Проект библиотеки управления распределенными данными **Didal** для разработки параллельных программ для вычислительных систем с распределенной памятью

Щукин Г.А.
Supercomputer Software Department, ИВМиМГ СО РАН
Новосибирск, 2019

Желаемые свойства параллельного программирования

- Простота применения
- Переиспользуемость
- Переносимость
- Расширяемость
- Оптимизируемость

Задачи

- Предоставить пользователю средства для упрощения создания параллельных программ в распределенной памяти
- Обеспечить переносимость, переиспользуемость и расширяемость полученного кода
- Обеспечить возможность оптимизации эффективности исполнения полученного кода
- Обеспечить полную или частичную совместимость с другими популярными средствами и инструментами последовательного и параллельного программирования

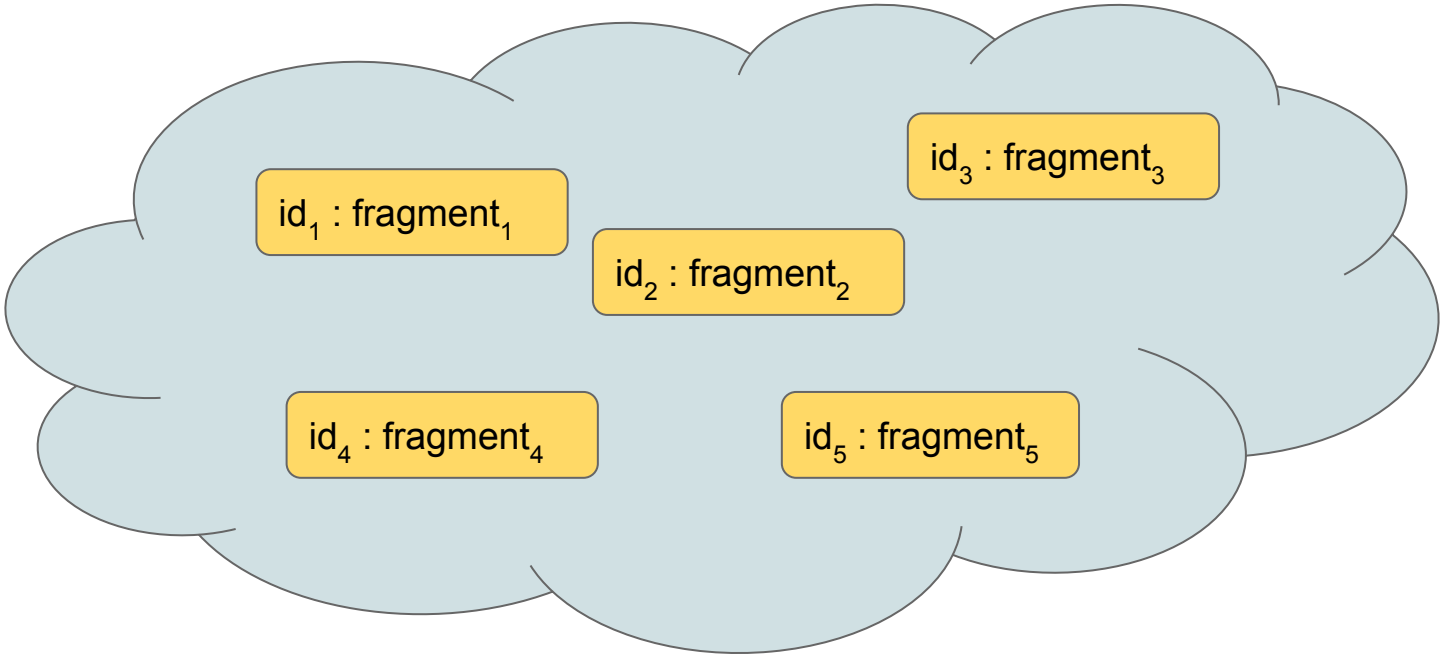
Related works

- LuNA
- Chapel
- Charm++
- DVM
- DASH
- ...

