基于海量医疗数据的症状自查服务的云框架设计

周作建[1, 2+] 林文敏[1, 2] 王斌斌[1,2] 潘金贵[1, 2]
[1]南京大学 计算机软件新技术国家重点实验室, 南京 中国 210046
[2]南京大学 计算机科学与技术系, 南京 中国 210046
+lygzzj@163.com

Design of the Cloud Framework for the Symptom self-Inspection Services based on the Massive Medical Data

Zhou Zuojian[1,2], Lin Wenmin[1,2], Wang Binbin[1,2], and Pan Jingui[1,2]
[1] (State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China)
[2] (Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China)

Abstract With the increase of the sub-health in current society, symptom self-inspection services have become more and more important. However, the establishment of the regional health cloud platform based on the EHR provides the data support of the symptom self-inspection services to us. For example, we can find similar EHRs through this platform. As proposing the symptom self-inspection services based on the cloud framework, we face the challenge that a large amount of the EMRs should be acquired, stored, searched and analyzed. To solving those problems, we propose the symptom self-inspection services based on the cloud framework. Specifically, we build a Hadoop cluster to store and retrieve the massive medical data so that we can improve the response time of searching the EMRs. Firstly, we design the distributed search node cluster based on the Lucene project to retrieve, analysis and filter the massive EMRs in real time. Secondly, we discuss the implementation of the symptom self-inspection services which includes the selection of the searching nodes, the indexing of the EMRs, the method of calculating the similarity of EMRs and the sorting Algorithm. In the end, an experiment has been conducted which proved the scalability and effectiveness of our cloud framework.

Key words Cloud Framework; Symptom Self-inspection Services; Massive Medical Data; EMR; Hadoop; Lucene

摘要 随着当前社会“亚健康”人群的增加，症状自查服务显得愈发重要。各地基于居民健康档案的区域卫生信息平台的建立，为症状自查服务实现提供了数据基础，但是我们仍面临着海量电子病历的获取、存储、搜索以及数据分析计算等诸多挑战。鉴于上述问题，本文提出一种基于云框架的症状自查服务模型。首先，本文建立了 Hadoop 集群，用来对海量医疗数据的存储以及索引的建立，提高电子病历的搜索响应时间。其次，本文设计了基于 Lucene 的分布式搜索节点集群，用来对海量的电子病历进行实时检索、数据分析和隐私过滤。此外，本文对症状自查服务的实现进行讨论，包括搜索节点的选择、病历索引文件的建立、病历相似度的计算及排序方法。最后，本文通过实验证实症状自查服务的云框架模型具有可扩展性和有效性。

关键词 云框架；症状自查服务；海量医疗数据；电子病历；Hadoop；Lucene

中图法分类号 TP391

收稿日期：2014-11-15
基金项目：计算机软件新技术国家重点实验室自主课题（ZZKT2013B11）；
江苏省省级现代服务业（软件产业）发展专项引导资金资助项目（2013771）

1. 引言

随着医疗卫生行业计算机信息化水平不断提高以及医疗卫生业务体系不断发展和完善，人们对医疗健康的信息化要求也越来越高。一方面，当前社会的“亚健康”人群越来越多。根据世界卫生组织的报告，“亚健康”的人群占到世界人口的 75%[1]。在中国，“亚健康”的人群也达到了 9 亿 [2]。另一方面，各级医疗机构需要将每天产生的医疗数据上传到区域卫生信息平台。通过对海量医疗数据的存储、分析和检索，为实现症状自查服务提供了有效的数据基础。因此，无论是“亚健康”还是患病的人群均希望通过信息技术手段进行必要的自我诊断，获取类似患者的相关治疗方案，并进一步加深患者对疾病的了解[3][4][5]。

由于云计算具有可扩展性、面向服务等显著特点，它被广泛地应用于大数据处理领域[6][7]。作为云计算技术中的佼佼者，Hadoop 是近几年发展比较成熟的云计算平台之一[8]，由于其高容错性、高性能及其低成本的优势，得到了有海量数据处理领域的广泛应用，同时也得到了学术界的关注。Hadoop 是一个分布式文件存储系统，可以运行在普通的计算机硬件设备上[9]，它强调在低成本的条件下实现大数据的处理能力，并且可以通过数据节点的简单增加实现存储能力的线性增长，从而大幅度降低了数据存储成本。Hadoop 的出现为海量的、异构的医疗数据提供了全新的、高效的存储、查询、分析和挖掘方法。

