Intelligent Failure Prediction on Linux Systems

Timo Jokiaho, Sanil Kumar D

3rd LinuxCon Japan, Yokohama

2011-06-02

www.huawei.com

For a crash-less world…!

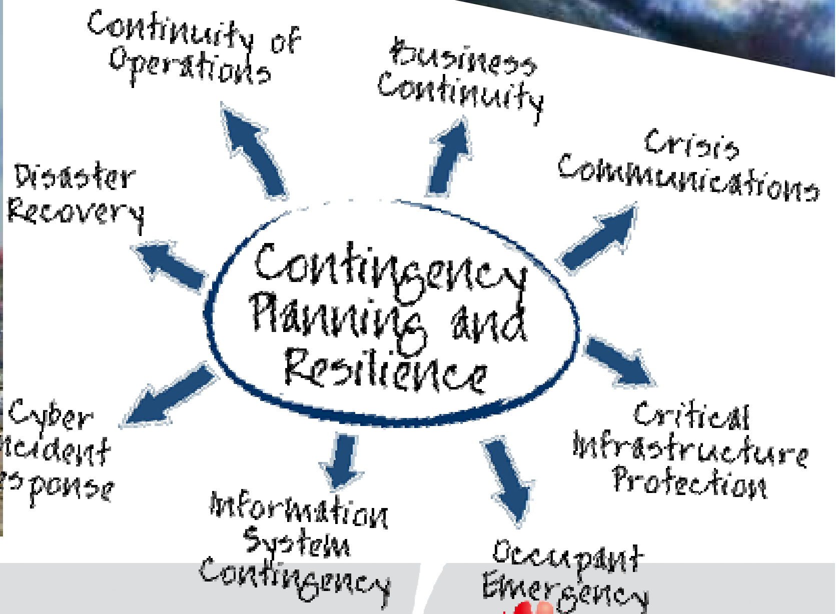
```
eax: 00000000   ebx: 00000000   ecx: 00000020   edx: 00000000
esi: c2384c00   edi: 00000004   ebp: c2384c00   esp: c2197e08
ds: 007b   es: 007b   fs: 00d8   gs: 0033   ss: 0068
Process init (pid: 953, ti=c2197000 task=f749f840 task.ti=c2197000)
Stack: c0462048 00000000 00000020 00000004 c2384c00 fffffff4 c2384c00 c05c9015
       f749f840 00000020 f7461400 c2384c00 c2197edc 0000000c c05ca7b0 0000000c
       c063bc00 f7461400 00000001 0000000c c05ae235 00000010 000003b9 ffffffff
Call Trace:
 [<c0462048>] krealloc+0x24/0x60
 [<c05c9015>] netlink_realloc_groups+0x59/0xcd
 [<c05ca7b0>] netlink_bind+0x4e/0x11d
 [<c05ae235>] sys_bind+0x75/0xa0
[<c0486de9>] d_instantiate+0x5b/0x5f
<c0618df6>] do_page_fault+0x0/0x69f
c06198f9>] do_page_fault+0x303/0x69f
c05ae9c2>] sys_socketcall+0x93/0x261
04820cb>] do_fcntl+0xd2/0x249
04822bf>] sys_fcntl64+0x7d/0x85
04d92>] syscall_call+0x7/0xb
================
2 00 34 00 40 02 00 75 03 8b 52 0c 8b 02 04 c8 b8 00 00 00 00 0f 49 d0 09
01 42 03 f0 10 74 34 e8 c2 ff ff ff 85 c0 75 04 <0f> 0b eb fe 8b 40 10 85
 0f 0b eb fe 8b 10 f6 c6 0c 74
472fb1>] ksize+0x10/0x3e SS:ESP 0068:c2197e08
```

