



研究与开发

一种数据中心网络虚拟机快速在线迁移算法

马腾, 胡宇翔

(国家数字交换系统工程技术研究中心, 河南 郑州 450002)

摘要: 数据中心网络中, 虚拟机在线迁移需要在网络核心链路上完成大量的数据传输, 造成虚拟机承载的网络应用及其他应用性能下降。在继承现有相同内存页重传避免方法的基础上, 引进带链表的计数型布隆过滤器查找结构, 避免了内存页查找的假阳性问题。进一步提出了最大化剪枝算法, 实现链表长度的最大化缩减, 加速查找匹配过程, 完成数据中心网络中机架级的虚拟机快速在线迁移。实验结果表明, 该方法比现有方法的数据传输量更低, 迁移时间更短, 降低了迁移对网络应用性能造成的影响。

关键词: 虚拟机迁移; 数据中心网络; 布隆过滤器; 剪枝算法

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2017129

Fast and live virtual machine migration in data center networks

MA Teng, HU Yuxiang

National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China

Abstract: In data center networks, virtual machine live migration causes large volume of data transfer on core links of the network, which leads to performance degradation of the baring application and other background services. Based on inheriting the existing memory page de-duplication technique, a new search structure was presented composed of a counting Bloom filter with each bucket pointing to a link list, which overcame the problem of false positive of memory page lookup. Furthermore, a pruned algorithm was proposed to reduce the length of each link list and accelerate the lookup as much as possible to accomplish live and fast virtual machine migration in rack level. Experiment results show that the proposed method transfers less data on core links, takes less time to finish migration, and decreases the degradation of network applications caused by migration.

Key words: virtual machine migration, data center network, Bloom filter, pruned algorithm

1 引言

虚拟机 (virtual machine, VM) 在线迁移是

云数据中心网络虚拟化的关键技术之一^[1-3], 其重要的应用价值在于实现云数据中心网络的负载均衡、电源管理、设备维护等功能。作为数据密集

收稿日期: 2017-03-15; 修回日期: 2017-05-10

基金项目: 国家自然科学基金资助项目 (No.61521003, No.61572519, No.61502530, No.61672471); 国家重点基础研究发展计划 (“973”计划) 基金资助项目 (No.2013CB329104); 国家高技术研究发展计划 (“863”项目) 基金资助项目 (No.2015AA016102)

Foundation Items: The National Natural Science Foundation of China (No.61521003, No.61572519, No.61502530, No.61672471), The National Basic Research Program of China (973 Program) (No.2013CB329104), The National High-Tech Research & Development Program of China (863 Program) (No.2015AA016102)

型业务, 迁移过程中产生的数据量可达 TB 级别^[4], 一般假设非运行时数据存储于专用的网络存储设备, 不参与迁移。大量数据迁移严重消耗了网络核心链路带宽, 导致共享核心链路带宽的其他应用性能下降; 同时带来迁移时间的延长, 损害了迁移虚拟机本身承载的网络应用。网络管理人员迫切希望减少在线迁移的数据量, 减少迁移时间, 保证网络应用的性能。

目前虚拟机在线迁移研究工作可分为 3 类: 预复制、后复制、混合复制^[4]。这些算法在探索高效、灵活利用网络带宽的同时, 通过内存页压缩或相同内存页重传避免方法减少迁移数据量。现有算法的不足之处在于识别相同内存页时, 采用了通用的散列表结构, 每个散列桶均指向一个很大长度的链表, 在链表中执行查找匹配操作时速率低下, 在数据中心网络这种高带宽、低时延应用场景中, 影响了迁移效率^[5]。

本文引入计数型布隆过滤器 (counting Bloom filter)^[6]的一种变体查找结构代替散列表, 力图提高查找速率。通过 SHA1^[7]算法获得内存页信息指纹后, 以信息指纹为基本查找元素, 建立计数型布隆过滤器^[8]查找结构。然后针对计数型布隆过滤器的各计数器 (类似于散列桶), 分别建立对应的信息指纹链表, 查找时进行 2 次匹配。这样处理有效克服了经典布隆过滤器的假阳性 (false positive) 问题, 但由于内存页的信息指纹需要在多个链表中存储, 冗余信息指纹的大量存在使得链表长度急剧增加, 查找匹配速率大大降低, 本文通过设计最大化剪枝算法, 去除链表中不必要的信息指纹, 提高查找匹配速率, 使之适用于云数据中心网络的大规模虚拟机在线迁移场景。

