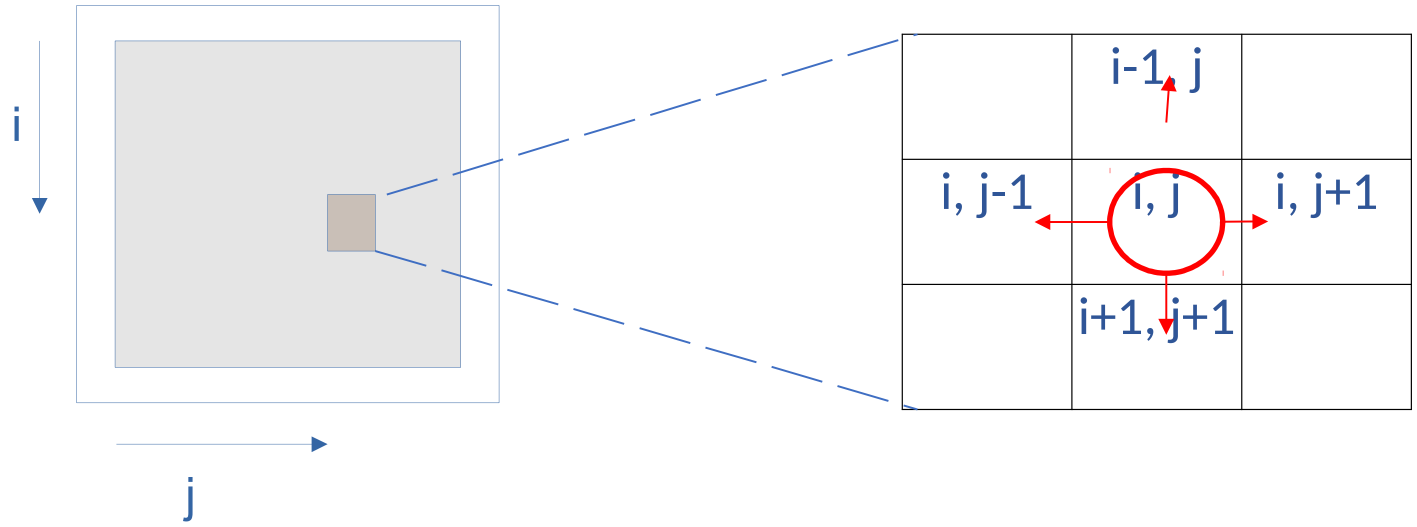
Jacobi solver: 5-point Stencil

Input: matrix u ← Not modified

```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
    for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +   // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -   // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                     // element u[i][j]
        utmp[n*i + j] = tmp/4;                    // element utmp[i][j]
    }
}}
```

Output: matrix utmp



Calculation of each element utmp[i][j] needs to READ five elements of u: u[i][j] and four neighbors (upper, lower, left, right)

There are no *loop carried dependencies*: from iteration to iteration there are no dependencies and different elements of utmp can be computed in parallel.

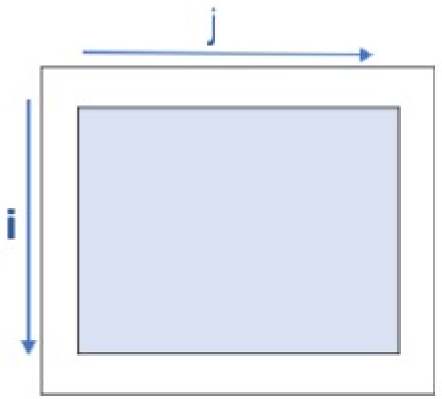
Jacobi solver: Task definition → compute all iterations of loops i & j

```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;
```

task code

```
for (i = 1; i < n-1; i++) {
    for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +     // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -     // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                       // element u[i][j]
        utmp[n*i + j] = tmp/4;                      // element utmp[i][j]
    }
}}
```

t_{body} time units



What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n - 2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$

TDG :

