arXiv:2305.02444v2 [cs.DC] 9 May 2023

FT-GEMM：一种适用于 x86 CPU 的容错高性能 GEMM 实现

Shixun Wu
swu264@ucr.edu 美国加利福尼亚大学河滨分校，美国加利福尼亚州河滨市

Yujia Zhai
yzhai015@ucr.edu 美国加利福尼亚大学河滨分校，美国加利福尼亚州河滨市

Jiajun Huang
jhuan380@ucr.edu 美国加利福尼亚大学河滨分校，美国加利福尼亚州河滨市

Zizhe Jian
zjian106@ucr.edu 美国加利福尼亚大学河滨分校，美国加利福尼亚州河滨市

Zizhong Chen
chen@cs.ucr.edu 美国加利福尼亚大学河滨分校，美国加利福尼亚州河滨市

摘要

通用矩阵乘法（GEMM）对科学计算和机器学习至关重要。然而，计算平台规模的不断扩大引发了人们对硬件和软件可靠性的担忧。本文介绍 FT-GEMM，这是一种能够在运行时容忍软错误的高性能 GEMM 实现。我们通过将内存密集型操作融合到 GEMM 汇编内核中，在算法层面实现容错功能，并为并行 FT-GEMM 设计了一种缓存友好型方案。在英特尔 Cascade Lake 上的实验结果表明，FT-GEMM 具有高可靠性和高性能：对于串行和并行 GEMM，即使每分钟注入数百个错误，其速度仍比 Intel MKL、OpenBLAS 和 BLIS 快 3.50 %~ 22.14%。

ACM 参考文献格式：
Shixun Wu、Yujia Zhai、Jiajun Huang、Zizhe Jian 和 Zizhong Chen。2023。FT-GEMM：x86 CPU 上的容错高性能 GEMM 实现。载于第 *32* 届国际高性能并行与分布式计算研讨会论文集（*HPDC '23*），*2023*年*6*月*16*日至*23*日，美国佛罗里达州奥兰多。ACM，纽约州纽约，美国，2 页。https://doi.org/ 10.1145/3588195.3595947

1 引言

由于性能增强技术的应用，处理器芯片更容易受到瞬态故障的影响。瞬态故障可能会改变信号传输，或在不被察觉的情况下破坏存储值中的比特。因此，美国能源部已将可靠性确定为百亿亿次级计算面临的主要挑战之一。我们将研究范围限定为来自计算逻辑单元（例如， 1+1=3）的故障后继续运行错误，即软错误。针对核心计算例程 GEMM，已经提出了若干容错方案，以较低开销容忍软错误。然而，鉴于计算速度与内存传输速度之间存在巨大差距，这些方法在使用支持 AVX-512 的处理器时效率较低。在此

本文介绍了 FT-GEMM，这是一种能够动态容忍软错误的高性能 GEMM [1, 3, 4]。我们在英特尔 Cascade Lake 处理器上，将我们的实现与最先进的 GEMM 实现 OpenBLAS [2], BLIS 和 Intel MKL 进行了比较。我们的主要贡献包括：

- 使用 AVX-512 汇编指令的 GEMM，其性能比 OpenBLAS、BLIS 和 MKL 高出 3.33%–22.19%。
- FT-GEMM 的额外开销可忽略不计（0.35%–3.10%）：通过将内存密集型操作、校验和编码及验证融合到 GEMM 汇编代码中实现。
- 缓存友好的多线程 FT-GEMM，开销可忽略不计（0.16%-3.53%）。
- 高性能（比 OpenBLAS、BLIS 和 MKL 快 3.5%–22.1%），并且在注入数百个错误的情况下仍具有可靠性。