本文充分利用区域卫生信息平台的数据资源，通过云框架实现对海量的、异构的医疗卫生数据的分析、存储和管理，实现症状自查服务，合理分配医疗资源，更好的提高居民健康服务质量。

2. 相关知识

2.1 Hadoop：HDFS 和 MapReduce

对海量医疗数据存储、计算处理的传统方式通常采用两台高性能小型计算机进行双机热备份操作，因此需要昂贵的硬件资源投资。而 Hadoop 利用大量低成本计算机实现对大数据的分布式存储和并行计算，正好可以解决上述问题。Hadoop 主要由分布式文件系统（Hadoop Distributed File System，HDFS）、MapReduce 编程模型两大核心组件组成[10][11]。

HDFS 是一个可扩展的分布式文件系统，系统构建在低成本的计算机硬件设备上，能够为大量并发用户提供可靠的服务，具有低成本、高可靠性、高吞吐量等特点。HDFS 中的文件通常被分割为多个数据块（默认的文件块大小为 64MB）存储在多个数据节点上。

MapReduce 是一种利用底层分布式计算环境对大数据进行并处理的计算模式，它包括两个过程 Map 和 Reduce。Map 过程就是将以“键值对”形式的文档数据记录传入 Map 函数，Map 函数将处理这些键值对，并以另一种键值对形式输出。Reduce 过程就是由 Map 输出的一组键值对将被进行合并处理，即将同样主键下的不同数值合并到一个列表中，对传入的中间结果列表数据进行某种整理或进一步的处理，并产生某种形式的最终输出结果。

2.2 Lucene：全文检索库

Lucene 是一个高性能、可伸缩的全文索引引擎工具包，通过提供的 API 接口可以方便的集成到各种应用程序中从而实现全文检索功能。它由 Apache 软件基金会支持和提供[12][13][14]。

Lucene 建立关键词与文档的映射关系索引，逻辑上合并多个索引的精确查询，从而实现了全文检索。众所周知，电子病历是患者在临床就诊过程中产生的各种文档的集合，常以 XML 的格式进行存储。本文利用 Lucene 对需要检索的电子病历文档进行预处理，建立关键词、病历文档编号、出现频率之间的映射关系，并将这种映射关系存储在索引文件中，从而实现对电子病历的全文检索。

3. 症状自查服务的云框架

3.1 云框架的概述

本文提出的云框架充分的利用了云计算的众多特点，如：松耦合、弹性计算、海量存储等。采用云框架模型的主要目的是对海量医疗数据进行存储和分析，为最终用户提供业务服务。具体框架如图 1 所示。

图 1 所描述的云框架中，主要包括 Hadoop 集群和分布式在线集群。Hadoop 集群主要用来对海量医疗数据存储及计算。HDFS 用于存储电子病历、PACS、继续教育视频、电子病历索引等海量医疗数据。MapReduce 主要用于并行创建电子病历索引文件。为了保证用户实时检索的高并发性，MapReduce 在建立病历索引文件时，将索引文件分成 N 片索引（N 是搜索节点集群中行服务器的节点数量），并将第 i 片索引放到搜索节点集群中每行的第 i 个服务器节点上。

分布式在线集群主要用于实时的处理用户提交的查询集合，包括症状相似度的计算及相关病历排名等。为了保证高并发性，搜索节点集群是由 M×N 个服务器节点组成。分发服务器采用集群的方式，通过负载均衡设备将用户请求分配到相应的节点上，减少

