Автоматизированная система параллельного программирования с использованием схем программ

Д.А. Уланов, С.В. Востокин

Самарский государственный аэрокосмический университет им. акад. С.П.Королева

Сегодня общепризнанна исключительная роль высокопроизводительной вычислительной техники в области математического моделирования. В некоторых специальных областях, таких как моделирование наноструктур, процессов нелинейной динамики, в авиационной, атомной промышленности, материаловедении, биотехнологии и других отраслях достичь адекватности математической модели реальным физическим процессам удается исключительно с использованием численных ресурсоемких моделей. Важным приложением высокопроизводительных вычислительных систем также являются задачи оптимизации из инженерной практики. В настоящее время в России достигнуты определенные успехи в области внедрения и разработки суперкомпьютеров, однако, как отмечают специалисты, уровень практического применения суперкомпьютеров и других высокопроизводительных систем (кластерных и многопроцессорных SMP-систем) остается низким.

Существуют объективные причины такого положения вещей. Основной причиной являются другие принципы программирования новых численных моделей. В эру персональных компьютеров и мощных рабочих станций построение численного алгоритма по аналитической модели фактически сводило дальнейшее решение к рутинной процедуре кодирования на известном процедурно-ориентированном языке программирования (Фортране или Си) и численным экспериментам по стандартным методикам. В случае использования суперкомпьютеров переход от последовательного алгоритма к программе на данный момент остается самостоятельной научной и инженерной проблемой, которую инженер-математик традиционной квалификации не в состоянии эффективно решать. Причина – в необходимости работать в необычной для него области параллельного и системного программирования. Экстенсивный путь преодоления этой проблемы заключается в возврате к методам организации численных экспериментов 1970-1980х годов, когда над проблемой работала группа специалистов, включающая, помимо инженеров-математиков, системных и прикладных программистов, операторов, техников. Естественно, добавление новых лиц в процесс разработки требует дополнительных расходов как на самих специалистов, так и на координирование их совместных действий. Целью предлагаемого в работе подхода является создание программного сервиса, позволяющего программистам-математикам, используя привычный для них способ построения и исследования математической модели, эффективно применять высокопроизводительную технику в своих исследованиях.

Наш метод имеет ряд отличий от традиционных архитектур сервисов. Обычно автоматизация работы с суперкомпьютером или кластерной системой заключается в построении web-интерфейса, устраняющего необходимость знания специальных команд управления системой. Задачу упрощения доступа к вычислительному ресурсу также решают @home-сервисы с той разницей, что вычислительный ресурс является не централизованным, а распределенным.

Предлагаемый сервис реализует парадигму «виртуального суперкомпьютера», который, в отличие от известных подходов, является пассивным хранилищем устанавливаемых на него программ. Причем сами устанавливаемые программы – это обычные последовательные программы на исходном языке (например, на Си), но написанные с учетом ограничений, проверяемых сервисом. Выполнив установку на «виртуальный суперкомпьютер», мы получаем доступ к настоящей откомпилированной параллельной

программе для высокопроизводительных вычислений, которую можно установить на реальную систему. Основное внимание в нашем сервисе уделяется процессу автоматизации программирования, компиляции, дистрибуции, гарантии свойств создаваемых сервисом программ, в том числе, связанных с безопасностью и эффективностью исполнения. При этом мы стремимся минимизировать зависимость от целевой вычислительной системы: в сервисе генерируется несколько версий исполняемого кода для обеспечения переносимости. Также мы стремимся к независимости от платформы программирования и, частично, от языка программирования. Таким образом, предлагаемый сервис инкапсулирует не физический ресурс, а системный код, выполняющий функции сборки программ и управления вычислительным процессом, разработка которого как раз и вызывает затруднения у инженеров-математиков.

Теоретической основой предлагаемого подхода является теория схем программ и современные неформальные методы проектирования, основанные на паттернах проектирования. Ядро сервиса использует на следующую простую формализацию понятия паттерна проектирования.

Пусть алгоритм реализуется на некотором языке L и пусть $S_0, S_1, S_2, ..., S_n, P_1, P_2, ..., P_n$ – фрагменты цепочек языка L, т.е. для них можно найти такие фрагменты языка α и β, что $\alpha \cdot S_i \cdot \beta \in L$ и $\alpha \cdot P_i \cdot \beta \in L$:

$$S_i : \exists \alpha, \beta \quad \alpha \cdot S_i \cdot \beta \in L, \qquad (1)$$
$$P_i : \exists \alpha, \beta \quad \alpha \cdot P_i \cdot \beta \in L.$$

Дадим определение понятию шаблона. Под шаблоном будем понимать язык $S(S_0, S_1, S_2, ..., S_n) \subset L$, цепочки которого будут иметь вид

$$S_0 \cdot P_1 \cdot S_1 \cdot P_2 \cdot S_2 \cdot ... \cdot P_{n-1} \cdot S_{n-1} \cdot P_n \cdot S_n \cdot \qquad (2)$$

Реализация шаблона S - это цепочка на языке L, которая получается в результате подстановки пользовательских значений в параметры $P_1, P_2, ..., P_n$ шаблона. Здесь фрагменты S_i цепочки (2) фактически являются участками системного кода, который отвечает за исполнение и реализацию алгоритма в рамках шаблона, а фрагменты P_i - участки «полезного» кода, непосредственная реализация конкретного алгоритма, при этом алгоритм однозначно задается фрагментами $P_1, P_2, ..., P_n$.

