Проблемы выбора аппаратно-программной платформы, соответствующей потребностям прикладной области

Администрирование ИС

Лекция 11

Выбор аппаратной платформы и конфигурации системы представляет собой чрезвычайно сложную задачу.

- Это связано с *характером прикладных систем*, который в значительной степени может определять рабочую нагрузку вычислительного комплекса в целом.
- Часто оказывается просто *трудно с достаточной точностью предсказать саму нагрузку*, особенно в случае, если система должна обслуживать несколько групп разнородных по своим потребностям пользователей.

*Например, иногда даже бессмысленно говорить, что для каждых N пользователей необходимо в конфигурации сервера иметь один процессор, поскольку для некоторых прикладных систем, в частности, для систем из области механических и электронных САПР, может потребоваться 2-4 процессора для обеспечения запросов одного пользователя. С другой стороны, даже одного процессора может вполне хватить для поддержки 15-40 пользователей, работающих с прикладным пакетом Oracle*Financial.*

Другие прикладные системы могут оказаться еще менее требовательными.

Следует помнить, что даже если рабочую нагрузку удается описать с достаточной точностью, обычно скорее *можно только выяснить, какая конфигурация не справится с данной нагрузкой, чем с уверенностью сказать, что данная конфигурация системы будет обрабатывать заданную нагрузку*, если только отсутствует определенный опыт работы с приложением.

Обычно рабочая нагрузка существенно определяется "типом использования" системы.

Например, можно выделить серверы NFS (Network File System), серверы управления базами данных и системы, работающие в режиме разделения времени.

Эти категории систем перечислены в порядке увеличения их сложности.

Как правило **серверы СУБД** значительно более сложны, чем **серверы NFS**, а **серверы разделения времени**, особенно обслуживающие различные категории пользователей, являются наиболее сложными для оценки.

К счастью, существует ряд упрощающих факторов.

Во-первых, как правило *нагрузка на систему в среднем сглаживается особенно при наличии большого коллектива пользователей* (хотя почти всегда имеют место предсказуемые пики).

Например, известно, что нагрузка на систему достигает пиковых значений через 1-1.5 часа после начала рабочего дня или окончания обеденного перерыва и резко падает во время обеденного перерыва. С большой вероятностью нагрузка будет нарастать к концу месяца, квартала или года.

Во-вторых, *универсальный характер большинства наиболее сложных для оценки систем -* ***систем разделения времени****, предполагает и большое разнообразие, выполняемых на них приложений, которые в свою очередь как правило стараются загрузить различные части системы*.

Далеко не все приложения интенсивно используют процессорные ресурсы, и не все из них связаны с интенсивным вводом/выводом.

Поэтому смесь таких приложений на одной системе может обеспечить достаточно равномерную загрузку всех ресурсов.

Естественно неправильно подобранная смесь может дать совсем противоположенный эффект.

Все, кто сталкивается с **задачей выбора конфигурации системы**, должны начинать с определения ответов на **два главных вопроса**:

какой сервис должен обеспечиваться системой

и

какой уровень сервиса может обеспечить данная конфигурация.

Имея *набор целевых показателей производительности конечного пользователя* и *стоимостных ограничений*, необходимо спрогнозировать возможности определенного набора компонентов, которые включаются в конфигурацию системы.

Любой, кто попробовал это сделать, знает, что *подобная оценка сложна и связана с неточностью*.

Почему оценка конфигурации системы так сложна? Причины:

1. *Подобная оценка прогнозирует будущее:* предполагаемую комбинацию устройств, будущее использование программного обеспечения, будущих пользователей.
2. *Сами конфигурации аппаратных и программных средств сложны*, связаны с определением множества разнородных по своей сути компонентов системы, в результате чего *сложность быстро увеличивается*. Несколько лет назад существовала только одна вычислительная парадигма: мейнфрейм с терминалами. В настоящее время по выбору пользователя могут использоваться несколько вычислительных парадигм с широким разнообразием возможных конфигураций системы для каждой из них.
3. *Скорость технологических усовершенствований во всех направлениях разработки компьютерной техники* (аппаратных средствах, функциональной организации систем, операционных системах, ПО СУБД, ПО "среднего" слоя (middleware)) уже *очень высокая и постоянно растет*. Ко времени, когда какое-либо изделие широко используется и хорошо изучено, оно часто рассматривается уже как устаревшее.
4. *Доступная потребителю информация о самих системах, операционных системах, программном обеспечении инфраструктуры* (СУБД и мониторы обработки транзакций) как правило *носит очень общий характер.* Структура аппаратных средств, на базе которых работают программные системы, стала настолько сложной, что *эксперты в одной области редко являются таковыми в другой*.
5. *Информация о реальном использовании систем редко является точной.* Более того, пользователи всегда находят новые способы использования вычислительных систем как только становятся доступными новые возможности.

