arXiv:2512.20960v1 [cs.DS] 24 Dec 2025

k 服务器问题中的公平性

Mohammadreza Daneshvaramoli[1],Helia Karisani[1],Mohammad Hajiesmaili[1],Shahin Kamali[2],和 Cameron Musco[1]

[1]马萨诸塞大学阿默斯特分校， mdaneshvaram@umass.edu, hkarisani@umass.edu, hajiesmaili@cs.umass.edu, cmusco@cs.umass.edu[2]约克大学， kamalis@yorku.ca

摘要

我们首次对 k 服务器问题中的公平性进行形式化研究。该问题的目标不仅是最小化总移动成本，还要在各服务器之间公平地分配成本。我们首先定义了 (α, β)-公平性的一般概念：对于参数 $\alpha \geq 1$ 和 $\beta \geq 0$，任何服务器承担的成本都不超过总成本的 α/k 倍再加上一个加性项 β。随后，我们证明，在离线和在线场景中，都能在不损失竞争性的前提下实现公平性。在离线场景中，我们给出一种确定性算法：对于任意 $\varepsilon > 0$，该算法可将任意最优解转化为 (α, β)-公平解，其中 $\alpha = 1 + \varepsilon$ 和 $\beta = O$(直径 $\cdot \log\ k/\varepsilon$)，且解的成本仅增加一个加性 O(直径 $\cdot k \log\ k/\varepsilon$) 项。这里，直径表示底层度量空间的直径。我们还在在线场景中给出了类似结果：任意竞争算法都可以转化为随机在线算法，该算法面对 oblivious 对手时以高概率满足公平性，并且仍具有竞争性，仅有少量损失。

上述结果留下了一个重要问题：在在线设置中，面对完全自适应的对手，能否通过确定性算法或随机算法实现公平性？我们对此问题取得了进展，证明经典的确定性双重覆盖算法（DCA）在线度量上公平；当 $k = 2$ 时，在树度量上也公平。不过，我们也证明了一个否定性结果：对于一般树度量，DCA 对任何非平凡参数都不公平。我们进一步证明，在均匀度量（即分页问题）上，确定性的先进先出（FIFO）算法是公平的。我们还证明，任何“标记算法”（包括最近最少使用（LRU）算法）都满足一种较弱但仍有意义的公平性概念。

1 引言

k服务器问题由 Manasse、McGeoch 和 Sleator [MMS90], 提出，是在线算法理论的核心问题。在这个问题中， k 服务器在度量空间内移动，以响应一系列请求，目标是将总移动成本降至最低。具体来说，算法接收一个输入序列 $\sigma = (\sigma_1, \ldots\ldots, \sigma_T)$，其中包含 T个请求，每个请求都对应度量空间中的一个点。在时间 $t \in \{1, \ldots, T\}$，它必须将某台服务器移动到

请求 σ_t所在的位置，并产生一笔成本，金额等于该服务器移动的距离。挑战在于每一步都要决定移动哪台服务器，才能将总移动成本降至最低。在在线设置中，作出决策时无法预先获知后续请求。

在过去三十年中， k-服务器问题一直是研究在线决策和竞争分析的基础性基准，并与分页、度量任务系统及其他领域有着深刻联系 [FKL+91、BLS92、BE98、Kou09、FRR94、KMWW20]。人们研究了该问题的许多变体，包括服务器成本带权或由不同度量空间决定的情形 [BEKN23、AC21、ACS24、KT04、CV20、BEKN23], ，以及请求同时具有取件地点和送达地点的情形 [CK19、BCN22、CC26] 。该问题也在建议复杂度框架下得到研究：在此框架中，算法会以若干无错误建议比特的形式接收关于输入的信息 [EFKR11、RR15、BKKK17、GKL16], ；近来，研究者还在带有预测的学习增强框架下研究该问题，在这些框架中会提供额外信息，但这些信息可能存在错误 [LMS22、ACE+23b、ACE+23a、CSW23]。

服务器问题中的公平性 k-服务器。 k-服务器问题的动机来自这样一些场景：服务提供商派遣移动代理前往客户地点，目标是尽量缩短总行驶距离。在此类情形中，公平性也十分重要：理想情况下，应平衡工作负载，避免任何单个服务器负担过重。同样，在分布式系统和云平台中，提供相同服务的 k 虚拟机（服务器）可能需要在节点之间迁移，以处理传入的请求 [CFK+22, A+97, DK14]。在这些领域，除了尽量降低总迁移成本外，还应确保这些成本在各服务器之间公平分配。

公平性已在广泛的离线和在线决策问题中得到研究。例如，各种形式的聚类 [AEKM19, MV21, KSDH23, DEMZ24], 匹配[HJS+24, HHIS23, AS22], 调度 [AAN+98, NTV23, JLM25, BS06], 选择问题（如背包问题） [PKL21, LSS+24, FSTW19], 以及秘书问题 [KVT12, BMM24]。对公平性的一种常见理解，是确保不同群体或代理之间按比例保持平衡。例如，公平聚类方法可能会考虑带颜色的点，并力求在确保每个簇中各种颜色所占比例大致相同的同时，尽量降低标准聚类成本 [AEKM19, KSDH23]。类似地，公平类别匹配研究的是顶点与代理相关联的情形，目标是在确保每个代理在匹配中所占比例适当的同时，尽可能多地匹配顶点 [HHIS23, HJS+24, wNM25]。

与 k-服务器问题更为相关的是，Singh 等人 [SKC23]研究了 *k-FOOD* 问题，其中请求由源-目的地对及其对应的取件时间窗定义，从而有效地为服务请求设置了截止期限。在该问题的标准版本中，目标是最小化所有服务器行驶的总距离；而在公平 变体中，目标是最小化任一服务器行驶的最大距离。 [SKC23] 证明了该问题标准版本和公平版本的离线版本均为 NP-困难；其困难性主要源于时间窗和截止期限等额外约束。请注意，经典的 k-服务器问题中不存在这些约束，且该问题的离线版本可在多项式时间内求解 [BE98]。

Martinez-Sykora 等人 [MMCF24] 研究了一个带有公平性约束的食品配送问题，

其目标是最小化各服务器之间等待时间的范围——他们将该问题表述为一个整数线性规划，并证明该问题可在小规模输入实例上求解，但未证明其一般情况下具有可处理性。Ong 等人 [OPY+24] 研究了旨在确保客户（即请求而非服务器）公平性的乘车调度算法——他们主要关注实验评估，而非理论保证。Chiplunkar 等人 [CHKV23] 研究了针对 k-分页最小最大公平性目标的在线竞争算法；分页是 k-服务器问题的一种特殊情形。他们同样关注请求之间而非服务器之间的公平性。

尽管关于算法设计中公平性的文献浩如烟海，其中也包括与 k-服务器相关的问题，但或许令人意外的是，直接探讨 k-服务器问题本身公平性的研究却寥寥无几。Manasse、McGeoch 和 Sleator [MMS90] 最初提出了平衡算法（*BAL*），该算法在每一步都通过贪心地最小化这一步之后所有服务器所产生的最大成本，来选择要移动的服务器。该算法明确以公平性为设计动机，并已证明在包含 $k+1$ 个点的度量空间上具有 k-竞争性 [MMS90]。然而遗憾的是，BAL 在一般度量下无法达到有界竞争比。因此，据我们所知，为 k-服务器问题设计兼具公平性和竞争性的算法这一问题仍完全悬而未决。

广为研究的分页问题 是 k-服务器问题的一个经典特例，其底层度量空间是均匀的——也就是说，所有点对之间的距离都相等 [ST85, KMMO88]。分页问题模拟操作系统、存储层次结构和网页缓存中的缓存行为。当请求访问的页面（顶点）当前未存储在任何一个 k 缓存槽（服务器）中时，就会发生缺页故障。在 k-服务器的解释下，每个缓存槽对应一台服务器，每次缺页故障都对应着以单位成本将一台服务器移动到被请求的顶点。经典的确定性分页算法包括 First-In-First-Out (FIFO)和 Least-Recently-Used (LRU)。在这种解释下，FIFO 移动最长时间未移动过的服务器，而 LRU 移动最近一次移动发生时间最久远的服务器。FIFO 和 LRU 在分页问题上都达到了最优的确定性竞争比 k [ST85]。分页与 k-服务器问题之间的紧密联系，促使我们在后文分析 FIFO、LRU 和标记算法在均匀度量下的公平性属性。

1.1 我们的贡献

在这项工作中，我们首次对公平性作为 k-服务器问题的优化目标进行了形式化研究。我们的贡献总结如下。

1.1.1 (α, β)-公平性定义

我们首先为 k-服务器问题形式化一个自然的公平性概念：如果在请求序列 σ 上，没有任何服务器支付的成本超过 $\alpha \cdot \text{cost}(A,\sigma)_k + \beta$，我们就称确定性算法 A 是 (α, β)-公平的；其中 cost(A, σ) 是所有服务器为服务 σ 而支付的总成本。对于所有服务器支付相同成本的完全公平算法，$\alpha = 1$ 和 $\beta = 0$ 将成立。

对于随机化算法，我们区分事前公平性和事后公平性 [AFSV24]。事前公平性意味着期望意义上的公平性——即，如果在请求序列 σ 上，没有任何服务器的期望成本大于 $\alpha \cdot \mathbb{E}[\text{cost}(A\,\sigma\,]/k + \beta$，我们就称随机算法 A 是 (α, β)-事前公平的。通过简单的随机交换策略，很容易在保持竞争性的同时实现 (1,diam)-事前公平性。这里，diam 表示底层度量空间的直径。假设我们从给定度量空间上的一个算法开始，其在输入 σ 上的期望成本为 $\mathbb{E}[\text{cost}(A, \sigma)]$。在算法开始时，我们随机排列服务器身份，按照该排列交换服务器位置，并在之后也按照该排列重新分配服务器的工作负载。随后，除去每台服务器为初始交换支付的加性成本 $\leq$diam，根据对称性，所有服务器的期望成本都完全相同，等于 $\mathbb{E}[\text{cost}(A\,\sigma\,]/k$。

然而，上述方法并不十分令人信服。虽然在初始随机交换之后，各服务器的期望成本仅相差一个较小的加性因子，但对于任意给定的交换选择，服务器的成本分布仍可能非常不公平。因此，在全文中，我们采用随机算法的一个事后公平性概念。具体而言，我们称一个随机算法A 在请求序列 σ 上是 (α, β)-公平的，即以随机选择为条件的高概率下，每台服务器产生的成本均不超过 $\alpha \cdot \text{成本}(A,\sigma)/k + \beta$。该界限成立的确切概率自然会影响可实现的 α 和 β。

虽然本文重点研究 (α, β)-公平性，但我们证明了它要么涵盖其他自然的公平性概念，要么可以很容易地化归到这些概念，例如乘法公平性，即所有服务器承担的成本彼此之间相差不超过一个固定的乘法因子；或者加法公平性，即所有服务器承担的成本彼此之间相差不超过一个固定的加性因子。我们将在论文完整版中讨论这些替代性定义，以及它们之间的化归和联系。

平等主义成本模型。 虽然我们的主要结果聚焦于标准的功利主义目标（总移动成本），但我们的公平性保证也适用于平等主义成本模型。在该模型中，算法的成本定义为任一服务器所行经的最大距离。关于此模型的正式定义、界限和证明，我们留待附录 B 给出。

1.1.2 离线公平性

我们的首要贡献是证明，在离线场景中，无须显著牺牲竞争性即可实现 $(\alpha,\ \beta)$-公平性。具体而言，我们在第 3 节证明：

定理 1 (定理 8 重述)。对于任意 k-服务器输入序列 σ，若其所在度量空间的直径为 D_i，且任意 $\varepsilon > 0$，则存在针对 σ 的离线解，该解对 $\alpha = 1+\varepsilon$ 和 $\beta = O\left(\log k \cdot \text{D_i}_{\varepsilon}\right)$ 是(α, β)-公平的，且总成本不超过 $\text{OPT}(\sigma) + O\left(k \log k \cdot \text{D_i}_{\varepsilon}\right)$，其中 $\text{OPT}(\sigma)$ 为离线最优成本。此外，该公平解可在多项式时间内计算。

也就是说，我们能够实现具有 $\alpha = 1 + \varepsilon$ 因子的公平性，同时只需付出一个较小的加性因子 β，并使最优离线成本略有增加；这一增量取决于 k、直径和 $1/\varepsilon$，但重要的是，它与输入长度及离线最优成本无关。

证明概述。 为了证明定理 1，我们引入一种算法（算法 1），它将输入序列 σ 的任意一个最优解，通过一系列成对的服务器交换转换为一个公平解。成对交换是对解序列的一种修改：在给定时间 z，两个服务器在度量空间中交换位置；此后所有时间 z，它们负责服务对方此前服务的请求。

我们的算法分轮次进行，迭代改进初始解的公平性。每一轮中，算法会找出当前承担最高成本的服务器 H 以及当前承担最低成本的服务器 L 。此外，算法还会找出一个时间步，使得若在此时交换这两台服务器，它们的负载将近乎均衡。我们证明，只要解尚未达到 (α, β)-公平，那么在任意服务器参与至少两次交换后，最大服务器负载就会降低一个乘法因子 $\approx 1+\varepsilon$。因此，每台服务器经过 $O(\log_{1+\varepsilon} k) = O(\frac{\log k}{\varepsilon})$ 次交换后，最大负载会从可能高达平均负载的 k 倍，降至至多平均负载的 $(1+\varepsilon)$ 倍。由于每次交换本身会产生 $\leq 2 \cdot \text{diam}$ 的加性成本，因此我们得到界 $\alpha = 1+\varepsilon, \beta = O\left(\frac{\log k \cdot \text{diam}}{\varepsilon}\right)$，而相较于 $\text{OPT}(\sigma)$，我们的成本增幅为 $O\left(\frac{k \log k \cdot \text{diam}}{\varepsilon}\right)$。

不难看出，我们的交换算法可在多项式时间内运行。因此，由于 k-服务器的离线最优解可在多项式时间内计算，[BE98]，(α, β)-公平性可以在多项式时间内实现。这与近期关于相关公平调度问题的研究形成对比；这些研究证明了此类问题在计算上是困难的 [SKC23, MMCF24]。

关于 diam 的依赖性。我们的一些公平性保证包含一个加性项，该项由参数 diam 确定。diam 表示算法执行过程中任意两个服务器之间曾达到的最大间距（而非底层度量空间的直径）。在所有服务器都从同一位置出发这一标准假设下，diam 始终不超过算法产生的总移动成本。

特别是，对于满足 $\text{diam} \leq \gamma \cdot \text{cost}(A, \sigma)$ 的某些执行（其中 $\gamma \in (0, 1)$），我们界中的加性 diam 相关项可以改写为乘法松弛。例如，形如以下形式的保证

$$c_i(A, \sigma) \leq \frac{\alpha}{k} \text{cost}(A, \sigma) + \beta$$

将变为

$$c_i(A, \sigma) \leq \left(\frac{\alpha}{k} + \gamma \cdot \frac{\beta}{\text{diam}}\right) \text{cost}(A, \sigma).$$

如此解读，即使在度量直径很大、但总移动成本占主导的执行过程中，我们的结果仍然有意义。

1.1.3 使用随机化实现在线公平性

我们的第二项贡献表明，对于在线 k-服务器问题，使用随机算法对抗 oblivious 对手，也可以实现 (α, β)-公平性——也就是说，请求序列 σ 预先选定，且不依赖于算法的决策（这是研究在线 k-服务器问题的随机化算法时所采用的标准设定）。具体而言，我们在第 4 节证明：

定理 *2*（重述定理 *9*）。对于任意固定的 $\gamma > 0$ 和 $\varepsilon > 0$，存在一个随机在线k-服务器算法；对于度量直径为 *diam* 的度量空间上的任意固定输入序列 σ ，该算法以概率 $1 - k \cdot exp\left(-\Omega\left(\frac{\varepsilon^2}{\gamma \cdot k} \cdot c(\sigma)^{1/(\gamma+1)}\right)\right)$，实现 $\alpha = 1 + \varepsilon$ 和 $\beta = O(c(\sigma)^{1/(1+\gamma)} \cdot diam)$ 的 (α, β)-公平性。此外，该算法产生的总成本至多为 $c(\sigma) + O(k \cdot c(\sigma)^{1/(1+\gamma)} \cdot diam)$。这里， $c(\sigma)$ 是在给定度量空间上，随机算法已知可达到的最佳成本；例如，对于 n 点度量， $c(\sigma) = polylog(k, n) \cdot \mathrm{OPT}(\sigma)$ *BBMN15, BCL*$^+$*18*]。

为了理解定理 2，考虑包含 $\gamma = 1$ 和 $\varepsilon = 1$的简化设置。该定理表明，可以实现 (α, β)-公平性，同时以概率 $1 - k \cdot \exp(-\Omega(c(\sigma)^{1/2})$ 达到 $\alpha = 2$, $\beta = O(c(\sigma)^{1/2} \cdot \mathrm{diam})$，且成本增加$O(k \cdot c(\sigma)^{1/2} \cdot \mathrm{diam})$。对于 $c(\sigma)$ 较大的序列，该概率非常接近 1，并且 β 和成本增加均为 $o(c(\sigma))$，因此与总成本相比可以忽略不计。因此，定理 2 与定理 1 类似，证明了在在线设置中，可以实现强公平性而不会显著影响竞争性。参数 γ在离线结果中并未出现，它在 β 、成本增加和成功概率之间权衡。

证明概述。为了证明定理 2，我们采用随机服务器交换技术，方法与第 1.1.1 节中实现事前（即期望）公平性的做法类似。不过，为了给出一种能在任意输入序列上以高概率实现公平性的算法，我们进行一系列交换，而不是只交换一次，并在每次交换时随机排列服务器。我们通过集中不等式论证，服务器的总负载以高概率相近。

难点在于确定交换的频率——交换过多会使原算法的成本大幅增加，而交换过少则意味着无法以高概率保证公平性。为了平衡这一取舍，我们提出了一种分阶段算法（算法 2），每当累计总成本增加到足够大的倍数时就进行交换。例如，在 $\gamma = 1$ 的情况下，算法在累计成本达到 1 后交换，之后每当累计成本再增加 2、再增加 3，依此类推时，就再次交换。累计成本约为 $\sum_{i=1}^{s} s = \Omega(s^2)$ 后，它会进行s 次交换。因此，在累计成本达到 $c(\sigma)$ 后，算法已经进行了 $O(c(\sigma)^{1/2})$ 次交换，而每次交换之间产生的总成本至多为 $O(c(\sigma)^{1/2})$。这样，我们便可以证明，与交换相关的额外加性成本以及服务器负载的不平衡，以高概率均被界定在 $O(c(\sigma)^{1/2})$ 以内，因此与总成本 $c(\sigma)$ 相比可以忽略不计。我们通过定理 2 中的参数 γ 来权衡成功概率与成本增幅——大致而言，选择成本增幅为 i^γ 的阶段 i ，使得经过 s 次交换后产生的总成本为 $\Omega(s^{1+\gamma})$。增大 γ 会减少交换次数，但也会降低成功满足公平性保证的概率。