Между шаблонами можно определить операцию отображения. Шаблон S' может быть отображен в шаблон S'', если для любого набора фрагментов $\{P_1, P_2, ..., P_n\}$ цепочки $S_0' \cdot P_1 \cdot S_1' \cdot ... \cdot P_n \cdot S_n'$ и $S_0'' \cdot P_1 \cdot S_1'' \cdot ... \cdot P_n \cdot S_n''$ будут иметь одинаковый смысл с точки зрения алгоритма, представленного множеством $\{P_1, P_2, ..., P_n\}$. Иначе говоря, программа после отображения из шаблона S' в шаблон S'' (фактически замены S_i' на S_i'') сохраняет свою логику с точностью до неопределенностей исполнения

Назовем два шаблона S' и S'' взаимно отображаемыми, если S' можно отобразить в S'' и S'' можно отобразить в S'. Множество шаблонов $\{S', S'', ..., S^{(k)}\}$ в котором любые два шаблона взаимно отображаемы, будем называть схемой. Схема по сути представляет собой описание одного паттерна параллельного программирования в нескольких его реализациях, например, в последовательном или параллельном виде. При использовании схемы программист, как правило, реализует свой алгоритм при помощи наиболее простого последовательного шаблона (который достаточно удобен для локальной отладки), а далее при необходимости реализации этого алгоритма в любом распределенном виде система осуществляет отображение в соответствующие шаблоны схемы.

В предложенной модели шаблоны могут быть вложены друг в друга, т.е. в шаблоне на месте фрагментов P_i вместо пользовательского кода могут присутствовать цепочки других шаблонов. Таким образом, на основе простых шаблонов возможно создание сложных структур. Частным случаем вложенности шаблонов является практически полная замена пользовательских фрагментов P_i на системные фрагменты таким образом, что в результирующем шаблоне программисту необходимо будет только ввести входные параметры алгоритма и обработать результат вычисления. То есть в предложенной модели присутствует возможность конкретизации шаблонов.

Одной из задач системы распределенного программирования является определение принадлежности программы $C \in L$ шаблону $S(S_0, S_1, S_2, ..., S_n)$ для случая, когда пользователь загружает в систему локально созданную программу на основе одного из шаблонов. В рамках этой задачи распознаются границы между фрагментами S_i и P_i, т.к. при выделении из множества можно однозначно сделать вывод о шаблоне программы. Данная задача решается путем поиска вхождений фрагментов S_i каждого шаблона в программу *C* с точностью до лексемы языка *L*. Если программу *C* не удалось сопоставить ни одному шаблону, то делается вывод об изменении пользователем локальных фрагментов , система отказывает пользователю в занесении программы *C* в систему. Алгоритм поиска вхождений является общим для всех шаблонов.

Еще одной задачей системы является обеспечение безопасности удаленно исполняемого кода. Система должна анализировать код пользователя и не запускать на исполнение код, который может содержать вредоносные фрагменты. Для этого в языке L выделяют подмножество L', в которое входят только те лексемы языка, использование которых не может привести к написанию вредоносного кода (базовые типы, циклы, ветвления, математические операторы и константы и пр.). Данное множество необходимо дополнить множеством локальных переменных пользователя P', а также множеством общедоступных переменных шаблона S'. Перед запуском на исполнение все лексемы программы пользователя C проверяются на вхождение во множество L' $\cup$ P' $\cup$ S'. Если в программе присутствуют лексемы, которые не входят в указанное множество, то происходит отказ от исполнения программы C.

Нами был разработан действующий прототип сервиса, реализующий следующие функции:

- Создание программы на основе одной из существующих схем;
- Загрузка и редактирование программы в виде локального шаблона схемы;
- Проверка безопасности пользовательской программы;
- Исполнение программы в виде шаблонов для кластера (с использованием технологии Silverlight в качестве среды исполнения) и для суперкомпьютера (с использованием MPI);
- Хранение программ, истории и результатов исполнения.

Система реализована на основе сервис-ориентированного подхода и представляет из себя набор серверов. Каждый из серверов представляет набор WCF-сервисов в соответствии со своими функциональными обязанностями (рисунок 1). Разделение функциональности по серверам позволяет достичь масштабируемости системы без ущерба производительности.

