接收与处理分离的实时大数据处理模型

彭建华 [1]，李臣明 [1]，邱军林 [1]，李晓芳 [2],徐立中 [1+]

1. 河海大学计算机与信息学院 南京 210098

2. 常州工学院 常州 213002

Real-Time Big Data Processing Model Base on Receiving and Processing Separation

Peng Jianhua[1],Li Chenming[1], Qiu Junlin[1], Li Xiaofang[2] , Xu Lizhong[1+]

1. College of computer and information, Hohai University, NanJing 210098
2. Changzhou institute of technology, ChangZhou 213002
+ Corresponding author: E-mail: pengjianhuacc@hhu.edu.cn

Peng Jianhua,Li Chenming, Qiu Junlin, et al. Real-Time Big Data Processing Model Base on Receiving and Processing Separation

Abstract: The system must have high data processing efficiency when it processed big data. We designed a receiving and processing separation model of data processing in the article. The model includes the data receiving unit, the memory database, the original data distribution unit, the processing unit and the data merging unit. The receiving unit receives and integrates structure and unstructured data to a whole structure data and put it into the memory database. The original data distribution unit checks data from the memory database and dispatches the data to the processing unit using the big data load balancing method. The processing unit processes data and put the processing result data into the memory database again. The data merging unit gets data from the memory database and merges the data. The experimental results show that the system kept with the high efficiency using the model.
Key words: big data; Load Balancing; Real-time analysis and processing

摘 要：在大数据处理系统中，系统对数据处理效率、安全、稳定性有非常高的要求。为了满足对大数据实时、高效、稳定处理的需求，文章提出了一种接收与处理分离的数据处理模型，数据处理模型由数

* The National Natural Science Foundation of China under Grant No. 51179047 (国家自然科学基金)

据接收单元、内存数据库、原始数据分发单元、数据处理单元、处理数据分发单元、数据归并单元组成，接收单元负责接收、整合结构化数据与非结构化数据，把每条完整的数据放入内存数据库中，分发单元从内存数据库中检测获取数据，按照海量数据负载均衡算法把数据分发到数据处理单元，数据处理单元处理数据，处理结果放入内存数据库，处理数据分发单元继续从内存数据库中获取处理后的数据并按照海量数据负载均衡算法把数据分发给数据归并单元，实验证明，使用本模型方法，系统保持了非常高效的处理效率。

关键词：大数据；负载均衡；实时分析处理

中图法分类号 TP311

1 引言

现在，全世界每两天产生的数据量达到 5 穰，在网络宽带不断增加的情况下，数据还在快速增长，2012 年全球信息总量已经达到 2.7ZB，而到 2015 年这一数值预计会达到 8ZB[1]，同时，网路大数据的规模与复杂度的增长远远超出了符合摩尔定律增长的机器处理和计算能力[2]，通过预测，到 2020 年，全世界产生的数据规模将达到今天的 44 倍。数据必须被分析，通过分析发现有价值的数据后，这些被产生的浩瀚数据才具有价值，要实时分析这些海量数据，一是要有高性能的计算机，二是要有高效的算法模型，本文提出的接收与处理分离的实时大数据处理方法，能够让高性能的计算机以较低的代价高效、并行处理数据，实时分析处理数据的效率能够有效提高。

2 相关工作

通过多线程系统架构设计，系统能够在多核计算机上高速、并行处理数据，多核高性能计算机的效率能够得到尽可能的发挥[8][9][10]；在高速数据处理过程中，有效的负载均衡方法是保证系统性高效率处理数据的必要条件，文献[15]-[22]提供了各种有效的负载均衡方法；MapReduce 提供了一种高效、分布式处理大数据的系统模型[11]–[14]；文献[6]结合多线程技术，提出了一种海量数据并行优化模型，该模型通过把数据处理工作流分为 3 个上下文，通过多个队列对数据流进行并行处理；文献[4]通过优化 MapReduce 模型，提出了一种针对高速数据流的大规模数据实时处理方法 RTMR，ＲＴＭＲ的处理过程为预处理历史数据并将中间结果分布缓存到各个节点上，在节点上基于 SEDA 构造从 MAP 阶段到 Reduce 阶段的本地阶段化流水线，充分利用本地计算和存储资源实现数据流同历史数据的实时计算。

