Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (05.13.11)

УДК 519.716.35
DOI: 10.24160/1993-6982-2020-3-102-110

Алгоритмические параллельные процессы и их сложность

В.П. Кутепов, В.Н. Фальк

Предложен метод оценки сложности параллельных процессов по критериям среднего времени их абсолютно параллельного выполнения и необходимых для этого ресурсов (количеству узлов выполняющей процесс системы). Указанные критерии рассматриваются для достаточно общего языка параллельных процессов, по их значениям можно судить, насколько реальные их значения, полученные при выполнении процесса на конкретной системе, отличаются от предельно возможных.
Важно отметить, что рассматриваемый язык параллельных процессов реализован на многоядерных компьютерах, на его основе разработаны операционные средства для эффективного управления выполнением функциональных параллельных программ. В качестве языка описания параллельных программ взят созданный и успешно применяемый на практике язык функционального параллельного программирования. Описанные в статье методы оценки сложности параллельных процессов необходимы на стадии проектирования и оптимизации функциональных параллельных программ.

Ключевые слова: параллельные процессы, оценка сложности.

Для цитирования: Кутепов В.П., Фальк В.Н. Алгоритмические параллельные процессы и их сложность // Вестник МЭИ. 2020. № 3. С. 102—110. DOI: 10.24160/1993-6982-2020-3-102-110.

Parallel Algorithmic Processes and Their Complexity

V.P. Kutepov, V.N. Falk

A method for estimating the complexity of parallel processes according to the criteria of the average time of their absolutely parallel execution and the resources required for executing them (the number of nodes of the system performing the process) is proposed. These criteria are considered for a fairly general language of parallel processes, and their values can be used to judge how their real values obtained during the process on a particular system differ from their ultimately possible levels.
It is important to note that the considered language of parallel processes has been implemented on multi-core computers, and the operating means for effectively managing the execution of functional parallel programs have been developed on its basis. The functional parallel programming language that has been developed and successfully applied in practice is used as a language for describing parallel programs. The described methods for estimating the complexity of parallel processes are necessary at the stage of designing and optimizing functional parallel programs.

Key words: parallel processes, complexity estimation.

For citation: Kutepov V.P., Falk V.N. Parallel Algorithmic Processes and Their Complexity. Bulletin of MPEI. 2020;3:102—110. (in Russian). DOI: 10.24160/1993-6982-2020-3-102-110.

Введение

Понятие процесса (рассматриваются дискретные процессы) уточняется как наблюдаемое частично-упорядоченное следование во времени его актов (неделимых действий), подчиненных определенным причинно-следственным отношениям. Применительно к системам и физическим явлениям протекающие в них процессы обычно определяются через понятие состояния и правила изменения состояния. Для описания и исследования различных классов процессов создаются специальные модели и языки. Однако, чтобы получить полную информацию об интересующем классе процессов, следует рассматривать триаду: организацию и алгоритм выполнения работ, язык, на котором процессы с теми или иными ограничениями могут быть описаны, и возможные варианты их практической реализации

на предполагаемых для использования системах. При этом время выполнения и используемые для этого ресурсы системы — основные критерии сложности процессов и их способности выполнять соответствующие работы и алгоритмы. Проблема достижения определенного времени выполнения предполагает процедуру распараллеливания на трех указанных стадиях.

Именно параллелизм с его различными формами существенно усложняет работу по созданию соответствующих языков для их описания и эффективной реализации процессов на системах по сравнению, например, с последовательным программированием [1 — 5].

Понятия акта и актора процесса, средства группирования актов, формы динамического порождения при их выполнении — главные конституенты, определяющие особенности моделей и языков процессов [6]. В то же время на самом нижнем уровне выполнение процесса можно представить как изменяемое во времени множество одновременно выполняемых и взаимодействующих между собой актов.

Эта абстракция реализована в средах параллельного программирования PVM, MPI [7, 8]. В средах ERLANG, MULTITHREADING, FPTL, HOPE, Haskell имеется набор примитивов, предназначенных для группирования процессов, их описания и определений рекурсивно порождаемых процессов [5, 7 — 12].

Процессы часто ассоциируются с функционированием систем, и в серии статей [13] эта взаимосвязь обсуждается с разных сторон. В частности, дано формальное определение состояния, введено естественное ограничение на правила изменения состояний, которые могут происходить только на основе информации о поведении процесса в прошлом [14], рассмотрена обобщенная модель динамического порождения процессов и реконфигурирования системы на основе теории самовоспроизведения [15]. Эти основополагающие результаты не утратили своей актуальности при исследовании процессов и систем.

Различные варианты формализации дискретных параллельных процессов получили свою, в определенном смысле, завершенную форму в работах [16, 17]. В отличие от моделей последовательных процессов, при построении которых используются бинарные операции последовательной композиции, операции выбора по условию продолжения процесса и рекурсивные определения процессов или определения их с помощью циклов (частный случай правосторонней рекурсии) в моделях процессов [16, 17] добавлена операция параллельной композиции. Она трактуется как произвольное чередование актов процессов при их выполнении, а операция выбора интерпретируется как операция случайного выбора одного из соединяемых ею процессов.

В [6] предложена версия языка параллельных процессов, позволяющего описывать более широкий класс параллельных процессов, которые, в частности, порождаются при выполнении сложных рекурсивных параллельных программ [5, 18]. Он более точно отображает временной фактор, возможность выполнения актов процесса с упреждением и др. [19].

Сети Петри — еще одна популярная модель описания дискретных процессов и систем [20]. Она интересна двумя факторами: наглядным графическим представлением описываемого процесса в виде сети и трактовкой акта процесса как элемента со многими входами, принимающего инициализирующие акт сигналы от других актов, и многими выходами, по которым акт после своего завершения воздействует на другие акты. Можно сказать, что двумерная графическая форма представления процессов более выразительна по сравнению с их одномерным текстовым представлением, позволяет часто избежать повторное вхождение одного и того же по смыслу акта в описании процесса [6].

