Алексей Коченгин

Один из методов выбора процессора для вычислительной системы

Введение

В своей повседневной работе команда проектного офиса ПРОСОФТ сталкивается с новыми проектами, требующими проработки программно-аппаратной структуры. При проработке архитектуры систем возникает много проблем и ограничений, связанных, например, с подбором оборудования для размещения в металлической стойке, турникете или терминале, которое бы соответствовало ряду требований:

- технические требования заказчика;
- форм-фактор, который позволил бы конструкторам не только разместить оборудование в ограниченном пространстве, но и соблюсти эргономические требования;
- наличие современного интерфейса связи оборудования с вычислителем;
- наличие SDK (Software Development Kit – комплект средств разработки ПО) или API (Application Programming Interface – интерфейс прикладного программирования) для интеграции в проприетарное программное обеспечение. Можно сказать, что не все производители серьёзно относятся к этому пункту и в составе своей продукции предоставляют только закрытое пользовательское приложение, отсекая какие-либо возможности сторонней программной интеграции, а ведь интеграция оборудования в сторонние системы открывает дополнительный рынок сбыта, который вполне мог бы окупить затраты на разработку и описание внешнего API;
- наличие вычислителя или просто компьютера, удовлетворяющего требованиям по производительности работы в режиме 24/7/365 и имеющего пассивное охлаждение. Это важный момент, поскольку использование вентиляторов в таких системах приводит к скоплению пыли и, как следствие, к увеличению частоты и сложности обслуживания.

При проработке архитектуры систем почти все эти требования носят детерминированный характер, и от архитектора системы или инженера требуется подобрать удачную комбинацию всех технических характеристик. Однако в процессе проектирования остаётся одна трудно формализуемая характеристика, которую сложно рассчитать и предсказать, – это производительность компьютера. Как определить, какой процессор необходимо выбрать? Сколько оперативной памяти требуется? А ведь в 99% случаев на стадии проектирования структуры аппаратных средств и составления спецификаций у нас нет информации о том, насколько ресурсоёмким будет используемое программное обеспечение. Будет ли, к примеру, без заметных задержек работать система видеораспознавания?

Определение узких мест

В этом вопросе всегда приходится балансировать, выбирая между производительной платформой и ценой. Не всегда бюджет проекта позволяет заказать ЦП intel core i7 или i9 последних поколений и 16 ГБ оперативной памяти и не беспокоиться о том, что для установленного ПО может оказаться недостаточно ресурсов. Возникает вопрос, как найти оптимальное соотношение производительности и цены.

Давайте разбираться. Как правило, относительную оценку производительности компьютера выполняют по характеристикам оперативной памяти, центрального и графического процессоров (ЦП и ГП).

При оценке требуемого размера оперативной памяти обычно ограничиваются ручным тестированием ПО, используя диспетчер задач, чтобы определить среднее необходимое количество оперативной памяти при работе наиболее ресурсоёмких задач.

При подборе ЦП необходимо обратить внимание на две основные характеристики, которые обусловливают его относительную производительность:

1) частота, определяющая количество операций в секунду на одном ядре;
2) количество физических ядер в пределах одного процессора.

Данные характеристики не учитывают различий в архитектуре (AMD против Intel, Haswel против Kaby Lake и т.д.), тем не менее при сравнении двух процессоров они являются основными параметрами, которые определяют относительную производительность процессора. Если программное обеспечение, которое планируется использовать в системе, задействует только одно ядро процессора, то частота процессора является хорошим и достаточно точным показателем производительности.

Однако если программное обеспечение может использовать несколько ядер, то произвести оценку производительности процессора становится затруднительно, поскольку практически любое приложение не будет максимально эффективно задействовать все ядра. Возникает вопрос оценки эффективности распараллеливания вычислений используемым ПО.

Оценка производительности процессора для определённого приложения

Существует несколько различных способов расчёта эффективности распараллеливания вычислений. В данной статье воспользуемся законом Амдала [1–3], который иллюстрирует ограничение роста производительности системы с увеличением количества вычислителей.

