【问题的形式化描述】

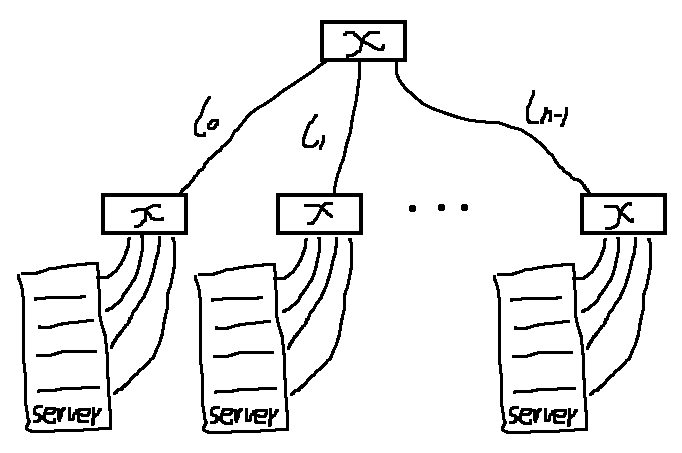
假设计算集群中存在台服务器，运行有个虚拟机。每个虚拟机需要一定的资源，设虚拟机需要占用个CPU，（MB）的内存和（GB）的硬盘。每台服务器可以提供一定的资源，设服务器能够供给个CPU，的内存和（GB）的硬盘。虚拟机两两之间有网络通信产生的数据流量。假设经过某常数的时间，虚拟机与间产生的网络流量为（）。

我们的目标是设计一个策略在计算集群中选择（）个虚拟机进行迁移操作（从一台服务器上迁移到另一台服务器上），使得跨机架的链路的最大网络流量尽可能小。

问题的解决主要分为两部分：计算一个新的虚拟机放置格局；计算从当前的虚拟机放置格局转换为新格局的迁移方式。本章将着重讨论第一部分，即新的虚拟机放置格局的求解。第二部分迁移方式将在下一章算法细节部分中讨论。

1 网络拓扑结构的假定

假设网络的拓扑结构为高度为的树形结构，即所有机架上的交换机连接至同一台交换机上。如图所示 脚注：交叉信息研究院系统组实验室的计算集群采用的是此种网络拓扑结构 我们优化的对象是跨机架的网络流量，即图中链路上的网络流量。



2 混合整数线性规划问题的构造

只要求未知变量中的部分变量为整数变量的线性规划问题叫做混合整数规划问题。引用文献？本节介绍如何通过构造并求解一个混合整数线性规划问题来求解新的虚拟机放置格局。求解得到的结果为每个虚拟机（）在新的放置格局中应该位于的机架。引用到后文缩小问题规模处

设计算集群中存在个机架，定义整数变量描述虚拟机放置的位置：



由每个虚拟机的存在性和唯一性，我们引入变量约束



记选择出的待迁移的$m’$个虚拟机构成集合引用到下文缩小规模处。（）链路上的网络流量由三部分组成：不在中的虚拟机之间进行通信产生的网络流量，记为；中的虚拟机和不在中的虚拟机之间进行通信产生的网络流量，记为；中的虚拟机之间进行通信产生的网络流量，记为。即



其中







上面那个公式整理为



下面那个公式整理为



定义变量表示网络流量最大的链路上的网络流量，添加约束条件



我们需要最小化的优化目标函数为



这是一个混合整数二次规划问题。为了简化问题模型，我们需要通过引入新的变量和约束，将约束上面公式中关于变量的二次项消除，从而将问题转换为混合整数（线性）规划问题。具体地，我们引入整数变量，令



