СПОСОБ И ПРОГРАММНЫЙ ПАКЕТ ДЛЯ ОРГАНИЗАЦИИ РАЗРЯДНО-ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ ВЫСОКОЙ ТОЧНОСТИ

К.С. Исупов

Введение.

Одной из наиболее важных задач при выполнении ресурсоемких численных расчетов на суперкомпьютерах является обеспечение требуемой точности вычислений. При выполнении крупных инженерных и научных расчетов важно быть уверенным, что полученные в конечном итоге результаты будут корректными и пригодными для дальнейшего использования. Это привело к весьма активному развитию актуального направления вычислительной математики – высокоточным и безошибочным вычислениям. Ввиду того, что подавляющее большинство численных методов оперируют с вещественными числами, основной задачей данного направления является обеспечение высокой точности вычислений в форматах с плавающей точкой (ПТ). Результатом решения данной задачи в настоящий момент, как правило, является применение *специальных многоразрядных форматов* данных и программных решений, построенных на их основе, в случаях, когда точности, обеспечиваемой машинными форматами с ПТ оказывается недостаточно.

Наибольшее распространение среди всех прочих получили методы и программные средства, обеспечивающие высокую точность вычислений с ПТ посредством применения длинной позиционной арифметики (когда мантисса представляется одним многоразрядным позиционным числом, либо несколькими числами, а порядок рассматривается как отдельная величина). К числу таких средств можно отнести широко используемые библиотеки для языков программирования: GMP, MPFR, HPA. Как показывают эксперименты [1], данные средства обеспечивают корректную (без потери точности) обработку операндов с ПТ практически неограниченной длины. Главный недостаток таких средств – катастрофическое падение быстродействия при увеличении разрядности обрабатываемых чисел (зависимость времени выполнения арифметических операций от точности в большинстве случаев носит экспоненциальный характер) [1]. Это существенным образом затрудняет решение высокоточных задач за приемлемое время, и поэтому возникает необходимость разработки новых способов и алгоритмов, а также программных средств на их основе, которые:

- во-первых, являются максимально приближенными к форматам, используемым в современных процессорах (наибольшее распространение получили двоичные форматы стандарта IEEE 754), но лишены недостатков последних, связанных с нехваткой точности: даже восьмикратной точности (4*double) зачастую оказывается недостаточно для корректного решения численной задачи, не говоря уже о двойной точности (double), поддерживаемой аппаратно производителями современных универсальных процессоров;
- во-вторых, обеспечивают минимизацию зависимости времени выполнения операций от точности;
- в-третьих, необходимо, чтобы данные способы позволяли выполнять эффективное распараллеливание вычислений на уровне арифметических операций, так как не всегда алгоритмический параллелизм задачи позволяет при ее решении эффективно использовать все ресурсы многопроцессорной вычислительной системы, построенной на базе многоядерных вычислителей, поэтому проблема полномерного использования ресурсов суперкомпьютеров остается актуальной и будет лишь усиливаться на пути к преодолению экзафлопсного барьера производительности.

В работе предлагается новый подход к организации высокоточных параллельных вычислений с плавающей точкой – модулярно-позиционный (квазимодулярный) формат данных, а также созданный на его основе программный пакет High Precision Digit-Parallel Solver (HPDP-Solver), в результате экспериментального исследования быстродействия которого получены весьма достойные результаты, относительно известных мировых аналогов.

1. Модулярно-позиционный формат для высокоточных разрядно-параллельных вычислений с плавающей точкой.

В двоичных форматах с плавающей точкой (здесь рассматриваются лишь форматы IEEE-754) любое вещественное число представляется трехэлементным набором:

$$[M, e, s] \quad (1)$$

где M принадлежит интервалу [0, 2) и обозначает рациональную мантиссу, e принадлежит интервалу $[e_{min}, e_{max}]$ и выражает порядок (экспоненту), $e_{min}=2-2^{w-1}$, $e_{max}=2^{w-1}-1$, $s=\{0, 1\}$ – знак числа.

Величина чисел, записанных в таком формате, выражается формулой -1^e*M*2^e. Машинными представлениями чисел вида (1) являются (w+t+1)-разрядные двоичные векторы $(s\, r_w \ldots r_2 r_1 d_t \ldots d_2 d_1)$, где

разряды с d_1 по d_t отводятся под представление рациональных двоичных мантисс $M=d_t . d_{t-1}\ldots d_2 d_1$, разряды с r_1 по r_w отводятся под представление целочисленных порядков e, записанных в форме с избытком (в смещенной форме) $E=r_w r_{w-1}\ldots r_2 r_1=e+e_{max}$, разряд s выражает знак числа.

Принимая во внимание условие, что под целую часть рациональной мантиссы $M=d_t . d_{t-1}\ldots d_2 d_1$ в двоичных форматах вида (1) отводится 1 бит d_t, определим *целочисленную мантиссу* $M'=d_t d_{t-1}\ldots d_2 d_1$ как t-разрядное неотрицательное целое двоичное число, такое что $M=M'*2^{1-t}$. Определим *перемещенный порядок* λ, как целое двоичное число со знаком, такое, что λ=e-t+1, , где *e* – *w*-разрядный порядок числа, представленного в двоичном формате (1).

Зададим *n* целочисленных положительных оснований системы остаточных классов (СОК) [2, 3] $p_1,p_2,\ldots,p_n$ таких, что для всех i, j=1,2,…,n, при i<>j: $\gcd(p_i,p_j)=1$, где $\gcd(p_i,p_j)$ – наибольший общий делитель для p_i и p_j.

Далее целочисленную мантиссу $M'=d_t d_{t-1}\ldots d_2 d_1$ преобразуем в СОК с заданными основаниями $p_1,p_2,\ldots,p_n$, получая тем самым *модулярную мантиссу* $M''=(m_1,m_2,\ldots,m_n)$:

$$M''=(m_1,m_2,\ldots,m_n)=(M' \bmod p_1, M' \bmod p_2,\ldots, M' \bmod p_n),$$

где m_i принадлежит $[0, p_i-1)$, i=1,2,…,n – цифры (модулярные разряды) модулярной мантиссы M'', $M' \bmod p_i$ – операция получения остатка от деления M' на i-ое основание p_i.

Другими словами, значение каждой знакопозиции m_i модулярной мантиссы M'' получается посредством вычисления выражения $m_i = M' \bmod p_i = M'-]M' / p_i[*p_i$, где $]M' / p_i[$ - наибольшее целое, не превышающее M' / p_i.

Все операции над знакопозициями m_i модулярных мантисс $M''=(m_1,m_2,\ldots,m_n)$ будем выполнять относительно выбранных оснований $p_1,p_2,\ldots,p_n$ согласно правилам выполнения арифметических операций в модулярной арифметике [2, 3]. Основания $p_1,p_2,\ldots,p_n$ являются общими для всех операндов и хранятся в общей памяти вычислительных устройств.

