

Interactive Parallelizing Assistance Tool for OpenMP: iPat/OMP

Fifth European Workshop on OpenMP
EWOMP `03
Aachen University, Germany
September 22-26, 2003

✦ Makoto Ishihara (*1),
Hiroki Honda (*1),
Toshitsugu Yuba(*1),
Mitsuhisa Sato(*2)

*1: The Graduate School of Information Systems, University of Electro-Communication

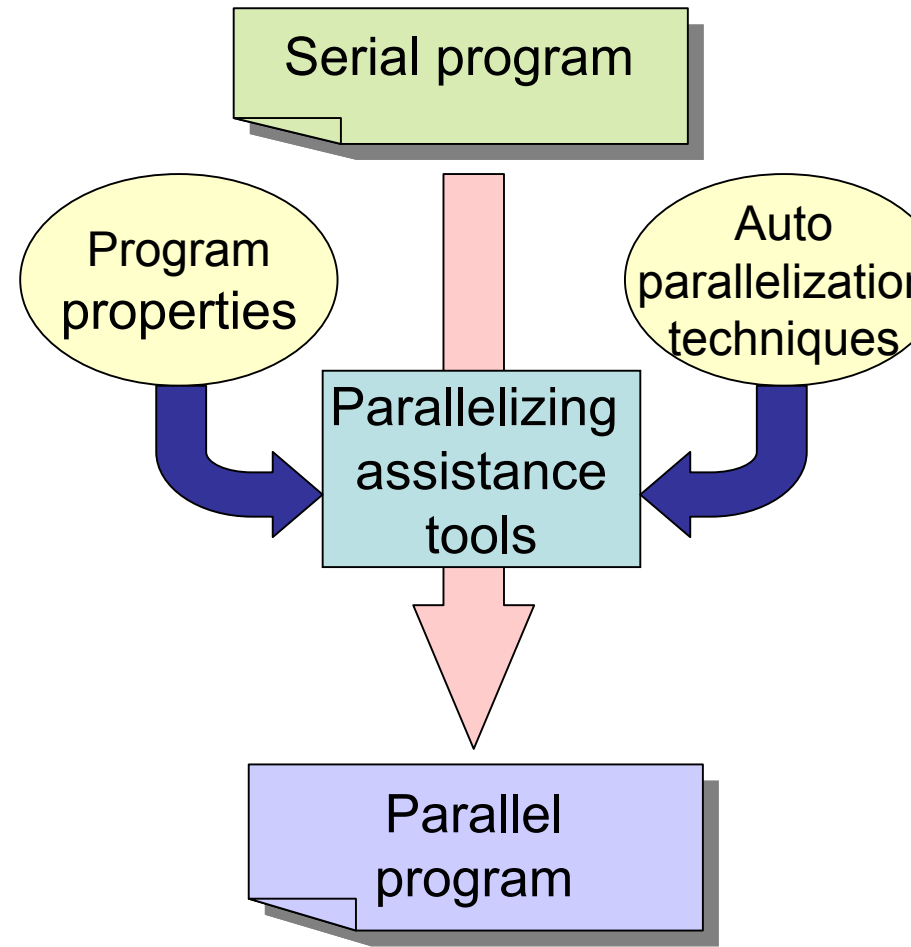
*2: Institute of Information Sciences and Electronics, University of Tsukuba

Table of contents

1. Background
2. Capabilities and design concepts
3. Current implementation status
4. Demonstration
5. Summary and future work

Background

- Parallelizing assistance tools combine:
 - Auto parallelization techniques
 - Human knowledge of the program properties



Related works

- URSA MINOR / INTERPOL:
 - Shows:
 - Static analysis results
 - Performance analysis data
 - Enables manual parallelizing
- CAPO:
 - Based on CAPTools
 - Shows dependence analysis results
 - Inserts OpenMP directives

Development

- Developing an interactive assistance tool which assists manual program parallelization using OpenMP.
 - This tool was named **iPat/OMP**
- Features of iPat/OMP:
 - Shows static analysis results and performance analysis data
 - Enables manual parallelizing
 - Allows the user to execute parallelizing assistance functions in a commodity editor environment

