Распределенная математическая среда на основе IARnet*

А. С. Тарасов

Институт системного анализа Российской академии наук (ИСА РАН)

1. Введение. Что такое IARnet

Среда IARnet [1] предназначена для объединения существующих в сети ресурсов в единую распределенную сеть. К каждому ресурсу добавляется специальный агент доступа, после чего ресурс становится ИАР — информационно алгоритмическим ресурсом, доступным из среды. В среде есть возможность создавать сценарии совместной распределенной работы ряда ресурсов, необходимых для решения той или иной задачи.

Например, пусть есть задача, в которой в качестве подзадач требуется вычисление решения обыкновенного дифференциального уравнения, решение задач линейного программирования, обращение матрицы. Тогда можно написать алгоритм, который для решения этих подзадач вызывает специализированные ресурсы, находящиеся в сети Интернет и зарегистрированные в среде IARnet.

Среда IARnet находится на более высоком уровне по отношению к развивающимся GRID-системам, обеспечивающим доступ к сети компьютеров, но оставляющих программирование на совести пользователя среды GRID. В системе IARnet предполагается, что математику, реализующему распределенный алгоритм, не понадобится вникать в тон-

* Работа выполнена при поддержке ОМН РАН (Программа №4 фундаментальных исследований).

кости сетевой и распределенной работы, то есть не придется осваивать не интересные и не свойственные ему части работы. Вся эта «рутинная» часть должна быть реализована в среде IARnet и скрыта от глаз пользователя.

2. Технические достоинства и недостатки IARnet

Система IARnet предоставляет универсальную распределенную среду, позволяющую объединять компоненты, находящиеся физически на разных компьютерах в единое виртуальное сетевое приложение.

В результате достоинствами системы IARnet является:

- Возможность, писать сложные сетевые приложения с использованием различных существующих компонент.
- Возможность создавать алгоритм с большой вычислительной нагрузкой и со сложным распределенным взаимодействием, работающий не на одном вычислительном кластере, в среде со значительно большей суммарной вычислительной мощностью.
- Возможность создавать распределенный алгоритм, не выходя за рамки предметной области разработчика.
- Возможность повторного использования уже существующих ИАР-ов.
- Возможность быстрого программирования несложных задач, состоящих из интеграции уже существующих алгоритмов — «собирание алгоритма из кубиков».

Недостатки проистекают из архитектуры системы.

В системе IARnet могут эффективно решаться легко декомпозируемые задачи. В этих задачах взаимодействие между блоками относительно более слабое, чем внутреннее взаимодействие внутри блока.

Если в задаче нет естественной декомпозиции, то эффективность работы будет заметно снижена, и кроме этого возникнет задача правильного распараллеливания обработки и управления взаимодействием различных подзадач.

Если требуется создать распределенную вычислительную сеть, состоящую из однородных компонентов, в смысле типов задач или среды

на которой написаны эти задачи, то может оказаться более эффективно использовать специализированные средства.

В частности для объединения java компонент, более естественно использовать внутреннюю java технологию — RMI [3].

Пусть даже требуется выполнить задачу, которая естественно разбивается на декомпозируемые части, для которых, уже существуют ресурсы и эти ресурсы гетерогенны. Все равно может оказаться ситуация, когда сделать частное решение может быть проще, чем воспользоваться универсальной системой.

В случае, когда работа алгоритма не требует большой вычислительной нагрузки и есть возможность собрать на основе компонент алгоритм, работающий на одном компьютере, может оказаться что это будет более простой задачей, чем создание распределенного алгоритма.

Подводя итог можно сказать, что среда IARnet предназначена для достаточно сложных задач и поэтому сама по себе достаточно тяжеловесна. И для ряда относительно простых ситуаций могут существовать более простые частные решения создания распределенных алгоритмов.

Однако остается ряд задач, которые невозможно решить в этих более узких рамках, и по нашему представлению, эти задачи наиболее эффективно реализовывать именно с помощью среды IARnet. И в случае развития среды IARnet и появления широкого спектра готовых ресурсов в среде сложность создания распределенных алгоритмов будет падать.

И через какое время, сделать алгоритм в сети IARnet будет проще, чем делать специализированное решение.

3. Коммерческие достоинства и недостатки IARnet. Сравнение с чистым открытым ПО и коммерческим

Так же важным моментом, на котором хотелось бы остановить внимание, является способ предоставления программного обеспечения. Существующие способы можно разбить на две основные группы: открытое ПО и коммерческое.

