特集

# PRIMEQUESTを支えるLinux OSの機能強化への取組み

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


### あらまし

Linuxはオープンソースソフトウェアであり，Linuxコミュニティにおける様々なボランティアの開発，テスト，障害修正活動により急速に進歩してきた。富士通は，Linuxをボリューム領域からミッションクリティカル領域までをカバーするIAサーバ用の主要OSとして位置付けて，積極的にLinuxコミュニティに参加し，機能強化に取り組んでいる。

本稿では，Red Hat Enterprise Linux 5のベースであるカーネル2.6.18版以降に追加・改善された機能の中から，富士通がコミュニティで機能強化や品質強化に参加した，Cgroup（システムの資源管理），トレース機能（システムの挙動の可視化），スケジューラ改善（システムの応答性改善），Xeon 7500番台のMCAリカバリ機能（マシンチェック機構を利用した影響範囲の限定と回復）について述べる。

### Abstract

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 server that covers 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 2.6.18 kernel 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 Recovery Architecture, which has been introduced to the Xeon 7500 series recently, to minimize the effects of hardware failures and recover from failures.



石井宏延（いしい　ひろのぶ）

プラットフォームソフトウェア事業本部Linux開発統括部 所属
現在，Linuxの改善活動と顧客サポート業務に従事。

## ま え が き

Linuxはオープンソースソフトウェアであり，Linuxコミュニティにおける様々なボランティアの開発，テスト，障害修正活動により急速に進歩してきた。近年，サーバ市場はメインフレームやUNIXサーバからIAサーバに急速にシフトしており，富士通はLinuxをボリューム領域からミッションクリティカル領域までをカバーするIAサーバ用の主要OSとして位置付け，積極的にLinuxコミュニティに参加している。また，PRIMEQUESTが適用されるミッションクリティカル領域では，メインフレームや大型UNIXサーバ並みの機能や信頼性が求められるため，Linuxコミュニティへの参加に加え，Linuxの最大手ディストリビュータでありミッションクリティカル指向も一致するRed Hat社と戦略提携を2003年に締結し，Linux OSの強化に取り組んでいる。

本稿では，富士通がミッションクリティカルシステムの強化をねらいとして，Linuxコミュニティで活動し，Red Hat Enterprise Linux 5（RHEL5）のベースであるLinuxカーネル2.6.18版以降で実現した以下の四つの機能を紹介する。

・Cgroup（システムの資源管理）
・トレース機能（システムの挙動の可視化）
・スケジューラ改善（システムの応答性改善）
・Xeon 7500番台のMCAリカバリ機能（マシンチェック機構を利用した影響範囲の限定と回復）

## 富士通のLinuxへの取組み

Red Hat社のRHELの開発モデルと富士通の取組みを**図-1**に示す。

RHELの機能開発では，以下の三つの開発フェーズが中心的役割を担っている。

(1) Linuxコミュニティ

最上流のLinuxコミュニティにおいて，多くのボランティアによって開発，レビュー，テストが行われLinux標準機能となる。

(2) RHEL開発

Red Hat社によって様々なコミュニティで開発されたソースを集約した後，長い時間をかけてテストを行い，必要な修正の取込みや遅れてコミュニティに入った必要な機能の取込みを行う。