При этом с порядком λ продолжаем работать в двоичной системе и в СОК не преобразуем. Таким образом, любое число с плавающей точкой вида (1) можно представить в следующем *модулярно-позиционном (квазимодулярном) формате* (рисунок 1):

$$[(m_1,m_2,\ldots,m_n), \lambda, s] \qquad (2)$$

где $(m_1,m_2,\ldots,m_n)$ – набор разрядов-цифр модулярной мантиссы M'', λ – позиционный перемещенный порядок, представляющий собой целое двоичное число со знаком.

В формате (2) модулярные мантиссы $M''=(m_1,m_2,\ldots,m_n)$ в общем случае могут изменяться в пределах интервала [0, P-1], где $P=p_1*p_2*\ldots*p_n$ – произведение выбранных оснований $p_1,p_2,\ldots,p_n$.

Рис. 1 - Модулярно-позиционный формат с плавающей точкой

Рассмотрим получение модулярно-позиционных представлений чисел на примерах. Пусть числа представляются в 10-разрядном двоичном формате вида (1), в котором под порядок e, отводится четыре бита (максимальный порядок $e_{max}=24-1-1=7$, соответственно e=E-7), под дробную часть мантиссы – пять бит (т.е. t=6, причем целая часть d6 рациональной мантиссы M=d6 . d5…d2 d1 в явном виде не записана) и под знак числа – один бит. Пусть для представления мантисс в модулярно-позиционном формате $[(m1,m2,\ldots,m_n), \lambda, s]$ выбраны три основания: p1=3=22-1, p2=7=23-1, p3=31=25-1.

Пример 1. Перевести число $X=[1.5, -1, 0]=-1^0*1.5*2^{-1}$, представленное в двоичном формате [M, e, s], в модулярно-позиционный формат $[(m_1,m_2,\ldots,m_n), \lambda, s]$.

С учетом принятых характеристик двоичного формата [M, e, s], число *X* будет записано в памяти ЭВМ в виде двоичного вектора (0 0110 10000).

1. Выделяем составные части числа *X*: знак числа s=0, дробная часть рациональной мантиссы $d_5\ldots d_2 d_1=10000_2$, избыточный порядок $E=0110_2=6$.

2. Восстанавливаем целую часть d_6 мантиссы $M=d_6.d_5\ldots d_2d_1$: $d_6=1$, т.к. $E>0$, следовательно $M=1.10000_2$.

3. Определяем порядок e: $e=E-e_{max}=6-7=-1$, т.к. $E>0$.

4. Определяем перемещенный порядок λ и целочисленную мантиссу M':

$$\lambda=e-t+1=-1-6+1=-6,$$

$$M'=d_td_{t-1}\ldots d_2d_1=110000_2=48.$$

5. Находим модулярную мантиссу $M''=(m_1,m_2,m_3)$: $M''=(48 \bmod 3, 48 \bmod 7, 48 \bmod 31)=(0,6,17)$.

В результате получили число *X*, представленное в модулярно-позиционном формате (2):

$$X=[(0,6,17), -6, 0]=-1^0*(0,6,17)*2^{-6}.$$

Пример 2. Перевести число $X=[0.625, -6, 1]=-1^1*0.625*2^{-6}$, представленное в двоичном формате [M, e, s], в модулярно-позиционный формат $[(m_1,m_2,\ldots,m_n), \lambda, s]$.

С учетом принятых характеристик двоичного формата [M, e, s], число *X* будет записано в памяти ЭВМ в виде двоичного вектора (1 0000 10100).

1. Выделяем составные части числа *X*: знак числа s=1, дробная часть рациональной мантиссы $d_5\ldots d_2\,d_1=10100_2$, смещенный (избыточный) порядок $E=0000_2=0$.

2. Восстанавливаем целую часть d_6 мантиссы $M=d_6.d_5\ldots d_2d_1$: $d_6=0$, т.к. $E=0$, следовательно $M=0.10100_2$.

3. Определяем порядок e: $e=e_{min}=2-2^{4-1}=-6$, т.к. $E=0$.

4. Определяем перемещенный порядок λ и целочисленную мантиссу M':

$$\lambda=e-t+1=-6-6+1=-11,$$

$$M'=d_td_{t-1}\ldots d_2d_1=010100_2=20.$$

5. Находим модулярную мантиссу $M''=(m_1,m_2,m_3)$: $M''=(20 \bmod 3, 20 \bmod 7, 20 \bmod 31)=(2,6,20)$.

В результате получили число *X*, представленное в модулярно-позиционном формате (2):

$$X=[(2,6,20), -11, 1]=-1^1*(2,6,20)*2^{-11}.$$

Примеры показывают, что в результате выбора трех оснований $p_1=3=2^2-1$, $p_2=7=2^3-1$, $p_3=31=2^5-1$, разрядность которых не превышает пять бит, обеспечивается корректная работа с мантиссами, позиционная разрядность которых составляет десять бит (произведение оснований *P*=651), то есть точность представления увеличивается в два раза. При выборе большего числа оснований, диапазон допустимых значений модулярных мантисс может быть существенно расширен.

Обоснование новизны подхода

Необходимо отметить, что ранее, в работе [4], был представлен модулярный формат $[(a_1, a_2, \ldots,a_n), t]$, схожий с предлагаемым модулярно-позиционным форматом $[(m_1,m_2,\ldots,m_n), \lambda, s]$. Однако модулярный формат $[(a_1, a_2,\ldots,a_n), t]$ служит для представления чисел в формате с фиксированной, а не с плавающей точкой, и поэтому в нем модулярная величина $(a_1, a_2,\ldots,a_n)$ выражает разряды всего числа с учетом знака и порядка: знак числа $A=[(a_1, a_2,\ldots,a_n), t]$ определяется в соответствии с диапазоном, которому принадлежит модулярное число $(a_1, a_2,\ldots,a_n)$, а порядок t принадлежит интервалу $[0, k_f]$ и определяет позицию фиксированной точки в числе A и заложен в модулярной форме $(a_1, a_2,\ldots,a_n)$, причем значение каждой знакопозиции a_i в модулярном формате $[(a_1, a_2,\ldots,a_n), t]$ определяется выражением $a_i = (K/2^t) \bmod p_i$, где K – целое число такое, что $|K| \le 2^{nf+kf} - 1$, а n_f и k_f – длины соответственно целой и дробной частей числа в позиционном формате с фиксированной точкой.

Алгоритмы выполнения базовых арифметических операций, а также округления и определения знака, над числами, представленными в модулярном формате с фиксированной точкой $[(a_1, a_2,\ldots,a_n), t]$, принципиально отличны от алгоритмов выполнения операций над числами с плавающей точкой, для представления которых предлагается модулярно-позиционный формат $[(m_1,m_2,\ldots,m_n), \lambda, s]$.

Помимо модулярного формата $[(a_1, a_2,\ldots,a_n), t]$, в работе [4] также предлагается единый модулярно-позиционный формат (МПФ) $[(a_1, a_2,\ldots,a_n), t, m_f, p_f]$, где m_f – мантисса числа во вложенном позиционном формате с плавающей точкой (т.е. представляется, в свою очередь, в виде пары чисел – позиционной мантиссы и порядка), p_f – порядок числа в МПФ. Данный формат позволяет осуществлять обработку чисел как с фиксированной, так и с плавающей точкой. Однако подход к использованию модулярных систем счисления в формате $[(a_1, a_2,\ldots,a_n), t, m_f, p_f]$ остается неизменным – модулярное число $(a_1, a_2,\ldots,a_n)$ все также выражает число $A=[(a_1, a_2,\ldots,a_n), t]$, записанное с учетом знака и порядка в формате с фиксированной точкой, т.е. каждая

его знакопозиция вычисляется по формуле $a_i = (K/2^t) \bmod p_i$, а для работы с числами в форме с плавающей точкой используется позиционная мантисса m_f и порядок p_f.

