平成29年度 卒業論文

分散版Jungleデータベースの性能測定方法

琉球大学工学部情報工学科

145762E 仲松　栞

指導教員 河野 真治

目 次

図 目 次

ソースコード目次

第1章　はじめに

1.1　研究背景

天気予報やニュース、エンタメや生活に必要なありとあらゆる情報は、インターネット上で手軽に閲覧できるようになり、また、SNS により情報の発信も気軽にできるようになった。その利便性から、パソコンやスマートフォン、タブレット端末等のメディアがますます普及し、Web サービスの利用者は増大する一方で、サーバ側への負荷は増加している。Web サービスの負荷を減らすために、データベースには処理能力の高さ、すなわちスケーラビリティがますます求められてきている。データベースの処理能力を向上させるために、スケールアウトとスケールアップの方法が考えられる。スケールアップとは、ハードウェア的に高性能なマシンを用意することでシステムの処理能力を上げることを指す。スケールアウトとは、汎用的なマシンを複数用意し、処理を分散させることでシステムの処理能力を上げることを指す。単純に処理能力を上げる方法として、スケールアップは有効であるが、高性能のマシンには限界がある。コストはもちろん、そのマシン単体では処理できない程負荷がかかる可能性がある。それに対して、スケールアウトは、処理が重くなるたびに汎用的なマシンを順次追加していくことで性能を上げるため、ハードウェア的に高性能なマシンを用意せずにすみ、また柔軟な対応を取ることができる。よって、データベースの性能を向上させる方法として、このスケールアウトが求められている。本研究で扱うスケーラビリティとはスケールアウトのことを指す。

分散データシステムは、データの整合性 (一貫性)、常にアクセスが可能であること (可用性)、データを分散させやすいかどうか (分割耐性)、この 3 つを同時に保証するこ とは出来ない。これは CAP 定理と呼ばれる。一貫性と可用性を重視しているのが、現在最も使われているデータベースである Relational Database(RDB) である。そのため、データを分割し、複数のノードにデータを分散させることが難しく、結果スケールアウトが困難になってしまうという問題がある。分断耐性を必要とする場合は、NoSQL データベースを選択する。

当研究室では、これらの問題を解決した、煩雑なデータ設計が必要ないスケーラビリティのあるデータベースを目指して、非破壊的木構造データベース Jungle を開発している。Jungle は NoSQL を元に開発されているため、分断耐性を持っている。また、Jungle は、全体の整合性ではなく、木ごとに閉じた局所的な整合性を保証している。 整合性のある木同士をマージすることで新しい整合性のある木をす繰り出すことも 可能であるため、データの伝搬も容易である。Jungle は、これまでの開発によって木構造を格納する機能をもっている。

1.2 研究目的

Jungle は現在、Java と Haskell によりそれぞれの言語で開発されている。本研究で扱うのは Java 版である。

これまでに行われた分散環境上での Jungle の性能を検証する実験では、huskell で書かれた jungle の方が、java で書かれた jungle よりも、読み込みで 3.25 倍, 書き込みで 3.78 倍の性能が確認できた。

huskell は、モダンな型システムを持ち, 型推論と 型安全により、信頼性に重きを置いてプログラミングを行う関数型言語である。対して、Java はコンパイラ型言語であり、構文に関しては C や C#の影響を受けており、プログラムの処理に関しては Huskell よりもパフォーマンスが高い言語であるといえる。よって、java で書かれた jungle が、huskell で書かれた jungle より性能が遅くなってしまった原因として、性能測定時に使用するテストプログラムのフロントエンドに Web サーバー Jetty が使用されていたことが考えられる。

本研究では、新たに Web サーバーを取り除いた測定用プログラムを用いて、純粋な java 版 Jungle の性能を測定する方法を提案する。

第2章　分散版jungleデータベース

Jungle は、スケーラビリティのあるデータベースの開発を目指して当研究室で開発されている分散データベースである。現在 Java と Haskell によりそれぞれの言語で開発されており、本研究で扱うのは Java 版である。本章では、分散データベース Jungle の構造と分散部分について触れる。

2.1　Jungleデータベースの構造

Jungle は、当研究室で開発を行っている木構造の分散データベースで、Java を用いて実装されている。一般的なウェブサイトの構造は大体が木構造であるため、データ構造として木構造を採用している。

