Параллельная производительность стохастических алгоритмов*

С.В. Ковальчук, И.А. Пименов, А.В. Бухановский

Обсуждаются вопросы построения моделей производительности параллельных стохастических алгоритмов для различных классов задач. Предложен параметрический подход, который позволяет описать параллельное ускорение в форме детерминированной функции. Ее параметрами являются случайные величины, характеризующие объективные свойства алгоритма, и не зависящие от особенностей программно-аппаратной реализации. Это позволяет в аналитической форме исследовать зависимость вероятностных характеристик производительности от алгоритма и специфики параллельной вычислительной архитектуры.

1. Введение

Для решения широкого круга научных и инженерных задач применяются стохастические вычислительные алгоритмы, основанные на операциях с последовательностями псевдослучайных чисел. К ним относятся, например, численные методы Монте-Карло, генетические алгоритмы, случайные алгоритмы на графах и др. В силу относительно слабой сходимости, обусловленной статистическими свойствами алгоритмов, получение достоверного результата требует больших объемов вычислений, что, в свою очередь, делает обоснованным применение параллельных алгоритмов и соответствующих им методов и технологий параллельного программирования [1].

Важнейшей характеристикой параллельной программы, исполняемой на p вычислителях (ядрах, процессорах, узлах) является ее производительность. Она характеризуется как общим временем работы T_p, так и безразмерными характеристиками – параллельным ускорением S_p и эффективностью ε_p. Классическое определение параллельного ускорения, как отношения времени выполнения программы на одном и p вычислителях при одинаковых условиях [2], дает надежный способ экспериментального определения S_p для обычных, детерминированных алгоритмов. На его основе предложены различные методы моделирования производительности, которые позволяют предсказывать поведение алгоритма на различных вычислительных архитектурах (например [3-4]).

Экспериментально определить, и, тем более, спрогнозировать параллельную производительность стохастического алгоритма, пользуясь традиционными подходами, оказывается затруднительно. Это обусловлено тем, что значения времени работы T_1, T_p последовательной и параллельной программ могут быть случайными величинами. Как следствие, каждое измерение времени работы программы соответствует в общем случае *разным* условиям эксперимента, что не позволяет для ее измерения напрямую воспользоваться классическими определениями ускорения и эффективности.

Исследования производительности стохастических алгоритмов выполнялись применительно к некоторым классам задач (итерационным методам решения задач оптимизации), см. например [5-7]. В данной работе предлагается вероятностный подход к построению моделей производительности параллельных стохастических алгоритмов для различных классов задач, на основе представления времени работы программы в виде детерминированной функции случайных аргументов – параметров самого алгоритма.

*Работа выполнена при частичной финансовой поддержке гранта Intel SPB/Don/118/2006 «Естественные технологии распараллеливания алгоритмов высокопроизводительной обработки данных для многопроцессорных вычислительных комплексов на основе многоядерных микропроцессоров».

2. Классы параллельных стохастических алгоритмов

На рис. 1 параллельные стохастические алгоритмы классифицированы по признаку влияния источника стохастизации на время выполнения программы. Квазидетерминированный алгоритм (рис. 1а) является вырожденным случаем стохастического алгоритма: каждый вычислитель выполняет одни и те же операции с одинаковым, наперед заданным количеством псевдослучайных чисел. Такие алгоритмы характерны, например, для задачи вычисления многомерных интегралов методом Монте-Карло [8]. Алгоритмы со случайным выходом (рис. 1б-в) соответствуют итерационным алгоритмам, применяемым, например, в задачах оптимизации [9,10]. Алгоритмы со случайным накоплением (рис. 1г-д) характерны для методов, основанных на многократном повторении испытаний со случайным выходом, например, решение систем линейных уравнений методом Монте-Карло [11]. Отдельный класс представляют алгоритмы со случайными входными данными (рис. 1е): для них время выполнения программы определяется совокупным влиянием источника стохастизации самого алгоритма и изменчивостью данных.

