特集

Linuxテクノロジの最先端

Cutting-Edge Linux Technologies

あらまし

近年，Linuxはデスクトップ用途やサーバ用途における基盤OSの一つとなっており，さらにその適用範囲を拡大するために，インターネットを通じて世界中のカーネル開発者によって改良や性能改善が行われている。現在開発が進められているLinuxカーネルでは，ファイルシステム強化やプロセススケジューラの性能改善など主に大規模基幹業務での運用を強く意識した改良に加え，組込み用途なども考慮した新機能の追加などが行われている。

本稿では，とくに，Linuxが最も普及しているハードウェアプラットフォームであるIA32（32-bit Intel Architecture）を対象として，現在開発が進められているLinuxカーネルで導入されている最先端技術について概説する。加えて，富士通研究所で行っているLinuxの大規模基幹業務への適応に向けた取組みについても簡単に紹介する。

Abstract

Linux has already become one of the major operating systems in use on desktop and server machines. Kernel developers around the world have been upgrading and improving the performance of Linux via the Internet to expand the scope of its application. The Linux kernel presently being developed will take large-scale backbone operations such as enhancements of file systems and improvements of process scheduler performance into strong consideration. It will also include the addition of new functions that have been designed for embedded applications. This paper describes the leading-edge technologies incorporated in the Linux kernel currently being developed for the 32-bit Intel architecture (IA32), which is the most prevalent hardware platform for Linux systems. This paper also introduces Fujitsu Laboratories' research and development activities for Linux application in large-scale backbone operations.

山村周史（やまむら　しゅうじ）

ITコア研究所グリッド&バイオ研究部 所属
現在，Linuxカーネルの性能評価・性能チューニングの研究に従事。

久門耕一（くもん　こういち）

ITコア研究所グリッド&バイオ研究部 所属
現在，IAベースシステムの性能評価やLinuxカーネルチューニング，クラスタコンピューティングおよび高速インタコネクトの研究に従事。

ま え が き

Linuxは，1991年リーナス・トーバルズ（Linus Torvalds）氏によって開発が始められてから現在に至るまで，その性能や安定性・ユーザビリティを着実に向上させてきた。Linuxは，すでに多くのデスクトップやサーバで盛んに利用されており，この分野での基盤OSの一つとして確固たるシェアを築きつつある。最新のLinuxであるバージョン2.5は，さらにその適用範囲を拡大すべく，大規模サーバ向けの拡張や組込み用途向けの改良をはじめとした数多くの新機能が追加されている。

本稿では，現在最も普及しているIA-32（32-bit Intel Architecture）を対象として，Linuxの最先端技術について概説するとともに，Linuxの大規模基幹業務への適応に向けた富士通研究所の取組みについても紹介する。

Linuxカーネルの開発スタイル

Linuxシステムは，「カーネル」と呼ばれるOSの中核部分とそれを取り巻く多数のライブラリやツールなどで構成されている。カーネルは，OSの機能を定める根幹部であり，メーリングリストを通して世界中の開発者によりオープンな議論のもとで改良が続けられている。Linuxのカーネルのバージョン番号には特殊なルールがあり，2番目の数字が偶数のものを安定版カーネル，奇数のものを開発版カーネルとして区別している。安定版カーネルについては，基本的には細かなバグ修正しか行われず，データ構造やアルゴリズムの変更といった大幅な修正は開発版カーネルで行われる。

開発版カーネル2.3を経て2001年1月に発表された現在の安定版カーネル2.4は，デスクトップ領域を越えたサーバ・エンタプライズ分野での利用を意識した改良や機能追加が重点的に行われた。例えば，64 Gバイトのメモリサポートやファイルサイズ制限の引上げ，BKL（Big Kernel Lock）(注)の細分化によるマルチプロセッサ機の性能改善などである。これを発展させた最新の開発版カーネル2.5は，さらなる機能改善を推し進める形で大規模基幹業務への適応を視野に入れ，次期安定版カーネル2.6に向け現在その開発が進んでいる。

最新カーネル2.5での機能強化

最新カーネル2.5は，2001年11月にカーネル2.4.16から枝別れしてその開発がスタートした。2002年11月現在での最新バージョンは2.5.48であり，カーネル2.4に比べ，実に数多くの機能追加や改善が行われている。ここでは，とくに重要な改良が加えられたポイントについて順に解説する。

