УДК 681.3:681.3.06:681.3.01:681.3.068
doi:10.15217/issn1684-8853.2015.6.57

ТЕХНОЛОГИИ РЕШЕНИЯ СЛОЖНЫХ ЗАДАЧ НА ОСНОВЕ ДИНАМИЧЕСКИХ АВТОМАТНЫХ СЕТЕЙ

В. А. Торгашев[a], *доктор техн. наук, профессор*
И. В. Царев[a], *ведущий программист*
[a]*Санкт-Петербургский институт информатики и автоматизации РАН, Санкт-Петербург, РФ*

Постановка проблемы: *современные суперкомпьютеры, используемые для решения сложных задач, обладают рядом недостатков как по технико-экономическим характеристикам (габариты, энергопотребление, стоимость), так и по сложности программирования реальных задач, требующих специальных приемов распараллеливания программ, в результате чего производительность суперкомпьютеров при решении реальных задач оказывается значительно ниже, чем их теоретическая производительность. Целью работы является разработка методов создания суперкомпьютеров и технологий программирования, основанных на теории динамических автоматных сетей, позволяющих существенно улучшить удельные характеристики суперкомпьютеров, а также упростить параллельное программирование для решения соответствующих задач.* ***Результаты:*** *сформулированы базовые принципы создания суперкомпьютеров с динамической архитектурой на основе динамических автоматных сетей, включая реализацию динамических автоматов с использованием либо интегральных микросхем с гибкой программируемой логикой, либо специально разрабатываемых отечественных больших интегральных схем. Это обеспечивает крайне высокую регулярность структуры микросхем, что существенно упрощает создание суперкомпьютеров с динамической архитектурой по сравнению с традиционными суперкомпьютерами. Предложены технологии решения сложных задач с использованием специально разработанного языка программирования, основанного на динамических автоматных сетях, а также метод «гибридного программирования», позволяющий сочетать различные аппаратные и программные средства для решения одной задачи.* ***Практическая значимость:*** *предложенные методы дают возможность создавать суперкомпьютеры с динамической архитектурой, многократно (в десятки и сотни раз) превосходящие традиционные суперкомпьютеры по таким удельным (в расчете на единицу производительности) характеристикам, как габариты, энергопотребление и стоимость, при этом структура аппаратной реализации суперкомпьютера значительно упрощается. Предложенные технологии программирования для решения сложных задач, включая «гибридное программирование», упрощают сам процесс параллельного программирования и повышают эффективность решения сложных задач на суперкомпьютерах.*

Ключевые слова *— динамические автоматные сети, параллельное программирование, суперкомпьютеры с динамической архитектурой.*

Введение

За последние десятилетия со времени появления первых персональных компьютеров (ПК) вычислительная техника совершила гигантский скачок в части производительности компьютеров, объемов памяти не только за счет успехов в микроэлектронике, прежде всего микроминиатюризации, увеличения тактовых частот, уменьшения потребления электроэнергии, но и отчасти благодаря изменениям в архитектуре и программном обеспечении. К этому добавилось появление специализированных графических процессоров, которые в настоящее время присутствуют в каждом компьютере на плате графического контроллера (видеокарты), что позволяет весьма эффективно обрабатывать разнообразную графическую информацию и отображать ее на экране монитора.

В результате обычный современный настольный ПК с четырехъядерным процессором и тактовой частотой около 2–3 ГГц на порядки (в сотни и тысячи раз) превосходит по производительности и объемам памяти (как оперативной, так и внешней, дисковой) не только первые ПК, но и лучшие ЭВМ предыдущих поколений, например такие, как БЭСМ-6 или даже первый отечественный мультипроцессорный вычислительный комплекс «Эльбрус», состоявший из десяти процессоров. То же можно сказать и о любых других ЭВМ 70-х — начала 80-х годов прошлого века, которые занимали целый зал любого вычислительного центра, требовали больших затрат электроэнергии и специального охлаждения (как минимум кондиционирования).

Современный ПК способен успешно и эффективно решать множество разнообразных задач, удовлетворяя потребности самых различных пользователей, работающих с задачами вычислительного характера, с базами данных, с сетями, включая Интернет, с компьютерной графикой и т. д.

Однако существует целый ряд задач, для которых вычислительная мощность даже современных ПК явно недостаточна, но не в том смысле, что на ПК эти задачи совсем невозможно решить, а это решение требует весьма больших затрат времени и объемов дополнительной памяти.

Будем далее называть такие задачи *«сложными»*. К ним относятся, прежде всего, так называемые «массивно-параллельные» задачи, включающие работу с большими матрицами и векторами размерностью в сотни, тысячи и более элементов (для матриц — строк и столбцов), решение

линейных алгебраических уравнений и дифференциальных уравнений большого порядка, а также некоторые задачи искусственного интеллекта, включающие многократный перебор и сравнение различных объектов, распознавание образов, анализ сцен и пр. Также к «сложным» задачам можно отнести некоторое множество задач реального времени, включающее задачи обработки различных сигналов, управления реальными сложными объектами, моделирования в реальном времени сложных объектов и систем и т. д.

