软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(6):1860−1874 [doi: 10.13328/j.cnki.jos.005710]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向大数据分析作业的启发式云资源供给方法*

吴悦文[1,3], 吴 恒[1], 任 杰[3], 张文博[1], 魏 峻[1,2,3], 王 焘[1], 钟 华[1,2,3]

[1](中国科学院 软件研究所 软件工程技术中心,北京 100190)

[2](天基综合信息系统重点实验室(中国科学院 软件研究所),北京 100190)

[3](中国科学院大学,北京 100049)

通讯作者: 吴恒, E-mail: wuheng@otcaix.iscas.ac.cn

摘 要: 云计算已成为大数据分析作业的主流运行支撑环境,选择合适的云资源优化其性能面临巨大挑战.当前研究主要考虑大数据分析框架(如 Hadoop,Spark 等)的多样性,采用机器学习方法进行资源供给,但样本少容易陷入局部最优解.提出了大数据环境下基于负载分类的启发式云资源供给方法 RP-CH,基于云资源共享特点,获取其他大数据分析作业的运行时监测和云资源配置信息,建立负载分类与优化云资源配置的启发式规则,并将该规则作用到贝叶斯优化算法的收益函数.基于 HiBench,SparkBench 测试基准的结果显示:RP-CH 相对于已有方法 CherryPick、大数据分析作业的性能平均提升了 58%,成本平均减少了 44%.

关键词: 大数据分析;云计算;启发式;云资源供给;贝叶斯优化

中图法分类号: TP316

中文引用格式: 吴悦文,吴恒,任杰,张文博,魏峻,王焘,钟华.面向大数据分析作业的启发式云资源供给方法.软件学报,2020,31(6):1860−1874. http://www.jos.org.cn/1000-9825/5710.htm

英文引用格式: Wu YW, Wu H, Ren J, Zhang WB, Wei J, Wang T, Zhong H. Heuristic based resource provisioning approach for big data analytics in cloud environment. Ruan Jian Xue Bao/Journal of Software, 2020,31(6):1860−1874 (in Chinese). http://www.jos.org.cn/1000-9825/5710.htm

Heuristic Based Resource Provisioning Approach for Big Data Analytics in Cloud Environment

WU Yue-Wen[1,3], WU Heng[1], REN Jie[3], ZHANG Wen-Bo[1], WEI Jun[1,2,3], WANG Tao[1], ZHONG Hua[1,2,3]

[1](Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](Science & Technology on Integrated Information System Laboratory (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[3](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: It is a big challenge to pick up the best cloud configuration for recurring big data analytics jobs running in clouds. Prior efforts may get in a sub-optimal configuration due to a broad spectrum of cloud configurations with a few test runs, such as CherryPick. RP-CH, presented in this paper, is a resource provisioning system that leverages heuristic rules based on classification information to identify the optimal cloud configuration for big data analytics jobs, while the insight is classifying a job by comparing its resource preference and usage information with other jobs. Then, heuristic rules are used to distinguish bad samples from good ones in Bayesian

* 基金项目: 国家重点研发计划(2017YFB1400804); 北京市自然科学基金(4182070); 蚂蚁金服科研基金(XZ502017000730); 中国科学院青年创新促进会人才专项(2018144)

Foundation item: National Key Research and Development Program of China (2017YFB1400804); Beijing Natural Science Foundation (4182070); Ant Financial Research Fund (XZ502017000730); Youth Innovation Promotion Association of Chinese Academy of Sciences Fund (2018144)

收稿时间: 2018-06-06; 修改时间: 2018-09-30; 采用时间: 2018-11-15

optimization algorithm. The experiments on HiBench and SparkBench in Aliyun ECS show that the performance of job has been improved by 58% in average comparing with CherryPick, meanwhile the resource cost has been reduced by 44% in average.

Key words: big data analytics; cloud computing; heuristic; cloud resource provisioning; Bayesian optimization

云计算已成为大数据分析作业的主流运行支撑环境,Gartner 数据显示,超过半数的全球大型组织已使用云计算进行大数据分析[1].其中,40%的大数据分析作业具有周期性处理相似规模数据的特点,例如销售额日统计等.相关研究表明,适合的云资源供给对于优化大数据分析作业性能和节约成本具有重要意义.如 Ernest[2]对比了云资源的人工选择和优化配置,大数据分析作业的性能会相差 2 倍~3 倍,CherryPick[3]认为:在相同的大数据分析作业性能前提下,云资源成本可能会相差 12 倍.已有研究主要分为两类:一类是模型驱动,面向特定的大数据分析框架进行建模,如 Elastisizer[4],MRTuner[5],Ernest;另一类是采用机器学习方法,具有与大数据分析框架松耦合的优势,逐渐成为近年来研究的热点,典型工作如 CherryPick.

然而,周期性作业的云资源供给面临样本少、搜索空间过大的问题,容易陷入局部最优解.如图 1 所示,机器学习典型方法 CherryPick 找到全局优化解的样本个数(纵轴)随着云资源配置数量(横轴)增加而增长,特别是在候选方案超过 66 个时(CherryPick 实验上限).在本文,云资源配置是指主机个数和云主机类型(计算、内存、网络)的选择.

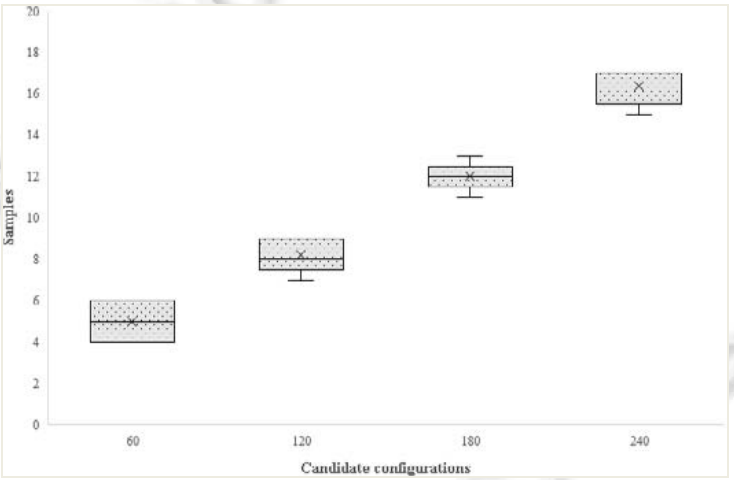


Fig.1 Samples for picking up global optimal of CherryPick (TeraSort)

图 1 CherryPick 选出全局优化解的采样次数(TeraSort)

本文提出了大数据环境下基于负载分类的启发式云资源供给方法 RP-CH(resource provisioning based on classification and heuristic),基于云资源共享特点,获取其他大数据分析作业的运行时监测数据和云资源配置信息,建立大数据分析作业分类与优化云资源配置的启发式规则,并将该规则作用到贝叶斯优化算法的收益函数 *AC*(acquisition function),可减少贝叶斯优化算法搜索空间,在小样本下可快速收敛找到优化的资源供给方案.

本文的主要贡献如下.

(1) 设计面向大数据分析作业的分类器,建立大数据分析作业分类与优化云资源配置的启发式规则,通过计算新的大数据分析作业与历史性能数据(CPU、内存、磁盘和网络)的弗雷歇距离(Fréchet distance)[4],对新的大数据分析作业进行归类;

(2) 将启发式规则通过贝叶斯优化的 *AC* 函数进行表示,其目的是在不影响优化结果的前提下减少搜索空间,在样本少、搜索空间大的场景下,解决冷启动和 *K* 阶欺骗问题,提高小样本下找到全局优化解的概率;

(3) 通过实验验证 RP-CH 在相同的小样本下,相比于 CherryPick,大数据分析作业的性能平均提升了 58%,成本平均减少了 44%.

