УДК 004.222

МОДУЛЯРНО-ПОЗИЦИОННЫЙ ФОРМАТ И ПРОГРАММНЫЙ ПАКЕТ ДЛЯ РАЗРЯДНО-ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ВЫСОКОЙ ТОЧНОСТИ В ФОРМАТЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ

К.С. Исупов

Рассматривается новый способ организации высокоточных вычислений с плавающей точкой, позволяющий распараллеливать арифметические операции вплоть до уровня отдельных цифр многоразрядных мантисс путем использования модулярно-позиционного формата представления данных. Основная концепция данного формата заключается в представлении мантисс чисел в многомодульной системе остаточных классов (СОК), а порядков – в позиционной системе счисления. Мантиссы сопровождаются позиционной характеристикой, которая способствует реализации эффективных алгоритмов выполнения немодульных операций в СОК, таких как деление (частный случай) и округление. На основе данного подхода разрабатывается программное решение High Precision Digit-Parallel Solver (HPDP-Solver). Комплекс HPDP-Solver может быть гибко настроен на конфигурацию конкретной машины, в результате чего обеспечивается наиболее эффективное использование ее ресурсов. В результате экспериментального исследования быстродействия пакета HPDP-Solver были получены результаты, доказывающие его преимущества при решении высокоточных численных задач перед имеющей мировую известность позиционной библиотекой GNU Multiple Precision Arithmetic Library. Пакет HPDP-Solver может быть применен при решении задач, которые предъявляют особо высокие требования к вычислительной точности.

Ключевые слова: плавающая точка, система остаточных классов, модулярно-позиционный формат, параллельная арифметика, высокоточные вычисления.

Введение

Проблема обеспечения достаточной точности вычислений всегда была актуальным направлением теоретических исследований в области компьютерной математики. Однако сейчас, как никогда ранее, несмотря на всю свою фундаментальную значимость, проблема точности приобретает все более четко выраженный прикладной аспект, проявляющийся, в первую очередь, при решении инженерных и научных задач большой размерности и высокой временной сложности, когда исследуемые процессы формализуются в виде дифференциальных уравнений, которые затем интегрируются численно, либо когда требуется обработка больших массивов исходных данных. Время решения таких задач даже при использовании суперкомпьютеров может составлять несколько часов, дней, месяцев. При таких «растянутых во времени» расчетах важна уверенность в получении корректных результатов, пригодных для практического применения. Причем, если алгоритмическую корректность решения можно верифицировать еще до начала вычислений, то задача определения значений погрешностей, возникающих на каждой итерации расчетов из-за недостаточной точности, является весьма нетривиальной проблемой.

Большинство численных методов оперируют над полем вещественных чисел, в силу этого актуальна проблема создания новых способов их компьютерной аппроксимации и организации параллельной обработки на многоядерных процессорах общего назначения, так как современные вычислительные системы реализуются в большинстве своем именно

на процессорах этого класса. В рамках данной проблемы приоритетными являются исследования, направленные на разработку новой математической базы, методов и масштабируемых алгоритмов высокоточных вычислений с плавающей точкой (ПТ), которые:

1) обеспечивают отображение всех величин, поддерживаемых используемыми в современных процессорах форматами с ПТ (наибольшее распространение получили двоичные форматы стандарта IEEE-754, включающие кроме чисел так же бесконечности и нечисловые величины – NaN), но лишены недостатков последних, связанных с ограничением 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 процессоров;

2) позволяют выполнять эффективное распараллеливание вычислений вплоть до уровня параллельной обработки отдельных разрядов мантисс операндов, так как не всегда алгоритмический параллелизм задачи позволяет при ее решении эффективно использовать все ресурсы многопроцессорной вычислительной системы;

3) обеспечивают минимизацию зависимости скорости выполнения арифметических операций от вычислительной точности этих операций.

В работе предлагается новый подход к организации высокоточных параллельных вычислений с плавающей точкой – модулярно-позиционный (квазимодулярный) формат данных и созданный на его основе программный пакет High Precision Digit-Parallel Solver (HPDP-Solver). Статья организована следующим образом. В разделе 1 дается формальное определение модулярно-позиционного формата, обосновывается его научная новизна, рассматриваются преимущества и области применения. Раздел 2 посвящен описанию основных модулей программного пакета HPDP-Solver. В разделе 3 приводятся результаты экспериментального исследования быстродействия пакета HPDP-Solver в сравнении с позиционной библиотекой высокой точности The GNU MP Bignum Library.

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

1.1. Формальное определение модулярно-позиционного формата

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

$$[M,\ e,\ s], \tag{1}$$

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 $(s\ r_w...r_2r_1d_t...d_2d_1)$, где разряды с d_1 по d_t отводятся под представление рациональных двоичных мантисс $M = d_t\,.\,d_{t-1}...d_2d_1$, разряды с r_1 по r_w отводятся под представление целочисленных порядков e, записанных в форме с избытком (в смещенной форме) $E = r_wr_{w-1}...r_2r_1 = e+e_{\max}$, разряд s выражает знак числа.

Принимая во внимание условие, что под целую часть рациональной мантиссы $M = d_t\,.\,d_{t-1}...d_2d_1$ в двоичных форматах вида (1) отводится 1 бит d_t, определим *целочисленную мантиссу* $M' = d_td_{t-1}...d_2d_1$ 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 $\lambda = 1–t+e$, , где e – w-разрядный порядок числа, представленного в двоичном формате (1).

