修士論文

Hadoop データセンタにおけるオークションメカニズムを用いた料金設定と計算資源割当て手法の設計と評価

真鍋 優

2019 年 3 月 13 日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士（工学）授与の要件として提出した修士論文である．

真鍋 優

審査委員：

笠原 正治 教授	（主指導教員）
飯田 元 教授	（副指導教員）
笹部 昌弘 准教授	（副指導教員）
川原 純 助教	（副指導教員）

Hadoop データセンタにおけるオークションメカニズムを用いた料金設定と計算資源割当て手法の設計と評価*

真鍋 優

内容梗概

記憶容量の増大や計算性能の上昇といったハードウェアの発展に伴い，アプリケーションの取り扱うデータ量や計算量は日々増大している．そこで注目されているのが大規模なコンピュータクラスタにおいて処理を分散することで高速化を図る Hadoop 等の分散処理フレームワークである．Hadoop は元来少人数での使用を想定されているため，大規模な計算資源を多人数で共有するには計算資源の割当てを行う新しいスケジューラが必要である．加えて，データセンタ等において商業目的で Hadoop を運用するには，データセンタ管理者の収益を考慮した料金の設定手法が必要になる．本研究は利用者の入札に基づいたオークションにより計算資源の割当てを行うスケジューラを提案する．本提案手法は 2 つの要素から構成される．1 つ目は，利用者が実行するアプリケーションの実行時間を予測するシステムである．アプリケーションの予測実行時間を利用者に提示することで，利用者がアプリケーション完了までの締め切りである期限と完了までに支払うことのできる金額である予算に合わせた評価額を設定することが可能になる．2 つ目は利用者にとって自身の評価額をそのまま入札することが最適となるようなオークションメカニズムである．本提案手法の評価実験として離散事象シミュレーションを実施した．評価実験では，Hadoop の標準スケジューラの 1 つである FIFO スケジューラおよび先行研究で提案された Dynamic Priority parallel task scheduler（以後 DP スケジューラ）と比較を行った．比較した項目は，予算内でアプリケーションを完了した利用者の割合である予算内完了率，期限内でアプリケーションを完了した利用者の割合である期限内完了率，データセンタ管理者の収益の 3 点である．評価実験では，提案手法は利用者の参加および離脱が頻繁な状況において，特にデータセンタ管理者

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, 2019 年 3 月 13 日.

の収益に関して DP スケジューラと比較した場合最大で約 125% 増加した.

キーワード

分散処理，Hadoop，データセンタ，スケジューリング，オークションメカニズム

Design and Evaluation of Pricing and Resource Allocation on Hadoop Data Centers Using Auction Mechanisms*

Yu Manabe

Abstract

As storage capacity and computing performance grow due to the development of hardware it is possible for applications to handle much more data and computation resources. Therefore, distributed processing frameworks that accelerate execution of jobs in a large sale cluster such as Hadoop attract the attention. Since Hadoop is designed to be concurrently used by a small number of users, if a large number of users share big data, a new scheduler for resource allocation is needed. In addition, in order to manage a Hadoop data center for commercial purposes, a method of pricing that increases the revenue of a data center is required. In this research, we propose an auction based scheduler for resource allocation. The proposed method consists of two components. The first component is a system to predict the execution time of a user's application. By informing the predicted time to the user, it becomes possible to decide the appraised price of executing jobs according to the budget, the money that the user can pay, and the deadline, the time limit for the application being completed. The second component is an auction mechanism such that it is optimal for users to bid their own appraised price as they are. Discrete event simulations are carried out to evaluate the proposed method. In the evaluation experiment, we compare the method to two other schedulers, FIFO scheduler, which is implemented in Hadoop, and Dynamic Priority Parallel Task Scheduler

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, March 13, 2019.

(DP scheduler), which is proposed in previous research. The proposed method is evaluated in terms of the ratio of the number of users who complete their applications within their budget to that of all users, the ratio of the number of users who complete their applications within their deadline to that of all users, and the revenue of the Hadoop data center. The evaluation experiment indicates that the proposed method increases the revenue of the Hadoop data center by about 125% compared with the DP scheduler in the situation where user participation and leaving are frequent.

Keywords:

Distributed processing system, Hadoop, Data center, Scheduling, Auction mechanism

目次

図目次

表目次

第 1 章　はじめに

記憶容量の増大や計算性能の上昇といったハードウェアの発展に伴い，アプリケーションの取り扱うデータ量や計算量は日々増大している．そこで注目されているのが大規模クラスタにおいて処理を分散することで高速化を図る Apache Hadoop [1] 等の分散処理フレームワークである．Hadoop とは Google が開発した MapReduce [2] に影響を受け開発されたオープンソースソフトウェアである．MapReduce とはアプリケーションを Map として細分化した上で各計算ノードで実行し，結果を Reduce としてまとめる分散処理フレームワークである．

Hadoop バージョン 2 では Hadoop バージョン 1 のスケーラビリティを改善するため，YARN [3] が実装された．図 1.1 に YARN のイメージ図を示す．YARN はマスターとスレイブのノードから構成され，それぞれ ResourceManager と NodeManager がアプリケーションおよび各ノードの管理を担う．利用者がアプリケーションを ResourceManager に投入すると ResourceManager は NodeManager が管理するコンテナにおいて ApplicationMaster を起動する．コンテナとは計算資源の単位であり，アプリケーションに対し複数個割当てられる．ApplicationMaster とは，アプリケーションごとに起動され，アプリケーションの実行を監視，管理する機能を持つ．ResourceManager により起動された ApplicationMaster は，割当

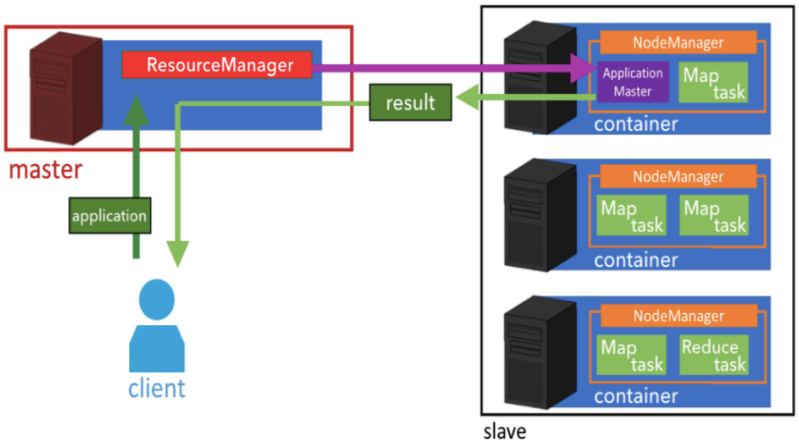

図 1.1　Hadoop イメージ図

てられたコンテナにおいて Map および Reduce 処理を並列で実行する．

YARN の導入により 4,000 ノードを超える大規模クラスタの管理を実現した [3] ことで，計算量の膨大な科学技術計算やビッグデータの解析等のための運用が期待されている．しかし大規模なクラスタや膨大な量のデータセットを利用者が個々人で用意することは容易ではなく，それらを複数人で共有することが求められる．本研究は利用者の利用料金とデータセンタ管理者の収益を考慮した，複数人で共有可能な Hadoop データセンタを構築することを目指す．

Hadoop を導入したクラスタで複数人の同時利用を可能とするために解決すべき主な課題として，動的な計算資源割当て手法の考案が挙げられる [4]．先行研究において，計算資源の割当てを利用者の申告する時間あたりの利用料金から決定する，Dynamic Priority parallel task scheduler [4]（以後 DP スケジューラ）なる手法が提案されている．DP スケジューラとは，単位時間あたりに利用者が支払い可能な金額である支出率を申告し，全利用者の内どれほどの支出率を申告したかで支払い金額と利用可能なコンテナ数が決定されるスケジューラである．しかし DP スケジューラでは，支出率の割合で利用可能なコンテナ数が決定するため，並列する利用者の数が少ないとき安価な利用料金で多くの計算資源を利用できてしまう．この問題により，データセンタ管理者の収益が減少することが考えられる．また，利用者にとってアプリケーションが完了するまでに必要な支払い金額が不明であり，支出率を適切に設定することが困難である．支出率が適切ではない場合，利用者が持つアプリケーション完了までの締め切りである期限内にアプリケーションが完了しないことや，利用者がアプリケーションの実行に支払える金額の上限である予算を支払い金額が上回る恐れがある．

本研究では，アプリケーションの実行時間を考慮したオークションメカニズムを導入した新しいスケジューラを提案する．本提案手法はアプリケーションが完了までに必要な予測時間を利用者に提示することで，単位時間あたりに必要な金額を明らかにし，より多くの利用者の期限内かつ予算内のアプリケーション完了を目指す．本提案手法は 2 つの要素から構成される． 1 つ目は，利用者が実行するアプリケーションの実行時間を予測するシステムである．本研究では，Hadoop 環境で実行するアプリケーションは，MapReduce によるファイル操作のような，処理時間がデータ量に比例するものを想定している．本システムによってアプリケーション

の予測実行時間を利用者に提示することで，利用者がアプリケーション完了までの締め切りである期限と完了までに支払うことのできる金額である予算に合わせた評価額の設定をすることが可能になる．２つ目は利用者にとって自身の評価額をそのまま入札することが最適となるようなオークションメカニズムである．

本提案手法の評価実験として離散事象シミュレーションを実施し，予算内でアプリケーションを完了した利用者の割合である予算内完了率，期限内でアプリケーションを完了した利用者の割合である期限内完了率，データセンタ管理者の収益の 3 点について既存手法と比較を行った．また，Hadoop クラスタへの実装を行い，利用者の入札額に従うコンテナの割当てを実現した．

続く第 2 章では Hadoop クラスタ共有のための計算資源割当てを行うスケジューラや料金設定に関する先行研究について紹介する．第 3 章では本研究で提案する手法について説明し，第 4 章においてその評価のために行ったシミュレーション実験と Hadoop クラスタにおける実装について述べる．第 5 章で本研究の結論と今後の課題を記す．

