Децентрализованная “выживающая” схема

Тестовый пример: heat2d

Библиотека отказоустойчивости:UFLM

Параллельная программа решения двумерного уравнения теплопроводности

- Решение двумерного уравнения теплопроводности, реализовано методом *последовательных итераций Якоби*
- Метод распараллелен в стандарте MPI путем *двумерной декомпозиции расчетной области*
- Каждому процессу назначена *подматрица и теневые ячейки* по всем четырем направлениям, для расчета значений на границах выделенной области
- Для обеспечения восстановления вычислительного процесса в программу добавлен функционал *формирования контрольных точек*

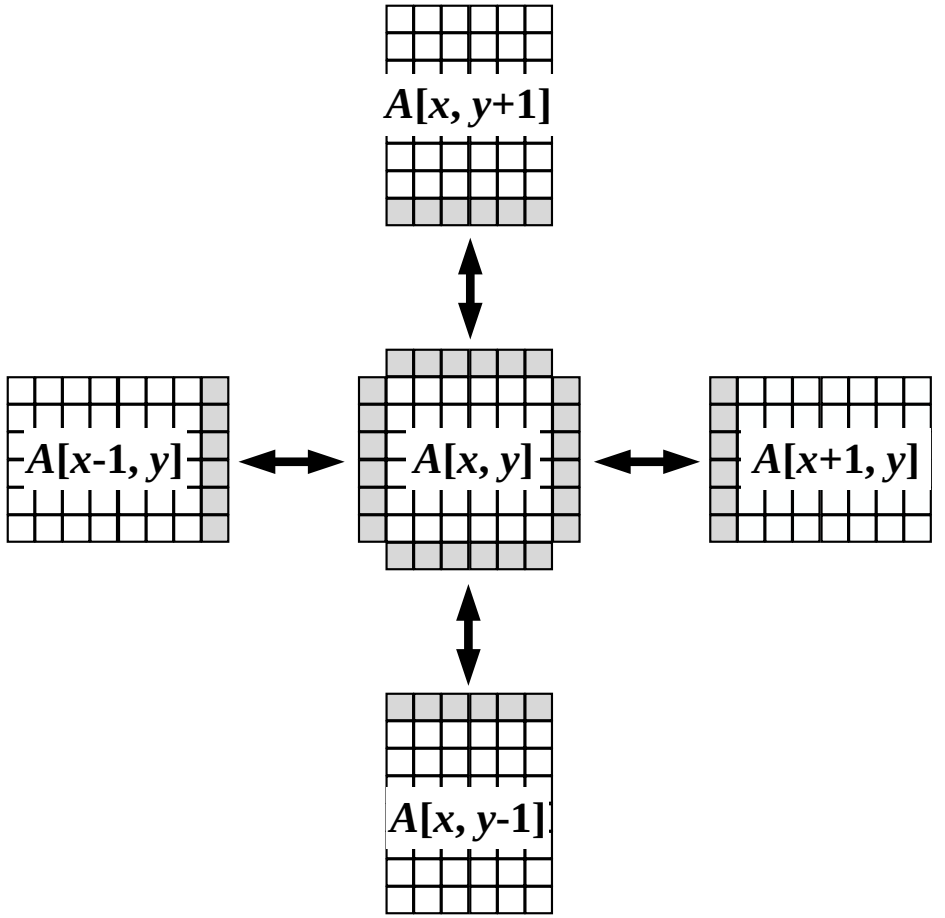


Схема обмена теневыми ячейками подматрицы $A[x, y]$

Параллельная программа решения двумерного уравнения теплопроводности

```
while true do
    for 1 to ny
        for 1 to nx
            // Расчет значений внутренних точек
        end for
    end for

    for 1 to ny
        for 1 to nx
            maxdiff = fmax(maxdiff,...)
        end for
    end for

    if maxdiff < EPS then
        // Проверка условия на выход
        break
    end if

    // Обмен теневыми ячейками с соседями
    // Создание контрольной точки
end while
```

Главный цикл вычисления значений в подматрице $A[x, y]$ и обмена теневыми ячейками

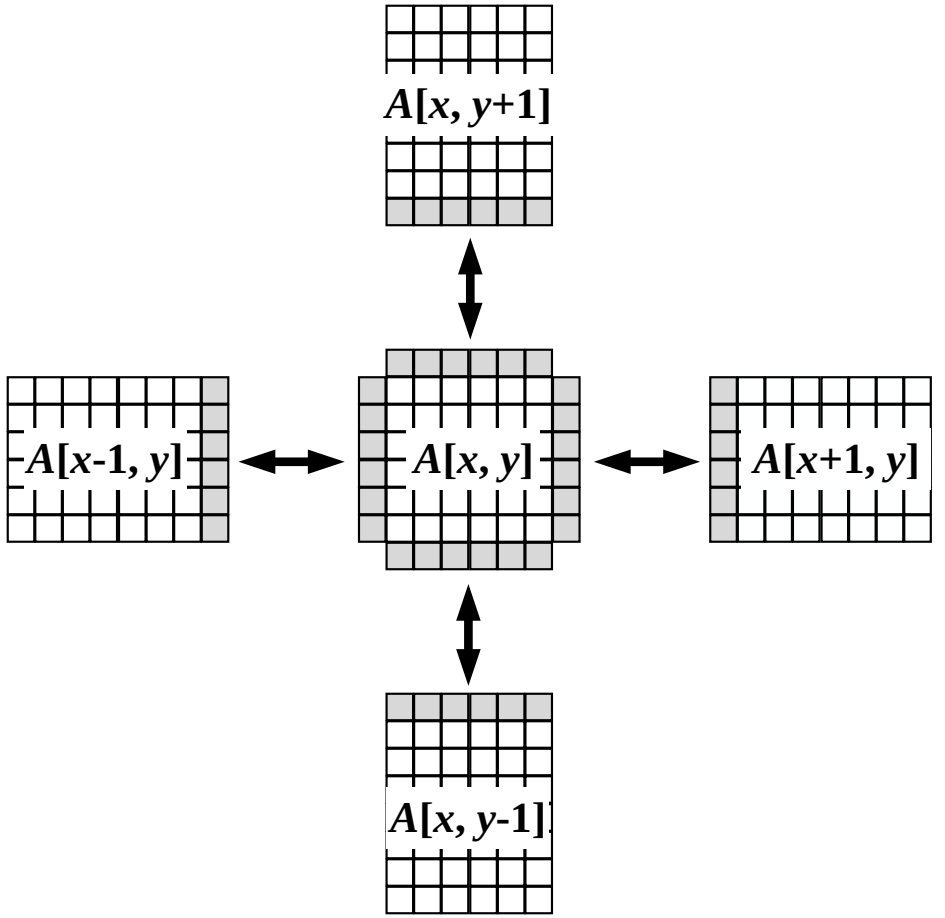


Схема обмена теневыми ячейками подматрицы $A[x, y]$

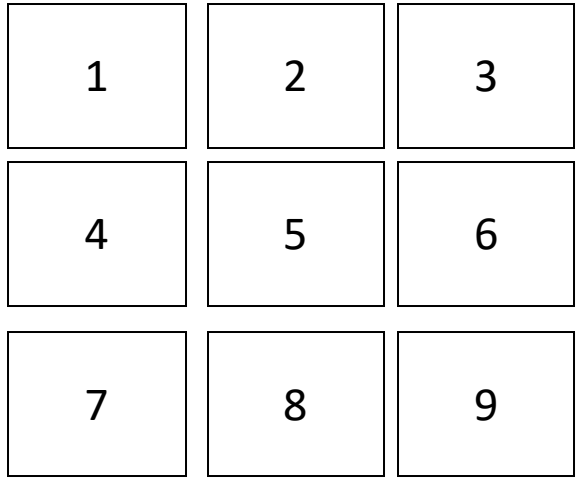
Исходный код тестовой программы

```
int main(int argc, char *argv[])
{
        /* Block Initialize */
        /* Save checkpoint function */
        for (;;) {
                /* Update interior points */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                local_newgrid[IND(i, j)] =
                                        (local_grid[IND(i - 1, j)] +
                                        local_grid[IND(i + 1, j)] +
                                        local_grid[IND(i, j - 1)] +
                                        local_grid[IND(i, j + 1)]) * 0.25;
                        }
                }
                /* Check termination condition */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                maxdiff = fmax(args);
                        }
                }
                /* Exchange shadow points with neughbors */
                MPI_Irecv(args);
                MPI_Isend(arg);
                /* Save checkpoint function */
                /* In simulation case:
                 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
                 */
        }
        /* Save checkpoint function */
        /* Block show result */
}
```

