Разработка и реализация алгоритмов динамического планирования нагрузки на гетерогенный мультипроцессор

Студент: Беляев Н. А. НГТУ ФПМИ 1 курс маг.
Научный руководитель: Перепелкин В. А.

Термины

- Гетерогенным мультипроцессором будем называть мультипроцессор, в составе которого присутствуют процессоры, построенные на разных архитектурах или имеющие различные объемы доступной памяти

Описание проблемы

- На сегодняшний день для достижения высокой производительности при решении задач вычислительной математики с использованием суперкомпьютеров активно применяются гетерогенные мультипроцессоры и мультикомпьютеры
- Правильное распределение вычислительной нагрузки между процессорами мультипроцессора и узлами мультикомпьютера является критичным для достижения высокой производительности

Цель работы

- Обеспечить возможность эффективной работы с вычислителями, построенными на архитектурах Nvidia CUDA, Xeon PHI в системе LuNA

Обзор существующих средств параллельного программирования

OpenCL

- OpenCL – это открытый фреймворк для написания параллельных программ под процессоры, построенные на различных архитектурах (CUDA, MIC, FPGA, …)

OpenCL

- Пример кода OpenCL C

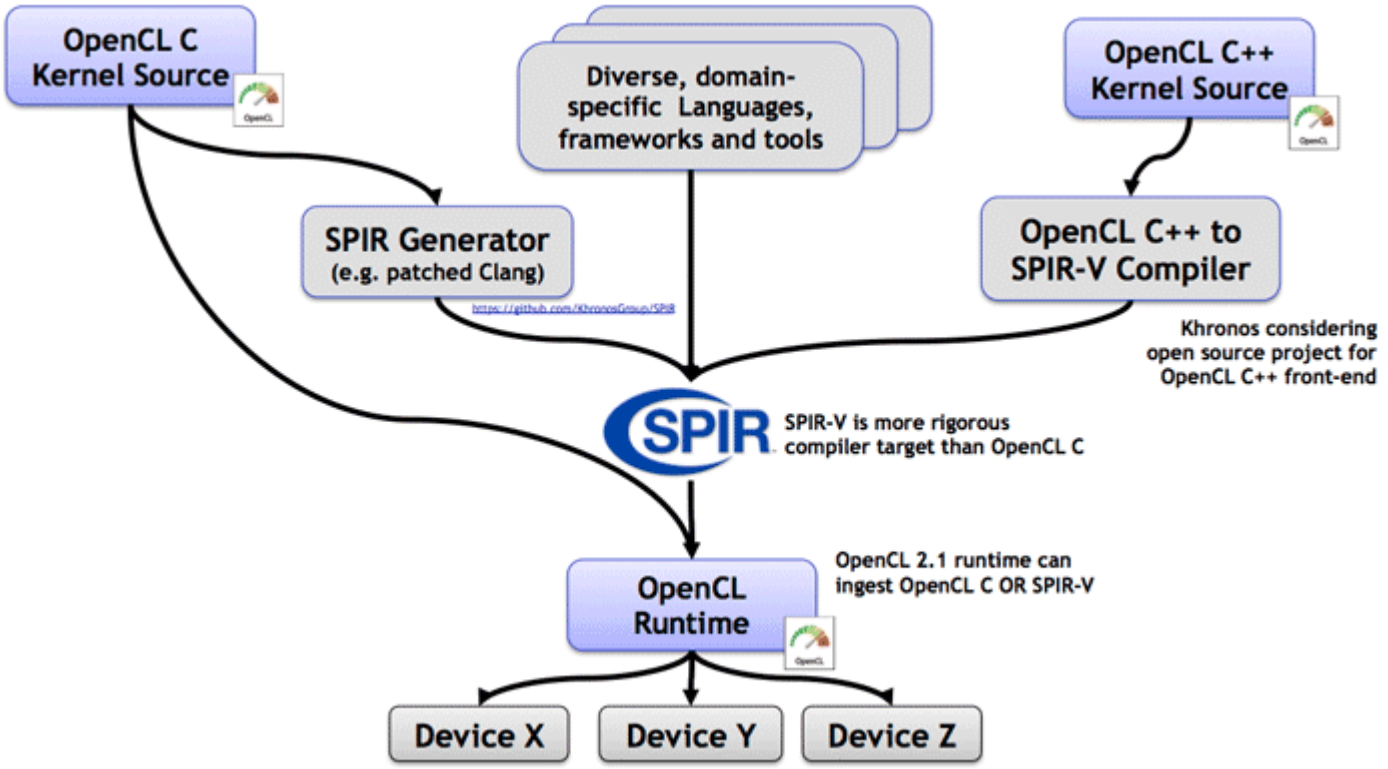
```
__kernel void mmmKernel(__global float4 *matrixA,
                        __global float4 *matrixB,
                        __global float4* matrixC,
            uint widthA, uint widthB)
{
    int2 pos = (int2)(get_global_id(0), get_global_id(1));


    float4 sum0 = (float4)(0);
    float4 sum1 = (float4)(0);
    float4 sum2 = (float4)(0);
    float4 sum3 = (float4)(0);

    /* Vectorization of input Matrices reduces their width by a factor of 4 */
    widthB /= 4;

    for(int i = 0; i < widthA; i=i+4)
    {
```

