ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)
Факультет информационных технологий
Кафедра параллельных вычислений

Разработка и реализация средств для визуального анализа исполнения фрагментированных программ

Выполнил: Голиков Максим Олегович

Введение

- При написании параллельных программ возникают такие задачи, как:
  - распараллеливание алгоритма;
  - равномерное распределение данных по вычислительным узлам;
  - балансировка нагрузки;
  - обеспечение коммуникаций между вычислительными узлами.
- Технология фрагментированного программирования и система LuNA
  - динамическая балансировка нагрузки;
  - установка коммуникационных взаимодействий между потоками исполнения;
  - настройка под имеющиеся аппаратные ресурсы.
- В связи со сложностью написания эффективных параллельных программ возникает потребность в оценке качества распараллеливания алгоритма. Критерием может служить время исполнения программы, однако оно не позволяет узнать причины плохой производительности
- Для определения причин неэффективности распараллеливания используются профилировщики - инструменты, собирающие некоторые характеристики исполнения программы, наиболее влияющие на время выполнения

Обзор средств профилирования параллельных программ

- В терминах операционной системы и терминах языка низкого уровня (C/C++/Fortran):
  - MPE (Jumpshot)
  - Extrae / Paraver
  - Intel Trace Analyzer and Collector
  - Scalasca (KOJAK)
  - CrayPat
  - HPCToolkit
- В терминах фрагментов:
  - LuNA Evlog Analyzer
- Вывод: не существует удобного средства профилирования фрагментированных программ

Цель и задачи

Цель работы: разработать интерактивное визуальное средство профилирования фрагментированных программ, написанных для системы LuNA.

Задачи:

- Выбрать и проанализировать характеристики исполнения фрагментированных программ.
- Определить способ получения характеристик исполнения.
- Спроектировать и реализовать систему визуального профилирования.

Характеристики производительности фрагментированных программ

- При профилировании программы ограничимся следующими вопросами:
  - оценка эффективности использования ресурсов
  - оценка потребления памяти программой
  - определение “горячих точек” программы
- Был предложен ряд характеристик исполнения, которые помогают ответить на вышеописанные вопросы, например:
  - “вычислительная нагрузка”
  - “дисбаланс нагрузки потоков исполнения”
  - “состояние потока исполнения”
  - характеристики передач фрагментов данных
  - потребление памяти фрагментами данных

Получение характеристик исполнения

- Для получения характеристик производительности фрагментированных программ было решено использовать логирование
- Был составлен список событий, необходимых для получения характеристик исполнения
- Были описаны алгоритмы расчета значений характеристик

Проект визуальной системы профилирования фрагментированных программ

- Характеристики исполнения можно разделить на два вида:
  - интегральные, визуализация в виде чисел
  - зависящие от времени, визуализация в виде графиков
- Разделы:
  - статистическая информация об исполнении
    - данные о вычислительной системе, время исполнения, количество фрагментов
  - оценка эффективности использования ресурсов
    - “вычислительная нагрузка”, “состояние потока исполнения”, дисбаланс нагрузки
  - оценка потребления памяти программой
    - графики потребления памяти, максимальное потребление
  - граф исполнения программы
    - восстановленное по логам дерево вызовов с характеристиками исполнения фрагментов вычислений

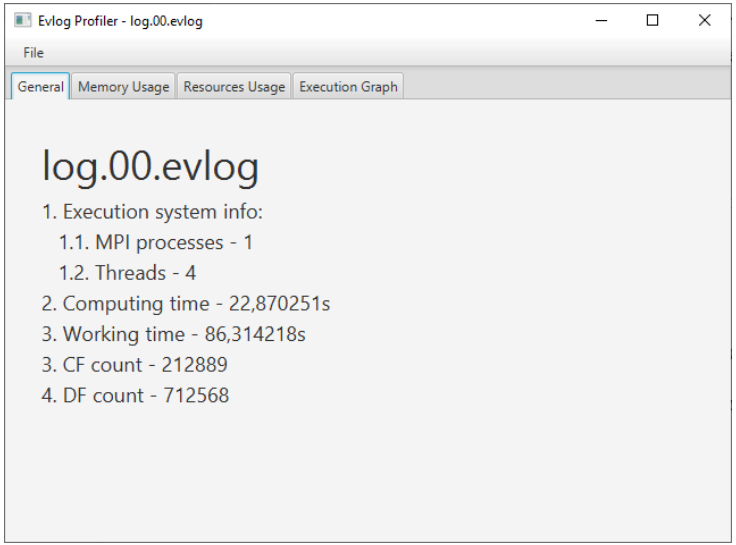
Реализация прототипа визуальной системы профилирования

- В рамках ВКР был реализован прототип визуальной системы профилирования, обеспечивающий часть запланированного функционала
- Прототип состоит из двух частей:
  - встроенный в систему LuNA модуль логирования. Данный модуль реализован на языке C++, как и текущая версия системы LuNA;
  - визуальное средство профилирования, анализирующее логи исполнения фрагментированных программ. Данное средство реализовано на языке Java с использованием графической библиотеки JavaFX.

