Improving Lustre Testing

Version 1.0

Nathan Rutman
Chris Gearing
Roman Grigorev

Advancing Digital Storage Innovation

Improving the Lustre Test Framework

LAD, Sep 2012

Nathan Rutman

Roman Grigoryev

Agenda

- Current Problems
- Possible Remedies
- Change Hurts
- A Plan
- Xperior

Current test problems

- Packaging
  - Test version tied to Lustre version
- Language
  - Bash language is not modular or portable
  - Test scripts are not coherent
- Test selection
  - No test "filters" based on attributes
  - Many tests can't be run individually
    - no random testing
    - no directed testing
- Coverage
  - Poor code coverage (66% functions, 44% branches)
  - Some tests are ineffective, duplicates

Current test problems, continued

- Framework has limited abilities to determine test status
  - universal timeout
  - crashes stop tests
  - output is not machine-readable
- Configuration control is a mess
  - inconsistent formatting abilities
  - inconsistent mounting abilities
  - no/limited control over
    - deployment
    - versions
    - distros
    - vm's
- No verification of the t-f itself

Current test problems, continued

- Can't test real systems
    - setup
    - safety
    - connectivity
- No ability to setup, configure, or explicitly test routing

Possible remedies

- Better test language
- Clarify local/remote operations, roles
- Remote commands via LNET
- Separate configuration and setup/cleanup from tests
- Clarify/enforce test independence
- Remove version dependence
- Package independence

How far down the rabbit hole?

A. Minimally change tests, new framework emulates old
  - Less effort
  - No need for test verification
  - Limited benefits
  - Difficult interfacing with bash

B. New framework, wholesale test changes
  - Tests will have strict, verifiable syntax
  - Single language
  - Strict test life cycles
  - Must develop all functions, port all tests
  - Must verify test correctness

C. New framework for new tests, wrapper around old tests
  - Smooth transition
  - Must develop all functions

A Plan

- Aiming for option C
- Clarify and enforce the API to framework functions
- Add API for
  - filter-by-label
  - precondition checks - 'skip' shouldn't be a test result
  - permissions mask
  - parameter setting
  - file creation
- Add test metadata labels (e.g. failover, safe, min_ost_count=2)
- Test output in YAML form
  - pass/fail
  - timing
  - error codes, line nums
- Create separate test package
- Add LNET remote procedure testing upcall (maybe)

Separate tests

- Either fully separate or combine interrelated tests
  - if tests are separate, they must have value when executed separately
- Call error fn at any stage that may fail
  - report error code and line number
- Every test restores original state
  - parameter settings (record/restore)
  - leftover files
- Test ordering must be randomizable
- Test ordering must be repeatable
- Parallel test execution

Permission mask

- Impossible to separate setup from some tests
- Instead, explicitly grant permissions for tests
  1. small client-based filesystem operations
  2. large (OST fill) client ops
  3. change debugging levels / capture log ouput
  4. set a temporary parameter (set_param)
  5. set a permanent parameter (conf_param)
  6. set obd_fail_loc
  7. register a changelog user
  8. start/stop a client
  9. start/stop a server
  10. failover node to itself
  11. failover a node to partner
  12. format a target as an mdt/ost
- These are good candidates for an API

Xperior

To put to the test

- Xyratex is developing for test execution
  - open source when finished
- Features
  - Results are in YAML
    - Also supports TAP and HTML output
  - Per test configuration (YAML) in separate files
    - tags, timeouts, etc
    - config file correctness checker
  - Test selection lists
  - Framework unit tests verify framework itself

Xperior, con't

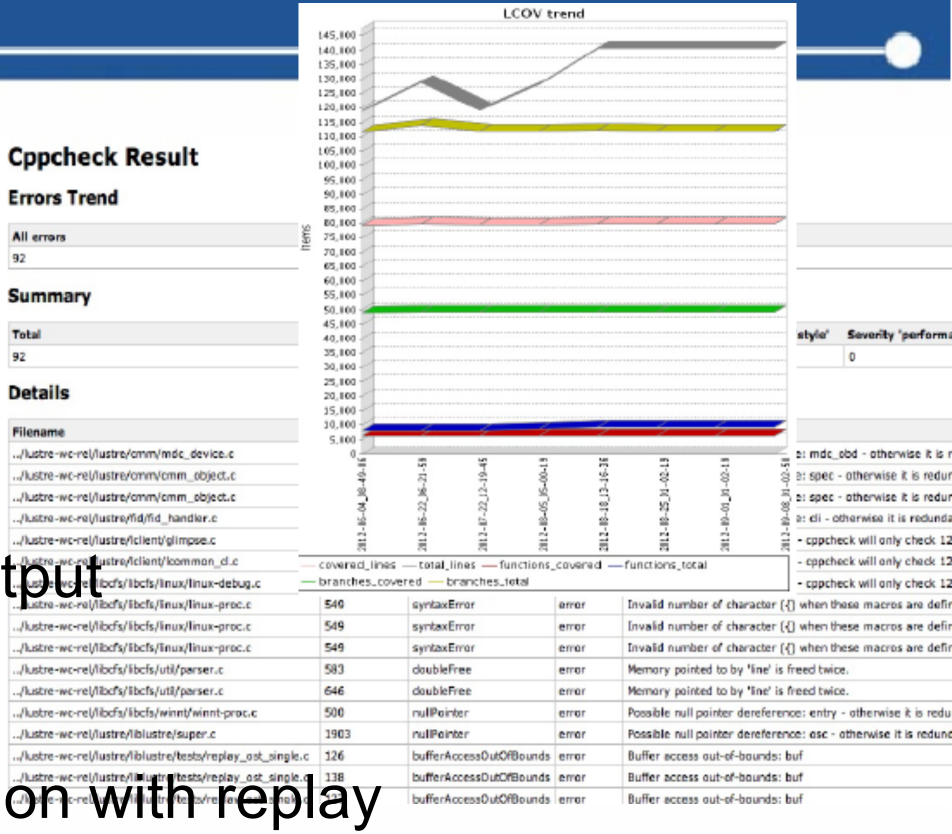
- Framework divided into 4 components
  - management: execution, reporting, filtering
  - configuration
- continuous integration (git triggers)
- plugins: additional execution and behavior modules

