计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2021.20200383
58(2):305-318,2021

基于蚁群优化算法的纠删码存储系统数据更新方案

李 乾 胡玉鹏 叶振宇 肖 叶 秦 拯
(湖南大学信息科学与工程学院 长沙 410082)
(qianli160@hnu.edu.cn)

An Ant Colony Optimization Algorithms Based Data Update Scheme for Erasure-Coded Storage Systems

Li Qian, Hu Yupeng, Ye Zhenyu, Xiao Ye, and Qin Zheng
(*College of Computer Science and Electronic Engineering, Hunan University, Changsha* 410082)

Abstract Owing to the high availability and space-efficiency of erasure codes, they have become the de facto standard to provide data durability in large scale distributed storage systems. The update intensive workloads of erasure codes lead to a large amount of data transmission and I/O cost. As a result, it becomes a major challenge to reduce the amount of data transmission and optimize the use of existing network resources so that the update efficiency of the erasure codes could be improved. However, very little research has been done to optimize the update efficiency of the erasure codes under multiple QoS (quality of service) metrics. In this paper, the proposed update scheme, the ACOUS (ant colony optimization algorithm based multiple data nodes update scheme) employs a two-stage rendezvous data update procedure to optimize the multiple data nodes updates. Specifically, the two-stage rendezvous data update procedure performs the data delta collection and the parity delta distribution efficiently, based on a multi-objective update tree which is built by the MACOU (multi-objective ant colony optimization update routing algorithm). Under typical data center network topologies, extensive experimental results show that, compared with the traditional TA-Update scheme, the proposed scheme is able to achieve 26% to 37% reduction of update delay with convergence guarantee at the cost of negligible computation overhead.

Key words distributed storage system; data update; erasure codes; ant colony optimization; update delay

摘 要 由于纠删码具备高可用性和高存储空间有效性的特点,采用纠删码为大规模分布式存储系统提供数据持久性已成为事实标准.然而,纠删码的密集型更新操作将导致大量的数据传输和 I/O 开销.如何减少数据传输量,优化现有网络资源的利用率,以提高纠删码的更新效率,成为纠删码存储系统面临的重要挑战.然而,在多重服务质量(quality of service, QoS)指标下,目前对纠删码更新效率的优化研究很少.针对此问题,提出一种基于蚁群优化算法的多数据节点更新方案(ant colony optimization algorithm based multiple data nodes update scheme, ACOUS),采用 2 阶段数据更新方式以优化多数据节点更新

收稿日期:2020-06-07;**修回日期**:2020-08-13
基金项目:国家自然科学基金项目(61872130,61572181);湖南省交通厅科技项目(201928);长沙市重点研发计划项目(kq1907103)
This work was supported by the National Natural Science Foundation of China (61872130, 61572181), the Science and Technology Project of Hunan Provincial Department of Communications (201928), and the Key Research and Development Program of Changsha (kq1907103).
通信作者:胡玉鹏(yphu@hnu.edu.cn)

过程.具体而言,基于多目标蚁群优化更新路由算法(multi-objective ant colony optimization update routing algorithm, MACOU)所构建的多目标更新树,2 阶段数据更新方式能有效地进行数据增量收集和校验增量分发.大量的实验结果表明,在典型的数据中心网络拓扑结构下,与 TA-Update 方案相比,所提方案能够在保证算法收敛的前提下,以可忽略的计算开销为代价,将更新时延降低 26%～37%.

关键词 分布式存储系统;数据更新;纠删码;蚁群优化;更新时延

中图法分类号 TP338.8

由于纠删码具备高可用性和高存储空间有效性的特点,采用纠删码为分布式存储系统以及数据中心[1-4]提供数据持久性已成为约定俗成的标准.纠删码把大数据对象划分成多个小的数据块,然后将这些小的数据块通过编码生成多个校验块,最后将这些数据块和校验块部署在不同存储群集的存储节点上.

数据更新在分布式存储系统中很常见[5-8].在许多企业的服务器和网络文件系统中,更新请求常常主导了写入工作负载(通常超过 90%)[9-10].在典型的最大距离可分(maximum distance separable, MDS)MDS(n,k)纠删码存储系统中,1 个数据块的更新请求涉及到 $n-k$ 个校验块的更新.根据纠删码更新过程中是否有传输整个数据块,可以将更新分为 2 类:基于 raid 的更新和基于增量的更新.其中,基于 raid 的更新方案需要在数据节点和校验节点之间传输整个数据块,为了完成对数据节点的更新,数据节点首先需要收集所有新的数据块,然后重新计算更新后的校验块,并将其传到相应的校验节点.基于增量的更新方案只需要将数据节点上数据增量(数据块修改的部分)通过广播的方式传输到每一个校验节点,相比之下,基于增量的更新可以节省更多的 I/O 操作和网络带宽.然而,频繁的数据更新也会导致巨大的 I/O 操作和带宽开销.尤其在使用纠删码的键值存储系统中,对键值数据进行密集的小型更新会导致巨大的 I/O 操作和昂贵的网络流量[7,11-15].

提高纠删码的更新效率具有十分重要的意义[16].最近一些研究工作者投入了大量精力来优化纠删码更新性能,以减少 I/O 操作和降低网络时延[6,17-19].在现有的更新方案中,如 CodFS[17] 和 Azure[20],采用追加式更新方式,或采用替换式更新和追加式更新的混合方式.一些研究者试图通过优化更新顺序和更新过程来减少网络传输开销[6,18-19].

从网络性能角度来看,数据增量最好是能够沿着数据节点与校验节点之间的最佳网络路径进行传输.虽然当前的路由算法能够可靠地处理分布式存储系统的大量网络流量[21-22],但是数据更新路由技术缺乏对异构存储系统(异构是指系统中的存储节点的吞吐量、I/O 速率不同,以及各段网络链路的有效带宽均不同)的深入研究,从而导致网络资源利用率不高.因此,优化大规模分布式存储系统路由极为重要,需要深入研究纠删码存储系统中数据更新路由.为了设计多目标路由优化更新算法,包括内存 I/O 速率和其他服务质量(quality of service, QoS)指标,如时延和带宽,本文提出一种基于蚁群优化算法的多数据节点更新方案(ant colony optimization algorithm based multiple data nodes update scheme, ACOUS),ACOUS 采用 2 阶段的数据更新过程来优化多数据节点的更新路由,并且使用基于多目标蚁群优化更新路由算法(multi-objective ant colony optimization update routing algorithm, MACOU)建立的更新树进行数据增量的收集和校验增量的分发.

具体而言,所提出的 ACOUS 方案采用集合式数据更新模式,主要包含 2 个阶段:在第 1 阶段,数据更新由数据节点触发.可以有多个数据节点(最多 k 个)进行更新,而其中的 1 个节点将被选为集合节点,其他所有数据节点计算数据增量并根据 MACOU 算法将数据增量发送到集合节点.在第 2 阶段,集合节点根据 MACOU 算法建立的多目标更新树,将较验增量依次分发到相应的校验节点.

本文的主要贡献有 3 个方面:

1) 针对大规模异构纠删码存储系统的内存吞吐量和跨节点链路带宽不相等的问题,本文提出的 ACOUS 是第一个深入研究异构纠删码存储系统中多数据节点更新过程和更新路由的工作.由于蚁群算法在解决 QoS 路由问题上的广泛应用[23],因此本文使用多目标蚁群优化算法来提高纠删码更新效率.

2) 分别定量分析了集合式更新和分布式更新方式下的数据更新 I/O 操作数量和数据传输开销,分析了多个数据节点的更新路由问题.

3) 开发了一个原型系统,进行了大量实验以评估在典型数据中心网络拓扑下 ACOUS 的性能.实验结果表明,本文提出的 MACOU 更新算法优于

TA-Update[5],在保证收敛性的前提下显著提高了更新效率.

1　背景及相关工作

本节先介绍纠删码以及蚁群优化路由算法的原理,然后介绍目前常用的纠删码的更新方法.

