ВНИИА
РОСАТОМ

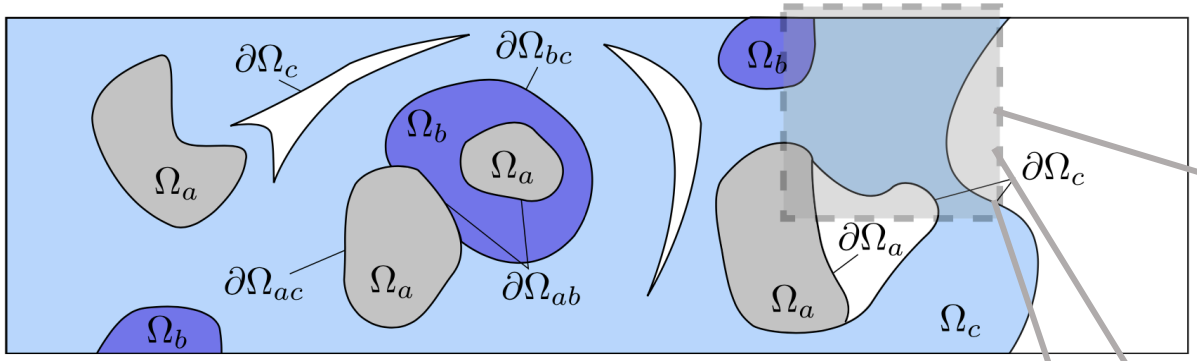
Сопряжение численных методов в рамках единого комплекса программ

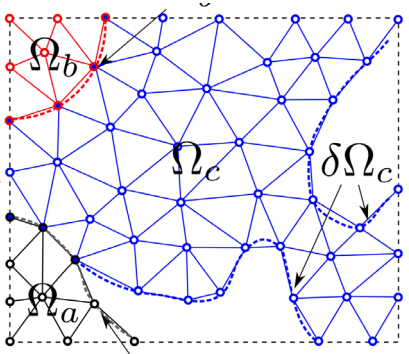
XVI Международная конференция
«Забабахинские научные чтения»
29 мая – 2 июня 2023 г.

С.А. Дьячков, С.Ю. Григорьев, Р. В. Муратов

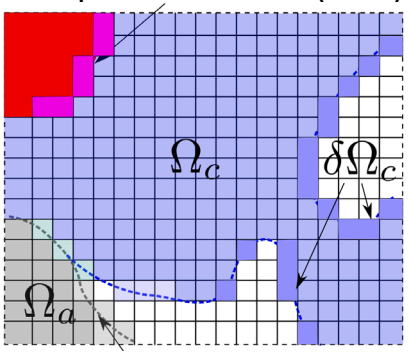
ФГУП «ВНИИА» им. Н.Л. Духова

Мотивация

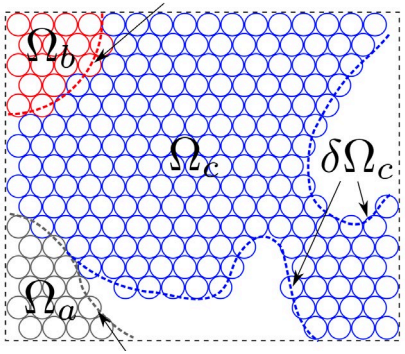




Лагранжева сетка (FEM)



Эйлерова сетка (FVM)



Бессеточный (SPH)

- Дискретизация расчетной области может быть выполнена для разных численных схем
- Обычно программа реализуется для одного численного метода, который имеет свои преимущества и недостатки
- При правильной организации структур данных можно обеспечить возможность одновременного счета различными численными методами
- Использование преимуществ различных численных методов позволяет расширить круг решаемых задач

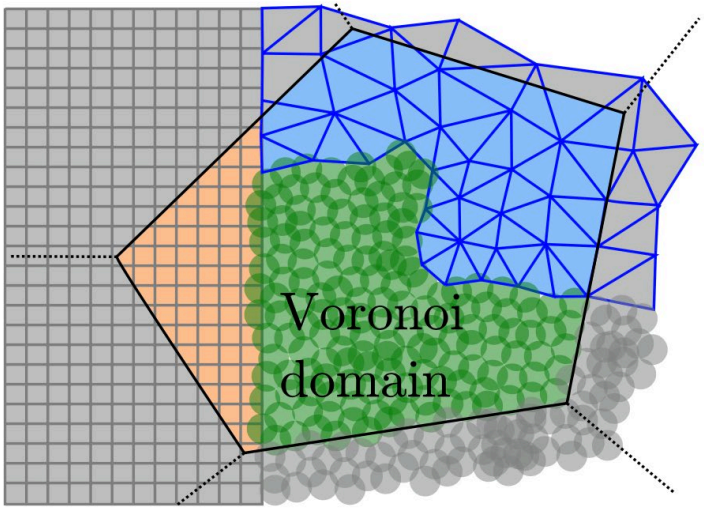
ВНИИА
РОСАТОМ

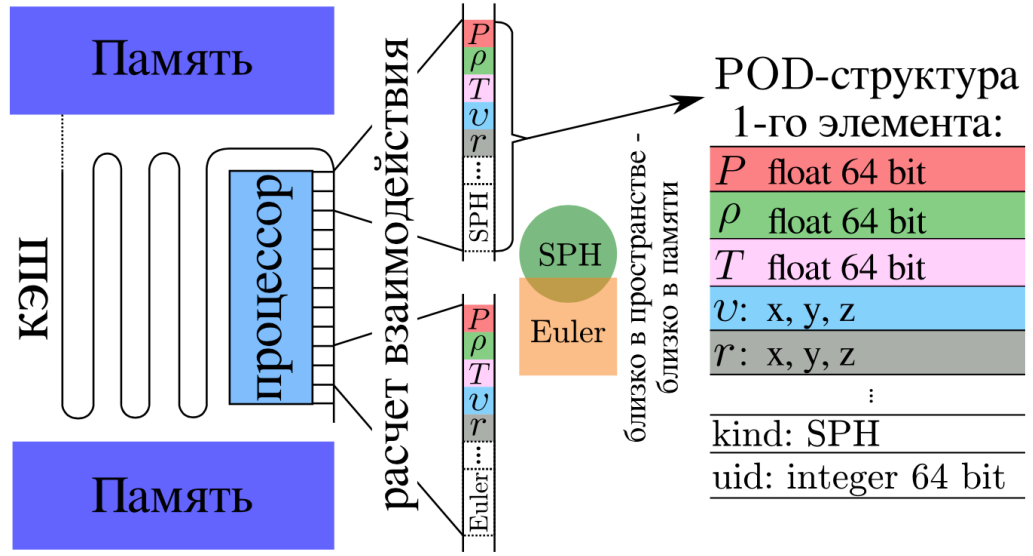
Общность описания элементов дискретизации

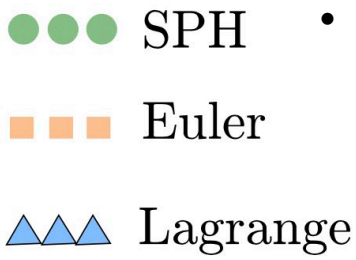
Неподвижная сетка	Подвижная сетка	Частица SPH	Атом MD
координата центра	координата центра	координата центра	координата центра
радиус поиска соседа	радиус поиска соседа	радиус поиска соседа	радиус поиска соседа
размер элемента	размер элемента	размер элемента	размер элемента
тип элемента	тип элемента	тип элемента	тип элемента
тип материала	тип материала	тип материала	тип материала
скорость вещества	скорость вещества	скорость вещества	скорость атома
плотность вещества	плотность вещества	плотность вещества	сила (ускорение)
внутренняя энергия	внутренняя энергия	внутренняя энергия	
производная энергии	производная энергии	производная энергии	
скорость звука	скорость звука	скорость звука	
грани ячеек	масса элемента	масса элемента	
потоки через грани	скорость деформации	скорость деформации	
	вершины ячеек		

Требования к архитектуре программного комплекса PyPHIA

- Единое представление (формат) данных вектора состояния элементов, оптимизированное для высокопроизводительных вычислений
- Единые алгоритмы определения связности (соседства) дискретных элементов







- Единые алгоритмы параллелизации вычислений для разных типов дискретизации

Единый формат

ВНИИА
РОСАТОМ

```
struct ElementData {
    vector r;
    vector v;
    double density;
    double pressure;
    double energy;
    ...
    dtype  field;
    ...
};

ElementData elements[100];
```

Набор массивов

```
vector r[100];
vector v[100];
double density[100];
double pressure[100];
double energy[100];
```

- Вектор состояния одного элемента среды представляется в виде структуры данных
- Поля структуры полностью определены до компиляции и запуска программы
- При добавлении новых моделей в программу список полей структуры растет, но не все из них используются, из-за чего возникает рост потребления памяти и несовместимость версий программы
- Хранение вектора состояния не в одной структуре, а в виде массивов приводит к неоднородному доступу к памяти

Нужна высокопроизводительная структура данных в C++, аналогичная NumPy/HDF5 массиву, где поля можно задавать в ходе работы программы

Единый формат

ВНИИА
РОСАТОМ

- Использование POD-структур для полей Storage обеспечивает совместимость с NumPy-массивом (Python), хранением в формате HDF5, передачей через MPI
- Конвертация данных между Storage, NumPy, HDF5 и MPI осуществляется путем изменения только блока метаданных (без изменения самих данных) и происходит моментально

