Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки. 2014. № 3 (36). С. 183–195
ISSN: 2310-7081 (online), 1991-8615 (print) doi: http://dx.doi.org/10.14498/vsgtu1341

УДК 519.681.2

РАЗРАБОТКА И ПРИМЕНЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ МОДЕЛИ ТИПОВЫХ РЕШЕНИЙ. ПРИМЕР ИСПОЛЬЗОВАНИЯ «ПОРТФЕЛЯ ЗАДАЧ» ДЛЯ ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ HRBF

В. Г. Литвинов

Самарский государственный аэрокосмический университет им. ак. С. П. Королёва (национальный исследовательский университет),
443086, Россия, Самара, Московское ш., 34.

Аннотация

Рассматривается проблема отображения задач вычислительной математики на архитектуру вычислительных систем и метод ее решения. В качестве метода предлагается разработанный подход реализации параллельных паттернов, заключающийся в построении точной модели вычислительного процесса в виде графа и его динамической интерпретации. Для описания модели вычислительного процесса используется темпоральная логика Лампорта. На примере типового решения «портфель задач» показано, каким образом при помощи графической нотации выполняется связь абстрактной модели исполнения с операциями типового решения. В терминах этих операций определяется конкретный алгоритм обучения нейронной сети типа HRBF (Hyper-Radial Basis Function), который распараллеливается единообразно для различных типовых решений и зависит только от программно-аппаратной платформы целевой системы исполнения. Приведены вычислительные эксперименты, которые подтверждают, что применение типовых решений не снижает производительность результирующих программ и пригодно в случаях, когда специалисту предметной области требуется построить параллельный алгоритм и выполнить вычислительный эксперимент на высокопроизводительной технике.

Ключевые слова: кластер, суперкомпьютинг, язык Templet, типовое решение, портфель задач, скелетон, модель вычислений, нейронная сеть HRBF.

doi: http://dx.doi.org/10.14498/vsgtu1341

Введение. Эпоха увеличения тактовых частот вычислительных ядер процессоров закончена, её конец ознаменовал начало эпохи многоядерных процессоров и распределенных вычислительных систем [1]. На текущий момент огромное количество задач математического моделирования требует разработки параллельных (распределенных) программ. Это требование связано с необходимостью увеличения размерности и сокращения времени решения

Образец для цитирования: Литвинов В. Г. Разработка и применение вычислительной модели типовых решений. Пример использования «портфеля задач» для обучения нейронной сети HRBF // *Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки*, 2014. № 3 (36). С. 183–195. doi: 10.14498/vsgtu1341.

Сведения об авторе: *Владимир Геннадьевич Литвинов* (litvinov_vg@ssau.ru), ассистент, каф. информационных систем и технологий.

задачи. Инженерная сложность такой разработки во много раз выше, чем стандартный последовательный подход и переход от численного метода к реализации программы для высокопроизводительных вычислений. На данный момент это остается самостоятельной научной и инженерной проблемой. Данная проблема выделена академиком Г. И. Марчуком как фундаментальное научное направление, называемое проблемой отображения [2]. Одним из методов её решения является повторное использование имеющихся паттернов (шаблонов или типовых решений), что отражено в работе Э. Гамма, Р. Хелма, Р. Джонсона и Дж. Влиссидеса [3]. В области типовых решений для параллельного программирования известны работы отечественных и зарубежных ученых: М. И. Коула [4], С. Макдональда [5], Д. Шмидта [6], П. К. Берзигиярова [7], В. Э. Малышкина [8] и других. Описано около двух десятков типовых решений в области параллельного программирования. Однако задача разработки моделей типовых решений, удобных для понимания специалистами прикладных областей и для конструирования программ на их основе, сохраняет актуальность.

1. Метод. В работе предлагается подход к реализации параллельных паттернов, близкий к языкам координации [9, 10]. Модели выполнения данных языков отражают только аспекты, относящиеся к синхронизации и обмену данными в параллельных алгоритмах. Таким образом, можно строить скелеты типовых решений, которые пользователь затем дополняет последовательным кодом, относящимся к конкретной прикладной задаче. Строится точная модель исполнения в виде графа и его динамической интерпретации. Далее на примере типового решения «портфель задач» [11] показано, каким образом при помощи графической нотации выполняется связь абстрактной модели исполнения с операциями типового решения. Наконец, в терминах этих операций определяется конкретный алгоритм обучения нейронной сети типа HRBF (Hyper-Radial Basis Function). Распараллеливание алгоритма благодаря введённой модели выполнения производится единообразно для различных типовых решений и зависит только от программно-аппаратной платформы целевой системы исполнения. Предлагаемый подход близок к реализованному в библиотеке `tbb::gragh` пакета `Intel Threading Building Blocks` [12], однако он не ограничен только языком `C++`. По сравнению с техниками инкрементного распараллеливания (`Cilk` [13], `OpenMP` [14], `DVM` [15]), в которых последовательный код пользователя дополняется распараллеливающими инструкциями, в данном подходе код каркаса (последовательный, но с известным методом распараллеливания) дополняется пользовательским кодом. Проводимые в дальнейшем вычислительные эксперименты призваны подтвердить, что это не снижает производительность результирующих программ и пригодно для применения в случаях, когда специалисту предметной области требуется построить параллельный алгоритм и выполнить вычислительный эксперимент на высокопроизводительной технике.

2. Построение имитационной модели. Для описания типовых решений использовалась темпоральная логика Лампорта [16, 17]. С её помощью структуру алгоритмов можно представить ориентированным графом вида

$$G_M \triangleq (P, C), \quad P \subset N, \quad C \subset N \times N,$$

в котором вершины P представляют собой процессы, а дуги C — каналы, по

которым передаются сообщения между процессами. Состояние вычислительного процесса — значения множества переменных, связанных с процессами и каналами:

$$f \triangleq (i \in P\colon a_i; (i,j) \in C\colon e_{i,j}, d_{i,j})\,,$$

где $a_i \in \{1,0\}$, $b_{i,j} \in \{1,0\}$, $d_{i,j} \in \{1,0\}$.

