基于trace的网络存储系统评测研究

赵晓南 李战怀 张 晓 曾雷杰
(西北工业大学计算机学院 西安 710129)

摘 要 网络存储系统规模的膨胀和应用范围的扩大,使网络存储系统评测研究越来越重要。网络存储系统的评测包括性能、可用性、可管理性和功耗等内容。论述了网络存储系统主要评测指标的研究内容和研究现状,给出了网络存储系统评测的逻辑框架,分析了在多项指标的实际评测中如何用基于trace的方法分别从trace收集、重放和可用性等几方面对其相关研究现状和常用工具进行讨论。通过分析、总结研究中的问题,展望了相关研究在未来的发展方向和前景。

关键词 网络存储系统,系统测评,性能,可用性,traces
中图法分类号 TP393.06 **文献标识码** A

Research on Trace-based Network Storage System Evaluation

ZHAO Xiao-nan LI Zhan-huai ZHANG Xiao ZENG Lei-jie
(School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract Network storage system evaluation becomes more and more important due to the network storage scale becomes huge sharply and its application scope becomes wide as well. Network storage system evaluation includes performance, availability, manageability, power consumption and other metrics. This paper discussed the current research topics and the status on network storage system evaluation, proposed a logical framework for it, and discussed about how to use trace as a basic means to evaluate each metrics, the challenges and solutions in trace collection, trace replay and trace availability. Finally, gave out expectation on the future research topics in this field.

Keywords Network storage system, System evaluation, Performance, Availability, Trace

1 背景

网络存储系统广泛应用在具有海量数据存储、处理、交换和传输相关业务的各个领域,具有数据量巨大、体系结构复杂等特点。由于其在IT系统中处于核心位置,使得应用对存储系统在处理速度、可靠性和安全性上提出了严格甚至是苛刻的要求。

在实际应用中,由于业务对互联网的依赖,使得网络存储系统往往是各种存储网络结构同常用的互联网结构(如P2P, Cluster等)相融合的复杂系统结构。在如此复杂的系统上开展重要的业务,并保证服务的质量,首要问题是保证系统自身的可用性、可靠性和安全性。因此,形成一套完整的评测方案,确立评价标准,依据一定的模型对这个复杂系统自身进行测试评价,是构建一个网络存储系统所必须进行的工作。目前,在工业界和研究界,相关的研究主要是集中在存储系统的性能评测方面,而对于可用性和可管理性等方面的理论、方法等的研究都比较肤浅,对完整的系统全方位评价体系及方法的研究更是不多。

本文对网络存储系统评测的研究内容进行分析,讨论了trace在网络存储系统各项指标评测中的应用和作用。从trace收集、重放和可用性等方面讨论其研究现状,并结合目前网络存储系统的发展分析了现阶段基于trace的网络存储系统评测研究所面临的问题、难点,分析了未来的研究方向和前景。

2 网络存储系统评测研究

通用的网络存储系统如图1所示,大体分为3层:(1)存储框架层,包含构成网络存储系统的全部硬件设备,如网络连接设备、服务器、客户应用终端、存储设备(包括磁盘阵列、硬盘、磁带等)。(2)系统管理层,负责实现对网络存储系统硬件资源的部署、配置,需要完成如扩容、负载均衡、日志管理等对硬件管理所必需功能。(3)应用服务层,在这一层存储系统同具体应用联系起来,使存储系统的硬件真正能够为实际应用服务,如构建保证高效、安全处理业务的金融数据管理系统,构建支持各种形式数据的数字图书馆系统等。

显然,网络存储系统构成非常复杂,这也决定了对其测试评价的复杂性。首先,由于网络存储系统自身具有层次结构,对系统的评测同样应按相应的层次展开,针对具体的层次需

到稿日期:2009-03-06 返修日期:2009-05-20 本文受国家自然科学基金(60720106001)资助。
赵晓南(1979-),女,博士生,主要研究方向为海量数据存储、管理,E-mail:zhaoxn@nwpu.edu.cn;李战怀(1961-),男,教授,主要研究方向为海量数据存储、数据管理;张 晓(1978-),男,博士,主要研究方向为海量数据存储、管理;曾雷杰(1977-),男,博士生,主要研究方向为海量数据存储、软件工程。