2 相关工作

虚拟机迁移的主要目的在于改变虚拟机在网络中的位置, 以实现优化的效果, 这里的优化包含的内容较为广泛, 如物理资源整合、功耗降低、

设备维护等。本文不对这些常见应用做具体介绍, 重点关注虚拟机迁移过程中的内存页重传避免机制和迁移优化算法。

重复页检测方法被广泛应用于虚拟机迁移^[9,14-18], 这些方法有效降低了一个或多个虚拟机在迁移过程中涉及的数据量。参考文献[14]利用散列算法检测虚拟机镜像中已经发送到目的端的数据块, 加快了单个虚拟机的非在线迁移过程。VMFlock^[15]则通过检测多个虚拟机的相同数据块, 实现了高带宽、高时延的广域网中虚拟机非在线集群迁移的加速。CloudNet^[16]基于内存页脏化速率和传输速率, 在合适时机开启宕机直传, 有效降低了预复制阶段的迭代次数, 从而实现了单个虚拟机广域网上在线迁移的优化。参考文献[17]进一步在内存页传输时采用页级别重复内容检测技术, 仅传输内存页脏化过程中的差异内容。而参考文献[18]通过自适应的内存页压缩技术优化单个虚拟机的迁移。GMGD^[9]是唯一适用于云数据中心网络大规模虚拟机在线迁移方法。它识别和追踪机架内多个待迁移虚拟机的相同内存页, 当多个虚拟机迁移时, 只传输相同内存页的一个副本, 在目的机架内进行内存页复制和插入, 完成虚拟机的迁移, 大大降低了传送的数据量。

3 系统架构

一个典型的由多个机架、主机、交换机组成的数据中心网络如图 1 所示。机架内的主机通过架顶交换机 (top of rack, ToR) 连接, 架顶交换机通过一个或多个核心交换机连接, 链路带宽在 10 Gbit/s 以上。机架之间的虚拟机迁移在连接核心交换机的链路上造成了带宽开销, 为降低带宽开销, 本文提出一种新的虚拟机快速迁移算法, 其核心结构是基于带剪枝链表的计数型布隆过滤器 (pruned counting Bloom filter for VM migration, PCBF-VMM)。算法引入机架级重复消

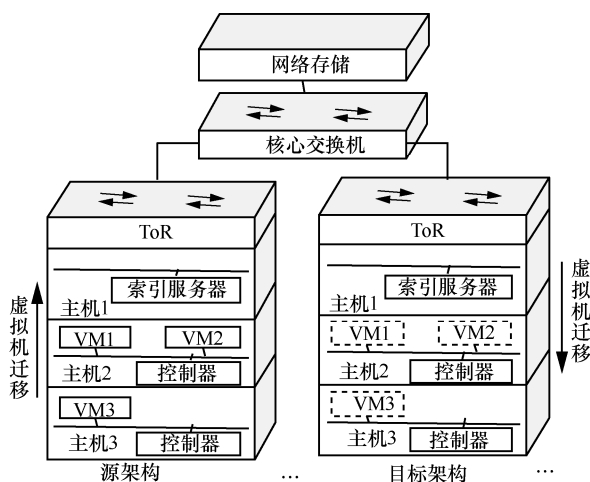


图1 机架级虚拟机迁移系统架构

除机制，识别并跟踪多虚拟机在不同机架主机间迁移时出现的相同内存页。在图1中，PCBF-VMM检测前往相同目的机架虚拟机的重复内存页，只发送一份该内存页的副本到目的机架。在目的机架处，第一个接收到该重复内存页的主机将其转发到其他需要该页的主机。本文基于预复制机制在QEMU/KVM^[10]平台上实现了的PCBF-VMM原型系统，事实上，其他在线迁移机制也可以从中获益，比如后复制。PCBF-VMM原型系统的两个关键模块是索引服务器和分布式控制器。在机架内的某台主机上部署索引服务器，主要负责维护设计的布隆过滤器变体和迁移时的重复内存页查找匹配；在各主机处分别部署控制器（进程），负责协同处理该主机承载的所有虚拟机的重复页检测、跟踪。

PCBF-VMM将迁移过程分为两个阶段。

(1) 重复页检测

主机控制器进程引导驻存其上的虚拟机对应的QEMU/KVM进程扫描虚拟机内存镜像，采用标准的SHA1算法获得内存页的160 byte信息指纹，若存在相同信息指纹，则将该指纹存入控制器中的散列表，并上报索引服务器。

