(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102200906 B
(45)授权公告日 2013.12.25

(21)申请号 201110135906.0

(22)申请日 2011.05.25

(73)专利权人 上海理工大学
地址 200093 上海市杨浦区军工路516号

(72)发明人 陈庆奎 那丽春 周澍民 刘伯承
王海峰 郝聚涛 霍欢 赵海燕
庄松林 丁晓东

(74)专利代理机构 上海申汇专利代理有限公司 31001
代理人 吴宝根

(51)Int.Cl.
G06F 9/38(2006.01)
G06F 9/48(2006.01)

审查员 邢白灵

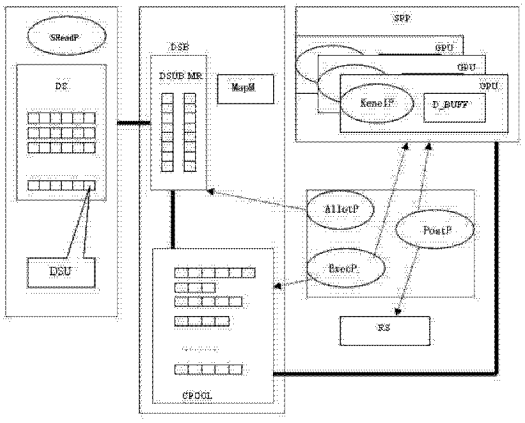
权利要求书3页 说明书7页 附图2页

(54)发明名称

大规模并发数据流处理系统及其处理方法

(57)摘要

一种大规模并发数据流处理系统及其处理方法，涉及数据处理技术领域，所解决的是提高流处理器处理效率的技术问题。该系统包括数据流单元缓冲区、数据流单元聚类队列池、数据流单元映射表、流处理器池、数据流读取部件、DSU聚类分配部件、任务调度部件、计算后处理部件，所述流处理器池由多个GPU构成，其中数据流读取部件用于将并发数据流写入数据流单元缓冲区，DSU聚类分配部件用于对数据流单元缓冲区中当前被处理的数据流单元进行分类，任务调度部件用于将数据流单元聚类队列池中的就绪队列加载至流处理器池中的GPU上执行流计算，计算后处理部件用于将GPU的计算结果返回到数据流。本发明提供的系统，能提高流处理器的处理效率。



CN 102200906 B

1. 一种大规模并发数据流处理系统，其特征在于，包括：

（1）数据流读取部件，用于读取数据流；

数据流单元缓冲区，是一个二元组 DSB(DSUB,MR)，其中 DSB 为数据流单元缓冲区，DSUB 及 MR 均是由 p 个元素构成的一维数组，p 为并发数据流中的数据流数量，DSUB 中的每个数组元素为一个 DSU，MR 中的每个数组元素是一个取值为 0 或 1 的整型数，该数组用于数据流流水处理的同步标志；

所述 DSU 是指数据流单元，一个数据流单元是一个九元组 DSU（id,sno,segno,seq,t,type,prog,data,odata)，其中 DSU 为数据流单元，id 为该 DSU 的标识符，且该 id 具有唯一性，sno 为该 DSU 的数据流号，segno 为该 DSU 的数据流段号，seq 为该 DSU 的在 segno 数据流段中的单元序号，用于表示其在数据流段中的位置，t 为一个时间印，用于记载该 DSU 被处理的时刻，type 为该 DSU 的类型，data 为该 DSU 所承载的数据对象，odata 为该 DSU 处理后的输出数据对象，prog 是该 DSU 的 data 的处理程序；

所述数据流段是由多个 seq 连续的 DSU 构成的序列，记为 DSS={DSU1,DSU2,DSU3,…,DSUn,DSUE}，其中 DSS 为数据流段，每个 DSS 均有一个数据流段号 segno 被分别存储在构成该 DSS 的每个 DSU 中，DSS 序列尾的 DSUE 为该 DSS 的结束标志，是一个 type 为常量值 EOS 的 DSU，其 prog、data、odata 均为空；

所述数据流是由多个 segno 连续的 DSS 构成的序列，记为 DS= {DSS1,DSS2,DSS3, …, DSSo}，每个 DS 均有一个数据流号 sno 被分别存储在构成该 DS 的各个 DSS 的 DSU 中；

所述并发数据流由多个并发传输的 DS 构成，每个 DS 均以 DSU 作为并发处理的单位，并以 DSS 作为多个数据流并发同步的单元；

数据流单元聚类队列池，由 |TS| 个 DSU 队列构成，记为 CPOOL= {$DSUQ_1$, $DSUQ_2$, …, $DSUQ_{|TS|}$}，其中 CPOOL 为数据流单元聚类队列池，DSUQ 为数据流单元聚类队列，TS 为应用系统数据流单元类型集合，该集合是 DSU 类型的集合，TS 中的元素个数为 m，则 |TS|=m，同一个 DSU 队列由同类型的 DSU 构成，这些 DSU 来自 p 个并发数据流的当前处理单元，有：

$$\sum_{i=1}^{|TS|} |DSUQ_i| = p$$ ；

数据流单元映射表，由多个表元构成，记为 MapM（nu，sno，segno，seq，t，qso，qoffset)，其中 MapM 为数据流单元映射表，nu 为序号，sno 为数据流号，segno 为数据流段号，seq 为数据流单元号，t 为时间印，qso 为聚类队列号，qoffset 为聚类队列内部元素位置号；

流处理器池，由多个 GPU 构成，所述 GPU 为二元组 GPU（KernelP，D_BUFF)， 其中 KernelP 为该 GPU 当前执行 SPMD 任务的计算核心部件，D_BUFF 为 KernelP 执行 SPMD 操作的多个 DSU 集合；