Предлагаемый модулярно-позиционный формат с плавающей точкой $[(m_1,m_2,\ldots,m_n), \lambda, s]$ предполагает рассмотрение модулярной мантиссы $M''=(m_1,m_2,\ldots,m_n)$ отдельно от порядка λ и знака s, т.е. значение каждой знакопозиции m_i вне зависимости от λ и s определяется в соответствии с выражением $m_i = M' \bmod p_i$, где M' – полученная в результате эквивалентных преобразований положительная целочисленная позиционная мантисса (при вычислении каждого модулярного разряда m_i не предполагается дополнительное деление M' на 2^{kf}, где k_f – длина дробной части рациональной мантиссы $M=d_t . d_{t-1}\ldots d_2 d_1$), причем $M''=(m_1,m_2,\ldots,m_n)$ не несет в себе информацию обо всем числе $[(m_1,m_2,\ldots,m_n), \lambda, s]$, а лишь указывает на старшие значащие разряды в записи его модуля, без определения позиции разделяющей точки и знака, поэтому для определения абсолютной величины числа $[(m_1,m_2,\ldots,m_n), \lambda, s]$ необходимо умножить M'' на экспоненциальную форму 2^{λ} порядка λ. Знак s рассматривается как отдельный элемент в записи числа, т.е. не закодирован в модулярной мантиссе $M''=(m_1,m_2,\ldots,m_n)$.

Эти моменты определяют принципиально отличные от предложенных в работе [4] методы и алгоритмы записи чисел в модулярно-позиционном формате, алгоритмы выполнения базовых арифметических операций, преобразования, округления модулярных мантисс, определения знака, и, как следствие, принципиально иной способ организации вычислений.

Преимущества модулярно-позиционного формата

Представление чисел с плавающей точкой в модулярно-позиционном формате (2) позволяет решить сразу несколько проблем, присущих позиционным представлениям вида (1):

- во-первых, увеличивается точность вычислений, которая определяется разрядностью мантисс — для достижения точности 4*double достаточно выбрать восемь 32-битных оснований.
- во-вторых, ввиду взаимной независимости разрядов m_i, i=1,2,..,n, обеспечивается возможность распараллеливания арифметических операций на уровне отдельных разрядов;
- в-третьих, решается задача минимизации зависимости времени выполнения операций от точности, так как при добавлении дополнительных оснований p_i для увеличения точности, в случае, если их число превышает число вычислительных устройств, время выполнения операций увеличивается линейно, а не экспоненциально или квадратично, как в высокоточных позиционных библиотеках.

Область применения модулярно-позиционного формата

Преобразование чисел, представленных в позиционных форматах с плавающей точкой вида (1) в модулярно-позиционный формат (2) и обратно, из формата (2) в формат (1), связано с определенными временными затратами, поэтому класс задач, которые могут быть эффективно решены с использованием модулярно-позиционного формата ограничен, но вместе с тем он достаточно широк и включает в себя: множество задач, сводимых к выполнению итеративных операций над массивами данных (например, сложные матричные и матрично-векторные операции); задачи решения систем линейных алгебраических уравнений, в том числе и плохо обусловленных; задачи решения дифференциальных уравнений с применением явных разностных схем, проекционных методов, либо метода конечных элементов; ресурсоемкие задачи, решение которых составляет множество итераций, в результате чего накопление ошибок округления неизбежно приводит к потере основной доли значащих разрядов результата, а также других задач науки и техники, требующих работы с вещественными числами, которые либо лежат в непосредственной близости к числовому нулю, либо удалены от него в сторону больших и сверхбольших машинных диапазонов. Другими словами, использование модулярно-позиционного формата видится целесообразным, когда время, затрачиваемое на прямое и обратное преобразование мало, по сравнению со временем непосредственно расчетов, использование форматов с фиксированной точкой принципиально невозможно в силу необходимости оперирования вещественными числами, а машинные форматы не обеспечивают требуемой точности вычислений.

Рис. 2 - Примеры алгоритмов выполнения арифметических операций с плавающей точкой в модулярно-позиционном формате

Для модулярно-позиционного формата разработаны эффективные параллельные алгоритмы выполнения арифметических операций (сложение, вычитание и деление с ПТ), алгоритмы округления модулярных мантисс (используются итерационные алгоритмы округления модулярных мантисс M'' до четного с последующим делением их на двойку и одновременным увеличением порядков λ на единицу до; число итераций определяется типом операции, которая должна быть выполнена в следующий момент времени) и определения исключений, обеспечивающие регулярность и предсказуемость вычислительного процесса. В качестве примера, на рисунке 2 представлены схемы параллельных алгоритмов умножения и сложения модулярно-позиционных операндов. Для реализации операций деления и сравнения по величине модулярных мантисс, определения их выхода за допустимый диапазон представления и преобразования модулярных мантисс в позиционную систему, являющихся наиболее сложными операциями в СОК, предлагается использовать эффективные методы и алгоритмы, предложенные в работах [3, 5-7].

Введение новых параллельных вычислительных структур в обязательном порядке должно сопровождаться проектированием схем декомпозиции данных по узлам и ядрам вычислительных устройств. Поэтому была произведена разработка и анализ схем распределения модулярно-позиционных структур данных между узлами и вычислительными ядрами в пределах узлов. На рисунке 3 представлены упрощенные модели этих схем для систем с общей памятью. Для многопроцессорных кластерных систем данные схемы аналогичны.