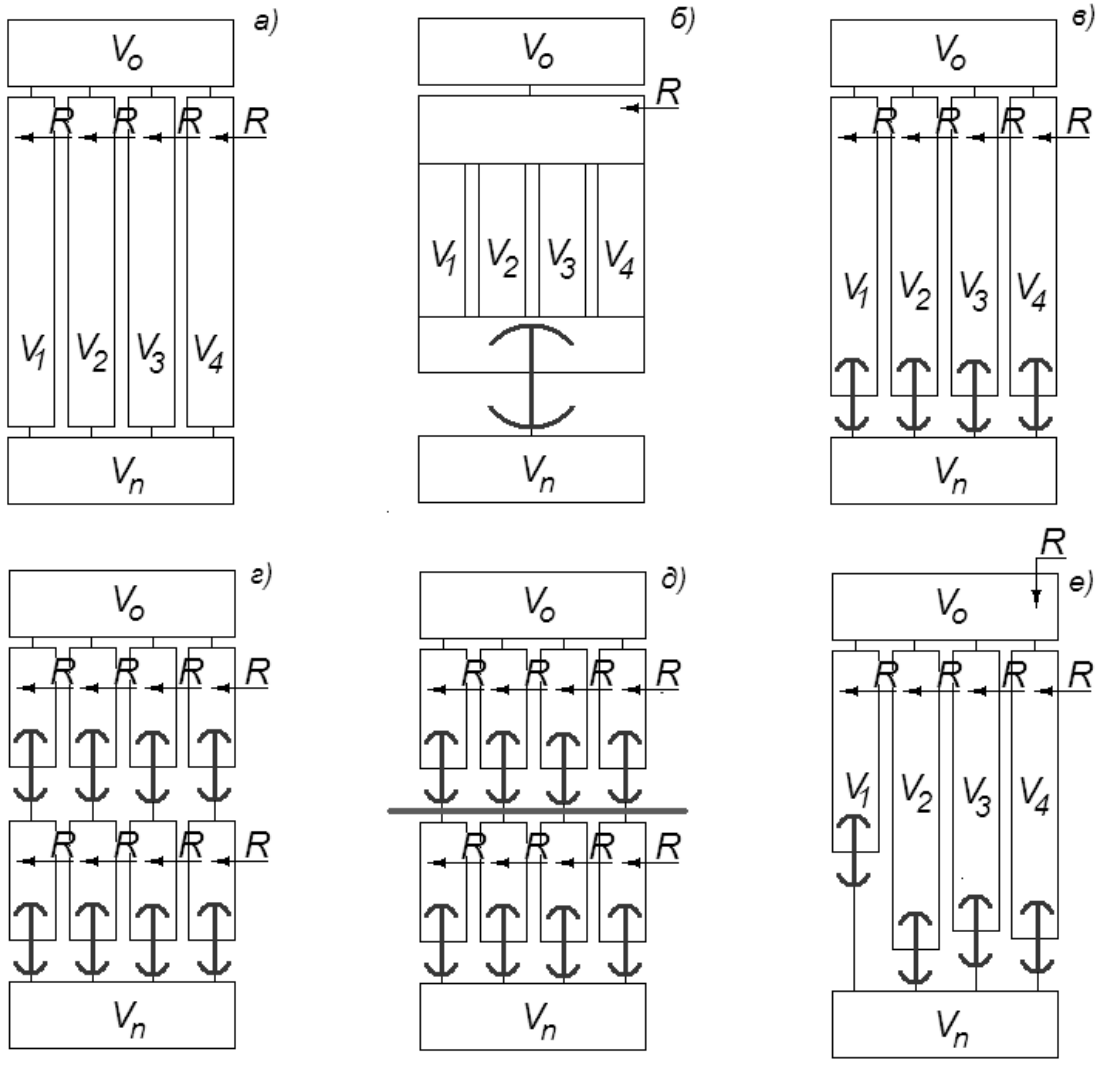

Рис. 1. Основные классы параллельных стохастических алгоритмов

3. Вероятностный подход к описанию параллельной производительности

Из рис. 1 следует, что время работы программы, реализующей параллельный стохастический алгоритм, определяется структурой самого алгоритма и характеристиками псевдослучайных величин, составляющих источник стохастизации. Это дает основание представить время

