Р. И. Идрисов

ОБЛАЧНЫЙ СЕРВИС ДЛЯ НАУЧНЫХ ВЫЧИСЛЕНИЙ И ОБРАЗОВАНИЯ [1]

ВВЕДЕНИЕ

На сегодняшний день всё большую популярность набирают облачные сервисы. Конечно, зачастую под активно употребляемым словом «облачный» скрываются обычные вещи, которые просто были названы по-новому. Согласно последней редакции российской Википедии[2], на момент написания статьи облачный сервис – это просто некоторый доступный ресурс в сети, который может быть использован без знания его внутренней структуры.

Статистика поисковых запросов в некотором смысле отражает интересы общества. Если проследить статистику Google[3] по двум терминам – «parallel» и «cloud», можно увидеть, что популярность «cloud» с 2004 года выросла вдвое и продолжает расти, а популярность «parallel», наоборот, неуклонно падает и с 2004 года сократилась вдвое. Рост популярности этого термина обусловлен тем, что облачность представляет пользовательский интерфейс для использования какого-то ресурса. Сам термин «облачный» по одной из версий образовался из того, что Интернет изображался в виде облака: скрытая от пользователя структура, находящаяся в сети и является этим самым облаком. И хотя концепция появилась уже давно, но, как и с функциональными языками программирования, которые были описаны задолго до их популярности, массовое применение облачности стало возможным с развитием соответствующих технологий.

Продолжают набирать популярность облачные хостинги Amazon, cloud9, которые предоставляют вычислительные ресурсы. В частности, они позволяют выполнять код на V8 (JavaScript от Mozilla). Этот язык во многом совместим с браузерным JavaScript, что позволяет создавать переносимый код. Одна и та же программа может быть исполнена как на персональной рабочей станции, так и на потенциально мощном облачном вычислите-

[1] Работа поддержана грантом РФФИ № 12-07-31060 мол_а.

[2] http://ru.wikipedia.org

[3] http://www.google.com/insights/search/

ле. При этом переносимость в данном случае совсем не такая, как в случае Java. Программное обеспечение не требует дополнительной установки библиотек поддержки времени исполнения, а выполняется внутри браузера. Кроме того, это исполнение предполагается безопасным для пользователя. Растущее количество сервисов в сети Интернет, предоставляющих различные услуги, прямо или косвенно связанные с вычислениями, говорит о популярности такого подхода. Возможность в любой момент зайти на соответствующую страницу в сети и выполнить интересующий код, не имея компилятора, очень привлекательна. Подобные сервисы варьируют от совершенно аскетичных, рассчитанных на исключительно короткие программы[4], до предоставляющих среду разработки с группировкой по проектам, подсветкой синтаксиса[5] и т.д.

Несмотря на большое количество таких ресурсов, определённая ниша всё же остаётся незаполненной. В частности, нет таких ресурсов, которые предоставляли бы возможность отладить и запустить научную задачу с массивным параллелизмом.

Следует отметить, что такой подход не является чем-то абсолютно новым, из-за большого количества разных параллельных систем разработчики стараются использовать переносимые варианты, которые могут быть исполнены в различных условиях, в тех случаях, когда речь не идёт о специальном программном обеспечении для конкретного вычислителя.

В этой статье мы опишем систему поддержки параллельного программирования для облачных вычислений, разрабатываемую в ИСИ СО РАН. Дальнейшее изложение организованно следующим образом: в первой главе обсуждается входной язык для системы; во второй главе мы рассмотрим возможности системы с точки зрения пользователя и то, как образовательные задачи могут сочетаться с научными. В заключении будет приведён текущий статус разрабатываемой системы.

ИСПОЛЬЗУЕМЫЙ ЯЗЫК

Разрабатываемая система ставит перед собой две цели: научную и образовательную. С нашей точки зрения, для научной цели более важна масштабируемость, а для образовательной – доступность. Кроме того, было бы неправильно ориентироваться только на один язык программирования, поскольку не существует единого мнения о наилучшем учебном языке. Для

[4] http://codepad.org
[5] http://c9.io – Cloud 9

масштабируемости требуется универсальность описания параллелизма: это значит, что программа не должна быть адаптирована для структуры конкретной вычислительной системы. Согласно работам А. П. Ершова это достигается, если язык программирования приближается к языку описания задач, а не к языку описания алгоритмов. В Институте систем информатики СО РАН мы продолжаем разработку потокового языка программирования Sisal [1] [2], эта работа ставит перед собой именно такие цели. Рассмотрим некоторые возможности Sisal в сравнении с другими языками для параллельного программирования.