Основная цель настоящей работы — предложение конструктивных методов оценки сложности параллельных процессов по критериям времени их параллельного выполнения и необходимых для этого ресурсов системы. Это позволит разработчику процессов и систем оценить границу между предельными и реально достижимыми значениями этих критериев.

Описан вариант языка параллельных процессов из [6] и определены процессная семантика рекурсивно определенных процессов и множество траекторий выполнения процесса, по которым определяется сложность процесса.

Кратко рассмотрена проблема однозначной идентификации динамически порождаемых процессов, которая важна при реализации процессов на системах.

Представлены методы оценки среднего времени абсолютно параллельного выполнения процессов и требуемые для этой цели ресурсы (количество узлов системы).

Указаны ждущие решения проблемы.

Язык параллельных процессов

Описываемый язык параллельных процессов основан на языке функционального параллельного программирования FPTL (Functional Parallel Typified Language) [5] и модели параллельных процессов, определённой в [6].

Модели параллельного выполнения программ на данных языках являются достаточно общими, и известные модели параллельных процессов представляют их частный случай [16, 17]. Это означает, что любой процесс, описываемый на указанных языках, имеет эквивалентное представление на предлагаемом языке параллельных процессов [6] с сохранением всех возможностей его параллельного выполнения. Более того, многие формы параллелизма [1, 2], выражаемые средствами этого языка, не имеют адекватного представления на языках процессов [16, 17].

Формальное определение языка

Пусть АСТ = $A \cup P \cup X \cup \{\text{stop}\}$ — не более чем счетное множество актов, где A — множество константных актов, обозначаемых a_i, $i = 1, 2, \ldots$; P — множество пар ортогональных актов-условий, обозначаемых $(p_i, \overline{p}_i)$, $i = 1, 2, \ldots$; X — множество переменных актов, обозначаемых x_i, $i = 1, 2, \ldots$; stop — акт останова процесса.

Процесс на множестве актов АСТ определяется в виде системы в общем случае рекурсивных уравнений $x_i = \tau_i$, $i = 1, 2, \ldots, n(*)$, где $x_i \in X$; τ_i — процессный терм, представляющий композицию актов, полученную посредством следующих операций композиции: $\bullet$, $*$, $\oplus$, $\rightarrow$.

Терм определяется индуктивно.

Каждый акт $a \in A \cup X$ — терм.

Если τ_1, τ_2 — термы, то $(\tau_1 \bullet \tau_2)$ и $(\tau_1 * \tau_2)$ — термы.

Если τ_1, τ_2 — термы, и p, $\overline{p}$ — акты множества P, то $((p \rightarrow \tau_1) \oplus (\overline{p} \rightarrow \tau_2))$ и $(((\tau \bullet p) \rightarrow \tau_1) \oplus ((\tau \bullet \overline{p}) \rightarrow \tau_2))$ — термы.

Других термов нет.

Все операции композиции ассоциативны, операции композиции $*$, $\oplus$ — коммутативны. Следующий порядок старшинства операций композиции: $\bullet$, $\rightarrow$, $*$, $\oplus$ позволяет опускать ряд скобок в записи термов.

Данный язык является вариантом языка функционального параллельного программирования FPTL, реализованного на многоядерных компьютерах [5] и языка процессов, предложенного в [6].

В качестве примера приведем описание процесса параллельного выполнения значений функции $Y_1(x)$, определяемой системой рекурсивных функциональных уравнений:

$$Y(x) = \text{if } p(f_1(x), f_2(x)) \text{ then } f_3(x) \text{ else } f_4(Y_1(f_5(x)), f_6(Y_2(x)));$$

$$Y_2(x) = \text{if } p_2(f_6(x)) \text{ then } f_7(x) \text{ else if } p_3(f_8(x)) \text{ then}$$
$$\text{then } f_9(f_{10}(x), f_{11}(x)) \text{ else } Y_2(f_{12}(x)).$$

Процесс абсолютно параллельного выполнения функции $Y_1(x)$, заданной этой системой уравнений, имеет следующее описание:

$$Y_1 = (f_1 * f_2) \bullet p_1 \rightarrow f_3 \oplus (f_1 * f_2) \bullet \overline{p_1} \rightarrow (f_5 \bullet Y_1 * Y_2 \bullet f_6) \bullet f_4;$$

$$Y_2 = f_6 \bullet p_2 \rightarrow f_7 \oplus f_6 \bullet \overline{p_2} \rightarrow$$

$$\rightarrow \left(f_8 \bullet p_3 \rightarrow (f_{10} * f_{11}) \bullet f_9 \oplus f_8 \bullet \overline{p_3} \rightarrow (f_{12} \bullet Y_2)\right).$$

Сетевое представление процессов

Для формулирования правил параллельного выполнения процессов введём модель их сетевого представления.

Сеть любого акта $a \in$ АСТ имеет представление, изображённое на рис. 1, *а*.

Если τ_1, τ_2 — термы, то композиции $(\tau_1 \bullet \tau_2)$ и $(\tau_1 * \tau_2)$ имеют сетевое представление на рис. 1, *б*, *в*.

Сетевое представление терма $(\tau \bullet p \rightarrow \tau_1) \oplus (\tau \bullet \overline{p} \rightarrow \tau_2)$ дано на на рис. 1, *г*. Сетевое представление терма

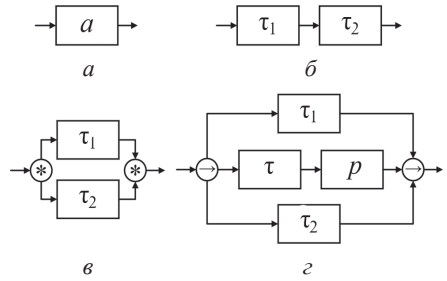

Рис.1. Сетевые представления простейших термов

$p \rightarrow \tau_1 \oplus \overline{p} \rightarrow \tau_2$ отличается от приведённого удалением блока терма τ.