работы программы на p вычислителях в форме детерминированной функции $T_p = T(p, R)$, зависящей от набора аргументов R, которые являются случайными величинами. Если известна плотность распределения $f_R(x)$, то плотность распределения времени выполнения программы $f_{T_p}(x)$, а также соответствующие плотности распределения параллельного ускорения $f_S(x)$ и эффективности $f_\varepsilon(x)$ выражаются через нее однозначно [12]. Рассмотрим применимость данного подхода на конкретных примерах алгоритмов в рамках PRAM-модели параллельной программы [14], реализуемой посредством технологии OpenMP для систем с общей памятью. В рамках PRAM-модели накладные расходы, затрачиваемые на создание потоков, в первом приближении не учитываются, что упрощает содержательную интерпретацию результатов

3.1 Алгоритмы со случайным выходом без барьера

Параллельный алгоритм со случайным выходом без барьера (рис. 1б) является традиционным для численного решения задач оптимизации, например, разными модификациями метода случайного поиска [14]. Этот алгоритм задает управляемый случайной последовательностью итерационный процесс, на каждом шаге которого целевая функция оптимизации вычисляется параллельно. Собственно в параллельных вычислениях случайная последовательность не используется. Однако она определяет общее количество итераций n для достижения заданной точности, и, как следствие, время выполнения работы алгоритма:

$$T_1 = T_0 + nT_c \qquad \text{и} \qquad T_p = T_0 + n\frac{T_c}{p}. \tag{1}$$

Здесь T_0 - операции вне итерационного цикла (последовательная часть кода), T_c - трудозатраты на одну итерацию поиска (вычисление целевой функции, параллельная часть кода).

Число итераций n является случайной величиной. В общем случае закон ее распределения неизвестен и во многом зависит от вида целевой функции и характеристик решаемой задачи. Однако в ряде случаев вывод о классе распределения может быть сделан на основании общих соображений. В частности, для задач случайного поиска, сводящихся к последовательности положительных и отрицательных испытаний (зависимых испытаний Бернулли) общее количество итераций тяготеет к нормальному распределению. В дальнейшем мы ограничимся этим предположением, и будем характеризовать число итераций в (1) двумя параметрами – средним значением m_n и среднеквадратичным отклонением σ_n. На рис. 2а в качестве подтверждения приведен квантильный биплот распределения числа итераций в процедуре адаптивного случайного поиска с линейной тактикой в рамках задачи аппроксимации климатических спектров морского волнения [15]. Квантильный биплот представляет собой двумерный график, по одной из осей которого отложены значения порядковых статистик рассматриваемой величины, а по другой — теоретические квантили соответствующего распределения. Подобное построение позволяет оценить соответствие распределения величины теоретическому по степени приближенности графика к прямой линии.

В рамках предположений о нормальности n величины T_1 и T_p также будут нормально распределены. Тогда параллельное ускорение

$$S(p) = \frac{T_1}{T_p} = \frac{T_0 + nT_c}{T_0 + n\dfrac{T_c}{p}}, \tag{2}$$

при одной и той же величине n в числителе и знаменателе, будет иметь плотность распределения

$$f(S) = f(n(S))\left|n'(S)\right|, \tag{3}$$

где обратная к (2) функция

$$n(S) = \frac{T_o}{T_c}\frac{S-1}{1-S/p}, \tag{4}$$

выражает зависимость числа итераций через ускорение системы. На рис. 2б приведен пример сопоставления аналитического выражения (3) с результатами экспериментального определения ускорения в терминах (2), полученного путем одинаковой инициализации последовательностей псевдослучайных чисел при вычислениях на 1 и 2 ядрах. Данные представлены в форме вероятностного биплота. Из рис. 2б видно, что выражение (3) хорошо отображает данные измерений.