3.1. Открытое ПО

Этот способ весьма естественен и, как следствие, популярен для научной среды.

В мире существуют десятки тысяч математических пакетов с открытой лицензией.

В случае если необходимый вам пакет имеет лицензию GPL или подобную ей, вы можете свободно скачивать ее, использовать в своих приложениях, модифицировать и развивать.

Однако есть ряд факторов, сдерживающих развитие этой формы программного обеспечения.

Самый главный — это низкая отдача от программного обеспечения. Абсолютное большинство свободно распространяемого ПО предоставляет собой ресурс сделанный одним человеком, который этот человек, так или иначе, сделал бы для себя и небольшого круга лиц. То есть открытое ПО развивается по остаточному принципу.

При очень большом выборе свободно распространяемого ПО очень небольшая его часть пригодна сразу для использования в вашей задаче. Чаще всего эти библиотеки приходится доделывать для своих задач самостоятельно. Даже для действительно качественных продуктов доведенных в существенной своей части до готовности, число основных разработчиков обычно невелико и представляет собой программиста одиночку или небольшой коллектив. Это в свою очередь накладывает ограничение на размер этих компонент и спектр решаемых задач. То есть эти пакеты в большинстве своем узко специализированны. Сделать качественный свободно распространяемый пакет с широким применением, по описанным выше причинам весьма затруднительно.

Еще одним свойством свободного ПО является эволюционный способ развития, когда все изменения вырастают постепенно. При ряде достоинств это имеет свои негативные стороны, а именно препятствует созданию больших пакетов с целостной идеологией. Объединение различных свободно распространяемых пакетов затрудняется разными подходами к созданию отдельных частей и отсутствию общих интерфейсов взаимодействия.

3.2. Коммерческое ПО

Коммерческое ПО лишено этих недостатков. Существует ряд крупных пакетов с широкими возможностями и цельной идеологией. Например, Mathematica [5], MATLAB [6], Maple [7].

Проблемами, препятствующими развитию подобных пакетов являются риски, которые возникают при создании коммерческого пакета. Если свободное ПО может просто постепенно эволюционировать, то разработка коммерческого программного обеспечения требует вложения денег, которые потом необходимо вернуть с помощью продаж этого ПО. При этом природа распространения ПО оказывается обычно такая, что проект либо оказывается неуспешным, либо гиперуспешным, когда полученная прибыль сильно превышает вложения. Эти риски затрудняют создание проектов.

Кроме того, создание коммерческого ПО требует большого количества деятельности, не связанной непосредственно с научным содержанием этого ПО — маркетинговый анализ, продажи и т. п.

В случае, если эти проблемы были преодолены и созданное ПО оказалось успешным, при его использовании все равно возникают некоторые трудности. Во-первых, платность продукта. Покупка программы обычно оказывается не невозможной или сильно затрудненной для российских научных организаций по причине цены этого ПО, а так же сложностей с оформлением таких закупок. Кроме того, цена программы обычно слабо кореллирует со степенью использования данного ПО, даже если в ПО предусмотрены различные лицензии.

Так же, так как фирмы, выпускающие коммерческое ПО, являются коммерческими организациями, они имеют тенденцию подминать все под себя. Это часто является препятствием для их использования для решения научных задач. Так как, например, в случае недостаточности API конкретного коммерческого пакета, его тяжело интегрировать со сторонним ПО. Исходные коды программы недоступны, имеющееся API обычно предполагает интеграцию при помощи решений поставляемых той же фирмой. Эта особенность проявляется тем сильнее, чем больше размер коммерческой фирмы, предоставляющей данное ПО. Самым ярким представителем данного подхода является фирма Microsoft.

Создание и распространение библиотек написанных на коммерческом ПО предполагает наличие соответствующего ПО, что льет воду на мельницу в первую очередь авторам данного ПО.

Коммерческое ПО плохо встраивается в академическую среду, так как научные статьи по своей сути открыты и выработаны другие механизмы обратной связи для создания научных результатов.

3.3. Сервис-ориентированная технология

Коммерческие свойства IARnet определяются тем, что она является сервис-ориентированной средой. Создание сервисов в сервис-ориентированной среде, с коммерческой точки зрения находится между открытым и закрытым ПО. Авторы некоторого сервиса могут не показывать свой открытый код, но предоставлять к нему бесплатный доступ. Конечно, никто не мешает при этом сделать и платный доступ, или наоборот открыть исходники.

