第 3 章　基于启发式的基因组序列比对大数据分发模型

为了解决大数据集的全比较问题，将大量的数据文件分发到分布式计算系统中会对整体计算性能产生很大的影响。本章提出了一个启发式的数据分配策略来处理同构分布计算系统中的全比较问题；从构建原则出发，从不同的角度讨论了为全比较问题分发数据文件所面临的挑战；在分析数据分发问题的基础上，提出一种基于贪婪思想的启发式数据分发算法。该数据分发策略不仅能够节省存储空间和数据分发时间，而且可以实现全比较问题的所有比较任务的负载均衡和良好的数据本地化。基于数据分发的结果，我们还提出了一种静态任务调度策略和数据分发策略，实现了让系统以静态负载均衡的方式分配比较任务。最后，不同的实验验证了该数据分发策略在同构分布计算系统中的有效性。

3.1　文件分发模型的构建

3.1.1　构建原则

在分布式系统中进行全比较计算的典型场景描述如图 3.1 所示。一般来说，数据管理器应该首先管理所有数据并将其分发给计算节点。然后，计算任务由作业追踪器生成并分配给计算节点。最后，任务执行者执行计算任务来处理相关的数据集。

从图 3.1 的工作流程可以看出，要有效地解决全比较问题，需要改进数据分配和计算阶段。

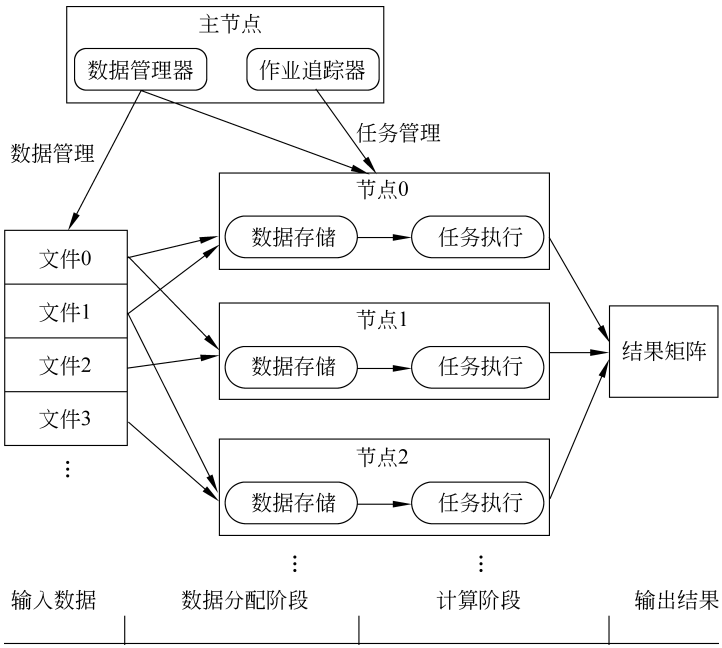

图 3.1　分布式环境下全比较计算工作流程

数据本地化和数据分发是分布式环境下大数据处理的整体计算性能的两个关键因素。

数据本地化是处理大数据问题的基本原则。这意味着把计算操作分配给拥有计算任务所需数据的计算节点通常可以获得更高的计算效率。由于繁重的网络通信和数据传输,需要访问远程数据集的计算任务可能非常低效。

当所有计算节点都分配了与其处理能力相匹配的适当数量的计算任务时,大数据问题的分布式计算效率更高。在这种情况下,系统的所有计算能力都可以得到有效利用。

因此,用于全比较问题的分布式计算的数据分发策略应能够满足两个条件:执行比较任务所需的数据具有良好数据本地化;通过对具有良好数据本地化的任务进行调整来平衡比较任务的负载。

3.1.2　当前不足

全量分发和使用 Hadoop 数据策略是解决全比较问题时广泛使用的两种数据分发策略。在这一部分中，我们将深入讨论这两种数据分发策略的不足。

1. 全量分发策略问题

全量分发是指将所有数据文件存储到系统中的每个计算节点，这是分布式环境下解决数据分发问题的一种简单方法。许多解决方案[1-2]都选择让每个计算节点存储所有必需的数据文件。使每个数据文件具有最大副本数，并且将任何比较任务分配给任何计算节点，可以实现高可靠性，但是这种数据分发策略不适合处理大量的数据文件。

（1）巨大的存储量需求。将所有数据文件存储到全部的计算节点是由集中式计算解决方案所产生的策略。采用这种数据分发策略的分布式系统设计简单，因为其避免了划分数据集带来的问题，使得应用系统只需考虑调度计算任务。将所有数据文件放在任何地方的策略，由于存储使用量可能太大，使得应用系统无法处理大数据问题。例如，将 M 个数据文件放在 N 个计算节点，为了将所有数据文件存储到每个计算节点，系统中需要存储总共 $M \times N$ 个数据文件。

（2）典型的全比较问题通常需要处理大量的数据文件。例如，在 Moretti 等[3]提出的一个实验中，在 100～200 台不同的机器上，对近 60000 个数据文件进行了配对比较。在这种情况下，至少需要在系统中存储 600 万个数据文件，这将花费大量的数据分发时间，并且将产生巨大的存储使用量。

