Intelligent Failure Prediction on Linux Systems

Timo Jokiah, Sanil Kumar D
3rd LinuxCon Japan, Yokohama
2011-06-02
For a crash-less world...!

- Exploring the possibility of unified Intelligent Failure Prediction for Linux Systems

and

- Visualizing the huge scope and growth for Linux in the market
Cost of a failure!
What is failure prediction?

Get the information in advance on any abnormal behavior of a system parameter which can lead to the system failure.

Failure = System is unable to provide the intended threshold result!
Failure predictions around us!

- Mobile Battery Failure
- Self-Monitoring, Analysis, and Reporting Technology (SMART) in disk drives
- Failure prediction for power transformers
- Nuclear Reactors
- Predictions for electronic/integrated circuits
- Safety critical system failure indication...

...and...most of the systems provide warning and alerts...!
What is this paper all about…?!

- Propose the idea and feasibility of Intelligent Failure Prediction on Linux as a unified solution
  - Some of the current prediction features
  - IFP Architecture
  - Failure Symptoms and Processing

- Scope and Future
Intelligent Failure Prediction

Key Technology / Research Need:
- Deep Check of OS and Algorithm to predict and handle the failure
- Kernel Development tuning to handle the failure
- Unified IFP Solution in Kernel

Key Value:
- Zero Down Time
- High Competitive Feature for industry from Linux
- Business Impact on QoS and OPEX
## Failure Prediction Algorithms

<table>
<thead>
<tr>
<th>Type Of Data</th>
<th>Approach / Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Log Files</td>
<td>SVM(Support Vector Machine)</td>
</tr>
<tr>
<td>Failure Log</td>
<td>Spherical Covariance &amp; Stochastic Model</td>
</tr>
<tr>
<td>Error Logs</td>
<td>SEP (Standard Error Prediction)</td>
</tr>
<tr>
<td>Failure Log</td>
<td>FT-Pro</td>
</tr>
<tr>
<td>Log</td>
<td>Semi Markov</td>
</tr>
<tr>
<td>Log Files</td>
<td>Cox Proportion Model</td>
</tr>
<tr>
<td>RAS Event Logs</td>
<td>Customized Nearest Neighbour</td>
</tr>
<tr>
<td>Monitoring</td>
<td>FFP (Failure Filtering)</td>
</tr>
<tr>
<td>Sensor And Failure Information</td>
<td>RBF(Radial Base Function)</td>
</tr>
<tr>
<td>RAS Event Logs</td>
<td>Dynamic Meta</td>
</tr>
<tr>
<td>RAS Event Logs</td>
<td>Learner</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type Of Data</th>
<th>Approach / Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAS Event Logs &amp; Error Logs</td>
<td>Meta Learner</td>
</tr>
<tr>
<td>Event Log</td>
<td>UBF(Universal Base Function)</td>
</tr>
<tr>
<td>Event Log, Sar Data, Node Topology</td>
<td>Rule Based Model Time Series, Rule Based, Bayesian Network</td>
</tr>
<tr>
<td>Quantum Smart Dataset</td>
<td>Naive Bayes Em</td>
</tr>
<tr>
<td>Failure Data</td>
<td>Weibull Distribution</td>
</tr>
<tr>
<td>Event Log</td>
<td>Multivariate Statistical Techniques</td>
</tr>
<tr>
<td>Time To Failure Data</td>
<td>ER Algorithm</td>
</tr>
<tr>
<td>Error Logs</td>
<td>DFT</td>
</tr>
</tbody>
</table>
## Tools Available…

<table>
<thead>
<tr>
<th>Tool</th>
<th>Key Features</th>
<th>Key Data</th>
</tr>
</thead>
</table>
| monit | Utility for managing and monitoring processes, files, directories and devices on a Unix system. Monit conducts automatic maintenance and repair and can execute meaningful causal actions in error situations. E.g. monit can start a process if it does not run, restart a process if it does not respond and stop a process if it uses too much resources. You may use monit to monitor files, directories and devices for changes, such as timestamps changes, checksum changes or size changes. | cpu load, Memory usage, swap usage, Process state, file size, inode usage, permissions, timestamps, checksum.  
- The monit monitors these parameters and also logs in syslog when a configured threshold value is met. |
| linux-pptools | This is a toolset designed to adjust process's parameters in modern linux system                                                                                       | process's scheduler, real-time priority, max and min priority |
| dstat  | dstat is a versatile replacement for vmstat, iostat and ifstat. Dstat overcomes some of the limitations and adds some extra features.                                                             | cpu load, Memory usage, paging, locks, disk statistics, interrupts, network statistics. |
## Tools Available...(contd...)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Key Features</th>
<th>Key Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysrq</td>
<td>Proc entry which can fetch information from running kernel.</td>
<td>locks, stack, memory info, process states.</td>
</tr>
</tbody>
</table>
| servicelog | servicelog is a database intended to store log entries relevant to system serviceability, | Indications on:  
- Serviceable events, including device failures that require the failing device to be replaced.  
- Informational entries relevant to system service  
- Repair actions have taken place, such as part replacement  
- Notifications of the availability of dump data |
| top    | The top program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of tasks currently being managed by the Linux kernel. | cpu load, Memory usage, paging statistics, swap usage, Process states etc.                  |
| vmstat | Report virtual memory statistics                                             | processes, memory, paging, block IO, traps, and cpu activity.                                 |
Key Challenges

