

引用格式:包冲,张善从. 面向 GPU 的通用矩阵乘法计算的容错研究[J]. 微电子学与计算机,2021,38(1):22-26. [BAO C, ZHANG S C. Research on the Fault-tolerant of GEMM computation for GPU[J]. Microelectronics & Computer,2021,38(1): 22-26.]

面向 GPU 的通用矩阵乘法计算的容错研究

包 冲¹,张善从²

(1 中国科学院大学,北京 100049; 2 中国科学院 空间应用工程与技术中心,北京 100094)

摘 要: 矩阵计算是 GPU 最擅长的工作之一,NVIDIA 公司在 CUDA 中提供了线性代数库 cuBLAS,用于矩阵和向量相关的计算.但是 GPU 容易受到电磁或者宇宙射线影响,而发生“位”反转问题,从而发生静默数据损坏错误.针对这个问题,利用基于算法的容错方法,提出了带容错的,用于通用矩阵乘法计算的方法,并以 CUDA 库函数的方式实现.论文讨论了算法的原理,用一种高效的方法实现了容错计算,并提出了一个低开销、高准确率阈值计算方法用于在线的快速纠错和检错.在两款嵌入式 GPU 平台上对带容错功能的 GEMM 库函数进行了评估,其纠错和检错能力与预期一致,并且在大部分情况下,额外性能开销能够控制在 50% 以内,证明了该 GEMM 函数可以在较低的性能开销情况下,能够很好的实现 GEMM 计算的检错和纠错,在某些结果一关键的高性能计算中,具有一定的实用价值.

关键词: 通用矩阵乘法;基于算法的容错;静默数据损坏;浮点运算;校验和;检错与纠错

中图分类号: TN911-34;TP312

文献标识码: A

文章编号: 1000-7180(2021)01-0022-05

Research on the Fault-tolerant of GEMM computation for GPU

BAO Chong¹,ZHANG Shan-cong²

(1 University of Chinese Academy of Sciences, Beijing 100049, China;

2 Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China)

Abstract: Matrix computing is one of the work that GPUs are good at. NVIDIA provides a linear algebra library cuBLAS, in CUDA for the calculations related with matrices and vectors. However, GPU is vulnerable to the problem of bit inversion due to electromagnetic interference or cosmic rays, resulting in silent data corruption (SDC) errors. In order to resolve this problem, a general CUDA library function with fault tolerance for General Matrix Multiplication (GEMM) is implemented by using the algorithm-based fault-tolerant method (ABFT). The principle and implementation of the algorithm, as well as the judging mechanism in the process of error correction and detection are fully discussed. The GEMM library function with fault tolerance is evaluated on two embedded GPU platforms, and its error correction and error detection capabilities are consistent with expectations, and the additional performance overhead can be controlled within 50%. It is proved that this GEMM function with fault-tolerant can well perform detection and correction of the GEMM computing with lower performance overhead. In some result-key high performance computing applications, this function has more practical application value.

Key words: General Matrix Multiplication (GEMM); Algorithm Based Fault Tolerant (ABFT); Silent Data Corruption (SDC); Floating Point Calculation; Checksum; Error Detection and Correction

1 引言

由于其超强的浮点计算能力, GPU 已经成为高性能计算的首选平台之一. 随着 NVIDIA 公司嵌入式 GPU 的推出, GPU 开始应用于各种复杂环境(如高电磁或高宇宙射线辐射环境), 由于 GPU 本身集成度高, 工作频率高, VLSI 特征尺寸小, 内核电压低等特点, 使其相比于其它电子元器件更容易受到“位”翻转影响, 比较典型的就是空间环境中的 SEU 问题^[1], 这使得 GPU 的容错研究成为一个热点^[2]. 静默数据损坏错误是 GPU 受“位”翻转影响而最容易发生的错误之一, 这种错误不易被察觉, 直接导致最终计算结果错误, 在某些关键应用中, 可能会造成严重的后果. 而浮点矩阵运算, 则是很多算法的基础, 也是 GPU 最常实现的计算类型. 本文研究了一种基于算法容错的 GEMM 计算的方法, 并以 CUDA 库函数的方式实现, 可以为某些“计算结果-关键”的应用提供了一种计算结果具有更高可信度的基础函数.