(2) 在线迁移

在该阶段，多个虚拟机并行迁移到各自目的

主机。迁移前，虚拟机对应的QEMU/KVM进程查询索引服务器，以获取内存页状态。若其状态为已发送，则本次迁移只将该内存页的标识和位置发往目的机架，在目的机架处将预先复制的相同内存页转发到目的主机；若其状态为未发送，则需要发送完整的内存页和位置信息到目的机架。

需要注意的是：各主机处的控制器维护一张检测重复内存页的散列表即可，不需要将其设计为索引服务器中的布隆过滤器变体，原因在于迁移时虚拟机对应的QEMU/KVM进程仅在索引服务器处进行重复页查找匹配。

4 带剪枝链表的计数型布隆过滤器

4.1 带链表的计数型布隆过滤器

布隆过滤器是一种高效低耗的查找匹配工具，其原理是利用 m 比特向量 v 表示含 n 个元素的集合 $S = \{x_1, x_2, \dots, x_n\}$ 。初始化时将向量 v 的所有比特置0，当插入元素时，用 k 个独立的散列函数 $h_1, h_2, \dots, h_k$ 将向量 v 的下列比特置1： $h_1(x), h_2(x), \dots, h_k(x)$ ，其中 $x \in S$ ， $h_i(x) \in \{1, 2, \dots, m\}$ ($\forall i \in \{1, 2, \dots, m\}$)。当检查某元素 y 是否属于 S 时，只需要检查由 S 得到的向量 v 中下列比特是否被置1： $h_i(y)$ ($\forall i \in \{1, 2, \dots, m\}$)，如果全被置1则匹配成功，否则匹配失败。

直接使用布隆过滤器做内存页重复检测时，具有如下两个弊端。

- 判断一个元素是否属于某个集合时，有可能把不属于这个集合的元素误认为属于该集合，即假阳性。由于假阳性的存在，当使用布隆过滤器做内存页重复检测时，可能导致某些非重复的关键内存页未被传输到目的主机，系统出现致命错误。因此，必须设法消除假阳性。
- 布隆过滤器不支持删除操作。某些先前被认定为重复的内存页，很长时间没有再次

出现,需要进行“删除”,以降低其他内存页检测的复杂度、减少检测时间。

为了支持删除操作,考虑使用布隆过滤器的一种变体——计数型布隆过滤器,它能够很好地支持元素的添加、删除、查找操作,满足本文需求;为了克服假阳性,考虑引入链表^[8],让计数型布隆过滤器的每个计数器指向一个链表,通过在链表中进行二次匹配,避免假阳性的出现。综上所述,本文的查找结构选择带链表的计数型布隆过滤器(counting Bloom filter with links, CBFL)。

4.2 信息指纹的添加、删除、查询

本节将给出带链表的计数型布隆过滤器添加、删除、查询内存页信息指纹的具体算法。作为对比,首先简要叙述 GMGD 中采用的简单散列表(simple Hash table, SHT)的相关处理流程。

SHT 的长度记为 m , 即该 SHT 中包含 m 个散列桶。以 X 表示插入 SHT 中的元素(即信息指纹)集合, X^i 表示第 i 个散列桶指向的链表中的元素集合, X_j^i 表示第 i 个散列桶指向的链表中的第 j 个元素。则式(1)、式(2)成立:

$$X^i = \{X_1^i, X_2^i, X_3^i, \dots, X_{a_i}^i\} \quad (1)$$

$$X = \bigcup_{i=1}^L X^i \quad (2)$$

式(1)、式(2)中, a_i 表示第 i 个散列桶指向的链表的长度, L 为 SHT 包含的链表数量。在图2所示的 SHT 中, $X_1^3 = z$, $X_2^3 = w$, $a_3 = 2$, $L = 3$ 。在 SHT 中, 添加一个元素 x , 可表示为 $X^{h(x)} = X^{h(x)} \cup x$; 类似地, 删除元素 x 表示为 $X^{h(x)} = X^{h(x)} - x$, 而元素查询则为 {if $x \in X^{h(x)}$ return true; else return false;}, 其中 $h()$ 为均匀随机散列函数。