1.1　纠删码和蚁群优化路由算法

由于分布式存储系统有持久性和存储效率的需求,纠删码在大规模存储系统的应用成为研究焦点.众所周知,里德-所罗门码 RS(n,k)[24](reed solomon codes, RS)纠删码首先将一个大小为 D 字节的文件分为 k 个大小相等的数据块 d_i($1\leqslant i\leqslant k$),每个数据块大小为 D/k 字节.然后,这些等大的数据块通过编码生成 k 个数据块和 $n-k$ 个校验块,这些数据块和校验块组成一组(称为条带 Stripe),分布在 n 个不同的存储节点($D_1\cdots D_k;P_1\cdots P_{n-k}$)中,这 n 个节点属于不同的群集,以此来最大限度地提高系统的可靠性.每个校验块 p_j($1\leqslant j\leqslant n-k$)根据式(1)进行计算,其中 α_i^j 表示 d_i 到 p_j 的系数.基于这种线性编码,n 个块中任何不多于 k 个块出现故障就可以重建整个原始文件.

$$p_j=\sum_{i=1}^{k}\alpha_i^j\times d_i. \tag{1}$$

数据节点上的更新可以通过将数据增量广播到所有校验节点,使用预先定义的系数将增量添加到校验节点以保持数据一致性.$\Delta d=d_i'\oplus d_i$ 代表数据增量,当新数据 d_i' 覆盖原始数据 d_i 后,校验节点接收由数据节点发送的数据增量后进行计算并更新.如果有 u($1\leqslant u\leqslant k$)个数据节点进行更新,每个校验节点根据式(2)计算更新后的 p_j':

$$p_j'=\sum_{i=1}^{u}\alpha_i^j\times\Delta d_i+p_j. \tag{2}$$

蚁群算法是一种启发式智能优化算法,可以用来在图中寻找最优路径,该算法起源于在环境变化后,蚂蚁通过感知其他蚂蚁留下的信息素这种相互合作方式快速找到到达食物源的最短路径.蚁群算法已经被用来解决一系列组合优化问题,如最短路径问题、最优任务分配问题、QoS 路由问题等[23].其原理是每只蚂蚁在爬行的过程中都会留下信息素,每条路径上的信息素随着时间的推移不断增加或减少,所有的蚂蚁在爬行时更倾向于选择信息素更多的路径.定义时刻 $t+1$ 节点 i 到节点 j 的路径上的信息素的数量为

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t), \tag{3}$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^k(t), \tag{4}$$

在式(3)中,ρ 表示信息素挥发系数,$\tau_{ij}(t)$表示在时刻 t 节点 i 到节点 j 的路径上信息素的数量,$\Delta\tau_{ij}(t)$表示时刻 t 节点 i 到节点 j 的路径上的信息素增量.在式(4)中,$\Delta\tau_{ij}^k(t)$表示第 k 只蚂蚁在本次循环中从节点 i 到节点 j 的路径上留下信息素的数量,即经过某一条路径上的蚂蚁越多,这条路径上的信息素量就越多.

每只蚂蚁在爬行的过程中,当下一时刻有多条路径可以选择时,蚂蚁选择某条路径的概率又会受到可选路径上信息素的量和全局启发式因子的影响,式(5)说明了蚂蚁从节点 i 移动到节点 j 的公式,这个公式也称为状态转移公式.

$$P_{ij}^k(t)=\frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum\limits_{s\in\varphi_k(i)}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t)}, \tag{5}$$

其中,$\varphi_k(i)$表示节点 i 的下一跳候选集,η_{ij} 是蚂蚁从节点 i 移动到节点 j 的全局启发式因子,α 和 β 分别表示信息素和全局启发式因子的权重.通过这种方式,每条路径上的信息素的数量不断地增加或者减少.最终所有的蚂蚁都会沿着最短的那条路径爬行,这也称为蚁群算法的收敛性和正反馈机制.

1.2　相关工作

目前存在 2 种删除纠删码的更新方案,分别是基于 raid 的更新和基于 delta 的更新.基于 raid 的更新方案需要在数据节点和校验节点之间传输整个数据块.基于 delta 的更新只需要在数据节点和校验节点之间传输原数据块和新数据块之间不同的部分.与基于 raid 的更新方案相比,基于 delta 的更新方案数据传输量更小,后者数据更新效率更高.因此,本文采用基于 delta 的更新方案.在基于 delta 的更新方案中,又存在 3 种更新方式,分别为覆盖式更新、追加式更新和混合式更新.

覆盖式更新方式通过利用新数据直接覆盖原始数据块和校验块,实现了数据的实时更新.因为覆盖式更新能够确保数据的一致性和恢复效率,所以它被广泛地应用于纠删码的更新.但是为了完成检验块的更新,覆盖式更新方式又引入了大量的 I/O 操作和网络开销[5].

相比之下,追加式更新方式节约了校验块更新的 I/O 开销,通过将增量添加在数据块和校验块的后面,在适当的时间进行合并来完成更新.很多利用

纠删码存储方式的文件系统,比如,GFS[25],HDFS[14],Azure[20],采用追加式的更新方式.然而,追加式更新方式成为以读请求为主的存储系统的性能瓶颈,因为数据组合操作必须在访问数据时执行 1 次.

混合式更新方式是在允许数据块和校验块异步更新的前提下,数据块采用覆盖式更新,校验块采用追加式更新.这种更新方式可以确保数据块的访问效率以及校验块的更新效率.比如 PL[26],PLR[17]采用覆盖式更新方式,直接用新的数据覆盖原始数据块,然后将校验增量追加在校验块之后.

总而言之,以上 3 种更新方式中,数据节点必须向校验节点发送数据增量,以保证数据一致性.而 ACOUS 与现有的方法不同,不是关注单个节点的数据更新操作,而是优化整个更新过程中的 I/O 开销,并且致力于提高数据传输效率,针对多个 QoS 指标提高纠删码更新效率.

多约束 QoS 路由问题是一个典型 NP-C 问题[23].由于蚁群算法具备鲁棒性和收敛性,已成功地应用于求解许多离散优化问题.蚁群优化技术在解决 QoS 路由问题中得到广泛的应用.文献[27]采用基于蚁群优化的路由技术,将节点的实时位置和负载 2 个参数作为路由指标,以提高网络的 QoS 性能.文献[28]提出了一种改进的基于蚁群算法的动态源路由方案,该方案可以在较低的时延、较低的路由开销以及较低的能耗的情况下产生高数据包投递率.文献[29]利用蚁群算法提高以信息为中心的网络性能.

2 挑 战

通常情况,在大规模分布式存储系统中执行碎片化的、小数据块的纠删码更新操作可能导致性能显著下降.这是因为数据更新涉及到多个校验节点的更新,这不可避免地会导致相当大的 I/O 和带宽开销.尤其是对于分布式存储系统的更新操作,例如,云存储系统、分布式内存键值存储系统和块存储系统,大多数都是写请求,这将导致较大的时延.云存储系统 Azure[20]采用了基于日志的更新方案来分摊更新开销.分布式键值存储系统中的更新操作可能会导致一些小的更新,这将引入巨大的编码操作和网络流量.例如,在基于 Memcached 的 Cocytus[11]中,对于密集的小规模更新操作,即 set 操作,需要频繁地修改分布在多个校验节点上的数据.对于纠删码的磁盘阵列或块存储系统文件,如结构化数据库文件和虚拟机文件系统,对一个文件的更新可能导致多个节点在不同的集群中同时需要进行更新操作[8].纠删码更新主要面临如下挑战:

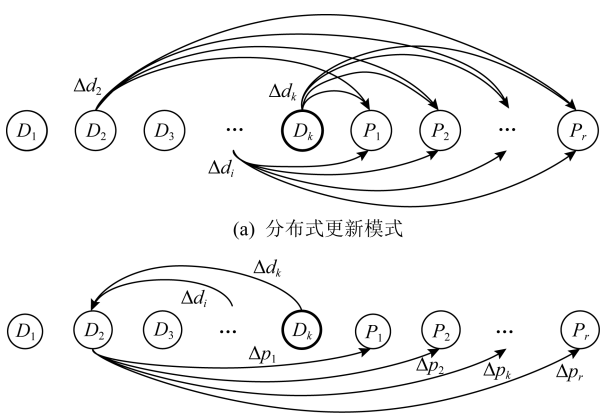
1) 当有多个数据节点需要更新时,这些节点之间的协作会导致大量的有限域运算和网络流量.根据式(2),虽然增量和编码计算可以并行地在所有数据节点上高效执行以完成更新,但如何协调更新的数据节点,以最小的网络流量将增量发送到相应的节点仍是一个难题.

2) 在异构的大规模纠删码存储系统中,如何在不相等的 I/O 吞吐量、跨节点的链路带宽和其他 QoS 约束情况下,为每个数据增量和校验增量找到最优的传输路径.这本质上是一个多目标优化路由问题,即如何充分利用网络资源,提高数据更新效率.

3 多个节点更新的路由问题分析

3.1 纠删码存储系统多数据更新的 2 种模式

当有多个数据节点需要更新时,有 2 种更新模式:分布式和集合式.图 1(a)显示了分布式更新模式的更新过程,其中用于更新的每个数据节点分别向每个校验节点发送 Δd_i.图 1(b)显示了集合式更新模式的更新过程,所有更新的数据节点 D_i 在完成更新之后计算 Δd_i,并将其发送到集合节点(rendezvous node, RN),由集合节点计算 Δp_j,然后将 Δp_j 发送到相应的校验节点 P_j.



