NickWare Group Science
Бабич Н. А.
02.12.2017

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В СРЕДЕ MATLAB

История появления многоядерных микропроцессоров

Первые многоядерные микропроцессоры представляли собой самые простые схемы – два процессорных ядра, размещённые на одном кристалле, без разделения каких-либо ресурсов, кроме шины памяти. В современных же многоядерных микропроцессорах вычислительные ядра совместно используют кэш третьего или второго уровня, а также другие технологии, повышающие производительность (например, возможность разделять работу одного ядра на два потока) [1].

Сейчас трудно представить современную вычислительную систему, работающую на одноядерном микропроцессоре. Почти все устройства, которыми люди пользуются ежедневно, работают на чипах с двумя и более ядрами, не говоря о специализированных системах, используемых, например, учёными.

Первым массовым многоядерным микропроцессором стал POWER4, содержащий два ядра PowerPC на одной подложке, которые использовали общий кэш второго уровня. Он был выпущен компанией IBM в 2001 году, имел 174 миллиона транзисторов и производился по 180-нанометровой технологии [2].

В 2005 году компания Sun Microsystems (была поглощена компанией Oracle в 2010 году) представила первый двухъядерный микропроцессор архитектуры SPARC – UltraSPARC IV. Имел он 66 миллионов транзисторов и производился по 130-нанометровой технологии [3]. В том же году компания AMD показала первый двухъядерный микропроцессор для персональных компьютеров Athlon 64 X2 (архитектура x86-64, 154 миллионов транзисторов, 90 нанометров) [4], а позднее компания Intel представила Pentium D (архитектура x86-64, 230 миллионов транзисторов, 90 нанометров). Он был построен на не очень удачной микроархитектуре NetBurst (имелись значительные проблемы с производительностью и тепловыделением) и, по сути, состоял из двух микропроцессоров на одной подложке с общим кэшем третьего уровня [5].

Таким образом, современная эра многоядерных микропроцессоров началась в 2005-2006 годах. Дальнейшее развитие технологий позволило значительно увеличить количество ядер. Если говорить о микропроцессорах общего назначения с архитектурой x86, в настоящее время выпускаются модели с 32 ядрами, по два потока на каждое ядро (серверные процессоры AMD Naples) [6].

Многоядерность и MATLAB

Необходимость параллельных вычислений при моделировании сложных процессов, например, в физике, вполне закономерна. Если задачу можно разделить на несколько независимых подзадач и каждую из них решать на отдельном процессорном ядре, то можно значительно сократить время, которое необходимо потратить на расчёты, а также снизить энергопотребление вычислительной системы.

Среда MATLAB, разрабатываемая компанией The MathWorks, не имела поддержки параллельных вычислений до 2007 года. Ранние эксперименты показали, что использование параллельных вычислений на существовавших тогда многопроцессорных компьютерах приводило не к увеличению скорости вычислений, а давало обратный эффект.

В 1987 году разработчики решили провести эксперимент, в котором использовался один из первых многопроцессорных компьютеров Intel iPCS с распределённой моделью памяти, состоявший из 128 процессорных узлов и управляющего компьютера, на котором запускалась среда MATLAB [7]. Решаемая задача легко разделяема на подзадачи – операции над матрицей можно производить параллельно для подматриц. Таким образом, матрицу брали из памяти управляющего компьютера, разделяли её на подматрицы, рассылали их на процессоры и потом собирали результаты обратно. Объём памяти управляющего компьютера был слишком мал, чтобы хранить в нём матрицы больших размеров, и получилось так, что рассылка подматриц на процессоры занимала больше времени, чем собственно вычисления. Выигрыш от распараллеливания получить не удалось. Эксперименты и исследования параллельных вычислений продолжались, но было принято решение не вводить такой возможности в среду MATLAB [7].

Но всё изменилось в 2007 году, когда компания The MathWorks решила добавить в среду MATLAB пакет Parallel Computing Toolbox. Без значительных изменений кода, этот пакет позволяет использовать параллельные процессы на ПК, графический процессор видеокарты (имеется поддержка CUDA), облачные вычисления или компьютерный кластер. Графические процессоры NVIDIA оптимизированы на выполнение математических операций с большими числами. Операция умножения двух чисел float выполняется на них за один такт [7].

