УДК 004.272:004.4

ПРИМЕНЕНИЕ ПРЕДМЕТНЫХ ЯЗЫКОВ И АКТОРНОЙ МОДЕЛИ ДЛЯ АВТОМАТИЗАЦИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

© 2016 С.В. Востокин, Е.Г. Скорюпина, Д.М. Наширванов

Самарский национальный исследовательский университет имени академика С.П. Королёва

Статья поступила в редакцию 11.11.2016

В данной работе рассматривается сочетание предметных языков и языков общего назначения для автоматизации высокопроизводительных научных вычислений на базе облачного сервиса Templet Web. Автоматизация программирования при составлении задачи в системе Templet Web основана на концепциях скелетного программирования, предметных языков и использовании специализированных визуальных редакторов предметных языков в составе интегрированной среды разработки. Для предметного языка, использующего акторную модель, был разработан пример тестовой программы. Пример подобран таким образом, чтобы быть достаточно простым (проверка выполнения тригонометрического тождества для заданного значения) и в тоже время использовать все возможности предметного языка. Высокоуровневое описание задачи на предметном языке преобразуется в код, который использует систему времени выполнения. Для рассматриваемой системы времени выполнения, основанной на акторной модели, приводится заключение о степени автоматизации предлагаемого метода программирования на основании анализа относительных размеров библиотечного, сгенерированного и рукописного кода. В качестве критерия сложности и степени автоматизации рассматриваются относительные размеры кода разных частей программы. Код примера размещен на сервисе GitHub по адресу: https://github.com/templet-language/newtemplet. Высокая степень автоматизации достигается благодаря простоте используемой модели акторов, а также способу сопряжения библиотеки времени выполнения с программой пользователя: код сопряжения строится автоматически по описанию на предметном языке. В текущей версии системы Templet Web, развернутой по адресу http://templet.ssau.ru/app на базе вычислительного кластера «Сергей Королёв» Самарского университета, в настоящее время поддерживаются предметный язык, использующий акторную модель вычислений Templet, и скелеты программ «портфель задач», «конвейер». Данная технология показала свою эффективность для решения прикладных задач анализа многомерных динамических систем и процессов при исследовании гироскопических систем ориентации космических аппаратов, а также при обучении студентов принципам работы на суперкомпьютере.

Ключевые слова: параллельное программирование, автоматизация программирования, акторная модель вычислений, предметно-ориентированный язык, скелетное программирование

Работа выполнена при государственной поддержке Министерства образования и науки РФ в рамках реализации мероприятий Программы повышения конкурентоспособности Самарского национального исследовательского университета имени академика С.П. Королева среди ведущих мировых научно-образовательных центров на 2013-2020 годы. Работа частично поддержана грантом РФФИ № 15-08-05934 А.

ВВЕДЕНИЕ

Предметные языки (domain-specific language, DSL) широко применяются для автоматизации научных расчетов. Наиболее известные из них: MATLAB (векторные вычисления), Wolfram Language (комбинация символьной и численной математики), Maple (символьные вычисления), Mathcad (интерактивные вычисления, визуализация). Преимущество данных языков в предметной области численного моделирования состоит в том, что их синтout of boxаксис и семантика адаптированы к традиционной математической нотации, что существенно снижает трудоемкость программирования.

В области высокопроизводительных научных вычислений, где требуется оптимизация кода, предметные языки уступают языкам общего назначения высокого уровня C/C++. Поэтому критичные к производительности части кода перечисленных выше предметных языков реализуются на языках C/C++.

Для ускоренной разработки программного обеспечения были предложены методики, позволяющие сочетать предметные языки и языки общего назначения в одном проекте [1, 2]. В данной работе рассматривается применение этого подхо-

Востокин Сергей Владимирович, доктор технических наук, профессор кафедры информационных систем и технологий. E-mail: easts@mail.ru
Скорюпина Екатерина Григорьевна, магистр. E-mail: kat.skorupina@yandex.ru
Наширванов Дамир Маратович, магистр. E-mail: damir.nashirvanov@gmail.com

да для автоматизации высокопроизводительных научных вычислений на базе облачного сервиса Templet Web [3]. Предварительные результаты данного исследования были представлены на конференции ПИТ-2016 в работе [4].

