第 33 卷 第 23 期 计 算 机 工 程 2007 年 12 月
Vol.33 No.23 Computer Engineering December 2007
·网络与通信· 文章编号：1000—3428(2007)23—0105—03 文献标识码：A 中图分类号：TP301.6

多 Markov 链预取模型在网络内存系统中的应用

袁清波，孙国忠，陈明宇
（中国科学院计算技术研究所计算机系统结构重点实验室，北京 100080）

摘 要：主机通过高速网络访问远程内存的性能已经达到或远高于访问本地磁盘的性能，通过各种优化手段，网络内存系统的性能能得到更好的提升。该文基于一个 Linux 网络内存系统(LNMS)，在客户端一级提出了一种新的预取算法 m-ppm，该算法发展了多 Markov 链预取模型，使之更适合 LNMS。在 LNMS 上实现了另 2 种常用的预取算法以作比较，实验数据表明，m-ppm 算法对多用户模式更有效。

关键词：网络内存系统；多 Markov 链预取模型；性能优化

Applicedion of Multi-Markov Chains Prefetching Model in Network Memory System

YUAN Qing-bo, SUN Guo-zhong, CHEN Ming-yu
（Key Lab of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080）

【Abstract】 Good performance can be archived when fetching data from remote memory through high-speed network than that from local disk, and performance of network memory system can be improved by kinds of optimization. Based on a Linux network memory system(LNMS), a new prefetching algorithm——m-ppm is proposed to improve the performance in client part. The algorithm extends multi-Markov chains prefetching model to make it more suitable for LNMS. Other two common prefetching algorithms are implemented in LNMS for comparison. Experimental results show that m-ppm method is more effective, especially in multi-user mode.

【Key words】 network memory system; multi-Markov chains prefetching model; performance optimization

分布式网络环境下，基于Linux平台构建的系统得到了广泛的应用。由于单个节点内存资源有限，因此出现了许多利用其他节点的内存来扩展本地内存以提高系统性能的研究：SAMSON系统[1]提供给客户端的基本抽象是一个网络内存虚拟单元，其实质是一个虚拟内存页面数组，可通过Unix块设备驱动接口访问；HPBD[2]在Infiniband机群系统上构建，作为核心虚拟存储层的交换设备，负责和远端内存交互，实现页面换入和换出功能；LNMS(Linux network memory system)[3]基于Linux环境构建，采用两级服务结构，具有高扩展性和灵活性。这些传统的网络内存系统用于代替慢速磁盘，因此，访问远程内存的延迟问题并不突出，对其延迟的改进往往被忽视。但采取有效的预取手段降低延迟，通常可以不同程度地提升系统的整体性能。

1 LNMS

LNMS 是一种 2 级服务结构，基于 Linux 环境构建。如图 1 所示。

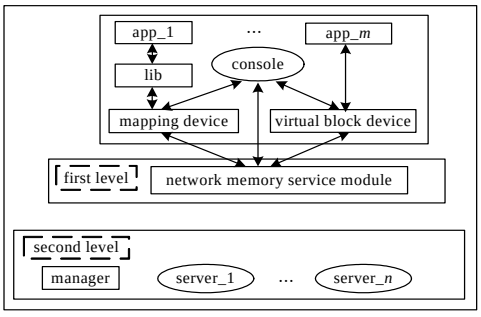

图 1 LNMS 体系结构

LNMS 第 1 级服务通过网络内存服务模块(network memory service module)进行抽象，向多个上层模块提供服务；第 2 级服务由多个内存服务器和一个管理服务器组成，向客户端提供服务。现有的系统中，上层模块包括 VBD 模块(virtual block device)、网络内存映射模块(mapping device)、lib 库模块。

2 预取技术

从图 1 可以看到，整个系统的访问延迟瓶颈在客户端与服务器的数据传输部分，因此，如何减少这部分的访问时间是本文的重点。解决方法是预测用户的访问序列，将用户需要的数据提前从服务器取回本地供将来使用。

2.1 多 Markov 链预取模型

多 Markov 链预取模型从单 Markov 链预取模型发展而来。单 Markov 链预取模型适用于单用户模式下的访问，在多用户模式下，其性能降低比较明显。于是，邢永康等人提出了多 Markov 链预取模型。多 Markov 链模型的主要特点是将用户分类，同一类别的用户具有相同或相近的特征，而不同类别的用户特征差异较大，须用不同的 Markov 链来描述。

定义 1 多 Markov 链模型可以表示为一个四元组：$<X, K, P(C), MC>$。其中，X 是一个离散随机变量，值域为 $\{x_1, x_2, \cdots, x_n\}$，每个 x_i 表示模型的一个状态；K 表示模型包含的用户类别的数目；$C=\{c_1, c_2, \cdots, c_k\}$ 表示用户的类别；其分布函数 $P(C)$ 表示不同类别用户的概率分布；$MC=$

