УДК 519.87

О. Ф. МИХАЛЬ

МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ СРЕДСТВАМИ ЛОКАЛЬНО-ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ОБРАБОТКИ НЕЧЕТКОЙ ИНФОРМАЦИИ

Одной из перспективных областей исследования и моделирования объектов живой природы применительно к развитию средств информатики и вычислительной техники является распределенная обработка информации. Нейроны головного мозга обмениваются импульсами с *индивидуальной* задержкой, т. е. независимо, определяя моменты выдачи импульсов; клетки организма выполняют дифференцированные функции, взаимно согласуя свое функционирование на уровне химических механизмов взаимодействия; отдельные особи взаимодействуют с окружающей средой в интересах индивидуального выживания и обмениваются сигналами друг с другом в интересах выживания популяции. Приведенные и подобные явления могут рассматриваться как примеры систем распределенной обработки информации. Важным техническим аналогом по этому направлению являются распределенные информационно-управляющие системы (РИУС), состоящие из отдельных пространственно разнесенных устройств обработки информации, каждое из которых реализует специфические функции общего в информационном процессе, направленного на достижение системой определенной цели.

Разработка РИУС предполагает оптимальный выбор параметров производительности, в частности, предельно допустимых уровней загруженности объектов: узлов обработки информации и (или) линий связи. Подобные оптимизационные задачи после формализации сводятся к моделям многопоточных систем массового обслуживания. Методом их решения является статистическое моделирование. При реализации модели РИУС на вычислительном устройстве общего назначения параллельные процессы модели происходят в *физическом* времени последовательно, что замедляет работу модели пропорционально ее размеру (количеству объектов). В [1] предложен вариант сокращения времени работы модели в результате использования нечетко-логических локально-параллельных алгоритмов [2, 3]. Целью настоящей работы является конкретизация предложенного подхода, рассмотрение вариантов реализации и оценки достигаемого выигрыша.

Предметная область

РИУС, безотносительно к характеру обрабатываемой информации, может рассматриваться как совокупность *объектов*: *узлов* $A:\{A_1, A_2 \dots A_n\}$, в которых происходит обработка информации, и *линий связи* между узлами a_{ij}, ($i,j \in \{1, 2 \dots n\}$, $i \neq j$, n – количество узлов), по которым узлы обмениваются информацией. Процесс пересылки информации из A_i в A_j по a_{ij} без утраты общности можно рассматривать как обработку информации на узле a_{ij}, имеющем вход A_i и выход A_j, поэтому далее узлы и линии связи рассматриваются единообразно, как *объекты* РИУС. На всех или некоторых объектах A_i не синхронно в случайные моменты времени порождаются задания $T(k)$, $T(l)$ случайной длительности выполнения, где k, l – параметры заданий. Некоторые задания выполняются на объекте порождения $T(A_i)$, другие – на других объектах, куда они передаются $T(A_i \rightarrow A_j)$. Некоторые задания в процессе выполнения порождают другие задания $T(k) \Rightarrow T(l)$. В частности, если на A_i порождается задание $T(A_i \rightarrow A_j)$, это эквивалентно $T(a_{ij})$, которое при выполнении порождает $T(A_j)$, т.е. $T(A_i \rightarrow A_j) \equiv T(a_{ij}) \Rightarrow T(A_j)$. Ресурсы объектов ограничены, поэтому из заданий образуются очереди на использование ресурсов. Задания могут иметь приоритетность выполнения и ограниченную актуальность: приоритетные задания выстраиваются в приоритетные очереди, а задания, «время жизни» которых вышло, удаляются из очередей без выполнения, как неактуальные. Очереди могут иметь ограниченный размер, при превышении которого задания отвергаются во избежание перегрузки системы.

