RPCs & Linearizability

RPCs

- Why?

- The most common way that processes in implemented distributed systems interact

  b) Convenient way to define

  a) The structure of messages
      - CONTENTS
      - ENCODING

  b) Hide Networking Details

- Some of the papers we read describe protocols in terms of RPC calls & responses

Note: The model in the paper is what we talk about closely corresponds to what gRPC, Thrift, etc. provide
on recv (echo, m)
  from s :
    send s, m

send a, (echo, "hello")
recv "hello" from a

msg' echo (msg m)
  return m;

net = a & echo ("hello")
assert (net == "hello")

SEEMS SIMPLE ENOUGH, BUT THERE ARE DIFFERENCES

① RE-ENTRANCY/ATOMICITY

on recv n from y:
  a = at n
  b = 2 * a - 1
  send (z, (up-a, a))
  send (z, (up-b, b))
  send (y, a+b)

Most of the protocols we encounter that are expressed
in messages
- Assume 1 message processed at
RPC: Making a call ⇒

Caller must wait (block) for response

Runtime (usually) can process other calls while blocked.

Handling other RPC calls was mandated by Binnell & Nelson

\[
\begin{align*}
\text{int } & \pi A(\text{int}) \equiv \\
\cdots & \equiv Y \cdot \pi B(\cdots) \\
\text{int } & \equiv z = Y \cdot \pi B(\cdots); \\
\cdots & \\
\end{align*}
\]

\[\text{int } \pi C(\cdots) \equiv \]

\[
\begin{align*}
\text{int } & \equiv h = X \cdot \pi C(\cdots); \\
\cdots & \\
\end{align*}
\]

\[\]

Allowing concurrent RPC calls helps with
Do not need to consider implementation of function being called.

2. EFFECT OF REMOTE FAILURES

Y crashes before returning to X

Message model: X never receives from Y
- on recv... logic never executed

RPCs: Implementation dependent but
- call will timeout
fail

⇒ Return value indicates failure (of call not process)

⇒ Exception thrown

⇒ ...

Will show up in papers as
RPC request fails OR
No response is received

**LINEARIZABILITY**

Motivation/Goal:

One reason to build distributed systems is to tolerate faults

PROGRAM ⇒ Dist. Program
Q: How does program behavior differ to dist. program behavior?

Differences
- Concurrency
- Failures

Linearizability: Consistency model

Given ADT: Queue
Stack
...

How specified: Sequential specification

Queue: FIFO container

\[
\begin{align*}
\text{enq}(5) & \quad \text{enq}(6) \\
\text{deq}() & \rightarrow 5 & \text{enq}(7) & \quad \text{enq}(8) \\
\text{deq}() & \rightarrow 6 & \text{deq}() & \rightarrow 7
\end{align*}
\]
Stack \[\text{LIFO}\]

\[
\begin{align*}
\text{push}(1) \\
\text{push}(2) \\
\text{pop}(1) \\
\ldots
\end{align*}
\]

Q: How does a concurrent queue or stack behave?
Many possibilities

(a) Eve's observed order
(b) Alice - Beth - Alice - Beth - ...
(c) LINEARIZABILITY
(d) SEQ CST
(e) CAUSAL CONSISTENCY

Defining Linearizability

1. \(\text{Alice Enq}(5) \rightarrow \langle \text{Alice Inv Enq}(5) \rangle\)
   \(\langle \text{Alice Res} OK \rangle\)

2. History - Sequence of inv, res interactions

Go from whose perspective?
History partial order

\( c_1 <_H c_2 \Rightarrow \text{Re} \downarrow c_1 < \text{Inv} c_2 \)

\[ A \text{ Enq (5)} \]

\[ B \text{ Enq (7)} \]

\[ A \text{ Enq (6)} \]

\[ B \text{ Deq (7)} \]

Some other useful definitions

\( H \) equivalent to \( H' \)

\[ \Rightarrow H \upharpoonright \rho = H' \upharpoonright \rho \text{ \( \uparrow \) } \rho \in \text{IT} \]

Complete \( (H) \) \( \Rightarrow \) maximal subsequence where every request has a response.

\( H \) sequential \( \Rightarrow \) first event is invocation \( \Downarrow \)
No concurrent requests ($\leq_H$ is total order)

Assume $H[p]$ sequential $\forall p \in P$

Defn

History $H$ is linearizable if

$H$ can be Extended to $H'$ s.t.

1. $\exists$ Sequential history $S$, where

   $S$ is equivalent to $\text{complete}(H')$

2. $\langle \leq_H \rangle \subseteq \langle \leq_S \rangle \bigcup \langle \circ S \rangle$

   Valid for ADT Seq. Spec

Note, 2 is wrong: Sela, Herlihy, Petrman

PODC '21 [Reading for next week]  

Should be

$\langle \text{complete}(H') \rangle \subseteq \langle \leq_S \rangle$
Properties

1. Locality: \( a_1, a_2, \ldots, a_n \) on objects

   Linearizable \( H \iff \)

   \[ H(a_1, H(a_2, \ldots, H(a_n) \ldots)) \]

   is linearizable.

2. Non-Blocking:

   \( S \leftarrow H \) linearizable, contains invocation \( \langle A \text{ Inv } \text{ Deq}() \rangle \)

   \[ \exists \text{ response } R \text{ s.t. } \]

   \[ H \circ \langle A \text{ Inv } \text{ Deq}() \rangle \langle A \text{ Res } R \rangle \]

   is linearizable.

   Instructive to see examples where this is not true:

   \( I_0 \cdot \text{deq}(C) \) \( R \text{OK}(1) \) \( I_0 \cdot \text{eng}(2) \) \( R \text{OK} \)
If

\[ \text{deg}\, \alpha \]

\[ \text{deg}\, \beta \]

\[ \text{deg}\, \gamma \]

\[ \text{deg}\, \delta \]

\[ \text{deg}\, \epsilon \]

\[ \text{deg}\, \zeta \]
Complete($H'$) = $t_1'$

$S | A = G | A$

$S | B = H' | B$

$O_2 \text{deg. } ORC2$  $O_1 \text{En}(x)$  $O_1$

$O_4$

$O_1 \leq O_2$

$O_3 \leq O_4$

$O_2 \leq O_3$

$O_4 \leq O_1$

$I\text{Enq}(x)$  $R\text{OFC}$  $I\text{Deg}()$  $R\text{OFC}(y)$

A

B

$I\text{Enq}(y)$
\[ H_{\text{ext}} = H \cdot \langle R, \text{OK}(C) \rangle \]

\[ \langle H_{\text{ext}}, \text{OP}_1 \rangle \langle H_{\text{ext}}, \text{OP}_2 \rangle \]

\[ H \xrightarrow{\text{buggy-line}} H' \]

Complete \((H') = H'\)

\[ S = \langle \text{OP}_3, \text{OP}_1, \text{OP}_2 \rangle \]
$\langle H_{ext} = \{ (op_1, op_2), (op_1, op_3) \}$

$S$