**北京航空航天大学计算机学院**

**学位论文开题报告**

**论文题目**：基于消息量预测的并行图计算系统性能优化研究

**专 业**：计算机技术

**研究方向**：分布式图计算系统

**研 究 生**：李东泽

**学 号**：ZY1706212

**指导教师**：樊文飞

**北京航空航天大学计算机学院**

2018年11月18日

目 录

1 论文选题的背景与意义 1

2 国内外研究现状及发展动态 2

2.1 同步计算模型 2

2.2 异步计算模型 3

2.3 混合计算模型 4

2.4 自适应计算模型 5

3 论文的研究内容及拟采取的技术方案 5

3.1 自适应计算模型的定义 5

3.2 课题目标与主要研究内容 6

3.3 拟采用的回归模型 8

4 关键技术与难点 8

5 论文研究计划 9

5.1 时间计划 10

5.2 预期效果 10

6 主要参考文献 10

**基于消息量预测的并行图计算系统性能研究**

# 论文选题的背景与意义

图计算是以“图论”为基础的对现实世界“图”结构的抽象表达。在实际的应用中，存在着许多图计算问题，如图的连通性[1]、单源最短路径[2]、网页排序[3]等，同样在社会或科学领域中的许多问题都可以通过转换成图计算模型从而得到解决，例如: Google公司每天都需要对数亿级别的网页进行排序[4]; 生物医学上需要对蛋白质进行子图匹配，分析蛋白质间的相互作用从而开发出更有效的临床药物[5]。因此，对图数据的分析和计算具有很重要的现实意义。

早期的图计算数据规模较小，大部分使用BGL等单机图处理算法[6]，极大的限制了图计算问题的规模。后来随着图规模的增长，出现了Parallel BGL[7]等图计算系统，但是系统本身不具备良好的稳定性。近些年来，随着大数据时代的来临，大图数据的分析给图计算带来了极大的挑战。于此同时，MapReduce[8]作为一个并行的计算框架，及其开源实现Hadoop[9]为海量数据的处理提供了便利，但是由于图计算过程中通常涉及多轮迭代，因此该框架不能高效的进行并行图计算。

针对上述图计算特点与图处理中所面临的问题，面向大规模图计算的研究吸引了越来越多的研究者投身其中。最近，在计算模型和运行优化方面涌现出不少具有前瞻性的图计算系统，代表性的图系统包括Giraph[10]、GPS[11]、GRACE[12]、GraphLab[13]、PowerSwitch[14]、GRAPE+[15]。它们在计算模型上分别属于同步、异步、混合以及自适应计算模型，它们围绕着减少迭代次数加快算法收敛、减少消息传递数目、实现负载均衡这3个优化目标，分别采用一种或多种优化技术对自身进行优化，但是它们往往都是在优化某个目标的同时牺牲了其它性能。

因此本课题以自适应并行图计算为框架，采用机器学习模型，针对图计算过程中接收的消息量进行预测，最终合理的动态调节各计算节点间的相对进度，从而达到性能优化。

# 国内外研究现状及发展动态

## 2.1 同步计算模型

1990年，图领奖获得者Valiant提出了BSP (bulk synchronous parallel)计算模型[16]，这种模型区分于现有的MapReduce计算框架在于其适合做数据的迭代计算。其核心思想是将每一次迭代称为一个超步(superstep)，整个图计算划分为多个超步，并行迭代直到全部顶点收敛或达到预设的最大迭代次数。在每一次的超步计算中，又会分为本地计算、消息传递、全局同步三个阶段，图2.1展示了同步模型的整体计算流程，其中，每轮计算中所有顶点只能使用上一轮迭代计算的更新消息，而计算得到的最新消息也只能通过同步在下一轮迭代中使用。因为全局同步的存在，使得整个图计算调度十分简单，同时，每轮更新的消息会在全局同步下批量传送，极大限度的利用了网络流量。



图2.1 同步计算模型

为高效的处理大规模图数据的计算问题，在上述BSP计算模型的启发下，Google首次提出了以点为中心的计算框架Pregel[17]。在该计算模型中，用户只需编写一个与具体应用紧密相关的计算函数，而不需要考虑数据的处理过程。在函数内，涉及处理上轮迭代的更新消息、本地计算、消息传递等逻辑。Pregel框架从容错性、一致性等方面给出了可行的解决方案，整个计算模型简单、易用、可拓展性强。同样采用上述BSP模型的分布式图计算框架还有GPS[11]、Giraph[10]以及GRAPE[21]等系统。

