



Совместное использование MPI и OpenMP на кластерах

Киреев С.Е., Городничев М.А., Калгин К.В., Перепелкин В.А.

Отдел МО ВВС
ИВМиМГ СО РАН

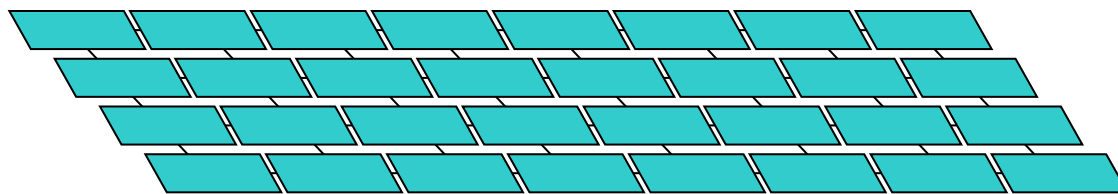


План

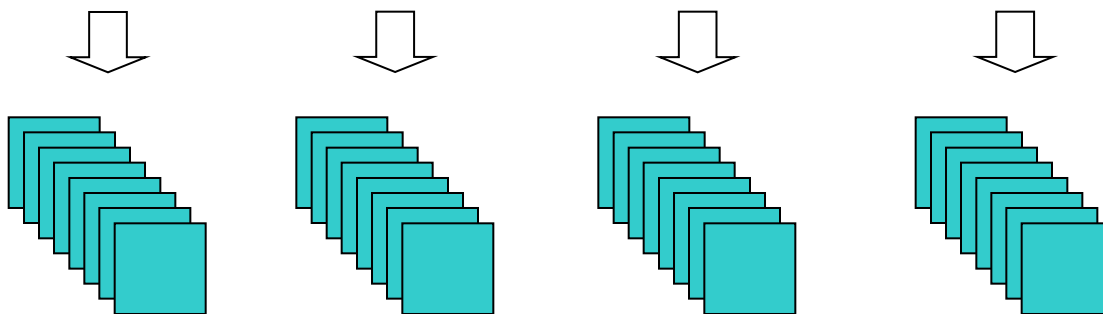
- **Введение в архитектуру: теория**
- Практические следствия
 - Привязка процессов к ядрам
 - Ограничение пропускной способности памяти узла
 - Планирование (shedule) обхода данных несколькими потоками в OpenMP
 - Использование несколькими потоками одной кэш-строки
- Сравнение производительности MPI и MPI+OpenMP
- Выводы

Отображение процессов MPI на узлы кластера

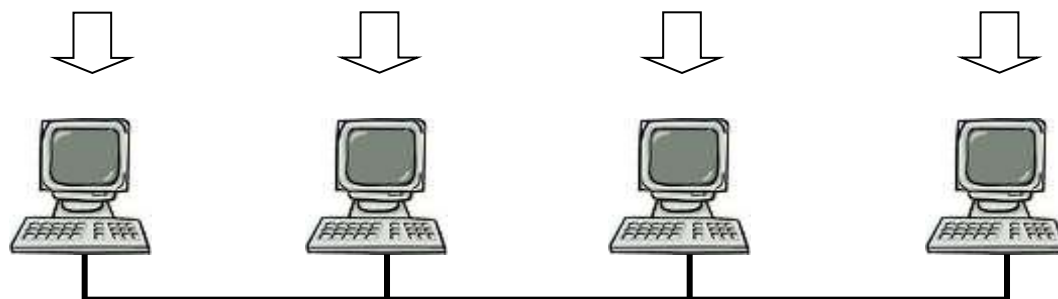
MPI-программа



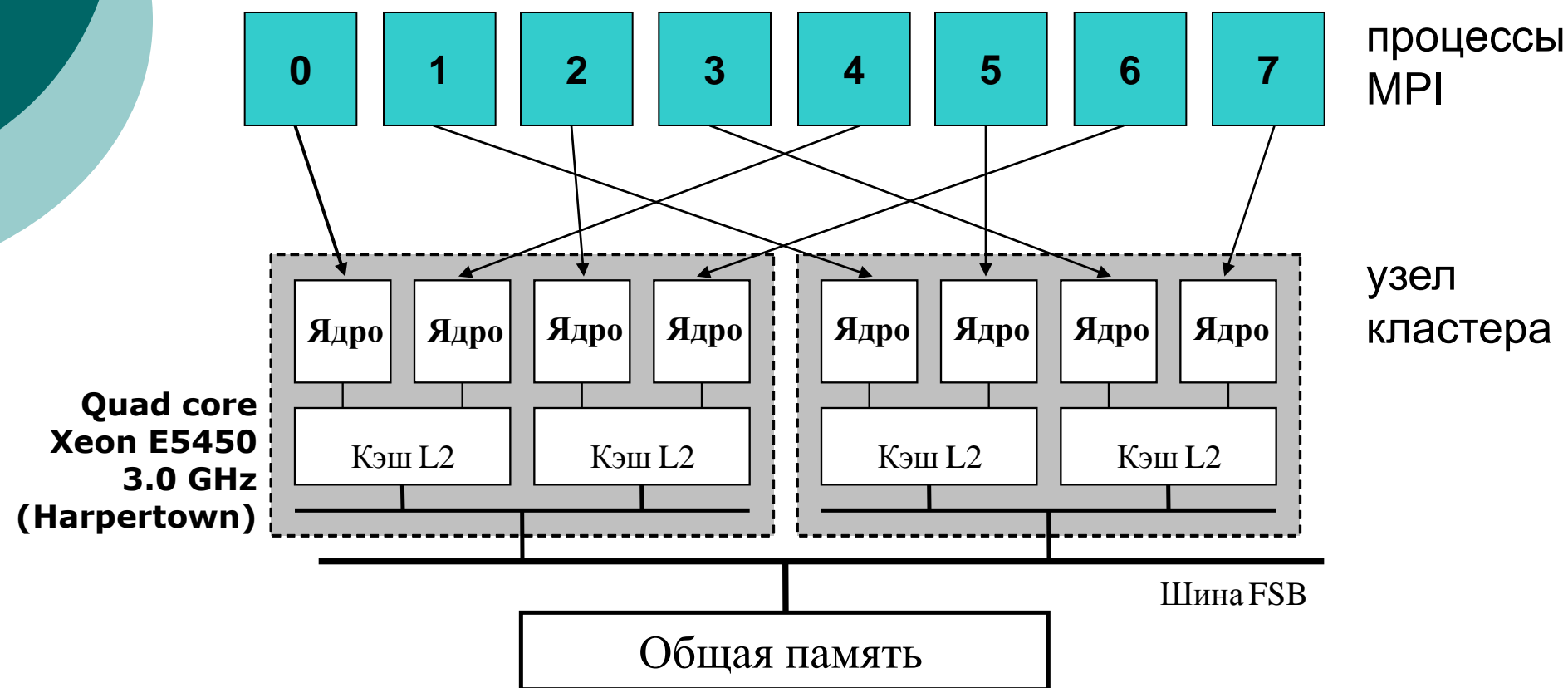
MPI-процессы



Узлы кластера



Отображение процессов MPI на ядра в узле кластера



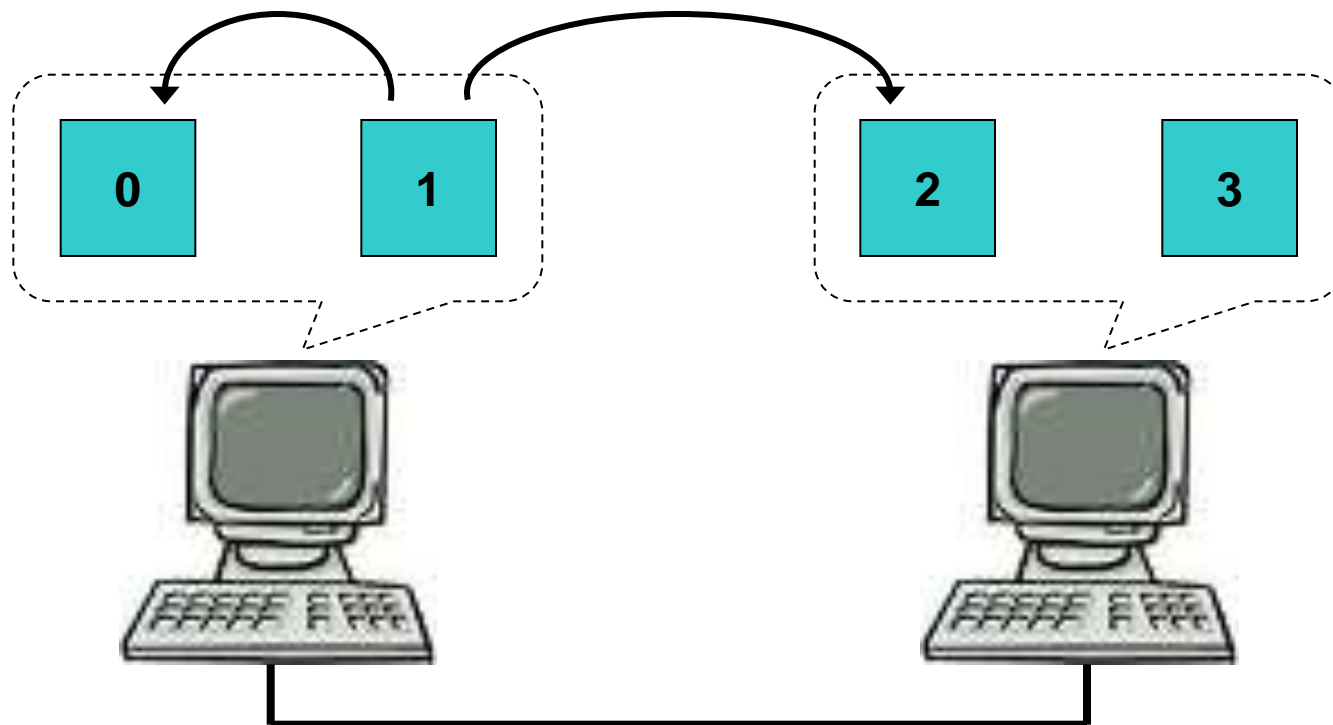
Кластер МВС-100К (Москва)

Кластер НКС-30Т (Новосибирск)

Реализация в MPI передачи сообщений

Копирование
данных в памяти
(быстро)