（3）存储空间的浪费。此外，以这种方式存储数据可以使大多数数据文件永远不被使用。对于大型数据集的全比较问题来说，仅仅为了实现高可靠性而存储数据文件，总体来说可能是一个漫长过程，并且将造成存储资源的浪费。

以 9 个数据文件和 3 个计算节点为例，其总共需要执行 36 个比较任务。图 3.2 显示了这 9 个数据文件的可能数据分发策略。如图 3.2 所示，分发数据文件可以使每个计算节点分配 12 个不同的比较任务。因此，所有的比较任务都可以在不进一步移动数据的情况下完成，并且还可以实现系统负载均衡。

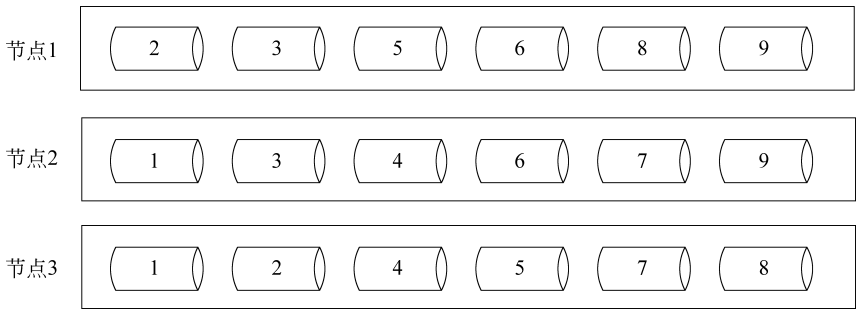

图 3.2　在 3 个计算节点分布式系统中处理 9 个数据文件的情况

在这个例子中，对于 9 个数据文件，只需要在每个计算节点上存储 6 个数据文件。这意味着，与向每个计算节点存储 9 个数据文件相比，这将造成 30％的存储浪费。当数据文件的数量和计算节点的数量增加时，结果可能会更不理想。

2. Hadoop 数据策略问题

Hadoop 数据策略已经被广泛应用于解决多对多比较问题的研究中。Hadoop 分布式文件系统（HDFS）提供了一种分发和存储大数据集的策略。在 HDFS 数据策略中，数据项在所有存储节点之间以固定数量的副本被随机分发。尽管 HDFS 中的多个数据副本增强了存储的可靠性，但由于存在数据本地化情况差、任务负载不均衡和巨大的数据分发解空间等问题，对全比较问题来说效率低下[4]。

下面将讨论使用 Hadoop 数据策略解决全比较问题的不足。

首先，HDFS 数据策略中数据本地化差。Hadoop 的数据策略是为了在提高数据可靠性与低读写成本之间进行权衡而设计的。从这个设计角度来看，它没有考虑后续比较任务的数据需求。

例如，考虑一个有 6 个数据项和 4 个计算节点的场景。Hadoop 数据策略的一个可能的解决方案Ⅰ如图 3.3 所示。从图 3.3 可以看出，尽管 6 个数据项中的每一个都有两个副本，但是没有包含某些比较任务(1,3)、(1,4)、(2,6)、(3,5)、(4,5)所需的所有数据的计算节点，这表明这些比较任务的数据本地化不好。在这种情况下，无法避免节点之间通过网络通信在执行比较任务时产生数据文件的调度。这将导致运行时存储成本的增加，也会影响全比较问题的整体计算性能。当计算规模变大时，这个问题变得更不理想。

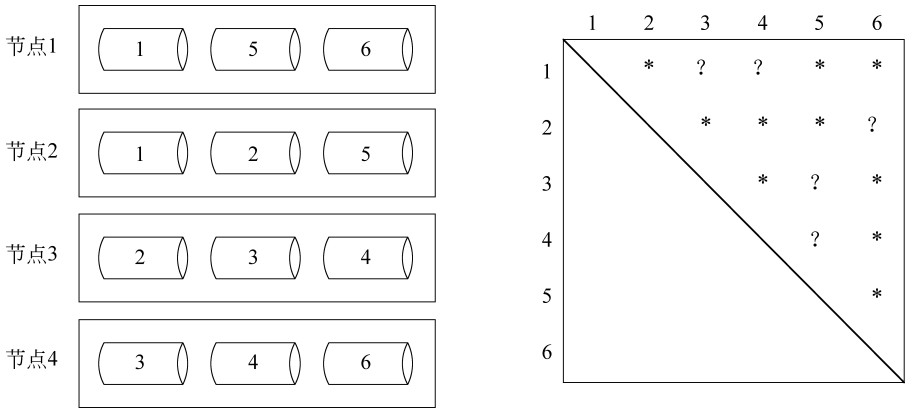

图 3.3　Hadoop 数据策略的一个可能的解决方案Ⅰ

其次，HDFS 数据策略导致的不平衡任务负载。要讨论 HDFS 数据策略导致的不平衡任务负载，可以考虑一个包含 4 个数据项和 3 个计算节点的场景。图 3.4 描述了可能的 Hadoop 数据分发解决方案，将 6 个比较任务分配给 3 个计算节点。HDFS 数据策略无法确保均衡的比较任务分配，如果需要实现任务的均衡分配，需要为 3 个计算节点中的每个节点分配 2 个比较任务。

