An Overlapping Compiler for Planguage (PC) of Parallel Program

Poster: Yingqun Wang
Advisor: Dr. Ernesto Gomez

Department of Computer Science
California State University San Bernardino
Background (PC)

- PC is an extension of C and one part of the Planguage.
- PC compiler already supports MPI (The Message Passing Interface Standard), PVM (Parallel Virtual Machine) and IPSC.
- It simplifies the writing of message passing code for multiprocessors with distributed memory.

Two new operators: ‘@’ and ‘{}’ ease the task of writing parallel programs harness massively. Code comparison of PC and MPI is following:

```c
x=y@p;
<=====
if (me==p) MPI_Send(buffer, length, MPI_BYTE, i, type,
                  MPI_COMM_WORLD);
else MPI_Recv(buffer, length, MPI_BYTE, node, type,
             MPI_COMM_WORLD, MPI_status);
```
Background (SOS)

- **SOS**: Streams, Overlapping communications and Shortcutting function library built on top of MPI.
  - **Stream**: Set of programs executing along the same control path in parallel program.
  - **Overlapping**: Interval between definition and use at one process overlap the interval at another process. Overlapping schedule communication within this interval to avoid synchronization wait by place communication initiation as close as possible to data definition and communication termination.
  - **Shortcut**: One process can tell other processes to quit if they have same testing condition.
- **Advantage**: Speedup for parallel processing of problems with irregular or asymmetric computation.
Project Purpose

- Improving the PC compiler by add new expression analysis. Previous PC compiler permit one ‘@’ operator at right hand side of expression. The new compiler will recognize the expression like this:

  \[ x = y@p1 + z@p2; \]

- Extending PC compiler to support SOS. Incorporate automatic analysis and placement of overlapping function of SOS calls into codes generated by the PC language as optimization.
PC program is pre-scanned to find communication variables. PC compiler generates C code with SOS overlapping functions (C-SOS). The C-SOS code is compiled by native C compiler linking with SOS API library and PC API library to create executable program.
Implementation (Pre-scan)

• Purpose:

SOS system build up a queue collecting all variables transferred between processors. To identify the variable of the queue an integer called index is used. All indexes are global variables eliminating parameters augment when communication variables are passed between subroutines. To declare global index of them, communication variables must be known beforehand.

• Design:

- Pre-scan module is designed using LEX analysis.
- The PC program is scanned by pre-scan program, all variables whose right side are ‘@’ operator are communication variables. For example, only ‘x’ in next primary expressions are desired.

  \( x@p, x[k]@p. \)
Implementation (grammar)

• Locate appropriate place to implant SOS function call.
  
  1. SOS system must be initialized before any SOS function call.
  2. SOS index variables are declared as global and initialized after communication variables declaration.
  3. To guarantee the completion of communication, the "sosdorh_()" and "sosdolh_()" must be called before referring any communication variables. For example:
     
     sosdolh_(i); for (i=0;i<5;;){i=i@1; sosdorh_(i);} 

• Improve compiler by permitting two ‘@’ operators appear at the right hand side of expression.
Results

- The compiler is built successfully because it passes the test of programs.
- Ten varieties communication are tested.

<table>
<thead>
<tr>
<th>Communication</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple broadcast</td>
<td>a=b@ip;</td>
</tr>
<tr>
<td></td>
<td>a=b@[ (THISNODE+1)%NUMNODE];</td>
</tr>
<tr>
<td>Complex broadcast</td>
<td>a=b@ip1 + c@ip2;</td>
</tr>
<tr>
<td>One array broadcast</td>
<td>a[0:10]=b[0:10]@ip;</td>
</tr>
<tr>
<td>Array broadcast</td>
<td>c[0:4][0:4]=d[3:7][5:9]@ip;</td>
</tr>
<tr>
<td>Simple copy</td>
<td>a@ip1=b@ip2;</td>
</tr>
<tr>
<td>One array copy</td>
<td>a[0:10]@ip1=b[0:10]@ip2;</td>
</tr>
<tr>
<td>Array copy</td>
<td>a[0:5][0:5]@ip1= b[0:5][0:5]@ip2;</td>
</tr>
<tr>
<td>Limited broadcast</td>
<td>x@[0:5] = y@1;</td>
</tr>
<tr>
<td>Limited array broadcast</td>
<td>c[0:10][5:15]@[1:2]=</td>
</tr>
<tr>
<td></td>
<td>d[10:20][0:10]@ip;</td>
</tr>
<tr>
<td>Reduce</td>
<td>i1 = +{i1};</td>
</tr>
<tr>
<td></td>
<td>i2 = {*}i2;</td>
</tr>
</tbody>
</table>

September 2003
Conclusion

- The project is successfully augmenting the PC compiler by combining SOS overlapping functions.
- The new version of compiler can benefit the users by brief PC code and taking SOS advantage into parallel program.
- The current compiler will be evolved by implementing shortcut and stream functions.