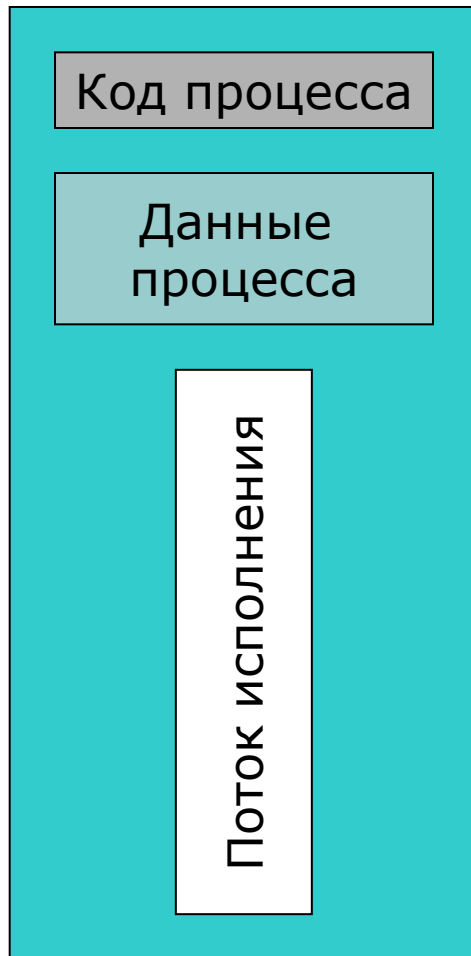
Передача данных
по сети
(медленно)



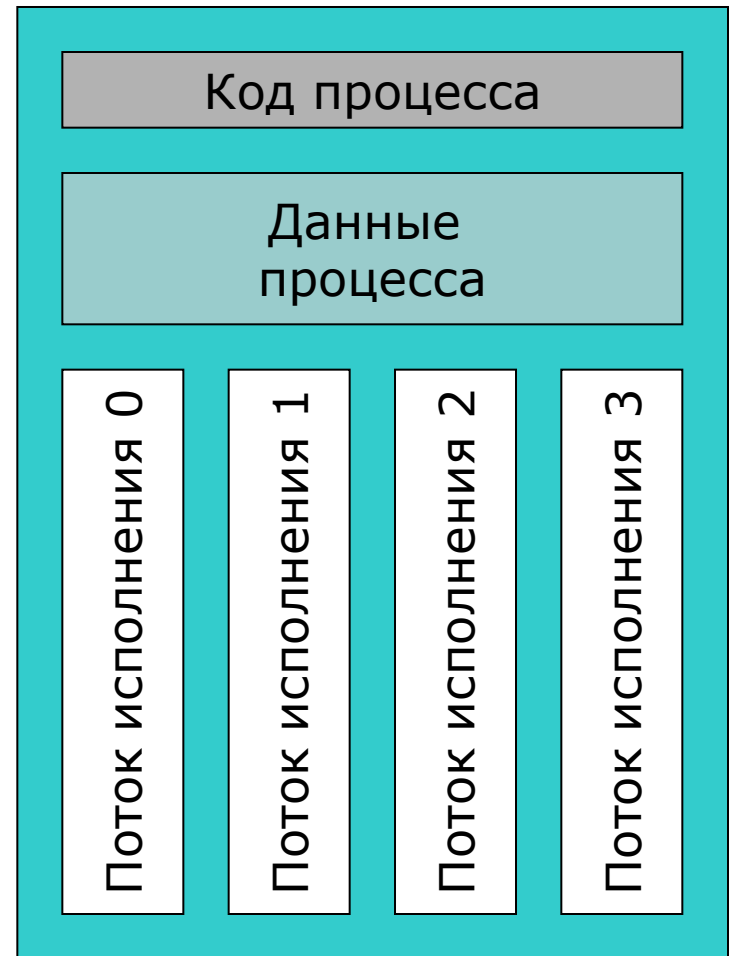
процессы
MPI

Процессы и потоки

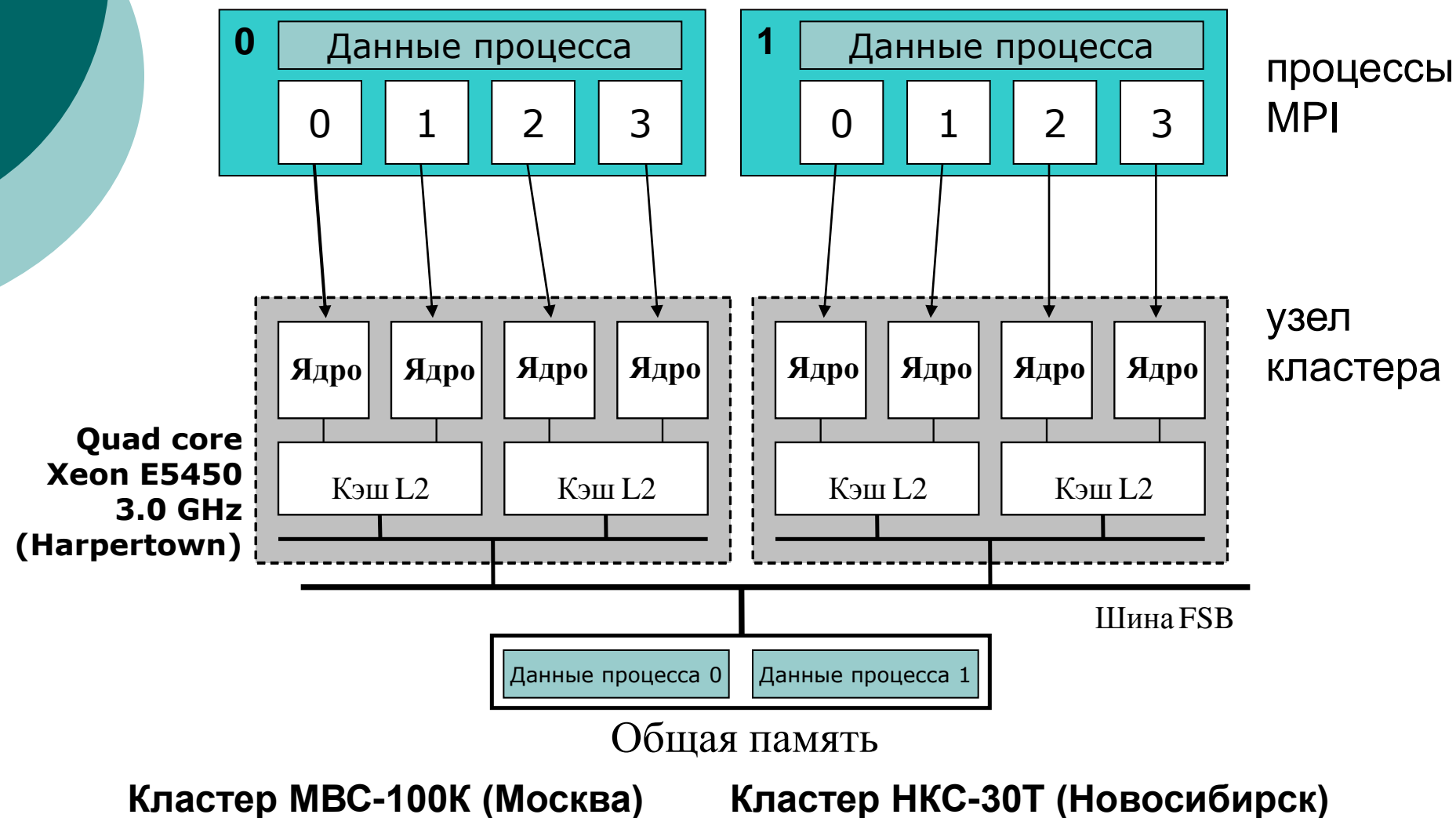
Однопоточный процесс




Многопоточный процесс



Отображение процессов и потоков на ядра в узле кластера





Сравнение процессов MPI и потоков в узле: первый взгляд

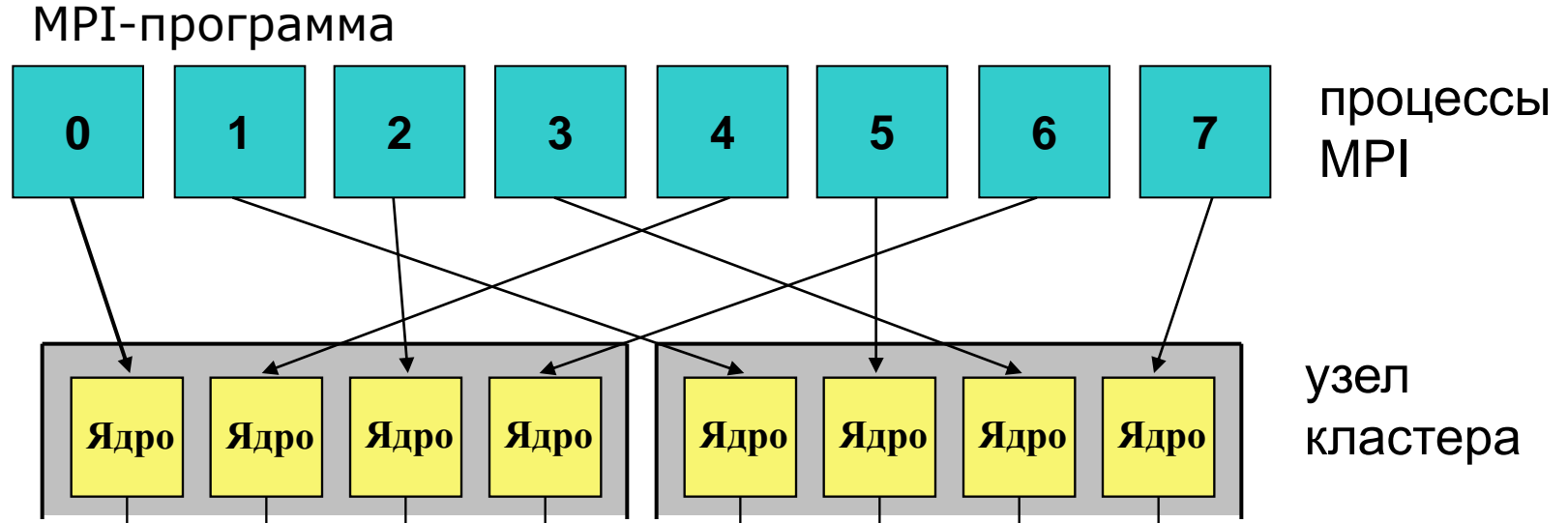
- Процессы MPI
 - Дублирование данных:
 - границы подобластей
 - Затраты на передачу сообщений:
 - Сборка данных
 - Копирование в другой процесс
 - Разборка данных
- Потоки (OpenMP)
 - Затраты на синхронизацию

План

- Введение в архитектуру: теория
- Практические следствия
 - **Привязка процессов к ядрам**
 - Ограничение пропускной способности памяти узла
 - Планирование (shedule) обхода данных несколькими потоками в OpenMP
 - Использование несколькими потоками одной кэш-строки
- Сравнение производительности MPI и MPI+OpenMP
- Выводы

