ЛАБОРАТОРИЯ СИНТЕЗА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Зав. лабораторией д.т.н. Малышкин В. Э.

Важнейшие достижения

Разработка и реализация алгоритма балансировки трассы для повышения производительности исполнения программ при воспроизведении трасс в системе LuNA

д.т.н., проф. В.Э. Малышкин, к.т.н. В.А. Перепёлкин, А.С. Лямин

Воспроизведение трасс — это ранее разработанная техника повышения эффективности выполнения автоматически конструируемых параллельных программ численного моделирования в системе LuNA. Её суть в том, что параллельная программа конструируется и исполняется под управлением исполнительной системы, которая обеспечивает динамические свойства программы. Информация о ходе исполнения программы фиксируется в форме трассы, достаточной для повторного воспроизведения хода исполнения программы без исполнительной системы. Отсутствие накладных расходов на работу исполнительной системы обуславливает увеличение производительности исполнения программы. В работе предлагается дальнейшее развитие этой техники, которое состоит в предварительной модификации трассы таким образом, чтобы распределение вычислительной нагрузки по узлам мультикомпьютера было более равномерным (рис. 1). Предложенный алгоритм позволил (таблица 1) существенно повысить эффективность вопроизведения трасс в системе LuNA на примере задачи моделирования эволюции протопланетного самогравитирующего диска методом частиц-в-ячейках (PIC). Показано, что предложенный алгоритм также позволяет автоматически распределять вычисления по узлам мультикомпьютера, адаптировать трассу к другому количеству вычислительных узлов, чем при исходном исполнении, а также применяться итеративно к последующим воспроизведениям трассы. Значимость полученных результатов состоит в том, что предложенный алгоритм позволяет повышать эффективность автоматически конструируемых параллельных программ, тем самым расширяя множество задач, где ручное параллельное программирование может быть вытеснено менее трудоёмким автоматическим конструированием параллельных программ.

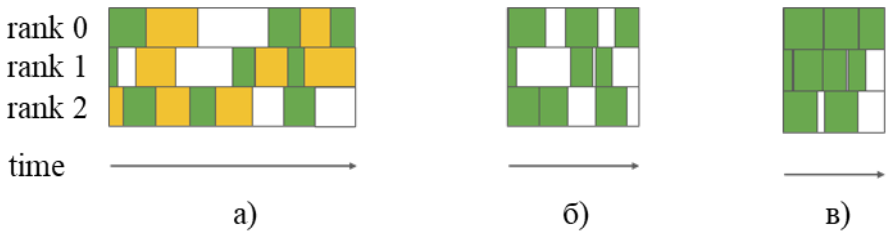


Рис. 1. Схема загрузки мультикомпьютера во времени: полезная нагрузка (зелёный), накладные расходы (жёлтый) и простои (белый) при обычном исполнении программы (а), в режиме воспроизведения трассы (б) и при воспоизведении сбалансированной трассы

Таблица 1. Время выполнения астрофизического численного эксперимента методом PIC в системе LuNA и в режиме воспроизведения трассы (LuNA-TP).

LuNA	LuNA-TP	LuNA-TP с балансировкой трассы (параметр — шаг квантования)				
		0.5 с.	1 с.	5 с.	10 с.	15 с.

601.185	136.055	127.94	115.685	110.503	100.808	136.340

Результаты исследований опубликованы в работе:

1. Malyshkin, V., Perepelkin, V., Lyamin, A. (2023). Trace Balancing Technique for Trace Playback in LuNA System // Parallel Computing Technologies. PaCT 2023. Lecture Notes in Computer Science, vol 14098. Springer, Cham. https://doi.org/10.1007/978-3-031-41673-6_4

Результаты исследований представлены на конференции:

1. 17th Int. Conf. On Parallel Computing Technologies, Aug 18-25, 2023, Astana, Kazakhstan.

Отчет по этапам научно-исследовательских работ, завершенным в 2023 г. в соответствии с планом НИР института

Проект НИР номер 1 "Суперкомпьютерные технологии решения больших задач естествознания, математические модели, методы анализа и оптимизации сложных информационных систем".
Номер государственной регистрации НИР FWNM-2022-0005.

Руководители: д.ф-м.н. Марченко М. А., к.ф-м.н. Черных И. Г.

Раздел 1. "Высокопроизводительные вычисления. Создание методов, алгоритмов, инструментальных средств и пакетов прикладных программ для вычислительных систем сверхвысокой производительности".

Руководитель – д.т.н Малышкин В. Э.

Этап 2023 г. Блок 2 "Конструирование экспериментальной БАЗ в численном моделировании на суперкомпьютерах и экспериментальное исследование библиотеки клеточно-автоматных топологий, применение программной реализации, использующей библиотеку, для решения задач моделирования".

2. Разработана эффективная фрагментированная программа решения краевой задачи фильтрации двухфазной жидкости.

