алгоритмов. Например, на ней легко демонстрируются параллельные формы алгоритмов, показывается зависимость ширины ярусов, числа задействованных процессоров и получаемого ускорения. Если бы концепция неограниченного параллелизма не нужна была ни для чего другого, ее все равно следовало бы создать. Просто из-за того, что параллельные формы являются одним из важнейших инструментов изучения структуры алгоритмов.

ЛЕКЦИЯ 4

Характеристики вычислительных процессов

Содержание: *простое и конвейерное функциональное устройство, загруженность, производительность, ускорение, система устройств, влияние связей между устройствами, законы Амдала и следствия.*

Для того чтобы вычислительная система имела высокую производительность, она должна состоять из большого числа одновременно работающих функциональных устройств (ФУ). Для оценки качества их работы предложено много различных характеристик. Нередко одни и те же по смыслу характеристики вводятся по-разному. И не всегда легко понять, отражают ли эти различия существо дело или же они связаны с какими-то другими причинами. Показательной в этом отношении является такая характеристика как производительность системы. Различных ее определений введено столь много и они столь сильно отличаются друг от друга, что объявляемая производительность вычислительной системы воспринимается лишь как элемент рекламы.

Тем не менее, характеристики процессов работы системы в целом и отдельных ее элементов очень важны, поскольку позволяют находить и устранять узкие места. Общий недостаток всех вводимых ранее характеристик – это отсутствие четкого обоснования. Именно их обоснованию и будет посвящена данная лекция. Будем рассматривать характеристики в рамках некоторой модели процесса работы устройств, вполне достаточной для практического применения. Мы не будем интересоваться содержанием операций, выполняемых функциональными устройствами. Они могут означать арифметические или логические функции, ввод-вывод данных, пересылку данных в память и извлечение данных из нее или что-либо иное. Более того, допускается, что при разных обращениях операции могут означать разные функции. Для нас сейчас важны лишь времена выполнения операций и то, на устройствах какого типа они реализуются.

Пусть введена система отсчета времени и установлена его единица, например, секунда. Будем считать, что длительность выполнения операций измеряется в долях единицы. Все устройства, реальные или гипотетические,

основные или вспомогательные, могут иметь любые времена срабатывания. Единственное существенное ограничение состоит в том, что все срабатывания *одного и того* же ФУ должны быть *одинаковыми* по длительности. Всегда мы будем интересоваться работой каких-то конкретных наборов ФУ. По умолчанию будем предполагать, что все другие ФУ, необходимые для обеспечения процесса функционирования этих наборов, срабатывают *мгновенно*. Поэтому, если не сделаны специальные оговорки, мы не будем в таких случаях принимать во внимание факт их реального существования. Времена срабатываний изучаемых ФУ будем считать *положительными*.

Назовем функциональное устройство *простым,* если никакая последующая операция не может начать выполняться раньше, чем закончится предыдущая. Простое ФУ может выполнять операции одного типа или разные операции. Разные ФУ могут выполнять операции, в том числе одинаковые, за разное время. Примером простого ФУ могут служить обычные сумматоры или умножители. Эти ФУ реализуют только один тип операции. Простым устройством можно считать многофункциональный процессор, если он не способен выполнять различные операции одновременно, и мы не принимаем во внимание различия во временах реализации операций, предполагая, что они одинаковы. Основная черта простого ФУ только одна: оно *монопольно* использует свое оборудование для выполнения каждой отдельной операции.

В отличие от простого ФУ *конвейерное* ФУ распределяет свое оборудование для одновременной реализации нескольких операций. Очень часто оно конструируется как линейная цепочка простых элементарных ФУ, имеющих одинаковые времена срабатывания. На этих элементарных ФУ последовательно реализуются отдельные этапы операций. В случае конвейерного ФУ, выполняющего операцию сложения чисел с плавающей запятой, соответствующие элементарные устройства последовательно реализуют такие операции как сравнение порядков, сдвиг мантиссы, сложение мантисс и т.п. Ничто не мешает считать конвейерным ФУ линейную цепочку универсальных процессоров и рассматривать каждый из процессоров как элементарное ФУ. Принцип конвейерности остается одним и тем же. Сначала на первом элементарном ФУ выполняется первый этап первой операции, и результат передается второму элементарному ФУ. Затем на втором элементарном ФУ реализуется второй этап первой операции. Одновременно на освободившемся первом ФУ реализуется первый этап второй операции. После этого результат срабатывания второго ФУ передается третьему ФУ, результат срабатывания первого ФУ передается второму ФУ. Освободившееся первое ФУ готово для выполнения первого этапа третьей операции и т. д. После прохождения всех элементарных ФУ в конвейере операция оказывается выполненной. Элементарные ФУ называются *ступенями* конвейера, число ступеней в конвейере – *длиной* конвейера. Простое ФУ всегда можно считать конвейерным с длиной конвейера, равной 1. Как уже говорилось, конвейерное

ФУ часто является линейной цепочкой простых ФУ, но возможны и более сложные конвейерные конструкции.