作者简介：袁清波(1982－)，男，博士，主研方向：高性能计算；孙国忠，博士；陈明宇，研究员

收稿日期：2007-01-15 **E-mail：**yuanbor@ncic.ac.cn

$\{mc_1, mc_2, \cdots, mc_k\}$为类 Markov 链的集合，每一个元素 mc_k 是描述类别为 c_k 的用户访问特征的 Markov 链，称为类 Markov 链，它的转移矩阵和初始状态分布分别表示为

$$\boldsymbol{A}_k = (p_{ki\,j}) = \begin{pmatrix} p_{k11} & p_{k12} & \cdots & p_{k1j} & \cdots & p_{k1n} \\ p_{k21} & p_{k22} & \cdots & p_{k2j} & \cdots & p_{k2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ p_{ki1} & p_{ki2} & \cdots & p_{kij} & \cdots & p_{kin} \\ \vdots & \vdots & & \vdots & & \vdots \\ p_{kn1} & p_{kn2} & \cdots & p_{knj} & \cdots & p_{knn} \end{pmatrix}$$

$$\lambda_k = (p_{ki}) = (p_{k1}, p_{k2}, \cdots, p_{kn})$$

假设已知某个用户的浏览序列$(x_1, x_2, \cdots, x_l)$，要对该用户的下一个状态作出预测。由于多 Markov 描述不同类别用户的浏览特征，因此其预测过程分为以下 2 步：

(1)判定用户类别

根据贝叶斯公式，该用户属于类别 c_k 的概率为

$$P(C = c_k \mid x_1, x_2, \cdots, x_l) = \frac{P(x_1, x_2, \cdots, x_l \mid C = c_k) P(C = c_k)}{P(x_1, x_2, \cdots, x_l)}$$

其中，分母表示序列$(x_1, x_2, \cdots, x_l)$的边际概率，对于不同的分类，该值都不会改变。判定规则为

如果

$$P((x_1, x_2, \cdots, x_l) \mid C = c_k) P(C_k) = \max_{j=1,2,\cdots,K} (P((x_1, x_2, \cdots, x_l) \mid C = c_j) P(C = c_j))$$

则用户类别为 c_k。

(2)进行预测

如果用户类别为 c_k，则该用户的访问特征由其类 Markov 链 mc_k 描述，可以采用与单 Markov 链模型相同的方法对用户的访问进行预测：

$$\boldsymbol{V}(t) = \boldsymbol{H}(t-1) \times \boldsymbol{A}_k$$

其中，向量 $\boldsymbol{H}(t-1) = (0, 0, \cdots, 0, 1, 0, \cdots, 0)$ 表示用户在时刻$(t\text{-}1)$的状态，如果此时用户处于状态 x_i，则该向量的第 i 维等于 1，其余各维都为 0；矩阵 $\boldsymbol{A}_k$ 是类 Markov 链 mc_k 的转移矩阵；向量 $\boldsymbol{V}(t) = [P(X_t = x_1), P(X_t = x_2), \cdots, P(X_t = x_n)]$ 表示在时刻 t 系统的状态概率向量，在向量 $\boldsymbol{V}(t)$ 中，概率值最大的那一维所对应的状态就是用户在时刻 t 最可能的状态。

2.2 多 Markov 链预取模型的实现

本文依据多Markov链预取模型提出了多数据流预取算法m-ppm。由多Markov链模型预取步骤可知，m-ppm算法首先识别出数据流的种类，然后对该数据流采用ppm算法[4]进行预取。本文则采用效率更高的last-stride算法代替标准的贝叶斯算法来识别特征流。

2.2.1 多访问流识别算法 last-stride

根据数据访问的空间局部性原理，当前访问数据其附近的数据也将很快被访问，因此，对于多个流组成的访问序列，如果连续 2 个访问地址的差值较小，则可认为属于一个流。

该算法的主要操作是：将当前访问地址和所有已存在流的最后一个地址相减，得到对应第 i 个流的 *stride_i* 值，从所有的 *stride_i* 值中选择最小值 *stride_min*，如果 *stride_min* 值小于等于给定的阈值 *threshold*，则当前访问地址属于 *stride_min* 所对应的访问流 如果 *stride_min* 值大于 *threshold*，则是一个新的流。

