# Моделирование суперкомпьютерной вычислительной системы объектно-атрибутной архитектуры с управлением потоком данных

Салибекян Сергей Михайлович, Московский институт электроники и математики НИУ ВШЭ, к.т.н.
Панфилов Пётр Борисович, Минэкономразвития РФ, к.т.н., доцент

**Аннотация**

В статье описаны особенности математического и имитационного моделирования разработанных в МИЭМ НИУ ВШЭ атрибутной (А) и объектно-атрибутной (ОА) архитектур вычислительных систем (ВС), реализующих принцип управления вычислениями с помощью потока данных (dataflow-парадигма), и результаты имитационного моделирования dataflow-ВС, полученные в рамках выполнения НИР по разработке и моделированию архитектуры суперкомпьютерной dataflow-ВС по ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2007-2013 годы» при финансовой поддержке Минобрнауки России. Для математического моделирования dataflow-ВС был значительно переработан аппарат сетей Петри с тем, чтобы приспособить его к моделированию вычислительных систем с управлением потоком данных. Был предложен новый формализм А-сетей (атрибутных сетей). Для имитационного моделирования dataflow-ВС была разработана среда программирования и имитационного моделирования, основанная на формальном аппарате А-сетей.

.

**Ключевые слова**: вычислительная система с управлением потоком данных, dataflow-парадигма, моделирование, математическая модель, язык программирования, среда программирования, сетевая модель, А-сети.

## Введение

Уже более 10 лет в Московском институте электроники и математики Научно-исследовательского университета «Высшая школа экономики» (МИЭМ НИУ ВШЭ) ведется работа над милликомандной или атрибутной (А) и объектно-атрибутной (ОА) архитектурами вычислительных систем (ВС) [1,2]. Данные архитектуры работают по принципу управления вычислительным процессом с помощью потока данных (dataflow-парадигма организации вычислений). До настоящего времени универсальные dataflow-ВС коммерческого применения не нашли, несмотря на то, что они более приспособлены к реализации параллельных вычислений, чем классические системы, работающие по принципу управления вычислениями с помощью потока команд (controlflow-парадигма). Однако разработанные коллективом исследователей МИЭМ НИУ ВШЭ архитектуры обладают рядом преимуществ, которые, предположительно, позволят им составить конкуренцию классической (фон-неймановской) архитектуре ВС.

А- и ОА-ВС представляют собой набор функциональных устройств (ФУ), выполняющих вычислительную работу и обменивающихся между собой данными, оформленными в виде информационной пары (ИП) – простейшего токена, состоящего из двух полей: атрибута (идентификатора, представляющего собой целое число) и данных или указателя на ячейку оперативной памяти (ОП), где хранятся данные. В каждый момент времени ФУ может принимать только одну

информационную пару. Вычислительный процесс задается с помощью описания обмена данными между ФУ (описание обмена данными между ФУ будем называть А- или ОА-образом). Основным отличием разработанных архитектур от существующих универсальных dataflow-систем является то, что исполняемый пакет (полный комплект данных для операции), собирается не в отдельном блоке поиска совпадений (англ. matching), а непосредственно во внутренних регистрах самих функциональных устройств (ФУ считывает данные с линии передачи информации и формирует исполняемый пакет исходя из атрибутов ИП). Данное техническое решение позволяет:

- сделать ВС распределенной (т.е. нет централизованного блока, который контролирует весь вычислительной процесс);
- реализовывать многооперандные команды;
- существенно упростить ВС (нет необходимости в применении коммутационных сред, множестве пересылок данных между блоками, входящими в состав ВС; в частности, можно обойтись без применения громоздкой ассоциативной памяти и т.д.) и тем самым снизить объем оборудования, входящего в состав ВС;
- обеспечить масштабируемость ВС;
- обеспечить одинаковую эффективность работы как режиме крупнозернистого, так и мелкозернистого параллелизма;
- создавать реконфигурируемые ВС (причем, реконфигурацию можно осуществлять непосредственно во время вычислительного процесса).

В то время как разработанная авторами А-архитектура более подходит к аппаратной реализации dataflow-ВС, ОА-архитектура ВС оказалась применимой как к аппаратной, так и к программной частям ВС. Для описания алгоритма работы ОА-ВС был разработан ОА-язык параллельного программирования [5], на основе которого затем была создана среда программирования и компилятор языка. Благодаря тому, что язык программирования и «железо» функционируют по одним и тем же принципам, в ОА-системе практически отсутствует проблема семантического разрыва между языком программирования высокого уровня и аппаратной частью ВС.

