基于国微DSP的大数量排序算法设计与实现

王文青　张　超　张　涛
深圳市国微电子有限公司　广东　深圳　518100

摘　要：排序算法是工程和算法研究中的基础问题，大数量的排序算法运算量大、耗时长。通用DSP（数字信号处理）芯片虽然具备非常快速的数据处理能力，但是受限于DSP系统结构、与算法的适配等原因，算力无法完全用于大数量排序。国微DSP是一款完全自主知识产权的异构多核高性能芯片，本文结合国微DSP中异构多核的架构特征，充分发挥DSP核和CPU核的协同优势。通过实测，相较于应用广泛的高性能DSP芯片TMS320C6678，基于国微DSP设计与实现的大数量排序算法，具备更优良的效率，提升两倍以上。

关键词：DSP；高性能；异构多核；大数量排序

1 引言

排序是算法研究中的一个基础问题。

很多算法通常把排序作为关键子程序。例如，在一些基于图理论（graph）的图像分割算法中，需要对建立的边数据根据其权重进行排序[1]。

现有的排序算法数量非常庞大，其中所使用的技术也非常丰富，在不同的算力平台下有不同的解决方案。

单纯的DSP芯片具备快速的数据处理能力，但是一次性能处理的数据有限，不适合处理大量的数据排序[2]。

国微DSP融合双核CPU主处理器和双核DSP协处理器，最高浮点运算性能可达512GFLOPS，具有20MB大容量片上存储和丰富的IO接口。

本文提出一种快速的通用排序算法，针对不同的卫星数据（satellite data），采用统一的索引（INDEX），屏蔽卫星数据差异。并基于国微DSP提供的异构算力，结合CPU+DSP不同的应用特点，针对不同算力单元设计其适配的算法，最终实现了高效的大数量排序算法。

2 改进的排序算法设计

排序问题一般描述为：

输入：n个数的一个序列$<a_1,a_2,\ldots,a_n>$。

输出：输入序列的一个排列（重排）$<a_1',a_2',\ldots,a_n'>$，使得$a_1' \leqslant a_2' \leqslant \ldots \leqslant a_n'$。

通常的排序算法很少是独立的值，都是由关键字（KEY）和卫星数据（satellite data）构成，关键字和卫星数据一一对应。其中，关键字就是排序问题中要重排的值（上述排序描述中的序列），而卫星数据则需要跟随关键字进行重排。不同的应用场景下，卫星数据的数据量和内容是不定的[3][4]。

本文提出一种通用的排序算法实现，提出关键字和索引结合的方式，为不同应用场景下的排序需求提供解决方案。并结合CPU+DSP的不同算力，以异构计算的方式，在国微DSP上实现了一种通用快速的排序方案。

算法设计数据如下：待排序数据为KEY+INDEX，其中索引数据不限制具体意义，针对通用的卫星数据，一般意义上为关键字对应的卫星数据所对应的id号（关键字为待比较数据）。排序后满足，根据KEY排序结果，重组织整数索引INDEX的值，根据INDEX索引值，重组卫星数据。

3 基于国微 DSP 的排序算法设计

3.1 国微DSP芯片简介

国微DSP为融合双核CPU主处理器和双核DSP协处理器的高性能异构智能计算SoC，最高性能可达512GFLOPS，具有20MB大容量片上存储器和丰富的IO接口，可支持数字信号处理、图像处理、网络安全与隐私计算等多种类型的应用。

