OPENCL

Episode 6 - Shared Memory Kernel Optimization

David W. Gohara, Ph.D.
Center for Computational Biology
Washington University School of Medicine, St. Louis
email: sdg0919@gmail.com

THANK YOU

MACCIUS™
Application Hosting. Managed Services.

MacResearch
http://www.macresearch.org

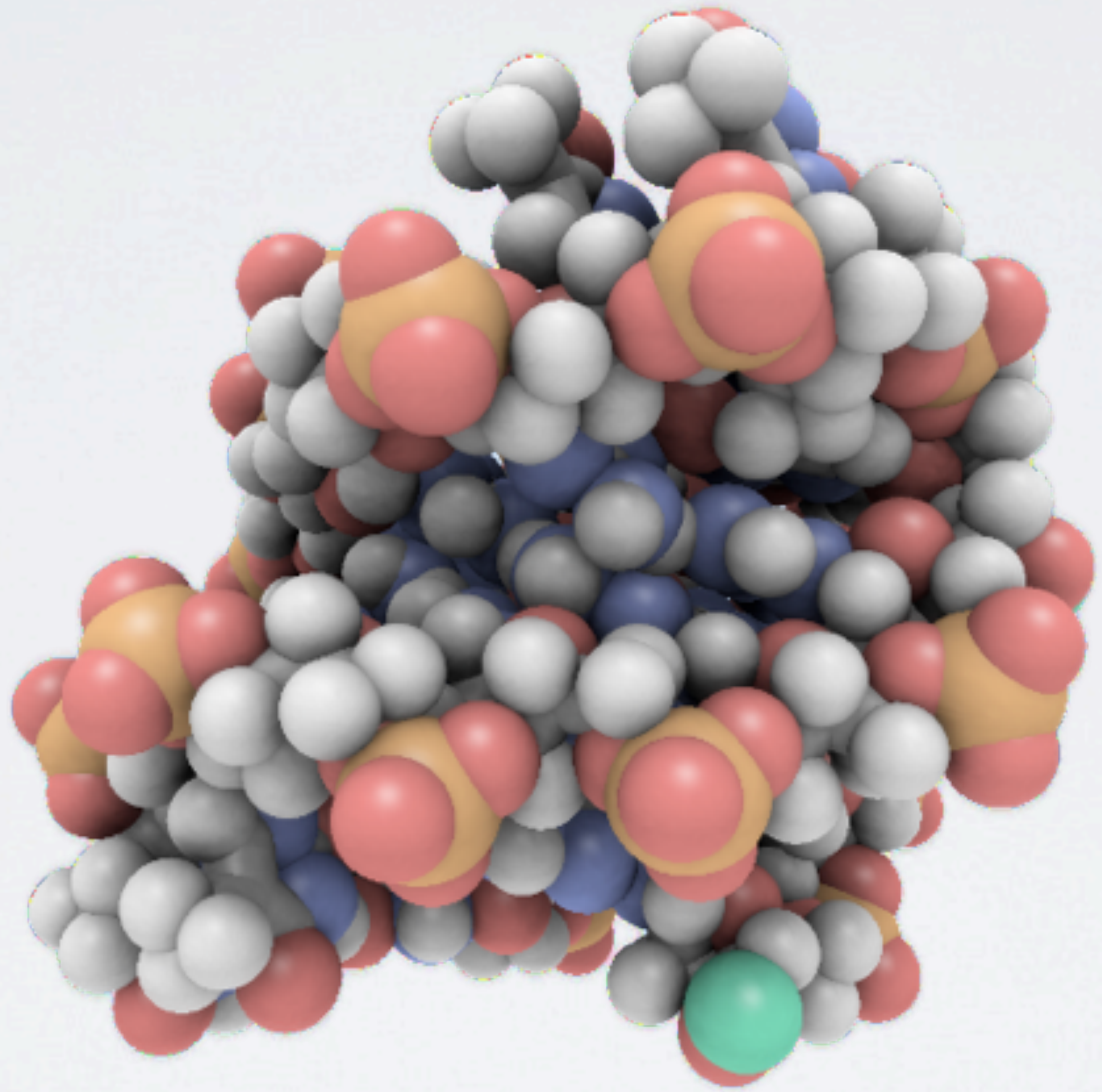
SHARED MEMORY OPTIMIZATION

- Xcode project derived from real-world code
- Use of shared memory to increase performance
  - Commonly accessed data
- Use of synchronization points

