Parallel Evaluation of Mathematica Programs in Remote Computers

Santanu K. Maiti and S. N. Karmakar

E-mail: *santanu.maiti@saha.ac.in*

Theoretical Condensed Matter Physics Division
Saha Institute of Nuclear Physics
1/AF, Bidhannagar, Kolkata 700 064, India

Abstract

Mathematica is a versatile equipment for doing numeric and symbolic computations and it has wide spread applications in all branches of science. *Mathematica* has a complete consistency to design it at every stage that gives it multilevel capability and helps advanced usage evolve naturally. *Mathematica* functions work for any precision of number and it can be easily computed with symbols, represented graphically to get the best answer. *Mathematica* is a robust software development that can be used in any popular operating systems and it can be communicated with external programs by using proper Math-link commands.

In this article we study about the parallel computations by using more than one computer on a network. It helps us to do large computational jobs at a time and hence the operations are so time consuming. To do parallel computation by using remote computers it is needed to open several *Mathematica* kernels in those machines. Parallel computation supports any version of *Mathematica* and thus if different versions are installed in different machines then parallel computation will be performed without any trouble. These kernels can run under any supported operating system like Unix, Linux, Windows and Macintosh. Here we describe the techniques for parallel evaluation of *Mathematica* jobs by using Unix and Linux computers.

Contents

1 Introduction

Mathematica is the world's best powerful computing system which has been released in 1988 and have profound evidences on the way of computations that are used in technical and other fields of work. The key intellectual aspect in *Mathematica* is the invention of a new kind of symbolic computation language that can manipulate the very wide range of objects needed to achieve the generality required for technical computing by using a very small number of basic primitives. *Mathematica* has wide spread applications in every branch of sciences-physical, biological, social, and other and has played a crucial role in many important discoveries and has been basis for thousands of technical papers.

Mathematica is a versatile and powerful package for calculating mathematics and publishing mathematical results. It can be used in almost all popular workstation operating systems like, Microsoft Windows, Apple Macintosh operating system, Linux and other Unix-based systems. As a programming tool *Mathematica* [1] provides a rich set of programming extensions. Programming can be done in different ways like functional, logical i.e., rule-based or procedural type or a mixture of all these three. The another most important aspect is that *Mathematica* provides Math-link [2, 3] which allows *Mathematica* programs to communicate with external programs written in C, Java, XL-Fortran languages or any other languages.

Mathematica is now emerging as an important tool in many branches of computing, and today it stands as the world's best system for general computation.

In the articles [3, 4] the detailed descriptions have been given how to start *Mathematica*, write programs, connect external programs those are written in a same computer with *Mathematica* where *Mathematica* notebook is operated and then run *Mathematica* programs in background through proper batch-files. To do large computations it is necessary to do parallel computation by using remote computers those are available in the network. Here we describe how remote machines are connected by using proper Math-link connections and then how to perform parallel jobs by using different *Mathematica* kernels.

2 Starting Kernels in Local Machine

To do parallel jobs by using remote machines first it is necessary to know how to use several *Mathematica* kernels in a local machine. In the *Mathematica* notebook initially the following package has to be loaded to do the parallel operations.

```
Needs["Parallel`Parallel`"]
```

and to enable optional features it is needed to load the following.

```
Needs["Parallel`Commands`"]
```

Now care should be taken always in the double quotations " ", like as used in the above two expressions.

To connect and open a Math-kernel in a local machine following command is used.

```
LaunchSlave["localhost", "math -noinit -mathlink"]
```

By using this command we can open several kernels from the master kernel, but it is useful to specify different kernels in different names so that one can run specific jobs at specific kernels without any trouble. Below we give some examples how different kernels can be started in different names.

```
link1=LaunchSlave["localhost", "math -noinit -mathlink"]
link2=LaunchSlave["localhost", "math -noinit -mathlink"]
link3=LaunchSlave["localhost", "math -noinit -mathlink"]
```

Here link1, link2 and link3 are three different kernel names.

Once the kernels open successfully it is necessary to know the details of the different kernels and these things are available by using the following command.

```
TableForm[RemoteEvaluate[{$ProcessorID, $MachineName, $SystemID,
$ProcessID, $Version}], TableHeadings→{None,{"ID", "host", "OS",
"process", "Mathematica Version"}}]
```

The output of the above command will be as follows (as an example).

