Параллельная реализация алгоритма решения задачи энтропийно-робастного оценивания на вычислительных системах гетерогенной архитектуры[1]

А.Ю. Попков, Д.В. Зубарев

Аннотация. Работа посвящена разработке параллельных технологий для численного решения задач энтропийно-робастного оценивания характеристик рандомизированных моделей при малом объеме данных. Предлагается использовать итерационный алгоритм, базирующийся на пакетных итерациях Монте Карло. Алгоритм вместе с большим объемом вычислений, потенциально обладает высокой степенью параллелизма по данным, поэтому его реализация ориентирована на современные вычислительные системы, допускающие массивно-параллельную обработку на разных уровнях параллелизма. Реализация алгоритма ориентирована на стек технологий MPI+OpenMP+CUDA, которые эффективно отображаются на современные гетерогенные архитектуры вычислительных систем. Работоспособность и эффективность предлагаемой технологии подтверждается экспериментальными исследованиями на тестовой задаче.

Ключевые слова: энтропийно-робастное оценивание, параллельные вычисления, Монте Карло, пакетные итерации, гетерогенная архитектура.

Введение

Качественное развитие вычислительной техники, наблюдаемое в последнее десятилетие, привело к появлению новых подходов к решению различных задач, реализация которых ранее была либо слишком трудоемкой, либо невозможной на существовавшем уровне развития соответствующего инструментария.

Одним из таких направлений является довольно обширная тематика, связанная с извлечением полезной информации или «знаний» из некоторого набора «данных». Под одной из интерпретаций этой общей проблемы понимается параметрическое и непараметрическое оценивание характеристик модели с использованием реальных данных. К настоящему времени в различных отраслях науки накоплен большой опыт решения подобных задач в разных областях, среди которых можно выделить математическую статистику [1-3], эконометрику [4-6], финансовую математику [7], теорию управления [8], распознавание образов [9-11] и ряд других научных дисциплин.

Методы, разработанные в рамках этих научных дисциплин, базируются на некоторых предположениях о моделях и используемых данных. Относительно моделей предполагается, что их параметры являются детерминированными, но с неизвестными значениями. Предположения относительно данных обычно

[1] Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 15-07-03005).

базируются на их статистических свойствах, таких, как достаточность их количества для получения статистически значимых оценок, качество выборок, нормальность плотностей распределения вероятностей используемых случайных величин и ряд других.

В [12, 13] предлагается метод энтропийно-робастного оценивания, ключевой особенностью которого является переход к рандомизированной модели, в которой параметры являются случайными объектами. Оценками таких параметров будут являться не значения, а их вероятностные характеристики, которые реализуются функциями плотности распределения вероятностей. Имея оценки вероятностных характеристик появляется возможность извлекать больше «информации» из доступных измерений (данных), получая таким образом более информативный прогноз.

Получение энтропийно-робастных оценок сопряжено с решением задачи функционального энтропийно-линейного программирования. Ключевым этапом ее решения является решение соответствующей системы уравнений, содержащей многомерные интегральные компоненты, не допускающие их эффективное аналитическое исследование. Последнее обстоятельство приводит к трудностям в применении существующих методов решения подобных задач.

Для решения указанной задачи предлагается использовать метод, основанный на пакетных итерациях Монте Карло [14, 15], на каждой итерации которого реализуется трехфазовая процедура, результатом работы которой является приближенное значение глобального экстремума заданной функции, оценка окрестности точного решения и оценка вероятности попадания приближенного значения глобального экстремума в эту окрестность.

Численная реализация пакетных итераций, в процессе которых необходимо вычислять указанные вероятностные характеристики, приводит к большому объему вычислений. Поэтому для обеспечения его эффективности необходимо использовать параллельные вычислительные схемы, ориентированные на реализацию на современных вычислительных системах с гетерогенной архитектурой.

1. Постановка задачи энтропийно-робастного оценивания

Рассмотрим модель объекта в виде «вход–выход», связь между которыми определяется соотношением:

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{a}), \qquad (1)$$

где $\mathbf{F}$ — заданная вектор-функция, $\mathbf{a}$ — вектор параметров модели, $\mathbf{x}$ и $\mathbf{y}$ — соответственно вход и выход. Задача состоит в получении оптимальных относительно некоторого критерия оценок параметров $\mathbf{a}$ используя данные, полученные при «измерении» входов и выходов модели.

Данные, полученные с помощью «измерительного прибора», как правило, содержат ошибки, связанные с его внутренним устройством. Поэтому в соотношении (1) необходимо учитывать наличие ошибок измерений (шума):

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{a}) + \xi, \qquad (2)$$

где ξ - шум измерений выхода соответственно.