Далее из множества $\{2^f+1, 2^f–1\}$ выберем n целочисленных положительных оснований (модулей) системы остаточных классов (СОК) [2, 3] $p_1,p_2,...,p_n$ таких, что для всех $i, j = 1,2,...,n$, при $i \neq j$: $\gcd(p_i, p_j) = 1$, где $\gcd(p_i, p_j)$ – наибольший общий делитель для p_i и p_j.

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

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

где m_i принадлежит $[0,\ p_i–1]$ – i-ая знакопозиция (модулярный разряд) модулярной мантиссы M'', представляющая собой наименьший неотрицательный остаток от деления M' на i-ый модуль p_i.

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

При всем 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* (рис. 1):

$$[(m_1,m_2,...,m_n),\ \eta(M''),\ \lambda,\ s] \qquad (2)$$

где $M'' = (m_1,m_2,...,m_n)$ – модулярная мантисса числа, $\eta(M'')$ – функция, от модулярной мантиссы, значение которой отражает результат сравнения величины M'' и $(P–1)/2$, где $P = p_1*p_2*...*p_n$, $\lambda = 1–t+e$ – позиционный порядок, представляющий собой целое двоичное число со знаком, s – знак числа в модулярно-позиционном формате.

Значение функции $\eta(M'')$ определяется исходя из условий:

$$\eta(M'') = 1, \text{ если } M'' \leq (P–1)\ /\ 2,$$
$$\eta(M'') = –1, \text{ если } M'' > (P–1)\ /\ 2,$$
$$\eta(M'') = 0, \text{ если превосходство } M'' \text{ над } (P–1)\ /\ 2 \text{ не определено.}$$

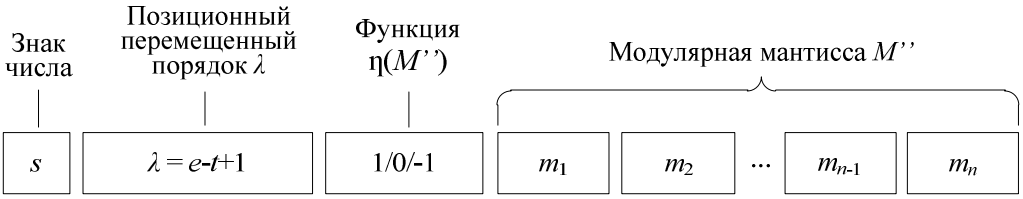

Рис. 1. Модулярно-позиционный формат с плавающей точкой

Сопровождение мантиссы каждого числа в модулярно-позиционном формате функцией $\eta(M'')$ позволит организовать эффективное выполнение алгоритмов работы со знаками, деления на двойку, минимизации числовой избыточности и округления модулярных мантисс. Все эти операции играют важную роль в организации вычислительного процесса над числами с плавающей точкой, и вместе с тем в системе остаточных классов реализуются весьма сложно. Для быстрого определения значения функции $\eta(M'')$ при известной модулярной мантиссе $M'' = (m_1,m_2,...,m_n)$ разработан специальный табличный алгоритм сравнения в СОК. Во время преобразования двоичной мантиссы M' к модулярной M'', определение значения функции $\eta(M'')$ будем выполнять на основании сопоставления M'

и (P–1) / 2, т.к. выполнять сравнение по величине позиционных чисел гораздо проще и быстрее, чем модулярных.

В формате (2) модулярные мантиссы $M'' = (m_1,m_2,...,m_n)$ в общем случае могут изменяться в пределах интервала [0, P-1], где $P = p_1*p_2*...*p_n$, поэтому становится возможным управлять точностью вычислений, задавая число и величину модулей p_i (как известно, точность в вычислениях с плавающей точкой определяется разрядностью мантисс операндов). Знакопозиции $m_1,m_2,...,m_n$ являются независимыми, что позволяет, при наличии многоядерных вычислителей, обрабатывать их параллельно и тем самым реализуется отдельный вид параллелизма - параллелизм на уровне арифметических операций.

Пример. 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 $e_{\max} = 2^{4\text{-}1}–1$, соответственно $e = E–7$), под дробную часть мантиссы – пять бит (т.е. $t = 6$, причем целая часть d_6 рациональной мантиссы $M = d_6 . d_5 \ldots d_2 d_1$ в явном виде не записана) и под знак числа – один бит. Пусть для представления мантисс в модулярно-позиционном формате $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$ выбраны три попарно взаимно простых модуля: $p_1 = 2^2–1$, $p_2 = 2^3–1$, $p_3 = 2^5–1$. Требуется перевести число $X = [1.5, –1, 0] = –1^0*1.5*2^{-1}$ из двоичного формата (1) в модулярно-позиционный формат (2).

С учетом принятых характеристик двоичного формата, 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 $d_5 \ldots d_2 d_1 = 10000_2$, порядок в форме с избытком $E = 0110_2 = 6$.

2. Восстанавливаем целую часть от M: $d_6 = 1$, т.к. $E > 0$, поэтому $M = 1.10000_2$.

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

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

$$\lambda = 1–t+e = –1–6+1 = –6,$$

$$M' = d_6 d_5 \ldots d_2 d_1 = 110000_2 = 48.$$

5. $M' <$ (P-1) / 2, следовательно $\eta(M'') = 1$.

