基于 Spark 的分布式大数据分析算法研究

宋泊东 张立臣 江其洲*
(广东工业大学计算机学院 广东 广州 510006)

摘 要 随着大数据时代的到来,数据计算的实时性和数据量面临许多挑战。为了满足庞大的数据量和大数据高速处理的要求,研究将 Apache 作为一种集成的资源管理系统。采用 Apache Storm、Apache Spice 及 SARK RDD 处理大型分布式实时数据流,使用 Apache Kafka 作为消息中间件来支持异步消息的通信。设计一种支持并行运算规则的分布式大数据分析处理算法。实验结果表明:该算法可有效降低海量数据的分析速度,且支持系统内各子系统间的异构信息沟通与数据存储,足以满足高频交易市场的短期趋势预测需求。在高频、大数据处理系统中具有较高的应用价值。

关键词 Apache Kafka 分布式 Spark RDD n 层 实时数据流

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.01.008

DISTRIBUTED BIG DATA ANALYSIS ALGORITHM BASED ON SPARK

Song Bodong Zhang Lichen Jiang Qizhou*
(*School of Computers, Guangdong University of Technology, Guangzhou 510006, Guangdong, China*)

Abstract With the coming of the big data era, the real-time and data quantity of data computation is facing with many challenges. To meet the requirements of large data volume and high-speed processing of big data, we took Apache as an integrated resource management system. We adopted Apache Storm, Apache Spice and SARK RDD to deal with large-scale distributed real-time data streams, and used Apache Kafka as message middleware to support communication of asynchronous message. A distributed big data analysis and processing algorithm was designed, which supported parallel operation rules. Experimental results show that the algorithm can effectively reduce the analysis speed of massive data and support heterogeneous information communication and data storage among subsystems. It is sufficient to meet the demands of short-term trend forecast in high-frequency trading market. It has high application value in high frequency and big data processing system.

Keywords Apache Kafka Distributed Spark RDD n layer Real-time data stream

0 引 言

大数据将在更高的层面、更广的视角、更大的范围帮助用户提高洞察力、提升决策力。但是,一些具有价值往往隐藏在大数据中,表现出了价值密度极低、分布极其不规律、信息隐藏程度极深、发现有用价值极其困难的鲜明特征。

这就要求这大数据计算系统具备高性能、实时性、分布式、易用性、可扩展性等特征。如股票市场的高频交易(HFT),因为短期的市场趋势和快速的报价,人们很难及时决定何时买进或卖出,对大数据分析的准确性、快速性都有极高的要求。根据《纽约时报》2012 年发布的数据,美国所有证券交易市场都有 50% 的高频交易。因此,本文试图通过设计一个 n 层分布式计算模型,来解决海量数据的分析和处理,满足 HFT 系统计算、处理大型实时、弹性数据流的要求。

1 算法设计

1.1 n 层分布式计算体系结构

HFT 系统需要在短时间内处理大数据集。这些大

收稿日期:2018 - 07 - 23。国家自然科学基金项目(61572142)。**宋泊东**,硕士生,主研领域:数据挖掘,信息安全,信息物理融合系统。**张立臣**,教授。**江其洲**,硕士生。

数据集节点之间的通信非常复杂。因此,需要统一消息中间件来在每个服务之间传输消息。王聪等[1]提出了一种面向云计算的数据中心网络体系结构设计。朱虎明[2]的基于集群计算的免疫优化算法及其应用研究提出了一种去噪方法,取得了更好的计算效率。分布式系统需要一个面向消息的中间件来实现“松耦合”。朱晓明等[3]提出了一种基于松散耦合的分布式信息系统的数据挖掘,以提高复杂网络相关分析的计算速度。李卫榜等[4]提出分布式大数据函数依赖发现分析技术,该检测算法在检测效率方面明显提升。许智宏等[5]提出一种基于 Spark 的改进协同过滤算法,该算法明显提高了推荐的准确度。李文昊等[6]研究了确定性分布式数据库中长事务处理方法,改进方法降低了长事务对确定性分布式的影响。宋顶利等[7]建立了交通运行状态数据分析模型,相比标准算法具有更高的准确性。参考上述研究成果,我们使用 Kafka 作为 HFT 系统的消息传递中间件,因为它具有良好的性能和松散耦合的接口。并通过 Apache Storm 将各层信息传递到集群中的多个节点。实现对所有服务进行统一管理,最大化地利用集群资源。

