

Mirror

two-way real-time syncing

Goals of presentation

- Demo mirror
- Discuss implementation choices and evolution
- Walk through how it leverages GRPC
- General things that worked well

Demo Mirror

The problem

1. Desktop has CPUs + RAM + build system
2. Laptop has my IDE + windowing system

1st Step: Don't build it yourself

- Check for tools in this problem space
 - NFS (sshfs, etc.): too slow, no inotify
 - rsync: too slow, not real-time
 - Remote desktop: too laggy
 - git: Commit + push for every file save is too slow
 - FTP: one-way
- If no existing tools, why not?

Approach sanity check

- Files change
- Check if needs sent
- Send file changes to other machine

Files Change: JDK7 Watch Service

- Push notifications
- Cross platform
 - inotify on Linux, etc.

```
@Test
    public void test() throws IOException, InterruptedException {
//step1
        final WatchService service =
            monitorCreationIn(directory(), withWatchService());

//step2
        final String location = newFileIn(directory());

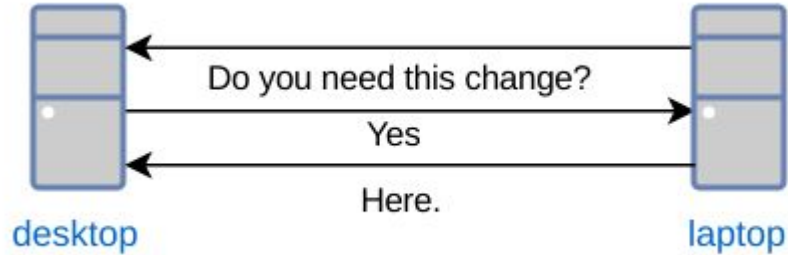
//step3
        final WatchKey modification = service.take();
        assertEquals(directory(), modification.watchable());
        for (final WatchEvent event : modification.pollEvents()) {
            if (OVERFLOW.equals(event.kind())) continue;
            assertEquals(ENTRY_CREATE, event.kind());
            assertEquals(1, event.count());
            assertEquals(location, ((Path) event.context()).endsWith(location));
        }
        modification.reset();
    }
}
```

Approach sanity check

- Get file notifications from OS
- Check if needs sent
- Send file changes to other machine

Send File Approaches

1. Master/slave

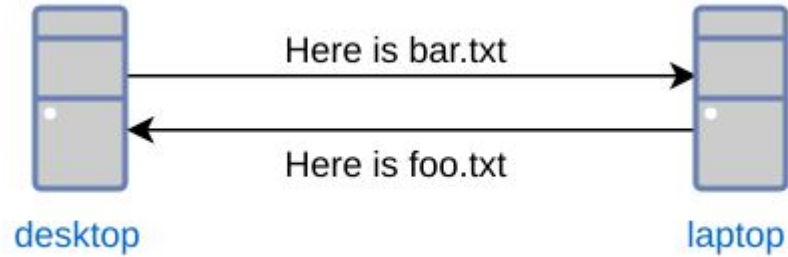


State is the simplest; all in the desktop.

Lots of chatter from laptop asking what to do.

Send File Approaches

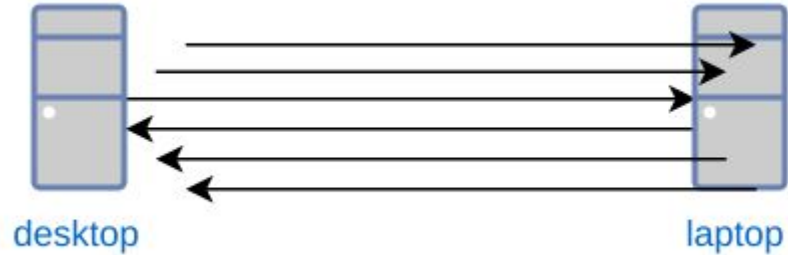
2. Peer-based



Both sides know what each other has.
Like nodes in a cluster.

