计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2018.20170053
55(7):1569-1583,2018

面向处理器微体系结构评估的高通量 MicroBenchmark 研究

薛 瑞[1,2] 苗福涛[3] 叶笑春[1] 孙凝晖[1] 徐文星[4]

[1](计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

[2](中国科学院大学 北京 100049)

[3](中国农业银行 北京 100073)

[4](北京石油化工学院 北京 102617)

(xuerui@ict.ac.cn)

High Throughput MicroBenchmark Research for Processor MicroArchitecture Evaluation

Xue Rui[1,2], Miao Futao[3], Ye Xiaochun[1], Sun Ninghui[1], and Xu Wenxing[4]

[1](*State Key Laboratory of Computer Architecture* (*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190)

[2](*University of Chinese Academy of Sciences*, *Beijing* 100049)

[3](*Agricultural Bank of China*, *Beijing* 100073)

[4](*Beijing Institute of Petrochemical Technology*, *Beijing* 102617)

Abstract Benchmarks are important means to evaluate processor microarchitecture. The high-throughput application is a kind of application that focuses on throughput efficiency and contents a large number of loosely coupled small-scale jobs. The typical characteristics of high-throughput application are high throughput, hard real-time and high concurrency. The key target of processor microarchitecture design for high-throughput application is how to improve the throughput efficiency of operations. The design of high-throughput processor microarchitecture needs micro benchmark from high-throughput application as evaluation basis for designing high efficient processing architecture. While for now, existing benchmarks can not effectively and comprehensively evaluate the processor microarchitecture design for high-throughput application. In this paper, we propose a suit of new benchmarks—HTC-MicroBench—for the evaluation of designing the processor microarchitecture for high-throughput application. Firstly, we present a classification method for high-throughput applications based on the features of workloads. Secondly, according to the characteristics of high-throughput application, we present a parallelization model based on Pthread model to design and implement HTC-MicroBench. Furthermore, we evaluate HTC-MicroBench from many aspects, such as concurrency, data coupling and cache efficiency. Finally, we use HTC-MicroBench to evaluate the speedup of TILE-Gx and Xeon. The evaluation results show that HTC-MicroBench can effectively evaluate the processor microarchitecture design for high-throughput application.

收稿日期:2017-02-06;**修回日期**:2017-06-23

基金项目:国家重点研发计划项目(2016YFB0200501);国家自然科学基金项目(61332009);国家自然科学基金委员会"创新研究群体科学基金"(61521092);数学工程与先进计算国家重点实验室开放基金(2016A04)

This work was supported by the National Key Research and Development Program (2016YFB0200501), the National Natural Science Foundation of China (61332009), the National Natural Science Foundation of China "Innovation Research Group Science Foundation" (61521092), and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing (2016A04).

Key words high-throughput applications; processor microarchitecture design; benchmarks; parallelization; Pthread model

摘 要 基准测试程序是评估处理器微体系结构设计的重要手段,然而当前的基准测试程序无法有效全面地评估面向高通量应用的处理器微体系结构的设计.基于此,针对高通量应用的特征,提出了用于评估面向高通量应用的处理器微体系结构设计的基准测试程序——HTC-MicroBench.首先,提出一种基于应用特征的高通量应用分类方法,并基于此分类方法对高通量应用中的 Workload 进行分类.其次,针对高通量应用的特征,提出了一种基于线程的作业处理节点并行化模型,基于此模型完成了 HTC-MicroBench 的设计和实现.最后,从作业并发性、作业之间的耦合性和 Cache 使用效率等指标对 HTC-MicroBench 进行实验评估;并基于 HTC-MicroBench 对 TILE-Gx 和 Xeon 两种处理器的并行加速能力做了评估,高并发、低耦合和由 Workload 特征所体现出的不同 Cache 命中率的评估结果说明了 HTC-MicroBench 能够准确刻画高通量应用的特征,并对面向高通量应用的处理器微体系结构的设计进行有效的测评.

关键词 高通量应用;处理器微体系结构设计;基准测试程序;并行化;Pthread 模型

中图法分类号 TP311

随着互联网、云计算、社交网络等技术的飞速发展,全球产生的信息量急剧增长.全球数据库每天以 2.5 MB 的数据增长,其中 90% 的数据是在过去 2 年创造的,这些数据无处不在[1].例如能源、制造业、交通运输业、服务业、科教文化、医疗卫生等领域都积累了 TB 级、PB 级乃至 EB 级的大数据.著名的全球连锁超市沃尔玛每小时需要处理 100 余万条的用户请求,维护着一个超过 2.5 PB 的数据库;在高能物理实验中,2008 年开始投入使用的大型强子对撞机每年产生超过 25 PB 的数据;社交网络 Facebook 现已存储超过 500 亿张照片.

海量数据的快速处理的需求,对面向高通量应用的处理器微体系结构设计提出了更高的要求[2-3].然而,现阶段面向高通量应用的 Benchmark,如 DCBench, LinkBench, CloudSuite, BigDataBench 等,大都是在 Hadoop 等计算机集群平台针对目标系统进行测试,主要衡量的是网络、IO 等系统能力,难以对处理器微体系结构的设计进行有效的评估.因此,针对高通量处理器微体系结构评估的需求,也急需一套新的、面向高通量应用的处理器微体系结构评估的基准测试程序.

针对上述问题,本文首先从高通量的典型应用出发,分析高通量应用实例的整体特征;然后选取高通量应用中的典型 Workload 作为代表,分析面向高通量应用的处理器的微体系结构需求,设计并实现了一套面向高通量应用的处理器微体系结构评估的基准测试集:HTC-MicroBench;最后通过 HTC-MicroBench 对现有的代表性多核和众核处理器微体系结构进行性能测试和分析.

1 相关工作

Benchmark 的主要作用是对计算机系统或计算机的各组件进行测试评价,从而更好地指导计算机系统的设计或者指导消费者选择合适的计算产品.而对处理器微体系结构的研究作为计算机系统结构研究中的重点方向,也离不开对相应 Benchmark 研究的支持.

传统的高性能计算领域的 Benchmark 有 SPEC 基准测试体系[4],SPEC CPU 2006 是业界常用的一套程序集,包括整型计算和浮点计算,覆盖到了计算机编程、算法、人工智能、基因、流体力学、分子动力学和量子计算等方面,保证了测试的完整性.PARSEC 是多线程应用程序组成的测试程序集[5],具有多线程、新型负载、非针对高性能和研究性等几个典型特征,主要应用于计算机视觉、视频编解码、金融分析、动画物理学和图像处理等.HPCC 是测试高性能计算能力的基准测试集[6],包含 HPL, DGEMM, STREAM, PTRANS, FFTE, RandomAccess 和带宽延迟[7]测试 7 个主要的测试程序,可以对浮点计算能力、持续内存带宽大小、处理器协作能力、随机内存访问效率和内部高速互联网络的性能等方面进行测试.以上 Benchmark 主要针对高性能应用中计算量大的特点进行基准测试程序的设计,而没有考虑高通量应用的主要特征.

对于大数据相关应用领域的 Benchmark 也已经有大量的研究成果[8]. HiBench 是 Intel 开放的 Hadoop Benchmark 集[9],包含 9 个典型的 Hadoop 负载,用于测评运行 Hadoop 集群的性能;雅虎开发的 YCSB 用于对比 NoSQL 数据库的性能[10],其目的是评估键值和云数据库;DCBench 是针对数据中心负载的 Benchmark 集[11-12];第 1 个发行版本含有 19 个代表性的负载,根据应用特征的不同,可以分为 on-line 和 off-line 这 2 类,根据编程模型的不同,可以分为 MPI,MapReduce 等[13];LinkBench 是 Facebook 开发的一套用于对社交网络数据库进行性能测评的工具集[14];专门用于测试存储社交图谱和网络服务的数据库;CloudSuite[15] 是针对云计算开发的一套测试程序集,CloudSuite2.0 选取了数据分析、数据缓存、数据服务、图分析、流媒体、软件测试、网络搜索和网络服务等云计算中常用的负载;BigDataBench[16-17] 是由中国科学院计算技术研究所开发的一套互联网大数据应用相关的 Benchmark 集,其覆盖了结构数据、半结构数据和非结构数据,其负载模拟了搜索引擎、社交网络和电子商务等业务模型[18].