В состав MATLAB также входит пакет Distributed Computing Toolbox, который отвечает за серверную часть системы удалённых распределённых вычислений. Планировщик MathWorks Job Manager координирует выполнение заданий и асинхронно

распределяет задачи по рабочим станциям – исполнителям. Он устанавливается на любой машине, находящейся в сети, и может обрабатывать задания различных пользователей различных платформ [7].

Задача описывается на языке среды MATLAB, поэтому можно использовать функции любых других пакетов, входящих в её состав. Вычисления с использованием пакетов, которые поддерживают параллельные вычисления, также будут выполняться быстрее.

Parallel Computing Toolbox предоставляет до 12 работников (вычислительных блоков MATLAB) для запуска приложений локально на многоядерном ПК [8].

На рисунке 1 приведена схема, объясняющая принцип работы Parallel Computing Toolbox с использованием. Каждый работник MATLAB запускается на отдельном ядре, производит свою часть вычислений и отправляет результат обратно в Client.

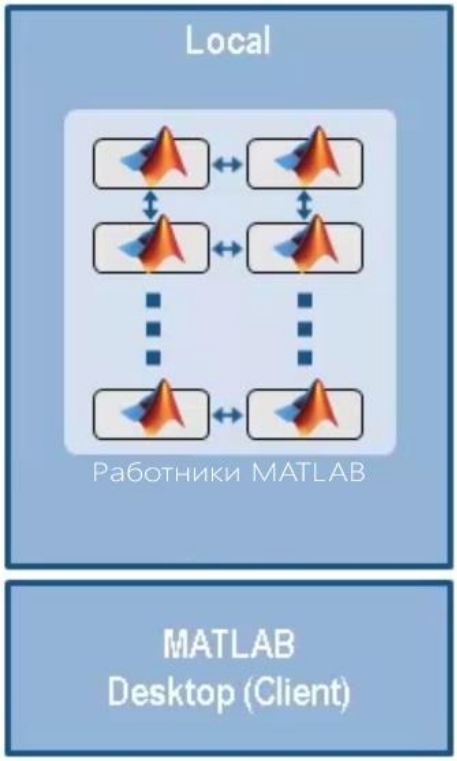

Рисунок 1 – Принцип работы параллельных вычислений в MATLAB

Таким образом, современные версии среды MATLAB позволяют без особого труда использовать все возможности современного «железа» для ускорения вычислений и экономии времени. Инструменты, позволяющие использовать сложные вычислительные системы (например, кластеры), делают среду MATLAB по-настоящему мощной, удобной и простой в использовании платформой для решения научных задач.

Экспериментальная проверка

Для экспериментальной проверки ранее изложенного материала был написан простой скрипт на языке среды MATLAB, решающий систему дифференциальных уравнений:

$$\frac{dV}{dt} = \frac{P \cdot cos\alpha}{m} - (c_{x\alpha 0} + A \cdot \alpha^2) \cdot \frac{q \cdot S}{m} - g \cdot sin\Theta,$$
$$\frac{d\Theta}{dt} = \frac{P \cdot sin\alpha}{m \cdot V} + \frac{c_{y\alpha}^{\alpha} \cdot q \cdot S \cdot \alpha}{m \cdot V} - \frac{g \cdot cos\Theta}{V},$$
$$\frac{d\omega_z}{dt} = \left(m_z^{\alpha} \cdot \alpha + m_z^{\omega_z} \cdot \frac{L}{V} \cdot \omega_z + m_z^{\delta_{II}} \cdot \delta_B\right) \cdot \frac{q \cdot S \cdot L}{I_z},$$
$$\frac{d\vartheta}{dt} = \omega_z,$$
$$\frac{dy}{dt} = V \cdot sin\Theta,$$
$$\frac{dx}{dt} = V \cdot cos\Theta,$$
$$\frac{dm}{dt} = -m_c.$$

Данная система дифференциальных уравнений описывает движение летательного аппарата, а перебираемое начальное значение ϑ_0 – ничто иное, как начальный угол тангажа.

