Анализ механического поведения крупноячеистых пространственно-армированных композитов проводился численно с использованием коммерческой лицензионной версии пакета ANSYS 11.0 на кластере Центра высокопроизводительных вычислительных систем Пермского государственного технического университета. Запуск на счет проводился с использованием удаленного доступа. Среднее количество элементов и узлов геометрических моделей составляет 380 000 и 513 000 соответственно.

С.В. Панков

Южный Федеральный университет, г. Ростов-на-Дону

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ ПРОЦЕССОВ В СИСТЕМАХ С АРХИТЕКТУРОЙ ТИПА ТОР

Рассматривается класс задач, адекватно реализуемых на параллельных распределённых вычислительных системах (кластерах) с архитектурой типа тор. При этом схема взаимодействия процессов в таких реализациях характеризуется высокой степенью регулярности. Многие задачи, реализуемые с помощью технологии распределённого программирования SPMD (Single Program Multiply Data), могут быть отнесены к этому классу. Например, задачи математического моделирования, использующие сеточные (конечно-разностные) вычислительные методы.

Представляется подход к моделированию и анализу поведения таких систем на базе формализма *L*-программ. Этот формализм включает в себя логико-математическую модель параллельных вычислений – *L*-программы (впервые описана в [1]) и средства анализа *L*-программ (разработаны в [2]). Обсуждается свойство нетупиковости систем. Этот подход направлен на автоматизацию анализа рассматриваемых систем. Обеспечивает компактность модели, независимость её вида от числа процессов. Это, в свою очередь, влечёт за собой независимость анализа системы от её размера.

Основные понятия. Моделирование распределённых систем осуществляется на уровне абстракции, не учитывающем вычислительные действия процессов и принимающем во внимание только операции посылки и приёма сообщений. В качестве архитектуры вычислительной системы здесь рассматривается прямоугольная решётка, замкнутая сама на себя в тор. Процессы развиваются на вычислительных узлах решётки. Горизонтальные линии связи решётки (с номерами от 0 до m-1) образуют продольные кольца тора, а вертикальные линии связи (с номерами от 0 до n-1) формируют поперечные кольца тора. Каждый процесс идентифицируется парой координат, образованной номером продольного кольца тора и номером его продольного кольца.

L-программы представляют собой систему переходов $S=(Q,T)$, где Q – множество состояний, а $T \subseteq Q \cdot Q$ – отношение перехода на Q. Множество состояний $P \subseteq Q$ называется *инвариантом* системы переходов S, если справедливо $\forall q',q \in Q$ $((q',q) \in T \ \& \ q' \in P) \Rightarrow q \in P)$. Через $fin[S]$ обозначим множество *заключительных состояний* системы переходов S, т.е. $fin[S]=\{q \in Q \mid \forall q' \in Q((q,q') \notin T)\}$.

Модели регулярных схем взаимодействия процессов в виде *L*-программ.

1. *Язык предметной области клеточных L.*

Сорта:

LONG – множество значений $\{0,1,...,m-1\}$ продольной координаты процесса;

LATER – множество значений $\{0,1,...,n-1\}$ поперечной координаты процесса.

Функции:

$\_+_m1$: $LONG \to LONG$ – увеличивает значение продольной координаты процесса на 1 по модулю m;

$\_-_m1$: $LONG \to LONG$ – уменьшает значение продольной координаты процесса на 1 по модулю m (определяется через функцию $+_m1$, как $i-_m1=(i+m-2)+_m1$, где $i \in LONG$);

$\_+_n1$: $LATER \to LATER$ – увеличивает значение поперечной координаты процесса на 1 по модулю n;

$\_-_n1$: $LATER \to LATER$ – уменьшает значение поперечной координаты процесса на 1 по модулю n.

В L также будут входить стандартные арифметические операции и отношения.

Предикаты:

!: $LONG \times LATER \times LONG \times LATER$ – задаёт для процесса, определяемого первой парой аргументов, запрос на передачу сообщения процессу, задаваемому второй парой аргументов;

?:$LONG \times LATER \times LONG \times LATER$ – задаёт для процесса, определяемого первой парой аргументов, запрос на приём сообщения от процесса, задаваемого второй парой аргументов.

Переменные: i,i': $LONG$; j,j': $LATER$**.**

Во всех следующих ниже моделях предполагается, что в начальный момент справедливо $\forall i,j,i',j'\ (\neg?(i,j,i',j') \wedge \neg!(i',j',i,j))$.