Jungle は名前付きの複数の木の集合からなり、木は複数のノードの集合でできている。ノードは自身の子のリストと属性名、属性値を持ち、データベースのレコードに相応する。通常のレコードと異なるのは、ノードに子供となる複数のノードが付くところである。

通常の RDB と異なり、Jungle は木構造をそのまま読み込むことができる。例えば、XML や Json で記述された構造を、データベースを設計することなく読み込むことが可能である。

また、この木を、そのままデータベースとして使用することも可能である。しかし、木の変更の手間は木の構造に依存する。特に非破壊木構造を採用している Jungle では、木構造の変更の手間は O(1) から O(n) となりえる。つまり、アプリケーションに合わせて木を設計しない限り、十分な性能を出すことはできない。逆に、正しい木の設計を行えば高速な処理が可能である。

Jungle はデータの変更を非破壊で行なっており、編集ごとのデータをバージョンとして TreeOperationLog に残している。Jungle の分散ノード間の通信は木の変更の TreeOperationLog を交換することによって、分散データベースを構成するよう設計されている。

2.2　分散機構

Jungle の分散機構は、木構造、すなわちツリー型を想定したネットワークトポロジーを形成し、サーバー同士を接続することで通信を行なっている。ツリー型であれば、データの整合性をとる場合、一度トップまでデータを伝搬させることで行える。トップまたはトップまでの間にあるサーバーノードでデータを運搬中に衝突が発生したら Merge を行い、その結果を改めて伝搬すればよいからである (図 2.1)。

(図 2.1）の矢印の流れを以下に示す。

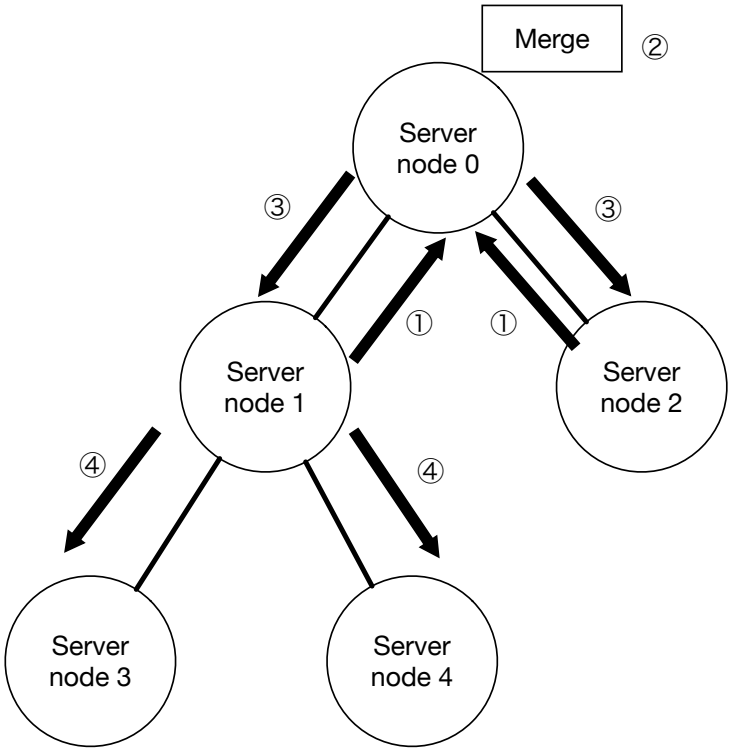

図 2.1: ツリー型のトポロジー

1. servernode 1, servernode 2 からきたデータが servernode 0 で衝突。
2. 衝突したデータの Merge が行われる。
3. Merge されたデータが servernode 1,servernode 2 へ伝搬
4. servernode1 から Merge されたデータが servernode 3、servernode 4 へ伝搬。全体でデータの整合性が取れる。

リング型 (図 2.2 ）やメッシュ型 (図 2.3 ）のトポロジーでは、データの編集結果を他のサーバーノードに流す際に、流したデータが自分自身に返ってくることでループが発生してしまう可能性がある。ツリー型であれば、閉路がない状態でサーバーノード同士を繋げることができる為、編集履歴の結果を他のサーバーノードに流すだけですみ、結果ループを防ぐことができる。