● ファイルI/Oの強化

大規模サーバや基幹業務でのLinuxの利用を目的とした場合には，例えば大規模データベースアプリケーションなど，巨大なファイルを効率的にアクセスできることが望まれる。カーネル2.5では，このような利用に耐え得るようにファイルI/O機能について大幅な改善が行われている。

【ブロック（ディスク）I/Oの改良】

アプリケーションが発行したディスクI/Oのリクエストは，カーネル内部のリクエストキューにいったん蓄えられ，その後ディスクに対して順に発行される。カーネル2.4では，このキューが“io_request_lock”と呼ばれる一つのスピンロックで保護されており，複数のCPUが同時にキューにアクセスすることはできなかった。そのため，CPU台数を増やしても性能スケーラビリティが得られないという問題があった。カーネル2.5では，各デバイス単位にロック処理を細分化することで複数のCPUがブロックI/O処理を並列に実行できるように改良された。

【非同期I/O機能の追加】

「非同期I/O（Asynchronous I/O）」とは，アプリケーションからのI/Oリクエストを OSが受け付けた後，実際にI/O処理が終了するまで待つことなく，アプリケーションの実行を再開できる仕組みである。この機能を使用するとI/O処理完了待ちの間に，別の計算処理を並列して実行することができるので，主にデータベースアプリケーションなどで多くのI/O要求を発行しながら動作するプログラムで効果を発揮する。

【ページ単位でのI/O処理】

カーネル2.4までは，ディスクへの書込みの際に，

(注) カーネル2.4以前は，マルチプロセッサシステムにおいて，カーネルの大部分が並列実行できないようにスピンロックによって保護されていた。このスピンロックのことを“Big Kernel Lock”と呼ぶ。

アプリケーションが書き込むデータをカーネル内部でいったん小さなブロック単位に分割した後，実際の書込みを行っていた。この仕組みによって，各ブロックを並べ替えることでディスクへの書込み処理を最適化することができるが，近年のファイルサイズの大容量化に伴い，ブロック単位での処理では効率が悪くなってきていた。そこで，これを一新してページ単位でのI/O処理を行うように変更された。これによって，ブロック単位への分割処理が省略され，大きなサイズのファイルI/O処理を効率的に行うことができるようになった。

これらのほかにも，ディスクキャッシュを使用せず，直接ユーザ空間とファイルの間でDMA（Direct Memory Access）によるデータ転送を行うことのできる「ダイレクトI/O機能」や，1 Gバイト以上のメモリ空間に存在するデータを，バッファへのコピーを介さずにI/O処理するための高速化など，様々な最適化が随所に施されている。

● プロセススケジューラの改良

カーネル2.5において，最も大きな変更の一つがプロセススケジューラ（以下，スケジューラ）の改良である。Linuxは，多くのプロセスを並行実行できるマルチタスクOSである。カーネルは，ある時間間隔で複数のプロセスをCPU上で切り替えながら実行している。このとき，システム内の様々なリソースを考慮しながら，CPU上で次に実行すべきプロセスを選択するのが「スケジューラ」と呼ばれる部分である。

カーネル2.4でのスケジューラは，システム上に存在するすべてのプロセスを，RUNキューと呼ぶ1本のリストを用いて管理していた（**図**-1）。カーネルは，つぎに実行するプロセスを決定する際，RUNキュー上のプロセスを順にすべてチェックして最適なプロセスを選択していた。このため，サーバ分野などにおいて多数個のプロセスが動作している場合には，RUNキューが非常に長くなるためスケジューリングに多くの時間を浪費していた。また，RUNキューは，複数のCPUが同時にアクセスできないように制限されていたためスケーラビリティが得られず，より問題が深刻化していた。

この問題を一気に解決したのが，カーネル2.5で実装された「O(1)スケジューラ」である。O(1)スケジューラでは，**図**-2のように，“ active ”と“ expired ”と呼ばれるプロセス優先度を添え字とした二つの配列を持つ。各配列要素には，同一の優先度を持つプロセスがキュー構造でリンクされている。カーネルは，各キュー上でのプロセスの有無を示すビットマップを参照することで，最も高い優先順位にあるプロセスを直ちに見つけることができる（計算に要するオーダが1）。スケジューリング後のプロセスはexpiredキューに移動され，activeキューが空になった時点で二つのキューの役割を交換する。さらに，CPUごとにキュー管理が行われるために，複数のCPUが動作するシステムにおいても競合が

