

С.М. Салибекян, П.Б. Панфилов

## Моделирование распределенных вычислительных dataflow-систем в среде объектно-атрибутного программирования и моделирования.

Аннотация. Тема статьи – создание и испытание имитационной модели распределенной вычислительной системы вычислительной системы с управлением потоком данных (DATAFLOW), построенной по принципам объектно-атрибутного (ОА) подхода к организации вычислительного процесса.

В статье приводится описание ОА-архитектуры распределенной ОА-системы, обеспечивающую масштабируемость на распределенной вычислительной системе: планирование вычислений, организация обмена данными между распределенными вычислительными узлами, способ перераспределение программы по вычислительным узлам, маршрутизация сообщений, передаваемых между вычислительными узлами.

Для проверки эффективности ОА-архитектуры была разработана методика имитационного моделирования в парадигме дискретных событий (discrete events simulaton (DES)). На основе методики была разработана среда ОА-программирования и моделирования, которая была использована для симулирования решения задачи перемножения матриц.

С статье также приведены результаты моделирования, показывающие масштабируемость вычислений в распределенной ОА-системе на задаче перемножения квадратных матриц больших размеров.

*Ключевые слова и фразы:* Вычислительная система с управление потоком данных, dataflow, объектно-атрибутная архитектура, имитационное моделирование, масштабируемость вычислений, распределенная вычислительная система, параллельные вычисления, перемножение матриц.

## Введение

Современные параллельные системы довольно часто строятся на базе распределенных ВС. Это объясняется тем, что стоимость распределенной ВС, собранной из серийных вычислительных модулей, объединенных вычислительной сетью, намного ниже, нежели один мощный суперкомпьютер. Подобные ВС весьма сложны в программировании, т.к. программа состоит из множества изолированных блоков, запущенных на различных вычислительных узлах и обменивающихся между собой сообщениями. Это также вызывает трудности синхронизации вычислений и их масштабирования, связанные с тем, что в современной вычислительной технике преобладает control flow подход (управление потоком данных), слабо приспособленный для параллельных вычислений. В результате программисту приходится вручную производить распараллеливание вычислений: разбивать вычислительную задачу на несколько параллельно работающих подзадач, организовывать обмен данными между этими подзадачами. С масштабированием вычислительной задачи также возникают проблемы: при изменении размерности задачи или изменении состава вычислительного комплекса программу приходится существенно модифицировать (вплоть до полного переписывания программы).

Эти трудности преодолеваются в парадигме dataflow, где вычисления синхронизируются автоматически по данным [1]. Dataflow также решает проблему масштабирования вычислений: вершины программы жестко не привязываются к определенным вычислительным ресурсам, т.е. выделение ресурсов происходит динамически по время вычислительного процесса. Однако dataflow-парадигма вычислений не получила широкого распространения – в настоящий момент не создано ни одной коммерческой универсальной вычислительной системы, хотя весьма распространены специализированные dataflow-системы на базе FPGA и нарастает популярность языков

программирования, построенных по этому принципу (например, Caltrop, Erlang [1]), что подтверждает перспективность подобного класса вычислительных систем. Сложность универсальной dataflow-BC влечет за собой большую оборудоемкость BC, что значительно повышает ее стоимость и снижает быстродействие.

В МИЭМ НИУ ВШЭ была разработана атрибутная (ОА) архитектура, относящаяся к классу dataflow, обладающая предельной простотой: наипростейший формат токена для передачи данных, отсутствие сложного оборудования для хранения промежуточных данных поиска комплекта данных для формирования операции [2,3,4]. Со временем возникла необходимость исследования эффективности ОА-архитектуры на различных классах вычислительных задач. Для решения этой проблемы был создан способ моделирования ОА-вычислительного процесса на базе методики Discrete Event Simulation (DES) [5]. Разработанная модель используется для моделирования различных задач, в статье же внимание будет уделено моделированию перемножения матриц, так как матричное умножение является базовой операцией линейной алгебры, и используется для решения многих научных приложений: решение СЛАУ, интегрирование систем дифференциальных и обычных уравнений [6].

## **1. Принцип работы распределенной вычислительной системы ОА-архитектуры**

ОА-архитектура обладает предельной простотой и целостностью, благодаря чему обеспечиваются весьма полезные свойства. Одной из таких свойств – масштабируемость, которая достигается благодаря тому, что BC состоит из трех частей – виртуальной части (ОА-образ), системной (алгоритмы работы функциональных устройств (ФУ) – ОА-платформа) и аппаратной (назовем его движком). Движок представляет собой распределенную BC, состоящую из вычислительных узлов (ВУ) (многопроцессорных BC с общей памятью), объединенных между собой линиями связи. Узлы и линии связи могут образовывать граф любой топологии; единственное требование, чтобы все узлы могли осуществлять коммуникацию между