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

and

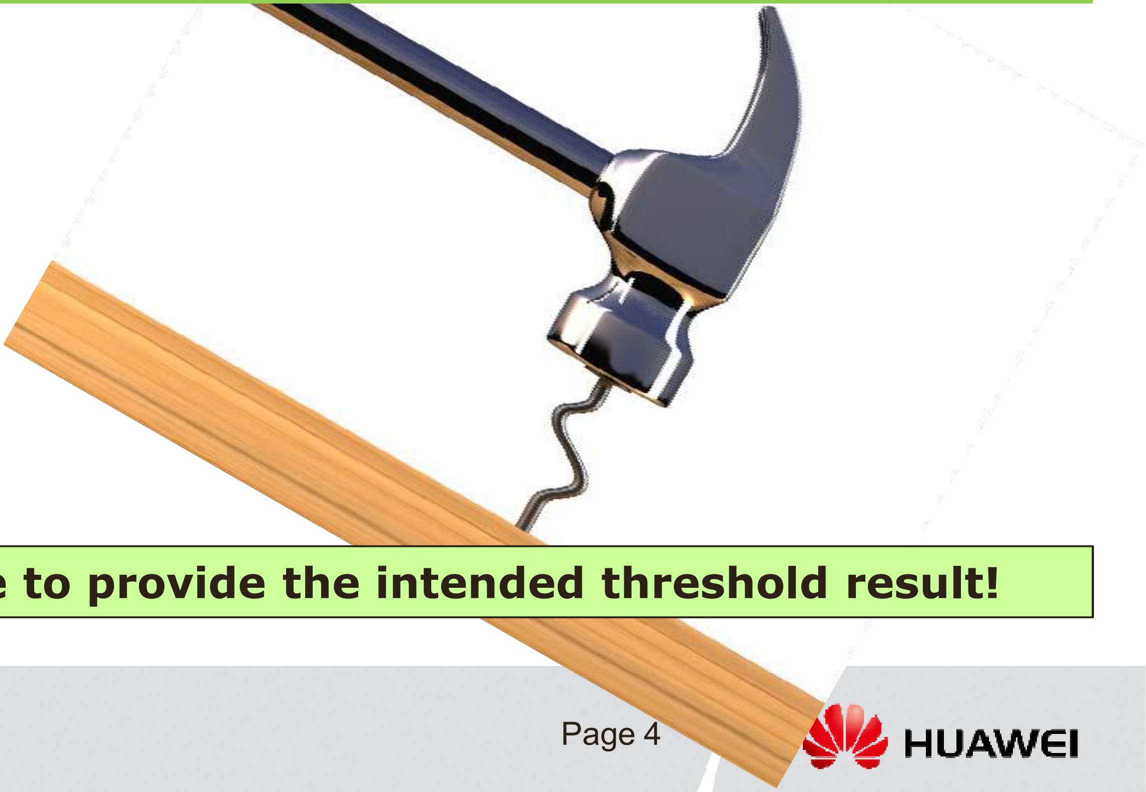
- Visualizing the huge scope and growth for Linux in the market

Cost of a failure!



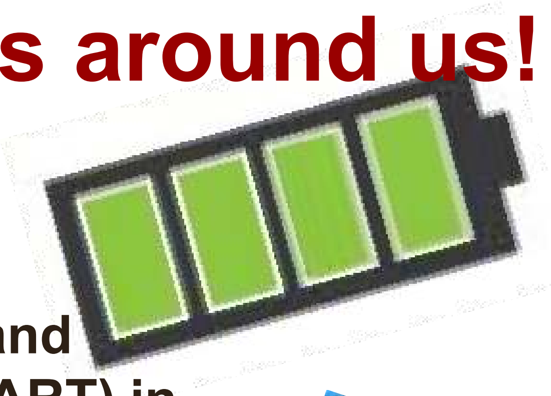
What is failure prediction?

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

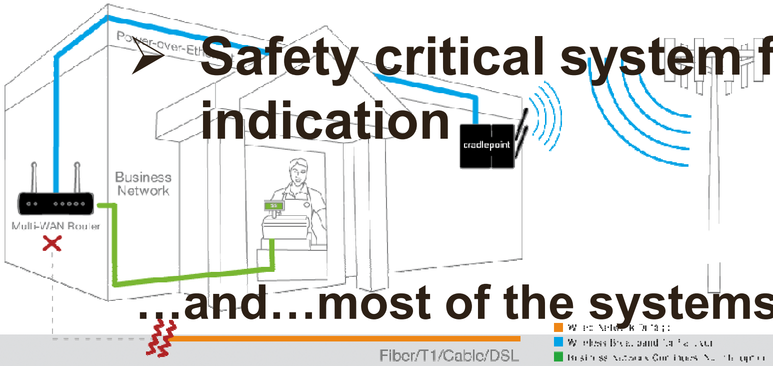


Failure = System is unable to provide the intended threshold result!

