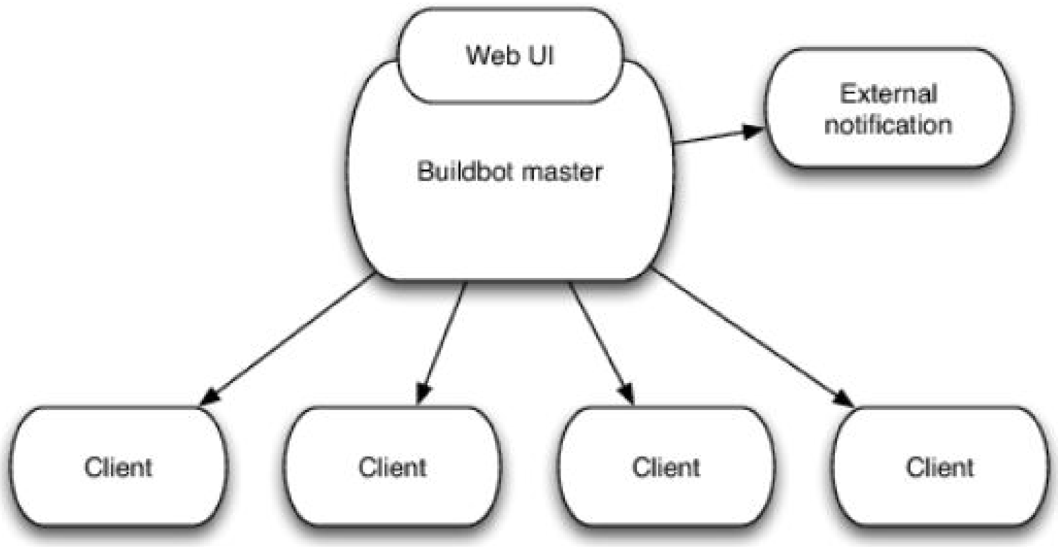
Architecture

Often the architecture of the CI server is modelled with two distinguishable types of servers existing:

1. Master
2. Slave

The master server is responsible to run different code units and tests on the slave servers. One master server can operate multiple slave servers. The differences in the architectures adds on the “Master and slave” design of other types of servers to abstract the functionality, or even mutate the means by which the orders and results are sent for better operations.

Biuldbot



Using the master-slave design, one server is sending orders to multiple clients to run codes and get test results. In this design clients are fully connected to the master. That basically implies that the master is in full control over the clients configuration settings, the connection should never be disrupted. This fact created some disadvantages.

1. Security defect: As the master is in full control and always connected to the submaster or the clients, clients could be affected if the master has been attacked.
2. Any error or warning messages in the OS of the client is going to be ignored, the fact that the master has no awareness of any messages popping up on the clients. The master cares only to be connected, so no overloads or errors are reported back to the master.

CDash

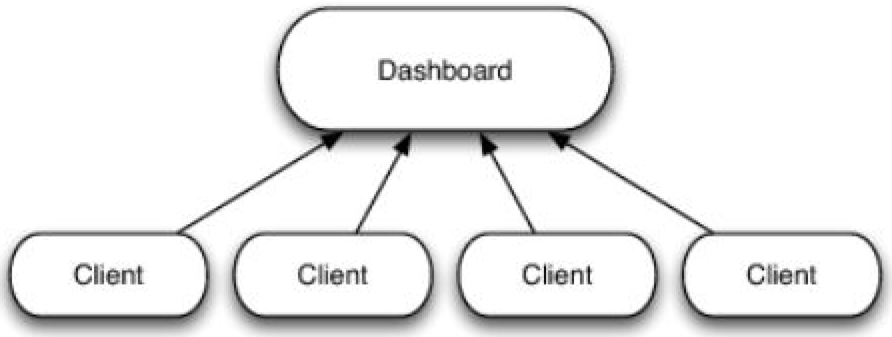
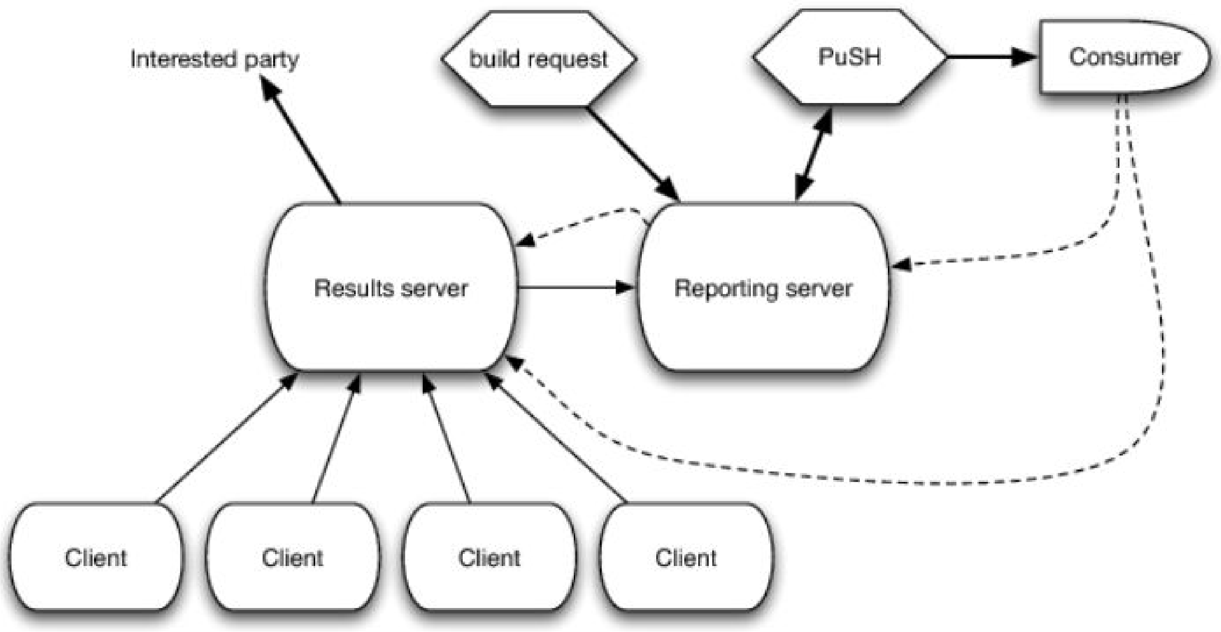


Figure 9.3: CDash Architecture

The CDash’s model mitigates some of the disadvantages of the Buildbot architecture. The CDash’s master allows clients to report back to it in XML-format. Clients are able to stop the connection, and the master could detect clients overload for better load balance and operation. Yet the model of CDash lacks the centralized model which was in the Buildbot example. there is no way to both globally request a build, and guarantee that anonymous clients perform the build in response to a check in clients must be considered unreliable.

Jenkins
Jenkins is a hybrid between the centralized model and the reporting model. A central server can master ssh to other servers, and other servers are able to send back xml JUnit feedback.

The Pony-build



This model adds to the CDash’s with additional structures that helps do the following:

1. Easier communication / modification.
2. Supports multiple languages of implementation.
3. Atomicity makes the partitions isolated, making it testable.
4. Configuration is easier.
5. Results could be easier to be accessed.

Yet the problems of this model is that requesting a build is difficult and independent of the results server. Real-time modeling is not well set up, and lack of resource locking makes it less efficient. In order to provide master resource locks with unreliable clients the master lock controller must have a policy in place for clients that take a lock and never release it, either because they crash or because of a deadlock situation.

Advantages and disadvantages of each model is the key to choose which architecture to work with for each company and project.