则应有



因此那个麻烦的公式化简为



同时我们需要考虑虚拟机的迁移过程产生的开销引用到下文细节讨论3处。用常量表示虚拟机原先所处的位置：



设虚拟机（）进行迁移所带来的开销为，设所有虚拟机进行迁移带来的总开销为，则有



省略常数项对混合整数规划问题的求解无影响，因此我们将上式简化为



因此总的目标函数为



综上所述，我们构造了混合整数线性规划问题VMP如下：

**VMP:**

minimize



subject to



variables

【算法细节讨论】

1 合理缩小问题求解的规模

混合整数规划问题被分类为NP困难问题。本文上节中构建的VMP问题是混合整数规划问题中的一类特殊问题——混合0-1整数规划问题，它亦被分类为NP困难问题。引用文献？Karp's 21 problems因此对于该问题尚不存在高效求解的算法。如果将OpenStack虚拟机的放置问题形式化为求解每台虚拟机应当被放置于哪台服务器，则构造出的混合0-1整数规划问题将包含个未知变量和个约束。在真实计算集群规模较大时脚注：实验室的计算集群规模如何，该问题的求解时间将过长。因此为了提高求解策略的高效性和实用性，我们有必要采用启发式的算法，合理地缩小问题的规模。

本文提出的OpenStack虚拟机放置策略同时采取两种策略缩小问题求解的规模脚注：变量的数目和约束的数目：选取部分虚拟机而不是全部虚拟机进行计算求解它们应当被迁移到的目标节点引用到第2节；计算虚拟机应当被迁移到哪个机架，而不是计算虚拟机应当被迁移到哪台服务器。一方面，直观上看，这两种策略分别以一定比例减小了变量和的值；另一方面，这两种策略也简化了问题本身的数学模型，VMP问题包含的未知变量数目和约束数目分别为和。实验测试表明，这两种策略的采用使得问题求解的时间极大地缩短，同时算法对网络流量的优化效果依旧显著。引用后文实验环节运行时间部分

2 待迁移虚拟机的选取

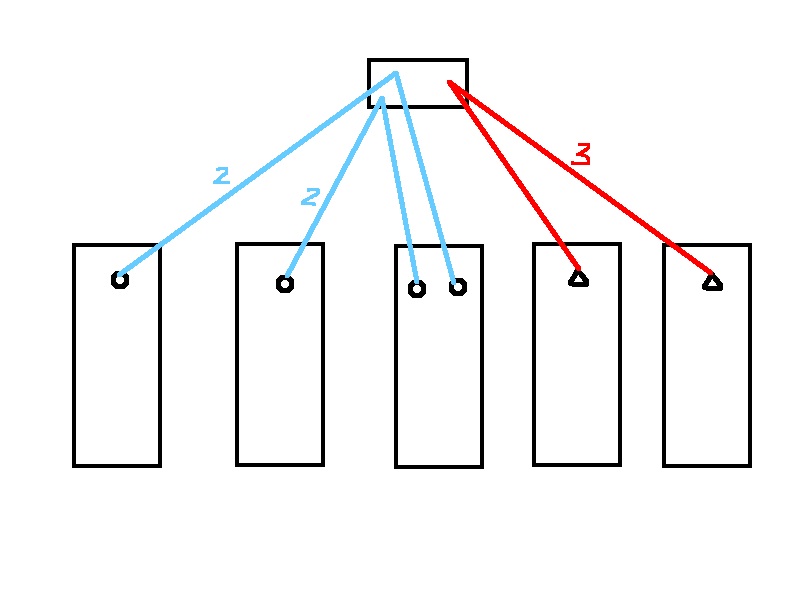
采用启发式算法选取部分虚拟机而不是全部虚拟机进行迁移调度的合理性是基于以下的经验规律：通常情况下大部分虚拟机间的网络通信流量相对较小，因而它们不是造成链路间网络流量分布不均匀关键因素；计算集群中个别链路上出现网络拥塞往往是由某几个虚拟机之间的密集网络通信造成。此处需要实验说明？因此我们只需要关注某些关键虚拟机的调度，便可以有效地缓解链路上网络拥塞的问题。相关测试实验的结果引用后文收敛性也支持这一点。

