

Computational Biology and the X10 language

The Fast Multipole Algorithm

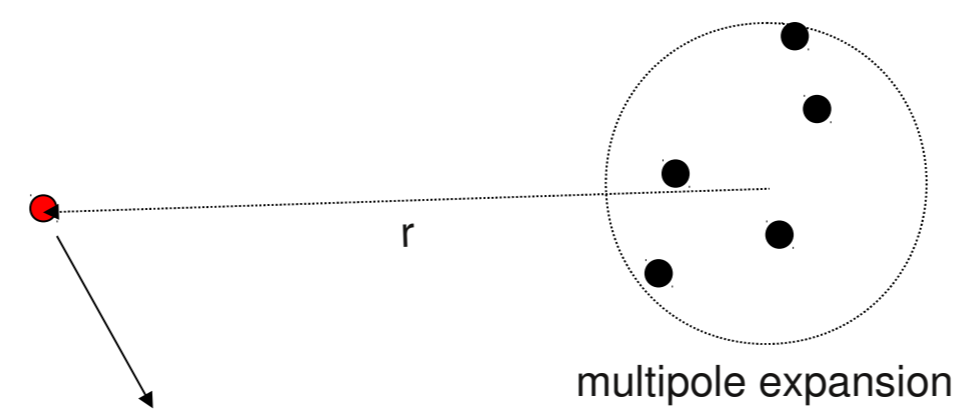
Evaluating the energy of a system of N bodies interacting via a pairwise potential is naively an $O(N^2)$ problem. The Fast Multipole Method^[1] allows approximation to a specified level of accuracy in only $O(N)$ time. It is widely used in molecular dynamics and astrophysical simulations.

Steps

- 1) Create the octree
- 2) Create multipole expansions for each box at the lowest level of octree
- 3) Combine expansions for child boxes into an expansion for each parent box (upward pass)
- 4) Transform multipole expansion for each box to local expansion for all boxes in interaction list; translate parent expansion to each child box (downward pass)
- 5) At lowest level, use local expansion for each box to calculate potential and force due to all remote particles; use direct evaluation for near particles

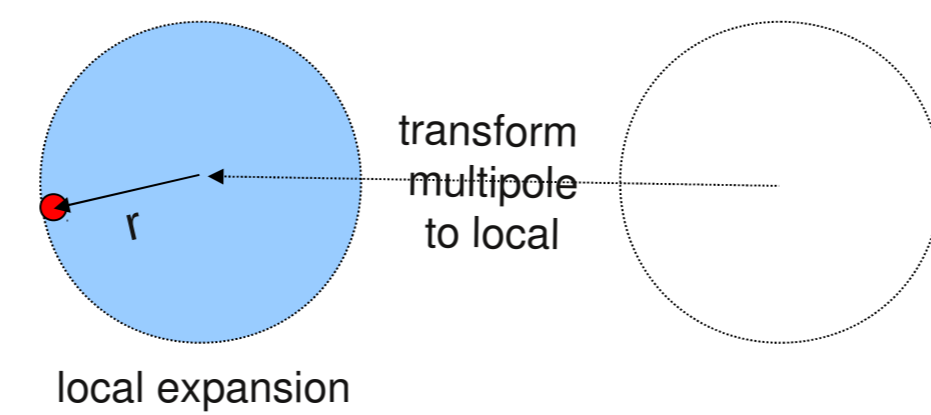
Expansions

A multipole expansion approximates the potential at the origin due to particles within a sphere centered at a distant point r .



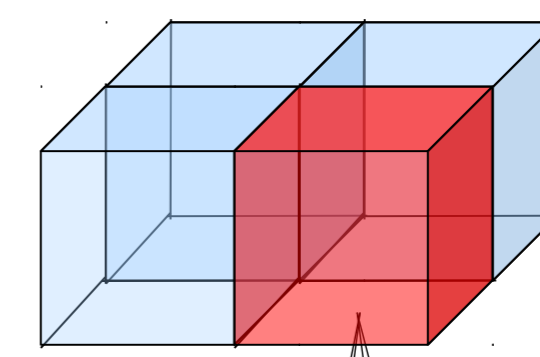
$$\Phi(\mathbf{r}) = \sum_{m=0}^{\infty} \sum_{l=-m}^m \frac{M_m^l}{r^{m+1}} \cdot Y_m^l(\theta, \phi)$$

A local expansion approximates the potential at any point r within a sphere centered at the origin, due to distant particles.



$$\Phi(\mathbf{r}) = \sum_{j=0}^{\infty} \sum_{k=-j}^j L_j^k \cdot Y_j^k(\theta, \phi) \cdot r^j$$