Представляют интерес задачи *оценки надежности* работы (например, вероятности "коллапса" системы вследствие перегрузки) для конкретной конфигурации РИУС при заданной структуре загрузки (например, при заданных функциях распределения длительностей выполнения заданий $f(t_1)$ и интервалов следования заданий $f(t_2)$). Интересны также *оценки пороговой загрузки* для каждого объекта системы, при превышении которой вероятность "коллапса" превышает заданную. Представляют ценность выводы и рекомендации по реконфигурации системы в целях повышения ее работоспособности (улучшения загрузочных или пропускных характеристик) при проектировании системы, штат-

ной эксплуатации, аварийных ситуациях (работа системы вблизи перегрузки вследствие выхода из строя некоторых объектов системы) или модернизации отдельных фрагментов системы. Важна также *система поддержки принятия решений*, способная оказать оперативную консультативную помощь диспетчеру РИУС в штатном и аварийном режиме. Для решения указанных и подобных задач РИУС должна *исследоваться* (более или менее тщательно в зависимости от сложности и важности системы) при разработке, в период штатной эксплуатации и при модернизации. Методом исследования системы может быть моделирование.

Последовательное моделирование

Математическим аппаратом для формализации и исследования представленной предметной области являются *системы массового обслуживания* (СМО). Ввиду громоздкости адекватных описаний и проработанности аппарата СМО только для сравнительно простых структур возможности получения *аналитических* решений для случаев, имеющих практическую ценность, проблематичны. Более приемлемо *статистическое моделирование* на основе СМО-формализации. Объектам РИУС при этом ставятся в соответствие индивидуальные генераторы случайных чисел (ГСЧ) с заданными функциями распределения $f(t_1)$ и $f(t_2)$, с помощью которых формируются очереди заданий. Текущие состояния очередей отслеживаются и анализируются в качестве выходной информации. Распространено два варианта организации статистических моделей [4, с. 138], различающиеся способом продвижения *модельного времени* (МВ): по особым состояниям (ОС) и малыми дискретными интервалами, не связанными с ОС.

В первом варианте МВ представляет собой последовательность ОС – дискретных моментов начала и завершения выполнения заданий. Список ОС – "временная ось" МВ – составляется как суперпозиция моментов времени начала и завершения заданий, формируемых генераторами случайных чисел *всех* объектов. ОС последовательно просматриваются. Если текущее ОС является *завершением* задания, то соответствующий объект переключается в состояние "свободен". Если ОС является *началом* задания, то проверяется занятость объекта, задание ставится на выполнение (если объект свободен), переносится на другой момент (вставляется новая запись в список ОС) либо уничтожается (если срок его актуальности истек).

Второй вариант предполагает продвижение МВ малыми дискретными интервалами Δt, не связанными с ОС. В каждый момент просматриваются начала очередей и соответствующие задания ставятся на выполнение, снимаются либо уничтожаются аналогично первому варианту.

Первый вариант отличается компактностью: в каждый из рассматриваемых моментов *происходят* какие-либо изменения с заданиями. Второй вариант более расточителен: Δt должен быть меньше типичного интервала между ОС, иначе осложняется обработка завершения заданий; поэтому значительную часть составляют опросы моментов времени, в которые *не происходит* никаких изменений с заданиями. Однако второй вариант целесообразен при наличии в моделируемой системе аналоговых объектов, характеристики которых рассчитываются по аналитическим выражениям, как функции МВ.

Оба варианта модели виртуально содержат *параллельные* очереди заданий, которые на однопроцессорных вычислительных средствах физически реализуются (просматриваются) как *последовательные*. В связи с этим работа моделей замедляется пропорционально количеству объектов. В [1] предложен вариант сокращения времени работы моделей РИУС в результате использования нечеткологических (НЛ) локально-параллельных (ЛП) алгоритмов [2,3].