但文献[6]主要侧重于描述大数据流分段处理，在数据流超过系统负荷后，系统对数据流的处理方式没有描述；文献[7]对 MapReduce 模型进行了改进，对多核高性能计算机的数据处理能力并无改进；文献[22]提出的一致性 hash，通过构造环形 hash 空间，构造虚拟节点的方式，把数据通过哈希算法分配到大量的虚拟节点，并把虚拟节点通过同一哈希算法映射到物理节点，在系统节点动态改变情况下，这种方式最大限度减少了系统性能降低的问题，但不能解决非结构化数据分析耗时，数据结构不一致导致的负载不均衡问题。

3. 实时高效处理大数据的应用模型

本应用模型提出了接收与处理分离的一种数据处理逻辑结构，在高性能计算机处理大数据的应用

中，解决了如下问题：

- 使用内存数据库作为接收数据与处理数据的中间缓存，在处理大数据过程中，没有进行频繁的 I/O 读写操作，数据处理性能得到充分发挥；
- 使用负载均衡的海量数据处理方法，监测每个节点空闲率，动态调控数据派遣线程的数据派遣速率，当一段时间的数据风暴到来时，系统平稳处理数据，数据不丢失。
- 使用负载均衡的海量数据处理方法，通过对每个数据处理线程的相对空闲率与数据处理效率进行监测、均衡处理，提高了数据处理线程的数据命中均衡性。

3.1 接收与处理分离数据处理应用模型

为了高效、稳定处理数据，对大数据处理进行实体与逻辑上的划分，进行分层设计。数据处理完整过程包含：数据发送节点与数据处理节点建立数据传输通道、数据处理节点接收数据、分析数据、归并数据、数据处理结果输出。整个过程中，数据接收与数据输出效率最高，数据分析、数据归并效率受到数据结构类型影响最大 ,效率相对而言较低。在大数据分析中，被分析数据大部分是非结构化数据，因此比分析、归并结构化数据效率要低。基于数据处理流程、数据分析与数据归并在应用系统中的特点，在应用系统的数据接收与数据分析线程之间加上内存数据库，数据接收线程接收、整合结构化数据与非结构化数据，把数据直接放入内存数据库中 ,数据派遣线程 1 使用一致性 hash 算法[4] ,把内存数据库中的数据分配到各个数据处理线程的内存空间，数据处理线程从自己的内存空间获取数据进行数据分析处理，分析处理后的结果数据再放入内存数据库，数据派遣线程 2 通过同样的一致性 hash 算法，把分析处理后的数据分配到数据归并线程的内存区域，数据归并线程读取自己内存区域的数据进行归并，然后把归并结果进行输出。数据派遣线程通过一致性 hash 算法进行散射后 ,由于数据结构不一致以及一致性 hash 算法本身散射性的影响，数据分析处理线程之间负载并不十分均衡，同样道理，数据归并线程之间负载也不十分均衡，为了达到负责均衡，整个系统使用海量数据动态负载均衡算法进行均衡处理。应用模型见图 1:

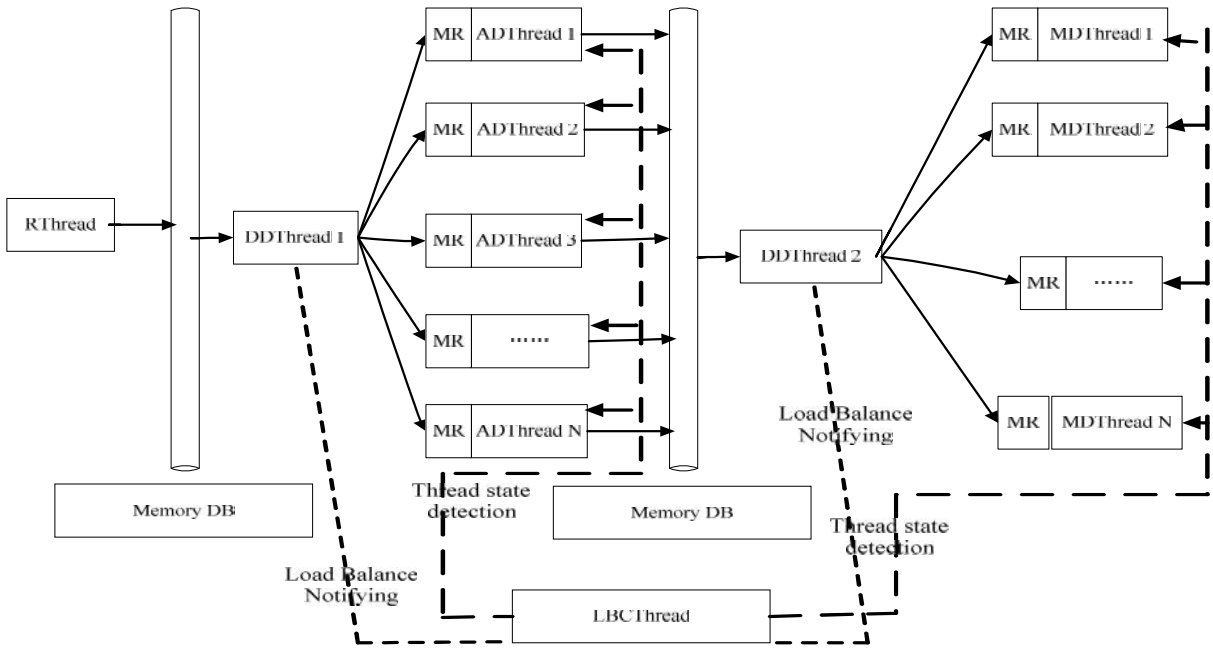

Fig.1 Application model

图 1 应用模型

应用模型流程：

- 接收线程 RThread 接收数据，放入内存数据库（Memory DB）;
- 数据派遣线程 DDThread 1 循环从内存数据库中读取来至于 RThread 线程的数据，通过一致性 hash 算法把数据派遣到数据分析处理线程 ADThread 的 MR 中，数据分析处理线程依据 FIFO 原则处理自己 MR 中的数据，把处理后的数据放入内存数据库（Memory DB）;
- 数据派遣线程 DDThread 2 循环从内存数据库中读取来至于 ADTread N 线程的数据，通过第二步的一致性 hash 算法把数据派遣到数据归并线程 MDThread 的 MR 中，数据归并线程 MDThread 依据 FIFO 原则处理自己 MR 中的数据，对处理后的数据进行输出；
- 动态负载均衡线程负责检测数据处理线程与数据归并线程内存区状态，计算数据处理线程与数据归并线程的数据处理效率与相对空闲率，当数据处理线程与数据归并线程空闲率值与处理效率值达到一定差值时，通知数据派遣线程进行数据派遣调整，进行动态负载均衡数据迁移。
- 数据派遣线程 DDThread 检测数据分析处理线程与数据归并线程空闲率，通过空闲率控制数据派遣速度。

3.2 接收与处理分离数据处理数学模型

数据处理完整过程包含：数据发送节点与数据处理节点建立数据传输通道、数据处理节点接收数据、分析数据、归并数据、数据处理结果输出 5 个环节，这 5 个环节对应于数据处理的 4 个流程：数据接收、数据分析、归并数据以及输出结果数据。处理海量数据的模型是一个排队模型，令 N（t）表示时间(0,t)内到达的数据量，则：

- N(0)=0，即在 0s 内没有数据到达；
- $\{N(t), t \geq 0\}$ 具有无记忆性，在不相交的时间区间内到达的数据相互独立，即对任取 n 个时刻 $0 < t_1 < t_2 ... < t_n$，随机变量 $N(t_1) - N(0), N(t_2) - N(t_1), ...N(t_n) - N(t_{n-1})$ 是相互独立的。
- $\{N(t), t \geq 0\}$ 具有平稳增量，在 $(t, t + \Delta t)$ 内到达的数据量只与时间间隔 Δt 有关，而与时间 t 无关。