要着重关注各自不同的内容和指标。其次，评测本身也是一个体系，有必要对评测内容按类别进行划分，其中主要包括性能、可用性、可管理性、能耗及功能等多个方面。

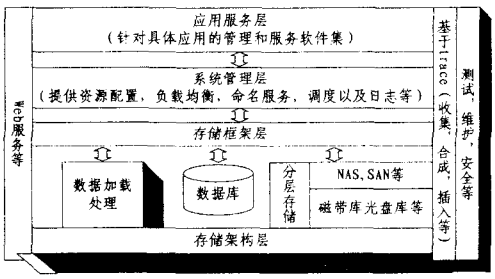

图1 网络存储系统通用框架

2.1 性能

性能研究是存储系统研究的重要组成部分，也是系统评测中一项极为重要的内容。性能研究方法有3类：依赖经验做出判断；对给定的系统采用最直观的实测；建模法，包括建立分析模型和仿真模型两类。在实测方法中，已开发出许多具有针对性的基准测试程序，其主要如表1所列。

表1 主要的基准测试程序

类型	优点	缺点
基于文件系统的基准测试	能表现出文件系统整体的访问性能	评价对象单一，无法覆盖整个系统的评测
基于trace重放的基准测试	能对比在特定应用环境中不同存储系统间的性能差异	若完全记录时空开销大，实现复杂
基于特定操作的基准测试	可获取在特定操作方面的性能差异	执行方式直接影响测试效果

文献[3]中对研究中用到的基准测试工具做了较全面的总结。其中，文件级性能测试工具最常用的是IOZone和IOMeter等，可以完成多平台上的负载生成和评测。SPC(Storage Performance Council)还推出模拟多种应用类型测试系统的一系列标准。基于trace的采集和重放是性能基准测试中的一种重要方法，有关其采集和重放的方法以及相关工具将在下一节具体论述。

由于网络存储系统的复杂性，基于经验判断或基于实测的方法的工作量和时间消耗都比较大，因此在网络存储系统的性能研究中主要采用建模的方法。排队模型、Petri网模型、CART[1]模型等在研究中都被用到。在有关磁盘阵列研究中，较著名的有DiskSim，RaidFrame和Pantheon。而SANSim[2]和SANMO²[3]则是系统级研究的代表。

2.2 可用性

系统可用性是指一段时间内系统处于可用状态的时间百分比。先实际测量平均无故障时间(MTTF，Mean Time To Failure)和平均修复时间(MTTR，Mean Time To Repair)，然后根据MTTF/(MTTF+MTTR)*100%计算获得。

伯克利大学的Brown[4]提出了一种测试可用性的方法：评估系统随时间变化的QoS(Quality of Service)情况，同时评估系统的性能和故障承受程度。利用已有的标准测试程序对系统产生一定的负载，通过插入故障影响其可用性，并记录相应情况下的QoS。Sun公司提出了3R[5]评价基准体系，并提出了利用故障注入方法来评估系统的可用性。故障注入(Fault Injection)是一项在可用性评价中广泛使用的技术，其研究内容包括故障的注入方式、注入类型及注入策略。故障注入技术可分为基于模拟的故障注入、基于原型的和基于物理的故障注入。

2.3 可管理性及功耗

存储管理一方面要提高存储资源的利用率，另一方面要减少人为干预，提高存储管理的智能程度，即研究自动化管理。目前相关的研究主要有CMU PDL实验室的Self-*存储系统[6]、IBM Almaden研究中心提出基于策略的存储管理方法[7]等。随着自动化管理研究的发展，如何评价存储系统的可管理性，已成为一个重要的课题[8]。文献[9]中采用事件注入技术，模拟发生5类故障或异常事件来评价系统的自治能力。哈尔滨工程大学利用事件注入技术来测试网络的可靠性[10]，而国外对这方面的研究还不多。

