平成26年度 修士論文

アプリケーションの特徴に基づいた
仮想マシン再配置手法の研究

電気通信大学 大学院情報システム学研究科
情報システム基盤学専攻

学籍番号　　：1353026
氏名　　：藤井 淳
主任指導教員：本多 弘樹 教授
指導教員　　：古賀 久志 准教授
指導教員　　：新谷 隆彦 准教授
提出日　　：平成27年 1月 26日

要旨

近年、データセンタでは仮想化技術を用いて 1 台の物理サーバに複数の仮想マシン（VM）を動作させており、その際の課題の 1 つに VM の適切なサーバへの再配置が挙げられる。

データセンタで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が、一度に全ハードウェアを更新することは稀であり、通常は徐々に入れ替えが行われるためデータセンタはヘテロジニアスなサーバ環境であることが多い。そのため、VM 上で動作させるアプリケーションの処理性能は VM を配置するサーバによって異なると考えられ、VM 上で動作させるアプリケーションの特徴によって、その処理性能を最大にする配置サーバが変わり得る。

また VM 間ではサーバのメモリやキャッシュなどのリソースを共有しているため、同一のサーバに配置されている VM で動作させるアプリケーションの特徴の組み合わせによってもその処理性能は変わることが考えられる。

本研究では、同じサーバに同時に配置する VM によって発生するリソース競合による影響、ライブマイグレーションする VM とライブマイグレーション先サーバで既に配置されている VM によるリソース競合による影響、アプリケーションとサーバ構成との相性の 3 つを考慮した仮想マシン再配置手法を提案する。

初期配置の際にリソース競合が発生する VM が偏っている場合、リソース競合が発生する VM が途中で増加する場合について実験を行い、再配置を行わない場合と比べて最大で 26%VM 上で実行されるアプリケーションの実行時間を短縮することができ、データセンタのスループットを向上可能であることを示した。

目 次

図 目 次

表 目 次

1　はじめに

1.1　研究背景

近年、クラウドサービスの急速な普及にともない、データセンタの重要性が高まっている [9]。そのためデータセンタではサーバ台数が増加し、消費電力や設置スペースが問題となっており、仮想化技術を用いて 1 台の物理サーバに複数の仮想マシン (VM) を動作させている。この際の課題の 1 つに VM の適切なサーバへの再配置が挙げられる。

また、データセンタで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が、一度に全ハードウェアを更新することは稀であり、通常は徐々に入れ替えが行われるため、データセンタはヘテロジニアスなサーバ環境であることが多い [11]。

そのため、VM 上で動作させるアプリケーションの処理性能は VM を配置するサーバによって異なると考えられ、VM 上で動作させるアプリケーションの特徴によって、その処理性能を最大にする配置サーバが変わり得る。

さらに VM 間ではサーバのメモリやキャッシュなどのハードウェアリソースを共有しているため、同一のサーバに配置されている VM で動作させるアプリケーションの特徴の組み合わせによってもその処理性能は変わることが考えられる。

ヘテロジニアスなサーバ計算機環境のデータセンタのスループットを向上させるには、ハードウェアリソース競合を避けるために VM 間で共有するリソースを多く使用する VM を同じサーバ計算機に配置しないようにし、VM 上で実行するアプリケーションの処理の特徴に適したサーバ計算機にその VM を配置することが求められる。

1.2　研究目的

本研究は性能の異なるサーバが混在するヘテロジニアスなサーバ計算機環境において、与えられた複数のアプリケーションに対してそれぞれのアプリケーションの処理の特徴に適したサーバ計算機への再配置手法を提案し、与えられた全アプリケー

ション終了までの実行時間を短縮することで、データセンタ全体のスループットを向上させることである。

1.3　関連研究

仮想マシン再配置に関する研究は数多く存在する。

Xu ら [10] は、ライブマイグレーションにかかる時間と、同時に配置される VM 間の影響を最小限にするために、ライブマイグレーションにかかる時間と同時に配置される VM 間の影響を数理モデルを作成することで見積もり、再配置を行う手法を提案している。

Novaković ら [8] は、物理サーバ上で動作中の VM をコピーして物理サーバ上で単体動作させた場合の性能を取得し、コピー元の VM の状態と比較することでコピー元の VM でリソース競合による性能低下を検知することで再配置を行う手法を提案している。

これらの研究では、使用する物理サーバが全て同じ構成であるホモジニアスなサーバ計算機環境を想定しており、本研究ではヘテロジニアスなサーバ計算機環境を考慮しているため既存研究とは異なる。

本研究ではアプリケーションの処理の特徴に着目して VM を再配置する手法を提案するが、アプリケーションの処理の特徴に着目してアプリケーションの実行を行う物理サーバを判断する研究は存在する [12]。

アプリケーションを実行する VM を物理サーバに配置すること、アプリケーションを実行する物理サーバを判断することの違いとして、VM はライブマイグレーションが可能であるため一度配置を決めたあとで再配置が可能である点、複数の VM を同じ物理サーバに配置した場合にリソース競合が発生する可能性を考慮する必要がある点で異なる。

1.4　本論文の構成

本論文の構成は以下のとおりである。2 章では、従来のデータセンタでの VM 配置方法と仮想マシン再配置について述べる。3 章では、提案する仮想マシン再配置

手法について述べる。4 章では、提案手法に対する評価実験に対する結果と考察について述べる。5 章では、まとめと今後の課題について述べる。

2　仮想化技術と仮想マシン再配置

本章ではデータセンタでの VM 配置方法と VM 再配置の必要性についてのべる。

2.1　データセンタでの VM 実行サーバの決め方

データセンタでは消費電力・設置スペースなどの金銭的なコストを削減するために、仮想化技術を用いて 1 台の物理サーバに複数の VM を動作させている。

一例として、OpenStack[2] での VM の実行サーバ決定の流れを図 1 に示す。ユーザ (ex. 企業、一般人) から VM リクエストが来た際に、VM を何処かのサーバに配置する必要があるが、仮想マシンをどの物理サーバに配置するかを決める nova-scheduler では、VM 上でどのようなアプリケーションを実行するかは事前にわからない。そのためメモリなどのリソース使用量が少ないサーバに VM を配置する。

しかし、このように単純に VM をサーバに配置してしまうと、メモリ帯域幅やネットワーク帯域幅などのハードウェアリソースの競合が発生したり、サーバ構成と VM 上で動作させるアプリケーションの特徴に最も適した物理サーバに配置できない可能性がある。

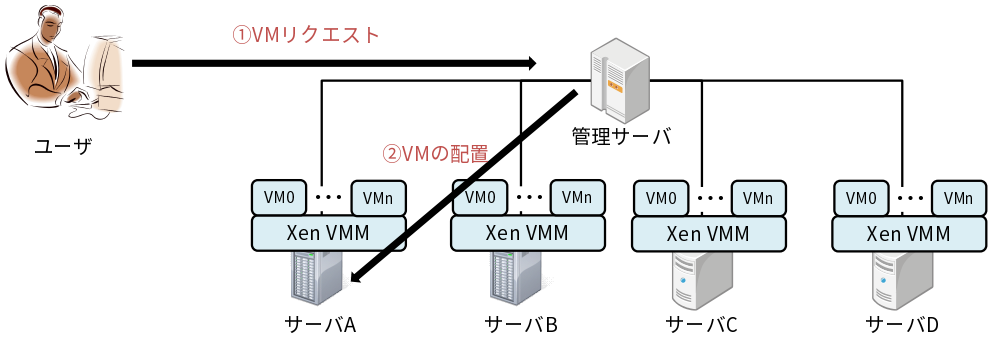

図 1: データセンタでの VM 実行サーバの決め方

2.2　仮想マシン再配置

Xen や KVM などの仮想環境ではライブマイグレーション（以後、本稿ではライブマイグレーションをマイグレーションと表記する）を用いることで、ある物理サー

バ上で稼働中の VM をサービスの停止を伴わずに別の物理サーバ上に移動させることが可能である。そのため初期配置においてある物理サーバでリソース競合が発生していることが判明した場合には、マイグレーションを行いリソース競合を緩和することで、理想的な VM 配置に再配置することが求められる。

例えば、表 1 に示すようなサーバ構成のサーバ A とサーバ B がそれぞれ 1 台づつ存在する環境において 7 つの VM を作成し、その上でアプリケーションを実行したいとする。

表 1: サーバ構成

サーバ名	搭載 CPU	コア数	CPU 周波数 (GHz)	L3 Cache (MB)	メモリ (GB)	メモリ周波数 (MHz)
サーバ A	Xeon X5675	6	3.06(3.46)	12	12	1333
サーバ B	Xeon E5640	4	2.66(2.93)	12	12	1066

7 つのうち 4 つの VM では、SPEC CPU 2006 の CPU バウンドなアプリケーションである gobmk を動作させ、残りの 3 つの VM では SPEC CPU 2006 のメモリバウンドなアプリケーションである libquantum を動作させる。VM インスタンスのタイプは表 2 に示す 1 種類のみとし、物理サーバに CPU コア数以上の VM を配置しないものとする。

表 2: 想定 VM インスタンス

VCPU 数	メモリ (GB)	OS	kernel	仮想化方式
1	2	CentOS 5.10	Linux 2.6.18-371.12.1-Xen	準仮想化

図 2,3,4 のように配置した場合の各 VM の実行時間を計測する。なお図の灰色は CPU バウンドなアプリケーションを実行する VM を示し、黒色はメモリバウンドなアプリケーションを実行する VM であることを示す。

各サーバの各 VM の実行時間は図 5 のようになる。

図 2 の場合は、サーバ B にメモリバウンドなアプリケーションを実行する VM が集中しているため、サーバ B でメモリ帯域幅競合が発生するので VM5,VM6,VM7 の実行時間が長くなっている。

図 3 の場合には、サーバ A にもメモリバウンドなアプリケーションを実行する VM

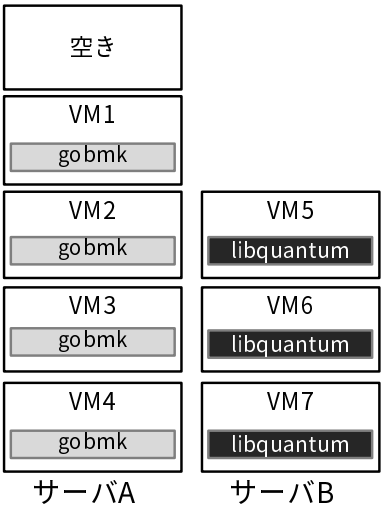

図 2: サーバ B にメモリバウンドなアプリケーションを実行する VM が偏った VM 初期配置

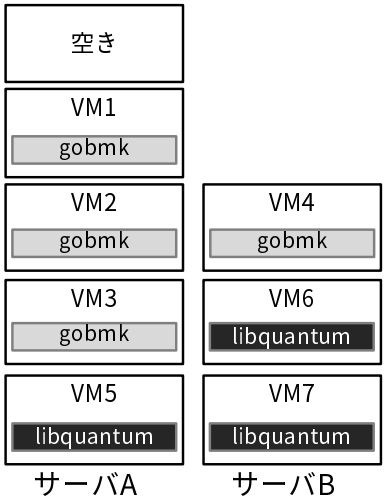

図 3: サーバ A と B でメモリバウンドなアプリケーションを実行する VM が分散している VM 初期配置 (サーバ B のほうがメモリバウンドな VM が多い場合)

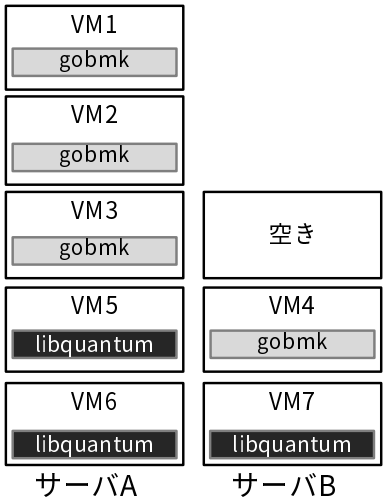

図 4: サーバ A と B でメモリバウンドなアプリケーションを実行する VM が分散している VM 初期配置 (サーバ A のほうがメモリバウンドな VM が多い場合)

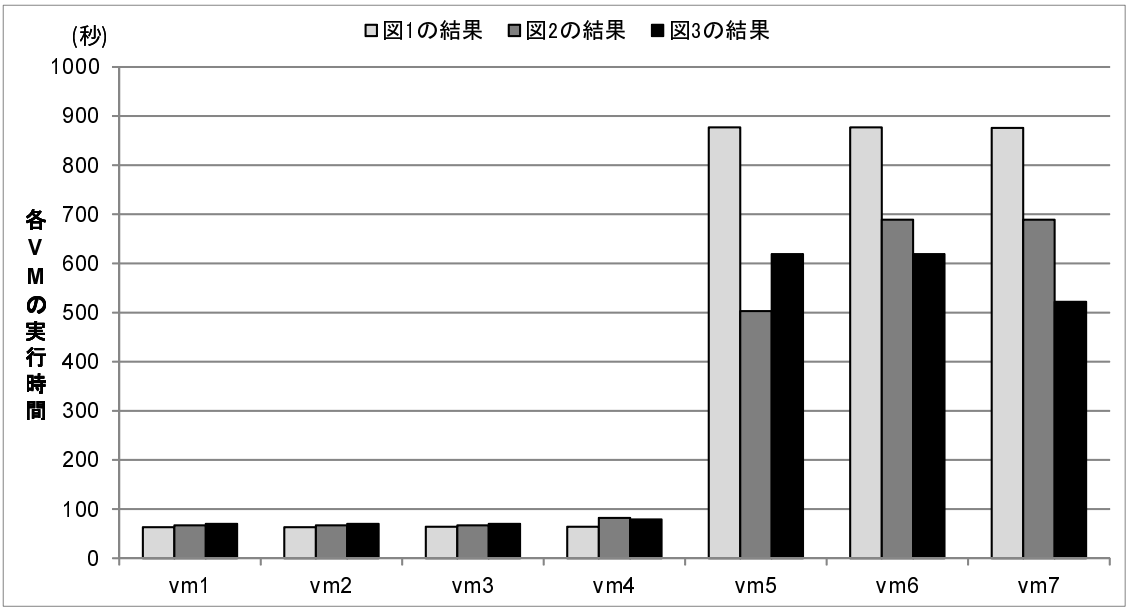

図 5: 図 2,3,4 の配置にした際の各 VM の実行時間

を配置することで、図 2 の場合よりもサーバ B でメモリ帯域幅競合が少なくなるため、図 2 よりも各 VM の実行時間が短くなっている。

図 4 の場合には、サーバ A のほうがサーバ B よりも高速なメモリ転送速度が早いため、メモリバウンドなアプリケーションを実行する VM をサーバ A に多く配置することで、VM5 は図 3 の配置よりも遅くなるが、VM6,VM7 は図 3 の配置よりも早くなる。そのためより多くの VM 上で実行するアプリケーションの実行時間を短縮することができているので、今回の場合図 4 の配置が最も望ましい。

しかし、VM 上でどのようなアプリケーションを実行するのかは実行してみるまではわからないため、初期配置の際にはリソース競合を避けて配置することができず、また物理サーバとのハードウェア的な相性を考えて配置することもできないため、VM の再配置が有効となる。