(a) 分布式更新模式

(b) 集合式更新模式

Fig. 1 Two update patterns for multiple data nodes

图 1 多数据节点更新的 2 种模式

表 1 显示如果 $u(1\leqslant u\leqslant k)$个数据块需要更新,2 种更新模式产生的数据传输开销和数据读写开销.从表 1 可以看出,2 种模式在并行计算下的 I/O 开销几乎相同.分布式更新模式下的数据传输开销

随着 u 的增加迅速超过了集合模式中的数据传输开销,因此本文采用集合式更新模式来完成多节点更新问题.

Table 1　The Overhead of Two Update Patterns When u Data Nodes for Update

表 1　当有 u 个数据块需要更新时 2 种更新模式的负载

更新模式	读(本地)	传输	写(本地)
分布式	$u+r$	$u\times r$	$u+r$
集合式	$u+r$	$u+r-1$	$u+r$

3.2　路由的 QoS 度量

本文的目标是围绕 3 个典型的 QoS 指标进行研究,以提高大规模异构纠删码存储系统中多个数据节点的更新效率.

1) 距离.节点 s 到节点 d 之间的距离 $D(s,d)$ 是根据 2 个节点之间的跳数计算,距离通常被用来构建数据传输的最小生成树,由于网络的物理拓扑结构在一段的时间内保持稳定,TA-Update[5] 基于 Prim 算法以网络距离作为度量来构造更新树,通过更新树可以反映节点之间的路径长度,同时更新树的构造是以数据节点为根节点,采取自上而下的数据传输方式,最终的目的是使得整个更新过程数据传输距离最小.网络距离越小表示数据传输的跳数较少,意味着数据传输时延较小.

2) 带宽.带宽是衡量网络中链路容量的指标.数据传输的瓶颈由传输路径上的最小带宽确定[30],由于存储节点之间链路上带宽是异构的,树状的拓扑结构由此形成,文献[30]利用带宽来构造再生树,并将再生树的构建和再生码进行结合,从而有效利用存储节点之间每条链路上的可用带宽,降低网络流量,提高再生码的恢复效率.但实时可用带宽难以获得,因此,本文利用平均带宽 $B(e)$ 作为链路 e 的带宽,$B(s,d)$ 表示路径 $p(s,d)$ 上的最小平均带宽.

$$B(s,d)=\min\{B(e),e\in path(s,d)\}. \quad (6)$$

3) 时延.本文采用时延作为衡量更新效率的关键指标.时延越小代表数据传输效率越高.路径 $path(s,d)$ 上的总时延 $delay(s,d)$ 是处理时延 d_{proc}、传输时延 d_{trans} 和传播时延 d_{prop} 之和.式(7)表示总时延的计算方式,本文没有考虑计算排队时延,因为排队时延超出了本文的讨论范围.

$$delay(s,d)=\sum_{e\in path(s,d)}(d_{\text{prop}}+d_{\text{proc}}+d_{\text{trans}}). \quad (7)$$

距离、带宽、时延这 3 个因素是影响纠删码更新的主要网络因素[5],但是 TA-Update 仅以距离为度量并基于 Prim 算法来构造更新树,存在局限性.本文利用多目标蚁群优化算法可以综合考虑这 3 个因素,实现多目标优化,以此提高纠删码的更新效率.

4　基于蚁群优化的多数据节点更新方案

本节将针对 3.2 节列出的 3 个 QoS 指标详细说明 ACOUS 更新方案.首先介绍 2 阶段集合式更新模式的过程,然后介绍多目标蚁群优化更新路由算法.

4.1　集合式更新的 2 个阶段

ACOUS 的主要思想是采用 2 阶段集合式更新方案,多目标更新树实现对 $RS(k+r,k)$ 编码高效的数据增量收集和校验增量分发.

第 1 阶段. 数据增量收集.图 2(a)描述了在集合式更新中用数据节点 D_2 作为集合节点更新的第 1 阶段.当有 $u(u\leqslant k)$ 个数据节点需要更新,每个数据节点直接用新的数据块 $d_i'(1\leqslant i\leqslant k)$ 覆盖原始数据块 d_i,计算数据增量 Δd_i,并将数据增量 Δd_i 通过本文 5.2 节中介绍的 MACOU 算法传递到集合节点 D_2.通过这种方式,完成数据增量的收集,这个阶段产生的数据传输量为 $u-1$,本地读取次数为 u,本地写入次数为 u.

第 2 阶段. 校验增量分发.图 2(b)描述了集合式更新的第 2 阶段.RN 基于第 1 阶段收到的 Δd_i,使用公式 $\Delta p_j=\sum_{i=1}^{u}\alpha_i^j\times\Delta d_i,(1\leqslant j\leqslant r)$ 计算得到每个校验节点的增量,并使用 MACOU 算法构造多目标更新树来分发给相应的校验节点.然后,每个校验

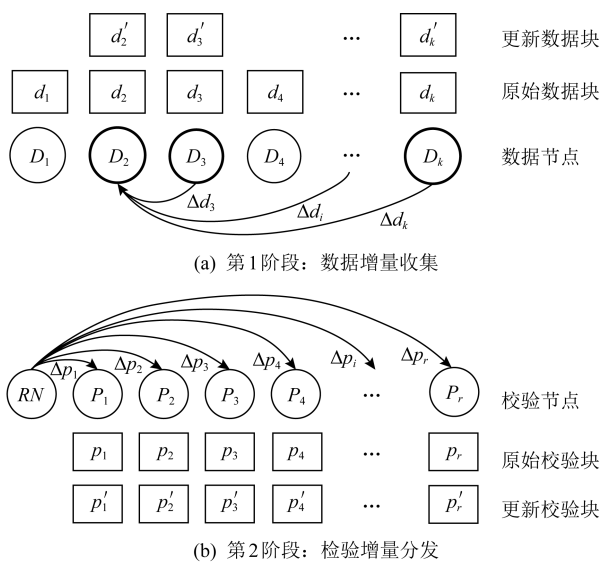


Fig. 2　The two-stage rendezvous update scheme

图 2　集合式更新的 2 个阶段

节点将其原始校验数据 p_j 通过公式 $p'_j=\Delta p_j+p_j$ 进行更新.为方便起见,本文原型系统在单个节点采用的是覆盖式更新操作,需要说明的是,具体到固态硬盘或非易失性内存(non-volatile memory, NVM)介质,数据不一定是定点更新,数据更新时会执行Flash或NVM的标准写操作和数据擦除,以及垃圾回收,这方面已有很多相关的工作,不属于本文的讨论范畴.第2阶段产生的数据传输量为 r,本地读取次数为 r,本地写入次数为 r.

4.2 多目标蚁群优化更新路由算法

表2解释了算法中涉及的符号.式(8)描述了节点 s 和节点 d 之间的最佳更新路径的问题.其中节点的权重代表节点的处理时延以及传输时延的大小,边的权重代表传播时延的大小.

$$\arg\min\{delay(s,d)\},\ \text{s.t.}\ B(e)\geqslant B_{\text{req}},\ e\in path(s,d). \tag{8}$$

Table 2 Ant Colony Algorithm Parammeters

表2 蚁群算法参数

符号	含义
P_{ij}	从节点 i 移到节点 j 的概率
τ_{ij}	边 $e(i,j)$ 上信息素的数量
η_{ij}	节点 i 到节点 j 上的全局启发式因子
α	信息素的权重
β	距离的权重
λ	时延的权重
M	迭代的次数
K	每次迭代中蚂蚁的数量
$D(j,d)$	节点 j 到节点 d 之间的距离
$pt(m,k)$	第 m 次迭代中第 k 只蚂蚁的爬行路径
$sd(m,k)$	第 m 次迭代中第 k 只蚂蚁的时延
i	蚂蚁所在的当前节点
$\varphi(i)$	节点 i 的相邻节点

算法1详细描述了MACOU算法,其目的是在带宽约束下搜索从节点 s 到节点 d 的最小时延的路径.

算法1. 多目标蚁群优化更新路由算法MACOU.

输入:网络拓扑图 $G(V,E)$、每个节点的权重 $W_i(i\in V)$ 和每个边的权重 $W_e(e\in E)$、每条边的带宽 $B(i,j)$ 和最小带宽 B_{req} 约束、源节点 s 和目标节点 d;

输出:满足最小带宽约束下时延最小的路径 $p(s,d)$.