Однократное присваивание

Как и многие функциональные языки программирования, Sisal использует однократное присваивание. Этот подход в программировании требует, чтобы каждое значение описывалось в программе только один раз и не изменялось (не присваивалось повторно). Некоторые могут сказать, что любая императивная программа может быть приведена к форме с однократным присваиванием, и что использование подобных ограничений не имеет смысла. Рассмотрим следующий пример на языке Си:

```
int g=0;
void foo(void) { g=1; }
```

В этом случае для приведения программы к форме с однократным присваиванием требуется обозначить новую копию глобальной переменной *g* в момент определения, но этого нельзя сделать внутри функции, а это было бы удобно, поскольку функция может быть не вызвана. Конечно, если записать программу полностью, то такая ситуация будет разрешимой, но для этого потребуется привлечение дополнительных методик анализа и проверок. Идея заключается в том, что программирование в рамках семантики однократного присваивания это нечто вроде исключения оператора goto, которое не только упрощает анализ, но и положительным образом сказывается на описании алгоритма.

Массивы и потоки

Использование этих типов данных не является обычным для функциональных языков программирования, где, как практически всё по возможности выражается в терминах списков и рекурсии. Но массивы существенно упрощают некоторые задачи вычислительного программирования, в частности, с массивами существенно понятней становится описание матрицы. При этом массивы сами по себе не являются императивным или последова-

тельным элементом. Покомпонентное перемножение векторов на языке Sisal можно определить следующим образом:

```
for i in 1, N repeat
 R := A[i] * B[k]
      returns array of R
```

Многословный синтаксис

Это свойство делает программы более читаемыми (проще для человеческого восприятия), и, как результат, долгосрочная поддержка таких программ упрощается. Большинство функциональных языков программирования «страдают» от слишком короткой записи, что, с одной стороны, позволяет описывать алгоритмы более коротко и выразительно, с другой – усложняет понимание этих алгоритмов. Как следствие, гораздо меньше людей в принципе способны на такое программирование. В качестве примера приведём известную программу быстрой сортировки на языке Haskell:

```
qsort :: Ord a => [a] -> [a]
qsort []     = []
qsort (x:xs) =
qsort [y | y <- xs, y < x] ++ [x] ++ qsort [y | y <- xs, y >= x]
```

и для сравнения такая же процедура на языке Sisal:

```
function qsort (Data : array[real] returns array[real] )
if array_size( Data ) > 2 then
let
L, Middle, R := for E in Data
 returns array of E when E < Data[ 1 ]
array of E when E = Data[ 1 ]
array of E when E > Data[ 1 ]
 end for
in
qsort( L ) || Middle || qsort( R )
end let
else
Data
end if
end function
```

Обработка ошибок

На сегодняшний день достаточно популярен механизм try-catch, но он не является естественным для параллельных программ, поскольку требует досрочного завершения определённого участка кода при возникновении ошибки, что приводит к откату части параллельных вычислений, выпол-

няемых в данный момент, либо к потере детерминизма программы. Рассмотрим пример на языке Java:

```
try {
 for (int i=0;i<N;i++) {
     a[i]=a[i]/((i+1)%K);
 }
} catch (Exception e) {
 // display partial results stored in "a"
}
```

В этом случае итерации цикла являются независимыми, что даёт возможность их одновременного исполнения при наличии системы автоматического распараллеливания. Но семантика языка остаётся последовательной, и в момент возникновения ошибки требуется остановить все потоки и создать состояние, соответствующее моменту возникновения первой из ошибок вычислений. При этом первая обработанная ошибка вычислений может не быть первой по семантике языка, поскольку некоторый поток с номером i может получить набор итераций, в которых самая первая ошибочная, а поток i-1 набор, где ошибочная итерация находится ближе к концу. Для сохранения детерминизма и изначальной семантики языка требуется большое количество дополнительной работы системы контроля времени исполнения.

Язык Sisal реализует модель всюду завершаемых частичных вычислений, что означает наличие специального «ошибочного» значения, просачиваемого по графу потока данных. Например, при вычислении матрицы часть элементов в результате может иметь ошибочное значение, что означает ошибку. Таким образом, семантика языка является изначально удобной для параллельных вычислений, и сложного контроля со стороны системы времени исполнения (Run Time System) не требуется.