第 2 章　先行研究

本章では Hadoop に標準で実装されているスケジューラと，複数人による計算資源共有のため提案された先行研究，そして本研究で用いるオークションメカニズムに関連する先行研究について述べる．

2.1　Hadoop の標準スケジューラ

Hadoop は元来少人数による利用が想定されており，それは計算資源の割当てを行うため Hadoop が標準で搭載している FIFO，Fair，Capacity という 3 種のスケジューラの仕様からも伺い知ることができる．

FIFO スケジューラはアプリケーションを到着順に実行するスケジューラである．最初に到着したアプリケーションに全ての計算資源を割当て，終了次第，次に到着したアプリケーションに計算資源を割当てる．

Fair スケジューラは全利用者に割当てる計算資源の量を均等にすることを試みるスケジューラである．Fair スケジューラでは事前に利用者ごとに対応するプールとその優先度を設定する必要がある．利用者は設定されたプールにアプリケーションを投入する．投入されたアプリケーションには，同プール内の他利用者が投入したアプリケーションに割当てられた計算資源の量と等しくなるように計算資源が割当てられる．プールを複数の利用者で共有するため，FIFO とは異なり，計算資源を長時間保持している場合に割込みを発生させることが可能となっている．

Capacity スケジューラは事前に利用者毎に用意されたキュー設定に入力した割合だけ計算資源を割当てるスケジューラである．例えば利用者 A のキュー設定を 30%，利用者 B のキュー設定を 70% と設定した場合，利用者 A が投入するアプリケーションはクラスタ全体の計算資源の 30% を，利用者 B は 70% を使用することができる．この割合はクラスタ全体の CPU 性能あるいはメモリ使用量を表している．

これらのスケジューラは，割込みができないため大人数の利用者が並列で実行することが不可能であったり，事前にデータセンタ管理者によって割当てる計算資源の量を全利用者ごとに設定する必要があるなどの特徴がある．そのため，複数の利

用者の参加離脱が考えられる状況で必要となる動的な性質に欠けているという問題があり，大人数で利用する際には新たなスケジューラが必要となる．

2.2 先行研究におけるスケジューラ

大人数により Hadoop クラスタを共有する際に，データのロケーションに従いスケジューリングを行う Delay スケジューリング [5] が提案された．Hadoop のファイルシステムにおいて，データは複数のファイルに分割され，クラスタを構成するノードが分担し保存する．Delay スケジューリングでは，アプリケーションが使用するデータが多く保存されているノードにおける計算資源をそのアプリケーションに優先的に割当てる事で，データの移動によるオーバーヘッドを軽減する．また，各利用者が使用するデータのローカリティの偏りを軽減することで，利用者間における公平な計算資源の利用を可能にする．使用するデータが多く保存されているノードの計算資源が他のアプリケーションに使用されている場合は，意図的にアプリケーションの実行を遅らせる事で，よりデータローカリティを高めることを可能としている．

Hadoop データセンタにおいて各利用者が実行するアプリケーションは，ファイル I/O バウンドや CPU バウンド等，それぞれ特徴があり，要求する計算資源のリソースの種類はそれぞれ異なる．そこで，アプリケーションの必要とするリソースを推定し，その特徴に従った計算資源の割当てを行う Triple-Queue スケジューラ [6] が提案された．Triple-Queue スケジューラでは，各利用者が異なる種類のアプリケーションを実行した際に，異なる種類のアプリケーションを同一のノードにおいて実行することで，CPU およびディスクの使用率を改善することができ，Hadoop のスループットを約 30% 向上させることを実現した．

他にも Hadoop クラスタにおけるスケジューラとして，ノードの数やアプリケーションの計算量，アプリケーションが必要とするリソースの種類等を合わせて判断し，計算資源の割当てを行う HybridMR スケジューラ [7] や，各ノードごとの CPU，メモリ，ネットワーク利用率をそれぞれ監視し，ノードごとに実行する Map 数や Reduce 数を調整する Dynamic Slot Controller [8] などが提案されている．Hadoop のスケジューラはリソース，データ，計算量，アプリケーションの特徴な

どの観点から多くのスケジューラが考案されている [9]．本研究では，データセンタを商業目的で運営する事を想定しているため，特に利用者の支払い金額に着目したスケジューラについて注目した．

Hadoop を用いた環境において，利用者の支払い金額に従い計算資源を共有する DP スケジューラ [4] が提案された．DP スケジューラの利用者は事前に，各自が時間ごとに 1 コンテナあたりに支払うことのできる金額である支出率を申告する．そしてそれぞれに割当てられる計算資源の量は，支出率を用いた以下の式から決定する．

$$\left(s_i / \sum_j s_j \right) \times c$$

s_i は利用者 i の支出率，c はクラスタ全体のコンテナ数を表す．例えば利用者 A が支出率を 0.3，利用者 B が 0.7 と申告し利用者がその 2 名のみとすると，割当てられるコンテナは利用者 A に全体のコンテナ数の 30%，利用者 B に 70% となる．割当てられた複数のコンテナにおいて利用者はアプリケーションを実行し，一定間隔ごとに時間あたりに使用したコンテナ数に支出率を掛け合わせた分の料金を支払う．

DP スケジューラを利用する場合，新しく到着する利用者や，実行を終え離脱する利用者の設定をデータセンタ管理者は参加および離脱の度に行う必要がない．また，一定間隔ごとに計算資源の割当ての比率を計算することで，動的な利用が可能となっている．先行研究における評価実験において，DP スケジューラは各ユーザに割当てる計算資源の割合を動的に変化させることと，最大 80 名のユーザによる並列利用が可能であることが示されている．

DP スケジューラの仕様では，利用できる計算資源の量は全体の支出率の単純な割合から決定している．この仕様により，2 つの問題が発生すると考えられる．1 つ目は利用者は全体の利用者が自身のみであると知った場合，支出率を極端に減少させることが可能である点である．例えば，利用者 i のみがデータセンタを利用している場合，利用できるコンテナ数は利用者 i の支出率 s_i を用いて，$(s_i/s_i) \times c = c$ となる．この際，s_i の大小によって利用できるコンテナ数は増減しない．この問題により，Hadoop データセンタにおける利用者数が減少した際，利用者は自らの支出率を低下させ，それに伴いデータセンタ管理者の収益が減少することが考えられ

る．2 つ目は利用者がアプリケーションの実行をある期限内に完了させたい場合，どれほどの支出率を申告することが必要か不明である点である．これは支出率の大小と利用できるコンテナ数の関係が不明瞭であり，また他の利用者の支出率に大きく左右されることが原因である．

データセンタ管理者の収益を確保するためには，並列する利用者の数が少ないとき，利用者の計算資源に対する評価額を下回る申告でも多くの計算資源を利用可能となってしまうことを防ぐ必要がある．加えて，利用者に対して Hadoop データセンタというサービスを利用する利点を示す必要があり，十分な評価額を持つ利用者は，期限および予算内でアプリケーションの処理を完了することが重要となる．

2.3 オークション理論に関する研究

本研究では，利用者の評価額を正直に申告することが最適な戦略となるような手法の実現にオークションメカニズムを用いる．本研究において用いるオークションメカニズムとはオークションの仕組みを取り入れた計算資源割当て手法であり，データセンタの利用者および管理者の利得を増加させることを目的とする．オークション理論とは社会全体の利得である社会的余剰を最大化することを目的とした資源分配の理論であり，仮想マシン等の計算資源割当てや，インターネットオークションの分野において昨今注目を集めている [10] [14] [17] ．基本的に，オークションでは売り手が公開した単一の財，あるいは複数の財に対して買い手は自らの評価額を入札し，最大の評価額を入札した買い手が財を落札する．オークション理論を導入した仕組みを設計するにあたって，不正が行われないことと，買い手にとって評価額を正直に入札することが最適であるように設計することが求められる．この特徴により利用者の虚偽の申告を防止することが可能となる．

オークション理論はデータセンタでの利用にも応用できると期待されており，関連する先行研究として [10] や [11] がある． [10] は商用のクラウドサービスにおける料金設定に関する内容となっている．Amazon EC2 [12] のスポットインスタンス [13] 等，仮想マシンをオークションによる料金設定に基づいて提供するサービスが存在する．しかし，現状ではオークションによって決定される仮想マシンの利用料金は，仮想マシンの性能の違いを十分に考慮できていない．そのため市場の要求

に適切に応えるためには，仮想マシンに対する利用者の要求・評価を正しく把握し，複数の財を取り扱う組み合わせオークションによって異なる種類・場所の仮想マシンをまとめて取り扱う必要がある．[10] ではデータセンタや仮想マシンの種類の違い，利用者の予算等，多くの制約を含んだオークションメカニズムを設計している．

[11] は電力を効率的に送電するスマートグリッドにおいて，データセンタへの送電量にデマンドレスポンスを導入することを目標とした研究である．デマンドレスポンスとは，電気料金の高騰のような状況の変化に応じて需要者側から送電量を変化させることである．デマンドレスポンスは消費電力を削減し，電力の効率的運用を行う優れた手法であるが，データセンタにおいて電力の減少は可用性の面から避けるべきと考えられ，導入が難しいとされている．[11] ではその課題の解決のためにオークション理論を導入し，データセンタとスマートグリッドの社会的余剰を最大化することを試みた．

売り手の利益を保証するためには，オークションに出品する財の下限を設けることが有効であると考えられる．財の料金の下限を用いた複数財のオークションメカニズムとして Leveled Division Set protocol [14] (以後 LDS プロトコル) が挙げられる．LDS プロトコルでは売り手が事前に財の全ての組み合わせに売値の下限である留保価格を設定する．一方買い手もそれらの財の組み合わせ全てに入札を行い，オークションを行う．オークションは組み合わせた財の数が最も多いものから行い，その組み合わせの留保価格を上回る入札があった場合，落札となる．落札した際の支払額は一般化 Vickrey オークション [15] により決定される．一般化 Vickrey オークションとは，最高額の入札を行った買い手が 2 番目に高い入札額で落札する Vickrey オークション [16] を複数財に適用した手法である．一般化 Vickrey オークションにおいて，落札した買い手の支払額は，落札した買い手が参加しない場合の社会的余剰から落札した買い手が参加した場合の他の買い手の社会的余剰を引いた額となる．これらの値は自身の評価額に左右されることはないため，買い手は自身の評価額通りの入札を行うことが最適となる．LDS プロトコルではそのような一般化 Vickrey オークションの特徴を満たしつつ，一般化 Vickrey オークションの課題であった架空名義による入札の問題を解決することに成功した [14].

