基于固态硬盘内部并行的数据库表扫描与聚集

范玉雷 赖文豫 孟小峰
(中国人民大学信息学院 北京 100872)

摘 要 随着基于闪存的固态硬盘在个人计算机和企业服务器上的广泛应用,固态硬盘受到学术界和工业界越来越多的关注.除了具有闪存存储器的优良特性之外,固态硬盘内部还具有丰富的并行特性.传统数据库系统的物理操作表扫描和上层聚集操作是针对磁盘的机械特性和对称读写特性而设计的,并不能发挥固态硬盘内部并行特性的优势.文中首先将固态硬盘作为一个黑盒进行探测以了解其内部的并行特性.在此基础上,对传统数据库表扫描操作进行相应的改进,提出一种并行表扫描模型 ParaSSDScan 以充分利用固态硬盘内部丰富的并行特性.其次,基于并行表扫描模型,文中还提出一种高效的并行聚集操作模型 ParaSSDAggr,并利用该聚集操作模型实现几种常见聚集操作.最后,通过实验表明并行表扫描和并行聚集操作的性能较之传统数据库表扫描和聚集操作的性能分别提高了 3 倍和 4 倍,同时实验结果还表明并行聚集操作对内存的需求不大.并行表扫描和并行聚集操作大大提高了表扫描和聚集操作的性能,充分说明了固态硬盘内部并行特性的优越性.

关键词 固态硬盘;闪存数据库;并行表扫描;并行聚集
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2012.02327

Database Table Scan and Aggregation by Exploiting Internal Parallelism of SSDs

FAN Yu-Lei LAI Wen-Yu MENG Xiao-Feng
(*School of Information, Renmin University of China, Beijing* 100872)

Abstract With the extensive application of flash-based SSDs in personal computers and enterprise servers, SSDs have attracted more and more attention from the academia and industry. In addition to the excellent characteristics of flash memory, there is wealth internal parallelism in SSDs. Table scan and aggregation in traditional database systems are designed based on properties of hard disk, such as mechanical property and symmetrical read/write property. They can't take advantages of the internal parallelism of SSDs when traditional database systems are built on SSDs. Firstly, we detect internal parallelism of SSDs seemed as a black box. And then, we propose a parallel table scan model, ParaSSDScan, to take advantages of the internal parallelism of SSDs. Secondly, based on ParaSSDScan, we also propose an efficient parallel aggregation model, ParaSSDAggr, and achieve several common aggregation operations with ParaSSDAggr. Finally, experiments show that, compared to traditional table scan and aggregation operations, there are 3x and 4x improvement of the performance for ParaSSDScan and ParaSSDAggr which cost little memory. ParaSSDScan and ParaSSDAggr largely speed up table scan and aggregation operations, which fully show the superiority of internal parallelism of SSDs.

Keywords solid state disk; flash-based database; parallel table scan; parallel aggregation

收稿日期:2012-06-05;最终修改稿收到日期:2012-09-09.本课题得到国家自然科学基金项目(60833005,61070055)和核高基重大专项(2010ZX01042-002-003)资助.**范玉雷**,男,1984 年生,博士研究生,主要研究方向为闪存数据库存储管理、查询处理和查询优化、索引和恢复技术研究.E-mail: fyl815@ruc.edu.cn.**赖文豫**,男,1989 年生,硕士研究生,主要研究方向为闪存数据库系统存储管理、查询处理和查询优化技术研究.**孟小峰**,男,1964 年生,博士,教授,博士生导师,主要研究领域为 Web 数据集成、XML 数据库系统、移动对象管理.

1 引 言

随着闪存技术的飞速发展,闪存设备凭借其小巧轻便、抗震、耐高/低温、耗电量小、读写速度快等优良特性在各种场景中都得到极大的发展与应用,小到传感器,大到服务器都能看到它的影子.近几年,固态硬盘作为一种大容量、非易失、可替代磁盘的闪存设备,受到了学术界和工业界的热烈追捧.固态硬盘是一种由闪存存储器、闪存转换层和控制器组成的块设备[1],因此固态硬盘具有很多闪存存储器的优良特性.随着固态硬盘技术的发展,固态硬盘在个人计算机和企业服务器上都得到了广泛应用.在个人计算机中固态硬盘可以作为二级存储设备部分或者完全替代传统磁盘,比如苹果笔记本 MacBook Air 以及一些超级本中.在企业服务器领域,固态硬盘可以作为磁盘的缓冲区,例如微软在多人在线游戏和数据去重等系统中均使用固态硬盘作为磁盘的写缓冲[2-4],从而大大提高了系统的处理能力.此外,固态硬盘还可和磁盘一起做二级存储设备[5].

除了具有闪存存储器的优势之外,固态硬盘还具有丰富的内部并行特性[6].固态硬盘内部并行特性使得存储设备支持并行处理成为可能,而磁盘的请求处理主要是通过磁盘关键组件磁头来完成,磁头同一时刻只能为一个请求进行寻道、定位和读取数据,不能支持存储设备上的并行处理.传统数据库是根据磁盘的机械运动和读写对称等特性而设计,所以直接把传统数据库移植到固态硬盘之上,很难充分发挥固态硬盘的优势,尤其是固态硬盘内部丰富的并行特性.

扫描操作是数据库系统中一个最基本的物理操作.扫描操作分为两类[7]:表扫描和索引扫描.针对磁盘机械特性设计的表扫描从磁盘上一块接一块的读取指定表的数据[7].数据库系统提供的部分查询处理过程被转化为底层的表扫描这个物理操作[7],例如排序、聚集和连接,因为它们都需要读取整个表的数据块,所以物理操作表扫描对查询处理有着重要影响.聚集操作[7]主要有求和、求最大/最小、计数和求平均值等,在 OLTP 系统和 OLAP 系统中都有着十分重要的作用.

本文针对固态硬盘内部并行特性设计高效的物理操作表扫描和聚集操作模型,并且利用该模型实现求和、求平均值、计数和求最大/最小.总体来说,本文的主要贡献如下:

(1)通过测试研究了不同厂家不同型号固态硬盘的内部并行特性.

(2)设计高效的物理操作表扫描模型 ParaSSDScan,使其充分利用固态硬盘内部并行特性,从而大大提高表扫描性能.

(3)针对 ParaSSDScan 设计高效的聚集操作模型 ParaSSDAggr,使其充分利用固态硬盘内部并行特性,并基于该模型实现了几种聚集操作.

