УДК 681.3.068
DOI: 10.15827/0236-235X.137.083-094
Дата подачи статьи: 21.07.21, после доработки: 30.09.21
2022. Т. 35. № 1. С. 083–094

Метод создания параллельных программных средств моделирующих комплексов военного назначения

М.А. Аксенов [1], *адъюнкт, aksen1985@mail.ru*

[1] *Военная академия воздушно-космической обороны им. Г.К. Жукова, г. Тверь, 170100, Россия*

В статье рассмотрены вопросы выбора алгоритмов распараллеливания, реализованных в инструментальных средствах разработки параллельных программ для многоядерных (многопроцессорных) вычислительных систем с общей памятью.

Целью данного исследования является оценка влияния времени выполнения распараллеленных циклических участков целевой программы при многопоточном параллельном выполнении программы в многоядерных (многопроцессорных) ПЭВМ на показатели результатов имитационного моделирования боевых действий. Научная новизна заключается в разработке нового метода создания параллельных программных средств моделирующих комплексов военного назначения.

Проведенный анализ современных программных средств моделирующих комплексов военного назначения показал, что на оперативность их применения при использовании по назначению в значительной степени оказывает влияние продолжительность расчетов при проведении моделирования.

В работе приведены примеры расчетов в среде Mathcad. Для исключения ошибок выбора предпочтительных алгоритмов распараллеливания анализ производился на основе элементов математической статистики с введением вероятности доверительного интервала для оценки времени выполнения цикла определенным алгоритмом по верхней границе доверительного интервала. Предложен вариант построения программных средств на примере внедрения технологических разработок в программную архитектуру моделирующего комплекса.

Ключевые слова: *программное средство, моделирующий комплекс, алгоритм распараллеливания, цикл, число итераций, время выполнения.*

Результаты анализа применения *программных средств моделирующих комплексов военного назначения* (ПС МК ВН) на мероприятиях оперативной подготовки показали, что наряду с их достоинствами имеются и существенные недостатки. Основным из них является значительное превышение нормативных сроков представления результатов имитационного моделирования (выходных параметров) в условиях ограничений по времени применения ПС МК ВН в соответствии с их назначением, что не согласуется с временным алгоритмом работы должностных лиц ВС РФ [1].

В современных архитектурах ПЭВМ, поддерживающих параллелизм в многоядерных (многопроцессорных) вычислительных системах с общей памятью, наиболее распространены универсальные средства программирования: стандарты и технологии, предназначенные для создания современных параллельных программ, такие как языковые средства задания параллельности в программе, автоматические средства поиска параллельных вычислений и последующего представления их с использованием существующих библиотек поддержки параллельности [2]. Для распараллеливания целесообразно рассматривать циклические участки программы, так как около 80 % возможностей для распараллеливания заключено именно в циклах [3].

В современной технической литературе существуют работы, описывающие методы сравнения средств распараллеливания, которые применяют различные метрики. Однако они имеют общий недостаток: сравнение технологий осуществляется либо на подготовленных тестах, либо теоретически, а в некоторых из них рассматриваются только время обучения и трудоемкость реализации процесса [4–6 и др.]. Имеется тенденция к объединению нескольких средств распараллеливания в одной разрабатываемой программе.

Сопоставительный анализ разработанных ранее методов применения средств параллельного программирования показывает, что они не учитывают следующее:

– случайный характер изменений параметров в структуре одних и тех же циклов программы по количеству итераций цикла и времени выполнения, что приводит к изменению

критических по времени выполнения циклов и может стать причиной ошибочного выбора предпочтительного алгоритма распараллеливания;

– зависимость эффективности использования различных средств параллельного программирования от числа итераций цикла и от его внутренней структуры;

– разная зависимость временных затрат на выполнение одного того же участка программного кода для разных средств распараллеливания при одних и тех же требованиях к аппаратно-программному обеспечению.

Идея предлагаемого в данном исследовании метода состоит в создании системы параллельного программирования. Программист на традиционных языках программирования (C, C++) создает программу с параллельными вычислениями. В случае возникновения проблем с анализом он может добавить подсказки в программу, разрешив какой-либо конфликт, мешающий определению параллельности. Затем найденные параллельные вычисления автоматически представляются в параллельных терминах конкретной архитектуры. По сути, это некоторая интерактивная среда разработки параллельных программ, в которой сбалансированы усилия человека и автоматических средств. В предложенном подходе основную роль при анализе, модернизации и инструментировании параллельного кода выполняет средство распараллеливания, а разработчик (программист) прагмами разрешает конфликты (если они возникают), с которыми статический анализатор может не справиться.

Разработанный метод реализует автоматический выбор предпочтительного алгоритма распараллеливания из доступного набора алгоритмов под каждый конкретный циклический участок текущей программы по минимальному времени выполнения цикла. Для этого используются накопленные при тестировании статистические данные в виде массива временных профилей программы и составленного на их основе профиля проекта, представляющего собой совокупность кортежей предпочтительных алгоритмов распараллеливания, функционально связанных с текущим числом итераций цикла. В предлагаемом методе по входным параметрам из выполняемой программы ПС МК ВН (номеру цикла и количеству итераций данного цикла) с использованием составленного кортежа предпочтительных алгоритмов распараллеливания в реальном масштабе времени определяется номер алгоритма распараллеливания.

Целью данной статьи является оценка влияния процесса автоматизации выбора предпочтительного алгоритма распараллеливания из доступного набора алгоритмов под каждый циклический участок целевой программы при многопоточном параллельном выполнении программы в многоядерных (многопроцессорных) ПЭВМ на показатели результатов имитационного моделирования боевых действий.

Предлагаемый метод параллельного программирования ПС МК ВН представляет собой последовательность взаимосвязанных методик. Структурная схема метода проказана на рисунке 1. В рамках его создания были разработаны определенные научные методики.

Методика построения профилеобразующей базы программы

Для реализации методики необходимо выполнить определенную последовательность действий. В исходном коде ПС МК ВН выявляются циклы, имеющие счетчик числа итераций (циклы со счетчиком) и не имеющие в теле цикла команд принудительного выхода из него. Для разбора конструкции цикла проводится синтаксический анализ исходного кода программы, в ходе которого символы этого кода группируются в лексемы (распознанные группы символов) в соответствии с набором токенов, а затем в конструкции цикла в соответствии с деревом разбора. Изначально в исходном коде программы выделяются лексемы: параметр цикла, тело цикла. Выявление цикла выполняется по последовательности символов, содержащихся в токенах. В сформированных лексемах в ходе лексического анализа идентифицируются входящие в них токены, определяются параметры токенов. В ходе разбора конструкции цикла выявляются по вхождению в состав лексемы: вложенные циклы, тело цикла, параметры цикла. Циклы номеруются, клонируются, в клоны добавляются прагмы алгоритмов распараллеливания. Полученный код программы компилируется, и производится многократный запуск на тестовый прогон для получения первых статистических данных по всем циклическим участкам с примененными алгоритмами распараллеливания и вариантами числа итераций. В результате тестовых запусков формируется профилеобразующая база программы, состоящая из набора кортежей таблицы: $v = \left\{ v_p \right\}_{p=1}^{N_{vp}} = c_z ; n_s^{it} ; n_k^{sr} ; t_i^{vip}$, в которые входят номер цикла c, количество итераций n^{it} в