Локально-параллельные алгоритмы нечеткой логики

НЛ базируется на представлениях *теории нечетких множеств* (НМ) [5]. НМ A считается заданным, если перечислены составляющие его элементы A:$\{a_1, a_2, \ldots a_n\}$ (n – число элементов НМ A) и указаны значение *функции принадлежности* (ФП) μ_A:$\{\mu_A(a_1), \mu_A(a_2), \ldots \mu_A(a_n)\}$, $0 \le \mu_A(a_i) \le 1$, $i=1, 2, \ldots, n$. Для представления НМ A в локально-параллельной форме [2] значения ФП μ_A *масштабируются* ($0 \le \bar{\mu}_A(a_i) \le M$; $M=(2^m-1)$; m – число бит, выделяемых под хранение одного значения ФП; $m \in (1, 2, 3, \ldots)$; случай $m=1$ соответствует четкому множеству), *аппроксимируются* целочисленными значениями в интервале $[0, M]$ (например, с равномерным шагом $M/(n-1)$: $\mu_A(a_1)=0$, $\mu_A(a_2)=M/(n-1)$, $\mu_A(a_3)=2M/(n-1)$ $\ldots \mu_A(a_i)=(i-1)M/(n-1) \ldots \mu_A(a_n)=M$) и *конкатенируются*, образуя *регистровое представление* (РП) $(\overline{\overline{\mu}}_A)_R = \overline{\overline{\mu}}_A(a_1) \oplus \overline{\overline{\mu}}_A(a_2) \oplus \ldots \oplus \overline{\overline{\mu}}_A(a_n)$ (символ $\oplus$ здесь обозначает операцию конкатенации). Сказанное соответствует схеме

$$\mu_A \xrightarrow{масштабир.} \overline{\mu}_A \xrightarrow{аппрокс.} \overline{\overline{\mu}}_A \xrightarrow{конкат.} (\overline{\overline{\mu}}_A)_R.$$

НЛ операции в ЛП варианте производятся над РП, как над числами. Алгоритмы операций представляют собой последовательности алгебраических и поразрядных логических процедур, специальным образом подобранных так, что получаемые результаты интерпретируются как конкатенации результатов соответствующих операций, проделанных раздельно над ФП – конкатенантами РП. Имеется в виду следующее. Пусть $(\overline{\overline{\mu}}_A)_K$ и $(\overline{\overline{\mu}}_B)_K$ – РП, соответствующие НМ A и B; символ $\otimes$ обозначает НЛ операцию в ЛП варианте. Тогда результат $(\overline{\overline{\mu}}_C)_R=(\overline{\overline{\mu}}_A)_K\otimes(\overline{\overline{\mu}}_B)_K=\overline{\overline{\mu}}_C(c_1)\oplus\overline{\overline{\mu}}_C(c_2)\oplus\ldots\oplus\overline{\overline{\mu}}_C(c_n)$ таков, что $\overline{\overline{\mu}}_C(c_i)=\overline{\overline{\mu}}_{A\otimes B}(a_i,b_i)=\overline{\overline{\mu}}_A(a_i)\otimes\overline{\overline{\mu}}_B(b_i)$, где $i=1, 2, \ldots, n$.

Во [2, 3] представлены ЛП варианты некоторых алгоритмов НЛ. ЛП алгоритмы в сравнении с последовательными отличаются более высокой производительностью, в особенности если произведение *nm* не превышает разрядности процессора, т. е. если РП целиком помещается в регистре процессора, а не обрабатывается по частям. В качестве иллюстрации проведем сравнение последовательного НЛ алгоритма *ограниченной суммы* (Serial-BoundedSum(A, B)): $\mu_{A<+>B}(x)= min(1, \mu_A(x)+\mu_B(x))$ [3] и ЛП *масштабированного* варианта того же алгоритма (LocalParallel-BoundedSum(A, B)): $\mu_{A<+>B}(x)= min(M, \overline{\overline{\mu}}_A(x)+\overline{\overline{\mu}}_B(x))$. Ниже представлены формализованные записи алгоритмов в системе обозначений (псевдокоде) согласно [6, с. 21], сопровождаемые расчетом производительности. В колонках Σ и N представлены весовые коэффициенты временных затрат и показатели кратности выполнения соответствующих строк алгоритма.

Serial-BoundedSum(A, B)	Σ	N
1 **for** i←1 **to** *length*[A]	c_1	n
2 sum←($A[i]+B[i]$)	c_2	n
3 **if** sum<1	c_3	n
4 **then** $C[i]$←sum	c_4	k
5 **else** $C[i]$←1	c_5	$n-k$

Здесь A и B – входные данные – массивы значений ФП; *length*[A] – размер массива A (равен размеру n массива B); sum – промежуточная переменная; C – массив выходных значений; k – количество просуммированных пар ($A[i]+B[i]$), для которых sum<1, $k<n$. Время выполнения алгоритма Serial-BoundedSum (A, B) является функцией n: $T(n) = n(c_1+c_2+c_3)+kc_4+(n-k)c_5$; при этом, учитывая, что $c_4=c_5$, $T(n)=n(c_1+c_2+c_3+c_4)$. Легко видеть, что *length*[A] = n, таким образом, $T(n)$ *пропорционально* количеству значений ФП.