我们实现的OpenStack虚拟机放置策略设计了两种选取待迁移虚拟机的方式：选取跨机架网络通信密集的虚拟机，选取对网络流量最大的链路贡献大的虚拟机。

2.1选取跨机架网络通信密集的虚拟机

计算每个虚拟机与所有和它处在不同机架上的虚拟机的网络通信的流量之和，选取其中网络流量最大的（OpenStack管理员指定的参数）个虚拟机作为待迁移虚拟机。

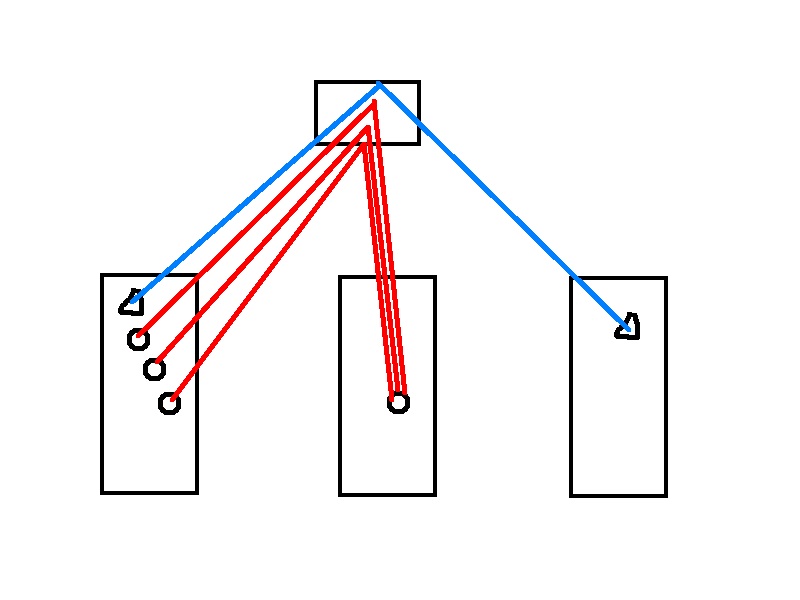
采用这种方式可以通过调度一些网络通信密集的虚拟机对一些链路上的网络流量进行优化。但单独采用这种方式并不保证一定能缓解流量最大的链路上网络拥塞的问题。如下图所示（应该挪圈儿 实际挪三角）



2.2 选取对网络流量最大的链路贡献大的虚拟机

计算每条链路上的网络流量总和，选取网络流量最大的（一条或条）链路，并计算每个虚拟机对这条链路网络流量产生了多少贡献。选取其中贡献最大的（OpenStack管理员指定的参数）个虚拟机作为待迁移虚拟机。

采用这种方式可以直接地有效减小网络最繁忙的链路上的网络流量。这种方式在某条链路上发生网络拥塞的场合可以通过迁移某些虚拟机有效地缓解网络拥塞问题。但是单独采用这种方式不一定是缓解网络拥塞问题的最高效的方式。如下图所示（应该挪中间的圈儿，实际挪左边的圈儿或者三角）

 我们实现的OpenStack虚拟机放置策略结合了2.1和2.2中介绍的两种选取待迁移虚拟机的方式，即最终选取的待迁移虚拟机由分别通过两种方式选取的虚拟机组合产生。同时OpenStack管理员可以自行设计待迁移虚拟机的选取方式并运用到我们实现的OpenStack虚拟机放置策略中。

3 迁移过程引入的开销

OpenStack对虚拟机的迁移分为两种：非在线迁移（non-live migration）和在线迁移（live migration）线上资料：configuring migraions。不同的迁移过程对虚拟机的影响不同，比如非在线迁移过程中虚拟机会被关闭一段时间，在线迁移过程中虚拟机的中断时间几乎为零。因此，不同的迁移方式会给用户或给计算集群的工作性能带来不同程度的负面影响，如使用户无法访问虚拟机，或给虚拟机上运行的计算任务的完成时间带来额外的延迟等。因此我们引入迁移开销系数来衡量迁移过程对目标函数带来的影响引用到上文VMP问题构造处。OpenStack管理员可以根据实际情况给每个虚拟机分别设定迁移开销系数，并作为输入传给我们实现的OpenStack虚拟机放置策略引用到下文编程实现处。

