Семинар 8: Модель SASA - реализация на ГП

1 Шаг 1

```
__global__ void generateSASAList_kernel(){
int d_i = blockIdx.x*blockDim.x + threadIdx.x;
if(d_i < c_sasaData.threadsCountTot){
int i = c_sasaData.d_threadAtom[d_i];
float4 r1 = tex1Dfetch(t_coord, i);
r1.w = tex1Dfetch(t_sasaRipr, (int)r1.w);

int sasaCount = c_sasaData.d_pairs12Counts[d_i];
for(i = 0; i < c_sasaData.d_pairsListCount[d_i]; i++){
float mult;
float4 r2;

int j = c_sasaData.d_pairsList[i*c_sasaData.widthTot + d_i];
r2 = tex1Dfetch(t_coord, j);
mult = tex1Dfetch(t_sasaRipr, (int)r2.w);
mult += r1.w;

r2 -= r1;
r2.w = sqrtf(r2.x*r2.x + r2.y*r2.y + r2.z*r2.z);
```

```
if(r2.w < mult){
c_sasaData.d_sasaList[sasaCount*c_sasaData.widthTot + d_i] = j;
sasaCount ++;
}
}
c_sasaData.d_sasaListCount[d_i] = sasaCount;
}
}
```

2 Шаг 2

```
__global__ void computeSASAPotentialB_kernel(){
int d_i = blockIdx.x*blockDim.x + threadIdx.x;
if(d_i < c_sasaData.threadsCountTot){
int i = c_sasaData.d_threadAtom[d_i];
int ati = c_gsystem.d_atomTypes[i];
float4 B;
float4 dr;
s_Ri[threadIdx.x] = tex1Dfetch(t_coord, i);
int atj;
float pij;
int covalentCount = c_sasaData.d_pairs12Counts[d_i];
for(i = 0; i < c_sasaData.d_sasaListCount[d_i]; i++){

atj = c_sasaData.d_sasaList[i*c_sasaData.widthTot + d_i];
if(i < covalentCount){
pij = c_sasaData.pij_cov;
```

```
} else {
pij = c_sasaData.pij_nb;
}
B = s_Ri[threadIdx.x];
dr = tex1Dfetch(t_coord, atj);
dr -= B;
dr.w = sqrtf(dr.x*dr.x + dr.y*dr.y + dr.z*dr.z);


atj = c_gsystem.d_atomTypes[atj];
B.w = dr.w*dr.w;
B.x = tex1Dfetch(t_sasaRipr2, ati);
B.y = tex1Dfetch(t_sasaRipr2, atj);
B.w = (B.y - B.x)/B.w;
float mult = (1.0f + B.w)/dr.w;
B.x = tex1Dfetch(t_sasaPiOverSi, ati);
B.y = tex1Dfetch(t_sasaRipr, ati);
mult *= B.x;
mult *= pij;
mult *= M_PI;
mult *= B.y;

B.x = mult*dr.x;
B.y = mult*dr.y;
B.z = mult*dr.z;

c_sasaData.d_BGrad[i*c_sasaData.widthTot + d_i] = B;
```

```
B.x = tex1Dfetch(t_sasaPiOverSi, atj);
B.y = tex1Dfetch(t_sasaRipr, atj);
mult = (1.0f - B.w)/dr.w;
mult *= B.x;
mult *= pij;
mult *= M_PI;
mult *= B.y;

B.x = mult*dr.x;
B.y = mult*dr.y;
B.z = mult*dr.z;

c_sasaData.d_BGradT[i*c_sasaData.widthTot + d_i] = B;

dr.x = tex1Dfetch(t_sasaRipr, ati);
dr.y = tex1Dfetch(t_sasaRipr, atj);
dr.z = tex1Dfetch(t_sasaPiOverSi, ati);

B.x = dr.x + dr.y;
B.x -= dr.w;
B.x *= M_PI;
B.x *= pij;
B.y = dr.y - dr.x;
B.y /= dr.w;

dr.w = tex1Dfetch(t_sasaPiOverSi, atj);

B.w = 1.0f + B.y;
```

```
B.w *= B.x;
B.w *= dr.x;
B.w *= dr.z;
B.w = 1.0f - B.w;

c_sasaData.d_BGrad[i*c_sasaData.widthTot + d_i].w = B.w;
c_sasaData.d_B[i*c_sasaData.widthTot + d_i] = B.w;

c_sasaData.d_BGradT[i*c_sasaData.widthTot + d_i].w =
1.0f - dr.w*dr.y*B.x*(1.0f - B.y);

}
}
}
```

3 Шаг 3

```
__global__ void computeSASAPotentialEnergy_kernel(){
int d_i = blockIdx.x*blockDim.x + threadIdx.x;
if(d_i < c_sasaData.threadsCountTot){
int a1 = c_sasaData.d_threadAtom[d_i];
int j;
int ati = c_gsystem.d_atomTypes[a1];
int sasaCount = c_sasaData.d_sasaListCount[d_i];
float pot = tex1Dfetch(t_sasaSigmaSi, ati);
for(j = 0; j < sasaCount; j++){
pot *= c_sasaData.d_B[j*c_sasaData.widthTot + d_i];
}
```

```
c_sasaData.d_sasaEnergies[a1] = pot;
}
}
```

4 Шаг 4

```
__global__ void computeSASAPotential_kernel(){
int d_i = blockIdx.x*blockDim.x + threadIdx.x;
if(d_i < c_sasaData.threadsCountTot){
int j, k;
int a1 =  c_sasaData.d_threadAtom[d_i];
float4 f = c_gsystem.d_forces[a1];
float mult;
float4 B;
float Vi = c_sasaData.d_sasaEnergies[a1];
for(k = 0; k < c_sasaData.d_sasaListCount[d_i]; k++){
j = c_sasaData.d_sasaList[k*c_sasaData.widthTot + d_i];
B = c_sasaData.d_BGrad[k*c_sasaData.widthTot + d_i];
mult = Vi/B.w;
f.x += mult*B.x;
f.y += mult*B.y;
f.z += mult*B.z;
B = c_sasaData.d_BGradT[k*c_sasaData.widthTot + d_i];
mult = c_sasaData.d_sasaEnergies[j];
mult /= B.w;
f.x += mult*B.x;
f.y += mult*B.y;
f.z += mult*B.z;
```

```
}
c_gsystem.d_forces[a1] = f;
}
}
```