В сервис-ориентированной архитектуре могут создаваться распределенные приложения, собранные из сервисов различных организаций, это похоже на естественное развитие свободного ПО. С другой стороны сервисы, которые требуют для создания и поддержки значительных усилий могут тем или иным образом оплачиваться.

Распределенность такого облака нужна не только для работы с высоконагрузочными задачами, а так же как некоторая среда, в которой могут быть одновременно большое количество независимых центров разработки. Как показывает практика, ни одна даже очень хорошая концепция не может охватить все случаи и подходить для всех задач.

Образно говоря, можно сравнить коммерческое ПО с городами и мегаполисами, открытое ПО — с небольшими деревнями, а сервис-ориентированную архитектуру — с сетью дорог, связывающую все это в единое целое.

В то же время стоит заметить, что при прочих равных условиях, алгоритм выполненный одной группой разработчиков с целостной постановкой задачи будет более удобен при использовании, чем подобный алгоритм, выполненный в среде IARnet. Однако, как оказывает жизнь, все равно существуют различные города и страны, и эти города все равно приходится связывать.

Возможности распределенной среды возрастают с ростом количества сервисов этой системы. Таким образом, возникает классический сетевой эффект, который является проблемой в начале развития среды, когда сервисов существует достаточно мало, и мощным источником развития, после набора некоторого количества сервисов.

4. Характер математических задач и возможные способы использования IARnet

Характерной особенностью задач, возникающих при решении фундаментальных проблем, является то, что они чаще всего выполняются однократно после написания. Действительно, чаще всего программа доказывает или опровергает некоторую гипотезу, что достаточно однократно или небольшое число раз.

Это говорит о том, что первоочередным фактором становится скорость написания программы, то есть простота освоения этой среды пользователем (математиком), и скорость написания и отладки кода.

Задачи можно разделить на два условных класса — требующие высокой вычислительной нагрузки и не требующей. Оба класса встречаются на практике, и они требуют различных подходов.

Если в первом случае нельзя пренебрегать объемом вычислений и требуется глубокая оптимизация кода, то во втором случае оптимизацией можно пренебречь.

Действительно, если существует распределенная сеть большой мощности, то лучше сэкономить час работы математика, потратив лишний терафлоп * час.

Существующие системы рассчитаны в основном на задачи первого класса.

Действительно, задача распределения нагрузки необходима в первую очередь для тех задач, которые невозможно выполнить на отдельном компьютере или даже кластере.

Основываясь на эмпирическом опыте автора, соотношение этих задач таково, что большая часть задач не требует большой вычислительной мощности, и при появлении соответствующей среды количество подобных задач возрастает. Для получения одного научного результата может понадобиться проверить десятки различных гипотез. Удобная среда, позволяющая относительно просто создавать подобные программы может значительно ускорить поисковый этап научной работы.

При этом для обеспечения быстрой возможности написать необходимую программу, нужно чтобы большая часть компонент была реа-

лизована и требовалась лишь собрать эти компоненты в единое целое, при необходимости дописав относительно небольшое количество компонент самостоятельно. Распределенная среда в таком случае представляет собой унифицированный доступ к как можно более широкому списку библиотек различного происхождения и местоположения, а так же средство компоновки этих модулей в необходимый в данном случае алгоритм.

Таким образом, при разработке распределенной среды для решения задач существует два значительно отличающихся друг от друга класса задач, оба из которых требуют создания распределенной среды, но совершенно по разным причинам и требующие разных подходов.

Тем не менее, не следует совсем разделять эти классы — они идут рука об руку и дополняют друг друга. Задача может разбиваться на подзадачи, сильно отличающиеся по вычислительной сложности.

5. Задачи, стоящие перед математической средой

Развивающиеся сейчас архитектуры предназначены в первую очередь для задач с большой вычислительной сложностью, второй класс задач и его задачи фактически не представлены. Если существующие библиотеки находятся в разных пакетах, то пользователю приходится самостоятельно устанавливать и интегрировать. Попробуем сформулировать задачи стоящие перед средой, направленной для задач именно второго «легкого» класса.

5.1. Простота освоения

Задачи, требующие долгой разработки, обычно выполняются людьми, имеющими соответствующую квалификацию. Потребность провести вычислительный эксперимент может возникать у любого математика, от которого мы можем требовать только владение математическим аппаратом и базовое понимание программирования.

В частности из этого следует вывод, что для совсем простых задач, требующих объединения компонент, какого-либо программирования не должно быть вообще.