国微DSP的计算任务主要由DSP协处理器来完成，具有以下特性：

1）采用VLIW/SIMD/MIMD多维度并行架构；

2）每个内核内部集成96KB指令Cache；

3）每个内核包含16个256位SIMD向量处理单元，支持INT8/16/32/64以及FP16/32数据类型；

4）每个内核内部集成2MB Local SRAM；

5）每个内核内部集成DSP专用DMA引擎，实现主动式高效数据搬移；

6）1GHz工作频率下，每个内核峰值计算能力256GFLOPS。

3.2 算法整体结构设计

单纯的DSP因为硬件结构和存储空间等限制，每次能进行排序的数据是有限的，因此大数量排序算法部署在DSP上效率比较低下。本文充分发挥国微DSP的异构架

构，综合CPU处理器的灵活调度和DSP处理器的高性能算力，协同完成排序算法[5]。

图1 国微DSP结构框图

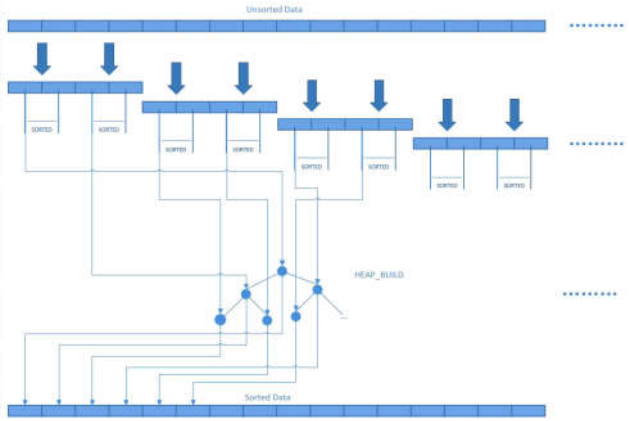

图2 排序算法流程示意图

如图2所示，排序算法分为两个步骤：

（1）双核DSP并行对局部数据进行排序，每次产生2组固定长度的有序子序列。

（2）CPU对DSP产生的有序序列建立最小堆（HEAP），二次归并排序，最终产生完整的排序序列。

因为算力的独立性，CPU归并和DSP排序之间独立进行，在时序上互相覆盖，进一步提升排序效率。

3.3 算法模块设计

3.3.1 DSP排序算法设计

排序过程的计算开销较大。一方面由于边的总数较大，另一方面边排序不是简单的数组排序，需要保持边的索引值INDEX同时调整位置。为此本文结合该排序算法特点和DSP计算结构体系进行专门优化，边排序操作由DSP和CPU协同完成，最后的归并操作不适合并行化由CPU完成。

定义VPE0所对应的bank0中边排序待排数据的数据结构为{w,a}。其中w代表排序算法的关键字KEY，a为索引INDEX。待排数据{w,a}划分成2个分区独立存储。

将w，a分开存储有以下优点：首先，边排序是对w进行排序，索引值a只是需要根据w的大小做相同的数据移动就可以，一个vpe可以同时进行8个float数据对的比较操作，16个vpe可以在一个周期完成128个比较操作，如果将w,a打包存储就会浪费一点的比较资源。其次，DSP内部具有相关的指令可以根据需求可以实现一些逻辑运算，边排序的过程需要对w进行比较且需要保留比较结果，用来判断a是否需要进行数据对换。

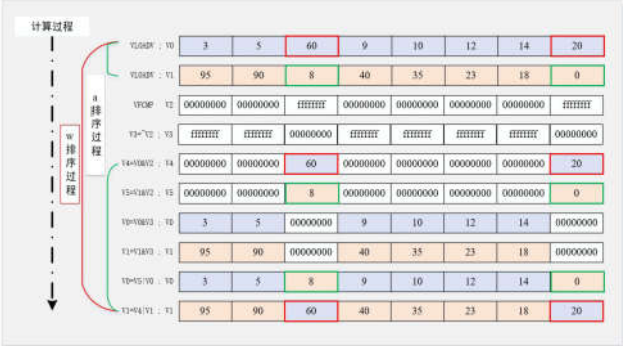

图3 排序处理过程示意图

图3选取了两组数据展示了最基本的排序处理过程，图中V0、V1等代表向量寄存器，用来存放输入，输出和中间结果，本示意图左边VLOADV、VFCMP等伪指令描述了该行的处理行为。整个排序操作按照列进行排序，一个VPE可以同时进行8组数据的的排序，一个排序算法的最小单元步骤如图4（左）所示。其中V0~V5为不同VPE代表向量寄存器。