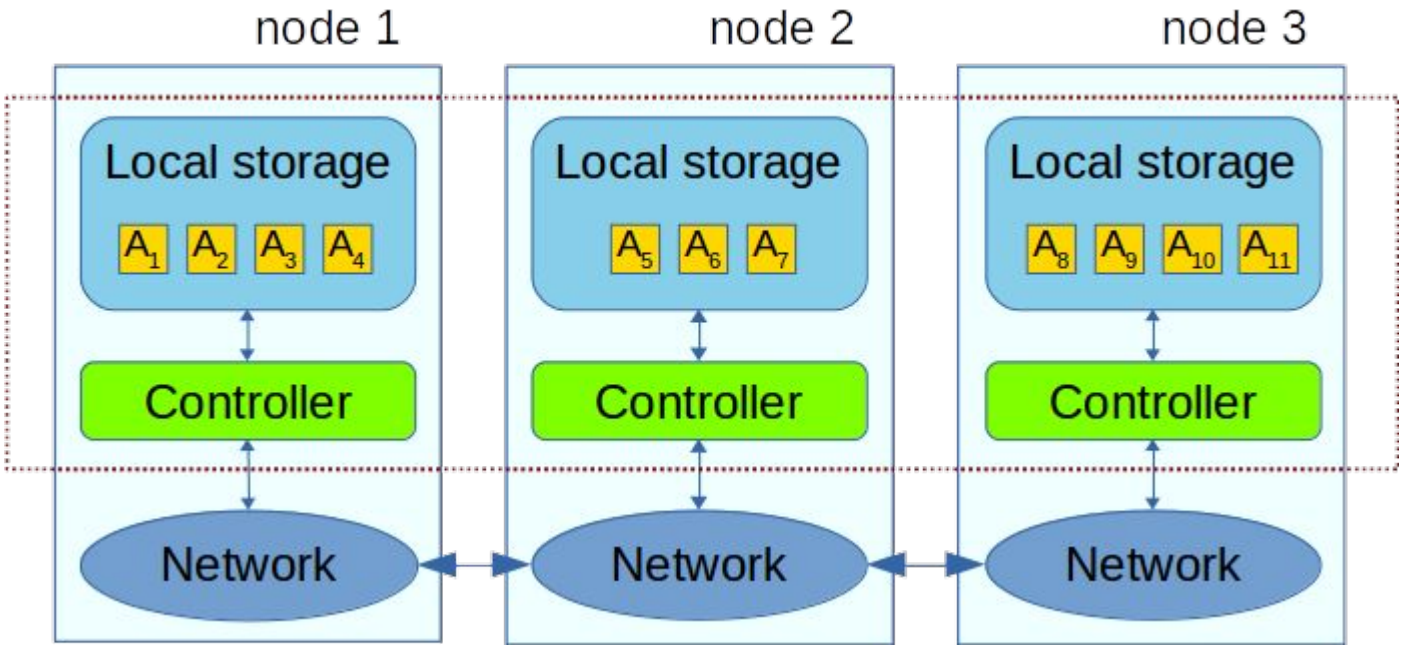
Didal: Distributed Data Library

- C++ библиотека
- Основные компоненты - распределенные данные и алгоритмы (операции)
- Реализация концепции фрагментированного программирования
- Для разработки параллельных программ в распределенной памяти
- Совместимость с MPI, OpenMP, CUDA

Распределенные данные (1)



Распределенные данные (2)