6. 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):

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

Рассмотренный пример показывает, что в результате выбора трех оснований $p_1 = 2^2–1$, $p_2 = 2^3–1$, $p_3 = 2^5–1$, разрядность которых не превышает пяти бит, обеспечивается корректная работа с мантиссами, позиционная разрядность которых составляет десять бит ($P = p_1*p_2*p_3 = 651$), то есть точность аппроксимации увеличивается в два раза. При выборе большего числа оснований, диапазон допустимых значений модулярных мантисс может быть существенно расширен.

1.2. Обоснование новизны подхода

Необходимо отметить, что ранее, в работе [4], был представлен модулярный формат $[(a_1,a_2, ...,a_n), t]$, схожий с предлагаемым модулярно-позиционным форматом

$[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$. Однако модулярный формат $[(a_1,a_2, ...,a_n), t]$ служит для представления чисел с фиксированной, а не с плавающей точкой, и поэтому в нем модулярная величина $(a_1,a_2, ...,a_n)$ выражает разряды всего числа с учетом знака и порядка: знак числа $A = [(a_1,a_2, ...,a_n), t]$ определяется в соответствии с диапазоном, которому принадлежит модулярная мантисса $(a_1,a_2, ...,a_n)$, а порядок t принадлежит интервалу $[0, k_f]$, определяет позицию фиксированной точки в числе A и заложен в мантиссе $(a_1,a_2, ...,a_n)$, причем значение каждой знакопозиции a_i в формате $[(a_1,a_2, ...,a_n), t]$ определяется выражением $a_i = (K \,/\, 2^t) \bmod p_i$, где K – целое число такое, что $|K| \le 2^{n_f+k_f}-1$, а n_f и k_f – длины соответственно целой и дробной частей числа в позиционном формате с фиксированной точкой. Алгоритмы выполнения арифметических операций, а также округления и определения знака в модулярном формате с фиксированной точкой $[(a_1,a_2, ...,a_n), t]$, принципиально отличны от алгоритмов соответствующих операций над числами с плавающей точкой, для представления которых предлагается модулярно-позиционный формат $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$.

Более того, модулярный формат $[(a_1,a_2, ...,a_n), t]$ не предусматривает сопровождение мантисс $(a_1,a_2, ...,a_n)$ их позиционными характеристиками для ускорения немодульных операций в СОК, такими, как функция $\eta(M'')$ в модулярно-позиционном формате.

Помимо модулярного формата, автор также предлагает единый модулярно-позиционный формат (МПФ) $[(a_1,a_2, ...,a_n), t, m_f, p_f]$, где m_f – мантисса числа во вложенном позиционном формате с плавающей точкой, p_f – порядок числа. Данный формат позволяет осуществлять обработку чисел так же и с плавающей точкой. Тем не менее, подход к использованию модулярных систем счисления в МПФ остается неизменным – в наборе $(a_1,a_2, ...,a_n)$ все также закодирован порядок и знак числа $A = [(a_1,a_2, ...,a_n), t]$ в формате с фиксированной точкой, а для работы с плавающей точкой используется позиционная мантисса m_f и порядок p_f.

Предлагаемый модулярно-позиционный формат с плавающей точкой предписывает рассмотрение модулярной мантиссы $M'' = (m_1,m_2,...,m_n)$ отдельно от порядка λ и знака s: значение каждого разряда m_i, вне зависимости от λ и s, определяется выражением $m_i = M' \bmod p_i$, где M' – целочисленная двоичная мантисса числа. Таким образом, при вычислении значения m_i не предполагается дополнительное деление M' на 2^{kf}, если принимать за k_f длину дробной части исходной мантиссы. Поэтому $M'' = (m_1,m_2,...,m_n)$ не несет в себе информацию обо всем модулярно-позиционном числе, а лишь указывает на старшие значащие разряды в записи его модуля, без определения позиции разделяющей точки и знака. Для определения абсолютной величины числа $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$ необходимо умножить M'' на 2^λ. Знак s – отдельный элемент в записи модулярно-позиционного числа.

Эти моменты определяют отличные от предложенных в работе [4] методы и алгоритмы записи чисел в модулярно-позиционном формате (формат $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$ позволяет отображать все величины, в том числе и нечисловые, определенные стандартом IEEE-754), алгоритмы выполнения базовых арифметических операций, преобразования, округления модулярных мантисс, определения знака, и, как следствие, принципиально иной способ организации вычислений.

1.3. Преимущества модулярно-позиционного формата

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

1) Увеличивается точность вычислений, которая определяется разрядностью мантисс – для достижения точности 4*double достаточно выбрать восемь 32-битных модулей СОК.

2) Благодаря независимости знакопозиций m_i, обеспечивается возможность распараллеливания арифметических операций вплоть до уровня отдельных разрядов мантисс;

3) Решается задача минимизации зависимости времени выполнения операций от точности: при добавлении дополнительных оснований p_i для увеличения точности, в случае, если их число превышает число вычислительных устройств, время выполнения операций увеличивается линейно, но не экспоненциально или квадратично, как в высокоточных позиционных библиотеках.

5) Сопровождение каждой модулярной мантиссы M'' ее позиционной характеристикой $\eta(M'')$ позволяет, в отличие от формата, предложенного в работе [4], эффективно выполнять такие сложные для системы остаточных классов операции, как определение знака, округление, деление на двойку и пр.