- Currently supports
  - Lustre tests (ONLY=)
  - IOR, mdtest
  - virtual and real machines

Xperior plugins

- Plugins
  - code coverage tool
  - static verification tool
  - reformat after every test
  - store console output
  - store lustre-diagnostics output

- Under development
  - Random order test execution with replay
  - MDSIM test execution
  - IPMI support



Cppcheck Result

Errors Trend

All errors
92

Summary

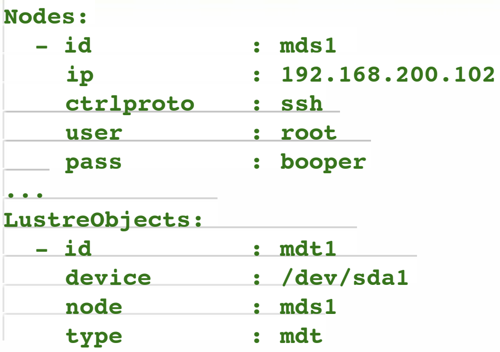
Total		style'	Severity 'performa
92			0

Details

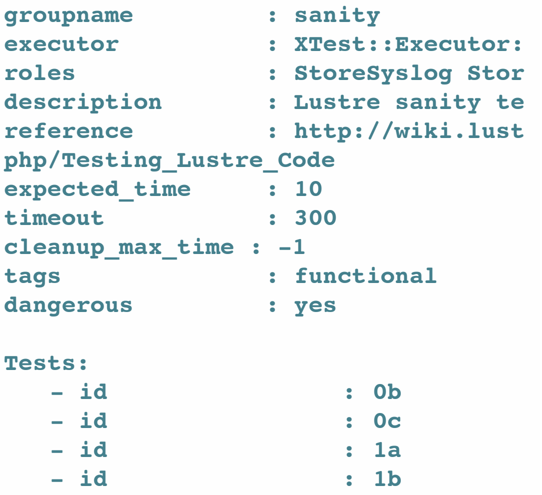
Filename				
../lustre-wc-rel/lustre/cmm/mdc_device.c				e: mdc_obd - otherwise it is r
../lustre-wc-rel/lustre/cmm/cmm_object.c				e: spec - otherwise it is redur
../lustre-wc-rel/lustre/cmm/cmm_object.c				e: spec - otherwise it is redur
../lustre-wc-rel/lustre/fid/fid_handler.c				e: cli - otherwise it is redunda
../lustre-wc-rel/lustre/lclient/glimpse.c				- cppcheck will only check 12
../lustre-wc-rel/lustre/lclient/lcommon_cl.c				- cppcheck will only check 12
../lustre-wc-rel/libcfs/libcfs/linux/linux-debug.c				- cppcheck will only check 12
../lustre-wc-rel/libcfs/libcfs/linux/linux-proc.c	549	syntaxError	error	Invalid number of character ({) when these macros are defir
../lustre-wc-rel/libcfs/libcfs/linux/linux-proc.c	549	syntaxError	error	Invalid number of character ({) when these macros are defir
../lustre-wc-rel/libcfs/libcfs/linux/linux-proc.c	549	syntaxError	error	Invalid number of character ({) when these macros are defir
../lustre-wc-rel/libcfs/libcfs/util/parser.c	583	doubleFree	error	Memory pointed to by 'line' is freed twice.
../lustre-wc-rel/libcfs/libcfs/util/parser.c	646	doubleFree	error	Memory pointed to by 'line' is freed twice.
../lustre-wc-rel/libcfs/libcfs/winnt/winnt-proc.c	500	nullPointer	error	Possible null pointer dereference: entry - otherwise it is redu
../lustre-wc-rel/lustre/liblustre/super.c	1903	nullPointer	error	Possible null pointer dereference: osc - otherwise it is redund
../lustre-wc-rel/lustre/liblustre/tests/replay_ost_single.c	126	bufferAccessOutOfBounds	error	Buffer access out-of-bounds: buf
../lustre-wc-rel/lustre/liblustre/tests/replay_ost_single.c	138	bufferAccessOutOfBounds	error	Buffer access out-of-bounds: buf
../lustre-wc-rel/lustre/liblustre/tests/replay_ost_single.c	[illegible]	bufferAccessOutOfBounds	error	Buffer access out-of-bounds: buf