Паспорт задачи для каждого процесса

- Хранить в памяти каждого процесса всю расчетную область
- Хранить в памяти каждого процесса 2D паспорт задач:

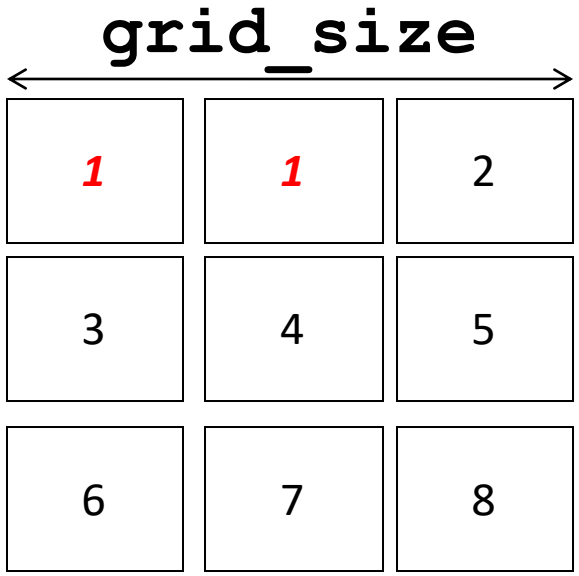
**необходимо для ассоциации процесса с подматрицей*




```
typedef struct
{
      int    number; // block number
      int    rank;   // processor rank
      size_t x;      // size by 'x'
      size_t y;      // size by 'y‘
} task_t;

task_t **gird_task;
```

Модификация исходного кода тестовой программы



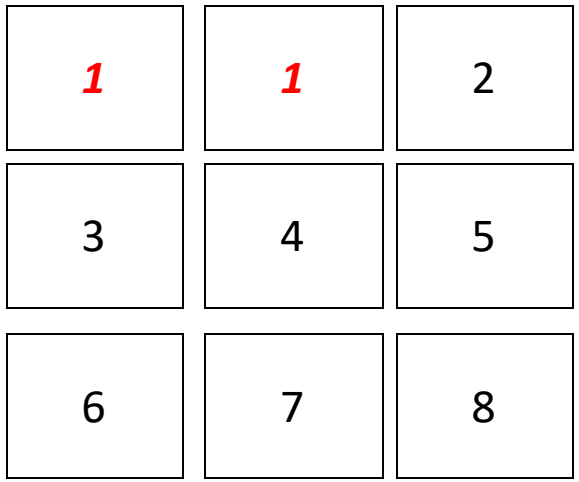

```
int main(int argc, char *argv[])
{
        /* Block Initialize */
        /* Save checkpoint function */
        for (;;) {
                /* Update interior points */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                local_newgrid[IND(i, j)] =
                                        (local_grid[IND(i - 1, j)] +
                                        local_grid[IND(i + 1, j)] +
                                        local_grid[IND(i, j - 1)] +
                                        local_grid[IND(i, j + 1)]) * 0.25;
                        }
```

```
task_t **gird_task;
```

```
}
```

- В случае выхода из строя процесса *i*, его подматрицу будет считать сосед (к примеру *i*+1, *i*-1, *i*+*grid_size*, *i*-*grid_size*)
- **Паспорт задачи хранит нужную техническую информацию*

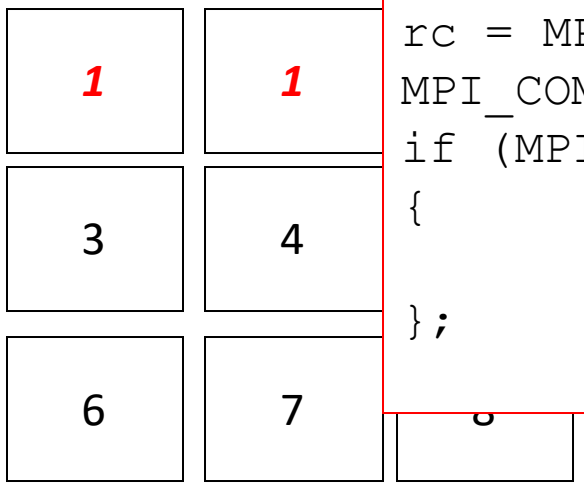
Модификация исходного кода тестовой программы




```
task_t **gird_task;
```

```
int main(int argc, char *argv[])
{
        /* Block Initialize */
        /* Save checkpoint function */
        for (;;) {
                /* Update interior points */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                local_newgrid[IND(i, j)] =
                                        (local_grid[IND(i - 1, j)] +
                                        local_grid[IND(i + 1, j)] +
                                        local_grid[IND(i, j - 1)] +
                                        local_grid[IND(i, j + 1)]) * 0.25;
                        }
                }
                /* Check termination condition */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                maxdiff = fmax(args);
                        }
                }
                /* Exchange shadow points with neughbors */
                MPI_Irecv(args);
                MPI_Isend(arg);
                /* Save checkpoint function */
                /* In simulation case:
                 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
                 */
        }
        /* Save checkpoint function */
        /* Block show result */
}
```

Модификация исходного кода тестовой программы

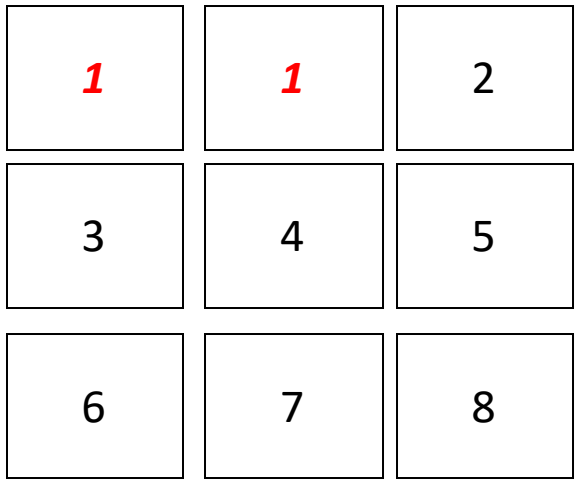



```
task_t **gird_task;
```

```
int main(int argc, char *argv[])
{
        /* Block Initialize */
        /* Save checkpoint function */
        for (;;) {
                /* Update interior points */
                }
                /* Exchange shadow points with neughbors */
                MPI_Irecv(args);
                MPI_Isend(arg);
                /* Save checkpoint function */
                /* In simulation case:
                 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
                 */
        }
        /* Save checkpoint function */
        /* Block show result */
}
```

```
rc = MPI_Allreduce(MPI_IN_PLACE, &maxdiff, 1, MPI_DOUBLE, MPI_MAX,
MPI_COMM_WORLD);
if (MPI_ERR_PROC_FAILED == rc)
{
        do_repair();
};
```

Модификация исходного кода тестовой программы

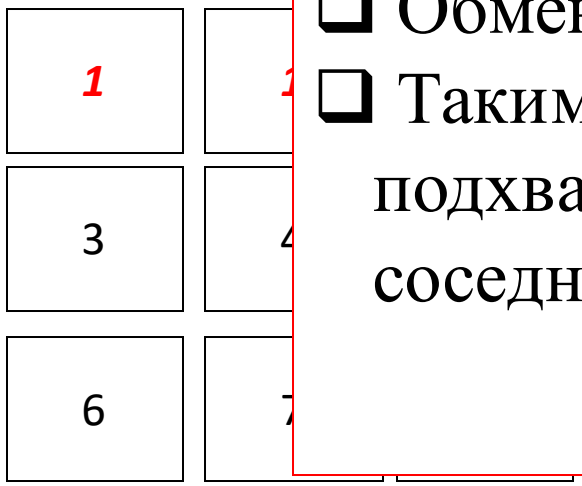



```
task_t **gird_task;
```

```
int main(int argc, char *argv[])
{
        /* Block Initialize */
        /* Save checkpoint function */
        for (;;) {
                /* Update interior points */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                local_newgrid[IND(i, j)] =
                                        (local_grid[IND(i - 1, j)] +
                                        local_grid[IND(i + 1, j)] +
                                        local_grid[IND(i, j - 1)] +
                                        local_grid[IND(i, j + 1)]) * 0.25;
                        }
                }
                /* Check termination condition */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                maxdiff = fmax(args);
                        }
                }
                /* Exchange shadow points with neughbors */
                MPI_Irecv(args);
                MPI_Isend(arg);
                /* Save checkpoint function */
                /* In simulation case:
                 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
                 */
        }
        /* Save checkpoint function */
        /* Block show result */
}
```

Модификация исходного кода тестовой программы (ver. 1.0)

