arXiv:2603.02621v1 [cs.MS] 2 Mar 2026

GoldbachGPU：用于验证哥德巴赫猜想的开源 GPU 加速框架

Isaac Llorente-Saguer[1]

[1]英国伦敦独立研究者*illorentes@gmail.com*

2026年3月4日

摘要

我们推出*GoldbachGPU*，这是一个开源框架，利用商用 GPU 硬件对哥德巴赫猜想进行大规模计算验证。此前基于 GPU 的方法报告称，由于采用整体式素数表分配，内存上限约为 10^{11} 。我们证明，这一限制源于架构而非根本因素：密集的位打包素数表示可将内存占用减少 $16\times$ ，而分段双筛设计则彻底消除了显存上限。通过反转验证循环，并将 GPU 快速路径与多阶段素性判定预言机相结合，该框架在单块 NVIDIA RTX 3070（8 GB 显存）上实现了最高 10^{12} 的穷举验证，且未发现反例。每个分段需要 14 MB 显存，因此墙钟时间为 $O(N)$，内存占用为 $O(1)$，运行于 N。严谨的 CPU 回退可保证数学完备性，不过实际运行中从未启用。采用 GMP 和 OpenMP 的任意精度检查器，通过同步批量搜索策略，将单数验证扩展至 10^{10000} 。该分段式架构在数据中心硬件上也展现出良好的多 GPU 扩展能力（在 $8\times$H100 上进行了测试）。所有代码均为开源，附有文档，并可在商用及高端硬件上复现。

1 引言

哥德巴赫猜想最早见于克里斯蒂安·哥德巴赫于 1742 年写给莱昂哈德·欧拉的一封信。该猜想断言，每个大于 2 的偶整数都可以表示为两个素数之和。尽管表述简单，这一猜想至今仍未得到证明。理论研究已取得重大进展，其中最重要的成果包括陈氏定理 [1], ，该定理表明，每个足够大的偶整数都是一个素数与一个半素数之和；此外，赫尔夫戈特对三元哥德巴赫猜想的证明 [5]也取得了突破——但二元形式仍未能得到证明。

因此，计算验证发挥了重要的补充作用。一个多世纪以来，算法和硬件日益精进，已将验证范围从十九世纪的 10^4 推进到 Oliveira e Silva、Herzog 和 Pardi 的最新分布式计算所达到的 4×10^{18} [6]。他们的工作依赖于高度优化、缓存高效的分段筛法，并在多台机器上进行了长达数年的大规模并行计算。

相比之下，通用硬件上的验证仍受到限制。通用图形处理器（GPGPU）计算的出现带来了强大的并行吞吐能力，但朴素的图形处理器实现会遭遇难以逾越的内存墙：当范围超过 10^{10}–10^{11} 时，在显存中存储一个整体素数表将变得不可行。此前基于图形处理器的研究 [10, 3] 明确指出，显存耗尽

和线程分歧是扩展规模的主要障碍，而且现有的开源框架都未能展示如何在单个图形处理器上对超过 10^{11} 的范围进行穷尽验证。

本研究填补了这一空白。我们提出 GoldbachGPU，这是一种开源框架，通过采用受 Oliveira e Silva 循环反转策略启发的分段双筛架构，消除了显存上限。通过结合稠密位压缩素数表示、正确的分段方案和多阶段素性判定预言机，我们的实现可在单张消费级 GPU（NVIDIA RTX 3070，8GB 显存）上对最高至 10^{12} 的范围进行穷举验证。严谨的 CPU 回退保证了数学完备性，而基于 GMP 的任意精度检查器则将单数验证扩展至 10^{10000}。