Рандомизация приведенной модели состоит в замене ее детерминированных параметров случайными, значения которых вместе с соответствующими шумами принадлежат интервалам:

$$\mathbf{a} \in [\mathbf{a}^-, \mathbf{a}^+], \quad \xi \in [\xi^-, \xi^+].$$

Характеристиками этих параметров и шумов, подлежащими оценке, являются функции их плотностей распределения вероятностей (ПРВ). Используя ограничения, накладываемые на функции ПРВ (нормированность и баланс первого момента наблюдаемого и измеренного значения выхода), и функционал правдоподобия, представляющий собой функционал обобщенной информационной энтропии Больцмана [16], формулируется задача максимизации энтропии, решение которой является требуемой оценкой параметров модели и шумов измерений:

$$\mathcal{H}[P(\mathbf{a}), Q(\xi)] \Rightarrow \max, \qquad (3)$$

где P и Q — функции ПРВ параметров и шумов соответственно.

Используя метод множителей Лагранжа, данная задача максимизации энтропии сводится к задаче безусловной максимизации, которая в свою очередь приводит к задаче поиска решения уравнения вида

$$\Phi(\theta) = 0, \qquad (4)$$

где Ф является вектор-функцией. Решение уравнения (4) сводится к минимизации невязки

$$J(\theta) = \| \Phi(\theta) \| = \left(\sum_k \phi_k^2(\theta) \right)^{\frac{1}{2}} = 0, \quad (5)$$

где ϕ_k — компоненты вектор-функции Ф.

2. Алгоритм решения задачи (5)

Подход к решению задачи (5) основан на GFS-алгоритме [15], который базируется на итерационном процесс, на каждой итерации k которого реализуется трехфазовая процедура SeqLocMin. Вычислительная схема алгоритма представлена на Рис. 1.

Фаза генерации состоит в построении пакета $\mathcal{Z}_k$ из N_k независимых, равномерно распределенных в единичном кубе случайных точек (векторов) $\{\mathbf{z}^1, \dots, \mathbf{z}^{N_k}\}$. Далее, пакет «фильтруется» с тем, чтобы оставить только допустимые точки, сформировав таким образом пакет допустимых векторов $\tilde{\mathcal{Z}}_k$. Для точек из этого пакета вычисляются значения функции $F(\mathbf{z}^j)$ и производится их селекция (выбор), находится экстремальное (минимальное или максимальное) «квази-глобальное» значение F_k^* и соответствующий ему аргумент $\mathbf{z}_k^*$. В силу случайного характера этой процедуры, «квази-глобальное» значение целевой функции F_k^* также случайное. На каждой итерации количество случайных точек увеличивается согласно некоторому правилу, например

$$N_{k+1} = \alpha\, N_k, \quad \alpha > 1.$$

В результате реализации итерационного процесса формируются последовательности «квази-глобальных» минимумов $\mathcal{F}^* = \{F_0^*, \dots, F_k^*, \dots\}$, их декрементов

$$\mathcal{U} = \{u_1, \dots u_k, \dots\},$$

где $u_k = |F_{k+1}^* - F_k^*|$, и логарифмов декрементов

$$\mathcal{V} = \{v_1, \dots, v_k, \dots\},$$

где $v_k = \log u_k$.

Утверждается, что для функций гельдеровского класса при достаточно больших k справедлива следующая линейная регрессия, оценки коэффициентов $\hat{A}_k$, $\hat{s}_k$ которой определяются с помощью метода наименьших квадратов [14]:

$$v_k = A - s\, n_k + \zeta_k$$

где $A = \log L$, $n_k = \log N_k$, $\mathcal{N} = \{n_1, \dots, n_k, \dots\}$, L, s — константы Гельдера, $\{\zeta_k\}$ — последовательность случайных величин с нулевым математическим ожиданием и конечной дисперсией.

На момент остановки итерационного процесса на шаге τ по оценкам $\hat{A}_\tau$, $\hat{s}_\tau$ в рамках

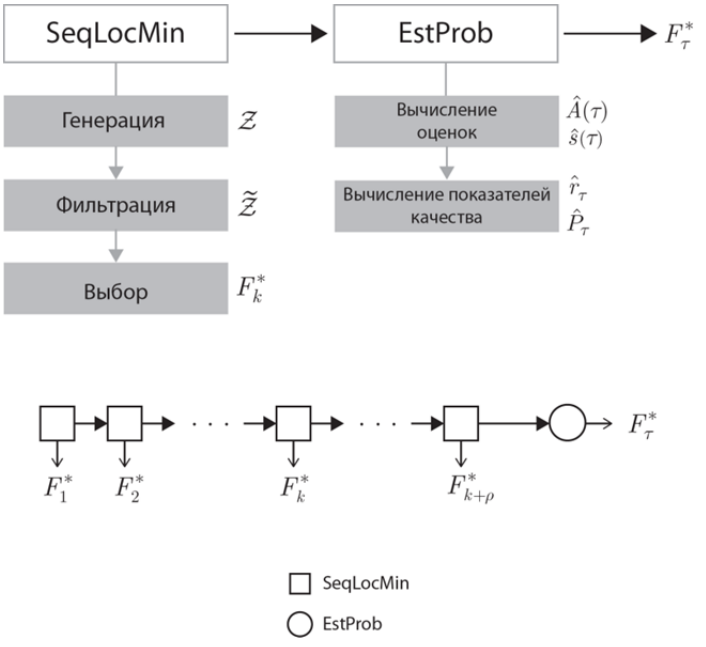


Рис. 1. Вычислительная схема GFS-алгоритма

процедуры EstProb вычисляются оценки размера окрестности точного значения глобального экстремума:

$$|F_\tau^* - F^*| \le \hat{r}_\tau = a^{\hat{A}_\tau} \left(\frac{\sqrt{d}}{2}\right)^{\hat{s}_\tau} N_\tau^{-\frac{\hat{s}}{d}}, \quad (6)$$

где a — основание используемого логарифма, d – размерность пространства, и нижняя оценка вероятности попадания найденного решения в эту окрестность:

$$\hat{P}_\tau = N_\tau \exp\left(-N_\tau^{d-1}\right). \quad (7)$$

3. Реализация алгоритма

Алгоритм характеризуется большим объемом вычислений, поэтому его реализация должна быть ориентирована на высокопроизводительные вычислительные системы. В настоящее время наиболее перспективной и распространенной архитектурой таких систем является гетерогенная архитектура, совмещающая вычислительные устройства различных типов.

К настоящему времени сформировался класс устройств, называемых ускорителями или сопроцессорами (также встречается термин «вычислительный процессор»), использующийся вместе с универсальными процессорами в различных конфигурациях. Современные ускорители построены на основе архитектуры many-core и ориентированы на выполнение массивно-параллельных вычислений [17, 18].

Гетерогенная архитектура в настоящее время является наиболее распространенной, поэтому новые реализации алгоритмов должны быть ориентированы на эту архитектуру в целом или ее конкретную реализацию. Последнее замечание является существенным, так как современные ускорители фактически не имеют общей технологии программирования. Существующие стандартные технологии такие, как OpenMP или OpenCL, либо не поддерживаются, либо их реализация на конкретной платформе недостаточно эффективна. Последнее обстоятельство также необходимо учитывать при разработке, выделяя зависимые от программно-аппаратной платформы компоненты в отдельные модули с тем, чтобы можно было легче переходить на иную реализацию.

Структура GFS-алгоритма допускает различные реализации, однако учитывая целевую архитектуру вычислительных систем, предлагается его параллельная реализация, ориентированная на использования иерархического (многоуровневого) параллелизма на основе стека технологий MPI+OpenMP. Такая архитектура позволяет задействовать как одиночные вычислительные модули с универсальными центральными процессорами, так и системы с распределенной памятью с множеством вычислительных модулей, в том числе оснащенных ускорителями различных архитектур (с помощью соответствующих интерфейсов).

В основе предлагаемой архитектуры лежит широко применяемая в настоящее время программная модель SPMD (Single Program Multiple Data) [19, 20], которая на абстрактном уровне состоит в выполнении экземпляров (копий) программы для каждого элемента набора входных данных одного типа. В рамках модели SPMD вычисления для каждого элемента данных (точки из пакета Z) реализуются в «виртуальных» потоках. Такая архитектура позволяет гибко управлять параллельной структурой алгоритма для адаптации к конфигурации конкретной вычислительной системы. При отображении алгоритма на архитектуру используемой вычислительной системы виртуальные потоки могут отображаться группами в MPI-процессы, и выполняться на центральном процессоре или ускорителе при их наличии. На Рис. 2 показана схема этого подхода и некоторые возможные варианты отображения.

Эта концепция приводит к следующей общей схеме реализации алгоритма. После определения характеристик используемой вычислительной системы, а именно: количество вычислительных узлов (модулей), количества центральных процессоров на каждом узле, количества ускорителей и их тип и минимальные характеристики (в текущей реализации для ускорителей NVidia значение их Compute Capability должно быть не менее 2.0), объем памяти на узле, данные, которые представляют собой пакет точек (векторов) Z для текущей итерации, равномерно разделяются между узлами вычислительной системы и, дополнительно, между ускорителями при их наличии.

