分类号 TP391??? 学号 17060062

U D C 681??? 密级 公开

工学博士学位论文

基于自然语言模型的文件访问模式分析与文件预取方法研究

博 士 生 姓 名 陈辉

学 科 专 业 计算机科学与技术

研 究 方 向 高性能计算

指 导 教 师 周文强 研究员

国防科技大学研究生院

二〇一九年十月

An NLP Model-Based Mechanism for I/O Access Pattern Analysis and File Prefetching

Candidate: **Hui Chen**
Supervisor: **Prof. Enqiang Zhou**

A dissertation
Submitted in partial fulfillment of the requirements
for the degree of **Ph.D of Engineering**
in **Computer Science and Technology**
Graduate School of National University of Defense Technology
Changsha, Hunan, P. R. China
October, 2019

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：基于自然语言模型的文件访问模式分析与文件预取方法研究

学位论文作者签名：_____ 日期： 年 月 日

学位论文版权使用授权书

本人完全了解国防科技大学有关保留、使用学位论文的规定。本人授权国防科技大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密学位论文在解密后适用本授权书。）

学位论文题目：基于自然语言模型的文件访问模式分析与文件预取方法研究

学位论文作者签名：_____ 日期： 年 月 日

作者指导教师签名：_____ 日期： 年 月 日

目　录

表 目 录

图 目 录

摘　要

关键词: 分布式文件系统；文件预取；词向量；循环神经网络

ABSTRACT

Many applications spend a large proportion of the execution time to access ☒ les. To narrow the increasing gap between computing and I/O performance, several optimization techniques were adopted, such as data prefetching and data layout optimization. However, the effectiveness of these optimization processes heavily depends on the understanding of the I/O behavior. Traditionally, spatial locality and temporal locality are mainly considered for data prefetching and scheduling policy. Whereas for most real-world workloads, the ☒ le access pattern is hard to capture.

For the goal of deeply and intelligently understanding the I/O access pattern of modern applications, and efficiently optimizing the performance of current file systems, we propose a new mechanism to embed file names to vectors and train a GRNN (Gated Recurrent Neural Network) to provide policies for file prefetching and cache replacing.

Key Words: Distributed File Systems, File Prefetching, Word Embedding, RNN

符号使用说明

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物，N- 苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物，N- 苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度

第一章　绪论

1.1　研究背景

Stacker. p.1 *Tiered Storage Systems*: p.1

分层存储技术

分层存储系统（Tiered Storage System），又称为层级存储管理（Hierarchical Storage Management），是当前各领域存储系统的主流架构。其基本特点是将文件数据存储在若干层级的存储介质中，不同层级的介质（DRAM，SSD，HDD 等）具有不同的容量、吞吐率、访问延迟与成本等。不同的应用负载和场景下，存储系统中的数据重要性存在差异，可以粗略地分为“热”数据与“冷”数据：正在访问与频繁访问的“热”数据优先存储于更接近 CPU、访问更快的内存、SSD 阵列等存储介质中；近期没有访问需求或访问频率低的“冷”数据主要存储在低层级的磁盘阵列或远程存储服务器中。在现实应用中，数据的“热”与“冷”不是静态的，而是随着应用的访问需求动态变化，数据在不同层级之间的迁移管理成为层次存储系统的基本任务之一。如图1.1所示，分层存储系统按层级从高到低由