ネットワークトポロジーは、当研究室で開発している並列分散フレームワークである Alice が提供する、TopologyManager という機能を用いて構成されている。

また、データ分散の為には、どのデータをネットワークに流すのか決めなければならない。そこで、TreeOperationLog を使用する。前セクションでも述べたが、TreeOperationLog には、どの Node にどのような操作をしたのかという、データ編集の履歴情報が入っている。この TreeOperationLog を Alice を用いて他のサーバーノードに送り、データの編集をしてもらうことで、同じデータをもつことが可能になる。

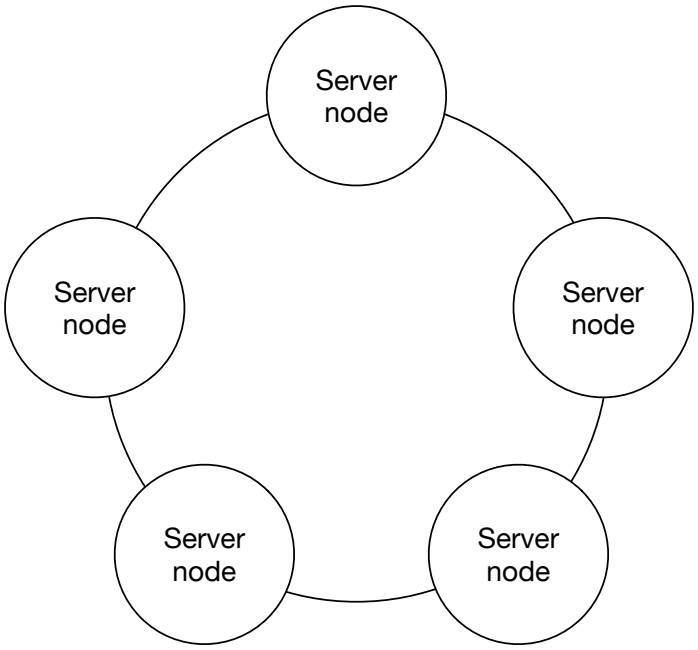

図 2.2: ring 型のトポロジー

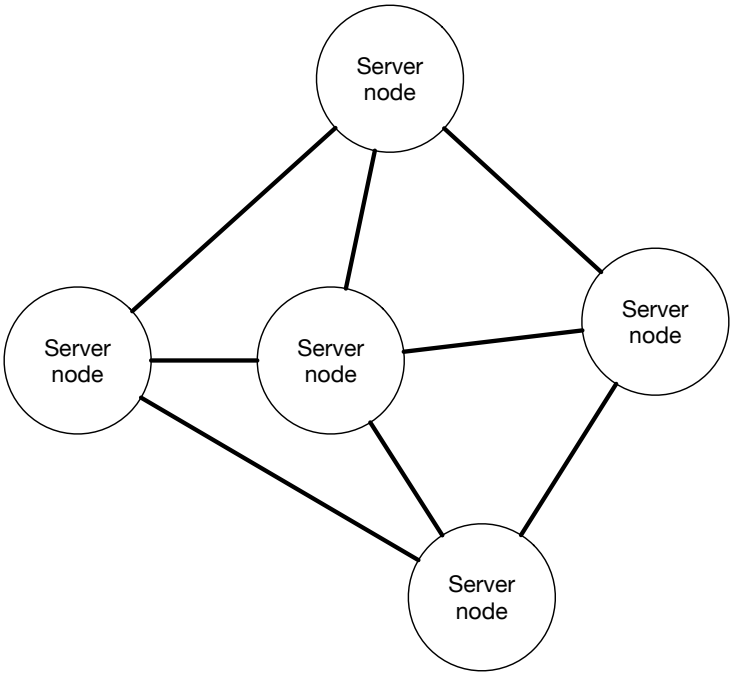

図 2.3: メッシュ型のトポロジー

第3章　評価実験

本研究は、Jungle の分散環境上での性能を正しく評価するための実験を行う。本章では実験の概要について述べる。まず、本研究の目的について述べ、次に、分散フレームワーク Alice による、本研究の分散機構を構成する方法について述べる。次に、木構造上に立ち上げた Jungle へ投入するタスクを制御するジョブスケジューラー、TORQUE について述べる。最後に、本実験の測定用プログラムについて述べる。