Высокоуровневое описание задачи на предметном языке в конечном итоге преобразуется в код, использующий систему времени выполнения, реализующую параллельные вычисления. В основе модели параллельных вычислений, принятой во многих языках программирования, лежат потоки. Помимо потоковой модели существуют другие подходы к параллелизму. Одним из таких подходов, который приобретает все большую популярность среди разработчиков, является акторная модель.

В основе акторной модели вместо параллельных потоков, взаимодействующих при помощи общей памяти и блокировок, лежат так называемые «акторы», которые обмениваются асинхронными сообщениями при помощи специальных почтовых ящиков. В ответ на полученные сообщения, актор может принимать локальные решения, создавать акторов, посылать сообщения, а также устанавливать, как следует реагировать на последующие сообщения [5].

В работе рассматривается основанная на акторной модели система времени выполнения, и проводится исследование степени автоматизации предлагаемого метода программирования на основании анализа относительных размеров библиотечного, сгенерированного и рукописного кода. Анализ реализации акторной модели программирования на платформе Java, а также с использованием технологии MPI, был впервые представлен в работах авторов на конференции ПИТ-2016 [6, 7].

ОСОБЕННОСТИ ПРИМЕНЕНИЯ ПРЕДМЕТНОГО ЯЗЫКА

Автоматизация программирования при составлении задачи в системе Templet Web основана на концепциях скелетного программирования [8, 9], предметных языков и использовании специализированных визуальных редакторов предметных языков в составе интегрированной среды разработки (IDE).

В системе Templet Web используются несколько предметных языков для решения перечисленных ниже задач автоматизации научных вычислений: организации вычислений в парадигме «портфель задач» [10]; применения акторной модели вычислений, реализованной в общей и распределенной памяти [11]; ввода данных; вывода данных; вызова функций стандартных математических библиотек; описания серий экспериментов.

Для пояснения назначения предметных языков в системе Templet Web ниже приведен скелет программы параллельного умножения матриц в распределенной и разделяемой памяти.

Под скелетом программы понимается ее схематичное описание, по форме напоминающее псевдокод. В отличие от псевдокода, код скелета пишется на языке программирования высокого уровня (в данной системе – на C/C++) и компилируется. Однако, код скелета не компонуется и не исполняется потому, что в нем не содержится необходимая компоновщику информация. Этим скелет отличается от обычной программы.

Роль предметного языка в заключается в передаче той информации, которая отсутствует в скелете. То есть предметные языки в исследуемой технологии автоматизации необходимы, чтобы доопределить объекты в скелете программы. Передаваемая на предметном языке информация используется для того, чтобы выполнить автоматическое преобразование скелета в исполняемую программу.

```
const int N=10;
double a[N][N],b[N][N],c[N][N];

struct Result{
 void save(ostream&s){ s<<num;
    for(int j=0;j<N;j++)s<<c[num][j]; }
 void rest(istream&s){ s>>num;
    for(int j=0;j<N;j++)s>>c[num][j]; }
  int num;// номер вычисленной строки
};
struct Task{
  void save(ostream&s){ s<<num; }
  void rest(istream&s){ s>>num; }
  int num;// номер вычисляемой строки
};
void Proc(Task&t,Result&r){
  int i=t.num;// параллельное вычисление
строки матрицы C
  for(int j=0;j<N;j++){
    c[i][j]=0.0;
      for(int  k=0;k<N;k++)c[i][j]+=a[i]
[k]*b[k][j];
  }
  r.num=i;
}
struct Bag{
  Bag(int argc, char* argv[]);
  void run();
  bool isTask(){return cur<N;}
  void put(Result&){ }
  void get(Task&t){t.num=cur++;}
 int cur;//номер текущей строки в матрице C
};
int main(int argc, char* argv[])
{
  Bag bag(argc,argv);
  // инициализация
  input(a,b);
  bag.cur=0;
  // параллельное умножение матриц
  bag.run();
  // вывод результата параллельного ум-
ножения
  output(c);
```

```
    start_time();
     strassen(a,b,c); // теперь умножаем
методом Штрассена
    end_time();
    return 0;
  }
```

В приведенном примере структуры Result, Task, Bag и функция Proc являются составными частями реализации метода организации вычислений «портфель задач». Это указано в аннотации на предметном языке, которая сопровождает приведенный код скелета программы умножения матриц (форма описания скелетов на DSL в данной статье не рассматривается).