Традиционно для решения подобных «сложных» задач используются либо специализированные процессоры или многопроцессорные комплексы, либо высокопроизводительные мультипроцессорные вычислительные системы, называемые «суперкомпьютерами». Производительность лучших современных суперкомпьютеров достигает петафлопного диапазона производительности (10^{15} операций с плавающей точкой в секунду и более), ведутся разработки по созданию суперкомпьютеров экзафлопного диапазона производительности (10^{18} операций с плавающей точкой в секунду и более). Характеристики лучших пятисот суперкомпьютеров дважды в год публикуются на сайте «top500.org». Из них только незначительная часть достигла производительности петафлопного диапазона.

Однако почти все современные суперкомпьютеры обладают рядом существенных недостатков. Прежде всего, это значительные габариты (практически любой современный суперкомпьютер занимает целый зал, как и компьютеры прежних поколений), высокое энергопотребление (обычно это мегаватты электроэнергии, при этом зачастую система охлаждения также потребляет мегаватты, иногда больше, чем сам суперкомпьютер).

Помимо чисто технических характеристик, современные суперкомпьютеры обладают следующим основным недостатком. Они, как правило, построены на множестве обычных последовательных процессоров, основанных на так называемой «традиционной» или «фон-неймановской» архитектуре, предполагающей представление задачи в виде алгоритма, т. е. заранее определенной последовательности действий, предложенной как самим Джоном фон Нейманом (в виде рукописного, неопубликованного, но общеизвестного доклада), так и, в некоторой другой форме, Аланом Тьюрингом [1]. Использование в отдельных суперкомпьютерах графических процессоров для работы с большими массивами принципиально ничего не меняет. В любом случае для решения вышеперечисленных задач необходимо искусственное «распараллеливание» вычислительного процесса с применением специальных приемов программирования, требующих весьма высокой квалификации программистов. Также при программировании таких параллельных процессов следует учитывать характеристики конкретного суперкомпьютера, по крайней мере количество процессоров, объемы памяти и т. д.

Несоблюдение этих требований может в каждом конкретном случае приводить к существенному снижению фактической, реальной производительности. Одной из основных причин этого является так называемый «семантический разрыв» (термин впервые введен Г. Майерсом [2]) между принципиально параллельной структурой решаемой задачи и последовательным характером процессоров (и языков программирования).

Целью данной работы является разработка архитектурных (а в дальнейшем и схемотехнических) решений для создания суперкомпьютеров с динамической архитектурой (СКДА), основанных на теории динамических автоматных сетей (ДАС), а также методов программирования и технологий решения задач с использованием СКДА, что позволит в значительной степени преодолеть вышеуказанные проблемы.

Динамические автоматные сети

В статье В. А. Торгашева [3] показано, что автор алгоритмической модели вычислений Джон фон Нейман сам полагал, что эта модель просуществует не более пятнадцати лет, а будущее — за автоматными сетями, которые он же и предложил в работе о «самовоспроизводящихся автоматах» [4]. Тем не менее уже практически 70 лет алгоритмическая модель существует и доминирует в разработке архитектуры любых компьютеров.

В 80-х годах прошлого века В. А. Торгашев предложил динамические автоматные сети в качестве модели любых вычислений, в том числе и параллельных. Идея ДАС основана на вышеупомянутой работе Джона фон Неймана, а также на работах Улама и Барзиня [5, 6].

Данная статья не предполагает подробного изложения теории ДАС, для ознакомления с теоретическими основами и некоторыми возможными реализациями ДАС отсылаем читателя к работам [7–11]. Однако приведем здесь краткое описание ДАС, достаточное для понимания проблемы.

Динамическая автоматная сеть представляет собой множество «динамических автоматов» (ДА), соединенных между собой «связями» в некоторую сетевую структуру. Каждый ДА — это конечный автомат, свойства которого, помимо обычных множества входов (входной алфавит), множества выходов (выходной алфавит), множества текущих состояний (включая начальное состояние) самого автомата и функции переходов, дополнены множеством связей и множеством функций связей (или одной такой функцией). Множество связей, собственно, и объединяет множество автоматов

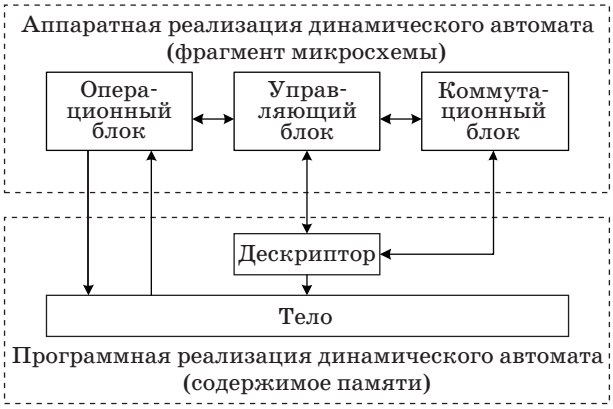
в сеть. Функции связей определяют возможность изменения связей между автоматами в сети, а также порождение новых автоматов или уничтожение автоматов, выполнивших свою функцию, что происходит в процессе решения некоторой конкретной задачи.

Поскольку в любой конкретной физической реализации количество ДА ограничено возможностями аппаратуры, то на практике имеется некоторое множество (список) «свободных автоматов», которые не включены ни в какую ДАС и конкретные функции которых не определены. Таким образом, создание (порождение) новых ДА сводится к исключению их из списка «свободных автоматов» и включению в ДАС посредством определения конкретных связей ДА с другими автоматами ДАС (называемыми «смежными») и приписыванию ДА конкретных функций. Уничтожение автоматов, выполнивших свою функцию, сводится к разрыву связей автомата, исключению его из ДАС и возвращению в список свободных автоматов.

