**密级:**



**博士学位论文**

**分布式数据并行应用内存溢出错误分析与诊断**

**作者姓名:**  许利杰

**指导教师:**  魏峻 研究员

中国科学院软件研究所

**学位类别:**  工学博士

**学科专业:**  计算机软件与理论

**培养单位:**  中国科学院软件研究所

**2015 年 10 月**

**Understanding and Diagnosing Causes of Out of Memory Errors in Distributed Data-parallel Applications**

**By**

**Lijie Xu**

**A Dissertation Submitted to**

**University of Chinese Academy of Sciences**

**In partial fulfillment of the requirement**

**For the degree of**

**Doctor of Computer Science**

Institute of Software

Chinese Academy of Sciences

**October 2015**

**独创性声明**

本人声明所呈交的论文是我个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或撰写过的研究成果。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。

签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**关于论文使用授权的说明**

本人完全了解中国科学院软件研究所有关保留、使用学位论文的规定，即：中国科学院软件研究所有权保留送交论文的复印件，允许论文被查阅和借阅；中国科学院软件研究所可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。

(保密的论文在解密后应遵守此规定)

签名：\_\_\_\_\_\_\_\_\_\_ 导师签名：\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_

摘要

大数据以数据量大、数据类型多样、产生与处理速度要求快、价值高的4V特性迅速成为工业和学术界关注的焦点。由于传统软件系统难以在可接受的时间范围内处理大数据，因此分布式数据并行处理框架如Hadoop，Spark等，成为了解决该问题的核心软件系统，并在工业界得到广泛应用。分布式数据并行应用指然而在框架处理大数据时面临诸多内存管理方面的问题： 如内存溢出频繁却不知道原因，内存消耗影响因素多且难以预知，大内存使用和回收效率低。本文旨在研究和解决分布式处理框架在内存分配、使用、回收方面的问题，分别从内存分析与预测、错误诊断与修复、内存管理机制改进方面开展研究。具体包括采用统计、模拟和规则方法对处理框架中的内存使用进行建模与预测，提出框架与应用的内存溢出错误诊断与修复方法，改进JVM内存分配，改进框架中间数据存储与处理方式，最终从可用性、容错性、效率方面提高框架处理数据性能。

分布式数据并行应用广泛内存溢出错误常见。用户不知道错误原因，本论文实证分析了内存溢出错误原因，具有良好扩展性的分布式处理框架现已被广泛应用于企业和科研机构的大数据处理与分析当中。然而，目前的主流框架如Hadoop、Spark等，在内存管理方面均存在性能和可靠性问题。具体表现在任务执行时内存消耗量大且难以预知，内存溢出频繁却难以定位发生原因，存在内存膨胀现象且内存回收效率低等问题。本文针对这几个问题，分别从内存消耗分析与预测、内存溢出错误诊断与内存管理机制改进三个方面开展具体研究。当前，已经完成内存消耗模型的建立与消耗量预测；已经对内存溢出错误进行了分析，总结出了诊断方法；在内存管理机制方面有一定分析，有待进一步深入研究。

本论文针对分布式数据并行应用，主要贡献如下：

**1) 内存溢出错误的实证分析：**分析了内存溢出错误原因，修复方法，及潜在的可以提高框架错误容忍能力，方便用户进行错误诊断的方法。

**2) 内存用量模型构建与用量估算：**建立应用内存用量模型，可以估算内存用量。

**3) 内存溢出错误的诊断方法：**提出一种半自动化诊断方法帮助用户诊断内存溢出错误。

本论文的研究成果对用户和框架设计者均有参考意义，可进一步应用于。综合上述研究成果，论文设计并实现了。该系统主要由。

**关键词：**分布式数据并行应用，MapReduce，内存溢出，分析诊断

**Unnderstanding and Diagnosing Causes of Out of Memory Errors in Distributed Data-parallel Application**

Lijie Xu (Computer Software and Theory)

Directed by Professor Jun Wei

Abstract

Out of memory (OOM) errors occur frequently in MapReduce applications, where memory space is shared by the MapReduce framework and user code. However, current memory profilers and memory leak detectors can only figure out what objects are (unnecessarily) persisted in memory, but cannot figure out where the objects come from and why they become so large.

In this paper, we observe that improper job configurations, data skew, and memory-consuming user code can lead to the excessive memory usage in MapReduce applications. Based on this observation, we propose a semi-automatic dataflow- centric memory backtracking approach to diagnose OOM errors in MapReduce applications. Our approach first automatically quantifies the relationship among memory usage, runtime data, and the configurations. Then, it uses quantitative rules to trace the OOM error back to the problematic user code, data, or configurations.

The evaluation on 20 real-world OOM errors in diverse Hadoop MapReduce applications shows that our approach can accurately identify the root causes of 15 OOM errors, and partly identify the root causes of 5 OOM errors where the user code is written by high-level languages.

**Keywords:** Data-parallel applications, MapReduce, Out of memory, Error diagnosis

目录

[摘要 I](#_Toc369296357)

[Abstract III](#_Toc369296358)

[目录 VII](#_Toc369296359)

[第一章 绪论 1](#_Toc369296360)

[1.1 研究背景 1](#_Toc369296361)

[1.2 问题分析 2](#_Toc369296362)

[1.3 论文工作 3](#_Toc369296363)

[1.3.1 研究思路 3](#_Toc369296364)

[1.3.2 研究内容 4](#_Toc369296365)

[1.4 论文组织 5](#_Toc369296366)

[第二章 分布式缓存技术研究综述 7](#_Toc369296367)

[2.1 分布式缓存概述 7](#_Toc369296368)

[2.1.1 分布式缓存的特性 7](#_Toc369296369)

[2.1.2 典型应用场景 8](#_Toc369296370)

[2.1.3 分布式缓存的发展 8](#_Toc369296371)

[2.1.4 分布式缓存与NoSQL 9](#_Toc369296372)

[2.1.5 分布式缓存与极限事务处理 9](#_Toc369296373)

[2.1.6 相关标准规范 10](#_Toc369296374)

[2.2 分布式缓存技术分析 10](#_Toc369296375)

[2.2.1 弹性资源供给 12](#_Toc369296376)

[2.2.2 可用性与可靠性 12](#_Toc369296377)

[2.2.3 敏捷性与自适应性 13](#_Toc369296378)

[2.2.4 多租户 14](#_Toc369296379)

[2.2.5 数据管理 15](#_Toc369296380)

[2.2.6 数据安全与隐私防护 19](#_Toc369296381)

[2.3 典型系统 19](#_Toc369296382)

[2.3.1 Oracle Coherence 19](#_Toc369296383)

[2.3.2 Memcached 20](#_Toc369296384)

[2.3.3 Terracotta 20](#_Toc369296385)

[2.3.4 Microsoft AppFabric Caching 20](#_Toc369296386)

[2.3.5 典型系统比较 21](#_Toc369296387)

[2.4 云计算环境下分布式缓存面临的新挑战 22](#_Toc369296388)

[2.4.1 缓存策略的自适应与自管理 22](#_Toc369296389)

[2.4.2 虚拟化环境下缓存数据迁移的优化 23](#_Toc369296390)

[2.4.3 缓存服务的性能隔离 23](#_Toc369296391)

[2.5 研究趋势 23](#_Toc369296392)

[2.6 本章小结 24](#_Toc369296393)

[第三章 基于机器学习的缓存策略自适应重配方法 25](#_Toc369296394)

[3.1 引言 25](#_Toc369296395)

[3.2 问题分析 26](#_Toc369296396)

[3.2.1 缓存策略的评价与分析 26](#_Toc369296397)

[3.2.2 缓存策略自适应面临的挑战 28](#_Toc369296398)

[3.3 关键机制 28](#_Toc369296399)

[3.3.1 总体架构 28](#_Toc369296400)

[3.3.2 数据预处理 29](#_Toc369296401)

[3.3.3 模型的参数选择 31](#_Toc369296402)

[3.3.4 性能模型的训练 33](#_Toc369296403)

[3.3.5 性能模型的持续优化 33](#_Toc369296404)

[3.3.6 缓存策略重配置 35](#_Toc369296405)

[3.3.7 缓存策略重配开销的优化 37](#_Toc369296406)

[3.4 实验与评价 37](#_Toc369296407)

[3.4.1 实验设置和评价方法 38](#_Toc369296408)

[3.4.2 缓存策略重配的有效性 39](#_Toc369296409)

[3.4.3 缓存策略重配周期的调整 40](#_Toc369296410)

[3.4.4 监测开销评价 42](#_Toc369296411)

[3.4.5 性能模型的优化 42](#_Toc369296412)

[3.4.6 缓存策略重配开销的优化 43](#_Toc369296413)

[3.4.7 扩展性评价 44](#_Toc369296414)

[3.4.8 进一步的讨论 45](#_Toc369296415)

[3.5 相关工作比较 45](#_Toc369296416)

