Fujitsu's Activities for Improving Linux as Primary OS for PRIMEQUEST

● Hironobu Ishii

Linux is open source software that has evolved rapidly owing to the development, tests, and bug fixes conducted by various volunteers in the Linux community. Fujitsu places Linux as a primary OS for IA servers that cover a range of needs from volume-related to mission-critical-related ones. Fujitsu is actively participating in the Linux community and helping to enhance Linux. This paper describes four features being developed by Fujitsu, in conjunction with the Linux community, after the release of the kernel 2.6.18 that is the base kernel of Red Hat Enterprise Linux 5. These are 1) the Cgroup which manages system resources, 2) tracing features to improve the visibility of system activities, 3) scheduler improvement for fairness and responsiveness, and 4) utilizing the Machine Check Architecture (MCA) Recovery, which has been introduced to the Xeon 7500 series, to minimize the effects of hardware failures and recover from failures.

1. Introduction

Linux is open source software that has evolved rapidly owing to the development, tests and bug fixes conducted by various volunteers in the Linux community. There has been a swift shift from mainframe and UNIX servers to IA servers in the recent server market. Thus, Fujitsu has placed Linux as a primary OS for IA servers that cover a range of needs from volume-related to mission-critical-related ones and has been actively participating in the Linux community. Mission-critical use to which PRIMEQUEST is applied requires functionality and reliability comparable to those of mainframe and large UNIX servers. Therefore, in addition to participating in the Linux community, Fujitsu forged a strategic partnership in 2003 with Red Hat, which is the biggest Linux distributor and is also focused on mission critical aspects, to work on improving Linux OS.

This paper presents the following four functions which Fujitsu has realized with Linux kernel 2.6.18—the base kernel of Red Hat Enterprise Linux 5 (RHEL5) and later versions. Fujitsu has achieved this by playing an active part in the Linux community to improve mission-critical systems.

- Cgroup (for managing system resources)
- Tracing features (for improving the visibility of system activities)
- Scheduler improvement (for fairness and responsiveness)
- Machine Check Architecture (MCA) Recovery of the Xeon 7500 series (for minimizing the effects of hardware failures and recovery from failures)

2. Fujitsu's activities in relation to Linux

Red Hat's development model of RHEL and Fujitsu's activities are shown in **Figure 1**.

In the functional development of RHEL, the following three development phases have the primary role:

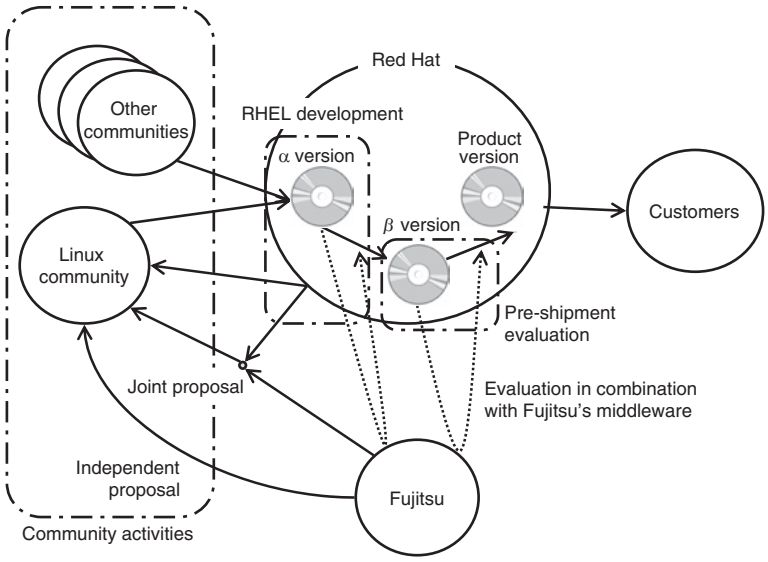

Figure 1
Development model of RHEL and Fujitsu's activities.

1) Linux community

In the uppermost-stream Linux community, many volunteers engage in development, reviews and tests to bring out Linux's standard features.

2) RHEL development

After sources developed by various communities are integrated into Red Hat, plenty of time is taken conducting tests to apply the necessary bug fixes and incorporate the required features that came in late to communities.