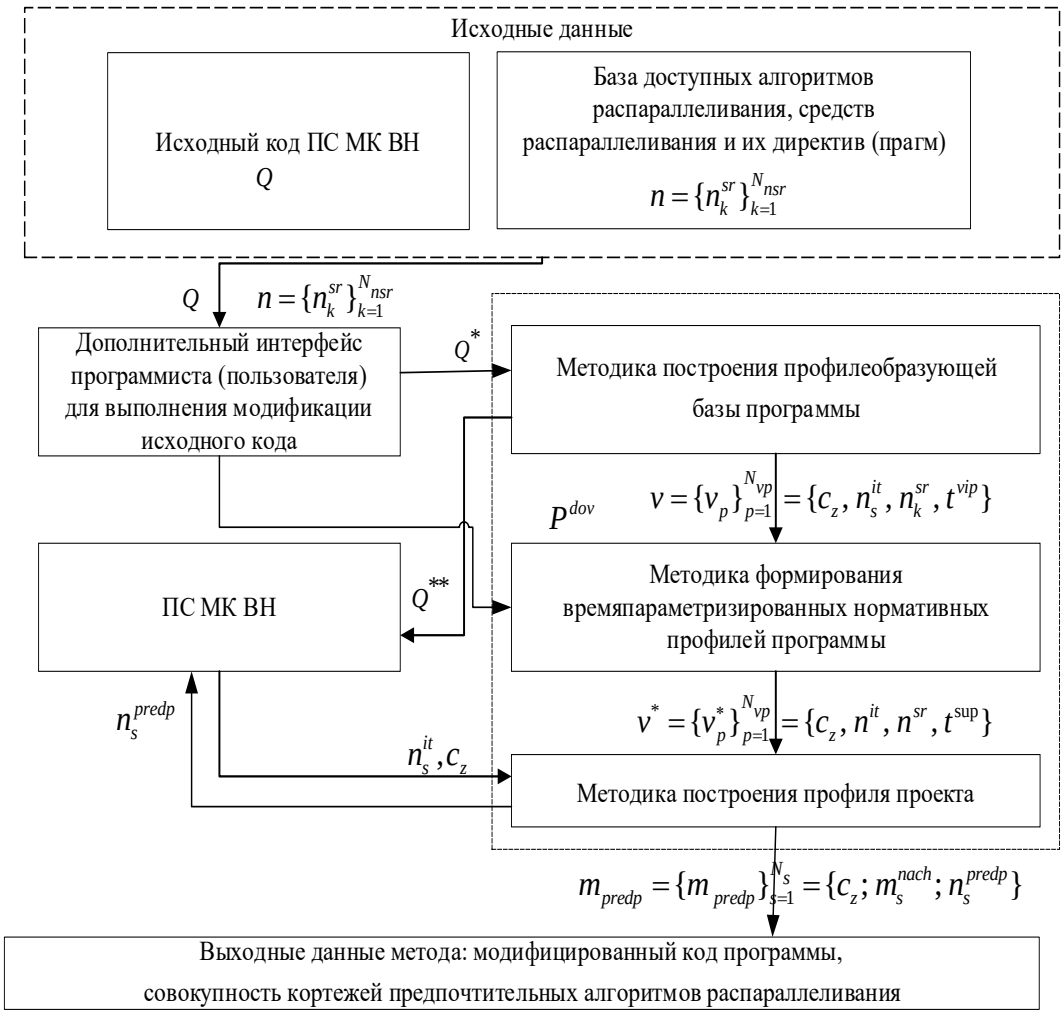

Рис. 1. Структурная схема метода создания параллельных ПС МК ВН

Fig. 1. A block diagram of a method for creating parallel software tools for modeling military complexes

данном цикле, номер алгоритма распараллеливания n^{sr} и время выполнения t^{vip} для каждого из модифицированных циклических участков программы.

В таблице 1 приведен пример профилеобразующей базы первичной статистики, который формируется в результате многократного тестового прогона цикла с возможностью распараллеливания пятью алгоритмами при 20 дискретных итерациях. Данный профиль программы первичной статистики представляет собой массив статистических данных, состоящих из записей четырех параметров: номер цикла c_z, количество итераций цикла n_s^{it}, номер алгоритма распараллеливания n_k^{sr}, время работы t_i^{vip}.

Методика формирования времяпараметризированных нормативных профилей

Поскольку функция распределения вероятности времени выполнения цикла неизвестна [7], из полученной профилеобразующей базы программы формируются эмпирические функции распределения времени выполнения каждого цикла следующим образом:

– формируется выборка из профилеобразующей базы первичной статистики для одного значения: номера цикла c_z, числа итераций n_s^{it} этого цикла, номера алгоритма распараллеливания n^{sr}; соответственно, выборка содержит массив времен выполнения цикла $\left\{t_i^{vip}\right\}_{i=1}^{N_t}$;

– элементы массива времен выполнения цикла $\left\{t_i^{vip}\right\}_{i=1}^{N_t}$ ранжируются по возрастанию значения времени;

– рассчитываются *j*-е значения функции распределения для ранжированного массива $\left\{t_i^{vip}\right\}_{i=1}^{N_t}$: $P_j = \frac{1}{N_t}\sum_{i=1}^{N_t} I_{ij}$, где $I_{ij} = \begin{cases} 1 \text{ при } t_i^{vip} \le t_j^{vip}, \\ 0 \text{ при } t_i^{vip} > t_j^{vip} \end{cases}$ – индикаторная функция.

Таблица 1

Пример профилеобразующей базы первичной статистики с алгоритмами распараллеливания

Table 1

An example of a profile forming base of primary statistics with parallelization algorithms

Номер алгоритма распараллеливания	Время работы, сек.
1	2.105
2	2.425
3	1.118
4	0.408
5	0.128
1	1.921
2	1.962
3	0.926
4	0.526
5	0.082
1	1.811
2	1.881
3	0.781
4	0.526
5	0.118
…	…

Примечание: номер цикла $c_z = 1$, количество итераций цикла $n_s^{it} = 20$.

Эмпирическая функция распределения аппроксимируется гладкой кривой. Так как вид функции распределения неизвестен, необходимо применять непараметрические методы аппроксимации. В [8] показано, что высокая точность аппроксимации непараметрическим методом как унимодальных, так и полимодальных распределений, заданных на конечных интервалах, обеспечивается применением полиномов Бернштейна. При этом оценка функции распределения описывается выражением

$$P^*\left(t^{vip}\right) = \sum_{n=1}^{N_b} \omega_n F\left(t^{vip}, \alpha, \beta\right),$$

где N_b – степень полинома Бернштейна; ω_n – весовые коэффициенты полинома Бернштейна; $F\left(t^{vip}, \alpha, \beta\right) = \int_0^{t^{vip}} t^{\alpha-1}\left(1-t\right)^{\beta-1} dt$ – функция бэта-распределения.

Расчет параметров оценочной функции $P^*(t^{vip})$ выполняется в два этапа: на первом этапе при заданной степени полинома N_b (начиная с минимального значения N_b = 2) определяются значения весовых коэффициентов ω_n, на втором повторяется первый этап для набора значений N_b и выбирается величина N_b, при которой обеспечивается требуемая точность аппроксимации.