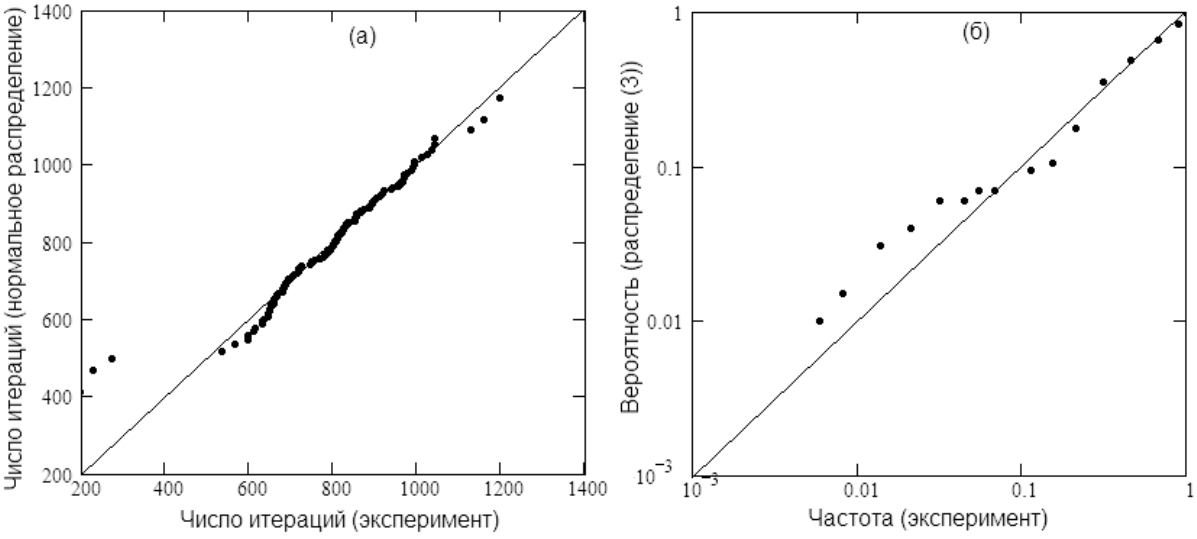

Рис. 2. Биплоты распределений числа итераций метода линейного поиска (а) и распределения (3) параллельного ускорения (б). По оси абсцисс отложены данные эксперимента, по оси ординат – квантили (вероятности) теоретических распределений.

На рис. 3а приведен пример результата оценки распределений параллельного ускорения по (3) в зависимости от количества вычислителей. Несмотря на то, что само время выполнения (1) нормально распределено, распределение ускорения (3) имеет явно выраженную левую асимметрию и естественное усечение при $S = p$. При увеличении количества вычислителей распределения ускорения становятся шире.

3.2 Алгоритмы со случайным выходом и барьером

Параллельные алгоритмы со случайным выходом и барьером (рис. 1в) также достаточно часто применяются для численного решения задач оптимизации, например, при глобальном поиске или при дискретной оптимизации. Они реализуют конкурирующую схему: параллельно выполняющиеся задачи реализуют один и тот же алгоритм над различными данными. По завершении работы всех задач их результаты сравниваются с целью выбора оптимального. Если время работы каждой задачи соответствует (1), то общее время работы параллельного алгоритма с барьером имеет вид:

$$T_p = \max_{i=1,p}\left[T_0 + n_i T_c\right] = T_0 + T_c \cdot \max_{i=1,p}\left[n_i\right], \tag{5}$$

где операция максимума соответствует наличию барьера. Однако, в отличие от (1), время работы последовательного алгоритма будет

$$T_1 = T_0 + T_c \sum_{i=1}^{p} n_i = T_0 + T_c \max_{i=1,p}\left[n_i\right] + T_c \sum_{k=1}^{p-1} n_k \,, \tag{6}$$

Последний член суммы $\vartheta = \sum_{k=1}^{p-1} n_k$ характеризует задачи, завершившиеся до момента достижения барьера.