（2）DSU 聚类分配部件，用于对数据流单元缓冲区中当前被处理的数据流单元进行分类；

（3）任务调度部件，用于将数据流单元聚类队列池中的就绪队列加载至流处理器池中的 GPU 上执行流计算；

（4）计算后处理部件，用于将 GPU 计算的 DSU 的 odata 按 MapM 的标志回归到 DSU 所在

的数据流。

2. 根据权利要求 1 所述的大规模并发数据流处理系统的处理方法，其特征在于：

数据流读取部件重复执行以下步骤直至并发数据流中的 DS 读取完毕：

1）根据并发数据流的个数，在 DSB 中为每个 DS 分配一个单元，并初始化 DSB 的 MR，置 MR[i] 值为 0，其中 1 ≤ i ≤ p，p 为并发数据流的个数；

2）读取并发数据流中所有 DS 的当前 DSS；

3）扫描并发数据流，对 i=1，2，…，p，对 DS_i 做步骤 4 的处理，所述 DS_i 是指第 i 个 DS；

4）如果 MR[i] 值为 1，则转至步骤 3 处理下一个 DS 的 DSU；

如果 MR[i] 值为 0，则提取 DS_i 的当前处理 DSU，并判断当前处理 DSU 的 type，如果当前处理 DSU 的 type 值为 EOS，则 DS_i 的当前 DSS 结束，则置 MR[i] 为 1，并转至步骤 3 处理下一个 DS 的 DSU，反之则判断 DSUB[i] 是否为空，如 DSUB[i] 为空，则把当前处理 DSU 存入 DSUB[i]；

5）如果 DSUB 的所有元素均置满数据，则等待至 DSUB 的所有元素都被 DSU 聚类分配部件置为空；

6）如果 DSB 中的 MR 的所有元素都为 1，则转至步骤 1 处理并发数据流中所有 DS 的下一个 DSS，反之则转向步骤 2 继续处理当前 DSS 的 DSU；

DSU 聚类分配部件重复执行以下步骤：

1）判别 DSB 的 DSUB 中是否置满数据，如果未满则重复本步骤，反之则转至步骤 2；

2）判别是否收到来自任务调度部件的“数据流处理完毕”消息，如果未收到则重复本步骤，反之则转至步骤 3；

3）对 i=1，2，…，p，分类处理 DSUB[i]，其分类处理步骤如下：

如果 DSUB[i] 的 type 值不是 EOS，则将 DSUB[i] 加入 CPOOL 的第 w 个数据流聚类队列 $DSUQ_w$ 中，其中 w 值等于 DSUB[i] 的 type 值；然后获取 DSUB[i] 在 $DSUQ_w$ 的位置下标，记为 pos，并置 MapM[i] 的 nu 值为 i，置 MapM[i] 的 sno 值为 i，置 MapM[i] 的 segno 值为 DSUB[i] 的 segno 值，置 MapM[i] 的 seq 值为 DSUB[i] 的 seq 值，置 MapM[i] 的 t 值为 DSUB[i] 的 t 值，置 MapM[i] 的 qso 值为 w 值，置 MapM[i] 的 qoffset 值为 pos，然后置 DSUB[i] 为空；

4）向任务调度部件发送“数据流聚类队列构建完毕”消息；

任务调度部件执行以下步骤：

1）判别是否收到来自 DSU 聚类分配部件的“数据流聚类队列构建完毕”消息，如果未收到则重复本步骤，反之则转至步骤 2；

2）为流处理器池中的各个 GPU 配置一个工作标志数组 work，并对 i=1，2，3，…，q，置 work[i]=0，其中 q 为流处理器池中的 GPU 数量；

3）从 CPOOL 中提取 q 个 DSUQ 以及每个队列所对应 GPU 的 KernelP，构成任务对（$DSUQ_1$，$Kernel_1$），（$DSUQ_2$，$Kernel_2$），…，（$DSUQ_q$，$Kernel_q$）；

4）对 i=1，2，…，q，分别加载（$DSUQ_i$，$Kernel_i$）到 GPU_i 执行步骤 5，其中 GPU_i 是指第 i 个 GPU；

5）向 GPU_i 的存储器申请 $DSUQ_i$ 大小的存储单元 D_BUFF_i，然后将 $DSQU_i$ 的内容加载到 D_BUFF_i，然后再提交 $Kernel_i$ 及 D_BUFF_i 到 GPU_i 执行；

6）监控所有 GPU 的执行状况，如果 GPU_i 执行完毕，则向计算后处理部件发送“GPU_i 数

据流处理完毕”消息，并从 CPOOL 中提取下一个未被执行的任务对（$DSUQ_i$，$Kernel_i$）后转至步骤 5；如果 CPOOL 中的所有 DSUQ 都被加载执行完毕，则向 DSU 聚类分配部件发送“数据流处理完毕”消息，并对所有的 i 置 work[i]=0，然后再转至步骤 1；

计算后处理部件执行以下步骤：

1）判别是否收到来自任务调度部件的“GPU_i 数据流处理完毕”消息，如果未收到则重复本步骤，反之则转至步骤 2；

2）向内存申请 D_BUFF_i 大小空间的 POST_DSUQ，所述 POST_DSUQ 的结构与 DSUQ 的结构一致；

3）先将 D_BUFF_i 的内容加载到 POST_DSUQ，再释放 D_BUFF_i 的空间；

4）扫描 POST_DSUQ 中的每个 DSU，将 DSU 按照 MapM 记载的位置映射信息还原到相应的 DS 中，保持原有 DS 的顺序，并把结果写入 RS；