Поддержка других языков программирования

Включение дополнительного языка не требует больших усилий при наличии реализованного интерпретатора на JavaScript. Так как система ориентирована на функциональные языки программирования, пошаговая отладка не предполагается. Требуется только возможность вычислять выражения без полного описания программы, что упрощает требования к компилятору.

СТРУКТУРА СИСТЕМЫ

С точки зрения надежности распределённые системы являются достаточно привлекательными, поскольку выход из строя или недоступность какой-то из частей далеко не всегда означает полную недоступность системы. Кроме того, не требуется специальной настройки сетевых дисков или постоянного переноса файлов при перемещении пользователя. Хранение предоставляется сервису, причём вместе с версионностью и возможностью отменить изменения, сделанные когда-то. В образовательных системах это открывает дополнительную возможность для преподавателя проследить процесс решения задачи в зависимости от времени. Студенту в этом случае не потребуется создавать версии – система будет автоматически сохранять данные через заданные интервалы времени, что также повышает вероятность быстрого восстановления кода после неожиданного отключения клиента облачного сервиса. Обратим внимание на несколько ключевых особенностей системы:

Иерархия

Распределение ресурсов часто упоминается среди особенностей облачных систем. Как для учебных, так и для научных задач часто требуется ограничивать возможности по использованию ресурсов с целью обеспечения их доступности для остальных. Здесь естественным образом можно использовать иерархию пользователей, которая бы отражала распределение вычислительных ресурсов и давала возможность контроля нижестоящих членов.

Версионность

Современные системы хранения практически немыслимы без версионности, но здесь для образовательных нужд можно дополнительно использовать темпоральную версионность, в которой акцент сделан не на номере версии, а именно на времени, в которое эта версия была реализована. При этом сохранение происходит неявным образом и у преподавателя появляется дополнительная возможность контроля.

Отладка

Кроме отладки при помощи значений, предполагается возможность дополнительной отладки на графах в будущем, но сам по себе интерпретатор

предоставляет возможность отладки только путём вычисления значений. Дополнительные возможности отладки предполагается реализовывать при помощи расширений системы.

ТЕКУЩИЙ СТАТУС И ПЛАНЫ

Структура разрабатываемой системы отражена на рисунке.

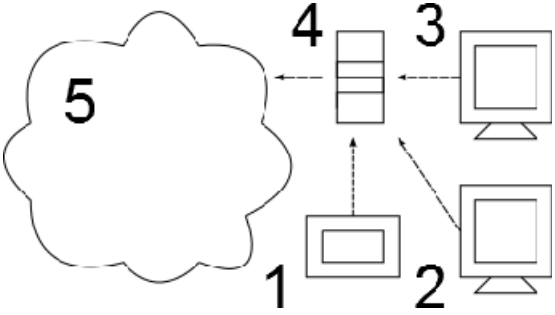

Рис.1. Структура разрабатываемой системы

Цифрами 1, 2 и 3 обозначены устройства, подключающиеся к облачному сервису. На этих устройствах интерпретатор JavaScript, находящийся внутри браузера, используется для реализации среды разработки и отладки программ на задачах меньшей размерности. При отключении от системы (потере интернет-соединения) работоспособность интерпретатора сохраняется.

Цифрой 4 обозначен сервер, предоставляющий доступ к облаку. Этот сервер доступен по фиксированному адресу и может не выполнять каких-либо вычислительных задач. 5 – облачная структура, выполняющая основные вычисления и хранящая пользовательские данные. Под данными предполагаются как программы, так и входная/выходная информация.

На данный момент выбран готовый редактор с подсветкой синтаксиса и открытым кодом. Реализован front-end интерпретатора Sisal на JavaScript, синтаксический анализатор построен при помощи PEG.js. Находится в разработке руководство пользователя языка Sisal. Использование системы

также предполагается в рамках курса по современным методам и языкам программирования, преподаваемого мною в Новосибирском Государственном Университете. Кроме языка Sisal для образовательных целей предполагается использовать Haskell и OCaml.

СПИСОК ЛИТЕРАТУРЫ

1. McGraw J. R. et al. Sisal: Streams and iterations in a single assignment language, Language Reference Manual Version 1.2 // Lawrence Livermore Nat. Lab. Manual M-146 (Rev.1). — Livermore, CA, 1985
2. Касьянов В. Н. Функциональный язык SISAL 3.0 / В.Н. Касьянов, Ю.В. Бирюкова, В.А. Евстигнеев // Поддержка супервычислений и Интернет-ориентированные технологии. — Новосибирск, 2001. — С. 54-67.