売り手と買い手がそれぞれ複数存在するようなダブルオークションの一例として [17] が挙げられる． [17] は [18] を元に提案された手法である．複数人の利用者と

複数のデータセンタ間でのオークションにおいて，利用者の入札額およびデータセンタの設定金額を状況に合わせて設定することで，データセンタの使用率の上昇，利用者の実行するジョブの設定された期限内での実行などを可能とする．利用者の入札額は 2 つの要素から決定される．一つ目は利用者の実行するジョブの実行状況である．ジョブの完了までの時間が長ければ入札額は大きくなり，短ければ小さくなる．二つ目は未使用データセンタの数である．ジョブが使用することのできるデータセンタ数が大きければ入札額は小さくなり，小さければ大きくなる．これら二つの要素から最終的な入札額が決定される．自動的に決定した入札額と設定金額から，ジョブの割当てが行われる． [17] において，シミュレータによる実験により利用者とデータセンタ管理者両者の利益を示した．

本研究においてオークションメカニズムを用いる理由は，利用者に自らの評価額を正直に入札させ，入札された評価額を用いた料金設定により利用者と管理者の社会的余剰を増加させるためである．利用者にとって自らの評価額を正直に入札することが最適な戦略となるオークションのことを正直なオークションという [19]．例えば，他者の入札を知ることができ，複数回の入札において最高額の入札を行なった入札者がその金額で落札するというイングリッシュオークションは正直なオークションではない．これはイングリッシュオークションにおいて入札者は他者の入札が無くなるまで自らの評価額を越えないように少しずつ入札額を増加させることが最適な戦略であるからである．自らの評価額未満において他者の入札が存在しなくなった場合，落札した入札者は本来支払う意思のある金額以下で財を入手できる．一方で売り手は，落札者の評価額を下回る金額で財を売ることになり，本来得ることのできる利益を失うこととなる．正直なオークションを実現するためには，落札者の入札額は落札者の支払う金額に影響せず，またその落札額は落札者の入札額以上にはならないという条件を満たす必要がある．先に述べた一般化 Vickrey オークションはこの条件を満たし，この条件下では入札者は自らの支払うことのできる限界である評価額通りの入札を行うことが最適な戦略となる．

第 3 章　提案手法

本研究において提案する手法は，正直なオークションの条件を満たすメカニズムを導入することで，利用者の評価額を正しく反映した料金設定を実現することを目的とする．また，利用者が自らの予算および期限内でアプリケーションを実行するための評価額を正しく設定可能になることを目指す．

本提案手法は大きく分けて 2 つの機能から構成される．1 つ目は利用者が自らの評価額を正しく設定するための実行時間予測システムである．実行するアプリケーションの実行時間と，利用者の持つ期限内に実行を完了するために必要なコンテナの数である必要コンテナ数を利用者に提示することで，利用者の期限と予算に合わせた評価額を設定可能とする．２つ目は正直なオークションメカニズムであり，利用者は評価額を正直に入札することが最適な戦略となる仕組みとなる．本章ではそれぞれの手法について詳細に述べる．

3.1　Hadoop 環境における実行時間予測

アプリケーションの実行時間予測はプログラムの仕様や実行環境等，多くの要因に影響され困難であるが，Hadoop における MapReduce に限定した場合，入力テキストデータの検索など，処理量が入力量に比例する場合はある程度の予測は可能である．これは MapReduce が，アプリケーションをおおよそ均一のサイズの Map に分割し実行するという特徴を持つためである．処理の開始から終了まで実行される処理量はほぼ同一であるため，逐次的に処理を観測することで，残りの実行時間を予測することが可能である．本研究ではその様なアプリケーションを扱う．アプリケーションの実行開始後，時間 t が経過した時点における残りの実行時間 τ_t は以下の式から算出する．

$$\tau_t = \frac{t(C_{max} - C_t)}{C_t}$$

C_{max} はアプリケーションが完了までに実行する Map 処理の個数を表し，この値は実行開始時に知ることができる．アプリケーションは割当てられた複数のコンテナにおいて並列に順次 Map 処理を実行していく．C_t はアプリケーションの実行開

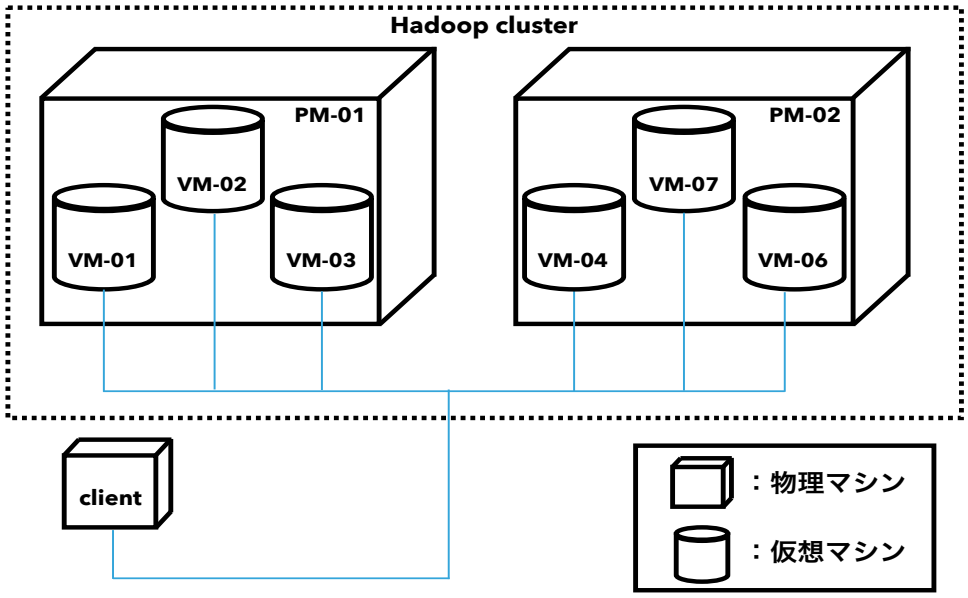

図 3.1　Hadoop クラスタ構成図

始から時間 t が経過するまでに完了した Map 数を表す．上記の式は残りの Map 数と 1 つの Map を完了するために必要な時間を掛け合わすことで残りの実行時間を予測している．

Hadoop 環境における実行時間予測の精度を検証するため，Hadoop を導入したコンピュータクラスタにおいて，実際に検証を行った．検証は複数台の仮想マシンから構成されるコンピュータクラスタにおいて実行し，一定間隔ごとに残りの実行時間を予測する．

3.1.1　実行環境

実行時間予測の検証は Hadoop データセンタに見立てた，複数の仮想マシンから構成される Hadoop クラスタにおいて実施した．全体の構成図を図 3.1 に記す．

本 Hadoop クラスタは 2 台の物理マシン上に導入した仮想マシン 6 台から構成される．物理マシン 2 台の性能は同等であり，仮想マシン 5 台も同じく性能を均一

に揃えている．また物理マシンはスイッチを介して物理回線で接続されている．それぞれのマシン性能の詳細について表 3.1 に記す．

表において PM はそれぞれ物理マシンを，VM は仮想マシンを表す．vCPU は

表 3.1　物理マシンおよび仮想マシンの性能

マシン名		OS	VCores	Memory[GB]	Storage[GB]	Node
PM-01		CentOS 7.5.1804	8	32	256	-
	VM-01	CentOS 7.3.1611	2	16	30	ResourceManager
	VM-02	CentOS 7.5.1804	2	8	30	NodeManager
	VM-03	CentOS 7.5.1804	2	8	30	NodeManager
PM-02		CentOS 7.5.1804	8	32	256	-
	VM-04	CentOS 7.5.1804	2	8	30	NodeManager
	VM-05	CentOS 7.5.1804	2	8	30	NodeManager
	VM-06	CentOS 7.5.1804	2	8	30	NodeManager

物理 CPU および仮想 CPU のコア数，Memory と Storage はそれぞれのメモリとストレージの容量を示している．また Node は Hadoop における項目であり，仮想マシンが担当する役割を表している．ResourceManager とはジョブの管理を行う機能であり，本クラスタにおいて VM-01 をマスターのノードとしている．NodeManager は同マシンが持つコンテナにおける処理を管理しており，VM-01 以外の仮想マシンはスレイブノードとしての役割を持つ．

3.1.2　検証結果

検証で使用したアプリケーションは，モンテカルロ法を用いた円周率計算と，ランダムに作成したテキストファイルに対し単語の出現回数を数えるワードカウントの 2 種類である．それぞれ 3 分，5 分ごとに予測時間を計算し記録を行った．予測結果を図 3.2 および図 3.3 に記す．

円周率計算では総じて 10,000 個の Map が，ワードカウントでは 64GB のテキストファイルを分割し 650 個の Map として処理が実行される．図 3.2 で示す円周率計算の予測時間の誤差は，最初の計測時が最も大きく，全体の実行時間に対し約 11% であり，3 回目の計測では約 8% まで誤差は減少している．図 3.3 のワードカ