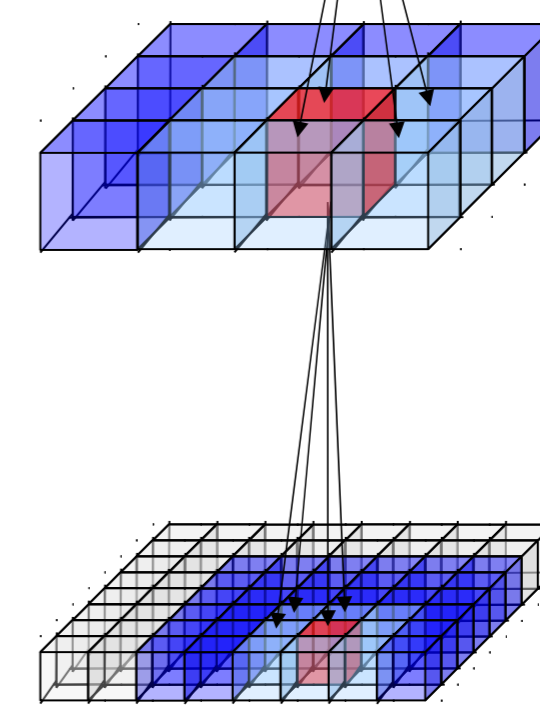
Hierarchical division of space



3D space is divided into an octree of cubic boxes. (Only 4 of 8 boxes shown here.)

For the red box:

- Multipole expansions used for every box in interaction list (dark blue)
- Non-well-separated boxes (light blue) evaluated at a lower level in the tree
- More remote boxes (white) are evaluated at a higher level



X10 – an emerging HPCS language

Traditionally, most high performance scientific applications have been developed in C++ or Fortran, using MPI for parallelism. However this model is becoming inadequate for productive development of high performance applications, due to code complexity and the failure to address the key conceptual challenges of distribution and heterogeneity. Also, C++ and Fortran are now rarely taught in undergraduate courses, having been supplanted by modern managed languages such as Java and C#.

In 2003, DARPA initiated the *High Productivity Computing Systems* (HPCS) program.^[2] One theme of the HPCS program is the creation of new parallel programming languages to support applications for emerging computer architectures. Two languages are in active development: Chapel^[3] and X10^[4].

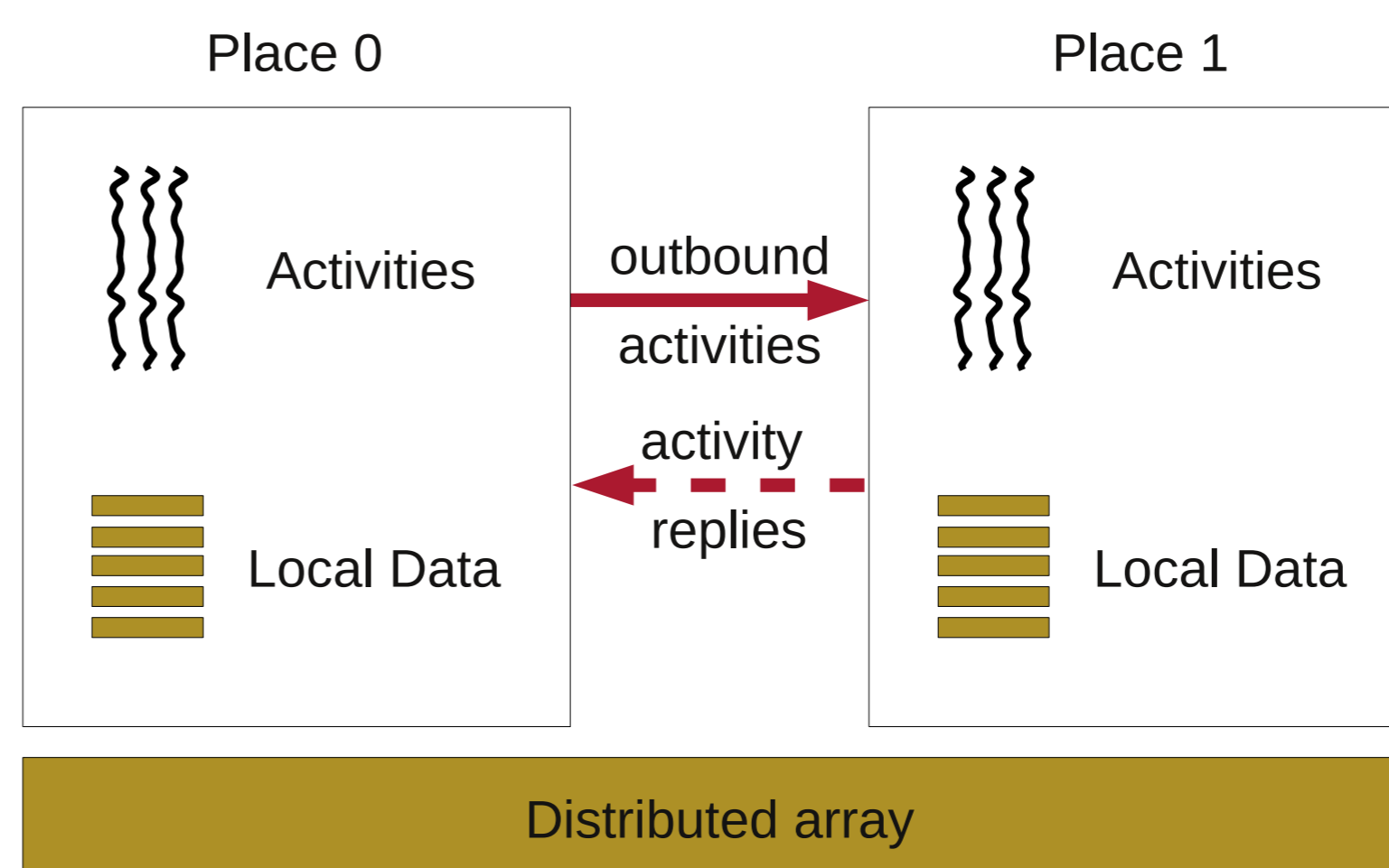
This work seeks to apply X10 to problems in the domain of computational biology, and explore issues of architectural heterogeneity and reliability. The initial aim is the development of molecular dynamics codes. The most computationally intensive part of these codes is the calculation of non-bonded electrostatic interactions, for which there are several alternative techniques including the Fast Multipole Algorithm.