Моделирование осуществлялось на интервале времени $[0; 6]$ с шагом $5 \cdot 10^{-3}$. Также осуществлялся перебор начального значения ϑ_0 на интервале $[0; 1.5]$ с шагом 10^{-4}. Таким образом, данная система параллельно моделировалась с разным начальным значением ϑ_0 в пределах указанного интервала.

Для распределения задачи по ядрам микропроцессора был использован параллельный цикл (parfor, входит в состав пакета Parallel Computing Toolbox). Фрагмент кода представлен ниже.

```
h = 0.005;                % шаг моделирования
T = 6;                    % время окончания моделирования
tic;                      % начало отсчёта времени выполнения
parfor i = 1:15000        % параллельный цикл
   X(i,:) = EulerS(50, 0.75, 0.5, i/10000, 150, 0.9, 1000, h, T);   % решение системы ДУ методом Эйлера
end
toc;                      % получение информации о времени выполнения
```

Итерации этого цикла выполняются на каждом из ядер в режиме конвейера. Например, если микропроцессор имеет четыре ядра, одновременно могут выполняться четыре итерации цикла. Если одно ядро завершает вычисления, на нём запускается следующая итерация, и так далее. Задача решалась на компьютере с восьмиядерным микропроцессором AMD FX-8370. Сначала под вычисления было задействовано одно ядро (без распараллеливания), затем два ядра, четыре, шесть и, наконец, все восемь ядер. То, как изменяется скорость решения поставленой задачи в зависимости от количества задействованных ядер, можно увидеть на рисунке 2.

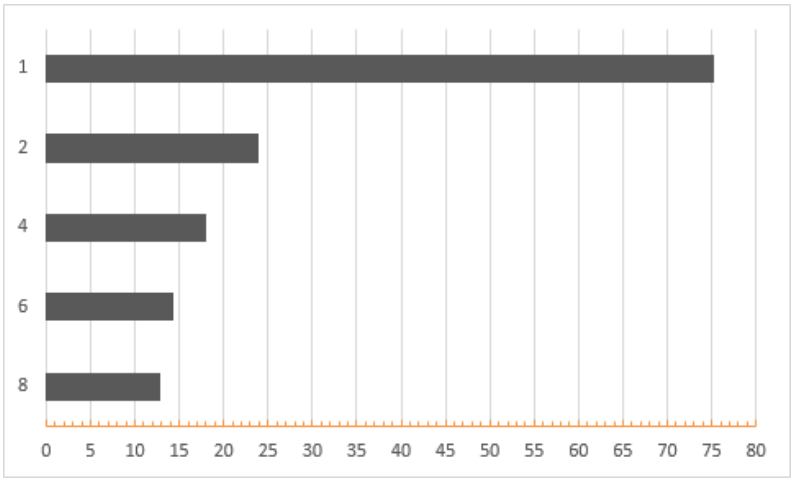

Рисунок 2 – Зависимость скорости решения задачи (горизонтальная ось, секунды) от количества задействованных ядер микропроцессора (вертикальная ось)

Таким образом, распараллеливание задачи на все восемь ядер дало увеличение скорости расчётов примерно в 6 раз.

Подводные камни

Рассмотрим аналогичную задачу, но с небольшими изменениями. Увеличим шаг моделирования системы и сделаем его равным 0.5, а шаг перебора начального угла тангажа ϑ_0 увеличим до 10^{-3}. Проведя аналогичный эксперимент, оказалось, что распараллеливание такой задачи приводит к совершенно неожиданному результату – к обратному. Это наглядно показано на рисунке 3.