собой (либо напрямую, либо по определенному маршруту через другие ФУ или устройства-маршрутизаторы), т.е. граф должен быть связным. Движок может быть реализован на базе любых вычислительных компонентов, в том числе и на стандартных (процессоры, память, коммуникационное оборудование), что значительно удешевит ВС. ОА-платформа – это совокупность алгоритмов, работы всех типов ФУ. Алгоритм анализирует пришедшие к нему токены (в ОА-архитектуре токен называется информационной парой (ИП)) с данными, по информационному полю токена (атрибут) ФУ опознает прикрепленные к нему данные, и принимает решение о том, как их обрабатывать (алгоритм обработки данных зависит от атрибута ИП и от состояния ФУ). ОА-платформа может быть реализована как аппаратно, когда алгоритм работы ФУ выполняется специализированным оборудованием, так и программно, когда алгоритм реализуется в виде подпрограммы, запускаемой на процессорном ядре. ОА-образ представляет собой совокупность контекстов (значений внутренних регистров всех ФУ, участвующих в вычислительном процессе, и набора данных для обработки: перед запуском вычислительного процесса происходит создание и инициализация контекста.

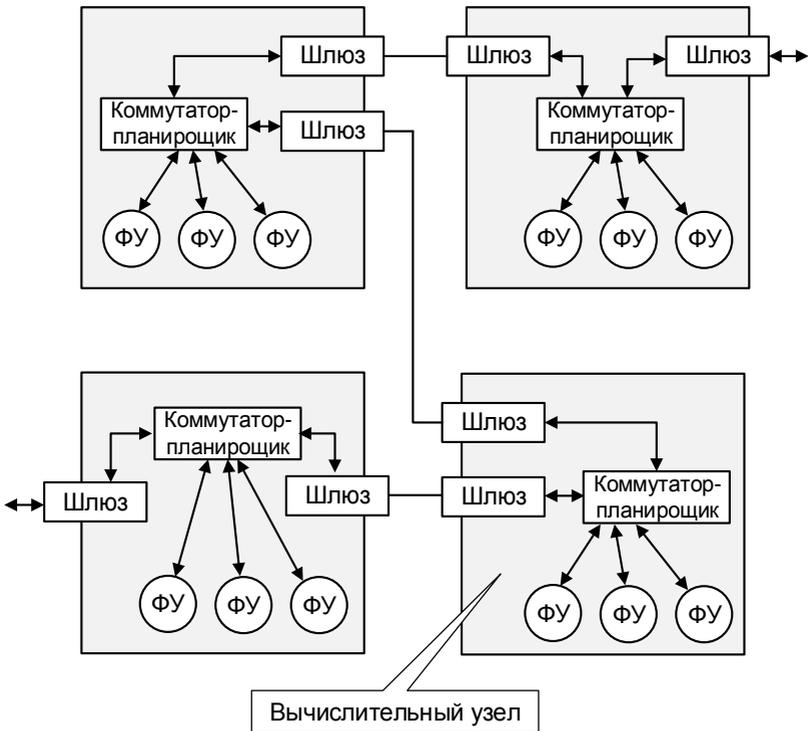


Рис. 1. Структура распределенной ОА-вычислительной системы

Такое деление на три части и позволяет обеспечить свойство масштабируемости ОА-ВС, так как ОА-образ, описывающий вычислительный алгоритм, можно наложить на распределенную ВС любой топологии. Дело в том, что ФУ образуют на ВС единое адресное пространство, независящее от топологии аппаратной части ВС. Это обеспечивается следующим образом (рис. 1)

ФУ, распределенные по вычислительным узлам, обмениваются между собой информацией, оформленной в виде ИП. По служебному полю ИП (атрибуту) можно однозначно определить, какому ФУ или группе ФУ адресуется данная ИП. Поэтому в состав ВУ входит один

или несколько планировщик-маршрутизаторов, обеспечивающих доставку ИП адресату (адресатам). Планировщик-коммутатор также занимается выделением вычислительных ресурсов для ФУ (если ИП приходит к конкретному ФУ, то для него выделяются вычислительные ресурсы и ему передается ИП для обработки). Если адресат находится на том же ВУ, то ИП сразу передается ему, если нет, то планировщик-маршрутизатор направляет ИП в канал передачи информации (шлюз) на другой ВУ.

Для запуска ОА-образа на выполнение необходимо распределить ФУ по ВУ и настроить соответствующим образом маршрутизаторы, чтобы они могли определить маршрут доставки ИП адресату (адресатам) и произвести настройку ФУ. Если конфигурация «движка» изменяется, то необходимо по-другому распределить ФУ по ВУ и произвести перенастройку планировщиков-маршрутизаторов, изменение ОА-образа не требуется. Таким образом обеспечивается полная масштабируемость ОА-ВС.