Возвращаясь к нашей задаче, можно сказать, что из данного закона следует, что при увеличении количества ядер процессора будет наблюдаться замедление прироста производительности при низкой эффективности использования всех ядер одновременно. Если вы знаете эффективность распараллеливания, можно математически рассчитать точку, в которой производительность при использовании меньшего числа ядер, работающих на более высокой частоте, выше, чем

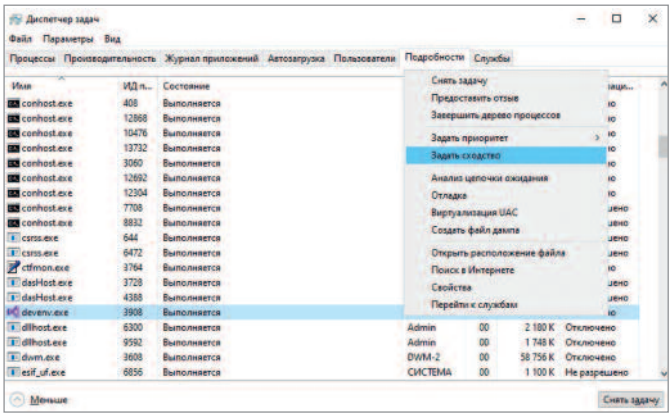

Рис. 1. Пример задания количества используемых программой ядер процессора

при использовании большего числа ядер, работающих на более низкой частоте. В упрощённой форме закон Амдала выглядит следующим образом:

$$S(n) = \frac{1}{1 - p + \frac{p}{n}}. \quad (1)$$

$S(n)$ – это теоретическая степень ускорения программных вычислений при использовании n ядер (потоков) процессора. Здесь p представляет собой часть алгоритма, которая может быть распределена среди нескольких ядер, то есть эффективность распараллеливания. Исходя из этого, $1–p$ – эта та часть программы, которая может выполняться только последовательно.

Самым простым и быстрым способом определения эффективности распараллеливания является проведение ряда тестовых измерений. Необходимо оценить время выполнения какой-либо цикличной операции ПО на разном количестве ядер. Не стоит пугаться. Для тестирования достаточно использовать программные средства, чтобы изменять количество задействованных ядер. В Windows для этого следует задать сходство для необходимого вам процесса через диспетчер задач (в ОС семейства Linux можно использовать команду taskset).

Обратите внимание, что, если процессор поддерживает Hyper-threading (гиперпоточность), на самом деле в списке будет в два раза больше ядер, чем у него имеется физически. Можно либо отключить Hyper-threading в BIOS перед выполнением тестирования, либо просто выбрать два потока для каждого ядра ЦП, которое вы хотите протестировать. Потоки с Hyper-threading всегда отображаются сразу после физического ядра в Windows, поэтому нужно выбрать два ядра для каждого физического ядра ЦП, которое вы хотите использовать в программе. Другими словами, выбор ядер 1 и 2 позволит программе использовать только одно физическое ядро ЦП, выбор ядер 1–4 даст возможность программе использовать два физических ядра ЦП и т.д. Перед измерениями рекомендуется отключать функцию TurboBoost в BIOS. Это позволит повысить точность результатов. На рис. 1 приведён пример задания сходства для процессов через диспетчер задач Windows.

Теперь, имея возможность ограничивать количество ядер, которое используется вашим ПО, можно рассчитать, за какой промежуток времени будет выполнен алгоритм. Например, имеется программа по распознаванию отсканированных документов. Вам необходимо зафиксировать, за какое время будет проведено распознавание и парсинг (Parsing – анализ) всех его полей при различном количестве задействованных

Таблица 1

Результаты проведения экспериментов

Количество задействованных ядер	Время выполнения	Рассчитанный прирост производительности	Прирост по закону Амдала (p = 70%)
1	5,2 с	1	1
2	3,1 с	1,67	1,53
3	2,7 с	1,92	1,87
4	2,5 с	2,08	2,1

ядер. В табл. 1 в первом и втором столбцах приведены примеры записи результатов измерений.

Следует отметить, что задание сходства устанавливается для конкретного экземпляра программы. Когда вы запустите программу в следующий раз, то придётся заново выполнить привязку. Однако если вы хотите быстро протестировать одно действие, используя различное количество ядер ЦП, вам не нужно закрывать программу перед изменением количества задействованных ядер – просто нажмите «Задать сходство» и измените его прямо во время выполнения программы. Тем не менее вы получите более точные результаты, закрыв программу между запусками тестируемого алгоритма, поскольку это очистит область памяти ОЗУ, выделенную для программы.