- ~~Отказаться от обмена теневыми ячейками~~
- Обмениваться целым блоком (подматрицей процесса)
- Таким образом каждый процесс сможет «на лету» подхватывать вычисления в случае выхода из строя соседнего процесса

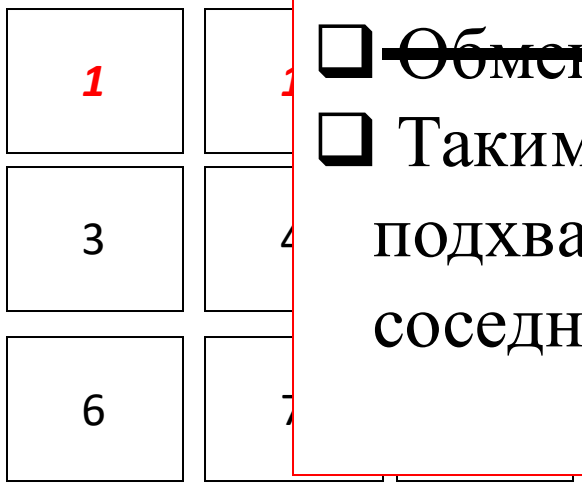



```
task_t **gird_task;
```

```
MPI_Irecv(args);
MPI_Isend(arg);
/* Save checkpoint function */
/* In simulation case:
 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
 */
}
/* Save checkpoint function */
/* Block show result */
}
```

Модификация исходного кода тестовой программы (ver. 1.1)

- Оставить обмен теневыми ячейками (все же стоит оставить этот подход)
- ~~Обмениваться целым блоком (подматрицей процесса)~~
- Таким образом каждый процесс сможет «на лету», подхватывать вычисления в случае выхода из строя соседнего процесса




```
task_t **gird_task;
```

```
MPI_Irecv(args);
MPI_Isend(arg);
/* Save checkpoint function */
/* In simulation case:
 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
 */
}
/* Save checkpoint function */
/* Block show result */
}
```

Модификация исходного кода тестовой программы

```
int main(int argc, char *argv[])
{
        /* Block Initialize */
        /* Save checkpoint function */
        for (;;) {
                /* Update interior points */
                for (int i = 1; i <= ny; i++) {
                        for (int j = 1; j <= nx; j++) {
                                local_newgrid[IND(i, j)] =
                                        (local_grid[IND(i - 1, j)] +
```

1	*1*	
3	4	
6	7	

```
rc = MPI_Waitall(8, reqs, MPI_STATUS_IGNORE);
if (MPI_ERR_PROC_FAILED == rc)
{
        do_repair();
};
```

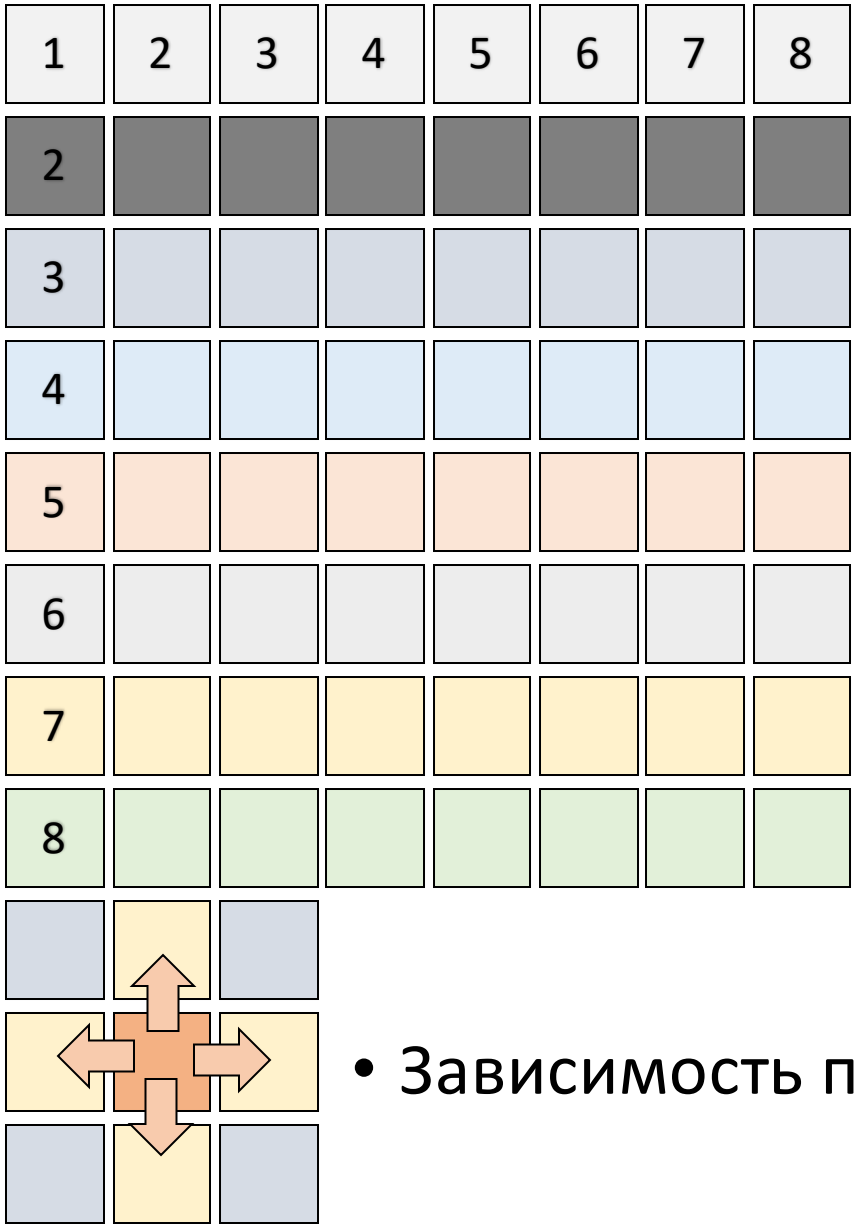
task_t **gird_task;

```
                /* Sync point */
                /* In simulation case:
                 * call -> MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
                 */
        }
        /* Save checkpoint function */
        /* Block show result */
}
```

Децентрализованная “выживающая” схема (продолжение)