Логические переменные a_i обозначают пассивное 0 или активное 1 состояние процесса i; $e_{i,j}$ — пассивное 0 или активное 1 состояние канала (i,j); $d_{i,j}$ — направление передачи сообщений по двунаправленным каналам ($1 - i \to j$; $0 - j \to i$).

В начальном состоянии некоторые каналы находятся в активном состоянии с передачей сообщений в направлении $i \to j$, все процессы находятся в пассивном состоянии:

$$I \triangleq \forall i \in P\colon \neg a_i \wedge \forall\,(i,j) \in C\colon\ (e_{i,j} \wedge d_{i,j}) \vee (\neg e_{i,j} \wedge \neg d_{i,j})\,.$$

Действия, выполняемые каналами, описываются формулой

$$A_c\,(i,j) \triangleq \left\{e_{i,j} \wedge \neg e'_{i,j} \wedge \left(\left(d_{i,j} \wedge d'_{i,j}\right) \vee \left(\neg d_{i,j} \wedge \neg d'_{i,j}\right)\right)\right\} \vee \\ \vee \left\{\neg e_{i,j} \wedge e'_{i,j} \wedge \left(\left(d_{i,j} \wedge \neg d'_{i,j}\right) \vee \left(\neg d_{i,j} \wedge d'_{i,j}\right)\right)\right\}. \quad (1)$$

Действия описывают соседние в истории вычислительного процесса состояния, переменные следующего состояния обозначены апострофом. Согласно формуле (1), состояние канала периодически меняется с активного на пассивное, а направление передачи данных — с прямого на обратное. Действия, выполняемые процессами, описываются формулой

$$A_p\,(i) \triangleq \left\{\neg a_i \wedge a'_i \wedge \left(\exists!\, j\colon e_{i,j} \wedge \neg e'_{i,j} \wedge \neg d'_{i,j}\right) \wedge \neg\left(\exists j\colon e_{j,i} \wedge \neg e'_{j,i} \wedge d'_{j,i}\right)\right\} \vee \\ \vee \left\{\neg a_i \wedge a'_i \wedge \left(\exists!\, j\colon e_{j,i} \wedge \neg e'_{j,i} \wedge d'_{j,i}\right) \wedge \neg\left(\exists j\colon e_{i,j} \wedge \neg e'_{i,j} \wedge \neg d'_{i,j}\right)\right\} \vee \\ \vee \left\{a_i \wedge \neg a'_i \wedge \bigvee_{k\colon (i,k)\in C} \left(\neg e_{i,k} \wedge e'_{i,k} \wedge d'_{i,k}\right)\right\} \vee \\ \vee \left\{a_i \wedge \neg a'_i \wedge \bigvee_{k\colon (k,i)\in C} \left(\neg e_{k,i} \wedge e'_{k,i} \wedge d'_{k,i}\right)\right\} \vee \left\{a_i \wedge \neg a'_i\right\}. \quad (2)$$

Процесс становится активным, когда из смежного с ним канала поступает сообщение. При этом такой канал сам переходит в пассивное состояние. Среди нескольких каналов, готовых передать сообщение в процесс, выбирается произвольный канал. Когда процесс завершит работу и перейдет в пассивное состояние, по одному или нескольким каналам от него могут передаваться сообщения в другие процессы.

С учетом выражений (1) и (2) действия, выполняемые моделируемым вычислительным процессом, можно представить в виде

$$N \triangleq \bigvee_{(i,j)\in C} A_c\,(i,j) \vee \bigvee_{i\in P} A_p\,(i)\,.$$

Способ планирования вычислительного процесса характеризуется выражением

$$F \triangleq \left(\forall i \in P : WF_f\left(a_i \wedge \neg a_i'\right)\right) \wedge \left(\forall\,(i,j) \in C : WF_f\left(e_{i,j} \wedge \neg e_{i,j}'\right)\right),$$

подразумевающим справедливую в слабом смысле стратегию планирования. Окончательно получаем модель вычислительных процессов для представления типовых решений:

$$\Phi_M \triangleq I \wedge \square\,[N]_f \wedge F. \tag{3}$$

По построению вычислительного процесса порядок смены состояний, удовлетворяющий формуле (3), легко воспроизвести дискретно-событийной имитационной моделью. Для этого организуется очередь каналов, находящихся в активном состоянии. На каждом шаге моделирования случайным образом выбирается активный канал. Далее он переводится в неактивное состояние, выполняется действие процесса и отправка сообщений из данного процесса.

Для представления типового решения далее требуется определить сообщения, передаваемые в каналах, и методы процессов, запускаемые при их обработке. Это реализуется использованием графической нотации `Templet` [18–20]. Так как типовыми схемами решения многих переборных и оптимизационных задач являются схемы «управляющий-рабочие», далее рассматривается модель одной из общих схем управления вычислениями из данного класса: вычисление с «портфелем задач». Это схема с активными рабочими, реализующая управление `pull`-типа. Здесь иллюстрируется описание более сложного протокола взаимодействия процессов с несколькими типами сообщений (по сравнению с исходными (1) протоколами «запрос-ответ»).