При автоматическом конструировании параллельных программ по их высокоуровневой спецификации возникает алгоритмически труднорешаемая проблема обеспечения высокой эффективности конструируемых программ. Важную роль играют исследования конкретных примеров прикладных программ на предмет возможностей автоматического обеспечения их высокой эффективности за счёт учёта специфичных для данной предметной области особенностей прикладного алгоритма, вычислителя и обрабатываемых данных. Такое исследование было выполнено для краевой задачи фильтраци двухфазной жизкости. А именно, была разработана вручную эффективная реализация последовательной программы с использованием традиционных средств параллельного программирования (MPI), была разработана и вручную оптимизирована фрагментированная программа на базе системы LuNA, а также были доработаны средства управления памятью в системе LuNA для обеспечения более высокой эффективности параллельной программы системой LuNA. Исследование количественно показало, насколько автоматически сконструированная программа отстаёт по производительности от параллельной программы, сконструированной вручную — примерно в 1,5–2,5 раза (рис. 1), что является хорошим результатом для систем такого класса, как LuNA. Также был выполнен анализ результатов экспериментального исследования, который позволил сформулировать, какие системные алгоритмы следует улучшать для достижения более высокой эффективности. Выполненная ручная оптимизация LuNA-программы может быть использована в дальнейшем для разработки алгоритмов автоматической оптимизации аналогичных программ, конструируемых системой LuNA.

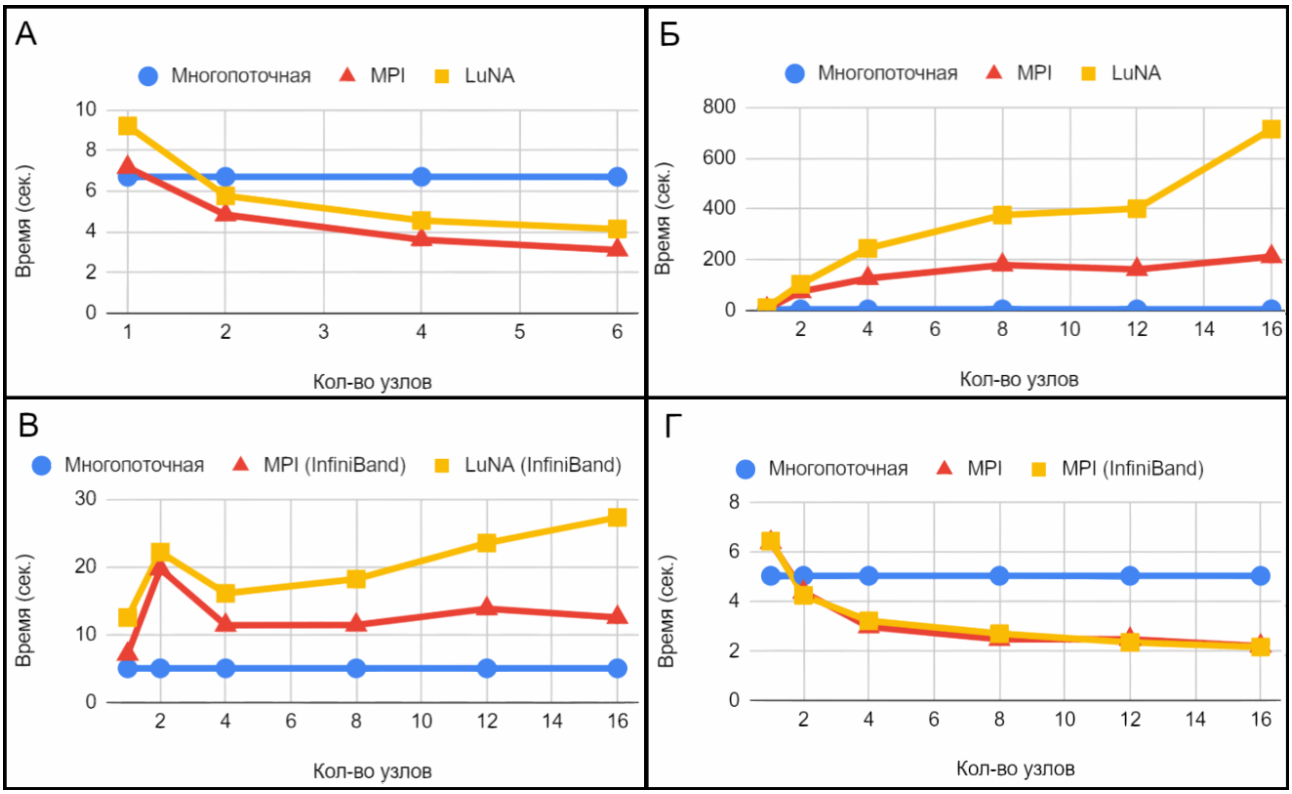


Рис. 1. Графики результатов тестирования MPI и LuNA реализаций на разных кластерах, А — на кластере НГУ, Б — на кластере МВС, В — на кластере МВС с использованием InfiniBand, Г — на кластере МВС без учета коммуникаций

3. Предложен мультиагентный подход к повышению эффективности исполнения фрагментированных программ в системе LuNA.