Тестовый пример: heat2d

Библиотека отказоустойчивости:UFLM



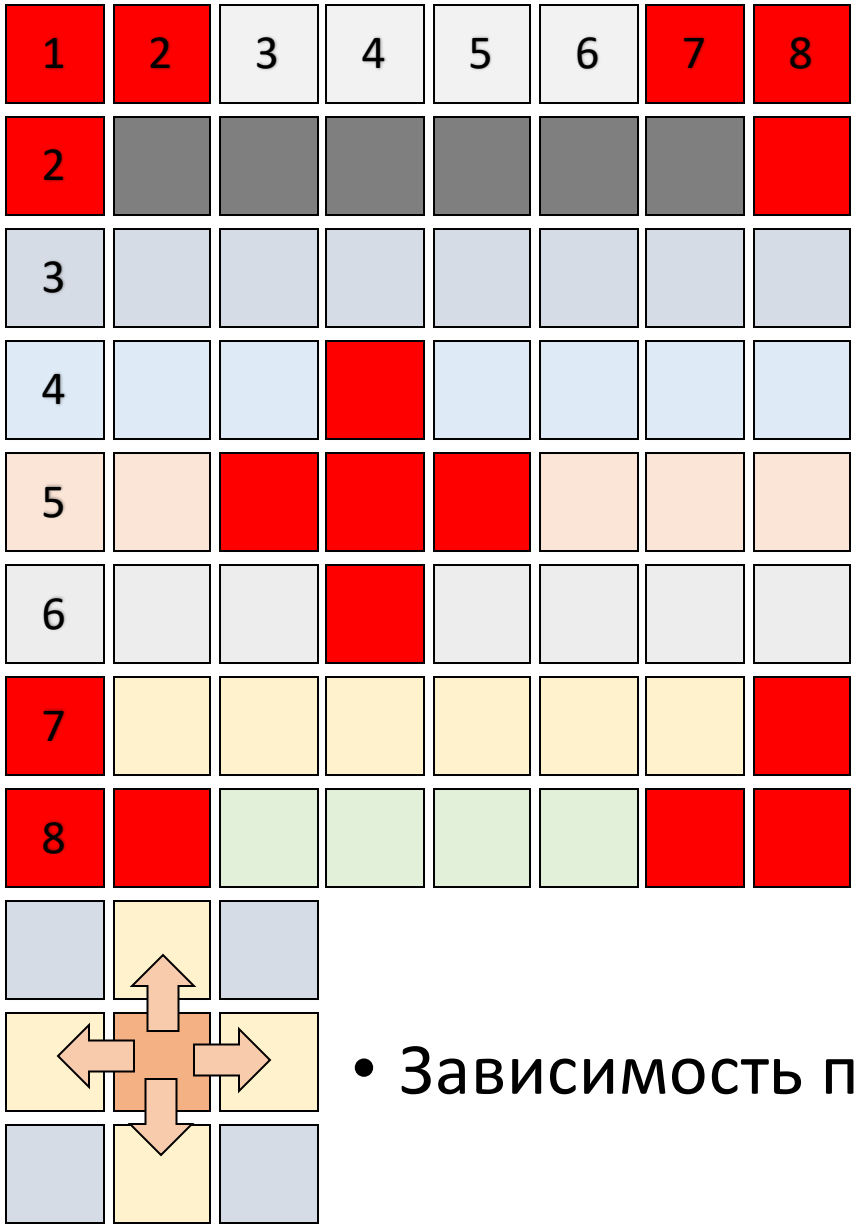

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Расчетная область

- Зависимость по данным



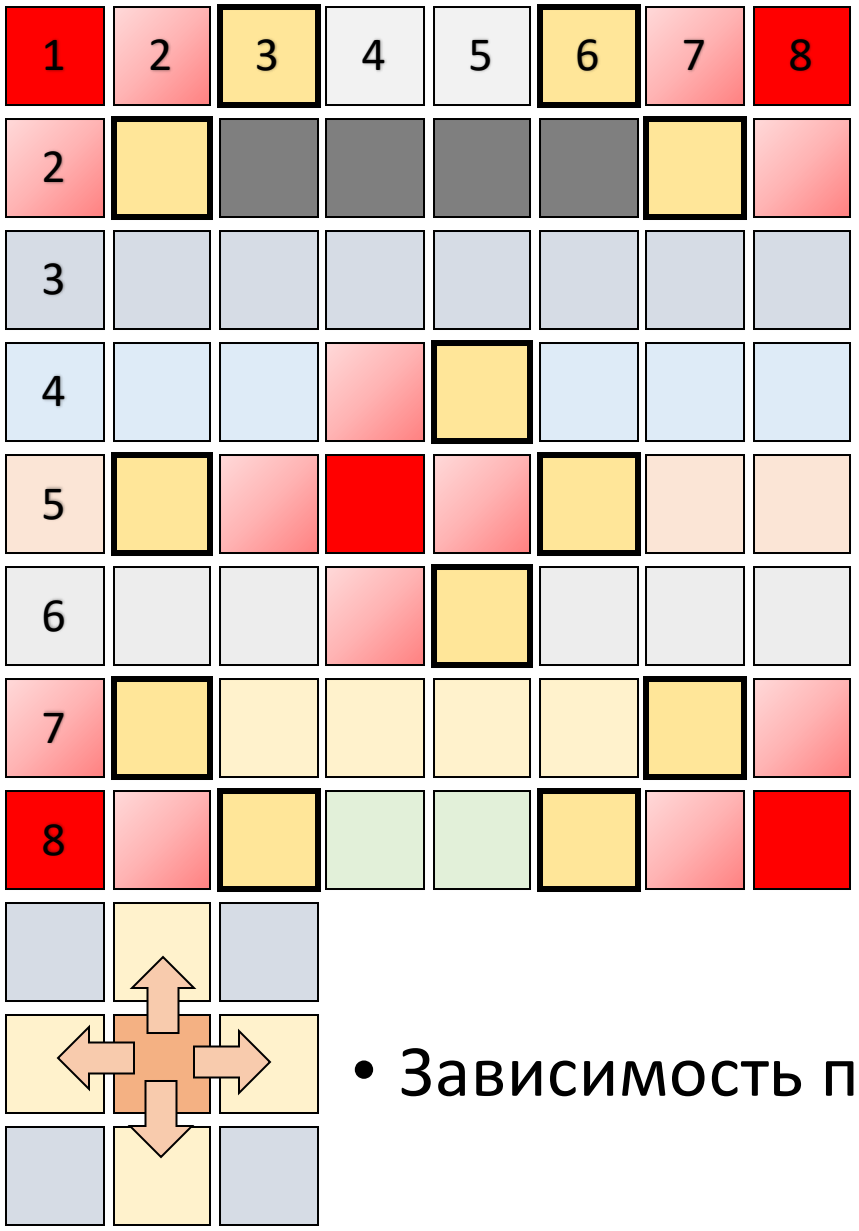

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- отказавший процесс

- Зависимость по данным



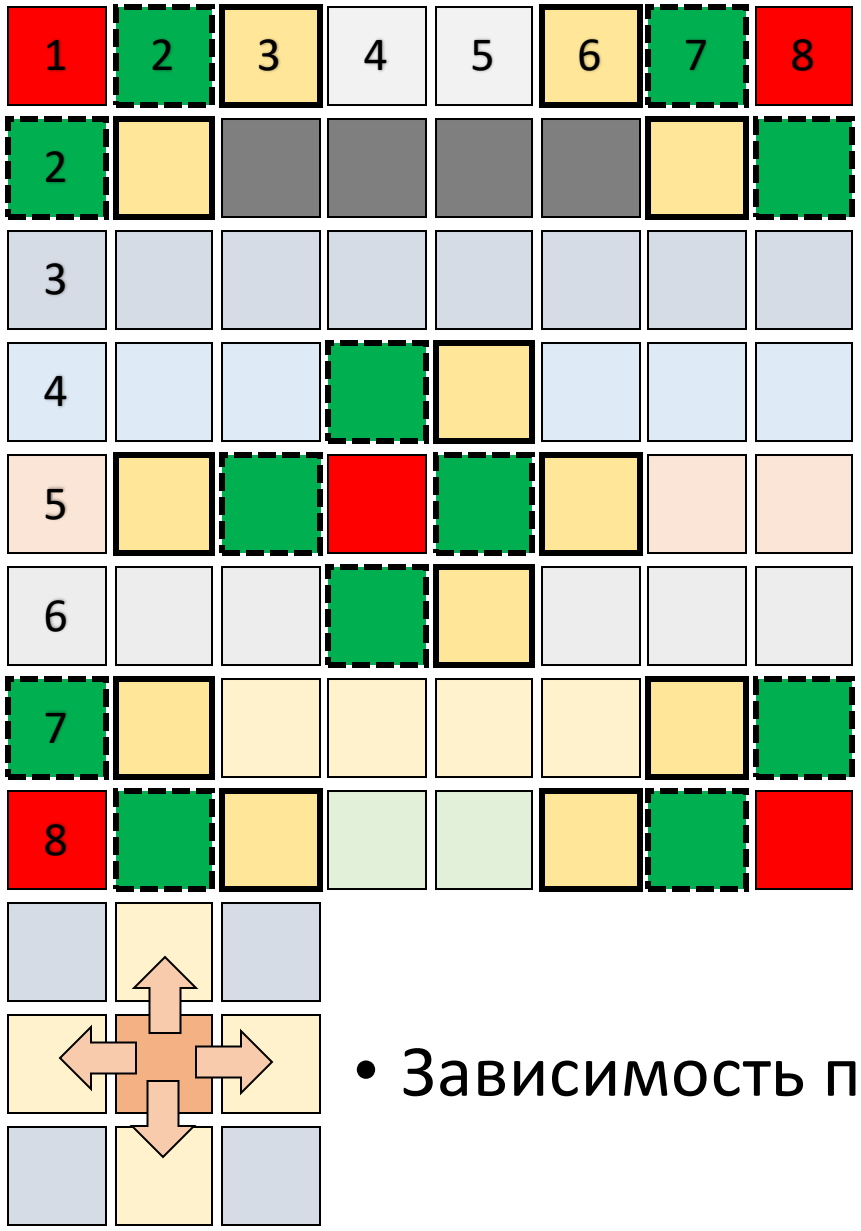

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- отказавший процесс
- восстанавливающий процесс (сосед)
- будет восстановлен соседом