LocalParallel-BoundedSum(A, B)	Σ	N
1 $AZ1$ ←(A **"and"** $E1$)	c_1	1
2 $AZ2$ ←(A **"and"** $E2$)	c_2	1
3 $BZ1$ ←(B **"and"** $E1$)	c_3	1
4 $BZ2$ ←(B **"and"** $E2$)	c_4	1
5 $sum1$ ←($AZ1+BZ1$)	c_5	1
6 $sum2$ ←($AZ2+BZ2$)	c_6	1
7 $mask1$ ←(($sum1$ **"and"** $E2$)-$L1$)	c_7	1
8 $mask2$ ←(($sum2$ **"and"** $E1$)-$L2$)	c_8	1
9 $sum1$ ←(($mask1$ **"or"** $sum1$) **"and"** $E1$)	c_9	1
10 $sum2$ ←(($mask2$ **"or"** $sum2$) **"and"** $E2$)	c_{10}	1
11 sum ←($sum1$ **"or"** $sum2$)	c_{11}	1

Здесь A и B – входные данные ФП в регистровом представлении; $E1$, $E2$, $L1$ и $L2$ – E-шаблоны и L-шаблоны-константы, соответствующие формату представления информации во входных данных:

$E1$=(... 111...11 000...00 111...11 000...00 111...11)$_2$,
$E2$=(... 000...00 111...11 000...00 111...11 000...00)$_2$
$L1$=(... 000...01 000...00 000...01 000...00 000...01)$_2$,
$L2$=(... 000...00 000...01 000...00 000...01 000...00)$_2$,

содержащие в двоичном представлении чередующиеся фрагменты длиной *m* из нулей и единиц. *E*- и *L*-шаблоны считаются известными алгоритму LocalParallel-BoundedSum(*A*, *B*) на момент поступления входных данных. Переменные *AZ1*, *AZ2*, *BZ1*, *BZ2*, *sum1*, *sum2*, *mask1*, *mask2* – промежуточные; *sum* – выходное значение.

Данный вариант реализации ЛП алгоритма ограниченной суммы немного изменен ("ускорен") по сравнению с представленным в [3]: формирование масок, *mask1* и *mask2* (пункты 7 и 8) сделано не зависящим от длины *m* фрагмента. Оценка выигрыша в производительности представленного варианта по сравнению с [3] требует отдельного рассмотрения.

Как следует из данных колонки **N**, время выполнения алгоритма LocalParallel-BoundedSum(*A*, *B*) является константой: $T = c_1+c_2+...+c_{11}$. Таким образом, при отсутствии ограничений по разрядности вычислительного средства для ЛП алгоритма, в отличие от последовательного, время выполнения не зависит от объема обрабатываемой информации.

Разумеется, далеко не каждая реальная задача, в которой применялся бы рассматриваемый алгоритм, не чувствительна к разрядности процессора; однако такие задачи могут быть указаны. В частности, некоторую прикладную значимость может иметь модель РИУС, содержащая до 20 объектов при 8 градациях значений длины (времени выполнения) заданий. Такая модель в ЛП варианте выполнима на 64-разрядном вычислительном устройстве. Упрощенный вариант рассмотрен ниже.

Кроме того, ЛП алгоритмы могут обеспечить независимость от объема обрабатываемой информации на *многопроцессорных* вычислительных устройствах, при синхронной параллельной организации регистров процессоров в единый "конкатенированный" регистр. Данный режим и в целом данное направление – использование локально-параллельных алгоритмов на глобально-параллельной вычислительной технике – требуют специального рассмотрения.