Описание скелета программы содержит способ именования составных элементов метода «портфель задач», способ реализации в общей или распределенной памяти, количество используемых узлов и/или потоков выполнения и другие детали.

Скелет программы также содержит неопределенные процедуры input, output для ввода и вывода данных. Аннотация на предметном языке описывает привязку этих процедур к имеющимся в системе Templet Web пользовательским интерфейсам ввода-вывода данных.

В коде скелета программы имеется неопределенная процедура strassen. Это ссылка на реализацию соответствующего алгоритма умножения матриц Штрассена. Пользователь на предметном языке в визуальном редакторе может выбрать конкретную библиотечную реализацию алгоритма. При этом ему не требуется знать сигнатуру реальной функции в библиотеке и способ связи с этой библиотекой. Аналогично можно сделать привязку к другим библиотечным процедурам. Например, в задаче анализа хаотического движения гироскопов [12], решаемой с использованием описываемого метода, используются разные варианты численного интегрирования пользовательских функций.

Предметный язык описания серий экспериментов в системе Templet Web предназначен для автоматического построения кода серии экспериментов на основе программы для единичного эксперимента. Например, в коде скелета программы умножения матриц параметр N может изменяться в пределах, описанных на DSL. Также можно автоматически выполнять статистическую обработку результатов при замере производительности. Например, пара функций start_time(), end_time() может обозначать секцию кода с замером времени. Полученное в серии время можно автоматически усреднить, определить минимум и максимум, СКО, построить гистограмму функции распределения вероятности времени исполнения.

ОЦЕНКА СЛОЖНОСТИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

Для предметного языка, использующего акторную модель, был разработан пример тестовой программы. Пример подобран таким образом, чтобы быть достаточно простым (проверка выполнения тригонометрического тождества для заданного значения) и в тоже время использовать все возможности предметного языка. В качестве критерия сложности и степени автоматизации рассматриваются относительные размеры кода разных частей программы. Код примера размещен на сервисе GitHub по адресу: https://github.com/templet-language/newtemplet.

Содержательная часть примера состоит в следующем. Имеется три актора. Управляющий актор получает некоторое число x и передает его в двух сообщениях двум другим «рабочим» акторам. Первый из рабочих акторов вычисляет квадрат синуса полученного в сообщении числа, второй – квадрат косинуса. Вычисленные значения передаются управляющему актору. Получив ответы от двух рабочих акторов, управляющий актор складывает содержащиеся в ответных сообщениях числа. Таким образом, признаком корректного завершения алгоритма является значение 1, полученное в результате сложения управляющим актором значений $\sin^2 x$ и $\cos^2 x$.

На предметном языке в данном случае аннотированы следующие классы: my_engine – класс системы времени выполнения; value_message – класс сообщения, содержащий вещественное число и тип сообщения; master – класс управляющего актора; worker – класс для двух рабочих акторов. Предметный язык предписывает, какие интерфейсы реализованы в перечисленных классах. Основные используемые в примере интерфейсы: интерфейс сохранения-восстановления состояния актора и сообщения, интерфейс для задания положения актора на вычислительном узле (необходимы в распределенной реализации), интерфейс для обработки поступающих сообщений от конкретного актора (2 интерфейса у управляющего и по 1 интерфейсу у рабочих акторов), интерфейсы для связывания акторов между собой для обмена сообщениями.

Код, реализующий пример, содержится в трех файлах. Файл lib/templet.hpp содержит систему времени выполнения для акторной модели вычислений. В примере имеются пять вариантов выполнения. Отладочное выполнение позволяет отладить логику работы программы на последовательной ЭВМ. В нем имитируются различные возможные последовательности отправки и доставки сообщений с использованием генератора случайных чисел. Также тестируется корректность сохранения-восстановления состояния акторов и сообщений.

Последовательное выполнение оптимизировано для вычислений на одном процессоре. Оно реализовано аналогично отладочному, за исключением кода для сохранения-восстановления и случайного выполнения программы.

