

一种高效的分布式爬虫系统负载均衡策略

张树涛^{1,2}, 谭海波¹, 陈良锋¹, 吕 波¹

(1. 中国科学院合肥物质科学研究院, 合肥 230039; 2. 中国科学技术大学 研究生院, 合肥 230039)

摘 要: 传统分布式爬虫系统负载均衡方法仅考虑少量的负载影响因素, 未对各爬虫节点负载情况进行全面有效的评估, 使得任务量的分配不合理。针对该问题, 提出一种面向分布式爬虫系统的高效负载均衡策略。分析影响爬虫节点运行时间的因素, 采用 BP 神经网络构建基于多影响因素的非线性分布式爬虫节点运行时间模型。以该模型预测的各子节点运行时间的最小方差为负载均衡策略的目标函数, 并利用带约束条件的改进粒子群优化算法求解目标函数, 确定负载均衡的任务分配方案。实验结果表明, 该负载均衡策略在满足爬虫节点高性能要求的前提下, 能有效缩短分布式爬虫系统的运行时间。

关键词: 分布式爬虫; 负载均衡; 预测模型; 粒子群优化算法; 约束条件

开放科学(资源服务)标志码(OSID):



中文引用格式: 张树涛, 谭海波, 陈良锋, 等. 一种高效的分布式爬虫系统负载均衡策略[J]. 计算机工程, 2019, 45(11): 62-67.

英文引用格式: ZHANG Shutao, TAN Haibo, CHEN Liangfeng, et al. An efficient load balance strategy for distributed crawler system[J]. Computer Engineering, 2019, 45(11): 62-67.

An Efficient Load Balance Strategy for Distributed Crawler System

ZHANG Shutao^{1,2}, TAN Haibo¹, CHEN Liangfeng¹, LÜ Bo¹

(1. Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230039, China;

2. Graduate School, University of Science and Technology of China, Hefei 230039, China)

[Abstract] Traditional load balance methods for distributed crawlers fail in providing comprehensively efficient evaluation of crawler node loads, as they consider only a small number of affecting factors in load. Thus the tasks are not reasonably assigned. To address the problem, this paper proposes an efficient load balance strategy for distributed crawlers. The strategy analyzes affecting factors in the running time of crawler nodes, and uses BP neural network to construct a non-linear running time model based on multiple affecting factors for distributed crawler nodes. The model predicts the running time of each sub-node, and the minimum variance of the running time is taken as the target function of load balance strategies. The target function is resolved by using improved particle swarm optimization algorithm with constraints to form a task assignment scheme with balanced loads. Experimental results show that the load balance strategy can efficiently reduce the running time of distributed crawlers while meeting the high performance requirements of crawler nodes.

[Key words] distributed crawler; load balance; prediction model; particle swarm optimization algorithm; constraint condition

DOI: 10.19678/j.issn.1000-3428.0053439

0 概述

随着大数据时代的到来, 网络数据体量不断扩大。IDC 最新预测, 到 2025 年, 全球数据规模将增

长至 163 ZB, 接近于 2016 年所产生数据量的 10 倍^[1]。高效的网络数据采集是数据挖掘的关键, 是使数据发挥价值的重要环节。网络爬虫因快速实现互联网数据的采集及结构化存储^[2], 已成为主流

基金项目: 安徽省科技重大专项“基于大数据的中小微企业精准智力服务平台”(711245801052)。

作者简介: 张树涛(1995—), 男, 硕士研究生, 主研方向为数据挖掘、网络安全; 谭海波, 研究员、博士; 陈良锋, 工程师、博士; 吕 波, 工程师、硕士。