На следующем шаге у нас появятся данные для определения эффективности распараллеливания p. Для начала мы можем оценить, насколько быстрее программа выполнила операции с n ядрами процессора относительно времени выполнения программы на одном ядре. Например, для четырёх ядер время выполнения составило 2,5 с, в то время как для одного ядра – 5,2 с, тогда получим рассчитанный прирост производительности 5,2/2,5 = 2,08.

Для точного расчёта параметра p по фактическим значениям ускорения можно использовать метод наименьших квадратов для нелинейной функции, однако это достаточно трудоёмкий и сложный процесс. Для нас достаточно произвести менее точную оценку. Да и время, потраченное высококвалифицированным специалистом на точный расчёт, возможно, будет стоит дороже, чем экономия бюджета при выборе оптимального решения.

Существует простой способ оценить с достаточной точностью эффективность распараллеливания вычислений. Предположим, что эффективность распараллеливания составляет 50%. Для этого значения нужно рассчитать по формуле (1) прирост производительности и построить график в любом удобном пакете, например Excel (рис. 2).

Из рис. 2 можно видеть, что фактическое ускорение при увеличении количества ядер выше, чем рассчитанное по за-

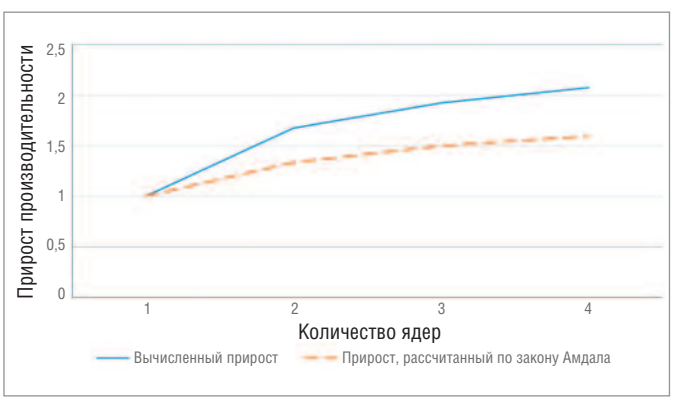

Рис. 2. График расчётного фактического ускорения и ускорения по закону Амдала при p = 50%

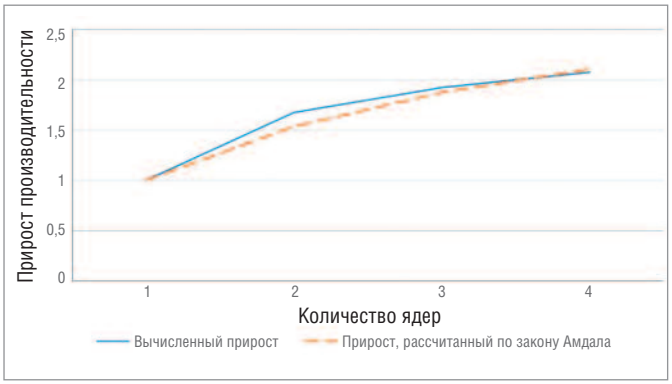

Рис. 3. График расчётного фактического ускорения и ускорения по закону Амдала при $p = 70\%$

кону Амдала $p = 50\%$, следовательно, эффективность распараллеливания должна быть выше. Экспериментально подбираем $p = 70\%$ (рис. 3).

Получив рассчитанную эффективность распараллеливания программного обеспечения на тестируемом процессоре, можно оценить производительность этого процессора и подобрать потенциальных кандидатов для проработки системы.

Сначала необходимо рассчитать эффективное количество используемых ядер, то есть такое число ядер ЦП, на котором программа выполняется с оптимальной производительностью.