一个 Benchmark 必须有特定的目标系统和特定的目标应用类型[16]. 因此,虽然以上 Benchmark 的目标应用具有高通量应用的特点,但是其目标系统不是处理器的微体系结构,没有针对处理器的微体系结构进行测试和评估. 例如 HiBench 用于对 Hadoop 集群性能测试、YCBS 用于对 NoSQL 数据库系统测试、LinkBench 用于对社交图谱数据库测试、BigDataBench 关注互联网服务系统整体性能等.

在目标系统层面,系统级的评价主要关注系统运行的整体性能,如集群内部协同工作的效率、板间互连和通信的效率、系统软件栈的性能等. 而对于处理器的微体系结构的测试和评估主要关注如众核处理器中多个核的并行处理能力、线程在处理器中的调度效率、共享存储的利用效率、Cache 效率、CPU 吞吐量等. 在目标应用层面,学术界对高通量应用的研究尚不完善,缺少一个整体的对高通量应用的归纳、分类和分析的工作. 本文的目标系统是高通量处理器微体系结构,目标应用是高通量应用.

2　高通量应用 Workload 分类模型

2.1　高通量应用及 Workload

高通量处理器是适用于互联网新兴应用负载特征的在强时间约束下能够全局可控地处理高吞吐量请求的高性能处理器系统. 高通量应用和传统的高性能应用存在本质上的区别,表 1 对高通量应用负载特性和高性能负载特性进行了对比. 从表 1 中可以看出,面向网络服务的新型高通量应用在很多方面和面向科学计算的高性能应用存在着不同之处. 传统高性能计算主要针对科学计算应用,程序往往具有较好的局部性,属于数据密集型和计算密集型应用,其所追求的目标是提高单个应用的执行速度,这类应用以 LINPACK[19] 为典型代表. 而高通量应用面向的是新型网络服务,任务并发度大且对实时性有较高要求,其数据量大且程序局部性较差,属于数据密集型和请求密集型应用[20]. 这类应用追求的目标是高通量,即提高单位时间内处理的并发任务数目[21].

Table 1　The Comparative Analysis of High-Throughput Load and Conventional High-Performance Load

表 1　高通量负载与传统高性能负载对比分析

Classification of Workload	Scenes	Goal	Locality	Data Structure	Real-time Demand	Intensity
High-performance Workload	Scientific Computing	High Speed	Good	Regular	Low Demand	Data Computation
High-throughput Workload	Network Services	High Throughput	Poor	Irregular	High Demand	Data Request

通过对业界已有典型的大数据 Benchmark[22-23] 和 UC Berkeley 提出的 patterns 中的相关内容,调研分析了当前热门的高通量研究领域和实用领域,总结出典型的高通量应用包括大数据分析、机器学习、网页搜索、社交网络、电子商务、无线网络控制器和流媒体等.

上述的每一类应用又包括各自特有的 Workload 集合. 表 2 所示是高通量应用及各应用的 Workload 汇总.

在此基础上,本文提出了一种基于高通量应用需求特点的高通量 Workload 的分类模型,对上述高通量 Workload 进行分类和分析.

Table 2 Summary of High-Throughput Application and Workload

表 2 高通量应用及 Workload 汇总

Domain	Workload
Big Data Analysis	Basic Operation: Sort; Grep; Wordcount
	Classification: Naïve Bayes; SVM
	Cluster: *K*-means; Fuzzy *K*-means
	Segmentation: HMM
	Linear Programming: Linear Programming
	Regression Analysis: Regression Analysis
	Recommendation: IBCF
Machine Learning	Semi-supervised Learning: Learning algorithm based on mixed-generation models; Learning algorithm based on low-density partitioning; Graph-based learning algorithm; Learning algorithm based on inconsistency; Collaborative training learning algorithm
	Ensemble Learning: Sequential ensemble learning algorithm; Parallel ensemble learning algorithm
	Probabilistic Graphical Model: Bayesian network model; Markov network model; Hidden markov network model
	Transfer Learning: Inductive transfer learning; Transductive transfer learning; Unsupervised transfer learning
Web Search	Web crawler: BFS; DFS
	Create index library: Html parser; Directory iterative; PageRank; TF-IDF
	Search: Query parser; Database query; Term weight; Document weight; MSet
Social Network	Maximize social network influence: Diffusion degree; Greedy
	Recommended algorithm: CF; CB
	Data sampling method: Breadth-first sampling method; Point-edge sampling method; User uniform sampling method; Peer-driven sampling method; Random walking sampling method
	Topology analysis: Calculate average path length; Calculate aggregation factor
	Data analysis: Link-based structural analysis; Content-based analysis
E-commerce	Search system: Create indexes; Respond to user queries
	Order system: Order creation; Order inquiry; Order modification; State machine
	Database system: Commodity management system; Member management system; Order management system
	Recommended system: Collaborative filtering algorithm; Content-based recommendation algorithm; Recommendation based on clustering algorithm; Product-to-product recommendation
RNC	MACD: SDU receive; Schedule
	RLC: Entity query; Segment
Stream Media	Video codec algorithm: MPEG; H264
	Streaming media transport protocol: Resource reservation protocol; Real-time transport protocol; Real-time transmission control protocol; Real-time streaming protocol

2.2 分类模型介绍及 Workload 选取

对表 2 列出的应用进行应用特征分析，可将高通量应用的需求分为：单位时间内能够处理尽量大的数据量、单位时间内能够处理尽量多的请求数、能够同时支持尽量多的用户在线实时处理数据. 根据上述 3 种高通量应用的需求，我们将高通量 Workload 分为数据处理类、数据服务类和实时交互类这三大类. 图 1 所示是基于高通量应用需求的高通量 Workload 分类模型.

本文所实现的 Benchmark 的选取主要基于 2 点原则：

1) 从 3 类高通量 Workload 中选取代表性 Workload. 根据分析可知，每一类高通量 Workload 在应用特征和性能指标方面有很大的相似性，因此，从每一类高通量 Workload 中集中选取包含至少一个应用领域的 Workload，来代表此类 Workload.

2) 选取的 Workload 是使用量较大的. 要保证从每一大类中选取出来的 Workload 具有代表性，

则需要此 Workload 在测试性能中的使用量大.

基于以上 2 点原则,表 3 所示是本文选取的 HTC-MicroBench 中的 Workload.

数据处理类选取了字符统计、Tera 数据排序、聚类和搜索匹配等 Workload 来实现 HTC-MicroBench[24],因为以上 Workload 均是大数据处理中的基础算法,使用量广泛.并且 Workload 之间相对独立,各个 Workload 所反映出的应用特点、程序特性和对硬件系统的需求也很相似.典型应用有大数据分析及机器学习.

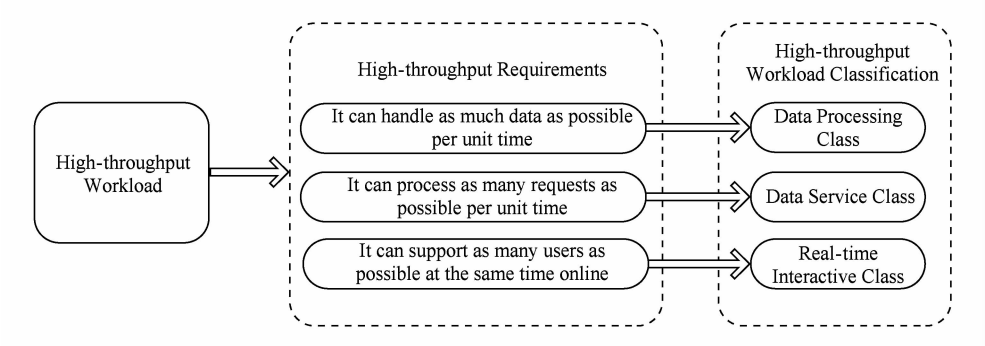

Fig. 1　High-throughput workload classification model

图 1　高通量 Workload 分类模型

Table 3　Workload in HTC-MicroBench

表 3　HTC-MicroBench 中的 Workload

Classification	Workload	Basic Operation	Domain
Data Processing Class	Wordcount	Character statistics	Big data analysis
	Terasort	Tera data sort	Big data analysis
	K-means	Clustering	Big data analysis; Machine Learning
	Grep	Search match	Big data analysis
Data Service Class	Query Parser	Parse the string	Web Search; Social Network; E-commerce; Stream Media
	Database Query	Database operation	Web Search; E-commerce; Stream Media
	Term Weight	Calculate word frequency	Web Search; Social Network; E-commerce
	Document Weight	Calculate web page weights	Web Search
	MSet	Create keyword matching group	Web Search; Social Network; E-commerce; Stream Media
Real-time Interactive Class	SDU Receive	Simple processing of source data	RNC; Social Network; Stream Media
	MACD Schedule	Control the analysis and dumping of data packets	RNC; Social Network; Stream Media
	Entity Query	User instance query	RNC; Social Network; Stream Media
	Segment	Packet protocol processing	RNC; Social Network; Stream Media

数据服务类选取了解析字符串、数据库操作计算词频、计算网页权重和建立关键词匹配组等 Workload 来实现 HTC-MicroBench,因为以上 Workload 的特征要求在单位时间内处理尽量多的请求数,最符合数据服务类 HTC-MicroBench 的特点.典型应用有网页搜索、社交网络、电子商务和流媒体.