3) RHEL pre-shipment evaluation

Red Hat not only asks server vendors, ISVs and IHVs to evaluate the β version but also conducts tests in-house to improve the quality to be ready for shipment.

Fujitsu's activities in the development phases mentioned above are as follows.

1) Linux community

Fujitsu helps to enhance quality and develop features essential to support mission-critical systems.

2) RHEL development

Fujitsu makes use of its cooperation with Red Hat to help incorporate features into and enhance the quality of RHEL. As a measure for quality enhancement, Fujitsu develops a test set to prevent the recurrence of failures generated at Fujitsu's customers and offers it as the standard test set of Red Hat.

3) RHEL pre-shipment evaluation

Verification is conducted from the viewpoint of evaluation in combination with Fujitsu's middleware.

As described above, Fujitsu is active in each of the RHEL development phases with a particular focus on activities in the Linux community. This is because Fujitsu believes the quickest way to enhance the functionality and quality of Linux is to conduct activities in the Linux community, which is its source. Past examples of Fujitsu's activities include the development of the dump function, enhancement of MCA for Itanium 2 and development of udev and hot plug functions intended for RHEL4 and 5.[1)]

3. Cgroup

Cgroup is an abbreviation of Control Group, also known as the resource management function, which divides processes that run on one OS into several groups to control the resources available to the individual groups. This feature is expected to be implemented on kernel 2.6.24

and later in the Linux community and offered as a new function of RHEL6. An overview of Cgroup is shown in **Figure 2** and the Cgroup interface provided by the kernel is offered as a virtual file system. Directory creation for this virtual file system is mapped to group creation. Under each directory representing a group, virtual files for viewing and setting parameters with the same names as the configuration parameter names are automatically allocated. In relation to these virtual files, writing corresponds to parameter setting and reading to parameter viewing. The following subsections explain the resource management function with a focus on memory and CPU, which are the most frequently used resources.

1) Memory Cgroup[2),3)]

One major problem with the existing Linux has been that, when a group of batch processing programs and a group of online processing programs are run simultaneously in one system, the online processing programs' response sometimes becomes unstable. With Linux, there is a policy of maximizing performance by making effective use of memory and caching I/O data as much as possible. For this reason, a large volume of writing to files by batch processing is generated, which fills most of the available memory of the system with dirty pages (pages that cannot be released without writing the content back to the disk). When online processing programs request new memory allocation under these memory usage conditions, the OS reallocates memory to the online processing programs after reclaiming the memory in use. If dirty pages are to be recovered, however, the page content needs to be written back to the disk and memory recovery and reallocation take longer. This causes the response of the online processing programs to be unstable.

Memory Cgroup offers the following functions to solve the problem above.

- Limits the total amount of anonymous memory used by processes in the group (dynamic data area), memory for file mapping (execution code, static data, mmap processing) and file cache. When the usage has reached the limit, part of the memory is reclaimed and, if the page to be reclaimed is dirty, it is written back to the swap or file.
- Limits the total amount of anonymous memory used by processes in the group, memory for file mapping, file cache and swap usage. When the usage has reached

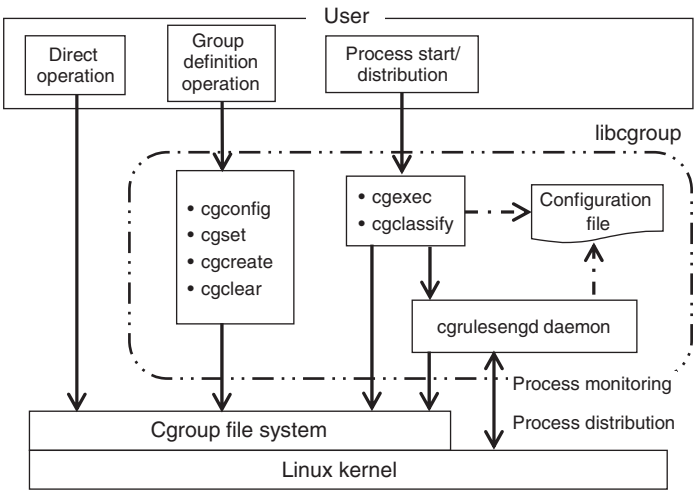

Figure 2
Cgroup overview.