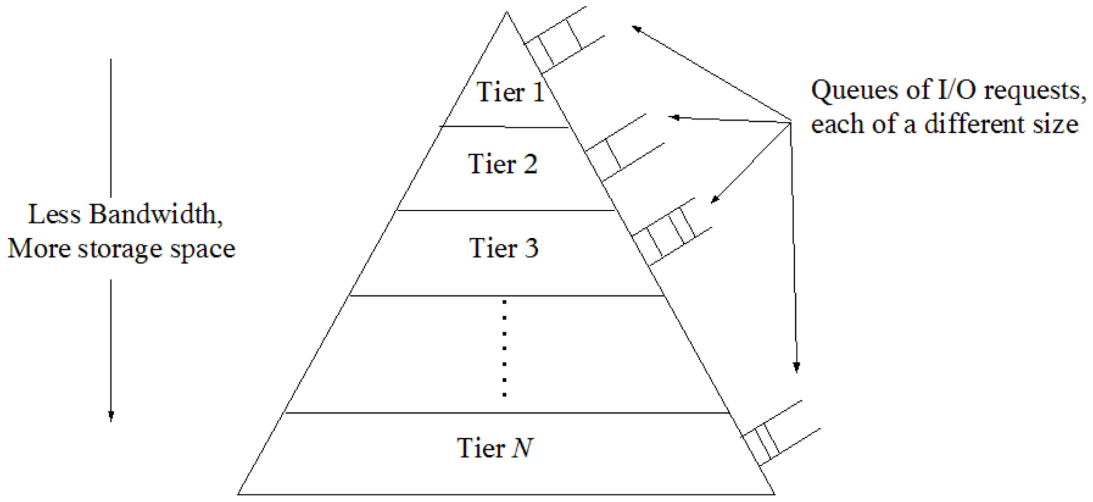

图 1.1　分层存储系统（Tiered Storage System）

不同存储介质组成：

- 高层级存储通常由内存 DRAM（Dynamic Random Access Memory）、非易失性存储（Non-volatile Memory）等介质组成，其特点是读写性能优异、存

储密度高，是实际应用场景中“热”数据的理想存储载体，同时也具有易失（DRAM）或寿命有限（NVM）等缺点

- 较低级存储通常由固态硬盘 (Solid-State Driver)、磁盘 (Hard Disk Drive) 阵列组成，且在生产环境下这些存储阵列通常部署在远程存储服务器 (Storage Servers)，通过 SANs (Storage Area Networks) 或高速互联网络 (InfiniBand) 与客户端或计算服务器连接以提供存储服务。低层级存储性能较弱，存储密度较低，但同时也具有造价低廉、容量大、稳定性好的优点，且可以通过冗余磁盘阵列（Redundant Array of Inexpensive Disks）以及数据复制（Replication）等技术进一步提高数据的稳定性和容错能力。因此这类低层级存储是“冷”数据的理想载体。

近几十年来，分层存储在商业性数据存储领域得到了大规模应用。例如 NetApp FAS 系列存储系统 [1]，IBM DS8880 [2] 等。这些支持 Flash 存储介质的存储系统方案主要可分为两类：第一类存储方案简单地将 Flash 存储介质作为较内存低一级的缓存 [1] ，以起到优化存储系统性能的作用。采用 Flash 存储介质作为缓存的主要优点是集成容易，不需要显式地考虑数据迁移策略，同时造价比 DRAM 低得多。第二类方案则是将 Flash 存储介质作为持续存储加入到层次存储结构中。

在大规模科学与工程计算领域，层次存储结构同样应用广泛。随着闪存技术的飞速发展，传统 HPC 所应用的三层技术架构（计算结点的共享内存 - 并行文件系统 - 归档存储）也随之发生变化。在 HPC 系统中，并行文件系统 (pFS) 对 HPC 性能影响非常大，在许多场景下决定了整个 HPC 的存储性能。传统 HPC 架构在应对超大规模 HPC 集群计算节点同时 Checking Point 需求时，显得力不从心，那就需要在 pFS 之上多加一层高速大容量 (相对于 Memory) 的缓存 (Burst Buffer)。典型代表有此处补充采用 Burst Buffer 技术的超算架构

分层存储架构中的数据迁移管理

层次存储系统中的不同层级间的数据迁移管理是文件系统性能的关键。可粗略分为被动缓存和主动预取。预取（prefetching），也称为预分页（prepaging）或预读（read-ahead），是操作系统数据读取过程中的重要优化方法 [3] [4] [5] 。其目标是预测将来的数据访问，并在请求数据之前将其提前读取到高层级的存储介质中，从而起到掩盖访问延迟的作用。预取是传统缓存技术的一种补充，不同于被动数据迁移，预取技术的关键在于对即将访问的数据内容和生存周期进行主动预测。

- **“热”数据预测**：主动预取需要预测程序下一阶段可能访问的数据，是对数据访问的空间局部性的扩充。针对“热”数据准确的预测将极大地降低访问延迟，而错误预测将会引发浪费传输带宽，挤占缓存空间等负面影响。