Send File Approaches

2b. Two-way streaming



Approach sanity check

- Get file notifications from OS
- Check if needs sent
- Push file changes to other machine

GRPC

- RPC framework
- Based on protobuf
- Rewrite of internal Google library

The main usage scenarios:

- Efficiently connecting polyglot services in microservices style architecture
- Connecting mobile devices, browser clients to backend services
- Generating efficient client libraries


Core Features that make it awesome:

- Idiomatic client libraries in 10 languages
- Highly efficient on wire and with a simple service definition framework
- Bi-directional streaming with http/2 based transport
- Pluggable auth, tracing, load balancing and health checking



Basic Service Definition

```
8 service Mirror {
9   rpc InitialSync(InitialSyncRequest) returns (InitialSyncResponse) {}
10
11  rpc StreamUpdates(stream Update) returns (stream Update) {}
12
13  rpc Ping(PingRequest) returns (PingResponse) {}
14 }
15
16 // Message for the client's initial state.
17 message InitialSyncRequest {
18   string remotePath = 1;
19   repeated string includes = 3;
20   repeated string excludes = 4;
21   repeated Update state = 2;
22 }
--
```



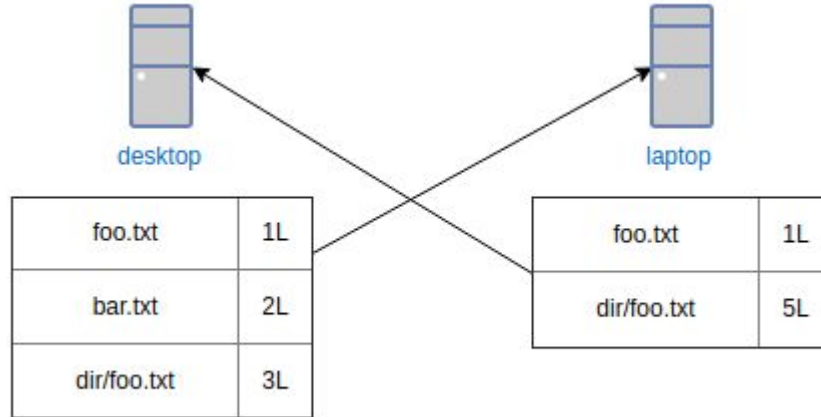
Approach sanity check

Mini design-doc:

- Get file notifications from OS
 - Java's Watch Service
- Check if new/ignored file
 - Some logic we write
- Push file changes to other machine
 - GRPC

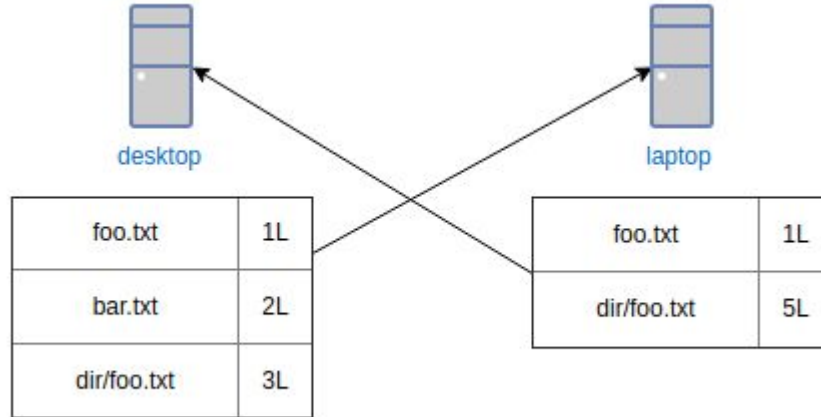
Checking files: 1st approach

- Map of path to mod time
- Exchange maps



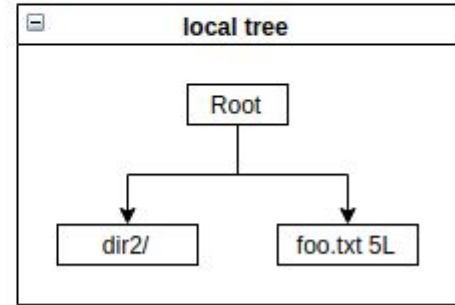
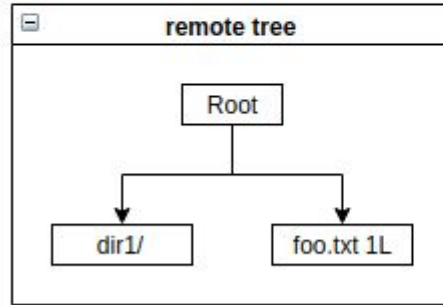
Checking files: 1st approach

- Map of path to mod time
- Exchange maps
- Pro: very simple
- Con: no notion of tree, have to check all paths