5）转至步骤 1；

GPU_i 上的 Kernel 执行以下步骤：

1）获取 $Kernel_i$ 及 D_BUFF_i，并计算出 D_BUFFi 中的 DSU 数量记为 g；

2）在 GPU_i 的各个物理流处理单元分配 DSU，每个物理流处理单元得到$\lceil g/h \rceil$个 DSU，其中 h 为 GPU_i 的物理流处理单元数量；

3）所有物理流处理单元并行地对其分配到的 DSU 执行 $Kernel_i$ 进行处理，并输出计算结果到其所处理的 DSU 的 odata；

4）GPU_i 计算结束。

大规模并发数据流处理系统及其处理方法

技术领域

[0001] 本发明涉及数据处理技术，特别是涉及一种大规模并发数据流处理系统及其处理方法的技术。

背景技术

[0002] 随着信息技术的飞速发展和互联网技术应用的普及，网络已经成为人们日常生活中重要的一部分。近年来，3G 网络和物联网技术应用的逐步展开为人们的生活、工作带来了极大的方便。然而，这些新技术的核心应用关键之一就是大规模并发数据流处理问题。所谓数据流就是从一个节点发往另一个节点的具有某种特征的数据单元构成的连续不断的信息流，数据流的处理和分析问题要求处理节点具有一定的实时处理能力。人们可以通过缓冲处理、并行处理机制来解决实时处理问题。然而，现实应用中的数据流并非一个，在两个处理节点间的数据流可能是成千上万个，这就形成了大规模并发数据流的处理需求。如 3G 通信视频流的质量实时分析问题，在这个问题中，数据抓取节点从 3G 骨干网络上同时抓去上万门 3G 通话的 IP 包，并还原成上万个 H.264 视频流，而 3G 视频质量分析系统需要提取这些视频流在某一时刻的上万个静态画面，并进行模糊度计算、块效应计算等工作。大规模并发数据流的实时处理问题给业界带来了极大的挑战。如何构建廉价的高性能处理装置是一个具有挑战性的研究课题。以流处理器为代表的新型计算装置为解决这些困难带来了曙光。流处理器是 GPU 走向通用计算领域的总称，因其内部采用上百个 Stream processor 并行架构而得名，其可以有效支持 SPMD 并行操作，非常适合做大规模并发数据流的处理工作，且其性能十分出众。

[0003] 但是，由于同一个流处理器核心在同一时刻只能运行同一个核心程序，而同时到来的大规模并发数据流的成千上万个数据流单元却不一定具有同一处理特征，因而无法满足流处理器进行高性能处理的数据特征要求，无法使流处理器发挥其最大能力，进而提高处理效率。

发明内容

[0004] 针对上述现有技术中存在的缺陷，本发明所要解决的技术问题是提供一种能发挥流处理器的最大能力，提高其处理效率的大规模并发数据流处理系统及其处理方法。

[0005] 为了解决上述技术问题，本发明所提供的一种大规模并发数据流处理系统，其特征在于，包括：

[0006] 数据流单元缓冲区，是一个二元组 DSB(DSUB,MR)，其中 DSB 为数据流单元缓冲区，DSUB 及 MR 均是由 p 个元素构成的一维数组，p 为并发数据流中的数据流数量，DSUB 中的每个数组元素为一个 DSU，MR 中的每个数组元素是一个取值为 0 或 1 的整型数，该数组用于数据流流水处理的同步标志；

[0007] 所述 DSU 是指数据流单元，一个数据流单元是一个九元组 DSU（id，sno，segno，seq，t，type，prog，data，odata)，其中 DSU 为数据流单元，id 为该 DSU 的标识符，且该 id

具有唯一性，sno 为该 DSU 的数据流号，segno 为该 DSU 的数据流段号，seq 为该 DSU 的在 segno 数据流段中的单元序号，用于表示其在数据流段中的位置，t 为一个时间印，用于记载该 DSU 被处理的时刻，type 为该 DSU 的类型，data 为该 DSU 所承载的数据对象，odata 为该 DSU 处理后的输出数据对象，prog 是该 DSU 的 data 的处理程序；

[0008] 所述数据流段是由多个 seq 连续的 DSU 构成的序列，记为 DSS= {DSU1，DSU2，DSU3，…，DSUn，DSUE}，其中 DSS 为数据流段，每个 DSS 均有一个数据流段号 segno 被分别存储在构成该 DSS 的每个 DSU 中，DSS 序列尾的 DSUE 为该 DSS 的结束标志，是一个 type 为常量值 EOS 的 DSU，其 prog、data、odata 均为空；

[0009] 所述数据流是由多个 segno 连续的 DSS 构成的序列，记为 DS= {DSS1，DSS2，DSS3，…，DSSo}，每个 DS 均有一个数据流号 sno 被分别存储在构成该 DS 的各个 DSS 的 DSU 中；

[0010] 所述并发数据流由多个并发传输的 DS 构成，每个 DS 均以 DSU 作为并发处理的单位，并以 DSS 作为多个数据流并发同步的单元；

[0011] 数据流单元聚类队列池，由 |TS| 个 DSU 队列构成，记为 CPOOL= {$DSUQ_1$，$DSUQ_2$，…，$DSUQ_{|TS|}$}，其中 CPOOL 为数据流单元聚类队列池，DSUQ 为数据流单元聚类队列，TS 为应用系统数据流单元类型集合，该集合是 DSU 类型的集合，TS 中的元素个数为 m，则 |TS|=m，同一个 DSU 队列由同类型的 DSU 构成，这些 DSU 来自 p 个并发数据流的当前处理单元，有：

$$\sum_{i=1}^{|TS|} |DSUQ_i| = p$$ ；

[0012] 数据流单元映射表，由多个表元构成，记为 MapM（nu，sno，segno，seq，t，qso，qoffset），其中 MapM 为数据流单元映射表，nu 为序号，sno 为数据流号，segno 为数据段号，seq 为数据流单元号，t 为时间印，qso 为聚类队列号，qoffset 为聚类队列内部元素位置号；