收稿日期: 2018-12-19

修回日期: 2019-01-23

E-mail: shutaozh@mail.ustc.edu.cn

的数据采集手段,其技术也得以迅速发展。传统集中式网络爬虫因无法满足大规模数据采集需求,构建高效的分布式爬虫系统已成为爬虫技术的应用热点^[3-4]。

在分布式爬虫系统中,因爬虫节点性能各异,任务量分配不合理,会造成节点负载不均衡,出现大量超载节点和空闲节点,严重影响系统整体的运行效率^[5-7]。为解决该问题,有学者提出基于分布式哈希表的平均分配策略,例如:文献[8]提出一种基于一致性哈希的均分负载空间算法进行任务调度;文献[9]将哈希函数映射到不同节点的位置,可使所有爬虫节点分配的地址空间达到动态均衡;文献[10]提出构造环形序列取代传统哈希散列函数的节点地址空间映射方法。该类方法均未考虑爬虫节点间的性能差异,即无论爬虫节点的性能情况如何,基于分布式哈希表算法均会平均地分配任务,在爬虫节点性能差异较大时,无法达到较好的负载均衡^[11-13]。另外一些学者提出综合爬虫节点性能参数的动态负载均衡方法,例如:文献[14]改进了一致性哈希算法,根据集群中每个服务器的性能差异,为每个服务器设置不同的虚拟节点数量,不再依赖于单个默认值;文献[15]提出加权轮询算法的分布式爬虫系统任务调度策略,根据爬虫节点运行情况调整权重值;文献[16]提出基于站点规模预测的分布式爬虫动态负载均衡方法,可根据爬虫节点自身的性能和预测出的站点规模计算爬虫所承载的任务负荷量。该类方法只考虑了少量的负载均衡影响因素,未对各爬虫节点负载情况实现更全面、有效的评估,使得任务量的分配不够合理^[17]。

本文提出一种分布式爬虫系统负载均衡策略,将负载均衡问题转化为带约束条件的目标函数求解问题。采用BP神经网络构建基于节点性能参数的爬虫节点运行时间预测模型,以各节点运行时间最小方差为系统目标函数,并利用受约束的粒子群优化算法求解该目标函数的参数值,基于确定的最优解分配负载任务,以降低分布式爬虫系统的总体运行时间。

1 分布式爬虫系统设计

本文构建的分布式爬虫系统运行在Hadoop分布式平台上^[18],系统采用主从式结构,由一个主爬虫节点和多个子爬虫节点组成,其整体架构如图1所示。主节点负责生成URL链接和任务分配,子节点负责爬取数据和数据存储。主节点收集子节点返回的性能参数值,根据本文提出的负载均衡策略计算出任务分配方案,降低分布式爬虫系统的整体运行时间。

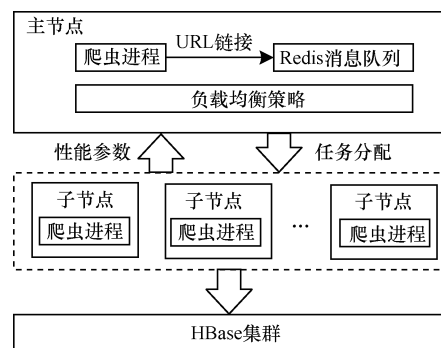


图1 分布式爬虫系统架构

分布式爬虫系统的通信主要表现为主节点和各子节点之间的通信,所有任务分配都由主节点负责,其他各个子节点之间不直接进行通信。本文使用Redis作为消息队列^[19-20]来实现主爬虫节点与各子节点之间的数据通信,主要进行待爬取任务、性能参数值和任务分配方案的传递。以上3种数据消息对应Redis消息队列中的3种消息状态值:request, capability和assignment,其中,request队列的值为待爬取的URL链接,capability队列的值为子爬虫节点的编号和对应的性能参数值,assignment队列的值为子爬虫节点的编号和对应的任务量。

分布式爬虫系统的运行流程如下:

- 1) 主爬虫节点对目标网站进行爬取,提取出目标网站中的URL链接。
- 2) 主爬虫节点将URL链接加入Redis的request队列,并利用set集合进行去重。
- 3) 统计所有的子爬虫节点当前自身的性能参数,并将对应的节点编号和性能参数值加入capability队列。
- 4) 主爬虫节点从Redis的capability队列中获取每一个子节点的信息,包括子节点的编号和性能参数,并根据负载均衡策略计算出任务分配方案,加入assignment队列。
- 5) 子节点按照assignment队列中的任务分配方案,从request队列中获取相应任务量的待爬取任务,并下载网页文本内容。
- 6) 信息提取,通过对应网站的信息提取规则从非结构化的网页数据中提取需要的信息,形成结构化数据。
- 7) 将结构化数据存储到HBase集群中。

2 负载均衡策略

2.1 负载影响参数

本文的负载均衡策略是根据不同子爬虫节点的性能情况合理分配相应的任务量,降低系统运行时间。因此,影响负载均衡的参数就是影响系统运行

时间的参数,包括爬虫节点的性能情况以及承载的任务量。

爬虫节点的性能参数涉及 CPU 性能、内存性能、磁盘性能和网络性能:1)CPU 性能由 CPU 核数和主频决定;2)内存性能主要由内存大小决定,上述参数会影响爬虫进程的运行速度;3)磁盘性能由磁盘 I/O 决定,该参数将影响分布式爬虫系统的数据存储;4)网络性能主要由网络带宽决定,其直接影响到爬虫节点下载网页的速度。

承载任务量是对主节点分配给某个子爬虫节点任务量的衡量,用 URL 链接的个数来表示。

2.2 运行时间预测模型

分布式爬虫系统的运行过程是一个复杂的非线性模型,影响其运行时间的参数包括 CPU 核数、CPU 主频、内存大小、磁盘 I/O、网络带宽和任务量。本文采用 BP 神经网络算法^[21]构建上述性能参数与运行时间间的非线性函数,选定的 3 层 BP 网络定义如下:

1)输入层。以上述性能参数为网络输入,组成 6 个神经元输入模组,对应输入向量为 $X_i = [cpu_core_i, cpu_frequency_i, memory_i, net_band_i, disk_io_i, n_i]$ 。由于输入输出值之间具有不同的量纲和量纲单位,需要通过式(1)对数据进行归一化处理。

$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

其中, x 为输入输出数据的真实值, x_{\max} 为一组输入输出数据中的最大值, x_{\min} 为一组输入输出数据中的最小值, y 为归一化处理后的数据值。

2)隐含层。根据 Kolmogorov 定理^[22],取隐含层神经元个数为 6,单元激活函数为:

$$H_j = g\left(\sum_{i=1}^6 \omega_{ij} X_i + a_j\right) \quad (2)$$

其中, ω_{ij} 为输入层到隐含层的权重, a_j 为输入层到隐含层的偏置, g 为激活函数 \tanh 。

3)输出层。网络输出为爬虫节点运行时间预测值,神经元个数为 1,输出函数为:

$$O = \sum_{j=1}^{13} H_j \omega_j + b \quad (3)$$

其中,隐含层神经元个数为 13, ω_j 为隐含层到输出层的权重, b 为隐含层到输出层的偏置。

4)参数更新过程。取节点运行时间的误差为:

$$E = \frac{1}{2} (Y - O)^2 \quad (4)$$

其中, Y 为运行时间的实际值, O 为运行时间的预测值。

权重更新公式为:

$$\omega_{ij} = \omega_{ij} + \eta H_j (1 - H_j) X_i \omega_j (Y - O) \quad (5)$$

$$\omega_j = \omega_j + \eta H_j (Y - O) \quad (6)$$

根据式(5)和式(6),不断调整输入层到隐含层的权重 ω_{ij} 以及隐含层到输出层的权重 ω_j ,直到式(4)的误差值小于给定的阈值,得到运行时间预测模型 $f(cpu_core, cpu_frequency, memory, net_band, disk_io, n)$ 。