① 初始化 $\alpha,\beta,\lambda,k,m,i=s,converge=$false;

② 初始化每条路径上的信息素 τ;

③ 计算每个节点 i 到目的节点 d 的距离 $D(i,d),i\in V$;

④ for ($m=0$; $m<M$; $m++$) do

⑤ for ($k=0$; $k<K$; $k++$) do

⑥ while $i\neq d$ 或 $\varphi(i)\neq\varnothing$ do

⑦ for each $j,j\in\varphi(i)$ do

⑧ 计算概率 P_{ij};

⑨ end for

⑩ 从 P_{ij} 中选择最大的 $P_{iv},v\in\varphi(i)$;

⑪ end while

⑫ 将节点 v 加入到路径 $pt(m,k)$,并且计算时延 $sd(m,k)=sd(m,k)+W_e+W_i$;

⑬ if $v==d$ then

⑭ 保存路径 $pt(m,k)$ 和时延 $sd(m,k)$;

⑮ else

⑯ $i=v$;

⑰ end if

⑱ end for

⑲ $converge=isconverge(pt)$;

⑳ if $converge\neq$true then

㉑ return $pt(m,k)$ 和 $sd(m,k)$;

㉒ else

㉓ 更新所有路径上的信息素;

㉔ end if

㉕ end for

MACOU算法首先根据输入初始化行①②中每个参数的值,行③计算每个节点到目标节点的距离.

行④~㉕是实现从节点 s 到节点 d 的路径迭代搜索过程.在爬行过程中,蚂蚁当前所处的节点 i 如果不是目的节点并且还有路径可走(节点 i 到邻居节点 j 的链路带宽 $B(i,j)>B_{\text{req}}$ 且节点 j 没有被访问过),即 $\varphi(i)$ 不为空,则根据式(5)选择概率最大的节点 j 作为下一个要访问的节点,式(5)中的全局启发式因子计算方式如下:

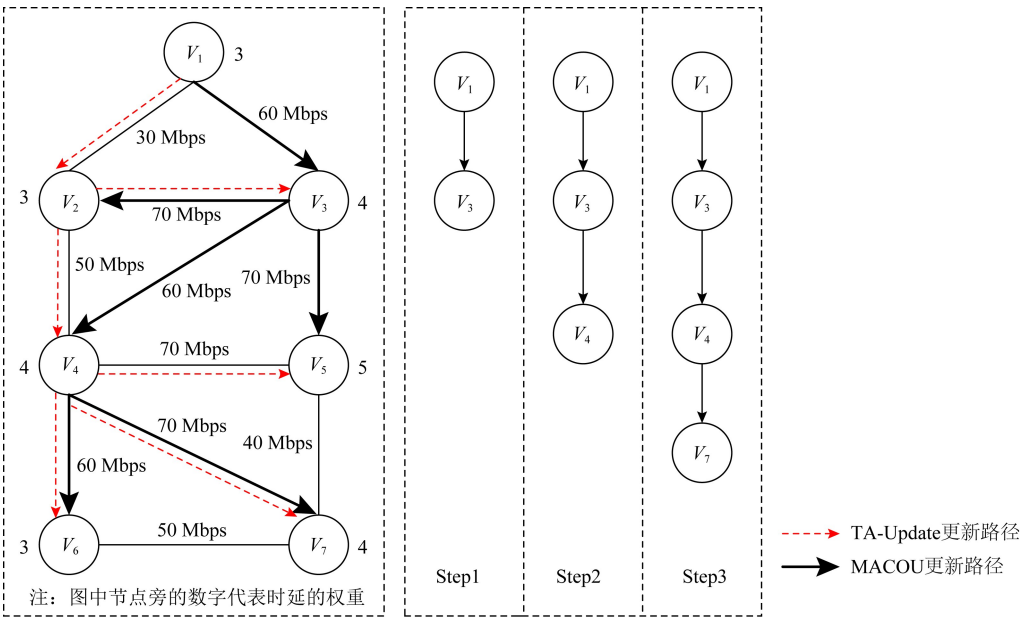
$$\eta(i,j)=(1/(D(j,d))^{\beta}\times(1/(W_i+W_e))^{\lambda},\ j\in\varphi(i). \tag{9}$$

行⑲判断路径是否收敛,行㉓根据式(3)(4)更新信息素.

例1. 图3说明了MACOU算法的过程.图3(a)是多目标更新树,其中 V_1 是集合节点,其他所有节点为校验节点.在图3(a)中节点权重代表时延的总和,边的权重代表带宽的大小.在实际的分布式存储系统中,所有节点都通过树状拓扑结构连接,其中一些

节点连接底层交换机,然后底层交换机连接上层交换机.节点之间的所有路径都通过交换机中继.每条边的权值表示其可用平均带宽(单位是 Mbps).每条边的权值表示其可用平均带宽.假设最小带宽要求是 $B_{req}=50$ Mbps,本文需要找到一个具有最小时延的更新树,其每条边 e 应满足 $B(e)\geqslant 50$ Mbps.本文提出的更新路由算法 MACOU 可以在网络初始化过程中进行部署,因此,每个数据节点能够在一段时间内保持到每个校验节点的最优路由,直到下一次网络重新配置.图 3(a)中的黑色粗线箭头和所有节点的边构造了以 V_1 为根节点的多目标更新树.

图 3(b)描绘了路径 $p(V_1,V_7)$ 的搜索步骤.蚂蚁从 V_1 开始,在初始信息素相同的情况下,由于 $B(V_1,V_2)<50$ Mbps,因此下一跳是 V_3.类似地,蚂蚁从 V_3 出发,距离 $D(V_2,V_7)=2$,距离 $D(V_4,V_7)=D(V_5,V_7)=1$.但是,V_5 的时延大于 V_4.所以,蚂蚁选择 V_4 作为下一跳.最后,V_4 可以直接到达 V_7.由于 MACOU 的正反馈机制,选择路径:$V_1\rightarrow V_3\rightarrow V_4\rightarrow V_7$ 的蚂蚁数量越多,该路径上的信息素就越多.因此,经过多次迭代后,从 V_1 到 V_7 的路径最终收敛于 $V_1\rightarrow V_3\rightarrow V_4\rightarrow V_7$,且这条路径满足最小更新时延和带宽约束.



(a) 从节点 V_1 出发的更新树　(b) 从节点 V_1 到节点 V_7 的路径

Fig. 3　A case study of multi-objective update tree construct with MACOU

图 3　基于 MACOU 算法的多目标更新树

类似地,本文可以找到 V_1 到其他校验节点最佳更新路径,并由此构造出类似图 3(a)中的更新树.值得注意的是,第 1 阶段的数据增量收集也可以利用 MACOU 算法和校验增量分发的方式构建.MACOU 与图 3(a)中基于 Prim 算法的 TA-Update 方案形成的虚线箭头组成的更新树相比,前者具备更小的更新时延.例如,需要搜索从 V_1 到 V_2 的路径,由于带宽约束,通过 MACOU 算法搜索出的路径是 $V_1\rightarrow V_3\rightarrow V_2$,比 TA-Update 搜索的路径 $V_1\rightarrow V_2$ 效率更高.2 条路径之间的总时延随传输数据包大小的增大而增大,例如,传输大小为 1～9 MB 的数据包,路径 $V_1\rightarrow V_3\rightarrow V_2$ 和路径 $V_1\rightarrow V_2$ 的时延分别是 0.03～0.27 ms 和 0.033～0.3 ms.

5　集合节点的选择

通常,关于多个 QoS 度量,最优节点选择是 NP-hard 问题.ACOUS 以随机或时延最小的方式简化集合节点的选择.由于集合节点具备数据增量收集和校验增量分发的功能,所以 RN 的选择也是一个重要的问题.其中,本文将随机选择集合节点的选择方式称为 RRN(random rendezvous node),通过 RRN 选择的节点称为 D_{RRN}.将时延最优的集合节点的选择方式称为 ORN(optimal rendezvous node),通过 ORN 选择的节点称为 D_{ORN}.本文在每个条带中随机分配一个条带头来执行节点选择算法.根据节点

地址,从数据节点中随机选择 RRN,由条带头进行更新.虽然 RRN 可能不是最优的,但是它的选择过程比 ORN 的选择过程更有效.如算法 2 所示,当有 u 个节点进行更新时,ORN 以最小时延为代价执行数据增量收集和校验增量分发,基于算法 1,选择 ORN 的问题可以定义为 $\arg\min_i, i \in V\{sumdelay[i]\}$,其中 $sumdely[i]$ 的定义为

$$sumdelay[i]=\sum_{k=1}^{u} delay(D_i,D_k)+\sum_{j=1}^{r} delay(D_i,P_j). \tag{10}$$

算法 2. ORN 选择过程.

输入:网络拓扑 $G(V,E)$、每个节点的权重 W_i $(i\in V)$ 和每条边的权重 $W_e(e\in E)$、每条边上的带宽 $B(i,j)$、需要更新的数据节点的集合 D 和校验节点的集合 P;

输出: D_{ORN}.

