TACC Technical Report IMP-14

1.5D All-pairs Methods in the Integrative Model for Parallelism

Victor Eijkhout*

February 22, 2016

* eijkhout@tacc.utexas.edu, Texas Advanced Computing Center, The University of Texas at Austin

Abstract

abstract

1 Problem description

We abstract the N-body problem as follows:

- We assume that there is a vector $c(\cdot)$ of the particle charges/masses and location information.
- To compute the location update, we need the pairwise interactions $F(\cdot,\cdot)$ based on storing tuples $C_{ij} = \langle c_i, c_j \rangle$.
- The interactions are then summed to a force vector $f(\cdot)$.

In the Integrative Model for Parallelism (IMP) model, the algorithm is sufficiently described if we know the respective data distributions; we are not immediately concerned with the local computations.

2 Particle distribution

An implementation based on particle distribution takes as its starting point the particle vector c distributed as $c(u)$, and directly computes the force vector $f(u)$ on the same distribution. We describe the computation as a sequence of three kernels, two of data movement and one of local computation[1]:

$$\begin{cases} \{\alpha\colon c(u)\} & \text{initial distribution for } c \text{ is } u \\ C(u,*) = c(u) \cdot_\times c(*) & \text{replicate } c \text{ on each processor} \\ \text{local computation of } F(u,*) \text{ from } C(u,*) & \\ f(u) = \sum_2 F(u,*) & \text{local reduction of partial forces} \\ \text{particle position update} & \end{cases} \tag{1}$$

The final kernel is a reduction with identical α and β-distribution, so it does not involve data motion. The local computation kernel also has not data motion. That leaves the gathering of $C(u,*)$ from the initial distribution $c(u)$. Absent any further information on the distribution u this is an allgather of N elements, at a cost of $\alpha \log p + \beta N$. In particular, the communication cost does not go down with p.

With a suitable programming system based on distributions, the system of equations (1) can easily be translated into code.

3 Work distribution

The particle distribution implementation used a one-dimensional distribution $F(u,*)$ for the forces conformally with the particle distribution. Since F is a two-dimensional object, it is also possible to

1. In the full IMP theory [2] the first two kernels can be collapsed.

$$\begin{array}{ll} \{\alpha\colon C(D\colon\ <I,I>)\} & \text{initial distribution on the processor diagonal} \\ C(I,I) = C(I,p) + C(q,I) & \text{row and column broadcast of the processor diagonal} \\ F(I,I) \leftarrow C(I,I) & \text{local interaction calculation} \\ f(D\colon\ <I,I>) = \sum_2 F(I,D\colon *) & \text{summation from row replication on the diagonal} \end{array} \tag{2}$$

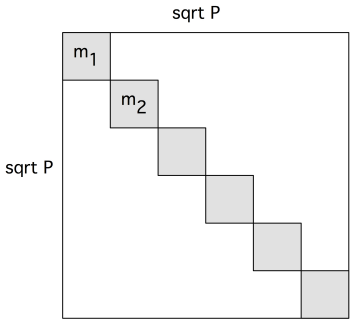
Figure 1: IMP realization of the 1.5D N-body problem

use a two-dimensional distribution. We will now explore this option, which was earlier described in [1]; our purpose here is to show how this strategy can be implemented and analyzed in the IMP framework. Rather than expressing the algorithm as distributed computation and replicated data, we use a distributed temporary, and we derive the replication scheme as collectives on subcommunicators.

For a two-dimensional distribution of F we need $(N/b)\times(N/b)$ processors where b is a blocksize parameter. For ease of exposition we use $b=1$, giving a number of processors $P=N^2$.

We will now go through the necessary reasoning. The full algorithm in close-to-implementable form is given in figure 1. For simplicity we use the identify distribution $I\colon\ p\mapsto\{p\}$.

Initial store on the diagonal of the processor grid We first consider the gathering of the tuples $C_{ij}=\langle c_i,c_j\rangle$. Initially, we decide to let the c vector be stored on the diagonal of the processor grid; in other words, for all p, processor $\langle p,p\rangle$ contains C_{pp}, and the content of all other processors is undefined.