2.3 目标函数

本文选取子爬虫节点运行时间的最小方差为系统目标函数,子节点的运行时间由其性能参数与分配的任务量决定,目标函数为:

$$\argmin \text{var}(n_1, n_2, \dots, n_M) = \frac{\sum_{i=1}^M \left(f(w_i, n_i) - \frac{\sum_{i=1}^M f(w_i, n_i)}{M} \right)^2}{M} \quad (7)$$

$$\text{s. t. } \sum_{i=1}^M n_i = N \quad (8)$$

$$n_i \geq 0, i = 1, 2, \dots, M \quad (9)$$

$$n_i \in \mathbb{Z}, i = 1, 2, \dots, M \quad (10)$$

其中, N 为主节点中待分配任务量, M 为子爬虫节点数,第 i 个子节点的性能参数 $w_i = [cpu_core_i, cpu_frequency_i, memory_i, net_band_i, disk_io_i]$,分配的任务量为 n_i ,根据运行时间预测模型计算出子节点 i 的运行时间为 $f(w_i, n_i)$ 。约束条件式(8)表示分配给每个子节点的任务量之和等于主节点中的总任务数。约束式(9)表示分配给子节点的任务量不能小于 0。约束式(10)表示任务数必须为整数。

2.4 参数计算

分布式爬虫系统任务分配存在多种不同方案,需要迭代寻优,采用穷举法的时间复杂度为 $O(N^M)$,当总任务量 N 和子爬虫节点数 M 较多时,无法满足分布式爬虫系统的要求。

本文采用一种改进的粒子群优化算法,解决分布式爬虫系统负载任务分配问题。记每一种任务分配方案为一个粒子,以上述目标函数为适应度函数,评价方案的优劣,通过上述目标函数的约束条件设立不同任务分配方案的迭代约束,直至找出满足适应度函数值的最优解,即确定一组任务分配方案。算法具体步骤如下:

步骤 1 粒子群初始化。粒子的搜索空间维度等于子爬虫节点数 M ,粒子个数定义为 30,第 i 个粒子的位置向量为 $X_i = [n_{i1}, n_{i2}, \dots, n_{iM}]$,速度向量为 $V_i = [v_{i1}, v_{i2}, \dots, v_{iM}]$ 。

本文将粒子的位置向量初始化为总任务量的平均值,即 $n_i = \lfloor \frac{N}{M} \rfloor$,加快粒子群算法的收敛速度,同时避免陷入局部极值。粒子的速度向量初始化为 $[0, 1]$ 之间的随机数。

步骤 2 计算每个粒子的适应度。选取上文目标函数为粒子适应度函数,并采用罚函数的思想对

粒子群算法的解空间进行约束,得出的适应度函数为:

$$fitness = \frac{\sum_{i=1}^M \left(f(w_i, n_i) - \frac{\sum_{i=1}^M f(w_i, n_i)}{M} \right)^2}{M} + \frac{(\sum_{i=1}^M n_i - N)^2 + (\sum_{i=1}^M (|n_i| - n_i))^2}{M} \quad (11)$$

记 $g = (\sum_{i=1}^M n_i - N)^2$ 、 $h = (\sum_{i=1}^M (|n_i| - n_i))^2$ 分别为目标函数约束式(8)和约束式(9)的罚函数,当 g 和 h 取值趋近于 0 时,适应度函数最优化问题转化为目标函数的最优化问题,实现目标函数的无约束转化。