网络存储系统规模不断膨胀在能耗方面也带来了巨大的挑战。在绿色计算成为IT行业热门话题的今天，功耗成为存储系统的一个重要评测指标。有关系统能耗测试的研究包括SNIA提出的绿色存储工业规范草案、SPEC组织提出的首个针对服务器系统的能耗-性能测试标准SPECpower_ssj2008等。另外，IBM的“Project Big Green”项目首次将能耗问题列为数据中心设计的关键问题。IBM提供了集成的管理软件[11]，可用来监测存储系统的电力使用情况，并根据预定的策略对其进行调整。

2.4 trace的应用

I/O trace可以应用在网络存储系统的测试、管理和应用中。收集大规模网络存储系统中的I/O trace，研究、分析I/O trace特征，建立恰当的I/O负载模型，对存储系统的设计、优化和评估等有十分重要的意义。

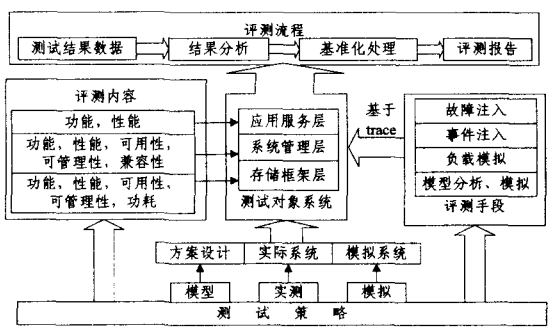

图2 系统评测关系框架

图2具体给出了系统评测的逻辑关系图，图中给出了不同的存储系统层次上需要评测的内容以及评测的策略和评测的流程，表明了各评测内容及评测手段在体系中的位置，说明了各部分之间的关系。其中评测手段主要有4类：故障注入、事件注入、负载模拟和模型分析与模拟。负载模拟和模型分析与模拟是性能测试中最主要的手段，也是trace及其相关技术、工具最主要的应用场所。事件注入技术就是通过向目标系统注入各种系统事件，观测和回收系统的响应信息，并对其进行分析，得出结论的过程。在注入事件和记录信息中要用到trace和相关的技术及工具。故障注入也是类似的情况。

3 trace收集和重放

从上一节的讨论可以看到，在网络存储系统的评测研究中，多种衡量指标的评测手段都依赖于基于trace的方法，或借助、借鉴与其相关的技术和工具等。关于trace的研究可以分为trace收集、trace重放、trace的可用性、额外开销等几个

问题。

3.1 trace 收集

对于评测中使用的 trace，通常有两种来源：实际环境中以一定的方式收集产生 trace 集；根据一定模型和少量的训练数据进行合成，形成仿真 trace 集。尽管有所不同，但两种情况都必须进行收集操作。

根据对应评测研究的层次不同，实际 trace 收集的对象也牵涉到从应用级到块级等各个层次。哈佛大学网络存储服务研究中[12]，将欲收集的 trace 对象分为应用级、kernel-syscall 级、internet kernel/file 级、网络级、块级等不同的类型。目前，最常见的是通过系统调用来收集 trace 这一级。因为系统调用 API 的存在，这一级的收集比较容易实现。捕捉不到内存映射操作的情况是该方法的致命缺点。而文件级的 trace 收集可以覆盖有 cache 和不使用 cache 的需求，还包含了内存映射的情况，但是这种基于文件级的 trace 收集工具受操作系统的限制比较大，即使同一个操作系统的不同版本之间也难以通用，跨平台则更难实现。网络级的 trace 收集，仅仅适用于网络文件系统，这一级的 trace 可以通过特定的硬件设备收集也可以利用软件工具收集。对于设备级 trace 收集，只能是没有 cache 的请求，而且如果没有预先提供的或是推断的附加信息，很难将需求和相关的元数据关联起来。