1.2 分布式消息传递系统:Kafka 和 Netty

目前,大数据中应用较多的有 Hadoop、Hive、Flume、Kafka、Storm、Spark 这些技术。其中,Storm 是 Twitter 开源的一个分布式的实时大数据计算系统。Kafka 是一个分布式、支持分区的、多副本的基于 Zookeeper 协调的分布式高速消息传递系统。为了获得更高的吞吐量和更快的处理速度,我们将服务分为两层:实时业务逻辑层和非实时业务逻辑层。在实时业务逻辑层中,我们在集群信息拓扑中使用一个名为 Netty 的超高速消息传递中间件。在 Storm 中,Netty 具有高性能、低资源消耗、低延迟的信息传输特征。同时 TIBCO FTL 也是一个高性能的消息传递中间件,也适用于 HFT 系统中的消息传输。

1.3 实时流媒体数据处理框架:Apache Storm

Apache Storm 是一个免费开放的分布式实时计算框架。该框架具有可伸缩性、弹性、可扩展、高效且易于管理员使用。Storm 支持许多用例,比如实时分析、在线机器学习、持续计算、分布式 RPC(远程过程调用)、ETL(提取-转换-加载)等。目前,Apache Storm 日趋成熟,在很多社交媒体网站及公司中得到了应用,如阿里巴巴和搜索引擎百度。Storm 为大数据流的实时计算提供了很大的效率。Apache Storm 可以每秒处理超过 100 万个节点消息。同时,Storm 集成了一些现有的技术,如传统的消息队列系统、分布式计算技术和数据库系统,如 Kafka、Apache Hadoop 和 HBase。因此,将 Storm 应用于处理连续数据流,可以有效补充 HFT 系统中最初 Hadoop 架构的不足。如在高频金融系统中,由于 Map/Rreduce 编程模型难以处理流媒体市场报价,因此,采用 Apache Storm 可以有效解决高频金融体系结构中的海量流数据的处理问题。

Storm 是一个分布式、可靠、容错的流式数据处理系统。Storm 处理工作被分派给不同类型的组件,每个组件分别负责一个简单、特定的处理任务。处理 Storm 集群输入流的组件叫喷口(spout),喷口再将数据传给一个叫螺栓(bolt)的组件,并在螺栓中处理数据,处理完成之后,螺栓要么将这些数据存储起来(存储在数据库、磁盘甚至是对象中),要么将它传给其他螺栓。因此,可以将 Storm 集群想象成一个螺栓链,每个螺栓都会对喷口发送的数据做出一些处理。图 1 显示了一个简单的 Storm 拓扑结构。

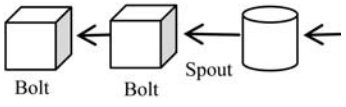

图 1 Storm 的一般拓扑结构

在 Storm 集群中有两种节点:主节点和工作节点。主节点上运行了一个叫 Nimbus 的后台进程,它负责在集群内分发代码、为每个工作节点指派任务、监控失败的任务。工作节点上运行了一个叫 Supervisor 的后台进程,它执行分布在不同的机器上的 topology 的工作节点。图 2 是 Storm 的模型的高级架构。

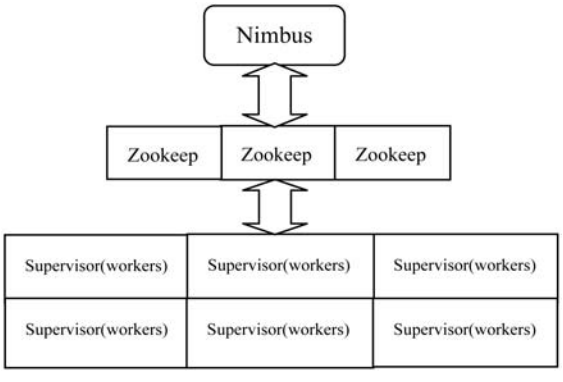

图 2 Storm 的高级拓扑结构

在高频交易系统中,使用 Storm 实时处理市场报价并及时计算市场状态。可以在几毫秒内计算出数百万个市场状态。

1.4 预测模型构建

为提高数据处理进程。本研究采用 Apache Spark RDD 建立大型数据集处理框架。通过 Spark RDD,HFT 系统的速度比 Hadoop Map/Reduce 在迭代机器学习作业中的速度快 10 倍。陈锐忠等设计了一种基于 Hadoop 的海量数据处理模型,该模型具有高效率、低成