本研究作出以下贡献：

- 一种 **分段双筛架构**，消除了以往图形处理器方法的显存上限，并能在通用硬件上实现对 10^{12} 的穷举验证。
- 一种 **稠密素数位集**，与字节数组表示相比可实现 $16\times$ 的内存缩减，从而支持将素数表驻留于图形处理器上，范围最高达 10^{11}。
- 一种 **两阶段验证设计**，结合图形处理器快速路径与严谨的 CPU 穷举回退，无论最小哥德巴赫素数如何增长，都能确保正确性。
- 一个使用 GMP 和 OpenMP、采用异步批量搜索策略的**任意精度检查器**，已验证至 10^{10000}。
- 一个完全开源、有文档记录且可复现的实现，面向消费级硬件。

2 方法

该框架由五种互为补充的验证工具组成，能力依次递增。每种工具都通过与前一种工具进行验证，建立从 CPU 预言机到任意精度验证的正确性链。

2.1 CPU 基线 (`cpu goldbach`)

顺序 CPU 实现作为真实基准预言机。分段式埃拉托斯特尼筛法生成不超过 N 的所有素数。对于每个偶整数 $n \in [4, N]$，验证器扫描从 2 p到 $n/2$ 的素数，并通过直接位集查找检查 $q = n - p$ 是否为素数。由此建立参考实现，所有图形处理器工具均以此为基准进行验证。

2.2 GPU 全局位集验证器（`goldbach gpu2`）

该工具引入了稠密位集表示，使大规模 GPU驻留素数表成为可能。

稠密素数位集。 偶数（2除外）绝不是素数，因此不纳入存储。对于奇数 $n \geq 3$，位索引为

$$\text{word} = \left\lfloor \frac{(n-3)/2}{64} \right\rfloor, \qquad \text{bit} = \left(\frac{n-3}{2} \right) \bmod 64,$$

存储于64位字数组中。相对于按数字逐字节的存储方式，这实现了 $16\times$ 内存缩减。在 $N = 10^{11}$时，该位集需要5.96 GB显存（处于RTX 3070的8 GB上限之内），而字节数组则需要95 GB。

显存硬上限。 由于整个素数表都必须驻留在显存中，因此在 8GB GPU 上，此设计无法超过 $N \approx 10^{11}$ 。若要扩展到 10^{12} ，则需要 ≈ 59GB 显存。

GPU 内核。 每个偶整数 n都分配一个 CUDA线程。每个线程使用位集扫描从 2 到 $\min(n/2,\ N)$ 的素数 p。边界检查可防止越界访问内存，而在找到第一个有效分拆后立即退出，则可减少不必要的工作。CPU 位集通过使用 OpenMP（20 个线程，提速 $1.7\times$，受内存带宽限制）的并行分段筛法构建。

2.3 分段双重筛验证器（goldbach_gpu3）

goldbach_gpu3 是本工作的主要贡献。它通过正确的分段策略和多阶段素性判定预言机，消除了 goldbach_gpu2 的显存上限。

2.3.1 正确分段

对素数表进行简单分段是不够的：对于给定的偶整数 n，有效分拆（$p,\ q$）中的 $p+q=n$ 可能位于数轴上的任意位置。任何强制要求 q 与 p 位于同一分段的设计，都会漏掉有效分拆，因此无法用于穷举验证。

遵循 Oliveira e Silva 的双重筛法 [6], ，我们反转了验证循环。我们不再遍历偶数 n 并寻找合适的 p，而是遍历候选素数p 并标记所有满足 n 的偶数 $q = n - p$ 为素数。这种反转使得素数表能够安全地进行分段。对于每个分段 $[A, B]$，CPU 都会执行分段式埃拉托斯特尼筛法，以确定该分段中所有奇数的素性。生成的位集大小为 14 MB（固定分段大小为分段大小 $= 10^7$ 个偶数），并被传输到显存中，在该分段的验证期间始终驻留于显存。

对于每个偶数 $n \in [A, B]$，GPU 内核会遍历候选素数 $p \le P_{\mathrm{SMALL}}$ 并计算 $q = n - p$。 q 的素性通过一个三路判定器确定，并严格按照以下优先级顺序进行评估：

