Работа с параллельными структурами данных в инструментальном комплексе ORLANDO TOOLS

А.Г. Феоктистов, С.А. Горский

В статье обсуждаются способы организации параллельных структур данных и методы их обработки в инструментальном комплексе ORLANDO (Objects Relations in LANguage of DescriptiOns) TOOLS, предназначенном для разработки и применения пакетов параллельных программ. Приводится алгоритм управления асинхронным вычислительным процессом.

1. Введение

Математическое моделирование с применением средств вычислительной техники является широко используемым инструментом проведения исследований при анализе сложных систем (экономических, производственных, информационных и др.) в различных отраслях науки, образования и бизнеса. Одним из основных этапов этого процесса, вслед за построением математической модели системы, является вычислительный эксперимент, позволяющий осуществить численное исследование полученной модели путем воспроизведения поведения системы в различных условиях или в различных ее модификациях. При этом, зачастую, возникает необходимость выполнения массовых ресурсоемких вычислений.

На сегодняшний день для решения подобных задач имеются разнообразные программные и аппаратные средства проведения высокопроизводительных вычислений на параллельных компьютерах. Однако массовое использование такого рода вычислений сдерживается рядом нерешенных на сегодняшний день проблем параллельного программирования, в том числе, сложностями для исследователя (специалиста в своей предметной области), возникающими при разработке им параллельных алгоритмов, написании и выполнении параллельных программ и др. (см., например, [1]).

Одним из подходов, позволяющих частично решить эти проблемы, является применение для организации параллельных и распределенных вычислений пакетов прикладных программ в рамках технологии модульного программирования [2]. В этом случае появляется необходимость использования новых параллельных структур данных и средств их обработки. Способы решения этих вопросов в инструментальном комплексе ORLANDO TOOLS, разработанном авторами доклада, представлены в данной работе.

2. Модель предметной области

Пакет программ рассматривается в нашем подходе традиционно [3] – как совокупность прикладного и системного программного обеспечения. Прикладное программное обеспечение (функциональное наполнение пакета программ) представляет собой набор (библиотеку) вычислительных модулей. Под модулем понимается подпрограмма на языке программирования Си или Фортран, спецификации по ее назначению, применению, формату входных и выходных формальных параметров, времени выполнения для той или иной программно-аппаратной платформы, инструкции по ее трансляции, компиляции и дальнейшему размещению объектных и загрузочных модулей и т.п. Системное программное обеспечение включает средства для описания модели предметной области, формирования постановок исследовательских задач и организации процесса их решения.

Модели предметных областей, создаваемые в рамках подобных пакетов программ, относятся к классу вычислительных моделей [4]. Они позволяют описывать вычислительные возможности пакета программ при решении определенного класса задач. Такую вычислительную модель можно определить как совокупность значимых величин (параметров) предметной области и информационно-логических связей между ними, реализуемых модулями пакета программ. Таким образом, вычислительная модель, по сути, определяет правила применения и со-

четания модулей в процессе решения задачи и позволяет автоматически осуществлять планирование вычислений по непроцедурной постановке задачи вида «по заданным значениям параметров $x_1, x_2, \ldots, x_k$ вычислить значения параметров $y_1, y_2, \ldots, y_r$».

Множество допустимых типов значений параметров включает стандартные простые типы данных (целый, вещественный, логический и символьный), используемые для определения скалярных величин (параметров-скаляров). На их основе образуются структурированные типы данных – массивы. Параметры-массивы представляются в виде векторов и матриц. Их границы изменения индексов задаются параметрами-скалярами целого типа.

Концептуальная модель предметной области представляется в виде структуры $S=<Z, T, F>$, где: $Z=\{z_1, z_2, \ldots, z_n\}$ – множество параметров предметной области; $T=\{t_1, t_2, \ldots, t_k\}$ – множество допустимых типов данных; $F=\{f_1, f_2, \ldots, f_m\}$ – множество вычислительных модулей предметной области.