6) В ЭВМ работа со значениями знакопозиций m_i будет осуществляться как с переменными скалярных арифметических типов, что позволит применять к ним высоко оптимизированные массовые операции, заложенные в известных библиотеках для организации параллельных вычислений (например, операцию REDUCTION библиотеки OpenMP, либо MPI_REDUCE интерфейса MPI). Применять подобные операции невозможно к числам, представленным в многоразрядных позиционных форматах.

1.4. Области применения модулярно-позиционного формата

Класс задач, при решении которых применение модулярно-позиционного формата может оказаться эффективным, достаточно широк и включает в себя: множество задач, сводимых к выполнению итеративных операций над массивами данных (например, сложные матричные операции над одним наборами исходных данных), задачи численного интегрирования дифференциальных уравнений (задачи подобного рода возникают при моделировании самых разнообразных физических явлений, например, при исследовании долговременной орбитальной эволюции небесных тел), задачи, решаемые с применением рекуррентных формул (например, вычисление суммы рядов), задачи решения СЛАУ, и другие задачи, решение которых составляет множество итераций, в результате чего накопление ошибок округления неизбежно приводит к потере основной доли значащих разрядов результата, и др. Другими словами, использование модулярно-позиционного формата видится целесообразным, когда время, затрачиваемое на прямое и обратное преобразование мало, по сравнению со временем расчетов.

На основе модулярно-позиционного формата разработаны эффективные параллельные алгоритмы выполнения операций (сложение, вычитание, умножение, выравнивание порядков и деление), алгоритмы округления модулярных мантисс (используются итерационные алгоритмы округления мантисс M'' до четного с последующим делением на двойку и увеличением порядков λ на единицу; число итераций округления определяется типом операции, которая должна быть выполнена в следующий момент времени), алгоритмы сравнения и др. В качестве примера на рис. 2 представлены схемы параллельных алгоритмов умножения и сложения модулярно-позиционных операндов.

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

Для реализации операций деления модулярных мантисс (общий случай), определения их выхода за допустимый диапазон представления и преобразования в позиционную систему, являющихся наиболее сложными операциями в СОК, предлагается использовать эффективные методы и алгоритмы, предложенные в работах [3, 5, 6].

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 рис. 3 представлены упрощенные модели этих схем для систем с общей памятью. Для многопроцессорных систем данные схемы аналогичны.

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 рис. 4.

Серверная часть является «ядром» пакета и реализует выполнение всех заложенных в нее алгоритмических решений по обработке модулярно-позиционных структур данных. Сервер 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 на решение конкретной задачи на конкретной вычислительной системе 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, требуемая точность вычислений, схема распределения модулярно-позиционных структур данных.

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.

