# 绪论

1.1 研究背景与意义

随着云计算、物联网、移动互联、社交媒体等新兴信息技术和应用模式的快速发展，推动人类社会不断地向大数据时代迈进。2010年，全球数据量就已经跨入了ZB时代，据IDC预测，至2020年全球将拥有35ZB的数据量，大量数据实时地影响我们的工作、生活，甚至国家经济、社会发展，大数据时代已经到来。大数据具有数据量巨大、数据类型多样、流动速度快和价值密度低的特点，大数据技术为我们分析问题和解决问题提供了新的方法和思路，其研究已经成为热点。一般意义上来讲，对大数据就是利用现有的理论知识以及各种技术手段和工具都很难再可以接受的时间之类完成数据分析计算，而整体都呈现价值巨大的海量复杂的数据集合体。与此同时，这些巨大的价值量往往都是隐藏在海量的数据当中，变现出了价值密度低、分布极其不规律、信息隐藏深度极深、发现价值极其困难等特点，这些特征必然会为大数据的研究和分析带来前所未有的挑战和机遇。

大数据的分析计算模式主要分为批量计算（batch computing）、流式计算（stream computing）、交互式计算（interactive computing）、图形计算（graph computing）等等。其中批量计算和流式计算这两种计算模式不管是在学术界还是在工业界都是主要的研究模式，同时各自都有广泛的大数据应用场景。其中批量计算是一种适用于大估摸并行批量处理作业的分布式计算模式，也就是我们大家都十分熟悉的MapReduce计算模式。MapReduce的简单易用性使其成为目前大数据处理最成功的主流并行计算模式。在开源社区的努力下，开源的Hadoop系统目前已成为较为成熟的大数据处理平台，并已发展成一个包括众多数据处理工具和环境的完整的生态系统。目前几乎国内外的各个著名IT企业都在使用Hadoop平台进行企业内大数据的计算处理。此外，Spark系统也具备批处理计算的能力。而流式计算是一种高实时性的计算模式，需要对一定时间窗口内应用系统产生的新数据完成实时的计算处理，避免造成数据堆积和丢失。很多行业的大数据应用，如电信、电力、道路监控等行业应用以及互联网行业的访问日志处理，都同时具有高流量的流式数据和大量积累的历史数据，因而在提供批处理计算模式的同时，系统还需要能具备高实时性的流式计算能力。流式计算的一个特点是数据运动、运算不动，不同的运算节点常常绑定在不同的服务器上。Facebook的Scribe和Apache的Flume都提供了一定的机制来构建日志数据处理流图。而更为通用的流式计算系统是Twitter公司的Storm、Yahoo公司的S4以及Apache Spark Steaming。

大数据时代下的流式计算呈现出了鲜明的高带宽、低时延的应用需求，传统的流式计算平台的构造往往是建立在传统的数据库的基础之上的，难以满足大数据流式计算的需求。如何构建一个低时延、高带宽、持续可靠、长期运行的大数据流式计算系统成为了当前亟待解决的问题。Redis这种基于内存计算的、可进行数据持久化的Key-Value存储系统的诞生，为大数据流式计算带来了很好的一个解决方案。Redis数据库最初是为了解决像SNS类网站在数据存取过程中的实时性等刚性需求的，而传统的关系型数据库越来越难以胜任了，这也使得redis这种数据库也越来越受到人们的关注。如今redis数据库已经得到了广泛的应用，不论是在高速缓存系统中，还是在海量文件的实时检索中，甚至是在如何如荼的各种推荐系统中，redis都起着中心地位的作用。Redis的基于内存的数据计算和高效的数据存储策略也能够很好的满足实时流计算问题中的低时延的刚性需求。因此，研究redis的内存计算以及存储策略并将其运用到实时流式计算模型中具有重要的意义和实用价值。

在流式数据处理中，它不像传统的批量数据处理，它无需事先存储数据，无法确定数据是什么时候到来以及按什么顺序到来，因此，不需要事先对流式数据进行存储，而是当流动的数据到来后在内存中直接进行数据的实时计算和分析。就像我们熟悉的Twitter的Storm、Yahoo的S4就是典型的流式数据处理框架，数据在任务拓扑中被计算，最后输出有价值的信息。目前这些流行的流式处理框架都有一个共同的缺点就是，没有一个方便的能够快速根据业务构建数据任务的拓扑计算流程，也就是我们所说的计算流（flow），同时也缺乏数据的流化功能。Node-red是基于Node.js的，可视化流程编辑框架，它允许开发人员仅仅使用一个基于浏览器的可视化界面流程编辑器来完成设备、服务器以及API应用的连接。Node-red本身是IBM Emerging Technology团队创建的一个新型开源工具，它允许用户通过组合各种部件来编写应用程序。这些部件可以是硬件设备、Web API或者是在线服务。Node-red被广泛用于物联网领域，实现数据的流式传输。在node-red中从数据的接入，到数据的解析分析，最后到结果的输出都是通过各种各样的节点来完成的，IBM Emerging Technology团队在开发这个工具的时候只引入了少量的大众化的节点，比如常用的http节点、tcp节点、udp节点、debug等数据输入输出节点，还有一些用于数据分析的节点比如sentiment节点，还有一些用于访问存储设备的节点；node-red除了原始已经提供的这些节点外，还运行用户自己按照开发原则开发自己需要的节点。为了能够充分利用node-red的可视化流程编辑的直观性，结合redis数据库的内存计算的特点，探索开发适应于流式数据分析的数据输入输出和数据处理节点，这对流式数据分析有着重要的实际意义。

