

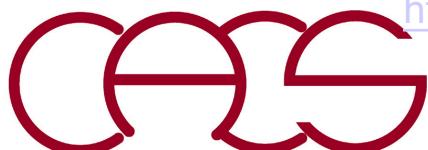
Pair Distribution Computation on GPU

Aiichiro Nakano

*Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
Department of Biological Sciences
University of Southern California*
Email: anakano@usc.edu

Goal: Using multidimensional Grid & Block

See B. G. Levine et al., *J. Comput. Phys.* **230**, 3556 (2011)
<http://cacs.usc.edu/education/cs596/Levin-RDFonGPU-JCP11.pdf>

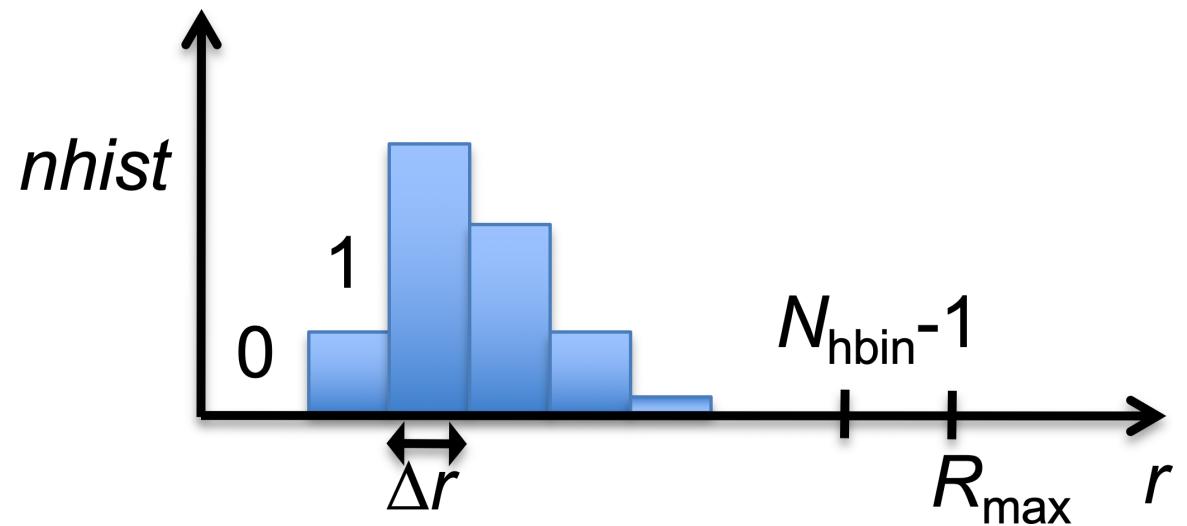
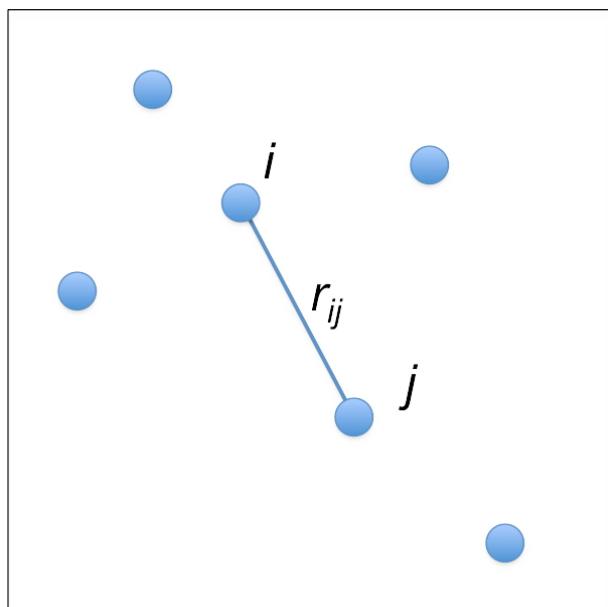


Pair Distribution

- Pair-distance histogram, `nhist[Nhbin]`

```
for all histogram bins i
    nhist[i] = 0
for all atomic pairs (i,j)
    ++nhist[ $\lfloor |\vec{r}_{ij}| / \Delta r \rfloor$ ]
```

reset
count



Pair Distribution Function

- **Pair-distribution function, $g(r)$**

$$g(r_i) = \frac{nhist(i)}{2\pi r_i^2 \Delta r \rho N}$$

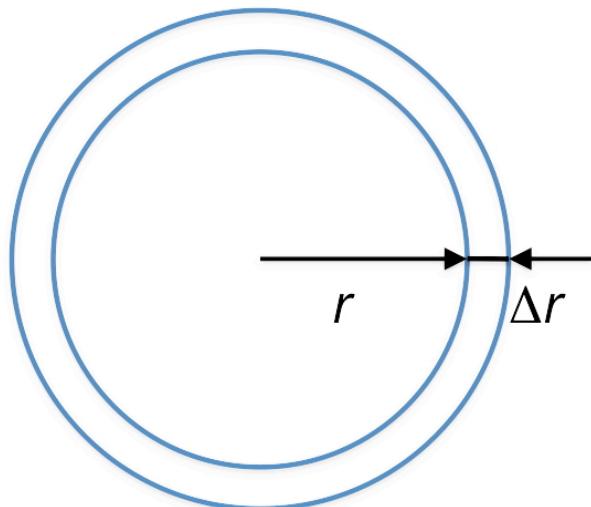
N: # of atoms ρ : # density

$g(r)$: For each atom, how many other atoms are distance r apart, normalized by # of atoms expected from average density; deviation from 1 signifies correlation with an atom at $r = 0$