[3.5.1 基于模拟的方法 46](#_Toc369296417)

[3.5.2 基于规则的方法 46](#_Toc369296418)

[3.5.3 基于机器学习的方法 47](#_Toc369296419)

[3.5.4 其他方法 47](#_Toc369296420)

[3.6 本章小结 48](#_Toc369296421)

[第四章 性能干扰与开销敏感的缓存数据迁移方法 49](#_Toc369296422)

[4.1 引言 49](#_Toc369296423)

[4.2 问题分析 50](#_Toc369296424)

[4.2.1 Xen 的I/O架构 50](#_Toc369296425)

[4.2.2 虚拟化环境对缓存数据迁移的影响 51](#_Toc369296426)

[4.3 关键机制 53](#_Toc369296427)

[4.3.1 性能干扰感知的预测模型 54](#_Toc369296428)

[4.3.2 数据迁移开销模型 56](#_Toc369296429)

[4.3.3 开销敏感的数据迁移算法 57](#_Toc369296430)

[4.4 系统实现 59](#_Toc369296431)

[4.4.1 负载均衡度检测 59](#_Toc369296432)

[4.4.2 预测模型的训练 60](#_Toc369296433)

[4.4.3 预测模型的优化 60](#_Toc369296434)

[4.5 实验与评价 61](#_Toc369296435)

[4.5.1 实验环境和设置 61](#_Toc369296436)

[4.5.2 预测模型的性能 62](#_Toc369296437)

[4.5.3 预测模型的鲁棒性 62](#_Toc369296438)

[4.5.4 数据迁移算法的有效性 63](#_Toc369296439)

[4.5.5 数据迁移算法的扩展性 65](#_Toc369296440)

[4.5.6 迁移开销模型的有效性 65](#_Toc369296441)

[4.5.7 进一步的讨论 66](#_Toc369296442)

[4.6 相关工作比较 66](#_Toc369296443)

[4.7 本章小结 67](#_Toc369296444)

[第五章 面向多租户的云缓存服务性能隔离方法 69](#_Toc369296445)

[5.1 引言 69](#_Toc369296446)

[5.2 问题分析 70](#_Toc369296447)

[5.2.1 研究背景 70](#_Toc369296448)

[5.2.2 内存资源竞争 71](#_Toc369296449)

[5.2.3 网络资源竞争 72](#_Toc369296450)

[5.3 关键机制 73](#_Toc369296451)

[5.3.1 总体架构 73](#_Toc369296452)

[5.3.2 缓存空间划分机制 74](#_Toc369296453)

[5.3.3 租户敏感的缓存替换算法 76](#_Toc369296454)

[5.3.4 租户带宽分配机制 76](#_Toc369296455)

[5.4 系统实现 78](#_Toc369296456)

[5.5 实验与评价 78](#_Toc369296457)

[5.5.1 实验设置和评价方法 79](#_Toc369296458)

[5.5.2 性能隔离框架的有效性 80](#_Toc369296459)

[5.5.3 缓存空间划分评价 81](#_Toc369296460)

[5.5.4 租户带宽分配评价 81](#_Toc369296461)

[5.5.5 性能开销 82](#_Toc369296462)

[5.5.6 进一步的讨论 82](#_Toc369296463)

[5.6 相关工作比较 83](#_Toc369296464)

[5.7 本章小结 84](#_Toc369296465)

[第六章 系统原型实现 85](#_Toc369296466)

[6.1 系统总体架构 85](#_Toc369296467)

[6.2 缓存服务器 86](#_Toc369296468)

[6.2.1 请求处理 86](#_Toc369296469)

[6.2.2 数据迁移管理 87](#_Toc369296470)

[6.2.3 性能监测与统计 87](#_Toc369296471)

[6.3 节点管理器 88](#_Toc369296472)

[6.3.1 缓存节点管理 89](#_Toc369296473)

[6.3.2 资源监测与统计 90](#_Toc369296474)

[6.3.3 缓存节点状态管理 91](#_Toc369296475)

[6.4 主节点 91](#_Toc369296476)

[6.4.1 缓存集群管理 91](#_Toc369296477)

[6.4.2 节点部署 93](#_Toc369296478)

[6.4.3 管理控制台 94](#_Toc369296479)

[6.5 缓存客户端 94](#_Toc369296480)

[6.5.1 命令处理 95](#_Toc369296481)

[6.5.2 配置管理 96](#_Toc369296482)

[6.6 关键技术 96](#_Toc369296483)

[6.7 本章小结 98](#_Toc369296484)

[第七章 结束语 99](#_Toc369296485)

[7.1 论文工作总结 99](#_Toc369296486)

[7.2 进一步的工作 100](#_Toc369296487)

[参考文献 103](#_Toc369296488)

[发表文章目录 113](#_Toc369296489)

[参与科研项目目录 115](#_Toc369296490)

[获得发明专利目录 117](#_Toc369296491)

[获得软件著作权目录 119](#_Toc369296492)

[致 谢 121](#_Toc369296493)

# 绪论

本章简要介绍了论文的研究背景，并分析了论文研究的三个主要问题，接下来给出了论文的研究思路和研究内容。在本章的结尾，列出了论文的组织方式。

* 1. 研究背景

### 大数据与大数据处理

数据一直是工业界和学术界的重要研究对象。从数据中获益，从数据中获取知识成为数据处理的主要目的。例如，搜索引擎每天都在收集、处理、分析海量的网页及多媒体数据，并对外提供数据查询服务。社交网站每天记录大量的用户数据，组织形成虚拟的人际网络。商业智能公司依靠分析企业生产和销售的数据，来为企业提供商务决策支持。学术研究机构也在天文、物理、化学、地理、生命科学等方面不断积累大量的实验数据，从数据中分析挖掘各种科学知识。

互联网、云计算、移动计算、物联网等技术的发展使得数据的产生速度越来越快、数据规模越来越大、数据类型越来越多。诸如社交网站Facebook每天要处理25亿条消息，500+TB的新数据，用户每天上传32亿张照片，每半个小时扫描的数据大约为105TB [FB]。早在2008年，Google每天就要处理约20,000 TB（20PB）的数据。Youtube网站每分钟用户会上传越48小时的视频。Twitter每天大约有1.75亿条新微博 [Youtube]。

为了描述数据的新特性，“大数据”的概念被提出。大数据具有数据量大、数据类型多样、产生与处理速度要求快、价值高的4V特性。这些特性也使得传统软件系统难以在可接受的时间范围内对大数据进行获取、管理和处理。例如，出现于70年代的关系数据库很好地解决了关系型数据的存储、管理以及OLTP（在线事务处理）。之后出现的数据仓库将数据按主题组织，解决了数据建模以及OLAP（在线数据分析）。然而由于传统数据库和数据仓库在扩展性方面的问题，使得大数据的处理与分析变得困难。为了解决这个问题，工业届和学术界设计了可扩展的分布式数据并行处理框架，这些被提出并得到广泛应用。

### 分布式数据并行处理框架

为了解决大数据的有效高效处理，工业界和学术界提出了很多分布式并行处理框架。分布式数据并行框架采用“分治－聚合”策略来对大数据的分布处理。以Google MapReduce为代表，包含非开源框架Microsoft Dryad，Apache Spark等。框架的运行模式是MapReduce-like。拥有共同的编程范型。

2004年Google在OSDI会议上提出了基于分治、归并、函数式编程思想的MapReduce分布式计算框架，获得巨大成功。之后模仿GFS和MapReduce实现的开源系统Apache Hadoop将大数据的分布式存储与处理技术推广到了工业界和学术界，并迅速成为大数据的主流存储与处理平台。2007年微软公司提出了Dryad分布式计算框架，虽然是基于MapReduce，但与MapReduce固定数据处理流程不同的是Dryad允许用户将任务组织成DAG（有向无环图），任务结构更加灵活，表达能力也更强。2012年UC Berkeley的AMPLab提出了基于内存，适合迭代计算的Spark[[1]](#endnote-1)分布式大数据处理框架。该框架允许用户将公共或中间处理数据缓存到内存，极大缩短了数据处理的时间。

### 分布式数据并行应用

运行在分布式处理框架之上的应用被成为分布式数据并行应用。

大数据分布式处理的框架的产生使得上层的大数据分析应用成为可能。Yahoo开发了从SQL语言到MapReduce job的转换框架。Facebook开发了建立在Hadoop之上的分布式数据仓库Hive，核心也是将类SQL查询分析语言转变成MapReduce job。Mahout是建立Hadoop之上的机器学习框架。HBase是模仿Google BigTable，建立在Hadoop分布式文件系统HDFS上的Key-Value数据库。应用被广泛应用于网页索引构建。分布式数据并行应用指的是运行在分布式数据并行框架之上，对大数据进行分布处理的作业（Job）。下面本文简称为“数据并行应用”。数据并行应用广泛应用于网页索引构建、日志挖掘、大数据SQL语句执行、机器学习、社交网络图分析等。