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



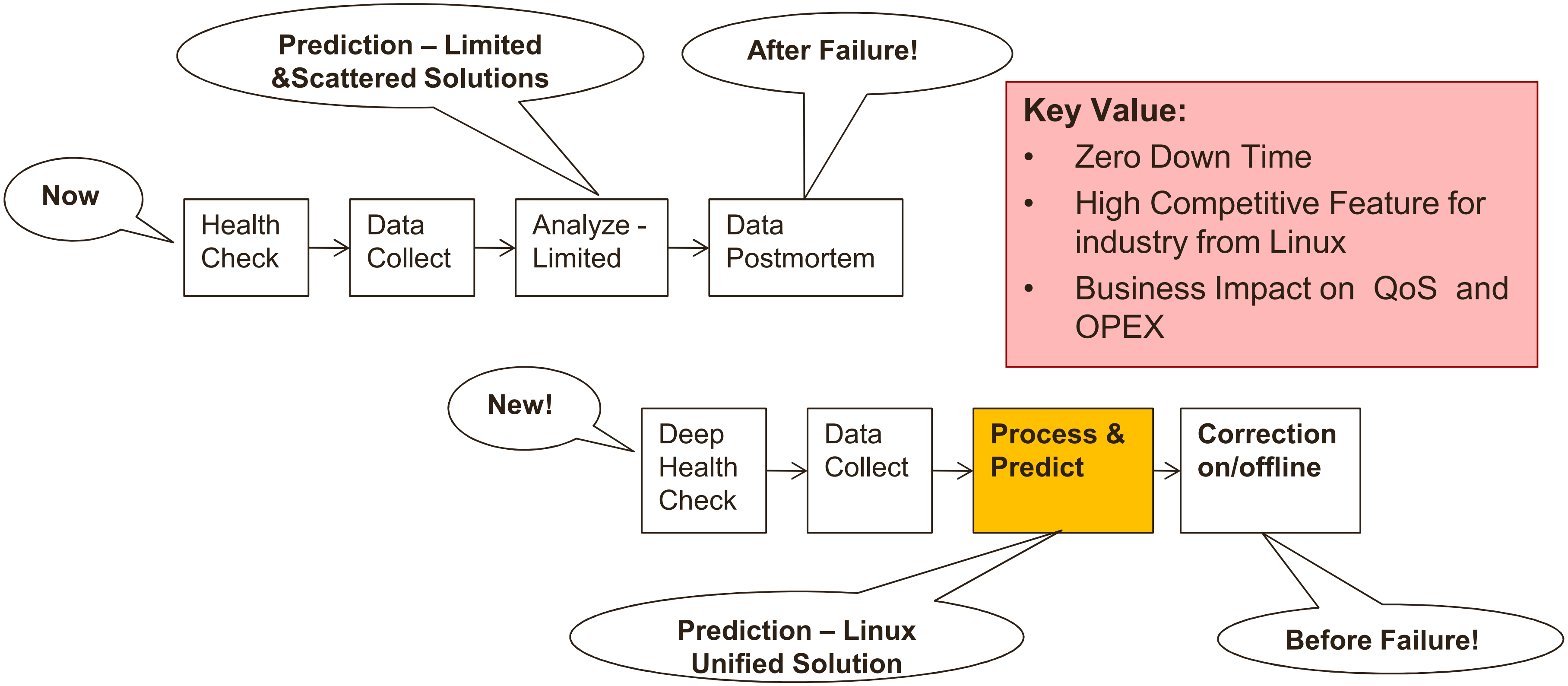
…and…most of the systems provide warning and alerts…!

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



Now

Health Check → Data Collect → Analyze - Limited → Data Postmortem

Prediction – Limited &Scattered Solutions

After Failure!

New!

Deep Health Check → Data Collect → **Process & Predict** → **Correction on/offline**

Prediction – Linux Unified Solution

Before Failure!