With minimum-image convention,

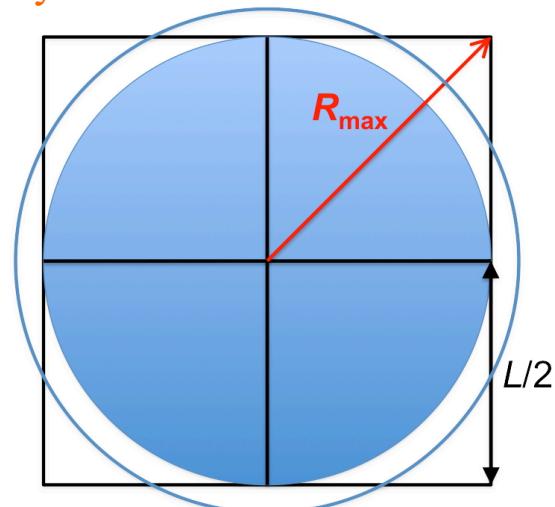
$$R_{\max} = \sqrt{\sum_{\alpha=x,y,z} \left(\frac{\text{al}[\alpha]}{2} \right)^2}$$

$$\Delta r = R_{\max}/N_{\text{hbin}}; r_i = (i+1/2)\Delta r$$



$$\therefore g(r_i) = \frac{2 \times nhist(i)}{N} \times \frac{1}{4\pi r_i^2 \Delta r \times \rho}$$

of other atoms j in a
concentric shell with
thickness Δr at distance
 r_i from atom i ; factor 2,
since each pair is
computed only once
Volume of
the shell
Average
number
density



Big Loops over Atomic Pairs

input: $r[]$, n Program: pdf0.c (atomic positions in pos.d)

```
for (i=0; i<n-1; i++) {
    for (j=i+1; j<n; j++) {
        rij = 0.0;
        for (a=0; a<3; a++) {
            dr = r[3*i+a]-r[3*j+a];
            /* Periodic boundary condition */
            dr = dr-SignR(alth[a],dr-alth[a])-SignR(alth[a],dr+alth[a]);
            rij += dr*dr;
        }
        rij = sqrt(rij); /* Pair distance */
        ih = rij/drh;
        nhis[ih] += 1.0; /* Entry to the histogram */
    } // End for j
} // Endo for i
```

Minimum-image convention (cf. MD lecture)

output: $nhis[]$

- n : Number of atoms
- $r[3*n]$: $r[3*i|3*i+1|3*i+2]$ is the $xlylz$ coordinate of the i -th atom
- $alh[a] = al[a]/2$: Half the simulation box lengths

```
float SignR(float v, float x) {if (x > 0) return v; else return -v;}
```

Variables in Device Memory

```
__constant__ float DALTH[3];
__constant__ int DN;
__constant__ float DDRH;
float* dev_r;           // Atomic positions
float* dev_nhis;        // Histogram

cudaMalloc((void**)&dev_r,sizeof(float)*3*n);
cudaMalloc((void**)&dev_nhis,sizeof(float)*NHBIN);

cudaMemcpy(dev_r,r,3*n*sizeof(float),cudaMemcpyHostToDevice);
cudaMemset(dev_nhis,0.0,NHBIN*sizeof(float));      / memory offset

cudaMemcpyToSymbol(DALTH,alth,sizeof(float)*3,0,cudaMemcpyHostToDevice);
cudaMemcpyToSymbol(DN,&n,sizeof(int),0,cudaMemcpyHostToDevice);
cudaMemcpyToSymbol(DDRH,&drh,sizeof(float),0,cudaMemcpyHostToDevice);

// Compute dev_nhis on GPU: dev_r[] → dev_nhis[]
cudaMemcpy(nhis,dev_nhis,NHBIN*sizeof(float),cudaMemcpyDeviceToHost);

cudaFree(dev_r);
cudaFree(dev_nhis);
```

- **cudaMemcpyToSymbol:**
Destination (in device) is either an address or variable name
Memory offset (in bytes) is added to destination

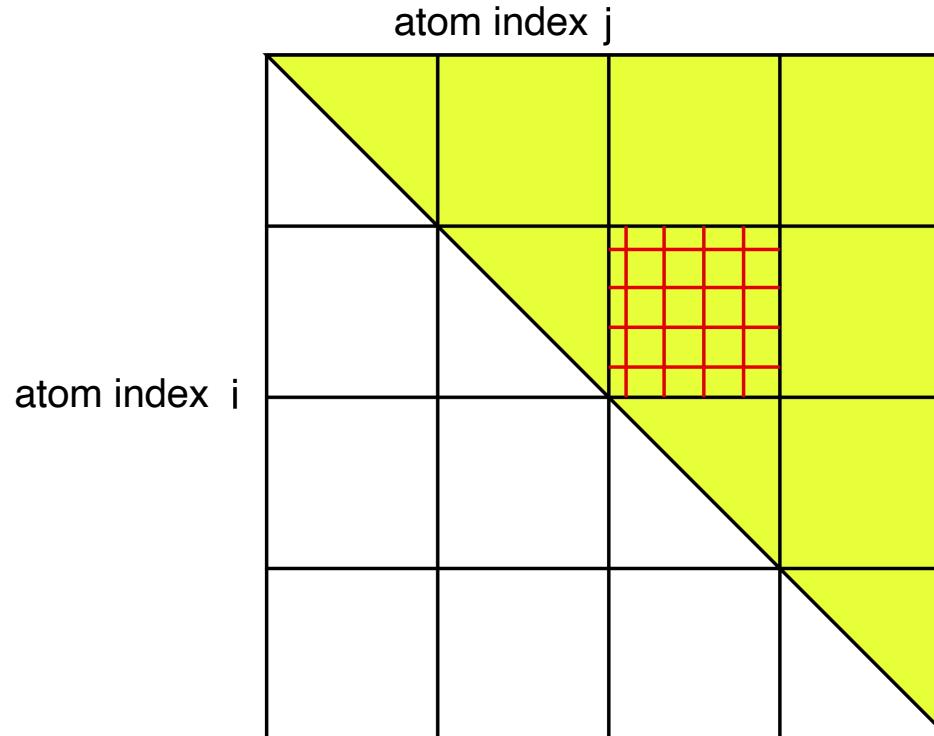


Nested Block & Thread Decompositions

- Nested decompositions
 - > Spatial decomposition among blocks
 - > Loop-index interleaving among threads within each block

In host program:

```
dim3 numBlocks(8,8,1);
dim3 threads_per_block(16,16,1);           in      out
gpu_histogram_kernel<<<numBlocks,threads_per_block>>>(dev_r,dev_nhis);
```