1.1.4 迈向确定性在线公平性

定理1和定理2证明，通过随机化并对抗 oblivious 对手，可以在离线场景和在线设置中实现 k-服务器问题的公平性。很容易验证，定理 2 对于自适应在线对手同样成立；该对手可以根据每一轮的服务器位置更改输入，

度量	算法	竞争比	公平性	参考文献
Line	DCA	k [CL91]	(1， $O(k \cdot$直径))	定理 3
含有 $k = 2$ 的树	DCA	$k(= 2)$ [CL91]	(1。5， $1.75 \cdot$直径)	定理 4
均匀度量（分页）	FIFO	k [ST85]	(1,1)	定理 6
含有 $k + 1$ 个点的度量	BALANCE	k [MMS90]	(1，直径)	命题 1

表 1：不同度量下确定性公平在线算法的总结 s

但无法根据服务器的 ID 区分服务器（因此无法判断是否发生了服务器的随机交换）。然而，能否对抗更强的在线对手实现公平性，或者是否能以确定性方式实现公平性，仍是一个开放问题。例如，以下问题尚未解决：

是否存在一种确定性在线 k-服务器算法，在一般度量下具有 O（k）-竞争性，同时对于 $\alpha = O$（1）和 β 也是（α, β）-公平的，且仅取决于 k 和直径？

为了推进上述问题的研究，我们在第 5 节考察了若干在特定度量类别上具有竞争性的、广泛研究的确定性在线 k-服务器算法的公平性属性。具体而言，我们研究了双重覆盖算法（DCA）的公平性，该算法已知在树度量下具有 k-竞争性。我们还研究了几种适用于均匀度量（即经典分页问题）的 k-竞争性确定性算法的公平性。这些算法包括先进先出（FIFO）算法，以及任何标记算法，其中包括最近最少使用（LRU）算法。

树度量。 我们首先给出 DCA 的两个正面结果。我们证明，该算法在线度量（即底层图为直线的树度量）下具有强公平性。具体而言：

定理 3 (定理 10 重述)。 对线度量上任意输入序列 σ 运行 *DCA* 算法，所得解中任意两台服务器支付的成本之差的绝对值为 $O(k \cdot$直径)。

定理 3 意味着，DCA 在线度量上是 (1， $O(k \cdot$直径))-公平的。证明相当直接：在线上，服务器可以按从左到右的顺序排列。我们很容易证明，任何旨在最小化总成本的算法（包括 DCA）都不会改变它们的相对顺序。因此，每台服务器都会左右移动，并始终服务于落在它与左右相邻服务器之间（或直线端点之一）的请求。DCA 将相邻服务器的移动联系起来：每当请求落在两台服务器之间时，两台服务器都会朝请求移动。由于这种联动，我们可以论证，相邻服务器之间的成本差仅取决于它们的净移动距离，而该距离不超过直径。对所有 k 台服务器之间的差距进行累加，可得成本的最大总体差距为 $O(k \cdot$直径)。

我们还证明，对于一般树度量上的双服务器问题，当 $\alpha = O(1)$ 和 $\beta = O($直径) 时，DCA 是 (α, β)-公平的。特别地：

定理 4 （重述定理 11）。 对于双服务器问题，在直径为 直径 的树度量上运行的 *DCA* 算法，对任意输入序列 σ 都是 $(1.5, 1.75 \cdot$ 直径$)$-公平的。

我们通过将每台服务器承担的总成本拆分为两部分来证明定理 4：当服务器远离另一台服务器移动时产生的发散成本，以及当服务器朝另一台服务器移动时产生的收敛成本。我们在引理 4 中证明，这两种成本都占服务器所承担总成本的一个常数比例。由于 DCA 会将收敛成本关联起来（当一台服务器朝另一台移动时，另一台服务器也会移动），因此我们可以据此论证，两台服务器承担的成本大致相差一个常数乘法因子以内。

尽管已有上述正面结果，我们还是证明，令人意外的是，DCA 在一般树度量上可能表现出显著的不公平性，导致一台服务器承担大部分总工作量。

定理 5 (定理 12 重述)。 存在一种树度量及其上的输入序列 σ ，使得 *DCA* 输出的解中，单台服务器承担的成本为$\Omega(k \cdot \mathrm{OPT}(\sigma))$，其中$\mathrm{OPT}(\sigma)$ 是请求序列 σ 的离线最优成本。

注意到，已知 DCA 在一般树度量上具有 k 竞争性，因此其总成本为 $\leq k \cdot \mathrm{OPT}(\sigma)$。定理 5 表明，单个服务器就可能承担这项成本的一个常数比例，因此对于任意 $\alpha = o(k)$ 和固定的 β, DCA 都不是 (α, β)-公平的。

困难的情况是树度量类似于一条直线：这棵树由一串主要 节点构成，每个节点都连接着一个距离极近的次要 叶节点。请求在直线上的中点和附近的次要节点之间交替出现，反复迫使最左侧的服务器穿越整个结构，越过其他已被移至次要节点、以便让它能够“通过”的服务器。尽管 DCA 的整体竞争性为 k，这仍会导致最左侧的服务器承担 $\Omega(k \cdot \mathrm{OPT})$ 的成本。

均匀度量。接下来，我们将注意力转向均匀度量——其中所有点彼此之间的距离都相同。这种 k-服务器问题也称为分页问题。我们为现有算法证明了若干积极结果。为简单起见，假设度量经过缩放，使所有点之间的距离均为 1。首先，我们证明经典的先进先出（FIFO）算法在均匀度量上具有 k 竞争性 [ST85], ，且达到了几乎最强的公平性：由于该算法只需让服务器（即缓存位置）依次轮换，便可处理任何尚未被覆盖的到达请求（即填补任何缓存未命中），因此所有服务器承担的成本都相同，最多相差一个加性因子。形式化地说，

定理 6 （重述定理 13）。 *FIFO* 是 （1，1）-公平。

接下来我们指出（定理 14），广受欢迎的最近最少使用（LRU）算法并不具备如此强的公平性保证：存在一些输入实例，使得某台服务器承担了该算法几乎全部的成本（也就是说，某个缓存槽几乎在每次缓存未命中时都会被重新加载）。不过，LRU，实际上包括更广泛的一类标记算法，仍然实现了一种有趣的公平性概念：在请求序列 σ 上，没有任何服务器承担的成本超过 $\mathrm{OPT}(\sigma)$。换言之，在竞争比为 k的最坏情况输入序列上，没有任何服务器承担的成本超过每台服务器的平均成本。形式化地说：

定理 7（重述定理 15）。 对于均匀度量上的 k-服务器问题，任意给定服务器在输入 σ 上产生的成本至多为 $\text{OPT}(\sigma)$。

这一结果与定理 5 形成对比。例如，定理 5 表明，对于一般树度量上的 DCA，一个服务器的成本可能高达最优离线成本的 k 倍。证明采用了与证明标记算法具有 k竞争比时类似的方法：我们证明，在导致同一缓存位置被重新加载的任意两次缓存未命中之间，至少会请求 k 个不同的文件。因此，包含这两次未命中的序列，即使对最优离线解而言，也至少会导致一次缓存未命中。反复应用这一论证可知，在整个输入序列中，没有任何缓存位置（即服务器）的缺页次数会超过 $\text{OPT}(\sigma)$。

具有 $k+1$ 个点的度量。 最后，我们考虑具有 $k+1$ 个点的度量，这类度量在 k-服务器问题的文献中得到了广泛研究。众所周知，对于任何具有 $k+1$ 个点的度量，确定性算法都无法达到优于 k 的竞争比 [MMS90]。此外，众所周知， BALANCE 算法达到了这一最优竞争比 [MMS90]。回想一下，该算法本质上是公平的：它总是移动这样一台服务器，即在可能服务当前请求之后，其总成本在所有服务器中最低。因此，我们可以得出以下结论：

命题 1. BALANCE 是一个(1,直径)-公平算法，适用于含有 $k+1$ 个点的度量空间。

1.2 路线图

本文其余部分的组织如下。第 2 节介绍 k-服务器问题，并对我们的 (α， β)-公平性概念作形式化定义。我们在附录 A 中探讨其他公平性概念。第 3 节介绍我们的离线 (α， β)-公平算法，第 4 节介绍我们的随机在线算法。第 5 节评估确定性算法在特定度量空间中的公平性，例如 DCA、FIFO 和 LRU。最后，我们在第 6 节提出一些开放问题。

2 问题设置与(α, β)-公平性

本节将正式定义 k-服务器问题，并介绍全文使用的符号。随后，我们将定义所探讨的核心公平性概念。

-服务器问题： k-服务器问题定义在度量空间 (M， d) 上，其中有 k 台服务器，编号为 1、.....、k，初始位置为 $s_1(0)$、.....、 $s_k(0) \in M$。在每个时间步$t = 1$，2、.....、T，都会有一个请求 $\sigma_t \in M$到达。算法必须选择一台服务器，将其移动到请求的位置，并承担等于移动距离的成本——也就是说，如果移动服务器 i，且它先前位于位置 $s_i(t-1)$，则成本为 $d(s_i(t-1)\ \sigma_t)$。总成本是完整请求序列 $\sigma =(\sigma_1$、.....、 $\sigma_T)$ 中各项成本之和。

成本记号： 设 A 是一个 k-服务器算法。对于固定的请求序列 σ，令 $c_i(A, \sigma, t)$ 表示算法 A在时间 t 使服务器 i 移动的距离，并定义 $c_i(A, \sigma) = \sum_{t=1}^{T} c_i(A, \sigma, t)$ 为服务器 i产生的总成本。该算法的总成本为 $\text{cost}(A, \sigma) = \sum_{i=1}^{k} c_i(A, \sigma)$。

我们用 $\text{OPT}(\sigma)$ 表示任何离线算法在输入 σ上能够达到的最小总成本。当 A 和 σ 的含义可由上下文明确时，我们有时会省略它们作为参数，并用 $c_i(t)$、 c_i、成本和 OPT 指代上述量。

竞争比：如果存在一个常数 $C \geq 0$（可能取决于 k 和底层度量空间），使得对于所有输入序列 σ，成本$(A, \sigma) \leq c \cdot \text{OPT}(\sigma) + C.$，则称算法 A 是 c-竞争的。对于随机化算法，我们考虑期望成本；若A 满足 $\mathbb{E}[\text{成本}(A, \sigma)] \leq c \cdot \text{OPT}(\sigma) + C.$，则称其为 c期望意义下 - 竞争的。

k-服务器问题通常在在线设置下研究：算法必须在看到请求 σ_t后、看到任何后续请求之前，于时间 t 作出决策。众所周知，对于一般度量 [，不存在竞争比为 $< k$ 的确定性在线算法 KP95]。这一下界在树度量上由著名的双重覆盖算法（DCA） [CL98], 达到；而工作函数算法（WFA）则在竞争比上相差一个常数因子，它在一般度量空间上是$2k-1$ 竞争的 [KP95, Kou94], ，并且在直线度量、星形度量以及含有 $k+2$ 个点的度量空间上是 k-竞争的 [BK04]。对于面对无知对手的随机化算法，目前已知的上界关于 k 是多对数级的 n [BBMN15, BCL$^+$18]。最近，长期以来关于可达到 $O(\log k)$ 的猜想被推翻；现已证明，在某些度量下，随机化竞争比为 $\Omega(\log^2 k)$ [BRV23]。

2.1 $(\alpha,\ \beta)$-公平性

现在我们定义核心公平性概念：它要求任何单个服务器承担的成本都不会显著超过其总成本的比例份额。

定义 1 ($(\alpha,\ \beta)$-公平性)。 设 A 是一个 k-服务器算法， $w(\sigma)$ 是与请求序列 σ 关联的参考成本基准。若对于所有输入序列 σ 和所有服务器 $i \in [k]$， A 都满足以下条件，则称其关于 w 是 (α, β)-公平的：

$$c_i(A, \sigma) \leq \frac{\alpha \cdot w(\sigma)}{k} + \beta, \tag{1}$$

其中， $c_i(A,\ \sigma)$ 表示算法 A 在输入 σ 上由服务器 i 产生的总成本。

随机化情形。 若对于每个输入序列 σ，随机化算法 A 以概率 $1-\delta$ 关于 w -公平，即：

$$\Pr\left[\forall i \in [k],\ c_i(A, \sigma) \leq \frac{\alpha \cdot w(\sigma)}{k} + \beta\right] \geq 1 - \delta. \tag{2}$$

当 $\delta \leq 1/$ 为 poly($k,\ |\sigma|$) 时，我们只称 A 以高概率为 ($\alpha,\ \beta$)-公平 $ty.$

请注意，我们的一般定义允许相对于选定的基准 w 来表示公平性，例如最优离线成本 $\text{OPT}($ $\sigma)$，或算法自身的总成本 cost($A,\ \sigma$)。

注释 1（公平性与竞争性）。 我们指出，当 $w(\sigma) =$ 成本($A,\ \sigma$)、($\alpha,\ \beta$)-公平性并不是一个有趣的概念，除非算法 A 还被要求具有有界的竞争

比率。否则，我们可以让所有服务器同步移动，从而实现完美公平，但代价是竞争性受损。粗略地说，在在线设置中，给定一个 A 具有α 竞争性的算法，并假设每当一台服务器移动以服务请求 σ_t 时，我们就让所有其他服务器移动相同的距离（但不改变它们的位置），那么就能得到一个 $\alpha \cdot k$-竞争的完美公平算法。因此，我们始终会寻求优于这一简单基准的公平算法。

3 AFair 离线算法

我们首先证明，在离线场景中，可以获得一个成本近似最优且能够在服务器之间实现可证明公平的解。具体而言，我们设计了一种转换（算法 1），它将任意一个（可能不公平的）最优离线解转换为一个对于 $\beta = O$(直径 $\cdot \log\ k/\varepsilon$) 而言（$1+\varepsilon$， β）-公平的解。该转换至多使总成本增加一个加性常数。其核心思想是通过一系列成对交换，在服务器之间重新分配成本，从而逐步平衡负载。该转换简单、确定且具有黑盒特性——不需要底层最优解具备任何特定结构。

定理 8 （（$1+\varepsilon$， β）-公平离线 k-服务器）。 考虑任意一个 k-服务器输入序列 σ，其长度为 T，定义在直径为 直径 的度量空间上。令 $\{s_i\ (t)\ \}_{i\in[k],\ t\in[T]}$ 为一个服务器位置序列，该序列在此输入序列上实现最优成本。

对于任意 $\varepsilon > 0$，算法 1 以 $\{s_i(t)\}_{i\in[k],\ t\in[T]}$ 作为输入，并返回修改后的位置序列 $\{\hat{s}_i(t)\}_{i\in[k],\ t\in[T]}$ 该序列相对于 $\mathrm{OPT}(\sigma)$ 是 $(1+\varepsilon, \beta)$-公平的，满足 $\beta = O\left(\log\ k\cdot$直径$_{\varepsilon}\right)$。此外，输出解的总成本为 $\mathrm{OPT}(\sigma) + O\left(k \log\ k\cdot$直径$_{\varepsilon}\right)$.

证明概要。 算法 1 通过在成本最高和最低的服务器之间进行一系列交换，迭代降低输入解中的最高服务器成本。每次交换都会从选定的时间 z 起互换服务器的剩余工作量。利用平衡引理（引理 3），我们证明，总能选择一种交换，使被交换服务器之间的成本差至多为一个加法直径因子。每次交换都会使新算法的总成本至多增加 $2\cdot$diam。此外请注意，如果解尚不公平，则最高服务器负载至少是最低服务器负载的 $\approx 1+\varepsilon$ 倍。因此，我们可以论证，交换后，两台服务器的成本都比先前的最高成本低约一个 $1/(1+O(\varepsilon))$ 乘法因子（断言 1）。我们还观察到，最高服务器负载总会因交换而下降，且未参与交换的服务器成本保持不变（命题 2）。接下来，我们考察任意一个包含$k+1$ 次交换的窗口。根据抽屉原理，至少有一台服务器会在两次交换中都作为成本最高的服务器出现。在这两次交换之间，该服务器的成本必定降低了一个 $\approx 1/(1+O(\varepsilon))$ 因子。因此，我们可以论证，最大成本每$k+1$ 轮至少会下降一个 $(1+\varepsilon)$ 因子（声明 3）。重复此过程 $O\left(k\cdot\log_{1+\varepsilon} k\right) = O\left(k_{\log_{\varepsilon} k}\right)$ 次交换，以将初始乘法失衡从 k 降低至 $(1+\varepsilon)$，即可得到满足 $(1+\varepsilon, \beta)$-公平性的解。完整细节见附录 C.1。

□

算法 1： 通过成对交换实现离线公平 k-服务器

输入： 最优服务器位置 $\{s_i(t)\}_{i\in[k],\ t\in[T]}$ 及其对应的成本分配 $\{c_i(t)\}_{i\in[k],\ t\in[T]}$ 和参数 ε

输出： 修改后的位置序列 $\{\hat{s}_i(t)\}_{i\in[k],\ t\in[T]}$ 满足 $(1+\varepsilon,\ \beta)$-公平性。

设置轮次计数器 $r \leftarrow 0$

对所有 $i \in [k],\ t \in [T]$，初始化 $\hat{s}_i^{(0)}(t) \leftarrow s_i(t)$ 和 $\hat{c}_i^{(0)}(t) \leftarrow c_i(t)$

$\beta \leftarrow 2\cdot(1+\varepsilon)\cdot diam\cdot(\frac{3}{2}+\log_{\frac{2+2\varepsilon}{2+\varepsilon}} k)$

当 $\max_{i\in[k]} \sum_{t=1}^{T} \hat{c}_i^{(r)}(t) > (1+\varepsilon)\cdot\frac{\mathrm{OPT}(\sigma)}{k} + \beta$ **时**