Если в задаче требуется написать более сложный сценарий взаимодействия или дописать самостоятельно некоторые компоненты, это должно делаться на языке максимально приближенного к тому языку, на котором рассуждает сам математик.

В частности математические формулы должны выглядеть не в строковом представлении sqrt(1/2), а визуально $\sqrt{1/2}$. Это упрощает написание программы и улучшает ее читаемость. Кроме этого, должны поддерживаться привычные математические нотации: возможность ввода букв различного алфавита, верхних и нижних индексов, обозначений булевой алгебры, теории множеств и пр.

5.2. Минимизация времени программирования

От пользователя должны быть скрыты все задачи по объединению ресурсов воедино.

Простые задачи, не требующие фактически сложного программирования должны делаться в течение нескольких минут.

5.3. Визуализация

Часто вычислительный эксперимент требуется не только для вычисления некоторого числа или проверки некоторого утверждения, но и для создания некоторого изображения, которое пользователь может визуально просмотреть. Например, пользователь может строить график некоторой функции, чтобы на глаз определить некоторые ее свойства. Это необходимая вещь на начальном этапе исследований, когда ученому требуется почувствовать, в чем собственно состоит задача.

Должна существовать большая расширяемая библиотека различных способов визуализации. Должна существовать простая, по возможности автоматическая, система компоновки пользовательского интерфейса из различных существующих визуальных компонент. Можно сформулировать следующие стандартные компоненты визуализации:

- Визуализатор формул.
- Система, показывающая один или несколько 2D графиков.
- Системы, показывающие множества, заданные на прямой, плоскости, комплексной плоскости.

- Просмотрщик трехмерных поверхностей и тел.
- Визуализация одномерных и двумерных массивов чисел.
- Визуализация графов.
- Визуализация многомерных массивов с объектов сложной структуры при помощи различных, легко настраивающихся срезов.
- Просмотр графических, анимированных и видео объектов.
- Предложение сохранения объектов в виде файла в соответствующем формате.

5.4. Интерактивность

У пользователя должна быть возможность легко играться с системой, настраивая различные параметры, то есть выполнять алгоритм при различных начальных данных, так чтобы итерация — «запустил алгоритм, посмотрел результат, перезапустил» проходила максимально быстро и легко. Система должна быть, что называется, «на кончиках пальцев».

Можно сформулировать некоторые общие интерактивные компоненты:

- Ввод чисел, как численный, так и координатно-графический (в том числе и для комплексных чисел).
- Графический планшет для геометрических построений.
- Редактор графов.
- Все просмотрщики графиков и множеств, должны быть обеспечены удобными средствами навигации.

5.5. Широкий список компонент (сервисов)

Высокая скорость написания программ может быть возможна только в случае, если большинство компонент нужно только подключить.

По мере развития системы количество сервисов будет увеличиваться за счет пользователей. Однако на начальном этапе развития, отсутствие большого списка готовых компонент будет одним из главных факторов, препятствующих развитию среды.

Один из способов преодолеть подобный замкнутый круг заключается в том, чтобы наиболее полно сделать список сервисов для достаточно узкой тематики. Тогда сетевой эффект начнет работать достаточно быстро. После этого предполагается постепенно осваивать смежные задачи.

5.6. Преобразование данных

Система должна уметь хорошо интегрировать данные различных форматов, уметь преобразовывать один в другой, не теряя необходимых данных, проверять входящие данные на соответствие необходимому формату.

6. Концепция, реализующая необходимые условия

Идеальную систему, учитывающую все эти факторы, построить на данный момент невозможно или очень сложно. По мнению автора, оптимальная система, наиболее полно сочетающая в себе рассмотренные качества, должна быть построена на основе тонкого клиента, когда пользователь заходит на определенный сайт при помощи браузера. Сайт в свою очередь представляет фронтенд для распределенной среды, включающей в себя различные существующие сервисы. На сегодня, благодаря развитию концепции ajax [13], веб-сайты могут обеспечивать высокий уровень интерактивности и визуализации, достаточный для большинства математических задач. Кроме этого, возможны специализированные фронтенды, которые необходимо скачивать и устанавливать.

Такой подход обеспечивает минимальный уровень сложности интеграции уже существующих компонент. Практически с точки зрения пользователя системы, интеграция будет сводиться к установке необходимых плагинов в браузер, что делается стандартными средствами.