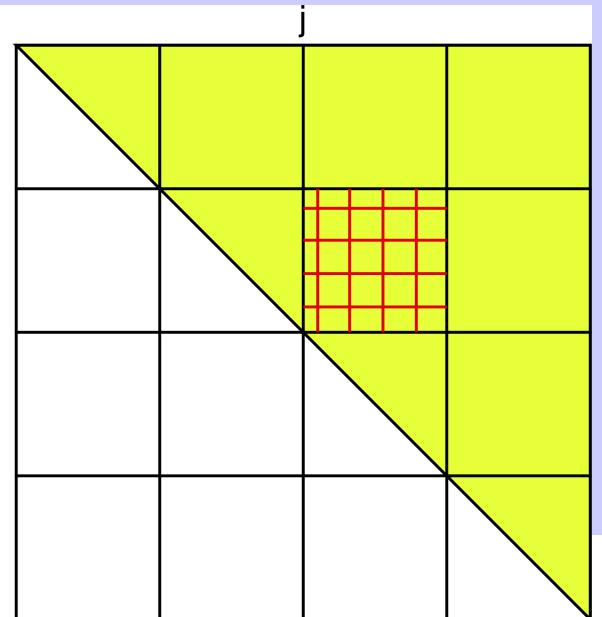
- Use a large enough number of blocks to reduce load imbalance among streaming multiprocessors (SMs)

Device Program for Histogram

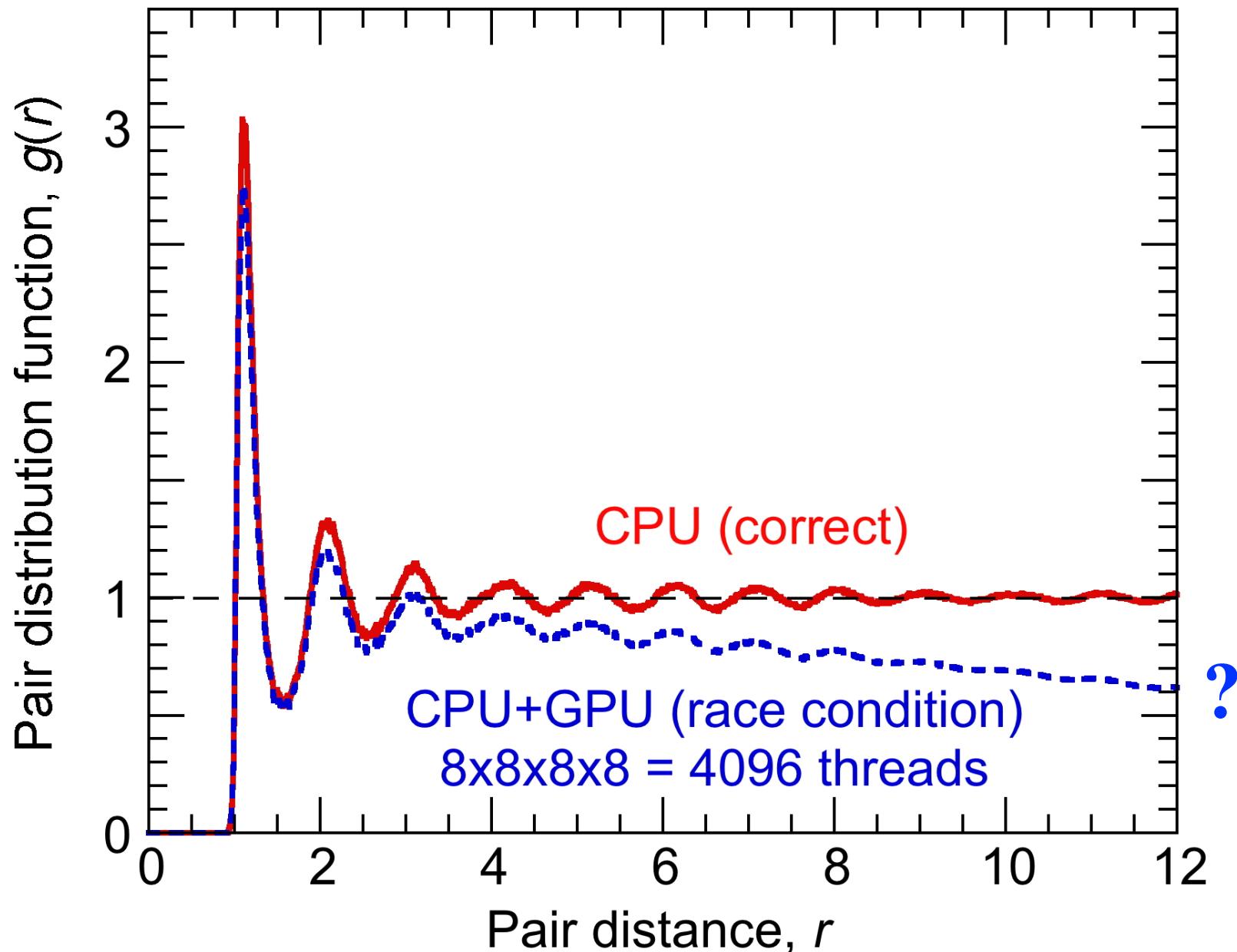
```
__device__ float d_SignR(float v, float x) {if (x > 0) return v; else return -v;}  
This is only called from the device program  
  
__global__ void gpu_histogram_kernel(float *r, float *nhis) {  
  
int iBlockBegin = (DN/gridDim.x)*blockIdx.x;  
int iBlockEnd = min((DN/gridDim.x)*(blockIdx.x+1),DN);  
  
int jBlockBegin = (DN/gridDim.y)*blockIdx.y;  
int jBlockEnd = min((DN/gridDim.y)*(blockIdx.y+1),DN);  
  
for (i=iBlockBegin+threadIdx.x; i<iBlockEnd; i+=blockDim.x) {  
    for (j=jBlockBegin+threadIdx.y; j<jBlockEnd; j+=blockDim.y) {  
        if (i<j) {  
            // Process (i,j) atom pair  
            rij = 0.0;  
            ...  
            nhis[ih] += 1.0;  
        } // end if i<j  
    } // end for j  
} // end for i  
}
```