Моделирование средствами нечетких ЛП алгоритмов

Представленный выше ЛП алгоритм ограниченной суммы удобен для использования в статистической модели и интересен тем, что позволяет организовать учет времени выполнения заданий. Рассмотрим подобную модель применительно к РИУС.

Моделирование осуществляется в виде последовательности дискретных циклов. Период следования соответствует минимальной требуемой продолжительности моделируемого события. ЛП алгоритмы производят бинарные НЛ операции над конкатенациями $\overline{\overline{\mu}}_C = \overline{\overline{\mu}}_A \otimes \overline{\overline{\mu}}_B$. При этом бинарная операция НЛ над отдельной составляющей – *i*-й конкатенантой $\overline{\overline{\mu}}_C(c_i) = \overline{\overline{\mu}}_A(a_i) \otimes \overline{\overline{\mu}}_B(b_i)$ – интерпретируется как модель события в *i*-м узле РИУС, а операция над конкатенациями в целом – как модель, охватывающая *n* узлов. Протяженное во времени событие считается завершившимся, если результат операции достигает порогового значения: $\overline{\overline{\mu}}_C(c_i)=M$, иначе в следующем цикле $\overline{\overline{\mu}}_A(a_i) = \overline{\overline{\mu}}_C(c_i)$ и генерируется новое значение $\overline{\overline{\mu}}_B(b_i)$. Таким образом, значения ФП в конкатенантах $\overline{\overline{\mu}}_A(a_i)$ являются текущими данными о степени завершенности соответствующих событий. Регистрация факта завершения события в *i*-й конкатенанте осуществляется формированием маски $L=(00..0)_{(1)}(00..0)_{(2)}..(11..1)_{(i)}..(00..0)_{(n)}$ с помощью алгоритма, аналогичного описанному в [1]. Операнды $\overline{\overline{\mu}}_A$ и $\overline{\overline{\mu}}_B$ порождаются ГСЧ: генерируется число разрядностью *nm*, из которого с помощью маски *L* вырезается составляющая, соответствующая требуемой конкатенанте.

В зависимости от типа выбранной логической операции ⊗ и статистических характеристик ГСЧ, конкатенанты $\overline{\overline{\mu}}_C(c_i)$ достигают пороговых значений за разное количество циклов, что позволяет вводить масштабирование и варьировать параметры различных ветвей моделируемой РИУС в соответствии с особенностями предметной области.

Алгоритмическая реализация

Рассмотренная НЛ модель РИУС реализована в двух вариантах: ЛП и последовательном. Реализация осуществлена для РИУС, представляющего собой СМО из *m* независимых обрабатывающих устройств одинаковой производительности с очередями нулевой длины. Данная структура не слишком содержательна как модель реальной системы, однако вполне достаточна для сравнения производительностей ЛП и последовательного НЛ алгоритмов. На рисунке представлены блок-схема алгоритма, фрагменты протоколов для ЛП и последовательной моделей, а также график – результат моделирования.

ЛП и последовательная модели реализованы в виде двух отдельных программ, максимально сопоставимых по структуре (иллюстрируется блок-схемой), языку (Turbo Pascal 7.0) и стилю выполнения (процедурное программирование, ввод и вывод информации через *input* и *output* файлы).

Рассмотрим блок-схему модели. *Генератор заданий* выдает псевдослучайные значения T_i, соответствующие длительностям заданий. В ЛП варианте это одно число, интерпретируемое в виде конкатенации сегментов, как описано выше; в последовательном варианте – набор чисел, масштабированных так, чтобы соблюдалось соответствие с ЛП вариантом. В дальнейшем применяется алгоритм масштабированной нечеткой ограниченной суммы, поэтому время выполнения задания T_i равно (M-T_i) циклов. *Генератор фильтров заданий* выдает псевдослучайные значения, используемые далее при принятии решений о выдаче заданий. В ЛП варианте в сгенерированном псевдослучайном двоичном числе F младшие разряды сегментов "вырезаются" и "растягиваются" аналогично представленному выше в пунктах 7 и 8 алгоритма LocalParallel-BoundedSum(A, B):

$$F \leftarrow ((F \text{ “}AND\text{”} (L_1 \text{ “}OR\text{”} L_2))<<m) - (F \text{ “}AND\text{”} (L_1 \text{ “}OR\text{”} L_2)).$$

В последовательном варианте, например, для каждого из заданий проверяется попадание псевдослучайного числа в заданный интервал. Таким образом, задания и решения о выдаче заданий генерируются раздельно, вследствие чего задания имеют случайную длину и случайный момент выдачи. При синтезе моделей конкретных систем генераторы заданий и фильтров заданий могут регулироваться раздельно для получения требуемых законов распределения потоков заданий.

