ISSN 1814-1196
Научный вестник НГТУ
том 72, № 3, 2018, с. 107–120

http://journals.nstu.ru/vestnik
Science Bulletin of the NSTU
Vol. 72, No. 3, 2018, pp. 107–120

ИНФОРМАТИКА,
ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
И УПРАВЛЕНИЕ

INFORMATICS,
COMPPUTER ENGINEERING
AND CONTROL

УДК 681.3.01:681.3.068 DOI: 10.17212/1814-1196-2018-3-107-120

Система имитационного моделирования динамических автоматных сетей*

И.В. ЦАРЕВ

199178, РФ, г. Санкт-Петербург, 14-я линия В.О., 39, Санкт-Петербургский институт информатики и автоматизации Российской академии наук

civ@iias.spb.su

Данная работа посвящена отдельным аспектам создания программной системы имитационного моделирования сложных вычислительных систем с сетевой структурой, основанных на модели параллельных вычислений, известной как «динамические автоматные сети». В число таких систем входят мультипроцессоры и суперкомпьютеры с динамической архитектурой. Система имитационного моделирования динамических автоматных сетей позволяет на этапе разработки исследовать работоспособность структуры таких систем, а также структур, функций и алгоритмов работы отдельных компонентов системы, осуществлять исследование проектируемой системы с целью доказательства ее работоспособности, а также оценки основных характеристик системы. В результате данной работы разработана структура системы имитационного моделирования, предложен неалгоритмический язык для описания моделируемых систем и решаемых на них задач, транслятор для преобразования описания системы во внутреннее представление, используемое в качестве входной информации как для программы имитационного моделирования, так и для реальной разрабатываемой аппаратуры. Разработаны алгоритмы работы транслятора, позволяющие осуществлять настройку транслятора на конкретную версию языка с возможностью изменения синтаксиса и семантики языка. Разработаны основные алгоритмы и структуры данных системы моделирования. Отмечены основные проблемы создания такой системы имитационного моделирования и определены возможные способы решения этих проблем. Практическая значимость предложенных решений состоит в возможности исследования работоспособности и характеристик сложных вычислительных систем с сетевой структурой, создаваемых на основе динамических автоматных сетей. Разработанная система имитационного моделирования позволяет определить работоспособность проектируемой системы и оценить ее основные характеристики.

Ключевые слова: динамические автоматные сети, автотрансформация, параллельные вычисления, суперкомпьютеры с динамической архитектурой, имитационное моделирование, неалгоритмические языки программирования, трансляторы, алгоритмы моделирования автоматных сетей

* *Статья получена 14 марта 2018 г.*

ВВЕДЕНИЕ

Создание высокопроизводительных вычислительных систем, называемых также «суперкомпьютерами», является одним из важнейших направлений развития современной вычислительной техники. Самые мощные современные суперкомпьютеры достигли петафлопного диапазона производительности, т. е. могут теоретически выполнять 10^{18} и более операций в секунду. Однако подавляющее большинство таких суперкомпьютеров имеет ряд общих недостатков, к которым можно отнести прежде всего высокое энергопотребление, значительные геометрические размеры и крайне высокую стоимость (обычно эти характеристики представляются как удельные, т. е. в расчете на единицу производительности). Одной из основных причин этого являются устаревшие подходы к проектированию архитектуры таких вычислительных средств.

В связи с этим постоянно предпринимаются попытки решить эти проблемы за счет принципиальных изменений в архитектуре как подобных систем в целом, так и отдельных процессоров, входящих в их состав. Примеров таких попыток существует огромное количество. В качестве одного из многочисленных примеров можно привести концепцию, приведенную в статье Ю.И. Митропольского [1]. Однако эта концепция, как и многие другие, всё же основывается на традиционной архитектуре вычислительных средств (процессоров или их множеств).

Более перспективными представляются работы, основанные на программируемой (или реконфигурируемой) архитектуре, наиболее яркими из которых представляются работы НИИ МВС ЮФУ (НИИ многопроцессорных вычислительных систем Южного федерального университета, г. Таганрог), описания которых можно найти в работах [2, 3]. В данном случае реконфигурация вычислительной системы осуществляется при помощи соответствующих программных средств однократно, перед решением задачи. В процессе решения задачи конфигурация системы не меняется, впрочем, и эти системы по удельным характеристикам оказываются существенно более эффективными, чем большинство существующих суперкомпьютеров, прежде всего за счет экономии аппаратных средств, настраиваемых на решение конкретной задачи.

Более перспективным для повышения эффективности суперкомпьютеров является использование идеи машин (мультипроцессоров) с динамической архитектурой (МДА), основанных на концепции динамических автоматных сетей, рассматриваемой ниже.

Проектирование вычислительных систем требует на разных стадиях исследования их работоспособности и прогнозируемых характеристик посредством программного моделирования.

