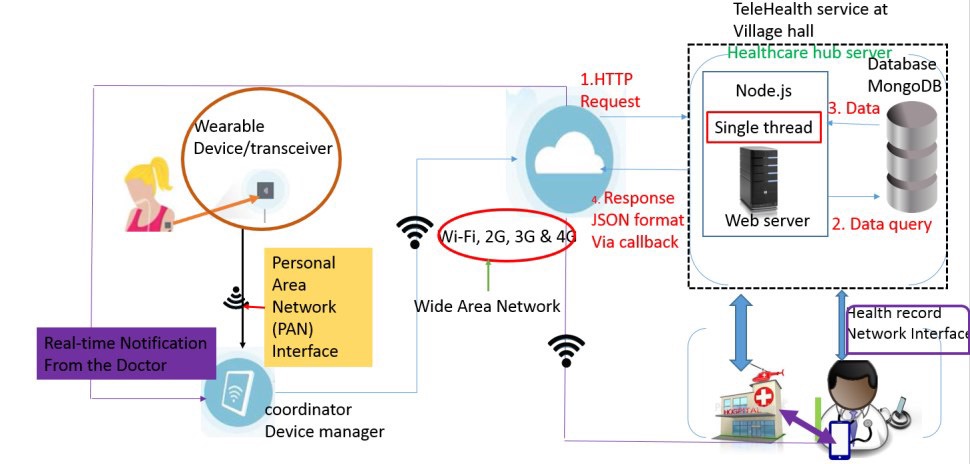
医疗中心服务器在远程医疗监控系统中的服务器端渲染javascript性能评估

1. **介绍**

像物联网医疗系统这样的产品将设备连接起来组成一个巨大的系统的点子是基于物联网（IoT）。这个物联网医疗系统是利用连接在一起的设备所形成的网络利用无线连接以及因特网访问能力共享有用的信息到云端的中央处理器以及控制服务器。

远程医疗监测（RHM）是一种技术，用于监测居住在一个孤立的村庄的患者，无法获得对常规临床设置的有效健康监测。然而，大多数RHM技术遵循由四个组件组成的通用架构。第一组件是通过无线通信以测量生理参数的各种设备传感器。第二组件是在患者站点处的协调器移动设备（智能电话）上的本地数据存储，其在传感器和集中式数据存储库与健康监护集线器服务器之间进行接口。协调器设备配备有无线连接，例如Wi-Fi或3G / 4G LTE启用的互联网，以向医疗保健提供者发送收集的生理数据。第三个组件是存储从传感器，本地数据存储，诊断应用程序和医疗保健提供者发送的数据的集中式存储库。最后一个组件是诊断应用软件，其基于通过触发先前由患者和医生商定的一系列动作基于对收集的数据的分析来向患者发送实时警报。在本文中，我们使用术语“保健中心监视服务器”来引用第三和第四部分。图1显示的是RHM系统的架构。

RHM系统的健康护理中心监视服务器旨在以并发方式处理来自患者的大量请求，以有效地处理治疗建议。并发性是在系统中同时发生的事情的趋势。并发性在RHM系统中特别重要，它们必须实时响应外部事件，并且通常有严格的最后期限来满足。随着患者请求的数量以同步的方式增加，位于村庄厅的保健中心服务器会遇到瓶颈。因此，该RHM应用不能检测患者健康的恶化，减慢患者和医生先前约定的预定义动作。作为后果，急诊部门访问次数增加，住院治疗和住院时间延长了医疗保健的交付成本。此外，患者不能接收实时警报，这是在医生的帮助下自动发生的一系列动作。这些警报提醒患者采取一些预防措施。当医生无法从医疗中枢服务器接收新数据和患者的历史时，这种情况可能导致患者的死亡。此外，患者不应受益于任何立即的医疗护理。



图一. 远程医疗监测系统架构在孤立的位置，无法获得当地医疗。

当我们谈论并发时，提出了两种方法。第一种方法是在单个进程中拥有多个线程。第二种方法是具有多个单次处理过程。使用单独的并发进程也有一个额外的优点。优点是在通过网络连接的不同机器上运行分离进程。虽然这增加了通信成本。另一种方法是在单个进程中运行多个线程。每个线程独立于其他线程运行，并且每个线程可以运行不同的指令序列。多线程方法的缺点是单个进程中的所有线程共享相同的资源。在共享资源失败的情况下，在单个进程中包括的活动线程也被破坏。

在本文中，我们提出了RHM应用程序的设计任务，用于处理并发任务。在设计任务的过程中，通过采用多级抽象来最好地理解并发。首先，RHM系统的功能需求必须被充分地理解以获得期望的行为 。接下来，应检查可能的并发功能。这最好使用线程的抽象。明显地实现并发的方式是构建面向对象的环境，支持消息在并发对象之间传递，并通过单个进程最小化或消除对多线程控制的使用。在软件开发中，用于开发服务器端Web应用程序的跨平台运行时环境Node.js具有用于设计RHM系统的技术。