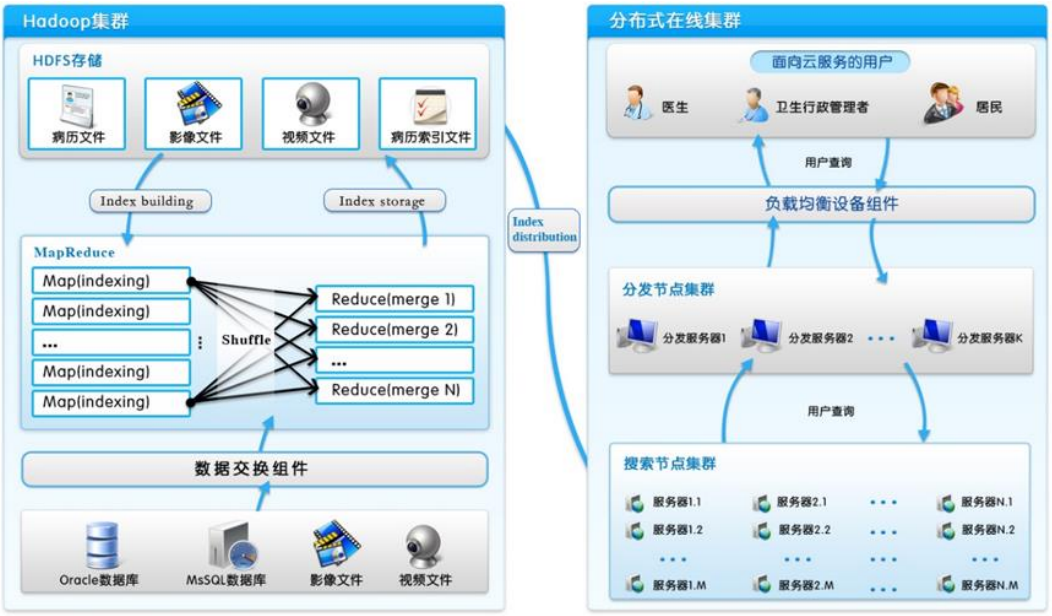

图 1 症状自查服务的云框架图

用户等待响应时间。

3.2 Hadoop 集群

医疗卫生机构信息系统数据源包括 Oracle 数据库、MsSql 数据库、影像文件、视频文件等，通过数据交换（DataExchange）组件准确实时地传输到 Hadoop 集群上，MapReduce 并行计算编程模型实现电子病历索引文件的建立，并将已建立的电子病历索引文件进行存储，以便快速的响应用户提交的疾病症状的进行查找和分析。

图 2 描述了 MapReduce 实现电子病历索引文件的建立流程。首先，用户程序中的 MapReduce 程序将输入的电子病历文件分为几个病历片段，Hadoop 集群启动很多程序副本，其中的一个程序副本被指定为主控程序，其余的为工作机。主控程序指定多个空闲的工作机运行 Map 任务，对所分割的病历片段进行索引文件的建立。将每个 Map 节点上建立的索引文件分割成 N 份（N 是搜索节点集群中行服务器节点数量）并定期写入每个 Map 工作机的本地硬盘，在本地硬盘的位置信息会发送到主控程序。其次，由主控程序通知 Reduce 工作机，Reduce 工作机在读取到所需要的全部中间数据后，对数据进行排序，并按照指定的 Reduce 函数进行处理，N 份索引片段文件建立完成后，将存储在 HDFS 中。此外，并将 N 份索引文件存放到搜索节点中。

3.3 分布式在线集群

分布式在线集群主要由负载均衡、分发服务器集群、搜索节点服务器集群三个部分组成。下面讨论每个部分的主要作用。

● 负载均衡

负载均衡就是把大量并发访问或数据流量分担到分发服务器集群上进行处理，减少用户等待响应的时间。此处的负载均衡是硬件资源，用户查询请求后，根据一定的处理规则，将相关请求转发到相应的分发服务器集群中的节点。

● 分发服务器集群

为了支持高并发性，分发服务器集群由一组 K 节点服务器组成。每个服务器节点负责响应和转发到搜索节点集群。具体来说，用户查询请求通过负载均衡设备转发到分发服务器集群中的节点，分发服务器节点将选择搜索节点服务器集群中的每列搜索节点来执行记录检索。每个搜索节点返回的结果进行合并，再提交给搜索节点集群进行处理。在所有数据返回给最终用户之前，并进行隐私权限处理。从图 2 我们可以看出，每个集群都包含多个节点，因此，对于每个用户查询，节点选择算法选择最佳的节点进行执行用户查询。节点选择算法将在症状自查服务的实现中进行讨论。

● 搜索节点集群

搜索节点集群由弹性的、可扩展的 M×N 搜索矩阵构成，矩阵中的节点存放索引片段并用来管理实时的电子病历数据检索，采用 Lucene 全文检索库的方式进行。N 是列的数量，M 是行的数量。第 i 的节点存放索引中的第 i 片段。为了提高电子病历的高并发性，每行中的节点复制上一行中所对应节点的数据。对于每个用户查询，每列的一个搜索节点的选择是由分发服务器集群确定的。通常，N 值的大小是由索引