再次，在 HDFS 中增加文件副本的效率低下。在 HDFS 数据策略中，用

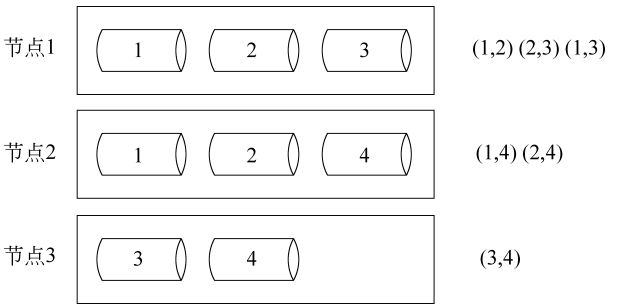

图 3.4　Hadoop 数据策略的一个可能的解决方案Ⅱ

户能够手动设置数据副本的数量。但是没有关于如何设置这个数字的指导，因此用户倾向于使用默认的数字 3。一旦设置了这个数字，它就成为一个常数，与要使用的机器数量和要分发的数据文件数量无关。此外，数据文件副本的位置是随机确定的。这会导致数据文件的数据本地化差，并会使得分布式计算性能受到影响。我们曾认为，进一步增加数据副本数目可能会有所帮助。然而，正如稍后的实验所证明的，情况并非如此。除非将数据文件复制到全部节点，否则在 HDFS 中增加数据副本数目并不能显著提高数据本地化，但由于网络通信对运行时数据移动的需求增加，因此会导致显著的计算时间的增加。

图 3.5 为 6 个数据项和 8 个计算节点的 HDFS 解决方案。从图 3.5 中可以看出，即使数据文件副本的数量增加到 4，仍然存在较差的数据本地化和不平衡任务负载。如果运行时不采用远程调度数据文件的方式，则无法完成比较任务(1,2)、(3,4)、(5,6)。因此数据文件必须在运行时被移动，以完成全比较问题的分布式计算。

最后，HDFS 数据分发的巨大解空间。将比较任务分配给计算节点的问题可以看作组合数学中的一个经典问题：将 M 个对象放入 N 个框中。这种成对数据分发问题具有以下特点。

(1) 所有比较任务都是可区分的。对于全比较问题，每个比较任务都是

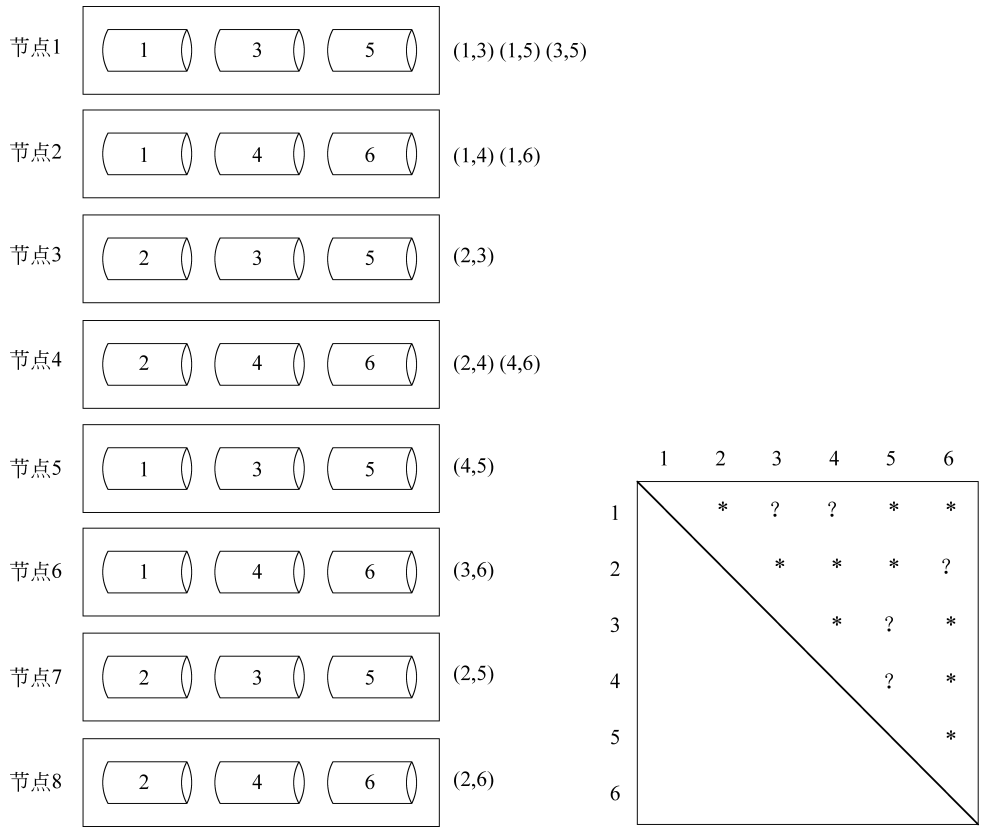

图 3.5　Hadoop 数据策略的一个可能的解决方案Ⅲ

不同的，并且处理不同的数据对。在分发相关数据集时应考虑此特性。

（2）在同构分布计算系统中，所有计算节点都是不可区分的。在本书中，假设计算节点具有相同的处理能力和存储空间，以至于让这些计算节点可以被视为不可区分。

考虑前面提到的数据本地化需求。分发数据对还意味着分配相关的比较任务。我们的目标是将 Q 个可区分的比较任务分配给 N 个不可区分的计算节点。在组合数学中，可行解的总数可以用斯特林数 $S_t=(Q,N)$ 表示[5]。第二类斯特林数表示将一组具有 Q 个可区分的元素划分到 N 个非空子集的方法总数。根据第二类斯特林数的基本性质，如果 Q 或 N 中的一个或两个