这项工作的主要贡献如下：

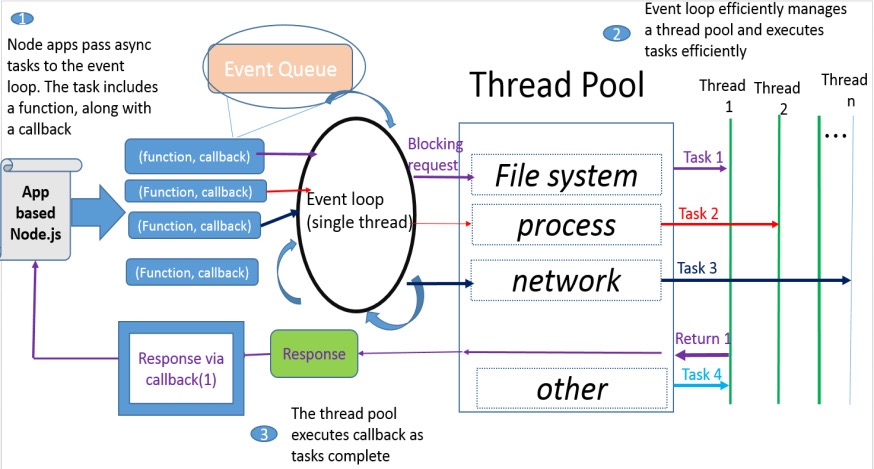
* 为RHM应用程序构建并行系统有意义的设计任务在位于村庄大厅的医疗中心服务器上运行。还描述了构成保健中心服务器的组件的设计。
* 为了研究包括后端服务器，Web编程框架和数据库的医疗中心监控服务器的性能，能够支持大量并且不断增加的并发患者请求。性能指标是吞吐量，响应时间和错误率，以便使用Google的V8 javaScript引擎和Rhino来比较服务器端的javaScript实现。

本文的其余部分安排如下。首先，我们讨论第二部分研究相关技术的背景。我们在第三部分中阐述了建立医疗中心服务器的一般系统设计模型。在第四部分，我们简要介绍模拟，实验和分析的结果，说明服务器端javaScript处理并发连接的远程医疗监护系统的性能。第五节中的结论总结了我们Node.js的性能，以及在部署需要高性能，异步，事件驱动的网络服务器且需求低的应用程序时应具有的优势。

1. **与研究相关的技术背景**

当涉及到服务器端JavaScript编程时，开发人员可以在Rhino javaScript引擎和基于V8的解决方案（如Node.js，SilkJS和其他）之间进行选择。 Rhino是一个基于java的JavaScript解释器，它允许JavaScript程序访问整个Java API。 Node.js是构建事件驱动的网络程序的平台。这是一个谷歌的V8 JavaScript引擎。它运行单线程，使其同时服务于多个客户端。如图1所示。 事件循环对新请求和阻塞I / O请求进行排队。单线程通过设置所有请求的简单映射来执行事件循环。事件循环逐渐从队列中出队请求，然后处理请求，最后获取下一个请求或等待新的请求提交。

使用Node.js允许编写在其设计中继承并发性的可扩展网络应用程序。这是由事件循环完成的。为了避免阻塞，涉及输入/输出的所有主要应用程序编程接口（API） - 调用是异步的。因此，事件循环可以执行队列中的下一个事件，而不是等待函数返回，而当API调用完成时，它调用在调用时指定的回调函数。有一个无限循环在单个进程中运行，持续查看发生了什么事件和需要执行什么回调。它处理自动排队所有事件，并且它尽可能快地调用适当的回调。作为使用Node.js的开发人员，你并不需要知道这一点，尽管你只需要知道当事件发生时尽快调用回调函数。参见图2。



图二. Node.js内部围绕着事件循环的概念。有一个无限循环在单个进程中运行，持续查看发生了什么事件和需要执行什么回调。它自动处理排队的所有事件， 保持调用适当的回调尽可能快。

1. **在医疗中心服务器上设计并发远​​程医疗监控**

由并发组件采用，软件和数据结构的块或程序都不能为并发组件创建非常自然的模型，但对象似乎是设计并发软件的一种非常自然的方式。

图3将对象图示为在医疗中心服务器上构建RHM应用程序的并发组件。考虑我们位于村庄大厅的RHM系统的对象模型。患者的状态数据对象允许收集患者的数据，然后与由医生（医生）对象设置的一系列参数进行比较。基于结果，我们的RHM的对象模型发送消息到一系列动作以运行对象，其执行先前由患者和医生商定的一系列动作。这里通过通知患者并更新医生（或医生）片剂的患者历史数据来执行动作，使得他们能够更好地评估患者的状况并且达到更准确的诊断。

为了支持异步通信，Node.js的当前运行实例侦听所有新的传入任务。运行实例的特征在于其上提供有消息队列的事件循环。运行实例引用活动对象。被动对象对应于针对Node.js应用程序运行的不同任务，例如由医生设置的参数范围，患者数据（图3）。被动对象包括与待被封装在被动对象中的任务完成时对应于由活动对象（线程池）执行的响应相对应的函数和回调。回调保存执行非阻塞和长任务的被动对象的状态信息。