(3) RHEL出荷前評価

Red Hat社は，サーバベンダ，ISV，IHVなどにベータ版の評価を依頼すると同時に，社内でもテストを行いながら出荷に向けて品質を高めていく。

上記の開発フェーズにおいて，富士通の活動を紹介する。

(1) Linuxコミュニティ

ミッションクリティカルシステムを支えるために不可欠な機能の開発，品質強化に参加している。

(2) RHEL開発

Red Hat社との協業を生かして，RHELへの機能取込みや品質強化に協力している。品質強化の施策として，富士通のお客様で発生した障害の再発を防

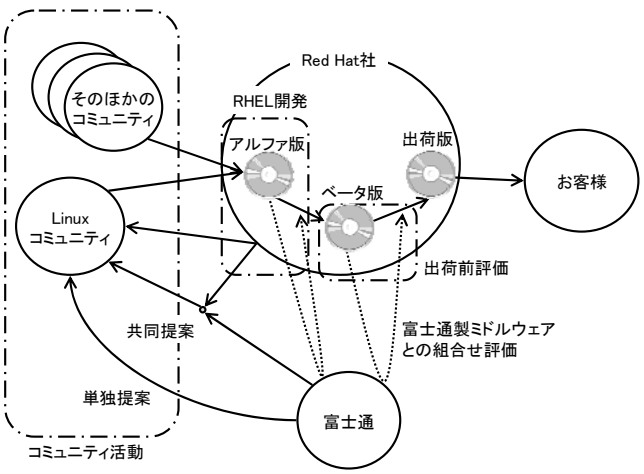


図-1　RHELの開発モデルと富士通の取組み
Fig.1-Development model of RHEL and Fujitsu's activities.

止するためにテストセットを開発して，Red Hat社の標準テストセットとして使ってもらう活動も実施している。

(3) RHEL出荷前評価

富士通製ミドルウェアとの組合せ評価の観点も加えて検証を行っている。

以上のように，富士通はRHEL開発の各フェーズで活動しているが，一番力を入れているのは，Linuxコミュニティでの活動である。Linuxの機能，品質を高めるためには，源流であるLinuxコミュニティで活動することが一番の近道であると信じるからである。富士通が取り組んだ過去の事例としては，RHEL4，5に向けて，dump機能，Itanium2用MCA強化，udev機能，ホットプラグ機能の開発に取り組んだ[1]。

## Cgroup

CgroupはControl Groupの略であり，資源管理機能とも呼ばれる。これは，一つのOSの中で動作するプロセスを，いくつかのグループに分けて，グループごとの利用可能な資源量を制御する機能である。この機能は，Linuxコミュニティのカーネルの2.6.24版以降に実装され，RHEL6の新機能として提供される見込みである。Cgroupの全体像は，**図-2**のようになっており，カーネルが提供するCgroupのインタフェースは，仮想ファイルシステムとして提供される。この仮想ファイルシステムに対しての，ディレクトリ作成操作はグループ作成の操作となる。グループを表わす各ディレクトリの下には，設定パラメータ名と同じ名前を持つパラメータ参照・設定用の仮想ファイルが自動的に配置される。この仮想ファイルに対する書込みはパラメータの設定，読出しはパラメータ参照の動きとなる。以降，最も使用頻度の高い資源であるメモリとCPUの資源管理機能に絞って説明を進める。

(1) メモリCgroup[2],[3]

従来のLinuxの大きな問題点の一つに，同じシステムの中で，バッチ処理を行うプログラム群と，オンライン処理を行うプログラム群を同時に実行した場合，オンライン処理プログラムのレスポンスが不安定になるという問題が発生する場合があった。Linuxでは，メモリを有効利用してI/Oデータを最大限キャッシュし，性能を極大化しようとする思想がある。このため，バッチ処理による大量のファイルへの書込みの発生により，システムの空きメモリの大部分がダーティなページ（ディスクへ内容を書き戻さないと解放できないページ）で埋め尽くされる。このようなメモリの使用状況で，オンライン処理プログラムが新たにメモリ割当てを要求した場合，OSは使用中のメモリを回収した後に，オンライン処理プログラムに再割当てを行うが，ダーティなページが回収対象になると，ページ内容のディスクへの書戻しが発生し，メモリ回収と再割当てに要する時間が長くなる。これによりオンライン処理プログラムのレスポンスが不安定となる。