CALCULATION

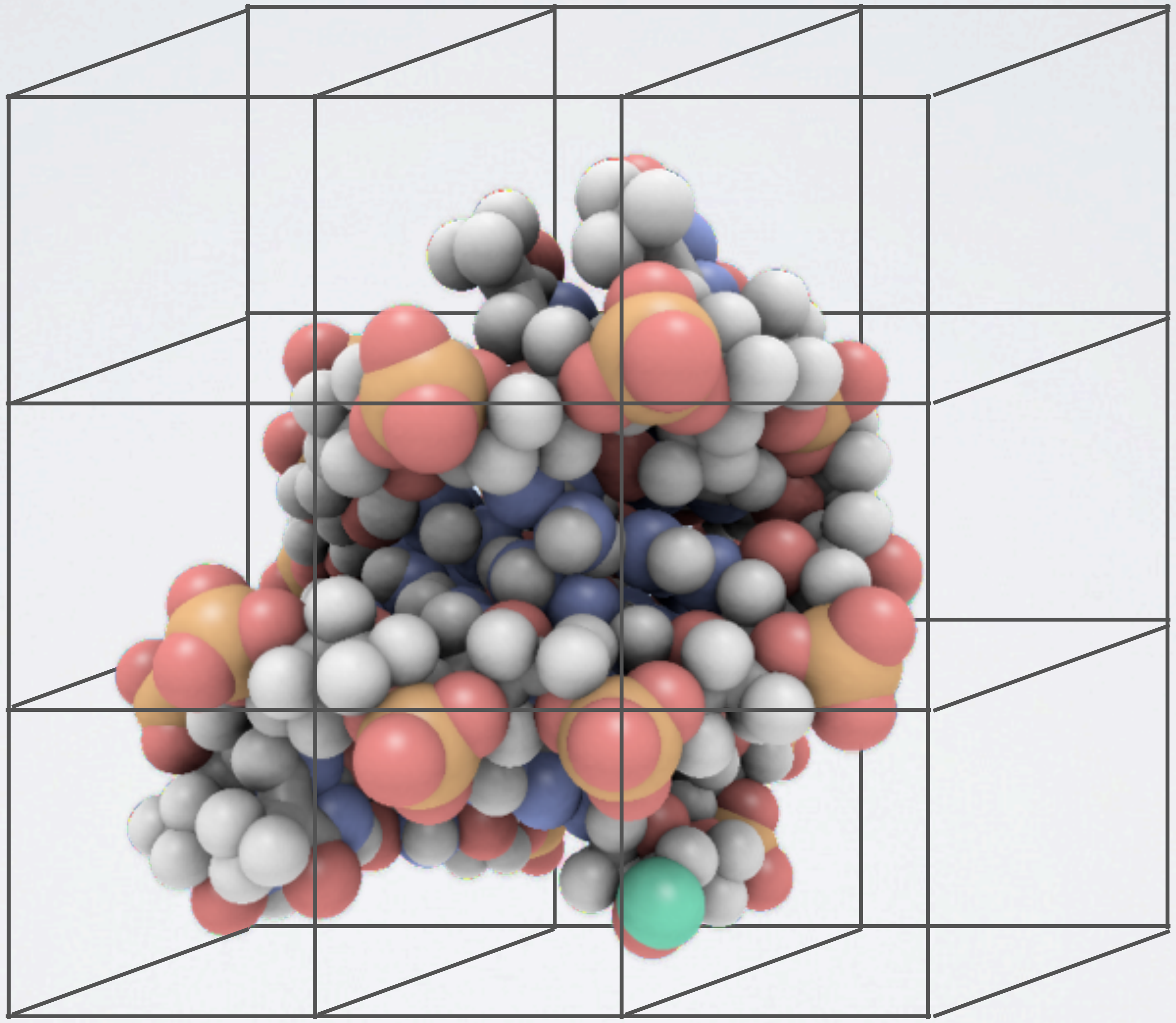
- Boundary value setup of a discretized problem on a grid
- Calculation performed over all "atoms" in a model for each grid point
- CPU vs. GPU