2.2.2 m-ppm 算法

m-ppm 首先根据 last-stride 多流识别算法识别出访问流中的多个特征流，然后针对每个流构建相应的 ppm 预取模型。m-ppm 算法如下：

(1)接收访问地址 *addr*。

(2)根据 last-stride 流识别算法定位该地址所属的流，如果该地址不属于任何流，则新建一个访问流。

(3)得到 *addr* 所属的访问流 *stream_I*。

(4)针对 *addr* 所属的访问流采用 ppm 模型进行处理：

1)如果是新的访问流，则对 ppm 模型进行初始化；

2)否则，把 *addr* 地址输入到 ppm 模型中进行处理：如果 ppm 中的地址间隔数达到初始设定的地址间隔数，更新预测表；否则，继续初始化 ppm 模型，记录新的地址间隔以及 *last_addr_i*；

3)如果 ppm 中的地址间隔数达到初始设定值，且预测的块不在 prefetch_buffer 中则进行预取：如果是顺序模式，取多个块；否则只取一个块。然后更新 prefetch_buffer。

4)更新 ppm 模型参数，包括记录新的地址间隔以及 *last_addr_I*。

2.3 其他预取算法

2.3.1 顺序预取(eq)

大多数应用都具有顺序访问的特征，Linux, Unix 等操作系统核心针对文件系统及交换磁盘都采用了顺序预取算法。顺序预取多个块可以提高整个系统的 IO 吞吐量，充分利用带宽优势。

2.3.2 ppm

ppm 是基于 Markov 思想的一种预测算法。它根据前面发生的序列计算下一个会出现的值，把具有最高发生概率的值作为预测值。如给定一个输入的历史访问 ABCABDABC，预测器记录 AB 后跟 C 的概率为 2/3，跟 D 的概率为 1/3，于是在程序依次访问了 A, B 后，预测器就将 C 预取回来。

3 实验及评价

为了更好地对上述 3 种预测算法进行对比，本文进行了 2 种类型的测试：基于 trace 的测试和基于真实应用的测试。

3.1 基于 trace 的测试

这里使用的trace[5]记录了程序的地址访问序列，将这些序列输入到网络内存系统中，系统按正常流程进行读写，默认每次读写大小为 4KB。trace的执行时间如图 2 所示，其中，横坐标表示 4 种trace；纵坐标是各种情况和无预取情况下trace的执行时间之比。

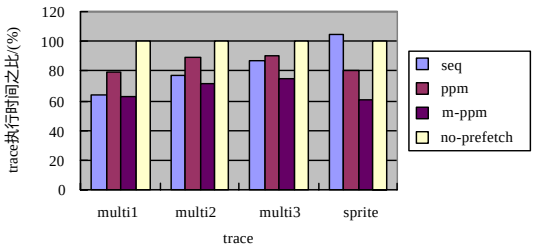

图 2 trace 执行时间

由图 2 可以看出，执行 multi1, multi2 和 multi3 时，3 种预取算法均有不同程度的性能提升，特别是顺序预取算法和 m-ppm 算法，这也说明了 m-ppm 算法完全适应具有顺序访问特征的应用；对于 sprite，顺序预取算法反而降低了系统的性能，这主要是因为算法预取的正确率较低，导致大量预取数据之后未被使用，但 ppm 算法和 m-ppm 算法分别有 20%和 40%的性能提高。综合 4 种 trace 测试结果可以看出：顺序预取算法和 ppm 算法均有局限性，而 m-ppm 算法综合了 2 种算法的长处，应用范围更加广泛。

3.2 基于真实应用的测试

trace 的测试过程是一个模拟读写的过程，为了作进一步的对比，本文测试了快速排序程序在该系统上的运行结果。测试结果如图 3 所示，其中，横坐标表示性能评测的 3 种指标：预取算法的命中率(hit-ratio)，预取块的使用率(use-ratio)，排序的总执行时间相对于无预取的比率(exec-time)；而纵坐标根据不同的性能指标有不同的含义。

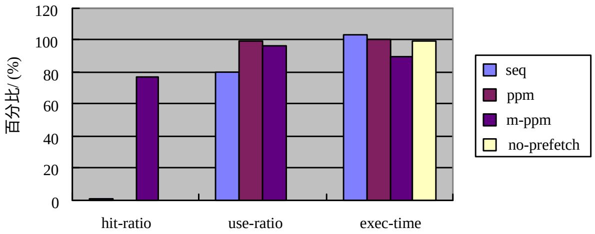

图 3　多线程模式下算法的比较

从图 3 中可以看出，多线程模式下，顺序预取算法(seq)和 ppm 算法的命中率几乎为 0，因为顺序预取算法和 ppm 算法将所有线程的读写作为一个统一的流进行处理，而每个线程的访问特征并不一致，所以整体读写访问特征不明显，这 2 种算法预取的数据并不会被访问到，因此，采用这 2 种算法不仅无法缩短该应用的执行时间，还会因预取的开销导致执行时间变得更长，但 m-ppm 算法在运行过程中会将多个线程的访问序列分开处理，理想情况下可以得到类似单线程的预取效果。实验显示，m-ppm 算法的命中率接近 80%，执行时间比无预取的情况减少了 10%。综上所述，m-ppm 算法比顺序预取算法和 ppm 算法更适合多线程模式。