虽然BSP模型极大简化了图计算任务，充分利用了网络流量，但是该模型仍有以下几方面限制：1) 具有木桶效应：因为全局同步的存在，每轮迭代的计算时间取决于最慢的计算节点，严重影响图计算效率，除此之外，每轮迭代只能使用上一轮迭代的更新信息，导致迭代次数较多，收敛速度较慢；2) 某些需要协调图中邻接点的算法无法在BSP模型下收敛，例如图着色问题[19]，因为同步操作的过程中可能会出现这两个节点颜色不断翻转的情况，即本轮迭代具有相同颜色的邻接顶点在下轮的迭代仍会选取相同的颜色；3) 消息量大，导致通信代价昂贵。在分布式的图处理框架中，通常需要将原始图数据划分为多个子图并分配到不同的计算节点，导致计算节点间需要要大量的消息传递，然而，在实际的图数据中，尤其是社交网络等图应用数据，图中顶点的度数服从幂律分布，从而使得通信的代价成为影响系统性能的瓶颈之一。

## 2.2 异步计算模型

为解决上述BSP同步计算模型的诸多问题，科研工作者提出了异步计算模型AP(Asynchronous Parallel)，如图2.2所示。



图2.2 异步计算模型

异步模型中，相邻两次迭代没有明显的界限，任何计算节点随时接收、发送消息。因为异步模型摒弃了全局同步的操作，使得每个计算节点不用关心其它节点的进度和状态，可随时进行下轮计算，因此解决了同步模型下“木桶效应”问题。经大量的理论分析和实验表明，异步模型在执行效率及资源的利用率方面相对于同步模型具有较大的优势，尤其是在图处理系统负载不均衡时，这种优势体现的更加明显[12]，但是异步模型也同样存在着以下方面的不足：1) 虽然异步模型可以加速图计算的收敛，但由于没有全局同步的限制，导致迭代轮数的增多, 造成了大量的冗余计算；2) 异步模型在分布式图系统的设计方面相对于同步模型来说更加复杂，需考虑数据的一致性问题，因为异步模型下的任何计算节点随时接收来自其它计算节点的消息，因此必须设计一套额外的机制来保证相同数据对象可以被不同的进程和线程互斥访问，例如GraphLab[22]采用分布式锁的形式来保证数据的一致性，而GRAPE+[15]采用了聚合函数来保证数据的一致性；3) 异步模型的计算过程与计算结果具有不确定性，如果算法设计不合理，迭代计算可能无法达到预期效果，甚至不能收敛，即使能够收敛，也会在迭代计算的最终收敛前造成大量的冗余计算。不仅如此，对于采用异步模型的分布式图计算系统来说，编程难度有所增加，调试十分困难[25]。

## 2.3 混合计算模型

最近的研究表明[14]，同步模型与异步模型在图的算法、图数据的规模以及集群规模等不同方面具有不同的优势，同样针对某一具体算法，不同的图计算阶段也需要不同的计算模型，因此为了解决上述问题，提出了Hybrid Model计算模型[14]，其通过一组启发式的算法建立代价收益模型来动态预测同步模型和异步模型两种调度方式的性能，并实现了在计算过程中对两种计算模型自由的切换，如图2.3所示。



图2.3 混合计算模型

[14]的实验结果表明，混合计算模型可以准确预测同步与异步的性能，并完成调度模式的切换。其相对于使用单一计算模模型，在PageRank算法、单源最短路径算法、图着色算法等算法在执行效率上得到了不同程度的提高。但这种计算模型在每轮计算后，需要重新预测下一轮的计算模型，且需要解决模型间切换的数据一致性问题，因此产生了额外的内存与时间开销。

## 2.4 自适应计算模型

Adaptive Asynchronous Parallel (AAP)[15]计算模型，也叫自适应计算模型。该模型摒弃了全局同步的操作，因此属于异步模型。但于上述模型不同的是，自适应计算模型中每个计算节点通过保留一些参数来测量本节点相对于其它计算节点的进度，并根据当前已接收的消息量决定是否立即进入下轮计算或等待一段时间后再进入下轮计算，上述过程只在本计算节点下完成，不用关心其它任何节点的执行状态。AAP模型不同与其它计算模型主要体现在以下几个方面：1) **同步模型**：AAP模型通过异步消息传递机制解决了同步模型下“木桶效应”问题；2) **异步模型**：AAP模型通过引入限定值使得各个计算节点自己决定每轮迭代计算前需等待的时间，从而聚集更多的消息进行更有效的计算，因此很大程度上减少了异步模型下的冗余计算；3) **混合模型**：相对于混合模型，AAP模型不需要为同步与异步之间的切换设置全局的预测开关，相反，AAP模型允许每个计算节点自己决定计算模式，因此在图算法的执行期间，某些计算节点采用同步模式，与此同时，其它节点采用异步模式。除此之外，由于AAP模型动态调整计算节点间的相对进程，因此减少了模型间切换所带来的额外开销。

