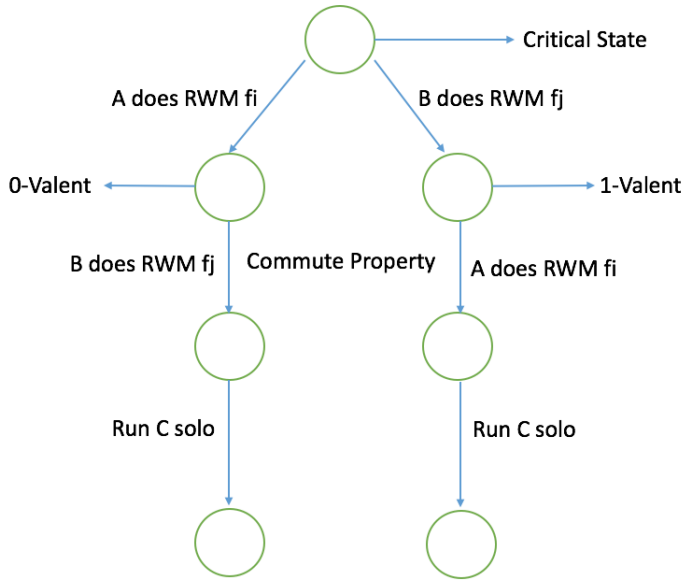
An read-and-modify object can be characterized by function f:
(1) RMW is commute if: fi fj = fj fi (for any i, j), where fi is the function applied by process i;
(2) RMW is overwrite if: fi(fj(x)) = fi(x) (for any i, j);

Theorem:
Any **non-trivial** (f is not identity, if f is idenity, f is read operation f(value, x) = value, which has consensus number 1) RMW object with either commuting or overwriting property has consensus number equals to two.

Proof(consensus number is at least 2):
Use non-triviality: Value init to null. P0 and P1 store their proposals to an int[2] array. There exists value

such that f(value)!=value. Then call RMW, if return initial value then choose my value; else choose the other value;

Proof(consensus number is at most 2):
There is no protocol that solves consensus for 3 processes using RMW object. Any protocol has to reach some critical state:



From critical state, A does RMW(fi) reach 0-valent then B does RMW(fj); for another path, B does RMW(fj) reach 1-valent then A does RMW(fj). Because RMW has commute property, these two path has same state, then C can run. For C, the initial state is same, so, C will always reach same consensus and it cannot decide on different things. However, for the first path, C should reach 0-valent and for the second path, C should reach 1-valent, which contradict with uni-valent property.

19.1.2 2-Write-1-Read-Object ((2, 1) Object)

2-Write-1-Read-Object: Write on two locations atomically, read any one location.
Protocol to solve consensus using (2, 1) object:

int[3] (initial with null);
Write proposal on int[3] array;
P0 writes (a, a) in location 0 and 1;
P1 writes (b, b) in location 1 and 2;
Whichever writes first wins.

Possible configurations during the process:
{a, a, null}, (P0 won at this point);
{a, b, b}, (P0 won);

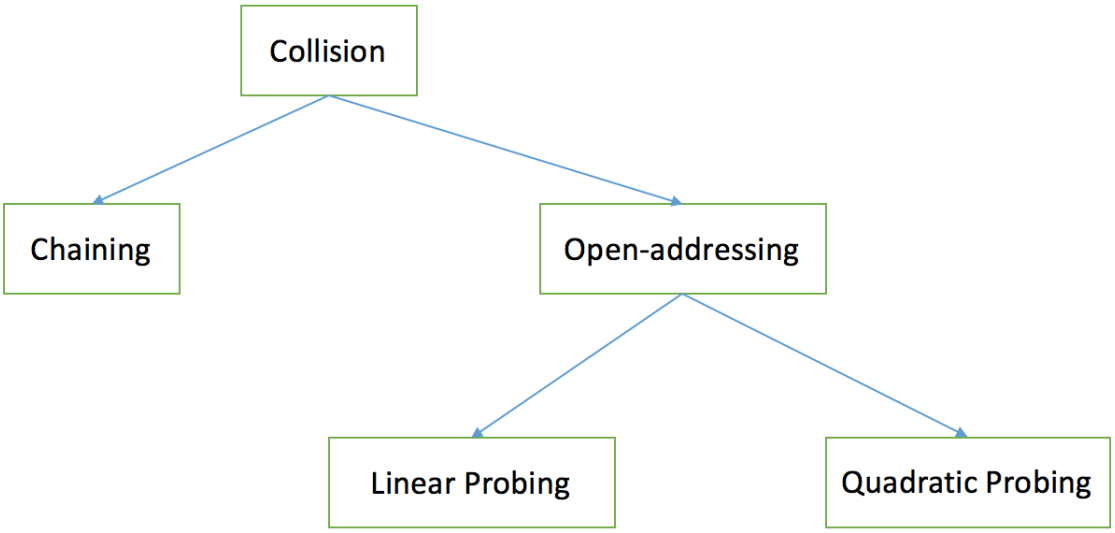
{null, b, b}, (P1 won);
{a, a, b} (P1 won);

This can solve consensus on 2 processes.

19.2 Universal Construction

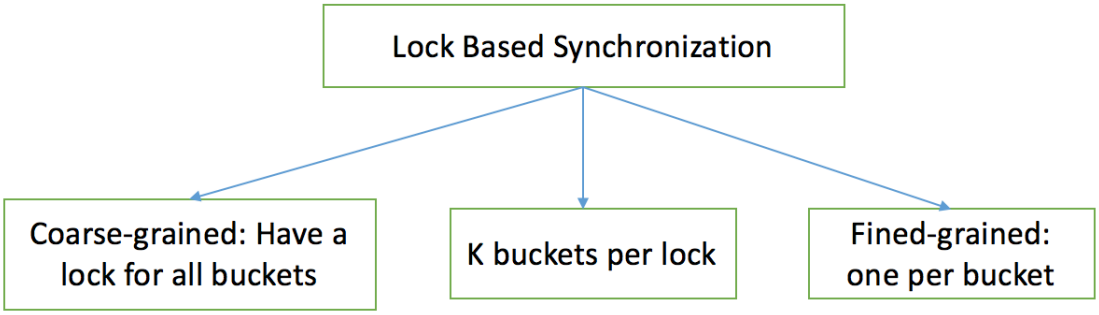
Given some objects of consensus number m, construct an object with consensus number less than or equal to M.
Deterministic Objects: state + operation cause a deterministic state.
Universal Construction only applies to deterministic state.
Correct Way: use a LinkedList, see code in github Initail state(Tail) ->apply method1 ->apply method2 ->apply method3(head)
Each time invoking a method, add a node in the linkedlist as the new head.

19.3 Hashing



Linear Probing: insert in the next empty slot. May cause clustering effect and damage performance.
Quadratic Probing: Try h(i) ->h(i)+1 ->h(i)+2^2 $->h(i)+3^2$ $->h(i)+4^2$

Parallel Chaining:


Lock Based Synchronization
Coarse-grained: Have a lock for all buckets
K buckets per lock
Fined-grained: one per bucket