3.1　実験目的

Jungle は現在、Java で実装されたものと、Huskell で実装されたものがある。Java 版は、処理速度が早く、よりスケーラビリティの高いデータベースの実装を目的に開発された。対して Huskell 版は、モダンな型システムと、型推論と型安全という特徴を生かし、信頼性の高データベースの実装を目的に開発された。そして、これまでの研究で、Java 版と Haskell 版の Jungle の性能を測定する実験が行われている。性能測定実験では、それぞれ Jetty,Wrap という web サーバーをフロントエンドに用いた Web 掲示板サービスを使用している。Java 版と Huskell 版の Web 掲示板サービスをブレードサーバー上で実行される。計測方法は、掲示板に対して読み込みと書き込みを行い、ネットワークを介して weighhttp で負荷をかける。weighthttp の設定は、 1 スレッドあたり 100 並列のリクエストを、10 スレッド分投入し、合計 100 万のリクエストを処理させる。Java と Haskell の測定結果が表 3.1 のようになった。

測定	Haskell	Java
読み込み	16.31 s	53.13 s
書き込み	20.17 s	76.4 s

表 3.1: Haskell と Java の比較

Huskell 版は、 Java 版と比較して、読み込みで 3,25 倍、書き込みで 3.78 倍 の性能差がでている結果となってしまった。処理速度においては Huskell よりも高いことを予想されていたのにもかかわらず、Java 版が Huskell 版よりも遅くなってしまった原因は、測定時の Web 掲示板サービスのフロントエンドに、どちらも Web サーバーを用いているということが考えられる。しかも、その際は言語の問題から、異なる種類の Web サーバーを使用している。これでは、この性能結果が、異なる言語で実装された Jungle の性能差に

よるものなのか、Web サーバーの性能差によるものなのかがわからない。そこで、本研究では Java 版の Jungle において、Web サーバーを取り除いた、純粋な Jungle の性能を測定するプログラムを実装した。

3.2 実験概要

Jungle を 31 台立ち上げ、ツリー型のトポロジーを構成し、そのうち 16 台の Jungle に 100 回書き込まれたデータが、ルートノードの Jungle へ到達した時間を計測する実験である。(図 3.1)

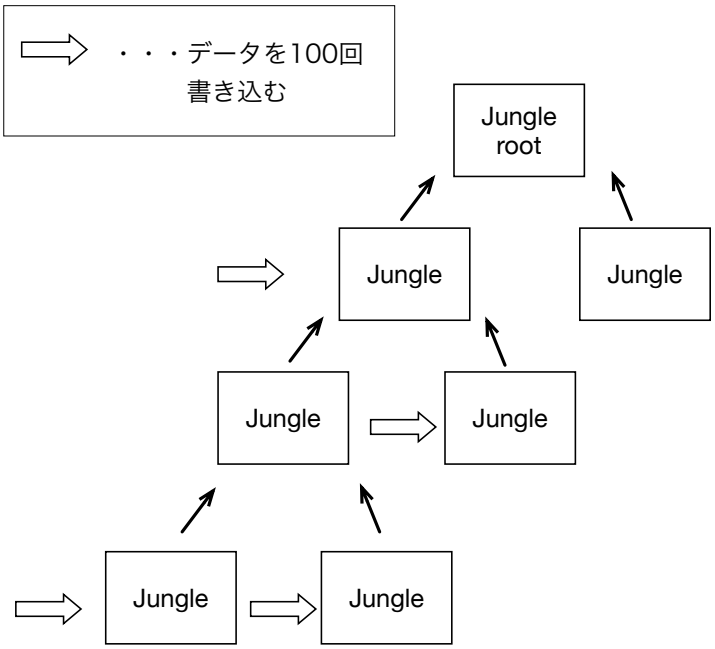

図 3.1: 複数の jungle に書き込まれたデータが root の jungle へ到達する時間を計測する

3.3 実験環境

学科の KVM 上の仮想マシンによる仮想クラスタ環境を用いて実験を行った。分散環境上での実験を行うにあたり、他の利用者とリソースが競合しないよう、TORQUE ジョブスケジューラーを利用している。KVM と仮想マシンの性能はそれぞれ表 3.2、表 3.3 である。