[0013] 流处理器池，由多个 GPU 构成，所述 GPU 为二元组 GPU（KernelP，D_BUFF），其中 KernelP 为该 GPU 当前执行 SPMD 任务的计算核心部件，D_BUFF 为 KernelP 执行 SPMD 操作的多个 DSU 集合；

[0014] 数据流读取部件，用于读取数据流；

[0015] DSU 聚类分配部件，用于对数据流单元缓冲区中当前被处理的数据流单元进行分类；

[0016] 任务调度部件，用于将数据流单元聚类队列池中的就绪队列加载至流处理器池中的 GPU 上执行流计算；

[0017] 计算后处理部件，用于将 GPU 计算的 DSU 的 odata 按 MapM 的标志回归到 DSU 所在的数据流。

[0018] 本发明提供的大规模并发数据流处理系统的处理方法，其特征在于：

[0019] 数据流读取部件重复执行以下步骤直至并发数据流中的 DS 读取完毕：

[0020] 1）根据并发数据流的个数，在 DSB 中为每个 DS 分配一个单元，并初始化 DSB 的 MR，置 MR[i] 值为 0，其中 1 ≤ i ≤ p，p 为并发数据流的个数；

[0021] 2）读取并发数据流中所有 DS 的当前 DSS；

[0022] 3）扫描并发数据流，对 i=1，2，…，p，对 DSi 做步骤 4 的处理，所述 DS_i 是指第 i 个

DS；

[0023] 4）如果 MR[i] 值为 1，则转至步骤 3 处理下一个 DS 的 DSU；

[0024] 如果 MR[i] 值为 0，则提取 DS_i 的当前处理 DSU，并判断当前处理 DSU 的 type，如果当前处理 DSU 的 type 值为 EOS，则 DS_i 的当前 DSS 结束，则置 MR[i] 为 1，并转至步骤 3 处理下一个 DS 的 DSU，反之则判断 DSUB[i] 是否为空，如 DSUB[i] 为空，则把当前处理 DSU 存入 DSUB[i]；

[0025] 5）如果 DSUB 的所有元素均置满数据，则等待至 DSUB 的所有元素都被 DSU 聚类分配部件置为空；

[0026] 6）如果 DSB 中的 MR 的所有元素都为 1，则转至步骤 1 处理并发数据流中所有 DS 的下一个 DSS，反之则转向步骤 2 继续处理当前 DSS 的 DSU；

[0027] DSU 聚类分配部件重复执行以下步骤：

[0028] 1）判别 DSB 的 DSUB 中是否置满数据，如果未满则重复本步骤，反之则转至步骤 2；

[0029] 2）判别是否收到来自任务调度部件的“数据流处理完毕”消息，如果未收到则重复本步骤，反之则转至步骤 3；

[0030] 3）对 i=1，2，…，p，分类处理 DSUB[i]，其分类处理步骤如下：

[0031] 如果 DSUB[i] 的 type 值不是 EOS，则将 DSUB[i] 加入 CPOOL 的第 w 个数据流聚类队列 $DSUQ_w$ 中，其中 w 值等于 DSUB[i] 的 type 值；然后获取 DSUB[i] 在 DSUQw 的位置下标，记为 pos，并置 MapM[i] 的 nu 值为 i，置 MapM[i] 的 sno 值为 i，置 MapM[i] 的 segno 值为 DSUB[i] 的 segno 值，置 MapM[i] 的 seq 值为 DSUB[i] 的 seq 值，置 MapM[i] 的 t 值为 DSUB[i] 的 t 值，置 MapM[i] 的 qso 值为 w 值，置 MapM[i] 的 qoffset 值为 pos，然后置 DSUB[i] 为空；

[0032] 4）向任务调度部件发送“数据流聚类队列构建完毕”消息；

[0033] 任务调度部件执行以下步骤：

[0034] 1）判别是否收到来自 DSU 聚类分配部件的“数据流聚类队列构建完毕”消息，如果未收到则重复本步骤，反之则转至步骤 2；

[0035] 2）为流处理器池中的各个 GPU 配置一个工作标志数组 work，并对 i=1，2，3，…，q，置 work[i]=0，其中 q 为流处理器池中的 GPU 数量；

[0036] 3）从 CPOOL 中提取 q 个 DSUQ 以及每个队列所对应 GPU 的 KernelP，构成任务对 $(DSUQ_1, Kernel_1)$，$(DSUQ_2, Kernel_2)$，…，$(DSUQ_q, Kernel_q)$；

[0037] 4）对 i=1，2，…，q，分别加载 $(DSUQ_i, Kernel_i)$ 到 GPUi 执行步骤 5，其中 GPU_i 是指第 i 个 GPU；

[0038] 5）向 GPU_i 的存储器申请 $DSUQ_i$ 大小的存储单元 D_BUFF_i，然后将 $DSQU_i$ 的内容加载到 D_BUFF_i，然后再提交 Kerneli 及 D_BUFF_i 到 GPU_i 执行；

[0039] 6）监控所有 GPU 的执行状况，如果 GPU_i 执行完毕，则向计算后处理部件发送“GPU_i 数据流处理完毕”消息，并从 CPOOL 中提取下一个未被执行的任务对 $(DSUQ_i, Kernel_i)$ 后转至步骤 5；如果 CPOOL 中的所有 DSUQ 都被加载执行完毕，则向 DSU 聚类分配部件发送“数据流处理完毕”消息，并对所有的 i 置 work[i]=0，然后再转至步骤 1；

