

# 基于云技术的智慧城市中大规模房屋数据处理<sup>①</sup>

曾羽琚<sup>1</sup>, 陈明辉<sup>2</sup>

<sup>1</sup>(湖南大学长沙环境保护职业技术学院 环境信息系, 长沙 410004)

<sup>2</sup>(湖南商学院 教务处, 长沙 410205)

**摘要:** 在智慧城市中传感器和设备收集各种数据, 实现增值服务. 主要关注来自智慧城市中的智能房屋数据, 提出用于存储和处理大规模房屋数据的 Scallop4SC 平台, 实现智慧城市服务. 房屋数据分为房屋日志数据和房屋配置数据, 分别使用 HBase 和 MySQL 管理. 通过 HBase 键值以 schemaless 方式存储管理异构日志数据. MySQL 管理配置数据, 完成房屋数据查询. 多节点 Hadoop/MapReduce 集群处理巨量房屋日志数据. 最后通过实际一年的房屋日志数据, 计算智能设备能耗并完成实验性评估. 根据实验结果, 探讨 Scallop4SC 处理智慧城市的数据规模.

**关键词:** 智慧城市; 智能住宅; 数据平台; Hadoop; HBase

## Using Cloud Technologies for Large-Scale House Data in Smart City

ZENG Yu-Ju<sup>1</sup>, CHEN Ming-Hui<sup>2</sup>

<sup>1</sup>(Changsha Environmental Protection College, Hunan University, Changsha 410005, China)

<sup>2</sup>(Hunan University of Commerce, Changsha 410205, China)

**Abstract:** In the smart city environment, a wide variety of data are collected from sensors and devices to achieve value-added services. In this paper, we especially focus on data taken from smart houses in the smart city, and propose a platform, called Scallop4SC, that stores and processes the large-scale house data. The house data is classified into log data or configuration data. Since the amount of the log is extremely large, we introduce the Hadoop/MapReduce with a multi-node cluster. On top of this, we use HBase key-value store to manage heterogeneous log data in a schemaless manner. On the other hand, to manage the configuration data, we choose MySQL to process various queries to the house data efficiently. We propose practical data models of the log data and the configuration data on HBase and MySQL, respectively. Then we show how Scallop4SC works as an efficient data platform for smart city services. We conduct an experimental evaluation to calculate device-wise energy consumption, using actual house log recorded for one year in our smart house. Based on the result, we discuss the applicability of Scallop4SC to city-scale data processing.

**Key words:** smart city; smart house; data platform; Hadoop; HBase

智慧城市旨在利用信息和通信技术实现社会的可持续发展. 智慧城市利用传感器和设备收集各种类型的数据包括交通道路条件、房屋电力消耗、居民健康状况的和环境措施. 分析和处理这些数据实现智慧城市中各种服务如交通优化, 社区节能, 当地经济趋势分析, 娱乐, 社区卫生服务, 灾难预防和农业支持<sup>[1]</sup>. 本研究的目标是实现一个数据平台, 可以有效存储和

处理大规模房屋数据. 需要解决的两个问题包括如何管理巨量房屋数据和定义一个独立于特定的服务或应用程序可靠的房屋数据模型.

## 1 总体设计

### 1.1 系统架构

Scallop4SC(智慧城市可伸缩日志数据平台)数据

<sup>①</sup> 基金项目:湖南省职业教育与成人教育学会、中国职业技术教育学会教学工作委员会科研规划立项课题(XHB2013015)

收稿时间:2013-12-18;收到修改稿时间:2014-06-23

平台应可以存储和处理大规模异构数据。Scallop4SC 广泛使用 Hadoop 技术, 具体地说, 它采用 Hadoop/MapReduce<sup>[2]</sup>并行处理的房子日志数据。同时采用 HBase 键值存储管理异构数据。Scallop4SC 还包含一个关系数据库 MySQL 支持房屋配置数据查询。因此, Scallop4SC 采用一种混合数据库配置包括 SQL 数据和 NOSQL 数据库。

Scallop4SC 的总体架构见图 1。房屋日志数据收集器采集每户智能住宅的房屋日志数据并及时发送到 Scallop4SC。大量房屋日志数据存储于 HBase 中, Hadoop 分析房屋日志数据提供进一步服务, 房屋配置数据存储于 MySQL 数据库中。每新增或更新配置数据, 智能住宅都能自动更新配置数据。