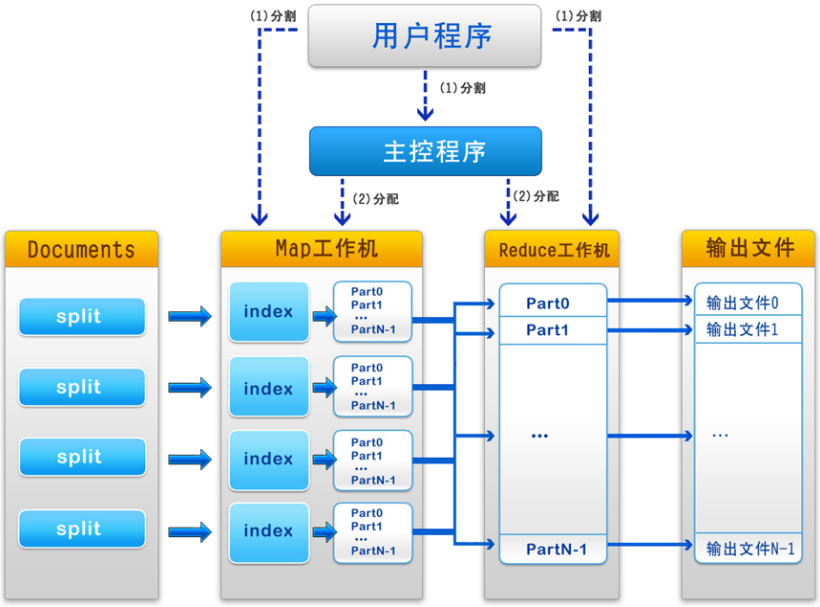

图 2 MapReduce 实现病历索引文件流程

文件的大小和搜索节点机器中内存容量的大小决定的。M 值的大小是由用户查询并发数量决定的。

4. 症状自查服务的实现

4.1 搜索节点选择算法

对于每个用户查询，分发服务器集群将选择搜索节点服务器集群中的某个节点来提供服务。对于每个节点的选择，最简单有效的方法就是采用循环队列进行依次选择。然而该调度算法扩展性较差，因为它没有考虑每个节点服务性能的差异性。在搜索节点服务器集群中，每行 i(0<i≤N)个搜索节点的索引文件集合构成了一个完整的索引文件，每列 j(0<j≤M)个搜索节点存储完全相同的索引文件。因此，每个服务节点可表示为 $Node_{i,j}$。本文提出一个公平的动态节点选择算法，它根据服务器的处理性能优先选择合适的节点以满足不同用户提出的服务。该算法主要涉及以下几个参数：

定义 1：服务完成时间（Service Completion Time, SCT），搜索节点从预计开始执行任务到结束所用的时间。不同的节点完成相同的服务所需要的时间是不同的。因此，我们只能依据以前服务运行时间和该节点其他就绪服务的服务预计运行时间进行估算。服务节点 的服务完成时间 由公式 1 表示：

$$SCT_{i,j}(S_c)=SCT^{e}_{i,j}(S_c)+\sum_{l=1}^{L_{i,j}}SCT^{e}_{i,j}(S_l) \quad (公式\ 1)$$

其中，S_c 表示当前执行的服务，$SCT^{e}_{i,j}(S_c)$ 表示节点 $Node_{i,j}$ 运行当前服务 S_c 的平均服务完成时间，$L_{i,j}$ 表示节点 $Node_{i,j}$ 服务队列中待执行服务的数量。SCT 值越小，节点 $Node_{i,j}$ 就能越快完成服务。

定义 2：服务失败率（Service Failure Rate, SFR），搜索节点在一段时期内没有响应查询的数量比例。节点 $Node_{i,j}$ 的服务失败率 $SFR_{i,j}$ 由公式 2 表示：

$$SFR_{i,j}=\frac{\sum f}{\sum s} \quad (公式\ 2)$$

其中，$\sum f$ 表示该节点服务失败次数之和，$\sum s$ 表示该节点服务总次数。$SFR_{i,j}$ 的值越小，节点 $Node_{i,j}$ 的服务质量越好。