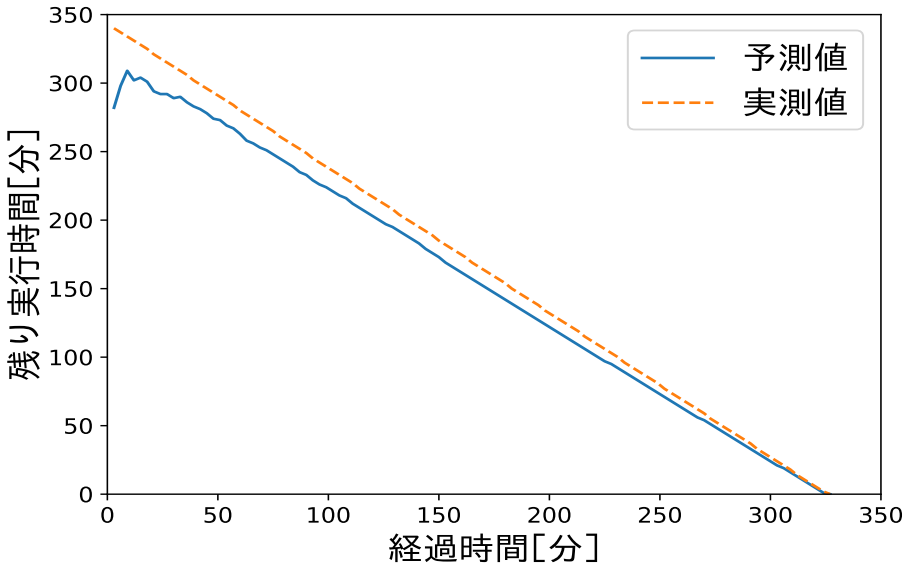

図 3.2　円周率計算における実行時間予測

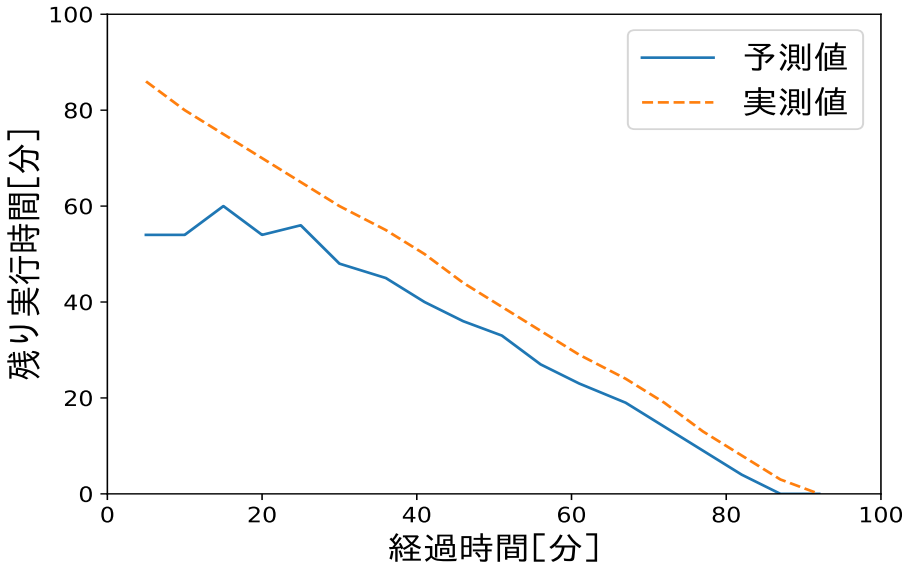

図 3.3　ワードカウントにおける実行時間予測

ウントでは，最初の 2 回の計測で誤差が全体の実行時間に対し約 38%，33% となるが，その後は約 17% 以下となる．円周率計算とワードカウントのどちらの結果においても，最初の 5 回の計測における誤差を除くと予測の誤差は最大 17% であった．前半の残り実行時間の予測誤差が大きいことは，コンテナの初期化等のオーバーヘッドが発生していることが原因と考えられる．予測を全体の処理の約 8% が完了するまで実行することで，残りの実行時間をほぼ正確に予測できることが期待される．

この予測実行時間を用いることで，利用者の期限内にアプリケーションを完了するために必要なコンテナ数を計算する．利用者 i における必要コンテナ数 n_i^{need} はデータセンタ全体のコンテナ数を N としたとき以下の式により定義する．

$$n_i^{need} = N \times \left(\frac{p_i}{l_i} \right)$$

p_i は利用者 i の実行するアプリケーションを，全コンテナを用いて実行した際の予測実行時間を表す．l_i は利用者 i の設定する期限であり，利用者 i は到着してから時間 l_i が経過するまでにアプリケーションが完了する事を希望している．上記の式では予測実行時間の計測の際に全コンテナ N を使用しているため，必要コンテナ数が全コンテナ数を上回ることはできず，$p_i < l_i$ であることが条件となる．また，利用者はアプリケーションの実行において期限をデータセンタに申告するため，データセンタ管理者は全利用者の予測実行時間 p_i および期限 l_i を把握している．

3.2 オークションメカニズム

本研究では利用者 i はアプリケーションの完了までの期限 l_i と，完了までに支払い可能な金額である予算 B_i を持つと仮定する．利用者は自らの期限と予算に合わせて入札を行い，コンテナを落札するためオークションに参加する．オークションは一定期間ごとに実施され，オークションの合間に入札を行った利用者を対象に割当てるコンテナ数と利用料金を決定する．オークションによってコンテナが割当てられた利用者は利用料金を支払い，次回のオークションが実施されるまでの間アプリケーションを実行することが可能となる．本提案手法は次のステップで実施する．

1. 利用者が処理と期限をデータセンタに提示.
2. データセンタは実行時間予測システムを用いて予測実行時間と必要コンテナ数を利用者に提示.
3. 利用者は自らの予算と予測実行時間，必要コンテナ数を参考に入札.
4. 入札した利用者を対象にオークションを実施し，コンテナを割当て，料金を徴収.
5. 利用者は落札したコンテナにおいて次回のオークションが開始されるまでアプリケーションを実行.
6. 次のオークションを実施，利用者はアプリケーションが完了していない場合続けて入札.

オークションでは入札額の大きい利用者から順に割当てを開始する．割当てられるコンテナの数はコンテナ数ごとに設定された最低予算額を用いて決定される．例えば必要コンテナ数が 10 の利用者に対して割当てが行われる場合，コンテナ数 10 の最低予算額と当利用者の入札額を比較する．入札額が最低予算額以上であった場合はコンテナ 10 個が割当てられ，最低予算額未満であった場合は，コンテナ数 9 の最低予算額と同様に比較を行い割当てを試みる．最低予算額は全入札中，2 番目に大きい入札を行った利用者の情報を用いて以下の式より計算される．

$$m_n = \left[\left(\frac{b_2}{l_2} \right) \times \left(p_2 \times \frac{N}{N-n+1} \right) \right]^{1/\alpha}$$

上記の式は全コンテナ数が N 個であるデータセンタにおいて，コンテナ n $(= 1, \ldots, N)$ 個を利用するために必要な入札額である最低予算額 m_n を表す．b_2 は第二価格入札を行った利用者の入札額である．α はデータセンタ管理者が任意に決定するパラメータであり，この値を変更することで最低予算額を調整することができる．割当てるコンテナ数を決定する際に第二価格の入札を用いる手法は [15]，[16] を参考にしている．最低予算額は計算方法のみが利用者に公開されており，[15], [16] における第二価格入札と同様に，具体的な値は利用者に対して非公開である．

支払い金額は次点の入札から決定される．例えばオークションに参加し，コンテナを落札した利用者が利用者 A，利用者 B，利用者 C であり，それぞれの入札額が 30, 20, 10 であった場合，利用者 A の支払い金額は次点の入札である利用者 B の入

札額 20，利用者 B の支払い金額は利用者 C の 10 となる．利用者 C の支払い金額は次点の入札が存在しないため，データセンタ管理者が設定するダミーの入札額となる．ダミー入札 d_n はオークション開始前にデータセンタ管理者によってコンテナ数 n ごとに設定され，全利用者に公開される．ダミー入札は利用者が 1 名のみであった場合の最低予算額にも使用されるため，利用者は利用したいコンテナ数に設定されているダミー入札以上の入札をする必要がある．ダミー入札を導入することで利用者の入札額を一定以上にすることができ，データセンタ管理者の収益を確保することが可能となる．

利用できるコンテナ数を決定する最低予算額は第二価格を用いて算出される．利用者は第二価格入札を行った利用者が不明なため，自らの入札額を低下させることで最低予算額を低下させることはできない．また，利用料金も同様に不明である次点の入札額となるため，利用できるコンテナ数と支払い料金のどちらも利用者自らの入札額を調整することで低下させることができない．この仕様により本提案手法は，利用者はより多くのコンテナを利用するためにより大きな入札を行うことが最適となる正直なオークションとなる．

本提案手法について例を用いて詳細に説明する．全コンテナ数 N におけるデータセンタにおいて，データセンタ管理者はサービス開始前にダミー入札 d_n $(n = 1, \ldots, N)$ を設定する．サービス開始後，利用者 $S = 1, 2, \ldots, k$ が逐次到着する．利用者 i はデータセンタにアプリケーションを投入する前に，期限 l_i を申告する．アプリケーションと期限を申告されたデータセンタは，アプリケーションを実行し予測実行時間 p_i と必要コンテナ数 n_i^{need} を計測し，利用者に提示する．予算 B_i を持つ利用者 i は提示された情報を元に入札額 b_i を設定しデータセンタに対し入札する．コンテナ n_i^{need} 個の利用を希望する場合，利用者の入札 b_i はデータセンタ管理者が公開しているダミー入札 $d_{n_i^{need}}$ 以上である必要がある．データセンタ管理者が設定した一定間隔ごとにオークションは実行され，オークションに参加する利用者は前回のオークションに参加し実行が未完了の利用者と，前回のオークション実施後に入札を行った利用者を対象とする．オークションにおいて，全利用者の入札額を比較する．ここでは例として $b_1 \geq b_2 \geq \ldots \geq b_k$ とし，最大の入札を行った利用者 1 から順に割当てを開始する．入札額が同額であった場合は，データセンタに先に到着した利用者がより大きな入札を行ったこととする．