针对各个不同的层次，trace 收集的任务和内容自然有所不同。在应用级上，需要收集的信息将和程序的具体应用领域有直接的关系，例如数据库应用中数据的存储和查询是其典型的特点，针对这一应用的 trace 收集必须予以考虑，以使得所收集的 trace 不但可以用于重复测试应用级的性能，同时可以作为对应用行为分析的根据。SQL Server Profiler[13] 就是进行跟踪和分析的工具。它可以用来捕获发送到 SQL Server 的所有语句以及语句的执行性能相关数据（如语句的 R/W 页面数目、CPU 的使用量以及语句的持续等）。可以分析 trace 文件中(1)运行最频繁的语句；(2)最影响系统性能的关键语句；(3)各类语句群占有的比例以及相关性能统计信息。在文件系统级上，trace 收集的信息主要集中在文件的变化，因此与元数据的关系非常密切。在收集的过程中，应对用户的隐私进行保护，即考虑实现保护用户隐私的文件级 trace 收集方法。在块级 I/O trace 收集中，关心的内容比较单一，即分析 I/O 时需要的基本信息，包括读写频率、数据量、数据块位置所属等。而通过对块级 I/O trace 的分析，可以得出存储系统设备的性能、使用率等信息。显然，trace 收集的层次越靠近硬件，方法越趋于简单。

trace 收集是 trace 重放的前提。一般情况下，trace 重放的环境应该同所使用的 trace 集收集时的环境比较接近时，效果会比较好。当然也有研究将 trace 集放在一个差别很大的环境下重放，如 Dimitrijevic[14] 等人的研究。

在系统调用级最常用的 trace 收集工具是 strace，还有 DFSTrace。虚拟文件级的有 Linux 下的工具 tracef[15]，而文献[16]则给出了 Windows NT 环境下的工具；传统的 trace 收集方法是一种被动的收集方式，其好处是对系统自身的运行和应用尽量不产生影响。nfstrace，tcpdump 和 nfsdump 都是基于这种方法的 trace 收集工具。nfsdump 是开源的且收集信息最全，与前两者相比最大的不同的是能够获得任意系统调用的有效 UID 和 GID。哈佛大学在相关的研究中是基于 tcpdump 工具的，将有关 UID，GID，IP 地址等信息采用整数替换的映射方式加以保护。

显然，对于不同级别收集 trace 的内容差别很大，因此实现 trace 收集工具的可调节性非常必要。同时，还要统一 trace 输出的形式，以简化后继的解析和重放时的预处理。trace 的预处理是 trace 重放的准备阶段。Zhu 等人[17] 在研究基于 trace 模拟器降低磁盘阵列功耗时，讨论了相关问题。在并行系统中，找出各个节点的相互依赖关系以及交叉 I/O 的运算时间，会有助于提高 trace 重放的正确率。同时，在 trace 收集过程中，尤其是在网络级收集的 trace 中，可能有一些错误的记录，在重放之前必须去除 trace 集中的干扰数据等等。

3.2 trace 重放

trace 重放是一类重要的基准测试，是指把记录的程序状态以某种方式重现出来。在此重现过程中，可以对状态进行任意查询。由于重放并不重新执行程序，因此重放速度随意可调，还可以跳到任意一点进行观察。trace 重放的精确度取决于记录的精确度。一般，trace 重放方法以事件模型为基础，程序的执行过程就是一个不断产生事件的过程。记录这个事件序列，就记录了程序执行的一个抽象过程。重放即这个过程的再现，它通过正确的重现实际的工作负载，实现性能模型测试、验证等目的，也可以用于评测实际的存储系统。

