Building a Robust Linux Kernel piggybacking
The Linux Test Project

Subrata Modak & Balbir Singh,
Linux Technology Center, IBM,
India

OLS 2008

Dated 25/07/2008
Agenda

- Introduction – some background checks
- Addressing some Criticism
- Breathing a new life into LTP
- Kernel code coverage statistics
- Role of LTP in testing Linux
- Early & Effective testing
- Simplest way to write an LTP test case
- Future Plans
- Conclusion
Everybody is happy, and... they should be

- Linux Kernel is growing at a rapid rate and runs across numerous architectures...neighbor’s envy – owner’s pride

Mission Critical...

- Keeping the kernel reliable, stable & robust

Somebody doing the Job...??

- Yes, The Linux Test Project
- It was established to meet the very goals stated above
Beware to ignore testing...!! you will end up in

- Frequent crashes
- Frequent Updates
- Unhappy users
Introduction – some background checks

- First created by SGI in 2001 - brought organized and automated testing to Linux for first time
- Primary goal: provide a test suite that can validate the robustness, stability & reliability of Linux kernel
- A functional and regression test suite that allows to do stress testing as well
- No. of tests at kernel 2.3 was mere 100
- Today at 2.6.25, it stands at 3000+ and growing...
- Tests numerous kernel features, namely syscalls, MM, IPC, I/O, device drivers, FS, Networking, etc
- What’s the code written in ??... 95% in C, remaining in SHELL & PERL
Addressing some Criticism

People pointed out these gray areas to me

- LTP does not have
  - automatic kernel build, install, reboot and test
  - Good code coverage
  - parsable output/logs
  - Broken test cases

My observation instead

- LTP was not designed to do auto build and test – *it was designed more of an handy regression test suite, testing all possible kernel APIs*

- Kernel code coverage cannot be drastically improved without corresponding test cases being made available – *we cannot enforce this, but can highlight the impact of not doing so*

- LTP logs/output are very neatly designed with proper tags – *analysis by human and programmes can be simple*

- Broken issues: some test cases were not cleaned for long – *LTP clearly distinguishes and documents meaning of keywords like PASS, FAIL, CONF, WARN, BROK, RETR & INFO*
LTP´s Output & Log samples

**Sample LTP Output**

```plaintext
<<<test_start>>>
tag=remap_file_pages01 stime=1208361993
cmdline="remap_file_pages01"
contacts=""
analysis=exit
initiation_status="ok"
<<<test_output>>>
remap_file_pages01
1 PASS : Non-Linear shm file OK
<<<execution_status>>>
duration=1 termination_type=exited
termination_id=0 corefile=no
cutime=7 cstime=2
<<<test_end>>>
```

