武汉大学 计算机学院

2017级硕士研究生课程

**采用智能SSD的高性能数据挖掘**

学 院： 计算机学院

学 号： 2017282110273

姓 名： 刁文秀

摘 要

有效的大数据处理变得愈加重要，但是由于现有的硬盘驱动器的高I / O带宽和低访问延迟限制，固态硬盘（SSD）已经迅速取代硬盘驱动器（HDD）。由于采用多路多通道交错技术，固态硬盘无需使用机械设备，因此可提供较高的内部I / O带宽。但是在SSD环境下，主机接口的I / O带宽成为数据密集型应用的新瓶颈。为了解决主机接口的性能瓶颈问题，我们提出了一种称为智能SSD（iSSD）的方法。本文主要介绍智能SSD的概念及其设计，通过在数据挖掘应用的各种设置下评估它们的潜在优势，并实际进行在iSSD模拟器上的开发，通过iSSD模拟器展示iSSD在数据挖掘中的潜力。我们的iSSD模拟器在gem 5模拟器上执行，完全模拟在iSSD中循环运行的数据挖掘算法的所有过程级别的准确性，然后进一步讨论如何利用所有的计算资源在新的计算环境下来有效地处理数据挖掘算法。在由CPU，GPU和ISSD组成的计算环境中异构调度过程中，考虑计算资源的类型，计算资源中的存储器大小，以及包括SSD中IO时间的处理器间通信时间。我们的调度结果表明，协作环境中的处理性能比传统处理性能高10倍左右。

关键词：智能SSD；数据挖掘；异构调度；协作处理

**ABSTRACT**

Effective big data processing has become increasingly important, but solid state drives (SSDs) have rapidly replaced hard disk drives (HDDs) due to the high I / O bandwidth and low access latency limitations of existing hard drives. With multi-channel multi-channel interleaving, SSDs eliminate the need for mechanical equipment and therefore provide higher internal I / O bandwidth. However, in the SSD environment, I / O bandwidth of the host interface becomes a new bottleneck for data-intensive applications. In order to solve the performance bottleneck of the host interface, we propose a method called Smart SSD (iSSD). This article describes the concept and design of smart SSDs by demonstrating their potential benefits under various settings for data mining applications and actually developing them on the iSSD simulator to demonstrate the potential of iSSD in data mining through the iSSD simulator . Our iSSD simulator, executed on the gem 5 simulator, fully mimics the accuracy of all process levels of the data mining algorithms running in the iSSD, and then discusses how to utilize all of the computing resources to efficiently process in the new computing environment Data mining algorithm. In heterogeneous scheduling in a computing environment comprised of CPUs, GPUs, and ISSDs, consider the type of computing resources, the size of memory in computing resources, and the interprocessor communication time including IO time in SSDs. Our scheduling results show that the processing performance in the collaborative environment is about 10 times higher than the traditional processing performance.

**Key words**: intelligent SSD; data mining; heterogeneous scheduling; collaborative processing

目 录

[1 引言 1](#_Toc24218)

[2 相关工作 2](#_Toc8953)

[2.1智能SSD 2](#_Toc18493)

[2.2 将数据挖掘算法建模为同步数据流 3](#_Toc29116)

[3 评估 6](#_Toc6154)

[3.1 iSSD模拟器的实现 6](#_Toc12460)

[3.2数据挖掘算法的实现 7](#_Toc10325)

[3.3 评估模拟器 7](#_Toc4126)

[4 协作环境下的数据处理 8](#_Toc29219)

[5 实验过程 9](#_Toc12006)

[5.1 实验设置 9](#_Toc854)

[5.2 确定粒度 11](#_Toc3152)

[5.3 实验结果 11](#_Toc11055)

[5.4 更改参数 13](#_Toc31513)

[6 结论 14](#_Toc14327)

[7 思考 14](#_Toc9466)

# 

# 1 引言

最近，大数据分析对于从大量数据中提取相关模式愈加重要。由于硬盘驱动器高I / O带宽和低访问延迟的限制，它迅速的被固态硬盘所取代。

在HDD环境下，由于HDD通过物理旋转磁盘来读取数据，所以HDD的I / O带宽低是数据密集型应用中的主要性能瓶颈。 然而由于固态硬盘采用多路多通道交错技术，无需使用机械设备，因此可提供较高的内部I / O带宽。但是在SSD环境下，主机接口的I / O带宽却成为数据密集型应用的新瓶颈。