マシン台数	Haskell
CPU	16.31 s
物理コア数	20.17 s
論理コア数	
CPU キャッシュ	
Memory	

表 3.2: KVM の詳細

マシン台数	Haskell
CPU	16.31 s
物理コア数	20.17 s
仮想コア数	
CPU キャッシュ	
Memory	

表 3.3: 仮想クラスタの詳細

3.4 TORQUE Resource Manager

分散環境上での Jungle の性能を測定するにあたり、VM32 台に Jungle,Alice の TopologyManager を起動させた後、Jungle を立ち上げた VM でデータを書き込むプログラムを動作させる。プログラムを起動する順番やタイミングは、TORQUE Resource Manager というジョブスケジューラーによって管理する。

TORQUE Resource Manager は、ジョブを管理・投下・実行する 3 つのデーモンで構成されており、ジョブの管理・投下を担うデーモンが稼働しているヘッダーノードから、ジョブの実行を担うデーモンが稼働している計算ノードへジョブが投下される（図 3.2 ）。

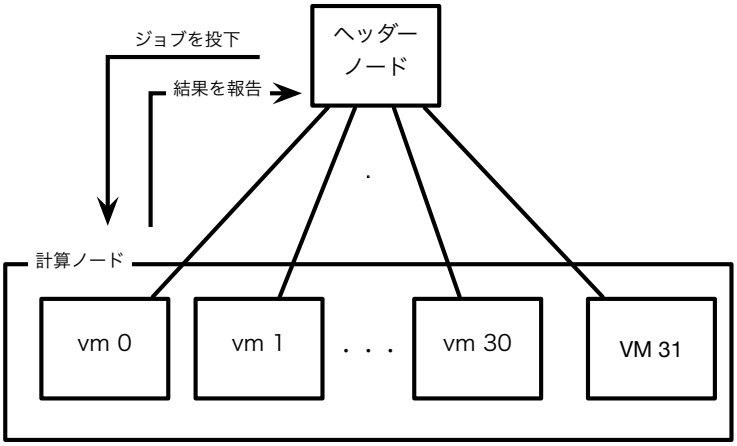

図 3.2: TORQUE の構成

ユーザーはジョブを記述したシェルスクリプトを用意し、スケジューラーに投入する。その際に、利用したいマシン数や CPU コア数を指定する。TORQUE は、ジョブに必要なマシンが揃い次第、受け取ったジョブを実行する。今回、ジョブに投入するためのシェルスクリプトを作成した。以下 (ソースコード 3.1) に示す。

ソースコード 3.1: 本実験で投入するジョブスクリプト

```
1 #!/bin/sh
2 #PBS -q jungle
3 #PBS -N LogUpdateTest
4 #PBS -l nodes=16,walltime=00:08:00
5
6 cd /mnt/data/jungle_workspace/Log
7 /usr/bin/perl /mnt/data/jungle_workspace/scripts/LogupdateTest.pl
```

6 行目で指定されたディレクトリに移動し、7 行目ではそのディレクトリで、指定した別の階層にある perl スクリプトを実行している。

3.5 分散フレームワーク Alice による分散環境の構築

本研究では、分散環境上での Jungle の性能を確認する為、VM32 台分のサーバーノードを用意し、それぞれで Jungle を起動することで、Jungle 間で通信をする環境をつくる。Jungle を起動したサーバーノード間の通信部分を、当研究室で開発している並列分散フレームワーク Alice[1] にて再現する。

Alice には、ネットワークのトポロジーを構成する TopologyManager[2] という機能が備わっている。サーバーノードは TopologyManager に、誰に接続を行えばよいかを尋ねる。TopologyManager は尋ねてきたサーバーノードに順番に、接続先のサーバーノードの IP アドレス、ポート番号、接続名を送り、受け取ったサーバーノードはそれらに従って接続する。この時、TopologyManager 自身は VM0 を用いて立ち上げる。よって、TopologyManager は Jungle をのせた VM1 から VM32、計 VM31 台分のサーバーノードを、木構造を形成するように采配する（図 3.3 ）。

