目录

[1 绪论 4](#_Toc476929379)

[1.1 研究背景 4](#_Toc476929380)

[1.2 研究内容 5](#_Toc476929381)

[1.3 论文组织 6](#_Toc476929382)

[2 大数据系统及应用可靠性相关工作 7](#_Toc476929383)

[2.1 大数据系统及应用 7](#_Toc476929384)

[2.1.1 大数据系统 7](#_Toc476929385)

[2.1.2 大数据应用 10](#_Toc476929386)

[2.2 可靠性问题分析 10](#_Toc476929387)

[2.3 测试基准框架现状 11](#_Toc476929388)

[3 可靠性测试基准关键技术 12](#_Toc476929389)

[3.1 可靠性测试基准系统架构 12](#_Toc476929390)

[3.2 典型应用构造 12](#_Toc476929391)

[3.3 数据生成方法 12](#_Toc476929392)

[3.4 参数组合测试方法 12](#_Toc476929393)

[3.5 流式负载生成方法 12](#_Toc476929394)

[4 大数据系统应用可靠性测试基准框架实现 13](#_Toc476929395)

[4.1 系统架构 13](#_Toc476929396)

[4.2 系统模块实现 13](#_Toc476929397)

[4.2.1 可靠性基准核心模块实现 13](#_Toc476929398)

[4.2.2 典型应用实现模块 13](#_Toc476929399)

[4.2.3 数据生成器 13](#_Toc476929400)

[4.2.4 组合参数发生器 13](#_Toc476929401)

[4.2.5 系统监测器 13](#_Toc476929402)

[4.2.6 报告生成器 13](#_Toc476929403)

[4.3 应用验证 13](#_Toc476929404)

[4.3.1 实验环境 13](#_Toc476929405)

[4.3.2 大数据系统应用可靠性测试实例及分析 13](#_Toc476929406)

[5 结束语 14](#_Toc476929407)

[5.1 论文贡献 14](#_Toc476929408)

[5.2 未来工作展望 14](#_Toc476929409)

[参考文献 16](#_Toc476929410)

[发表文章 17](#_Toc476929411)

[致谢 19](#_Toc476929412)

# 绪论

本章介绍论文的研究背景、主要工作和论文组织方式。

## 研究背景

随着互联网、移动互联网、物联网的发展，越来越多的领域产生了“海量”和“高速”的数据[1]。例如，在金融领域，其日常运营过程中会产生大量的数据，这些数据产生速度快，并且时效性短[2]。又如，在移动通信领域，一个大型城市中每分钟都会有超过8万条的位置更新数据，每天的网络承载流量高达100TB[1]。这些数据具有数据规模大，产生速度快的特点，而且在这些数据中也隐藏着巨大的价值。在金融领域，通过对这些海量数据进行计算和分析，发现隐藏在其中的特征，可以帮助金融行业进行实时决策，从而更好的进行风险管理以及实现商业智能化。在移动通信领域，如果可以对这些海量数据进行实时的挖掘分析，可以降低电话诈骗造成的损失。因此，对大数据应用场景下的海量数据的处理分析已经成为一个迫切的需求。

为了应对海量数据产生以及发现其背后隐藏的巨大价值，许多大数据系统及大数据应用应运而生。常见的大数据系统有MapReduce[]、Storm[3]、Hadoop[4]、Spark[5]、Flink[6]等，这些分布式的大数据处理框架已被广泛应用到了工业界中的大数据应用的处理分析中。常见的大数据应用有SQL操作（如，Select、Aggregate等）、大规模图分析（如，PageRank、ConnectedComponents等）、流式数据处理（如，多个流的join操作等）以及机器学习（如，Kmeans、SVM等），这些应用针对不同的大数据场景提供了解决方案。这些系统针对大数据处理给出了自己的解决方案，可以应对大数据所呈现的易失性、突发性、无序性等特点，但是仍然存在着许多关键性的问题。应用在处理大数据时，经常出现内存溢出、IO异常、任务超时等运行时错误，以及在流处理过程中的数据完整性和计算结果错误等，这些错误会直接造成应用执行失败。