В процессе решения задачи ДАС постоянно меняет свою конфигурацию посредством порождения или уничтожения автоматов, входящих в нее, а также изменения связей между ними. Это называется «автотрансформацией» ДАС. Постоянное изменение конфигурации ДАС, включая и уничтожение автоматов, позволяет существенно сэкономить аппаратные ресурсы и объемы используемой памяти, а также более эффективно задействовать для вычислений всю имеющуюся аппаратуру.

Каждый ДА может быть отнесен к одному из двух основных классов — «операционным автоматам» или «ресурсным автоматам». Первые предназначены для выполнения некоторой вычислительной функции, например некоторой арифметической или логической операции, при этом разрядность аргументов операции (операндов), как и результатов выполнения операции, и формат их представления (разрядность, фиксированная или плавающая точка и т. п.) теоретически могут быть любыми.

«Ресурсные автоматы» являются «хранилищем данных», т. е. содержат данные в любой форме (в том числе в форме ссылок или массивов ссылок на другие ресурсные автоматы, а также на другие, более сложные структуры данных) и могут быть «аргументами» (операндами) операционных автоматов или их «результатами» в зависимости от того, к каким входам или выходам операционного автомата присоединен данный ресурсный автомат посредством ссылок (связей). Они же могут быть не только хранилищем, но и поставщиком данных от некоторого внешнего источника (множества некоторых датчиков, радиолокационных, гидроакустических или телеметрических систем и т. д.).



■ ***Рис. 1.*** Аппаратно-программная реализация динамического автомата

Структура аппаратно-программной реализации ДА представлена на рис. 1.

В работе [11] приведен пример автотрансформации ДАС на основе простейшей задачи перемножения двух матриц. К сожалению, объем любой статьи не позволяет привести иллюстрацию более сложных преобразований ДАС (таких как при обработке сигналов по алгоритмам быстрого преобразования Фурье), но вариант программы перемножения матриц является достаточно информативным. Главное здесь заключается в том, что любой операционный автомат, аргументом которого является некоторый массив (матрица, вектор) или структура данных, вместо своей вычислительной функции выполняет функцию порождения «подсетей», в каждой из которых присутствует копия того же операционного автомата, а его аргументами и результатами являются фрагменты исходных и результирующих данных (элементы массивов, строки и столбцы матриц, поля структуры и т. д.).

Таким образом, как правило, в начале решения задачи ДАС расширяется путем порождения новых подсетей и автоматов, а к концу решения — уменьшается, при этом распараллеливание задачи, т. е. разделение задачи на параллельные ветви и их распределение между аппаратными ресурсами, происходит автоматически. Решением задачи (результатом вычислений) в общем случае является некоторая структура данных (сеть из ресурсных автоматов), потерявшая способность к автотрансформации, поскольку не остается ни одного операционного автомата, способного порождать новые подсети, так как все автоматы, выполнившие свою функцию, уже уничтожены. Такой результат может быть любым способом выведен во внешнюю среду (записан на дисковый накопитель, распечатан, выведен на экран монитора и т. д.). В любом случае распараллеливание вычислительного процесса не требует каких-то особых приемов программирования и каких-либо дополнительных усилий от программиста.

Теперь рассмотрим возможное представление любого ДА и ДАС в целом на уровне аппаратно-программной реализации. Операционный автомат, прежде всего, может быть представлен как фрагмент некоторой интегральной микросхемы, в котором реализована некоторая вычислительная функция (операционный блок). Обычно это схема умножения/деления с плавающей точкой, из которой посредством логического отсечения отдельных цепей могут быть получены виртуальные устройства для операций с фиксированной точкой или логических операций. Отсечение легко выполняется посредством использования некоторого управляющего двоичного кода, по сути представляющего собой «код операции», биты которого, в зависимости от состояния, разрешают или блокируют те или иные цепи операционного блока. Например, блокировка работы с порядком превращает операцию с плавающей точкой в операцию с фиксированной точкой.

Кроме вычислительной функции, любой автомат обладает также коммутационной и управляющей функциями (и содержит соответствующие аппаратные блоки). Коммутационная функция позволяет передавать информацию между различными автоматами и подсетями (без ущерба для выполнения основной вычислительной функции), порождать новые автоматы и подсети либо уничтожать их. Управляющая функция анализирует состояние (статус) самого автомата и связанных с ним ресурсных автоматов (аргументов и результатов), принимая решение о том, какая из функций автомата будет выполняться (вычислительная функция или порождение/уничтожение подсетей). При этом функции автомата всех трех видов выполняются параллельно и не мешают друг другу, поскольку используют различные фрагменты аппаратуры.

Операционный автомат начинает выполнять свою вычислительную функцию, когда он сам находится в состоянии «готовности», его «аргументы» (ресурсные автоматы) также находятся в том же состоянии, а «результаты» находятся в состоянии «неопределенности».

По завершении выполнения вычислительной функции операционный автомат уничтожается, уничтожаются и его аргументы, если они не являются также аргументами других операционных автоматов. В то же время ресурсный автомат, являющийся «результатом», переходит в состояние «готовности», что может, в свою очередь, вызвать выполнение вычислительной функции некоторого другого операционного автомата, аргументом которого он является.