再配置が有効となる可能性のあるタイミングは次の場合が考えられる。

- VM が新たに作成され物理サーバに割り当てられた場合
- VM の実行が終了し物理サーバ上から VM が取り除かれた場合
- VM 上で実行するアプリケーションが変化した場合

これらのタイミングは事前に知ることができないため、上記のイベントが発生する毎に再配置を行う必要があるか確認する必要がある。特に、VM 上で実行するアプリケーションが変化したかは定期的に VM を監視しなければ検知ができないため、定期的に VM を監視する必要がある。

また再配置を行う際には、ライブマイグレーションを用いて VM を移動するため次のことを考えて決定する必要がある。

- マイグレーション元サーバ
- マイグレーションする VM
- マイグレーション先サーバ

3 提案手法

3.1 想定環境

本提案手法では物理サーバを管理する管理サーバ、Xen[6] をハイパーバイザとして用いたヘテロジニアスなサーバ構成の物理サーバ、VM イメージを置くための NFS サーバが存在する環境を想定している。また VM 上で実行するアプリケーションは単体の VM で動作を行い、他の VM と通信を行うことはないものとする。

サーバに配置可能な VM の最大数は、Xen の管理用 OS に割り当てる CPU コア数を除いた、(CPU コア数 -1) とし、同時にマイグレーションを行う VM の数は 1 つとする。

3.2 提案手法概要

同じ物理サーバに配置されている VM 間でリソース競合が発生すると VM の性能が低下するため、リソース競合が発生しているかを判断する必要がある。また、ヘテロジニアスなサーバ環境の場合には、VM を配置するサーバによって VM の処理性能を最大にする配置サーバが異なるため、それを考慮した VM の配置が必要である。

以上よりヘテロジニアスな物理サーバが複数混在する環境で、次のことを考慮した仮想マシン再配置手法を提案する。

- 同じサーバに配置する VM によって発生するリソース競合による影響
- マイグレーションする VM とマイグレーション先サーバで既に配置されている VM によるリソース競合による影響
- アプリケーションとサーバ構成との相性

提案手法は運用前と運用時の二段階に別れており、まず運用前にリソース競合が発生しているサーバを判定するための VM 上で動作しているアプリケーションの特徴分類モデルと、アプリケーションとサーバ構成の相性を判定するための VM 上で動作しているアプリケーションとサーバの相性モデルを機械学習を用いて作成する。

運用時には、リソース競合が発生するアプリケーションを実行する VM が特定のサーバに偏っている場合に、リソース競合が発生する VM の数を均一になるように

再配置を行い、マイグレーション先が複数存在する場合にはアプリケーションとサーバ構成の相性が最も良いサーバをマイグレーション先する。以上より、提案手法の概要について図 6 に示す。

リソース競合が発生するアプリケーションを実行する VM を均一にする理由としては、サーバ間の性能差が大き過ぎなければ、リソース競合を発生する原因となる VM の数を同数になるようにマイグレーションしたほうが、処理性能が高くなると考えられるためである。

3.3　提案手法運用前のモデルの作成

提案手法ではリソース競合が発生している VM の数を均一にするため、リソース競合が発生しているサーバを調べる必要がある。またマイグレーション先サーバを選ぶ際に、アプリケーションとサーバ構成との相性を用いてサーバを選択するため、相性を判別する必要もある。

そこで、運用前に同じ物理サーバに配置されている VM 間でのリソース競合の検知と、VM の処理性能を最大にする配置サーバの判別を行うためのモデルを作成し、リソース競合が発生しているサーバの判断と VM の処理性能を最大にする配置サーバの判断を行う。

モデルの作成には統計解析向けのプログラミング言語である R[4] を用いて機械学習を行い作成する。なお本稿では以後、モデルへの入力を特徴量、モデルからの出力を目的変数とする。

3.3.1　リソース競合が発生しているサーバの判定

物理サーバで未知のアプリケーションが実行されている際に、リソース競合が発生しているサーバかどうかを判定する流れを次に示す（図 7）。

(i) アプリケーションの特徴分類モデルを作成:
入力として単位命令数あたりのパフォーマンスカウンタの値を与えると、特徴（メモリバウンド）に分類するモデルを作成する。

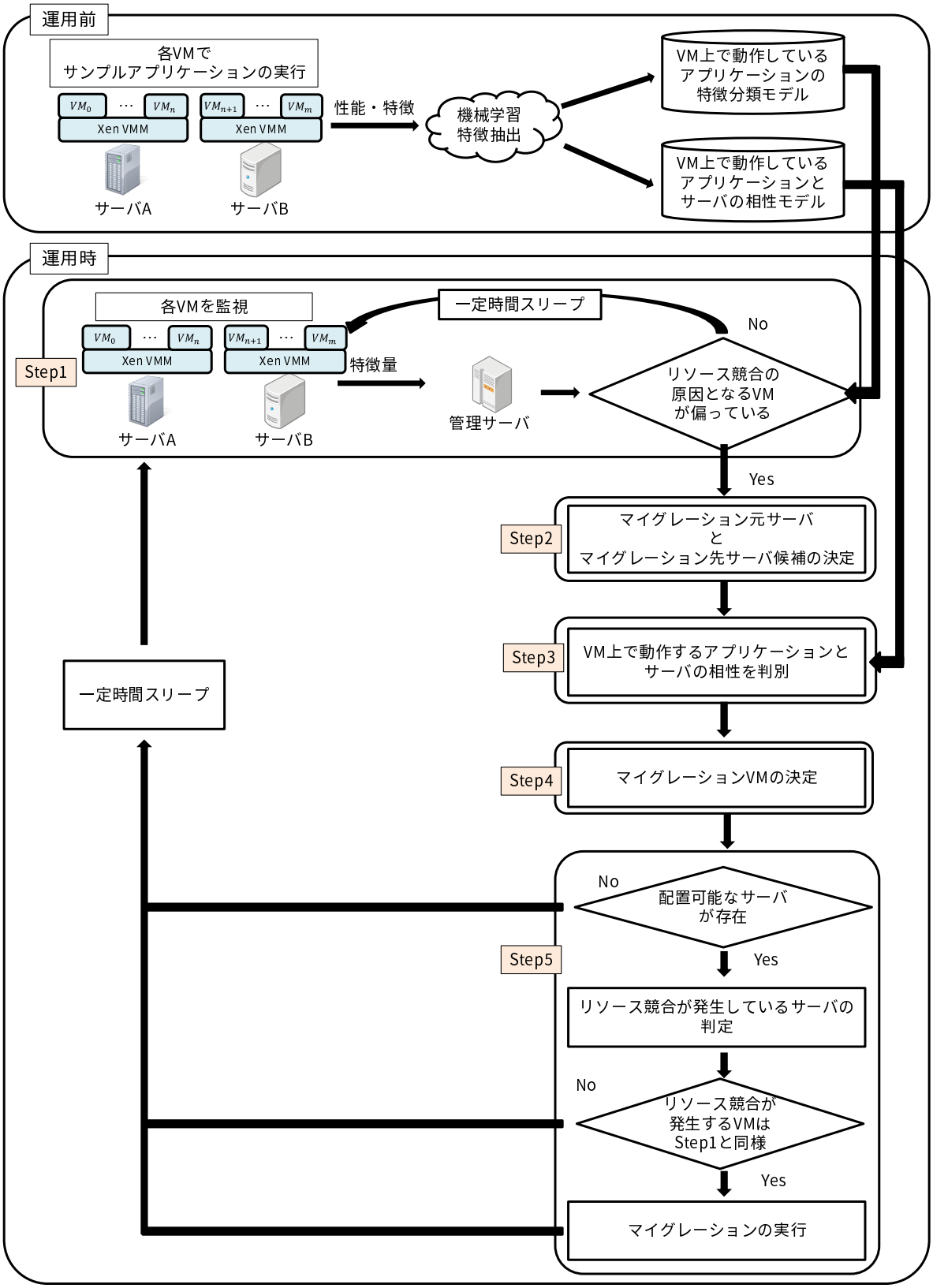
運用前
各VMで
サンプルアプリケーションの実行
VM_0 … VM_n
Xen VMM
VM_{n+1} … VM_m
Xen VMM
サーバA
サーバB
性能・特徴
機械学習
特徴抽出
VM上で動作している
アプリケーションの
特徴分類モデル
VM上で動作している
アプリケーションと
サーバの相性モデル
運用時
各VMを監視
Step1
特徴量
管理サーバ
一定時間スリープ
No
リソース競合の
原因となるVM
が偏っている
Yes
Step2
マイグレーション元サーバ
と
マイグレーション先サーバ候補の決定
Step3
VM上で動作するアプリケーションと
サーバの相性を判別
Step4
マイグレーションVMの決定
Step5
No
配置可能なサーバ
が存在
Yes
リソース競合が発生しているサーバの
判定
No
リソース競合が
発生するVMは
Step1と同様
Yes
マイグレーションの実行
一定時間スリープ

図 6: 提案手法の概要

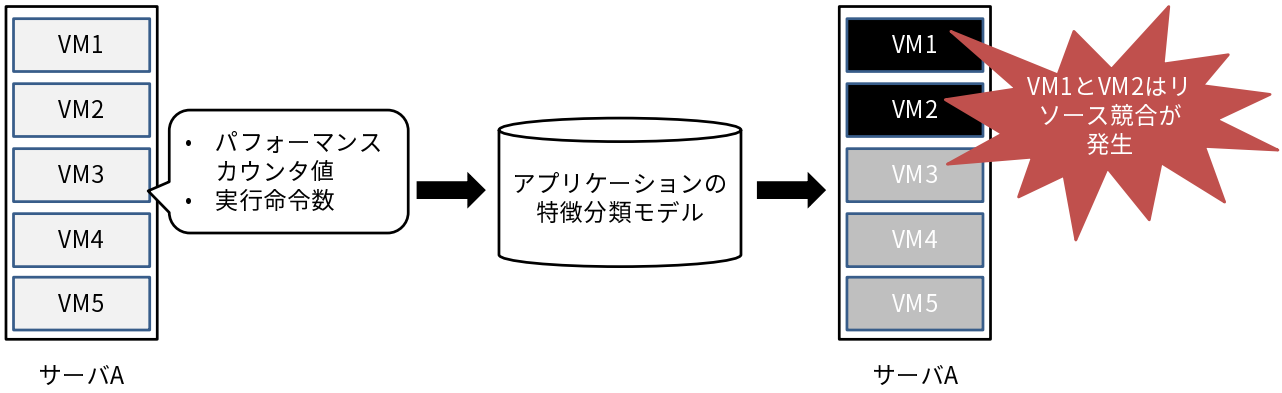

図 7: リソース競合が発生しているサーバの判定の流れ

(ii) 各 VM の特徴を分類:
(i) で作成したモデルを用いて、各物理サーバ上の各 VM で動作しているアプリケーションの特徴を分類する。

(iii) 各物理サーバでの VM 間の相性を判別:
同じ物理サーバ上で、リソース競合が発生する特徴のアプリケーションを実行する VM が複数台存在する場合、それらの VM をリソース競合が発生する VM とする。

アプリケーションの特徴分類モデル作成方法

ある物理サーバの VM 上で動作しているアプリケーションが、同じ物理サーバで動作している他の VM 上のアプリケーションとの間でリソース競合が発生しているかを調べるために、アプリケーションの分類を行う。アプリケーションの分類には、アプリケーションの特徴分類モデルを作成することで分類する。

特徴分類モデル作成の流れは次のとおりである (図 8)。

(a). 各物理サーバ上の VM で各種アプリケーションを実行:
各サーバにおいて 1VM 作成し、その VM 上で各種アプリケーションを実行した際のパフォーマンスカウンタ値と実行命令数を取得する。

(b). 特徴量と目的変数の作成:
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

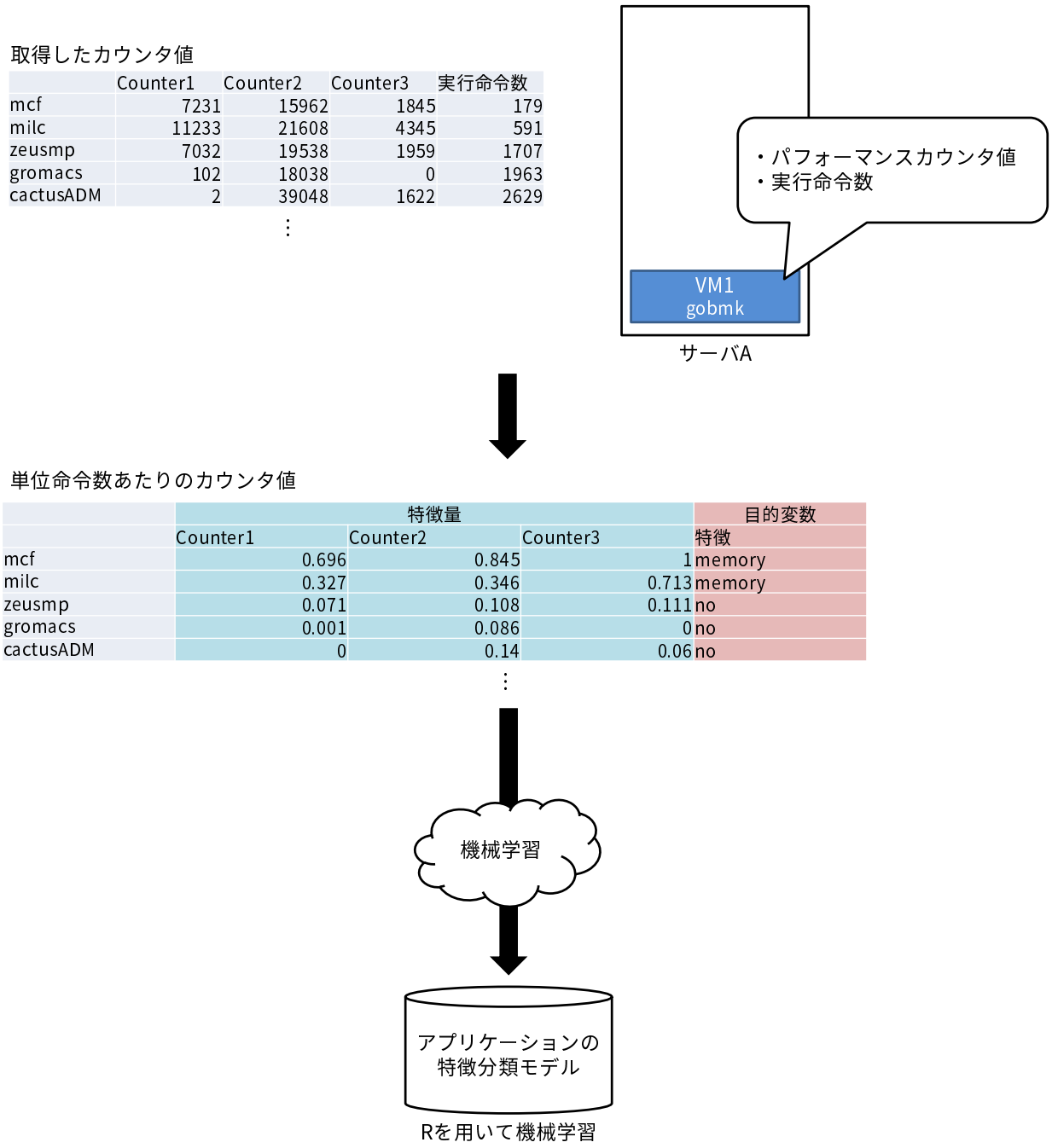

図 8: アプリケーションの特徴分類モデル作成の流れ

ンタ値を計算し、その値の最大値と最小値を用いて 0 から 1 の間の値を取るようにスケーリングした値を特徴量とする。

また、各種アプリケーションの特徴を判断 (詳細は後述) し目的変数とする。

(c). R を用いて機械学習:
分類器には SVM を用いて、(b) で作成した特徴量と目的変数を用いて機械学習により特徴分類モデルを作成する。

分類する特徴

特徴分類モデルは、VM 上で動作しているアプリケーションがリソース競合を発生するかどうかを調べるために作成するので、リソース競合が発生する特徴かどうかを分類できる必要がある。