Xperior, con't

- Initiated via Jenkins
- System config YAML
- Set up cluster
- Install any required test packages
- Test description YAML
- Results YAML

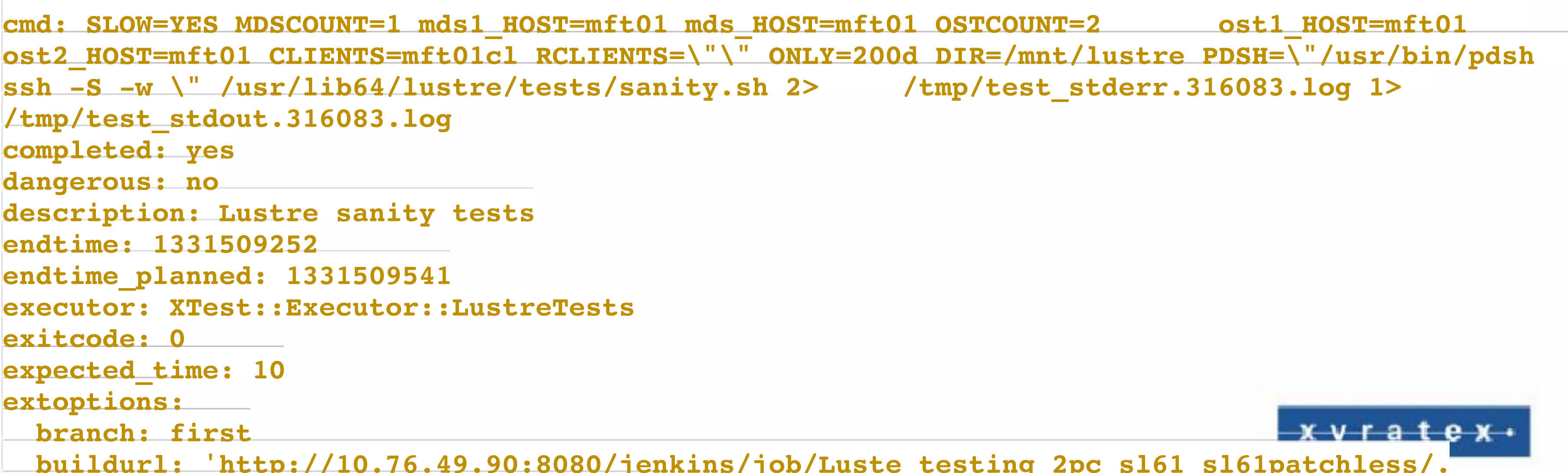


```
Nodes:
  - id            : mds1
    ip            : 192.168.200.102
    ctrlproto     : ssh
    user          : root
    pass          : booper
...
LustreObjects:
  - id            : mdt1
    device        : /dev/sda1
    node          : mds1
    type          : mdt
```



```
groupname          : sanity
executor           : XTest::Executor:
roles              : StoreSyslog Stor
description        : Lustre sanity te
reference          : http://wiki.lust
php/Testing_Lustre_Code
expected_time      : 10
timeout            : 300
cleanup_max_time : -1
tags               : functional
dangerous          : yes

Tests:
   - id                    : 0b
   - id                    : 0c
   - id                    : 1a
   - id                    : 1b
```



```
cmd: SLOW=YES MDSCOUNT=1 mds1_HOST=mft01 mds_HOST=mft01 OSTCOUNT=2       ost1_HOST=mft01
ost2_HOST=mft01 CLIENTS=mft01cl RCLIENTS=\"\" ONLY=200d DIR=/mnt/lustre PDSH=\"/usr/bin/pdsh
ssh -S -w \" /usr/lib64/lustre/tests/sanity.sh 2>     /tmp/test_stderr.316083.log 1>
/tmp/test_stdout.316083.log
completed: yes
dangerous: no
description: Lustre sanity tests
endtime: 1331509252
endtime_planned: 1331509541
executor: XTest::Executor::LustreTests
exitcode: 0
expected_time: 10
extoptions:
  branch: first
  buildurl: 'http://10.76.49.90:8080/jenkins/job/Luste_testing_2pc_sl61_sl61patchless/.
```

What Next

- We will continue to work on
  - Xperior
  - Separating tests
  - Integrating new features into the current t-f API
- Need community discussion/help on
  - Further defining the new framework API
  - Choosing and porting tests to a new language
  - Changing old tests to meet the new API