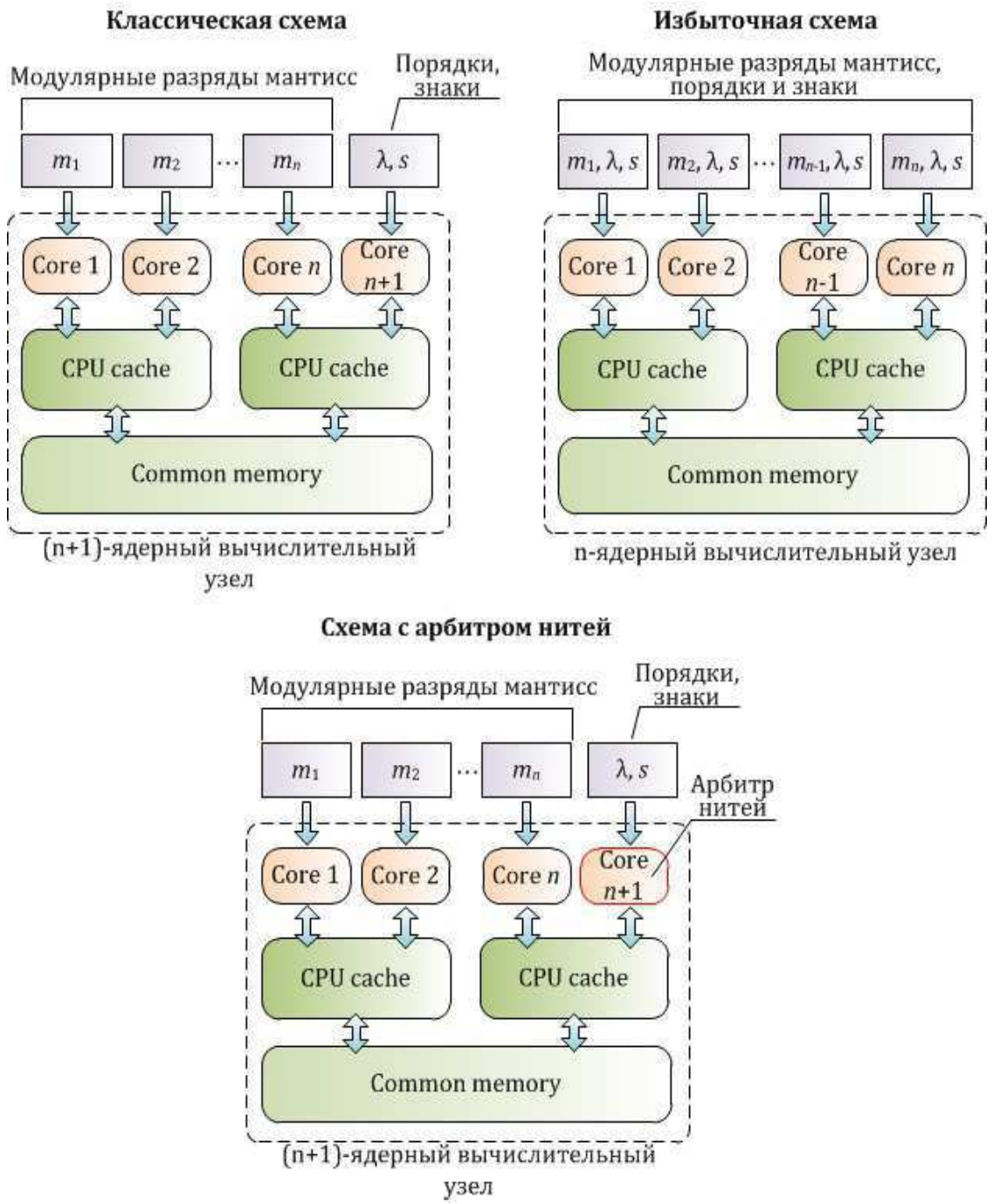

Рис. 3 Схемы распределения модулярно-позиционных структур данных для вычислительных систем с общей памятью

2. Программный пакет High Precision Digit-Parallel Solver.

На основе модулярно-позиционного формата и созданного для него алгоритмического аппарата высокоточных разрядно-параллельных вычислений разрабатывается трехзвенный программный пакет *High Precision Digit-Parallel Solver* (HPDP-Solver). Обобщенная структура пакета представлена на рисунке 4.

Серверная часть является «ядром» пакета и реализует выполнение всех заложенных в нее алгоритмических решений по обработке модулярно-позиционных структур данных. Сервер приложения состоит из трех частей: блок инициализации параметров, блок приведения данных к параллельным модулярно-позиционным структурам, вычислительный блок.

1. Блок инициализации параметров позволяет настроить программный пакет для работы на конкретной вычислительной системе, а также на решение конкретной задачи посредством задания доступных параметров вычислений. К таким параметрам относятся:

- число необходимых для использования вычислительных узлов;

- число вычислительных ядер в каждом узле;
- необходимость использования межпроцессорных коммуникаций;
- разрядность вычислительных устройств;
- число параллельных процессов и нитей для решения задачи;
- требуемая точность вычислений (задается посредством указания разрядности мантисс операндов);
- схема распределения модулярно-позиционных структур данных.

В блоке инициализации параметров заложены правила выбора оптимальной для конкретной задачи и вычислительной системы схемы распределения модулярно-позиционных структур (рисунок 3).

Рис. 4 Обобщенная структура программного пакета HPDP-Solver

2. Блок приведения данных к параллельным модулярно-позиционным структурам осуществляет взаимодействие с хранилищем, загружая исходные позиционные данные с плавающей точкой. Далее загруженные данные преобразуются к модулярно-позиционному формату и становятся доступными для вычислительного блока. В основе функционирования блока приведения данных к модулярно-позиционным структурам заложены полностью параллельные масштабируемые алгоритмы, которые разрабатывались с учетом особенностей типов данных IEEE-754 [7].

3. Вычислительный блок реализует выполнение разрядно-параллельных численных расчетов в модулярно-позиционном формате. В вычислительном блоке определяется полезный для конечного пользователя функционал пакета HPDP-Solver. К функциям, реализуемым вычислительным блоком, относятся:

a) Элементарные высокоточные разрядно-параллельные операции с ПТ:
- высокоточное умножение;
- высокоточное сложение;
- деление с заданной точностью;
- возведение в натуральную степень;
- вычисление логарифмов;
- вычисление прямых и производных тригонометрических функций.

b) Высокоточные операции над массивами данных (в настоящий момент ведется их разработка):
- вычисление скалярного произведения векторов;
- умножение матриц / векторов;

- сложение матриц / векторов;
- возведение матриц в степень;
- умножение матрицы / вектора на константу;
- поиск определителя и ранга матрицы;
- обращение матриц (предполагается использование итерационного алгоритма Гаусса с обратным ходом).

В вычислительном блоке на программном уровне реализуются схемы распределения модулярно-позиционных структур данных, алгоритмы межпроцессорных обменов, а также базовые разрядно-параллельные алгоритмы выполнения высокоточных арифметических операций с плавающей точкой. *Основная концепция*, заложенная в вычислительном блоке – максимально-возможное отделение алгоритмического параллелизма задачи от арифметического параллелизма модулярно-позиционных структур, благодаря использованию гибридных технологий параллельной обработки (OpenMP+MPI).

В настоящее время производится разработка, отладка и тестирование основных функций для работы с массивами данных. Для каждой функции проектируется и исследуется специфическое алгоритмическое решение, отражающее природу модулярно-позиционных структур, нацеленное на параллельное исполнение и максимально возможную масштабируемость вычислительного процесса. Кроме этого, для каждой функции рассматриваются возможности ее программной реализации в различных спецификациях: MPI+OpenMP, MPI, OpenMP и CUDA.

3. Результаты экспериментов.

Для оценки эффективности применения модулярно-позиционного формата при высокоточном решении численных задач над массивами данных с плавающей точкой, был проведен крупный эксперимент, направленный на исследование быстродействия серверной части пакета HPDP-Solver в сравнении с широко известной во всем мире программной библиотекой The GNU MP Bignum Library (GMP [8]). *Целью эксперимента* являлось установление зависимости времени решения задачи от точности (т.е. от разрядности мантисс операндов), при удалении в область больших и сверхбольших машинных диапазонов, и от числа используемых вычислительных ядер.