所以数据处理节点接收的数据流服从泊松分布，因数据处理的时间也具有无记忆性，所以满足负指数分布，同时，数据在数据处理应用模型中，被 DDThread 1 线程按照 FIFO 方式进行派遣，因此本文描述的实时高效处理大数据的应用模型是 $M/M/1/\infty$ 排队系统模型。

3.3 负载均衡的海量数据处理模型

定义 1（数据处理效率）: 单位时间内处理数据包数的移动平均数。效率模型数学表达式：

$$\begin{cases} V_n = \dfrac{(n-1)V_{n-1} + D_n}{n} & n>0 \\ V_0 = 0 & n=0 \end{cases} \quad (1)$$

时间 t_n 中处理数据包数：

$$D_n = \frac{1}{t_n} \quad (2)$$

其中：

$$t_n = T_E - T_B \quad (3)$$

其中：

T_B：数据包分析处理前系统时间；

T_E：数据包分析处理后系统时间；

t_n 表示处理第 n 个数据包的时间。

定义 2（相对空闲率）：数据区中待处理数据包的数量与数据处理效率的乘积成反比，通过如下公式表示

$$M=\frac{V}{C} \quad (4)$$

式中，M 为相对空闲率；C 为该数据处理线程的待处理数据包个数；V 为该数据处理线程的数据处理效率。

通过实时计算、监控每个节点的数据处理效率，当整个系统中节点的数据处理效率相差达到一定程度时，就进行数据动态迁移，对一些需要调整的虚拟节点重新进行哈希分配，把数据向数据处理效率高的线程移动；实时监控每个节点的相对空闲率，当整个系统中节点的空闲率相差达到一定程度时，就进行数据动态迁移，对一些需要调整的虚拟节点重新进行哈希分配，把数据向空闲率高的线程移动，整个系统各个节点数据处理达到平衡。对虚拟节点进行迁移，整个系统达到平衡时，保证了尽量少的迁移数据。

4 实验与结果分析

4.1 实验环境与数据

为了测试本文描述的模型与算法，编写了一个采集与分析测试系统，该系统由数据发送端与接收端两部分构成，数据发送端发送大话务数据，采集与分析对接收到的数据进行分析处理。

数据采集与分析处理是多线程服务程序，根据本文模型与算法，该服务程序启动时创建了一个数据接收线程，创建 M 个数据处理线程（M 通过配置文件进行配置，缺省值是 4），开辟 N 个数据存储区（N 通过配置文件进行配置，缺省值是 200），空闲率计算间隔 0.2s。

数据归并线程的处理逻辑与数据分析线程完全一致，因此本实验数据不对数据归并线程的逻辑进行验证与描述。

数据使用从扬州电信采集的 HTTP、WEIBO、QQ 网路数据（非结构化数据），数据接收线程通过用户信息的 IMEI 进行 hash 散列到 N 个不同的数据存储区中。

发送端机器配置见表 1 所示。

Table 1 The sender machine configuration

表 1 发送端机器配置

操作系统	Windows XP 专业版 32 位 SP3
cpu	英特尔 酷睿 2 双核 E7400
memory	2GB
network card	Marvell YuKon 88E8056 PCI-E Gigabit Ethernet Controller

接收与处理端机器配置见表 2 所示

Table 1 Receiving and processing terminal machine configuration

表 2 接收与处理端机器配置

操作系统	Windows XP 专业版 32 位 SP3
cpu	AMD Athlon II X4 640 四核
memory	3GB
network card	RTL8168D PCI-E Gigabit Ethernet NIC

通过测试，以上接收与处理机器在处理发送的非结构化数据时，最大稳定处理效率是 6.5w/s.