- **“热”数据生命周期预测**：与缓存机制中的时间局部性类似，主动预取需要针对缓存数据的时效性和生命周期建立有效的评估。非缓存数据的及时预取有助于提高存命中率，而清除短期内不再读取的数据将提高缓存空间的利用率。

预取的主要流程可总结如下：

1. 针对特定负载，提取访问日志；
2. 分析访问日志，对该负载的文件访问模式进行抽象表达；
3. 将负载的访问模式作为依据，引导文件系统进行主动预取。

缓存 vs 预取概念对比图

1.2 国内外研究现状

p.14

Eshel 等 [6] 提出并实现了缓存文件系统“Panache”，该系统使用 pNFS 以分布式缓存的方式存储 GPFS 中的缓存数据。Frings 等 [7] 针对动态链接库加载过程进行数据预取，从而提高并行应用程序的性能。Rajachandrasekar 等 [8] 提出了一种用户级文件系统，将检查点请求保留在主内存中，并同时写入到持久性存储中。他们的方法包括对远程直接内存访问（RDMA）的支持。

Zhao 等 [9] 提出了另一种缓存中间件，采用一种双阶段缓存技术来减少计算结点和 I/O 结点之间的数据传输量。Isaila 等 [10] 提出了分别位于客户端和 I/O 节点之间，以及 I/O 节点和存储服务器之间的两级预取方案，从而改进了 IBM Blue Gene 的 I/O 转发层的数据传输性能。Prabhakar 等 [11] 通过线性规划对两级缓存系统上的最佳缓存分配进行建模。

Kandemir 等 [12] 定义了 I/O 请求紧迫度的概念，该概念由请求可以延迟多长时间而不影响应用程序性能给出。在对 I/O 请求进行紧迫度分析后，通过优先处理紧急请求来改进缓存机制。Seelam 等 [13] 实现了能够追踪和分析应用 I/O 访问模式的库，并借此引导预取线程将数据提前读取到本地存储。Patrick[14]，He[15]，Tang[16] 等提出了类似的方法（使用访问模式检测指导预取）。

Suei 等 [17] 提出了一种使用 SSD 作为 HDD 缓存的存储集群缓存设计。其设计侧重于响应时间和缓存命中率。

Welch 和 Noer[18] 根据并行文件系统中小文件占多数的特点，将小文件存储在 SSD 中以优化对它们的访问。He 等 [19] 提出了一个代价模型来辅助数据迁移决策。该模型能够评估文件不同区域的访问成本，并将高成本区域放置在 SSD 中。

1.3 本文主要工作

- 研究分析 GlusterFS 的原理、架构和功能实现，提出基于 Tiering 模块的数据迁移管理框架设计。
- 针对 POSIX 文件系统的目录结构、命名习惯等特点，提出采用词向量模型进行量化分析，即，将文件或目录映射到高维向量空间来分析文件、目录间的逻辑关系。
- 针对特定工作负载的 I/O 访问模式进行研究，在文件 / 目录向量化工作的基础上，采用循环神经网络提取和分析 I/O 访问模式，并用于替代传统的缓存算法（LRU 等）指导 GlusterFS 进行数据迁移。

1.4 论文结构

第二章　基于分布式文件系统 GlusterFS 的数据迁移研究

2.1　GlusterFS 整体架构

p.2

2.1.1　GlusterFS 整体架构

GlusterFS[20] 是一个开源、可扩展的分布式文件系统，目前被广泛应用于各类商业存储服务器集群，其主要特性如下：

- 全局命名空间。
- 集群存储管理。
- 模块化的层次机构。
- 内置 replication and geo-replication 特性。
- 自修复功能。
- 高效负载均衡。

2.2　基于 GlusterFS Tiering 功能的缓存管理模块设计

2.2.1　Trace 模块

2.2.2　访问模式识别模型建立

2.2.3　运行时缓存管理

2.3　本章小结

第三章　基于自然语言处理模型的文件访问模式识别

3.1　基于词向量模型的文件名向量化

3.1.1　词向量概述

3.1.2　基于子词模型的文件名向量化

p.1

词向量是将单词在向量空间中分布式表示的自然语言模型，该类模型可帮助学习算法通过对相似单词进行分组来在自然语言处理任务中实现更好的性能。最早使用单词表示法的方法可以追溯到 1986 年 Rumelhart，Hinton 和 Williams 的工作。此想法此后已成功应用于统计语言建模 [1]。后续工作包括自动语音识别和机器翻译 [14、7] 以及各种 NLP 任务 [2、20、15、3、18、19、9] 的应用。