为了解决主机接口的性能瓶颈问题，我们提出了一种称为智能SSD（iSSD）的方法，其中大容量数据由SSD内的处理单元直接处理，只有少量的结果数据被传输到主机。采用这种方法，使得主机接口不再是数据密集型应用中的性能瓶颈。

SSD中的内部I / O带宽由于可以通过增加多路通道来轻松扩展。因此在数据处理中可以充分利用内部I / O带宽的增加使能够在数据密集型应用程序中获得显著的性能提升。

数据挖掘是数据密集型应用的典型例子，具有以下特点：（1）在执行过程中频繁地访问存储的数据; （2）对访问的数据进行扫描和过滤等，反复进行比较简单的操作。在本文中，我们将介绍设计注意事项以及iSSD在数据挖掘应用方面的潜在优势并实际进行在iSSD模拟器上的开发。

目前，CPU，GPU和SSD在大多数计算环境中都配备在一起。如果以后SSD替换为iSSD，我们将面临一个新的有效协作计算环境。在本文的研究中，我们考虑计算资源的协作，特别是iSSD，图形处理单元（GPU）和CPU之间的协作。

GPU是处理与图形相关的应用程序的计算资源。最近有一些方法是利用GPU来处理一般应用程序，而不是与图形相关的应用程序。GPU由多个大量内核组成，每个内核之间并行处理同一组指令。一些现有的研究已经通过使用单个计算资源来解决应用数据，而其他一些方法也有利用两种不同资源（例如，CPU和GPU或CPU和iSSD）的协作来进行更有效的处理数据。我们应该解决如何利用这三种计算资源来高效处理数据挖掘算法，毕竟每个计算资源都具有各自的特点。

协作环境是一个由不同类型的计算资源提供不同的性能所组成的异构环境，每个计算资源都有自己独立的内存，所以需要处理器间通信（IPC）时间。在这种环境下，需要进行调度来决定哪个内核要运行哪个函数，以最佳方式处理目标算法，所以我们需要为协作环境提供异构调度。具体调度时间表可以通过调度目标算法来获得，它通常被建模为一个图。这种建模需要考虑计算资源的类型，计算资源的内存大小以及计算资源之间的IPC时间。在文章的协作环境中，对一个算法的图形建模是这样进行的：首先，我们将函数分为两类，可并行化函数和合并函数。GPU和iSSD具有适合于并行处理数据的架构，因此在分配给可并行化功能时是有利的。另一方面，CPU分配给需要合并的功能或时间复杂度比较高的应用。其次通过函数确定一次要处理的数据量。例如，GPU通过并行使用大量内核来同时处理大量数据。在GPU中设置要与核心数量匹配的数据量更为有效。本文提出了一种基于上述策略的三种计算资源处理数据挖掘算法的有效方法。

本文的结构如下：第2节介绍智能SSD的设计以及将数据挖掘算法建模为同步数据流的内容。第3节主要介绍了基于iSSD模拟器的实现以及算法在其上的实现和评估结果。第4节介绍了所提出的协作环境，并考察了协同处理的异构调度策略。第5节通过各种实验展示了协作处理的有效性。第6节总结了文章的主要工作。第7节对该篇文章的不足提出部分思考以做进一步的研究。

# 2 相关工作

## 2.1智能SSD

图2.1显示了iSSD的整体架构。iSSD是具有配备通用内核（称为FMC内核）和DRAM的闪存控制器（FMC），它们用于在每个通道中进行数据处理，并在现有SSD中采用更强大的SSD内核。但是这些内核在iSSD中的性能相对于主机CPU来说相对较低。它也拥有SRAM和DRAM存储，可以从正常SSD读取/写入单元的数据。数据在iSSD中的DRAM和主机的主存储器之间通过由主机接口控制器控制的主机接口传输。