При автоматическом конструировании параллельных программ по их высокоуровневой спецификации с обеспечением динамических свойств программ (например, динамической балансировки нагрузки на узлы мультикомпьютера) возникает задача уменьшения накладных расходов на работу исполнительной системы, обеспечивающей динамические свойства. Суть проблемы состоит в том, что высокоуровневая спецификация допускает множество допустимых вариантов выполнения вычислений, и выбор конкретного варианта подразумевает принятие множества решений касательно порядка выполнения вычислительных операций, распределения данных по узлам мультикомпьютера и пр. При этом часть решений требуется принимать динамически (в исполнительной системе), а часть статически, на этапе компиляции, чтобы уменьшить накладные расходы на работу исполнительной системы. Это, в свою очередь, требует разработки модели вычислений, которая позволила бы совмещать статическое и динамическое принятие решений и могла быть эффективно реализована. Такая модель, учитывающая специфику фрагментированного программирования, была предложена на основе мультиагентного подхода. В её основе лежит идея преобразования исходного представления фрагментированного алгоритма в мультиагентную программу, которая генерируется на языке C++, что позволило использовать мощный инструментарий компиляции традиционных программ для снижения накладных расходов во время выполнения. Предложенный подход позволил сочетать статическое принятие решений (их результат генерируется в виде программ агентов) с динамическим (процесс принятия решения вкладывается в программу агента). Применение данного подхода показало существенное увеличение эффективности программ, конструируемых системой LuNA.

4. Создана база семантических ошибок во фрагментированных программах для системы LuNA.

Разработаны алгоритмы обнаружения сематических ошибок во фрагментированных программах для использования в инструментальных средствах статического анализа. Алгоритмы пригодны для анализаторов на базе графа зависимостей по данным (DDG - Data Dependency Graph) и абстрактного синтаксического дерева (AST – Abstract Syntax Tree). Также модернизирован метод статического анализа на базе генерации Prolog-программы по исходной фрагментированной программе с последующими запросами и анализом вывода на данные запросы.

Система автоматизированного конструирования параллельных программ LuNA и одноименный язык реализуют технологию фрагментированного программирования. Ключевыми понятиями для системы являются фрагмент кода (ФК), фрагмент данных (ФД) и фрагмент вычислений (ФВ). Ввиду особенностей языка LuNA-программам свойственны специфические семантические ошибки, не характерные для стандартных языков и технологий параллельного программирования. Была создана база данных с ошибками, их описанием и исходным кодом, их воспроизводящим. На текущий момент база содержит 22 различных ошибок. В частности:

- попытка использования неинициализированного фрагмента данных;
- повторная инициализация фрагмента данных;
- неиспользуемый фрагмент данных;
- несоответствие типов аргументов при вызове атомарного фрагмента кода;
- и др.

Для обнаружения семантических ошибок в программах, использующих различные языки и технологии, как правило разработчики на практике применяют интерактивную диалоговую отладку. Однако для параллельных программ и программ для таких систем, как LuNA, этот подход слабо применим по ряду причин. Соответственно требуется развитие автоматизированных методов, таких как статический анализ, автоматизированный контроль корректности во время исполнения и по собранной трассе («post-mortem» анализ), сравнительная отладка, Model Checking и др.

В течение года разрабатывались и реализовывались алгоритмы статического анализа фрагментированных LuNA-программ на базе AST и DDG. Созданный анализатор на базе AST способен обнаружить такие ошибки, как:

- импорт двух функций под одним именем;
- отсутствие функции main;
- ФД с одинаковыми именами в одной области видимости;
- несовпадение количества аргументов при объявлении ФК и его вызове;
- и др.

Анализируя DDG, который в свою очередь строится из AST, второй разрабатываемый анализатор обнаруживает такие ошибки, как повторная инициализация ФД и отсутствие использования проинициализированного ФД.

В качестве примера ниже продемонстрирован DDG (рис. 1) для программы из листинга 1:

Листинг 1. Пример ошибочной LuNA-программы

```
import c_print(real) as print;          // импорт атомарных ФК на C/C++
import c_init(real, name) as init;   // инициализация ФД (второго аргумента)
                                        // вещественным числом
import c_bar(string) as bar;
```

```
sub main(int a){
        df a, b, c, d, e;  // объявление ФД
        foo(a);
        bar(1);
        init(1, b);
        init(2, b);
        print(e);
}

sub foo(name x){
        init(1, x);
        print(x);
}
```

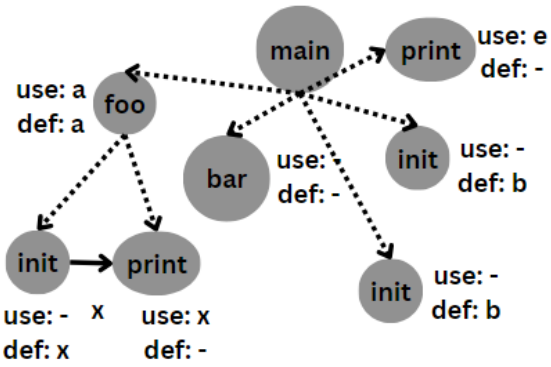


Рис. 1. Граф зависимостей по данным для программы (1)

В этом графе сплошными стрелками указаны дуги зависимостей по данным, а пунктирными стрелками – нахождение одной вершины внутри блока второй (например, foo и bar находятся внутри блока main). Выполняя поиск по графу можно без труда обнаружить 2 вершины, где инициализируется ФД b. Поскольку LuNA допускает лишь единственное присваивание, такая ситуация является ошибочной.

В рамках третьего разрабатываемого инструмента статического анализа LuNA-программ было решено использовать язык логического программирования Prolog. За минувший год были модифицированы алгоритмы генерации набора фактов на Prolog (часть базы знаний), описывающих инициализацию и использование ФД в исходной LuNA-программе.