Имитационное моделирование является общепринятым методом исследования любых вычислительных систем, традиционно применяемым на различных этапах разработки их структуры и проектирования конкретных аппаратных устройств любой сложности. Тем более применение этих методов является крайне важным при разработке современных сложных многопроцессорных систем, таких как суперкомпьютеры, а также любых систем с сетевой

структурой (например, мультипроцессоров и суперкомпьютеров с динамической архитектурой).

Целью данной работы является разработка структуры, методов и алгоритмов программной системы имитационного моделирования, позволяющей моделировать различные вычислительные системы с параллельной обработкой информации, включая мультипроцессоры (и суперкомпьютеры) с динамической архитектурой (МДА, СКДА).

Динамические автоматные сети (ДАС) – это разработанная в СПИИРАН в 80-х годах прошлого века мощная и универсальная модель любых вычислений с параллельными вычислениями любой степени вычислительной сложности. На основе модели ДАС был разработан и реализован целый ряд МДА и СКДА, некоторые из которых были успешно испытаны (включая государственные испытания МДА ЕС-2704 в конце 80-х годов) на различных задачах и применялись в нескольких научных и научно-технических организациях СССР и РФ.

Концепция ДАС основана отчасти на теории самовоспроизводящихся автоматов фон-Неймана [4], а также частично на работах Улама [5] и Бардзиня [6], но в целом является вполне самостоятельной.

Динамическая автоматная сеть включает в себя теоретически неограниченное количество динамических автоматов (ДА) с произвольным набором вычислительных функций и практически любой конфигурацией связей между ними. Эти ДА соединены посредством связей в сеть с некоторой заранее определенной начальной конфигурацией. Любой ДА принадлежит к одному из двух классов – «операторы» и «данные» (автоматы последнего класса также называются ресурсными автоматами), также ДА могут быть «структурными автоматами», представляющими собой подсети, копии которых с соответствующими изменениями могут порождаться в процессе вычислений.

В общем случае в процессе вычислений конфигурация ДАС постоянно изменяется, порождаются новые автоматы и связи, в начале вычислений количество ДА увеличивается, позднее – уменьшается, поскольку автоматы, выполнившие свою функцию, уничтожаются, в чём, собственно, и состоит «динамика» автоматной сети. Этот процесс называется автотрансформацией динамической автоматной сети. В конце вычислительного процесса в ДАС остаются только ДА класса «данные», совокупность и структура которых и представляет собой результат вычислений. В то же время возможны и структуры ДАС, которые не уничтожаются в процессе вычислений, постоянно получают новые данные на входах ДАС и вырабатывают новые результаты. Такие ДАС могут, например, использоваться в системах обработки сигналов, в системах обработки телеметрической информации, в системах искусственного интеллекта и во множестве других систем. Более подробное изложение теоретических и практических аспектов ДАС, включая и историю создания, можно найти в работах [7–12].

Теория ДАС не накладывает никаких ограничений на набор функций операторных автоматов, а гибкость рассматриваемой в данной работе системы имитационного моделирования ДАС позволяет вводить новые вычислительные функции автоматов по мере необходимости. Существенным свойством ДАС, которое может быть использовано как для разработки

МДА/СКДА, так и других вычислительных систем с сетевой архитектурой, является возможность представления массивов или любых других сложных структур из автоматов.

В течение последних двадцати пяти лет никаких работ в области ДАС и МДА, кроме работ сотрудников СПИИРАН, нигде в мире не выполнялось, хотя эти работы широко известны, и никаких принципиально новых решений в этой области предложено не было. Этим объясняется отсутствие в данной статье ссылок на работы, кроме вышеупомянутых.

Данная статья посвящена рассмотрению некоторых аспектов разработки системы программного моделирования ДАС (СИМДАС), включая общую структуру системы и некоторые основные алгоритмы работы системы, а также основных проблем, возникающих при ее создании. Поскольку СИМДАС включает в себя довольно разнородные и сложные компоненты, рассмотреть их подробно в рамках одной статьи представляется невозможным и может быть предметом нескольких отдельных статей. Так что настоящую статью можно считать «постановочной», подразумевая дальнейшую серию публикаций. Кроме того, разработка и отладка самой программы СИМДАС на данный момент еще не завершена, так что возможно появление в дальнейшем дополнительных проблем, не рассмотренных здесь.

1. ПОСТАНОВКА ЗАДАЧИ

Одной из основных задач настоящей работы является разработка структуры системы моделирования ДАС и различных вычислительных средств на их основе, в том числе высокопроизводительных вычислительных систем – мультипроцессоров и суперкомпьютеров с динамической архитектурой (МДА, СКДА), а также разработка соответствующих программ.

Такая система позволит показать возможность представления СКДА в виде динамической автоматной сети, провести эксперименты на основе имитационного моделирования ДАС, а также оценить временные и другие характеристики ДАС для их дальнейшей практической реализации.