本文第 1 节为当前相关工作的分类与总结.第 2 节阐述问题分析和研究思路.第 3 节介绍启发式规则的建立.第 4 节介绍 RP-CH 算法.第 5 节为实验,验证小样本下资源供给的应用性能和资源成本.第 6 节对本文工作进行总结,并对未来工作进行展望.

1 相关工作

目前,云资源供给方法的研究主要面向两大主要应用场景,即传统服务类应用和大数据分析应用.由于两种应用类型的特征不同,相关工作关注的研究内容也有所区别.传统服务类应用(如 Web 服务器、数据库等)需要长时间运行(数月)在云服务器上,且资源需求的特征相对确定(如峰值),因此对于此类应用,通常根据其资源需求做出资源使用的事前规划,以提升应用性能和云服务器的资源利用率;另一方面,大数据分析应用相比于传统服务类应用,其运行周期通常较短(几分钟~数小时),且资源需求的特征具有不确定性[6],因此对于此类应用,通常根据数学模型进行应用的性能预测和估算,以匹配最优云资源配置,进而提升应用性能和减小云资源成本.

面向大数据分析的云资源供给方法主要分为性能建模和机器学习两类.

- 性能建模方法

通过分析计算框架运行原理,采用如控制理论、排队论等理论基础来进行性能预测,并据此推荐资源供给方案.MRTuner[6]对 Hadoop 系统内部运行原理进行分析,并将 Hadoop 作业分解成 Producer-Transporter-Consumer 这 3 个阶段,根据控制理论分别估算作业各个阶段的调度、计算和数据传输的时间开销,并最终汇总为作业的完成时间.Elastisizer[12]基于排队论分析 MapReduce 执行过程中由于资源受限导致的作业等待现象,将 MapReduce 作业划分成多个批次执行,通过日志分析、代码插桩等手段收集数据,最终精确估算作业分批次执行的完成时间.此类方法基于运行原理分解作业,使得估算的准确性较高;缺点在于方法需要对目标系统有很深的理解,尤其在面对众多大数据分析框架时难以复用同一个模型,难以应对大数据分析框架多样性特点.

- 机器学习方法

对计算框架进行抽象,收集通用数据并建立学习系统,用于预测新作业的资源需求和性能.AROMA[7]收集作业执行过程中底层资源计数器的数据,采用聚类算法对作业的类型进行聚类,将资源估算问题转化为计算点到平面之间的距离.Ernest[8]面向分布式计算框架,根据统计分析将作业划分成多对一、树状、多对多等 3 种数据交互模式,并据此建立模型,利用离线分析过程的小规模采样样本对模型进行参数学习,从而获得配置与性能之间的模型关系.CherryPick[2]提出了采用贝叶斯优化算法来进行资源推荐,只需记录作业的完成时间和云资源配置,通过统计概率分析下一个潜在的更优配置,并不断迭代逼近最优配置.此类方法通常采用跨平台通用的数据如性能数据、交互模式作为特征,屏蔽了作业的执行细节,模型的通用性更强.与此同时,学习方法依赖于数据样本的训练结果,而获取足够多数据样本的成本过高.相关工作权衡利弊之后,往往采用小样本和有限的迭代获取数据[8,9],学习的结果容易陷入局部最优解,准确性难以保障.

本文提出了两阶段的云资源供给方法,即离线负载分类阶段和在线搜索阶段.与我们之前的研究工作[10]相比较,本文在负载分类的方法上有所改进,不再通过人工设置阈值的方式来划分负载类型,而改为采用测算资源使用曲线的相似度,提升了系统对于负载变化的场景适用性.

综上所述,已有研究采用机器学习的方法应对大数据分析框架的多样性特点,但在小样本下容易陷入局部最优解.

2 问题分析和研究思路

2.1 问题定义

本文的目标是根据作业的资源需求推荐优化的云资源供给方案,在保障作业性能的同时,尽可能降低资源成本.形式化地,大数据分析作业的资源供给问题可用如下公式定义.

$$\left.\begin{array}{l}\text{minimize } C(\vec{x})\times T(\vec{x})\\ \text{subject to } T(\vec{x})\leqslant T_{\max}\end{array}\right\} \tag{1}$$

其中,$\vec{x}$ 是以向量表示的作业特征和云资源配置的集合,作业特征主要指作业数据量大小,云资源配置包含主机个数、内存大小、CPU 核心数、磁盘带宽、网络带宽等属性;$T(\vec{x})$ 表示在 配置下运行作业的完成时间;$P(\vec{x})$ 表示 $\vec{x}$ 配置下单位时间产生的费用;$T_{\max}$ 为用户的期望时间;$C(\vec{x})$ 表示在 $\vec{x}$ 下运行作业所需要的成本.则问题定义为在找到用户期望时间 $T_{\max}$ 内的资源供给方案,同时该方案的 $C(\vec{x})$ 成本最小.

2.2 研究思路

本文通过收集其他作业运行时监控和云资源配置信息,对目标作业进行分类,建立基于负载分类信息的启发式规则,并将启发式规则作用到贝叶斯优化的 *AC* 函数.形式化地,与启发式规则结合的贝叶斯优化 RP-CH 的目标函数见公式(2):

$$\left.\begin{array}{l}C(\vec{x})=\arg\min f(\vec{x})+\varepsilon,\vec{x}\in\{\vec{x}_1,\vec{x}_2,...,\vec{x}_n\}\\ \partial(\vec{x})\to classifier(\vec{x}),\vec{x}\to\{(\vec{x}_1,\vec{x}_3,...),(\vec{x}_2,\vec{x}_4,...),(\vec{x}_5,\vec{x}_6,...)\}\\ f(\vec{x})=\max\text{imize } AC(\vec{x}),\vec{x}\in\{(\vec{x}_1,\vec{x}_3,...)\}\end{array}\right\} \tag{2}$$

其中,$C(\vec{x})$ 是符合贝叶斯优化模型 $f(\vec{x})$ 的资源成本,ε是环境噪声,$\partial(\vec{x})$ 为启发式规则,$f(\vec{x})$ 取当前样本中 *AC* 函数值最大的样本.为了获得云资源供给的全局优化解,需要对云资源配置的候选集 $\vec{x}$ 进行全局搜索,然而 $\vec{x}$ 包含作业、资源的多个维度的特征,算法的搜索空间较大,在样本少时容易陷入局部最优解,因此需要分析作业的执行特征,形成作业的分类信息,并据此减小搜索空间.公式(2)中基于启发式规则的分类器 $\partial(\vec{x})$,将搜索空间 $\{\vec{x}_1,\vec{x}_2,...,\vec{x}_n\}$ 分为 $\{(\vec{x}_1,\vec{x}_3,...),(\vec{x}_2,\vec{x}_3,...),(\vec{x}_5,\vec{x}_6,...)\}$ 这 3 个区间,在求解目标函数 $f(\vec{x})$ 时优先在作业负载类型匹配的区间 $\{(\vec{x}_1,\vec{x}_3,...)\}$ 内进行采样,以达到缩小搜索空间的目的.

然而,如何建立启发式规则,并实现基于启发式规则的云资源供给算法面临挑战,具体的分析如下所述.

1) 启发式规则的建立.启发式规则建立在作业分类的基础之上,作业分类的难点在于相同作业在不同云资源配置下执行结果存在差异,单纯统计作业的平均资源利用率难以做到准确分类.如何对作业进行分类,并建立作业分类信息与云资源供给优化的启发式规则面临挑战;
2) 启发式规则的表示.传统贝叶斯优化算法在样本少、搜索空间大时容易陷入局部最优解是因为算法存在冷启动和 *K* 阶欺骗问题[11],本文的思路是通过启发式规则减少样本搜索空间,优化样本选择.如何将启发式规则表示为贝叶斯优化采样过程面临挑战.