定义 3：服务完成效率（Service Completion Efficiency, SCE），综合考虑搜索节点的服务完成时间和服务失败率。节点 $Node_{i,j}$ 的服务完成效率 $SCE_{i,j}$ 见公式 3：

$$SCE_{i,j}=SCT_{i,j}(S_c)\times\omega_1+SFR_{i,j}\times\omega_2 \quad (公式\ 3)$$

其中，； ω_1、ω_2 分别是服务完成时间和服务失败率

的权重，并且 $\omega_1+\omega_2=1$。$SCE_{i,j}$ 的值越小，节点 $Node_{i,j}$ 的服务完成效率越好。

基于公平的搜索节点选择算法的原则是保持节点之间的负载均衡，提高节点的整体运行性能。算法 1 搜索节点选择算法，具体依据以下的内容设置：

- 如果用户考虑服务快速完成，则可优先考虑服务完成时间参数。将服务安排到 $SCT_{i,j}$ 值最小的节点上执行，如果多个节点值相同，可再考虑服务失败率参数。
- 如果用户考虑服务成功执行，则可优先考虑服务失败率参数。将服务安排到 $SFR_{i,j}$ 值最小的节点上执行，如果多个节点值相同，则可再考虑服务完成时间参数。
- 如果用户既考虑完成时间又考虑服务成功情况，则考虑服务完成效率参数即可。将服务安排到 $SCE_{i,j}$ 值最小的节点上执行，如果多个节点值相同，可考虑服务执行次数最多的节点上执行。

算法 1. 搜索节点选择算法

Input: Select-Mode 服务节点选择模式
Min-heap[1…*N*] 查询条件拆分集合
Output: *SA* 所选择的服务节点集合

```
1.begin
2.  SA =Φ;
3.  if (Select-Mode == 1)
      //优先考虑服务快速完成
4.     for each Min-heap[i] (1≤i≤N) do
5.        pop out search node sn_i-j with smallest SCT_i-j in the Min-heap[i];
6.        add sn_i-j into SA;
7.        adjust SCT_i-j in the Min-heap[i];
8.     end for
9.  end if
10.  if (Select-Mode == 2)
      //优先考虑服务成功执行
11.    for each Min-heap[i] (1≤i≤N) do
12.       pop out search node sn_i-j with smallest SFR_i-j in the Min-heap[i];
13.       add sn_i-j into SA;
14.       adjust SFR_i-j in the Min-heap[i];
15.    end for
16.  end if
17.  if (Select-Mode == 3)
      //优先考虑服务完成效率
18.    for each Min-heap[i] (1≤i≤N) do
19.       pop out search node sn_i-j with smallest SCE_i-j in the Min-heap[i];
20.       add sn_i-j into SA;
21.       adjust SCE_i-j in the Min-heap[i];
22.    end for
23.  end if
24.  return SA;
25. end
```

4.2 建立病历索引文件

电子病历中记录了医生对患者进行诊断依据的主要描述，它包括症状、体格检查及实验室检查结果等。例如，一个病历中可能包括超过 20 个症状描述，对于每种疾病临床的表现症状也可能超过 200 多种。而每一种的描述从本质上来说都是文本的。因此，本文利用 Hadoop 离线集群为电子病历建立两种类型的索引：病历倒排索引和病历概要索引，以便快速响应用户提交的查询。

- 病历倒排索引

病历倒排索引是实现“症状-电子病历文档矩阵”的一种具体存储形式，通过倒排索引，可以根据症状快速获取包含这个症状的电子病历列表。然而，对于一个大规模的电子病历集合来说，这就需要高效的数据结构来对疾病症状进行构建和查找，以便实现用户快速的查询。文献[15]和[16]使用 Bloom-filter 提高疾病症状的查询。文献[17]证明 Hash 函数个数 K、位数组大小 m、加入的字符串数量 n 的关系。

Bloom-filter 算法使用的核心方法是 Hash 编码，用 BF 表示，考虑到误判率和搜索成本，我们采用的方法是分别对人体部位和症状进行 Hash 编码：
(1)创建一个 m+n 位的位数组(m：人体部位对应的长度，n:症状对应的长度)，每一位都置为 0；

(2)选择 k 个不同的 Hash 函数 $h_1,h_1,\cdots,h_k$ 对数据集合映射到长度为 m+n 位的向量 V 中，所对应 m+n 的位置置 1。例如：症状集合={发烧、咳嗽、胸痛、呼吸困难}，假设 m+n=4+10,经过 Hash 函数计算，h1(胸部)=3，h1(发烧)=3，h1(咳嗽)=5，h1(胸痛)=6，h1(呼吸困难)=9。h2(胸部)=3，h2(发烧)=2，h2(咳嗽)=3，h2(胸痛)=7，h2(呼吸困难)=8。则 h1、h2 的 4+10 位的向量值为 01000100110100 和 01000011000110，对二者进行与操作则有 01000111110110。因此，电子病