Анализ задачи показал, что помимо собственно рассматриваемой программы имитационного моделирования (симуляции) необходимы также средства для построения исходной ДАС-программы в формате, который является входным для программы моделирования, а также для проектируемой аппаратуры МДА/СКДА. Под ДАС-программой (или ДАС-файлом) понимается некоторая структура данных, соответствующая внутреннему представлению начального состояния ДАС, предназначенной для решения некоторой конкретной задачи, наряду с числовыми данными. Таким образом, задача сводится не только к созданию собственно программы моделирования, но и некоторого входного языка и транслятора с него в коды, потребляемые этой программой в качестве входных данных. В результате становится необходимым создание некоторой системы программирования, а не только самой моделирующей программы. Структура такой системы рассмотрена ниже.

2. СТРУКТУРА СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДАС

В состав системы имитационного моделирования ДАС (СИМДАС) входят следующие компоненты:

- язык программирования ЯРД-2017;
- транслятор с языка программирования ЯРД-2017 во внутреннее представление ДАС (ДАС-файл);
- программа моделирования ДАС.

На рис. 1 представлена структура СИМДАС в виде, аналогичном тому, как показана технология решения задач на основе ДАС, изображенная в статье [13] на рис. 3. Как видно из этих рисунков, общая схема работы с СИМДАС отличается от этой технологии прежде всего тем, что вместо аппаратной реализации МДА (СКДА) используется моделирующая программа.

3. ЯЗЫК ПРОГРАММИРОВАНИЯ ЯРД-2017 И КОМПИЛЯТОР

Язык программирования ЯРД-2017 разработан на основе языка ЯРД, который впервые был сформулирован в середине 90-х годов, тогда же был создан и первый транслятор с этого языка, в 2007–2008 годах язык был несколько скорректирован. Краткое описание этой версии языка ЯРД приведено в статье [14]. Ряд конструкций языка был ориентирован на структуры аппаратных реализаций МДА, которые существовали в то время. В частности, одной из особенностей тех реализаций была возможность программировать исполнительные, управляющие и коммутационные функции динамических автоматов. Это нашло свое отражение и в некоторых структурах языка, предполагавшего максимальную гибкость системы программирования. Это не влияло принципиально на синтаксис самого языка, но влияло на представление некоторых структур данных и конструкций, определяющих функции операторов, которые, в частности, могли быть представлены на уровне аппаратной реализации. Современная концепция аппаратной реализации предполагает существенное упрощение структуры и функций ДА (см., например, статью [15]), что, в свою очередь, отражается и на языке.

Разработанный в рамках настоящей работы язык программирования ЯРД-2017, сохраняя основные принципы и синтаксический стиль исходного языка ЯРД, был существенно упрощен. Рассмотрение синтаксиса и семантики языка не входит в рамки этой статьи, а список упрощений языка весьма объемный, что также не позволяет его вместить в объем статьи. Однако некоторые особенности новой версии языка приведены в [13].

Следует упомянуть некоторые характеристики транслятора (компилятора) с языка ЯРД-2017 в ДАС-программу. Структура компилятора показана на рис. 2. Как и в любом трансляторе, в нем имеются типовые основные части – лексический анализатор, синтаксический анализатор и генератор кода. Лексический анализатор основан на традиционных алгоритмах разбора автоматной грамматики, но позволяет настраивать его на конкретный вариант лекси-

ческой части грамматики. Синтаксический анализатор построен на основе модифицированного алгоритма Эрли [16]. Модификация этого широко известного алгоритма состоит в значительной оптимизации использования оперативной памяти, экономится до 80 % используемой памяти по сравнению с классической формой алгоритма. Синтаксический анализатор, так же как и лексический, допускает настройку на конкретный вариант грамматики языка, последняя описывается в традиционном виде – в модифицированной форме Бэкуса-Наура (МБНФ). Настройка компилятора на конкретную грамматику осуществляется специальными утилитами, преобразующими текстовое представление грамматики во внутреннюю форму, которая и используется для настройки анализаторов. Синтаксический анализатор строит дерево разбора, а генератор кода преобразует его в ДАС-программу. Это является достаточно простой задачей, поскольку структура ДАС-программы является сетевой и, как правило, имеет вид дерева. К сожалению, объем статьи не позволяет подробно рассмотреть структуру ДАС-программы. Следует отметить, что по аналогии со структурой самой ДАС, она представляет собой множество расположенных в произвольном порядке дескрипторов ДА и структур данных, представляющих собой тела ДА и связанных между собой различными связями (ссылками на положение в файле других дескрипторов). Тело структурного оператора представляет собой массив ссылок на дескрипторы узлов, составляющих некоторую подсеть, реализующую функции структурного оператора. ДАС-программа (ДАС-файл) является основной входной информацией для программы имитационного моделирования ДАС наряду с массивами обрабатываемых данных.

