НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информационных технологий
Кафедра параллельных вычислений
09.03.01 Информатика и вычислительная техника. Программная инженерия и компьютерные науки

Разработка и реализация специализированного алгоритма распределенной динамической сборки мусора в системе LuNA

Выполнил: Иванченко Данила Валерьевич, студент группы 19207 ФИТ НГУ

Руководитель: Власенко Андрей Юрьевич, доцент каф. ПВ ФИТ НГУ, к.т.н.

Соруководитель: Перепёлкин Владислав Александрович, ст. преп. каф. ПВ ФИТ НГУ

Рецензент: Карабцев Сергей Николаевич, к.ф.-м.н, зав.каф. ИБ КемГУ

23.06.2023

Новосибирск 2023

Введение в предметную область

Мусор – объект в памяти, созданный программой, который после некоторого момента времени больше не используется или не может быть достигнут программой, но все еще занимает память компьютера.

Сборка мусора – форма автоматического управления памятью, основной задачей которой является освобождение памяти, занятой мусором.

Распределенная сборка мусора - сборка мусора, которая происходит в системе, где вычисления распределены между несколькими компьютерами.

Актуальность

В системах, имеющих возможность динамического выделения памяти, должен быть механизм освобождения выделенной памяти.

Автоматизация программирования, в частности автоматический механизм управления памятью, позволяет снизить сложность написания программ для распределенных систем.

LuNA

LuNA является системой фрагментированного программирования, которая позволяет упростить написание параллельных программ. Данная система является подходящей площадкой для исследования проблемы распределенной сборки мусора из-за особенностей архитектуры, которые позволяют системе накапливать частные решения различных проблем в области распределенных систем.

Цели и задачи работы

Целью работы является разработка и реализация специализированного алгоритма распределенной динамической сборки мусора для итерационных задач в системе фрагментированного программирования LuNA.

Задачи:

1. Проанализировать существующие решения для сборки мусора.
2. Разработать алгоритм динамической сборки мусора в системе LuNA
3. Реализовать алгоритм
4. Провести тестирование реализации на реальных вычислительных задачах

Существующие решения

Алгоритмы, на основе подсчета ссылок

1. Стандартный алгоритм подсчета ссылок
2. Взвешенный алгоритм подсчета ссылок
3. Алгоритм на основе списка ссылок

Алгоритмы, на основе трассировки

1. Mark and Sweep
2. Алгоритм трассировки с использованием временных меток Хьюза

Вывод: существующие алгоритмы не закрывают проблему в полной мере, так как различные распределенные системы и задачи в этой области имеют различные требования и ограничения. Это приводит к необходимости разработать собственный специализированный алгоритм, который будет эффективным на узком классе практически значимых задач.

Постановка задачи

Фрагмент данных (ФД**)** – агрегат из переменных. Каждый фрагмент является неизменяемым и инициализируется единожды.

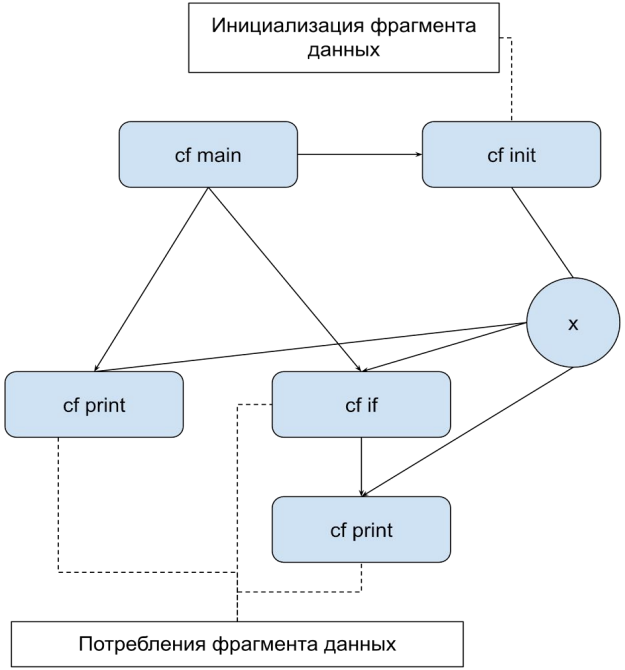
Фрагмент вычислений (ФВ**)** – исполняемая единица программы, содержащая описание входных и выходных фрагментов данных и исполняемого кода фрагмента без побочных эффектов.