### 应用运行的三层结构

整个应用从开发到运行产生结果可以拥有三层结构，用户层、框架执行层、物理内存层。

用户层负责撰写应用，应用（Job）的组成元素是 <data，configurations，user code>。数据（data）指的是输入的大数据，大数据一般分布存放在分布式文件系统（如Hadoop FileSystem上）。配置参数（configurations）用于指定框架运行时需要的信息，如partition个数。用户代码包含map(), reduce()以及可选的combine()。

框架执行层负责job的执行，一个job包含多个map/reduce 任务（task）。框架负责启动每个task，并按照分布式数据流的方式执行固定的处理步骤。如在Hadoop中的处理阶段可以分为map，shuffle，reduce等阶段。数据流指运行时的输入／输出／中间数据，以及数据以来关系。每个task是一个进程或线程，比如Hadoop MapReduc中每个task运行在一个JVM中。

物理内存层反映应用任务（task）的真实内存消耗，应用的内存使用情况是框架使用一部分内存，用户代码也消耗一部分内存。框架在内存暂存一些数据，用户代码在处理输入数据时也会消耗内存。

* 1. 论文工作

### 研究问题及研究动机

尽管框架为用户提供了简单的编程范型、编程接口，用户不需要具有分布式系统的知识也可以使用。这种设计方式方便用户只关注于处理逻辑本身，无需考虑应用是如何分布执行的及中间处理过程。这种方式方便用户撰写数据逻辑，但对用户来说，如果应用出现运行时错误，那么对错误的诊断和修复都是困难的。内存溢出错误是一种常见的的错误，由于应用会在内存中处理大数据。内存溢出错误目前还不能被框架的错误容忍机制处理（tolerate）。除了内存溢出问题，我们也发现了用户不明白应用的消耗。

**现象1: 内存溢出错误常见，不明白错误原因及修复方法**

例如，在程序员的公开论坛StackOverflow.com中有很多关于内存溢出的问题，比如

① 我很诧异为什么我的 job 会在 map 阶段出现内存溢出错误，内存 溢出的原因是什么？[1]

② 如何处理并修复内存溢出错误？我调了一些配置参数（如reducer数目），但不起作用[2]

③ 我的Spark程序出现了内存溢出错误，我不能找到

我们收集到 276 个 real-world OOM errors

发现各个运行阶段均存在内存溢出错误（map，shuffle，reduce）

发现各类应用均存在内存溢出（用户手写代码或高层语言产生的代码）

**现象2: 用户不明白内存用量，也不知道该为内存分配多少内存。**

① 为什么我的 map 任务会消耗如此多的内存？[1]

② 我能把任务的内存设为小于 2GB 吗？ [2]

③ 我想准确地预测出 job 的内存用量，因为内存设的不合理，我的 job 在 40 个节点的集群上跑了 5 天后失败了[3]

**现象3: 用户对框架运行不理解，手工诊断内存溢出错误困难。**

① 我的job总是内存溢出错误，这是Hadoop自己的bug吗？我不懂处理流程中哪些步骤会消耗过多内存，希望能够得到建议。

[http://stackoverflow.com/questions/17975335/hadoop-streaming-memory-usage]

② 我不是很理解MapReduce数据流的工作原理。我的一个很紧要的job在reduce阶段出现了内存溢出错误，希望大家帮我看一下为什么会出现这种错误。[http://stackoverflow.com/questions/19490723/detailed-dataflow-in-hadoops-mapreduce]。

问题本质及研究问题

**内存溢出错误原因**：影响内存用量的因素太多（有静态配置也有动态数据流），不知道哪个是内存溢出错误原因

研究问题1：有没有常见的错误原因类型及修复方法？

**用量预测**：预测难是因为难以确定静态因素（如配置，用户代码）与动态内存用量之间的关系

研究问题2：有没有内存用量模型来指导内存用量预测？

**手工诊断困难**：目前没有诊断方法，需要用户对框架，应用及内存使用都非常了解，有丰富的调试经验

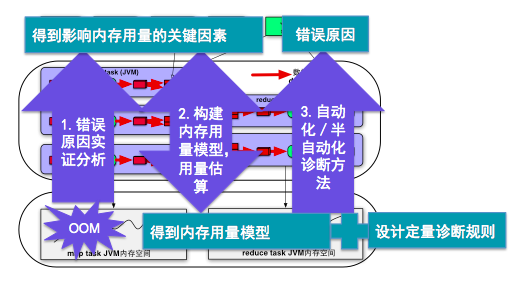
研究问题3：有没有可以直接或者辅助用户诊断的方法？

然而在应用使用分布式处理框架处理大数据时，经常面临着内存消耗未知、内存溢出、内存分配和回收效率低等内存管理问题。内存消耗方面：因为磁盘和内存的价格问题，内存容量一般要差磁盘一个数量级。内存容量有限而且所有任务的输入、输出和临时数据都要流经内存。因此内存消耗一般较大，变化较快，且受框架配置参数、数据流影响，难以分析和预测。内存溢出方面：框架不仅在内存中处理数据，而且将部分中间数据存储在内存中，加上配置参数、数据流和用户定义的程序影响，内存溢出难以诊断原因和修复。在著名的程序错误问答网站stackoverflow.com上有超过10000个question是关于“hadoop outofmemory” ，在Spark邮件列表上有111条（当前共917条）是关于“outofmemory”的。本人在实践实验过程中，也经常碰到“Heap Error”等内存错误，其中在使用Mahout的K-means算法分析20GB数据时，碰到的“Heap Error”至今也没找到出错原因。内存分配和回收方面：JVM在管理大对象和巨量对象时存在严重的GC（垃圾回收）效率问题，而框架在处理大数据时会产生大数组、生命周期不一的大量对象。由于JVM内部的分区管理问题，GC发生频繁且持续时间长。另外，JVM在管理超过130G的内存时会出现诸多问题 。

本论文主要研究大数据分布式处理框架的内存管理技术。以主流的开源分布式框架Hadoop、Spark为平台，解决框架和应用在内存分配、使用和回收方面的诸多问题。针对内存消耗影响因素多且难以预知的问题，给出内存消耗的分析方法，帮助用户和系统管理人员理解框架的内存消耗影响因素与规律；给出内存消耗建模与预测方法，帮助用户在运行job时能够提前预知内存消耗情况。针对内存溢出的问题，给出错误诊断方法，定位出错阶段和错误来源；给出错误修复方法，帮助用户或框架再次运行job时能够成功。针对内存分配和回收效率低的问题，一方面改进现有JVM的区域分配与GC机制，降低GC频率，提高大内存管理能力；另一方面改进框架的中间数据处理方式，更合理地使用内存，提升job的执行性能。总体为了提高分布式处理框架的可用性、容错性和执行效率。

### 研究思路

通过实证分析来总结内存溢出的常见原因及修复方法，得出影响内存用量的关键影响因素。通过构建内存用量模型，对内存用量进行预测。有了内存用量模型以后，设计定量规则，根据job的静态和动态信息，自底向上定位出错原因。



### 研究内容

错误原因实证分析：分析内存溢出错误原因，修复方法，及潜在可以避免内存溢出错误的方法，RQ1、RQ2、RQ3。

内存用量模型构建：建立静态因素 <data, configurations, user code> 与动态内存用量之间的关系。

内存溢出错误诊断：给定job，如何诊断出其静态因素中的错误原因，比如有问题的配置，内存消耗量高的代码段，数据流异常以及有问题的配置参数。

基于上述研究路线，论文借鉴 思想， 具有直接的意义，，也可为相关关键技术的研究提供一定的借鉴和指导。论文研究内容具体包括以下几个部分：

1. **内存溢出错误原因及修复方法实证分析**

论文研究三个问题。

1. **内存用量模型构建与内存用量估算**

论文首先研究了虚拟化环境对缓存数据迁移的影响。基于上述模型，提出了一种开销敏感的数据迁移算法。

1. **缓存服务性能隔离方法研究**

论文围绕问题展开研究。在内存资源分配方面，针对已有工作的不足，设计了一种新的缓存空间划分机制。在此基础上，提出。

### 创新点

本文针对分布式并行应用的内存用量及内存溢出错误开展研究，属于性能与可靠性问题。以前的性能问题主要研究应用的执行时间以及网络传输量优化，我们研究了内存使用问题。以前的可靠性问题主要研究一般类型的错误，我们专门研究内存溢出错误。内存溢出错误在一般的应用程序中很常见，我们相信研究也结果对未来问题有借鉴意义。

