Семинар 12
Стандарт MPI (часть 5)

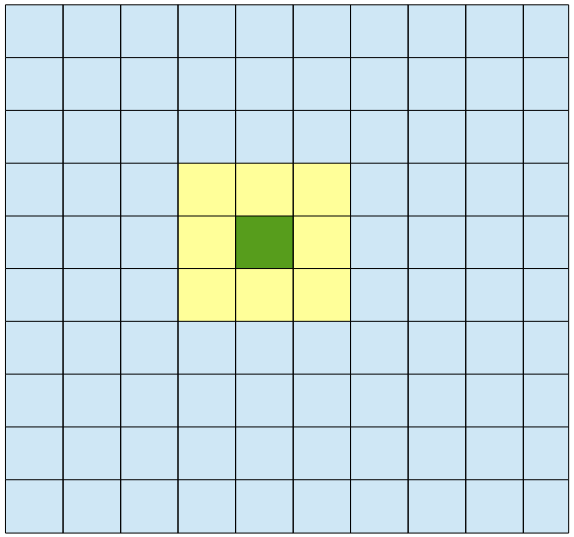
Михаил Курносов

E-mail: mkurnosov@gmail.com
WWW: www.mkurnosov.net

Цикл семинаров «Основы параллельного программирования»
Институт физики полупроводников им. А. В. Ржанова СО РАН
Новосибирск, 2015

Conway's Game of Life

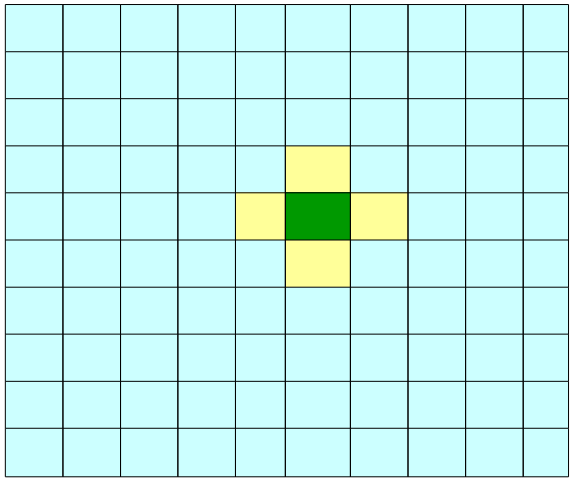
- **Игра «Жизнь»** (Game of Life, Дж. Конвей, 1970)
- Игровое поле — размеченная на клетки плоскость
- Каждая клетка может находиться в двух состояниях: «живая» и «мёртвая», и имеет восемь соседей
- Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего:
  - в мертвой клетке, рядом с которой три живые клетки, зарождается жизнь
  - если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить; в противном случае (соседей < 2 или > 3) клетка умирает



http://en.wikipedia.org/wiki/Conway's_Game_of_Life

Модификация правил игры

- У каждой клетки 4 соседа
- Клетки за границами поля мертвы
  (периодические граничные условия отсутствуют)
- Каждое следующее поколение рассчитывается
  на основе предыдущего:
  - в мертвой клетке, рядом с которой 1 или 2 живые клетки, зарождается жизнь
  - если у живой клетки есть 1 или 2 живые соседки, то эта клетка продолжает жить; в противном случае (соседей < 1 или > 2) клетка умирает



Последовательная версия

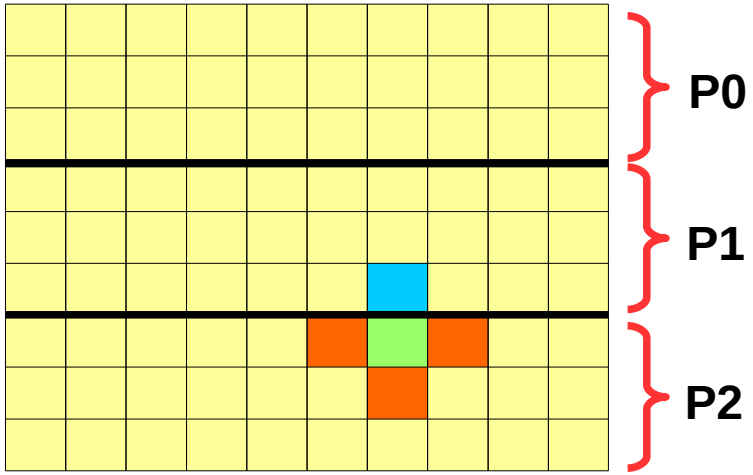
```
Create initial generation (random) - grid[rows, cols]

for t = 1 to NTICKS do

    for i = 1 to rows do
        for j = 1 to cols do
            count = grid[IND(i - 1, j)] + grid[IND(i + 1, j)] +
                    grid[IND(i, j – 1)] + grid[IND(i, j + 1)]
            newgrid[i, j] = (count == 1 || count == 2) ? 1 : 0;
        end for
    end for

    swap_pointers(newgrid, grid)
end for

Save grid
```

