

# ПОСТРОЕНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ И ПРОГРАММНЫХ МОДЕЛЕЙ С ПОМОЩЬЮ КОМПОЗИЦИИ ОБЪЕКТОВ

- Типовая текущая ситуация - декомпозиция/композиция на программном уровне представления физической области:
  - либо ручном режим с большими трудозатратами,
  - либо полуавтоматически только для простейших случаев.
- Предлагаемое решение – декомпозиция на уровне физической модели / композиция на вычислительном и программном уровнях.

# Метод решения

- Стандартизация интерфейсов для объектов, из которых собирается модель, и их связей.
- Суть предлагаемого решения – локальность всех описаний: связей, времени и алгоритмов эволюции подобластей.
- Модель всей области получаем практически автоматически путем композиции подобластей. **Буквальный цифровой аналог натурального моделирования.**

# Текущее состояние разработки

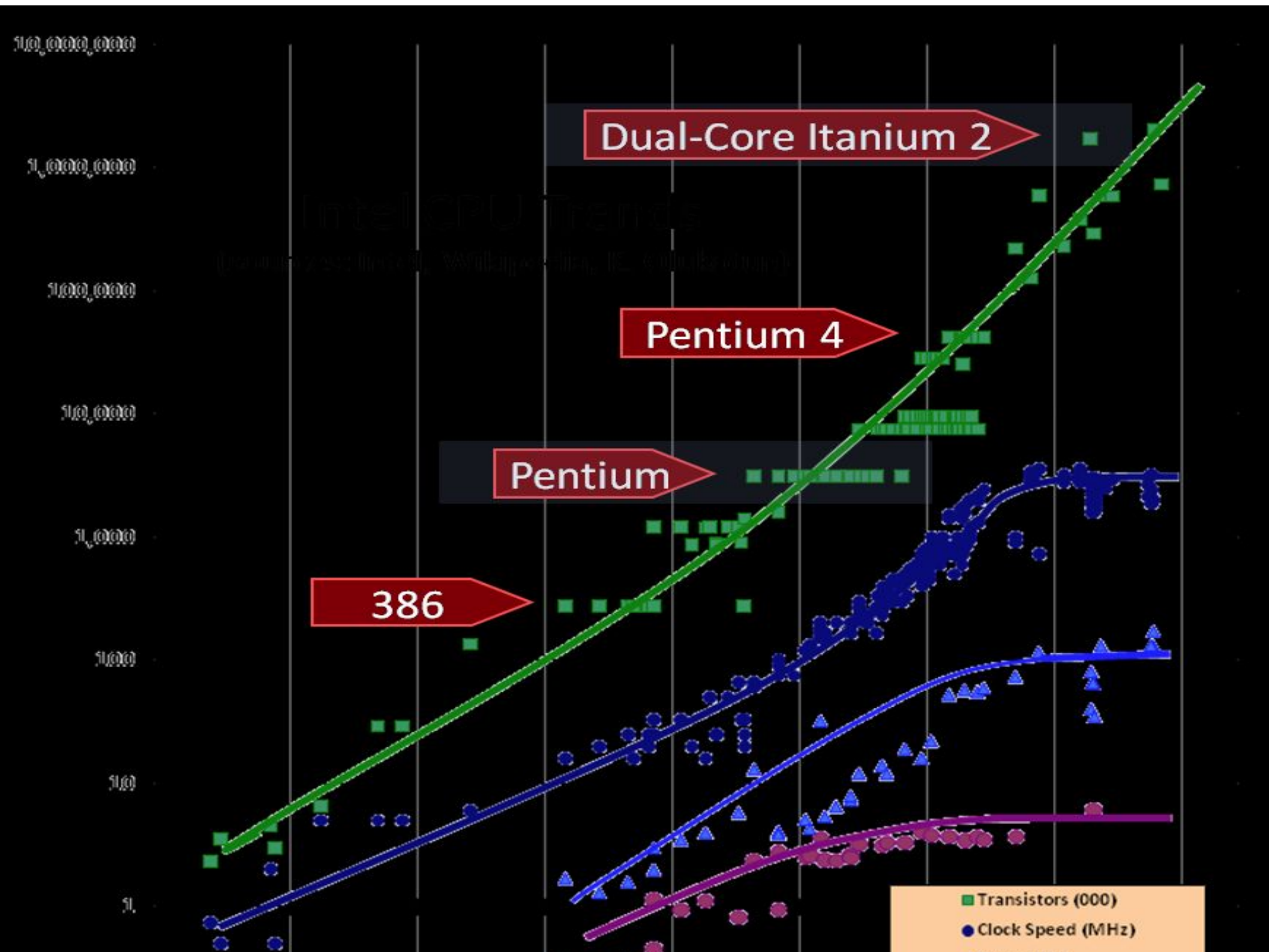
**В настоящий момент** отлаживается **третья версия** системы OST (Objects –Space – Time) на K100 (ИПМ РАН).