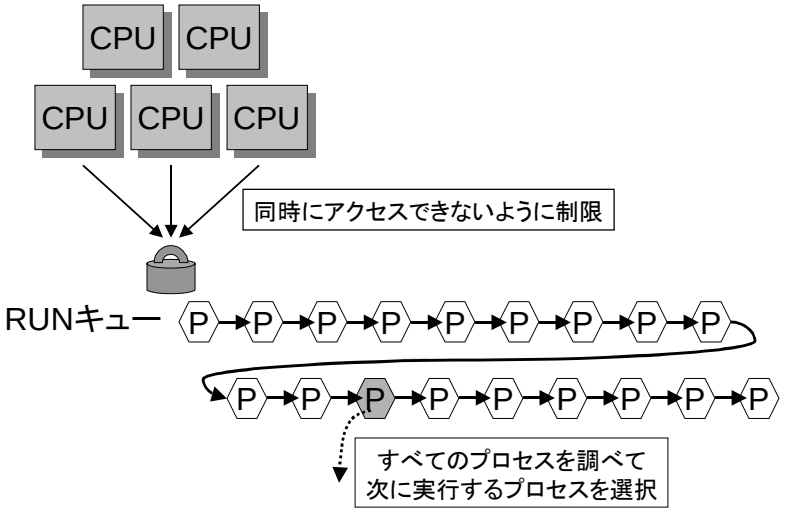


図-1　RUNキューの構造
Fig.1-Structure of RUN queue.

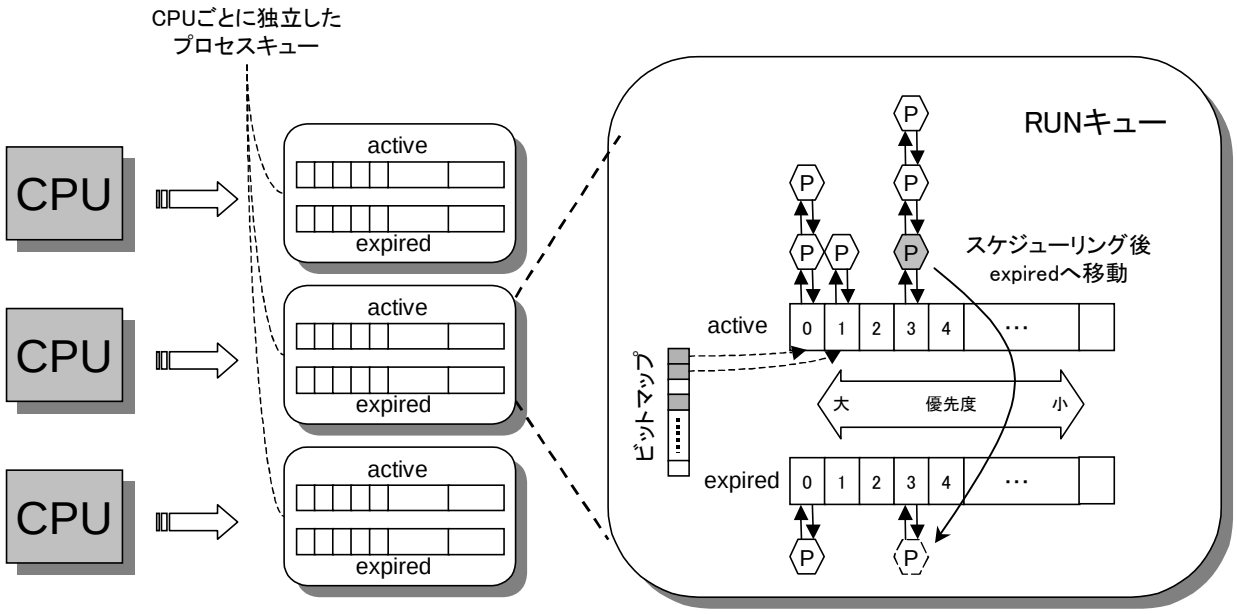


図-2　O(1)スケジューラの構造
Fig.2-Structure of O(1) scheduler.