① 初始化 $sumdelay[i]=0$;
② for each D_i in D' do
③ for each $D_k(D_k\neq D_i)$ in D' do
④ 使用算法 1 计算时延 $delay(D_i,D_k)$;
⑤ $sumdelay[i]=sumdelay[i]+delay(D_i,D_k)$;
⑥ end for
⑦ for each P_j in P do
⑧ 使用算法 1 计算时延 $delay(D_i,P_j)$;
⑨ $sumdelay[i]=sumdelay[i]+delay(D_i,P_j)$;
⑩ end for
⑪ end for
⑫ 从 $sumdelay[i]$ 选出最小时延对应的 D_i;
⑬ $D_{\mathrm{ORN}}=D_i$;
⑭ return D_{ORN}.

本文通过在原型系统上进行实验,以验证 ORN 和 RRN 的更新效率.如图 4 所示,随着更新节点数量的增加,在忽略集合节点选择计算的情况下,ORN 的时延比 RRN 的时延更低.在存在 30 多个节点原型存储系统中,基于 MACOU 算法获得的路由信息,算法 2 的选择过程不超过 0.01 s.值得注意的是,在网络重新配置之前,MACOU 算法为每个数据节点获取的路由信息在一段时间内不会频繁更新.因此,考虑到计算资源的普遍可用性,大规模存储系统最好选择 ORN.由于计算成本可以忽略,本文在第 6 节中对 2 个不同的集合节点的更新时延进行了更多的评估,由于节点的更新序列是随机的,ORN 基本均匀分布在不同的节点上,可以实现负载均衡,因此 ORN 节点的性能不会成为纠删码更新效率的瓶颈.

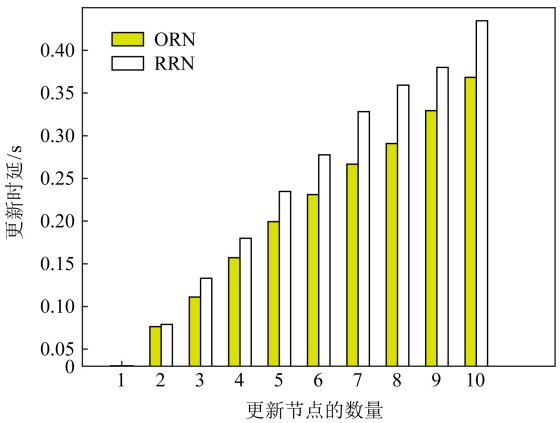


Fig. 4 Comparisons of update delay under different RNs

图 4 不同 RN 下的时延

6 实验评估

本文在基于存储集群的腾讯云虚拟机上实现了 ACOUS 更新方案,并研发了一个原型系统[①]来评估本文提出的方案.同时本文在更大的网络拓扑(大于 200 个节点)上进行仿真,以进一步评估本文的更新方案.属于同一条带的数据节点和校验节点通常被随机部署在不同的集群中.由于 ORN 中集合节点选择操作的时延通常是以微秒为单位,与数据更新时延相比可以忽略,因此本文在下面的评估中忽略了计算时间.

6.1 原型系统评估

本文的原型存储集群由多达 35 台腾讯云虚拟机(Ubuntu 服务器 14.04.1 LTS,4 GB RAM 和双核 Intel Xeon Cascade Lake (2.5 GHz))组成,它们通过带宽为 1.5 Gbps 的网络连接在同 1 个子网中.本文利用腾讯云的性能监测工具来估计平均带宽.

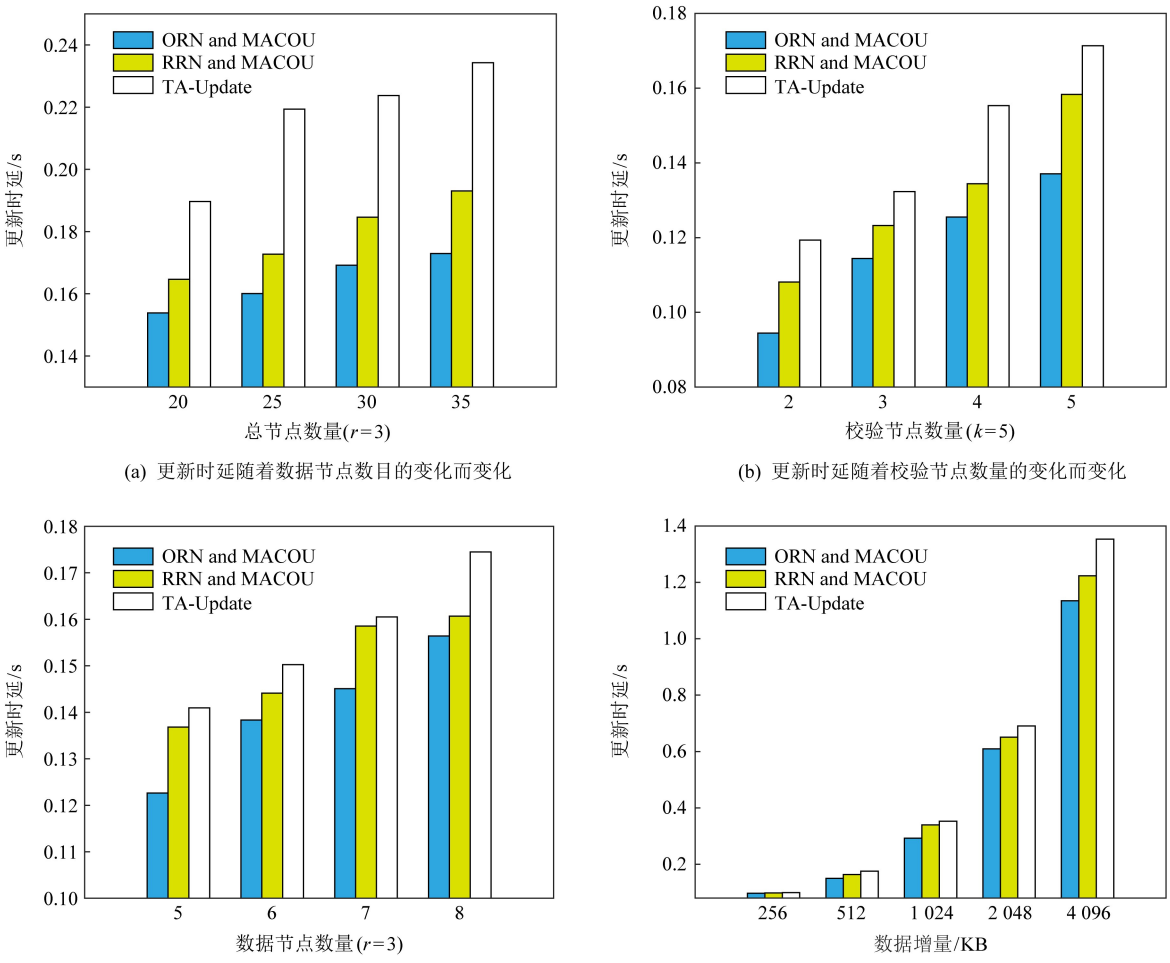
在图 5 中,我们提供了不同方案下更新时延的广泛比较.如图 5(a)所示,当 $r=3$ 时更新时延随更新数据节点数量 k 增加的情况.如图 5(b)所示,当 $k=5$ 时,更新时延随校验节点数量 r 增加的情况.

① https://github.com/599020328/acous

随着数据节点或校验节点数量的增加,将会有更多的数据增量和校验增量沿着本文方案中最优更新树有效地传递.因此,当 k 或 r 增加时,本文的方案能够减少更多的更新时延.与 TA-Update 相比,MACOU 方案降低了 30%～32%的更新时延.

如图 5(c)所示,本文的原型系统在有 5 个数据节点和 5 个校验节点需要更新的情况下,系统中不同节点数量下的更新时延,这些节点随机部署在不同的存储集群中.

随着节点数量的增加,本文将它们安排到更多的存储集群中以实现负载平衡,从而产生了更多的传递跃点和数据包中继.因此,随着存储系统的扩展,本文的方案与 ORN 和 MACOU 结合后降低的时延从 28%迅速增加到 37%左右.如图 5(d)所示,随着数据增量增加,本文的方案可以减少的时延几乎线性增加的.