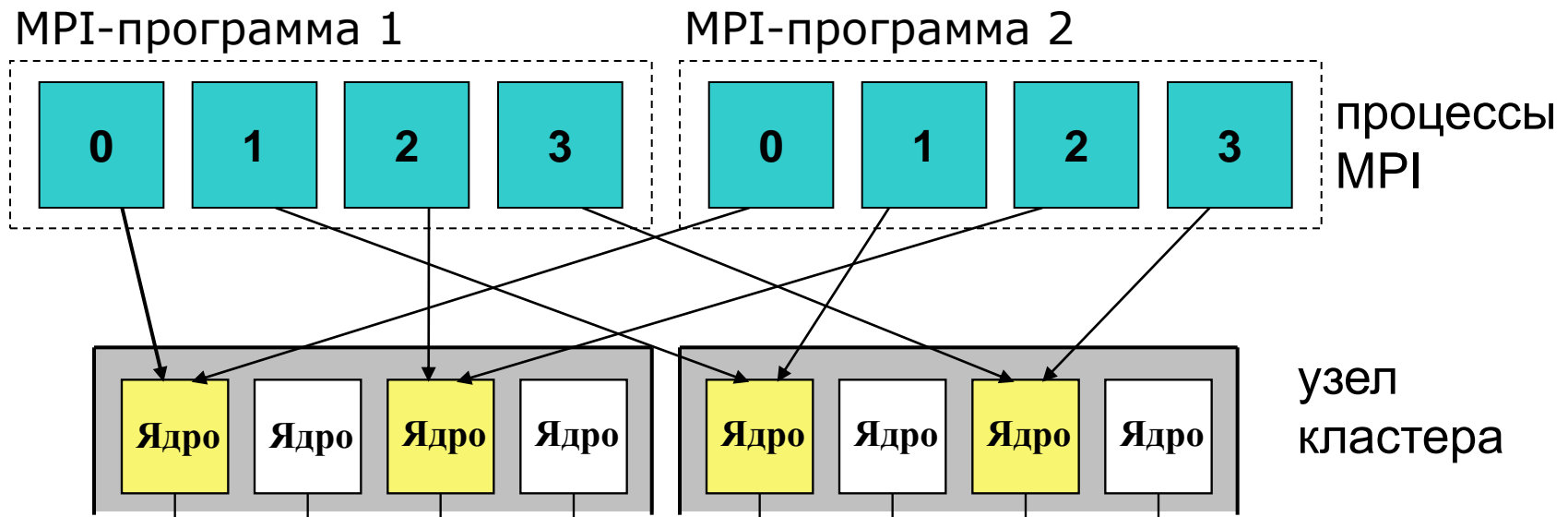
Привязка процессов к ядрам

- По умолчанию процессы MPI жестко привязаны к ядрам:



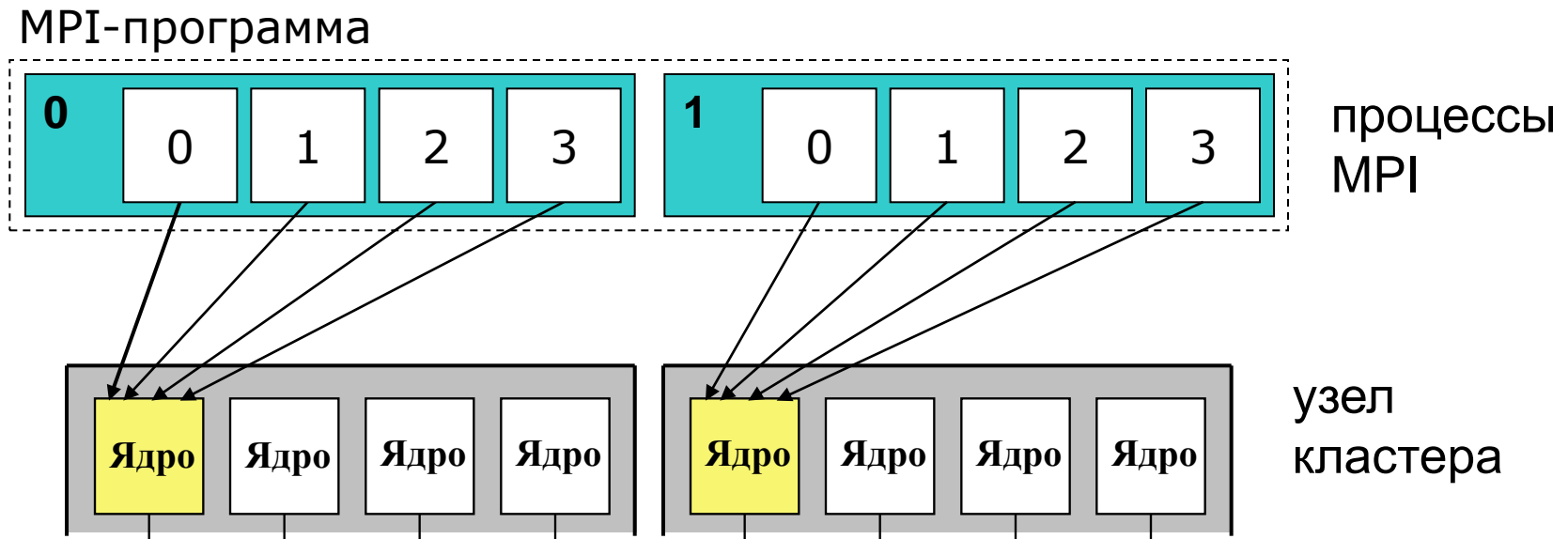
Привязка процессов к ядрам

- Следствие 1:
 - Разные MPI-программы при работе в одном узле занимают одни и те же ядра
 - Возможны конфликты двух задач пользователя и задач разных пользователей кластера. **Один из способов решения – запрашивать узел эксклюзивно.**
 - Незанятые ядра простаивают



Привязка процессов к ядрам

- Следствие 2:
 - Все потоки внутри одного процесса занимают одно ядро
 - Незанятые ядра простаивают

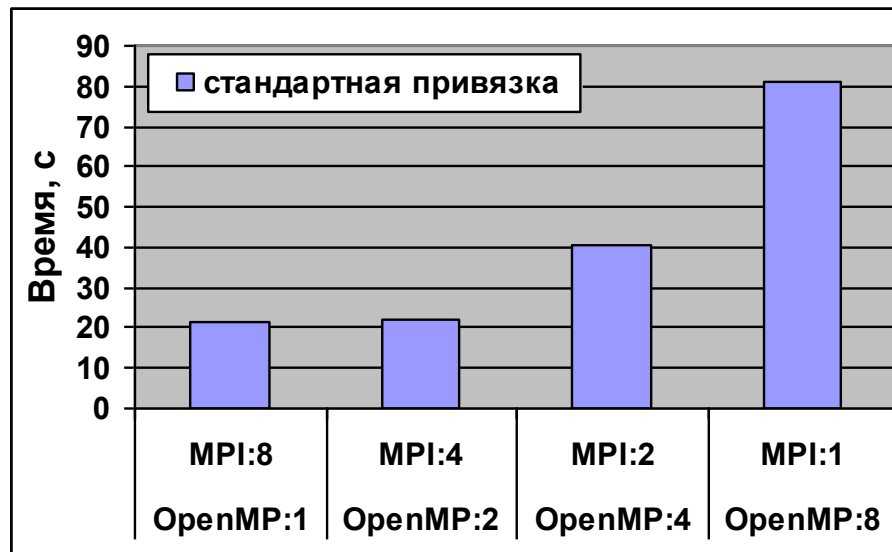


Привязка процессов к ядрам

○ Задача

- Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем памяти: 976 MB
- Декомпозиция пространства моделирования

○ MPI+OpenMP со стандартной привязкой:



На 8 потоках одно ядро загружено в 4 раза больше.

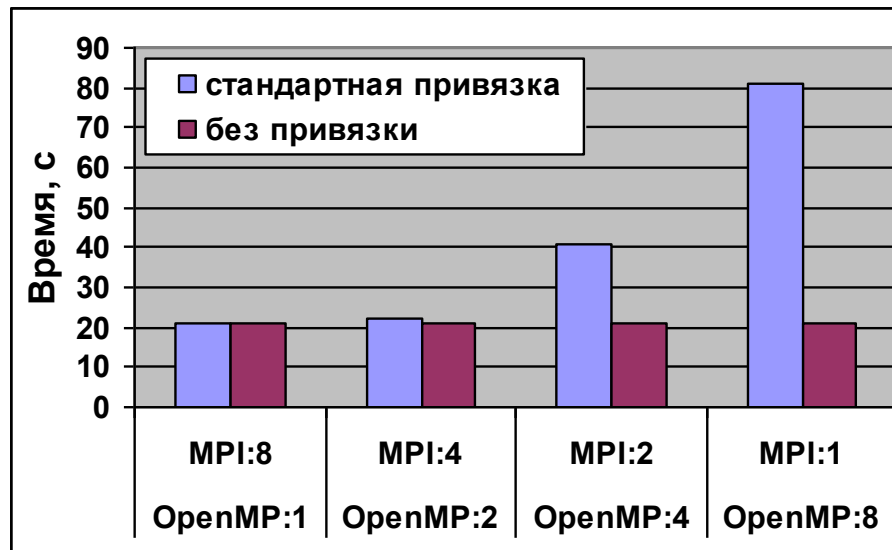
Привязка процессов к ядрам

○ Решение проблемы

- Использовать средства Linux
 - `sched_setaffinity(0,sizeof(mask),mask);`
- Intel MPI (nks-30t) – отключение привязки
 - Указать в скрипте запуска переменную окружения:
`export I_MPI_PIN_DOMAIN=node`
`mpirun ...`
 - или указать параметр `mpirun`:
`mpirun -env I_MPI_PIN_DOMAIN node ...`
- MVAPICH (mvs100k) – отключение привязки
 - Указать параметр `mpirun`:
`mpirun VIADEV_USE_AFFINITY=0 ...`

Привязка процессов к ядрам

- Задача
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем данных: 976 МВ
 - Декомпозиция пространства моделирования
- Влияние привязки процессов MPI к ядрам:



При отключении привязки OpenMP дает ожидаемое ускорение.