4. АЛГОРИТМЫ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКОГО АВТОМАТА

Программа моделирования разработана на языке программирования Borland Pascal With Objects в среде программирования Delphi-7. В программе широко используются методы модульного и объектно ориентированного программирования.

Рассмотрим два основных алгоритма работы СИМДАС, определяющих работу динамических автоматов и, как следствие, ДАС в целом. Но прежде нужно привести описание структуры дескриптора ДА в таком виде, в каком оно присутствует в программе СИМДАС – дескриптор ДА представляется как объект (класс). Назначение полей структуры и методов объекта (процедур и функций) указано в комментариях. В целом описываемая структура соответствует той, которая изображена на рис. 2 в статье [13].

```
Type
  RefDA = ^TDADescriptor; {ссылка на дескриптор ДА}
  TDADescriptor = class {описание дескриптора ДА}
    TypeAndStatus: DoubleWord; {тип и статус ДА}
    BodySize: Integer; {размер тела ДА}
    Body: DoubleWord;
    Master: RefDA; {ссылка на «хозяина» ДА}
    NumberInMaster: Integer; {номер данного ДА в теле «хозяина»}
```

```
        ArgResNumber:    DoubleWord;    {количество    аргумен-
тов/результатов}
        Arg1: RefDA; {ссылка на дескриптор первого аргумента}
        Arg1Parameter1: DoubleWord; {параметры аргумента}
        Arg1Parameter2: DoubleWord;
        Arg2: RefDA; {ссылка на дескриптор второго аргумента}
        Arg2Parameter1: DoubleWord;
        Arg2Parameter2: DoubleWord;
        Res: RefDA; {ссылка на дескриптор результата}
        ResParameter1: DoubleWord; {параметры результата}
        ResParameter2: DoubleWord;
        procedure Create(Source: RefDA; FreeDA_Pool: TList);
    {процедура создания ДА}
        procedure Destroy(Source: RefDA; FreeDA_Pool: TList);
    {процедура уничтожения ДА, выполнившего свою функцию}
        procedure  Control;  {процедура  выполнения  управляющей
функции ДА}
        function CheckReady : Boolean; {функция проверки го-
товности ДА}
        function CheckUndefined : Boolean; {функция проверки
состояния неопределённости ДА}
        procedure Execute(var Successful: Boolean); {процедура
выполнения исполнительной (вычислительной) функции ДА}
        procedure GenerateSubnet (Node: RefDA); {процедура со-
здания (генерации) подсети структурного оператора}
        procedure  GetStatus  (var  DA_Status  :  DoubleWord);
{процедура получения статуса (состояния) ДА}
        procedure ChangeStatus (var DA_Status : DoubleWord);
{процедура изменения статуса (состояния) ДА}
        procedure GetType (var DA_Type : DoubleWord); {проце-
дура получения типа ДА}
        procedure ChangeType (var DA_Type : DoubleWord); {про-
цедура изменения типа ДА}
        procedure  SetToControlQueue  (ControlQueue  :  TQueue);
{процедура постановки ДА в очередь к управляющей подсистеме}
        procedure GetFromControlQueue (ControlQueue : TQueue);
{процедура извлечения ДА из очереди к управляющей подсистеме}
        procedure SetToExecQueue (ExecQueue : TQueue); {проце-
дура постановки ДА в очередь к исполнительной подсистеме}
        procedure GetFromExecQueue (ExecQueue : TQueue); {про-
цедура извлечения ДА из очереди к исполнительной подсистеме}
    end;
```

Отбросив многочисленные вспомогательные функции системы, а также функции компилятора, рассмотрим основные алгоритмы работы модуля, который определяет работу программы СИМДАС в целом. Следует отметить, что работа этой основной части системы осуществляется с помощью системы очередей, реализуемых как кольцевые двусторонние списки. Основными являются очереди к управляющей и исполнительной подсистемам.

При работе модуля (по сути это основная часть программы моделирования ДАС) выполняются следующие шаги алгоритма.

Шаг 1. Создание списка (пула) свободных автоматов FreeDA_Pool. При этом используется параметр «количество свободных автоматов», задаваемый в настройках программы СИМДАС, и выделяется соответствующая память. Если программа работает в режиме использования виртуальной памяти, то создается образ соответствующей структуры данных на жестком диске.

Шаг 2. Ввод исходного ДАС-файла (файла ДАС-программы).

Шаг 3. Для каждого дескриптора из ДАС-файла в цикле выполняется создание множества динамических автоматов при помощи процедуры Create. Если дескриптор описывает некоторый массив данных или структурный оператор, то выделяется память для тел соответствующих объектов (динамических автоматов). Каждый созданный автомат ставится в очередь к управляющей подсистеме.