メモリCgroupは以下の機能を提供することで，上記の問題を解決する。

・グループ内のプロセスが使用する無名メモリ（動

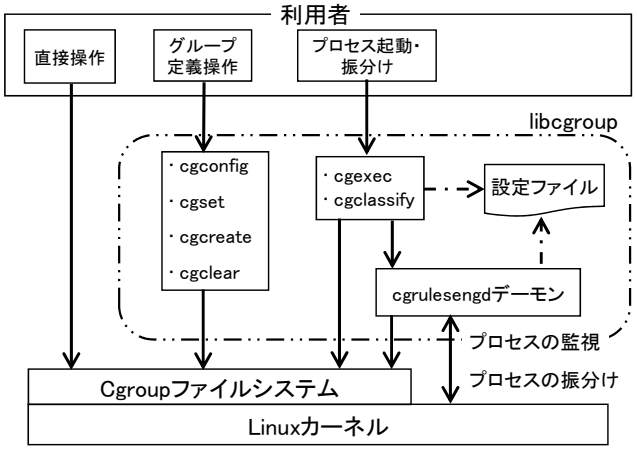


図-2 Cgroup概要
Fig.2-Cgroup overview.

的データ領域），ファイルをマップするためのメモリ（実行コード，静的データ，mmap処理），ファイルキャッシュの合計値を制限する。使用量が制限値に達した場合，一部のメモリが取り上げられ，取り上げるページが，ダーティな場合は，スワップあるいはファイルへ書き戻される。

・グループ内のプロセスが使用する無名メモリ，ファイルをマップするためのメモリ，ファイルキャッシュとスワップ使用量の合計値を制限する。使用量が制限値に達した場合，一部のメモリが回収される。制限値に達したが，これ以上回収できる資源が存在しない場合は，同じグループ内のいずれかのプロセスが，OOM-Killer（Out Of Memory Killer）により，強制終了されることがある。

・どのグループにも属さないプロセスは，Root Cgroupに属するとして扱われ，これらのプロセスに対しては，Cgroupの制御は及ばない。また，Cgroupのグループは，階層構造にすることが可能である。

例えば，オンライン処理プログラム群とバッチ処理プログラム群を別のグループに属すると定義し，それぞれのグループの使用可能なメモリ量やメモリ量＋スワップ量を設定することで，バッチ処理による大量のファイルキャッシュ利用が発生しても，オンライン処理プログラム用の空きメモリ，あるいは，すぐに回収可能なページ（ディスクへ書き戻すべきデータを保持していないページ）が維持できるため，オンライン処理のレスポンス悪化を防止できる（**図-3**）。

（2） CPU Cgroup[4]

CPU Cgroupは，グループに属するプロセスへのCPU配分を管理する機能である。各グループへのCPU配分は，シェア値という整数で表される。CPUグループを作成直後のグループのシェア値は，1024であり，例えば，A，B，Cの三つのCPUグループを作ったとすると，グループ作成直後のA，B，Cのシェア値は，

A：B：C＝1024：1024：1024＝1：1：1

となる。その後，A，Bのシェア値を各グループのシェア値設定用の仮想ファイルに対して，それぞれ4096，2048と書き込んで変更すると，

A：B：C＝4096：2048：1024＝4：2：1

となり，Aに所属するプロセス群は，Cに所属するプロセス群の4倍，Bに所属するプロセス群は，Cに所属するプロセス群の2倍のCPU時間を獲得できる。ただし，個々のプロセスが獲得できるCPU時間は，そのグループに属するアクティブなプロセス数が影響する。また，Root CgroupのCPUシェア値は1024で固定である。例えば，先ほどの例で，Root CgroupにCPUを大量に消費するプロセスがある場合，各CPUグループへのCPU配分は，Root Cgroupを考慮に入れて，以下のようになる。