Подробное описание реализации шаблона приведено в работе [21], для дальнейшей ясности приведем фрагмент диаграммы управляющего процесса с именем `Master` (рис. 1), относящийся к порту $\mathtt{p_N}$, где `N` — номер порта. Методы процесса `get`, `put`, `if_task` и `start` реализуют специфические для решаемой задачи операции: `get` — извлечение очередной задачи; `put` — сохранение результата; `if_task` — проверку наличия задачи для обработки. Метод `start` может использоваться для инициализации. Методы `if_active`, $\mathtt{wait_N}$, $\mathtt{stop_N}$, $\mathtt{waiting_N}$ являются универсальными для произвольного численного метода. Метод `if_active` возвращает истинное значение, если имеются активные рабочие процессы. То есть имеются каналы, не находящиеся в состоянии «доставлено сообщение `start` или `result`», что в программной реализации определяется простым подсчетом при поступлении и отправке сообщений. Метод $\mathtt{wait_N}$ устанавливает признак, что канал находится в состоянии ожидания появления заданий, метод $\mathtt{waiting_N}$ возвращает значение данного признака. Метод $\mathtt{stop_N}$ используется при отправке сообщения `stop` рабочему для начала очистки.

Роль рабочих процессов заключается в периодическом получении задания от управляющего процесса в сообщении `task` и отправке результата его обработки `result`. Обработка выполняется в методе `do`. Цикл обработки начинается по инициативе рабочего с отправки сообщения `start` и заканчивается при получении от управляющего процесса сообщения `stop`.

Остановка вычислений и очистка начинаются, если нет задач и нет активных рабочих процессов. Когда сформирована и отправлена очередная задача, происходит поиск ожидающего рабочего процесса. Если такой процесс

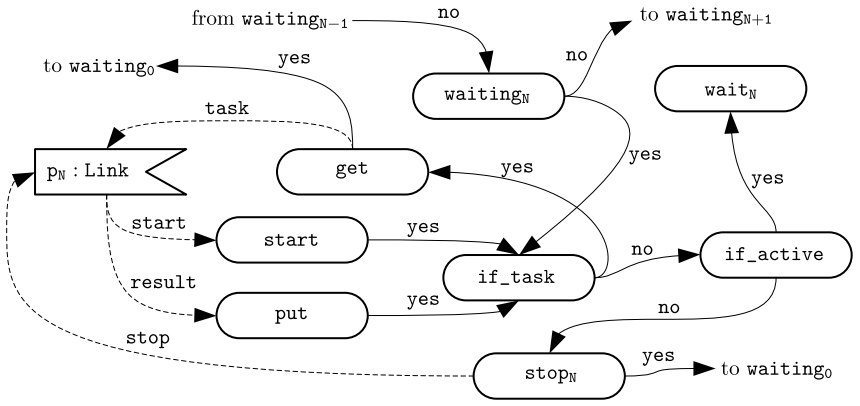

Рис. 1. Фрагмент диаграммы управляющего процесса `Master`
[Figure 1. Fragment of the `Master` control process diagram]

найдется и есть еще одна задача для обработки, то выполняется ее формирование и отправка рабочему. В противном случае управляющий процесс либо заканчивает обработку сообщения и ждет ответа от рабочих, либо начинает остановку вычислений и очистку.

Для типового решения «портфель задач» разработаны среды многопоточного исполнения на основе `Windows API`, `POSIX` и `MPI`, позволяющие проводить высокопроизводительные вычисления на системах с общей и распределённой памятью.

3. Реализация алгоритма обучения. Параллельный алгоритм обучения был реализован для HRBF нейронной сети [22,23] на языке `C++` с использованием библиотеки `CLAPACK`. Он основан на использовании метода грубой силы и выполняет обучение и подбор количества нейронов скрытого слоя [24]. Алгоритм заключается в итеративном добавлении различающихся случайным положением центра нейронов в идентичные копии нейронных сетей. На каждой итерации производится выбор лучшей сети (по критерию достижения минимального значения целевой функцией) и её клонирование. Применение типового решения «портфель задач», фрагмент которого показан на рис. 1, состоит в определении его методов `get`, `put`, `if_task`. В «портфеле задач» для управления вычислениями хранятся:

1) $\mathtt{K}$ — текущее количество нейронов в скрытом слое;
2) $\mathtt{K}_{\mathtt{max}}$ — максимальное число нейронов в скрытом слое;
3) $\mathtt{W}$ — число решенных задач;
4) $\mathtt{W}_{\mathtt{max}}$ — максимальное количество задач (попыток добавления нейрона);
5) $\mathtt{E}_{\mathtt{end}}$ — значение, при достижении целевой функцией которого процесс обучения останавливается;
6) $\Psi_{\mathtt{K}}$ — коэффициенты обученной нейронной сети с $\mathtt{K}$ нейронами (включают вектора центров, матрицы диаметров и вектор весов выходного слоя);
7) $\Psi_{\mathtt{K}+1}$ — коэффициенты обученной нейронной сети с $\mathtt{K}+1$ нейронами;
8) $\mathtt{E}_{\mathtt{K}+1}$ — значение целевой функции сети Ψ.

Задача определяется коэффициентами $\Psi'_{\mathtt{K}}$ и положением центра добавляемого нейрона ψ'. Результат её вычисления — коэффициенты $\Psi'_{\mathtt{K}+1}$ и значение

ошибки $\mathtt{E}'_{\mathtt{K}+1}$. Как и в последовательном случае, при решении задачи производится уточнение коэффициентов сети Ψ'_{n+1}-гибридным алгоритмом. В начальном состоянии (`start`) определяются постоянные алгоритма $\mathtt{K}_{\max}$, $\mathtt{W}_{\max}$, $\mathtt{E}_{\mathtt{end}}$; переменные $\mathtt{W} := 0$, $\mathtt{K} := 2$, $\Psi_{\mathtt{K}} := \Psi_2$ (параметры сети с двумя нейронами), $\Psi_{\mathtt{K}+1}$ — не определены, $\mathtt{E}_{\mathtt{K}+1} \gg \mathtt{E}_{\mathtt{end}}$.

`Проверка наличия задачи (if_task)`. Возможна выборка очередной задачи на обработку, если одновременно выполняются следующие условия: не достигнута заданная ошибка обучения $\mathtt{E}_{\mathtt{K}+1} < \mathtt{E}_{\mathtt{end}}$; остались невыданные задачи $\mathtt{W} < \mathtt{W}_{\max}$.