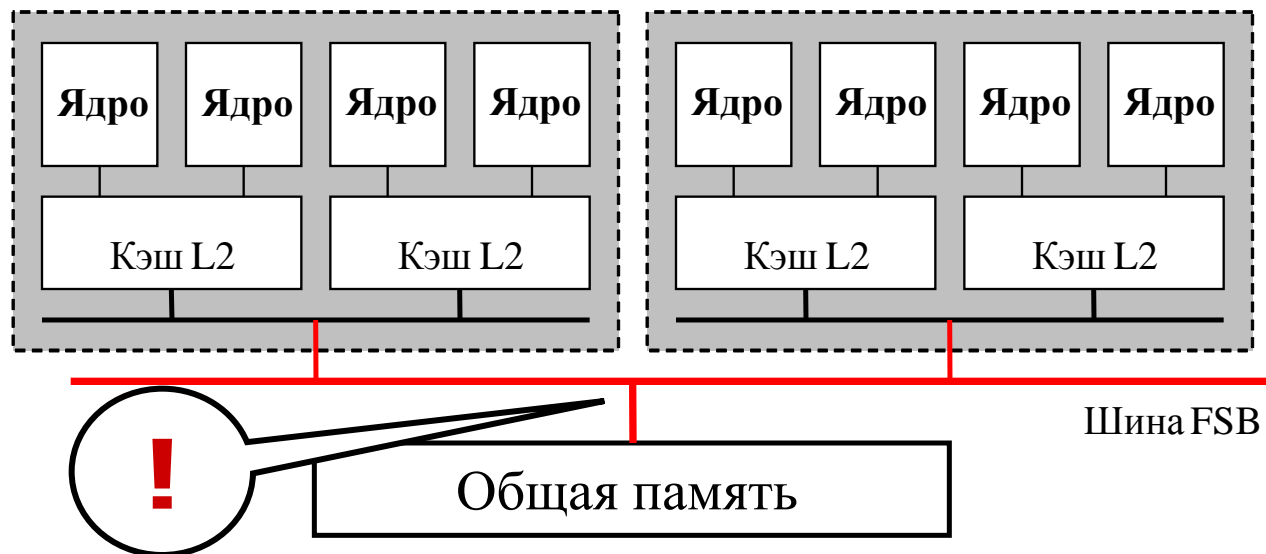
План

- Введение в архитектуру: теория
- Практические следствия
 - Привязка процессов к ядрам
 - **Ограничение пропускной способности памяти узла**
 - Планирование (shedule) обхода данных несколькими потоками в OpenMP
 - Использование несколькими потоками одной кэш-строки
- Сравнение производительности MPI и MPI+OpenMP
- Выводы

Ограничение пропускной способности памяти узла

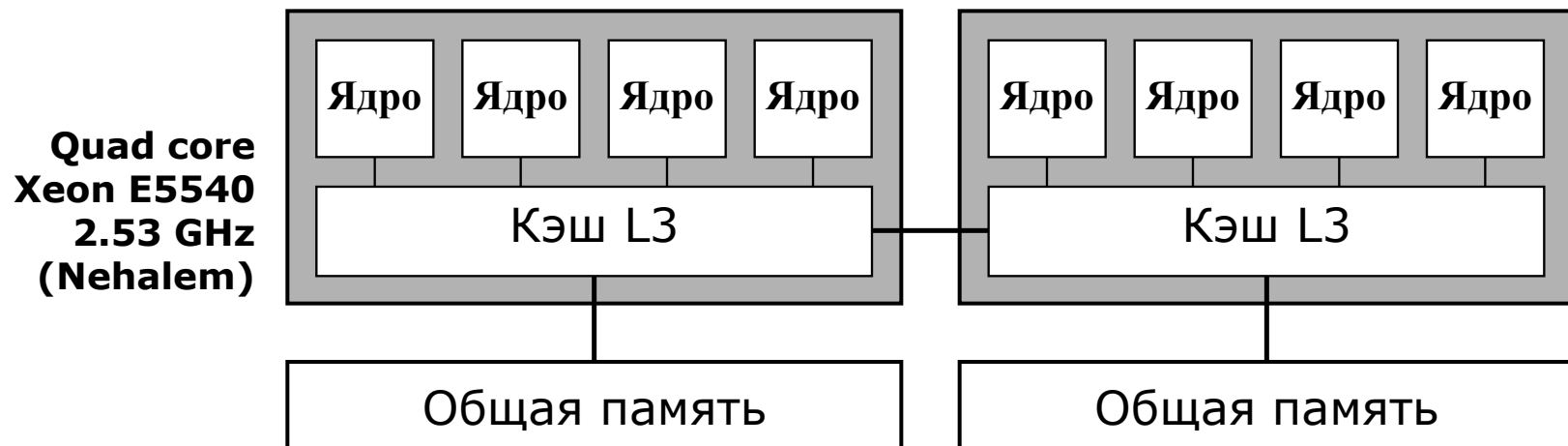
- Доступ к памяти узла – узкое место
 - Шина данных не успевает обрабатывать запросы всех ядер

Quad core
Xeon E5450
3.0 GHz
(Nahptown)



Ограничение пропускной способности памяти узла

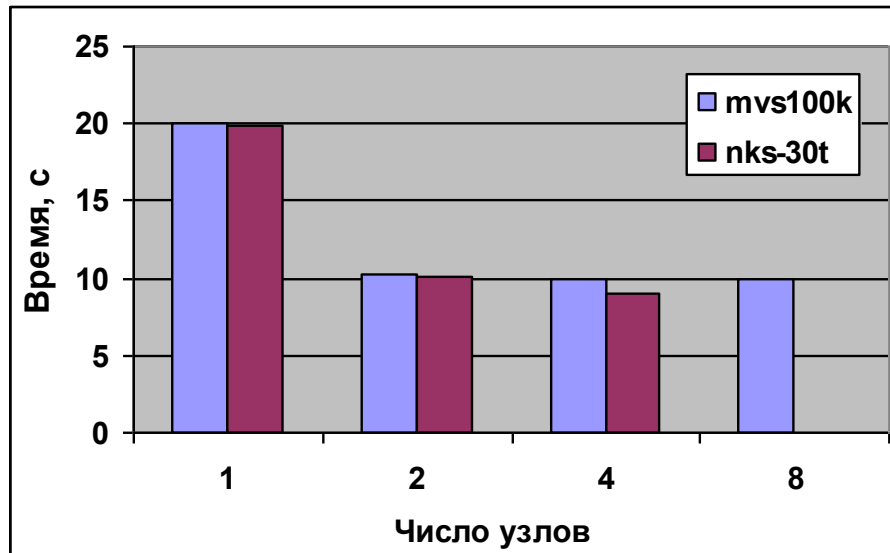
- Попытка преодоления узкого места при доступе к памяти:
 - AMD Opteron
 - Intel Nehalem



Кластер НКС-30Т (Новосибирск) – дополнительные узлы

Ограничение пропускной способности памяти узла

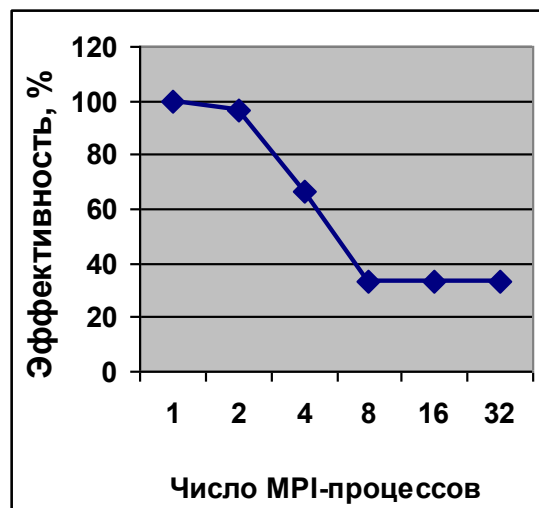
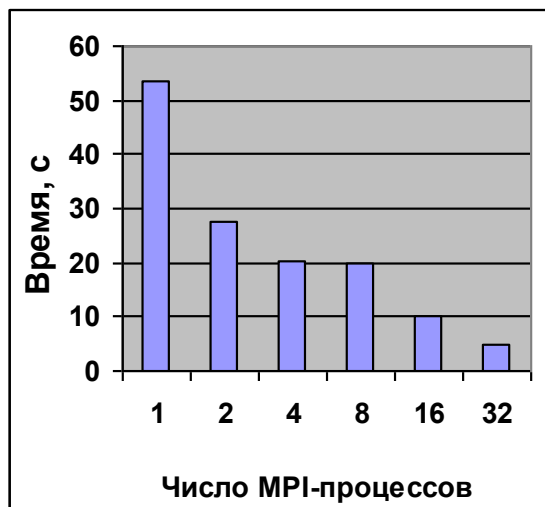
- Задача
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем данных: 976 МВ
 - Декомпозиция пространства моделирования
- 8 процессов MPI на разном числе узлов:



Шина успевает доставить данные только для 4 ядер.

Ограничение пропускной способности памяти узла

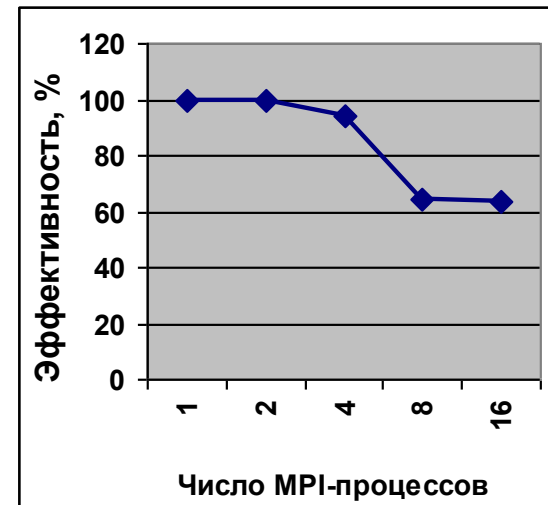
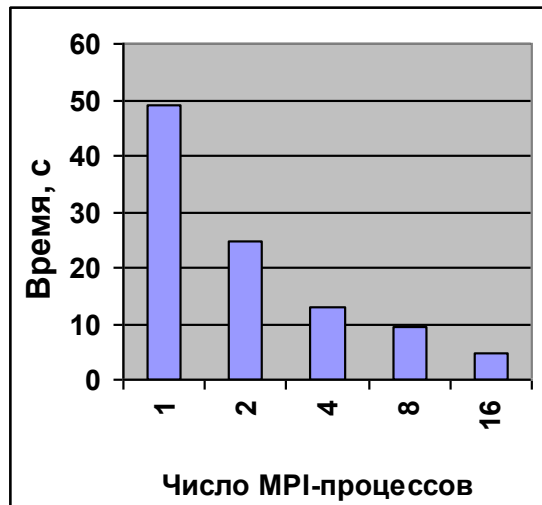
- Задача
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем данных: 976 MB
 - Декомпозиция пространства моделирования
- nks-30t (Новосибирск, ССКЦ)



Видно падение производительности внутри узла.

Ограничение пропускной способности памяти узла

- Задача
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем данных: 976 MB
 - Декомпозиция пространства моделирования
- Nehalem (Новосибирск, ССКЦ)



Видно падение производительности внутри узла.