本文第 2 节介绍固态硬盘内部并行特性、问题描述和相关工作;第 3 节主要从测试角度分析固态硬盘内部并行特性;第 4 节提出一种新的基于固态硬盘内部并行特性的扫描操作模型;第 5 节描述基于并行表扫描操作的聚集操作模型以及一些聚集操作的实现;第 6 节通过实验对比分析阐述基于固态硬盘并行特性的表扫描操作和聚集操作的性能优势;最后,总结本文.

2 问题定义及相关工作

固态硬盘具有磁盘所不具有的丰富的内部并行特性,然而传统数据库却不能充分发挥固态硬盘的这种特性,因此越来越多的研究者开始关注固态硬盘内部并行特性在数据库中的应用.

2.1 固态硬盘结构和内部并行结构

相对于磁盘的盘片和磁头等机械构造,固态硬盘没有机械组件,主要由接口逻辑、内置缓冲区、控制器和闪存存储器组成[8],如图 1 所示.

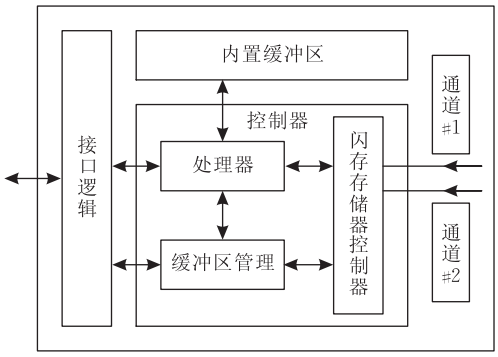


图 1 固态硬盘内部结构

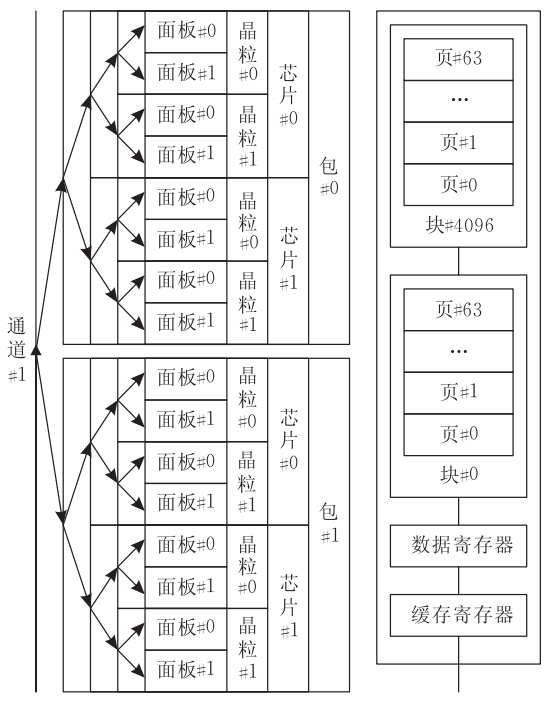
接口逻辑把闪存存储器封装成类磁盘的块设备.内置缓冲区的作用主要有两个:(1)缓冲逻辑地址和物理地址之间的映射信息;(2)缓冲用户读写数据.内置缓冲区可以使用 DRAM,但是掉电会丢失数据.为了防止掉电时丢失数据,一般都会采用非易失

存储器或者带电池的 DRAM[9] 作为内置缓冲区.

控制器是固态硬盘的核心组件，主要包括处理器、缓冲区管理单元和闪存存储器控制器. 缓冲区管理单元主要负责管理内置缓冲区. 处理器主要实现闪存转换层和错误校验等功能. 闪存转换层负责闪存存储器的物理地址和接口接收的逻辑地址之间的映射、闪存存储器空间回收以及闪存存储器的磨损平衡. 闪存转换层按照地址映射模式可以分为 4 类：页级映射、块级映射、混合映射和其它映射模式[10].

闪存存储器控制器主要负责闪存存储器的连接、控制、读写命令传输、地址传输和数据传输等等. 闪存存储器控制器通过通道与闪存存储器相连. 通道是闪存存储器和闪存存储器控制器之间的一个桥梁，负责命令、地址和数据的传输. 如图 1 所示的闪存存储器控制器连接有两个通道. 两个通道是可以并行操作的，故第 1 级并行的并行度是 2.

闪存存储器在通道上的组织方式可用图 2(a)来表示. 按照并行粒度从大到小的顺序它们分别是通道级并行、包级并行、芯片级并行、晶粒级并行和面板级并行. Q 个可以并行访问的包被连接到同一个通道上；包是由 P 个可以并行访问的芯片构成；芯片是由 K 个可以并行访问的晶粒构成；晶粒是由 L 个可以并行访问的面板构成. 面板是最小的并行操作单元，即第 5 级并行. 图 1 和图 2(a)所示的固态硬盘中，每一级的并行度均为 2，即 $L=K=P=Q=2$.



(a) 5层并行结构　　(b) 面板

图 2　固态硬盘内部并行结构

如图 2(b)所示，面板又是由 M 个闪存块、数据寄存器和缓存寄存器通过串联的方式构成. 块是由 N 个页以串联的方式构成的最基本的擦除操作单元. 页是最基本的读写操作单元. 如图 2(b)所示，$N=64$，$M=4096$.

2.2　问题定义

磁盘的关键组件是盘片与磁头，通过盘片的旋转和磁头的寻道来找到指定位置，然后读取或者写入数据. 当读写数据时，盘片的旋转和磁头的寻道开销很大程度上制约了磁盘系统的性能. 此外，磁盘的磁头在任何时候都不可能实现真正的并行，即在同时刻不能为两个线程定位和读写数据. 传统数据库是基于磁盘特性设计的，所以主要还是利用盘片旋转和磁头寻道来定位读写数据所在的位置.

在传统数据库系统中，扫描操作分为表扫描和索引扫描两类. 表扫描是一种关键的物理操作，尤其是对于需要读取整个表的上层操作——聚集、排序和连接等等. 在传统数据库系统中，表扫描采用一个数据块接着一个数据块的方式读取数据. 在固态硬盘上也可以采用同样的方式读取数据，但是不能充分发挥固态硬盘内部并行的优势. 所以我们遇到的新问题是：如何让表扫描操作和上层的聚集操作充分利用固态硬盘的内部并行特性使其最大限度地提高性能？

2.3　相关工作

对于固态硬盘内部并行，研究者主要从 3 个角度进行考虑：(1) 固态硬盘内部并行结构设计者的角度；(2) 固态硬盘内部并行结构探测者的角度；(3) 固态硬盘内部并行特性应用者的角度.