CBFL 添加元素时, 首先需要构建计数型布隆过滤器计数器向量, 初始值为全 0。以 C 表示 m 个计数器组成的阵列, 其中 C_i 表示第 i 个散

列桶对应的计数器。 k 个散列函数依次为 $h_1(), \dots, h_k()$ 。依次添加集合 $X = \{x_1, x_2, \dots, x_n\}$ 中的 n 个元素: 对于每个元素 x , 利用 k 个相互独立的散列函数 $h_1, h_2, \dots, h_k$ 将计数器向量 v 的如下分量加 1: $h_1(x), h_2(x), \dots, h_k(x)$, 其中 $x \in S$, $h_i(x) \in \{1, 2, \dots, m\}$ ($\forall i \in \{1, 2, \dots, m\}$)。但需要注意的是, 同一个元素被多个散列函数映射到同一个计数器对应的链表中时, 该计数器仅执行一次加 1 操作, 且该元素的插入操作也只执行一次, 防止同一个元素多次插入一个链表。CBFL 的元素添加算法如下。

算法 1 CBFL 元素添加算法

```

procedure Insert_Item_CBFL(x)
  for  $i = 1:k$ 
    if ( $h_i(x) \neq h_j(x) \forall j < i$ )
       $C_{h_i(x)} = C_{h_i(x)} + 1$ 
       $X^{h_i(x)} = X^{h_i(x)} \cup x$ 
    end if
  end for
end procedure

```

算法 1 中 if 语句内的判断就是为了处理同一个元素的多个散列函数值相同的情况。图 2 给出了 $n = 4$ 时一个最终结果的实例, 元素 x, y, z 均被某个散列函数映射到了第 3 个计数器, 因此对应的链表包含 3 个元素, 而计数器的数值为 3。

CBFL 的元素查询方法: 给定待查询的元素 x ,

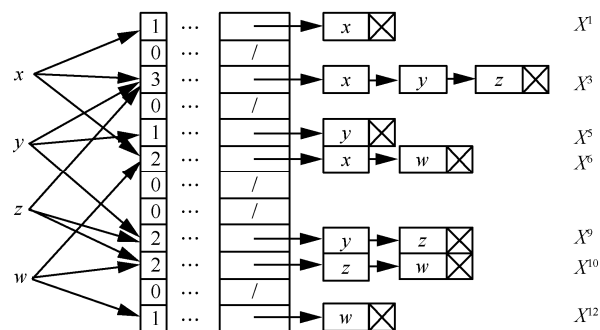


图2 添加 x, y, z, w 后的 CBFL



计算 k 个散列函数的值, 读取相应的计数器。若相应计数器值均非零, 表明该元素极有可能命中, 需要进一步查询这些计数器对应的链表。事实上, 只需要查询一个链表就可以判断命中与否。为加快查询匹配的速度, 选取其中最短的链表进行查询, 该计数器值最小。CBFL 元素查询算法如下。

算法 2 CBFL 元素查询算法

procedure Search_Item_CBFL(x)

$C_{\min} = \min\{C_{h_1(x)}, \dots, C_{h_k(x)}\}$

if ($C_{\min} == 0$)

return false

else

$i = \text{SmallestIndexof}(C_{\min})$

if ($x \in X^i$)

return true

else

return false

end if

end if

end procedure

需要注意的是, 查询时若读取到多个非零的相同的最小计数器值, 则选取索引号最小的计数器对应的链表进行查询。

CBFL 元素删除方法与添加方法类似, 见算法 3, 其中 if 语句内的判断就是为了处理同一个元素的多个散列函数值相同的情况。

算法 3 CBFL 元素删除算法

procedure Prune_Set_CBFL(x)

for $i = 1 : k$

if ($h_i(x) \neq h_j(x) \forall j < i$)

$C_{h_i(x)} = C_{h_i(x)} - 1$

$X^{h_i(x)} = X^{h_i(x)} - x$

end if

end for

end procedure

4.3 最大化剪枝

添加了元素链表的计数型布隆过滤器能够完全避免假阳性的发生。但是该结构存在着链表长度大、查找匹配速率低的问题, 原因在于使用了 k 个散列函数, 使得每个元素会出现在多个链表中。事实上, 既然二次匹配时只需要对一个非零的计数器对应的链表内容进行查询就能确定结果, 故可以利用剪枝的方法, 将各链表中冗余的元素副本删掉。

本文引入的最大化剪枝算法, 将最大化降低链表长度, 使得每个元素仅在一个链表中存储, 大大提高了查找匹配过程, 使之适应机架级的大规模内存页快速迁移场景。在 CBFL 中, 具体的剪枝按以下步骤操作。