[0040] 计算后处理部件执行以下步骤：

[0041] 1）判别是否收到来自任务调度部件的“GPU_i 数据流处理完毕”消息，如果未收到则重复本步骤，反之则转至步骤 2；

[0042] 2）向内存申请 D_BUFF_i 大小空间的 POST_DSUQ，所述 POST_DSUQ 的结构与 DSUQ 的结构一致；

[0043] 3）先将 D_BUFF_i 的内容加载到 POST_DSUQ，再释放 D_BUFF_i 的空间；

[0044] 4)扫描 POST_DSUQ 中的每个 DSU，将 DSU 按照 MapM 记载的位置映射信息还原到相应的 DS 中，保持原有 DS 的顺序，并把结果写入 RS；

[0045] 5）转至步骤 1；

[0046] GPUi 上的 Kernel 执行以下步骤：

[0047] 1）获取 $Kernel_i$ 及 D_BUFF_i，并计算出 D_BUFF_i 中的 DSU 数量记为 g；

[0048] 2）在 GPU_i 的各个物理流处理单元分配 DSU，每个物理流处理单元得到$\lceil g/h \rceil$个 DSU，其中 h 为 GPU_i 的物理流处理单元数量；

[0049] 3）所有物理流处理单元并行地对其分配到的 DSU 执行 $Kernel_i$ 进行处理，并输出计算结果到其所处理的 DSU 的 odata；

[0050] 4）GPU_i 计算结束。

[0051] 本发明提供的大规模并发数据流处理系统及其处理方法，通过对并发数据流的流水分拣机制聚类数据流单元，构建了数据流单元聚类队列池来收集同类数据流单元，运用批调度机制加载数据流单元聚类队列到流处理器实施并行处理，运用多维标识机制聚类数据流单元，在处理结束后又可重组数据流单元的计算结果重构数据流，使得同一时刻被处理的数据集合具备同样的数据处理方法，大大发挥了流处理器 SPMD 计算模式的特长，使流处理器能发挥其最大能力，进而提高处理效率。

附图说明

[0052] 图 1 是本发明实施例的大规模并发数据流处理系统的结构示意图；

[0053] 图 2 是本发明实施例的大规模并发数据流处理系统的处理过程示意图。

具体实施方式

[0054] 以下结合附图说明对本发明的实施例作进一步详细描述，但本实施例并不用于限制本发明，凡是采用本发明的相似结构及其相似变化，均应列入本发明的保护范围。

[0055] 为了对本发明实施例的一种大规模并发数据流处理系统作进一步详细描述，本说明书作如下设定：

[0056] 设定 1（数据流单元），一个数据流单元是一个九元组 DSU（id，sno，segno，seq，t，type，prog，data，odata），其中 DSU 为数据流单元，id 为该 DSU 的标识符，且该 id 具有唯一性，sno 为该 DSU 的数据流号，segno 为该 DSU 的数据流段号，seq 为该 DSU 的在 segno 数据流段中的单元序号，用于表示其在数据流段中的位置，t 为一个时间印，用于记载该 DSU 被处理的时刻，type 为该 DSU 的类型，每个 DSU 根据其数据被加工的特点被划分成多个类型，data 为该 DSU 所承载的数据对象，data 可以是一个简单对象，也可以是由多个简单对象构成的复合对象，odata 为该 DSU 处理后的输出数据对象，prog 是该 DSU 的 data 的处理程序，prog 对 data 进行处理分析，并把输出结果写入 odata；

[0057] 设定 2(数据流段)，一个数据流段是由多个 seq 连续的 DSU 构成的序列，记为 DSS={DSU_1，DSU_2，DSU_3，…，DSU_n，DSUE}，其中 DSS 为数据流段，每个 DSS 均有一个数据流段号

segno 被分别存储在构成该 DSS 的每个 DSU 中，DSS 序列尾的 DSUE 为该 DSS 的结束标志，是一个 type 为常量值 EOS 的 DSU，其 prog、data、odata 均为空；

[0058] 设定 3（数据流），一个数据流是由多个 segno 连续的 DSS 构成的序列，记为 DS={DSS_1，DSS_2，DSS_3，…，DSS_o}，每个 DS 均有一个数据流号 sno 被分别存储在构成该 DS 的各个 DSS 的 DSU 中；

[0059] 设定 4（并发数据流），由多个并发传输的 DS 构成，每个 DS 均以 DSU 作为并发处理的单位，并以 DSS 作为多个数据流并发同步的单元。

[0060] 如图 1 所示，本发明实施例所提供的一种大规模并发数据流处理系统，其特征在于，包括：

[0061] 数据流单元缓冲区，是一个二元组 DSB(DSUB,MR)，其中 DSB 为数据流单元缓冲区，DSUB 及 MR 均是由 p 个元素构成的一维数组，p 为并发数据流中的数据流数量，DSUB 中的每个数组元素为一个 DSU，MR 中的每个数组元素是一个取值为 0 或 1 的整型数，该数组用于数据流流水处理的同步标志；

[0062] 数据流单元聚类队列池，由 |TS| 个 DSU 队列构成，记为 CPOOL={$DSUQ_1$，$DSUQ_2$，…，$DSUQ_{|TS|}$}，其中 CPOOL 为数据流单元聚类队列池，DSUQ 为数据流单元聚类队列，TS 为应用系统数据流单元类型集合，该集合是 DSU 类型的集合，TS 中的元素个数为 m，则 |TS|=m，同一个 DSU 队列由同类型的 DSU 构成，这些 DSU 来自 p 个并发数据流的当前处理单元，有：

$$\sum_{i=1}^{|TS|} |DSUQ_i| = p$$；

[0063] 数据流单元映射表，由多个表元构成，记为 MapM（nu，sno，segno，seq，t，qso，qoffset），其中 MapM 为数据流单元映射表，nu 为序号，sno 为数据流号，segno 为数据段号，seq 为数据流单元号，t 为时间印，qso 为聚类队列号，qoffset 为聚类队列内部元素位置号；