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

すべてのプロセスをRoot Cgroup以外に配置する

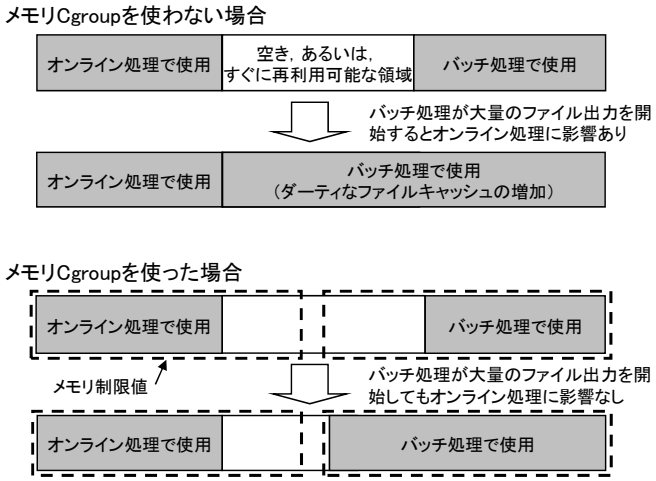


図-3　メモリCgroupの効果
Fig.3-Advantage gained from Memory Cgroup.

ことは手間がかかるため，監視用などの軽量なプロセス群はRoot Cgroupに配置される。Root CgroupのCPU割当ては変更できないため，Root Cgroupにビジーなプロセス群を配置すると，監視用の軽量プロセスのCPU時間の確保ができなくなり，監視業務に影響を与えることが想定される。このため，ビジーなプロセス群は，Root Cgroup以外に配置することが望ましい。

このCPU Cgroup機能は，KVM（Kernel-based Virtual Machine）と呼ばれるLinuxカーネル上で仮想マシンを実現する仕組みにおいても，ゲストOS間のCPU時間の割当てを制御するために，必須の機能である。

## トレース機能

富士通の思い描くミッションクリティカルなシステムでは，発生した問題に対して再現テストを行うことなく原因究明ができるべきであり，そのためには，動作しているシステムの挙動を，航空機におけるフライトレコーダのように常時記録し続けておく機能が必要と考えている。従来のLinuxカーネルでは，このような解析作業を行うには，カーネルのソースに手を加えてメッセージ出力を追加し，再コンパイルして，それを用いて現象を再現させて解析を進める必要があった。しかし，このような方法には以下の問題がある。

- メッセージ出力追加により，システムの動作タイミングが大きく変わり，問題が再現できなくなる。
- ソース修正，再コンパイル後にシステムの再起動が必要となるため，運用中のシステムでそのまま調査することができない，かつ，再起動により問題が再現できなくなる可能性が高い。

このような課題に対処するために，Linuxコミュニティでは様々な形のトレース機能の導入が検討されてきたが，大別すると以下の二つの方式がある。

(1) 動的プローブ方式

観測対象の処理の機械命令に動的にパッチを当てて，プローブとして挟んだ命令コードに処理を移し，その後に，観測対象の元の処理に制御を戻す方式である。プローブとして挟み込む命令列は，ハンドラと呼ばれ，カーネルモジュールとして動的にロードされる。[5]-[7] またハンドラの作成を容易にするために，systemtapと呼ばれるツールがある。[8] これは，tapsetと呼ばれるスクリプト言語からハンドラとなるカーネルモジュールの生成，モジュールのロード，測定結果の取出しまでを自動化するツールである。

(2) 静的プローブ方式

あらかじめプローブ用の命令列をカーネルの実行コードの中に埋め込んでおく方式である。現在実装されているものには，ftraceと呼ばれるコンパイラのプロファイリング機能を利用して，カーネル関数の入口，出口をプローブする方法[9],[10]と，event tracingや，tracepointと呼ばれるカーネルソースに明示的にトレースするポイントを記載する方法[11],[12]がある。

富士通は，まず，systemtapスクリプトによるフライトレコーダ機能（常時トレース機能）を実装した。これはスクリプトであるため，性能に問題があり，採取できるトレースポイント数に制約があった。つぎに，カーネルモジュールを作成して静的プローブ基盤のtracepointを直接プローブすることを試みた。[13] この方式のフライトレコーダは，富士通からRHELのサポートを購入されたお客様に，RHEL5用として既に提供されており，問題解決の短縮に役立っている。ただし，この方式は，継続的に提供するためのコストが高いという課題があった。このため，コミュニティと共同で，より高機能，高性能な，静的トレースのインフラであるftraceの改善に注力した。[14]-[17] RHEL6ではftraceによるフライトレコーダ機能が利用可能になる見込みである。