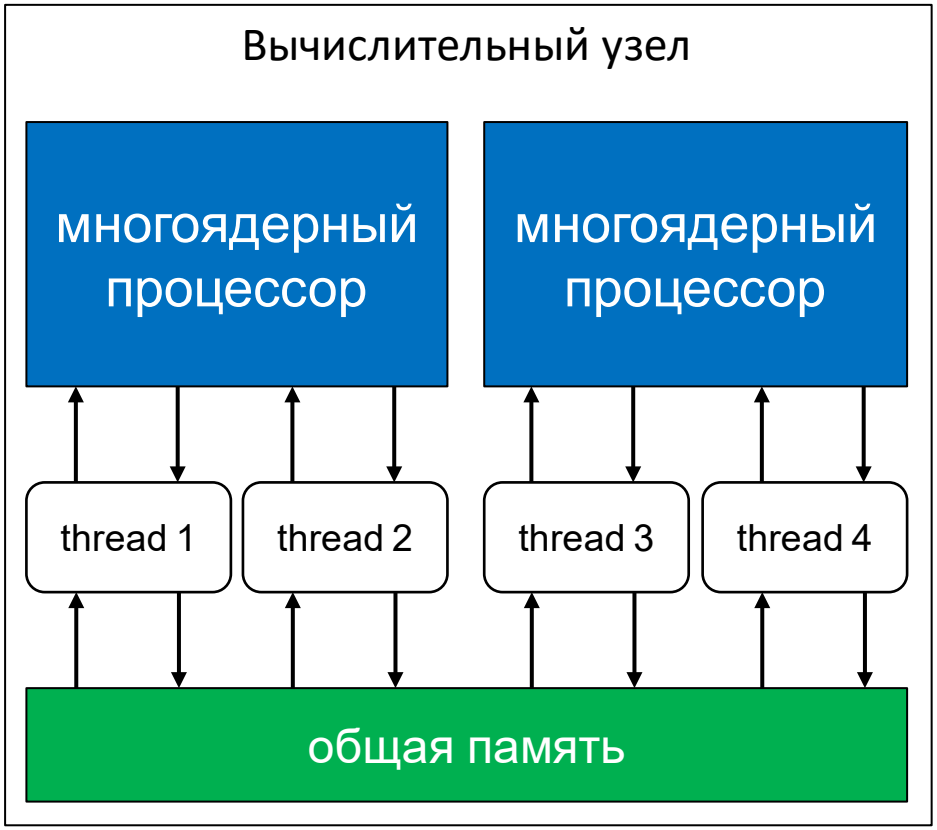
Python - NumPy

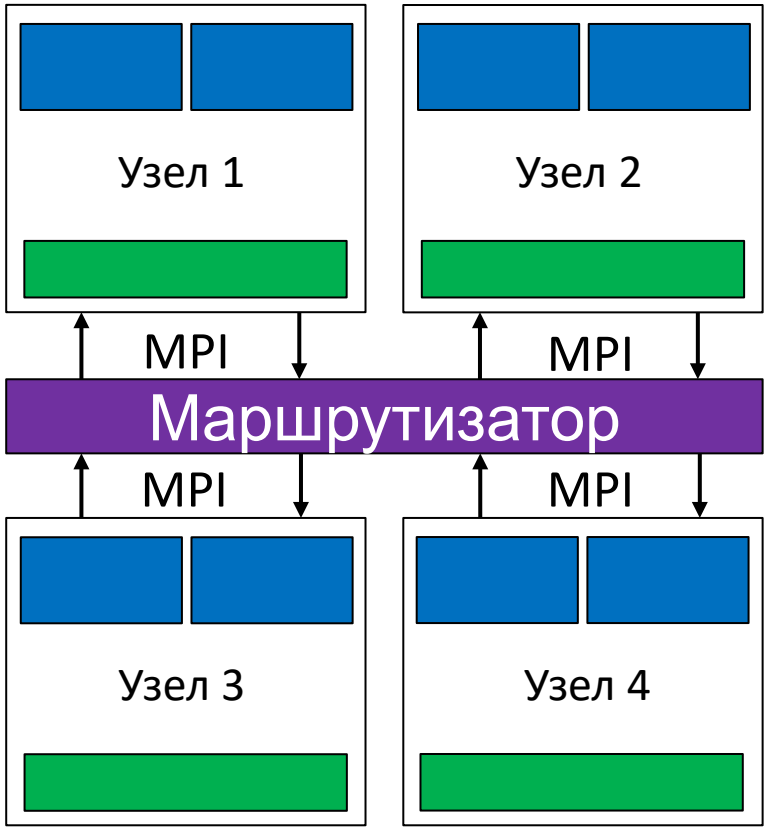
```
from   pyphia.data.types import *
from   pyphia.data import Storage

storage = Storage(
    np.dtype([
        coords,
        velocity,
        size
    ]), 10
)
storage.data["coords"]["x"]    = 5.0
storage.data["velocity"]["x"] = 5.0
storage.data["size"]["value"] = 5.0
```

HDF5

```
DATASET "storage" {
  DATATYPE  H5T_COMPOUND {
     H5T_COMPOUND {
        H5T_IEEE_F64LE "x";
        H5T_IEEE_F64LE "y";
        H5T_IEEE_F64LE "z";
     } "coords";
     H5T_COMPOUND {
        H5T_IEEE_F64LE "x";
        H5T_IEEE_F64LE "y";
        H5T_IEEE_F64LE "z";
     } "velocity";
     H5T_COMPOUND {
        H5T_IEEE_F64LE "value";
     } "size";
  }
  DATASPACE  SIMPLE { ( 10 ) / ( 10 ) }
}
```

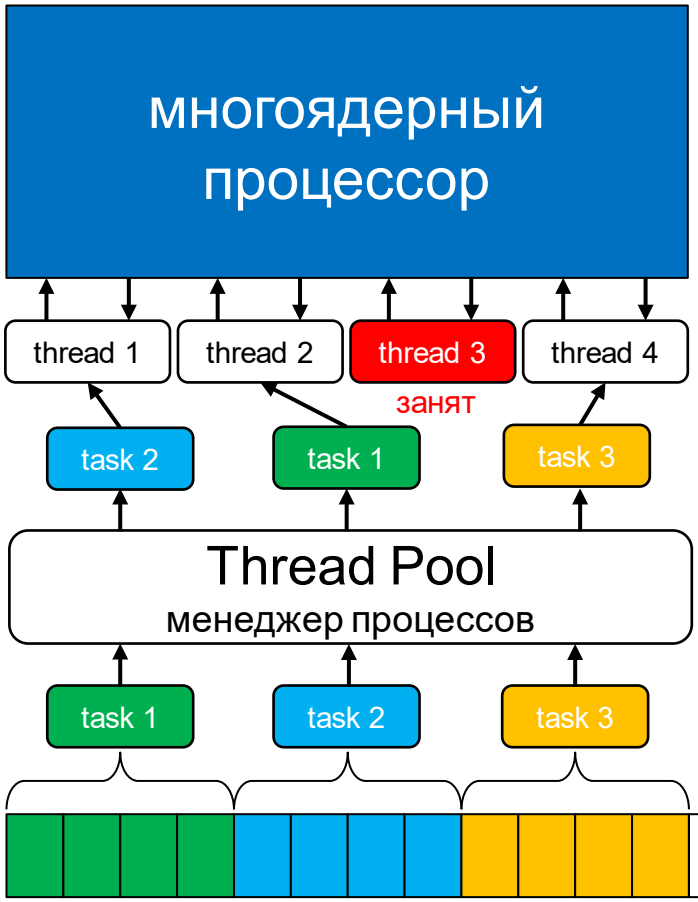
Общая и распределенная память



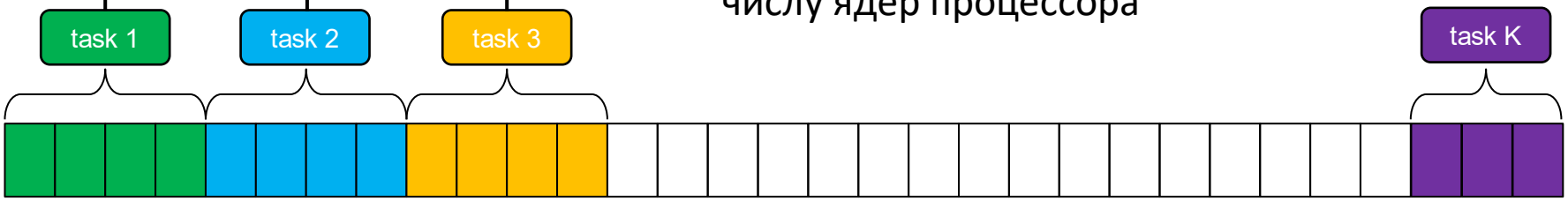


- Использование параллелизации с общей памятью (Threads, openMP) внутри одного узла позволяет избежать лишних сетевых (медленных) коммуникаций
- Сетевые коммуникации необходимы при использовании параллелизации с распределенной памятью (между узлами)

Параллелизация внутри узла

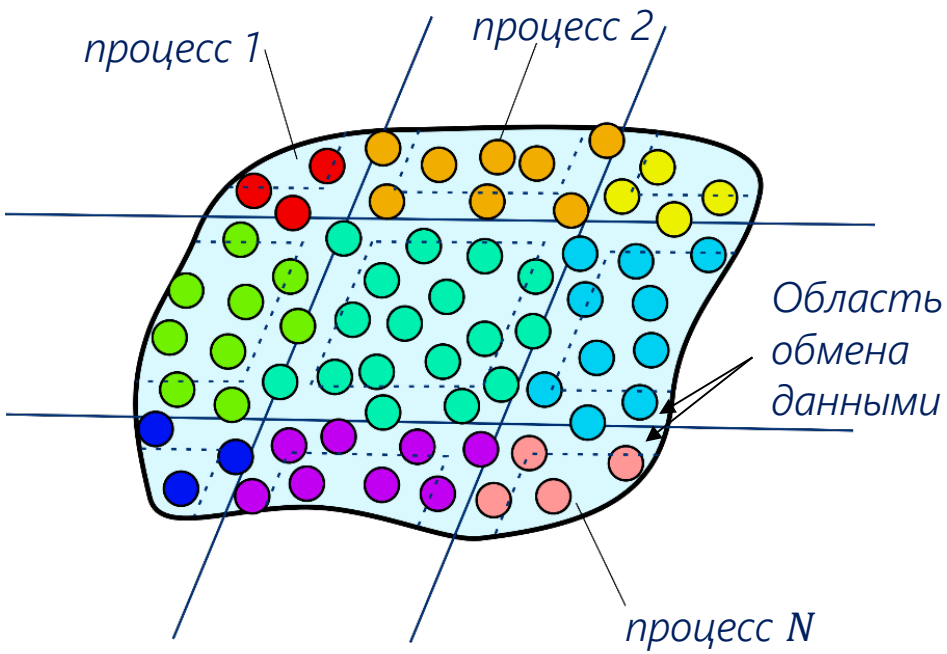


- Запускается менеджер процессов, который назначает задачи заданному набору подконтрольных процессов (threads)
- Задача (task) может представлять собой, например, обработку m элементов массива
- Создать список из K задач можно, например, разделив массив из N элементов на K частей
- Число задач может быть больше числа управляемых менеджером процессов, в этом случае задачи будут ожидать в очереди
- Число процессов задается пропорционально числу ядер процессора



