

ИССЛЕДОВАНИЕ МЕТОДОВ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ ПРОГРАММЫ ПРИ ИСПОЛЬЗОВАНИИ РАСПРЕДЕЛЕННЫХ СИСТЕМ

Аннотация: В данной работе было проведено исследование методов повышения быстродействия программ при использовании распределенных систем. Данная методика апробирована на примере задачи моделирования процесса роста трещины пластины.

Ключевые слова: распределенная система, усталостная трещина, трехзвенная архитектура, фактор ускорения

Вступление

В предыдущей статье [1] рассматривалась задача, для решения которой было достаточно мощности одной физической машины. Однако иногда возникают ситуации, когда решение может быть достигнуто только при использовании распределенных вычислений.

Постановка задачи

Распределенные вычисления являются частным случаем параллельных вычислений. Они используются для решения трудоемких вычислительных задач с использованием нескольких компьютеров, объединенных в параллельную вычислительную систему. При их проектировании, как и при проектировании программного обеспечения (ПО) в общем, все же следует учитывать специфические особенности конкретной задачи.

Рассмотрим задачу о росте *усталостной трещины* в тонкой прямоугольной пластине и на ее примере спроектируем распределенную систему.

Конкретизация задачи.

Рассматривается тонкая пластина (рисунок 1) длиной h и шириной W , содержащая центральную трещину начальной длины $2\ell_0$. На краях пластины вдоль оси OY перпендикулярно по отношению к берегам трещины приложено одноосное циклическое нагружение

$$\tilde{\sigma} = \sigma_a g(n), \quad (1)$$

где σ_a – амплитуда циклического напряжения; $g(\cdot)$ – известная периодическая функция числа циклов нагружения n ($n = ft$); t – физическое время; f – частота нагружения. Берега трещины свободны от нагрузки.

Задача состоит в определении зависимости, устанавливающей взаимосвязь между переменными, характеризующими кинетику роста трещины, параметрами нагружения и содержащими набор материальных констант \tilde{N}_i ($i = 1, k$), в виде:

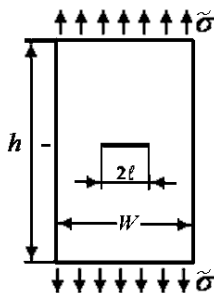


Рис. 1 – Конфигурация пластины с трещиной

$$\frac{d\ell}{dn} = F(\sigma_a, \ell, n, h, W, C_i) \Rightarrow \ell = \ell(\sigma_a, n, h, W, C_i). \quad (2)$$

Адаптация данной задачи к архитектуре распределенных систем

Определение потенциального параллелизма алгоритма.

Решение задачи строится на основе использования двухстадийной модели роста усталостной трещины [2] и сводится к совместному решению краевой задачи теории упругости с подвижной границей и эволюционного уравнения накопления усталостных повреждений.

Алгоритм решения задачи можно распараллелить, заменив последовательное, при помощи итераций, решение краевой задачи о напряженном состоянии (НС) пластины с движущимся фронтом разрушения, на параллельное решение множества k задач о НС пластины с трещиной длиной $\ell_i \in [\ell_0, \ell_k]$.

Распределение напряжений у фронта трещины

Решение задачи о НС пластины конечных размеров для текущей длины трещины ℓ_i получено на основе численно-аналитического метода граничных коллокаций [2], и при циклическом нагружении уравнение принимает вид

$$\Delta\sigma_{yy} = \frac{\sigma_a}{\sqrt{2}} \sqrt{\frac{\ell_i}{z - \ell_i}} \cdot \left(1 + \sum_{n=1}^N 2A_n z^n \right), \quad (3)$$

где $\Delta\sigma_{yy}$ – размах напряжений отрыва в окрестности вершины усталостной трещины длиной ℓ_i , $z = x + iy$ – комплексное число. Коэффициенты A_n определяются численно из решения замкнутой системы алгебраических уравнений при ограничении ряда в выражении (2) конечным числом членов при удовлетворении граничным условиям в n точках коллокаций на внешнем контуре пластины.

Путем аппроксимации множества k численных решений (2) для трещин дискретной длины $\ell_0 \leq \ell \leq \ell_k$, получаем закон распределения на-

пряжений в окрестности усталостной трещины произвольной длины ℓ , в виде

$$\Delta\sigma_{yy} = \frac{\sigma_a}{\sqrt{2}} \sqrt{\frac{\ell(n)}{z - \ell(n)}} \cdot f(h/W, \ell/W), \quad (4)$$

где $f(h/W, \ell/W)$ – корректирующая функция, учитывающая влияние граничных условий.

