© 2006 г. Вл. В. Воеводин
(НИВЦ МГУ им.М. В. Ломоносова)

РЕШЕНИЕ БОЛЬШИХ ЗАДАЧ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ[1]

Данная работа посвящена исследованию методов построения вычислительных сред для решения больших задач на доступных распределенных неоднородных компьютерных ресурсах. Разработанные подходы положены в основу системы X-Com, предназначенной для проведения широкомасштабных метакомпьютерных расчетов. Кратко описаны принципы построения системы, приведены результаты реальных расчетов.

1. Введение

Большие задачи — это тот класс задач, для прямого решения которых не хватает ресурсов доступных компьютеров. Такие задачи появились одновременно с рождением вычислительной техники, и будут существовать всегда. Они всегда вызывают повышенный интерес, поскольку являются отражением новых компьютерных методов проведения исследований в различных областях науки, расширяя и дополняя традиционные подходы. Вычислительная аэро- и гидродинамика, молекулярное моделирование, квантово-химические методы, вычислительные технологии биоинформатики и биоинженерии имеют колоссальный потенциал для решения реальных задач, однако все они предъявляют исключительно жесткие требования к параметрам используемых компьютерных систем, превышающие возможности существующей вычислительной техники. Чтобы к подобным задачам все же можно было подступиться, их постановки приходится упрощать, понижать размер или размерность, снижать точность проведения исследований. Процесс упрощения продолжается до тех пор, пока не удается достичь той грани, где задача не теряет физического смысла, и может быть решена на доступных компьютерах. Ищется тонкая грань. Своего рода точка равновесия. Шаг в одну сторону и задача становится не интересной, шаг в другую - ее решение выходит за разумное время. Большие задачи всегда балансируют на этой грани. Как только компьютерная техника предоставляет новые возможности, они мгновенно используются для смещения точки в пользу задачи, а затем опять работа на грани, но уже на грани новых возможностей. Поддерживать такое состояние не просто, многие характеристики нужно привести в согласованное состояние, множество тонких эффектов нужно учитывать и уметь контролировать.

Последние десять лет идет быстрое развитие сетевых технологий. Пропускная способность каналов связи неуклонно растет, латентность взаимодействия компьютеров снижается, коммуникационные характеристики многих локальных и глобальных сетей уже превзошли параметры первых кластерных систем, и процесс развития сетей продолжается. Обладая целым рядом положительных свойств, сети создают основу для нового поколения компьютерных систем — распределенных вычислительных сред. Берем какое-то количество компьютеров

[1]Работа выполнена при поддержке грантов РФФИ: № 05-07-90206, 05-07-08009.

в локальной сети или Интернет: офисные машины, рабочие места в учебных классах, кластеры, и рассматриваем это как единый параллельный компьютер. Или суперкомпьютер — все определяется масштабностью такого объединения [1]. Суммарная производительность компьютеров сети весьма высока, и, следовательно, есть необходимый потенциал для создания инструмента решения больших задач. В отличие от традиционных суперкомпьютеров данный инструмент не является уникальной единичной установкой, он дешев и доступен всем. В самом деле, среды формируются на основе уже действующих компьютеров сети, новых затрат практически не требуется, что и делает такой подход экономически привлекательным. Более того, развитие и модернизация вычислительных сред происходит автоматически по мере обновления компьютерного парка сети.

2. Распределенные вычислительные среды

Вычислительные среды обладают параллельной архитектурой и распределенной памятью, что на первый взгляд делает их похожими на вычислительные кластеры. Но это лишь внешнее сходство. Истинная сущность распределенных вычислительных сред совершенно иная и определяется набором свойств, которого не было ни у одной из ранее существовавших компьютерных систем.

1) *Масштабность.* Суммарное число процессоров, общий объем памяти, число одновременно работающих пользователей и приложений в вычислительных средах может быть огромным. Чем больше значения подобных параметров, тем сложнее обеспечить эффективность расчета. Вместе с этим, чем масштабнее объединение компьютеров в среде, тем больший потенциал для решения больших задач.

2) *Распределенность.* Вычислительные узлы среды могут быть удалены друг от друга на тысячи километров, что вызывает большую латентность в их взаимодействии.

3) *Неоднородность.* В состав среды могут входить компьютеры произвольной архитектуры и производительности, работающие каждый в своем режиме под управлением различных операционных систем, связанные между собой каналами с различной пропускной способностью.

4) *Динамичность.* Архитектура среды может изменяться. Любой компьютер может выйти из состава среды в любой момент времени, но это не должно приводить к аварийной остановке вычислительного процесса. К среде могут добавляться новые ресурсы, которые нужно уметь оперативно подключать к текущему расчету.

5) *Различная административная принадлежность.* Вычислительные узлы могут принадлежать различным людям, организациям, государствам. У всех свои правила администрирования и использования, свои режимы работы, свои политики обеспечения безопасности.

Присутствие даже одного из таких свойств создает серьезные проблемы на практике, а в распределенных средах все они зачастую присутствуют одновременно. Особенно ярко это проявляется в сети Интернет, представляющей для нас особый интерес. Если мы хотим использовать колоссальный потенциал вычислительных сред для решения больших задач, нужно научиться работать в этих условиях. Нужна такая модель вычислительного процесса, в которой все эти свойства были бы отражены столь же естественно, сколь естественно они проявляются в самих вычислительных средах. Если у нас в распоряжении оказалось большое

число компьютеров с различной архитектурой, принадлежностью, назначением, режимом работы, которые время от времени могут отдавать свои ресурсы, то нужно найти технологию решения задач именно в таких условиях. Такова реальность. Масштабность вычислительных сред создает необходимые предпосылки для построения компьютерных систем огромной производительности, что и требуется для решения больших задач. Основной вопрос заключается в нахождении эффективного способа организации процесса вычислений с использованием систем такого класса.

