Abstractions to Separate Concerns in Semi-Regular Grid Computations

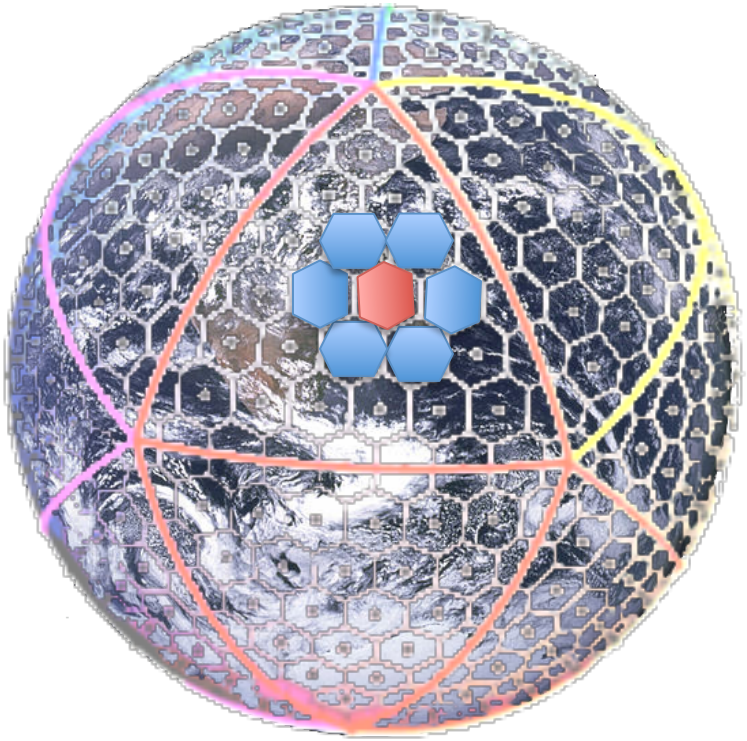
International Conference on Supercomputing
June 12, 2013

Andy Stone
Michelle Strout

Motivation

- Scientists use Earth Simulation programs to:
  - **Predict** change in the weather and climate
  - **Gain insight** into how the climate works
- Performance is crucial: it impacts **time-to-solution** and **accuracy**

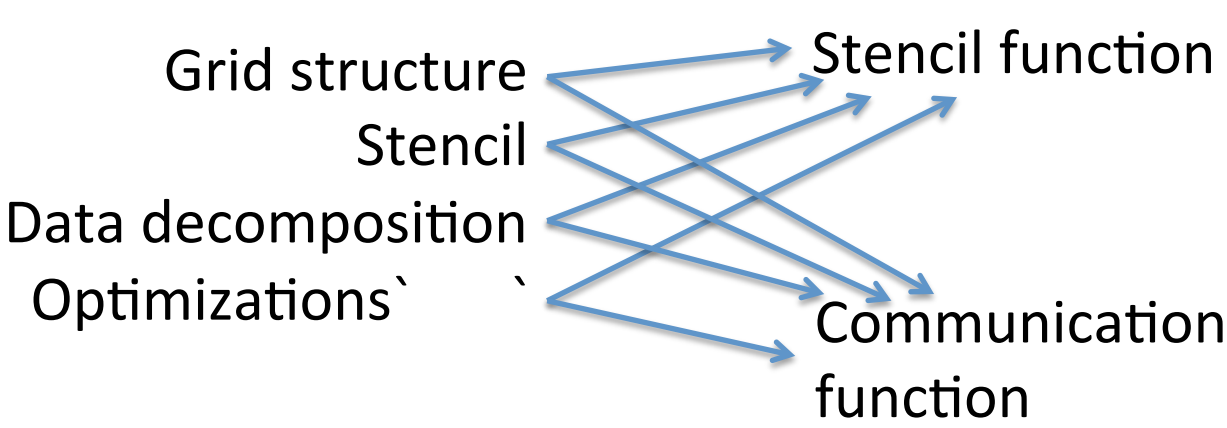
Discretize!



Apply a stencil!

Problem: Implementation details tangle code

```
Loop {
    communicate()
    stencil()
}
```



5% (150 SLOC) **CGPOP**
25% (1770 SLOC) **SWM**

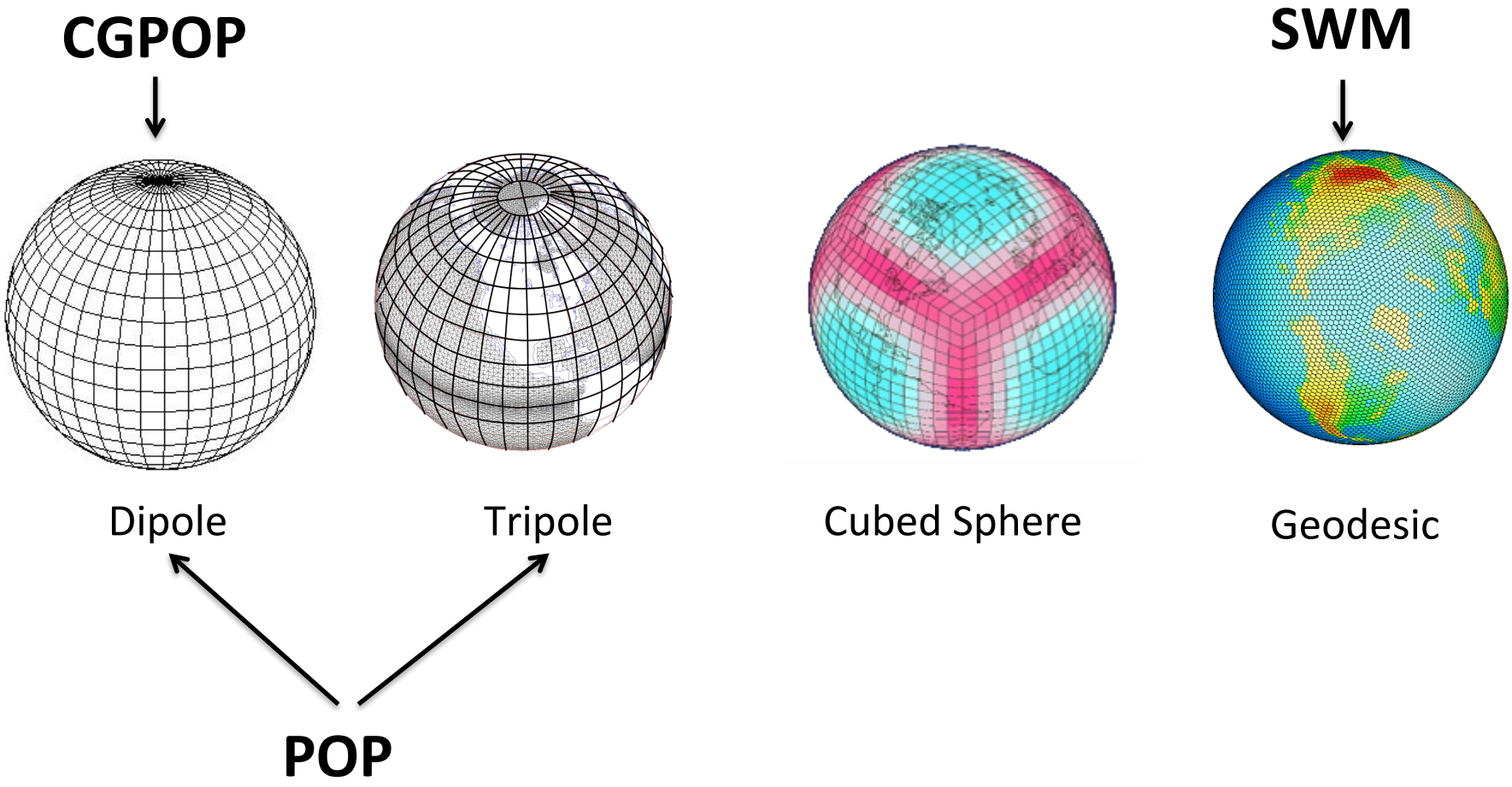
19% (570 SLOC) **CGPOP**
30% (2120 SLOC) **SWM**

Generic stencil and communication code isn't written due to need for performance.