Для принятия решения и оценки погрешности по возможности использования аппроксимирующей функции рассчитывается доверительный интервал для каждого *j*-го значения эмпирической функции распределения:

$$z_j^{(1,2)} = \frac{2N_t P^*(t_j^{vip}) + \varepsilon^2}{2(N_t + \varepsilon^2)} \pm \pm \frac{\sqrt{4N_t\varepsilon^2 P^*(t_j^{vip})\left(1 - P^*(t_j^{vip})\right) + \varepsilon^4}}{2(N_t + \varepsilon^2)},$$

где ξ – квантиль нормального распределения для заданной доверительной вероятности (например, для P^{dov} = 0.99, значение ξ = 2.576).

Аппроксимирующая функция принимается при выполнении для всех *j*-х значений эмпирической функции распределения условия $z_j^{(1)} \le P_j \le z_j^{(2)}$.

В противном случае степень полинома N_b увеличивается на единицу и выполняется повторный расчет весовых коэффициентов.

Расчет значения верхней границы $t^{\sup}$ времени выполнения цикла осуществляется путем решения уравнения $P^*(t^{\sup}) = P^{dov}$. Ввиду того, что отсутствуют аналитические способы решения данного уравнения, поиск $t^{\sup}$ выполняется численным методом.

При этом появляется возможность сравнить по данному детерминированному параметру разные средства распараллеливания цикла и произвести выбор с минимальным временем.

Далее с использованием рассчитанного значения $t^{\sup}$ формируется времяпараметризированный нормативный профиль программы: $\nu^* = \left\{c; n_s^{it}; n^{sr}; t^{\sup}\right\}$. Он представляет собой массив данных, состоящий из векторов параметров: количество итераций цикла n_s^{it}, номер алгоритма распараллеливания n_k^{sr}, верхняя граница времени выполнения $t^{\sup}$ и номер цикла c_z.

В таблице 2 представлен времяпараметризированный нормативный профиль с расчетным временем (по верхней границе доверительного интервала) выполнения цикла, кото-

рый получен в результате переработки профилеобразующей базы первичной статистики таблицы 1 по методике формирования времяпараметризированных нормативных профилей.

Таблица 2

Пример обработанного времяпараметризированного нормативного профиля программы

Table 2

An example of a processed time parameterized standard program profile

Номер алгоритма распараллеливания	Время выполнения, сек.
1	2.096
2	2.402
3	1.101
4	0.702
5	0.1
…	…

Примечание: номер цикла $c_z = 1$, количество итераций цикла $n_s^{it} = 20$.

Представим результаты расчетов в Mathcad 15.0 математического аппарата методики формирования времяпараметризированных нормативных профилей, заключающейся в построении эмпирической функции распределения случайной величины времени выполнения одного заданного цикла заданным средством распараллеливания при заданном числе итераций цикла (20) и пятикратном запуске. В приведенном примере значение верхней границы времени выполнения цикла $t^{\sup}$ заданным средством составляет 2.096 сек. Для выбора алгоритма с минимальным временем в данном примере необходимо аналогично строить эмпирические функции распределения для всех номеров алгоритмов распараллеливания с определением времени верхней границы заданного доверительного интервала.

Рассмотрим следующие исходные данные:

– вектор времени выполнения цикла:

$$tt\_vip = \begin{pmatrix} 2.105 \\ 1.921 \\ 1.811 \\ 1.652 \\ 1.413 \end{pmatrix};$$

– доверительная вероятность: $P\_dov := 0.99$,

$t\_\max := \max(tt\_vip)$, $t\_\max = 2.105$.

Решение:

1. Сортировка вектора времен по возрастанию: $t\_sort := sort(tt\_vip)$.

Размерность исходного вектора времен выполнения цикла:

$N\_tt := rows(tt\_vip)$, $N\_tt := 5$.

2. Формирование вектора времени выполнения цикла без повторяющихся значений времени:

$t\_vip_0 := 0$,

$$t\_vip := \left| \begin{array}{l} j \leftarrow 1 \\ i \leftarrow 0 \\ \text{while } i < N\_tt \left| \begin{array}{l} t\_vip_j \leftarrow t\_sort_i \\ \quad \text{if } t\_sort_i > t\_vip_{j-1} \\ j \leftarrow j+1 \text{ if } t\_sort_i > t\_vip_{j-1} \\ i \leftarrow i+1 \end{array} \right. \\ t\_vip \end{array} \right.$$

$$t\_sort = \begin{pmatrix} 1.413 \\ 1.652 \\ 1.811 \\ 1.921 \\ 2.105 \end{pmatrix}.$$

Размерность преобразованного вектора времен выполнения цикла:

$N\_t := rows(t\_vip)$, $N\_t = 6$.

3. Формирование эмпирической функции распределения: $P_0 := 0$, $P_1 := 0$

$$P := \left| \begin{array}{l} j \leftarrow 1 \\ i \leftarrow 0 \\ \text{while } i < N\_tt \left| \begin{array}{l} P_j \leftarrow \dfrac{(i+1)}{N\_tt} \text{ if } t\_sort_i > t\_vip_{j-1} \\ j \leftarrow j+1 \text{ if } t\_sort_i > t\_vip_{j-1} \\ i \leftarrow i+1 \end{array} \right. \\ P \end{array} \right.$$

$$t\_vip = \begin{pmatrix} 0 \\ 1.413 \\ 1.652 \\ 1.811 \\ 1.921 \\ 2.105 \end{pmatrix}, \quad P = \begin{pmatrix} 0 \\ 0.2 \\ 0.4 \\ 0.6 \\ 0.8 \\ 1 \end{pmatrix}.$$

4. Упрощенный расчет верхней границы времени:

$$t\_\sup := \left| \begin{array}{l} j \leftarrow 1 \\ \text{while } i < N\_t \left| \begin{array}{l} t\_\sup \leftarrow t\_vip_{j-1} + \\ \quad + \dfrac{(P\_dov - P_{j-1})(t\_vip_j - t\_vip_{j-1})}{(P_j - P_{j-1})} \\ \text{if } (P\_dov - P_{j-1})(P\_dov - P_j) \le 0 \\ i \leftarrow i+1 \end{array} \right. \\ t\_\sup \end{array} \right.$$

$t\_\sup = 2.096$

Методика построения профиля проекта

Для исследуемого порядкового номера цикла *c* времяпараметризированный нормативный профиль перерабатывается и представляется в виде матрицы

$$v_c^* = \begin{pmatrix} n_1^{it} & n_2^{it} & \dots & n_s^{it} & \dots & n_{N_{it}}^{it} \\ t_{1.1}^{\sup} & t_{1.2}^{\sup} & \dots & t_{1,s}^{\sup} & \dots & t_{1,N_{it}}^{\sup} \\ t_{2.1}^{\sup} & t_{2.2}^{\sup} & \dots & t_{2,s}^{\sup} & \dots & t_{2,N_{it}}^{\sup} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{q.1}^{\sup} & t_{q.2}^{\sup} & \dots & t_{q,s}^{\sup} & \dots & t_{q,N_{it}}^{\sup} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{N_{sr}.1}^{\sup} & t_{N_{sr}.2}^{\sup} & \dots & t_{N_{sr},s}^{\sup} & \dots & t_{N_{sr},N_{it}}^{\sup} \end{pmatrix},$$

где $s = \overline{1, N_{it}}$ – порядковый номер числа итераций цикла; $q = \overline{1, N_{sr}}$ – порядковый номер алгоритма распараллеливания.