Параллельная версия

- Входные параметры: rows, cols, ticks
- Схема декомпозиции игрового поля (domain decomposition):
  - 1D (по строкам, по столбцам)
  - 2D (подмассивы)
- Распределение частей игрового поля по процессам
- Вычисления над локальными данными
- Сохранение игрового поля

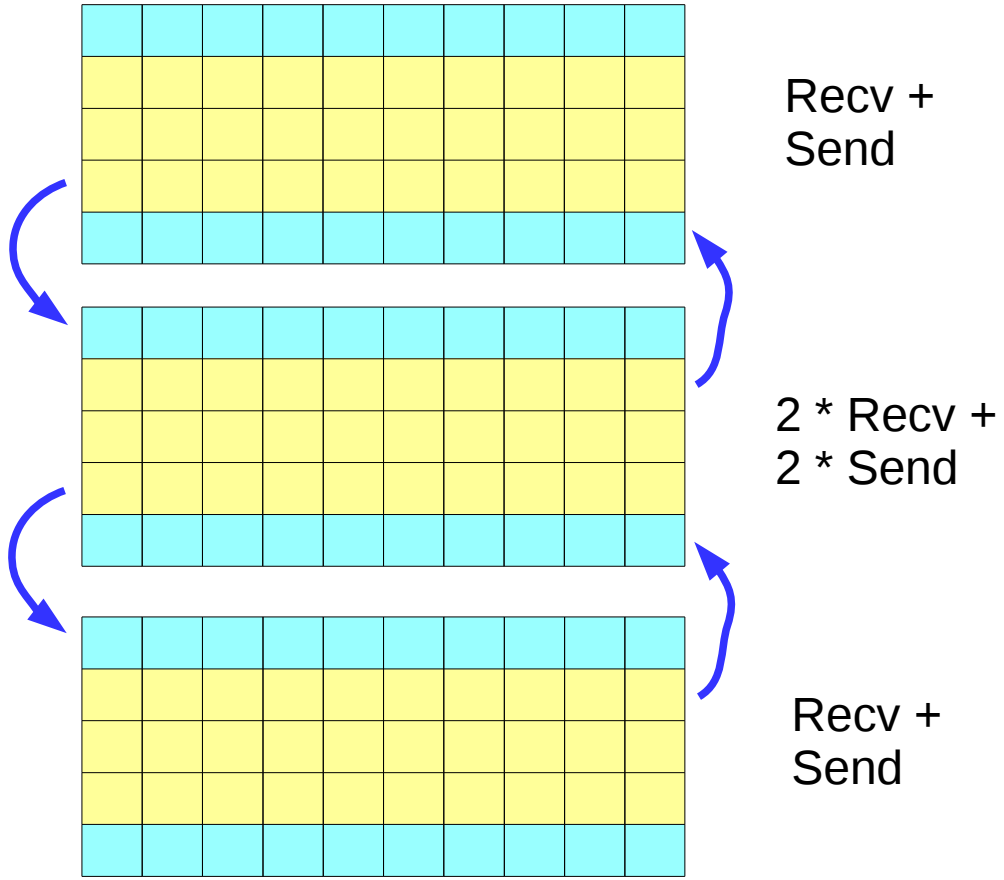
Одномерная декомпозиция по строкам (1D)

grid[rows, cols]



Для вычисления состояния пограничных ячеек нужны значения ячеек соседних процессов