## スケジューラ改善

富士通は，ミリ秒オーダの安定したトランザクション性能を達成できるシステム[18]を目指して性能チューニングを実施している過程で，RHEL5のスケジューラであるO(1) スケジューラ（Order One Scheduler）に以下の二つの問題を検出し，改善に取り組んだ。

(1) CPU資源の割当て平準化

この改善は，高負荷時にもプロセスやスレッドへのCPU資源割当てを均等に保つために行った。RHEL5のO(1) スケジューラでは，スリープしている割合の多い対話的プロセスに，CPU時間を優先的に割り当て，応答性を確保していた。しかし，対話的プロセスが多いシステムでは，一部の対話的プロセスにのみCPU資源が割り当てられ，システ

ム全体の性能が劣化するという問題を検出した。この問題の解決策をコミュニティで議論した。[19] コミュニティではO (1) スケジューラを捨てて新しいCFS（Completely Fair Scheduler）スケジューラに移行しようとする時期であったため，議論はCFSの改善に取り込まれた。[20] この改善は，RHEL6に取り込まれる見込みである。RHEL5に対しては，この議論の結果がO (1) スケジューラ用の修正として取り込まれた。これらによりCPU資源割当てが平準化され，かつ，対話的プロセスの応答性も極端に悪くならない安定した性能を引き出すことが可能となった。

(2) 応答性向上

従来のLinuxのスケジューラはCPUのキャッシュのヒット率向上によるスループット増加をねらってスケジュールしていた。このために，一度あるCPUの上で動作を開始したプロセスがスリープした後，スリープ状態から起床するときはできるだけ同じCPU上でスケジュールするような考慮がされていた。このため，大規模SMP（Symmetric Multi Processor）システムでは，負荷のかかり方によっては，すべてのCPUを使い切っていないにもかかわらず，システムの性能が頭打ちになる場合があった。この問題に対して，富士通は新たなスケジューリング手法を提案して採用された。[21] 具体的には，システムのグループスケジューラの動作を変える，sched_relax_domain_levelと呼ばれるパラメータを導入した。これは，プロセスを起床するときに，アイドル（実行すべきプロセスを持たない状態）なCPUを探す範囲をチューニングするパラメータである。全く別のアイドルなCPUを探さないことを意味する0から，システム全体のアイドルなCPUを探す5まで，6段階にアイドルなCPUの探索範囲をチューニングできるようにしている。[22] この機能により，大規模SMPシステムのCPUの能力をフルに引き出して，高い応答性能を持続することが可能となった。

## MCAリカバリ機能

本特集で紹介しているPRIMEQUEST 1000シリーズは，Xeon 7500番台（インテル社開発コードNehalem-EX）のCPUを採用している。[23] このCPUは，x86のCPUとして初のMachine Check Architecture Recovery機能（MCAリカバリ機能）を実装している（**図-4**）。この機能は，従来のPRIMEQUEST 400/500シリーズで使用していた，Itanium2プロセッサーに実装されていた機能と同等の機能であり，富士通はItanium2版PRIMEQUESTでの経験を生かして，この機能の品質強化に貢献した。

Xeon 7500番台のCPUでは，ハードウェア故障の自己診断・エラー訂正機構が備えられている。この機能により，ハードウェア故障に対して，まずはハードウェアのレイヤでエラー訂正を試みる。訂正が成功すれば，ソフトウェアの処理は続行し，エラー訂正が起きたことだけを通知するCMCI