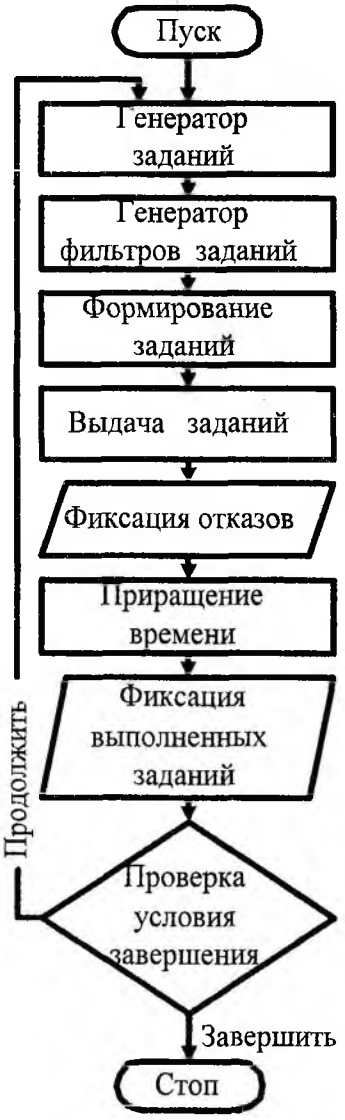
В ЛП варианте *формирование заданий* представляет собой "просеивание" сгенерированного пакета заданий G через фильтр заданий F и *шаблон свободных сегментов* S (в начальном состоянии все сегменты свободны: $S \equiv E_1 \text{“}OR\text{”} E_2$); *выдача заданий* – суммирование сформированных заданий с конкатенацией сегментов модели; *фиксация отказов* осуществляется "просеиванием" G через дополнение к шаблону S: $G \text{“}AND\text{”}((E_1 \text{“}OR\text{”} E_2)\text{-}S_0)$ и подсчетом числа ненулевых сегментов. В последовательном варианте указанные операции реализуются соответствующими логическими функциями.

Далее *приращение времени* реализуется как ограниченное НЛ суммирование, а *фиксация выполненных заданий* – как регистрация элементов, достигших максимального значения M (в ЛП варианте – формирование S).

Проверка условия завершения возможна, например, по достижению требуемого числа сформированных, отказанных либо выполненных заданий (столбцы **I** (*in*), **L** (*lost*) и **E** (*end*) соответственно, протоколы, (рис. 1); либо по числу циклов суммирования (тактов модельного времени), как это реализовано в тех же протоколах.

Обращает на себя внимание качественно отличный характер МВ от рассмотренных выше вариантов: по принципу ОС и с малыми дискретными интервалами, не связанными с ОС. В рассмотренной модели МВ идет малыми дискретными интервалами, но моменты времени *связаны* с ОС и синхронизируются ими. Время не определяется событиями, которые в нем происходят, а задается в виде равномерной дискретной сетки (столбец 1), к которой события (ОС) привязаны. В связи с этим для каждого момента времени указано не только начало либо завершение *какого-либо* задания, но и состояние выполнения *каждого* задания (столбец 5). Очевидно, если события в модели происходят редко, большое число моментов МВ оказывается пустым. Но такой режим эксплуатации системы не представляет особого интереса с точки зрения моделирования: в нем не происходит ничего критического, что соответствовало бы сформулированным выше задачам моделирования РИУС. Обычно в режимах, представляющих практический интерес, события в МВ располагаются достаточно плотно, и, следовательно, данный вариант МВ отличается высокой эффективностью: модель в процессе работы редко рассматривает пустые ситуации.

