УДК 004

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ. ОСОБЕННОСТИ ОРГАНИЗАЦИИ

Пишта Я.В., Замятин Д.С.
Национальный технический университет Украины
кафедра Специализированные компьютерные системы

Аннотация:
Пишта Я.В., Замятин Д.С. Распределенные вычисления. Особенности организации
Рассмотрены способы организации распределенный вычислений. Определены приоритетные направления развития и создания новых продуктов.

Общая постановка проблемы

Современные научные исследования крайне «прожорливо» относятся к вычислительным ресурсам, вынуждая ученых прибегать к использованию дорогостоящих суперкомпьютеров и кластерных сетей. Однако каждый час работы подобных вычислительных систем расписан на многие месяцы вперед. Поэтому многочисленные исследовательские группы чуть ли не дерутся между собой за доступ к этим высокопроизводительным числодробилкам.

Тем временем в мире простаивают без дела технические средства с совокупной вычислительной мощностью, способной заткнуть за пояс все суперкомпьютеры мира. Этот пока малоосвоенный резерв – процессоры и видеокарты наших персональных компьютеров, а также игровые приставки. Владельцу ПК совсем не сложно подключить его к глобальной сети распределенных вычислений (*distributed computing*), которая уже обеспечила десятки научных проектов недостающими вычислительными ресурсами.

Исследование

На домашних и офисных ПК центральные процессоры и видеокарты большую часть времени работают на «холостом» ходу – их загрузка не превышает десятой части их вычислительных способностей. Предоставив пропадающее даром процессорное время на нужды науки, владелец компьютера не чувствует неудобств (кроме разве что шума от вентиляторов из-за повышенного тепловыделения) – вычисления выполняются с минимальным приоритетом, не сказываясь на работе других программ, играх, просмотре фильмов или других развлечениях. Соединение с Интернет необходимо лишь для отправки результатов на сервер и получения новых заданий – расходы на трафик вырастут незначительно на фоне обычных прогулок в сети.

Нередко уступая профессиональным кластерным системам в удельной производительности на единичный процессор, персональные компьютеры, особенно с установленными мощными видеокартами, а также ноутбуки и сервера участников распределенных вычислений берут числом: их армия растет стремительными темпами, тогда как запуск новых суперкомпьютеров – событие редкое, попадающее в выпуски новостей.

Для организации распределенных вычислений (GRID-вычислений) необходима соответствующая программная платформа. Система должна уметь разбивать одну большую задачу на множество маленьких подзадач, распределять эти подзадачи по вычислительным узлам, принимать результаты вычислений и объединять их в единое целое. Для этого были созданы различные программные «прослойки» между управляющим сервером и вычислительными узлами. К сожалению большинство из низ обладает рядом недостатков, связанных с безопасностью компьютера клиента, подключающегося к определенному проекту, передачей результатов вычислений по незащищенным каналам, что “дарит” злоумышленнику возможность на подрыв, вычислений. В следствии этого необходима

разработка платформы, которая позволит усилить меры защиты, для повышения безопасности, а также более легкую разработку и развертывание распределенных вычислений, и что не маловажно – подобная система должна быть независима от платформы, операционной системы клиента.

Но кроме способов организации непосредственно среды, следует уделить внимание способам организации вычислений. Рассмотрим основные типы алгоритмов, которые могут быть использованы в распределенных системах:

1. Централизованный алгоритм. Один из процессов выбирается в качестве координатора. Когда какой-либо процесс хочет войти в критическую секцию, он посылает запрос координатору и ждет ответа. Если в этот момент ни один из процессов не находится в критической секции, то координатор присылает ответ с разрешением. Если же некоторый процесс выполняет критическую секцию, ответ не присылается. Он ставится в очередь и после освобождения критической секции ему присылается ответ-разрешение.

2. Распределенный алгоритм. Если процесс хочет войти в критическую секцию, он формирует сообщение, содержащее имя нужной ему критической секции, номер процесса и текущее значение времени. Это сообщение рассылается всем другим процессам. Возможны три варианта: принимающий процесс находится не в критической секции — отсылается разрешение; процесс находится в критической секции — он не отправляет никакого ответа и ставит запрос в очередь; процесс хочет войти в критическую секцию, но еще не сделал этого — он сравнивает временную отметку поступившего сообщения со значением времени, которое содержится в его собственном запросе, разосланном всем другим процессам. Если время поступившего к нему сообщения меньше, то есть его собственный запрос возник позже, он посылает сообщение-разрешение. В противном случае он не посылает ничего и ставит поступившее сообщение в очередь. Процесс может войти в критическую секцию только в том случае, если он получил ответные сообщения-разрешения от всех остальных процессов. Ели процесс покидает критическую секцию, он посылает разрешение всем процессам из своей очереди и очищает ее.

3. Алгоритм Token Ring. Все процессы в системе образуют логическое кольцо, то есть каждый процесс знает номер своей позиции в кольце, а также номер ближайшего к нему процесса. Когда кольцо инициализируется, процессу под номером 0 передается маркер (token). Маркер циркулирует по кольцу. Он переходит от процесса n к процессу n+1. Когда процесс получает маркер, он анализирует ситуацию, не надо ли ему войти в критическую секцию. Если да, он оставляет маркер у себя и начинает выполнение критического участка программы, после выхода из него он передает маркер дальше по кольцу. Если же процесс не заинтересован во вхождении в критическую секцию, он сразу же передает маркер дальше по кольцу. То есть, если ни один из процессов не заинтересован во вхождении в критическую секцию, маркер просто циркулирует по кольцу.

Выводы

Сравним эти три алгоритма. Централизованный алгоритм является наиболее простым и эффективным. При его использовании требуется только три сообщения. При использовании распределенного алгоритма для выполнения одной критической секции требуется послать (n-1) сообщений-запросов (где n-число процессов), по одному на каждый процесс, и получить (n-1) разрешений, то есть всего необходимо 2(n-1) сообщений. В алгоритме Token Ring число сообщений переменно: от одного, если каждый процесс входит в критическую секцию, до бесконечно большого числа при циркуляции маркера по кольцу. К сожалению, алгоритмы такого типа плохо защищены от отказов.

Литература

1. Сидоров И.А., Опарин Г.А., Феоктистов А.Г. Разработка и применение распределенных пакетов прикладных программ // Программные продукты и системы. 2010. No 2. С. 108-111.
2. Бычков И.В., Опарин Г.А., Новопашин А.П., Феоктистов А.Г., Корсуков А.С. Управление потоками заданий в интегрированной кластерной системе // Параллельные вычисления и задачи управления: Пленарные доклады V Междунар. конф. – М.: Изд-во ИПУ РАН, 2010. С. 39-45.
3. http://www.ibm.com/developerworks/ru/library/l-grid/index.html
4. http://boinc.berkeley.edu