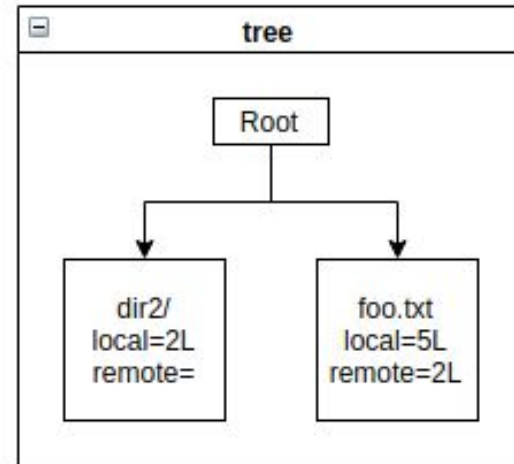
Checking files: 2nd approach

- Exchange trees, then do tree traversal for the diffs

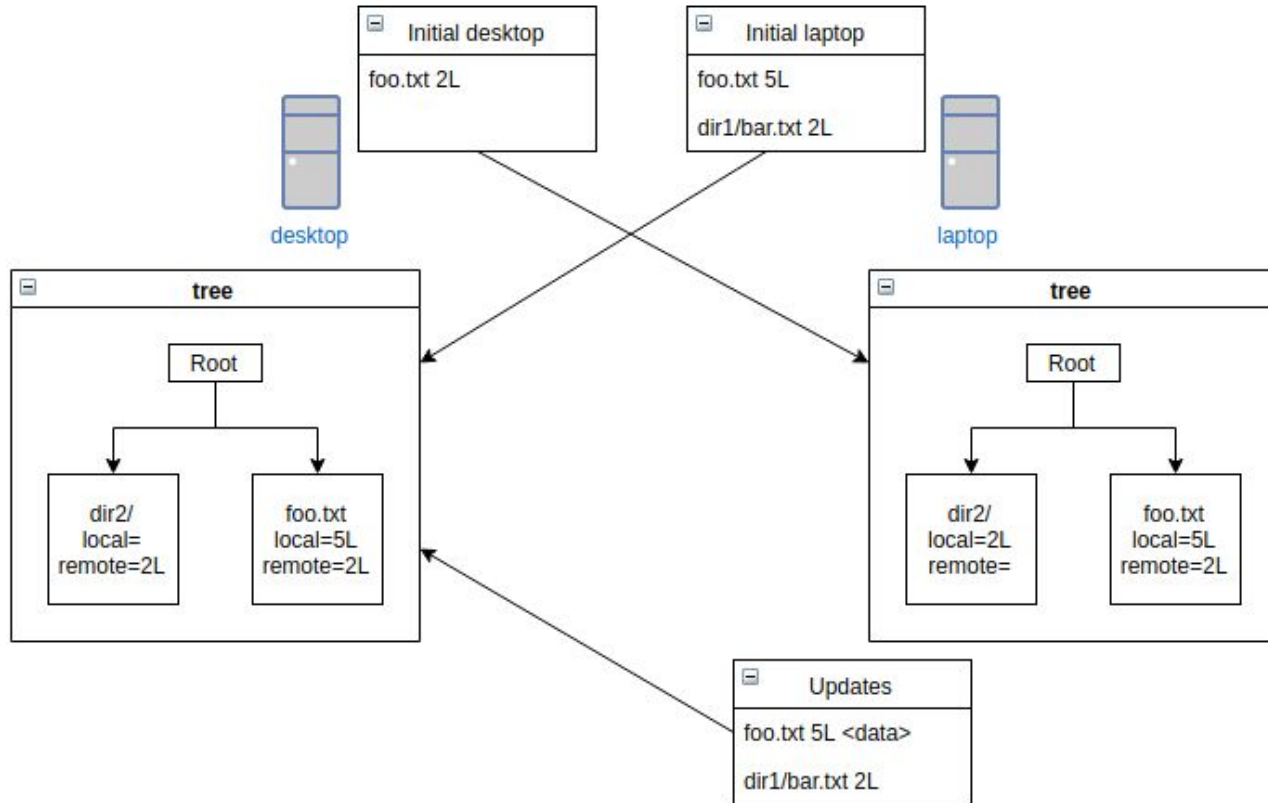


Checking files: 3rd approach

- Combined tree, where each node has a local value + remote value

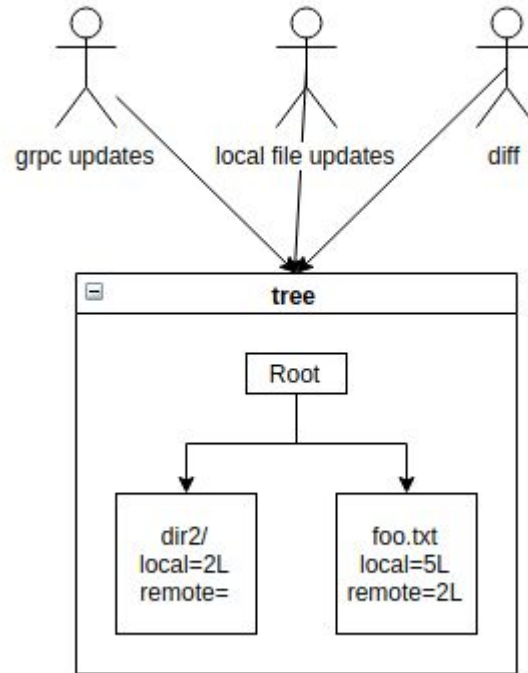


Checking files: Overall flow

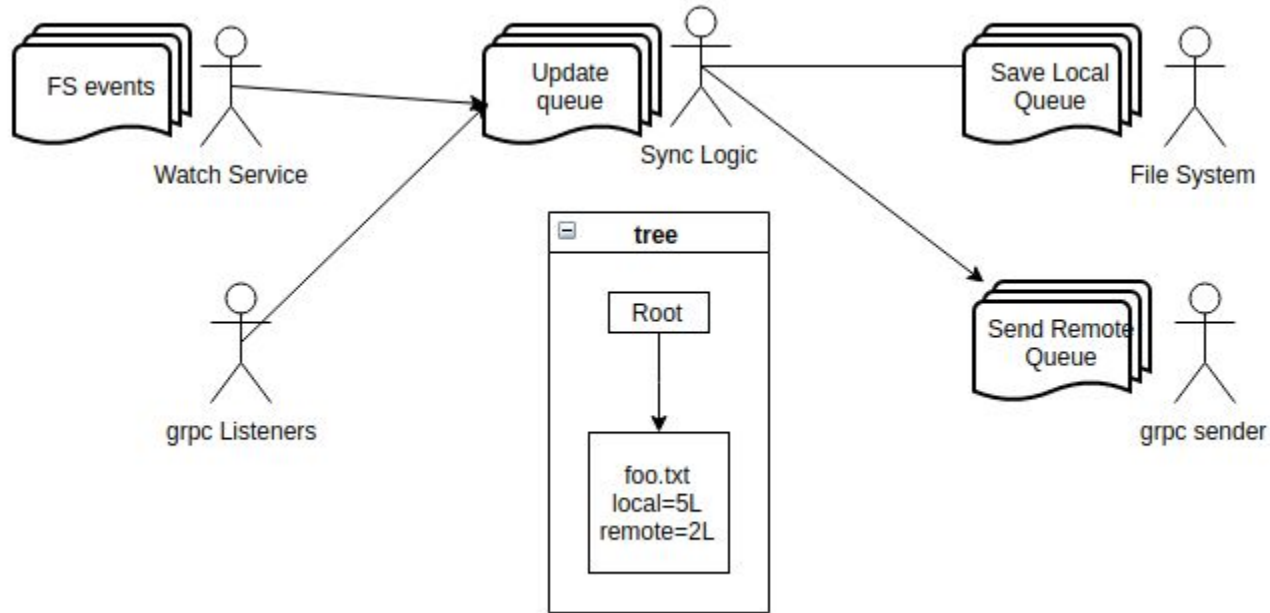


Threading: Naive approach

- Shared tree
- Have to lock/synchronize readers and writers



Threading: Actor approach



Approach sanity check

- Get file notifications from OS
- Check if new/ignored file
- Push file changes to other machine

Takeaways/Recommendations

- Have a design in your head
- Know what tools you can leverage to make your work easy
 - grpc-java
 - Jgit
 - Watch Service (or watchman)
- Start simple
 - List of files, add tests
 - Then two trees
 - Then one tree
- For tools, have both unit + integration tests
- Judicious use of marker objects