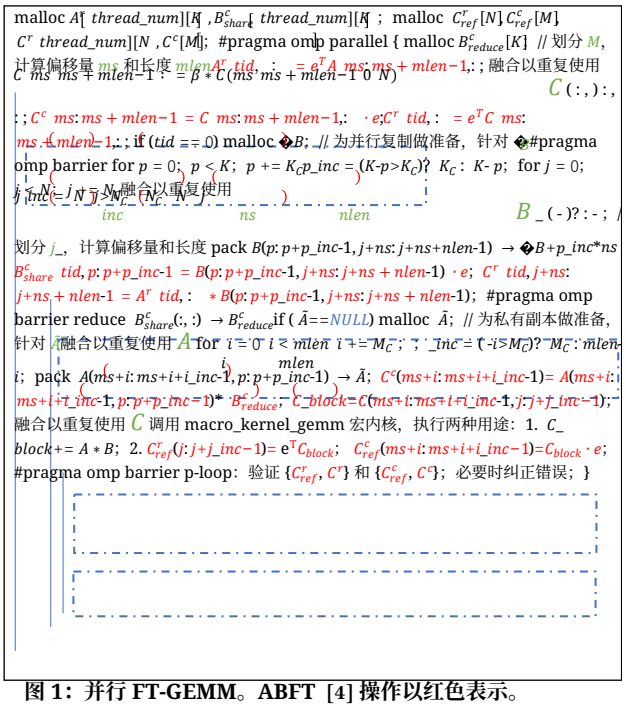


图 1：并行 FT-GEMM。ABFT [4] 操作以红色表示。

2 FT-GEMM

2.1 双精度通用矩阵乘法的实现

我们采用打包和缓存分块框架。将 for 循环最外层的三层划分，使 A 和 B 的子矩阵能够驻留在指定的缓存层中。这三个 for 循环的步长 M_C、 N_C 和 K_C 定义了宏内核的形状，该形状由各级缓存的大小决定。宏内核通过在微内核中迭代相乘 A（$M_R \times K_C$）与 B（$K_C \times N_R$），来更新 C 的一个 $M_C \times N_C$ 子矩阵。

2.2 FT-DGEMM

如上一节所述，内存传输与浮点计算之间的巨大差距，导致 $O(n^3)$ GEMM 无法再摊销与 $O(n^2)$ 校验和相关的操作。因此，我们设计了一种融合式 ABFT [4]方案，以尽可能减少校验和操作的内存占用。具体而言， C^c 和 C^r 的编码与矩阵缩放例程融合在一起。

$C=\beta C$。当我们加载 B 并将其打包到连续内存缓冲区$\tilde{B}$中时，通过重用 B，同时计算校验和 B^c和校验和 C^r。在这个融合打包例程中，每次加载都会将每个 B元素重用三次。类似地，为了进行打包而加载的 A中的每个元素都会被重用，以更新列校验和 C^c。在计算 C_{block}+=$\tilde{A}\cdot\tilde{B}$的宏内核中，我们在寄存器级重用已计算出的 C元素，以更新参考校验和 C^r_{ref}和 C^c_{ref}，从而验证计算的正确性。通过融合 ABFT 内存占用，容错开销变为纯计算开销，从约 15% 降至 2.94%。

2.3 并行 FT-DGEMM

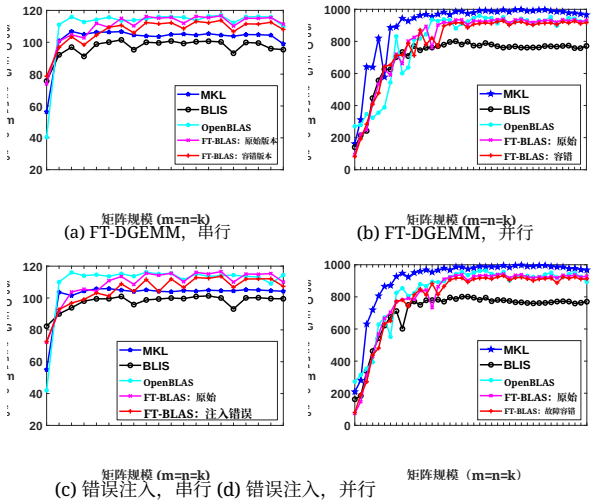
除了提供高效的串行实现外，我们还为有容错和无容错的双精度通用矩阵乘法提供多线程支持。在英特尔 Cascade Lake 服务器 CPU 上，物理核心共享一个大型统一 L3 缓存，而每个物理核心拥有较小的 L2 缓存。为了在多线程实现中映射这一缓存层次结构，我们为 B 分配一个由所有线程共享的内存缓冲区$\tilde{}$，并为 $\tilde{A}$ 让每个线程申请一个私有内存缓冲区。C 矩阵上的计算工作负载沿 M 维度划分。由于内存缓冲区 $\tilde{A}$ 是线程私有的，每个线程都会将矩阵 A 中的数据打包到自己的 $\tilde{A}$ 缓冲区中。将矩阵 C 打包到共享内存$\tilde{}$缓冲区 $\tilde{B}$时，内存访问工作负载沿N维度划分，每个线程负责打包一块 $\tilde{B}$。我们对 A（A^r）的行校验和向量以及 C（C^c，C^r）的完整校验和向量进行校验和编码。为计算C 校验和，我们沿 M 维度划分 C 矩阵，使每个线程计算一部分列校验和C^c，同时维护自己的行校验和向量C^r的本地副本。类似地，我们沿 M 维度划分 A 矩阵，以并行计算其行校验和 A^r。B^c 的校验和编码$\tilde{}$与将 B 打包到 $\tilde{B}$ 的并行打包操作融合，同时更新 C的参考行校验和。因此，从主存加载的每个 B 元素都会被重复利用三次。由于并行复制操作沿N维度划分 B，因此需要在线程间额外执行一次归约操作，以计算最终的列校验和 B^c。