При стольких неопределенностях просто удивительно, что многие конфигурации систем работают достаточно хорошо.

Оценка конфигурации все еще остается некоторым видом искусства, но к ней можно подойти с научных позиций.

- Намного *проще решить, что определенная конфигурация не сможет обрабатывать определенные виды нагрузки, чем определить с уверенностью, что нагрузка может обрабатываться внутри определенных ограничений производительности*.
- Более того, *реальное использование систем показывает, что имеет место тенденция заполнения всех доступных ресурсов*. Как следствие, системы, даже имеющие некоторые избыточные ресурсы, со временем не будут воспринимать дополнительную нагрузку.

Для выполнения *анализа конфигурации, система* (под которой понимается весь комплекс компьютеров, периферийных устройств, сетей и программного обеспечения) должна *рассматриваться как ряд соединенных друг с другом компонентов*.

Например, сети состоят из клиентов, серверов и сетевой инфраструктуры.

Сетевая инфраструктура включает среду (часто нескольких типов) вместе с мостами, маршрутизаторами и системой сетевого управления, поддерживающей ее работу.

В состав клиентских систем и серверов входят центральные процессоры, иерархия памяти, шин, периферийных устройств и ПО.

Ограничения производительности некоторой конфигурации по любому направлению (например, в части организации дискового ввода/вывода) обычно *могут быть предсказаны исходя из анализа наиболее слабых компонентов.*

Поскольку современные комплексы почти всегда включают несколько работающих совместно систем, точная оценка полной конфигурации требует ее *рассмотрения как на* ***макроскопическом уровне*** *(уровне сети), так и на* ***микроскопическом уровне*** *(уровне компонент или подсистем)*.

Эта же методология может быть использована для настройки системы после ее инсталляции: настройка системы и сети выполняются как правило после предварительной оценки и анализа узких мест.

Настройка конфигурации представляет собой процесс определения наиболее слабых компонентов в системе и устранения этих узких мест.

Следует отметить, что **выбор той или иной аппаратной платформы и конфигурации** определяется и рядом **общих требований**, которые предъявляются к характеристикам современных вычислительных систем. К ним относятся:

- *отношение стоимость/производительность*
- *надежность и отказоустойчивость*
- *масштабируемость*
- *совместимость и мобильность программного обеспечения.*

Отношение стоимость/производительность.

Появление любого нового направления в вычислительной технике определяется требованиями компьютерного рынка. Поэтому у разработчиков компьютеров нет одной единственной цели.

Большая универсальная вычислительная машина (мейнфрейм) или суперкомпьютер стоят дорого. Для достижения поставленных целей *при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики*. Суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM относятся именно к этой категории компьютеров.

Другим крайним примером может служить *низкостоимостная конструкция, где производительность принесена в жертву для достижения низкой стоимости.* К этому направлению относятся персональные компьютеры различных клонов IBM PC.

Между этими двумя крайними направлениями *находятся конструкции, основанные на отношении стоимость/производительность, в которых разработчики находят баланс между стоимостными параметрами и производительностью.* Типичными примерами такого рода компьютеров являются миникомпьютеры и рабочие станции.

Для сравнения различных компьютеров между собой обычно используются ***стандартные методики измерения производительности*.**

Эти методики позволяют разработчикам и пользователям использовать полученные в результате испытаний *количественные показатели для оценки тех или иных технических решений,* и в конце концов именно *производительность и стоимость* дают пользователю рациональную основу для решения вопроса, *какой компьютер выбрать*.

Надежность и отказоустойчивость.

Важнейшей характеристикой *вычислительных* систем является **надежность**.

Повышение надежности основано на принципе предотвращения неисправностей путем

- *снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции,*
- *снижения уровня помех,*
- *облегченных режимов работы схем,*
- *обеспечение тепловых режимов их работы, а также за*
- *счет совершенствования методов сборки аппаратуры.*

Следует помнить, что *понятие надежности включает не только аппаратные средства, но и программное обеспечение.*

Главной целью повышения надежности систем является целостность хранимых в них данных.