ФУ «Планировщик-коммутатор», осуществляет стыковку аппаратной части ОА-ВС с его ОА-образом (рис. 2). Одна из функций этого устройства – выделение аппаратных ресурсов. Необходимость их выделения возникает, когда к ФУ приходит ИП и требуется ее обработать. Если вычислительных все исполнительные устройства (процессорные ядра) заняты, то запрос на выделение вычислительных ресурсов помещается в очередь ожидания ресурсов, а ИП помещается в очередь на выполнение ФУ. Очередь попадают на ожидание ФУ ИП попадает и в том случае, когда ФУ занято обработкой предыдущей ИП. Предоставление и выделение вычислительных ресурсов для ФУ производится планировщиком-коммутатором согласно определенному алгоритму (рис. 2).

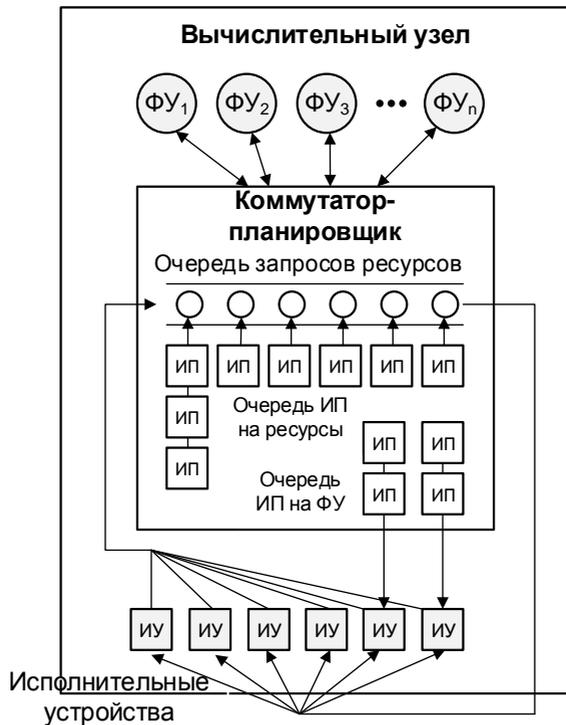


Рис. 2 Функциональная схема вычислительного узла  
ОА-ВС

## 2. Методика моделирования распределенной ОА-вычислительной системы

Для исследования ОА-ВС была разработана среда ОА-программирования и моделирования, которая позволяет производить моделирование вычислительного процесса в распределенной ОА-ВС. Среда представляет совокупность подпрограмм реализации логики работы ФУ, составляющих ОА-платформу; компилятор специализированного ОА-языка, служащего для программирования ОА-ВС (с помощью ОА-языка можно описать не только алгоритм работы ОА-ВС, но и ее архитектуру, а также задать параметры имитационного моделирования). Для проведения имитационного моделирования в

состав ОА-платформы входят ФУ Eventser, обеспечивающее контроль событий, происходящих во время моделирования; а также ФУ Scheduler, эмулирующее ФУ «Планировщик-коммутатор». В имитационной модели могут присутствовать сразу несколько ФУ Scheduler (Scheduler эмулирует работы отдельного ВУ) и только один Eventser, т.к. он контролирует последовательность событий во всей ВС.

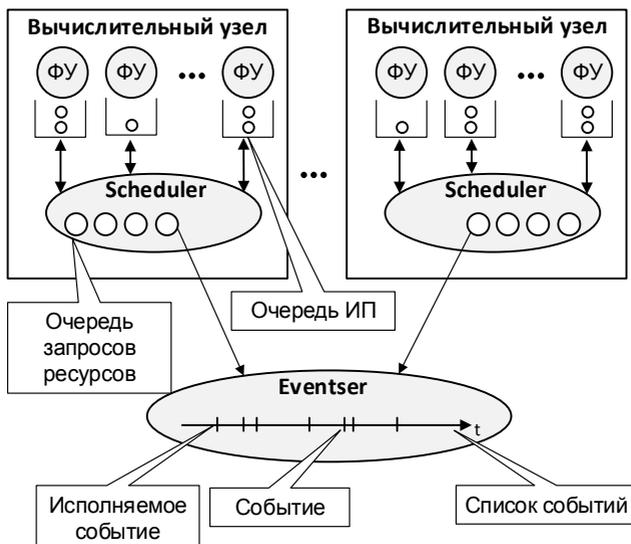


Рис. 3. Методика моделировать ОА-вычислительного процесса

В ОА-системе модель строится по принципу дискретных событий (Discrete Events Simulation (DES) [7]), где событием является передача ИП от одного ФУ к другому. Процесс моделирования строится следующим образом (рис...). ФУ после поступления на него ИП, ставит его в очередь ожидания и посылает сообщение на Scheduler об длительности производимой операции (ссылка на Scheduler входит в контекст исполнительного ФУ) и переходит в режим ожидания. Scheduler, выделяя для ФУ вычислительный ресурс, передает на Eventser информацию о предстоящей операции, производимой

ФУ, и время ее выполнения. Eventser делает отметку на временной шкале, когда необходимо дать разрешение на выполнение операции для ФУ:  $t=t_0+\tau$ , где  $t_0$  – текущее модельное время,  $\tau$  - предполагаемая длительность обработки ИП. На временной шкале Eventser-а выстраиваются сразу несколько событий (список событий), и Eventser активирует то ФУ, чья метка активации оказывается самой ранней, т.е. стоит в конце списка. Процесс моделирования завершается, когда в очереди событий не остается ни одной метки (рис. 3) или при необходимости может быть добавлено иное условие завершения процесса моделирования.