`Алгоритм извлечения задачи (get)`. Присвоить $\Psi'_{\mathtt{K}} := \Psi_{\mathtt{K}}$; определить случайное положение центра нейрона ψ'. Увеличить счетчик запущенных задач $\mathtt{W} := \mathtt{W} + 1$.

`Алгоритм помещения результата вычисления задачи (put)`. Если $\mathtt{E}'_{\mathtt{K}+1} < \mathtt{E}_{\mathtt{K}+1}$, то определить $\mathtt{E}_{\mathtt{K}+1} := \mathtt{E}'_{\mathtt{K}+1}$ и $\Psi_{\mathtt{K}+1} := \Psi'_{\mathtt{K}+1}$. Если $\mathtt{W} = \mathtt{W}_{\max}$, то закончено обучение слоя. В этом случае, если $\mathtt{K} < \mathtt{K}_{\max}$, переходим к следующему слою: $\mathtt{K} := \mathtt{K} + 1$; $\mathtt{W} := 0$.

Алгоритм завершается, когда будут обучены все сети с числом нейронов до $\mathtt{K}_{\max}$ включительно либо будет достигнуто заданное значение ошибки обучения $\mathtt{E}_{\mathtt{K}+1} \leqslant \mathtt{E}_{\mathtt{end}}$.

4. Численный эксперимент. Эксперимент по обучению нейронной сети параллельным алгоритмом на основе типового решения «портфель задач» был проведен на кластере Сергей Королёв (http://hpc.ssau.ru). Запуск задач производился на узлах 3-х типов с 3-мя различными типами процессоров соответственно:

1) узлы с процессорами типа Intel®Xeon®X5560 (8M Cache, 2.80 GHz, 6.40 GT/s Intel®QPI), имеющими 8 вычислительных ядер;
2) узлы с процессорами типа Intel®Xeon®X5670 (12M Cache, 2.93 GHz, 6.40 GT/s Intel®QPI), имеющими 12 вычислительных ядер;
3) узлы с процессорами типа Intel®Xeon®E5-2665 (20M Cache, 2.40 GHz, 8.00 GT/s Intel®QPI), имеющими 16 вычислительных ядер.

В рамках данного эксперимента на быстродействие влияло количество доступных вычислительных ядер на узле, остальные параметры ощутимого влияния не оказывали.

Условия эксперимента состояли в следующем.

1. Варьировалось максимальное количество рабочих процессов P «портфеля задач» (объявлено в конкретном эксперименте).
2. Предельное количество нейронов $\mathtt{K}_{\max}$ во всех экспериментах равнялось 100.
3. Количество попыток добавления нейронов $\mathtt{W}_{\max}$ во всех экспериментах равнялось 100.

Для осуществления сравнительного анализа были произведены замеры времени исполнения последовательного и параллельного алгоритмов. Для минимизации воздействия сторонней динамической нагрузки процессоров на измеряемое время производилось по 20 запусков. При этом полученное время исполнения параллельного алгоритма $t_{\mathrm{pr}_i}(P)$ было усреднено:

$$t_{\mathrm{pr}}(P) = \frac{1}{20}\sum_{i=1}^{20} t_{\mathrm{pr}_i}(P). \tag{4}$$

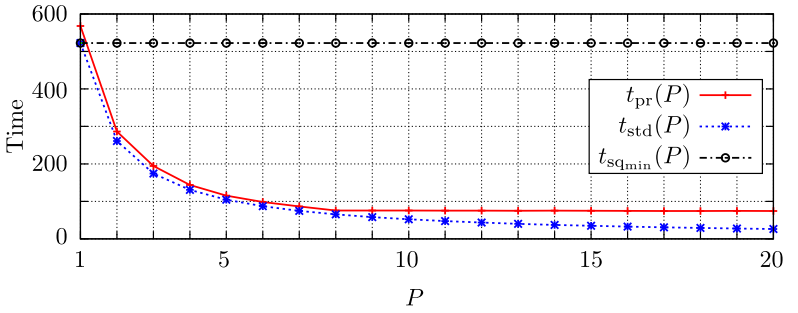

Рис. 2. Зависимость времени исполнения от количества рабочих процессов (8-ядер) [Figure 2. Dependence of the performance on the number of worker processes (8 cores)]

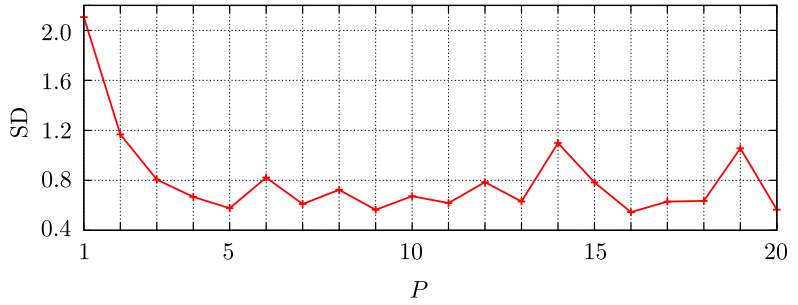

Рис. 3. Зависимость СКО от количества рабочих процессов (8-ядер) [Figure 3. Dependence of the standard deviation (SD) on the number of worker processes (8 cores)]