实时交互类选取了源数据的简单处理、控制数据包的解析转存、用户的实例查询和数据包的协议处理等 Workload 来实现 HTC-MicroBench,因为以上 Workload 是根据协议的定义,模拟数据包处理流程,并且各个 Workload 之间是一个有机的整体,均需要能够对大量用户进行实时响应与处理,与实时交互类 HTC-MicroBench 表现特征一致.典型应用有无线网络控制器、社交网络、流媒体.

3　HTC-MicroBench 的实现

由第 2 节分析可知,高通量应用的典型特点是含有大量小规模作业,作业之间耦合性低.因此,要提高处理器对高通量应用的吞吐效率,必须支持多处理节点并行处理.如 Hadoop 等系统级的软件栈[25-26],提高应用程序吞吐效率的重要手段就是采用分布式

系统,并行化多节点处理作业.在处理器级别能够并行工作的载体是线程,因此,要实现 HTC-MicroBench,需要将不同的作业处理节点用线程实现,以达到在处理器级别多节点并行的效果.

本文提出了一种适用于面向高通量处理器微体系结构评估的 HTC-MicroBench 的并行模型思想:基于线程的作业处理多节点并行模型.

3.1 数据处理类 HTC-MicroBench 多节点并行化模型

数据处理类 HTC-MicroBench 通常使用 MapReduce 框架,将算法的处理分为 Map 阶段和 Reduce 阶段[27].本文参考以共享内存方式实现的 MapReduce 框架的 Phoenix 系统对数据处理类 HTC-MicroBench 进行了实现[28].在基于线程来实现 MapReduce 框架时,我们在每个阶段,将算法中需要处理的大量数据分块,每块数据作为一个作业,交给不同的处理节点处理,一个处理节点为一个线程.图 2 所示是数据处理类 HTC-MicroBench 实现模型:

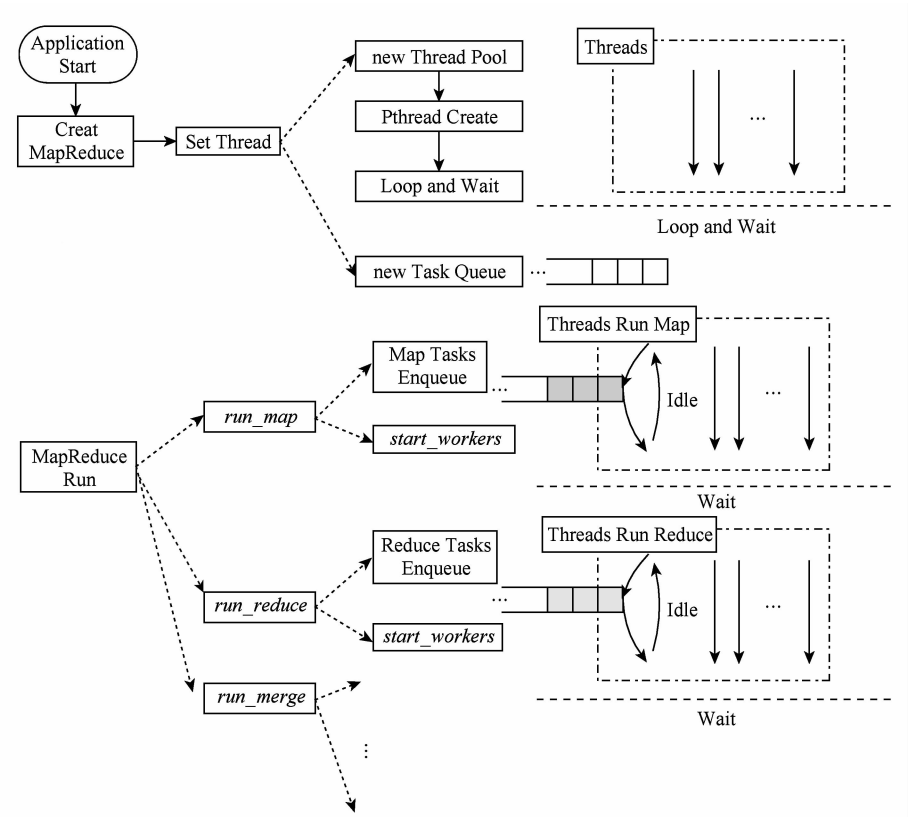

Fig. 2 Implementation model of data processing

图 2 数据处理类实现模型

算法开始运行时,首先会创建 MapReduce 类,在 MapReduce 类中创建线程,当线程创建时,会运行一个线程体,当线程体没有执行任何算法时,线程体会被一个信号量阻塞,进入 Wait 状态,不执行具体操作,同时将作业压入 Task Queue 中.然后,当算法开始执行后,执行到函数 *run_map* 时,DPUMRLIB 内部会调用函数 *start_workers*,打开信号量,此时线程体开始执行通过函数指针传入的函数,同时从 Task Queue 中取作业.运行完一个作业后,线程会再从 Task Queue 中取下一个作业来执行.最后,直到 Task Queue 中的作业被执行完毕,线程再次进入 Wait 状态,等待下一阶段(Reduce 或 Merge)的函数 *start_workers*.

此设计的优点在于,即使 Workload 的 Task 都不相同,但每个物理线程执行的工作量是相同的,有利于均衡.另外,将线程处理的数据记录在 Task 中,并将线程执行的函数用函数指针表示增加了灵活性,线程可以在创建之后完成多种任务直到线程被停止.

基于此模型,本文对大数据分析中的 Wordcount,Terasort,*K*-means,Grep 算法进行了实现,作为数据处理类 HTC-MicroBench.

Wordcount 算法是经常用于大数据文本处理的 1 个算法,用来统计文本中各单词出现的次数.构建 Wordcount 测试用例需要完成 *split*,*map*,*reduce* 这 3 个功能函数.

函数 *split* 用于切分输入数据,将大量数据切分成小块数据,分发给多个函数 *map* 创建的不同线程. 函数 *map* 是 MapReduce 中多线程并行处理部分,多个函数 *map* 由不同的线程来并行执行,函数 *map* 对各自对应的数据块进行词频统计. 函数 *reduce* 用于合并各个线程得到的词频统计的结果. Wordcount 算法执行过程如图 3 所示:

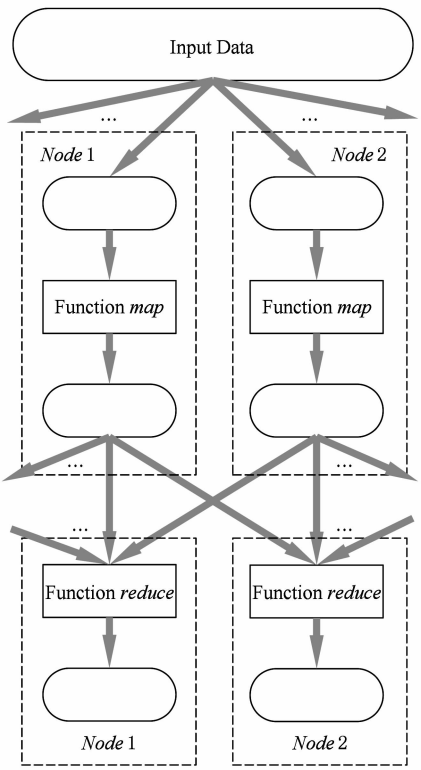

Fig. 3　Execution flow of Wordcount

图 3　Wordcount 算法执行过程

Terasort 算法是一个大规模键值对数据排序的算法,是计算机领域作为系统计算性能测评的标准算法. Terasort 测试用例的构建主要包括 3 个步骤:

1) 数据采样. 从需要排序的所有键值对数据中选取部分数据,采用传统的排序方式进行排序,然后根据需要的分割点个数,从排序后的数据中等步长的距离选择数据点作为分割点,组织到 1 棵 Trie 树中. 在 Map 阶段,此 Trie 树中的数据作为将每个数据分到不同任务中的依据.