Тестирование прототипа

- С помощью разработанного прототипа был проведен анализ производительности фрагментированной программы на языке LuNA, решающей модельную численную задачу решения уравнения Пуассона методом Якоби
- Входные параметры программы:
  - пространственная сетка 500 на 500 узлов
  - шагов по времени - 1000
  - фрагментация сетки - 4 на 4 блока
  - тип ячеек сетки - double
  - 4 потока исполнения
  - 1 процесс

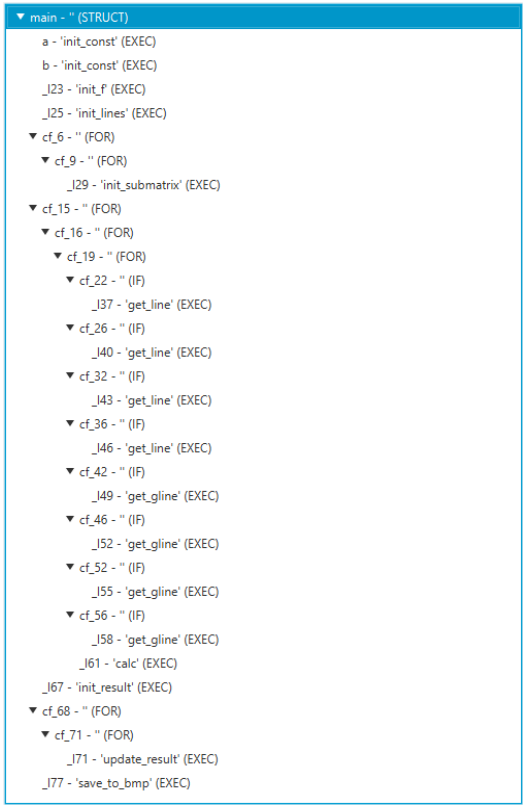
Тестирование



Статистическая информация об исполнении программы

Потребление памяти программой

Тестирование



Восстановленное дерево вызовов фрагментов вычислений

```
sub main() {
    df FX, FY, F, M, GL, GR, GU, GD, LL, LR, LU, LD, R;

    cf a: init_const($FRAGS_X, FX);
    cf b: init_const($FRAGS_Y, FY);

    init_f(F);

    init_lines(GL, GR, GU, GD);

    for i=0..FY-1 {
        for j=0..FX-1 {
            init_submatrix(FX, FY, i, j, M[0][i][j]);
        }
    }

    for t=1..$NT {
        for i=0..FY-1 {
            for j=0..FX-1 {
                if j!=0 {
                    get_line(0, M[t-1][i][j-1], LL[t-1][i][j]);
                }
                if j!=FX-1 {
                    get_line(2, M[t-1][i][j+1], LR[t-1][i][j]);
                }
                if i!=0 {
                    get_line(3, M[t-1][i-1][j], LU[t-1][i][j]);
                }
                if i!=FY-1 {
                    get_line(1, M[t-1][i+1][j], LD[t-1][i][j]);
                }
                if j==0 {
                    get_gline(2, M[t-1][i][j], GL, LL[t-1][i][j]);
                }
                if j==FX-1 {
                    get_gline(0, M[t-1][i][j], GR, LR[t-1][i][j]);
                }
                if i==0 {
                    get_gline(1, M[t-1][i][j], GU, LU[t-1][i][j]);
                }
                if i==FY-1 {
                    get_gline(3, M[t-1][i][j], GD, LD[t-1][i][j]);
                }

                calc(M[t-1][i][j], LL[t-1][i][j], LR[t-1][i][j], LU[t-1][i][j],
                        (M[t-1][i][j], LL[t-1][i][j], LR[t-1][i][j], LU[t-1][i]
            }
        }
    }

    init_result(GL, GR, GU, GD, R[0]);

    for i=0..FY-1 {
        for j=0..FX-1 {
            update_result(M[$NT][i][j], R[i*FX + j], R[i*FX + j + 1]) @ {
                delete R[i*FX + j];
            };
        }
    }

    save_to_bmp(R[FY*FX]);
```

Исходный код тестовой программы

Тестирование

Раздел “эффективность использования ресурсов”

“Вычислительная нагрузка” подмножеств фрагментов вычислений

Тестирование. Выводы

- В результате работы профилировщика были получены характеристики исполнения в соответствии с проектом системы
- Проанализирована тестовая программа
- Установлено, что она была фрагментирована слишком мелко, из-за чего примерно две трети времени исполнения было потрачено на создание фрагментов вычислений

Заключение

- В результате работы было достигнуто следующее:
  - Проведен обзор и анализ существующих средств профилирования параллельных программ. Составлен набор характеристик исполнения фрагментированных программ, определен способ получения данных характеристик.
  - Описан проект интерактивной визуальной системы профилирования фрагментированных программ и реализован прототип данной системы, обеспечивающий часть запланированной функциональности.
  - Тестирование показало, что данный прототип позволяет проанализировать исполнение фрагментированной программы и определить причины недостаточной производительности.

Направления дальнейшей работы

- Реализация всей запланированной функциональности визуальной системы профилирования.
- Реализация гибкой настройки параметров модуля логирования и визуальной системы профилирования.
- Автоматический поиск шаблонов неэффективного поведения программы для определения возможных проблем с производительностью.

Спасибо за внимание!