Contributions。

* 1. 已有研究工作不足

分布式数据并行应用存在诸多内存使用问题

1. **内存溢出错误常见**

拟解决的问题是，这其中面临两项关键挑战：一是；二是。已有研究工作，本文的研究对象为内存

1. **内存用量的建模与估算**

解决的问题。已有研究工作，如何在因此，需要找到一种方法。

1. **内存溢出错误原因诊断**

错误原因难以诊断。



图1.2论文整体研究路线

* 1. 论文组织

论文总共分为七章，其中，后续章节的组织结构和内容如下：

第二章首先分析介绍了的特性、典型应用场景、发展阶段、相关标准规范以及推动缓存技术发展的若干关键要素。在此基础上，基于该框架对当前缓存技术进行总结和分析，并对典型系统进行了比较，指出了存在的问题和不足，这些构成了本文研究的出发点。

第三章以为研究对象，关注当前问题。

第四章主要解决优化问题，提出一种性能干扰与开销敏感的数据迁移方法，重点关注VM性能干扰的刻画以及迁移时间与性能衰减值如何权衡的问题。

第五章针对 。

第六章介绍了 。

第七章总结论文工作，并对进一步的研究工作进行了展望。

# 背景知识

作为云计算环境下提升应用性能的一种重要手段，指出了存在的问题和不足，这些构成了本文研究的出发点。

* 1. 分布式数据并行应用概述

### 编程范型

什么是分布式数据并行应用？

### 用户代码

什么是分布式数据并行应用？

### 配置

什么是分布式数据并行应用？

### 数据流

### 典型系统Hadoop和Spark

# 数据并行应用内存溢出错误实证研究

摘要。

* 1. 概述

该部分实证研究试图回答以下三个研究问题：

**RQ1**：应用内存溢出的错误原因是什么？有没有一些常见的错误类型（cause pattern）？

**RQ2**：用户是怎么修复内存溢出错误的，有没有一些常见的修复策略（pattern）？

**RQ3**：有没有可以提高框架错误容忍能力的方法，有没有可以方便错误诊断的方法？

* 1. 研究方法

### 研究对象

我们选择运行在Apache Hadoop和Apache Spark上的分布式数据并行应用作为我们的研究对象。因为目前没有专门针对内存溢出错误的缺陷库（bug repository），用户在遇到内存溢出错误时会选择公开论坛或者Hadoop/Spark的邮件列表等公开途径来询问错误原因及修复方法。因此，我们以这些公开应用为研究对象，获得应用的方法是，我们先在公开渠搜索内存溢出错误的关键字，比如“Hadoop out of memory”，“Hadoop outofmemory”，“Spark OOM”，在StackOverflow.com，Hadoop邮件列表[HadoopMail]，Spark users/dev邮件列表[SparkMail]，开发者的博客，和两本关于MapReduce的畅销书籍中的例子[MBook]。

通过关键词搜索，我们一共发现1151个例子（issue），我们手工检测每个例子看是会否满足下列三个条件：

1. 例子是Hadoop/Spark的内存溢出错误。

其中有786例子不是内存溢出错误，比如只包含部分的关键字“Hadoop Memory”，只讲Hadoop Memory的配置方法。

1. 内存溢出错误发生在Hadoop/Spark应用里

有些内存溢出错误发生在Hadoop/Spark系统本身，比如发生在Hadoop调度器或Hadoop的资源管理器（TaskTracker）里。

经过筛选，最后留下276个错误例子，我们发现其中123个例子的错误原因已经下面的方法诊断出来，所以我们的研究对象是这123个内存溢出错误例子。这些例子既有用户手工撰写代码生成的应用，也有依赖高层语言（如Apache Pig）和高层库（如MLlib）生成的应用，应用的详细分布如下表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **框架** | **应用来源** | **手写** | **Pig** | **Hive** | **Mahout** | **Cloud9** | **GraphX** | **MLlib** | **总数** | **重现数** |
| Hadoop | StackOverflow.com | 20 | 4 | 2 | 4 | 0 | 0 | 0 | 30 | 16 |
| Hadoop mailing list | 5 | 5 | 1 | 0 | 1 | 0 | 0 | 12 | 6 |
| Developers’ blogs | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 2 |
| MapReduce books | 8 | 3 | 0 | 0 | 0 | 0 | 0 | 11 | 2 |
| Spark | Spark mailing list | 16 | 0 | 0 | 0 | 0 | 1 | 2 | 19 | 3 |
| StackOverflow.com | 42 | 0 | 0 | 0 | 0 | 1 | 5 | 48 | 14 |
| 总数 |  | 93 | 13 | 3 | 4 | 1 | 2 | 7 | 123 | 43 |

### 错误原因与修复方法搜集

对于每一个错误例子，我们手工分析用户的错误描述信息以及专家的诊断信息。专家包含Hadoop/Spark的框架设计开发者，有经验的应用开发者或者MapReduce书籍作者，这些专家有来自Hadoop/Spark框架提供商，比如cloudera.com和databricks.com，也有来自框架用户比如ebay.com，huawei.com。我们通过下面的方法来确定错误原因：

1. 专家诊断出了错误原因，用户也接受了专家了回答，一共66个。
2. 用户自己诊断出了错误原因。用户详细原因，包括数据异常信息，不正确的配置，或者异常的用户代码逻辑。一共45个。
3. 我们自己重现了错误（重现方法见下节），手工分析诊断出了错误原因。一共12个。

修复方法类型的确定：我们发现这123错误中有42个错误有专家的修复建议或者用户修复所使用的方法。我们首先抽取相应的修复方法，然后将类似的修复归并在一起，最终得到11个修复方法类型（pattern）。

* 1. 研究结果（常见错误原因）

研究结果

* 1. 研究结果（常见的修复方法）

具体定义如表3.1所示。

* 1. 框架改进研究结果
  2. 讨论
  3. 本章小结

系统评价。

# 应用内存用量模型建立及用量估算方法

摘要

* 1. 引言

。

* 1. 问题定义
  2. 问题分析
  3. 问题解决思路
  4. 内存用量模型

### 数据流模型

### 用户代码模型

* 1. 内存用量预测方法

离线训练阶段

* 1. 实验与评价

本节

### 实验环境和设置

实验环境

### 实验对象与数据

图4.11中 。

### 进一步的讨论

上述 。

* 1. 相关工作比较

指导意义。

* 1. 本章小结

是本章重点解决的问题

# 应用内存溢出诊断方法

本文最后基于YCSB对方法的有效性进行了系统评价。

* 1. 引言

公平性问题。

本章建。

本章的组织结构如下：5.2节给出了研究背景与问题分析；5.3节给出了关键机制介绍，包括总体架构、租户缓存空间划分机制、租户敏感的缓存替换算法以及租户带宽分配机制；5.4节描述了系统实现；5.5节给出本文方法的实验及结果分析；5.6节比较了相关研究工作。5.7节对本章工作加以总结。

* 1. 问题分析

### 研究背景

* 1. 关键机制
  2. 系统实现

缓存服务 实验与评价

本节针对前文所述方法设计了一系列实验，5.5.1节主要介绍了实验环境和评价方法，5.5.2节评价了缓存性能隔离框架的有效性，5.5.3和5.5.4节分别评价了缓存空间划分和租户带宽分配机制的有效性，5.5.5节对性能开销进行了评价，5.5.6节对方法的不足给出进一步的讨论。

### 实验设置和评价方法

#### 实验设置

实验环境的详细配置

#### 评价方法

### 性能开销

### 进一步的讨论

上述实验分别评价了缓存空间划分和租户带宽分配机制的有效性。实验结果表明，*Estral*可有效消除租户间的性能干扰，同时保障租户的SLA需求得到满足。仍存在的不足有以下几点：

1. 采用缓存客户端监测服务响应时间，当应用实例数量较多时，方法的鲁棒性和扩展性存在挑战。
2. 未考虑副本因素。当节点中某个租户存在访问瓶颈时，除增加内存配额外，可考虑利用其他节点的缓存副本均衡负载。
3. 主要关注缓存服务端的资源管理，未考虑客户端VM性能干扰对响应时间可能产生的影响。
   1. 相关工作比较
   2. 本章小结

|  |  |  |  |
| --- | --- | --- | --- |
| **操作** | **可能条件** | **服务端响应** | **客户端执行的操作** |
| Get操作 | 客户端与服务端路由版本号一致；服务端状态为Stable； | 标志位(1)；服务端状态(Stable)；数据 | 无 |
| 客户端路由版本号低于与服务端；服务端状态为Stable； | 标志位(0)；服务端状态(Stable)；NULL | 1. 获取最新版本的路由表 2. 基于上述路由表进行数据请求 |
| 客户端与服务端路由版本号一致；服务端状态为Unstable； | 标志位(1)；服务端状态(Unstable)；数据 | 无 |
| 客户端路由版本号低于服务端；服务端状态为Unstable； | 标志位(0)；服务端状态(Unstable)；NULL | 根据上一版本的路由表请求数据； |
| 标志位(0)；服务端状态(Unstable)；数据 | 无 |
| Update操作 | 客户端与服务端路由版本号一致；服务端状态为Stable； | 标志位(1)；服务端状态(Stable)；执行更新 | 无 |
| 客户端路由版本号低于与服务端；服务端状态为Stable； | 标志位(0)；服务端状态(Stable)；不执行更新 | 1. 获取最新版本的路由表 2. 基于上述路由表进行数据更新 |
| 客户端与服务端路由版本号一致；服务端状态为Unstable； | 标志位(1)；服务端状态(Unstable)；不执行更新 | 1. 根据上一版本的路由表进行更新 2. 根据最新路由表再次进行更新 |
| 客户端路由版本号低于服务端；服务端状态为Unstable； | 标志位(0)；服务端状态(Unstable)；不执行更新 | 1. 获取最新版本的路由表 2. 基于上述路由表进行数据更新 |