Определение (5) допускает поиск класса распределений T_p в рамках теории экстремальных значений случайных величин [16]. В частности, если распределение числа итераций n_i нор-

мально, то распределение времени работы на p вычислителях будет асимптотически описываться первым предельным распределением (распределением Гумбеля, или Фишера-Типпета) с плотностью:

$$f_{n_{\max}}(x) = a_p \exp\left[-a_p(x-b_p)\right]\exp\left[-\exp\left[-a_p(x-b_p)\right]\right], \quad (7)$$

где параметры распределения зависят от количества вычислителей

$$a_p = \frac{\sqrt{2\ln(p)}}{\sigma_n},\ b_p = \left[\sqrt{2\ln(p)} - \frac{\ln(\ln(p)) + \ln(4\pi)}{2\sqrt{2\ln(p)}}\right]\sigma_n + m_n. \quad (8)$$

Если определить параллельное ускорение через (5) и (6) как

$$S(p) = \frac{T}{T_p} = \frac{T_0 + n_{\max}T_c + T_c\vartheta}{T_0 + n_{\max}T_c}, \quad (9)$$

и воспользоваться определением (3), то плотность распределения ускорения будет зависеть от случайного параметра ϑ. Как следствие, это дает возможность воспользоваться моделью комбинированного распределения

$$f(S) = \int_0^\infty f_{\max}(n_{\max}(S,\vartheta))\left|n'_{\max}(S,\vartheta)\right|\varphi(\vartheta \mid m_\vartheta, \sigma_\vartheta)d\vartheta, \quad (10)$$

где $n_{\max}(S) = \frac{T_o}{T_c}\frac{1-S+T_c\vartheta/T_0}{S-1}$ получается как обратная функция к (9), а φ - плотность распределения величины ϑ. Она асимпотически может рассматриваться как нормальная с параметрами $m_\vartheta = (p-1)\widetilde{m}_n$, $\sigma_\vartheta = \sqrt{(p-1)}\widetilde{\sigma}_n$. Здесь знак ~ соответствует моменту усеченного нормального распределения с точкой усечения $n_{\max} \equiv (p-1)/p$. В силу приближенности этого предположения выражение (10), в отличие от точного выражения (3), не имеет ограничения при $S = p$, что должно учитываться при анализе. На рис. 3б приведен пример расчета ускорения по (10) при тех же параметрах, что и на рис. 3а.

