基于局部性原理的最大误差并行检测方法

冀立光, 杨鸿儒, 周玉畅, 崔梦琦, 何昊天, 许瑾晨

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240800018

基于局部性原理的最大误差并行检测方法

冀立光　杨鸿儒　周玉畅　崔梦琦　何昊天　许瑾晨
信息工程大学网络空间安全学院　郑州 450001
(1194868658@qq.com)

摘　要　浮点数采用有限的位数来表示无限的实数进行计算,因此浮点数计算天然具有不准确性,这种不准确性可以用最大误差来度量。传统浮点数最大误差检测算法采用串行计算思维并结合经典搜索算法,当采样点数量较少时,容易将局部极大值作为全局最大值处理,从而遗漏最大误差值。如果大规模提升采样点数量,那么检测程序用时大幅增加,检测性能降低。通过应用并行计算模式指数级增加采样点数量,同步结合局部性原理在误差热点附近采用浮点动态采样策略,大幅提高检测结果的准确性。这种方法可以最大限度地发挥并行计算的算力,不仅可以提升浮点数最大误差的检测精度,还可以压缩检测程序的执行时间并提升性能,加速比可以达到1136.3,检测出的最大误差值优于当前主流检测工具,这为衡量浮点数计算指标提供了新的检测方法。

关键词:浮点运算;并行优化;区间采样;误差检测;申威异构架构
中图分类号　TP314

Maximum Error Parallel Detection Method Based on Locality Principle

JI Liguang,YANG Hongru,ZHOU Yuchang,CUI Mengqi,HE Haotian and XU Jinchen
School of Cyberspace Security,University of Information Engineering,Zhengzhou 450001,China

Abstract　Floating-point numbers use a finite number of digits to represent infinite real numbers for computation,so floating-point computation is inherently inaccurate,which can be measured by the maximum error. The traditional floating-point maximum error detection algorithm uses serial computing thinking combined with classical search algorithm. When the number of sampling points is small,it is easy to treat the local maximum as the global maximum,thus omitting the maximum error value. If the number of sampling points is increased on a large scale,the time of the detection program will be greatly increased and the performance will be reduced. In this paper,the parallel computing mode is used to exponentially increase the number of sampling points,and the floating-point dynamic sampling strategy is used to near the error hot spot in combination with the principle of synchronization and locality,which greatly improves the accuracy of the detection results. This method can maximize the computing power of parallel computing,which can not only improves the detection accuracy of the maximum error of floating-point number calculation,but also reduces the execution time of the detection program and improves the performance,and the acceleration ratio can reach 1136.3. The maximum error value detected is better than the current mainstream detection tools,which provides a new detection method for measuring the floating-point number calculation index.

Keywords　Floating point arithmetic,Parallel optimization,Interval sampling,Error detection,Sunway heterogeneous architecture

1　引言

浮点数是编程语言中必不可少的数据类型,尤其大量应用于数值运算、科学计算类型的程序中[1]。确保计算的安全性和可靠性是浮点程序的一个关键问题。计算机中的数采用二进制编码,然而二进制数却无法精确表示所有的十进制数。例如,十进制小数0.1在二进制数中就无法精确表示,只能选取距离0.1最近的浮点数来近似表示。同时,计算机中的浮点数是采用科学计数法存储的,计算机有限的尾数位无法精确表示无限的实数,浮点计算误差天然存在[2]。而浮点程序中存在大量的浮点运算,浮点误差在运算中会逐渐传播和累积,最终造成严重甚至灾难性的后果[3]。例如温哥华证券交易所由于浮点数显著误差,股指出现混乱;在阿丽亚娜-5运载火箭发射时,由于浮点数显著误差,处理器运算溢出,火箭在发射37秒后引爆自毁;在海湾战争中,一枚爱国者导弹由于浮点数显著误差偏航,造成28人死亡;无人驾驶汽车因误差积累而失控撞毁等等。因此,对浮点数的最大误差进行及时有效的检测和评估具有重要意义。

最大误差检测不仅可以有效评估浮点数计算的可靠性和准确性,帮助程序员和工程师发现和解决浮点数计算中可能

到稿日期:2024-08-05　返修日期:2024-10-16
通信作者:许瑾晨(atao728208@126.com)

存在的各类误差问题,也可以帮助浮点程序开发者了解浮点运算的精度范围,从而采取有效的措施来减小浮点误差对程序执行结果的影响[4]。同时在高性能计算领域,最大误差检测也可以用来验证一些科学计算的正确性和有效性。传统最大误差检测主要是基于串行计算模式策略研发的,如果采样点太多,计算时间会增加,计算的时效性会降低;如果采样点太少又很难发挥出计算策略的效能[5]。随着国产申威超算平台的优化升级和并行计算策略的迭代发展,并行思维已经成为解决问题基本遵循的方式。

本文将并行计算模式成功运用到浮点数最大误差检测算法领域,充分利用并行计算强大的计算能力,大幅提升采样点数量,并且调用高精度数学函数库来计算精确值,这样不仅保证了计算精度,也提升了计算性能,最后利用信息传递接口函数返回最优结果。

采用并行计算模式并结合浮点采样策略进行浮点数最大误差检测研究,可以在精度和性能两个关键指标上取得突破,为衡量浮点数计算指标提供新的检测模式。本文提出的基于局部性原理的最大误差并行检测方法(Maximum Error Parallel Detection Method Based on Locality Principle,MEPDLP),是一款充分利用和发挥国产申威超算平台的计算性能,结合动态浮点数采样策略而研发的浮点数最大误差检测工具。

本文的贡献如下:

1)在浮点数计算最大误差检测方法研究领域引入并行计算思维,拓展了浮点数计算最大误差检测方法的研究思路和检测手段。

2)充分利用申威平台的主从架构特点,编写可在申威平台上高效运行的最大误差检测工具;可以同步做到提升计算的精度和性能,检测出的最大误差值高于当前主流检测工具,并行加速比突破100000%。

3)借鉴计算机存储局部性原理,设计浮点数动态采样策略,提升浮点误差检测效果。

本文第2章简要说明了浮点数最大误差研究的现状;第3章主要介绍了浮点数的基础知识;第4章给出了MEPDLP算法流程和实现方法;第5章给出了对比实验和结果分析;最后总结算法中存在的不足和后期改进的方向。

2 研究现状

浮点数最大误差检测一直是计算机科学技术的研究热点。定位浮点误差在数据计算中的关键环节、建立误差累积和传播模型、优化误差检测算法等研究手段,是目前比较常见的形式和方向。综合而言,客观评价浮点数计算的两大指标就是计算精度和计算性能[6]。在运算条件一定的情况下,计算精度越高,计算用时越长,相应性能就会下降;计算精度降低,计算用时就会缩短,相应性能就会提升。目前,高性能计算研究领域主要通过创新计算策略、优化混合精度策略、提升计算机计算性能等研究方向来平衡二者的矛盾[7]。

利用启发式搜索算法来检测浮点数最大误差的方法也是当前比较常见的研究手段。在AutoRNP工具中,通过差分演化算法和马尔可夫链蒙特卡罗检测精度缺陷,通过搜索触发最大误差的输入来检测最大误差[8]。S3FP工具是基于二进制引导随机测试来搜索最大误差[9]。在ATOMU工具中,通过使用原子状态函数来搜索触发最大误差的输入[10],将浮点数计算最大误差检测问题转换为搜索触发较大原子状态函数的问题。Herbie是目前比较先进的浮点误差检测工具[11],同时具备精度优化的作用,其原理是通过随机测试定位误差热点区间,在相应区间内运用表达式重写的方法提升计算精度。这些算法虽然很成熟,但是也存在一些不足,如二进制引导随机测试需要预定义搜索时间,时间长短直接决定搜索结果的效果;原子状态函数会产生误报情况,这些因素都制约了启发式搜索算法检测浮点数最大误差的效果。

为了解决上述浮点数最大误差检测方法带来的问题和不足,本文提出了一种浮点数最大误差并行检测方法,充分发挥并行计算的强大计算能力,科学动态地进行浮点数采样,精准可靠地检测出浮点数最大误差。从抽样统计学定理来讲,提高采样点的规模和采样频率,提升采样点的代表性,就会大幅提高计算结果的科学性和客观性。本文的研究思路是,利用并行计算的强大算力大幅增加采样点的规模来计算出误差分布的热点趋势,然后在误差热点所在的局部区间再次进行区间采样与最大误差定位操作,最终得到全局最大误差值。这种方法高效可靠,精度和性能都很高,为浮点数最大误差检测提供了一种有效的方法和模式。

3 基础知识

3.1 局部性原理

中央处理器(CPU)中的存储(Cache)可以有效支撑CPU进行高速数据处理,虽然它的速度很快,但是其容量极小,这就需要Cache及时高效地与内存进行数据交换来保证CPU数据处理进程不间断[12]。如何保证内存中的数据被高效精准命中,这就是计算机局部性原理的强大威力所在。局部性原理分为时间局部性原理和空间局部性原理。当内存中的数据在当前时间被使用,那么在后续的一段时间也有可能被使用;当某一数据块被使用,那么它相邻的数据块被使用的可能性也很大。依据这一原理设计的数据交换协议,极大地提高了数据交换的命中率,提升了计算机内存的使用效率。局部性原理在计算机硬件发展中占有非常重要的地位,对其他领域也有着深刻的借鉴作用。

3.2 浮点数表示原理

浮点数(Floating-point Number)是实数的一个子集,是基于二进制和科学计数法原理表示的一种实数[13]。在计算机中,浮点数由3个部分组成,即符号位(Sign)、指数位(Exponent)和尾数位(Significand),如表1所列,是常见类型浮点数的表示。

表1 IEEE 754浮点数表示

Table 1 IEEE 754 floating-point number representation

浮点数	位数/bits	符号位	指数位	尾数位
双精度	64	1	11	52
单精度	32	1	8	23
半精度	16	1	5	10

浮点数广泛使用的是IEEE 754标准,规定其规格化表示形式为:

$f=(-1)^S\times M\times 2^E$

其中，S 为符号位，占据 1 个比特位表示浮点数为正或负，S 取值为 0 或 1，当 $S=0$ 时，浮点数取值为正，$S=1$ 时，浮点数取值为负；$M=d_0.d_1d_2\cdots d_{p-1}$ 为尾数位，即浮点数的有效数字，p 为有效数字的长度，表示浮点数的精度；E 为指数位，直接决定浮点数取值的范围大小。最常用的浮点数类型为双精度(Double)、单精度(Float)和半精度(Half Float)，符号位、指数位和尾数位如图 1 所示。

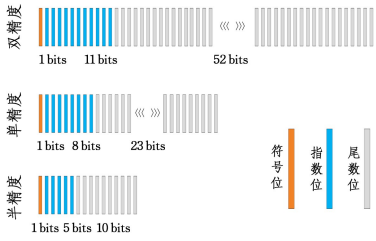


图 1 IEEE 754 常用浮点数类型

Fig. 1 IEEE 754 common floating point number types

3.3 浮点误差

最大误差检测就是检测待测浮点对象在特定数学库下，依据某种运算策略的计算值和客观实际值之间误差的最大值[14]。计算机使用二进制数计算，而十进制中的大部分数字在二进制中无法精确表示，会导致舍入误差。这种误差会导致数值计算结果不准确，这种不准确所带来的影响程度会因浮点计算误差大小而不同，特别是在科学计算、金融结算、精确制导、电子商务等领域，对浮点计算精度极高，需要将浮点计算的误差控制在一定的范围内。