- 令 $H_r \leftarrow \mathrm{argmax}_{i\in[k]} \sum_{t=1}^{T} \hat{c}_i^{(r)}$ ({v4}); // 负载最重的服务器
- 令 $L_r \leftarrow \arg\min_{i\in[k]} \sum_{t=1}^{T} \hat{c}_i^{(r)}(t)$; // 负载最轻的服务器
- 令 $z \in [T]$ 为任意一个索引，使得 H_r 和 L_r 的总成本之差在时间 z 交换后小于直径; // 这样的 z 总是存在（引理 3）
- 为所有 $i,\ t$ 初始化当前成本/位置：$\hat{c}_i^{(r+1)}(t) \leftarrow \hat{c}_i^{(r)}(t),\ \hat{s}_i^{(r+1)}(t) \leftarrow \hat{s}_i^{(r)}(t)$
- **对于** $t > z$，**执行**
  - $\hat{s}_{H_r}^{(r+1)}(t) \leftarrow \hat{s}_{L_r}^{(r)}(t),\ \hat{s}_{L_r}^{(r+1)}(t) \leftarrow \hat{s}_{H_r}^{(r)}(t)$; // 在 z 之后交换位置
  - $\hat{c}_{H_r}^{(r+1)}(t) \leftarrow \hat{c}_{L_r}^{(r)}(t),\ \hat{c}_{L_r}^{(r+1)}(t) \leftarrow \hat{c}_{H_r}^{(r)}(t)$; // 交换 z 之后的成本
- $\hat{c}_{H_r}^{(r+1)}(z+1) \leftarrow d(\hat{s}_{H_r}^{(r+1)}(z+1), \hat{s}_{H_r}^{(r+1)}(z))$; // 交换惩罚
- $\hat{c}_{L_r}^{(r+1)}(z+1) \leftarrow d(\hat{s}_{L_r}^{(r+1)}(z+1),\ \hat{s}_{L_r}^{(r+1)}(z))$; // 交换惩罚
- 更新轮次：$r \leftarrow r+1$

返回 $\{\hat{c}_i^{(r)}(t)\}$ 和 $\{\hat{s}_i^{(r)}(t)\}$

4 一种公平的随机化在线算法

接下来，我们展示如何在在线设置中使用随机算法，在尽可能减少竞争性损失的同时实现公平性。与算法 1 类似，我们假设可以使用某个已有的在线k-服务器算法 A，其竞争比有界，但可能不提供任何公平性保证。我们使用一系列随机服务器交换来平衡服务器负载，使算法实现公平。

在每个交换点，我们随机排列所有服务器标识符，按照该排列交换服务器位置，并在之后也按照该排列重新分配各服务器的工作负载。因此，在两次交换之间，根据对称性，所有服务器产生的期望成本相同。借助集中不等式，只要交换足够频繁，我们就能证明所有服务器以高概率承担相近的成本，从而实现 $(\alpha,\ \beta)$-公平性。关键挑战在于确定交换的频率——交换次数过多会大幅增加原算法的成本，而交换次数过少则无法保证公平性以高概率成立。为平衡这一权衡，我们引入一种分阶段算法（算法 2），每当产生的总成本增加到足够大的倍数时就进行交换。我们证明如下：

定理 9（随机在线公平性保证）。设 A 是任意在线 k-服务器算法，σ 是任意请求序列。那么，对于输入 A 且参数为 $\gamma > 0$ 的算法 *2*，满足：对于

算法 2：通过分阶段随机交换实现在线公平 k-服务器 输入：在线请求序列 $\sigma = (\sigma_1, \sigma_2, \ldots\ldots)$、$k$-服务器算法 A、初始服务器配置（$s_1(0)\ldots\ldots, s_k(0)$）、阶段指数 $\gamma > 0$1 初始化：阶段编号 $\ell \leftarrow 1$、阶段预算 $\phi_\ell \leftarrow \ell^\gamma$ 和累积成本 $U_\ell \leftarrow 0$。 2 通过随机排列 π_ℓ 重新放置服务器：对于所有 $i \in [k]$，设置 $s_i(0) \leftarrow s_{\pi_\ell(i)}(0)$，其中 π_ℓ 是均匀随机选取的。 3 对于每个请求 σ_t，执行 4 使用重新放置前标记的服务器，通过算法 A 服务 σ_t。$\{s_{\pi_\ell^{-1}(i)}(t)\}$. 5 更新 $U_\ell \leftarrow U_\ell + cost(A, t)$。 6 如果 $U_\ell \geq \phi_\ell$，则7 $\ell \leftarrow \ell + 1$、$\phi_\ell \leftarrow \ell^\gamma$ 和 $U_\ell \leftarrow 0$。 8 通过随机排列 π_ℓ 重新放置服务器：对于所有 $i \in [k]$，设置 $s_i(t) \leftarrow s_{\pi_\ell(i)}(t)$，其中 π_ℓ 是均匀随机选取的。

任意 $\varepsilon > 0$，以至少 $1 - k \cdot \exp\left(-\Omega\left(\frac{\varepsilon^2}{\gamma \cdot k} \cdot cost(A,\sigma)^{1/(\gamma+1)}\right)\right)$ 的概率，每台服务器的成本至多为

$$(1+\varepsilon) \cdot \frac{cost(A,\sigma)}{k} + O\left(cost(A,\sigma)^{1/(\gamma+1)} \cdot \mathrm{diam}\right).$$

此外，所有服务器的总成本有界于 $cost(A,\sigma) + O\left(cost(A,\sigma)^{1/(\gamma+1)} \cdot k \cdot \text{直径}\right)$。

就 (α, β)-公平性而言，定理 9 意味着算法 2 关于 $\mathrm{cost}(A,\sigma)$ 是 (α,β)-公平的，对于 $\alpha = 1+\varepsilon$，$\beta = O\left(\mathrm{cost}(A,\sigma)^{1/(\gamma+1)} \cdot \text{直径}\right)$，这一结论以高概率成立。它付出的总成本惩罚为 $O\left(\mathrm{cost}(A,\sigma)^{1/(\gamma+1)} \cdot k \cdot \text{直径}\right)$。$\beta$ 和这一总成本增量都随 $\mathrm{cost}(A,\sigma)$ 次线性增长，因此可视为低阶项。成功概率取决于 k、γ 和 ε，但当 $\mathrm{cost}(A,\sigma)$ 趋于无穷时，它趋于 1。

证明概要。 算法 2 将输入序列划分为若干阶段，每当基础算法 A 累积的成本至少达到 $\phi_\ell = \ell^\gamma$ 时（第 7 行），当前阶段便结束。每个阶段开始时（第 8 行），算法会对服务器身份进行随机排列，并交换它们的位置。这样可以确保，无论服务器此前的历史如何，每个服务器分摊到的阶段成本期望值都相同。

令 m 表示阶段数。由于每个阶段产生 $\phi_\ell = \ell^\gamma$ 成本，且 $\sum_{\ell=1}^{m} \ell^\gamma = \Theta(m^{\gamma+1})$，我们有：$m = \Theta\left(\text{成本}(A)^{1/(\gamma+1)}\right)$. 因此，每个服务器的预期成本为：

$$\mathbb{E}[c_i] = \frac{\mathrm{cost}(A)}{k} + O\left(m \cdot \mathrm{diam}\right) = \frac{\mathrm{cost}(A)}{k} + O\left(\mathrm{cost}(A)^{1/(\gamma+1)} \cdot \mathrm{diam}\right).$$

为了证明高概率公平性，我们对表示每个服务器每阶段成本的随机变量应用集中不等式。每个变量的上界为 ϕ_ℓ，而最大阶段成本为 $\phi_m = m^\gamma = \Theta\left(\text{成本}(A)^{\gamma/(\gamma+1)}\right)$。因此，我们可以将这些随机变量的总方差界定为 $\sigma^2 = O\left(\sum \phi_\ell^2/k\right)$。

应用伯恩斯坦不等式及这些上界可得，每台服务器的总成本仍以至少 $1 - \exp\left(-\Omega\left(\frac{\varepsilon^2}{\gamma \cdot k} \cdot \mathrm{cost}(A,\ \sigma)^{1/(\gamma+1)}\right)\right)$ 的概率与其期望相差不大（$1+\varepsilon$）。

对所有 k 服务器应用并集界，可得最终的高概率保证：以至少$1 - k \cdot \exp\left(-\frac{\varepsilon^2}{\gamma \cdot k} \cdot \Omega\left(\text{cost}(A,\ \sigma)^{1/(\gamma+1)}\right)\right)$ 的概率，对所有 $i \in [k]$,

$$c_i \leq (1+\varepsilon) \cdot \frac{\text{cost}(A)}{k} + O\left(\text{cost}(A)^{1/(\gamma+1)} \cdot \text{diam}\right).$$

证明的完整细节请参见附录 C.3。 □

备注 2（扩展到 ID-Oblivious 对手）。 我们注意到，当 σ由对 *ID* 不敏感的 自适应对手生成时，定理 9 仍然成立——也就是说，对手可以根据轮次 $t' < t$ 时服务器的位置选择 σ_t ，但不能根据服务器的身份进行选择。在每个阶段开始时，算法 2 会对服务器 ID 应用一个随机排列，同时不改变其位置的总体集合。因此，ID-Oblivious 对手在算法 2 下观察到的位置序列与在基础算法 A 下观察到的相同，所以针对 oblivious 对手情形的分析可以不加修改地沿用。

5 迈向确定性在线公平性

我们的随机公平性结果留下的一个核心开放问题是：公平性是否能够通过确定性的 在线算法实现，或者能否在观察服务器身份的完全自适应对手下实现。本节研究特定度量类别上经典确定性 k-服务器算法的公平性属性。我们重点关注树度量上的双重覆盖算法（DCA），以及均匀度量上的确定性分页算法（FIFO、LRU 和标记算法）。从竞争性分析的角度来看，这些算法都已得到充分研究；分析它们的公平性属性，有助于我们了解哪些类型的公平性可以通过确定性方式实现。

5.1 DCA 在特殊情形下的公平性

双重覆盖算法（DCA）适用于树度量，并会将所有“未受阻”的服务器以相同速度同时朝请求点移动；这些服务器的路径上没有其他服务器，移动持续到该请求得到服务为止。

我们首先证明，由于所有未受阻服务器同时移动这一机制本身具有公平性，对于任意 k ，该算法在线度量上都是公平的；对于 $k = 2$，该算法在一般树度量上也是公平的。然而，我们还证明了一个强烈的负面结果：对于一般的 k，DCA 在一般树度量上可能极不公平。

我们先给出正面结果：对于任意 k，DCA 在线度量上相对于 $\textsc{Opt}$ 是 $(1, O(k \cdot \text{直径}))$-公平的。

定理 10. 设 σ 是直径为 直径 的线度量上的任意请求序列，并令 c_i 表示服务器 i 使用 *DCA* 服务 σ 时产生的总成本。则有 $\max_{i,\ j \in [k]} |c_i - c_j| = O(k \cdot \text{直径})$。因此，*DCA* 在线度量上是 $(1, O(k \cdot \text{直径}))$-公平的。

证明概要。 由于 DCA 不会让服务器在直线上彼此越过，我们可以根据服务器从左到右的初始位置，为它们确定唯一的编号：$1, \ldots, k$。双重

覆盖方法确保，对于每个请求，距离请求最近的两台服务器会以相同的速度朝请求移动，直到其中一台到达请求位置。因此，每当服务器 i 向右移动时，服务器 $i+1$都会相应地向左移动同样的距离，反之亦然。由此得到递推关系：

$R_i = L_{i+1}$，</style>其中 是服务器 的向右移动，是其向左移动。

.....的总成本　　服务器 i 的成本为 $c_i = R_i + L_i$。令 $d_i = R_i - L_i$ 为服务器 i的净移动距离，

$$c_{i+1} - c_i = L_{i+1} + R_{i+1} - (L_i + R_i) = R_{i+1} - L_i = d_{i+1} + d_i.$$

由于所有移动都沿着直径为 diam 的直线进行，因此对于所有 i，都有 $|d_i| \le \text{diam}$。因此，相邻服务器之间的成本差异有如下界：

$$|c_{i+1} - c_i| \le |d_i| + |d_{i+1}| \le 2 \cdot \text{diam}.$$

将所有 $k-1$ 对服务器之间的两两差异相加，可得：

$$\text{for any } i, j \in [k] \ |c_i - c_j| \le 2(k-1) \cdot \text{diam},$$

如上所述。完整证明见附录 C.4。 □

接下来我们将证明，在 $k = 2$ 的特殊情况下，对于任意树度量，DCA 对 $\alpha = O(1)$ 和 $\beta = O$(直径) 是 (α， β)-公平的。

定理 11。 当 $k = 2$ 时，对于任意树度量，*DCA* 是 (1.5， $1.75 \cdot$ 直径)-公平的。

证明。 固定一个输入 σ。对于每台服务器，令 $con_i(\sigma)$ 表示 i 在向另一台服务器“汇聚”（向另一台服务器移动得更近）过程中移动的总距离。同样，令 $div_i(\sigma)$ 表示 i 在远离另一台服务器“发散”过程中移动的总距离。例如，当请求位于 i 和 i' 之间的最短路径上时，两台服务器中任意一台移动的距离都会计入 $con_i(\sigma)$。当 i 远离另一台服务器移动（而另一台服务器不移动）时， i 移动的距离会计入 $div_i(\sigma)$。

直观地说，对于两台服务器，发散项和收敛项直接反映了它们之间的距离如何随时间变化：发散时，两台服务器彼此远离；收敛时，它们彼此靠近。由于它们之间的距离永远不会超过度量的直径（也不会小于零），因此发散超过收敛的总量，或收敛超过发散的总量，都必然有上界 diam。

形式上，对于任意输入序列 σ，以及 $k = 2$ 和任意一台服务器 i，以下不等式成立（证明见附录 C.5 中的引理 4）。

$$-\text{diam} \le div_i(\sigma) + div_{i'}(\sigma) - 2con_i(\sigma) \le \text{diam}. \tag{3}$$

由于每个单位　服务器 i 的移动要么增加、要么减少其与另一台服务器之间的距离　　，

我们可以写作： $c_i = con_i(\sigma) + div_i(\sigma)$。此外，根据 DCA 的定义和汇聚移动的性质，我们有 $con_i(\sigma) = con_{i'}(\sigma)$，因为在汇聚移动过程中，两台服务器以相同的速度彼此靠近（对于 $k > 2$，这一论证不成立，因为一台服务器可能会被第三台服务器阻挡而无法移动）。因此，利用（3）中的不等式，我们可以得出：

$$\begin{aligned} c_i = con_i(\sigma) + div_i(\sigma) &\leq 3 \cdot con_i(\sigma) - div_{i'}(\sigma) + \text{diam} \\ &\leq 3 \cdot con_i(\sigma) + \text{diam} \\ &= 1.5 \cdot con_i(\sigma) + 1.5 \cdot con_i(\sigma) + \text{diam} \\ &\leq 1.5 \cdot con_i(\sigma) + 0.75 \cdot (div_i(\sigma) + div_{i'}(\sigma)) + 1.75 \cdot \text{diam}. \end{aligned}$$

将 c_i 和 $c_{i'}$ 的定义以及 $con_i(\sigma) = con_{i'}(\sigma)$ 这一事实代入前一个界，得到：

$$c_i \leq 0.75 \cdot c_i + 0.75 \cdot c_{i'} + 1.75 \cdot \text{diam} \leq 1.5 \cdot \frac{cost(DCA, \sigma)}{2} + 1.75 \cdot \text{diam}.$$

□

5.2 DCA 在一般树度量上的不公平性

接下来我们将说明，遗憾的是，定理 10 和定理 11 的正面结果无法推广到 DCA 具有 k-竞争性的所有情形。具体而言，对于一般的 k，DCA 在一般树度量上可能无法满足任何合理的公平性概念：它会将大部分工作——事实上是 $\Omega(k \cdot \text{OPT}(\sigma))$ 的工作量——分配给单个服务器。

定理 12. *存在一个树度量及其上的请求序列 σ，使得在该度量上运行双重覆盖算法（DCA）时，σ 中某个服务器的成本为 $\Omega(k \cdot \text{OPT}(\sigma))$，其中$\text{OPT}(\sigma)$ 表示离线最优解的成本。*

证明概要。 我们构造一棵树，其中 $k+1$ 个主要节点构成一条路径（“主干”），且每个主要节点都连接一个距离为 ε 的次要叶节点，其中 ε 可任意小。将主干节点标记为 $\{1, 2, \ldots, k+1\}$，并将服务器标记为 $s_1, s_2, \ldots, s_k$，初始时分别放置在位置 $1, 3, 4, \ldots, k+1$，跳过位置 2 以形成一个间隙。目标是迫使 s_1 从主干起点移动到终点，产生 $\Omega(k)$ 的成本，而所有其他服务器（以及离线最优解）的成本仅为 $O(1)$。困难的请求序列由重复的步骤构成：

- 首先，在位置 2 发出一个请求，导致位于位置 1 和 3 的两台服务器 s_1 和 s_2 向请求移动，将距离扰动一个任意小的量，使得 s_2 先到达并实际处理该请求。

- 接下来，请求与位置 2 相连的次要叶节点。这会迫使 s_2 离开主干，从而空出位置 2。 s_1 被 s_2 “ 阻挡 ”，因此不会移动。

- 现在，在位置 3 发出请求；位于位置 2 的 s_1 和位于位置 4 的 s_3 会朝它移动，而另一个服务器（此时为 s_3）再次处理该请求，并被移出主干。

这一模式会重复出现：服务器 s_1 缓慢向右移动，在其他服务器被引导离开主路径的同时，逐一经过每台服务器。经过 k 步后， s_1 产生 $\Omega(k)$ 的成本，而所有其他服务器产生的成本仅为 $O(1+\varepsilon)$。图1对此进行了说明。

此外，离线最优值只需进行一次长度为 1 的服务器移动即可处理主干上的所有请求（s_1 在第一步移动到位置 2），并以 ε 次移动处理次要叶节点请求。因此，$\mathrm{OPT}(\sigma) = O(1+\varepsilon k)$，于是令 ε 任意小， s_1 产生的成本为 $\Omega(k \cdot \mathrm{OPT}(\sigma))$，正如所需。完整细节见附录 C.6，其中还给出了一个论证，说明可以重复这一困难序列，以适用于 $\mathrm{OPT}(\sigma)$ 相对于 k 和度量空间直径任意大的情况。