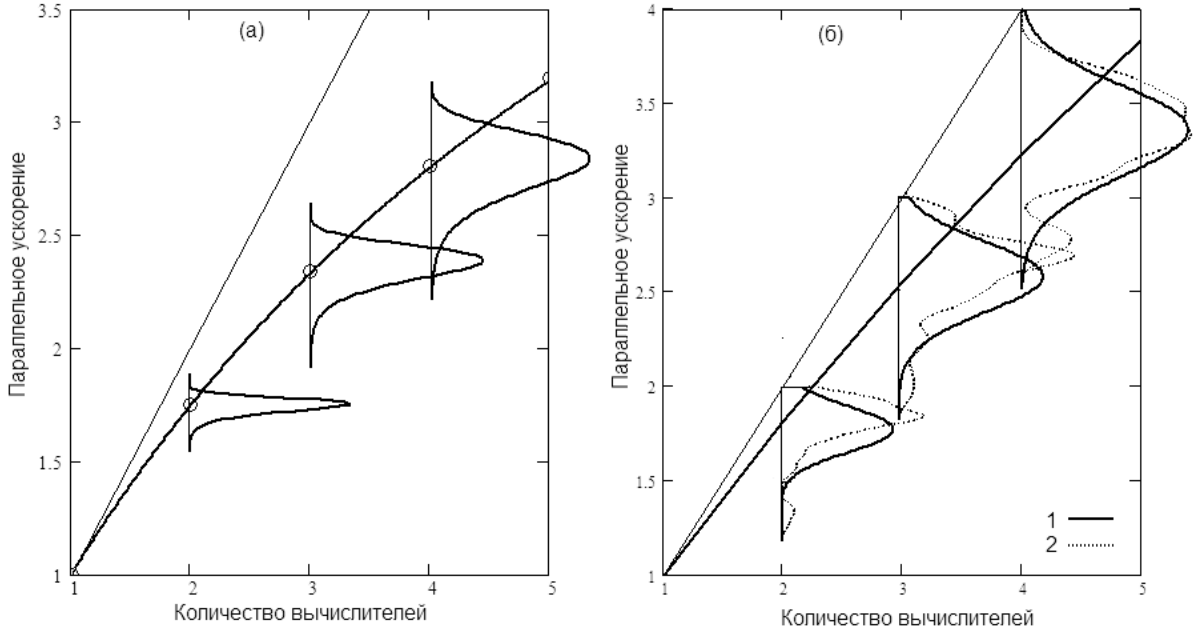

Рис. 3. Распределения параллельного ускорения в зависимости от количества вычислителей: (а) - алгоритм со случайным выходом без барьера; (б) – алгоритм со случайным выходом и барьером. 1 – теоретические значения; 2 – ядерная оценка по данным эксперимента

3.3 Алгоритмы со случайным накоплением

Параллельные алгоритмы со случайным накоплением возникают в задачах потоковой обработки больших объемов данных, где на каждом из M шагов используется алгоритм из раздела 3.1 или 3.2. В силу независимости статистических испытаний, общее время работы таких

алгоритмов будет асимптотически нормальным с параметрами $m_T = Mm_{T_i}, \sigma_T = \sqrt{M}\sigma_{T_i}$. В частности, если на каждом шаге используется алгоритм со случайным выходом без барьера, то распределение параллельного ускорения получается в соответствии с (1-4) путем замены переменных:

$$\widetilde{T}_0 = MT_0, \widetilde{m}_n = Mm_n, \widetilde{\sigma}_n = \sqrt{M}\sigma_n, \tag{11}$$

т.е. масштабных преобразований исходного распределения. При увеличении M среднее значение ускорения остается прежним, а размах распределения убывает пропорционально $1/\sqrt{M}$.

4. Метрологический анализ процедуры измерения параллельной производительности

Параллельные алгоритмы со случайным накоплением являются характерным примером алгоритмов, которые при значительных M можно рассматривать как детерминированные. Н практике эта величина ограничивается допустимой точностью определения параллельного ускорения по данным измерений. Это связано с тем, что даже для детерминированных алгоритмов на точность измерения времени работы влияет ряд дополнительных источников стохастизации [17]. К таким источникам могут быть отнесены:

- недетерминированность работы библиотечных процедур, используемых в процессе работы программного средства;
- ряд оптимизаций, вносимых компилятором (которые, в некоторых случаях, могут оказывать косвенное влияние при работе с методами операционной системы);
- шум, вносимый работой операционной системы в управления распределением вычислительных ресурсов между процессами;
- аппаратный шум вычислительной системы.

В качестве примера в таблице 1 приведены экспериментальные оценки характеристик относительной ошибки времени работы и ускорения для детерминированной[1] реализации алгоритма случайного поиска, см. раздел 3.1, в зависимости от приоритета рабочих потоков в операционной системе. Видно, что даже при низком приоритете случайная ошибка определения ускорения составляет не более 2%.

Таблица 1. Относительная ошибка (%) времени выполнения и параллельного ускорения в зависимости от приоритета рабочих потоков в операционной системе (двухъядерный процессор).

Приоритет	Низкий	Нормальный	Высокий	Реальное время
T_1	0,6	0,3	0,05	0,07
T_2	1,5	1,0	0,26	0,32
S	1,6	0,9	0,27	0,35

Приоритет реального времени соответствует максимальному допустимому приоритету. При этом процесс превалирует над остальными задачами операционной системы, включая ряд системных процессов.

В том случае, если коэффициент вариации параллельного ускорения в рамках описанного выше подхода сопоставим со случайной ошибкой измерений, то алгоритм допустимо рассматривать как детерминированный. Для алгоритма, описанного в разделе 3.1, эта характеристика может быть получена приближенно (без интегрирования (3)), используя метод статистической линеаризации. Коэффициент вариации параллельного ускорения (как относительная ошибка) имеет вид

[1] 100 испытаний при одной и той же реализации случайной последовательности.

$$\delta_S = \left[\frac{1}{\frac{T_{посл}}{T_{пар}} + 1} - \frac{1}{\frac{T_{посл}\, p}{T_{пар}} + 1} \right] \delta_n , \qquad (12)$$