Параллельное выполнение в общей памяти реализовано с использованием техники потокового пула и использует стандартные потоки версии ISO/IEC 14882:2011 языка C++. В работе используется пул фиксированного размера, количество потоков пула задается, исходя из аппаратных возможностей конкретной системы. В данной реализации произвольный актор может обслуживаться произвольным потоком из пула и наоборот.

Акторные библиотеки времени выполнения широко используются также на платформе Java. Функциональность рассматриваемой в работе акторной модели можно реализовать с использованием стандартной модели потоков Java, основанной на классе Thread. Дальнейшее исследование предполагает проверку трудоемкости и сравнительной эффективности акторного кода при исполнении нагрузочных тестов по сравнению с библиотеками Kilim [13, 14], Actors Guild [15], ActorFoundry [16], Akka [17]. В качестве технологии преобразования API потоков в акторную модель будет рассмотрена технология, применяемая в библиотеке templet.hpp. Вопросы эффективности параллельной реализации вычислений в данной работе не рассматриваются.

Библиотека templet.hpp также реализует дискретно событийное имитационное выполнение вычислений. Этот метод позволяет измерить потенциал распараллеливания программы на идеальном вычислителе, обладающим произвольным числом процессоров (т. н. паракомпьютере). В результате имитационного выполнение вычисляются: максимальное число процессоров, которое может задействовать исследуемая программа; ускорение на паракомпьютере; ускорение на системе с заданным числом процессоров, оцененное с использованием закона Амдала.

Распределенная реализация акторной модели в библиотеке templet.hpp основана на разделении вычислений в программе на следующие этапы: (1) создание акторов в виде объектов/структур языка C++; (2) построение коммуникационной топологии путем связывания акторов каналами передачи сообщений; (3) определение привязки конкретных акторов к процессам MPI; (4) ввод исходных данных в программу; (5) выполнение вычислений; (6) вывод/сохранение результатов работы.

Этапы (4) и (6) выполняются исключительно на мастер-процессе MPI, этап выполнения вычислений (5) обычно выполняется на рабочих процессах, но так же можно использовать и мастер-процесс. Остальные этапы протекают для всех процессов. Алгоритм управления вычислениями основан на технике потокового пула в разделяемой памяти. Он предполагает, что акторный алгоритм является статическим в том смысле, что количество акторов и сообщений известно и не изменяется во время вычислений.

В табл. 1 представлены размеры частей кода библиотеки времени выполнения для рассматриваемого примера. Можно заметить крайне малый абсолютный и относительный размер кода интерфейса (759 байт, 4,5 %), связывающего библиотеку с остальным кодом. Это достигается благодаря простоте используемой модели акторов, а также способу сопряжения библиотеки с программой пользователя: код сопряжения строится автоматически по описанию на предметном

Таблица 1. Размеры частей кода акторной системы поддержки времени выполнения предметного языка templet.hpp

Часть кода	Размер части	
	байты	%
Объявление интерфейса системы времени выполнения	759	4,5
Отладочное выполнение (без сохранения/восстановления)	1362	8,1
Последовательное выполнение	1377	8,2
Параллельное выполнение	2091	12,4
Имитационное выполнение	2620	15,6
Распределенное выполнение (без сохранения/восстановления)	5739	34,1
Операции сохранения/восстановления	699	10,1
Прочий код (инструкции препроцессора языка C++)	1169	7,0
Всего (без разделителей и комментариев)	**15816**	**100**
Размер файла templet.hpp	22616	

Таблица 2. Размеры частей кода примера программы проверки тригонометрического тождества (в байтах)

Часть кода	Размер части	
	байты	%
Код пользователя, всего	1143	100
В том числе: - логика вычислений, объявления данных - описание связей между акторами, ввод/вывод - анализ производительности при имитационном выполнении	 545 412 186	 47,7 36,0 16,3
Скелет программы, построенный по описанию на предметном языке	621	–
Исполняемый код, построенный по описанию на предметном языке	2166	–

языке. Поэтому детали сопряжения, например, инкапсуляция кода библиотеки внутри библиотеки не учитываются. За них отвечают подсистемы, связанные с обработкой предметного языка.

