氏　　名　Yongzhe Zhang

学位(専攻分野)　博士(情報学)

学 位 記 番 号　総研大甲第 2188 号

学位授与の日付　2020 年 9 月 28日

学位授与の要件　複合科学研究科　情報学専攻
学位規則第6条第1項該当

学 位 論 文 題 目　User-friendly and Efficient Distributed Graph Processing

論 文 審 査 委 員　主　査　教授　合田　憲人

教授　高須　淳宏

助教　加藤　弘之

助教　対馬　かなえ

教授　胡　振江
北京大学

(Form 3)

Summary of Doctoral Thesis

Name in full Yongzhe Zhang

Title User-friendly and Efficient Distributed Graph Processing

Nowadays, graphs play a very important role in representing the data in our daily life, such as the world wide web, users and their relations in a social network, geography or biological information. With the increasing demand of analyzing large-scale graphs generated by the modern applications, lots of research has been invested in distributed graph systems, a technique that can make use of the memory space of a bunch of computation devices, in order to make large-scale graph processing practical and efficient. However, analyzing graphs on distributed-memory is known to be challenging, and the difficulties come from many aspects of a graph analytic task including the high complexity in distributed graph algorithms, the limited parallelism in many graph computations, and all kinds of practical issues like the fault tolerance, load balancing, cache effect and so on.

In the past decade, great efforts have been devoted to making large-scale graph processing practical, and the key is the computational model that bridges the algorithmic thinking and the parallel execution. The mainstream graph analytics systems can be classified into two classes based on their computation model, the systems using the vertex-centric paradigm and the systems using the linear algebra abstraction, both trying to provide users with user-friendly programming interface and high efficiency. The most popular vertex-centric paradigm is based on the Bulk-synchronous (BSP) model with message passing. In this model, to implement a graph algorithm, A user-specified vertex-program is executed on every vertex, and the whole graph computation is an iterative procedure consisting of synchronous rounds, each of which performs the vertex-centric computation and the synchronized communication phase between the vertices. Then, the linear algebra approach is also drawing attention. It uses standardized matrix and vector operations to describe graph computation and can be implemented on many kinds of hardware architectures. Such clear separation of algorithm design and implementation makes this technique more portable on various frameworks.

The rapid development of the graph processing frameworks in both computational models not only shows new ideas for improving large-scale graph analytics, but also reveals the limitations of the existing solutions. A crucial problem at present is that, with various sophisticated techniques proposed to improve graph analytics frameworks, it is becoming more difficult to have the user-friendliness and efficiency at the same time.

In the vertex-centric paradigm, a big problem is that the programming interface is not user-friendly. For example, the Shiloach-Vishkin (SV) algorithm for solving the connected component problem is very complicated due to the need of frequent state transition and low-level message passing mechanism. Even though there are domain-specific languages (DSLs) proposed for simplifying the programming in the vertex-centric model, the main issue is that not all commonly used graph algorithms can be implemented. Essentially, these DSLs do not support general remote data access, reading or writing attributes of other vertices through references, making them less expressive in practice. In addition to the programming interface, there are also many potential issues in the vertex-centric model that may hurt the performance. Lots of research has proposed effective optimizations for dealing with these problems, but these optimizations do not compose, which means for some complex algorithms requiring multiple optimizations they can only choose which problem to solve but cannot enjoy all the optimizations.

For the linear algebra approach, it also has severe limitations in both user-friendliness and efficiency on distributed-memory. First of all, a graph computation may have different implementations using the linear algebra approach, and for ordinary users it is not easy to know which one runs faster on a particular architecture. A typical example is the connected component problem which can be solved by the traditional label propagation algorithm (a.k.a. parallel BFS), ParConnect, LACC, FastSV and so on. Then, in addition to the different choices of algorithms, even the same algorithm can be implemented in different ways in GraphBLAS API due to the redundancies in the standard. For example, GraphBLAS distinguishes the matrix-vector multiplication with the vector-matrix multiplication despite their similarity in functionality, and in some libraries their performance can differ a lot depending on which graph format is used. Therefore, although developing a linear algebra graph algorithm is not difficult, tuning a program can be very tricky. The linear algebra approach also has limitations in dealing with performance issues. Unlike the vertex-centric paradigm that the system designer can use any optimization that works for the graph computation, in linear algebra, many of those optimizations are not easy to apply. The reason is that, in the vertex-centric graph frameworks, many kinds of external information are used to trigger the optimizations (e.g. the number of active vertices, the vertices' degree distribution and so on), but in linear algebra such information is not captured in the semantics of the APIs in the GraphBLAS standard. In practice, none of the linear algebra libraries on distributed-memory are GraphBLAS compliant in order to use more powerful optimizations for some graph algorithms.