Массив ранжирован по возрастанию значений n_s^{it}. Первый столбец матрицы состоит из нулевых элементов. Элементами следующих строк являются верхние границы времен выполнения цикла с учетом разных номеров примененных алгоритмов распараллеливания. Затем выполняются интерполяция и экстраполяция зависимости верхней границы времени выполнения цикла $t^{\sup}$ от числа итераций n_s^{it} для всех рассматриваемых алгоритмов распараллеливания.

Интерполяция и экстраполяция времяпараметризированного нормативного профиля программы v_c^* необходимы для расчета величины $t^{\sup}$ для любых значений n_s^{it} (не совпадающих со значениями n_s^{it}, содержащимися в профиле). При интерполяции и экстраполяции используется одинаковый подход, основанный на применении кубических сплайнов.

При использовании кубических сплайнов на каждом из отрезков $[n_{s-1}^{it}, n_s^{it}]$ определяется аппроксимирующая функция для q-го алгоритма распараллеливания:

$$f_{q,s}\left(n^{it}\right) = a_{q,s} + b_{q,s}\left(n^{it} - n_s^{it}\right) + c_{q,s}\left(n^{it} - n_s^{it}\right)^2 + d_{q,s}\left(n^{it} - n_s^{it}\right)^3,$$

где n^{it} – текущее число итераций цикла; $a_{q,s}, b_{q,s}, c_{q,s}, d_{q,s}$ – коэффициенты сплайна для s-го числа итераций цикла n_s^{it} q-го алгоритма распараллеливания; n_s^{it} – s-е число итераций цикла.

Рассчитанные значения коэффициентов сплайна сохраняются в оперативной памяти для их дальнейшего использования.

После нахождения аппроксимирующих функций $f_{q,s}(n^{it})$ зависимости $t^{\sup}$ от n^{it} они сглаживаются гладкими кривыми. На рисунке 2 приведен пример аппроксимации верхней

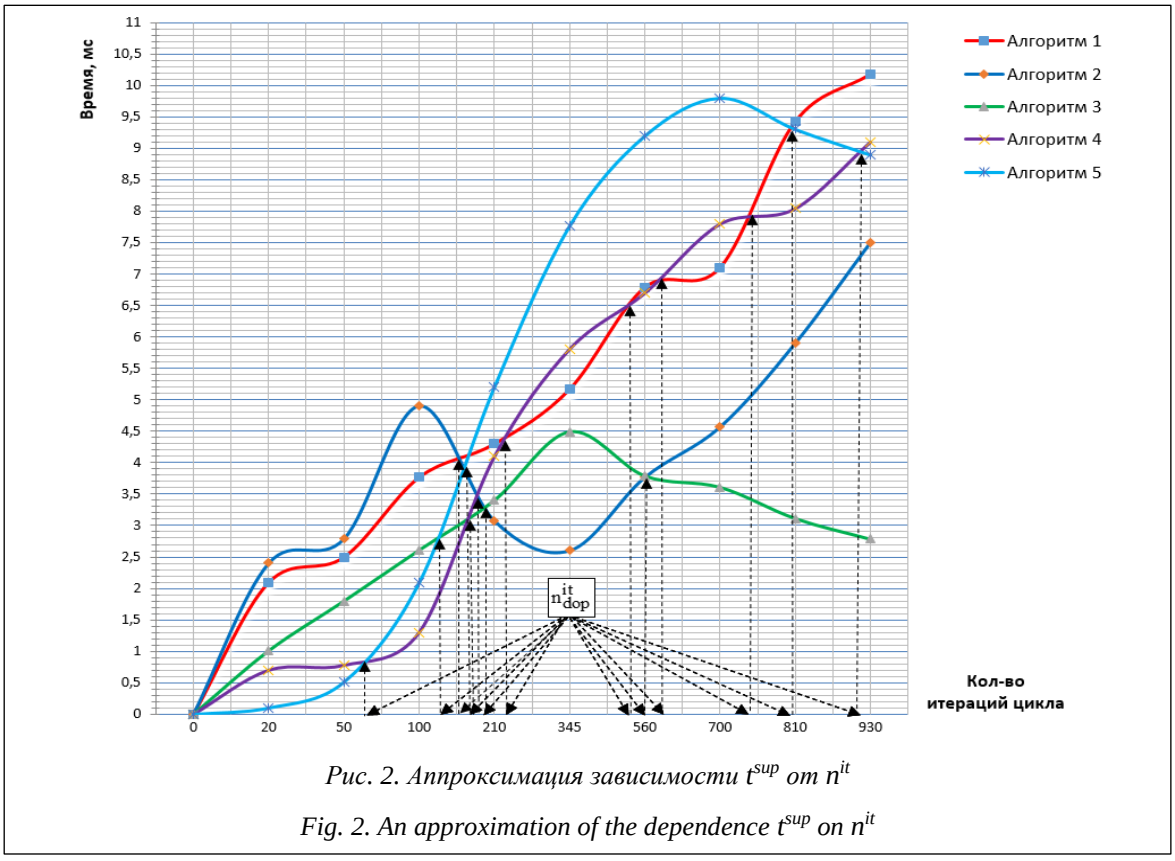

Рис. 2. Аппроксимация зависимости t^{sup} от n^{it}

Fig. 2. An approximation of the dependence t^{sup} on n^{it}

границы времени выполнения цикла $t^{\sup}$ от числа итераций для пяти алгоритмов распараллеливания. Из примера понятно, что могут возникать пересечения аппроксимирующих кривых разных алгоритмов распараллеливания.

Поскольку предпочтительный алгоритм выбирается в процессе штатной работы программы, необходимо минимизировать время выбора. Для этого времяпараметризированный нормативный профиль программы $v^* = \{c; n_s^{it}; n^{sr}; t^{\sup}\}$ преобразуется в кортеж предпочтительных алгоритмов распараллеливания: $m_{predp} = \{c; m^{nach}; n^{predp}\}$, где m^{nach} – начальное число итераций цикла; n^{predp} – номер предпочтительного алгоритма распараллеливания для этого цикла при определенном числе итераций. Такое преобразование требует нахождения участков $[m_s^{nach}; m_{(s+1)}^{nach}]$, на которых отсутствуют пересечения аппроксимирующих сплайнов разных алгоритмов распараллеливания. Для этого в сформированный массив профиля программы необходимо добавить столбцы по каждому такому дополнительному значению числа итераций n_{dop}^{it}, возникающему в местах пересечения сплайнов. Данная операция выполняется путем формирования набора векторов-столбцов дополнительных значений числа итераций цикла.

Для этого организуется цикл по числу итераций $s = \overline{2; N_{it}}$. На каждом s-м шаге цикла для значений числа итераций n_{s-1}^{it}, n_s^{it} формируются наборы пар алгоритмов распараллеливания из $q1$-го и $q2$-го алгоритмов $(q1(2) = \overline{1, N_{sr}})$. Для каждого элемента этой пары из оперативной памяти считываются соответствующие значения коэффициентов сплайна $a_{q1(2),s}$, $b_{q1(2),s}$, $c_{q1(2),s}$, $d_{q1(2),s}$, рассчитанные ранее.