步骤3 将步骤2中每个粒子的适应度值与个体极值 $pbest$ 的适应度值进行比较,如果某个粒子的适应度值大于当前的个体极值,则将其作为新的个体极值。

步骤4 将步骤2中每个粒子的适应度值与全局最优值 $gbest$ 的适应度值进行比较,如果某个粒子的适应度值大于当前的全局最优值,则将其作为新的全局最优值。

步骤5 根据式(12)、式(13)对粒子的速度和位置进行调整:

$$V_i = \omega V_i + c_1 r_1 (pbest_i - X_i) + c_2 r_2 (gbest_i - X_i) \quad (12)$$

$$X_i = X_i + [V_i] \quad (13)$$

其中, c_1 和 c_2 为加速权重,分别取 2, r_1 和 r_2 为 $[0, 1]$ 上的随机数, ω 为惯性权重,取 0.8。

由于 r_1 、 r_2 和 ω 为非整数,因此需要在式(13)中对速度 V_i 进行取整,保证粒子群算法每次在整数空间进行迭代,即满足约束式(10)。

步骤6 若达到终止条件,则返回当前全局最优值 $gbest$,将其作为分布式爬虫系统的任务分配方案;否则返回步骤2继续进行迭代优化。

3 实验结果与分析

本文的分布式爬虫系统由运行在 Hadoop 分布式平台上的 1 个主爬虫节点和 5 个子节点组成,主节点和子节点的软硬件环境均为 Windows10 x64 位操作系统,分布式爬虫系统采用 Python 编程语言实现。下文将对单节点中的不同负载影响参数和系统负载均衡策略进行性能测试,并与基于分布式哈希表的平均分配策略进行对比分析,验证本文提出的负载均衡策略的有效性。

3.1 节点负载参数测试

本文的负载影响参数包括 CPU 核数、CPU 主频、内存大小、磁盘 I/O 速度、网络带宽以及子爬虫节点承载的任务量,下文将分别对上述 6 个影响参数进行测试。

子节点的 CPU 核数分别设置为 1、2、4, CPU 主

频为 2.4 GHz, 内存大小为 4 GB, 网络带宽为 18 Mb/s, 磁盘 I/O 为 18 Mb/s, 任务量为 100, 实验结果如图 2 所示。

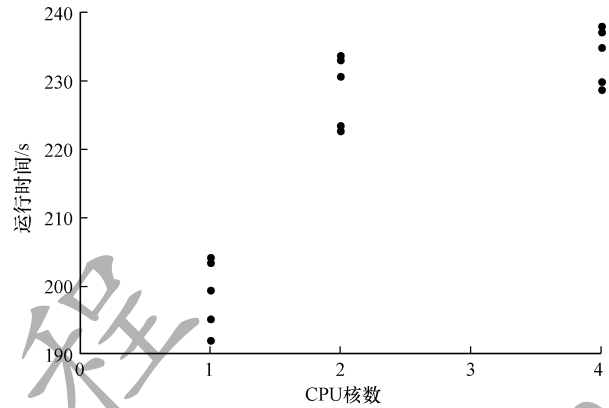


图2 CPU核数对运行时间的影响

子节点的 CPU 核数为 4, CPU 主频分别设置为 2.4 GHz 和 2.6 GHz, 内存大小为 4 GB, 网络带宽为 22 Mb/s, 磁盘 I/O 为 2 Mb/s, 任务量为 100, 实验结果如图 3 所示。

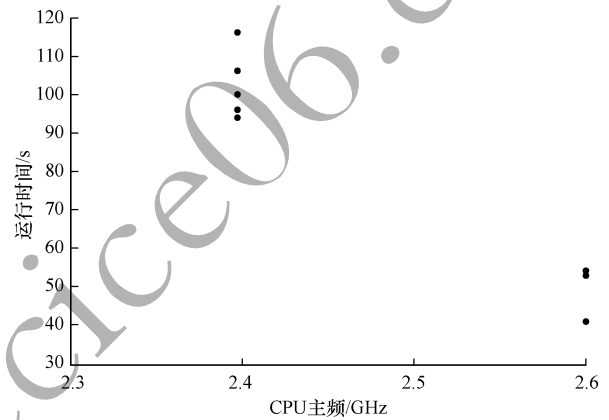


图3 CPU主频对运行时间的影响

子节点的 CPU 核数为 4, CPU 主频为 2.4 GHz, 内存大小分别设置为 0.5 GB、1.0 GB、1.5 GB、2.0 GB、2.5 GB 和 3.0 GB, 网络带宽为 2 Mb/s, 磁盘 I/O 为 2 Mb/s, 任务量为 100, 实验结果如图 4 所示。