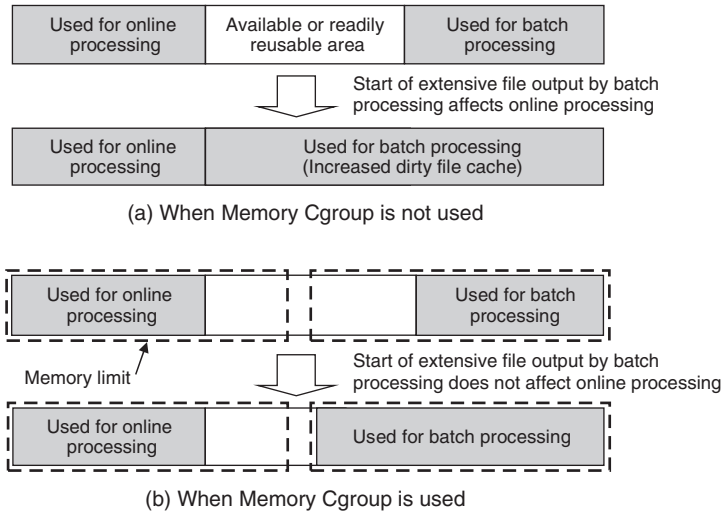

Figure 3
Advantage gained from Memory Cgroup.

the limit, part of the memory is reclaimed. If the usage has reached the limit but no more recoverable resource exists, a certain process in the same group may be forced to terminate by the Out Of Memory Killer (OOM-Killer).

- Processes that do not belong to any group are regarded as belonging to the Root Cgroup and control by the Cgroup does not affect these processes. The Cgroup group can be made into a hierarchical structure.

For example, the groups of online processing programs and batch processing programs can be defined as belonging to separate groups to specify the amount of memory or the total amount of memory and swap available to the respective groups. In this way, even if a large volume of file cache is used by batch processing, available memory for online processing programs or readily recoverable pages (pages that do not contain data to be written back to the disk) can be maintained. This prevents the response of online processing from deteriorating (**Figure 3**).

2) CPU Cgroup[4)]

CPU Cgroup is a function that manages CPU allocation to processes belonging to groups. CPU allocation to the individual group is represented by an integer called a share value. The share value of a group immediately after the creation of the CPU group is 1024. If three CPU groups A, B and C are created, for example, the share values of A, B and C immediately after the creation of the groups are:

A:B:C = 1024:1024:1024 = 1:1:1.

If the share values of A and B are subsequently changed to 4096 and 2048 respectively by writing to the virtual files for setting share values of the respective groups, the values become:

A:B:C = 4096:2048:1024 = 4:2:1.

This means that the group of processes belonging to A can acquire four times as much CPU time as the group of processes belonging to C, and the group of processes belonging to B can acquire twice as much CPU time as the group of processes belonging to C. However, the CPU time each process can acquire is affected by the number of active processes that belong to that group. The CPU share value of the Root Cgroup is fixed at 1024. For example, if the Root Cgroup has a process that consumes a large amount of CPU time in the previous example, CPU allocation to the individual CPU groups is as shown below with the Root Cgroup taken into

consideration:

A:B:C:Root = 4096:2048:1024:1024
= 4:2:1:1.

Since allocating all processes to groups other than the Root Cgroup needs much effort, a group of light processes such as that for monitoring is allocated to the Root Cgroup. Because CPU allocation of the Root Cgroup cannot be changed, allocating a group of busy processes to the Root Cgroup may make it impossible to ensure the CPU time for a light process for monitoring, adversely affecting the monitoring operation. For this reason, it is desirable to allocate a group of busy processes to a group other than the Root Cgroup.

This CPU Cgroup function is indispensable for controlling the allocation of CPU time between guest OSs also in the mechanism called Kernel-based Virtual Machine (KVM), which realizes a virtual machine on the Linux kernel.

4. Tracing features

Fujitsu's concept of a mission-critical system is one where it should be possible to find out the cause of any problem generated without conducting a reproduction test. To that end, we believe a function is necessary to continuously record the behavior of the system in operation, as a flight recorder in an aircraft does. With the existing Linux kernel, this type of analysis required the addition of message output by modifying the kernel source, followed by recompilation to use for reproducing the phenomenon. However, this method has the following problems.