Block spatial decomposition
via index offset

Thread interleaving
by skipping indices



Numerical Results



Race Condition

We just “saw” race condition in action!

```
for (i=iBlockBegin+threadIdx.x; i<iBlockEnd; i+=blockDim.x) {  
    for (j=jBlockBegin+threadIdx.y; j<jBlockEnd; j+=blockDim.y) {  
        if (i<j) {  
            rij = 0.0;  
            for (a=0; a<3; a++) {  
                dr = r[3*i+a]-r[3*j+a];  
                /* Periodic boundary condition */  
                dr = dr-d_SignR(DALTH[a],dr-DALTH[a])-d_SignR(DALTH[a],dr+DALTH[a]);  
                rij += dr*dr;  
            }  
            rij = sqrt(rij); /* Pair distance */  
            ih = rij/DDRH;  
            nhis[ih] += 1.0; /* Entry to the histogram */  
        } // end if i<j  
    } // end for j  
} // end for i
```

- In newer versions of CUDA, use atomic update
`atomicAdd(&nhis[ih], 1.0);`

Running CPU & GPU Versions at HPC

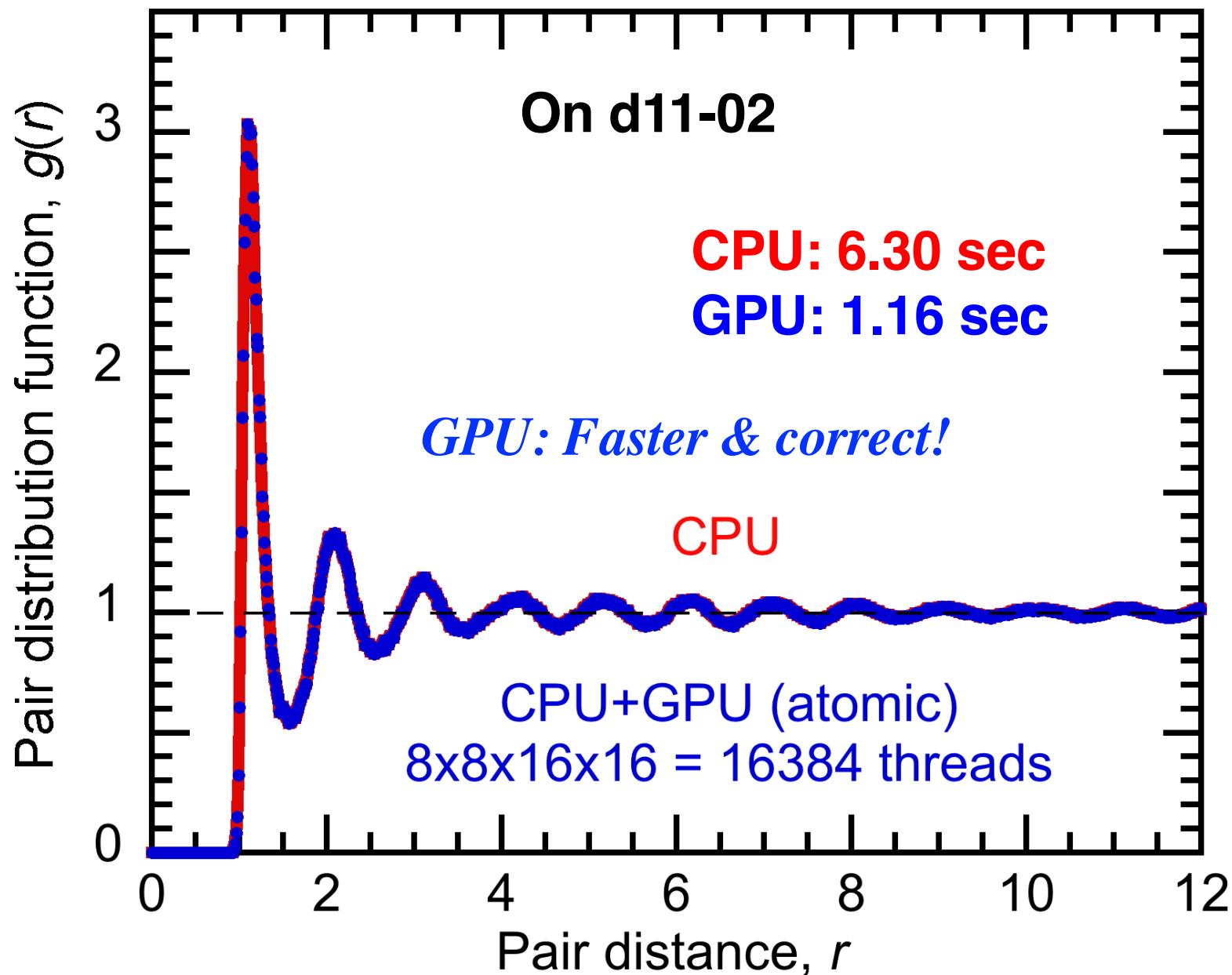
Script

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --gres=gpu:1
#SBATCH --time=00:00:59
#SBATCH --output=pdf.out
#SBATCH -A anakano_429
echo '##### CPU: gcc -o pdf0 pdf0.c -lm #####'
./pdf0
echo '##### GPU: nvcc -o pdf1 pdf1.cu      #####'
./pdf1
```

Output