Для формального описания модели предметной области применим следующую нотацию [5]: *R(O1(n1,n2):O2(n3,n4)/c), где: O1, O2* – множества объектов предметной области, связанные бинарным отношением *R; n1*, *n2* – кардинальные числа, которые означают соответственно минимальное и максимальное число элементов *O1*, связанных с элементом *O2*; *n3*, *n4* – кардинальные числа, которые означают соответственно минимальное и максимальное число элементов *O2*, связанных с элементом *O1*; *c* – ограничение целостности, накладываемое на элементы *O2*, участвующие в отношении *R*. Тогда связи между множествами объектов предметной области можно определить следующими отношениями (Рис. 1.):

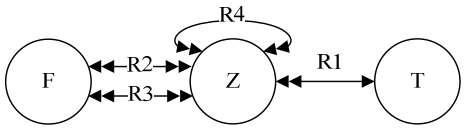


Рис. 1. Концептуальная модель предметной области

- *R1(Z(1,n): T(1, 1))* – типизированные параметры;
- *R2(ZI(1,ni):F(1,m),c1)* – входные параметры из $ZI \subset Z$ для модулей из F, $1 \le ni < n$; с каждым модулем f_i связано множество $In(f_i) \subset Z$ его входных параметров; *c1* – ограничение на множества входных параметров модуля ($In(f_i) \cap Out(f_i) = \varnothing$).
- *R3(ZO(1,no):F(1,m),c1)* – выходные параметры из $ZO \subset Z$ для модулей из F, $1 \le no < n$; с каждым модулем f_i связано множество $Out(f_i) \subset Z$ его выходных параметров; *c1* – ограничение на множества выходных параметров модуля ($In(f_i) \cap Out(f_i) = \varnothing$).
- *R4(ZB(1,2):ZA(1,na)/c2)* – параметры из $ZB \subset Z$, задающие границы изменения индексов параметров-массивов из $ZA \subset Z$, $1 \le na < n$; *c2* – ограничение требующее, чтобы эти параметры были скалярами целого типа.

Введем новый тип данных *параметр-список* для определения параллельных структур данных[1]. Параметр-список создается на основе одного параметра любого вида (скаляра, вектора или матрицы) и включают множество вариантов значений этого параметра. Число элементов параметра-списка задается параметром-скаляром целого типа. Основное отличие параметра-списка от параметра-массива заключается в способе обработки элементов параметров таких типов. Параметр-массив целиком передается в вычислительный модуль, в котором далее осуществляется его обработка, в то время как параметр-список может обрабатываться поэлементно, в общем случае, параллельно, множеством экземпляров вычислительного модуля на разных процессорах.

Наличие в модели предметной области параметров-списков приводит к появлению в процессе вычислений новых объектов: элементов параметров-списков и обрабатывающих их экземпляров модулей. Модель вычислений, таким образом, представляется в виде структуры $CM=<S, PI, FI>$, где S – это рассмотренная выше модель предметной области, PI – множество элементов параметров-списков, FI – множество экземпляров модулей предметной области. От-

[1] Под параллельной структурой данных понимается способ организации данных, ориентированный на параллельную обработку элементов этой структуры в рамках конкретной вычислительной системы.

ношения между множествами объектов модели вычислений представлены следующими отношениями (Рис. 2.):

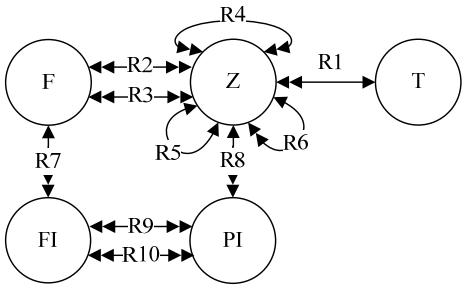


Рис. 2. Модель вычислений

- *R5(ZL(1, 1):ZU(1,1)/c3)* – параметры-списки из $ZL \subset Z$, создаваемые на основе параметров из $ZU \subset Z$; *c3* – ограничение, накладываемое на тип параметра, на основе которого создается список (он может быль параметром любого типа, кроме списка);
- *R6(ZB(1,1):ZL(1,nl)/c2)* – параметры из $ZB \subset Z$, задающие число элементов параметров-списков из $ZL \subset Z$, $1 \le nl < n$; *c2* – ограничение целостности требующее, чтобы эти параметры были скалярами целого типа;
- *R7(F(1, 1):FI(1,r))* – экземпляры модуля, $r \in N$;
- *R8(Z(1, 1):PI(1,s))* – элементы параметров-списков, $s \in N$;
- *R9(FI(1,r):PI(1,s)/c4)* – входные элементы параметров-списков экземпляра модуля, $r, s \in N$; *c4* – ограничение, требующее, чтобы индексы этих элементов и обрабатывающего их экземпляра модуля совпадали;
- *R10(FI(1,r):PI(1,s)/c5)* – выходные элементы параметров-списков экземпляра модуля, $r, s \in N$; *c5* – ограничение, требующее, чтобы индексы этих элементов и обрабатывающего их экземпляра модуля совпадали.

3. Алгоритм управления асинхронным вычислительным процессом

В процессе вычислений параметры-списки могут обрабатываться как неделимые структуры данных (Рис. 3.а), так и поэлементно (Рис. 3.б).