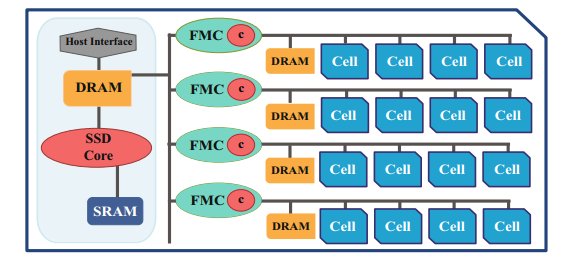


图2.1 iSSD结构

为了充分利用iSDD内部的I / O带宽进行数据处理，iSSD内部需要有足够的处理能力，否则可能会成为新的性能瓶颈。文章为解决该类问题，在iSSD内部的每个通道上添加一个廉价的（虽然性能相对较低的）处理器，称为通道CPU。

这种方法的优点如下：（1）数据可以在所有通道CPU上并行处理。在通道CPU中处理的数据的大小与iSSD中的通道数成正比。因此如果我们增加iSSD内的通道数量，每个通道CPU就足够处理它的数据。（2）在iSSD中增加一个通道不仅可以扩展内部带宽，还可以提高内部处理能力，从而不会给主机接口和主机CPU造成负担，因为它仅接收由iSSD处理的较小结果。（3）在散热方面，一些廉价通道的CPU比单个或几个高性能的核心CPU要好。

在存储设备中处理数据的研究有很多，主要集中在硬盘驱动器上。由于HDD由于其物理特性（即非常高的查找时间和旋转延迟）而对内部I / O带宽的扩展有限制，所以它们的方法尚未实际采用。

相比之下，由于SSD不采用机械设备，并且具有很高的内部I / O带宽的可扩展性，iSSD可以成为处理大规模数据挖掘应用的一个很好的解决方案。为了支持相应的观点，最近几项研究已经探索了在iSSD上存储处理的可行性。

## 2.2 将数据挖掘算法建模为同步数据流

文章采用同步数据流来描述数据挖掘算法的流程。在SDF图中，其中的函数和调用分别表示为节点和边。边的两端都有数字，表示连接到边缘交换机的两个节点的IO令牌的数量（称为采样率）。图2.2显示了一个SDF图的例子，有三个节点。节点A只在执行期间生成一个IO令牌，节点B需要两个IO令牌（从节点A获得）执行，并在执行过程中生成一个IO令牌。最后Node C只需要一个IO令牌来执行。

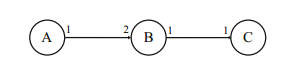


图2.2 SDF示例

在本文中，我们选择了kmeans，PageRank和SimRank算法，因为它们在很多应用领域非常流行，而且也是属于数据密集型的算法。对它们构建算法的SDF图并进行相应评估。

k-means算法是数据挖掘领域应用最广泛的聚类算法之一。执行过程如下： 首先，它读取存储设备中的每个对象（Read）并计算对象与当前k个簇的质心之间的距离。根据距离，将对象分配到距离对象最近的聚类（SelClust）。分配所有对象后，重新计算新创建的k个簇的新质心（NewClust）。它重复这个过程预定的次数，处理完成后，将结果存储到存储设备（写入）中。图2.3表示k-means方法的SDF图，其中n表示数据中的对象的数量，p表示每个节点的数据粒度，k表示中央集群的数量。

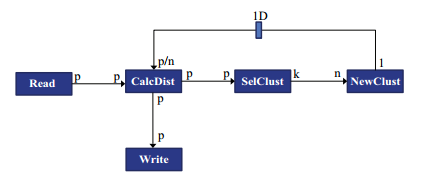


图2.3 k-means SDF

PageRank是一个众所周知的排名算法。其中ri是排名向量，W是邻接矩阵，a是阻尼因子。



PageRank算法的执行如下：从存储设备读取邻接矩阵（Read）。然后通过将矩阵W的行乘以向量ri-1（乘法）来获得元素值。矢量是通过组合Multiply（MakeVec）所计算的所有元素来完成的。向量ri是通过在MakeVec（AddDamp）中添加此向量和ar0生成的。这个过程重复了预定义的次数。当过程完成时，vectorri被存储到一个存储设备（Write）。 图2.4示出了具有m次迭代的PageRank的SDF图，其中n表示排序向量中的元素的数量，1D是延迟的缓冲器，其保持用于乘法的初始向量r0。