Все указанные выше свойства в той или иной мере повлияют на решение задач в распределенных вычислительных средах, однако относительно неоднородности и динамичности нужно сделать несколько особых замечаний. Начать следует с того, что в средах *все вычислительные узлы потенциально ненадежны.* При этом мы делаем акцент не на поломке или сбое в работе компьютера, а на факте его возможного выхода в любой момент времени из состава среды. В самом деле, если кто-то согласился отдать ресурсы своего компьютера для расчета, то на правах хозяина он всегда может забрать его назад под свои собственные нужды. Опять же, непредсказуемый сбой в работе маршрутизатора может привести к отключению от сети целого сегмента. Любой узел может "исчезнуть" вне зависимости от того, завершил ли он обработку выделенного ему задания или нет. Звучит парадоксально, но у компьютеров, составляющих вычислительную среду, нет никаких обязательств перед самой средой. В других компьютерных системах такого произвола нет. Узлы кластера работают только на кластер, и обязательства перед системой максимальные. Компьютеры, входящие в грид-системы, соблюдают определенный регламент в предоставлении своих ресурсов, определяющий их вклад в работу системы в целом. В нашем случае все по-другому. Ориентация на подключение максимального числа компьютеров к расчетам не позволяет выдвигать каких-либо требований на режим их использования, что и приводит к их потенциальной ненадежности.

Далее, *состав узлов среды может меняться со временем.* Причем изменения возможны не только в структуре среды, меняться могут и отдельные характеристики узлов, в частности, их производительность. Владелец компьютера может разрешить использование узла на фоне его собственной работы. В этом случае вклад узла в работу всей среды будет определяться уровнем его текущей загрузки, который спрогнозировать невозможно. И, наконец, *между узлами нет регулярной статической структуры связей.* Мы не вправе рассчитывать, что компьютеры среды физически будут связаны между собой каким-либо регулярным способом, например, составят кольцо, звезду или решетку. Нельзя полагаться даже на то, что исходная коммуникационная структура связи узлов среды будет постоянной все время расчета: изменение маршрутизации в Интернет отражается на физической структуре связей между узлами среды. И, конечно же, реальная пропускная способность каналов связи может меняться, поскольку в сети мы, как правило, не одиноки.

Итак, как можно описать основу вычислительной среды? Как *неструктурированное множество различных ненадежных вычислительных узлов, которые иногда могут предоставлять какую-то часть своих ресурсов.* По существу, это все, что мы имеем. Ничего из такого описания нельзя выкинуть,

как ничего более конструктивного нельзя и добавить. Компьютерное сообщество, в котором каждый узел живет по своим законам, но может привнести малое, формируя мощный вычислительный потенциал среды в целом. Наша задача — не нарушая законов сообщества, найти способы его объединения и перенаправить имеющийся потенциал на решение реальных больших задач.

3. Вычислительные среды и прикладные задачи

Особенности архитектуры компьютерных систем всегда накладывают отпечаток на этапы технологической цепочки решения задач. Так и в нашем случае: структура процесса вычислений в среде, технология программирования, требования на систему поддержки выполнения распределенных программ — все это будет определяться свойствами вычислительных сред (рис. 1). Чтобы понять, как решать задачи в средах, мы должны последовательно ответить на ряд вопросов. Каким требованиям должна удовлетворять задача, чтобы ее решение в вычислительной среде было бы целесообразным и эффективным? Какова структура вычислительного процесса решения задач? Как должна быть устроена программа для распределенных сред? Как обеспечить поддержку эффективного выполнения распределенных программ? Мы рассмотрим все эти вопросы, но начнем с главного — с обсуждения класса задач, которые можно решать на подобных компьютерных системах.

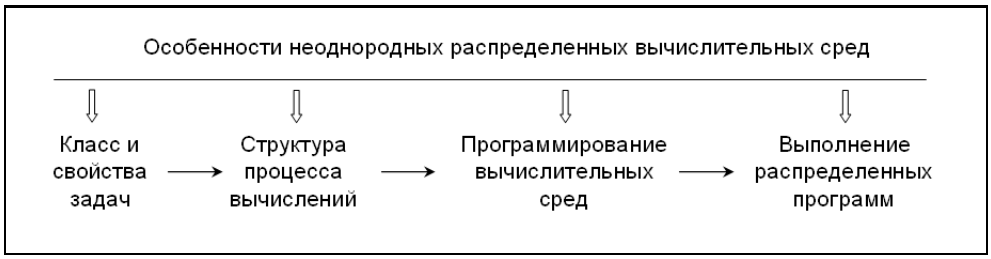

Рис. 1. *Решение задач в распределенных вычислительных средах*

Масштабность вычислительной среды предполагает объединение большого числа компьютеров, работающих независимо друг от друга. Это значит, что среда обладает мощным ресурсом параллелизма, поэтому столь же мощным внутренним параллелизмом должна обладать и решаемая задача. Более того, поскольку задача является “большой”, то разумно предположить, что она в состоянии не только занять все предоставляемые средой ресурсы, но и использовать их длительное время.

Распределенность среды неизбежно вызовет заметные задержки во взаимодействии удаленных узлов. Единственный способ повысить эффективность решения задачи состоит в выделении частей с высокой вычислительной мощностью. Чем выше отношение числа операций к объему данных, необходимых для работы некоторой части программы, тем выше эффективность ее обработки. Чем больше параллельные процессы будут считать, и чем реже они будут общаться, тем эффективнее будет проходить весь вычислительный процесс. Отсюда и требование на задачу: для эффективной организации ее решения в распределенной среде необходима возможность ее разбиения на независимые части с высокой вычислительной мощностью. Заметим, что уже сегодня это требование не является

столь жестким, каким может показаться на первый взгляд. В настоящее время самой доступной технологией построения локальной сети организации является Fast Ethernet. Это означает, что 1 Мбайт данных между компьютерами этой сети будет передан меньше, чем за 1 секунду, чем и определяется нижняя граница времени обработки такого объема данных на узле для разумной организации процесса решения задачи. Со временем влияние этого требования будет постоянно ослабевать, сетевые технологии быстро развиваются, и все более широкий класс задач можно будет решать в вычислительных средах.