### 3. Имитационное моделирование ОА-ВС для перемножения матриц

Проверить масштабируемость распределенной ОА-ВС и эффективность ее работы было решено на примере алгоритма перемножения матриц. Для ее решения была разработана следующая архитектура ВС. Все вычисления производятся на исполнительных ФУ, которые аккумулируют все необходимые данные для получения элемента  $n$  результирующего массива, который вычисляется по формуле:

$$c_{ij} = \sum_{r=1}^n a_{ir}b_{rj} \quad (i = 1, 2, \dots, m; j = 1, 2, \dots, q).$$

, где  $a_{ij}$ ,  $b_{ij}$  – элементы исходных матриц. Для ускорения процесса умножения матриц разработаны параллельные алгоритмы [8]. Мы же будем использовать следующую методику перемножения матриц.

Элементы исходных матриц А и В помещаются в информационные пары (ИП) и передаются в исполнительные ФУ, которые производят вычисление одного элемента результирующего массива (таким образом, число исполнительных ФУ будет равно числу элементов результирующего массива). В атрибуте ИП, которую получает исполнительное устройство, зашифровываются индексы элементов исходного массива, а также указывает к какому из массивов (А или В), он относится. Выдача ИП с исходными данными возложена на ФУ «Менеджер». Результат, полученный исполнительным устрой-

ством оформляется в виде ИП и передается ФУ-коллектору, который формирует результирующую матрицу. Нами исследовались два вида топологий ФУ: шина (рис. 4) и 2-мерная сетка (матрица) (рис. 5). Совокупность исполнительных устройств в матричной топологии будем называть сеткой ФУ.

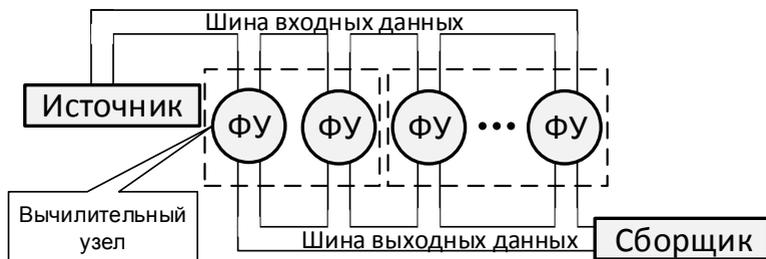


Рис. 4 Топология шина

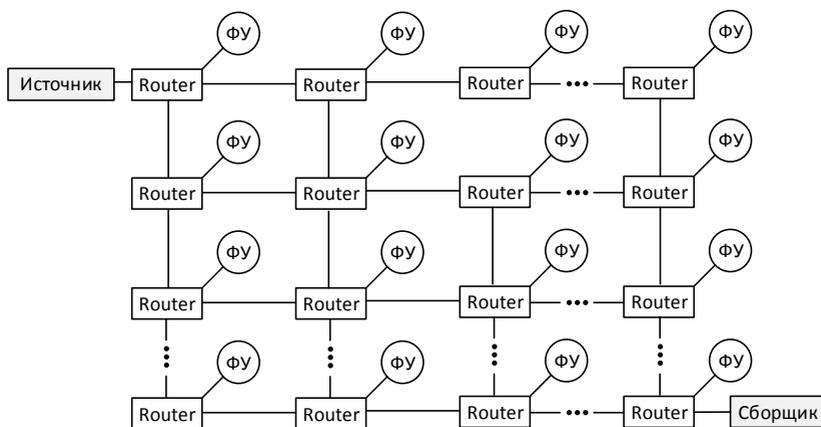


Рис. 5 Топология матрица

При реализации шинной топологии в модель вводится ФУ, эмулирующее общую шину. Выданные на шину ИП поступает на входы всех ФУ и ФУ, исходя из атрибута, принимает решение считывать

данные или нет. Результат вычислений ФУ также выдают на шину, с которой его считывает коллектор. Для эмуляции SMP-систем исполнительные ФУ могут распределяться между несколькими ВУ, т.е могут быть связаны с разными ФУ Scheduler.

Для топологии «матрица» с каждым исполнительным ФУ ассоциируется ФУ-маршрутизатор, ответственный за пересылку данных (как доставку для ФУ исходных данных, так и передачу вычисленных исполнительными устройствами элементов выходного массива коллектору).