步骤 1 对于集合 X 中的每一个元素 x , 依次计算散列值 $h_1(x), h_2(x), \dots, h_k(x)$, 获取对应位置处的计数器数值 $C_{h_1(x)}, \dots, C_{h_k(x)}$, 记最小的计数器数值为 $C_{\min}$, 转至步骤 2。

步骤 2 获取所有计数值为 $C_{\min}$ 的计数器索引, 记最小的索引号为 i , 转至步骤 3。

步骤 3 对于所有 k 个元素 x 被映射到的链表, 若对应的计数器索引号不是 i , 则剪除该链表中的元素 x , 直至 k 个链表中仅有索引号为 i 的计数器对应的链表中存储有元素 x , 转至步骤 4。

步骤 4 重复执行步骤 1 的工作, 直至所有元素的重复副本剪除完毕。

最终的结果是, 所有元素在带链表的计数型布隆过滤器中仅有一个副本, 比剪枝前的链表平均长度降低了 k 倍。CBFL 最大剪枝算法的伪码见算法 4。

算法 4 CBFL 最大剪枝算法

procedure Prune_Set_CBFL(x)

for (each $x \in X$)

$C_{\min} = \min\{C_{h_1(x)}, \dots, C_{h_k(x)}\}$

$i = \text{SmallestIndexof}(C_{\min})$

```

for  $l = 1:k$ 
  if ( $h_l(x) \neq i$ )
     $X^{h_l(x)} = X^{h_l(x)} - x$ 
  end if
end for
end for
end procedure

```

图2中添加 x 、 y 、 z 、 w 后的 CBFL, 经剪枝, 得到的结果如图 3 所示。通过图 3 可以看出, 剪枝后各链表的长度大大降低, 每个元素仅在一个链表出现, 且只出现一次。经过剪枝, 内存页信息指纹查询速度大幅提高, 也将降低虚拟机迁移时间。需要注意的是, 剪枝算法不会将计数器置 0, 因此, 不会影响查询时对计数器向量的初始判断。

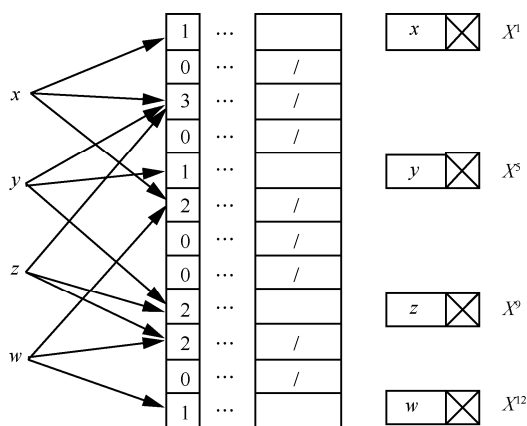


图3 添加元素 x 、 y 、 z 、 w 并剪枝后的 CBFL

5 性能评估

在 QEMU/KVM 虚拟化平台实现了 PCBF-VMM 原型系统。实验平台为 2 个机架连接而成的简单数据中心网络, 每个机架包含 7 台物理服务器, 其中一台为索引服务器, 其他 6 台为实验用虚拟机宿主, 分为源主机和目的主机。每个物理服务器包括 2 个 2 GHz 的双核 CPU、16 GB 内存、一个吉比特网卡。图 4 展示了虚拟机迁移实验用到的拓扑。这里源主机集群和目的主机集

群分别连接两台以太网交换机, 作为架顶交换机使用, 核心链路带宽为 10 Gbit/s。实验时, 源主机上的所有虚拟机同时启动, 虚拟机内存页从源主机通过 ToR 交换机之间的 10 Gbit/s 核心链路传输到目的主机。实验中, 每台物理主机上驻存 4 台虚拟机, 每台虚拟机配置为 2 个虚拟 CPU、1 GB 内存。

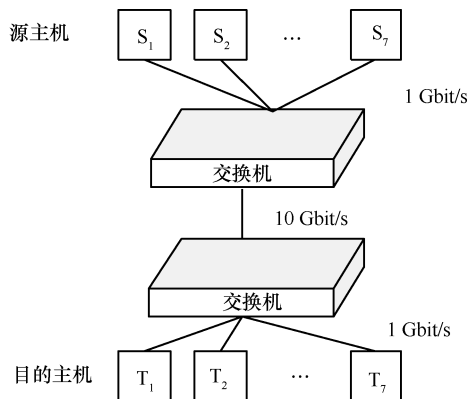


图4 虚拟机迁移实验拓扑结构

实验对比的算法包括: 基线算法, 即将全零的内存页压缩传输; GMGD 算法, 核心查找结构为带链表的简单散列表; PCBF-VMM, 即本文提出的虚拟机迁移算法, 核心查找结构为带链表的计数型布隆过滤器。