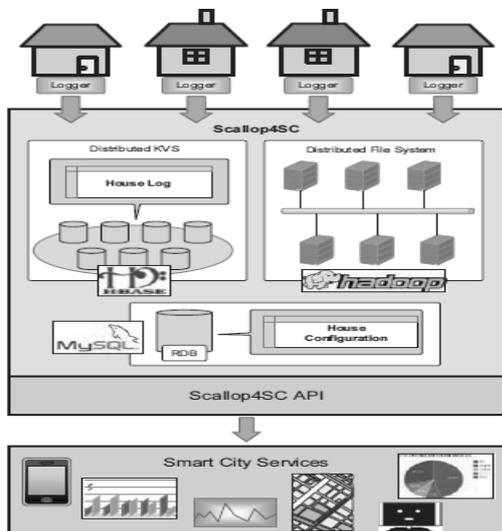


图 1 系统总体设计

通过 Scallop4SC API, 房屋数据存储平台能提供更多的外部服务和应用程序。假设 API 作为一个独立的 Web 服务器平台。针对一个给定的查询, API 从 HBase 和 MySQL 检索出必要的数据库。

## 1.2 数据类型

智能房屋的数据发送到外部云, 为智慧城市提供有益服务<sup>[3]</sup>。本系统将房屋数据分作两种。一种是动态数据, 其数值随时间推移变化。如能源消耗, 设备状态, 房间温度等等。这些数据与人类活动、日期和时间相关联, 并记录日志。称为房屋日志数据。另一种是静态数据, 主要解释房屋配置, 包括房屋地址, 设备的编号, 设备名称和类型, 平面图, 居民信息等。称为房屋配置数据。房屋日志数据和房屋配置数据共同作用实现智慧城市服务。

房屋日志数据又分作以下三种类型。能源日志: 它描述了能源的消费历史。能源包括到电力、水、天然气等。例如, “2012-07-28 12:34:56 Power of TV is 600W.”是一个能源日志。

设备日志: 描述设备的状态和操作使用情况。例如“2012-07-28 12:34:56 TV is off.”这是电视机设备的状态。“2012-07-28 12:35:00 TV is turned on.”是电视设备的操作使用情况。

环境日志: 描述住宅的环境状况, 数据来自传感器。例如“2012-07-28 12:34:56 temperature is 24 degree.”是来自温度传感器的一个环境日志。同样“2012-07-28 12:34:56 number of people is 3.”是来自计数器的。

房屋配置数据分作以下三类型。房屋信息: 描述智慧城市中所有智能房屋的注册信息。设备信息: 描述每户智能房屋中所有设备的信息(例如电器、传感器、设备)。个人信息: 描述每户智能房屋中人的信息。

## 1.3 房屋日志数据的获取与存储

房屋日志数据收集器负责收集智能住宅里日志数据。收集器通过执行适当的 API 获取家居设备数据。例如, CS27-HNS 首先通过 web-API, <http://cs27-hns/TVService/getStatus> 获取电视的状态。然后收集器将所得数据与当前日期、时间、日志类型、房子 ID 和设备 ID 整合。最后, 收集器把房屋日志数据发送到 Scallop4SC 平台。

房屋日志数据收集器根据指定的周期访问指定的设备和传感器, 及时将数据上传 Scallop4SC。根据应用程序的需求和网络容量设定具体的访问周期。如果周期很短, 对实时应用程序是有利的, 但是将产生大量的日志数据。如果周期太长, 数据规模变小了, 但这就有可能会错过一些事件。

房屋日志数据包含着大量异构类型的数据, 所以不可能预先定义严格的数据类型。因此, 采用 schemaless 方式管理数据及简单的键值存储。数据值包括了 5 个部分, HBase<sup>[4]</sup>可以灵活地将他们描述出来, 如表 1。

表 1 描述表

Key	Value
2012-07-28T12:34:56.cs27.Energy.tv01	{value:600,unit:W}
2012-07-28T12:34:56.cs27.Device.tv01	{status:[power:off]}
2012-07-28T12:35:00.cs27.Device.tv01	{operation:on()}
2012-07-28T12:34:56.cs27.Env.temp2	{value:24.4,unit:celsius}
2012-07-28T12:34:56.cs27.Env.pcount3	{value:3,unit:people}

我们定义了作为连接的主键 date、time、house ID、log type、device ID，作为每户智能房屋日志数据独特的字符串。至于键的值，每个日志数据定义自己的散列来描述内容。键值存储的优势是高可伸缩性和容易复制。

