一种流数据多播接口的设计、实现与应用

谢正茂 张 帆 李晓明

（北京大学信息科学与技术学院 北京 100871）

摘 要 海量流数据的分析与处理是信息社会面对的一个基本问题。各种传感器汇聚的数据是流数据，人们发出的短信对于移动通信运营商的数据中心来说是流数据，人们写的微博对于新浪或者腾讯来说是流数据，搜索引擎网页爬取子系统传给后台处理的数据也可以看成是流数据。尽管它们的应用背景不同，但有共同的特征，即存在一个网络上的汇聚节点，从该节点的角度看，数据源源不断地到来。通常，这些数据会以某种特定的格式缓存起来，待某个特定的后续系统处理。启发本文工作的问题是：那些数据常常是有多方面价值的，有些甚至是当前没有想到的，我们有必要同时开放一个流数据接口供未来可能出现的新应用调用。该接口应该具有如下特征：（1）向外输出原始流数据；（2）允许其他（多个）应用程序动态接入和退出；（3）接入的应用程序的行为不影响数据搜集和最初设计的后续系统的功能。本文以连续运行了10年以上的天网搜索引擎和中国Web博物馆（WebInfomall）为例，讨论其网页搜集子系统的改造以适应上述需求，IP多播是采用的基本技术。在介绍了设计思想和实现要点后，我们也给出一个“新应用”的实际例子。这样一个接口的实现，为各种网页流信息分析应用打开了一扇窗口。该接口的设计思想也可以用于其他流数据汇聚系统中。

关键词 流数据；多播；网页抓取；API；松耦合

On the Design, Implementation and Application of a Multicast Interface for Streaming Data

XIE Zheng-mao ZHANG Fan LI Xiao-ming

(*School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871)

Abstract A novel programming interface (API) is introduced in this paper. The basic requirement is for multiple and asynchronous calling processes to capture a data stream without affecting each other. IP multicasting is the primary tool employed for this purpose. UDP is used at transport layer. The current implementation is based on the stream of web pages produced by a high performance web crawler. Along with the interface, a demo application (WordCount) is also implemented. Experiments have verified the robustness of the design, and preliminary measurement shows 1-3% data loss is observed, which is acceptable for many streaming data applications.

Keywords streaming data; multicasting; web crawler; API; loose coupling

1 问题背景与意义

随着经济社会信息化的全面深入，各种物联网的部署和应用，云计算技术的采纳，我们不难想象：在信息网络上存在（或者将会存在）大量“节点”，这些“节点”不停地接收源源不断涌来的数据，对它们进行分析和处理。原则上讲，任何一个到达的数据在节点上存留的时间都是有限的，尽管节点可以有很大的存储缓冲。我们称这样的数据为“流数据”。遍布一个城市中的交通状况监控摄像机传给交通管理部门数据中心的数据是流数据；从千千万万信用卡刷卡机上汇聚到信用卡公司处理系统的数据是流数据；人们发出的短信对于移动通信运营商的数据中心来说是流

谢正茂，助理研究员，研究兴趣包括Web数据挖掘、分布式系统和海量数据的组织管理。**张帆**，研究生，现从事与大规模网页信息处理技术相关的研究与开发工作。**李晓明**，教授，网络与信息研究所所长。E-mail：lxm@pku.edu.cn。

数据；人们写的微博对于新浪或者腾讯来说是流数据；搜索引擎网页爬取子系统传给后台处理子系统的数据也可以看成是流数据。所谓这样的数据在节点上的“存留时间有限”有两层含义：一是数据本身价值的时效性很强，过了一定时间（例如1分钟）后就没意义了，因此可以在那之后丢掉，二是数据虽然可能有“历史意义”，有必要保存较长时间，但由于新的数据在不断到来，老数据即便存贮起来也不会像新数据那样容易访问，对于某些要求而言，相当于“不再可用”。

一个典型的情况是，当人们设计一个涉及流数据的处理系统的时候，往往是针对某种特定应用的。系统的数据接收部分常常会以某种与该应用绑定的格式将数据输出给后续系统处理。但是，人们随后会发现，所涉及的流数据常常是有多方面价值的，有些甚至是当前没有想到的，因此不断会有新的应用构思产生出来。然而，最初设计的数据格式很可能对新应用是不合适的。因此，有必要同时提供一个流数据接口供未来可能出现的新应用调用。该接口应该具有如下特征：（1）向外输出原始流数据，而不是特定应用相关格式的数据；（2）允许其他（多个）应用程序动态接入和退出，原则上没有个数限制且互相不干扰；（3）接入的应用程序的行为（例如出错宕机）不影响数据接收和最初设计的应用系统的功能。