trace 重放中有一些必须考虑的关键问题：(1)地址重定向问题。由于 trace 收集环境和重放环境之间存在差异，如收集和重放协议层的不同，如何实现较好的环境转换非常重要。(2)重放的速度问题。根据实际需要，重放速度可能要调整为较收集速度快或是慢。而某些特定的测试环境下，会要求 trace 重放的速度越快越好。利用双 buffer、预取是改变重放速度常用的方法。工具 Replayfs[18] 重放的速度可以比收集时快，但是由于这个工具是在用户级上实现的，因此不会超过最高的 I/O 速度。(3)trace 的依赖关系。在系统收集的 trace 中，尤其是在复杂的网络结构系统中（如集群系统、并行系统等），不同 trace 单元之间可能存在着一定的依赖关系，或者是时间顺序上的关系。而这种关系并不直接等价于 trace 所记录的先后顺序，如何反映这种关系非常重要。譬如在收集 trace 时，记下对应记录的 IP 等信息，对重放确定 trace 的依赖关系将有很大帮助。如 TBBT 采用了一种叫“保守顺序”的方法来解决依赖性问题，即保证每个操作都将在其前面开始操作全部结束时才开始执行。(4)顺序保证问题。这也可以视为一种依赖关系，即时间依赖关系。

目前的 trace 重放研究主要是基于文件系统层和网络层的。TCPivo[19] 在调整（加快、放慢）重放速度上做了一些工作，可以重放多种工具收集的真实 trace，并且性能较好。另外，有关网络信息流的重放工具，还有 Monkey[20]，TCPopera[21] 等。Buttress 和 DFSTrace[18] 可以在用户级上重放系统调用的 trace，而 Drive-Thru 则可以重放系统调用级和设备级的 trace。另外，DiskSim，Pantheon 分别是 CMU 和 HP 开发的模拟工具，利用这两个工具也可以实现 I/O trace 的重放，进而对其进行分析。EXPERT[22] 是一种离线的 trace 自动分析工具，它将原始 trace 分成若干个部分，然后在多个处理器上并行重放，再将重放的结果拼接起来进行分析。工具 nfsstat，iostat 和 nfswatch[12] 可以提供分析过的 NFS 数据流等等。

3.3 trace可用性及其他

关于trace可用性需考虑两方面问题：trace文件存储的空间开销和trace重放时的系统资源开销。一般，trace文件很大，例如后文提到的HP实验室的trace文件有9G大小，因此trace的存储方式必须要考虑如集中存储等。对此，SNIA的IOTTA(IO Trace Tools and Analysis)技术工作组不但提供了包含世界范围的、存储相关的trace的数据仓库，还给出了一些统一、转换trace格式用到的工具等。事实上，无论采用何种方式收集trace，都会带来系统的额外开销。在网络存储系统评测中，由于高带宽并行、数据量大，需要收集的trace量也相应较大。因此，如何将系统额外开销降到最低，降低收集trace的动作对正常业务的影响，是非常必要的。在并行性系统相关的研究中，为了实现trace重放时的顺序保证，研究trace收集的方法和信息收集对象的变化也是必不可少的。在应用级trace收集的过程中还会涉及到一个非技术难题——隐私问题。应用级的trace中难免包含应用程序使用者的信息，如何保证用户信息的私密性也是trace收集技术研究中一个不容忽视的问题。此外，从测试角度考虑，应用级trace的所有者是需要区分的，但并不需要具体了解每个用户是谁。所以，trace收集工具需要考虑对这部分信息进行一种线性的映射处理，这样既可以保护用户隐私，又可以区分用户的不同，保证重放的正确性。现在，已经有相关的研究在进行了。