ID	host	OS	process	Version
1	tcmpibm	AIX-Power64	463002	5.0 for IBM AIX Power (64 bit) (November 26, 2003)
2	tcmpibm	AIX-Power64	299056	5.0 for IBM AIX Power (64 bit) (November 26, 2003)
3	tcmpibm	AIX-Power64	385182	5.0 for IBM AIX Power (64 bit) (November 26, 2003)

The results shown in the tabular form are for the above three links (link1, link2 and link3) those are performed in the local machine, named 'tcmpibm'.

By using the following command the total number of kernels that are opened can be informed.

```
Length[$Slaves]
```

So in this case the total number of slaves is 3.

3 Starting Kernels in Remote Machine

Here we discuss about the kernel operations in remote machines by using ‘ssh’ command which offers secure cryptographic authentication and encryption of the communication between the local and remote machine. To perform the operation in remote kernels through ‘ssh’ first it is necessary to know whether ‘ssh’ is configured correctly or not. It can be checked by using the following command.

```
ssh remotehost math
```

Suppose we want to connect a remote machine, named ‘tcmpxeon’ then the command will be as,

```
ssh tcmpxeon math
```

If ‘ssh’ is properly configured then after giving proper password the command ‘In[1]:=’ will come.

Once ‘ssh’ is working, then by using the following command a kernel can be started on a remote (Unix or Linux) machine.

```
LaunchSlave[“remotehost”, “ssh -e none `1` math -mathlink”]
```

In below we set some examples how to start different kernels in remote computers.

```
link1=LaunchSlave[“tcmpxeon.saha.ac.in”, “ssh -e none `1` math -mathlink”]
link2=LaunchSlave[“tcmp441d.saha.ac.in”, “ssh -e none `1` math -mathlink”]
link3=LaunchSlave[“tcmpxeon.saha.ac.in”, “ssh -e none `1` math -mathlink”]
link4=LaunchSlave[“tcmp441d.saha.ac.in”, “ssh -e none `1` math -mathlink”]
```

Here link1, link2, link3 and link4 are four different kernels those are started at two different remote machines named as ‘tcmpxeon’ and ‘tcmp441d’. So in this way several remote kernels can be started in different remote machines those are available in the network.

The details of these several remote kernels used here are given in the tabular form as

ID	host	OS	process	Version
1	tcmpxeon	Linux	5137	5.0 for Linux (November 18, 2003)
2	tcmp441d	Linux	11323	5.0 for Linux (November 18, 2003)
3	tcmpxeon	Linux	5221	5.0 for Linux (November 18, 2003)
4	tcmp441d	Linux	11368	5.0 for Linux (November 18, 2003)

Here the total number of slaves is 4.

Thus we are familiar how to start different *Mathematica* kernels both in local and remote machines by using proper Math-link commands. So now we try to describe the way of doing parallel jobs through these kernels by giving an example.

4 Running of Mathematica Programs in Parallel Through Different Mathematica Kernels

This section focuses the technique of running *Mathematica* programs in parallel by using different *Mathematica* kernels. Here we set an example below to describe this parallel evaluation.

Problem: *Construct three square matrices at three different computers. One at local computer (tcmpibm), another one at a remote computer (tcmpxeon) and the last one at another remote computer (tcmp441d). Write the two matrices in the respective computers those are evaluated at remote computers and read these matrices in the local computer (tcmpibm). Finally take the product of these three matrices and calculate the eigenvalues of the product matrix at local computer.*

To solve this problem we have to do the following steps one by one in a *Mathematica* notebook.

Step-1 : First install the package for parallel computation and to get optional features install another package for it as mentioned earlier. Then open a kernel 'link1' (say) at the local computer 'tcmpibm', a kernel 'link2' (say) at 'tcmpxeon' and the other kernel 'link3' (say) at 'tcmp441d' by using proper Math-link commands as stated in previous sections.

Step-2 : Now write three programs in local computer for the construction of three separate square matrices.

