Parallelizing the n-body solvers

- Let's try to apply Foster's methodology to the n-body solver. Since we initially want lots of tasks, we can start by making our tasks the computations of the positions, the velocities, and the total forces at each timestep.

- In the basic algorithm, the algorithm in which the total force on each particle is calculated directly from Formula.2, the computation of Fq (t), the total force on particle q at time t, requires the positions of each of the particles $S_r(t)$, for each r. The computation of $V_q(t+\Delta t)$ requires the velocity at the previous timestep, $V_q(t)$ and the force, F_q (t) at the previous timestep.
- Finally, the computation of Sq(t+Δt) requires Sq(t) and Vq(t). The communications among the tasks can be illustrated as shown in Figure 3 below.

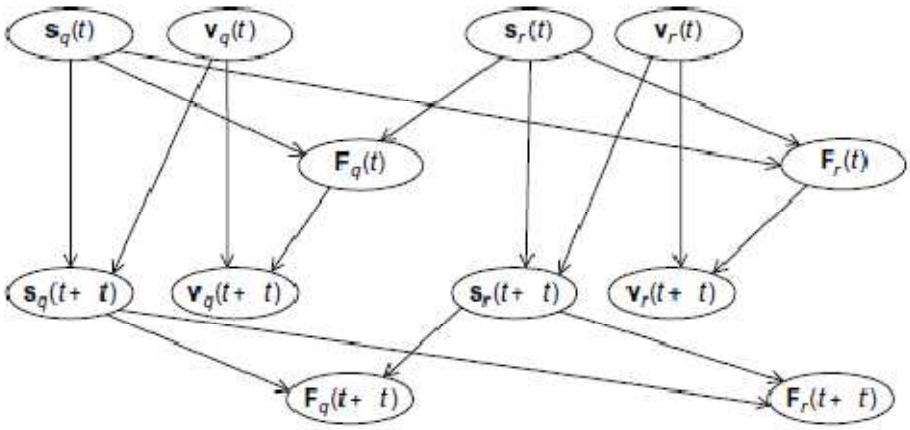

Fig3: communications among the tasks in the basic n-Body solver

- The figure makes it clear that most of the communication among the tasks occurs among the tasks associated with an individual particle.
  If we agglomerate the computations of $S_{q(}t)$,$V_{q(}t)$ and $F_q(t)$, our intertask communication is greatly simplified (see Figure .4).
- Now the tasks correspond to the particles and, in the figure, we've labeled the communications with the data that's being communicated. For example, the arrow from particle q at timestep t to particle r at timestep t is labeled with sq, the position of particle q.
- For the reduced algorithm, the "intra-particle" communications are the same. That is, to compute Sq(t+Δt) we'll need Sq(t) and Vq(t), and to compute Vq(t+Δt),
  we'll need vq(t) and Fq(t).

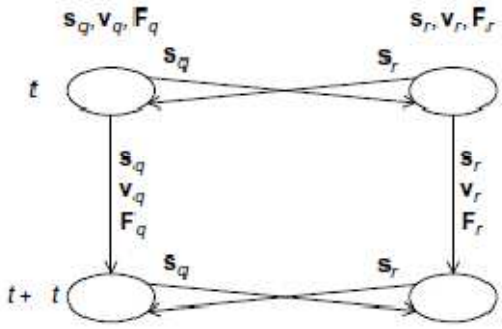

Fig 4: Communications among agglomerated tasks in the basic n-body solver

- In the reduced algorithm, we make use of the fact that the force $f_{rq} = -f_{qr}$. So if q < r, then the communication from task r to task q is the same as in the basic algorithm—in order to compute Fq(t), task/particle q will need Sr(t) from task/particle r. However, the communication from task q to task r is no longer Sq(t), it's the force on particle q due to particle r, that is, fqr(t). Fig 5.

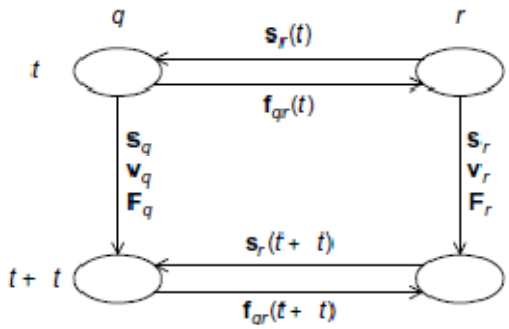

Fig 5: Communications among agglomerated tasks in the reduced n-body solver (q < r)

- The final stage in Foster's methodology is mapping. If we have ***n*** particles and ***T*** timesteps, then there will be nT tasks in both the basic and the reduced algorithm. Astrophysical n-body problems typically involve thousands or even millions of particles, so n is likely to be several orders of magnitude greater than the number of available cores. However, T may also be much larger than the number of available cores.

- At first glance, the assignment of particles to cores that assigns roughly n/thread count particles to each core will do a good job of balancing the workload among the cores, and for the basic algorithm this is the case.

- In the basic algorithm the work required to compute the position, velocity, and force is the same for every particle. However, in the reduced algorithm the work required in the forces computation loop is much greater for lower-numbered iterations than the work required for higher-numbered iterations.

- To see this, recall the pseudocode that computes the total force on particle q in the reduced algorithm:

```
for each particle k > q {
   x_diff = pos[q][X] - pos[k][X];
   y_diff = pos[q][Y] - pos[k][Y];
   dist = sqrt(x_diff*x_diff + y_diff*y_diff);
   dist_cubed = dist*dist*dist;
   force_qk[X] = G*masses[q]*masses[k]/dist_cubed * x_diff;
   force_qk[Y] = G*masses[q]*masses[k]/dist_cubed * y_diff;

   forces[q][X] += force_qk[X];
   forces[q][Y] += force_qk[Y];
   forces[k][X] -= force_qk[X];
   forces[k][Y] -= force_qk[Y];
}
```

- Then, for example, when q = 0, we'll make n-1 passes through the for each particle k > q loop, while when q = n-1, we won't make any passes through the loop. Thus, for the reduced algorithm we would expect that a cyclic partition of the particles would do a better job than a block partition of evenly distributing the computation.

- Therefore with a composite task consisting of all of the computations associated with a single particle throughout the simulation, we conclude the following:
  1. A block distribution will give the best performance for the basic n-body solver.

  2. For the reduced n-body solver, a cyclic distribution will best distribute the workload in the computation of the forces. However, this improved performance may be offset by the cost of reduced cache performance in a shared-memory setting and additional communication overhead in a distributed-memory setting.