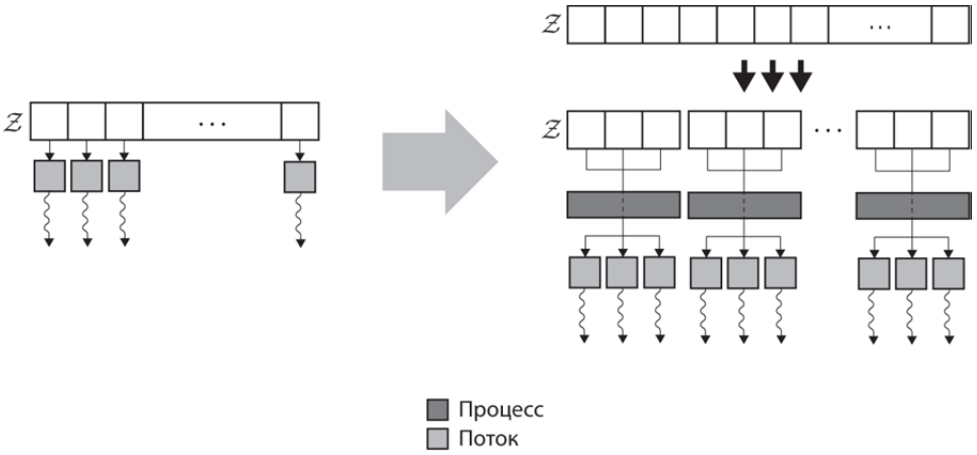


Рис. 2. Отображение алгоритма на архитектуру вычислительной системы

Далее, с помощью доступных вычислительных ресурсов происходит реализация фаз генерации, фильтрации и выбора для заданной функции и области поиска.

Для реализации вычислений в виртуальном потоке используется библиотека Thrust [21, 22]. Главной ее особенностью является предоставление унифицированного со стандартной библиотекой шаблонов C++ интерфейса к структурам данных и алгоритмам, реализованным с помощью технологий OpenMP [23], TBB [24] и CUDA [25]. Необходимо также отметить, что фазы генерации и фильтрации представляют собой абстрактную операцию сканирования (scan), а фаза выбора — абстрактную операцию редукции или свертки (reduce). Эффективная реализация этих операций определяется целевой программно-аппаратной платформой [26]. Таким образом, при использовании библиотеки Thrust появляется возможность изменять реализацию в зависимости от целевой программно-аппаратной платформы.

Вычисление функции для каждого значения аргумента из массива в рамках библиотеки Thrust реализуется функцией thrust::transform, которая применяет к каждому элементу массива функцию, реализованную объектом, называемым функтором (functor). Функтор является так называемым функциональным объектом, который «будучи объектом, может быть вызван так, как будто он является функцией» (пер. авт.) [27]. В рамках языка программирования C++ функторы реализуются объектом класса, в котором перегружен оператор вызова функции. Эта техника позволяет, в частности, обеспечить передачу функций в другие функции в качестве параметров и создавать функции с внутренним состоянием.

Хранение числовых массивов требуемого типа (одинарной (float) или двойной (double) точности) осуществляется с помощью одномерного контейнера thrust::device_vector, конкретная реализация которого зависит от используемой среды выполнения. При использовании процессоров (ускорителей) NVidia и технологии CUDA для работы с ним, такой массив сохраняется в глобальной памяти процессора, а за все перемещения между глобальной памятью процессора и оперативной памятью хоста отвечает библиотека Thrust, подставляя необходимые вызовы CUDA Runtime API при компиляции программного кода.

Точки из пакета Z_k на каждой итерации k хранятся в виде набора контейнеров std::vector<thrust::device_vector>, в каждом из которых хранится соответствующая координата точки (вектора). Таким образом, логическую схему организации хранения точек из пакета Z_k можно представить как матрицу, в которой столбцы представляют точки, а строки — координаты точки в пространстве соответствующей размерности. Необходимо отметить, что данная организация является в общем случае логической, конкретная схема размещения данных в памяти может быть другой. Такая логическая организация точек связана с особенностями

реализации в библиотеке Thrust контейнеров на архитектуре CUDA.

Элементом данных, для которого вычисляется значение требуемой функции, является точка в -мерном пространстве, представленная набором чисел. В силу особенностей организации хранения точек в программных объектах и, соответственно, в памяти, необходимо обеспечить передачу соответствующих элементов контейнеров в функцию (функтор) и при этом сохранить возможность параллельной обработки многих элементов данных. Для решения этой задачи используется итератор специального вида thrust::zip_iterator, который «упаковывает» соответствующие элементы контейнеров в кортеж (tuple), который в свою очередь передается внутрь функтора.

Таким образом, общая схема алгоритма предусматривает реализацию в функторе фаз генерации и фильтрации для одной точки (элемента данных) из пакета Z с последующим применением сканирования с этим функтором для обработки всего пакета. Этот этап реализуется функцией thrust::transform. Фаза выбора реализуется функцией thrust::min_element, которая производит свертку (редукцию) для пакета данных, полученных на предыдущем этапе.

4. Экспериментальное исследование

Для исследования реализации алгоритма предлагается использовать тестовую задачу оценивания параметров дискретной динамической рандомизированной модели (ДДРМ) со структурированными нелинейностями второй степени и аддитивным случайным шумом.