3 启发式规则的建立

3.1 特征的选择

相关研究表明[12],当前企业内部运行的作业,可根据作业负载的资源使用特征,将作业分为计算密集型作业、I/O 密集型作业和混合型作业.计算密集型作业在作业执行过程中存在大量的计算步骤,对于 CPU 个数、CPU 主频、内存大小、CPU 内存比率等与运算相关的属性比较敏感.如浮点运算程序、天文运算、图形处理等等.I/O 密集型作业特点是运行过程中需要进行大量的数据交换、传输,主要操作为数值交换、节点间通信、数据的读写等,以网络带宽、磁盘大小、磁盘读写速度等为主要资源瓶颈的作业,如简单排序、SQL 聚合等作业.如果作业同时具备计算密集型特征和 I/O 密集型特征,则定义为混合型作业,如流式计算、数据清洗等作业.研究表明:具有相似资源需求的数据分析型作业的性能表现相近[13],每一类作业都存在云资源优化与云资源属性之间的关联关系,如对一个 IO 密集型作业进行云资源配置推荐,同时提高磁盘带宽、磁盘大小、网络带宽等属性,则会更快地找到全局优化解.

本文的云资源供给场景建立在云资源共享的前提条件下.以亚马逊、微软、阿里云为代表的主流云计算平台具有用户共享云资源的特性,云服务供应商可以通过运行时监测和日志分析获取其他作业执行的资源使用和配置信息.

相关领域的研究工作中,采用运行时监测和日志分析等手段优化求解过程是一种主流的研究手段[14−16].

3.2 负载分类

本文通过分析作业负载的资源使用偏好对作业进行分类.具体的步骤如图 2 所示,首先分析历史作业负载的资源使用曲线,通过分类器分析多维资源的使用曲线(CPU、内存、磁盘、网络),并将作业分为计算密集型、IO 密集型和混合型这 3 种类型;当新作业执行时,分类器将比较资源使用曲线的相似度,并判断新作业的类型;最终,作业分类知识将应用到 RP-CH 算法中.

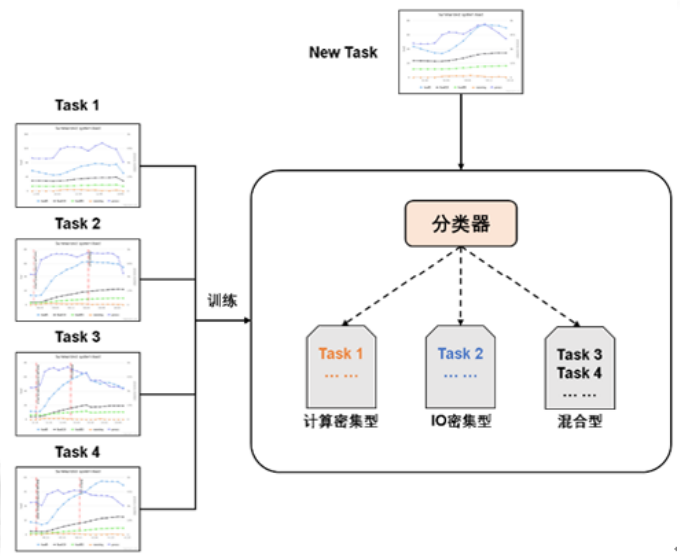


Fig.2 An example of RP-CH's task classification working process

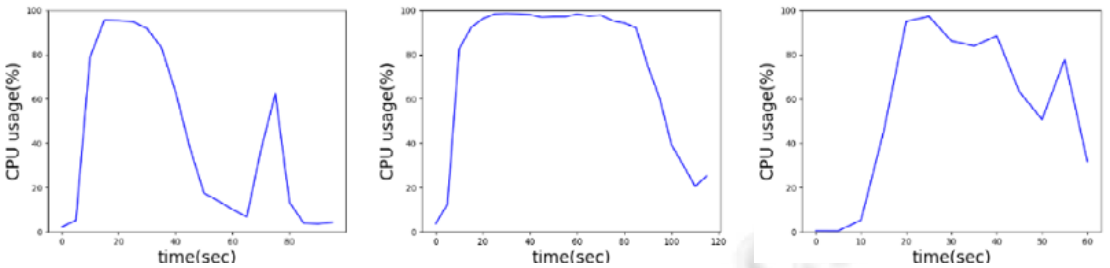
图 2 作业分类的基本原理

在对作业负载进行资源使用曲线的相似度比较时,相关工作[17]通过操作系统底层的性能计数器收集数据,将性能数据分别转化成一组点集,并分别对点集计算基于最长公共子序列(longest common subsequence,简称 LCSS)的距离,最终计算出来的距离数值成为作业类型判断的标准.

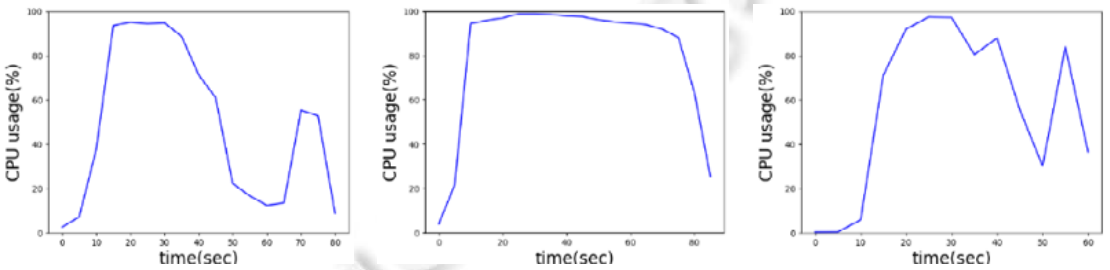
资源使用曲线相似度比较的难点是考虑云资源配置差异的影响.如图 3 所示,展示了阿里云云计算平台中在不同云资源配置下运行 HiBench 的 TeraSort,WordCount,SQL Aggregation 作业的结果,其中可明显看出:TeraSort 和 WordCount 的配置 1 执行比配置 2 执行的完成时间更长,曲线差异表现为时间轴上的图像平移,SQL Aggregation 的两次执行结果的差异表现为局部数值上的缩放.由此可见:在比较数据分析型负载的资源使用曲线时,需要统筹考虑云资源配置差异的时间翘曲距离(canonical warping distance)[18].

如图 4 所示,横轴表示多种资源,纵轴表示距离,距离越小表示曲线越相似.相关工作中[19]采用基于点集相似度的计算方法 LCSS,则结果将产生较大的误差.如图 4(a)所示,LCSS 的结果显示,TeraSort 和 WordCount 的各种资源使用的距离总和更小,判断两者相似度更高.而实际上,TeraSort 在不同云资源配置下的两次执行应该更相似,但由于云资源配置差异导致的图像平移和局部数据缩放现象,如图 3(a)和图 3(d)所示:TeraSort 在两种不同配置下的图像差异,使得 LCSS 方法出现较大的误差.

为了解决云资源配置差异导致的相似度比较误差问题,本文采用基于曲线形状的相似度比较方法,其中,弗雷歇距离[20]应用广泛,具有较强的噪声容忍性.其原理是:基于曲线斜率比较形状相似度,根据斜率的差值在离散的点中间插入新的点以解决噪声问题,增加形状相似度比较的准确性,对于噪声引起的图像平移和数值缩放有较好的效果.图 4(b)展示了 Fréchet 算法的结果,最终判断 2 次 TeraSort 执行的资源曲线更相似,修正了云资源配置差异造成的误差.