- The addition of message output significantly changes the system operation timing, which hinders the reproduction of the problem.
- A system reboot is necessary after the source modification and recompilation and the problem cannot be investigated in a system in operation. In addition, rebooting is likely to hinder the reproduction of the problem.

To address these challenges, the introduction of various forms of tracing features has been discussed in the Linux community. Such introduction can be roughly classified into the following two types.

1) Dynamic probe method

A patch is applied dynamically to the machine instruction for the routine to be observed, the control is transferred to the instruction code inserted as a probe and then control is brought back to the original routine to be observed. Instructions inserted as the probe are called a handler and dynamically loaded as a kernel module.[5)–7)] There is a tool called systemtap to facilitate the creation of the handler.[8)] This automates the process from the creation of a kernel module to be used as a handler with a scripting language called tapset through the loading of the module to the extraction of the measurement result.

2) Static probe method

Instructions for the probe are embedded in the kernel execution code in advance. Methods currently implemented include the ftrace which probes the kernel function's entry and exit by using a compiler profiling feature[9),10)] and explicit logging of points to trace to the kernel sources, such as event tracing and tracepoint.[11),12)]

Fujitsu first implemented a flight recorder function (continuous tracing function) via the systemtap script. Because it is a script, it had some performance problems and there was a limit to the number of trace points that could be extracted. Next, Fujitsu attempted direct probing of static probe-based tracepoints by creating a kernel module.[13)] A flight recorder of this type is already provided for RHEL5 to customers who bought RHEL support from Fujitsu, and has proved itself useful for expediting problem solving. This method, however, had a problem of high cost when provided continuously and Fujitsu worked with the community to improve ftrace, which is higher-functionality, higher-performance static tracing infrastructure.[14)–17)] With RHEL6, a flight recorder function via ftrace

is expected to be available.

5. Scheduler improvement

In the process of tuning performance to build a system with a stable transaction performance in the order of milliseconds,[18] Fujitsu found the following two problems in the Order One Scheduler (O(1) Scheduler), which is the scheduler of RHEL5, and worked to improve them.

1) CPU resource allocation equalization

This improvement has been made to maintain equal CPU resource allocation to processes and threads under a high load condition. With RHEL5 O(1) scheduler, CPU time was preferentially allocated to interactive processes, which are often in sleep, to ensure a certain level of response. However, a problem has been found that, in a system with many interactive processes, CPU resources are allocated only to some of the interactive processes and the overall performance of the system deteriorates. The solution to this problem has been discussed in the community.[19] The community was in a phase in which a shift from the O(1) Scheduler to the new Completely Fair Scheduler (CFS) was taking place and the discussion was incorporated into the improvement of the CFS.[20] This improvement is expected to be applied to RHEL6. Regarding RHEL5, the result of this discussion has been incorporated as a modification intended for the O(1) Scheduler. These have equalized CPU resource allocation and allowed stable performance without significantly compromising the response of interactive processes.

2) Response improvement

The existing Linux scheduler is scheduled to have a greater throughput by improving the CPU cache hit rate. Accordingly, measures were taken so that when a process that started to run on a CPU and once went to sleep was to wake up from the sleep state, it would be scheduled on the same CPU whenever possible. For this reason, in a large-scale symmetric multiprocessor (SMP), the system performance sometimes peaked even when not all CPUs were exhausted, depending on the loading conditions. To address this problem, Fujitsu proposed a new scheduling technique, which has been adopted.[21] Specifically, a parameter called sched_relax_domain_level, which modifies the behavior of the system group scheduler, has been introduced. This parameter tunes the range of search for a CPU that is idle (without a process to execute) when waking up the process. The range of search for an idle CPU can be tuned in six stages between 0, which means do not search for another idle CPU, and 5, which means search for idle CPUs in the entire system.[22] This function has made it possible to maximize the capacity of CPUs of a large-scale SMP system and maintain high response performance.