```
##### CPU: gcc -o pdf0 pdf0.c -lm #####
Execution time (s) = 6.300000e+00
##### GPU: nvcc -o pdf1 pdf1.cu      #####
Execution time (s) = 1.160000e+00
```

Numerical Results



Summary: CUDA Pair-Distribution Computing

copy: host → device

input: $r[]$

constants: $alth[], n, drh$

Multithreading
(SPMD):

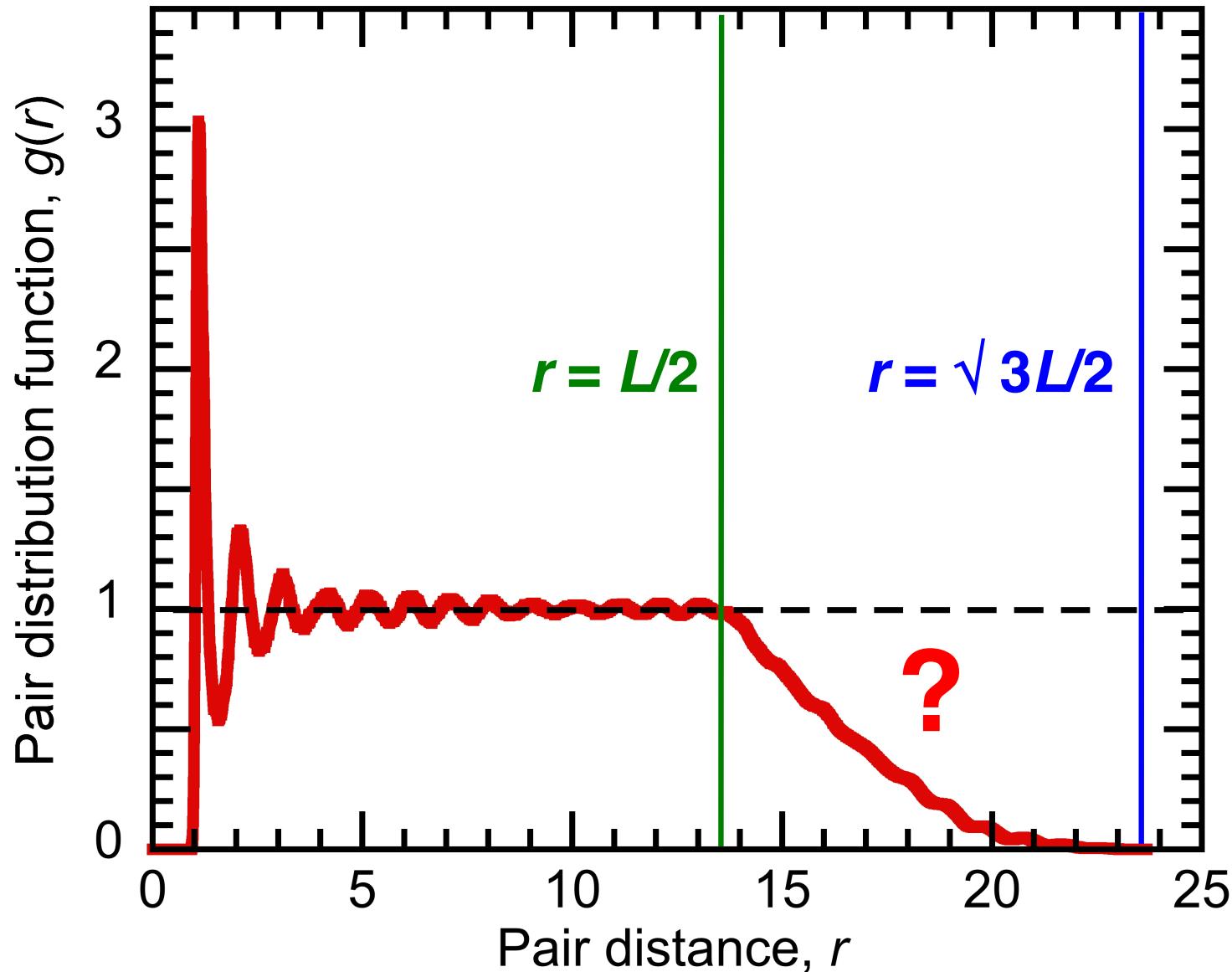
big (i, j) loop —
hybrid block
(SM) spatial
decomposition
& thread (SP)
interleaving

copy: host

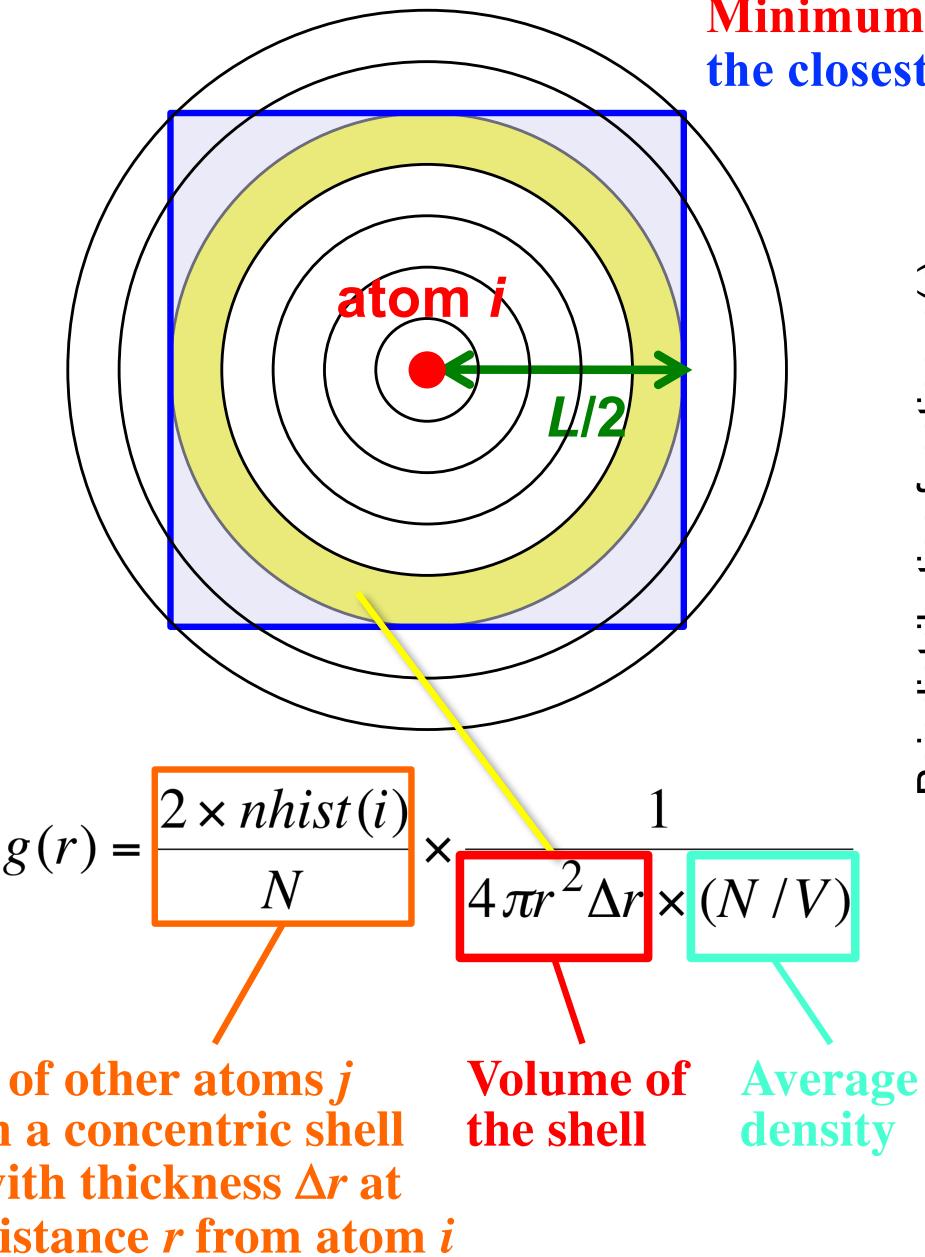
