Vol.2011-ARC-194 No.12
2011/3/11

機械学習により抽出されたアプリケーションの特徴を利用したタスク配置の検討

穂園 智哉[†1]　近藤 正章[†1]
平澤 将一[†1]　本多 弘樹[†1]

近年，ウェブサービスの提供やデータベース処理の需要の増加，さらにはクラウドコンピューティングの普及などにより，サーバ計算機を多数運用するデータセンタの重要性が高まっている．データセンタは，増大するサービス要求を処理するために，稼働するハードウェアの更新が欠かせない．しかし，一度に全ハードウェアを更新することは稀で，通常は徐々に入れ替えが行なわれるため，ヘテロジニアスなサーバ計算機環境になることが多い．そのため，タスクの MIPS 値は実行するサーバによって異なると考えられ，タスクの特徴によってタスク処理の際の MIPS 値を最大にする実行サーバが変わり得る．本稿では，タスク処理の特徴と各サーバの相性を機械学習により予測し，それに基づいてサーバへタスク配置を行うことで，データセンタのスループットを向上させる手法を提案する．また，機械学習による相性予測の精度と，予測結果を用いたタスク配置によるデータセンタのスループット向上の有効性を検証する．

A Task Allocation Strategy for Heterogeneous Server Systems with A Machine Learning Approach

Tomoya Hozono,[†1] Masaaki Kondo,[†1]
Shoich Hirasawa[†1] and Hiroki Honda[†1]

The total performance of server computers is one of the important concerns in data-center operations. To meet the increasing service demands, the hardware of server systems in data-centers is replaced frequently. Since a replacement is done for a part of the server systems, there is a heterogeneity in data-centers. Therefore, the performance of a task depends on which server is used for processing the task. The server that brings the best performance may vary task by task. In this paper, we propose a task allocation technique based on predicting the best performing server to improve the total data-center throughput. The prediction is realized by a machine learning approach. We study the accuracy of the prediction method and the effectiveness of the task allocation technique.

1. はじめに

近年，インターネットの急速な普及やクラウドコンピューティングの進展により，それらの処理を行うサーバ計算機の重要性が高まっている．データセンタでは，上記目的や他のサービス提供のためにサーバ計算機が多数稼働している．データセンタでのサーバ計算機運用は，機能強化や安定稼働のためのソフトウェアバージョンアップや，稼働するハードウェアの更新が欠かせない．一般的にデータセンタ内のサーバ計算機の更新は，予算に応じて新規格のものを導入したり，あるいは保守期間終了に合わせて入れ替えたりなど様々なタイミングで徐々に行われる．そのためデータセンタは異なる性能のサーバが混在するヘテロジニアス環境となっていることが多い．

また，データセンタでは様々なアプリケーションタスクが処理されている．タスクの処理には，メモリアクセスがボトルネックになるもの，ディスクアクセスがボトルネックになるもの，処理のほとんどが CPU の計算に費やされるものなど様々な特徴がある．このため，性能的にヘテロジニアスな環境のデータセンタでは，タスク毎にそのタスクの処理と相性が良い，つまり最大の性能 (MIPS 値) が得られるサーバが異なると考えられる．ヘテロジニアス環境において，このようなタスクとサーバの相性を考慮せずにタスクをサーバに配置してしまうと，稼働しているサーバ全体のスループットの低下に繋がる．

そこで本研究では，その処理の特徴が未知のタスクと各サーバとの相性の予測を行い，予測結果を用いたタスク配置手法を提案し，サーバ全体のスループットを向上させることを目的とする．本手法はサーバ運用前にあらかじめ多数のサンプルタスクを実行し，サーバ計算機の処理情報を示す様々なハードウェアイベントの定量的な値から機械学習を行い，タスクとサーバの相性予測のためのモデルを構築する．その上で，運用時には未知タスクのハードウェアイベントにおける値をモデルに適用し未知タスクとサーバの相性を予測する．そして得られた予測結果をもとに，できるだけ相性の良いサーバへタスクを配置することでデータセンタ全体の性能を向上させる．本稿では機械学習による相性予測の精度と，予測結果を用いたタスク配置によるサーバ全体でのスループット向上の有効性を検証する．

†1 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications

Vol.2011-ARC-194 No.12
2011/3/11

2. タスク配置手法と機械学習

2.1 データセンタのタスク配置

本稿では，データセンタ内での処理としてユーザ (クライアント) からタスクの実行要求が送られ，それらがサーバ上で実行されることを想定する．通常クライアントから送られたタスクは，タスクを各サーバに配置する役目を持つスケジューラへと送られる．スケジューラが各サーバへタスクを配置する方法には，順次配置手法や RoundRobin 手法などがある．順次配置手法は，各計算機資源を監視し，処理が終了して CPU やメモリリソースが空き，タスクが処理可能となったサーバへ，スケジューラのキューの先頭にあるタスクを順次的に配置していく手法である．RoundRobin 手法は，スケジューラへタスクが送られてくると，そのタスクを各サーバに順番に配置していく手法である．この手法はサーバ台数が多いほど負荷分散による性能向上効果があり，例えば大規模掲示板群の運用に用いられていると言われている．