利用者 1 の入札額 b_1 がコンテナ数 n_1^{need} における最低予算額である $m_{n_1^{need}}$ 以上であった場合，希望するコンテナ数である n_1^{need} が割当てられる．最低予算額はオークション開始時に第二価格入札を行った利用者 2 の入札額等を用いて計算される．$b_1 < m_{n_1^{need}}$ の場合，割当てを試みるコンテナ数を $n_1^{need} - 1$ に減らし，b_1 と $m_{n_1^{need}-1}$ を比較する．$b_1 \geq m_{n_1^{need}-1}$ の場合，利用者に $n_1^{need} - 1$ 個のコンテナを割当て，$b_1 < m_{n_1^{need}-1}$ の場合は同様に割当てを試みるコンテナ数を減少させる．コンテナが割当てられた場合，利用者 1 は次点の利用者 2 の入札額である b_2 を利用料金として支払い，アプリケーションの実行を開始する．このようにコンテナ数ごとに入札額と最低予算額を比較していき，利用者 1 が 1 つ以上のコンテナを割当てられるか，最後までコンテナが割当てられなかった場合，次点の入札を行った利用者 2 に対して同様にコンテナの割当てを試みる．コンテナの割当ては全てのコンテナが割当てられるか，全ての利用者の入札を確認するまで繰り返す．

全ての利用者の入札を確認後に未割当てのコンテナが n^{remain} 個存在する場合，すでに 1 個以上のコンテナが割当てられている利用者に対して未割当てコンテナを分配する．利用者 i に分配される追加コンテナの数 n_i^{add} は入札額の比率によって以下の式より決定される．

$$n_i^{add} = n^{remain} \times \frac{b_i}{b_{all}}$$

b_{all} はオークションに参加し，1 個以上のコンテナが割当てられた利用者の入札額合計を表す．既にコンテナ n 個を割当てられていた利用者 i が追加のコンテナを得るためには，入札額 b_i が最低予算額 $m_{n+n_i^{add}}$ 以上である必要がある．追加のコンテナが割当てられることで追加の料金は発生しない．この仕組みにより，利用者は必要コンテナ数以上のコンテナを得る機会が生まれ，より大きな入札を行う動機となる．

第 4 章　評価実験

4.1　シミュレータ上における提案手法の検証

本提案手法をマシンの性能やその他多くの要因を排し，料金設定の影響に限定して検証を行うため，シミュレータ上にて離散事象シミュレーション実験を行なった．本シミュレータは Python3 言語を用いて開発し，利用者の到着とジョブの投入，オークションによるコンテナの割当てと料金の徴収を仮想的に実行する．

シミュレーション開始後，利用者は各オークション間に到着する利用者数の平均を表すパラメータ λ を用いた指数分布に従い到着する．一定間隔 Δt ごとにオークションが実施され，前回のオークションまでに到着していた利用者に対して全体で N 個のコンテナをオークションの結果に従い割当てる．各利用者は予算と期限がランダムで設定されており，1 人につき 1 つのアプリケーションを持つ．アプリケーションにはデータセンタにおける全コンテナを使用した際の予測実行時間がランダムで設定されている．それぞれの設定値を表 4.1 に記す．表 4.1 における random(x, y) は，x から y の範囲で数値をランダムに設定することを意味する．

提案手法を用いたシミュレーションの例を次に示す．オークションは $t = 3, t =$

表 4.1　シミュレーションにおける各パラメータの設定値

記号	設定値	概要
T	48	シミュレーションの全実行時間
Δt	3	オークションの実施間隔
b_i	random(250,350)	利用者 i の入札額
p_i	random(1,4)	利用者 i が持つアプリケーションの予測実行時間
l_i	random(p_i, T)	利用者 i の期限
B_i	$b_i \times p_i$	利用者 i の予算
λ	3	オークション間に到着する利用者数の平均

$6, t = 9$ において実施される．利用者は利用者 A，利用者 B，利用者 C の 3 名であり，それぞれ $t = 0, t = 3, t = 6$ に到着し処理を投入している．

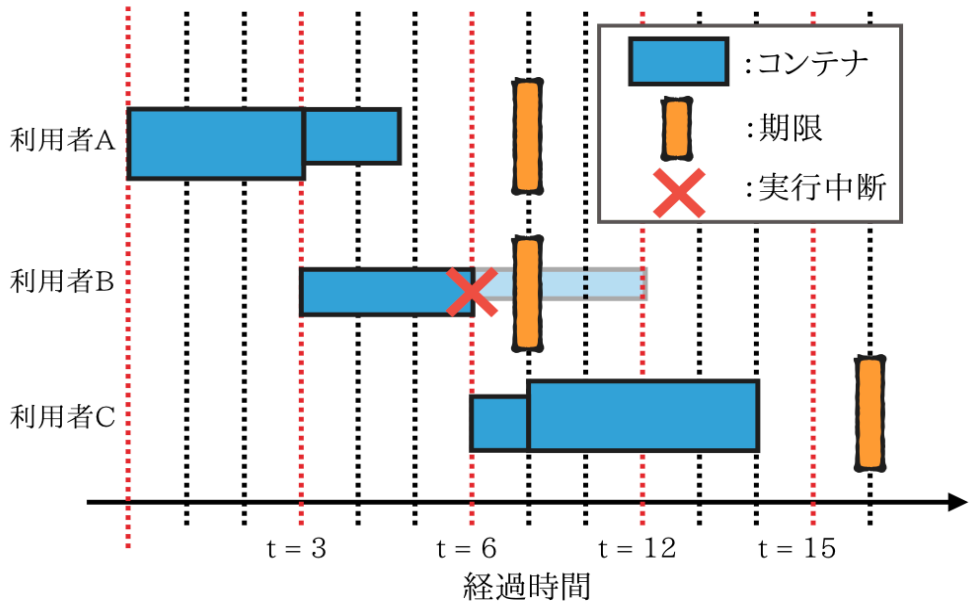

図 4.1　シミュレーションイメージ図

図 4.1 において，利用者 A が $t = 0$ において最初に到着する．他の利用者は存在しないため，ダミー入札を第二価格としてオークションを開始する．オークションの結果，利用者 A はコンテナを獲得し処理を開始する．次に利用者 B が $t = 3$ に到着し，2 回目のオークションが開始される．2 回目のオークションでは利用者 A と利用者 B が到着しており，入札額の小さい利用者 B を第二価格としてオークションが開始される．結果，利用者 A に，より多くのコンテナが割当てられる．2 回目のオークション終了後 $t = 5$ において，利用者 A は処理が完了し離脱する．この場合利用者 A の期限であった $t = 7$ よりも前に処理が完了しているため，利用者 A は期限内に処理を成功したとする．$t = 6$ において利用者 C が到着し，3 回目のオークションに参加する．3 回目のオークションにおける参加者は利用者 B と利用者 C の 2 名であり，入札額の小さい利用者 B を第二価格としてオークションが行われる．3 回目のオークションにおいて利用者 C により多くのコンテナが割当てられる．利用者 B はこの時点で期限内に処理を完了することが不可能となり，期限内の処理の完了は失敗として中断，離脱することになる．

シミュレータを用いて最低予算額を設定する際のパラメータである α の適切な値

を検証した．また，本提案手法と既存の手法である FIFO スケジューラおよび DP スケジューラを，期限内に処理を完了した比率を表す期限内完了率，合計の支払い金額が予算内であった利用者の割合を表す予算内完了率，全利用者が支払った料金の合計である収益合計を用いて比較した．それぞれの結果について次に述べる．

4.1.1 最低予算額算出に用いるパラメータの検証

本提案手法において割当てるコンテナの数は，割当てる対象となる利用者の入札額と，オークションに参加する利用者間における第二価格入札額から算出される最低予算額によって決定される．最低予算額を算出する際には，第二価格入札を行った利用者の予測実行時間等の値の他，データセンタ管理者が設定するパラメータ α が使用される．α の値が小さい場合，コンテナ数の増加に伴う最低予算額の変動が急になり，より多くのコンテナを使用するために必要な入札額が大きくなる．一方，α の値が大きい場合は最低予算額の変動が緩やかになり，少ない入札額であっても多くのコンテナを使用することが可能となる．データセンタ管理者は利用者のコンテナに対する需要や，データセンタにおける計算資源の使用率に合わせてこのパラメータを操作する．図 4.2 に全コンテナ数 20 個のデータセンタにおいて，第二価格を固定した上で α を変更した際のコンテナ数ごとの最低予算額の変動を示す．

α の大小に従い最低予算額が増減することにより，利用者の使用可能なコンテナ数が変化し，期限内完了率や予算内完了率，収益合計に影響することが考えられる．本研究ではシミュレータを用いることで，期限内完了率，予算内完了率，収益合計を α ごとに比較し，適切な値を検証した．以下に全コンテナ数 20，全利用者数 15 名において α を 0.1 から 1.5 まで変化させた際の，各 α における期限内完了率を図 4.3 に，予算内完了率を図 4.4，収益合計を図 4.5 に示す．

図 4.3 および図 4.4 より，パラメータ α を 0.7 まで増加させることで期限内完了率および予算内完了率は上限に到達した．図 4.5 に示すデータセンタ管理者の収益合計はパラメータ $\alpha = 0.5$ において最高額であり，その後 $\alpha = 0.7$ 以降はほぼ一定となった．この結果は，データセンタに到着する利用者の頻度や，投入されるアプリケーションの計算量によって変動するため，常に最適となるわけではないが，