Key Value:

- Zero Down Time
- High Competitive Feature for industry from Linux
- Business Impact on QoS and OPEX

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

Failure Prediction Algorithms

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

Type Of Data	Approach / Algorithm
RAS Event Logs & Error Logs	Meta Learner
Event Log	UBF(Universal Base Function)
Event Log, Sar Data, Node Topology	Rule Based Model Time Series, Rule Based, Bayesian Network
Quantum Smart Dataset	Naive Bayes Em
Failure Data	Weibull Distribution
Event Log	Multivariate Statistical Techniques
Time To Failure Data	ER Algorithm
Error Logs	DFT

Tools Available…

Tool	Key Features	Key Data
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, cheksum. - The monit monitors these parameters and also logs in syslog when a configured threshold value is met.
linux-ptools	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…)

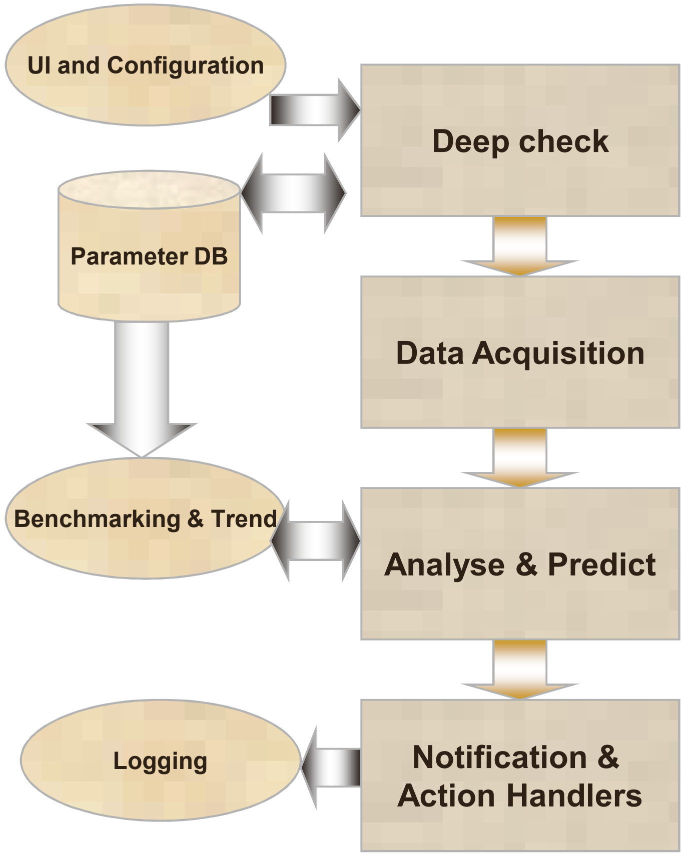
Tool	Key Features	Key Data
iostat	Report I/O statistics	cpu, I/O, disk statistics.
sysrq	Proc entry which can fetch information from running kernel.	locks, stack, memory info, process states.
servicelog	servicelog is a database intended to store log entries relevant to system serviceability,	Indications 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.