Так как в точках пересечения сплайнов значения аппроксимирующих функций $f_{q1,s}(n^{it})$, $f_{q2,s}(n^{it})$ имеют одинаковую величину, они приравниваются друг к другу. В результате выполняется поиск действительных корней кубического уравнения относительно переменной n^{it} (абсцисс точек пересечения):

$$(a_{q1,s} - a_{q2,s}) + (b_{q1,s} - b_{q2,s})x_s + $$
$$+ (c_{q1,s} - c_{q2,s})x_s^{\,2} + (d_{q1,s} - d_{q2,s})x_s^{\,3} = 0,$$
$$x_s = n^{it} - n_s^{it}.$$

Далее анализируется попадание найденных корней n^{it} в рассматриваемый интервал $[n_{s-1}^{it}, n_s^{it}]$ по выполнению условия $n_{s-1}^{it} < n^{it} < n_s^{it}$. При этом для последнего значения s = N_{it} при проверке предыдущего условия принимается $n_s^{it} = \infty$, так как последний отрезок используется для экстраполяции зависимости $t^{\sup}$ от n^{it}. В случае выполнения условия найденные действительные корни n^{it} фиксируются как дополнительное значение числа итераций n_{dop}^{it}. Для каждого значения n_{dop}^{it} рассчитываются величины $t_{q,s}^{\sup}$ по каждому q-му средству распараллеливания. Таким образом, формируются дополнительные столбцы матрицы v_c^*, которыми дополняется ранее полученная матрица. В результате формируется расширенная матрица профиля программы v_c^{p*}. Столбцы этой матрицы ранжируются в порядке возрастания значений n_s^{it}:

$$v_c^{p*} = \begin{pmatrix} n_1^{it} & n_{dop1}^{it} & n_2^{it} & n_{dop2}^{it} & \dots & n_s^{it} & \dots & n_{N_{it}}^{it} \\ t_{1.1}^{\sup} & t_{1,dop1}^{\sup} & t_{1.2}^{\sup} & t_{1,don2}^{\sup} & \dots & t_{1,s}^{\sup} & \dots & t_{1,N_{it}}^{\sup} \\ t_{2.1}^{\sup} & t_{2,dop1}^{\sup} & t_{2.2}^{\sup} & t_{q,don2}^{\sup} & \dots & t_{2,s}^{\sup} & \dots & t_{2,N_{it}}^{\sup} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{q.1}^{\sup} & t_{q,2\,dop1}^{\sup} & t_{q.2}^{\sup} & t_{q,don2}^{\sup} & \dots & t_{q,s}^{\sup} & \dots & t_{q,N_{it}}^{\sup} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{N_{sr}.1}^{\sup} & t_{N_{sr},dop1}^{\sup} & t_{N_{sr}.2}^{\sup} & t_{N_{cp},don2}^{\sup} & \dots & t_{N_{sr},s}^{\sup} & \dots & t_{N_{sr},N_{it}}^{\sup} \end{pmatrix}$$

Для каждого интервала $[m_s^{nach}; m_{s+1}^{nach}]$ кортежей предпочтительных алгоритмов определяется номер алгоритма распараллеливания, имеющий минимальное значение верхней границы времени выполнения цикла $t_{q,s}^{\sup}$:

$$n_s^{opt} = \arg\min_q t_{q,s}^{\sup}.$$

В результате после объединения смежных интервалов с одинаковыми предпочтительными алгоритмами распараллеливания формируется профиль проекта с числом элементов $s = N_m : m_{predp} = \{c; m_s^{nach}; n_s^{predp}\}_{s=1}^{N_m}$. Полученный профиль m_{predp} сохраняется в БД.

Пример профиля проекта представлен в таблице 3. Он получен в результате обработки времяпараметризированного нормативного профиля таблицы 2 с расчетным временем (по верхней границе доверительного интервала) выполнения цикла по методике построения профиля проекта.

В таблице 3 отражен конечный профиль проекта для указанного диапазона итераций цикла. Он представляет собой совокупность кортежей, состоящих из трех параметров: номер цикла c_z, начальное количество итераций

цикла m_z^{nach}, номер предпочтительного алгоритма распараллеливания n_s^{predp}. Поскольку некоторые смежные интервалы могут иметь одинаковые предпочтительные алгоритмы распараллеливания, выполняется объединение таких смежных интервалов.

Таблица 3

Пример профиля проекта

Table 3

An example of a project profile

Номер цикла	Начальное количество итераций	Номер предпочтительного алгоритма распараллеливания
1	0	5
1	62	4
1	171	3
1	196	2
1	562	3

Приведем пример расчетов в Mathcad 15.0 по методике построения профиля проекта, представляющий собой совокупность кортежей предпочтительных алгоритмов распараллеливания. В нем значительно расширены значения параметра количества итераций до десяти дискретных точек, первая из которых с временем t^{sup} = 2.096 сек., при количестве итераций 20 (табл. 1–3).

$$v\_c = \begin{pmatrix} 0 & 20 & 50 & 100 & 210 & 345 & 560 & 700 & 810 & 930 \\ 0 & 2.096 & 2.499 & 3.71 & 4.301 & 5.17 & 6.792 & 7.106 & 9.43 & 10.185 \\ 0 & 2.402 & 2.789 & 4.908 & 3.08 & 2.603 & 3.768 & 4.569 & 5.902 & 7.507 \\ 0 & 1.101 & 1.8 & 2.61 & 3.402 & 4.493 & 3.791 & 3.61 & 3.113 & 2.784 \\ 0 & 0.702 & 0.78 & 1.301 & 4.1 & 5.811 & 6.703 & 7.803 & 8.05 & 9.11 \\ 0 & 0.1 & 0.511 & 2.09 & 5.202 & 7.769 & 9.204 & 9.808 & 9.312 & 8.904 \end{pmatrix}$$

Решение.

1. Число алгоритмов распараллеливания: $N\_sr := rows(v\_c) - 1,\ N\_sr = 5.$

2. Число итераций цикла $N\_it := cols(v\_c)$, $N\_it := 10$.

3. Вектор дополнительного числа итераций: $n\_it\_dp :=$

$$= \left| \begin{array}{l} \text{for} \\ s \in 0..(N\_it-2) \\ \text{for} \\ i \in 0..(N\_dop-1) \\ \text{for} \\ q \in 1..N\_sr \\ \\ \\ \\ v\_c\_dop \end{array} \right| \left| \begin{array}{l} v\_c\_dop_{o,i} \leftarrow n\_it\_dop_i \\ \text{if } v\_c_{0,s} \le n\_it\_dop_i < v\_c_{0,(s+1)} \\ v\_c\_dop_{q,i} \leftarrow \\ \dfrac{[v\_c_{q,(s+1)} \cdot (n\_it\_dop_i - v\_c_{0,s})}{v\_c_{0,(s+1)} - v\_c_{0,s}} - \\ - \dfrac{v\_c_{q,s} \cdot [n\_it\_dop_i - v\_c_{0,(s+1)}]]}{v\_c_{0,(s+1)} - v\_c_{0,s}} \\ \text{if } v\_c_{0,s} \le n\_it\_dop_i < v\_c_{0,(s+1)} \end{array} \right|$$

Целая часть: $n\_it\_dop := floor(n\_it\_dp)$.

Число дополнительных итераций:

$N\_dop := rows(n\_it\_dp)$, $N\_dop = 15$.