device

output: $nhis[]$

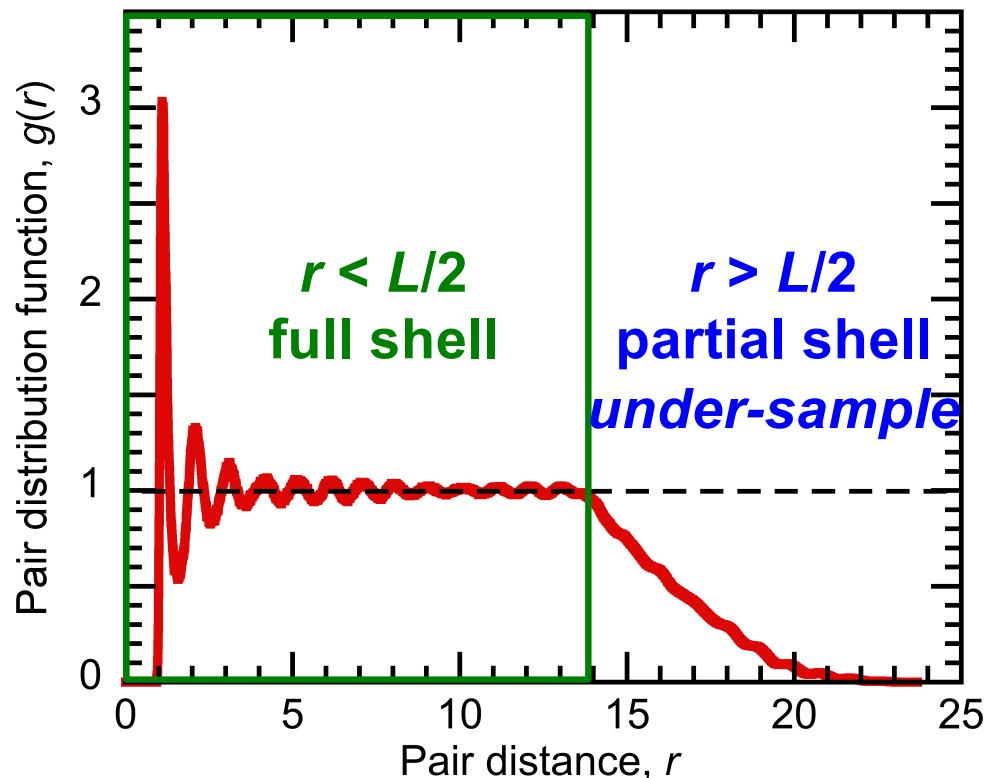
Finite-Size Effect on $g(r)$



Geometric Factor in $g(r)$

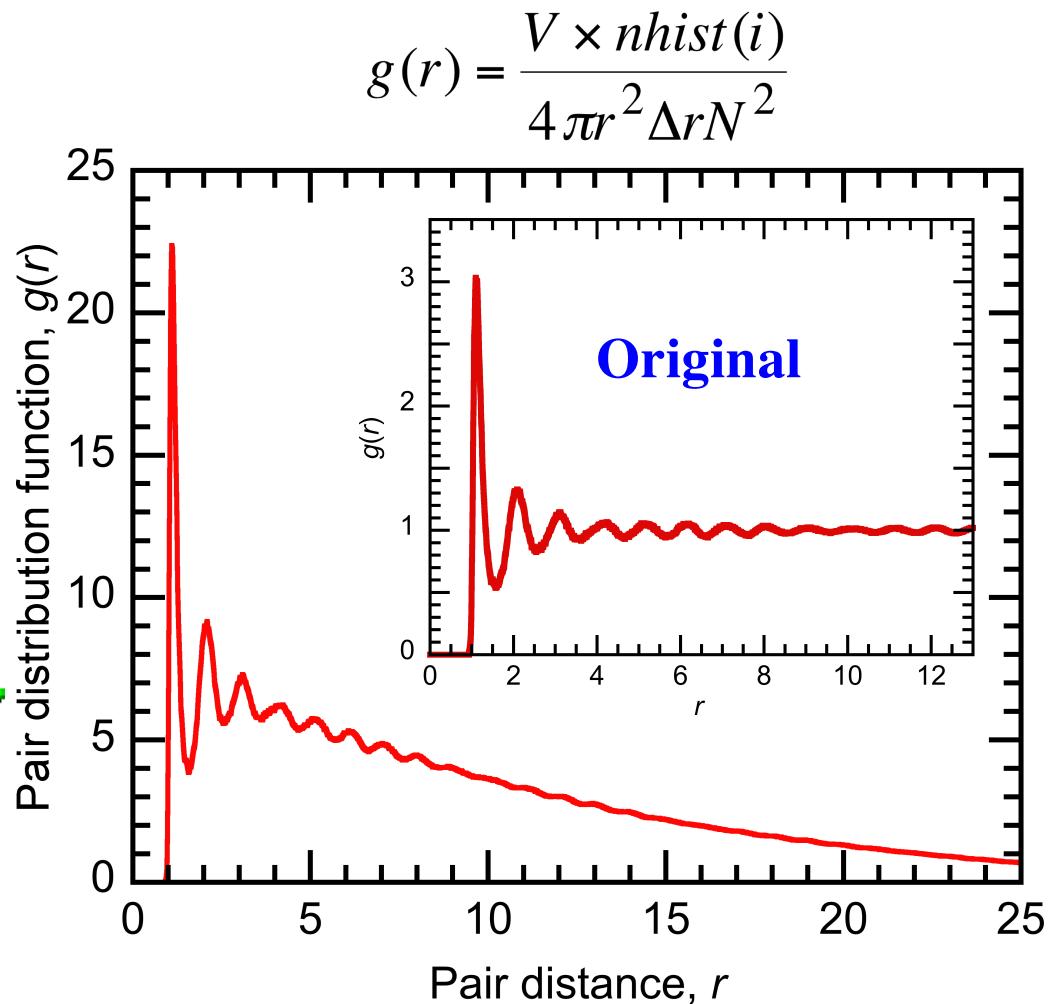
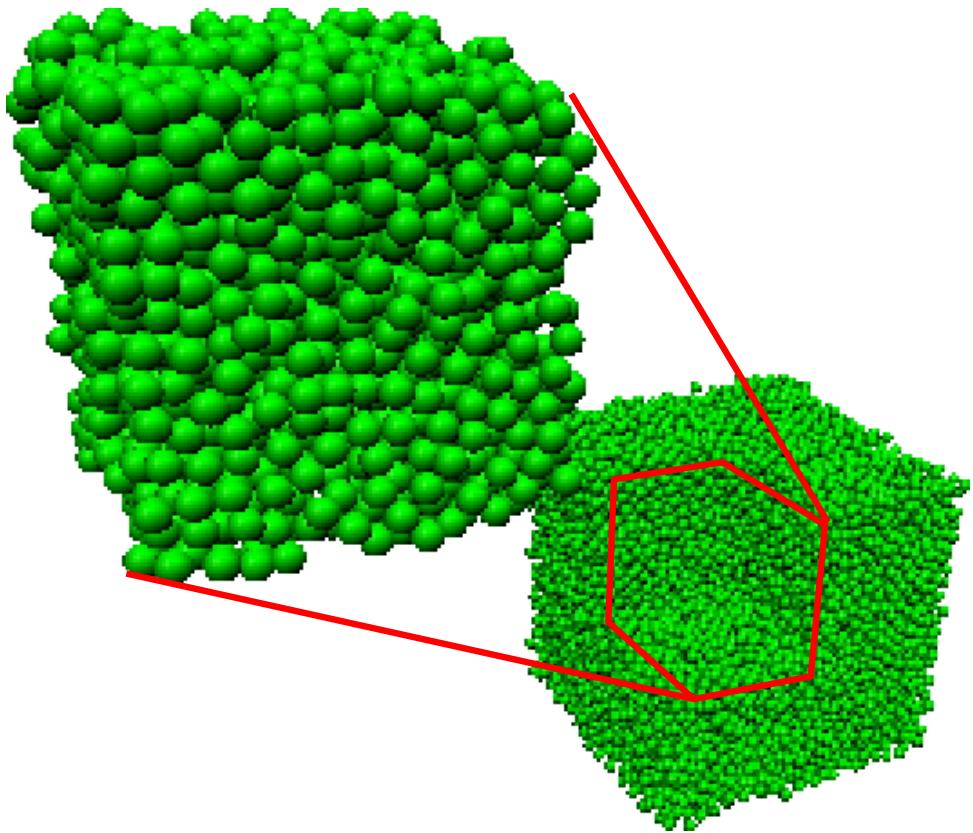


Minimum-image convention: For atom i , pick the closest periodic image of neighbor atom j



Large-scale Correlation in $g(r)$

One octant of the system
cut-out & displaced

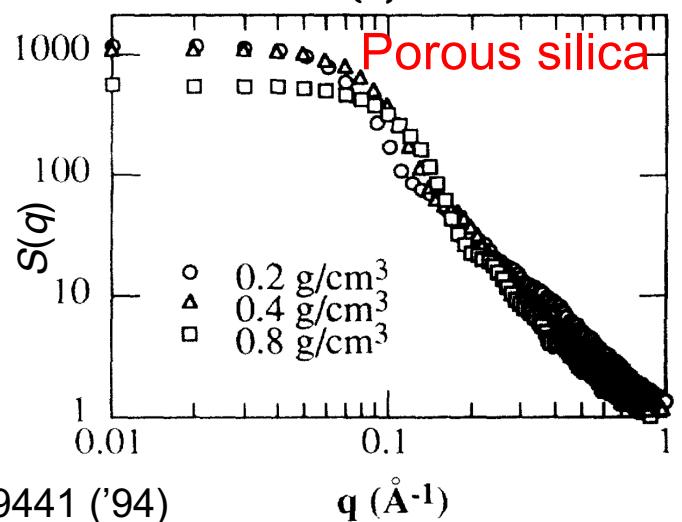
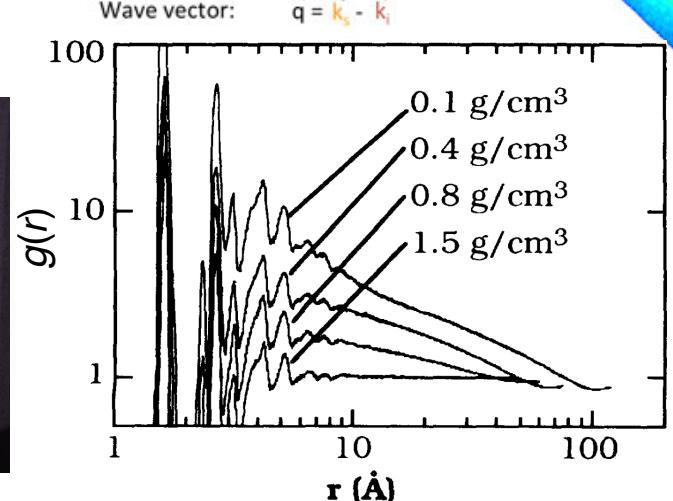