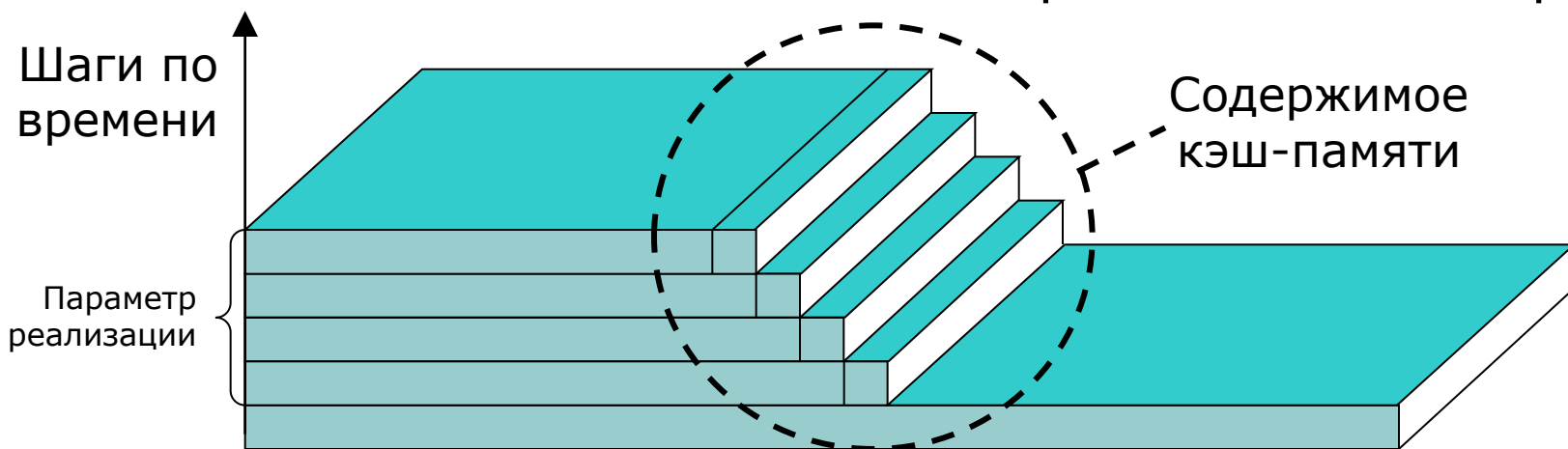
Ограничение пропускной способности памяти узла

- **Решение проблемы**

- **Выполнять больше полезных вычислений за одно чтение данных**

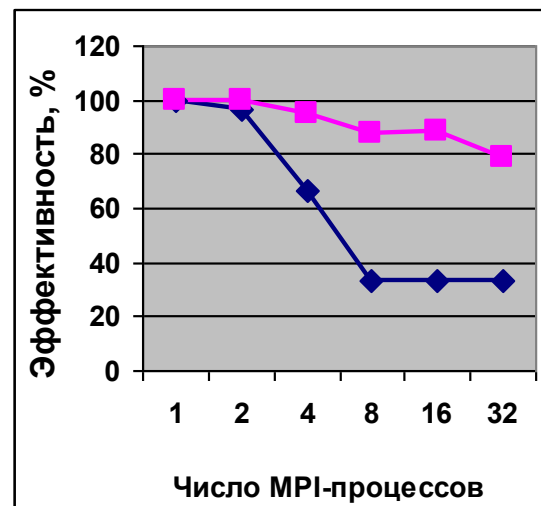
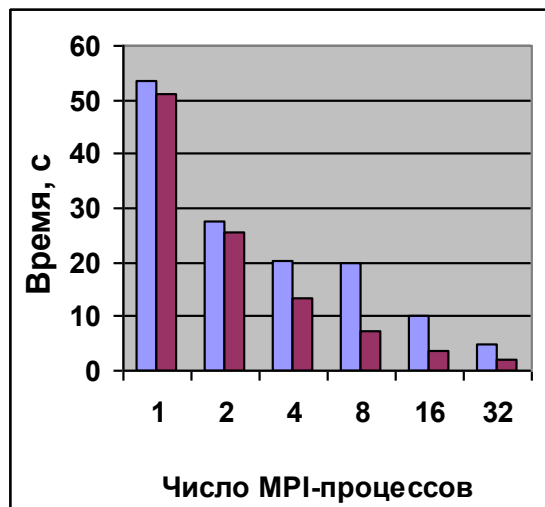
- **Для вычислений по явной схеме**

- **несколько шагов по времени за один проход**



Ограничение пропускной способности памяти узла

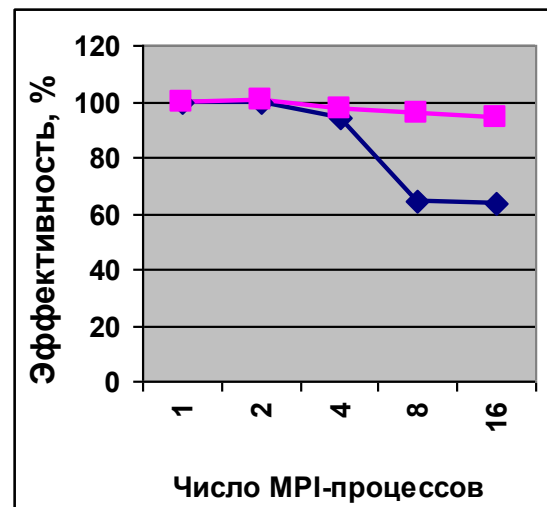
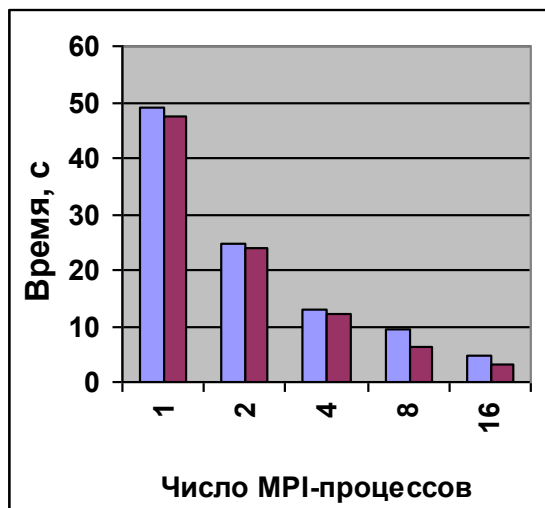
- Задача
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем данных: 976 MB
 - Декомпозиция пространства моделирования
- nks-30t (Новосибирск, ССКЦ)



Преодолели ограничение на пропускную способность памяти.

Ограничение пропускной способности памяти узла

- Задача
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
 - Сетка: 8000×8000
 - Объем данных: 976 MB
 - Декомпозиция пространства моделирования
- Nehalem (Новосибирск, ССКЦ)



Преодолели ограничение на пропускную способность памяти.



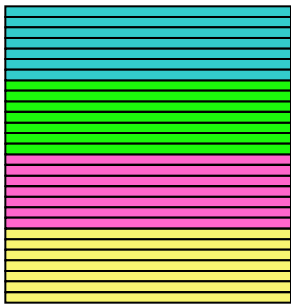
План

- Введение в архитектуру: теория
- Практические следствия
 - Привязка процессов к ядрам
 - Ограничение пропускной способности памяти узла
 - **Планирование (shedule) обхода данных несколькими потоками в OpenMP**
 - Использование несколькими потоками одной кэш-строки
- Сравнение производительности MPI и MPI+OpenMP
- Выводы

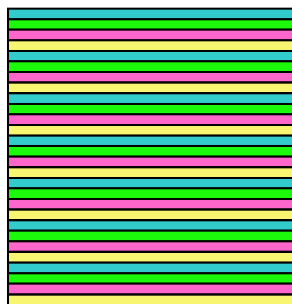
Планирование обхода данных несколькими потоками в OpenMP

- Распределение итераций между потоками в цикле обхода данных
 - Параметры директивы OMP FOR

`schedule(static)`



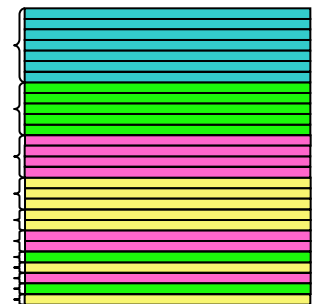
`schedule(static,1)`



`schedule(dynamic)`



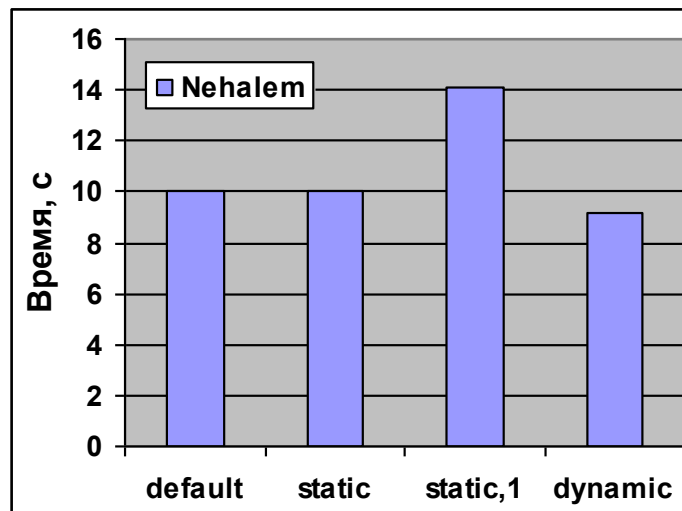
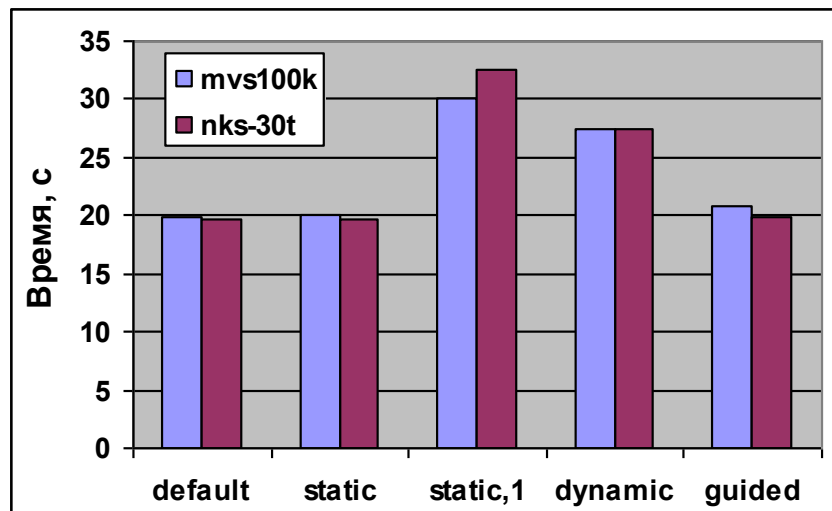
`schedule(guided)`



Пример распределения итераций для 4-х потоков.