分析产生上述问题的原因主要有以下两点：（1）系统本身是否可靠，即部署好或升级后的系统是否存在缺陷；（2）系统提供的应用是否可靠，即开发好的系统应用是否存在潜在的运行时错误。其中，系统错误，即部署好或升级后的系统是否存在缺陷，包括硬件错误和软件错误。硬件错误包括CPU、内存、网络以及硬盘等错误；软件错误包括特定的逻辑错误、数据竞争等。另外，对于应用错误，已有的实证研究发现：（a）对于IO异常、内存溢出及任务超时等运行时错误，其产生原因[7][8][9]包括不恰当的配置、数据异常以及用户代码缺陷等。不恰当的配置包括输入的数据块过大，分区数目过小，分区函数不均衡等；数据异常包括异常的输入数据、中间结果或输出数据，例如倾斜数据以及高维数据等；用户代码缺陷包括内存泄漏、较高的时间或空间复杂度等。（b）对于数据和计算完整性，根据博客、论坛等的实证分析[10][11][12]，其主要错误有数据错误（如数据丢失、数据重复等）及状态错误（状态丢失、快照错误等），产生原因主要有数据流速过快、Task失效、快照恢复机制不完善等。

对于用户来说，他们希望了解部署好的应用是否存在潜在的应用错误；对于系统设计者和管理者来说，他们希望了解部署的或更新的系统是否有潜在的系统错误。测试是一种常用的且有前景的方法，但是目前的解决方案[13][14]大都是出现问题之后，再针对某一类应用及问题进行分析诊断，当前还没有一个通用的检测方法能够提前发现系统、应用和数据的潜在问题。另外，现有的大数据平台的测试基准如HiBench[15]、BigSQL benchmark[16]、Spark-perf[17]、Graphalytics[]以及SparkBench[]等，关注的焦点主要是特定的大数据平台上的性能及扩展性的基准测试，且主要使用常规的输入数据（固定的真实数据集或其简单合成）以及固定的参数配置进行测试，它们不能直接用于检测潜在错误。对于大数据系统的可靠性测试目前还没有一套类似的测试基准提供支持。

大数据系统在处理“海量”和“高速”数据时，是否可以高可靠的应对高负载的场景已经成为一个亟待研究的课题。针对上述问题以及研究现状，本文将研究如何开发一个针对大数据系统的可靠性测试基准框架，该框架的系统目标主要包括以下几点：

1. 构建一个大数据系统的可靠性测试基准，支持SQL操作、大规模图分析、流式数据处理以及机器学习中使用广泛的典型应用的可靠性测试；并提供可配置的数据生成以及自动化的参数组合测试。
2. 集成数据生成、数据存储、参数组合测试以及报告生成模块，构建一站式大数据系统的可靠性测试基准平台；将开发人员从算法学习、测试脚本编写以及接口学习等复杂繁琐的工作中抽离出来。
3. 构建基于SSH的Web可视化界面，可配置的数据生成以及参数组合测试界面，降低测试人员的使用难度，加快测试速度；自动生成的测试报告，降低分析人员的分析成本。

## 研究内容

在本文中，我们设计实现了大数据系统的可靠性测试基准框架。为了检测潜在的应用程序或系统错误，可靠性测试基准通过生成异常输入数据，并结合特定的系统及应用的配置来测试应用程序。与性能测试基准不同，该基准（1）根据应用特性生成异常输入数据，同时（2）基于参数特征减少参数组合空间。本文将提出的可靠性基准应用在了Spark及Flink大数据系统上，通过构造典型应用、生成异常数据、组合参数来对大数据系统应用进行测试，从而提前暴露系统和应用存在的问题和缺陷。该大数据系统可靠性测试基准框架开发中的关键技术，主要包括以下几点：