Shallow Water Model (SWM) Stencil Code

```
x2(:,:) = zero
DO j = 2,jm-1
   DO i = 2,im-1
      x2(i,j) = area_inv(i,j)* &
                      ((x1(i+1,j  )-x1(i,j))*laplacian_wghts(1,i,j)+ &
                       (x1(i+1,j+1)-x1(i,j))*laplacian_wghts(2,i,j)+ &

                                          . . .

    ENDDO
ENDDO

! NORTH POLE
i = 2; j = jm
x2(i,j) = area_inv(i,j)*laplacian_wghts(1,i,j)*((x1(i+1,j)- . . .

! SOUTH POLE
i = im; j = 2
x2(i,j) = area_inv(i,j)*laplacian_wghts(1,i,j)*((x1(  1, jm)- . . .
```

Earth Grids



Solution

- Create set of abstractions to separate details
  - Subgrids, border maps, grids, distributions, communication plans, and data objects
- Provide abstractions in a library (**GridWeaver**)
- Use code generator tool to remove library overhead and optimize

Talk outline

Grid connectivity

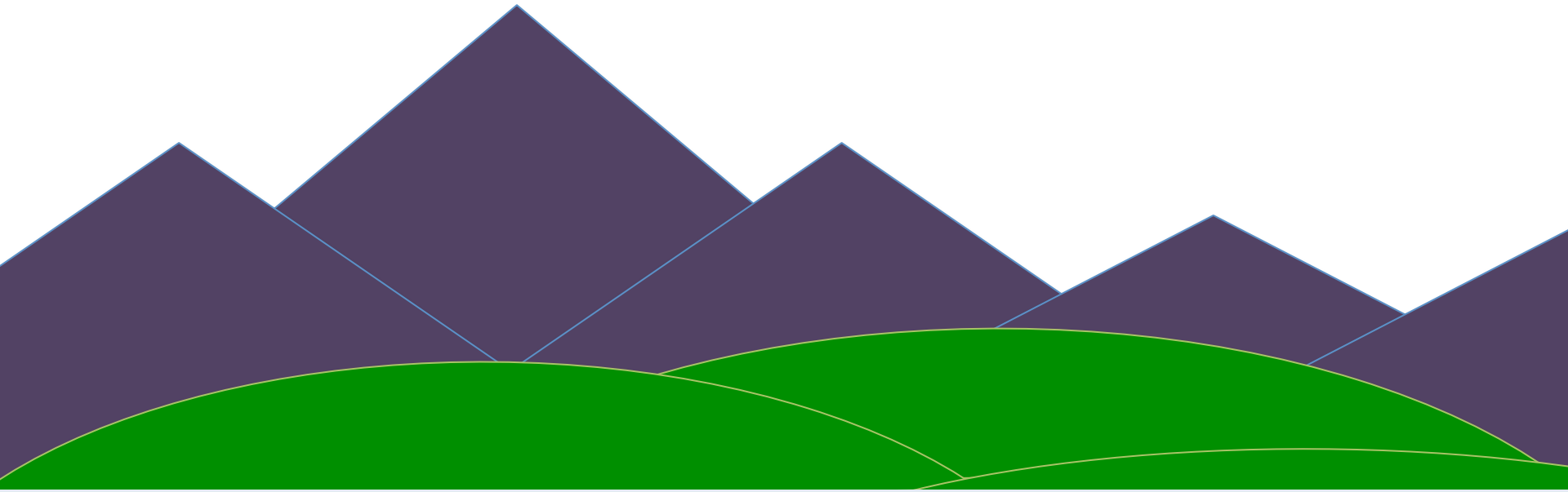
Parallelization

Planning Communication

Maintaining Performance

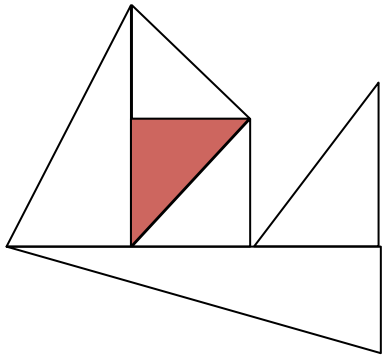
Conclusions

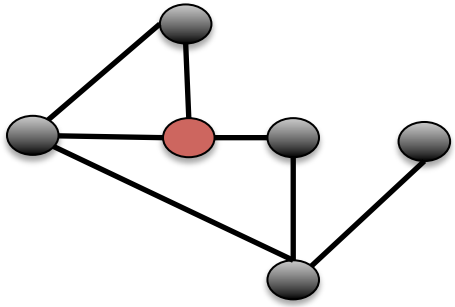
Grid connectivity



Grid type tradeoffs

Irregular



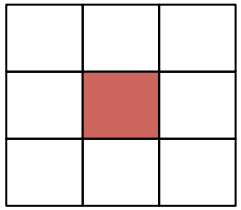


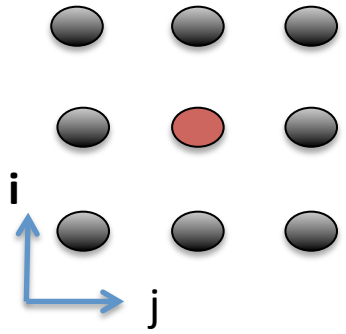
✔ Most general

Structured with a graph

✗ Connectivity is explicitly stored

Regular





✗ Restricted to a regular tessellation of an underlying geometry

Structured with an array

✔ Storage efficient

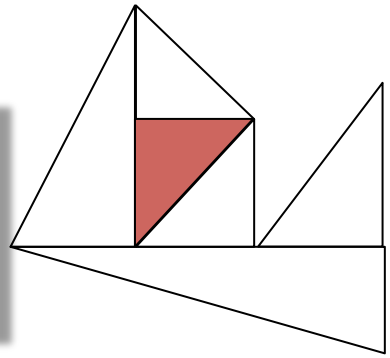
Impact on code

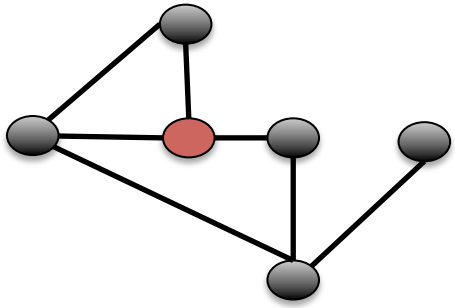
Generic:

```
for each cell c
    c = sum(neighbors(c))
```

Irregular

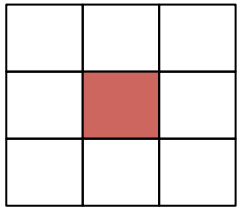
```
for x = 1 to n
  for neigh = 1 to numNeighs(x)
    A(x) += A(neighbor(x, neigh))
```

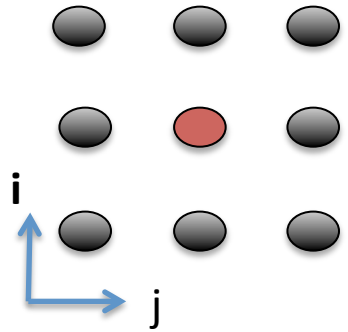




Regular

```
for i = 1 to n
  for j = 1 to m
    A(i,j) = A(i-1, j)   + A(i+1, j) +
             A(i,   j-1) + A(i,   j+1)
```





Impact of indirect access

```
int direct() {
  int sum = 0;

  for(int i = 0;
      i < DATA_SIZE; i++)
  {
    sum += A[i];
  }

  return sum;
}
```

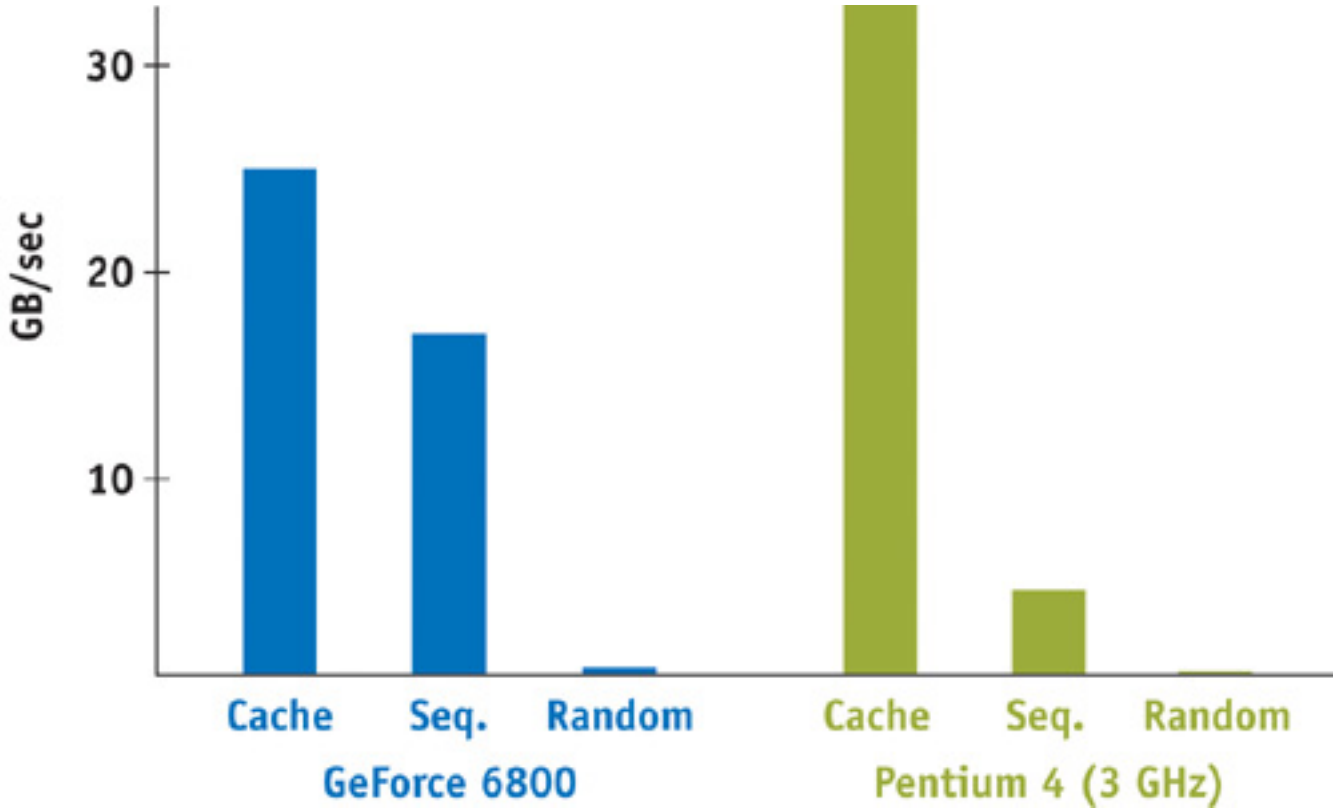
3.47 secs

```
int indirect() {
  int sum = 0;

  for(int i = 0;
      i < DATA_SIZE; i++)
  {
    sum += A[B[i]];
  }

  return sum;
}
```