固态硬盘内部结构的设计主要包括闪存存储器连接方式、固态硬盘内置缓冲区管理和固态硬盘内请求调度处理. 文献[11-13]从闪存存储器的连接方式入手，分别设计了页级、芯片级和通道级并行连接方式，并且针对不同的并行连接粒度设计所需的内存数据结构和操作处理方法. 文献[14]重新设计了基于并行结构的固态硬盘内置缓冲区管理策略，提高固态硬盘的整体性能. 文献[8]通过设计有效的固态硬盘内请求调度处理方法，提高固态硬盘的写性能.

在文献[15-17]中，作者把固态硬盘作为黑盒，探测固态硬盘的特性. 文献[16]根据目前公开的固态硬盘物理结构提出了一种抽象的固态硬盘组织结构，该结构的 3 个关键部分分别为最合适操作单元(数据块)、并行粒度以及固态硬盘内采用的地址映射策略(详见 3.1 节). 文献[17]阐述了固态硬盘支持的高级命令、空间分配和数据粒度与固态硬盘并

行特性的相互影响.

固态硬盘之上的数据库系统需要重新设计使其充分利用固态硬盘内部并发特性.在文献[18]中,作者根据探测得到的固态硬盘内部并行特性修改操作系统的IO调度提交策略,为上层应用提供一种更好的并行处理方式.然后在此基础之上提出新的B^+树索引——PIO B树,同时提出针对PIO B树的更新策略和日志恢复策略.文献[19]采用分块排序策略来提高索引扫描的性能.

3 固态硬盘内部并行特性探测

固态硬盘内部的架构细节作为生产商的一种核心知识产权,通常是不会公布的.因此,要了解固态硬盘内部并行特性的最直接最有效的方式就是探测.

3.1 探测模型

文献[16]基于公开的文档定义了一个通用模型来抽象固态硬盘的组织结构.其中,一个"域"代表共享一组特定资源(如通道)的闪存芯片集合,一个"块"是一个域中连续分配的数据单元,依照映射策略,不同块被交错地放置在了N个域中.本文测试与固态硬盘内部并行相关的两个关键参数:块大小和域个数.

把固态硬盘看做一个黑盒,通过注入精心设计的I/O访问模式,可以观察到固态硬盘的一些反应,并记录一些关键性能指标,例如延迟和带宽.利用这些探测信息就可以推测出固态硬盘的一些内部细节.我们试图以一种简单但却有效的方式来了解固态硬盘的关键架构特征而非固态硬盘内部的所有细节.固态硬盘块大小和域个数探测伪代码如算法1和2所示.

算法1. 固态硬盘块大小探测伪代码.

```
Max: 最大可能块大小
init_SSD();          //初始化固态硬盘(顺序写满硬盘)
for (i=512B; I<=Max; i+=512B) //I/O请求大小
  for (j=0; j<2*Max; j+=512B) //I/O请求偏移
    for(k=0, latency=0; k<1000; k++)
      latency+=read_data();    //读取数据并计时
    latency=latency/1000;      //计算请求响应延迟
```

算法2. 固态硬盘域个数探测伪代码.

```
Dom: 最大可能域个数
stride_read(n): 两个并发线程,一个线程从偏移为0
处读一块;另一个线程从偏移n块后的位置读一块
init_SSD();                          //初始化固态硬盘
for (j=1; j<=4*Dom; j++)             //跳读距离
  bw=stride_read(j); //跳读两块数据并返回I/O带宽
```

测试时需要注意的事项:(1)主板BIOS需开启AHCI[①](Serial ATA Advanced Host Controller Interface,串行ATA高级主控接口/高级主机控制器接口)模式,以支持固态硬盘接口逻辑中最新融入的NCQ[②]技术,该技术对挖掘利用固态硬盘内部并行特性至关重要;(2)固态硬盘文件访问需设置为DirectIO模式以避免操作系统文件缓冲区的影响.

3.2 探测结果

本文选择了3款具有代表性的固态硬盘作为实验对象,规格参数详见表1.其中两款建立在多层单元(MLC)闪存芯片的基础上,主要针对大容量市场;另一款则是针对高端市场,它建立在速度更快、使用寿命更长的单层单元(SLC)闪存芯片的基础上.出于商业保密的原因,我们暂且将这3款SSD分别命名为SSD-1,SSD-2和SSD-3.特别说明一点就是:这3款SSD均支持NCQ技术.它们的设计与主流的SSD设计一致,是市场上主流技术趋势的典型代表.

表1 待测固态硬盘规格参数

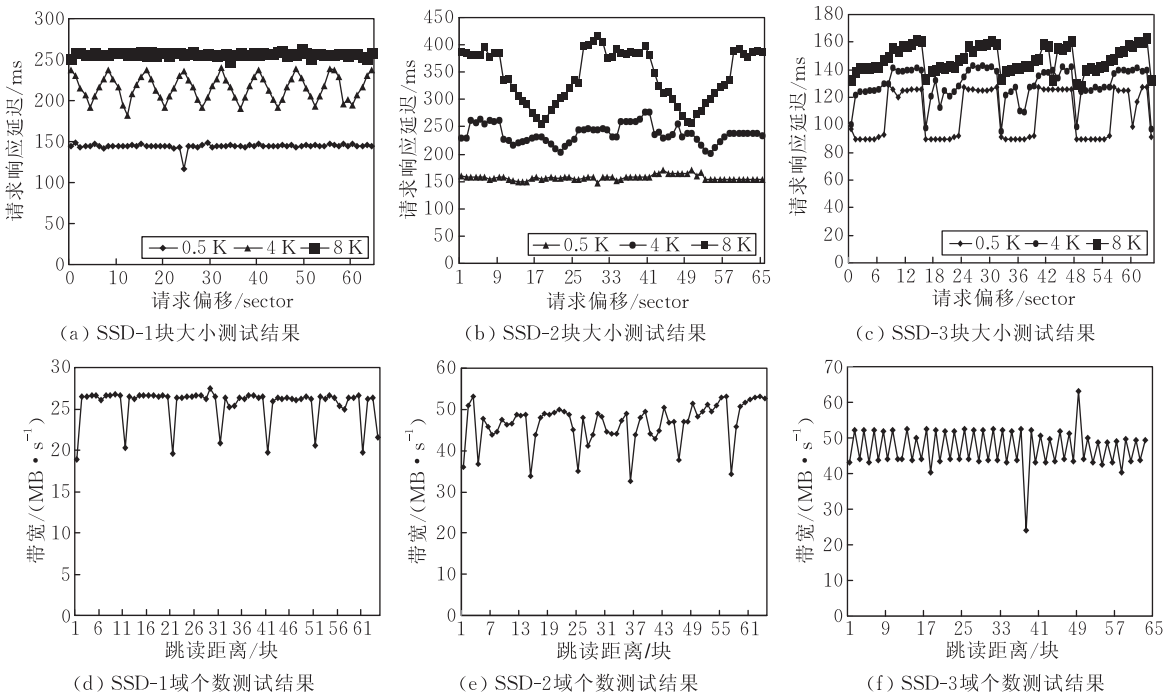
	生产厂家	芯片类型	容量/GB	接口类型	NCQ
SSD-1	Intel	SLC	32	SATA2	32
SSD-2	Intel	MLC	160	SATA2	32
SSD-3	OCZ	MLC	60	SATA2	32