Поскольку конвейерное ФУ уже само является системой связанных устройств, необходимо установить наиболее общие правила работы систем ФУ. Будем считать, что любое ФУ не может *одновременно* выполнять операцию и сохранять результаты предыдущих срабатываний, т.е. оно *не имеет собственной памяти*. Однако будем допускать, что результат последнего срабатывания может сохраняться в ФУ до момента начала очередного его срабатывания, *включая сам этот момент*. После начала очередного срабатывания ФУ результат предыдущего срабатывания *пропадает*. Предположим, что все ФУ работают по индивидуальным командам. В момент подачи команды конкретному ФУ на его входы передаются аргументы выполняемой операции либо как результаты срабатывания других ФУ с их выходов, либо как входные данные, либо как-нибудь иначе. Сейчас нам безразлично, как именно осуществляется их подача. Важно то, что в момент начала очередного срабатывания ФУ входные данные для него доступны, а сам процесс подачи не приводит к задержке общего процесса. Любое функциональное устройство может начинать выполнение операции в любой момент времени, начиная с момента готовности ее аргументов. Конечно, мы предполагаем, что программы, определяющие работу всех ФУ, составлены корректно и не приводят к тупиковым или неправильным ситуациям.

При определении различных характеристик, связанных с работой ФУ, приходится иметь дело с числом операций, выполняемых за какое-то время. Это число должно быть целым. Если отрезок времени равен T, а длительность операции есть τ, то за время T можно выполнить порядка T/τ операций. Чтобы не загромождать выкладки и формулы символами целочисленности и различными членами малого порядка, мы будем всюду приводить результаты в главном, что эквивалентно переходу к пределу при $T \to \infty$. Говоря иначе, все характеристики и соотношения будут носить *асимптотический* характер. Данное обстоятельство нисколько не снижает практическую ценность получаемых результатов, но делает их более наглядными.

Назовем *стоимостью операции* время ее реализации, а *стоимостью работы* – сумму стоимостей всех выполненных операций. Стоимость работы – это время последовательной реализации всех рассматриваемых операций на простых ФУ с аналогичными временами срабатываний. *Загруженностью устройства* на данном отрезке времени будем называть отношение стоимости реально выполненной работы к максимально возможной стоимости. Ясно, что загруженность p всегда удовлетворяет условиям $0 \le p \le 1$. Также очевидно, что максимальная стоимость работы, которую можно выполнить за время T, равна T для простого ФУ и nT для конвейерного ФУ длины n.

Будем называть *реальной производительностью* системы устройств количество операций, реально выполненных в среднем за единицу времени. *Пиковой производительностью* будем называть максимальное количество операций, которое может быть выполнено той же системой за единицу времени *при отсутствии связей* между ФУ. Из определений вытекает, что как реальная, так и пиковая производительность системы равна сумме соответственно реальных или пиковых производительностей всех составляющих систему устройств. Пусть система состоит из s устройств, в общем случае простых или конвейерных. Если устройства имеют пиковые производительности $\pi_1, \ldots, \pi_s$ и работают с загруженностями $p_1, \ldots, p_s$, то реальная производительность r системы выражается формулой.

$$r = \sum_{i=1}^{s} p_i \pi_i .$$

Поскольку реальная производительность системы равна сумме реальных производительностей всех ФУ, то достаточно рассмотреть одно устройство. Пусть для выполнения одной операции на ФУ требуется время τ и за время T выполнено N операций. Независимо от того, каков тип устройства, стоимость выполненной работы равна $N\tau$. Если устройство простое, то максимальная стоимость работы равна T. Поэтому загруженность устройства равна $N\tau/T$. По определению реальная производительность ФУ есть N/T, а его пиковая производительность равна $1/\tau$. Поэтому соотношение выполняется. Предположим теперь, что устройство конвейерное длины n. Максимальная стоимость работы в данном случае равна nT. Поэтому загруженность устройства равна $N\tau/nT$. Реальная производительность снова есть N/T, а пиковая производительность будет равна n/τ. И соотношение снова выполняется.