38.41 secs

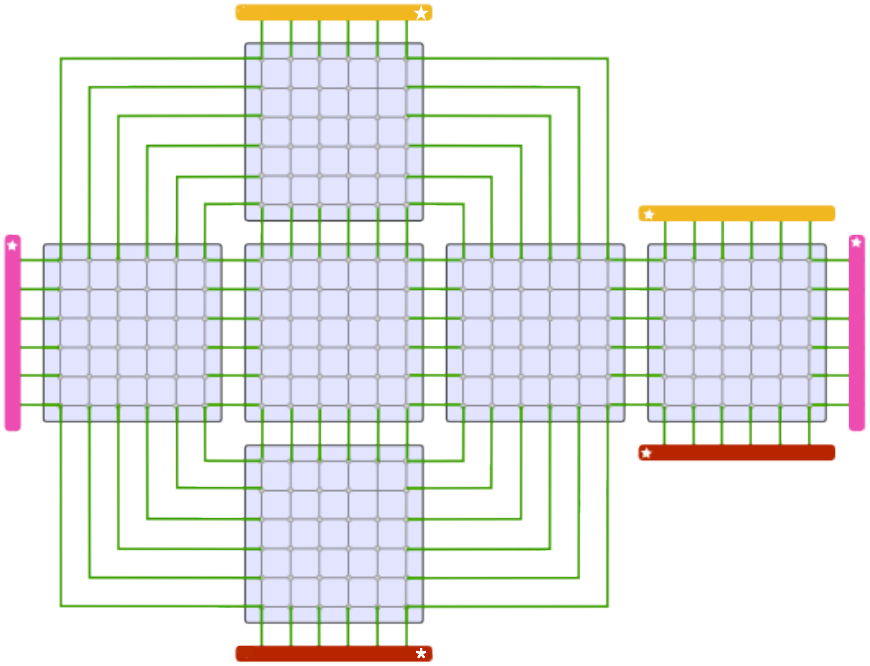
with DATA_SIZE = 500,000,000 (approx. 1.8 GB)
2.3 GHz Core2Quad machine with 4GB memory

Penalty for Irregular Accesses



Source: Ian Buck, Nvidia

Many Earth grids are not purely regular



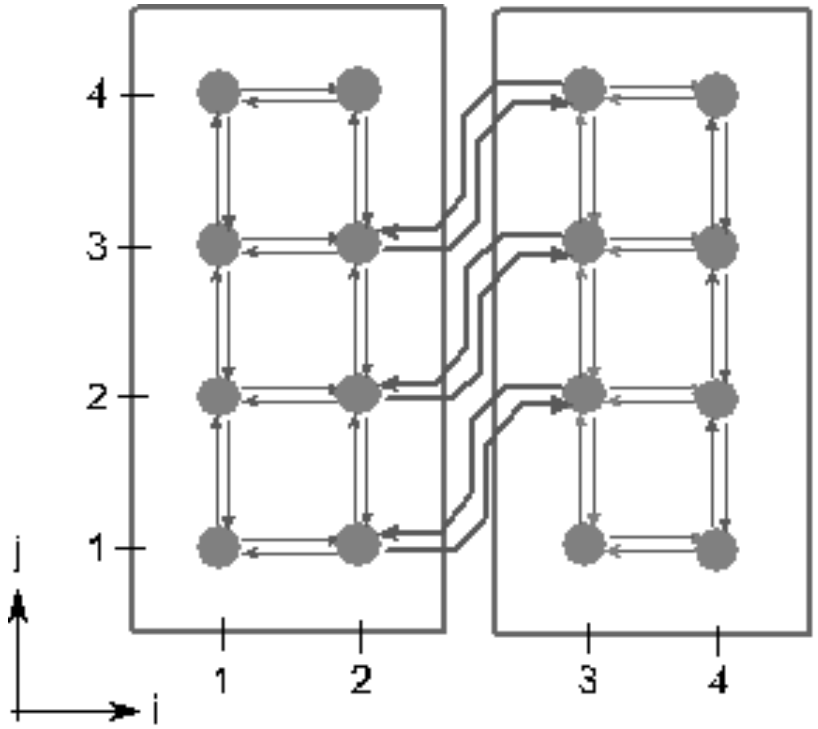
This is the structure of the cubed-sphere grid

Set of regular grids (arrays)

Connected to one another in an irregular fashion

Specialized communication code is needed to handle the irregular structure.

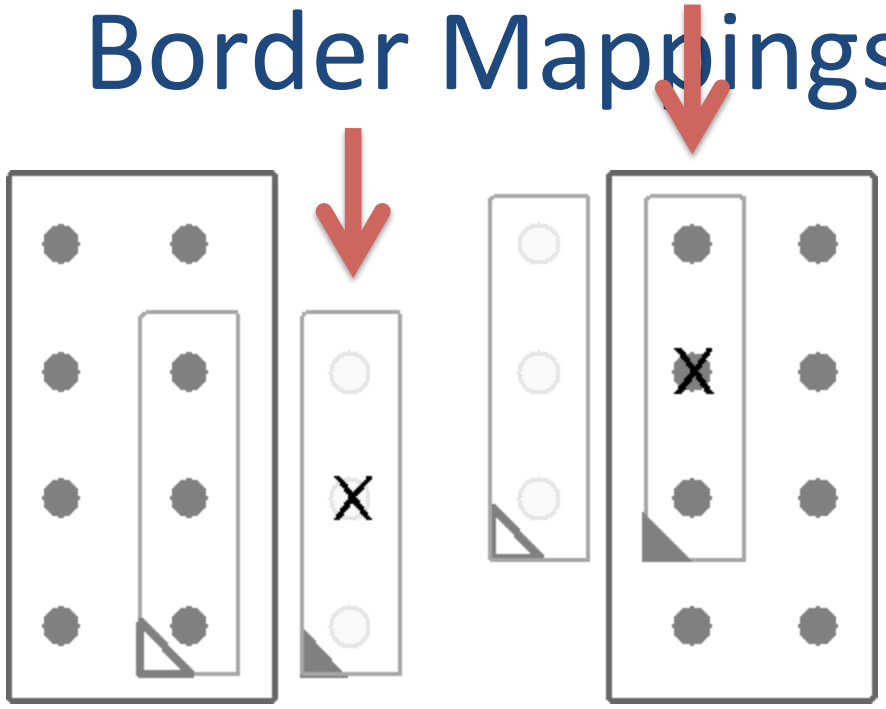
Subgrids and grids



sgA **sgB**

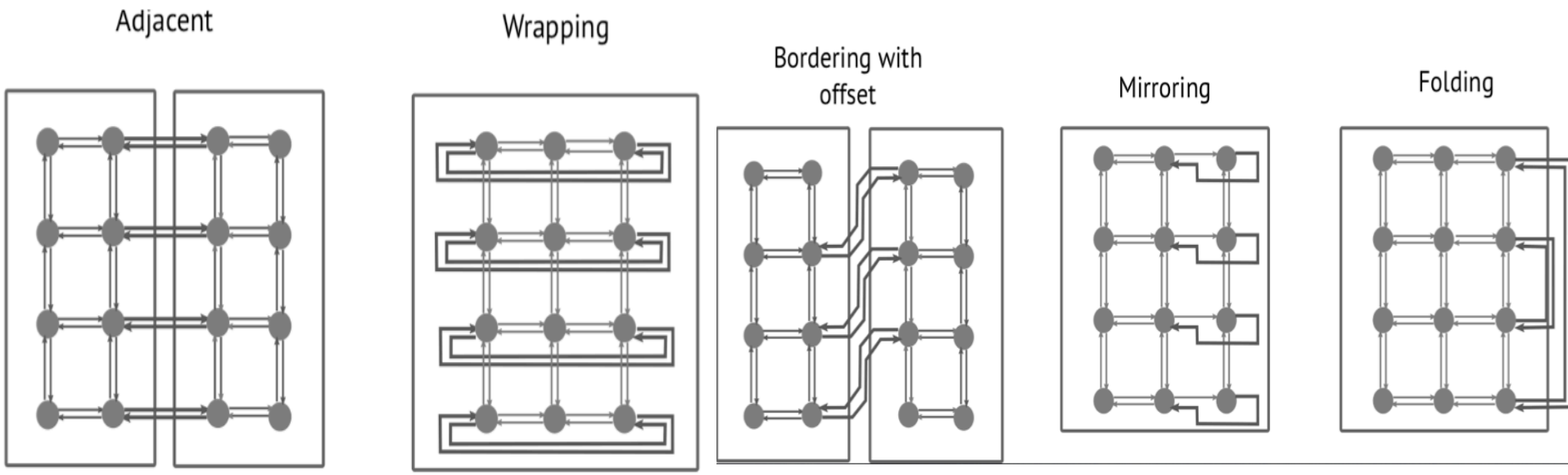
```
type(subgrid) :: sgA, sgB
type(Grid) :: g

sgA = subgrid_new(2, 4)
sgB = subgrid_new(2, 4)

g = grid_new()
call grid_addSubgrid(sgA)
call grid_addSubgrid(sgB)
```