- Зависимость по данным



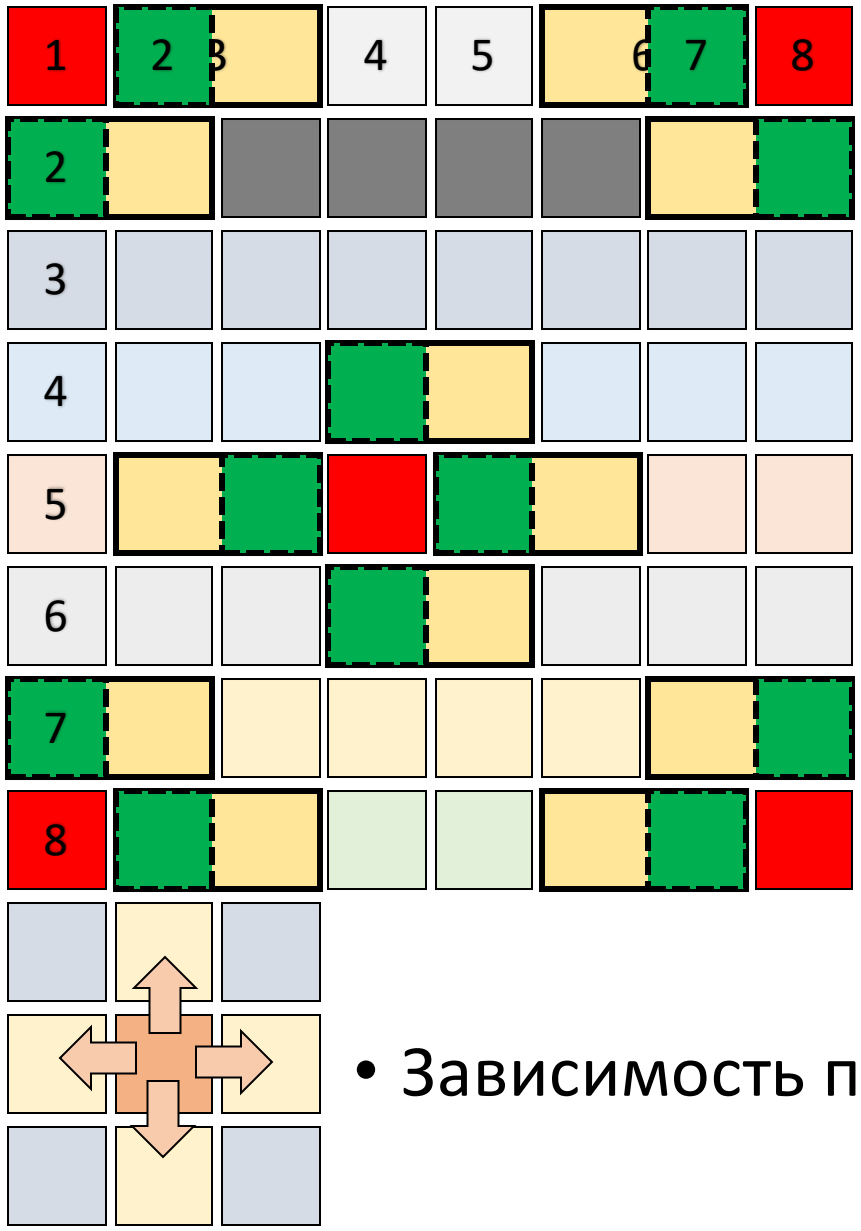

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- отказавший процесс
- восстанавливающий процесс (сосед)
- восстановленный процесс соседом

- Зависимость по данным



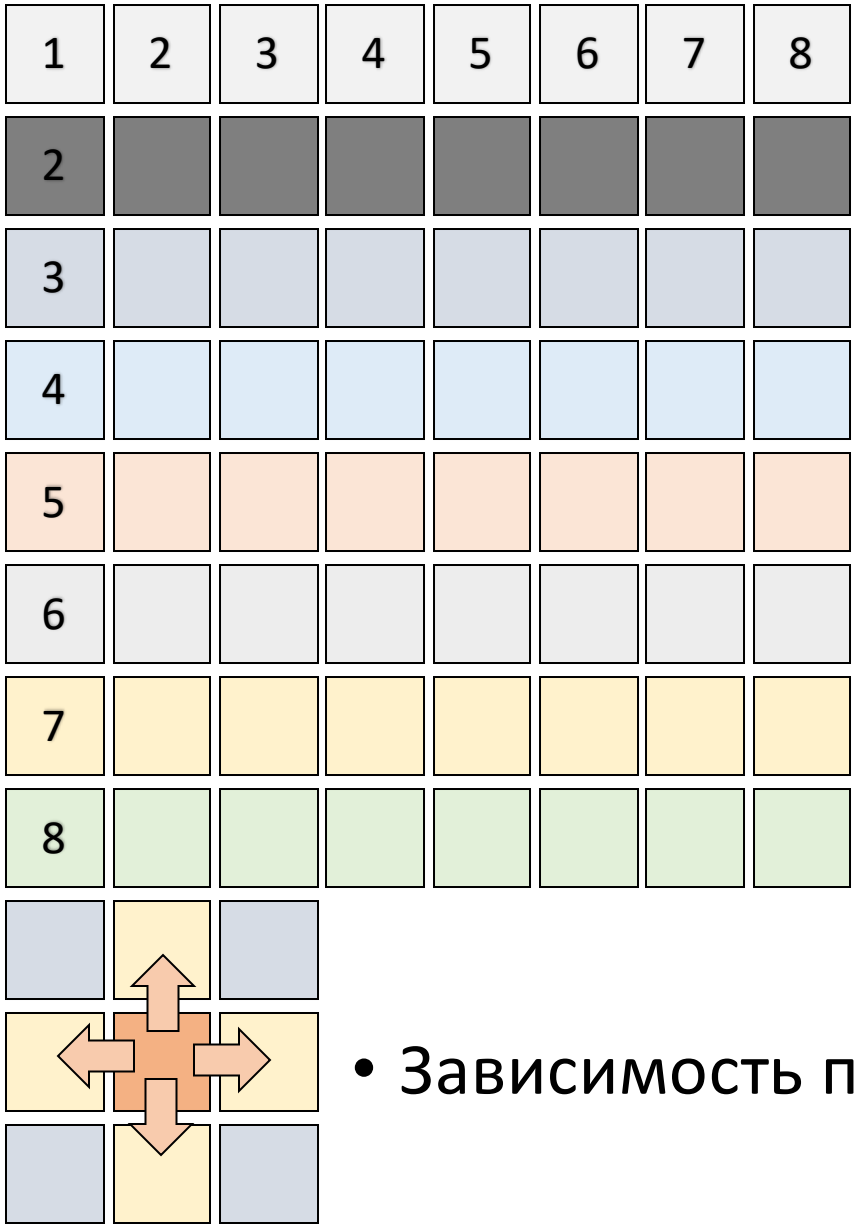

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- отказавший процесс
- восстанавливающий процесс (сосед)
- восстановленный процесс соседом
- восстановление не возможно!