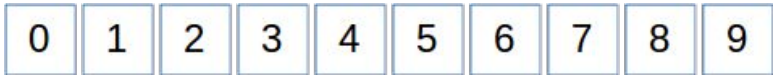
Distributed data object

Распределенный массив

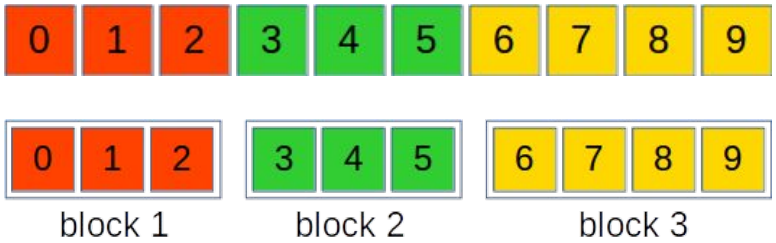
```
ddl::Environment env;

ddl::DistributedArray<double,      // тип элементов
    ddl::Array<double>,            // тип фрагментов
    ddl::IndexRange>               // тип формы
array (&env,
    100,                           // число элементов
    ddl::BlockDecomposition(5),    // стратегия фрагментации
    ddl::UniformDistribution(),    // стратегия распределения
    env.allNodes()                 // используемые узлы
);
```

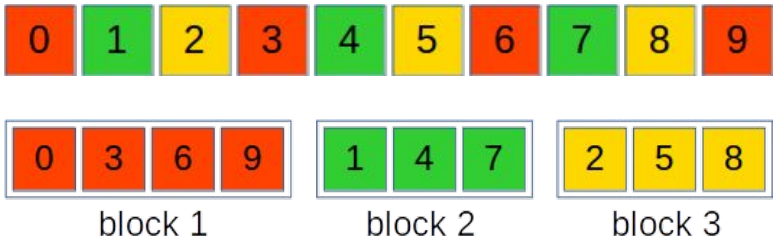
Декомпозиция данных



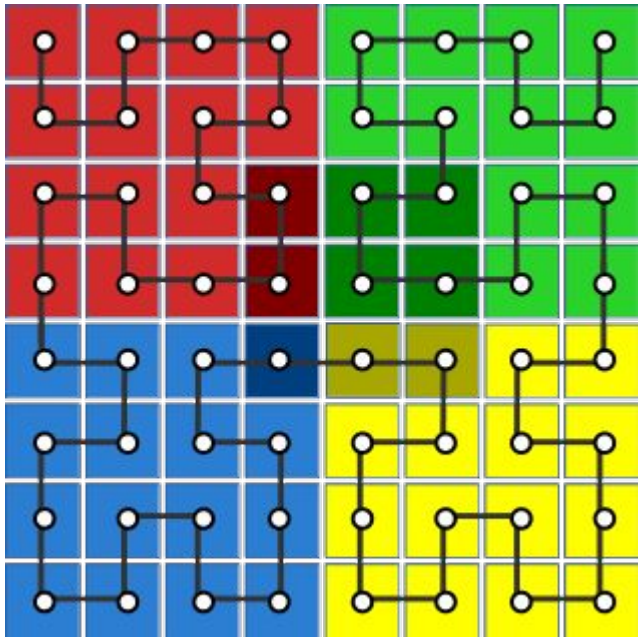
ddl::BlockDecomposition(size=3)



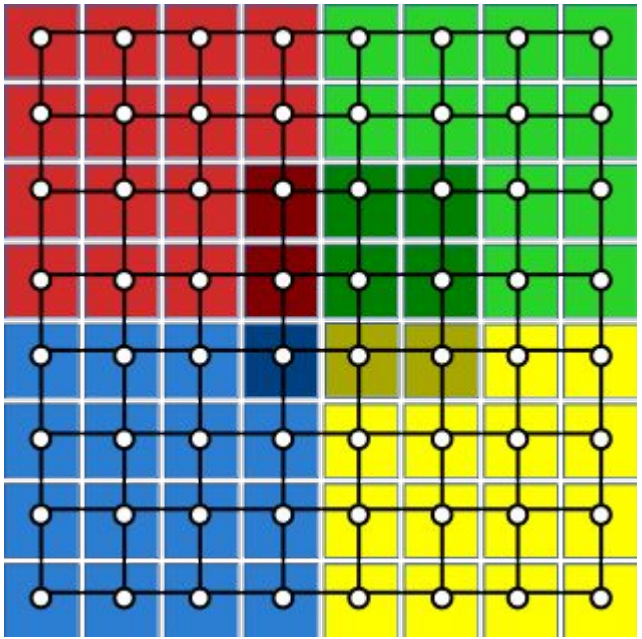
ddl::CyclicDecomposition(step=3)