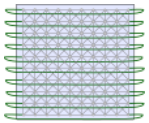
Border Mappings



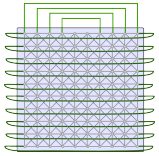

```
call grid_addBorderMap(3,1,   3,3,   sgA,
                       1,2,   1,4,   sgB)
call grid_addBorderMap(0,2,   0,4,   sgB,
                       2,1,   2,3,   sgA)
```

Connectivity between subgrids

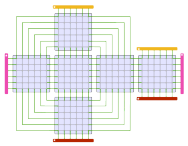




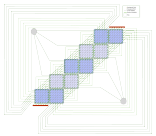
Dipole: Wrapping



Tripole: Wrapping, folding

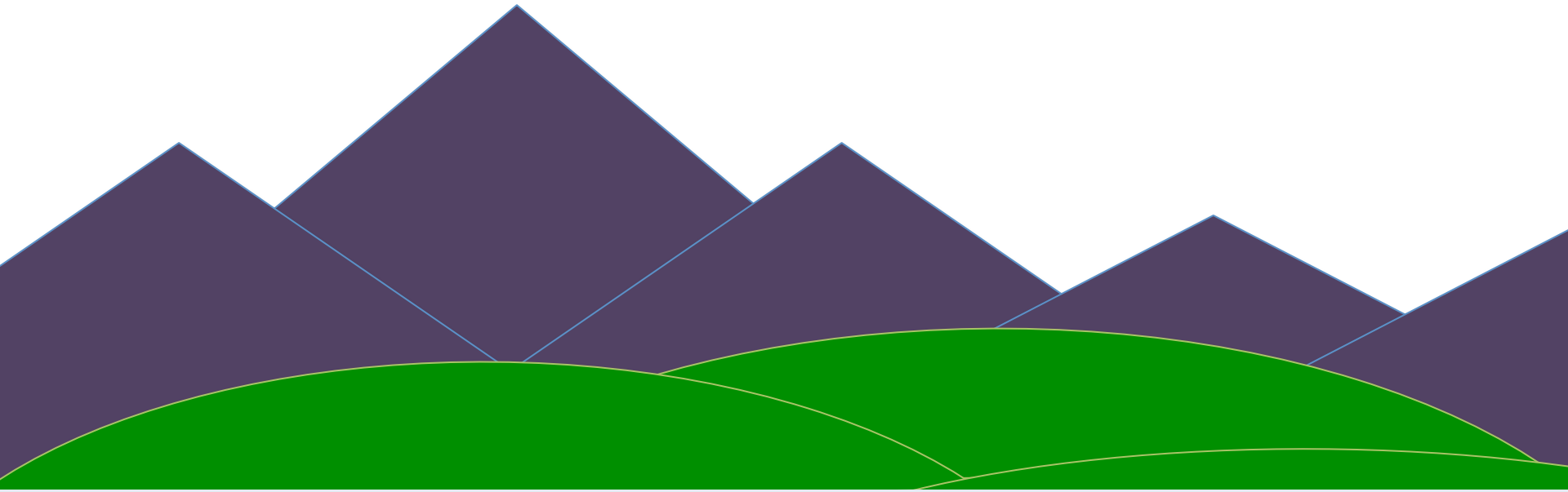


Cubed Sphere: Adjacent



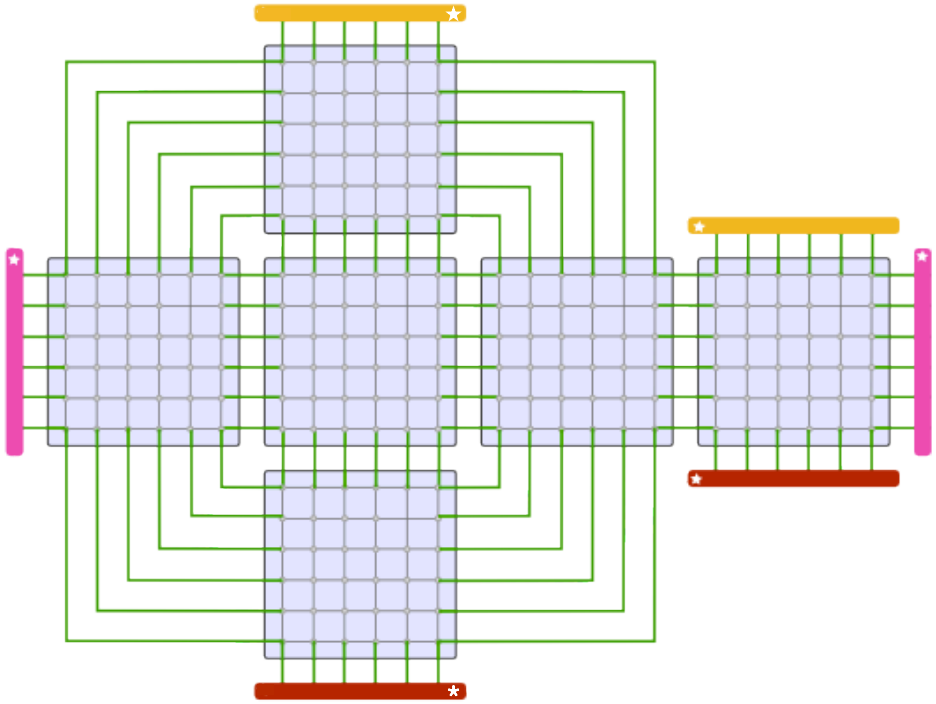
Icosahedral: Adjacent, offset

Parallelization



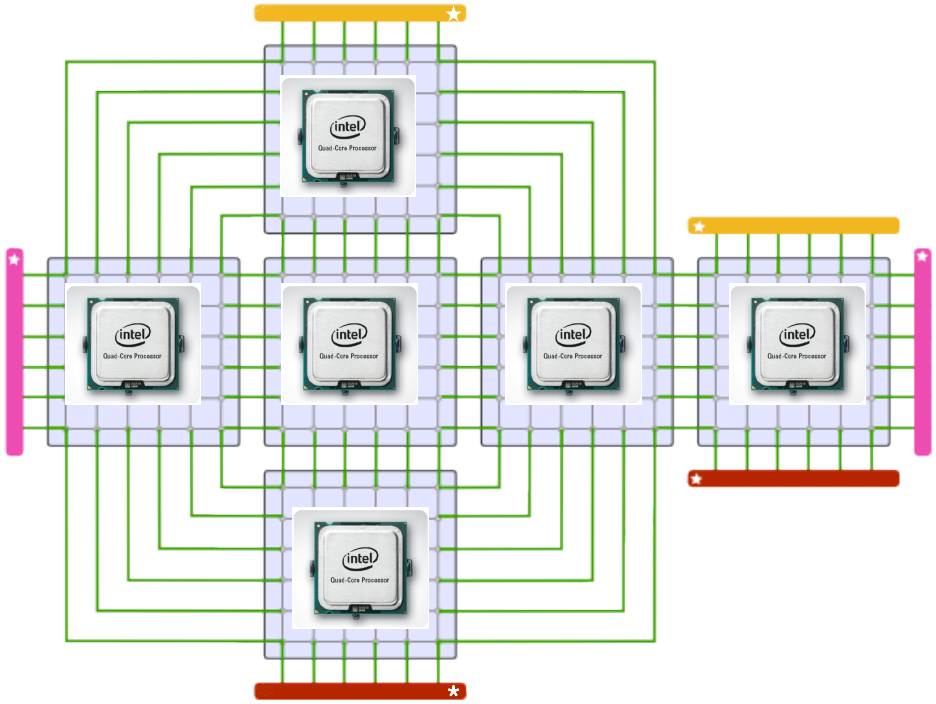
Domain decomposition for parallelization

How to map data and computation onto processors?



