Solving n-body Problem Using HPX

Adrian Serio[1], Hartmut Kaiser[1,2]

[1]Center for Computation and Technology, [2]LSU Department of Computer Science

stellar.cct.lsu.edu

Introduction

The n-body problem, i.e. the prediction of the motion of a group of objects that interact with each other under the influence of a force, is a method that continues to present a computational challenge to scientist in a broad range of application areas, like astrophysics or computational biology. Existing codes are usually scaling-challenged causing overly long runtimes for real-world problem sizes. We hope to overcome some of the challenges that face computing an n-body problem by using a message driven, inherently asynchronous approach based on the HPX (High-Performance ParalleX) library. This runtime system provides researchers the ability to assign work to nodes, independent of other nodes, and allows users to have control of the timing of the execution of the work. Our goals for this project are to produce a program using HPX which calculates the forces due to gravity between all of the points, only calculates the forces between a pair of points once, updates the coordinates to a new position every timestep, and breaks the global barriers that prevent work form being done.

Methods

In order to calculate the forces between each point we used a nested for loop:

```
for (i=0; i+1<k; i++)
 for (L=i+1; L<k; L++)
```

In this way, the force between each point "i" and each subsequent point "L" is calculated (see Figure 1). Latter on, this calculated force is added to a vector which stores all of the forces acting on a single point.

Force Calculations

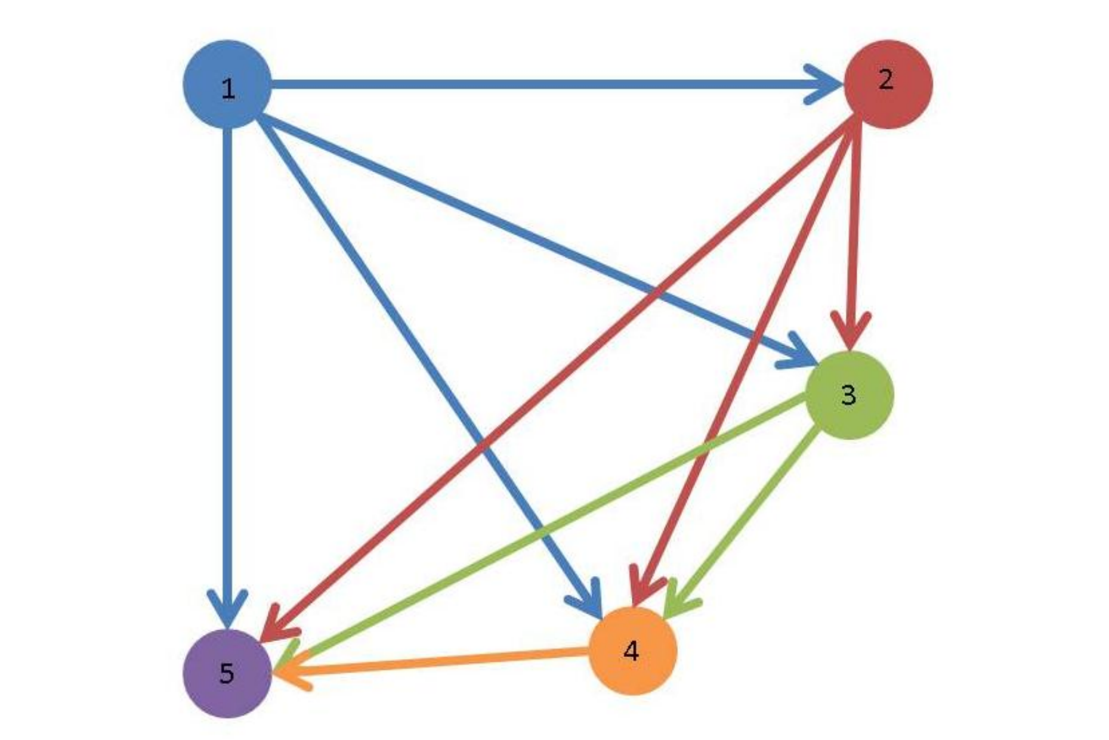

Figure 1: Each successive point has to calculate one less force than the previous point.

A point's new location is calculated using basic kinematic equations:

$$x_f = x_o + v_o t + \frac{1}{2} a t^2$$

And

$$v_f = v_o + at$$

As implied by the equations, this program also stores the values of each particles velocity.

One of the problem common to the n-body problem is that all of the forces must be calculated between all of the points before the particles' new positions can be calculated. In traditional parallel computation models, this created a "Global Barrier" in the program where all of the processes had to wait until every force had been calculated. Using HPX, however, we were able to "break" this barrier by breaking the problem into smaller work packets which after gathering the dependent variables can begin working (see Figure 2). In this way, nodes will be less starved for work and the calculation can proceed quicker (see Figure 3).