4 固定不能移动的虚拟机

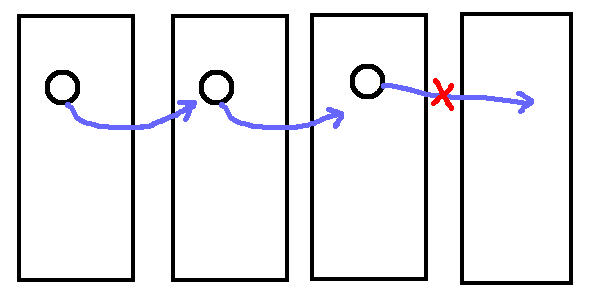
OpenStack管理员在调用本文提出的OpenStack虚拟机放置策略时，可以指定固定一部分虚拟机不被迁移。若计算集群中OpenStack被配置为采取非在线迁移方式线上资料：configuring migraions，虚拟机在被迁移时会被关闭一段时间，直至迁移操作完成之后才会被重新开启。如果虚拟机的用户此时在与虚拟机进行实时交互（在线会议，网络聊天，操作交易，DOTA团战等），虚拟机的非在线迁移可能会对用户造成较大的负面影响。因此我们实现的OpenStack虚拟机放置策略支持OpenStack管理员指定一个虚拟机的集合，算法将保证这些虚拟机不会被迁移。这与OpenStack管理员将集合中的虚拟机的迁移开销系数设置得足够大是等效的。但单独设置这项功能可以使OpenStack管理员能够在一些普遍出现的场景中更便捷地调用实现的虚拟机放置策略。

5 从机架中选取目标服务器

VMP问题求解的结果是一系列变量表示新格局中每个虚拟机应当处于的机架。当某个虚拟机需要迁移到别的机架上时，我们需要一个调度算法来决定它应当被放置到哪台服务器上。我们借鉴了OpenStack默认的调度器Filter Scheduler引用到网页scheduling采用的调度方式，用贪心法来选取虚拟机迁移的目标服务器。首先我们计算并获得机架中所有剩余资源（cpu数目，内存大小，硬盘空间大小）满足虚拟机需求的服务器，接着我们选取某种剩余资源（默认为内存）的量最大的服务器，作为虚拟机迁移的目标服务器。OpenStack管理员可以自行设计从机架中选取服务器的方式并运用到我们实现的OpenStack虚拟机放置策略中。

需要注意的是，当VMP的求解确定了某台虚拟机迁移的目标机架后，在上可能不存在能够满足资源需求的服务器。我们将在【存在的问题】中讨论这一点。当这种情况发生时，我们将放弃对虚拟机的迁移操作，使其留在原来的服务器上。

虚拟机的迁移将导致服务器可用资源的改变。设有两个待迁移虚拟机和，的目标机架是当前所处的机架。不迁移则无法容纳，将被迁移到的中非的服务器上。然而的迁移使的可用资源增加，从而使得可以容纳。如果我们选择目标服务器的算法对迁移的次序有所假定，并根据迁移的次序将资源的改变量也考虑在内，则可能做出将虚拟机迁移到的决策。这样的算法似乎能获得更优的解，然而却存在着隐患。首先OpenStack无法保证对虚拟机实际的迁移按照给定的次序进行引用到第6节，其次即使虚拟机迁移的次序是可以指定的，在真实的计算集群中，可能会有各种不可控的工程因素使得对某个虚拟机的迁移失败。根据上述的算法，在某些情形下一个虚拟机迁移的失败可能会导致其后一系列虚拟机的迁移都无法成功，如图所示。