**Sample LTP Log**

```

<table>
<thead>
<tr>
<th>Testcase</th>
<th>Result</th>
<th>Exit Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>remap_file_pages01</td>
<td>PASS</td>
<td>0</td>
</tr>
<tr>
<td>faccesat01</td>
<td>FAIL</td>
<td>1</td>
</tr>
<tr>
<td>fallbackate03</td>
<td>WARN</td>
<td>1</td>
</tr>
</tbody>
</table>

Total Tests: **3**
Total Failures: **1**
Kernel Version: 2.6.18-53.1.13.el5
Machine Architecture: i686
Hostname: <sniff>```
Breathing a new life into LTP

- LTP started afresh from the early days of 2007 – developers put huge effort and added lot of fuel to testing Linux
- Numerous new test cases added, testing varied regions and kernel types – kdump, RT Linux etc
- Massive cleanup to existing test cases done – applied 350 patches; adding 1000 sources, modifying 1000 equally and deleting around 250
- Numerous broken test case(s) issues were fixed
List of test cases added – Jan07 till Mar08

<table>
<thead>
<tr>
<th>Test cases Type</th>
<th>Total Sources</th>
<th>Avg. code size</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kdump,</td>
<td>26</td>
<td>2312</td>
<td>Sachin, Poornima, Jeff Burke, Cai Quan</td>
</tr>
<tr>
<td>Uts, Sysvipc, &amp; Pid Namespace</td>
<td>27</td>
<td>2614</td>
<td>Rishikesh, Sergei, Suka &amp; others</td>
</tr>
<tr>
<td>Inotify</td>
<td>4</td>
<td>5894</td>
<td>Andrew Vagin, Ricardo, Vivi Li &amp; Others</td>
</tr>
<tr>
<td>Writev</td>
<td>7</td>
<td>7712</td>
<td>Breno Leitao</td>
</tr>
<tr>
<td>Swapon</td>
<td>4</td>
<td>8975</td>
<td>Ricardo &amp; others</td>
</tr>
<tr>
<td>Numa</td>
<td>6</td>
<td>6986</td>
<td>Sivakumar, Pradeep &amp; others</td>
</tr>
<tr>
<td>Remap_file_pages</td>
<td>3</td>
<td>6565</td>
<td>Ingo, Nick, Ricardo</td>
</tr>
<tr>
<td>Nfs Check Tests</td>
<td>1</td>
<td>1834</td>
<td>Kumar Gala</td>
</tr>
<tr>
<td>Posix_Fadvise &amp; Fadvise64</td>
<td>5</td>
<td>4003</td>
<td>Masatake Yamato &amp; others</td>
</tr>
<tr>
<td>Madvise</td>
<td>4</td>
<td>6572</td>
<td>Pavan Naregundi</td>
</tr>
<tr>
<td>Sendfile64</td>
<td>7</td>
<td>5625</td>
<td>Masatake Yamato</td>
</tr>
<tr>
<td>Arm Specific</td>
<td>1</td>
<td>1091</td>
<td>Riaz Ur Rahaman</td>
</tr>
<tr>
<td>Real Time Linux</td>
<td>101</td>
<td>3400</td>
<td>Nivedita, Sudhanshu, Chirag, Ankita, Darren, Sebastien, Gilles, Robert Schwabel &amp; others</td>
</tr>
<tr>
<td>Fallocate</td>
<td>5</td>
<td>7071</td>
<td>Sharyathi Nagesh</td>
</tr>
<tr>
<td>Filescaps</td>
<td>11</td>
<td>2579</td>
<td>Sergei, Andrew Morgan &amp; Henry Yei</td>
</tr>
<tr>
<td>Cpu Controllers</td>
<td>17</td>
<td>5134</td>
<td>Sudhir Kumar</td>
</tr>
<tr>
<td>Msgctl</td>
<td>12</td>
<td>7985</td>
<td>Nadia Derbey</td>
</tr>
<tr>
<td>Ti-Rpc</td>
<td>588</td>
<td>3218</td>
<td>Aurélien Charbon</td>
</tr>
</tbody>
</table>
Issues addressed in LTP-Refresh

- Release pattern revived to include results on various architectures
- Total 169 packages (265 Mb code) released, 31458 packages downloaded making avg. of 65 downloads/day
- GCOV kernel patches from linux-2.6.18 till 2.6.25
- Made RHEL5 LSPP EAL4+ certification test Suite available for IBM Hardwares
- SGI Common Criteria EAL4 certification test suite for RHEL5.1 on SGI Altix 4700 (ia64) and Altix XE (x86_64) Systems was also made available
Infrastructural improvements during LTP-Refresh

- Infrastructure gives the methodology/ways to run tests
- Adding discrete sequential run capability – run as many loops I want
- Auto mail back option of reports – bundle output/logs/failed tests and mail them back
- Generating default file for failed tests – this can be taken a command file for next ltp run
- Integrating better stress generation capability – allow more hogging of the CPU, Memory, I/O Channels, Storage & Network during test run
- Allow concurrent execution of test cases – test the SMP code
