第38卷 第24期
Vol.38 No.24

计 算 机 工 程
Computer Engineering

2012年12月
December 2012

·软件技术与数据库·

文章编号：1000—3428(2012)24—0037—05 文献标识码：A 中图分类号：TP311

一种大规模分布式应用性能分析系统

臧冬松[1,3]，Vincent Garonne[2]，孙功星[1]

(1. 中国科学院高能物理研究所，北京 100049；2. 欧洲核子研究中心，瑞士 日内瓦 CH-1211；3. 中国科学院研究生院，北京 100049)

摘 要：在网格和云计算环境下，由于平台和网络环境的复杂性，使得对大规模分布式应用的有效监控和性能分析变得非常困难。为此，提出一种基于数据流管理的大规模分布式应用性能分析系统，利用消息队列收集、缓冲和分发追踪消息，使用分布式实时处理框架分析和追踪消息。将该系统部署到一个Petabyte级别的分布式数据管理系统中，通过事例演示追踪消息的重要性。应用结果表明，该系统能够满足大规模分布式应用环境下大数据量处理能力和实时性的要求，为监控并分析系统性能、预测用户行为等提供了较好的平台支持。

关键词：分布式应用；性能分析；数据流管理；消息跟踪；消息队列；NoSQL数据库

A Performance Analysis System for Large-scale Distributed Application

ZANG Dong-song[1,3], Vincent Garonne[2], SUN Gong-xing[1]

(1. Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China;
2. European Organization for Nuclear Research, Geneva CH-1211, Switzerland;
3. Graduate University of Chinese Academy of Sciences, Beijing 100049, China)

【Abstract】Monitoring and analyzing large-scale distributed applications in grid or cloud environment is very difficult, due to the complexity of platform and network environment. This paper describes a system to monitoring and analysis such applications. This system is based on the concept of data stream management. It uses message queues to collect, cache and distribute trace messages, uses distributed computing framework to analyze the trace messages in real time. The prototype is deployed in a real Petabyte-scale distributed data management system. The usefulness of the collected trace messages is demonstrated by examples. Application result shows that this system is easy to deploy and has little affection on the applications, can well suit the requirement of big data analysis and real-time compute, provides a platform to analyze the performance of large-scale distributed system, predict user behavior.

【Key words】distributed application; performance analysis; data stream management; message trace; message queue; NoSQL database

DOI: 10.3969/j.issn.1000-3428.2012.24.009

1 概述

分布式系统的性能分析需要对各个站点/节点的信息进行收集、处理和呈现。在网格或者云计算环境中，应用分布在全球不同国家、不同站点之上。如何对这些分布式应用进行有效集中监控，并利用监控信息来预测系统使用模式，为分布式应用的改进和策略制定提供依据，是一个非常有挑战性的课题。

目前存在的监控系统有很多，比如Nagios、NWS、Ganglia、MonALISA、JobMon、R-GMA、CODE、GridICE。这些监控系统通过在被监控主机上安装代理程序，通过轮询或者监听机制获取各个节点的监控信息，然后将信息进行聚合汇总，处理之后进行显示。这种结构的一个主要的缺点就是需要安装部署代理程序，特别是在网格或者云计算环境中，这种安装部署通常很难实施。另外，这些系统无法对应用内部进行更细粒度的监控。本文提出一种新的监控框架，通过在分布式应用中嵌入轻量的客户端，将追踪消息通过简单文本协议返回给消息服务器，以便于进行各种处理。这种结构的优点是无需部署，对系统的影响非常小，适合于网格和云计算环境下的分布式应用的监控。

系统的另一难点是数据流的处理。在数据流管理领域，文献[1]讨论数据流系统的扩展模型，支持持续查询、近似算法、自适应、强制时序、滑动窗口等功能。文献[2]提出数据流管理的概念并介绍实现框架Aurora。近年来以Facebook的Puma[3]、Yahoo的S4[4]、Twitter的Storm[5]为代表的实时流计算平台成为了继Mapreduce之后并行处理的又一个研究热点。本文借鉴S4、Storm的数据流系统

基金项目：国家自然科学基金资助重点项目(90912004)