Open CL


OpenCL C
Kernel Source
Diverse, domain-
specific Languages,
frameworks and tools
OpenCL C++
Kernel Source
SPIR Generator
(e.g. patched Clang)
OpenCL C++ to
SPIR-V Compiler
Khronos considering
open source project for
OpenCL C++ front-end
SPIR
SPIR-V is more rigorous
compiler target than OpenCL C
OpenCL
Runtime
OpenCL 2.1 runtime can
ingest OpenCL C OR SPIR-V
Device X
Device Y
Device Z

OpenACC

- OpenACC – стандарт для параллельного программирования, описывающий набор директив компилятора (C, C++, Fortran), предназначенный для создание параллельных программ для гетерогенных мультипроцессоров

OpenACC

- Пример кода с использованием OpenACC

```
while ( error > tol && iter < iter_max ) {
  error = 0.0;
  #pragma acc kernels
  {
    for( int j = 1; j < n-1; j++) {
      for( int i = 1; i < m-1; i++ ) {
        Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1]
                            + A[j-1][i] + A[j+1][i]);
        error = fmax( error, fabs(A[j][i] - Anew[j][i]));
      }
    }

    for( int j = 1; j < n-1; j++) {
      for( int i = 1; i < m-1; i++ ) {
        A[j][i] = Anew[j][i];
      }
    }
  }

  if(iter % 100 == 0) printf("%5d, %0.6f\n", iter, error);
  iter++;
}
```

Задачи

- Обеспечить в системе LuNA возможность запуска атомарных ФВ на вычислителях, построенных на различных архитектурах и имеющих различный объем памяти в составе узла мультикомпьютера.
- Разработать и реализовать алгоритм динамического планирования нагрузки на вычислители в системе LuNA.

Задачи

Имеется гетерогенный мультипроцессор, состоящий из вычислителей, построенных на разных архитектурах (например, x86 CPU и Nvidia CUDA).

Каждый вычислитель обладает набором следующих характеристик:

- Количество ядер
- Объем памяти
- Быстродействие
- Необходимость передачи данных во внутреннюю память вычислителя (такая необходимость имеет место при работе с MIC и CUDA)
- Быстродействие канала

В контексте данной работы будем называть задачей атомарный ФВ.

Задачи

Каждый ФВ (задача) обладает следующим набором характеристик:

- Вычислительный вес
- Размер и количество входных ФД
- Размер и количество выходных ФД

В свою очередь каждый входной ФД также обладает следующими характеристиками:

- Размер
- Нахождение в памяти вычислителя

Необходимо обеспечить в системе LuNA возможность эффективного распределения ФВ и ФД на различные вычислители в составе мультипроцессора.

