Т-подход к автоматизированному распараллеливанию программ в распределенной вычислительной среде

М.Н.С. КАЗЬМИН О.О.

НАУЧНО ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ МЕХАНИКИ МГУ ИМ. М.В.ЛОМОНОСОВА

4 ФЕВРАЛЯ 2010

Предпосылки создания Т-подхода

В настоящее время самым распространенным средством для создания параллельных программ является стандарт MPI (Message Passing Interface). Однако он обладает рядом недостатков:

- разработка параллельных программ посредством MPI является сложной задачей;
- использование MPI приводит к дублированию кода.

Еще одной причиной возникновения Т-подхода является повсеместное распространение неоднородных распределенных вычислительных установок, построенных по архитектуре GRID.

Особенности GRID-систем

Распределенные неоднородные вычислительные установки, построенные по архитектуре GRID, имеют ряд особенностей, отличающих их от традиционных вычислительных систем. В их числе:

- гетерогенность;
- различная производительность вычислительных узлов, входящих в состав комплекса;
- неоднородность каналов связи;
- объективно более частые отказы вычислительных узлов и каналов связи.

Особенности распараллеливания в распределенной вычислительной среде

При реализации автоматизированного распараллеливания на распределенных системах возникают принципиально новые задачи. Среди них:

- Эффективное распределение нагрузки между узлами различной мощности с учетом ограниченной пропускной способности каналов связи
- Передача данных между узлами с различной аппаратной архитектурой
- Обеспечение отказоустойчивости

Применимость MPI в GRID-системах

Применение MPI накладывает определенные ограничения на способы организации вычислений, среди которых можно выделить следующие:

- порядок, в котором приходят сообщения, и их размеры должны быть заранее известны получателю;
- множество узлов, на котором производятся вычисления, фиксируется при запуске и не может изменяться в процессе вычисления;
- возможность пересылать данные одновременно с вычислением отсутствует или реализуется только для сообщений небольшого размера.

Данные ограничения приводят к значительному снижению эффективности применения MPI на распределенных вычислительных установках.

Ключевая идея Т-подхода

Ключевой идеей Т-подхода является сочетание функционального и императивного подхода к программированию.

Такое сочетание дает, с одной стороны, возможность проведения автоматического анализа и преобразования программ. С другой стороны, оно позволяет добиться высокой производительности, которая недостижима при использовании лишь одних функциональных языков программирования.

Основные положения Т-подхода

1. Параллельные программы разрабатываются на традиционных, широко распространенных языках.
2. Исходные программы должны быть написаны в функциональном стиле.
3. Для описания параллелизма в программу добавляются модификаторы, которые прозрачны для обычного компилятора.
4. Определение того, какие части программы могут работать параллельно, и распределение их между узлами происходит динамически.
5. Ускорение достигается путем параллельного запуска нескольких функций на разных узлах или процессорах.

Ключевые понятия Т-подхода

Т-функцией называется функция, объявленная и реализованная специальным образом, позволяющим запускать ее отложено на другом процессоре или на другом вычислительном узле. Т-функция может передаваться по сети.

Т-переменной называется специальная переменная, которая может содержать либо готовое, либо неготовое значение.

Т-переменная ***готова***, если она содержит готовое значение. В противном случае она называется ***неготовой***.

Пример T-программы

```
tfun double long_func (double x)
{
  … // сложные вычисления
}

tfun double func(double x)
{
  tval double a = long_func(x);
  double b = another_long_func(y);
  // long_func будет параллельно вычисляться
  // на другом узле или процессоре
  c = a + b; // ожидание завершения long_func,
                // если она еще не завершилась
  …
}
```

Планирование параллельных вычислений

Эффективность разрабатываемых с помощью Т-подхода программ напрямую зависит от используемых методов распределения вычислительной нагрузки. Т-система использует два таких метода, которые мы будет называть ***планировщиками***.

Использующиеся планировщики:

1. Балансирующий планировщик.
2. Планировщик Fishing.

Планировщик Fishing

Алгоритм основан на методе заимствования заданий (work stealing).

Обмен данными о состоянии узлов не происходит. Задача пересылается только в ответ на запрос от одного из узлов. Поиск задачи на удаленных узлах производится только в случае отсутствия локальных готовых к исполнению задач.