通过检测浮点数计算的最大误差，可以了解计算机在进行浮点数运算时的精度限制，并采取适当的措施来避免误差过大或对误差敏感的运算。例如，可以使用更高精度的数据类型或算法，或者在必要时进行误差调整或容错处理。在高性能计算研究领域，浮点程序的计算值和理论精确值之差的绝对值就是绝对误差($Error_{abs}$)[15]，绝对误差与真实值的比值就是相对误差($Error_{rlt}$)[16]，浮点数的相对误差通常用百分比表示，它反映了计算机进行浮点数运算的精度限制。给定一个算术表达式 $f(x)$，假设在 $x=x_0$ 下被计算机获得的正确舍入值或者精确值表示为 $F(x_0)$，计算值表示为 $f(x_0)$，那么该表达式的绝对误差($Error_{abs}$)和相对误差($Error_{rlt}$)的定义如式(1)和式(2)所示：

$$Error_{abs}=|f(x_0)-F(x_0)| \tag{1}$$

$$Error_{rlt}=\left|\frac{f(x_0)-F(x_0)}{f(x_0)}\right| \tag{2}$$

本文对比实验中采用了 Bits 误差($Error_{Bits}$)。Bits 误差的定义如式(3)和式(4)所示：

$$DB_{number}\{F(x),f(x)\}=|\{a_i\in\mathbb{F}|\min(F(x),f(x))\leqslant a_i\leqslant\max(F(x),f(x))\}| \tag{3}$$

$$Error_{Bits}\{F(x),f(x)\}=\log_2(DB_{number}\{F(x),f(x)\}) \tag{4}$$

其中，DB_{number} 表示精确值 $F(x)$ 与计算值 $f(x)$ 之间相差的浮点数个数，$Error_{Bits}$ 是度量精确值 $F(x)$ 与计算值 $f(x)$ 两者的数值在数位上有多少比特位是不同的。例如双精度浮点数 1.0 和 2.0 之间相差 4 503 599 627 370 496 个浮点数，对其取 log2 得 52，说明 1.0 和 2.0 之间相差了 52 位，Bits 误差为 52。

3.4 申威并行处理器概述

并行计算指同时使用多种计算资源解决计算问题的过程，它是一种一次可执行多个指令的算法，目的是提高计算速度，并通过扩大问题求解规模，解决大型而复杂的计算问题。

国产申威处理器是执行并行计算的核心部件，采用“主处理器+加速部件”的主从核异构架构[17]。每个处理器有 4 个核组，一个核组上有 64 个计算从核与 1 个管理主核。其中，主核主要负责管理和计算任务的分发与结果汇总，也可以担负同从核一样的角色；从核主要负责计算功能的实现。一般来讲，核数越多，计算性能越好，并行性越高。可以把一个计算任务分解到不同的核组上，每一个核组通过主核把计算任务细分到每一个从核上。这样就实现了两级并行计算的任务。核组执行的程序为进程，每个进程都有自身的进程号，如 rank 0。从核执行的程序块是线程，每个线程都有自身的线程号，如 my-id 0。并行计算模式一般有主从加速并行模式、主从协同并行模式、主从异步并行模式和主从动态并行模式。

并行计算相比串行计算的优势主要体现在以下几个方面。

1)执行速度快：并行计算可以同时执行多个任务或操作，从而在处理大量数据时显著提高速度。

2)资源利用率高：并行计算可以充分利用计算机的多核处理能力，同时执行多个任务，从而提高计算机资源的利用率。

3)任务分解和并行执行：并行计算可以将一个大型任务分解成多个子任务，并将这些子任务分配给不同的处理单元并行执行。这样可以同时处理多个子任务，从而缩短总的处理时间。

4)提高计算效率：并行计算可以在处理大量数据时提高计算效率，因为它可以同时处理多个任务或操作，缩短了处理时间。

需要注意的是，并行计算并非没有缺点。比如，并行算法的缺点是实现复杂，需要考虑任务分解、通信、同步等问题。在实际应用中，需要综合考虑具体的需求和场景来选择适合的算法和计算方式。

4 MEPDLP 算法的实现流程

MEPDLP 算法是基于并行计算模式实现的，共分为输入、并行采样子程序、并行误差计算子程序、并行计算结果处理子程序和输出 5 个关键处理流程。MEPDLP 算法严格依据浮点数分布规律，借鉴计算机存储局部性原理的思想在输入区间内进行动态并行采样操作；然后，按照误差模型计算采样点处的误差值，依据排序筛选规则定位出各局部区间的局部最大误差值；最后，利用消息传递接口函数并行归约计算出全局最大误差值和对应的区间。这一并行计算处理过程支持自定义循环操作，可以对返回的区间再次进行采样、误差计算、最大误差定位等操作，最终输出整轮检测的最大误差值。具体实现流程如图 2 所示。