Варианты последовательного и отладочного выполнения реализуются относительно простым способом. Примерно в пять раз (по размеру кода) сложнее вариант распределенного выполнения. Имитационное и отладочное выполнение сложнее последовательного примерно в 1,5 раза. Однако код распределенного выполнения достаточно компактен (5739 байт).

В табл. 2 приведено сравнение частей кода, относящихся к логике взаимодействия акторов и логике вычислений тестовой задачи. Заметим, что доля вычислений по задаче, в силу ее простоты, незначительна (545 байт, 47,7 % процентов рукописного кода). Размер этой части кода практически не зависит от метода реализации: последовательного, параллельного с использованием OpenMP, распределенного с использованием MPI, рассматриваемого в работе метода на основе акторов. Поэтому пропорции кода пользователя в реальных примерах будут примерно соответствовать приведенным или изменяться в сторону увеличения части логики вычислений и объявления данных.

Размер скелета программы примерно в 3,5 раза меньше размера генерируемого из него исполняемого кода. Это наглядно показывает, что скелетное описание упрощает программирование. Полностью рукописная реализация примера на C++ с использованием стандартной библиотеки в акторном стиле потребовала бы 2166 + 15816 + 1143 = 19125 байт кода, в то время как пользователь в описываемой технологии вручную реализует лишь 1143 байта или около 6% от общего кода. Разумеется, сложность реализации сильно зависит от того, насколько метод распараллеливания решаемой задачи соответствует акторной модели. Однако, как показывает практика, большое число практически значимых задач хорошо отображаются на акторную модель вычислений.

ЗАКЛЮЧЕНИЕ

В текущей версии системы Templet Web, развернутой по адресу templet.ssau.ru/app на базе вычислительного кластера «Сергей Королёв» Самарского университета, в настоящее время поддерживаются предметный язык, использующий акторную модель вычислений Templet, и скелеты программ «портфель задач», «конвейер». Данная технология показала свою эффективность для решения прикладных задач анализа многомерных динамических систем и процессов при исследовании гироскопических систем ориентации космических аппаратов [12], а также при обучении студентов принципам работы на суперкомпьютере [3]. В дальнейшем планируется развивать данную технологию на основе подсистемы интегрированной среды разработки (IDE), встроенной в сервис Templet Web [18].

СПИСОК ЛИТЕРАТУРЫ

1. *Ward M.P.* Language-oriented programming // Software-Concepts and Tools, vol. 15, no. 4, pp. 147-161, 1994.
2. *Dmitriev S.* Language oriented programming: The next programming paradigm // JetBrains onBoard, vol. 1, no. 2, pp. 1-13, 2004.
3. *Артамонов Ю.С., Востокин С.В.* Применение облачного сервиса Templet Web при проведении лабораторных практикумов на суперкомпьютере «Сергей Королев» // X Международная научно-практическая конференция «Современные информационные технологии и ИТ-образование», МГУ, Москва, 2015. Том 2. С. 409-414.
4. *Востокин С.В.* Применение предметных языков для автоматизации высокопроизводительных вычислений // Перспективные информационные технологии (ПИТ 2016): труды Международной научно-технической конференции [под ред. С.А. Прохорова]. Самара: Издательство Самарского научного центра РАН, 2016. С. 490-493.
5. *Ага Г., Мейсон И., Смит С., Талкотт К.* Основания для вычислений акторов // Journal of Functional Programming, 1993.
6. *Скорюпина Е.Г, Востокин С.В.* Анализ реализаций акторной модели на платформе Java // Перспективные