都为 0，则：

$$S_t\{0,0\}=1,\quad S_t\{Q,0\}=0,\quad S_t\{0,N\}=0$$

如果 $Q\geqslant 1$ 且 $N\geqslant 1$，则：

$$S_t(Q,N)=NS_t(Q-1,N)+S_t(Q-1,N-1)$$

让我们考虑一下斯特林数的一些特殊情况。我们在之前的研究中讨论了 $N=2$ 的一个特殊情况[6]。对于 3 个计算节点（$N=3$）的另一种特殊情况，则：

$$S_t(Q,3)=\frac{3^Q-3\times 2^Q+3}{6} \tag{3-1}$$

$S_t(Q,3)$ 解空间的增长趋势如图 3.6 所示。图 3.6 所示的趋势表明，在实际操作中，即使对于 3 个计算节点（即 $N=3$）的非常简单的情况，也无法访问太多可能的分发解决方案。这意味着，我们通常不可能评估所有可能的解决方案。因此，有必要开发数据分发的启发式解决方案，以便在合理的时间内找到最佳答案。

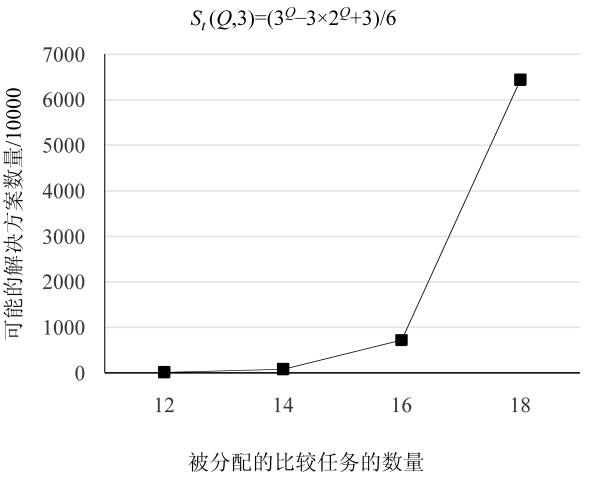

图 3.6　$S_t(Q,3)$解空间的增长趋势

3.2　文件分发算法设计

本节介绍数据分发策略的设计要求，从总体考虑和假设开始，然后是降低存储使用率和提高计算性能，最终讨论数据分发的特殊情况。

3.2.1　概述和假设

为了使用图 3.1 所示的工作流程来解决全比较问题，需要开发数据分发策略来指导所有数据文件的分配。在接下来的场景中，数据分发策略将首先生成数据分发的解决方案，然后根据提供的解决方案部署所有的数据文件。设计数据分发策略需要考虑以下几个方面。

（1）分布式系统的存储使用。对于大数据集的全比较问题，在所有计算节点之间分发数据集不仅要考虑每个节点在其容量内的存储空间使用情况，还要将用于数据分发的总时间限定在可接受的水平。

（2）比较计算的性能。对于分布式计算来说，分配比较任务以充分利用系统中所有可用的计算能力是很重要的。此外，对于具体的比较任务，执行任务所需的数据具有良好的数据本地化将有助于提高计算性能。因此，为了提高比较计算的性能，需要设计数据分发策略来分配数据项。

为设计同构分布计算系统下的数据分发策略，本章进行了以下假设：分布式系统中所有的计算节点具有相同的处理能力和存储能力；所有的数据文件大小相同；所有比较任务的执行时间相同。

这些假设不一定是现实的，主要是为了便于对本章中启发式数据分发策略的理解。因此，在保持数据分发策略的基本原则的基础上，可以放宽这些限制以实现更现实的开发。

下面分析前面提到的分布式系统的存储使用需求和比较计算的性能。

3.2.2 降低存储使用

许多因素造成了数据分发时间的增加，比如给定网络带宽、网络拓扑结构和数据项的大小。从上述假设出发，数据分发的总时间与要分发的数据项的数量成正比。数据分发总时间 T_{data} 可以表示为

$$T_{\text{data}} \propto \sum_{i=1}^{N}(|D_i|) \tag{3-2}$$

除此之外，如前所述，每个计算节点的存储空间使用必须在其限制范围内。根据假设，将所有的数据集均匀分布在分布式系统中，这是可以实现的。

因此，在存储使用方面，数据分发策略旨在达到两个目的：减少分布式系统中存储的数据文件总数；减少每个计算节点上存储的数据文件数量。

首先，我们考虑数据分发时间和存储限制。$|D_i|$ 表示分配给计算节点 i 的文件的数量，则数据分发策略期望最大限度地减少 $|D_1|,|D_2|,\cdots,|D_N|$ 的最大值，即

$$\text{Minimize } \max\{|D_1|,|D_2|,\cdots,|D_N|\} \tag{3-3}$$

选择最小化计算节点中数据文件的最大数量有以下好处。该目标使所有计算节点具有相似数量的数据文件。在理想情况下，节点之间的数据文件数量的差异最多只有一个，这意味着分布式系统的数据存储使用平衡。考虑到可以执行的比较任务的数量与计算节点中存储的数据文件的数量成正比，此目标还使得所有计算节点承担相似数量的比较任务。

