文章编号:1671-4598(2024)07-0274-06 DOI:10.16526/j.cnki.11-4762/tp.2024.07.040 中图分类号:TP393.4 文献标识码:A

基于 Spark 云计算的生物基因多序列比对方法

杨 波[1],陈洋广[2],徐胜超[1]
(1.广州华商学院 数据科学学院,广州 511300;2.广州华商学院 会计学院,广州 511300)

摘要:在生物基因多序列比对过程中,早期的方法仅计算了单一的 Spark 集群参数,导致算法的并行效果较差;为此,设计了基于 Spark 云计算的生物基因多序列比对方法;基于获得的生物遗传序列数据,对其进行了优化,并通过计算不同序列间的匹配度,对生物基因多序列比对任务进行动态规划;利用 Spark 云计算技术,构建 Spark 集群,并对多个 Spark 集群的参数进行计算;利用多种生物基因序列之间的相似性与差异性来选择最佳的匹配路径,在此基础上,建立多个生物基因序列比对的并行计算模型,并对其进行求解,得到对应的多个序列对比对的并行算法;实验结果表明:该方法具有更好的并行性,能够有效提高多序列比对的性能。

关键词:Spark 云计算;生物基因;生物信息学;基因多序列比对;并行算法

Multiple Sequence Alignment Method for Biological Genes Based on Spark Cloud Computing

YANG Bo[1], CHEN Yangguang[2], XU Shengchao[1]
(1. School of Data Science, Guangzhou HuaShang College, Guangzhou 511300, China;
2. School of Accountancy, Guangzhou HuaShang College, Guangzhou 511300, China)

Abstract: In the multi sequence alignment process of biological genes, early algorithms only calculate a single Spark cluster parameter, resulting in poor parallel performance of the algorithms. For this purpose, a multi sequence alignment parallel algorithm for biological genes based on Spark cloud computing was designed. The obtained biological genetic sequence data was optimized, and the dynamic planning of the biological gene multi sequence alignment was carried out by calculating the matching degree between different sequences. Spark cloud computing technology was used to build Spark clusters and calculate the parameters of multiple Spark clusters. By utilizing the similarities and differences between multiple biological gene sequences, the optimal matching path was selected. On this basis, the parallel computing model for multiple biological gene sequences was established and solved, and the corresponding parallel algorithm for aligning multiple sequences was obtained. Experimental results show that the algorithm has better parallelism and can effectively improve the performance of multiple sequence alignment.

Keywords: spark cloud computing; biological genes; bioinformatics; multiple gene sequences; parallel algorithms

0 引言

在生物信息学中,多序列比对是一个非常重要的工作,它对于理解基因的演化历程、发现基因之间的相似性和差异性以及进行基因组的组装等方面都具有重要的价值。然而,近年来,高通量测序技术的迅速发展为海量基因组数据的获得提供了可能。这些数据包含了来自不同个体、物种或种群的大量基因序列。比较并分析了这些基因的顺序,可以揭示基因组的演化、功能注释以及相关的遗传变异等重要信息。当前的并行算法仅计算了单一的集群参数,对所有计算节点应用相同的参数配置,但是,在某些情况下,这可能不适用于生物基因多序列比对,因为数据的特点和计算任务的复杂性可能会导致不同节点上的任务负载不均衡。目前现有的序列分析技术已经无法满足大规模数据处理的需求。因此,多序列比对算法并行化是当前生物信息学研究的热点。

文献[1]根据实际情况,构建二级 Hash 索引,快速作图与筛选出该基因的长序列,并对其进行哈希值的计算,实现先并行后对长序列的排列。该算法使用二级 Hash 的索引结构,可以减少内存占用,降低空间复杂度,但在长序列比对中,可能存在某些操作之间有依赖关系,必须按照顺序执行,意味着并行执行的能力受到限制,并行效果不佳。文献[2]优化了输入的基因序列数据,将其分割,利用 Spark 实现分布式计算。在混沌遗传算法的作用下,对基因序列进行比对和分析,再设计相应的并行策略,完成并行化处理。该算法基于云计算平台,可以根据实际需要动

收稿日期:2024-03-02; **修回日期**:2024-05-06。
基金项目:国家自然科学基金面上项目(61972444);广州华商学院校内科研导师制项目资助(2023HSDS34)。
作者简介:杨 波(1977-),男,硕士,副教授。
徐胜超(1980-),男,硕士,副教授。
引用格式:杨 波,陈洋广,徐胜超.基于 Spark 云计算的生物基因多序列比对方法[J].计算机测量与控制,2024,32(7):274-279,287.