**Первая версия** - 2009 год, кластер из нескольких РС, тесты, решение одной двумерной задачи газовой динамики (M2DGD) и одномерной задачи теплопроводности.

**Вторая версия** – 2010 год, RSC4 (ИПМ РАН) – варианты двумерной задачи газовой динамики, тестовые задачи.

**Кто работает:**

1. Два сотрудника ИПМ РАН и кафедры «Вычислительная механика» МГУ.
2. Аспиранты и студенты МГУ и ИПМ РАН.



**Главная проблема:** Джон Хэннеси, президент

Стэнфордского университета –

“... когда мы начинаем говорить о параллелизме и легкости использования действительно параллельных компьютеров, мы говорим о проблеме, которая труднее любой проблемы, с которой встречалась наука о компьютерах ... Я бы запаниковал, если бы я работал в промышленности.”

**Главная цель** – свести трудоемкость создания параллельных моделей к трудоемкости создания последовательных моделей для максимально широкого класса задач.

**Миф** о последовательном характере большинства решаемых задач

**Уровни представления физической области:**

**Физическая модель -> Математическая модель ->**

**Вычислительная модель -> Программная модель -> МВС**

**Подобие уровней представления:**

- Сквозное употребление терминов. Объект, связь, интерфейс,

...

- Подобие проблем на разных уровнях

- Параллельная эволюция частей и их взаимодействие

- В общем случае невозможно восстановить из последовательной программы (алгоритма) исходный естественный параллелизм физической области.

- Буквальное отображение параллельной физической модели на параллельные вычислительную и программную модель.

Сохраняется деление на части и структура связей.

**Вычислительная модель - множество вычислительных объектов (каждый объект - набор матриц, векторов, скаляров плюс последовательный алгоритм – это представление физической подобласти) отображается на множество программных объектов, распределенных по МВС (RPC) – программная модель.**

3. Способы композиции на уровне вычислительной модели.

3.1. Метод Шварца для декомпозиции областей.

3.2. Метод композиции вычислительных объектов – построение области путем композиции первичных подобластей с выделением приграничных полос между подобластями.

3.3. Использование приграничных «потоков» для композиции подобластей.

и т.д.

Конкретных способов много. Стандарты для достаточно широкого класса случаев отсутствуют.



# РЕКУРСИВНАЯ ДЕКОМПОЗИЦИЯ

Дано: сетка  $(\Delta t, \Delta x, \dots)$ ,  $S$ ,  $S_i$ ,  $L(\Delta t)$   $S_i$  – решатель для  $i$ -ой подобласти  
 **$L$  – максимальное расстояние, на которое может распространиться сигнал (возмущение) за  $\Delta t$ .**

Решетка из полос ширины  $2 * L$  – некорректные значения после первого шага.

Независимая

эволюция «внутри»

подобласти

в течении  $\Delta t$ .