Планировщик Fishing(2)

На узел, определяемый случайным образом, отправляется запрос **TaskRq**. Если очередь этого узла не пуста, в ответ отправляется сообщение с задачей **Task**. Иначе, **TaskRq** пересылается на другой узел, также определяемый случайным образом.

Если количество пересылок **TaskRq** превысит определенное значение, на исходный узел отправляется сообщение **TaskRqExpired**. В этом случае, исходный узел пропускает некоторое время и вновь отправляет **TaskRq**.

Выбор узла в планировщике Fishing

В некоторых практически важных программах большинство задач создаются на одном узле. В этом случае метод случайного поиска задачи неэффективен, время поиска сильно зависит от числа узлов в вычислительной системе.

Решение: запоминать узел, с которого была получена последняя задача. Этот узел выбирается в качестве первого узла для **TaskRq**. Дальнейшие пересылки **TaskRq** осуществляются случайным образом.

Эта модификация позволяет находить готовую к исполнению задачу в таких программах за одну пересылку **TaskRq**.

Выбор узла в распределенной среде

В распределенной вычислительной среде планировщик Fishing может быть неэффективен из-за различной скорости каналов связи между узлами. Если в качестве узла запроса выбирается узел удаленного кластера, то происходит ожидание доставки пакета по «медленной» сети, хотя возможно есть свободная задача на узле локального кластера.

Решение: ввести дополнительный параметр, определяющий вероятность запроса на узлы локального кластера.

Данное решение позволяет уменьшить вероятность повышенной задержки по причине отправки запроса на удаленный кластер.

Испытания на распределенной неоднородной системе

НИИ механики МГУ, 16 x Athlon MP 1800+
ССКЦ СО РАН, 160 x Itanium2 1.6 Ггц

Планировщик Fishing, ttl = 10, delay = 0.01.

Программа	T_1, с	T_2, с	$T_{теор}$, с	T_{1+2}, с	КЭ, %
fib(42)	130.73	126.28	64.23	67.00	95.9
RT 2048	68.84	60.58	32.22	35.57	90.6

Время исполнения программы в неоднородной системе с использованием 16 + 26 процессоров.

Программа	T_1, с	T_2, с	$T_{теор}$, с	T_{1+2}, с	КЭ, %
fib(42)	130.73	33.51	26.67	31.08	85.8
RT 2048	68.84	18.33	14.48	17.80	81.3

Время исполнения программы в неоднородной системе с использованием 16 + 100 процессоров.

Координация вычислений в слабосвязанных комплексах

Вычисления в слабосвязанных комплексах требуют специальных механизмов передачи данных.

Подходы на основе MPI обладают существенными недостатками.

- Невозможность изменить вычислительное поле в процессе вычислений
- Отказоустойчивость при аварийной остановке ряда процессоров и коммуникаций не обеспечивается.