4.2 数据派遣均衡性

数据派遣均衡性实验，主要验证线程处理效率一致情况下，算法在数据分发上的均衡性。发送 QQ 类型网络数据，每个数据分析线程分析数据的时间将保持一致。数据发送端每秒发送 6.5W 条 QQ 网络类型数据。

数据接收线程 RT 接收数据 D 并放数据 D 进入

内存区 MR 中，接收数据速率为 RV，则 MR 中进入数据总量 TDM：

TDM=RV*T　　T 为接收数据的时间　　(5)

数据派遣线程 DT 根据一致性 hash 算法，对 MR 中的数据进行分发派遣，并放派遣的数据到每个数据处理线程 DDT 对应的内存区 MRD 中，每个 MRD 中获取的数据速率 DRVi，其中 i 的取值为 1 到 4，每个 MRD 接收的数据总量：

TDDMi=DRVi*T　T 为接收数据的时间 (6)

DDT 对应的数据命中率:

DRRi= TDDMi/ TDM　　(7)

负载均衡线程 BAT 计算每个 DDT 的数据处理效率 Vi，得到每个 DDT 的相对空闲率 Mi，比较 Mi 的大小，在本实验中，Vi 的大小顺序：

V1>V3>V2>V4　　(8)

根据 Vi 差值大小，调整一致性 hash 算法的虚拟节点，把派遣到 DDT4 的一部分节点派遣到 DDT1 中，同时调整派遣到 DDT2 中的一部分节点到 DDT3 中，循环计算 DDT 的空闲率，对虚拟节点进行动态迁移调整，直到 Vi 空闲率差值小于 £。

图 2 所示为数据派遣均衡性实验结果，可以看出，数据在 1 秒内被完全散射后，4 个线程分别被命中不同的数据，随后数据按照现有散射的规律命中各个线程，在 4 秒到 6 秒这段时间期间，系统对不均衡的数据进行迁移，4 秒后线程数据命中率被重新定位并达到稳定，线程数据命中率接近，通过实验结果可以得知，该数据处理模型有非常好的数据派遣均衡性。

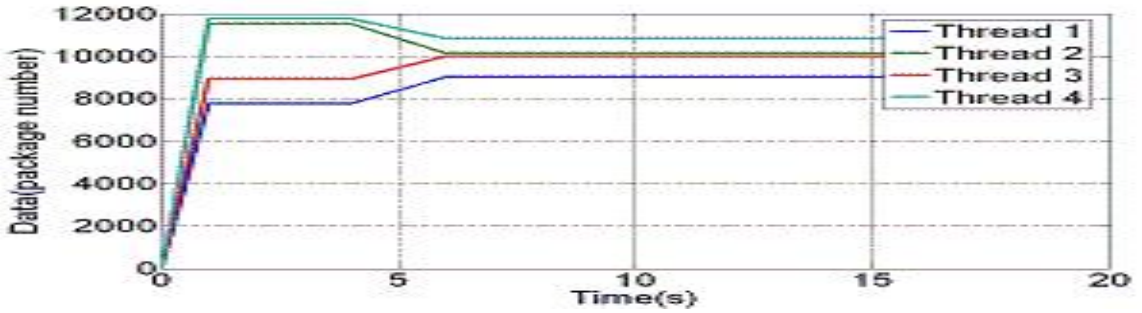

Fig.2 Data dispatch balance

图 2 数据派遣均衡性

4.3 数据风暴过程中系统的稳定性

在准备的实验数据中，本实验环境系统数据处理最大值为 6.5W/s，模拟发送数据情况如下表 3：

Table 3 Analog transmission data

表 3 模拟发送数据

时间段	每秒网络数据量
1s—6s	5W
6s—10s	9W
10s 以后	5W

数据接收线程 RT 接收数据 D 并放数据 D 进入内存区 MR 中，接收数据速率为 RV，则 MR 中进入数据总量 TDM：

TDM=RV*T　　T 为接收数据的时间　　(9)

数据派遣线程 DT 根据一致性 hash 算法，对 MR 中的数据进行分发派遣，并放派遣的数据到每个数据处理线程 DDT 对应的内存区 MRD 中，每个 MRD 中获取的数据速率 DRVi，其中 i 的取值为 1 到 4，每个 MRD 接收的数据总量：

TDDMi=DRVi*T　T 为接收数据的时间　　(10)

DDT 的数据处理效率为 Vi，DDT 处理数据量：

TDDDi=Vi*T　　T 为处理数据的时间　　(11)

在数据风暴情况下，DDT 被派遣的数据速率 DRVi 大于其数据处理效率 Vi，这样内存数据区中数据增加量：

MIi= TDDMi- TDDDi (12)

通过内存数据库，对数据进行缓存，数据风暴结束后，TDDMi 小于 TDDDi，MRi 中被缓存的数据逐渐减少，实验结果见图 3 所示.

