Using Premia and Nsp on a Parallel Architecture for Risk Management Benchmark

Jean-Philippe Chancelier, Bernard Lapeyre, Jérôme Lelong

Université Paris-Est, CERMICS, École des Ponts
and
École Nationale Supérieure de Techniques Avancées

Pdcof 2009
Introduction

Premia: a library for numerical computations in finance

Nsp

Nsp toolboxes

Practical experiments

Conclusion
Context of risk evaluation

- Banking legislation imposes to financial institutions the daily evaluation of the risk they are exposed to.
- Banks own very large portfolios of contingent claims (several thousands of claims).
- For a given contingent claim, the price evaluation requires a computation time from a few milliseconds to dozens of minutes.
- A huge number of independent computations (around $10^6$) is necessary to evaluate the risk of the whole portfolio.
- These computations must be carried out on a daily basis (batch code every night).
A proposed software environment

- Being able to have a free access to a realistic portfolio descriptions (models and parameters) would be useful for benchmarking parallel architectures.
- Unfortunately, for obvious confidentiality, no such information exists.
- Moreover, algorithms for portfolio evaluation are seldom available in a unified package for free.

We propose a software architecture for constructing realistic models and portfolios based on freely available softwares:

- Premia: a library used to compute financial products prices.
- Mpi: to control parallelism.
- Nsp: provides a unified access to MPI and Premia primitives.
Premia: A library

- A library devoted to the computation of prices and hedges for derivatives. A major issue for financial institutions.
- Developed by the MATHFI project: A research team involved in numerical methods for probability and finance from INRIA and ENPC.
- Keeps track of the most recent advances in computational finance in a well-documented way.
- Focuses on the implementation of numerical analysis techniques, probabilistic and deterministic methods.
- Provides an important entry point for numerical algorithms in Finance.
- A powerful testing platform for comparing numerical methods.
- A fairly complete library with regards to what is currently used in advanced financial mathematics.
Premia: A consortium

- Developed in interaction with a consortium of financial institutions: Calyon, Natixis, Société Générale, Raiffeisen Zentralbank, Bank Austria.
- The members support the development of Premia.
- The members help to determine the directions in which the project should evolve.
Nsp

- A Matlab-like Scientific Software Package, GPL license.
- High-level programming language: scripting language or programming language.
- Gives an easy access to efficient numerical routines.
- Online help, Gui and graphics facilities.
- Extendable: glue code called interfaces can be used to dynamically embed an external library into Nsp.
- Shares many paradigms with other Matlab-like Scientific Softwares as for example: Matlab, Octave, ScilabGtk and also with scripting languages such as Python for instance.
- Two typical toolboxes for this work:
  - Nsp Premia toolbox gives access at Nsp level to Premia.
  - MPI interface, gives at Nsp level access to mainly all MPI-2 functions.
MPI toolbox for Nsp

- Provides a direct access to MPI functions within the Nsp scripting language.
- Thus, gives an easy way to get familiar to MPI functions which can be tested interactively.
- Hides the tedious work of packing and unpacking complex data.
- Similar toolboxes are available: Mpitb provides such a full MPI interface for the Matlab and Octave languages.
An example

It is possible to launch a master Nsp and then to spawn slaves Nsp, using the MPI_Comm_spawn primitive.

MPI_Init();
COMM = mpicomm_create('SELF');
INFO_NULL=mpiinfo_create('NULL');

cmd = "exec('src7.x/loader.sce');MPI_Init();";
cmd = cmd + "parent=_MPI_Comm_get_parent();";
cmd = cmd + "[NEWORLD]=MPI_Intercomm_merge(parent,1);";

nspace = getenv('NSP') + '/bin/nsp';
args=['-name', 'nsp-child', '-e', cmd];
children, errs = MPI_Comm_spawn(nspace, args, 1, INFO_NULL, 0, COMM);
// child will execute cmd

[NEWORLD] = MPI_Intercomm_merge (children, 0);
Example continued

- The previous code starts a new Nsp which will execute the transmitted cmd to start interacting with the master through a merged communicator.