2 相关技术

2.1 技术背景

GPU 是一种辅助处理器, 一方面可以分担 CPU 高性能计算的工作, 另一方面还需要 CPU 对其进行管理. 每次 GPU 运行所需的程序和数据, 都由 CPU 进行管理. GPU 具有非常复杂的内存结构, 包括集成在芯片内部的 cache 存储器、寄存器等, 以及外部的 DDR 存储器. 这些存储单元和计算单元(如 ALU 单元)都容易发生 SDC 错误.

嵌入式 GPU 采用了 SOC 的方式, 将多个 ARM 内核和 GPU 内核集成在一个芯片中, 使其应用场景更为广泛, 其与传统 GPU 不同之处在于, GPU 内核与 CPU 内核共享外部 DDR 存储器, 所以没有数据搬移的过程.

NVIDIA 公司为方便 GPU 开发而推出的 CUDA 编程模型, 为 GPU 提供了一个功能齐全的软件开发生态系统, 其中包括编译器、调试器和库. CUDA C/C++ 程序由不同的代码段组成, 这些代码段要么在主机系统的 CPU 上执行, 要么在 GPU 上执行, 其中, 在 GPU 上执行的代码段称为内核函数.

CUDA 提供的 cuBLAS 库是在 CUDA 运行时 API 之上的基本线性代数计算的实现, 包括矩阵和向量相关的各种计算. CUDA 库的使用, 可以减少

用户设计的工作量, 并给用户的程序带来非常大的性能提升. 但是这些库并没有容错设计, 在某些恶劣环境应用中, 其计算结果的可信性会存在问题, 如图 1 所示.

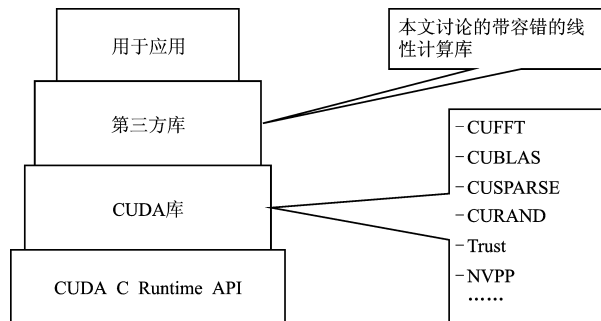


图1 CUDA 库在 CUDA 中的位置

2.2 相关研究

目前, 通过中子束测试实验已经证明, 辐射对 GPU 的影响主要表现在 SDC 错误或功能中断(调度器错误)^[3]. NVIDIA 也意识到了 SDC 错误对 GPU 的严重影响, 在 Fermi 架构的 GPU 上对关键存储单元增加了 ECC 的保护. 不过多个实验研究证明, ABFT 机制在增加 GPU 数据的恢复性方面, 比 ECC 更有效, 此外, ECC 功能也会严重的降低处理器的计算能力^[4].

GPU 容错问题的研究主要集中在两个方面: 一是, 针对 GPGPU 系统的检查点-重启方案的研究^[4-6]等. 这类研究主要针对大规模, 非实时性的并行计算, 一般不考虑性能开销; 二是, 针对 ABFT 的研究, 其中, 利用校验和的方法来实现矩阵计算的容错, 是基于算法容错的基本思想^[7]. 针对 GPU 的 ABFT 容错的在线检错和纠错, 也是一个研究方向, 主要考虑如何降低性能开销^[8], 如何解决浮点计算舍入误差导致的错误检测不准确的问题^[9].

3 容错算法

3.1 AFBT 容错原理

本文主要依据文献^[7]提出的用于矩阵计算的容错方法来展开讨论, 其算法原理如图 2 所示.