Кинетика роста центральной трещины в прямоугольных пластинах

Решение задачи сводится к решению системы уравнений [3]:

$$\begin{cases} \frac{d\ell}{dn} = \left(1 + \frac{1}{q}\right) D \frac{1}{[\lambda(\ell(n))]^{\frac{q}{2}-1}} \cdot \left(\sigma_a \sqrt{\ell} \cdot f(h/W, \ell/W)\right)^q \\ n_* = \frac{1}{(1+q)D} \left[\frac{1}{\sigma_a}\right]^q \left[\frac{2\lambda(\ell_0)}{\ell_0}\right]^{\frac{q}{2}} \cdot f(h/W, \ell_0/W)^{-q} \end{cases}, \quad (5)$$

где первое уравнение описывает стадию роста трещины, а второе – длительность инкубационного периода, $\lambda(\ell)$ – длина пластической зоны у вершины трещины, D, q – коэффициенты, определяющие сопротивление материала усталостному разрушению

Длительность стадии роста усталостной трещины определяем из уравнения вида

$$n = n_* + \frac{\pi^{q-2}}{\left(1 + \frac{1}{q}\right) D (4\sigma_T)^{q-2} \sigma_a^2 \int_{\ell_0}^{\ell} \left(f\left(\frac{h}{W}, \frac{\ell}{W}\right)\right)^2 \ell d\ell}. \quad (6)$$

Конечный распараллеленный алгоритм решения задачи будет выглядеть следующим образом:

Как видно с рисунка 2, только Модуль 1 обладает уникальной последовательностью выполняемых операций. Задачи, которые выполняют Модули 2 – 4, могут быть с легкостью распределены на другое количество вычисляющих единиц. Выбранное количество есть оптимальным с точки зрения загрузки модулей и сложности вычисляющей системы.

Адаптация последовательных фрагментов к возможности их параллельного выполнения.

Все алгоритмы работы с матрицами заменяем на параллельно-выполняемые. Они будут осуществляться на дополнительном 5-м вычислительном модуле.

Для процесса распараллеливания существует 2 подхода:

- с дублированием данных (при использовании каналов передачи с низкой пропускной способностью);
- с минимальным объемом хранимых данных (высокоскоростные каналы).

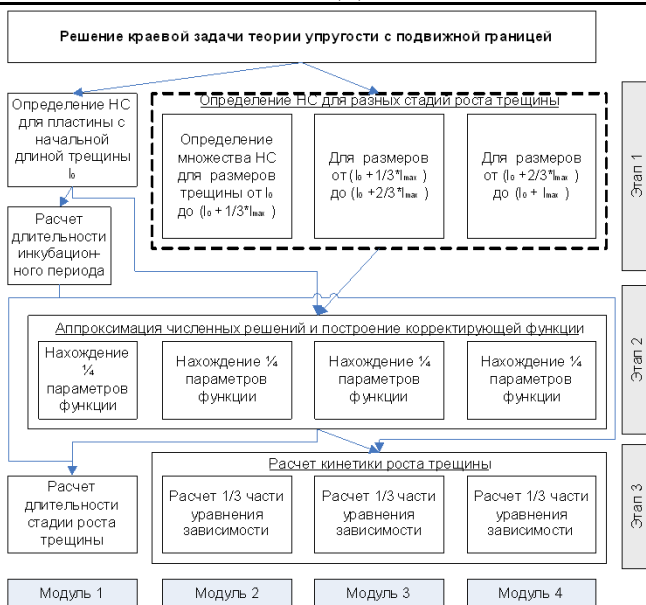


Рис. 2 – Схема распараллеливания алгоритма

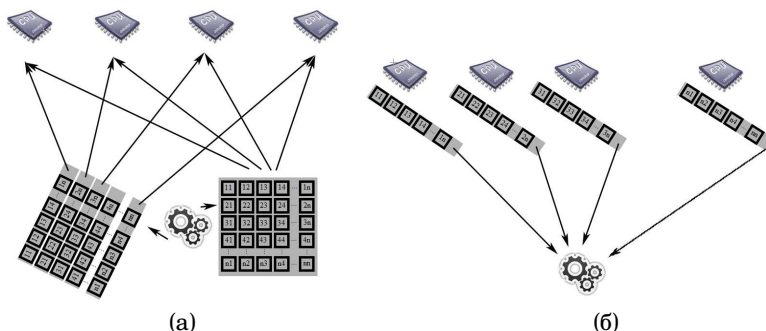


Рис. 3 – Схема передачи данных с дублированием для выбранного 4 - ядерного модуля – опровка (a) и сбор данных (b)

Так как все ядра будут внутри одного вычислительного модуля, то наш канал передачи данных считается скоростным. Следовательно, будем использовать подход с минимальным дублированием.

Выбор архитектуры и структурной схемы

Для данной задачи подходит трехзвенная архитектура [4].

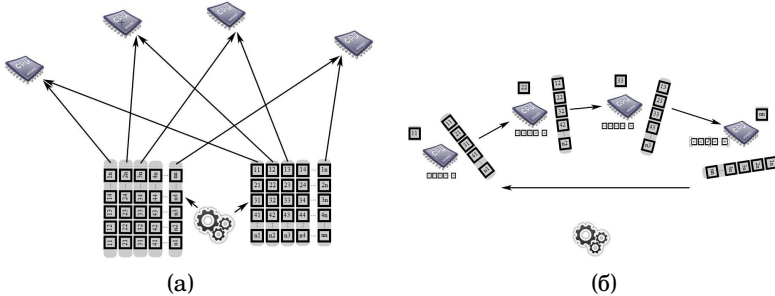


Рис. 4 – Схема передачи данных с минимальным дублированием для выбранного 4 – ядерного модуля - оправка (а) и обмен частичными данными между элементами до момента получения конечного результата (б)

Тонкий клиент будет отвечать только за ввод-вывод начальным параметров задачи и отображения результатов расчетов. Обмен данными будет происходить только с сервером бизнес – логики.