これらの配置手法は，ヘテロジニアスな環境によるタスクとサーバの相性を考慮していない．そのため，場合によっては相性の悪いサーバにタスクが配置され，サーバ全体の性能が低下する可能性がある．本稿ではサーバ全体のスループットの向上を目的に，タスクとサーバの相性を考慮したタスク配置手法を検討する．

2.2 機 械 学 習

機械学習とは人間が行う「学習」をコンピュータで実現させる技術と手法である．統計学との関連が深く，サンプルデータ集合を対象に解析を行い，そのデータから有用な規則，判断基準を抽出する．機械学習は強化学習・教師なし学習・教師あり学習の大きな 3 つの枠組みがあり[1)]，検索エンジンや音声認識など幅広い分野で用いられている．本稿ではタスクとサーバの相性の抽出に教師あり学習での階層型ニューラルネットワークを用いている．ここでは階層型ニューラルネットワークについて説明する．

階層型ニューラルネットワークとは，人間の脳の神経回路網をコンピュータ上にシミュレーションする技術であり，教師あり学習において数値予測に用いられるニューラルネットワークの代表的なモデルである[2)]．3 層の階層型ニューラルネットワークを**図 1** に示す．階層型ニューラルネットワークは，複数のユニットと，それらのユニットの繋がりによって構成される．入力層へ入力されるデータは入力信号，出力層から出力されるデータは出力信号と呼ばれる．入力層と出力層の間の層を中間層と呼ぶ．統計学の立場からは入力信号は予測変数，独立変数，説明変数などと呼ばれ，出力信号は基準変数，目的変数，あるいは従属変数，被説明変数などと呼ばれる．入力信号は入力層から出力層までの各ユニットで変換されながら順に伝達され，逆戻りすることはない．

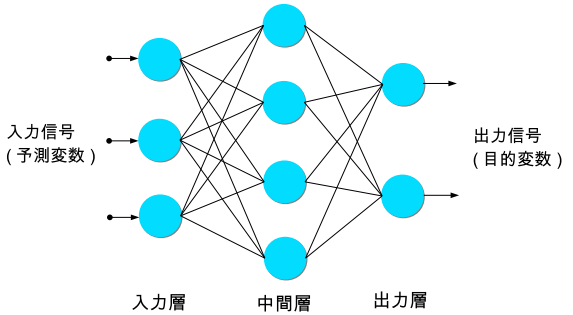

図 1 ニューラルネットワーク

階層型ニューラルネットワークの学習は，ユニット内部での信号の変換とユニット間の信号の変換によって行われる．x, y をそれぞれ入力信号，出力信号とする．ユニット内部では，

$$y_{ij} = \frac{1}{1 + exp\{-x_{ij} - \beta_{ij}\}} \tag{1}$$

のような変換が行われ，情報が出力される．ここで添字 i は層を表現している $(i = 1, 2, \ldots, a)$．添字 j は，第 i 層を形成する個々のユニットである．第 i 番目の層を形成するユニットの数は b_i あるものとする $(j = 1, 2, \ldots, b_i)$．β_{ij} は，ユニット ij の閾値である．

階層型ニューラルネットワークでは，同一層内のユニット間では情報の伝達はなく，第 1 層から第 a 層まで情報が順に伝達される．第 $i-1$ 層の全ユニットから出力された信号は，

$$x_{ij} = \sum_{k=1}^{b_i - 1} w_{ijk} y_{(i-1)k} \tag{2}$$

のように変換され，第 i 層の j 番目のユニットの入力信号となる．添字 k は，第 $i-1$ 層の各ユニットを表している．w_{ijk} は第 $i-1$ 層の k 番目のユニットから第 i 層の j 番目のユニットの結びつきの強さを表す重みである．以上のように入力信号と出力信号の変換を第 1 層から第 a 層まで繰り返し，最終的な出力を得る．

階層型ニューラルネットワークは中間層の層数および中間層のユニット数が学習に直接影響を与える．中間層のユニット数を増やすほど，より複雑な学習が可能となる．しかし通

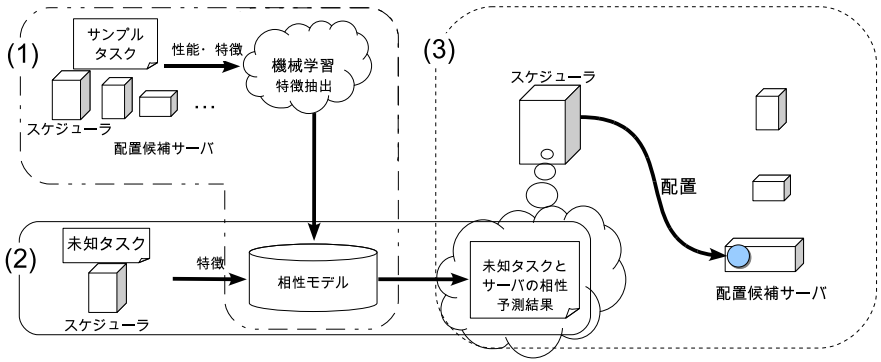

図 2 機械学習で構築した予測モデルとその適用による予測結果取得

常，学習サンプルには測定ミスなどによる例外データが混ざっている．これらを多数の中間層と中間ユニット数で学習しようとすると，例外データも含めた過学習となってしまう．