对于矩阵 A, B 和 C , 其大小分别为 $m \times k, k \times n$ 和 $m \times n$, 且有 $C = A \times B$. 通过增加矩阵 A 的列校验和向量 X_{A_c} , 以及矩阵 B 的行校验和向量 X_{B_r} , 可以完成一次带校验和的矩阵计算 $C' = A' \times B' \quad C' = A' \times B'$. 通过将结果矩阵 C 的列校验和 X_{C_c} 与行校验和 X_{C_r} , 与 C' 中计算生成的相应校验和向量 X'_{C_c} 和 X'_{C_r} 进行比较, 根据差异元素的情况, 就可以检测

和纠正结果矩阵 C 中的错误。

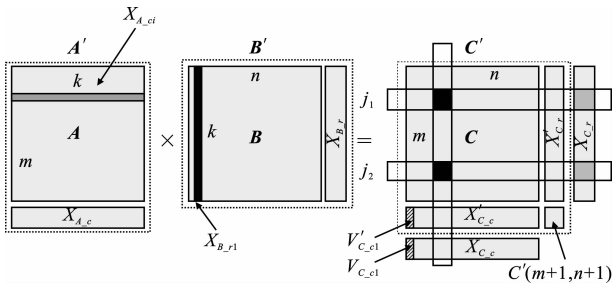


图 2 基于算法的 GEMM 容错原理
(示例两个错误的情况)

设两个列校验和向量存在差异的元素个数为 s , 差异元素的坐标为 $i_1, i_2 \cdots i_s$, 两个行校验和向量存在差异的元素个数为 t , 差异元素坐标为 $j_1, j_2 \cdots j_t$. 表 1 列出了该算法在不同情况下的检错和纠错能力。

表 1 ABFT 算法的检错和纠错能力

序号	情况	错误检测	错误纠正
1	$s = 0$ 且 $t = 0$	无错误	无需纠错
2	$s = 1$ 且 $t = 0$ 或 $s = 0$ 且 $t = 1$	校验和向量错误	无需纠错
3	$s = 1$ 且 $t \geq 1$ 或 $s \geq 1$ 且 $t = 1$	结果矩阵 C 中出现错误的元素为 $C(i, j_1), C(i, j_2) \cdots C(i, j_t)$.	根据式(1) 进行纠错
4	$s \geq 1$ 且 $t \geq 1$	错误元素为 $C(i_m, j_n)$, 其中 $M \in \{1, 2 \cdots s\}, n \in \{1, 2 \cdots t\}$	错误无法被 纠正

在第 3 种情况下, 如果 $s = 1$ 且 $t \geq 1$, 则, 对于其中的某个错误元素 $C(i, j_n)$, 其纠正值为:

$$C_{\text{correct}}(i, j_n) = C(i, j_n) + X'_{C,r}(j_n) - X_{C,r}(j_n) \quad n \in \{1, 2 \cdots t\} \quad (1)$$

3.2 浮点计算误差

在 ABFT 方法中, 两组校验和向量的比较是必不可少的. 在图 2 中, 以校验和向量 $X'_{C,c}$ 与 $X_{C,c}$ 的第一个元素 $V'_{C,cl}$ 与 $V_{C,cl}$ 基于 1 的坐标系) 的计算过程为例进行分析, 有:

$$V'_{C,cl} = \sum_{i=1}^m [X_{A,ci} \cdot X_{B,r1}] \quad (2)$$

$$V_{C,cl} = \left[\left(\sum_{i=1}^m X_{A,ci} \right) \cdot X_{B,r1} \right] \quad (3)$$

式中, $[a \cdot b]$ 表示向量 a 和 b 的内积.

$$-1^{\text{sign}} \times 1.\text{Mantissa} \times 2^{\text{Exponent}-127}$$

Sign(1 bit) 符号域	Exponent(8 bit) 指数域	Mantissa(23 bit) 尾数域
--------------------	------------------------	-------------------------

图 3 单精度浮点数的表示方法

可以看出, ABFT 算法本质上通过异构(不同计算顺序)的冗余计算来实现的. 由于浮点表示的特

点(参见图 3), 在浮点运算中会因为截断而产生计算误差, 而不同的计算顺序, 必然造成不同的计算误差. 所以对于两组校验和向量的比较, 要区分出“误差”和“错误”, 就需要确定一个阈值 ϵ , 满足:

$$\text{判断结果} = \begin{cases} \text{误差, } |V'_{C,ci} - V_{C,ci}| \leq \epsilon \\ \text{错误, } |V'_{C,ci} - V_{C,ci}| > \epsilon \end{cases} \quad (4)$$