首先,让我们分析一下SSD块大小测试结果.图3(a)~(c)显示了3款SSD的请求响应延迟随请求偏移(单位是sector,1sector=0.5K)位置的变化而变化的情况.图中3条线分别展示了请求大小为0.5K、4K和8K 3种情况.类似于文献[16],图3(a)和(b)中请求大小为0.5K的时候,请求偏移位置的变化不会影响请求响应延迟的变化,趋近于平线,说明每一次请求都落在一个块内并且不会跨多个块,同时也说明每一块的读写时间是相同的.对于SSD-1来说,当请求大小为4K的时候,请求响应延迟随着请求偏移位置的增加呈现出周期性的上下波动,且没有出现平稳状态,一个周期的大小即4K.当请求大小为8K的时候,请求响应延迟近似为一恒定值,分析其原因有两个:固态硬盘的内部并行特性和8K已经超过一个周期的大小4K.对于SSD-2来说,请求大小为4K和8K的时候,请求响应延迟随着请求偏移位置的增加成周期性的上下波动变化,周期大小为固定的16K,请求大小4K和8K小

① http://en.wikipedia.org/wiki/Advanced_Host_Controller_Interface

② http://zh.wikipedia.org/wiki/NCQ

于 16 K，所以在图 3(b)中与 SSD-1 不同的是在一个周期内会出现相对稳定状态. 文献[16]指出一个周期的跨度就是块大小. 可见 SSD-1 和 SSD-2 的块大小分别为 4 K 和 16 K. 在图 3(c)中，无论请求大小是 0.5 K，还是 4 K，还是 8 K，请求响应延迟都随着请求偏移位置的增加呈现出周期性的变化，周期大小恒定为 8 K. 根据文献[16]可知 SSD-3 的块大小为 8 K. 但是请求大小为 0.5 K 的时候，随着请求偏移的大小一个周期内出现快慢两个阶段，这就说明 SSD-3 具有快慢块的现象，即不是所有块的读写速度都一样. 这就使得请求大小为 4 K 和 8 K 的时候，请求响应延迟也具有两个阶段，即快慢两个阶段. 综上可知，SSD-1、SSD-2 和 SSD-3 的块大小分别是 4 K、16 K 和 8 K.



(a) SSD-1块大小测试结果　(b) SSD-2块大小测试结果　(c) SSD-3块大小测试结果

(d) SSD-1域个数测试结果　(e) SSD-2域个数测试结果　(f) SSD-3域个数测试结果

图 3　固态硬盘内部并行测试

然后，我们再来分析域个数测试结果. 图 3(d)～(f)显示了 3 款 SSD 带宽随跳读距离(单位是块)大小的变化而变化的情况. 图 3(d)～(f)中每条线的请求大小分别为 4 K、16 K 和 8 K，即上面测试得到的块大小. 通过图 3(d)～(f)可以观察到带宽大小随着跳读距离大小的增长呈现出周期性的变化. 对于 SSD-1 和 SSD-2，通过测试结果可知 SSD-1 和 SSD-2 都具有两层的并行结构，即“域”和“子域[16]”，分别对应着图 3(d)～(e)的大周期和小周期，大周期为 20，小周期为 10，所以域的个数为 20. 由于 SSD-3 具有快慢块的现象，所以 SSD-3 的测试结果显示出了非平稳现象，即一直处于上下波动状态，但是可以观察到带宽较小的点(跳读距离为 21、41 和 61)之间也呈现了周期为 20 的规律，所以 SSD-3 的域个数也为 20.

通过上面的测试可知：对于某些厂家(Intel 和 OCZ)的产品，虽然他们的制作工艺上会有很大区别(芯片类型等)，但是通过测试可知，这些固态硬盘具有很多相似的特性，其中最主要的就是内部并行特性. 内部并行特性主要有 3 个指标：块大小、域个数和映射方式(本文给出了块大小和域个数的测试结果). 对于不同种类的产品，它们的块大小和域个数可能不同. 有些产品可能还具有其它特性，比如 OCZ 产品的快慢块现象. 可见在使用固态硬盘之前，通过探测了解固态硬盘的特性是首要的也是最重要的，了解固态硬盘之后可以针对其特性进行合理配置和优化.

4　并行表扫描 ParaSSDScan

首先，基于固态硬盘内部并行特性优化数据存储. 然后，提出利用固态硬盘内部并行特性的并行表扫描 ParaSSDScan.

4.1　数据存储模型

传统关系数据库多采用行存储模式[7]进行数据存储，本文数据块内数据仍然采用行存储模式. 数据

块对应传统关系数据库的数据页，块内数据组织结构类似于传统关系数据库的数据页的逻辑结构.但是，与传统关系数据库的数据页有一点不同，即数据块内需要维护一个指针，用以指向同一域内的同一关系表的下一数据块.通过指针可以快速查找到同一关系表在同一域内的下一数据块.

数据块顺序的导入到固态硬盘，数据在固态硬盘上的分布状况类似于磁盘阵列 RAID-0[①]，即条带化存储模式，如图 4 所示，此时数据平均分配到不同的域中，性能可以达到最优.如果数据分布不均，单次操作的性能就完全取决于数据块最多的域内数据大小.本文主要关注固态硬盘内部并行特性对查询的影响，对于更新的影响将作为下一步工作进行研究.