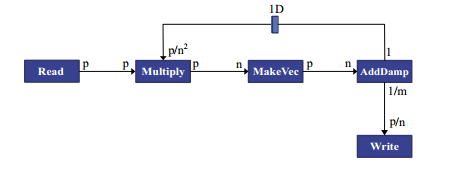


图2.4 PageRank SDF

SimRank是一种通过采用矩阵乘法计算基于链接的相似度的算法，如公式2所示，其中S代表相似度矩阵，W代表邻接矩阵，c代表阻尼因子。



SimRank执行过程如下：从存储设备读取邻接矩阵（Read）。矩阵与Sk-1相乘，然后矩阵W乘以和Sk-1（MakeMat）相乘得到的矩阵。 Sk是通过将MakeMat和c的结果相乘并将（1-c）S0加到它（AddDamp）得到的。这个过程重复预定的次数值。矩阵 Sk存储在存储设备（写入）中。 图2.5示出了具有m次迭代的SimRank的SDF图，其中n是矩阵Sk-1中的元素的数量，并且1D是保持初始矩阵S0的延迟缓冲器。

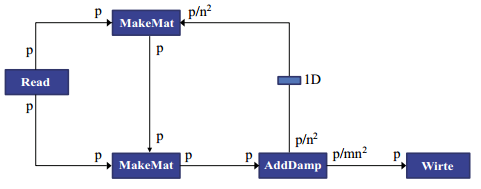


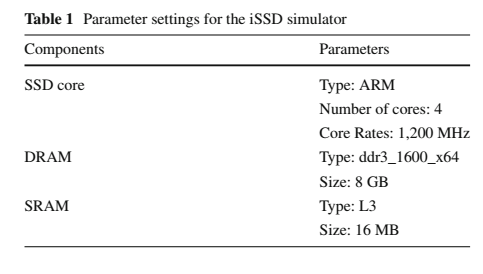
图2.5 SimRank SDF

# 3 评 估

为衡量三种数据挖掘算法的性能，先开发iSSD模拟器,使得能够在模拟器中运行数据挖掘算法,并评测相应的数据处理性能。

## 3.1 iSSD模拟器的实现

文章在gem5模拟器上实现iSSD模拟器。gem5模拟器提供了一个虚拟计算环境，可以允许不同类型和性能的处理器，以及内存和目标系统的缓存，还提供了多种功能和组件，使其具有很大的灵活性。文章采用处理器、内存和缓存作为SSD内核，DRAM和SRAM,为提高模拟器的准确性,将CPU模型设置为AtomicSimple，它是一个最小的单个IPC CPU模型，系统模式采用系统调用仿真模式，避免了模拟设备或操作系统系统级服务。表1给出了iSSD模拟器的相关参数。



由于gem5的部分限制，以gem5为基础进行了部分改进：

gem5难以仿真一个使其行为与iSSD的相同频道的渠道。为使每个通道能够独立于其他通道处理自己单元的数据，文章将iSSD中的FMC核心替换为SSD核心，并且将SSD核心的数量和速率设置为遵循FMC核心的数量和速率。

在gem 5模拟器中不允许IO模拟。为计算IO时间，文章通过将总数据大小除以页面大小（在操作系统中定义）来统计IO的数量（#I O）、通过参考由SSD制造商提供的目录，找到用于从/向SSD读/写单页的IO时间（T\_I O）、通过乘以＃I O和T \_I O来计算给定数据的IO时间、将总IO时间除以SSD内核（#SSDcore）的数量。iSSD中的IO时间按照公式3：



在iSSD仿真器中对算法进行如下处理：

（1）将算法的二进制码与主机的数据一起传输到iSSD;

（2）每个SSD内核作为单个线程执行二进制代码;

（3）从SSD核心获得的结果被传送到主机。

我们通过计算在这三个步骤中花费的时间来测量总处理时间。

## 3.2数据挖掘算法的实现

文章仔细设计了数据挖掘算法的函数粒度来决定哪个函数与多个线程并行执行，或者用单个线程执行。

k-means：SelClust作为一个单线程来实现，因为它简单并且所需的计算量比其他函数少。 CalcDist和NewClust并行处理，实现为多个线程，以更高效地处理数据。