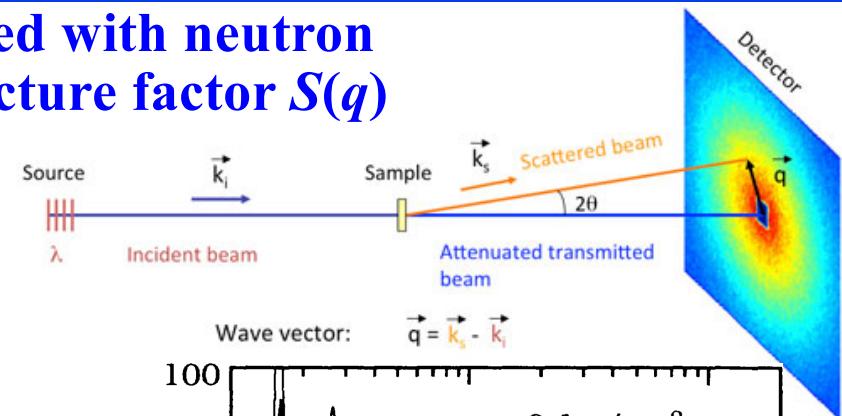
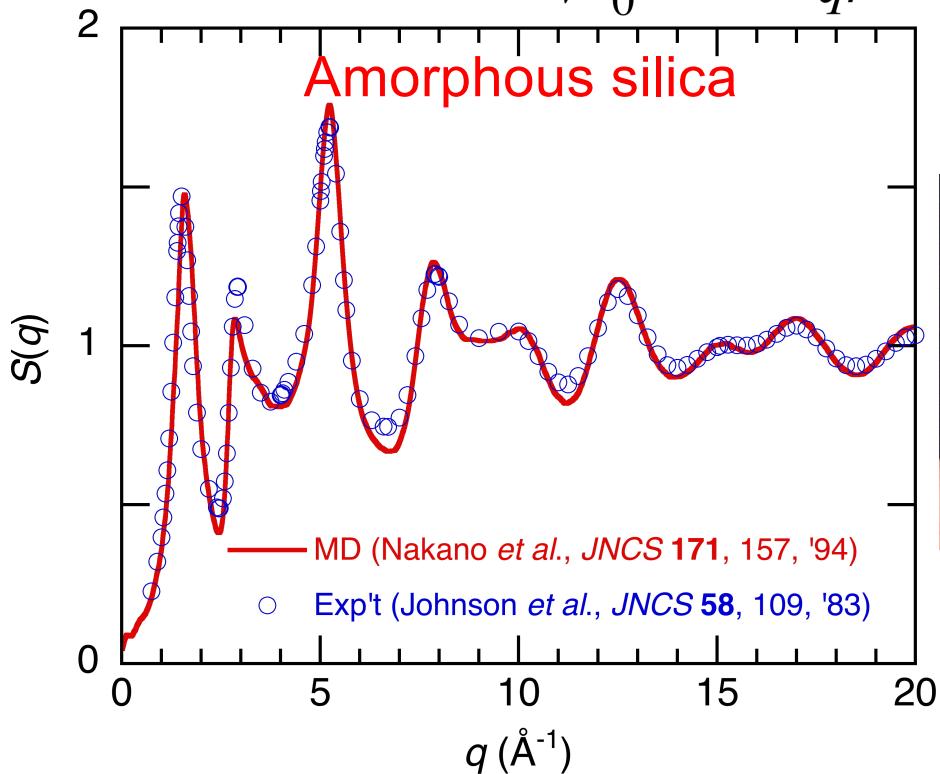


- Short-range correlation (*i.e.* peak positions) unchanged, just magnified by the lower average density, N/V_{expanded}
- Superimposed with larger length-scale geometric factors

Experimental Connection

- Short-range correlations are directly compared with neutron or X-ray scattering measurements of the structure factor $S(q)$

$$S(q) = 1 + 4\pi \frac{N}{V} \int_0^\infty dr r^2 \frac{\sin(qr)}{qr} [g(r) - 1]$$