4 函数的实现

4.1 计算流程及分工

带容错的 GEMM 计算流程如图 4 所示. 其主要计算过程在 GPU 中完成, 除 kernel-6 和 kernel-7 外, 其余的内核函数中均使用 cuBLAS 库中相应的函数来完成, 这样可以最大程度提升计算性能。

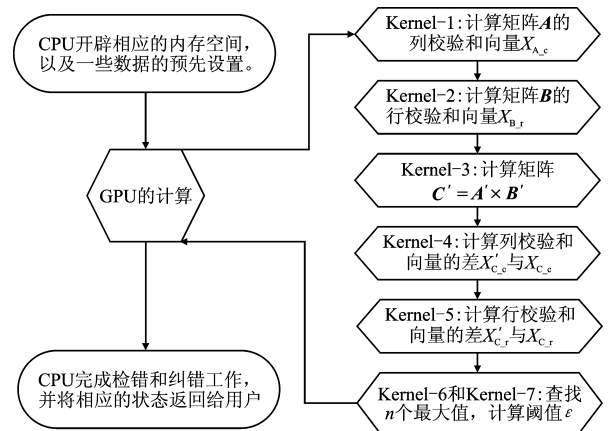


图 4 带容错的 GEMM 计算的过程

与 cuBLAS 库函数的调用不同的是, 带容错的函数中, CPU 需要承担两部分额外工作: 一是, 开辟一些内存空间, 用于校验和相关的计算, 这是用户不需要了解的, 所以由函数来完成; 二是, 一部分的检错和纠错工作, 由于需要较多的逻辑判断和分支跳转, 这并不是 GPU 所擅长的, 所以由 CPU 来完成。

4.2 主计算流程的实现

由于 NVIDIA 的 GPU 的浮点运算是加-乘融合的^[10], 所以单纯的加法计算与加-乘计算的耗时是一样的, 所以我们使用 cuBLAS 函数 cublasSgbmv() 来完成矩阵 A 和 B 的校验和计算(kernel-1 和 kernel-2), 矩阵 C 的校验和的计算, 包括校验和向量的相减计算(kernel-4 和 kernel-5)。

为此, 我们定义两个常量向量 $X_{A,o}$ 和 $X_{B,o}$, 长度分别为 $m+1$ 和 $n+1$, 两个向量除最后一个元素为 -1 外, 其余元素均为 1. 用这两个向量的前 m 或 n 个元素构成的单位向量, 分别与矩阵 A 和矩阵 B 相乘(前者右乘矩阵 A 的转置, 后者右乘矩阵 B), 得到两

者的校验和向量 X_{A_c} 和 X_{B_r} . 在存储过程中, 矩阵 A 按转置方式进行存储, 矩阵 B 按非转置方式进行存储, 两者均为列优先方式存储, 这样校验和向量 X_{A_c} 和 X_{B_r} 可以直接存储在矩阵 A 和 B 的后面, 和后者一起构成矩阵 $A'T$ 和矩阵 B' , 而不需要进行内存搬移, 并直接使用 cuBLAS 函数 `cublasSgbmm()` 来完成 $C' = A' \times B'$ 的计算(kernel-3).

得到带校验和的结果矩阵 C' 后, 将其分别与向量 X_{A_o} 和 X_{B_o} 进行相乘(需要进行必要的转置, 不再赘述) 就可以得到两组校验和向量的两个差值向量:

$$\begin{aligned} X_{\text{differ}_c} &= X'_{C_c} - X_{C_c} \\ X_{\text{differ}_r} &= X'_{C_r} - X_{C_r} \end{aligned} \quad (5)$$

这样, 通过 5 个库函数就可以快速实现主计算流程. 其中, $C' = A' \times B'$ 的计算与 $C = A \times B$ 的计算, 相差不大, 而额外的四个矩阵与向量的乘法计算, 和矩阵与和矩阵的计算相比, 性能开销也相对较小.

4.3 纠错和检错功能的实现

对 GEMM 函数计算结果进行纠错和检错的工作是由 GPU 和 CPU 共同完成的.

4.3.1 GPU 实现的功能

GPU 主要负责确定合适的阈值 ϵ (kernel-6 和 kernel-7), 这部分计算可以并行实现.