- Зависимость по данным



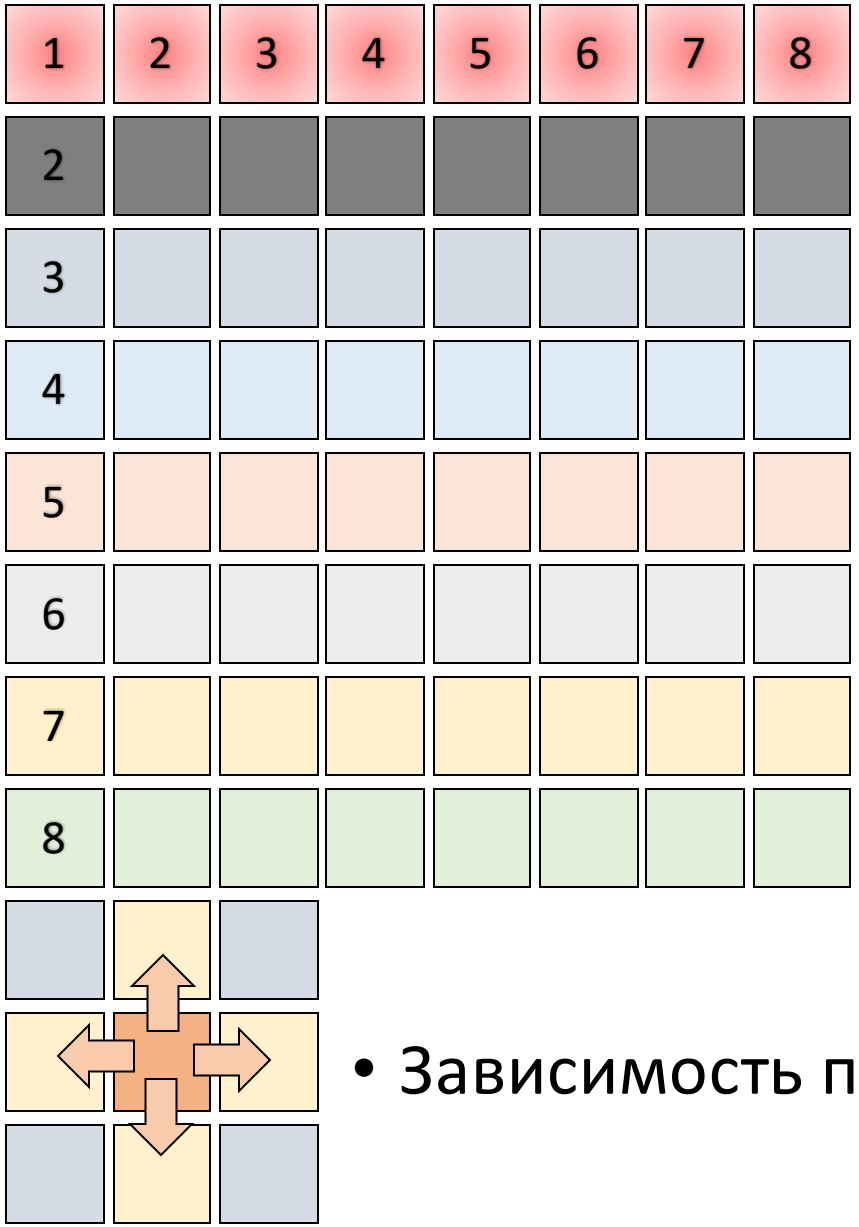

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Расчетная область

- Зависимость по данным



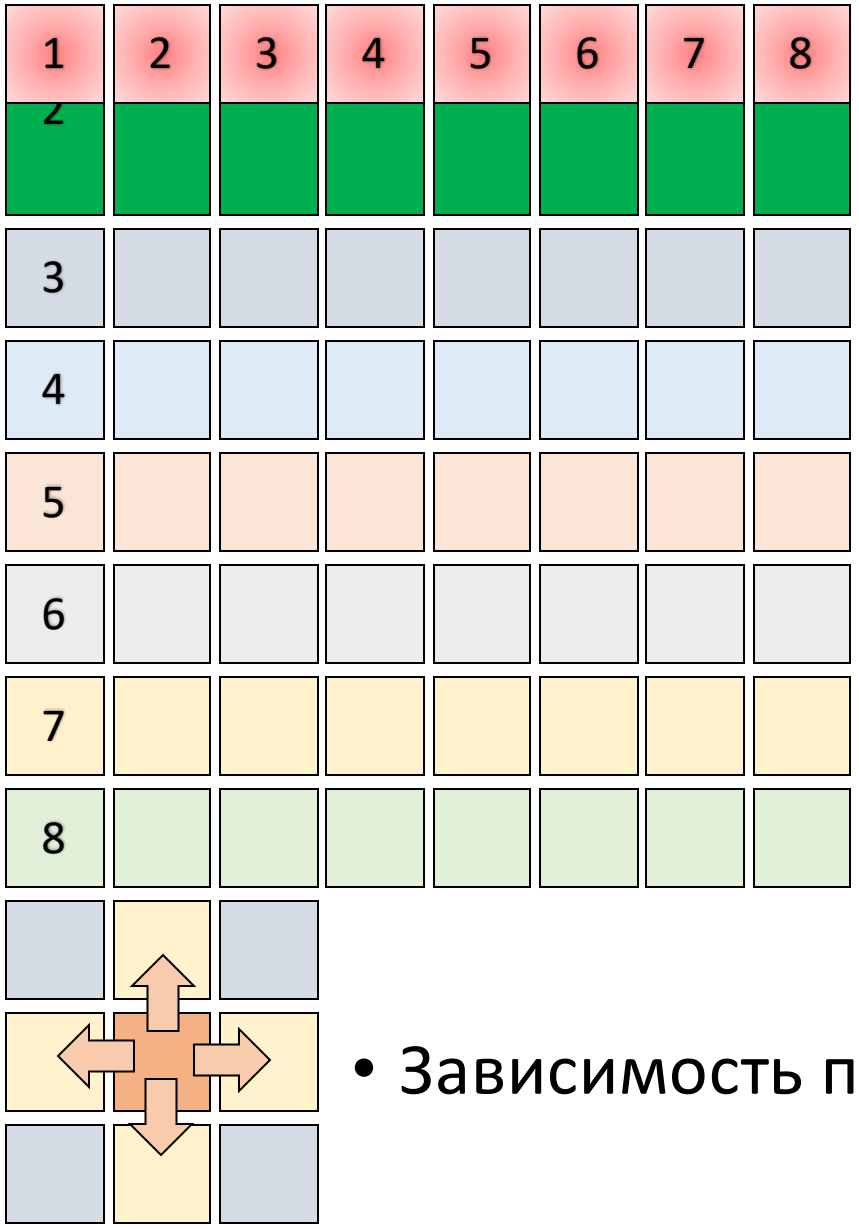

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Выход из строя одного узла, приведет к выходу из строя 8 процессов