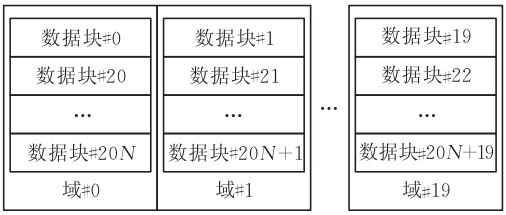


图 4 数据存储模型

4.2 ParaSSDScan 概述

根据 3.1 节提出的模型可知，域是固态硬盘的并行单元，即域之间可以并行访问，每个域都是由若干个数据块组成.表扫描操作主要是.5 描一张表的所有数据块.在传统数据库中，表扫描采用的是一块接着一块地把数据页从磁盘读出，但是受限于磁盘的磁头不能并行操作磁盘，而固态硬盘没有这种机械组件，并且固态硬盘具有丰富的内部并行特性，所以提出基于固态硬盘内部并行特性的并行表扫描 ParaSSDScan 来提高扫描的效率.

如图 5 所示，ParaSSDScan 中最基本的操作单元是“单域扫描”.单域扫描把数据从单个域中一块接着一块地扫描放到缓冲区中.为减少资源共享带来的性能损失，让每个单域扫描都有独享的缓冲区，把一个单域扫描对应的缓冲区叫做“缓冲子区”.这样，多个单域扫描可以无干扰地并行执行，因为每个单域扫描对应一个属于自己的缓冲子区.由多个单域扫描和多个缓冲子区构成“多域并行扫描”.域是固态硬盘上的并行单元，故表数据存储时要考虑：为尽可能发挥固态硬盘内部并行特性，单个表所有数据块应尽可能地分散到不同的域内.

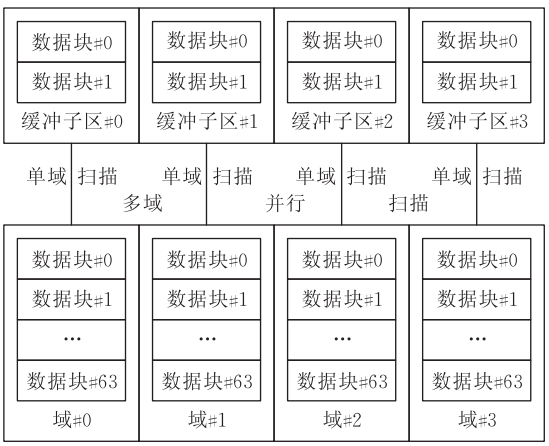


图 5 ParaSSDScan 结构图

4.3 单域扫描和多域并行扫描

单域扫描与传统数据库中的表扫描类似.传统数据库中的表扫描把指定表从数据库中一块接着一块地读出到缓冲区进行处理.单域扫描与传统表扫描的不同点在于读取数据不同，单域扫描是从一个域中把数据一块接着一块读取到指定的缓冲子区中.每个单域扫描都有一个缓冲子区与其对应.缓冲子区用以缓冲单域扫描读取的数据.由于内存容量有限，一个域内存储的被扫描表的数据大小很可能超过该单域扫描对应的缓冲子区大小，那么在处理过程中，为了读取未被处理的数据，缓冲子区中的被处理过的内容需要被替换掉.因为我们这里不考虑数据的写操作，所以缓冲子区的管理就相对简单很多——只有换入没有换出，即读取未处理的数据块覆盖缓冲子区中的已被处理数据块.

多域并行扫描，根据域个数和处理器支持最大线程数产生若干线程，每个线程对应一个单域扫描，然后让这些线程无干扰地并行执行.整个缓冲区根据线程个数被平均化分为若干个缓冲子区，每个缓冲子区采用相同的管理方式——只有换入没有换出.多域并行扫描的性能取决于其并行度，并行度不但与域的个数有关，还与处理器物理线程数和固态硬盘支持的 NCQ 有关.

5 并行聚集 ParaSSDAggr

聚集操作被查询计划解析器转化为物理操作表扫描，表扫描性能对聚集操作性能有直接影响.本文对此提出基于并行表扫描的并行聚集模型 ParaSSDAggr，并实现几种简单的聚集操作.

5.1 并行聚集模型 ParaSSDAggr

基于传统表扫描的聚集操作主要过程如下：扫描指定表的数据块并一块接着一块地读取到缓冲区

① http://en.wikipedia.org/wiki/RAID

中，然后进行聚集计算. 如果内存不够大，不能存储指定表的所有数据块，那么只能一次读取一部分数据块到缓冲区中，进行聚集计算，然后再读取一部分，再进行聚集计算，依此重复执行直到指定表的数据块全部读取到缓冲区并进行聚集计算.

对于聚集操作模型，则需要针对 ParaSSDScan 进行适当的改进. 如图 6 所示，改进后的并行聚集模型 ParaSSDAggr 把聚集操作分成 3 个阶段：单域扫描从域中读取数据写入到相应缓冲子区中，“单聚集”从缓冲子区中读取数据进行单聚集计算并把计算结果写入到“单聚集结果”中，“总聚集”从单聚集结果中读取数据并进行总聚集计算并把结果写入到“总聚集结果”中.

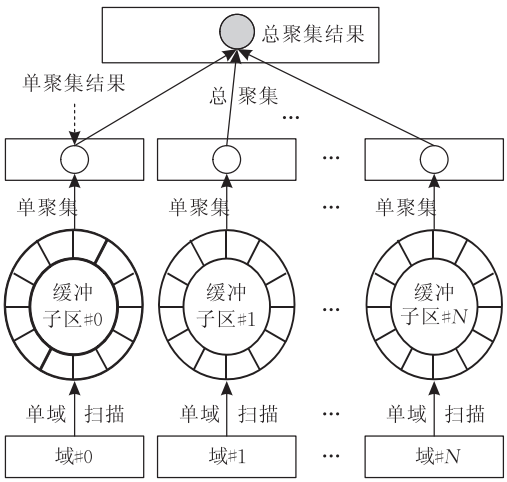


图 6　并行聚集模型 ParaSSDAggr

单域扫描和单聚集都需要操作缓冲子区，缓冲子区不能使用简单的覆盖策略. 缓冲子区的所有页采用环状组织形式. 缓冲子区有两个指针，分别是读指针和写指针. 单域扫描使用写指针确定向缓冲子区中写数据的位置. 单聚集使用读指针确定从缓冲子区中读数据的位置，读取数据之后进行单聚集计算(由具体聚集操作决定)，并把计算结果写入到单聚集结果中. 单聚集与传统的聚集很相近，不同点在于读取数据的方式不同. 因为固态硬盘域间的并行性，单域扫描可以并行执行；不同域对应缓冲子区之间没有数据交互，不同单聚集之间也没有互相影响，所以单聚集操作也可以并行执行.