4. Матрица дополнительных элементов: $m\_nach\_opt :=$

$$= \left| \begin{array}{l} \text{for} \\ s \in 0..N\_itr-2) \\ \\ \\ \\ v\_c\_dop \end{array} \right| \left| \begin{array}{l} m\_na\_dop_{o,i} \leftarrow n\_it\_dop_i \\ \text{if } v\_c_{0,s} \le n\_it\_dop_i < v\_c_{0,(s+1)} \\ v\_c\_dop_{q,i} \leftarrow \\ \leftarrow \dfrac{[v\_c_{q,(s+1)} \cdot (n\_it\_dop_i - v\_c_{0,s})}{v\_c_{0,(s+1)} - v\_c_{0,s}} - \\ - \dfrac{v\_c_{q,s} \cdot [n\_it\_dop_i - v\_c_{0,(s+1)}]]}{v\_c_{0,(s+1)} - v\_c_{0,s}} \\ \text{if } v\_c_{0,s} \le n\_it\_dop_i < v\_c_{0,(s+1)} \end{array} \right|$$

5. Расширенный профиль:

объединение матриц по горизонтали $v\_c\_r := augment(v\_c, v\_c\_dop)$.

6. Отсортированный профиль (по возрастанию числа итераций):

$v\_c\_r\_sort := rsort(v\_c\_r, 0)$.

Количество итераций в расширенном профиле $N\_itr := cols(v\_c\_r\_sort)$, $N\_itr = 25$.

7. Начальный профиль проекта предпочтительных алгоритмов распараллеливания: $m\_nach\_opt :=$

$$= \left| \begin{array}{l} \text{for} \\ s \in 0..(N\_itr-2) \\ \\ \\ \\ m\_nach\_opt \end{array} \right| \left| \begin{array}{l} m\_nach\_opt_{s,0} \leftarrow \\ \leftarrow v\_c\_r\_sort_{0,s} \\ vv \leftarrow \\ \leftarrow submatrix \begin{bmatrix} v\_c\_r\_sort, \\ 1, N\_sr, (s+1), \\ (s+1) \end{bmatrix} \\ m\_nach \leftarrow match(\min(vv), vv) + 1 \\ m\_nach\_opt_{s,1} \leftarrow m\_nach_{0,0} \end{array} \right|$$

Количество строк начальной матрицы:

$N\_nach := rows(m\_nach\_opt)$, $N\_nach = 24$.

8. Конечный профиль предпочтительных алгоритмов распараллеливания: $m\_kon\_opt :=$

$$= \left| \begin{array}{l} s \leftarrow 1 \\ \text{for } i \\ m\_kon\_opt_{0,i} \leftarrow \\ \leftarrow m\_nach\_opt_{0,i} \\ \text{for} \\ k \in 0..(N\_nach-2) \\ \\ \\ \\ \\ \\ m\_kon\_opt \end{array} \right| \left| \begin{array}{l} \text{for } n \in 0..1 \\ m\_kon\_opt_{s,n} \leftarrow \\ \leftarrow m\_nach\_opt_{(k+1),n} \\ \text{if } m\_nach\_opt_{k,1} = \\ = m\_nach\_opt_{(k+1),1} \\ s \leftarrow s+1 \\ \text{if } m\_nach\_opt_{k,1} = \\ = m\_nach\_opt_{(k+1),1} \end{array} \right|$$

Количество строк:

$N\_kon := rows(m\_kon\_opt),\ N\_kon = 5.$

9. Выбор предпочтительного алгоритма распараллеливания:

$$m\_kon\_opt = \begin{pmatrix} 0 & 5 \\ 62 & 4 \\ 171 & 3 \\ 196 & 2 \\ 563 & 3 \end{pmatrix}.$$

Входной параметр – число итераций цикла $n\_z = 65$.

$$n\_opt := \left| \begin{array}{l} n\_opt \leftarrow m\_kon\_opt_{(N\_kon-1),1} \\ \text{if } m\_kon\_opt_{(N\_kon-1),0} \le n\_z \\ \text{for } ss \in 0..(N\_kon-2) \\ \quad \left| \begin{array}{l} n\_opt \leftarrow m\_kon\_opt_{ss,1} \\ \text{if } m\_kon\_opt_{ss,0} \le n\_z \le \\ \le m\_kon\_opt_{(ss+1),0} \end{array} \right. \\ \\ n\_opt \end{array} \right.$$

$n\_opt = 4$.

В ходе штатного выполнения программы определяется текущее число итераций цикла n_z. Из БД считывается профиль проекта m_{predp}. Далее методом дихотомии осуществляется поиск предпочтительного алгоритма распараллеливания n_s^{predp}. Применение метода дихотомии позволяет ускорить поиск предпочтительного алгоритма.

Опишем разработанную программу, реализующую предлагаемый метод [9]. Структура макета программы содержит перечень соответствующих программных модулей, представленных на рисунке 3.

На рисунке 4 приведена структурная схема дополнительной БД, включающей в себя объекты и их атрибуты, необходимые для реализации программы [10].

Таблицы БД связаны с программной реализацией и исполняемым кодом ПС МК ВН. Таблица «Алгоритмы распараллеливания» заполняется до начала разработки средствами используемой СУБД. Таблица «Профилеобразующая база программы» заполняется в процессе проведения тестовых запусков разрабатываемого ПС с использованием добавленной в ПС соответствующей библиотеки, реализующей заявленный метод. Таблицы «Нормативный времяпараметризированный профиль» и «Профиль проекта» рассчитываются и заполняются после окончания выполнения тестовых запусков разрабатываемой программы. Таблица «Профиль проекта» дополнительно обновляется в процессе работы готового ПС МК ВН.

На рисунке 5 представлена схема программной реализации предлагаемого метода. Тело программного кода увеличивается на определенное количество клонов циклов, равное количеству алгоритмов распараллеливания, реализованных средствами распараллеливания, но при этом продолжительность исполнения программы остается прежней, так как выполняется только один выбранный клон. Процесс управления и выбор клона (с примененным алгоритмом распараллеливания) позволяет снизить временные затраты на выполнение циклических участков программы и в целом всей программы. Это связано с тем, что временные затраты на выполнение одного и того же циклического участка с заданным количеством итераций при применении разных средств рас-