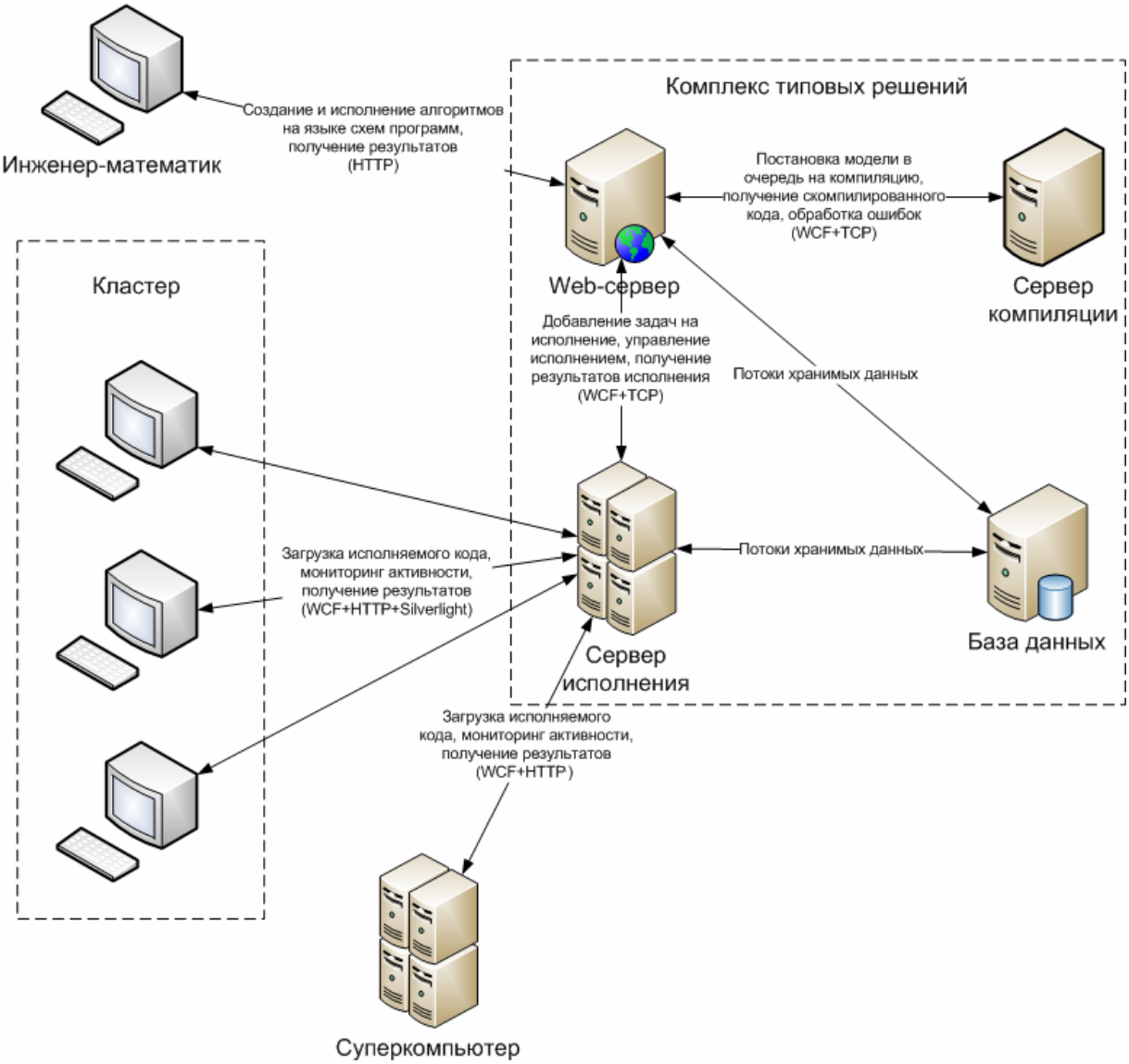

Рисунок 1. Структурная схема автоматизированной системы.

Работа математика с системой происходит следующим образом. При создании программы необходимо выбрать схему и ввести код алгоритма (фрагменты кода P_i) посредством web-интерфейса либо при помощи локального средства разработки (для этого необходимо загрузить проект с фрагментами кода S_i на локальную ЭВМ, ввести фрагменты P_i и выгрузить обновленный проект обратно на сервер). Далее осуществляется проверка безопасности кода по предложенной методике, и в случае успеха программу в дальнейшем можно компилировать при помощи одного из шаблонов схемы, после успешной компиляции – запускать на соответствующей вычислительной среде. На данный момент в каждой схеме системы поддерживаются следующие типы шаблонов:

- последовательный шаблон, наиболее удобный для ввода кода алгоритма и его отладки;
- кластерный шаблон, программа на таком шаблоне предназначена для исполнения на кластере. Исполняемый код на каждый из кластеров загружается и исполняется в браузере посредством подписанного объекта Silverlight.
- шаблон на MPI, программы на данном шаблоне исполняются на удаленном суперкомпьютере.

В системе предусмотрена авторизация, разграничение прав доступа и подсистема администрирования.

Целью дальнейших исследований является дальнейшее развитие системы в рамках практического обоснования предлагаемого подхода, в частности разработка достаточно полной библиотеки схем, охватывающей часто применяемые модели вычислительных процессов.

Литература

1. Востокин С.В. Графическая объектная модель параллельных процессов и ее применение в задачах моделирования – Самара: Изд-во Самарского научного центра РАН, 2007. – 286 с.
2. Берзигияров П.К. Программирование на типовых алгоритмических структурах с массивным параллелизмом – Москва: Вычислительные методы и программирование, Т. 2, разд. 2, 2001. – 16 с.