[0064] 流处理器池 SPP，由多个 GPU 构成，所述 GPU 为二元组 GPU（KernelP，D_BUFF），其中 KernelP 为该 GPU 当前执行 SPMD 任务的计算核心部件，D_BUFF 为 KernelP 执行 SPMD 操作的多个 DSU 集合；

[0065] 数据流读取部件 SReadP，用于读取数据流；

[0066] DSU 聚类分配部件 AllotP，用于对数据流单元缓冲区中当前被处理的数据流单元进行分类；

[0067] 任务调度部件 ExecP，用于将数据流单元聚类队列池中的就绪队列加载至流处理器池中的 GPU 上执行流计算；

[0068] 计算后处理部件 PostP，用于将 GPU 计算的 DSU 的 odata 按 MapM 的标志回归到 DSU 所在的数据流。

[0069] 如图 2 所示，本发明实施例所提供的大规模并发数据流处理系统的处理方法，其特征在于：

[0070] 数据流读取部件 SReadP 重复执行以下步骤直至并发数据流中的 DS 读取完毕：

[0071] 1）根据并发数据流的个数，在 DSB 中为每个 DS 分配一个单元，并初始化 DSB 的 MR，置 MR[i] 值为 0，其中 1 ≤ i ≤ p，p 为并发数据流的个数；

[0072] 2）读取并发数据流中所有 DS 的当前 DSS；

[0073] 3）扫描并发数据流，对 i=1，2，…，p，对 DS_i 做步骤 4 的处理，所述 DS_i 是指第 $_i$ 个 DS；

[0074] 4）如果 MR[i] 值为 1，则转至步骤 3 处理下一个 DS 的 DSU；

[0075] 如果 MR[i] 值为 0，则提取 DS_i 的当前处理 DSU，并判断当前处理 DSU 的 type，如果当前处理 DSU 的 type 值为 EOS，则 DS_i 的当前 DSS 结束，则置 MR[i] 为 1，并转至步骤 3 处理下一个 DS 的 DSU，反之则判断 DSUB[i] 是否为空，如 DSUB[i] 为空，则把当前处理 DSU 存入 DSUB[i]；

[0076] 5）如果 DSUB 的所有元素均置满数据，则等待至 DSUB 的所有元素都被 DSU 聚类分配部件 AllotP 置为空；

[0077] 6）如果 DSB 中的 MR 的所有元素都为 1，则转至步骤 1 处理并发数据流中所有 DS 的下一个 DSS，反之则转向步骤 2 继续处理当前 DSS 的 DSU；

[0078] DSU 聚类分配部件 AllotP 重复执行以下步骤：

[0079] 1）判别 DSB 的 DSUB 中是否置满数据，如果未满则重复本步骤，反之则转至步骤 2；

[0080] 2）判别是否收到来自任务调度部件 ExecP 的“数据流处理完毕”消息，如果未收到则重复本步骤，反之则转至步骤 3；

[0081] 3）对 i=1，2，…，p，分类处理 DSUB[i]，其分类处理步骤如下：

[0082] 如果 DSUB[i] 的 type 值不是 EOS，则将 DSUB[i] 加入 CPOOL 的第 w 个数据流聚类队列 $DSUQ_w$ 中，其中 w 值等于 DSUB[i] 的 type 值；然后获取 DSUB[i] 在 DSUQw 的位置下标，记为 pos，并置 MapM[i] 的 nu 值为 i，置 MapM[i] 的 sno 值为 i，置 MapM[i] 的 segno 值为 DSUB[i] 的 segno 值，置 MapM[i] 的 seq 值为 DSUB[i] 的 seq 值，置 MapM[i] 的 t 值为 DSUB[i] 的 t 值，置 MapM[i] 的 qso 值为 w 值，置 MapM[i] 的 qoffset 值为 pos，然后置 DSUB[i] 为空；

[0083] 4）向任务调度部件 ExecP 发送“数据流聚类队列构建完毕”消息；

[0084] 任务调度部件 ExecP 执行以下步骤：

[0085] 1）判别是否收到来自 DSU 聚类分配部件 AllotP 的“数据流聚类队列构建完毕”消息，如果未收到则重复本步骤，反之则转至步骤 2；

[0086] 2）为流处理器池 SPP 中的各个 GPU 配置一个工作标志数组 work，并对 i=1，2，3，…，q，置 work[i]=0，其中 q 为流处理器池 SPP 中的 GPU 数量；

[0087] 3）从 CPOOL 中提取 q 个 DSUQ 以及每个队列所对应 GPU 的 KernelP，构成任务对 $(DSUQ_1, Kernel_1)$，$(DSUQ_2, Kernel_2)$，…，$(DSUQ_q, Kernel_q)$；

[0088] 4）对 i=1，2，…，q，分别加载 $(DSUQ_i, Kernel_i)$ 到 GPU_i 执行步骤 5，其中 GPU_i 是指第 i 个 GPU；

[0089] 5）向 GPU_i 的存储器申请 $DSUQ_i$ 大小的存储单元 D_BUFF_i，然后将 $DSQU_i$ 的内容加载到 D_BUFF_i，然后再提交 $Kernel_i$ 及 D_BUFF_i 到 GPU_i 执行；