至此，在一个vpe中完成了8对数的比较操作且，完成了大数与小数的交换完成了一轮排序操作。图3中，红色括号为权值w的排序流程，两个绿色括号为索引值INDEX的排序流程，可以直接使用V2、V3中关于这一轮w的比较标志位，对INDEX做相同的处理，就完成了一个最小单位的排序操作。

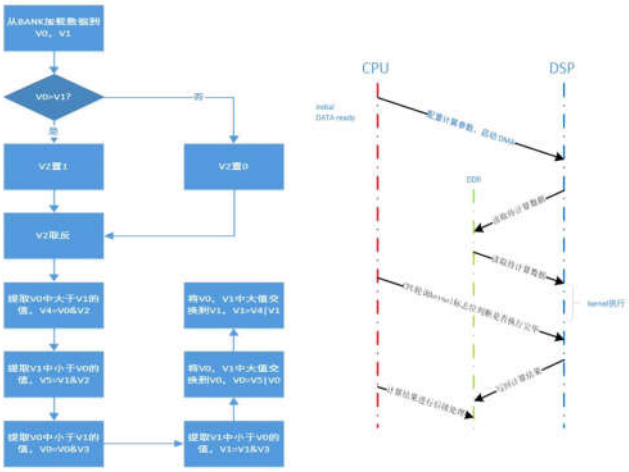

图4 最小单位排序流程图（左）DSP和CPU协同处理流程图（右）

CPU和DSP之间通过集成于DSP内部的DMA模块和以及一些寄存器进行通信。图4（右）展示了DSP和CPU协同处理流程图。以边排序计算任务为例，DSP通过一个主机和一个从机接口接入系统。首先CPU通过AXI4的master接口下发边排序任务的相关参数到DSP中，配置DSP内部DMA搬移运算数据到内部SRAM，最高可载入1024*128个{w,a}数据，输出128个1024短序顺序序列，然后启动运算任务，软件轮询DSP是否执行完毕，执行完毕后CPU配置DSP内部DMA将计算结果搬出，然后执行下一块任排序任务。

DSP最终输出为有序子数据对。每个数据对由一个关键字KEY和一个索引值INDEX构成，定义为<KEY，INDEX >。该数据在内存内排列为一个有序数组，根据KEY值由小到大排列。然后CPU对DSP输出的短序列进行归并排序。

3.3.2 CPU归并排序算法设计

CPU算力负责对有序的子序列进行二次排序，排序算法采用堆（HEAP）排序+归并（Merge）相结合的方法[6]。

算法大致可以分为两个步骤：

（1）以每个子序列首值为索引建立最小堆（MIN-HEAP）。该堆（HEAP）满足如下性质，根节点数值小于等于子节点数值。定义单个堆节点数据结构为Node，Node定义为<data* ,idx, len>。其中data*为指向有序子序列的首地址，idx为当前Node所指向的data数据位置，len为该有序子序列的长度，最小堆建立方法为经典方法[6]。

（2）最小堆（MIN-HEAP）建立完毕后，依次对堆顶数据进行归并输出。首先输出堆顶Node所指向的data[idx]数值，此值为当前最堆内最小值。随后idx加1，并根据当前data[idx]值更新最小堆，保证堆顶Node所指向的data[idx]为堆内最小值，持续维护最小堆性质[6]。当某个Node内idx等于len时，即该Node内所有数据皆已经输出完毕，该Node出堆。重复该过程，直到堆内所有Node皆出堆，则该最小堆输出完毕。

（3）此时INDEX已经根据关键字key重排序，根据INDEX可以直接索引到内存中原始卫星数据（satellite data）位置。根据具体应用场景，可以获得最大，最小，或任意分位关键字KEY对应的卫星数据。或者顺序遍历所有INDEX，获得重排序后的卫星数据。

4 验证与结果分析

为验证本文算法的正确性，我们将随机生成的KEY值数据输入上述算法模块，检测排序后数据单调性。同时，记录处理时间，分析比较算法效率。

4.1 实测数据处理结果

如图5（左）所示，随机生成4,194,304个32位整形KEY数据，均匀分布在经过该算法处理后，KEY数据排列如图5（右）所示，呈现单调性，证明了本文采用的架构的正确性和有效性。