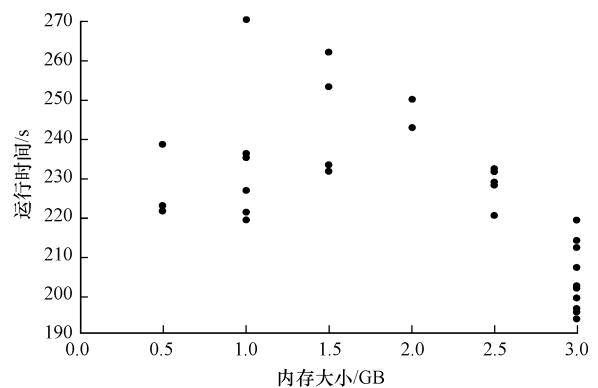


图4 内存大小对运行时间的影响

子节点的 CPU 核数为 4, CPU 主频为 2.4 GHz, 内存大小为 4 GB, 网络带宽在 2 Mb/s ~ 30 Mb/s 内变化, 磁盘 I/O 为 18 Mb/s, 任务量为 100, 实验结果如图 5 所示。

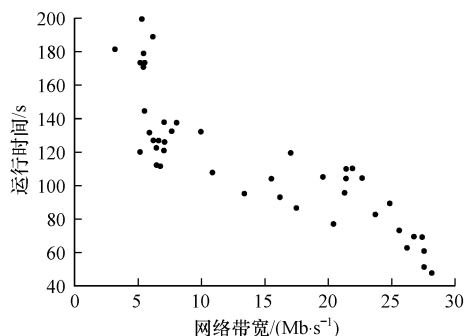


图 5 网络带宽对运行时间的影响

子节点的 CPU 核数为 4, CPU 主频为 2.4 GHz, 内存大小为 4 GB, 网络带宽为 18 Mb/s, 磁盘 I/O 速度为 2 Mb/s 和 18 Mb/s, 任务量为 100, 实验结果如图 6 所示。

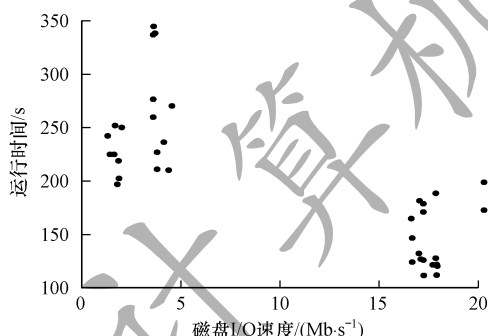


图 6 磁盘 I/O 速度对运行时间的影响

子节点的 CPU 核数为 4, CPU 主频为 2.4 GHz, 内存大小为 4 GB, 网络带宽为 6 Mb/s, 磁盘 I/O 为 18 Mb/s, 任务量设置为 1 ~ 150, 实验结果如图 7 所示。

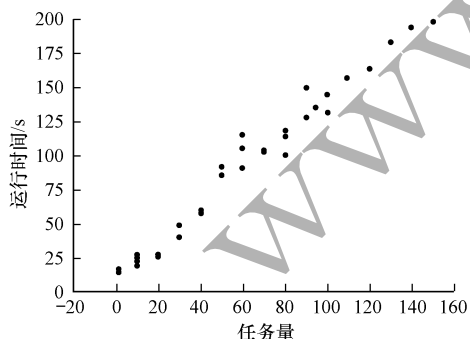


图 7 任务量对运行时间的影响

从图 2 ~ 图 7 可以看出:

1) 子节点性能参数 CPU 核数、CPU 主频、内存大小、磁盘 I/O、网络带宽及承载的任务量均对爬虫运行时间产生影响。

2) 其余参数固定, CPU 核数与子节点爬虫运行

时间正相关, 因单个爬虫进程在不同 CPU 内核运行, 造成进程迁移开销较大。

3) 除 CPU 核数和任务量外, 其余参数的单一变化与子节点爬虫运行时间均呈现负相关。

3.2 负载均衡策略测试

实验分别对本文提出的负载均衡策略和基于分布式哈希表的平均分配策略进行测试, 各子节点的性能参数如表 1 所示。

表 1 爬虫节点的性能参数

子节点 编号	CPU 核数	CPU 主频 /GHz	内存大小 /GB	网络带宽 /(Mb · s ⁻¹)	磁盘 I/O 速度 /(Mb · s ⁻¹)
1	4	2.4	4	18	18.0
2	4	2.4	4	5	18.0
3	4	2.4	1	12	3.0
4	4	2.6	3	2	1.5
5	1	2.4	3	12	1.0

主爬虫节点中的总任务量为 500, 若采用本文提出的负载均衡策略, 则每个子节点分配的任务量分别为 164、70、120、36、110; 若采用基于分布式哈希表的平均分配策略, 则将 Redis 中的 URL 链接进行编号, 哈希函数采用除留余数法, 保证所有任务平均分配给各子节点, 即每个子节点分配到的任务量均为 100。负载均衡策略的各子节点运行时间如图 8 所示。

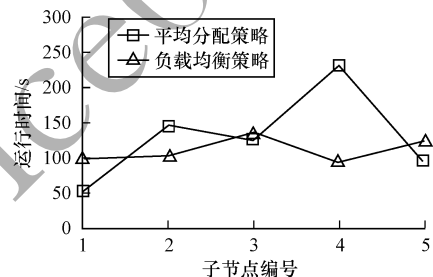


图 8 负载均衡策略的子节点运行时间比较

由图 8 可知, 本文提出的负载均衡策略能够根据子节点的性能参数分配不同的任务量, 系统总体运行时间为 134 s, 平均分配策略对应的总体运行时间为 233 s。可以看出, 本文提出的负载均衡策略能够保证每个子节点的运行时间差保持在较小范围内, 在很大程度上缩短了系统整体运行时间。

4 结束语

本文设计一种高效的分布式爬虫系统负载均衡策略, 全面分析了 CPU 核数、CPU 主频、内存大小、网络带宽、磁盘 I/O 速度和任务量等参数对系统负载均衡的影响。基于 BP 神经网络构建多参数分布式爬虫系统节点运行时间模型, 并以各节点间运行时间最小方差构建系统负载均衡策略的目标函数。通过设计含罚函数的粒子适应度函数, 完成带约束