В данный момент реализована генерация базы знаний для ограниченного подмножества языка LuNA: поддерживаются вызовы ФК (как атомарных, так и структурированных) с параметрами типа name и параметрами примитивных типов, и оператор for. Границы for могут быть как константами, так и выражениями, содержащими константы и одиночные ФД (например, 0, 1 + 1, 2 * N - 1).

При помощи разработанных для Prolog-analyzer алгоритмов обнаруживаются такие ошибки, как использование неинициализированных ФД, несоответствие границ циклов инициализации и использования ФД.

5. Продолжается разработка библиотеки управления распределенными данными Didal (Distributed Data Library), которая используется для создания параллельных фрагментированных программ для вычислительных машин с общей и распределенной памятью. На данный момент с использованием библиотеки созданы параллельные фрагментированные реализации таких задач, как:

- моделирование течений в релятивистской гидродинамике
- моделирование пылевого облака с помощью метода частиц-в-ячейках
- моделирование взаимодействий частиц методом молекулярной динамики

Проведено тестирование эффективности полученных параллельных реализаций.

В технологии фрагментированного программирования параллельная программа представляется как множество фрагментов данных и вычислений; параллельное выполнение программы осуществляется за счет параллельного выполнения ее фрагментов вычислений. Данный подход позволяет автоматизировать такие моменты выполнения параллельной программы как распределение данных и вычислений по вычислительным узлам, динамическая балансировка нагрузки, восстановление программы после сбоев и др..

Библиотека Didal (Distributed Data Library) предназначена для упрощения создания эффективных параллельных фрагментированных программ для вычислительных машин с общей и распределенной памятью. Библиотека не требует использования дополнительного языка (в отличие, например, от системы LuNA — Language for Numerical Algorithms), а получаемые фрагментированные программы могут обладать достаточно высокой эффективностью. Также библиотека нацелена в первую очередь на фрагментацию задач численного моделирования.

Результаты по разработанным ранее фрагментированным реализациям таких задач, как моделирование пылевого облака с помощью метода частиц-в-ячейках опубликованы в 2023 году в статье Malyshkin V., Schukin G.: Didal: Distributed Data Library for Development of Parallel Fragmented Programs.

В 2023 году с помощью Didal были разработаны фрагментированные реализации таких задач, как моделирование течений в релятивистской гидродинамике. Полученные предварительные результаты эффективности распараллеливания выполненной с помощью Didal фрагментированной реализации задачи моделирования течений в релятивистской гидродинамике приведены на Рис. 1 и 2. На Рис. 1 показана эффективность распараллеливания с фиксированным размером всей задачи (т. н. сильная масштабируемость). В некоторых случаях эффективность на 10-20% выше, чем эффективность других параллельных реализаций данной задачи (выполненных с помощью MPI и Coarray Fortran). На Рис. 2 показана эффективность распараллеливания с фиксированным размером задачи на процесс (т. н. слабая масштабируемость). В данном случае эффективность Didal реализации отличается максимум на ±10% от эффективности MPI или Coarray Fortran реализаций. По результатам исследований планируется публикация.

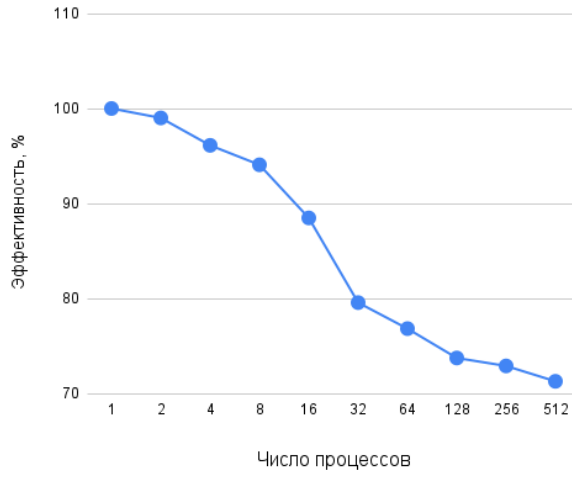


Рис. 1. RHD3D-DDL, сетка 256^3, 8^3 фрагментов, эффективность распараллеливания

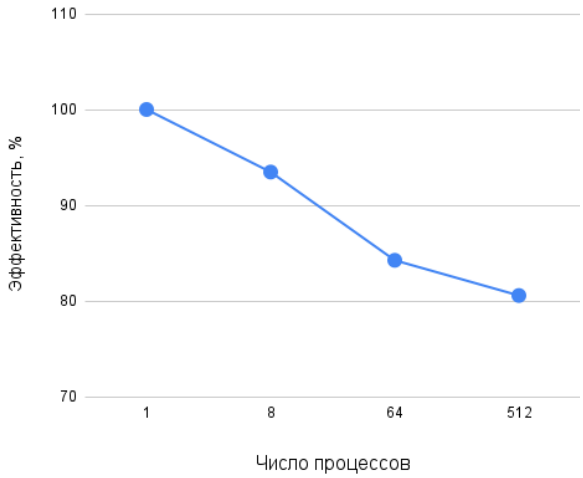


Рис. 2. RHD3D-DDL, сетка 128^3 на процесс, 8^3 фрагментов на процесс, эффективность распараллеливания