本以及易维护的特性[8]。Spark 可以访问任何 Hadoop 支持的存储系统,以补充 Hadoop 进行大规模数据分析[9]。这对于 HFT 系统非常重要,因为 Spark 可以从基于 Hadoop 的存储中访问市场状态的大数据集并在内存中使用它。从而显著降低机器学习任务的运行时间,降低内存消耗。

1.5　资源管理:Cloudera 和纱线

随着分布式计算技术的不断使用,HFT 系统的资源管理变得越来越重要。根据魏祖宽等[10]编著的基于 Hadoop 的大数据分析和处理,Cloudera 提供了一个高性能的资源管理工具,采用 Cloudera 可以轻松地在集群上部署和安装 Hadoop。便于用户配置资源。因此,在处理海量数据时,可以选择 Cloudera 来管理分布式服务。如 Storm cluster、Kafka broker 和其他支持服务[11]。

2　算法实体结构

2.1　HFT 高频交易系统总体架构

分布式计算 HFT 高频交易系统运行在一个分布式集群上,包含几个主要的子系统和服务。这些子系统负责最关键的功能,如计算市场状态、建立机器学习算法模型或运行模型以预测未来趋势。我们将整个系统安排为 n 层分布式体系结构:表示层(PT)、前端交换层(FST)、后端交换层(BST)、实时业务逻辑层(RBLT)、非实时业务逻辑层(NRBLT)和数据访问层(DAT),如图 3 所示。

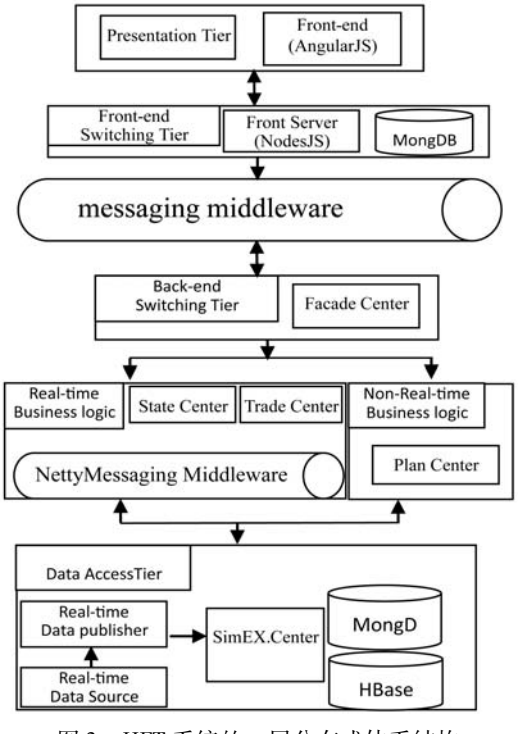

图 3　HFT 系统的 n 层分布式体系结构

这种架构是从三层分布式架构演化而来的。最终,我们将业务逻辑层分离为实时业务逻辑层和非实时业务逻辑层。此外,我们使用两个级别的消息传递中间件来解决整个系统中的高频需求。

2.1.1　表示层(PT)

表示层这一层从 BLT 获取数据,并准备 Web 页面呈现给在线浏览的用户。为了加快加载速度,降低访问时间的延迟,我们使用 Facade 服务来处理从用户到后端集群的所有请求。使体系结构更具松散耦合性。

2.1.2　前端转换层

前端交换层负责接收 Web 请求,并通过 Kafka 消息系统将它们传递给 Facade。此层包含部署在节点上的前端服务器。考虑到运算效率,我们将 MongoDB 部署于前端服务器,使之通过前端交换层发送到 Kafka,不必进入后端集群进行数据处理。

2.1.3　后端转换层(BST)

后端切换层从 Kafka 获取消息,通过 BST 入口接口进行前端服务器与后端交换层进行信息传输。

2.1.4　实时业务逻辑层(RBLT)