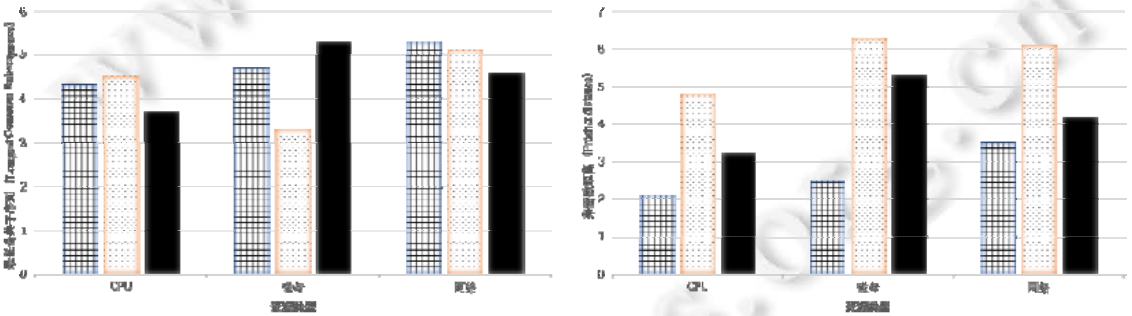
(a) TeraSort 配置 1 (b) WordCount 配置 1 (c) SQL Aggregation 配置 1



(d) TeraSort 配置 2 (e) WordCount 配置 2 (f) SQL Aggregation 配置 2

Fig.3 CPU utilization of TeraSort, WordCount and SQL Aggregation tasks during different configurations

图 3 TeraSort,WordCount,SQL Aggregation 不同云资源配置下运行的 CPU 曲线图



(a) 资源的 LCSS 距离 (b) 资源的弗雷歇距离

Fig.4 Distances for CPU, disk and network resources

图 4 资源曲线的 CPU、磁盘和网络距离比较

3.3 启发式规则

在获取作业分类信息的基础之上,通过启发式规则构建作业分类和云资源配置的关联关系,用于 RP-CH 算法中缩减搜索空间,并指导算法选择下一个采样点,选择的特征见表 1,其中,{CPU memory ratio,CPU speed,disk speed,network speed}这 4 个特征集合用于判断该向量 $\vec{x}$ 是否与作业分类信息相匹配,例如,计算密集型作业符合{高,高,低,低}的特征集合(IO 资源需求低以降低资源成本);特征 Scale memory ratio 用于判断向量 $\vec{x}$ 的云资源配置是否满足作业数据规模的资源需求,例如,当 Scale=40GB 的作业在总内存≥40GB 的云资源配置方案中执行效果更佳,取值范围从 0~50,表明本文实验环境中赋予“数据规模/内存比率”的上限是 50,是因为当该值超过 50 时,以 Spark 为代表的内存计算型 Benchmark 可能因为内存不足而崩溃[21].

Table 1 System features selected from $\vec{x}$ for building heuristic rules
表 1 特征向量 $\vec{x}$ 中用于构建启发式规则的特征

特征	描述	类型	范围
CPU memory ratio	CPU 内存比率	标称(nominal)	高、中、低
CPU speed	CPU 速度	标称(nominal)	高、中、低
Disk speed	磁盘速度	标称(nominal)	高、中、低
Network speed	网络速度	标称(nominal)	高、中、低
Scale memory ratio	作业数据规模/总内存比率	数值(numeric)	(0,50]

启发式规则的函数表达式见公式(3):

$$\begin{aligned}&\partial(\vec{x})=\theta_1 rate_1(\vec{x})+\theta_2 rate_2(\vec{x})\\&rate_1=\{CPU\ memory\ ratio, CPU\ speed, Disk\ speed, Network\ speed\}\\&rate_2=Scale\ memory\ ratio\end{aligned} \tag{3}$$

其中,

- $rate_1$ 代表云资源配置 $\vec{x}$ 与作业负载分类的匹配程度,$rate_1$ 由 4 维的资源特征组成,由于负载的资源特征与云资源配置之间的匹配关系难以用单一规则进行约束,因此本文设计了一个多维的启发式规则,如图 5 所示,横轴代表 4 维资源,纵轴代表具体的资源配置(high,medium,low 通过阈值控制,本文中设置为 75%~100%,25%~75%,0~25%),图中带数值的元素(圆、三角和方形)代表云资源配置 $\vec{x}$ 与负载的匹配度,例如:当 $\vec{x}=\{high,low,low,high\}$ 且作业属于计算密集型时,则 $rate_1$=(1−1+1+0)/4=0.25;
- $rate_2$ 代表云资源配置 $\vec{x}$ 与作业规模的匹配程度,对于大数据分析作业而言,数据规模与资源配置(可理解为数据处理能力)之间的关系将直接影响到作业性能,因此需要量化该指标的影响[22];
- 参数 θ_1 和 θ_2 为 $rate_1$ 和 $rate_2$ 的影响因子.通过大量实验验证 θ_1:θ_2=2:1 时,启发式规则的效果最佳,具体实验结果在后文第 5.2 节.

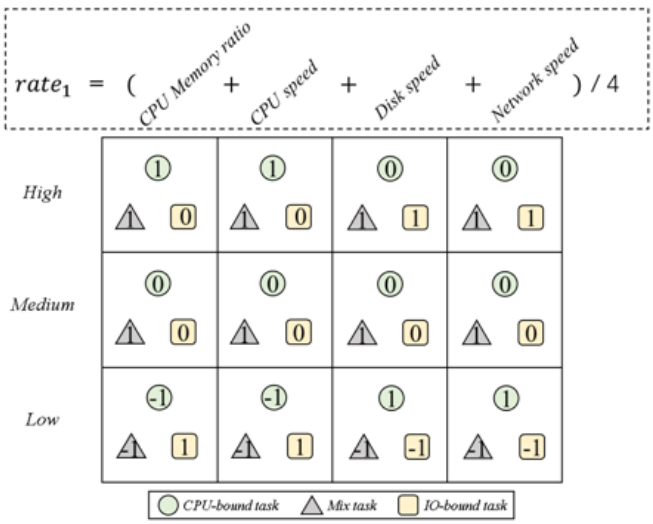


Fig.5 Heuristic rules based on jobs classification
图 5 基于负载分类匹配度的启发式规则

4 启发式规则的表示

RP-CH 将启发式规则作用到贝叶斯优化的采样过程中,通过作业负载的分类信息,强化匹配区间的样本选择,达到算法快速收敛的目的.RP-CH 的基本计算模型与贝叶斯优化相似,通过不断地迭代执行进行优化,每次执行即为一次算法采样过程,在 RP-CH 中加入了作业分类和启发式规则应用的步骤,算法基本流程如图 6 所示.

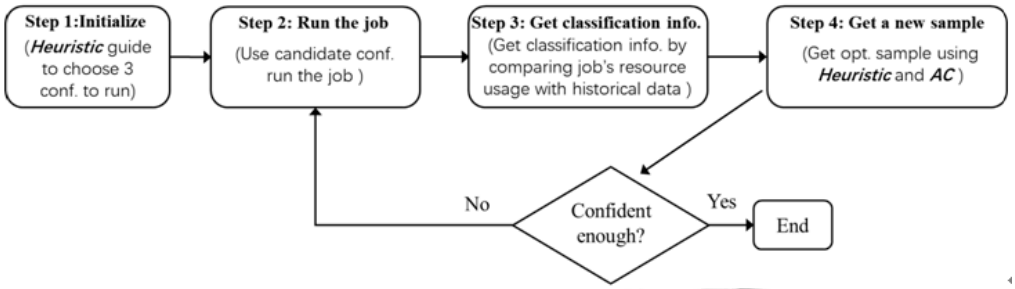


Fig.6 RP-CH workflow

图 6 RP-CH 流程图

算法的基本流程为:

1) 采用启发式规则结合作业的分类信息(通过首次执行获得)进行初始选点;
2) 运行作业;
3) 收集并分析作业负载的资源使用数据,与系统中其他作业比较相似度,对该作业进行分类;
4) 启发式规则结合 *AC* 函数从候选集中选出新的采样点,若新采样点符合选点目标,则结束算法迭代过程;否则,进入步骤 2).

伪代码见表 2.