Столбец 5 позволяет наблюдать выполнение каждого из заданий. Например, согласно протоколу ЛП модели (рисунок) первый объект (младший сегмент) за 10 первых циклов работы модели загружался трижды: во 2-м и 4-м циклах заданиями продолжительностью 2 цикла и в 7-м цикле заданием продолжительностью 5 циклов. Последнее задание на момент завершения 10-го цикла осталось не довыполненным на один цикл. "Наблюдаемость" представляется очень полезным свойством модели. Например, модель может быть усложнена так, чтобы в ней воспроизводились ситуации с прерыванием выполнения заданий при определенных перегрузках или аппаратурных отказах. Тогда по протоколам модели аварийные ситуации могут быть проанализированы во времени и в контексте взаимодействия с другими элементами системы.



	I	L	E	Локально-параллельная модель									
1	1	0	0	11	000	000	000	000	000	000	000	000	
2	3	0	0	100	110	101	000	000	000	000	000	110	
3	3	1	3	101	000	110	000	101	000	000	000	000	
4	1	0	1	110	000	000	000	110	000	000	000	110	
5	4	2	3						110	000	101	000	
6	1	0	1							101	110	000	
7	3	1	1		110	000	000	000	000	110	000	011	
8	1	1	2									100	
9	1	0	0							100	000	101	
10	1	0	0						11	101	000	110	
	19	5	11										

	I	L	E	Последовательная модели									
1	3	0	0	0	0	0	2	3	0	0	0	0	2
2	3	0	0	5	0	0	3	4	6	0	0	2	3
3	6	3	0	6	7	3	4	5	7	2	0	3	4
4	4	4	2	7	0	4	5	6	0	3	0	4	5
5	6	5	1	0	5	5	6	7	0	4	0	5	6
6	4	2	1	6	6	6	7	0	2	5	0	6	7
7	5	5	2	7	7	7	0	0	3	6	0	7	0
8	2	1	4	0	0	0	3	0	4	7	0	0	0
9	4	1	1	0	0	0	4	6	5	0	0	4	7
10	5	2	1	0	5	0	5	7	6	7	5	5	0
	42	23	12										

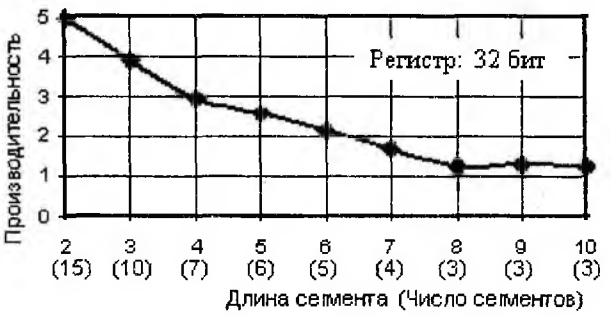


Рис.

Разумеется, подобной "прозрачностью" для наблюдения обладает также представленный последовательный вариант модели (столбец 5 протокола, рисунок), однако не следует забывать, что такое решение не продиктовано структурной необходимостью модели, а сделано искусственно, для обеспечения единообразия и сопоставимости ЛП и последовательной моделей по сложности и стилю разработки. Классический вариант модели РИУС с организацией МВ по принципу ОС обычно более громоздкий. Моменты времени в нем отсчитываются не регулярно, а только по началам и окончаниям заданий. Расчет состояния выполнения каждого оставшегося задания обычно не делается, так как это тормозит работу модели и не слишком информативно: соответствующие временные метки следуют нерегулярно, что осложняет их интерпретацию.

Под столбцами 2, 3 и 4 указаны суммарные значения **I, L и E**, между которыми соблюдается "закон сохранения" с учетом незавершенных заданий. Например, для ЛП алгоритма незавершенных заданий (ненулевых сегментов в 10-й строке) 3, поэтому 19=5+11+3. Наличие данного инварианта демонстрирует корректность представленных моделей.