(a) 更新时延随着数据节点数目的变化而变化

(b) 更新时延随着校验节点数量的变化而变化

(c) 更新时延随着数据节点数目的变化而变化

(d) 更新时延随着传输的数据增量的变化而变化

Fig. 5　Experiment in prototype system

图 5　原型系统中的实验

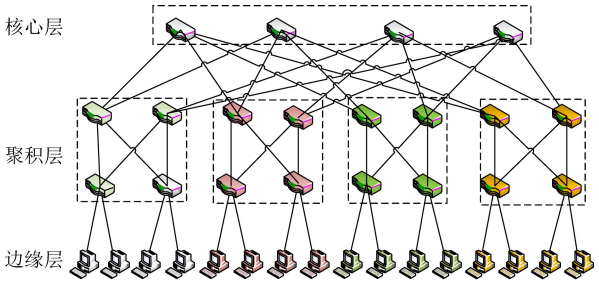
6.2　大规模网络仿真

本文在基于组件的模拟器 OPNET[31] 上进行了大量的实验,以评估本文在大规模异构分布式存储网络系统中更新方案的性能,并且该系统定期执行更新.除了构建 I/O 和带宽的随机拓扑结构外,本文还评估了其他 2 种典型数据中心网络拓扑的更新方案,如图 6 所示,即 Fat tree 和 DCell,这 2 种网络拓扑结构都有很高的网络容量和健壮的连通性.在相同的条件下,本文还在相同的条件下与 TA-Update 方案进行对比.表 3 列举了仿真系统的一些关键参数.

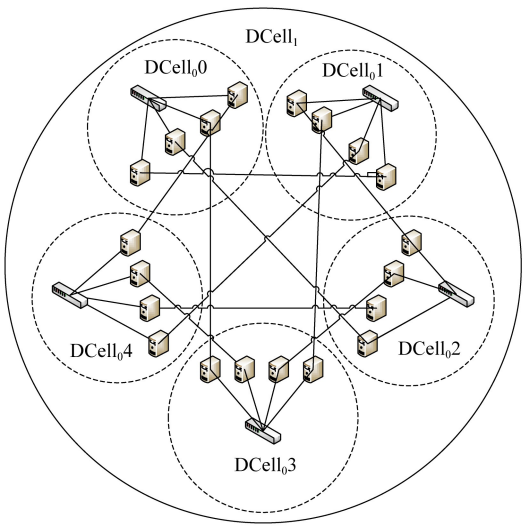
6.2.1　不同大小数据传输量的系统更新时延

本节比较了不同数据传输量下更新时延的仿真结果.首先比较了在传输不同数据增量时,ORN 和 MACOU、RRN 和 MACOU 与 TA-Update 方案更新效率的不同,如图 7 所示,更新时延随着传输数据增量的增大而增大.可以看到,与 TA-Update 方案相比,采用 RRN 和 MACOU 的方案可以节省大约

26%的更新时延,采用 ORN 和 MACOU 方案的时延降低了 18%左右.



(a) 胖树(N_{pod}=4)



(b) DCell

Fig. 6 Two typical data center network topologies

图 6 2 种典型的数据中心网络拓扑

Table 3 Experiment Parameters

表 3 实验参数

参数	范围	默认值
节点的数量	{200,220,240,260,280,300,320}	300
更新的数据节点的数量 k	{5,30}	5
更新的校验节点的数量 r	{3,30}	5
传输的数据增量 s/KB	{256,512,1 024,2 048,4 096}	512

图 8 显示了当 $k=9$ 时,更新时延随 r 增加的情况.图 9 显示了当 $r=5$ 时更新时延随 k 增加的情况. 与图 5 中原型系统的评估结果一致,随着数据节点或校验节点数量的增加,本文方案的优势变得更加明显.可以看到,本文采用 ORN 和 MACOU 的方案与 TA-Update 相比,时延降低了 30%.

6.2.2 不同规模的系统更新时延

图 10 描述了当更新的数据节点和校验节点的数量分别占总节点数量的 5%时,不同大小规模下

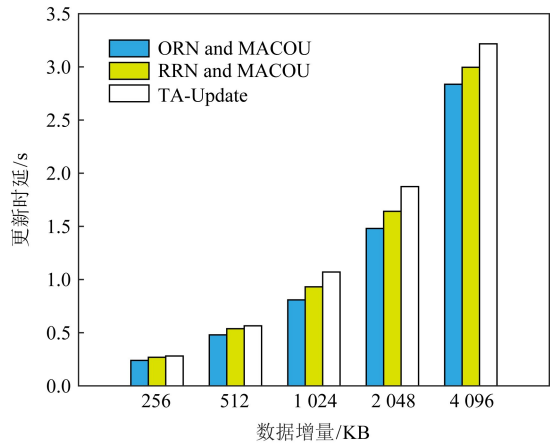


Fig. 7 Delay under data delta

图 7 数据增量下的时延

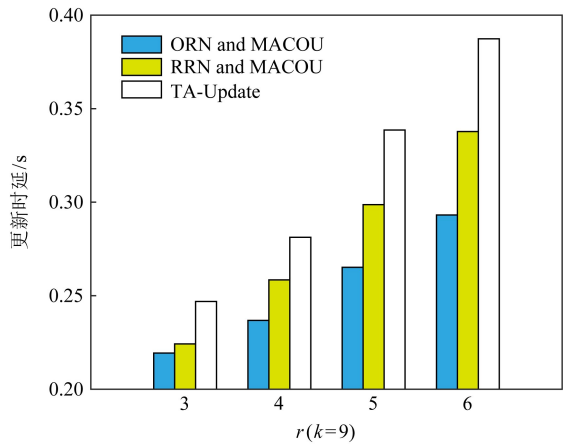


Fig. 8 Delay under varying r when $k=9$

图 8 更新时延随 r 的变化

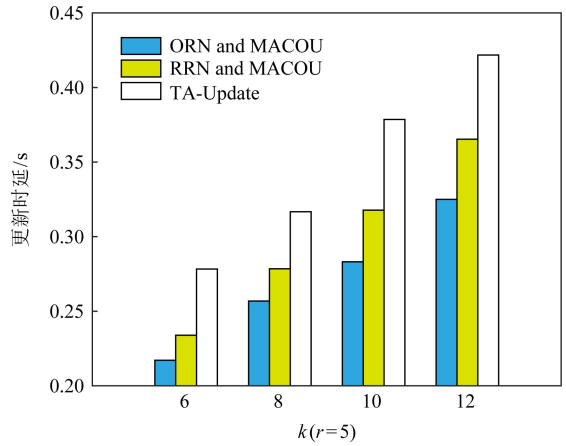


Fig. 9 Delay under varying k when $r=5$

图 9 更新时延随 k 的变化

的更新时延情况.这些用于更新的节点随机部署在定期启动更新过程的存储系统中.随着系统规模的

增大,TA-Update 更新方案与本文采用 ORN 和 MACOU 的方案相比,TA-Update 更新的时延增加得更快.图 11 显示了系统中只有 5 个数据节点和 5 个校验节点进行更新时,系统中总节点数在不同情况下的时延.根据图 5(c)原型系统的评价结果,并从图 10 和图 11 可以看出,由于本文采用了 ORN 和 MACOU 结合的方案,随着系统的扩展,纠删码的更新效率明显优于 TA-Update 算法.因此,当节点数超过 250 时,本文的方案可以将更新时延降低 28%～30%.

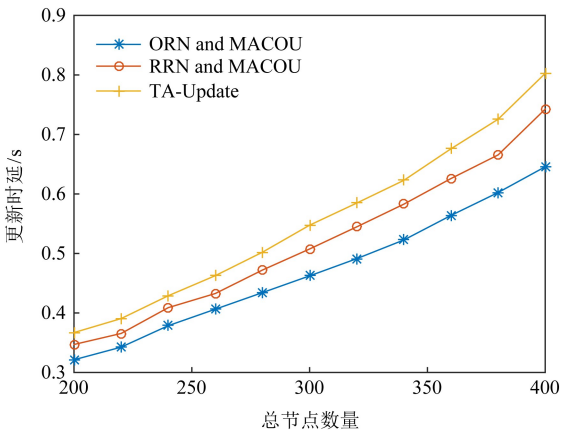


Fig. 10　Update delay under the varying system scale

图 10　不同网络规模下的时延(r,k 占总节点 5%)

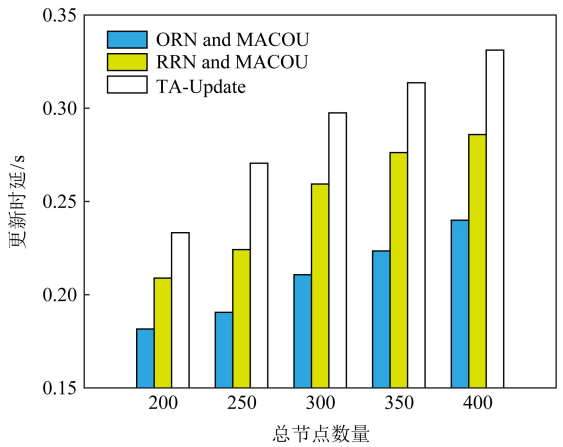


Fig. 11　Update delay under the varying system scale

图 11　不同网络规模下的时延($r=5$,$k=5$)

6.2.3　不同网络拓扑的更新时延

为了进一步研究本文提出的方案的适应性,如图 6 所示,本文在 2 个典型的数据中心网络拓扑结构(即胖树和 DCell)下进行了大量的实验.Fat tree 拓扑网络分为 3 个层次:自上而下分别边缘层、汇聚层和核心层,其中汇聚层交换机与边缘层交换机构成一个 pod.在图 12 和图 13 中给出了在不同数据中心拓扑结构下更新时延的实验结果.与之前的实验结果类似,本文的方案也能够节省几乎相同的时延,这表明本文提出的方案具有良好的适应性.