1. $q \le P_{\mathrm{SMALL}}$：查询常驻的小素数位集（≈ 122KB）。
2. $q \in [A, B]$：在当前分段位集（≈ 14MB）中查找。
3. $P_{\mathrm{SMALL}} < q < A$：确定性 64 位米勒–拉宾检验 [9]。

此优先级链涵盖了 q 的所有可能位置，无需在显存中存储一个庞大的素数表。由于分段大小固定，因此显存用量在所有范围内都保持恒定；无论哥德巴赫分拆的分布如何，查找逻辑都能保证正确性。图1 展示了该判定器的结构。

2.3.2 阶段 1：GPU 快速路径与 H(x) 包络

最坏情况下的最小哥德巴赫素数 $H(x) = \max_{n \le x} p_{\min}(n)$ 增长极其缓慢。对 Oliveira e Silva 的上包络数据进行二次拟合（$R^2 = 0.991$） [6] 得到

$$H(x) \approx 38.572(\log_{10} x)^2 - 193.21 \log_{10} x + 297.36,$$

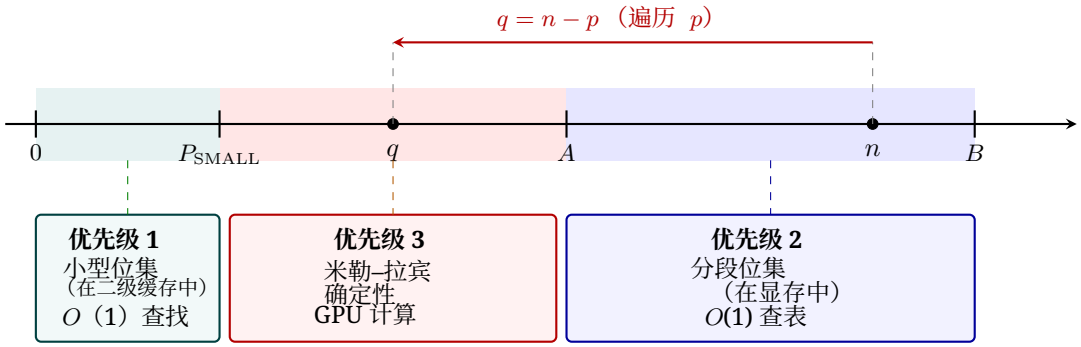

图1：goldbach gpu3 中的素性判定预言机选择。对于当前分段 $[A, B]$中的每个偶整数 n ，验证器遍历候选素数 p并计算 $q = n - p$。 q 的素性由严格优先级链中第一个适用的预言机判定，以尽量降低 GPU 计算成本：**(1)** 当 $q \le P_{\mathrm{SMALL}}$ 时，使用常驻的 122 KB 小素数位集（其中 P_{SMALL} 是永久常驻小素数表的上限）；**(2)** 当 $q \in [A, B]$时，使用在 CPU 上预先筛选的 14 MB 分段位集；或**(3)** 对过渡区域 $P_{\mathrm{SMALL}} < q < A$进行确定性的 64 位米勒–拉宾检验。这种三路结构可确保完整覆盖所有可能的 q 值，而无需将一个整体素数表常驻于显存中。

预测 $H(10^{12}) \approx 2000$。我们设定 $P_{\mathrm{SMALL}} = 10^6$，远远超过此上界。原子计数器会跟踪尚未解决的 n 值；每处理完一个素数批次，只需一次 8 字节传输即可确定是否需要继续处理后续批次。

2.3.3 阶段 2：穷举式 CPU 回退

如果阶段 1 未能为某个 n找到分拆，CPU 回退会通过试除法测试从 3 到 p 的所有奇数 $n/2$ ，且对 p不设上限。这保证了数学完备性，不受方程 2.3.2精度的影响。在此处报告的所有实验中，阶段 2 从未被调用。