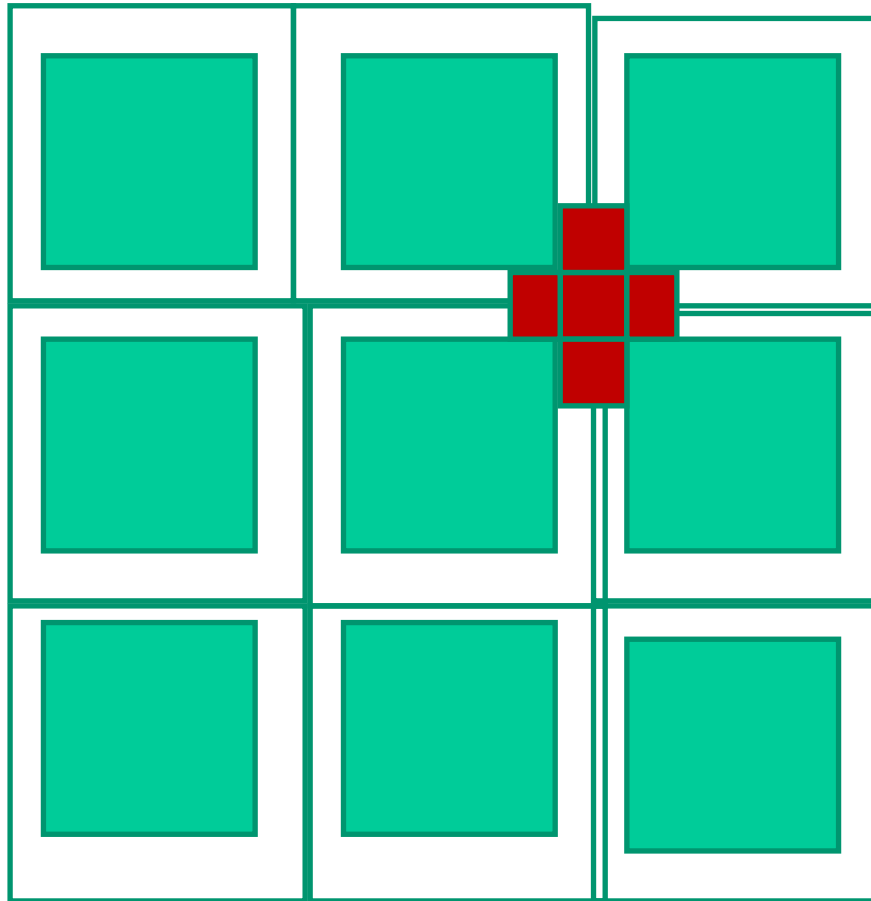
$K_p > 0.9$

$T_{шага} > 10 * T_{коррекции}$

Аналогия: физ. среда –

аппаратура – одни и те же

физ. ограничения



Кресты – «диаметром»  $4 * L$  – некорректные значения после второго шага

## 4. Композиция программных объектов в системе OST (Objects – Space -Time).

### 4.1. Определение интерфейса между объектом и его окружением.

4.1.1 Интерфейс самого объекта для объектов из его окружения (окрестности) в виде списка операций, выполняемых объектом, и их параметров.

Язык JAVA

```
interface Left_Neighbour {  
int get_border(double left_array[]);  
}
```

4.1.2. Интерфейс окружения для объекта в виде “списка формальных соседей” - объектов с их интерфейсами.

(Left\_Neighbour Left, Right\_Neighbour Right)

Здесь:

Left\_Neighbour – указание на интерфейс с соседом;

Left – имя ссылки на соседа, в которую OST после определения связей подставит ссылку на фактического соседа.

Вызов операции в соседе разрешен всегда, но фактически будет выполнен только при совпадении локальных времен соседей.