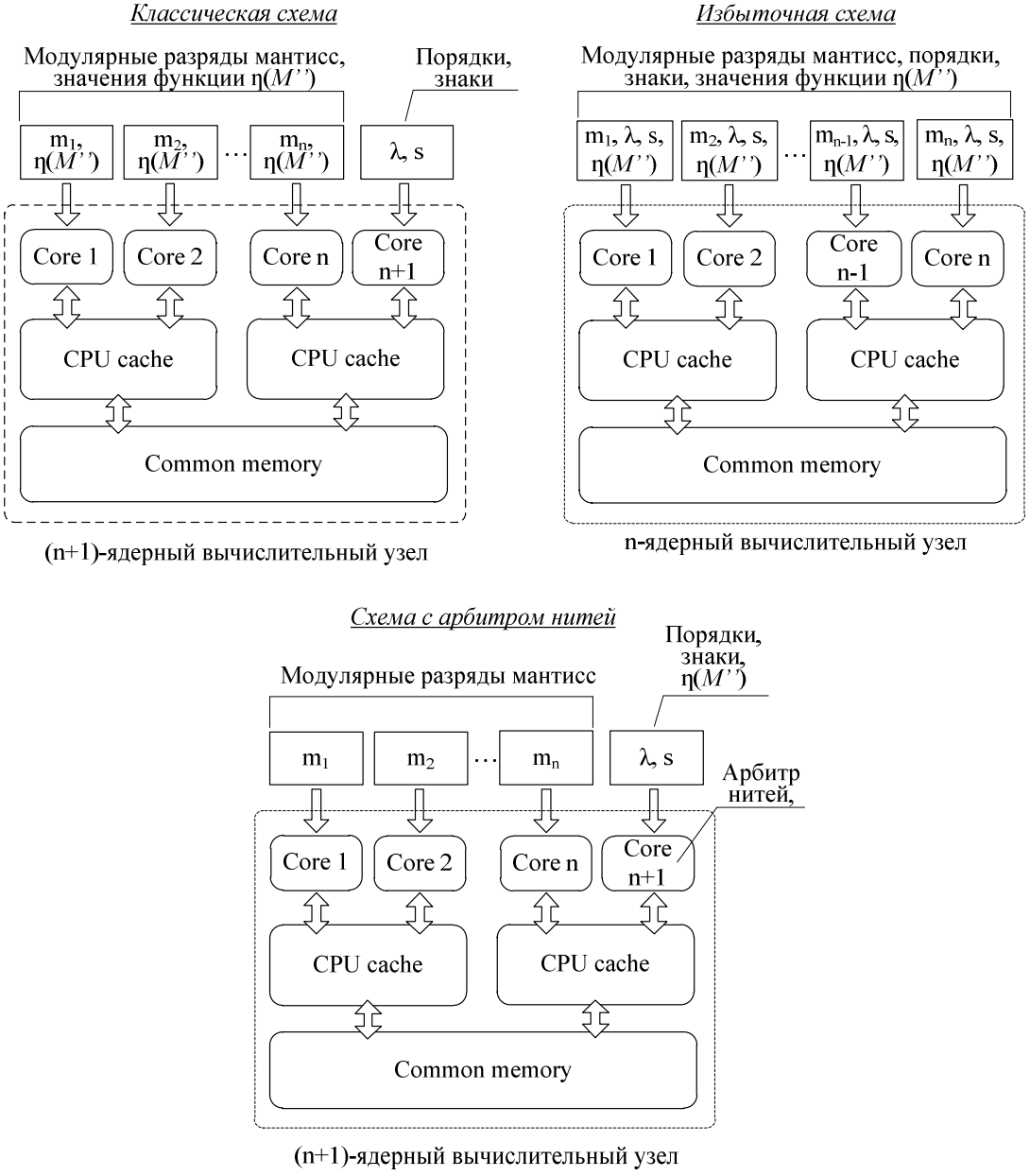

Рис. 3. Схемы распределения модулярно-позиционных структур данных для вычислительных систем с общей памятью

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:

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

3.2) 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). Часть этих операций уже реализована и протестирована.

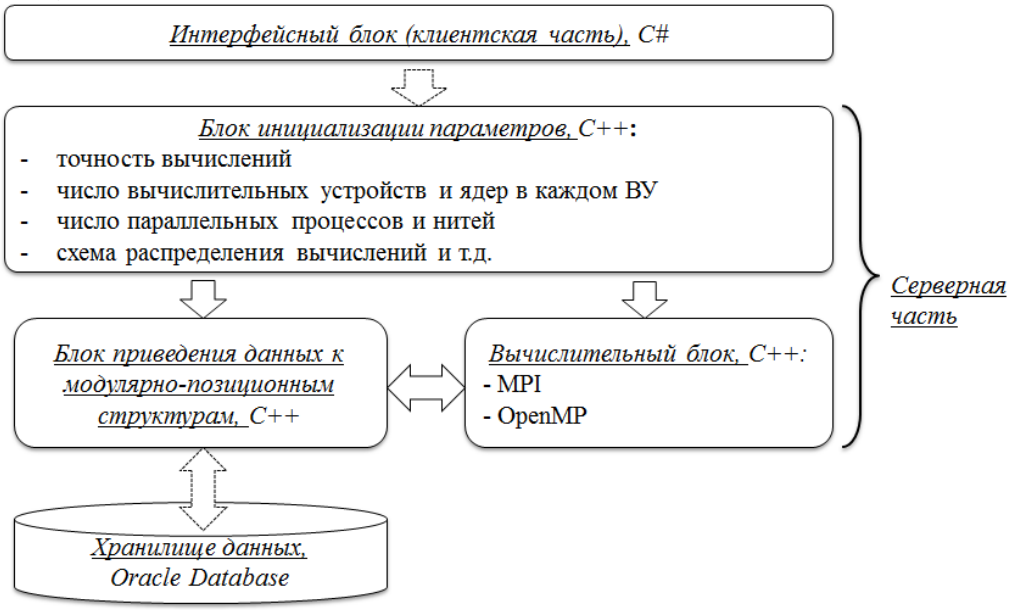

Рис. 4. Обобщенная структура программного пакета HPDP-Solver

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, были проведены эксперименты, направленные 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 $C = A*B$, $A,B,C \in R^{n\times n}$ 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 предполагают 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 рис. 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 $p_1,p_2,...,p_n$, используемых для представления модулярных мантисс $M'' = (m_1,m_2,...,m_n)$, 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 рис. 5 представлены графики зависимостей времени умножения матриц (размерности 1500×1500) от обеспечиваемой вычислительной точности при распараллеливании решения на 8, 16 и 32 вычислительных ядра. Легко заметить, что с увеличением разрядности элементов время решения задачи при использовании библиотеки GMP возрастает существенным образом. Так, при счете на восьми вычислительных ядрах и разрядности мантисс 31 бит задача решается за 431 секунду, а при разрядности мантисс 1891 бит – за 26361 секунд, т.е. более, чем за 7 часов! Таким образом, с увеличением вычислительной точности в 61 раз, время решения задачи линейно возросло также в 61 раз. При использовании пакета HPDP-Solver зависимость времени счета от точности гораздо менее выражена: при разрядности мантисс 31 бит задача была решена 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 известная библиотека высокоточных вычислений GMP (при выполнении вычислений на восьми ядрах). Аналогичные зависимости наблюдаются при счете на 16 и 32 ядрах. Полученные результаты в полной мере соответствуют определенным априорным оценкам.