作者简介：臧冬松(1981－)，男，博士研究生，主研方向：分布式计算，海量数据管理；Vincent Garonne，博士；孙功星，研究员

收稿日期：2012-02-13 **修回日期：**2012-04-08 **E-mail：**donal0412@gmail.com

模型，结合大规模分布式应用的特点，提出一个简单、可靠、可扩展的数据流实时处理框架。相对于 Strom 等平台，该框架利用消息队列实现消息排序，确保数据的可靠性，采用推送-确认(push-ack)模式实现负载调度，利用分布式计数器进行数据聚合，提供了简单方便的用户接口。

2 性能分析系统框架

2.1 系统需求

大型分布式系统(网格、云计算)中应用的性能分析需要充分考虑系统的分治性和异构性以及监控数据流量的大小及突发性。比如 WLCG 网格由分布于全球 34 个国家的 170 多个站点及国家网格组成，各个站点由网格中间件和通信接口相互连接，站点内部的硬件和软件环境各不相同，站点的维护和访问策略通常由内部管理员来管理，这为网格上的分布式应用的集中监控带来了难度。这种环境下的监控系统应该易于统一部署，尽量减少对应用的影响，应对高通量的数据流分析需求，并可以根据需求方便地对系统进行扩展，能够可靠地对数据流进行处理，提供数据持久化(Persistence)和容错(Fault-Tolerance)机制。

2.2 系统结构

性能分析系统的总体结构包括数据流(Stream)的产生、分析和查询(如图 1 所示)。本文中的数据流是一个抽象概念，每个数据流由无限个有序元组(Tuple)组成，每个元组包含任意个 key：value 格式的键值对，客户端(Client)程序被嵌入到分布式应用中，客户端产生的各种数据流汇聚到相应的消息队列(MQ)中，消息队列中的元组被集群中的处理单元(PE)并行地进行处理。系统提供了查询调用接口(Query API)，供用户对数据流进行实时查询。

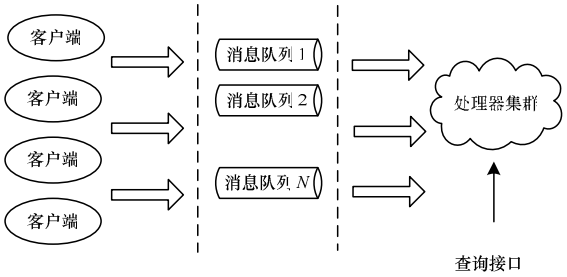


图 1 分布式性能分析系统框架

在实际生产环境中，消息队列服务器一般需要进行容错。比如 2 台或者多台服务器同时运行，通过 DNS 提供统一的名称，并按照一定的调度策略将数据流分布到所有服务器上。当其中的一台服务器失效时，数据流被自动发送到其他服务器，不影响服务的正常运行。

2.3 数据流排序

在数据流处理中，元组的顺序非常重要。比如要对某个时间段，或者某个序号范围内的元组进行统计查询，就需要对来自不同客户端元组进行统一的排序。由于客户端的系统设置差异以及网络因素，来自客户端的元组中的时间戳无法作为统一排序的依据。在 Babcock B 的研究中提出滑动窗口的概念，给出了 2 种时间戳标记方法：隐式(implicit)和显式(explicit)，并讨论了数据流排序的几种解决方法。本文采用隐式时间戳，以元组到达队列的时间先后进行排序，忽略网络传输因素造成的误差。这种误差(<1 min)对于大规模的分布式监控来说是可以接受的。

2.4 数据流的并行处理

数据流的并行处理需要合理的调度和高效可靠的合并(aggregate)操作以及数据的持久化和出错处理等。

元组的调度控制采用 Pub-Sub 方式：处理单元根据配置首先向相应的消息队列注册，当有数据到达消息队列时，由消息队列将数据推送到某个注册过的处理单元进行处理。如图 2 所示，消息队列 Queue1 中的元组被推送给多个处理单元(PE1 PE2…PE*n*)进行并行的处理(Map 过程)，处理后的结果以 key：value 的格式推送到聚合器 Aggregator 进行合并得到最终的结果(Reduce 过程)。图 2 中的虚线表示确认消息 ACK 被返回给消息队列，表明消息已经被正确处理，可从队列中删除。队列收到确认消息后，会继续向该处理单元推送数据；否则系统认为该处理单元异常，将不再向其发送数据，同时消息继续保留在队列中，保证消息不丢失。