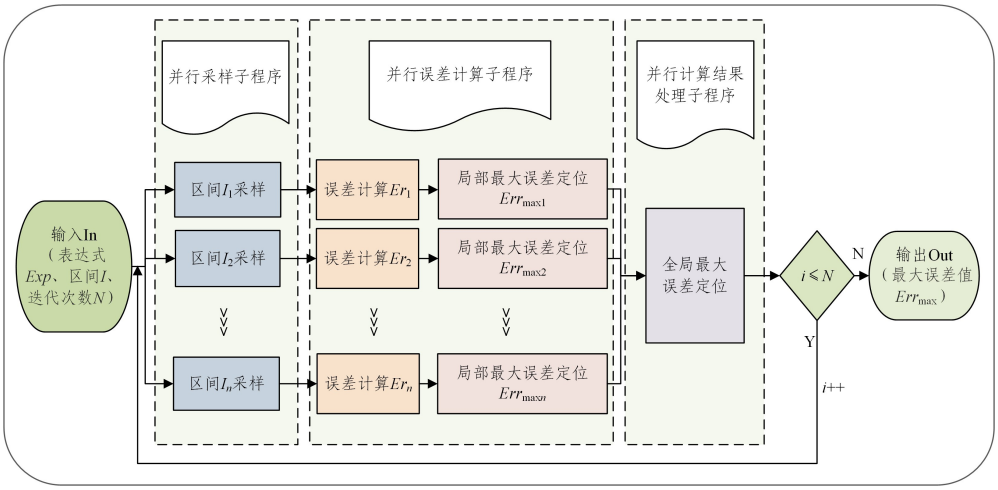


图 2 MEPDLP 算法的流程

Fig. 2 Flow chart of MEPDLP algorithm

4.1 输入

MEPDLP 算法是面向单参数表达式的浮点数最大误差检测工具，程序有 3 个输入参数，分别是表达式 Exp、对应的输入区间 I 和循环次数 N。其中 N 的取值为 1 和 2 两个值，分别对应两种不同的应用模式。当 N 取值为 1 时，MEPDLP 算法选用运算性能优先策略，可以应用于快速定位浮点数最大误差值的量级。当 N 取值为 2 时，MEPDLP 算法选用计算精度优先策略，可以保证在符合性能要求的前提下，进一步提升浮点数最大误差值的精度。

4.2 并行采样子程序

并行采样子程序根据局部性原理，利用申威平台的计算核组所属的进程号（Process-ID）和计算从核所属的线程号（Thread-ID）对输入区间 I 进行科学合理的划分，得到局部区间（$Range_{Local}$）和末端区间（$Range_{End}$）。进程号（Process-ID）将输入区间 I 划分为不同的局部区间（$Range_{Local}$）；在每个局部区间内，通过线程号（Thread-ID）将所属的局部区间划分为不同的末端区间（$Range_{End}$）。具体划分方法如图 3 所示。

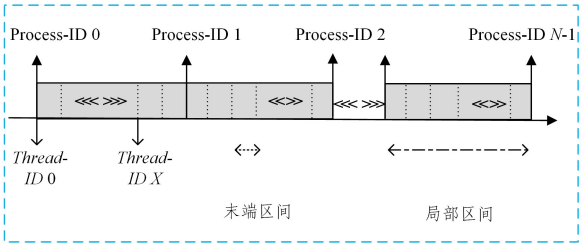


图 3 区间划分示意图

Fig. 3 Schematic diagram of interval division

各计算核组和计算从核只需要处理自己所属区间的采样、误差计算、最大误差值筛选等计算任务即可，各进程和线程之间独立并行运行，在关键节点完成计算结果的并行同步和结果归约操作。

各计算核组在对应的局部区间内，依据浮点数“近 0 端密，远 0 端疏”的分布规律并行执行浮点数采样操作。浮点数并行采样的核心原则是，严格遵循“靠近‘0 端’的局部区间长度小，采样密度大；远离‘0 端’的局部区间长度大，采样密度小”这一采样协议。这样设计很好地弥补了采用随机采样和均匀采样等简单采样操作带来的采样精度差和样本数据代表性弱的缺点，提升了浮点数采样的效果，具体执行过程如图 4 所示。这些采样点作为后续并行误差计算程序的输入并完成误差计算。

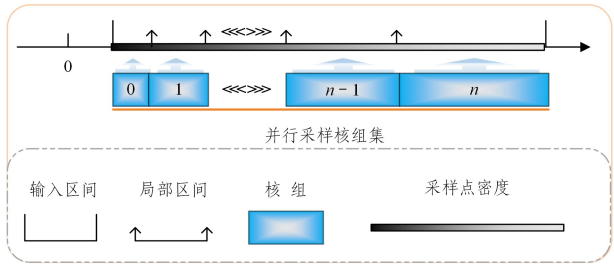


图 4 核组并行采样示意图

Fig. 4 Schematic diagram of kernel group parallel sampling

4.3 并行误差计算子程序

并行误差计算子程序由误差计算和局部最大误差定位两个模块组成。每个计算从核独立并行计算出各自末端区间内的最大误差值，利用排序算法筛选出其中的最大值作为对应计算核组中的局部最大误差值 $Err_{localmax}$（局部区间内误差的最大值）。

4.3.1 误差计算

本文计算了 3 种误差，分别是绝对误差、相对误差、Bits 误差。对于一个采样点来讲，调用双 Double 高精度数学计算函数库（128 位计算精度）来计算该点的值，并将其作为精确值 $F(x)$，这里应用双 Double 高精度数学计算函数库而非常用的 Mpfr 数学库，原因是双 Double 高精度数学计算函数库拥有更好的计算性能。同时，调用标准数学库计算采样点的计算值 $f(x)$。二者做差得出该点的绝对误差 $Error_{abs}=|F(x)-f(x)|$。绝对误差与对应计算值的比值就是该点的相对误差 $Error_{rlt}=\left|\dfrac{F(x)-f(x)}{f(x)}\right|$。以相对误差为例，其计算流程如图 5 所示，示例代码如算法 1 所示。因为对比工具 Herbie 输出的是 Bits 误差，对比实验中根据精确值和计算值之间相差的浮点数个数可以得到 Bits 误差（$Error_{Bits}$）。由于双精度浮点数只有 64 位，因此任意两个浮点数之间的 $Error_{bits}$ 都大于等于 0，小于 64。相比绝对误差，Bits 误差可