リソース競合が発生すると考えられるハードウェアリソースは、共有しているリソースである次の 4 つが考えられる。

1. CPU
2. LLC(Last Level Cache)
3. メモリ帯域幅
4. ネットワーク帯域幅

本研究では、物理サーバの CPU コア数以上に VM を作成しない環境を想定しているため、CPU は競合しない。また、今回各 VM 上で動かすアプリケーションは他の VM と通信を行わないアプリケーションとしたので、ネットワーク帯域幅も競合しない。

メモリアクセスが多いアプリケーションでは LLC を多く使用すると考えられるので、同じ特徴として扱うことにする。よって今回はリソース競合が発生する特徴としては、メモリバウンドなアプリケーションのみを扱う。そのため、作成するモデルはメモリバウンドなアプリケーションか否かを判断するモデルである。

特徴の判断方法

メモリバウンドなアプリケーションであるかどうかの定義は明確には存在しない。

そこで、あるサーバの VM の最大同時実行台数を N として定義し、N 台の VM で同じアプリケーションを同時に実行した際に、単体の VM でアプリケーションを実行した場合の実行時間と比べて、実行時間が伸びているアプリケーションをあるサーバ構成にたいしてメモリバウンドなアプリケーションとして判断することにする。

最大同時実行台数で判断する理由としては、最もアプリケーションの特徴が表れるためである。

また、最大数同じアプリケーションを実行する VM を 1 つのサーバに同時配置して実行しても、単体の VM で実行した場合と比べて実行時間が伸びないならば、アプリケーションが使用するリソースへのアクセス回数が N 倍になってもリソース競合が発生しないと考えられるため、リソース競合に関係しないアプリケーションであると判断できるからである。

使用パフォーマンスカウンタの選別

最近のプロセッサにはパフォーマンスカウンタが 100 個以上存在するものが存在するため、VM 再配置の際に VM 上で実行するアプリケーション 1 つに対して全てのパフォーマンスカウンタの値を取得するとなると、取得に多くの時間がかかってしまう。また機械学習を行う際の特徴量が多い場合には過学習が発生するため、作成したいモデルが得られない可能性がある。そのため特徴量として用いるパフォーマンスカウンタは最適な配置ができる範囲で、可能な限り少なくする必要がある。

作りたいモデルは、アプリケーションの特徴分類モデルであり、分類したい特徴毎にパフォーマンスカウンタを選ぶ必要がある。今回は分類する特徴はメモリバウンドなアプリケーションであるか、そうでないかなので、キャッシュとメモリに関するカウンタを見れば良いと考えられる。

そこで用いるパフォーマンスカウンタは学習時に次のように選別する。

(1) 各物理サーバ上の VM でアプリケーションを実行:
各サーバにおいて 1VM 作成し、その VM 上で各種アプリケーションを実行した際のパフォーマンスカウンタ値を取得する。

(2) パフォーマンスカウンタ間の相関係数を取得し選別:
各種アプリケーションを VM 上で実行した際の全パフォーマンスカウンタを

取得し、値が出力されなかったカウンタと、値が極端に少ないカウンタを排除する。

あるカウンタ A とあるカウンタ B で強い相関があることがわかれば、カウンタ A の値が変動するとカウンタ B の値も同じような変動を示すため、両方のカウンタを特徴量として用いても意味がない。そこで、強い相関を持つカウンタの片方を除外し、選別した後のパフォーマンスカウンタどうしで新たに相関があったり 1 回の除外では十分にパフォーマンスカウンタの値を減らせない場合には数回この選別をおこなう。

特徴量のスケーリング

特徴量は、より詳細には次のように定義する。

$c_{ij}(i = 1, ...p, j = 1, ...q)$ をアプリケーション i, パフォーマンスカウンタ j のカウンタ値とし、$t_{ik}(i = 1, ..., p, k = 1, ..., n)$ をアプリケーション i を実行する VM を物理サーバ k で実行した際の実行時間、I_{ik} アプリケーション i を実行する VM をサーバ k 上で実行した際の実行命令数とする。取得した各カウンタ c_{ij} を実行命令数 I_{ik} で割り、単位実行命令数あたりのカウンタ数を特徴分類モデルへの入力、アプリケーションの特徴を出力とする。

$$c_{ij}^{'} = \frac{c_{ij}}{I_{ik}} \tag{1}$$

学習の精度を良くするために、0 から 1 の間の値を取るように特徴量をスケーリングするのが望ましいので、次の手順でスケーリングする。

あるアプリケーション 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 $c_{ib}^{'}$ の最大値である、$max(c_{ib}^{'} : i = 1, ..., q)$ と最小値 $min(c_{ib}^{'} : i = 1, ..., q)$ を用いてスケーリングする。

最終的に

$$x_{ab} = \frac{c_{ab}^{'} - min(c_{ib}^{'} : i = 1, ..., q)}{max(c_{ib}^{'} : i = 1, ...q) - min(c_{ib}^{'} : i = 1, ..., q)} \tag{2}$$

となる。機械学習ではスケーリングした値 x を特徴量の教師値として用いる。

3.3.2　アプリケーションとサーバ構成との相性判断方法

ヘテロジニアスなサーバ環境において、VM 上で動作させるアプリケーションの特徴によってその処理性能を最大にする配置サーバが変わり得るため、処理性能を最大にするサーバを予測し、マイグレーション先サーバを選択するために用いる相性である。

相性の判断にはアプリケーションとサーバ構成の相性モデルを作成し、あるアプリケーションを実行する VM をあるサーバで実行した場合の予測 MIPS 値を用いて行う。現在実行中のサーバでの予測 MIPS 値と他のサーバで実行した場合の予測 MIPS を比較して、現在のサーバでの予測 MIPS 値より高い MIPS 値が予測されるサーバを相性が良いとし、現在のサーバでの予測 MIPS 値未満の MIPS 値が予測されるサーバを相性が悪いとする。

アプリケーションとサーバ構成の相性モデルの作成方法を次に示す。

(a). 各物理サーバ上の VM で各種アプリケーションを実行:
各サーバにおいてひとつの VM 作成し、その VM 上で各種アプリケーションを実行した際のパフォーマンスカウンタ値、実行命令数、実行時間を取得する。

(b). 特徴量と目的変数の作成:
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計算し、その値の最大値と最小値を用いて 0 から 1 の間の値を取るようにスケーリングした値を特徴量とする。

また、取得した実行命令数、実行時間から性能として MIPS 値を計算する。計算した MIPS 値の最大値と最小値を用いてスケーリングを行い、各物理サーバ上の VM 上で各種アプリケーションを実行する際の MIPS 値の相対性能を計算し、その値を目的変数とする。

(c). R を用いて機械学習:
作成した特徴量と目的変数を用いて機械学習により、アプリケーションとサーバの相性モデルを作成する。分類器にはニューラルネットワークを用いる。

使用パフォーマンスカウンタの選別

3.3.1 節のパフォーマンスカウンタの選別と同様の理由から、パフォーマンスカウ

ンタの選別を行う。全てのパフォーマンスカウンタの値を取得し、相関係数による選別を行い、使用するカウンタの選別を行う。

(1) 全てのパフォーマンスカウンタ値を取得し選別:
各種アプリケーションを 1VM 上で実行した際の全パフォーマンスカウンタを取得し、値が出力されなかったものや値が極端に少ないものを排除する。

(2) カウンタ同士を相関係数によって選別:
(1) の分類を行っても十分にパフォーマンスカウンタの値を減らせない場合には、パフォーマンスカウンタ同士の相関係数を求める。選別した後のパフォーマンスカウンタどうしで新たに相関があったり 1 回の除外では十分にカウンタの数を減らせない場合には数回この選別をおこなう。

特徴量・目的変数のスケーリング

3.3.1 節の特徴量のスケーリングと同様の理由から特徴量・目的変数のスケーリングを行う。

取得した各カウンタ c_{ij} を実行命令数 I_{ik} で割り、単位実行命令数あたりのカウンタ数を入力、MIPS 値を出力とする。

$$c_{ij}^{'} = \frac{c_{ij}}{I_{ik}} \tag{3}$$

$$t_{ik}^{'} = \frac{I_{ik}}{t_{ik}} \tag{4}$$

あるアプリケーション 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 $c_{ib}^{'}$ の最大値である、$max(c_{ib}^{'} : i = 1, ..., q)$ と最小値 $min(c_{ib}^{'} : i = 1, ..., q)$ 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の値は a の相対性能 $t_{ac}^{'}$ については、全 MIPS 値 $t_{ik}^{'}(i = 1, ..., q, k = 1, ..., n)$ の最大値 $max(t_{ik}^{'} : i = 1, ..., q, k = 1, ..., n)$ と最小値 $min(t_{ik}^{'} : i = 1, ..., q, k = 1, ...n)$ を用いてスケーリングを行う。

最終的に

$$x_{ab} = \frac{c_{ab}^{'} - min(c_{ib}^{'} : i = 1, ..., q)}{max(c_{ib}^{'} : i = 1, ...q) - min(c_{ib}^{'} : i = 1, ..., q)} \tag{5}$$

$$y_{ac} = \frac{t_{ac}^{'} - min(t_{ik}^{'} : i = 1, ..., q, k = 1, ..., n)}{max(t_{ik}^{'} : i = 1, ...q, k = 1, ..., n) - min(t_{ik}^{'} : i = 1, ..., q, k = 1, ..., n)} \tag{6}$$

となる。機械学習ではスケーリングした値 x と y を特徴量と目的変数の教師値として用いる。

3.4 提案手法運用時の手順

提案手法の概要を示した図 6 の運用時の各 Step について説明する。

(Step1) リソース競合の原因となる VM が偏っているかの判定:

各サーバにおいて、一定時間ごとに管理サーバが各 VM でのパフォーマンスカウンタの値取得し、取得した値と学習データを 3.3.1 節の特徴量のスケーリングにしたがってスケーリングを行う。スケーリングした値を用いて VM 上で動作しているアプリケーションの特徴分類モデルを作成し、各 VM 上で動作しているアプリケーションの種類を判別する。

各サーバ毎にリソース競合が発生している VM の数を調べ、(サーバ上でリソース競合が発生している VM 数の最大値 – サーバ上でリソース競合が発生している VM 数の最小値) > 1 の場合にリソース競合が発生する VM が偏ってると判断し、次のステップに進む。これはマイグレーションが原因で、マイグレーション先サーバでマイグレーション元サーバよりも大きなリソース競合が発生し、データセンタ全体のスループットが低下する可能性があるので、マイグレーションによってリソース競合が発生する VM の数が逆転しないようにすることで、データセンタ全体のスループットが低下する可能性を避けるためである。

リソース競合が発生する VM が偏っていない、もしくは存在しない場合には、一定時間スリープし、(Step1) を行う。

(Step2) マイグレーション元サーバとマイグレーション先サーバ候補の決定:

サーバ上でリソース競合が発生している VM の動作数が最も多いサーバをマイグレーション元サーバとし、リソース競合が発生している VM の動作数が最も少ないサーバ全てをマイグレーション先サーバ候補とする。

(Step3)VM 上で動作するアプリケーションとサーバの相性を判別:

マイグレーション元サーバにおいて各 VM でのパフォーマンスカウンタの値を取

得し、取得した値と学習データを 3.3.2 節の特徴量・目的変数のスケーリングにしたがってスケーリングを行い、スケーリングした値を用いてアプリケーションとサーバの相性モデルを作成し、各サーバ構成での予測 MIPS 値を取得する。

(Step4) マイグレーション VM の決定:

各マイグレーション先サーバ候補毎に、マイグレーション元サーバにおいてリソース競合が発生する VM の中で現在実行中のサーバで予測される MIPS 値よりも、マイグレーション先サーバ候補で予測されすることで最も高い MIPS 値が予測される VM をマイグレーション VM とする。これにより、マイグレーション先サーバ候補とマイグレーション VM の組み合わせを決定する。

(Step5) マイグレーション先サーバの決定:

(Step4) で判断したマイグレーション先サーバとマイグレーション VM の組み合わせの中で、現在実行中のサーバよりもマイグレーション先サーバで予想される MIPS 値の上昇する割合が最も高いマイグレーション先サーバとマイグレーション VM の組から順番に、候補サーバに配置可能であるかを確認する。配置が可能なら候補 VM と (Step1) で調べたリソース競合が発生する VM の数が変化していないかを確認する。

これは (Step1) から (Step5) までを実行するのにある程度の時間がかかるため、その間に状況が変化し、マイグレーション VM の実行が終了されていないか、マイグレーション元サーバはリソース競合が発生する VM の数が最も多いサーバのままなのか、マイグレーション先サーバはリソース競合が発生する VM の数が最も少ないサーバなのかを調べる必要があるためである。

変化していなかった場合には、そのサーバをマイグレーション先サーバとし、マイグレーションを行い、一定時間のスリープのあと (Step1) に戻る。

変化していた場合と、空いているサーバがない場合には一定時間スリープし、(Step1) に戻る。

4　評価実験

4.1　実験環境

本研究で提案する仮想マシン再配置手法を評価するために、ヘテロジニアスなサーバ構成のサーバが存在する実験環境を構築した。管理サーバが 1 台、VM を配置するサーバが 4 台、NFS サーバが 1 台の環境である。サーバ A とサーバ B、サーバ C とサーバ D はそれぞれ同じサーバ構成であり、(CPU コア数 − 1) 個まで VM が配置可能であるため、同時に 16 台まで VM が実行できる環境である。使用したサーバのサーバ構成を表 3 に示し、作成した環境を図 9 に示す。