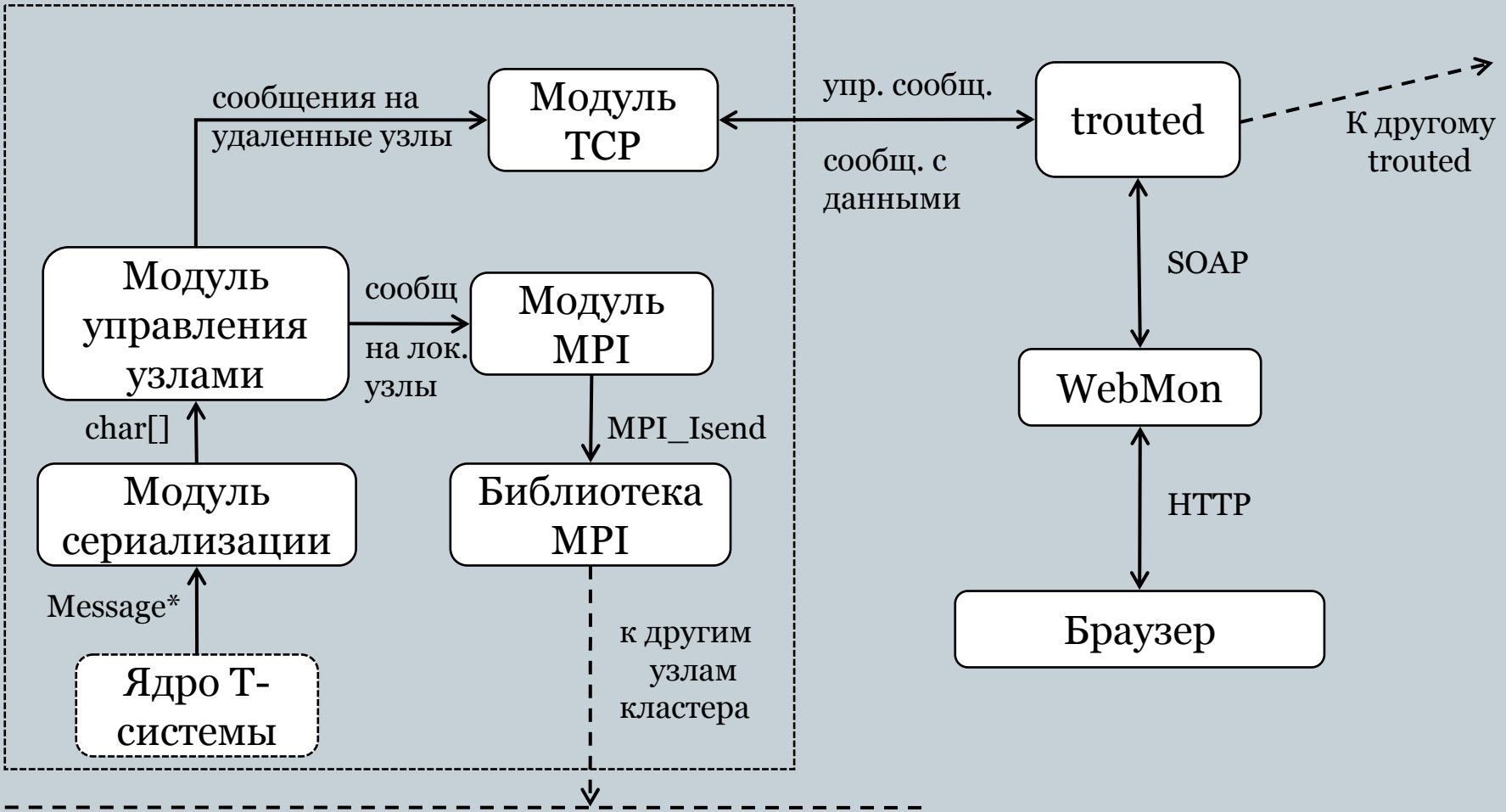
По этим причинам такие подходы не применимы при работе в распределенной среде.

Межкластерный транспорт

Как правило, узлы вычислительных кластеров не имеют прямого доступа в Интернет из соображений безопасности. Поэтому узлы удаленных кластеров не могут связаться друг с другом с использованием TCP-транспорта.

Решение: разработать фоновый процесс (**trouted**), основной задачей которого является маршрутизация сообщений. Он также осуществляет координацию работы нескольких вычислительных кластеров.

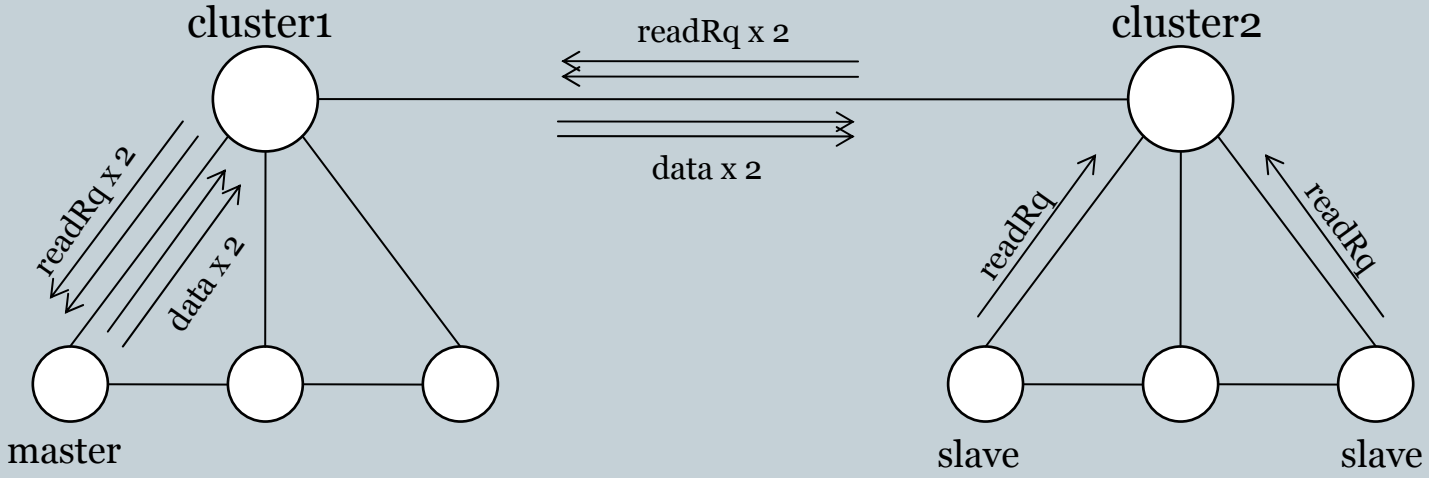
Структура потоков данных при работе в распределенной среде



