New Test Framework – Language

Test Framework - Today

Currently 37000 lines of bash

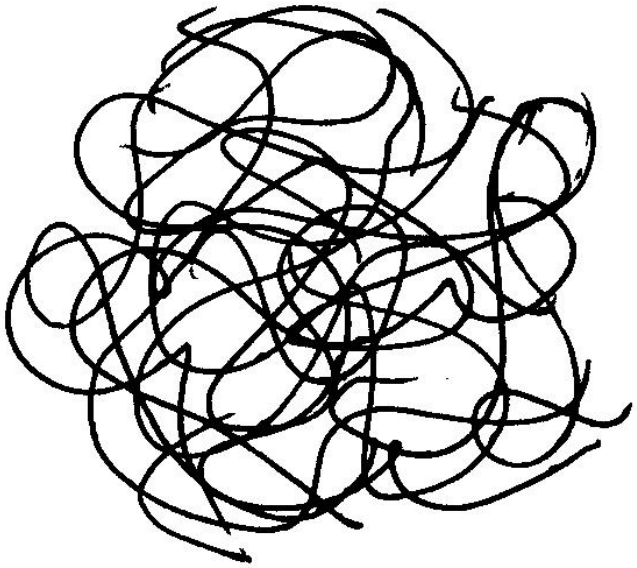
- Bash was not designed for this scale of development
- A lot of duplication with mild but critical variation

Test developer needs to fully understand the test framework to create scalable tests

- Tests do not naturally scale
- Very easy for test developer to created global debt

RPC mechanism problematic

- Use rpc, pdsh… which require separate sometimes impossible configuration from Lustre.

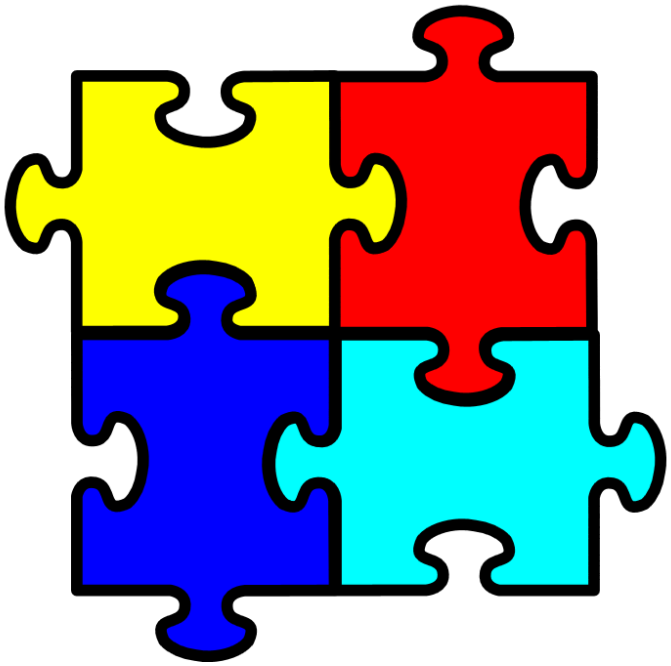


Test Framework – Tomorrow

Structured environment makes scalable tests the default behaviour

Environment must *encourage* people to develop tests that are

- Easy to read
- Structured
- Naturally Scalable
- Unnaturally Singular
  - Must be hard to produce non-scalable tests



Test Framework – Tomorrow

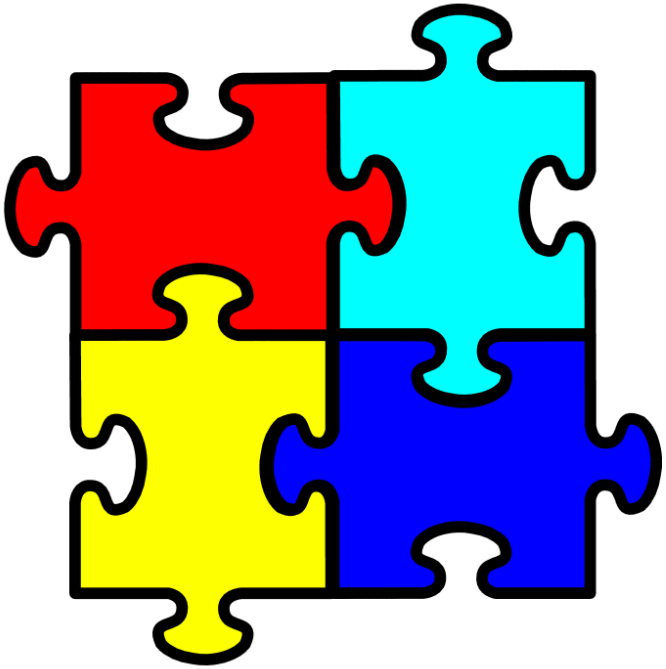
Structured environment that makes scalable tests the default behaviour

Language should be a modern language providing object oriented facilities

Language should provide the vocabulary to do everything required clearly and without ambiguity

Development should be easy even for the non-expert

Test scope must be limited to the test, no scope for global affects or global debt