6. Сформулированы критерии, предъявляемые к программным средствам, предназначенным для реализации клеточных автоматов, проведен обзор таких программных средств. Ни одно из известных специализированных программных решений для построения клеточно-автоматных моделей не позволяет запускать расчеты на распределенных вычислительных системах. Выполнена параллельная реализация библиотеки клеточно-автоматных топологий для синхронных клеточных автоматов. Проведено ее тестирование на кластере МСЦ РАН, результаты которого показали эффективность параллельной реализации более 65% (рисунок 4). Максимальный размер клеточного массива, который удалось обрабатывать при эффективном использовании ресурсов кластера, составил 3.6 млрд. клеток, при этом задание запускалось на всех узлах кластера. Была реализована модель построения покрытия прямоугольного поля плитками домино, на которой и проводилось тестирование. На рисунке 5 приведен пример такого покрытия. Основой модели послужил двумерный синхронный клеточный автомат с квадратным соседством Мура ранга 2. Алфавит состояний клеток – булев. Операционный режим клеточного автомата – синхронный. Полученный результат подтвердил, что использование библиотеки клеточно-автоматных топологий в имитационном моделировании повышает качество программного кода.

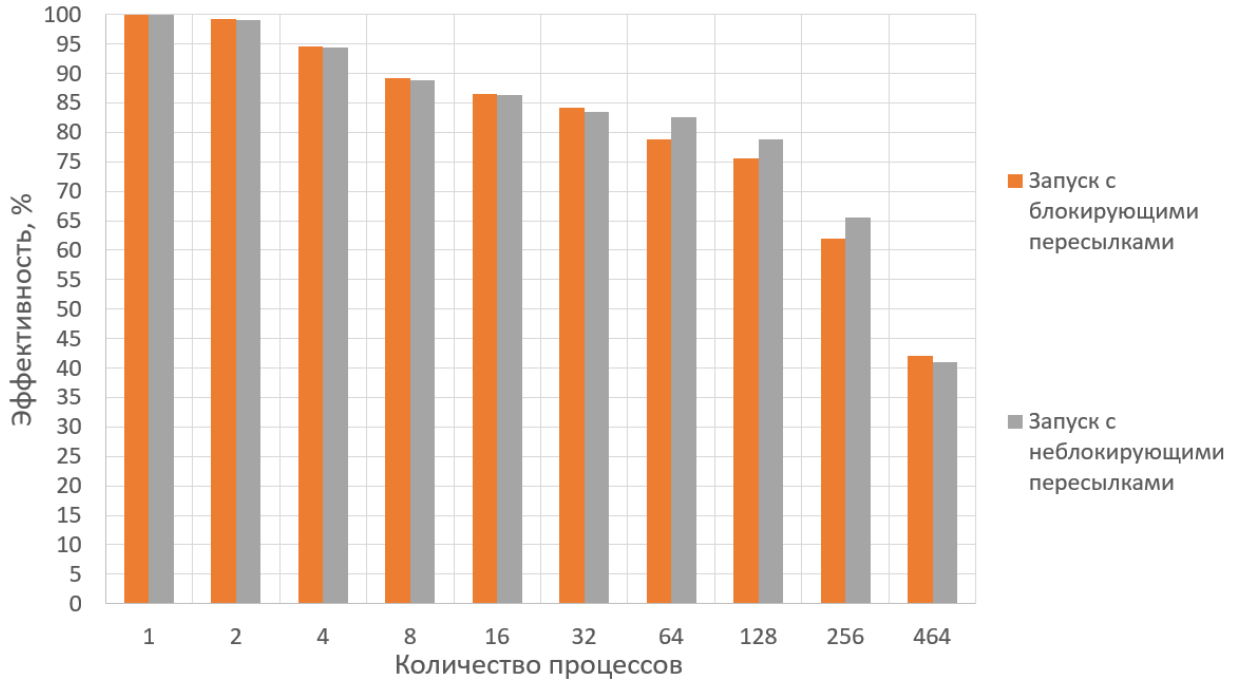
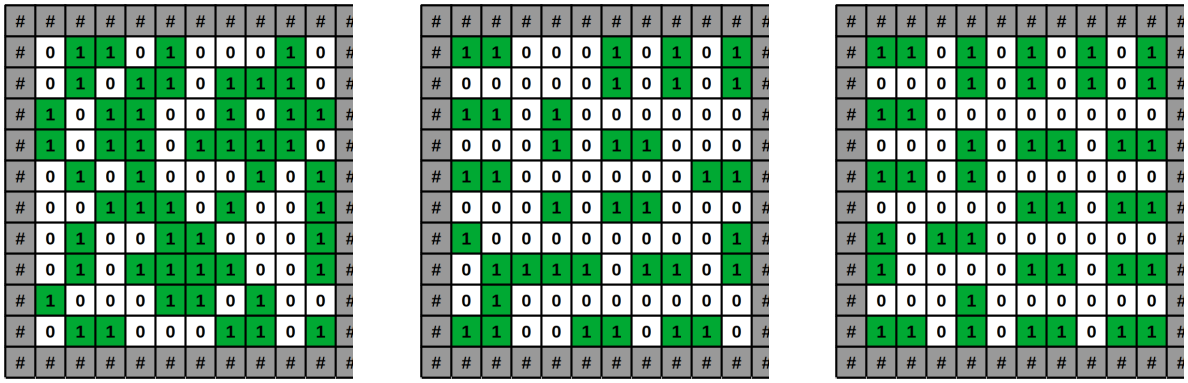


Рисунок 4 – Эффективность параллельной реализации библиотеки клеточно-автоматных топологий



а) начальное состояние

б) покрытие не сформировано (372 итерации)

в) покрытие сформировано (1715 итераций)

Рисунок 5 – Результат моделирования покрытия домино на поле 10*10

7. Спроектированы и реализованы модули экспертной системы Клио для управления знаниями:

- модуль автоматической генерации задач на основе шаблонов для областей математика», «физика», «информатика».
- базовый модуль автоматической генерации приложений-тренажеров (текстовый интерфейс).