Динамичность конфигурации среды почти сразу приводит нас к задачам с тривиальной коммуникационной структурой, не предполагающей взаимодействия параллельных процессов по сколько-нибудь сложной схеме. Если узел **А** вычисляет данные для узла **Б**, то **Б** не может начать работу пока **А** эти данные ему не предоставит. Однако **А** в любой момент может выйти из состава среды, и **Б** будет ждать, пока часть задачи, ранее выданная **А**, не будет выполнена каким-то другим узлом. Мы используем, но не являемся хозяевами ни узла **А**, ни какого-либо другого узла среды, поэтому гарантировать вычисление данных для **Б** к заранее определенному сроку не можем. Вспомним и о том, что узлы среды не имеют тесной связи между собой, поэтому опять-таки обработка одних частей задачи не должна сильно зависеть от скорости и порядка обработки других. Вопрос, можно ли эффективно организовать вычислительный процесс с нетривиальной коммуникационной структурой (например, решетка или дерево) в средах с описанными выше свойствами, очень интересен, но его исследование оставим за рамками данной работы.

Потенциальная неоднородность среды, под которой мы сейчас будем понимать различие в производительности узлов, проявляется двояко. С одной стороны, она выдвигает требования аналогичные тем, что мы только что обсуждали, говоря о динамичности, добавляя аргументов в пользу задач с тривиальной коммуникационной структурой. С другой стороны, неоднородность заставляет аккуратно подбирать объем работы, выдаваемой разным узлам. Возможность формирования вычислительной нагрузки в зависимости от мощности узла должна быть изначально предусмотрена в исходной задаче. В реальности, для нас это является не самым сложным требованием: внутренний параллелизм задачи велик, поэтому высокую скорость работы узла можно учесть простым объединением нескольких порций входных данных в один передаваемый блок.

На практике тяжелых задач с подобными свойствами много. К этому классу можно отнести многие задачи, возникающие в криптографии, комбинаторные задачи или задачи, использующие методы типа Монте-Карло. Такими же свойствами обладают задачи проектирования новых лекарственных препаратов, решение которых опирается на независимый анализ взаимодействия большого числа химических соединений с белками-мишенями. Специальные подходы к исследованию ряда задач, в частности, задач электродинамики на основе решения интегральных уравнений, опираются на работу с большими матрицами, генерация которых является основным вычислительно емким местом. В таких случаях очень часто помогает то, что вычисление отдельных элементов матриц можно производить параллельно. Именно по такой схеме решалась задача дифракции электромагнитного поля на диэлектрическом анизотропном теле произвольной формы [2]. Задача была

решена за 26 дней на 4-х разных кластерах суперкомпьютерного комплекса НИВЦ МГУ. Использовались только те интервалы времени, когда процессоры кластеров не были заняты пользователями. Весь расчет проведен на фоне штатной работы суперкомпьютерного комплекса только за счет использования периодов незанятости процессоров. Эффективность работы созданной вычислительной среды составила 98%. Решение этой же задачи в обычном режиме заняло бы 4 года работы одного компьютера.

Перечисление задач с такими свойствами можно продолжать и дальше, их немало в самых разных областях. Давайте сделаем следующий шаг и попытаемся понять, как нужно организовать процесс вычислений, чтобы решение задач в вычислительных средах было бы эффективным.

4. Процесс вычислений в распределенной среде

Приступая к созданию программы для компьютера с параллельной архитектурой, мы должны, прежде всего, представлять модель вычислительного процесса. Нужна четкая картина, показывающая, как параллелизм компьютера отражается в структуре вычислений. Как организованы параллельные потоки команд в программе? Как они взаимодействуют между собой? Что в структуре вычислений статично, а что может изменяться в динамике? Подобных вопросов не много, но они помогают представить общую схему выполнения параллельной программы на компьютере.

Посмотрим на структуру процесса вычислений в распределенных средах. Как в среде может быть выражен параллелизм? Большое число узлов каждый со своим потоком управления сразу дает однозначный ответ — с помощью независимых процессов. Нити, а тем более идеи data-flow обработки, не применимы из-за отсутствия единого адресного пространства. Каждый процесс работает независимо, нет общей памяти, нет общих переменных или других объектов, и все общение между процессами проходит опосредованно через ту или иную форму обмена сообщениями. Конечно же, всегда можно смоделировать более удобные понятия высокого уровня, но эффективность вычислительного процесса все равно будет определяться физической основой среды.

Среда динамична, это ее одно из самых характерных свойств, число узлов может легко меняться, поэтому и количество процессов жестко фиксировать нельзя. Появление или исчезновение узла среды приводит к изменению числа параллельно работающих над задачей процессов, что должно столь же естественно и динамично отражаться в структуре всего процесса вычислений. Аналогичные соображения приводят и к отказу от статического распределения работы. Если процесс исчез, то назначенная ему ранее работа должна быть в динамике перераспределена между оставшимися. Если появились новые процессы, то они должны сразу подключаться к решению задачи, взяв на себя часть общей работы.

Должны ли все процессы работать по одной и той же программе? Вопрос очень интересный и неоднозначный. С одной стороны, однородность процессов удобна. Целое множество задач может быть эффективно решено по классической схеме, в рамках которой все процессы равноправны и работают по одной программе, выполняя однотипную обработку различных наборов данных. Более того, потенциальная ненадежность узлов среды и отсутствие фиксированной структуры

связей между ними ставят под сомнение целесообразность их специализации и опять-таки являются серьезными аргументами в пользу одинаковости процессов. Вместе с этим, отражение нетривиальных свойств алгоритмов в разумной структуризации распределенной среды, безусловно, является перспективным направлением. Почему бы, например, не организовать макроконвейер, ступенями которого являются отдельные вычислительные среды? В принципе, в каком-то виде это можно сделать и сейчас, однако истинный потенциал этого направления раскроется позже по мере развития компьютерной отрасли и накопления необходимого опыта в решении представительного класса задач. Оставим возможность специализации пока в стороне, и в рамках данной работы будем предполагать, что все процессы работают по одной программе. Все, за исключением одного: должен быть выделенный процесс, который всегда может гарантированно дать ответ, завершен ли общий процесс решения задачи или нет, который может отследить исчезновение узла и необходимость перераспределения работы, который знает объем еще невыполненной работы и может выделить из нее часть для новых узлов.