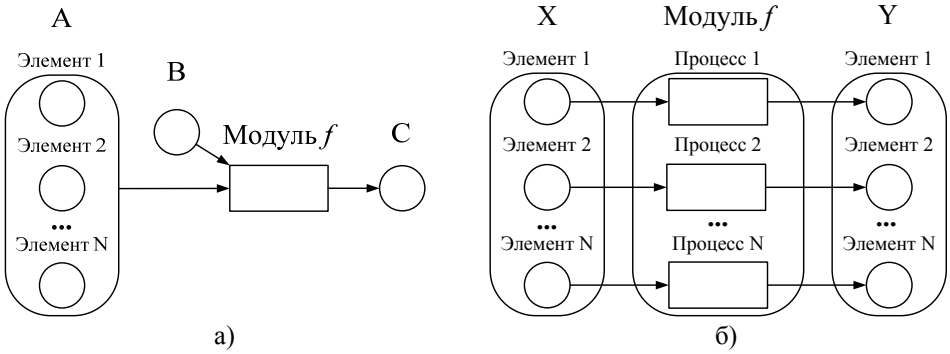


Рис. 3. Способы обработки параметров-списков: а) обработка параметра-списка как неделимой структуры данных; б) поэлементная обработка параметра-списка

Во втором случае под поэлементной обработкой параметров-списков, подразумевается следующее. Пусть *x*, *y* – параметры-списки, модуль f_i: $In(f_i) \rightarrow Out(f_i)$ предназначен для их поэлементной обработки, $x \in In(f\_i)$, $y \in Out(f\_i)$. Интерпретация модуля *f* выполняется следующим образом: 1) происходит параллельный запуск *N* экземпляров (*N* – размерность списков *x*, *y*) модуля *f*, *i*-му экземпляру модуля передается *i*-й элемент списка *x;* 2) результат присваивается *i*-му элементу списка *y*.

Ниже приведен разработанный авторами алгоритм управления асинхронным вычислительным процессом.

Рассмотрим условие выполнимости модуля. Пусть:

1) $C(f) \subset Z$ – множество параметров модуля f, значение которых вычислено;

2) $P(f) \subset ZL$ – множество параметров-списков, обрабатываемых модулем поэлементно и являющихся входными для него;

2) $E(f)=\{e_1, e_2, ..., e_u\}$, где e_i – номер индекса элементов во всех $p_j \in P(f)$, значение которых вычислено.

3) $A(f) = In(f) \setminus C(f) = \{z_1, z_2, ..., z_k\}$ – множество входных параметров модуля f, значения которых еще не вычислены.

Введем условия возможности запуска модуля. Модуль $f \in F$ может быть выполнен, если:

(1) $(A(f)=\varnothing) \vee (A(f) \subseteq P(f) \wedge E(f) \neq \varnothing)$.

Алгоритм работы программы представлен в таблице 1. Описание вспомогательных функций приведено в таблице 2.

Таблица 1. Алгоритм работы программы

I. Ввод исходных данных.

II. Проверка готовности модулей к запуску.

```
for (i=1 … m){
  if (!IsParallel(i)) then{
/*Проверка возможности запуска единственного экземпляра модуля f[i]*/
    f[i]->Calculate=check(i);}
  else{
/*Проверка возможности запуска множества экземпляров модуля f[i]*/
     for (j=1 … f[i].count){
       f[i]->Calculate[j]=check (i, j);}}}
```

III. Запуск модулей.

```
for (i=1 … m){
   if (is(f[i])) then{
     run(f[i])}}
```

IV. Если все модули выполнены, то происходит завершение работы программы. В противном случае осуществляется ожидание завершения выполнения очередного модуля, после чего совершается переход на шаг II.

Таблица 2. Описание вспомогательных функций

Функция	Комментарий
`IsParallel(i)`	Функция возвращает true, если модуль f_i подразумевает запуск нескольких экземпляров, иначе – false.
`is(fi)`	Функция определяет, помечен ли модуль f_i, как готовый к запуску;
`run(fi)`	Функция производит запуск модуля f_i, если выполнено ограничение на максимально возможное количество, запущенных в системе процессов.

`check(int i){`
`  For(k=1…  MaxPar){`
`  if (Link[i,k]==1 && Is(k));{`
`  return false;}}`
`  return true;}`
	Функция выполняет проверку условия (1) по множеству входных параметров модуля: если значения всех параметров вычислены, то функция возвращает true, иначе – false.