2) Map 阶段. Map 阶段的输入数据是对源数据进行简单切分后的一块数据,对其中的每一个数据,根据 Trie 树中的分割点信息,将数据分配到 Reduce 的各线程中.

3) Reduce 阶段. 每个 Reduce 的线程将从 Map 阶段得到的数据进行内部排序,然后按照 Reduce 的线程编号,顺序输出每个线程内部排序后的结果,即可得到最终排序结果.

K-means 算法是用于聚类分析的经典算法,用于将多维空间中的大量数据点按照距离分到不同的集合. 本文所实现的 *K*-means 测试用例流程如下:

1) 随机生成 *D* 维的数据点 *P* 个和 *D* 维的聚类中心点 *C* 个. 对于每个聚类,定义一个 *D* 维的 *Sum*,用于记录一次循环中,属于此聚类的各个数据点各个维度之和,定义一个 *Count* 用于计数属于此聚类的点的个数. *Sum* 和 *Count* 用于在每次循环后,计算新的聚类中心点.

2) 每个数据点作为一个 Map 作业,每个 Map 作业计算此数据点到所有聚类中心点的距离,然后取距离最小的,作为此数据点此次循环所属的聚类. 将此数据点的各维度的值与所属聚类的 *Sum* 各维度的值求和,得到新的 *Sum*,*Count* 值加 1.

3) 所有数据点计算完成后,执行 Reduce 阶段,Reduce 阶段对每个聚类计算新的中心点,计算公式为 *Sum*/*Count*.

4) 循环执行过程 2)3),直到所有数据点新计算出的所属聚类与上次循环计算出的所属聚类相同,结束循环. 得到所有聚类中心点和所有数 据点所属的聚类.

Grep 算法是大数据分析中最基本的算法. 用于在大文本中匹配模式字符串. MapReduce 框架实现 Grep 算法主要流程有 3 个步骤:

1) Split 阶段对较大的源数据文件进行切分,每一数据块对应一个 Map 作业.

2) Map 阶段在对应的数据块中查找模式字符串.

3) Reduce 阶段对 Map 阶段匹配的结果做汇总.

3.2　数据服务类 HTC-MicroBench 多节点并行化模型

数据服务类 HTC-MicroBench 主要是指 C/S 或 B/S 结构中的服务器端程序,数据服务类 HTC-MicroBench 的实现模型,图 4 所示.

具体流程包括 4 个步骤:

1) 创建一个 Listen Thread,用于监听请求,将收到的请求发送到不同线程对应的 Job Queue 中.

2) 每个线程对应一个 Job Queue,用于缓存 Listen Thread 接收到的用户请求.

3) 创建线程池,大量线程作为作业处理节点,循环从对应的 Job Queue 中获取作业,然后做相应处理.

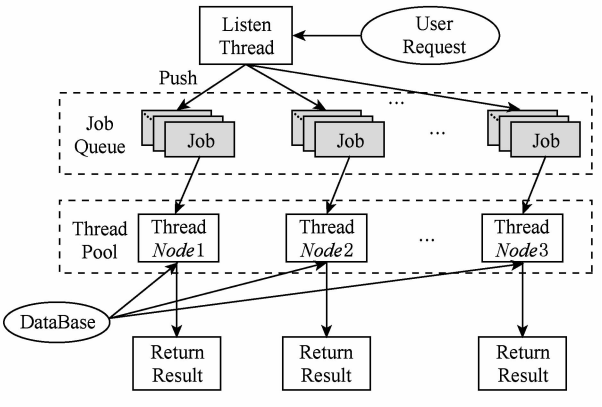

Fig. 4 Programming model of data service

图 4 数据服务类实现模型

4) 将处理结果返回给用户,作为对用户操作的响应.

基于此模型,本文对网页搜索应用响应用户请求中的 Query Parse,Database Query,Term Weight,Document Weight,MSet 算法进行了测试程序的实现,作为数据服务类 HTC-MicroBench.

网页搜索应用中响应用户请求部分的执行过程如图 5 所示.

具体流程有 4 个步骤:

1) 由 Query Parser 对用户输入的 Key Words 进行解析得到 Query,此过程主要是去掉 Key Words 中的 Stop Words,提取单词词干得到 Term,组织各个 Term 之间的布尔关系等,最后形成一个 Query Tree;

2) 对 Query Tree 中每个叶节点的 Term,进行 Database Query,得到相应的数据;

3) 计算 Term Weight,并结合 Query Tree 中的布尔关系,计算 Document Weight;

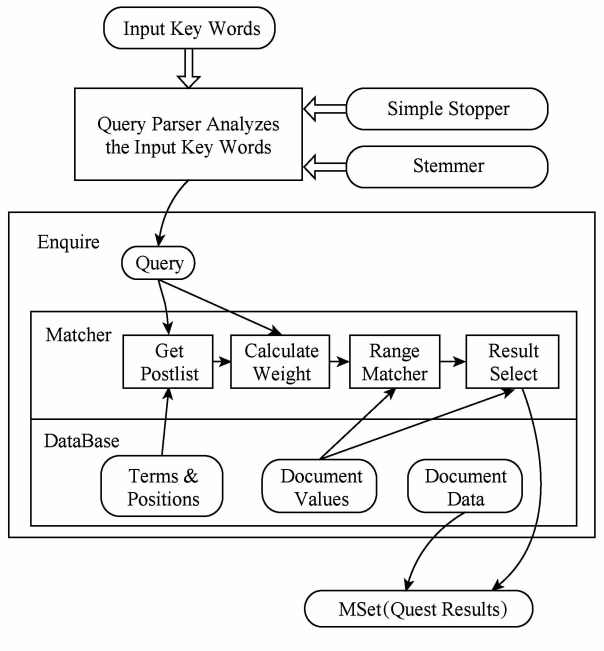

Fig. 5 Execution flow of response to the use request

图 5 响应用户请求部分执行过程

4) 执行 MSet 过程,从查询结果中选取最符合要求的结果,返回给用户.

3.3 实时交互类 HTC-MicroBench 多节点并行化模型

实时交互类 HTC-MicroBench 中,每个作业按照协议处理用户数据. 实时交互类 HTC-MicroBench 的编程模型,如图 6 所示.

具体包括 3 个部分:

1) 每个用户注册之后都会有与具体协议相关的实例表,处理数据时需要查询;

2) 线程池,大量线程作为作业处理节点;

3) 每个作业处理节点对应多个用户,轮询处理每个用户的数据包 Buffer,处理时查询相应的实例表.

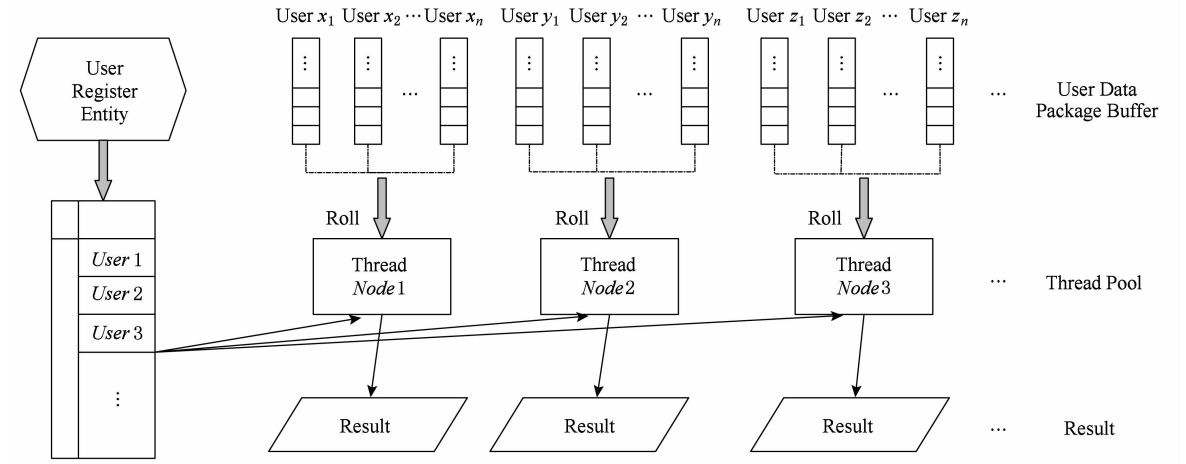

Fig. 6 Programming model of real-time interaction

图 6 实时交互类实现模型

无线网络控制器应用是移动通信陆地无线接入网中数据处理的核心部分,包含用户面和控制面.控制面主要控制初始化、资源的分配以及数据处理结束后的资源释放,用户面负责对接入用户的数据包接收、存储、数据的协议处理和转发,整个用户面按照协议来处理用户数据.