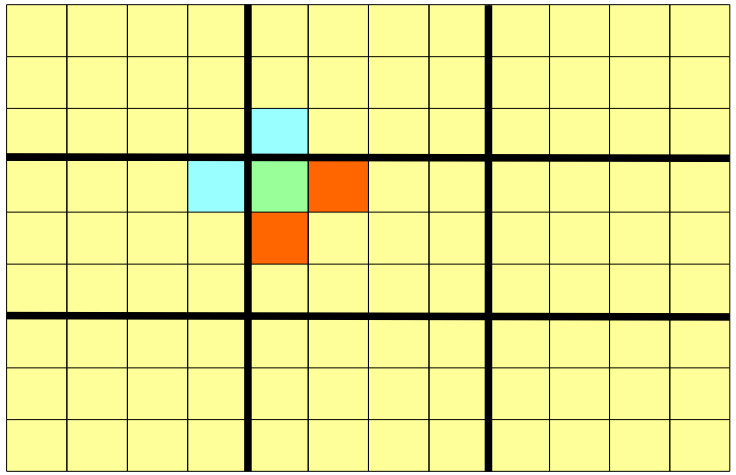
local_grid[rows/p + 2, cols]



Каждый процесс обменивается пограничными ячейками (halo) со смежными процессами

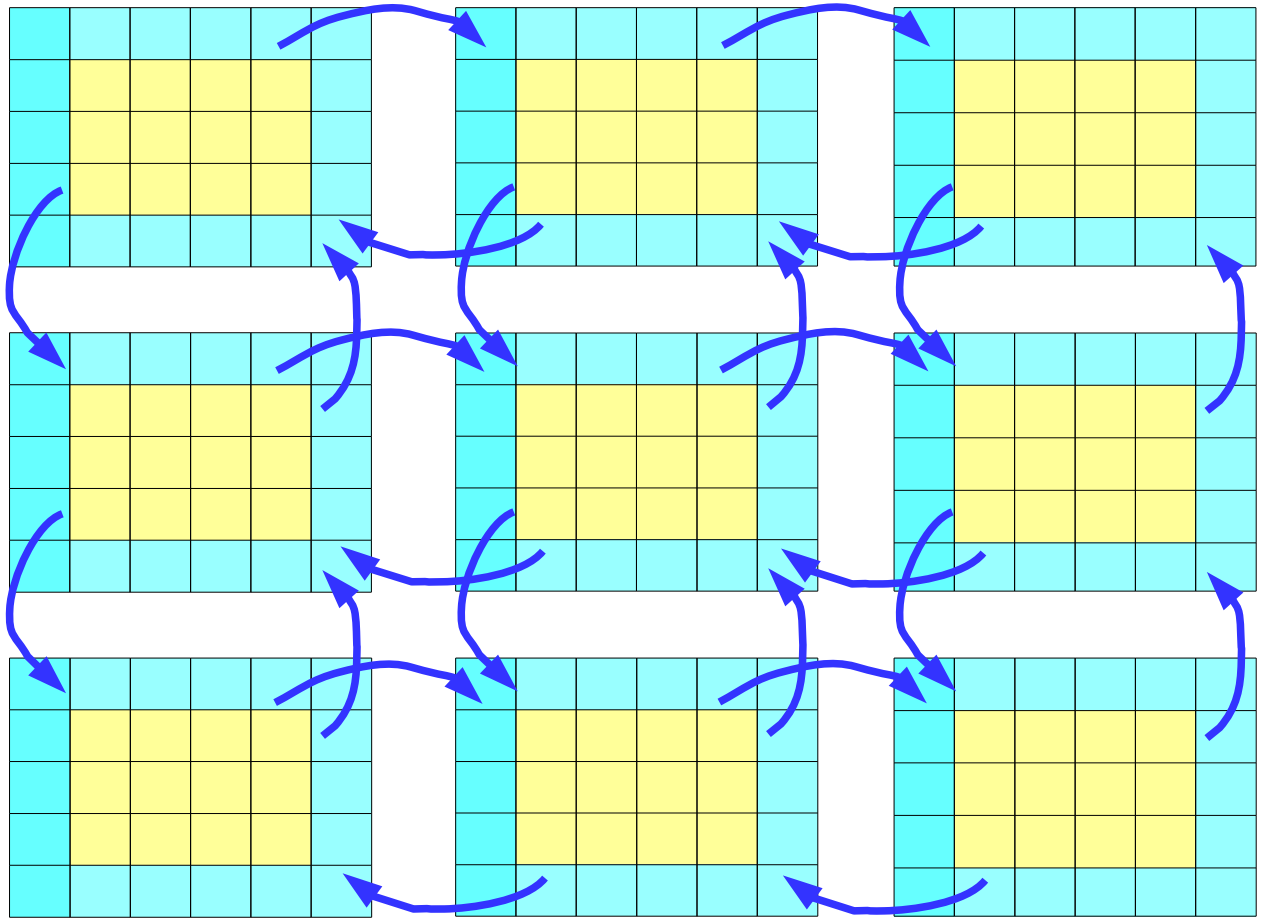
Двумерная декомпозиция (2D)