Потребление фрагмента данных – использование фрагмента данных при исполнении кода фрагмента вычислений.

Жизненный цикл фрагмента данных – набор всех потреблений фрагмента данных.

Постановка задачи

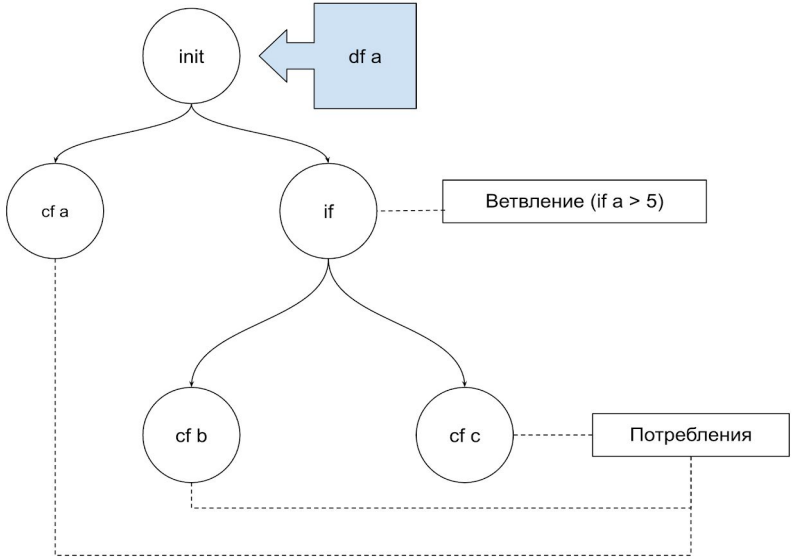
```
1. sub main()
 2. {
 3.     df x;
 4.     init(a);
 5.     cf print: print(a);
 6.
 7.     cf if: if(a > 0) {
 8.         print(a);
 9.     }
10. }
```