Процесс, заданный в виде системы уравнений $x_i = \tau_i$, $i = 1, 2, \ldots$, имеет представление в виде последовательности сетевых равенств: $x_i = N_i$, $i = 1, 2, \ldots, n$, где N_i — сеть терма τ_i.

На рис. 2 приведён пример сетевого представления, описанного выше процесса.

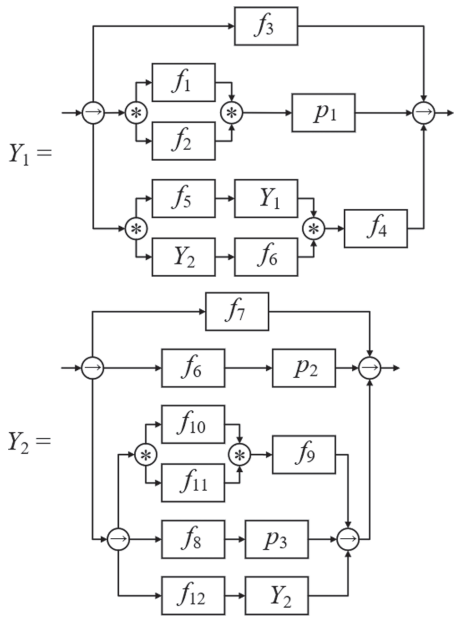

Рис. 2. Сетевое представление уравнений Y_1, Y_2

Правила выполнения процесса

Правила параллельного выполнения процесса на сети формулируются таким образом, чтобы сохранялась возможность абсолютно параллельного выполнения процесса. Предположим, что выполняются следующие условия:

- длительность выполнения любого акта, за исключением акта stop, является конечной и ненулевой, длительность акта stop равна 0;
- инициализация любого акта процесса проходит без задержек по его готовности для выполнения;
- завершение выполнения любого акта, не являющегося переменным, сопровождается посылкой инициализирующих сигналов другим актам, на которые акт непосредственно влияет, то есть по дугам на сети,

ведущим от акта, завершившего выполнение, к другим актам;

● выполнение переменного акта x_i сопровождается порождением нового процесса, описанного термом τ_i, а в сетевой интерпретации подстановкой сети терма τ_i вместо акта x_i и инициализацией выполнения без задержек процесса представленного τ_i;

● выполнение акта условия $p \in P$ завершается выбором одного из двух возможных его значений, обозначаемых true или false, при этом при истинном значении p процесс τ_1 в сетевом представлении терма $(\tau \bullet p \rightarrow \tau_1) \oplus (\tau \bullet \bar{p} \rightarrow \tau_2)$ продолжает свое выполнение, если он ещё не завершён, а процесс, представленный в сети термом τ_2, прерывается. При ложном значении p процесс τ_2 продолжает выполнение, если он не завершился в момент завершения акта p, а процесс, представленный термом τ_1, прерывается;

● парные узлы сети, представляющие пары операций $*$ и $\rightarrow$, выполняют роль управляющих актов. Открывающий узел $*$ после инициализации активизирует одновременное исполнение непосредственно следующих за ним актов по обеим исходящим из него связям на сети, а закрывающий парный ему узел активизируется только по завершении актов на обоих его входах. Открывающий акт операции $\rightarrow$ после своей инициализации ведёт себя аналогично открывающему акту $*$, а парный ему закрывающий акт передает инициализирующий сигнал на выход после завершения акта τ_1 при истинном значении завершения акта p или после завершения акта τ_2 при ложном значении завершения акта p;

● выполнение акта stop завершается без задержек после его инициализации и следующий за stop процесс не может быть инициализирован;

Не нарушая общности, положим, что выполнение процесса, заданного системой уравнений (*), начинается с выполнения акта x_1, инициализирующего выполнение процесса, описанного τ_1.

Каждая конкретная реализация или траектория выполнения процесса, основанного на перечисленных правилах, определяется функциями: φ_1:АСТ$\times T \rightarrow T$ и φ_2:$P \times T \rightarrow$ [0 — 1] (замкнутый интервал действительных чисел), где φ_1 определяет для каждого акта в момент его инициализации длительность его выполнения (T — заданная шкала временных отсчётов с отношениями $<$, $\leq$, $>$, $\geq$ и $=$ на ней), а φ_2 задаёт вероятность в момент t завершения выполнения акта $p \in P$ принимать значение «истина» или «ложь», причём $\varphi_2(\bar{p}, t) = 1 - \varphi_2(p, t)$.

В реальности длительность выполнения актов процесса — случайная величина. Простой пример этого — длительность выполнения арифметических операций компьютера, зависящая от распределения единиц и нулей в мантиссах чисел, для которых выполняется арифметическая операция. Поэтому полное представление о поведении процесса можно получить, изучая множество всех траекторий выполнения процесса (историй, протоколов, трасс — эквивалентные термины), каждая из которых — следствие реализованных функций φ_1 и φ_2.

Таким образом, о характере поведения процесса PR можно судить, рассматривая функцию φ, порождающую множество всех траекторий его выполнения $TR = \varphi(PR, \varphi_1, \varphi_2)$ для заданного множества пар в общем случае случайных функций φ_1 и φ_2.

Множество всех возможных траекторий $TR = \varphi(PR, \varphi_1, \varphi_2)$ выполнения процесса PR конструктивно порождаемо. Для этого в [6] введена автоматная модель параллельного выполнения процесса, в которой состояние процесса определено как множество всех актов, одновременно выполняющихся на некотором шаге (в некоторый момент времени при отображении выполнения процесса на ось реального времени). Условием перехода из S_i в состояние S_{i+1}, $i = 1, 2, \ldots$, является завершение одного или одновременно нескольких актов в момент t, выполняемых в состоянии S_i. Множество актов в состоянии S_{i+1} содержит не завершившие выполнение акты в состоянии S_i и все акты процесса, которые получают право на выполнение после завершивших выполнение актов в соответствии с функцией φ_2 для условных актов.