Постановка задачи (формальная)

- Дано множество задач $T = \{t_1, t_2, .., t_n\}$ где
- $t = (CF_i, C_e, |\text{in}(CF_i)|, |\text{out}(CF_i)|, ins(CF_i), outs(CF_i))$ где CF - ФВ, Ce – вычислительный вес ФВ, $|\text{in}(CF_i)|$ - количество входных ФД $|\text{out}(CF_i)|$ - количетсво выходных ФД для ФВ CF,
- ins(CF), outs(CF) – размеры входных и выходных ФД

- Дано также множетсво процессоров $P = \{p_1, p_2, .., p_m\}$, где
- $p = \{(m, cs, b) |\ m \in \mathbb{Z}^+, cs \in \mathbb{Z}^+, b \in \{0,1\}\}$ где m – размер памяти,

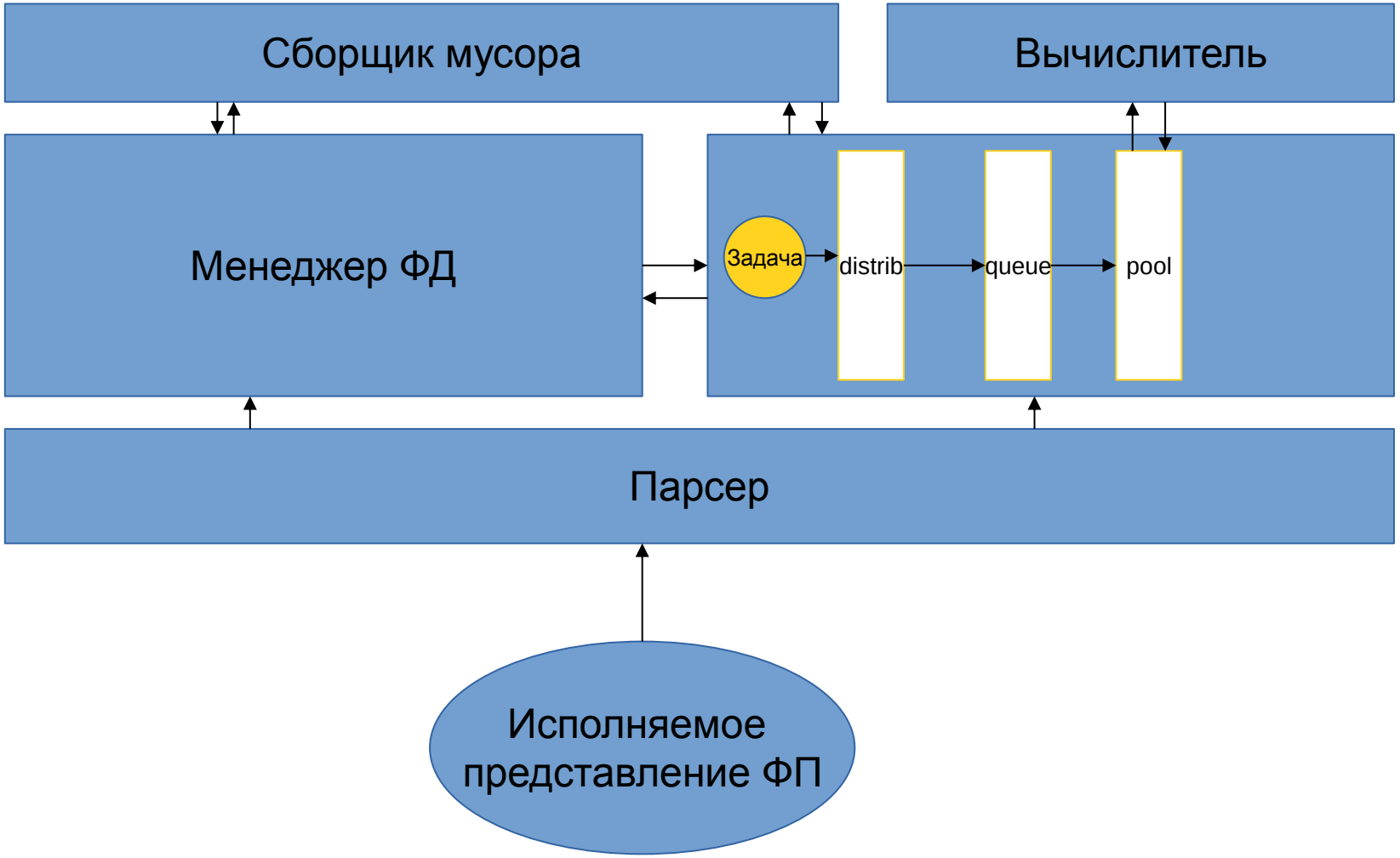
- доступной для процессора p, cs – скорость передачи данных канала между ОП CPU и памятью процессора p, b – признак доступности процессора p (b = 0 – процессор свободен, b = 1 – процессор занят), Обозначим PTi множетсво задач, назначенных на процессор pi