To express this formally, we let D be the diagonal $\{\langle p,p\rangle\colon\ p\in P\}$. Using the mechanism of partially defined distributions (section **??**), the initial α-distribution is then

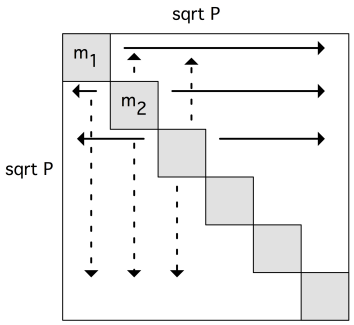
$$C(D\colon\ <I,I>)\equiv D\ni\langle p,q\rangle\mapsto C(I(p),I(q))=C_{pq}. \tag{3}$$

Replication for force computation The local force computation $f_{pq}=f(C_{pq})$ needs for each processor $\langle p,q\rangle$ the quantity C_{pq}, so we need a β-distribution of $C(I,I)$. The key to our story is the realization that

$$C_{pq}=C_{pp}+C_{qq}$$

so that

$$C(I,I)=C(I,p)+C(q,I)$$



(where p stands for the distribution that maps each processor to the index value p and similarly for q.)

To find the transformation from α to β-distribution we consider the transformation of the expression $C(D\colon \langle I,I\rangle)$. In general, any set D can be written as a mapping from the first coordinate to sets of values of the second:

$$D \equiv p \mapsto D_p \qquad \text{where} \qquad D_p = \{q\colon \langle p,q\rangle \in D\}.$$

In our particular case of the diagonal we have

$$D \equiv p \mapsto \{p\}.$$

With this we write

$$C(D\colon \langle I,I\rangle) = C(I,D_p\colon I) = C(I,\{p\}\colon p). \tag{4}$$

Note that equation (4) is still the α-distribution. The β-distribution is $C(I,p)$, and it is an exercise in pattern matching to see that this is attained by a broadcast in each row, which carries a cost of

$$\alpha \log \sqrt{p} + \beta N/\sqrt{P}.$$

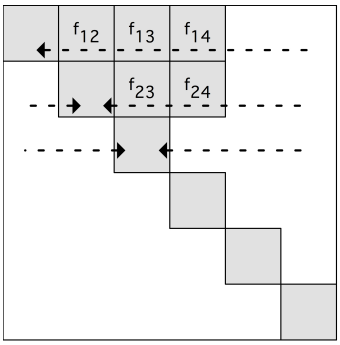
Likewise, $C(q,I)$ is found from the α-distribution by column broadcasts. We conclude that this variant does have a communication cost that goes down proportionally with the number of processors.

Local interaction calculation The calculation of $F(I,I) \leftarrow C(I,I)$ has identical α and β distributions, so it is trivialy parallel.

Summing of the forces We have to extend the force vector $f(\cdot)$ to our two-dimensional processor grid as $f(\cdot,\cdot)$. However, conforming to the initial particle distribution on the diagonal, we only sum on the diagonal. The instruction

$$f(D\colon\ <I,I>) = \sum_2 F(I,D\colon *)$$

has a β-distribution of $I,D:*$, which is formed from the α-distribution of I,I by gathering in the rows.



Summary This section shows a possible IMP syntax, albeit rather mathematical, for an algorithm with non-trivial communication [1]. The attraction of our approach is that the programmer specifies the essential operations, and not the communication scheme. The actual communication is derived from formal reasoning on distributions, not from any analysis by the programmer. This again bolsters our case that the IMP model allows for high-level expression of parallel algorithms.

In a homogeneous environment, for instance message passing on a distributed memory cluster, the realization of our algorithm in terms of programming primitives is straightforward. However, since we did not actually specify any communication but let it be derived, on a heterogeneous architecture the data dependencies could be realized through a combination of inter-node communication and intra-node task scheduling. This argues for our claim of largely architecture-independent programming.

References

[1] Michael Driscoll, Evangelos Georganas, Penporn Koanantakool, Edgar Solomonik, and Katherine Yelick. A communication-optimal N-body algorithm for direct interactions. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 213.

[2] Victor Eijkhout. A theory of data movement in parallel computations. *Procedia Computer Science*, 9(0):236 – 245, 2012. Proceedings of the International Conference on Computational Science, ICCS 2012, Also published as technical report TR-12-03 of the Texas Advanced Computing Center, The University of Texas at Austin.