LTP Output/Log (Report Generated on Mon Mar 31 12:14:29 CDT 2008)

Detailed Report

<table>
<thead>
<tr>
<th>No</th>
<th>Test-Name</th>
<th>Command-Line</th>
<th>Test-Output</th>
<th>Termination-id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abort01</td>
<td>ulimit -c 1024;abort01</td>
<td>abort01 1 PASS : Test passed</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>facessat01</td>
<td>facessat01</td>
<td>facessat01 6 FAIL : facessat() Failed, errno=20 : Not a directory</td>
<td>1</td>
</tr>
<tr>
<td>94</td>
<td>fallocate01</td>
<td>fallocate01</td>
<td>fallocate01 0 WARN : system doesn't support execution of the test</td>
<td>0</td>
</tr>
<tr>
<td>183</td>
<td>ftruncate04</td>
<td>ftruncate04</td>
<td>ftruncate04 1 CONF : The filesystem where /tmp is mounted does not support mandatory locks. Cannot run this test.</td>
<td>0</td>
</tr>
</tbody>
</table>

Summary Report

<table>
<thead>
<tr>
<th>Test Summary</th>
<th>Pan reported some Tests FAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTP Version</td>
<td>LTP-20080331</td>
</tr>
<tr>
<td>Start Time</td>
<td>Mon Mar 31 12:14:29 CDT 2008</td>
</tr>
<tr>
<td>End Time</td>
<td>Mon Mar 31 13:05:21 CDT 2008</td>
</tr>
<tr>
<td>Log Result</td>
<td>/root/subrata/ltp/ltp-full-20080331/results</td>
</tr>
<tr>
<td>Output/Failed Result</td>
<td>/root/subrata/ltp/ltp-full-20080331/output</td>
</tr>
<tr>
<td>Total Tests</td>
<td>860</td>
</tr>
<tr>
<td>Total Failures</td>
<td>3</td>
</tr>
<tr>
<td>Kernel Version</td>
<td>2.6.21.3</td>
</tr>
<tr>
<td>Machine Architecture</td>
<td>i386</td>
</tr>
<tr>
<td>Hostname</td>
<td>sniff</td>
</tr>
</tbody>
</table>
Kernel code coverage statistics

<table>
<thead>
<tr>
<th>Directory</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>49.8% 10135/20367 lines</td>
</tr>
<tr>
<td>include/asm</td>
<td>49.4% 595/1204 lines</td>
</tr>
<tr>
<td>include/linux</td>
<td>58.7% 2239/3812 lines</td>
</tr>
<tr>
<td>include/net</td>
<td>56.2% 990/1762 lines</td>
</tr>
<tr>
<td>ipc</td>
<td>52.8% 1442/2729 lines</td>
</tr>
<tr>
<td>kernel</td>
<td>38.2% 9880/25837 lines</td>
</tr>
<tr>
<td>lib</td>
<td>42.2% 2105/4992 lines</td>
</tr>
<tr>
<td>mm</td>
<td>51.5% 6899/13396 lines</td>
</tr>
<tr>
<td>net</td>
<td>65.4% 630/964 lines</td>
</tr>
<tr>
<td>security</td>
<td>51.9% 666/1283 lines</td>
</tr>
</tbody>
</table>

Code coverage Improvements

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>% Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filesystems</td>
<td>3.1</td>
</tr>
<tr>
<td>include/asm</td>
<td>1.5</td>
</tr>
<tr>
<td>include/linux</td>
<td>1.3</td>
</tr>
<tr>
<td>include/net</td>
<td>1.4</td>
</tr>
<tr>
<td>ipc</td>
<td>3.6</td>
</tr>
<tr>
<td>kernel</td>
<td>0.9</td>
</tr>
<tr>
<td>lib</td>
<td>1.0</td>
</tr>
<tr>
<td>mm</td>
<td>1.2</td>
</tr>
<tr>
<td>net</td>
<td>0.3</td>
</tr>
<tr>
<td>security</td>
<td>0</td>
</tr>
</tbody>
</table>

**Test cases excluded from code coverage: Kdump, RT, Dots, Open Posix, Open HPI, Pounder21 & SE Linux**
Analysis of code coverage

• Coverage better between 2 runs – indicative of LTP’s progress
• Subsystems fs & include/asm is now above 50%
• Some more interesting facts
  – LTP needs to do a better job covering error paths – we propose LTP-robust subproject to work well with fault injection
  – Not possible to test every config/boot option & extract coverage
  – Not possible to handle code coverage not exposed to user space – machine configured with SPARSEMEM, FLATMEM, DISCONTIGMEM
  – Several areas/subsystems does not have any code coverage – we need to write test cases for them as well
Role of LTP in testing Linux

- Broader software testing categories includes – compilation, unit, functional, system, stress & performance testing
- Unit testing – execution for individual/isolated feature alone,