В качестве алгоритмической базы для эксперимента выступали высокоточные параллельные алгоритмы умножения плотных матриц $C = A*B$, $A, B, C \in R^{nxn}$ и скалярного произведения векторов $c = a \times b$, $a, b, c \in R^{n}$. Данные операции, как база для исследований, выбраны не случайно: во-первых, они предполагает выполнение операций умножения с плавающей точкой, что приводит к увеличению разрядности мантисс, во-вторых, для получения каждого элемента результатного массива (либо результата скалярного произведения) необходимо сложить полученные частичные произведения, что позволяет оценить также скорость сложения и выравнивания порядков с плавающей точкой. Таким образом, оцениваются сразу три основные операции: умножение, сложение и выравнивание порядков.

При проведении численных экспериментов задействовалось от 8 до 32 вычислительных ядер кластерной системы Enigma (HP Hewlett-Packard Cluster Platform 3000 BL460c, Intel EM64T Xeon 5345, 2,3 ГГц, Infiniband) Вятского государственного университета. В качестве элементов матриц использовались положительные числа с плавающей точкой, имеющие разрядность мантиссы от 31 до 1891 бит и разрядность порядка 32 бита, т.е. изменяющиеся в пределах диапазона $[0, 2^{1891}*(2^{31}-1)]$. В качестве элементов векторов при вычислении скалярного произведения использовались положительные числа с плавающей точкой, имеющие разрядность мантиссы от 31 до 961 бит и разрядность порядка 32 бита.

Программный пакет HPDP-Solver выполнял умножение матриц согласно избыточной схеме распределения модулярно-позиционных структур, представленной на рисунке 3 (т.е. потенциальный алгоритмический параллелизм задачи, реализуемый посредством разбиения одного из массивов на полосы либо блоки, не использовался). Достижение требуемой точности вычислений обеспечивалось путем увеличения числа оснований $p_1,p_2,\ldots,p_n$, используемых для представления модулярных мантисс $M''=(m_1,m_2,\ldots,m_n)$, из расчета, что каждое основание, по величине не превосходящее 2^{31}, позволяет повысить точность на 31 бит. В программе, построенной на основе библиотеки GMP, была реализована классическая схема горизонтального ленточного разбиения.

На рисунке 5 представлен график зависимости времени умножения матриц 1500x1500 от точности счета при распараллеливании вычислений на восемь ядер.

Согласно данному графику, с увеличением разрядности элементов время счета при использовании библиотеки GMP возрастает существенным образом: при разрядности мантисс 31 бит, задача умножения решается за 431 секунду, а при разрядности 1891 бит – за 26361 секунд (т.е. более, чем за 7 часов). В данном эксперименте можно заметить интересный момент: с увеличением точности вычислений в 61 раз, время решения задачи возросло также в 61 раз. При использовании пакета HPDP-Solver зависимость времени счета от точности гораздо менее существенна: при разрядности мантисс 31 бит, умножение матриц выполнилось за 344 секунды, а при разрядности 1891 бит – за 2702 секунды, т.е. с увеличением точности вычислений в 61 раз, решение задачи замедлилось лишь в 7,8 раз.

Таким образом, при работе с 1891-битными мантиссами пакет HPDP-Solver решил задачу в 9,7 раз быстрее, чем библиотека GMP, что, учитывая мировую известность проекта The GNU MP Bignum Library [8], непременно, является достойным результатом.

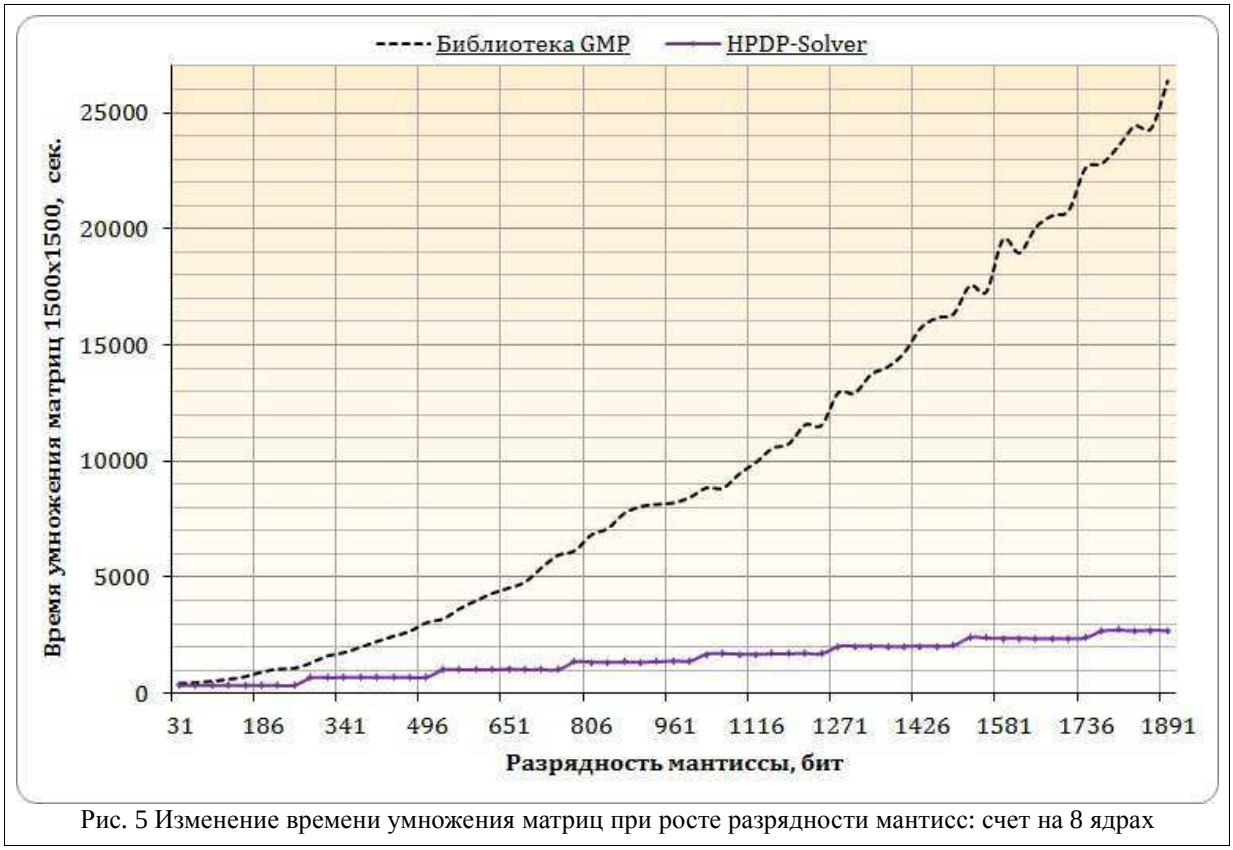

Рис. 5 Изменение времени умножения матриц при росте разрядности мантисс: счет на 8 ядрах

На рисунках 6 и 7 представлены аналогичные графики, при счете соответственно на 16 и 32 ядрах. Полученные результаты в полной мере соответствуют определенным ранее теоретическим оценкам. Так, использование 16 вычислительных ядер позволило вдвое сократить время счета (относительно восьми ядер) как при использовании библиотеки GMP, так и пакета HPDP-Solver, на всем промежутке изменения разрядности мантисс обрабатываемых чисел. Аналогично, при счете на 32 ядрах время решения задачи сокращается приблизительно в 4 раза.