Шаг 4. Основной цикл работы программы СИМДАС. Рассматривается отдельно (ниже).

Шаг 5. Завершение работы программы. Выполняется в случае, если в ДАС не остается ни одного активного элемента (оператора), т. е. очереди к управляющей и исполнительной системам пусты.

Шаг 6. Освобождение памяти всех структур данных – списка свободных автоматов, очередей и данных.

Теперь рассмотрим работу основного цикла программы (шаг 4 предыдущего алгоритма). Основная часть алгоритма работы ДАС выполняется в цикле, выполнение которого заканчивается только при отсутствии активных элементов в ДАС, либо в случае критической ошибки (аварийное завершение). Однако существуют случаи, когда цикл принципиально не заканчивается (а в случае критической ошибки предпринимаются некоторое специальные действия, не прерывающие выполнения программы). К таким случаям относятся, например, выполнение задачи обработки сигналов и других задач, упомянутых выше.

В реальной аппаратуре МДА/СКДА все функции всех динамических автоматов выполняются параллельно и одновременно, включая управляющие и исполнительные функции каждого автомата. В моделирующей программе, которая в принципе является последовательной, использование многопоточного выполнения программы (многоядерного программирования) принципиального значения не имеет, это лишь увеличивает производительность программы, роль управляющих и исполнительных функций автоматов выполняют соответствующие подсистемы программы (управляющая и исполнительная). Для имитации параллельного исполнения в цикле выполняется сначала некоторая последовательность действий управляющей подсистемы (шаги 1–6 нижеописанного алгоритма), а затем некоторая последовательность действий исполнительной подсистемы (шаги 7–12 нижеописанного алгоритма).

Таким образом, в этом потенциально бесконечном цикле выполняются следующие действия.

Шаг 1. Извлечение очередного объекта (автомата, а точнее, его дескриптора) из очереди к управляющей подсистеме.

Шаг 2. Определение состояния извлеченного объекта.

Шаг 3. Если объект не готов, то пропускаются шаги 4–6, а сам объект ставится снова в очередь к управляющей подсистеме (в конец очереди).

Шаг 4. Если объект готов, то определяются состояния связанных с ним объектов.

Шаг 5. Если аргументы данного оператора готовы, а результат находится в неопределенном состоянии, то объект ставится в очередь к исполнительной подсистеме.

Шаг 6. Объект удаляется из входной очереди (сдвигается соответствующий указатель очереди).

Шаг 7. Извлечение очередного объекта (автомата, а точнее, его дескриптора) из очереди к исполнительной подсистеме.

Шаг 8. Если объект структурный, то выполняется порождение подсети, описание которой имеется в описании соответствующего структурного объекта (соответствующие структурные дескрипторы имеются в исходном ДАС-файле), т. е. выполняется процедура GenerateSubnet. При этом пропускаются шаги 9–12. Все порожденные объекты ставятся в очередь к управляющей подсистеме.

Шаг 9. Выполняется процедура Execute текущего объекта. Она представляет собой большой оператор case, селектором которого является код операции данного оператора. В каждой ветви оператора case производятся вычисления, соответствующие коду операции данного оператора.

Шаг 10. Результат операции записывается в дескриптор или в тело автомата, соответствующего результату. Запись в тело подразумевает запись в соответствующий элемент массива, являющегося результатом, в соответствии с индексами, указанными в дескрипторе оператора. Если в качестве результата выступает другой оператор, то данный оператор изменяет свой класс на класс «данные» (выполняется процедура ChangeType). Результату присваивается состояние готовности, и он ставится в очередь к управляющей подсистеме..

Шаг 11. Уничтожаются объекты, являющиеся аргументами данного оператора, если они не являются также аргументами других операторов.

Шаг 12. Если тип оператора не был изменен на тип данных и оператор не имеет признака «статический», то он уничтожается.

Шаг 13. Если в обеих очередях к управляющей и исполнительной подсистемам не содержится ни одного объекта (автомата) либо если произошла критическая ошибка, то устанавливается признак завершения цикла (цикл завершается).

Следует отметить, что вышеприведенные описания алгоритмов являются крайне упрощенными и предназначены лишь для приблизительного понимания принципов работы разрабатываемой программы имитационного моделирования ДАС.

5. ОСНОВНЫЕ ПРОБЛЕМЫ МОДЕЛИРОВАНИЯ ДАС

Можно выделить две основные проблемы, от решения которых зависит эффективность работы системы, а также возможность исследования некоторых характеристик моделируемых МДА и СКДА на основе ДАС.

Первая проблема заключается в количественных характеристиках моделируемых систем МДА (СКДА).

При разработке программы моделирования учитывались ранее полученные результаты анализа проблем моделирования вычислений ДАС, являющихся принципиально параллельными и многопроцессорными. Имеется в виду, что под «множеством процессоров» на самом деле понимается множество элементарных динамических автоматов ДА. При этом следует учесть, что количество ДА в СКДА может быть весьма велико – несколько миллионов и миллиардов.