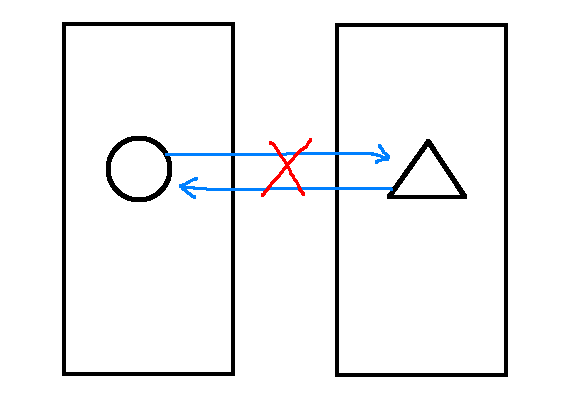
 因此我们设计的算法不对虚拟机迁移的次序做任何假定，计算任何一个虚拟机的目标服务器时，我们都假设其他所有的虚拟机都各就各位没有移动。这种分配目标服务器的方式虽然比较保守，但可以在一定程度上提高虚拟机迁移的成功率，同时也简化了对死锁问题引用第7节的处理。

6 迁移操作的次序

OpenStack异步地处理管理员迁移虚拟机的请求，同时OpenStack也可能同时进行多个虚拟机的迁移操作，所以我们无法对虚拟机迁移操作完成的次序作任何假定。因此我们设计的虚拟机放置策略没有必要输出一个有先后次序的虚拟机迁移操作的序列，而只需要输出一个虚拟机迁移操作的集合引用到【编程实现】部分。

7 死锁问题

服务器资源的有限性使得多个虚拟机迁移的操作彼此之间可能存在冲突，从而造成死锁状态引用死锁示意图。我们采用引用第五节中介绍的算法并不能解决解除死锁，我们的解决方式仅仅是放弃部分迁移操作，让那些处于死锁状态的虚拟机留在原处。我们不采用一些运筹学算法来处理死锁，一方面是为了避免把算法复杂化，另一方面也是为了避免解除死锁不得不引入的额外的迁移操作，从而减小迁移操作带来的开销。解决死锁问题不是本文关注的重点，当然OpenStack管理员可以使用自己设计的从机架中选取服务器的方式，并在其中加入解决死锁问题的策略。

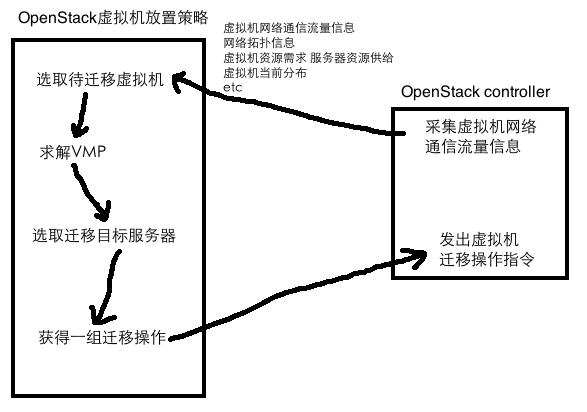


8 控制算法的稳定性

在个链路上网络流量分布已较均匀（网络拥塞已消除）的情况下，调度算法仍有可能做出进行一系列迁移操作的决策，以进一步降低最繁忙链路上的网络流量。但此时这样的决策对网络流量的优化效果已经非常微弱，意义不大。为了提升调度算法的稳定性，减少不必要的迁移操作，我们在算法中设计一个常数阈值。在将迁移操作集合返回给OpenStack前，调度算法将预计算执行这些迁移操作后新的最繁忙链路上的网络流量值。如果这个值相比于原先的减小比例小于，那么调度算法认为当前各链路上网络流量的分布已相对较平均，计算出的新的虚拟机放置策略对网络流量情况的改观已不显著，因而向OpenStack返回空集，即告知OpenStack无需进行任何迁移操作。相关实验见后文4

【算法工作流程】

整个算法的工作流程如下图所示。



【编程实现】

1 整数线性规划问题的求解

VMP问题参见VMP构造部分是一个混合整数线性规划问题。我们采用IBM ILOG CPLEX Optimization Studio引用官网对其进行求解。CPLEX采用分支定界算法引用分支定界求解混合整数线性规划问题。