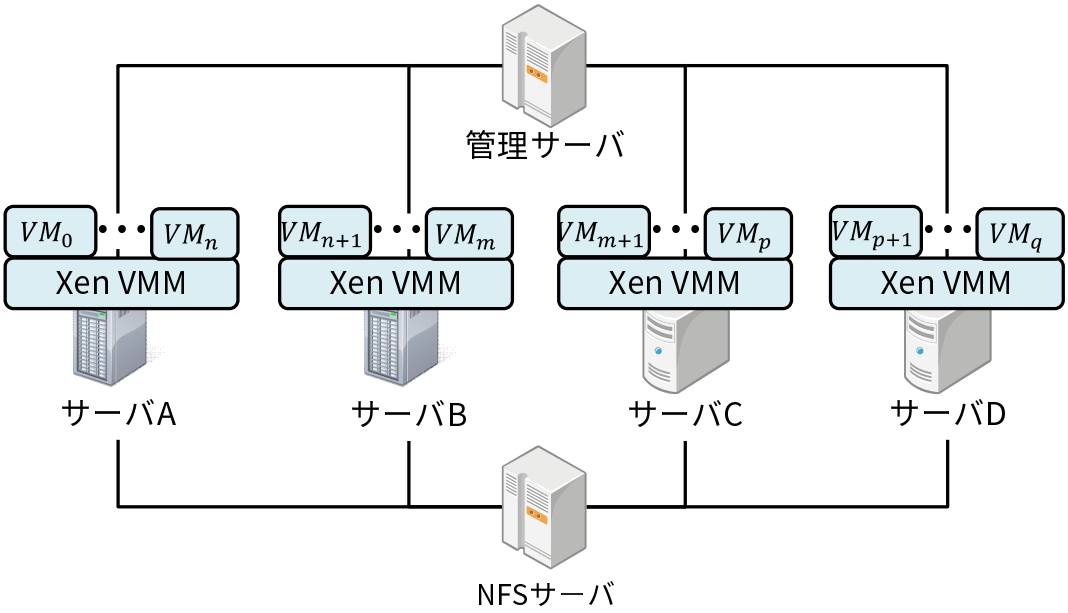

図 9: 作成した環境

各サーバでは CentOS 5.10 が Linux 2.6.18-371.12.1-Xen カーネルで動作しており、Xen のバージョンは xen-4.1.4-2 を使用した。

VM インスタンスの数は表 4 の 1 つのみとした。これは Amazon EC2[1] のスモールインスタンスを想定している。

全ての VM イメージは、NFS サーバに存在する。

表 3: 実験に使用したサーバのサーバ構成

サーバ名	搭載 CPU	コア数	CPU 周波数 (GHz)	L3 Cache (MB)	メモリ (GB)	メモリ周波数 (MHz)
サーバ A,B	Xeon X5675	6	3.06(3.46)	12	12	1333
サーバ C,D	Xeon E5640	4	2.66(2.93)	12	12	1066

表 4: 使用した VM インスタンス

VCPU 数	メモリ (GB)	OS	kernel	仮想化方式
1	2	CentOS 5.10	Linux 2.6.18-371.12.1-Xen	準仮想化

学習及び評価用プログラム

運用前のモデル作成の際と、評価に用いるアプリケーションとして SPEC CPU 2006[5] の整数・浮動小数点ベンチマークアプリケーション 28 個を用いた。用いたベンチマークアプリケーションとその概要を表 5 に示す。評価実験では 28 個のベンチマークを学習用と評価用に分けて使用する。

4.2　運用前のモデルの作成と評価

3 章で説明した運用前に作成する、アプリケーションの特徴分類モデルとアプリケーションとサーバ構成との相性モデルを作成する。パフォーマンスカウンタの取得には、Oprofile[3] にパッチ [7] を当てたものを用いて取得を行った。

4.2.1　アプリケーションの特徴分類モデルの作成

カウンタの選別

3.3.1 節にしたがってパフォーマンスカウンタの選別を行う。本実験環境に用いるサーバの CPU は Intel 社の Xeon X5675 と Xeon E5640 であり、両 CPU ともパ

表 5: SPEC CPU 2006 の使用ベンチマークアプリケーション

プログラム名	説明
perlbench	Programming Language
bzip2	Compression
gcc	C Compiler
gamess	Quantum Chemistry.
mcf	Combinatorial Optimization
milc	Physics / Quantum Chromodynamics
zeusmp	Physics / CFD
gromacs	Biochemistry / Molecular Dynamics
cactusADM	Physics / General Relativity
leslie3d	Fluid Dynamics
namd	Biology / Molecular Dynamics
gobmk	Artificial Intelligence: Go
dealII	Finite Element Analysis
soplex	Linear Programming, Optimization
povray	Image Ray-tracing
calculix	Structural Mechanics
hmmer	Search Gene Sequence
sjeng	Artificial Intelligence: chess
GemsFDTD	Computational Electromagnetics
libquantum	Physics / Quantum Computing
h264ref	Video Compression
tonto	Quantum Chemistry
lbm	Fluid Dynamics
omnetpp	Discrete Event Simulation
astar	Path-finding Algorithms
wrf	Weather
sphinx3	Speech recognition
xalancbmk	XML Processing

フォーマンスカウンタは 277 個存在する。ここでは 277 個のパフォーマンスカウンタを数個まで選別した。

まず、3.3.1 節で述べたように取得できないカウンタと、値が小さいカウンタの除外を行うことでパフォーマンスカウンタを 194 個まで選別した。

次に選別したカウンタの中からキャッシュとメモリに関係のあるカウンタのみを選別する。

最後に相関係数による選別を行う。取得した全カウンタどうしの相関係数とその平均値を計算し、平均相関係数絶対値 0.5 以上のものをカウンタ同士に強い相関があると判断し除外した。これを数回繰り返し、3 つのカウンタに選別した。

モデルへの入力は単位命令数値のカウンタ値なので、3 つのカウンタに加えて実行命令数も取得する。最終的に取得するカウンタを表 6 に示す。L2 control unit である Super Queue に関するカウンタがストールしたサイクル数、L2 からラインが追い出された回数、LLC でミスが起こりメモリアクセスされた回数を取得することで、LLC、メモリアクセスに関するカウンタを選択した。

表 6: 選んだカウンタ

カウンタ名	ユニットマスク	説明
SQ_FULL_STALL_CYCLES		Super Queue full stall cycles
L2_LINES_OUT	0x02	L2 modified lines evicted by a demand request
MEM_LOAD_RETIRED	0x10	Retired loads that miss the LLC cache
INST_RETIRED	0x01	Instructions retired

選んだカウンタを用いて特徴分類モデルを作成する。

アプリケーションの特徴の分類

3.3.1 節にしたがって 28 個のベンチマークの特徴を分類する。サーバ A,B においては配置可能な VM 数は 5 個なので、あるアプリケーションを VM5 台で同時実行させた際の実行時間が、単体の VM で実行した場合の実行時間と比べて伸びた場合にはそのアプリケーションをサーバ A,B においてメモリバウンドなアプリケーションであるとする。同様にサーバ C,D では配置可能な VM 数は 3 個なので、あるアプリケーションを VM3 台で同時実行させた際の実行時間が、単体の VM で実行した

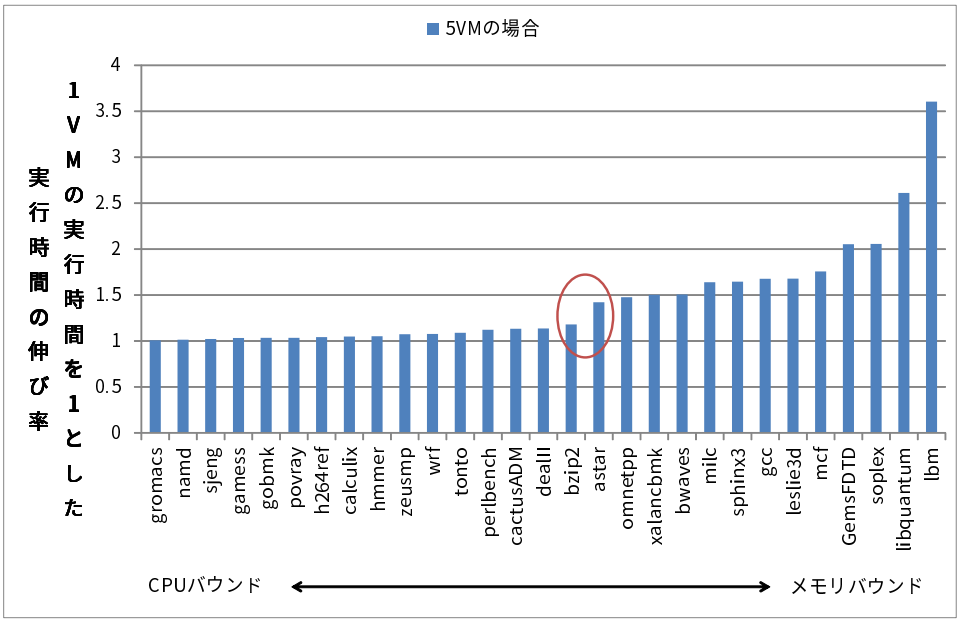

図 10: サーバ A,B のサーバ構成に配置可能な最大数の VM を配置した場合の実行時間

場合の実行時間と比べて伸びた場合にはそのアプリケーションをサーバ C,D においてメモリバウンドなアプリケーションであるとする。

サーバ A,B において 5VM 全てで同じアプリケーションを実行させた実行時間が、ひとつの VM 上でアプリケーションを実行させた場合の実行時間と比べてどれだけ伸びるかのグラフを図 10 に示し、サーバ C,D において 3VM 全てで同じアプリケーションを実行させた実行時間が、ひとつの VM 上でアプリケーションを実行させた場合の実行時間と比べてどれだけ伸びるかのグラフを図 11 に示す。

横軸は実行したアプリケーション名を示し、縦軸はひとつの VM のみを実行した際の実行時間を 1 とした実行時間の伸び率を示す。

今回実験に使用した SPEC2006 のアプリケーションは、CPU とメモリの性能を測定するアプリケーションであるため、複数の VM で同じアプリケーションを同時実行した際に実行時間が伸びる原因としては、リソース競合が発生するメモリ帯域幅か LLC の競合によって伸びることが考えられる。

実際に、複数の VM で同じアプリケーションを実行した際のアプリケーションの実行時間の伸びが、リソース競合によるものなのかを調べる必要があるため、LLC とメ

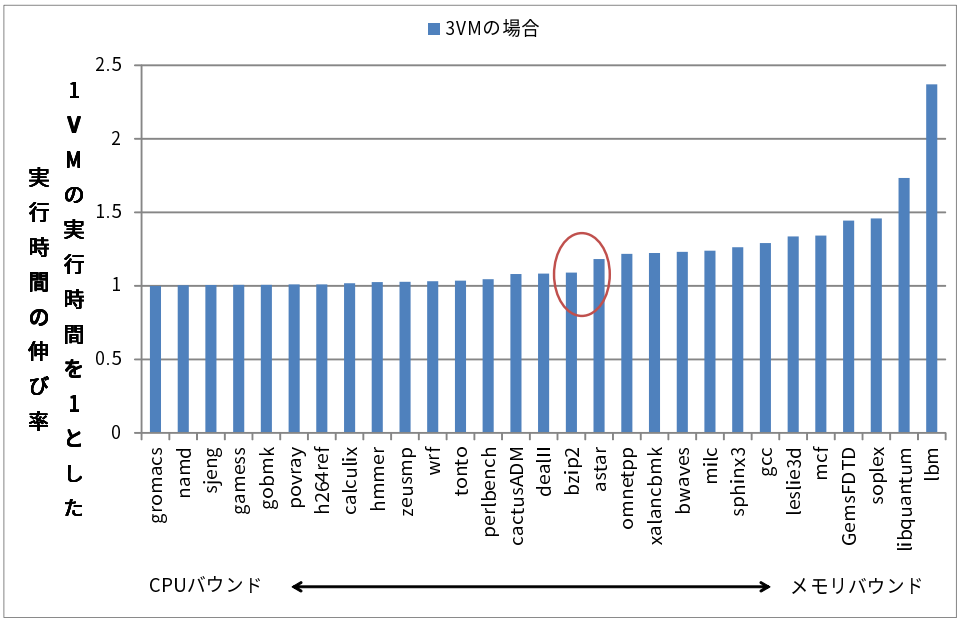

図 11: サーバ C,D のサーバ構成に配置可能な最大数の VM を配置した場合の実行時間

モリに関するパフォーマンスカウンタを取得し、実行時間との伸び率と相関係数を計算する。メモリバウンドなアプリケーションはLLCアクセスが多いことから、LLCアクセスに関するカウンタのみを取得する。LLCはL2キャッシュミスが発生した際にアクセスされるため、L2キャッシュが置き換えられた回数であるL2_LINES_OUT:0x02を取得すると、実行時間の伸び率との相関係数はサーバ構成A,Bとサーバ構成C,Dの両方の平均で0.82であり、一般的に相関があると言われる0.7よりも高いことから、実行時間の伸び率はリソース競合と関係があると言える。そのため図10,11のどちらの場合にも、横軸の右側ほどメモリバウンドなアプリケーションであると言える。

どちらの結果に置いても、図中の丸で囲った部分から実行時間が伸びていることから、その部分から右側にあるアプリケーションをメモリバウンドなアプリケーションとして判断する。

最終的に、各アプリケーションの分類は表7のようになった。今回は全ての物理サーバにおいて各アプリケーションで違う特徴に分類されたものはなく、同じ特徴として分類された。

表 7: SPEC CPU 2006 の特徴分類結果

プログラム名	特徴	プログラム名	特徴
perlbench	no	povray	no
bzip2	no	calculix	no
gcc	memory	hmmer	no
gamess	no	sjeng	no
mcf	memory	GemsFDTD	memory
milc	memory	libquantum	memory
zeusmp	no	h264ref	no
gromacs	no	tonto	no
cactusADM	no	lbm	memory
leslie3d	memory	omnetpp	memory
namd	no	astar	memory
gobmk	no	wrf	no
dealII	no	sphinx3	memory
soplex	memory	xalancbmk	memory

作成したモデルの評価

28 個のベンチマークのうち 21 個のアプリケーションを学修データとして用いてモデルを作成し、7 個のアプリケーションをテストデータとしてモデルの評価を行う。

作りたいモデルが作成できているかを確認するために、4-fold cross-validation を行った結果を表 8 に示す。全体の精度は約 91%であり、2 クラス分類をランダムに分類した場合の精度である 50%より大きく、分類を行える精度であると考えられる。

4.2.2　アプリケーションとサーバ構成との相性

カウンタの選別

3.3.2 節にしたがってパフォーマンスカウンタの選別を行う。まず、取得できないカウンタと、値が小さいカウンタの除外を行うことでパフォーマンスカウンタを 130 個まで選別した。

表 8: サーバ A,B の構成でのアプリケーションの特徴の予測結果

	アプリケーション名	サーバ構成 A,B での予測特徴	サーバ構成 C,D での予測特徴	実際の特徴
ケース 1	soplex	memory	memory	memory
	libquantum	memory	memory	memory
	mcf	memory	memory	memory
	povray	no	no	no
	gobmk	no	no	no
	hmmer	no	no	no
	namd	no	no	no
ケース 2	lbm	memory	memory	memory
	milc	memory	memory	memory
	Xalan	no	no	memory
	cactusADM	no	no	no
	perlbench	no	no	no
	calculix	no	no	no
	h264ref	no	no	no
ケース 3	GemsFDTD	memory	memory	memory
	sphinx	no	no	memory
	omnetpp	memory	memory	memory
	dealII	no	no	no
	tonto	no	no	no
	gamess	no	no	no
	sjeng	no	no	no
ケース 4	leslie3d	memory	memory	memory
	gcc	memory	memory	memory
	astar	memory	memory	memory
	bzip2	no	memory	no
	wrf	no	no	no
	.zeusmp	no	no	no
	gromacs	no	no	no