实时业务逻辑层是高频系统的关键部分,主要负责实时数据的处理和计算。它包含两个重要的服务:数据分析和决策。如一个股票交易价格预测系统,就需要一个风暴拓扑,以快速处理实时报价流,并存储到 HBase 中。供 HDFS 计算 Rket 状态。即:计算出买入或卖出的信号。如图 4 所示,如果将用户端和交易平台划分为两个拓扑网络。通过 Kafka 消息系统计算市场状态并将其传递给用户。为了提高效率和更高的速度,我们合并了这两个拓扑,并将 Kafka 消息传递中间件替换为 Netty,实现信息的高频处理和传输。在 Storm 拓扑中,Netty 的速度大约是 Kafka 的 10 倍。

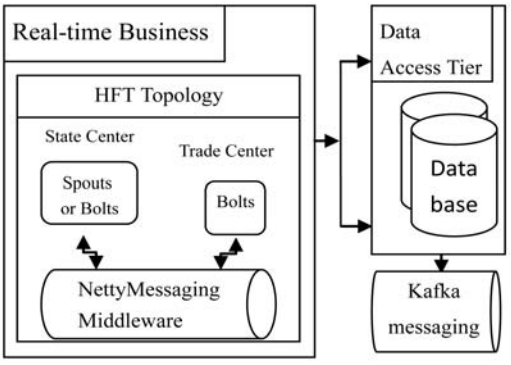

图 4　实时业务逻辑层

2.1.5　非实时业务逻辑层(NRBLT)

非实时业务逻辑层的功能是使用户可以根据大数据计算的信息结果进行决策。决策出的策略存储在 MongoDB 中,便于用户从前端节点快速访问。利用 R 程序和 Spark RDD,就可以获得快速访问大型数据集

的接口。

2.1.6 数据访问层(DAT)

数据访问层用于访问来自数据库或外部数据源的所有数据。如 DAT 提供了一个订单接口,并能将大数据信息组合成一个 K-Bar,通过 Kafka 统一消息传递中间件向用户即时反馈外部数据信息。

2.2 市场状态计算和低延迟存储

通过上述模型架构,我们虚拟开发一个股票交易大数据分析决策计算实例,对算法流程予以分析。股票交易价格由网络交易平台提供实时报价和市场交易状态。由于需要满足机器学习和快速计算的要求。我们首先将网络交易中心作为一个拓扑,来实现算法的低延迟。图 5 是整个状态中心拓扑。

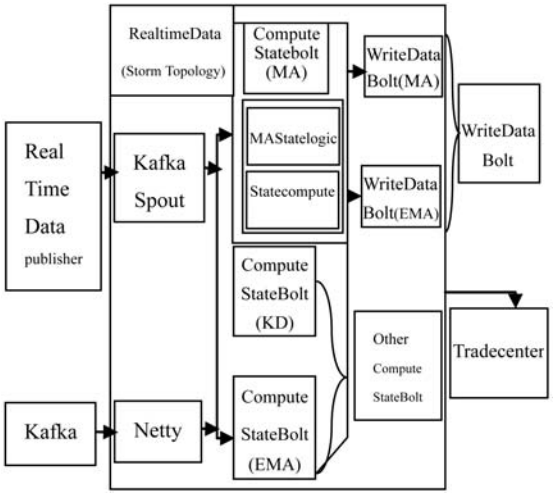

图 5 状态中心拓扑结构

在拓扑中,KafkaSpout 从外部 RealtimeDataPublisher 服务接收实时报价,并通过分布式消息系统 Kafka 不断地发送市场实时交易价格。然后 KafkaSpout 将价格传递给随后的 ComputeStateBolt。每个 ComputeStateBolt 带有不同的计算机逻辑,并使用它来计算特定 TA 逻辑中定义的市场状态。然后,ComputeStateBolt 将结果市场状态发送给特定的 TA WriteDataBolt。每个 WriteDataBolt 都将相应的 TA 数据写入 HBase。例如,ComputeStateBolt 将市场状态发送给 MAWriteDataBolt,专门存储 MA 市场状态。在拓扑之外,所有黑线都表示 Kafka、Netty 分布式消息传递系统传输。