Language features

X10 is a managed object-oriented language based on a Java-like serial subset, with the addition of explicit localization through *places* and asynchronous *activities* for concurrency. It supports a variety of common parallel programming idioms.

X10 provides a rich array sublanguage. An array is associated with a *region* (the set of index points) and a *distribution* specifying the location in memory of each point in the region. Regions and distributions provide the foundation for parallel constructs such as *ateach* which executes a statement over all the points in an array distribution.

To support synchronization X10 provides *clocks*, a generalization of MPI barriers for a dynamically varying set of activities.



X10 places are collections of activities and the data upon which they operate

Data representation

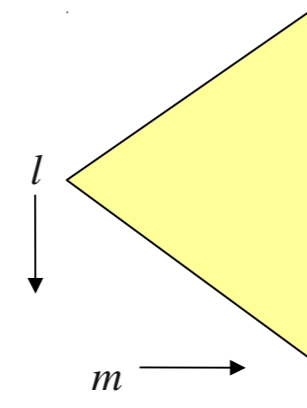
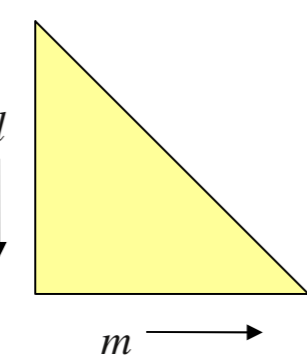
X10's array sublanguage permits succinct representation of key polynomials.

Expansions are defined in terms of associated Legendre polynomials P_m^l truncated at a number of terms p , where $0 \leq l \leq m \leq p$

```
val Plm = Array.make[Double] (
  Region.makeLowerTriangular(p+1));
```

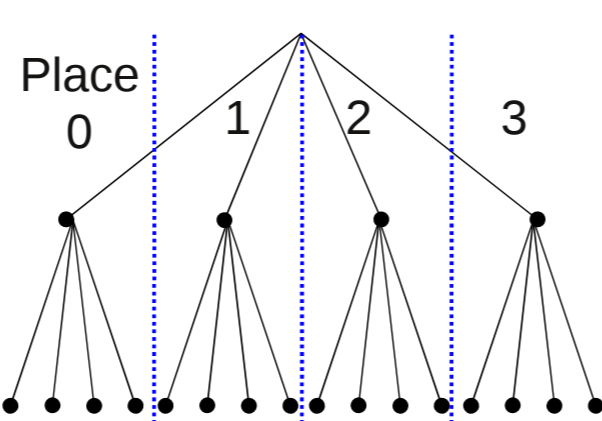
Multipole expansions O_m have unusually shaped regions $0 < m \leq p$; $abs(l) \leq m$

```
public class ExpansionRegion
  extends BaseRegion{self.rank==2} {
  global val p : Int;
  ...
}
this.terms = Array.make[Complex] (
  new ExpansionRegion(p),
  (Point) => Complex.ZERO);
```



Distribution

Current implementation has block distribution of lowest level boxes to all places.



- There are more efficient options for dividing the simulation space (e.g. costzones, Hilbert / Morton ordering)^[4].
How can X10 distributions be used to succinctly represent these options?

- Most communication is between nearby boxes in the 3D simulation space.
Is it possible to improve performance by assigning nearby boxes to nearby (lower communication latency) places?

Remote memory access

X10 *futures* support asynchronous access to remote data^[5], allowing communication to be overlapped with computation.

```
// start retrieve of data from box2
val box2ME = future(box2.location)
box2.multipoleExp;
... // perform other computation
// use retrieved data
box1.localExp.transformAndAddToLocal(
  transform21, box2ME());
```

This simplifies expression of data dependencies, compared to message passing (e.g. MPI); but OpenMP is still simpler (no data locality)!

References

- [1] <http://highproductivity.org/>
- [2] Chamberlain, Callahan and Zima, *Parallel programmability and the Chapel language*, Int. J. High Perform. Comput. Appl., 21 (2007), pp. 291–312.
- [3] Charles et al., *X10: an object-oriented approach to non-uniform cluster computing*, OOPSLA (2005)
- [4] O. Coulaud, P. Fortin and J. Roman, *Hybrid MPI-Thread parallelization of the Fast Multipole Method*, ISPCD '07 (2007)

Preliminary results