次に相関係数による選別を行う。図 12 に示すように、取得した全てのカウンタ同士の相関係数を算出しその平均値を計算し、平均相関係数絶対値 0.4 以上のものをカウンタ同士に強い相関があると判断し除外した。これを数回繰り返し、6 つのカウンタに選別した。

最終的に選んだカウンタを、表 9 に示す。マイクロオペレーションに関連するカウンタや、フェッチ、デコードに関するカウンタが選択されていることから、命令セットに関わるカウンタが MIPS 値と相関があるといえる。

	相関係数平均	CPU_CLK_UNHALTED 0x00 20000000	NHALTED C	RED 0x00	SSES 0x41	EFS 0x4f 2	ETIRED 0x	D_RETIRE	LOCK 0x02
CPU_CLK_UNHALTED 0x00 20000000	0.493995535	1	0.990404	0.811214	0.272933	0.277546	0.446721	0.173417	0.27291
CPU_CLK_UNHALTED 0x00 100000	0.505161759	0.990403954	1	0.762866	0.361441	0.345268	0.401908	0.126903	0.299433
INST_RETIRED 0x00 20000000	0.413208785	0.811213785	0.762866	1	−0.07405	−0.02388	0.734941	0.246633	0.106177
LLC_MISSES 0x41 50000	0.293171402	0.272932582	0.361441	−0.07405	1	0.710446	−0.20991	−0.18743	0.167845
LLC_REFS 0x4f 2000000	0.269970289	0.27754583	0.345268	−0.02388	0.710446	1	−0.08121	−0.10363	0.064701
BR_INST_RETIRED 0x00 500000	0.348491396	0.446720803	0.401908	0.734941	−0.20991	−0.08121	1	0.612786	−0.13011
BR_MISS_PRED_RETIRED 0x00 50000	0.253217248	0.173416632	0.126903	0.246633	−0.18743	−0.10363	0.612786	1	−0.19062
LOAD_BLOCK 0x02 200000	0.190785381	0.272909601	0.299433	0.106177	0.167845	0.064701	−0.13011	−0.19062	1
SB_DRAIN 0x07 200000	0.392694358	0.477845839	0.539486	0.168065	0.418667	0.409342	−0.18358	−0.24292	0.199915
STORE_BLOCKS 0x08 200000	0.460950338	0.76295486	0.794496	0.606865	0.283102	0.3155	0.35364	0.006857	0.159472
PARTIAL_ADDRESS_ALIAS 0x01 200000	0.135727482	0.13448569	0.149176	0.029958	0.213456	0.069744	−0.13377	−0.12176	0.810848
DTLB_LOAD_MISSES 0x01 200000	0.303552907	0.499572604	0.5399	0.03454	0.421716	0.514609	−0.26299	−0.14491	0.295158
DTLB_LOAD_MISSES 0x04 200000	0.300141757	0.482683246	0.527891	0.007308	0.484085	0.573583	−0.26313	−0.1315	0.283539
DTLB_LOAD_MISSES 0x10 200000	0.193912062	0.27986014	0.292488	−0.03302	0.202333	0.445869	−0.01649	−0.05759	0.15766
MEM_INST_RETIRED 0x01 2000000	0.444672299	0.913090959	0.885123	0.927232	0.05218	0.054367	0.55852	0.176022	0.242583
MEM_INST_RETIRED 0x02 2000000	0.36082651	0.782684905	0.756256	0.54517	0.084238	−0.03013	0.227878	0.127146	0.426895
UOPS_ISSUED 0x01 2000000	0.444917924	0.886953813	0.848026	0.97304	−0.01377	0.025034	0.667436	0.274732	0.226064
UOPS_ISSUED 0x02 2000000	0.285709389	0.644863809	0.632928	0.357741	0.061932	0.085363	0.153643	0.054233	0.131472
LOAD_DISPATCH 0x01 2000000	0.408060807	0.781128929	0.722897	0.964071	−0.13481	−0.06046	0.783554	0.411827	0.06014
LOAD_DISPATCH 0x02 2000000	0.351764741	0.662287345	0.679275	0.46681	0.272178	0.151338	0.047655	−0.138	0.615401
LOAD_DISPATCH 0x04 2000000	0.362503558	0.690988873	0.726669	0.294558	0.419969	0.416832	−0.02497	−0.11122	0.298103
LOAD_DISPATCH 0x07 2000000	0.458499859	0.939102817	0.913208	0.912191	0.101182	0.146469	0.586014	0.219493	0.260189
INST_QUEUE_WRITES 0x01 2000000	0.395427847	0.775749637	0.710442	0.906889	−0.12614	−0.00127	0.775883	0.542958	0.08606
INST_DECODED 0x01 2000000	0.199450031	0.211539431	0.185969	0.292114	−0.24635	0.168616	0.320438	0.346478	0.079712
TWO_UOP_INSTS_DECODED 0x01 2000000	0.173862117	0.207296721	0.190319	0.274742	−0.19474	0.207549	0.198793	0.199026	0.108429
INST_QUEUE_WRITE_CYCLES 0x01 2000000	0.369978322	0.763332006	0.69268	0.902078	−0.14241	−0.0777	0.73311	0.48756	0.057293
L2_RQSTS 0x01 200000	0.457264256	0.790451492	0.823948	0.543748	0.331399	0.384059	0.191444	−0.02624	0.303353
L2_RQSTS 0x02 200000	0.249942315	0.261743429	0.325032	−0.03847	0.744075	0.972365	−0.08144	−0.08087	0.093407
L2_RQSTS 0x03 200000	0.451961008	0.78661082	0.823528	0.511168	0.479982	0.590011	0.193125	−0.01751	0.283751
L2_RQSTS 0x04 200000	0.253468262	0.18641407	0.208937	0.138373	0.041278	−0.00825	−0.05497	−0.22206	0.212134
L2_RQSTS 0x0c 200000	0.277721546	0.256056458	0.286265	0.13498	0.093204	0.07629	−0.11225	−0.26497	0.212292
L2_RQSTS 0x40 200000	0.214853973	0.08978787	0.103369	0.144084	0.028277	0.099963	−0.05284	−0.12183	−0.09702
L2_RQSTS 0x80 200000	0.383764711	0.396312449	0.458226	0.431957	0.3705	0.344385	0.336159	0.02939	0.034673
L2_RQSTS 0xaa 200000	0.396113997	0.446092458	0.525313	0.313308	0.629551	0.715585	0.192298	−0.03386	0.06884
L2_RQSTS 0xc0 200000	0.384252867	0.385293401	0.439508	0.435677	0.321884	0.314596	0.294187	−0.00432	0.01146
L2_RQSTS 0xff 200000	0.479181547	0.663565952	0.722648	0.474675	0.47718	0.563408	0.181966	−0.08462	0.197233
L2_DATA_RQSTS 0x01 200000	0.264306421	0.321724948	0.386947	0.002533	0.812154	0.933253	−0.06308	−0.10559	0.088635
L2_DATA_RQSTS 0x04 200000	0.461227281	0.836449443	0.862737	0.609491	0.28401	0.264701	0.218747	−0.03496	0.30793
L2_DATA_RQSTS 0x08 200000	0.20848942	0.336760879	0.274194	0.38093	−0.26492	−0.13831	0.2879	−0.05477	0.143059
L2_DATA_RQSTS 0x0f 200000	0.439034467	0.828033192	0.850026	0.515047	0.398138	0.551032	0.130204	−0.07062	0.31212
L2_DATA_RQSTS 0x10 200000	0.394840596	0.425419264	0.478061	0.496661	0.32321	0.35868	0.361758	−0.00737	0.02099
L2_DATA_RQSTS 0x10 200000	0.394840596	0.425419264	0.478061	0.496661	0.32321	0.35868	0.361758	−0.00737	0.02099
L2_DATA_RQSTS 0xf0 200000	0.407284727	0.492724535	0.533844	0.578372	0.302963	0.262341	0.423716	−0.04152	0.089142

図 12: パフォーマンスカウンタどうしの相関係数

作成したモデルの評価

学習データとは別に用意したテストデータを用いて評価する。テストデータを入力として与えた際に出力された MIPS 値を、実際に取得した正解の MIPS 値との間の相関係数を取得することによって評価した。

作りたいモデルが作成できているかを確認するために 4-fold cross-validation を行った結果を、表 10 と表 11 に示す。GemsFDTD アプリケーションでは予測 MIPS

表 9: 選んだカウンタ

Name	ユニットマスク	説明
PARTIAL_ADDRESS_ALIAS	0x01	False dependencies due to partial address aliasing
SSEX_UOPS_RETIRED	0x08	SIMD Packed-Double μ OPs retired
L2_RQSTS	0x40	L2 instruction fetch hits
ILD_STALL	0x02	Stall cycles due to BPU MRU bypass
UOPS_ISSUED	0x02	Fused μ OPs issued
MEM_LOAD_RETIRED	0x40	Retired loads that miss L1D and hit an previously allocated LFB

値と実際の MIPS 値の差が約 0.7 ポイント近くなっているベンチマークも存在していたが、4 回の相関係数の平均値はサーバ A,B とサーバ C,D を合わせて 0.7 であった。相関係数は、一般的に 0.7 以上で強い相関があると言われることから、予測 MIPS 値と実際の MIPS 値には強い相関があると考えられ、相性の予測が可能であるといえる。

4.3　アプリケーションの特徴に基づいた仮想マシン再配置手法の評価

4.3.1　評価方法

物理サーバに予め VM が配置されており、各 VM 上では表 12 に示す学習に用いていない 7 個のアプリケーションのうち 1 つ実行させる。

初期配置から再配置を行わない場合と、提案手法を用いて再配置を行う場合との全 VM が実行終了するまでの時間を比較する。初期配置はマイグレーションができるようにいくつかのサーバでは実行可能な VM の最大数より少なく動作させておくものとする。

VM 再配置の有効性は初期配置のリソース競合が発生する VM の偏りかたに依存するため、いくつかの初期配置において比較を行う必要がある。そこで初期配置が次の場合の評価を行う。

- リソース競合が発生する VM の数が大きく偏っている初期配置の場合（提案手法が最も有効な場合）

表 10: 目的変数予測結果 (サーバ A,B の場合)

	アプリケーション名	サーバ A,B			サーバ C,D		
		実際の値	予測値	エラー	実際の値	予測値	エラー
ケース 1	soplex	0.247	0.302	0.055	0.197	0.249	0.052
	libquantum	0.553	0.764	0.211	0.488	0.626	0.138
	mcf	0.019	0.118	0.099	0.000	0.114	0.114
	povray	0.551	0.310	0.241	0.444	0.255	0.189
	gobmk	0.408	0.433	0.025	0.326	0.343	0.017
	hmmer	0.787	0.509	0.278	0.645	0.404	0.241
	namd	0.637	0.603	0.034	0.518	0.476	0.042
ケース 2	lbm	0.485	0.584	0.099	0.392	0.478	0.086
	milc	0.247	0.269	0.022	0.204	0.224	0.020
	Xalan	0.466	0.176	0.290	0.376	0.151	0.225
	cactusADM	0.247	0.599	0.352	0.191	0.491	0.300
	perlbench	0.646	0.605	0.041	0.524	0.494	0.030
	calculix	1.000	0.806	0.194	0.823	0.695	0.128
	h264ref	0.802	0.639	0.163	0.656	0.519	0.137
ケース 3	GemsFDTD	0.371	1.000	0.629	0.318	0.999	0.681
	sphinx	0.677	0.997	0.320	0.549	0.986	0.437
	omnetpp	0.141	0.018	0.123	0.104	0.025	0.079
	dealII	0.645	1.000	0.355	0.529	0.999	0.470
	tonto	0.608	0.663	0.055	0.494	0.538	0.044
	gamess	0.745	0.931	0.186	0.609	0.838	0.229
	sjeng	0.480	1.000	0.520	0.387	1.000	0.613
ケース 4	leslie3d	0.359	0.034	0.325	0.292	0.043	0.249
	gcc	0.305	0.170	0.135	0.255	0.156	0.099
	astar	0.235	0.354	0.119	0.174	0.291	0.117
	bzip2	0.623	0.729	0.106	0.509	0.593	0.084
	wrf	0.520	0.679	0.159	0.418	0.546	0.128
	.zeusmp	0.366	0.249	0.117	0.295	0.212	0.083
	gromacs	0.419	0.601	0.182	0.334	0.482	0.148

表 11: 目的変数予測結果 (サーバ C,D の場合)

	アプリケーション名	サーバ A,B			サーバ C,D		
		実際の値	予測値	エラー	実際の値	予測値	エラー
ケース 1	soplex	0.247	0.280	0.033	0.197	0.232	0.035
	libquantum	0.553	0.639	0.086	0.488	0.514	0.026
	mcf	0.019	0.129	0.110	0.000	0.123	0.123
	povray	0.551	0.276	0.275	0.444	0.230	0.214
	gobmk	0.408	0.463	0.055	0.326	0.371	0.045
	hmmer	0.787	0.999	0.212	0.645	0.992	0.347
	namd	0.637	0.792	0.155	0.518	0.656	0.138
ケース 2	lbm	0.485	0.368	0.117	0.392	0.293	0.099
	milc	0.247	0.266	0.019	0.204	0.217	0.013
	Xalan	0.466	0.086	0.380	0.376	0.082	0.294
	cactusADM	0.247	0.476	0.229	0.191	0.378	0.187
	perlbench	0.646	0.542	0.104	0.524	0.438	0.086
	calculix	1.000	0.708	0.292	0.823	0.585	0.238
	h264ref	0.802	0.600	0.202	0.656	0.487	0.169
ケース 3	GemsFDTD	0.371	0.002	0.369	0.318	0.004	0.314
	sphinx	0.677	0.960	0.283	0.549	0.893	0.344
	omnetpp	0.141	0.011	0.130	0.104	0.018	0.086
	dealII	0.645	0.927	0.282	0.529	0.816	0.287
	tonto	0.608	0.618	0.010	0.494	0.498	0.004
	gamess	0.745	0.804	0.059	0.609	0.680	0.071
	sjeng	0.480	0.958	0.478	0.387	0.879	0.492
ケース 4	leslie3d	0.359	0.114	0.245	0.292	0.115	0.177
	gcc	0.305	0.108	0.197	0.255	0.110	0.145
	astar	0.235	0.000	0.235	0.174	0.000	0.174
	bzip2	0.623	0.716	0.093	0.509	0.583	0.074
	wrf	0.520	0.768	0.248	0.418	0.634	0.216
	.zeusmp	0.366	0.543	0.177	0.295	0.434	0.139
	gromacs	0.419	0.171	0.248	0.334	0.158	0.176

表 12: 評価に使用したアプリケーション

アプリケーション名	特徴
mcf	memory
soplex	memory
libquantum	memory
namd	no
gobmk	no
povray	no
hmmer	no