2.3 大数据集处理模型

高频交易市场数据分析的目的是为用户采集市场交易和价格状态。因此,我们需要采用机器学习算法,学习历史市场数据,然后根据历史趋势变化规律,帮助构建投资策略。为了解决大数据集快速学习问题,本文采用 Apache Spark 框架内的 Plan Center 运行机器学习算法,从 HBase 加载大型历史数据集,并在短时间内学习和分析。Plan Center 支持向量机(SVM)、逻辑回归(LR)和分类。通过 Spark RDD, Plan Center 就可以将数百 GB 的市场状态数据加载到内存中,并在集群中的多个节点计算分析。帮助用户提供交易策略。交易策略生成后,用户可以选择在 Web 页面上使用的投资决策。大数据集处理模型架构如图 6 所示。

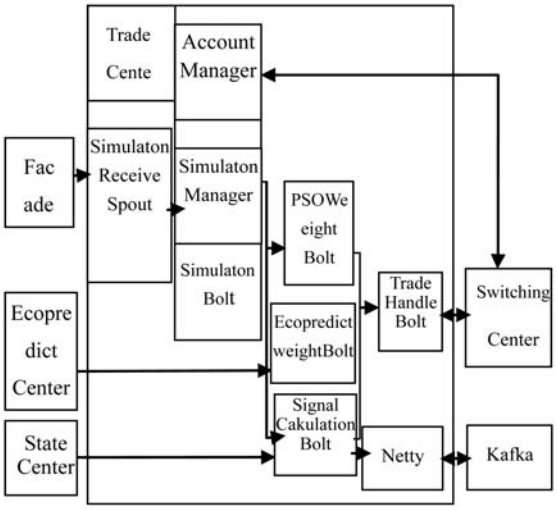

图 6 原始设计

2.4 系统整合与集群云管理

为了减少大数据分析、处理和传输的开销时间,我们将状态中心拓扑和交易合并为一个大型拓扑,形成一个大型 HFT 系统。

整合后的 HFT 从 Kafka 队列提取数据,并将数据写到 HBase 和 MongoDB 中。图 7 显示了整个 HFT 拓扑结构。通过对网络交易中心和用户端的整合,就可以信息传输的成本时间缩短到几毫秒。但是,由于大型交易系统体系结构的复杂性,必须进行高效的集群资源管理,才能有效提高算法的计算速度。因此,我们使用纱线来启动大多数服务并管理集群上的所有资源,并通过定制 Hadoop 服务的配置监视 Cloudera 上每个节点和 Hadoop 服务状况,实现大数据集的集群服务。

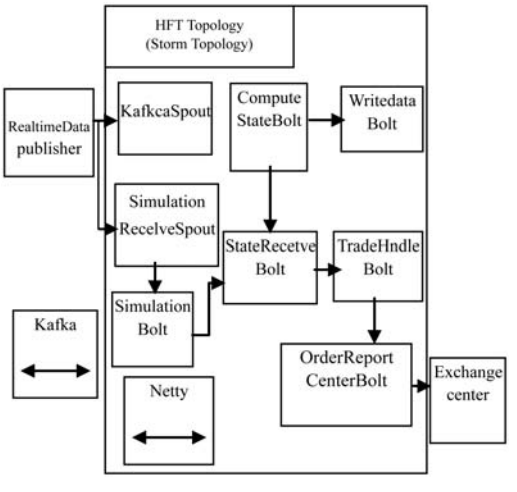

图 7 HFT 拓扑结构

3　实　验

在本节中,我们给出了实验结果,以证明该体系结构的性能在高频交易的可靠性。

3.1　实验环境

实验环境如表 1 所示。由于高频和实时数据处理要求,交易中心需要在 1 秒内计算出数百万个市场状态。因此,本文搭建了模拟实验环境,比较不同数量期货交易的算法性能处理结果。我们准备了 8 台计算机作为集群,其中 6 台作为管理器来运行 Storm 拓扑。

表 1　集群的详细信息

主机	CPU	内存	硬盘	操作系统	Kafka
nccu-master	Intel(R)(TM)i5-2400 CPU@3.10 GHZ	16 GB	1 TB	Ubuntu 12.04	- - - - - -
nccu-mgmt	Intel(R)(TM)2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	- - - - - -
nccu-n01	Intel(R) Core(TM) 2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	- - - - - -
nccu-n02	Intel(R) Core(TM) 2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	- - - - - -
nccu-n03	Intel(R)(TM)2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	- - - - - -
nccu-n04	Intel(R) Core(TM) 2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	代理
nccu-n05	Intel(R) Core(TM) 2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	代理
nccu-n6	Intel(R)(TM)2Quad CPU Q8400 @ 2.66 GHz	8 GB	500 GB	Ubuntu 12.04	代理