以保持度量一致性，也避免了相对误差会出现除 0 错误，同时也支持 NaN 和±∞。

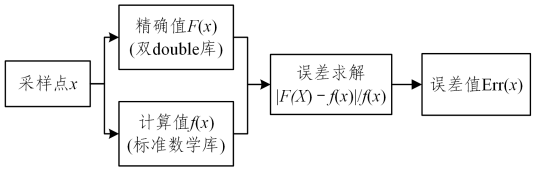


图 5 相对误差计算流程

Fig. 5 Flow chart of relative error calculation

算法 1 单点误差计算(以相对误差为例)

输入：采样点 x

输出：单点误差，error(x)

```
1. c_dd_add(x,a1,b1);
2. F(x)=b1;
3. f(x)=x+a1;
4. error(x)=(F(x)-f(x))/ f(x).
```

4.3.2 局部最大误差定位

局部最大误差定位就是并行计算出每一个核组所属局部区间内的最大误差值 $Err_{local_{max}}$。每个计算从核在相应的末端区间内采样 1000 万个点，运用快速排序算法得到每个从核在相应的末端区间内的最大误差值 $Err_{end_{max}}$，然后将各末端最大误差值存储在对应的容器中，最后比较筛选出对应核组的局部最大误差值 $Err_{local_{max}}$，具体操作过程如图 6 所示。

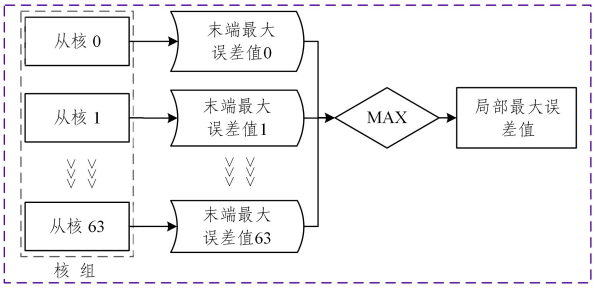


图 6 局部最大误差定位流程

Fig. 6 Flow chart of local maximum error location

4.4 全局最大误差定位

在整个输入区间内的最大误差值为全局最大误差 Err_{max}，这也是整个系统要求的最终目标值。各核组经过并行局部最大误差定位后，计算出各自的局部最大误差值 $Err_{local_{max}}$，然后调用特定函数实现各核组计算同步后，并利用消息传递接口并行规约函数返回所有核组的局部最大误差值 $Err_{local_{max}}$ 中的最大值和该最大值所处的末端区间，这个值就是本轮并行计算中得到的最大误差值，该末端区间就是执行下一次循环计算的输入区间，具体流程如图 7 所示。

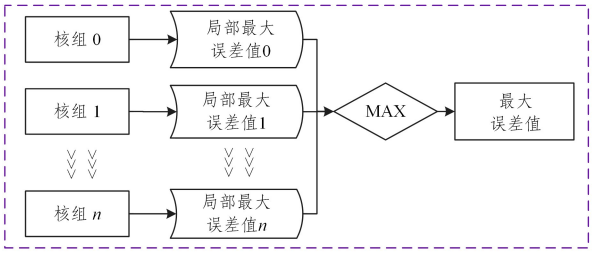


图 7 最大误差值定位流程

Fig. 7 Flow chart of maximum error location

依据空间局部性原理，在现有的最大误差值对应的采样点附近仍将有较大的概率出现更大的误差值点[18]，因此对返回的末端区间，再次循环执行程序代码，进行并行区间采样、误差计算等代码，返回整个计算过程中的全局最大误差值 Err_{max}。

5 实验分析

本实验是在国产申威某超算平台上进行的，该平台的每个中央处理器(CPU)中包含 4 个核组，每个核组含 64 个计算从核与 1 个管理主核，实验程序采用主从架构两级并行的模式。程序采用并行 C 语言编写，分为主核代码和从核代码两个独立子程序，二者分别编译后需要再次混合编译方可执行程序，得出最大误差和运行时间。在对比实验中，与对应的串行程序对比性能，为了验证 MEPDLP 检测浮点算术表达式误差的有效性，本文使用 FPBench 基准测试集[19]中 32 个单参算术表达式与目前最先进的误差检测工具 Herbie，S3FP，HSED 进行精度评估和性能评估。其中 Herbie 采用的是 Bits 误差，S3FP 和 HSED[20]采用的是相对误差。

5.1 测试用例

本文的基准测试信息如表 2 所列，其中 FPBench 基准测试集中共有 46 个单参算术表达式，本文选取了 32 个作为测试对象，排除了 14 个包含循环或判断或表达式重复的基准。检测区间按照 FPBench 给定的默认区间，如果 FPBench 没有给出默认区间则按照[0.01,100]区间，该区间为常用区间。