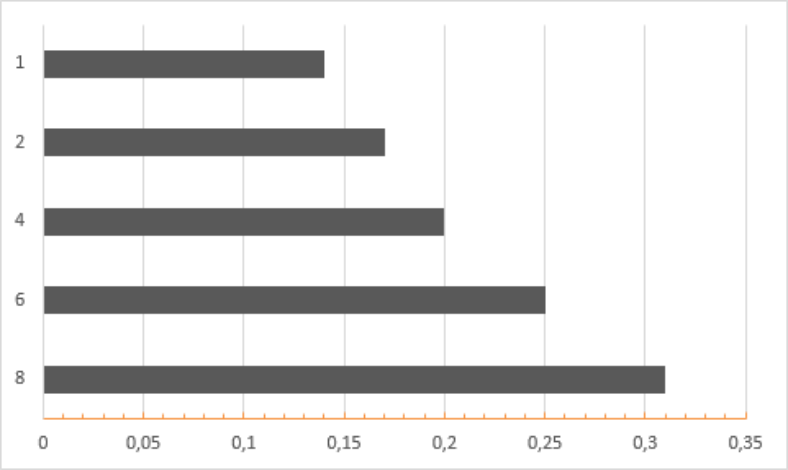

Рисунок 3 – Зависимость скорости решения задачи с увеличенными шагами (горизонтальная ось, секунды) от количества задействованных ядер микропроцессора (вертикальная ось)

Несмотря на то, что задача по-прежнему отлично разделяется на подзадачи и вычисления распараллеливаются, такой подход даёт снижение скорости вычислений. Так происходит из-за того, что время, затрачиваемое на обработку одной итерации значительно меньше времени, необходимого для их распределения по ядрам микропроцессора, а объём опыта недостаточно большой для того, чтобы получить хоть какую-то выгоду от распараллеливания. В данном случае намного оптимальнее будет использовать классический последовательный цикл.

Таким образом, идею параллельных вычислений можно применять не для любой задачи. Даже если она хорошо разделяется на подзадачи, могут возникнуть ситуации, когда этот подход приведёт не к ускорению, а к замедлению вычислений, либо, в лучшем случае, не даст никакого эффекта. Следует оценивать и чётко представлять, в каком случае имеет смысл распараллеливать вычисления, а в каком – оптимальнее будет этого не делать. Это

особенно актуально в следствие необходимости модифицировать (пусть и незначительно) уже имеющиеся алгоритм, чтобы его можно было запускать в параллельном режиме.

Кроме рассмотренного в статье параллельного цикла parfor, в среде MATLAB имеется множество других параллельных операторов и функций, например, специальные массивы и параллельные численные алгоритмы. Также среда MATLAB поддерживает не только многоядерность, но и многопроцессорность – возможность запускать задачи независимо на нескольких микропроцессорах одного компьютера.

Список использованных источников

1. Л. Черняк. Многоядерные процессоры и грядущая параллельная революция // «Открытые системы». 07.06.2007. URL: https://www.osp.ru/os/2007/04/4219910/ (дата обращения: 16.11.17)
2. Icons of Progress. Power 4. The first Multi-Core, 1Ghz Processor. URL: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/power4/ (дата обращения: 16.11.17)
3. Sun Microsystems UltraSPARC IV 1.05 GHz specifications. URL: http://www.cpu-world.com/CPUs/UltraSparc-IV/Sun%20Microsystems-UltraSparc%20IV%20SME1167ALGA-1050%20-%20SME%201167A%20LGA%201050%20MHz.html (дата обращения: 20.11.17)
4. AMD Athlon 64 X2 Dual-Core processor. URL: http://www.chiplist.com/AMD_Athlon_64_X2_Dual_Core_processor_Manchester_Rev_E4/tree3f-subsection--2118-/ (дата обращения: 21.11.17)
5. Спецификация продукции Intel Pentium D Processor 805. URL: https://ark.intel.com/ru/products/27511/Intel-Pentium-D-Processor-805-2M-Cache-2_66-GHz-533-MHz-FSB- (дата обращения: 30.11.17)
6. Новости о высокопроизводительных центральных процессорах Naples для серверов. URL: https://www.amd.com/ru/events/naples-tech-day (дата обращения: 30.11.17)
7. Матюшкин И. В. Моделирование и визуализация средствами MATLAB физики наноструктур. Москва: Техносфера, 2011. -168 с.
8. Официальный сайт Matlab. Parallel Computing Toolbox. URL: https://matlab.ru/products/parallel-computing-toolbox (дата обращения: 30.11.17)