态调整计算资源，适应不同规模和负载的基因序列比对任务，但存在计算资源需求高、技术支持限制和实验数据集限制等问题，从而导致其应用范围相对较窄。文献［3］利用读取映射器技术实现数据集，与序列数量成线性关系，能够在几秒钟内比对数万个完整的生物基因组。该算法采用一种分而治之的策略，能够高效地处理大规模的病毒基因组数据集，但进行比对时需要借助参考序列进行引导，导致并行性受限。

文献［4］提出了一种基于序列长度的高效多序列比对算法，该算法首先根据基因序列的长度将其分为若干段，然后对每个分段进行排序并与原序列比对。随着高通量测序技术的发展，序列比对变得越来越简单。文献［5］提出了一种基于时间窗的 DNA 序列分段方法，该方法首先将 DNA 序列按照其长度划分为若干个区间，然后对每个区间进行比较。文献［6］提出了一种基于启发式策略的多序列比对算法，该算法首先利用启发式策略对多个基因进行排序，然后将所有排序后的结果进行比较。但是，也出现了一些问题。在 DNA 序列比对算法中，由于 DNA 序列长度大、序列之间重叠率高，传统的比对算法过程中会花费大量时间。此外，由于 DNA 序列长时间不变，且每次比对只有一个结果，导致基因数据信息损失大，影响比对准确率。

云计算通过虚拟化的计算资源和服务，为用户提供了弹性、灵活、可靠和成本较低的计算环境，推动了数字化转型和创新应用的发展。该技术可以根据实际需求快速分配和释放计算和存储资源、并通过冗余和备份机制提供高可靠性的服务，支持大规模生物基因多序列比对的处理和分析。

云计算平台和 Spark 内存计算框架具备良好的并行化处理能力，可以将数据集切分为多个子集，在不同的节点或计算单元上进行并行处理，且对于多序列比对算法非常重要，通过对多个序列进行逐一比对，实现并行执行从而提高效率。因此，本文设计了基于 Spark 云计算的生物基因多序列比对并行算法。根据获取的生物基因数据，对生物基因多序列比对任务动态规划，以 Spark 云计算为基础，建立多个基因序列对比对的并行计算模型，并对该模型进行求解。

本文设计的算法将考虑基因数据的多样性和复杂性，采用适当的并行策略和优化方法，从而改善算法的精度与稳定性。通过获取的生物基因序列数据，计算不同序列间的匹配度，实现生物基因多序列比对任务的动态规划，具有良好的准确性。最后通过仿真实验验证了本文方法的优秀性能。

1　生物基因多序列比对任务动态规划

为实现对生物基因多序列比对并行算法的设计，需要先获取大量的生物基因多序列数据。当获得生物基因多重序列资料时，一般从美国生物技术信息中心（NCBI，national center for biotechnology information）、学术论文等多种渠道进行获取。获取的生物基因多序列数据类型众多，分别为脱氧核糖核酸（DNA，deoxyribo nucleic acid）、核糖核酸（RNA，ribonucleic acid）、蛋白质序列数据等[7-9]。将上述数据作为基础，首先，将高品质的资料过滤掉，在顺序资料中剔除重复与品质不佳的资料，然后格式化处理数据，为比对工作提供基础。同时，还需要对所获得的序列数据作进一步的优化，以进一步改善其质量[10-12]。其具体优化处理的过程如下所示：