Table 2 RP-CH algorithm

表 2 RP-CH 算法

Heuristic based Bayesian optimization algorithm
Let x_i be the ith cloud configuration Let $S=\{x_i:i=1,\ldots,\#Configuration\}$ be the set of all configurations Let $AC(x_i)$ be the Acquisition function of a configuration x_i Let p be the acceptable endpoint of the algorithm Let *Heuristic* be the optimized *AC* function based on Heuristic rules Let *Classifier* be our offline training workload classifier
1. step 1 #INITIALIZE 2. for allthe configurations in S 3. *Heuristic* guide to choose 3 configurations to run the job and then initialize Bayesian optimization procedure 4. Let *init*(3) be the initial 3 configurations
5. step 2 #RUN THE JOB 6. start a new run with candidate configuration
7. step 3 #GET Classification INFORMATION 8. for i in *init*(3) 9. comparing*workload* with *Classifier* 10. let *Classification* be the resource classification of the job
11. step 4 #GET A NEW SAMPLE 12. use *Heuristic* and *Classification* to reorder $AC(x_i)$ 13. let x_{max} be the *currentbest configuration* 14. try x_{max} as new sample and start a new run with x_{max} 15. if new run's p is confident enough 16. return x_{max} as the *best configuration* 17. else 18. jump to procedure 2

RP-CH 的改进在于作业分类信息和启发式规则的应用,其目的在于解决第 2.2 节提到的贝叶斯优化算法的两个局限性,即冷启动问题和 *K* 阶欺骗问题,以下介绍 RP-CH 如何解决这两个问题.

- 解决算法冷启动问题

贝叶斯优化通过已知采样点的 *AC* 函数运算,估算候选采样点的值,贝叶斯优化算法启动需要 $n\geqslant 3$ 个初始选点以获得足够的计算空间(标准差通过协方差矩阵表示),因此在算法启动阶段,贝叶斯优化或随机选择采样

点,或根据规则在指定采样区间选点,在本文场景中,由于样本少(6 次采样)且搜索空间大,算法随机采样的效果难以保障,导致算法冷启动问题.为此,本文的 RP-CH 在算法启动阶段运用已知的启发式规则,表示为贝叶斯优化的超参数设置,加强算法在特定区间的选点以优化算法的启动阶段.贝叶斯优化的 *AC* 函数主要由均值和标准差两部分组成:均值越大,代表该候选点离最值越近;标准差越大,代表该候选点未探索程度越高.Snoek[23]的论文介绍了 *AC* 函数的多种评分策略,其中应用最广泛的是贪心策略(expected improvement,简称 EI).

CherryPick 的贝叶斯优化采用的基于贪心策略的 *EI* 函数,形式化地,*EI* 函数的表达式如公式(4)所示:

$$AC_{EI}(x_t)=\arg\max\ j\sigma_{t-1}(\vec{x})-\mu_{t-1}(\vec{x}) \tag{4}$$

其中,x_t 表示下一个采样点,$AC_{EI}(x_t)$表示该点的收益值,$\mu_{t-1}(\vec{x})$ 表示已有采样点的均值,$\sigma_{t-1}(\vec{x})$ 表示已有采样点的方差,j 是人工设定的超参数.由于没有下一个采样点的额外特征描述,在函数启动阶段会浪费资源进行全局采样,导致算法启动较慢.

为此,RP-CH 通过算法的首次执行获取作业的分类信息,通过公式(3)的启发式规则对候选点 x_t 进行评分,并作用到 *EI* 函数的超参数中.形式化地,结合启发式规则的 *EI* 函数的表达式见公式(5):

$$AC_{EI}(x_t)=\arg\max\partial(\vec{x})\sigma_{t-1}(\vec{x})-\mu_{t-1}(\vec{x}) \tag{5}$$

其中,$\partial(\vec{x})$ 由启发式规则的函数表达公式(3)而来,目的是结合作业的分类信息,加强算法在特定区间的选点,优化采样过程.

解决 *K* 阶欺骗问题:贝叶斯优化由遗传算法演变而来,在计算过程中,难以避免特征从低维变换开始导致的搜索空间爆炸问题,CherryPick 论文指出:其搜索开销是性能建模方法 Ernest 的 4 倍,求解时间是性能建模方法 Ernest 的 10 倍以上.然而,本文在掌握了作业的分类信息之后,能够采用解决遗传算法 *K* 阶欺骗问题的思想[24],加速计算过程.*K* 阶欺骗问题是指特征之间存在内部关联关系,当算法在非 *K* 阶空间中搜索,将导致算法效率下降[25].如计算密集型作业对 CPU 个数、CPU 主频、内存大小、CPU 内存比率同时进行调整,更可能获得优化结果.因此,特征集 *U*{*cpu*,*cpu_speed*,*mem*,*cpu_mem_ratio*}是提升搜索效率的 *K* 阶搜索空间.

CherryPick 中算法在演化过程中搜索空间为所有特征的全体集合,样本特征在低维度变化时,搜索空间中的样本数量呈指数增长,虽然最终能通过遗传算子的积木块效应[26]累积成高阶搜索空间,但是大多数时候直到种群规模达到上限(求解时间超时)仍未获得最优解,导致结果准确性不足.

表 3 比较了 *K* 阶欺骗时算法的搜索空间和效率,当面对 4 阶欺骗问题时,RP-CH 算法求解效率比 CherryPick 提升了 16 倍.假设算法的最大种群规模为 200(求解时间约 20 分钟),则 CherryPick 在面对高阶欺骗时难以获得全局最优解.为此,RP-CH 算法结合启发式规则对搜索空间的数据进行筛选,通过启发式规则加速 *K* 阶积木块的形成.形式化地,启发式规则见公式(6):

$$\begin{aligned}&AC(K)=P(K)\prod\nolimits_t^n AC(x_t)\\&K\in\{CPU\ bound,Mix,IO\ bound\}\end{aligned} \tag{6}$$

其中,$AC(x_t)$为父节点所对应的收益值,$AC(K)$为先验概率 $P(K)$与所有这样的 *K* 阶结构的 *AC* 值的乘积,先验概率 $P(K)$是高斯过程公式计算而来,不影响评分机制,在此不再赘述.*K* 在不同的作业分类中有不同的取值,计算密集型取 *U*{*cpu*,*cpu_speed*,*mem*,*cpu_mem_ratio*},混合型取 *U*{*cpu_mem_ratio*,*scale_mem_ratio*},IO 密集型取 *U*{*disk_speed*,*network_speed*,*disk_size*}.公式(6)的好处在于 *K* 阶搜索空间的优先级将被大幅度提升,当获得作业分类知识后,能够加速积木块效应,使得算法向着指定方向收敛.

Table 3 Comparing RP-CH with CherryPick of *K*-Order building block's population scale and runtime

表 3 *K* 阶欺骗的搜索空间比较

K-Order	CherryPick		RP-CH	
	Population scale	Runtime (s)	Population scale	Runtime (s)
1	10	170	10	170
2	256	1 400	16	220
3	512	2 800	32	300
4	1 024	6 000	64	375

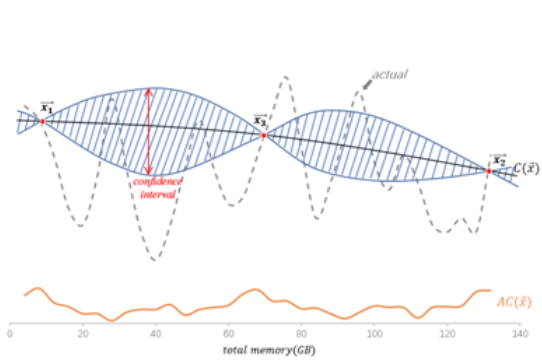
综上所述,RP-CH 主要在贝叶斯优化算法初始化阶段和后续选点阶段结合了启发式算法,旨在解决贝叶斯优化的冷启动和 K 阶欺骗问题.