3 实验评估

为验证我们优化的有效性，我们比较了 FT-BLAS 与三种先进 BLAS 库的性能对比：Intel oneMKL（2020.2，本节简称 MKL）、OpenBLAS（0.3.13）和 BLIS（0.8.0）。测试在一台配备 3.70 GHz 基准频率的 Intel Xeon W-2255 Cascade 处理器以及 32 GB DDR4-2933 RAM 的计算机上进行。硬件预取器按照 Intel BIOS 默认设置启用。每项测量重复二十次，并报告平均性能。平均性能的测试矩阵规模范围为 2048_2 至 10240_2。对于多线程并行基准测试，矩阵规模范围为 512_2 至 20480_2。我们使用 icc 19.0 和优化标志 -O3 编译代码。

3.1 FT-GEMM 的性能

本节中的结果是在容错 DMR 和 ABFT 正常运行、但未进行主动故障注入的情况下获得的。在图 2(a) 中，



(a) FT-DGEMM，串行 (b) FT-DGEMM，并行

(c) 错误注入，串行 (d) 错误注入，并行

图 2：英特尔 Cascade Lake 上 FT-GEMM 的性能比较。

我们的基线 GEMM 实现（FT-GEMM：Ori）与 MKL、OpenBLAS 和 BLIS 相比，表现出相当或更优的性能。针对计算密集型双精度通用矩阵乘法，我们融合的容错策略相较于基线平均产生 1.17%-3.58% 的开销。图 2(b) 比较了具备容错能力的 FT-GEMM 的并行性能。借助可扩展的并行设计，以及融合到打包例程和汇编内核中的 ABFT 操作，FT-DGEMM 仅产生可忽略的开销（1.79%）。我们的具备容错能力的双精度通用矩阵乘法性能比 BLIS 快 16.97%，与 OpenBLAS 相当，同时略低于闭源的 Intel MKL。

3.2 错误注入下的性能

我们通过向每个计算内核注入多个计算错误，并将最终计算结果与 MKL 进行比对，验证了所提出的容错方案的有效性。外部错误注入工具往往会显著降低原生程序的运行速度。因此，我们在源代码层面注入错误，以尽量减少对原生程序性能的影响。如图 2(c) 所示，即使能够容忍 20 个注入错误，我们的保护方案仍比 OpenBLAS 和 BLIS 分别快 22.89% 和 21.56%，并比闭源的 MKL 快 4.98%。如图 2(d) 所示，我们的 FT-BLAS 性能与 OpenBLAS 相当，比 BLIS 快 16.83%。

4 致谢

本研究得到美国能源部科学办公室先进科学计算研究办公室“通过先进计算实现科学发现”（SciDAC）计划的资助，资助编号为 DE-SC0022209。

参考文献

[1] Kuang-Hua Huang 和 Jacob A Abraham。1984 年。矩阵运算的基于算法的容错。*IEEE* 计算机学报100，6（1984），518–528。[2] OpenBLAS。2021 年获取。https://github.com/xianyi/OpenBLAS。在线。[3] Shixun Wu、Yujia Zhai、Jinyang Liu、Jiajun Huang、Zizhe Jian、Bryan Wong 和 Zizhong Chen。2023 年。GPU 上支持在线容错的高性能 GEMM 的剖析。*arXiv* 预印本 *arXiv:2305.01024*（2023）。[4] Yujia Zhai、Elisabeth Giem、Quan Fan、Kai Zhao、Jinyang Liu 和 Zizhong Chen。2021 年。FT-BLAS：一种支持在线容错的高性能 BLAS 实现。载于 *ACM* 超级计算国际会议论文集。127–138。