Определение. Траекторией процесса для заданных функций φ_1, φ_2 назовём в общем случае неограниченную последовательность $S_1, S_2, \ldots$ следования состояний при выполнении процесса, начальным состоянием которого является выполнение x_1 в системе уравнений (*), описывающей процесс.

Анализируя все возможные отношения «раньше» или «одновременно» между временем завершения подмножества актов в состояниях, определённых функцией φ_1, и все возможные сочетания значений «истина» и «ложь», присваиваемых завершившим выполнение актам-условиям функцией φ_2, можно определить множество всех возможных последующих состояний процесса. При повторении этой процедуры для каждого состояния конструктивно воспроизведутся все траектории выполнения процесса.

На рисунке 3 приведен простой пример, иллюстрирующий построение всех возможных траекторий абсолютно параллельного выполнения процесса $x_1 = (p \rightarrow a_1 \oplus \bar{p} \rightarrow a_2 \bullet a_3) \bullet$ stop. Завершившие выполнение акты указаны на переходах между состояниями. Рассматривается случай, когда длительность акта, выполняемого в различных состояниях, может быть произвольной.

Отметим, что после завершения акта-условия в любом состоянии процесса в зависимости от его значения «истина» или «ложь», прерывается выполнение всех актов в этом состоянии, которые были ранее активизированы и не влияли на продолжение выполнения процесса.

Перейдем к математической трактовке процессов, определенных в виде системы процессных уравнений

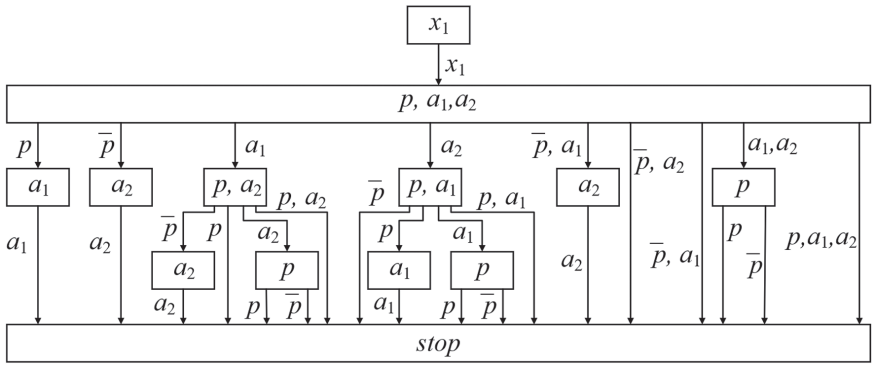

Рис. 3. Множество траекторий процесса

(∗), которая должна дать ответ на вопрос, что является решением для каждого x_i, $i = 1, 2, \ldots, k$ в этой системе.

Определим последовательность процессов $x_i^{(n)}$ для каждого $i = 1, 2, \ldots, k$ и $n = 0, 1, \ldots$, положив $x_i^{(0)} = \text{stop}$, $x_i^{(n+1)} = \left[x_j^{(n)} \middle/ x_j \middle| j = 1,2,\ldots,k \right] \tau_i$, где в правой части указан результат одновременной подстановки $x_j^{(n)}$, $j = 1, 2, \ldots, k$, вместо всех вхождений x_j в τ_i.

Очевидно, что множество всех траекторий $x_i^{(n)}$ для любого n конечно.

Определение. Введем отношение $\leq$ строгого порядка на множестве траекторий процессов PR_1 и PR_2, полагая $TR(PR_1) \leq TR(PR_2)$, если для каждой траектории $tr \in TR(PR_1)$ существует траектория $tr' \in TR(PR_2)$ такая, что tr является начальным отрезком траектории tr' (префиксом последовательности состояний tr').

Здесь $TR(PR)$ — множество траекторий процесса для заданных функций φ_1 и φ_2; $TR(\text{stop})$ — пустое множество.

Утверждение. Для любого $n \geq 0$ выполняется $TR(x_i^{(n)}) \leq TR(x_i^{(n+1)})$, $i = 1, 2, \ldots, k$.

Следствие 1. Последовательность $TR(x_i^{(n)})$ для $n = 0, 1, \ldots$, образует цепь, наименьшим элементом которой является пустое множество. Цепь имеет в качестве предела наименьшую верхнюю грань $\lim\limits_{n\to\infty} TR\left(x_i^{(n)}\right)$, $i = 1, 2, \ldots, k$.

Согласно известному результату Д. Скотта, данный предел — наименьшее (относительно операции $\subseteq$ включения множеств) решение системы процессных уравнений (∗).

Данное утверждение остается в силе, если одновременную подстановку в определении $x_i^{(n+1)}$, $n = 0, 1, \ldots$, заменить на последовательное выполнение в произвольном порядке подстановок $x_j^{(n)}$ вместо x_j в терм τ_i.

Следствие 2. Процесс, реализуемый в соответствии с описанными правилами его выполнения, порождает для каждого x_i, $i = 1, 2, \ldots, k$ в качестве начального состояния множество траекторий, являющееся наименьшим решением для x_i для системы процессных уравнений.

Отношение включения, определенное на множестве траекторий процессов, позволяет формально сравнивать их процессные возможности и говорить об их эквивалентности. Ранее выполнено определение множества траекторий процесса при анализе всех возможных отношений раньше или одновременно между моментами завершения выполнения актов в каждом состоянии процесса. В реальности время выполнения актов процесса часто заранее фиксировано, что уменьшает множество его траекторий.

Процессы, как и программы, разрабатываются для определенной цели, и при их создании главные усилия сосредоточены на поиске такой семантически эквивалентной формы, которая сможет обеспечить необходимую степень параллелизма и эффективность (время выполнения процесса и используемые ресурсы). Разработанный язык функционального параллельного программирования реализован на многоядерных компьютерах и имеет средства приведения программы путем её эквивалентных преобразований к эквивалентной форме с большей или меньшей степенью параллелизма в процессе его выполнения [5]. Самое интересное в нём то, что рассматриваемые операции композиции процессов изначально были введены как операции композиции функций, что чрезвычайно важно, когда мы пытаемся определить семантическое значение изучаемых процессов.