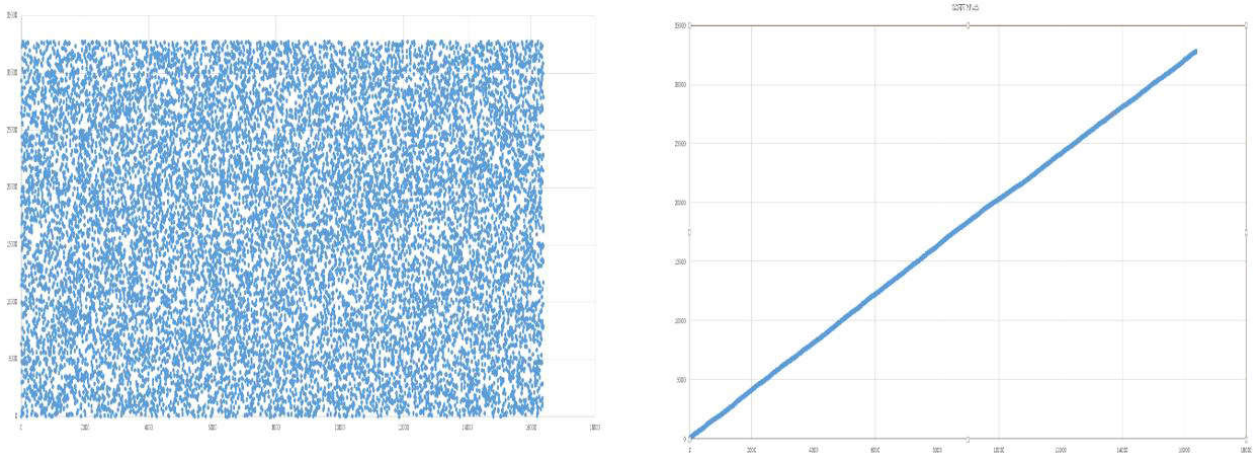

图5 排序前数据呈现随机性（左）排序后数据呈现单调性（右）

4.2 处理时间分析

表1

数据量	本文系统耗时（s）	TMS320C6678耗时（s）	性能提升比例（%）
1,048,576	0.15s	0.38s	253%
2,097,152	0.35s	0.80s	228%
4,194,304	0.8s	1.69s	211%

表1是在不同数据量下测得的排序算法实际耗时。可以看到，在不同尺寸下，相比在单纯算力的TMS320C6678上采用的标准快速排序算法，本文系统所设计的算法都有明显的效率优势，提升均在两倍以上，效果显著。

5 结论

本文设计了一种基于异构计算架构（CPU+DSP）的大数量通用排序算法。本算法与卫星数据（satellite data）差异，利用DSP协处理器的高性能智能计算加速和CPU主处理器的冗余算力和灵活控制，同步对数据进行局部排序和归并排序，兼顾了大数量排序的可行性和实时性。与单纯算力平台的排序算法相比，显著提升了效率，为ARM+DSP计算架构的运算平台，提供了一套切实可行的通用算法方案。

参考文献

[1]侯叶.基于图论的图像分割技术研究[D].西安电子科技大学,2011.

[2]马晓东,李冰琪,魏鹏,等.DSP技术发展与应用研究综述[J].电子世界,2018,(24):46-47.DOI:10.19353/j.cnki.dzsj.2018.24.021.

[3]算法导论[M].(美)科曼(Cormen,T.H.)等,著.机械工业出版社.2006

[4]吴光生,范德斌.排序算法研究[J].软件导刊,2007,(07):97-98.

[5]朱怀宇,冯雪,姜群兴.一种基于FPGA、DSP和ARM的异构运算构架及实现方案[J].工业控制计算机,2019,32(11):20-21.

[6]杜双敏.堆排序的构造方法探究[J].电脑知识与技术,2020,16(27):67-69.DOI:10.14004/j.cnki.ckt.2020.2876.