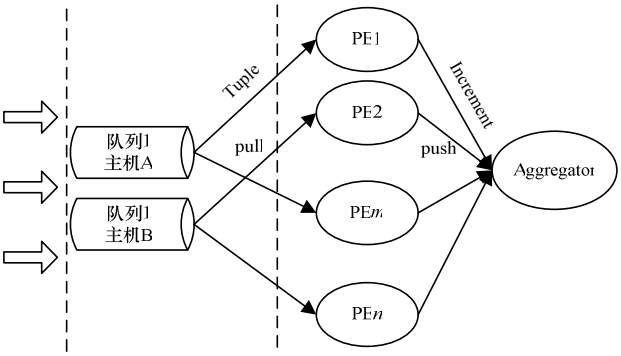


图 2 数据流分布式处理流程

为了简化处理单元的逻辑，聚合类统计(比如 count、sum、avg、max、min 等)结果的合并采取增量(Increment)模式。处理单元缓冲一定时间的元组进行统计，将结果以 key：increment 的格式发送给聚合器，聚合器将 key 的现有值增加：value+=increment。增量模式的一个优点是处理单元可以实时地将结果发送给聚合器，而不必等待全部数据到达。比如要统计每天的元组个数，采用增量模式，聚合器中将可以实时地查询到今天到当前为止的个数。另一个优点是处理单元中缓存的时间较短，在处理单元失效后可以很快的从消息队列中恢复，处理单元本身不需要做持久化工作。

增量模式的缺点是对聚合器的写性能要求较高，同时需要保证结果的一致性。一种可行的方法是使用多个聚合器，将处理单元的结果按照键值(key)的哈希值发送到不同的聚合器进行合并。考虑到最终结果的持久化以及聚合器失效等问题，使用一种 key：value 格式，多副本的分布式数据库作为聚合器的实现是一个不错的选择。

下面给出分布式性能分析框架的一个具体的实现。

3　性能分析系统实现

Don Quijote2(DQ2)[6]是一个 Petabyte 级别的分布式数据管理系统，它运行在全球 LHC 计算网格(WLCG)之上，用来管理高能物理 ATLAS 实验[7]的数据。WLCG 是一个多中间件网格基础设施，包括欧洲高效电子科学网格(EGEE)[8]、北欧的先进资源链接器网格(ARC)[9]和美国的开放科学网格(OSG)[10]。在 WLCG 网格环境下，应用的管理和控制是在分布式异构环境下进行的，而且管理员的数量跟组成网格的站点数量相比，通常非常小。

DQ2 负责管理所有的实验数据。这些数据分布在全球不同的基础设施、不同的站点，因此，是一个多中间件的应用。DQ2 和其他的实验软件交互并提供分布式数据访问的唯一接口。数据管理的逻辑单元叫做数据集(DataSet)，表示一组文件的结合。数据集存储在网格站点的海量存储系统中。

3.1　数据访问模型

主要的用户接口 DQ2Clients 包括 3 个命令行工具：dq2-get，dq2-put 和 dq2-ls。这些工具可以作为交互式命令使用，也可以集成到自动工作流中使用。dq2-get 从远程网格站点中下载数据集；dq2-put 将数据集存储到分布式系统中；dq2-ls 使用户可以在分布式系统中搜索数据。