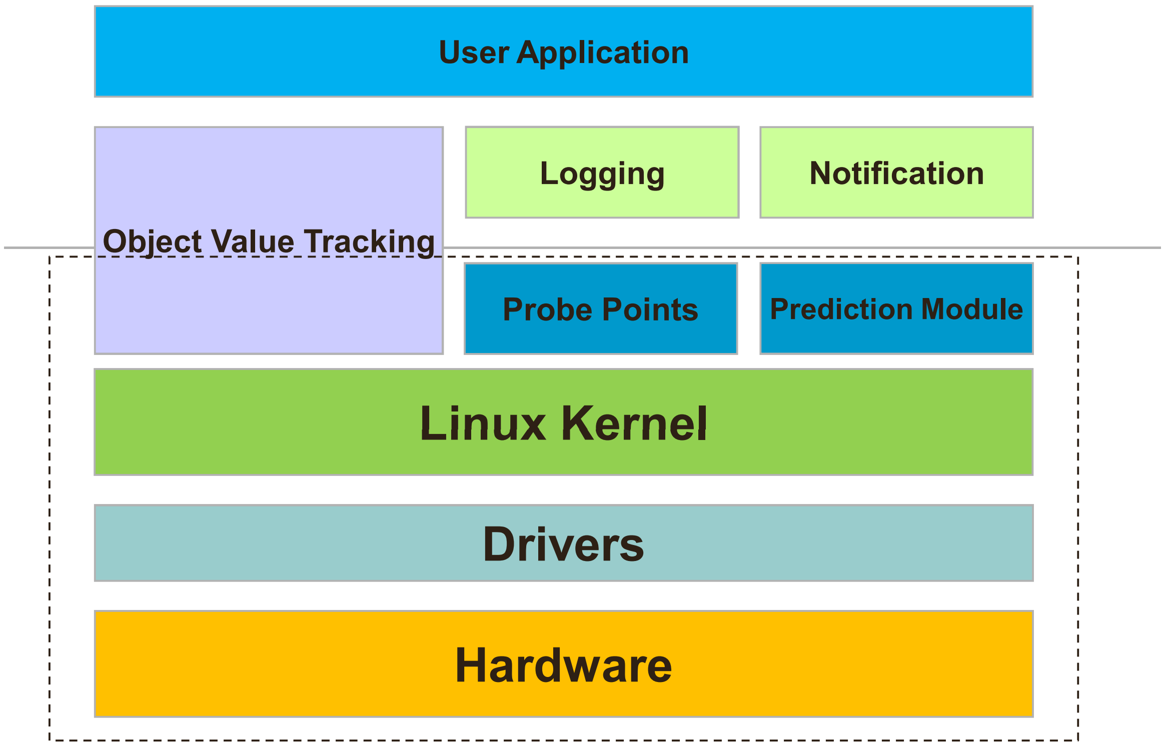
Key Challenges

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

Solution : Unified Intelligent Failure Prediction

IFP : Architecture





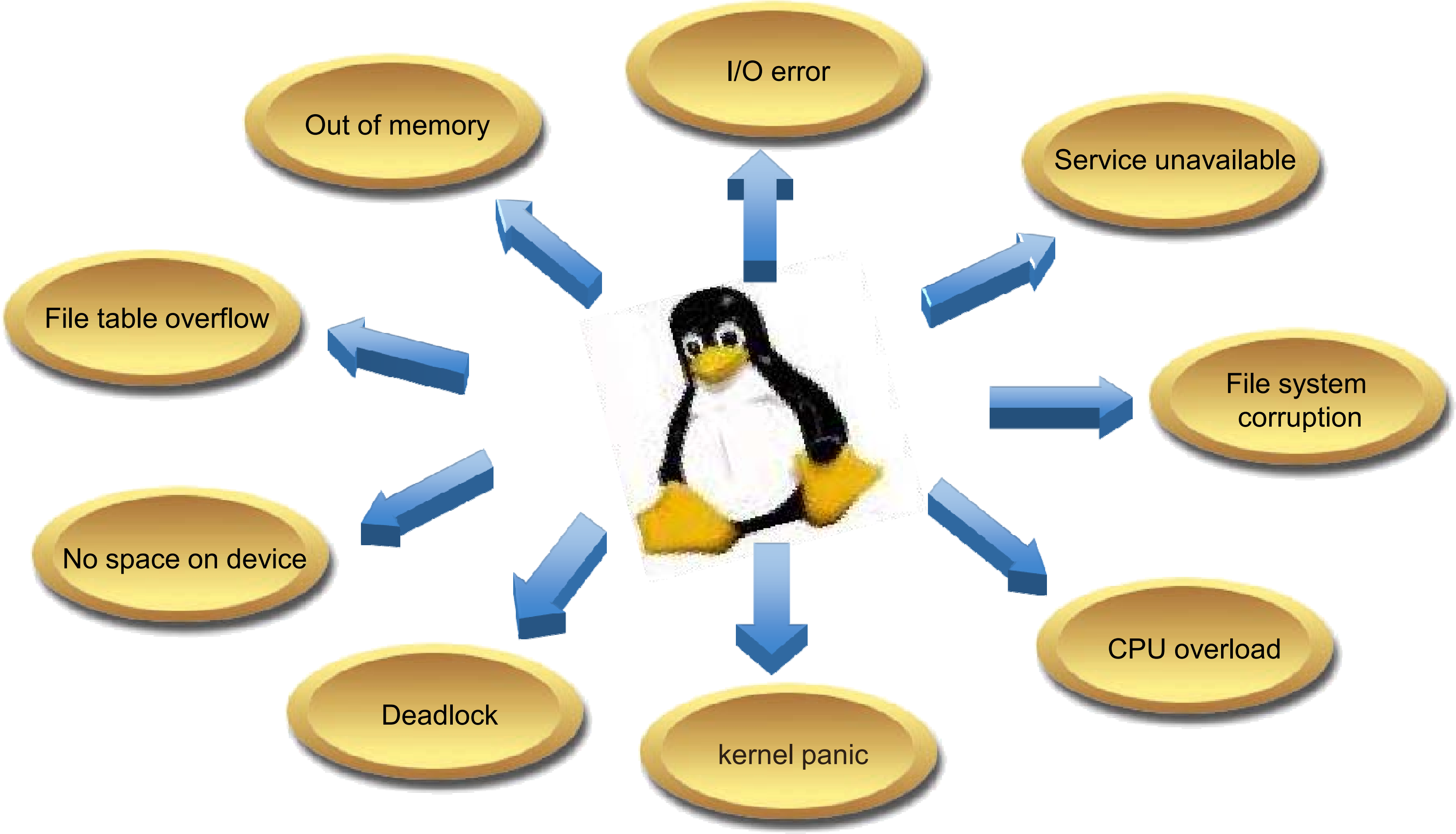
Multilevel Algorithms Need to considered for IFP