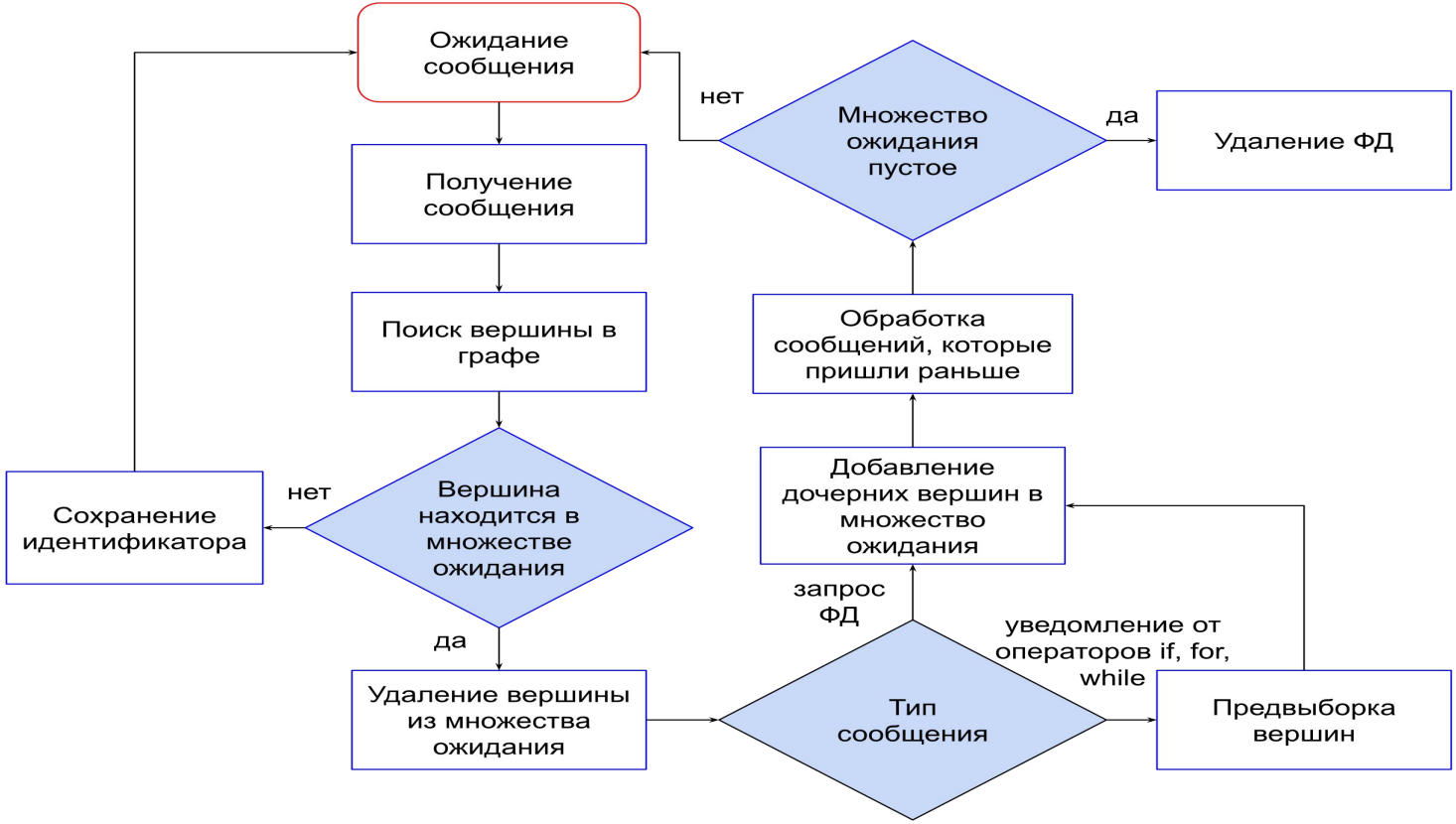
Основная идея алгоритма

Граф потреблений фрагмента данных - граф, вершинами которого являются операторы, соответствующие фрагментам вычислений, которые потребляют фрагмент данных, а дугами - информационные зависимости между соответствующими фрагментами вычислений.