# Пример прикладного класса

Язык Python

```
class objectExample (OST_Object_Abstract) :
    double left_array[];
    <...>
    def run () :
        <...>
        #вызов метода в соседе
        self.Left_Neighbors.get_border(double left_array[]);

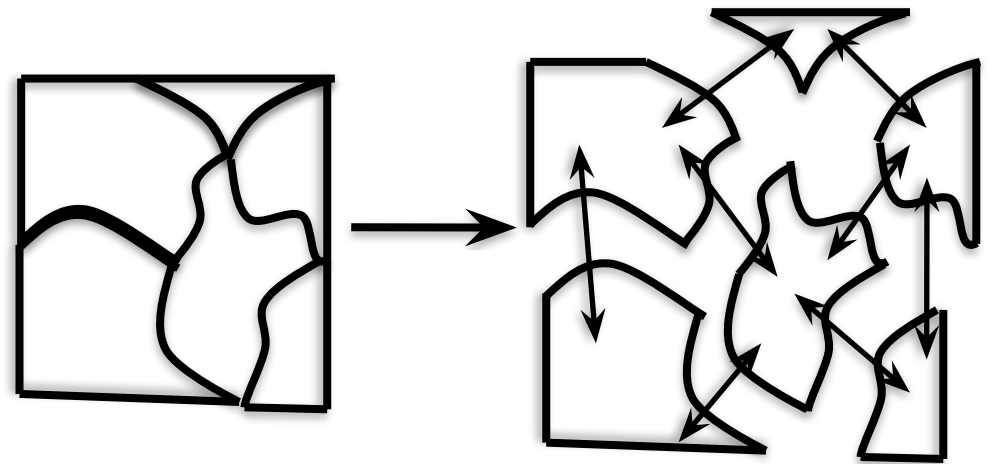
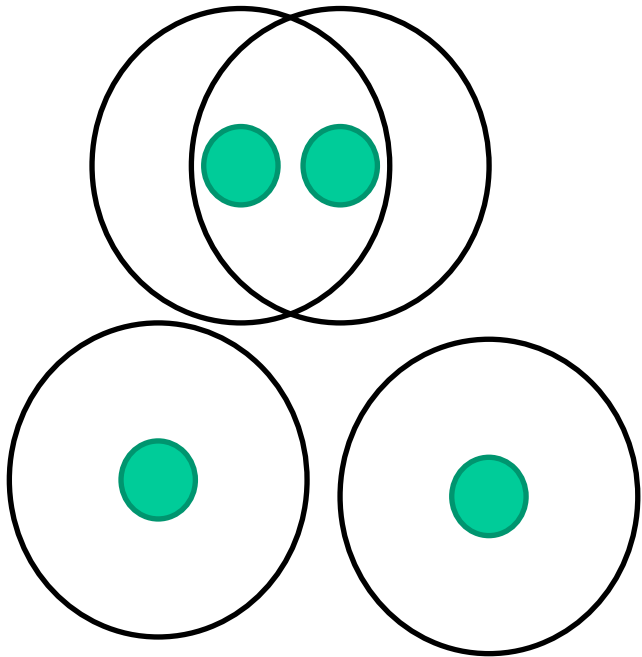
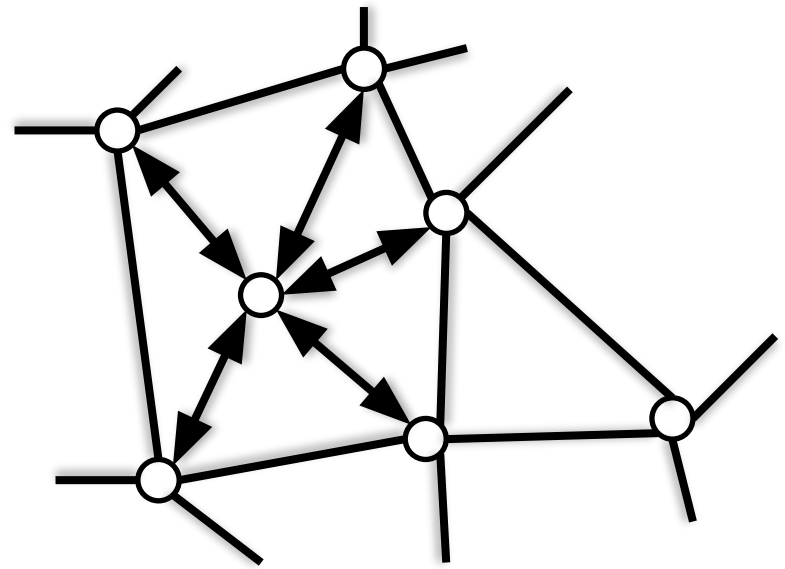
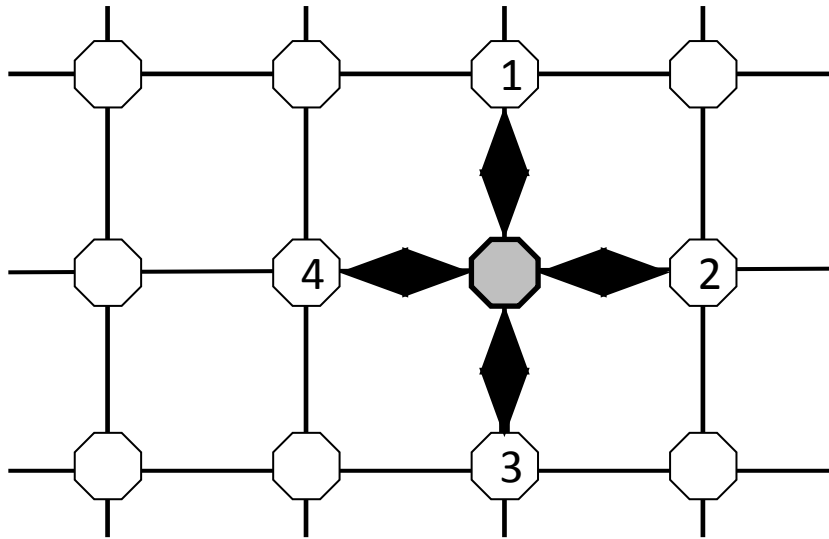
        <...>
        self.time += 1
        #изменение координат
        if <...>:
            self.x += 1

        self.setXYZT ()
```