□

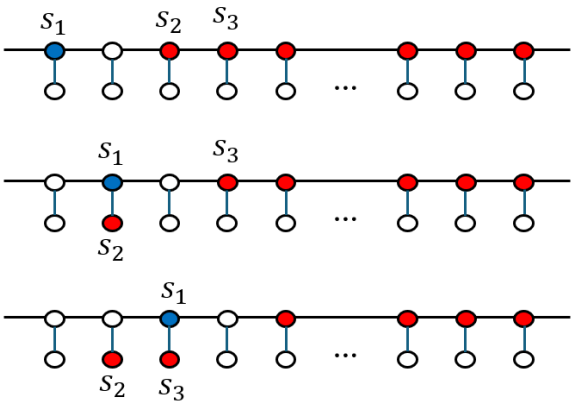

图1：定理 12 的困难实例。顶部： 主干上服务器位置的初始化。中部： 请求 1–2 使 s_1 向右移动一步，并通过一个次要叶节点将 s_2 推离主干。底部： 请求 3–4 使用 s_3重复这一过程，依此类推，迫使 s_1 遍历整个主干，同时所有其他服务器产生 $O(1)$ 的额外成本。

5.3 均匀度量与分页

最后，我们考察均匀度量下确定性 k-服务器算法的公平性。为简化起见，我们将所有距离缩放为 1。此设置等价于经典分页问题，其中 k 服务器对应于缓存中的 k 内存单元。当请求的页面当前不在缓存中时，就会发生缺页故障；等价地说，当请求发往一个没有服务器驻留的节点（页面）时，就会发生缺页故障。因此，最小化服务器移动总量的目标等价于最小化缺页次数。

鉴于这种等价关系，我们采用分页领域的术语。为简化讨论，我们只关注惰性分页算法，这类算法仅在发生缺页故障且缓存已满时才会淘汰页面。非惰性算法（例如满时刷新）在 k-服务器背景下也同样相关，

我们先来看关于先进先出（FIFO）算法的一项重要正面结果。该算法会淘汰在缓存中停留时间最长的页面。众所周知，FIFO 具有k竞争性 [ST85]。我们现在证明，FIFO 还保证了最强意义上的公平性。

定理 13. *FIFO* 在均匀度量下是(1,1)-公平的。

证明。 设 A 和 B 是两个服务器（内存单元），并令 a_i, b_i 分别表示输入在 A 和 B处第 i次缺页故障的索引。假设 A 处的第一次缺页故障早于B处的第一次缺页故障。根据 FIFO 规则，对于任意 i，有

$$a_i < b_i < a_{i+1} < b_{i+1}.$$

这意味着 A 处的缺页次数至多比 B 处多一次。将这一推理应用于所有服务器对，即可得到 (1,1)-公平性保证。 □

因此，FIFO 不仅具有最优竞争性（在确定性算法中），而且公平性也最优。我们可能会问，其他页面置换算法（例如最近最少使用（LRU））是否也具有这一公平性。LRU 同样是 k-竞争的 [ST85], 。下面的定理表明并非如此。回想一下，LRU 会从缓存中淘汰最近最少使用的页面。

定理 14. 存在一个均匀度量上的请求序列 σ，使得在 *LRU* 下，一个服务器的成本为 Ω(cost(LRU σ))，而所有其他服务器的成本都只有 $O(1)$。因此，对于任意 $\alpha = o(k)$ 以及任意 $\beta = o(\text{cost}(\text{LRU } \sigma))$，*LRU* 相对于自身总成本都不满足$(\alpha, \beta)$-公平性。

证明。 我们构造一个输入，使得单个服务器承担 LRU 产生的几乎全部成本。考虑输入序列

$$(\alpha, 1, 2, \ldots, k-1, \beta, 1, 2, \ldots, k-1)^m.$$

根据 LRU 规则，页面 1、2、.....、$k-1$ 始终留在缓存中，分别位于位置 2、.....、k（即由服务器 s_2、.....、s_k覆盖）。第一个缓存位置（服务器 s_1）在请求 β 时发生缺页故障，而在重复该序列后，则在请求 α时发生缺页故障。因此，除最初的 $k-1$ “冷缺页”（缓存填满前发生的缺页故障）之外，算法的所有成本都由这台服务器承担。

□

尽管上述结果是负面的，但可以证明，LRU 以及更一般而言所有标记算法都能保证一种较弱但仍有意义的公平性：每台服务器承担的成本不超过最优离线解。也就是说，相对于 $k \cdot \text{OPT}(\sigma)$，它们是 (1，0)-公平的； $k \cdot \text{OPT}(\sigma)$ 是算法在最坏情况下、面对使其具有 k-竞争性的输入时可能承担的最高成本。我们在附录 A 中进一步探讨这一概念，并将其称为可接受公平性 （定义 4）。

标记算法的工作方式如下：命中时，只需将所请求的页面标记。发生缺页故障时，如果所有页面都已标记，则清除所有标记；然后驱逐一个未标记的页面

并插入所请求的页面并将其标记。LRU 符合这一框架，因为其“最近最少使用”概念与未标记页面完全对应：某个页面恰好在最近的 k 个不同请求中未被请求时，就会失去其标记，这与经典的标记定义相吻合。

定理 15. 对于均匀度量上的任意标记算法，在输入序列 σ 上，每台服务器产生的成本至多为 $\mathrm{OPT}(\sigma)$。

证明. 我们采用标准的阶段划分技术。固定输入序列 σ，将包含 k 个不同页面的最长前缀定义为第一阶段。然后在序列剩余部分上递归地定义后续阶段。设 m_σ 为阶段数。由于每个阶段都会引入 $k+1$ 个不同页面，因此至少会使最优值产生一次缺页故障，于是 OPT(σ) $\geq m_\sigma$。另一方面，在每个阶段中，标记算法的任意一台服务器至多产生一次缺页故障：在该阶段首次发生缺页故障后，其页面便会被标记，并在该阶段剩余时间内受到保护。因此，每台服务器承担的成本至多为 $\textsc{Opt}(\sigma)$。

□

6 结论

在这项工作中，我们对 k-服务器问题中的公平性开展了形式化研究，并引入（α，β）-公平性作为统一框架，用以刻画服务器之间公平的成本分布。在此框架下，我们证明，无须显著牺牲竞争性，也能实现强有力的公平性保证。具体而言，我们提出了一种离线变换，在总成本仅受控增加的情况下实现近乎完美的公平性；还提出了一种随机在线算法，能够以高概率在 oblivious 对手面前实现公平性。

除了算法设计之外，我们还分析了几种经典在线算法的公平性表现。我们的结果表明，尽管 DCA 等算法具有最优或近乎最优的竞争比，但它们在一般度量空间中仍可能表现出显著的不公平性，这凸显了在线设置中竞争性与公平性之间固有的矛盾。

仍有若干开放问题有待探讨。确定性在线算法在一般度量上，或面对完全自适应的对手时，能否实现与我们的随机化结果相当的公平性保证？公平性界对结构参数的依赖还能否进一步降低？更广泛地说，研究其他重要推广中的 (α，β)-公平性也很有意义，例如度量任务系统 [BLS92]。我们希望这项工作能为进一步研究公平性作为在线算法设计中的首要目标奠定基础。

致谢

本研究得到美国国家科学基金会资助（项目编号 2045641、2325956、2512128 和 2533814），以及加拿大自然科学与工程研究委员会（NSERC）的资助（资助参考编号 RGPIN-2025-07295）。

参考文献

[A+97] M. Ananthanarayanan 等人。《分布式 k-服务器问题： k-服务器算法的竞争性分布式转换器》。发表于《算法杂志》，23(2):241–264，1997年。[AAN+98] Mikl´os Ajtai、James Aspnes、Moni Naor、Yuval Rabani、Leonard J. Schulman 和 Orli Waarts。《调度中的公平性》。发表于《算法杂志》，29(2):306–357，1998年。[AC21] Nikhil Ayyadevara 和 Ashish Chiplunkar。《加权 k-服务器的随机化竞争比至少为指数级》。载于第29届欧洲算法研讨会（ESA）论文集，LIPIcs 第204卷，第9:1–9:11页，2021年。[ACE+23a] Antonios Antoniadis、Christian Coester、Marek Elias、Adam Polak 和 Bertrand Simon。《在线度量算法的混合预测》。载于第40届国际机器学习会议（ICML）论文集，《机器学习研究论文集》第202卷，第969–983页，2023年。[ACE+23b] Antonios Antoniadis、Christian Coester、Marek Elias、Adam Polak 和 Bertrand Simon。《具有不可信预测的在线度量算法》。发表于 ACM 算法汇刊，19(2):19:1–19:34，2023年。[ACS24] Nikhil Ayyadevara、Ashish Chiplunkar 和 Amatya Sharma。《加权 k-服务器问题的分解方法》。载于第44届软件技术与理论计算机科学基础国际会议（FSTTCS）论文集，LIPIcs 第323卷，2024年。[AEKM19] Sara Ahmadian、Alessandro Epasto、Ravi Kumar 和 Mohammad Mahdian。《避免过度代表的聚类》。载于第25届 ACM SIGKDD 国际知识发现与数据挖掘会议（KDD 2019）论文集，第267–275页，2019年。[AFSV24] Haris Aziz、Rupert Freeman、Nisarg Shah 和 Rohit Vaish。《两全其美：资源分配中的事前与事后公平性》。发表于《运筹学》，72(4):1674–1688，2024年。[AS22] Elad Aigner-Horev 和 Erel Segal-Halevi。《二分图中的无嫉妒匹配及其在公平分配中的应用》。发表于《信息科学》，587:164–187，2022年。[BBMN15] Nikhil Bansal、Niv Buchbinder、Aleksander Madry 和 Joseph Naor。《一种用于 k-服务器问题的多对数竞争算法》。发表于 Journal of the ACM，62(5):1–49，2015年。[BCL+18] Sebastien Bubeck、Michael B. Cohen、Yin Tat Lee、James R. Lee 和 Aleksander Madry。《通过多尺度熵正则化求解 k-服务器问题》。载于论文集

第50届 ACM 计算理论年会（STOC）论文集，页码 3–16，2018 年。[BCN22] Niv Buchbinder、Christian Coester 和 Joseph Naor。通过双重覆盖和时间反向原始-对偶方法解决在线 k-出租车问题。《数学规划》，197(2):499–527，2022 年。[BE98] Allan Borodin 和 Ran El-Yaniv。《在线计算与竞争分析》。1998 年。第 10 章。[BEKN23] Nikhil Bansal、Marek Elias、Grigorios Koumoutsos 和 Jesper Nederlof。均匀度量下广义 k-服务器问题的竞争性算法。《ACM 算法汇刊》，19(1):8:1–8:15，2023 年。[BK04] Yair Bartal 和 Elias Koutsoupias。关于工作函数算法在 k-服务器问题中的竞争比。《理论计算机科学》，324(2–3):337–345，2004 年。[BKKK17] Hans-Joachim B¨ockenhauer、Dennis Komm、Rastislav Kr´alovic 和 Richard Kr´alovic。关于 k-服务器问题的建议复杂度。《计算机与系统科学杂志》，86:159–170，2017 年。[BLS92] Allan Borodin、Nathan Linial 和 Michael E. Saks。度量任务系统的最优在线算法。《Journal of the ACM》，39(4):745–763，1992 年。[BMM24] Eric Balkanski、Will Ma 和 Andreas Maggiori。带有不公平预测的公平秘书问题。载于《第37届神经信息处理系统进展大会（NeurIPS）论文集》，2024 年。[BRV23] S´ebastien Bubeck、Yuval Rabani 和 Sergei Vassilvitskii。随机 k-服务器猜想不成立。《Journal of the ACM》，70(1):1–45，2023 年。[BS06] Vincenzo Bonifaci 和 Leen Stougie。在线 k-服务器路由问题。载于《第4届近似与在线算法国际会议（WAOA 2006）论文集》，页码 83–94，2006 年。[CC26] Tze-Yang Poon Christian Coester。一般度量下的在线 3-taxi。载于《第37届 ACM-SIAM 离散算法年会（SODA）论文集》，2026 年。[CFK+22] Jannik Castenow、Bj¨orn Feldkord、Till Knollmann、Manuel Malatyali 和 Fried-helm Meyer auf der Heide。带偏好的 k-服务器问题。载于《第34届 ACM 算法与体系结构并行性研讨会（SPAA）论文集》，页码 345–356，2022 年。[CHKV23] Ashish Chiplunkar、Monika Henzinger、Sagar Sudhir Kale 和 Maximilian V¨otsch。在线最小最大分页。载于《第55届 ACM 计算理论年会（STOC）论文集》，页码 1545–1565，2023 年。

[CK19] Christian Coester 和 Elias Koutsoupias。在线 k-出租车问题。载于《第51届 ACM 计算理论年会（STOC）论文集》，页码 1136–1147，2019 年。[CL91] Marek Chrobak 和 Lawrence L. Larmore。树上的 k-服务器最优在线算法。《SIAM 计算杂志》，20(1):144–148，1991 年。[CL98] Marek Chrobak 和 Lawrence L. Larmore。服务器问题与在线博弈。载于《在线算法：研究现状》，《计算机科学讲义》第 1442 卷，页码 64–96。1998 年。[CSW23] Nicolas Christianson、Junxuan Shen 和 Adam Wierman。学习增强的度量任务系统的最优鲁棒性—一致性权衡。载于《第26届人工智能与统计国际会议（AISTATS）论文集》，第 206 卷，页码 9377–9399，2023 年。[CV20] Ashish Chiplunkar 和 Sundar Vishwanathan。加权和广义 k-服务器问题的随机无记忆算法。《ACM 算法汇刊》，16(1):14:1–14:28，2020 年。[DEMZ24] John Dickerson、Seyed A. Esmaeili、Jamie Morgenstern 和 Claire Jie Zhang。公平聚类：批评、注意事项与未来方向。arXiv 预印本 arXiv:2406.15960，2024 年。[DK14] Christina Delimitrou 和 Christos Kozyrakis。Quasar：资源高效且具备 QoS 感知能力的簇管理。载于《第19届编程语言与操作系统体系结构支持国际会议（ASPLOS）论文集》，页码 127–144，2014 年。[EFKR11] Yuval Emek、Pierre Fraigniaud、Amos Korman 和 Adi Ros´en。带建议的在线计算。《理论计算机科学》，412(24):2642–2656，2011 年。[FKL+91] Amos Fiat、Richard M. Karp、Michael Luby、Lyle A. McGeoch、Daniel D. Sleator 和 Neal E. Young。竞争性分页算法。《算法杂志》，12(4):685–699，1991 年。[FRR94] Amos Fiat、Yuval Rabani 和 Yiftach Ravid。竞争性 k-服务器算法。《计算机与系统科学杂志》，48(3):410–428，1994 年。[FSTW19] Till Fluschnik、Piotr Skowron、Mervin Triphaus 和 Kai Wilker。公平背包。载于《AAAI 人工智能会议（AAAI）论文集》，页码 1941–1948，2019 年。[GKL16] Sushmita Gupta、Shahin Kamali 和 Alejandro L´opez-Ortiz。稀疏度量下 k-服务器问题的建议复杂度。《计算系统理论》，59(3):476–499，2016 年。

[HHIS23] Hadi Hosseini、Zhiyi Huang、Ayumi Igarashi 和 Nisarg Shah。《在线匹配中的类别公平性》。《Artificial Intelligence》，2023。初步版本发表于 2023 年 AAAI 会议。[HJS+24] MohammadTaghi Hajiaghayi、Shayan Chashm Jahan、Mohammad Sharifi、Suho Shin 和 Max Springer。《在线类别匹配中的公平性与效率》。载于《第37届神经信息处理系统进展大会》（NeurIPS），2024。[JLM25] Sven J¨ager、Alexander Lindermayr 和 Nicole Megow。《多面体约束下非全知调度中比例公平性的力量》。载于第36届年度 ACM-SIAM离散算法研讨会（SODA）论文集，第3901–3930页，2025。[KMMO88] Anna R. Karlin、Mark S. Manasse、Lyle A. McGeoch 和 Susan Owicki。《竞争性窥探式缓存》。《Algorithmica》，3:77–119，1988。[KMWW20] Predrag Krnetic、Darya Melnyk、Yuyi Wang 和 Roger Wattenhofer。《均匀度量空间中带延迟的 k服务器问题》。载于第31届国际算法与计算研讨会（ISAAC）论文集，LIPIcs第181卷，第61:1–61:13页，2020。[Kou94] Elias Koutsoupias。《在线算法与 k服务器猜想》。博士论文，加州大学圣迭戈分校，1994年6月。[Kou09] Elias Koutsoupias。《 k服务器问题》。《理论计算机科学》，2009:105–118，2009。[KP95] Elias Koutsoupias 和 Christos H. Papadimitriou。《关于 k服务器猜想》。《Journal of the ACM》，42(5):971–983，1995。[KSDH23] Marina Knittel、Max Springer、John P. Dickerson 和 MohammadTaghi Hajiaghayi。《公平的多对数近似低成本层次聚类》。载于《神经信息处理系统进展》第36卷（NeurIPS），2023。[KT04] Elias Koutsoupias 和 David Scot Taylor。《CNN问题及其他 k服务器变体》。《理论计算机科学》，324(2–3):347–359，2004。[KVT12] Mario K¨oppen、Rodrigo Verschae 和 Masato Tsuru。《Masp：一种面向多目标优化、采用公平决策者的多属性秘书问题方法》。载于第6届遗传与进化计算国际会议（ICGEC 2012）论文集，第324–327页，2012。[LMS22] Alexander Lindermayr、Nicole Megow 和 Bertrand Simon。《结合机器学习建议的双重覆盖》。载于第13届理论计算机科学创新大会（ITCS）论文集，第215卷，第99:1–99:18页，2022。