Statistical Performance Testing

Chris Gearing - chris.gearing@intel.com
Sr. Staff Engineer

Statistical Performance Testing

Analogue testing results

Existing method for performance testing

Statistical performance testing

First steps to statistical performance testing

Broadening performance testing

Problem Statement

Discussion Document

Not all testing is straight pass or fail

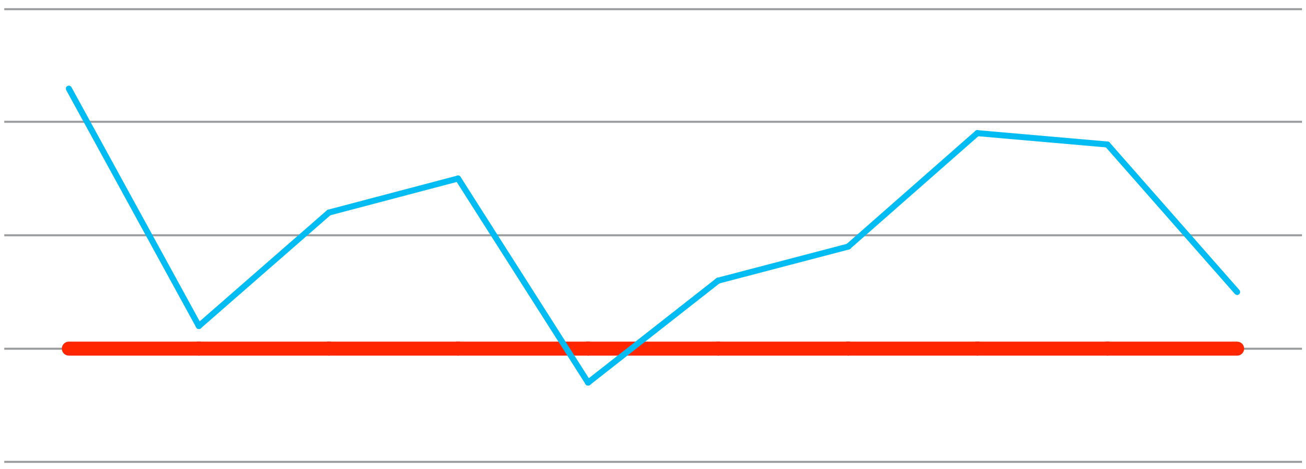
Functional testing generally is

- Did the file rename correctly
- Did the data write correctly
- Has the OSS successfully failed over

Performance testing generally is not

- What was the data rate of the writes
- How quickly did the OSS fail over
- How many RPC's did we generate per megabyte of data

Performance Testing Today



Errors are dealt with by

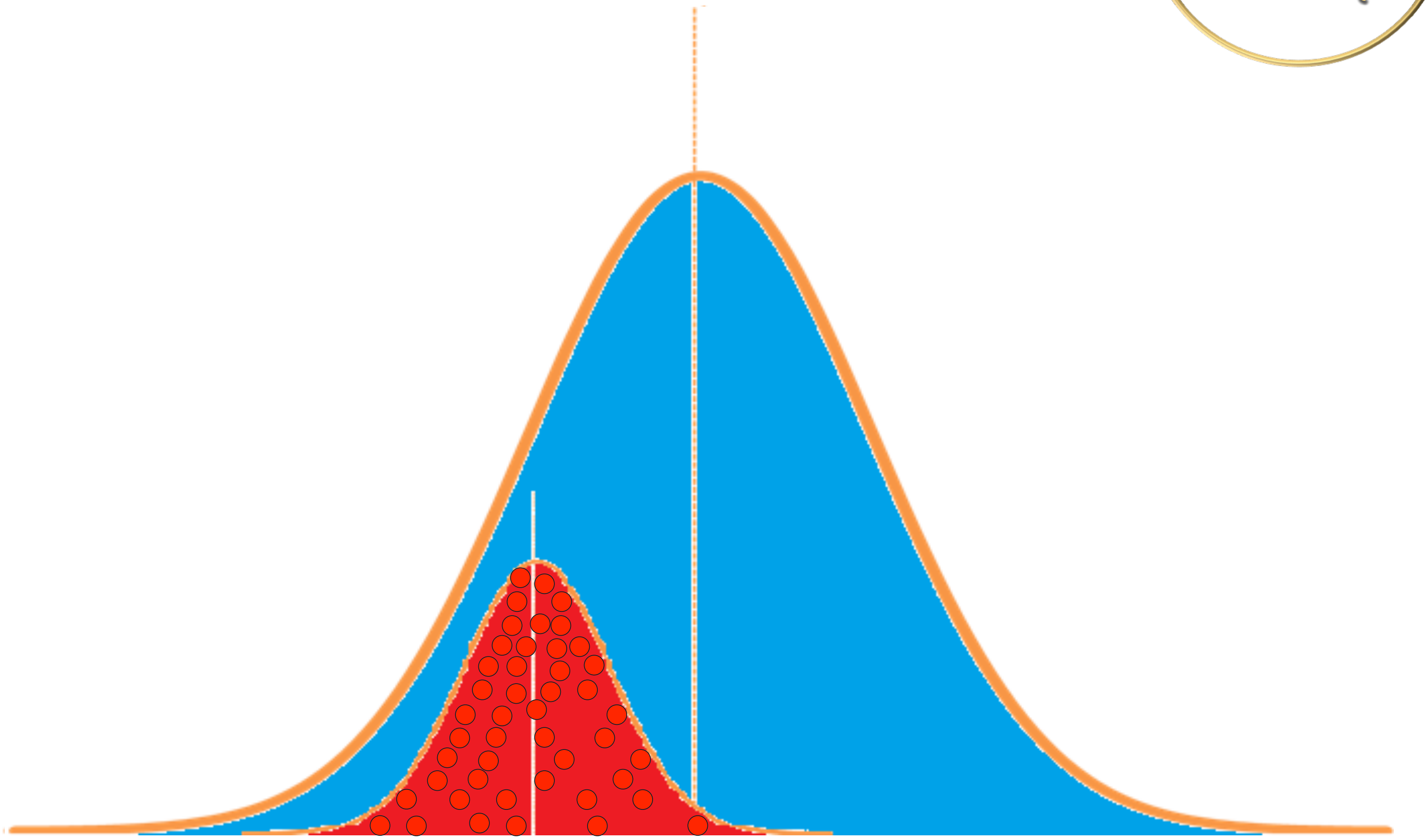
- Re-running test
- Changing hardware
- Making ‘experience based assessments’