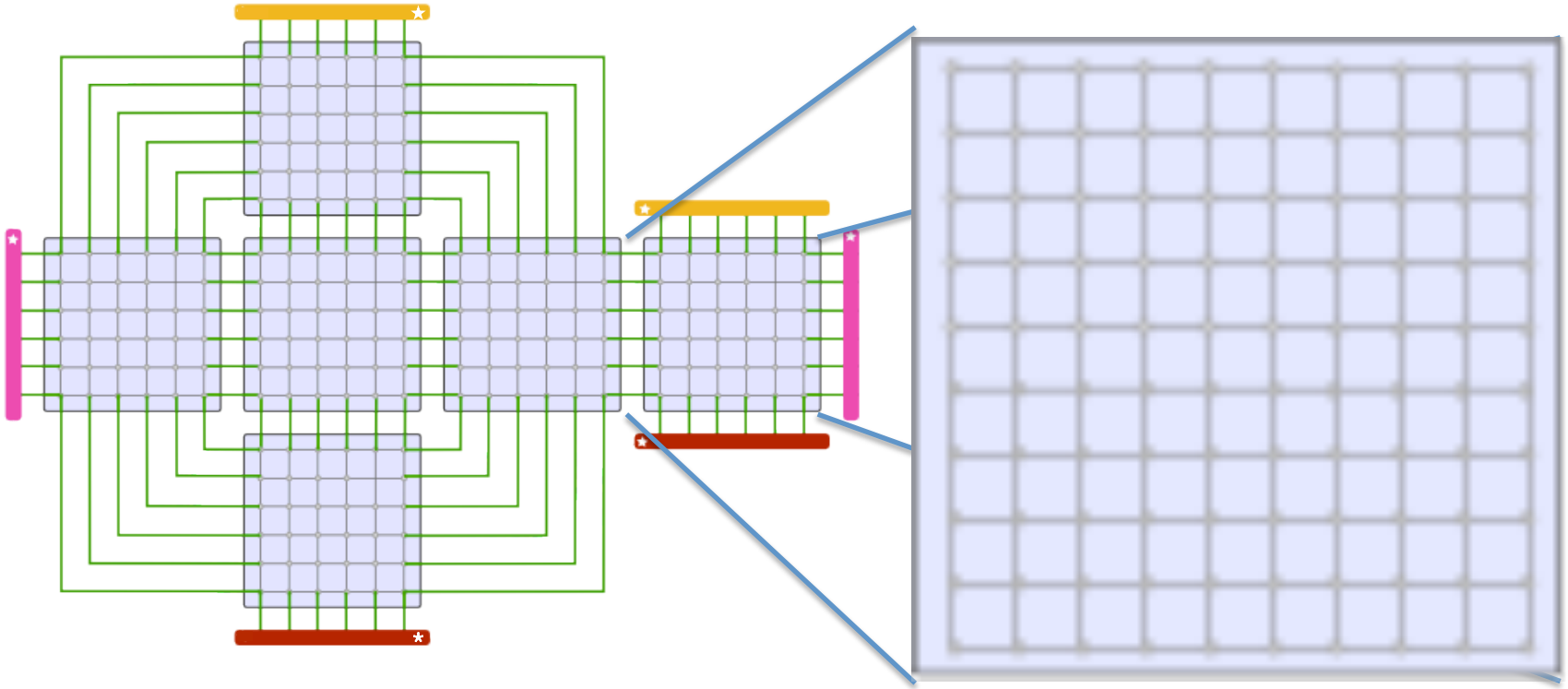
Domain decomposition for parallelization

How to map data and computation onto processors?



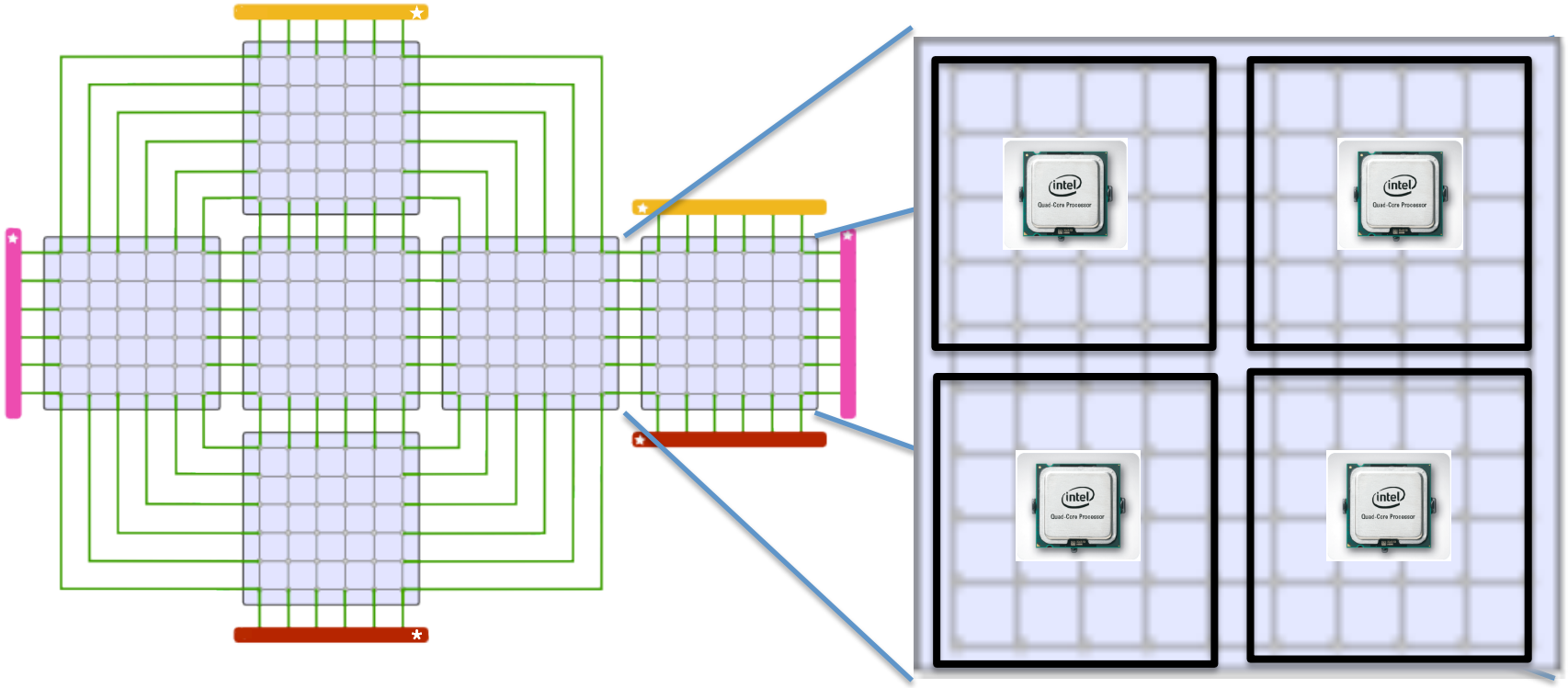
Domain decomposition for parallelization

How to map data and computation onto processors?

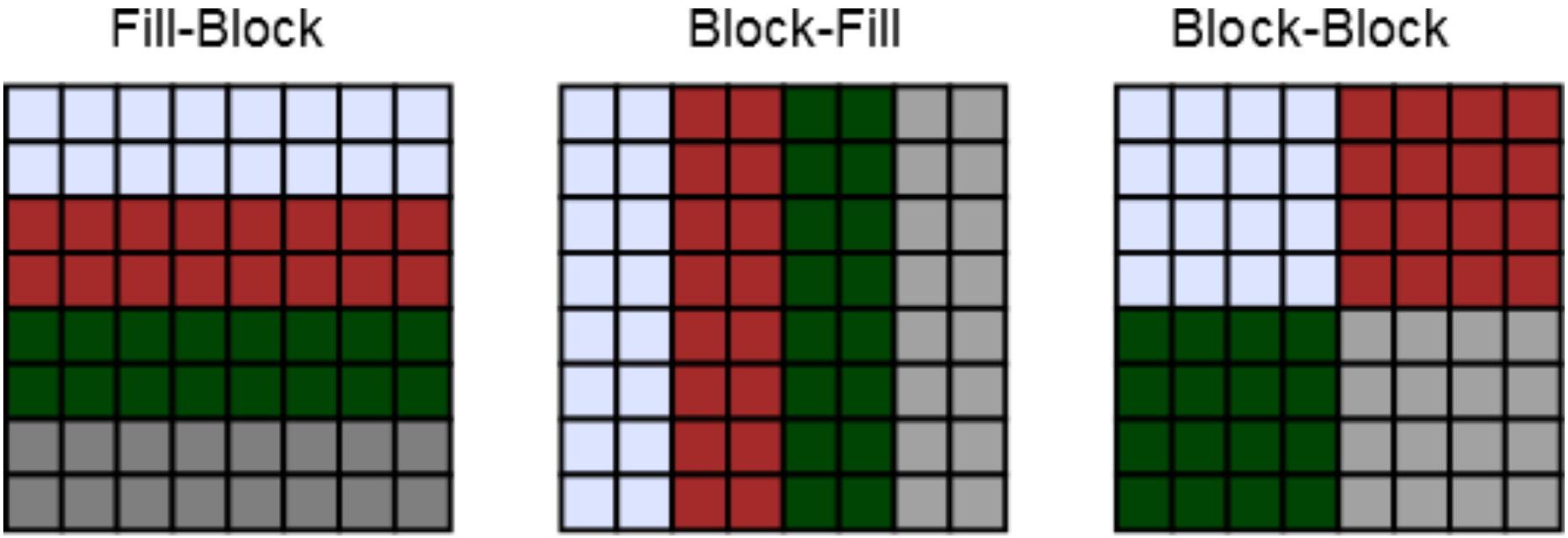


Domain decomposition for parallelization

How to map data and computation onto processors?