Для простых пользовательских приложений, должно существовать визуальное средство связывания различных компонент между собой. Ближайшим аналогом по тому, как это должно быть сделано с точки зрения концепции, интерфейса и уровня реализации можно назвать

сервис Yahoo.pipes [14], которые обеспечивает подобную задачу для объединения и обработки RSS-потоков [15]. Эта система не должна быть сложной и должна обеспечивать объединение сервисов в цепочку выполнения (pipe идеология), в несложный граф без циклов с несколькими источниками и одним выходом. Как максимум, могут использоваться компоненты, заключающие некоторую процедуру в цикл, или распараллеливающие задачу. Это тоже должно делаться прозрачно для пользователя.

Более сложные задачи должны реализовываться на скриптовом языке, обеспечивающим естественную для математиков нотацию. К сожалению, языков, близких к этой концепции, достаточно мало. Можно в качестве наиболее близких примеров привести внутренние языки пакетов Mathematica и Maple. К сожалению, они коммерческие и их использование затруднено по техническим и лицензионным соображениям.

В качестве альтернативы можно выбрать один из популярных программистских скриптовых языков — JavaScript [8], ECMAScript [9], python [10], ruby [11], языки использующиеся математиками Haskell [12], внутренний язык открытого пакета Maxima [4]. Возможности написания скриптов не должны ограничиваться одним языком, должна быть возможность подключать различные языки, так как разные пользователи знают разные языки и для разных задач лучше подходят разные языки. В идеале должна быть возможность создавать скрипты на всех встроенных языках мат. пакетов (MATLAB, Mathematica, Maple, Maxima), на языке Haskell, и как минимум на одном из программистских скриптовых языков.

7. Концепция математической среды на основе IARnet для комбинаторной геометрии

Как говорил Козьма Прутков «нельзя объять необъятное». Поэтому надо создавать подобную систему, начиная с отдельного узкого сегмента. В качестве подобного раздела автор берет близкий ему раздел, состоящий из комбинаторной геометрии, изучения свойств упаковок, разбиений, выпуклых многогранников.

7.1. Сервисы

Для полноценного функционирования системы должен быть создан необходимый минимум сервисов общего пользования, на основе которых пользователи будут конструировать свои задачи.

Должны быть подключены сервисы, решающие стандартные геометрические задачи:

- обращение матрицы;
- выпуклая оболочка точек и многогранников;
- пересечение выпуклых многогранников;
- сумма по Минковскому;
- нахождение Чёбышевского радиуса;
- полярное преобразование;
- преобразования объектов при помощи движений, проецирований, аффинных, проективных и т. п. преобразований;
- вычисление длин, площадей, объемов, плоских и двугранных углов, кривизн объектов заданных форматов;
- создание многогранника с реберным остовом, совпадающим с заданным графом;
- вычисление разбиений Вороного и триангуляции Делоне;
- сечения многомерных разбиений.

Должны быть созданы сервисы, генерирующие различные геометрические объекты:

- платоновы, архимедовы и другие классические многогранники;
- пирамиды, конусы, призмы и антипризмы на основе заданных многоугольников;
- зоноэдры и параллелоэдры по заданным наборам векторов;
- классические решетки;
- различные специальные виды многогранников (например, многогранник, все грани которого остроугольные треугольники).

Должны быть созданы сервисы, преобразующие форматы данных и проверяющие определенные условия. Эти сервисы должны обеспе-

чивать прозрачное объединение различных компонент и по возможности должны быть скрыты от пользователя.

- Преобразование между различными форматами представления выпуклых многогранников (OFF, M, VRML, EIG и др.).
- Выборки и срезы из многогранников: все вершины, все ребра, все грани, грани соседние данной, вершины, соседние заданной.
- Проверка выполнения различных условий: корректность многогранника, ориентированность граней, выпуклость, простота невыпуклого многогранника, простота вершин, треугольность граней и т. п.
- Проверка наличия заданной точки внутри отрезка, многоугольника (на плоскости и в пространстве), многогранника.
- Проверка наличия в представлении многогранника дополнительных параметров: ребер как граней (двуугольников), цветов вершин и ребер, дополнительных свойств.
- Преобразование между различными типами данных: реберный остов многогранника как граф, развертка многогранника.

7.2. Визуализация

Должны быть сделаны следующие визуализирующие системы:

- Показ 2D графиков, графов, множеств и разбиений.
- Показ 3D многогранников.
- Показ 2D и 3D срезов многомерных объектов.

Все системы должны уметь раскрашивать объекты по заданным алгоритмам (в частности, должна быть возможность раскрашивания граней, ребер, вершин) настраивать отображение тех или иных объектов, иметь гибкие возможности навигации и манипуляции объектами.