- Functional/regression testing through comparison of successive kernel releases
- System testing – "C" Library and user interfaces provided by the kernel
- Stress testing – through specific test cases, additional background noise, concurrent test execution
- Unable to do compilation & performance testing
Early & Effective testing

- Spend more time on design – else be ready to spend 40 to 1000 times more time in fixing code after deployment
- Testing cannot catch bug during requirement analysis, but can definitely do before code deployment

Hypothetical Example: Sample bug fix flow for a bug introduced in version x and fixed in version x+3
Simplest way to write an LTP test case

• This Paper will not focus in intricate details of writing an LTP test case
• We instead focus on presenting a set of work flows
• Ways to write test cases can be found from already published papers
  – Testing Linux with Linux Test Project, Paul Larson, OLS 2002
  – Improving the Linux Test Project with Kernel Code Coverage Analysis, Paul Larson, OLS 2003
LTP Suite execution framework
Individual test case execution framework
Future Plans

- LTP-devel rpms – a light-weight package containing LTP-specific headers, libraries, man pages – enabling developers to write LTP unit test cases
- LTP-mm tree: contribute test cases to LTP-mm tree when your feature is in kernel-mm tree, or, you have just submitted your patch to LKML
- Enabling test case submission in any programming language which returns just 0/1 on pass/fail
- LTP aims to include test case in the areas of Power Management, Controllers/Containers, KDUMP, Union Mount, Shared subtree, missing syscalls, bleeding edge kernels, etc
- Enhancement in LTP Infrastructure
  - Development of XML logs/output
  - .config based build & execution
  - Network based installation, execution & report collection
- Entire bunch of test cases to be made concurrency safe
- Efforts on functional & regression testing to strengthen; also to add benchmark infrastructure in the long run
Conclusion

- LTP is Open Source: everybody can say how it should move forward, what it should address and... avoid !!
- LTP community highly appreciates patches in any form – the benefit of which directly goes to the community
- It will continue to be as a major functional & regression suite, and will keep growing along with the unstoppable kernel

For all these to happen, LTP will require:

1) More active participation from Kernel Developers, and
2) Tons & tons of test cases from you all
Acknowledgments

- Robert Williamson, IBM, for his inputs to and review of the draft of this paper
- Premalatha Nair & Sudarshan Rao (my Managers) & my team mates for their active support, encouragement and enthusiasm
- A special thanks to all developers whose immense contribution keeps this project growing
Legal Statement

- International Business Machines Corporation 2008. Permission to redistribute in accordance with Linux Symposium submission guidelines is granted; all other rights reserved.

- This work represents the view of the authors and does not necessarily represent the view of IBM.

- IBM, IBM logo, ibm.com, and WebSphere, are trademarks of International Business Machines Corporation in the United States, other countries, or both.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

- Other company, product, and service names may be trademarks or service marks of others.

- References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

- INTERNATIONAL BUSINESSMACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
References

Questions

Mail your Suggestions to:

Subrata Modak <subrata@linux.vnet.ibm.com>,
Balbir Singh <balbir@linux.vnet.ibm.com>

➢ https://sourceforge.net/projects/ltp
➢ http://ltp.sourceforge.net/