### 1.4 房屋配置数据的存储

房屋配置数据不同于房屋日志数据巨量或异构数据。同时，由于需要允许不同的应用程序和服务端来访问数据，需要一个固定数据模型。因此它们方便存储在 MySQL 中关系数据库。

### 1.5 使用 Hadoop 来处理大规模的房屋数据

智慧城市服务是基于各种住房的全面统计，而非仅仅局限于单个住房。例如，需要解决的服务包括：今天这个城市空调使用的能耗是多少？有多少户居民在看中央一台？80%居民什么时候关灯？解决这些问题，不需要非常复杂的算法和依赖关系，只需要对房屋日志数据进行总量统计<sup>[5]</sup>。由于 Hadoop 和 MapReduce 是并行，Hadoop 和 MapReduce 能处理巨量的房屋日志数据。

基本的数据处理策略如下：对于一个既定的查询 q，首先根据 MySQL 中的房屋配置数据，查找相关的设备 Dq 信息；然后，根据存储在 HBase 中的房屋日志数据，假设一个集合 Lq。并且 L ∈ Lq 是 d ∈ Dq 的子集；最后，利用 MapReduce 平台得到 Lq 结果。例如根据 MySQL 中的房屋配置数据，首先查找智慧城市中所有空调设备的 ID 号，然后根据 HBase 中房屋日志数据，检索所有能源日志数据涉及到的设备 ID 号和具体时间。最后，获得电力消费总值。

实施数据处理程序，考虑使用本地 Java MapReduce 和 Pig。

### 1.6 Scallop4SC API

Scallop4SC 提供 API，允许外部应用程序和服务访问房屋数据。考虑以下三种类型。

- 1) SimpleDataAPI: 提供简单的数据库操作包括获取、插入、更新、删除数据。
- 2) StatisticAPI: 提供通用的统计操作包括总和、平均值、最大值、最小值等。
- 3) CustomMapReduceAPI: 接收一个自定义程序，执行程序 MapReduce，并返回结果。设计和实现的 API 目前在进行。

## 2 建立房屋数据模型

解决第二个问题，将对房屋数据进行数据建模。我们提出两个数据模型。第一个是房屋配置数据的数据模型，第二个房屋日志数据的数据模型。

### 2.1 房屋配置数据的数据模型

房屋配置数据存储于 MySQL，因此建立常规数据模型。图 2 显示了房屋配置数据模型，这是一个 ER 关系图。

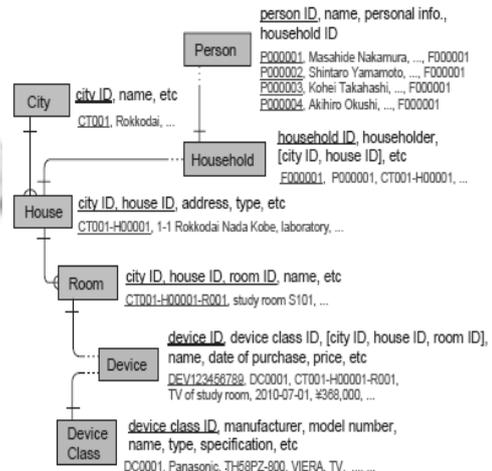


图 2 房屋配置数据模型

在这个 ER 图中，矩形表示一个实体，其数据项在右侧。有下划线的表示主键，[]代表复合外键。实例代表在各个不同实体。实体之间，有父子关系(+—∈描述)，或参考关系(+—...描述)。

数据模型涉及三个主要的房屋配置信息。

房屋信息：房屋信息存储在结构化实体：City, House 和 Room, City 实体表示智慧城市的实例信息。每个城市有许多房屋。House 为一个地址，平面图，类型等。房屋有多个房间。Room 是房屋信息的最小单位，设备安装或部署在房间内。根据 ER 图，House 和 Room, City 和 House 之间存在着子关系，子关系不能独立存在。

设备信息：设备信息由两个实体定义，Device 和 Device Class. Device 表示每个具体设备，而 Device Class 定义了一类设备。Device 指的是房间内部署或安装的设备即一个设备 ID，一个设备类名称以及杂项信息如购买的日期，购买价格等。允许应用程序通过位置，房屋，业主等信息识别该设备等。Device Class 通过普通元数据定义，由多实例组成。它包括制造商、型

号, 类型、规格。

个人信息: 个人信息是由两个实体定义, Person 和 Household. Person 实体定义单独的个人信息, 如姓名、年龄等. 每个人都绑定到家庭. Household 定义了生活在智慧城市中的家庭, Household 中引用到房屋. 如果家庭是搬到另一座房屋中, 关联关系将改变.