Для этого можно опять воспользоваться формулой (1), но при расчётах использовать не суммарное количество физических и виртуальных ядер, а только количество физических ядер.

$$efc = \frac{1}{1 - p + \frac{p}{phc}}. \quad (2)$$

Здесь *efc* – эффективное количество ядер, а *phc* – физическое количество ядер процессора. Исходя из этого, мы можем умножить количество эффективных ядер на рабочую частоту каждого ЦП *f*, чтобы узнать, сколько операций с плавающей запятой в секунду ЦП может выполнить. Такую величину обычно называют флопс (FLOPS – FLoating-point Operations Per Second).

$$FLOPS = f \times efc. \quad (3)$$

Флопс является внесистемной единицей оценки производительности вычислительных систем. В данном случае неважно, сколько операций с плавающей запятой за один такт выполняет конкретный процессор: во-первых, при анализе спецификаций на ЦП эта величина отсутствует; во-вторых, исходя из предпосылок применения данного метода, предполагается сравнение процессоров одного семейства или архитектуры. Теперь мы можем оценить, какое время T_{perf} потребуется для выполнения на потенциальном процессоре такой же операции, которую мы тестировали на предыдущем этапе.

$$T_{perf} = \frac{FLOPS_{cpu1}}{FLOPS_{cpu2}} \times \min(RT), \quad (4)$$

где $\min(RT)$ – минимальное время выполнения тестируемой операции, это значение мы получили во втором столбце табл. 1; *cpu*1 – тестируемый ЦП; *cpu*2 – сравниваемый ЦП.

Давайте проведём экспериментальную оценку. Результаты в табл. 1 были протестированы на процессоре intel core i5 7200. Предположим, что нас не устраивает время сканирования документа в 2,5 с. Рассмотрим, как повлияет на производительность, если мы заменим процессор на более старшую модель intel core i7 7500. Для этого проведём расчёты по формулам (2) и (3) для каждого процессора и оценим, насколько увеличится время выполнения операции. У этих процессоров количество эффективных ядер совпадает, различия только в частоте. Тогда получается:

$$T_{perf} = \frac{3,1}{3,5} \times 2,5 = 2,21. \quad (5)$$

Таким образом, при переходе на core i7 той же архитектуры показатель времени на сканирование документа сократится примерно на 0,3 с. Дальнейшие выводы о том, является ли данная оптимизация приемлемой и стоит ли она переплаты за чуть более мощный процессор, должен сделать архитектор системы.

Выводы

Необходимо сделать важные замечания. Данный способ оценки предполагает, что сравниваемые процессоры имеют схожую архитектуру. Если вы заинтересованы в ЦП, который использует совершенно другую архитектуру, всё равно можно применять этот метод для определения относительной разницы в производительности между несколькими различными моделями ЦП одного семейства, но, скорее всего, он не будет точно отражать фактическую производительность. Также существуют другие ограничения использования данного метода. Например, следует помнить, что закон Амдала применим только в том случае, если узким местом системы является процессор. Если задача не ограничивается ресурсами процессора, вы обнаружите, что дальнейшее увеличение количества ядер перестанет влиять на прирост производительности. Если производительность видеокарты, ОЗУ или жёсткого диска не позволяет программе работать быстрее, добавление большего количества ядер ЦП никогда не поможет, даже если программа работает на 100% параллельно.

Описанным в статье способом команда проектного офиса пользуется при проведении первичной оценки требуемого оборудования. Полученная оценка, конечно же, не даёт точных результатов, но позволяет формализованными методами рассчитать и обосновать спецификацию предлагаемой системы. Мы планируем частично автоматизировать данный процесс. Ведь можно собирать статистику по используемым конкретным приложением ресурсам системы, делать простейшие расчёты на основе описанного метода (данные о технических характеристиках различных платформ можно выгрузить из открытых баз) и на основе расчётов реализовать систему принятия решений, которая будет рекомендовать использование определённого процессора для достижения требуемой производительности. ●

Литература

1. Gustafson J.L. Amdahl's Law // Encyclopedia of Parallel Computing. – Boston : Springer, 2011.
2. Batch M. Estimating CPU performance Using Amdahls Law. – USA : PugetSystems, 2015.
3. Popov G., Mastorakis N., Mladenov V. Calculation of the Acceleration of Parallel Programs as Function of the Number of Threads // Latest Trends on Computers. – 2014. – Vol. 2.

Автор – сотрудник фирмы ПРОСОФТ
Телефон: (495) 234-0636
E-mail: info@prosoft.ru