以 dq2-get 为例描述它与跟踪(Trace)消息的交互过程，图 3 显示了 dq2-get 的工作流程。

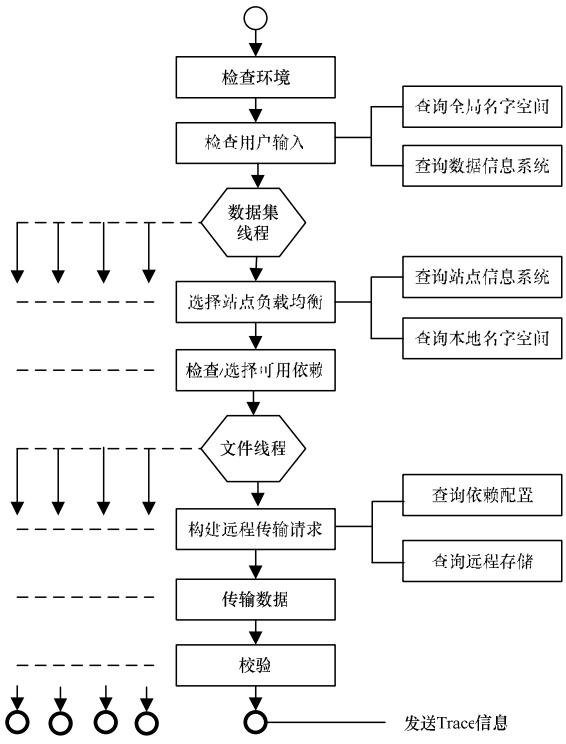


图 3　dq2-get 工作流程

如果用户想获取所有符合 dataset.*的数据集，只需要执行 dq2-get dataset.*。这会获取符合条件的所有数据集 dataset.1、dataset.2、dataset.3。用户不需要知道关于网格协议和配置的任何信息，因为 dq2-get 自动获取需要的信息，启动并监控传输进程。对于要获取的每一个数据集，都会启动一个单独的请求，因为不同的数据集可能在不同的站点上，而且同一个数据集中的不同文件的访问协议也可能不同。当 dq2-get 结束时，关于该操作的相关 Trace 信息就会被发送出去。

3.2　系统实现

性能分析系统的客户端程序被嵌入到 DQ2Clients 中，运行在不同的硬件上，通过 stomp 协议[11]将 Trace 消息发送到中心消息队列。中心消息队列使用 Apache 开源的 ActiveMQ[12]，使用 2 台服务器，采用 DNS 做负载均衡。在 2 台商业服务器上部署数据流处理单元集群，集群通过一个 Master 进程进行管理，包括处理单元的状态监控和动态启停。聚合器采用由 10 个节点组成的 Cassandra[13]集群，负责结果的持久化以及聚合操作的增量运算，提供查询 API。消息服务器和 Cassandra 集群作为通用服务，是与其他应用共同使用。以上描述了一个实际系统的具体实施，下文对该系统进行评估。

4　性能分析系统评估

分布式性能分析系统的评估一方面需要评估整个框架的性能是否能够满足需求；另一方面是评估所收集的跟踪消息是否有用。

4.1　性能测试

图 4 显示了单个 ActiveMQ 服务器 Trace 消息写入性能，每个 Trace 消息(大小为 1 MB)建立一个短连接，测试表明服务器能稳定提供每秒 1 000 个消息的写入速度。当前 DQ2 系统每小时的 trace 消息数大概为 20 万条(如图 5 所示)，高峰时期，消息数可达到每秒 250 个。因此，2 台 ActiveMQ 服务器能够满足性能需求。

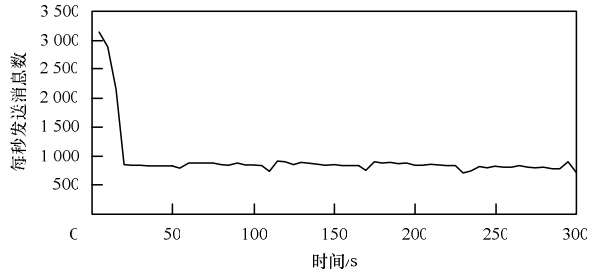


图 4　ActiveMQ 写性能测试结果

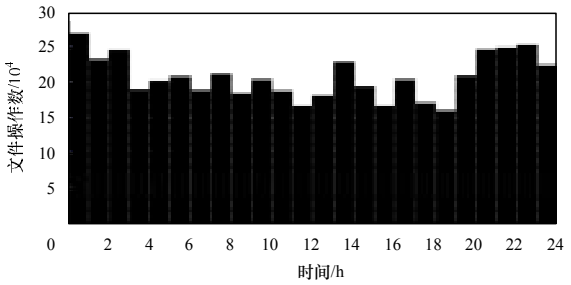


图 5　WLCG 网格上的 ATLAS 文件访问频率

如图 6 所示，Cassandra 集群的写入性能可以达到每秒 10 万条 Trace 消息，增量计数器[14]的性能应该更高。而处理单元的缓冲和批量写入，也能减少读写次数。处理单元的效率与处理逻辑有关。而无论是消息队列服务器，