# Топология связей между объектами.

Определение связей между объектами через локальное описание окрестности для каждого объекта. Автоматическая замена формальных соседей на фактические во время счета.

# Примеры связей

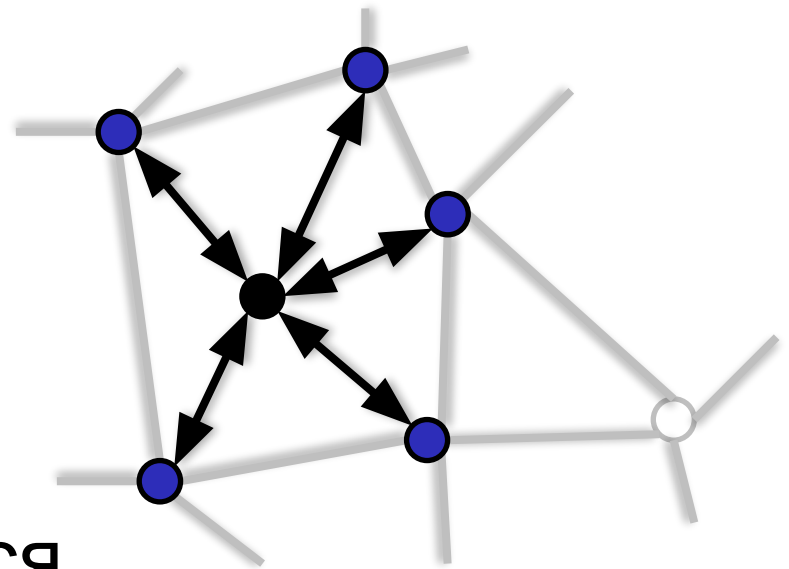


# Окрестность узла

Окрестность - узлы, с которыми соединен данный узел.

Узлы можно отождествить с вычислительными объектами

Окрестность может изменяться по ходу вычислений.



# Локальная топология

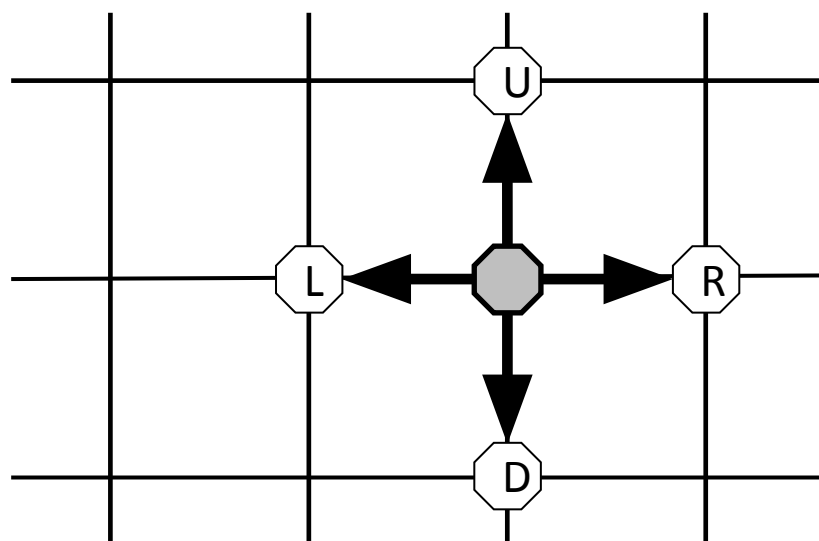
**Локальная топология** – это топология, описывающая множество соседей для данного вычислительного объекта в локальных координатах