Reality is results are probability based

- Humans not good at assessing probability

Simple example of problem

Discussion Document

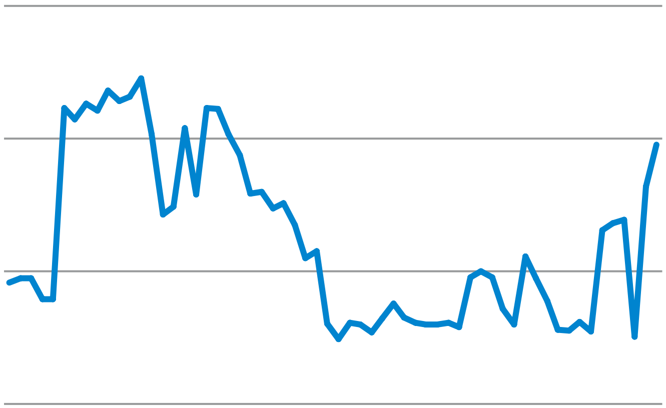


Variation of performance

Discussion Document

We can think of performance variation in 2 ways

- Natural causes
  - Are when variability of results are within the normal range
  - The process is under control
- Assignable causes
  - Are when the variability of results are outside the normally expected range
  - The process is out of control and a change has happened that we must be able to assign a cause to



Natural Variations

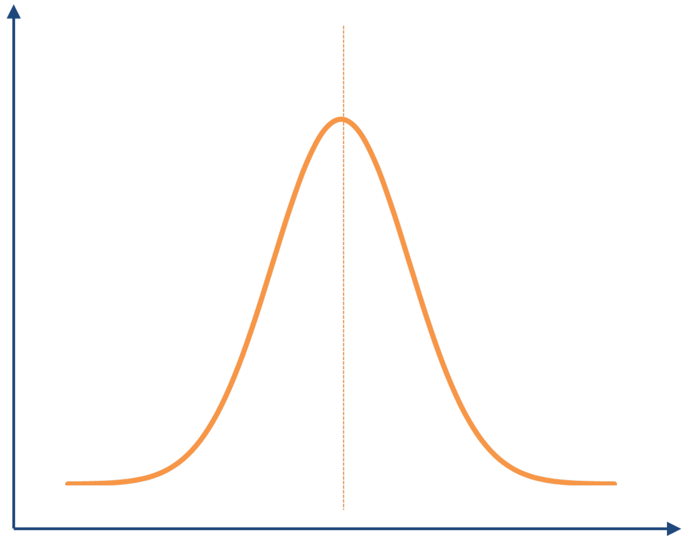
Discussion Document

Natural variations in the test process

These are to be expected

Output measures follow a probability distribution

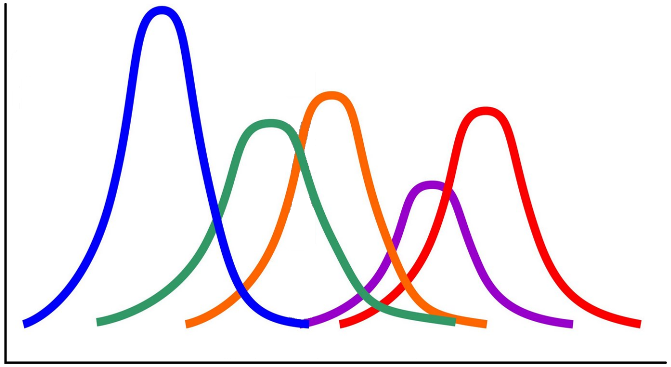
For any distribution there is a measure of central tendency and dispersion