通过观察式(2)和式(3), 可知, 校验和向量中每个元素的计算误差依赖于矩阵 A 或矩阵 B 中的所有元素, 而不是某一个或某一些元素, 因此, 每个结果元素的计算误差可以认为是具有等概率的独立事件. 所以, 我们假设, 在无错误情况下, 两个差值向量的数值分布在以 0 为中心(理想状态下, 两个校验和向量的结果应该是一致的) 的一定范围内($\pm \epsilon$ 之间), 这是一种典型的正态分布, 即 $X_{\text{differ}} \sim N(\mu, \sigma^2)$. 我们通过生成各种输入数据来对结果进行观察, 也证实了这种假设.

两个内核函数首先需要分别查找两个差值向量中绝对值最大的前 n 个数, 这 n 个数是最有可能发生错误的. 考虑到自然环境中, “位” 反转错误发生的概率非常低, 所以我们认为在一次矩阵计算中发生错误的数量是有限的, 可以根据具体应用环境, 来设定 n 的数值.

在计算差值向量的均值 μ 和均方差 σ 之前, 要剔除这 n 个最大值, 防止这些可能的错误值影响均值和均方差计算结果的正确性. 我们将均值 μ 直接设置为 0, 不再进行计算, 这样既合理又高效(实际上,

由于样本空间有限, 均值 μ 不一定等于 0). 方差 σ 的计算则通过 GPU 并行计算来完成. 对于阈值, 我们设定如下:

$$\epsilon = K \times \sigma \quad (6)$$

根据正态分布的特点, 校验和向量的差值落在左右 5 个均方差范围外的概率为 10^{-11} 级别. 可以认为, 这种情况下出现的差值肯定代表了错误而不是误差, 所以我们取 $K=5$. 此外由于两个差值向量具有相关性, 可以只计算一个差值向量的阈值, 而用于对两个差值向量的比较判断.

4.3.2 CPU 实现的功能

在 GPU 计算完成后, CPU 根据 GPU 的计算得到的阈值 ϵ 对查找出的 n 个最大值元素进行检查, 来判断是否发生了错误, 并结合这些元素的原始坐标, 根据表 1 中描述的不同情况分别进行处理: 出现情况 3 时, 通过向量中错误元素的坐标, 得到结果矩阵 C 中的错误坐标, 并按照公式(1), 对相应的错误进行纠正. 出现情况 4 时, 由于计算结果出现的错误无法被纠正, 则 CPU 可以采用“检查点-重启”的方法, 重新进行计算.

5 评估

5.1 有效性评估

在图 4 所示流程中, 在内核函数调用的间隙, 在调试模式下对 GPU 全局内存的数值进行修改置错, 来检测我们容错方法的有效性, 实际测得的纠错和检错能力与表 1 的描述一致.

针对阈值设置的有效性, 我们利用 MATLAB, 分别为矩阵 A 和矩阵 A 的不同行或者列, 生成了符合多种分布形态(均匀分布, 泊松分布, 指数分布, 正态分布), 多种数值范围的数据, 用于测试. 最终证明, 用 4.3.1 节所描述的方法来设置阈值, 在置错和不置错的情况下, 都没有发现虚警和漏警的情况. 证明了这种在线快速阈值设置的方法是有效的.

5.2 性能评估

我们采用了 NVIDIA 公司最新的 SOC 型 GPU, TX2 模块和 Xavier 模块进行测试. 这两款 GPU 在内核数量和性能上都存在较大的差异, 具有一定的比较意义.

在没有错误的情况下, 我们主要对比了 cuBLAS 库中的 `cublasSgemm` 库函数与本文所讨论的带容错的 GEMM 函数的运行时间. 为了简化测试过程, 我们仅使用 $m \times m$ 的方阵进行测试, m 的大小从 64 到 8192 (以 2 的整数幂次方递进). 使用

CUDA 中的 Event API 函数,对被测程序采用多次循环,求平均值的测量方法,进行评估.

带容错的 GEMM 函数的额外性能开销定义为:

$$P = \frac{t_{\text{ABFT}} - t_{\text{naive}}}{t_{\text{naive}}} \quad (7)$$

式中, t_{ABFT} 和 t_{naive} 分别代表带容错的 GEMM 函数和原始 GEMM 函数的运行时间.