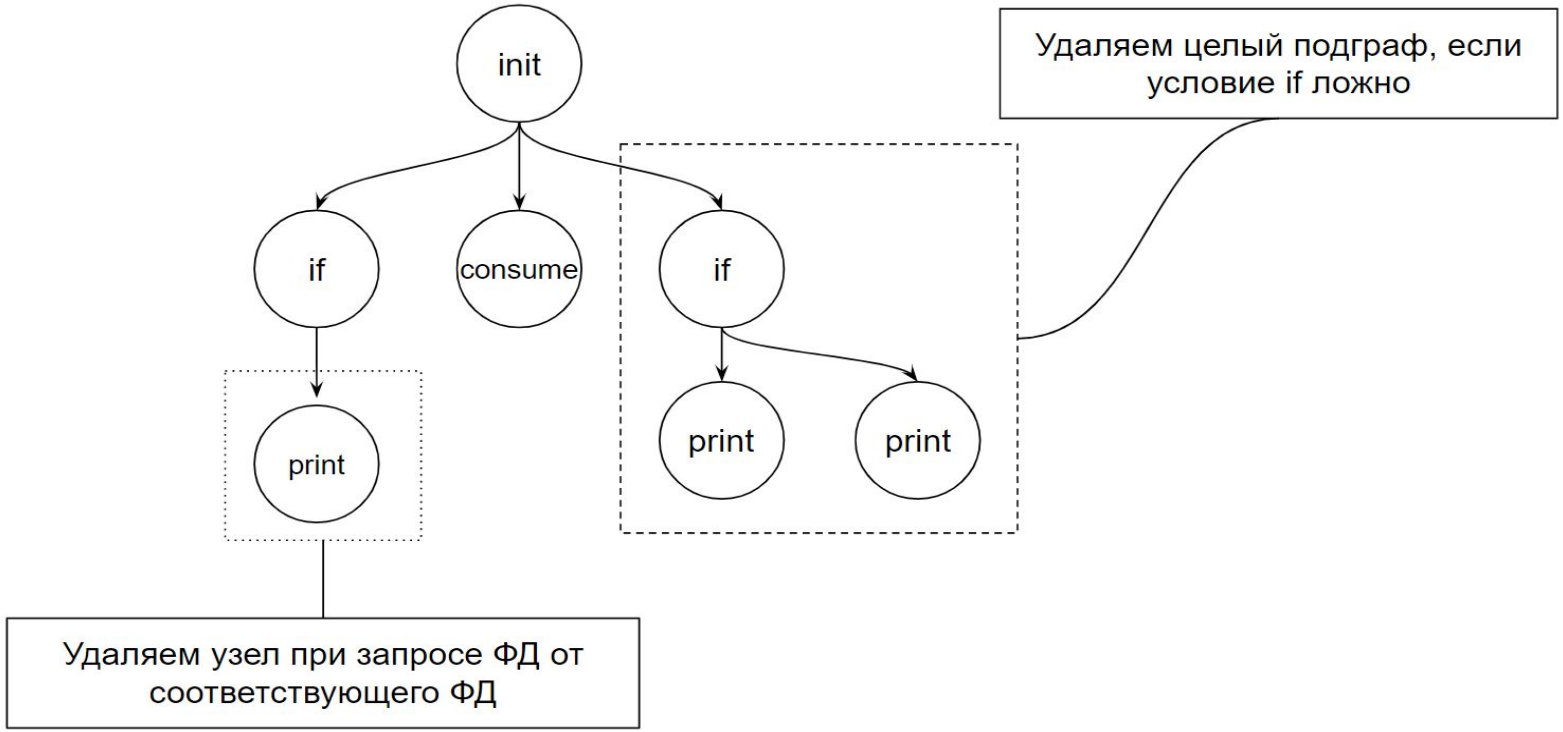
```
 1. sub main()
 2. {
 3.     df a;
 4.     init(a);
 5.     cf a: consume(a);
 6.
 7.     if(a > 0) {
 8.         cf b: print(a);
 9.         cf c: print(a);
10.     }
11. }
```



Описание алгоритма



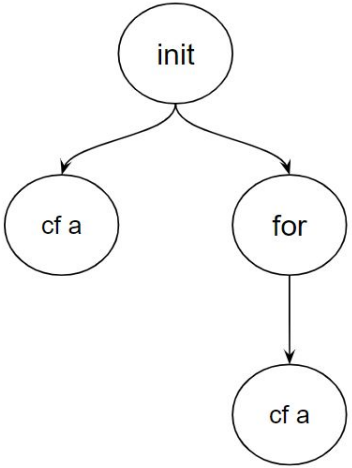
Описание алгоритма



Описание алгоритма.Индексируемые фрагменты данных

Требуется общий граф потреблений для индексируемых фрагментов данных.
При использовании косвенной адресации для получения значения индексов фрагмента данных необходимо запросить другие фрагменты данных.В случае косвенной адресации для удаления всех элементов необходима широковещательная рассылка.