基于此模型,本文对无线网络控制器应用中的 SDU Receive,MACD Schedule,Entity Query,Segment 算法进行 Benchmark 的抽取实现,作为实时交互类 HTC-MicroBench.

无线网络控制器应用中响应用户请求部分的执行过程有 4 个步骤:

1) 利用 SDU Receive 算法,从核心网获取数据,得到含有用户业务信息的源数据,对源数据进行几个操作相对简单的协议层协议处理,得到 RLC 层的服务数据单元(SDU),并将 RLC SDU 缓存于 SDU 缓存区.

2) 为了保证服务质量,MACD Schedule 算法控制每 10 ms 中完成一个用户 SDU 数据包的解析和转存.设计一个作业调度线程,配合一个定时器来进行用户 SDU 数据包调度和分发.当计时完成,作业调度线程从 SDU 数据包缓存区中读取每个用户的数据包,发送到不同作业处理节点等待处理.

3) 利用 Entity Query 算法进行实例查询.每个用户在接入系统时,注册一个用户实例,此实例中包含业务处理的相关信息.在对每个用户的 SDU 数据包解析之前,需要根据用户 *ID*、查询实例表,得到用户 SDU 数据包处理的相关信息,以进行 SDU 数据包的处理.

4) 利用 Segment 函数对数据包进行协议处理.从当前作业节点对应的作业队列中,读取一个 SDU 数据包,解析 SDU,获取头部信息,从头部信息中读取用户 *ID*,根据用户 *ID* 查询实例表,根据实例表中的信息进行数据包的处理,然后将得到 RLC PDU 缓存.

经过 4 个步骤,一个 SDU 数据包即可被成功地分段重组成 PDU 并放入缓存区.

4　实验评估

4.1　基本程序特征评估

根据高通量应用的特征分析,HTC-MicroBench 主要从 3 个方面进行验证:

1) 能够多节点并发处理作业.面向高通量应用的处理器的一个重要目的是在线程级别,支持高通量应用的大规模并发作业,因此,处理器的吞吐效率与用于处理作业的节点数量的关系,可以体现出设计的 Benchmark 是否具有良好的并发性,从而可以验证 HTC-MicroBench 设计的正确性和有效性.

2) 作业之间耦合性较低.由于本文所实现的 Benchmark 是基于共享内存并行编程模型的[29],每个作业是由一个线程来处理,所有线程均在多核或众核处理器上运行.因此,核间共享数据量的大小,可以反映出作业之间耦合性的大小.

3) 数据处理类 HTC-MicroBench 由于处理的数据量大、处理的数据存储规则,因此 Cache 命中率高.数据服务类和实时交互类 HTC-MicroBench 由于处理的是不同用户的数据,数据离散性强,地址空间访问不规则数据空间局部性差,因此 Cache 命中率较低.

基于上述分析,本部分实验主要从作业并发性、作业之间耦合性和 Cache 使用效率 3 方面对 HTC-MicroBench 设计的合理性和有效性进行验证.

4.1.1　作业的并发性评估

对数据处理类 HTC-MicroBench 进行实验.本实验所用平台是 Intel Xeon 处理器,处理器配置如表 4 所示:

Table 4　Intel Xeon Configuration

表 4　Intel Xeon 处理器配置

Processor Information Item	Processor Information Value
Processor Model	Intel Xeon X7550
The Number of Core	8 Cores 16 Threads
CPU Clock Speed/GHz	2.0
Operating System Version	Linux 2.6.32-431
L1 DCache Size/KB	8×32
L1 ICache Size/KB	8×32
L2 Cache Size/KB	8×256
LLC Cache Size/MB	18

各测试程序所用数据集分别为 Wordcount(500 MB),Terasort(200 MB),*K*-means(12 MB),Grep(500 MB),数据处理效率与线程数的关系,如表 5 所示.

本部分主要关注各测试程序吞吐效率的并行加速比,因而对所有测试程序得到的数据,以 1 线程的情况为标准进行归一化,归一化数据吞吐率与线程数之间的关系,如表 6 所示.

数据服务类 HTC-MicroBench 关注的需求指

标是单位时间内处理的请求数. 对数据服务类 HTC-MicroBench,即网页搜索应用中的响应用户请求部分进行实验. 单位时间处理请求量与线程数的关系,如表 7 所示.

Table 5 The Relationship Between Data Processing Efficiency and The Number of Threads

表 5 数据处理效率与线程数的关系 MB/s

Number of Threads	Wordcount	Terasort	*K*-means	Grep
1	11.55	1.44	0.08	67.61
2	21.66	2.75	0.16	138.03
4	39.28	4.66	0.26	265.91
8	65.81	6.31	0.56	504.88

Table 6 The Relationship Between Normalized Data Throughput Rate and The Number of Threads

表 6 归一化数据吞吐率与线程数之间的关系

Number of Threads	Wordcount	Terasort	*K*-means	Grep
1	1.000	1.000	1.000	1.000
2	1.875	1.905	1.936	2.042
4	3.401	3.231	3.221	3.933
8	5.698	4.377	6.942	7.468

Table 7 The Relationship Between the Throughput and Number of Threads

表 7 单位时间处理请求量与线程数的关系

Number of Threads	Throughput Rate/(unit·s^{-1})	Normalized Throughput Rate
1	113.180	1.000
2	225.209	1.990
4	450.178	3.978
8	887.872	7.845

实时交互类 HTC-MicroBench 关注的高通量应用需求指标是在保证对每个用户的服务质量的前提下,能够同时支持在线用户数. 实时交互类 HTC-MicroBench,即无线网络控制器应用进行实验. 支持用户数与线程数之间的关系,如表 8 所示:

Table 8 The Relationship Between the Number of Sustaining Users and the Number of Threads

表 8 支持用户数与线程数之间的关系

Number of Threads	Number of Users Supported	Normalized Number of Users Supported
1	5700	1.000
2	7500	1.316
4	9000	1.579
8	10160	1.782

对以上各个测试程序的归一化测试结果做图表,HTC-MicroBench 中各测试程序的并行加速能力,如图 7 所示:

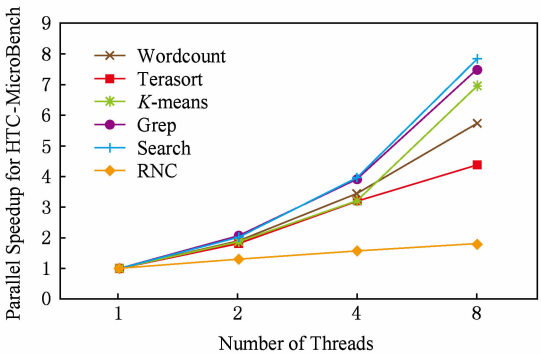

Fig. 7 The ability of parallel acceleration for HTC-MicroBench

图 7 HTC-MicroBench 各测试程序并行加速能力

由图 7 实验结果可以看出,HTC-MicroBench 中的测试程序均有较好的并行加速特性,即各测试程序的作业之间具有良好的并发性,能够反映出高通量应用并发性的特点. 当线程数为 8 时,加速比较线程数为 1 时有了 4～8 倍的提高.

4.1.2 作业之间耦合性

本实验使用 Intel 的硬件性能测试软件 VTune 进行实验,通过统计核间共享数据量占访存总量的比例来反映作业之间的耦合性,实验结果如图 8 所示.