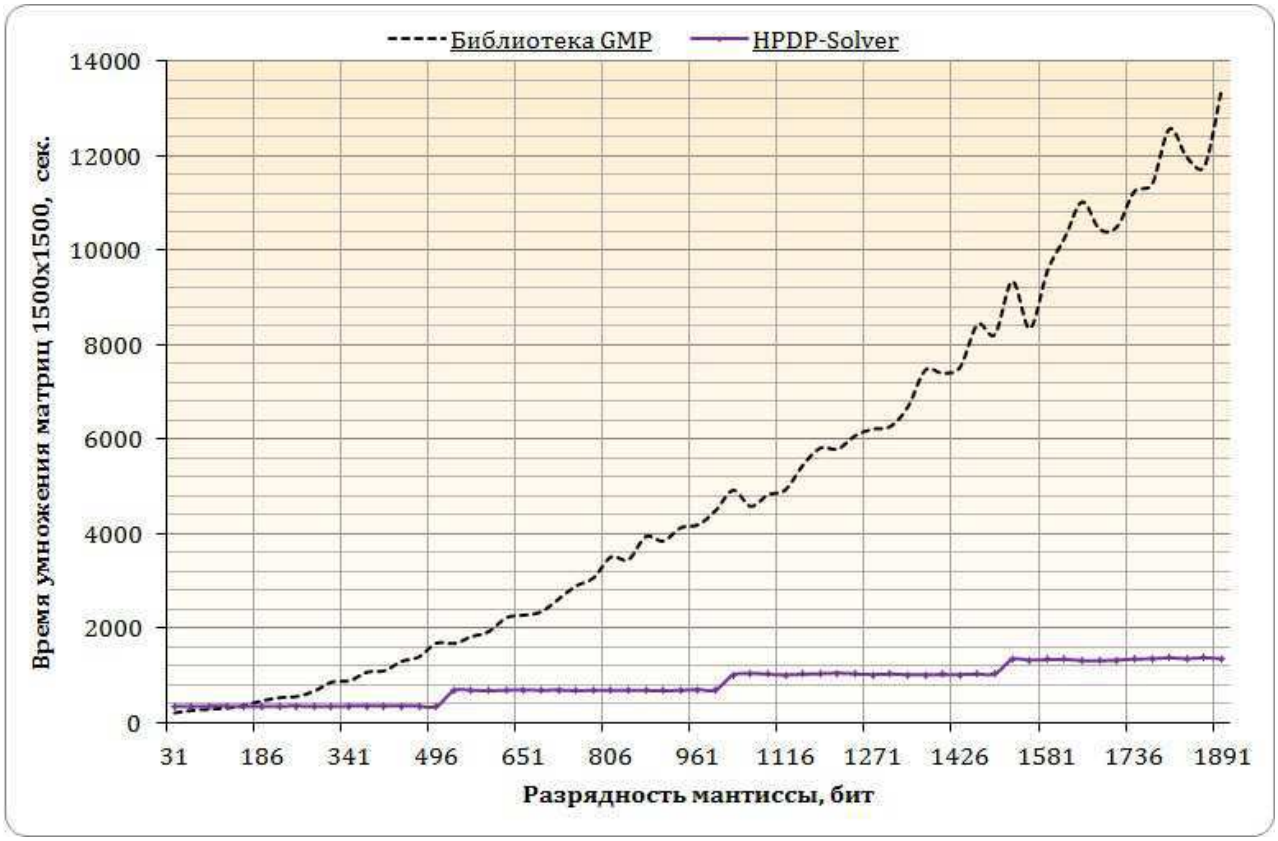

Рис. 6 Изменение времени умножения матриц при росте разрядности мантисс: счет на 16 ядрах

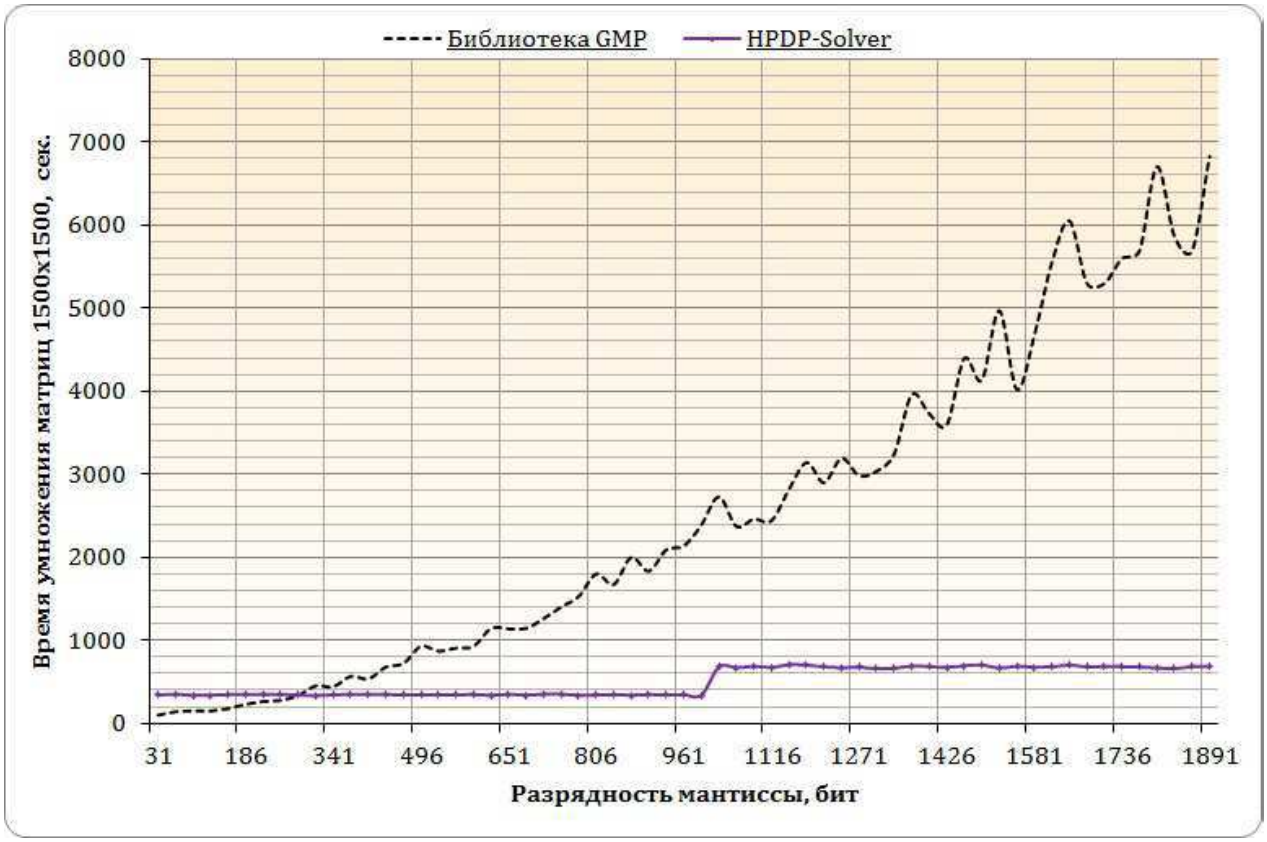

Рис. 7 Изменение времени умножения матриц при росте разрядности мантисс: счет на 32 ядрах

Следует заметить, однако, что добиться таких высоких показателей масштабируемости при использовании библиотеки GMP стало возможным лишь благодаря высокому алгоритмическому параллелизму самой задачи умножения матриц (арифметические операции в GMP не распараллеливались), отсюда следует, что при решении высокоточных плохо распараллеливаемых на алгоритмическом уровне задач добиться линейного ускорения с использованием данной библиотеки будет невозможно.