Планирование обхода данных несколькими потоками в OpenMP

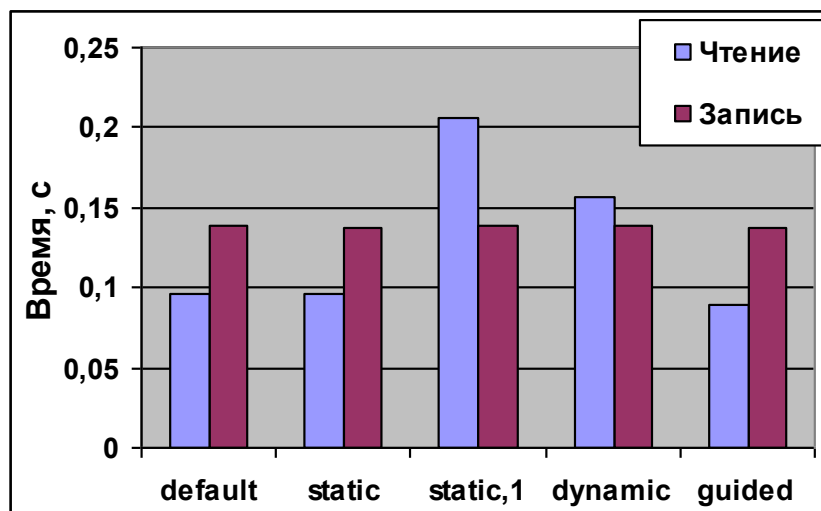
- Задача:
 - Решение 2D волнового уравнения с помощью двухслойной явной схемы
- Распределение итераций между потоками в цикле обхода данных
 - 8 потоков OpenMP в узле
 - Параметры директивы OMP FOR



Вывод: нужно использовать static (или не указывать параметр планирования, что в данной реализации OpenMP эквивалентно static).

Планирование обхода данных несколькими потоками в OpenMP

- Задача:
 - Последовательный обход массива данных, 8 потоков
 - Чтение по 3 строки массива
 - Запись по 1 строке массива
- Распределение итераций между потоками в цикле обхода данных
 - 8 потоков OpenMP в узле
 - Параметры директивы OMP FOR



Основное влияние на производительность оказывает чтение данных по 3 строки.

static выигрывает за счет эффективного использования кэш-памяти.

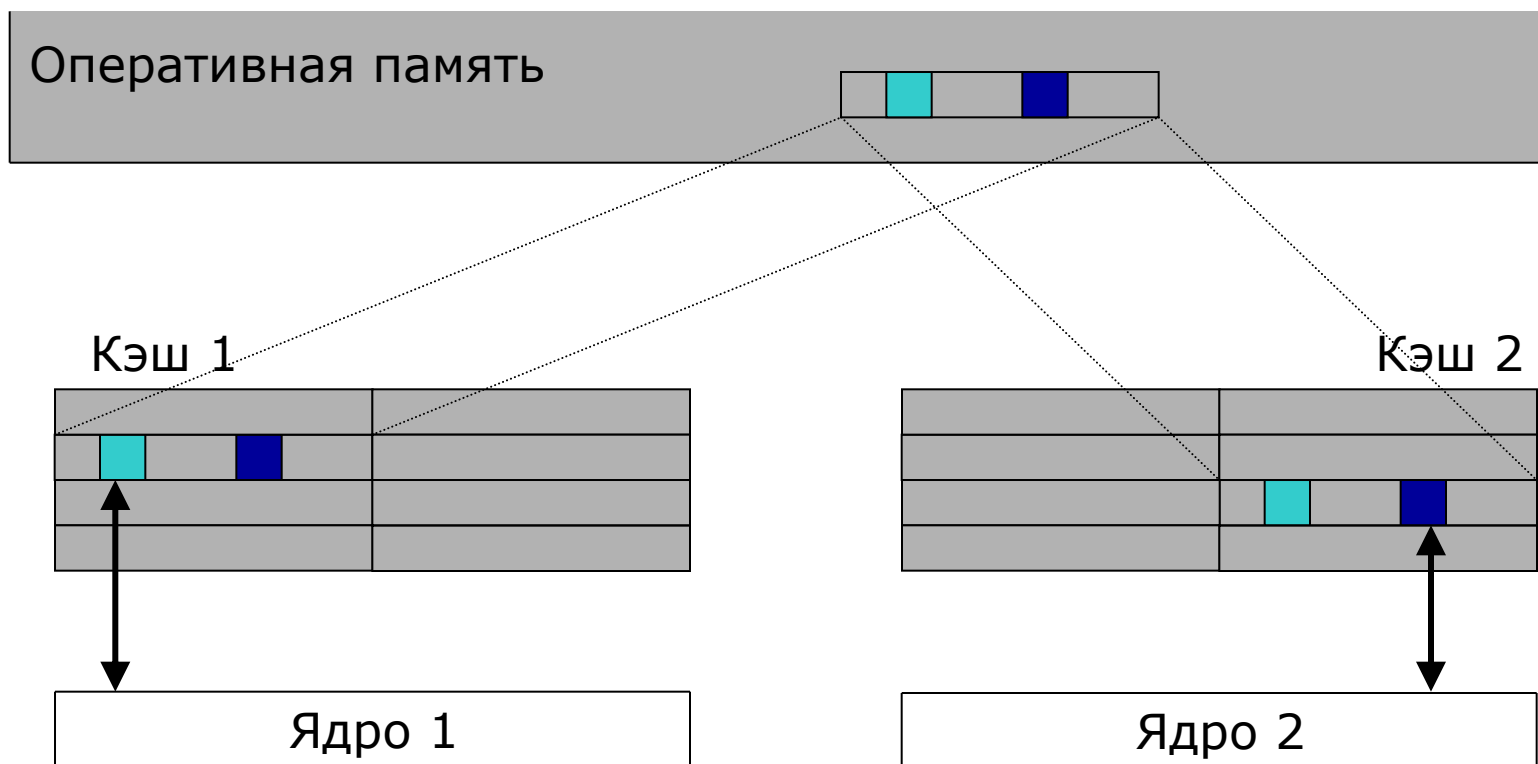


План

- Введение в архитектуру: теория
- Практические следствия
 - Привязка процессов к ядрам
 - Ограничение пропускной способности памяти узла
 - Планирование (shedule) обхода данных несколькими потоками в OpenMP
 - **Использование несколькими потоками одной кэш-строки**
- Сравнение производительности MPI и MPI+OpenMP
- Выводы

Использование несколькими потоками одной кэш-строки

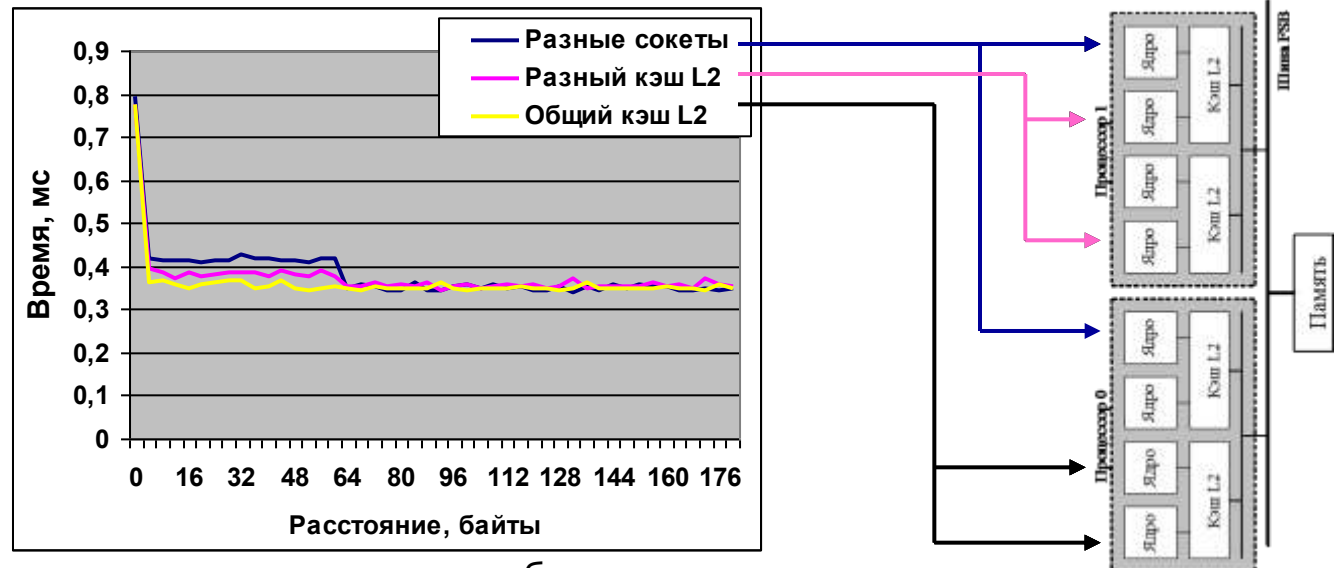
- Многократное изменение одной кэш-строки несколькими потоками приводит к неэффективному использованию кэша.



Использование несколькими потоками одной кэш-строки

○ Задача

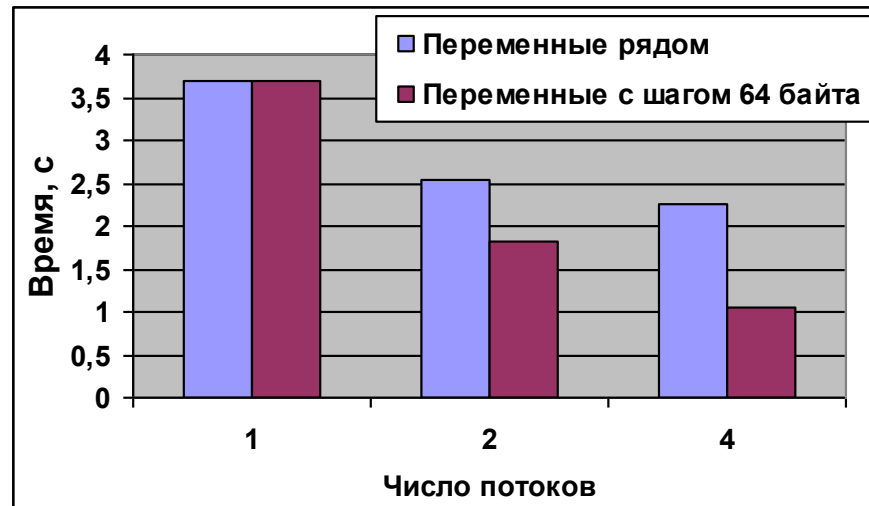
- Многократное изменение двух переменных в общей памяти несколькими потоками
- Зависимость времени от расстояния между переменными:



Медленнее, если потоки над разными кэшами работают с различными переменными, расположенными на расстоянии менее 64 В друг от друга.