5.1 数据传输量对比

5.1.1 虚拟机空闲时的情况

图 5 (a) 给出了当虚拟机均为空闲 (idle) 状态时, 随着迁移虚拟机数量的增加, 迁移时各种算法核心链路传输的数据量, 此处每个物理服务器上运行 4 个 1 GB 的虚拟机。由于基线算法仅仅是将数据为全零的内存页压缩传输, 大量的相同内存页 (非零内存页) 均全部原样传输, 故传输的数据量在各算法中最大。因为迁移虚拟机处于空闲状态, 因此 GMGD 和 PCBF-VMM 算法的数据传输量几乎相等, 微量差异出现的原因在于 PCBF-VMM 的重复内存页查询速度快于 GMGD, 由此带来传输速率的提升, 空闲状态下由于传输时间的差异导致

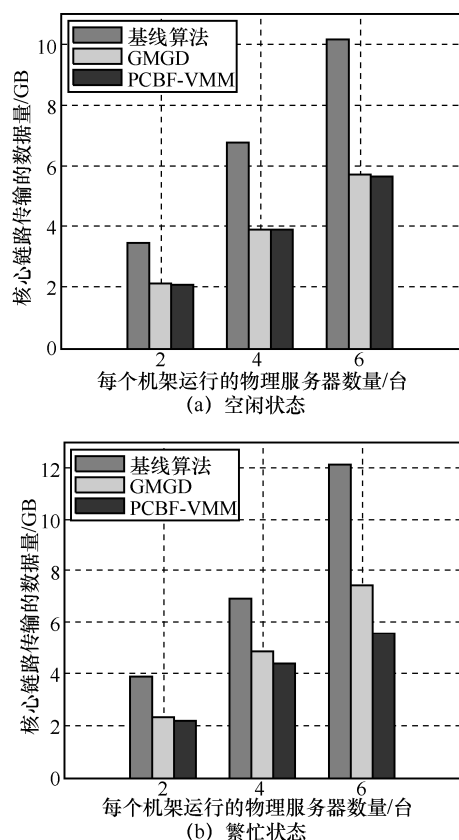


图5 各算法核心链路传输的数据量对比

的内存页出现微量的更新。

5.1.2 虚拟机繁忙时的情况

为考察虚拟机繁忙 (busy) 状态下核心链路传输的数据量, 在每个迁移的虚拟机上均运行 Dbench^[11] 文件系统。Dbench 会产生大量的动态数据, 为各算法的数据传输量对比提供了良好的条件。图 5 (b) 给出了当虚拟机均为繁忙状态时, 随着迁移虚拟机数量的增加, 迁移时各种算法核心链路传输的数据量。可以看出, 运行的物理服务器较少时, PCBF-VMM 和 GMD 的数据传输量相当; 随着服务器数量的增加, PCBF-VMM 传输的数据量少于 GMD, 且服务器数量愈多优势愈加明显。迁移的虚拟机数量为 $6 \times 4 = 24$ 台时, 迁移数据量较基线算法减少 52%, 较 GMD 算法减少 28%。原因在于快速的相同内存页查询算法保障了迁移吞吐率的增大, 脏化的内存页的数量减少。

5.2 总迁移时间对比

为测量各种算法的总迁移时间, 采用计算每个虚拟机迁移时间平均值的方法。在数据中心网络中, 在虚拟机迁移的路径上, 往往存在其他网络流量。在负载较重的交换机中, 虚拟机迁移流量将影响到这些流量的传输性能。实验中为了尽可能模拟真实数据中心网络的流量模式, 在两个交换机之间的核心链路上, 除了虚拟机迁移的流量, 同时加上了 8 Gbit/s 的背景流量。背景流量是由 Netperf^[12] 产生的 UDP 数据分组, 这些数据分组由每一个源端主机发往目的端主机, 将与虚拟机迁移流量竞争核心链路带宽。实验同时对比了有背景流量和无背景流量条件下, 虚拟机迁移总时间随物理服务器开启数量的变化情况。