Для матричной топологии производилось моделирование распределенной ВС. Для этого в состав модели было введено ФУ шлюз – устройство, эмулирующее передачу данных по линии связи. Каждый вычислительный узел (ВУ), включающий в себя сегмент расчетной сетки, снабжался собственным Scheduler-ом, который эмулировал распределение вычислительных ресурсов в одном ВУ. Схема распределенной ВС представлена на рис. 6.

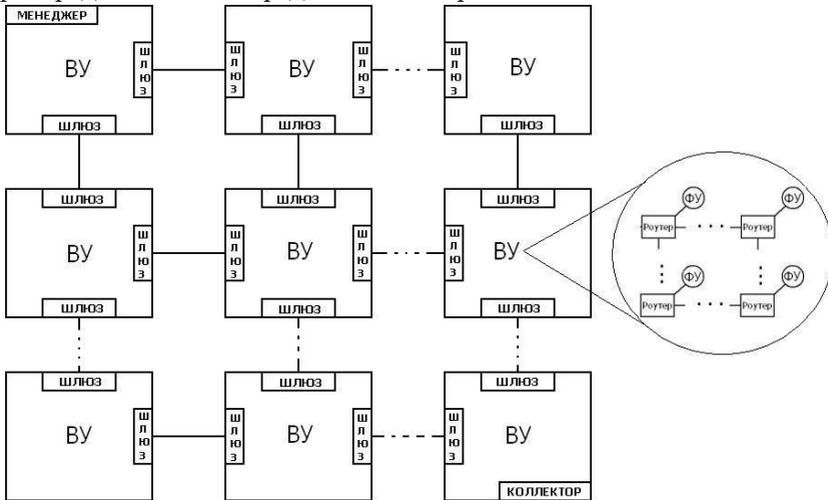


Рис. 6 Схема распределенной вычислительной системы для перемножения матриц

Разработанные подпрограммы реализации логики работы ФУ были встроены в среду ОА-программирования и моделирования, и дальнейшее исследование модели производилось в ней. Среда имеет встроенный язык программирования, с помощью которого имеется возможность создавать виртуальные ФУ, производить их инициализацию, вводить исходные данные и получать результаты вычислений, а также контролировать параметры моделируемого вычислительного процесса.

Разработанная модель контролирует следующие параметры:

Входные параметры модели:

- Размерность исходных матриц
  - Время записи данных в исполнительное ФУ (LoadTime)
  - Время передачи данных между ФУ (BusTime)
  - Время работы исполнительного ФУ (время умножения) (MulTime)
  - Время записи результата на коллектор (SaveTime)
  - Время передачи данных по каналу связи (задержка шлюза) (SluiceTime)
  - Топология связей между исполнительными ФУ: «шина», «матрица».
  - Количество ядер на ВУ.
  - Размер сегмента в вычислительной сетке.
  - Количество последовательно вычисляемых матриц (ВС произвела умножение нескольких матриц подряд.
- И т.д.

Выходные (анализируемые) параметры модели:

- время решения вычислительной задачи;
- количество задействованных вычислительных ресурсов (исполнительных устройств);

- количество ИП, находящихся в очереди ожидания ресурсов (данный параметр влияет на необходимый объем оперативной памяти на ВУ);
- длина очереди ИП на ожидание передачи данных по каналу связи;
- количество задействованных в вычислениях исполнительных устройств.

#### 4. Результаты моделирования

Разработанная модель была использована для проверки масштабируемости вычислительной системы. Во время исследований проверялась зависимость времени вычислений, объема буферов, необходимых для хранения промежуточных данных на вычислительных узлах и шлюзах от размерности вычислительной сетки. Объем буферов вычислялся следующим образом: во время прогона модели определялась максимальная длина очереди ИП на Scheduler-ах и на передачу по шлюзам, образующиеся на каждом вычислительном узле, затем все полученные величины суммировались (также модель позволяет вывести максимальных длины очередей ИП для каждого сегмента вычислительной сетки). Данный параметр важен ввиду того, что под буфера требуется оперативная память, объем которой влияет на стоимость вычислительной системы.

В результате исследований было выяснено, что в случае довольно малого времени, затрачиваемого на выполнение операции умножения по сравнению с операциями пересылки данных, производительность вычислительной системы растет пропорционально квадрату от размерности поля вычислительных устройств (или линейно относительно числа вычислительных узлов в составе системы). Т.е. если размерность вычислительной сетки  $n \times n$ , то производительность пропорциональна  $n^2$ . Объем необходимых буферов для хранения промежуточных данных растет пропорционально  $n^3$ , а объем буферов для передачи данных растет пропорционально  $n^2$ . Такой результат получается вне зависимости от размерности вычислительно сегмента (проверялось размерности  $3 \times 3$ ,  $4 \times 4$ ,  $5 \times 5$ ,  $10 \times 10$ ). На рисунке 7 приведены графики зависимости времени вычислений,

и объема необходимых буферов для планировщика и шлюзов в зависимости от размерности вычислительной сетки для системы с размерностью вычислительного сегмента  $2 \times 2$  (по оси абсцисс отложена размерность вычислительной сетки). Также приведены графики корня от времени вычислений и кубического корня от объема буферов на планировщиках и квадрата объема буферов на шлюзах. Эти графики представляют собой прямые линии, что подтверждает степенную зависимость параметров.