Распределение данных и балансировка нагрузки



Rope

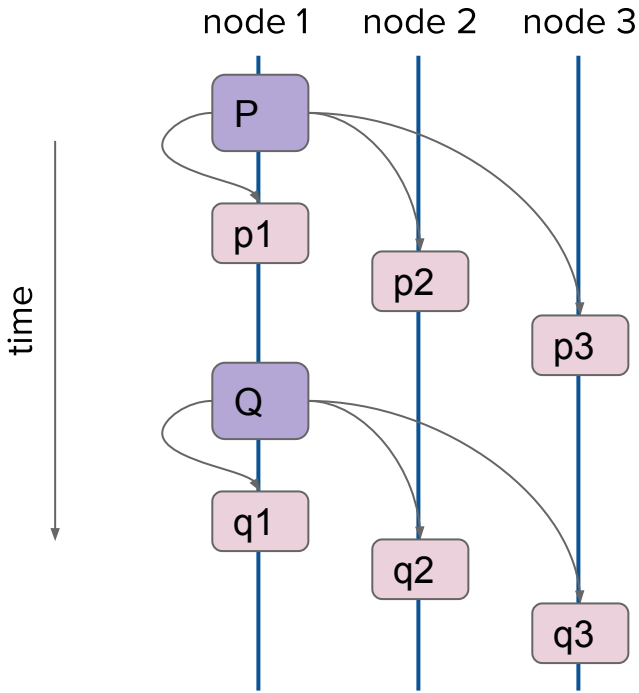


Patch

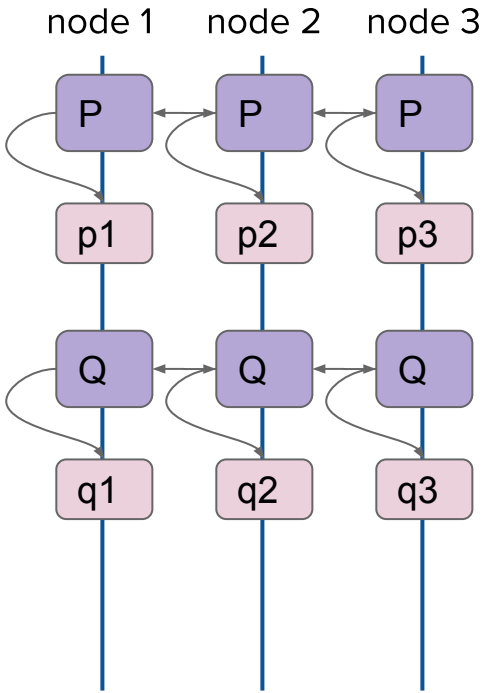
Распределенные операции (1)