各算法的虚拟机迁移总时间结果如图 6 所示。从图 6 (a) 可以看出, 无背景流量的状态 (虚拟机处于空闲状态) 下, 各算法的总迁移时间不因参与迁移的虚拟机数量而发生剧烈变化, 原因在于无背景流量时网络的核心链路带宽充裕, 满足所有虚拟机迁移的需求, 且大量带宽资源处于闲置状态。从图 6 (b) 可以看出, 有背景流量的状态下, 各算法的总迁移时间均高于无背景流量的情况。而 PCBF-VMM 的总迁移时间小于基线算法和 GMD 算法, 原因在于: 在背景流量大量占据核心链路带宽的情况下, 迁移的数据量是影响总迁移时间的一个重要因素, 而 PCBF-VMM 的迁移数据量较其他算法少; 快速的重复内存页查找算法保证了 PCBF-VMM 对核心链路剩余带宽的较高利用率。

同时, 实验中对比了 Netperf UDP 流量的分组丢失情况, 结果如图 7 所示, 可以看出, 当网络中有 $2 \times 4 = 8$ 台虚拟机同时迁移时, 核心链路带宽充足, Netperf 数据分组没有发生丢失的情况。随着迁移虚拟机数量的增加, 开始发生分组丢失。当参与迁移的虚拟机达到 $2 \times 4 = 8$ 台时, PCBF-VMM 的分组接收率较 GMD 高 10%, 较基线算法高 25%。

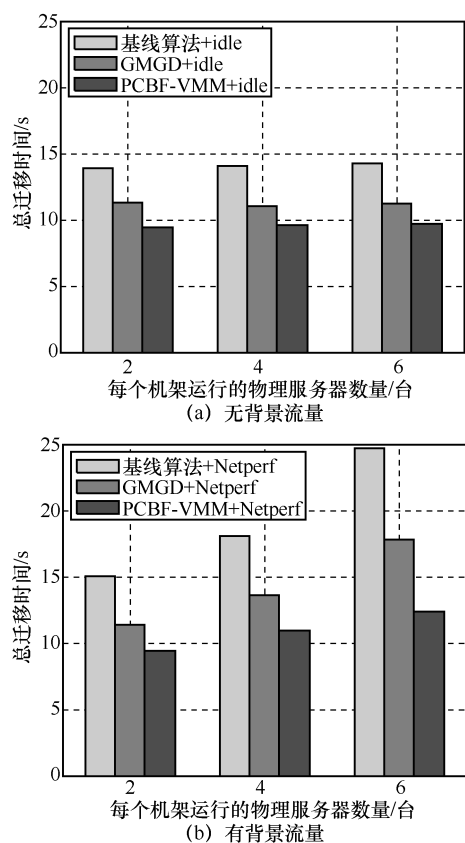


图6 虚拟机总迁移时间对比

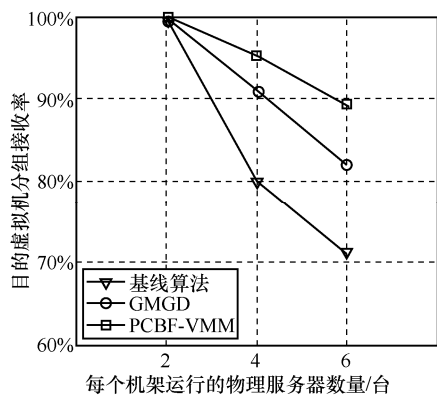


图7 各算法 Netperf 流量分组接收率

6 结束语

本文针对 DCN 中虚拟机在线迁移造成网络核心链路大量数据传输, 导致网络应用性能下降的问题, 在继承现有的相同内存页重传避免方法的基础上, 引进一种新的查找结构—剪枝的带链表的计数型布隆过滤器, 加快了相同内存页的查找速率, 使得虚拟机在快速迁移的

同时, 脏化的内存页数量减少, 从而降低了核心链路传输的迁移数据量; 同时, 快速在线迁移降低了对虚拟机承载的应用的性能影响。在小型数据中心网络拓扑上的实验结果证实了上述结论。

参考文献:

- [1] BLOOM B H. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426.
- [2] 赵辉, 丁鸣, 程青松, 等. SDN 与 NFV 技术在云数据中心的规模应用[J]. 电信科学, 2016, 32(1): 144-151.
ZHAO H, DING M, CHENG Q S, et al. Application of SDN and NFV technology in the cloud data center[J]. Telecommunications Science, 2016, 32(1): 144-151.
- [3] 顾戎, 王瑞雪, 李晨, 等. 云数据中心 SDN/NFV 组网方案、测试及问题分析[J]. 电信科学, 2016, 32(1): 126-130.
GU R, WANG R X, LI C, et al. Analysis on network scheme and resolution test of SDN/NFV technology co-deployed in cloud datacenter[J]. Telecommunications Science, 2016, 32(1): 126-130.
- [4] AHMAD R W, GANI A, HAMID S H A, et al. A survey on virtual machine migration and server consolidation frameworks for cloud data centers[J]. Journal of Network & Computer Applications, 2015, 52(C): 11-25.
- [5] DESHPANDE U, KULKARNI U, GOPALAN K. Inter-rack live migration of multiple virtual machines[C]//The 6th International Workshop on Virtualization Technologies in Distributed Computing Date, June 18, 2012, Delft, Netherlands. New York: ACM Press, 2012: 19-26.
- [6] FAN L, CAO P, ALMEIDA J, et al. Summary cache: a scalable wide-area web cache sharing protocol[J]. IEEE/ACM Transactions on Networking (TON), 2000, 8(3): 281-293.
- [7] OpenSSL SHA1 hash. [2017-02-11]. <https://www.openssl.org/docs/manmaster/man3/SHA1.html>.
- [8] SONG H, DHARMAPURIKAR S, TURNER J, et al. Fast hash table lookup using extended bloom filter: an aid to network processing[J]. ACM SIGCOMM, 2005, 35(4): 181-192.
- [9] DESHPANDE U, SCHLINKER B, ADLER E, et al. Gang migration of virtual machines using cluster-wide deduplication[C]//2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 13-16, 2013, Delft, Netherlands. New Jersey: IEEE Press, 2013: 394-401.
- [10] KIVITY A, KAMAY Y, LAOR D, et al. KVM: the Linux virtual machine monitor[C]//The 2007 Ottawa Linux Symposium, June 27-30, 2007, Ottawa, Canada. [S.l.: s.n.], 2007: 225-230.
- [11] A file system benchmark named Dbench [Z/OL]. [2017-02-15]. <http://samba.org/ftp/tridge/dbench>.



- [12] A network performance benchmark named Netperf [Z/OL]. [2017-02-15]. <http://www.netperf.org/netperf>.
- [13] 李丹, 刘方明, 郭得科, 等. 软件定义的云数据中心网络基础理论与关键技术[J]. 电信科学, 2014, 30(6): 48-59.
LI D, LIU F M, GUO D K, et al. Fundamental theory and key technology of software defined cloud data center network[J]. Telecommunications Science, 2014, 30(6): 48-59.
- [14] SAPUNTZAKIS C, CHANDRA R, PFAFF B, et al. Optimizing the migration of virtual computers[C]//Operating Systems Design and Implementation, December 1, 2002, Boston, USA. New York: ACM Press, 2002: 307-399.
- [15] KISWAMY S, SUBHRAVETI D, SARKAR P, et al. VMflock: virtual machine co-migration for the cloud[C]//High Performance Distributed Computing, June 8-11, 2011, San Jose, California, USA. New York: ACM Press, 2011: 159-170.
- [16] WOOD T, RAMAKRISHNAN K K, SHENOY P, et al. Cloud-Net: dynamic pooling of cloud resources by live wan migration of virtual machines[J]. IEEE/ACM Transactions on Networking, 2015, 23(5): 1568-1583.
- [17] ZHANG X, HUO Z, MA J, et al. Exploiting data deduplication to accelerate live virtual machine migration[C]//International Conference on Cluster Computing, September 2010, Spet 20-24,

2010, Heraklion, Greece. New Jersey: IEEE Press, 2010: 88-96.

- [18] JIN H, DENG L, WU S, et al. Live virtual machine migration with adaptive, memory compression[C]//2009 Cluster Computing and Workshops, Aug 31-Sep 4, 2009, New Orleans, USA. New Jersey: IEEE Press, 2009: 1-10.

[作者简介]



马腾(1987-), 男, 国家数字交换系统工程技术研究中心博士生, 主要研究方向为数据中心网络、网络体系结构。



胡宇翔(1982-), 男, 博士, 国家数字交换系统工程技术研究中心副研究员, 主要研究方向为网络体系结构。