房屋配置数据模型在 MySQL 中存储, 可以使用强大的数据搜索 SQL. 例如, 要获取城市 CT001 中所有空调设备的 id, 只指定一个 SQL 语句如下:

```
select 'deviceID'from'DEVICE'WHERE
'deviceClassID'IN( select 'deviceClassID'
from 'DEVICECLASS'
```

```
where 'type'='air-conditioner'
```

## 2.2 房屋日志数据模型

Scallop4SC 通过 HBase 管理房屋日志数据<sup>[6]</sup>. 针对大规模数据, 键值存储具有高伸缩性和容易复制特点. 但是, 它无法实现灵活的查询搜索. 因此建立一种基于 HBase 的简单数据模型, 应用程序能高效访问

房屋日志数据的数据模型. 然而, 这个模型有以下问题. 其一搜索查询是明显受限: 由于约束的 HBase, 数据基本上是搜索前缀行键. 在简单模型, 可以按日期搜索房屋的日志数据. 但是, 我们不能按照设备或日志类型搜索日志数据. 其二数据分布不均匀: 单调递增的行键将数据集中记录在一个单独的服务器. 因此, 房屋的日志数据不均匀地分布在多个 Hadoop 服务器集群, 这会导致性能瓶颈.

应对这些问题, 我们设计的数据模型包括两种 HBase 表: HouseLog 和 HouseIndex.

表 2 为 HouseLog 表. 它由着两种类型房屋日志数据的列族组成: Data 和 Info. Data 列存储日志数据的内容, Info 列存储房子日志数据的元数据. 通过一个值过滤信息, 特定行信息. 对于行键(Row Key), 将携带时间戳哈希编码的字符串均匀分发到存储在服务器集群中的数据行. 行键的前缀搜索是不能直接查找 HouseLog. 于是, 我们提出 HouseIndex 表.

表 2 HouseLog 表

Row key	Column families							
	Data	info						
		Date	Time	Device	house:	Unit	Location	Type
Hash()@<timestamp> 46018eab@2012-07-28T12:34:56	600	2012-07-28	12:34:26	Tv01	Cs27	W	Living room	Energy
53cd5fad@2012-07-28T12:34:56	[power:off]	2012-07-28	12:34:26	Tv01	Cs27	Status	Living room	Device
8afd4abd@2012-07-28T12:35:00	On()	2012-07-28	12:34:26	Tv01	Cs27	Operation	Living room	Device
e85afaad@2012-07-28T12:34:56	24.0	2012-07-28	12:34:26	Temp3	Cs27	Celsius	Living room	Environment
13ffadbc@2012-07-28T12:34:56	3	2012-07-28	12:34:26	Pcount3	Cs27	people	Living room	environment

表 3 为 HouseIndex 表, 有指针指向每个房子日志数据. 行键(Row Key)由 time stamp, log type, houseID 和 device ID 构成. 通过这四个特征字段能查找到任何设备的房屋日志数据. 列族 hlkey 包含 HouseLog 表中行键. 在表 2 中, 共有 24(= 4!) 条记录指向房屋日志数据. 因此, 应用程序在 HouseIndex 表中进行前缀搜索, 获得相应的行键房屋日志记录. 然后, 应用程序通过行

键从 HouseLog 表中获得相应的房屋日志数据

## 3 实施平台

基于上述研究, 实现了 Scallop4SC 平台. 该平台是基于 12 台 Linux 服务器的 Hadoop 集群<sup>[7]</sup>. 平台和中间件的架构是: Vine Linux 6.0, Java SE 7u5, Hadoop 0.20.2+737, HBase 0.90.3 和 Pig 0.9.1. 表 4 显示了该

集群的配置. HDP00 是主节点,集中处理分布式数据. HDP01,HDP02 和 HDP03 以及 ZooKeepers 处理疏散的

节点间关联. 集群是由低性能的两种不同类型的处理器(奔腾 4 和 Athlon 64)组成.