Отказоустойчивость - это такое свойство вычислительной системы, которое обеспечивает ей, как логической машине, возможность продолжения действий, заданных программой, после возникновения неисправностей.

Введение отказоустойчивости *требует избыточного аппаратного и программного обеспечения*.

Направления, связанные с предотвращением неисправностей и с отказоустойчивостью, - основные в проблеме надежности.

Концепции параллельности и отказоустойчивости вычислительных систем естественным образом связаны между собой, поскольку в обоих случаях *требуются дополнительные функциональные компоненты*.

Поэтому, собственно, *на параллельных вычислительных системах достигается как наиболее высокая производительность, так и, во многих случаях, очень высокая надежность*.

Имеющиеся ресурсы избыточности в параллельных системах могут гибко использоваться как для повышения производительности, так и для повышения надежности.

Структура многопроцессорных и многомашинных систем приспособлена к автоматической реконфигурации и обеспечивает возможность продолжения работы системы после возникновения неисправностей.

Масштабируемость

Масштабируемость представляет собой возможность наращивания числа и мощности процессоров, объемов оперативной и внешней памяти и других ресурсов вычислительной системы.

Масштабируемость должна обеспечиваться *архитектурой* и *конструкцией компьютера*, а также соответствующими *средствами программного обеспечения*.

Добавление каждого нового процессора в действительно масштабируемой системе должно *давать прогнозируемое увеличение производительности и пропускной способности при приемлемых затратах*.

Одной из основных задач при построении масштабируемых систем является **минимизация стоимости расширения компьютера и упрощение планирования**.

В идеале добавление процессоров к системе должно приводить к линейному росту ее производительности.

Однако это не всегда так. *Потери производительности могут возникать, например, при недостаточной пропускной способности шин из-за возрастания трафика между процессорами и основной памятью, а также между памятью и устройствами ввода/вывода.*

В действительности **реальное увеличение производительности трудно оценить заранее,** поскольку оно в значительной степени зависит от динамики поведения прикладных задач.

Возможность масштабирования системы определяется не только архитектурой аппаратных средств, но **зависит от заложенных свойств программного обеспечения**.

Масштабируемость программного обеспечения *затрагивает все его уровни от простых механизмов передачи сообщений до работы с такими сложными объектами как мониторы транзакций и вся среда прикладной системы*.

В частности, *программное обеспечение должно минимизировать трафик межпроцессорного обмена*, который может препятствовать линейному росту производительности системы.

Аппаратные средства (процессоры, шины и устройства ввода/вывода) являются только частью масштабируемой архитектуры, на которой программное обеспечение может обеспечить предсказуемый рост производительности.

Важно понимать, что простой переход, например, на более мощный процессор может привести к перегрузке других компонентов системы.

Это означает, что *действительно масштабируемая система должна быть сбалансирована по всем параметрам.*

Совместимость и мобильность программного обеспечения.

Концепция программной совместимости впервые в широких масштабах была применена разработчиками системы IBM/360.

Основная задача при проектировании всего ряда моделей этой системы заключалась *в создании такой архитектуры, которая была бы одинаковой с точки зрения пользователя для всех моделей системы независимо от цены и производительности каждой из них*.

Огромные **преимущества** такого подхода, позволяющего *сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели* были быстро оценены как производителями компьютеров, так и пользователями.

Начиная с этого времени практически все фирмы-поставщики компьютерного оборудования взяли на вооружение эти принципы, поставляя **серии совместимых компьютеров**.

Следует заметить однако, что *со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность внесения радикальных изменений архитектуру и способы организации вычислительных систем*.

В настоящее время *одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий*, является **ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств**.

Это объясняется прежде всего тем, что *для конечного пользователя в конце концов важно программное обеспечение*, позволяющее решить его задачи, а не выбор той или иной аппаратной платформы.