Мы естественным образом приходим к структуре вычислений, которая известна как Мастер/Рабочие (Master/Slaves). Есть один выделенный процесс Мастер, управляющий всем ходом расчета. Есть множество равноправных между собой процессов Рабочих. Все Рабочие получают от Мастера задания, выполняют их по одной и той же программе, и возвращают полученный результат процессу Мастеру. Тривиальная коммуникационная структура задачи не предполагает непосредственного общения между Рабочими, им требуется только взаимодействие с Мастером (рис. 2а).

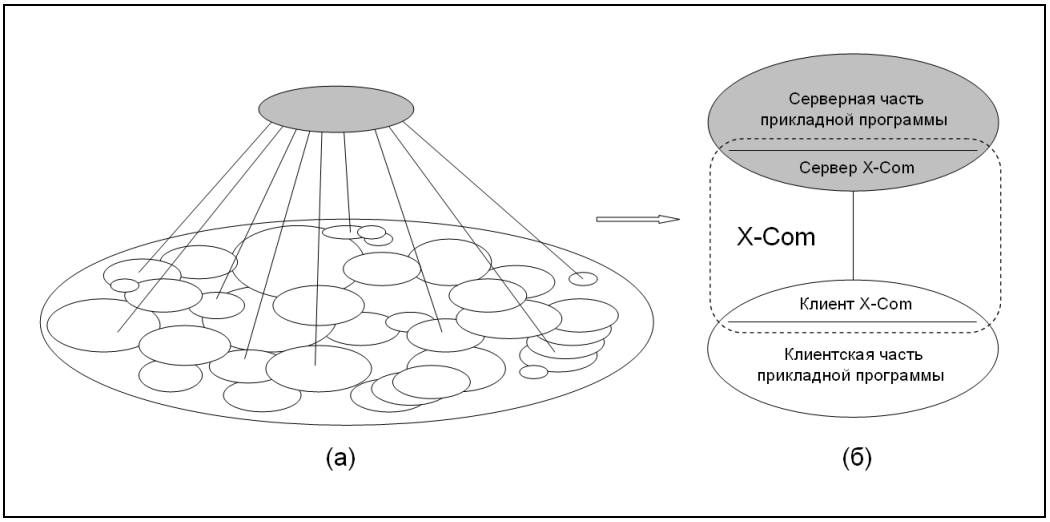

Рис. 2. *Управляющий сервер и вычислительные узлы распределенной среды. Компоненты X-Com и прикладной программы на сервере и узлах*

Функции контроля и управления, возложенные на процесс Мастер, безусловно, предъявляют повышенные требования к надежности его работы. Он не может работать на обычном узле вычислительной среды, поскольку исчезновение узла приведет к остановке всего расчета. Структура вычислений выдвигает встречные требования к аппаратной части вычислительной среды, в состав которой *должен*

входить хотя бы один надежный элемент, необходимый для работы процесса Мастер. Это дополнительное требование на состав вычислительной среды, но заметим, что требование минимальное.

5. Система X-Com: программирование вычислительных сред

Определившись со структурой вычислительного процесса, нужно решить вопрос, как и с помощью каких технологий должна быть записана программа, реализующая исходную задачу. Следуя логике организации процесса вычислений в распределенной среде, программа должна также состоять из двух частей. Одна отвечает процессу Мастер, знает весь объем работы и возможности его распределения между подчиненными процессами, обладает информацией о потенциальном параллелизме задачи. Другая часть программы соответствует процессам Рабочим, которые выполняют обработку поступивших от Мастера данных, ничего не зная о других Рабочих и их числе. Заметим, что процесс Мастер в каждый момент времени знает текущий статус расчета и выполняет основную координирующую роль, однако в силу динамичности среды активной стороной в паре Мастер-Рабочие выступает, не он, а Рабочие, выставляющие Мастеру запрос на очередное задание. С этой точки зрения программа полностью отвечает классической клиент-серверной архитектуре, поэтому процесс Мастер мы будем называть сервером, а каждый экземпляр процесса Рабочего — клиентом.

Для выполнения программ в распределенных вычислительных средах необходим посредник, который взял бы на себя все сервисные и управляющие функции. Такую роль в наших исследованиях играет система X-Com, созданная в НИВЦ МГУ. Она скрывает от прикладной программы все особенности конкретной распределенной среды, позволяя организовать процесс решения задачи по единой модели на любых вычислительных ресурсах. Пользователь знает лишь общую схему, согласно которой сервер раздает задания клиентам, а все детали реализации такой схемы система X-Com берет на себя.

Одна из функций системы X-Com — это обеспечение обмена данными между сервером и клиентами. Прикладная программа не должна зависеть от конкретных механизмов или технологий организации взаимодействия процессов, характерных для какой-то операционной системы или сети. Реализация всего транспортного уровня заложена в систему X-Com, предоставляющей прикладной программе простой интерфейс, выраженный в терминах общения между клиентом и сервером (рис. 2б). Предположим, что клиент программы хочет обратиться с запросом к серверу. Для этого он вызывает соответствующую функцию, и управление передается клиенту X-Com, работающему на том же узле среды, что и клиент прикладной программы. Клиент X-Com связывается с сервером X-Com, далее следует запрос к серверу прикладной программы, а полученный ответ по той же цепочке возвращается назад расчетной части программы.

Распределение заданий всегда проходит через сервер X-Com, и именно он в каждый момент времени контролирует ход процесса вычислений. Узел вернул результат расчета, сервер пометил соответствующую часть работы выполненной. Если к среде подключатся новые узлы, то они на общих правах получат порции данных для обработки. Если от какого-то узла долго нет ответа, то сервер считает его вышедшим из состава среды, а выданную ему ранее порцию данных отдаст

на обработку другому узлу. Пользователь не должен думать о потенциальной ненадежности узлов, контроль за выходом узла из состава среды и необходимостью повторной обработки соответствующих порций данных полностью лежит на X-Com. Если случится, что через какое-то время ответ от пропавшего узла все же придет, то результат обработки порции данных, попавший к серверу последним, будет игнорироваться. Если порция данных была выдана не двум, а большему числу узлов, то система сохранит ответ, пришедший первым.