条件的系统负载均衡策略目标函数的求解。实验结果表明,本文提出的整体负载均衡策略能够有效缩短分布式爬虫节点的计算时间,从而提升系统整体运行效率。

参考文献

- [1] REINSEL D, GANTZ J, RYDNING J. Data age 2025; the evolution of data to life-critical don't focus on big data; focus on the data that's big [EB/OL]. [2018-11-08]. https://assets.ey.com/content/dam/ey-sites/ey-com/en_gl/topics/workforce/Seagate-WP-DataAge2025-March-2017.pdf.
- [2] MITCHELL R. Web scraping with python [M]. Sebastopol, USA: O'Reilly Media, Inc., 2016.
- [3] SHKAPENYUK V, SUEL T. Design and implementation of a high-performance distributed Web crawler [C]//Proceedings of the 18th International Conference on Data Engineering. Washington D. C., USA: IEEE Press, 2002:357.
- [4] SHI Yuliang, ZHANG Ti. Design and implementation of a scalable distributed Web crawler based on Hadoop [C]//Proceedings of the 2nd International Conference on Big Data Analysis. Washington D. C., USA: IEEE Press, 2017:537-541.
- [5] LIN K, CHUNG S, LIN C. A fast and distributed algorithm for mining frequent patterns in congested networks [J]. Computing, 2016, 98(3):235-256.
- [6] JAGANATHAN P, KARTHIKEYAN T. Highly efficient architecture for scalable focused crawling using incremental parallel Web crawler [J]. Journal of Computer Science, 2015, 11(1):120-126.
- [7] 董禹龙, 杨连贺, 马欣. 主动获取式的分布式网络爬虫集群方法研究 [J]. 计算机科学, 2018, 45(S1):428-432.
- [8] 李婷. 分布式爬虫任务调度与 AJAX 页面抓取研究 [D]. 成都: 电子科技大学, 2015.
- [9] KARGER D, RUHL M. Simple efficient load balancing algorithms for peer-to-peer systems [C]//Proceedings of the 16th ACM Symposium on Parallel Algorithms and Architectures. New York, USA: ACM Press, 2004:27-30.
- [10] 王霓虹, 张露露. 分布式爬虫任务调度策略的优化 [J]. 黑龙江大学自然科学学报, 2016, 33(5):671-675, 701.
- [11] NASRI M, SHARIFI M. Load balancing using consistent hashing: a real challenge for large scale distributed Web crawlers [C]//Proceedings of 2009 International Conference on Advanced Information Networking and Applications Workshops. New York, USA: ACM Press, 2009:715-720.
- [12] YING Zheyu, ZHANG Fengli, FAN Qingyu. Consistent hashing algorithm based on slice in improving scrapy-redis distributed crawler efficiency [C]//Proceedings of 2018 IEEE International Conference on Computer and Communication Engineering Technology. Washington D. C., USA: IEEE Press, 2018:334-340.
- [13] MAKRIS A, TSERPEIS K, ANAGNOSTOPOULOS D, et al. Load balancing for minimizing the average response time of get operations in distributed key-value stores [C]//Proceedings of the 14th International Conference on Networking, Sensing and Control. Washington D. C., USA: IEEE Press, 2017:263-269.
- [14] HU Licong, XU Yajing, XU Huimin. A dynamic load balancing algorithm based on consistent hash [C]//Proceedings of the 2nd IEEE Advanced Information Management, Communication, Electronic and Automation Control Conference. Washington D. C., USA: IEEE Press, 2018:2387-2391.
- [15] GE Dajie, DING Zhijun. A task scheduling strategy based on weighted round robin for distributed crawler [C]//Proceedings of Concurrency and Computation-Practice and Experience. Washington D. C., USA: IEEE Press, 2015:848-852.
- [16] 付志辉. 分布式爬虫的动态负载均衡方法研究 [D]. 哈尔滨: 哈尔滨工业大学, 2014.
- [17] 孙守兴. 基于可扩展哈希算法的并行爬虫动态负载均衡实现 [D]. 哈尔滨: 哈尔滨工业大学, 2010.
- [18] SU Linping, WANG Fengxiao. Web crawler model of fetching data speedily based on Hadoop distributed system [C]//Proceedings of the 7th IEEE International Conference on Software Engineering and Service Science. Washington D. C., USA: IEEE Press, 2016:927-931.
- [19] LIU Lina, LIU Xuemin, ZHANG Shibo, et al. High efficient distributed cache system based on Redis cluster [J]. Computer Systems and Applications, 2018, 27(10):91-98.
- [20] 闫明. 高可用可扩展集群化 Redis 设计与实现 [D]. 西安: 西安电子科技大学, 2014.
- [21] ZHUO Guirong, WANG Bingxue. Structure designing of BP neural network in the application of reference velocity estimation [C]//Proceedings of 2014 IEEE International Conference on Mechatronics and Automation. Washington D. C., USA: IEEE Press, 2014:1481-1485.
- [22] NICLSCN R. Kolmogorov's mapping neutral network existence theorem [C]//Proceedings of the 1st International Conference on Neural Networks. Washington D. C., USA: IEEE Press, 1987:11-13.

编辑 陆燕菲