Параллельная обработка массива

Алгоритм параллелизации с распределенной памятью



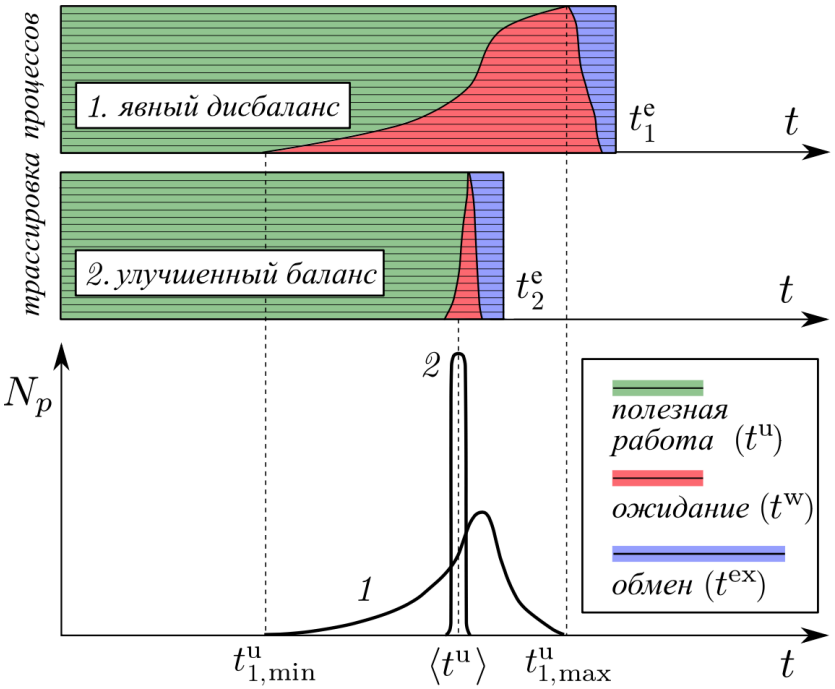
Концепция пространственной декомпозиции между узлами

$$t^e = t^u + t^w + t^{ex}$$

Шаг моделирования t^e состоит из

- Расчета взаимодействия
- Обмена данным
- Ожидания синхронизации шага

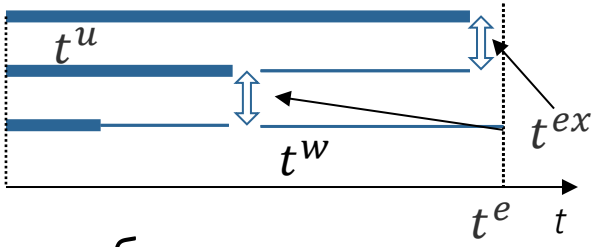
- Вычислительный кластер состоит из узлов со своей локальной памятью
- Обмен данными возможен только через «медленную» сеть и должен быть минимален
- Каждый шаг моделирования требует синхронизации между узлами



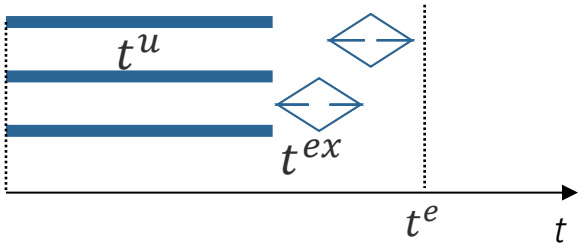
Как достичь баланса вычислительной нагрузки?

Алгоритм параллелизации с распределенной памятью

- Дисбаланс нагрузки: процесс 1 тратит много времени на расчет

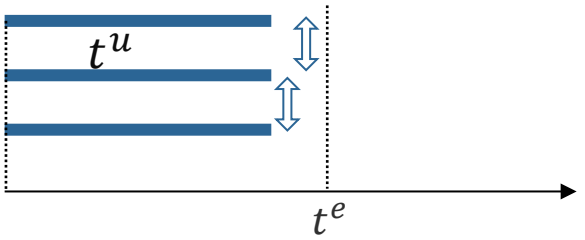


- Есть баланс нагрузки, но много времени занимает обмен



- Баланс нагрузки и минимум обмена данными

t^e в этом случае минимально



Перегруженные узлы замедляют общий ход расчета и должны перераспределять данные между другими

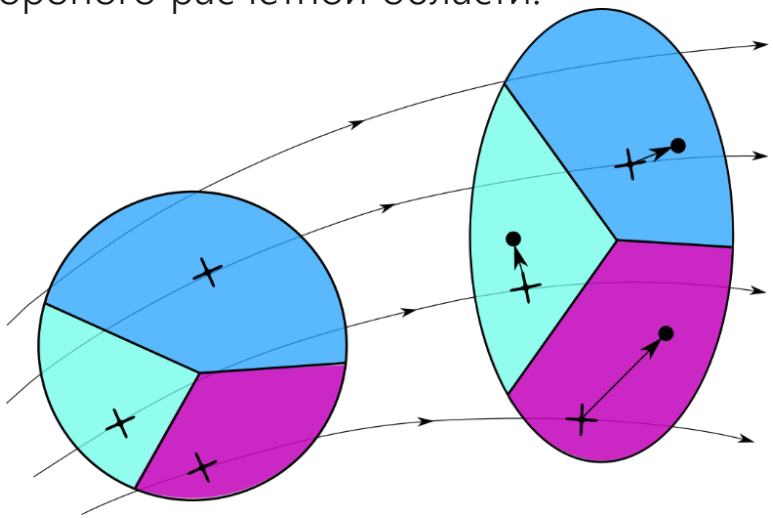
Алгоритм параллелизации с распределенной памятью

Определение (Диаграмма Вороного, VD):

1. Пусть $\{G_k\}_{k=1}^{N_{\hat{V}}} \in \bar{\Omega}$ множество точек (центров) подобластей расчетной области $\bar{\Omega}$.
2. Подобласть Вороного $\hat{V}_k$ множество точек, ближайших к G_k:

$$\hat{V}_k = \{\vec{x} \in \Omega : |\vec{x} - \vec{g}_k| < |\vec{x} - \vec{g}_l|, \; l = 1, \dots, N_{\hat{V}}, l \neq k\},$$

   где $\vec{g}_k$ — вектор из начала координат к G_k.
3. Множество $\{\hat{V}_k\}_{k=1}^{N_{\hat{V}}}$ является диаграммой Вороного расчетной области.



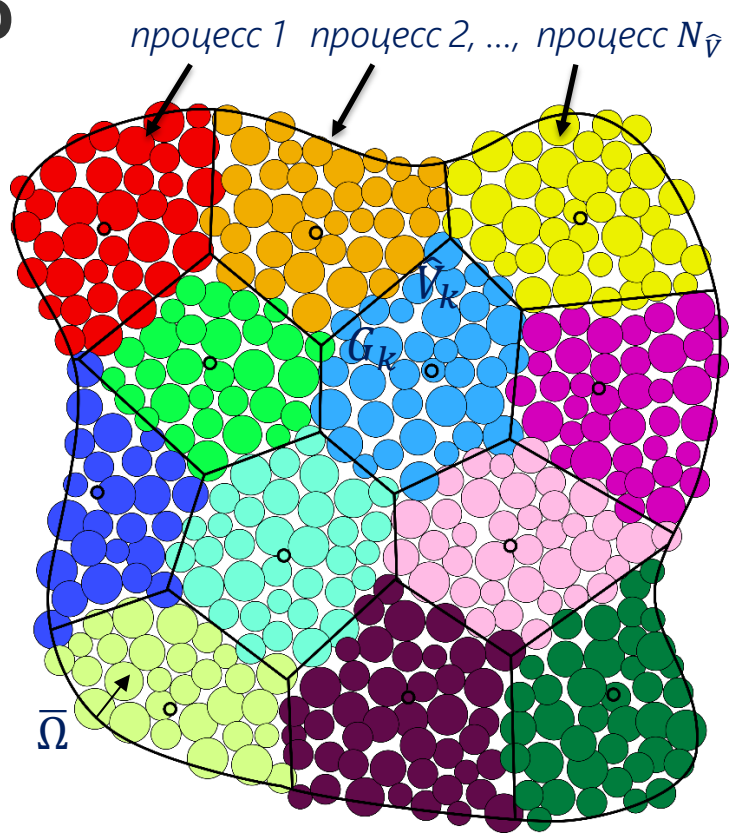
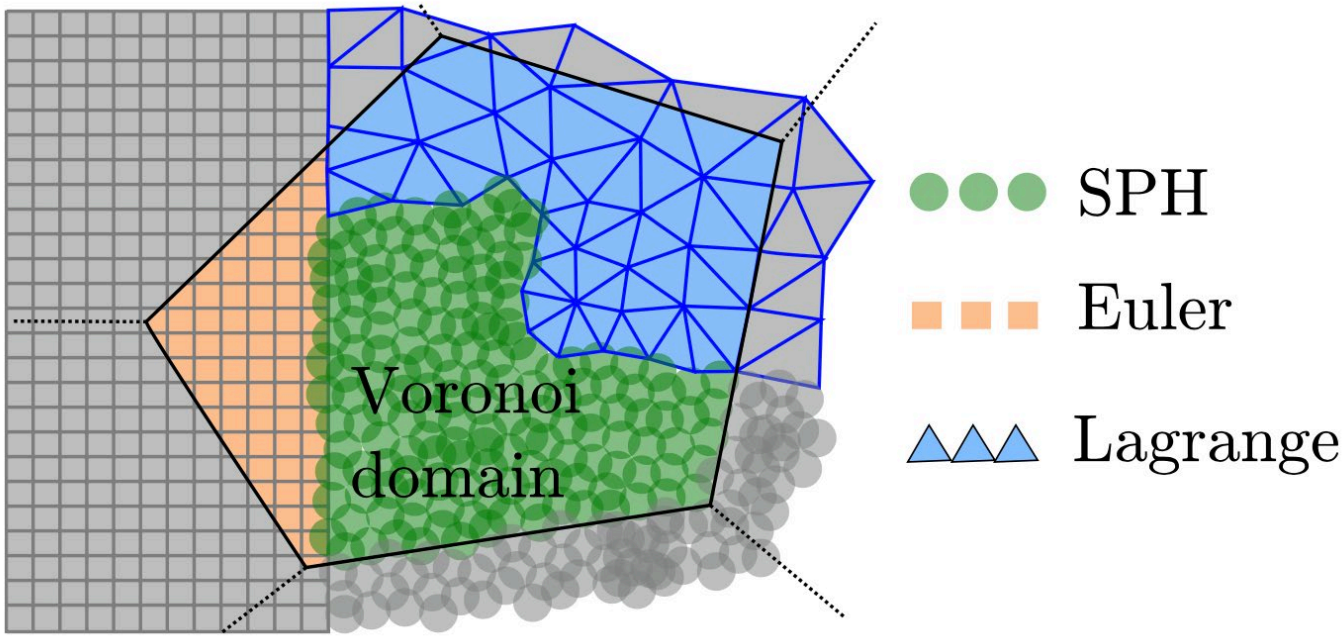