北大网络实验室十多年前开发出“天网搜索引擎”和“中国Web博物馆”，一直维护运行至今。其中一个关键部分是网页搜集子系统，是上述两个系统的共同数据来源。该搜集子系统运行在一个机群上，并发成千上万个线程，在中国Web上7×24不断爬取网页，汇聚到我们实验室的服务器上。因此，它的输出就可以看成是一个数据流（平均每分钟到达约5000网页）。

以此为背景数据，通过改造原有机群的网络结构，采用IP多播技术，我们设计并实现了一个应用接口（API），满足前述三个特征要求。本文的剩余部分组织如下。第2节介绍该接口的设计思想。第3节讲实现要点与测试情况。第4节介绍一个应用实例（WordCount），我们可以看到，这个例子虽然简单，但已经很好地展现了流数据应用的基本问题，即数据分析处理速度与数据到达速度不匹配的问题。第5节为结语和进一步可以考虑的工作。可以看到，尽管本文报告的工作的核心是一个应用接口，但整个工作的系统性、集成性相当强，体现在硬件与软件的综合考虑，体现在要保证已有应用与新应用的共存。

2 需求分析与设计思想：以天网网页抓取子系统为例

如上所述，流数据的接收与处理是一类比较普遍的问题。但我们这项具体工作最初来源于对天网网页抓取子系统改造的实际需求。下面给出分析和由此导致的访问接口（API）的设计思想，虽然一些细节的针对性比较强，但我们相信其精神对设计其他类型流数据接口也有借鉴意义。

2.1 API需要满足的条件

搜索引擎是互联网上最重要的一类应用之一，它通过对互联网上的信息（主要是网页）进行收集、加工处理，然后对公众提供查询服务。互联网网页的收集是所有后续工作的基础，由专门的网页抓取子系统完成。天网网页抓取子系统负责同时为我们的两个Web应用－天网搜索引擎（http://e.pku.edu.cn）和中国Web博物馆（http://www.infomall.cn）－提供网页数据。当我们希望提高这两个应用的信息服务质量时，首先考虑的就是提高数据来源的质量，也就是抓取到的网页的覆盖面和质量。为此，需要对网页抓取子系统的状态进行监测，对得到的网页数据进行及时分析和评判。这是我们设计这套流数据多播接口的最初动机。

为了抓取中国Web百亿规模的海量网页，网页抓取子系统必须采用分布式多节点结构。天网网页抓取子系统具有良好的可扩展性，现在的实际运行节点数在10到20之间，且分布在不同网段的两个机房内。每个节点可以运行上千个抓取进程，每个进程负责对一个Web主机进行网页的抓取。系统对中国Web采用增量式抓取策略，在初始化阶段每天收获的网页数量为3000万到5000万。进入增量持续抓取阶段后，收获的网页数量维持在每天400万左右。

面对如此巨大的数据量和相对复杂的系统结构，特别是保持长期持续运行的要求，如果想了解系统的运行情况，很长一段时间我们只有下面的两种方法：

（1）在网页抓取的过程中抽查进程日志，必要时用调试工具进入进程内部；

（2）采用后期处理的方法，按照约定的格式，网页抓取系统把一段时间内得到的网页存在磁盘上，提供给Web应用。同时进行复制，以磁盘文件的形式分发给网页统计分析程序，经其处理之后获知该批网

页的数量、分布和质量。

第一种方法以手工的方式进行，费时费力而且只能得到个别网站的最粗浅信息。第二种方法得到的信息或多或少总是陈旧的，并且还有分发数据带来的风险和代价：原数据的安全，网络带宽和额外的磁盘空间。于是我们希望，在抓取网页的时候能实时对网页进行统计分析，并看到它们的状态。这里的关键就是要解决“网页抓取”和“网页分析”两个功能之间如何传递网页数据的问题。两者有相当不同的特点：

网页抓取作为基础功能，需求已经相当明确，并且经过长期的开发、运行、改进，其算法与程序结构都趋于稳定。一个时间只有一套系统在运行，由它为下游应用提供数据：互联网上每时每刻都在出现新的网页，网页抓取也需要每天24小时有连续产出。为了降低维护成本，我们对它的运行稳定性提出了较高的要求：至少能够几个月不宕机。