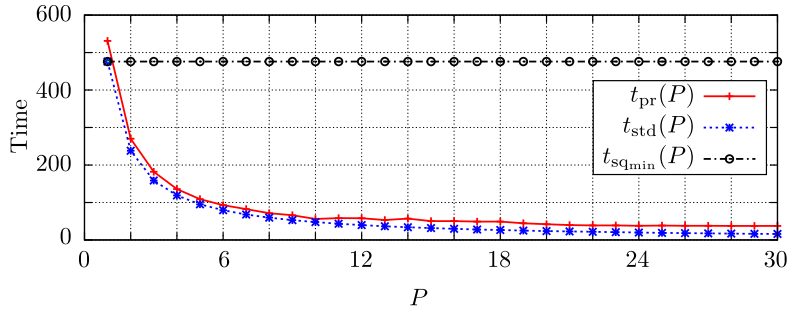

Рис. 4. Зависимость времени исполнения от количества рабочих процессов (12-ядер) [Figure 4. Dependence of the performance on the number of worker processes (12 cores)]

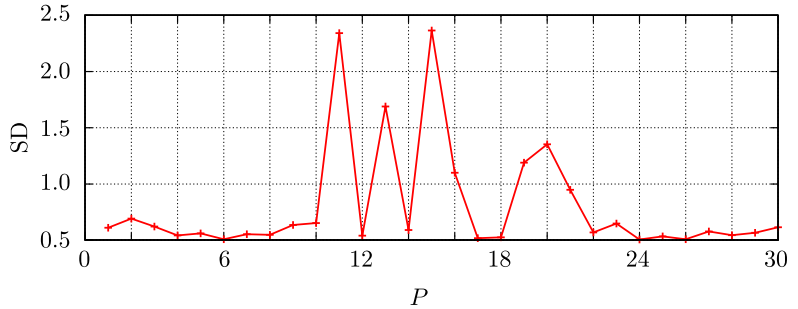

Рис. 5. Зависимость СКО от количества рабочих процессов (12-ядер) [Figure 5. Dependence of the standard deviation (SD) on the number of worker processes (12 cores)]

Для времени исполнения последовательного алгоритма t_{sq_i} было найдено минимальное значение $t_{\mathrm{sq}_{\min}} = \min\left(t_{\mathrm{sq}_1}, t_{\mathrm{sq}_2}, \ldots, t_{\mathrm{sq}_{20}}\right)$, и на его основе рассчитано эталонное время исполнения параллельного алгоритма

$$t_{\mathrm{std}}(P) = t_{\mathrm{sq}_{\min}}/P. \tag{5}$$

Среднеквадратическое отклонение (СКО) рассчитывалось для $t_{\mathrm{pr}_i}(P)$.

На узлах с процессорами Intel®Xeon®X5560 максимальное количество рабочих процессов P изменялось в интервале $[1, 2, \ldots, 20]$. Данные, полученные при проведении эксперимента, приведены на рис. 2, соответствующее среднеквадратическое отклонение изображено на рис. 3.

На узлах с процессорами Intel®Xeon®X5570 максимальное количество рабочих процессов P изменялось в интервале $[1, 2, \ldots, 30]$. Данные, полученные при проведении эксперимента, приведены на рис. 4, соответствующее среднеквадратическое отклонение изображено на рис. 5.

На узлах с процессорами Intel®Xeon®E5-2665 максимальное количество рабочих процессов P изменялось в интервале $[1, 2, \ldots, 40]$. Данные, полученные при проведении эксперимента, приведены на рис. 6, соответствующее среднеквадратическое отклонение изображено на рис. 7.

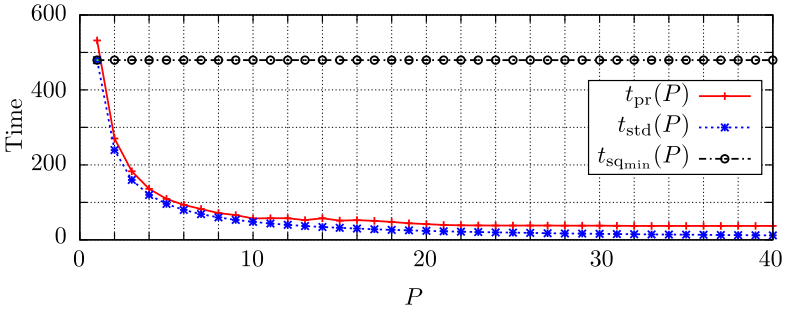

Рис. 6. Зависимость времени исполнения от количества рабочих процессов (16-ядер) [Figure 6. Dependence of the performance on the number of worker processes (16 cores)]

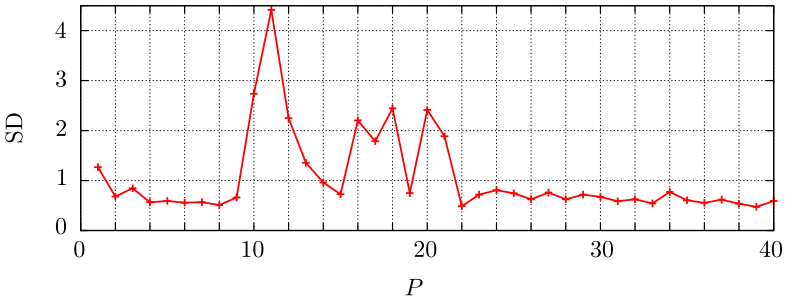

Рис. 7. Зависимость СКО от количества рабочих процессов (16-ядер) [Figure 7. Dependence of the standard deviation (SD) on the number of worker processes (16 cores)]

Заключение. Анализируя графики, изображенные на рис. 2–7, можно сделать вывод, что усредненное время исполнения параллельного алгоритма (4), разработанного с использованием типового решения «портфель задач», лишь немного больше эталонного времени исполнения последовательного алгоритма (5).

Также можно заметить, что при достижении количеством рабочих процессов значения, приблизительно равного удвоенному количеству физических вычислительных ядер, уменьшение времени исполнения параллельного алгоритма практически прекращается. Это объясняется наличием технологии Hyper-threading в процессорах (удвоенное количество вычислительных потоков по отношению к количеству ядер). Если продолжать увеличивать количество рабочих процессов, то в связи с ростом накладных расходов время исполнения начнёт возрастать. Следовательно, число рабочих процессов, равное количеству вычислительных потоков, можно считать оптимальным и рекомендовать для использования в экспериментах с применением типового решения «портфель задач».