発生しにくくなり，高速なスケジューリングとともに高い性能スケーラビリティが得られる。実際に，“Chat micro benchmark”(1)と呼ばれる多プロセスを起動する負荷テストにおいては，Pentium Ⅲ 550 MHz 8CPUシステムで従来のスケジューラと比較した場合，最大80％以上の性能向上が得られた。

また，複数CPUシステムを対象として，プロセスを動作させるCPUを限定することができる「プロセスバインディング機能」もスケジューラ内部に組み込まれた。アプリケーションによっては，この機能を使用することで，CPU間での余分なプロセス移動を抑制して性能チューニングを行うことができる。

● プリエンプション可能カーネルのサポート

エンタプライズ分野に対する機能強化だけでなく，デスクトップ用途や組込み用途での機能強化も進んでいる。その一つが，カーネルプリエンプション機能のサポートである。

プロセスは，外部からの割込みやシステムコールの発生などを契機として，カーネルがスケジューリングを行う。このように，プロセスの実行が割り込まれてスケジューリングが発生することを「プリエンプション」と呼ぶ。カーネル2.4では，カーネル自身が実行されている間はプリエンプションが発生しないように設計されていた。この実装は，カーネル内部のアルゴリズムを単純化する上での利点となっていたが，リアルタイム性が要求される組込みシステムなどでは，応答性に問題が発生する可能性があった。

カーネル2.5では，カーネル2.4で既に実装されていたスピンロック機構を流用することでプリエンプション機能をシンプルに実装している。もともとマルチプロセッサ用カーネルでは，カーネル内部で並列実行できない部分をスピンロックと呼ぶ機構を使用して保護していた。逆に言えば，スピンロックで保護された領域以外は並列実行してもよいことになる。そこで，スピンロック解除時にスケジューラを起動することで，カーネルプリエンプション機能を実現している。

● User-mode Linuxの導入

近年，単一コンピュータ上に一つ，あるいはそれ以上の数の仮想的なPC環境を構築し，それぞれで独立したシステムを動作させる仮想OS技術が登場している。“User-mode Linux（UML）”は，この技術の一種であり，Linuxシステム上のユーザプロセスとして動作するLinuxカーネルである。カーネル2.5では，この機能が導入されており，実際のLinuxシステム上に仮想的に複数のLinux環境を作成・起動することができる。仮想環境間では，なん

ら互いに悪影響を与えないので，例えばカーネルの新機能のデバッグやテスト，仮想ネットワークの構築・テストなどに活用できる。また，単一サーバでの高信頼性・高可用性の強化などへの利用も期待される。

● システム分析機能の追加

カーネル2.5では，システムの挙動を分析するために“Oprofile”と呼ぶプロファイリング機能が導入されている。この機能は，時間ベースでのプロファイリングだけでなく，CPU内部に装備されているハードウェア性能モニタリング機能[2]を使用することで様々な測定項目（例えばキャッシュミスなど）によるプロファイリングを行うことができる。効果的に活用すれば，カーネルの性能ボトルネックとなっている部分を調査することができる。

● そのほかの機能追加

カーネル2.5では，ここまでに述べたもののほかにも様々な改良が加えられている。例えば，

・4 Kバイト以上のラージページのサポート
・2 Tバイトの上限ファイルサイズを撤廃
・新CPUへの対応（x86-64，PowerPC64）
・管理できるデバイスの上限引上げ
・新スレッドライブラリのサポート
・SCTP（Stream Control Transmission Protocol）のサポート
・MMU（Memory Management Unit）を装備しない組込み向けCPUのサポート
・大規模NUMA（Non Uniform Memory Access）向け機能強化

といった項目が挙げられる。

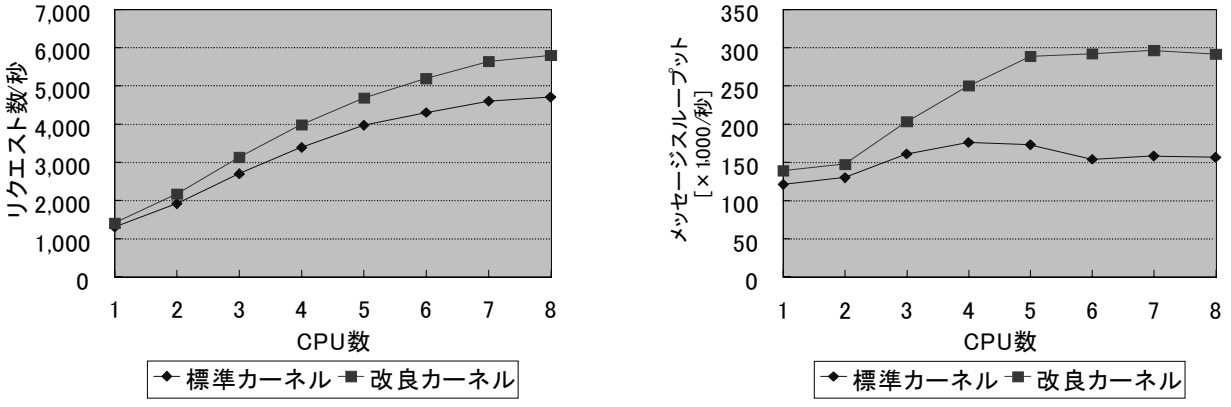
なお，エンタプライズ市場・大規模システム向けのLinux製品であるRed Hat Linux Advanced Server 2.1には，これまでに述べたカーネル2.5で採用されたファイルI/O強化やO(1)スケジューラの導入といった機能改善が取り込まれており，大規模基幹業務における高性能・高信頼性をサポートするための最適化が図られている。