网页分析在很多方面恰恰相反。不但在已有需求上不断有新的算法产生，而且新的需求也不断出现。每个需求、每个算法都可以开发一套独立的分析程序，多个程序同时运行。它们通常没有经过严格的测试，有些干脆采取“边运行，边调试”的方式，稳定性没有任何保证。通常人们对它们的可靠性也没有很严格要求，宕机后改掉错误重新来，只要不影响其他进程。

认识到这种情况，网页分析要实时处理抓取的网页，两者之间的通信API需要满足下面的条件：

（1）不能在同一个进程内：否则网页分析代码中的错误会带着网页抓取一起宕机；

（2）不能在同一个节点上：否则网页分析可能会争抢网页抓取所需的CPU、内存、硬盘等资源，影响网页抓取的运行；

（3）在同一个网络设备上，网页分析所需的数据传递，不能抢占网页抓取所需的连接外部互联网的带宽；

（4）传递网页时，抓取系统不提供与分析网页的同步：因为各种分析算法的复杂性差异很大，分析一个网页所需的时间难以确定；

（5）通信是单向的：只需要把网页数据从抓取系统传递给分析程序；

（6）需要有良好的可扩展性，能够支持多个数据发送者——网页抓取的分布式节点；和多个数据接收者——网页分析程序；

（7）通信给网页抓取带来的开销要尽可能的小；

（8）容许在通信中丢掉个别的网页。

满足上述条件的API即在网页抓取子系统的基础上提供了一个开放的网页数据流平台，支持其他研究者进行实时网页分析，同时隔离了那些分析程序可能出错对抓取系统的影响。

2.2　实现技术的选择

根据上面提到的条件，我们选择使用无连接的UDP协议进行IP多播[3]。它通过网络传递数据，满足1、2条件。数据源只需要执行一次发送，多个接收者都能得到数据，满足4，5，6，7条件3可以通过增加网络硬件满足。由于网页抓取系统速度受到它所连接的互联网带宽的限制，需要在API中传递的数据速率并不是很大，使用现在主流的千兆交换机就绰绰有余。

同时我们也看到了它受到的重要约束：多播包需要路由器支持，才能跨路由器。不幸的是，出于安全的考虑，现在多数路由器禁用了IP多播。如果我们的通信API需要覆盖一个较大的范围，IP多播就难以行得通了。好在我们的网页抓取系统和网页分析程序只是分布在少数几个机房里，而且它们之间的路由我们能够控制。

由于UDP的本质决定[2]，我们这里的多播不是绝对可靠的。首先在网络中传输的时候就存在丢包或者错序的可能；其次发送者根本不清楚网络中有哪些节点在接收自己的数据，它发送了网页之后并不等待接收者的确认；另外，一段时间内如果某个接收者没有保持接收状态，那么它就会错过这段时间内的所有网页，而且没有挽回的机会。人们提出过一些可靠多播协议，比如PGM(pragmatic general multicast)[4]，在现有IP多播基础之上增加了出错检测和重传的功能，在提高了多播可用性的同时也增加了发送者的开销，由于还没有得到广泛采用，本文不做考虑。

3　实现要点与测试

API实现的技术细节主要来自参考文献［1］，它是网络编程的一本经典。

3.1　实现要点

本API采用C++由类CMultiCast实现，它的两个方法有三个主要功能，分别完成初始化、发送和接收：

（1）void CMulticast::init(const string &maddr, const string &serv, const string &ifname)；

使用API首先需要对CMulticast对象进行初始

化，maddr指定多播池使用的地址，serv指定端口号，如果在节点上有多个网卡或者网络接口，可以用ifname指定多播使用的网络接口。

（2）void CMultiCast::send(const CWebPage &page)；

在完成初始化之后就可以通过多播对象发送网页了，send()唯一的参数是需要发送的网页。这里的网页类CWebPage不仅仅是网页正文，它还包括了在抓取该网页过程中的其他有用信息：网页URL、网页抓取时间、返回网页的服务器地址、网页请求状态、HTTP应答头和HTTP应答体等多项内容，HTTP应答体用于保存网页正文。在send()的实现中，第一步是把网页类编码成一个字符串消息。然后由于网页大小不同，产生的消息长度也不一样，而通过多播每次发送出去的数据长度是有限制的，超出的话则一条消息需要分多次发送。在不同的平台上这个长度限制也不一样，从几千到几万字节不等。最后我们采取了一个非常保守的方法：把每个消息切分成若干个长度为1024的片段和一个长度小于1024的结束片段。一次多播发送一个片段，附带发送的信息还有该片段的长度、片段号和片段所属的消息号，这些信息用于消息在接收方的重新组装。