在两款 GPU 上测得的带容错的 GEMM 函数的额外性能开销如图 5 所示. 可以看出,总体上,额外开销随着矩阵尺寸的减小而增加,这在 TX2 上表现更为明显. 这主要是因为 cuBLAS 的矩阵计算使用的线程块大小为 $256 \times 1 \times 1$,当矩阵尺寸小于 256 时,计算的并行性变差,同时内核程序加载时间相对不变(有多达 7 个内核程序需要加载),从而导致额外性能开销的增加.

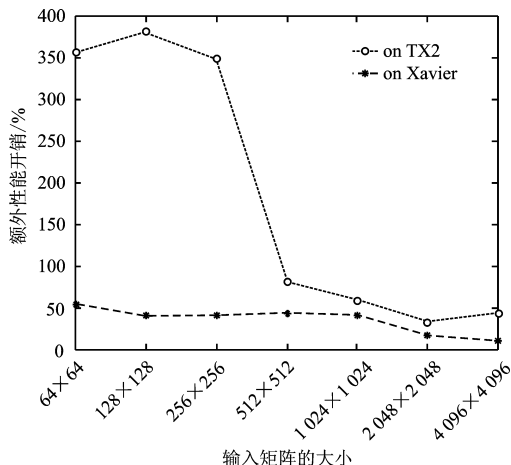


图 5 带容错功能的 GEMM 函数的额外性能开销

此外, Xavier 的表现明显要优于 TX2, 其额外性能开销能够控制在 50% 以下, 在矩阵尺寸较大时, 能够降低到 20% 左右. 这是因为 TX2 的内核数量(2 个 MP, 每个 MP 包含 128 个内核)明显少于 Xavier 的内核数量(8 个 MP, 每个 MP 包含 64 个内核), 所以在并行性和计算能力上, Xavier 都要强于 TX2.

6 结束语

本文讨论了一种基于 ABFT 的用于 GEMM 计算的库函数, 经过评估测试, 证明了这种方法可以以相对较小的额外开销, 在一定程度上的解决在恶劣环境中由于 GUP 的“位”翻转而发生的 SDC 错误问题, 为用户提供结果更具有可信性的计算结果, 因此, 具有较强的实用价值. 文中关于容错计算的高效实现, 以及关于判决阈值的在线计算方法, 都是在目前已知

的相关研究中没有涉及的, 具有一定的参考意义.

目前这种方法已经应用在某个低轨航天器的科学实验计算中, 取得了较好的效果. 在将来, 对其它一些矩阵相关的复杂运算(如矩阵分解)进行容错研究, 并对一些算法进行矩阵化表示, 来应用这种 ABFT 方法, 将是本研究的关注重点.

参考文献:

- [1] PETERSEN E. 空间单粒子效应—影响航天电子系统的危险因素[M]. 韩郑生, 沈自才, 丁义刚, 等译. 北京: 电子工业出版社, 2016.
PETERSEN E. Single event effects in aerospace[M]. HAN Z S, SHEN Z C, DING Y G, et al trans. Beijing: Publishing House of Electronics Industry, 2016.
- [2] 徐丹妮, 贺占庄. 一种基于 GPU 通用计算的容错方法[J]. 微电子学与计算机, 2014, 31(2): 18-22. DOI: 10.19304/j.cnki.issn1000-7180.2014.02.005.
XU D N, HE Z Z. A fault tolerance method based on GPGPU[J]. Microelectronics & Computer, 2014, 31(2): 18-22. DOI: 10.19304/j.cnki.issn1000-7180.2014.02.005.
- [3] DE DAG, PILLA L L, SANTINI T, et al. Evaluation and mitigation of radiation-induced soft errors in graphics processing units[J]. IEEE Transactions on Computers, 2016, 65(3): 791-804. DOI: 10.1109/TC.2015.2444855.
- [4] XU X H, YANG P, MA Z C, et al. A fast and iterative migration for GPU applications[C]//Proceedings of the Fourth International Conference on Information Science & Cloud Computing. Guangzhou, China, 2015. DOI: 10.22323/1.264.0051.
- [5] ZENO L, MENDELSON A, SILBERSTEIN M. GP-UpIO: the case for I/O-driven preemption on GPUs [C]//Proceedings of the 9th Annual Workshop on General Purpose Processing using Graphics Processing Unit. Barcelona, Spain: ACM, 2016: 63-71. DOI: 10.1145/2884045.2884053.
- [6] 孟晨, 曹宗雁, 王龙, 等. 基于 Charm++ 运行时环境的异构计算应用容错研究[J]. 计算机工程与应用, 2016, 52(13): 1-7. DOI: 10.3778/j.issn.1002-8331.1601-0299.
MENG C, CAO Z Y, WANG L, et al. Charm++ RTS based fault tolerance mechanism of heterogeneous computing[J]. Computer Engineering and Applications, 2016, 52(13): 1-7. DOI: 10.3778/j.issn.1002-8331.1601-0299.