图 7 展示了计算密集型作业在 RP-CH 和 CherryPick 算法初始化和后续采样阶段的比较,其中,横坐标表示配置方案的总内存大小,纵坐标表示作业的完成时间,虚线 actual 表示作业实际的资源成本曲线,曲线 $C(\vec{x})$ 表示算法预测的作业负载的资源成本曲线,斜线标注的区间表示符合高斯过程的 95%置信区间, $AC(\vec{x})$ 为当前迭代的收益函数, $\vec{x}_n$ 表示算法的第 n 次采样.

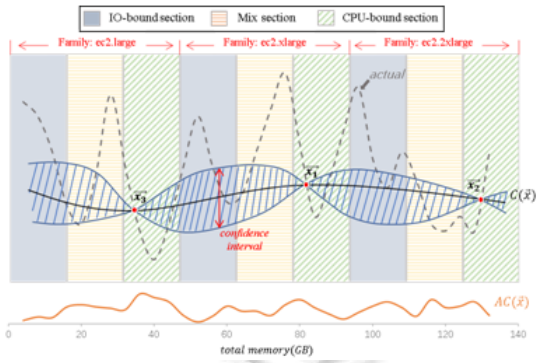
RP-CH 作业分类信息的实际应用如图 7(b)和图 7(d)所示,配置方案的总内存从小到大增加的过程中,根据主流云计算平台(阿里云)的配置规格分为 large,xlarge 和 2xlarge 这 3 类,每个配置规格又细分为 IO 密集型、混合型和计算密集型这 3 个区间,便于观察启发式规则的效果.

CherryPick 的初始化阶段如图 7(a)所示,为了保障选点的全局性,初始化 3 个点一定会选择总内存的极小值、极大值和中值.RP-CH 则通过启发式规则,将计算密集型作业快速收敛到与作业分类信息相匹配的区间,如图 7(b)所示,此时,RP-CH 更有可能获得比 CherryPick 更好的初始化结果.

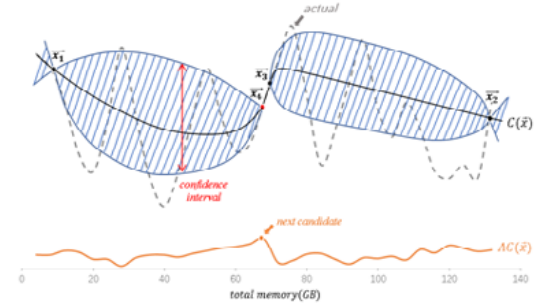
初始化结束后,CherryPick 通过 $AC(\vec{x})$ 计算下一个候选点作为算法采样点,如图 7(c)所示.由于实际曲线 actual 比较复杂,CherryPick 的采样有可能陷入局部最优解.而 RP-CH 通过启发式规则改变了算法的选点位置,达到快速收敛的效果,如图 7(d)所示,虚线 $AC(\vec{x})$ 函数为原始 AC 函数,实线为加入启发式规则的 AC 函数.



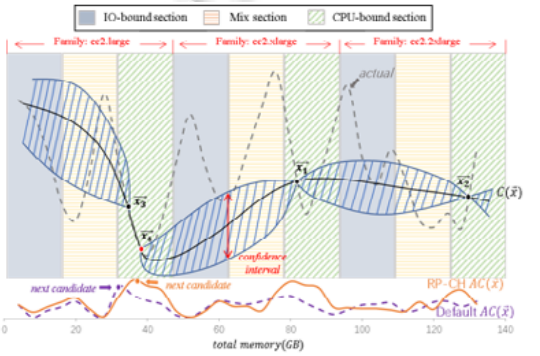
(a) CherryPick 算法启动阶段



(b) RP-CH 算法启动阶段



(c) CherryPick 通过收益函数 $AC(\vec{x})$ 选择下一个点



(d) RP-CH 通过收益函数 $AC(\vec{x})$ 选择下一个点

Fig.7 Comparing RP-CH with CherryPick on step initialization and get candidate

图 7 RP-CH 与 CherryPick 算法比较,包括算法启动阶段和选点阶段

5 实验及结果分析

本文在阿里云计算平台上对 RP-CH 进行实验验证,选用了 240 种资源供给方案;采用主流的 2 种工业级测

试基准共计 8 种应用.实验 1 验证作业负载资源分类的有效性,比较了 3 种相似度匹配算法;实验 2 验证启发式规则参数调节的效果;实验 3 与 CherryPick 进行比较,验证相同小样本下,云资源供给的应用性能和资源成本.

5.1 实验环境

- 测试基准程序:

实验的测试基准见表 4,覆盖不同资源偏好、不同运行场景的 8 个程序,分属于 HiBench[27]和 SparkBench[28]测试基准.

(1) HiBench:由 Intel 推出的面向大数据的基准测试套件,既涵盖以 TeraSort,WordCount 等为代表的微基准测试,也包括主流的机器学习、类 SQL、网络搜索、流式计算等测试基准,本文中,数据规模微基准测试为 30G,其他测试选择程序默认配置的 Huge 规模;

(2) SparkBench:由 IBM 推出的 Spark 基准测试套件,测试类型包括机器学习、图计算、类 SQL 和流式计算.本文中,测试参数的设置依据 SparkBench 论文[3],控制数据规模在 20G~30G 之间,在此不一一阐述.

Table 4 Benhmark applications

表 4 测试基准

	类型	名称	归属
资源偏好分类	计算密集型	WordCount	HiBench
		K-Means	HiBench
	I/O 密集型	Matrix factorization	SparkBench
	混合型	Sort	HiBench
应用场景分类	机器学习	Logistic regression	SparkBench
	网络搜索	SVD++	SparkBench
	类 SQL	Aggregation	HiBench
	流式应用	TwitterStreaming	SparkBench

- 资源供给方案

阿里云提供了丰富的云主机系列和云主机规格,本文选择通用型、计算型、内存型这 3 个系列共计 48 种配置规格,每种配置规格分别对应 5 种云主机个数(2,4,6,8,10),共计 240 种资源供给方案.

- 比较对象

RP-CH 的主要比较对象为 CherryPick,CherryPick 采用贝叶斯优化进行云资源配置选择,其贡献在于将贝叶斯优化应用到云资源配置优化场景,对比性能建模方法,解决了场景通用性问题.

- 评价指标

RP-CH 选择以下指标作为算法有效性的评价指标.

(1) 作业性能:在小样本场景下,使用算法推荐资源供给方案运行作业的完成时间;

(2) 资源成本:在小样本场景下,使用算法推荐资源供给方案运行作业的资源成本.

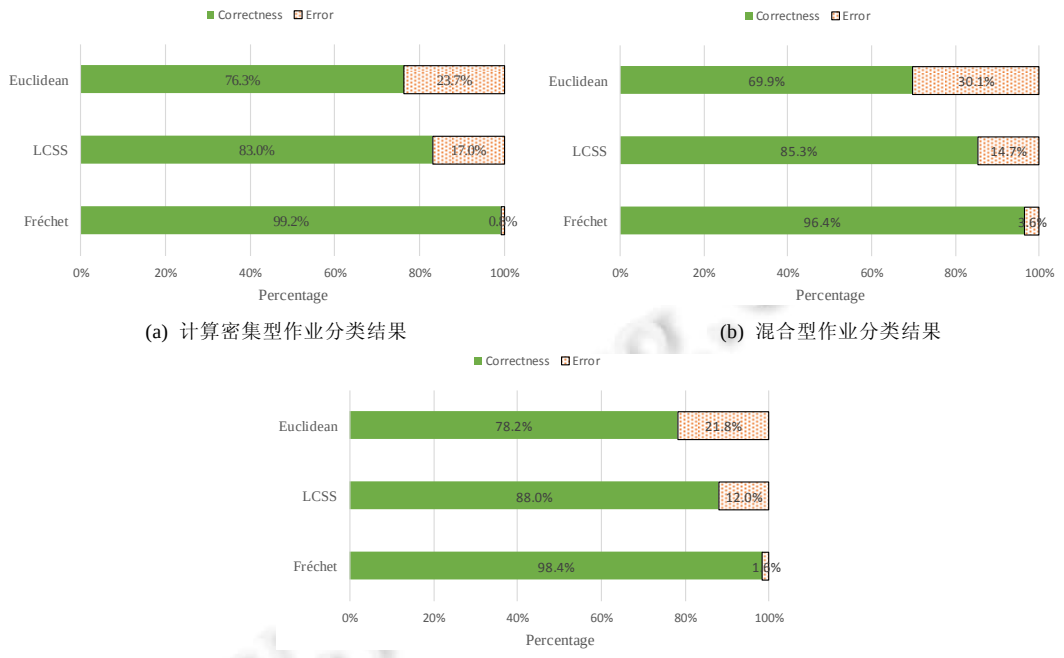
5.2 RP-CH有效性验证

本节通过多个实验对 RP-CH 的云资源供给有效性进行验证.