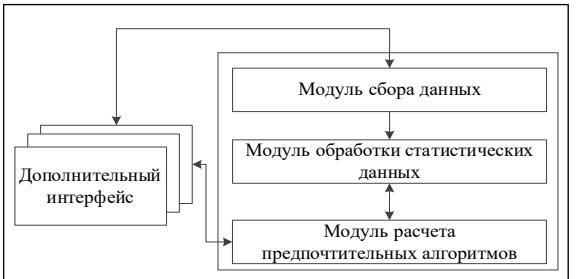

Рис. 3. Модули, реализующие метод

Fig. 3. The modules implementing the method

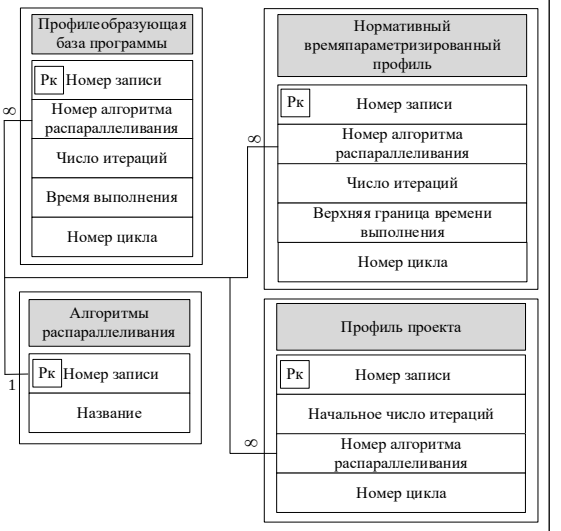

Рис. 4. Структурная схема дополнительной БД

Fig. 4. The structural diagram of the additional database

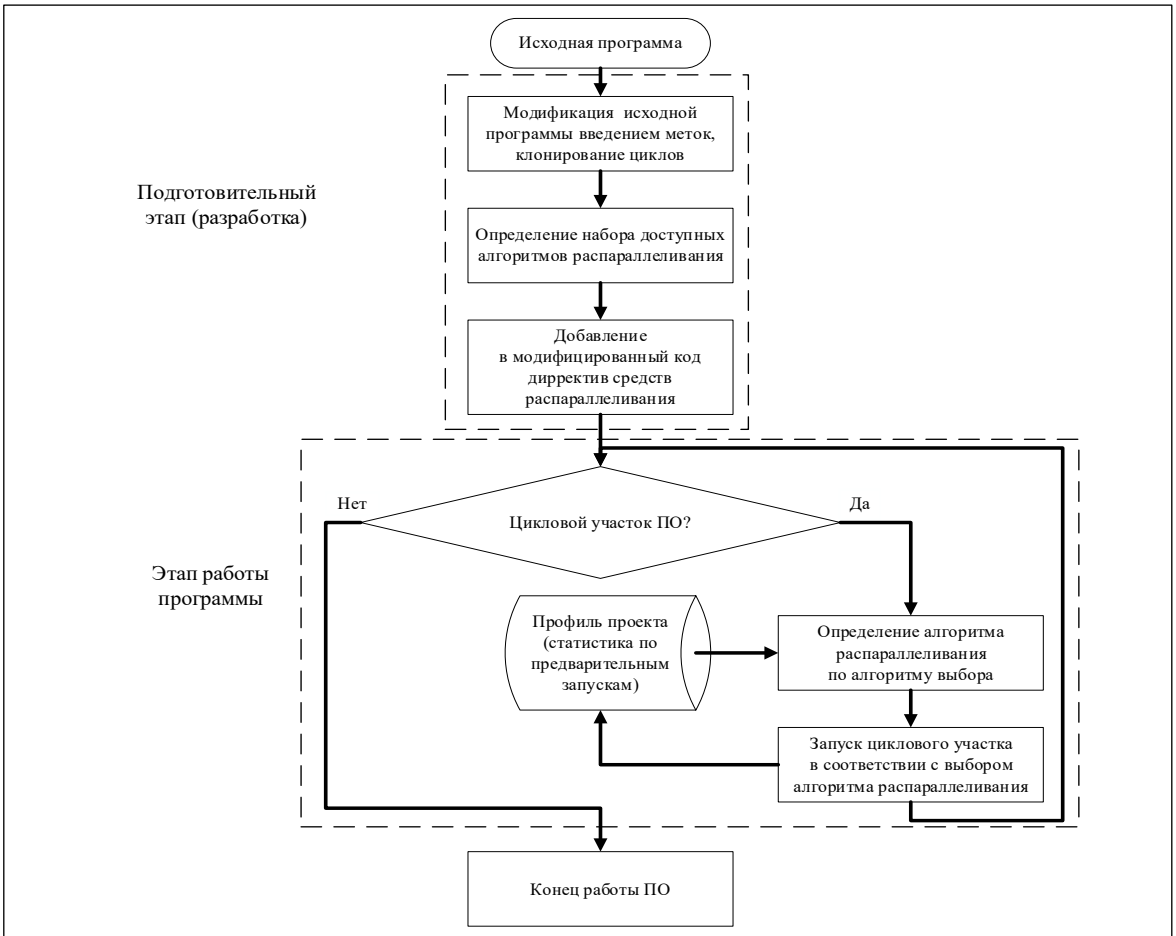

Рис. 5. Схема реализации метода

Fig. 5. A method implementation scheme

параллеливания будут существенно отличаться. Поэтому оптимизация выбора ветки (клона) программы позволяет снизить временные затраты на выполнение цикла и программы в целом. Работа автоматизирована, по входным параметрам циклического участка целевой программы (номера цикла и количества итераций в цикле) происходит выбор предпочтительного алгоритма распараллеливания.

Реализация предполагает два этапа. Первый этап – подготовительный, необходим для загрузки кода целевой программы, исходных данных и библиотек алгоритмов распараллеливания, а также выполнения тестового прогона целевой программы и наполнения начальной статистики. Второй этап – непосредственно работа устройства с автоматическим выбором предпочтительного алгоритма распараллеливания под каждый циклический участок целевой программы.

Апробация предлагаемого метода и имитационные эксперименты были проведены на базе Военной академии воздушно-космической обороны (г. Тверь) с использованием многофункционального МК ВН «Небосвод 2.0», позволяющего моделировать боевые действия противоборствующих группировок в воздушно-космической сфере и оценивать их результаты в едином интерактивном пространственно-временном контуре управления. Программные средства комплекса разработаны в программно-технической среде Microsoft Visual Studio 2015 (язык программирования C++, СУБД PostgreSQL) [11].

В качестве варианта построения программных средств МК ВН можно привести пример внедрения технологических решений в универсальную программную архитектуру имитационно-моделирующей системы «ИМС 2.0» (см. http://www.swsys.ru/uploaded/image/2022-1/2022-1-dop/8.jpg) [12]. Это позволит организовать дальнейшие разработки (модернизацию) перспективных программных средств на единой методической и технологической основе с учетом их использования по назначению.

Заключение

Таким образом, представленный метод создания параллельных ПС МК ВН дает возможность комбинировать алгоритмы распараллеливания и при необходимости комплексно применять их для каждого распараллеливаемого цикла, что позволяет сократить время выполнения ПС МК ВН, то есть время имитационного моделирования отражения удара противника с использованием программных средств моделирования автоматизированных систем военного назначения.

Научная новизна заключается в том, что разработанный метод за счет выбора предпочтительного алгоритма распараллеливания под каждый циклический участок программы с учетом времени их выполнения позволит минимизировать время выполнения ПС МК ВН, сократив таким образом время моделирования.

В предлагаемом методе установлена функциональная связь между количеством итераций текущего цикла и предпочтительным алгоритмом распараллеливания. Автоматический выбор предпочтительного алгоритма распараллеливания производится на основе классификационного отбора по входному параметру (количеству итераций) цикла в сформированном профиле проекта, состоящем из совокупности кортежей предпочтительных алгоритмов распараллеливания.

Метод может быть использован при разработке ПС МК ВН, применяемых при моделировании боевых действий в воздушно-космической сфере.

Литература

1. Лясковский В.Л., Ризаев Р.Н., Допира Р.В., Кучеров Ю.С. Методика оценки степени влияния модели системы управления на эффективность применения частей и подразделений радиоэлектронной борьбы в моделирующих комплексах военного назначения // Вопросы радиоэлектроники. 2020. № 10. С. 46–52. DOI: 10.21778/2218-5453-2020-10-46-52.