2.3.4 复杂度与内存

每段包含 S 个偶数，处理这些偶数需要 $O(S \cdot \pi(P_{\mathrm{SMALL}}))$ 次运算，其中 $\pi(P_{\mathrm{SMALL}})$ 表示不超过 P_{SMALL} 的素数数量。当 S 和 P_{SMALL} 固定时，每段的处理开销相对于 N 为常数。因此，总墙钟时间为 $O(N)$，显存用量为 $O(1)$，即 N（每段固定为 14 MB）。相比之下，goldbach gpu2 需要 $O(N/16)$ 字节显存，并且在 8 GB 设备上无法超过 $\approx 10^{11}$ 。

2.4 GPU 单数检查器（`single check`）

对于单个 64 位偶整数（最大可达 $\approx 1.8 \times 10^{19}$），单次检验使用基于 GPU 的确定性米勒–拉宾检验。小素数 $p \le n/2$ 由 CPU 分批生成；对于每一批，GPU 线程网格会使用已证明对所有 64 位整数均正确的 12 个见证数的确定性变体，检验 $q = n - p$ 是否为素数 [9]。由于并发执行，返回的分拆有效，但不一定是最小分拆。

2.5 任意精度检查器（big check）

要验证超出 64 位范围的偶数，big check 使用 GNU 多重精度库（GMP） [4]。对于给定的 n（以十进制字符串形式提供），小素数 p 会生成至 10^7。对于每个 p，使用 GMP 算术精确计算其补数 $q = n - p$，并使用 GMP 的米勒–拉宾实现检验 q的素性 [7]，共进行 25 轮（误报概率$< 10^{-15}$）。

同步批量搜索。朴素的 OpenMP 并行会导致线程失控：快速线程推进到较大的素数时，较慢的线程却因耗时的素性检验而停滞。为避免这种情况，素数按每批 1000 个进行处理，批次之间设置同步屏障。任一线程找到有效分拆后，原子标志便会发出提前终止信号。在批次内，线程检测到该标志后会立即退出。这既能避免批次间失控，也能避免批次内失控，并通过 schedule(dynamic,1) 确保良好的负载均衡。

big check 的实际上限取决于位数 d 下 GMP 素性检验的 $O(d^3)$ 开销。在 $d = 10$，001（10^{10000}）时，每次检验耗时数毫秒；在 $d = 100$，001（10^{100000}）时，每次检验耗时数秒，因此穷举搜索变得不可行。

3 结果

所有实验均在以下设备上进行：Intel i7–12700H CPU（20 个逻辑线程，WSL2 Ubuntu 24.04）、NVIDIA RTX 3070 GPU（8 GB 显存，448 GB/s 带宽）以及 32 GB 内存。所有计时均为当前代码库的实际运行时间。本文所述实验使用该软件的 release v1.1.0 版本运行，该版本已存档于 Zenodo（https://zenodo.org/records/18837081）。

3.1 CPU 基线

表 1 报告了顺序 CPU 预言机 cpu goldbach的性能。这些结果为衡量图形处理器的加速效果提供了参照。基线性能随 N线性增长。

表 1：CPU 基线（cpu goldbach）：顺序筛法和哥德巴赫扫描。所有上限均未出现故障。

上限	已检查偶数	筛法	检查	总计
10^7	4,999,999	27 毫秒	89 毫秒	117 毫秒
10^8	49,999,999	768 毫秒	1,052 毫秒	1,820 毫秒
10^9	499,999,999	9,836 毫秒	12,786 毫秒	22,622 毫秒
10^{10}	4,999,999,999	119,947 毫秒	188,388 毫秒	308,335 毫秒

3.2 范围验证与内存效率

表2 展示了 goldbach gpu2 的结果，该方案使用全局驻留于GPU的位集。与 CPU 基线相比，goldbach gpu2 在 16× 时实现了 10^9 的 12× 倍加速， 10^{10}。在较小的上限（10^7–10^8）下，内核启动开销会削弱图形处理器的优势，但从 10^9 起，图形处理器始终占据优势。

表2：使用 goldbach gpu2 进行范围验证（全局位集、OpenMP 筛法、20 个线程）。无失败。无法达到 10^{11}：位集需要 5,960 MB，在一台 8 GB 设备上会剩余不足的余量。