В то же время в обычных условиях исследователи и разработчики таких систем не имеют доступа к мощным суперкомпьютерам и вынуждены использовать для моделирования пусть мощные, но персональные компьютеры. Например, в распоряжении автора настоящей статьи имеется компьютер на базе четырехъядерного процессора Intel Core i7 с тактовой частотой 2.66 гигагерца, тремя гигабайтами оперативной памяти и двумя жесткими дисками общим объемом 3 терабайта. Таким образом, мы вынуждены моделировать принципиально параллельные вычислительные процессы в среде последовательно работающих вычислительных средств. Использование многоядерных процессоров (например, вышеупомянутого процессора Intel Core i7, позволяющего с учетом гипертрейдинга организовать одновременное выполнение восьми потоков) не меняет принципиально ситуации, лишь несколько повышая производительность.

Таким образом, данная проблема сводится к обработке очень больших объемов информации и, учитывая последовательное выполнение вычислений, весьма большого времени проведения экспериментов. Первое приводит к необходимости использования виртуальной памяти, а второе – к необходимости периодического прерывания моделирования с сохранением полного текущего состояния системы. Виртуальная память позволяет без особых дополнительных средств решить проблему постоянного сохранения текущего состояния модели с целью возможной приостановки процесса моделирования и последующего его возобновления. В то же время организация виртуальной памяти должна быть осуществлена в рамках данной программы, по возможности без использования соответствующих механизмов, поддерживаемых операционной системой, поскольку управление последними крайне затруднено и неэффективно.

Вторая проблема заключается в оценке некоторых характеристик моделируемой системы и прежде всего – временных характеристик. В вычислительной модели ДАС не существует общего времени, каждый автомат имеет собственное время, а синхронизация работы всего множества автоматов осуществляется по событиям, главными из которых являются события изменения статуса (состояния) отдельных динамических автоматов, в результате чего активируются автоматы, связанные с ними. Для измерения временных характеристик моделируемой системы необходим некоторый механизм привязки собственного времени автомата к общей шкале реального времени. То есть каждый автомат должен иметь свой счетчик времени, но привязанный к общей шкале времени. При возникновении некоторого события автомат копирует текущее глобальное время и отсчитывает свое собственное время от него, добавляя к нему время выполнения собственной вычислительной функции. Таким образом, все автоматы, инициализация работы которых

привязана к данному событию, будут учитывать одно и то же время, хотя в последовательной моделирующей программе эта работа выполняется в разные моменты времени.

Обе эти проблемы вполне решаемы, но объем статьи не позволяет рассмотреть их более подробно.

ЗАКЛЮЧЕНИЕ

Главным практическим результатом работы является разработка системы имитационного моделирования ДАС (СИМДАС), дающая возможность исследования работоспособности и оценки характеристик любой сложной вычислительной системы с сетевой структурой, проектируемой на основе динамических автоматных сетей. К таким системам относятся мультипроцессоры, включая суперкомпьютеры, с динамической архитектурой, создаваемые на основе ДАС. Система СИМДАС обладает высокой степенью гибкости, что выражается не только в возможности описания структуры и функций любой такой системы на языке программирования ЯРД-2017, но и в возможности изменения синтаксиса и семантики самого языка в соответствии с особенностями моделируемой вычислительной системы.

СПИСОК ЛИТЕРАТУРЫ

1. *Митропольский Ю.И.* Новые концепции построения вычислительных суперсистем // Труды Физико-технологического института. – 2016. – Т. 25. – С. 22–37.

2. *Каляев А.В.* Многопроцессорные системы с программируемой архитектурой. – М.: Радио и связь, 1984. – 240 с.

3. *Каляев И.А., Левин И.И.* Многопроцессорные вычислительные структуры с динамически реконфигурируемой архитектурой на основе ПЛИС // Сборник научных трудов ИТМиВТ. – М., 2008. – Вып. 1: Материалы конференции «Перспективы развития высокопроизводительных архитектур. История, современность и будущее отечественного компьютеростроения». – С. 44–45.

4. *Neuman J. von.* Theory of self-reproducing automata. – Urbana; London: University of Illinois Press, 1966. – 403 p.

5. *Ulam S.M.* Random processes and transformations // Proceedings of the International Congress of Mathematicians, Cambridge, 1950. – Providence, 1952. – Vol. 2. – P. 264–275.

6. *Барзинь Ю.М.* Проблема универсальности растущих автоматов // Доклады АН СССР. – 1964. – Т. 57, № 3. – С. 542–545.

7. *Торгашев В.А.* Автоматные сети и компьютеры: история развития и современное состояние // История информатики и кибернетики в Санкт-Петербурге (Ленинграде). – СПб.: Наука, 2012. – Вып. 3. – С. 46–66.