历中的症状集合 BF 签名就是 01000111110110。

用户提交查询症状时，针对查询提交的症状集合首先通过 Bloom-filter 算法进行构造 sq，一一比较电子病历中的 BF 签名 sd。如果 $sq \wedge sd = sq$，用户要查找的可能在集合 sq 中。灵活的位操作可以有效地提高程序运行的效率。

BF 的索引文件是以键值对的形式存在的和以序列文件方式存储在 HDFS 上。每个值都包括电子病历的系统标识号和症状集合的原始数据。通过倒排文件中的 BF 签名的查询，大量不合格的集合将会被剪掉。然而，剩余的数据集合仍然并不准确。因此，我们还需要针对用户提交的症状集合与本次查询的数据集合进行相拟度和排序计算，以确保完全匹配的原始症状查询结果。图 3(a)描述了病历倒排索引的样式。

- 病历概要索引

病历概要索引用于存放“电子病历文档-疾病临床表现”数据对象，或称“对象-属性”结构。在向用户提交相关返回结果时，病历概要索引被用来进行最后的相似度和排序计算。图 3(b)描述了病历概要索引的样式。

4.3 相似度和排序计算

搜索节点将用户提交的查询条件与查询初步结果的相似度进行计算。典型地，如果两个对象 O_i 和 O_j 不相似，则它们的相似性度量将返回 0。相似性值越高，对象之间的相似性越大。由于关键词为多值特征集合，本文利用 Dice 系数[18]计算病历文档的相似度。如果每份电子病历由一组症状关键词 $key(D_i)$ 描述，那么 Dice 系数计算病历文档 D_i 和 D_j 之间的相似度用公式 4 表示：

$$Dice(D_i, D_j) = \frac{2 \times |key(D_i) \bigcap key(D_j)|}{|key(D_i)| + |key(D_j)|} \text{ (公式 4)}$$

由于不同的等级医院所书写的电子病历文档质量相差很大，对于病历的相似性有所影响。因此，本文定义了医院等级权重，用于修正相似度的值。

定义 4：医院等级权重(Hospital Grade Weight, HGW)，每份电子病历文档提供者的医院等级所占的权重。不同等级医院的权重不同，经验取值：三级甲等 0.3，三级乙等 0.25，三级 0.2，二级甲等 0.15，其它 0.1。修正的病历文档相似度可用公式 5 表示：

$$Dice'(D_i, D_j) = Dice(D_i, D_j) \times HGW \text{ (公式 5)}$$

因此，上述相似度和排序 方法能够充分考虑到症状集合的不同维度，进一步解决病历文档中混合结构化及非结构化描述的问题，从而向用户提供可匹配的查询结果。

5. 实验评估

5.1 实验环境及实验流程

我们实现基于海量医疗数据的症状自查服务的云框架，我们建立了一个 Hadoop 集群和分布式在线集群，具体系统配置如表 1 所示。Hadoop 集群主要用于电子病历的存储和索引文件的建立。分布式在线集群用来实现用户的实时提交检索。

Hadoop 集群（HDFS： http://10.10.100.10:50070 和 JobTracker 节点 http://10.10.100.10:50030，VPN 网络）由 7 台节点组成（一台 Master 节点和 6 台 Slave 节点），每个节点配置 2 颗 6 核 Intel 主频 1.86GHz 处理器和 32G 内存。集群服务器运行 Redhat Enterprise Linux Server 6.0, Java 1.6.0 以及 Hadoop-0.20.205.0[19]。分布式在线集群由 19 台 PC 构成，运行的系统软件是 Ubuntu 9.0, Java 1.6.0 和 Lucene-4.5.1，实现用户实时查询处理。其中搜索节点共 15 台，组成 3 * 5 的搜索矩阵。另外 4 台用于实现分发服务器集群。所有的方法实现都是采用 Java、标准的 Hadoop MapReduce API 函数和 Lucene API 函数。

表 1 云框架的系统配置清单

	分布式在线集群	Hadoop 集群
硬件	**3*5 搜索节点** **3 个分发服务器** IBM X3850 X5 机架式服务器 2 颗 6 核 Intel 主频 1.86GHz 处理器，2G 内存，3*300GB SAS 硬盘，冗余电源，2 块千兆网卡，1 块 HBA 卡	**Masters 节点** **(NameNode, JobTracker):** CPU: 2 颗 6 核主频 1.86GHz； 内存: 32G 内存； 硬盘: 3*300GB SAS 硬盘； **Slaves 节点(6 nodes)** CPU: 2 颗 6 核主频 1.86GHz 处理器； 内存: 32G 内存； 硬盘: 3*300GB SAS 硬盘
软件	Ubuntu 9.0, Lucene-4.5.1	Red Hat Enterprise Linux Server 6.0; Java 1.6.0; and Hadoop-0.20.205.0