上限	偶数 n 已检查	筛法（CPU）	内核（GPU）	总计
10^7	4,999,999	16 毫秒	6 ms	22 毫秒
10^8	49,999,999	46 毫秒	75 毫秒	121 毫秒
10^9	499,999,999	687 毫秒	698 毫秒	1,386 毫秒
10^{10}	4,999,999,999	17,923 毫秒	7,520 毫秒	25,443 毫秒

稠密位集的内存占用汇总于表3中。与 16×字节数组相比，内存占用减少，因此图形处理器验证在上限达到 10^{11}, 之前是可行的；但当上限达到 10^{12}, 时，连位集也变得不可行，因此采用`goldbach gpu3`的分段设计。

表3：显存占用：字节数组与稠密位集。 16× 缩减幅度恒定。在 10^{12}, 时，必须进行分段（ goldbach gpu3）。

上限	字节数组	位集	在 8 GB GPU 上可行吗?
10^9	953 MB	59 MB	Both
10^{10}	9,536 MB	596 MB	仅位集
10^{11}	95,367 MB	5,960 MB	仅位集
10^{12}	953,674 MB	59,605 MB	都不是

3.3 分段范围验证

表 4 报告了 `goldbach gpu3` 在 $P_{\mathrm{SMALL}} = 10^6$ （78,498 个驻留于 GPU 的素数）下的结果。在任何上限下均未观察到失败。所有分段均在阶段 1 中完全解决，没有发生 CPU 回退，这意味着 $p_{\min}(n) \le 2\times10^6$ 对于所有偶数 $n \le 10^{12}$。这与包络预测 $H(10^{12}) \approx 2000$ 一致，该预测来自公式 2.3.2，我们的界限比预测的最坏情况高出三个数量级。

尽管 goldbach gpu3 因逐段筛法构造和显存传输，在相同上限下比 goldbach gpu2 慢，但它是唯一能够在 8GB GPU 上达到 10^{12} 及以上的设计。

表 4：使用 goldbach gpu3（分段式， $P_{\mathrm{SMALL}} = 10^6$，分段大小 $= 10^7$）进行穷举验证。所有上限下阶段 2 的 CPU 回退次数均为零。未发现失败。

上限	偶数 n已检查	阶段 1 成功	总耗时
10^9	499,999,999	100%	2,048 毫秒
10^{10}	4,999,999,999	100%	23,547 毫秒
10^{11}	49,999,999,999	100%	（3.8 分钟）226,490 毫秒
10^{12}	499,999,999,999	100%	（41 分钟）2,457,040 毫秒

3.4 运行时间扩展

图 2 展示了这三种工具的双对数运行时间。两种图形处理器实现的运行时间均随 N线性增长，这与第 $O(N$ 节预测的 行为一致。随着 N 从 10^9 增加到 10^{12}„ 分段设计的单段成本近乎恒定，证实了理论分析。