3.2.3 提高计算性能

在全比较问题的分布式计算中，执行计算任务的总计算时间由最后完成的计算节点决定。要完成每个比较任务，相应的计算节点必须访问和处理所需的数据项。

令 K、$T_{\text{comparion}(i)}$ 和 $T_{\text{accessdata}(i)}$ 分别表示分配给最后完成的计算节点的比较任务数、任务 i 的比较操作时间和访问任务 i 所需数据的时间。然后，全比

较问题的总已用计算时间 T_{task}表示为

$$T_{\text{task}}=\sum_{i=1}^{K}(T_{\text{comparison}(i)}+T_{\text{accessdata}(i)}) \tag{3-4}$$

由于假设所有比较任务具有相同的比较时间 C，因此计算时间 T_{task}可以简化为

$$T_{\text{task}}=CK+\sum_{i=1}^{K}T_{\text{accessdata}(i)} \tag{3-5}$$

从式(3-5)可以看出，我们的数据分发策略通过满足两个约束来最小化计算时间 T_{task}：计算节点上比较任务的负载均衡，以及所有成对比较任务的良好数据本地化。

对于负载均衡，分配给最后完成比较任务的计算节点，各个节点上的比较任务的最大数量 K 可以被最小化。让 T_i 表示计算节点 i 执行的成对比较任务的数量。对于包含 N 个计算节点和 M 个数据文件的分布式系统，需要为分布式系统中的计算节点分配 $M(M-1)/2$ 个比较任务的总数。

将 K 的值最小化可以表示为

$$\forall T_i\in\{T_1,T_2,\cdots,T_N\}\quad\left(T_i\leqslant\left\lceil\frac{M(M-1)}{2N}\right\rceil\right) \tag{3-6}$$

其中$\lceil\cdot\rceil$是向上取整函数。

良好的数据本地化也可以用数学公式表示。如果比较任务所需的所有数据都存储在执行该任务的节点上，则该任务不需要通过网络通信远程访问数据。良好的数据本地化意味着 $T_{\text{accessdata}(i)}$的最小值，其最小可能值为 0。设 $C(x,y)$、T、T_i、D_i 分别表示：数据 x 和数据 y 的比较任务、所有比较任务的集合、计算节点 i 执行的比较任务集合、存储在计算节点 i 中的数据集合。所有比较任务的良好的数据本地化可以表示为

$$\forall C(x,y)\in T,\quad\exists i\in\{1,2,\cdots,N\}\quad(x\in D_i\cap y\in D_i\cap C(x,y)\in T_i) \tag{3-7}$$

换句话说，对于比较任务 $C(x,y)$必须至少有一个节点 i 能够仅使用本

地数据完成执行。

3.2.4 数据分发过程优化

如前所述，考虑到存储使用和计算性能，数据分发策略预计将满足式(3-3)中的目标要求，并满足式(3-6)和式(3-7)中的约束。满足式(3-3)中的目标要求，可以减少所有计算节点的存储使用，从而减少分发所有数据集所花费的时间(T_{data})。满足式(3-6)和式(3-7)中的约束意味着可以最小化总体比较时间 T_{task}。因此，有效地减少了数据分发和任务执行所需的总时间：

$$T_{\text{total}} = T_{\text{data}} + T_{\text{task}} \tag{3-8}$$

数据分发问题能够表示成一个约束优化问题：

$$\text{Minimize } \max\{|D_1|,|D_2|,\cdots,|D_N|\}$$

$$\text{s.t.}\begin{cases} \forall T_i \in \{T_1,T_2,\cdots,T_N\} \quad \left(T_i \leqslant \left\lceil \dfrac{M(M-1)}{2N} \right\rceil\right) \\ \forall C(x,y) \in T \quad (\exists i \in \{1,2,\cdots,N\}, x \in D_i \cap y \in D_i \cap C(x,y) \in T_i) \end{cases} \tag{3-9}$$

如前所述，大量的数据组合和相关的比较任务使得上述数据分发优化问题在实际应用中难以解决。

3.2.5 理论结果

对 $\max\{|D_1|,|D_2|,\cdots,|D_N|\}$进行了理论分析，得到了一个下界 $d_{\max}$，并对数据可用性的系统可靠性项进行了深入的研究。这些理论结果归纳为两个定理。

定理 1：对式(3-9)中定义的约束优化问题，将 M 个数据文件分配到 N 个计算节点，一个解决方案是将 $\max\{|D_1|,|D_2|,\cdots,|D_N|\}$的下界$d_{\max}$设置为

$$\max\{|D_1|,|D_2|,\cdots,|D_N|\} \geqslant \frac{1}{2}\left(1+\frac{\sqrt{4M^2-4M+N}}{\sqrt{N}}\right) \overset{\text{def}}{=\!=} d_{\max} \tag{3-10}$$