2 Python编程实现算法

我们使用CPLEX提供的Python API编写程序求解VMP问题。我们设计的OpenStack虚拟机放置策略涉及到的其他算法均用Python语言实现，共含约500行Python代码引用github链接，以模块的形式组织，并对外提供api。OpenStack 的controller的调度器是用Python语言实现的，因此我们实现的虚拟机放置策略可以方便地供 OpenStack调用。

3 OpenStack网络信息采集及虚拟机迁移

这个第3部分我觉得可以忽略不谈？

【测试实验】

董宇茜脚注：合作研究者在交叉信息研究院系统组实验室的计算集群中进行监控并采集了若干组数据，每组数据包含一段时间内虚拟机的网络通信流量信息，虚拟机的分布格局等一系列信息。具体而言，董宇茜对实验室计算集群中分布在12个机架，165台服务器上的198个虚拟机进行了监控，以10分钟为单位对它们的网络通信流量进行了统计，并计算出了平均每秒钟虚拟机两两之间进行网络通信产生的数据流量。我们以这些数据作为测试实验使用的部分测试样例，在下文中我们用test\_case（是编号）指代这些测试样例。

1 单次调度后对链路上网络流量分布的影响

参数：选取流量最大虚拟机，对最大链路贡献最大虚拟机，对次大链路贡献最大虚拟机（为选取待迁移虚拟机的数目），迁移开销1（极小的正值）

取18

整理成图表（已按照降序对流量排序）

test\_case1

迁移前：

5274450, 4772082, 4429466, 4244057, 3909209, 3392898, 3010892, 2939816, 1652784, 1416975, 883647, 14044

迁移后：

3909209, 3855753, 3655323, 3652603, 3648707, 3444431, 3401221, 3392898, 3010892, 2357152, 1416975, 679264

最大流量下降：25.88%

test\_case2

迁移前：

3493427, 3015687, 3013049, 3012346, 2670350, 2453589, 2171790, 1905528, 964084, 932172, 517475, 13623

迁移后：

2670350, 2526554, 2494460, 2453589, 2367357, 2302618, 2244433, 2171790, 2028420, 1826929, 1657577, 13623

最大流量下降：23.56%

test\_case3

迁移前：

5444968, 4595190, 4277177, 4086988, 3968359, 3192537, 3017675, 2794559, 1395481, 1324714, 771072, 12310

迁移后：

3485152, 3470450, 3385425, 3284467, 3271592, 3192537, 3017675, 2977170, 2794559, 2540666, 2156540, 2085035

最大流量下降：36.00%

单次调度可以有效地减小网络流量最大的链路上的流量值，同时使得网络流量在各个链路上的分布更加趋于平均。

另外，我们提出的OpenStack虚拟机放置策略的性能表现取决于各链路上网络流量的分布情况。网络流量分布越不均匀，最繁忙的链路上网络流量相对越大，我们的调度算法对网络流量的优化效果就越好。特别地，在极端情况下，我们的算法可以将最繁忙链路上的网络流量从任意大的值降低至零。

2 带迁移虚拟机的数目对优化效果的影响

参数：选取流量最大虚拟机，对最大链路贡献最大虚拟机，对次大链路贡献最大虚拟机（为选取待迁移虚拟机的数目），迁移开销1（极小的正值）

整理成图表

test\_case1 链路流量：

883647, 4772082, 3392898, 3909209, 5274450, 4429466, 1652784, 4244057, 2939816, 3010892, 14044, 1416975

max: 5274450 (link 4) second: 4772082 (link 1)

c: 3 migration 2 max: 4550206 link 4

c: 6 migration 3 max: 4429466 link 5

c: 9 migration 3 max: 4429466 link 5

c: 12 migration 3 max: 4429466 link 5

c: 15 migration 6 max: 3909209 link 3

c: 18 migration 7 max: 3909209 link 3

c: 21 migration 8 max: 3655323 link 5

c: 24 migration 9 max: 3655323 link 5

c: 27 migration 10 max: 3655323 link 5

c: 30 migration 15 max: 3655323 link 5

test\_case2 链路流量：

517475, 3013049, 2453589, 3012346, 3493427, 2670350, 964084, 3015687, 1905528, 2171790, 13623, 932172