В процессе выполнения НИР по разработке и моделированию суперкомпьютерной dataflow-ВС ОА-архитектуры в условиях отсутствия макетного образца ОА-системы для получения и оценки вычислительных характеристик и параметров ОА-ВС по производительности, масштабируемости и др. активно применялось имитационное моделирование ВС. В связи с этим встала задача разработки подходов к математическому и имитационному моделированию А- и ОА-архитектур ВС и собственно dataflow-ВС.

## 1. Принципы математического моделирования dataflow-ВС

В настоящее время для моделирования параллельных ВС широко используются сетевые модели и, в частности, сети Петри. Однако сети Петри, в чистом виде, оказались не очень удобным формализмом для моделирования параллельных ВС и, в частности, dataflow-систем. Так профессор Массачусетского технологического института, США, Карл Хьюит (англ. Carl Hewitt), отмечает следующие недостатки сетей Петри:

- Сети Петри имеют следующее ограничение: они моделируют управление потоком, но не сам поток данных.
- Сложность описания одновременных действий, происходящих во время вычислительного процесса.

- Физическая интерпретация перехода в сетях Петри весьма сомнительна.

Поэтому потребовалось разработать новый формализованный аппарат описания dataflow-ВС, который был бы свободен от вышеописанных недостатков. В результате, был предложен формализм модели атрибутной сети или А-сети (Рис. 1 и 2), который представляет собой восьмерку:

$$A=\{I,C,O,EC,CO,OE,EM,OM\},$$

где I – множество входных узлов;
C – множество вычислительных узлов;
O – множество выходных узлов;
IC: $I \rightarrow C$ – множество дуг из входных вершин в вычислительные узлы;
CO: $C \rightarrow O$ – множество дуг из вычислительных узлов в выходные узлы;
OI: $O \rightarrow I$ – множество дуг, соединяющих выходные вершины с входными;
IM – вектор маркировок входных узлов;
OM – вектор маркировок выходных узлов.

В теоретико-графовой интерпретации А-сеть – это трёхдольный граф, где одна доля (множество С) – это вычислительные вершины (на них производится исполнение команды), вторая (множество I) – входные данные, а третья (множество О) – выходные данные для вычислительных вершин (рис. 1).

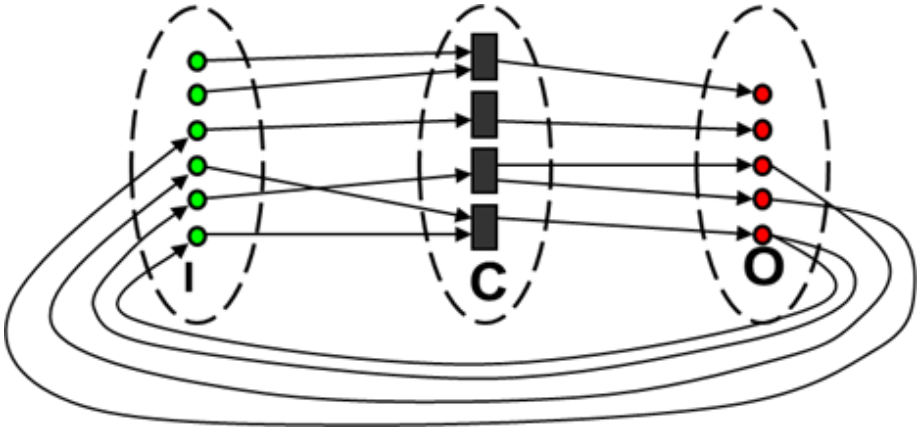


Рис. 1 – 3- дольный граф атрибутной сети

На рис. 2 атрибутная сеть представлена в другом (многоярусном) виде. Здесь вычислительная вершина обозначена прямоугольником, входные и выходные узлы обозначены кружками зеленого и красного цвета, соответственно. Для входных и выходных вершин вводится маркировка, имеющая только два значения – «пусто» и «заполнено». «Пусто» означает, что в вершину еще не поступили данные, «заполнено» - поступили. Маркировка задается с помощью вектора маркировки входных (IM) и выходных (OM) данных. Первый ярус на рис. 2 обозначает входные данные для вычислительного процесса, последний – выходные данные.