そのため，学習の際には中間層の層数，ユニット数を適切に設定することが重要である．これらは入力データの構造によるので一概にいくつが良いと言うことはできないが，一般的には 3 層の階層型ニューラルネットワークでかなりの問題が解けると言われている．

3. 提 案 手 法

3.1 概　　要

提案手法の概要は次のとおりである (**図 2**)．まず，(1) サーバ運用前にスケジューラと各サーバでの各種タスクの処理の特徴と性能を取得し，機械学習を用いて処理の特徴と各サーバの相性予測モデルを構築する．次に，(2) サーバ運用時にはスケジューラで処理内容が未知のタスク処理の特徴を取得し，運用前に構築した相性予測モデルに適用させ，そのタスクと計算機の相性を予測する．最後に，(3) 予測された相性をもとにタスク配置を行う．本稿では相性として，タスクの各サーバでの相対的な MIPS 値を用いる．すなわち，タスクの MIPS 値が高いサーバはそのタスクにとって相性が高く (良く)，MIPS 値の低いサーバはそのタスクにとって相性が低い (悪い) とする．

なお，本稿で扱うタスクは，タスク間に依存関係がない，独立したタスクである．また，タスク配置の効果に注力するため，ひとつのサーバでは複数のタスクを並列実行・並行実行はしないこととする．次節以降で，(1) 機械学習による相性抽出，(2) 未知タスクとサーバの相性予測，(3) 予測結果を用いたタスク配置のそれぞれについて述べる．

3.2 機械学習による相性抽出

本手法は，まずサーバの運用前にスケジューラ上で実行した場合の各タスクの特徴から，各サーバ上で実行した際の相対性能，すなわち相性を予測するモデルを機械学習により構築する．タスクの特徴は，プロファイラを用いてサンプルタスクのパフォーマンスカウンタの値を観察することで抽出する．相性予測の手順は以下の通りである．

(1) スケジューラで全サンプルタスクを実行しパフォーマンスカウンタを取得：

スケジューラで全サンプルタスクを実行し，実行中の全パフォーマンスカウンタ値と実行命令数をプロファイラを用いて取得する．特徴の取得に用いるカウンタ数が多いと，タスクの特徴取得に膨大な時間が必要となるため，本稿では以下のようにして使用するカウンタを選別する．

(1-1)全てのパフォーマンスカウンタ値を取得し選別：スケジューラにおいて全サンプルタスクの全パフォーマンスカウンタ値を取得する．値が取得できなかったものや全タスクで極端に小さい値だけのもの，つまり取得しても意味がないと考えられるものを除外する．

(1-2)相関係数を算出し選別：残ったパフォーマンスカウンタ間の相関係数を算出する．例えばカウンタ A とカウンタ B に強い相関関係があることが分かれば，カウンタ A の値が変化するとそれと同等な変化をカウンタ B もとるため，両方のカウンタを予測に用いても意味がない．そこで，カウンタどうしの平均の相関係数が高いものを除外する．1 回の相関係数による選別で十分にカウンタの数が減らせない場合は数回この選別を行う．本稿では，相関係数の絶対値が 0.5 以下であるものを 2 カウンタ間の相関が弱いとみなし，使用するカウンタとして選別した．

(2) 各サーバで全サンプルタスクを実行し実行時間と実行命令数を取得：

各サーバでは実行タスクの実行時間と実行命令数を取得し，単位時間当たりの実行命令数，つまり MIPS 値を算出する．

(3) 予測変数と目的変数の作成：

スケジューラで取得したパフォーマンスカウンタ値と実行命令数から単位命令数あたりのカウンタ値を求め，それを予測変数とする．また，全サーバの全サンプルタスクをにおける MIPS 値の最大値を 1 として，各サーバが各サンプルタスクを処理する際の MIPS 値の相対性能を算出し，それを目的変数とする．より詳細には，予測変数と目的変数は次のように定義する．