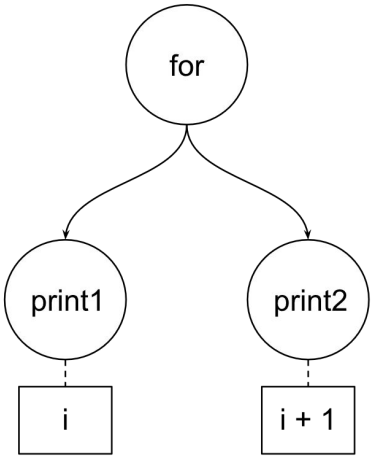
```
1. sub main()
2. {
3.     df A, x;
4.     init(A);
5.     init(x);
6.
7.     cf a: print(A[x – 1]);
8.     for i=0..x {
9.         cf b: print(A[i]);
10.    }
11.}
```

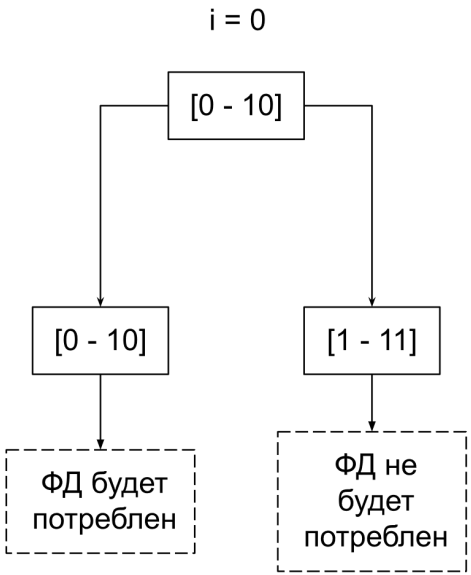


Описание алгоритма. Механизм предвыборки

1. Для индексируемых ФД вершина содержит выражения для индексов
2. При порождении ФВ операторы for, while отправляют диапазон для счётчика на узлы
3. Исходя из диапазона происходит выбор вершин, индексы которых попадают в диапазон
4. В случае вложенных циклов, полученные диапазоны наследуются дочерними вершинами(for, while, if)
5. В случае, если ФД не зависит от счетчиков циклов, то он должен потребиться N = кол-во порождающих итераций раз

```
1. for i=0..10 {
2.     print1(A[i+1]);
3.     print2(A[0]);
4. }
```





Реализация. Исполнительная система

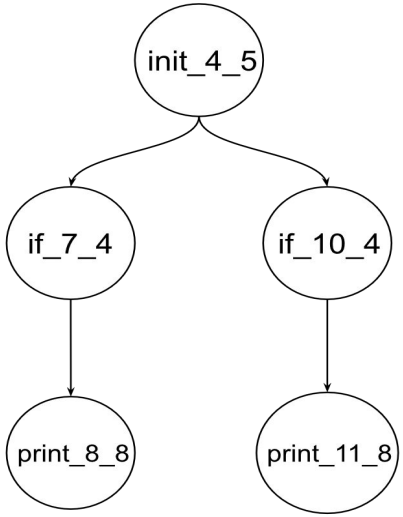
Были модифицированы следующие части исполнительной системы:

1. Добавлен ассоциативный массив, представляющий множество ожидания фрагментов данных на узле, где ключом является идентификатор ФД, а значением - набор указателей на вершины графа, соответствующих фрагментам вычислений, от которых ожидаются сообщения.
2. Был изменен формат сообщения о запросе фрагмента данных, в сообщение был добавлен идентификатор запрашивающего ФВ.
3. Добавлена функция поиска вершины по идентификатору, удаление вершины и добавление дочерних вершин.
4. Добавлены обработчики уведомлений от операторов if, for, while, которые реализуют механизм предвыборки.