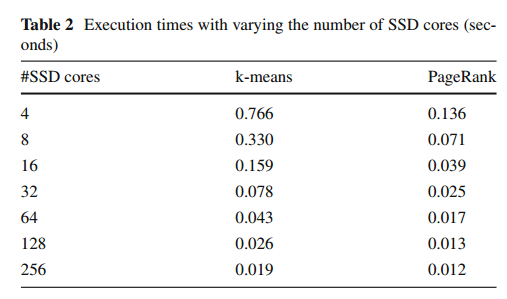
PageRank：MakeVec和AddDamp作为一个单线程来实现，简单并且比Multiply需要更少的计算。因为乘法需要并行化以实现高效处理，所以被实现为多个线程，每个线程被分配相同数量的节点。

SimRank：MakeMat和AddDamp分别由多个线程和一个线程组成。原因分别与PageRank实现中的Multiply和AddDamp相同。

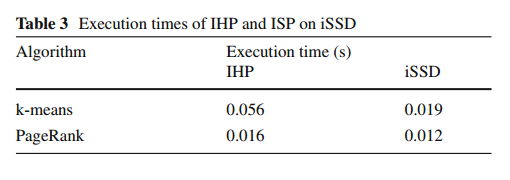
## 3.3 评估模拟器

为了在数据挖掘算法上评估iSSD上的ISP（In-Storage Processing），配备Intel i5 3.2 GHz，8 GB主内存和Intel SSD 530的Linux服务器上进行了一系列的实验.iSSD的参数如表1所示，在实验中使用了小数据集。

为评估在iSSD中使用ISP执行的数据挖掘算法的性能的实验结果。 测量了执行时间，同时更改了iSSD内SSD内核的数量。表2显示了结果：k-means和PageRank都表明，执行时间随着SSD上ISP的SSD核心数量的增加而减少。



为评估IHP的性能，比较iSSD的ISP（存储处理）和IHP（主机内部处理）的性能。文章测量了在主机CPU上使用与ISP中使用的数据集相同的数据挖掘算法的时间。为了评估ISP的性能，在iSSD模拟器中，SSD内核的数量和速率分别设置为256和1200 MHz，表3显示了实验的结果。iSSD上的ISP显示，在k-means和pageRank算法中，性能比IHP提高了约300和130％。在这些算法中，并行执行的功能在其总执行中占很大比例。因此配备更多SSD内核的iSSD执行时间较短些。



# 4 协作环境下的数据处理

如何在具有CPU，GPU和iSSD的协作环境中处理数据挖掘算法，以实现最高的效率？文章选择了最佳虚拟水平（BIL）的异构静态调度，通过实验比较它比一般动态水平（GDL）调度提供了更好的性能。

BIL调度通过使用非循环优先扩展图（APEG）产生一个调度。它可以减少节点的执行时间。它选择APEG中具有最长执行时间的节点，并且分配一个速度最快的处理器。BIL调度需要以下输入：（1）目标算法的APEG（APEG中的节点和边分别代表函数和调用关系）（2）时间表（包括每个功能模块的处理器执行时间和处理器间的IPC时间）。

关于目标算法的APEG意味着我们需将数据挖掘算法建模为APEG。在此过程中我们需要确定图中的函数粒度和数据粒度。在新环境中，GPU拥有大量的内核，提供了高度的并行性。iSSD提供了一定程度的并行性和快速的IO速度。同时CPU具有较高的处理能力，有利于执行合并功能等开销较高的功能。因此，文章将函数分为两类：（1）将来自其他函数的结果进行合并的数据的函数和（2）能够并行处理数据的函数。

为确定算法的最有效调度的最佳数据和功能粒度。执行以下过程：首先使用多个节点为每个函数建立尽可能细的粒度，并将每个节点分为上述两个类别之一。 然后选择一个节点分配给GPU，根据公式确定其数据粒度。现在有一个最细粒度的APEG，合并属于同一个函数的节点后，以类似的方式构建一个粗粒度的APEG。然后为这样获得的APEG创建不同的时间表。用这些时间表运行算法多次，并选择最好的一个作为未来使用的算法的最终时间表。

