

Ian Buck, Vishal Vaidyanathan, Vidya Rangasayee, Vijay Pande, Eric Darve
Computer Science Dept, Mechanical Engineering Dept, Chemistry Dept
Stanford University

Background

This project attempts to use the massive compute power available in today's GPUs to model the **folding of human proteins**. Learning more about this process will have a tremendous impact on biology and medicine and will help medical researchers understand diseases and find cures.

One of the leading software packages used to simulate protein folding is **Gromacs**. Up to 80-90% of the computation time in simulating a molecular system is computing the non-bonded or intermolecular forces, usually consisting entirely of the water molecules which surrounding a protein. In this work, we focus on this force calculation for implementing on the GPU as well as exploring additional algorithms deployed in Gromacs.

Folding of the Villin Headpiece

Gridding Technique

- The most expensive part of the simulation is the computation of the forces $\vec{F}(\vec{r}_i)$.
- A **cutoff** is applied so that all particles which are at a distance greater than a cutoff radius do not interact.

Applying cutoff

Gridding Technique
Molecules in the green cell only have interaction with those within cutoff, lying in the same cell and in the blue cells.

Brook Implementation

Brook is an extension of standard ANSI C and is designed to incorporate the ideas of data parallel computing and arithmetic intensity into a familiar and efficient language. Brook provides a familiar C-like programming environment for GPU-based computing.

```
kernel void f_00( float3 pos[], float iinr<,
                float shift<, float3 shiftvec[],
                float4 neighbors<, float qq,
                float c6, float c12, out float3 force< )
{
    float3 imolecule;
    float3 jmolecule1, jmolecule2, jmolecule3, jmolecule4;
    float3 d1, d2, d3, d4;
    float4 rinv, rinvsg, fs, rinvsix, vnb6, vnb12, vccoul;

    imolecule = pos[iinr.x] + shiftvec[shift];
    jmolecule1 = pos[neighbors.x]; jmolecule2 = pos[neighbors.y];
    jmolecule3 = pos[neighbors.z]; jmolecule4 = pos[neighbors.w];

    d1 = imolecule - jmolecule1; d2 = imolecule - jmolecule2;
    d3 = imolecule - jmolecule3; d4 = imolecule - jmolecule4;

    rinv.x = rsqrt(dot(d1, d1)); rinv.y = rsqrt(dot(d2, d2));
    rinv.z = rsqrt(dot(d3, d3)); rinv.w = rsqrt(dot(d4, d4));

    rinvsg = rinv*rinv;
    rinvsix = rinvsg*rinvsg*rinvsg;
    vnb6 = rinvsix*c6;
    vnb12 = rinvsix*rinvsix*c12;
    vccoul = qq*rinv;
    fs = (12.0*vnb12-6.0*vnb6*vccoul)*rinvsg;

    force = d1 * fs.x + d2 * fs.y + d3 * fs.z + d4 * fs.w;
}
```

- Brook files can co-exist within existing C source. This allows us to port only the compute intensive portions of GROMACS to Brook. The compiled Brook files link with the rest of Gromacs code.
- Brook simplifies GPU programming permitting developers with no graphics programming experience to benefit from GPU-based computing.
- Brook supports both Linux and Windows, NVIDIA and ATI, OpenGL and DirectX. This allows us to search for the fastest compute platform for molecular dynamics

Brook oxygen-oxygen force kernel

Numerical ODE in Molecular Dynamics

- Gromacs simulates molecular dynamics as an N-body problem
 - Newton's equations of motion

$$m_i \frac{d^2 \vec{r}_i}{dt^2} = -\nabla_{\vec{r}_i} U(\vec{r}_1, \dots, \vec{r}_N) = \vec{F}(\vec{r}_i)$$
 - Potential energy U is defined as the sum of several terms:
 - Non-bonded Interactions

Coulomb electrostatics $\frac{q_i q_j}{4\pi\epsilon_0 |\vec{r}_i - \vec{r}_j|^2}$

Lennard-Jones vdW $\frac{1}{|\vec{r}_i - \vec{r}_j|^6}$
 - Bonded Interactions

Bond stretching $\frac{1}{2} kx^2$

Bond bending $\frac{1}{2} k\theta^2$

Torsion $1 + \cos(n\theta)$
- The Ordinary Differential Equations (ODE) can be solved with the **Leap-frog scheme**.

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \Delta t \vec{v}_i(t) + \frac{\Delta t^2}{2} \vec{F}(\vec{r}_i(t))$$

$$\vec{v}_i(t + \frac{\Delta t}{2}) = \vec{v}_i(t - \frac{\Delta t}{2}) + \Delta t \frac{\vec{F}(\vec{r}_i(t))}{m_i}$$
- Using this method, it is possible to simulate the complex trajectories of atoms and molecules for very long periods of time. However, only the advent of more powerful supercomputers will allow studying molecules over time scales which are biologically and experimentally relevant.

Force Calculation Algorithm

- Load initial input data into streams. Because of hardware constraints, all neighbor lists must be of the same length for the code to run efficiently. Consequently, molecules with extra neighbors have multiple neighbor lists. Molecules with too few neighbors are given dummy neighbors.
 - A. Molecule list: 0 0 1 2 2 2 ... 899 (Dummy molecule index)
 - B. Neighbor list: 5 49 759 7 632 4 900 900 ... 4 555 38 900
- Compute the forces between each molecule and the molecules in its neighbor list(s). Store the forces in a new stream.
- Reduce forces to compute total molecule force.
 - C. Partial Forces: F(0) F(0) F(1) F(2) F(2) F(2) ... F(899)
 - D. Total Forces: F(0) F(1) F(2) F(3) ... F(899) (Partial forces are combined to generate the final force stream)

Preliminary Results

Water-Water Force Kernel

Component	Pentium4 3.0 GHz	ATI Radeon X800XT
GPU Readback	~15s	~10s
Force Function	~20s	~15s
Other	~10s	~5s

- Initial test illustrates force kernel is 20% faster than CPU version. By implementing the rest of the compute portions of GROMACS (black portion) on the GPU, we can eliminate the readback step.

Maintaining Bond Constraints

After the force update, we must apply the bond constraints to connected atoms. Gromacs uses the SHAKE algorithm, an iterative constraint solver.

Updating atom positions:
 (1,2) \rightarrow (1',2')
 (2',6) \rightarrow (2'',6'')
 (2'',3) \rightarrow (2''',3''') ...

SHAKE for GPUs operates on disjoint atoms in parallel:

Pass 1	Pass 2	Pass 3
(1,2) \rightarrow (1',2')	(2',6) \rightarrow (2'',6'')	(2'',3') \rightarrow (2''',3''')
(3,4) \rightarrow (3',4')	(4',7) \rightarrow (4'',7'')	(4'',5) \rightarrow (4''',5''')

Current Status

Our goal is to have all key components of Gromacs executing on the GPU.

- Non bonded forces
- Bonded forces
- Dihedrals
- constraints
- Important MD Methods
 - Temperature/Pressure Coupling
 - Virial calculation

- We are also looking at other aspects of biocomputing:
 - Protein matching: Comparing the sequence of a target protein against known protein sequence databases.
 - Protein docking: Exploring the fit between two proteins to predict potential active sites.