3.3 实验总结

根据单用户和多用户模式下的实验结果可以看出，不同的应用具有不同的访问特征，某些应用适合用顺序模型来描述，某些应用则适合用 ppm 模型来描述，还有一些应用这 2 种模式均无法描述。因此，对于一个通用的网络内存系统来说，无论用顺序预取模型还是 ppm 模型都不适合。本文基于多 Markov 链预取模型思想的 m-ppm 算法在多种情况下均有很好的适应能力，适合在通用网络内存系统中使用。

4 结束语

本文在一个真实网络内存系统 LNMS 上实现了 3 种预取算法。顺序预取最简单，只要最近一次访问的地址和当前访问的地址连续，就可从当前访问的数据后连续取多个数据块到本地缓冲区中，在理想情况下，这些刚预取的数据块会马上被访问到。ppm 算法基于单 Markov 链预取模型，适合非顺序访问情况，它们通常只在单用户模式下有效。对于多用户模式，则需要使用本文提出的 m-ppm 算法，算法基于多 Markov 链模型，可以分析出多用户模式的访问规律，且算法完全兼容单用户模式，是一种比较通用的预取算法。

参考文献

1 SAMSON Network Memory Server Project[Z]. (2003-08). http://bsd7.starkhome.cs.sunysb.edu/~samson/.

2 Liang S, Noronha R, Panda D K. Swapping to Remote Memory over InfiniBand[C]//Proc. of IEEE International Conference on Cluster Computing. 2005-09.

3 Sun Guozhong, Tang Huan, Chen Mingyu, et al. A Scalable Dynamic Network Memory Service System[C]//Proc. of the 8th International Conference on High Performance Computing in Asia Pacific Region, Beijing, China. 2005.

4 James O, Reed D A. Markov Model Prediction of I/O Requests for Scientific Applications[C]//Proc. of ACM International Conference on Supercomputing. 2002-06.

5 Jiang S, Zhang X. LIRS: An Efficient Low Inter-reference Recency Set Replacement Policy to Improve Buffer Cache Performance[C]//Proc. of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems. 2002.

（上接第 104 页）

6 结束语

本文基于已有的网络构件研究，提出了构成网络构件的 3 要素：物理域，个体域和整体域，并对其进行了形式化定义。随后分析了网络构件的基本特征，探讨了一种基于 Web service 技术的网络构件系统架构，并将该架构运用到具体实践中。实践证明，采用 Web service 技术的网络构件系统架构具有良好的自适应性和可复用性，能较好地体现网络构件的各种特性，充分发挥 Internet 的计算潜力。

参考文献

1 Brown A, Walinau K. The Current State of CBSE[J]. IEEE Software, 1998, 15(5): 37-46.

2 李　阳, 吴朝晖. 网络构件软件体系模型并行算法研究[J]. 浙江大学学报(工学版), 2004, 38(4): 393-396.

3 徐正权, 张　华. 基于 Web 的软件构件互操作性研究[J]. 计算机应用研究, 2002, 19(9): 48-50.

4 杨芙清, 梅　宏. 浅论软件技术发展[J]. 电子学报, 2002, 30(12).

5 薛云皎, 徐如志. Internet 计算环境下的新型软件形态[J]. 计算机工程与应用, 2004, 40(14): 38-40.

6 Kreger H. Web Services Conceptual Architecture[EB/OL]. (2001-10). http://www-128.ibm.com/developerworks/cn/webservices/ws-wsca/part1/.

7 Tsalgatidou A, Pilioura T. An Overview of Standards and Related Technology in Web Services[J]. Distributed and Parallel Data-bases, 2002, 12(2/3): 135-162.

8 唐飞龙, 李明禄. 网格环境下的一种服务模型及其应用[J]. 计算机工程, 2005, 31(16): 14-16.

9 Fellenstein C, Joseph J. Grid Iron[EB/OL]. [2006-10]. http://www-128.ibm.com/developerworks/cn/db2/library/techarticles/mag_05q1craig/index.html.

10 Steen M V, Tanenbaum A, Sips H. A Scalable Middle-ware Solution for Advanced Wide-aera Web Services[J]. Distributed Systems Engineering, 1999, 6(1): 34-42.