* 1. 本章小结

本章对缓存服务支撑系统*ElastiCamel*进行了简要的分析和阐述，并给出了系统的功能模块划分和总体架构设计，该系统由缓存服务器、节点管理器、主节点和缓存客户端四部分组成，本章依次对上述四部分的设计与实现进行了介绍，最后，阐述了数据迁移机制的原理和技术实现。

# 结束语

本章系统总结了本论文的主要贡献，包括：

* 1. 论文工作总结
  2. 进一步的工作

# 参考文献

[

[ACE] Aliyun Cloud Engine (ACE). http://www.aliyun.com/product/ace/

[Amazon 2011] Amazon ElastiCache. 2011. http://aws.amazon.com/elasticache/

[Ari 2002] Ari I, Amer A, Miller EL, Brandt SA, Long DE. Who is more adaptive? ACME: adaptive caching using multiple experts. In：Proc. of Workshop on Distributed Data and Structures (WDAS '02). 2002.

[Amza 2005] Amza C, Cox AL, Zwaenepoel W. A Comparative Evaluation of Transparent Scaling Techniques for Dynamic Content Servers. In：Proc. of the 21st Int’l Conference on Data Engineering (ICDE '05). 2005. 230-241.

[Bhide 2002] Bhide M, Deolasee P, Katkar A, Panchbudhe A, Ramamritham K, Shenoy P. Adaptive Push-Pull: Disseminating Dynamic Web Data. IEEE Trans. on Computers, 51 (6):652-668, 2002.

[Blum 1998] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In：Proc. of the 11th annual conf. on Computational learning theory. 92-100. 1998.

[Brewer 2000] Brewer EA. Towards robust distributed systems. In: Proc. of the 19th Annual ACM Symp. on Principles of Distributed Computing (PODC 2000). 2000.

[Burnham 2002] Burnham KP, Anderson DR. Model selection and multi-model inference: a practical information-theoretic approach. 2nd ed., New York: Springer-Verlag, 2002.

[Chen 2009] 陈康, 郑纬民. 云计算:系统实例与研究现状. 软件学报. 2009, 20 (5): 1337-1348.

[Chiu 2010] Chiu D, Shetty A, Agrawal G. Elastic Cloud Caches for Accelerating Service-Oriented Computations. In：Proc. of the ACM/IEEE Int’l Conf. for High Performance Computing, Networking, Storage and Analysis (SC '10). 2010. 1-11.

[Chockler 2010] Chockler G, Laden G, Vigfusson Y. Data caching as a cloud service. In：Proc. of the 4th Int’l Workshop on Large Scale Distributed Systems and Middleware (LADIS '10). 2010.

[Choi 2002] Choi J, Noh SH, Min SL, Ha EY, Cho YK. Design, Implementation, and Performance Evaluation of a Detection-Based Adaptive Block Replacement Scheme. IEEE Trans. on Computers, 51(7): 793-800, 2002.

[Chong 2008] Chong EKP, Zak SH. An Introduction to Optimization, 3rd ed., New Jersey: John Wiley & Sons Inc., 2008.

[Cloud\_computing 2007] Cloud computing. Wikipedia. 2007. http://en.wikipedia.org/ wiki/Cloud\_computing

[Cohen 1995] Cohen WW. Fast effective rule induction. In：Proc. of the 12th Int’l Conf. on Machine Learning (ICML '95). 115–123. 1995.

[Cooper 2010] Cooper BF, Silberstein A, Tam E, Ramakrishnan R, Sears R. Benchmarking Cloud Serving Systems with YCSB. In：Proc. of ACM Symp. on Cloud Computing (SoCC '10). 2010. 143-154.

[Couceiro 2011] Couceiro M, Romano P, Rodrigues L. PolyCert: Polymorphic Self-Optimizing Replication for In-Memory Transactional Grids. In：Proc. of ACM/IFIP/USENIX 12th Int’l Middleware Conference (Middleware '11). 2011. 309-328.

[Couchbase] Multitenancy with couchbase. Couchbase. http://www.couchbase.com /docs/couchbase-manual-1.8/couchbase-introduction-architecture-buckets. html

[Das 2011] Das S, Nishimura S, Agrawal D, Abbadi AE. Albatross: Lightweight Elasticity in Shared Storage Databases for the Cloud using Live Data Migration. In：Proc. of the 37th Int’l Conf. on Very Large Data Bases (VLDB '11). 2011. 494-505.

[Dell 2005] Dell white paper. An Overview of Xen Virtualization. http://www. dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf. 2005

[Deolasee 2001] Deolasee P, Katkar A, Panchbudhe A, Ramamritham K, Shenoy P. Adaptive Push-Pull: Disseminating Dynamic Web Data. In：Proc. of Int’l World Wide Web Conference (WWW '01). 2001. 265-274.

[Draper 1998] Draper NR, Smith H. Applied Regression Analysis. 3rd ed., New York: John Wiley & Sons Inc., 1998.

[Elmore 2011] Elmore AJ, Das S, Agrawal D, Abbadi AE. Zephyr: Live Migration in Shared Nothing Databases for Elastic Cloud Platforms. In：Proc. of the annual ACM SIGMOD Conference (SIGMOD '11). 2011. 301-312.

[Extreme\_Transaction\_Processing 2012] Extreme Transaction Processing. Wikipedia. 2012. http://en.wikipedia.org/wiki/Extreme\_Transaction \_Processing

[Earls 2010] Earls A. Distributed data grids: Foundation for future cloud computing? 2010. http://searchsoa.techtarget.com/news/1518647/Data-Grids- Foundation-for-future-cloud-computing

[Feng 2008] Feng D, Lu C, Wang F. eMuse: QoS Guarantees for Shared Storage Servers. In：Proc. of the 22nd Int’l Conf. on Advanced Information Networking and Applications Workshops (AINAW '08). 2008. 264-269.

[Fetai 2012] Fetai I, Schuldt H. Cost-Based adaptive concurrency control in the cloud. Technical Report, CS-2012-001, Basel: The University of Basel, 2012.

[Gartner 2010] Fenn J. Hype cycle for emerging technologies. Gartner Report, 2010. http://www.planetlarg.net/my-scripts/docs-to-read/gartner/hype\_cycle\_for\_emerging\_tech\_2010.pdf

[GigaSpaces 2009] Security concepts. Gigaspaces XAP. 2009. http://www.gigaspaces.com/ wiki/display/XAP7/Security+Concepts

[GigaSpaces 2010] Making dynamic scaling simple. GigaSpaces XAP. 2010. http://www.gigaspaces.com/xap7-1

[Gilbert 2002] Gilbert S, Lynch N. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant Web services. ACM SIGACT News, 2002, 33(2).

[Gold 2011] Gold E. Extreme transaction processing. 2011. http://natishalom.typepad.com/nati\_shaloms\_blog/files/gigaspaces-dream -machine.pdf

[Gossip 2007] Gossip. Wikipedia. 2007. http://en.wikipedia.org/wiki/Gossip\_protocol

[Gramacy 2003] Gramacy R, Warmuth M, Brandt SA, Ari I. Adaptive caching by refetching. Advances in Neural Information Processing Systems 15. 2003. 1465–1472.

[Gualtieri 2010] Gualtieri M, Rymer JR. The forrester wave: Elastic caching platforms. Q2, 2010. ftp://ftp.software.ibm.com/software/solutions/soa/pdfs/wave\_ elastic\_caching\_ platforms \_q2\_2010.pdf

[Gualtieri 2010\*] Gualtieri M. Elastic caching platforms balance performance, scalability and fault tolerance. 2010. http://blogs.forrester.com/mike\_ gualtieri/10-03-18-elastic\_caching\_platforms\_balance\_performance\_scalability\_and\_fault\_toleranc

[Gulati 2009] Gulati A, Ahmad I, Waldspurger C. PARDA: Proportional Allocation of Resources for Distributed Storage Access. In Proc. of USENIX Conference on File and Storage Technologies (FAST '09). 2009. 85-98.

[Gupta 2005] Gupta D, Gardner R, Cherkasova L. XenMon: QoS Monitoring and Performance Profiling Tool. HP Technical Report (HPL-2005-187). 2005.