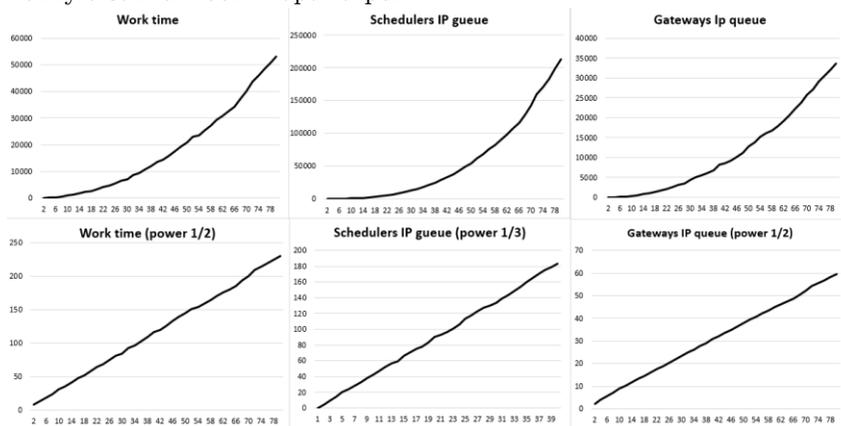


Рис. 7 Зависимость параметров системы от размерности вычислительной сетки

В том случае, когда длительность операции умножения больше времени пересылки данных по вычислительной сетке ФУ, то время вычислений растет линейно, объем буферов на планировщиках – квадратично, а размер буферов для шлюзов – линейно. Этот факт можно объяснить с помощью графика, иллюстрирующего вычислительный процесс (рис 8). Зеленый график обозначает количество ФУ, выполняющие операцию умножения с накоплением, красным – общее число работающих ФУ (исполнительные устройства и маршрутизаторы). На первом графике представлен вычислительный процесс, в котором преобладает время выполнения операции умножения

(а). В этом случае наступает момент, когда на все исполнительные ФУ приходят исходные данные и на графике возникает «полка». На графике (b) вычисления выполняются над матрицами размером  $60 \times 60$ , поэтому преобладает время пересылки исходных данных по вычислительной сетке и результаты вычислений начинают выдаваться исполнительными устройствами еще до того, как все исходные данные будут доставлены своим потребителям. В результате динамика вычислительного процесса для случая (b) существенно меняется.

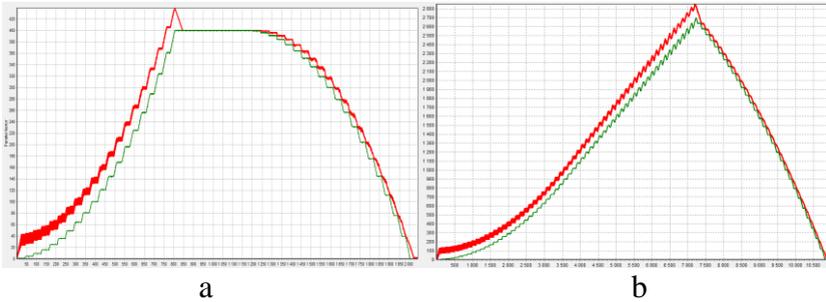


Рис. 8. Зависимость параллелизма вычислений во время перемножения матриц размерностью  $15 \times 15$ (a) и  $60 \times 60$ (b)

Модель позволяет наблюдать распределение необходимых буферов ИП по вычислительным узлам. На рис. 9 приведены результаты трех прогонов модели (слева – график необходимых буферов на планировщиках, справа – график необходимых буферов для шлюзов) Проводилось моделирование умножение матриц размерностью  $60 \times 60$ , с сегментами размерами  $5 \times 5$  при различных параметрах модели.