Оценка производительности

На графике (рисунок) представлены основные результаты сравнения ЛП и последовательной моделей – график зависимости выигрыша в производительности ЛП модели по сравнению с последовательной (ось ординат) от длины сегмента (в битах), либо числа сегментов при 32-разрядном регистре процессора. Выигрыш в производительности определялся как отношение времени работы последовательного алгоритма ко времени работы ЛП алгоритма при одинаковом числе циклов, которое

выбиралось так, чтобы разброс измерений времени составляет 1-3 %. На компьютере с процессором AMD-K5 75 МГц число циклов составляло 0,5 млн.

Поскольку машинные эксперименты проводились при фиксированной разрядности процессора, длина сегмента взаимно однозначно соответствует числу сегментов (значения по оси абсцисс, в скобках). Шкала длин сегментов линейная. Обращает на себя внимание наличие на графике трех разных линейных (с учетом погрешностей) участков: 2-4, 4-8 и 8-10. Последний участок практически параллелен оси абсцисс, два других – наклонные, причем первый – круче. Ситуация проясняется, если рассматривать график по шкале чисел сегментов. Участок 4-8 является равномерным по числу сегментов, с шагом 1: (7)-(3). Участок 2-4 неравномерный и имеет более крутой шаг: 2-3→(15)-(10), шаг 5; 3-4→(10)-(7), шаг 3. Участок 8-10 стягивается в точку (3). В результате, если перерисовать график с линейной осью чисел сегментов, экспериментальные точки (в пределах погрешностей) лягут на прямую, соответствующую участку 4-8. Таким образом, с учетом ограничений данного машинного эксперимента, выигрыш в производительности ЛП алгоритма по сравнению с последовательным *пропорционален* числу сегментов.

Пропорциональность выигрыша в производительности числу сегментов, продемонстрированная в машинном эксперименте, хорошо согласуется с приведенным выше для алгоритма LocalParallelBoundedSum (A, B) выводом на основе расчета T, о независимости производительности от объема обрабатываемой информации. Необходимо отметить, что в пределах машинного эксперимента разрядность процессора не изменялась, что соответствует условию отсутствия ограничений по разрядности вычислительного средства, при котором сделан указанный вывод для алгоритма LocalParallelBoundedSum (A, B).

Как следует из графика, выигрыш имеется даже при трех сегментах, а при 3- и 4-битных сегментах (7 и 15 градаций значений длины заданий, что вполне может представить интерес для практического моделирования) выигрыш 4- и 3-кратный соответственно.

Выводы

Предложен подход к моделированию распределенных информационно-управляющих систем на основе локально-параллельных алгоритмов обработки нечеткой информации. Рассмотрен вариант реализации машинной модели на алгоритме ограниченной суммы. Показано, что при отсутствии ограничений по разрядности вычислительного средства, для локально-параллельного варианта этого алгоритма, в отличие от последовательного, время выполнения не зависит от объема обрабатываемой информации. Особенности использования алгоритма продемонстрированы на примере системы массового обслуживания с очередями нулевой длины. Модель системы реализована в локально-параллельном и традиционном последовательном вариантах. Показано, что локально-параллельная реализация модели при 10 узлах обработки по производительности превышает аналогичную модель на последовательных алгоритмах в среднем в три раза.

Список литературы: 1. *Михаль О.Ф.* Моделирование распределенных систем средствами локально-параллельных алгоритмов обработки нечеткой информации // 7-я междунар. конф. "Теория и техника передачи, приема и обработки информации": Сб. научн. тр. 2001. С. 224-225. 2. *Михаль О.Ф., Руденко О.Г.* Принцип локальной параллельности в задачах обработки нечеткой информации. Базовые операции // Доповіді НАН України. 1999. № 8. С. 71-75. 3. *Михаль О.Ф., Руденко О.Г.* Принцип локальной параллельности в задачах обработки нечеткой информации. Расширенный набор операций // Доповіді НАН України. 2000. №1. С.75-78. 4. *Альянах Н.Н.* Моделирование вычислительных систем. Л.: Машиностроение, 1988. 5. *Заде Л.А.* Понятие лингвистической переменной и его применение к принятию приближенных решений. М.: Мир, 1976. 6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2001.

Поступила в редколлегию 10.15.2001