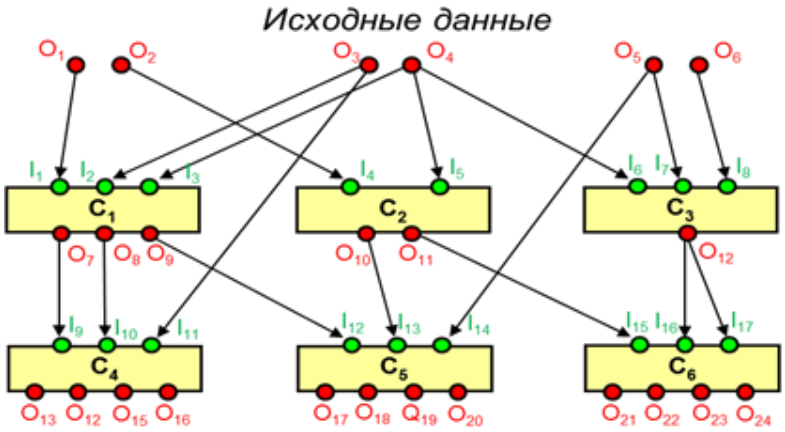


Рис. 2 – Атрибутная сеть

Выполнение А-сети представляет собой последовательность событий двух видов:

- Активация (выполнение) k-й вычислительной вершины (OM′ = OM+CO($c_k$), где $c_k \in C$, CO – функция активации вершины, OM′ – новая маркировка выходных вершин А-сети, которая получается после активации вычислительного узла). Вершина может быть активирована только в том случае, когда соответствующие ей входные вершины являются помеченными как «заполненные», т.е. для вычислительного узла поступили все необходимые для выполнения операции данные. В результате выполнения этого события изменяется массив маркировки выходных данных (OM): все смежные с вершиной $c_k$ выходные узлы помечаются как «заполненные».

- Передача данных из i-й выходной вершины во входные вершины (IM′=IM+TO($o_i$), где $o_i \in O$, TO – функция активации перехода, IM′ массив маркировок входных вершин после активации передачи данных). Это событие может быть активизировано лишь в том случае, когда вершина $o_i$ помечена как «заполненная».

Разработанная модель А-сети оказалась свободной от недостатков сетей Петри, указанных Карлом Хьюитом, а именно:

- А-сеть моделирует поток данных, т.к. маркировка отображает перемещение данных в ВС (так событие «передача данных» есть не что иное, как передача токена от одного узла ВС к другому узлу).

- Простота описания одновременных действий (высокий параллелизм ВС не приводит к сильному усложнению математической модели, как это происходило в случае сети Петри).

- Физическая интерпретация событий в А-сети весьма конкретна: активация вычислительной вершины – это окончание вычислений, производимых в вычислительном узле (в реальности вычислительный узел представляет собой исполнительное устройство) и запись получившегося результата в выходные регистры исполнительного устройства (ИУ); передача данных – это пересылка токена с данными из выходного регистра ИУ во входные узлы (в реальности входными узлами А-сети являются входные регистры ИУ).

Разработанная модель может быть применена в том числе и для временно́го моделирования вычислительного процесса, разворачивающегося в dataflow-системе. Для этих целей вводится временная ось, на которой отмечаются предстоящие события (рис. 3). Также вводятся времена пересылки данных и выполнения операции

вычислительной вершиной. При активации вычислительной вершины, которая происходит в момент модельного времени T, на временную ось наносится метка в точке $T+\tau_C$, где $\tau_C$ – время выполнения операции. Аналогично при активации пересылки данных в момент T, на временную ось наносится метка в момент $T+\tau_t$, где $\tau_t$ – время выполнения пересылки токена с данными на входные узлы (входные регистры исполнительных устройств, которые принимают данные). В текущий момент модельного времени выполняется действие, метка которого имеет наименьшее время; после выполнения действия текущая метка удаляется.