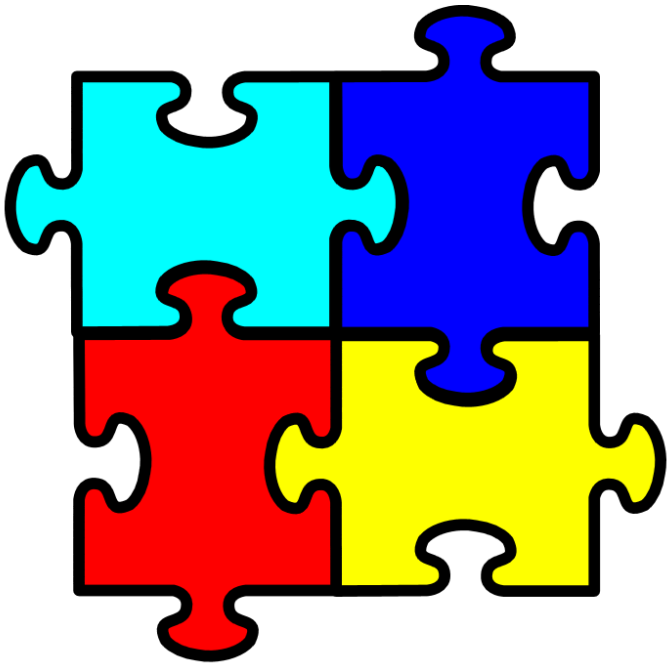
Test Framework – Tomorrow

Natural support for whole system capture of statistical information such as IO counts, RPC counts, Bandwidth

- Tests can require certain outcomes and results can be independently captured for late external analysis

Ability to inject memory/data/network errors, node wide stutter, node failure etc.

- Both as part of test and globally independently of test



Test Framework – Tomorrow

Full LibLustre API

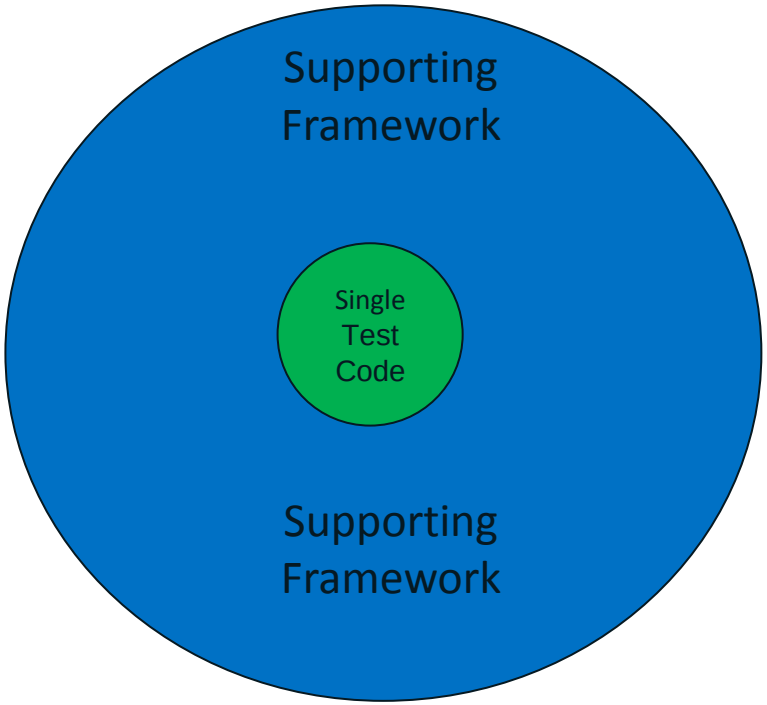
- API without the magic values or developer voodoo

Single test can be run without pre-preparation

- mot my-test.mot

Supported environment

- Separate project to Lustre
- Lustre developers own the Lustre tests
- Test Framework developers develop the framework
- Ownership equals quality



Test Development Tool – Tomorrow

This snippet would be the whole source for a test. Residing in a single source file.

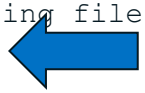
```
// name: Directory Create Test
// class: Sanity
// description: mkdir /d5 /d5/d2; ch          /d2
client_test(Environment env)
{
   mkdir d5
   mkdir d5/d2
   chmod 0707, d5/d2
   check_permissions FileType.file,0x707,"d5/d2"
}
```

This test is run on every client simultaneously.

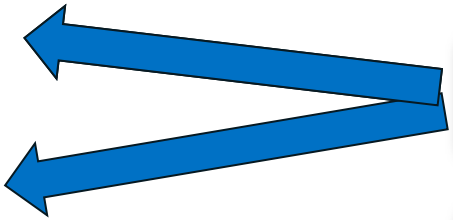
Failures in here reported by throwing of an exception

Test Development Tool – Tomorrow

```
// name: Directory Failover Writetest
// class: Failover
// description: failover OSSs whilst writing files
client_test_single(Environment env)
{
   every_client
   {
      read_write_server
   }

   every_oss
   {
       oss.failover
   }
}

read_write_server
{
   file = open("file","w+")

   write 100GB
   some_sync_with_all_mechanism
   read and verify 100 GB

   close file
}
```



This test is run on a single instance



This code is run on every client/oss simultaneously.

Failures reported by throwing of exceptions

Test Development Tool - Tomorrow

Require full featured Lustre wide network

- Real parameter driven generic typed RPC's

Use lnet as the underlying transport layer

- Reaches everywhere Lustre reaches
- Configuration option preventing security issues

Requires reliable lnet

- Lnet failure fails safe to test failure

Chicken and egg situation at startup

- How do we install lnet!