表 3 HouseIndex 表

row key	column family
	hlkey
[timeStamp.type.houseID.deviceID]!	hash()@<timeStamp>
2012-07-28T12:34:56.energy.cs27.tv01	46018eab@2012-07-28T12:34:56
2012-07-28T12:34:56.energy.tv01.cs27	46018eab@2012-07-28T12:34:57
.....	.....
tv01.cs27.2012-07-28T12:34:56.energy	46018eab@2012-07-28T12:34:78
tv01.cs27.energy.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:79

表 4 配置列表

Node Name	Node		Type		spec
	hadoop	hbase	CPU	Mem	
HDP001	Slavre	HRegionMaster	P4 3.00GHZ	400MB	
HDP002	Slavre	Zookeeper	P4 3.01GHZ	1G	
HDP003	Slavre	Zookeeper	P4 3.02GHZ	1G	
HDP004	Slavre	Zookeeper	P4 3.03GHZ	1G	
HDP005	Slavre	HRegionSever	P4 3.04GHZ	1G	
HDP006	Slavre	HRegionSever	P4 3.05GHZ	1G	
HDP007	Slavre	HRegionSever	P4 3.06GHZ	1G	
HDP008	Slavre	HRegionSever	P4 3.07GHZ	1G	
HDP009	Slavre	HRegionSever	Athlon 64 3800+	1G	
HDP010	Slavre	HRegionSever	Athlon 64 3801+	1G	
HDP011	Slavre	HRegionSever	Athlon 64 3802+	1G	

## 4 测试评估

### 4.1 实验概述

通过实验对 Scallop4SC 原型评估. 实验中,利用 CS27-HNS 记录多年房屋日志数据,即每件设备的能源实际值求和并进行批处理. 在 Scallop4SC 中,通过三种方法完成该工作. 一个是本地 Java MapReduceprogram, 一个是 Pig script, 第三个是通过导入部分数据MySQL中, 同样也用SQL完成工作. 最后比较三种方法的处理时间.

### 4.2 目标房屋数据

在 CS27-HNS 里存储着近两年各种房屋日志数据包括环保措施, 设备状态和能耗. 实验中,使用电能消费日志数据. 该数据记录了 32 种电器每六秒消耗的电能, 数量目前达到 460 百万条记录量. 实验中批处理是根据房屋日志数据记录的 32 种电器设备电能消耗计算总能耗. 根据一系列时间变换从一个月、两个月、三个月、六个月、一年、两年、三年和四年, 来

查看的房屋日志数据的大小变化.

### 4.3 实现批处理作业

本机 MapReduce 程序<sup>[8]</sup>: 基于 JAVA 的 MapReduce 程序处理房屋日志数据. 其数据流处理如图 3. 在 mapper 阶段, 分布在多个节点的能源消耗日志记录转变成 deviceID 为主键的键值. 在 Shuffle 阶段, 标记 deviceID 主键的数据聚集在同一节点. 最后, 在 Reduce 阶段, 根据每个 deviceID 计算出电力消耗的总和. 在这个实验中, Java MapReduce 的代码除了空白和注释行一共 57 行.

Pig Script: 使用高层语言 Pig 可以做同样的工作. 使用 Pig, 可以利用 Hadoop 更加容易和交互式实现大规模数据分析. 实验中使用的示例脚本显示如下.

```
records= LOAD'hbase://HouseLogTable'
USING HBaseStorage('info:device data*',!..)
as(id:bytearray,deviceID:chararray,power:double);
grouped_records=GROUP records by deviceID;
```

```
sum=FOREACH grouped_records
GENERATE group, SUM(records.power);
STORE sum INTO 'RESULT.txt';
```

传统的 SQL: 只是一个参考,我们通过 SQL 实现相同的工作. 该智能装置的能耗能通过“sum”和“GROUP BY”的语句计算出来.

```
select'DeviceID',Sum('powerconsumption')
from'consumptionlogTable'group by'DeviceID'
```

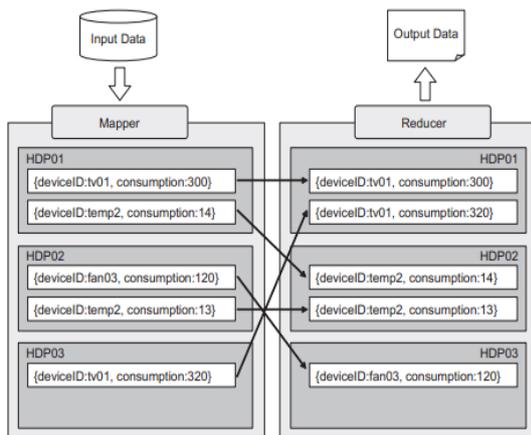


图 3 MapReduce 数据处理

#### 4.4 结果

实验结果显示如图 4. x 轴表示房屋日志数据的数量, y 轴表示执行批处理作业的时间. 图表数据从左到右对应的时间分别为一个月,两个月,三个月、六个月,一年,两年,三年和四年.