**Локальные координаты** – система координат, связанная с окрестностью данного объекта.



# Пример: целочисленная решетка

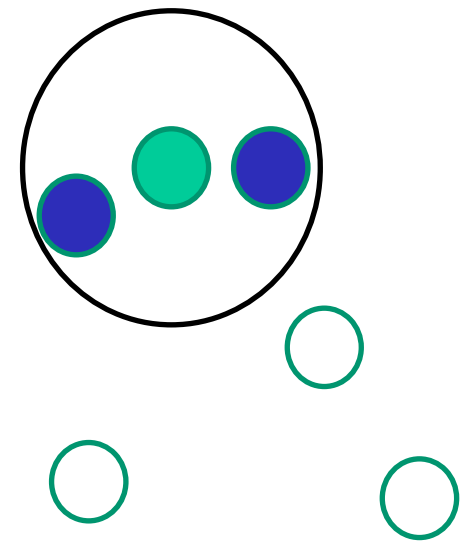
Координаты соседей  
отличаются на  $\pm 1$  по  
одной из координат



# Пример: плоскость

Соседи удалены не более,  
чем на  $r$ :  $|x - y| \leq r$

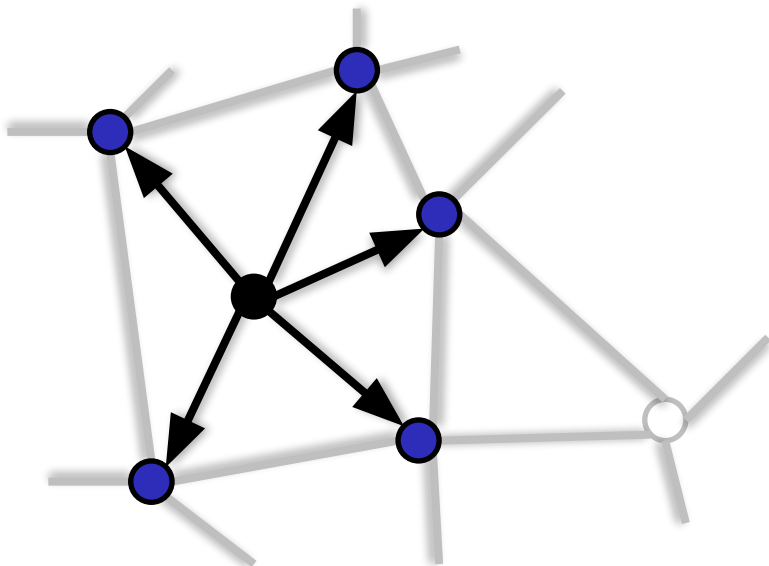
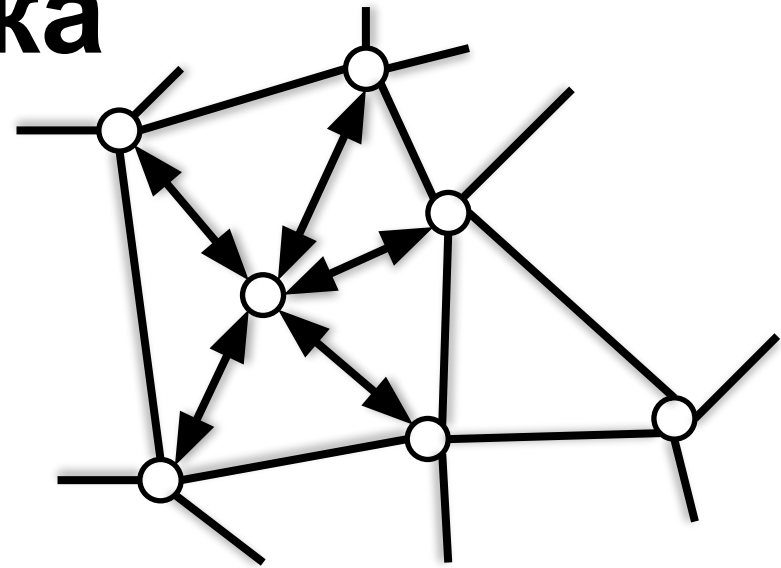
Функция близости проверяет  
критерий попадания  
в окрестность (круг)



