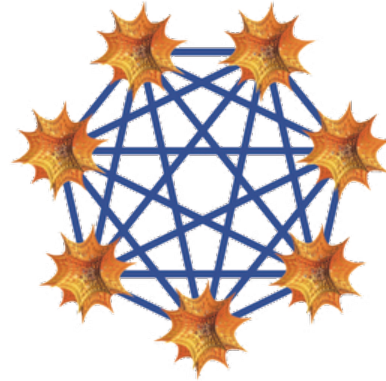


Supercomputing Engine for Mathematica an independent report

an independent researcher experienced with both high-performance computing and Wolfram Research's Mathematica introduces highly advantageous features of this new parallel computing software package



SEM Report 2008.11.17
Yuko MATSUDA
Tokyo Institute of Technology

Why Symbolic Computation

Symbolic computational languages like Mathematica have a big advantage when it comes to programming style. In a straightforward manner you can write your own ideas for solutions in many fields of study, such as differential equations, game theory, and so forth. By contrast, when you use typical procedural languages, like Fortran, C, or Java, you must translate your own ideas, as well as later incremental modifications, and adapt to their computer-oriented data structure and control structure.

Symbolic computational language is only one language which provides you “think-and-run” environments for users.

Why Mathematica

Mathematica is both a symbolic computational language and a functional language. This property affects the design concept of language Mathematica and the syntax of its programs. Most utilities and external interface to, for example, operating system are in almost every programming language. This style provides users readability, maintainability, and extendability of codes.

While Mathematica is a little bit slower than any other symbolic programming

languages, it has instead strong support for developing high quality programming results.

Why Parallel Computation

Symbolic computation needs much more memory and is memory-intensive, so in most cases it runs slower than any other procedural languages. Mathematica provides several high performance functions for to improve its efficiency, but it isn't always enough.

One solution is to parallelize these sequential codes. But how?

Why MPI

MPI (Message Passing Interface) is a de facto standard parallel library for almost every programming language. So many know how to implement using MPI, and MPI users easily apply this knowledge to modify their serial codes into parallel ones. Wolfram provides its own parallel functions and libraries. That is one way to support parallel work, but that approach is an exception.

Why SEM and Pooch

The SEM (Supercomputing Engine for Mathematica) with Dauger Research's Pooch technology provides the only MPI environment for Mathematica. It provides most of the commonly used MPI routines, with a syntax similar to the original MPI functions.

The following simple C program computes the summation of 1 through 7.

```
// summation of 1..7
int main(){
    int sum=0, n=7, i;
    for(i=1; i<n; i++){
        sum=sum+i;
    }
    printf("%d\n", sum);
}
```

The parallel version of the summation is written in MPI this way. MPI codes in C need many supporting details, such as type declaration for variables corresponding to these features and some housekeeping functions for MPI.

```
// MPI version in C
#include "mpi.h"
main( argc, argv )
int argc;
char **argv;
{
    int pid;
    int nproc;
    int sum, v;
    int i;
    MPI_Status status;
    MPI_Init( &argc, &argv );
        MPI_Comm_rank(MPI_COMM_WORLD,
&myrank );
        MPI_Comm_size(MPI_COMM_WORLD,
&nproc );

    if (myrank == 0){
        sum = 0;
        for (i=1; i<nproc; i++){
            MPI_Recv(&v, 1, MPI_INT, i, 99,
MPI_COMM_WORLD, &status);
            sum = sum + v;
        }
    }
    else{
        MPI_Send(&pid, 1, MPI_INT, 0,
99, MPI_COMM_WORLD);
    }

    if (pid == 0){
        printf("sum of all ranks = %d
\n", sum);
    }
    MPI_Finalize();
}
```

Finally, the MPI version of Mathematica is below. This is the whole code. For housekeeping of running MPI is managed by SEM technology, so user need not worry about any issues related to MPI running environment. For Mathematica users, SEM stands in an advantageous position compared to the Parallel Computing Toolkit (does not use MPI and adopts a Wait/Queue protocol) by Wolfram, which also makes users set up and link its process pool.

```
// MPI version in Mathematica
pid = $IdProc
If [pid == 0, sum = 0
    Do[mpiRecv[v, i, 99, $mpiCommWorld];
    sum = sum + v, {i, $NProc - 1}],
    mpiSend[pid, 0, 99, $mpiCommWorld]]
If [pid == 0 , Print[sum]]
```

Note: A better optimized solution than using Send and Receive functions for this problem exists.

For more details see:

<http://daugerresearch.com/pooch/mathematica>

<http://advclustersys.com>

© 2008 Advanced Cluster Systems
and Dauger Research, Inc.



Advanced Cluster
Systems



Dauger Research