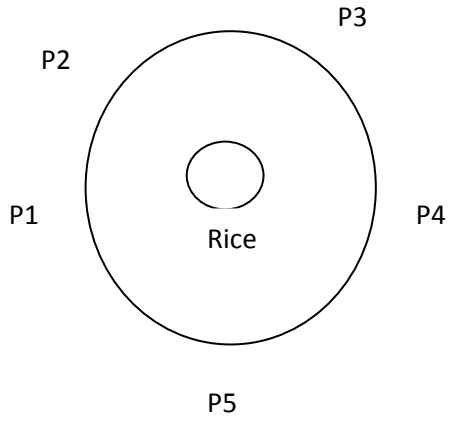
CS 5600 Scribe Notes (1st hour)
Ashish Musale
29 September 2010

Dining Philosophers Problem:



The dining philosopher's problem is a problem with five philosophers sitting at a table. The five philosophers sit at a circular table with a bowl of rice in the center. A fork is placed in between each pair of adjacent philosophers; each philosopher has one fork to his left and one fork to his right.
Each philosopher requires 2 forks to eat the rice. Each philosopher can only use the forks on his immediate left and immediate right.

Each philosopher can be one of the following states:

- ➔ THINKING
- ➔ EATING
- ➔ HUNGRY

Solution using semaphore:

```
semaphore chopstick[5];

philosopher(i){
        wait(chopstick[i]);
        wait(chopstick[i+1]);

        //EATING

        signal();
        signal();
}
```

Situation of Deadlock:
Each philosopher will pick up the chopstick on the left and waits for other to release and hence all the philosophers have one chopsticks.

TRANSACTION

Critical Section:

- Access of shared data.
- Series of read and writes that performs an operation.

We need to make sure that each critical section is executed critical section may not be in order.

Data Consistency
Data is consistent before entering and after exiting the critical section. No guarantee if the data is consistent in CS.

ATOMICITY:
It means that the result is either A or B where
A = everything happened
B = nothing happened
It looks like either the transaction was completely executed or not executed at all.

Transaction can either be:
COMMITED: The changes are done and saved.
ABORTED: This will happen when other transaction attempt to access same data either read or write. It could also happen because of failure of one of the system in a distributed system (e.g. transfer of money).Whenever a transaction is aborted, it needs to be rolled back. Changes done as part of the transaction that was aborted needs to be reverted to original values before transaction started. Hence, we need to keep track of the original data or values.

Types of Storage:

1) Volatile Storage: (RAM registers) Storage not maintained under storage crash.
2) Non Volatile Storage: (Disk, flash) Storage generally maintained under storage crash.
3) Stable: (RAID, Tape, Optical disk) Storage that never fails.

The data is saved on the non volatile storage while doing the transaction.
To perform a rollback, we need to save additional data about the data. It is called 'LOG'.

LOG data are stored on stable storage and this files are generally APPEND only.

Write Log Entry:

- Data location
- New value
- Old value
- Name of transaction

Started Log Entry:

- Name of transaction

Committed Log Entry:

- Name of transaction

Aborted Log Entry:

- Name

Note: Write the log entries before writing data to disk. So that we can avoid a situation where a data is written to the disk and then the system crashes. So, no log entry is made in that case which results in incorrect state.

For every transaction, there is one of three things that can happen:

1) NO FINISH
2) COMMITED - <commit Tx>
3) ABORT - <abort Tx>

How rollback is done when a transaction is aborted?
Scan the log files to revert back the changes of the aborted transaction

Scan 1: Identify the states of each of the transaction in the log file.
Scan 2: Reapply Writes
Those transactions that are committed are written back to disk.

Example log file:
<start T_A>
<write T_A=X old:0 new:5>
<start T_B >
<write T_B=X old:0 new:5>
<commit T_A>
<abort T_B>
<start T_C>
<write T_C=X old:5 new:8>

Assuming this is the complete log file, we need to create a stable state.
We perform the rollback operation:
Scan 1: We find the state of each of the operation.
A = committed
B = Aborted
C = No state
Scan 2: Reapply committed writes.
Hence, reapply writes by Transaction A.
Reject Transaction B and C as those are never committed and does not affect the system in anyway.

What if we don't want to start building again from scratch using the log files?
In that case we don't erase any data from the system, but we find a consistent state where we can say that the system was stable at this point. Such a point is called "CheckPoint"