Реализация. Транслятор

В транслятор была добавлена новая рекомендация для описания графа потребления ФД.

```
1. sub main()
2. {
3.     df x;
4.     init(x) @{gc_graph=>{init=>{if=>{print}, if{print}}}};
5.
6.
7.     if(...) {
8.         print(x);
8.     }
9.
10.     if (...) {
11.         print(x + 1);
12.     }
13. }
```



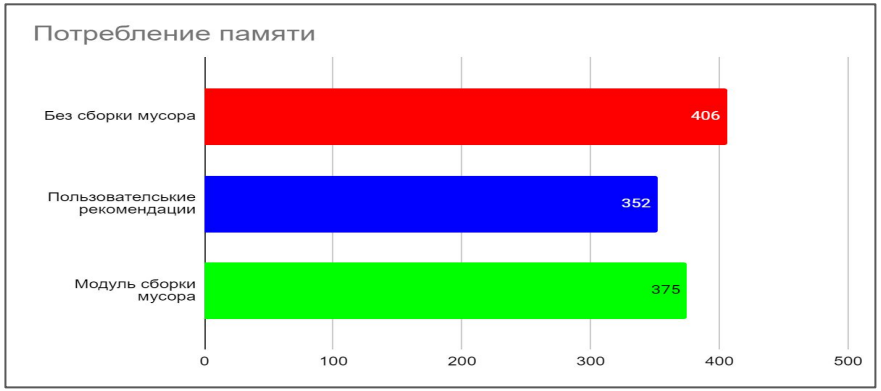
Тестирование

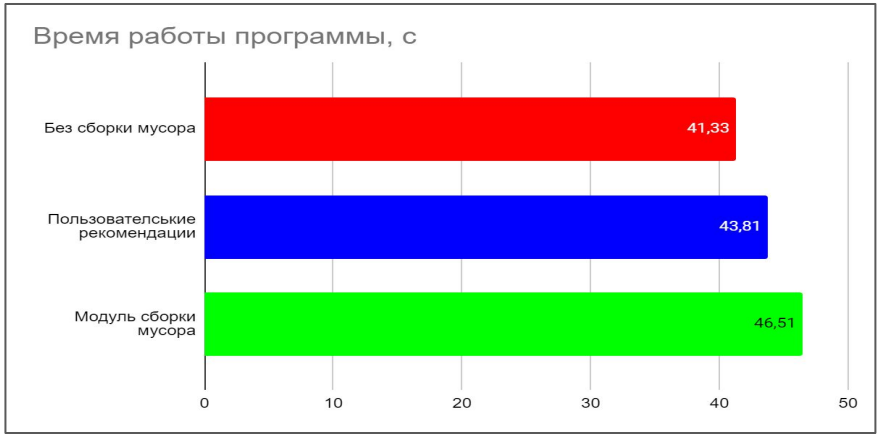
Для тестирования работы алгоритма были проведены два теста:

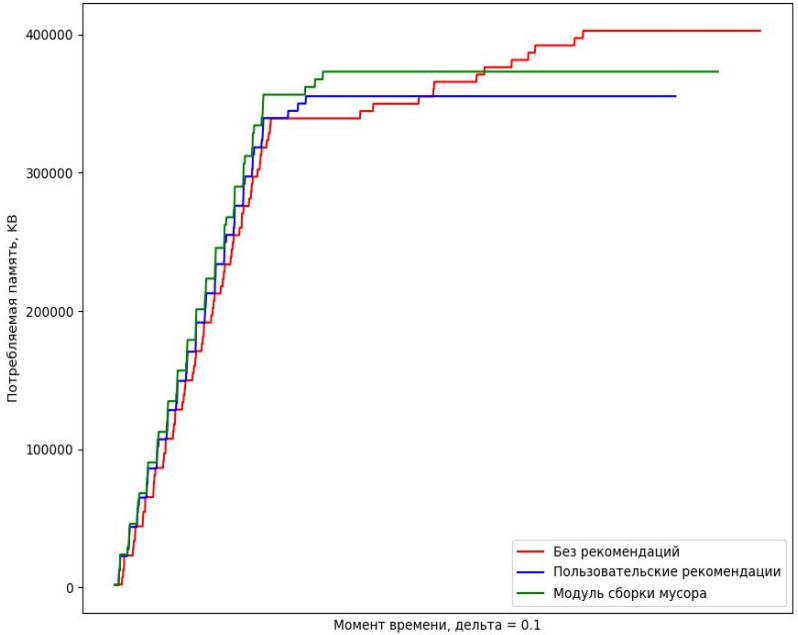
1. Вывод матрицы(выводятся элементы $a_{i,j}$ и $a_{i+1,\,j+1}$)
2. Моделирование самогравитирующего пылевого протопланетного диска методом частиц-в-ячейках.