Если r, π, p суть соответственно реальная производительность, пиковая производительность и загруженность одного устройства, то имеет место равенство $r = p\pi$. Отсюда видно, что для достижения наибольшей реальной производительности устройства нужно обеспечить наибольшую его загруженность. Для практических целей понятие производительности наиболее важно, потому что именно оно показывает, насколько эффективно устройство выполняет полезную работу. По отношению к производительности понятие загруженности является вспомогательным. Тем не менее, оно полезно в силу того, что указывает путь повышения производительности, причем через вполне определенные действия.

Хотелось бы и для системы устройств ввести понятие загруженности, играющее аналогичную роль. Его можно определять по-разному. Например, как и для одного ФУ, можно было бы считать, что загруженность системы ФУ есть отношение стоимости реально выполненной работы к максимально возможной стоимости. Такое определение вполне приемлемо и позволяет сделать ряд полезных выводов. Однако имеются и возражения. В этом

определении медленные и быстрые устройства оказываются равноправными и если необходимо повысить загруженность системы, то не сразу видно, за счет какого ФУ это лучше сделать. К тому же, в данном случае не всегда будет иметь место равенство $r = p\pi$ с соответствующими характеристиками системы.

Правильный путь введения понятия загруженности системы подсказывает полученное соотношение. Пусть система состоит из s устройств, в общем случае простых или конвейерных. Если устройства имеют пиковые производительности $\pi_1, \ldots, \pi_s$ и работают с загруженностями $p_1, \ldots, p_s$, то будем считать по определению, что *загруженность системы* есть величина

$$p = \sum_{i=1}^{s} \alpha_i p_i, \quad \alpha_i = \frac{\pi_i}{\sum_{j=1}^{s} \pi_j}.$$

Отсюда следует, что загруженность системы есть взвешенная сумма загруженностей отдельных устройств, так как

$$\sum_{i=1}^{s} \alpha_i = 1, \quad \alpha \geq 0, \quad 1 \leq i \leq s.$$

Поэтому, как и должно быть для загруженности, выполняются неравенства $0 \leq p \leq 1$. Из полученных соотношений заключаем, что для того, чтобы загруженность системы устройств равнялась 1, необходимо и достаточно, чтобы равнялись 1 загруженности каждого из устройств. Это вполне логично. Если система состоит из одного устройства, т.е. $s = 1$, то на ней понятия загруженности системы и загруженности устройства совпадают. Данный факт говорит о согласованности только что введенного понятия загруженности системы с ранее введенными понятиями. По определению, пиковая производительность π системы устройств равна $\pi_1 + \ldots + \pi_s$. Следовательно, всегда выполняется очень важное равенство $r = p\pi$.

Большое число ФУ, также как и конвейерные ФУ, используются тогда, когда возникает потребность решить задачу быстрее. Чтобы понять, насколько быстрее это удается сделать, нужно ввести понятие "ускорение". Как и в случае загруженности, оно может вводиться различными способами, многообразие которых зависит от того, что с чем и как сравнивается. Нередко ускорение определяется, например, как отношение времени решения задачи на одном универсальном процессоре к времени решения той же задачи на системе из s таких же процессоров. Очевидно, что в наилучшей ситуации ускорение может достигать s. Отношение ускорения к s называется *эффективностью*. Заметим, что подобное определение ускорения применимо только к системам, состоящим из одинаковых устройств, и не

распространяется на смешанные системы. Понятие же эффективности в рассматриваемом случае просто совпадает с понятием загруженности.

При введении понятия ускорения мы поступим иначе. Пусть алгоритм реализуется за время T на вычислительной системе из s устройств, в общем случае простых или конвейерных и имеющих пиковые производительности $\pi_1, \ldots, \pi_s$. Предположим, что $\pi_1 \le \pi_2 \le \ldots \le \pi_s$. При реализации алгоритма система достигает реальной производительности r. Будем сравнивать скорость работы системы со скоростью работы гипотетического простого универсального устройства, имеющего такую же пиковую производительность π_s, как самое быстрое ФУ системы, и обладающего возможностью выполнять те же операции, что все ФУ системы.