- The interface between Nsp and MPI involves functions and also new Nsp objects devoted to MPI. `mpicomm_create` creates a Nsp communicator object which internally contains a MPI communicator.

- Since starting a set of Nsp slaves is a classic task, the previous given code can be written in a Nsp function `NSP_spawn` and it is then possible to start n slaves by the simple Nsp command:

  ```
  NEWORLD=NSP_spawn(n);
  ```
Sending/receiving data

Nsp objects exchanged with MPI_Send_Obj and MPI_Recv_Obj.

- Directly for basic objects:
  
  ```
  -nsp->A=list('string',%t,rand(4,4));
  -nsp->MPI_Send_Obj(A,rank,TAG,MCW)
  ...
  -nsp->B=MPI_Recv_Obj(rank,TAG,MCW)
  ```

- Using Nsp serialization for complex objects:
  
  ```
  -nsp->A=sparse(rand(2,2));S=serialize(A);
  -nsp->MPI_Send_Obj(S,rank,TAG,MCW)
  ...
  -nsp->B=MPI_Recv_Obj(rank,TAG,MCW);
  -nsp->B.equal[A]
  ans = b (1x1)
    | T |
  ```
Premia toolbox for Nsp

Embedding Premia into Matlab-like Scientific Software provides two ways of accessing the library:

- through the scripting language. Then premia can interact with other toolboxes
- using the graphical capabilities of the software (Gui).

Since the license of Premia gives right to freely distribute the version of Premia two year older that the current release. Using a free Scientific Software was important.

The internal class system of Nsp enables to easily add new objects in the interpreter. This is how we introduced a new type named PremiaModel.

Through the PremiaModel objects the wide range of Premia pricing problems are made available from Nsp.
Valuation of a portfolio

- For practitioners, the daily valuation of a complex portfolio is a burning issue: a possible answer is to use MPI/Nsp/Premia.
- A pricing problem in Nsp is a `PremiaModel` instance. It can be created by script or by Gui. It can be saved, loaded to file and serialized as all Nsp objects.

```python
P = premia_create()
P.set_asset[str="equity"]
P.set_model[str="Heston1dim"]
P.set_option[str="PutEuro"]
P.set_method[str="MC_Alfonsi"]
save('fic', P)
```

- A list of problems can be described by a list of files.
- Using MPI toolbox: A master reads files, creates instances of `PremiaModel` class and send serialized instances to slaves.
Practical experiments

- A basket of 60 American put options in dimension 3
- Maturities varying from 1 month to 5 years by steps of one month.
- Scheduler written in Nsp/Mpi: First, the master sends one job to each slave and as soon as a slave sends the answer back, it is assigned a new job. This mechanism goes on until all the jobs have been treated.
- When using several node, we have made sure that the master is located on a different machine from the slaves. This ensures that when comparing the speed-ups are taking into account the time spent exchanging data.
### Speed up table

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>CPU time</th>
<th>speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>182</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>89.8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>45.4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>26.79</td>
<td>6.7</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>8.7</td>
</tr>
<tr>
<td>14</td>
<td>18.3</td>
<td>9.9</td>
</tr>
<tr>
<td>19</td>
<td>12.7</td>
<td>14.3</td>
</tr>
</tbody>
</table>
Nsp script

```plaintext
if ~MPI_Initialized() then   MPI_Init(); end
MPI_COMM_WORLD=mpicomm_create('WORLD');
[rank] = MPI_Comm_rank (MPI_COMM_WORLD);
[size] = MPI_Comm_size (MPI_COMM_WORLD);
SLV = (rank <> 0)
MST = ~ SLV;
TAG=4;

if SLV
   while %t then
      Maturity=0;
      MPI_Recv (Maturity,-1,TAG,MPI_COMM_WORLD); // receives the vector
      if Maturity < 0 then break; end
      exec ('premia.sce');
      result= [rank,Maturity,L(3)(3)];
      MPI_Send(result,0,TAG,MPI_COMM_WORLD); // sends the results back
   end
else // Here at master
   Nt = 60;
   nb_per_node = floor (Nt / (size-1));
   Maturities = linspace(0.25,10,Nt);
```
Nsp script continued

```matlab
for slv=1:size-1
    MPI_Send (Maturities(slv),slv,TAG,MPI_COMM_WORLD);
end
Maturities(1:size-1)=[];
res=[];
result=ones(1,3);
while %t
    MPI_Recv(result,-1,TAG,MPI_COMM_WORLD);
    sl=result(1);
    res=[res;result];
    if isempty(Maturities) then
        MPI_Send (Maturities(1),sl,TAG,MPI_COMM_WORLD);
        Maturities(1)=[];
    else
        break;
    end
end
for slv=1:size-2    // we still have size-2 Recv to perform
    MPI_Recv(result,-1,TAG,MPI_COMM_WORLD);
    res=[res;result];
end
for slv=1:size-1    // tell the slaves to stop working
    MPI_Send([-1],slv,TAG,MPI_COMM_WORLD);
end
save(’matu.bin’,res,CPU=t);
end
MPI_Finalize(); // finalize slaves and master
```
Speed up table for a basket of 792 claims