为了测试该体系结构的极端效率并找出最适合集群的配置,我们对每个实验的所有市场状态的平均计算时间进行了比较。

3.2　算法性能

为了检验算法性能,我们在原来的拓扑结构中添加了一个新 bolt,以快速收集市场状态的所有计算度量数据,通过计算平均值检验算法性能,并对比原算法和改进算法。图 8 显示了同期货量的平均计算时间(延迟),图 9 显示了 N 个股票的市场状态数。

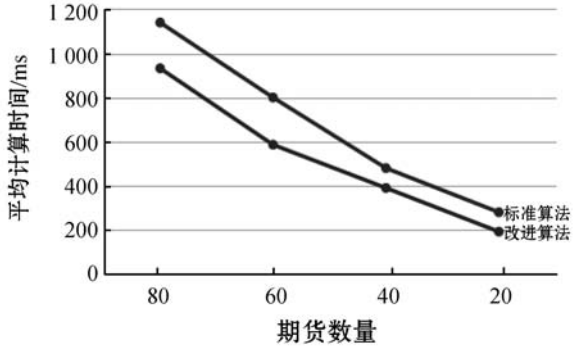

图 8　市场状态的平均计算时间

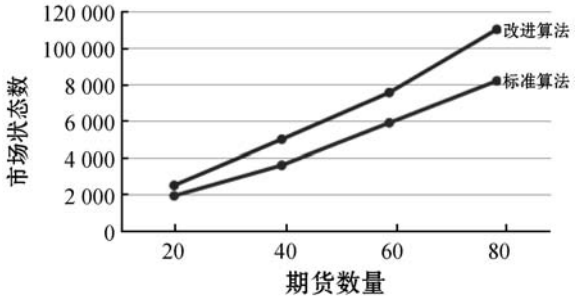

图 9　每秒钟不同期货量的市场状态数

在图 8 中,我们可以看到原算法和改进算法处理相同期货量的不同延迟时间;图 9 中显示了两种算法每秒处理相同期货量的不同市场状态数。性能结果显示在相同条件下改进后的算法延迟时间明显减少,每秒钟处理的市场状态数明显提高。

如果时间粒度为 1 秒,那么每 1 秒系统将收到 1 280 个市场状态。如果期货的数量是 50,计划中心每秒钟可以计算和处理 64 000 个市场状态。

3.3　低延迟的计算

在图 8 中,我们可以看到每个 k-bar 的市场状态计算的平均花费时间。当期货数量为 10 时,即 HFT 系统同时处理 10 个期货报价时,市场状态计算的平均成本时间为 94.5 毫秒。虽然期货的数量是 20,但平均成本时间是 192.0 毫秒。平均成本时间和期货数量之间存在近似线性关系。如果期货的数量为 80,则市场状态计算的平均成本时间为 932.6 毫秒。因此,HFT 系统有 6 个逻辑处理子系统,可以处理 80 个期货市场状态的计算,延迟小于 1 秒(1 000 毫秒)。

低延迟计算是大数据高效处理的关键因素。本研究将 State Center 构建在 Storm 框架之上,并使用 Storm 中内置的 Netty 消息传递,具有较高的数据处理性能。实验表明:HFT 系统可以每秒钟发出 110 260 个市场状态,仅有 932.6 毫秒的延迟。在 6 个逻辑处理子系统和 50 个期货的情况下所有市场状态计算成本时间都在 500 毫秒以下。

4 结 语

本文提出了一种基于 Storm、Spark、Kafka 等分布式计算技术的分布式多层大数据分析处理模型，并采用这种模型，通过高性能的消息传递中间件 Apache Kafka，分析、处理实时和非实时数据。实现了一个高频交易(HFT)系统。该模型包含表示层、前端切换层和业务逻辑层。每个层负责不同的功能，可以根据表示层接收到的用户信息请求，在前端切换层和业务逻辑层完成计算分析，可以在数十毫秒到数百毫秒之间处理数百万个市场状态的计算和订单提交。实验结果表明，HFT 系统的性能优越。改进后的算法性能优于原始算法，相同条件下平均计算时间明显减少，使得每秒钟可处理的市场交易额得到提升。在正常情况下，可以支持用 6 台 PC 计算 80 个期货的市场状态，且延迟在 1 秒内，足以预测高频交易市场的短期趋势。该体系结构还可以用于许多其他高频、大数据和流数据处理系统。

参 考 文 献

[1] 王聪，王翠荣，王兴伟，等. 面向云计算的数据中心网络体系结构设计[J]. 计算机研究与发展，2012，49(2)：286－293.