Итак, будем называть отношение $R = r/\pi_s$ ускорением реализации алгоритма на данной вычислительной системе или просто *ускорением*. Выбор в качестве гипотетического простого, а не какого-нибудь другого, например, конвейерного ФУ объясняется тем, что одно простое универсальное ФУ может быть полностью загружено на любом алгоритме. Теперь имеем

$$R = \frac{\sum_{i=1}^{s} p_i \pi_i}{\max\limits_{1 \le i \le s} \pi_i}.$$

Анализ определяющего ускорение выражения показывает, что ускорение, которое обеспечивает система, состоящая из s устройств, никогда не превосходит s и может достигать s в том и только в том случае, когда все устройства системы имеют одинаковые пиковые производительности и полностью загружены.

Подводя итог проведенным исследованиям, приведем для одного частного случая полезные выводы. Если система состоит из s простых или конвейерных устройств одинаковой пиковой производительности, то

– загруженность системы равна среднему арифметическому загруженностей всех устройств;

– реальная производительность системы равна сумме реальных производительностей всех устройств;

– пиковая производительность системы в s раз больше пиковой производительности одного устройства;

– обеспечиваемое системой ускорение равно сумме загруженностей всех устройств;

– если система состоит только из простых устройств, то ускорение равно отношению времени реализации алгоритма на одном универсальном простом устройстве с той же пиковой производительностью к времени реализации алгоритма на системе.

Пусть система устройств функционирует и показывает какую-то реальную производительность. Если производительность недостаточна, то для ее повышения необходимо увеличить загруженность системы. Для этого, в свою очередь, нужно повысить загруженность любого устройства, у которого она еще не равна 1. Но остается открытым вопрос, всегда ли это можно сделать. Если устройство загружено не полностью, то его загруженность заведомо можно повысить в том случае, когда данное устройство не связано с другими. В случае же связанности устройств ситуация не очевидна.

Снова рассмотрим систему из s устройств. Не ограничивая общности, будем считать все устройства простыми, так как любое конвейерное ФУ всегда можно представить как линейную цепочку простых устройств. Допустим, что между устройствами установлены направленные связи, и они не меняются в процессе функционирования системы. Построим ориентированный граф, в котором вершины символизируют устройства, а дуги – связи между ними. Дугу из одной вершины будем проводить в другую в том и только том случае, когда результат каждого срабатывания устройства, соответствующего первой вершине, обязательно передается в качестве аргумента для очередного срабатывания устройству, соответствующему второй вершине. Назовем этот граф *графом системы*. Не ограничивая существенно общности, можно считать его связным. Предположим, что каким-то образом в систему введены необходимые для начала работы данные. После запуска системы все ее функциональные устройства начинают работать под собственным управлением, соблюдая описанные выше правила.

Исследуем максимальную производительность системы, т.е. ее максимально возможную реальную производительность при достаточно большом времени функционирования. Пусть система состоит из s простых устройств с пиковыми производительностями $\pi_1, \ldots, \pi_s$. Если граф системы связный, то максимальная производительность $r_{\max}$ системы выражается формулой

$$r_{\max} = s \cdot \min_{1 \le i \le s} \pi_i$$

В самом деле, предположим, что дуга графа системы идет из i-го ФУ в j-ое ФУ. Пусть за достаточно большое время i-ое ФУ выполнило N_i операций, j-ое ФУ – N_j операций. Каждый результат i-го ФУ обязательно является одним из аргументов очередного срабатывания j-го ФУ. Поэтому количество операций, реализованных j-ым ФУ за время Т не может более, чем на 1, отличаться от количества операций, реализованных i-ым ФУ, т.е. $N_i - 1 \le N_j \le N_i + 1$. Так как граф системы связный, то любые две его вершины могут быть связаны цепью, составленной из дуг. Допустим, что граф системы содержит q дуг. Если k-ое ФУ за время T выполнило N_k операций , а l-ое ФУ – N_l операций, то из последних неравенств вытекает, что $N_l - q \le N_k \le N_l + q$ для любых $k, l, 1 \le k, l \le s$. Пусть устройства перенумерованы так, что $\pi_1 \le \pi_2 \le \ldots \le \pi_s$. Принимая во внимание эту упорядоченность и полученные для числа выполняемых операций соотношения, находим, что

$$r = \sum_{i=1}^{s} \left(\frac{N_i}{\pi_i T} \right) \pi_i \le \frac{N_1 s}{T} + \frac{q(s-1)}{T},$$

$$r \ge \frac{N_1 s}{T} - \frac{q(s-1)}{T}.$$

Вторые слагаемые в этих неравенствах стремятся к нулю при T, стремящемся к бесконечности. Для всех k, $1 \le k \le s$, обязаны выполняться неравенства $N_k \le \pi_k T$. В силу предполагаемой упорядоченности производительностей π_k и того, что все N_k асимптотически равны между собой, число операций, реализуемых каждым ФУ, будет асимптотически максимальным, если выполняется равенство $N_1 = \pi_1 T$. Это означает, что максимально возможная реальная производительность системы асимптотически будет равна $s\pi_1$, что и требовалось доказать.

