Implementing caches

Doug Woos
Logistics notes

Textbook chapter for Friday—no (graded) discussion
\[
\text{put } (k1, f(data)) \\
\text{put } (\text{done1, true}) \\
\text{while(get(done1) == false) } \\
\text{put } (k2, g(\text{get(k1)})) \\
\text{put } (\text{done2, true}) \\
\text{while(get(done2) == false) } \\
rslt = h(\text{get(k1), get(k2)})
\]
Can we start done1 write before k1 write is complete?

Yes, provided we’re talking to a single server
Sharding

- Different data on different servers
- Partitioned via some function on keys
- Implement in Lab 4!

Another axis of scaling, cf. replication

- Usually, shard then replicate
- Each piece of data lives on one replicated shard
Can we start done1 write before k1 write is complete, if done1 and k1 live on different servers?

Not if we want sequential consistency!
Can we start done1 write before k1 write is complete, if done1 and k1 live on different servers?

Not if we want sequential consistency!

- Must serialize writes
What if there are caches? What might go wrong?
\begin{verbatim}
put (k1, f(data))
put (done1, true)
while(get(done1) == false) {
    put (k2, g(get(k1));
    put (done2, true)
}
while(get(done2) == false) {
    rslt = h(get(k1), get(k2))
}
\end{verbatim}

- Asia sees updated done1, cached k1
- Africa sees updated done2, cached k1 and k2
- Africa sees updated done2, k2, cached k1 (!)
Rule for caches and shards

Due to Sarita Adve

Suppose each process specifies ops in some order

Sequentially consistent if:

1. Ops applied in processor order, and
2. All ops to a single key are serialized (as if to a single copy)

So how do we ensure ops go to a single copy?
Invalidations vs. Leases

Invalidations
  - Track where data is cached
  - When doing a write, invalidate all (other) locations
  - Data can live in multiple caches for reading

Leases
  - Permission to serve data for some time period
  - Wait until lease expires before applying updates
  - Must account for clock skew!
Write-through vs. write-back

Write-through
- Writes go to the server
- No modified caches

Write-back
- Writes go to cache
- Dirty cache written to server when necessary
## Write-through vs. write-back

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Write policy</th>
<th>Invalidation</th>
<th>Leases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write-through</td>
<td>Write-through</td>
<td>AFS (Andrew FS)</td>
<td>DNS</td>
</tr>
<tr>
<td>Write-back</td>
<td>Write-back</td>
<td>Sprite</td>
<td>NFS</td>
</tr>
</tbody>
</table>
Write-through invalidations

Track all reading caches

On a write:

- Send invalidations to all caches
- Each cache invalidates, responds
- Wait for all invalidations, do update
- Return

Reads can proceed:

- If there is a cached copy
- If no write waiting at server
Client
put (k1, f(data))
put (done1, true)
while(get(done1) == false);
put (k2, g(get(k1));
put (done2, true)
Client
while(get(done2) == false);
Client
rslt = h(get(k1), get(k2))
\begin{center}
\begin{tikzpicture}
\node[draw,rectangle,fill=white] at (0,0) {Client};
\node[draw,rectangle,fill=white] at (5,0) {Client};
\node[draw,rectangle,fill=white] at (10,0) {Client};
\draw[->] (0,0) -- (5,0);
\draw[->] (5,0) -- (10,0);
\draw[->] (0,0) -- (10,0);
\end{tikzpicture}
\end{center}

\texttt{put (k1, f(data))}
\texttt{put (done1, true)}
\texttt{while(get(done1) == false)}
\texttt{\quad ;}
\texttt{put (k2, g(get(k1));}
\texttt{put (done2, true)}
\texttt{while(get(done2) == false)}
\texttt{\quad ;}
\texttt{rslt = h(get(k1), get(k2))}
\texttt{done1=false}
Client

```
put (k1, f(data))
put (done1, true)
while(get(done1) == false)
    put (k2, g(get(k1));
    put (done2, true)
while(get(done2) == false)
    rslt = h(get(k1), get(k2))

done1=false
done2=false
```
put (k1, f(data))
put (done1, true)
while(get(done1) == false)
    ;
put (k2, g(get(k1)));
put (done2, true)
while(get(done2) == false)
    ;
rslt = h(get(k1), get(k2))
done1=false
done2=false
put (k1, f(data))
put (done1, true)
while(get(done1) == false)
    put (k2, g(get(k1)));
    put (done2, true)
endwhile
rslt = h(get(k1), get(k2))

done1=false
done2=false
Client

put (k1, f(data))
put (done1, true)

while (get(done1) == false)
    ;
put (k2, g(get(k1)));
put (done2, true)

rslt = h(get(k1), get(k2))

done1 = false

Client

Client

Client
Client

put (k1, f(data))
put (done1, true)

while (get(done1) == false)
    put (k2, g(get(k1)))
    put (done2, true)

Client

rslt = h(get(k1), get(k2))

Client

while (get(done2) == false)
    done2 = false

Client

done1 = true
Questions

While a write is waiting on invalidations, can clients read old values from caches?
Questions

While a write is waiting on invalidations, can the writing client perform a different write?
Questions

While a write is waiting on invalidations, can the server process a read to a different location?
Questions

While a write is waiting on invalidations, can the server process a read to the same location?
Questions

While a write is waiting on invalidations, can the server process a write to a different location?
While a write is waiting on invalidations, can the server process a write to the same location?
More Questions

Why does the server wait until write is applied before returning to the client?

Why queue incoming requests during a write?

How much directory state is needed at server?