总聚集，即对多个单聚集结果进行总聚集计算(由具体聚集操作决定)，产生总聚集结果的过程. 获得单聚集结果的方式有两种：“单聚集上推”和“总聚集下拉”. 单聚集上推，指的就是单聚集对串行组内所有数据进行完聚集计算之后产生单聚集结果，主动传递给总聚集使用. 总聚集下拉，指的是总聚集向单聚集操作索取所需的单聚集结果. 并行聚集 ParaSSDAggr 的性能取决于并行扫描处理能力和处理器物理线程数，而并行扫描与域个数、处理器物理线程数和固态硬盘支持的 NCQ 都有关. 综上分析可知，并行聚集模型 ParaSSDAggr 的性能与域个数、处理器物理线程数和固态硬盘支持的 NCQ 有关.

5.2　聚集操作实现

聚集操作中比较典型且常用的有求和、求最大、求最小、计数和求平均值. 本部分详细描述这几种常用聚集操作的实现. 根据单聚集和总聚集的聚集计算是否相同以及单聚集产生结果的个数可以分为 3 类：

(1) 求和、求最大和求最小. 对于该类聚集操作，聚集模型 ParaSSDAggr 中的单聚集计算过程和总聚集计算过程采用相同的聚集计算，并且单聚集结果均为单个数据. 首先利用单域扫描读取指定域中数据到缓冲子区中，然后使用单聚集对读入到缓冲子区中的数据进行求和、求最大或者求最小，然后把计算结果写入到单聚集结果中. 当单聚集计算完单个域内的聚集结果之后，把单聚集结果传递给总聚集，总聚集对所有单聚集结果进行求和、求最大或求最小，把结果作为总聚集结果反馈给上层用户.

(2) 计数. 对于该聚集操作，单聚集结果也是单个数据，不过聚集模型 ParaSSDAggr 中的单聚集计算和总聚集计算采用不同的聚集计算. 单聚集从指定缓冲子区中读取数据并进行计数，最后获得单聚集计数结果并存入单聚集结果. 然后，把单聚集结果传递到总聚集中，总聚集对所有单聚集结果进行“求和”计算获得总聚集结果，也就是最终的计数结果，反馈给用户.

(3) 求平均值. 对于该聚集操作，聚集模型 ParaSSDAggr 中的单聚集和总聚集采用不同的聚集计算，且单聚集结果为两个数据. 单聚集读取指定缓冲子区数据并进行求和和计数两种聚集计算，即单聚集结果为两个值(分别称为“单聚集和”和“单聚集计数”). 单聚集计算完成后，把单聚集结果传递给总聚集，总聚集分别对“单聚集和”和“单聚集计数”进行求和，结果分别记为“总聚集和”和“总聚集计数”，那么总聚集结果就为“总聚集和”和“总聚集计数”相除，也就是最终的平均值.

6　实验结果及分析

6.1　实验环境

实验平台为 Lenovo 昭阳 K46A，其处理器为

Intel 酷睿 i5 450MHz，双核心四线程，内存大小 2GB，硬盘为一块 5400RPM 500GB SATA 接口西部数据磁盘. 操作系统使用的是 Fedora14，内核版本为 Linux 2.6.35. 待测 3 款固态硬盘参数详见表 1. 为了能够充分利用固态硬盘的内部并行特性，还需要对系统进行一些额外的配置. 在 BIOS 下开启 AHCI 模式，当其开启后固体硬盘支持的 NCQ 才能发挥效用.

ParaSSDScan(物理操作：从固态硬盘读取数据到内存)和 ParaSSDAggr(基于 ParaSSDScan 的上层聚集操作：针对主键的求和、求最大、求最小、计数和求平均 5 种聚集操作)采用标准 C 语言编写，为了尽可能体现算法性能，避免文件系统缓冲区的干扰，打开文件时要以 DirectIO 方式打开，同时还需要设置对齐访问方式，用以保证数据对齐访问.

本文主要测试物理操作表扫描和聚集操作在磁盘和固态硬盘上的性能表现，采用 4.1 节的方式顺序地存储数据到磁盘和固态硬盘之上. ParaSSDScan 和 ParaSSDAggr 的性能可以完全由数据大小验证，而无需考虑数据类型，故本实验数据集采用 TPC-C 标准数据集 Customer 表的数据(87 万条记录，1G 数据). 性能衡量指标采用完成扫描或者聚集操作的运行时间，即请求响应延迟(单位：ms). 测试平台 CPU 具有 4 线程，所以测试 4 线程下的性能. 由 3.2 节中的测试结果可知固态硬盘 SSD-1、SSD-2 和 SSD-3 都有 20 个域，所以测试 20 个线程下的性能. 另外，测试 10 个线程情况下的性能用于对比分析.

为方便解释，定义符号 ParaSSDScan-N 和 ParaSSDAggr-*N* 分别代表使用 *N* 线程运行 ParaSSDScan和ParaSSDAggr. 传统表扫描 TraScan 的主要过程是一块数据接着一块数据从外存介质读取数据，采用的是单线程的处理方式，即 TraScan 等同于 ParaSSDScan-1. 传统的聚集操作 TraAggr 主要是基于传统表扫描 TraScan 实现的，故 TraAggr 等同于 ParaSSDAggr-1.

6.2 ParaSSDScan 的性能分析

表扫描性能测试结果如图 7 所示. 我们测试了 TraScan(ParaSSDScan-1)以及 ParaSSDScan-4、ParaSSDScan-10 和 ParaSSDScan-20 在磁盘和固态硬盘上的运行性能.