BOUNDARY VALUE PROBLEM



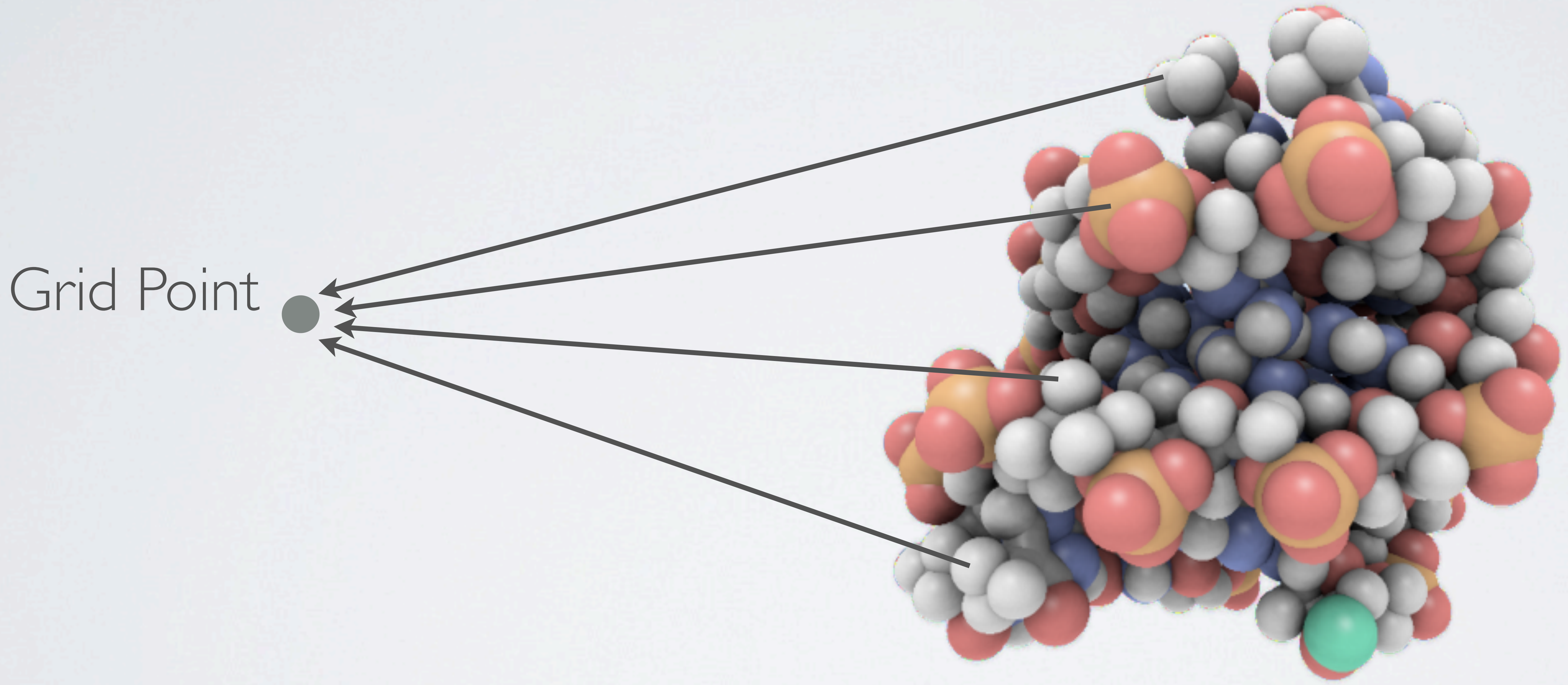
Rendered with vAPBS

BOUNDARY VALUE PROBLEM



Rendered with vAPBS

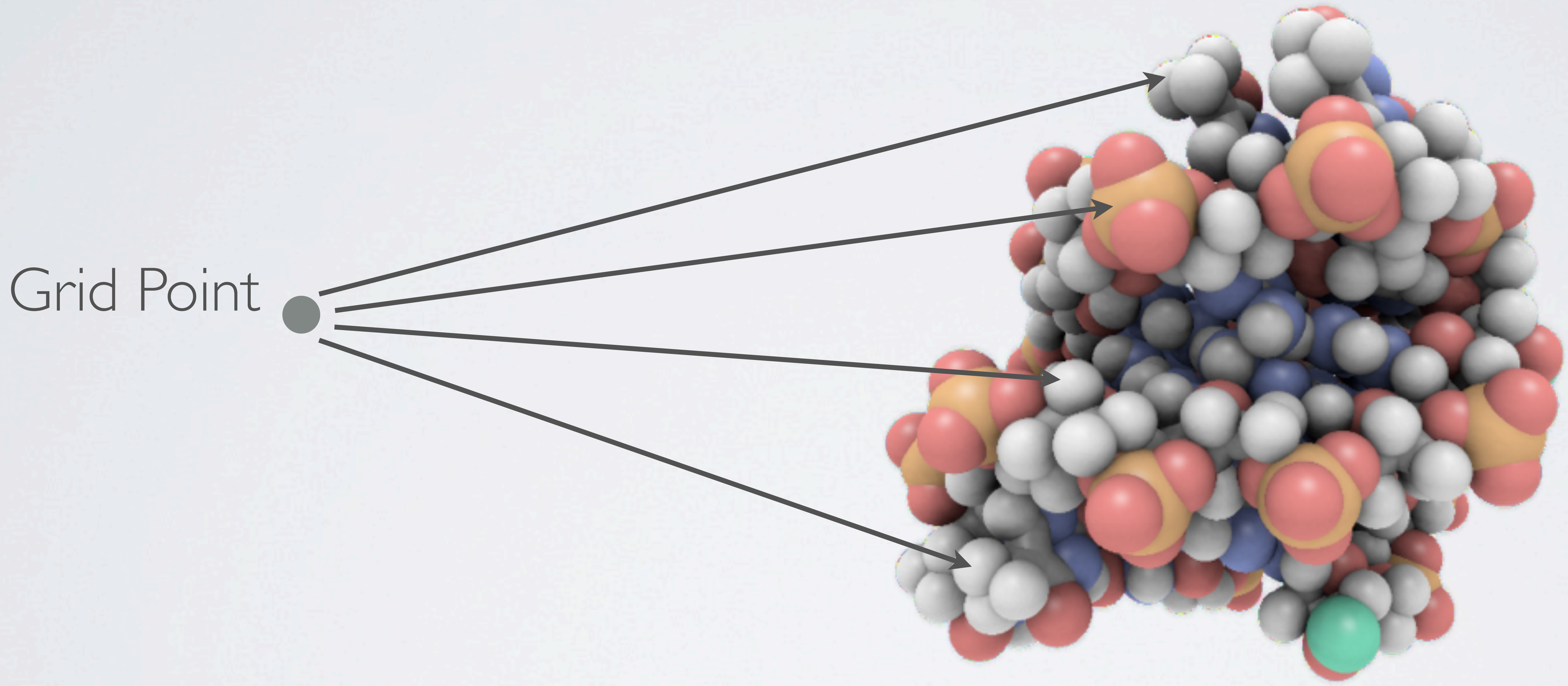
BOUNDARY VALUE PROBLEM



Atom-centric - Requires Locks or Reductions

Rendered with vAPBS

BOUNDARY VALUE PROBLEM



Grid-centric - No Locks or Reductions

Rendered with vAPBS

CPU CODE

```
for(igrid=0;igrid<ngrid;igrid++){
     for(iatom=0; iatom<natom; iatom++){
        dist = sqrtf((gx[igrid]-ax[iatom])*(gx[igrid]-ax[iatom]) +
                     (gy[igrid]-ay[iatom])*(gy[igrid]-ay[iatom]) +
                     (gz[igrid]-az[iatom])*(gz[igrid]-az[iatom]));

        val[igrid] += pre1*(charge[iatom]/dist) *
                             expf(-xkappa*(dist-size[iatom])) /
                            (1+xkappa*size[iatom]);
     }
  }
```

GPU CODE - CORE

NDRange = ngrid

```
for(igrid=0;igrid<ngrid;igrid++){
      for(iatom=0; iatom<natom; iatom++){
         dist = sqrtf((gx[igrid]-ax[iatom])*(gx[igrid]-ax[iatom]) +
                     (gy[igrid]-ay[iatom])*(gy[igrid]-ay[iatom]) +
                     (gz[igrid]-az[iatom])*(gz[igrid]-az[iatom]));

         val[igrid] += pre1*(charge[iatom]/dist) *
                              expf(-xkappa*(dist-size[iatom])) /
                             (1+xkappa*size[iatom]);
      }
  }
```