- リソース競合が発生する VM の数が偏っている初期配置の場合

また、VM 上で動作するアプリケーションが途中で変化することにより、リソース競合が発生する VM の偏り方が変化する場合においても評価を行う。

4.3.2　VM 上で動作するアプリケーションの特徴が変化しない場合

リソース競合が発生する VM の数が大きく偏っている初期配置の場合（提案手法が最も有効な場合）

サーバ A とサーバ D にメモリバウンドなアプリケーションを実行する VM が偏っている環境を作成し、リソース競合が発生する VM の数が大きく偏っている初期配置の場合の評価を行う。VM の配置方法と実行するアプリケーションを図 13 に示す。黒色の四角がメモリバウンドなアプリケーションを示しており、灰色の四角は CPU バウンドなアプリケーションを示している。VM 上で実行するアプリケーションは表 12 の 7 つのアプリケーションからランダムに選んだ。

リソース競合が発生する VM の数が偏っている初期配置の場合

サーバ A にメモリバウンドなアプリケーションを実行する VM が偏っている環境を作成し、リソース競合が発生する VM の数が偏っている初期配置の場合の評価を行う。VM の配置と実行するアプリケーションを図 14 に示す。黒色の四角がメモリバウンドなアプリケーションを示しており、灰色の四角は CPU バウンドなアプリ

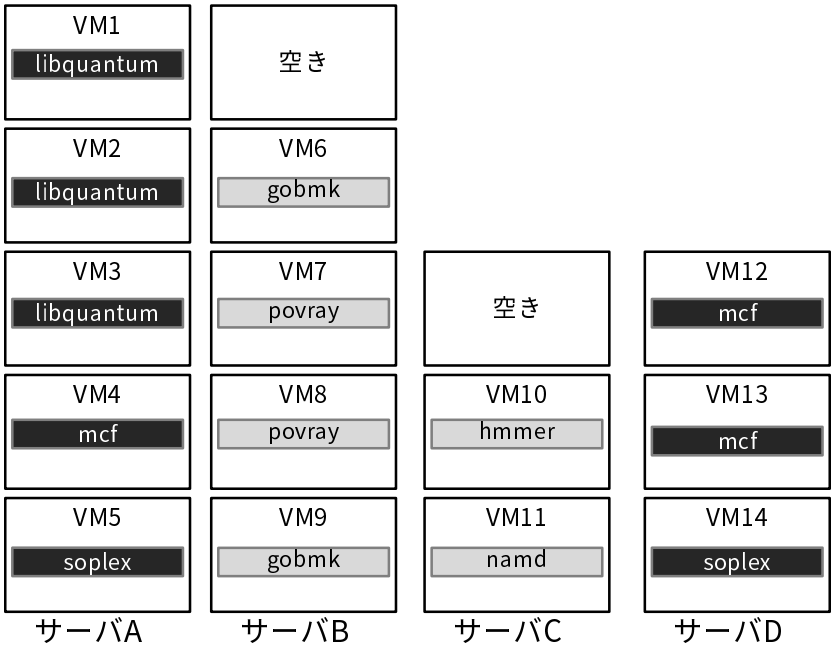

図 13: リソース競合が発生する VM の数が大きく偏っている VM 初期配置

ケーションを示している。VM 上で実行するアプリケーションは表 12 の 7 つのアプリケーションからランダムに選んだ。

4.3.3　VM 上で動作するアプリケーションの特徴が変化しない場合の結果

リソース競合が発生する VM の数が大きく偏っている初期配置の場合（提案手法が最も有効な場合）の結果

リソース競合が発生する VM が特定のサーバに偏っている場合の各 VM の実行時間の結果を図 15 に示す。横軸は各 VM と初期配置のサーバを示し、縦軸は各 VM 上のアプリケーションの実行が終了するまでの時間を示している。再配置なしの場合と比べて提案手法を用いた場合には、最大で約 26%VM 上で実行されるアプリケーションの実行時間を短縮することができている。

提案手法がリソース競合を緩和し、相性の良いサーバにマイグレーションがされているかを見るために各 VM が実行されていたサーバについて図 16 に示す。縦軸は初期配置の各 VM の初期配置のサーバについて示しており、各棒グラフの色は実行

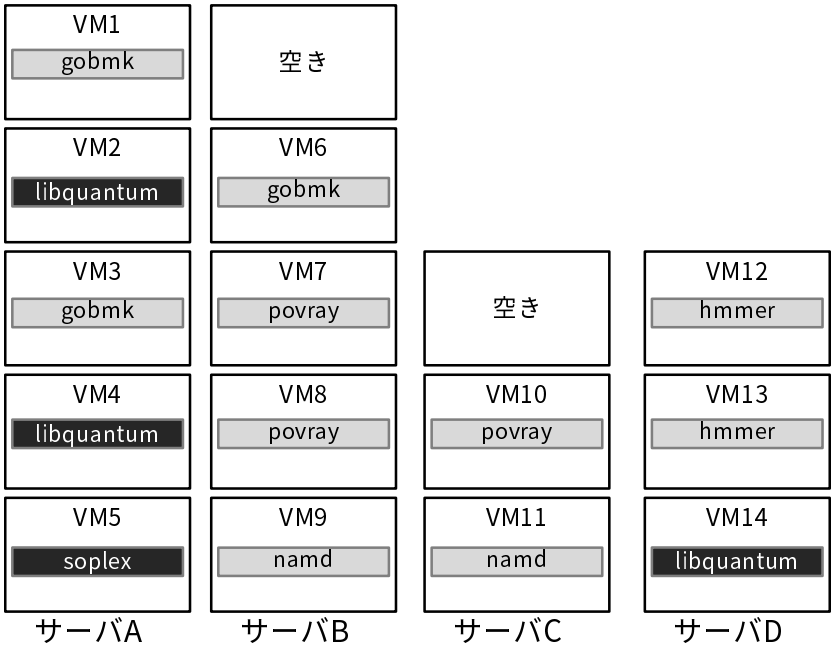

図 14: リソース競合が発生する VM の数が偏っている VM 初期配置

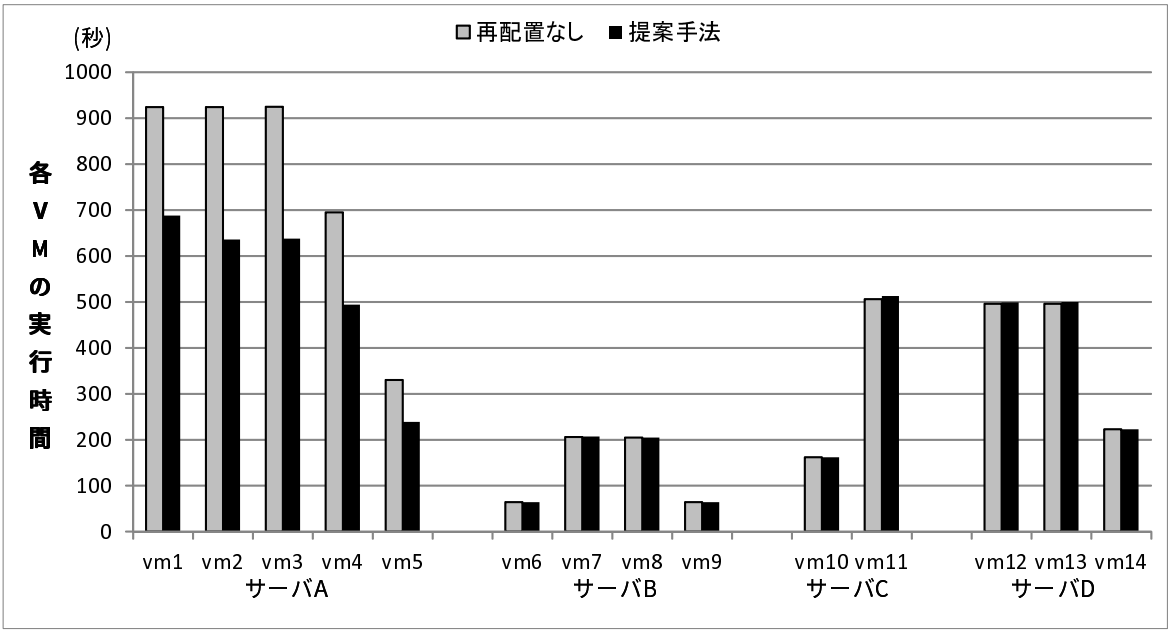

図 15: リソース競合が発生する VM の数が大きく偏っている VM 初期配置の結果

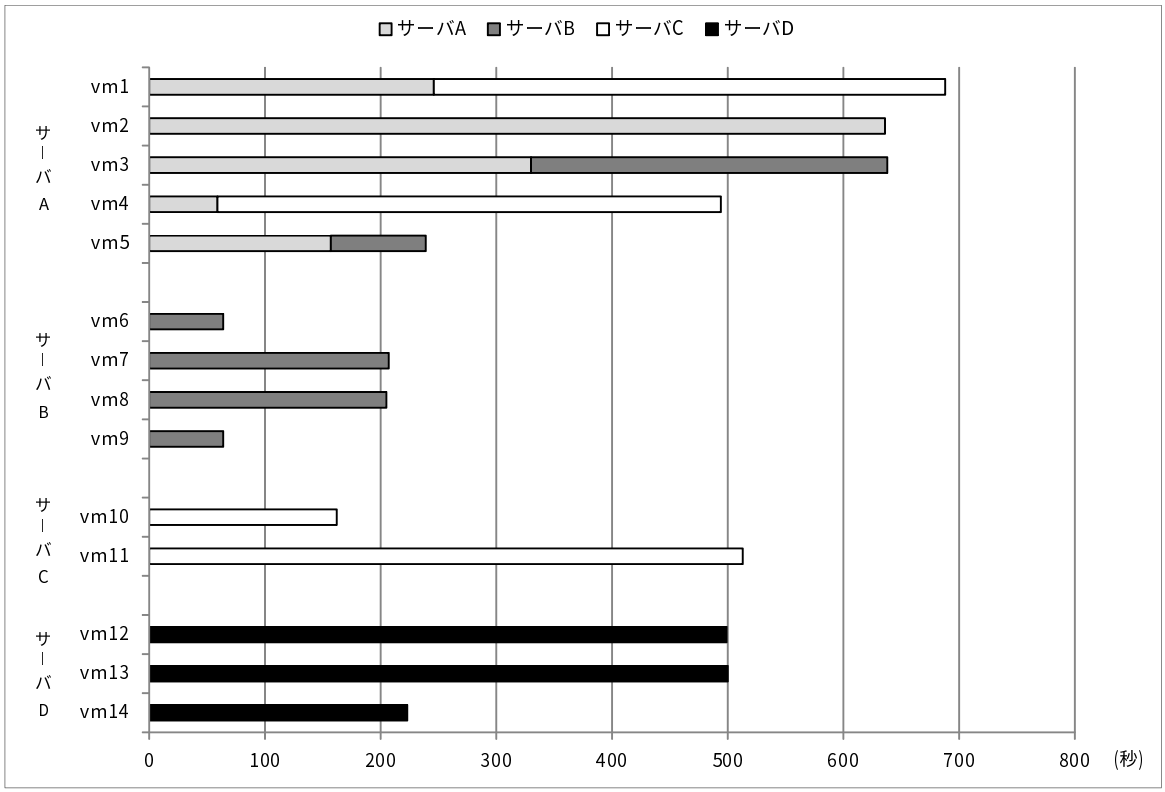

図 16: リソース競合が発生する VM の数が大きく偏っている VM 初期配置のマイグレーションの経過

されていたサーバを示している。

マイグレーションが行われた回数は 4 回だった。

1 回目は開始から 59 秒に、サーバ A の VM4 がサーバ C にマイグレーションされた。マイグレーション元サーバは最もリソース競合が発生する VM の数が多いサーバを選択するため、サーバ A が選択される。マイグレーションする VM は開始から 59 秒のサーバ A において VM4 が選ばれている。図 10 をみると、VM1,2,3 動作している libquantum が最も実行時間の伸び率が大きいため、サーバ A で動作している VM の中で VM1,2,3 のいずれかの VM をマイグレーションするとサーバ A のリソース競合が発生している VM の中で最も MIPS 値が改善されると考えられるが、アプリケーションとサーバ構成との相性モデルによってサーバ A のリソース競合が発生している VM の中でマイグレーションすると最も MIPS 値が伸びると判断されたのは VM4 であり、期待通りではなかった。マイグレーション先サーバ候補には、リ

ソース競合が発生するVMの数が最も少ないサーバBとサーバCが選ばれ、単体で実行した際にはサーバA,Bのほうが全てのアプリケーションでサーバC,Dより実行時間が短いことから、サーバBが選ばれることが期待されるが、アプリケーションとサーバの相性モデルの予測がサーバCのほうが高くなっており予測がうまくいっていなかった。

2回目は開始から157秒にサーバAからVM5がサーバBにマイグレーションされた。マイグレーション元サーバは最もリソース競合が発生するVMの数が多いサーバを選択するため、1回目と同じサーバAが選択される。マイグレーションするVMは開始から157秒のサーバAにおいて、1回目と同様の理由からVM1,2,3のどれかが選ばれることが期待されるが、実際にはVM5が選ばれており期待通りの動作ではなかった。マイグレーション先サーバ候補には、リソース競合が発生するVMの数が最も少ないサーバが選択されるためサーバBが選択されることが期待され、実際にもサーバBが選ばれている。

3回目は開始から246秒にサーバAからVM1がサーバCにマイグレーションされた。マイグレーション元サーバは最もリソース競合が発生するVMの数が多いサーバを選択するため、1,2回目と同じサーバAが選択される。マイグレーションするVMは開始から246秒のサーバAでは全て同じアプリケーションを実行するVMなのでどれを選んでも同じである。マイグレーション先サーバ候補には、リソース競合が発生するVMの数が最も少ないサーバBとサーバCが選ばれ、単体で実行した際にはサーバA,Bのほうが全てのアプリケーションでサーバC,Dより実行時間が短いことから、サーバBが選ばれることが期待されるが、アプリケーションとサーバの相性モデルの予測がサーバCのほうが高くなっておりここでも予測がうまくいかなかった。

4回目は開始から330秒にサーバAからVM3がサーバBへマイグレーションされた。マイグレーション元サーバは最もリソース競合が発生するVMの数が多いサーバを選択するため、サーバAかサーバDが選択されるが、同数の場合には物理サーバ名のアルファベットが若い順番から選ぶためサーバAが選ばれている。マイグレーションするVMは開始から330秒のサーバAでは全て同じアプリケーションを実行するVMなのでどれを選んでも同じである。マイグレーション先サーバ候補には、リソース競合が発生するVMの数が最も少ないサーバBが選ばれている。