1. 提出了针对大数据应用的异常数据生成方法。定义了异常数据的概念，并通过分析应用程序特征，给出了特定应用的异常数据的生成规则，进而生成相应的异常数据（如分布异常、维度过高等特征）。
2. 提出了一种针对大数据应用的异常参数生成方法。该方法通过贪心方式组合系统与应用参数进行测试，并通过二分法分析参数相关性来削减组合测试空间。
3. 提出了针对流式大数据应用的高负载生成方法。该方法通过构建多个负载发生客户端，并通过并行划分数据流量、流速的方法，来生成高负载的应用数据。

最后，论文介绍了大数据系统的可靠性测试基准的设计与实现，并将该可靠性测试基准框架应用在Spark和Flink大数据系统上。通过对Spark及Flink上的典型应用进行可靠性测试，已经在5个应用中发现了三种类型的错误（如，内存溢出错误、运行超时以及计算结果错误），并给出了测试报告。

## 论文组织

论文的后续章节组织方式如下：

第二章介绍大数据系统和大数据应用的发展现状，大数据系统和大数据应用的可靠性问题，以及现有的大数据系统和大数据应用的测试基准框架。

第三章介绍大数据系统应用可靠性测试基准的关键技术。首先，介绍了可靠性测试基准系统架构；接着，介绍了大数据系统典型应用的构造；然后，介绍了大数据系统应用数据生成方法以及参数组合测试方法；最后介绍了大数据系统应用流式负载生成方法。

第四章介绍大数据应用可靠性测试基准框架实现。首先，介绍了系统架构以及相关实现技术；然后，介绍了系统主要模块的实现，包括可靠性基准核心模块、典型应用实现模块、数据生成器、组合参数发生器、系统监测器以及报告生成器；最后，对实现的可靠性测试基准框架进行应用验证，介绍了实验环境以及可靠性测试的实例分析。

第五章对全文做了总结，主要包括论文的工作、做出的贡献以及对未来工作的展望。

# 大数据系统及应用可靠性相关工作

本章介绍了大数据系统及应用的发展现状，大数据系统及应用的可靠性问题分析以及大数据系统及应用测试基准框架现状。

## 大数据系统及应用

随着社交网络、移动互联网、电子商务等技术的不断发展，以及云计算、物联网、互联网+等新兴技术的兴起，全球数据量正在急剧增长，大数据已经成为了当今时代的主旋律。这些网络通信、电子设备以及个人消费等产生的大数据信息对于企业分析用户需求和进行风险管理以及政府管理交通治安等都有着极大的帮助。为了处理和分析这些大数据，并挖掘其背后隐藏的巨大价值，许多大数据系统和大数据应用迅速的发展起来。

### 大数据系统

目前，大数据的处理模式主要包括对静态数据的批量处理、以及在线数据的实时处理[]。因此，现有的大数据处理系统主要分为了批处理系统、流处理系统以及批处理和流处理的混合系统。

* + 批处理系统

当前比较流行的批处理系统有Apache Hadoop等。Apache Hadoop是一种以MapReduce作为处理引擎、以HDFS作为存储的批处理框架。MapReduce[]是一种分布式的编程模型，用于处理和生成大型数据集。通过共享大规模系统集群，MapReduce拥有良好的可伸缩性。MapReduce将分布式编程分为了map和reduce两个阶段：（1）map阶段，用户通过指定map()函数来处理键/值对，以生成一组作为中间结果的键/值对；（2）reduce阶段，通过reduce()函数，将中间结果键值相同的键/值对合并。HDFS是一种提供可扩展的、可靠的、用于数据存储的分布式文件系统，通过使用大量低配置、低成本的服务器代替高配置、高成本的大型单机服务器，通过键/值对代替关系表[]，为MapReduce处理产生的大型数据集提供了可靠的存储策略。