（3）bool CMulticast::recv(CWebPage &page, const CMultiCastSession* session)；

与send()把网页切分成片段编码成消息发送相反，recv()收到片段消息之后需要先进行组装，再对消息解码得到网页提交给调用进程。片段的组装采用了最简单的实现，如果出现片段的丢失或者错序，则丢掉整条消息。recv()要考虑消息会来自多个不同的发送者，它在多播对象为每个发送者都建立会话，用于组装消息并记录发送者的状态。recv()的返回值为本次接收消息是否成功，当成功时page即为接收到的消息；session指向与本次接收相关的会话。

多播会话类CMultiCastSession负责把片段组装成消息，并向调用进程提供有关会话的统计信息。它有下列数据成员：

（1）String from；会话另一端发送数据的节点地址和网络端口，以“addr:port”的形式存储在一个字符串中；

（2）Int MsgNo；当前消息的序号；

（3）Int totalMesgs；本次会话收到的消息总数；

（4）Int totalBytes；本次会话收到的消息字节总数。

除了前面的两个总数之外，多播会话对象还对最近1000个消息进行统计，作为即时信息提供给调用者。

（1）Int speedBytes；即时的数据接收速率，每秒钟接受到的字节数；

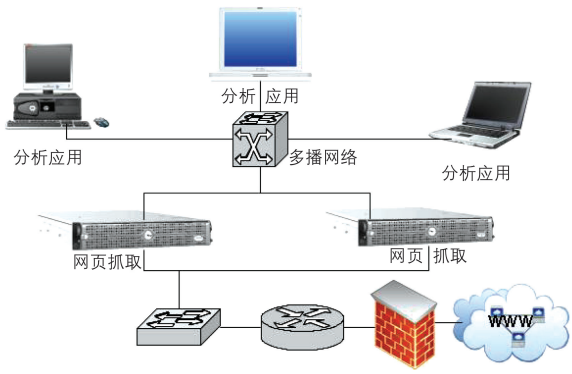
（2）Int lostRatio；即时的消息丢失率，每100个消息中丢失的消息个数。

在会话终结时，还提供本次会话数据接收的平均速率avgSpeed和丢消息的平均比率avgLost。

两个消息丢失率lostRatio和avgLost是衡量网页多播API可用性的重要指标，下面将主要针对消息丢失率讨论该API的可用性测试。

3.2 可用性测试

使用API在网页抓取子系统和网页分析程序之间通信的逻辑结构如图所示：



具体测试的设置稍有不同，我们没有为多播建立专门的网络，而是让抓取网络和多播网络共享了一个48口的千兆交换机，在该交换机上连接了十个网页抓取节点和三个多播接收节点。网页抓取节点为天网系统实际使用的机器，而三个多播接收节点分别为性能较好的PC服务器、性能中等的台式机、和性能最次的笔记本电脑。为了不干扰抓取系统的正常运行并模拟各种速度的网页多播，我们在每个抓取节点上另外运行了独立的网页投放程序，由它们向多播池中投放事先准备好的网页。为了排除分析应用对多播接收的影响，接收节点在收到网页之后只记录多播会话中的统计数据，不对网页进行额外的分析处理。

前文中提到了天网网页抓取子系统在不同阶段的网页抓取速度。对应每天抓取3000万到5000万网页的最高速率，我们设计了让十个网页投放程序同时以每秒40个网页的速度向多播池投放网页，这时在接收节点上观察到的网页丢失率在3%左右。对应增量搜集下每天抓取400多万网页的速度，我们设计了让十个网页投放程序同时以每秒5个网页的速度向多播池投放网页，这时在接收节点上观察到的网页的丢失率始终

在1%以下。

测试结果在不同性能的接收节点上表现一致，丢失率没有受到节点CPU性能的影响。在每秒接收400个网页的情况下，性能最差的一个节点（Pentium4 1.6GHz）的CPU占用率在10%左右，主要用于片段的组装和消息的解码。而在网页抓取节点上，网页投放程序的CPU占用率不超过1%。鉴于此，我们认为API给多播发送者和多播接受者带来的额外开销都是很小的。

