УДК 681.5

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ЧИСЛЕННОГО РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В СИСТЕМАХ С РАСПРЕДЕЛЁННОЙ ПАМЯТЬЮ С ТОПОЛОГИЕЙ РЕШЁТКА-ТОР

Фельдман Л.П., Назарова И.А., Кожухов А.Е.
Донецкий Национальный Технический Университет

Рассматриваются параллельные алгоритмы численного решения задачи Коши для систем обыкновенных линейных дифференциальных уравнений в системах с распределённой памятью и топологией типа решётка–тор. Получены оценки времени выполнения, ускорения и эффективности распараллеливания метода Рунге-Кутта с учётом коммуникационных затрат на обмены данными. Показаны зависимости характеристик параллельных алгоритмов от размерности решётки, сложности исходной задачи и числа шагов метода Рунге-Кутта.

В настоящее время имеют место бурное развитие науки и, особенно, рост сложности решаемых человеком практических задач. Во многих случаях процесс решения задач такого типа сопряжен с большими материальными и вычислительными затратами, если вообще возможен. В таких случаях оправдано и необходимо применение наиболее эффективных вычислительных методов и их эффективная реализация в высокопроизводительных вычислительных системах [1].

Многие сложные задачи (наиболее часто – моделирование сложных вычислительных систем) сводятся к обыкновенным дифференциальным уравнениям и их системам. Огромный практический интерес связан с задачами моделирования сложных динамических систем с сосредоточенными параметрами.

Решение вышеописанных задач моделирования имеет особый интерес в случае систем обыкновенных линейных дифференциальных уравнений [2]. Этот интерес обусловлен возможностью сведения алгоритма решения к последовательности матрично–векторных арифметических операций, которые могут быть эффективно распараллелены в системах с распределённой памятью.

Численное решение задачи Коши вида (1) для системы обыкновенных линейных дифференциальных уравнений [2]:

$$\frac{dx}{dt} = A \cdot x + f(t)$$
$$x(t_0) = x^{(0)} = (x_1^0, x_2^0, \ldots, x_m^0)^T \quad (1)$$

Решение задачи вида (1) может быть получено по правилу Рунге-Кутта и определяется формулой (2).

$$x^{(n+1)} = x^{(n)} + \frac{1}{6} \cdot (k_1^{(n)} + 2k_2^{(n)} + 2k_3^{(n)} + k_4^{(n)})$$
$$k_1^{(n)} = \tau (A \cdot x^{(n)} + f^{(n)})$$
$$k_2^{(n)} = \tau (A \cdot (x^{(n)} + 0.5k_1^{(n)}) + f^{(n+0.5)}) \quad (2)$$
$$k_3^{(n)} = \tau (A \cdot (x^{(n)} + 0.5k_2^{(n)}) + f^{(n+0.5)})$$
$$k_4^{(n)} = \tau (A \cdot (x^{(n)} + 0.5k_3^{(n)}) + f^{(n+1)})$$

Формула (2) определяет итерационный процесс, который сводится к операциям матрично-векторной арифметики. На каждом шаге процесса вычислений происходит расчёт промежуточных векторов k_i, порядок вычисления которых строго последователен. Как результат, параллелизм может быть реализован только при расчёте каждого из промежуточных векторов в процессе вычисления результатов матрично–векторных операций.

Альтернативный метод решения задачи Коши по правилу Рунге-Кутта – определение полного оператора (матрицы) перехода. Эта матрица рассчитывается предварительно и единожды (до выполнения итераций метода) по формуле (3). Таким образом, весь итерационный процесс сводится к серии последовательных умножений матрицы на вектор [2].

$$x^{(n+1)} = A' \cdot x^{(n)}$$
$$A' = (E + \tau A(E + \frac{\tau A}{2}(E + \frac{\tau A}{3}(E + \frac{\tau A}{4})))) \quad (3)$$

Преимущество подхода с расчётом матрицы перехода проявляется при большом количестве итераций метода Рунге–Кутта, вычислительная сложность последовательного алгоритма характеризуется формулой (4). Соответственно, для малого количества итераций лучшие показатели будет демонстрировать подход с расчётом промежуточных величин, сложность последовательного алгоритма которого представлена формулой (5).