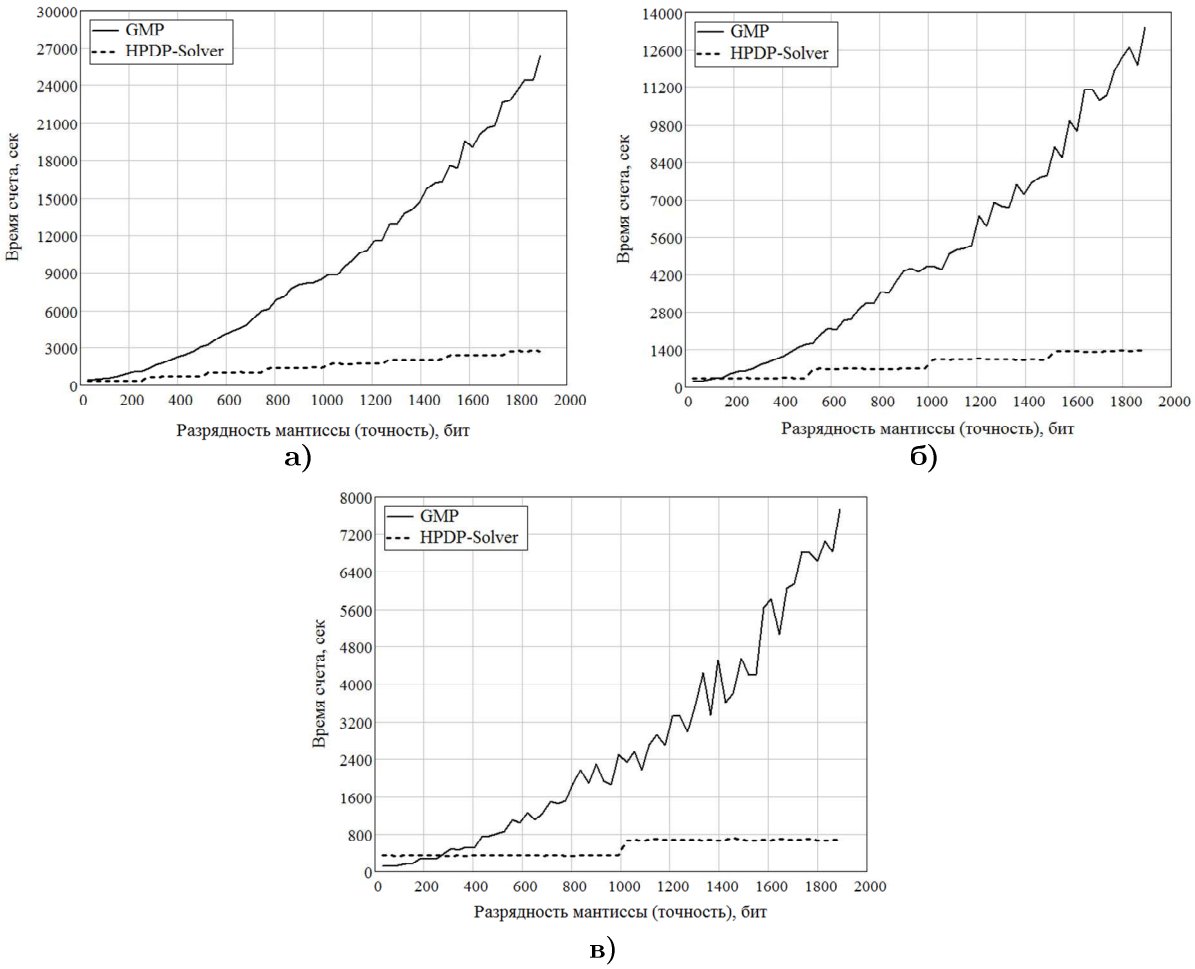

Рис. 5. Изменение времени умножения матриц при росте разрядности мантисс: а) счет выполнялся на 8 вычислительных ядрах, б) счет на 16 ядрах в) счет на 32 ядрах

Стоит заметить, что использование 16 ядер позволило вдвое сократить время счета как при использовании GMP, так HPDP-Solver, на всем диапазоне изменения разрядности мантисс. Аналогично, при счете на 32 ядрах время решения задачи сокращается примерно в 4 раза. Однако стоит учесть, что добиться таких высоких показателей масштабируемости решения на основе библиотеки GMP стало возможным лишь благодаря высокому алгоритмическому параллелизму самой задачи умножения матриц, т.к. 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 рис. 6 представлены сводные результаты исследования быстродействия программного пакета HPDP-Solver от точности вычислений при различном числе используемых параллельно работающих ядер. Анализ этих результатов 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 $M'' = (m_1,m_2,...,m_n)$, позиционная разрядность которых больше, чем 248 бит, необходимо использовать больше

восьми 31-битных оснований p_1,p_2,...,p_n, поэтому 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. 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 рис. 6, останется неизменным.