Желание подключить максимум вычислительных ресурсов возлагает на X-Com и функцию контроля за режимом их использования. Часть компьютеров может быть полностью отдана в монопольное использование средой, это самый простой и удобный режим. Другие компьютеры можно использовать только в периоды их простоя, когда они не заняты основной работой. Система должна фиксировать факт простоя, занимать свободный компьютер, а в случае возобновления штатной активности сворачивать на нем свою деятельность. Третий режим связан с подключением в среду компьютеров, на которых работает та или иная система управления прохождением пользовательских заданий (PBS, Cleo и другие). В таком режиме организуется работа на большинстве суперкомпьютерных систем, где нужно обеспечить бесконфликтное одновременное выполнение целого множества различных программ. Система X-Com берет на себя взаимодействие с системой управления заданиями, от своего имени ставит клиентские части прикладной программы в очередь, делая возможным использование и таких вычислительных ресурсов в среде.

Очень важно, что в рамках предложенного способа организации вычислительных сред использование ресурсов, имеющих различную производительность, никаких проблем не вызывает. Классический подход программирования неоднородных систем предполагает аккуратное разбиение множества данных на блоки, размер которых выбирается исходя из мощности соответствующего вычислительного узла. Тяжелая задача, которая усложняется во много раз, если пытаться такую программу сделать еще и переносимой. В нашем же случае неоднородность ресурсов среды учитывается автоматически. Задача является “большой”, следовательно, порций данных много, и узлы в процессе работы программы сами возьмут себе столько, сколько смогут обработать. Распределение заданий идет не директивно из “центра”, а регулируется “снизу” скоростью работы самих узлов. Так задача поиска молекул-ингибиторов для заданных белков-мишеней была решена за 11 дней с использованием 270 процессоров трех кластеров и одного учебного класса НИВЦ МГУ (Москва) и ЮУрГУ (Челябинск). Эффективность расчета составил более 97%, хотя среда формировалась из самых разных процессоров: Intel Pentium III 500MHz, Intel Xeon 2.6 GHz, Intel Xeon EM64T 3.2 GHz, AMD Opteron 2.2 GHz и других.

Немного неожиданно, но в нашем случае не нужен сложный и тяжелый механизм контрольных точек, традиционно используемый в системах расчета больших задач. Ненадежность вычислительных узлов означает, что обработка любой части задачи может быть прервана в любое время. Это нормальный режим функционирования среды, который уже был учтен при проектировании системы X-Com: за составом среды и необходимостью повторной обработки порций данных следит сервер X-Com. Какой-то узел вышел из строя, обработка выданных ему данных не была завершена, поэтому она автоматически будет инициирована сервером заново на другом компьютере. Системе нет смысла перезапускать обработку единичной порции

с прерванного места. Задача — большая, порций — огромное количество, проще и эффективнее запустить повторную обработку на ином доступном ресурсе, чем постоянно обеспечивать возможность перезапуска процесса на каждом из узлов среды. Одновременно заметим, что выход узла из состава среды никогда не приведет к падению распределенной программы в целом, поскольку ее серверная часть и сервер X-Com работают на надежном сервере среды.

Система X-Com является одним из примеров реализации изложенных в данной работе принципов. Детали реализации и дистрибутивы X-Com доступны на http://X-Com.Parallel.ru.

6. Система X-Com: поддержка масштабных расчетов

Эффективность предложенной схемы в процессе расчета определяется соотношением трех параметров: временем обработки порций на узлах, объемом данных, передаваемых между сервером и узлами, и числом самих узлов. Интуитивно понятно, чем меньше узлы ждут задания, тем эффективнее работает программа. Основными причинами простоя узлов являются низкая скорость передачи данных по сети и занятость управляющего сервера обслуживанием других узлов. Причины серьезные, но на практике их влияние можно значительно ослабить, если перейти к иерархической структуре организации среды, показанной на рис. 3.

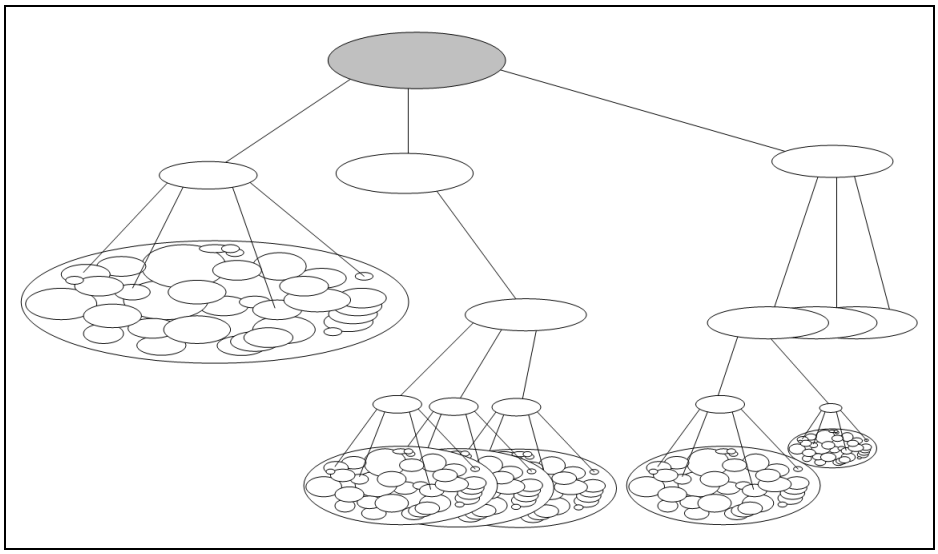

Рис. 3. *Иерархическая структура организации распределенной среды*

Между сервером, который теперь будем называть центральным, и вычислительными узлами, расположим *промежуточные серверы.* В такой структуре у каждого сервера, центрального или промежуточного, есть какое-то число подчиненных ему узлов или других промежуточных серверов. У каждого промежуточного сервера и вычислительного узла есть один сервер более высокого уровня иерархии, которому он подчинен. Все вычислительные узлы, как и прежде, расположены в листовых вершинах. Промежуточный сервер получает от своего "начальника" входные данные задачи, которые распределяет для обработки между своими "подчиненными".