处理单元集群，还是 Cassandra 集群，都有很好的扩展性和容错性。

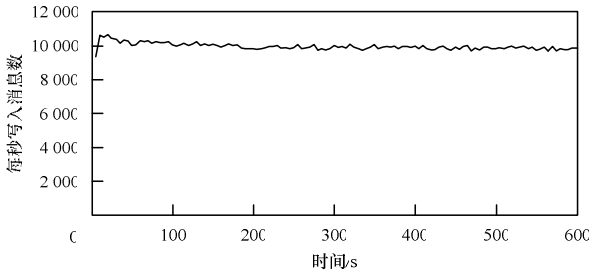


图 6 Cassandra 写性能测试结果

4.2 跟踪消息的重要性

对跟踪消息的分析能够提供许多以下非常有价值的信息：

(1)最直接的用处是对系统的实时监控，如图 5 所示的数据访问量和访问频率以及对历史数据的汇总分析，比如分析数据类型和数据传输的目的站点的比例。由于查询是基于处理单元的处理结果，因此速度非常快，一般可以在 1 s 内完成。

(2)对跟踪消息的分析可以得到最近哪些文件的访问次数最多，哪些文件的访问次数最少，这便是 popularity 服务[15]。而基于 popularity 服务的动态副本放置服务以及副本删除服务，很好地解决了数据副本在全球站点的合理分布，提高了磁盘的利用效率。原来的 popularity 服务基于 Oracle 上的数据聚合查询，速度比较慢，时间延长一般在 1 天左右，利用新的分布式处理框架，可以提供实时的服务。

(3)好的性能分析系统应该能够自动地对系统提供支持，比如调度和决策制定。以大型分布式数据管理系统 DQ2 为例，预测每个海量存储系统(Mass Storage System, MSS)的访问量非常重要。海量存储系统的性能与并发请求数直接相关。如图 7 所示，32 个用户的并发请求能使海量存储系统的性能急剧下降。虽然可以通过一些静态的调度策略来控制最大并发访问数，但是如果能够动态地预测那些会对系统造成影响的用户行为，就可以动态地调整数据调度策略，减少突发访问对系统造成的影响。

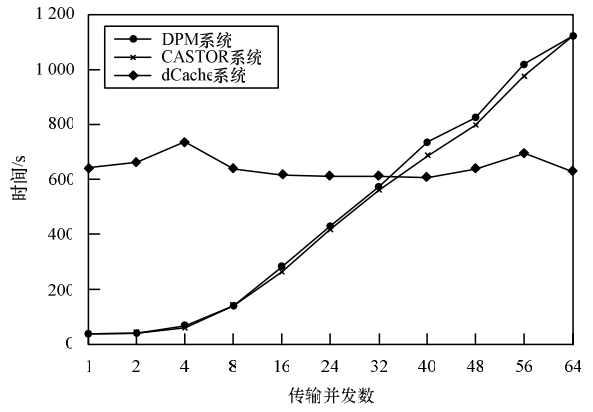


图 7 并发请求对海量存储系统的影响

下面演示如何通过收集的跟踪消息来预测用户行为。

以每天的数据访问量做时间序列图，如图 8 所示，显示数据整体上有一个线性增长的趋势，表明时间序列是非平稳的。对序列的自相关函数(Autocorrelation Function, ACF)进行分析，如果序列是一个随机过程，那其自相关函数应该接近于 0。如图 9 所示，时间序列的自相关值远大于 0。对非平稳序列进行平稳化处理，去除趋势因素后得到自相关和偏自相关函数如图 10 所示，可以看到序列具有明显的周期性，每 7 天具有一个峰值。