Диаграмма Вороного расчетной области $\bar{\Omega}$

- $\hat{V}_k$ соответствует процессу k
- G_k движется вместе с частицами
- G_k движется к загруженным процессам

M. S. Egorova et al. Computers Physics Communications **234,** 112-125 (2018)

Алгоритм параллелизации с распределенной памятью



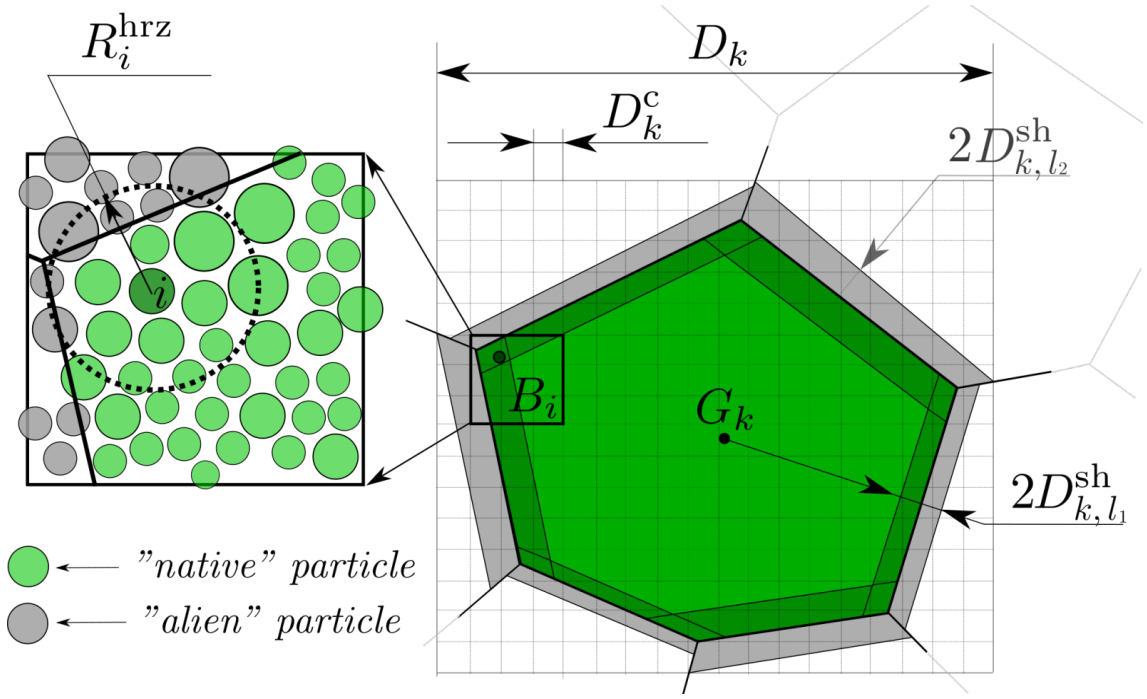
- Для успешной работы алгоритма декомпозиции на подобласти Вороного достаточно координат центра ячейки (частицы) и радиуса поиска соседей в описании элемента среды
- Алгоритм может успешно применяться для декомпозиции расчетной области, в которой имеются элементы разных типов
- Механизм балансировки нагрузки определяется только временем счета и не зависит от типа элементов

Алгоритм поиска ближайших соседей

R_i^{hrz} – радиус вокруг центра элемента, для включения потенциальных соседей в список

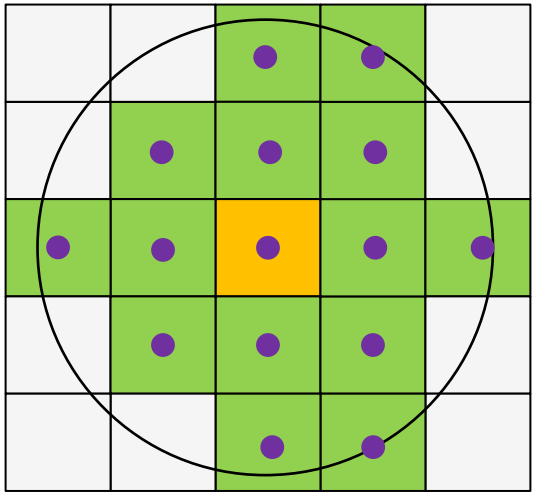
D_k^c – размер ячейки для предварительной сортировки элементов

D_{kl}^{sh} – ширина зоны обмена элементами между ячейками Вороного

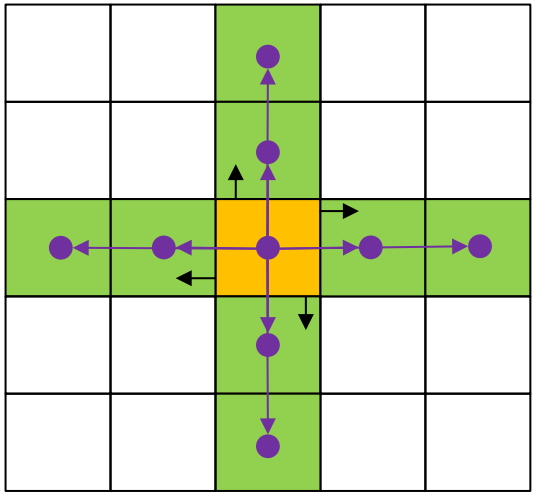


- Для работы алгоритма требуются только центры ячеек и радиус поиска соседа
- *“alien”* элементы хранятся в отдельном массиве

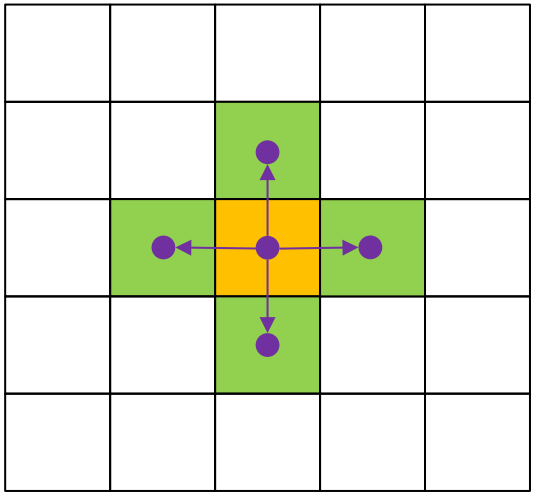
Восстановление топологии сеточных элементов



Определили потенциальных соседей с помощью списка Верле



Для каждой грани нашли $\cos\phi = (\boldsymbol{n}, \boldsymbol{r}_{ij})/|r_{ij}|$, где ϕ - угол между $\boldsymbol{n}$ и $\boldsymbol{r}_{ij}$