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 alert*
  - *inode usage > 99% then stop*

Failures on linux



Symptoms for failure…

Reducing Free Memory

Frequent Swapping

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

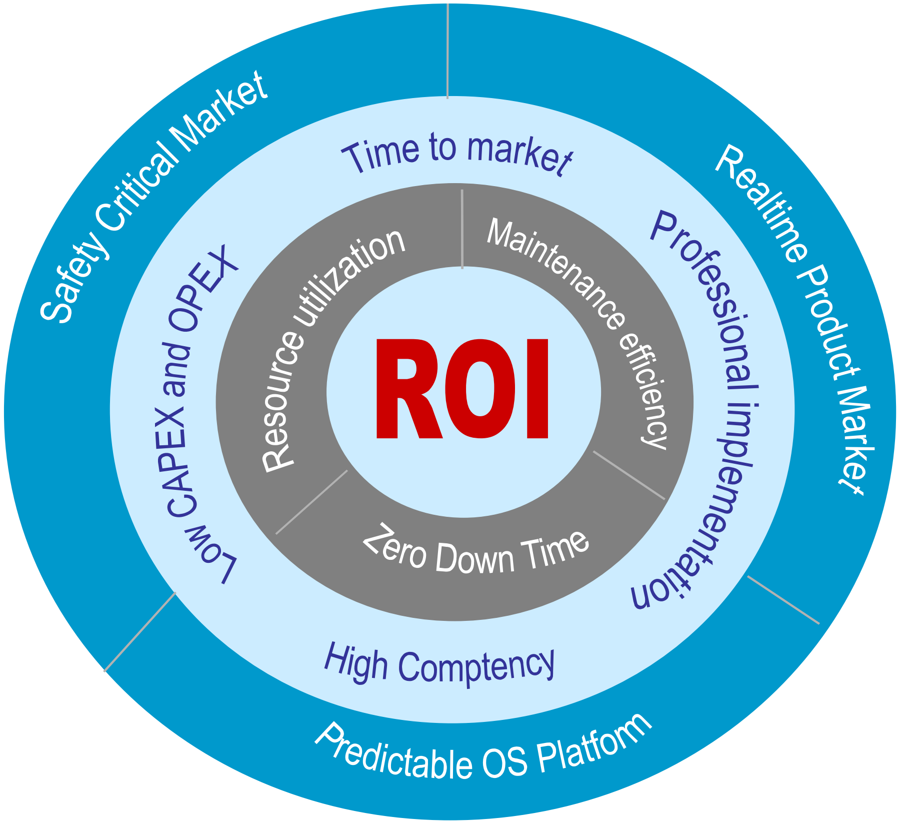
It’s just a beginning!

- 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

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

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



…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