Реализация параллельного алгоритма с использованием данного типового решения значительно снижает трудоёмкость построения параллельного численного метода и сокращает объем кода, введенного ручным способом [25]; избавляет программиста от необходимости проектирования параллельной программы и, следовательно, уменьшает вероятность ошибок.

Еще одно достоинство применения данного метода заключается в платформонезависимости. Для переноса программы с одной операционной системы на другую или с платформы, имеющей общую память, на платформу с распределенной памятью достаточно лишь заменить библиотеку времени исполнения, при этом программа будет компилироваться без каких-либо доработок (при условии, что она написана по стандартам `ANSI C`/`C++`).

Разработанное в настоящей работе типовое решение «портфель задач» включено в состав веб-сервиса автоматизации параллельных вычислений на суперкомпьютере Сергей Королёв Самарского государственного аэрокосмического университета, развернутого по адресу `http://templet.ssau.ru` [26].

Благодарности. Автор выражает благодарность Самарскому государственному аэрокосмическому университету за поддержку этого исследования. Работа выполнена при государственной поддержке Министерства образования и науки РФ в рамках реализации мероприятий Программы повышения конкурентоспособности СГАУ среди ведущих мировых научно-образовательных центров на 2013–2020 годы.

Отдельную благодарность автор выражает своему научному руководителю Сергею Владимировичу Востокину — за советы и неоценимую помощь в ходе работы.

ORCID
Litvinov, Vladimir: `http://orcid.org/0000-0002-7082-9805`

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Sutter H. The free lunch is over: A fundamental turn toward concurrency in software // *Dr. Dobb's journal*, 2005. vol. 30, no. 3. pp. 202–210.
2. Марчук Г. И., Котов В. Е. Проблемы вычислительной техники и фундаментальные исследования // *Автом. и вычисл. техн.*, 1979. № 2. С. 3–14.
3. Gamma E., Helm R., Johnson R., Vlissides J. *Design patterns: elements of reusable object-oriented software*. Westford: Addison-Wesley, 1995. 395 pp.
4. Cole M. I. *Algorithmic skeletons: structured management of parallel computation*. Massachusetts, Cambridge, Boston: MIT Press, 1989. v+132 pp.
5. MacDonald S., Szafron D., Schaeffer J., Bromling S. Generating parallel program frameworks from parallel design patterns / *Euro-Par 2000 Parallel Processing* / Lecture Notes in Computer Science, 1900. Berlin, Heidelberg: Springer, 2000. pp. 95–104. doi: 10.1007/3-540-44520-X_13.

6. Schmidt D. C., Huston S. D. *C++ Network Programming.* vol. I: Mastering Complexity with ACE and Patterns: FT Press, 2001. 336 pp.
7. Берзигияров П. К. Программирование на типовых алгоритмических структурах с массивным параллелизмом // *Вычислительные методы и программирование*, 2001. Т. 2, № 2. С. 1–16.
8. Kraeva M. A., Malyshkin V. E. Assembly technology for parallel realization of numerical models on mimd-multicomputers // *Future Generation Computer Systems*, 2001. vol. 17, no. 6. pp. 755–765. doi: 10.1016/s0167-739x(00)00058-3.
9. Botorog G. H., Kuchen H. Skil: An imperative language with algorithmic skeletons for efficient distributed programming / *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing,* HPDC-96, 1996. 243–252 pp.. doi: 10.1109/hpdc.1996.546194
10. Dorta A. J., González J. A., Rodríguez C., De Sande F. Llc: A parallel skeletal language // *Parallel Processing Letters*, 2003. vol. 13, no. 03. pp. 437–448. doi: 10.1142/s0129626403001409.
11. Iosup A., Sonmez O., Anoep S., Epema D. The performance of bags-of-tasks in large-scale distributed systems / *Proceedings of the 17th International Symposium on High Performance Distributed Computing 2008,* HPDC'08, 2008. pp. 97–108. doi: 10.1145/1383422.1383435.
12. Reinders J. *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism.* Beijing, Cambridg, etc.: O'Reilly Media, 2007. 336 pp.
13. Blumofe R. D., Joerg C. F., Kuszmaul B. C., Leiserson C. E., Randall K. H., Zhou Y. Cilk: An Efficient Multithreaded Runtime System // *Journal of Parallel and Distributed Computing*, 1996. vol. 37, no. 1. pp. 55–69. doi: 10.1006/jpdc.1996.0107.
14. OpenMP: an industry standard API for shared-memory programming // *IEEE Computational Science and Engineering*, 1998. vol. 5, no. 1. pp. 46–55. doi: 10.1109/99.660313.
15. Konovalov N. A., Krukov V. A., Mihailov S. N., Pogrebtsov A. A. Fortran dvm — a language for portable parallel program development / *Proceedings of Software For Multiprocessors & Supercomputers: Theory, Practice, Experience*, 1994. pp. 124–133.
16. Lamport L. The temporal logic of actions // *ACM Transactions on Programming Languages and Systems*, 1994. vol. 16, no. 3. pp. 872–923. doi: 10.1145/177492.177726.
17. Lamport L. *Specifying systems: The TLA+ Language and Tools for Hardware and Software Engineers.* Boston, San Francisco, etc.: Addison-Wesley, 2002. xviii+364 pp.
18. Востокин С. В. *Графическая объектная модель параллельных процессов и ее применение в задачах численного моделирования.* Самара: СНЦ РАН, 2007. 286 с.
19. Востокин С. В. Система автоматизации параллельного программирования Graphplus Templet / *Параллельные вычисления и задачи управления,* PACO'2010: Труды Пятой Международной конференции. М.: ИПУ РАН, 2010. С. 1143–1156 http://paco2010.ipu.ru/pdf/C105.pdf.
20. Востокин С. В. Templet – метод процессно-ориентированного моделирования параллелизма // *Программные продукты и системы*, 2012. № 3. С. 11–14.
21. Востокин С. В., Литвинов В. Г., Макагонова Д. Д., Хайрутдинов А. Р. Визуальное моделирование параллельных алгоритмов в процессно-ориентированной нотации Templet / *Параллельные вычисления и задачи управления,* PACO'2012: Тр. Шестой Международн. конф-ции. Т. 1. М.: ИПУ РАН, 2012. С. 260–269 http://paco2012.ipu.ru/procdngs/E208.pdf.
22. Осовский С. *Нейронные сети для обработки информации.* М.: Финансы и статистика, 2004. 344 с.
23. Haykin S. O. *Neural Networks and Learning Machines* (3rd Edition): New York, 2008. 936 pp.
24. Литвинов В. Г., Солдатова О. П. Методы построения оптимальной структуры HRBF нейронной сети для решения задачи прогнозирования / *Перспективные информационные технологии для авиации и космоса,* ПИТ-2010: Труды Международной конференции с элементами научной школы для молодежи. СГАУ: Самара, 2010. С. 252–254.

