Parallel programming in the last 25 years — forward or backward?

Jun Makino
Interactive Research Center of Science
Tokyo Institute of Technology

MODEST-10d: High-Level Languages for Hugely Parallel Astrophysics Simulations: Dialogues between Computer Scientists and (Astro)physicists Aug 23-24, 2011, CPS, Kobe
Summary

- Until around early ’90s, there were “parallel languages” available on high-performance computers.
- They are no longer there.
- HPF is the last to die (still available on NEC parallel vector machines)
- Why did this happen? What could/should we do?
Talk structure

- Parallel Programming @2011
- Parallel Programming @1986
- Why things has changed?
- What could we do?
- What should we do?
Parallel Programming @2011

Intelspeak (http://www.intel.com/intelpress/sum_mcp.htm)

Software developers can no longer rely on increasing clock speeds alone to speed up single-threaded applications; instead, to gain a competitive advantage, developers must learn how to properly design their applications to run in a threaded environment. Multi-core architectures have a single processor package that contains two or more processor ”execution cores,” or computational engines, and deliver-with appropriate software-fully parallel execution of multiple software threads. Hyper-Threading Technology enables additional threads to operate on each core.
Parallel Programming @2011

Intelspeak (http://www.intel.com/intelpress/sum_mcp.htm)

Software developers can no longer rely on increasing clock speeds alone to speed up single-threaded applications; instead, to gain a competitive advantage, developers must learn how to properly design their applications to run in a threaded environment. Multi-core architectures have a single processor package that contains two or more processor ”execution cores,” or computational engines, and deliver—with appropriate software—fully parallel execution of multiple software threads. Hyper-Threading Technology enables additional threads to operate on each core.

*Sounds like a description of Cray XMP. Are we back in 1982?*
In practice...

- Use MPI for multi-node parallelism
- Use either Pthread or OpenMP for multi-core
- Use either automatic vectorization or SIMD intrinsics for SIMD units
MPI

- Based on message-passing model
- Probably the model with lowest possible programming productivity
  - Each process can only access its local memory
  - Need to use library calls (usually on both sides of communication) to access remote data
  - “correct” parallel execution managed by barriers and handshakes
An non-MPI program

```c
int main(int argc, char *argv[])
{
    int n, i;
    double PI25DT = 3.141592653589793238462643;
    double pi, h, sum, x;
    n = 100000000;
    h = 1.0 / (double) n;
    sum = 0.0;
    for (i = 0; i < n; i ++){
        x = h * ((double)i - 0.5);
        sum += f(x);
    }
    pi = h * sum;
    printf("pi is approximately %.16f, Error is %.16f\n", pi, fabs(pi - PI25DT));
    return 0;
}
```
A MPI program

int main(int argc, char *argv[]) {
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(processor_name, &namelen);
    if (myid == 0) {
        n = 100000000;
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    }
}
h = 1.0 / (double) n;
sum = 0.0;
for (i = myid + 1; i <= n; i += numprocs){
    x = h * ((double)i - 0.5);
    sum += f(x);
}
mypi = h * sum;
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
if (myid == 0) {
    printf("pi is approximately %.16f, Error is %.16f\n", pi, fabs(pi - PI25DT));
}
MPI_Finalize();
return 0;
Problem with MPI

- Program becomes 2x or more longer...
- Each process need to figure out what it is supposed to do from its ID.
- Data layout and almost every control structure need to be changed
- mapping between local array and “global” array should be taken care of manually. Nothing is automatic
void node_domain::calc_accel()
{
    int myindex;
    mono int i,k;
    double dx[NDIM];
    acc[0] = acc[1] = acc[2] = 0.0;  potential = 0.0;
    myindex = (int) this - (int) &node[0];
for(i = 0; i < nbody; i++){
    double pot, rsq, rsqinv, rinv;
    if(myindex != i){
        rsq = eps2;
        for (k = 0; k < NDIM; k++){
            dx[k]=node[i].position[k]-position[k];
            rsq += dx[k]*dx[k];
        }
        rsqinv = 1.0/rsq;
        rinv = sqrt(rsqinv);
        pot=node[i].mass *rinv;
        potential+=pot;
        pot *=rsqinv;
        for (k=0; k<NDIM; k++) acc[k]+=pot*dx[k];
    }
}
Connection Machine C*

- Machine image: huge SIMD machine with local memory
- Programming model: virtual node. Programmers need not make mapping between physical processors and data structure.
- Communication is done through assignment operation.

You still need to rewrite your program, but it is not 2x longer.
The Connection Machine

1986 Thinking Machine Corporation
SIMD machine with up to 64k one-bit processors
2048 Weitek floating-point processors
around 10Gflops peak
Why things has changed?

• Because Thinking Machines Corporation went bankrupt in 1994

• Vector supercomputers and other parallel machines were all killed by first RISC processors and then x86 processors

• It was impractical to use parallel languages on a cluster of RISC processors
Why parallel languages are bad on clusters

- practical parallel languages are data-parallel languages
- memory hierarchy made efficient implementation of data-parallel languages very difficult
- high-latency, low-bandwidth network added more difficulty
What could we do?

Constraints:
Compared to floating-point performance,

- Relative bandwidth to off-chip memory will keep decreasing.
- Relative communication bandwidth will keep decreasing too.

So data-parallel languages will become even more unpractical.
What should we do?
What should we do?

...
Summary

- Until around early ’90s, there were “parallel languages” available on high-performance computers.
- They are no longer there.
- HPF is the last to die (still available on NEC parallel vector machines)
- Why did this happen? What could/should we do?