6. MCA Recovery

For the PRIMEQUEST 1000 series, which is presented in this Special Issue, the Xeon 7500 series CPUs (Intel development codename: Nehalem-EX) are used.[23] These are the first x86 CPUs equipped with the MCA Recovery function (**Figure 4**). This feature is equivalent to that implemented in the Itanium 2 processor, which has been used for the existing PRIMEQUEST 400/500 series. Further, Fujitsu has used its experience with the Itanium 2 version of PRIMEQUEST to help enhance the quality of this feature.

The Xeon 7500 series CPUs are provided with a mechanism for hardware failure self-diagnosis and error correction. To handle any hardware failure, this feature first attempts error correction on the hardware layer. If the correction succeeds, software processing continues and Corrected Machine Check Interrupt (CMCI) is generated to simply make a notification of the error correction. If the error cannot be corrected on the hardware level, a Machine Check Exception (MCE) is generated to request the OS to conduct a recovery process.

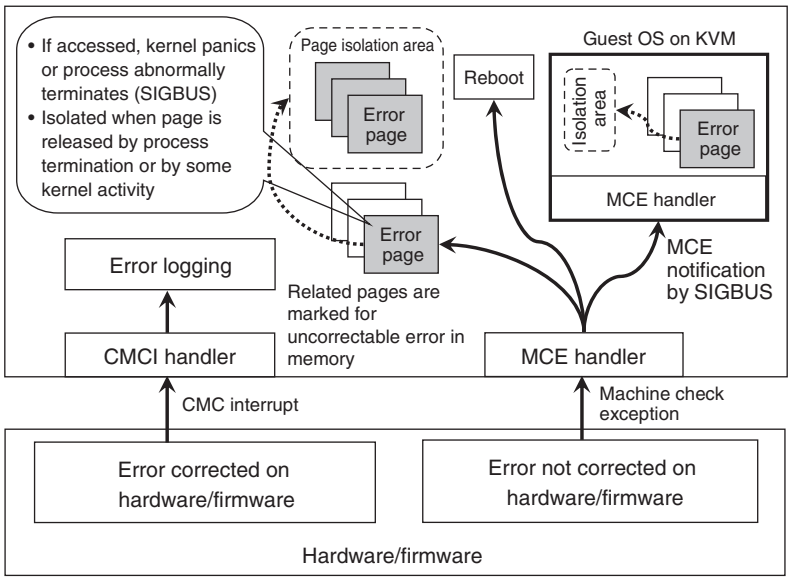

Figure 4
Concept of MCA Recovery.

When a CMCI is received, the OS only logs the error. When an MCE is received, the OS analyzes the error content and conducts a recovery process according to the error. Any uncorrectable error generated in the CPU is a fatal error and the OS that received the MCE panics. One example of an uncorrectable error that the OS attempts to recover is an ECC multi-bit error generated on the memory or cache. The memory controller of the Xeon 7500 series is equipped with a feature called memory scrubbing, which regularly accesses and diagnoses the memory. And if any uncorrectable ECC multi-bit error is detected by this diagnosis, the OS is notified of an MCE. The OS checks the content of the MCE and marks the related pages with an error. If these pages are not used, they are marked as unusable and isolated. If they are in use, the OS waits for them to be released and marks them as unusable for isolation at that time.

Fujitsu has helped enhance the quality of this feature by reviewing it after the release of the first patch from Intel and creating a module that emulates failures for debugging.[24)–27)]

7. Conclusion

Fujitsu believes that, to enhance the functionality and quality of Linux, it is important to continue to work in the Linux community, which is its source. This paper has described four functions that Fujitsu has helped to develop and enhance by participating in the Linux community: Cgroup, tracing features, scheduler improvement and MCA Recovery.

Of these functions, the tracing features and scheduler improvement have already made contributions to stable operation of various systems that use RHEL5. In addition, Cgroup, the flight recorder function (continuous tracing function) via ftrace, CFS scheduler and MCA Recovery function are expected to be offered with RHEL6. This will hopefully help customers' systems to operate even more stably.

Fujitsu is committed to remaining active in the community to further improve Linux.

Lastly, we would like to extend our gratitude to the members of the community who are striving to improve Linux together with us.