1.2 国内外研究现状

1.2.1实时流数据处理模型的研究应用现状

大数据时代下的数据处理主要的两种方式就是实时流式处理和批量处理。实时流数据处理主要适合于那些无需事先进行数据存储，可以直接进行数据分析处理，实时性要求比较严格，但数据的准确度要求比较宽松的应用场景。而对于传统的批量数据处理，首先要进行数据的存储，然后再对存储的静态数据进行集中或者分布式计算。目前，对于传统的批量数据处理模型的技术和研究成果已经相对成熟了，最初有Google公司的MapReduce并行编程模型的提出，再有后来在开源社区的努力下开发的Hadoop系统为代表的批处理系统，都已经是稳定而高效的批处理系统。而对于流式数据处理模型的研究仅仅处于一个初级阶段，在早期关于流式数据的研究也主要集中在以数据库为中心而开展的，主要是研究了数据计算的流式化，数据规模也比较小，数据对象也比较单一，很难适应在大数据时代下流式数据处理所呈现出来的新特性。因为，在新时期的流式数据主要呈现出实时性、突发性、无序性等特点，对新的流式计算系统就有了更高更严格的要求。

在国外，Yahoo推出了S4流式数据处理系统，随后在2011年，Twitter也推出了自己的流式数据处理系统Storm,还有就是近年来开源社区新兴的MOA（Massive Online Analysis）、Spark Stream都是流式处理系统，这在一定程度上推动了流式数据处理的发展和应用。但是像S4、Strom这样的流式数据处理系统在可伸缩性、容错性、数据吞吐量等方面存在着明显的不足，而对于MOA，Spark Stream这样的系统，虽然功能和API十分丰富，但是在稳定性和易用性上不尽如人意。所以，如果构建一个低延迟、高吞吐、易用且能持续可靠地运行的流式数据处理系统，是一个亟待解决的问题。

在国内，目前关于流式数据处理模型的研究还比较少，但目前国内主要有百度公司自主研发的Dstream和TM实时计算平台，在学术界主要是有一下关于流式数据挖掘算法的研究。但是，流式数据的可视化分析已经在很多场景得到了应用，比如各大银行都陆续建立的大屏监控系统，就是实时地监控银行的业务状况、系统运行状况、用户行为分析等，又比如政府网站群的监控，也是通过实时监控网站的访问数据，分析用户的行为。在这些应用的背后，如何建立一个高效、稳定、易于维护的实时处理模型显得尤为重要。

文献[1]，

文献[2]，

1.2.2 Node-red的研究应用现状

Node-red作为一种在物联网时代的新型产物，是一种用来快速搭建物联网应用程序的流式处理框架，在信息无处不在的时代，Node-red也越来越受到业界的关注和研究。

它是由IBM Emerging Technologies团队发起的一个开源项目，其中Nick O’Leary 和Dave Conway-Jones工程师为Node-red的设计和开发做出了巨大的贡献。2013年，Node-red以开源项目的形式被发布，经过一年的发展，Node-red已经拥有了一大批活跃的用户和开发人员。Node-red依然是一个新型科技，时至今日，但凡用过Node-red的制造商、实验人员和一大批大大小小的公司，都已经见证了Node-red极具价值的应用之处。

在国外，IBM公司率先将Node-red应用起来，Node-red被集成到IBM公司的最新云产品Bluemix上。通过Bluemix提供的云服务，用Node-red来建立和管理一个实例（也就是一个应用流程），就可以实现消息的推送服务。Node-RED 的使用，与 Bluemix 中简单的 Push 服务相结合，使整个流程变得非常简单，需要调整的部分也少得多。

在国内，目前也有很多智能设备制造公司在使用Node-red，可以很方便地通过Node-red节点来控制硬件设备的状态，比如拿Node-RED搭配Arduino，是一个快速原型化的好用工具.例如控制RPI的某根脚位去点亮LED。只要简单的拉四个node，串一串再写一点程序代码即可做到。因为Node-red还在进一步完善当中，原始开发的节点可能很难满足实际的需求，所以，我们在运用Node-red来管理数据流程的时候，还需要自己开发需要的功能节点。在这一点上，目前在不少银行的业务监控系统中引入了redis的访问节点。

1.2.3 Redis的研究应用现状

Redis作为存储系统之中的后起之秀，由于其数据结构丰富、基于内存计算、支持网络又可进行数据持久化等特点，迅速为许多企业和开发者所爱戴。不论是在学术界还是在工业界，对Redis的研究都从未停止过。