* + 流处理系统

当前使用比较广泛的流处理系统有Apache Storm[][]等。Apache Storm具有高度可扩展性，易于使用，并提供低延迟、有保证的数据处理，因此被广泛的应用于数据的实时处理中。一个Storm作业只需实现一个Topology及其所包含的Spout与Bolt，并通过指定连接方式便可满足流式作业的需求。Storm使开发人员能够以任何编程语言虚拟地开发他们的逻辑，这样可以支持stdin/stdout中的基于JSON的协议通信。Storm的主要特点是（1）编程模型简单，Storm提供了一个非常简单的、类似于MapReduce的操作方式来提供实时处理；（2）快速、可靠的处理消息：Storm提供“至少一次”的语义保障，任务失败时，从数据源恢复数据；（3）扩展性好：Storm提供多个线程和进程，并行的进行流式计算，有良好的水平扩展的能力。

* + 混合处理系统

当前应用较广的混合处理系统有Apache Spark[]、Apache Flink等。Spark和Flink同时提供了批处理和流处理两种大数据处理模式，并提供了更加丰富的生态系统。

Spark是一个基于内存的可扩展的大数据处理系统，它引入了一个称为弹性分布式数据集（RDDs）的抽象概念。Spark利用内存进行快速的数据计算，同时具有快速查询的优势，弥补了MapReduce中网络延迟及磁盘I/O过重的劣势[]，这使得Spark在机器学习等应用中要比Hadoop快10倍~100倍。同时Spark Streaming可以亚秒级增量的对流进行缓冲，然后将这些缓冲作为小规模的固定数据集进行批处理[]。这种使用微批处理的方式来处理流式数据的缓冲机制，提高了系统的吞吐率，但同时也带来了较高的延迟。因此，Spark Streaming并不适合处理对延迟要求过高的流处理场景。图？是Spark的生态系统。



图表 1Spark生态系统

Flink也是基于内存的大数据处理框架，与Spark不同的是，Flink是一种支持批处理任务的流式数据处理框架。Flink基于同一个Flink运行时（Flink Runtime），提供支持流处理和批处理两种类型应用的功能，分别使用DataStream API和DataSet API。Flink提供了自己的内存管理，而不是完全依赖于JVM额内存管理机制，可以灵活的避免频繁GC（垃圾回收）带来的性能波动。Flink通过构建的DAG（有向无环图）来执行任务，其提供的Logical Plan可以实现自动优化迭代，同时在数据流运行时，实现了高吞吐速率和低延迟。图？是Flink的生态系统。



图表 2 Flink生态系统

图？和图？分别展示了Flink和Spark的计算迭代流程。Flink提出了增量迭代的计算模型，在迭代中可以显著的减少计算。因此，Flink比Spark的流处理速度更快，内存利用率更高，更加适合流式计算以及迭代计算。



图表 3 Flink计算迭代流程



图表 4 Spark计算迭代流程

针对上述介绍的大数据系统，本文针对大数据系统的处理类型、计算模式、处理延迟以及处理速度等进行了总结，如表？。

表格 1 大数据系统对比表格

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **大数据系统** | **处理类型** | **计算模式** | **处理延迟** | **处理速度** |
| **Hadoop** | 批处理 | 基于磁盘 | 高 | 慢 |
| **Storm** | 流处理 | 基于磁盘 | 低 | 快 |
| **Spark** | 批/流处理 | 基于内存 | 中 | 快 |
| **Flink** | 批/流处理 | 基于内存 | 低 | 快 |

在上述大数据系统中，Spark和Flink不仅提供了丰富的处理类型，同时有着更加丰富的生态系统，并针对机器学习、交互式查询、大规模图计算等任务提供了支持。Spark发展迅速，社区活跃，并广泛应用在了工业界；Flink以其高效的内存管理和增量迭代，在流式计算中有着很强的优势，并且也在飞速的发展着。本文后续将Spark和Flink作为主要研究对象。

### 大数据应用

互联网、云计算、物联网等应用场景中都会产生大量数据。这些应用场景主要包括社交网络、搜索引擎、数据采集及数据查询等，并根据这些应用场景衍生出了一系列用于处理特定领域的典型应用。

