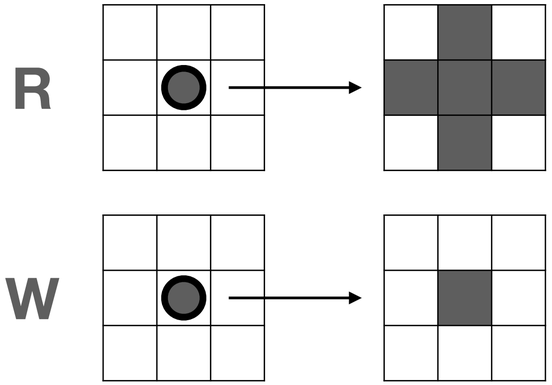
Automated Partitioning of Data-Parallel Kernels using Polyhedral Compilation

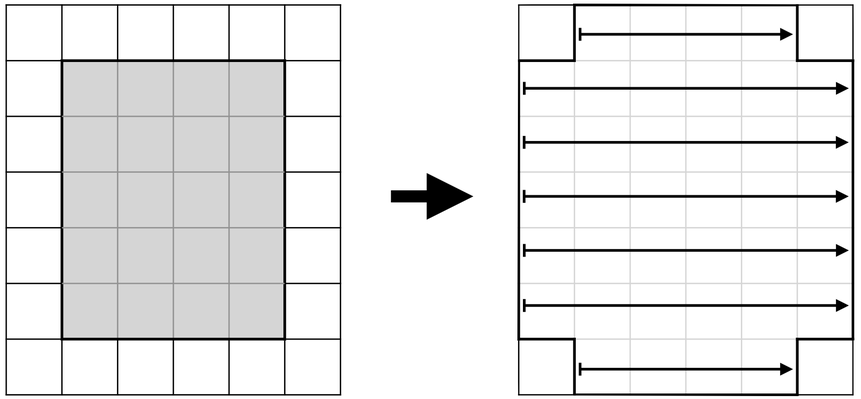
1) Kernel Code

```
__global__ void stencil(float* A, float* B, int N) {
  idx = threadIdx.x + blockIdx.x * blockDim.x;
  idy = threadIdx.y + blockIdx.y * blockDim.y;
  if (idx >= N || idy >= N) return;
  auto x1 = idx.x-1 >= 0 ? A[idx.y * N + idx.x-1] : 0;
  auto x2 = idx.x+1 < N ? A[idy.y * N + idx.x+1] : 0;
  auto x3 = idx.y-1 >= 0 ? A[(idx.y-1) * N + idx.x] : 0;
  auto x4 = idx.y+1 < N ? A[(idx.y+1)* N + idx.x] : 0;
  B[idx.y * N + idx.x] = 0.25*(x1+x2+x3+x4);
}
```

2) Memory Access Patterns



3) Polyhedral Code Generation



4) Host Transformation

```
int main() {
  ...
  kernel<<<threadGrid, threadBlocks>>>(arr_in, arr_out, N);
  ...
}
```

```
int main() {
  ...
  for (int 0; i < numGPUs(); ++i) {
    auto newGrid = calcGrid(threadGrid);
    redistribute_data(arr_in, newGrid);
    kernel<<<newGrid, threadBlocks>>>(arr_in, arr_out, N);
    update_distribution(arr_out, newGrid);
  }
  ...
}
```

5) Data Distribution