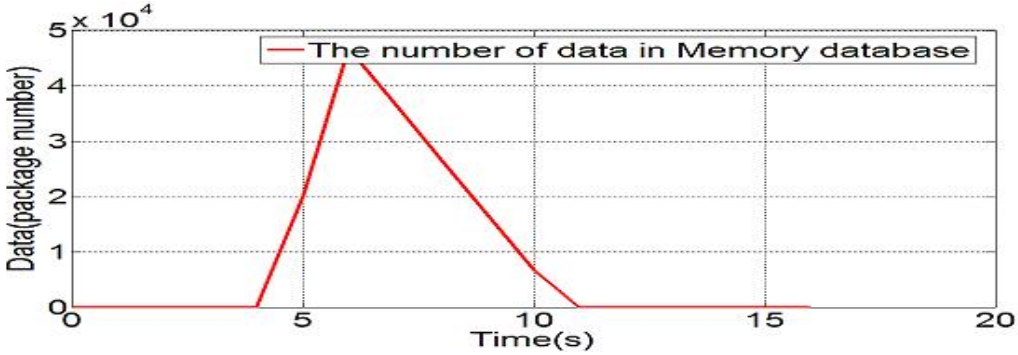

Fig.3 The data changing trend in the memory

图 3 内存数据库中数据变化趋势图

通过图 3 可以看出，系统对瞬间与持续数据风暴具有很好的处理能力以及稳定性能。

5 结论

本文通过对大数据处理进行实体与逻辑上的划分，根据划分提出了一种实时高效处理大数据的应用模型，通过与现有数据处理模型以及负载均衡算法的实验比较可以得出，该模型具有好的数据处理效率，同时在如下方面具有优势：

- 使用内存数据库作为接收数据与处理数据的中间缓存，在处理大数据过程中，没有进行频繁的 I/O 读写操作，数据处理性能得到充分发挥；
- 使用负载均衡的海量数据处理方法，监测每个节点空闲率，动态控制数据派遣线程数据派遣速率，一段时间数据风暴到来时，数据不丢失，系统稳定处理数据。
- 使用负载均衡的海量数据处理方法，通过对每个数据处理线程的相对空闲率与数据处理效率的监测并进行均衡处理，数据处理线程的数据命中均衡性平衡。

本文提出的方法是为解决真实商业系统性能问题，由于本身技术能力、实验环境因素，不能对其它类似技术进行实验验证(比如一致性 hash 算法)，因此没有给出本文提出的方法与其他类似方法的实验对比，下一步的研究重点是对与本文类似技术进行实验验证，通过对实验结果比较分析，进一步改进本文提出的方法。

[1]. Feng Deng-Guo, Zhang Min, Li Hao. Big data security and privacy protection. Chinese Journal of Computers, 2014, 37(1): 246-258 (in Chinese)
(冯登国, 张敏, 李昊. 大数据安全与隐私保. 计算机学报, 2014, 37(1): 246-258)

[2]. Wang Yuan-Zhuo, Jin Xiao-Long, Cheng Xue. Network Big Data: Status and Prospects. Chinese Journal of Computers, 2013, 36(6):1125-1138 (in Chinese)
(王元卓,靳小龙,程学. 网络大数据：现状与展望. 计算机学报, 2013, 36(6):1125-1138)

[3]. Y. Wang, Z. Zhou, L. Liu and W.Wu. Replica-aided load balancing in overlay networks. In Proceedings of J. Network and Computer Applications. 388-401,2013.

[4]. Liu, Z., Lin, M., Wierman, A.. Low, S.. Andrew, L.L.H. IEEE/ACM Transactions on Networking, 2014.

[5]. Hongseok Kim, Gustavo De Veciana, Xiangying Yang, Muthaiah Venkatachalam. IEEE/ACM Transactions on Networking, Volume: 20, Issue: 1, Pages: 177-190, 2012

[6]. Sun Xiao-Juan,Sun Ning-Hui,Lei Bin. One kind of massive data streaming applications Parallel Optimization Model
(孙小涓,孙凝晖,雷斌;一种海量数据流应用并行优化模型; Journal of Software, Vol.20, Supplement, December 2009, pp.23 – 33.)