5.2 实验结果及性能分析

我们分别从并发性和索引文件的大小来验证云框架的可扩展性。我们针对 86358 份电子病历，每份电子病历平均大小为 170KB，总共大小为 14GB。通过 Hadoop 集群建立了病历倒排索引文件、病历概要文件，文件大小为 4.2GB。每份索引文件被分成 N 份(N 是搜索节点集群中每行服务器的数量),并分布在

每行中的每个节点上。

● N 值对性能的影响

第一种测试用例中，M 的值固定为 1，设置不同的 N 值(N=1,2,3,4,5)来评估云框架的扩展性能。在每一次实验中，通过 LoadRunner 模拟在一个客户端中初始 2 个并发线程，连续地发送 50,000 查询到分发服务集群节点。表 2 记录每次实验的平均延迟、CPU 利用率、I/O 等待的统计结果。

表 2 N 台搜索节点读/写性能测试统计

N	索引文件/搜索节点(GB)	平均延迟时间(ms)	CPU 利用率(%)	I/O 等待(%)
1	4.2	1209.6	21.5	9.7
2	4.2/2=2.1	161.3	35	8.7
3	4.2/3=1.4	18.3	80	0.2
4	4.2/4=1.05	17.5	90	0.1
5	4.2/5=0.8	17.3	93	0.1

通过表 2，我们发现当 $N \le 3$ 时，平均延迟时间和 I/O 等待时间随着 N 的减少而急剧的减少。CPU 的利用率随着 N 的增加而急剧的增加。当 $N > 3$ 时，各项指标相对趋于稳定。这是因为，当 $N \le 3$ 时，每台搜索节点的内存（2GB）同索引文件的大小相比是较小的。在这种情况下，大多数时间都浪费在 I/O 交换操作上，形成了 CPU 利用率较低，平均延迟时间就远远高于 $N > 3$ 的情况。当增加搜索节点，从表中也可以看出，达到一定的值后，各项指标的变化趋于稳定。这是因为，节点增加，分布在每个搜索节点的索引文件就越来越小，I/O 等待就越来越小，CPU 就无需等待 I/O 操作，因此再增加节点就造成了资源的浪费。事实上，N 的值的大小是由索引文件与内存容量的比值决定的，增加每行的节点数量应根据此比值。

● M 值对性能的影响

在此测试用例中，N 的值固定为 3，设置不同的 M 值（M=1,2,3,4,5,6）来评估云框架的扩展性能。同样，通过 LoadRunner 模拟在一个客户端中初始 6 个并发线程，连续地发送 50,000 查询到分发服务集群节点。平均延迟、CPU 利用率、I/O 等待的统计结果如表 3 所示。

表 3 M 台搜索节点读/写性能测试统计

M	平均延迟时间(ms)	CPU 利用率 (%)	I/O 等待(%)
1	43	98	0.2
2	26.7	98.1	0.2
3	17.4	70.3	0.2
4	17.2	60.3	0.2
5	16.4	48.4	0.2
6	15.9	41.2	0.2

从表 3 可以看出，I/O 等待时间并没有受 M 值的影响，这是因为索引的大小固定在 4.2/3=1.4G，CPU 无需等待 I/O 交换操作。平均延迟时间和 CPU 利用率的值随着 M 的增加而减少。当 $M \le 3$ 时，CPU 的利用高而平均延迟时间较长，形成系统的瓶颈。

为了改善此情况，通过增加搜索节点集群中的每行数量。但也并不是 M 值越大性能价格比越好，从表中可看出，当 $M > 3$ 时，平均延迟时间并没有出现显著的提高。

基于上述两种测试用例，分布式在线集群的性能决定于搜索节点集群中行、列服务器的数量。由于云框架的可扩展性，我们可以根据索引文件的大小来调整搜索节点 N 的大小，通过调整 M 的大小来实现用户的高并发性。再加上 Hadoop 集群，整个云框架充分的实现了云计算的松耦合、弹性计算、海量存储等特点。