25. Востокин С. В., Хайрутдинов А. Р., Литвинов В. Г. Программный комплекс параллельного программирования Graphplus Templet // *Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки*, 2011. № 4(25). С. 146–153. doi: 10.14498/vsgtu1027.
26. Артамонов Ю. С., Востокин С. В., Назаров Ю. П. Templet – Сервис непрерывной интеграции для разработки высокопроизводительных приложений / *Высокопроизводительные параллельные вычисления на кластерных системах*: Материалы XII всероссийской конференции. Нижний Новгород: Изд-во НГУ, 2012. С. 82.

Поступила в редакцию 18/VIII/2014;
в окончательном варианте — 03/IX/2014;
принята в печать — 10/IX/2014.

Vestn. Samar. Gos. Techn. Un-ta. Ser. Fiz.-mat. nauki. 2014. Issue 3 (36). Pp. 183–195
[J. Samara State Tech. Univ., Ser. Phys. & Math. Sci. 2014. Issue 3 (36). Pp. 183–195]

ISSN: 2310-7081 (online), 1991-8615 (print) **doi:** http://dx.doi.org/10.14498/vsgtu1341

MSC: 68Q10, 68Q85

DEVELOPMENT AND APPLICATION OF THE COMPUTATIONAL MODEL FOR SKELETON SOLUTIONS. CASE STUDY – USING "BAG-OF-TASK" FOR HRBF NEURAL NETWORK LEARNING

V. G. Litvinov

S. P. Korolyov Samara State Aerospace University (National Research University),
34, Moskovskoe sh., Samara, 443086, Russian Federation.

Abstract

The article proposes a solution to the problem of mapping an algorithm from the field of Computational Mathematics on the target computing environment. The solution is based on a formal method for constructing parallel skeletons. The method comprises a specification of concurrency with the directed graphs and a formula for interpretation of dynamic behavior of such graphs. This interpretation is based on Temporal Logic of Actions approach proposed by Leslie Lamport. To illustrate the use of the method the "bag-of-tasks" parallel skeleton is discussed hereinafter. We present graphically basic skeleton operations with the proposed computational model. After that we specify a learning algorithm of hyper-radial basis function neural network in the terms of skeleton operations as a case study. This made it possible to parallelize the leaning algorithm and map it on desired computing environments with predefined run-time libraries. Computational experiments confirming that our approach does not reduce the performance of the resulting programs are presented. The approach is suitable for researchers not familiar with parallel computing. It helps to get a reliable and effective supercomputer application both for SMP and distributed architectures.

How to cite Reference: Litvinov V. G. Development and application of the computational model for skeleton solutions. Case study – using "bag-of-task" for HRBF neural network learning, *Vestn. Samar. Gos. Tekhn. Univ., Ser. Fiz.-Mat. Nauki* [J. Samara State Tech. Univ., Ser. Phys. & Math. Sci.], 2014, no. 3 (36), pp. 183–195. doi: 10.14498/vsgtu1341. (In Russian)
Author Details: *Vladimir G. Litvinov* (litvinov_vg@ssau.ru), Assistant, Dept. of Information Systems & Technology.

Keywords: cluster, supercomputing, Templet language, pattern, bag-of-tasks, skeleton programming, model of computation, HRBF neural network.

doi: http://dx.doi.org/10.14498/vsgtu1341

Acknowledgments. The author thanks Samara State Aerospace University named after academician S. P. Korolyov for its support of this research. The work has been done being supported by The Ministry of Education and Science of the Russian Federation to realize the items of the Samara State Aerospace University named after academician S. P. Korolyov under competitiveness improvement program among leading world scientific-educational centers for the period of 2013–2020.

I am especially thankful to Sergey Vladimirovich Vostokin my scientific supervisor for his advices and great help during the work.