Анализируя процессы, связанные с физическими явлениям, мы вынуждены иметь дело с реальным, тем или иным способом и определенной точностью измеряемым временем. При этом условие инициализации акта процесса в общем случае зависит от отношения между моментами времени поступления воздействующих на него «сигналов». Для формализации этого необходимо существенное расширение языка процессов, связанное с заданием иной семантики процесса [6].

Однозначное именование инициализируемых актов

Еще один важный аспект процессной реальности связан с однозначным именованием порождаемых актов при выполнении процесса. Во-первых, акт с одним и тем же именем может несколько раз входить в термы τ_i в описании процесса (∗). Хотя такие акты рассматриваются в предполагаемой их интерпретации как эквивалентные, в процессе они считаются различны-

ми. Операция инициализации терма τ_i при готовности для выполнения переменного акта x_i также приводит к активизации актов, имена которых совпадают с именами выполняемых актов. Формально можно разными способами ввести однозначное именование подобного рода актов. Однако в реализации процессов на практике, например при выполнении параллельных программ на компьютерной системе, при решении проблемы однозначного именования инициализируемых актов время и требуемая память являются основными критериями оптимальности. В реализации языка граф-схемного потокового программирования на компьютерных системах [18] каждому инициализируемому акту присваивается уникальный идентификатор, представляющий собой пару: номер узла, где порождается акт, и уникальный номер (натуральное число), фиксируемый в специальной таблице номеров для исключения повторений. Для уменьшения размера таблицы после завершения выполнения акта его номер считается доступным для нового использования. Отметим, что в системах параллельного программирования PVM и ERLANG алгоритмы однозначного именования процессов реализованы, однако вопрос об их оптимальности остался открытым и более точно не рассматривался [7, 9]

Сложность параллельных процессов

Основными критериями сложности параллельных процессов являются время выполнения и требуемые ресурсы для достижения этого времени. Пусть задано константное время выполнения каждого акта процесса, определяемое функцией φ_1:АСТ → Т, и вероятность акта-условия $p \in P$ принимать истинное или ложное значение, заданное функцией φ_2:$P \to [0; 1]$.

Временная сложность процессов

Для всякого процесса (при заданной длительности актов) множество траекторий его выполнения однозначно определяется только возможностями вероятности актов-условий принимать истинное или ложное значение. В термах $(p \to \tau_1) \oplus (\bar{p} \to \tau_2)$ и $(\tau \bullet p \to \tau_1) \oplus (\tau \bullet \bar{p} \to \tau_2)$ оба слагаемых являются ортогональными процессами в том смысле, что только один из них может завершиться успешно. Последнее зависит от того, истинное или ложное значение принимает акт-предикат p. Если известна длительность всех актов процесса и вероятность $\varphi_2(p)$ принятия истинного значения для предиката p при выполнении процесса, то среднее время выполнения рассматриваемых термов определяется функцией t:

$$t(p \to \tau_1 \oplus \bar{p} \to \tau_2) = \varphi_2(p) \times \max\{\varphi_1(p), t(\tau_1)\} +$$
$$+\varphi_2(\bar{p}) \times \max\{\varphi_1(\bar{p}), t(\tau_2)\};$$

$$t(\tau \bullet p \to \tau_1 \oplus \tau \bullet \bar{p} \to \tau_2) = \varphi_2(p) \times$$
$$\times \max\{t(\tau) + \varphi_1(p), t(\tau_1)\} +$$
$$+\varphi_2(\bar{p}) \times \max\{t(\tau) + \varphi_1(\bar{p}), t(\tau_2)\}.$$

Существует процедура эквивалентного преобразования любого процесса, сохраняющего среднее время выполнения любого терма τ, в форму $\tau = \tau_1 \oplus \tau_2 \oplus \ldots \oplus \tau_m$ такую, что термы в этом разложении попарно ортогональны и не содержат вхождений операции ⊕.

Следующие аксиомы эквивалентности термов из [5] позволяют выполнить указанное преобразование:

$$\tau \bullet (\tau_1 \oplus \tau_2) = \tau \bullet \tau_1 \oplus \tau \bullet \tau_2;$$
$$(\tau_1 \oplus \tau_2) \bullet \tau = \tau_1 \bullet \tau \oplus \tau_2 \bullet \tau;$$
$$\tau * (\tau_1 \oplus \tau_2) = \tau * \tau_1 \oplus \tau * \tau_2;$$
$$(\tau_1 \oplus \tau_2) * \tau = \tau_1 * \tau \oplus \tau_2 * \tau;$$
$$\tau \to (\tau_1 \oplus \tau_2) = \tau \to \tau_1 \oplus \tau \to \tau_2;$$
$$(\tau_1 \oplus \tau_2) \to \tau = \tau_1 \to \tau \oplus \tau_2 \to \tau.$$

Приведём пример преобразования терма

$$\tau = a_1 \bullet p_1 \to (p_2 \to a_2 \oplus \bar{p}_2 \to a_3) * a_4 \oplus$$
$$\oplus a_1 \bullet \bar{p}_1 \to a_5 * (p_3 \to a_6 \oplus \bar{p}_3 \to a_2) \bullet a_7$$

к эквивалентной форме ортогонального разложения:

$$\tau = a_1 \bullet p_1 \to (p_2 \to a_2) * a_4 \oplus a_1 \bullet p_1 \to (\bar{p}_2 \to a_3) *$$
$$*a_4 \oplus a_1 \bullet \bar{p}_1 \to a_5 * (p_3 \to a_6) \bullet a_7 \oplus a_1 \bullet \bar{p}_1 \to a_5 *$$
$$*(\bar{p}_3 \to a_2) \bullet a_7.$$

Очевидно, что среднее время выполнения терма τ, приведённого к эквивалентной форме ортогонального разложения $\tau = \tau_1 \oplus \tau_2 \oplus \ldots \oplus \tau_m$, определяется в виде $t(\tau) = q(\tau_+) \times t(\tau_1) + q(\tau_2) \times t(\tau_2) + \ldots + q(\tau_m) \times t(\tau_m)$, где $q(\tau)$ — вероятность успешного завершения процесса, представленного термом τ. Эта вероятность — произведение вероятностей всех входящих в τ актов-условий (предикатов) при условии, что они получат значение «истина».

