プログラム融合変換を用いた分散処理環境の開発

1. 背景

Web 業界では Amazon や Google といった企業が自社の持つ巨大な分散フレームワークを自社のサービスを提供するためだけに使用するのではなく WebAPI などを通して積極的に外部に公開することで新たな市場を開拓しようとする潮流が存在する。

この分野では Amazon が最も先行しておりさまざまな自社の分散フレームワークを WebAPI を通して使用するための機能を提供している。今後も分散処理プラットフォームをすでに持っている企業の参入が相次ぐと予想されるが現状では以下のような問題がある。

一般に Web サービスを開発する際に言われる設計の際の原則に MVC の分離があげられる。これはプログラムにおける各機能をデータを保存、取得する Model 部分、Model から取得したデータを加工し View に渡したり、ユーザからのデータを Model が処理できるようにする Controller、ユーザに対しての表現部分を作成する View という 3 つの部分に大きく分けて設計すると柔軟性の高いプログラムになるという設計指針であるが、この設計指針をプログラムから Web 全体に移して考えてみると AmazonS3、AmazonSimpleDB などは Model、AmazonEC2 や GoogleAppEngine は MVC すべての機能を備えたものとしてみることができる。

このようにしてこれらのサービスを捉えた場合例えば AmazonS3、AmazonSimpleDB などの Model 機能のみを提供するサービスを利用しようとすると、自身のサービスのボトルネックが Model にあるならばそれでよいが Controller にある場合はこれらのサービスはスケールに関して役に立たない。

そこで MVC を分離するという設計指針に戻ると Model を提供するサービスはすでに存在するので Controller の機能だけを提供してくれるサービスがあると選択の柔軟性があがると考えられる。つまりデータやプログラムの保存などを強要することなく、ユーザが求める分散処理能力のみを提供するようなサービスである。

Google は MapReduce といわれる分散処理フレームワークを保持しており、今後このフレームワークを外部から利用できる形で提供することが考えられるが以下の課題がある。

1　MapReduce とは Map と Reduce という二つの関数を組み合わせることで大量のデータに対しての高速な処理を実現しているが、これを利用するユーザはこのようなフレームワークに対して十分な知識を保持している必要がある。また十分な知識を保持していたとしても、ユーザが作成したプログラムが本当に最適なのか判断することは難しい。よってもしそのようなサービスが処理時間に対して課金するようなモデルで提供されていた場合はユーザが余分な対価を支払う可能性がある。

2　Web サービスとして提供する場合に Google の MapReduce のアーキテクチャでは単一障害点が存在するために持続的にサービスを提供する必要がある Web サービスで用いるためにはその可用性の点で懸念が残る。

以上の議論から、本提案では Web 上で処理能力のみを提供することを目的としたサービスを構築する際に現れる課題を解決する分散処理フレームワークの開発を目標とする。

2. 目的

本プロジェクトでは開発者である人見が代表を務める(株)アスピレーションが運営する Web サービス MyRemix の機能を拡張するための分散処理フレームワークの開発を目的とする。本プロジェクトの成果を MyRemix に統合することにより、ユーザがサービスを通じて必要な時に必要な分の計算資源を利用するクラウドコンピューティングサービスを提供することが可能になる。

本プロジェクトでは成果物が Web サービスの一部として一般に利用されることを想定しているため

・ユーザが分散処理について特別な知識を有していなくても、最適なプログラムの実行が可能であること

・継続的にサービス可能であるために高可用性を保持していること

の 2 点を開発上の課題と捉え、分散処理に関しては言語仕様に対する工夫と、プログラム融合変換を用いて、高可用性に関しては分散協調システムと呼ぶシステムを開発し、それと分散処理システムを連携させることによりこれらの課題の解決を図った。

3. 開発の内容

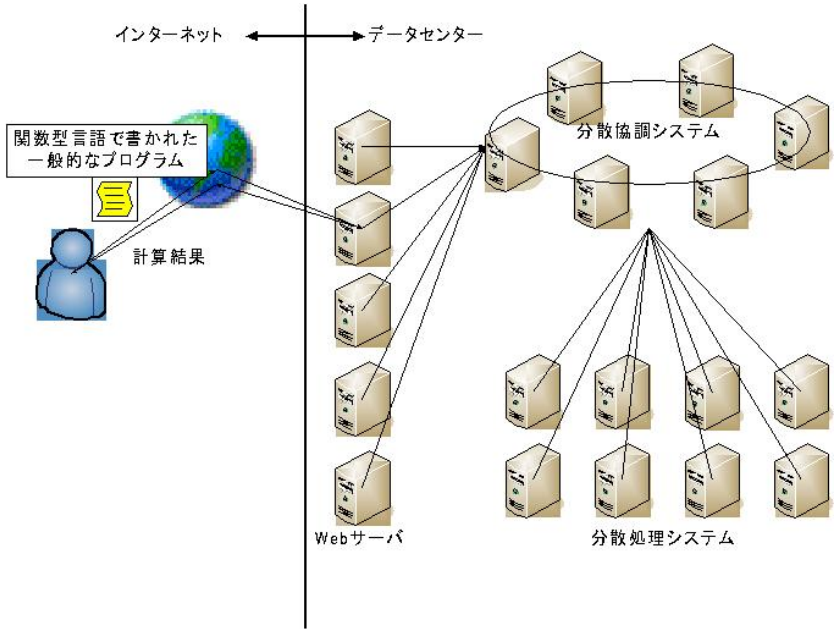

図 1　システム全体図

本プロジェクトでは、上記の 2 つの課題のうち 1 つ目の課題を解決するために関数型の言語で書かれた一般的なプログラムを入力として受け取りそれをプログラム融合変換という手法を用いて、分散処理しやすいような性質を保ったまま関数間で受け渡しされる中間データが最小限になるように自動的に変換し、変換後のプログラムに対して分散処理計画を動的に決定し実行する分散処理システムを開発した。2 つ目の課題を解決するために分散協調システムと呼ぶシステムを開発し、分散協調システムと分散処理システムを連携動作させることで単一障害点の存在しない分散処理システムを実現した。(図 1)