Передача данных в слабосвязанных системах

При работе на территориально распределенных системах кластеры, как правило, связаны каналом, пропускная способность которого значительно ниже, чем у внутрикластерных каналов.

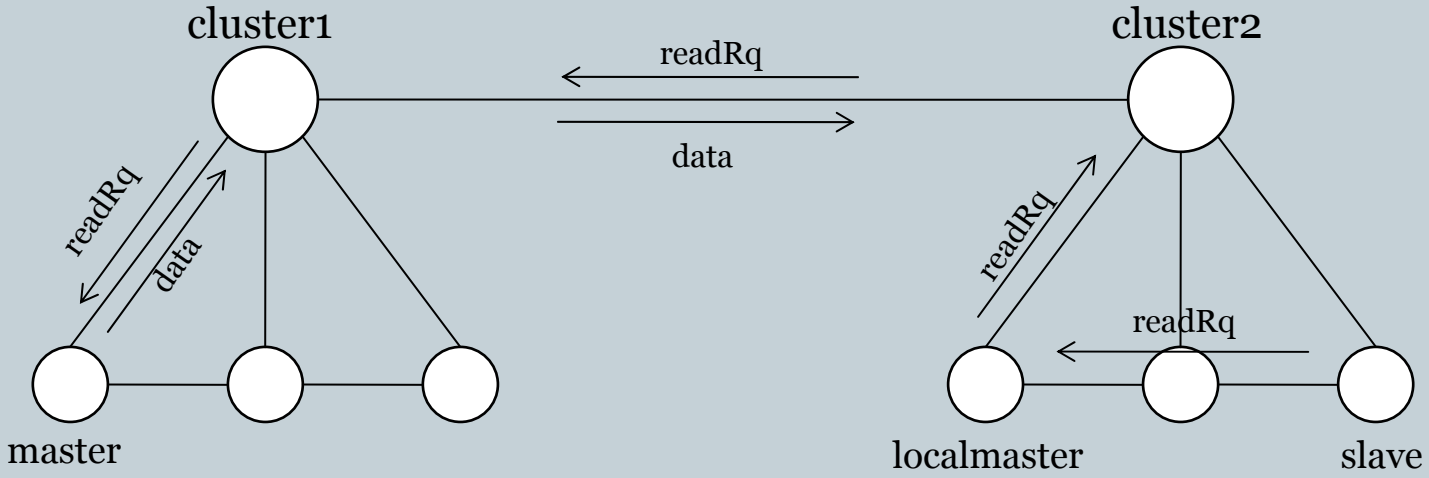
Традиционный подход приводит к многократным передачам одних и тех же данных по медленному межкластерному каналу.



Модифицированная передача данных в слабосвязанных системах

Для уменьшения количества пересылок в Т-системе реализован механизм иерархического кэширования данных.

Дерево строится в зависимости от топологии сети с корнем в узле, на котором хранятся данные. Запрос на получение данных посылается не на узел, хранящий данные, а родителю в иерархическом дереве.



Результаты

Текущие реализации Т-подхода предоставляют средства для автоматизированного динамического распараллеливания программ, в которых решены такие задачи как:

- Динамическая балансировка вычислительной нагрузки
- Возможность изменения вычислительного поля в процессе работы
- Распараллеливание программ на комплексах с различной аппаратной и программной архитектурой
- Возможность работы на сетях с каналами различной пропускной способности

Спасибо за внимание

М.Н.С. КАЗЬМИН О.О.

NUTOK@YANDEX.RU

4 ФЕВРАЛЯ 2010