# Пример: неструктурированная сетка

Все вершины графа пронумерованы 1,2,3,...

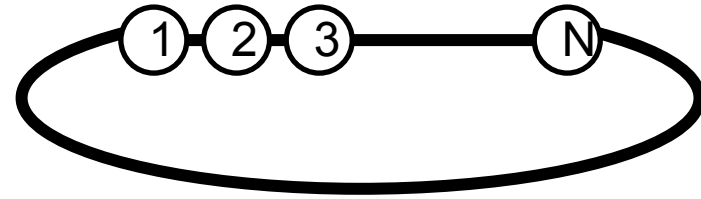
Для каждого узла явное описание списка соседей



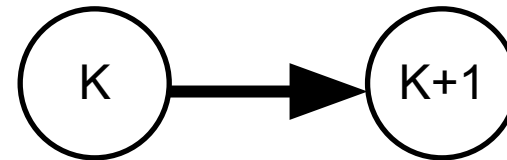
Простое использование формата METIS для такого описания

# Пример: кольцо

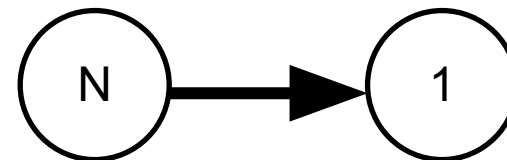
Кольцо объединяет в себе 2 вида локальных топологий



$N - 1$  одинаковых одномерных



1 вырожденная, замыкающая кольцо



# Базовая топология – связный граф

## Случай неструктурированной сетки

```
#Создание объектов в функции инициирования модели  
for <...>:
```

```
    myobject = objInit.createObject(objectClass)
```

```
<Заполнение данными объекта myobject >
```

```
    object.array1 = <...> ...
```

```
<Передача базового объекта topology в object>
```

```
myobject.topology = topologyLocal()
```

```
<Занесение номера объекта в глобальный каталог>
```

```
    objInit.topology.set(myobject, index)
```

```
# Задание окрестности для каждого объекта
```