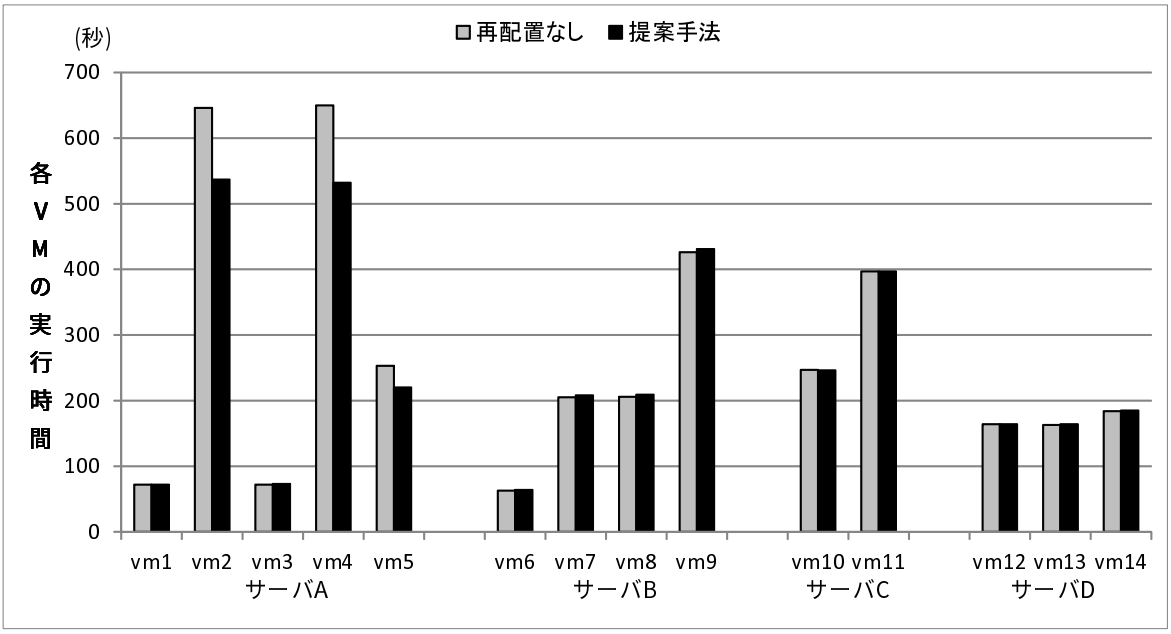

図 17: リソース競合が発生する VM の数が偏っている初期配置の場合の結果

アプリケーションとサーバの相性モデルを用いたマイグレーション先サーバの予測は予測を行った 2 回中全て間違っていたが、VM 上で動作するアプリケーションの特徴分類モデルの予測は正しかったため、各物理サーバにおいてリソース競合を緩和することができ、各 VM 上で実行するアプリケーションの実行時間を短縮することができた。

リソース競合が発生する VM の数が偏っている初期配置の場合の結果

リソース競合が発生する VM が特定のサーバに偏っている場合の各 VM の実行時間の結果を図 15 に示す。横軸は各 VM と初期配置のサーバを示し、縦軸は各 VM 上のアプリケーションの実行が終了するまでの時間を示している。再配置なしの場合と比べて提案手法を用いた場合には、最大で約 18%VM 上で実行されるアプリケーションの実行時間を短縮することができている。

提案手法がリソース競合を緩和し、相性の良いサーバにマイグレーションがされているかを見るために各 VM が実行されていたサーバについて図 18 に示す。縦軸は初期配置の各 VM の初期配置のサーバについて示しており、各棒グラフの色は実行されていたサーバを示す。マイグレーションが行われた回数は 1 回だった。

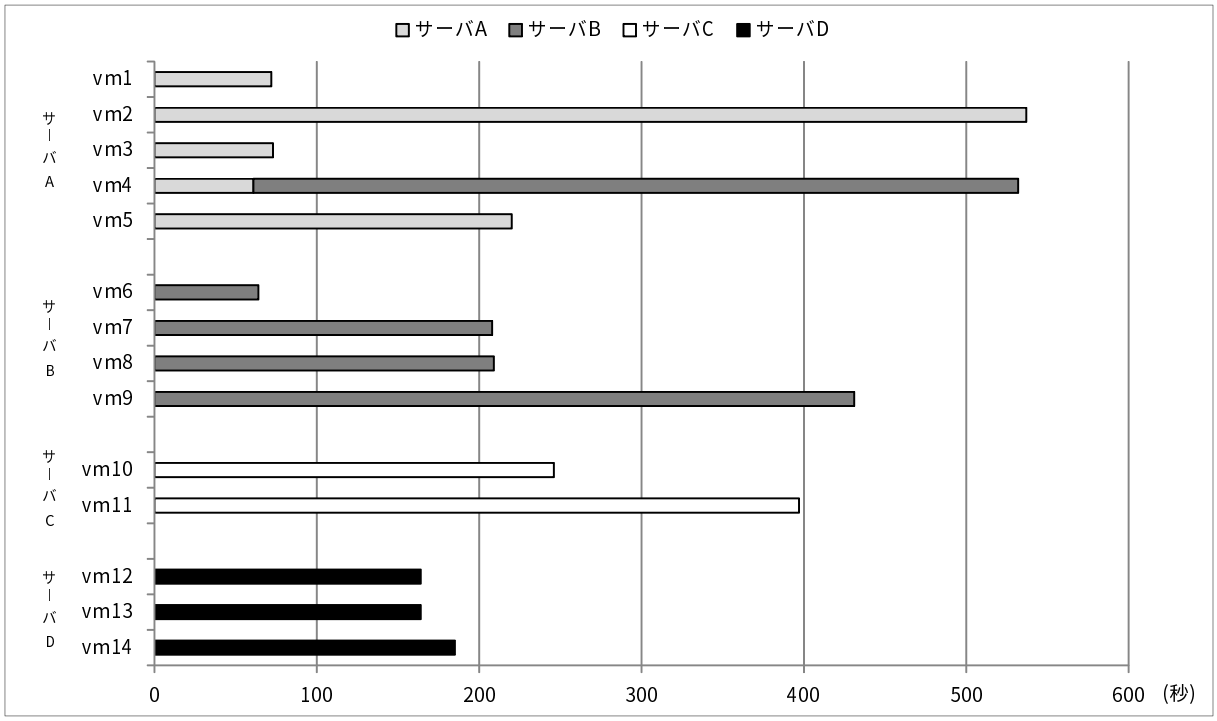

図 18: リソース競合が発生する VM の数が偏っている初期配置の場合のマイグレーションの経過

開始から 61 秒に、サーバ A の VM4 がサーバ B にマイグレーションされている。マイグレーション元サーバは最もリソース競合が発生する VM の数が多いサーバを選択するため、サーバ A が選択される。マイグレーションする VM は開始から 61 秒のサーバ A において、図 10 をみると VM2,3 で動作している libquantum が最も実行時間の伸び率が多いため、サーバ A で動作している VM の中で VM2,4 のいずれかの VM をマイグレーションするとサーバ A のリソース競合が発生している VM の中で最も MIPS 値が改善されると考えられる。実際にアプリケーションとサーバ構成との相性モデルによってサーバ A のリソース競合が発生している VM の中でマイグレーションすると最も MIPS 値が伸びると判断されたのは VM4 であり、期待通りのマイグレーション VM が選ばれていた。

マイグレーション先サーバ候補には、リソース競合が発生する VM の数が最も少ないサーバ B が選ばれることが期待され、実際にサーバ B が選択されている。

マイグレーション VM の選択は期待どおりの VM が選ばれ、リソース競合が発生

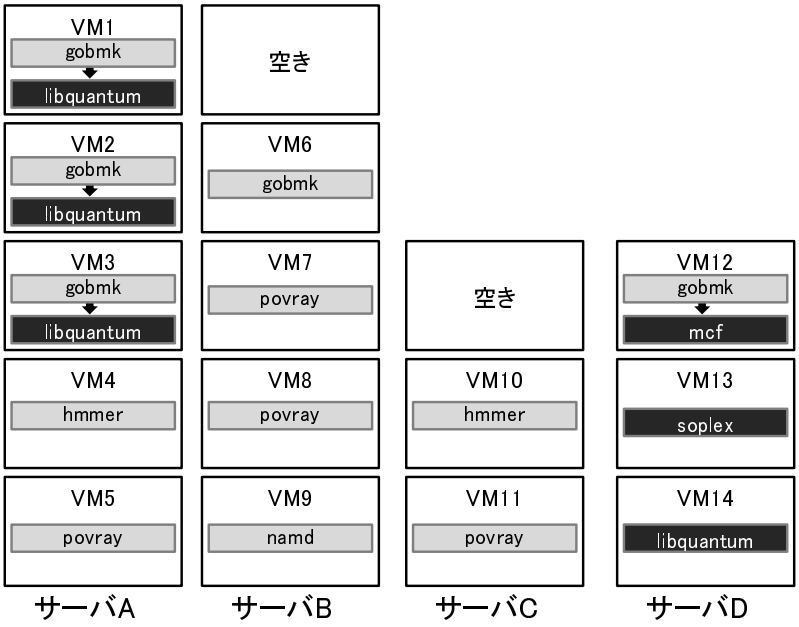

図 19: VM 上で動作するアプリケーションの特徴が途中で変化する場合

するVMは正しく判別出来ていることから、マイグレーション元サーバとマイグレーション先サーバは期待通りの選択がなされており、各物理サーバにおいてリソース競合を緩和することができ、各VM上で実行するアプリケーションの実行時間を短縮することができた。

4.3.4　VM 上で動作するアプリケーションの特徴が途中で変化する場合

サーバAとサーバDでVM上で動作するアプリケーションの特徴が途中で変化するVMを配置した環境を作成し評価を行う。配置と実行するアプリケーションを図19に示す。黒色の四角がメモリバウンドなアプリケーションを示しており、灰色の四角はCPUバウンドなアプリケーションを示す。マイグレーションができるようにサーバBとサーバCは実行可能なVMの最大数より少なく動作させている。

VM 上で動作するアプリケーションの特徴が途中で変化する場合の結果

リソース競合が発生するVMが特定のサーバに偏っている場合の各VMの実行時間の結果を図20に示す。再配置なしの場合よりも提案手法を用いた場合には、最大

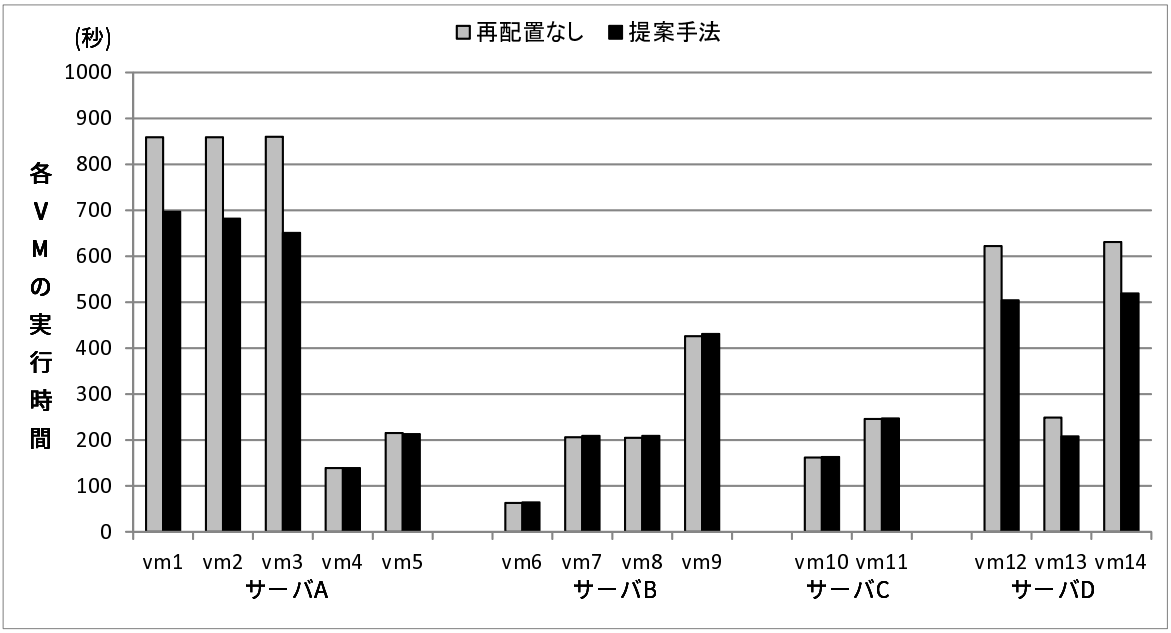

図 20: VM 上で動作するアプリケーションの特徴が途中で変化する場合の結果

で約 19%VM 上で実行されるアプリケーションの実行時間を短縮することができている。提案手法がリソース競合を緩和し、相性の良いサーバにマイグレーションがされているかを見るために各 VM が実行されていたサーバについて図 21 に示す。縦軸は初期配置の各 VM の初期配置のサーバについて示しており、各棒グラフの色は実行されていたサーバを示す。マイグレーションが行われた回数は 3 回だった。

1 回目は開始から 59 秒に、サーバ D の VM14 がサーバ B にマイグレーションされている。マイグレーション元サーバはリソース競合が発生する VM の数が最も多いサーバを選択するため、サーバ D が選択される。マイグレーションする VM は開始から 59 秒のサーバ D において、図 11 をみると VM3 で動作している libquantum が最も実行時間の伸び率が多いため、サーバ A で動作している VM の中で VM3 をマイグレーションするとサーバ A のリソース競合が発生している VM の中で最も MIPS 値が改善されると考えらる。実際にアプリケーションとサーバ構成との相性モデルによってサーバ A のリソース競合が発生している VM の中でマイグレーションすると最も MIPS 値が伸びると判断されたのは VM3 であり、期待通りのマイグレーション VM が選ばれていた。マイグレーション先サーバ候補には、リソース競合が発生する VM の数が最も少ないサーバ A,B,C が選ばれ、単体で実行した際には

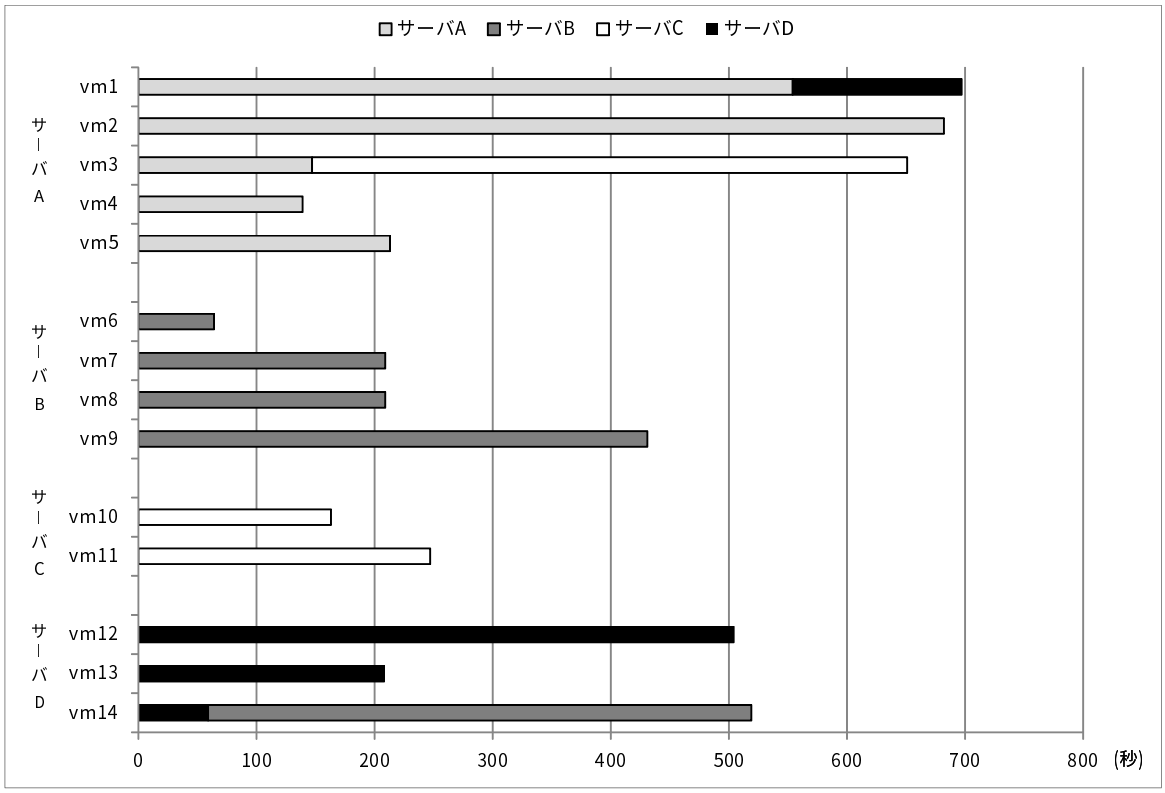

図 21: VM 上で動作するアプリケーションの特徴が途中で変化する場合のマイグレーションの経過

サーバ A,B のほうが全てのアプリケーションでサーバ C,D より実行時間が短いことから、サーバ A か B が選ばれることが期待され、サーバ A に空きが無いのでサーバ B が選ばれていた。