$B = F(A) \dashrightarrow B_i = f(A_i)$

Модель исполнения

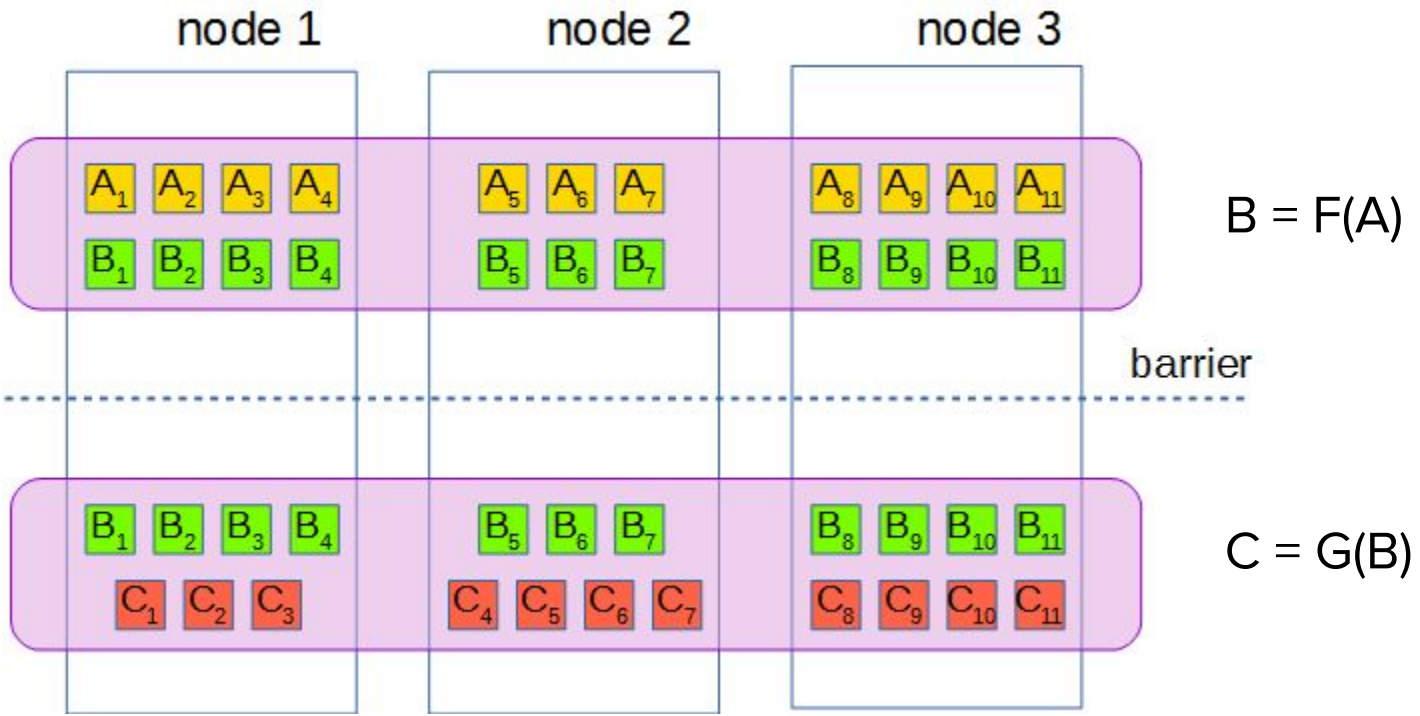


Multiple Program Multiple Data

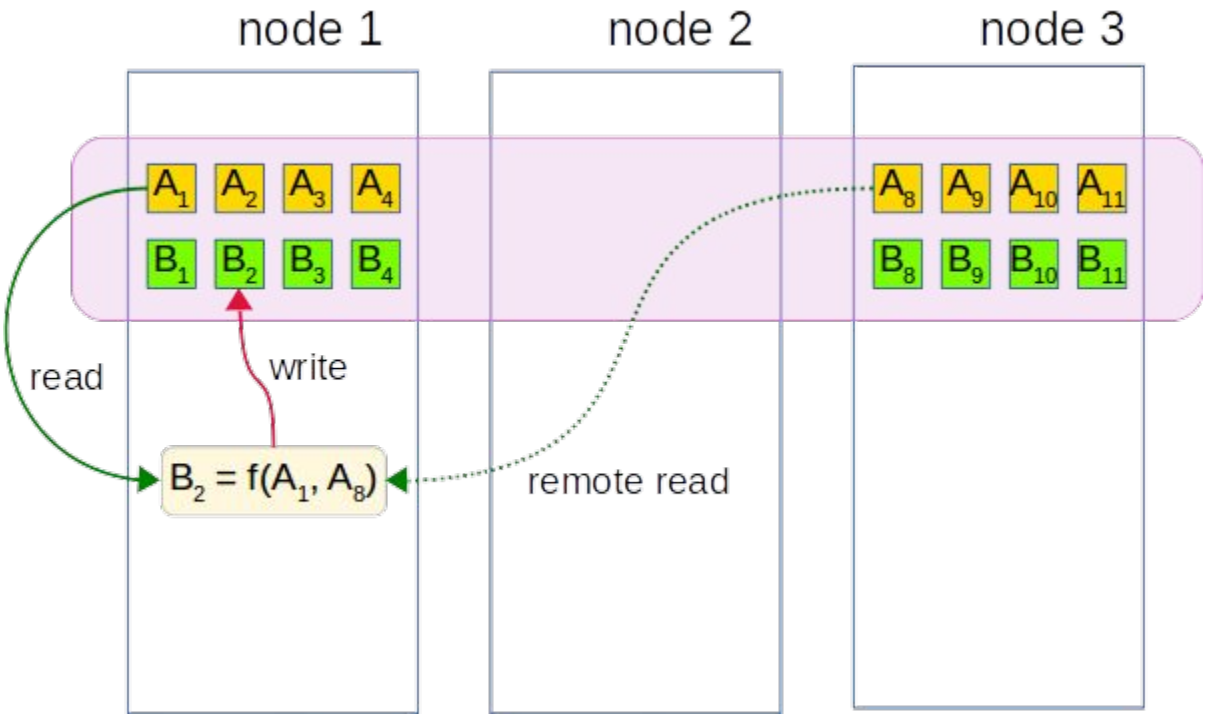


Single Program Multiple Data

Распределенные операции (2)



Доступ к распределенным данным



Пример 1: сумма элементов массива

```
#include "didal.h"

int main(int argc, char **argv) {
      ddl::Environment env(&argc, &argv);

      ddl::DistributedArray<double> a(&env, 100,
            ddl::BlockDecomposition(5), ddl::UniformDistribution(),
            env.allNodes());

      ddl::forEach (a, [](ddl::Array<double> &block) {
            for (size_t i = 0; i < block.size(); i++) {
                  block[i] = block.toGlobalIndex(i) * 2;
            }
      });

      const auto sum = ddl::reduce(a, [](double el, double curr) {return el + curr;}, 0.0);
      std::cout << "Sum of elems is " << sum << std::endl;
      return 0;
}
```

Пример 2: сумма элементов массива (Complex)

```
class Complex {
public:
	Complex(double r, double i) : real(r), imag(i) {}
	double real {0.0}, imag {0.0};
};

std::ostream& operator<<(std::ostream &out, const Complex &c) {