[0090] 6）监控所有 GPU 的执行状况，如果 GPU_i 执行完毕，则向计算后处理部件 PostP 发送“GPU_i 数据流处理完毕”消息，并从 CPOOL 中提取下一个未被执行的任务对 $(DSUQ_i, Kernel_i)$ 后转至步骤 5；如果 CPOOL 中的所有 DSUQ 都被加载执行完毕，则向 DSU 聚类分配部件 AllotP 发送“数据流处理完毕”消息，并对所有的 i 置 work[i]=0，然后再转至步骤 1；

[0091] 计算后处理部件 PostP 执行以下步骤：

[0092] 1）判别是否收到来自任务调度部件 ExecP 的“GPU_i 数据流处理完毕”消息，如果未

收到则重复本步骤，反之则转至步骤 2 ；

[0093] 2）向内存申请 D_BUFF_i 大小空间的 POST_DSUQ，所述 POST_DSUQ 的结构与 DSUQ 的结构一致 ；

[0094] 3）先将 D_BUFF_i 的内容加载到 POST_DSUQ，再释放 D_BUFF_i 的空间 ；

[0095] 4)扫描 POST_DSUQ 中的每个 DSU，将 DSU 按照 MapM 记载的位置映射信息还原到相应的 DS 中，保持原有 DS 的顺序，并把结果写入 RS ；

[0096] 5）转至步骤 1 ；

[0097] GPU_i 上的 Kernel 执行以下步骤 ：

[0098] 1）获取 $Kernel_i$ 及 D_BUFF_i，并计算出 D_BUFF_i 中的 DSU 数量记为 g ；

[0099] 2）在 GPU_i 的各个物理流处理单元分配 DSU，每个物理流处理单元得到$\lceil g/h \rceil$个 DSU，其中 h 为 GPU_i 的物理流处理单元数量 ；

[0100] 3）所有物理流处理单元并行地对其分配到的 DSU 执行 $Kernel_i$ 进行处理，并输出计算结果到其所处理的 DSU 的 odata ；

[0101] 4）GPU_i 计算结束。

[0102] 本发明实施例中所述的大规模并发数据流是指数据流数量在 1200 个以上的并发数据流，本发明实施例实际应用时也可用于处理数据流数量少于 1200 个的并发数据流。

[0103] 以下以一个简化的 3G 视频数据流中的 H.264 解码问题的预测阵计算实例来进一步说明本发明实施例的处理方法 ：

[0104] H264 把视频图像编码若干个 16X16 点阵规模的宏块，每个宏块有其编码方法，一个宏块相当一个 DSU，而一个视频帧相当于一个 DSS，所以一个 H264 视频流可以由多个 DSS 构成，一个 DSS 由多个 DSU 构成，实际上在 H264 的编码中，每个宏块又会根据图像实际情况分为多个 4X4 或 8X8 的子快，因而对于 H264 的 I 帧具有 21 种预测阵计算方法 ；为了简化描述，我们假设本应用中的宏块只是 16X16 宏块，因此示例模型的 H264 视频流是由多个 16X16 宏块构成，根据 H264 编码规则，16X16 宏块有 4 种预测阵处理模式，即只有 4 种类型的 DSU，因而 CPOOL 只有 4 个数据流单元聚类队列，也就是说只有 4 类 Kernel P 部件。

[0105] 实际应用中，利用本发明实施例构建了基于流处理器并行环境的 3G 网络质量监测系统，该系统由 2 个功能集群构成 ：9 台高性能计算机构成 I/O 任务密集型 CPU 集群，9 台流处理器机器构成计算密集型 GPU 集群，集群通过 2 台基板带宽为 48Gbps 的千兆以太网交换机连接而成 ；所有计算节点用 MPI 通信协议互联，每个计算节点配置 UBANTU10 操作系统，流处理器采用 NVIDIA GTX480，开发环境为 NVIDIA CUDA，每个 CPU 集群节点有一个与之对应的 GPU 集群节点，在这两个节点间有 1120 个视频流进行数据流传输，9 对计算节点可以处理 10080 个并发视频流，每对节点采用本发明实施例的系统实现，该集群运用 CPU 集群把 3G 视频流 H.264 参数提取，然后 GPU 节点对 10080 个视频图像参数进行计算（图像还原成 YUV、模糊度、块效应、平滑度分析），10080 个 3G 视频流大约有 20GB 的网络带宽，该系统对 10080 个视频流的 10080 个当前 I 帧的分析处理单位平均时间为 1.5 秒，能满足电信对 3G 视频的分析规模的要求和性能的要求。

