ISBN 5-94052-127-4 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2006

УДК 004.272:004.4

А. А. Московский

T-Sim — библиотека для параллельных вычислений на основе подхода T-системы

Аннотация. Основой T-Sim послужили основные идеи концепции Т-системы, среды автоматического динамического распараллеливания вычислений: бесконфликтная модель вычислений на основе чистых (не имеющих побочных эффектов) функций и «неготовых значений» как средства синхронизации доступа к результатам вычислений (при попытке доступа к данным поток-потребитель приостанавливается до тех пор, пока значение не будет вычислено потоком-производителем). Как было впервые продемонстрировано при создании современной реализации Т-системы — Open TS — современный язык C++ позволяет, в принципе, обойтись без введения новых ключевых слов для выражения понятий Т-системы. В статье приведено описание интерфейса и реализации такой библиотеки C++ — T-Sim — «упрощенной» реализации Т-системы.

Ключевые слова и фразы: параллельные вычисления, C++, Т-система.

1. Введение

Развитие информационных технологий все время предоставляет в распоряжение пользователей все новые и новые платформы для параллельных вычислений: вычислительные кластеры, распределенные системы (грид), многоядерные процессоры. Перенос приложений между этими платформами может потребовать существенной переработки приложений (например, при переходе от Posix Threads для многоядерных процессоров к использованию Message Passing Interface для кластера).

Подход к параллельным вычислениям Т-системы позволяет создавать приложения, легко переносимые между различными платформами: адаптация приложения может заключаться в перекомпиляции или же просто в замене настроек среды исполнения программ.

Под «Т-подходом» понимаются следующие основные способы организации параллельного счета:

Работа выполнена при поддержке программы фундаментальных исследований Президиума РАН «Разработка фундаментальных основ создания научной распределенной информационно-вычислительной среды на основе технологий GRID», а также гранта РФФИ № 05–07–08005 _офи_а.

- Т-функции: в качестве гранул параллелизма используются «чистые» функции — функции без побочных эффектов. Поскольку результат вычисления такой функции зависит только от аргументов, вычисления можно осуществить в отдельном потоке или передать для расчета на другой узел кластера — результат от этого не изменится. Таким образом, Т-функции особым образом выделяются в программе, а вызов Т-функции может (но не обязан) породить отдельный поток вычислений или удаленный вызов.
- «Неготовые» значения для доступа к данным. При попытке доступа к данным,, находящимся в «неготовой переменной» из потока потребителя, проверяется, вычислены ли уже эти данные потоком-производителем. Если да, то поток-потребитель продолжает работу, если нет, то работа потока-потребителя приостанавливается до тех пор, пока какой-либо поток не вычислит данные для неготовой переменной.

В настоящее время, последней реализацией Т-системы является Open TS — «Т-система с открытой архитектурой» [1]. В Open TS для написания параллельных программ используется расширение языка C++ — язык T++. В данной работе приводится описание C++ библиотеки T-Sim — «упрощенной» реализацией подхода Т-системы (T-Simplified, сокращенно, T-Sim). Мотивацией при создании T-Sim послужили как желание исследовать возможности C++ для реализации Т-подхода, такие, как списки типов [5], так и иметь полигон для испытаний новых способов выравнивания нагрузки в распределенных вычислительных средах.

2. Сравнение с аналогами

К сожалению, подробный обзор систем и сред параллельного программирования далеко выходит за рамки данной статьи. Например, список литературы обзора [2], написанного в 1999 году, насчитывает более 60 статей. Отметим лишь, что подходы, близкие к подходу Т-системы относятся к крупноблочным системам потоков данных (coarse-grain dataflow), такие как Mentat Programming Language [3]. Существенным отличием Т-подхода от классических систем потоков данных (dataflow) является использование неготовых переменных, что делает концепцию Т-системы близкой ко многим языкам параллельного программирования, основанным на функциональных языках, например Multilisp [4]. В то же время, Open TS использует

язык C++ в качестве базового, что позволяет осуществлять низкоуровневую оптимизацию программ, в отличие от ситуации, в которой бы использовался какой-либо язык функционального программирования.