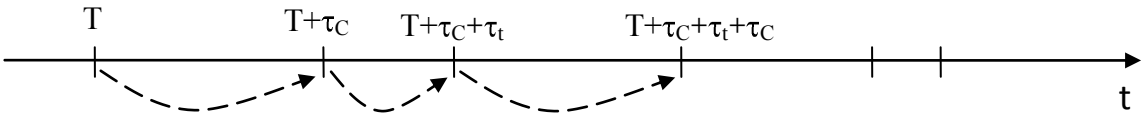

Рис. 3 – Временно́е моделирование A-сети

Например (рис. 3), после активации вычислительной вершины в момент T на временную ось в момент времени $T+\tau_C$ ставится метка пересылки данных. В момент же времени $T+\tau_C$ происходит запись полученных результатов вычислений в соответствующие выходные вершины A-сети; заполнение данными выходных вершин порождает необходимость пересылки данных во входные вершины, поэтому на временной оси через время $\tau_t$ (время пересылки данных между выходными и входными вершинами) помещается метка. Затем в момент времени $T+\tau_C+\tau_t$ происходит запись результата вычислений во входную вершину. Наличие всех необходимых выходных данных для активизации вычислительной вершины может породить активизацию новых вычислительных вершин и т.д.

Разработанный подход (на базе формализма A-сетей) к математическому моделированию dataflow-ВС лег в основу принципов имитационного моделирования dataflow-ВС A- и OA-архитектур.

## 2. Принципы имитационного моделирования ВС A- и OA-архитектур

Имитационное моделирование вычислительных систем A- и OA-арихитектур произвести намного легче, чем классических ВС ввиду нескольких причин. Во-первых, функциональные устройства (ФУ) в OA-ВС могут быть реализованы как аппаратно, так и программно, поэтому имитационное моделирование ВС сводится к созданию виртуальных копий аппаратных ФУ. Во-вторых, в OA-ВС во время имитационного моделирования не требуется применения специальных средств, производящих распараллеливание вычислений и отслеживание работы параллельно работающих устройств. Вычислительный процесс описывается как обмен данных между ФУ, то есть, одни ФУ, получив результат вычислений, пересылают его на другие ФУ, и при наличии полного комплекта данных ФУ активизируются и пересылают свои данные другим ФУ. Таким образом, вычислительный процесс несколько напоминает собой «цепную реакцию», в результате чего происходит самораспараллеливание вычислений, и, потому, не возникает потребности специально разрабатывать алгоритм распараллеливания вычислений. В-третьих, усилия, затраченные на создание имитационной модели, не пропадают зря, т.к. созданный OA-образ (описание информационного обмена между ФУ) после моделирования без изменения переносится на реальную ВС (аппаратно реализованные ФУ имеют точно такую же логику работы, как и виртуальные) [3,4]. Это существенно убыстряет и удешевляет процесс создания и отладки dataflow-ВС и их приложений.

Для осуществления имитационного моделирования была разработана среда программирования и имитационного моделирования объектно-атрибутной ВС, которая включает в себя:

- ОА-платформу (совокупность программ, реализующих логику работы виртуальных ФУ);
- компилятор ОА-языка параллельного программирования;
- инструментальные средства программирования (текстовый редактор ОА-программы, панели вывода результатов и служебных сообщений, панель инструментов, операционные подсказки и т.д.);
- средства, позволяющие осуществлять имитационное моделирование ОА-ВС.

Для осуществления имитационного моделирования параллельных dataflow-ВС потребовалось:

- Ввести в состав каждого ФУ очередь ожидания информационных пар (ИП). ИП попадает в очередь в том случае, когда ФУ, которому она адресована, уже занято вычислительной работой или ему недоступны аппаратные ресурсы (ИУ) для осуществления вычислений.
- Ввести в состав ФУ регистры, в которых хранятся длительности выполнения каждой команды, выполняемой ФУ.
- Создать ФУ, эмулирующее работу планировщика вычислений (англ. Scheduler), в обязанности которого входит распределение аппаратных ресурсов (например, процессорных ядер) между ФУ.
- Создать ФУ, эмулирующее передачу данных между вычислительными узлами (вычислительный узел может, например, представлять собой многопроцессорную/многоядерную ВС с общей памятью). Назовем такое ФУ «Шлюз».
- Создать ФУ, контролирующее время наступления событий в А- или ОА-системе (англ. Eventser). В реальной (аппаратной или программной) dataflow-ВС данный ФУ присутствовать не будет – он необходим только для проведения имитационного моделирования. ФУ Eventser создается в среде программирования и моделирования в единственном экземпляре и контролирует время наступления событий во времени во всей ОА-ВС.

На рис. 4 приведена схема принципа имитационного моделирования ОА-ВС. Данный принцип основывается на модели А-сети: виртуальные ФУ заменяют собой вычислительные вершины А-сети, контект ФУ (совокупность внутренних регистров ФУ) заменяет собой входные и выходные узлы А-сети. Контроллер же событий (Eventser) осуществляет функции контроля времени наступления событий в моделируемой ОА-системе.