Заметим, что предложенная модификация структуры вычислительной среды по существу не требует введения новых сущностей, поскольку промежуточные серверы лишь объединяют в себе функции центрального сервера и вычислительного узла. Для всех подчиненных вершин промежуточный сервер играет роль центрального сервера, раздающего порции данных на обработку. Для вышележащего сервера промежуточный сервер выглядит обычным вычислительным узлом, способным выполнять повышенный объем работы.

Такая схема значительно улучшает масштабируемость вычислительной среды. Центральному серверу уже нет необходимости поддерживать соединения со всеми узлами (рис. 2а), число которых может быть сколь угодно большим. В новой схеме каждому серверу необходимо обеспечить взаимодействие только с компьютерами следующего уровня, а этим числом можно управлять при формировании среды. При таком способе организации вычислительной среды *ни один узел в системе не знает ни общего числа уровней иерархии, ни общего числа узлов.* Каждая вершина знает только о том, какому серверу она подчинена непосредственно и какие вершины следующего уровня непосредственно подчинены ей. Ни один компьютер, даже центральный сервер, не имеет полного описания текущей конфигурации всей вычислительной среды, и ни в одной точке системы теперь нет внутренних управляющих структур, размер которых был бы пропорционален общему числу ее элементов.

Введение промежуточных серверов решает целый ряд проблем, в частности, позволяет скрыть низкую скорость передачи данных между компьютерами среды. Пока подчиненные узлы заняты обработкой выданных заданий, сервер может скачивать новый блок данных, не дожидаясь от них сигнала о завершении работы. Более того, во многих расчетах часть входных данных, передаваемых на узлы, одинакова — эта часть один раз передается промежуточному серверу, который уже и дублирует ее по подчиненным узлам. И, наконец, промежуточный сервер может объединять входные/выходные данные узлов в блоки, снижая как интенсивность запросов к центральному серверу, так и накладные расходы, связанные с высокой латентностью взаимодействия в сети.

Иерархическая структура среды хорошо отображается на существующую структуру сетевых коммуникаций. Центральный сервер расположен в городе **А**, вычислительные узлы в городе **Б**. Города **А** и **Б** соединяет сетевая магистраль, по которой проходит весь обмен данными между компьютерами этих городов. В городе **Б** логично организовать промежуточный сервер, который возьмет на себя общение с центральным сервером из города **А** и будет координировать его взаимодействие с узлами из города **Б**. Аналогичные соображения применимы при подключении к вычислительной среде кластеров, учебных классов или компьютеров из одной локальной сети.

Все эти изменения не меняют саму прикладную программу, которая ничего не знает о промежуточных серверах и с ними никак не взаимодействует. Серверная часть программы передает порции данных серверу X-Com, который распределяет их для обработки по узлам. Часть непосредственно видимых серверу X-Com “узлов” может в реальности оказаться промежуточными серверами X-Com, транслирующими полученные задания дальше вплоть до уровня настоящих расчетных узлов. Никаких действий со стороны программиста не требуется, весь

механизм скрыт в самой системе X-Com. В частности, по такой схеме решалась задача определения скрытой периодичности в генетических последовательностях. Расчет был выполнен за 63 часа с использованием 407 процессоров из 10 организаций, расположенных в 8 городах и 6 часовых поясах. Несмотря на значительную географическую распределенность, эффективность процесса вычислений составила более 90%. Заметим, что время решения этой же задачи традиционными методами заняло бы больше 2 лет непрерывной работы одного хорошего персонального компьютера.

Подведем итог. Являясь посредником между вычислительной средой и программой, система X-Com учитывает все те свойства распределенных сред, о которых мы говорили ранее. Масштабность выражается в поддержке различных режимов использования ресурсов, отсутствием нестандартных компонентов ПО на узлах, простотой запуска клиента, масштабируемостью архитектуры системы, возможностью использования большинства доступных через Интернет компьютерных ресурсов. Распределенность сред сглаживается введением промежуточных серверов. Неоднородность сред поддерживается как многоплатформенностью системы, так и естественным механизмом учета различной производительности узлов. Потенциальная динамичность отслеживается сервером X-Com, который контролирует все выданные на обработку задания. Использование компьютеров, имеющих различную принадлежность, больших проблем не вызывает, поскольку установка и работа клиентов не требует привилегированных прав доступа к ресурсам узлов.

7. Вычислительные структуры и распределенные среды

Рассмотрим различные варианты организации работы в распределенных средах на основе предложенных технологий. В качестве базовой конструкции введем понятие X-среды решения некоторой задачи (рис. 4). Под X-средой задачи будем понимать совокупность программно-аппаратных средств, состоящей из X-сервера задачи и собственно вычислительной среды. X-сервер задачи, выполняющий управляющие функции в процессе решения задачи, объединяет сервер X-Com и серверную часть прикладной программы. Вычислительная среда предоставляет компьютерные ресурсы, которые сгруппированы в подходящую иерархическую структуру для выполнения вычислительной части программы. Предполагается, что X-среда сформирована согласно некоторому регламенту, определяемому политикой предоставления ресурсов и требованиями задачи. В частности, в X-среду входят только те узлы, которые подходят для данной задачи (требования на ОС, тип процессора, объем памяти и т.п.) и не нарушают административно установленных правил ее решения на предоставленных ресурсах. При создании X-среды можно наложить ряд дополнительных ограничений на ее функционирование, например, разрешить подключение к ней любых новых узлов либо допустить использование узлов только из некоторого фиксированного подмножества.