$$S_k(i) = \max(\lambda_i, \lambda_j) \tag{1}$$

$$S_k'(i) = \frac{S_k(i) - S_u(i)}{S_\sigma(i)} \tag{2}$$

$$S_u(i) = \frac{1}{n}\sum_{k=1}^{n} S_k(i) \tag{3}$$

$$S_\sigma(i) = \frac{1}{n}\sum_{k=1}^{n} (S_k(i) - S_u(i))^2 \tag{4}$$

式中，$S_k(i)$ 表示筛选的基因序列数据，λ_i、λ_j 分别表示基因序列 i 和 j 的评估值，$S_k'(i)$ 表示优化处理后的基因序列数据，$S_u(i)$ 表示基因序列的均值，$S_\sigma(i)$ 表示基因序列的方差，n 表示基因序列数据的数量[13-14]。通过上述公式，对基因序列进行筛选，并对其进行优化，以进一步改善其质量，为后续的任务规划奠定基础。将上述基因序列数据的优化结果作为基础，结合实际的基因序列比对需求，对生物基因多序列比对任务进行动态规划。其动态规划过程如下所示：

$$S(i,j) = \max(G(i,j), H(i,j)) \tag{5}$$

$$G(i,j) = m_s + g_s \times n_g \tag{6}$$

$$H(i,j) = h_s + g_s \times n_g \tag{7}$$

式中，$S(i,j)$ 表示生物基因多序列比对任务动态规划结果，$G(i,j)$ 表示序列 i 与序列 j 之间的匹配度，$H(i,j)$ 表示序列 i 与序列 k 之间的不匹配度，m_s 表示不同序列的匹配值，g_s 表示不同序列间的间隙值，n_g 表示不同序列间的间隙权重，h_s 表示不同序列间的不匹配值[15-17]。通过上述公式，完成生物基因多序列比对任务动态规划。

2　基于 Spark 云计算的基因多序列比对并行计算模型构建

将上述生物基因多序列比对任务动态规划的结果作为基础，构造如图 1 所显示的 Spark 云计算平台。

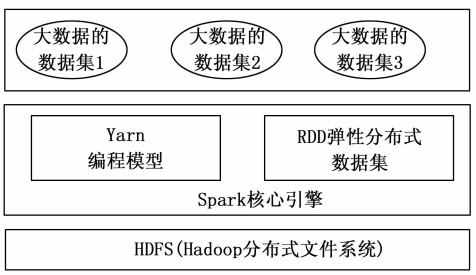

图 1　Spark 内存的框架处理大数据

Spark 的核心代码是以一种称为 Scala 的语言来完成，Scala 语言可以像操作本地集合对象一样轻松地操作分布式

数据集。提供比 Hadoop 更上层的 API，Spark 的程序设计代码比较精简，相同的功能只有 Hadoop 的 1/10 甚至更少。

选择了 Spark 框架，再结合 Yarn 规范，生物基因多序列比对就可以通过“Map（映射）”和“Reduce（归约）”这样两个操作来构成序列比对的（工作单元），用户只需提供自己的 Map 函数以及 Reduce 函数即可进行序列比对子任务的并行处理。

把 Spark 并行编程应用到生物基因多序列比对并行计算模型的构建中，能够降低计算压力，提高生物基因多序列的比对速度，并行计算模式的优化[21]。由此，基于 Spark 云计算的基因多序列比对并行计算模型的构建过程如图 2 所示。