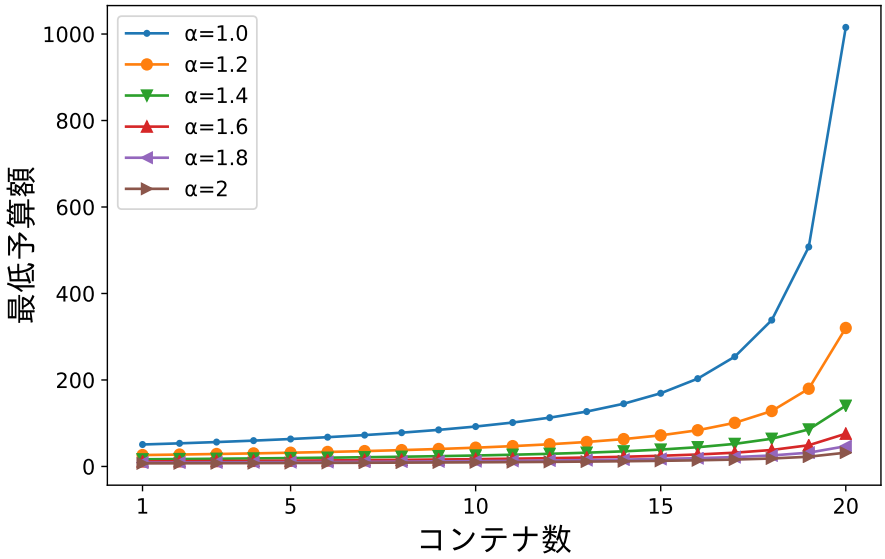

図 4.2　パラメータ α の変化によるコンテナ数ごとの最低予算額

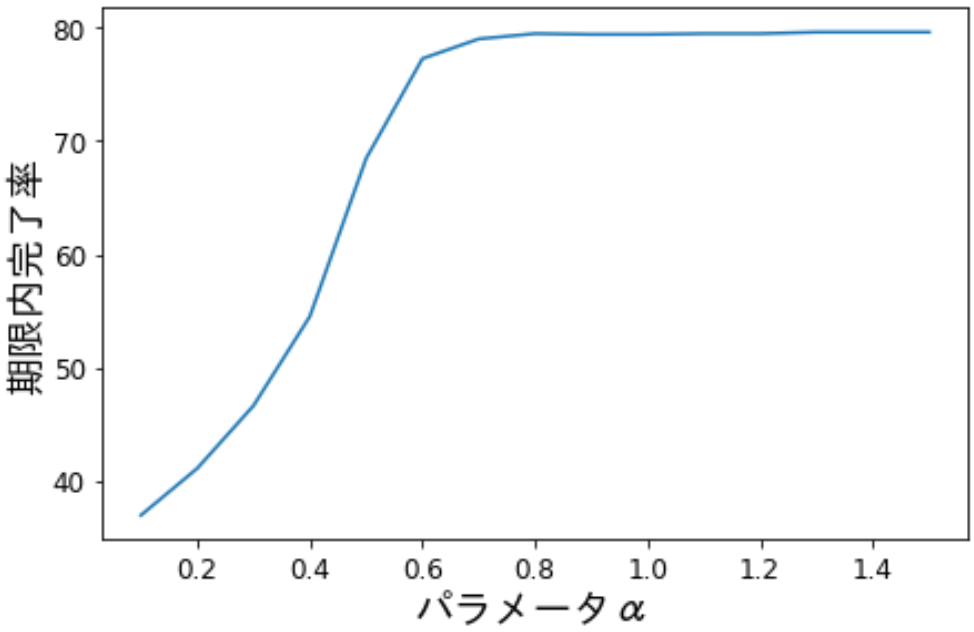

図 4.3　パラメータ α の変化による期限内完了率

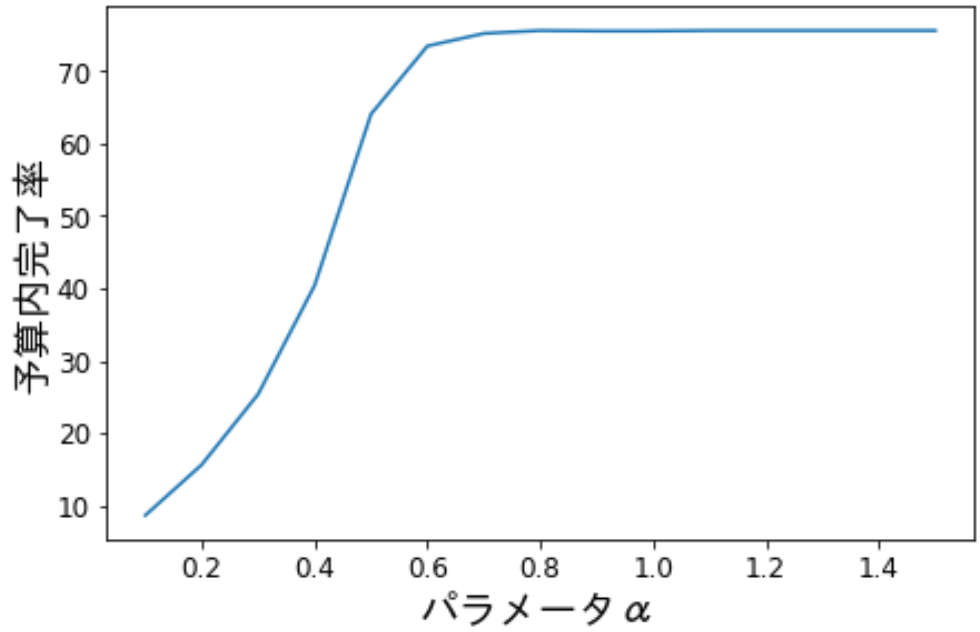

図 4.4　パラメータ α の変化による予算内完了率

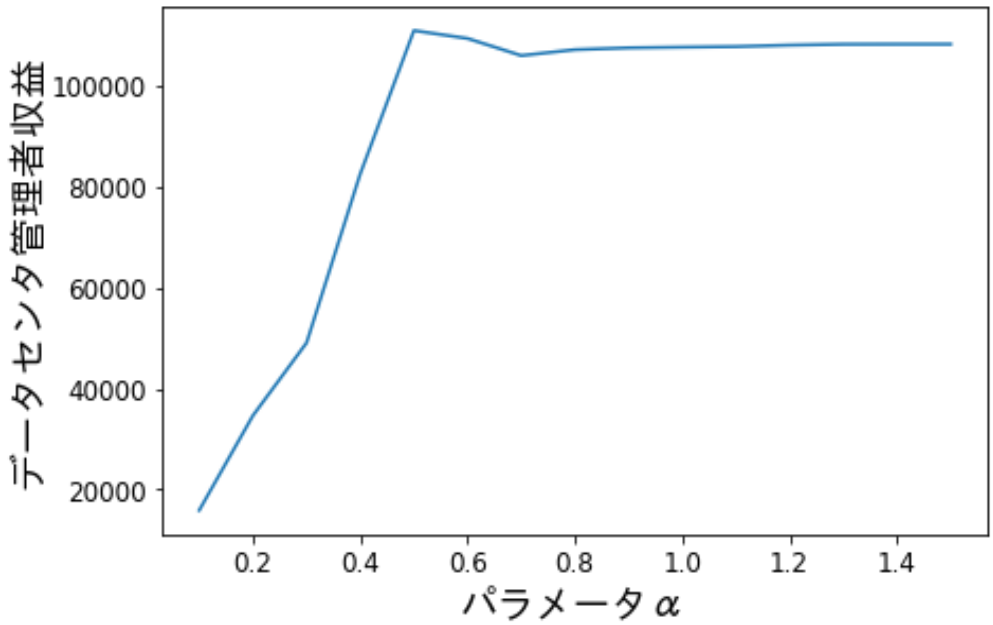

図 4.5　パラメータ α の変化による収益合計

続く既存手法との比較においても $\alpha = 0.7$ とする．

4.1.2 既存手法との比較

評価実験として，全コンテナ数 $N = 20$ 個，全体のシミュレーション時間を 48，オークションの間隔 $\Delta t = 3$ に設定し，利用者数を 1 名から 30 名まで 1 名ずつ変更しシミュレーションを行なった．検証の際には，提案手法の他に FIFO および DP スケジューラ [4] によるシミュレーションを行い，それぞれのスケジューラにおいて期限内完了率，予算内完了率，および収益合計を計測し比較した．

DP スケジューラにおける支出率は提案手法における入札額を全コンテナ数で割った値にしている．これは，提案手法では単位時間あたりに支払う金額を入札していたが，DP スケジューラでは単位時間において利用するコンテナ 1 個あたりの支払い金額を入札しているための対処である．

FIFO では最初に到着した利用者に対して全てのコンテナを割当て，利用料金は定額としている．本シミュレーションでは提案手法における全コンテナ数に対するダミー入札 d_{20} を定価とする．コンテナを割当てられた利用者はアプリケーションが完了するまでコンテナを占有し続け，実行途中で到着した他の利用者は先に到着した利用者が実行を完了するまで待機することとなる．

図 4.6，図 4.7，図 4.8 にそれぞれのスケジューラにおいて利用者数を 1 名から 30 名まで変化した際の期限内完了率，予算内完了率，データセンタ管理者の収益合計を示す．これらの結果は，それぞれの利用者数ごとに利用者の到着タイミングや予算等のパラメータを変化させ，500 回ずつ実行した結果の平均である．

図 4.6 はそれぞれのスケジューラにおける期限内完了率を縦軸に，利用者数を横軸に表示している．図 4.6 において，既存手法である FIFO，DP に比べて提案手法の期限内完了率が劣ることが示された．原因として，提案手法は最低予算額により割当てるコンテナ数に制限があるのに対し，FIFO，DP 共に全てのコンテナを余すことなく割当てていることが考えられる．しかしこの特徴は同時に到着している利用者数が少ない場合であり，利用者数が 15 名以上の場合の期限内完了率は他のスケジューラの値とおおよそ一致する．