(1)分散処理システム

本システムではユーザが記述するプログラミング言語として関数型言語を選択し、並列計算をしやすい様な仕様を設定することによって、プログラマが、作成したプログラムがどのように分散処理されるのかを意識しな

くてもいいように工夫した。また作成したプログラムが、本システムで実行される際に、最適な形で実行されることを保証するためにプログラム融合変換という手法を用いてプログラムを最適化する。

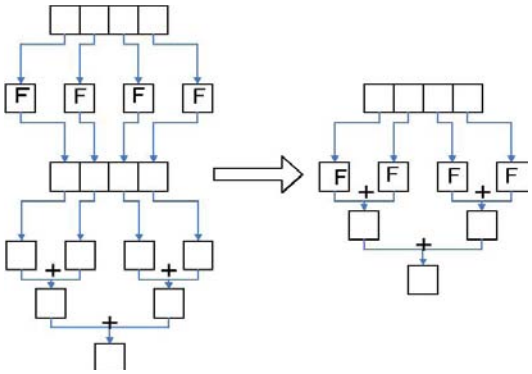

図 2　プログラム融合変換規則 2　reduce(+) . map f = reduce(+, f)

図 2 は map 関数の後に reduce 関数が連続している場合には 1 つの reduce 関数に変換することができるという融合変換の規則を表している。この変換を行うことによって、中間の 1 度データを集めてもう一度ばらすという処理がなくなっており、これによって処理が高速化される。

sum,length,reverse のような関数は map 関数と reduce 関数の組み合わせで表現できることが知られている。そこで内部的に上記のような関数を map 関数と reduce 関数の組み合わせて保持しておくことによって、ユーザが意識することなく、自動的にプログラム融合変換を適用し、最適化することが可能になる。

⑵分散協調システム

単一障害点が存在しない分散処理システムを作成するために、本プロジェクトでは分散協調システムと呼ぶシステムを別に作成し、それと分散処理システムを連携動作させることによって単一障害点をなくすことに成功した。また、分散協調システムに計算の途中の状態を表すメタ情報を保持することによって、計算の途中で特定のサーバがダウンし、計算の継続が不可能になったとしても、計算の途中から計算を再開することが可能である。これによって信頼性の高い分散処理システムを構築することができた。

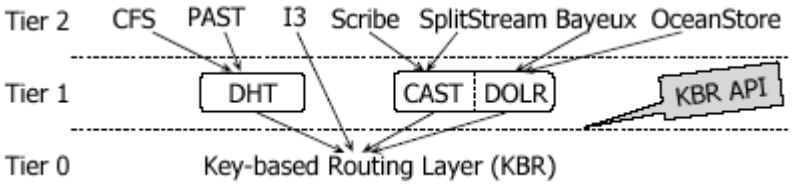

図 3　Key-based routing(KBR)

本プロジェクトでは、図 3 の様に Dabek らの抽象化[1]に従いルーティング層の上に DHT や本プロジェクトで作成する分散協調システム等の高レベルサービス層が乗るという構造を採用している。つまり本プロジェクトで利用する分散協調システムは、様々な分散システムを構築可能なフレームワーク上で動作する 1 つのアプリケーションとして実現されている。

4. 従来の技術(または機能)との相違

一般に分散処理を効率的に行うプログラムを記述することは非常に困難であるが、本プロジェクトで作成し

た分散処理システムでは言語仕様を工夫しさらにプログラム融合変換を適用し最適化することによって、ユーザは特に分散処理について何も意識することなしに、システムの上で最も効率のよい分散処理を行うことが可能である。

また Google の MapReduce など既存の分散処理システムは Master と呼ばれるサーバが単一障害点になっており、そのサーバがダウンすると、1 から計算をやり直す必要がある。本プロジェクトで作成した分散処理システムは、分散協調システムと連携して動作させることにより、単一障害点と呼ばれるサーバがなくなっており、たとえ特定のサーバがダウンしたとしても、キャッシュされた計算結果を用いることによって、計算を途中から再開することが可能になっている。これによって Web サービスとしてユーザに提供するという想定に必要な信頼性を得ることができた。

5. 期待される効果

本システムが下位のアーキテクチャを完全に抽象化することによって、コンピュータ科学の分野の研究者やプログラマに限らず非コンピュータ科学の分野の研究者やプログラマ以外の人にも簡単に利用可能な HPC(High Performance Computing)環境の提供が可能になる。

6. 普及(または活用)の見通し

大量の計算資源を必要としている物理シミュレーション・フーリエ変換・音声処理・デジタル画像処理・ビデオフォーマット変換処理・レイトレーシング・分子動力学法・流体計算・気候シミュレーション・データベース処理・格子ボルツマン法・暗号解読・市況分析・天体シミュレーションなどの分野における研究やサービスのバックエンドシステムとして広く利用されることを期待している。

7. 開発者名(所属)

人見　琢也
株式会社アスピレーション代表取締役
東京大学大学院情報理工学系研究科創造情報学専攻竹内研究室修士

(参考)

MyRemix: http://myremix.jp/

[1] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. Towards a common API for structured peer-to-peer overlays. In Proc. IPTPS'03, February 2003.