* + SQL查询

SQL是一种比较持久的查询语言，目前已经有广泛的工具支持它。基本的SQL查询类型主要包括原子查询、中间结果查询以及复杂查询。另外，Big Data benchmark，TPC-H以及TPC-DS等主流的数据库评测基准提供了更加丰富的SQL处理语句。

目前的大数据处理系统中，Spark提供的Spark SQL以及Flink提供的Table API和SQL都支持SQL查询的应用场景。

* + 大规模图分析

当前的社交网络发展越来越迅速，Facebook、微博等建立了大量的社会关系网络，并且可以用图来表示人与人之间相互通信的关系。社交网络中的复杂的人际关系形成了一个个具有不同属性的社区，也因此衍生出了基于图的社区发现等算法。在搜索引擎应用场景中，可以用图来表示网页之间的超链接关系。为了衡量特定网页相对于搜索引擎索引中的其他网页的重要程度，Google提出了PageRank算法。在交通领域的应用场景中，可以用图来表示车辆或行人的出行轨迹。收集的位置信息构成的图，可以用来在动态的交通网络中查找最短路径，或进行行程规划等。

由此可见，社交网络、搜索引擎以及交通领域等产生的数据可以用图来表示其中的关联关系。图中的点和边存在着一定的关联性，为了挖掘其中有价值的信息，需要对图数据进行一系列的操作。这些常见场景中的大规模的图数据分析成为了大数据系统需要支持的主流应用。目前Spark中的GraphX以及Flink中的GElly都提供了对图算法的支持。

* + 机器学习

机器学习[]是一门人工智能的科学，通常可以分为监督学习、无监督学习和增强学习三类。（1）监督学习是从给定的训练集中学习出一个函数，当有新的数据到来时，可以根据这个函数预测结果。监督学习的训练集包括特征和目标，并且其中的目标是由人标注的。常见的监督学习有回归分析、统计分类等算法。（2）无监督学习中，训练集没有人为标注。常见的无监督学习包括[聚类](https://zh.wikipedia.org/wiki/%E8%81%9A%E7%B1%BB)、降维等算法。（3）增强学习通过观察来学习动作，是一种试错学习。由于没有直接的指导信息，因此，学习对象需要不断的与环境进行交互，通过试错的方式做出判断。

目前，Spark中的MLlib以及Flink中的ML都对机器学习中应用比较广泛的算法（如聚类、回归等）提供了支持。

* + 流式应用

随着信息技术网络化、泛在化、智能化的发展，以及数据规模大、来源丰富、类型复杂、变化迅速等诸多特征，使得高时效、可扩展的数据处理成为保障信息系统服务质量的必要条件。当前，以智慧城市、智能装备、智能制造、极端交易等为代表的具有大数据、实时流式处理等特征的数据处理场景大量涌现。

流式应用可以从三个维度进行表征[SparkBench]：延迟，吞吐量和状态大小。流式应用对大数据系统的处理速度以及处理延迟都有很高的要求。目前，Spark Streaming以微批的方式处理流式数据，但有一定的处理延迟；Flink是真正的流式处理框架，其提供的DataStream API为流式应用提供了接口支持。

## 可靠性问题分析

当前Hadoop、Spark、Flink等分布式处理框架被广泛应用于大数据的处理分析，如Web索引建立、日志挖掘、机器学习、大规模图分析等。然而这些应用在处理大数据时，经常会出现内存溢出、IO异常、任务超时等运行时错误；同时，在流处理过程中会遇到数据完整性和计算结果错误等情况。这些运行时错误以及计算结果错误等会直接造成应用执行失败。大数据应用及系统的可靠性已经成为学术界和工业界的关键问题。