3. Пользовательский интерфейс

Библиотека T-Sim состоит из набора классов-шаблонов и статически связываемой библиотеки C++. Основным средством синхронизации вычислений и обмена данными является «неготовое значение». Поток исполнения не прерывается до тех пор, пока не запросит данные переменной, которая пока неготова. Неготовую переменную можно объявить с помощью класса-шаблона `TVal<>`

```
TVal<type> x;
```

Где «`type` »—это «базовый тип» для T-величины. Для непосредственного доступа к данным, используется перегруженный оператор преобразования типов `type&`.

```
int x;
TVal<int> tx;
x = tx;
```

Если `tx` до сих пор не получил значения в момент вызова оператора преобразования типов, выполнение вызывающего потока будет приостановлено до тех пор, пока какой-либо другой поток не вычислит значение. Для присваивания значения, в шаблоне TVal определен оператор присваивания в «базовому типу»

```
tx = y;
```

T-величина может получить значение только один раз, присваивание значения T-величине немедленно позволяет продолжить вычисления всем потокам, ожидающим данной величины. Для высвобождения памяти, пользователь должен явно вызвать соответствующий метод шаблона

```
tx.release();
```

После того, как вызван метод `release()` никакой доступ к переменной оказывается невозможен. Перенос управления памятью на плечи пользователя позволил не только упростить архитектуру системы, но также дает пользователю возможность выбрать наилучший момент для операций освобождения памяти для обеспечения максимальной производительности приложения.

Помимо `TVal` в T-Sim реализованы буфера пользовательских данных, с той же семантикой значений

```
TBuf <int>  bx;
```

Для присваивания значений в `TBuf` используется метод `assign`

```
int *px = new int[100];
... //работа с px
bx.assign(px,100) ;
```

Для доступа к данным используется перегруженный оператор преобразования типов `type` . Аналогично `TVal`, `type` приостанавливает работу потока если данные буфера еще не готовы. Для высвобождения памяти необходимо вызвать метод `release`.

Другая т-сущность в T-Sim—это «T-ссылка». Ссылка может ссылаться либо на буфер, либо на переменную.

```
TRef<type>  rx;
```

Для доступа к данным используется оператор преобразования типов `type*` . Для `TRef` также определен метод `release`—для высвобождения памяти объекта, на который ссылается `TRef`. Главное различие между `TRef` и `TVal/TBuf` заключается в том, что T-ссылки могут лишь ссылаться на объекты, имеющие данные, но не иметь собственные данные. В отличие от Open TS, можно определять статические и глобальные объекты `TBuf, TVal` и `TRef` в случае необходимости.

Гранулой параллелизма в T-Sim служат функции, связанные с из аргументами—«замыкания» в теминологии LISP и «связывания» в C++. Пользователь может создать объект-связывание из любой C-функции, которая не возвращает результат (имеет тип возвращаемого результата «void»).

```
TFUNDEF_2(fib,int,TVal<int>,TFib);
```

После такой декларации, вызов T-функции осуществляется теперь следующим образом:

```
TFib(10,res);}
```

Данная конструкция создает и помещения в очередь заданий вызов функции fib. Поток-планировщик T-Sim получит из очереди объект и примет решение, следует ли запустить функцию на исполнение на локальном узле, или следует передать ее на другой узел для выполнения. При запуске T-функции, для ее выполнения будет создан отдельный поток (нить, тред)—T-Sim использует для этих целей ставшую компонентой glibc библиотеку NPTL, позволяющей создавать до нескольких сотен и даже тысяч потоков.

Как и в T-системе, пользователь должен иметь ввиду, что в качестве аргументов T-функций не могут выступать C++-ссылки или объекты, содержащие в себе ссылки, так как адресное пространство параллельных процессов различно.

Для запуска параллельного счета, пользователь должен создать объект типа TSimRuntime. Конструктор объектов данного класса имеет необязательный аргумент — ссылку на класс, инкапсулирующий стратегию планирования вычислений (планировщик).

По умолчанию, все планировщики в ходе создания считывают список доступных узлов из конфигурационного файла «tsim.cfg» где перечисляются все узлы (с номерами портов для коммуникации), на которых будут запущены параллельные процессы. До узлов перечисляются опции среды исполнения

```
TRANSPORT = XMLRPC
SYNC_MODE = F
...
Node1 30099
Node2 30099
Node3 30099
...
```

«Мастер-хостом» считается первый узел из этого списка,
на остальных компьютерах создание объекта TSimRuntime не возвращает управление в вызвавшую функцию (хотя до этого момента было несколько параллельных потоком управления). Это может быть удобно, если приложению требуется прочесть конфигурационный файл до запуска параллельного счета и/или инициализировать глобальные объекты.