$c_{ij}(i=1,\ldots,p,j=1,\ldots,q)$ をアプリケーション i，パフォーマンスカウンタ j のカウンタ値とする．また，$t_{ik}(i=1,\ldots,p,k=1,\ldots,n)$ をアプリケーション i のサーバ k での実行時間と定義する．これらはスケジューラ S におけるアプリケーション i の実行命令数 I_{iS} と，サーバ k でのアプリケーション i の実行命令数 I_{ik} をを用いて，以下のように単位実行命令数あたりのカウンタ数と単位実行命令数あたりの実行時間すなわち MIPS 値に変換する．

$$c'_{ij} = \frac{c_{ij}}{I_{iS}} \tag{3}$$

$$t'_{ik} = \frac{I_{ik}}{t_{ik}} \tag{4}$$

上記の変数は，学習の精度を良くするため，0 から 1 の間の値を取るようにスケーリングすることが望ましい．そこで，次の手順でスケーリングする[2)]．

アプリケーション a のカウンタ b の値 c'_{ab} は，全アプリケーションのカウンタ b の値 $c'_{ib}(i=1,2,\ldots,q)$ の最大値である $max(c'_{ib}:i=1,\ldots,q)$ と最小値である $min(c'_{ib}:i=1,\ldots,q)$ を用いてスケーリングする．また，サーバ c におけるアプリケーション a の相対性能 t'_{ac} については，全 MIPS 値 $t'_{ik}(i=1,\ldots,q,k=1,\ldots,n)$ の最大値 $max(t'_{ik}:i=1,\ldots,q,k=1,\ldots,n)$ と最小値 $min(t'_{ik}:i=1,\ldots,q,k=1,\ldots,n$ を用いてスケーリングを行う．最終的に，

$$x_{ab} = \frac{c'_{ab} - min(c'_{ib}:i=1,\ldots,q)}{max(c'_{ib}:i=1,\ldots,q) - min(c'_{ib}:i=1,\ldots,q)} \tag{5}$$

$$y_{ac} = \frac{t'_{ac} - min(t'_{ik}:i=1,\ldots,q,k=1,\ldots,n)}{max(t'_{ik}:i=1,\ldots,q,k=1,\ldots,n) - min(t'_{ik}:i=1,\ldots,q,k=1,\ldots,n)} \tag{6}$$

となる．機械学習では，このスケーリングされた x と y をそれぞれ，予測変数と目的変数の教師値として用いる．

(4) 機械学習によるタスクとサーバの相性抽出：

作成した予測変数と目的変数を教師値としてニューラルネットワークによる機械学習を行い，タスクの特徴と各サーバの相性を抽出し，相性の予測モデルを構築する．構築する相性予測モデルはニューラルネットワークやベイジアンネットワークが適していると考えられる．

ここで構築した相性予測モデルは，スケジューラで未知タスクのパフォーマンスカウンタ値を適用することで未知タスクと各サーバの相性を予測することができるものとなる．サーバ運用時には，以上のようにして構築した相性予測モデルを用いて予測することで，未知タスクと各サーバの相性を予測しつつタスク配置を行う．この予測手法について次節で述べる．

3.3 未知タスクとサーバの相性予測

本節ではサーバ運用時にクライアントから実行要求が送られた処理の特徴が未知のタスクをスケジューラから各サーバに配置する方法を述べる．

(1) スケジューラで未知タスクをサンプリング実行し特徴を取得：

未知タスクをスケジューラで数秒間サンプリング実行し，パフォーマンスカウンタの値と実行命令数を取得する．なお，本稿では一度のタスク実行でひとつのパフォーマンスカウンタしか取得できないため，ひとつの未知タスクは，取得するパフォーマンスカウンタの数の回数だけ実行する必要がある．

(2) 各サーバでの相対的な実行性能 (相性) を予測：

取得したカウンタ値は予測変数と同様に変換し，学習に用いたサンプルタスクのカウンタ値を用いてスケーリングする．その値を運用前に構築した相性予測モデルに適用することで未知タスクと各サーバでの相対的な実行性能 (相性) の予測を得る．

以上のようにして未知タスクとそれぞれのサーバとの相性を予測する．

3.4 予測結果を用いたタスク配置

未知タスクの特徴をモデルに適用することでそれぞれのサーバとの相性の予測結果を取得した．これを用いて，基本的には最も相性の良い，すなわち予測された相対性能が最も良いサーバのキューへタスクを投入する．しかし，配置が少数のサーバに偏ってしまうと，タスクが配置されていないサーバはタスクを実行せずに使用されない状態になってしまうため，サーバ全体のスループットが低下する．**図 3** にこのような場合の本手法での解決法を示す．サーバ C 以外に相性予測が偏り，サーバ C にタスクが配置されていない状態を想定する．この時，サーバ A，B に配置されているタスクの相性予測の結果を参照し，その中でサーバ C での相対性能値 (相性) が最も高いタスクをそのサーバのキューから削除し，C のキューへ投入する．これにより，予測結果を用いたタスク配置手法では，タスクの実行が終了したサーバに，各サーバでの相対性能が予測されたタスクのうち，そのサーバでなるべく相対性能が良いタスクを配置することができる．

4. 評　　価

提案手法の有効性を評価するため，性能がヘテロジニアスなサーバ環境上で，実際に機械学習により相性予測モデルを構築し，相性の予測精度を評価する．また，予測結果を用いた