在下一部分中，利用组合数学和图论的两种方法证明定理 1。

证明过程如下：对于 M 个数据文件，一个全比较问题中比较任务的总数为 $M(M-1)/2$。考虑以下极端情况：如果每个计算节点分配的比较任务不超过$\lceil M(M-1)/(2N)\rceil$(式(3-6))，则至少需要多少数据项才能完成所有比较？由于每个比较任务需要两个不同的数据项，如式(3-7)所示，我们有以下关系：

$$\binom{\max\{|D_1|,|D_2|,\cdots,|D_N|\}}{2}=\left\lceil\frac{M(M-1)}{2N}\right\rceil \tag{3-11}$$

解方程得到式(3-10)中的结果。

完全图 $G=(V,E)$有 M 个顶点，共有 $M(M-1)/2$ 条边。考虑到一个特殊情况，图中所有边的权重被设置为 1，这意味着每个边 $e\in E$ 被一个且只有一个子图覆盖。因此，导出子图的最大阶数应该是理论最小值。

在这种情况下，具有最大阶的子图是具有$\lceil M(M-1)/(2N)\rceil$条边的完全图。如果我们令 $v=\max\limits_{i=1,2,\cdots,N}|V_i|$表示此子图中的顶点数，$U=\lceil M(M-1)/(2N)\rceil$，基于顶点和边的关系，则有

$$\binom{v}{2}=\boldsymbol{U} \tag{3-12}$$

在解方程之后，可以得到

$$\max_{i=1,2,\cdots,N}|V_i|\geqslant\frac{1}{2}\left(1+\frac{\sqrt{4M^2-4M+N}}{\sqrt{N}}\right) \tag{3-13}$$

与式(3-10)的结果相同。

如果一个数据文件在分布式系统中只有一个副本，那么如果存储该数据文件的节点出现故障，它将无法从任何地方访问。在这种情况下，分布式计算系统对所有的比较任务是不可靠的。因此，对于每个数据文件，在不同的节点上存储至少有两个数据副本是必不可少的可靠性要求。以下定理表明，本章提出的数据分发方法保证了系统的可靠性要求。

定理 2：对于式(3-9)中定义的将 M 个数据文件分发到 N 个计算节点的约束优化问题，本章提出的分发策略给出了一个解决方案，该解决方案保证每个数据文件至少存储在两个不同节点上。

证明过程如下：如果存储在一个计算节点中的数据项不在另一个节点上重复，为了满足式(3-7)中的约束，所有其他数据项必须在同一个计算节点上存在副本，这意味着有一个计算节点存储了所有的数据项。此外，如果一个节点存储了所有的数据项，那么根据式(3-3)，其他节点都应该存储全部的数据项，这意味着全部数据都被分发到所有的节点，数据分发优化失效。因此，对于生成的任何优化结果，需要保证每个数据文件至少有两个存储在不同节点中的副本。

3.2.6 案例分析

本节将从一些案例开始分析数据分发问题的解决方案。此处使用数据 (M,N) 表示 M 个数据项和 N 个计算节点的数据分发问题。

为了分析以下情况，令 $D=\{d_i \mid i=1,2,\cdots,M\}$ 表示所有数据需要分配，max 表示在所有节点之间存储的最大数据数。

1. Data(M,2)

$$\max\{|D_1|,|D_2|\}=M \tag{3-14}$$

证明：考虑到如果计算节点 1 不存储数据项 d_i，那么与 d_i 相关的所有比较任务都必须由计算节点 2 执行，这意味着计算节点 2 应该存储数据项 d_i 和所有其他数据项。因此，计算节点 2 必须存储所有数据项。

2. Data(M,3)

$$\max\{|D_1|,|D_2|,|D_3|\}\geqslant\left\lceil\frac{2}{3}M\right\rceil \tag{3-15}$$

证明：假设所有计算节点存储的数据项少于$\left\lceil \frac{2}{3}M \right\rceil$个。考虑其中的任意两个，例如计算节点 1 和 2 这两个节点的相同数据项数应为 $D_1 \cap D_2$。显然是 $D_1 \cap D_2 < \left\lceil \frac{1}{3}M \right\rceil$。在这种情况下，第三个计算节点必须执行 $D-D_1$ 和 $D-D_2$。我们可以看到 $|(D-D_1)\cup(D-D_2)| > \left\lceil \frac{2}{3}M \right\rceil$，这不符合我们的假设。因此，所有 3 个计算节点都应至少存储$\left\lceil \frac{2}{3}M \right\rceil$个数据项。

3. Data(M, $M(M-1)/2$)

$$\max\{|D_1|, |D_2|, \cdots, |D_N|\} = 2 \tag{3-16}$$

证明：当计算节点的数量正好等于比较任务的数量时，很明显每个计算节点应该只分配两个不同的数据项。3.2.7 节提出一个启发式算法来解决数据分发问题的一般情况。

3.2.7　数据分发的启发式规则

得到式(3-9)中分发问题可行解的一种方法是满足式(3-6)和式(3-7)中的约束条件。从式(3-7)中的约束可以看出，如果我们能够确定特定比较任务 $C(x, y)$ 的位置，那么也可以确定所需数据 x 和 y 的位置。因此，我们以满足式(3-6)和式(3-7)中约束的方式将所有比较任务分配给计算节点。通过任务分配，得到了数据分发问题的可行解。