I.

```
sample1[ns_]:=Block[{esi= 0, t = 1.2, p = 2.1,vacuum1={}},
Do[Do[a1=If[i==j,esi,0];
a2=If[i < j && Abs[i − j]== 1, t, 0];
a3=If[i > j && Abs[i − j]== 1, p, 0];
a4=a1+a2+a3;
a5 = AppendTo[vacuum1,a4], {j, 1, ns}], {i, 1, ns}];
a6 = Partition[a5, ns]]
```

II.

```
sample2[ns_]:=Block[{esi= 0, q = 2.6, r = 1.8,vacuum2={}},
Do[Do[a1=If[i==j,esi,0];
a2=If[i < j && Abs[i − j]== 1, q, 0];
a3=If[i > j && Abs[i − j]== 1, r, 0];
a4=a1+a2+a3;
a5 = AppendTo[vacuum2,a4], {j, 1, ns}], {i, 1, ns}];
a6 = Partition[a5, ns]]
```

III.

```
sample3[ns_]:=Block[{esi= 0, u = 2, v = 3,vacuum3={}},
Do[Do[a1=If[i==j,esi,0];
a2=If[i < j && Abs[i − j]== 1, u, 0];
a3=If[i > j && Abs[i − j]== 1, v, 0];
a4=a1+a2+a3;
a5 = AppendTo[vacuum3,a4], {j, 1, ns}], {i, 1, ns}];
a6 = Partition[a5, ns]]
```

Now try to describe something about these three programs. In these programs the variable 'ns' gives the order of the square matrix. 'vacuum1={}', 'vacuum2={}' and 'vacuum3={}' are the empty lists used in these programs respectively, where the result for each 'Do' loop operation is written. Partitioning the list 'a5' by 'ns' a square matrix of order 'ns' is formed which is expressed by 'a6'.

So our programs for the construction of three square matrices are completed and now we are going to discuss how the eigenvalues of the product matrix can be calculated in the local computer (tcmpibm) by evaluating one square matrix at the local computer and the other two square matrices in remote computers.

Step-3 : The final program needed for the parallel evaluation is written in the local computer and the structure of the program is as follows.

```
sample4[ns_]:=Block[{},
ExportEnvironment["Global`"];
mat1 = sample1[ns];
RemoteEvaluate[Export["data1.dat", sample2[ns]], link2];
RemoteEvaluate[Export["data2.dat", sample3[ns]], link3];
mat2 = RemoteEvaluate[ReadList["data1.dat", Number, RecordLists→ True],
link2];
mat3 = RemoteEvaluate[ReadList["data2.dat", Number, RecordLists→ True],
link3];
mat4 = mat1.mat2.mat3;
Chop[Eigenvalues[mat4]]]
```

This is our complete program. Now explain the meaning of several commands those are used in this program. The 2nd line of the program is used to inform all the symbols and definitions to the remote kernels. The 3rd line evaluates one square matrix at local computer 'tcmpibm' which is written in 'mat1', while, the 4th line performs the matrix operation for another matrix at 'tcmpxeon' and writes the output of the matrix in 'data1.dat'. Similarly, the 5th line does for the other matrix at 'tcmp441d' and writes it in 'data2.dat'. By using the command 'ReadList' datas can be transported outside from the *Mathematica* notebook to the *Mathematica* notebook. Here 'mat2' and 'mat3' store the matrices from the respective remote computers in the local computer, those are performed by using proper 'ReadList' command as stated in the program. 'mat4' gives the product of the three matrices (mat1, mat2 and mat3) and then eigenvalues are calculated by using the command 'Eigenvalues[mat4]'.

So now by running the program 'sample4[ns]' in the local computer eigenvalues of any square matrices can be obtained.

After the completion of all parallel computations, we should stop all remote kernels before exiting the local *Mathematica* kernel and front end by using the following command.

```
CloseSlaves[ ]
```

Concluding Remarks

In conclusion, we have described initially about the techniques for starting several kernels in local computer as well as in remote computers by using proper Math-link connection commands. Next we have discussed about the way of writing programs for parallel computations in *Mathematica* and how to perform it by using different remote computers. To do large computational operations either numerical or analytical or both, it is necessary to use *Mathematica* and also sometimes it is needed to use parallel computations for time consumption. This article clearly describes the way of starting several *Mathematica* kernels and running *Mathematica* programs in parallel by using remote computers those are available in the network.

References

[1] Stephen Wolfram. *Mathematica-5.0*.

[2] Roman E. Maeder. *About Parallel Computing Toolkit*. A Wolfram Research Application Package.

[3] Santanu K. Maiti. *A Basic Introduction on Math-Link in Mathematica*, Ref. cs.MS/0603005.

[4] Santanu K. Maiti. *How to Run Mathematica Batch-files in Background* ?, Ref. cs.MS/0604088.