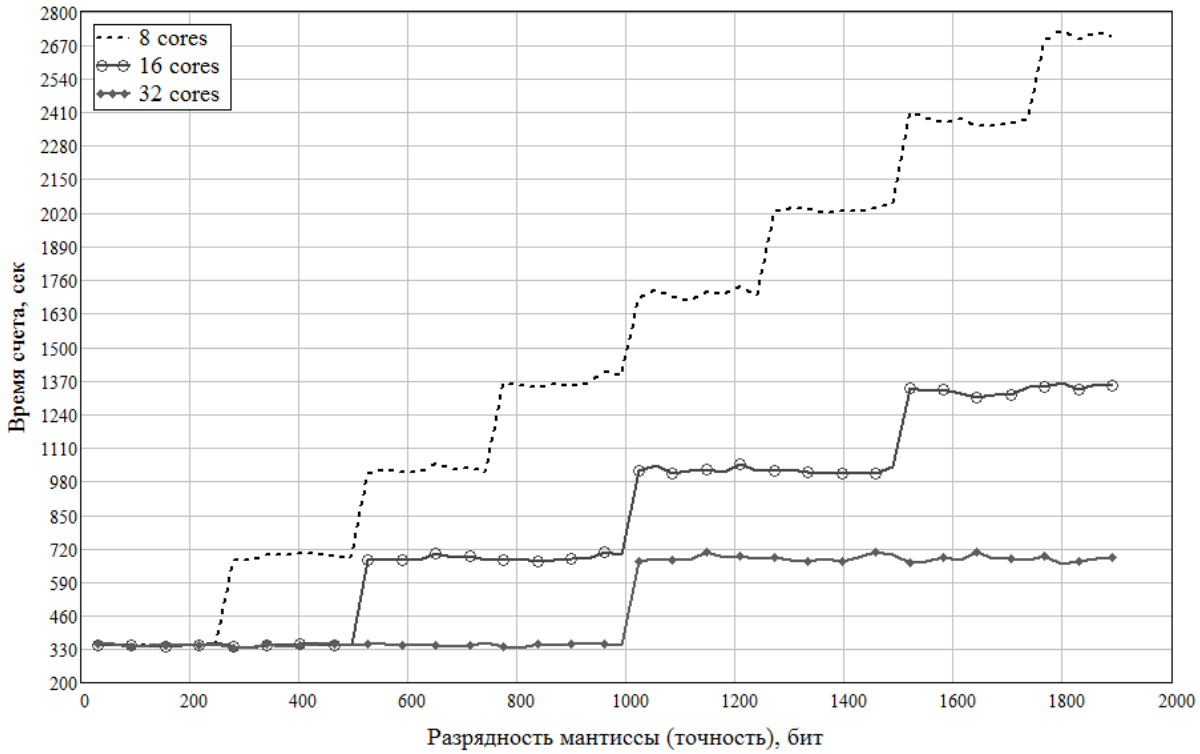

Рис. 6. Сводный график зависимостей быстродействия программного пакета HPDP-Solver от точности при решении задачи матричного умножения с использованием различного числа параллельно работающих вычислительных ядер

На рис. 7 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.

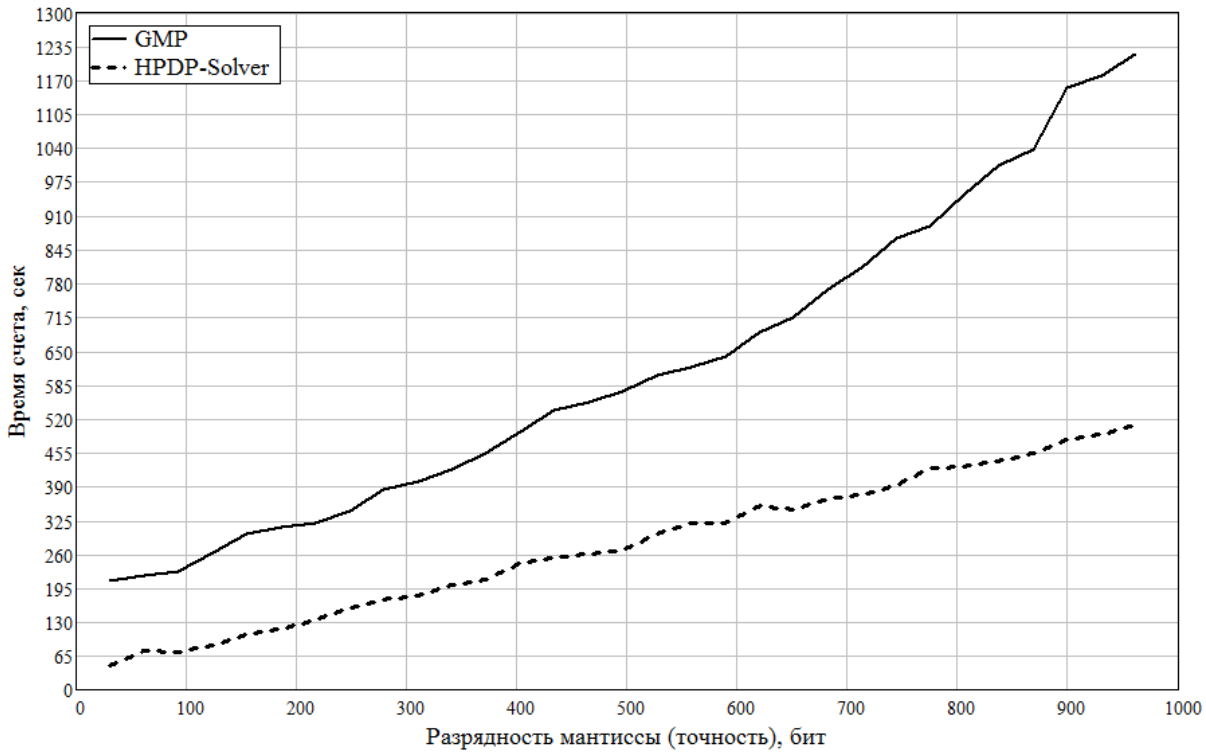

Рис. 7. Зависимость времени скалярного произведения векторов от точности при счете на 8 ядрах

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 $M'' = (m_1,m_2,...,m_n)$ – модулярная мантисса, сопровождаемая позиционной характеристикой $\eta(M'')$, выражающей соотношение между M'' и половиной произведения выбранных модулей, задающих СОК, 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 эксперименты показали преимущества пакета HPDP-Solver 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:

1) Позволяет работать со всеми величинами машинных (IEEE-754) форматов данных с плавающей точкой, но обеспечивает значительно более высокую точность.

2) Позволяет минимизировать зависимость времени вычислений от точности.