[Gupta 2006] Gupta D, Cherkasova L, Gardner R, Vahdat A. Enforcing Performance Isolation Across Virtual Machines in Xen. In：Proc. of the 7th ACM/IFIP/USENIX Middleware Conference (Middleware '06). 2006. 342-362.

[Hahn 2008] Hahn S, Ostendorf M. A Comparison of Discriminative EM-Based Semi-Supervised Learning algorithms on Agreement/Disagreement Classification. In：Proc. of NIPS Workshop on Speech and Language: Learning-based Methods and Systems. 2008.

[Hall 2003] Hall MA, Holmes G. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. IEEE Trans. on Knowledge and Data Engineering, 15 (6):1437-1447, 2003.

[Hall 1999] Hall MA. Correlation-based Feature Selection for Machine Learning. [PhD thesis]. University of Waikato, 1999.

[Hastorun 2007] Hastorun D, Jampani M, Kakulapati G, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: Amazon’s highly available key-value store. In：Proc. of ACM Symp. on Operating Systems Principles (SOSP '07). 2007. 205-220.

[Hwang 2013] Hwang J, Wood T. Adaptive Performance-Aware Distributed Memory Caching. In：Proc. of Int’l Conf. on Autonomic Computing (ICAC '13). 2013. 33-43.

[IBM 2004] IBM Websphere extreme scale. 2004. http://www.redbooks.ibm.com/ redbooks/SG247683/wwhelp/wwhimpl/js/html/wwhelp.htm

[IBM 2006] IBM white paper. An architectural blueprint for autonomic computing. 2006.

[Ingenthron 2012] Ingenthron M. Growing data sets beyond memory. 2012. http://www.couchbase.org/wiki/display/membase/Growing+Data+Sets+ Beyond+Memory

[Johnson 2007] Johnson RA, Wichern DW. Applied multivariate statistical analysis. 6th ed., New Jersey: Prentice Hall Inc., 2007.

[Jroups 2002] Jroups. 2002. http://www.jgroups.org/overview.html

[JSR-107 2001] JSR-107. 2001. http://jcp.org/en/jsr/detail?id=107

[JSR-342 2011] JSR-342. 2011. http://jcp.org/en/jsr/detail?id=342

[JSR-347 2011] JSR-347. 2011. http://jcp.org/en/jsr/detail?id=347

[Karger 1997] Karger D, Lehman E, Leighton T, Panigrahy R, Levine M, Lewin D. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In：Proc. of the 29th Annual ACM Symp. on theory of Computing (STOC '97). 1997. 654-663.

[Kari 2011] Kari C, Kim Y, Russell A. Data Migration in Heterogeneous Storage Systems. In：Proc. of the 31st Int’l Conf. on Distributed Computing Systems (ICDCS '11). 2011. 153-160.

[Khan] Khan I. Using read-through & write-through in distributed cache. http://www.alachisoft.com/resources/articles/readthru-writethru-writebehind.html

[Kira 1992] K. Kira and L. Rendell. A Practical Approach to Feature Selection. In：Proc. of the 9th Int’l workshop on Machine learning. 249-256. 1992.

[Kononenko 1994] Kononenko I. Estimating Attributes: Analysis and Extensions of Relief. In：Proc. of 7th European Conf. on Machine Learning (ECML ' 94). 171-182. 1994.

[Kundu 2010] Kundu S, Rangaswami R, Dutta K, Zhao M. Application Performance Modeling in a Virtualized Environment. In：Proc. of IEEE 16th Int’l Symp. on High Performance Computer Architecture (HPCA '10). 2010. 1-10.

[Kunkle 2008] Kunkle D, Schindler J. A load balancing framework for clustered storage systems. In：Proc. of 15th Int’l Conf. on High Performance Computing (HiPC '08). 2008. 57-72.

[Lim 2010] Lim HC, Babu S, Chase JS. Automated control for elastic storage. In：Proc. of the 7th Int’l Conf. on Autonomic computing (ICAC '10). 2010. 1-10.

[Lin 2010] 林海略,韩燕波.多租户应用的性能管理关键问题研究.计算机学报,2010,33(10):1881-1895.

[Liu 1996] Liu H, Setiono R. A probabilistic approach to feature selection - A filter solution. In：Proc. of the 13th Int’l Conf. on Machine Learning (ICML '96). 319-327, 1996.

[Lu 2001] Lu Y, Saxena A, Abdelzaher TF. Differentiated caching services: a control-theoretical approach. In：Proc. of the 21st Int’l Conf. on Distributed Computing Systems (ICDCS '01). 2001. 615-622.

[Lu 2002] Lu C, Alvarez GA, Wilkes J. Aqueduct: online data migration with performance guarantees. In：Proc. of USENIX Conf. on File and Storage Technologies (FAST '02). 2002. 219-230.

[Lu 2004] Lu Y, Abdelzaher TF, Saxena A. Design, Implementation, and Evaluation of Differentiated Caching Services. IEEE Trans. on Parallel and Distributed Systems, 15(5):440-452, 2004.

[Machina 2009] Machina J, Sodan A. Predicting cache needs and cache sensitivity for applications in cloud computing on CMP servers with configurable caches. In：Proc. of the IEEE Int’l Symposium on Parallel & Distributed Processing (IPDPS '09). 2009. 1-8.

[Megiddo 2003] Megiddo N, Modha DS. Arc: A self-tuning, low overhead replacement cache. In：Proc. of the 2nd USENIX Conf. on File and Storage Technologies (FAST '03). 2003. 115-130.

[Mei 2013] Mei Y, Liu L, Pu X, Sivathanu S, Dong X. Performance analysis of network I/O workloads in virtualized data centers. IEEE Trans. on Service Computing, 6(1):48-63, 2013.

[Memcached 2003] Memcached. Wikipedia. 2003.

http://en.wikipedia.org/wiki/ Memcached

[Memcached\_FAQ 2011] Memcached FAQ. 2011.

http://code.google.com/p/ memcached/wiki/NewStart

[Menon 2006] Menon A, Cox AL, Zwaenepoel W. Optimizing Network Virtualization in Xen. In：Proc. of USENIX Annual Technical Conference (USENIX ATC '06). 2006. 15-28.

[Menascé 1999] Menascé DA, Almeida VAF, Fonseca R, Mendes MA. A Methodology for Workload Characterization of E-commerce Sites. In：Proc. of the 1999 ACM Conf. on Electronic Commerce (E-COMMERCE ' 99). 1999.

[Microsoft 2011] Programming model. Windows Server AppFabric Caching. 2011. http://msdn.microsoft.com/en-us/library/hh334298.aspx

[Microsoft 2011\*] Security model. Windows Server AppFabric Caching. 2011. http://msdn.microsoft.com/en-us/library/ff718179.aspx

[Moore 2003] Moore AW. Information Gain. 2003.

http://www.autonlab.org/tutorials/ infogain11.pdf

[Multicollinearity 2005] Multicollinearity. Wikipedia. 2005.

http://en.wikipedia.org/ wiki/Multicollinearity

[Multitenancy 2008] Multitenancy. Wikipedia. 2008.

http://en.wikipedia.org/wiki/ Multitenancy

[NCache 2008] NCache: Caching topologies. 2008.

http://www.alachisoft.com/ ncache/caching-topology.html

[Neter 1985] Neter J, Waserman W, Kutner M. Applied Linear Statistical Models: Regression Analysis of Variance and Experimental Designs, 3rd ed., New York: McGraw Hill, 1985.

[Nishtala 2013] Nishtala R, Fugal H, Grimm S, Kwiatkowski M, Lee H, Li HC, McElroy R, Paleczny M, Peek D, Saab P, Stafford D, Tung T, Venkataramani V. Scaling Memcache at Facebook. In：Proc. of 10th USENIX Symp. on Networked Systems Design and Implementation (NSDI '13). 2013. 385-398.

[Nori 2010] Nori AK. Distributed caching platforms. In: Proc. of the 36th Int’l Conf. on Very Large Data Bases (VLDB 2010). 2010. 1645-1646.

[NoSQL 2009] NoSQL . Wikipedia. 2009. http://en.wikipedia.org/wiki/NoSQL.

[Oracle 2009] Platform-as-a-Service private cloud with oracle fusion middleware. Oracle White Paper, 2009. http://www.oracle.com/us/technologies/ cloud/036500.pdf

[Oracle 2011] Caching data sources. Oracle. 2011. http://download.oracle.com/docs/ cd/E24290\_01/coh.371/e22837/cache\_rtwtwbra.htm#CFHEDIGA

[Patrick 2009] Patrick CM, Garg R, Son SW, Kandemir M. Improving I/O Performance using Soft-QoS Based Dynamic Storage Cache Partitioning. In：Proc. of 11th IEEE Int’l Conf. on Cluster (Cluster '09). 2009. 1-10.