Mikolov 等于 2013 年引入了 Skip-gram 模型，这是一种从大量非结构化文本数据中学习单词的高质量向量表示的有效方法。与大多数以前使用的用于学习单词向量的神经网络体系结构不同，Skip-gram 模型的训练不涉及密集矩阵乘法，因此训练过程非常高效：经过优化的单机实现可以在一天中培训超过 1000 亿个单词。

More details about Skip-gram.

p.1 通过为每个单词分配不同的向量，学习此类表示形式的流行模型会忽略单词的形态。这是一个限制，特别是对于具有大量词汇和许多稀有单词的语言。在本文中，我们提出了一种基于跳过图模型的新方法，其中每个单词都表示为一包字符 n-gram。向量表示与每个字符 n-gram 相关；单词被表示为这些表示的总和。我们的方法快速，可以快速在大型语料库上训练模型，并允许我们为未出现在训练数据中的单词计算单词表示。我们在词的相似性和类比任务上用九种不同的语言评估词的表示形式。通过与最近提出的形态词表示法进行比较，我们表明，我们的向量在这些任务上达到了最先进的性能。

3.2　基于循环神经网络的文件访问模式分析

3.2.1　循环神经网络及其在时间序列分析中的应用

3.2.2　采用 GRNN（Gated Recurrent Neural Network）的文件访问模式分析模型

3.3　本章小结

第四章　实验设计与分析

4.1　实验环境与工作负载

4.1.1　存储服务器集群搭建

4.1.2　工作负载选取

编译。网页访问

4.2　实验数据采集与预处理

4.2.1　文件名向量化

4.2.2　文件访问日志记录与预处理

4.3　循环神经网络模型训练与仿真测试

4.4　实验结果与分析

4.5　本章小结

第五章　总结

5.1　本文工作总结

5.2　未来工作展望

致　谢

衷心感谢导师 xxx 教授和 xxx 副教授对本人的精心指导。他们的言传身教将使我终生受益。

感谢 NudtPaper，它的存在让我的论文写作轻松自在了许多，让我的论文格式规整漂亮了许多。

参考文献

[1] NetApp FAS Storage Systems. https://www.netapp.com/us/documentation/fas-storage-systems.aspx.

[2] IBM DS8880 Data Systems. https://www.ibm.com/cn-zh/marketplace/ds8000.

[3] Griffioen J, Appleton R. Reducing File System Latency using a Predictive Approach. [C]. In USENIX summer. 1994: 197–207.

[4] Amer A, Long D D, Burns R C. Group-based management of distributed file caches [C]. In Proceedings 22nd International Conference on Distributed Computing Systems. 2002: 525–534.

[5] Nanopoulos A, Katsaros D, Manolopoulos Y. A data mining algorithm for generalized web prefetching [J]. IEEE Transactions on Knowledge & Data Engineering. 2003 (5): 1155–1169.

[6] Eshel M, Haskin R L, Hildebrand D, et al. Panache: A Parallel File System Cache for Global File Access. [C]. In FAST. 2010: 1–14.

[7] Frings W, Ahn D H, LeGendre M, et al. Massively Parallel Loading [C/OL]. In Proceedings of the 27th International ACM Conference on International Conference on Supercomputing. New York, NY, USA, 2013: 389–398. http://doi.acm.org/10.1145/2464996.2465020.

[8] Rajachandrasekar R, Moody A, Mohror K, et al. A 1 PB/s file system to checkpoint three million MPI tasks [C]. In Proceedings of the 22nd international symposium on High-performance parallel and distributed computing. 2013: 143–154.

[9] Zhao D, Qiao K, Raicu I. HyCache+: Towards Scalable High-Performance Caching Middleware for Parallel File Systems [C]. In 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. May 2014: 267–276.

[10] Isaila F, Blas J, Carretero J, et al. Design and Evaluation of Multiple-Level Data Staging for Blue Gene Systems [J]. IEEE Transactions on Parallel and Distributed Systems. 2011, 22 (6): 946–959.