6. 总结

由于云计算体系结构的复杂性，要将该技术落实到医疗行业是一项非常复杂的系统工程。本文提出了基于海量医疗数据的症状自查的云框架，充分利用了 Hadoop 的软件框架以及 Lucene 全文检索库，对实现该框架进行了详细的介绍，并针对云框架下的疾病自查服务的实现方法进行了深入的说明。最后，通过实验结果证实了我们提出的云框架的可扩展性。下一步工作我们将重点解决个人隐私安全问题，使云计算技术真正落地到医疗卫生行业。

我们提出的基于海量医疗数据的症状自查服务的云框架目前已成功的实现并在推广应用。在我们的下一步工作中，我们将在连云港、盐城、泰州等地区的区域卫生信息平台项目中抓紧部署症状自查服务，并进一步的部署在移动健康医院云平台中，更好的向居民提供健康服务。

致 谢

感谢潘金贵教授对本文研究工作的指导。本文由计算机软件新技术国家重点实验室自主课题（ZZKT2013B11）及江苏省省级现代服务业（软件产业）发展专项引导资金资助项目（2013771）提供支持。

参 考 文 献

[1] C. He, X. Fan, Y. Li, Toward ubiquitous healthcare services with a novel efficient cloud platform, IEEE Trans. on Biomedical Engineering 60 (1).

[2] D. Hongjuan, H. Jiancheng, W.Wenwu, The sub-health evaluation based on the modern diagnostic technique of traditional chinese medicine, in: First International Workshop on Education Technology and Computer Science (ETCS'09), Vol. 1, 2009, pp. 269–273.

[3] P. Rashidi, D. J. Cook, Keeping the resident in the loop: Adapting the smart home to the user, IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans 39 (5) (2009) 949–959..

[4] D. J. Cook, M. Youngblood, E. O. Heierman III, K. Gopalratnam, S. Rao,

A. Litvin, F. Khawaja, Mavhome: An agent-based smart home, in: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications(PerCom 2003), IEEE, 2003, pp. 521–524.

[5] F. Doctor, H. Hagras, V. Callaghan, A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments, IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans 35 (1) (2005) 55–65.

[6] J. Canny, H. Zhao, Big data analytics with small footprint: Squaring the cloud, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 95–103.

[7] Y. Cheng, C. Qin, F. Rusu, Glade: big data analytics made easy, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012, pp. 697–700.

[8] The Hadoop Distributed File System： Architecture and Design

[9] Above the Clouds：A Berkeley View of Cloud Computing [D].

[10] W. Shang, Z. M. Jiang, H. Hemmati, B. Adams, A. E. Hassan, P. Martin, Assisting developers of big data analytics applications when deploying on hadoop clouds, in: Proceedings of the 2013 International Conference on Software Engineering, IEEE, 2013, pp. 402–411.

[11] Y. Cheng, C. Qin, F. Rusu, Glade: big data analytics made easy, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012, pp. 697–700.

[12] X. Ochoa, E. Duval, Relevance ranking metrics for learning objects, IEEE Trans. on Learning Technologies 1 (1) (2008) 34–48.

[13] E. Hatcher, O. Gospodnetic, M. McCandless, Lucene in action, Manning Publications Greenwich, CT, 2004.

[14] Apace Lucene Documentation. http://lucene.apache.org/core/4_10_0.

[15] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422–426.

[16] 谢鲲, 闵应骅, 张大方, 谢高岗, 文吉刚. 分档布鲁姆过滤器的查询算法[J].计算机学报,2007,04:597-607.

[17] P. Wang, Z. Ding, C. Jiang, M. Zhou, Design and implementation of a web-service-based public-oriented personalized health care platform, IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans 43 (4) (2013) 941–957.

[18] Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich. Recommender Systems An Introduction, Posts & Telecom Press,2013,pp.31-35.

[19] Apache, “Hadoop”, available at: http://hadoop.apache.org. M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption, IEEE Trans. on Parallel and Distributed Systems 24 (1) (2013) 131–143.

周作建，男，1976 年生，博士研究生，研究员级高级工程师，主要研究领域为云计算、数据挖掘，E-mail: lygzzj@163.com

林文敏，女，1986 年生，博士研究生，研究领域为服务计算、云计算等。

王斌斌，男，1989 年生，硕士研究生，研究领域为服务计算、协同计算等。

潘金贵，男，1952 年生，教授、博士生导师，主要研究方向为主要研究领域为知识工程及应用，多媒体软件写作工具和多媒体远程教学系统等。