2. *Синхронная модель взаимодействия процессов со всеми своими соседями.*

1) $\$\ i,j,i',j'\ \ \neg!(i,j,i',j') \wedge (i'=i-_m1 \wedge j'=j \vee i'=i+_m1 \wedge j'=j \vee j'=j-_n1 \wedge i'=i \vee j'=j+_n1 \wedge i'=i) \Rightarrow$

$!(i,j, i',j')$

2) $\$\ i,j,i',j'\ \ \neg?(i,j,i',j') \wedge !(i',j',i,j) \wedge (i'=i-_m1 \wedge j'=j \vee i'=i+_m1 \wedge j'=j \vee j'=j-_n1 \wedge i'=i \vee$

$j'=j+_n1 \wedge i'=i) \Rightarrow ?(i,j,i',j')$

3) $\$\ i,j,i',j'\ \ ?(i,j,i',j') \wedge !(i',j',i,j) \wedge (i'=i-_m1 \wedge j'=j \vee i'=i+_m1 \wedge j'=j \vee j'=j-_n1 \wedge i'=i \vee$

$j'=j+_n1 \wedge i'=i) \Rightarrow \neg?(i,j,i',j') \wedge \neg!(i',j',i,j)$

Эта L-программа состоит из трёх *правил*. Каждое правило имеет левую часть (*условие* в виде L-формулы) и правую часть (*действие*, L-формула специального вида [2]), которые объединены знаком $\Rightarrow$. Стоящая перед правилами конструкция $\$\ i,j,i',j'$ называется *синхронизатором* по переменным i,j, i',j'. Первое правило моделирует инициализацию операции посылки сообщения процессом с координатами (i,j) процессу с координатами (i',j'). Благодаря синхронизатору $\$\ i,j,i',j'$ это правило срабатывает для всех значений переменных i,j,i',j', для которых истинно его условие. Тем самым моделируется инициализация посылки

сообщения каждым процессом (i,j) каждому из четырёх своих соседей – (i',j'). (В реальной системе это соответствует рассылке процессом сообщений своим соседям в любом порядке.) L-программа работает по шагам. На первом её шаге может сработать только первое правило, условия остальных правил будут тождественно ложны.

Второе правило может исполниться только после первого и моделирует инициализацию операции приёма сообщения каждым процессом (i,j) от каждого из четырёх своих соседей – (i',j'). В реальной системе это соответствует любому порядку приёма сообщений процессом от своих соседей. Третье правило может исполниться только после второго и моделирует завершение всех (благодаря синхронизатору) операций приёма и передачи сообщений. Затем снова исполняется первое правило, и т.д.

3. ***Модели взаимодействия процессов со своими соседями в поперечных и продольных кольцах тора.***

1) $\$\, i,j,i' \quad \neg!(i,j,i',j)\wedge(i'=i+_m 1) \Rightarrow !(i,j,\, i',j)$

2) $\$\, i,j,i' \quad \neg?(i,j,i',j)\wedge!(i',j,i,j)\wedge(i'=i-_m 1) \Rightarrow ?(i,j,\, i',j)$

3) $\$\, i,j,i' \quad ?(i,j,i',j)\wedge!(i',j,i,j)\wedge(i'=i-_m 1) \Rightarrow \neg?(i,j,i',j)\wedge\neg!(i',j,i,j)$

Первое правило моделирует инициализацию посылки сообщения процессом с координатами (i,j) своему нижнему соседу с координатами $(i+_m 1,j)$. Синхронизатор $\$\, i,j,i'$ вынуждает осуществить это каждый процесс (i,j). Второе правило моделирует инициализацию процессом (i,j) приёма сообщения от верхнего соседа, процесса $(i-_m 1,j)$. Синхронизатор $\$\, i,j,i'$ вынуждает это проделать каждый процесс (i,j). Третье правило моделирует завершение всех операций приёма и передачи в поперечных кольцах тора. Как и в предыдущей модели, правила могут срабатывать циклически, вслед друг за другом (первое, второе, третье, снова первое и т.д.). Аналогичным образом моделируется взаимодействие процессов со своими соседями в продольных кольцах тора:

1) $\$\, i,j,j' \quad \neg!(i,j,i,j')\wedge(j'=j+_m 1) \Rightarrow !(i,j,\, i,j')$

2) $\$\, i,j,j' \quad \neg?(i,j,i,j')\wedge!(i,j',i,j)\wedge(j'=j-_m 1) \Rightarrow ?(i,j,\, i,j')$

3) $\$\, i,j,j' \quad ?(i,j,i,j')\wedge!(i,j',i,j)\wedge(j'=j-_m 1) \Rightarrow \neg?(i,j,i,j')\wedge\neg\, !(i,j',i,j)$

В случаях поперечных и продольных колец процесс передаёт сообщение следующему в кольце процессу, а принимает от предыдущего. Если объединить две последние L-программы, будет моделироваться взаимодействие и в поперечных, и в продольных кольцах, осуществляемые относительно друг друга асинхронно. Выбор правил, срабатываемых на одном шаге L-программы, из этих двух групп правил происходит недетерминированно.