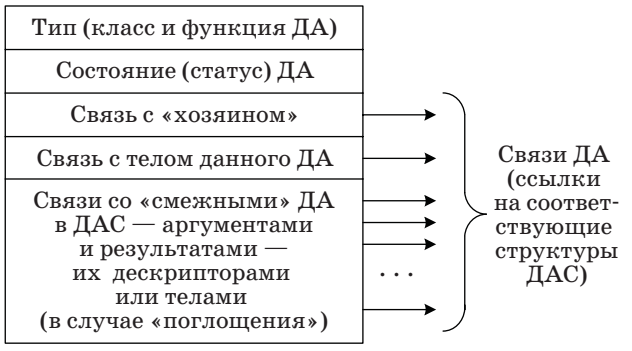
В то же время каждый автомат (операционный или ресурсный), помимо аппаратного представления, представлен некоторой структурой в памяти, состоящей в общем случае из двух компонентов — дескриптора и тела.

Дескриптор — это некоторая структура данных, которая содержит информацию о типе автомата (его классе и функции, представленной кодом операции), его состоянии («статусе»), определяющем готовность, неготовность, неопределенное состояние и т. п., а также информацию о связях этого автомата с другими автоматами ДАС (аргументами, результатами). Связи автомата представлены ссылками на дескрипторы «смежных» автоматов, связанных с данным в рамках ДАС. Для «свободных» (т. е. не включенных в ДАС) автоматов «смежными» являются автоматы из множества (списка) «свободных» автоматов. Особо следует отметить, что автомат, породивший некоторую подсеть (т. е. некоторое множество автоматов, связанных между собой), является «хозяином» этой подсети, соответственно, каждый автомат имеет связь (ссылку) с его «хозяином», а «хозяин» имеет в качестве своего «тела» некоторый массив ссылок на порожденные им автоматы. Если «терминальный» автомат, т. е. автомат, не породивший своих «потомков», уничтожается сразу после выполнения своей функции, то «хозяин» может быть уничтожен только после уничтожения всех его «потомков», выполнивших свою функцию.

Структура дескриптора ДА представлена на рис. 2.

«Тело» автомата — это некоторая структура данных в памяти, связанная с данным автоматом посредством ссылок (адресов памяти). Для операционных автоматов это может быть, например, некоторая структура из ссылок на порожденные автоматы (их дескрипторы), для ресурсного автомата — это обычно некоторый массив или структура данных. Уничтожение автомата приводит и к уничтожению его тела, т. е., по сути, к освобождению памяти, занимаемой данным автоматом, который выполнил свою функцию.

Перед выполнением вычислительной функции операционного автомата может быть выполнено «поглощение» операционным автоматом сво-



■ ***Рис. 2.*** Структура дескриптора динамического автомата

их аргументов, которое сводится к замене ссылок на «дескрипторы» аргументов (ресурсных автоматов) на прямые адреса массивов, составляющих «тело» автоматов-аргументов. При этом автоматы-аргументы уничтожаются, точнее, уничтожаются их дескрипторы и соответствующие связи.

Возможно также и преобразование класса автомата. Например, операционный автомат, выполнивший свою функцию, может быть преобразован в ресурсный автомат, содержащий результат вычислений (если предварительно выполнено «поглощение» операционным автоматом также и ресурсного автомата, соответствующего своему результату). В этом случае автомат теряет связь со своей операционной функцией (изменяется его тип, код операции и статус), но сохраняет связь с «телом» результата.

Построение СКДА на основе динамических автоматных сетей

Использование «динамической архитектуры» и теории динамических автоматных сетей для разработки СКДА позволяет создавать суперкомпьютеры в широком диапазоне производительности, от нескольких терафлопс (10^{12} операций с плавающей точкой в секунду) до петафлопного и экзафлопного диапазона производительности. При этом не требуется никаких изменений в архитектурных, схемотехнических и программных решениях благодаря свойству СКДА, называемому «неограниченной масштабируемостью». Либо существующие суперкомпьютеры вообще не обладают свойством масштабируемости, либо масштабируемость ограничивается небольшими степенями двойки (2–4–8–16).

Неограниченная масштабируемость СКДА дает потенциальному пользователю возможность наращивать производительность суперкомпьютера по мере возникновения потребности в большей производительности путем добавления новых блоков и соединения их стандартным образом, никак не меняя при этом программное обеспечение.

Важнейшим свойством реализации СКДА является возможность существенного (в разы и даже на порядки) улучшения, по сравнению с существующими суперкомпьютерами, таких удельных (в расчете на 1 терафлопс производительности) характеристик, как габариты, энергопотребление и стоимость. Достигается это за счет отказа от использования сложных последовательных процессоров, работающих на частотах в несколько гигагерц, в пользу простых автоматов, выполняющих всего одну функцию и работающих на частотах в несколько сотен мегагерц, что, в свою очередь, существенно снижает требования к системам охлаждения. Кроме того, использование низких частот позволяет отказаться от сложных многоуровневых структур памяти (кэш-памяти разных уровней и т. д.), поскольку существующие микросхемы памяти вполне способны работать на этих частотах. А общая эффективность системы достигается благодаря одновременной параллельной работе большого множества ДА.

Реализация множества ДА возможна как с использованием интегральных схем с программируемой (гибкой) логикой, так и на основе специально разработанных «больших интегральных схем», при этом вполне возможна разработка отечественных больших интегральных схем на основе доступных в России технологий (в настоящее время уже доступна технология 28 нм). Количество ДА, реализуемых в одной микросхеме, может исчисляться сотнями тысяч. Соответственно, небольшой образец СКДА с производительностью до 20 терафлопс может содержать многие сотни тысяч или миллионы ДА, имея геометрический объем меньший, чем у системного блока обычного ПК. Пропорционально уменьшается и энергопотребление, и стоимость СКДА.