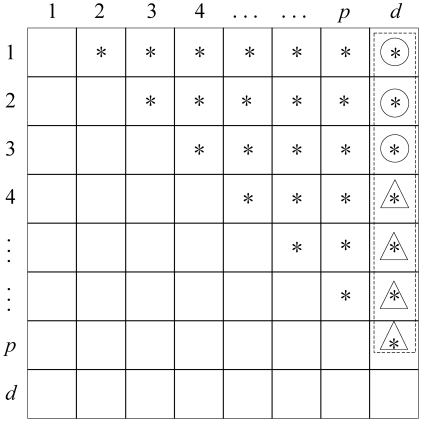

图 3.7　向节点 k 添加新文件

图 3.7 中比较矩阵的右栏表示在将新数据文件 d 分发到已经存储了 p 个数据

文件的节点时，可以分配给特定节点 k 的其他比较任务。因此，如果我们可以为每个新的数据文件 d 分配尽可能多的比较任务给节点 k，那么需要分发的数据文件的总数就可以最小化。

通过向节点 k 添加新数据 d 而引入的新的比较任务，以前从未分配过的任务（图 3.7 中用圆圈标记）和已经分配过的任务（图 3.7 中用三角形标记）。相关的数据分发规则如下。

规则 1：对于以前从未分配过的比较任务，在设计的数据分发策略中，通过遵循式(3-6)的约束，将尽可能多的比较任务分配给节点 k。

规则 2：对于已经分配的比较任务，可以设计数据分配策略，通过式(3-6)的约束重新分配每个任务。例如，如果比较任务 t 已经分配给节点 q，则该策略比较节点 k 和节点 q 之间分配的比较任务的数量。如果节点 k 的比较任务较少，则将任务 t 重新分配给节点 k，否则继续让节点 q 执行任务 t。

根据这些启发式规则，可以为实际的数据分发工作开发一个具有详细步骤的算法。

3.2.8 启发式数据分发策略的实现

上述启发式和任务驱动数据分发算法将在以下步骤中详细描述。

(1) 查找所有未分配的成对比较任务。

(2) 查找这些未分配的成对比较任务所需的所有数据文件。将这些数据文件放入集合 I，将集合 I 初始化为空。

(3) 从集合 I 中，找到数据文件 d，在未分配的比较任务中数据文件 d 与最大数量的比较任务有关。

(4) 选择一组存储节点，该组存储节点具有以下性质：①这些节点没有被分发数据文件 d；②这组节点被分配了最少数量的成对比较任务；③这组节点存储的数据文件最少。

令 C 表示这组存储节点。

(5) 为集合 C 中的所有节点检查 3.2.7 节中的规则 1。如果没有一个节点满足式(3-6)中的约束,则从集合 I 中删除数据文件 d 并返回步骤(3)。

(6) 在集合 C 中找到一个节点 k,该节点为空或者能够被分配给通过添加数据文件 d 而引入的且以前没有分配过的最大数量的新比较任务。将数据文件 d 分发到此节点 k。

(7) 对于在步骤(6)中通过添加数据文件 d 而引入的但以前已经分配给其他节点的比较任务,使用 3.2.7 节中的规则 2 重新分配这些任务。

(8) 重复步骤(1)～(7),直到将所有成对比较任务分配给计算节点。

这种启发式数据分发算法有助于解决式(3-9)中的最小化问题。让每一个比较任务的分配工作分配尽可能少的数据项,会减少要分发的数据项的总数。在所有计算节点之间均匀分布数据有助于满足每个节点的存储限制。如果所有节点具有相同或相似数量的比较任务,则很容易满足式(3-6)中作为优化式(3-9)约束的负载均衡要求。

3.2.9　启发式数据分发策略分析

为了对启发式数据分发策略进行分析,考虑一个包含 6 个数据文件(文件编号:0,1,2,3,4,5)和 4 个计算节点(节点编号:A,B,C,D)的示例。我们的数据分发算法给出了如表 3.1 所示的解决方案。

表 3.1　数据分发案例 1

节点编号	分发的数据文件编号	给相应节点分配的任务
A	0, 2, 3, 4	(0, 3) (0, 4) (2, 4) (3, 4)
B	0, 1, 4, 5	(0, 1) (1, 4) (1, 5) (4, 5)
C	0, 2, 3, 5	(0, 2) (0, 5) (2, 5) (3, 5)
D	1, 2, 3	(1, 2) (1, 3) (2, 3)

数据分发算法不仅将数据文件分配给计算节点,还将与数据文件分配相

对应的比较任务进行分配。因此,每个计算节点都被分配了数据文件和比较任务。

对于数据文件分配,每个计算节点存储整个数据集的一部分。与将全量分发相比,我们的数据分发算法减少了每个节点存储的数据量。在这里考虑的特定示例中,每个计算节点存储的数据项不超过 4 个。