# 论文的研究内容及拟采取的技术方案

上述介绍的4种计算模型，其中同步模型存在“木桶效应”问题；异步模型存在大量的冗余计算；混合计算模型为实现同步与异步间自由切换产生了额外的内存与预测开销；而自适应模型中，每个计算节点通过引入限定值动态调整节点间相对进度，从而解决了同步、异步、混合模型下的问题，但是目前自适应模型在实现上还没有给出一个合理的方案，因此，本课题以自适应计算模型为基础，从计算过程中消息量预测角度出发，给出该模型下一个可行且实现方案，并进一步研究该模型下的性能优化问题。

## 3.1 自适应计算模型的定义

自适应并行图计算系统中，为解决同步模型下的“木桶效应”及异步模型下的冗余计算问题，每个计算节点引入了限定值，之后每轮迭代计算开始前，各计算节点需判断是否等待长的时间以积累更多的消息，其中基于下述函数进行动态调整：

上述函数中的各参数描述如下，表示某一计算节点：

(1) ：表示计算节点当前接收的消息量。直观上来看，如果 越大，则当前计算节点接收消息越多，应尽快开始下轮迭代计算。

(2) ：表示本轮迭代计算开始前期望接收的消息量。该变量定义如下：，其中可由用户自定义初始化，为了尽快减少冗余计算；为消息到达速率；是 的一部分，后者为运行时间。

(3) ：表示计算节点为接收到期望的消息量而等待的时间，定义为；表示计算节点前一轮计算后的空闲时间，以防止无限期的等待。

经过上述分析，在自适应并行图计算系统中，对于任何一个计算节点，只需预测下一轮迭代计算的运行时间及消息到达速率，便可计算出延迟计算的等待时间，以保证计算开始前积累更充分的消息，进行更有效的计算从而减少迭代轮数，加快收敛速度。

## 3.2 课题目标与主要研究内容

**本课题目标**：本课题采用基于机器学习技术对期望接收的消息量进行更有效预测，从而优化图数据的并行计算。

**本课题主要研究内容**：首先，如上述分析得出，针对消息量的预测可转化成对运行时间及消息到达速率的预测；其次，我们对与的预测定义成机器学习中的回归问题，我们拟采用均方相对误差作为评价指标；然后，我们选取机器学习中适合做数值型预测的回归模型作为候选，进行比较，从而分析各模型在图计算预测问题上的优缺点；最后我们选取出一种或多种模型并针对多种不同类型的图算法进行训练与预测，并嵌入进自适应图计算系统。

**本课题具体研究内容如下**：

(1) **问题定义**：首先我们将运行时间及消息到达速率 的预测定义成回归问题，我们以的预测为例：对于某一计算节点及给定的运行时信息，我们需要训练出一个预测模型，用于评估下一轮迭代的运行时间，因为该预测模型的输出是以毫秒为单位的实数，因此该问题可以定义成机器学习中的回归问题，具体方程如下：

因此预测过程分为以下两个部分：1) 收集运行时信息，相对于机器学习中的特征提取。一个理想的信息应足以描述下一轮程序的运行状态；2) 针对岭回归、局部加权线性回归、神经网络以及随机森林等众多回归模型进行分析、设计并训练出一个适合的预测器。需要特别指出，由于估值阶段在图计算过程中仅执行一次，因此我们只针对接下来的迭代增量阶段进行运行时间与消息到达速率的预测。

(2) **图算法分类**：我们拟选取单源最短路径、图连通性、模式匹配、广度优先搜索、协同过滤(矩阵分解)、pagerank、标签传播、最小生成树、三角计数等图应用算法，针对这些图算法，我们从图计算预测的角度进行分类，针对不同类别的图算法，提取不同的特征，并采用不用的训练策略，最终给出不同种类图算法的预测思路及预测模型。

(3) **模型训练**：针对某一特定的数据集与图应用算法，首先，我们a) 从运行结束的图计算日志中收集数据；b) 筛选清洗数据，进行数据分析；c) 选取多种回归模型进行训练，求解回归系数；d) 进行数据测试、分析模型效果；e) 选择测试效果最好模型作为该算法在该数据集上的预测模型。其次，我们根据分类后的图算法，分析各预测模型的共同点，找到通用的回归模型；最后，将最终选取的模型嵌入并行图计算系统。