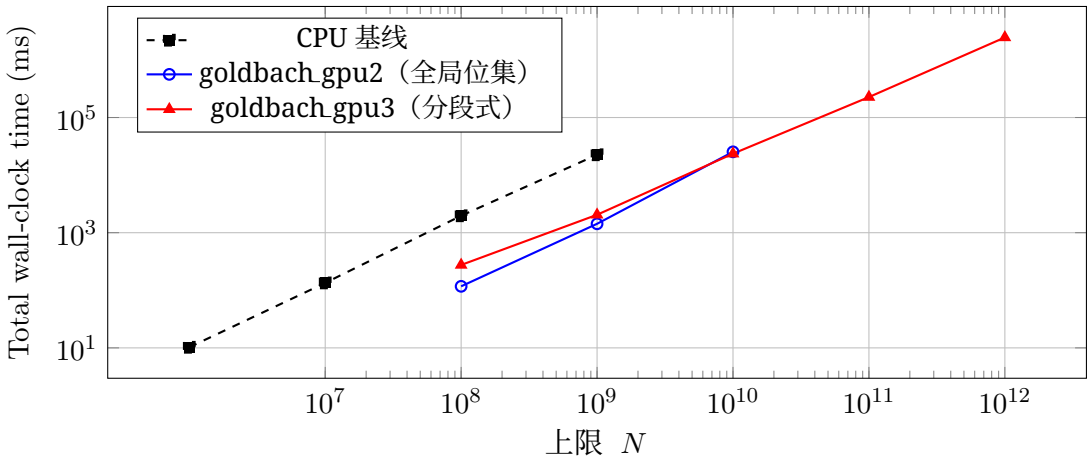

图 2：运行时间的对数—对数扩展图。CPU 基线（黑色虚线）、goldbach gpu2（蓝色，全局位集，实际运行中限制为 10^{10} ）和 goldbach gpu3（红色，分段式，无显存上限）。三者的扩展趋势都近似线性。在相同上限下，goldbach gpu3 因分段开销而运行较慢，但它是唯一能够达到 10^{12} 的工具。

3.5 任意精度结果

表5 报告了`big check`的结果。对于中等规模的输入，运行时间主要取决于第一个有效素数的索引 p：一个需要 $p = 113$ 、共 204 位的数字可在 43 毫秒内完成验证；而一个需要 $p = 26,$ 981、共 1,001 位的数字则需 363 毫秒。然而，对于超大规模输入，$O(d^3$GMP 米勒 – 拉宾检验的开销会成为主要因素，数字位数对总运行时间的影响也越来越大。同步批处理策略将 10^{10000} 运行时间从 231 秒（朴素 OpenMP）¨缩短至 182 秒，提升了 21%，这是通过避免批次内线程失控而实现的。

表 5：任意精度验证（big check、GMP、20 个 OpenMP 线程，批次大小为 1,000）。对于中等位数，耗时主要取决于第一个有效 p 的索引；对于超大输入，GMP 素性检验的 $O(d^3)$ 开销则会成为主要因素。

n	位数	p 找到	Time
10^{50}	51	383	43 毫秒
10^{100}	101	797	43 毫秒
10^{200}	204	113	43毫秒
10^{1000}	1,001	26,981	363毫秒
10^{10000}	10,001	47,717	182,000毫秒

3.6 多 GPU 扩展性与效率

goldbach gpu3 二进制程序接受 --gpus=i 标志，可将工作分配到指定的 GPU 上（也可使用 --gpus=-1 来使用所有可用设备）。我们在两个平台上进行了强扩展实验：一台最多配备 $6\times$ 块 NVIDIA RTX 3090 GPU 的消费级节点，以及一台配备 NVIDIA H100 GPU 的数据中心节点。

在 $N = 10^{10}$ 该框架在 RTX 3090 系统上展现出近乎线性的扩展性（表6）。扩展至 4 个 GPU 时，效率仍高于 91%；扩展至 6 个 GPU 时，效率仅适度降至 80.1%。这与阿姆达尔定律开始显现的情况一致：CPU 分段筛阶段开始限制吞吐量。

在 H100 平台上（单 GPU 基准），$N = 10^{10}$ 处的墙钟时间与单张 RTX 3090 几乎完全相同（$\approx$19.2 秒与 20.6 秒）：仍然受 CPU 限制。使用 4 张 H100 GPU 时，$N = 10^{11}$ 运行在 49.86 秒内完成（而使用 $4\times$ RTX 3090 时为 54.0 秒）。这些结果表明，需要提高验证上限（$N \gtrsim 10^{11}$），并针对架构对 SEG SIZE 和 PSMALL 进行调优，才能充分利用数据中心硬件更高的内存带宽和计算能力。