表 2 测试用例信息

Table 2 Test case information

编号	FPBench	测试区间
1	sqroot	[0,1]
2	sqrt_add	[1,1000]
3	exp1x	[0.01,0.5]
4	exp1x_log	[0.01,0.5]
5	NMSEexample37	[0.01,100]
6	NMSEproblem336	[0.01,100]
7	NMSEexample39	[0.01,100]
8	NMSEproblem341	[0.01,100]
9	NMSEsection311	[0.01,100]
10	NMSEproblem345	[0.01,100]
11	NMSEproblem337	[0.01,100]
12	verhulst	[0.1,0.3]
13	predatorPrey	[0.1,0.3]
14	logexp	[0.01,8]
15	sine	[−1.570796326 79,1.570796326 79]
16	carbonGas	[0.1,0.5]
17	NMSEproblem341	[0.01,100]
18	NMSEexample38	[0.01,100]
19	NMSEproblem334	[0.01,100]
20	NMSEproblem333	[0.01,100]
21	NMSEproblem331	[0.01,100]
22	NMSEexample36	[0.01,100]
23	NMSEexample35	[0.01,100]
24	NMSEexample34	[0.01,100]
25	NMSEexample31	[0,100]
26	test05_nonlin1_r4	[1.000 01,2]
27	test05_nonlin1_test2	[1.000 01,2]
28	intro-example-mixed	[1,999]
29	sineOrder3	[−2,2]
30	bsplines3	[0,1]
31	NMSEexample310	[0.001,1]
32	NMSEproblem343	[0.001,1]

5.2 精度结果及分析

MEPDLP与目前几款先进的误差检测工具Herbie，S3FP，HSED进行对比。针对相同的测试用例，在相同的输入区间内，将不同检测工具检测出的最大误差值 Err_{max} 进行对比。哪款检测工具检测出的最大误差值更大，说明其检测能力更优。

5.2.1 MEPDLP与Herbie的精度对比实验

Herbie输出的是Bits误差，因此MEPDLP与Herbie对比的是Bits误差，实验对比效果如图8所示，图中的横坐标是选取的测试用例(因为区域有限，按照5.1节中测试用例的顺序，选用编号代替测试用例的名称)，纵坐标是Bits误差的值。

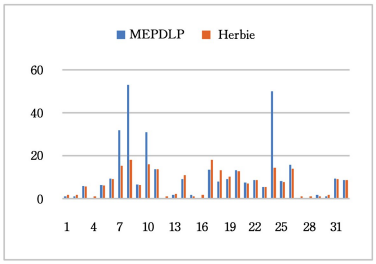


图8 MEPDLP和Herbie的精度对比

Fig. 8 Precision comparison between MEPDLP and Herbie

在MEPDLP与Herbie对比实验的32个测试用例中，MEPDLP有19项优于Herbie，有5项是大幅超越，6项差距较小。实验表明，MEPDLP的检测精度整体效果强于Herbie的检测精度。

5.2.2 MEPDLP与S3FP的精度对比实验

S3FP输出的是最大相对误差，因此MEPDLP与S3FP对比的是最大相对误差。MEPDLP与S3FP的对比实验效果如图9所示，图9中的横坐标是测试用例的编号值，纵坐标为最大相对误差(x)的数学变换 $f(x)=\log_{10}(x)+20$。

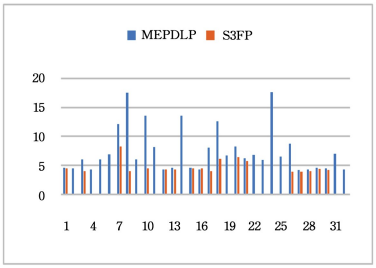


图9 MEPDLP和S3FP的精度对比

Fig. 9 Precision comparison between MEPDLP and S3FP

在MEPDLP与S3FP的对比实验中，MEPDLP有30项优于S3FP，而且32个测试用例都可以检测出最大误差值，S3PF工具需要指定TIMEOUT参数，以控制搜索时间预算，有14项测试用例无法得出最终结果。实验表明，MEPDLP的检测精度优于S3FP工具的检测精度。

5.2.3 MEPDLP与HSED的精度对比实验

HSED输出的是最大相对误差，因此MEPDLP与HSED对比的是最大相对误差。MEPDLP与HSED的对比实验效果如图10所示，图10中的横坐标是测试用例的编号值，纵坐标为最大相对误差(x)的数学变换 $f(x)=\log_{10}(x)+20$。

在MEPDLP与HSED的对比实验中，在32个测试用例中，MEPDLP有24项优于HSED，同时有2项实验结果相等。实验表明，MEPDLP工具的检测精度优于HSED工具的检测精度。

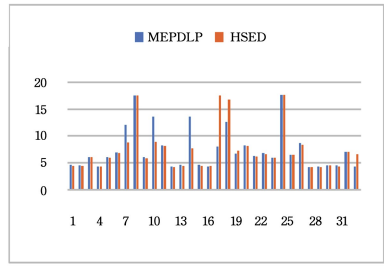


图10 MEPDLP和HSED的精度对比

Fig. 10 Precision comparison between MEPDLP and HSED

精度对比实验结果表明，MEPDLP工具在检测浮点数计算最大误差方面具有较高的精度，实验精度高于Herbie，S3FP，HSED检测工具。MEPDLP工具和S3FP，HSED检测工具的精度对比实验的实验效果优于MEPDLP工具和Herbie的精度对比实验效果，这说明MEPDLP工具检测相对误差方面的精度优于Bits误差检测。

5.2.4 MEPDLP检测精度的原因分析

通过对比实验可以看出，MEPDLP工具相比于Herbie，S3FP，HSED工具在检测最大浮点误差方面有着较高的精度。深刻分析其中的原因主要是，MEPDLP工具借鉴了计算机内存设计的局部性原理的思想精髓，采用并行计算模式，采样点数量规模大，因此大幅提高了命中最大误差点的概率。为了充分印证分析的正确性，设计了以下验证实验。