Компиляция программы может быть осуществлена практически любым современным компилятором C++ (проводилось тестирование с компиляторами gcc версий 3.2.x, 3.3.x, 3.4.1). Для запуска программы на выполнение можно использовать скрипт run_tsim.sh, использующий tsim.cfg для запуска программ.

Пример программы параллельного вычисления чисел Фибоначчи приведен внизу

```
#include <iostream>

#include "tsim.h"

using namespace std;
```



```
typedef TVal<int> TInt ;

TFUNDEF_2(fib,int,Tint,TFib);

void fib(int in,TInt __out)
{
  int out;
  if (in < 2) {
      __out = in;
      return;
  }

  TInt o1;
  TFib(in-1,o1);
  TInt o2;
  TFib(in-2,o2);
  out = o1+o2;

  o1.release();
  o2.release();

  __out = out;

  return;
}

int main (int argc,char *argv[])
{
  int t,res;
  TSimRuntime rt;

  if (argc < 2) t = 10;
  else t = atoi(argv[1]);

  TInt _res;
  fib(t,_res);
  res = _res;
  _res.release();
cout << "The FIB " << t << "th is " << res << endl;
  return 0;
}
```

В T-Sim можно указать стратегию, которую необходимо использовать при выравнивании нагрузки для того или иного типа функции. Для этих целей используется макроопределение

```
REGISTER_TFUN_SCHED(test_fun,RoundRobin)
```

Программист может указать одну из следующих стратегий, определяющих распределение T-функций между узлами в вычислительном кластере:

- барабанная (round-robin);
- барабанная для рекурсивных функций, с локальным выполнением T-функций после достижения пороговой глубины рекурсии;
- по наименьшей загруженности;
- по наибольшей мощности процессора;
- по расположению входных данных для функции.

Возможно также определить класс-стратегию в коде приложения, например

```
REGISTER_TFUN_SCHED(time_threshold,RoundRobin)
...
template <class TaskType>
class PairMeasure: public SchedTraitBase {
 public:
    PairMeasure(TSimSchedulerBase *pS):SchedTraitBase(pS) {
      if (pS->neighbours.size()!=2) {
        cerr<< "ERROR: It is possible to run a test on two nodes ONLY\n";
        exit(1);
      }
    };

    Dest recommend(TaskBase *t) {
      if (!pS->isMaster) return Dest::GetMyAddr();
      return  pS->neighbours[1];
    }
};
REGISTER_TFUN_SCHED(sender,PairMeasure);
...
typedef TTraitScheduler< TResBasedScheduler<TSimResDB> > MeasureSched;

int main (int argc , char *argv[]) {
    TSimRuntime rt (new MeasureSched);
...
}
```

В приведенном примере, определен специальный планировщик, позволящий для функции `time_threshold` применять «барабанную» стратегию, а для функции `sender` — специальную стратегию, переносящую все вычисления `sender` на второй узел кластера.

4. Сравнение с T-Sim и Open TS

Упрощенная версия T-системы создавалась, сначала для моделирования стратегий T-системы по управлению ресурсами параллельной вычислительной установки. T-Sim совместим с Open TS в ключевом аспекте — принципе распараллеливания вычислений, однако существенно отличается с точки зрения пользователя. Например, на момент написания статьи единственным средством обмена данными в T-Sim является протокол XML-RPC, в то время как в Open TS поддерживаются средства высокопроизводительного обмена данными на основе MPI. Сравнительный анализ двух систем приводится в таблице 1 на странице 191.

К преимуществам Open TS относятся, в первую очередь

(1) использование легковесных потоков для распараллеливания вычислений,
(2) поддержка MPI в качестве средства обмена данными,
(3) автоматическое высвобождение памяти неиспользуемых значений на основе механизма подсчета ссылок.

В то же время, к преимуществам T-Sim относятся:

(1) сравнительно легкая переносимость, так, например, T-Sim не нужен компилятор языка, требуется лишь компилятор C++,
(2) механизм сериализации объектов, ориентированный на гетерогенное окружение, использование платформо-независимого представления данных в XML,
(3) наличие возможности быстро конструировать стратегии выравнивания нагрузки, повторно использовать уже существующие.