- Long-range correlations are measured by small-angle neutron or X-ray scattering structure factors (SANS or SAXS)

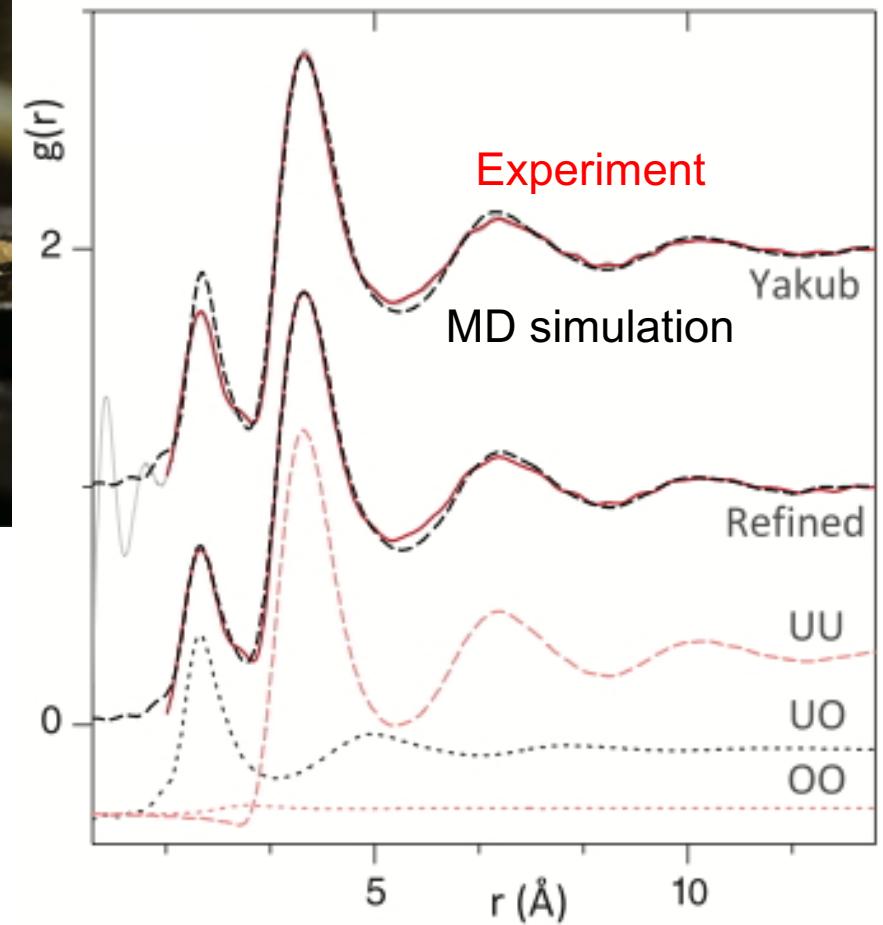
$$S(q) \propto q^{-d_f}$$

d_f : Fractal dimension

Nakano *et al.*, PRL 71, 85 ('93); PRB 49, 9441 ('94)

$g(r)$ of Molten UO_2

- X-ray scattering measurement using synchrotron radiation from 7 GeV electrons at the Advanced Photon Source of the Argonne National Lab.



Skinner et al., Science 346, 984 ('14)