富士通研究所の取組み

これまでに述べたように，Linuxはエンタプライズ領域への適用範囲の拡大を大きな目標としている。このような分野でのLinuxシステムの性能分析や問題点の発見・カーネル改良などは，一般のユーザにはハードウェア資源などの点で困難であり，近年は富士通をはじめとした企業への期待がますます高まっている。富士通研究所では，これに先駆けて1999年ごろからLinuxのエンタプライズ向けの性能評価やカーネル改善に取り組んできた。ここでは，富士通研究所での取組みを簡単に紹介する。

● NUMA性能チューニング

NUMAとは，マルチプロセッサシステムにおいて，CPUによってメモリアクセスに要する時間が均一でないシステム構成のことで，大規模サーバなどで採用されることが多い。このシステムでは，自CPUから物理的に近くにあるメモリ（ローカルメモリ）にデータを配置するようなチューニングが有



（a）Webサーバ性能　　（b）Chatサーバ性能

（注）GRANPOWER 5000（HS900）Pentium III Xeon 550 MHz 8CPU使用

図-3　スケジューラの改良による性能向上
Fig.3-Performance improvement of modified scheduler.

効である。これを実現するためにカーネルのメモリシステムに対して修正を加えた。これにより，2ノード4CPUシステム上でWebサーバ性能が最大20%向上した[3]

● スケーラビリティ改善

カーネル2.4では，CPUを増加させたときの性能スケーラビリティが得られないという問題があった。メモリシステムの観点からこれを調査することで，スケジューラ内部で多発するキャッシュミスが原因の一つであることを明らかにした。この問題を解決する修正を施すことで，**図-3**に示すようにWebサーバ性能が最大23.3%の向上，多プロセス動作時の負荷テストを行うChat micro benchmarkで最大90%の向上と，Linuxの性能スケーラビリティが大幅に向上した[4]

このほかにも，O(1)スケジューラの定量的な評価などLinuxを使用したシステムの性能評価・分析を行っており，得られた情報をオープンな形でLinuxコミュニティにフィードバックしている。さらに詳しい情報は，URL http://www.labs.fujitsu.com/techinfo/linux/を参考にされたい。

む　す　び

本稿では，最新のLinuxカーネル2.5に導入された様々な機能を簡単に解説した。カーネル2.5では，実に数多くの改良が加えられており，デスクトップからサーバ，あるいは大規模基幹業務にまでLinuxの適用範囲を広げる準備が確実に整いつつある。次期安定版カーネル2.6のリリースは2003年第2四半期ころと言われており，本稿で紹介した機能が盛り込まれる可能性が高く，その動向から目が離せない。著者らも，Linuxの大規模基幹業務への適応に向けて，最新Linuxの性能評価・分析・改良にこれまで以上に精力的に取り組む予定である。

参 考 文 献

(1) M. Kravetz et al.：Enhancing Linux Scheduler Scalability . 5th Annual Linux Showcase & Conference, November 2001.

(2) IA-32 Intel Architecture Software Developer's Manual Volume 3: System Programming Guide. "Debugging and Performance Monitoring", 2002.

(3) 平井聡ほか：NUMAマシンでのコマーシャルワークロード向けLinux最適化．情報処理学会論文誌, Vol.43, No.4, p.1018-1027（2002）.

(4) S. Yamamura et al.：Speeding Up Kernel Scheduler by Reducing Cache Misses. In Proc. of the USENIX 2002 Annual Technical Conf. FREENIX Track, p.275-285（June 2002）.