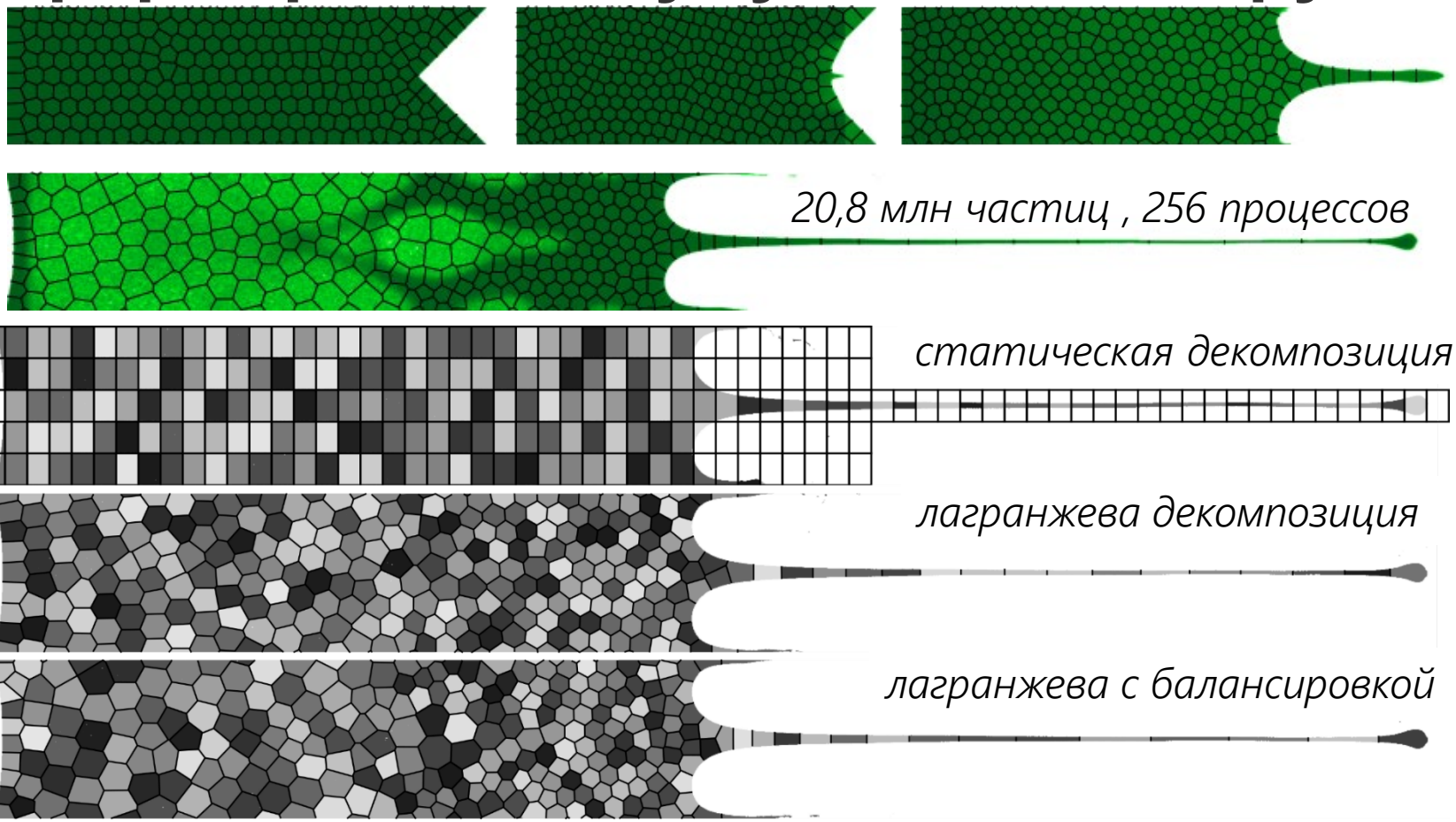


Ячейка с минимальными ϕ и r_{ij} является смежной

ВНИИА
РОСАТОМ

Демонстрация работы алгоритма балансировки нагрузки

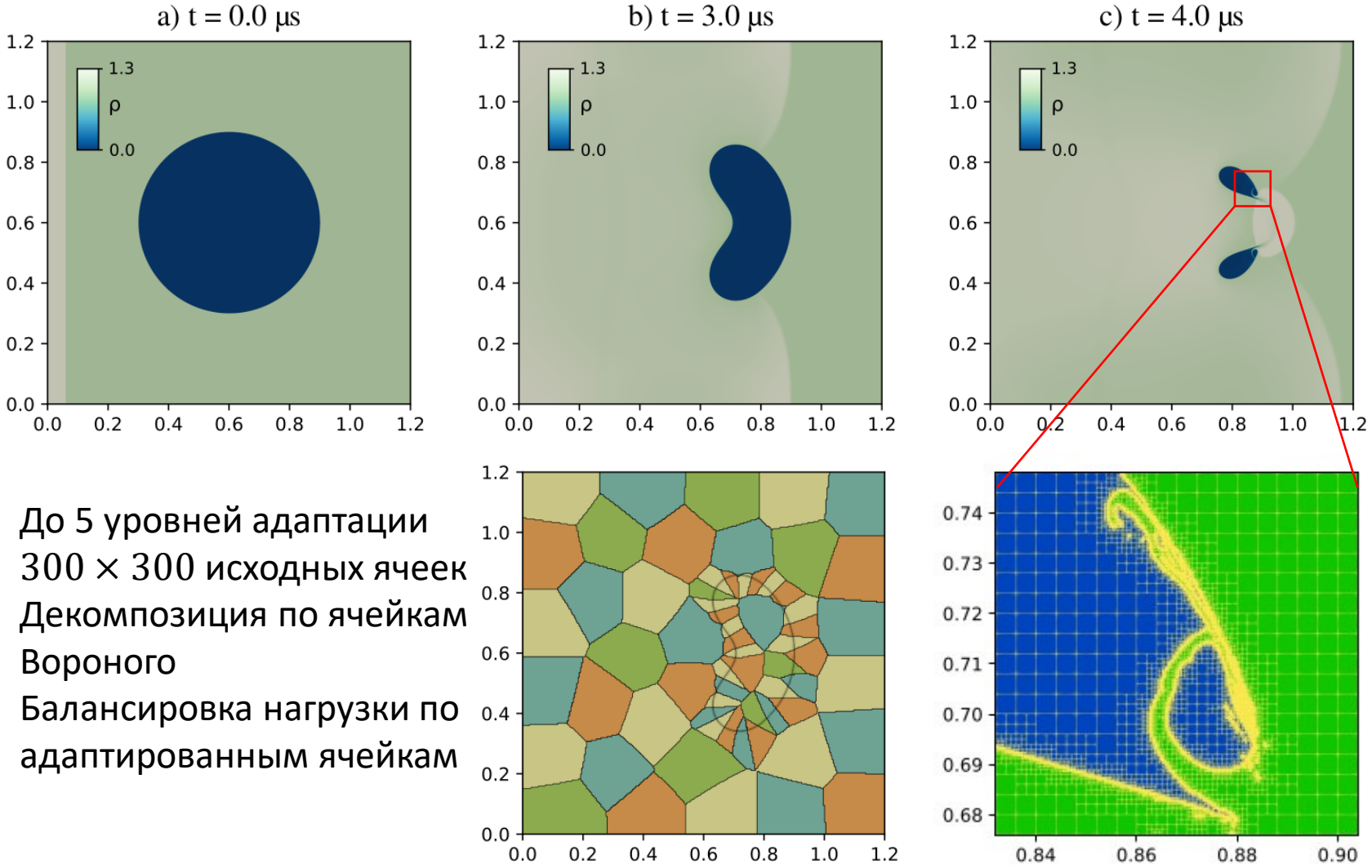
Динамическая задача, формирование кумулятивной струи



- Частицы каждой ячейки Вороного обрабатываются своим процессором
- Эффективное использование вычислительных ресурсов – поддерживается хороший баланс нагрузки, стало возможным моделировать до 1 млрд. частиц!

M. S. Egorova et al. Computers Physics Communications **234,** 112-125 (2018)

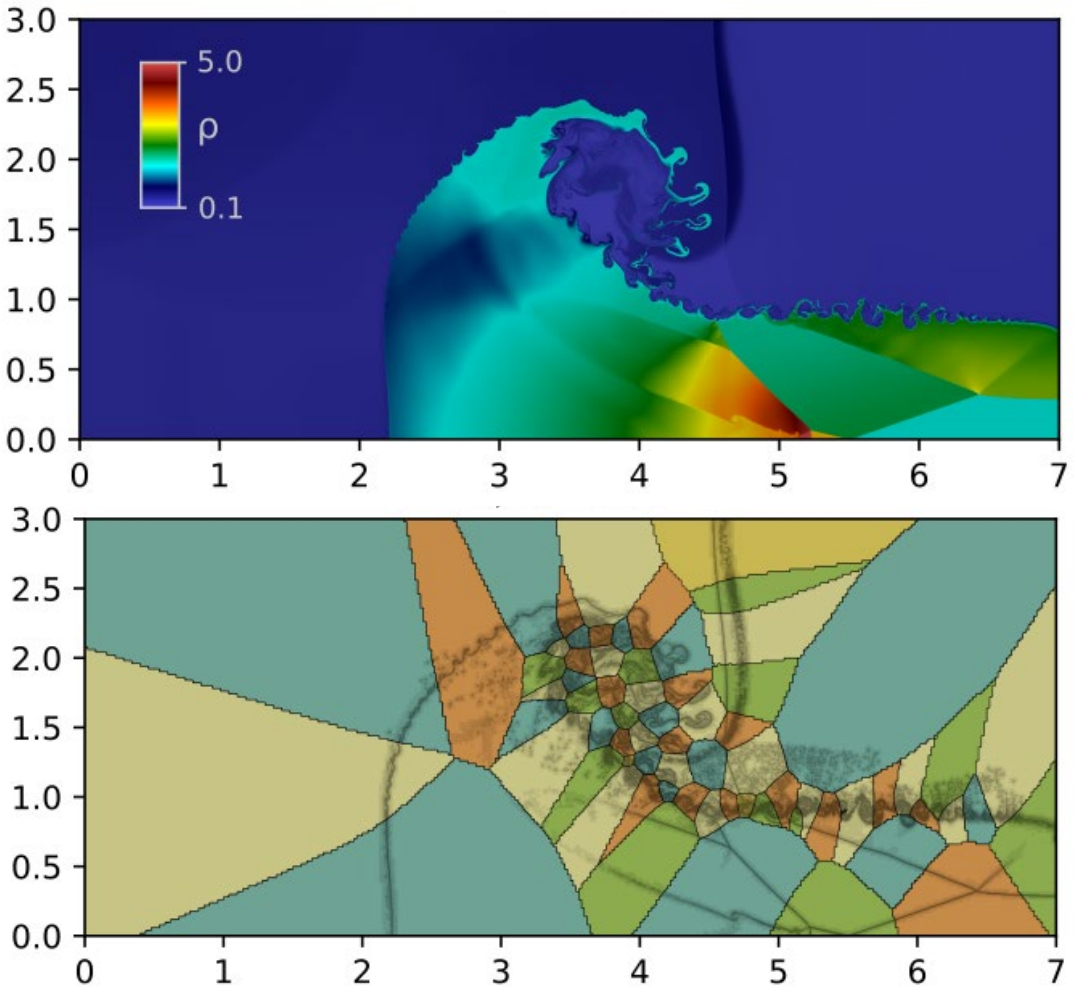
Декомпозиция сеточной области с локальной адаптацией