通过图 7 可知，在磁盘上运行 TraScan 和 ParaSSDScan-4/10/20，ParaSSDScan-4/10/20 并不一定比传统方法具有优势，例如 ParaSSDScan-20 运行时间比 TraScan 还要长. 图 7 还指出 TraScan 在

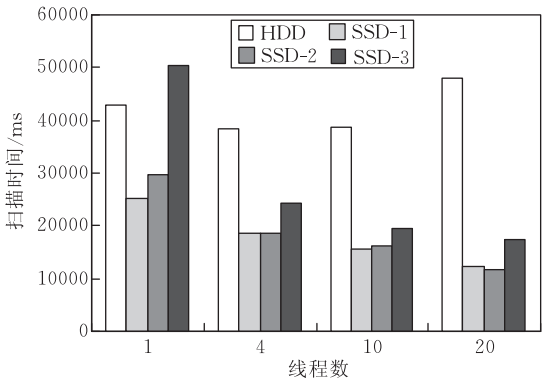


图 7 表扫描性能随线程数的变化

磁盘上的运行时间要比 ParaSSDScan-4/10/20 在 3 款固态硬盘上运行时间长很多，基本前者是后者的 2～3 倍，所以 ParaSSDScan-4/10/20 要比 TraScan 具有更好的时间性能. ParaSSDScan 在固态硬盘上的运行时间在随着线程数的增大而减小. 虽然 CPU 物理线程数只有 4 个，但是固态硬盘可以支持更多的并行操作，所以当并行线程数为 10 和 20 的时候，ParaSSDScan 的运行时间仍然减小，但是减小的程度已经有限，这主要是由于大量线程之间的切换开销的影响. ParaSSDScan-20 在固态硬盘上的运行时间大概是 TraScan 在磁盘上运行时间的三分之一.

6.3 ParaSSDAggr 的性能分析

本文实现了针对主键的聚集操作，即对主键进行求和、求平均、求最大/最小和计数. 因为该 5 种聚集操作相对都很简单，本文将它们集中实现在一起作为一种集中聚集操作处理，类似于执行了形如“Select count(*), sum(ID), avg(ID), max(ID), min(ID) from Customer;”的 SQL 语句. 对于每一个缓冲子区，默认设置其大小为 16 KB. 图 8 所示运行时间是 5 种聚集操作的总运行时间. 本文测试了 TraAggr(ParaSSDAggr-1)以及 ParaSSDAggr-4、ParaSSDAggr-10 和 ParaSSDAggr-20 在磁盘和固态硬盘上的运行性能.

如图 8 所示，ParaSSDAggr-4/10/20 在磁盘上运行的时间都比 TraAggr 在磁盘上的运行时间要长，所以 ParaSSDAggr-4/10/20 不适合磁盘数据库系统，主要是磁盘硬件不具有并行特性. 无论在哪款固态硬盘上运行 ParaSSDAggr-4/10/20 都要比在磁盘上运行 TraAggr 节省很多时间. 同时，ParaSSDAggr-4/10/20 在固态硬盘上的运行时间也是随着线程数的增大而下降，但是下降程度在减少，这点类似于 ParaSSDScan-4/10/20，主要是因为 ParaSSDAggr 的底层物理操作采用 ParaSSDScan.

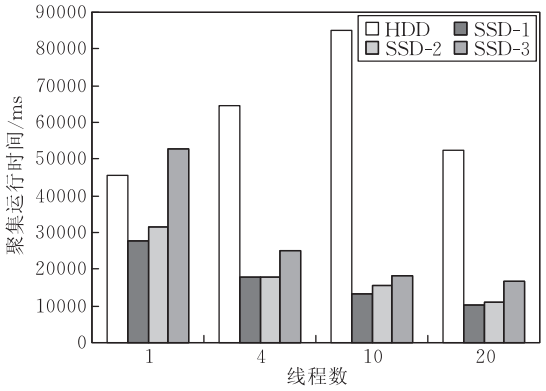


图 8　聚集随线程数性能变化

ParaSSDAggr-20 在固态硬盘上的运行时间大概是 TraAggr 在磁盘上运行时间的四分之一.

缓冲子区在上面的测试中默认设置为 16 KB，但是同时也希望通过增加内存提高操作性能.但是内存有限且成本较高，故本文测试聚集操作性能和缓冲区大小之间的关系.聚集操作性能随缓冲子区大小变化的性能如图 9 所示.图中 SSD-1(4)中的 4 表示 ParaSSDAggr 并行运行的线程数，其它类同.由图 9 可见，缓冲子区增大对聚集操作的运行时间并没有太大影响.由此可见，ParaSSDAggr 不但节省了时间还节省了内存.

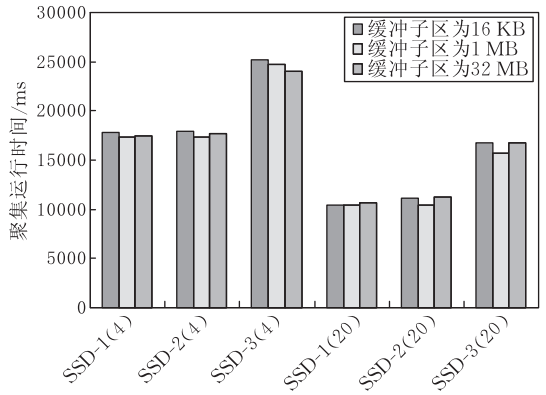


图 9　ParaSSDAggr 随缓冲子区大小变化

7　结　论

随着固态硬盘的应用越来越广泛，使得固态硬盘受到学术界和工业界越来越多的关注.由于采用类似磁盘冗余阵列的组织结构，使得并行特性成为固态硬盘固有的特性.本文探测了 3 款不同厂家型号的固态硬盘的内部并行特性，同时还发现了某些固态硬盘具有快慢块的现象.

针对固态硬盘内部并行特性提出了优化的并行表扫描和并行聚集操作模型.通过实验证明并行表扫描模型 ParaSSDScan 和聚集操作模型 ParaSSDAggr 较之传统数据库表扫描和聚集操作的性能分别提高了 3 倍和 4 倍.固态硬盘内部并行特性还可用于优化数据库的其它模块，比如缓冲区管理、索引、事务处理、查询处理和查询优化等等.