2 回目は開始から 147 秒にサーバ A から VM3 がサーバ C にマイグレーションされている。開始から 64 秒において、VM1,2,3 で実行されるアプリケーションが gobmk から libquantum になりメモリバウンドなアプリケーションを実行するようになるため、リソース競合が発生する VM の数が最も多いサーバはサーバ A となっている。よってマイグレーション元サーバはサーバ A となることが期待され、実際にその通りになっている。サーバ A で動作している VM 上では全て同じアプリケーションなのでマイグレーション VM はどれを選んでも同じだが、VM2 が選択されている。マイグレーション先サーバ候補には、リソース競合が発生する VM の数が最も少ないサーバ C が選択されている。

３回目は開始から、554 秒にサーバ A の VM1 がサーバ D にマイグレーションされている。マイグレーション元サーバには、リソース競合が発生する VM の数が最も多いサーバ A が選択され、マイグレーションする VM 上ではどちらも同じアプリケーションなのでどれを選んでも同じだが、VM1 が選択されている。サーバ B にマイグレーションされたサーバ D の VM3 が終了し、サーバ D の VM1 も実行が終了されたために、マイグレーション先サーバはサーバ B かサーバ D が選択され、アプリケーションとサーバの相性からサーバ B が選択されることが期待されるが、サーバ D にマイグレーションされている。

アプリケーションとサーバの相性モデルを用いたマイグレーション先サーバの予測はマイグレーションを行った 2 回中 1 回間違っていたが、VM 上で動作するアプリケーションの特徴分類モデルの予測は正しかったため、各物理サーバにおいてリソース競合を緩和することができ、各 VM 上で実行するアプリケーションの実行時間を短縮することができた。

4.4　提案手法が VM 上で実行するアプリケーションの実行時間に与える影響

全ての VM 上でメモリバウンドでないアプリケーションを実行し、再配置なしの場合と提案手法を用いた場合の実行時間を比較することで、提案手法が VM 上で実行するアプリケーションの実行時間に与える影響について調べる。VM の配置と実行するアプリケーションを図 22 に示す。

全ての VM 上でメモリバウンドでないアプリケーションを実行するために再配置が行われないので、再配置なしの場合と提案手法を用いた場合の各 VM 上で実行するアプリケーションの実行時間を比較することで、提案手法が VM 上で実行するアプリケーションの実行時間に与える影響を計測する。

提案手法が VM 上で実行するアプリケーションの実行時間に与える影響を計測するための初期配置の結果

提案手法が VM 上で実行するアプリケーションの実行時間に与える影響を計測するための初期配置の結果を図 23 に示す。横軸は各 VM と初期配置のサーバを示し、縦軸は各 VM 上のアプリケーションの実行が終了するまでの時間を示している。再

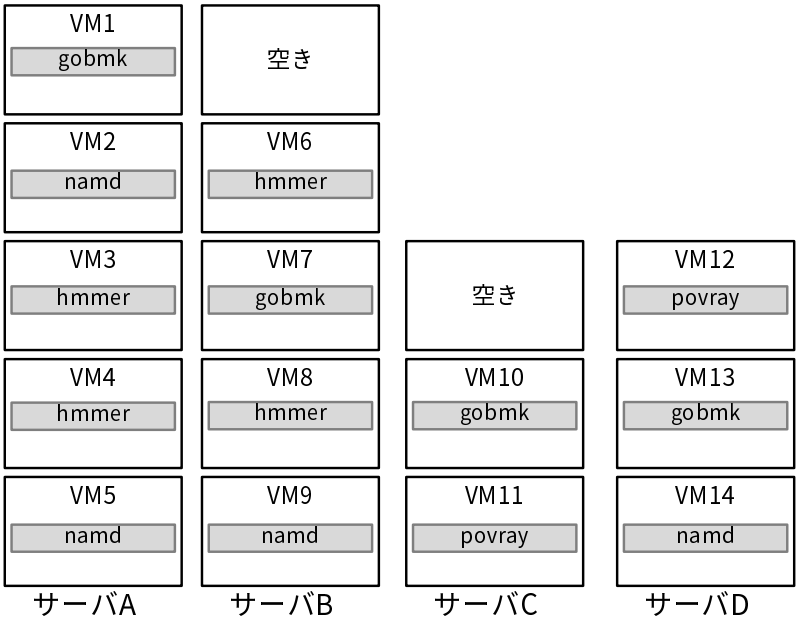

図 22: 提案手法が VM 上で実行するアプリケーションに与える影響を計測するための初期配置

配置なしの場合と比べて各 VM 上で実行するアプリケーションの実行時間の伸びは最大で約 1%であり、提案手法が各 VM 上で実行するアプリケーションの実行時間に与える影響はほぼないといえる。これは、提案手法は各 VM から特徴量としてパフォーマンスカウンタの値を取得してリソース競合の検知に用いているが、カウンタの値はプログラムの実行時間に影響を与えないように取得できるようにツールが設計されていることが多いために、カウンタ値を取得しても VM 上で実行するアプリケーションの実行時間に与える影響が小さいと考えられる。

4.5　考察

ヘテロジニアスなサーバ構成に対応できているか

ヘテロジニアスなサーバ構成に対応するために、アプリケーションとサーバ構成の相性を考慮しているが、評価実験で相性を用いてマイグレーション先を選んだ回数は、同じサーバ構成しかマイグレーション先候補がない場合を除いて 5 回であり、

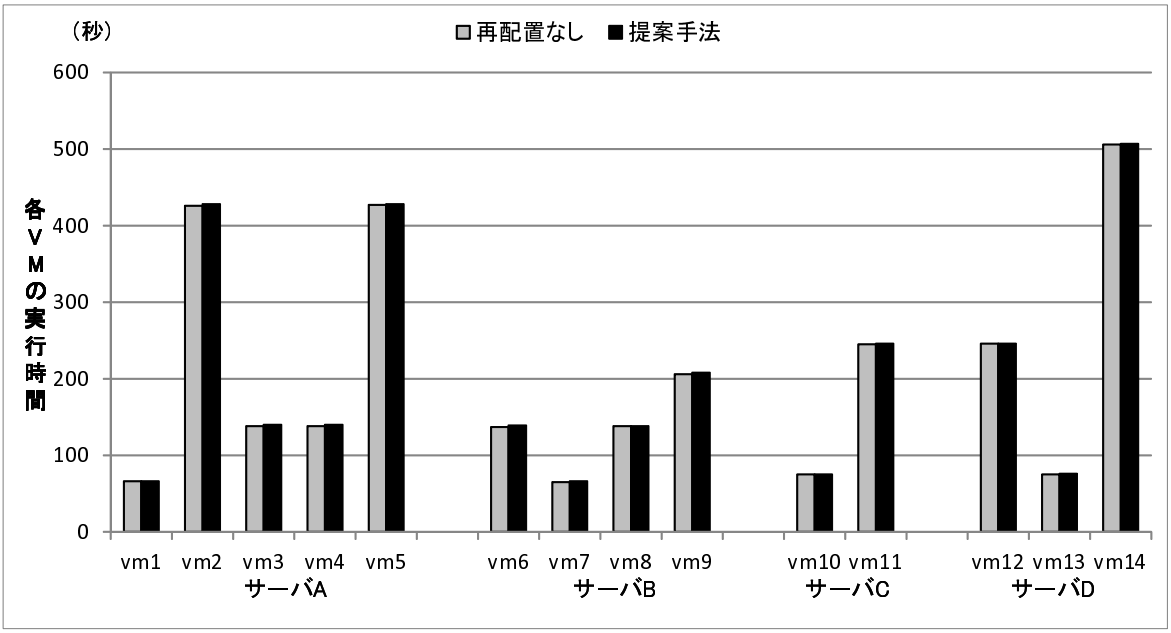

図 23: 提案手法が VM 上で実行するアプリケーションに与える影響を計測するための初期配置の結果

そのうち期待通りのマイグレーション先が選ばれたのは 2 回であった。

今回の実験ではマイグレーション候補となるメモリバウンドなアプリケーションは、表 12 に示している mcf,soplex,libquantum の 3 つであり、アプリケーション別にマイグレーション先候補サーバを見ると、soplex は相性モデルを用いて期待通りの結果が出ていたが、mcf,libqunatum の 2 つのアプリケーションの場合には期待通りの動作ではなかった。そのため、マイグレーションする VM 上で動作しているアプリケーションが mcf か libquantum であることが多かったために、期待通りにマイグレーションされた回数が少ないものと考えられる。

相性判別に用いる相性モデルは、ある VM があるサーバでどれだけの MIPS で実行できるかを予測するモデルだが、作成できたモデルは予測した値と正解の値の相関係数が 0.7 であり、期待通りの動作にならないアプリケーションも存在するモデルである。そのため、R のチューニングを現在よりもうまく選ぶか、モデルの作成に用いるパフォーマンスカウンタを現在よりも良いカウンタを選ぶことができれば、精度をより高くできると考えられる。

マイグレーション VM の選び方について

マイグレーション VM がマイグレーションすることで最も MIPS 値が改善されるものを選んでいるが、どれを選んでも同じ場合を除いて期待通り選べていたのは 4 回中 2 回であった。これは、アプリケーションとサーバの相性モデルが期待通りの動作にならないアプリケーションも存在するモデルであるからだと考えられる。

マイグレーションすることで最も性能が高くなる VM を選ぶ場合には、リソース競合が発生するアプリケーションのリソース競合の程度を考えることによって、リソース競合の程度が最も大きいアプリケーションを実行している VM をマイグレーション VM に選ぶ方法も考えられる。

5　おわりに

5.1　まとめ

ヘテロジニアスな物理サーバが複数混在する環境で、未知のアプリケーションを実行するVMをサーバに配置する際の初期配置では、VM間のリソース競合、VM上のアプリケーションとサーバ構成との相性を考慮することができない。

そこで、同時に配置するVMによって発生するリソース競合による影響、マイグレーションするVMとマイグレーション先サーバで既に配置されているVMによるリソース競合による影響、アプリケーションとサーバ構成との相性の3つを考慮した仮想マシン再配置手法を提案した。初期配置の際にリソース競合が発生するVMが偏っている場合、VM上で動作するアプリケーションの特徴が途中で変化する場合について実験を行い、その有効性を確認した。

5.2　今後の課題

今後の課題には以下が挙げられる。

- Rで用いたパラメータ最適化:
  本研究で用いた機械学習のパラメータ設定には、より良いパラメータが存在する可能性が十分に考えられる。特にVM上で動作しているアプリケーションの特徴分類モデルは予測を間違えるとリソース競合を強化してしまう可能性もあるため精度の改善は重要である。

  またアプリケーションとサーバ構成の相性モデルは、テストデータを入力として与えた際に出力されたMIPS値と、実際に取得した正解のMIPS値との間の相関係数は0.7であり、分類を行うのに十分に高い値であるとは言えない。

  パラメータを変えながらより良いパラメータを探すことが必要である。

- 異なる機械学習手法の適用:
  アプリケーションとサーバの相性モデルに、ニューラルネットワークではなく、ベイジアンネットワークなどの別の機械学習を用いることで予測精度が向上する可能性が考えられる。

- 学習に用いたアプリケーション:
  今回用いたアプリケーションは、CPU の性能評価を行うものであり、ディスクアクセスを行うアプリケーションや他の VM と通信を行うアプリケーションが存在していないため、それらのアプリケーションに対しては対応出来ない。

  実際のデータセンタでは、様々なアプリケーションが実行されることから上記のような処理を行うアプリケーションに対しても対応する必要があるので、アプリケーションの種類をより増やす必要がある。

謝辞

本研究を進めるにあたり、日頃からご指導頂いた指導教員の本多弘樹教授に深く感謝いたします。研究内容について的確なアドバイスをしてくださった近藤正章准教授 (現東京大学)、和田康孝助教 (現早稲田大学) に深く感謝いたします。並びに、日常の議論を通じて多くのアドバイスを頂いた本多研究室の皆様に感謝いたします。

また、多方面に関して相談に乗って頂いた情報システム基盤学講座の同期の皆様、特に尾久君に感謝いたします。

参考文献

[1] Amazon EC2. http://aws.amazon.com/jp/ec2/.

[2] OpenStack. http://www.openstack.org/.

[3] Oprofile. oprofile.sourceforge.net/news/.

[4] R. http://www.r-project.org/.

[5] SPEC CPU 2006. https://www.spec.org/cpu2006/.

[6] Xen. http://xenproject.org/.

[7] XenOprof. http://xenoprof.sourceforge.net/.

[8] Dejan Novaković, Nedeljko Vasić, Stanko Novaković, Dejan Kostić, and Ricardo Bianchini. Deepdive: Transparently identifying and managing performance interference in virtualized environments. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference*, USENIX ATC'13, pp. 219–230, Berkeley, CA, USA, 2013.

[9] Fei Xu, Fangming Liu, Hai Jin, and A.V. Vasilakos. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE*, Vol. 102, No. 1, pp. 11–31, Jan 2014.

[10] Fei Xu, Fangming Liu, Linghui Liu, Hai Jin, Bo Li, and Baochun Li. iAware: Making live migration of virtual machines interference-aware in the cloud. *IEEE Transactions on Computers*, Vol. 63, No. 12, pp. 3012–3025, Dec 2014.

[11] 首藤裕一, 波戸邦夫. パブリッククラウドの時間的性能変動および性能分布 (ポスト ip ネットワーキング, 次世代・新世代ネットワーク (ngn), 障害対策・bcp, ネットワークコーディング, セッション管理 (sip・ims), 相互接続技術/標準化, ネットワーク構成管理及び一般). 電子情報通信学会技術研究報告. IN, 情報ネットワーク, Vol. 113, No. 206, pp. 13–18, sep 2013.

[12] 穂園智哉, 近藤正章, 平澤将一, 本多弘樹. 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. 情報処理学会研究報告. 計算機アーキテクチャ研究会報告, Vol. 2011, No. 12, pp. 1–8, mar 2011.