[2] 朱虎明. 基于集群计算的免疫优化算法及其应用研究[D]. 西安：西安电子科技大学，2010.

[3] 朱晓明，刘卫东. 基于松散耦合的分布式信息系统的数据挖掘[J]. 计算机工程，2004，30(2)：181－182，197.

[4] 李卫榜，李战怀，陈群，等. 分布式大数据函数依赖发现[J]. 计算机研究与发展，2015；52(2)：282－294.

[5] 许智宏，蒋新宇，董永峰，等. 一种基于 Spark 的改进协同过滤算法研究[J]. 计算机应用与软件，2017，34(5)：247－254，278.

[6] 李文昊，李海芳. 确定性分布式数据库中长事务处理方法研究[J]. 科学技术与工程，2016，16(13)：92－95.

[7] 宋顶利，张昕，于复兴. 分布式优化 Apriori 算法的交通运行状态数据分析模型[J]. 科技通报，2016；32(10)：202－206.

[8] 陈锐忠，魏理豪，梁哲恒，等. 基于 Hadoop 的海量数据处理模型研究和应用[J]. 电子设计工程，2016；24(14)：101－103.

[9] Gu L，Li H. Memory or Time：Performance Evaluation for Iterative Operation on Hadoop and Spark[C]//2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing. IEEE，2014：721－727.

[10] 魏祖宽，刘兆宏. 基于 Hadoop 的大数据分析和处理[M]. 北京：电子工业出版社，2017.

[11] Jenie Y I，Kampen E J V，Ellerbroek J，et al. Taxonomy of Conflict Detection and Resolution Approaches for Unmanned Aerial Vehicle in an Integrated Airspace[J]. IEEE Transactions on Intelligent Transportation Systems，2017，18(3)：558－567.

(上接第 38 页)

5 结 语

数据耦合关系和控制耦合关系的测试覆盖目标是 DO-178C 标准中规定的机载软件结构覆盖中的目标，在本项目开展过程中通过评审、测试和分析三种验证方法结合的方式来满足。本文介绍了数据耦合和控制耦合测试覆盖目标满足的方法和过程，对后续机载软件研制过程中如何开展数据耦合和控制耦合分析具有实际的指导意义。

参 考 文 献

[1] 杨珂瑶，张小芳，曾雷杰. 基于 DSP 的嵌入式软件测试方法[J]. 计算机与现代化，2014(10)：61－65.

[2] DO-178C. Software considerations in airborne systems and equipment certifications[S]. RTCA December 13，2011.

[3] An analysis tool for coupling-based integration testing[EB/OL]. [2000－9]. https://cs.gmu.edu/~offutt/rsrch/papers/coupcover.pdf.

[4] Meyers S C. Automatic generation of data coupling and control coupling test conditions：US，15/331259 [P/OL]. 2018－04－26. http://www.freepatentsonline.com/y2018/0113796.html.

[5] https://www.rockwellcollins.com.

[6] 林枫. 简图页软件验证中的数据耦合与控制耦合分析[J]. 工业控制计算机，2014，27(7)：107－112.

[7] 张军才，王娟，潘卫. 基于 DO-178B 的结构覆盖分析研究[J]. 航空计算技术，2011，41(4)：67－69.

[8] Verification tools assessment study[EB/OL]. [2007－6]. http://www.tc.faa.gov/its/worldpac/techrpt/ar0654.pdf.

[9] Clarification of structural coverage analyses of data coupling and control coupling[EB/OL]. [2004－2－28]. https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-19.pdf.

[10] 倪红英. 民用飞机软件验证要求研究与实践[J]. 航空电子技术，2017，48(1)：43－47.