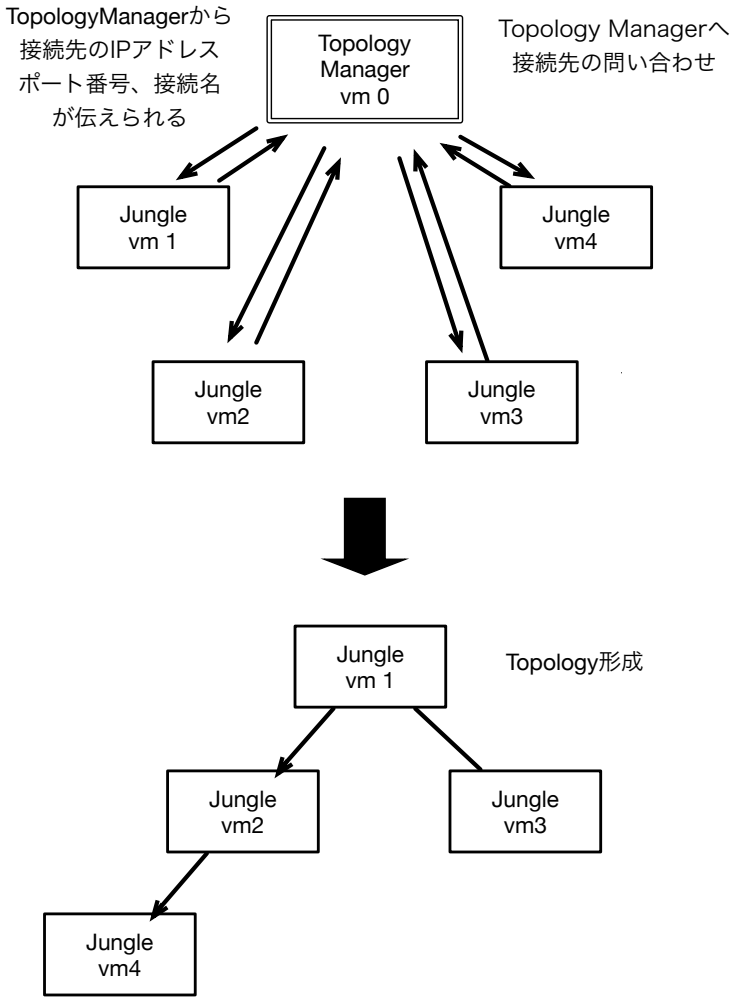

図 3.3: Alice による Jungle の木構造トポロジーの形成

Alice はタスクを行う CodeSegment と、CodeSegment で使用するデータを扱う DataSegment によってプログラムを行うスタイルを取る。CodeSegment は DataSegment が必要

なデータを受け取り次第、タスクを行う。DataSegment がデータを受け取る為には、その DataSegment を示すキーが必要である。

TopologyManager によって構成されたトポロジーのサーバーノードには、それぞれ自分自身を示す文字列であるキーが存在する。このキーは自身のサーバーノードの DataSegment がデータを受け取る際に指定する必要がある。