In this thesis give a comprehensive overview of the existing graph analytics systems, analyze the difficulties in achieving both user-friendliness and efficiency, and propose our solution for achieving both goals. Our main contribution is a vertex-centric graph analytics framework using our domain-specific language (DSL) as the programming

interface with a powerful back end for efficient graph analytics. Our framework is built on three key techniques, a more expressive high-level DSL to ease vertex-centric programming by hiding the message passing from users, an efficient vertex-centric back end that can arbitrarily combine various optimizations in the same vertex-program, and a novel compilation technique from our DSL to the back end as well as the cost-based compilation technique to choose the best optimizations for a graph algorithm. The resultant framework achieved comparable performance to the state-of-the-art vertex-centric system. We also made efforts to designing user-friendly and efficient graph analytics systems in the language of linear algebra. We currently focus on the linear algebra graph algorithms and their efficient implementation, and our results include a new connected component algorithm FastSV and Boruvka's minimum spanning forest algorithm, both of which outperform the state-of-the-art distributed algorithm significantly with our algorithm specific optimizations. In the future, we are interested in compilation techniques to make the detection of optimizations viable under the standardized linear algebra graph APIs.

Form8・Separate Sheet
（様式８・別紙１）

Results of the doctoral thesis screening

博士論文審査結果

Name in Full
氏 名　Yongzhe Zhang

Title
論文題目　User-friendly and Efficient Distributed Graph Processing

本論文は、大規模グラフを効率的に処理するための並列プログラミングモデルに関するものである。大規模グラフの処理に対して、代表的な並列プログラミングモデルは、頂点主体のグラフ並列計算モデルと線形代数に基づく並列計算モデルの二つある。しかし、これらのモデルは「使いやすさ」と「効率性」の面において大きな問題がある。頂点主体のグラフ並列計算モデルは、データ通信の記述が複雑で効率的な実装が難しい。一方、線形代数に基づく並列計算モデルは、一般性はあるものの、領域特定の性質を利用した効率的な実装手法は少ない。本論文は、大規模グラフ処理を記述するための特定領域言語の設計と新しい最適化手法の提案による問題の解決方法を与えた。本研究の主要貢献は (1) リモートアクセスを自然に記述できる高水準グラフ並列プログラミング言語 Palgol の設計と実現、(2) channel によるデータ通信を最適化する手法の提案と実現、(3) 重要な線形代数演算の効率的な実装とそれによるグラフ処理アルゴリズムの効率化、ということである。

本論文は、英語で記述されており、全 6 章から構成されている。

第 1 章は序論である。研究の背景、研究目的、既存研究の問題点、本研究の主要な貢献、論文全体の構成を述べている。

第 2 章は基礎知識の紹介である。分散グラフ処理の歴史と基礎知識、頂点主体のグラフ並列計算モデル、線形代数に基づく並列計算モデル、関連研究について議論している。

第 3 章では、頂点主体のグラフ並列計算モデル Pregel に対して、channel を導入することによるデータ通信を最適化する手法を提案するとともに、その実現方法、実験、評価を示している。

第 4 章では、第 3 章の効率的な Pregel をもとに、リモートアクセスを自然に記述できる高水準グラフ並列プログラミング言語 Palgol を設計し、Palgol プログラムから SQL を経由し効率のよい Pregel プログラムを生成する手法を示している。また、その有効性を具体的な実用例を通じて議論している。

第 5 章では、強連結成分分解と最小全域木問題という 2 つのグラフアルゴリズムに対して、線形代数による簡潔な記述から効率的な実装を導く手法を与え、いずれも既存の実装よりもはるかに改善されることとなった。

第 6 章は論文のまとめと今後の課題である。

審査会において、出願者は大規模グラフ処理を実現する並列プログラミングフレームワークの構築について、40 分のスライド発表と 40 分の質疑応答によって行われた。スライドの発表では、代表的なプログラミングモデルとして、頂点主体のグラフ並列計算モデルと線形代数に基づく並列計算モデルの研究背景、「使いやすさ」と「効率性」の側面からみた問題点、大規模グラフ処理を記述するための特定領域言語の設計と新しい最適化手法の

提案による問題の解決などが要領よく説明された。そのあと審査委員との質疑応答を行い、的確な回答がなされた。

以上のように、本論文では、大規模グラグ向けの並列プログラミングフレームワークの構築について、理論だけでなく実践的な観点における本研究の貢献が大きい。なお、本研究の成果は1件のトップ国際会議論文（IPDPS 2019）、2 件の国際会議論文（PP 2020, APLAS 2017)、 3件の共著論文（2件のジャーナル論文と1件の国際会議論文）という業績をあげている。また、開発したシステムもウェブで公開し自由にダウンロードできるようになっている。以上の理由により、審査委員会は、本論文が学位の授与に値すると判断した。