Следующие правила позволяют вычислить среднее время выполнения терма τ, не содержащего вхождений операции ⊕, где $t(\tau)$ — время выполнения терма τ .

Если $\tau = a_1$, $a_1 \in A$, $t(\tau) = \varphi_1(a_1)$.

Если $\tau = \tau_1 \bullet \tau_2$, $t(\tau) = t(\tau_1) + t(\tau_2)$.

Если $\tau = \tau_1 * \tau_2$ или $\tau = \tau_1 \to \tau_2$, $t(\tau) = \max\{t(\tau_1), t(\tau_2)\}$.

Если τ = stop, $t(\tau) = 0$.

Возвращаясь к общей форме задания процесса в виде системы уравнений (∗), для любого n среднее время выполнения $x_i^{(n)}$, $i = 1, 2, \ldots, k$, можно определить, следуя описанной процедуре. Среднее время выполнения процесса для $x_i^{(n)}$, $i = 1, 2, \ldots, k$ можно найти очевидным способом — $\lim\limits_{n \to \infty} t(x_i^{(n)})$. Для заданной точности вычисления ε следует использовать стандартную в таких случаях процедуру вычисления, применяя ме-

тод последовательных приближений и контролируя её окончание по формуле

$$\left| t\left(x_i^{(n+1)}\right) - t\left(x_i^{(n)}\right) \right| \le \varepsilon.$$

Для уравнений с правосторонней рекурсией, т. е. сводимых к циклическим определениям, среднее время можно определить, используя решение систем линейных уравнений.

В качестве примера рассмотрим функцию

$$F(x) = \text{if } p\left(f_1(x), f_2(x)\right) \text{then } f_3(x) \, else \, F\left(f_4\left(f_5(x)\right)\right),$$

вариант процесса параллельного выполнения которой можно описать в виде уравнения с правосторонней рекурсией:

$$x = \left(f_1 * f_2\right) \bullet p \to f_3 \oplus \left(\left(f_1 * f_2\right) \bullet \overline{p} \to f_4 \bullet f_5\right) \bullet x.$$

Определение среднего времени вычисления процесса x сводится к решению линейного уравнения:

$$t(x) = \varphi_2(p) \times \max \begin{Bmatrix} \max\left\{\varphi_1(f_1), \varphi_1(f_2)\right\} + \\ +\varphi_1(p), \varphi_1(f_3) \end{Bmatrix} +$$

$$+ \varphi_2(\overline{p}) \times \max \begin{Bmatrix} \max\left\{\varphi_1(f_1), \varphi_1(f_2)\right\} + \\ +\varphi_1(p), \varphi_1(f_4) + \varphi_1(f_5) \end{Bmatrix} + \varphi_1(x)$$

Заметим, что для языка функционального параллельного программирования [5] данный метод вычисления среднего времени параллельного выполнения программ реализован в виде специального инструмента, используемого при разработке и оптимизации программы.

Приведенная оценка среднего времени выполнения процесса предполагает достаточное количество ресурсов для параллельного выполнения любого множества инициализируемых на разных шагах актов процесса. В реальности при ограниченных ресурсах эта наилучшая оценка времени выполнения процесса позволяет оценить удаленность нахождения реального результата от оптимального.

Есть еще один важный аспект оценки организационной сложности процесса, которая непосредственно влияет на его временную сложность, и, что не менее актуально, на сложность описания процесса, его анализ и оптимизацию. В [21] мы исследовали эту проблему для случая параллельного выполнения функциональных программ, в основе которой лежит анализ количества и «глубины» рекурсивных определений в описании процесса, степени взаимной рекурсивности и другое.

Изучение процессов с учётом их интерпретации, т. е. конкретной работы или задачи, которую они должны выполнять или решать, существенно расширяет круг проблем, с которыми приходится сталкиваться на практике. Это и выбор или построение языка процессов, адекватного решаемой проблеме, и поиск описания процесса, который удовлетворяет временным требованиям и ресурсным ограничениям при его выполнении. Язык функционального параллельного программирования интересен тем [5], что в нём между функциональным описанием и процессом параллельного выполнения соответствующей этому описанию программы существует строго устанавливаемая взаимосвязь. Другими словами, множество всех возможных порождаемых при выполнении программы траекторий конструктивно воспроизводится по её описанию [5], кроме того, существует система эквивалентных преобразований функциональных программ [22, 23], позволяющая приводить программу к максимально параллельной форме.

Ресурсная сложность процессов

Объем ресурсов, необходимых для реализации процесса, еще одна, возможно центральная, проблема, когда мы говорим о реализации процессов на системах. Она формулируется в общей постановке как требование минимизации времени выполнения процесса на заданном количестве ресурсов (компонентов) некоторой системы, её решение базируется на создании эффективных методов управления процессами, главными из которых являются методы планирования процессов и распределения ресурсов [3]. При этом для приближения решения проблемы к реальной практике приходится учитывать специфические особенности поведения процессов, являющихся в общем случае случайными, возможность достаточно точного прогнозирования изменения их характеристик [4].

Введем еще один критерий сложности, по которому можно определить предельные значения ресурсов, необходимых для их абсолютно параллельной реализации.

Для этого используем автоматную модель представления траекторий выполнения процесса, состояниями которых являются множества актов, одновременно выполняемые в определённые моменты времени.

Пусть $S(t)$ — состояние процесса в момент времени t, а $\overline{S}(t)$ – количество одновременно выполняемых актов в этом состоянии.