表 6：$N = 10^{10}$ 在 NVIDIA RTX 3090 上的强扩展（分段大小 $= 10^7$）。

GPUs	N	总耗时（秒）	并行效率
1	10^{10}	20.57	100%
2	10^{10}	10.52	97.7%
4	10^{10}	5.64	91.2%
6	10^{10}	4.28	80.1%
4	10^{11}	54.0	–

表 7：H100 扩展性结果（$N = 10^{10}$ 和 $N = 10^{11}$）。

配置	N	总耗时（秒）
$1\times$ H100	10^{10}	19.18
$2\times$ H100	10^{10}	11.04
$4\times$ H100	10^{10}	5.42
$4\times$ H100	10^{11}	49.86

4 讨论

4.1 架构贡献与性能贡献

结果表明，先前基于 GPU 的研究 [10]所报告的显存局限源于单体式素数表存储，而非 GPU 架构本身的固有限制。稠密位集表示消除了这一瓶颈，使验证范围达到 10^{11},；分段双筛设计则彻底消除了该瓶颈。结合使用这两种技术，便可在单台消费级 GPU 上将穷举验证推进至 10^{12}，比以往的单 GPU 工作高出一个数量级，而且无需分布式基础设施或专用硬件。

4.2 最小哥德巴赫素数包络的实证验证

在 10^{12} 运行的所有分段中，阶段 1 均取得了 100% 的成功率，这提供了穷举的实证确认： $p_{\min}(n) \le 2\times10^{6}$ 对所有偶数 $n \le 10^{12}$ 均成立。这与上包络预测 $H(10^{12}) \approx 2000$ 一致，该预测来自方程 2.3.2，并进一步印证了哥德巴赫彗星图景：哥德巴赫分拆的数量 $c(n)$ 随着 n增长，确保在整个测试范围内，小素数分拆始终十分丰富。

4.3 与最先进方法之间的算法差距

goldbach gpu3 中实现的算法——遍历候选素数 p 并检验 $n-p$ 是否为素数——与 Oliveira e Silva 的双重筛法理念相近。不过，我们的方法仍需对每个 n 进行素性查询，并偶尔执行米勒–拉宾检验。相比之下，完整的双重筛法会在整个分段上应用按位批量标记，从而消除对每个 n 进行素性检验的需求，并以极小的常数达到 $O(N \log\log N)$ 复杂度。

要将这一方法应用于 GPU，需要将已验证数组表示为位集，并用对齐字上的按位与／或运算取代逐个 n 查询。这样的设计可以消除米勒–拉宾回退路径，并大幅减少每个分段的工作量。我们认为，这是未来工作的主要方向，也是实现在单个 GPU 上达到10^{15} 最有希望的途径。

4.4 局限性

主要的运行时开销仍然是基于 CPU 的分段筛法，它必须在将素数表传输到 GPU 之前为每个分段重建素数表。即使在 H100 上，于 $N = 10^{10}$ 进行的单 GPU 运行所需墙钟时间（≈19.2 秒）也与单块 RTX 3090 基本相同，因为 GPU 内核在几百毫秒内即可完成，随后在等待下一个由 CPU 生成的分段时空闲。

性能还对分段大小和 PSMALL 相对于主机 CPU 缓存与内存带宽的选择较为敏感。当分段位集能够放入 L2/L3 缓存（分段大小为 $= 10^{7}$ 时为 ≈600KB）时，在最多 4 块 GPU 上的并行效率超过 90%。溢出到动态随机存取存储器的更大分段会暴露主机 ≈50–100 GB/s 的内存带宽上限，即使是高带宽 GPU（RTX 3090 为 448 GB/s，H100 为 3.35 TB/s）也会因此得不到足够的数据供给。因此，数据中心硬件只有在更大的验证上限（$N \gtrsim 10^{11}$）下才开始显现其优势。

对于 big check，GMP 素性检验的 $O(d^3)$ 成本使得实际验证在 20 个线程上几分钟内只能达到约 $10^{10,000}$ 。在 $10^{100,000}$, , 单次素性检验需要数秒，因此穷举批量搜索并不现实。