max: 3493427 (link 4) second: 3015687 (link 7)

c: 3 migration 1 max: 3050245 link 4

c: 6 migration 3 max: 3013049 link 1

c: 9 migration 4 max: 3012346 link 3

c: 12 migration 4 max: 3012346 link 3

c: 15 migration 5 max: 2670350 link 5

c: 18 migration 6 max: 2670350 link 5

c: 21 migration 5 max: 2670350 link 5

c: 24 migration 7 max: 2526554 link 1

c: 27 migration 17 max: 2170246 link 5

c: 30 migration 16 max: 2170246 link 5

test\_case3 链路流量：

771072, 4595190, 3192537, 3968359, 5444968, 4086988, 1395481, 4277177, 2794559, 3017675, 12310, 1324714

max: 5444968 (link 4) second: 4595190 (link 1)

c: 3 migration 1 max: 4767217 link 4

c: 6 migration 3 max: 4277177 link 7

c: 9 migration 5 max: 4086988 link 5

c: 12 migration 5 max: 4086988 link 5

c: 15 migration 7 max: 3968359 link 3

c: 18 migration 8 max: 3485152 link 7

c: 21 migration 8 max: 3485152 link 7

c: 24 migration 10 max: 3385425 link 5

c: 27 migration 15 max: 3385425 link 5

c: 30 migration 10 max: 3385425 link 5

随着待迁移虚拟机数目的增加，VMP求解得到的需要实际做迁移的虚拟机数目一般也随之增加，同时优化的结果也不断变得更好（网络流量最大的链路上的流量不断减小）。但其副作用是算法的时间开销不断增加参见实验6。

3 虚拟机迁移开销系数及固定虚拟机对迁移决策的影响

以test\_case3为例 c=18

771072, 4595190, 3192537, 3968359, 5444968, 4086988, 1395481, 4277177, 2794559, 3017675, 12310, 1324714

max: 5444968 (link 4) second: 4595190 (link 1)

cost: 0 migration: 17 max: 3485152 ratio: 36.00%

cost: 1 migration: 8 max: 3485152 ratio: 36.00%

cost: 150000 migration: 8 max: 3485152 ratio: 36.00%

cost: 160000 migration: 4 max: 4091384 ratio: 24.86%

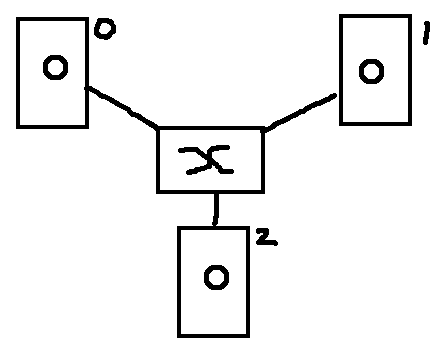
cost: 200000 migraiton: 3 max: 4277177 ratio: 21.45%

cost: 300000 migration: 1 max: 4767217 ratio: 12.45%

cost: 700000 migration: 0 max: 4595190 raito: 0.00%

同步等量地提高所有虚拟机的迁移开销系数将使得每次迁移操作的代价变得更加高昂，从而算法决定进行迁移的虚拟机的数目也随之减小，因此对算法对网络流量的效果也随之变差。迁移开销系数由0变为1时，最繁忙链路的网络流量的减小量并未改变，但虚拟机迁移的次数却大量减少。这是因为，仅考虑链路上网络流量的优化时，最优的解并不唯一，而迁移开销系数大于0将使得不能直接降低最大网络流量的迁移都不会发生。因此即使计算集群上OpenStack采取在线迁移方式引用到前文在线迁移，我们也倾向于将默认的迁移开销系数设成正值而非0，从而减少无谓的迁移操作次数。