Assignable Variations

Variations that can be traced to a specific reason

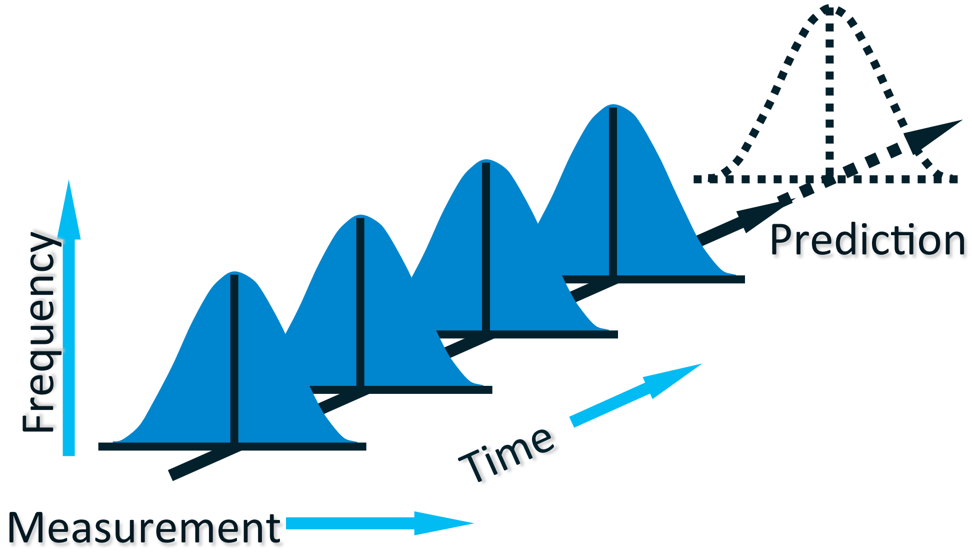
- Lustre bug
- Test infrastructure failure
- System configuration
- Expected change – bug fix / new feature
- …



The objective is to discover when assignable causes are present and eliminate or explain them

Sample Natural Variation

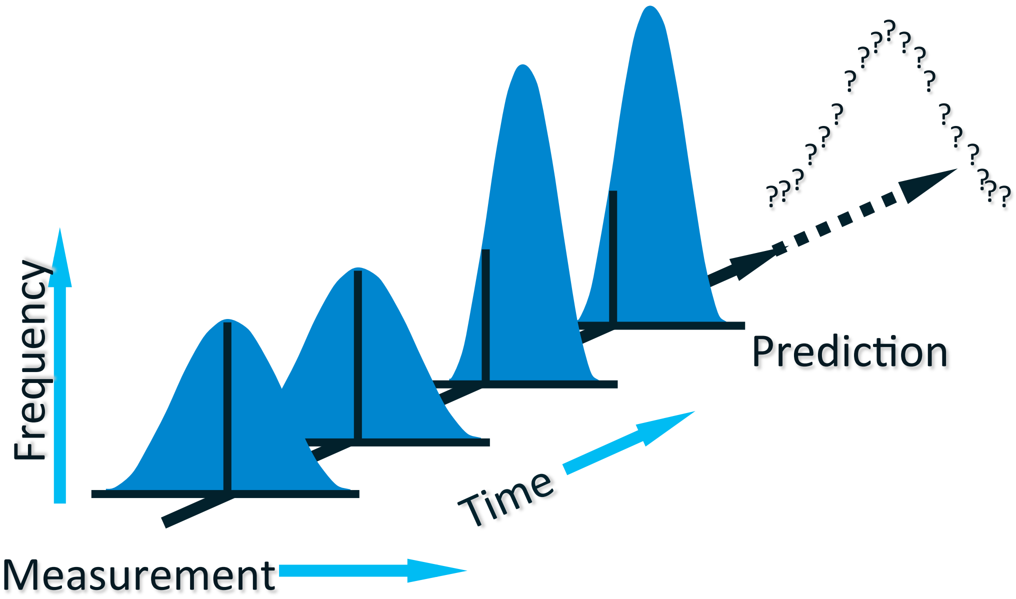
When only natural causes of variation are present, the output of a process forms a distribution that is stable over time and is predictable



Samples Assignable Variation

Discussion Document

When assignable causes are present, the process output is not stable over time and is not predicable



How do we spot Assignable Variation?