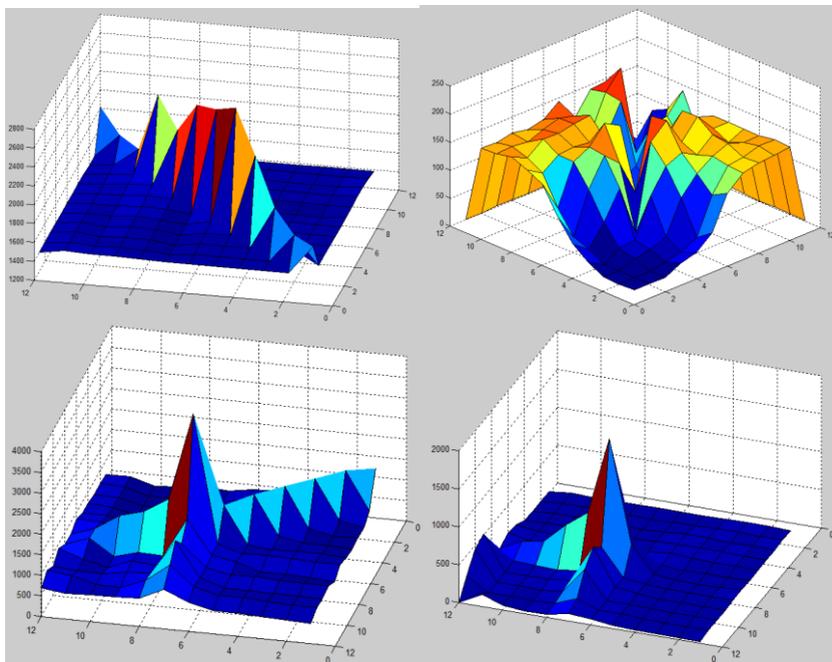


Рис. 9 Распределение максимальных длин очередей ИП на планировщиках (слева) и на шлюзах (справа) на вычислительном поле размерностью 60x60 с сегментами 5x5

## Заключение

Разработанная имитационная модель имеет возможность моделирования вычислительной задачи перемножения матриц на ОА-ВС и может быть полезной при проектировании ВС, с аппаратной частью построенной, как по фон-неймановской архитектуре, так и по ОА-архитектуре. Модель учитывает основные параметры ВС и позволяет производить ее анализ с целью нахождения оптимальных архитектуры, режимов работы и параметров ВС.

Проведенное нами моделирование позволило доказать реализуемость разработанной методики перемножения матриц как на SMP,

так и на МРР системах, а также масштабируемость ОА-ВС: ВС легко адаптируется под любые размеры матрицы, любую сеть вычислительных узлов, под любое количество исполнительных устройств (процессорных ядер) на вычислительном узле. Скорость работы ОА-ВС при небольших размерах перемножаемых матриц сравнима со скоростью блочного алгоритма перемножения матриц, которая пропорционально  $N^3/P$ , где  $N$  – размерность квадратной матрицы,  $P$  – число исполнительных устройств (процессорных элементов) [9]. При больших размерах перемножаемых матриц время вычислений определяется не временем операции умножения, временем пересылки данных между ВУ и пропорциональна числу ВУ.

Проведенное исследование использовалось нами не только для оценки конкретной вычислительной задачи, но и для отработки предложенного нами метода моделирования вычисления в ОА-системе. В исследовании был значительно расширен функционал среды ОА-программирования и моделирования: добавлена возможность расчёта новых параметров модели, расширены возможности вывода результатов моделирования.

Недостатком предложенного метода моделирования ОА-ВС является невозможность проведения распределенного моделирования, т.к. в настоящее время используется модель моделирования DES (модель с единственным таймером). В качестве таймера выступает ФУ Eventser, который обрабатывает все события, возникающие в модели. Поэтому в наших дальнейших планах расширение системы моделирования для реализации модели параллельного/распределенного DES (PDES) моделирования [10], в которой присутствуют несколько таймеров (Eventser-ов). В таком случае можно будет создать системы моделирования, в которую будут входить несколько вычислительных узлов, объединенных в сеть. На каждой такой машине будет инициализироваться свой Eventser, предназначенный для контроля событий в той части модели, которая запущена на данном вычислительном узле. Сведения о глобальных событиях будут передаваться на другие вычислительные узлы, занятые в процессе моделирования. Схема параллельной/распределенной ОА-модели представлена на рис 10.

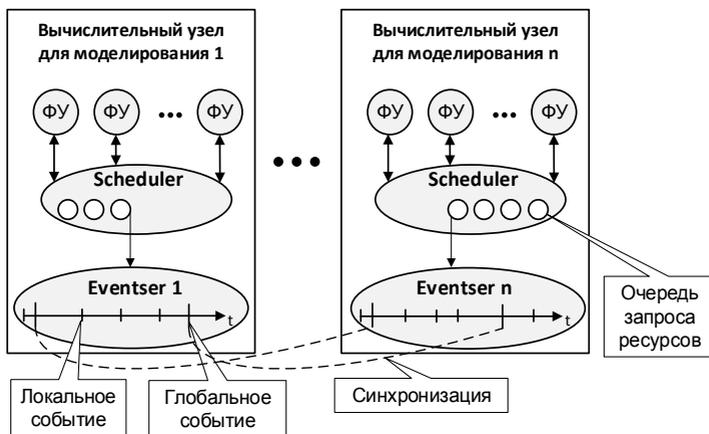


Рис. 10. Распределенное моделирование ОА-системы

Следует отметить, что разработанная нами среда ОА-программирования и моделирования применялась для имитационного моделирования в других областях: языковое моделирование, алгоритмы с альтернативными ветвями вычислений, системы автоматизации.