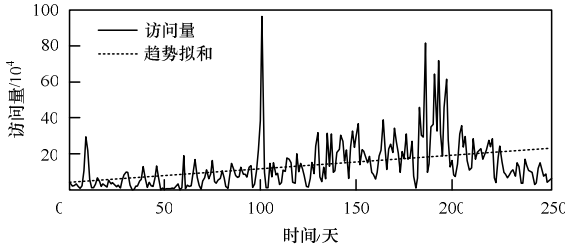


图 8 每日访问量时间序列

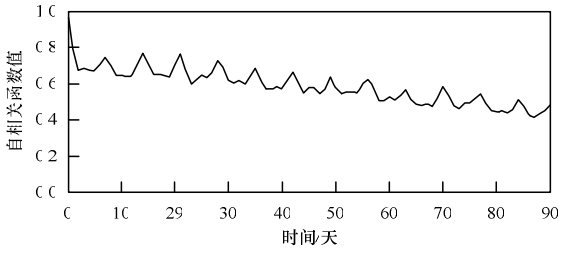
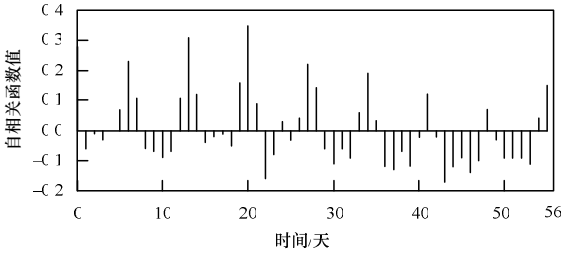
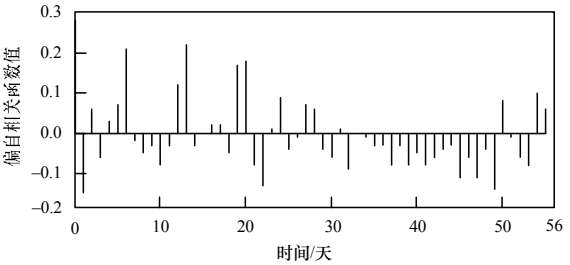


图 9 时间序列自相关函数



(a)平稳化后的自相关函数



(b)平稳化后的偏自相关函数

图 10 平稳化后的自相关和偏自相关函数

针对以上分析，本文给出 2 种预测模型：Last 7 和季节自回归移动平均模型(SARIMA)。Last 7 模型以过去 7 天平均值来预测下一个值；SARIMA 源于自回归移动平均模型(ARIMA)[16]。ARIMA 模型的具体表达形式是 ARIMA(p, d, q)，其中，AR 指自回归过程；MA 指移动平均过程；p 是自回归项数；q 是模型的移动平均项数；I

是单整；d 为时间序列由非平稳成为平稳序列需要差分的次数。当模型的随机时间序列存在季节性的非平稳性问题时，就要应用季节自回归移动平均模型，即：SARIMA$(p, d, q)\times(P,D,Q)^s$。其中，P 是季节自回归阶数；Q 是季节移动平均阶数。根据 AIC 准则，可以发现 SARIMA$(1,1,1)\times(1,1,1)^7$ 具有最小的 AIC 值。

本文通过标准均方根误差(Normalized Root Mean Square Error, NRMSE))检验 2 种模型，标准均方根误差越小，残差(residuals)值变动越小，说明模型预测的越准确。Last 7 模型的 NRMSE 为 0.158，SARIMA$(1,1,1)\times(1,1,1)^7$ 模型的 NRMSE 为 0.062。图 11、图 12 显示了对 2 种预测模型效果的比较：Last 7 模型没有能够预测到序列的高点，但总体的标准均方根误差并不大；SARIMA 模型的效果更好一些，尽管没能预测到第 24 天的高点，但总体上能够很好地预测未来的数据访问情况。