[Peralta 2007] Peralta P. Successfully scaling Java applications in spring. Oracle Corp. 2007. http://www.nejug.org/events/download?f=41

[Pfaffhauser 2010] Pfaffhauser F. Scaling a Cloud Storage System Autonomously [Master Thesis]. ETH. 2010.

[Pierre 1998] Pierre G, Makpangou M. Saperlipopette!:a distributed Web caching systems evaluation tool. In：Proc. of ACM/IFIP/USENIX 1st Int’l Middleware Conference (Middleware '98). 389-405. 1998.

[Pierre 2001] Pierre G, Kuz L, Steen M, Tanenbaum AS. Differentiated Strategies for replicating Web Documents. Computer Communications. 232-240. 2001.

[Pierre 2002] Pierre G, van Steen M, Tanenbaum AS. Dynamically selecting optimal distribution strategies for Web documents. IEEE Trans. on Computers, 51(6):637-651, 2002.

[Prabhakar 2010] Prabhakar R, Srikantaiah S, Kandemir M, Patrick CM, Kandemir M. Adaptive Multi-Level Cache Allocation in Distributed Storage Architectures. In：Proc. of the 24th ACM Int’l Conf. of Supercomputing (ICS '10). 2010. 211-221.

[Prabhakar 2009] Prabhakar R, Srikantaiah S, Patrick CM, Kandemir M. Dynamic Storage Cache Allocation in Multi-Server Architectures. In：Proc. of the ACM/IEEE Int’l Conf. for High Performance Computing, Networking, Storage, and Analysis (SC '09). 2009. 1-12.

[Pu 2010] Pu X, Liu L, Mei Y, Sivathanu S, Koh Y, Pu C. Understanding Performance Interference of I/O Workload in Virtualized Cloud Environment. In：Proc. of the 3rd IEEE Int’l Conf. on Cloud Computing (Cloud '10). 2010. 51-58.

[Pohl 2005] Pohl C. Adaptive caching of distributed components. [PhD Thesis]. TU Dresden, 2005.

[Qin 2011] Qin X, Zhang W, Wang W, Wei J, Zhong H, Huang T. Online Cache Strategy Reconfiguration for Elastic Caching Platform: A Machine Learning Approach. In：Proc. of 35th Annual IEEE Int’l Computer Software and Applications Conference (COMPSAC '11). 2011. 523-534.

[Qin 2013] 秦秀磊, 张文博, 魏峻, 王伟, 钟华, 黄涛. 云计算环境下分布式缓存技术的现状与挑战.软件学报,2013,24(1):50-66.

[Rosa 2011] Rosa L, Rodrigues L, Lopes A. Goal-Oriented Self-management of In-memory Distributed Data Grid Platforms. In Proceedings of the 3rd IEEE Int’l Conf. on Cloud Computing Technology and Science (CloudCom ’11). 2011. 587-591.

[SAE] Sina App Engine (SAE). http://sae.sina.com.cn/

[Salehie 2009] Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 4(2): 42 pages, 2009.

[Schubert 2010] Schubert L, Jeffery K, N-Lutz B. The future of cloud computing: Opportunities for European cloud computing beyond 2010. 2010. http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf

[Selvakuberan 2008] Selvakuberan K, Indradevi M, Rajaram DR. Combined Feature Selection and classification-A novel approach for the categorization of web pages. Journal of Information and Computing Science, 3(2):83-89, 2008.

[Shen 2007] Shen H, Xu C. Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured Peer-to-Peer Networks. IEEE Trans. on Parallel and Distributed Systems, 18(6): 849-862, 2007.

[Shue 2012] Shue D, Freedman MJ, Shaikh A. Performance isolation and fairness for multi-tenant cloud storage. In：Proc. of the 10th USENIX conference on Operating Systems Design and Implementation (OSDI '12). 2012. 349-362.

[Sivasubramanian 2003] Sivasubramanian S, Pierre G, Steen MV. A case for dynamic selection of replication and caching strategies. In：Proc. of the 8th workshop on Web Caching and Content Distribution. 2003. 275-282.

[Subramanian 2006] Subramanian R, Smaragdakis Y, Loh GH. Adaptive Caches: Effective Shaping of Cache Behavior to Workloads. In：Proc. of the 39th Annual IEEE/ACM Int’l Symp. on Micro architecture (MICRO '06). pp. 385-396, 2006.

[Terracotta 2009] Terracotta server arrays architecture. 2009. http://64.95.112.233/ documentation/ terracotta-server-array/server-arrays

[Terracotta 2011] Terracotta DSO documentation. 2011.

http://www.terracotta.org/ confluence/display/ docs/Home

[Tesauro 2007] Tesauro G. Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies. IEEE Internet Computing, 11(1):22-30, 2007.

[Thaler 1998] Thaler D, Ravishankar C. Using name-based mappings to increase hit rates. IEEE/ACM Trans. on Networking, 6(1):1–14, 1998.

[Totok 2006] Totok A, Karamcheti V. Improving Performance of Internet Services through Reward-Driven Request Prioritization. In：Proc. of 14th IEEE Int’l Workshop on Quality of Service (IWQoS '06). 2006. 60-71.

[TPC-W] TPC-W. http://www.tpc.org/tpcw/default.asp.

[Trushkowsky 2011] Trushkowsky B, Bodik P, Fox A. The SCADS Director: Scaling a Distributed Storage System under Stringent Performance Requirements. In：Proc. of USENIX Conf. on File and Storage Technologies (FAST '11). 2011. 163-176.

[Wang 1999] Wang HW. Partial least-squares regression method and applications. Beijing: National Defense Industry Press, 1999.

[Wei 2012] Wei Z, Pierre G, Chi C. Scalable join queries in cloud data stores. In: Proc. of the 12th IEEE/ACM Int’l Symp. on Cluster, Cloud and Grid Computing (CCGrid 2012). 2012. 547-555.

[Weinberger 2007] Weinberger KQ, Tesauro G. Metric Learning for Kernel Regression. Journal of Machine Learning Research, 2007, 2: 612-619.

[WEKA] WEKA. http://www.cs.waikato.ac.nz/ml/weka.

[Whittingham 2006] Whittingham MJ, Stephens PA, Bradbury RB, Freckleton RP. Why do we still use stepwise modeling in ecology and behaviour? Journal of Animal Ecology, 2006, 75(5): 1182–1189.

[Wiggins 2012] Wiggins A, Langston J. Enhancing the Scalability of Memcached. Intel Technical Report. http://download-software.intel.com/sites/default /files/m/0/b/6/1/d/45675-memcached\_05172012.pdf

[Zhang 2011] Zhang W, Wang S, Wang W, Zhong H. Bench4Q: A QoS-Oriented E-Commerce Benchmark. In：Proc. of 35th Annual IEEE Int’l Computer Software and Applications Conference (COMPSAC '11). 2011. 38-47.

[Zhu 2008] Zhu X. Semi-Supervised Learning Literature Survey. University of Wisconsin-Madison, technical report. http://pages.cs.wisc.edu/~jerryzhu/ pub/ssl\_survey.pdf. 2008.

[Zhu 2012] 朱鑫. 面向多租户的分布式缓存系统设计与实现. [硕士论文]. 中国科学院研究生院, 2012.