Постановка задачи (формальная)

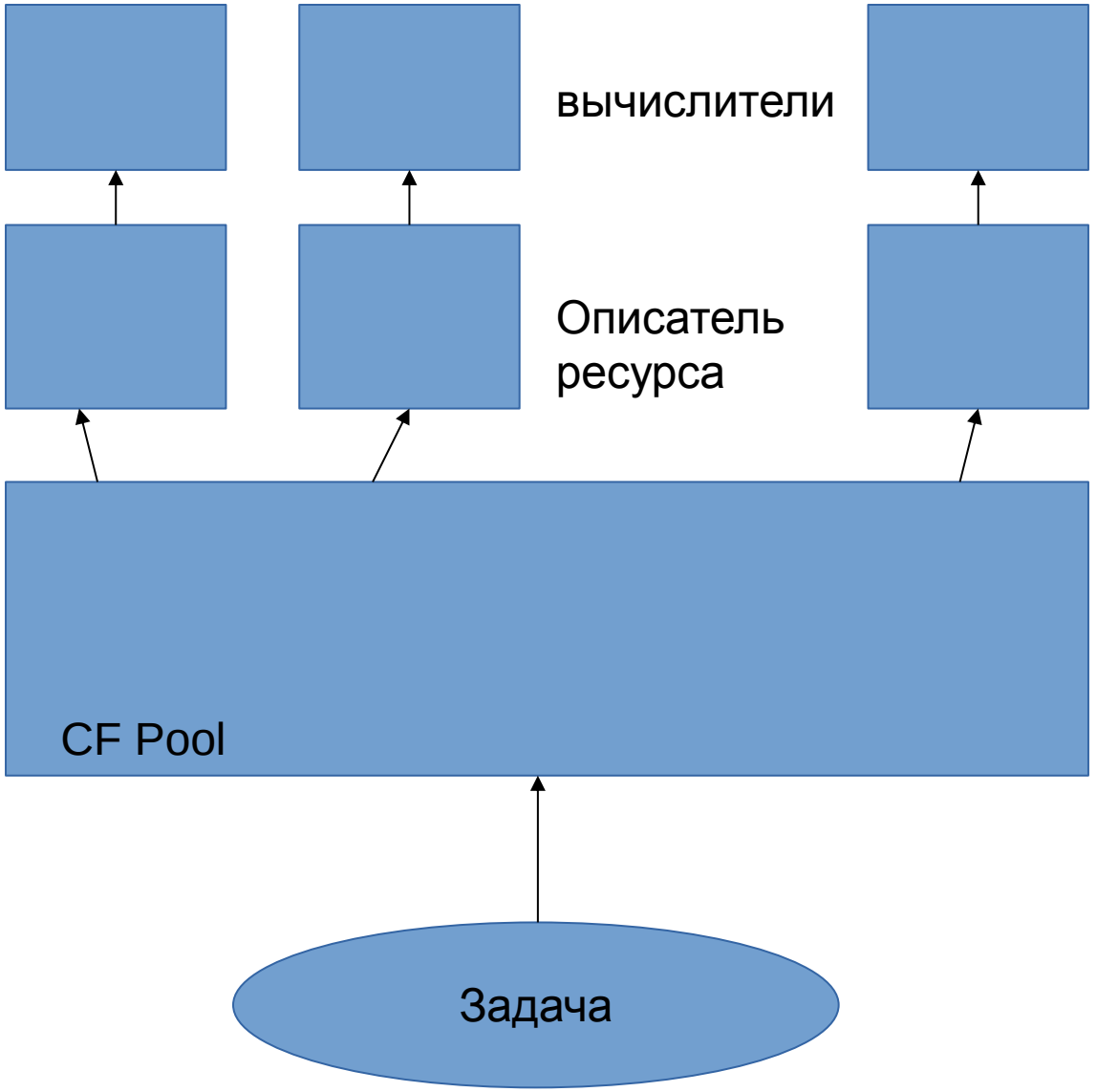
- Необходимо определить отображение $S: T \rightarrow P$ при назначении ФВ на процессоры мультипрцессора с учетом указанных свойств задачи и процессора
- Необходимо определить отношение порядка R на множестве PTi