- Зависимость по данным



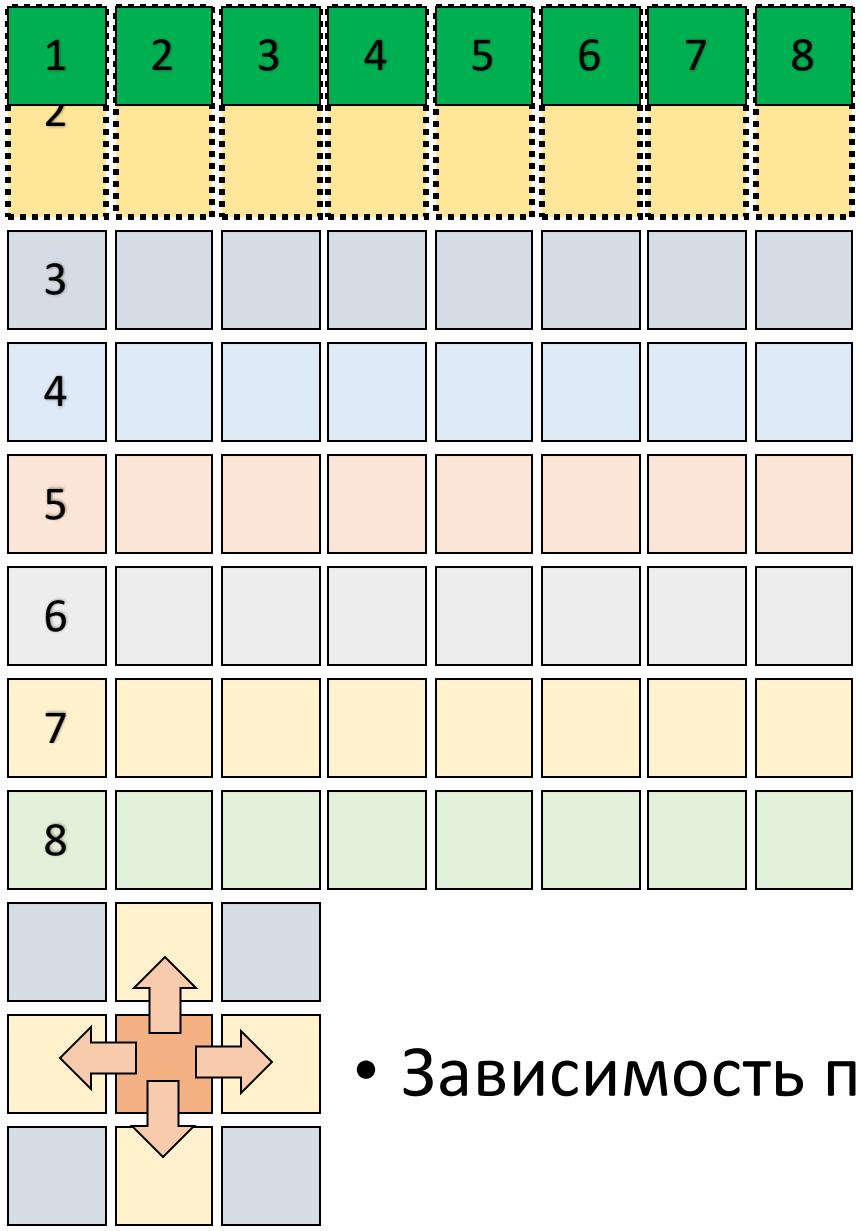

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Восстановление возможно

- Зависимость по данным



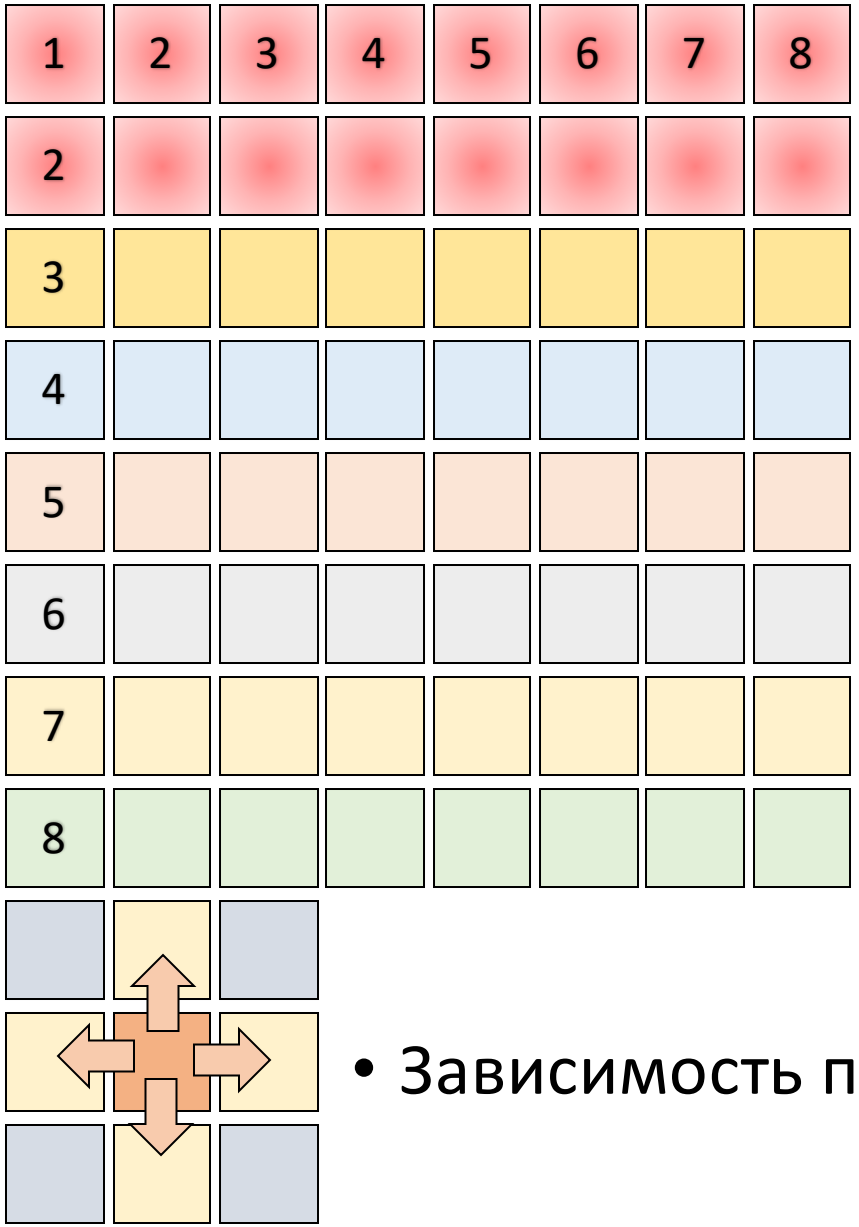

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Восстановление завершено

- Зависимость по данным



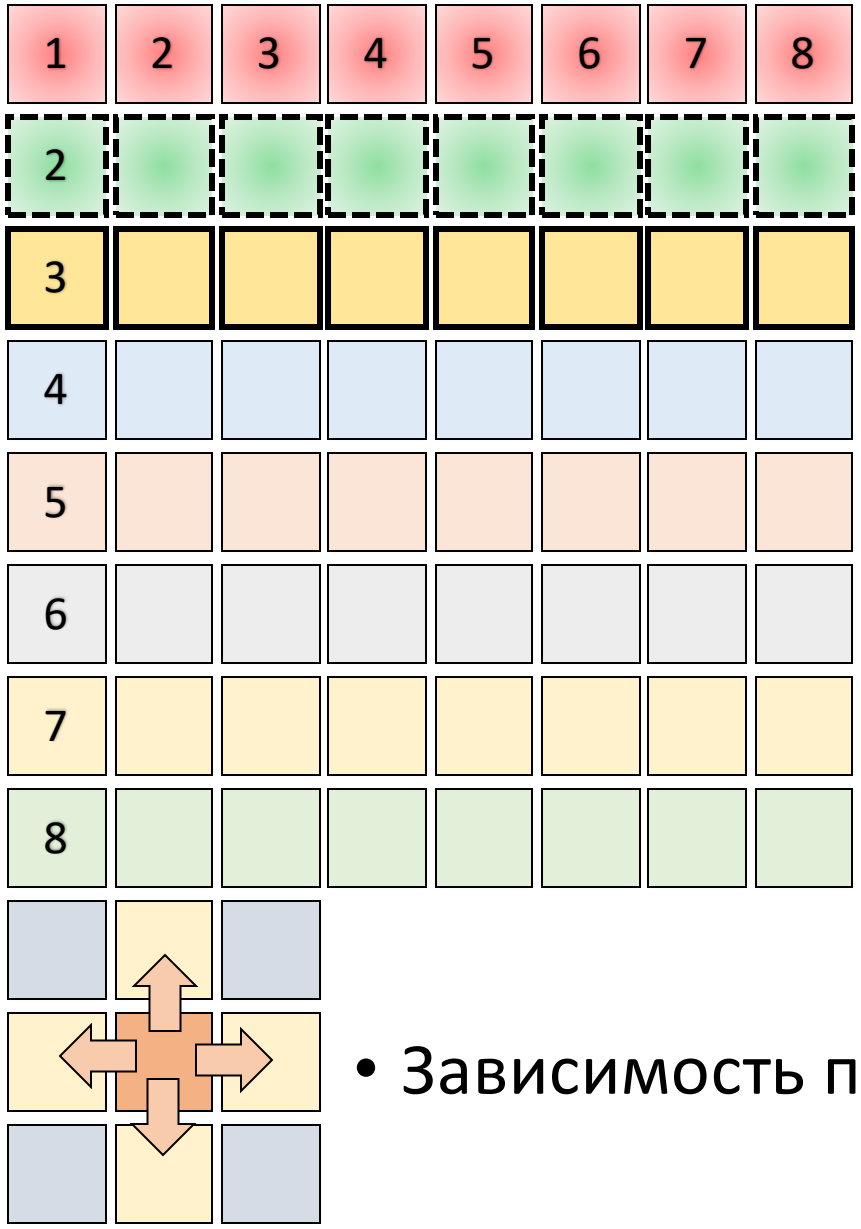

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Выход из строя двух узлов, приведет к выходу из строя 16 процессов

- Зависимость по данным



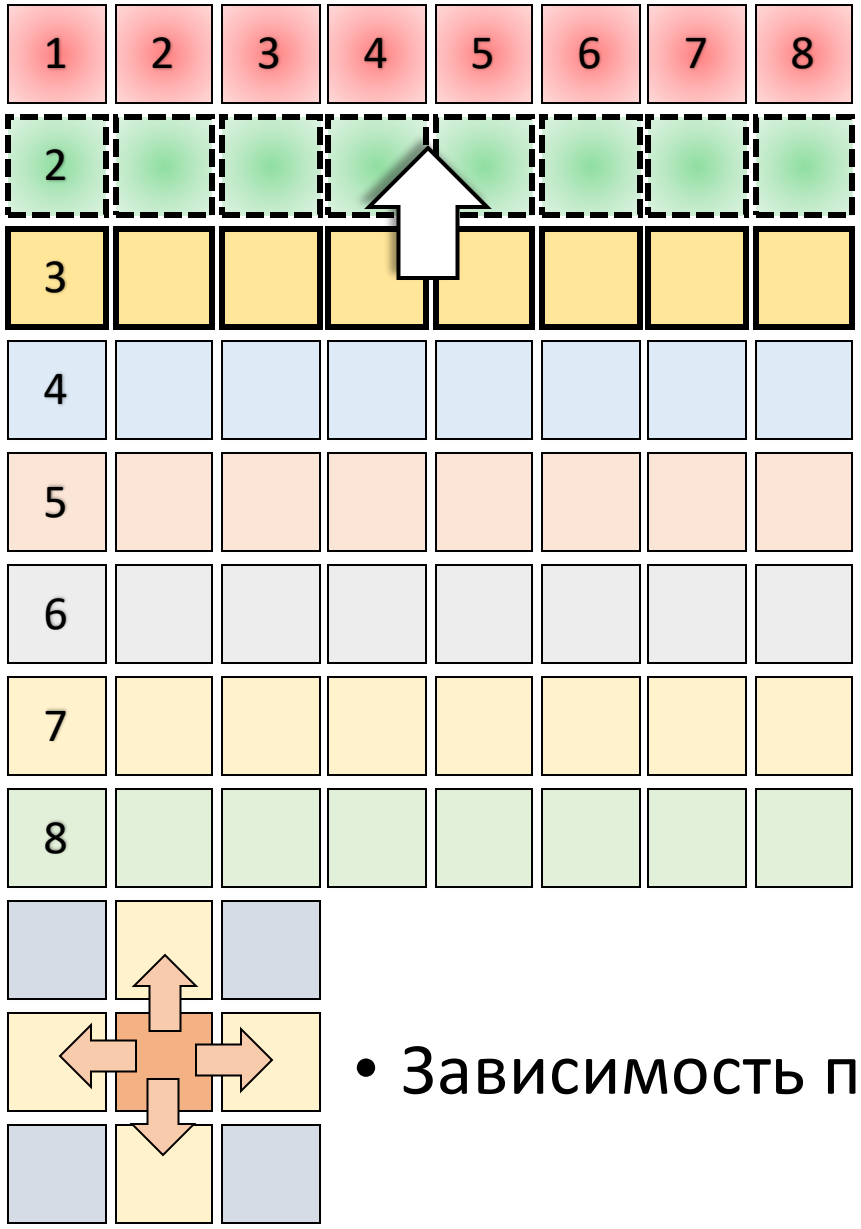

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Полностью, восстановить не возможно