Рассмотрим модель нелинейного динамического объекта вида (1), где $x = \{x[0], \dots, x[m], \dots, x[m+s]\}$ - вектор входных данных, $y = \{y[m], \dots, y[m+s]\}$ - вектор выходных данных объекта. Предполагается, что вход измеряется точно, а выход с ошибками, которые моделируются случайным вектором $\xi = \{\xi[m], \dots, \xi[m+s]\}$ с независимыми компонентами. Компоненты этого вектора интервального типа, то есть

$$\xi[j+m] \in \Xi_j = [\xi_j^-, \xi_j^+], \text{ для всех } j = \overline{0,s}.$$

На этих интервальных множествах существуют функции ПРВ $q_j(\xi[j+m]), j = \overline{0,s}$.

Зависимость между пакетами входных и выходных данных, с учетом ошибок измерений выхода объекта, описывается соотношением:

$$y[j+1] = \sum_{h=1}^{R} \left(\sum_{n=0}^{m} w^{(h)}[n] x[j+m-n] \right)^h + \\ + \xi[j+m], \quad j = \overline{0,s}, \tag{8}$$

где объем «памяти» $m = 1$, количество измерений $s = 1$ и $R = 2$.

Дискретные случайные весовые функции (случайные импульсные характеристики) $w^{(1)}[n] = w^{(2)}[n] = 0,\ n > 1$ описывают динамические свойства линейного и нелинейного блоков модели. Введем обозначения:

$$a_0^{(1)} = w^{(1)}[0],\ a_1^{(1)} = w^{(1)}[1];\ a^{(1)} = \{a_0^{(1)}, a_1^{(1)}\};$$

$$a_0^{(1)} = w^{(1)}[0],\ a_1^{(1)} = w^{(1)}[1];\ a^{(1)} = \left\{a_0^{(1)}, a_1^{(1)}\right\};$$

$$a_0^{(2)} = w^{(2)}[0],\ a_1^{(2)} = w^{(2)}[1];\quad a^{(2)} = \{a_0^{(2)}, a_1^{(2)}\};$$

$$x[j] = \{x[j+1], x[j]\}.$$

Здесь $x[0] = \{0{,}39; 0{,}19\}$, и $x[1] = \{0{,}93; 0{,}39\}$.

Случайные параметры $a_n^{(h)}$ - интервального типа, то есть они принимают значения на следующих интервальных множествах:

$$\mathcal{A}_n^{(h)} = [-A_n^{(h)}, A_n^{(h)}],\quad A_n^{(h)} = \beta^{(h)} \exp(-\alpha^{(h)} n).$$

Здесь $\alpha^{(1)} = 0{,}5$, $\alpha^{(2)} = 1{,}0$, $\beta^{(1)} = 1{,}0$, $\beta^{(2)} = 2{,}0$.

На интервальных множествах $\mathcal{A}_n^{(h)}$ определены функции ПРВ $P^{(1)}(\mathbf{a}^{(1)})$, $P^{(2)}(\mathbf{a}^{(2)})$.

Согласно методу энтропийно-робастного оценивания [12] энтропийно-оптимальные функции ПРВ параметров ДДРМ и шумов измерений будут иметь вид:

$$\tilde{P}^{(h)}(\mathbf{a}) = \\ = \frac{1}{\mathcal{R}^{(h)}(\theta)} \exp\left(-\sum_{j=0}^{s} \theta_j V_j^{(h)}(\mathbf{a}^{(h)}, x[j]) \right), \\ h = \overline{1,R};$$

$$\tilde{q}_j(\xi) = \frac{1}{\mathcal{Q}_j(\theta_j)} \exp(-\theta_j \xi),\quad j = \overline{0,s},$$

Табл. 1. Результаты решения тестовой задачи

M	Π_θ	J_τ^*	θ_τ^*	ε_τ	$\hat{r}_\tau$	$\hat{P}_\tau$	t (с)
10^6	$\theta_0 \in [-10,10]$ $\theta_1 \in [-10,10]$	0.0021	-2.4203 - 0.0603	0.0021	0.0558	≈ 1.0	1143
10^6	$\theta_0 \in [-3,-2]$ $\theta_1 \in [-0.5,0.5]$	0.0003	-2.4195 -0.0596	0.0003	0.0069	≈ 1.0	1142
10^7	$\theta_0 \in [-10,10]$ $\theta_1 \in [-10,10]$	0.0038	-2.3030 -0.0828	0.0038	0.1570	≈ 1.0	11429
10^7	$\theta_0 \in [-3,-2]$ $\theta_1 \in [-0.5,0.5]$	0.0001	-2.3930 -0.0657	0.0001	0.0038	≈ 1.0	11426