- До 5 уровней адаптации
- 300×300 исходных ячеек
- Декомпозиция по ячейкам Вороного
- Балансировка нагрузки по адаптированным ячейкам

R. V. Muratov et al. Computers Physics Communications 108790 (2023)

Декомпозиция сеточной области с локальной адаптацией



R. V. Muratov et al. Computers Physics Communications 108790 (2023)

ВНИИА
РОСАТОМ

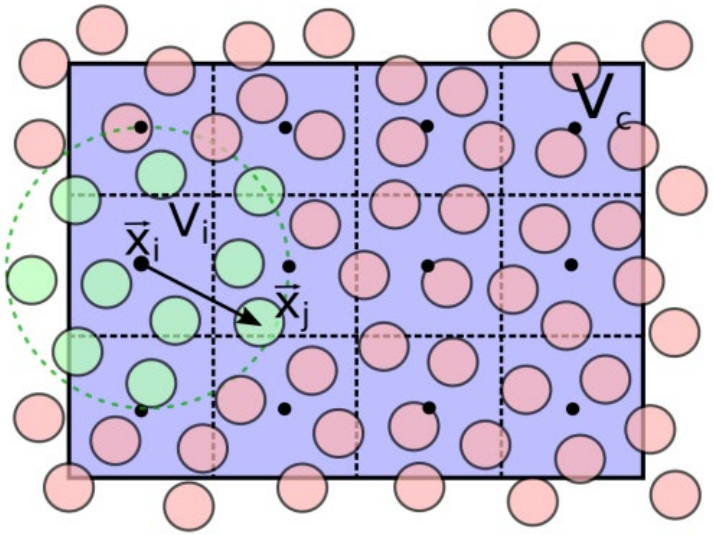
Сопряжение сеточного метода Эйлера с SPH

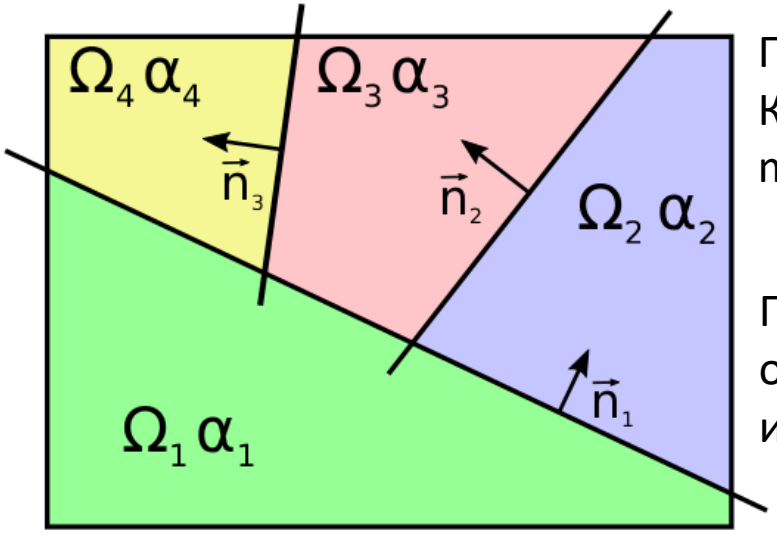
Конвертация данных

Пусть P – набор частиц, C - набор ячеек сетки. Конвертация частиц в сетку (particles to mesh, ptm) осуществляет переход:

$$ptm\left(\pi_{\boldsymbol{v},\rho,e,\alpha}P\right) \rightarrow \pi_{\boldsymbol{v},\rho,e,\beta}C$$

$$f_c = \sum_{i:\,|\boldsymbol{r}_c - \boldsymbol{r}_i|<h} \Delta V_i f_p^i \, W(|\boldsymbol{r}_c - \boldsymbol{r}_i|, h) + \mathcal{O}(h^2)$$





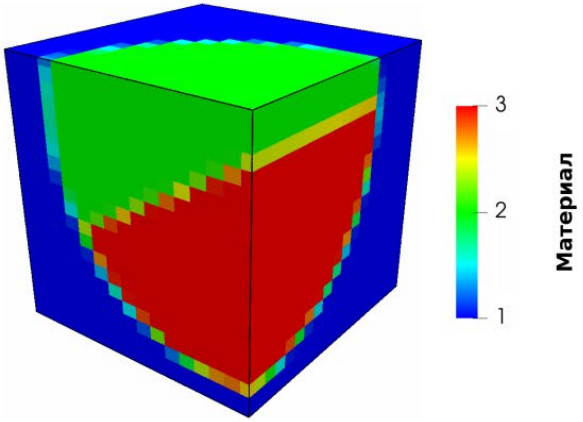
Пусть P – набор частиц, C - набор ячеек сетки. Конвертация сетки в частицы (mesh to particles, mtp) осуществляет переход:

$$mtp\left(\pi_{\boldsymbol{v},\rho,e,\beta}C\right) \rightarrow \pi_{\boldsymbol{v},\rho,e,\alpha}C$$

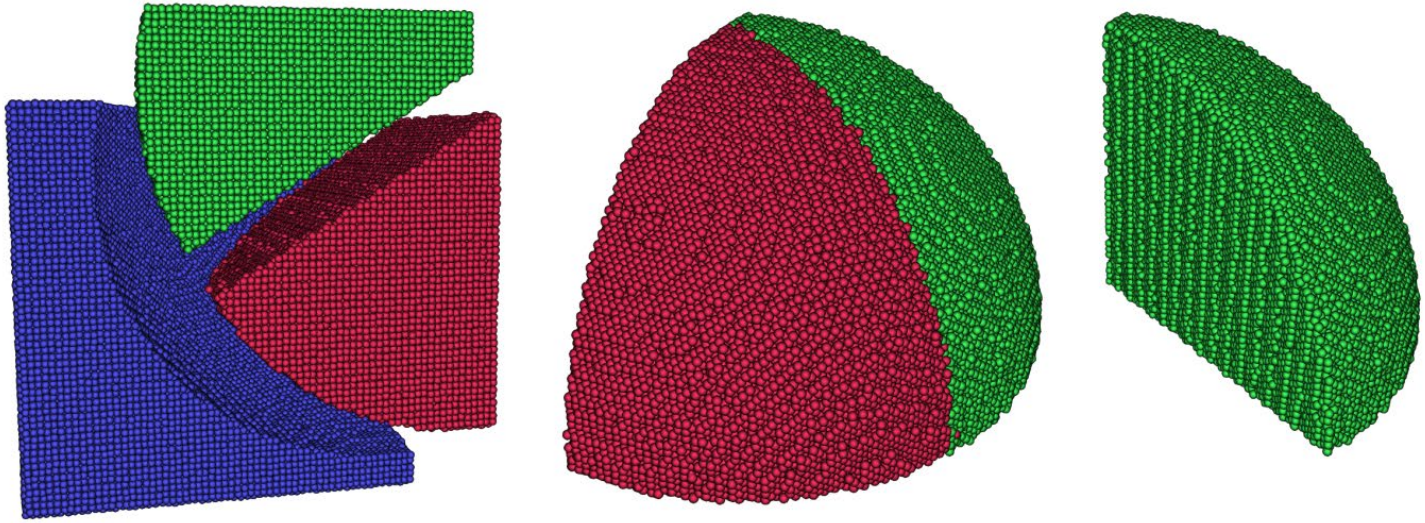
По градиентам объемных концентраций определяются контактные границы материалов, и затем переносятся данные