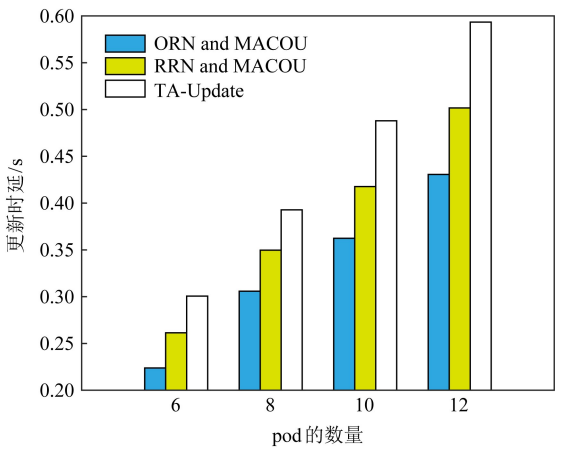


Fig. 12　Update delay under varying scale of Fat tree

图 12　不同规模的胖树结构下的时延

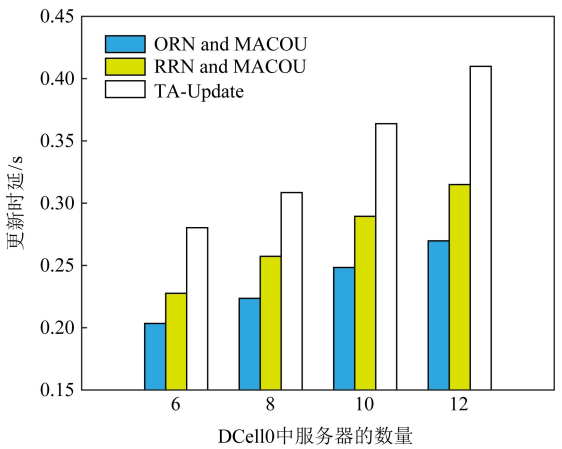


Fig. 13　Update delay under varying scale of DCell

图 13　不同规模的 DCell 结构下的时延

6.3　收敛性分析

如图 13 所示,当本文设置 MACOU 算法的相关参数 $\alpha=1$,$\beta=1$,$\lambda=1.6$,$\rho=0.4$ 时,更新时延随着蚂蚁数量的增加而迅速减少,然后达到最佳蚂蚁数量的下限,最后实现搜索路径的收敛.信息素的定期更新使蚁群算法能够快速收敛,图 14 评估了本文的方案的收敛性.当图 14(a)中的总节点个数为 300 时,在释放的蚂蚁数约为 500 个时实现了路径的收敛.如图 14(a)～(c)所示,在较大的网络规模中通常会导致较慢的收敛.例如,图 14(b)中当 pod 的数量 N_{pod} 分别为 6 和 8 时,最佳蚂蚁数量接近 250 和 500,而当 pod 数量 $N_{pod}=10$ 时,需要大约 1 000 个

蚂蚁才能完成更新路径搜索.在图 14(d)中比较了由 200 个节点组成的 3 种网络拓扑下的更新方案的收敛速度.可以看到,在 DCell 拓扑结构中,由于其高效的全局连接性,本文的方案在蚂蚁数量达到 230 左右时实现了最快的收敛速度.相反,由于随机网络的复杂性和冗余性,随机网络拓扑收敛速度最慢.

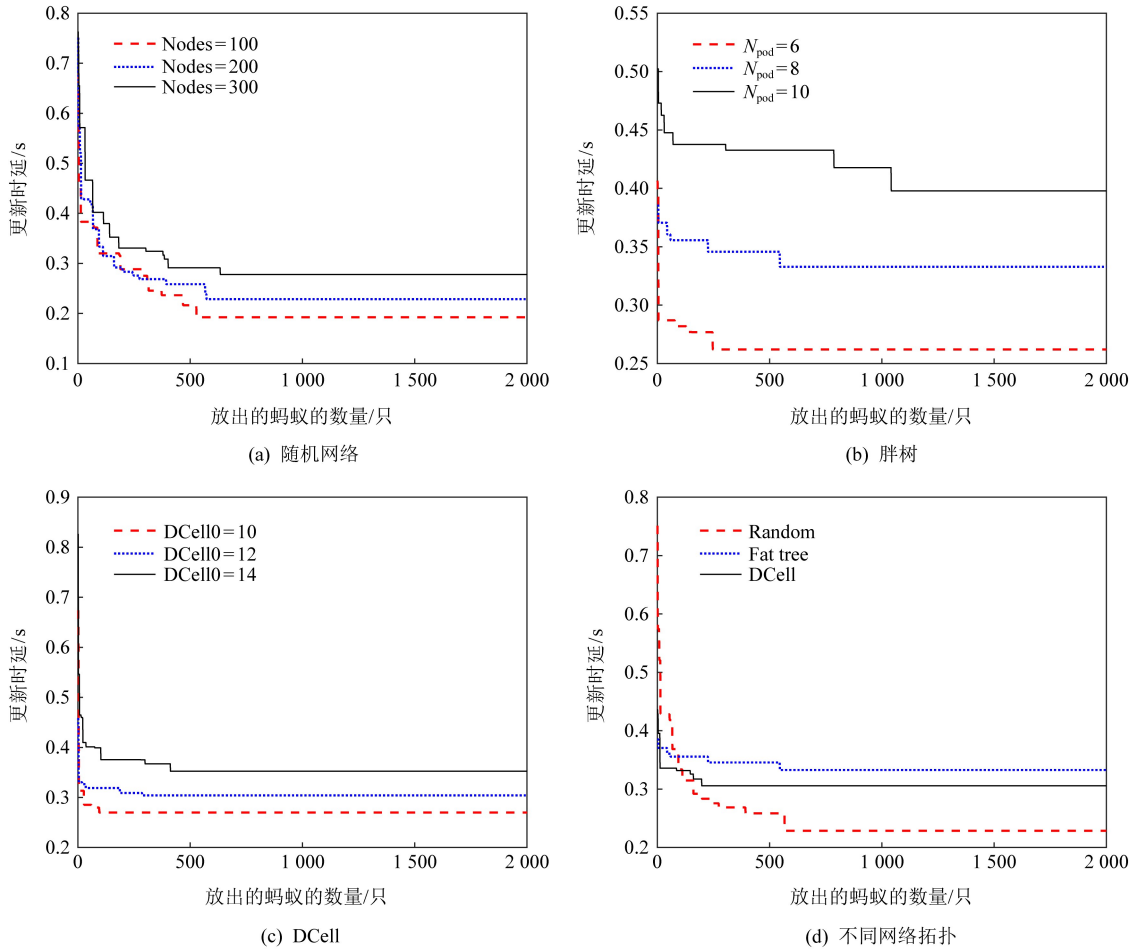


Fig. 14 Update delay with increasing number of ants

图 14 随着蚂蚁数量增加的更新时延

7 结 论

针对分布式纠删码存储系统中的频繁数据更新问题,本文尝试考虑多个 QoS 度量下的多数据节点的数据更新优化问题.本文提出的 ACOUS 更新方案是基于 MACOU 算法,采用 2 个阶段来优化多个数据节点更新.具体来讲,在数据更新过程中,使用基于蚁群优化路由算法构建的多目标更新树来实现数据增量的收集和校验增量的分发.在典型的数据中心网络拓扑下,与 TA-Update 更新方案相比,大量实验结果表明,本文的 ACOUS 更新方案以忽略不计的计算开销为代价实现了 MACOU 算法收敛,从而将纠删码的更新时延降低 26%~30%.

贡献声明:李乾负责论文的所有实验和论文撰写,胡玉鹏对论文研究方案做全面指点和论文修改,叶振宇参与论文实验模拟验证,肖叶参与论文后期修改,秦拯参与论文后期修改.