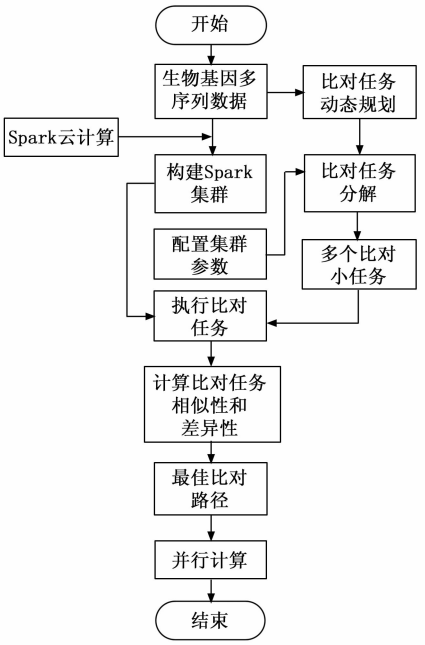

图 2 基于 Spark 云计算的基因多序列比对流程

如图 2 所示，在上述模型构建过程中，将上述获取的生物基因多序列数据作为基础，使用 Spark 云计算虚拟机来构建 Spark 集群。确保每台机器都有足够的内存、处理能力和网络连接，以满足 Spark 的运行要求。在每个工作节点上安装相同版本的 Spark，并配置好环境变量，确保各个节点之间可以互相通信，并且网络连接稳定。将生物基因数据加载到 Spark 集群中。根据数据的规模和计算任务的复杂性，将生物基因多序列比对任务划分为多个小的子任务。使用 Spark 提供的并行计算操作来处理划分好的子任务，收集并合并各个子任务的计算结果，然后在 Spark 集群中进行工作，这种序列比对子任务有专门的串行代码与运行库。在执行比对任务的过程中，需要分别计算生物基因多序列相似性和差异性，由此得到最佳的比对路径。在上述过程中，对 Spark 集群的参数进行计算的具体过程如下所示：

$$\begin{cases}\gamma_1 = \dfrac{e_d}{q} \\ \gamma_2 = \dfrac{o_f}{c_l}\end{cases} \tag{8}$$

式中，γ_1 代表 Spark 群集的记忆体参数，e_d 表示生物基因多序列数据的存储量[22]，q 表示动态规划的基因多序列比对任务数量，γ_2 代表 Spark 集群的节点数量，o_f 表示集群节点规模，c_l 表示集群的核心数量。根据上面的公式，实现 Spark 集群中所有参数的计算，根据计算结果，对比对任务进行分解，并计算非同比对任务间的异同。假设有两条 DNA 序列，它们有相同的碱基组成、相似的长度和相似的基因结构，这意味着它们具有较高的相似性。假设有两条蛋白质序列，它们的氨基酸序列有很多不同之处，则具有较大的差异性。其具体计算过程如下所示：

$$v(i,j) = 1 - d(i,j) \tag{9}$$

式中，$d(i,j)$ 表示计算出的基因序列数据的差异性，$v(i,j)$ 表示基因序列数据的相似性。通过上述公式，为基因序列数据求出最优排列路线，以此为基础，从中筛选出差异性最小的序列数据进行比对，由此筛选出最佳的比对路径，更快地完成对比对任务[23]。同时，通过 Spark 集群，多个子任务可以并行执行，进一步提高了比对的效率，由此完成并行计算模型的构建。至此，基于 Spark 云计算的生物基因多序列比对并行计算模型构建的设计完成。

3 Spark 云计算方法的优点分析

Spark 云计算方法的生物基因多序列比对算法的性能比 Map-Reduce 软件上要先进，前面我们分析了 Spark 云计算方法的的任务划分，其主要通过 Map-Reduce 编程模型来完成。根据 Hadoop2.0 的 Yarn 规范，Yarn 是 Hadoop1.0 的 Map-Reduce 规范的升级，保留了名空间服务器 NameNode 和数据服务器 DataNode。NameNode 用于元数据服务，而 DataNode 用于分散在一个集群中提供分布式文件存储服务。图 3 显示了把 Spark 云计算与 Yarn 框架结合的生物基因多序列比对的应用程序启动流程图。

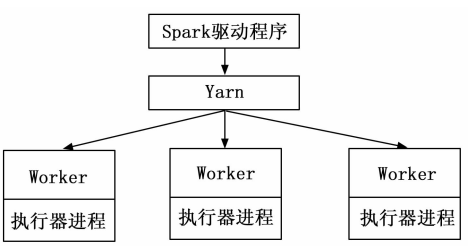

图 3 Spark 云计算与 Yarn 规范的融合

4 生物基因多序列比对并行算法的实现

将上述构建的生物基因多序列比对并行计算模型作为基础，结合实际情况，对并行计算模型进行求解。根据求解结果，生物基因多序列比对的对应并行算法。其具体求解过程如下所示：

$$f = m \times \vartheta \times \zeta \tag{10}$$

式中，f 表示并行计算模型求解结果的量化值，m 表示基因序列比对数量与总的基因序列数量的比对率，ϑ 表示并行参数，ζ 表示当前比对任务在并行模型中的权重值[24]。将上述计算结果作为基础，建立了多个生物基因序列匹配的并行