$$f_p = f_c + \nabla f \cdot (\boldsymbol{r}_p - \boldsymbol{r}_c)$$

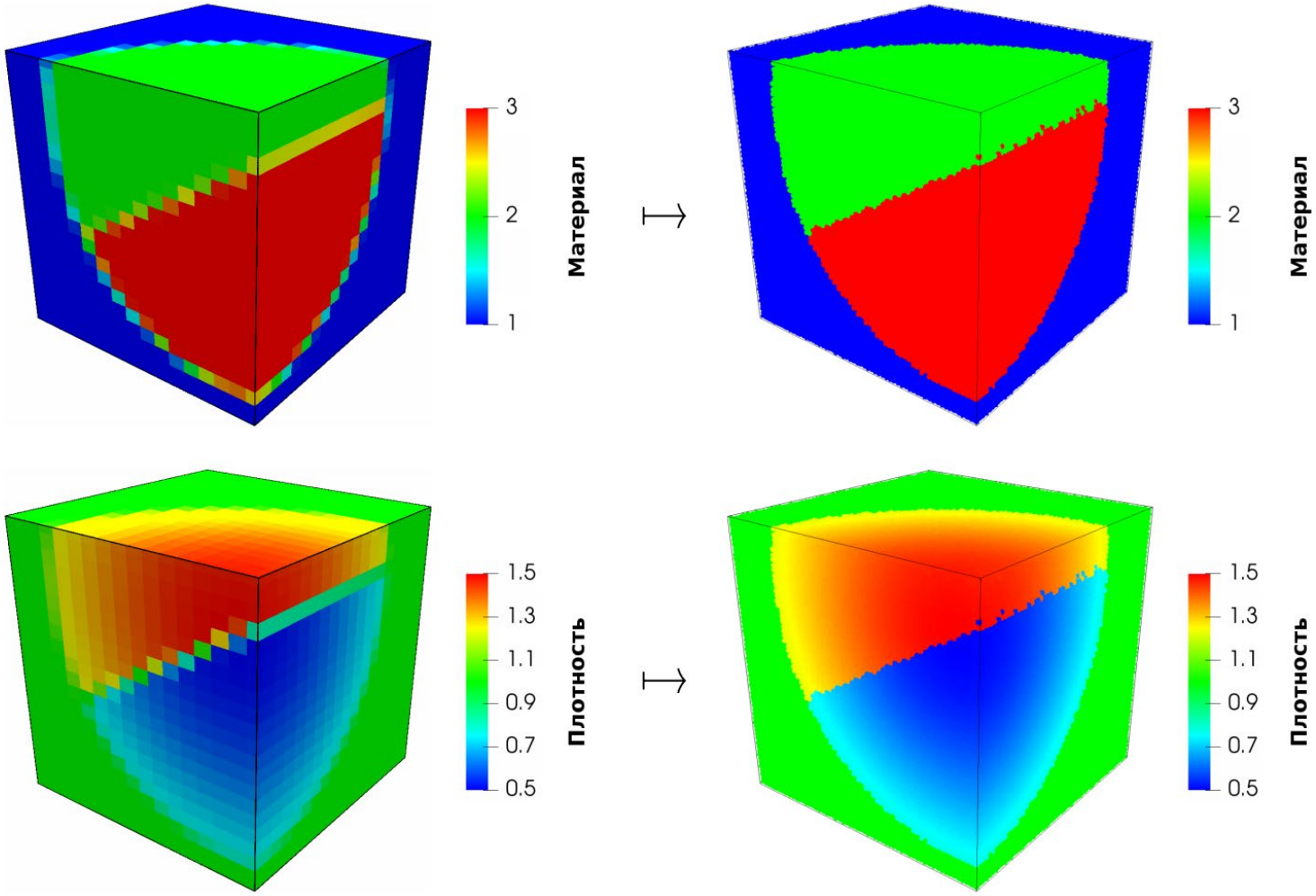
Одномоментный пересчет сетка → частицы



Восстановление границы в задаче с тремя материалами. Слева материалы на исходной сетке, внизу показаны материалы частиц после конвертации.

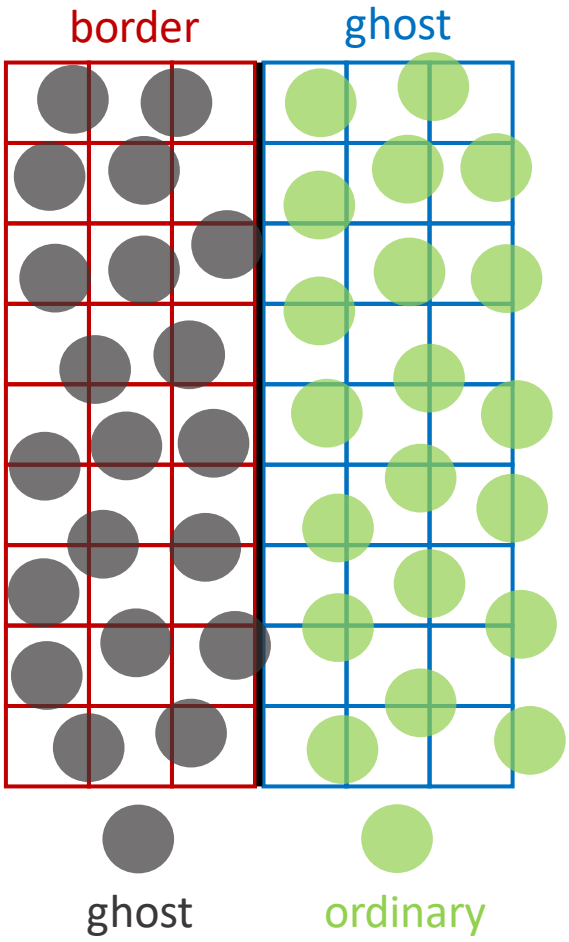


Одномоментный пересчет сетка → частицы



Интерфейс сетка-частицы

ВНИИА
РОСАТОМ



- В области ghost данные переносятся с ordinary частиц
- В области border генерируются ghost частицы по данным с сетки
- При переходе через интерфейс, частицы ghost становятся ordinary
- ghost частицы воздействуют на ordinary
- с ghost ячеек данные переносятся в border

$$\mathcal{P} := \mathcal{P} \cup \mathrm{gen}\left(\mathcal{C}^{\mathrm{bord}}_{G_c},\, \mathcal{P}_{G_p}\right); \qquad \mathcal{P}^{\mathrm{gh}} := \left\{p \in \mathcal{P} :\ p_{coords} \in \mathcal{C}^{\mathrm{bord}}_{G_c}\right\};$$

$$\mathcal{C}_{\boldsymbol{\alpha}} := \boldsymbol{\alpha}(\mathcal{C}_{\rho,p,\boldsymbol{\beta}});$$

$$\mathcal{P}^{\mathrm{gh}}_{\rho,\boldsymbol{v},p,\boldsymbol{\alpha},\boldsymbol{\beta}} := \mathrm{mtp}\left(\mathcal{C}^{\mathrm{bord}}_{G_c,\rho,\boldsymbol{v},p,\boldsymbol{\alpha},\boldsymbol{\beta}},\, \mathcal{P}^{\mathrm{gh}}_{G_p}\right);$$

$$\mathcal{P}^{\mathrm{gh}}_{e} := e\left(\mathcal{P}^{\mathrm{gh}}_{\rho,p,\boldsymbol{\alpha}}\right);$$

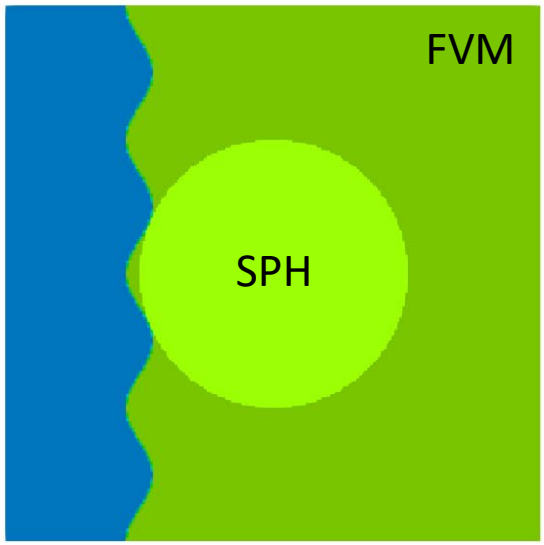
$$\mathcal{C}^{\mathrm{gh}}_{\rho,e,\boldsymbol{\beta}} := \mathrm{ptm}\left(\mathcal{P}_{G_p,\rho,\boldsymbol{v},e,\boldsymbol{\alpha}},\, \mathcal{C}^{\mathrm{gh}}_{G_c}\right), \qquad \mathcal{P}^{\mathrm{ord}} := \left\{p \in \mathcal{P} :\ p_{coords} \in \mathcal{C}^{\mathrm{gh}}_{G_c}\right\};$$

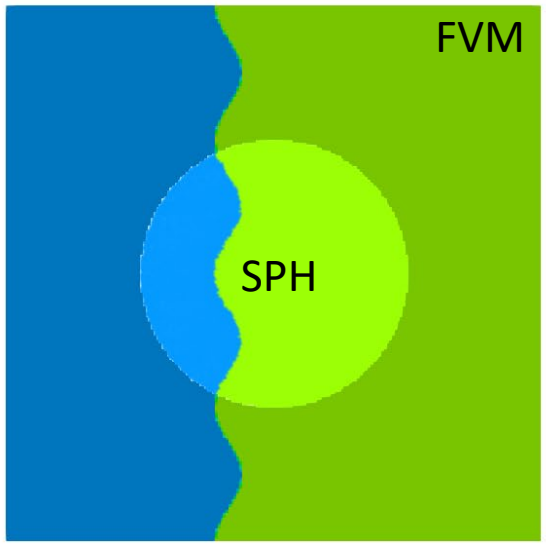
$$\mathcal{C}^{\mathrm{gh}}_{p} := p\left(\mathcal{C}^{\mathrm{gh}}_{\rho,e,\boldsymbol{\beta}}\right);$$

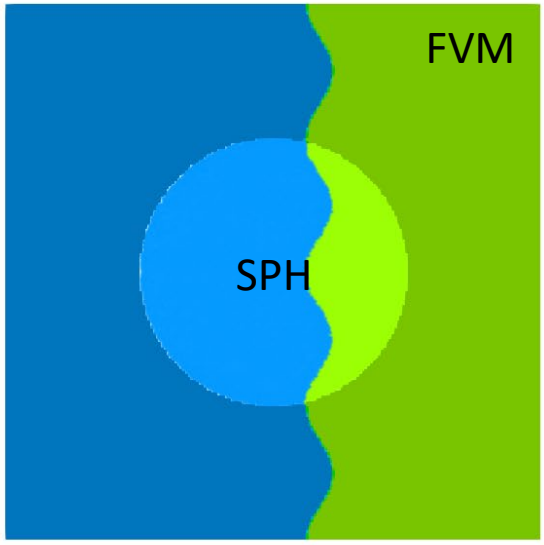
$$\mathcal{P}_{p,c} := \mathrm{eos}(\mathcal{P}_{\rho,e,\boldsymbol{\alpha}});$$