从实验用例中均匀选取了第1，6，11，16，21，26，31号测试用例，分别计算了采样规模在 3.84×10^{10}，3.84×10^{9} 和 3.84×10^{8} 情况下的各实验用例检测出的最大误差值(相对误差)，经过数学线性变换将结果缩放至0到20之间，保证同一测试用例在不同采样规模下检测出的最大误差值相对大小关系，直观展现出各测试用例在不同采样点规模下检测出最大误差值的变化趋势，实验结果如图11所示。

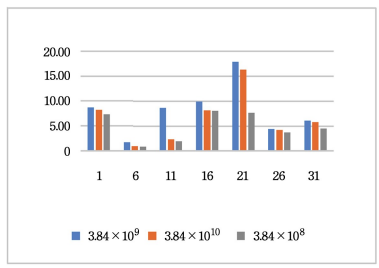


图11 最大误差值与采样规模之间的变化趋势

Fig. 11 Variation trend between maximum error value and sampling scale

5.3 性能结果及分析

本节对比实现相同功能的基于并行模式和串行模式的不同程序计算用时。因为并行计算程序的性能远高于普通的串行计算程序(这也正是本文选择并行模式检测浮点数最大误差的原因)，所以本节只选取了一个测试用例来对比并行程序

和串行程序的性能量级。

在对比实验中，选取的测试用例为 FPBench 基准测试集中的 NMSEproblem343。两个对比程序的采样点数量均为 3.84×10^{10}，测试区间为[0.001,1]。经实验测得并行版本的程序用时为 53.46 s，对应串行版本的程序用时为 60748.8 s，合计为 16.87 h。二者的差距为 1136.3 倍，实验的具体信息如表 4 所列，对比效果如图 12 所示。

表 3 并行程序和串行程序的性能对比

Table 3 Performance comparison between parallel and serial programs

程序类型	采样点数量	程序用时/s
并行版本	3.84×10^{10}	53.46
串行版本	3.84×10^{10}	60748.8

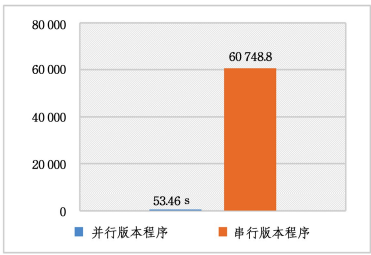


图 12 并行程序和串行程序的性能对比

Fig. 12 Performance comparison between parallel and serial programs

MEPDLP 工具相对于相同功能的串行程序拥有很高的性能优势，这也正是并行计算的独有优势。二者在时间方面相差 1136.3 倍，并不是二者计算核数量之比，这是因为并行程序在执行时需要通信开销和结果汇总时间，调用 MPI 函数也需要一定的时间，这些都是限制并行程序加速比的重要因素，在后续的研究中可以进一步优化和完善并行程序逻辑和代码执行，进一步提高并行程序的性能。

综合而言，MEPDLP 工具在浮点数最大误差检测方面拥有较好的精度和性能。

结束语 本文提出的基于局部性原理的最大误差并行检测方法(MEPDLP)，充分发挥了并行计算强大的算力，使采样点的数量和并行规模指数级增加，利用局部性原理在误差热点的邻域内继续细分并大规模地采样来检测更大的误差值。也许因为采样的偶然性和局部采样的均匀性，使得存在一定的概率漏采真正的最大误差值对应的输入点，把局部极大值作为全局最大值处理[21]，这也是在对比实验中存在一些测试用例检测结果的精度低于对比工具的原因。

在后期研究中，需要引入新的采样策略，使得程序能够跳出局部极大值，使得最终结果趋近客观最大值。同时还需要进一步优化并行策略，引入向量计算，提升并行计算的规模和并行度。在后续的最大误差检测算法研究中，将引入经典搜索算法并加以并行化改造，来改进最大误差检测的方法。当然，对于本文中的一些测试用例存在结果精度不足的情况，进行深入研究和分析后改进代码，以提升并行规模，并进一步完善实验结果。

参 考 文 献

[1] KOTIPALLI P V, SINGH R, WOOD P, et al. AMPT-GA: Automatic Mixed Precision Floating Point Tuning for GPU Applications[C]// Proceedings of the ACM International Conference on Supercomputing. New York: ACM, 2019: 160-170.

[2] DAMOUCHE N, MARTEL M. Salsa: An Automatic Tool to Improve the Numerical Accuracy of Programs[J]. Automated Formal Methods, 2018, 5: 63-76.

[3] SANCHEZ-STERN A, PANCHEKHA P, LERNER S, et al. Finding Root Causes of Floating-Point Error[C]// Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, 2018: 256-269.

[4] GUO H, RUBIO-GONZÁLEZ C. Exploiting Community Structure for Floating-Point Precision Tuning[C]// Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM, 2018: 333-343.

[5] SAIKI B, FLATT O, NANDI C, et al. Combining Precision Tuning and Rewriting[C]// 2021 IEEE 28th Symposium on Computer Arithmetic(ARITH). 2021: 1-8.

[6] MAGRON V, CONSTANTINIDES G, DONALDSON A. Certified Roundoff Error Bounds Using Semidefinite Programming [J]. ACM Transactions on Mathematical Software, 2017, 43(4): 34.

[7] DARULOVA E, HORN E, SHARMA S. Sound Mixed-Precision Optimization with Rewriting[C]// Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems. 2018: 208-219.

[8] YI X, CHEN L Q, MAO X G, et al. Efficient automated repair of high floating-point errors in numerical libraries[C]// Proceedings of the ACM on Programming Languages. 2019: 1-29.