Определение. Для каждой траектории $TR(x_i^{(n)})$, $n = 0, 1, \ldots$ определим величины, характеризующие максимальное и среднее количество одновременно выполняемых актов в состояниях траектории:

$$\max\left\{\overline{S}(t) | t \in \left[0 \ldots \overline{t}\right]\right\}; \quad 1 \Big/ \overline{t} \int_0^{\overline{t}} \overline{S}(t)\, dt,$$

где $\overline{t}$ — длительность траектории.

Максимальное значение первой величины, вычисляемое относительно всех траекторий процесса, дает информацию о максимальном количестве узлов системы, необходимой для абсолютно параллельной его реализации.

Суммарное значение среднего количества актов для каждой траектории процесса $x_i^{(n)}$, умноженное на вероятность траектории, характеризует среднее значение количества используемых узлов системы. Известно, что эффективность (время и используемые ресурсы) выполнения процесса на системе также существенно зависит от интенсивности обменных взаимодействий между узлами системы. В автоматном представлении выполнения процесса переходы между состояниями взвешены множествами актов, одновременно завершающих выполнение в исходном состоянии, а в новое состояние добавляются инициализируемые после перехода акты процесса. Все эти действия так или иначе определяют нагрузку на каналы и управление системы при выполнении процессов. По вычисляемым аналогичным образом параметрам можно получить данные о нагрузке на каналы и управлении системы при выполнении процессов.

Заключение

Рассмотренные в статье методы оценивания временной и ресурсной сложности параллельных процессов являются «оптимистичными». В действительности при выполнении процессов на реальных системах существенное влияние на время выполнения процесса оказывает ресурсная среда (количество узлов системы и их технические характеристики), время, затрачиваемое на управление процессами и реализацию обменных взаимодействий [3, 4]. Для параллельных программ принципиальным факторами, влияющими на время их выполнения, являются организация программы и степень распараллеливания [1, 2]. Последний фактор воздействует на время выполнения программы. Как показывает практика [5, 19], существует оптимальный уровень распараллеливания, при котором достигается минимальное время выполнения программы на используемой системе с заданным в ней количеством узлов и ядер.

Литература

1. **Кутепов В.П.** О параллелизме с разных сторон // Параллельные вычисления и проблемы управления: Материалы V Междунар. конф. М: Институт проблем управления, 2010. С. 41—52.

2. **Кутепов В.П., Фальк В.Н.** Формы, языки представления, критерии и параметры сложности параллелизма // Программные продукты и системы. 2010. № 3. С. 16—25.

3. **Кутепов В.П.** Интеллектуальное управление процессами и загруженностью в параллельных системах // Известия РАН. Серия «Теория и системы управления». 2007. № 5. С. 58—73.

4. **Кутепов В.П., Бражникова Ю.Н., Горицкий Ю.А., Панков Н.А.** Исследование методов прогнозирования загруженности компьютеров и компьютерных систем // Программные продукты и системы. 2015. № 2. С. 135—147.

5. **Кутепов В.П., Шамаль П.Н.** Реализация языка функционального параллельного программирования FPTL на многоядерных компьютерах // Известия РАН. Серия «Теория и системы управления». 2014. № 3. С. 46—60.

6. **Кутепов В.П.** Модели и языки для описания параллельных процессов // Известия РАН. Серия «Теория и системы управления». 2018. № 3. С. 116—127.

7. **Al Geist e. a.** PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. London: MIT Press, 1994.

8. **Хьюз К., Хьюз Т.** Параллельное и распределенное программирование с использованием C++. М.: Вильямс, 2004.

9. **Cesarini F., Thompson S.** ERLANG Programming: a Concurrent Approach to Software Development. Sebastopol: O'Really Media, 2009.

10. **Carver R.H., Kuo-Chung Tai.** Modern Multithreading: Implementing, Testing, and Debugging Multithreaded Java/C++/Pthreads/Win32. N.-Y.: Wiley-Interscience, 2005.

References

1. **Kutepov V.P.** O Parallelizme s Raznykh Storon. Parallel'nye Vychisleniya i Problemy Upravleniya: Materialy V Mezhdunar. Konf. M: Institut Problem Upravleniya, 2010:41—52. (in Russian).

2. **Kutepov V.P., Fal'k V.N.** Formy, Yazyki Predstavleniya, Kriterii i Parametry Slozhnosti Parallelizma. Programmnye Produkty i Sistemy. 2010;3:16—25. (in Russian).

3. **Kutepov V.P.** Intellektual'noe Upravlenie Protsessami i Zagruzhennost'yu v Parallel'nykh Sistemakh. Izvestiya RAN. Seriya «Teoriya i Sistemy Upravleniya». 2007;5:58—73. (in Russian).

4. **Kutepov V.P., Brazhnikova Yu.N., Goritskiy Yu.A., Pankov N.A.** Issledovanie Metodov Prognozirovaniya Zagruzhennosti Komp'yuterov i Komp'yuternykh Sistem. Programmnye Produkty i Sistemy. 2015;2:135—147. (in Russian).

5. **Kutepov V.P., Shamal' P.N.** Realizatsiya Yazyka Funktsional'nogo Parallel'nogo Programmirovaniya FPTL na Mnogoyadernykh Komp'yuterakh. Izvestiya RAN. Seriya «Teoriya i Sistemy Upravleniya». 2014;3:46—60. (in Russian).

6. **Kutepov V.P.** Modeli i Yazyki dlya Opisaniya Parallel'nykh Protsessov. Izvestiya RAN. Seriya «Teoriya i Sistemy Upravleniya». 2018;3:116—127. (in Russian).

7. **Al Geist e. a.** PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. London: MIT Press, 1994.

8. **Kh'yuz K., Kh'yuz T.** Parallel'noe i Raspredelennoe Programmirovanie s Ispol'zovaniem C++. M.: Vil'yams, 2004. (in Russian).