1 task

```
for (i=1; i<n-1…)
   for (j=1; j<n-1….)
```

Jacobi solver: Task definition → compute an iteration of outer loop i

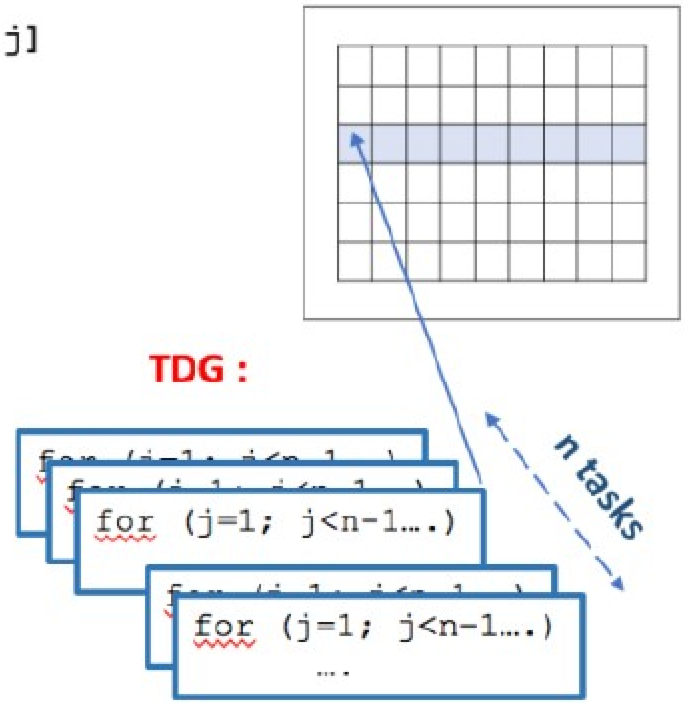
```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
     for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +     // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -     // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                       // element u[i][j]
        utmp[n*i + j] = tmp/4;                      // element utmp[i][j]
     }
}}
```

task code

t_{body} *time units*

What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n - 2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$



Jacobi solver: Task definition → compute an iteration of inner loop j

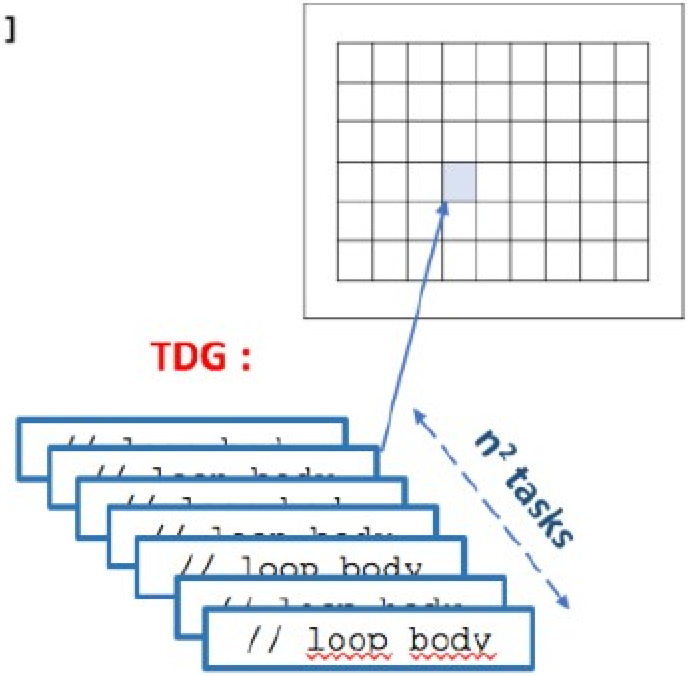
```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
    for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +      // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -      // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                        // element u[i][j]
        utmp[n*i + j] = tmp/4;                       // element utmp[i][j]
    }
}}
```

task code

t_{body} time units

What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n-2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$



Jacobi solver: Task definition → compute r consecutive iterations of loop i

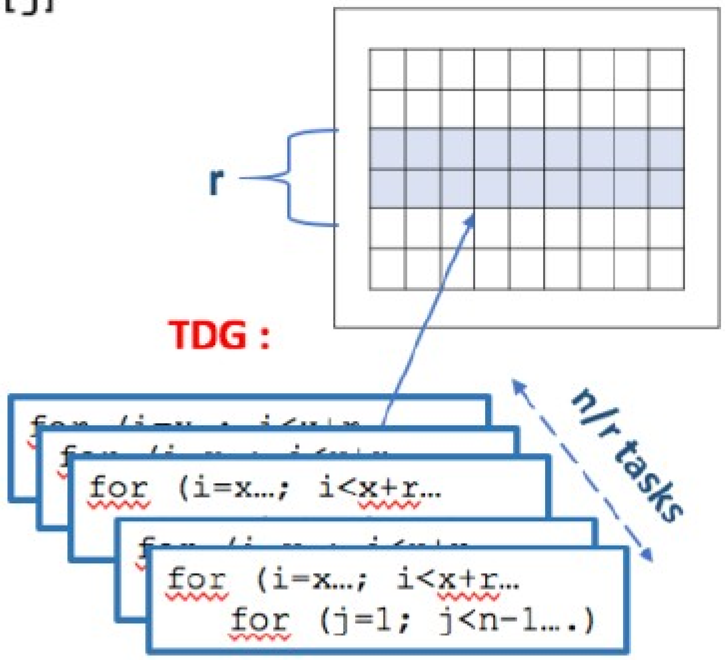
for (i=x; i < x+r; i++)
  for (j=1; j < n-1; j++)

```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {                              task code
     for (j = 1; j < n-1; j++) {
          tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +     // elements u[i+1][j] and u[i-1][j]
                u[n*i + (j+1)] + u[n*i + (j-1)] -     // elements u[i][j+1] and u[i][j-1]
                4 * u[n*i + j];                        // element u[i][j]
          utmp[n*i + j] = tmp/4;                       // element utmp[i][j]
     }
}}
```

t_{body} time units

What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n - 2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$