IOTTA将trace分为块级I/O trace，NFS trace，并行trace，静态snapshot trace和系统调用trace等几类。其中，块级I/O trace是HP实验室在实际设备上收集的，包括cello91 trace，cello92 trace，cello96 trace和cello99 trace，分别来自相应的年限，主要包括块级和块级协议(如SCSI，ATA，光纤等)的。很多存储I/O相关的研究都是在该trace集上进行的，如CMU在self-*项目中有关性能模型和性能预测等。NFS trace主要是哈佛大学的实验环境收集的用于研究网络文件系统的trace集。并行trace通常是在超级计算机上收集的，记录了多计算机并行运行时的系统调用情况，是加州大学伯克利分校收集的。静态snapshot trace是由微软公司从4800台计算机上收集的。系统调用trace则提供的是主要来自微软公司收集的相应trace数据。SAP trace是Oracle数据库服务器运行时收集的，它包括3000个用户，其中大多数操作是读操作。同时，SPC发布了Financial traces和WebSearch traces这两个trace集。前者是来自OLTP应用，后者是来自一个很受欢迎的搜索引擎，并且是只读的。此外还有TPC-H和OpenMail等公开的trace集[23]，要求存储设备的容量要达到32 TB级，因此比较适合在做海量I/O测试时使用。

4 分析

不难理解，在实际的trace收集工具的设计中，需要关注这样几个问题：(1)加载过程对用户应该是透明的。应做到不影响系统正常业务，无需重启系统或是挂载文件系统。(2)保证低开销。不能使系统处理业务的性能有明显的下降，而影响系统自身的可用性。(3)trace文件不应过大。(4)收集的trace文件要易于管理。

但是，由于评测体系的需要，trace收集必须根据使用目的的不同分别在不同的层次上进行。同时，由于目前市场上的存储产品差别很大，输出信息的形式也各不相同，因而导致采集内容和采集方法的多样性，很难统一。采集内容的多样性导致trace收集后的存储形式很难统一，因而trace输出格式具有多样性。这不仅给预处理和管理带来困难，还增加了trace通用的难度。此外，采用纯软件方式收集trace会给应用系统带来较大的额外负载。对于海量存储系统来讲，大量trace的收集对业务的影响也较大。要达成上述目标并不容易。

因此，应设计软硬件结合的trace收集方式，降低系统附加负载。海量存储系统支持多路I/O并发操作，为了评估系统的并行处理能力，必须同时记录多路I/O trace。而为了保证trace重放时序与收集时一致，必须研究多路trace收集器之间的时间同步及信息收集对象的时间记录方式。I/O trace重放工具需要研究基于用户数据分布模型的负载生成器体系结构设计、用户数据分布模型及系统研制，以支持海量存储系统的数据输入模拟，重点解决符合用户应用特点的大规模用户数据生成、处理结果回收等问题。

结束语 研究网络存储系统评测对推动海量存储技术的成熟和进步有很大意义，对设计、配置和改善海量存储系统有很好的指导作用。目前，国外对相关方面的基准测试和测试方法研究较多，但是对整个系统的评测研究并不多，而完整的评测体系方面的研究更少。在国内，存储市场主要是国外的产品，有关存储系统、网络、管理技术的研究发展较国外有相当的差距，在网络存储系统评测方面的研究更少，目前也没有较权威的海量存储系统评测机构。因此，在这一领域有很多的机遇，也有不少挑战。日前，国家已经启动了相关的863项目，这将使我国在这方面的发展前景更加光明。

参考文献

[1] Wang Mengzhi, Au Kimman, Ailamaki A, et al. Storage device performance prediction with CART models[R]. CMU-PDL-04-103. 2004

[2] Zhu Yaolong, Zhu Shuyu, Xiong Hui. Performance analysis and testing of the storage area network[C]// Proceeding of IEEE Symposium on Mass Storage System and Technologies. USA: IEEE, 2002

[3] 李超. 面向海量数字资源管理的FC-SAN性能分析与监测[D]. 北京：清华大学，2006

[4] Brown A, Patterson D A. Towards availability benchmarks : a case study of software raid systems[C] //Proceedings of 2000 USENIX Annual Technical Conference. San Diego: USENIX, 2000: 22-36

[5] Mauro J, Zhu Ji, Pramanick I. The system recovery benchmark [C]// Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing. 2004: 271-280