`check(int i, int j){`
`  For(k=1…  MaxPar){`
`  if (Link[i,k]==1 &&`
`  !IsParForOp(i, k, j));{`
`  return false;}}`
`  return true;}`
	Функция выполняет проверку условия (1) по множеству входных параметров экземпляра модуля: если значения всех параметров вычислены, то функция возвращает true, иначе – false.

`IsParForOp (int i, int k, int j){`
`  if(TLink[i][k]==1){`
`  return Iss(k, j);}`
`  if(TLink[i][k]==2){`
	Функция возвращает *true*, если значения всех элементы *k-го* параметра необходимые для *j*-го экземпляра модуля f_i вычислены, иначе – *false*.

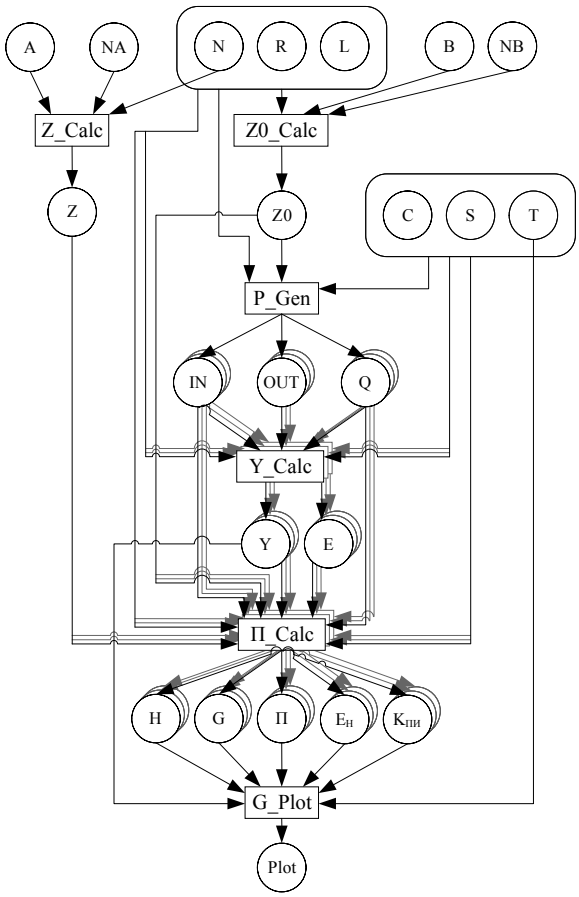
`return Is(k);}` `return false;}`	
`Is(k)`	Функция возвращает *true*, если значение *k-го* параметра вычислено, иначе – *false*.
`Iss(k, j)`	Функция возвращает *true*, если значение *j-го* элемент параметра *k* вычислено, иначе – *false*.
Структуры данных	
Link – массив содержащий информацию о входных и выходных параметрах модуля. Link[i,j] = 0, если $z_i \notin In(f_i) \cup Out(f_i)$; Link[i,j] = 1, если $z_i \in In(f_i)$; Link[i,j] = 0, если $z_i \in Out(f_i)$.	
TLink – массив содержащий информацию о том как параметр обрабатывается модулем. TLink[i,j] = 1, если параметр j обрабатывается модулем f_i как неделимая структура данных; TLink[i,j] = 2, если параметр j является параметром-списком и обрабатывается модулем f_i по-элементно.	

4. Пример

В качестве примера использования инструментального комплекса ORLANDO TOOLS для разработки пакетов параллельных программ рассмотрим задачу моделирования процесса сдачи предприятием в аренду объектов коммерческой недвижимости $o_i \in O,\ i=1,2,\ldots,N$. Каждый объект o_i относится к одной из категории объектов $v_j \in V,\ j=1,2,\ldots,R$ (например, офис, склад, гараж и т.п.). Заданы матрицы D и C, каждая размерности $N \times R$, определяющие соответственно возможность переоборудования объекта o_i (для перевода его из одной категории v_{j1} в другую категорию v_{j2}) и стоимость аренды объекта o_i категории v_j. Предприятие несет в момент времени τ расходы на текущее обслуживание этих объектов $Z(\tau)$ и на их переоборудование $Z^0(\tau)$. Предприятию требуется определять вероятные планы сдачи объектов в аренду, сроки аренды и возврата этих объектов из аренды для учета возможных финансовых поступлений. Для анализа эффективности реализации проекта и внесения корректирующих воздействий в ходе его выполнения предприятию необходимо производить расчеты финансовых показателей по аренде объектов за определенный период времени t. К этим показателям относятся: оценка потенциальной мощности услуг аренды $E(t)$; мощность освоенных услуг аренды $y(t)$; оценку мощности неосвоенных услуг аренды $E_н(t)$; коэффициент полезного использования аренды $k_{пи}$; суммарный доход по аренде $H(t)$; суммарные затраты $G(t)$; прибыль от услуг аренды $П(t)$ и др. Соотношения, определяющие указанные финансовые показатели, основываются на динамической модели экономического объекта [6], учитывающей временную структуру фондов, но в данной работе не приводятся ввиду ограниченности ее объема.