7.3. Интерактивность

Среда должна позволять вводить различные числовые параметры, как при помощи численного формата, так и в виде координат точек (включая 3d, которые, по видимому, стоит показывать в виде двух ортогональных плоских проекций).

Должен быть удобный интерфейс для навигации по плоскости, вращения трехмерных объектов и т. п.

7.4. Реализация

Систему предполагается делать на основе среды IARnet. Предполагается подключить к ней одни из лучших существующих открытых библиотек antiprism [16] и polymake [17]. А так же другие отдельные ресурсы, существующие в сети Интернет, среди которых стоит отдельно выделить java-апплет LiveGraphics3D [18], отображающий многогранники, заданные в формате пакета Mathematica). Далее предполагается самостоятельно написать универсальную среду компоновки интерфейса из различных модулей. Написать среды скриптового и визуального программирования алгоритмов в этой среде. Эти среды должны быть по возможности основываться на существующих библиотеках —

- ввод и отображение формул,
- интерпретаторы скриптовых языков,
- системы визуального построения алгоритма в виде блок-схем и им подобных.

К сожалению полностью удовлетворяющих требованиям среды библиотек на данный момент не найдено, в данный момент ведется поиск соответствующих библиотек.

Однако понятно, что библиотеки, полностью удовлетворяющей необходимым требованиям, вряд ли будет найдено.

8. Коммерческая составляющая

Подобная система в принципе может интегрировать в себя пакеты с различными лицензиями. Однако на данном этапе не предполагается делать систему учета взаимного использования ресурсов. Хотя эта часть работы представляется весьма важной и в дальнейшем она обязательно потребуется, но первоочередной задачей начального этапа развития является набор достаточной массы компонент, пользователей и способов использования. На этом этапе достаточно обходиться открытыми компонентами, а для запуска отдельных трудоемких задач договариваться отдельно.

Так же есть идея создать лицензию ПО, естественно интегрирующейся как с распределенными вычислениями, так и с научной средой. Это должно быть программное обеспечение, которым можно свободно пользоваться, но с обязательным цитированием ссылки на это ПО при публикации. Если данная компонента использует другие компоненты с подобной лицензией, то эту программу можно рассматривать как «статью» ссылающуюся на другие «статьи», то есть компоненты, вызываемые из данной.

Развивая среду таким образом можно создавать аналог индекса цитируемости для статей.

В данном случае точнее сказать индекс использования.

9. Заключение

Автор предполагает, что подобная среда, будучи разработанной, должна стать весьма популярной и, начиная с какого-то момента, будет наращиваться самостоятельно.

На данный момент автором разрабатываются некоторые прототипы подобной системы. Во-первых, внутренняя система, интегрирующая пакет antiprism и своих разработки для собственных исследовательских задач комбинаторной геометрии. Во-вторых, разрабатываются демонстрационные примеры применения IARnet для решения математических задач. Эти примеры в данное время находятся в стадии разработки, однако в ближайшее время будут опубликованы.

На этих прототипах отыгрывается концепция распределенной системы, отрабатываются различные технологии.

Литература

1. *Емельянов С. В., Афанасьев А. П., Волошинов В. В., Гринберг Я. Р., Кривцов В. Е., Сухорослов О. В.* Реализация Grid-вычислений в среде IARnet // Информационные технологии и вычислительные системы. М.: Институт микропроцессорных вычислительных систем РАН, 2005. №2. С.61–75.
2. RMI / http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
3. Maxima / http://maxima.sourceforge.net
4. Mathematica / http://www.wolfram.com
5. MATLAB / http://www.mathworks.com
6. Maple / http://www.maplesoft.com

7. JavaScript /http://en.wikipedia.org/wiki/JavaScript
8. ECMAScript / http://en.wikipedia.org/wiki/ECMAScript, http://tools.ietf.org/html/rfc4329
9. Python / http://www.python.org
10. Ruby / http://www.ruby-lang.org
11. Haskell / http://www.haskell.org
12. AJAX / http://en.wikipedia.org/wiki/Ajax_(programming)
13. Yahoo pipes / http://pipes.yahoo.com/pipes
14. RSS / http://en.wikipedia.org/wiki/RSS_(file_format)
15. Antiprism / http://www.antiprism.com
16. polymake / http://www.math.tu-berlin.de/polymake
17. LiveGraphics3D / http://www.vis.uni-stuttgart.de/~kraus/LiveGraphics3D