[LSS+24] Adam Lechowicz、Rik Sengupta、Bo Sun、Shahin Kamali 和 Mohammad Hajiesmaili。《在线背包问题中的时间公平性》。载于第*12*届国际学习表征会议（*ICLR*）论文集，2024。[MMCF24] Antonio Martinez-Sykora、Fraser N. McLeod、Tom J. Cherrett 和 Adrian Friday。《探索食品配送路径规划与调度问题中的公平性》。《专家系统与应用》，240:122488，2024。[MMS90] Mark S. Manasse、Lyle A. McGeoch 和 Daniel D. Sleator。《服务器问题的竞争算法》。《算法杂志》，11(2):208–230，1990。[MV21] Yury Makarychev 和 Ali Vakilian。《社会公平聚类的近似算法》。载于第*34*届年度计算学习理论大会（*COLT*）论文集，第134卷《机器学习研究论文集》，2021。[NTV23] April Niu、Agnes Totschnig 和 Adrian Vetta。《公平算法设计：公平且有效的机器调度》。载于第*16*届算法博弈论国际研讨会（*SAGT 2023*）论文集，《计算机科学讲义》第14238卷，第239–256页，2023。[OPY+24] Yi Cheng Ong、Nicos Protopapas、Vahid Yazdanpanah、Enrico H. Gerding 和 Sebas- tian Stein。《公平高效的乘车调度：一种偏好驱动方法》。《模拟杂志》，18(6):940–956，2024。[PKL21] Deval Patel、Arindam Khan 和 Anand Louis。《背包问题中的群体公平性》。载于第*20*届自主智能体与多智能体系统国际会议（*AAMAS 2021*）论文集，第1001–1009页，2021。[RR15] Marc P. Renault 和 Adi Ros´en。《关于针对 k服务器问题的带建议在线算法》。《计算系统理论》，56(1):3–21，2015。[SKC23] Daman Deep Singh、Amit Kumar 和 Abhijnan Chakraborty。《迈向含 k 个服务器的在线服务公平性及其在公平食品配送中的应用》。载于<style id='42}>AAAI 人工智能会议论文集，第37卷，第5856–5864页，2023。[ST85] Daniel Dominic Sleator 和 Robert Endre Tarjan。《列表更新与分页规则的摊销效率》。《美国计算机学会通讯》，28(2):202–208，1985。[wNM25] 《带噪声成员关系的稳健公平聚类》。学习增强的度量任务系统的最优稳健性—一致性权衡。载于第*28*届人工智能与统计国际会议（*AISTATS*）论文集，2025。

超越 (α, β)-公平性：替代公平性概念

我们的 (α, β)-公平性定义体现了一种自然的比例条件：在允许少量松弛量的前提下，没有任何服务器支付的成本会显著超过其公平份额。不过，形式化公平性的合理方式还有很多。本节将介绍并比较三种替代公平性概念——加法公平性、乘法公平性和可接受公平性，其中一些概念曾隐含地出现在特定领域的背景下 [MMS90]。

我们将探讨这些概念之间的关系，以及满足一种概念的算法能否转换为满足另一种概念的算法。首先介绍基本定义。为简单起见，本节始终考虑确定性算法。尽管所有定义也可以类似地应用于随机算法，正如我们的 (α, β)-公平性概念一样。

定义 2（加法公平性）。 若算法 A 满足以下条件，则称其对 $\beta \geq 0$ 具有 β-加性公平：对于任意输入序列 σ，均有：

$$\max_{i,j \in [k]} |c_i(A, \sigma) - c_j(A, \sigma)| \leq \beta.$$

定义 3（乘法公平性）。 如果对于任意输入序列 σ，算法 A 都满足：则称其对$\alpha \geq 1$ 是 α-乘法公平的 。

$$\max_{i,j \in [k]} \frac{c_i(A, \sigma)}{c_j(A, \sigma)} \leq \alpha.$$

定义 4（可接受公平性）。 如果对于任意输入序列 σ，算法 A 都满足：则称其对 $\gamma \geq 1$ 是 γ-可接受公平的 。

$$\max_{i \in [k]} c_i \leq \gamma \cdot \textsc{Opt}(\sigma),$$

其中 $\textsc{Opt}(\sigma)$是输入 σ的离线最优成本。

现在我们来探讨不同公平性概念之间的关系。具体而言，我们将证明，在对竞争性作出温和假设的条件下，某些公平性定义可以推出其他定义。这些归约有助于统一各种公平性保证，并表明，设计具有某一公平性目标的算法，可能会因此自然地满足其他目标。

需要指出的是，这些概念与（α, β）-公平性密切相关，但并不等价。具体而言，任何 β-加性公平算法在我们的（α, β）-公平性定义下，也都是 $(1, \Theta(\beta))$-公平的，因为有界的两两差异意味着，没有任何服务器的成本与平均成本相差超过一个加性的 $\Theta(\beta)$ 项。然而，反过来则不成立：算法可能是 $(1, \beta)$-公平的，却不满足加法公平性，因为后者要求所有成本两两差异都有统一的界。

类似地，任何 α-乘法公平算法都是 $(\Theta(\alpha),0)$-公平的，因为服务器成本之间的比率有界，就意味着相对于总成本也存在成比例的界限。最后， γ-可接受公平性直接对应于相对于 $\textsc{Opt}(\sigma)$ 的 $(\gamma, 0)$-公平性，因为它将任一服务器产生的最大成本限制在离线最优值的常数倍以内。

A.1 竞争性算法的乘法公平性与可接受公平性的等价性

我们首先证明，任何具有乘法公平性且为 $O(k)$-竞争的算法，也是可接受公平的。

定理 16. 任何满足 $O(1)$-乘法公平性且为 $O(k)$-竞争的在线算法，都是 $O(1)$-可接受公平的。

证明。 根据 α-乘法公平性，对于所有 $i, j \in [k]$，我们有 $c_j \leq \alpha \cdot c_i$，因此：

$$\sum_{i=1}^{k} c_i \geq k \cdot \frac{1}{\alpha} \cdot c_j.$$

根据 $O(k)$-竞争性，$\sum_{i=1}^{k} c_i \leq O(k \cdot \mathrm{OPT})$；因此，利用 $\alpha = O(1)$，对于任意服务器 j：

$$c_j \leq \frac{O(k \cdot OPT)}{k \cdot \frac{1}{\alpha}} = O(\mathrm{OPT}).$$

□

接下来，我们证明，任何可接受公平的算法都可以转换为乘法公平算法，且竞争性只会增加一个常数因子。结合定理 16，这表明对于 $O(k)$-竞争算法，乘法公平算法与可接受公平算法的存在性在本质上是等价的。

算法 3 实现了一个简单的负载均衡过程，可将可接受公平性转换为乘法公平性。其总体思路如下：在每个时间步，我们将每台服务器的累积成本与当前最大累积成本 h_t进行比较。如果某台服务器的负载“过轻”（也就是说，其累积成本远低于 h_t），算法就会强制其执行一次长度为 Θ(diam) 的纠正移动，从而使其成本接近最大值。已经达到最大成本的服务器则保持不变。由于每次纠正移动的长度至多为 2diam，当某台服务器落后时，只需额外移动 O(diam) 的成本，就能使其回到距 h_t 受控的加性距离之内。接下来，我们将给出一个定理，利用算法 3 说明如何将这一转换应用于可接受公平算法，从而得到乘法公平算法。

定理 17. 任何满足 $O(1)$-可接受公平性的在线算法（因此也具有 $O(k)$-竞争性），都可以转化为一个 $O(1)$-乘法公平算法，其竞争比为 $O(k)$。

证明。 定义 $cc_i(t, \mathrm{ALG}) = \sum_{j=1}^{t} c_i(j, \mathrm{ALG})$ 为算法 ALG 在处理请求 t之前，服务器 i 的累积成本。

首先，为了证明第 9 行的合理性，注意在直径为 diam 的任意度量空间中，对于任意点 x，都存在一点 y，使得 $d(x, y) \geq \mathrm{diam}/2$。否则，所有点都将位于一个半径为

算法 3： 将可接受公平算法转换为乘法公平算法

输入： k-服务器算法 A，每个时间步移动每台服务器的距离至多为直径。

1 定义 $cc_i(t,\ \text{ALG}) = \sum_{j=1}^{t} c_i(j,\ \text{ALG})$ 为算法 ALG 中服务器 i 的累积成本截至第 t 步。
2 $t \leftarrow 1$;
3 **对每个**请求 σ_t **执行**
4 模拟 $A(\sigma_t)$; // 在算法 A 上运行 σ_t
5 根据算法 A 的模拟移动所有服务器;
6 $h_t \leftarrow$ 最大值$_i$ $cc_i(t,\ A)$;
7 **对于** $i = 1$ **到** k **执行**
8 **如果** cc_i（t，此）$< h_t$ **则**
9 选取 x，使得 $d(s_i(t)\ x) \geq$ 直径$/2$;
10 将服务器 i 移动到 x，再移回 $s_i(t)$;
11 $t \leftarrow t + 1$;
12 **对于** $i = 1$ **到** k **执行**
13 **如果** cc_i（t，此）$= h_t$ **则**
14 选择 x 使得 $d(s_i(t)\ x) \geq$ 直径$/2$;
15 将服务器 i 移至 x，然后移回 $s_i(t)$;

小于 diam$/2$ 的球内，这意味着任意两点之间的最大距离都严格小于 diam，这与直径的定义矛盾。因此，如果一台服务器从当前位置移动到这样一点再返回，它行进的距离至少为 diam，至多为 2diam。

其次，我们证明，在每个时间步结束时（第 10 行），所有服务器支付的成本至少为 $h_t = \max_j cc_j(t,\ A)$。

引理1。 对于所有服务器 i 以及所有时间步 t，我们有

$$cc_i(t, this)\ \geq\ h_t\ =\ \max_j cc_j(t, A)$$

在算法 *3* 的第 *10* 行之后。

证明。 我们通过归纳法证明，在每个时间步结束后 t（第 10 行），都有 $cc_i(t,\ \text{this}) \geq h_t$对所有 i 成立。对于基本情况 $t = 0$，我们有 $cc_i(0,\ \text{this}) = h_0 = 0$，因此该陈述成立。

现在考虑任意一步 $t > 0$。根据算法 A 的定义（见算法 3），A 的模拟执行中的最大累积成本在相邻两步之间至多增加 diam；也就是说，

$$h_t \leq h_{t-1} + \text{diam}.$$

如果在第 8 行，我们有 $cc_i(t-1,\ \text{this}) < h_t$，那么该算法会执行一次纠正移动，使服务器 i 至少增加 diam。结合归纳假设以及对

h_t 增量的界，我们得到

$$cc_i(t, \text{this}) \;\geq\; cc_i(t-1, \text{this}) + \text{diam} \;\geq\; h_{t-1} + \text{diam} \;\geq\; h_t.$$

反之，如果 $cc_i(t-1, \text{此处}) \geq h_t$，则不会执行纠正移动，因此

$$cc_i(t, \text{this}) = cc_i(t-1, \text{this}) \geq h_t.$$

因此，在这两种情况下，不等式 $cc_i(t, \text{此处}) \geq h_t$ 在第 t 步结束时均成立，从而完成归纳法。

□

接下来，我们完成对每台服务器所付成本的界定，这将使我们能够证明算法 3 是乘法公平的。

引理 2。 算法 *3* 中任意服务器所付的总成本都位于区间

$$\big[\max(\max_j cc_j(A),\, \text{diam}),\, 2 \cdot \max_i c_i + 2 \cdot \text{diam}\big].$$

证明。 固定一台服务器 i。在每个时间步，服务器 i 可能因模拟算法 A 而移动，也可能因为其累积成本 $cc_i(t, \text{此处})$ 小于当前最大值 h_t 而移动。令 t^* 为满足 $cc_i(t^*, \text{此处}) < h_{t^*}$ 的最后一个步骤（算法 3 的第 8 行），也就是说，服务器会执行一次纠正移动（注意，t^* 可能为 0）。

在时间 t^* 之后，服务器 i 仅按照算法 A 移动，因此从那时起其成本至多为 $cc_i(A)$。在时间 t^*，服务器 i 截至第 t^* 步的成本严格小于 h_{t^*}；又因为 $h_{t^*} \leq \max_j cc_j(T, A)$，这给出了纠正移动之前成本的上界。每次纠正移动（见第 9 行）也至多增加 2diam，因此，服务器 i 的总移动量至多为 $\max_j cc_j(A) + 2\text{diam}$。

值得一提的是，算法 3 的最后四行仅移动成本满足以下条件的服务器

$$cc_i(T, \text{this}) = h_T = \max_j cc_j(A),$$

这些移动会迫使它们的最终成本处于以下范围内

$$[\max_j cc_j(A) + \text{diam},\; \max_j cc_j(A) + 2\text{diam}].$$

这确保了我们的上界始终对所有服务器成立，而对于这些服务器，我们的下界也同样成立。

对于所有其他服务器，根据引理1，它们至少支付 $\max_j cc_j(A)$；任何支付更多（而非相等）的服务器都必定在第 9 行被移动过，这至少会额外增加一个直径。因此，由此事实和引理1可知，每台服务器至少支付

$$\max(\text{diam},\; \max_j cc_j(A)),$$

且至多

$$2\max_j cc_j(A) + 2\text{diam}.$$

□

引理 2 保证，将 A 通过算法 3 进行转换后，每台服务器产生的成本至少为 max(max$_j$ c_j, diam)，至多为 $2 \cdot \max_j c_j + 2 \cdot$diam，从而确保所得算法是乘法公平的，且比率至多为 $4 \leq O(1)$。

N此外，由于该算法是 $O(1)$-可接受公平的，计算总成本可得 es

$$\sum_{i=1}^{k} c_i \;\leq\; k \cdot 2 \cdot \max_j c_j \;+\; 2k \cdot \text{diam} \;\leq\; O(k) \cdot \textsc{Opt},$$

这表明变换后的算法也具有 $O(k)$-竞争性。

□

A.2 与加法公平性的联系

虽然我们已经证明，在合理的竞争性假设下，可接受公平性与乘法公平性密切相关，但在在线设置中，加法公平性似乎从根本上更难保证。具体而言，加法公平性要求严格控制服务器之间的成本偏差——如果不知道输入序列何时结束，这可能无法实现。人们或许会认为，可以利用算法 3 将可接受公平算法转换为加性公平算法。然而，算法 3 只能使各服务器的成本保持在彼此相差一个常数乘法因子的范围内（事实上，这个因子可以改进到 2）；而要实现加法公平性，则要求每台服务器在任何时刻与当前成本最高的服务器之间的距离都不超过 diam。这在在线设置中会造成问题：考虑一个对抗序列，其中服务器 1、服务器 2、服务器 3 等依次成为唯一成本最高的服务器。每当成本最高的服务器发生变化，加法公平性转换就必须迫使其余 $k-1$ 台服务器通过额外移动 Θ(diam) 的距离来“追赶”，因此每个阶段的成本都会比原算法 A 多出约 ($k-1$)diam。若这一模式在多个阶段中反复出现，则每个阶段的总开销将达到 $k$diam 的量级，从而使竞争比大约增加 k 倍。因此，任何试图始终维持严格 diam- 加法公平性的策略，似乎都无法同时保留原有的 $O(k)$-竞争保证。

当算法知道请求序列何时结束时，加法公平性便可以实现；然而，这一保证较弱，因为算法仅在最终 步骤达到加性公平，而非在整个执行过程中始终如此。尽管如此，在这一额外假设下，我们仍能从一个公平性尚可的算法构造出一个加性公平算法，如下所示。

定理 18。 如果算法预先知道最后一个请求，那么任何 $O(1)$-可接受公平算法都可以转换为一个具有 O(直径)加性公平性的算法，其竞争比为 $O(k)$。

证明。 设 A 为任意可接受公平算法，并令 c_i 表示服务器 i 在 A下的最终成本，其中 $c_{\max} :=$ 最大值$_i\, c_i$。 由于 A 是可接受公平的，因此 $c_{\max} = O(\mathrm{OPT})$。

在模拟 A 后，对于每个满足 $c_i < c_{\max}$ –diam 的服务器 i，我们通过执行一次或多次额外移动，将其成本增加到距离处于 $[\mathrm{diam}/2,\ \mathrm{diam}]$ 范围内的点上（该点确实存在；参见定理 17 证明的第一段）。每次这样的移动都会使 c_i 至少增加 $\mathrm{diam}/2$，至多增加 diam。因此，每台服务器的最终成本都落在区间 $[c_{\max} - \mathrm{diam}\ c_{\max}]$ 内，因为步长至多为 diam 时，我们无法超出此范围。由此可确保加法公平性，且松弛量为 diam。

最后，由于 $c_{\max} = O(\mathrm{OPT})$，且所有服务器与其之间的距离都不超过 diam，因此总成本至多为

$$\sum_{i=1}^{k} c_i \le k \cdot c_{\max} = O(k \cdot \mathrm{OPT}),$$

从而保持 $O(k)$-竞争性。 □

这引出了以下猜想，它揭示了可接受公平性与加法公平性之间可能存在的差异。

猜想 1. 在不知道最后一个请求的情况下，不可能将每个$O(1)$-可接受公平的在线算法转换为竞争比为 $O(k)$的 $O(\mathrm{diam})$-加法公平算法。

最后，我们证明可以实现加法公平性，但代价是竞争比显著恶化。具体而言，以下构造应仅视为一个基准：所得的 $c \cdot k$ 竞争比通常相当弱，远逊于人们猜测通过从可接受公平性或乘法公平性进行转换所能达到的水平。

定理 19. 设 A 是针对 k-服务器问题的一个 c-竞争在线算法。那么， A 可以转换为一个具有 直径 加法公平性的算法，其竞争比为 $c \cdot k$。

证明。 假设我们采用与定理 18 类似的思路。运行原始算法 A；每当某台服务器的成本比最高服务器成本低 2diam 以上时，就执行任意多次校正移动，使其成本与最高服务器成本之差不超过 2diam。此操作会使成本增加介于 diam 和 2diam 之间，因此服务器不会因此成为成本最高的服务器（因为一旦其进入与最高成本相差 2diam 的范围，我们就会停止）。由此可实现 2diam 的加法公平性。在变换后的算法中，对于每台服务器 i，我们有：

$$cc_i(t, \mathrm{this}) \ge \max_j cc_j(t, \mathrm{this}) - 2\mathrm{diam}, \quad \text{and} \quad cc_i(t, \mathrm{this}) \le \max_j cc_j(t, \mathrm{this}),$$

确保加法公平性。

现在我们通过归纳法证明，在每个时间步 t，变换后的算法中的最大成本都不超过算法 A 截至时间 t的总成本，即，

$$\max_j cc_j(t, \text{this}) \leq \text{cost}(A, t).$$

基本情况（$t = 0$）： 最大成本和成本(A，0)均为零。

归纳步骤： 假设该结论对时间 $t-1$ 成立。令 $c_i(t)$ 表示服务器 i在 A 时间 t 产生的成本。我们知道，在我们的算法中，最大值只会因服务器依据 A进行移动而发生变化，而校正步骤不会改变最大值。因此，最大成本将为：

$$m = \arg\max_j cc_j(t, \text{this}), \qquad cc_m(t, \text{this}) = cc_m(t-1, \text{this}) + c_m(t). \tag{4}$$

现在，根据归纳假设，我们可以说：

$$cc_m(t-1, \text{this}) \leq \max_j cc_j(t-1, \text{this}) \leq \text{cost}(A, t-1), \tag{5}$$

从而得到

$$cc_m(t, \text{this}) = cc_m(t-1, \text{this}) + c_m(t) \leq \text{cost}(A, t-1) + c_m(t) \leq \text{cost}(A, t). \tag{6}$$