发表文章目录

1. **Xiulei Qin**, Wei Wang, Wenbo Zhang, Jun Wei, Xin Zhao, Hua Zhong, Tao Huang. *PRESC2*: Efficient Self-Reconfiguration of Cache Strategies for Elastic Caching Platforms. Computing Journal, Springer (SCI indexed, IF: 0.807).
2. **秦秀磊**, 张文博, 魏峻, 王伟, 钟华, 黄涛. 云计算环境下分布式缓存技术的现状与挑战. 软件学报. 2013, 24(1):50-66.
3. **秦秀磊**, 张文博, 王伟, 魏峻, 赵鑫, 钟华, 黄涛. 面向云端key/value存储系统的开销敏感的数据迁移方法. 软件学报. 2013, 24(6):1403-1417.
4. **Xiulei Qin**, Wenbo Zhang, Wei Wang, Jun Wei, Xin Zhao, Tao Huang. Towards a Cost-Aware Data Migration Approach for Key-Value Stores. In Proceeding of the 14th IEEE International Conference on Cluster Computing (*Cluster 2012*), September 24-28, 2012, Beijing. pp.551~556.
5. **Xiulei Qin**, Wenbo Zhang, Wei Wang, Jun Wei, Xin Zhao, Tao Huang. Optimizing Data Migration for Cloud-based Key-Value Stores. In Proceeding of the 21st ACM International Conference on Information and Knowledge Management (*CIKM 2012*), October 29-November 2, 2012, Maui, USA. pp.2204~2208.
6. **Xiulei Qin**, Wei Wang, Wenbo Zhang, Jun Wei, Xin Zhao, Tao Huang. *ElastiCat*: A load rebalancing framework for cloud-based key-value stores. In Proceedings of 19th IEEE International Conference on High Performance Computing (*HiPC 2012*), December 18-21, 2012, Pune, India. pp.1~10.
7. **Xiulei Qin**, Wenbo Zhang, Wei Wang, Jun Wei, Hua Zhong, Tao Huang. A Comparative Evaluation of Cache Strategies for Elastic Caching Platforms. In Proceedings of 11th International Conference on Quality Software (*QSIC 2011*), July 13-14, 2011, Madrid, Spain. pp.166~175.
8. **Xiulei Qin**, Wenbo Zhang, Wei Wang, Jun Wei, Hua Zhong, Tao Huang. Online Cache Strategy Reconfiguration for Elastic Caching Platform: A Machine Learning Approach. In Proceeding of the 35th Annual IEEE International Computer Software and Applications Conference (*COMPSAC 2011*), July 18-22, 2011, Munich, Germany. pp.523~534.
9. **Xiulei Qin**, Jun Wei, Wenbo Zhang, Hua Zhong, Tao Huang. A Two-Phase Approach to Subscription Subsumption Checking for Content-Based Publish/Subscribe Systems. In Proceeding of the 24th IEEE International Conference on Advanced Information Networking and Applications (*AINA 2010*), April 20-23, 2010, Perth, Australia. pp.1278~1285.
10. Wei Wang, Xiang Huang, **Xiulei Qin**, Wenbo Zhang, Jun Wei, Hua Zhong. Application-level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications. In Proceeding of IEEE 2012 International Conference on Cloud Computing (*CLOUD 2012*), June 24-29, 2012, Hawaii, USA. pp.439~446.
11. 朱鑫, 蒲卫, **秦秀磊**, 张文博, 钟华. 上下文感知的分布式缓存数据重均衡方法. 计算机工程与设计. 2013. 34(1): 207~314.
12. 朱鑫, **秦秀磊**, 王联华, 张文博, 钟华. 弹性分布式缓存动态扩展方法研究. 计算机科学与探索. 2012. 6(2): 97~108.
13. 朱鑫, **秦秀磊**, 王联华, 张文博, 钟华. 弹性分布式缓存动态扩展方法研究. 2011全国软件与应用学术会议 (NASAC 2011), 长春, 2011.
14. **秦秀磊**, 张文博. 面向云计算的弹性缓存平台及其若干关键问题. 四方国件(核高基期刊). 2011(2): 64-74.

参与科研项目目录

1. 国家自然科学基金, "面向多承租的弹性缓存服务关键技术研究" (批准号：61173003)
2. 国家高技术研究发展计划(863), "网构化软件运行支撑和在线管理技术与机制" (批准号：2012AA011204)
3. 核高基重大专项6-2, "集成化中间件套件服务质量保障关键技术与系统研发" (批准号：2009ZX01043-002-003-02)
4. 核高基重大专项6-3, "领域应用平台关键技术研究及国产中间件适配平台开发" (批准号：2009ZX01043-003-002-02)
5. 国家科技支撑计划, "技术交易全程服务支撑关键技术研究和平台设计" (批准号：2012BAH09F01)
6. 核高基重大专项8-1, "智能数字电视终端基础软件研发及产业化"
7. 中国科学院监察审计局，"信访管理系统"

获得发明专利目录

1. “一种支持负载均衡的分布式缓存动态伸缩方法及系统”，专利号ZL 201110230333.X 发明人：黄涛, **秦秀磊**, 张文博, 魏峻, 钟华, 朱鑫
2. “基于机器学习的分布式缓存策略自适应切换方法及系统”，专利号 ZL 201110167018.7 发明人：张文博, **秦秀磊**, 王伟, 魏峻, 钟华, 黄涛

获得软件著作权目录

1. “Once分布式缓存软件V1.0”，登记号2011SR016760

致 谢

博士五年多的学习生涯是一段既艰苦又令人激动的时光，这段经历令我受益终生。首先要衷心感谢我的导师黄涛研究员，黄老师给我创造了优越的学习与科研环境，使我能够参与到包括自然科学基金、863计划和核高基在内的多项重要的国家级课题中，实践能力和科研能力都得到了很好的锻炼。黄老师对我的论文从选题、研究到写作等各个方面都提出了严格的要求，并给予了悉心的指导和耐心的帮助。黄老师卓越的学术素养，渊博的学识，高尚的道德品格，敏锐深邃的洞察力和思考力，高屋建瓴的战略眼光都让我受益匪浅。“师生一场是缘分”，黄老师2011年对我讲的这句话今天仍言犹在耳，导师对我的影响是深刻而全方位的，体现在做人做事做学问等各个方面。有幸成为黄老师的学生，是我人生中最重要的收获之一。

其次要特别感谢魏峻研究员，在攻读博士学位期间，魏老师对我的研究工作给予了耐心细致的指导，提出了许多富有建设性的宝贵意见和建议，尤其在每个关键节点，都给予了我最无私的支持。魏老师对博士生讨论班的热情投入和高效组织，使我获益良多，同时也是我个人学术成长的一个重要基点；我对如何做科研、如何做好科研有了更加深刻的认识，同时也体味到了沟通对科研的重要意义。魏老师学识渊博，治学态度严谨，具有高度的工作责任感，敏锐的学术洞察力、鉴赏力以及深刻的思考力，幽默健谈，对学生真诚奉献，从魏老师这里，我体会到了永葆激情与梦想的价值所在。

感谢数据网格组组长王伟副研究员，对我而言，他不仅仅是一位项目组长，更是一位待人和善、乐于助人的师兄，一位开朗热情、谦虚正直的朋友，一位有着阅历丰富、随时准备和你侃侃而谈的饭友，一位保持微笑的永恒的合作者。王伟博士心胸宽阔、平易近人、追求品质与完美、具有良好的沟通表达能力、宽阔的知识面、极强的工作责任心以及优秀的组织和管理能力，一直是我和很多同学学习的榜样。

感谢平台组组长张文博副研究员，在平台组的这段日子里，无论是科研工作、项目组织实施还是生活的方方面面，张老师都给予了我本人许多关心和帮助，内心的感动无以言表。张老师敏锐的技术洞察力、认真负责和勤奋忘我的工作态度值得我去学习。

诚挚的感谢软件工程技术中心的其他各位老师。感谢钟华研究员对我工作，学习给予的许多建议和指导，钟老师学识渊博、工作严谨、待学生如朋友、亲切友善令我印象深刻。还要感谢冯玉琳研究员、金蓓弘研究员、叶丹研究员、徐罡副研究员、吴国全副研究员、刘杰、明路、刘玲玲等各位老师，我博士阶段的学习与工作离不开他们的支持与帮助。

特别感谢硕士期间的导师孙波教授，每当科研遇到瓶颈，踌躇不前时，总能得到孙老师最诚挚的鼓励和最坚定有力的支持，这成为我人生中的一笔宝贵的精神财富，支撑我秉持信念，正视困难，勇于迎接新的人生挑战。

感谢数据网格组的全体成员，他们是赵鑫、李萱、纪树平、王彦士、刘朝晖、杨鑫晟、唐震、陈铁南、王晓冉、支孟轩、任凯等。大家平日里的热情大方、勤勉努力、朴实真诚、尽职尽责以及良好的团队合作精神深深的感染着我。也要感谢平台组宋云奎、段智全、周欢云、赵占平、胡振碧、罗涛等诸位同学，感谢他们在科研工作方面给予我的支持和帮助。博士一年级所在项目组马志柔、吴怀林、魏克刚、张更欣和詹孟粮等老师和同学，同样给予了我许多启示，在此一并表示感谢。特别感谢为本文原型系统做出重要贡献的罗嵘、朱鑫两位师弟，大家彼此配合默契，一起见证了从*Idea*到*Impacts*的每一步转变，讨论争执中频频擦出灵感的火花。

感谢陪我一起走过博士生涯的各位同学，他们是陈伟、黄翔、张若定、王焘、高楚舒、白琳、吴恒、伍晓泉、张建华、孙耀、张利锋、徐继伟、窦文生、王卅、许利杰、邵小哲、陈三川、周晓炜、吴东尧、严慧、王帅、伍海江、朱锋、田斐、汪静甜、何海、高强、张晓杰等，感谢他们在科研工作、博士学习及生活中给我的建议和帮助。同时也要感谢并行实验室的室友袁良和颜深根同学，和他们一起度过了5年难忘的寝室岁月。

最后要感谢父母对我多年的抚育之恩以及无私的奉献与关爱。在我攻读学位期间，他们始终是我坚强的后盾和精神上的支柱，不断鞭策鼓励我乐观向上、勇于进取。几年下来，父母每次电话都会关心我的饮食、睡眠和身体情况。谁言寸草心，报得三春晖。在今后的日子里，我唯有用不懈的努力和优异的成绩才能回报父母。

再次感谢所有曾给予我关心和帮助的师长、亲人和朋友，碍于篇幅所限，无法一一表达谢意。

1. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, NSDI 2012 [↑](#endnote-ref-1)