由实验结果可以看出,HTC-MicroBench 核间共享数据量占访存指令总数的比例是极小的,Wordcount 只有不到 0.02%,Terasort 不到 0.19%,*K*-means 不到 0.06%,Grep,Search,RNC 均都不到 0.02%. 其中 Terasort 核间共享数据量所占比例最大,由于 Terasort 算法中,大部分需要多线程之间共享数据,但是,核间共享数据量也只有不到 0.19%,而其他几个算法所占比例几乎是可以忽略的. 因此,HTC-MicroBench 能够正确反映出各作业之间低耦合性的特点.

4.1.3 Cache 效率的评估

Splash2[30] 是传统并行应用领域中典型的 Benchmark,是斯坦福大学推出的共享存储并行应用 Benchmark,其中选取的算法和应用都是传统高性能并行应用,本实验以 Splash2 为对比,选取其中的 Raytrace, Cholesky, Radix, Ocean, Radiosity, Barnes 这 6 个测试程序进行指标测试,验证 HTC-MicroBench 与传统的并行应用在 Cache 使用效率方

面的差别,同时说明本文所实现的 HTC-MicroBench 的有效性.

本实验使用 Linux 的性能评估软件 Perf 进行实验测试,分别得到 HTC-MicroBench 和 Splash2 的 L1 Cache Hit Rate,L2 Cache Hit Rate,LLC Cache Hit Rate 这 3 个指标,其分别如图 9～11 所示.

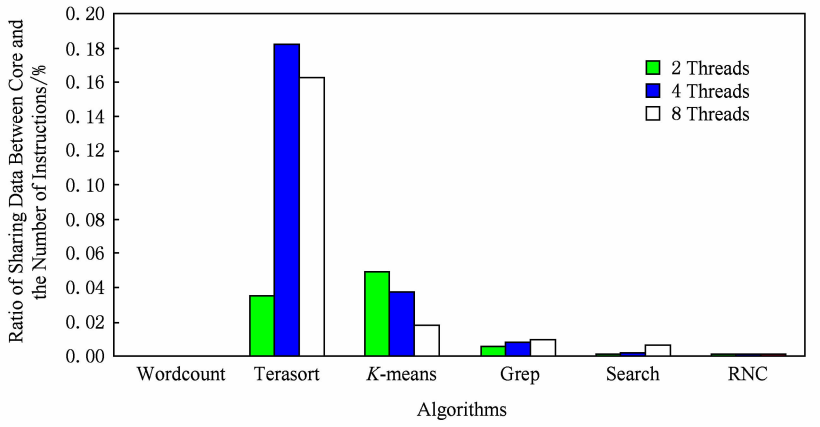

Fig. 8　The ratio of sharing data between cores for HTC-MicroBench

图 8　HTC-MicroBench 各测试程序核间共享数据情况

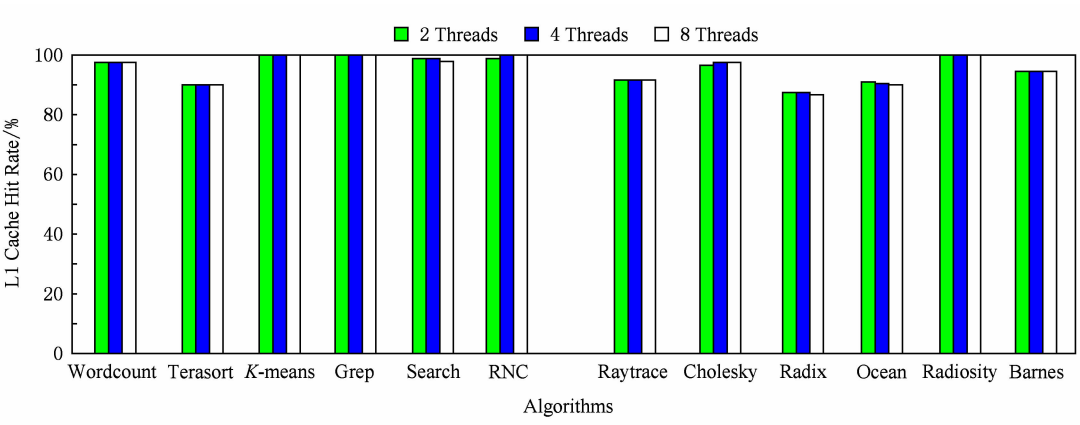

Fig. 9　The L1 cache hit rate of HTC-MicroBench and Splash2

图 9　HTC-MicroBench 和 Splash2 各测试程序 L1 cache 命中率

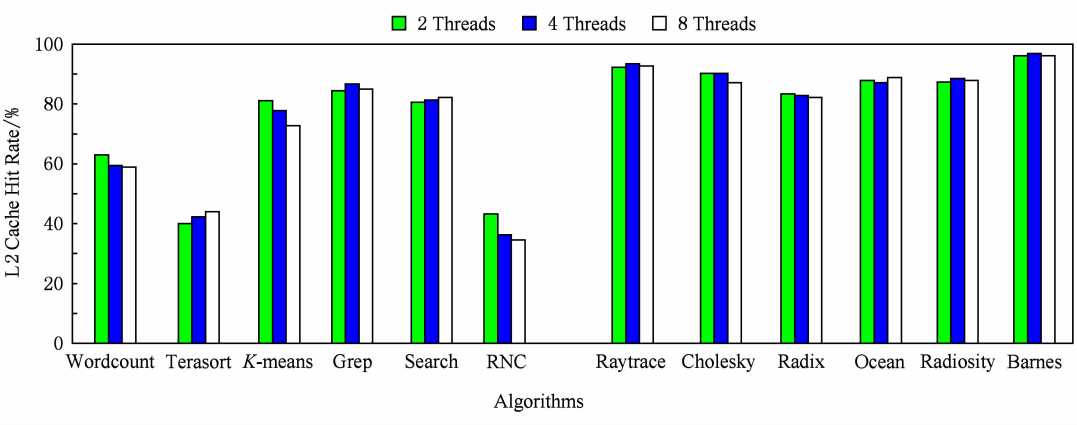

Fig. 10　The L2 cache hit rate of HTC-MicroBench and Splash2

图 10　HTC-MicroBench 和 Splash2 各测试程序 L2 cache 命中率

由实验结果看出,HTC-MicroBench 和 Splash2 的 L1 Cache Hit Rate 值都在 90%以上,没有明显的差别.因为两者数据都具有较好的局部性特征,因此 L1 Cache 命中率高.而对于 L2 Cache 和 LLC

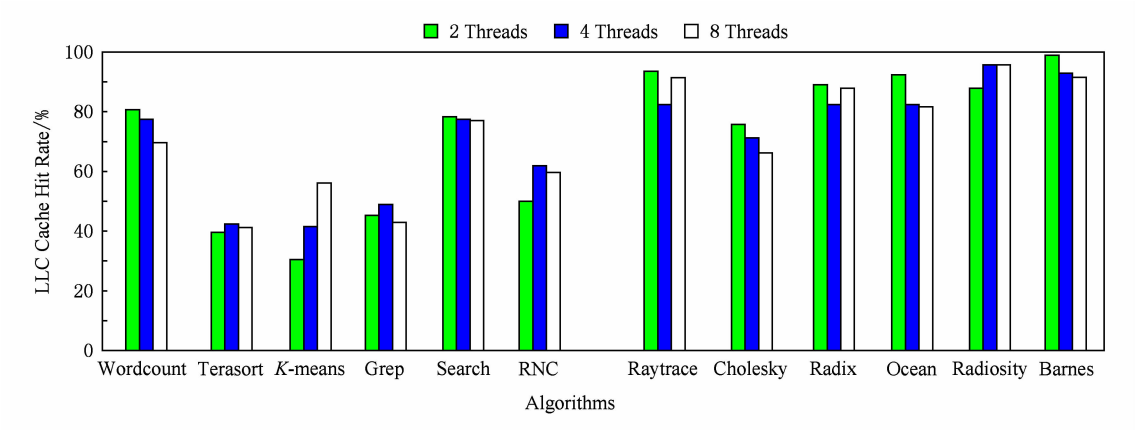

Fig. 11 The LLC cache hit rate of HTC-MicroBench and Splash2

图 11 HTC-MicroBench 和 Splash2 各测试程序 LLC cache 命中率

Cache,HTC-MicroBench 的平均命中率比 Splash2 程序的明显要低. 对于 L2 Cache,HTC-MicroBench 的平均命中率为 65%,而 Splash2 程序为 90%;对于 LLC Cache,HTC-MicroBench 的平均命中率为 50%,而 Splash2 程序为 80%.