算法。它产生的一些并行算法的代码具体如表 1 所示。

表 1　并行算法的代码

行号	代码
1	import multiprocessing
2	import numpy as np
3	def align(seq1, seq2):
4	matches = np.sum(seq1== seq2)
5	return matches
6	def parallel_align(seqs):
7	pool = multiprocessing.Pool()
8	results = []
9	for i in range(len(seqs)):
10	for j in range(i+1, len(seqs)):
11	results.append(pool.apply_async(align,args= (seqs[i], seqs[j])))
12	pool.close()
13	pool.join()
14	return results
15	seqs= ["AGTACGAC", "AGTAGAC", "AGTAGACG", "AGTAGACGAC"]
16	results = parallel_align(seqs)
17	similarities =[result.get() for result in results]
18	total_similarity= sum(similarities) / len(similarities)
19	print("Average similarity:", total_similarity)

以上代码是一个使用多进程的并行序列比对过程。下面对代码中的各个部分进行解释：导入了 multiprocessing 库，该库提供了用于并行计算的多进程功能。定义了一个 align 函数，用于计算两个序列之间的相似性（即匹配的元素个数）。定义了一个 parallel _ align 函数，该函数接受一组序列（seqs）作为参数，使用多进程并行地计算序列之间的相似性。在 parallel _ align 函数中，首先创建了一个 multiprocessing.Pool 对象，用于管理并行计算任务的线程池。在嵌套的两个循环中，对所有的序列进行两两比较，并将计算任务提交给线程池进行并行计算。使用 pool.apply _ async 方法提交并行计算任务，该方法会返回一个 multiprocessing.pool.AsyncResult 对象，表示异步计算的结果。将这些结果存储在列表 results 中。当所有计算任务提交完成后，调用 pool.close()和 pool.join()方法来等待所有计算任务完成并关闭线程池。最后，在主程序中，定义了一组序列 seqs，调用 parallel _ align 函数并获取相似性计算的结果。将每个计算任务的结果从 AsyncResult 对象中使用 result.get () 方法提取出来，并将其存储在 similarities 列表中。计算所有相似性的平均值，并打印输出。

根据上述并行计算模型的求解结果，先定义一个并行计算主控函数和比对函数，然后将若干条基因信息输入进去，对其进行优化，由此对上述数据进行并行计算，将并行计算结果进行汇总，并计算其平均相似度，由此保证比对效果。利用上述代码，完成对并行算法的设计，提高了基因多序列比对的计算效率。

5　仿真实验与性能分析

5.1　Spark 集群的搭建与编程

为验证本文设计的基于 Spark 云计算的生物基因多序列比对并行算法在实际应用中的效果，进行实验测试。实验中，本项目搭建 Spark 集群的运行环境，并开展相关的实验验证。具体 Spark 集群网络设备拓扑图如图 4 所示。

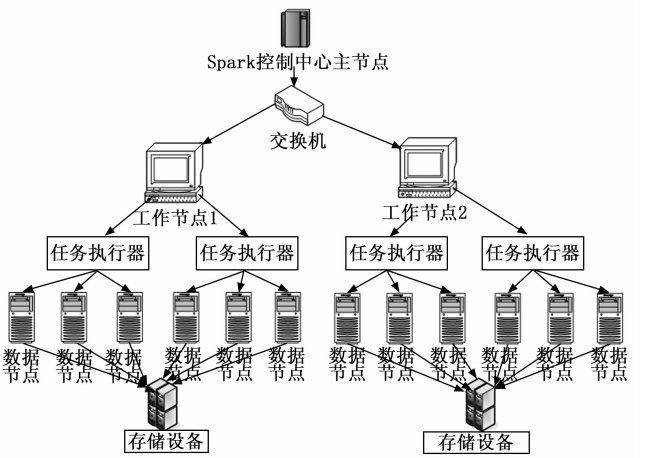

图 4　Spark 集群网络设备拓扑图

图 4 中主节点通过交换机与工作节点相连，每个工作节点上有多个任务执行器，用于并行执行 Spark 作业。实验的 Spark 集群的 Hadoop 部分由名空间节点（NameNode）和数据节点（DataNode）构成，其他的 Yarn 部分则由主控节点 ResourceManager 与 NodeManager 组成。通过 30 台 PC 机搭建 Spark 云计算集群，PC 配置参数为：8 核、2.80 GHz 的 CPU，16 GB 的内存，1 TB 的硬盘，RedHatLinux 操作系统，这些节点之间通过局域网内的 2 台 1000M24 口 TP-Link 交换机连接起来。30 台 PC 机都安装 RedHatLinux 操作系统，得到软件配置，如表 2 所示。