[7]. Qi Kai-Yuan,Zhao Zhuo-Feng,Ma Qiang. The real-time data processing methods for large-scale high-speed data stream. Chinese Journal of Computer, Vol. 35No., March 20.
(亓开元, 赵卓峰, 房俊, 马强; 针对高速数据流的大规模数据实时处理方法; Chinese Journal of Computer, Vol. 35No., March 20)

[8]. C. E. Oancea, J. W. A. Selby, M. Giesbrecht and S. M. Watt. Distributed Models of Thread-Level Speculation. In Proceedings of the 2005 International Conference on Parallel and Distributed Processing Techniques and Application, pages 920-927, 2005.

[9]. Jack LL, Susan JE, Joel SE, Henry ML, Rebecca LS, Dean MT. Converting thread-level parallelism into instruction-level parallelism via simultaneous multithreading. ACM Trans. on Computer Systems, 1997,15(2).

[10]. Susan JE, Joel SE, Henry ML, Jack LL, Rebecca LS, Dean MT. Simultaneous multithreading: A foundation for next generation processors. IEEE Micro, 1997,17(5).

[11]. Dean J, Ghemawat S, MapReduce: Simplified data processing on large cluster. ACM Communication, 2008, 51(1):107-113.

[12]. Ranger C, Raghuraman R, Penmetsa A, Bradski G, Kozyrakis C. Evaluating MapReduce for multi-cor and multiprocessor systems//Proceedings of the 13th International Conference on High-Performance Computer Architechure(HPCA 2007). Phoenix, USA, 2007:13-24.

[13]. Kasshoek F, Morris R, MaoY. Optimizing MapReduce for multicore architectures. MIT Computer Science and Artificial Intelligence Laboratory: Technical Report MIT-CSAIL-TR-2010-020, 2010.

[14]. Ekanayake J, Li H, Zhang B, Gunarathne T, Bae S, Qiu J, Fox G. Twister: A runtime for iterative MapReduce//Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing(HPDC 2010). Chicago, IL, USA, 2010:810-818.

[15]. KARGER, D., LEHMAN, E., LEIGHTON,F.,LEVINE, M.,LEWIN,D.,ANDPANIGRAHY,R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In Proc. 29th Annual ACM Symposium on Theory of Computing (El Paso, TX, May 1997), pp. 654–663.

[16]. Robert Devine. Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm. In Proceedings of 4th International Conference on Foundations of Data Organizations and Algorithms, 1993.

[17]. B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load balancing in dynamic structured P2P systems,” in Proc. IEEE INFOCOM , Mar. 2004, pp. 2253 –2262.

[18]. F. Kaashoek and D. Karger, “Koorde: A simple degree-optimal hash table,” in Proc. IPTPS, Feb. 2003, pp. 98 –107.

[19]. C.XuandF. Lau. Iterative dynamic load balancing in multicomputers. Journal of the Operational Research Society, 45(7), July 1994.

[20]. Iqbal, Saeed and Carey, Graham F. 2005. Performance analysis of dynamic load balancing algorithms with variable number of processors. Journal of Parallel and Distributed Computing, Elsvier Inc., Vol. 65, Issue 8, 934-948.

[21]. Prasanna Ganesan and Mayank Bawa. Distributed balanced tables: Not making a hash of it all.Technical Report 2003-71, Stanford University,Database Group, 2003.

[22]. Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma. Performance Analysis of Load Balancing Algorithms. World Academy of Science, Engineering and Technology 38 2008.

Peng Jianhua was born in 1973. He is Ph.D. candidate at Hohai University, NanJing, China. His current research interests include cloud computing, big data analytics and image analysis and processing. He is currently a software engineer in ZTE.

彭建华(1973—)， 男， 河海大学计算机与信息学院博士研究生，中兴通讯南京研究所软件工程师， 主要研究领域为云计算、大数据处理、图像分析与处理