Отметим несколько следствий. Пусть вычислительная система состоит из s простых устройств с пиковыми производительностями $\pi_1,\ldots,\pi_s$. Если граф системы связный, то

– асимптотически каждое из устройств выполняет одно и то же число операций;

– загруженность любого устройства не превосходит загруженности самого непроизводительного устройства;

– если загруженность какого-то устройства равна 1, то это – самое непроизводительное устройство;

– загруженность системы не превосходит

$$p_{\max} = \frac{s \min\limits_{1 \le i \le s} \pi_i}{\sum\limits_{i=1}^{s} \pi_i};$$

– ускорение системы не превосходит

$$R_{\max} = \frac{s \min\limits_{1 \le i \le s} \pi_i}{\max\limits_{1 \le i \le s} \pi_i}.$$

Заметим, что равенство $r = p\pi$ определяет многие узкие места процесса функционирования системы. Некоторые описанные в литературе узкие места связываются с именем Амдала, американского специалиста в области вычислительной техники. Объявленные им законы легко выводятся из полученных выше соотношений, поэтому мы не будем останавливаться на их доказательствах. С деталями можно познакомиться в [1]. Чтобы не терять узнаваемость различных фактов, мы будем оставлять именными

соответствующие утверждения, даже если они совсем простые. Возможно, лишь несколько изменим формулировки, приспособив их к текущему изложению материала.

1-ый закон Амдала. Производительность вычислительной системы, состоящей из связанных между собой устройств, в общем случае определяется самым непроизводительным ее устройством.

Следствие. Пусть система состоит из простых устройств и граф системы связный. Асимптотическая производительность системы будет максимальной, если все устройства имеют одинаковые пиковые производительности.

Когда мы говорим о максимально возможной реальной производительности, то подразумеваем, что функционирование системы обеспечивается таким расписанием подачи команд, которое минимизирует простой устройств. Максимальная производительность может достигаться при разных режимах. В частности, она достигается при синхронном режиме с тактом, обратно пропорциональным производительности самого медленного из ФУ, если конечно, система состоит из простых устройств и граф системы связный. Пусть система состоит из s простых устройств одинаковой производительности. Тогда как в случае связанной системы, так и в случае не связанной, максимально возможная реальная производительность при больших временах функционирования оказывается одной и той же и равной s-кратной пиковой производительности одного устройства.

Мы уже неоднократно убеждались в том, что различные характеристики процесса функционирования системы становятся лучше, если система состоит из устройств одинаковой производительности. Предположим, что все устройства, к тому же, простые и универсальные, т. е. на них можно выполнять различные операции. Пусть на такой системе реализуется некоторый алгоритм, а сама реализация соответствует какой-то его параллельной форме. Допустим, что высота параллельной формы равна m, ширина равна q и всего в алгоритме выполняется N операций. В сформулированных условиях максимально возможное ускорение системы равно N/m.

Пусть система состоит из s устройств пиковой производительности π. Предположим, что за время T реализации алгоритма на i-ом ФУ выполняется N_i операций. По определению загруженность i-го ФУ равна $N_i/\pi\, T$. Ускорение системы в данном случае равно

$$R = \frac{\sum_{i=1}^{s} \left(\frac{N_i}{\pi T} \right) \pi}{\pi} = \frac{N}{\pi T}.$$

При заданной производительности устройств время реализации одного яруса параллельной формы равно π^{-1}. Поэтому время T реализации алгоритма не меньше, чем m/π, и достигает этой величины, когда все ярусы реализуются

подряд без пропусков. Следовательно, ускорение системы при любом числе устройств не будет превосходить *N/m*. Это означает, что минимальное число устройств системы, при котором может быть достигнуто максимально возможное ускорение, равно ширине алгоритма.