表 2　Spark 云计算平台节点配置信息

节点 IP 地址	角色	操作系统	运行进程
192.168.25.1	Master	RedHatLinux	NameNode, ResourceManager
192.168.25.2	Worker	RedHatLinux	DataNode,NodeManager, Worker
192.168.25.3	Worker	RedHatLinux	DataNode,NodeManager, Worker
192.168.25.4	Worker	RedHatLinux	DataNode,NodeManager, Worker
192.168.25.5 ～30	Worker	RedHatLinux	DataNode,NodeManager, Worker

在实验过程中值得注意的是 Spark 对于资源管理与作业调度可以使用本地模式、Standalone 独立模式、Apache Mesos 模式及 Hadoop Yarn 模式来实现，图 3 就是一种 Hadoop Yarn 模式。它支持多种 Scheduler 和 Executor，用 Yarn 模式的实现也就非常容易。

Spark 由 Master（类似于 Map-Reduce 的 JobTracker）

和 Workers（类似于 Map-Reduce 的 TaskTracker 工作节点）组成。

5.2 生物基因动态规划数据的选取

本文动态规划是以基因序列数据结果作为基础，结合实际的基因序列比对需求，对生物基因多序列比对任务进行动态规划。文中表 3 生物基因序列数据集，为动态规划参数选择。实验中，从美国生物技术信息中心（NCBI，national center for biotechnology information）中获得生物基因多序列数据，为保证实验结果的可靠性，选取了较大规模的生物基因数据序列集作为数据库，并从中随机抽取多组基因序列数据。其具体抽取结果如表 3 所示。

表 3 生物基因序列数据集

数据集编号	数据类型	序列的条数
U1 数据集	蛋白质序列数据	25 634
U2 数据集	蛋白质序列数据	12 596
U3 数据集	蛋白质序列数据	26 947
U4 数据集	蛋白质序列数据	266 358
U5 数据集	蛋白质序列数据	158 512
N1 数据集	核苷酸序列数据	412 562
N2 数据集	核苷酸序列数据	233 695
N3 数据集	核苷酸序列数据	152 745
N4 数据集	核苷酸序列数据	236 142
N5 数据集	核苷酸序列数据	59 642

如表 3 所示，将上述抽取的生物基因序列数据集作为本次实验数据，进行实验测试。实验中，为了更好地分析生物基因序列数据集的实际情况，在此基础上，进一步研究这些基因在数据集上的分布规律。在本次实验中，将展示部分基因数据的序列分布情况。其具体分布情况如图 5 所示。

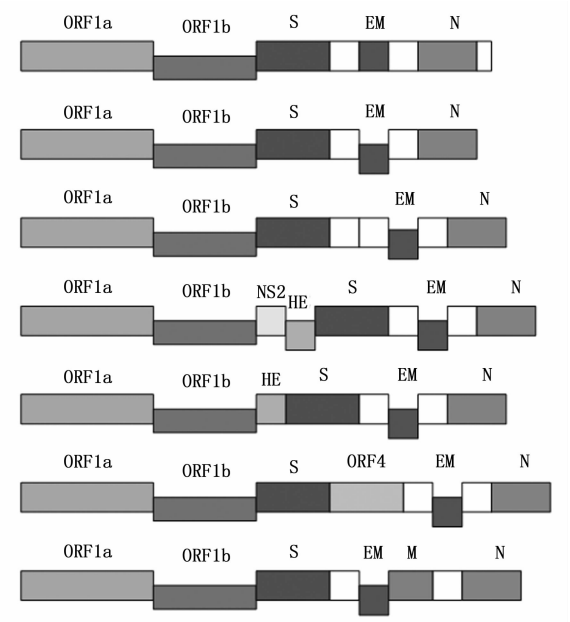

图 5 部分基因数据的序列分布情况

如图 5 所示，上述为生物基因的部分序列，将上述基因序列数据集作为基础，进行后续实验测试。为提高本次实验结果的可靠性，将文献［1］算法和文献［2］算法与本文设计的并行算法进行对比。本文设计的基于 Spark 云计算的生物基因多序列比对并行算法为算法 1，文献［1］算法为算法 2，文献［2］算法为算法 3。其中，文献［1］算法设置序列的碱基位数为 15,；匹配得分为 2；空隙罚分为 3；常规错误的拓展罚分为 1。文献［2］算法基于 Need-leman-Wunsch 建立遗传编码矩阵，并分别将移动规律用 R、B 和 D 表示，设染色体的编码为 BBBBBRBBBB。将 3 种算法的试验检测结果比较与分析，验证 3 种算法的应用性能。