9. **Cesarini F., Thompson S.** ERLANG Programming: a Concurrent Approach to Software Development. Sebastopol: O'Really Media, 2009.

10. **Carver R.H., Kuo-Chung Tai.** Modern Multithreading: Implementing, Testing, and Debugging Multithreaded Java/C++/Pthreads/Win32. N.-Y.: Wiley-Interscience, 2005.

11. **Burstall R.M., Sannella D.T.** HOPE: An Experimental Applicative Language // Proc. ACM Conf. on LISP and Functional Programming. 1980. P. 136—143.

12. **Марлоу С.** Параллельное и конкурентное программирование на языке Haskell. М.: ДМК Пресс, 2014.

13. **Общая** теория систем. М.: Мир, 1966.

14. **Заде Л.** Понятие состояния в теории систем // Общая теория систем. М.: Мир, 1966, С. 49—65.

15. **Майхилл Дж.** Абстрактная теория самовоспроизведения // Там же. С. 121—140.

16. **Milner R.** A Calculus of Communicating Systems Lecture Notes in Computer Science. N.-Y.: Springer-Verlag, 1980.

17. **Хоар Ч.** Взаимодействующие последовательные процессы. М.: Мир, 1989.

18. **Кутепов В.П., Маланин В.Н., Панков Н.А.** Граф-схемное потоковое параллельное программирование: язык, процессная модель, реализация на компьютерных системах // Известия РАН. Серия «Теория и системы управления». 2012. № 1. С. 67—82.

19. **Кутепов В.П., Зубов М.И.** Реализация и экспериментальное исследование эффективности упреждающего параллелизма // Вестник МЭИ. 2019. № 4. С. 119—126.

20. **Питерсон Дж.** Теория сетей Петри и моделирование систем. М.: Мир, 1984.

21. **Бажанов С.Е., Воронцов М.М., Кутепов В.П., Шестаков Д.А.** Структурный анализ и планирование процессов параллельного выполнения функциональных программ // Известия РАН. Серия «Теория и системы управления». 2005. № 6. С. 111—126.

22. **Кутепов В.П., Фальк В.Н.** Функциональные системы: теоретический и практический аспекты // Кибернетика. 1979. № 1. С. 46—58.

23. **Кутепов В.П.** Исчисление эквивалентности функциональных схем и параллельные алгоритмы // Программирование. 1979. № 6. С. 3—11.

11. **Burstall R.M., Sannella D.T.** HOPE: An Experimental Applicative Language. Proc. ACM Conf. on LISP and Functional Programming. 1980:136—143.

12. **Marlou S.** Parallel'noe i Konkurentnoe Programmirovanie na Yazyke Haskell. M.: DMK Press, 2014. (in Russian).

13. **Obshchaya** Teoriya Sistem. M.: Mir, 1966. (in Russian).

14. **Zade L.** Ponyatie Sostoyaniya v Teorii Sistem. Obshchaya Teoriya Sistem. M.: Mir, 1966:49—65. (in Russian).

15. **Maykhill Dzh.** Abstraktnaya Teoriya Samovosproizvedeniya. Tam zhe:121—140. (in Russian).

16. **Milner R.** A Calculus of Communicating Systems Lecture Notes in Computer Science. N.-Y.: Springer-Verlag, 1980.

17. **Khoar Ch.** Vzaimodeystvuyushchie Posledovatel'nye Protsessy. M.: Mir, 1989. (in Russian).

18. **Kutepov V.P., Malanin V.N., Pankov N.A.** Grafskhemnoe Potokovoe Parallel'noe Programmirovanie: Yazyk, Protsessnaya Model', Realizatsiya na Komp'yuternykh Sistemakh. Izvestiya RAN. Seriya «Teoriya i Sistemy Upravleniya». 2012;1:67—82. (in Russian).

19. **Kutepov V.P., Zubov M.I.** Realizatsiya i Eksperimental'noe Issledovanie Effektivnosti Uprezhdayushchego Parallelizma. Vestnik MEI. 2019;4:119—126. (in Russian).

20. **Piterson Dzh.** Teoriya Setey Petri i Modelirovanie Sistem. M.: Mir, 1984. (in Russian).

21. **Bazhanov S.E., Vorontsov M.M., Kutepov V.P., Shestakov D.A.** Strukturnyy Analiz i Planirovanie Protsessov Parallel'nogo Vypolneniya Funktsional'nykh Programm. Izvestiya RAN. Seriya «Teoriya i Sistemy Upravleniya». 2005;6:111—126. (in Russian).

22. **Kutepov V.P., Fal'k V.N.** Funktsional'nye Sistemy: Teoreticheskiy i Prakticheskiy Aspekty. Kibernetika. 1979; 1:46—58. (in Russian).

23. **Kutepov V.P.** Ischislenie Ekvivalentnosti Funktsional'nykh Skhem i Parallel'nye Algoritmy. Programmirovanie. 1979;6:3—11. (in Russian).

Сведения об авторах:

Кутепов Виталий Павлович — доктор технических наук, профессор кафедры прикладной математики и искусственного интеллекта НИУ «МЭИ», e-mail: kutepov@appmat.ru
Фальк Вадим Николаевич — доктор технических наук, профессор кафедры прикладной математики и искусственного интеллекта НИУ «МЭИ», e-mail: falkvn@yandex.ru

Information about authors:

Kutepov Vitaliy P. — Dr.Sci. (Techn.), Professor of Applied Mathematics and Artificial Intelligence Dept., NRU MPEI, e-mail: kutepov@appmat.ru
Falk Vadim N. — Dr.Sci. (Techn.), Professor of Applied Mathematics and Artificial Intelligence Dept., NRU MPEI, e-mail: falkvn@yandex.ru

Работа выполнена при поддержке: РФФИ (грант № 18-01-00548)
The work is executed at support: RFBR (Grant No. 18-01-00548)

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов
Conflict of interests: the authors declare no conflict of interest

Статья поступила в редакцию: *24.10.2019*
The article received to the editor: *24.10.2019*