Jacobi solver: Task definition → compute c consecutive iterations of loop j

for (j=x; j < x+c: j++)

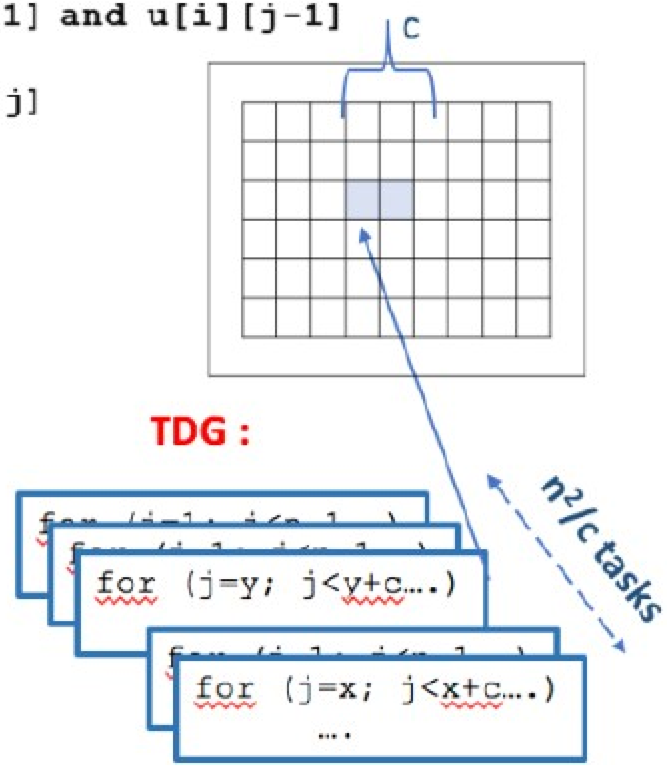
task code

```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
    for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +    // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -    // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                      // element u[i][j]
        utmp[n*i + j] = tmp/4;                     // element utmp[i][j]
    }
}}
```

t_{body} time units

What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n-2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$



Jacobi solver: Task definition → compute r x c iterations of loop i & j

for (i=x; i < x+r; i++)
  for (j=y; j < y+c; j++)

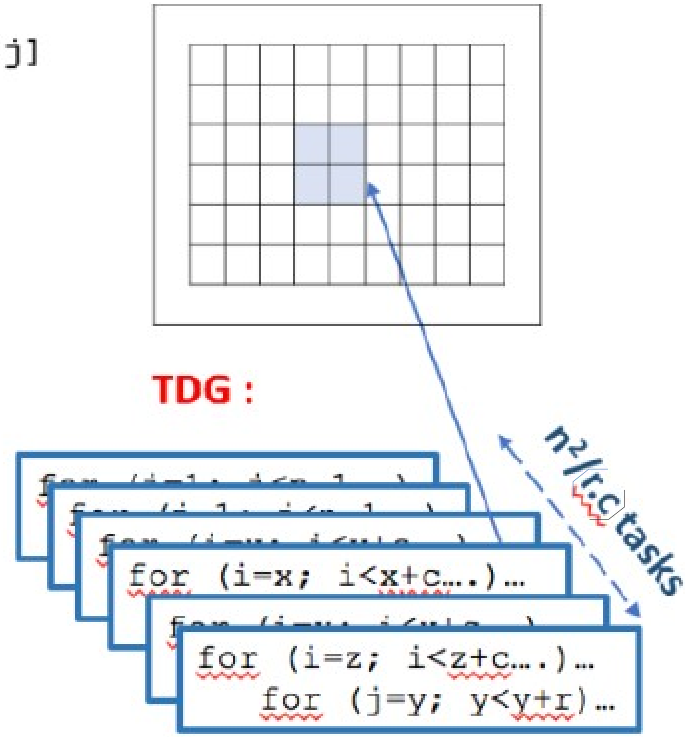
task code

```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
     for (j = 1; j < n-1; j++) {
          tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +     // elements u[i+1][j] and u[i-1][j]
                u[n*i + (j+1)] + u[n*i + (j-1)] -     // elements u[i][j+1] and u[i][j-1]
                4 * u[n*i + j];                       // element u[i][j]
          utmp[n*i + j] = tmp/4;                      // element utmp[i][j]
     }
}}
```

t_{body} time units

What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n-2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$