这是由于 L1 Cache 具有较高的命中率,基本消耗了 HTC-MicroBench 应用的空间局部性特征,而且 HTC-MicroBench 应用具有流式特征,时间局部性并不强,且数据量巨大,时间局部性超出了低级 Cache 的相联度,导致 L2 Cache 和 LLC Cache 命中率较小. Splash2 相对于 HTC-MicroBench 应用来说,时间局部性较强,数据量相对较低,因而 L2 Cache 和 LLC Cache 仍具有相对较高的命中率.

4.2 TILE-Gx 和 Intel Xeon 处理器评估

TILE-Gx 处理器是 Tilera 公司推出的一款众核处理器[31],使用了一种新的 iMesh 网络,iMesh 是一种 5 层网络结构,使得各个处理器核之间能够高效通信. TILE-Gx 处理器达到了处理效率更高,并发处理能力更强,能耗更小和扩展性更强等方面的效果. 而处理器中的每一个核的处理能力有限,尤其对浮点计算的支持能力很弱. 由于 TILE-Gx 处理器具有众核,并发处理能力强的特点,因此本实验采用的 TILE-Gx 系列中的 TILE-Gx 8036 处理器.

TILE-Gx 8036 处理器有 36 个核,L1 和 L2 这 2 级 Cache 是独立的,L3 是共享的. 表 9 所示是使用的 TILE-Gx 和 Xeon 这 2 种处理器的基本参数.

本实验主要关注 TILE-Gx 8036 与 Xeon X7550 在并行加速方面的性能对比. 由于 HTC-MicroBench 主要是处理大量的规模较小、耦合性较低的作业,因此处理器的并行加速能力对作业处理能力具有决定性的影响[32]. 由于 Xeon 处理器只有 16 个硬件线程,因此以下实验均在 16 个线程及以下进行并行加速能力的对比.

Table 9 The Basic Parameters of TILE-Gx and Xeon

表 9 TILE-Gx 和 Xeon 的基本参数

TILE-Gx 8036	Xeon X7550
36 Cores	8 Cores 16 Threads
64 b	64 b
1.2 GHz	2.0 GHz
26 MB L3 Cache	8×32 L1 DCache
Five Independent Mesh Networks	8×32 L1 ICache
12 MB On-chip Cache	8×256 KB L2 Cache
60 Tbps iMesh Bandwidth	18 MB L3 Cache

对数据处理类 HTC-MicroBench,在 Xeon 和 TILE-Gx 处理器分别对不同线程数的 Workload 单位时间处理数据量以单线程为标准进行归一化,得到归一化单位时间处理数据量与线程数之间的关系,如图 12 所示:

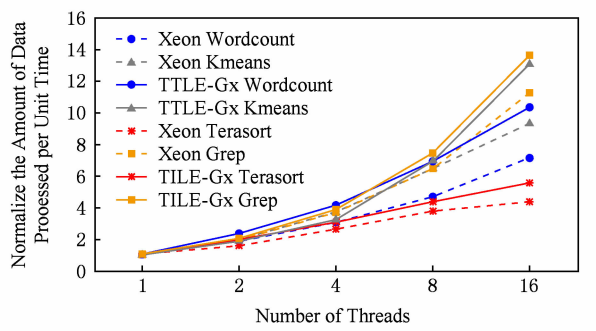

Fig. 12 The comparison between normalized throughput rate and the number of threads in Xeon and TILE-Gx

图 12 数据处理类测试程序在 2 种处理器平台并行加速比

由实验结果看出,对于数据处理类 HTC-MicroBench 中的 4 种 Workload,在 TILE-Gx 8036 处理器单位时间处理的数据量均大于在 Xeon 处理器单位时间处理的数据量.如矩形实折线部分 Grep 算法在 TILE-Gx 8036 处理器运行线程数达到 16 时,加速比比线程为 1 时提高了 14 倍,而矩形虚折线部分 Grep 算法在 Xeon 处理器运行线程数达到 16 时,加速比仅提高了 11 倍.因此 TILE-Gx 8036 处理器对数据处理类 HTC-MicroBench 有更好的并行加速比.

对数据服务类 HTC-MicroBench,本实验测试单位时间内在 Xeon 和 TILE-Gx 处理器分别所能处理的请求数量,以单线程为基准做归一化,如图 13 所示:

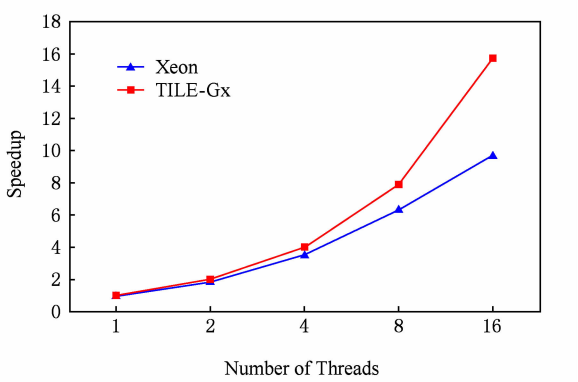

Fig. 13　The test result of speedup in Xeon and TILE-Gx processor (data service applications)

图 13　数据服务类测试程序在 2 种处理器平台并行加速比

由实验结果看出,TILE-Gx 8036 处理器在线程数达到 16 时,加速比有 16 倍的提高,而 Xeon 处理器对数据处理类 HTC-MicroBench 的加速比仅提高了 9 倍.TILE-Gx 8036 处理器对数据服务类 HTC-MicroBench 也具有更高的并行加速能力.

对实时交互类 HTC-MicroBench,主要关注在保证对每个用户的服务质量的前提下,能够同时支持在线用户数.对于 RNC 用户面 Benchmark,保证服务质量是指用户数据包由于系统繁忙而导致的丢包率在一定的范围之内,本文取丢包率小于 5% 为可接受的.

图 14 所示是实验得到 RNC 用户面 Benchmark 在 Xeon 和 Tilegx 这 2 种处理器上的结果.

由实验结果看出,TILE-Gx 8036 处理器在线程数提高到 16 时,能支持的用户数是线程数为 1 时的 11 倍;而 Xeon 处理器对数据处理类 HTC-MicroBench 的加速比仅提高了 3 倍.对于实时交互类 HTC-MicroBench,TILE-Gx 处理器同样具有更高的并行加速能力.

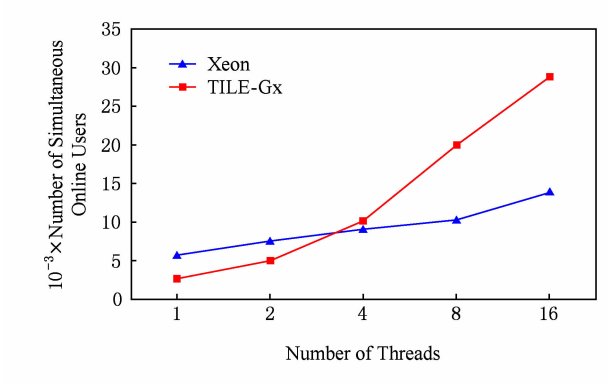

Fig. 14　The test result of speedup in Xeon and TILE-Gx processor (real-time interaction applications)

图 14　实时交互类测试程序在 2 种处理器平台并行加速比

综上实验结果可以看出,本文实现的面向高通量应用的 HTC-MicroBench 对 Tile-Gx 众核处理器和 Xeon 多核处理器均具有良好的并行加速比和并行处理能力,并且对于 Tile-Gx 众核处理器微体系结构具有相对更好的评估能力.

这也从另一个角度佐证了本文所抽取和实现的 Benchmark 对面向高通量应用的处理器微体系结构设计的评估是合理有效的.