Использование несколькими потоками одной кэш-строки

- Задача
 - Моделирование поверхностных химических реакций на катализаторе
 - Консервативный алгоритм моделирования дискретно-событийных систем
 - Счетчики локального времени для каждого потока – часто обновляют свое значение и читают значения соседних
 - Текущее состояние генераторов случайных чисел для каждого потока – часто обновляют свое состояние



Intel Core i7 920
2.67 GHz
Quad code

При разнесении переменных в памяти получаем большее ускорение.



План

- Введение в архитектуру: теория
- Практические следствия
 - Привязка процессов к ядрам
 - Ограничение пропускной способности памяти узла
 - Планирование (shedule) обхода данных несколькими потоками в OpenMP
 - Использование несколькими потоками одной кэш-строки
- **Сравнение производительности MPI и MPI+OpenMP**
- Выводы

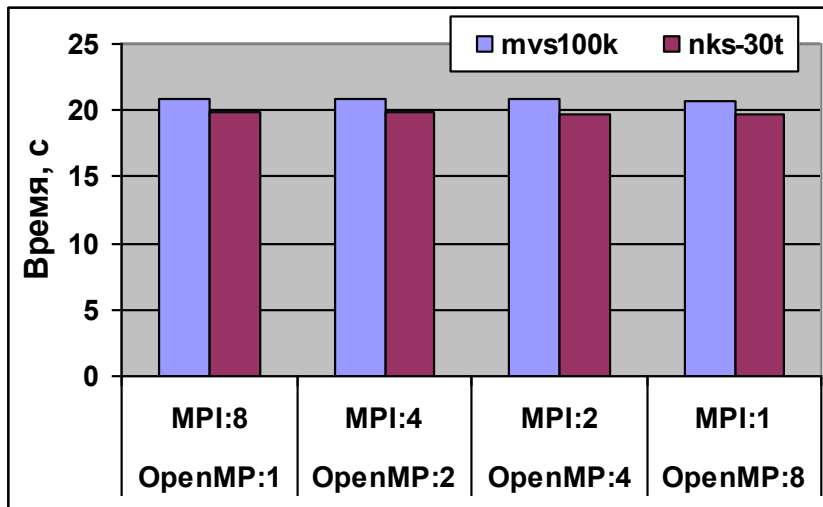
Сравнение MPI и MPI+OpenMP

○ Задача

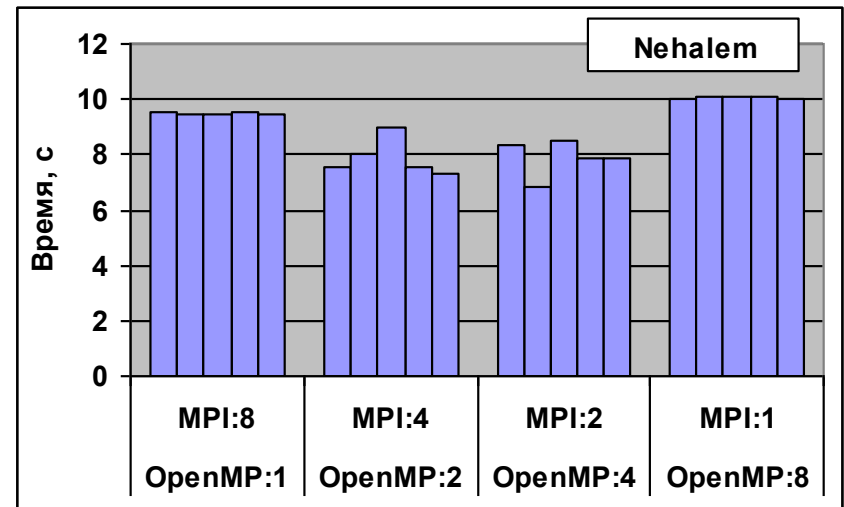
- Решение 2D волнового уравнения с помощью двухслойной явной схемы
- Декомпозиция пространства моделирования

○ MPI+OpenMP: сравнение производительности

- Использование одного узла



В одном узле MPI и MPI+OpenMP дают одинаковые результаты.



На Nehalem 2 и 4 потока на процесс работают быстрее.

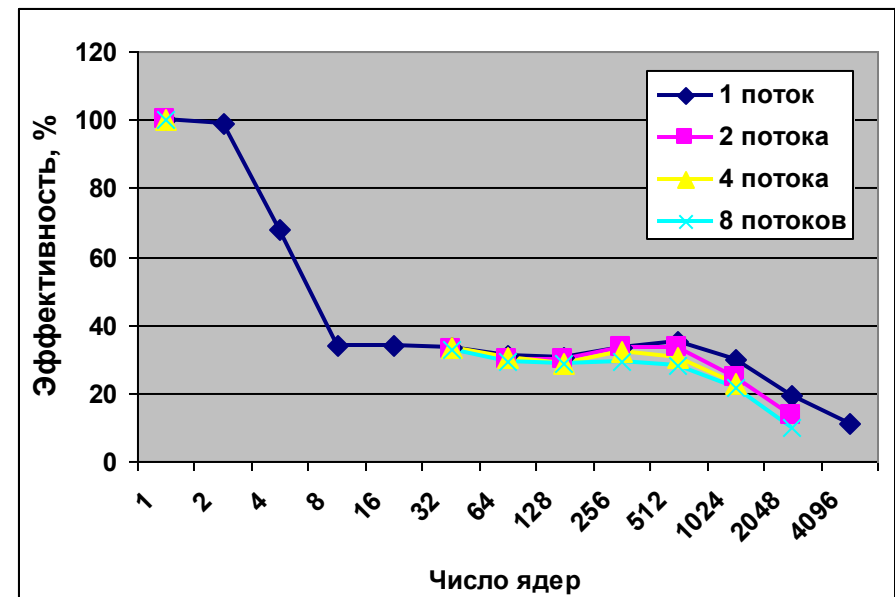
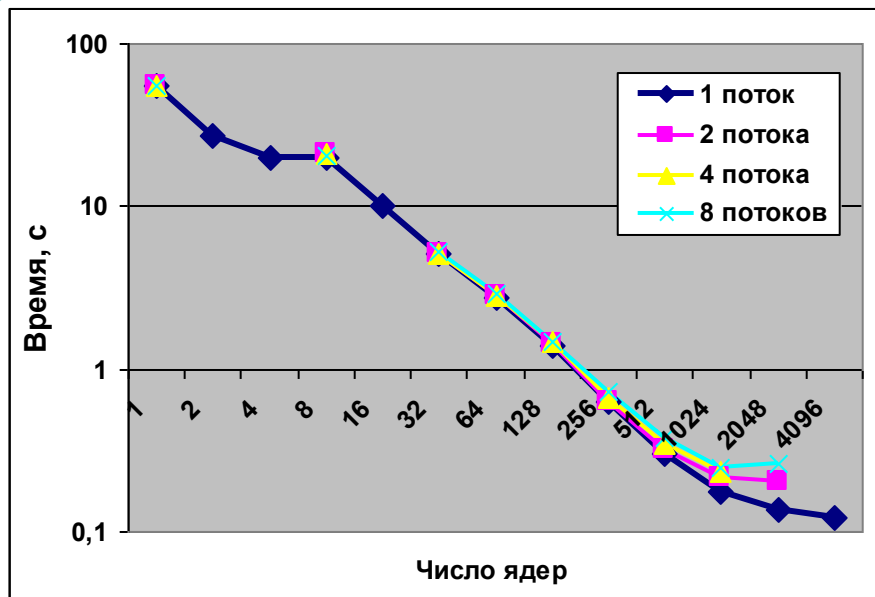
Сравнение MPI и MPI+OpenMP

○ Задача

- Решение 2D волнового уравнения с помощью двухслойной явной схемы
- Декомпозиция пространства моделирования

○ MPI+OpenMP: сравнение производительности

mvsv100k



На этой задаче MPI лучше, чем MPI+OpenMP.

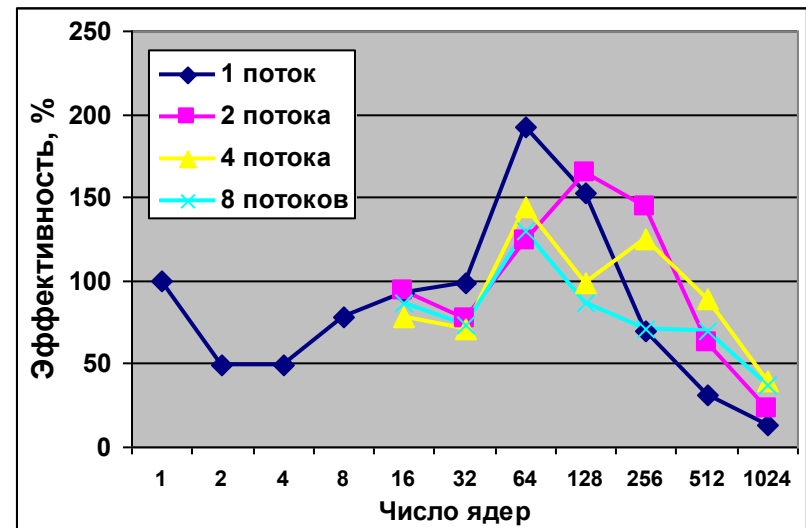
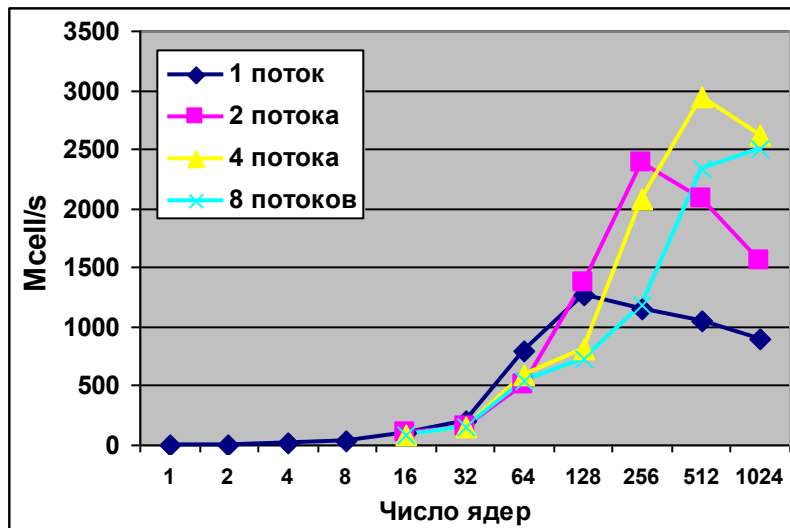
Сравнение MPI и MPI+OpenMP

○ Задача

- Моделирование поверхностных химических реакций на катализаторе
- Блочно-синхронный клеточный автомат
 - Сетка: 4000×4000
 - Объем данных: 30.5 МВ
- Декомпозиция пространства моделирования

○ MPI+OpenMP: сравнение производительности

mvsl00k



Здесь OpenMP позволяет повысить масштабируемость задачи с ростом числа используемых узлов.