Существенной особенностью аппаратной реализации СКДА является высокая степень регулярности структуры микросхем, так как блоки всех ДА идентичны.

Программирование, основанное на ДАС, не требует чрезмерно высокой квалификации программистов для решения любых задач, в том числе и «сложных». Любая задача легко может быть представлена в виде сети, в которой присутствуют как объекты (данные, в том числе получаемые с различных физических источников информации — множества датчиков, радиолокационных, гидроакустических, телеметрических и иных систем), так и некоторые операции над этими данными. Такое представление задачи предполагает не столько какие-то особые навыки в программировании, сколько хорошее знание самой задачи и умение изобразить ее в виде сетевой структуры, что не является особо сложным. В то же время распараллеливание задачи в СКДА и распределение фрагментов программы (узлов ДАС) между вычислительными ресурсами происходит автоматически, во время вычислений, без какого-либо участия программиста. Программисту также не требуется иметь какой-либо информации об аппаратных ресурсах (объемах памяти, количестве процессоров или аппаратно реализуемых автоматов). В случае недостатка аппаратных ресурсов порождение новых автоматов приостанавливается, пока не будут освобождены соответствующие ресурсы в результате завершения некоторой части вычислений.

Следует отметить, что типовая архитектура и схемотехника СКДА не предполагает непосредственного присоединения к его аппаратуре каких-либо внешних устройств, таких как запоминающие устройства, мониторы, клавиатуры,

источники сигналов и другой информации и пр., что существенно упрощает структуру самого СКДА. Для этого в состав любого СКДА включается «хост-машина», представляющая собой обычный ПК, который присоединяется к СКДА посредством обычного интерфейса (например, USB). Хост-машина обеспечивает хранение и ввод информации в СКДА, управление процессом загрузки и запуска задач, а также вывод во внешнюю среду результатов.

Еще одной особенностью реализации СКДА на основе ДАС, вытекающей из вышесказанного, является отсутствие собственной операционной системы. В СКДА нет необходимости (и возможности) в организации пользовательского интерфейса, обслуживании файловой системы и внешних устройств — все эти функции выполняются на хост-машине с обычной операционной системой типа Windows или Linux. Функции же операционной системы, связанные с распределением ресурсов, запуском и завершением программ (в данном случае — программ, представленных в виде ДАС), включая распараллеливание вычислительного процесса, выполняются на аппаратном уровне благодаря свойствам динамических автоматов и ДАС в целом.

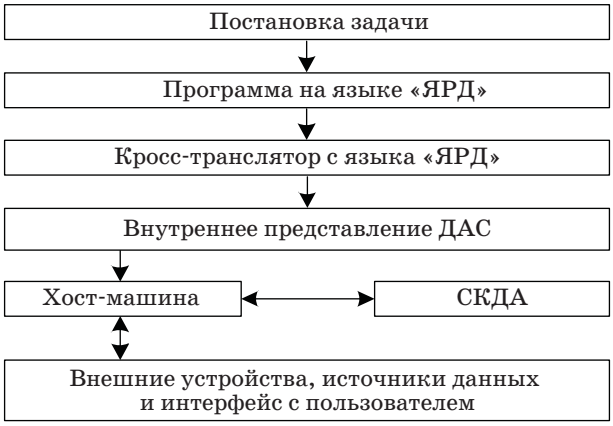
Технологии решения задач в СКДА

Любая технология предполагает наличие некоторых методов, средств и процессов, приводящих от постановки задачи к ее решению. В данном случае метод заключается в использовании ДАС для представления (и выполнения в СКДА) задачи, процесс решения состоит в автотрансформации ДАС, описанной выше. Средства решения задачи состоят в аппаратно-программной реализации ДАС, включая аппаратную реализацию СКДА, разработку средств программирования на основе ДАС, а также средств «гибридного программирования», которые описаны ниже.

Программирование на основе динамических автоматных сетей

Для представления задачи в виде ДАС был разработан специальный язык программирования «ЯРД» (язык рекурсивный динамический) [12] и кросс-транслятор с этого языка во внутреннее представление ДАС (дескрипторы и тела ДА и связи между ними). В настоящее время разрабатывается новый вариант этого языка с учетом изменений в возможной аппаратной реализации, что, скорее всего, приведет к некоторому, но не слишком принципиальному, упрощению языка. Технология решения задач на основе ДАС и языка «ЯРД» представлена на рис. 3.

Главной особенностью языка «ЯРД» является возможность представления программы в двух формах — графической и текстовой, которые отчасти взаимозаменяют, а в большей степени взаимодополняют друг друга.



■ ***Рис. 3.*** Технология решения задач на основе ДАС

Создание графической формы представления программы в языке «ЯРД» обеспечивается некоторым встроенным в транслятор графическим редактором. Практически пользователь (программист) может «нарисовать» ДАС, соответствующую представлению задачи или ее частей, в виде графа (множества графов), при этом используются специальные графические обозначения для операторов, данных, ссылок и других узловых компонентов сетевой программы, а связи между ними представляются стрелками. Могут быть использованы обозначения, стандартные в языке и снабженные идентификаторами или другими обозначениями (например, обозначениями операций). Но их можно, по желанию программиста, заменять на специально созданные специфические обозначения объектов, имеющие вид «иконок».