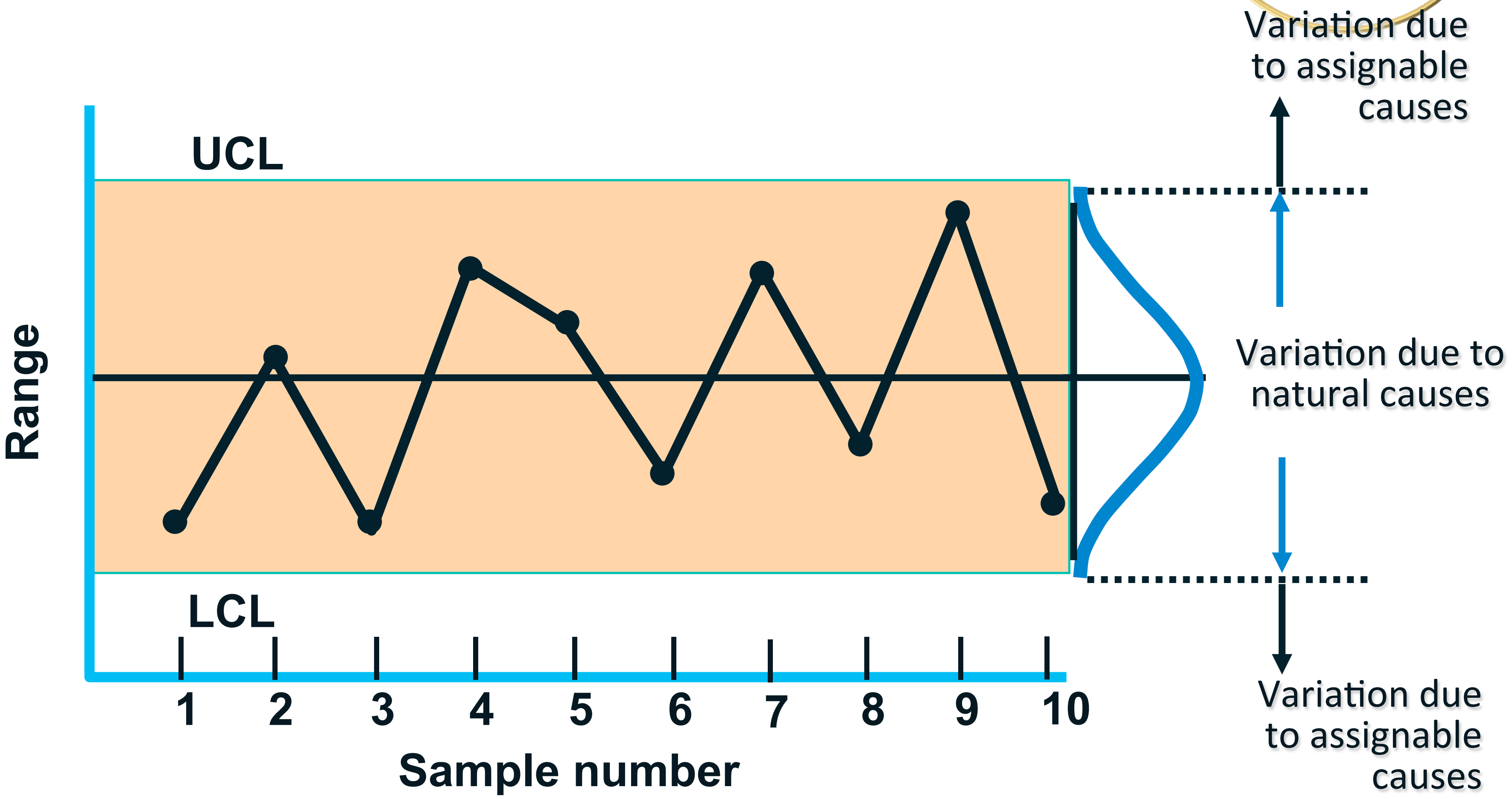
Statistical Process Control

- SPC is a methodology associated with the manufacturing production environment
  - We are manufacturing software and have a production line
- Control charts provide for differentiating Natural from Assignable

The procedure is

- Sample the process at regular intervals
- Plot the measure of performance, e.g.
  - Test execution time
  - RPC count
  - Bandwidth
  - …
- Check (graphically) if the process is under statistical control
- If the process is not under statistical control, do something about it