In order to break the problem down into smaller packets of work we split the nested "for" loop into two pieces. The first loop creates "promises", which are small packets of work that can be assigned to a node. These packets of work instruct the node to execute the second "for" loop, thus calculating all the forces between one point and the rest of the points.

Parallelism using HPX

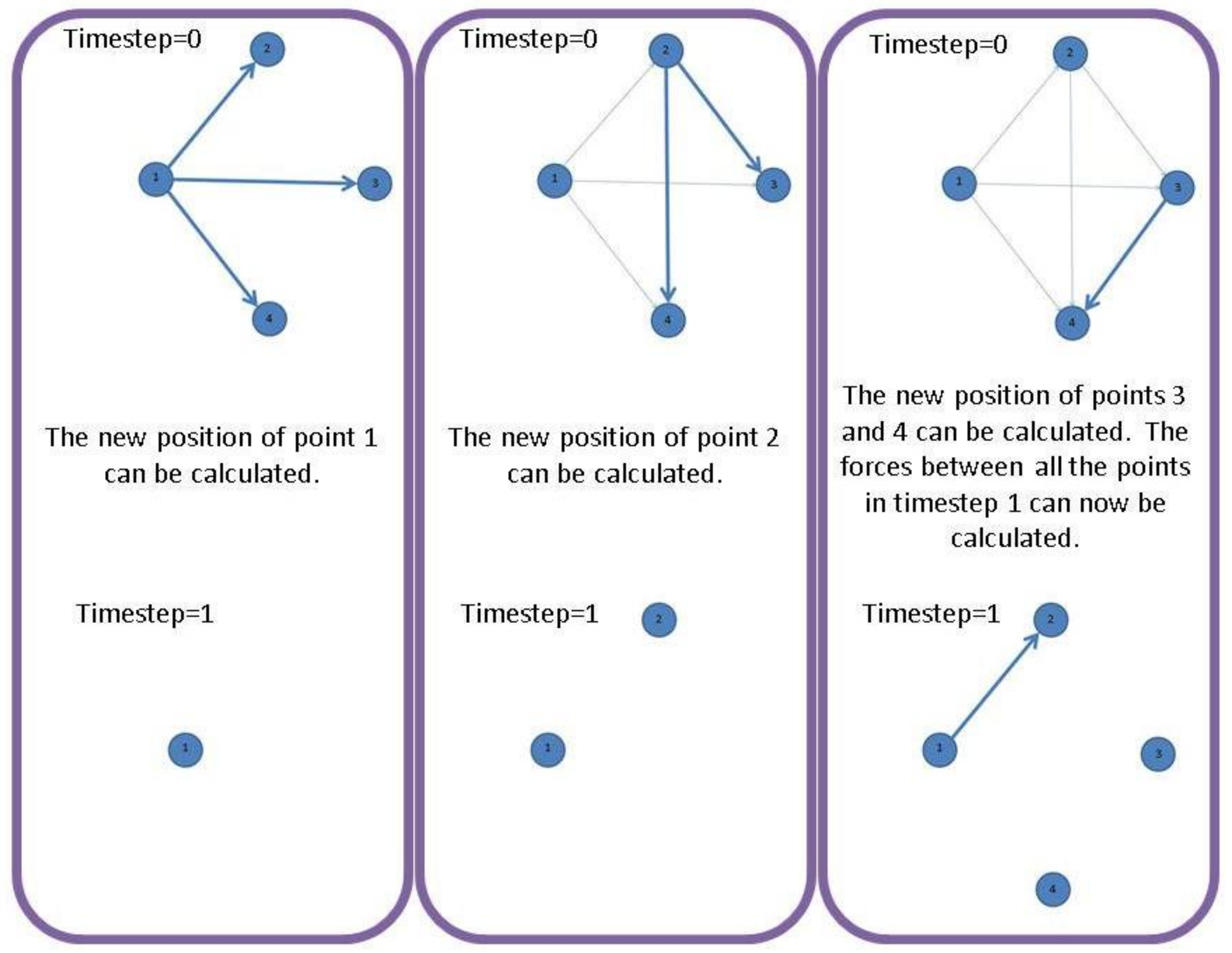
Timestep=0
The new position of point 1 can be calculated.
Timestep=1
Timestep=0
The new position of point 2 can be calculated.
Timestep=1
Timestep=0
The new position of points 3 and 4 can be calculated. The forces between all the points in timestep 1 can now be calculated.
Timestep=1

Figure 2: The top half of the chart shows what one node is computing. The bottom half shows what can now be computed after the computation is finished.

Additionally, the move function is wrapped in a promise, which allows work to begin on updating a point's position as soon as all the forces on a point are calculated. Because the two main functions of the program are wrapped in promises, HPX has the ability to assign all of the work at the start of the program. Therefore, as soon as the relevant information becomes available a node can begin the next step in computation.