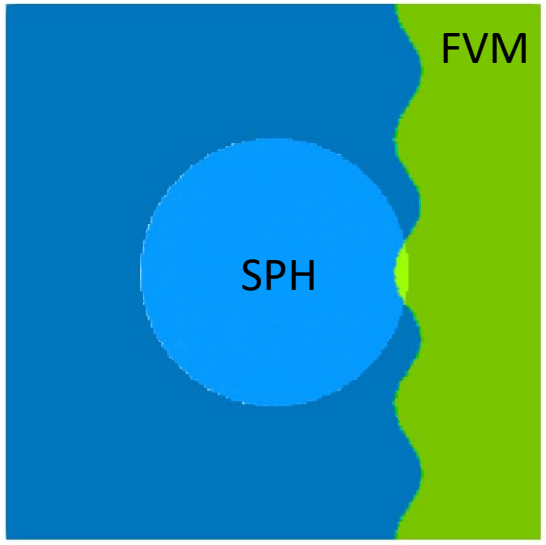
Перенос возмущенной границы

ВНИИА
РОСАТОМ





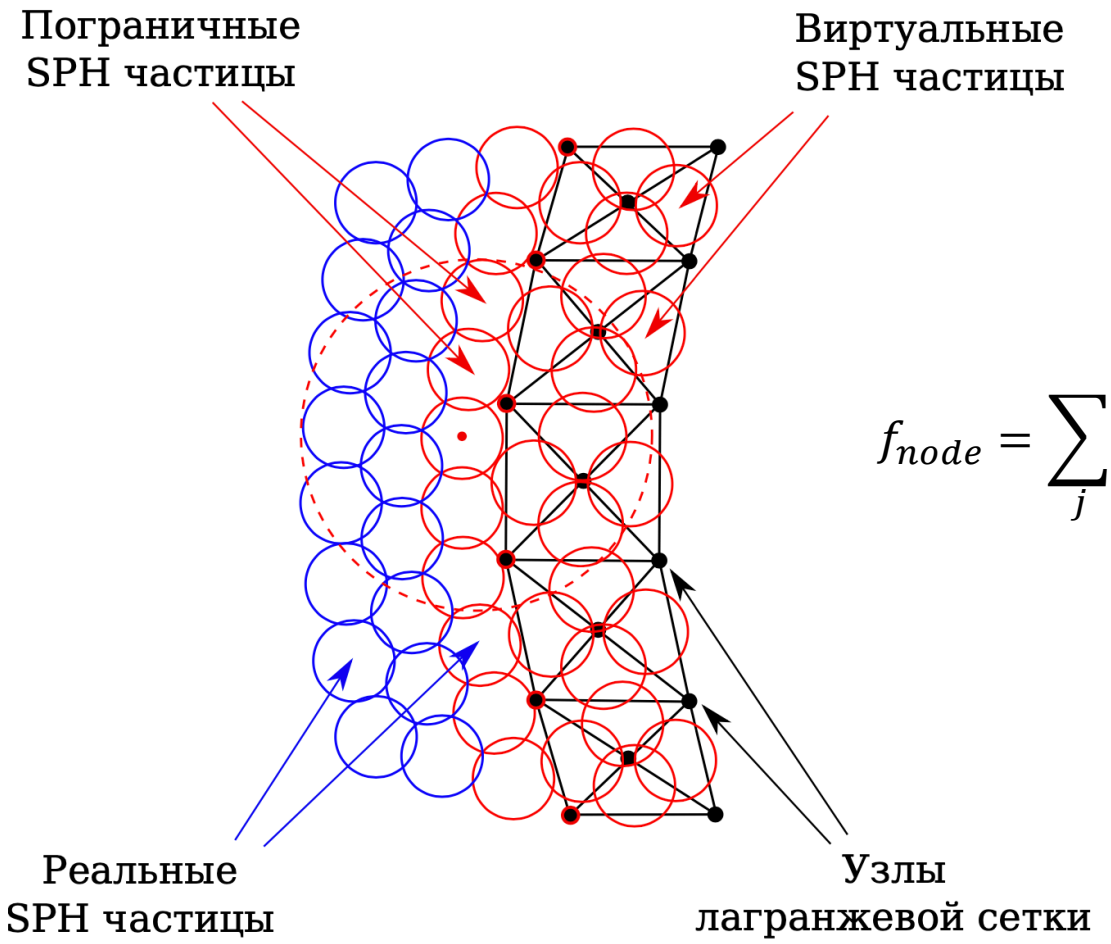




ВНИИА
РОСАТОМ

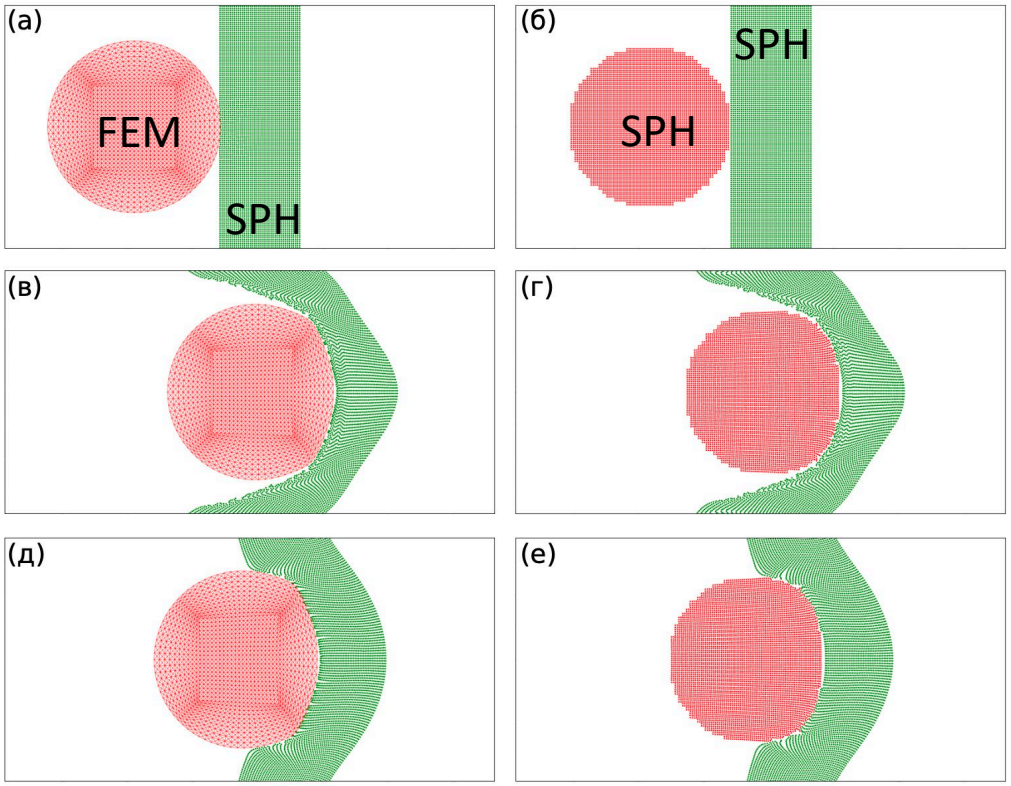
Сопряжение сеточного метода Лагранжа с SPH

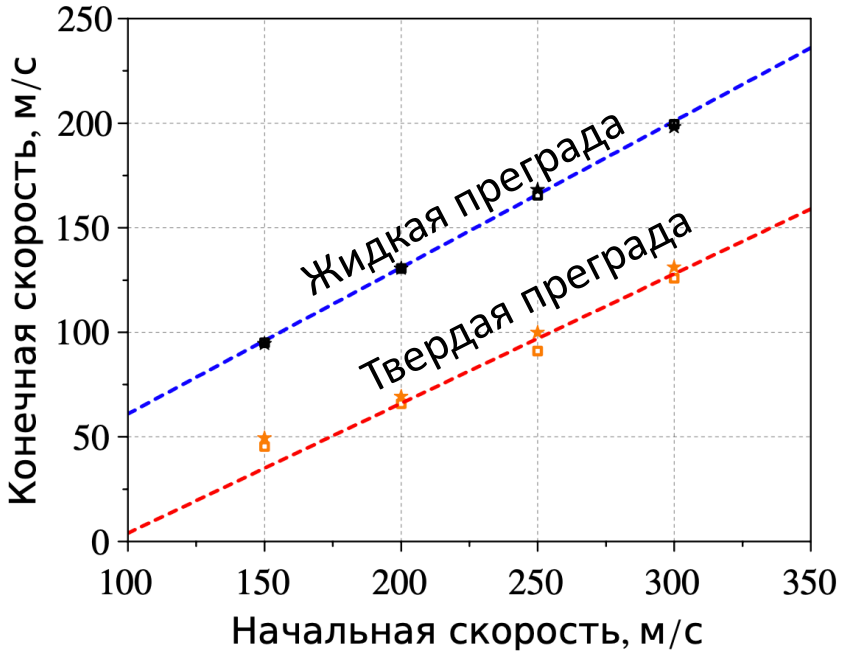
Схема сопряжения сетка-SPH



$$f_{node} = \sum_j \left(f_j^{mesh} + \frac{1}{dim+1} f_j^{SPH} \right)$$

Пробитие преграды





Шарик из твердого алюминия (сеточный метод Лагранжа) пробивает преграду из твердого или жидкого алюминия (SPH)

Важность использования Python

ВНИИА
РОСАТОМ

- Python – интерпретатор соответствующего языка, написан на C
- Python – основной язык Data Science
- Python-интерфейс к высокопроизводительным библиотекам имеется для многих популярных приложений:
  - Машинное обучение (sklearn, tensorflow, pytorch и др.)
  - Математика и статистика (NumPy, SciPy, pandas, и др.)
  - Гидродинамика и УЧП (FEniCS, FluidSim, …)
  - Биоинформатика (целый раздел bioconda) и другие
- Python позволяет легко интегрировать несколько библиотек в рамках одного расчета
- Python не требует компиляции, поэтому код можно сразу писать (или генерировать) и исполнять
- Программы с Python-интерфейсом легко интегрируются в клиент-серверную архитектуру с помощью библиотеки Django
- Студенты хорошо владеют Python, могут сразу решать задачи

Заключение

ВНИИА
РОСАТОМ

- Разработана программная платформа PyPHIA (Python Parallel Hydrodynamics Integrated API), в рамках которой обеспечен единый формат представления данных и общий алгоритм параллелизации вычислений
- В рамках платформы успешно реализованы сеточные и бессеточные методы решения задач механики сплошной среды
- Реализованы алгоритмы сопряжения сеточных методов с бессеточным методом SPH, что позволяет решать задачи разными численными методами одновременно
- Для выскопроизводительных модулей реализованы соответствующие интерфейсы на языке Python, поддерживающие NumPy, что позволяет проводить интеграцию модулей комплекса с большим количеством Python-библиотек

Спасибо
за внимание

SADyachkov@vniia.ru