модуль для работы с пользователями (авторизация, аутентификация, профиль, изучаемые проекты).

В 2023 году начаты работы по проекту Клио, целью которого является создание интеллектуальной экспертной системы помощи преподавателю, позволяющей автоматически генерировать различный контент для проверки знаний студентов с последующей автоматической проверкой ответов студентов. Исследование проводилось одновременно по трем направлениям:

- автоматическая генерация задач;
- автоматическая генерация вопросов;
- автоматическая генерация приложений-тренажеров.

По направлению автоматической генерации задач работа велась в соавторстве с Н.В. Лебедевым. Выполнен сравнительный анализ существующих методов генерации задач на основе шаблонов для областей «математика», «физика», «информатика», в результате которого для дальнейшей проработки был выбран подход на основе создания алгоритма для решения прямой или обратной задачи. В качестве технологии генерации задач использовались генерация на основе шаблонов и математических выражений. В результате была спроектирована и разработана интегрированная среда разработки (рис. 1) генераторов задач, позволяющая конструировать задачи различными методами: генерация условия и последующее решение некоторым алгоритмом, генерация ответа и алгоритмический вывод условий задачи, построение задачи в виде усложняющейся структуры с пересчетом текущего ответа.

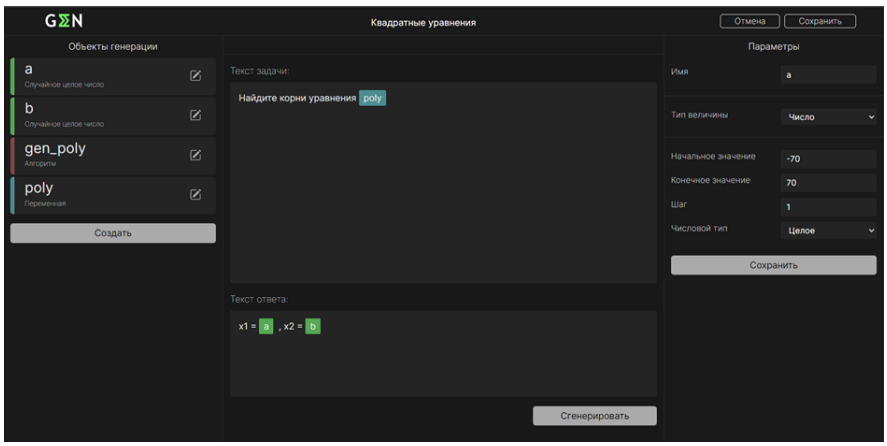


Рис. 1. Создание генератора

В интегрированной среде разработана библиотека готовых генераторов задач для некоторых из областей знаний (математика, информатика, физика) (рис. 2).

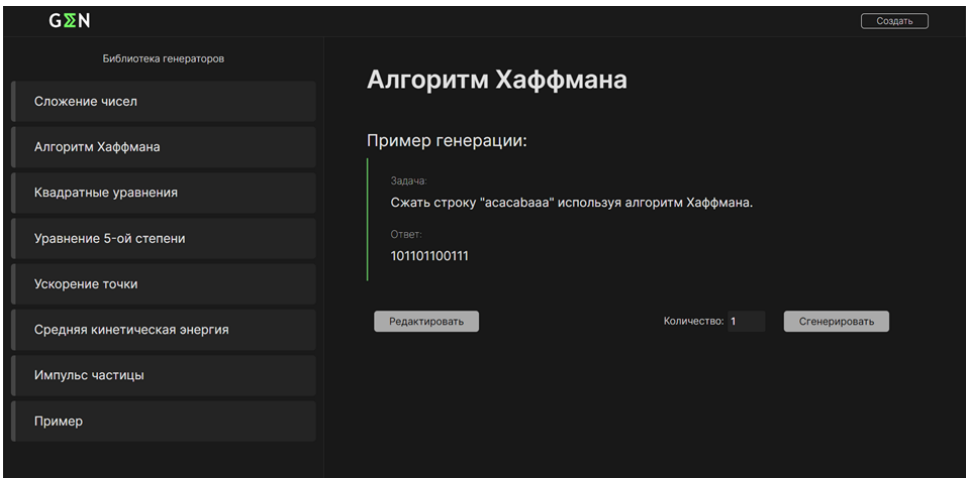


Рис. 2. Интерфейс библиотеки генераторов

По направлению автоматической генерация вопросов работа велась совместно с А.А. Балашовым. Выполнен сравнительный анализ различных методов представления знаний с точки зрения удобства генерации вопросов и качества сгенерированных вопросов. Рассматривались методы на основе онтологий, вычислительных моделей, и нейронных сетей. В качестве предметной области была выбрана «криптография», для которой была построена онтология в редакторе Protege (рис. 3).