5　结束语

本文实现了一个面向高通量应用的处理器微体系结构设计评估的 Benchmark——HTC-MicroBench.首先,本文总结了高通量应用以及各应用相应的 Workload,提出了面向高通量应用的处理器微体系结构设计评估的基准测试程序——HTC-MicroBench,并提出了一种基于需求特点的高通量应用 Workload 的分类方法.其次,提出并实现了一种基于线程的作业处理节点并行化模型,完成了 HTC-MicroBench 的设计和实现.最后,通过 2 组实验评估:1)对 HTC-MicroBench 中的 Workload 的并发性、耦合性和 Cache 使用效率进行了评估;2)使用 HTC-MicroBench 对比测试了 8 核 Xeon 处理器和众核 TILE-Gx 这 2 种处理器平台的并行加速能力,实验结果验证了本文所实现的 HTC-MicroBench 在面向高通量应用的处理器微体系结构评估方面的具有有效性和合理性.

参 考 文 献

[1] IBM. What is big data? Bringing big data to the enterprise [EB/OL]. [2016-10-18]. http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[2] Lü Chao, Dai Chen, Zhang Weihua. Research on benchmarks in computer architecture [J]. Computer Applications and Software, 2013, 30(10): 189-193 (in Chinese)
(吕超, 戴晨, 张为华. 计算机体系结构基准测试程序集的研究[J]. 计算机应用与软件, 2013, 30(10): 189-193)

[3] Divya P, Shivani G. Prototyping and in-depth analysis of big data benchmarking [C] //Proc of the 14th IEEE Int Conf on Ubiquitous Computing and Communications. Piscataway, NJ: IEEE, 2015: 1222-1229

[4] Xu Jie, Wang Hua, Wu Xiaohua, et al. Discussion on SPEC benchmark suites and evalution indicators [J]. Experiment Science and Technology, 2010, 8(6): 21-24 (in Chinese)
(徐洁, 王华, 吴晓华, 等. 浅析 SPEC 基准测试程序集及评价指标[J]. 实验科学与技术, 2010, 8(6): 21-24)

[5] PARSEC. What is PARSEC [EB/OL]. [2016-10-28]. http://parsec.cs.princeton.edu/

[6] Wang Xiaoying, Li Sanli. HPCC: A memory access model oriented benchmark—A potential test method to replace HPL in TOP500 [J]. Mini-Micro Systems, 2006, 27(5): 950-955 (in Chinese)
(王晓英, 李三立. HPCC: 面向存储访问模型的基准测试——一种可能替代 TOP500 HPL 的测试方法[J]. 小型微型计算机系统, 2006, 27(5): 950-955)

[7] Clapp R, Dimitrov M, Kumar K, et al. Quantifying the performance impact of memory latency and bandwidth for big data workloads [C] //Proc of IEEE Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2015: 213-224

[8] Wang Peng, Zhang Li. Review and outlook of large scale data processing system for big data [J]. Chinese High Technology Letters, 2015, 25(8): 793-801 (in Chinese)
(王鹏, 张利. 大数据处理系统的研究进展与展望[J]. 高技术通讯, 2015, 25(8): 793-801)

[9] Huang Shengsheng, Huang Jie, Dai Jinquan, et al. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis [C] //Proc of IEEE Int Conf on Data Engineering Workshops. Piscataway, NJ: IEEE, 2010: 41-51

[10] Cooper B F, Silberstein A, Tam E, et al. Benchmarking cloud serving systems with YCSB [C] //Proc of the 1st ACM Symp on Cloud Computing. New York: ACM, 2010: 143-154

[11] DCBench. What is DCBench [EB/OL]. [2016-11-09]. http://prof.ict.ac.cn/DCBench/

[12] Jia Zhen, Wang Lei, Zhan Jianfeng, et al. Characterizing data analysis workloads in data centers [C] //Proc of IEEE Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2013: 66-76

[13] Li Chao, Hu Yang, Liu Longjun, et al. Towards sustainable in-situ server systems in the big data era [C] //Proc of the 42nd ACM/IEEE Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2015: 14-26

[14] Armstrong T G, Ponnekanti V, Borthakur D, et al. LinkBench: A Database Benchmark Based on the Facebook Social Graph [M]. New York: ACM, 2013

[15] Ferdman M, Adileh A, Kocberber O, et al. Clearing the clouds: A study of emerging scale-out workloads on modern hardware [C] //Proc of the 17th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2012: 37-48

[16] Bigdatabench. A big data benchmark suite [EB/OL]. [2016-11-18]. http://prof.ict.ac.cn/BigDataBench/

[17] Gao Wanling, Zhu Yuqing, Jia Zhen, et al. BigDataBench: A big data benchmark suite from Internet services [C] //Proc of the 20th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2014: 488-499

[18] Jun Sang-Woo, Liu Ming, Lee Sungjin, et al. BlueDBM: An appliance for big data analytics [C] //Proc of ACM/IEEE Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2015: 1-13

[19] BLAS. High performance BLAS library homepage [EB/OL]. [2016-11-20]. http://www.cs.utexas.edu/users/flame/goto

[20] Wang Lei, Ren Rui, Zhan Jianfeng, et al. Characterization and architectural implications of big data workloads [C] //Proc of IEEE Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2015: 145-146

[21] Li Guojie. The challenge of large data to computer system [C/OL] //CNCC Big Data Forum. 2013 [2016-11-20]. http://cncc.ccf.org.cn/cn/news/index/16
(李国杰. 大数据对计算机系统的挑战[C/OL] //CNCC 大数据论坛. 2013 [2016-11-20]. http://cncc.ccf.org.cn/cn/news/index/16)

[22] Apache Hadoop. What is apache hadoop? [EB/OL]. [2016-12-10]. http://hadoop.apache.org/

[23] Sort Benchmark. Sort benchmark homepage [EB/OL]. [2016-12-18]. http://sortbenchmark.org/

[24] Xie Biwei, Liu Xu, Zhan Jianfeng, et al. Characterizing data analytics workloads on Intel Xeon Phi [C] //Proc of IEEE Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2015: 114-115

[25] Mandal B, Sethi S, Sahoo R K. Architecture of efficient word processing using Hadoop MapReduce for big data applications [C] //Proc of IEEE Int Conf on Man and Machine Interfacing. Piscataway, NJ: IEEE, 2015: 1-6

[26] Zhang Hong, Wang Xiaoming, Cao Jie, et al. Rearch on optimized MapReduce model od Hadoop cloud platform [J]. Computer Engineering and Application, 2016, 52(22): 22-25 (in Chinese)
(张红, 王晓明, 曹洁, 等. Hadoop 云平台 MapReduce 模型优化研究[J]. 计算机工程与应用, 2016, 52(22): 22-25)

[27] Kim S Y, Bottleson J, Jin Jingyi, et al. Power efficient MapReduce workload acceleration using integrated-GPU [C] //Proc of the 1st IEEE Int Conf on Big Data Computing Service and Applications. Piscataway, NJ: IEEE, 2015: 162-169

[28] Richard M Y, Anthony R, Christos K. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system [C] //Proc of IEEE Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2009: 198-207

[29] Mohammed M S, Abandah G A. Communication characteristics of parallel shared-memory multicore applications [C] //Proc of the 3rd IEEE Jordan Conf on Applied Electrical Engineering and Computing Technologies. Piscataway, NJ: IEEE, 2015: 1-6

[30] Bienia C, Kumar S, Li Kai. Parsec vs Splash-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors [C] //Proc of IEEE Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2008: 47-56

[31] Zhong Cheng, Qu Zengyan, Yang Feng, et al. Efficient and scalable thread-level parallel algorithms for sorting multisets on multi-core systems [J]. Journal of Computers, 2012, 7(1): 30-41

[32] Xiong Wen, Yu Zhibin, Lieven E, et al. SZTS: A novel big data transportation system benchmark suite [C] //Proc of the 44th IEEE Int Conf on Parallel Processing. Piscataway, NJ: IEEE, 2015: 819-828

Xue Rui, born in 1993. PhD candidate. Student member of CCF. Her main research interests include high throughput computing architecture and software simulation.

Miao Futao, born in 1989. Master, engineer. His main research interests include the utilization of big data and machine learning in financial software system.

Ye Xiaochun, born in 1981. PhD, associate professor. Member of CCF. His main research interests include high throughput computing architecture and software simulation.

Sun Ninghui, born in 1968. PhD, professor and PhD supervisor. Member of CCF. His main research interests include computer architecture, high performance computing and high throughput computing.

Xu Wenxing, born in 1982. PhD, associate professor. Her main research interests include modeling and optimization, intelligent computing and decision making.