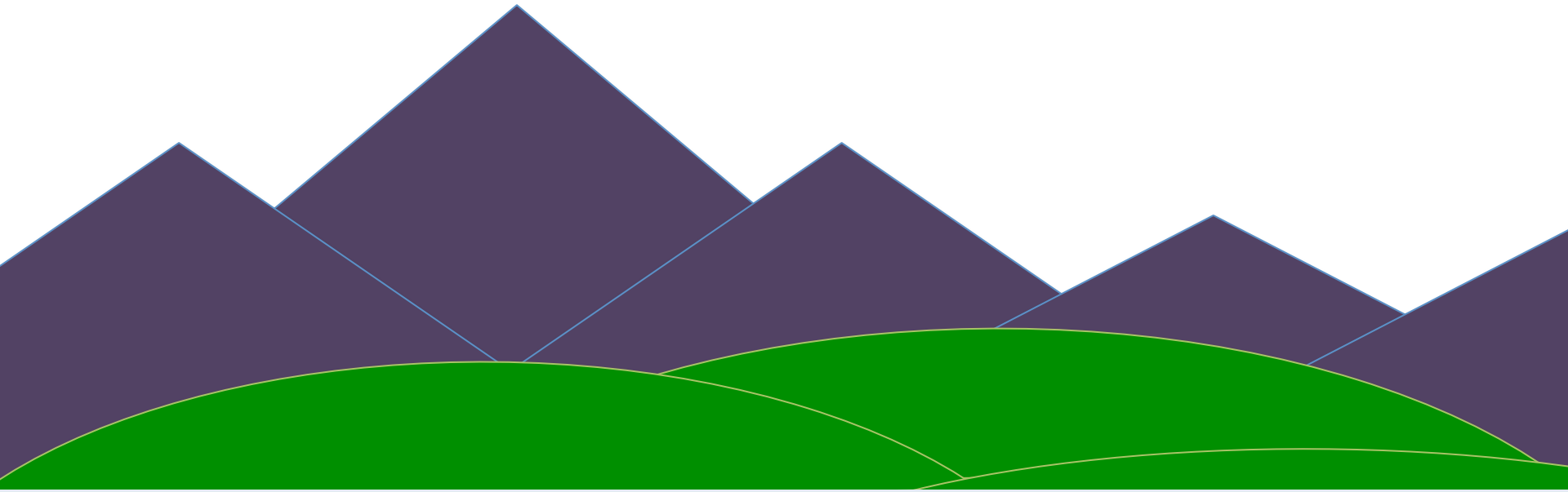
Distributions



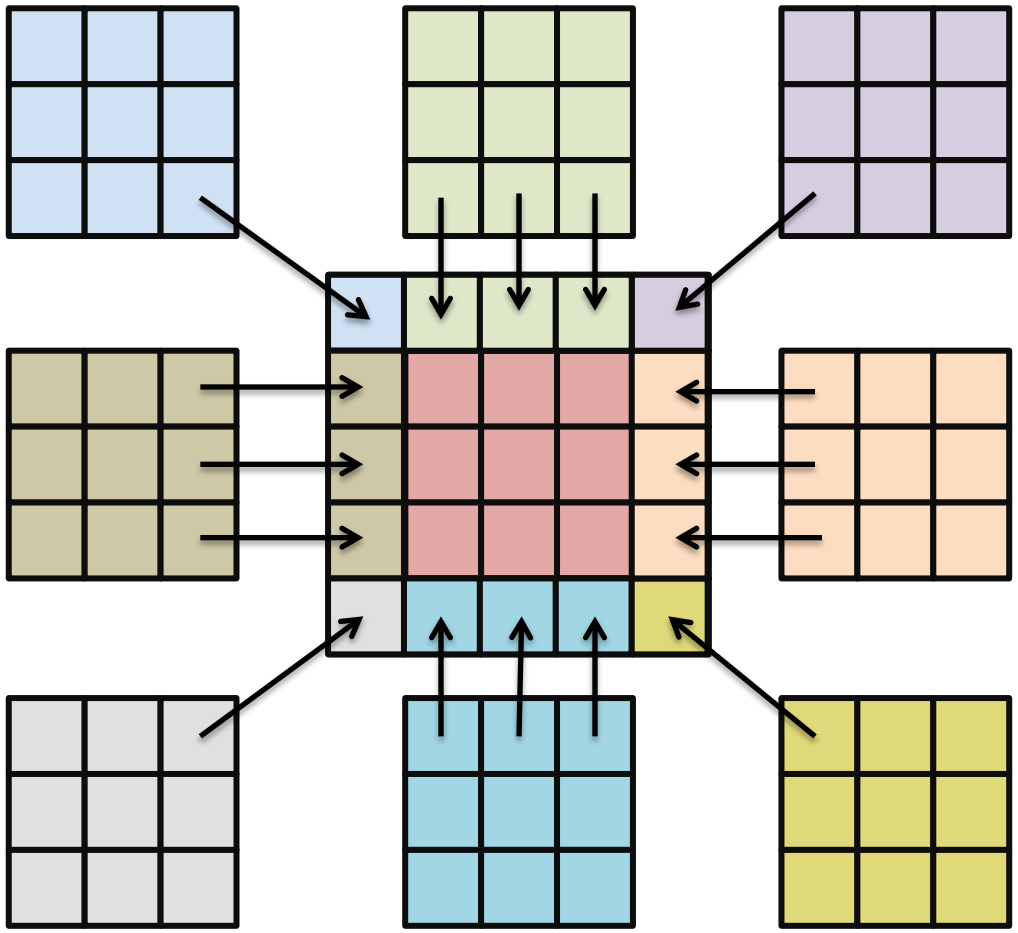
Example:

```
call grid_distributeFillBlock(g, 2)
```