Введенное понятие X-среды является базовой конструкцией. Оно относится к одной конкретной задаче и описывает технологию ее решения на совокупности распределенных ресурсов. Набор реальных вычислительных ресурсов от запуска к запуску может меняться, но общая схема останется неизменной. Используем данное понятие для построения более универсального механизма — X-структуры

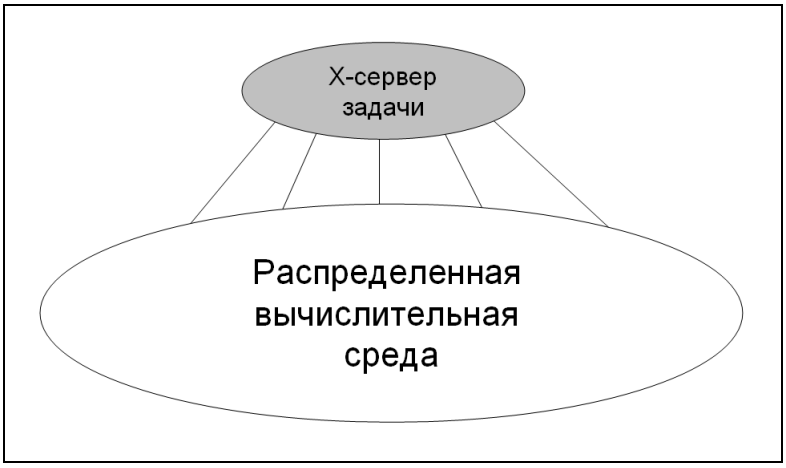

Рис. 4. *X-среда решения прикладной задачи*

(рис. 5). На вход X-структуры поступает поток разных задач, которые выполняются в порядке некоторой очереди. X-монитор выбирает очередную задачу, формирует под нее X-среду и следит за дальнейшим ходом ее работы. Попадая в очередь, задача предъявляет набор требований к узлам, на которых она может выполняться, что учитывается X-монитором при создании X-среды. По завершении задачи освободившиеся ресурсы могут быть использованы для старта новой задачи из очереди либо присоединены к другой, уже существующей, X-среде.

Задачи в X-структуру могут попадать различными путями. Самый очевидный вариант — это явная постановка задачи в очередь через командную строку или каким-либо иным способом. Более интересным вариантом является организация специализированных web-сервисов для решения задач по конкретным вычислительно сложным программам. Программа готовится заранее, а необходимый набор входных данных задается через web-форму или формируется динамически согласно некоторому интерфейсу, после чего задача автоматически ставится в очередь X-структуры. Когда от X-монитора поступит сигнал о завершении задачи, результаты расчета через web, электронную почту или другим способом будут возвращены назад. Предоставление подобного “вычислительного” web-сервиса является исключительно перспективным направлением по целому ряду причин. Во-первых, не требуется передавать исходных текстов, что решает многие вопросы авторского права. Во-вторых, пользователи освобождаются от необходимости самостоятельно выполнять инсталляцию и сложные настройки программ, поскольку все уже сделано при создании web-сервиса. Более того, сервис сразу завязан на вычислительные ресурсы, поэтому у пользователей нет и прямой необходимости в поиске требуемых компьютеров. И, наконец, web-сервисы могут напрямую взаимодействовать между собой аналогично вызовам подпрограмм в традиционном программировании, образуя сложные программные комплексы из готовых web-модулей.

Могут ли X-среды задач пересекаться по общим узлам? В принципе, это ничему не противоречит. Вместе с этим, мы предполагаем, что если задача заняла узел, то она использует его постоянно. С этой точки зрения большого смысла в пересечении нет. Если же по каким-то причинам в X-среде есть большие интервалы незанятости

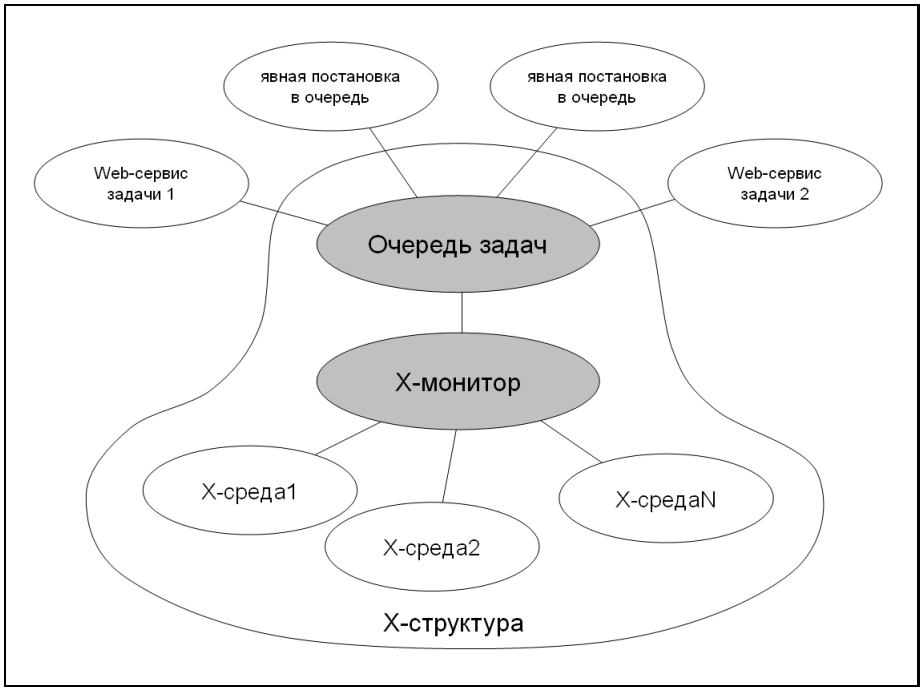

Рис. 5. *X-структура решения задач с явной постановкой задач в очередь и работой через web-сервисы*

узлов, то можно воспользоваться стандартным режимом использования периодов простоя компьютеров для их использования другой средой.

X-структуры имеют много интересных приложений. Например, на их основе можно легко сделать универсальное средство доступа к значительным компьютерным ресурсам для решения вычислительно сложных задач — своего рода распределенный суперкомпьютерный центр коллективного пользования. Эта же идея в масштабах локальной сети предприятия поможет собрать мощную вычислительную систему, использующую периоды незанятости уже имеющихся компьютеров. Это только кажется, что на пяти компьютерах лаборатории ничего серьезного не сделаешь! Оценим реальность. Пусть компьютеры полностью заняты делами лаборатории каждый будний день с 9.00 до 19.00. Это значит, что они свободны с 7 вечера до 9 утра в будни и все выходные дни. 1 год, 365 дней, 52 недели, 5 компьютеров — это 30750 процессорочасов, в течение которых компьютеры простаивают. Накопленный всего лишь на пяти компьютерах за год ресурс эквивалентен использованию 16-процессорного кластера в течение 80 суток! Есть над чем задуматься...