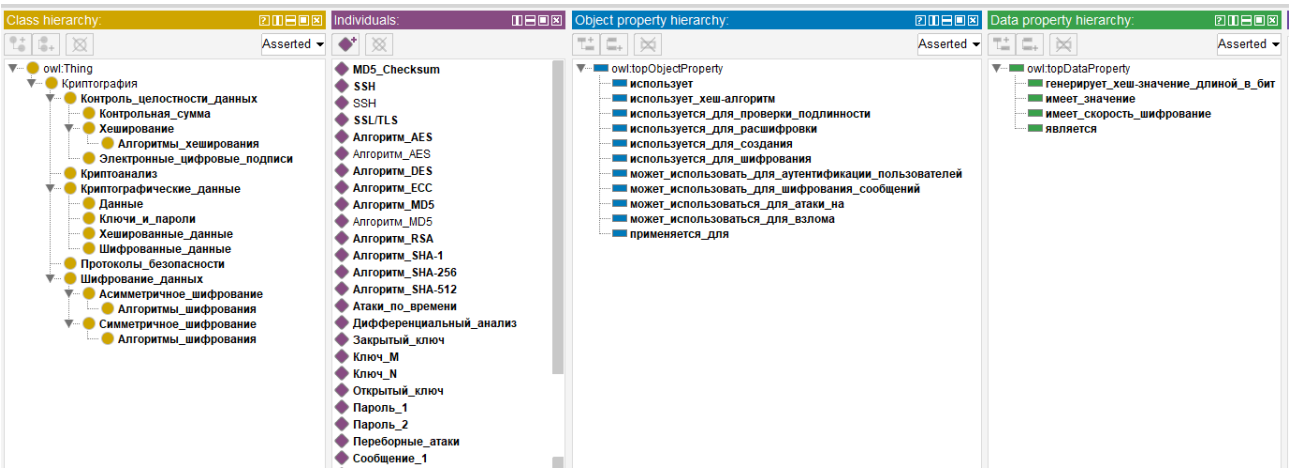


Рис 3. Представления знаний для выбранной области “Криптография” в виде онтологии

Предварительные результаты показывают, что хороший результат можно получить различными комбинациями этих методов, например, описав предметную область в свободной форме и уточнив запрос к нейросети ChatGPT с помощью этого описания, однако это требует дальнейшего исследования как с точки зрения проверки качества результата (увеличения выборки запросов), так и с точки зрения рассмотренных сетей (для полноты исследования необходимо добавить в сравнение нейросети Google Gemini, а также российские GigaChat и YandexGPT).

По направлению автоматической генерации приложений-тренажеров работа велась совместно с С.Д. Болдыревым. Разработано веб-приложение, которое станет основой для

будущей экспертной системы Клио и будет обеспечиваться взаимодействие пользователей с другими модулями системы, разработана схема для хранения тренажера, включающая в себя набор метаданных, декларативное описание UI и описание исполняемых функций. Разработан конструктор UI для тренажеров (рис. 4). Готовый тренажер можно выбрать из списка. Уже созданы тренажеры для изучения Машины Тьюринга (рис. 5), конечных автоматов, языка программирования SQL и некоторые другие.

Рис. 4. Редактор интерфейса тренажера

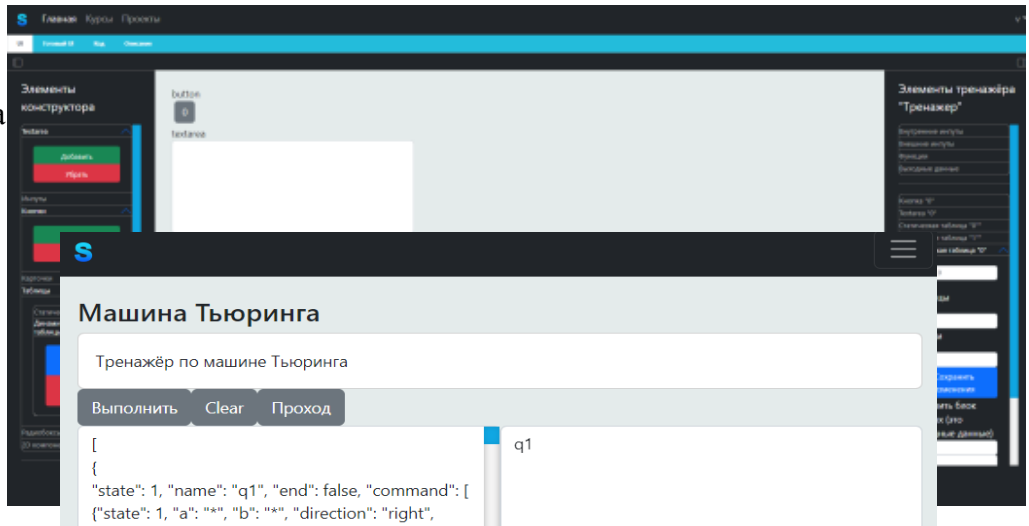


Рис. 5. Тренажер машины Тьюринга

8. Разработка концепции моделирования времени, отражающей потребность исследования развития систем с активными элементами.

Развитие методов проектного менеджмента при работе с нестабильными командами.

Разработка мелодических материалов для преподавания программирования и информатики в вузах с математической, физической и технической специализацией.