[11] Prabhakar R, Srikantaiah S, Kandemir M, et al. Adaptive Multi-level Cache Allocation in Distributed Storage Architectures [C/OL]. In Proceedings of the 24th ACM International Conference on Supercomputing. New York, NY, USA, 2010: 211–221. http://doi.acm.org/10.1145/1810085.1810115.

[12] Kandemir M, Yemliha T, Prabhakar R, et al. On Urgency of I/O Operations [C]. In 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). May 2012: 188–195.

[13] Seelam S, Chung I, Bauer J, et al. Masking I/O latency using application level I/O caching and prefetching on Blue Gene systems [C]. In 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS). April 2010: 1–12.

[14] Patrick C M, Kandemir M, Karak M, et al. Cashing in on Hints for Better Prefetching and Caching in PVFS and MPI-IO [C/OL]. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. New York, NY, USA, 2010: 191–202. http://doi.acm.org/10.1145/1851476.1851499.

[15] He J, Sun X, Thakur R. KNOWAC: I/O Prefetch via Accumulated Knowledge [C]. In 2012 IEEE International Conference on Cluster Computing. Sep. 2012: 429–437.

[16] Tang H, Zou X, Jenkins J, et al. Improving read performance with online access pattern analysis and prefetching [C]. In European Conference on Parallel Processing. 2014: 246–257.

[17] Suei P, Yeh M, Kuo T. Endurance-Aware Flash-Cache Management for Storage Servers [J]. IEEE Transactions on Computers. 2014, 63 (10): 2416–2430.

[18] Welch B, Noer G. Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions [C]. In 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST). May 2013: 1–12.

[19] He J, Bent J, Torres A, et al. I/O acceleration with pattern detection [C]. In Proceedings of the 22nd international symposium on High-performance parallel and distributed computing. 2013: 25–36.

[20] GlusterFS: A scalable network filesystem. http://www.gluster.org/.

附录 A　模板提供的希腊字母命令列表

大写希腊字母:

Γ \Gamma	Λ \Lambda	Σ \Sigma	Ψ \Psi
Δ \Delta	Ξ \Xi	Υ \Upsilon	Ω \Omega
Θ \Theta	Π \Pi	Φ \Phi	
$\varGamma$ \varGamma	$\varLambda$ \varLambda	$\varSigma$ \varSigma	$\varPsi$ \varPsi
$\varDelta$ \varDelta	$\varXi$ \varXi	$\varUpsilon$ \varUpsilon	$\varOmega$ \varOmega
$\varTheta$ \varTheta	$\varPi$ \varPi	$\varPhi$ \varPhi	

小写希腊字母:

α \alpha	θ \theta	o o	τ \tau
β \beta	ϑ \vartheta	π \pi	υ \upsilon
γ \gamma	ι \iota	ϖ \varpi	ϕ \phi
δ \delta	κ \kappa	ρ \rho	φ \varphi
ϵ \epsilon	λ \lambda	ϱ \varrho	χ \chi
ε \varepsilon	μ \mu	σ \sigma	ψ \psi
ζ \zeta	ν \nu	ς \varsigma	ω \omega
η \eta	ξ \xi	$\varkappa$ \varkappa	\digamma
$\upalpha$ \upalpha	$\uptheta$ \uptheta	o \mathrm{o}	$\uptau$ \uptau
$\upbeta$ \upbeta	$\upvartheta$ \upvartheta	$\uppi$ \uppi	$\upupsilon$ \upupsilon
$\upgamma$ \upgamma	$\upiota$ \upiota	$\upvarpi$ \upvarpi	$\upphi$ \upphi
$\updelta$ \updelta	$\upkappa$ \upkappa	$\uprho$ \uprho	$\upvarphi$ \upvarphi
$\upepsilon$ \upepsilon	$\uplambda$ \uplambda	$\upvarrho$ \upvarrho	$\upchi$ \upchi
$\upvarepsilon$ \upvarepsilon	$\upmu$ \upmu	$\upsigma$ \upsigma	$\uppsi$ \uppsi
$\upzeta$ \upzeta	$\upnu$ \upnu	$\upvarsigma$ \upvarsigma	$\upomega$ \upomega
$\upeta$ \upeta	$\upxi$ \upxi		

希腊字母属于数学符号类别，请用\bm命令加粗，其余向量、矩阵可用\mathbf。