[9] CHIANG W F, GOPALAKRISHNAN G, RAKAMARIC Z, et al. Efficient search for inputs causing high floating-point errors [J]. ACM SIGPLAN Notices, 2014, 49(8): 43-52.

[10] ZOUD, ZENG M, XIONG Y, et al. Detecting floating-point errors via atomic conditions [C]// Proceedings of the ACM on Programming Languages. 2019: 1-27.

[11] PANCHEKHA P, SANCHEZ-STERN A, WILCOX J R, et al. Automatically improving accuracy for floating point expressions [J]. ACM SIGPLAN Notices, 2015, 50(6): 1-11.

[12] WAGNER J, NISSAN M I, RASIN A. Database memory forensics: Identifying cache patterns for log verification[J]. Forensic Science International: Digital Investigation, 2023, 45(S): 301567.

[13] MENON H, LAM M O, OSEI-KUFFUOR D, et al. ADAPT: Algorithmic Differentiation Applied to Floating-Point Precision Tuning[C]// International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2018.

[14] LIUX, GUO H, SUN R J, et al. Large-scale Application Characteristics Analysis and E-level Scalability Research of "Shenwei Taihu Light" computer system [J]. Journal of Computer Science, 2018, 41(10): 2209-2220.

[15] SOLOVYEV A, JACOBSEN C, RAKAMARIĆ Z, et al. Rigorous Estimation of Floating-Point Round-off Errors with Symbolic Taylor Expansions[C]// Proceeding in 20th International

Symposium on Formal Methods. New York: ACM, 2015: 532-550.

[16] CHIANG W F, BARANOWSKI M, BRIGGS I, et al. Rigorous floating-point mixed-precision tuning[C]// Symposium on Principles of Programming Languages. New York: ACM, 2017: 300-315.

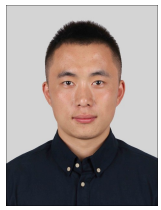
[17] IZYCHEVA A, DARULOVA E. On Sound Relative Error Bounds for Floating-Point Arithmetic[C]// 2017 Formal Methods in Computer Aided Design. IEEE, 2017: 15-22.

[18] CHERUBIN S, CATTANEO D, CHIARI M, et al. TAFFO: Tuning Assistant for Floating to Fixed Point Optimization[J]. IEEE Embedded Systems Letters, 2020, 12(1): 5-8.

[19] CATTANEO D, BELLO A D, CHERUBIN S, et al. Embedded Operating System Optimization through Floating to Fixed Point Compiler Transformation[C]// 21st Euromicro Conference on Digital System Design. IEEE, 2018: 172-176.

[20] ZHANG Z Y, XU J C, HAO J W, et al. Hierarchical search algorithm for error detection in floating-point arithmetic expressions [J]. Supercomput, 2023, 80: 1183-1205.

[21] DENIS C, CASTRO P D O, PETIT E. Verificarlo: Checking Floating Point Accuracy through Monte Carlo Arithmetic[C]// 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH). IEEE, 2016: 55-62.



JI Liguang, born in 1992, master. His main research interests includes high-performance computing.

XU Jinchen, born in 1987, Ph.D, associate professor. His main research interests includes high-performance computing.

（责任编辑：喻藜）

数智赋能，智启未来！CCF 2025 中国数字服务大会在烟台举办

2025 年 8 月 19 日至 21 日，由中国计算机学会（CCF）主办，CCF 服务计算专业委员会、烟台大学、哈尔滨工业大学（威海）联合承办的 CCF 2025 中国数字服务大会于山东烟台盛大举行。本次大会首次实现注册参会人数突破 1 000 人，创下历届参会人数的最高纪录！

2025 年 8 月 19 日至 21 日，由中国计算机学会（CCF）主办，CCF 服务计算专业委员会、烟台大学、哈尔滨工业大学（威海）联合承办的 CCF 2025 中国数字服务大会于山东烟台盛大举行。本次大会首次实现注册参会人数突破 1 000 人，创下历届参会人数的最高纪录！

中国工程院院士何友，CCF 会士、中国工程院院士赵春江，CCF 会士、中国科学院外籍院士、新加坡科学院院士、新加坡工程院院士黄铭钧，深圳理工大学计算机科学与控制工程学院院长潘毅，俄罗斯科学院院士 Igor Sheremet，IEEE 会士、墨尔本大学教授 Rajkumar Buyya，山东大学控制科学与工程学院院长张承慧，浙江大学软件学院院长尹建伟，武汉大学教授李兵，吉利汽车研究院人工智能中心主任陈勇，字节跳动 Code AI 高级技术专家刘杰等等国内外学术界和工业界的顶尖专家，以及来自全国高校、科研院所、知名企业的 1084 名代表齐聚一堂，共同探讨“数智服务：重塑产业体系，赋能社会治理”这一时代命题。

大会共收到论文投稿 116 篇，录用 49 篇，特别邀请了 4 位中外院士以及 7 位顶尖专家学者作主旨报告，组织了 16 场论坛、1 项保研/考研之夜活动以及 1 场创新大赛，呈现了 100 多场精彩报告。全方位展示了我国数字服务领域的最新研究成果与产业实践。

此外，本次大会还颁发了 CCF 服务计算专委博士学位论文激励计划、CCF 服务计算专委教学创新激励计划、CCF 服务计算专委青年学者激励计划、CCF 服务计算专委青年科技奖、CCF 服务计算专委海外科技人物奖等多个奖项。组织的 CCF 中国服务计算创新大赛、企业展览等特色活动也进一步增强了大会的互动性和参与度。经 CCF 服务计算专委会常务委员投票表决，下一届 CCF 中国数字服务大会将于 2026 年 8 月在四川成都举行。

据 CCF 微信公众号