8. *Торгашев В.А.* Динамические автоматные сети // Труды СПИИРАН. – 2013. – Вып. 4 (27). – С. 23–34.

9. *Торгашев В.А., Царев И.В.* Средства организации параллельных вычислений и программирования в мультипроцессорах с динамической архитектурой // Программирование. – 2001. – № 4. – С. 53–68.

10. *Торгашев В.А., Царев И.В.* Семейство суперкомпьютеров с динамической архитектурой – концептуальные основы // Искусственный интеллект. – 2009. – № 3. – С. 251–257.

11. *Торгашев В.А., Царев И.В.* Суперкомпьютерные технологии на базе динамических автоматных сетей // Суперкомпьютерные технологии: разработка, программирование, примене-

ние» (СКТ-2010): материалы Международной научно-технической конференции. – Таганрог, 2010. – Т. 1. – С. 161–165.

12. *Торгашев В.А., Царев И.В.* Динамические автоматные сети как модель параллельных вычислений в мультипроцессорах с динамической архитектурой // Вестник компьютерных и информационных технологий. – 2009. – № 3. – С. 11–20.

13. *Торгашев В.А., Царев И.В.* Технологии решения сложных задач на основе динамических автоматных сетей // Информационно-управляющие системы. – 2015. – № 6. – С. 57–65.

14. *Царев И.В.* ЯРД – язык сетевого программирования в распределенных вычислительных системах с динамической архитектурой // Искусственный интеллект. – 2008. – № 3. – С. 761–770.

15. *Торгашев В.А., Царев И.В.* Реализация суперкомпьютеров с динамической архитектурой на современной элементной базе // Информационно-управляющие системы. – 2016. – № 6. – С. 74–84.

16. *Earley J.* An efficient context-free parsing algorithm // Communications of the ACM. – 1970. – Vol. 13, N 2. – P. 94–102.

Царев Игорь Владимирович, ведущий программист лаборатории информационно-вычислительных систем и технологий программирования Санкт-Петербургского института информатики и автоматизации РАН. Основное направление научных исследований – системы программирования и организация параллельных вычислений в суперкомпьютерах с динамической архитектурой, языки программирования, трансляторы, операционные системы. Имеет более 40 публикаций. E-mail: civ@iias.spb.su

DOI: 10.17212/1814-1196-2018-3-107-120

A system for simulation modeling of dynamic automata networks[*]

I.V. TSAREV

St. Petersburg Institute for Informatics and Automation of RAS, 39, 14th Line, V. O., 199178, St. Petersburg, Russian Federation

civ@iias.spb.su

This paper is devoted to some aspects of creation a program system for simulation modeling of complex computer systems with a network structure based on the model of parallel computations known as “dynamic automata networks”. Such systems as multiprocessors and supercomputers with a dynamic architecture belong to these systems. The system of simulation modeling of dynamic automata networks makes it possible to investigate the operability of the structure of such systems as well as structures, functions and algorithms of individual components of the system on the stage of development, and to assess the main characteristics of the system designed. This research resulted in developing the structure of the system for simulation modeling. In addition, a non-algorithmic programming language to describe simulated systems and problems being solved on such systems is proposed. A compiler to convert the description of the system into an internal representation used as input information both for the program of simulation modeling and for real devices being developed. Algorithms of the compiler permitting us to perform tuning of the compiler for a concrete version of a programming language with a possibility of changing the syntax and semantics of the language are developed. The main algorithms and data structures of the simulation system are developed. General problems of creation of such a system are noted and some possible ways for solving these problems are defined. The practical value of the solutions proposed consists in a possibility of examining the

[*] *Received 14 March 2018.*

operability and characteristics of complex computing systems with the network structure created on the basis of dynamic automata networks. The developed system of simulation modeling permits us both to define the operability of the system being designed and to evaluate its basic characteristics.

Keywords: dynamic automata networks, autotransformation, parallel computations, supercomputers with a dynamic architecture, simulation modeling, non-algorithmic programming languages, compilers, algorithms of modeling automata networks

REFERENCES

1. Mitropol'skii Yu.I. Novye kontseptsii postroeniya vychislitel'nykh supersistem [New concepts for construction of computing supersystems]. *Trudy Fiziko-tekhnologicheskogo instituta*, 2016, vol. 25, pp. 22–37. (In Russian).

2. Kalyaev A.V. *Mnogoprotsessornye sistemy s programmiruemoi arkhitekturoi* [Multiprocessor systems with programmable architecture]. Moscow, Radio i svyaz' Publ., 1984. 240 p.

3. Kalyaev I.A., Levin I.I. [Multiprocessor computing structures with dynamically reconfigurable architecture on the base of FPGA]. *Sbornik nauchnykh trudov: ITMiVT* [Collection of proceedings of Lebedev Institute of Precise Mechanics and Computer Engineering]. Moscow, 2008, iss. 1, pp. 44–45. (In Russian).