где $T_{посл}, T_{пар}$ - доля последовательного и параллельного кода в программе соответственно, а $\delta_n = \sigma_n / m_n$ - коэффициент вариации числа итераций. Выражение в квадратных скобках можно рассматривать как безразмерный коэффициент перехода между среднеквадратической относительной ошибкой измерения ускорения и коэффициентом вариации числа итераций. На рис. 4 приведен график этого коэффициента (в %). Эта диаграмма может быть использована как для экспресс-оценки изменчивости значений параллельного ускорения, для оценивания целесообразности описания производительности стохастической моделью вида (1-4) или (5-10). Так, например, при коэффициенте вариации числа итераций равном $\delta_n = 0.25$, числе исполнителей $p = 7$ и отношении последовательного кода к параллельному $\frac{T_{посл}}{T_{пар}} = 0.17$ коэффициент вариации параллельного ускорения составляет $\delta_S \approx 0.1$, что примерно соответствует изолинии графика для 40%.

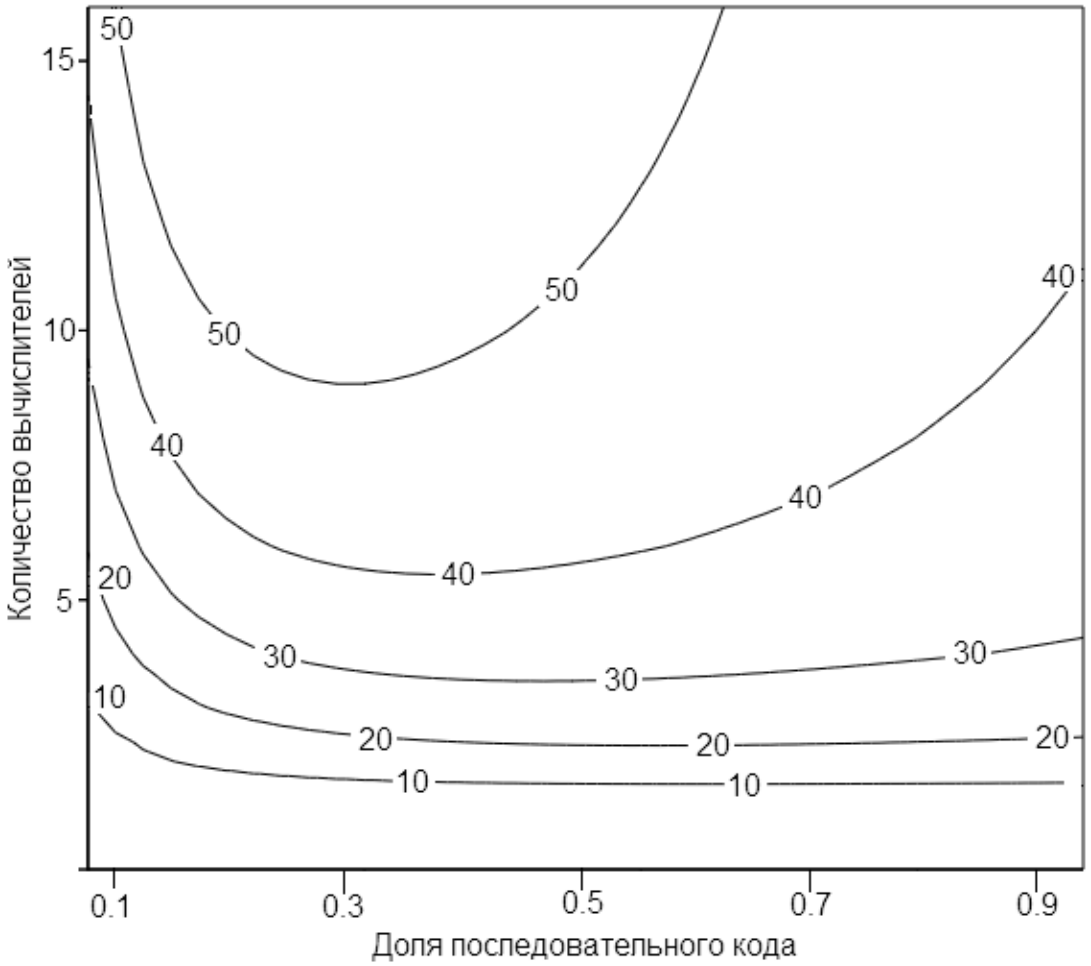

Рис. 4. Коэффициент перехода в (12) между относительной ошибкой определения параллельного ускорения в модели (1-4) и коэффициентом вариации числа итераций метода.