[6] Ganger G R, Strunk J D, Klosterman A J. Self-* storage: brick-based storage with automated administration[R]. CMU-CS-03-178. 2003

[7] Kaczmarski M, Jiang T, Pease D A. Beyond backup toward storage management[J]. IBM Systems Journal, 2003, 42(2): 322-337

[8] Lightstone S, Hellerstein J, Tetzlaff W, et al. Towards benchmarking autonomic computing maturity[C]// Workshop on Autonomic Computing Principles and Architectures (AUCOPA'2003). Alberta: IEEE, 2003: 51-59

(下转第170页)

Q. E. D.

一般地，在 SPA 中，对于差异度的每一次进一步分析都不可能遵循同样的同、异、反比例。即有：$a' \neq a; b' \neq b; c' \neq c$。

Proof:

$\mu = a_0 + b_0 i + c_0 j$

where $a_0 + b_0 + c_0 = 1$

more

$\mu = (a_0 + a_1 b_0) + b_0 b_1 i + (c_0 + c_1 b_0) j$

…

Let: $b_{-1} = 1$

for $\mu = a + bi + cj$

where: $a = \sum_{i=0}^{n} a_i \prod_{j=0}^{i} b_{j-1}$, $b = \prod_{i=0}^{n} b_i$, $c = \sum_{i=0}^{n} c_i \prod_{j=0}^{i} b_{j-1}$

Let: $a_{\max} = \max(a_i)$, $a_{\min} = \min(a_i)$;

$b_{\max} = \max(b_i)$, $b_{\min} = \min(b_i)$;

$c_{\max} = \max(c_i)$, $c_{\min} = \min(c_i)$;

where: $i = 0, 1, 2, \cdots, n$

$$\lim_{n \to +\infty} b_{\min}^{n} \leqslant \lim_{n \to +\infty} b \leqslant \lim_{n \to +\infty} b_{\max}^{n} \to \lim_{n \to +\infty} b = 0$$

$$\lim_{n \to +\infty} \sum_{i=0}^{n} a_{\min} b_{\max}^{i-1} \leqslant \lim_{n \to +\infty} a \leqslant \lim_{n \to +\infty} \sum_{i=0}^{n} a_{\max} b_{\max}^{i-1}$$

$$\to \frac{a_{\min}}{1 - b_{\min}} \leqslant \lim_{n \to +\infty} a \leqslant \frac{a_{\max}}{1 - b_{\max}}$$

$$\lim_{n \to +\infty} \sum_{i=0}^{n} c_{\min} b_{\max}^{i-1} \leqslant \lim_{n \to +\infty} c \leqslant \lim_{n \to +\infty} \sum_{i=0}^{n} c_{\max} b_{\max}^{i-1}$$

$$\to \frac{c_{\min}}{1 - b_{\min}} \leqslant \lim_{n \to +\infty} c \leqslant \frac{c_{\max}}{1 - b_{\max}}$$

$$\mu = \frac{a_i}{\max(a_i + c_j)} + \frac{c_j}{\max(a_i + c_j)} j = a_i + c_j j$$

where $a_i + c_j = 1, i, j = 0, 1, 2, \cdots, n$

Q. E. D.

结束语 本文在分析 RST 与 SPA 基本理论的前提下，从人类认知的角度，引入 Vague 集的真、假隶属度研究 RST 的边界区域，从而达到扩展 RST 的目的；同时，采用逐层嵌套的方式运用集对理论进一步分析 SPA 中的差异度，从而达到扩展 SPA 的目的。进而，为了说明 RST 与 SPA 的理论相通，进行了一般性假设和等价性构造，并且证明了逐层嵌套 SPA 的收敛性。本文认为这对进一步发展 RST 和 SPA 有着重要的意义。