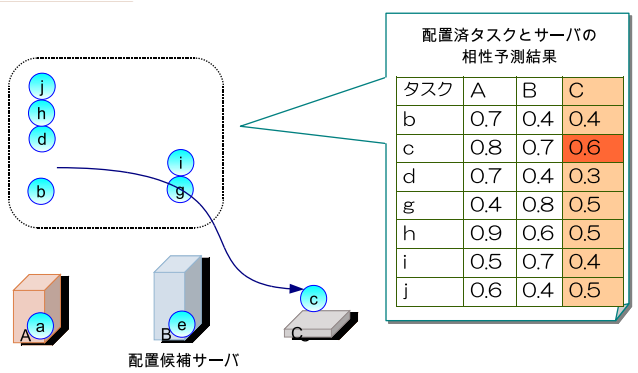

図 3 予測結果を用いたタスク配置

タスク配置を行い，その妥当性を検討する．

4.1 評価環境

本研究で提案する手法の評価のため，AMD 社の Phenom，PhenomII X6，Intel 社の Core i7 プロセッサを用い，それぞれストレージやメモリ性能などを変え，ヘテロジニアスな性能のサーバが複数台混在する実験環境を構築した．**表 1** に実験環境の一覧を示す．表中 $M_k(k = 1, 2, \ldots)$ はタスクを実行するサーバを表し，S はスケジューラを表す．なお，評価に用いるサーバは全てマルチコア CPU を搭載しているが，提案手法であるタスクとサーバの相性予測，それに基づいたタスク配置の効果の評価に注力するため，本稿では各サーバの CPU は 1 つのコアのみを用いることにする．本稿では統計解析環境 R[3)2)4)] を用いて機械学習を行い，相性予測モデルを構築する．相性予測モデルへの未知タスク特徴の適用にも R を用いる．また，パフォーマンスカウンタの取得には *Oprofile*[5)] を用いる．

様々な未知のタスクの相性を予測するため，学習には様々な特徴を持つタスクを選択する必要がある．そこで本稿では相性予測モデルの学習と評価に用いるカウンタ値を取得するためのタスクとして，広い分野のアプリケーションを網羅しているベンチマークである SPEC 2000CPU と SPEC 2006CPU を選択し，これらの中から整数・浮動小数点ベンチマークアプリケーション 46 個を用いる．評価実験では 46 個のベンチマークアプリケーションを学習用と評価用に分けて使用する．

なお，特徴抽出には 3.2 節で述べた選別の結果，以下の 5 個のパフォーマンスカウンタを用いることにした．

表 1 サーバとスケジューラの構成

Machine name	CPU name CPU Freq.	Memory Freq. channel	Storage rpm	備考
M_1	PhenomIIX6 3.80 GHz	1333 MHz UnGanged	SSD	OverClock
M_2	Core i7 2.93 GHz	1066 MHz Single	HDD 5,400	
M_3	Core i7 1.60 GHz	800 MHz Single	HDD 5,400	
M_4	Core i7 2.93 GHz	1066 MHz Triple	HDD 10,000	
S	Phenom 2.60 GHz	1066 MHz Ganged	HDD 7,200	

- DATA_ CACHE_ REFILLS_ FROM _ L2_ OR_ NORTHBRIDGE (L2 キャッシュもしくはノースブリッジからの L1 データキャッシュ置換要求回数)
- L1_ DTLB_ MISS_ AND_ L2_ DTLB_ HIT (L1 データ TLB がミスし，L2 データ TLB でヒットした回数)
- INSTRUCTION_ CACHE_ MISSES (命令キャッシュミス回数)
- RETURN_ STACK_ HITS (分岐予測に用いる，分岐命令のリターンアドレスを保持するリターンアドレススタックにヒットした回数)
- DECODER_ EMPTY (命令デコーダが空になった回数)

4.2 予測モデルの予測精度評価

4.2.1 評価方法

提案手法のうち，機械学習で構築した相性予測モデルを評価する．評価方法を以下に述べる．

(1) 一部のデータで予測モデルを構築：
本稿では 46 個のタスクのパフォーマンスカウンタ値と性能を取得している．評価実験では，まずこの中から 26 個を選択し，26 個のタスクのスケジューラ上でのカウンタ値と各サーバでの相対性能値を予測変数・目的変数の教師値として機械学習で相性を抽出し相性予測モデルを構築する．

(2) 学習に用いなかったデータを予測モデルに適用：
学習に用いなかった 20 個のタスクのスケジューラでのカウンタ値を相性予測モデルに適用させ，各サーバでの相対性能値の予測値を得る．