为每种方法执行时间与数据量成正比. 最开始的一个月 SQL 具有最佳的性能. 然而,随着数据量的增加,依次为 MapReduce、SQL 和 Pig<sup>[9]</sup>. SQL 处理最大的数据四年日志数据大约 26 分钟. 相比之下,MapReduce 程序可以完成数据处理在大约一半时间(14 分钟). 尽管 Pig 使用同样的 Scallop4SC 平台类似于 MapReduce 操作,但他它无法与其他两种方法并论.

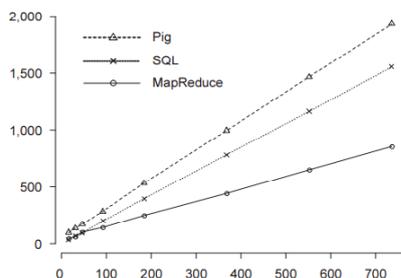


图 4 实验结果

#### 4.5 城市规模的房屋数据可行性

我们以智慧城市规模化房屋数据检测 Scallop4SC, 其结果是可行的. 我们选择两个不同规模的城市: 岳麓区和和一小区. 岳麓区大约由 62000 户家庭组成, 和一小区由 1200 户家庭组成.

假设智慧城市中的房屋从 30 件设备中收集房屋日志数据, 并定期将数据发送到 Scallop4SC. 收集两种情况的数据: 统计监测模式下, 每一分钟发送房屋日志数据; 实时服务模式下, 每六秒钟发送房屋日志数据. 智能设备每天的能耗总量需要花费多少时间来计算. 根据实验结果, 利用智慧城市的房屋日志数据, 能推导出 MapReduce 数据处理时间.

表 5 时间耗费

时间周期	岳麓区	和一小区
每 1 分钟	65min	1min
每 6 秒钟	651min	12min

表 5 显示了每个城市一天的房屋日志数据估计的处理时间. 结果表明, 第一情况下对于 Scallop4SC 原型而言, 是可以在合理时间范围内处理数据. 然而在第二情况,数据大小会显著增大即使是小区 1200 户需要花费 10 个小时. 因此,对于这样一个实时日志记录, 我们需要扩展 Scallop4SC,通过添加更多计算节点. 根据智慧城市的服务需要不同数据传输周期, 确定正确的服务器数量等等是下一步研究方向.

#### 5 结论

在本文中,提出了一个数据平台 Scallop4SC, 在智慧城市用来存储和处理大规模的房屋数据. Scallop4SC 广泛使用 Hadoop、MapReduce 和 HBase 工具管理大规模的房屋日志数据. 为房屋配置数据和房屋日志数据提出了具体的数据模型. 还建立了一个原型 Scallop4SC. 通过实验对 Scallop4SC 评价. 下一步, 正在开发 Scallop4SC api,使得外部应用程序访问平台更容易

#### 参考文献

- Hollands RG. Will the real smart city please stand up? City: Analysis of Urban Trends, Culture, Theory, Policy, Action, 2008, 12: 303-320.
- Apache Software Foundation. Apache hbase reference guide 2012. <http://hbase.apache.org/book/book.html>.

- 3 IBM. E health and collaboration---collaborative care and wellness. <http://www.ibm.com/smarterplanet/nz/en/healthcaresolutions/nextsteps/solution/X056151Y25842W14.htm>.
- 4 IHbase. <http://github.com/ykulbak/inbase>. 2012.
- 5 王珊,王会举,覃雄派,周烜.架构大数据:挑战、现状与展望.计算机学报,2011,(10):1741-1752.
- 6 李成华,张新访,金海,向文.MapReduce:新型的分布式并行计算编程模型.计算机工程与科学,2011,(3):129-135.
- 7 易小华,刘杰,叶丹.面向 MapReduce 的数据处理流程开发方法.计算机科学与探索,2011,(2):161-169
- 8 Borthakur D. The hadoop distributed file system: Architecture and design. <http://hadoop.apache.org/common/docs/r0.18.0/hdfsdesign.pdf>. 2007.
- 9 Upadhyaya P, Kwon YC, Balazinska M. A latency and fault-tolerance optimizer for online parallel query plans. Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11). 2011.

WWW.C-S-A.ORG.CN

WWW.C-S-A.ORG.CN