参　考　文　献

[1] Yoon Jin Hyuk, Nam Eyee Hyun, Seong Yoon Jae, Kim Hongseok, Kim Bryan S, Min Sang Lyul, Cho Yookun. Chameleon: A high performance flash/FRAM hybrid solid state disk architecture. Computer Architecture Letters (CAL), 2007, 7(1): 17-20

[2] Debnath Biplob, Sengupta Sudipta, Li Jin. Flashstore: High throught persistent key-value store. Proceedings of the Very Large Data Base (VLDB) Endowment, 2010, 3(2): 1414-1425

[3] Debnath Biplob, Sengupta Sudipta, Li Jin. Chunkstash: Speeding up inline storage deduplication using flash memory// Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'10). Boston, USA, 2010: 16-16

[4] Debnath Biplob, Sengupta Sudipta, Li Jin. Skimpystash: Ram space skimpy key-value store on flash-based storage// Proceedings of the 2011 International Conference on Management of Data (SIGMOD'11). Athens, Greece, 2011: 25-36

[5] Koltsidas Ioannis, Viglas Stratis D. Flashing up the storage layer//Proceedings of the 34th International Conference on Very Large Data Base (VLDB'08). Auckland, New Zealand, 2008: 24-30

[6] Agrawal Nitin, Prabhakaran Vijayan, Wobber Ted, Davis John D, Manasse Mark, Panigrahy Rina. Design tradeoffs for SSD performance//Proceedings of the 2008 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'08). California, USA, 2008: 57-70

[7] Ramakrishnan Raghu, Gehrke Johannes. Database Management Systems. 3rd Edition. New York: McGraw Hill, 2002

[8] Park Seon Yeong, Seo Euiseong, Shin Ji Yong, Maeng Seungryoul, Lee Joonwon. Exploiting internal parallelism of flash-based SSDs. Computer Architecture Letters (CAL), 2010, 9(1): 9-12

[9] Im Soojun, Shin Dongkun. Flash-aware RAID techniques for dependable and high-performance flash memory SSD. IEEE Transactions on Computers, 2011, 60(1): 80-92

[10] Ma Dongzhe, Feng Jianhua, Li Guoliang. LazyFTL: A page-level flash translation layer optimized for NAND flash memory//Proceedings of the 2011 International Conference on Management of Data (SIGMOD'11). Athens, Greece, 2011: 1-12

[11] Kim Jae Hong, Jung Dawoon, Kim Jin Soo, Huh Jaehyuk. A methodology for extracting performance parameters in solid state disks (SSDs)//Proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS'09). London, UK, 2009: 1-10

[12] Greenan Kevin M, Long Darrell D E, Miller Ethan L, Schwarz Thomas J E, AvaniWildani S J. Building flexible, fault-tolerant flash-based storage systems//Proceedings of the 5th Workshop on Hot Topics in System Dependability (HotDep'09). Lisbon, Portugal, 2009: 1-6

[13] Lee Sang-Won, Moon Bongki, Park Chanik. Advances in flash memory SSD technology for enterprise database applications//Proceedings of the 2009 International Conference on Management of Data (SIGMOD'09). Providence, Rhode Island, USA, 2009: 863-870

[14] Seol Jinho, Shim Hyotaek, Kim Jaegeuk, Maeng Seungryoul. A buffer replacement algorithm exploiting multi-chip parallelism in solid state disks//Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'09). Grenoble, France, 2009: 137-146

[15] Chen Feng, Koufaty David A, Zhang Xiaodong. Understanding intrinsic characteristics and system implications of flash memory based solid state drives//Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'09). New York, NY, USA, 2009: 181-192

[16] Chen Feng, Lee Rubao, Zhang Xiaodong. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing//Proceedings of the 17th International Symposium on High Performance Computer Architecture (HPCA'11). San Antonio, Texas, USA, 2011: 266-277

[17] Hu Yang, Jiang Hong, Feng Dan, Tian Lei, Luo Hao, Zhang Shuping. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity//Proceedings of the International Conference on Supercomputing (ICS'11). Tucson, Arizona, USA, 2011: 96-107

[18] Roh Hongchan, Park Sanghyun, Kim Sungho, Shin Mincheol, Lee Sang-Won. B^+-tree index optimization by exploiting internal parallelism of flash-based solid state drives. Proceedings of the Very Large Data Base (VLDB) Endowment, 2012, 5(4): 286-297

[19] Lee Eun Mi, Lee Sang Won, Park Sangwon. Optimizing index scans on flash memory SSDs. SIGMOD Record, 2011, 40(4): 5-10

FAN Yu-Lei, born in 1984, Ph. D. candidate. His current interests include storage management, query processing and optimization, index and recovery of Flash-based database systems.

LAI Wen-Yu, born in 1989, M. S. candidate. His current interests include storage management and query processing and optimization of flash-based database systems

MENG Xiao-Feng, born in 1964, Ph. D., professor, Ph. D. supervisor. His research interests include Web data management, native XML databases, mobile data management, etc.

Background

Due to its superiority such as low access latency, low energy consumption, light weight, and shock resistance, flash memory as a storage alternative for mobile computing devices has been steadily expanded into personal computer and enterprise server markets With the extensive application of flash-based SSDs in more and more devices, SSDs have attracted more and more attention by the academia and industry. In addition to the excellent characteristics of flash memory, there is wealth internal parallelism in SSDs. However, since flash memory exhibits better performance than that of Hard disk, existing database systems may not be able to take full advantage of flash memory without elaborate flash-aware data structures and algorithms.

We consider query processing for flash based database systems. Now there are a few of works on this issue, and the works has been published is either using a special storage structure, PAX, or exploiting low access latency of flash memory or optimizing some algorithms which are not take full advantage of internal parallelism of SSDs

Our research group focuses on the design and implementation for flash-based database systems. We have published several papers about storage management, buffer management, transaction processing, logging, index and query processing for flash-based databases on internal and external conferences. And this work is the first one about query processing and internal parallelism in our group. This paper is used for improve the full table scan and aggregation operations performance for flash-based databases.

In this paper, we detect internal parallelism of SSDs seemed as a black box. Based on the internal parallelism of SSDs, we propose a new full table scan method, ParaSSDScan. Based on ParaSSDScan, we also propose an efficient parallel aggregation model, ParaSSDAggr, which can also take full advantage of the internal parallelism of SSDs. And then we achieve several common aggregation operations with ParaSSDAggr. Experimental results show that, compared to traditional full table scan and aggregation operations, there are 3x and 4x improvement of the performance for ParaSSDScan and ParaSSDAggr which cost little memory. ParaSSDScan and ParaSSDAggr largely speed up full table scan and aggregation operations.

This research was partially supported by the grants from the National Natural Science Foundation of China under Grant Nos. 60833005, 61070055.