ORCID
Litvinov, Vladimir: http://orcid.org/0000-0002-7082-9805

REFERENCES

1. Sutter H. The free lunch is over: A fundamental turn toward concurrency in software, *Dr. Dobb's journal*, 2005, vol. 30, no. 3, pp. 202–210.
2. Marchuk G. I., Kotov V. E. Problems of computer technology and fundamental research, *Avtom. i vychisl. tekhn.*, 1979, no. 2, pp. 3–14 (In Russian).
3. Gamma E., Helm R., Johnson R., Vlissides J. *Design patterns: elements of reusable object-oriented software*. Westford, Addison-Wesley, 1995, 395 pp.
4. Cole M. I. *Algorithmic skeletons: structured management of parallel computation*. Massachusetts, Cambridge, Boston, MIT Press, 1989, v+132 pp.
5. MacDonald S., Szafron D., Schaeffer J., Bromling S. Generating parallel program frameworks from parallel design patterns, *Euro-Par 2000 Parallel Processing*, Lecture Notes in Computer Science, 1900. Berlin, Heidelberg, Springer, 2000, pp. 95–104. doi: 10.1007/3-540-44520-X_13.
6. Schmidt D. C., Huston S. D. *C++ Network Programming*, vol. I, Mastering Complexity with ACE and Patterns, FT Press, 2001, 336 pp.
7. Berzigiyarov P. K. Programming using standardized algorithmic structures with massive parallelism, *Vychislitel'nye metody i programmirovanie*, 2001, vol. 2, no. 2, pp. 1–16 (In Russian).
8. Kraeva M. A., Malyshkin V. E. Assembly technology for parallel realization of numerical models on mimd-multicomputers, *Future Generation Computer Systems*, 2001, vol. 17, no. 6, pp. 755–765. doi: 10.1016/s0167-739x(00)00058-3.
9. Botorog G. H., Kuchen H. Skil: An imperative language with algorithmic skeletons for efficient distributed programming, *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing,* HPDC-96, 1996, 243–252 pp.. doi: 10.1109/hpdc.1996.546194
10. Dorta A. J., González J. A., Rodríguez C., De Sande F. Llc: A parallel skeletal language, *Parallel Processing Letters*, 2003, vol. 13, no. 03, pp. 437–448. doi: 10.1142/s0129626403001409.
11. Iosup A., Sonmez O., Anoep S., Epema D. The performance of bags-of-tasks in large-scale distributed systems, *Proceedings of the 17th International Symposium on High Performance Distributed Computing 2008,* HPDC'08, 2008, pp. 97–108. doi: 10.1145/1383422.1383435.
12. Reinders J. *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism*. Beijing, Cambridg, etc., O'Reilly Media, 2007, 336 pp.
13. Blumofe R. D., Joerg C. F., Kuszmaul B. C., Leiserson C. E., Randall K. H., Zhou Y. Cilk: An Efficient Multithreaded Runtime System, *Journal of Parallel and Distributed Computing*, 1996, vol. 37, no. 1, pp. 55–69. doi: 10.1006/jpdc.1996.0107.
14. OpenMP: an industry standard API for shared-memory programming, *IEEE Computational Science and Engineering*, 1998, vol. 5, no. 1, pp. 46–55. doi: 10.1109/99.660313.

15. Konovalov N. A., Krukov V. A., Mihailov S. N., Pogrebtsov A. A. Fortran dvm — a language for portable parallel program development, *Proceedings of Software For Multiprocessors & Supercomputers: Theory, Practice, Experience*, 1994, pp. 124–133.
16. Lamport L. The temporal logic of actions, *ACM Transactions on Programming Languages and Systems*, 1994, vol. 16, no. 3, pp. 872–923. doi: 10.1145/177492.177726.
17. Lamport L. *Specifying systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Boston, San Francisco, etc., Addison-Wesley, 2002, xviii+364 pp.
18. Vostokin S. V. *Graficheskaia ob"ektnaia model' parallel'nykh protsessov i ee primenenie v zadachakh chislennogo modelirovaniia* [Graphical object model of parallel processes and its appication in the numerical modeling problem]. Samara, Samara Research Center of RAS, 2007, 286 pp. (In Russian)
19. Vostokin S. V. Parallel programming automation system Graphplus Templet, *Parallel'nye vychisleniia i zadachi upravleniia,* PACO'2010 [Parallel Computing and Control Problems], Proc. Fifth International Conf.. Moscow, Institute of Control Sciences, 2010, pp. 1143–1156 http://paco2010.ipu.ru/pdf/C105.pdf (In Russian).
20. Vostokin S. V. Templet – A method of process oriented simulation of parallelism, *Programmnye produkty i sistemy*, 2012, no. 3, pp. 11–14 (In Russian).
21. Vostokin S. V., Litvinov V. G., Makagonova D. D., Khairutdinov A. R. Visual modeling of parallel algorithms in process oriented notation Templet, *Parallel'nye vychisleniia i zadachi upravleniia,* PACO'2012 [Parallel Computing and Control Problems], Proc. Sixth International Conf., vol. 1. Moscow, Institute of Control Sciences, 2012, pp. 260–269 http://paco2012.ipu.ru/procdngs/E208.pdf (In Russian).
22. Osovskii S. *Neironnye seti dlia obrabotki informatsii* [Neural network for processing information]. Moscow, Finansy i statistika, 2004, 344 pp.
23. Haykin S. O. *Neural Networks and Learning Machines* (3rd Edition), New York, 2008, 936 pp.
24. Litvinov V. G., Soldatova O. P. Methods of constructing optimal structure of the HRBF neural network for solving forecasting problem, *Perspektivnye informatsionnye tekhnologii dlia aviatsii i kosmosa,* PIT-2010 [Advanced information technologies for aviation and cosmos]. Samara State Aerospace Univ., Samara, 2010, pp. 252–254.
25. Vostokin S. V., Khayrutdinov A. R., Litvinov V. G. Parallel programming software package Graphplus templet, *Vestn. Samar. Gos. Tekhn. Univ. Ser. Fiz.-Mat. Nauki*, 2011, no. 4(25), pp. 146–153 (In Russian). doi: 10.14498/vsgtu1027.
26. Artamonov Yu. S., Vostokin S. V., Nazarov Yu. P. Templet – A service of continuous integration to the development of high-performance applications, *Vysokoproizvoditel'nye parallel'nye vychisleniia na klasternykh sistemakh* [High-performance parallel computations on cluster systems], Proceedings of XII All-Russian Conference. Nizhni Novgorod, Nizhni Novgorod State Univ. Publ., 2012, pp. 82 (In Russian).

Received 18/VIII/2014;
received in revised form 03/IX/2014;
accepted 10/IX/2014.