Транслятор автоматически строит текстовую форму представления программы на основе графической (впрочем, возможно и обратное). Графическая форма является весьма наглядной и позволяет представить общую структуру задачи или ее частей, но отдельные компоненты ДАС и их свойства требуют уточнения на уровне текстовой формы языка (в том числе это возможно в диалоговом режиме посредством заполнения некоторых форм в графическом представлении программы, появляющихся при нажатии кнопки мыши, курсор которой указывает на соответствующий объект).

Программа может быть разработана и исключительно с применением текстовой формы языка, а автоматически формируемая графическая форма может быть в этом случае использована как наглядная иллюстрация.

Текстовая форма языка «ЯРД» во многом по синтаксису напоминает общеизвестный язык Pascal, хотя кардинально отличается в части семантики.

Прежде всего, не имеет никакого значения последовательность «операторов» и «выражений» в тексте программы, поскольку программа — это не описание последовательности действий (алгоритма), а описание начальной сетевой структуры задачи или ее частей, соответствующих ДАС.

В языке отсутствуют циклы, хотя имеется некоторая синтаксическая конструкция **«for»**, которая на самом деле определяет параллельное выполнение ее компонентов.

В языке запрещено повторное присваивание одной и той же переменной (идентификатору) значения какого-то выражения, поскольку каждая переменная представляет собой некоторый объект ДАС, а операция присваивания соответствует связи операционного автомата (или группы операционных автоматов, представленных в виде ДАС) с его результатом, и этот объект может не существовать в момент выполнения присваивания, если одно присваивание этой переменной уже было выполнено. То есть «переменная» (представленная идентификатором) в этом языке — это не указатель на область памяти, а уникальный динамический объект, реализуемый в виде ресурсного динамического автомата. «Выражение» также представляет собой не последовательность действий, а описание соответствующего фрагмента ДАС.

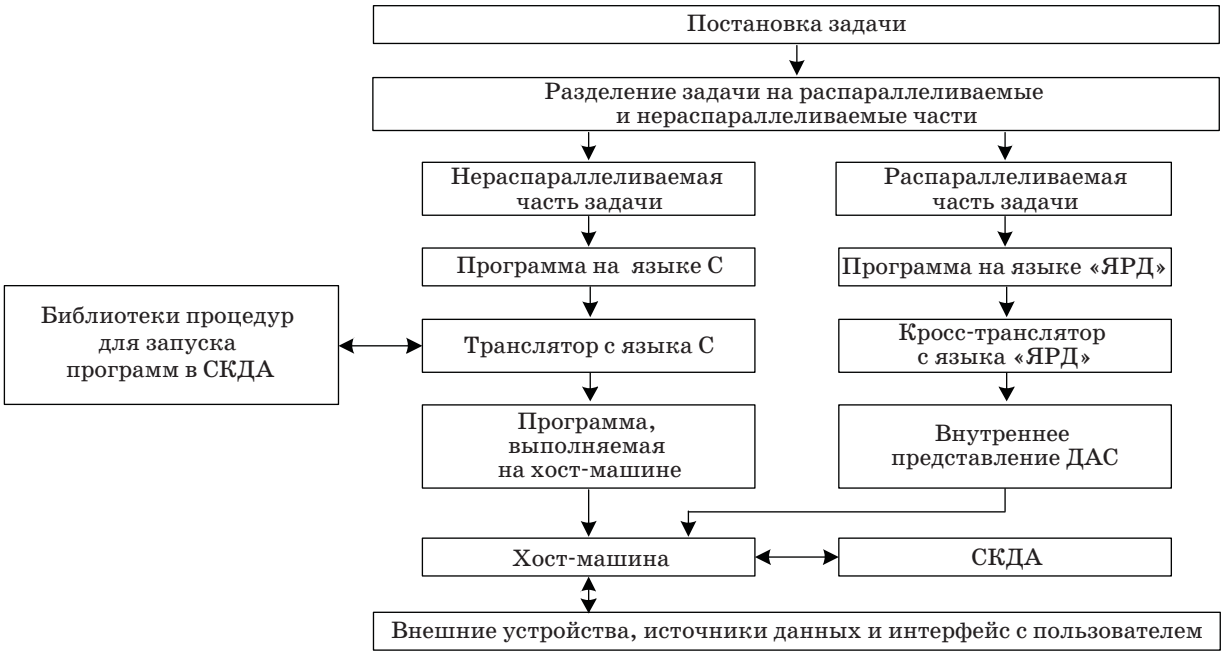
В целом идеология языка «ЯРД» соответствует принципам объектно-ориентированного программирования, т. е. каждый ДА в составе ДАС является, по сути, объектом, содержащим некоторые структуры данных и функции, оперирующие с этими данными, а также с данными, содержащимися в «смежных» ресурсных автоматах.

Гибридное программирование

Анализ структуры множества задач (и соответствующих программ), традиционно решаемых при помощи суперкомпьютеров, приводит к выводу, что значительная часть фрагментов (блоков, модулей, процедур, операторов) каждой из таких программ либо вообще не требует распараллеливания (и высокого уровня производительности), либо их распараллеливание принципиально невозможно. К первым относятся, например, все подготовительные или заключительные вычисления и другие действия, в том числе операторы ввода-вывода, выполняемые один раз (или ограниченное число раз) в процессе работы программы. Ко вторым относятся, например, итерационные циклы, поскольку вычисления в каждом таком цикле зависят от результатов вычислений в предыдущих циклах.

