論文の内容の要旨

論文題目　**Management Systems for Efficient Allocation of Computing Resources and Trustworthy Migration of Operating Systems**
（計算リソースの効率的な割り当てとオペレーティングシステムの確実なマイグレーションのための管理システム）

氏　名　　**Kuniyasu Suzaki 須崎 有康**

Abstract

This thesis describes the design of large scale management systems, which is based on abstraction that allows optimization for underlying architecture. The importance of the design was demonstrated by two implementations of different kinds of management system: a parallel scheduling and OS migration.

Recently, management systems cannot deal with overall configurations and conditions because size and complexity of hardware and rapid development of software. The underlying architecture which includes software architecture (e.g., file system, network protocol) is difficult to see from the management systems. It causes inefficient execution, implicit security problems, and frequent administration.

The goal of this thesis is to offer efficient and trustworthy management systems, which balance user requests and current computing resources. The design of management systems must be abstracted so as to balance the simplicity and performance, which take account of the underlying system architecture. Following the design approach, two management systems with different requirement and usage, were developed. One management systems is a process scheduling on massive parallel computers, which achieves quick response of each task and high processor utilization. The other management system deals with migration of operating systems through the Internet, which copes with the problem of software version management. For the former approach, I developed a scheduling which combined space sharing and time sharing. For the latter approach, I developed a framework to distribute bootable disk images through the Internet.

The scheduling which combines space sharing and time sharing offers abstracted parallel computers called slices. Tasks are allocated on a slice with a partitioning algorithm for each architecture. The slices are dispatched to real parallel computers and tasks are processed. The feature of the scheduling is co-existing of tasks among slices.

When the area of a task on a slice is free on other slices, the task can sit on these slices in order to attain better processor utilization. Multiple tasks reduce the number of slices and make quick responses. The scheduling was simulated on mesh-connected parallel computers (32×32) with two partitioning algorithms: Adaptive Scan and 2D Buddy. The simulation showed the increase of processor utilization was 38% and 19% on Adaptive Scan and 2D Buddy respectively. The scheduling was implemented on AP/Linux: a parallel operating system for Fujitsu AP1000+. The implementation utilized the Linux local schedulers which were loosely synchronized by hardware clock, and communication library which switched from busy wait to signal wait. It offered moderate co-scheduling and achieved good performance for fine- and coarse-grain processes. It differs from rigid gang scheduling in that it does not promise strict synchronization of a parallel process on all processors. Therefore, the scheduling was supposed to have processor thrashing for fine-grain parallel processes, but it avoided the thrashing problem by controlling all parallel processes with the organized hardware clocks. Conversely, it offered effective processing of coarse-grain parallel processes by overlapping with the communication library which switched from busy wait to signal wait. The co-scheduling skew of the implementation was 2% in a range of co-scheduling period 200ms, which was compensated by the high processor utilization of the scheduling.

The migration of operating system allows us to boot any operating systems without installation on any machine. It is used for bringing back an old OS image or trying a new OS function without side effect. The OS migration is called OS Circular and operating systems are transferred by the network transparent virtual disk called LBCAS (LoopBack Content Addressable Storage). LBCAS reconstructs a virtual disk from abstracted block files which are made from split and compressed block device. The file abstraction make easy to treat. The block file is named by the SHA-1 value of its contents, and the access to the virtual disk is a redirect addressing based on the SHA-1 block files. The same contents blocks are shared by the same SHA1 value and reduce the total volume. The update of application on the LBCAS creates new block files with new SHA-1 file name. The old block files are reserved and make possible to bringing back an old OS image. The new block files are creased for the updated data only, and the saver save the total volume of LBCAS. The integrity of block files is verified with the SHA-1 on a client. The feature makes possible to distribute the block files by un-trusted servers. The downloaded block files are cached on a local storage in order to eliminate redundant download.

The performance of LBCAS was affected by underlying software architectures: the

data, access pattern, and the size of block file. The feature was made cleared by the experiment and the optimizations for quick boot were applied. The target operating systems were Windows and Linux, and the LBCAS is optimized by reallocation tool of files system (DiskKeeper and PerfectDisk for Windows and ext2optimizer for Linux). The latency issues in the Internet were optimized by DLAHEAD and DNS-Balance. DLAHEAD downloads necessary block files in advance at boot time. DNS-Balance is the name resolver which uses routing information database and suggests the nearest server for each client. The techniques hid network latency and made possible to distribute the disk image worldwide.

The implementation of two management systems showed that the design compensated the abstraction overhead by allowing optimization for each architecture. The design balances portability and performance on management systems and offers the sophisticated computing environment.