Переход от однородных сетей программно совместимых компьютеров к построению неоднородных сетей, включающих компьютеры разных фирм-производителей, **в корне изменил и точку зрения на саму сеть**: *из сравнительно простого средства обмена информацией она превратилась в средство интеграции отдельных ресурсов - мощную распределенную вычислительную систему, каждый элемент которой (сервер или рабочая станция) лучше всего соответствует требованиям конкретной прикладной задачи.*

Этот переход выдвинул **ряд новых требований.**

- Прежде всего такая *вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач*.
- Во-вторых, она *должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т.е. обеспечивать мобильность программного обеспечения.*
- В третьих, эта *среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть*.

В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения **сформировалась концепция открытых систем**, *представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы*.

Одним из *вариантов моделей открытой среды* является **модель** OSE (Open System Environment), предложенная комитетом IEEE POSIX.

На основе этой модели национальный институт стандартов и технологии США выпустил документ "Application Portability Profile (APP). The U.S. Government's Open System Environment Profile OSE/1 Version 2.0", который *определяет рекомендуемые для федеральных учреждений США спецификации в области информационных технологий, обеспечивающие мобильность системного и прикладного программного обеспечения*.

Все ведущие производители компьютеров и программного обеспечения в США в настоящее время придерживаются требований этого документа.

Методы оценки производительности

Единицей измерения производительности компьютера является время: компьютер, выполняющий тот же объем работы за меньшее время является более быстрым.

Время выполнения любой программы измеряется в секундах.

Часто производительность измеряется как скорость появления некоторого числа событий в секунду, так что меньшее время подразумевает большую производительность.

Однако в зависимости от того, что мы считаем, **время может быть определено различными способами**.

1. Наиболее простой *способ определения времени называется астрономическим временем, временем ответа (response time), временем выполнения(execution time) или прошедшим временем (elapsed time).*

Это задержка выполнения задания, включающая буквально все: работу процессора, обращения к диску, обращения к памяти, ввод/вывод и накладные расходы операционной системы.

Однако при работе в мультипрограммном режиме во время ожидания ввода/вывода для одной программы, процессор может выполнять другую программу, и система не обязательно будет минимизировать время выполнения данной конкретной программы.

2. Для **измерения времени работы процессора на данной программе** используется **специальный параметр - время ЦП (CPU time),** которое не включает время ожидания ввода/вывода или время выполнения другой программы.

Очевидно, что время ответа, видимое пользователем, является полным временем выполнения программы, а не временем ЦП.

Время ЦП может далее делиться на время, потраченное ЦП непосредственно на выполнение программы пользователя и называемое **пользовательским временем ЦП**, и время ЦП, затраченное операционной системой на выполнение заданий, затребованных программой, и называемое **системным временем ЦП**.

В ряде случаев **системное время ЦП** игнорируется из-за возможной неточности измерений, выполняемых самой операционной системой, а также из-за проблем, связанных со сравнением производительности машин с разными операционными системами.

С другой стороны, системный код на некоторых машинах является пользовательским кодом на других и, кроме того, практически никакая программа не может работать без некоторой операционной системы.

Поэтому при измерениях производительности процессора часто используется сумма пользовательского и системного времени ЦП.

В большинстве современных процессоров скорость протекания процессов взаимодействия внутренних функциональных устройств определяется не естественными задержками в этих устройствах, а задается *единой системой синхросигналов, вырабатываемых некоторым генератором тактовых импульсов, как правило, работающим с постоянной скоростью.*

Дискретные временные события называются **тактами синхронизации (clock ticks), просто тактами (ticks), периодами синхронизации (clock periods), циклами (cycles) или циклами синхронизации (clock cycles)**.

Разработчики компьютеров обычно говорят о периоде синхронизации, который определяется либо своей длительностью (например, 10 наносекунд), либо частотой (например, 100 МГц).

Длительность периода синхронизации есть величина, обратная к частоте синхронизации.

Таким образом, **время ЦП для некоторой программы** может быть выражено двумя способами: **количеством тактов синхронизации для данной программы, умноженным на длительность такта синхронизации,** либо **количеством тактов синхронизации для данной программы, деленным на частоту синхронизации.**

Важной характеристикой, часто публикуемой в отчетах по процессорам, является **среднее количество тактов синхронизации на одну команду - CPI (clock cycles per instruction).** При известном количестве выполняемых команд в программе этот параметр позволяет быстро оценить время ЦП для данной программы.