	out << "(" << c.real << ", " << c.imag << ")";
	return out;
}

ddl::Writable& operator<<(ddl::Writable &w, const Complex &c) {
	return w << c.real << c.imag;
}

ddl::Readable& operator>>(ddl::Readable &r, Complex &c) {
	return r >> c.real >> c.imag;
}
```

Пример **2:** сумма элементов массива **(Complex)**

```
#include "didal.h"

int main(int argc, char **argv) {
	ddl::Environment env(&argc, &argv);

	ddl::DistributedArray<Complex> a(&env, 100,
		ddl::BlockDecomposition(5), ddl::UniformDistribution(),
		env.allNodes());

	ddl::forEach (a, [](ddl::Array<Complex> &block) {
		for (size_t i = 0; i < block.size(); i++) {
			block[i] = Complex(block.toGlobalIndex(i) * 2, double(block.toGlobalIndex(i)/2.0));
		}
	});

	const auto sum = ddl::reduce(a, [](const Complex &el, const Complex &curr) {
		return Complex(el.real + curr.real, el.imag + curr.imag); }, Complex(0.0, 0.0));
	std::cout << "Sum of elems is " << sum << std::endl;
	return 0;
}
```

Future...

- ddl::DistributedMatrix
  - различные форматы матриц
- ddl::DistributedMesh
  - различные типы сеток
- ddl::DistributedGraph
  - различные типы графов

Thank you!