常见的大数据应用，在大数据系统中，通常可以表示为（输入数据，参数，用户代码）。输入数据通常作为数据块存储在分布式文件系统上；参数包括系统特定配置（例如，输入块大小，分区号）和应用特定配置（例如，K-means的集群k）；用户代码指用户定义的函数，如map()，reduce()和join()，它们处理输入数据或中间结果。

已有的研究工作表明这些错误的原因多种多样，既包括系统缺陷（如设计缺陷和实现bugs），也包括数据异常和应用缺陷（如参数配置不当、代码缺陷等）。由于运行时错误的多样性和复杂性，当前还没有一个通用的错误检测方法能够提前发现系统、应用和数据的潜在问题。

已有的关于大数据系统的可靠性研究主要有以下几点：

1）在MapReduce应用内存使用问题方面：由Xu[7]等人通过研究123个真实Hadoop和Spark应用的内存溢出错误，发现了内存溢出错误的三大原因：框架暂存的数据量过大，数据流异常，及内存使用密集的用户代码。

2）大数据在线查询分析错误方面：Li[8]等人研究了250个SCOPE job（运行在微软的Dryad框架之上）的故障错误，发现错误主要原因是未定义的列，错误的数据模式，不正确的行格式等等。其中84.5%的错误是由于数据处理中的缺陷引起的。他们也发现3个内存溢出错误，错误原因是在内存累积了大量的数据（比如一个大表的所有行被存放到内存中）。

3）在大数据系统运行错误方面：Kavulya等人[9]分析了4100个执行失败的Hadoop jobs，这些jobs运行在Yahoo!管理的M45集群。他们发现36%的故障是数组访问越界错误，还有23%的故障是IO异常。Zhou等人[]研究了大数据平台在微软中存在的问题，他们发现36％的问题是由系统侧缺陷引起的，其中2个问题（1％）是内存问题。

4）在云系统中的开发和部署方面：Gunawi等人[10]研究了云系统上部署的系统（如Hadoop MapReduce、HDFS和HBase等）的3655个issues。他们发现87％的问题是软件故障，而13％的问题是硬件故障。他们还报告了HBase中的1个内存溢出（OOM）错误（用户在大型数据集上提交查询）和Hadoop文件系统中的1个内存溢出（OOM）错误（用户并行创建了数千个小文件）。这些研究帮助我们设计异常数据发生器和配置发生器。

通过上述研究发现，大数据框架在运行时产生错误的原因主要有以下三点：

1. 数据异常：如数据维度过高、数据倾斜等。
2. 不恰当的参数配置：如系统分配内存的大小、reducer数目太小、决策树深度过大等。
3. 用户代码缺陷：如时间或是空间复杂度太高、有内存泄漏的风险等。

## 测试基准框架现状

随着大数据系统及应用的广泛使用，针对大数据系统的测试基准也发展起来。常见的针对大数据系统的测试基准有BigBench[]、BigDataBench[]、HiBench[]和SparkBench[]等；针对特定的大数据应用的测试基准包括Spark SQL benchmark[]（SQL查询）、Graphalytics[]（大规模图分析）、StreamBench[]（流式应用）等。

* + 大数据系统测试基准

BigBench是一种端到端的大数据测试基准，其基本业务模式是产品零售商。该测试基准涵盖了一个数据模型和一个合成数据生成器，针对数据模型的一组查询，生成具有更大数据集、更高速率、更多数据类型（如，来自关系表的结构化数据、半结构化数据以及来自网络点击和社交媒体中的非结构化数据等）的合成数据。BigBench的应用场景比较单一，对大数据系统中的应用类型覆盖不全面。

BigDataBench是针对Web搜索引擎提供大数据测试基准，其相对于BigBench涵盖了更广泛的应用场景，同时提供了多样化和有代表性的真实数据集。该测试基准提供了6个真实数据集以及19个大数据工作负载，并针对不同的数据类型和数据来源提供不同的方法来合成大数据。BigDataBench提供的不同类型的工作负载（如社交网络、搜索引擎、关系查询等），分别对应于不同的大数据系统（如Spark、Hadoop、MySQL等）。