grid[rows, cols]



Сетка из p процессов
p = px * py

local_grid[rows/py + 2, cols/px + 2]



Декомпозиция 1D vs 2D

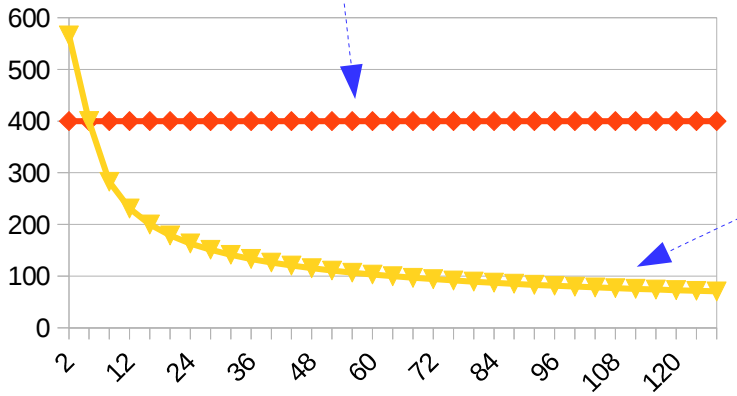
Время на обмены с соседними процессами (внутренние процессы, один шаг игры)

$$T_{Send}(m) = T_{Recv}(m) = \alpha + \beta m$$

1D decomposition

$$T = 2T_{Send}(c) + 2T_{Recv}(c)$$

$$T = 4(\alpha + \beta c)$$



2D decomposition

$$T = 2T_{Send}(\frac{c}{\sqrt{p}}) + 2T_{Recv}(\frac{c}{\sqrt{p}}) + 2T_{Send}(\frac{r}{\sqrt{p}}) + 2T_{Recv}(\frac{r}{\sqrt{p}})$$

$$T = 4\left(\alpha + \beta\frac{c}{\sqrt{p}}\right) + 4\left(\alpha + \beta\frac{r}{\sqrt{p}}\right)$$ // Send = recv

$$T = 8(\alpha + \beta\frac{c}{\sqrt{p}})$$ // *c = r = n*

Вычисления над локальной частью игрового поля

- Принимаем от соседних процессов их граничные строки и/или столбцы (Isend/Irecv, MPI one-sided, MPI Neigh. collectives)
- Отправляем соседним процессам свои пограничные области
- Вычисляем состояния клеток локальной области

Распределение частей игрового поля по процессам

- Корневой процесс формирует в своей памяти игровое поле и рассылает его части процессам (требуется порядка $O(r * c)$ ячеек памяти!!!)
- Корневой процесс формирует данные для каждого процесса и отправляет их (требуется порядка $O((r * c) / p)$ ячеек памяти, все поле в памяти не хранится)
- Каждый процесс сам создает и инициализирует свою часть игрового поля
- Каждый процесс загружает часть игрового поля из файла (MPI I/O)

Сохранение игрового поля

- Корневой процесс получает от процессов результаты и формирует в своей памяти игровое поле (требуется порядка $O(r * c)$ ячеек памяти!!!)
- Корневой процесс получает результаты от каждого процесса и дописывает их в файл (требуется порядка $O((r * c) / p)$ ячеек памяти, все поле в памяти не хранится)
- Каждый процесс сам дописывает свою область в результирующий файл
- Каждый процесс сохраняет результат в файл средствами MPI I/O

Examples