Начальная реализация

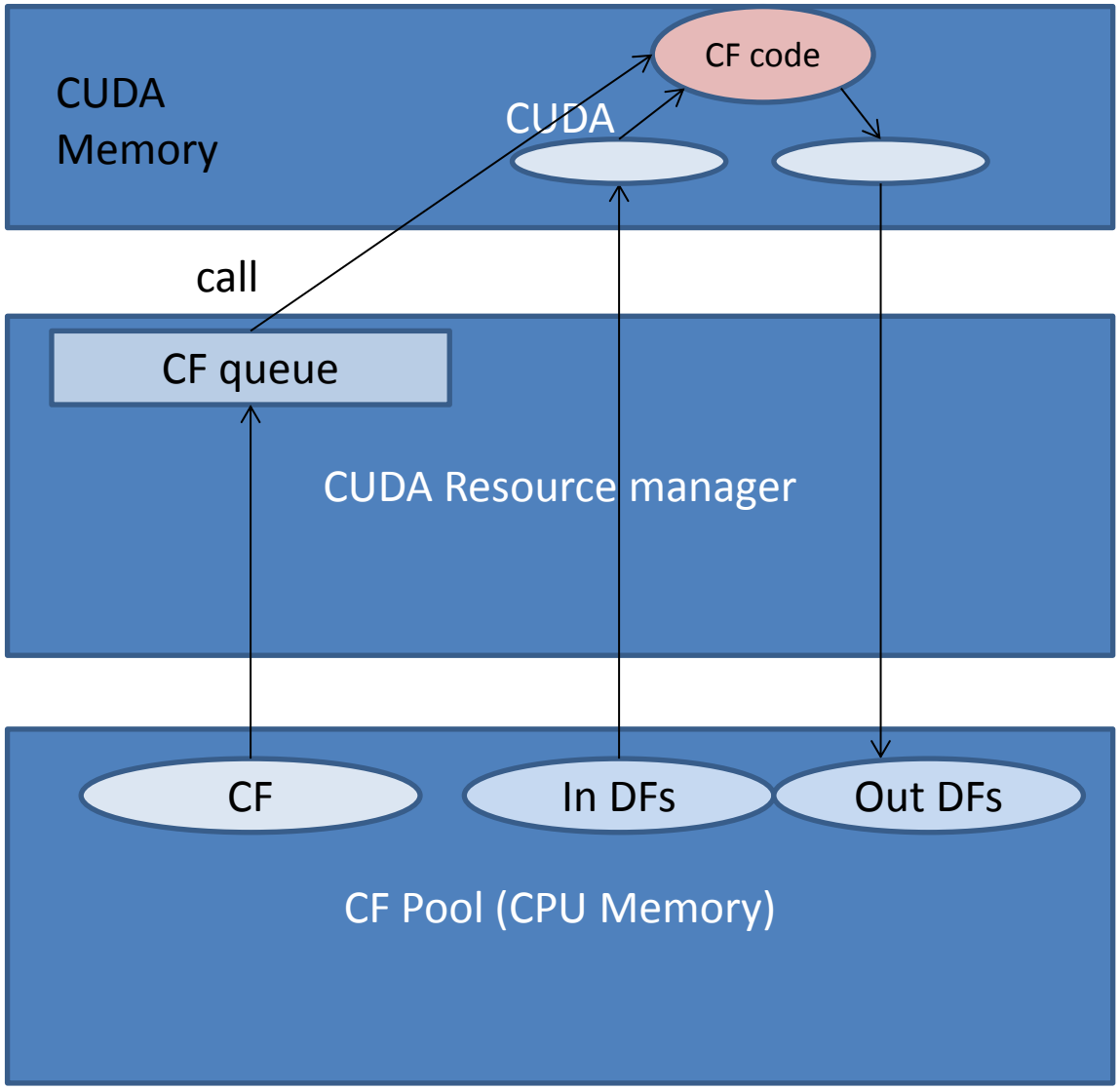
Архитектура исполниетльной системы LuNA



Пул задач с поддержкой гетерогенного мультипроцессора



Поддержка CUDA



Планы

- Протестировать производительность при наивном алгоритме назначении ФВ на вычислители и использовании CUDA
- Реализовать алгоритм динамического планирования исполнения ФВ на процессорах гетерогенного мультипроцессора