CS244b - FaRM

Learning Goals:
- μsec-scale systems
- High-end hardware capabilities
  Kernel-bypass, RDMA, NV DRAM
- More practice reasoning about transaction serializability & recovery
Sender

(head, tail)

(Length, Message, Trailer)

Receiver
FARM API

- ACID transactions
  - On global shared mem:
    Begin, Commit, Alloc, Free
    Read, Write
- Lock-free single object reads
- Function shipping
Figure 3. FaRM architecture

Set of machines \( S, 2 \text{GiB} \) regions

\[ \langle C, S, F, CM \rangle \]
void T1()
{
    if (x2 == 0)
        x1 = 1;
}

void T2()
{
    if (x1 == 0)
        x2 = 1;
}
Figure 4. FaRM commit protocol with a coordinator C, primaries on $P_1$, $P_2$, $P_3$, and backups on $B_1$, $B_2$, $B_3$. $P_1$ and $P_2$ are read and written. $P_3$ is only read. We use dashed lines for RDMA reads, solid ones for RDMA writes, dotted ones for hardware acks, and rectangles for object data.
Figure 16. False positives with different lease managers
suspect $S_3$

$CM=S_1$

Zookeeper

Update $<9, ...>$ to $<10, \{S_1, S_2, S_4\}, F, CM=S_1>$

stop RDMA reads to $S_3$

REMAP

$LastDrained$

1. SUSPECT 2. PROBE 3. UPDATE CONFIGURATION 4. REMAP REGIONS 5. SEND NEW CONFIGURATION 6. APPLY NEW CONFIGURATION 7. COMMIT NEW CONFIGURATION

Figure 5. Reconfiguration

Last Primary Change $[r]$  Last Replica Change $[r]$
Figure 6. Transaction state recovery showing a coordinator $C$, primary $P$, and two backups $B_1$ and $B_2$. 
Vote:
- commit - primary
- commit - backup
- lock
- abort
- truncated
- unknown

When to commit?
- If you see one commit primary
- One commit - backup and no abort/unknown
Fig. 9 (TATP)

Fig. 10 (TPC-C)
**Figure 7.** TATP performance

**Figure 8.** TPC-C performance