表 2 目的変数の本来の相対性能値と予測された相対性能値

タスク名	M_1			M_2			M_3			M_4		
	教師値	予測値	エラー	教師値	予測値	エラー	教師値	予測値	エラー	教師値	予測値	エラー
gcc	0.1240	0.0737	0.0503	0.0941	0.0473	0.0469	0.2095	0.1366	0.0729	0.0900	0.0520	0.0381
gobmk	0.0789	0.0715	0.0074	0.0820	0.0459	0.0361	0.2073	0.1329	0.0744	0.0906	0.0506	0.0400
gromacs	0.1064	0.1329	0.0266	0.1259	0.0848	0.0411	0.2935	0.2253	0.0682	0.1262	0.0901	0.0361
h264	0.0394	0.0735	0.0341	0.0392	0.0472	0.0079	0.4283	0.1363	0.2920	0.0446	0.0518	0.0072
hmmer	0.0500	0.1549	0.1049	0.0729	0.0991	0.0261	0.1922	0.2665	0.0742	0.0853	0.1025	0.0172
lbmr	0.1797	0.1132	0.0664	0.0610	0.0720	0.0111	0.1604	0.2010	0.0406	0.0685	0.0767	0.0082
leslie	0.0484	0.0860	0.0376	0.0515	0.0485	0.0031	0.1462	0.1180	0.0282	0.0565	0.0595	0.0030
libquantum	0.0572	0.0755	0.0183	0.0380	0.0410	0.0030	0.1061	0.0939	0.0121	0.0348	0.0531	0.0183
mcf	0.7321	0.4198	0.3123	0.2817	0.3042	0.0225	0.5469	0.6191	0.0722	0.2762	0.2801	0.0039
milc	0.2066	0.0855	0.1211	0.1667	0.0542	0.1124	0.2988	0.1532	0.1456	0.1248	0.0595	0.0653
namd	0.0297	0.1105	0.0808	0.0299	0.0698	0.0399	0.1094	0.1906	0.0812	0.0378	0.0754	0.0376
omnetpp	0.2291	0.1044	0.1247	0.1355	0.0667	0.0688	0.2689	0.1878	0.0810	0.1308	0.0713	0.0595
perlbench	0.0313	0.0789	0.0476	0.0298	0.0506	0.0207	0.1097	0.1455	0.0359	0.0400	0.0553	0.0153
povray	0.0537	0.0630	0.0093	0.0582	0.0406	0.0176	0.1646	0.1180	0.0466	0.0682	0.0451	0.0231
sjeng	0.0761	0.0782	0.0021	0.0601	0.0501	0.0100	0.1652	0.1442	0.0210	0.0698	0.0548	0.0150
soplex	0.2100	0.1681	0.0418	0.1176	0.1116	0.0061	0.2440	0.3002	0.0562	0.1115	0.1118	0.0003
sphinx	0.0740	0.0990	0.0250	0.0425	0.0632	0.0208	0.1301	0.1791	0.0490	0.0527	0.0679	0.0152
tonto	0.0569	0.0897	0.0327	0.0695	0.0573	0.0122	0.1849	0.1636	0.0213	0.0698	0.0620	0.0077
wrf	0.0568	0.1133	0.0565	0.0609	0.0729	0.0119	0.1668	0.2029	0.0361	0.0609	0.0772	0.0163
zeusmp	0.0936	0.0600	0.0336	0.0668	0.0404	0.0264	0.1678	0.0569	0.1108	0.0676	0.0566	0.0109

(3) 相関係数を用いた機械学習による予測精度評価：

すでに取得されている学習に用いなかった 20 個のタスクの本来の各サーバでの相対性能と予測された相対性能値の相関係数を算出する．本稿ではこの相関係数の値を，提案手法による相対性能予測精度の評価指標とする．

また，配置候補サーバが 2 台，3 台，4 台の場合をそれぞれ評価する．

4.2.2 評 価 結 果

ここでは配置先サーバが 4 台の場合で構築した相性予測モデルの評価結果を述べる．なお，使用したサーバは表 1 の S 以外の 4 台である．**表 2** に学習に用いなかった 20 個のタスクの目的変数の本来の値 (教師値) と予測値を示す．例えば，gcc ベンチマークにおける M_1 の相対性能値は 0.1240，予測された相対性能値は 0.0737 であり，誤差の値は 0.0503 である．教師値と予測値の相関係数を算出したところ，0.8 であった．これは一般に強い相関関係があると言われる相関係数の絶対値 0.7 以上である．各タスクで多少の誤差はあるが，学習に用いていないタスクを予測しているにも関わらず，0.8 という高い相関係数で予測ができていることから，本研究での提案手法はサーバ間での未知タスクの相対性能を予測する上で有用な手法となり得ると考えられる．

4.3 予測結果を用いたタスク配置評価

4.3.1 評 価 方 法

予測精度評価で構築した相性予測モデルを用いて，学習に用いていない 20 個のタスクをの相性を予測・配置する．以下のようにして評価を行った．

- 順次配置手法との比較：

  2.1 節で述べた順次配置手法による 20 個のタスクの配置と，提案手法である相性の予測結果をもとにした 20 個のタスクの配置を比較する．比較は全タスク終了までのそれぞれの手法による配置での実行時間で行う．

- 数ケースのタスク並び順での評価：

  順次配置の実行時間は 20 個のタスクの初期並びに依存する．そのため，配置は 20 個のタスクの順番をランダムに入れ替えた数ケースで行う．順次配置での全タスク終了