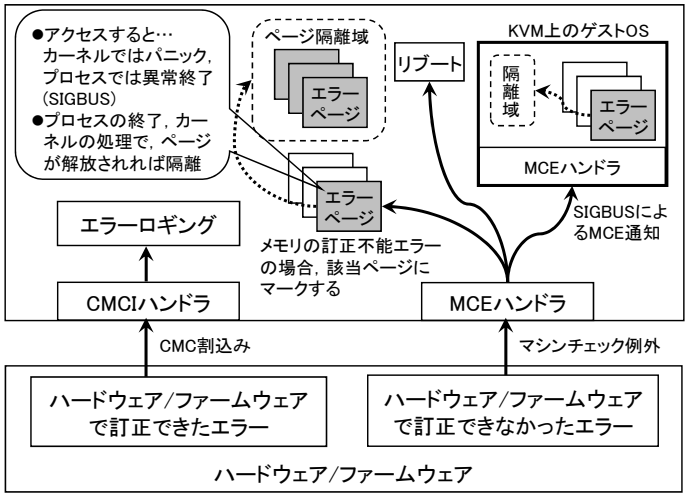


図-4 MCAリカバリの概念
Fig.4-Concept of MCA Recovery.

（Corrected Machine Check Interrupt：CMC割込み）が発生する。ハードウェアで訂正できない場合は，MCE（Machine Check Exception：マシンチェック例外）を発生させ，OSに回復処理を依頼する。

OSはCMC割込みを受けた場合は，エラーロギングだけを行う。マシンチェック例外を受けた場合は，エラー内容を解析した上で，エラーに応じて回復処理を行う。CPUの内部などで訂正不能なエラーが発生した場合は，致命的なエラーの発生となり，MCEを受けたOSはパニックする。OSが復旧を試みる訂正不能なエラーの例としては，メモリやキャッシュ上で発生したECCマルチビットエラーが挙げられる。Xeon 7500番台のメモリコントローラには，定期的にメモリをアクセスしてメモリを診断する「メモリスクラブ（Memory Scrubbing）」と呼ばれる機能があり，この診断で訂正不能なECCマルチビットエラーが検出された場合は，OSに対してMCEが通知される。OSは，MCEの内容を確認して，対象のページに対してエラーの印をつける。このページが利用されていない場合は，これらのページを利用不能として隔離する。利用中の場合には，解放されるタイミングを待ち，解放されたときに利用不能として隔離する。

富士通では，インテル社から初版のパッチがリリースされた後に，レビュー，故障をエミュレーションするモジュールを作成してデバッグするなどして，この機能の品質強化に貢献した。[24)-27)]

## む　す　び

富士通はLinuxの機能や品質を強化するためには，源流であるLinuxコミュニティで活動し続けることが重要であると考えている。本稿では，富士通がLinuxコミュニティで開発や品質強化に参加した，Cgroup，トレース機能，スケジューラ改善，MCAリカバリ機能の四つの機能について解説した。

このうち，トレース機能とスケジューラ改善は，RHEL5を採用する様々なシステムの安定稼働に既に貢献している。また，Cgroup，ftraceによるフライトレコーダ機能（常時トレース機能），CFSスケジューラ，MCAリカバリ機能はRHEL6で提供される見込みであり，お客様システムの更なる安定稼働に貢献が期待できる。

富士通は，今後も更なるLinuxの改善に向けて，積極的にコミュニティで活動していく決意である。

最後に，我々とともにLinuxの改善に尽力されているコミュニティの皆様に感謝いたします。

**参 考 文 献**

(1) 黒羽法男：急速に成熟するLinux OS（機能・信頼性）．*FUJITSU*，Vol.56，No.3，p.221-225（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) 平松雅巳：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，p.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) 富士通：アニュアルレポート2010，p.59（2010）. http://pr.fujitsu.com/jp/ir/annual/2010/pdf/all.pdf

(19) Satoru Takeuchi : [BUG] scheduler : strange behavor 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) 廣瀬元義ほか：社会基盤を支える高信頼基幹IAサーバPRIMEQUEST 1000シリーズ．*FUJITSU*，Vol.61，No.6，p.577-588（2010）.

(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