- **Limited Prediction**
  - Less Coverage of scenarios
  - Less number of algorithms integrated
- **Scattered**
- **Less Intelligence...!**

Solution: Unified Intelligent Failure Prediction
IFP : Architecture

UI and Configuration → Deep check → Data Acquisition → Analyse & Predict → Notification & Action Handlers

Parameter DB → Deep check

Benchmarking & Trend → Data Acquisition

Logging → Analyse & Predict

Notification & Action Handlers → Notification

User Application → Logging → Notification

Object Value Tracking → Probe Points → Prediction Module

Linux Kernel

Drivers

Hardware

Multilevel Algorithms Need to considered for IFP
Predictive Analysis with Symptom Data

- Use to predict future trends and behavior patterns
- Statistical analysis that deals with extracting information from data
- The core: Capturing relationships between explanatory variables and the predicted variables from past occurrences
- The accuracy and usability of results will depend greatly on the level of data analysis and the quality of assumptions

Specific way of processing the data
- Watermark / threshold based
  - Analyzing the symptoms
    - loadavg (1min) > 4 then alert
    - loadavg (5min) > 2 then alert
    - memory usage > 75% then alert
    - swap usage > 25% then alert
    - cpu usage (user) > 70% then alert
    - cpu usage (system) > 30% then alert
    - cpu usage (wait) > 20% then alert
    - space usage > 80% for 5 times within 15 cycles then alert
    - space usage > 99% then stop
    - inode usage > 30000 then stop
    - inode usage > 99% then stop
Failures on Linux

- Out of memory
- I/O error
- Service unavailable
- File system corruption
- Deadlock
- No space on device
- CPU overload
- kernel panic
Symptoms for failure...

- Frequent Swapping
- Reducing Free Memory
- Network Connection Loss
- Reducing Disk Space
- Hardware Resource Busy
- Increasing CPU Load
- Increased Lock Contention
- Increasing IO Time
- ...and more!
Advanced Linux System Data (ALSD)

- Deep check symptom points
  - Transition point analysis
  - Various response profiling (interrupt, stack layers, ipc)
  - Custom Probe Points

- Live Kernel Performance Benchmarking and Trend Analysis

- BSP and Driver Level Probe point interfaces

- Supporting Features
  - Flight Recorder
  - Hot patching
  - Live Debugging
Use cases in nutshell

- Normal
  - If value >=<
  - If value <>
  - If value A or B

- Trend
  - If value >=< for 3 continuous times over x time over a sampling of y
  - If value <> for once over x time over a sampling of y
  - If value A or B for 2 times over x times over a sampling of y

- Watermark Based
  - value_lower_water_mark
  - value_upper_water_mark
  - value_optimum_threshold

- All configurable user inputs
- Very huge scope of customization scenarios with lot of values
Where are we now…?

- The research has just started

- We plan to have unified architecture and bring all the available and new prediction methods under intelligent failure prediction (integrated or provide method to integrate seamlessly)

- In Parallel, prototyping with currently available tools for evaluation of the current situation.

- In coming months, we plan to have a prototype with multiple tools and certain new kernel parameters added

- Planning for open source initiative on this area
It’s just a beginning!

- Strong Collaboration and Research Needed
- Integration of prediction algorithms
- New Prediction Models
- Kernel fine tuning and restructuring
  - More Symptom Probe points
  - Architecture to make inbuilt failure prediction
- Association of Live Debugging and Maintenance Algorithms
Value and Future

- Prevention of a failure is invaluable!
- Highly predictable Linux
- Linux to safety critical and real time systems strongly!

Failure = System is unable to provide the intended threshold result!

ROI

- Safety Critical Market
- Time to market
- Zero Down Time
- Professional implementation
- ROI
- Predictable OS Platform
- Maintenance efficiency
- High Competency
- Low CAPEX and OPEX
...and...WE CAN DO IT!

- Like minded people
- Experts
- LF Workgroup
- Industry collaboration

......all for......

ZERO DOWN TIME LINUX!
Thank You...
...for your time and participation...

Timo Jokiaho:
30+ years experience in embedded systems in various industries. Currently leading the software technology planning at Huawei (Munich, Germany), focusing on Base Platforms (OS, Virtualization, HA and O&M), also for terminal software platforms (Android and MeeGo). Previously lead Strategy, Technology & Architecture work for network element platform development at Nokia and NSN and leading several R&D and business teams to develop equipment for communication, security and maritime navigation industry. First and present chairman of SCOPE Alliance. Was president of SAF and chair for technical working group.

Sanil Kumar D:
Leader of Architecture Team for Linux Domain (Bangalore, India). 11+ years experience in Embedded Systems and Linux. Experience in Kernel and Driver Design and developments for various hardware platforms. Several papers and presentations at Huawei technology events in Linux Domain (pNFS on Linux, Non Functional Design, Multicore and Linux Optimization).

Timo Jokiaho : timo.jokiaho@huawei.com  Sanil Kumar D : sanil@huawei.com