### Список литературы

- [1] Joe Armstrong, Robert Virding, Claes Wikström, Mike Williams. Ericsson Concurrent Programming in ERLANG. Second Edition. Telecommunications Systems Laboratories, Box 1505, S - 125 25 Älvsjö, Sweden (<http://www.erlang.se/publications/erlang-book-part1.pdf>)
- [2] Салибекян С.М., Панфилов П.Б. Объектно-атрибутная архитектура – новый подход к созданию объектных систем // Информационные технологии. 2012, №2 стр. 8-14
- [3] С.М. Салибекян, П.Б. Панфилов Моделирование суперкомпьютерной вычислительной системы объектно-атрибутной архитектуры с управлением потоком данных // Информационные технологии и вычислительные системы. №1, 2013 – стр. 3-10.

- [4] Салибекян С.М., Панфилов П.Б. Формализация dataflow-модели вычислительного процесса. // Объектные системы - 2013: материал V Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2013 г.) / Под общ. ред. П.П. Олейника. - Ростов-на-Дону: ШИ ЮРГТУ (НПИ), 2013. - С. 87-93 URL: [http://objectsystems.ru/files/2012/Object\\_Systems\\_2013\\_Proceedings.pdf](http://objectsystems.ru/files/2012/Object_Systems_2013_Proceedings.pdf)
- [5] Jerry Banks, John S. Carson II, Barry L Nelson, David M. Nicol. Discrete-Event System Simulation. Prentice Hall, fourth edition, 2005
- [6] Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. Пер. с англ. — М.: Мир, 2001. — 430 с.
- [7] Jerry Banks, John Carson, Barry Nelson and David Nicol. Discrete-Event System Simulation, Fourth edition, Pearson 2005
- [8] L. D. Jelfimova «A fast cellular method of matrix multiplication» Cybernetics and Systems Analysis May 2008, Volume 44, Issue 3, pp 357-361
- [9] G. Nimako, E.J. Otoo, D. Ohene-Kwofie Fast Parallel Algorithm for Blocked Dense Matrix Multiplication on Shared Memory Architectures. ICA3PP (1) 2012: 443-457.
- [10] Fujimoto, R. M. "Parallel and Distributed Simulation" in Handbook on Simulation, ed. J. Banks, Wiley, 1998.

*Об авторах:*



**Салибекян Сергей Михайлович**

Национальный исследовательский университет  
«Высшая школа экономики», к.т.н., доцент.

*e-mail: salibek@yandex.ru*

**Панфилов Пётр Борисович**

Минэкономразвития РФ, к.т.н., доцент

*e-mail: panfilov@miam.edu.ru*

S.M. Salibekyan, P.B. Panfilov

Implementation of Object-attribute Environment of Programming and Simulation for Simulating of Distributed Computing Dataflow System

ABSTRACT. PAPER TOPIC IS A CREATION AND TEST OF SIMULATING MODEL OF THE DISTRIBUTED COMPUTING DATAFLOW SYSTEM BASED ON OBJECT-ATTRIBUTE (OA) APPROACH OF COMPUTATION PROCESS ORGANIZATION.

IN ARTICLE IT IS THE DESCRIPTION OF THE DISTRIBUTED OA-ARCHITECTURE PROVIDED SCALABILITY ON THE DISTRIBUTED COMPUTING SYSTEM: PLANNING OF COMPUTING, THE ORGANIZATION OF DATA EXCHANGE AMONG THE DISTRIBUTED COMPUTING NODES, A WAY OF THE PROGRAMS REDISTRIBUTION FOR COMPUTING NODES, ROUTING OF THE MESSAGES TRANSFERRED BETWEEN COMPUTING NODES.

THE TECHNIQUE OF SIMULATION IN A DISCRETE EVENTS PARADIGM (DES) WAS DEVELOPED FOR OA-ARCHITECTURE EFFICIENCY VALUATION. THE OA-PROGRAMMING AND MODELING ENVIRONMENT WAS DEVELOPED ON THE BASIS OF THE TECHNIQUE. THE ENVIRONMENT WAS USED FOR SIMULATING OA-SYSTEM WITH THE MATRIXES MULTIPLICATION TASK. THE MODELING RESULTS SHOW COMPUTATION SCALABILITY IN THE DISTRIBUTED OA-SYSTEM ON A TASK OF MULTIPLICATION OF SQUARE MATRIXES OF THE BIG SIZE.

*Keywords and phrases:* dataflow, object-attribute architecture, simulating modeling, computation process scalability, the distributed computing system, parallel computing, multiplication of matrixes.