- Systems. Cambridge: ACM, 2015: 91-99.
- [3] BRAZIL G, YIN X, LIU X M. Illuminating pedestrians via simultaneous detection and segmentation[C]//Proceedings of 2017 IEEE International Conference on Computer Vision. Venice: IEEE, 2017: 4950-4959. DOI: 10.1109/ICCV.2017.530.
- [4] HWANG S, PARK J, KIM N, et al. Multispectral pedestrian detection: benchmark dataset and baseline [C]//Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston: IEEE, 2015: 1037-1045. DOI: 10.1109/CVPR.2015.7298706.
- [5] WAGNER J, FISCHER V, HERMAN M, et al. Multispectral pedestrian detection using deep fusion convolutional neural networks [C]//24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges: ES-ANN, 2016.
- [6] LIU J J, ZHANG S T, WANG S, et al. Multispectral deep neural networks for pedestrian detection[M]//WILSON R C, HANCOCK E R, SMITH W A P. Proceedings of the British Machine Vision Conference. British: BMVA Press, 2016. DOI: 10.5244/C.30.73.
- [7] GUAN D Y, CAO Y P, YANG J X, et al. Fusion of multispectral data through illumination-aware deep neural networks for pedestrian detection[J]. Information Fusion, 2019, 50: 148-157. DOI: 10.1016/j.inffus.2018.11.017.
- [8] KONIG D, ADAM M, JARVERS C, et al. Fully convolutional region proposal networks for multispectral person detection[C]//Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops. Honolulu: IEEE, 2017: 49-56. DOI: 10.1109/CVPRW.2017.36.
- [9] GUAN D Y, CAO Y P, YANG J X, et al. Exploiting fusion architectures for multispectral pedestrian detection and segmentation[J]. Applied Optics, 2018, 57(18): D108-D116.

作者简介:

金彦亮 男,(1974-),博士,副教授.研究方向为无线传感器网络、无线多媒体传感器网络、无线宽带接入和信号处理.

葛飞扬(通讯作者) 男,(1994-),硕士研究生.研究方向为目标检测、行人检测. E-mail: gfy0921@qq.com.

(上接第 26 页)

- [7] HUANG K H, ABRAHAM J A. Algorithm-based fault tolerance for matrix operations[J]. IEEE Transactions on Computers, 1984, C-33(6): 518-528. DOI: 10.1109/TC.1984.1676475.
- [8] CHEN J Y, LIANG X, CHEN Z Z. Online algorithm-based fault tolerance for choleskydecomposition on heterogeneous systems with GPUs[C]//Proceedings of 2016 IEEE International Parallel and Distributed Processing Symposium. Chicago, IL, USA: IEEE, 2016: 993-1002. DOI: 10.1109/IPDPS.2016.81.
- [9] BRAUN C, HALDER S, WUNDERLICH H. A-ABFT: autonomous algorithm-based fault tolerance for matrix multiplications on graphics processing units [C]//Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. Atlanta, GA, USA: IEEE, 2014: 443-454. DOI: 10.1109/DSN.2014.48.
- [10] WHITEHEAD N, FIT-FLOREA A. Precision and performance: floating point and IEEE 754 compliance for NVIDIA GPUs[R]. Santa Clara, USA: NVIDIA Corporation, 2011.

作者简介:

包冲 男,(1979-),硕士,高级工程师.研究方向为高性能计算及计算机可靠性. E-mail:17148897@qq.com.

张善从 男,(1977-),博士,二级研究员.研究方向为高可靠计算机应用、航天电子学工程.