参 考 文 献

[1] Zadeh L A. Fuzzy Sets[J]. Information and Control, 1965, 8: 338-353

[2] Pawlak Z. Rough sets[J]. International Journal of Computer and Information Sciences, 1982, 11: 341-356

[3] Deng Julong. Control problem of grey systems[J]. System & Control Letters, 1982, 3

[4] Atanassov K. Intuitionistic fuzzy sets[J]. Fuzzy Sets and Systems, 1986, 20: 87-96

[5] 赵克勤. 集对分析及其应用初探[M]. 杭州：浙江科技出版社，2000

[6] Gau W L, Buehrer D J. Vague Sets [J]. IEEE Transactions on Systems, Man and Cybernetics, 1993, 23(2): 610-614

[7] Bustine H, Burillo P. Vague sets are intuitionistic fuzzy sets[J]. Fuzzy Sets and Systems, 1996, 79: 403-405

[8] 张文修，等. 粗糙集理论介绍和研究综述[J]. 模糊系统与数学，2000, 14(4): 1-12

(上接第 157 页)

[9] Kanoun K, Spainhower L. Dependability Benchmarking for Computer Systems[M]. Chapter 1. IEEE, July 2008

[10] Wang Huiqing, Pang Yonggang, Du Ye, et al. Evaluation of network dependability using event injection[C]// Proceedings of International Workshop on Web-Based Internet Computing for Science and Engineering. Berlin: Springer, 2006: 991-998

[11] IBM. Georgia Tech implements a cool solution for green HPC with IBM, [EB/OL]. ftp://ftp.software.ibm.com/common/ssi/pm/ab/n/oic03003usen/OIC03003USEN.PDF

[12] Joseph Ellard D. Trace-based analyses and optimizations for network storage servers[D]. Harvard University, 2004

[13] http://www.qudong.com/soft/program/Sql%20Server/jiqiaoyunyong/20080328/4795_2.html

[14] Dimitrijević Z D, Rangaswami R, Cahng E. Design and implementation of semipreemptible IO[C]// Proceeding of the Second Usenix File and Storage Technology. San Francisco, 2003: 145-158

[15] Aranya A, Wright C P, Zadok E. Tracefs: A file system to trace them all[C]// Proceedings of the 3rd USENIX Conference on File and Storage Technologies. San Francisco: USENIX, 2004: 129-143

[16] Roselli D, Lorch J R, Anderson T E. A comparison of file system workloads[C]// Proceedings of the Annual USENIX Technical Conference. San Diego: USENIX, 2000: 41-54

[17] Zhu Qingbo, Chen Zhifeng, Tan Lin, et al. Hibernator: helping disk arrays sleep through the winter[C]// Proceedings of the 20th ACM Symposium on Operating Systems Principles. NY: ACM, 2005: 177-190

[18] Joukov N, Wong T, Zadok E. Accurate and efficient replaying of file system traces[C]// Proceedings of the 4th Conference on File and Storage Technologies. San Francisco: USENIX, 2005: 336-350

[19] Feng Wuchang, Oel A, Bezzaz A, et al. TCPivo: a high performance packet replay engine[C]// Proceeding of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research. NY: ACM, 2003: 57-64

[20] Cheng Yuchung, Hölzle U, Cardwell N, et al. Monkey see, monkey do: a tool for TCP tracing and replaying[C]// Proceedings of USENIX Annual Technical Conference. Berkeley: USENIX, 2004: 87-98

[21] Hong Seungsun, WU S F. On interactive internet traffic replay [C]// Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID). Berlin: Springer, 2005: 247-264

[22] Geimef M, Wolf F, Wylie B J N, et al. Scalable parallel trace-based performance analysis[C]// Proceeding in the 13th European PVM/MPI User's Group Meeting (Bonn, Germany). Berlin: Springer, 2006: 303-312

[23] Kurmas Z, Keeton K. Using the distiller to direct the development of self-configuration software[C]// Proceedings of International Conference on Autonomic Computing 2004. NY: IEEE, 2004: 172-179