たとえば、servernode0,servernode1,servernode 2 により、(図 3.4) のように木構造が構成されたとする。

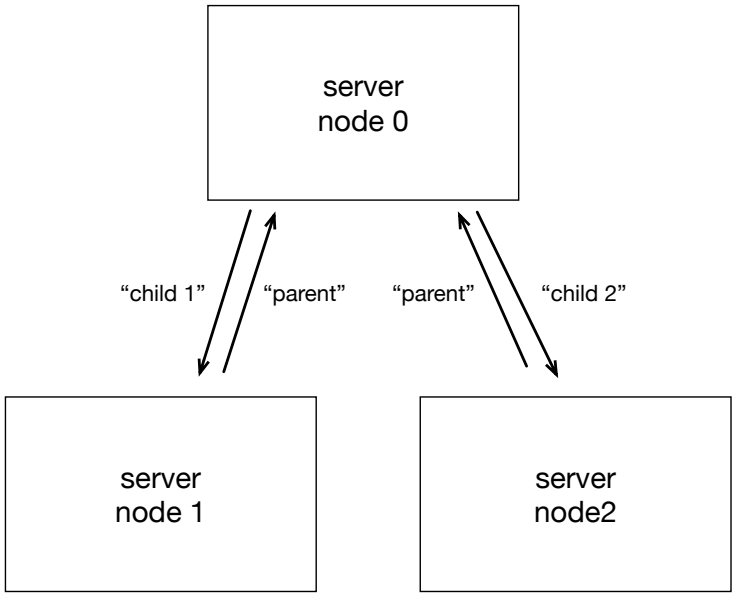

図 3.4: トポロジーの形成

この時、servernode0 は servernode 1 、servernode2 に対して親にあたる。逆に、servernode1,servernode 2 は servernode0 に対して子にあたる。よって、(図 3.4) に矢印の隣にかかれている文字列”parent”,”child 1”,”child 2”のようにキーを指定している。servernode0 から servernode1 へデータを送りたい場合、”child 1”というキーを追加すればいい。このように、データアクセスしたいサーバーノードのキーを追加することで、そのサーバノードの DataSegment へデータアクセスすることができる。他のサーバーノードの DataSegment へデータアクセスする際には、アクセス先のサーバーノードのキーを追加すればいい。

トポロジー構成後、Jungle 間の通信でのデータ形式には TreeOperationLog を利用する。TreeOperationLog は、Jungle によるノードの編集の履歴などの情報が入っている。TreeOperationLog は、Alice の DataSegment でも扱えるようシリアライズ化されたデータである。よって、Alice によって構成されたネットワークトポロジーのサーバノード間でのデータのアクセスが可能になっている。TreeOperationLog を Alice によって他の Jungle へ送る。送信先の Jungle では、送られてきた TreeOperationLog を参照して送信元の Jungle と同じノード編集を行う。こうして、Jungle 間でのデータの同期を可能にしている。

3.6 Jungleの分散性能測定用テストプログラムの実装

本実験において、Jungle の性能を測定する為にテストプログラムを作成した。テストプログラムは、木構造における複数の子ノードに、データを複数書き込む機能を提供する。複数の子ノードにデータをそれぞれ書き込み、最終的に root ノードへデータを merge していく (図 3.5)。データを複数書き込む機能は、Jungle を立ち上げる際に-write オプションと-count オプションをつけることで搭載される。

測定範囲は、複数の子ノードから書き込まれたデータが全て root ノードへ到達し、書き込みが終了するまでの時間である。

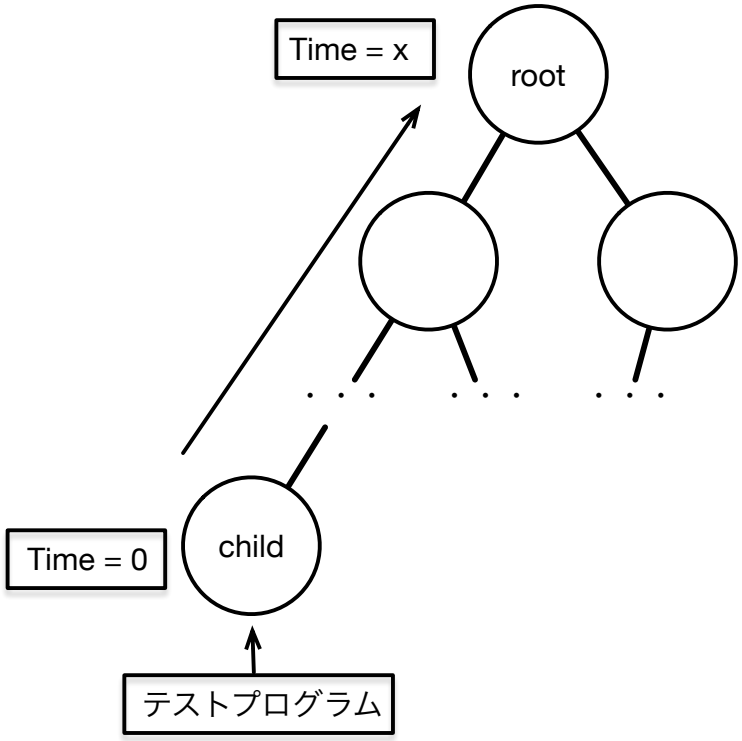

図 3.5: Test プログラムによる Jungle の性能測定

到達時間を測定するためには、Alice の TopologyManager を立ち上げる際に、-show Time オプションをつける必要がある。これにより、出力される結果に末端ノードから root ノードへのデータの到達時間が表示されるようになる。