3) Обеспечивает параллелизм вычислений на уровне арифметических операций.

4) Имеет возможности для 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 вычислительной точности и сводятся к выполнению массовых арифметических операций с плавающей точкой, когда время, затрачиваемое преобразование данных, существенно меньше времени расчетов. Под эти условия попадает множество задач из самых различных областей науки и техники.

Статья рекомендована к публикации программным комитетом международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений».

Литература

1. Исупов, К.С. Исследование эффективности современных средств поддержки высокоточных вычислений с вещественными числами / К.С. Исупов, А.Г. Иванов / Общество, наука, инновации (НТК-2012): Сб. материалов всероссийской научно-технической конференции (Киров, 16–27 апреля 2012 г.). – Киров: Изд-во ВятГУ, 2012. – 11 с.
2. Акушский, И.Я. Машинная арифметика в остаточных классах / И.Я. Акушский, Д.И. Юдицкий. – М.: Сов. Радио, 1968. – 440 с.
3. Omondi, A. Residue Number Systems: Theory and Implementation (Advances in Computer Science and Engineering Texts) / A. Omondi, B. Premkumar. – Imperial College Press, 2007. – 312 p.
4. Оцоков, Ш.А. Применение модулярной арифметики с фиксированной точкой для ослабления влияния ошибок округления компьютерных вычислений / Ш.А. Оцоков // Информационные технологии. – 2009. – № 12(160). – С. 50–54.
5. Sabbagh, A. New Arithmetic Residue to Binary Converters / A. Sabbagh, K. Navi // IJCSES International Journal of Computer Sciences and Engineering Systems. – 2007. – Vol. 1, No. 4. – P. 295–299.
6. Chang, C. A Division Algorithm For Residue Numbers / C. Chang, Y. Lai // Applied Mathematics and Computation. – 2006. – No. 172(1). – P. 368–378.
7. IEEE Standard for Floating-Point Arithmetic / IEEE. – NY 10016-5997. – USA. - 2008. – 70 p.
8. The GNU Multiple Precision Arithmetic Library. – URL: http://gmplib.org (дата обращения: 05.06.2011).

MODULAR-POSITIONAL DATA FORMAT AND SOFTWARE PACKAGE FOR DIGIT-PARALLEL HIGH-PRECISION FLOATING POINT CALCULATIONS

K.S. Isupov, *Vyatka State University (Kirov, Russian Federation)*

A new way of organization of high-precision floating point computations, which allows parallelizing arithmetic operations down to separate digits of multi-digit floating point mantissas through using a modular-positional data representation format, is considered. The main concept of this format is to represent the floating point mantissas in residue number system (RNS) and the exponent part in positional system. Floating point mantissas go with their positional characteristic that allows to successfully implement efficient algorithms for non-modular operations in RNS, such as division (special case), and rounding. Using this approach a software solution named High Precision Digit-Parallel Solver (HPDP-Solver) is developed. HPDP-Solver can be flexibly configured for a specific PC configuration, resulting in a more efficient use of its resources. The results obtained during the experimental performance study of HPDP-Solver proved its advantages in solving high-precision numerical problems if compared to a world-famous GNU Multiple Precision Arithmetic Library. HPDP-Solver can be used to solve problems that have some special demands on computational precision.

Keywords: floating point, residue number system, modular-positional data format, parallel arithmetic, high-precision calculations.

References

1. Isupov K.S., Ivanov A.G. Issledovanie jeffektivnosti sovremennyh sredstv podderzhki vysokotochnyh vychislenij s vewestvennymi chislami [Research Effectiveness Of Modern Means For High-Precision Calculations with Real Numbers]. Obwestvo nauka innovacii (NTK-2012): Sb. materialov vserossijskoj nauchno-tehnicheskoj konferencii (Kirov, 16-27 aprelja 2012) [Society, Science, Innovation (STC-2012): Sat. Materials of the Russian Scientific and Technologic (Kirov, Russia, April, 16-27, 2012)]. Kirov: Publishing of the Vyatka State University, 2012. 11p.
2. Akushskii I.Y., Yuditskii D.I. Mashinnaja arifmetika v ostatochnyh klassah [Computer Arithmetic in Residual Classes]. Sov. Radio, 1968. – 440 p.
3. Omondi A., Premkumar B. Residue Number Systems: Theory and Implementation (Advances in Computer Science and Engineering Texts. Imperial College Press, 2007. 312 p.
4. Ocokov Sh.A. Primenenie moduljapnoj apifmetiki s fiksipovannoj tochkoj dlja oslablenija vlijanija oshibok okpuglenija komp'jutepnyh vychislenij [Application of Fixed Point Modular Arithmetic for Reducing Influence of Round Errors of computer computation]. Informacionnye tehnologii [Information Technology]. 2009. No. 12(160). P. 50–54.
5. Sabbagh A., Navi K. New Arithmetic Residue to Binary. IJCSES International Journal of Computer Sciences and Engineering Systems. 2007. Vol. 1, No. 4. P. 295–299.
6. Chang C., Lai Y. A Division Algorithm For Residue Numbers. Applied Mathematics and Computation. 2006. No. 172(1). P. 368–378.
7. IEEE Standard for Floating-Point Arithmetic. IEEE. NY 10016-5997. USA. 2008. 70 p.
8. The GNU Multiple Precision Arithmetic Library. URL: http://gmplib.org (accessed: 05.06.2011)

Поступила в редакцию 10 января 2013 г.