Вычислительная модель разработанного авторами доклада пакета программ (Рис. 4) включает следующие модули: Z_Calc(N, A, NA –> Z) – вычисляет затраты на обслуживание объектов; Z0_Calc(N, R, L, B, NB –> Z0) – вычисляет затраты на их переоборудование; P_Gen(N, R, L, Z0, T, S, C –> In, Out, Q) – выполняет построение планов сдачи (возврата) объектов в аренду (из аренды) и ее продолжительность; Y_Calc(N, R, L, In, Out, Q, T, S, C –>Y, E) – вычисляет мощность освоенных услуг аренды; P_Calc(N, R, L, Z, Z0, T, In, Q, Y, E, S, C –> H, G, П, $E_н$, $K_{пи}$) – вычисляет доход, затраты, прибыль, оценку неосвоенной мощности услуг аренды и коэффициент полезного использования объектов; G_Plot(T, Y, H, G, П, $E_н$, $K_{пи}$ –> Plot) – выполняет построение графиков. После имени модуля слева и справа от стрелки располагаются списки его входных и выходных параметров. Для модулей Y_Calc и P_Calc выполняется параллельный запуск множества их экземпляров. Элементы параметров-списков In, Out, Q, Y, H, G, П, $E_н$, $K_{пи}$ обрабатываются независимо друг от друга в отдельных процессах – экземплярах этих модулей.

Инструментальные средства данного пакета позволили предметному специалисту предприятия сформулировать ряд постановок задач, автоматически сгенерировать для них параллельные программы на языке C++ (с использованием функций библиотеки PVM) и выполнить с их помощью требуемые многовариантные расчеты.



Назначение параметров: N – количество объектов; R – количество категорий объектов; L – количество вариантов цены аренды объектов; T – количество временных периодов; NA – число компонентов расходов на текущее обслуживание объектов; A(NA,N) – матрица значений компонентов расходов на текущее обслуживание объектов; NB – число компонентов расходов на переоборудование объектов; B(NB,N) – матрица значений компонентов расходов на переоборудование объектов; Z(N) – вектор значений затрат на текущее обслуживание объектов; Z0(N,R) – матрица значений затрат на переоборудование объектов; S(N) – вектор значений площадей объектов; C(N,R) – матрица стоимости аренды объектов; In(N,T,R) – матрица содержащая информацию о сдаче объектов в аренду; Out(N,T,R) – матрица содержащая информацию о возврате объектов из аренды; Q(N,T,R) – матрица содержащая информацию о сроках аренды объектов; E – оценка потенциальной мощности услуг аренды; Y – значение мощности освоенных услуг аренды; H – значение дохода от аренды; G – значение затрат на обслуживание и переоборудование объектов; П – значение прибыли; $E_н$ – оценка неосвоенных услуг аренды; $K_{пи}$ – коэффициент полезного использования аренды; Plot – график.

Рис. 4. Вычислительная модель

5. Заключение

В заключение следует отметить, что введение параллельных структур данных в вычислительную модель в значительной степени расширило функциональные возможности пакетов прикладных программ в плане организации многовариантных расчетов.

Применение инструментального комплекса ORLANDO TOOLS при решении ряда задач для различных предприятий показало достаточную простоту и легкость использования входящих в этот комплекс средств конечными пользователями и обеспечило значительное сокращение трудозатрат на этапах создания и эксплуатации программного обеспечения сложных информационно-вычислительных моделей экономических объектов и методов их исследования.

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.
2. Майерс Г. Надежность программного обеспечения. – М.: Мир, 1980. – 360 с.
3. Горбунов-Посадов М.М., Корягин Д.А., Мартынюк В.В. Системное обеспечение пакетов прикладных программ. – М.: Наука, 1990. – 208 с.
4. Тыугу Э.Х. Концептуальное программирование. – М.: Наука, 1984. – 256 с.
5. Цикритзис Д., Лоховски Ф. Модели данных. – М. Финансы и статистика, 1985. – 344 с.
6. Сиразетдинов Т.К. Динамическое моделирование экономических объектов. – Казань: Фән, 1996. – 223с.