我们也没有观察到一直保持运行的网页抓取系统受到了实验的影响。于是可以认为该网页多播API的可用性能够满足我们的需求。如果需要的话，通过提高交换设备的性能，可用性还有提升的空间。

4　动态词频显示：一个应用实例

根据上述API，我们在现有的网页抓取子系统中插入了多播发送网页的相关代码，并重新运行了网页抓取子系统。这时便可以随时异步从多播池中接收实时到来的网页流。为了体现这样一种能力，我们实现了一个网页动态词频显示应用，它包括下面的环节：

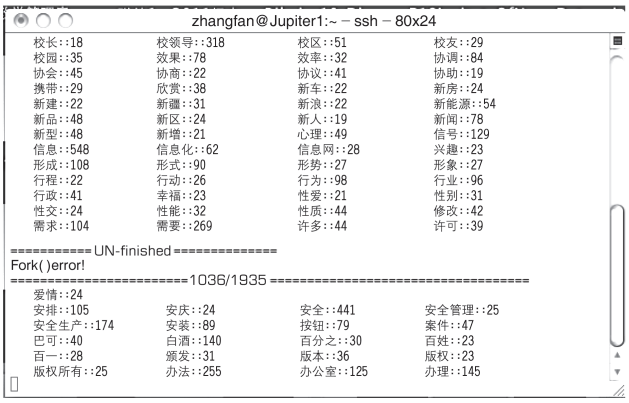
（1）从多播池中接收网页。在一个固定时间间隔T内收到的所有网页，我们称之为一帧；

（2）对各帧网页进行处理，提取网页正文，切词，对词频进行统计；

（3）向用户连续展示每一帧的词频数据，反映互联网网页词频的实时变化。

应用可以概括为接收、处理、展示三个环节。我们采用多进程的方式进行流水作业：多播池中的网页流是持续的，所以需要有一个监听进程专门负责对网页的不间断接收。它得到一帧网页后fork()一个子进程去处理这帧网页。与网页接收和结果展示的低CPU使用率不同，这里的网页处理需要进行中文切词，属于典型的CPU密集型任务，处理完一帧网页至少需要几十个T的时间。为了不让网页帧在处理环节积压，我们可以根据CPU的线程并发能力同时启动多个网页处理进程。如果启动了N个网页处理进程的话，我们限定每个进程处理一个帧的时间不能超过N*T，在限定时间不能完成的话，就把这一帧剩下的网页丢掉。子进程完成词频统计之后，把一帧网页中的top-K高频词的<词，词频>二元组输出到屏幕上。由于使用了多进程的并发结构，而输出设备只有一套，这样不同帧的输出结果可能交叉在一起，我们采用文件锁的方法避免了这种情况。

这样一个应用的运行情况如图所示，人们会看到各种词频信息源源不断地从下方流入，从上方流出，对应当前互联网上网页信息的状况。当然，也可以采用词云等信息可视化技术将它们展现得更吸引人些。



5　结　语

将搜索引擎网页搜集子系统产生的数据看成一个数据流，本文介绍了一个基于IP多播的流数据分发接口，包括设计思想、实现要点与一个演示应用。这样一个接口的实现，为各种网页流信息分析应用打开了一扇窗口；同时，该接口的设计思想也可以用于其他流数据的场合。我们特别希望指出，采用这套API的实时网页分析应用通常会面对的几个问题。第一，数据流是不间断的，应用程序需要安排一个进程或线程专门负责数据的接收。第二，分析网页的计算量通常很大，应用需要启用多个进程或者多个线程，以充分利用CPU的多核能力，不得已的话需要选择是丢弃部分数据或者降低计算的精度。第三，把数据分配给多个线程或者多个进程时，处理完成后还需要把结果汇总。还有一些后续工作可做，例如进一步降低流数据多播的数据丢失率。

致谢：此项工作得到虚拟现实技术与系统国家重点实验室开放课题项目支持。

参 考 文 献

[1] Stevens W R. Unix network programming v1: the sockets networking APIs [C] //Prentice Hall. 1998.

[2] Tanenbaum A S. Computer networks [M]. 4th edition: Pearson Education, 2003.

[3] http://en.wikipedia.org/wiki/IP_multicast

[4] Gemmell J, Montgomery T, Speakman T, et al. The PGM reliable multicast protocol [J]. Microsoft Research, 2003.