GPU CODE - CORE

```
for(iatom=0; iatom<natom; iatom++){
   dist = sqrtf((gx[igrid]-ax[iatom])*(gx[igrid]-ax[iatom]) +
             (gy[igrid]-ay[iatom])*(gy[igrid]-ay[iatom]) +
             (gz[igrid]-az[iatom])*(gz[igrid]-az[iatom]));

   val[igrid] += pre1*(charge[iatom]/dist) *
                     expf(-xkappa*(dist-size[iatom])) /
                    (1+xkappa*size[iatom]);
}
```

GPU CODE - UNOPTIMIZED

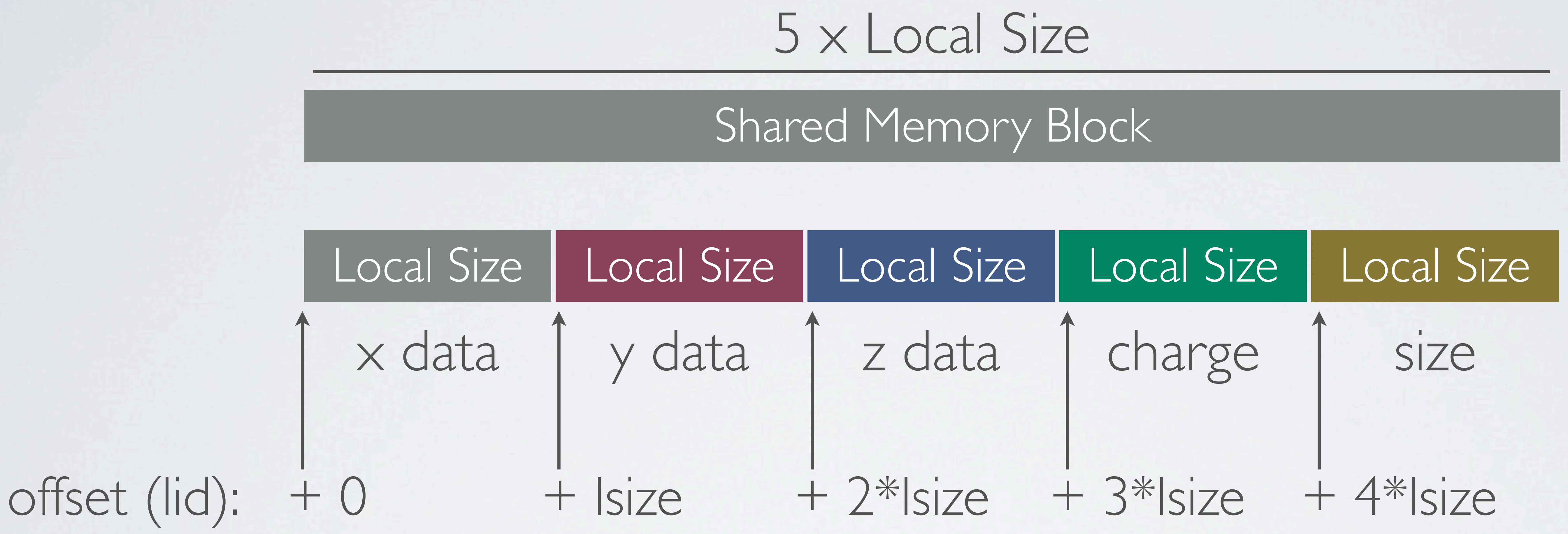
```
int igrid = get_global_id(0);
int iatom;
float v = 0.0f;

float lgx = gx[igrid];
float lgy = gy[igrid];
float lgz = gz[igrid];

for( iatom = 0; iatom < natoms; iatom++ )
{
     float dx = lgx - ax[iatom];
     float dy = lgy - ay[iatom];
     float dz = lgz - az[iatom];

     float dist = sqrt( dx * dx + dy * dy + dz * dz );
     v += pre1 * ( charge[iatom] / dist )  *
          exp( -xkappa * (dist - size[iatom])) /
          (1.0f + xkappa * size[iatom]);
}
val[ igrid ] = v;
```