где

$$V_j^{(1)}(\mathbf{a},x) = a_0^{(1)}x[j+1] + a_1^{(1)}x[j],$$
$$V_j^{(2)}(\mathbf{a},x) = \left(a^{(2)}x[j+1]\right)^2 + \left(a^{(2)}x[j]\right)^2 + 2a_0^{(2)}a_1^{(2)}x[j+1]x[j],$$

$$\mathcal{R}^{(h)}(\theta) = \int_{\mathcal{A}^{(h)}} \exp\left(-\sum_{j=0}^{s} \theta_j V_j^{(h)}\left(\mathbf{a}^{(h)}, x[j]\right)\right) d\mathbf{a}^{(h)},$$

$$\mathcal{Q}_j(\theta) = \int_{\Xi_j} \exp\left(-\theta_j \xi_j\right) d\xi_j, \quad j = \overline{0,s}.$$

Множители Лагранжа θ_0, θ_1 определяются решением следующей системы уравнений:

$$\Phi_j(\theta) = \sum_{h=1}^{R} \frac{\tilde{\mathcal{R}}_j^{(h)}(\theta)}{\mathcal{R}^{(h)}(\theta)} + \frac{\tilde{\mathcal{B}}_j(\theta_j)}{\mathcal{B}_j(\theta_j)} - y[j+1] = 0, \quad j = \overline{0,s}, \qquad (9)$$

где интегральные компоненты имеют вид:

$$\tilde{\mathcal{R}}_j^{(h)}(\theta) = \int_{\mathcal{A}^{(h)}} V_j^{(h)}\left(\mathbf{a}^{(h)}, x[j]\right) \times \exp\left(-\sum_{i=0}^{1} \theta_i V_i^{(h)}\left(\mathbf{a}^{(h)}, x[j]\right)\right) d\mathbf{a}^{(h)}, \qquad (10)$$

$$\tilde{\mathcal{B}}_j(\theta) = \int_{\Xi_j} \xi_j \exp\left(-\theta_j \xi_j\right) d\xi_j.$$

Измерения значений выхода объекта $y[1] = 0{,}5;\ y[2] = 2{,}2$.

Для оценки точности получаемых значений и верификации программного кода, данная тестовая задача была решена с помощью методов, реализованных в пакете MATLAB. Начальные условия для этих методов задавались в нескольких точках области поиска случайно или с помощью равномерной сетки. Интегральные компоненты вычислялись функцией integral2, реализующей численный метод, основанный на квадратурных формулах и обеспечивающий высокую точность. Решение задачи было получено с точностью (абсолютной погрешностью) более 10^{-4} и равно $\theta_0 = -2{,}4007,\ \theta_1 = -0{,}0642$.

Результаты расчетов различных конфигураций задачи представлены в Табл. 1. Интегральные компоненты (10) вычислялись методом Монте Карло по формуле [28, 29]:

$$I \approx \frac{V}{M} \sum_{i=1}^{M} f(x_i),$$

где f - подынтегральная функция, x_i - точки, равномерно распределенные в требуемой области, M - количество таких точек, V - объем области интегрирования. Решение уравнения (9) определялось для двух областей поиска Π_θ и различного количества точек интегрирования. Во всех расчетах $N_0 = 10,\ \alpha = 10,\ a = 10,\ \tau = 6,\ \varepsilon_\tau = |F^* - F_\tau^*|,\ t$ - время в секундах, затраченное на получение решения, и $\hat{r}_\tau,\ \hat{P}_\tau$ - оценки окрестности полученного приближенного решения от точного и его вероятности. Все расчеты проводились с использованием чисел с двойной точностью.

Вычисления проводились на вычислительной системе из двух узлов со следующими характеристиками:

- сервер (Тип 1), оснащенный 2 процессорами Intel Xeon X5650 (6 ядер, 12 потоков), работающим на частоте 2,66 ГГц, 32 Гб оперативной памяти и 2 ускорителями NVidia Tesla M2050;

- сервер (Тип 2), оснащенный 2 процессорами Intel Xeon E5-2650V2 (8 ядер, 16 потоков), работающим на частоте 2,6 ГГц, 64 Гб оперативной памяти и 2 ускорителями NVidia Tesla K20Xm.