(4) **并行图计算系统实现**：根据已选定的一种或多种预测模型，开发并实现出基于该模型预测的自适应并行图计算系统，并提供前端界面展示功能。

## 3.3 拟采用的回归模型

**局部加权线性回归模型**：从历史数据分析得出，线性回归模型不适用于针对图计算中运行时间及消息到达速率的预测，相反，往往采用一条类似于二次函数的曲线可对数据拟合的更好，而线性模型很容易出现欠拟合现象，不能取得最好的预测效果，但是局部加权线性回归模型允许给待预测点附近的每个点赋予一定的权重，从而在估计中引入一些偏差来降低预测的均方误差，最终解决了在非线性模型中建立线性模型的问题，因此比较适合图计算中运行时间与消息速率的预测。

**岭回归模型**：由于岭回归模型的对于简单问题的性能优势、并可以处理特征数多于样本数的情况，因此它特别适合图计算中运行时间与消息速率的预测。不仅如此，岭回归模型还有如下优势：(1) 在岭回归模型的预测过程中最多只需要几十次积累操作，因此其满足图计算的时间性能要求；(2) 虽然其计算复杂度为，但其只依赖于特征向量的维度，因此对于图计算预测中特征向量维度或样本数据较小的情形下具有一定优势。

**随机森林模型**：在随机森林模型中，包含了多个回归树，其中每个回归树构建时的样本都是由训练集经过有放回抽样得来的，并进行独立的预测。最终预测结果由各回归树投票决定。随机森岭模型不仅在高维度离散的数据数据下表现出很好的性能，其在以下方面同样特别适合图计算中运行时间与消息速率的预测：(1) 对于一个包含个回归树的随机森林，其中所有回归树的平均深度是，最好与最坏情况下，其预测的复杂度分别是是和，因此其满足图计算得时间性能要求；(2) [23]实验证明，大多数随机森林模型在训练复杂度上更趋近于最好情况而不是最坏情况，因此这为图计算的线上训练提供了可能性；(3) 由于最终的预测取决于所有回归树的投票，因此对于训练数据的轻微变动具有很好的适应性，这为图计算预测的正确定提供了保证；(4) 该模型只有一个超参数，即回归树的个数，因此该模型针对不同类别的图算法可能更具有通用型。

**其它模型**：除上述介绍的三个模型外，还有一些模型适合数值型变量的预测，包括逐步线性回归模型、神经网络、高斯过程回归等。这些模型针对训练数据的大小、特征向量的维度以及超参数的选择等不同方面具有不同的优势，这些模型均列入拟采用的技术方案中。

# 关键技术与难点

通过上文的介绍与分析，本课题的主要难点包括：

(1) **图应用算法的分类**：目前图计算领域的重要算法包括图遍历(深度优先、广度优先)、单源最短路径、连通性、子图匹配、最小生成树、pagerank、协同过滤、标签传播等。本课题从分布式图计算角度出发，会对上述算法从query、消息传递、本地计算以及图数据特性等不同方向分析，并将图算法进行分类，给出每一类的预测模型或模型的参数调节方向。

(2) **收集运行时信息**：收集运行时信息的过程及机器学习领域特征提取的过程，特征提取结果的好坏直接影响接下来模型的训练与预测，图计算中的特征提取涉及多方面因素，包括图结构本身，如图的规模、平均度数、各子图边界点个数；运行时信息，如消息的接收数量、接收消息中不同计算节点的个数等不同特征。

(3) **图计算约束**：除了机器学习领域中过拟合与预测的准确性外，图计算领域还关心额外几个指标：a) 预测时间约束，在迭代预测过程中，我们需保证预测时间足够短并尽可能减少预测开销，如果一次预测占用了迭代计算的大部分时间，那么此次预测没有任何意义；b) 训练数据约束，某些图应用算法中，可能无法采取线下训练的方式，必须依靠迭代的线上时间完成训练预测等工作，这需要我们的模型在小训练数据集上有较好的准确性；c) 超参数约束，虽然我们针对众多图应用算法采用不同的训练预测思路，但我们更希望针对多样化的数据输入给出一个通用的计算模型。

(4) **模型的选取**：虽然我们可以将运行时间的预测定义成机器学习中的回归问题，但如上节所述，存在众多的回归模型需要我们进行分析，判断出是否适合运行时间的预测，并针对一种或多种模型进行合理的训练与预测。

(5) **图计算系统实现**：最终，我们需要将训练好的模型嵌入分布式图计算系统中，嵌入过程主要包括运行时的特征提取以及模型预测，因此整个过程涉及分布式图计算系统的图存储模块、消息传递模块以及预测模型的开发。