HiBench是针对Hadoop系统提出的测试基准。该基准提供了微基准、Web搜索、机器学习等8个工作负载，并从速度（即作业运行时间）、吞吐量（即每分钟完成的任务数）、HDFS带宽以及系统资源（如CPU、内存及I/O等）方面来评估Hadoop性能。同时，HiBench还为Hadoop的测试提供了非结构化类型的真实数据集。

SparkBench是针对Spark的性能测试工具。尽管BigBench [29]，BigDataBench [26]和HiBench [23]等其他基准测试都涵盖了少量支持Spark的工作负载，但它们远远没有全面覆盖Spark支持的全套应用类型。SparkBench测试基准则覆盖了Spark支持的四种主流的应用类型，包括SQL查询、图计算、机器学习以及流式应用。并且测试中使用到的数据大部分由自带的数据生成器生成。通过对算法或应用的横向对比以及纵向对比测试，从系统资源（如，CPU、Memory等）、任务执行时间、数据处理速度等方面考察Spark的性能。

* + 大数据应用测试基准

Pavlo [15]设计了一个SQL基准测试来比较MapReduce和关系数据库之间的性能。Berkeley AMPLab开发了一个SQL基准[12]来比较Spark、Hive以及Impala等的性能。

Graphalytics是专门用于图处理平台的测试基准。该测试基准提供了有代表性的图计算应用，同时使用LDBC社交网络基准（SNB）数据生成器（Datagen）生成的Graphalytics合成数据集。目前，已经测试了比较流行的Giraph，GraphX和Neo4j等图处理平台。

StreamBench是一个用于流处理系统性能比较的基准框架。 该测试基准定义了一个公共API组件和一组核心工作负载，并为三种广泛使用的开源流处理系统，Apache Storm、Apache Flink和Spark Streaming，提供基准测试。StreamBench支持定义新的工作负载，同时支持对新的流处理系统进行基准测试。

表？总结了上述介绍的大数据系统及应用的测试基准的特性。

表格 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试基准** | **数据集** | **工作负载** | **测试对象** | **测试类型** |
| **BigBench** | 合成数据集 | 关系查询 | DBMS、Hadoop | 性能测试 |
| **BigDataBench** | 真实数据集、合成数据集 | 社交网络、搜索引擎、关系查询 | NoSQL系统、实时分析、在线分析系统 | 性能测试 |
| **HiBench** | 真实数据集 | 微基准、Web搜索、机器学习 | Hadoop、Hive | 性能测试 |
| **SparkBench** | 真实数据集、合成数据集 | SQL查询、图计算、机器学习以及流式应用 | Spark | 性能测试 |
| **Graphalytics** | 真实数据集、合成数据集 | 大规模图计算应用 | GraphX、Giraph、Neo4j | 性能测试 |
| **StreamBench** | 合成数据集 | 基本操作、join、迭代 | Storm、Flink、Spark Streaming | 性能测试 |

* + 现有的测试基准框架主要存在如下问题：

（1）仅支持大数据系统及应用的性能测试。

现有的测试基准仅针对大数据系统及应用的性能提供了基准测试，没有考虑大数据系统及应用存在的可靠性问题。

（2）仅提供常规测试。

现有的测试基准通常使用正常数据以及固定的配置来测试大数据系统的性能，缺乏测试的多样性。

# 可靠性测试基准关键技术

## 可靠性测试基准系统架构

## 典型应用构造

## 数据生成方法

## 参数组合测试方法

## 流式负载生成方法

# 大数据系统应用可靠性测试基准框架实现

## 系统架构

## 系统模块实现

### 可靠性基准核心模块实现

### 典型应用实现模块

### 数据生成器

### 组合参数发生器

### 系统监测器

### 报告生成器

## 应用验证

### 实验环境

### 大数据系统应用可靠性测试实例及分析

# 结束语

## 论文贡献

## 未来工作展望

参考文献

发表文章

致谢