5. Оценки производительности

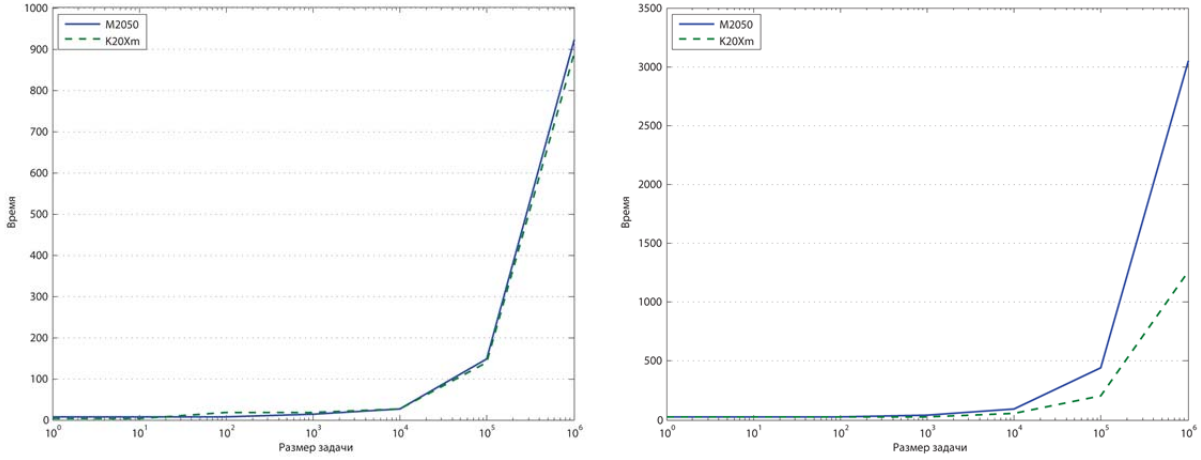
Оценка производительности проводилась с использованием вычислительных узлов, которые использовались при решении тестовой задачи. Расчеты каждой конфигурации задачи проводились отдельно на каждом узле с двойной точностью, производилось 10 независимых испытаний, соответствующие показатели вычислялись на основе средних значений. Во всех расчетах использовался один MPI-процесс и доступные ускорители, доступ к которым осуществлялся из параллельных OpenMP-потоков. Результаты расчетов представлены в Табл. 2 и на Рис. 3.

Расчеты, проведенные отдельно на каждом из узлов показали, что общая производительность вычислительной системы ограничена производительностью менее производительного узла. Этот факт подтверждается теорией и говорит о том, что при использовании подобных неоднородных по структуре вычислительных модулей систем необходимо прибегать к соответствующей балансировке нагрузки. В текущей реализации такой балансировки не производится, данные разделяются равномерно в зависимости от доступных вычислительных модулей.

Согласно спецификациям производителя [30, 31] процессор M2050 имеет 448 процессорных ядер и пиковую производительность при операциях двойной точности 515 гигафлопс (GFlops), в то время, как процессор K20Xm имеет 2688 процессорных ядер и более чем вдвое большую пиковую производитель-

Табл. 2. Время решения тестовой задачи, с

Процессор	M	1	10	10^2	10^3	10^4	10^5	10^6	10^7
2 × M2050	10^6	7,97	7,97	7,98	12,97	25,72	147,41	923,35	8810,40
	10^7	27,40	27,40	27,40	43,33	92,80	443,03	3052,08	
2 × K20Xm	10^6	2,67	2,73	17,18	17,18	25,57	139,41	889,70	7976,71
	10^7	26,74	27,22	27,38	27,30	58,86	203,97	1265,50	



а) б)

Рис. 3. Время расчета при $M = 10^6$ (а) и $M = 10^7$(б)

ность при операциях двойной точности равную 1310 гигафлопс (1,31 терафлопс). Результаты измерений показывают, что время вычислений растет линейно (экспоненциально в логарифмической шкале) с ростом размера задачи. При $M = 10^6$ наблюдается приблизительно одинаковый характер роста для ускорителей разных поколений, однако при $M = 10^7$ преимущество более совершенной архитектуры Kepler становится более очевидным и согласуется с данными, предоставляемыми производителем.

Также необходимо отметить, что при относительно небольшом размере задачи архитектура CUDA используется неэффективно [25, 32], поэтому не наблюдается заметного роста времени выполнения с ростом размера. При размере задачи $N \geq 10^4$ оборудование используется эффективно, поэтому рост времени вычислений соответствует увеличению размера задачи.

Заключение

В работе предложена реализация алгоритма решения задачи энтропийно-робастного оценивания, ориентированная на высокопроизводительные вычислительные системы гетерогенной архитектуры. Структура алгоритма характеризуется наличием параллелизма по данным, что позволяет разработать его реализацию, обладающую чрезвычайным параллелизмом. Тестирование алгоритма проведено на примере задачи оценивания параметров дискретной динамической рандомизированной модели с конечной памятью с использованием вычислительных узлов гетерогенной архитектуры, оснащенных ускорителями NVidia. Реализация базируется на стеке технологий MPI+OpenMP+CUDA. Экспериментальные исследования подтверждают работоспособность алгоритма.