Planning communication

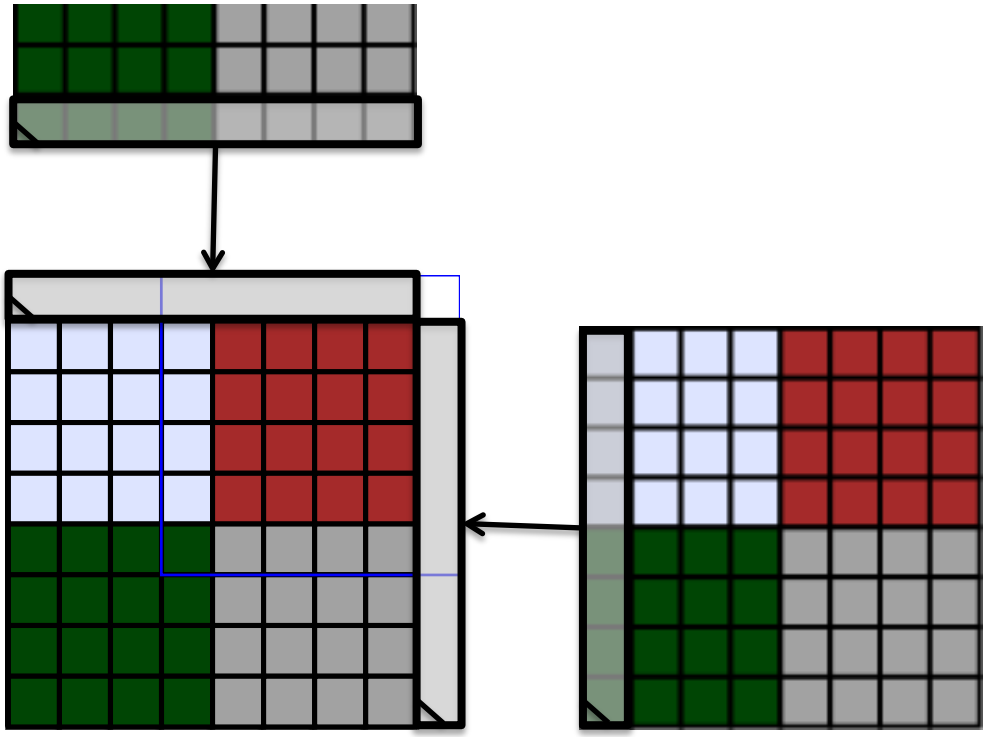


Halos around blocks

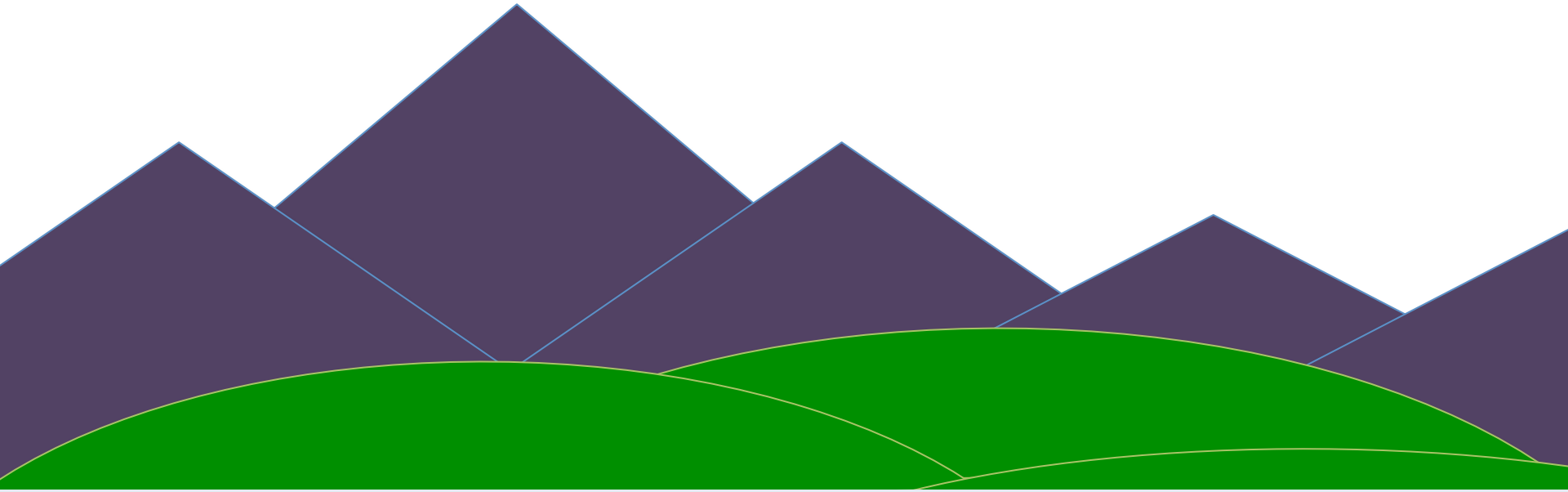


Communication plan abstraction

```
int  nMsgsRecv
int*  msgRecvFrom                    // msgID -> pid
int*  mTransferRecvAtLBID            // msgID,  tid -> lbid
Region**  mTransferRegionRecv // msgID, tid -> rgn
// Analogous structures exist for sending side
```



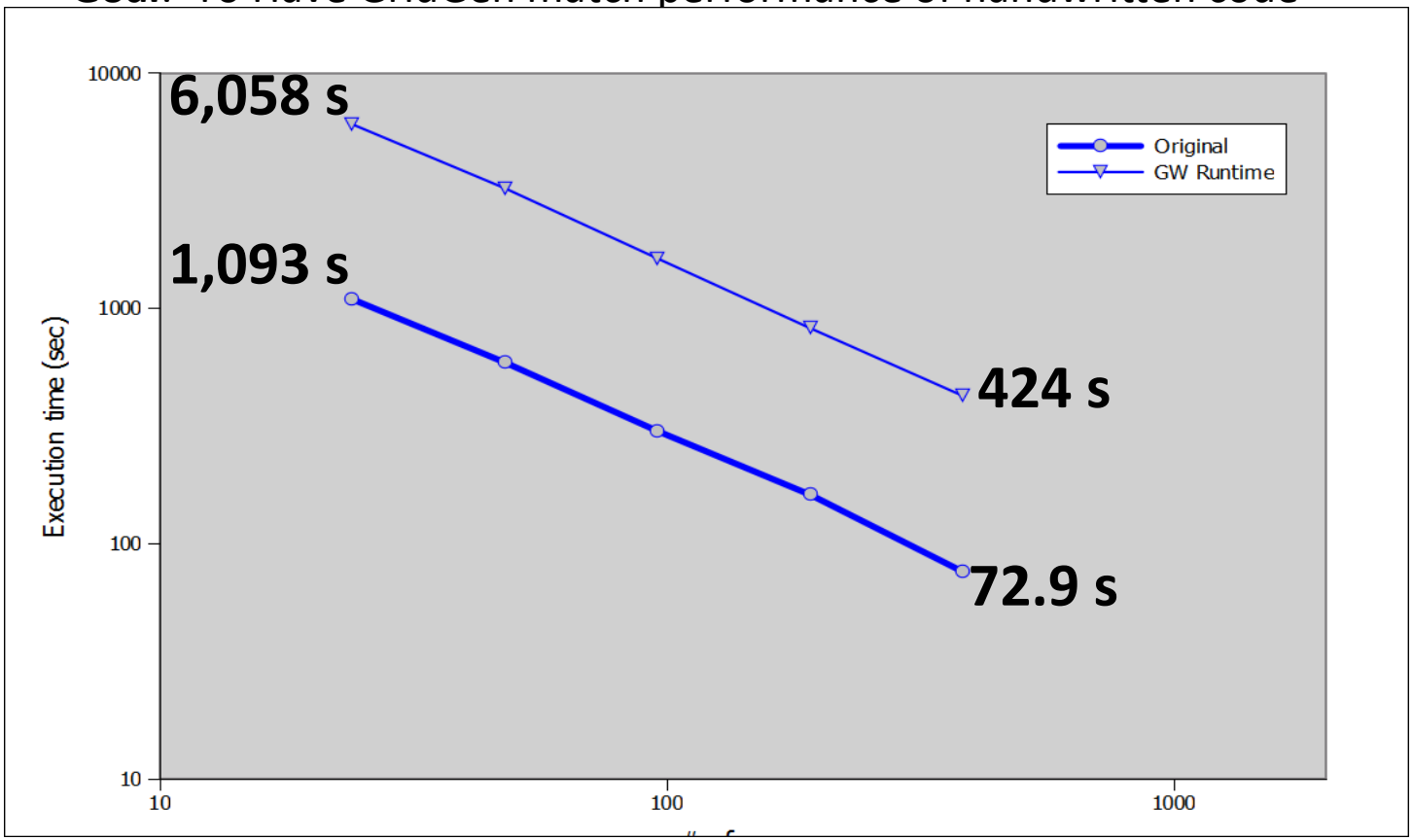
Problems with Performance



Impact of code generation

Implementation of stencil from CGPOP on a dipole grid

Goal: To Have GridGen match performance of handwritten code



Cray XT6m with 1248 cores across 52 compute nodes,
24 cores per compute node
2260 iterations on a10800 x 10800 grid

Cause of overhead

```
STENCIL(fivePt, A, i, j)
   fivePt = 0.2 *                    &
      (A(i, j)    + A(i-1, j) + &
       A(i+1, j) + A(i, j-1) + A(i, j+1))
end function
```

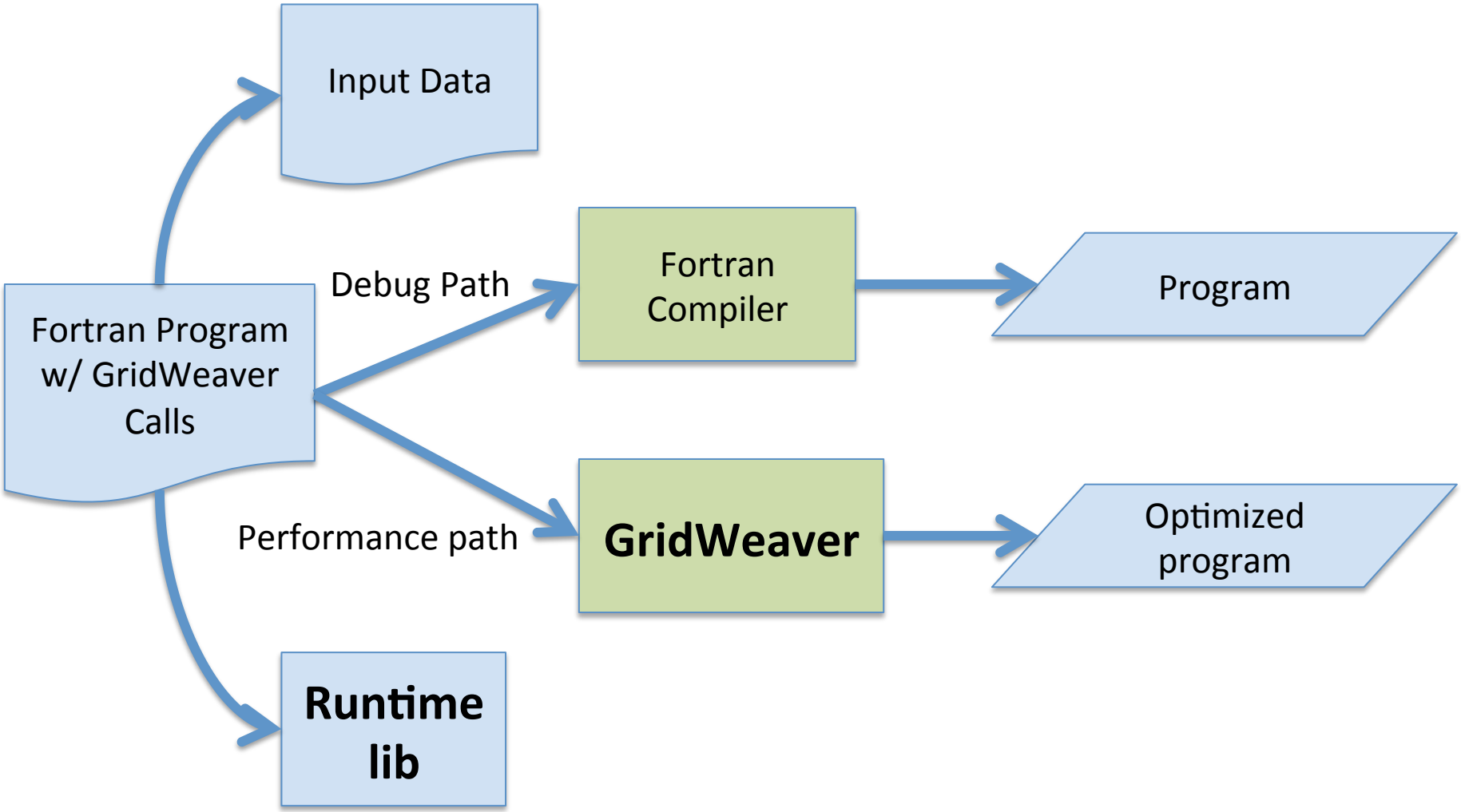
Cause of overhead

```
function fivePt(A, i, j)
  integer, intent(in) :: i, j

  interface
    integer function A(x,y)
      integer, intent(in) :: x,y
    end function
  end interface

  fivePt = 0.2 *                    &
    (A(i, j)   + A(i-1, j) + &
     A(i+1, j) + A(i, j-1) + A(i, j+1))
end function
```