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>CPU time</th>
<th>speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>No MPI</td>
<td>524</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>525</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>180</td>
<td>2.9</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>4.4</td>
</tr>
<tr>
<td>8</td>
<td>93</td>
<td>5.6</td>
</tr>
<tr>
<td>10</td>
<td>78</td>
<td>6.7</td>
</tr>
<tr>
<td>15</td>
<td>62</td>
<td>8.5</td>
</tr>
</tbody>
</table>
Nsp script

```plaintext
if ~MPI_Init() then   MPI_Init();end
MPI_COMM_WORLD=mpicc('WORLD');
[mpi_rank] = MPI_Comm_rank (MPI_COMM_WORLD);
[mpi_size] = MPI_Comm_size (MPI_COMM_WORLD);

SLV = (mpi_rank <> 0)
MST = ~ SLV;
TAG = 4;

function send_premia_pb( name, slv )
  load(name);
  MPI_Send_Obj (name,slv,TAG,MPI_COMM_WORLD);
  MPI_Send_Obj (P,slv,TAG,MPI_COMM_WORLD);
endfunction

exec('

premia_init()

if SLV // All slaves send result back
  while %t then
    name = MPI_Recv_Obj(0,TAG,MPI_COMM_WORLD); // receives the name
    if name == ' then break; end
    P=MPI_Recv_Obj (0,TAG,MPI_COMM_WORLD); // receives the object
    P.compute[];
    L = P.get_method_results[];
    result = list(mpi_rank,name,L(1)(3));
    MPI_Send_Obj(result,0,TAG,MPI_COMM_WORLD); // sends the results back
  end
```
Nsp script continued

```matlab
else // Here at master
    Lpb = glob(PBDIR + '/*.bin');
    Nt= size(pb_list, '*');
    nb_per_node = floor (Nt / (mpi_size-1));
    for slv=1:mpi_size-1 send_premia_pb (Lpb(slv), slv); end
    Lpb(1:mpi_size-1)=[];
    res=list();
    while %t
        result = MPI_Recv_Obj(-1,TAG,MPI_COMM_WORLD);
        sl= result(1);
        res.add_last(result);
        if isempty(Lpb) then send_premia_pb (Lpb(1), sl); Lpb(1)=[]; else break; end
    end
    // we still have mpi_size-2 Recv to perform
    for slv=1:mpi_size-2 // collect in any order
        result = MPI_Recv_Obj(-1,TAG,MPI_COMM_WORLD);
        res.add_last(result);
    end
    for slv=1:mpi_size-1  // tell slaves to stop working
        MPI_Send_Obj([''],slv,TAG,MPI_COMM_WORLD);
    end
    save('pb-res.bin',res);
end
MPI_Finalize();
```
Conclusion

- A web site http://cermics.enpc.fr/~lelong/gcpmf contains links to the tools described here.
- It contains the portfolio generator, some proposed portfolios benchmarks and test results.
- A test portfolio containing one instance of each possible Premia problems is also given.
- This web page will be regularly updated as we have more results from our experimentations on clusters.
- Our aim is to provide using Nsp syntax the descriptions of large portfolios of options which are representative of the computations carried on by banks to evaluate their risk level as they are required to do by the law.