Литература

1. Кендал М.Д., Стьюарт А. Статистические выводы и связи. − М.: Наука, 1973.
2. Крамер Г. Математические методы статистики. − М.: Мир, 1975.
3. Кобзарь А.И. Прикладная математическая статистика. − М. : Физматлит, 2006.
4. Айвазян С.А., Мхитарян В.С. Прикладная статистика и основы эконометрики. − М.: Юнити, 2001.
5. Golan A., Judge G., Miller D. Maximum Entropy Econometrics: Robust Estimation with Limited Data. − New York: John Wiley & Sons, 1996.
6. Golan A. Information and entropy econometrics - a review and synthesis // Foundation and Trends in Econometrics. 2006. Vol. 2, No. 1-2. P. 1–145.
7. Ширяев А. Н. Стохастическая финансовая математика. − М.: Наука, 2002.
8. Поляк Б.Т., Щербаков П.С. Робастная устойчивость и управление. − М.: Наука, 2002.
9. Вапник В.Н., Червоненкис А.Я. Теория распознавания образов. − М.: Наука, 1974.
10. Мерков А.Б. Распознавание образов: Введение в методы статистического обучения. − М. : URSS, 2011.
11. Мерков А.Б. Распознавание образов: Построение и обучение вероятностных моделей. − М.: ЛЕНАНД, 2014.
12. Попков Ю.С., Попков А.Ю., Лысак Ю.В. Оценивание характеристик рандомизированных статических моделей данных (энтропийно-робастный подход) // Автоматика и телемеханика. 2013. № 11. С. 114–131.
13. Попков Ю.С., Попков А.Ю., Лысак Ю.В. Оценивание характеристик рандомизированных динамических моделей данных (энтропийно-робастный подход) // Автоматика и телемеханика. 2013. № 5. С. 83–90.
14. Дарховский Б.С., Попков А.Ю., Попков Ю.С. Метод пакетных итераций Монте Карло для решения систем нелинейных уравнений и неравенств: вероятностные характеристики // Автоматика и телемеханика. 2015. № 5. С. 87–98.
15. Дарховский Б.С., Попков А.Ю., Попков Ю.С. Итерационный МК-алгоритм решения задач глобальной оптимизации // Автоматика и телемеханика. 2015, в печати.
16. Больцман Л. О связи между вторым началом механической теории теплоты и теорией вероятностей в теоремах о тепловом равновесии // Больцман Л.Э. Избранные труды. Серия «Классики науки.» / Под ред. Л.С. Шлак. − М.: Наука, 1984.
17. Mittal S., Vetter J.S. A Survey of CPU-GPU Heterogeneous Computing Techniques // ACM Computing Surveys. 2015. Vol. 47, No. 4. P. 1–35.
18. Vajda A. Multi-core and many-core processor architectures // Programming Many-Core Chips / Ed. by András Vajda. − Boston, MA: Springer US, 2011. P. 9–43.
19. Dongarra J.J. Sourcebook of parallel computing. − San Francisco CA : Morgan Kaufmann Publishers, 2003.
20. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. − СПб.: БХВ-Петербург, 2004.
21. Bell N., Hoberock J. Thrust: A Productivity-Oriented Library for CUDA // GPU Computing Gems Jade Edition / Ed. by Wen-mei W. Hwu. − San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. P. 359–371.
22. Thrust — Parallel Algorithms Library. URL: http://thrust.github.io/ (Дата обращения: 2.11.2015).
23. OpenMP Specifications. URL: http://openmp.org/wp/openmp-specifications/ (Дата обращения: 2.11.2015).
24. Threading Building Blocks Documentation. URL: https://www.threadingbuildingblocks.org/ documentation/ (Дата обращения: 2.11.2015).
25. About CUDA. URL: https://developer.nvidia.com/about-cuda (Дата обращения: 2.11.2015).

26. Lin C., Snyder L. Principles of parallel programming. − Boston, MA : Pearson/Addison Wesley, 2009.
27. STL Function Objects. URL: http://www.sgi.com/tech/stl/functors.html (Дата обращения: 2.11.2015).
28. Соболь И.М. Метод Монте-Карло. − М.: Наука, 1968.
29. Соболь И.М. Численные методы Монте-Карло. − М.: Наука, 1973.
30. Tesla M2050 / M2070 GPU Computing Module. URL: http://www.nvidia.co.uk/object/ product_tesla_M2050_ M2070_uk.html (Дата обращения: 2.11.2015).
31. NVidia Tesla Kepler Family Datasheet. URL: http://www.nvidia.com/content/tesla/pdf/ nvidia-tesla-kepler-family-datasheet.pdf (Дата обращения: 2.11.2015).
32. Kirk D., Hwu Wen-mei. Programming massively parallel processors: A Hands-on approach. 2nd edition. − Amsterdam and Boston: Elsevier/Morgan Kaufmann, 2013.

Попков Алексей Юрьевич. Ведущий научный сотрудник. Института системного анализа РАН Федерального исследовательского центра «Информатика и управление» РАН. Окончил МГУ им. М.В.Ломоносова в 2002 году. Кандидат технических наук. Автор 20 печатных работ. Область научных интересов: параллельные вычисления, параллельные алгоритма, энтропийные модели и методы. E-mail: apopkov@isa.ru

Зубарев Денис Владимирович. Инженер-исследователь Института системного анализа РАН Федерального исследовательского центра «Информатика и управление» РАН. Окончил Российский университет дружбы народов в 2014 году. Автор 3 печатных работ. Область научных интересов: искусственный интеллект, поисковые системы, распределенные системы. E-mail: zubarev@isa.ru