为了展示虚拟机的迁移开销系数对迁移行为的影响，我们构造小型实验如下：

3个机架，，上各有一台服务器，每台服务器上运行一个虚拟机，虚拟机，，两两之间的网络通信流量均相等。引用图

若三台虚拟机的迁移开销系数相等（为0或较小的正数），调度算法做出的决策是将和迁移到上。接着我们将，，的迁移开销系数设成，，使得且足够小，调度算法做出的决策是将和迁移到上。由此可见为不同的虚拟机设置不同大小的迁移开销系数将影响调度算法所做出的迁移决策：其他条件一定，且不影响网络流量优化结果的前提下，调度算法将优先考虑迁移开销系数小的虚拟机。

固定一些虚拟机不能移动可以更加直接地影响虚拟机迁移的决策。例如在上例中，保持且足够小的条件不变，将设置为固定虚拟机，调度算法的决策是将和迁移到上。

4 算法的收敛性和稳定性

r = 0.05

test\_case1

migration 7 max: 3909209 link 3

migration 4 max: 3648707 link 8

no migrations 2 rounds

test\_case2

migration 6 max: 2670350 link 5

migration 2 max: 2494460 link 3

no migrations 2 rounds

test\_case3

migration 8 max: 3485152 link 7

no migrations 1 round

r = 0.02

test\_case1

migration 7 max: 3909209 link 3

migration 4 max: 3648707 link 8

migration 2 max: 3524169 link 5

migration 4 max: 3392898 link 2

migration 14 max: 3285981 link 1

最后一次超时 4 rounds

test\_case2

migration 6 max: 2670350 link 5

migration 2 max: 2494460 link 3

no migrations 2 rounds

test\_case3

migration 8 max: 3485152 link 7

migration 2 max: 3385425 link 5

migration 3 max: 3282348 link 4

migration 7 max: 3192537 link 2

no migrations 4 rounds

在本组实验中我们重复运行调度程序，应用每次获得的虚拟机迁移操作改变虚拟机分布的格局，并作为下一次程序的输入（其他输入保持不变）。由实验测试可知调度算法能较快地收敛到稳定的虚拟机放置策略上。阈值的设定很好地避免了虚拟机放置策略的解的震荡，提升了算法的稳定性。调高的值能够加快调度算法的收敛。

6 算法的时间开销

Mac OS X

2.6 GHz Intel Core i5

8 GB 1600 MHz DDR3

c = 18

on average:

test\_case1: 1.188136

test\_case2: 2.139422

test\_case3: 1.512524

c = 12

on average:

test\_case1: 0.413116

test\_case2: 0.431385

test\_case3: 0.434207

c = 6

on average:

test\_case1: 0.151131

test\_case2: 0.152533

test\_case3: 0.178281

随机生成流量矩阵 c = 18

机架数，虚拟机数： 12, 360 12, 720 24, 720

时间： 0.8058463 1.732440 3.787211

在程序中可以指定CPLEX求解VMP问题的时间上限。超过时间上限CPLEX将自动中止VMP的求解并给出当前计算出的最优解。

【存在的问题】

1 本文根据OpenStack虚拟机放置问题的需求构建的数学模型(VMP问题)只支持简单的高度为2的树形结构的网络拓扑结构，即机架内所有服务器连接至一台交换机，所有机架上的交换机连接至一台核心交换机。因此我们设计的OpenStack虚拟机放置策略无法适应一般化的、复杂的网络拓扑结构。

2 VMP问题求解得到的迁移策略在理论上不一定能全部实现。因为VMP问题中所考虑的可用资源是整个机架中各个服务器的可用资源相加之和，而实际可用的资源可能碎片化地分散存在于各个服务器中。因此可能会出现虚拟机迁移的目标机架上单独每台服务器的可用资源都无法满足虚拟机需求的情况。

3 VMP问题求解得到的迁移策略即使不涉及资源碎片化的问题，也有可能无法实现。当虚拟机数目较多，占据服务器的可用资源较满的情况下，可能会有死锁状态出现。我们设计的调度算法在出现死锁问题时将会放弃对虚拟机的迁移操作。引用到前文死锁问题处这一点和上一点可能会导致实际进行虚拟机迁移操作的结果和理论计算出的最优结果不一致，从而对网络流量优化的效果产生一定的负面影响。