SHARED MEMORY USAGE



lid = local id

lsize = local size

MacResearch
http://www.macresearch.org

GPU CODE - OPTIMIZED

```
for( iatom = 0; iatom < natoms; iatom+=lsize )
{
     if((iatom+lsize) > natoms)
         lsize = natoms - iatom;

     if((iatom + lid) < natoms){
         shared[lid]              = ax[iatom + lid];
         shared[lid + lsize]      = ay[iatom + lid];
         shared[lid + 2*lsize]    = az[iatom + lid];
         shared[lid + 3*lsize]    = charge[iatom + lid];
         shared[lid + 4*lsize]    = size[iatom + lid];
     }
     barrier(CLK_LOCAL_MEM_FENCE);

     for(int i=0;i<lsize;i++){
          float dx = lgx - shared[i];
          float dy = lgy - shared[i + lsize];
          float dz = lgz - shared[i + 2*lsize];

          float dist = sqrt( dx * dx + dy * dy + dz * dz );
          v += pre1 * ( shared[i + 3*lsize] / dist )  *
               exp( -xkappa * (dist - shared[i + 4*lsize])) /
               (1.0f + xkappa * shared[i + 4*lsize]);
     }
     barrier(CLK_LOCAL_MEM_FENCE);
}
```

GPU CODE - OPTIMIZED

```
for( iatom = 0; iatom < natoms; iatom+=lsize )
{
    if((iatom+lsize) > natoms)
       lsize = natoms - iatom;

    if((iatom + lid) < natoms){
       shared[lid]              = ax[iatom + lid];
       shared[lid + lsize]      = ay[iatom + lid];
       shared[lid + 2*lsize]    = az[iatom + lid];
       shared[lid + 3*lsize]    = charge[iatom + lid];
       shared[lid + 4*lsize]    = size[iatom + lid];
    }
    barrier(CLK_LOCAL_MEM_FENCE);

    for(int i=0;i<lsize;i++){
        float dx = lgx - shared[i];
        float dy = lgy - shared[i + lsize];
        float dz = lgz - shared[i + 2*lsize];

        float dist = sqrt( dx * dx + dy * dy + dz * dz );
        v += pre1 * ( shared[i + 3*lsize] / dist )  *
            exp( -xkappa * (dist - shared[i + 4*lsize])) /
            (1.0f + xkappa * shared[i + 4*lsize]);
    }
    barrier(CLK_LOCAL_MEM_FENCE);
}
```