На рисунке 8 представлен сводный график зависимостей быстродействия программного пакета HPDP-Solver от точности вычислений при различном числе используемых параллельно работающих ядер. Анализ графика позволяет сделать вывод, что эффективность и масштабируемость алгоритмов, заложенных в пакете HPDP-Solver, обуславливается лишь числом оснований для представления модулярно-позиционных структур и принципиально не зависит от решаемой задачи. Так, при счете на восьми ядрах время счета удваивается каждый раз, когда разрядность модулярных мантисс увеличивается на 248 бит: для корректной обработки модулярных мантисс $M''=(m_1,m_2,\ldots,m_n)$, позиционная разрядность которых больше, чем 248 бит, необходимо использовать больше восьми 31-битных оснований $p_1,p_2,\ldots,p_n$, соответственно, минимум одно вычислительное ядро будет обрабатывать данные более чем по одному основанию p_i. Аналогично, для корректной обработки модулярных мантисс, имеющих разрядность более чем 526 бит, необходимо использовать более шестнадцати 31-битных оснований, и минимум одно вычислительное ядро будет обрабатывать данные более чем по двум основаниям, и т.д. В свою очередь, при счете на 32 ядрах, для того, чтобы минимум одно вычислительное ядро обрабатывало данные более чем по одному модулярному разряду m_i, необходимо, чтобы позиционная разрядность модулярных мантисс была больше, чем 992 бита и т. д. Следовательно, вне зависимости от потенциального параллелизма задачи, пусть даже алгоритм ее решения будет строго последователен, характер зависимостей, представленных на рисунке 8, останется неизменным.

На рисунке 9 представлен график зависимости времени выполнения операции скалярного произведения векторов, содержащих 1000000 элементов от точности при счете на восьми ядрах. При решении данной задачи параллельно вычислялись лишь частичные произведения, а их финальное суммирование было реализовано в последовательном режиме. Это объясняет рост времени счета от точности как в пакете HPDP-Solver, так и с использованием библиотеки GMP. Вместе с тем, характер зависимостей остается неизменным, и с увеличением точности счета наблюдается существенный отрыв по быстродействию пакета HPDP-Solver, относительно программного решения, построенного на основе GMP.

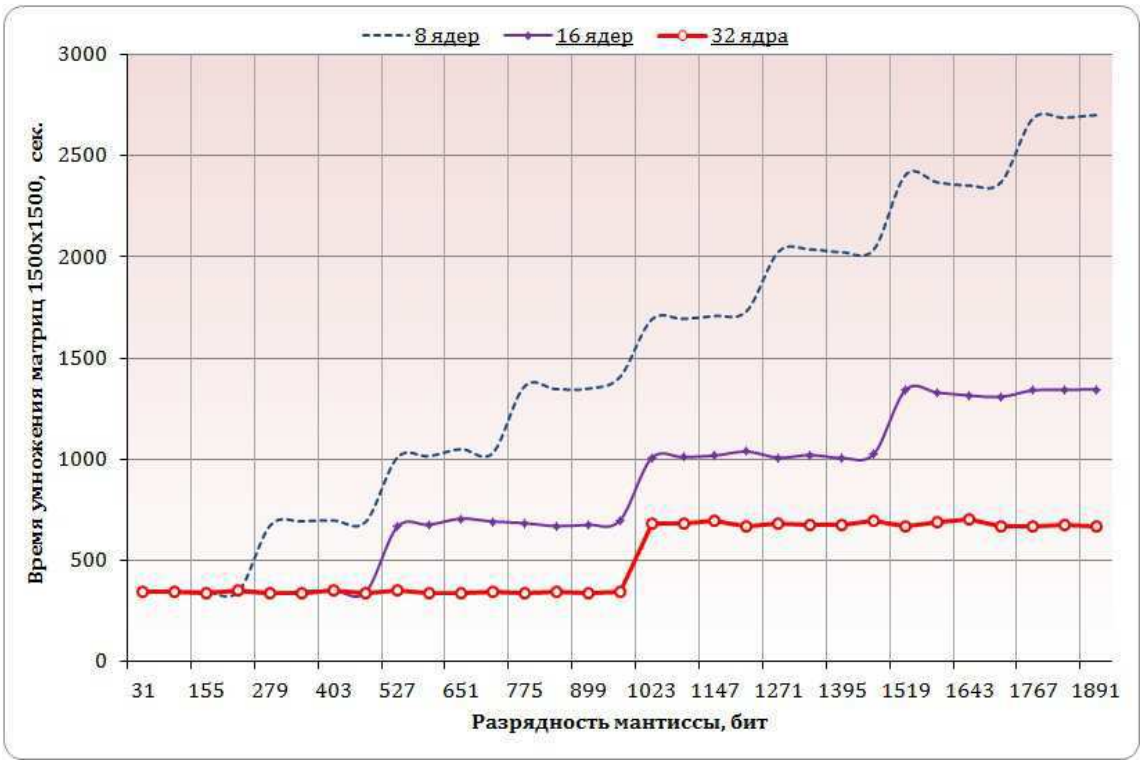

Рис. 8 Сводный график зависимостей быстродействия программного пакета HPDP-Solver от точности при различном числе параллельно работающих вычислительных ядер при решении задачи матричного умножения