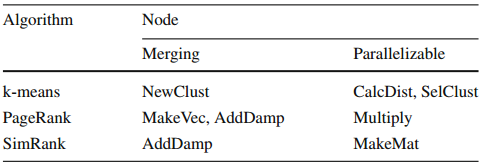
由于BIL调度是一个静态调度，函数和数据粒度需根据输入数据的大小来确定。若不同大小的数据到达，则需更改函数和数据粒度，同样不同的粒度导致了不同的时间表。为解决这个问题，文章通过略微修改APEG中节点的数据粒度来回收预先生成的时间表。首先生成第一个输入数据的时间表（称为基本时间表）。当一个新的不同大小的输入数据到达时，调整数据粒度来处理新的数据。同时根据调整的粒度重新计算节点的执行时间和IPC时间。

# 5 实验过程

## 5.1 实验设置

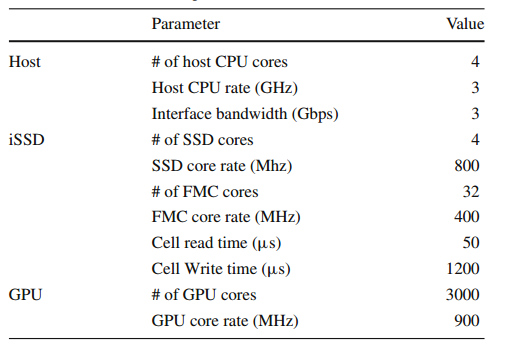
将三种算法的SDF图中的节点分类为合并节点和可并行化节点。表4 给出了相关节点分类。

表4 节点分类



关于性能评估，文章采用Vtune来衡量执行一个节点的时间和IPC的时间。表5给出了与硬件仿真相关的参数。在实验中以随机的方式综合生成了100GB的数据。k-means数据集包含一个对象列表，每个对象都有八个维度。每个维度的值范围从1到32,000。PageRank和SimRank的数据集由连接源节点和目标节点的边的列表组成，节点的数量约为800,000，对象中的每个维度的值和节点的索引都是随机选择的。假设数据均匀地存储在iSSD中的所有单元上，从而使所有FMC内核都能处理类似的IO工作负载。

表5 硬件相关参数



为了评估协作环境的影响，实验在不同的计算环境下反复运行BIL调度。把一个只有CPU和iSSD的环境分别称为IHP和ISP。此外将计算环境设置为CPU+GPU，CPU+iSSD，CPU+GPU+iSSD。由于IHP是一种异构调度算法，并不适合BIL调度，实验中的IHP执行时间是从一台真机上获得的。

## 5.2 确定粒度

为找到适合最佳时间表的功能和数据粒度文章进行了相关实验。首先使用章节中介绍的图表确定APEG在一个最细粒度的水平。然后对于可并行化节点，根据式（1）确定数据粒度。接下来建立一个粗粒度的APEG。在k-means中，文章将CalcDist和SelClust合并到一个粗粒度节点SetClust中。在PageRank中，实验通过将MakeVec和AddDamp结合到ResultVec（一个粗粒度节点）来构建一个SDF图。在SimRank中，为了在粗粒度级上构建SDF图，将MakeMat（可并行化节点）和AddDamp（合并节点）合并到粗粒度节点ResultMat中。

## 5.3 实验结果

表6显示了三种算法的执行时间，从不同粒度级别的APEG中获得的时间表。在这里方程4作为一个数据粒度。在k-means中，实验试图通过将两个可并行化的节点组合成一个节点来减少IPC时间。但是细粒度级别的APEG比粗粒度级别表现出更好的性能。所以在单个节点中包含CalcDist和SelClust的SetClust具有较大的数据粒度，以便在GPU中进行处理。但是同样注意到，SelClust（可并行化节点）可以在iSSD或CPU中处理，而不必传输到GPU。在PageRank中，通过将MakeVec和AddDamp结合到ResultVec中，性能会变得更好，从而减少IPC执行时间。在SimRank中，因为MakeMat和AddDamp属于不同的类别，所以使用单独的MakeMat和AddDamp更为有效。在下面的实验中，我们将APEG用于k-means和SimRank的细粒度级别，PageRank则为粗粒度级别。