5. Выводы

Время выполнения, параллельное ускорение и эффективность параллельных стохастических алгоритмов являются случайными величинами. Их распределения могут быть выражены через распределения случайных параметров самого алгоритма в рамках модели детерминированной функции случайного аргумента. При этом класс и параметры распределений параллельного ускорения существенно отличается от распределений времени работы и зависят от доли параллельного кода в программе и количества вычислителей. Целесообразность применения вероятностного подхода к описанию параллельного ускорения определяется мерой аппаратного и программного шума вычислительной системы, влияющего на точность определения временных характеристик.

Литература

1. Kenneth Tan C.J. On Parallel Pseudo-Random Number Generation // International Conference on Computational Science (1) — 2001. — pp. 589-596.
2. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем — М.: Мир, 1991.
3. Foster J. Designing and Building Parallel Programs — Adison-Wesley, 1995: [http://www-unix.mcs.anl.gov/dbpp]
4. Gerbessiotis A.V. Architecture independent parallel algorithm design: theory vs practice // Future Generation Computer Systems, 18, 2002, pp. 573-593.
5. Uresin A., Dubois M. Effect of asynchronism in the convergence rate of iterative algorithms // J. Parallel and Distributed Computing, 34, 1996, pp. 66-81.
6. Casanova H. Stochastic models for performance analysis of iterative algorithms in distributed environment // Ph.D. thesis Univ. Tennessee, Knoxville, 1998, 200 p.
7. Ларченко А.В., Дунаев А.В., Бухановский А.В. Анализ и моделирование производительности параллельных стохастических алгоритмов, адаптированных к особенностям многоядерных вычислительных архитектур // Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ: Труды Всероссийской научной конференции (24-29 сентября 2007 г., г. Новороссийск) — М.: Издательство МГУ, 2007. — с. 156
8. Бурова И.Г., Демьянович Ю.К. Лекции по параллельным вычислениям — СПб., 2003. — 132 с.
9. Карпенко А.П., Федоруг В.Г., Федорук Е.В. Балансировка загрузки многопроцессорной вычислительной системы при распараллеливании одного класса вычислительных задач // Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ: Труды Всероссийской научной конференции (24-29 сентября 2007 г., г. Новороссийск) — М.: Издательство МГУ, 2007. — с. 48-52
10. Birattari M., Dorigo M. How to assess and report the performance of a stochastic algorithm on a benchmark problem: mean or best result on a number of runs? // Optimization Letters, Volume 1, Number 3, Springer, 2007. — pp. 309-311
11. Fathi B., Liu B., Alexandrov V. Mixed Monte Carlo Parallel Algorithms for Matrix Computation // ICCS 2002, LNCS 2330 — Springer-Verlag Berlin Heidelberg, 2002. — pp. 609−618.
12. Лившиц Н.А., Пугачев В.Н. Вероятностный анализ систем автоматического управления. Т.1 — М., Советское радио, 1963. — 896 с.
13. Fortune S., Willie J. Parallelism in Random Access Machines // Proceedings of the 10th Annual Symposium on Theory of Computing — 1978. — pp. 114-118
14. Растригин Л.А. Адаптация сложных систем — Рига: Зинатне, 1981. — 375 с.
15. Boukhanovsky A.V., Lopatoukhin L.J., Guedes Soares C. Spectral wave climate of the North Sea — Applied Ocean Research, 2007.
16. Лидбеттер М., Линдгрен Г., Ротсен Х., Экстремумы случайных последовательностей и рядов — М., Мир, 1989. — 392 с.
17. Касперски К. Техника оптимизации программ. Эффективное использование памяти — СПб.: БХВ-Петербург, 2003. — 464 с.