- 实验 1:作业分类的有效性验证

本实验验证了作业的负载分类器的算法选择问题,选取了 3 种主流的相似度比较算法欧式距离 Euclidean、最长公共子序列 LCSS 和弗雷歇距离 Fréchet 进行分类有效性的比较.在经过多种负载类型的实验后,选择误差率最低的算法作为负载分类器的核心算法.实验结论为:本文将作业负载的资源使用曲线的相似度作为作业分类标准.在比较了 3 种主流相似度匹配算法后,可知弗雷歇距离更适合本文场景,3 组实验的平均误差率为 2%.

图 8 为弗雷歇距离、最长公共子序列、欧氏距离这 3 种主流相似度匹配算法的比较结果,实验选取了 3 个不同资源偏好的基准测试,在 120 个不同配置方案中运行,对算法结果进行统计.采用弗雷歇距离在计算密集型分类时只出现 1 次判断错误,因为基准测试中的计算密集型作业特征非常明显,CPU 保持持续高利用率,真实运行环境中,可能因作业特征不同而效果存在差异.



(a) 计算密集型作业分类结果 (b) 混合型作业分类结果

(c) IO 密集型作业分类结果

Fig.8 Classify accuracy using 3 different algorithms

图 8 3 个算法的准确性比较

- 实验 2:参数调节验证

本实验旨在验证第 3.3 节中启发式规则的参数设置问题,通过公式(3)的描述我们可知:参数θ_1和θ_2分别代表了云资源供给方案对于“作业类型的匹配度”和“作业数据规模的匹配度”,两者的取值将影响作业分类的效果.因此,本实验验证了 3 种不同的参数设置,从中选取优化效果最好的设置作为候选设置.

如图 9 所示,纵轴的相对运行开销越小,表示作业的性能越好.相对运行开销是指算法推荐解与全局优化解的比值,横轴表示 8 种基准测试程序,每一种测试程序运行 20 次,柱状图上的误差线下界表示 10%统计结果,上界表示 90%统计结果,实验选择算法 6 次采样所推荐的结果作为算法推荐解.从结果可以看出:当启发式表达函数中参数设置成θ_1:θ_2=2:1,算法运行效果更优.

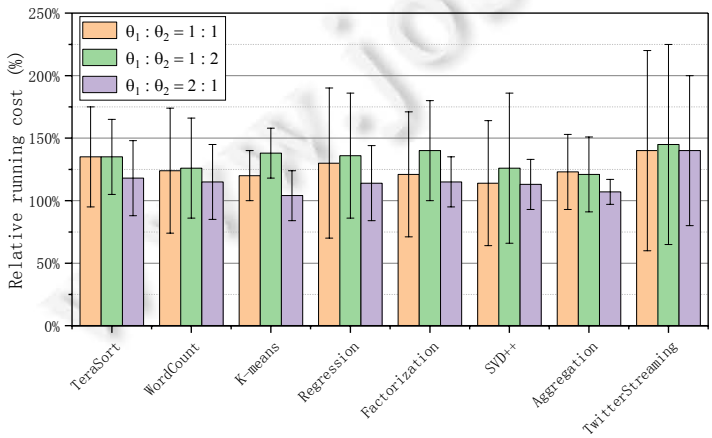


Fig.9 Tunning θ_1:θ_2 with 1:1, 2:1 and 1:2. Bars show 10th and 90th percentile

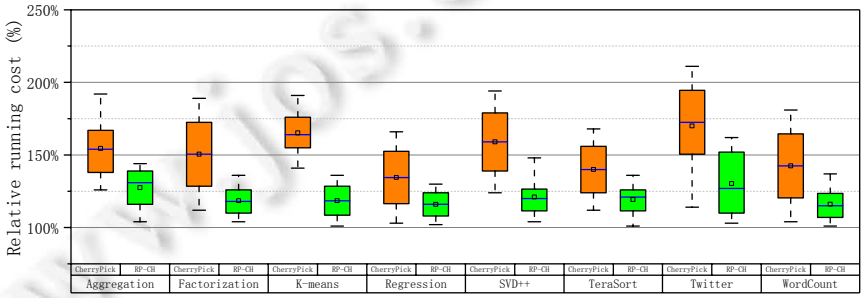
图 9 参数调节效果对比,误差线表示 10%和 90%结果

实验结论为:验证了启发式表达函数(3)中的参数赋值.若以 6 次采样为终止条件,则θ_1:θ_2=2:1 时采样结果比θ_1:θ_2=1:1 和θ_1:θ_2=1:2 优化了 20%~35%.

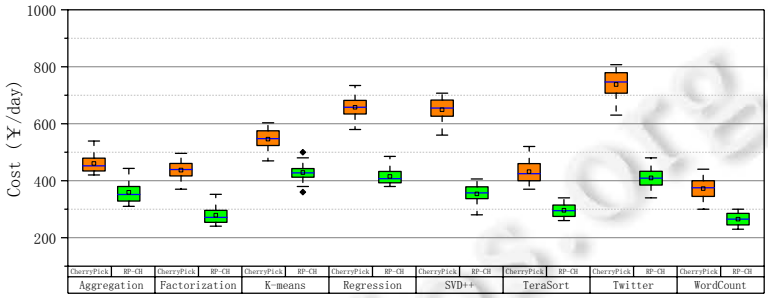
- 实验 3:应用性能和资源成本验证

本实验验证 RP-CH 对于大数据分析作业在应用性能和资源成本上的优化效果.实验从测试基准 HiBench 和 SparkBench 中选择了 8 个具有代表性的测试程序,在阿里云的 240 个候选云资源配置方案中进行实验.实验的比较对象为 CherryPick.

如图 10(a)所示,纵轴的相对运行开销越小,表示作业的性能越好.相对运行开销是指算法推荐解与全局优化解的比值,横轴表示 8 种基准测试程序,每一种测试程序运行 20 次,柱状图上的误差线下界表示 10%统计结果,上界表示 90%统计结果,实验选择算法 6 次采样所推荐的结果作为算法推荐解.从结果可以看出:RP-CH 对于多种类型的应用具有较好的推荐结果;6 次采样平均值比全局优化解的应用性能相差 18%,其中,*K*-means 和 Aggregation 的结果与全局优化解的应用性能相差不到 5%.图 10(b)显示了 RP-CH 和 CherryPick 采样结果的资源成本比较,RP-CH 资源成本平均减少 44%.



(a) 与 CherryPick 比较 6 次采样的作业性能



(b) 与 CherryPick 比较 6 次采样的资源成本

Fig.10 Comparing RP-CH with alternative solutions. Bars show 10th and 90th percentile

图 10 RP-CH 与候选方案比较,误差线表示 10%和 90%结果

实验结论为:若以 6 次采样为终止条件,则 RP-CH 的结果相比于 CherryPick 平均提升作业性能 58%,资源成本平均减少 44%.