- Зависимость по данным



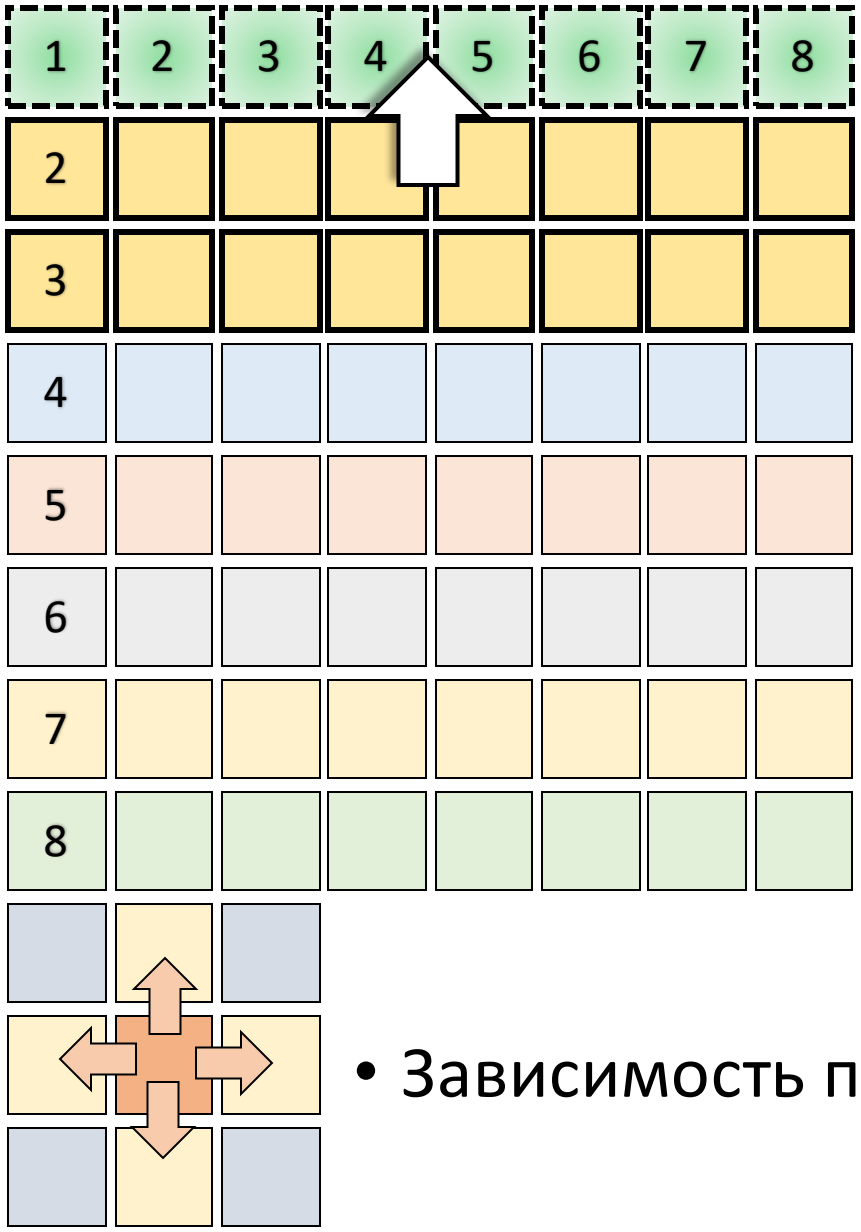

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Использовать подход, «ступенчатого» восстановления:
- На первом этапе восстановить линию 2 (см. рисунок)
- На втором этапе восстановить линию 1 (см. рисунок)
- И т.д. …

- Зависимость по данным



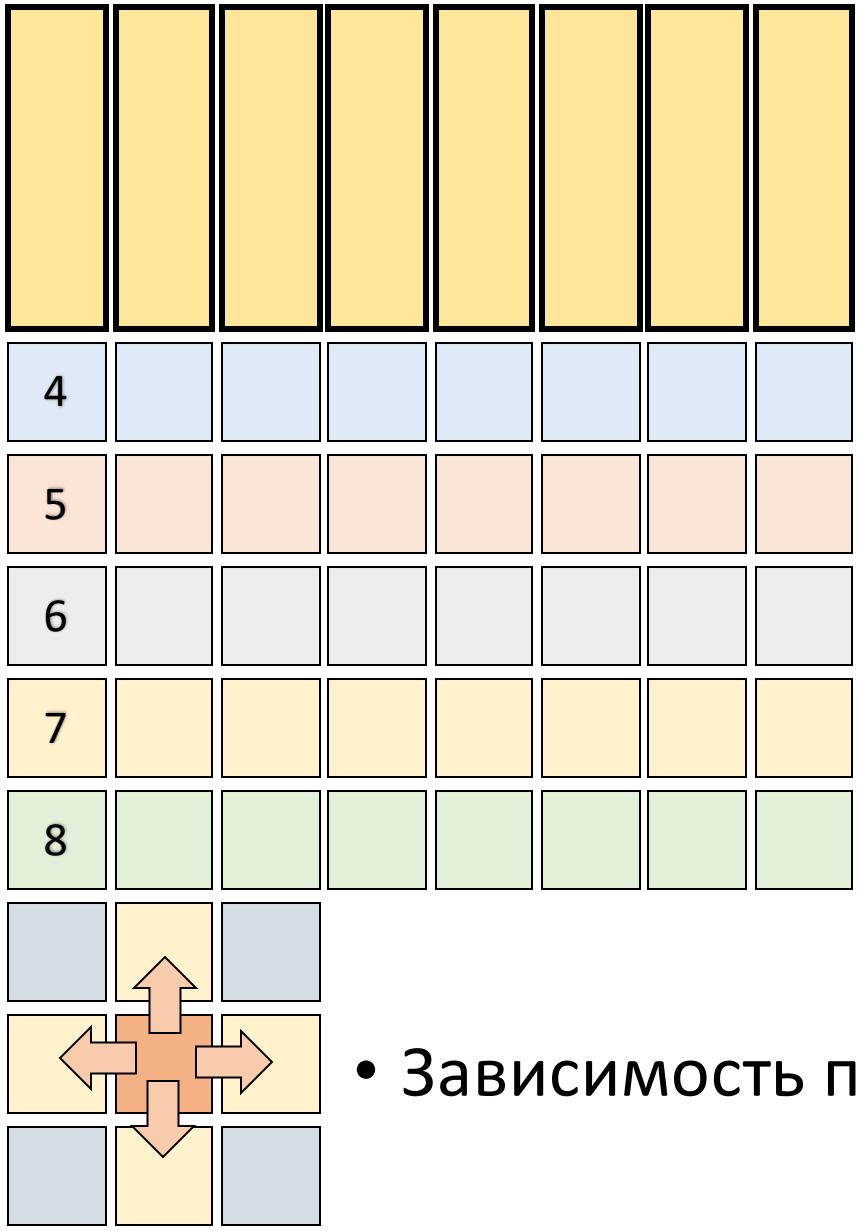

```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Использовать подход, «ступенчатого» восстановления:
- На первом этапе восстановить линию 2 (см. рисунок)
- На втором этапе восстановить линию 1 (см. рисунок)
- И т.д. …

- Зависимость по данным




```
#!/bin/bash

#SBATCH --nodes=8 --ntasks-per-node=8 mpiexec

./heat_2d 8192 8192
```

- Восстановление завершено

- Зависимость по данным