5.3 泊松估计值测试

为了检验这 3 个算法的实用性，以算法的泊松估计值为评价指标，对比 3 种算法的性能。泊松估计值计算公式如下：

$$P(X=k)=\frac{\lambda^k e^{-\lambda}}{k} \tag{11}$$

式中，k 表示事件发生的次数，λ 表示事件发生的平均次数。

泊松估计值越大，说明该方法具有较好的并行性。实验中，3 种不同的算法在基因序列中应用，统计其并行结果的泊松估计值，如图 6 所示。

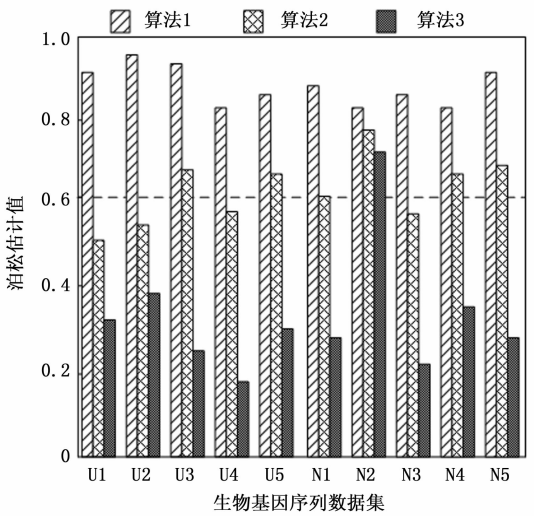

图 6 3 种算法的泊松估计值

从图 6 可以看出，在上面的试验结果中，和标准值相比，算法 1 的泊松估计值均在标准值之上，且数值较高。在算法 2 的并行结果中，存在部分数据集的泊松估计值数值较小，在标准值之下。在算法 3 的并行结果中，仅存在数据集 N2 的泊松估计值在标准值之上，其余均在标准值之下，算法 3 的并行效果较差。这是因为本文利用 Spark 云计算平台同时处理大规模的数据和任务，对生物基因多序列比对，并行算法可以将整个比对过程分解为多个子任务，并在 Spark 集群中并行执行，从而导致较高的事件发生次数，由此可知，文中所提出的算法具有更好的并行性能。

5.4 多序列比对时间测试

为了更好地验证这 3 个算法的并行性能，分析 3 种算法多序列比对所使用的时间，具体比对结果如表 4 所示。

表 4　3 种算法多序列比对时间

实验数据集	3 种算法多序列比对时间/s		
	算法 1	算法 2	算法 3
U1 数据集	5	15	18
U2 数据集	6	16	17
U3 数据集	8	14	17
U4 数据集	4	16	18
U5 数据集	5	15	16
N1 数据集	6	17	18
N2 数据集	7	13	15
N3 数据集	7	12	18
N4 数据集	5	14	17
N5 数据集	6	15	16

如表 4 所示，算法 1 的多序列比对时间，均低于算法 2 和算法 3，其能够在并行计算中取得较高的性能。这是因为本文构建生物基因多序列比对并行计算模型，对并行计算模型进行求解，生成相应的生物基因多序列比对并行算法，提高了并行算法的并行性能。

综上所述，本文设计的基于 Spark 云计算的生物基因多序列比对并行算法在实际应用中具有较高的泊松估计值和较低地多序列比对时间，并行效果较好，能够有效提高生物基因多序列比对并行的性能，促进相关行业的发展。

6　结束语

本文提出一种基于 Spark 云计算的生物基因多序列比对方法。该方法可以有效地处理大规模的生物基因序列数据集，并显著缩短比对时间，提高算法的并行性能，这样，就可以得到更加精确的对比结果。同时，本文设计的算法在处理更大规模生物基因数据集时，也能够取得较好的效果，这为大规模数据的处理提供了有力的支持，同时，还将为生物信息学领域的相关研究提供有效的分析手段，但本文没有涉及到对生物基因多序列比对算法的优化和改进。未来的工作可以着眼于改进比对算法，提高其效率和准确性，以适应更大规模和复杂的基因序列比对任务。

参考文献：

[1] 潘　登，钟　诚. 二级 Hash 全局和局部索引筛选的长序列比对并行算法 [J]. 小型微型计算机系统，2022，43 (9)：1999 - 2004.