Помимо работы в описанных выше режимах, X-структуры могут быть настроены на несколько дополнительных видов сервиса. Например, показанные на рис. 6 X-структуры **B**, **C** и **D** могут по запросу передавать друг другу часть вычислительных узлов, предоставляя *сервис узлов.* Постоянной приписки к какой-либо структуре у узла нет, он знает лишь адрес сервера, выдающего ему задания, который и нужно временно подменить. С этой точки зрения можно говорить даже о том, что X-структуры **B**, **C** и **D** работают над единым полем узлов, перераспределяя вычислительные ресурсы в зависимости от текущей нагрузки. Аналогично работает

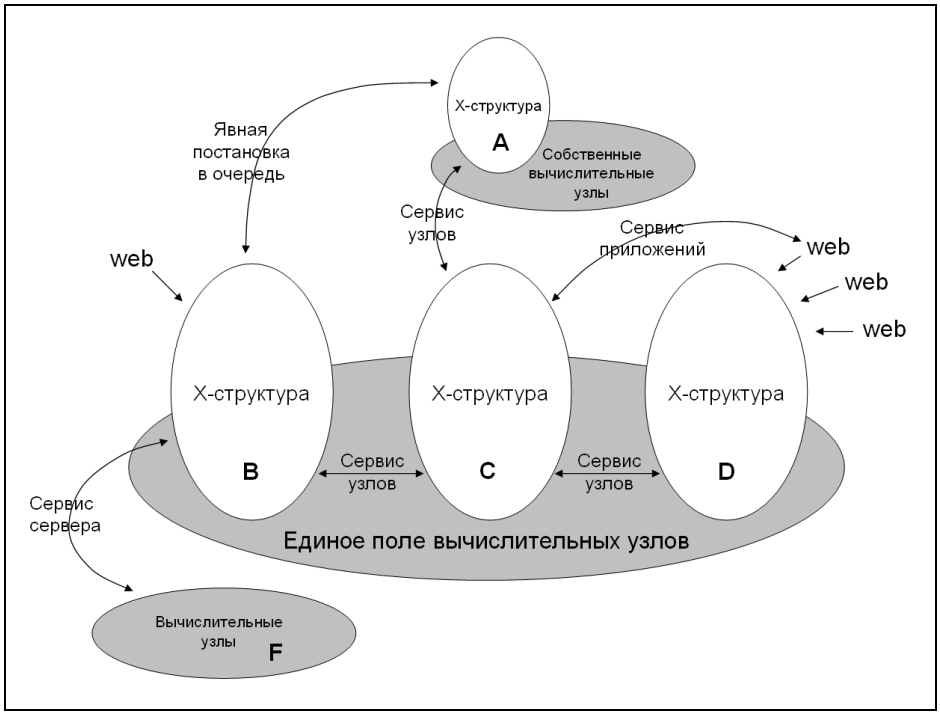

Рис. 6. *Дополнительные сервисы X-структур*

X-структура **A**. В обычном режиме она использует небольшой набор своих узлов, но в случае необходимости обращается за сервисом узлов к X-структуре **C** либо задания от своего имени направляет в очередь X-структуры **B**. Заметим, что, передавая свои узлы, X-структура может предписать режим их использования запрашиваемой стороне: монопольно или же только в периоды незанятости.

Если кому-то нужно решить задачу и у него есть подходящие для этого ресурсы, то не обязательно разворачивать собственную X-структуру целиком, можно воспользоваться *сервисом сервера.* Например, в очередь X-структуры **B** ставится задание и одновременно указывается набор вычислительных узлов **F**, которые должен использовать сервер для решения этой задачи. Все действия, необходимые для порождения и сопровождения работы сервера, выполняет X-структура, а узлы для формирования X-среды предоставляются извне. Это удобно в том случае, когда задача предъявляет к узлам какие-либо специфические требования, которым большинство узлов не удовлетворяет. И, наконец, еще один вид сервиса, который мы уже упоминали ранее — это *сервис приложений.* На том же рисунке X-структура **D** через web-сервис предоставляет доступ к некоторому приложению, чем может воспользоваться, например, X-структура **C**.

8. Заключение

В рамках данной работы предложен подход к формированию распределенных сред из доступных вычислительных ресурсов. Выделены характерные свойства распределенных сред, показана принципиальная возможность решения больших задач в распределенных неоднородных средах, разработан подход к организации вычислительного процесса на подобных компьютерных системах. Предложенные

идеи реализованы в системе X-Com, которая к настоящему времени апробирована в ходе большого числа реальных расчетов.

Обсудив базовые положения задачи формирования вычислительных сред, мы были вынуждены опустить целый ряд интересных деталей. В частности, в своих рассуждениях мы опирались на понятие "надежного" сервера. Как приблизится к такому понятию на практике? Можно возложить всю ответственность на существующие отказоустойчивые кластерные решения высокой степени готовности, а можно подумать о создании системы взаимосвязанных серверов, отслеживающих и страхующих работу друг друга. Мы говорили о важности масштабируемости системы при выполнении действительно больших расчетов, этот аспект требует дополнительного аккуратного исследования. Вопросы обеспечения безопасности в распределенных средах были сознательно оставлены в стороне: очевидные, но не всегда эффективные, решения лежат на поверхности, а комплексный анализ данной проблемы опять-таки выходит за рамки статьи и будет описан отдельно.

СПИСОК ЛИТЕРАТУРЫ

1. *Воеводин Вл.В., Филамофитский М.П.* Суперкомпьютер на выходные //Открытые системы. 2003, N5, С.43-48.

2. *Воеводин Вл.В., Смирнов Ю.Г., Филамофитский М.П., Цупак А.А.* Метод суперкомпьютерных вычислений для решения задачи о возбуждении прямоугольного резонатора // Сб.трудов Всероссийской научно-технической конференции "Параллельные вычисления в задачах математической физики" - Изд-во РГУ, Ростов-на-Дону. 2004 г. - С. 48-56.