   (Kireev S. A parallel 3D code for simulation of self-gravitating gas-dust systems // International Conference on Parallel Computing Technologies 2009. — Berlin, Heidelberg : Springer, 2009.— С. 406–413.)

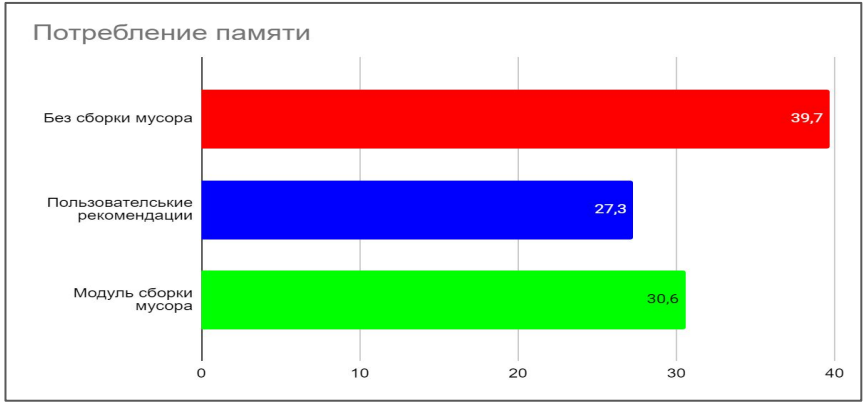
Результаты тестирования. Вывод матрицы

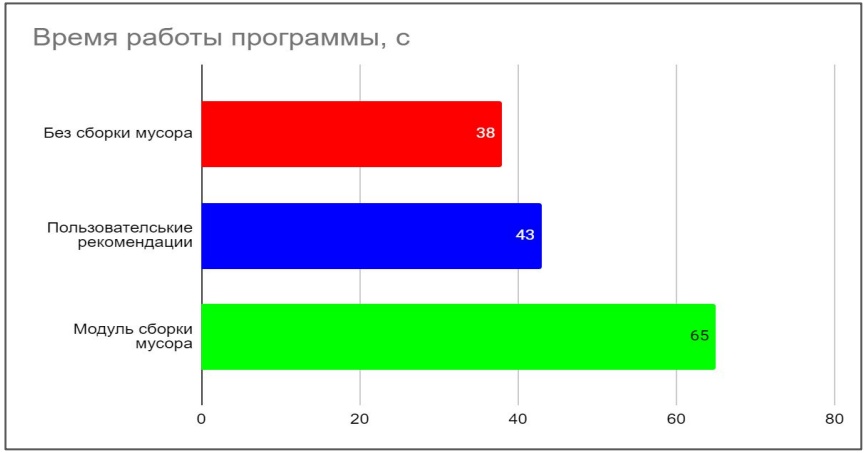


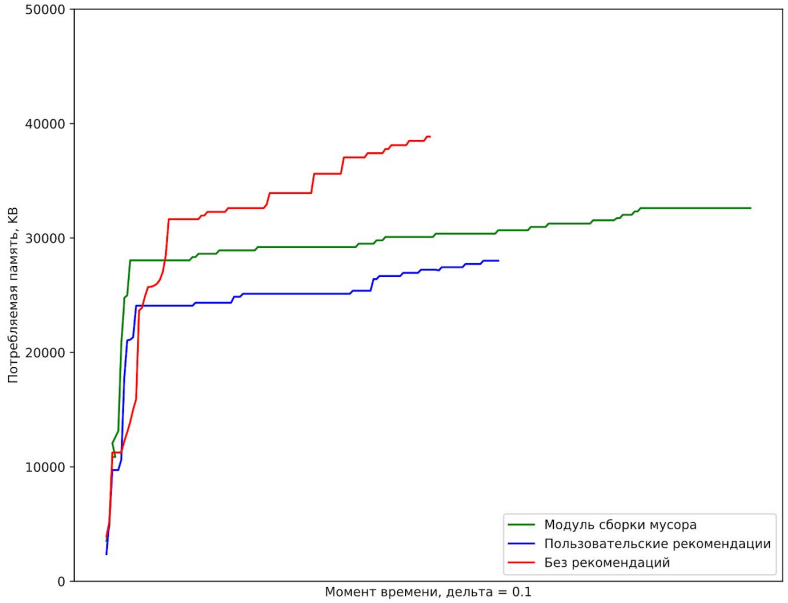




Результаты тестирования. Моделирование диска







Заключение

В ходе выполнения работы был разработан, реализован и встроен в систему LuNA специализированный алгоритм распределенной динамической сборки мусора, который снижает потребление памяти программой на итерационных методах численного моделирования.

Защищаемые положения:

1. Разработан и реализован специализированный алгоритм динамической распределенной сборки мусора в системе LuNA;
2. Проведено экспериментальное исследование работоспособности алгоритма динамической распределенной сборки мусора системе LuNA.

Планы на будущее

В дальнейшем планируется продолжить работу над сборкой мусора в системе фрагментированного программирования LuNA. Возможные направления развития:

1. Разработка динамического сборщика мусора, который эффективно работает с косвенной адресацией;
2. Разработка универсального гибридного сборщика мусора.

Апробация работы

Данная работа была опубликована на 61-й Международной научной студенческой конференции, г. Новосибирск, 2023 г. и была награждена дипломом третьей степени.

Спасибо за внимание!

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информационных технологий
Кафедра параллельных вычислений
09.03.01 Информатика и вычислительная техника. Программная инженерия и компьютерные науки

Разработка и реализация специализированного алгоритма распределенной динамической сборки мусора в системе LuNA

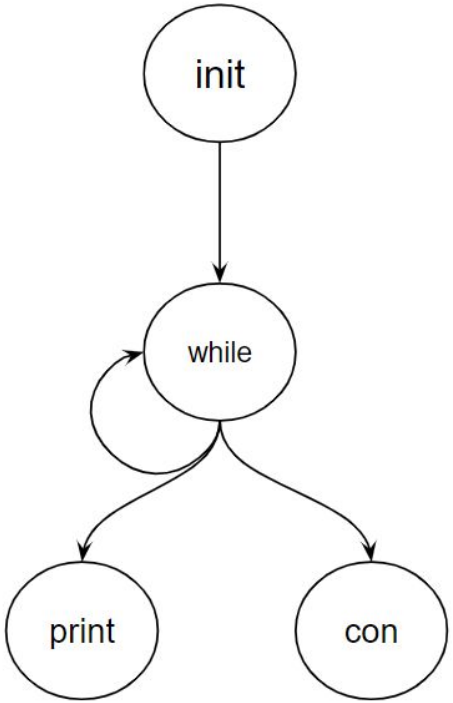
Выполнил: Иванченко Данила Валерьевич, студент группы 19207 ФИТ НГУ

Руководитель: Власенко Андрей Юрьевич, доцент каф. ПВ ФИТ НГУ, к.т.н.

Соруководитель: Перепёлкин Владислав Александрович, ст. преп. каф. ПВ ФИТ НГУ

Рецензент: Карабцев Сергей Николаевич, к.ф.-м.н, зав.каф. ИБ КемГУ

23.06.2023

Новосибирск 2023

```
1. sub main() {
 2.  df a;
 3.  init(a);
 4.
 5.  while(a[i] > 0), i=0..out {
 6.       cf a: print(a[i]);
 7.       cf b: consume(a[i]);
 8.  }
 9. }
```