図 4.7 は予算内完了率を縦軸に，利用者数を横軸に表示した．利用者数が 1 名の

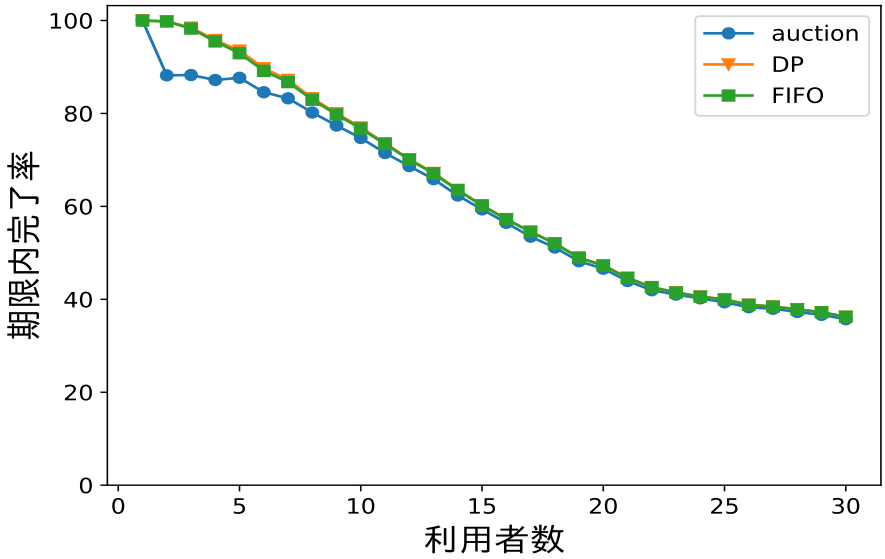

図 4.6　各スケジューラにおける利用者数ごとの期限内完了率

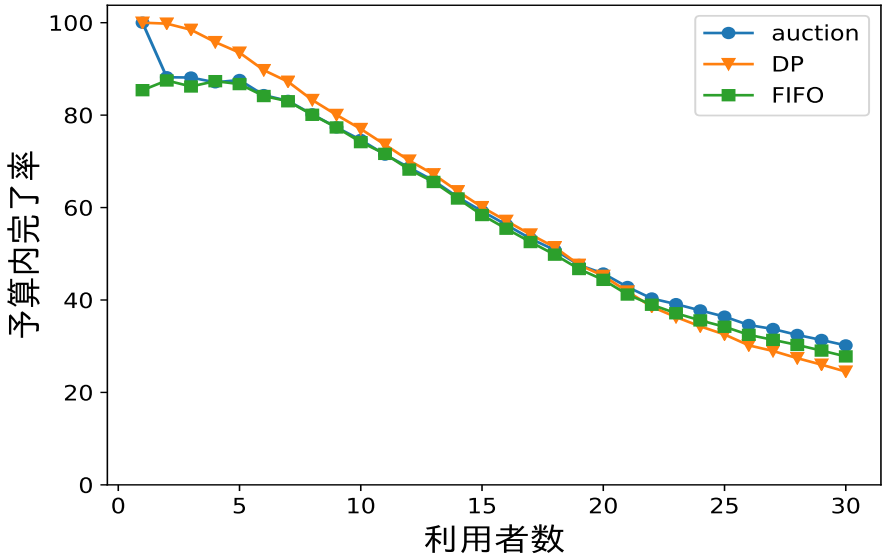

図 4.7　各スケジューラにおける利用者数ごとの予算内完了率

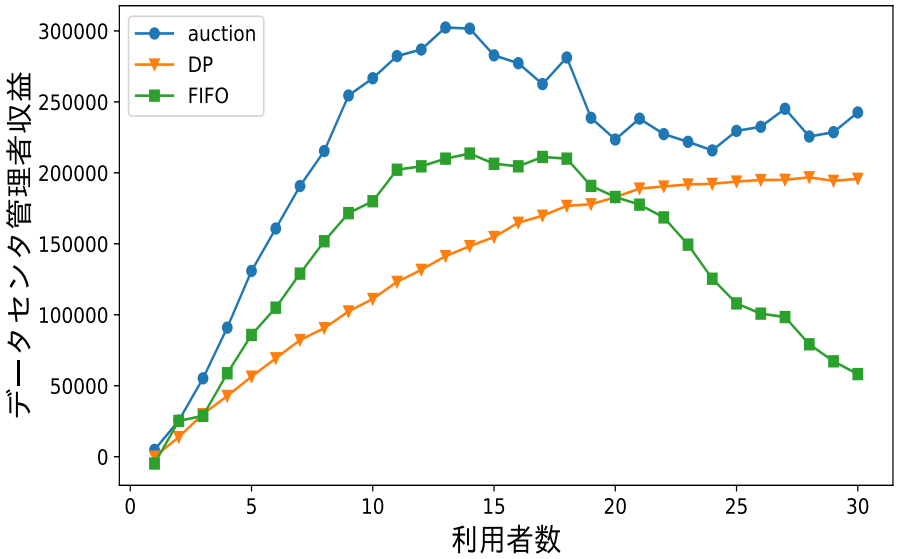

図 4.8　各スケジューラにおける利用者数ごとの収益合計

場合，DP スケジューラが予算内完了率 100% となっている．これは他の利用者が存在しない場合，支出率が最低額まで減少し，利用料金が大幅に低下することが原因である．利用者数が少ない場合は，支出率を減少させることが多くなり，DP スケジューラが最も良い結果を示した．一方で利用者数が増加するにつれ提案手法との差は小さくなっていき，利用者数が 20 名を超える場合は，提案手法は DP スケジューラより高い値を示す．

図 4.8 はスケジューラごとのデータセンタ管理者の収益合計を示す．提案手法は既存手法に比べ収益の合計額が大きいことが示された．DP スケジューラと提案手法の差は利用者数 13 名において最大となり，提案手法における収益は DP スケジューラより 125% 増加した．この結果は図 4.6 と図 4.7 で示した通り，予算および期限内で完了した利用者の割合はほぼ等しい状況であり，利用者の不利益を増加させることなくデータセンタ管理者の利益を増加することができた．

図 4.8 において DP スケジューラのデータセンタ管理者収益が他の手法に比べて小さかったことは，支出率の意図的な低下に因るところが大きく，これは利用者の総数や到着頻度に大きく影響される．DP スケジューラを用いたサービスを提供す

る場合は，料金の設定において改善が必要である．

FIFO は設定した定額によって大きく結果が変動することが確認されており，図 4.7 における予算内完了率はその大小によって増減する．一方で図 4.6 で示される期限内完了率は利用者の到着頻度や実行するアプリケーションの計算量によって大きく増減する．利用者の到着が予測困難な状況において，より多くの利用者が予算内で実行を完了し，かつデータセンタ管理者の収益を増加させることのできる定額を設定することは困難であり，利用者が頻繁に到着するような状況において運用することは難しい．

提案手法は，最低予算額によってコンテナの割当て数に制限をかけることで利用者の入札額の低下を防いだ上で，他の手法と比較し，予算内完了率と期限内完了率の低下を抑えることが可能であることを示した．予算内完了率は利用者数が多い状況では，他 2 種類のスケジューラを上回ることを示しており，本提案手法を大規模データセンタをより複数人に提供する場合に運用することが期待される．

4.2 Hadoop クラスタへの実装

本提案手法を Hadoop クラスタに実装し，複数人へのコンテナの動的な割当ておよび料金の徴収を実現した．実装した環境は，実行時間予測において利用した環境と同一のものである．

本提案手法は実装において，主要な機能を 2 つの要素に分けることができる．1 つ目は利用者の入札額や期限，予算を管理する agent 機能．2 つ目はオークションによりコンテナの割当てと利用料金の徴収を行う auctioneer 機能である．それぞれの詳細について述べる．

agent 機能は利用者によって入札された入札額や予算，期限を XML ファイルに登録する役割と，オークションが実施される際に auctioneer 機能へ渡す役割を持つ．XML ファイルには利用者名，状態，入札額，予算，予測実行時間，期限が登録される．XML ファイルの例をリスト 4.1 示す．

リスト 4.1　XML ファイル例

```
1 <property>
2   <name>hadoop-sub01</name>
3   <state>RUNNING</state>
4   <bid>300</bid>
5   <budget>1500</budget>
6   <predictTime>2</predictTime>
7   <deadline>5</deadline>
8  </property>
```

リスト 4.1 に表示しているのは利用者名 hadoop-sub01 の設定である．3 行目は hadoop-sub01 の状態を示しており，実行中の場合は RUNNING に，アプリケーションの投入を待機している場合や，コンテナが割当てられず実行を中断している場合は STOPPED と表示される．4,5,6,7 行目はそれぞれ入札額，予算，予測実行時間，期限を表している．

auctioneer 機能は Java 言語およびシェルスクリプトによって実装している．シェルスクリプトはマスターのノードにおいて実行する．オークションを実施する間隔はシェルスクリプト内で変更が可能であり，オークションが開始されると設定した間隔毎にコンテナ割当てと料金徴収を行うための Java 言語によって実装されたプログラムが実行される．Java 言語のプログラムは，実行時に agent 機能より XML ファイルを取得し，オークションを実施する．図 4.9 は 300,200,200 の入札を行った利用者 hadoop-sub01, hadoop-sub02, hadoop-sub03 の 3 名が参加した際の割当て結果を Web ページ上に示した図である．

全コンテナ数 40 のクラスタにおいて，hadoop-sub01 にはコンテナ 16 個が割当てられ，残りの 2 名にはコンテナ 12 個がそれぞれ割当てられた．図 4.9 には割当てられたコンテナ数に従い，Hadoop のキュー設定が変更されていることが示されている．