За отчетный период удалось обосновать корректность имитации динамических систем как поведения множества их элементов в пространстве активности последних. Отличительной особенностью пространства активности системы является представление активности ее элементов как частично упорядоченного множество состояний. Именно это отношение способно адекватно имитировать реальное время. В результате предлагается технологичный подход к выявлению фазового представления динамики развития процессов как системы, так и ее элементов.

За отчетный период исследования, связанные с методами менеджмента нестабильных команд достигли стадии, предполагающей экспериментальную проверку. Это обстоятельство, а также потребность эффективной поддержки дистанционной работы в связи с ковидной эпидемией стали причиной постановки эксперимента организации проектной деятельности двумя коллективами, один из которых формировался из студентов Новосибирского государственного университета, а другой — на базе кафедры информатики Алтайского государственного гуманитарно-педагогического университета. Специфика работы со студенческими группами как нельзя лучше подходит для проверки эффективности методик в условиях нестабильных команд. Результаты проведенного эксперимента показали, что для успешного решения разнообразных проектных задач нестабильными командами необходимы специально выделенные исполнители, выполняющие координационные функции. Этот факт подтверждает априорно выдвигаемую гипотезу. Другой результат, имеющий педагогическое значение — предложение принципиально новых методик работы со студентами, обучающихся дистанционно.

В течение ряда лет авторы указанных материалов преподавали программирование для студентов второго курса ММФ НГУ. Ими были выявлены различные подходы и изложению тематики, разработаны условия, выполнение которых необходимо для эффективного преподавания.

Публикации

Издания, включенные в реферативную базу данных Scopus

1. Malyshkin, V., Perepelkin, V., Lyamin, A. (2023). Trace Balancing Technique for Trace Playback in LuNA System // Parallel Computing Technologies. PaCT 2023. Lecture Notes in Computer Science, vol 14098. Springer, Cham. https://doi.org/10.1007/978-3-031-41673-6_4
2. Medvedev, Y., Kireev, S., Trubitsyna, Y. (2023). Expanding the Cellular Automata Topologies Library for Parallel Implementation of Synchronous Cellular Automata. In: Malyshkin, V. (eds) Parallel Computing Technologies. PaCT 2023. Lecture Notes in Computer Science, vol 14098. Springer, Cham. https://doi.org/10.1007/978-3-031-41673-6_8
3. Malyshkin, V., Schukin, G. (2023). Didal: Distributed Data Library for Development of Parallel Fragmented Programs. In: Malyshkin, V. (eds) Parallel Computing Technologies. PaCT 2023. Lecture Notes in Computer Science, vol 14098. Springer, Cham. P. 30-41. https://doi.org/10.1007/978-3-031-41673-6_3
4. Malyshkin V. (2023). Preface. In: Malyshkin, V. (eds) Parallel Computing Technologies. PaCT 2023. Lecture Notes in Computer Science, vol 14098. Springer, Cham. P. V. https://doi.org/10.1007/978-3-031-41673-6

Издания, включенные в библиографическую базу данных РИНЦ

1. Кудрявцев А. А., Малышкин В. Э., Нуштаев Ю. Ю., Перепелкин В. А., Спирин В. А. Эффективная фрагментированная реализация краевой задачи фильтрации двухфазной жидкости // Проблемы информатики. 2023. № 2. С. 45-73. DOI: 10.24412/2073-0667-2023-2-45-73
2. Малышкин В. Э., Перепелкин В. А. Мультиагентный подход к повышению эффективности исполнения фрагментированных программ в системе LuNA //"Проблемы информатики", 2023, № 3, с.55-67. DOI: 10.24412/2073-0667-2023-3-55-67.
3. Снытникова Т.В. Processing-in-Memory: текущие направления развития технологии //"Проблемы информатики", 2023, № 3, с. 37-54. DOI: 10.24412/2073-0667-2023-3-37-54
4. Снытникова Т. В. Библиотека реализации ассоциативных вычислений па графических ускорителях cuSTAR: представление данных для задач биоинформатики //"Проблемы информатики", 2023, № 1, с.60-68. DOI: 10.24412/2073-0667-2023-1-60-68.
5. Скопин И. Н. Модель времени для изучения развивающихся систем //"Проблемы информатики", 2023, № 1, с.12-32. DOI: 10.24412/2073-0667-2023-1-12-32.

Свидетельства о регистрации в Роспатенте

1. Свидетельство о государственной регистрации программы для ЭВМ № 2023662675 «Программный комплекс автоматизированного обнаружения семантических ошибок для программ в системе LuNA методом «посмертного» анализа». Авторы: Малышкин Виктор Эммануилович, Власенко Андрей Юрьевич, Мичуров Михаил Антонович (дата государственной регистрации в Реестре программ для ЭВМ – 13 июня 2023 г.)

Защиты диссертаций

1. Снытникова Т.В - кандидат технических наук. Тема: Эффективная реализация модели ассоциативных вычислений на графических ускорителях для решения задач на графах. Специальность: 2.3.5 - Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей. Дата защиты: 17.01.2023 г.

2. Перепёлкин В.А. - кандидат технических наук. Тема: Система LuNA автоматического конструирования параллельных программ численного моделирования на мультикомпьютерах. Специальность: 2.3.5 - Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей. Дата защиты: 21.02.2023 г.