В формулах величина m – размерность исходной задачи, N – число шагов метода Рунге–Кутта, τ – длительность модельного такта времени.

$$T_{RKStepSolveS}[m,N] = N \cdot \left(15 \cdot m\tau + 8 \cdot m^2\tau\right) \quad (4)$$

$$T_{RKTransMatrixS}[m,N] = 4 \cdot m\tau + 4 \cdot m^2\tau + 6 \cdot m^3\tau + 2N \cdot m^2\tau \quad (5)$$

В распределённых параллельных вычислительных системах наиболее трудоёмкими с вычислительной и коммуникационной точек зрения являются операции вычисления матричного и матрично–векторного произведений. Наиболее подходящими в плане коммуникационных затрат (при одинаковых вычислительных затратах) для таких операций являются топологии решётка (решётка-тор и её модификации) и гиперкуб.

Для топологии типа решётка вычисление матрично–векторных арифметических операций можно свести к выполнению операций аналогичного типа над блоками матриц. Сложение матриц сводится к однократному матричному сложению соответствующих блоков.

Умножение матриц является более сложным и может быть эффективно выполнено с помощью ряда алгоритмов семейств Кэннона и Фокса [3]. Алгоритмы семейства Фокса не меняют отображения блоков матриц-операндов и матрицы-результата на узлы вычислительной решётки. Особенностью алгоритмов Кэннона является смена отображения блоков одной из матриц–операндов или матрицы–результата на узлы вычислительной решётки. Каждый из алгоритмов Кэннона сохраняет отображение блоков соответствующей матрицы на узлы вычислительной решётки [4].

Наиболее эффективным среди рассматриваемых алгоритмов вычисления матричного произведения является алгоритм Кэннона, сохраняющий отображение блоков матрицы–результата.

Умножение матрицы на вектор может быть сведено к умножению матрицы на матрицу. При выполнении определённых условий эта операция может быть осуществлена по модифицированному алгоритму вычисления произведения матриц.

В системах с распределённой памятью время передачи данных пропорционально их объёму (справедливо для больших объёмов передаваемых данных) [1]. При уменьшении объёма передаваемых данных происходит снижение времени выполнения параллельного алгоритма. Таким образом, параллельный алгоритм умножения матрицы на вектор в общем случае будет эффективнее умножения матрицы на матрицу. Такое соотношение времён выполнения

алгоритмов может нарушаться, если в алгоритме умножения матрицы на вектор будут передаваться только блоки матрицы.

Реализация метода Рунге–Кутта с вычислением промежуточных коэффициентов распараллеливается на этапах выполнения матрично–векторных операций. Реализация метода Рунге–Кутта с вычислением матрицы перехода распараллеливается аналогичным образом.

Пусть в дополнение к введённым ранее обозначениям будут использованы: k – размерность блока вектора, p – число процессоров, t_s – временные затраты подготовки сообщения к отправке (латентность), β - пропускная способность канала передачи данных, t_w – время передачи слова данных, γ – относительная доля служебных данных (которыми обрамляется передаваемое сообщение) по отношению к размеру передаваемого сообщения.

Величины времени выполнения параллельных алгоритмов с вычислением промежуточных коэффициентов и матрицы перехода определяются формулами (6) и (7) соответственно.

$$T_{RKStepSolve}[N,k] = N \cdot \left(12 \cdot k\tau + 4 \cdot \left[p \cdot (t_s + kt_w + \gamma kt_w) + 3 \cdot p \cdot k^2\tau + 4 \cdot (t_s + kt_w + \gamma kt_w \cdot \left\lfloor \frac{p}{2} \right\rfloor) \right] \right) \quad (6)$$

$$\begin{aligned} T_{RKTransMatrix}[N,k] &= N \cdot \left[p \cdot (t_s + kt_w + \gamma kt_w) + 3 \cdot p \cdot k^2\tau + 4 \cdot (t_s + kt_w + \gamma kt_w \cdot \left\lfloor \frac{p}{2} \right\rfloor) \right] + \\ &+ 4 \cdot k\tau + 4 \cdot k^2\tau + 3 \cdot \left(2 \cdot t_s + 2 \cdot k^2 t_w + p \cdot (t_s + k^2 t_w + \gamma k^2 t_w) + 2 \cdot p \cdot k^3\tau + 2 \cdot \gamma k^2 t_w \cdot \left\lfloor \frac{p}{2} \right\rfloor \right) \end{aligned} \quad (7)$$