Сервер бизнес – логики будет отвечать за распределение подзадач на другие вычислительные модули, сбор и компоновку конечных данных, передачу конечных результатов клиенту.

Серверы (они же вычислительные модули) – 5 сервисов, которые будут решать специфические задачи.

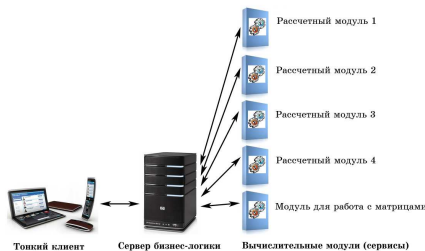


Рис. 5 – Предполагаемая архитектура распределенной системы

Анализ оценки производительности для данной задачи при использовании распределенных систем

Понятие “фактор ускорения” (“speedup factor”) [5,6] это отношение времени, которое тратит на выполнение работы один процессор к времени, которое тратит на эту же работу многопроцессорная система.

$$S(p) = T_S / T_P, \quad (7)$$

Общее время выполнения T_P состоит из серийного времени и максимального из тех, что мы распределили на несколько процессоров (выполняясь они все одновременно, но ждать нужно самого медленного).

Фактор ускорения рассчитывается по формуле:

$$S(p) = \frac{t_S}{ft_S + (1-f)t_S/p} = \frac{p}{1 + (p-1)f}, \quad (8)$$

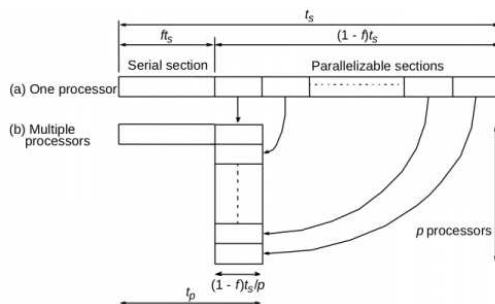


Рис. 6 – Схема выполнения программы для нескольких процессоров

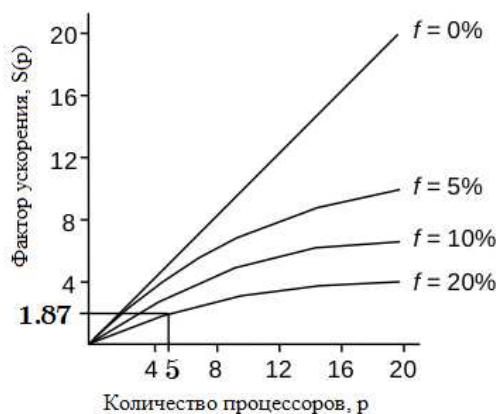


Рис. 7 – График зависимостей фактора ускорения от количества процессоров для алгоритмов разной степени распараллеливания

Для нашей задачи мы использовали 4 модуля с 1 ядром для каждого этапа и 1 модуль с 4 ядрами для операций над матрицами – всего 5 процессоров. Процентная часть последовательных расчетов (f) – 20%.

Рассчитанный фактор ускорения для данной задачи – 1,87.

Выводы

В данной работе было проведено исследование методов повышения быстродействия программ при использовании распределенных систем. Данная методика апробирована на примере задачи моделирования процесса роста трещины пластины. Для этого был определен потенциаль-

ный параллелизм алгоритма, построена схема возможного распараллеливания. Последовательные алгоритмы были заменены параллельными, представлена архитектура и структурная схема для практической реализации.

Также проведен анализ оценки производительности для данной задачи при использовании распределенных вычислений. Теоретические расчеты показали, что с использованием распределенной архитектуры данная задача будет решаться на 87% быстрее, чем при использовании последовательных вычислений.

Литература

1. Крылов Е.В., Плащинская А.В., Базурин П.А. Исследование методов повышения быстродействия программы при использовании операций над матрицами большой размерности // Адаптивные системы автоматического управления. -2011.- . 18(38).
2. Newman J.C. An improved method of collocation for the stress analysis of cracked plate with various shaped boundaries. – NASA TN D-6376, 1971. – P45.
3. Hudson C.M., Scardina J.T. Effect of stress ratio on fatigue-crack growth in 7075-T6 aluminum-alloy sheet // NASA TMX- 60125, 1967. – P24.
4. Приходько С.А. “Обзор и классификация распределенных систем”. – Донецкий НТУ.
5. Давлеткалиев Рахим “Параллельные вычисления: введение, ускорение, пример”. – “Your vision community”.
6. Афанасьев А.П., Посыпкин М.А., Хританков А.С “Аналитическая модель оценки производительности распределенных систем”. – журнал “Программные продукты и системы”.

Отримано 21.11.2011 р.