4. Neuman J. von. *Theory of self-reproducing automata*. Urbana, London, University of Illinois Press, 1966. 403 p.

5. Ulam S.M. Random processes and transformations. *Proceedings of International Congress of Mathematicians*, Cambridge, 1950. Providence, 1952, vol. 2, pp. 264–275.

6. Barzin' Yu.M. Problema universal'nosti rastushchikh avtomatov [The problem of universality of growing automata]. *Doklady Akademii nauk SSSR – Proceedings of the Russian Academy of Sciences*, 1964, vol. 57, no. 3, pp. 542–545.

7. Torgashev V.A. Avtomatnye seti i komp'yutery: istoriya razvitiya i sovremennoe sostoyanie [Automata networks: history of development and contemporary state]. *Istoriya informatiki i kibernetiki v Sankt-Peterburge (Leningrade)* [History of informatics and cybernetics in Saint-Petersburg (Leningrad)]. St. Petersburg, Nauka Publ., 2012, vol.3, pp. 46–66.

8. Torgashev V.A. Dinamicheskie avtomatnye seti [Dynamic automata networks]. *Trudy SPIIRAN – SPIIRAS Proceedings*, 2013, vol. 4 (27), pp. 23–34.

9. Torgashev V.A., Tsarev I.V. Sredstva organizatsii parallel'nykh vychislenii i pro-grammirovaniya v mul'tiprotsessorakh s dinamicheskoi arkhitekturoi [Means for organization of parallel computations and programming in multiprocessors with dynamic architecture]. *Programmirovanie – Programming and Computer Software*, 2001, no. 4, pp. 53–68.

10. Torgashev V.A., Tsarev I.V. Semeistvo superkomp'yuterov s dinamicheskoi arkhitekturoi – kontseptual'nye osnovy [Conceptual basis for a family of supercomputers with dynamic architecture]. *Iskusstvennyi intellect – Artificial Intelligence*, 2009, no. 3, pp. 251–257.

11. Torgashev V.A., Tsarev I.V. Supercomputer technologies on the base of dynamic automata networks. *Superkomp'yuternye tekhnologii: razrabotka, programmirovanie, primenenie (SKT-2010)*: materialy Mezhdunarodnoi nauchno-tekhnicheskoi konferentsii [Proceedings of International Scientific-Technical Conference “Supercomputer Technologies: Development, Programming, Implementation” (SCT-2010)]. Taganrog, 2010, vol. 1, pp. 161–165. (In Russian).

12. Torgashev V.A., Tsarev I.V. Dinamicheskie avtomatnye seti kak model' parallel'nykh vychislenii v mul'tiprotsessorakh s dinamicheskoi arkhitekturoi [Dynamic automata networks as a model of parallel computations in the multiprocessors with dynamic architecture]. *Vestnik komp'yuternykh i informatsionnykh tekhnologii – Herald of computer and information technologies*, 2009, no. 3, pp. 11–20.

13. Torgashev V.A., Tsarev I.V. Tekhnologii resheniya slozhnykh zadach na osnove dinamicheskikh av-tomatnykh setei [Technologies of solving complicated problems on the base of dynamic

automata networks]. *Informatsionno-upravlyayushchie sistemy – Information and Control Systems*, 2015, no. 6, pp. 57–65.

14. Tsarev I.V. YaRD – yazyk setevogo programmirovaniya v raspredelennykh vychislitel'nykh sistemakh s dinamicheskoi arkhitekturoi [YARD – a language for network programming in distributed computing systems with dynamic architecture]. *Iskusstvennyi intellect – Artificial Intelligence*, 2008, no. 3, pp. 761–770.

15. Torgashev V.A., Tsarev I.V. Realizatsiya superkomp'yuterov s dinamicheskoi arkhitekturoi na sovremennoi elementnoi baze [Modern circuitry implementation of dynamic architecture supercomputers]. *Informatsionno-upravlyayushchie sistemy – Information and Control Systems*, 2016, no. 6, pp. 74–84.

16. Earley J. An efficient context-free parsing algorithm. *Communications of the ACM*, 1970, vol. 13, no. 2, pp. 94–102.

Для цитирования:

Царев И.В. Система имитационного моделирования динамических автоматных сетей // Научный вестник НГТУ. – 2018. – № 3 (72). – С. 107–120. – doi: 10.17212/1814-1196-2018-3-107-120.

For citation:

Tsarev I.V. Sistema imitatsionnogo modelirovaniya dinamicheskikh avtomatnykh setei [A system for simulation modeling of dynamic automata networks]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Science bulletin of the Novosibirsk state technical university*, 2018, no. 3 (72), pp. 107–120. doi: 10.17212/1814-1196-2018-3-107-120.