参 考 文 献

[1] Zhang Zhe, Deshpande A, Ma Xiaosong, et al. Does erasure coding have a role to play in my data center? [J/OL]. Microsoft Research Technical Report, 2010 [2020-07-16]. https://www.microsoft.com/en-us/research/wp-content/uploads/2010/05/paper.pdf

[2] Liu Qing, Feng Dan, Jingy Hong, et al. Zcodes: General systematic erasure codes with optimal repair bandwidth and storage for distributed storage systems [C] //Proc of the 34th IEEE Symp on Reliable Distributed Systems (SRDS). Piscataway, NJ: IEEE, 2015: 212-217

[3] Sathiamoorthy M, Asteris M, Papailiopoulos D, et al. Xoring elephants: Novel erasure codes for big data [C] //Proc of the 39th Int Conf on Very Large Data Bases (VLDB). New York: ACM, 2013: 325-336

[4] Hu Yupeng, Liu Yonghe, Li Wenjia, et al. Unequalfailure protection coding technique for distributed cloud storage systems [J]. IEEE Transactions on Cloud Computing, 2017, 15(8): 1-14

[5] Wang Yijie, PeiXiaoqiang, Ma Xingkong, et al. TA-Update: An adaptive update scheme with tree-structured transmission in erasure-coded storage systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(8): 1893-1906

[6] Zhang Fenghao, Huang Jianzhong, Xie Changsheng. Two efficient partial-updating schemes for erasure-coded storage clusters [C] //Proc of the 7th IEEE Int Conf on Networking, Architecture, and Storage. Piscataway, NJ: IEEE, 2012: 21-30

[7] Xia Jie, Huang Jianzhong, Qin Xiao, et al. Revisiting updating schemes for erasure-coded in-memory stores [C] // Proc of the 12th Int Conf on Networking, Architecture, and Storage(NAS). Piscataway, NJ: IEEE, 2017: 1-6

[8] Shen Jiajie, Zhang Kai, Gu Jiazhen, et al. Efficient scheduling for multi-block updates in erasure coding based storage systems [J]. IEEE Transactions on Computers, 2018, 67(4): 573-581

[9] Adams I F, Storer M W, Miller E L. Analysis of workload behavior in scientific and historical long-term data repositories [J]. ACM Transactions on Storage, 2012, 8(2): 6:1-6:27

[10] Narayanan D, Donnelly A, Rowstron A. Write off-loading: Practical power management for enterprise storage [J]. ACM Transactions on Storage, 2008, 4(3): 10:1-10:23

[11] Chen Haibo, Zhang Heng, Dong Mingkai, et al. Efficient and available in-memory KV-store with hybrid erasure coding and replication [J]. ACM Transactions on Storage, 2017, 13 (3): 25:1-25:30

[12] Yiu M M T, Chan H H W, Lee P P C. Erasure coding for small objects in in-memory KV storage [C] //Proc of the 10th ACM Int Systems and Storage Conf. New York: ACM, 2017: 1-12

[13] Shankar D, Lu Xiaoyi, Panda D K. High-performance and resilient key-value store with online erasure coding for big data workloads [C] //Proc of the 37th IEEE Int Conf on Distributed Computing Systems. Piscataway, NJ: IEEE, 2017: 527-537

[14] Xia Mingyuan, Saxena M, Blaum M, et al. A tale of two erasure codes in HDFS [C] //Proc of the 13th USENIX Conf on File and Storage Technologies (FAST). Berkeley, CA: USENIX Association, 2015: 213-226

[15] Huang Cheng, Simitci H, Xu Yikang, et al. Erasure coding in windows azure storage [C] //Proc of the 21st USENIX Annual Technical Conf (USENIX ATC). Berkeley, CA: USENIX Association, 2012: 15-26

[16] Luo Xianghong, Shu Jiwu. Review of erasure codes in storage systems [J]. Journal of Computer Research and Development, 2012, 49(1): 1-11 (in Chinese)
(罗象宏, 舒继武. 存储系统中的纠删码研究综述[J]. 计算机研究与发展, 2012, 49(1): 1-11)

[17] Chan J C W, Ding Qian, Lee P P C, et al. Parity logging with reserved space: Towards efficient updates and recovery in erasure-coded clustered storage [C] //Proc of the 12th USENIX Conf on File and Storage Technologies (FAST). Berkeley, CA: USENIX Association, 2014: 163-176

[18] Pei Xiaoqiang, Wang Yijie, Ma Xingkong, et al. Efficient in-place update with grouped and pipelined data transmission in erasure-coded storage systems [J]. Future Generation Computer Systems, 2017, 69: 24-40

[19] Akash G J, Lee O T, Kumar S M, et al. Rapid: A fast data update protocol in erasure coded storage systems for big data [C] //Proc of the 17th IEEE/ACM Int Symp on Cluster, Cloud and Grid Computing (CCGRID). Piscataway, NJ: IEEE, 2017: 890-897

[20] Calder B, Wang J, Ogus A, et al. Window sazure storage: A highly available cloud storage service with strong consistency [C] //Proc of the 23rd ACM Symp on Operating Systems Principles. New York: ACM, 2011: 143-157

[21] Guillen L, Izumi S, Abe T, et al. SDN implementation of multipath discovery to improve network performance in distributed storage systems [C] //Proc of the 13th Int Conf on Network and Service Management (CNSM). Piscataway, NJ: IEEE, 2017: 1-4

[22] Ponnusamy P P, Abinaya S. Virtual network routing in cloud computing environment [C] //Proc of the 8th Int Conf on Computer Communication & Informatics(ICCCI). Piscataway, NJ: IEEE, 2016: 1-6

[23] Abdullahi, Riham. Modeling and Simulation of an OSPF Router [M]. Khartoum State, Sudan, University of Khartoum, 2014: 1-84

[24] Reed I S, Solomon G. Polynomial codes over certain finite fields [J]. Journal of the Society for Industrial and Applied Mathematics, 1960, 8(2): 300-304

[25] Ghemawat S, Gobioff H, Leung S T. The Google file system [C] //Proc of the 19th ACM Symp on Operating Systems Principles. Berkeley, CA: USENIX Association, 2003: 29-43

[26] Jin Chao, Feng Dan, Jiang Hong, et al. RAID6L: A log-assisted RAID6 storage architecture with improved write performance [C] //Proc of the 27th IEEE Symp on Mass Storage Systems and Technologies (MSST). Piscataway, NJ: IEEE, 2011: 1-6

[27] Brito J, Martínez F J, Moreno J A, et al. An ACO hybrid metaheuristic for close-open vehicle routing problems with time windows and fuzzy constraints [J]. Applied Soft Computing, 2015, 32: 154-163

[28] Chatterjee S, Das S. Ant colony optimization based enhanced dynamic source routing algorithm for mobile ad-hoc network [J]. Information Sciences, 2015, 295: 67-90

[29] Lü Jianhui, Wang Xingwei, Ren Kexin, et al. ACO-inspired information-centric networking routing mechanism [J]. Computer Networks, 2017, 126: 200-217

[30] Li Jun, Yang Shuang, Wang Xin, et al. Tree-structured data regeneration in distributed storage systems with regenerating codes [C] //Proc of the 29th IEEE INFOCOM. Piscataway, NJ: IEEE, 2010: 1-9

[32] Chang Xinjie. Network simulations with OPNET [C] //Proc of the 1999 Winter Simulation Conf. Piscataway, NJ: IEEE, 1999: 307-314

Li Qian, born in 1993. Master candidate. Student member of CCF. Her main research interests include erasure coding technology, distributed storage system.

李 乾,1993 年生. 硕士研究生,CCF 学生会员.主要研究方向为纠删码技术、分布式存储技术.

Hu Yupeng, born in 1981. Professor, PhD supervisor. Senior member of CCF, IEEE, ACM. His main research interests include big data, artificial intelligence, cloud storage reliability and security.

胡玉鹏,1981 年生.教授,博士生导师,CCF, IEEE,ACM 高级会员.主要研究方向为大数据、人工智能、云存储可靠性与安全.

Ye Zhenyu, born in 1996. PhD candidate. Student member of CCF. His main research interests include key-value storage system and erasure coding. (yezhenyu@hnu.edu.cn)

叶振宇,1996 年生.博士研究生,CCF 学生会员.主要研究方向为 K-V 存储系统和纠删码技术.

Xiao Ye, born in 1996. PhD candidate. Student member of CCF. His main research interests include file system, distributed storage system and security. (yexiao@hnu.edu.cn)

肖 叶,1996 年生. 博士研究生,CCF 学生会员.主要研究方向为文件系统、分布式存储系统与安全.

Qin Zheng, born in 1969. Professor and PhD supervisor. His main research interests include blockchain, data science, information security, and software engineering.

秦 拯,1969 年生,教授,博士生导师.主要研究方向为区块链、数据科学、信息安全与软件工程.