这就证明了归纳结论。

对于最终时间步 t，可得：

$$\max_j c_j(\text{this}) \leq \text{cost}(A).$$

因此，变换后的算法的总成本为：

$$\sum_{i=1}^{k} c_i(\text{this}) \leq k \cdot \max_j c_j(\text{this}) \leq k \cdot \text{cost}(A).$$

由于 A 是 c-竞争性的，总成本至多为 $c \cdot \text{OPT}$，因此：

$$\text{cost}(\text{this}) \leq c \cdot k \cdot \text{OPT}.$$

证明至此完成。

□

最后，我们注意到，对于 $k = O(1)$，可以证明存在具有 1-竞争性且加性公平的确定性算法。

定理 20. 对于任意常数 k，都存在一种用于 k-服务器问题的确定性算法，该算法具有 $O(1)$-竞争性，并且具有 直径-加性公平性。

证明. 将定理 19 应用于任意 $O(k)$-竞争算法（例如，树上的 DCA），可得到一种 $k \cdot O(k)$-竞争算法，该算法同时也是加性公平的。由于 k 是常数，竞争比仍为 $O(1)$。

□

B 平等主义成本模型

对于 k-服务器问题的平等主义成本模型，算法的成本定义为任意单个服务器行进的最大距离，而非所有服务器行进距离之和。对于请求序列 σ，令 $\mathrm{OPT}_e(\sigma)$ 表示任意离线解所能达到的最小平等主义成本。

考虑到最优平等主义解中所有服务器的总成本小于最大服务器成本的 k 倍，因此我们可以写出

$$\mathrm{OPT}_e(\sigma) \times k \;\geq\; cost(\mathrm{OPT}_e(\sigma), \sigma) \;\geq\; \mathrm{OPT}(\sigma).$$

因此，最优平等主义成本至少为常规模型中最优服务器成本的平均值，

$$\mathrm{OPT}_e(\sigma) \;\geq\; \frac{\mathrm{OPT}(\sigma)}{k},$$

其中，$\mathrm{OPT}(\sigma)$ 表示最优功利主义（总）成本。

现在假设算法 A 相对于功利主义成本是 $(\alpha,\ \beta)$-公平的，这意味着对于每个服务器 i，

$$c_i(A, \sigma) \;\leq\; \frac{\alpha}{k} \mathrm{cost}(A, \sigma) + \beta.$$

那么， A的平等主义成本满足

$$A_e(\sigma) \;=\; \max_{i \in [k]} c_i(A, \sigma) \;\leq\; \frac{\alpha}{k} \mathrm{cost}(A, \sigma) + \beta.$$

此外，如果在功利主义目标下， A 具有 c-竞争性，即对于所有 σ，均有 $\mathrm{cost}(A, \sigma) \leq c \cdot \mathrm{OPT}(\sigma)$，那么

$$A_e(\sigma) \;\leq\; \alpha c \cdot \mathrm{OPT}_e(\sigma) + \beta,$$

其中，不等式由 $\mathrm{OPT}_e(\sigma) \geq \mathrm{OPT}(\sigma)/k$ 得出。因此，在功利主义成本下的 (α, β)-公平性，直接意味着在平等主义成本模型中具有 αc 竞争保证，且乘法因子和加性松弛量相同。

C 省略的证明

现在我们给出完整的 p 本文主体中省略的、用于证明主要结果的部分。

C.1 定理 8 的证明

定理 21（定理 8 的重述）。考虑任意一个包含 k 台服务器、长度为 T 的输入序列 σ，该序列取自直径为 直径 的度量空间。令 $\{s_i(t)\}_{i\in[k],\ t\in[T]}$ 为在该输入序列上达到最优成本的服务器位置序列。

对于任意 $\varepsilon > 0$，算法 1 以 $\{s_i(t)\}_{i\in[k],t\in[T]}$ 作为输入，并返回修改后的位置序列 $\{\hat{s}_i(t)\}_{i\in[k],t\in[T]}$；该序列相对于 $\mathrm{OPT}(\sigma)$ 是 $(1+\varepsilon,\beta)$-公平的，其中 $\beta = O\left(\log k \cdot \mathrm{D_i}_{\varepsilon}\right)$。此外，所得解的总成本有界于 $\mathrm{OPT}(\sigma) + O\left(k \log k \cdot \mathrm{D_i}_{\varepsilon}\right)$.

证明。我们用术语**交换**来指从指定时间步开始交换两台服务器未来工作计划的行为。具体而言，如果服务器 H_r 和 L_r 在时间 z交换，则每台服务器从时间 z 起接管另一台服务器剩余的分配任务（第 11 行）。

交换会使每台参与交换的服务器的成本最多增加其交换时位置之间的距离。这体现在时间 $t = z$ 的转移步骤中，每台服务器都会产生一项额外惩罚，数值等于 $d(s_{H_r}^{(r)}(z),\ s_{L_r}^{(r)}(z))$。由于任意两点之间的距离至多为直径，每次交换导致的总成本增幅至多为 $2 \cdot \mathrm{D_i}$。

令 r 表示算法目前为止执行的交换次数。定义 $\hat{c}_x^{(r)}$为服务器 x 在经过 r 次交换后产生的总成本（也包括交换成本）。令 H_r，$L_r \in [k]$ 分别表示第 r 轮交换之前总成本最高和最低的服务器（见算法 1 第 5 行）

我们用 $\hat{c}_{H_r}^{(r)}$ 表示服务器 H_r 在第 r 轮产生的总成本，并用 $\hat{c}_{L_r}^{(r)}$ 表示服务器 L_r的成本。我们的分析着重于界定 $\hat{c}_{H_r}^{(r)} - \hat{c}_{L_r}^{(r)}$ 之差，并证明该差值在每一轮中都会以常数因子递减。

为了形式化描述总成本最高和最低的服务器之间的成本均衡过程，我们着重研究如何比较两个成本值序列。每台服务器未来的工作计划都可以视为一个由每个请求的成本构成的序列。当我们考虑交换两台服务器的剩余工作量时，希望找到一个分割点 z，使得交换它们在 z 之后的后缀后，两者的总成本能够达到均衡，且加性松弛较小。以下引理保证了这样一个分割点的存在。

引理 3. 设 $A = (A_1, A_2, \ldots, A_n)$ 和 $B = (B_1, B_2, \ldots, B_n)$ 是两个实数序列，其中 $A_i, B_i \in [0, c]$ 对所有 $i \in [n]$均成立。那么存在一个索引 $z \in [n]$ 使得：

$$\left|\left(\sum_{i=1}^{z} A_i + \sum_{i=z+1}^{n} B_i\right) - \left(\sum_{i=1}^{z} B_i + \sum_{i=z+1}^{n} A_i\right)\right| \le c. \tag{7}$$

证明。定义一个序列 D_j，其中 $j = 0$ 到 n 为：

$$D_j = \sum_{i=1}^{j} A_i + \sum_{i=j+1}^{n} B_i - \sum_{i=1}^{j} B_i - \sum_{i=j+1}^{n} A_i.$$

根据 D 的定义，只需证明存在 z 使得 $|D_z| \le c$。现在注意到：

$$D_0 = \sum_{i=1}^{n} B_i - \sum_{i=1}^{n} A_i, \quad D_n = \sum_{i=1}^{n} A_i - \sum_{i=1}^{n} B_i = -D_0.$$

因此， D_0 和 D_n 大小相等但符号相反。现在我们证明，存在一个索引 $x \in \{0, \ldots, n-1\}$，使得 D_x和 D_{x+1}符号相反（或其中一个为零）。反设不存在这样的索引——也就是说，对于所有 $j = 0,\ldots, n-1$， D_j和 D_{j+1}符号相同，且都不为零。

在这种情况下，整个序列（$D_0, D_1, \ldots, D_n$）的符号始终相同。但这与 D_0 和 D_n 符号相反这一事实矛盾。因此，这样的索引 x 必定存在。现在考察相邻两项之间的变化：

$$\begin{aligned}
D_x - D_{x-1} &= \sum_{i=1}^{x} A_i + \sum_{i=x+1}^{n} B_i - \sum_{i=1}^{x} B_i - \sum_{i=x+1}^{n} A_i - \sum_{i=1}^{x-1} A_i - \sum_{i=x}^{n} B_i + \sum_{i=1}^{x-1} B_i + \sum_{i=x}^{n} A_i \\
D_x - D_{x-1} &= A_x - B_x - B_x + A_x = 2(A_x - B_x), \\
\Rightarrow |D_x - D_{x-1}| &= 2|A_x - B_x| \le 2c.
\end{aligned}$$

因为 $A_x, B_x \in [0, c]$。

现在我们说明， x 或 $x+1$ 满足该引理。分两种情况：

- 如果 $|D_x| \le c$，那么 x 是 z的一个有效取值。
- 否则，由于 D_x 和 D_{x+1} 符号相反，且二者相差至多 $2c$ 和 $|D_x| > c$，必有 $|D_{x+1}| < c$，因此 $x+1$ 是一个有效的选择。

因此，存在某个 $z \in \{x, x+1\}$ 使得：

$$\left| \sum_{i=1}^{z} A_i + \sum_{i=z+1}^{n} B_i - \left(\sum_{i=1}^{z} B_i + \sum_{i=z+1}^{n} A_i \right) \right| \le c,$$

如所述。 □

对于 H_r 和 L_r之间的每次交换，考虑它们在每一步 t的每个请求的成本，记为$\{\hat{c}_{H_r}^{(r)}(t)\}$。该序列的长度为 n，且每一项至多为直径。应用引理 3，我们可以找到一个索引 z，使得在交换这两个序列的后缀时， $|D_z| \le$直径。

为继续证明，我们先确立三个关键论断。

断言 1. 在算法 *1* 中，经过 $\beta = 2 \cdot 1 + \varepsilon \cdot diam \cdot (\frac{3}{2} + \underbrace{\text{对数}_{2+2\varepsilon}\, k}_{2+\varepsilon})$后，在第 $r \le k \cdot \text{对数}_{2+2\varepsilon}\, k$轮交换服务器 H_{r-1}和 L_{r-1}后，两台服务器的总成本都降至交换前服务器最大成本的至多 $\binom{2+\varepsilon}{2+2\varepsilon}^{2+\varepsilon}$

这一比例。也就是说，

$$\hat{c}^{(r)}_{H_{r-1}} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)\cdot \hat{c}^{(r-1)}_{H_{r-1}},$$
$$\hat{c}^{(r)}_{L_{r-1}} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)\cdot \hat{c}^{(r-1)}_{H_{r-1}}.$$

每次交换后，参与交换的两台服务器会互换各自剩余的未来职责，使它们的总成本在加性惩罚 diam 的范围内达到平衡。利用引理 3，我们证明，这次交换会使每台服务器的成本接近交换前两者成本的平均值。通过使用交换阈值 β 和最大轮次数来界定加性项，我们证明两台服务器的成本都会缩小一个常数乘法因子。

证明。 在第 r轮进行交换后，服务器 H_{r-1} 被重新分配去执行原本分配给 L_{r-1}的剩余任务，反之亦然。

令 A_i 表示服务器 H_{r-1} 在时间 i产生的成本，并令 B_i 表示服务器 L_{r-1} 在时间 i产生的成本。我们有：

$$\hat{c}^{(r-1)}_{H_{r-1}} = \sum_{i=1}^{T} A_i, \quad \text{and} \quad \hat{c}^{(r-1)}_{L_{r-1}} = \sum_{i=1}^{T} B_i.$$

假设我们在引理 3 保证存在的索引 $z \in [T]$ 处进行交换。那么，服务器 H_{r-1} 在第 r 轮的新成本为：

$$\hat{c}^{(r)}_{H_{r-1}} \le \sum_{i=1}^{z-1} A_i + \mathrm{diam} + \sum_{i=z}^{T} B_i, \tag{8}$$

服务器 L_{r-1} 的新成本为：

$$\hat{c}^{(r)}_{L_{r-1}} \le \sum_{i=1}^{z-1} B_i + \mathrm{diam} + \sum_{i=z}^{T} A_i. \tag{9}$$

现在我们要证明，交换后，每台服务器产生的成本都接近其原始成本的平均值，至多相差一个加性惩罚。令：

$$F = \frac{1}{2}\left(\sum_{i=1}^{z-1} A_i + \sum_{i=z}^{T} B_i + \sum_{i=1}^{z-1} B_i + \sum_{i=z}^{T} A_i\right),$$
$$G = \frac{1}{2}\left(\sum_{i=1}^{z-1} A_i + \sum_{i=z}^{T} B_i - \sum_{i=1}^{z-1} B_i - \sum_{i=z}^{T} A_i\right).$$

然后，应用三角不等式来界定 $|F+G| \le |F| + |G|$，可得：

$$\sum_{i=1}^{z-1} A_i + \sum_{i=z}^{T} B_i = F + G \le \frac{1}{2}\left(\sum_{i=1}^{T} A_i + \sum_{i=1}^{T} B_i\right) + \frac{1}{2}\left|\sum_{i=1}^{z-1} A_i + \sum_{i=z}^{T} B_i - \sum_{i=1}^{z-1} B_i - \sum_{i=z}^{T} A_i\right|.$$

根据引理 3，绝对值项至多为直径。因此：

$$\sum_{i=1}^{z-1} A_i + \sum_{i=z}^{T} B_i \leq \frac{1}{2}\left(\sum_{i=1}^{T} A_i + \sum_{i=1}^{T} B_i\right) + \frac{\text{diam}}{2}.$$

由于 $\hat{c}_{H_{r-1}}^{(r-1)} = \sum_{i=1}^{T} A_i$ 和 $\hat{c}_{L_{r-1}}^{(r-1)} = \sum_{i=1}^{T} B_i$，我们得到：

$$\sum_{i=1}^{z-1} A_i + \sum_{i=z}^{T} B_i \leq \frac{1}{2}\hat{c}_{H_{r-1}}^{(r-1)} + \frac{1}{2}\hat{c}_{L_{r-1}}^{(r-1)} + \frac{\text{diam}}{2}.$$

交换后服务器 L_{r-1} 的成本也可通过对称论证得出。具体而言，使用相同的三角不等式推理，但以 $-G$ 替代 G，我们得到：

$$\sum_{i=1}^{z-1} B_i + \sum_{i=z}^{T} A_i \leq \frac{1}{2}\hat{c}_{H_{r-1}}^{(r-1)} + \frac{1}{2}\hat{c}_{L_{r-1}}^{(r-1)} + \frac{\text{diam}}{2}. \tag{10}$$

结合公式（10）与公式（8）和（9），我们得出：

$$\begin{aligned} \hat{c}_{H_{r-1}}^{(r)} &\leq \frac{3}{2} \cdot \text{diam} + \frac{1}{2}\hat{c}_{H_{r-1}}^{(r-1)} + \frac{1}{2}\hat{c}_{L_{r-1}}^{(r-1)}, \\ \hat{c}_{L_{r-1}}^{(r)} &\leq \frac{3}{2} \cdot \text{diam} + \frac{1}{2}\hat{c}_{H_{r-1}}^{(r-1)} + \frac{1}{2}\hat{c}_{L_{r-1}}^{(r-1)} \end{aligned} \tag{11}$$

令 $\textsc{Opt}$ 表示初始（最优）离线解的总成本。由于算法 1 中的每次交换给总成本增加的幅度至多为 $2 \cdot \text{diam}$（每台服务器产生一次转换惩罚），第 r 轮开始时的总成本至多为：

$$\text{TotalCost}^{(r-1)} \leq \textsc{Opt} + 2(r-1) \cdot \text{diam}.$$

因此，第 r 轮之前每台服务器的平均成本为：

$$\frac{1}{k} \cdot \left(\textsc{Opt} + 2(r-1) \cdot \text{diam}\right).$$

由于在第 r 轮开始时，L_{r-1} 是成本**最低**的服务器，因此必有：

$$\hat{c}_{L_{r-1}}^{(r-1)} \leq \frac{1}{k}\left(\textsc{Opt} + 2(r-1) \cdot \text{diam}\right). \tag{12}$$

由于算法在第 r 轮执行了交换，因此算法 1 中的公平性条件必然遭到违反。具体而言，具有最大成本的服务器 H_{r-1} 必须满足：

$$\hat{c}_{H_{r-1}}^{(r-1)} > \frac{(1+\varepsilon) \cdot \textsc{Opt}}{k} + \beta. \tag{13}$$

对 $\hat{c}_{L_{r-1}}^{(r-1)}$ 使用不等式 (12)，对 $\hat{c}_{H_{r-1}}^{(r-1)}$ 使用不等式 (13)，将两者结合可得：

$$\hat{c}_{L_{r-1}}^{(r-1)} \leq \frac{1}{k}\left(2(r-1)\cdot \mathrm{diam}\right) + \frac{1}{1+\varepsilon}\cdot \hat{c}_{H_{r-1}}^{(r-1)} - \frac{\beta}{1+\varepsilon}. \tag{14}$$

将公式 (14) 代入公式 (11)，得到：

$$\begin{aligned}\hat{c}_{H_{r-1}}^{(r)} &\leq \frac{3}{2}\cdot \mathrm{diam} + \frac{1}{2}\hat{c}_{H_{r-1}}^{(r-1)} + \frac{1}{2}\left(\frac{\hat{c}_{H_{r-1}}^{(r-1)}}{1+\varepsilon} - \frac{\beta}{1+\varepsilon} + \frac{2(r-1)\cdot \mathrm{diam}}{k}\right) \\ &= \left(\frac{1}{2} + \frac{1}{2(1+\varepsilon)}\right)\hat{c}_{H_{r-1}}^{(r-1)} - \frac{\beta}{2(1+\varepsilon)} + \frac{3}{2}\cdot \mathrm{diam} + \frac{(r-1)\cdot \mathrm{diam}}{k}.\end{aligned} \tag{15}$$

为完成证明，我们将所选的 β 值代入 (15)。由于算法最多执行 $r \leq k\cdot \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k$ 轮次，只需令以下取值，即可完全吸收 (15) 中的加法项：

$$\begin{aligned}\frac{\beta}{2+2\varepsilon} &= \frac{\left(2\cdot(1+\varepsilon)\cdot \mathrm{diam}\cdot\left(\frac{3}{2} + \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k\right)\right)}{2(1+\varepsilon)} = \frac{3}{2}\cdot \mathrm{diam} + \mathrm{diam}\cdot \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k, \\ &\geq \frac{3}{2}\cdot \mathrm{diam} + \mathrm{diam}\cdot \frac{r}{k} \geq \frac{3}{2}\cdot \mathrm{diam} + \frac{(r-1)\cdot \mathrm{diam}}{k}.\end{aligned}$$