Called once per grid node

Imposters of arrays

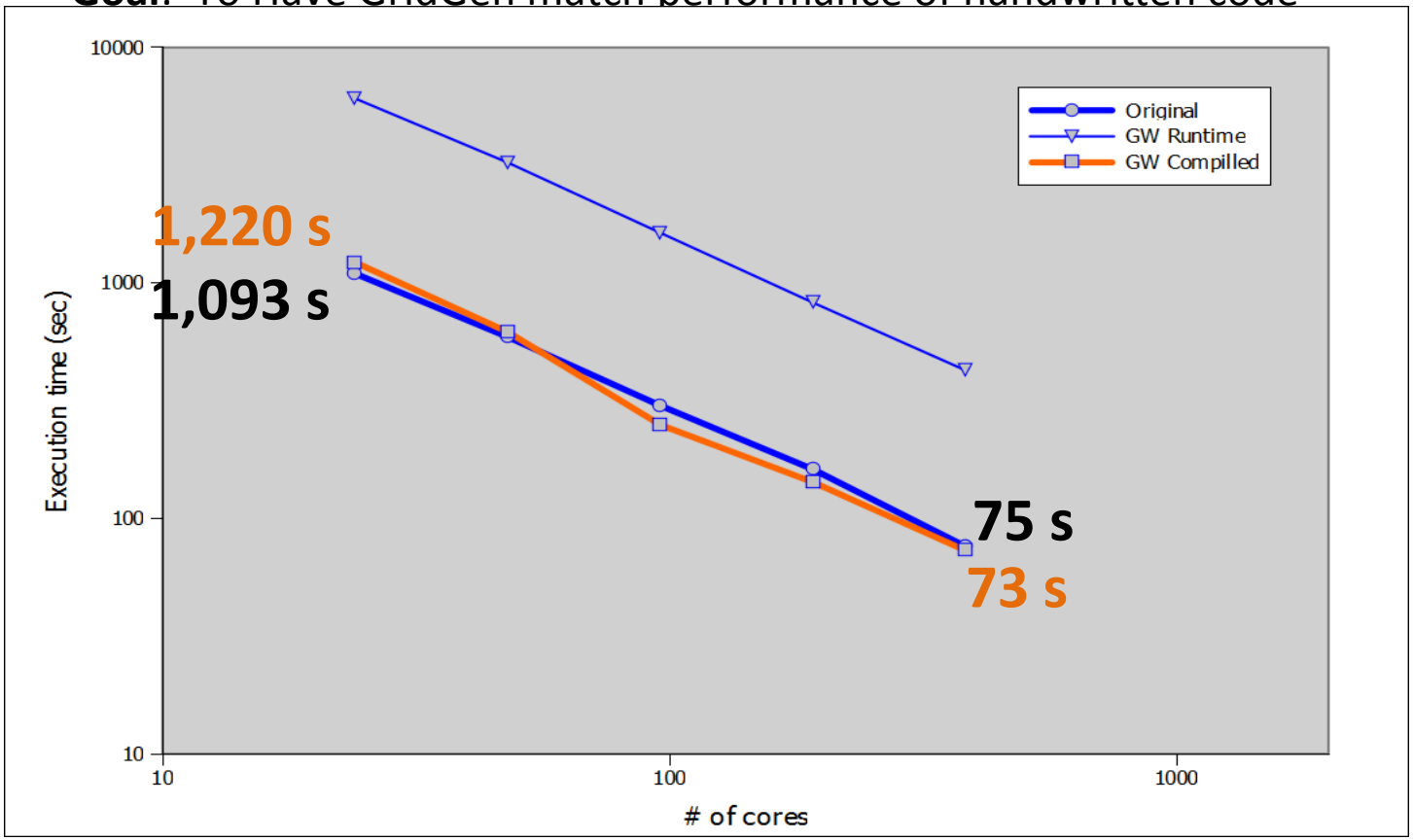
Code generation tool



Impact of code generation

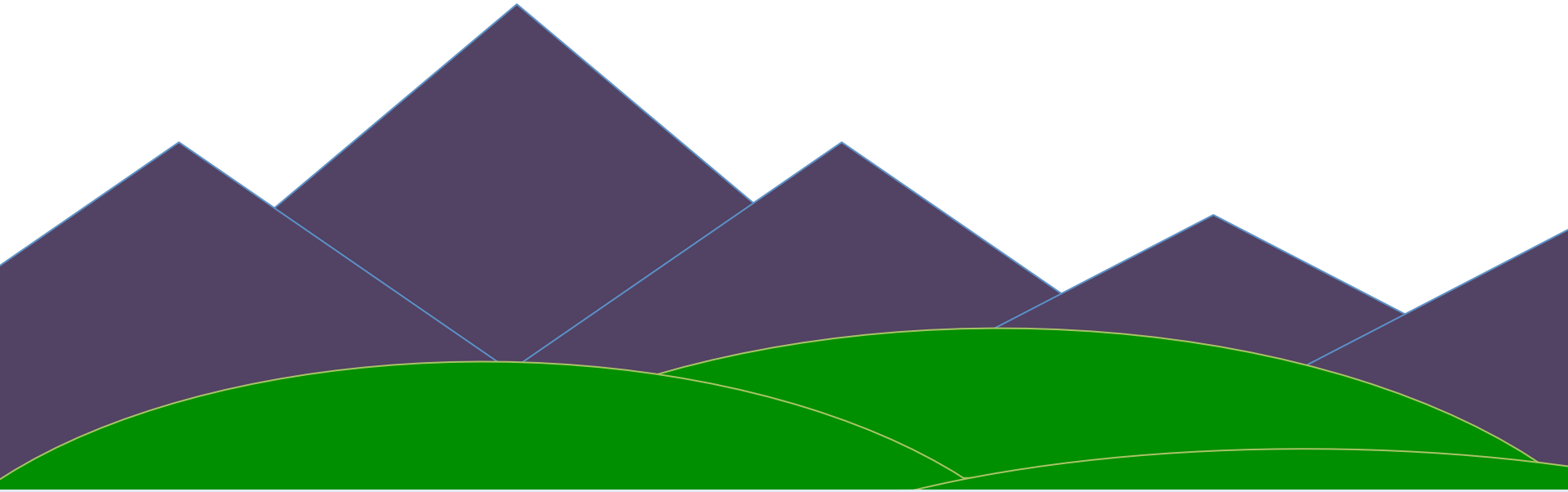
Implementation of stencil from CGPOP on a dipole grid

Goal: To Have GridGen match performance of handwritten code



Cray XT6m with 1248 cores across 52 compute nodes,
24 cores per compute node
2260 iterations on a10800 x 10800 grid

Conclusions



Talk Summary

Project goal:

To enable a **separation of concerns** for semiregular grid computations between **stencil** algorithm and **grid structure** and **parallelization** and **decomposition** details.

Approach:

Implementing grid, subgrid, border-mapping, distribution abstractions and automatically generating communication schedules to populate halos

Project goal:

Match performance of existing semi-regular stencil computations.

Approach:

Use code generator to inline library calls.