テストプログラムは、TopologyManager と Jungle の起動を行う。TopologyManager と Jungle は、用意された VM32 台に起動される。それぞれ、VM を何台用いて起動するかは、以下のように指定する。

まず、本実験のネットワークトポロジーを形成するため topokogymanager の起動を行う。TopologyManager は VM0 に起動する。Alice の TopologyManager の起動はソースコード 3.2 のように行う。

ソースコード 3.2: Alice によるネットワークトポロジーマネージャーの起動

```
1 % ssh $nodes[0] \"cd $logFile;java -cp ../../build/libs/logupdateTest
    -1.1.jar alice.topology.manager.TopologyManager -conf ../../scripts/
    tree.dot -p 10000 --showTime --noKeepAlive
```

-p オプションは TopologyManager が開くポートの番号、-conf オプションには dot ファイルのパスを渡している。　ポート番号は Alice のより記述された並列分散プログラムの起動時に渡す必要がある。　dot ファイルには、トポロジーをどのように構成するかが書かれている。dot ファイルを読み込んだ Alice の TopologyManager に対して、サーバーノードは誰に接続を行えばよいかを尋ねる。TopologyManager は尋ねてきたサーバーノードに対してノード番号を割り振り、dot ファイルに記述している通りにサーバーノードが接続を行うように指示をだす。このとき、子ノードからの書き込みが root ノードへ到達したときの時間の計測結果を表示する -showTime オプションも一緒につける。

-write オプションをつけることで、jungle にデータを書き込む機能をつけることができる。これを最大 16 台の Jungle につけて起動させる。また、Jungle がデータを書き込む回数は、-count オプションをつけることで指定できる。今回は、1 から 100 の回数分書き込みを行う。-write オプション、-count オプションを付けた write モードの jungle の起動はソースコード 3.3 のように行う。

ソースコード 3.3: write モードでの Jungle の起動

```
1 % ssh $nodes[$#nodes] \"cd $logFile;java -jar ../../build/libs/
    logupdateTest-1.1.jar -host $nodes[0] -p 10003 -port 10000  -write -
    count 10 --noKeepAlive
```

TopologyManager に 1 台、write モードで立ち上げる Jungle に 16 台使た後、残りの 15 台はそのまま Jungle を起動させる。起動はソースコード 3.4 のように行う。

ソースコード 3.4: Jungle の起動

```
1 % ssh $nodes[$i] \"cd $logFile;java -jar ../../build/libs/logupdateTest
    -1.1.jar -host $nodes[0] -p 10003 -port 10000  --noKeepAlive
```

第4章　性能評価

4.1　java版jungleとhuskell版jungleの比較

4.2　java版jungleの分散性能の評価

4.3　性能測定方法の評価

第5章　結論

5.1　まとめ

5.2　今後の課題

参考文献

[1] 杉本 優 : 分散フレームワーク Alice 上の Meta Computation と応用,

[2] 大城 信康 : 分散 Database Jungle に関する研究,

[3] 金川 竜己 : 非破壊的木構造データベース Jungle とその評価

[4] 大城 信康, 杉本 優, 河野真治 : Data Segment の分散データベースへの応用, 日本ソフトウェア科学会 (2013).

[5] 當間 大千 : 関数型言語 Haskell による並列データベースの実装,

[6] 杉本 優 : 分散ネットフレームワーク Alice による例題の作成

謝辞

本研究を行うにあたり, 日頃より多くの助言, ご指導いただきました河野真治准教授に心より感謝申し上げます。

また, 本実験の測定にあたり, Alice のプログラミングについてご指導くださった照屋のぞみ先輩、伊波 立樹先輩, torque の環境構築に協力してくださった前城健太郎先輩, また、たくさんの温かい励ましをくださった照屋のぞみ先輩、新里幸恵先輩、並列信頼研究室の全てのメンバーに深く感謝いたします。最後に、物心両面で支えてくれた両親に深く感謝いたします。