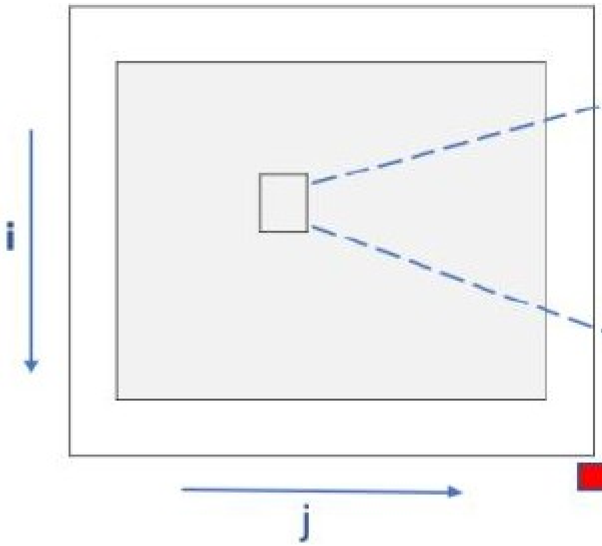
Jacobi solver: Task definition → including overheads

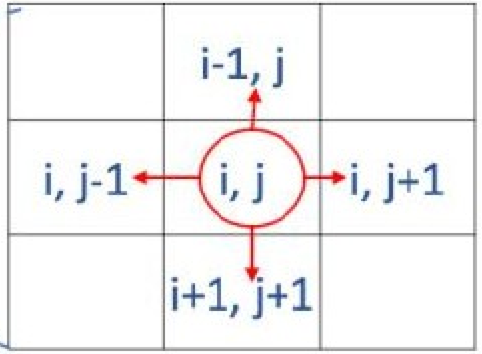
Task is … (granularity)	Num. tasks	Task cost	Task creation ovh
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	t_{create}
Each iteration of i loop	n	$n \cdot t_{body}$	$n \cdot t_{create}$
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{create}$
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$(n \div r) \cdot t_{create}$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$(n^2 \div c) \cdot t_{create}$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$(n^2 \div (r \cdot c)) \cdot t_{create}$

There is a trade-off between task granularity and task creation overhead

Gauss-Seidel solver: 5-point Stencil

```
void compute(int n, double *u ) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
    for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +   // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -   // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                     // element u[i][j]
        u[n*i + j] = tmp/4;                       // element u[i][j]
    }
}}
```





In a sequential execution elements are processed in the order:

- *from left to right (outer loop i) and*
- *from upper to bottom (inner loop j)...*

Matrix u is input / output

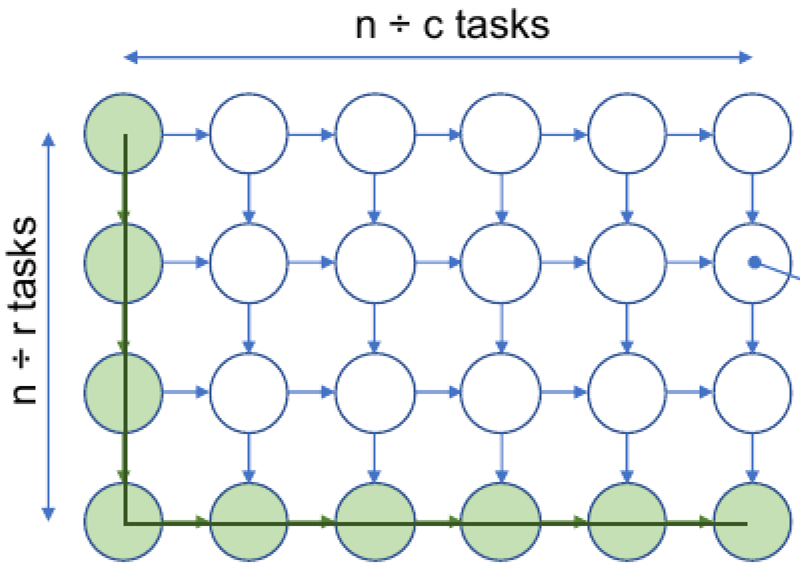
A parallel code needs to respect these dependencies

Elements u[i-1][j] and u[i][j-1] are **modified** before being used by calculation of u[i][j]

Elements u[i+1][j] and u[i][j+1] are **not modified** before being used by calculation of u[i][j]

Gauss-Seidel solver: Task definition, dependencies and parallel execution time

Assuming each task computes a block of $r \times c$ iterations of the `i` and `j` loops, respectively



$$t_{task} = (r \cdot c) \cdot t_{body}$$

$$T_\infty = \overbrace{(n \div c + n \div r - 1) \cdot t_{task}}^{\text{computation}} + \overbrace{(n \div c + n \div r - 2) \cdot t_{synch}}^{\text{synchronization ovh.}}$$

Again, trade-off between task granularity and task synchronization overhead