Разумеется, любой суперкомпьютер, в том числе и СКДА, принципиально может выполнить и эти вычисления или прочие действия, однако задействование мощностей многопроцессорного суперкомпьютера для выполнения одиночных или нераспараллеливаемых действий, в то время как значительная часть аппаратуры суперкомпьютера будет в этом случае простаивать, представляется нецелесообразным. Это, кстати, является одной из причин того, что большинство существующих суперкомпьютеров на реальных задачах показывают существенно (во много раз) меньшую производительность, чем на специально отобранных тестовых задачах (например, таких, как общеизвестный тест Linpack).

Технология решения задач на основе «гибридного программирования» представлена на рис. 4.



■ ***Рис. 4.*** Технология решения задач на основе «гибридного программирования»

В связи с вышеизложенным возникает идея «гибридного программирования». Суть этой идеи заключается в том, что значительная часть программы, не требующая распараллеливания (или для которой распараллеливание невозможно) программируется на обычных языках программирования (таких как C, C++, Pascal) и выполняется на хост-машине. Для иллюстрации в качестве примера выбран язык C. Те части программы, которые требуют высокой производительности и распараллеливания, отдельно программируются на языке «ЯРД», преобразуются с помощью соответствующего транслятора во внутреннее представление ДАС, а в основную программу, исполняемую на хост-машине, включаются обращения к процедурам из некоторой стандартной библиотеки, которые запускают соответствующие программы, представленные в виде ДАС, в СКДА и возвращают в хост-машину результаты.

Следует отметить, что хотя термин «гибридное программирование» впервые появился именно в данной статье, но соответствующая методика решения задач нами уже применялась неоднократно. Например, в середине 90-х годов она использовалась для решения на макетном образце СКДА задачи моделирования газодинамических потоков в реактивном двигателе, задач фрактальной графики и задачи трехмерного отображения движущихся объектов.

Заключение

Предложенные в статье аппаратно-программные методы реализации суперкомпьютеров с динамической архитектурой (СКДА) и технологии решения сложных задач в СКДА, основанные на теории динамических автоматных сетей, являются полностью оригинальными и не имеют аналогов в мире. Предложенный язык программирования «ЯРД» также является оригинальным.

Впервые сформулирована идея «гибридного программирования», предполагающая разделение решаемой задачи на распараллеливаемые и нераспараллеливаемые фрагменты, выполняемые при помощи различных аппаратных и программных средств.

Важнейшим результатом работы является возможность избавить программиста при решении сложных задач на суперкомпьютере от необходимости заботиться о распараллеливании вычислительных процессов и учитывать такие аппаратные характеристики, как количество процессоров или объемы памяти, поскольку реализация и работа динамических автоматных сетей предполагает полностью автоматическое распараллеливание и учет ресурсов.

Важным является также «неограниченная масштабируемость» аппаратуры СКДА, позволяющая наращивать вычислительную мощность СКДА по мере надобности, не изменяя программного обеспечения.

Главным практическим результатом является возможность создания СКДА (включая и возможность создания отечественной элементной базы для СКДА), характеристики которого значительно превосходят характеристики существующих суперкомпьютеров по габаритам, стоимости и энергопотреблению, а также технологии разработки соответствующих программ для решения сложных задач.

Литература

1. **Turing A. M.** On Computable Numbers with an Application to the Enstcheiaungsproblem // Proc. London. Math. Soc. 1937. Vol. 43. P. 230–265.
2. **Майерс Г.** Архитектура современных ЭВМ: в 2 кн. — М.: Мир, 1985. Кн. 1. — 364 с.
3. **Торгашев В. А.** Автоматные сети и компьютеры: история развития и современное состояние // История информатики и кибернетики в Санкт-Петербурге (Ленинграде): сб. — СПб.: Наука, 2012. С. 46–66.
4. **J. von Neuman.** Theory of Self-reproducing Automata. — Urbana and London: University of Illinois Press, 1966. — 403 p.
5. **Ulam S. M.** Random Processes and Transformation // Proc. of the Intern. Congress of Mathematicians. Providence. 1950. Vol. 11. P. 264–275.
6. **Барзинь Ю. М.** Проблема универсальности растущих автоматов// Докл. АН СССР. 1964. Т. 57. № 3. С. 542–545.
7. **Торгашев В. А.** Динамические автоматные сети // Тр. СПИИРАН. 2013. Вып. 4(27). С. 23–34.
8. **Торгашев В. А., Царев И. В.** Средства организации параллельных вычислений и программирования в мультипроцессорах с динамической архитектурой// Программирование. 2001. № 4. С. 53–68.
9. **Торгашев В. А., Царев И. В.** Семейство суперкомпьютеров с динамической архитектурой — концептуальные основы // Искусственный интеллект. 2009. № 3. С. 251–257.
10. **Торгашев В. А., Царев И. В.** Суперкомпьютерные технологии на базе динамических автоматных сетей // Суперкомпьютерные технологии: разработка, программирование, применение (СКТ-2010): материалы Междунар. науч.-техн. конф. Таганрог: Изд-во ТТИ ЮФУ, 2010. Т. 1. С. 161–165.
11. **Торгашев В. А., Царев И. В.** Динамические автоматные сети как модель параллельных вычислений в мультипроцессорах с динамической архитек-

турой // Вестник компьютерных и информационных технологий. 2009. № 3. С. 11–20.
12. **Царев И. В.** ЯРД — язык сетевого программирования в распределенных вычислительных системах с динамической архитектурой // Искусственный интеллект. 2008. № 3. С. 761–770.