図 4.10 は上記の入札を行った 3 名の利用者が順次到着した際の auctioneer 機能の出力結果である．

最初のオークションが開始された時点では hadoop-sub01 のみが到着しており，

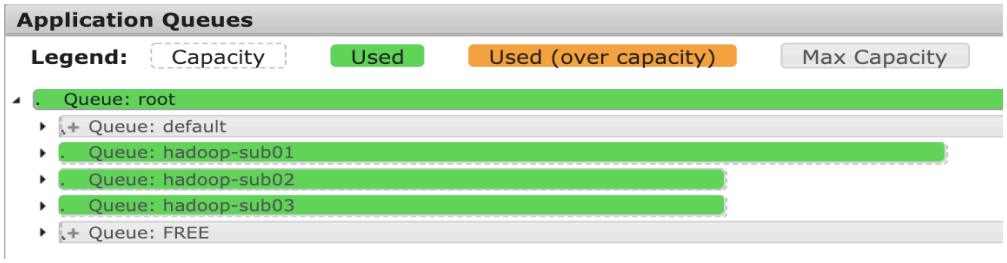

図 4.9　Web ページ上に表示された各利用者のコンテナ割当て状況

```
-----start auction-----
hadoop-sub01 is allocated 35 containers, and pay 100
19/01/30 19:32:04 INFO client.RMProxy: Connecting to ResourceManager
-----end auction-----
-----3m sleep-----
-----start auction-----
hadoop-sub01 is allocated 28 containers, and pay 200
hadoop-sub02 is allocated 12 containers, and pay 100
19/01/30 19:35:10 INFO client.RMProxy: Connecting to ResourceManager
-----end auction-----
-----3m sleep-----
-----start auction-----
hadoop-sub01 is allocated 16 containers, and pay 200
hadoop-sub02 is allocated 12 containers, and pay 200
hadoop-sub03 is allocated 12 containers, and pay 100
19/01/30 19:38:17 INFO client.RMProxy: Connecting to ResourceManager
-----end auction-----
-----3m sleep-----
```

図 4.10　auctioneer 機能によるコンテナ割当てと料金徴収

auctioneer 機能は，hadoop-sub01 にコンテナ 35 個を割当て，ダミー入札額である 100 を徴収している．3 分後，2 回目のオークションでは hadoop-sub02 が新たに到着しており，hadoop-sub01 に 28 個のコンテナを hadoop-sub02 の入札額である 200 で割当て，hadoop-sub02 に 12 個のコンテナをダミー入札の 100 で割当てている．同様に 3 回目のオークションでは，図 4.9 と同様の割当てが行われ，それぞれ料金を徴収している．以上より，各利用者の入札額に従いコンテナを割当て，料金を徴収する仕組みを実装した．

第 5 章　結論

本稿では複数人の同時利用が可能な Hadoop データセンタの構築を目指し，オークションメカニズムを用いた料金設定と計算資源割当て手法を提案した．本提案手法では支払い金額および利用できる計算資源の量は他の利用者の入札額に依存する仕組みを実現している．本提案手法は，この仕組みにより利用者の意図的な入札額低下を防止しており，評価額通りの入札を行うことが最適となる正直なオークションであると言える．

本稿におけるシミュレーション結果において，本提案手法は他の手法から予算内完了率および期限内完了率を低下させることなく，データセンタ管理者の収益を増加したことが示された．また，利用者数の増加に伴う一人当たりの利用可能なコンテナ数が減少による予算内完了率の低下は既存手法と比較し小さく，利用者数が 20 名以上の場合，予算内完了率において他 2 種のスケジューラを上回った．

今後の課題として，シミュレータ上における利用者の到着および離脱，入札額の変更等を発生させ，より動的な検証を行うことが挙げられる．利用者の満足度の増減に伴う Hadoop データセンタ利用者の参加および離脱を再現することで，料金設定とデータセンタ管理者の収益のバランスを明らかにすることが今後の課題である．

謝辞

本研究を取り組むにあたり，最高の環境を用意してくださり，ご指導くださった笠原正治教授に感謝の意を表します．研究会でいただいた数々のご指摘は，研究の方向性を決定する上で大変貴重なものとなりました．心から感謝申し上げます．

ご多忙の中，論文の審査をお引き受けくださった飯田元教授に厚く御礼申し上げます．中間発表の際にいただいたご意見は，研究を進める上で留意すべき点として心に残っております．心から感謝申し上げます．

本研究を進める中で，数々のご指導やご助言をくださった川原純助教に深くお礼申し上げます．知識の乏しい私に，参考となる書物や文献をご紹介くださる等，数々の学ぶ機会を与えてくださり大変感謝しております．大学院の 2 年間で知識や技術を得ることができたのは，川原助教のお力添えによるものだと感じております．ただただ感謝に堪えません．

研究会や研究室特論で多くのご指導，ご助言をくださった笹部昌弘准教授，張元玉助教に心より感謝申し上げます．発表における心構えや英語の知識等，いただいた多くのお言葉は長く心に留めさせていただきます．心よりお礼を申し上げます．

学外発表の際の手続きや，研究室における生活に際して，数多くのご助力をくださった橋本洋子様に心より感謝申し上げます．常日頃より親身に接してくださった事をとても嬉しく思います．ありがとうございました．

大規模システム管理研究室の先輩方，同輩，後輩方の皆様に心より感謝申し上げます．研究に行き詰まった際のアドバイスから，私生活に関する相談まで，大学院での 2 年間が充実したものとなったのは皆様のお心遣いによるものだと考えております．心から感謝申し上げます．

最後に，大学院だけに限らず，これまでの学生生活を通して生活面や金銭面，精神面でも支えてくださった家族に御礼申し上げます．常日頃より気にかけてくださった事，感謝に堪えません．心から感謝申し上げます．

参考文献

[1] Hadoop. The Apach Hadoop project available from <http://hadoop.apache.org/index.html> (accessed 2019–01–31).

[2] J. Dean and S. Ghemawat, “MapReduce: simplified dataprocessing on large clusters,” Communications of the ACM, Volume 51, Issue 1, pp. 107–113, 2008.

[3] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O’ Malley, S. Radia, B. Reed and E. Baldeschwieler. “Apache Hadoop YARN: Yet Another Resource Negotiator,” Proceedings of the 4th annual Symposium on Cloud Computing (SOCC2013), No. 5, 2013.

[4] T. Sandholm and K. Lai, “Dynamic proportional share scheduling in Hadoop,” Proceedings of the 15th international conference on Job scheduling strategies for parallel processing (JSSPP2010), pp. 110–131, 2010.

[5] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker and I. Stoica, “Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,” Proceedings of the 5th European conference on Computer systems, pp. 265–278, 2010.

[6] C. Tian, H. Zhou, Y. He and L. Zha, “A Dynamic MapReduce Scheduler for Heterogeneous Workloads,” Eighth International Conference on Grid and Cooperative Computing, pp. 218–224, 2009.

[7] B. Sharma, T. Wood and C. R. Das, “HybridMR: A Hierarchical MapReduce Scheduler for Hybrid Data Centers,” IEEE 33rd International Conference on Distributed Computing Systems, pp. 102–111, 2013.

[8] H. Mao, S. Hu, Z. Zhang and L. Xiao, “A Load-Driven Task Scheduler with Adaptive DSC for MapReduce,” IEEE/ACM International Conference on Green Computing and Communications, pp. 28–33, 2011.

[9] N. Tiwari, S. Sarkar, U. Bellur and M. Indrawan, “Classification Framework of MapReduce Scheduling Algorithms,” ACM Computing Surveys

(CSUR), Volume 47, Issue 3, pp. 49–88, 2015.

[10] W. Shi, L. Zhang, C. Wu, Z. Li and F. C. M. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," The 2014 ACM international conference on Measurement and modeling of computer systems (SIGMETRICS2014), pp. 71–83, 2014.

[11] Z. Zhou, F. Liu and Z. Li, "When Smart Grid Meets Geo-distributed Cloud: An Auction Approach to Datacenter Demand Response," IEEE Conference on Computer Communications (INFOCOM2015), pp. 2650–2658, 2015.

[12] Amazon Elastic Compute Cloud. available from <https://aws.amazon.com/ec2/> (accessed 2019–01–31).

[13] Amazon EC2 Spot Instances. available from <https://aws.amazon.com/ec2/spot> (accessed 2019–01–31).

[14] M. Yokoo, Y. Sakurai and S. Matsubara, "Robust Combinatorial Auction Protocol against False-name Bids," Artificial Intelligence, Volume 130, Issue 2, pp. 110–116, 2001.

[15] H. R. Varian, "Economic Mechanism Design for Computerized Agents," Proceedings of the First USENIX Workshop on Electronic Commerce, 1995.

[16] W. Vickrey, "Counterspeculation, auctions and competitive sealed tenders," Journal of Finance, Volume 16, Issue1, pp. 8–37, 1961.

[17] H. Izakian, A. Abraham and B. Tork Ladani, "An auction method for resource allocation in computational grids," Future generation computer systems, vol. 26, pp. 228–235, 2010.

[18] P.Anthony and N.R. Jennings, "Developing a Bidding Agent for Multiple Heterogeneous Auctions," ACM Transactions on Internet Technology, pp. 185–217, 2003.

[19] K. Deshmukh, A. V. Goldberg, J. D. Hartline and A. R. Karlin, "Truthful and Competitive Double Auctions," Proceedings of the 10th Annual European Symposium on Algorithms (ESA2002), pp. 361–373, 2002.

発表リスト

[1] 真鍋優，川原純，笠原正治，“Hadoop データセンタの運営における留保価格を用いた料金設定手法の検討,” マルチメディア，分散，協調とモバイル DICOMO2018 シンポジウム, pp. 754–759, 2018.7.
[2] 真鍋優，川原純，笠原正治，“Hadoop データセンタにおけるオークションメカニズムを用いた料金設定と計算資源割当て手法,” 電子情報通信学会 情報通信マネジメント研究会 (ICM2019), 2019.3 (発表予定).