Implementation Plan

VERY stale, ignore.

For immediate mirroring, the implementation plan is roughly this.

We will start with a rough prototype of basic functionality on an integration branch.
The prototype provides the following:
- Ability to create immediate mirror files
- Active Writer locking from client to server
- File level replication layout transitions on last active writer (ie, go from write_pending to immediate)
- IO duplication for direct IO writes

The prototype does not have (among other things):
- Most error handling
- IO duplication for buffered IO
- IO duplication for setattr, etc
- Handling for a mirror being unavailable on write ("fast fail" on write mirror)
- Client -> server error communication
- Server side recovery management (eviction, restart of MDT)

The idea is the prototype should allow developers to work on individual components.
eg, if someone is working on improving FLR state transitions on the server, the client direct IO splitting implementation is enough to allow testing it
or, if someone is working on IO duplication for buffered IO, they can rely on the simple server implementation to allow testing it

Tasks will be a mix of adding new functionality and making the prototype more robust, but most tasks should be able to proceed in parallel.

The integration branch will not pass all tests immediately, but it will have a simple set of integration tests which developers can run to verify immediate mirror functionality. As the prototype firms up, we will gradually resolve all issues with existing tests and continue adding new tests for specific functionality.

Developers will rebase regularly on the integration branch to ensure they're working with the latest code.

Task Breakdown
At a high level, I am breaking tasks down in to the following categories:

Groundwork (eg, creating the layout flag)
Integration/Testing
Client/CLIO
LDLM/error handling
MDT/recovery

And we will tag tasks accordingly.