这样，不等式 (15) 就化为纯乘法界：

$$\hat{c}_{H_{r-1}}^{(r)} \leq \left(\frac{1}{2} + \frac{1}{2(1+\varepsilon)}\right)\hat{c}_{H_{r-1}}^{(r-1)} = \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)\hat{c}_{H_{r-1}}^{(r-1)}.$$

对 $\hat{c}_{L_{r-1}}^{r}$ 进行对称论证即可，命题得证。

□

下一个命题表明，所有服务器中的最大成本会随着轮次推进而递减，且只有发生交换的服务器可能改变其成本。

命题 2. 假设 $r \leq k\cdot \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k$。则：

$$\hat{c}_{H_r}^{(r)} \leq \hat{c}_{H_{r-1}}^{(r-1)}, \tag{16}$$

$$\textit{and for all } x \notin \{H_{r-1}, L_{r-1}\}, \quad \hat{c}_x^{(r)} = \hat{c}_x^{(r-1)}. \tag{17}$$

证明。 根据断言 1， H_{r-1} 和 L_{r-1} 的成本都至少降低了 $\left(\frac{2+\varepsilon}{2+2\varepsilon}\right) < 1$这一乘法因子。由于在交换之前，$H_{r-1}$的成本最大，因此这意味着第 r 轮之后的新最大成本必定小于或等于之前的最大成本：

$$\hat{c}_{H_r}^{(r)} \leq \hat{c}_{H_{r-1}}^{(r-1)}.$$

此外，其他所有服务器 $x \notin \{H_{r-1}, L_{r-1}\}$ 均不受交换影响，在算法中，它们的成本分配保持不变（如交换前算法 1 第 8 行中的成本复制步骤所示）。

因此，算法 1 第 8 行中的交换）。因此，

$$\hat{c}_x^{(r)} = \hat{c}_x^{(r-1)} \quad \text{for all } x \notin \{H_{r-1}, L_{r-1}\}.$$

□

以下声明表明，在算法 1 的任意连续（$k+1$）轮次内，最大服务器成本至少会缩小一个乘法因子 $\left(\frac{2+\varepsilon}{2+2\varepsilon}\right)$。这构成了该算法几何收敛的基础。

声明 3. 设 r 是算法 *1* 中任意一轮，满足 $r \le k \cdot \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k - k$，并假设算法在 r之后又运行了 k 轮。则：

$$\hat{c}_{H_{r+k}}^{(r+k)} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \cdot \hat{c}_{H_r}^{(r)}. \tag{18}$$

证明概要。 在任意连续 $k+1$ 轮中，至少有一个服务器会在两轮中被选为成本最高的服务器。我们根据该服务器在这两轮之间是否曾作为低成本服务器参与交换，分两种情况进行分析。在两种情况下，应用前面的声明都可以证明最大成本会按一个常数乘法因子下降。完整论证见附录 C.2。

□

现在我们可以完成定理 8 的证明了。上述声明保证：

- 成本最高的服务器的成本始终会下降（命题 2）。
- 未参与交换的任何服务器，其成本保持不变（命题 2）。
- 在第 k 轮后被选为成本最高服务器的任何服务器，其成本至少会降低乘法因子 $\left(\frac{2+\varepsilon}{2+2\varepsilon}\right)$（声明 3）。

因此，成本最高的服务器的成本会随时间呈几何级数下降，直到满足公式 (1) 中给出的（$1+\varepsilon,\ \beta$）-公平性条件。

在证明的其余部分，我们将给出上界：在所有服务器成本满足公平性准则之前，所需交换的总次数。

根据声明 3，我们知道每经过 k 轮次，最大服务器成本就会缩小为原来的 $\left(\frac{2+\varepsilon}{2+2\varepsilon}\right)$。从第 0 轮开始，我们有：

$$\hat{c}_{H_k}^{(k)} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \hat{c}_{H_0}^{(0)},$$

$$\hat{c}_{H_{2k}}^{(2k)} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)\hat{c}_{H_k}^{(k)} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)^2 \hat{c}_{H_0}^{(0)},$$
$$\vdots$$
$$\hat{c}_{H_{qk}}^{(qk)} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)^q \hat{c}_{H_0}^{(0)}.$$

值得一提的是，$\hat{c}_{H_0}^{(0)}$ 表示执行初始离线最优算法 A 后、进行任何交换之前，所有服务器中的最大成本。由于算法 A 中的任何服务器产生的总成本至多为 OPT，我们有：

$$\begin{aligned}\hat{c}_{H_{qk}}^{(qk)} &\le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)^q \hat{c}_{H_0}^{(0)} \\ &\le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)^q \cdot \mathrm{OPT}.\end{aligned}$$

为确保最终成本至多为 $\frac{(1+\varepsilon)\cdot\mathrm{OPT}}{k}+\beta$，只需运行算法 qk 次，使得：

$$\left(\frac{2+\varepsilon}{2+2\varepsilon}\right)^q \cdot \mathrm{OPT} \le \frac{(1+\varepsilon)\cdot\mathrm{OPT}}{k}.$$

求解 q，得：

$$q \ge \log_{\frac{2+2\varepsilon}{2+\varepsilon}}\left(\frac{k}{1+\varepsilon}\right) = \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k - \log_{\frac{2+2\varepsilon}{2+\varepsilon}}(1+\varepsilon).$$

将交换总次数设为 $qk = k\log_{2+\varepsilon} k$，即可满足公平性保证以及关于 r 的声明 1、2和 3 的条件。对于任意 $\varepsilon > 0$，有 $\log_{\frac{2+2\varepsilon}{2+\varepsilon}} k = \log_{\frac{2+2\varepsilon}{2+\varepsilon}} k = \Theta\left(\frac{1}{\varepsilon}\log k\right)$，因此交换总次数为 $O\left(\frac{1}{\varepsilon}\cdot k\cdot\log k\right)$。由此完成证明：至多经过 $O\left(\frac{1}{\varepsilon}\cdot k\cdot\log k\right)$ 次交换后，算法 1 将返回一个 $(1+\varepsilon,\beta)$-公平解，正如所声称的那样。 □

C.2 声明 3 的证明

命题 4（命题 3 的重述）。设 r 是算法 *1* 中任意一轮，且满足 $r \le k\cdot$对数$_{\frac{2+2\varepsilon}{2+\varepsilon}} k-k$，并假设该算法在 r之后又运行了 k 轮。则：

$$\hat{c}_{H_{r+k}}^{(r+k)} \le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right)\cdot\hat{c}_{H_r}^{(r)}.$$

证明。假设我们从第 r轮开始，连续运行算法 1 共 $k+1$ 轮。每一轮中，恰有一台服务器被指定为成本最高的服务器并被交换。由于服务器只有 k 台，而轮次有 $k+1$ 轮，根据抽屉原理，至少有一台服务器在不同的两轮中被选为成本最高的服务器。

形式化地说，存在 存在一个服务器 $b\in[k]$ 以及两个不同的轮次 $r+m_1$和 $r+m_2$，满足 $0\le m_1 <$

$m_2 \le k$，使得 $H_{r+m_1} = H_{r+m_2} = b$. 也就是说，服务器 b 在轮次 $r+m_1$ 和 $r+m_2$中都被指定为成本最高的服务器。我们区分两种情况：

情形（I）。在$r+m_1$ 轮次与 $r+m_2$轮次之间的时间段内，服务器 b 从未作为低成本服务器被交换。

情形（II）。在该时间段内，服务器 b可能作为低成本服务器被交换。

我们首先考虑情形（I）。由于 b 在 $r+m_1$轮次和 $r+m_2$ 轮次中均被指定为成本最高的服务器，且期间未作为低成本服务器被交换，因此其成本在 $r+m_1+1$ 轮次和 $r+m_2$轮次之间保持不变。现在，我们逐步应用之前的各项命题：

$$
\begin{aligned}
\hat{c}_{H_{r+k}}^{(r+k)} &\le \hat{c}_{H_{r+m_2}}^{(r+m_2)} && \text{(by Claim 2, decreasing max cost)} \\
&= \hat{c}_b^{(r+m_2)} && \text{(since } H_{r+m_2} = b\text{)} \\
&= \hat{c}_b^{(r+m_1+1)} && \text{(no swaps involving } b \text{ between } r+m_1+1 \text{ and } r+m_2\text{)} \\
&\le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \cdot \hat{c}_b^{(r+m_1)} && \text{(by Claim 1, applied at } r+m_1\text{)} \\
&= \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \cdot \hat{c}_{H_{r+m_1}}^{(r+m_1)} && \text{(since } H_{r+m_1} = b\text{)} \\
&\le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \cdot \hat{c}_{H_t}^{(r)} && \text{(again by Claim 2).}
\end{aligned}
$$

现在考虑情形（II）。假设服务器 b 在$r+m_1$ 轮次和 $r+m_2$轮次之间至少一次作为低成本服务器被交换。令 $t+m_3$ 表示发生这种情况的最后一轮次，且 $m_1 < m_3 < m_2$。那么，根据命题 2（最大成本非递增），我们有：

$$
\begin{aligned}
\hat{c}_{H_{r+k}}^{(r+k)} &\le \hat{c}_{H_{r+m_2}}^{(r+m_2)} && && (19) \\
&= \hat{c}_b^{(r+m_2)} = \hat{c}_b^{(r+m_3+1)} \\
&\le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \cdot \hat{c}_{H_{r+m_3}}^{(r+m_3)} && \text{(by Claim 1)} && (20) \\
&\le \left(\frac{2+\varepsilon}{2+2\varepsilon}\right) \cdot \hat{c}_{H_r}^{(r)} && \text{(by Claim 2).} && (21)
\end{aligned}
$$

因此，在情形（I）和情形（II）中，经过 k 轮次后，服务器成本的最大值至少降低至原来的 $\left(\frac{2+\varepsilon}{2+2\varepsilon}\right)$倍，从而完成了声明 3 的证明。 □

“

C.3 定理 9 的证明

定理 22 （定理 9 的重述）。设 A 是任意在线 k-服务器算法， σ 是任意请求序列。那么，参数为 $\gamma > 0$ 的算法 *2* 满足以下性质：

- 每台服务器产生的期望成本为

$$\frac{cost(A,\sigma)}{k}+O\left(cost(A,\sigma)^{1/(\gamma+1)}\cdot\mathrm{diam}\right).$$

- 更重要的是，对于任意 $\varepsilon>0$，其概率至少为

$$1-k\cdot\exp\left(-\Omega\left(\frac{\varepsilon^2}{\gamma\cdot k}\cdot cost(A,\sigma)^{1/(\gamma+1)}\right)\right),$$

每台服务器产生的成本至多为

$$(1+\varepsilon)\cdot\frac{cost(A,\sigma)}{k}+O\left(cost(A,\sigma)^{1/(\gamma+1)}\cdot\mathrm{diam}\right).$$

证明。对于 $j\in[m]$，令 $\phi_j=j^\gamma$ 表示阶段 j的长度。令 X_{ij} 表示在阶段 j分配给服务器 i 的成本。不考虑交换时，我们将分配给服务器 $i\in[k]$ 的总成本定义为：

$$c_i'=\sum_{j=1}^{m}X_{ij}.$$

对于 $j\in[m]$，我们将 U_j 定义为每个阶段的总成本（见第 5 行），即分配给各服务器的所有成本之和：

$$U_j=\sum_{i=1}^{k}X_{ij}\le\phi_j+\mathrm{diam}.$$

请注意，在每个阶段 j中，我们都保持不变量 $U_j<\phi_j$ 直至该阶段结束，而最后一个请求最多会使阶段成本增加 diam。对于最后阶段，由于不会再发生阶段变更，因此有 $U_m\le\phi_m$。

每台服务器 i 的期望成本 c_i 为 c_i' 加上其交换成本，而交换成本至多为 $m\cdot\mathrm{diam}$：

$$\mathbb{E}[c_i]\le E[c_i']+m\cdot\mathrm{diam}=\mathbb{E}[X_{i1}]+\mathbb{E}[X_{i2}]+\cdots+\mathbb{E}[X_{im}]+m\cdot\mathrm{diam},$$

由于每个阶段内服务器 ID 的分配是均匀随机的，我们有：

$$\mathbb{E}[X_{ij}]=\frac{\sum_{i=1}^{k}X_{ij}}{k}=\frac{U_j}{k}.$$

因此，回顾算法 A 产生的总成本为 $\mathrm{cost}(A)=\sum_{j=1}^{m}U_j$，服务器 i 的预期总成本为：

$$\mathbb{E}[c_i]\le\sum_{j=1}^{m}\frac{U_j}{k}+m\cdot\mathrm{diam}=\frac{\mathrm{cost}(A)}{k}+m\cdot\mathrm{diam}.\tag{22}$$

为了将阶段数 m 与总成本 $\mathrm{cost}(A,\ \sigma)$ 联系起来，我们对各阶段产生的累积成本进行界定。根据算法 2 的构造，在每个阶段 $j<m$ 中，只有当累积成本达到阈值 ϕ_j后，该阶段才会结束，而服务最后一个请求可能会使

阶段成本最多增加 diam。因此，阶段 j 产生的成本至多为 ϕ_j +diam。对于最后阶段，不会再发生阶段切换，因此 $U_m \le \phi_m$。由此可得

$$\sum_{j=1}^{m-1} \phi_j \le \mathrm{cost}(A) \le \sum_{j=1}^{m} (\phi_j + \mathrm{diam}) = \sum_{j=1}^{m} j^\gamma + m \cdot \mathrm{diam}. \tag{23}$$

我们使用基于积分近似的标准界：

$$\int_1^m x^\gamma \, dx \le \sum_{j=1}^{m} j^\gamma \le \int_1^{m+1} x^\gamma \, dx,$$

由此得到

$$\sum_{j=1}^{m} j^\gamma = \Theta\left(\frac{m^{\gamma+1}}{\gamma+1}\right) \quad \text{for } \gamma > -1. \tag{24}$$

对于固定常数 $\gamma > 0$，利用公式（23）和公式（24），我们得出：

$$\mathrm{cost}(A) = \Theta\left(\frac{m^{\gamma+1}}{\gamma+1}\right), \quad \text{so} \quad m = \Theta\left(\mathrm{cost}(A)^{\frac{1}{\gamma+1}}\right).$$

将其代回关于服务器 i的期望成本的公式（22），我们得到：

$$\mathbb{E}[c_i] \le \frac{\mathrm{cost}(A)}{k} + O\left(\mathrm{cost}(A)^{\frac{1}{\gamma+1}} \cdot \mathrm{diam}\right).$$

1因此，对于任意固定常数 $\gamma > 0$，该算法在期望意义下是(1, $O(\mathrm{cost}(A)^{\gamma+1} \cdot \mathrm{diam})$-公平的（即事前公平）。

现在我们证明，该算法以高概率也是公平的（即事后公平），这是更有意义的结果。我们将使用伯恩斯坦不等式证明服务器随机成本的集中性。

事实1（伯恩斯坦不等式）。设 $X_1, \ldots, X_n$ 是独立随机变量，并假设每个变量都几乎必然满足 $|X_i| \le M$。设 $S_n = \sum_{i=1}^n X_i$，且令 $\sigma^2 = \sum_{i=1}^n \mathrm{Var}(X_i)$。那么对于任意 $\varepsilon > 0$,

$$\Pr\left(S_n - \sum_{i=1}^{n} \mathbb{E}[X_i] \ge \varepsilon\right) \le \exp\left(-\frac{\varepsilon^2}{2\sigma^2 + \frac{2}{3}M\varepsilon}\right).$$

为避免阶段溢出带来的复杂情况（即某个阶段的最后一个请求可能导致总成本超过阶段阈值 ϕ_j），我们定义新的随机变量 Y_{ij} 如下：令 Y_{ij}表示阶段 j中分配给服务器 i 的成本，仅计至达到阶段阈值 ϕ_j 为止。也就是说，

$$Y_{ij} \le X_{ij} \le Y_{ij} + \mathrm{diam},$$

由于每个阶段至多有一个请求超过阈值。

现在我们定义 D_i:

$$D_i := \sum_{j=1}^{m} Y_{ij},$$

它表示分配给服务器 i 的总成本，不包括阶段溢出项和交换成本。现在，我们使用伯恩斯坦不等式分析 D_i 的集中性。

$$\mathrm{Var}(Y_{ij}) = \mathbb{E}[Y_{ij}^2] - \mathbb{E}[Y_{ij}]^2 \le \phi_j^2 \left(\frac{1}{k} - \frac{1}{k^2}\right) < \frac{\phi_j^2}{k}.$$

下一步，定义

$$Z \;\coloneqq\; \sum_{j=1}^{m-1} j^{\gamma} \;+\; U_m.$$

于是，由于对于 $j < m$， $\mathbb{E}[Y_{ij}] = j^{\gamma}/k$ 和 $\mathbb{E}[Y_{im}] = U_m/k$，我们有

$$\mathbb{E}[D_i] \;=\; \sum_{j=1}^{m} \mathbb{E}[Y_{ij}] \;=\; \frac{1}{k}\left(\sum_{j=1}^{m-1} j^{\gamma} + U_m\right) \;=\; \frac{Z}{k}.$$

现在，我们应用伯恩斯坦不等式（事实1）来界定 D_i 显著高于其期望的概率（这也能将 c_i 界定在相差至多一个加性项 $2m$ ·直径的范围内）：

$$\Pr\left(D_i - \mathbb{E}[D_i] \ge \varepsilon \cdot \frac{Z_i}{k}\right) = \Pr\left(D_i \ge (1+\varepsilon) \cdot \frac{Z_i}{k}\right) \le \exp\left(\frac{-\varepsilon^2 \cdot Z_i^2/k^2}{2\sigma^2 + \frac{2}{3}M \cdot \varepsilon \cdot Z_i/k}\right),$$

其中： $M = \phi_m = m^{\gamma}$ 是任何 Y_{ij}可能取得的最大值，总方差满足：

$$\sigma^2 \le \sum_{j=1}^{m} \mathrm{Var}(Y_{ij}) \le \frac{1}{k}\left(\sum_{j=1}^{m-1} \phi_j^2 + U_m^2\right) \le \frac{1}{k}\left(\sum_{j=1}^{m} \phi_j^2\right),$$