References

1) Norio Kurobane: Rapidly Growing Linux OS: Features and Reliability. *Fujitsu Sci. Tech. J.*, Vol. 41, No. 3, pp. 318–322 (2005).
2) KAMEZAWA Hiroyuki: Memory Resource Controller. *http://lxr.linux.no/linux+v2.6.35/Documentation/cgroups/memory.txt*
3) KAMEZAWA Hiroyuki: Memory Resource Controller. *http://events.linuxfoundation.org/images/stories/slides/jls09/jls09_kamezawa.pdf*
4) Paul Menage: CGROUPS. *http://lxr.linux.no/linux+v2.6.35/Documentation/cgroups/cgroups.txt*
5) Masami Hiramatsu: SystemTap How-to. *http://www.linuxfoundation.jp/jp_uploads/seminar20061109/MHiramatsu.pdf*
6) A. Mavinakayanahalli et al.: Probing the Guts of Kprobes. Proceedings of the Linux Symposium, Volume Two, pp. 101–115 (July 19–22, 2006). *http://www.linuxsymposium.org/2006/linuxsymposium_procv2.pdf*
7) Jim Keniston et al.: Kernel Probes (Kprobes). *http://lxr.linux.no/linux+v2.6.35/Documentation/kprobes.txt*
8) SystemTap. *http://sourceware.org/systemtap/*
9) Mike Frysinger: function trace guts. *http://lxr.linux.no/linux+v2.6.35/Documentation/trace/ftrace-design.txt*
10) Steven Rostedt: ftrace-Function Tracer. *http://lxr.linux.no/linux+v2.6.35/Documentation/trace/ftrace.txt*
11) Teodore Ts'o: Event Tracing. *http://lxr.linux.no/linux+v2.6.35/Documentation/trace/events.txt*
12) Mathiew Desnoyers: Using the Linux Kernel Tracepoints. *http://lxr.linux.no/linux+v2.6.35/Documentation/trace/tracepoints.txt*
13) Zhao Lei et al.: Flight Recorder: A Solution for Investigating Linux Kernel Accidents. *http://video.linuxfoundation.org/video/1646*
14) Li Zefan: tracing: Allow to disable cmdline recording. *http://git.kernel.org/linus/e870e9a*
15) Li Zefan: tracing: Convert some block events to DEFINE_EVENT. *http://git.kernel.org/linus/77ca1e0*
16) Lai Jiangshan: tracing:add trace_bprintk(). *http://git.kernel.org/linus/1ba28e0*
17) Lai Jiangshan: tracing: infrastructure for supporting binary record. *http://git.kernel.org/linus/1427cdf*
18) Fujitsu: Developing the Next-Generation "arrowhead" Trading System. Annual Report 2010, p. 59. *http://www.fujitsu.com/downloads/IR/annual/2010/all.pdff*
19) Satoru Takeuchi: [BUG] scheduler: strange behavior with massive interactive processes. *http://lkml.org/lkml/2007/3/26/319*
20) Ingo Molnar: CFS Scheduler. *http://lxr.linux.no/linux+v2.6.35/Documentation/scheduler/sched-design-CFS.txt* *http://people.redhat.com/mingo/cfs-scheduler/sched-design-CFS.txt*
21) Hidetoshi Seto: sched, cpuset: customize sched domains, core. *http://git.kernel.org/linus/1d3504f*
22) Simon Derr et al.: CPUSETS. *http://lxr.linux.no/linux+v2.6.35/Documentation/cgroups/cpusets.txt*
23) Motoyoshi Hirose et al.: PRIMEQUEST 1000 Series: High-Reliability, Mission-Critical Intel Architecture Server Supporting Social Infrastructure Systems. *Fujitsu Sci. Tech. J.*, Vol. 47, No. 2, pp. 192–206 (2011).
24) Hidetoshi Seto: x86, mce: don't init timer if!mce_available. *http://git.kernel.org/linus/33edbf0*
25) Hidetoshi Seto: x86, mce: sysfs entries for new mce options. *http://git.kernel.org/linus/9af43b5*
26) Hidetoshi Seto: x86, mce:unify smp_thermal_interrupt. *http://git.kernel.org/linus/a65c88d*
27) Hidetoshi Seto: x86, mce:remove intel_set_thermal_handler(). *http://git.kernel.org/linus/8363fc8*

Hironobu Ishii
Fujitsu Ltd.
Mr. Ishii is currently engaged in Linux improvement activities and customer support operations.