Выводы

- Как в MPI, так и в OpenMP следует учитывать архитектурные ограничения.
- Нет универсального способа сделать MPI+OpenMP лучше, чем MPI.
- Но в классе задач, где коммуникации преобладают над вычислениями, использование OpenMP будет иметь смысл.



Спасибо за внимание



Дополнительные слайды

Постановка задачи

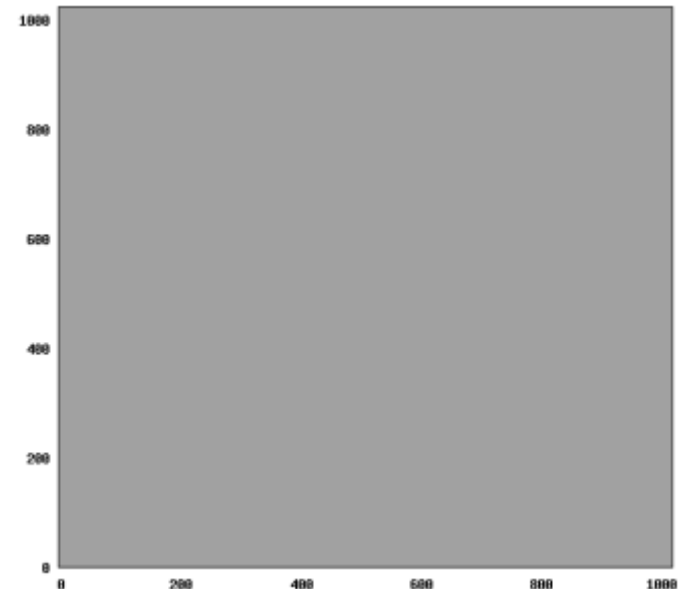
- Волновое уравнение:

$$\frac{\partial^2 u}{\partial t^2} = \nabla(p \nabla u) + f$$

$$\Omega = \{0 \leq x \leq a, 0 \leq y \leq b\}$$

$$u|_{t=0} = \frac{\partial u}{\partial t} \Big|_{t=0} = 0$$

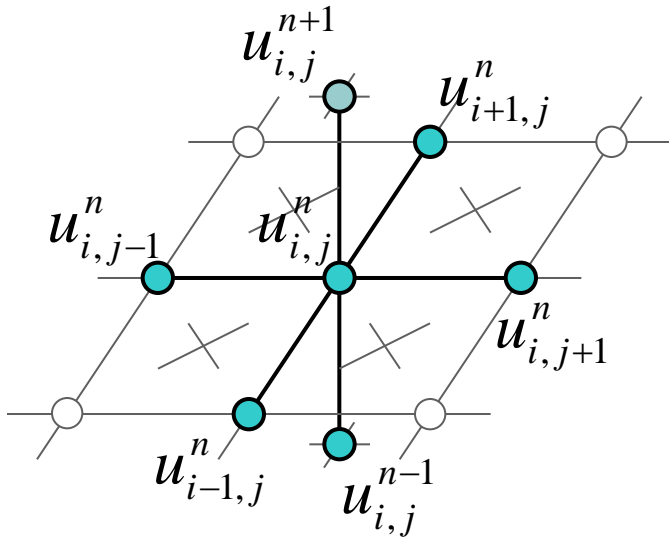
$$u|_{x=0} = u|_{x=a} = u|_{y=0} = u|_{y=b} = 0$$



Постановка задачи

○ Явная схема:

$$u_{i,j}^{n+1} = 2u_{i,j}^n - u_{i,j}^{n-1} + f_{i,j}\tau^2 + \frac{\tau^2}{h_x h_y} \sum_{k=1}^4 \gamma_k$$



$$\gamma_1 = \frac{h_y}{2h_x} (u_{i+1,j}^n - u_{i,j}^n) \left(p_{i+1/2,j-1/2} + p_{i+1/2,j+1/2} \right)$$

$$\gamma_2 = \frac{h_x}{2h_y} (u_{i-1,j}^n - u_{i,j}^n) \left(p_{i-1/2,j-1/2} + p_{i-1/2,j+1/2} \right)$$

$$\gamma_3 = \frac{h_y}{2h_x} (u_{i,j+1}^n - u_{i,j}^n) \left(p_{i-1/2,j+1/2} + p_{i+1/2,j+1/2} \right)$$

$$\gamma_4 = \frac{h_x}{2h_y} (u_{i,j-1}^n - u_{i,j}^n) \left(p_{i-1/2,j-1/2} + p_{i+1/2,j-1/2} \right)$$

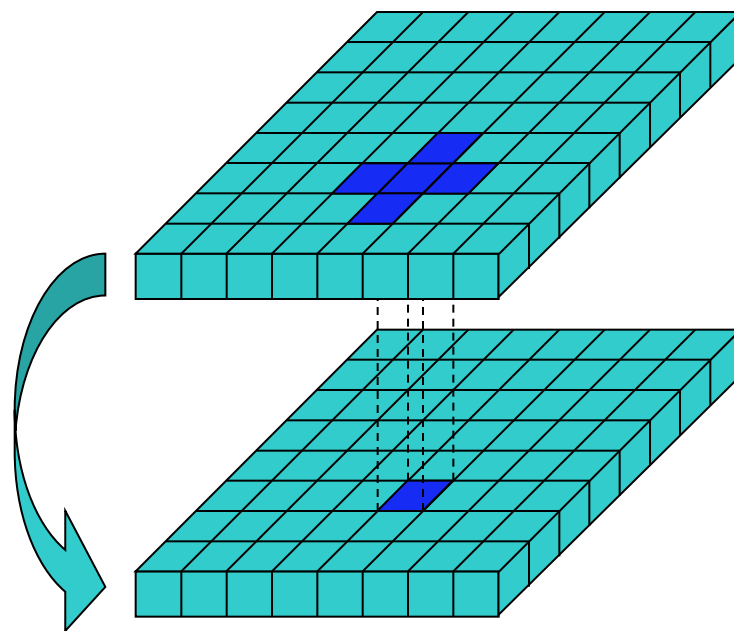
Параметры задачи

Данные задачи

- Размер сетки: 8000×8000
- Тип данных: double (8 байт)
- 2 массива

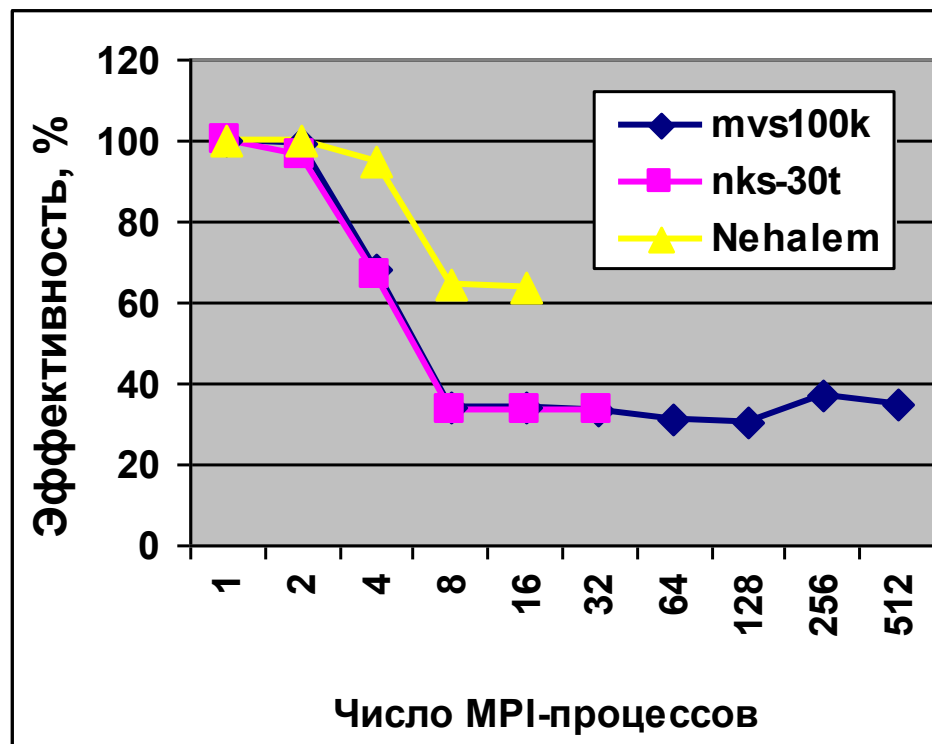
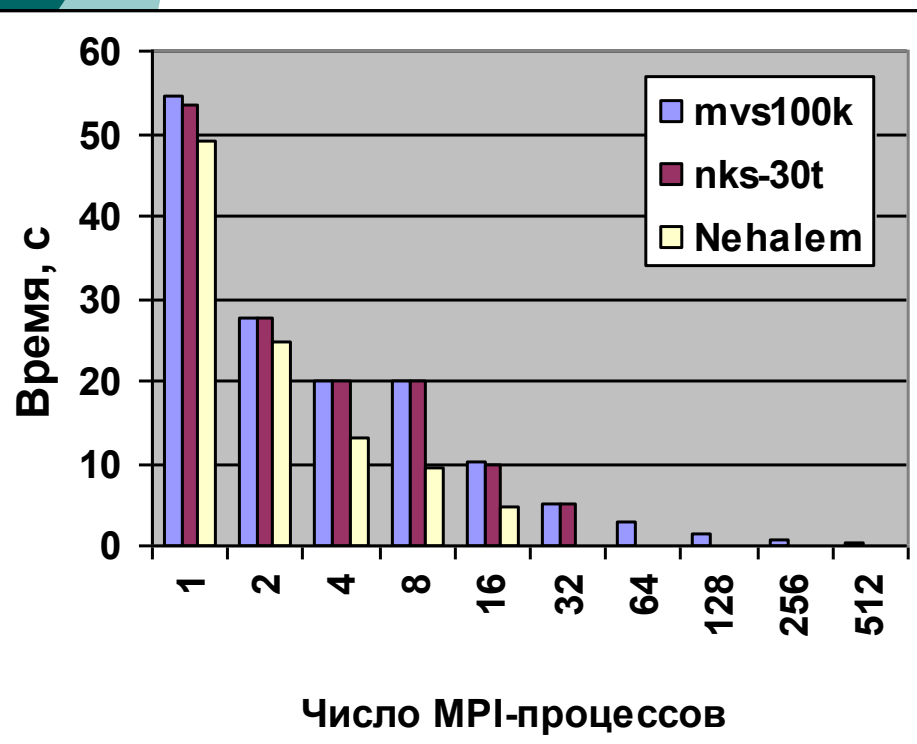
Всего: 976 Мбайт

- Число шагов: 100



Ограничение пропускной способности памяти узла

- Программа без оптимизации



Ограничение пропускной способности памяти узла

- Программа с оптимизацией