Отдельного рассмотрения заслуживает различие в семантике присваивания значений неготовым переменным. В T-Sim значение переменной можно присвоить только один раз, например,

```
int value=5;
TVal<int> x;
x=value;
```

Повторное присваивание `x=value` вызовет аварийное завершение программы.

В Open TS поддерживается механизм многократного присваивания

Свойства	Open TS	T-Sim
Язык	T++ — расширение C++, есть компилятор и конвертер	C++ — статическая библиотека
Передача данных	MPI (множество реализаций)	XML-RPC, сериализация сложных объектов
Механизм синхронизации	Неготовые переменные с семантикой многократного присваивания	Неготовые величины с однократным присваиванием
Гранулы параллелизма	T-функции — легковесные, невытесняющие потоки	C++ класс «замыкания», запускается в отдельном потоке ОС, на основе библиотеки NPTL
Средства обмена данными	Объектно-Ориентированная общая память	Простейшая реализация Объектно-Ориентированной общей памяти
Сбор информации о ресурсах	Встроенный	Настраиваемый, несколько стратегий
Управления памятью	Распределенный подсчет ссылок	На уровне пользователя
Управление вычислениями	Планировщик вычислений T-системы	Настраиваемая, «конструктор» планировщиков

Таблица 1. Сравнение свойств T-Sim и Open TS


```
int value=5,value2=6;
tval int x;
x=value;
x=value2;// разрешено
```

Однако это приводит к необходимости сформулировать и реализовывать довольно сложные правила, описывающие доступность значений потребителям. Как показывает опыт разработки программ на Open TS, прикладному программисту все-таки приходится изучать эти правила, например, при оптимизации программ. Так, по умолчанию, результат присваивания значения переменной-результату функции становится доступен потребителям не сразу, а только после завершения функции-производителя значения:

```
tfun int producer (int tout res) {
...
res=1;
// потоки-потребители res все еще не могут продолжить
// вычисления - они либо будут дожидаться окончания функции,
// либо необходимо явно вызвать tdrop(res);
...
return 0;
}
```

5. Заключение

Данная статья была посвящена, в первую очередь, описанию интерфейса библиотеки T-Sim. Многие важные аспекты, в первую очередь, относящиеся к производительности создаваемых с использованием библиотеки программ будут затронуты в последующих публикациях. В настоящее время, для библиотеки создано несколько десятков тестов, включающих в себя как тесты на производительность, так и модульные тесты функциональности. Дальнейшие планы развития библиотеки включает в себя, в первую очередь, обеспечение работы в распределенных системах, включающих в себя несколько кластеров, а также обеспечение эффективной работы приложений молекулярной динамики.

Список литературы

[1] Abramov S., Adamovich A. I., Inyukhin A., Moskovsky A., Roganov V., Shevchuk E., Shevchuk Yu., Vodomerov A. *OpenTS: An Outline of Dynamic Parallelization Approach.* // PaCT, 2005, с. 303–312.

[2] Talia D. *Advances in Programming Languages for Parallel Computing* // Annual Review of Scalable Computing, 2000, с. 28–58.

[3] Grimshaw A. S., Liu J. W.–S. *Mentat: An Object-Oriented Macro Data Flow System* // OOPSLA, 1987, с. 35–47.

[4] Halstead R. H. J. *Implementation of Multilisp: Lisp on a Multiprocessor* // LISP and Functional Programming, 1984, с. 9–17.

[5] Александреску А. Современное проектирование на C++. — Москва: Вильямс, 2002, с. 336.

Исследовательский центр многопроцессорных систем ИПС РАН

A. A. Moskovsky. *T-Sim — a library for parallel computing based on the T-system approach.* (in Russian.)

Abstract. The T-Sim is based on the cornerstone ideas of the T-system, a system for automatic, dynamic parallelization: conflict-free computational model utilizing pure (no side effects) functions as grains of parallelism and "'non-ready"' values for synchronizing access to computation results (consumer thread is suspended if attempts to access a value, and remains suspended until some producer threads computes the value). It was shown, during implementation of the most recent T-system version — Open TS — that modern C++ is sufficient to express the basic notions of the T-system, and language extension with extra keywords is, in principle, not required. The paper describes an interface and implementation of C++ static library — T-Sim — a "'simplified"' T-system.