までの実行時間を基準とした相対実行時間を算出し，数ケースでの平均相対実行時間を評価する．

なお，提案手法は未知タスクの実行要求がスケジューラに送られてから未知タスクをサンプリング実行し，特徴を抽出するため，サンプリング実行による時間が全タスク終了までのオーバヘッドとなる可能性がある．しかし，サーバがタスクを実行している状態であり，既に配置されたタスクがサーバのリソース待ち状態であれば，このオーバヘッドは隠蔽されると考えられる．本研究では複数のタスク実行要求がクライアントから常時送られてくる状況を想定しており，サーバ運用時にはオーバヘッドが隠蔽されると考えられるため，評価実験においてはオーバヘッドを考慮しないこととする．また，スケジューラからサーバへのタスク配置にかかる時間のオーバヘッドも実際には存在するが，提案手法による配置の評価に注力するため，このオーバヘッドも無視することとする．

4.3.2 評価結果

複数のタスク順のケースで順次配置を行い，それぞれのケースで提案手法の配置と実行時間を比較する．配置結果のガントチャートのうちひとつを**図 4** に示す． 図 4 は横軸に実行時間，縦軸に配置サーバをもつガントチャートである．図中のタスク名はそこでの実行タスクを表す．また，各チャートの色の濃淡によって各タスクが何番目に相性の良いサーバで実行されたかを表す．この相性の良し悪しは各タスクの本来の相性を用いて示している．図 4 より，順次配置手法と提案手法のガントチャートを比較すると，提案手法での配置のほうが全体的に色が濃く，多くのタスクが相性の良いサーバに配置されたことが分かる．そして全タスク終了までの実行時間は順次配置手法と比較して 5% ほど短縮されている．

6 つのケースの実行時間比較結果とその平均を**図 5** に示す． 図 5 は並び順ケースごとに順次配置のときの全タスク終了までの実行時間を 100% とした提案手法配置の相対実行時間を表している．予測結果を用いて配置する提案手法は平均で約 8% の実行時間を短縮することができた．これより，本研究で提案したタスク配置手法は，未知のタスクをサーバに配置し，サーバ全体のスループットを向上させる上で有用な手法となり得ると考えられる．

図 5 のケース 3 とケース 6 において，提案手法は順次配置より実行時間が延びてしまっている．この要因として，タスクの各サーバでの相対性能の予測が相関係数的には高くても，多少の誤差があるためタスクが効率的に配置できなかったことが考えられる．これを検証するため，100% の精度で予測ができたと仮定し，20 個のタスクの本来の相対性能値を用いて提案手法による配置を行った結果をあわせて図 5 に示した．実行時間が延びてしまっていたケース 3 とケース 6 でも実行時間が短縮できており，平均の実行時間短縮が約 15%

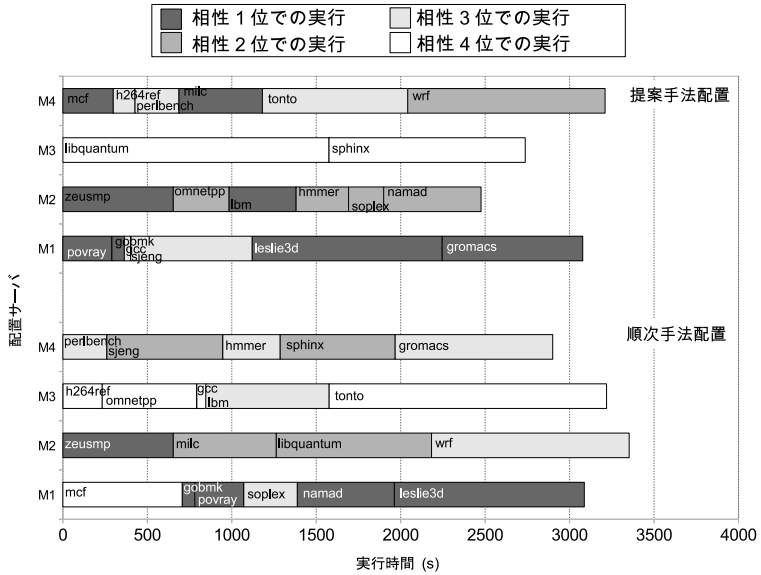

図 4 両配置手法の配置結果の比較

となった．このことから，さらに高い精度でタスクとサーバの相性が予測できれば，提案手法を用いたサーバ全体のスループットの向上の効果がさらに上がると考えられる．また他に全タスク終了までの実行時間が延びてしまった要因として，次のことが考えられる．本実験では M_3 と各タスクの本来の相性が 3 位と 4 位のものばかりであった．つまり相対的にタスクの実行性能が劣っており，M_3 に実行時間の長いタスクが多く配置されてしまうと，それだけで全体の性能が落ちてしまう可能性がある．しかし本研究では前述のとおり複数のタスク実行要求が常時クライアントから送られてくる状況を想定しているため，20 個のタスクで実行時間が長くなる結果であっても，タスク数が多くなればなるほど，他のサーバで相性の良いタスクを実行できる割合が増加し，本研究の目的であるサーバ全体でのスループットの向上が十分実現できると考えられる．