Предположим, что по каким-либо причинам *n* операций из *N* мы вынуждены выполнять последовательно. Причины могут быть разными. Например, операции могут быть последовательно связаны информационно. И тогда без изменения алгоритма их нельзя реализовать иначе. Но вполне возможно, что мы просто не распознали параллелизм, имеющийся в той части алгоритма, которая описывается этими операциями. Отношение $\beta = n/N$ назовем *долей последовательных вычислений*.

2-й закон Амдала. Пусть система состоит из *s* одинаковых простых универсальных устройств. Предположим, что при выполнении параллельной части алгоритма все *s* устройств загружены полностью. Тогда максимально возможное ускорение равно

$$R = \frac{s}{\beta\, s + (1-\beta)}.$$

3-й закон Амдала. Пусть система состоит из простых одинаковых универсальных устройств. При любом режиме работы ее ускорение не может превзойти обратной величины доли последовательных вычислений.

В проведенных исследованиях нигде не конкретизировалось содержание операций. В общем случае они могут быть как элементарными типа сложения или умножения, так и очень крупными, представляющими алгоритмы решения достаточно сложных задач. Современные вычислительные системы состоят из десятков и даже сотен тысяч процессоров. Они вполне укладываются в рассмотренные модели. Системы с большим числом процессоров должны быть загружены достаточно полно. Проведенные исследования говорят о том, что в реализуемых на таких системах алгоритмах доля последовательных вычислений должна быть порядка десятых и сотых долей процента. Этот факт говорит о больших проблемах, которые должны сопровождать конструирование подобных алгоритмов.

В заключение еще раз подчеркнем следующее. В обширной литературе, посвященной параллельным процессам и параллельным вычислительным системам можно встретить много различных определений и законов, касающихся производительности, ускорения, эффективности и т. п. Как правило, новые определения и законы возникают тогда, когда старые в чем-то не устраивают исследователей. Однако ко всем таким "новациям" следует относиться очень осторожно. Довольно часто в попытке что-то "улучшить" скрываются какие-то узкие места, одни понятия подменяются другими, иногда просто проводятся ошибочные рассуждения, как, например, при сравнении достигаемых ускорений, исходя из законов Амдала и Густавсона-Барсиса [1].

Уже отмечалось выше, что особенно много различных определений дается для производительности системы. Приведем один курьезный пример. Предположим, что система имеет два простых устройства одинаковой пиковой производительности. Пусть одно устройство есть сумматор, другое – умножитель. Допустим, что все обмены информацией осуществляются мгновенно и решается задача вычисления матрицы $A = B + C$ при заданных матрицах B,C. Очевидно, что при естественном выполнении операции сложения матриц реальная производительность будет равна половине пиковой, так как умножитель не используется. Спрашивается: "Можно ли каким-то образом *на данной задаче* повысить реальную производительность?" Ответ: "Можно". Запишем равенство $A = B + C$ в виде $A = B + 1 \cdot C$. Умножение элементов матрицы C на 1 позволяет загрузить умножитель. Формально реальная производительность увеличивается вдвое и сравнивается с пиковой.

Вас интересует такое увеличение производительности?

ЛЕКЦИЯ 5

Математически эквивалентные преобразования

Содержание: *математически эквивалентные преобразования, алгебраические законы на практике не выполняются, эквивалентные преобразования и устойчивость, эквивалентные преобразования и число операций, эквивалентные преобразования и параллелизм вычислений, принцип сдваивания, снова граф алгоритма, граф алгоритма и ошибки округления, оценка параллелизма алгоритма снизу.*

Общее математическое образование в вузах базируется на постулатах, широкое использование которых начинается еще в средней школе. Это, в первую очередь, предположения о выполнении законов ассоциативности, коммутативности и дистрибутивности при реализации операций над числами. Данные законы позволяют построить аппарат *математически эквивалентных преобразований* символьно-числовых выражений на основе расстановки и раскрытия скобок, приведения и создания подобных членов, перестановки символов и операций и т.п. Аппарат настолько эффективный, что без него не обходится изложение курсов ни по общей, ни по вычислительной математике. Само по себе его применение не вызывает никаких возражений, пока речь идет о проведении преобразований, не связанных с практическим счетом. Но как только дело касается реальных вычислений, формальное применение аппарата математически эквивалентных преобразований становится невозможным в принципе из-за нарушения базисных предположений.

Использование аппарата математически эквивалентных преобразований явно или неявно предполагает, что все операции над символами и числами