4. ***Асинхронная модель взаимодействия процессов со всеми своими соседями.***

1) $\neg!(i,j,\ i\text{-}_m1,j)\wedge\neg!(i,j,\ i+_m1,j)\wedge\neg!(i,j,\ i,j\text{-}_m1)\wedge\neg!(i,j,\ i,j+_m1) \Rightarrow$
$i,j,\ i\text{-}_m1,j)\wedge!(i,j,\ i+_m1,j)\wedge!(i,j,\ i,j\text{-}_m1)\wedge!(i,j,\ i,j+_m1)$

2) $\neg?(i,j,\ i\text{-}_m1,j)\wedge\neg?(i,j,\ i+_m1,j)\wedge\neg?(i,j,\ i,j\text{-}_m1)\wedge\neg?(i,j,\ i,j+_m1)\wedge$
$!(i\text{-}_m1,j,i,j)\wedge!(i+_m1,j,i,j)\wedge!(i,j\text{-}_m1,i,j)\wedge!(i,j+_m1,i,j) \Rightarrow$
$i,j,\ i\text{-}_m1,j)\wedge?(i,j,\ i+_m1,j)\wedge?(i,j,\ i,j\text{-}_m1)\wedge?(i,j,\ i,j+_m1)$

3) $\$\ i',j'\ \ ?(i,j,i',j')\wedge!(i',j',i,j)\wedge(i'=i\text{-}_m1\wedge j'=j\vee i'=i+_m1\wedge j'=j\vee j'=j\text{-}_n1\wedge i'=i\vee j'=j+_n1\wedge i'=i) \Rightarrow$

Первое правило на одном шаге L-программы может исполниться для произвольного множества процессов (i,j), для которых не инициализировались операции посылки сообщений всем своим четырём соседям. В результате исполнения правила такая инициализация моделируется. Второе правило на одном шаге L-программы может исполниться для произвольного множества процессов (i,j), для которых не инициализировались операции приёма сообщений от всех четырёх соседей, а операции посылки сообщений от них процессу (i,j) уже инициализированы. В результате исполнения правила моделируется инициализация приёма сообщений процессом (i,j) от всех своих соседей. Третье правило моделирует завершение операций приёма и передачи для всех (благодаря синхронизатору $\$\ i',j'$) соседей процессов (i,j), для которых такие операции были инициализированы. Число таких процессов (i,j) выбирается недетерминированно. Данная модель не предполагает глобальной синхронизации процессов, как в модели 1. Она учитывает различную скорость функционирования процессов.

Анализ нетупиковости. *Нетупиковым* состоянием системы S называются её заключительное состояние из $fin[S]$, в котором справедливо $\forall i,j,i',j' (\neg?(i,j,i',j') \wedge \neg!(i',j',i,j))$, т.е. ни один из процессов не имеет незавершённых операций посылки и приёма сообщений. Пусть $D \subseteq Q$ – это множество нетупиковых состояний. Система переходов S относительно заданного множества начальных состояний $I \subseteq Q$, если существует такое множество состояний $W \subseteq Q$, что справедливо следующее: 1) $I \subseteq W$; 2) $\forall q',q \in W ((q',q) \in T \ \& \ q' \in W) \Rightarrow q \in W)$. т.е. W – инвариант; 3) $W \cap fin[S] \subseteq D$.

Алгоритмы из [2] позволяют получить логические формулы, выражающие в языке L отношение перехода T, множество $fin[S]$ для моделей в виде L-программ. Если множества W и D также выражаются L-формулами, то анализ нетупиковости моделируемых систем сводится к доказательству истинности формул логики предикатов первого порядка. Для систем с регулярными схемами взаимодействия удаётся находить компактные формулы-инварианты, независящие от числа процессов, что в частности, показывалось в [3]. Всё это обеспечивает независимость анализа нетупиковости от размера системы.

Список литературы

1. Крицкий С.П. Модель асинхронных вычислений в структурах и языки программирования // Методы трансляции. – Ростов н/Д, 1981. – С. 92–100.

2. Крицкий С.П., Панков С.В. О верификации асинхронных программ продукционного типа // Программирование. – 1994. – № 5. – С. 40–52.

3. Панков С.В. Научный сервис в сети ИНТЕРНЕТ: материалы Всерос. науч. конф. Новороссийск, 19–24 сент. 2007 г. – М.: Изд-во МГУ, 2007. – С. 85–88.