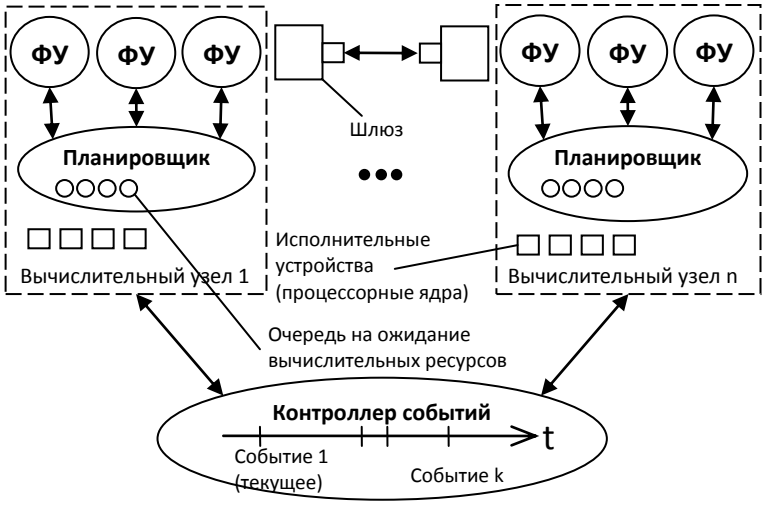


Рис. 4 – Принцип имитационного моделирования ОА-ВС

Алгоритм имитационного моделирования в этом случае выглядит сделующим образом:

1. При приходе информационной пары к ФУ, оно информирует об этом событии планировщик и переходит в режим ожидания. Если в вычислительном узле имеется хотя бы одно свободное ИУ (например, процессорное ядро), то оно выделяется для ФУ; в противном случае запрос на выполнение информационной пары ставится в очередь ожидания освобождения аппаратных ресурсов.

2. В том случае, когда для ФУ планировщиком предоставляется ИУ, на Eventser отправляется сообщение о длительности выполнения выполняемой ФУ операции – Eventser прибавляет к текущему модельному времени время задержки выполнения операции и ставит для этого события метку на временно́й оси.

3. Если соответствующая метка активизируется (это происходит, когда уже активизировались все предыдущие метки, расположенные на временно́й оси), то Eventser выдает для ФУ, которому эта метка соответствует, сигнал о выходе из режима ожидания. ФУ производит вычисления и пересылку полученных результатов вычислений на другие ФУ.

Процесс моделирования заканчивается в тот момент, когда на временной оси не осталось ни одной метки.

## 3. Результаты моделирования dataflow-ВС ОА-архитектуры

В рамках НИР по разработке и моделированию суперкомпьтерной dataflow-ВС были созданы и осуществлены прогоны в среде программирования и моделирования 7-ми тестовых пакетов, включая тесты из широкого класса физических задач (4 тестовых примера); тест Graph500 на скорость обработки теоретико-графовых задач; тест на обработку строк Grep; тест из набора SPEC (решение системы уравнений симплекс-методом). На рис. 5 приведен пример результатов моделирования dataflow-ВС ОА-архитектуры на тесте по обработке строк GREP: зависимость времени выполнения теста от числа ФУ и ИУ (число функциональных и исполнительных устройств варьируется от 1 до 50).