UDC 681.3:681.3.06:681.3.01:681.3.068
doi:10.15217/issn1684-8853.2015.6.57

Technologies of Solving Complicated Problems on the Base of Dynamic Automata Networks

Torgashev V. A.[a], Dr. Sc., Tech, Professor
Tsarev I. V.[a], Leading Programmer, civ@iias.spb.su
[a]Saint-Petersburg Institute for Informatics and Automation of RAS, 39, 14 Line, V. O., 199178, Saint-Petersburg, Russian Federation

Purpose: Modern supercomputers used for sophisticated problems have a number of imperfections concerning both their technical and economic characteristics (size, power consumption, cost) and the complexity of solving real programming problems when special methods are required to make the programs run concurrently. As a result, the actual performance of supercomputers when solving real problems is significantly lower than their theoretical performance. The goal of this research is the development of methods for creating supercomputers and programming technologies based on the theory of dynamic automata networks, which would considerably improve specific characteristics of supercomputers and simplify parallel programming for solving the corresponding tasks. **Results:** Fundamental principles have been formulated for creating dynamic architecture supercomputers based on dynamic automata networks, including the implementation of dynamic automata using either Field Programmable Gate Array or specially developed domestic chips. This provides extremely high regularity of the chip structure, considerably simplifying the development of supercomputers with dynamic architecture compared to the conventional supercomputers. Technologies have been proposed for solving complicated problems using a specially developed programming language based on dynamic automata networks, along with the method of "hybrid programming" which allows you to combine different hardware and software means to solve the same problem. **Practical relevance:** The proposed methods provide the possibility to create supercomputers with dynamic architecture which greatly (ten- or hundredfold) surpass the traditional supercomputers by such specific (calculated for a unit of the performance) characteristics as size, power consumption and cost. At the same time, the supercomputer hardware structure becomes much simpler. The proposed programming technologies for solving complicated problems, including "hybrid programming", simplify the very process of parallel programming and increase the efficiency of solving complicated problems on supercomputers.

Keywords — Dynamic Automata Networks, Parallel Programming, Supercomputers with Dynamic Architecture.

References

1. Turing A. M. On Computable Numbers with an Application to the Enstcheiaungsproblem. *Proc. London. Math. Soc.*, 1937, vol. 43, pp. 230–265.
2. Mayers G. Dj. *Advances in Computer Architecture*. Moscow, Mir Publ., 1985, vol. 1. 364 p. (In Russian).
3. Torgashev V. A. Automata Networks: History of Development and Contemporary State. *Istoriia informatiki i kibernetiki v Sankt-Peterburge (Leningrade)* [History of Informatics and Cybernetics in Saint-Petersburg (Leningrad)]. Saint-Petersburg, Nauka Publ., 2012, vol. 3, pp. 46–66 (In Russian).
4. J. von Neuman. *Theory of Self-reproducing Automata*. Urbana and London, University of Illinois Press, 1966. 403 p.
5. Ulam S. M. Random Processes and Transformation. *Proc. of the Intern. Congress of Mathematicians*, Providence, 1950, vol. 11, pp. 264–275.
6. Barzin U. M. The Problem of Universality of Growing Automata. *Doklady AN SSSR*, 1964, vol. 57, no. 3, pp. 542–545 (In Russian).
7. Torgashev V. A. Dynamic Automata Networks. *Trudy SPIIRAN*, 2013, vol. 4(27), pp. 23–34 (In Russian).
8. Torgashev V. A., Tsarev I. V. Means for Organization of Parallel Computations and Programming in Multiprocessors with Dynamic Architecture. *Programmirovanie*, 2001, no. 4, pp. 53–68 (In Russian).
9. Torgashev V. A., Tsarev I. V. A Family of Supercomputers with Dynamic Architecture — Conceptual Basis. *Iskusstvennyi intellekt* [Artificial Intelligence], 2009, no. 3, pp. 251–257 (In Russian).
10. Torgashev V. A., Tsarev I. V. Supercomputer Techbnologies on the Base of Dynamic Automata Networks. *Materialy Mezhdunarodnoi nauchno-tekhnicheskoi konferentsii "Superkomp'iuternye tekhnologii: razrabotka, programmirovanie, primenenie" (SKT-2010)* [Proceedings of International Scientific and Technical Conference "Supercomputer technologies: Development, Programming, Application" (SCT-2010)]. Taganrog, Taganrogskii tekhnologicheskii institut Iuzhnogo federal'nogo universiteta Publ., 2010, vol. 1, pp. 161–165 (In Russian).
11. Torgashev V. A., Tsarev I. V. Dynamic Automata Networks as a Model of Parallel Computations in the Multiprocessors with Dynamic Architecture. *Vestnik komp'uternykh i informatsionnykh technologii* [Herald of Computer and Information Technologies], 2009, no. 3, pp. 11–20 (In Russian).
12. Tsarev I. V. YARD — a Language for Network Programming in Distributed Computing Systems with Dynamic Architecture. *Iskusstvennyi intellect* [Artificial Intelligence], 2008, no. 3, pp. 761–770 (In Russian).