对于比较任务分配,将为每个计算节点分配在数量上近似的比较任务,从而平衡比较任务的数量。这满足了分布式系统中所有计算节点之间的负载均衡要求。在这里考虑的这个特定示例中,6 个数据文件将生成总共 15 个比较任务,需要分配给 4 个计算节点执行。有 3 个节点被分配了 4 个任务,1 个节点被分配了 3 个任务,这表明任务分配基本平衡。

同时分发数据文件和分配比较任务也保证了比较任务所需的数据具有良好的数据本地化。表 3.1 清楚地展示了我们所讨论的这个具体例子。在执行比较任务期间,不需要通过网络通信将运行时数据从一个节点移动到另一个节点,启发式的数据分发策略消除了数据请求时的任何运行时的网络通信成本,这种优势在基于 Hadoop 的数据分发策略的系统中是必然存在的。

3.3 实验设计与实验结果分析

本节介绍的实验证明了我们的数据分发策略的有效性。它包括在一个真实的分布式计算系统中的仿真研究和实验。本节制定了评估标准,并根据这些标准评估我们的数据分发策略的性能。

3.3.1 评价标准和实验设计

使用 3 个标准来评估我们的数据分发策略:存储节约率、任务执行性能和可扩展性。

提高存储节约率是数据分发策略的目标之一。我们通过一组不同存储

节点数的实验对其进行测量，并与 Hadoop 的数据分发策略的结果进行比较。

数据本地化反映了数据分发的质量，是衡量计算性能的重要指标。如 3.1 节和式(3-9)所述，具有良好数据本地化的全比较任务意味着它可以在本地访问所有必需的数据对。考虑到比较任务是基于数据文件分发情况来分配的，因此在数据分发之后可以测量数据本地化好的比较任务的数量。为了突显我们的数据分发策略和 Hadoop 数据分发策略之间的不同数据本地化情况，实验测试了不同数量的计算节点和 Hadoop 数据文件副本。

任务执行性能的衡量指标是完成全比较问题的总执行时间，我们的目标是根据存储使用情况和其他限制对执行性能进行改进。如 3.2 节所述，总执行时间(T_{total})包括数据分发时间(T_{data})和处理比较任务时间(T_{task})。所有这些时间度量都是通过在真实的分布式系统中计算全比较问题实现的。最后，对我们的数据分发策略和 Hadoop 数据分发策略的这些性能指标进行了比较。

可扩展性对于大规模分布式计算中的大数据集比较问题具有重要意义。在相同的实验环境中，通过改变计算节点上的处理器数量来研究各种场景，展示了我们分布式计算框架的可扩展性。使用我们的数据分发策略分配所有数据和相关的比较任务能充分说明我们的数据分发策略的可伸缩性。

3.3.2　存储节约率

假设在这样一种情况下，有 256 个数据文件和一组计算节点，节点数量为 1～64 个。结合式(3-9)中的数据分发优化问题，使用分发到计算节点的数据项的最大值来表征数据分发策略中的存储使用情况。

第一组实验将启发式数据分发策略与 Hadoop 数据分发策略进行了比较，Hadoop 中的数据重复次数设置为默认值 3。实验结果见表 3.2，显示了启发式数据分发策略和 Hadoop 数据分发策略的存储使用、存储节约和数据本地化。根据现有的方法，将所有数据文件分发到每个节点所需的存储空间来

计算存储节约率。

表 3.2　启发式数据分发策略和 Hadoop 数据分发策略存储节约和数据本地化比较

节 点 数	N	4	8	16	32	64
$\max\{\|D_1\|,\|D_2\|,\cdots,\|D_N\|\}$	$d_{\max}$(定理 1)	129	91	65	46	33
	启发式数据分发策略	192	152	117	85	59
	Hadoop(3)	192	96	48	24	12
存储节约率/%	启发式数据分发策略	25	41	54	67	77
	Hadoop(3)	25	63	81	91	95
数据本地化率/%	启发式数据分发策略	100	100	100	100	100
	Hadoop(3)	56	48	28	14	7

与将所有数据分发到全部节点的数据分发策略相比，从表 3.2 可以看出，启发式数据分发策略和 Hadoop 数据分发策略都为大规模的全比较问题节省了大量的存储空间，Hadoop 数据分发策略比启发式数据分发策略节省了更多的空间。这意味着其需要更少的数据分发时间，特别是当存储节点的数量变得很大时。例如，对于一个有 64 个计算节点的分布式系统，启发式数据分发策略节省的存储空间高达 77%，而 Hadoop 数据分发策略节省的存储空间却高达 95%。

尽管启发式数据分发策略节省了较少的存储空间，但对于所有计算任务，启发式数据分发策略都实现了 100%的数据本地化，见表 3.2。相比之下，Hadoop 数据分发策略虽然节省了更多的存储空间，但是牺牲了数据的本地化率。例如，对于一个有 64 个数据节点的分布式系统，Hadoop 数据分发的数据本地化低至 7%，而启发式数据分发策略的数据本地化为 100%。良好的数据本地化对于大规模的全比较问题尤为重要。它将在执行比较任务时减少计算节点之间的数据文件调度，避免网络拥堵。因此，它将有利于全比较问题的整体计算性能提升。