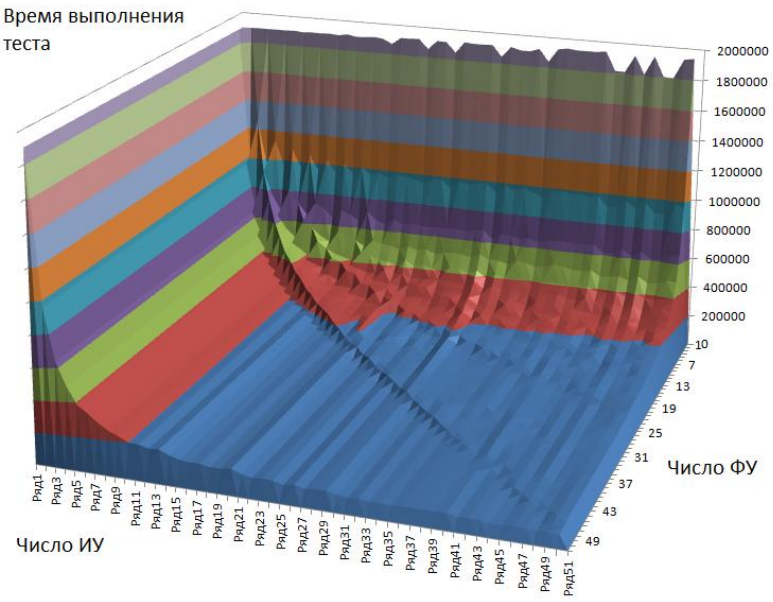


Рис. 5 – Результаты моделирования ОА-ВС на тестовом наборе GREP

По результатам проведенных прогонов тестовых примеров можно сделать следующие выводы:

1. ОА-архитектура показывает хорошие результаты при реализации на распределенных ВС: в частности, при решении физических задач широкого класса при грамотном выборе конфигурации ВС удалось добиться линейного роста производительности в зависимости от объема оборудования.

2. При моделировании расчетов широкого класса физических тестов наибольшая загрузка ОП наступала на начальном этапе вычислительного процесса (промежуточные данные, для обработки которых не хватает исполнительных устройств, накапливаются в очередях ожидания вычислительных ресурсов).

3. В ходе выполнения НИР был разработан алгоритм планировки вычислений, который позволяет при решении широкого класса физических задач несколько снизить пик потребления ОП в начале вычислительного процесса и тем самым снизить требуемый для реализации вычислений объем ОП на вычислительных узлах.

Создание среды программирования и имитационного моделирования и само моделирование ОА-ВС осуществлялось в рамках НИР по ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2007-2013 годы», финансируемой Министерством образования и науки РФ.

**Заключение**

В результате проведенной НИР по разработке и моделированию архитектуры суперкомпьтерной dataflow-ВС была найдена оптимальная архитектура ВС для решения конкретных вычислительных задач. В настоящее время ведутся подготовительные работы по инициированию и проведению ОКР по созданию макетного образца высокопроизводительной параллельной вычислительной системы ОА-архитектуры, реализующей dataflow-парадигму организации вычислений. Данная работа позволит найти удачные технические решения по созданию распределенных ОА-систем, и сделать задел для создания опытного мелкосерийного производства ВС ОА-архитектуры.

**Литература**


1. Салибекян С.М. Принципы милликомандной архитектуры как основа построения высокопроизводительных адаптивных вычислительных систем // Автоматизация и современные технологии. 2002. № 5. – Стр. 25-32.
2. Салибекян С.М., Панфилов П.Б. Перспективная суперкомпьютерная система на основе объектно-атрибутной модели вычислений с управлением потоком данных / Международная конференция «Развитие суперкомпьютерных и грид-технологий в России» в рамках «Второго Московский Суперкомпьютерного форума» Россия, Москва, ВВЦ 26–27 октября 2011 года URL: http://www.hpc-platform.ru/tiki-download_file.php?fileId=82
3. Салибекян С.М., Панфилов П.Б ОА-архитектура построения и моделирования распределенных систем автоматизации // Автоматизация в промышленности. N11, 2010. - Стр. 51-56.
4. S.M. Salibekyan, P.B. Panfilow Object-attribute architecture for design and modeling of distribute automation system. // Automation and remote control. Volume 73, Number 3, 587-595, DOI: 10.1134/S0005117912030174
5. Салибекян С.М., Панфилов П.Б. Анализ языка с помощью вычислительной системы объектно-атрибутной архитектуры. // Объектные системы - 2012: материал VI Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2012 г.) / Под общ. ред. П.П. Олейника. - Ростов-на-Дону: ШИ ЮРГТУ (НПИ), 2012. - С. 31-37 URL: http://objectsystems.ru/files/2012/Object_Systems_2012_Proceedings.pdf
6. Кельтон В., Лоу. А. Имитационное моделирование. Классика CS. 3-е изд. – СПб.: Питер; Киев: Издательская группа BHV, 2004
7. Питерсон Дж. Теория сетей Петри и моделирование систем: Пер. с англ. - М.— 254 с.
8. Харари Фрэнк. Теория графов / пер. с англ. и предисл. В.П. Козырева. Под ред. Г.П. Гаврилова. Изд. 2-е. - М.: Едиториал УРСС, 2003. - 296 с.