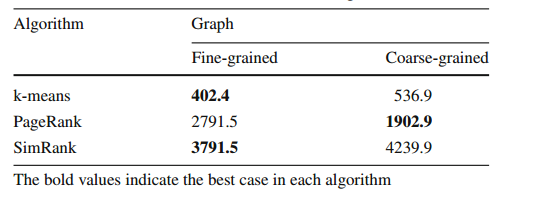
表6 不同算法的执行时间 

表7显示了三种算法根据不同数据粒度获得的时间表的执行时间。结果表明，当m被设置为GPU内核的倍数时，执行时间减少。在其他情况下，由于GPU中的内核未被充分利用，执行时间会变长。在下面的实验中，我们使用m设为1的数据粒度式。

表7 不同数据粒度的执行时间

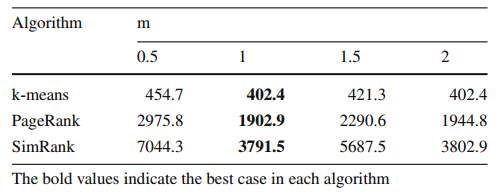
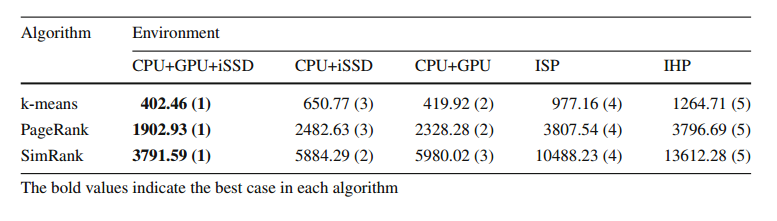


表8显示了每个计算环境的执行时间。结果表明：协作环境中的执行时间远小于非协作环境中的时间，性能差距随着节点时间复杂度和并行度的提高而增加。使用GPU和iSSD的CPU + GPU + iSSD显示最佳性能。分析原因是可并行化的节点可以在iSSD或GPU中并行处理，从而减少了执行时间。

表8 每种环境下的执行时间



CPU + GPU + iSSD中的计算资源在执行算法时彼此很好地协作。图5.1显示了每个函数的资源利用情况。x轴表示算法中的节点，y轴表示资源的利用率。 结果表明，在处理所有算法时，资源之间的协作是有效的。在k-means中，NewDist（合并节点）和SelClust（可并行化节点）主要在CPU中处理。CalcDist（可并行化节点）主要由BIL调度器在GPU中分配。iSSD在这之中似乎扮演了补充其他计算资源的角色，在PageRank中，ResultVec（合并节点）在CPU中处理，Multiply（可并行化节点）大部分在GPU和iSSD中处理。然而在SimRank中，AddDamp在除了GPU以外的所有计算资源中处理，由于SimRank时间复杂度高，文章猜测它会利用所有的资源。

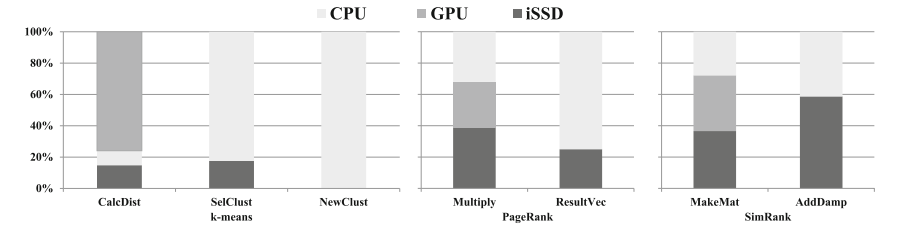


图5.1 资源的利用情况

## 5.4 更改参数

文章通过改变与iSSD和GPU内硬件组件相关的参数值来衡量执行时间。 在图5.2中，左侧显示了改变FMC核心数量的结果; 中心通过改变SSD内核的数量来做结果; 右侧的结果是改变了GPU内核的数量。 x轴表示核心数量，y轴表示执行时间。每一行表示计算环境的执行时间。