Breaking Barriers

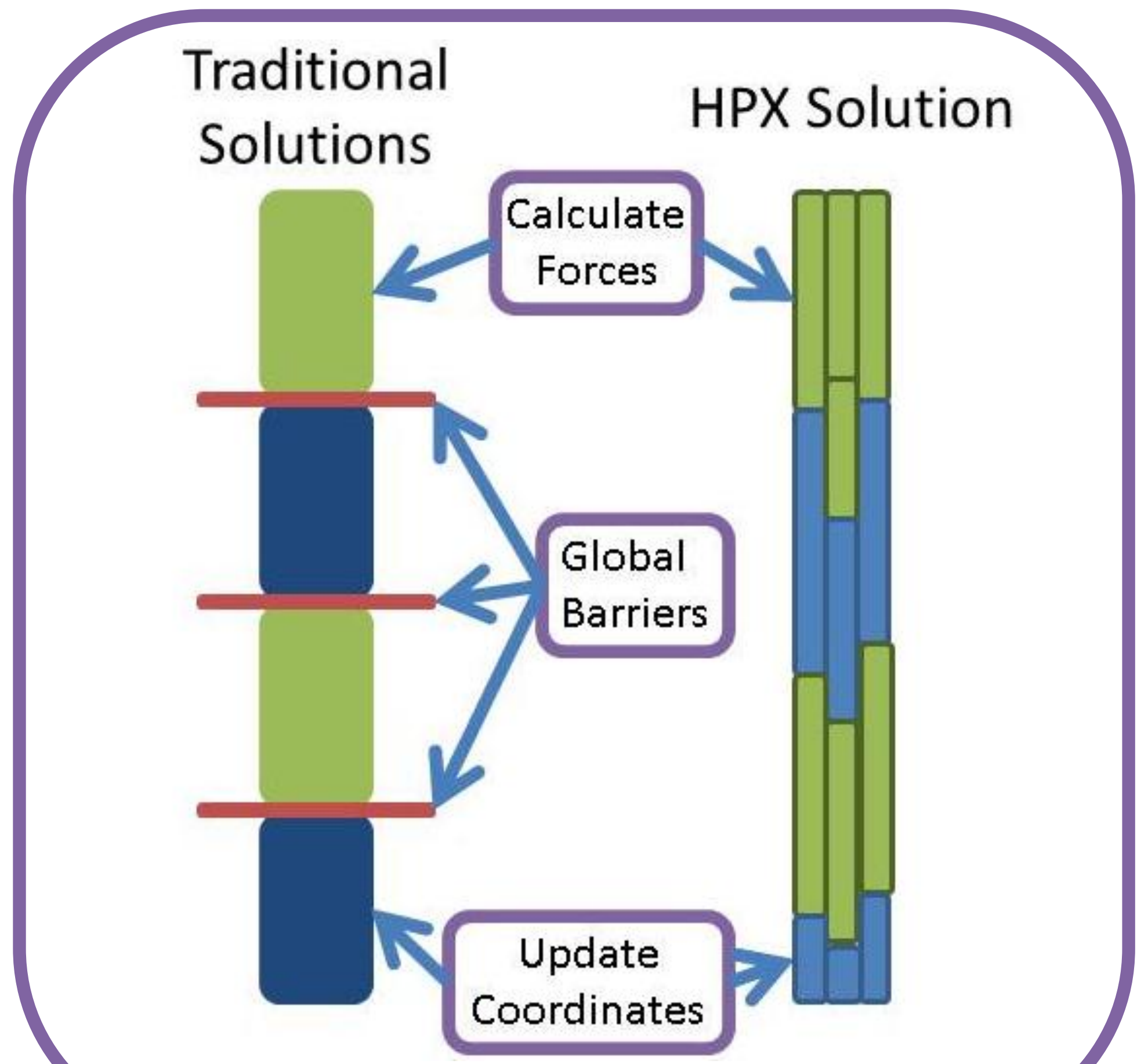

Figure 3: The HPX model

Conclusions

This programming model is one that we believe will propel computation into the future. By breaking down the problem into smaller pieces, we hope to achieve shorter computation times and better scaling, as more nodes will have better access to work. Our work with this program is far from over, while preliminary results are promising we hope to improve program stability by creating a independent queue which will handle all writing to the programs main storage vector, incorporate a cutoff distance to improve the speed of calculation of large datasets, and begin to optimize the programs code.

Acknowledgements:

- NSF Grants 1117470, 1048019, 1029161
- DARPA UHPC Funding
- LONI Allocation loni_hpx
- Bryce Adelstein-Lelbach
- Vinay C Amatya
- Maciej Brodowicz
- Thomas Heller
- Thomas Sterling