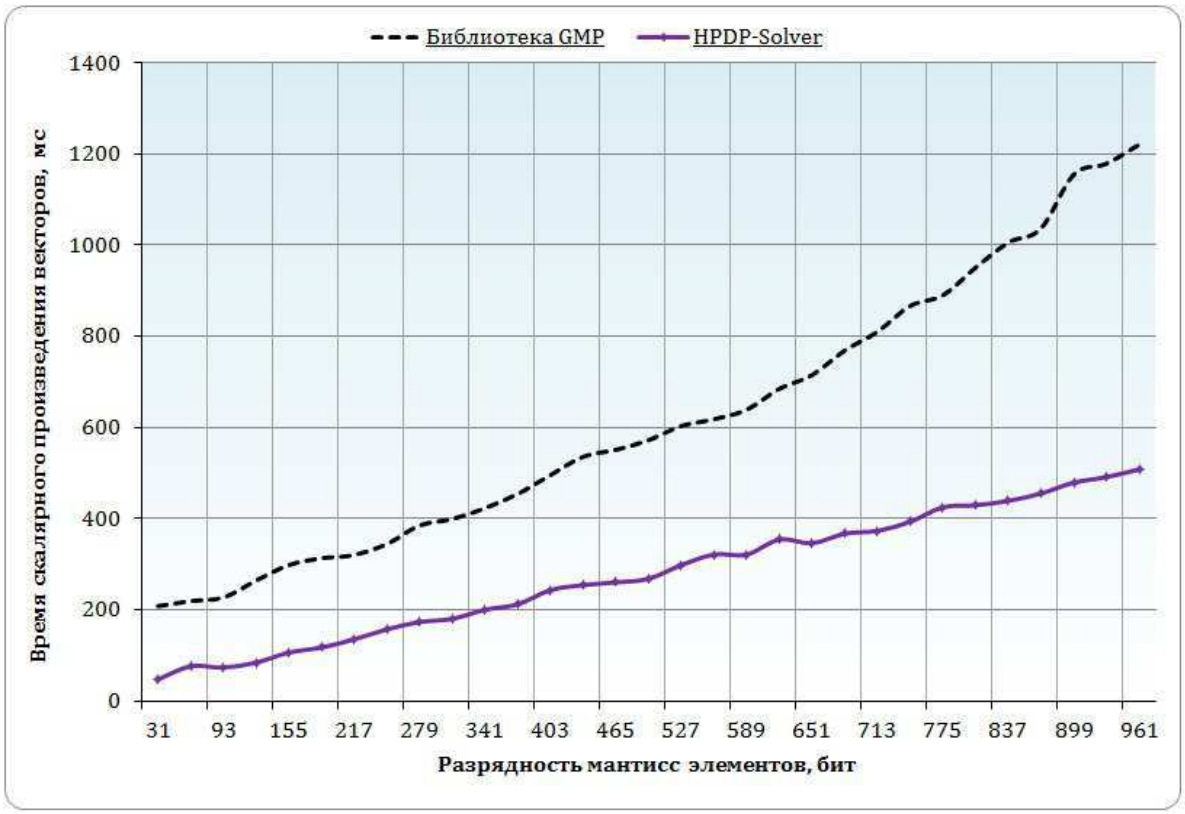

Рис. 9 Зависимость времени скалярного произведения векторов от точности при счете на 8 ядрах

Заключение.

Предложен, обоснован и реализован новый способ организации высокоточных вычислений с плавающей точкой – модулярно-позиционный формат, особенностью которого является использование для представления мантисс чисел многомодульной системы остаточных классов, а для представления порядков – позиционной системы счисления, при этом величина числа выражается формулой $-1^{s}*M''*2^{\lambda}$, где $M''=(m_1,m_2,\ldots,m_n)$ – модулярная мантисса, λ – позиционный порядок, а s – знак числа. Это позволяет расширить диапазон представления операндов, повысить точность вычислений и распараллелить выполнение основных арифметических операций, сохраняя при этом скорость обработки порядков. Кроме этого, модулярно-позиционные структуры данных гораздо более компактны по сравнению со структурами, основанными на длинной позиционной арифметике, что позволяет значительно экономить память.

Для модулярно-позиционного формата разработаны эффективные параллельные алгоритмические решения, которые легли в основу серверной части разрабатываемого программного пакета HPDP-Solver. Относительная независимость основных частей пакета – клиента, сервера и хранилища данных, позволит применять его в самых различных областях науки и техники, где необходимо выполнять массивные высокоточные вычисления.

Проведенные экспериментальные исследования быстродействия пакета HPDP-Solver показали его преимущества перед имеющей мировую известность позиционной библиотекой GMP, позволив решить задачу высокоточного умножения матриц в 9,7 раз быстрее.

На данном этапе разработки (выполняется проектирование и реализация функций для работы с массивами данных) программный продукт HPDP-Solver удовлетворяет всем поставленным задачам:

- рассчитан на работу со стандартными (IEEE-754) форматами исходных данных, но реализует значительно более высокую точность представления с плавающей точкой;
- позволяет минимизировать зависимость времени выполнения операций от точности;
- обеспечивает эффективное разрядно-параллельное выполнение арифметических операций;
- имеет возможности адаптации как под конкретную задачу (посредством задания необходимой схемы распределения модулярно-позиционных структур данных), так и под конкретную высокопроизводительную вычислительную систему (посредством задания разрядности оснований для представления модулярных мантисс и их числа).

Пакет HPDP-Solver может быть применен при решении крупных задач, которые предъявляют особо высокие требования к точности решения и сводятся к выполнению массивных операций сложения, вычитания

или умножения с плавающей точкой, когда время, затрачиваемое на прямое и обратное преобразование, существенно меньше времени непосредственно расчетов. Под эти условия попадает множество задач из самых различных областей науки и техники. В частности, к ним можно отнести задачи, решение которых основано на применении явных разностных схем, задачи матричного умножения, сложения, возведения в степень, решения плохо обусловленных СЛАУ и т.д., а также другие задачи, требующие выполнения операций с вещественными числами, которые либо лежат в непосредственной близости к числовому нулю, либо удалены от него в сторону больших машинных диапазонов.

ЛИТЕРАТУРА:

1. К.С. Исупов, А.Г. Иванов. Исследование эффективности современных средств поддержки высокоточных вычислений с вещественными числами // «Общество, наука, инновации (НТК-2012)»: Сб. материалов. Секция «Вычислительные системы и программное обеспечение для обработки данных и знаний». Статья № 4. Киров: ВятГУ, 2012. 11 с.
2. И.Я. Акушский, Д.И. Юдицкий. Машинная арифметика в остаточных классах [Текст]. – М.: Сов. Радио, 1968. – 440 с.
3. A. Omondi, B. Premkumar Residue Number Systems: Theory and Implementation (Advances in Computer Science and Engineering Texts) [Text]. - London : Imperial College Press, 2007. - 312 pages.
4. Ш.А. Оцоков. Структурно-алгоритмические методы организации высокоточных вычислений на основе теоретических обобщений в модулярной системе счисления: 05.13.05 - Элементы и устройства вычислительной техники и систем управления; 05.13.15 - Вычислительные машины, комплексы и компьютерные сети : дис…. доктора технических наук. – М., 2010. – 287 с.
5. A. Sabbagh, K. Navi. New Arithmetic Residue to Binary Converters // IJCSES International Journal of Computer Sciences and Engineering Systems. 2007. VOL.1, NO.4. pp. 295–299.
6. C. Chang, Y. Lai. A division algorithm for residue numbers // Applied Mathematics and Computation, 2006. NO.172(1). pp. 368-378.
7. G. Dimauro, S. Impedovo, G. Pirlo. A new technique for fast number comparison in the residue number system // IEEE transactions on computers. 1993. VOL.42, NO.5. pp. 608–612.
8. IEEE Standard for Floating-Point Arithmetic [Text] / IEEE. – 3 Park Avenue New York, NY 10016-5997, USA, 29 August 2008. – 70 pages
9. The GNU Multiple Precision Arithmetic Library [Electronic resource]. – Режим доступа: http://gmplib.org/. – Загл. с экрана. – 05.06.2011.