Таким образом, **производительность ЦП зависит от трех параметров:** *такта (или частоты) синхронизации, среднего количества тактов на команду и количества выполняемых команд.*

Невозможно изменить ни один из указанных параметров изолированно от другого, поскольку базовые технологии, используемые для изменения каждого из этих параметров, взаимосвязаны:

- *частота синхронизации определяется технологией аппаратных средств и функциональной организацией процессора*;
- *среднее количество тактов на команду зависит от функциональной организации и архитектуры системы команд;*
- *а количество выполняемых в программе команд определяется архитектурой системы команд и технологией компиляторов.*

Когда сравниваются две машины, **необходимо рассматривать все три компоненты, чтобы понять относительную производительность.**

В процессе поиска стандартной единицы измерения производительности компьютеров было принято несколько популярных единиц измерения, вследствие чего несколько безвредных терминов были искусственно вырваны из их хорошо определенного контекста и использованы там, для чего они никогда не предназначались.

В действительности единственной подходящей и надежной единицей измерения производительности является время выполнения реальных программ, и все предлагаемые замены этого времени в качестве единицы измерения или замены реальных программ в качестве объектов измерения на синтетические программы только вводят в заблуждение.

Одной из *альтернативных единиц измерения производительности процессора* (по отношению к времени выполнения) является **MIPS - (миллион команд в секунду)**. Имеется несколько различных вариантов интерпретации определения MIPS.

В общем случае MIPS *есть скорость операций в единицу времени*, т.е. для любой данной программы MIPS есть *просто отношение количества команд в программе к времени ее выполнения*.

Таким образом, *производительность может быть определена как обратная к времени выполнения величина, причем более быстрые машины при этом будут иметь более высокий рейтинг MIPS*.

Положительными сторонами MIPS является то, что эту характеристику легко понять, особенно покупателю, и что более быстрая машина характеризуется большим числом MIPS, что соответствует нашим интуитивным представлениям.

Однако использование MIPS в качестве метрики для сравнения наталкивается на **три проблемы**.

- Во-первых, MIPS зависит от набора команд процессора, что затрудняет сравнение по MIPS компьютеров, имеющих разные системы команд.
- Во-вторых, MIPS даже на одном и том же компьютере меняется от программы к программе.
- В-третьих, MIPS может меняться по отношению к производительности в противоположенную сторону.

Измерение производительности компьютеров **при решении научно-технических задач, в которых существенно используется арифметика с плавающей точкой**, всегда вызывало особый интерес.

Именно для таких вычислений впервые встал вопрос об измерении производительности, а по достигнутым показателям часто делались выводы об общем уровне разработок компьютеров.

Обычно **для научно-технических задач производительность процессора оценивается в MFLOPS (миллионах чисел-результатов вычислений с плавающей точкой в секунду, или миллионах элементарных арифметических операций над числами с плавающей точкой, выполненных в секунду).**

Как единица измерения, MFLOPS, *предназначена для оценки производительности только операций с плавающей точкой, и поэтому не применима вне этой ограниченной области.* Например, программы компиляторов имеют рейтинг MFLOPS близкий к нулю вне зависимости от того, насколько быстра машина, поскольку компиляторы редко используют арифметику с плавающей точкой.

Ясно, что *рейтинг MFLOPS зависит от машины и от программы*. Этот термин менее безобидный, чем MIPS. *Он базируется на количестве выполняемых операций, а не на количестве выполняемых команд.* По мнению многих программистов, одна и та же программа, работающая на различных компьютерах, будет выполнять различное количество команд, но одно и то же количество операций с плавающей точкой. Именно поэтому *рейтинг MFLOPS предназначался для справедливого сравнения различных машин между собой.*

Однако и с MFLOPS не все обстоит так безоблачно.

1. Прежде всего, это связано с тем, что **наборы операций с плавающей точкой не совместимы на различных компьютерах.**

Например, в суперкомпьютерах фирмы Cray Research отсутствует команда деления (имеется, правда, операция вычисления обратной величины числа с плавающей точкой, а операция деления может быть реализована с помощью умножения делимого на обратную величину делителя). В то же время многие современные микропроцессоры имеют команды деления, вычисления квадратного корня, синуса и косинуса.

2. Другая, осознаваемая всеми, проблема заключается в том, что **рейтинг MFLOPS меняется не только на смеси целочисленных операций и операций с плавающей точкой, но и на смеси быстрых и медленных операций с плавающей точкой.**

Например, программа со 100% операций сложения будет иметь более высокий рейтинг, чем программа со 100% операций деления.