Величины ускорений по сравнению с последовательными вариантами параллельных алгоритмов задаются формулами (8) и (9) соответственно.

$$S_m(N,m,p) \underset{p\to\infty}{\cong} \frac{m \cdot (15 + 8 \cdot m) \cdot \tau}{(4 \cdot t_s) \cdot p + 4 \cdot (4 \cdot t_s + mt_w \cdot (1+\gamma))} \quad (8)$$

$$S_m(N,m,p) \underset{p\to\infty}{\cong} \frac{2 \cdot m \cdot (2 + 3 \cdot m^2 + m \cdot (2+N)) \cdot \tau}{p \cdot (3+N) \cdot t_s + (6 + 4 \cdot N) \cdot t_s + mNt_w \cdot (1+\gamma)} \quad (9)$$

Величины эффективности использования процессоров параллельными алгоритмами описываются формулами (10) и (11) соответственно.

$$E_m(N,m,p) \underset{p\to\infty}{\cong} \frac{m \cdot (15 + 8 \cdot m) \cdot \tau}{(4 \cdot t_s) \cdot p^2 + 4 \cdot (4 \cdot t_s + mt_w \cdot (1+\gamma)) \cdot p} \quad (10)$$

$$E_m(N,m,p) \underset{p\to\infty}{\cong} \frac{2\cdot m\cdot\left(2+3\cdot m^2+m\cdot\left(2+N\right)\right)\cdot\tau}{p^2\cdot\left(3+N\right)\cdot t_s+p\cdot\left(6+4\cdot N\right)\cdot t_s+p\cdot mNt_w\cdot(1+\gamma)} \qquad (11)$$

Пусть алгоритм решения задачи Коши для системы линейных ОДУ с вычислением промежуточных данных по методу Рунге–Кутта будет обозначен через RKStepSolve. Тогда альтернативный алгоритм с предварительным вычислением матрицы перехода, реализующий метод Рунге–Кутта, будет обозначен через RKTransMatrix.

С учётом вышеприведённых формул могут быть сделаны сравнительные выводы по поводу параллельных алгоритмов решения систем линейных ОДУ:

1) для больших значений p преимущество алгоритма RKTransMatrix над алгоритмом RKStepSolve реализуется за счёт меньшего времени выполнения итерации метода Рунге–Кутта;

2) для относительно малых значений p время выполнения алгоритма RKTransMatrix будет хуже аналогичной величины для алгоритма RKStepSolve в связи с затратами на предварительное вычисление матрицы перехода;

3) для больших значений m преимущество алгоритма RKTransMatrix над алгоритмом RKStepSolve реализуется за счёт меньшего времени выполнения итерации метода Рунге–Кутта;

4) для относительно малых значений m время выполнения алгоритма RKTransMatrix будет хуже аналогичной величины для алгоритма RKStepSolve в связи с затратами на предварительное вычисление матрицы перехода;

5) для малого числа шагов метода Рунге–Кутта параллельный алгоритм RKTransMatrix будет показывать худшие показатели, чем параллельный алгоритм RKStepSolve;

6) алгоритм RKTransMatrix имеет меньшие временные затраты на выполнение итерации по сравнению с RKStepSolve.

Литература

1. Гергель В.П. Основы параллельных вычислений для многопроцессорных вычислительных систем. – Н.Новгород, ННГУ, 2001.
2. Фельдман Л.П. Параллельные алгоритмы численного решения систем линейных обыкновенных дифференциальных уравнений.
3. Gupta A., Kumar V. Scalability of Parallel Algorithms for Matrix Multiplication. Department of Computer Science, University of Minnesota Minneapolis, 1991.
4. Li J. A Poly-Algorithm for Parallel Dense Matrix Multiplication on Two-Dimensional Process Grid Topologies. Department of Computer Science, Mississippi State University, 1996.

Получено 01.06.07