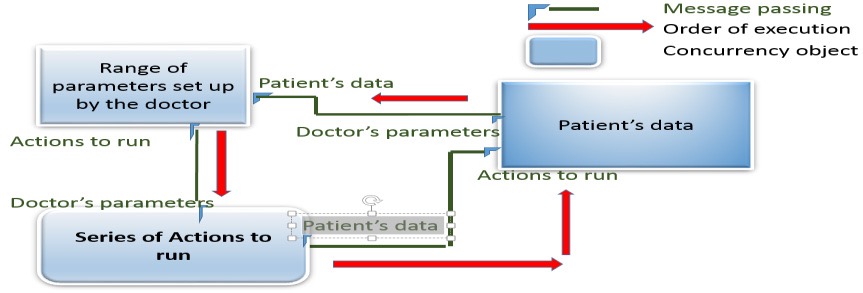


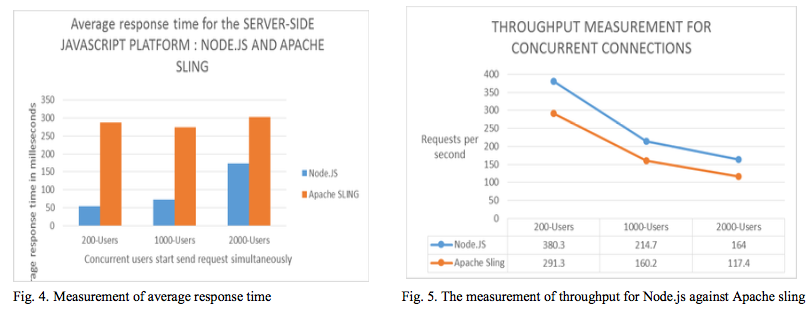
图 3. 在医疗中心服务器上设计用于RHM应用程序的并发对象。

1. **使用基于Node.js和实验结果构建医疗中心服务器**

测试计划使用JMeter11捕获吞吐量，Node.Js的单线程事件循环的响应时间结果与传统的基于应用程序的java在使用Apache sling的服务器端JavaScript解释器上的结果。需要进行容量和性能测试，以显示由后端服务器和数据库层组成的医疗中心监视服务器可以在大量并发用户可以同时访问这些后端服务器端和数据库时以可接受的响应速度运行。

对于每个客户端 - 保健中心服务器架构，目标不是测试web应用，而是以与从web浏览器提交http请求相同的方式监听从移动设备发送的http请求。被测应用程序是即时心率。它使用智能手机内置相机来跟踪指尖上与您的脉搏直接相关的颜色变化。这个android的应用程序是基于移动客户端服务器计算，其具有将存储在临时数据存储（如SQLite）上的准确速率数据上传到远程医疗监护应用服务器的模块。第一个客户端 - 保健中心服务器架构的测试环境包括确定如何创建，显示，存储，更改数据和数据层的逻辑层 。

第一个测试针对Node.js和Apache吊带同时从200个用户到2000个用户，50次。对于每个测试，“加速”时段是零，这意味着所有用户同时开始发送http请求。



我们可以看到图4。是响应时间随着并发请求数的增加而降低。例如，Node.JS-MongoDB在200个并发用户的平均响应时间为54ms，平均为2000个并发请求的响应时间为173ms。我们可以看到，对于服务器端的Apache-sling，平均响应时间与并发请求数几乎呈线性相关。这意味着并发用户数千倍的增长开始响应时间增加100倍。 。示出随着用户数量的增加，两个模型的吞吐量恶化。结果表明，对于与MongoDB相关的Node.js，

吞吐量为每秒200个并发用户的380个请求，以及2000个并发用户每秒164个请求。使用V8运行时引擎的基于Node.js的事件驱动方法的吞吐量比使用rhino作为基于服务器端基于JavaScript的Java的Apache-sling性能更好。使用Apache-sling-MongoDB配置，吞吐量可增加最多1000个并发用户，然后随着并发用户数量的增加而恶化。这导致了在服务器端用于构建web服务的Apache-sling所执行的并发用户的数量不是相对恒定的。因此，Node.JS大约快40％，例如对于对应于十万（100000）并发请求的2000个用户，每秒164个请求/秒，对于117个请求。

1. **结论和未来的工作**

客户端 - 保健中心服务器由Node.JS服务器构成，JavaScript端和MongoDB的速度比基于Java的服务器端JavaScript解决方案快40％，使用Apache吊带与MongoDB数据库实现移动客户端服务器计算相关的远程医疗监护系统。在本文中，单线程事件循环Node.js和多线程方法之间的差异并发模型有所不同。为了测试Node.js更高的并发级别，它应该超越多线程，其他问题，如增加请求的数量。对于未来的工作，我们将使用node.js实施RHM系统，该实时收集患者的数据，基于收集的数据的分析来控制治疗建议和干预。