Redis是由Salvatore Sanfilippo为实时统计系统LLOOGG量身定制的一个数据库，在2009年的时候将Redis开源发布，并开始于另外一位Redis代码贡献者Pieter Noordhuis一起继续Redis的开发，知道现在。随着Redis内存数据库的发布，短短几年内的时间就拥有了一个庞大的用户群体。在国外，像GitHub、Viacom、Pinterest等都是Redis的用户，Github利用Redis集群，来统计用户项目跟进状况。而在国内，新浪微博研究了Redis数据库的源码，搭建了有号称史上最大的Redis集群，实现了传统的SQL数据库难以实现的计数分析（counting）、反向缓存（reverse cache）、top 10 list等功能。近年来，也有不少银行，在自己的实时数据监控平台引入了Redis数据库，实现了数据的实时处理和分析，还有就是随着国家电子政务系统的逐渐推行，不少的地方政府也在自己的数据中心监控系统中引入了Redis数据库，来实现数据实时计算和处理。

文献[3]，

1.3 主要工作和研究内容

本文对大数据背景下流式数据处理过程中所遇到的挑战和难题进行了研究分析，详细研究了Node-red流式处理框架的编程模型和消息推送机制，Redis数据库的存储原理及其基于内存计算的原理。设计了一种新的基于Node-red的流式管理和Redis的内存计算的流式数据处理模型，并通过实现网站访问实时监控系统来验证了该模型的可行性。主要工作内容如下：

1. 本文对当下流行的几款流式处理框架Storm、Spark stream与Node-red流式管理框架进行了对比分析，详细阐述了各自的编程模型，同时对结合MQTT详细阐述了消息推送机制为什么适合流式数据处理。
2. 对Redis数据库做了深入研究，尤其是Redis的存储原理以及内存计算原理，还对Redis的Pub/Sub机制做了深入研究，阐述了在流式数据处理中为什么选择Redis作为数据计算和中间数据缓存的工具。
3. 在研究分析了流式数据的特点和流式数据处理的基本原理后，结合Node-red的编程模型和消息推送机制，设计了一种新的基于Node-red的流式管理和Redis的内存计算的流式数据处理模型。由于原始的Node-red缺乏对redis数据库的访问节点以及redis的pub/sub节点，重新设计了新的数据输入、输出节点以及数据处理节点，并成功发布到Node-red框架当中，实现数据的流式处理，和数据流的管理。
4. 本文最后还对设计好的流式处理模型，加以了应用验证。使用该模型对某政府网站的访问数据进行实时监控分析，设计了一套数据监控系统，包括了数据的采集，数据分析和处理，以及最后的数据可视化展示，并对结果进行了有效性分析。实现了从模型设计到模型应用的全过程。

1.4 结构安排

本论文共分为六章，其章节结构安排如下：

1. ，绪论，首先介绍了本论文的研究背景和意义，通过阅读大量相关文献和论文资料，总结了国内外流式数据处理模型的研究现状，以及Node-red、Redis的研究应用现状。然后简单的介绍了本论文的主要研究内容和全文的章节结构安排。
2. ，实时流数据处理的理论基础和技术，本章详细介绍了目前几个主流的流式处理框架的编程模型，有Storm的流式编程模型、Spark stream的流式编程模型、Node-red的编程模型以及他们各自的特点和不足之处。还介绍了MQTT协议以及在消息推送服务中的应用。最后还介绍了Node-red的异步非阻塞模式与事件驱动机制，以及Redis的存储技术和Pub/Sub机制。
3. ，基于Node-red和Redis的实时流数据处理模型的设计，本章首先对在实际场景中的流式数据处理应用做了详尽的需求分析，然后对模型的总体架构做了详细设计，最后对于Node-red中原本缺少的用于流式数据的输入和输出节点以及数据处理节点做了重新设计，并将设计的各个节点重新部署到Node-red框架当中，使其成为一个能够胜任流式数据处理的完整框架。
4. ，基于node-red与redis的实时流数据处理模型在网站访问监控方面的应用，本章主要是对设计的新模型加以应用，以此来验证模型的可行性。为此设计了一个实时网站访问的监控系统，数据处理就用到了我们设计好的流式数据处理模型，将数据处理的结果输出到前端页面Board做可视化展示。本章详细阐述了系统的功能，各个模块的设计与实现以及系统的展示。
5. ，系统测试与性能分析，这一章是整个模型的测试环节，主要是分析了模型对流式数据的处理能力，并与Storm和Spark stream流式处理系统在吞吐量和时延上做了对比分析。
6. ，全文总结与展望，是对本论文的主要工作进行最后总结，并对后续工作做了一些说明。

# 实时流数据处理的基础理论和技术

2.1 实时流数据处理的编程模型

2.1.1 storm的编程模型

2.1.2 spark stream的编程模型

2.1.3 node-red的编程模型

2.2 消息队列遥测传输协议MQTT

2.3 node-red可视化流式处理框架

2.3.1 node-red的概述

2.3.2 node-red的非阻塞模式与事件驱动机制

2.4 基于内存计算的数据库redis

2.4.1 redis数据库的概述

2.4.4 redis数据库的储存原理

2.4.5 redis数据库的pub与sub机制

2.5 本章总结