Control Charts

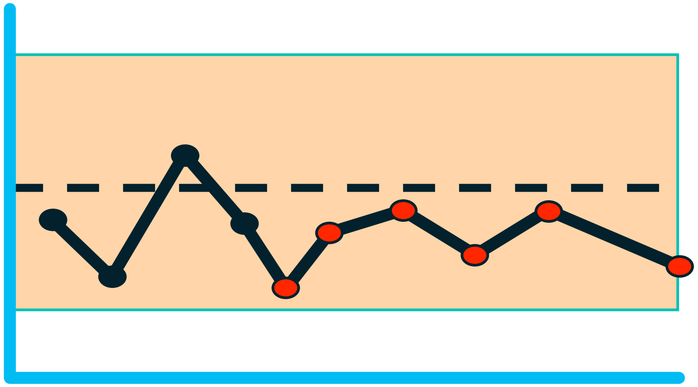


Variation due to assignable causes

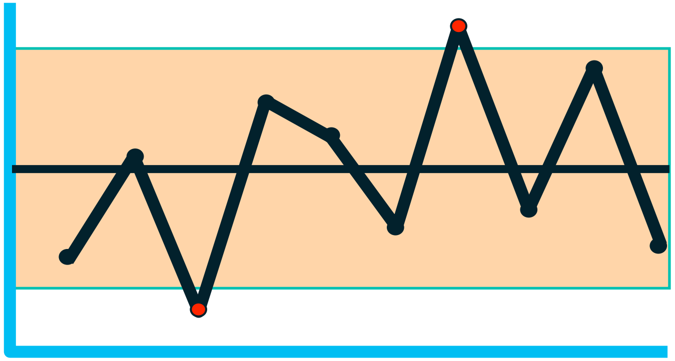
Variation due to natural causes

Variation due to assignable causes

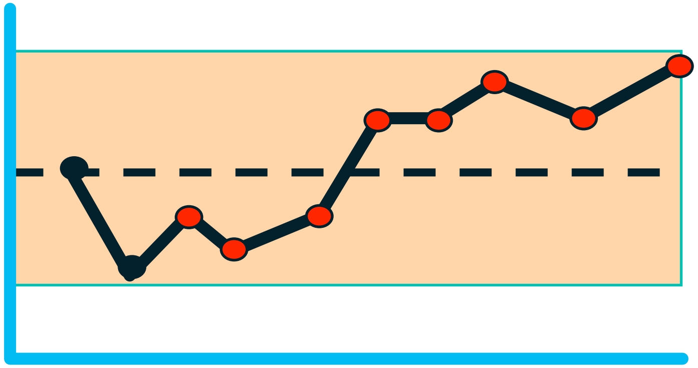
Control Charts



Results consistently above / below the center line



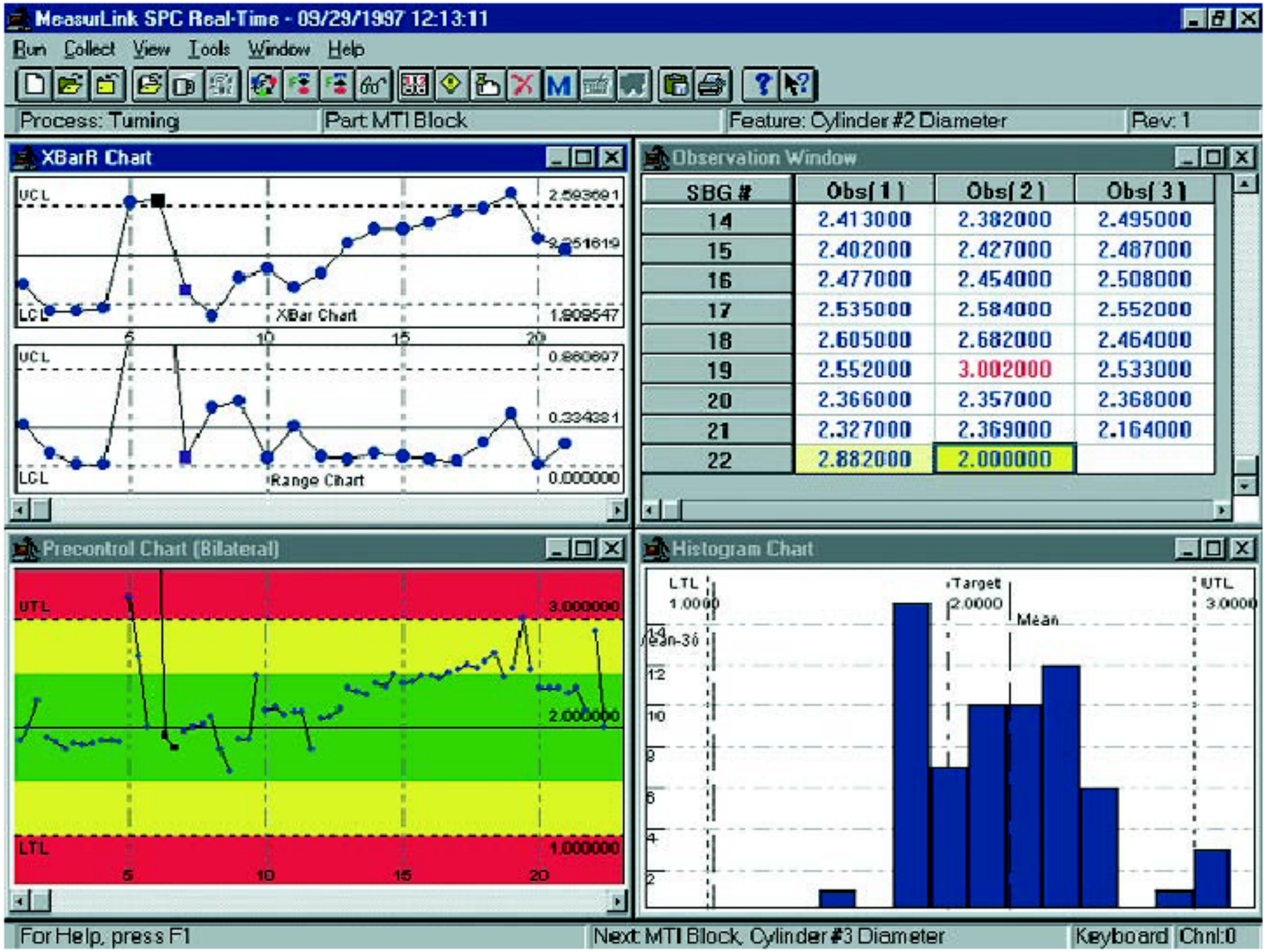
Results outside of specification



Results consistently increasing / decreasing

Automated Control Charts

Discussion Document



Process: Turning | Part: MTI Block | Feature: Cylinder #2 Diameter | Rev: 1

SBG #	Obs(1)	Obs(2)	Obs(3)
14	2.413000	2.382000	2.495000
15	2.402000	2.427000	2.487000
16	2.477000	2.454000	2.508000
17	2.535000	2.584000	2.552000
18	2.605000	2.682000	2.464000
19	2.552000	3.002000	2.533000
20	2.366000	2.357000	2.368000
21	2.327000	2.369000	2.164000
22	2.882000	2.000000	

For Help, press F1 | Next: MTI Block, Cylinder #3 Diameter | Keyboard | Chnl:0

Next Steps

Discussion Document

Focus on current ‘performance’ testing

Manually capture and analyse results

Use this manual process to validate performance for 2.4 release

Review after 2.4 release to begin developing processes moving forwards

Long term extend performance to mean every useful measure of Lustre behaviour

Summary

Discussion Document

We can use statistical methods to help control the changes in Lustre

SPC is a good *candidate* for that control mechanism

Using commercial tools and current testing the first steps are within reach

Autotest should allow broad analysis of Lustre performance and behaviour

Thank you