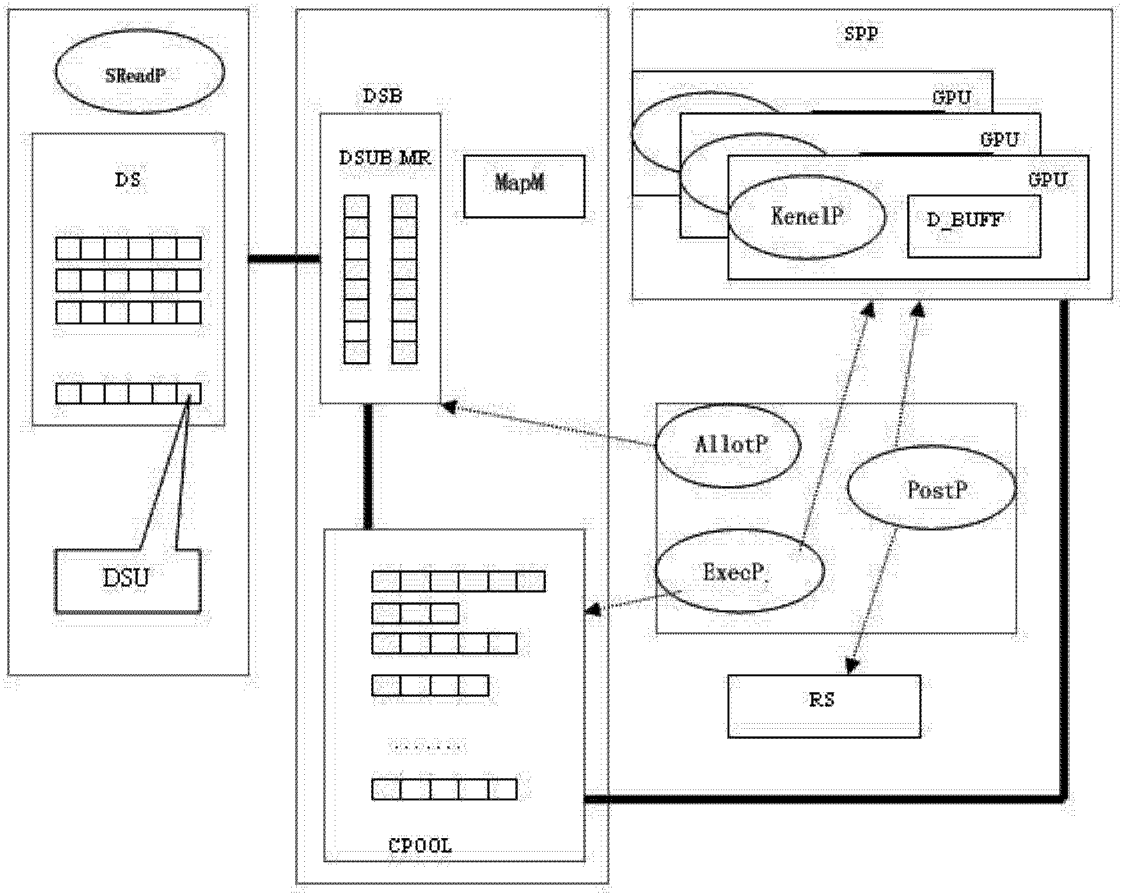

SReadP
DS
DSU
DSB
DSUB MR
MapM
CPOOL
SPP
GPU
GPU
GPU
KeneIP
D_BUFF
AllotP
PostP
ExecP
RS

图 1

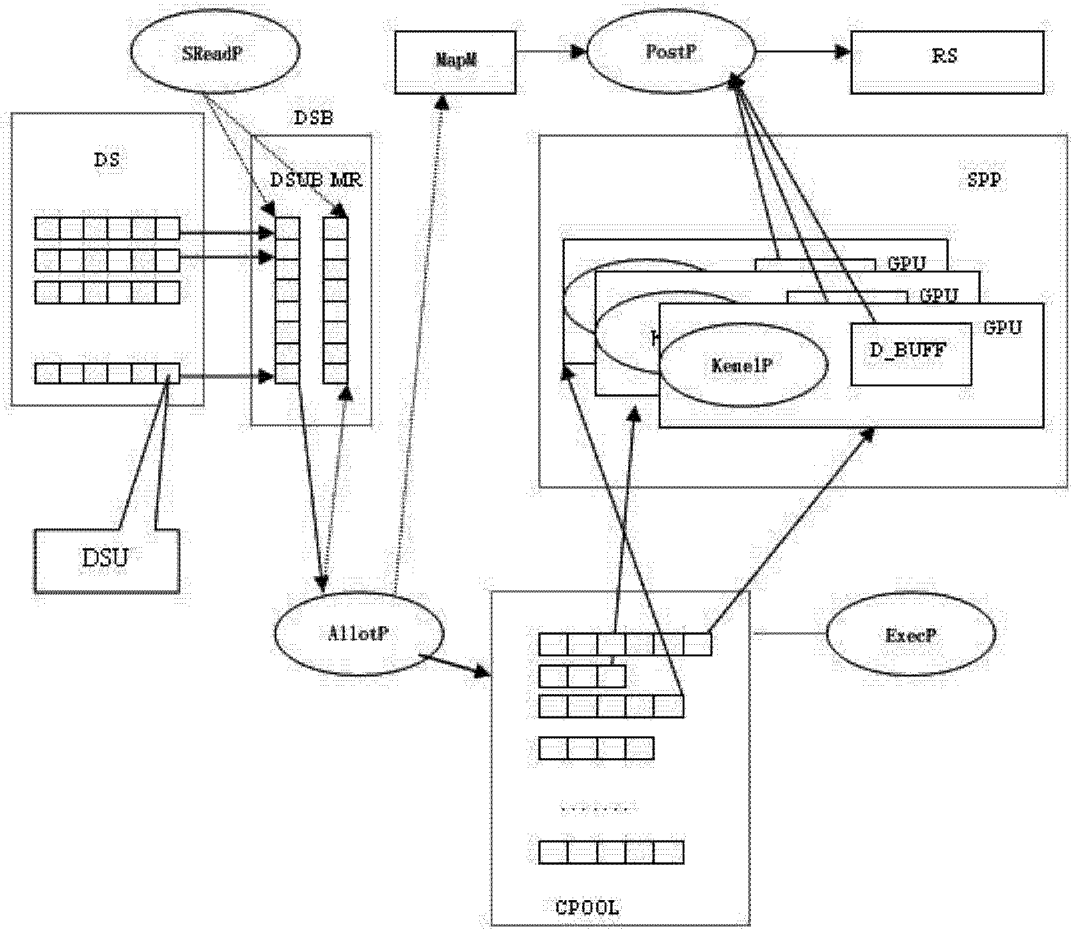

SReadP
MapM
PostP
RS
DSB
DS
DSUB MR
SPP
GPU
GPU
GPU
KernelP
D_BUFF
DSU
AllotP
ExecP
CPOOL

图 2