4.5 未来工作

有前景的研究方向包括 GPU 加速的分段筛法构建（以消除主机与设备之间的依赖）、采用 Oliveira 风格的按位批量标记内核，以实现 $O(N \log\log N)$ 复杂度、大规模计算哥德巴赫分拆计数 $c(n)$，以及针对 SEG SIZE/PSMALL 的架构特定调优，并部署于大内存 GPU 上，以推进穷举验证的前沿。

5 结论

本文介绍了一个开源框架，可在一块消费级 GPU 上用不到 2 小时，将哥德巴赫猜想穷举验证至 10^{12} 。稠密位集表示将显存需求降低了 $16\times$，从而支持 GPU驻留素数表，并在 10^9 时达到 CPU 基线的 $16\times$倍加速。分段双重筛验证器 goldbach gpu3 消除了显存硬上限：无论总范围多大，每个分段仅需 14 MB；该设计的墙钟时间为$O(N)$，内存占用为 $O(1)$，运行于 N。严谨的 CPU 回退保证了数学完备性；在所有实验中，它都从未被调用。任意精度GMP校验器将单数验证扩展至 10^{10000} ，在 20 个线程上耗时 182 秒。所有计算均未发现反例。

软件可用性与安装

该框架面向配备 NVIDIA CUDA工具包（≥ 11.0）、GMP库以及支持 OpenMP 的 C++ 编译器的 Linux 系统。构建过程通过 CMake 管理：

```
mkdir build && cd build cmake .. && make -j$ (
```

代码仓库（https://github.com/isaac-6/goldbach-gpu）包含详尽的文档、验证套件以及可复现的示例运行，以确保在不同 GPU 架构上行为一致。该项目依据仓库中指定的开源许可证发布。

致谢

作者感谢 CUDA、GMP 和 OpenMP 的开发者提供了使本研究得以实现的基础库。作者还向 Oliveira e Silva、Herzog 和 Pardi 的开创性计算工作致以谢意 [6], ，这些工作为本项目奠定了算法背景和基准。

参考文献

[1] J. R. Chen，关于将较大的偶整数表示为一个素数与至多两个素数乘积之和，载于哥德巴赫猜想，世界科技出版社，1984年，第253–272页。https://doi.org/10.1142/0069[2]R. Crandall 和 C. Pomerance，素数：计算视角，第2版，施普林格，2005年。[3] G. Andrade、G. Ramos、M. Holanda、R. Sousa、R. Goldschmidt 和 G. Araujo，评估 *CUDA* 统一内存与动态并行在哥德巴赫猜想验证应用中的表现，《超级计算期刊》，76:4769–4790，2019年。https://doi.org/10.1007/s11227-019-02756-2[4] T. Granlund 和 GMP 开发团队，*GNU*多精度算术库，6.3版，2023年。https://gmplib.org

[5] H. A. Helfgott，三元哥德巴赫问题，arXiv:1404.2224，2014年。[6] T. Oliveira e Silva、S. Herzog 和 S. Pardi，对偶哥德巴赫猜想的实证验证及素数间隙的计算，范围截至 $4 \cdot 10^{18}$，计算数学，83(288):2033–2060，2014年。https://doi.org/10.1090/S0025-5718-2013-02787-1[7] M. O. Rabin，用于检验素性的概率算法，数论杂志，12(1):128–138，1980年。https://doi.org/10.1016/0022-314X(80)90084-0[8] J. Richstein，验证哥德巴赫猜想，范围截至 $4 \cdot 10^{14}$，计算数学，70(236):1745–1749，2001年。https://doi.org/10.1090/S0025-5718-00-01290-4[9]J. Sinclair，适用于*64* 位整数的确定性米勒–拉宾素性检验，https://miller-rabin.appspot.com，2011年。访问日期：2026-02-26。[10] D. J. Varon、A. Carell 和 M. Misterka，使用 *OpenACC* 在 *GPU* 上验证哥德巴赫猜想，哈佛大学IACS CS205项目报告，2020年。