[2] 刘清雪，罗宇航. 基于 Spark 云计算及混沌遗传的基因序列比对研究与实现 [J]. 软件，2021，42 (3)：40 - 42.

[3] MOSHIRI N. ViralMSA：Massively scalable reference-guided multiple sequence alignment of viral genomes [J]. Bioinformatics，2021，37 (5)：714 - 716.

[4] WANG C，LIU Z，CHEN Z，et al. The establishment of reference sequence for SARS-CoV-2 and variation analysis [J]. Journal of Medical Virology，2020，92 (6)：667 - 674.

[5] CHOWDHURY R，BOUATTA N，BISWAS S，et al. Single-sequence protein structure prediction using a language model and deep learning [J]. Nature Biotechnology，2022，40 (11)：1617 - 1623.

[6] MADEIRA F，PEARCE M，TIVEY A R N，et al. Search and sequence analysis tools services from EMBL-EBI in 2022 [J]. Nucleic Acids Research，2022，50 (1)：276 - 279.

[7] 周　翔，赵仁生，崔艺璇，等. SARS-CoV-2 病毒全基因组序列比对及进化分析 [J]. 云南民族大学学报（自然科学版），2022，31 (2)：176 - 185.

[8] KIM M W，LEE D，KIM M M. Characteristics of patients with accommodative esotropia who need glasses for stable alignment after myopic shift [J]. Journal of the korean ophthalmological society，2021，62 (8)：1116 - 1122.

[9] SUBLETT C，TOVAR J. Community college career and technical education and labor market projections：a national study of alignment [J]. Community College Review，2021，49 (2)：177 - 201.

[10] MOK PL，KOH AE，FARHANA A，et al. Computational drug screening against the SARS-CoV-2 Saudi Arabia isolates through a multiple-sequence alignment approach [J]. Saudi Journal of Biological Sciences，2021，28 (4)：2502 - 2509.

[11] SHANG Y，RAN H. A parallel subgrid stabilized algorithm for incompressible flows with nonlinear slip boundary conditions [J]. Discrete and Continuous Dynamical Systems-B，2023，28 (7)：4087 - 4107.

[12] MULUMUDY R，MITRA P，PAL A. Modularity-based parallel protein design algorithm with an implementation using shared memory programming [J]. Proteins：Structure，Function，and Bioinformatics，2022，90 (3)：658 - 669.

[13] MAO Y，DENG Q，CHEN Z. Parallel association rules incremental mining algorithm based on information entropy and genetic algorithm [J]. Journal on Communications，2021，42 (5)：122 - 136.

[14] YANG X，GUI H，FU C F-X domain predictive filtering parallel algorithm based on compute unified device architecture [J]. Journal of Computer Applications，2021，41 (2)：486 - 491.

[15] TAN X，WANG Q，FENG J，et al. Fast modeling of gravity gradients from topographic surface data using GPU parallel algorithm [J]. Geodesy and Geodynamics，2021，12 (4)：288 - 297.

[16] HUANG R，ZHAO Y，YU T，et al. a Parallel two-tier blocked jacobi svd algorithm on gpu [J]. Journal on Numerical Methods and Computer Applications，2022，43 (4)：380 - 399.

[17] LI YE，MAO YIMIN，CHEN ZHIGANG. Winograd-based parallel deep convolutional neural network optimization algorithm [J]. Information and Control，2023，52 (4)：466 - 482.

[18] JIAO Y，DENG X，LI M，et al. Balancing of parallel U-shaped assembly lines with a heuristic algorithm based on bidirectional priority values：[J]. Concurrent Engineering，2022，30 (1)：80 - 92.

（下转第 287 页）