其中 $U_m \le \phi_m$ 是最后阶段的成本， ε 是任意固定常数。

为完成证明，我们验证伯恩斯坦不等式分母中的两项渐近地都被分子所支配。

回顾一下，

$$Z_i = \sum_{j=1}^{m-1} \phi_j + U_m.$$

由于 $\phi_j = j^{\gamma}$ 和 $U_m \le \phi_m$，我们有

$$Z_i = \Theta\left(\sum_{j=1}^{m} j^{\gamma}\right) = \Theta\left(\frac{m^{\gamma+1}}{\gamma+1}\right).$$

由此可得

$$Z_i^2 = \Theta\left(\frac{m^{2\gamma+2}}{(\gamma+1)^2}\right), \qquad \frac{Z_i^2}{k^2} = \Theta\left(\frac{m^{2\gamma+2}}{k^2(\gamma+1)^2}\right).$$

接下来，回顾

$$\sigma^2 = \sum_{j=1}^{m} \mathrm{Var}(Y_{ij}) \le \frac{1}{k}\sum_{j=1}^{m} \phi_j^2.$$

由于 $\phi_j = j^\gamma$，可得

$$\sigma^2 = O\left(\frac{1}{k}\sum_{j=1}^{m} j^{2\gamma}\right) = O\left(\frac{m^{2\gamma+1}}{k(2\gamma+1)}\right).$$

对于伯恩斯坦不等式，这些随机变量均有界，且

$$|Y_{ij}| \le M \coloneqq \phi_m.$$

因此，

$$MZ_i = \phi_m Z_i = \Theta\left(m^\gamma \cdot \frac{m^{\gamma+1}}{\gamma+1}\right) = \Theta\left(\frac{m^{2\gamma+1}}{\gamma+1}\right),$$

进而，

$$\frac{2}{3}\cdot\frac{MZ_i}{k} = \Theta\left(\frac{m^{2\gamma+1}}{k(\gamma+1)}\right).$$

将上面推导出的 $\mathbb{E}[D_i]$、σ^2 和 $M = \phi_m$ 的界代入伯恩斯坦不等式，并利用 $0 < \varepsilon \le 1$，得到

$$\Pr\left(D_i \ge (1+\varepsilon)\mathbb{E}[D_i]\right) \le \exp\left(-\Omega\left(\frac{\varepsilon^2}{\gamma}\cdot\frac{m}{k}\right)\right).$$

其中 $m =$ 成本$(A)^{1/(\gamma+1)}$。现在，对于固定的 $\varepsilon > 0$，我们得出结论：概率至少为

$$1 - \exp\left(-\varepsilon^2 \cdot \frac{1}{\gamma}\cdot\Omega\left(\mathrm{cost}(A)^{1/(\gamma+1)}/k\right)\right),$$

我们有

$$D_i < (1+\varepsilon)\cdot\frac{\mathrm{cost}(A)}{k}.$$

由于实际成本满足 $c_i \le D_i + 2m\cdot\mathsf{diam}$，因此以相同的概率，

$$c_i \le (1+\varepsilon)\cdot\frac{\mathrm{cost}(A)}{k} + O\left(\mathrm{cost}(A)^{1/(\gamma+1)}\cdot\mathrm{diam}\right).$$

最后，为确保这一结论对所有服务器同时成立，我们应用并集界。其概率至少为

$$1 - k\cdot\exp\left(-\varepsilon^2\cdot\frac{1}{\gamma}\cdot\Omega\left(\mathrm{cost}(A)^{1/(\gamma+1)}/k\right)\right),$$

我们有

$$\forall i \in [k]: \quad c_i \leq (1+\varepsilon) \cdot \frac{\mathrm{cost}(A)}{k} + O\left(\mathrm{cost}(A)^{1/(\gamma+1)} \cdot \mathrm{diam}\right),$$

正如所声称的。

□

备注 3（选择目标失败概率）。 固定 ε，$\gamma > 0$ 以及一个目标失败概率$\zeta \in (0, 1)$。定理 9 保证失败概率至多为 ζ，前提是

$$k \cdot \exp\left(-\Omega\left(\frac{\varepsilon^2}{\gamma k} \cdot \mathrm{cost}(A,\sigma)^{1/(\gamma+1)}\right)\right) \leq \zeta.$$

等价地，只需满足

$$\mathrm{cost}(A,\sigma) \;\geq\; \left(\Theta\left(\frac{\gamma k}{\varepsilon^2} \cdot \log\frac{k}{\zeta}\right)\right)^{\gamma+1}.$$

C.4 定理 10 的证明

定理 23（定理 10 的重述）。 设 σ 是线度量上的任意请求序列，其直径为 diam，并设 c_i 表示服务器 i 使用双重覆盖算法（*DCA*）服务 σ 时产生的总成本。那么，对于任意 i、$j \in [k]$,

$$|c_i - c_j| = O(k \cdot \mathrm{diam}).$$

因此，线度量上的 *DCA* 是$(1,\ O(k \cdot \mathrm{diam}))$-公平的。

证明。 我们考虑在线度量上的双重覆盖算法（DCA）。由于 DCA 从不允许服务器彼此超越，因此我们可以根据服务器的初始位置从左到右将其编号为 1，2，.....，k并且在整个执行过程中，这一顺序始终保持不变。

对于每个服务器 i，令 R_i 表示它向右移动的总距离，令 L_i 表示它向左移动的总距离。因此，服务器 i 产生的总成本为

$$c_i = R_i + L_i.$$

在直线上的 DCA 中，当且仅当请求到达服务器 i 和 $i+1$ 之间时，服务器才会向右移动；此时，这两台服务器以相同速度朝请求移动，直到请求得到处理。类似地，当且仅当请求到达服务器 $i-1$ 和 i 之间时，服务器 i 才会向左移动。因此，服务器 i 每向右移动一个单位，服务器 $i+1$ 就会相应地向左移动相同的距离，因此

$$R_i = L_{i+1}.$$

将服务器 i 的净位移定义为

$$d_i \coloneqq R_i - L_i.$$

由于所有服务器移动都发生在线段内，而该线段的直径为 diam，因此对于所有 i，都有 $|d_i| \le$ diam。

现在我们来关联相邻服务器的成本。根据上述定义，

$$c_i = R_i + L_i = 2R_i - d_i,$$
$$c_{i+1} = R_{i+1} + L_{i+1} = 2L_{i+1} + d_{i+1}.$$

利用 $R_i = L_{i+1}$，我们得到

$$c_{i+1} - c_i = d_{i+1} + d_i.$$

对于任意 $j \ge 1$，服务器 i 和 $i+j$ 之间的差值可以写成望远镜求和：

$$c_{i+j} - c_i = \sum_{\ell=0}^{j-1} (c_{i+\ell+1} - c_{i+\ell}) = d_{i+j} + 2d_{i+j-1} + \cdots + 2d_{i+1} + d_i.$$

由于对于所有 ℓ 都有 $|d_\ell| \le$diam，我们得出

$$|c_{i+j} - c_i| \le 2j \cdot \mathrm{diam} \le 2(k-1) \cdot \mathrm{diam}.$$

因此，任意两个服务器之间的成本差异都由 $O(k \cdot$diam$)$ 界定，这意味着 DCA 在线度量上是 $(1, O(k \cdot \mathrm{diam}))$-公平的（加性公平算法） □

C.5 定理11省略的细节

证明的关键思路是将各个服务器的成本与两台服务器之间的距离如何随时间演化联系起来。对于 $k = 2$，每次移动都会使服务器之间的距离增加或减少，而这一变化恰好由发散项和收敛项刻画。引理 4 通过利用度量的直径，对发散与收敛之间的不平衡进行了界定，从而将这一直觉形式化。

引理 4。对于任意输入 σ 以及 $k = 2$ 设置中的任意服务器 i，有

$$-\mathrm{diam} \le div_i(\sigma) + div_{i'}(\sigma) - 2\,con_i(\sigma) \le \mathrm{diam}, \tag{25}$$

其中，直径是该度量的直径。

证明。令 g 表示服务器 i 与 i' 之间的初始距离。我们跟踪它们在每一步之间的距离。每当恰好一台服务器偏离另一台服务器时，距离就增加偏离的量；每当两台服务器都靠近时，距离就减少它们各自靠近量的两倍。

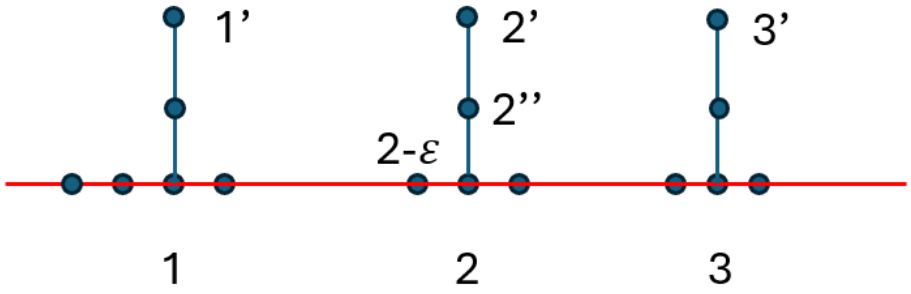

图 2：定理 12 证明中所用树度量的示意图。红色水平线表示该度量的主干，主要节点标记为 1、2、3。每个主要节点都连接着一个次要节点，两者距离为 ε ，该次要节点又连接着一个叶节点，距离为ε。此外，我们还在 $i \pm \varepsilon$ 处设置了节点，并在 $1 - 2\varepsilon$处设置了一个节点。该构造突出了度量的几何结构；证明中使用的请求序列利用这一结构，使服务器成本产生巨大的不平衡。

因此，

$$distance(i, i') \;=\; g + div_i(\sigma) + div_{i'}(\sigma) - 2\,con_i(\sigma).$$

在序列结束时，距离至多为 diam，因此我们得到

$$0 \;\le\; g + div_i(\sigma) + div_{i'}(\sigma) - 2\,con_i(\sigma) \;\le\; \text{diam}.$$

又由于 $g \le$diam，因此可得

$$-\text{diam} \;\le\; div_i(\sigma) + div_{i'}(\sigma) - 2\,con_i(\sigma) \;\le\; \text{diam}.$$

□

C.6 定理 12 的证明

定理 24 （定理 12 的重述）。存在一个树度量和一个请求序列σ ，使得在 σ上运行双重覆盖算法（*DCA*）时，某个服务器的成本为$\Omega(k \cdot \textsc{Opt}(\sigma))$，其中 $\textsc{Opt}(\sigma)$ 表示离线最优解的成本。

证明。我们首先描述 困难树度量，然后给出该度量上的困难请求序列。

度量。 固定一个较小的 ε。令 主干 为一条顶点为 1、2、.....、 $k{+}2$ 且边长均为单位的路径 $d(i, i{+}1) = 1$。将每条主干边靠近两端的位置细分，分别在 $(i{-}1, i)$ 上添加顶点 $i - \varepsilon$，并在 $(i, i{+}1)$ 上添加顶点 $i + \varepsilon$（因此沿主干有 $d(i, i - \varepsilon) = d(i, i + \varepsilon) = \varepsilon$）。对每个 $i \in \{1, \ldots, k{+}2\}$，在 i 处连接一条长度为 2ε 的叶节点（即由两条 ε-边组成的双边路径）。令 i' 表示该叶节点的远端点，并

令 i'' 表示该叶节点的中点，即距离 i 为 ε 的点。

最后，通过添加两个额外的点来延长主干：在顶点 1 左侧距离 2ε 处添加点 $1-2\varepsilon$，并在顶点 $k+2$ 右侧距离 2ε 处添加点 $k+2+2\varepsilon$。

初始配置。在 $1-2\varepsilon$、3、4、5、.....、k+1 处放置服务器（2 处没有服务器）。我们关注最左侧的服务器 s_1（从 $1-\varepsilon$ 开始）。

请求序列（一次右向扫描；简略形式）。我们按以下顺序发出请求：

$$2',\ 3',\ 4',\ \ldots,\ k',\quad k{+}2{+}2\varepsilon,\quad k{+}2{+}\varepsilon.$$

然后，我们将右侧的模式镜像 过来（将 i 替换为 $k{+}3{-}i$，将 ε 替换为 $-\varepsilon$），执行一次向左扫描并恢复初始模式；重复此过程即可得到任意长的序列。

每次请求之后的 DCA 动态。现在我们逐步描述 DCA 的行为：**在 $2'$ 处发出请求**。最初，$s_1 = 1-\varepsilon$、$s_2 = 3$、$s_3 = 4$ 等。通往 $2'$ 的唯一路径上未被阻塞的服务器是 s_1 和 s_2，因此二者都开始移动。时间 1 后，s_1 到达 $2-\varepsilon$，而 s_2 到达 2；此时 s_1 变为阻塞 状态（因为 s_2 位于其通往 $2'$ 的路径上），并停在 $2-\varepsilon$。随后 s_2 独自沿着边 $(2,\ 2')$ 移动距离 2ε，并处理了位于 $2'$ 的请求。因此：

$$s_1(1) = 2-\varepsilon,\quad s_2(1) = 2',\quad s_3(1) = 4,\ s_4(1) = 5,\ \ldots$$

请求于 $3'$。所有到达 $3'$ 的唯一路径畅通无阻的服务器开始移动：s_1、s_2 和 s_3。阻塞事件与时间安排。在时间 $t=\varepsilon$，s_1 到达节点 2；与此同时，s_2（沿叶节点而来）到达边 $(2, 2')$ 的中点顶点 $2''$。由于 s_1 位于 s_2 通往 $3'$ 的路径上，s_2 在此中点处被阻塞并停下。s_1 继续前往 3。

在时间 $t=1$，s_1 位于 $3-\varepsilon$，而 s_3 到达节点 3。由于 s_3 现在位于 s_1 通往 $3'$ 的路径上，s_1 在 $3-\varepsilon$ 处被阻塞 并停下。s_3 独自沿叶节点从 3 继续前往 $3'$，并于时间 $t = 1+2\varepsilon$ 服务该请求。

服务完 $3'$ 后的状态。

$$s_1(2) = 3-\varepsilon,\qquad s_2(2) = \text{midpoint of } (2, 2'),\qquad s_3(2) = 3',\qquad s_4(2) = 5,\ \ldots$$

继续处理 $4'$，$5'$、.....、k'。根据归纳法，在每一步中，指定的最左侧服务器 s_1 沿主干恰好再前进一个单位，并在朝请求 i' 移动后停在位置 $i{+}1{-}\varepsilon$。与此同时，最初位于 $i{+}1$ 的服务器被引向叶节点 i'，随后将移动到中点 $(i-1)''$。因此，在处理完 $i' = 5'$，$6'$、.....、k' 后，我们有

$$s_1(k-1) = k-\varepsilon,\qquad s_j(k-1) = j'' \text{ for } j = 2, 3, \ldots, k-1,\qquad s_k(k-1) = k'$$

总计，s_1 沿主干移动了 $k-1$ 个单位，而每台中间服务器s_j（$2 \le j \le k$）恰好进行了一次长度为 $1+2\varepsilon$ 的往返移动：它在作为请求 j'的右侧搭档时移动了 $1+2\varepsilon$，并在 s_1 前往下一个请求时移动了 ε。

请求位于 $k{+}2{+}2\varepsilon$。 s_k （位于 k'）和 s_1 （位于 $k-\varepsilon$）都未被阻挡，可以朝这个叶节点移动，并开始移动。经过距离 ε后， s_k 到达 k'' ，随后被位于节点 k的 s_1 挡住。从那时起，只有 s_1 继续沿主干移动，并服务位于 $k{+}2{+}2\varepsilon$的请求。因此，在这一步中， s_1 额外移动了 $2+\varepsilon$ 的距离，而 s_k 只移动了 ε。

请求位于 $k{+}2{+}\varepsilon$。此时，所有服务器朝向主干的方向均未受阻。每台停驻在 j'' 的服务器都移动距离 ε ，返回其主干节点 j；例如，服务器 s_k 从 k'' 移动距离 ε 返回 k，而 s_1 则从 $k{+}2{+}2\varepsilon$ 移动距离 ε 返回 $k{+}2{+}\varepsilon$。因此，每台服务器都恰好移动了 ε ，所有服务器均返回主干。所有服务器的位置都与其初始位置相似。

令 f 为如下定义的主干的左右镜像:

$$f(x) \coloneqq (k+3) - x,$$

通过映射 $i' \mapsto (k{+}3{-}i)'$、 $i'' \mapsto (k{+}3{-}i)''$ 和 $(i \pm \varepsilon) \mapsto (k+3-i) \mp \varepsilon.$，自然地扩展到细分点和叶节点。令 $\mathbf{s(0)} = (s_1(0) \ldots, s_k(0))$ 表示初始服务器配置，并令 $\mathbf{s}^{(\mathrm{R})}$ 表示右向扫描结束时的配置（即处理完请求 $k+2+\varepsilon$ 后）。那么，$\mathbf{s}(k+1)$ 在镜像对称下与 $\mathbf{s}^{(0)}$ 具有相同的模式，即:

$$\big(s_1(\mathrm{k}+1),\, s_2(\mathrm{k}+1),\, \ldots,\, s_k(\mathrm{k}+1)\big) = \big(f(s_1(0)),\, f(s_2(0)),\, \ldots,\, f(s_k(0))\big).$$

因此，发出镜像请求序列（将 x 替换为 $f(x)$，也就是说，将 i 替换为 $k{+}3{-}i$ ，并将 ε 替换为 $-\varepsilon$）会产生对称的向左扫描，并使服务器回到原始配置。

完成这次扫描后，我们反向操作，使 s_1 回到最左侧位置，然后重复这一过程。假设重复 r 次，则 s_1 服务器的成本为

$$c_1 = r \cdot (k+1+4\varepsilon) = \Theta(rk) \tag{26}$$

所有其他服务器产生的总成本为

$$\sum c_i = r \cdot (1+3\varepsilon) \cdot k = \Theta(rk). \tag{27}$$

此外，最优解的总成本小于 $r \cdot (2+8\varepsilon) + ((k-2) \cdot 2\varepsilon) = \Theta(r)$，因为该最优解只需移动 s_1 来完成 2’，随后每台服务器都可以在自己的叶节点处理请求，成本为 2ε；服务器 k 可以处理位于 $k+2+2\varepsilon$ 的请求，然后返回到 $k+2+\varepsilon$位置。因此，总成本将是每次扫描中 $1+3\varepsilon$ 的移动成本（共 s_1次）、每台中间服务器的 2ε ，以及 $2+5\varepsilon$ 在每次扫描中的 s_k （除了第一次扫描，因为第一次需要执行 3ε 而不是 5）。

因此，我们得到 s_1 支付成本$\Omega(k \cdot \mathrm{OPT}(\sigma))$，如所声称的那样。

□