информационные технологии (ПИТ 2016): труды Международной научно-технической конференции [под ред. С.А. Прохорова]. Самара: Издательство Самарского научного центра РАН, 2016. С. 532-534.
7. *Наширванов Д.М., Востокин С.В.* Акторная модель для статических алгоритмов в распределенных системах с использованием интерфейса передачи сообщений MPI // Перспективные информационные технологии (ПИТ 2016): труды Международной научно-технической конференции [под ред. С.А. Прохорова]. Самара: Издательство Самарского научного центра РАН, 2016. С. 515-517.
8. *Cole M.* Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming // Parallel computing, vol. 30, no. 3, pp. 389–406, 2004.
9. *González-Vélez H., Leyton M.* A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers // Software: Practice and Experience, vol. 40, no. 12, pp. 1135–1160, 2010.
10. *Литвинов В.Г.* Разработка и применение вычислительной модели типовых решений. Пример использования «портфеля задач» для обучения нейронной сети HRBF // Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. Науки, 3(36) (2014). С. 183–195.
11. *Востокин С.В.* Templet: язык разметки для параллельного программирования // Известия Самарского научного центра РАН. Том 17. 2015. № 2(5). С. 947-955.
12. Программный комплекс анализа многомерных динамических систем и процессов на суперкомпьютере «Сергей Королёв» / *С.В. Востокин, А.В. Дорошин, Ю.С. Артамонов, Ю.П. Назаров* // Управление движением и навигация летательных аппаратов: сборник трудов XVI Всероссийского семинара по управлению движением и навигацией летательных аппаратов. Самара: Издательство СНЦ РАН, 2013. С. 60-63.
13. Вторая волна разработки Java-приложений: Представляем Kilim. URL: https://www.ibm.com/developerworks/ru/library/j-javadev2-7/ (дата обращения 27.08.2016).
14. *Srinivasan S., Mycroft A.* Kilim: Isolation-Typed Actors for Java (A Million Actors, Safe Zero-Copy Communication), University of Cambridge Computer Laboratory. URL: https://www.cl.cam.ac.uk/research/srg/opera/publications/papers/kilim_ecoop08.pdf (дата обращения 27.08.2016).
15. Официальный сайт Actors Guild. URL: https://github.com/timjansen/actorsguild (дата обращения 27.08.2016).
16. Официальный сайт ActorFoundry. URL: http://osl.cs.illinois.edu/software/actor-foundry/ (дата обращения 27.08.2016).
17. Официальный сайт Akka. URL: http://akka.io/ (дата обращения 27.08.2016).
18. *Царёв Д.А., Артамонов Ю.С.* Сравнение основных возможностей и классификация облачных инструментов разработки // Перспективные информационные технологии (ПИТ 2016): труды Международной научно-технической конференции [под ред. С.А. Прохорова]. Самара: Издательство Самарского научного центра РАН, 2016. С. 539-542.

THE USE OF DOMAIN-SPECIFIC LANGUAGES AND ACTOR MODEL FOR THE AUTOMATION OF HIGH PERFORMANCE COMPUTING

© 2016 S.V. Vostokin, E.G. Skoryupina, D.M. Nashirvanov

Samara National Research University named after Academician S.P. Korolev

The article describes the use of actor-oriented domain-specific languages for the automation of high-performance scientific computing in the Templet Web cloud service. The automatic programming in Templet Web is based on the concepts of skeleton programming, domain-specific languages, and supporting visual editors. The sample program for checking the basic trigonometric identity was developed in the actor-based domain-specific language. While the program is simple enough, it implements all features of the domain-specific language. High-level task description is transformed into the code that uses actor-based run-time system. The article describes a measurement of the level of automation for the domain-specific language and actor programming method. The measurement is based on evaluating proportions of library code, generated code, and manually written code in the given sample. The criterion of complexity and degree of automation is based on a comparative analysis of code sizes from the different parts of the program. The sample is located at the GitHub service: https://github.com/templet-language/newtemplet. A high degree of automation is reached by simplicity of the actor model and by automatically constructed (from the domain-specific description) interface code. The current version of Templet Web cloud service is available at the following web address: http://templet.ssau.ru/app. It is deployed on supercomputer "Sergey Korolev" of Samara University. The service supports actor-based domain-specific language named Templet, and includes several ready-to-use skeleton programs: "TaskBag", "Pipeline", and others. The technology shows its effectiveness in the field of multivariate dynamic systems and processes, in researching gyroscope attitude control systems of spacecraft, and for teaching students the principles of work on a supercomputer.
Keywords: parallel programming, automatic programming, actor model of computation, domain-specific language, skeleton programming.

Sergey Vostokin, Doctor of Technics, Professor at the Information Systems and Technologies Department.
E-mail: easts@mail.ru
Ekaterina Skoryupina, Master Student.
E-mail: kat.skorupina@yandex.ru
Damir Nashirvanov, Master Student.
E-mail: damir.nashirvanov@gmail.com