2. Волконский В.Ю., Брегер А.В., Бучнев А.Ю., Грабежной А.В. и др. Методы распараллеливания программ в оптимизирующем компиляторе // МЦСТ. URL: http://www.mcst.ru/metody-rasparallelivaniya-programm-v-optimiziruyushhem-kompilyatore (дата обращения: 10.07.2021).

3. Биллиг В.А. Параллельные вычисления и многопоточное программирование. М.: ИНТУИТ, 2016. 310 с.

4. Самхарадзе К.К., Ерошенко Я.Б. Сравнительный анализ эффективности применения технологий параллельного программирования // Научный аспект. 2018. Т. 2. № 2. С. 223–236.

5. Керп А.С. Сравнение технологий распараллеливания вычислений на примере решения задачи о кручении стержня прямоугольного сечения // XVIII Всерос. конф. молодых ученых по матем. моделированию и информационным технологиям. 2017. С. 8–9.

6. Ханкин К.М. Сравнение эффективности технологий OpenMP, nVidia CUDA и StarPU на примере умножения матриц // Вестн. Южно-Уральского гос. ун-та. 2013. № 1. С. 34–41.

7. Вентцель Е.С. Теория вероятностей. М.: Юстиция, 2018. 658 с.

8. Голик Ф.В. Аппроксимация эмпирических распределений вероятностей полиномами Бернштейна // Журнал радиоэлектроники. 2018. № 7. URL: http://jre.cplire.ru/jre/jul18/5/text.pdf (дата обращения: 10.08.2021). DOI: 10.30898/1684-1719.2018.7.5.

9. Шлее М. Qt 5.10. Профессиональное программирование на C++. СПб: БХВ-Петербург, 2018. 1072 с.

10. Моргунов Е.П. PostgreSQL. Основы языка SQL. СПб: БХВ-Петербург, 2018. 336 с.

11. Гончаров А., Костров С, Бегларян С. Чем обусловлено создание нового тренажно-моделирующего комплекса // ВКС. 2017. № 4. С. 64–70.

12. Ляпин В.Р., Барвиненко В.В. Единая информационно-моделирующая среда в системах военного назначения // Военная мысль. 2015. № 4. С. 72–78.

Software & Systems
DOI: 10.15827/0236-235X.137.083-094

Received 21.07.21, Revised 30.09.21
2022, vol. 35, no. 1, pp. 083–094

The method for creating parallel software tools for modeling military complexes

M.A. Aksenov [1], *Adjunct, aksen1985@mail.ru*

[1] *Military Academy of the Aerospace Defence, Tver, 170100, Russian Federation*

Abstract. Nowadays modeling systems are actively created and used all over the world including the Armed Forces of the Russian Federation. The basis of these systems are modeling complexes, which are a set of technical and software tools providing calculations and imitation modeling.

The analysis of modern software tools for modeling military complexes has shown that the duration of the calculations performed during imitation largely influence the efficiency of their application when used directly.

Specific technological tools used in the development of parallelization of labor-intensive cyclic sections of modeling complexes allow minimizing the time spent on modeling under conditions of limited terms of using software tools. However, nowadays they are not implemented in the general software architecture of modeling complexes accepted for supply in the Armed Forces of the Russian Federation.

The paper considers the issues of choosing parallelization algorithms implemented in parallel software development tools for multi-core (multiprocessor) shared memory computing systems.

The purpose of the paper is to assess the impact of the execution time of parallelized cyclic sections of a target program with multithreaded parallel execution of the program in multi-core (multiprocessor) PCs on the results of combat imitation. The scientific novelty is in the development of a new method for creating parallel software tools for modeling military complexes.

The paper provides numeric examples of calculations in the Mathcad. To avoid errors in choosing preferred parallelization algorithms, the entire analysis is based on mathematical statistics elements with the probability of a confidence interval for estimating the cycle execution time by a certain algorithm considering the upper limit of the confidence interval. The author proposes a variant of constructing software tools on the example of introducing technological developments into a software architecture of a modeling complex.

Keywords: software tool, modeling complex, parallelization algorithm, cycle, number of iterations, execution time.

References

1. Lyaskovsky V.L., Rizaev R.N., Dopira R.V., Kucherov Yu.S. Methodology for assessing the influence of control system on efficiency of application of electronic warfare units in military modeling complexes. *Issues of Radio Electronics*, 2020, no. 10, pp. 46–52. DOI: 10.21778/2218-5453-2020-10-46-52 (in Russ.).

2. Volkonskiy V., Breger A., Buchnev A., Grabezhnoy A. et al. Program parallelization methods implemented in optimizing compiler. *MCST*. Available at: http://www.mcst.ru/metody-rasparallelivaniya-programm-v-optimiziruyushhem-kompilyatore (accessed July 10, 2021) (in Russ.).

3. Billig V.A. *Parallel Computing and Multithreaded Programming*. Moscow, 2016, 310 p. (in Russ.).

4. Samkharadze K.K., Eroshenko Ya.B. Comparative analysis of the effectiveness of the use of parallel programming technologies. *Scientific Aspect*, 2018, vol. 2, no. 2, pp. 223–236 (in Russ.).

5. Kerp A.S. A comparison of computational parallelization technologies by the example of solving the problem of rectangular bar torsion. *Proc. 18th All-Russ. Conf. of Young Scientists in Math. Modelling and IT*, 2017, pp. 8–9 (in Russ.).

6. Khankin K.M. Efficiency comparison of OpenMP, nVidia CUDA and StarPU technologies by the example of matrix multiplication. *Bull. of the South Ural State University*, 2013, no. 1, pp. 34–41 (in Russ.).

7. Venttsel E.S. *Probability Theory*. Moscow, 2018, 658 p. (in Russ.).

8. Golik F.V. Approximation of the empirical probability distributions by Bernstein polynomials. *Journal of Radio Electronics*, 2018, no. 7. Available at: http://jre.cplire.ru/jre/jul18/5/text.pdf (accessed August 10, 2021) (in Russ.). DOI: 10.30898/1684-1719.2018.7.5.

9. Shlee M. *Qt 5.10 Professional Programming on C++*. St. Petersburg, 2018, 1072 p. (in Russ.).

10. Morgunov E.P. *PostgreSQL. The Basics of the SQL Language*. St. Petersburg, 2018, 336 p. (in Russ.).

11. Goncharov A., Kostrov S., Beglaryan S. What caused the creation of a new training and modeling complex. *Aerospace Sphere J.*, no. 4, pp. 64–70 (in Russ.).

12. Lyapin V.R., Barvinenko V.V. Unified information-modeling environment in military systems. *Military Thought*, 2015, pp. 72–78 (in Russ.).

Для цитирования

Аксенов М.А. Метод создания параллельных программных средств моделирующих комплексов военного назначения // Программные продукты и системы. 2022. Т. 35. № 1. С. 083–094. DOI: 10.15827/0236-235X.137.083-094.

For citation

Aksenov M.A. The method for creating parallel software tools for modeling military complexes. *Software & Systems*, 2022, vol. 35, no. 1, pp. 083–094 (in Russ.). DOI: 10.15827/0236-235X.137.083-094.