# 论文研究计划

## 5.1 时间计划

2018年12月 – 2019年3月：在已有的分布式图计算系统上完成众多图应用算法的开发，并将图算法进行的分类。

2019年4月 – 2019年6月：测试、分析并选出适合运行时间与消息到达速率预测的回归模型，并针对每类图算法，进行模型的训练与预测，撰写小论文。

2019年7月 – 2019年9月：针对不同种类的图算法，完成模型的训练与预测，开发基于机器学习预测的并行图计算系统，并与同步图计算系统系统、异步图计算模型进行实验对比，迭代调优。

2019年10月 – 2020年1月：整理资料，撰写毕业论文。

## 5.2 预期效果

最终，开发出基于机器学习预测的自适应并行图计算系统，针对不同种类的图算法与目前学术界和工业界主流的的并行图系统进行实验对比，预计时间性能快出至少15%。

# 主要参考文献

[1] <https://en.wikipedia.org/wiki/Connected_component_(graph_theory)>

[2] Frigioni D, Marchetti-Spaccamela A, Nanni U. Fully dynamic output bounded single source

shortest path problem[C]// Acm-Siam Symposium on Discrete Algorithms. Society for

Industrial and Applied Mathematics, 1996:212-221.

[3] Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine[J]. Computer

Networks, 2012, 56(18):3825-3833.

[4] Google. How search works. <http://www.google.com/insidesearch/howsearchworks/thetory>/

[5] Pržulj N. Protein-protein interactions: making sense of networks via graph-theoretic

modeling.[J]. Bioessays News & Reviews in Molecular Cellular & Developmental Biology,

2011, 33(2):115-123.

[6] Siek J G, Lee L Q, Lumsdaine A. The boost graph library: user guide and reference manual[M].

Addison-Wesley Longman Publishing Co. Inc. 2002.

[7] Gregor D, Lumsdaine A. Lifting sequential graph algorithms for distributed-memory parallel

computation[C]// Acm Sigplan Conference on Object-oriented Programming. ACM, 2005:423

437.

[8] Dean J , Ghemawat S . MapReduce: simplified data processing on large clusters[M]. ACM, 2008.

[9] Apache Software Foundation. Apache Hadoop [EB/OL]. <https://hadoop.apache.org/> , 2013

[10] Apache Giraph. <http://giraph.apache.org/>

[11] Salihoglu S , Widom J . GPS: a graph processing system[J]. 2013.

[12] Spielman D A, Teng S H. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems[C]// 2004:81-90.

[13] Bickson D. GraphLab: Asynchronous Graph Computation in the Clouds and Beyond[J]. Archives of Biochemistry & Biophysics, 2006, 452(2):138-148.

[14] Xie C, Chen R, Guan H, et al. SYNC or ASYNC: time to fuse for distributed graph-parallel computation[C]// Acm Sigplan Symposium on Principles & Practice of Parallel Programming. ACM, 2015:194-204.

[15] Fan, Wenfei, Xu,LuPing, XiaoJian, Luo, et al. Adaptive Asynchronous Parallelization of Graph Algorithms[J]. Proceedings of the SigMod, 2018, 6(10): 1141-1156

[16] Valiant L G . A bridging model for parallel computation[J]. Comm Acm, 1990, 33(8):103-111.

[17] Malewicz G, Austern M H, Bik A J C, et al. Pregel:a system for large-scale graph processing[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2010:135-146.

[18] Slota G M , Rajamanickam S , Devine K , et al. Partitioning Trillion-edge Graphs in Minutes[C]// Parallel & Distributed Processing Symposium. IEEE, 2017.

[19] Hendrickson B, Leland R. A multilevel algorithm for partitioning graphs[C]// Supercomputing, 1995. Proceedings of the IEEE/ACM SC95 Conference. IEEE, 1995:28.

[20] Spielman D A, Teng S H. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems[C]// 2004:81-90.

[21] Fan, Wenfei, Xu, Jingbo, Wu, Yinghui, et al. GRAPE: parallelizing sequential graph computations[J]. Proceedings of the VLDB Endowment, 2016, 10(12):1889-1892.

[22] Bickson D. GraphLab: Asynchronous Graph Computation in the Clouds and Beyond[J]. Archives of Biochemistry & Biophysics, 2006, 452(2):138-148.

[23] Hutter F, Xu L, Hoos H H, et al. Algorithm runtime prediction: Methods & evaluation[J]. Artificial Intelligence, 2014, 206(206):79-111.