論文要旨

本論文は下位のアーキテクチャの最適化を考慮する抽象化を基とした大規模管理システムについて述べる．この設計の重要性は異なる二種類の管理ステムである並列スケジューリングと OS マイグレーションの実装を通して明らかにする．

近年，管理システムはハードウェアの規模や複雑さ，およびソフトウェアの頻繁な開発のために全体を見渡すことが難しくなっている．下位のアーキテクチャはソフトウェアアーキテクチャ（例えばファイルシステムやネットワークプロトコル）であっても，管理システムから見えにくくなっている．このことは実行の非効率化，セキュリティ問題の潜在化，管理業務の増大化を招いている．

本研究の目標は，ユーザの要求と利用可能な計算資源を考慮した効率的，且つ信頼できる管理システムを提供することである．管理システムの設計は下位のアーキテクチャの最適化を考慮して単純さと効率を両立するように抽象化しなければならない．この設計方針の下，それぞれ異なる利用形態および制約条件を持つ２つの管理システムの開発を行う．一つは大規模並列計算機においてユーザが投入する個々をタスクの応答性を良く，且つ全体の計算機利用率を高く実行するプロセススケジューリングである．もう一つは，ソフトウェアのバージョン管理のためのインターネット経由の OS マイグレーションを扱う．前者ために，空間分割と時間分割を効率的に融合したスケジューリングを開発した．後者のために，起動可能なディスクイメージをインターネット経由で配信するフレームワークを開発した．

空間分割と時間分割を融合したスケジューリングでは，任意の並列計算機を抽象化する

slice を用意し，その上に並列計算機のアーキテクチャを考慮したパーティショニングアルゴリズムでタスクを割り当てる．slice は実並列計算機に割り当てられてタスクの処理を進める．このスケジューリングではスライス間に共存するタスク管理を特徴とする．ある slice でタスクが占有している領域を別の slice で未使用の場合，そのタスクを複数の slice に配置することによりプロセッサ利用率の向上を計る．多重タスクは slice 数の削減を行い，素早い応答性を可能にする．このスケジューリングは 2 つのパーティショニングアルゴリズム Adaptive Scan と 2D Buddy を対象に，メッシュ結合並列計算機（32×32）でシミュレーションを行った．シミュレーションはプロセッサ利用率の向上が Adaptive Scan と 2D Buddy でそれぞれ 38%および 19%であることを示した．

提案したスケジューリングは実計算機 AP1000+ のオペレーティングシステムである AP/Linux に組込まれた．実装においては各プロセッシングエレメント上の物理時計に緩やかに同期する Linux のローカルスケジューラと busy wait から signal wait に切り替わる通信ライブラリを活用した．この方式は fine 及び coarse-grain の並列プロセスを効率的に実行する緩やかな co-scheduling を提供した．本方式は厳格なプロセス同期を要求するギャングスケジューリングと異なり，並列プロセスが同時に実行されていることを保証しない．このため頻繁な通信を行なう fine-grain 並列プロセスの場合，プロセッサスラッシングが予想されるが，一元管理された物理時計による全並列プロセスを管理することで解決できた．また， coarse grain 並列プロセスも一定時間で busy wait を signal wait に切り替える通信ライブラリを併用することで並列プロセスをオーバーラップさせ，効率的に処理できた．co-scheduling の周期を 200ms とした場合，co-scheduling skew が 2%程度であったが，これはスケジューリングのプロセッサ利用率の向上により十分に打消される範囲であった．

OS マイグレーションは任意のマシン上でインターネットからインストールなしで OS を利用可能する．この機能により，古い OS イメージへのロールバックや新しい OS の新機能を副作用なしで確認することを可能にする．この OS マイグレーションを OS Circular と呼び，OS は LBCAS (LoopBack Content Addressable Storage)と呼ぶネットワーク透過な仮想ディスクにより転送される．LBCAS はブロックデバイスを分割圧縮して作られた抽象化ブロックファイルから仮想ディスクを再構築する．ファイルによる抽象化は取扱いを簡単にする．ブロックファイルはその内容の SHA-1 値をファイル名とし，仮想ディスクへのアクセスは SHA-1 ブロックファイルによる間接アドレスシングとなる．同一内容のブロックファイルは同一 SHA-1 値で共有され，総容量を削減する．LBCAS でのソフトウェア更新では新しい SHA-1 値のブロックファイルを追加することになる．古いブロックファイルはそのまま保存され，古いＯＳイメージへロールバックを可能とする．また，新しいブロックファイルは変更のあったデータのみに対して作られるため，サーバの総容量を減らすことができる．クライアントでは SHA-1 によるコンテンツの完全性を検証する．この性質は信頼できないサーバからの配信も可能にする．ダウンロードしたブロックファイルは不用

なダウンロードを抑制するため手元の二次記憶にキャッシュされる．

LBCAS 性能は，収録するデータ，アクセスパターン，および分割サイズなどの下位のソフトウェアーキテクチャによって影響される．この特徴は実験により明らかにし，高速な起動のための最適化を行った．起動する OS は Windows と Linux とし，ファイルシステムのブロック再配置ツール(Windows では DiskKeeper と PerfectDisk，Linux では ext2optimizer)を利用して LBCAS を最適化した．インターネットのレイテンシは DLAHEAD と DNS-Balance により最適化された．DLAHEAD は必要なブロックファイルを先行ダウンロードする．DNS-Balance はルーティング情報データベースに基づくネームサーバであり，各クライアントに対して近接のサーバを導く．これらの技術はネットワークの遅延を隠蔽し，世界規模のディスクイメージ配信を可能とした．

２つの管理システムの実装を通して，設計が各アーキテクチャに対する最適を考慮することで抽象化のオーバーヘッドを打ち消すことを示した．この設計が管理システムの汎用化と効率のバランスを取り，洗練された計算環境を提供することができた．