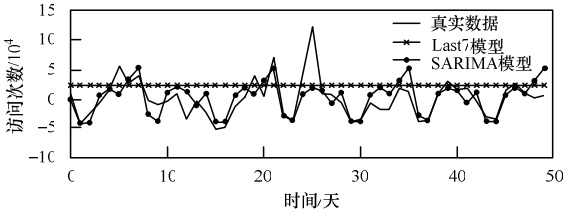
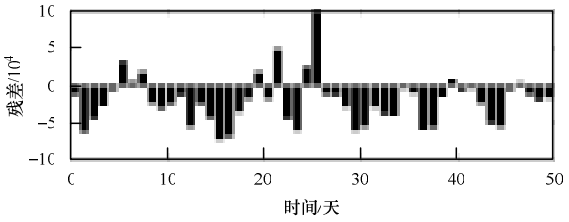
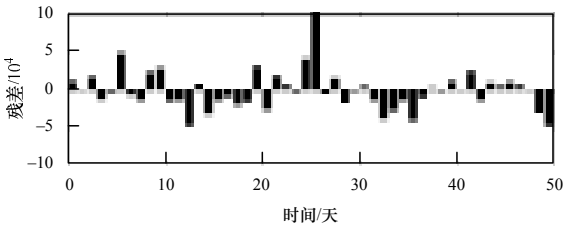


图 11　真实数据与预测模型曲线



(a)Last 7 模型的残差



(b)SARIMA 模型的残差

图 12　2 种预测模型的残差比较

5　结束语

本文介绍一种网格和云计算环境下的分布式应用性能分析系统，解决了复杂分布式环境下集中式监控的部署和管理难度，最少限度地减少了对应用本身的影响。将数据流处理的概念引入到分布式应用性能分析中，利用消息队列和分布式数据流处理系统实现性能分析系统的高效性、可靠性、可扩展性。最后通过一个实际应用验证了该框架的可行性，并通过几个事例说明了大型分布式性能分析系统的重要性。相对于过去基于关系数据库的查询模型，本文系统基于数据流系统的一个缺点是用户接口不像 SQL 语句那样容易使用，需要预先定义监控的内容，并通过 API 进行查询，但相对获取了高效的查询效率和系统的扩展性。今后将完善该大规模分布式应用性能分析系统的易用性和通用性，开发更多的性能分析模型，并将其推广到其他类似的网格和云计算应用中。

参考文献

[1] Babcock B, Babu S. Models and Issues in Data Stream Systems[C]//Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York, USA: ACM Press, 2002.

[2] Abadi D J, Carney D. Aurora: A New Model and Architecture for Data Stream Management[J]. The VLDB Journal, 2003, 12(2): 120-139.

[3] Shao Zheng. Data Freeway and Puma: Realtime Data Streams and Analytics[C]//Proc. of HiC'11. Beijing, China: [s. n.], 2011.

[4] Yahoo. S4[EB/OL]. (2010-05-17). http://incubator.apache.org/s4/.

[5] Twitter. STORM[EB/OL]. (2011-04-02). https://github.com/nathanmarz/storm.

[6] Branco M. Managing Very Large Distributed Datasets[C]//Proc. of OTM'08. Berlin, Germany: Springer-Verlag, 2008.

[7] ATLAS Collaboration. ATLAS Technical Proposal[EB/OL]. (1994-07-11). http://atlas.web.cern.ch/.

[8] Laure E. Middleware for the Next Generation Grid Infrastructure[C]//Proc. of the 2nd Workshop on Grids Meets Autonomic Computing. New York, USA: ACM Press, 2004.

[9] Ellert M. Advanced Resource Connector Middleware for Lightweight Computational Grids[J]. Future Generation Computer Systems, 2007, 23(2): 217-240.

[10] Avery P. Open Science Grid[J]. (2007-03-05). https://www.opensciencegrid.org/bin/view.

[11] Github. The Simple Text Oriented Messaging Protocol[EB/OL]. (2004-06-12). http://stomp.github.com/.

[12] Apache. ActiveMQ[EB/OL]. (2007-08-16). http://activemq.apache.org/.

[13] Apache. Cassandra[EB/OL]. (2009-11-04). http://cassandra.apache.org/.

[14] Apache. Cassandra Increment Counters[EB/OL]. (2009-11-08). https://issues.apache.org/jira/browse/CASSANDRA-1072.

[15] Angelos M. Popularity Framework to Process Dataset Traces and Its Application on Dynamic Replica Reduction in the ATLAS Experiment[J]. Journal of Physics: Conference Series, 2011, 331 (7): 23-55.

[16] Box G, Jenkins G, Reinsel G. Time Series Analysis: Forecasting and Control[M]. [S. l.]: Prentice Hall, 1976.

编辑　陆燕菲