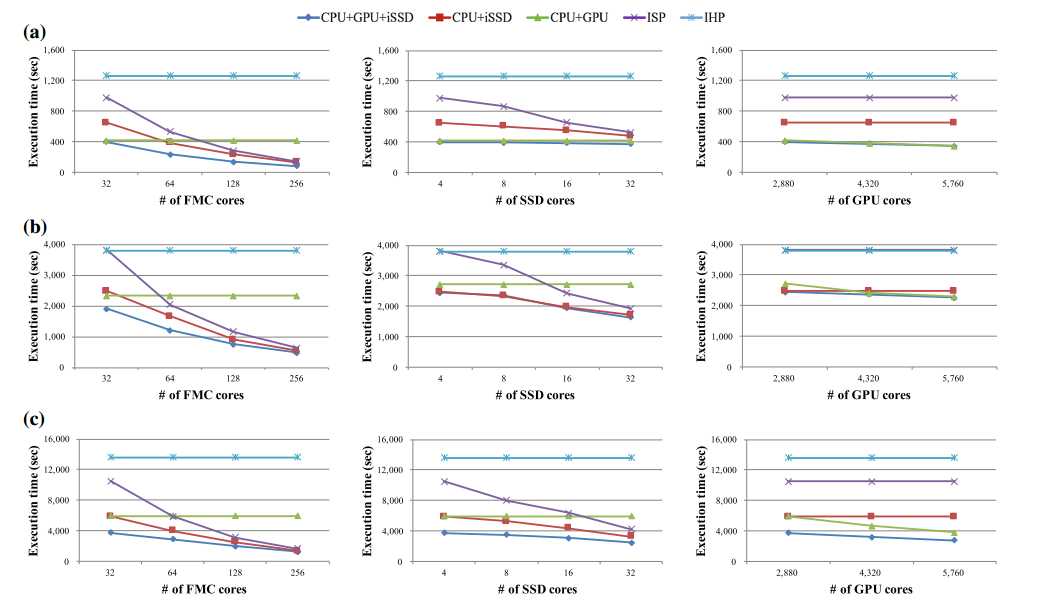


图5.2 更改资源上的核心数量下的执行时间

结果表明：随着FMC数量的增加，ISP的性能会有所提高。同样相应的协作环境的表现也可以得到改善。由于FMC内核主要处理iSSD中的算法，因此SSD内核数量的性能提升要低于FMC内核。配备GPU的环境的执行时间也随着GPU内核数量的增加而减少。结果显示CPU + GPU + iSSD的性能比IHP高出约10倍。

# 6 结论

文章开发了一个基于gem 5仿真器的iSSD仿真器，并评估数据挖掘算法在仿真器上处理数据的有效性。通过这个基于模拟器的评估，文章成功验证了iSSD的潜力。并且考虑了可能嵌入CPU，GPU和iSSD等不同的计算资源的计算环境，由于三种计算资源具有不同的特征，通过实验结果看出CPU功能最强大，适合处理计算复杂度高的功能。在GPU的情况下，其SIMT架构能够同时处理大量数据的相同作业。iSSD是一种内部具有内核的未来SSD，比其他两种更容易访问数据，并且适合于以低并行计算复杂度来处理功能。

文章解决了在三种计算环境下如何协作有效地执行数据挖掘算法。为充分利用这三种资源，设计了异构调度来将数据挖掘算法的功能分配给适当的计算资源。并解决了确定影响协作环境性能的数据和功能粒度的问题。

最后，我们通过一系列实验展示了数据处理在协作环境中的优越性。结果显示，我们的协作环境中的数据处理性能比非协作环境下的处理性能高出10倍以上。

# 7 思考

文章仅仅是在单机上验证了使用单个CPU、单个GPU和单个iSSD组成的协作环境上执行数据挖掘算法是有效的。作为进一步的研究，我们还需要研究如何在一台机器上具有多个CPU，多个GPU和多个iSSD的环境中处理数据挖掘算法，以及如何在由多个节点组成的分布式环境中有效的执行数据挖掘算法，并实现高性能的数据挖掘。