5. 関連研究

本稿ではタスクとサーバの相性を予測するために機械学習によるアプローチをしている

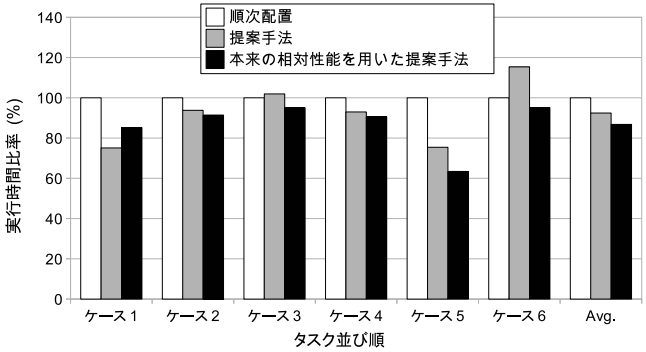

図 5 順次配置と提案手法との比較

が，機械学習を用いた他の研究には，プログラム実行時における最適化のためにハードウェアから得られる動的な情報と，コンパイラから得られるプログラムのフェーズの区切りといった静的な情報から機械学習により精度の高い最適化を行う研究もある[6]．この研究では提案手法を低消費電力化手法に適用しその有用性を確かめ，また，ハードウェアプリフェッチによる性能の最適化のためのモデルの構築を検討している．その他にも，機械学習により最適なコンパイラオプションを選択する研究[7]や，SMT (Simultaneous MultiThreading) プロセッサにおけるリソース競合のモデル化に機械学習を利用する研究も行われている[8]．また，マルチコアプロセッサの共有資源管理に機械学習を用いる研究[9]もある．

データセンタの消費電力削減を目的に，Service Level Agreement (SLA) を考慮したタスクスケジューリングの研究も行われている[10]．この研究では，ヘテロジニアスなサーバ環境下において，タスクをマイグレーションした際の SLA に対する許容度を予測するために，CPU 使用率と SLA 許容度の関係を線形回帰分析を用いた機械学習によりモデル化している．ただし，各サーバとの相性は考慮されておらず，この点で本研究とは異なるものである．

6. まとめと今後の課題

データセンタは徐々に行われるハードウェア更新により，性能的にヘテロジニアスなサーバ計算機環境になることが多い．また，データセンタで処理されるタスクは，メモリがボトルネックになるものなど様々な処理の特徴を持っている．そのため，タスクの特徴によって，タスク処理の際に最大の MIPS 値を得られる実行サーバが変わり得る．本稿では，タスク処理の特徴と各サーバの相性を機械学習により予測し，それに基づいてサーバへタスク配置を行うことで，サーバ全体のスループットを向上させる手法を提案した．また，本手法により高精度で相性予測が可能であることを示し，予測結果を用いたタスク配置の評価より，本手法はサーバ全体のスループットの向上させる上で有用な手法となり得ることを示した．

本手法の今後の課題として次のことが挙げられる．本稿で用いタスクは CPU の性能評価を行うもののみであるため，機械学習し構築した相性予測モデルは，例えばディスク I/O や通信を要するタスクには対応できないと考えられる．そのため，例えばディスク I/O が頻発するタスクなど本稿では用いていない特徴をもつタスクも用いて本手法の有効性を検証する必要がある．さらに高精度での予測を行うために，例えばベイジアンネットワークなど異なる手法を用いた機械学習で本手法を検証する必要もある．仮想化されたデータセンタ運用への本手法の応用の検証も今後の課題である．

謝辞 本研究の一部は，科学技術振興機構・戦略的創造研究推進事業 (CREST) の研究プロジェクト「革新的電源制御による超低電力高性能システム LSI の研究」，および文部科学省科学研究費補助金 (若手研究 (B) No.21700055) の支援によって行われた．

参 考 文 献

1) 銅谷賢治：脳の学習機構の理解を目指して，数理科学 NO.505-513 (2005).
2) 豊田秀樹：データマイニング入門，東京書籍 (2008).
3) R: . http://www.r-project.org.
4) 舟尾暢男，高浪洋平：データ解析環境「R」，工学社 (2005).
5) Oprofile: . http://oprofile.sourceforge.net/.
6) 佐々木広，池田佳路，近藤正章，中村　宏：統計情報に基づく実行時最適化の検討，情報処理学会研究報告 ARC-169，pp.175–180 (2006).
7) Pinkers, R.P.J., Knijnenburg, P.M.W., Haneda, M., Wijshoff, H.A.G.：Statistical selection of compiler options, *MASCOTS*, pp.494–501 (2004).
8) Moseley, T., Kiihm, J.L., Connors, D.A. and Grunwald, D.: Methods for Modeling Resource Contention on Simultaneous Multithreading Processors, *ICCD*, pp. 373–380 (2005).
9) Martinez, J. and Ipek, E.: Dynamic Multicore Resource Management: A Machine Learning Approach, *IEEE Micro*, pp.8–17 (2009).
10) Berral, J.L., Goiri, I., Nou, R., Julia, F., Guitart, J., Gavalda, R. and Torres, J.: Towards energy-aware scheduling in data centers using machine learning, *e-Energy'10*, pp.215–224 (2010).