6 总结与展望

综上所述,本文实现了大数据环境下基于负载分类的启发式云资源供给方法 RP-CH.首先,通过作业负载的资源使用曲线的相似度比较,对作业进行分类;然后,基于分类信息的启发式规则改进贝叶斯优化算法的 *AC* 函数,解决小样本下贝叶斯优化的局部最优解问题.经过实验验证,RP-CH 比 CherryPick 的作业性能平均提升 58%,成本平均减少 44%.

对于本文所提出的方法本身,仍然存在一定可提升的优化空间.如:在对于流式计算作业时,由于流式计算实时接收数据的特性,导致预测结果误差较大;启发式规则的参数调节需设计大量实验,验证参数调节的场景适

应性;在解决冷启动问题时,可评估其他收敛策略如局部贪婪策略的效果;贝叶斯算法本身有一些优化工作,如通过贝叶斯网络求解等.由于不是本文关注重点,考虑在将来的工作中继续研究.

随着机器学习技术的发展,对于数据量少、样本获取成本高的场景下,会有更多更好的算法被提出.实际上资源供给问题,关注点本身不在于算法实现,其关键问题是资源的使用预测.假如能够准确实现预测作业在某一配置下的运行时间,那么自然可以选择出最优的配置方案.同时,本文的研究工作可用于辅助资源调度、厂商资源定价、系统运维、可靠性保障等相关场景.

References:

[1] Gartner. https://www.gartner.com/webinar/3555919

[2] Venkataraman S, Yang Z, Franklin M, Recht B, Nsdi I. Ernest: Efficient performance prediction for large-scale advanced analytics. In: Proc. of the ACM Symp. on Networked Systems Design and Implementation. 2016.

[3] Alipourfard O, Liu HH, Chen J, Venkataraman S, Yu V, Zhang M. CherryPick: Adaptively unearthing the best cloud configurations for big data analytics. In: Proc. of the NSDI, Vol.2. 2017. 4.2−9.4.

[4] Herodotou H, Dong F, Babu S. No one (cluster) size fits all: Automatic cluster sizing for data-intensive analytics. In: Proc. of the ACM Symp. on Cloud Computing. 2011. 1−14.

[5] Shi J, Zou J, Lu J, Cao Z, Li S, Wang C. MRTuner: A toolkit to enable holistic optimization for mapreduce jobs. Proc. of the VLDB Endowment, 2014,7(13):1319−1330.

[6] Gartner. Technical Report 100 Data and Analytics Predictions Through 2021. https://www.gartner.com/events-na/data-analytics

[7] Lama P, ZhouX. AROMA: Automated resource allocation and configuration of mapreduce environment in the cloud. In: Proc. of the Int'l Conf. on Autonomic Computing. 2012. 63−72.

[8] Juve G, Deelman E. Wrangler: Virtual cluster provisioning for the cloud. In: Proc. of the Int'l Symp. on High Performance Distributed Computing. 2011. 277−278.

[9] Shkapsky A, Yang M, Interlandi M. Big data analytics with datalog queries on spark. In: Proc. of the 2016 Int'l Conf. on Management of Data (SIGMOD 2016). 2016.

[10] Wu YW, Wu H, Zhang WB, Xu YJ, Wei J, Zhong H. HW3C: A heuristic based workload classification and cloud configuration approach for big data analytics. In: Proc. of the 10th Asia-Pacific Symp. on Internetware (Internetware 2018). ACM, 2018. 10.

[11] Snoek J, Larochelle H, Adams RP. Practical Bayesian optimization of machinelearning algorithms. In: Proc. of the Int'l Conf. on Neural Information Processing Systems. 2012. 2951−2959.

[12] Ousterhout K, Canel C, Ratnasamy S, Shenker S. Monotasks: Architecting for performance clarity in data analytics frameworks. In: Proc. of the Symp. on Operating Systems Principles. 2017. 184−200.

[13] Verma A, Cherkasova L, Campbell RH. ARIA: Automatic resource inference and allocation for mapreduce environments. In: Proc. of the Int'l Conf. on Autonomic Computing (ICAC 2011). Karlsruhe, 2011. 235−244.

[14] Ferguson AD, Bodik P, Kandula S, Boutin E, Fonseca R. Jockey: Guaranteed job latency in data parallel clusters. In: Proc. of the European Conf. on Computer Systems (EUROSYS). 2012. 99−112.

[15] Cherkasova L. Performance modeling in mapreduce environments: Challenges and opportunities. In: Proc. of the ACM/SPEC Int'l Conf. on PERFORMANCE Engineering. 2011. 5−6.

[16] Yadwadkar NJ, Hariharan B, Gonzalez JE, *et al*. Selecting the best VM across multiple public clouds: A data-driven performance modeling approach. In: Proc. of the 2017 Symp. on Cloud Computing. ACM, 2017. 452−465.

[17] Xie D, Li F, Phillips JM. Distributed trajectory similarity search. In: Proc. of the PVIDB. 2017. 1478−1489.

[18] Keogh E. Exact indexing of dynamic time warping. In: Proc. of the 28th Int'l Conf. on Very Large Data Bases. 2002.

[19] Interlandi M, Tetali SD, Gulzar MA. Optimizing interactive development of data-intensive applications. In: Proc. of the ACM Symp. on Cloud Computing. 2016. 510−522.

[20] Civicioglu P. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Computers & Geosciences, 2012,46:229−247.

[21] Jiang J, Sun S, Sekar V, *et al*. Pytheas: Enabling datadriven quality of experience optimization using group-based exploration-exploitation. In: Proc. of the NSDI'17, 2017.

[22] Singh S, Chana I. A survey on resource scheduling in cloud computing: Issues and challenges. Journal of grid computing, 2016,14(2):217–264.

[23] Swersky K, Snoek J, Adams RP. Multi-Task Bayesian optimization. In: Proc. of the Advances in Neural Information Processing Systems. 2013. 2004–2012.

[24] Feurer M, Springenberg T, Hutter F. Initializing Bayesian hyperparameter optimization via meta-learning. In: Proc. of the AAAI Conf. on Artificial Intelligence. 2015.

[25] Bergstra J, Bengio Y. Algorithms for hyper-parameter optimization. In: Proc. of the Int'l Conf. on Neural Information Processing Systems. 2011. 2546–2554.

[26] Jiang YZ, Hao ZF, Zhang YS, Huang H, Wang YL, He HJ. Bayesian forecasting evolutionary algorithm. Ji Suan Ji Xue Bao/ Chinese Journal of Computers, 2014,37(8):1846−1858 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2014.01846]

[27] HiBench. https://github.com/intel-hadoop/HiBench

[28] Li M, Tan J, Wang Y, Zhang L, Salapura V. SparkBench: A comprehensive benchmarking suite for in memory data analytic platform Spark. In: Proc. of the ACM Int'l Conf. on Computing Frontiers. 2015. 53.

附中文参考文献:

[26] 姜允志,郝志峰,张宇山,等.贝叶斯预测型进化算法.计算机学报,2014,37(8):1846−1858. [doi: 10.3724/SP.J.1016.2014.01846]

吴悦文(1988－),男,湖南衡阳人,助理研究员,主要研究领域为分布式系统,云计算.

吴恒(1983－),男,博士,副研究员,CCF 专业会员,主要研究领域为分布式系统,云计算.

任杰(1993－),男,硕士,主要研究领域为分布式系统,云计算.

张文博(1976－),男,博士,研究员,博士生导师,CCF 专业会员,主要研究领域为分布式系统,云计算.

魏峻(1970－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为分布式系统,软件工程.

王焘(1982－),男,博士,副研究员,CCF 高级会员,主要研究领域为云计算,软件可靠性运行时监测.

钟华(1971－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为分布式系统,软件工程.