MacResearch
http://www.macresearch.org

GPU CODE - OPTIMIZED

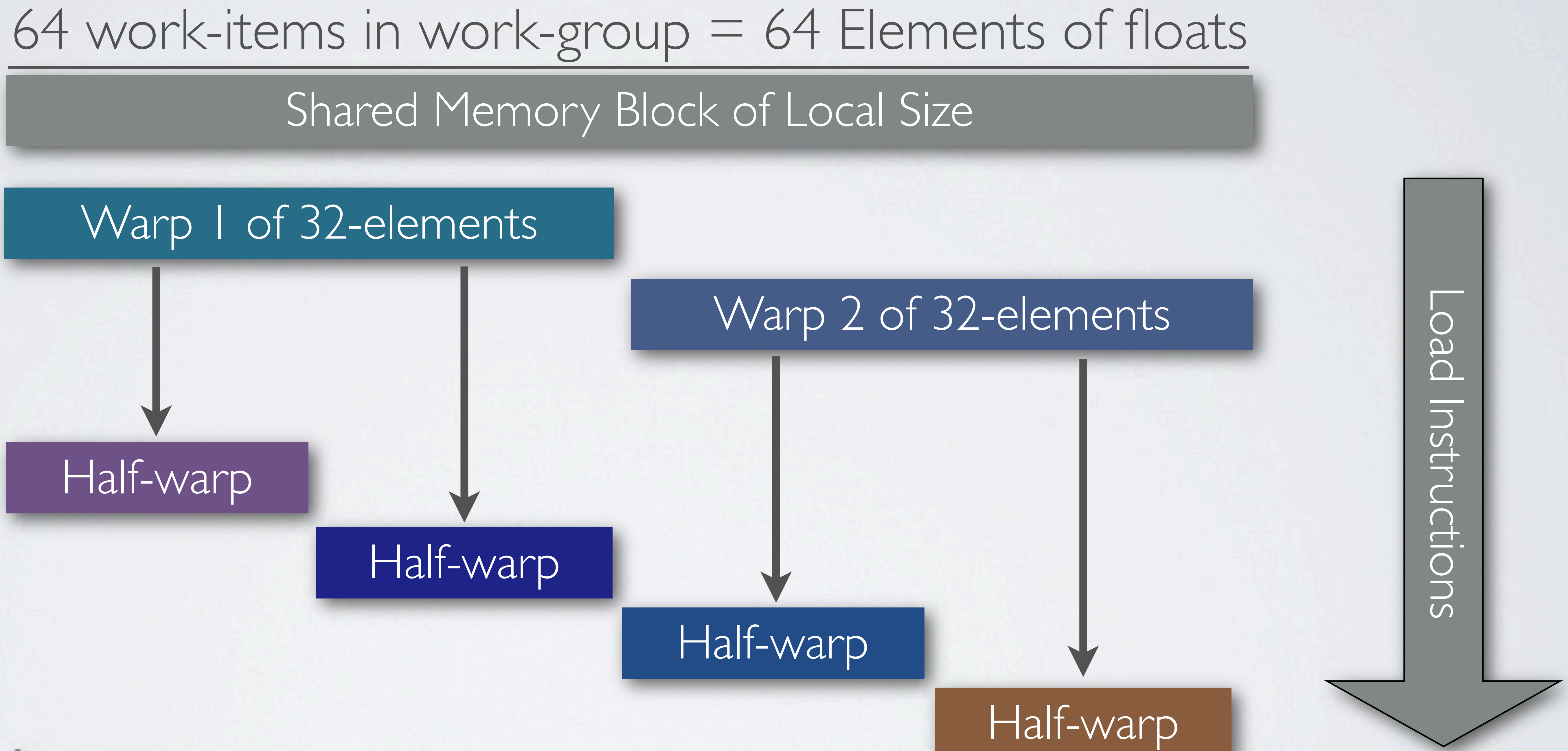
```
for( iatom = 0; iatom < natoms; iatom+=lsize )
{
     if((iatom+lsize) > natoms)
        lsize = natoms - iatom;

     if((iatom + lid) < natoms){
        shared[lid]                = ax[iatom + lid];
        shared[lid + lsize]        = ay[iatom + lid];
        shared[lid + 2*lsize]      = az[iatom + lid];
        shared[lid + 3*lsize]      = charge[iatom + lid];
        shared[lid + 4*lsize]      = size[iatom + lid];
     }
     barrier(CLK_LOCAL_MEM_FENCE);

     for(int i=0;i<lsize;i++){
         float dx = lgx - shared[i];
         float dy = lgy - shared[i + lsize];
         float dz = lgz - shared[i + 2*lsize];

         float dist = sqrt( dx * dx + dy * dy + dz * dz );
         v += pre1 * ( shared[i + 3*lsize] / dist )  *
             exp( -xkappa * (dist - shared[i + 4*lsize])) /
             (1.0f + xkappa * shared[i + 4*lsize]);
     }
     barrier(CLK_LOCAL_MEM_FENCE);
}
```

WHY BARRIERS?



GPU CODE - OPTIMIZED

```
for( iatom = 0; iatom < natoms; iatom+=lsize )
{
     if((iatom+lsize) > natoms)
         lsize = natoms - iatom;

     if((iatom + lid) < natoms){
         shared[lid]               = ax[iatom + lid];
         shared[lid + lsize]       = ay[iatom + lid];
         shared[lid + 2*lsize]     = az[iatom + lid];
         shared[lid + 3*lsize]     = charge[iatom + lid];
         shared[lid + 4*lsize]     = size[iatom + lid];
     }
     barrier(CLK_LOCAL_MEM_FENCE);

     for(int i=0;i<lsize;i++){
          float dx = lgx - shared[i];
          float dy = lgy - shared[i + lsize];
          float dz = lgz - shared[i + 2*lsize];

          float dist = sqrt( dx * dx + dy * dy + dz * dz );
          v += pre1 * ( shared[i + 3*lsize] / dist )  *
                exp( -xkappa * (dist - shared[i + 4*lsize])) /
                (1.0f + xkappa * shared[i + 4*lsize]);
     }
     barrier(CLK_LOCAL_MEM_FENCE);
}
```

MacResearch
http://www.macresearch.org

XCODE

MORE INFORMATION

- MacResearch.org
  - OpenCL - http://www.macresearch.org/opencl
  - Amazon Store - http://astore.amazon.com/macreseorg-20
- Khronos Group - http://www.khronos.org/opencl