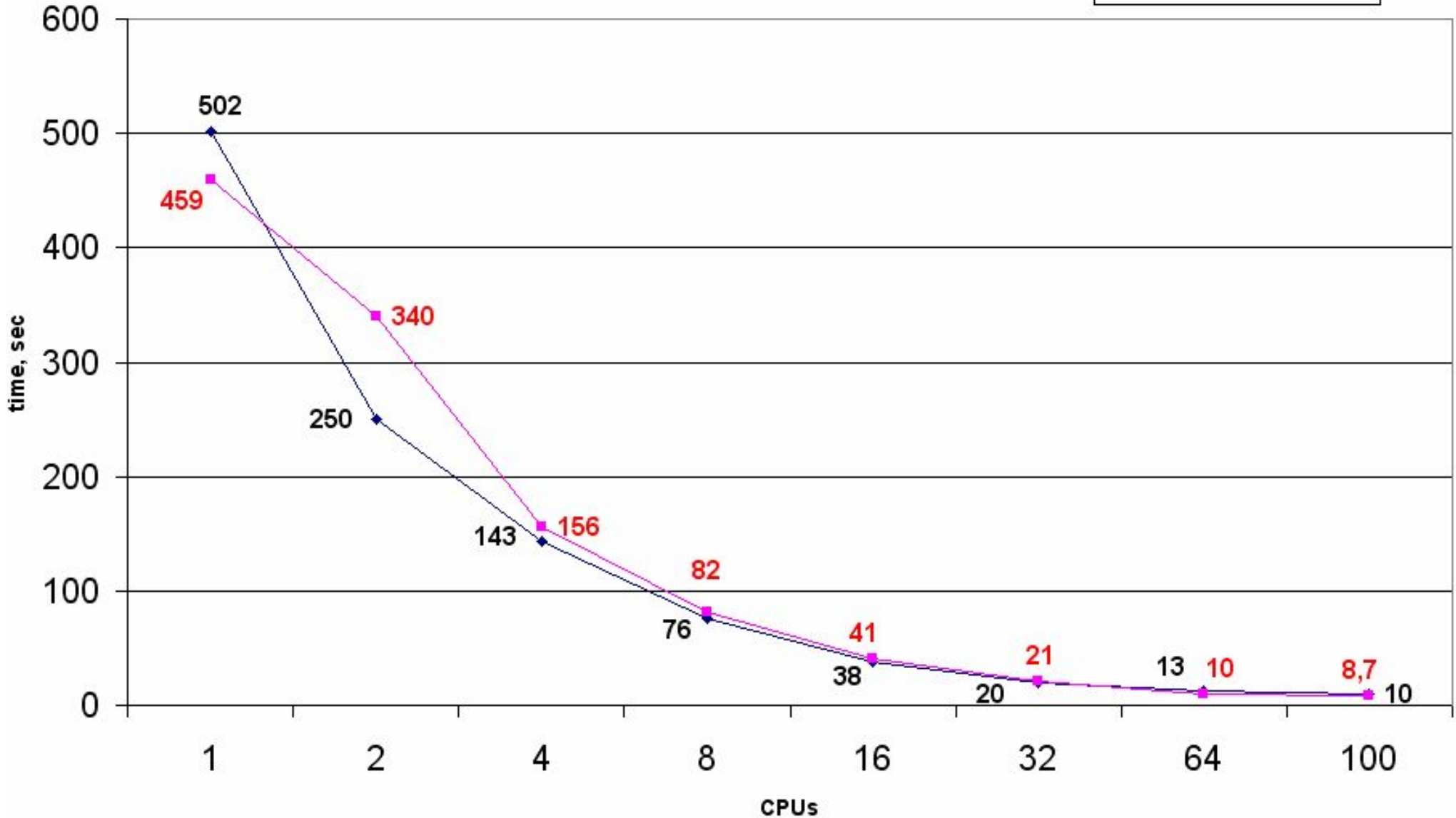
```
for <...>:
```

```
<Определение списка соседей>
```

```
    myobject.topology.set(  
        index, [nei_index1, ..., nei_indexN] )
```

# Результаты счета, оценка эффективности

1800x540x10, time



# Что конструктивно новое

Формализация «окружения» объекта в виде списка формальных соседей.

Автоматическое построение связей между объектами и замена формальных соседей фактическими.

Актуализация ссылок на фактических соседей при совпадении локальных времен.

# Технические характеристики

1. Коэффициент эффективности параллельного счета  $K_p > 0.9$
2. Практически автоматическая сборка (установление связей, синхронизация взаимодействия по локальным временам объектов) автономно запрограммированных объектов (моделей физических подобластей). То-есть, трудоемкость параллельного программирования сводится к трудоемкости раздельного программированию последовательных алгоритмов для подобластей.
3. Счетная часть объектов – C++, Фортран, управляющая часть Python.
4. Средства отладки – визуализация, отладчик, профилирование и т.д.



5. Хранения множества объектов программной модели в файле объектов (базе данных). Контрольные точки и рестарты. Мобильность хранимой модели – возможность перенести модель на другую вычислительную установку после окончания очередного этапа счета и продолжить его на новом месте. Пример, короткий отладочный счет на РС и продолжение на МВС.
6. Планировщик ресурсов, обеспечивающий автоматическую подкачку/выталкивание программных объектов между процессорами и файлом объектов в стиле подкачки страниц в операционных системах. Счет на основе “рабочего множества” объектов.

# Заключение

Суть предлагаемого решения – локальность всех описаний: связей, времени и алгоритмов эволюции подобластей. Модель всей области получаем практически автоматически путем композиции подобластей.

Дополнительная информация на сайте [ost.kiam.ru](http://ost.kiam.ru)

Перспективы - разнонаправленные