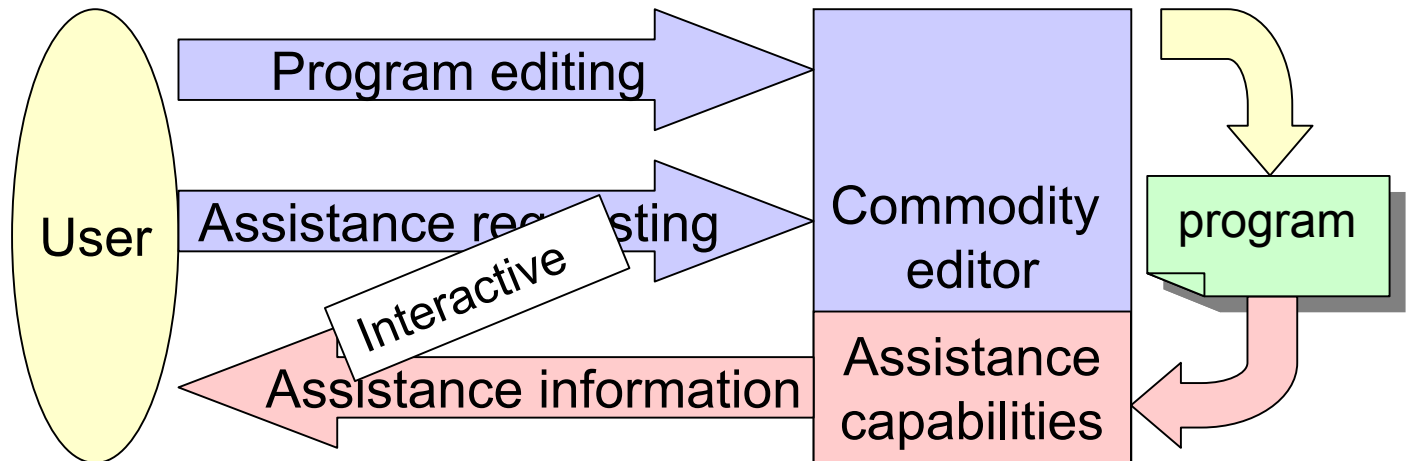
Parallelizing assistance capabilities of iPat/OMP

- Parallelism analysis capability:
 - analyzing the parallelism of a program
 - showing the results
- OpenMP directives creation capability:
 - creating the OpenMP directive required for parallelization
 - showing the directives
- Program restructuring capability:
 - restructuring the programs which can benefit from parallelization or optimization
- Execution time analysis capability:
 - analyzing execution time
 - showing the results

Design concepts

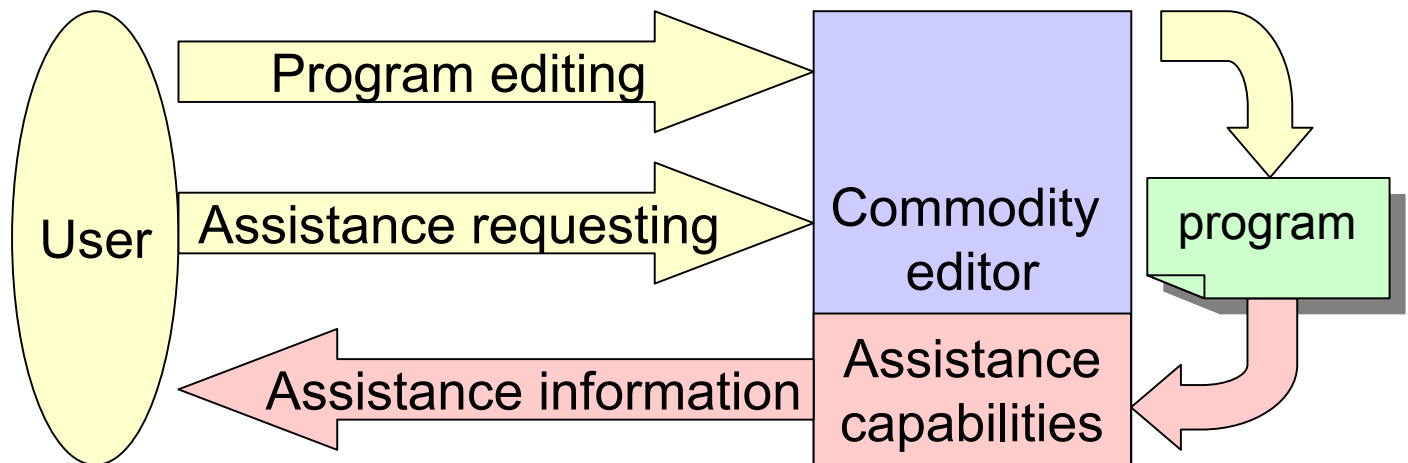
Editor-embedded system

- Assistance capabilities are embedded within a commodity editor.
 - In the same environment, the user can carry out:
 - Program editing
 - Interactive program parallelization



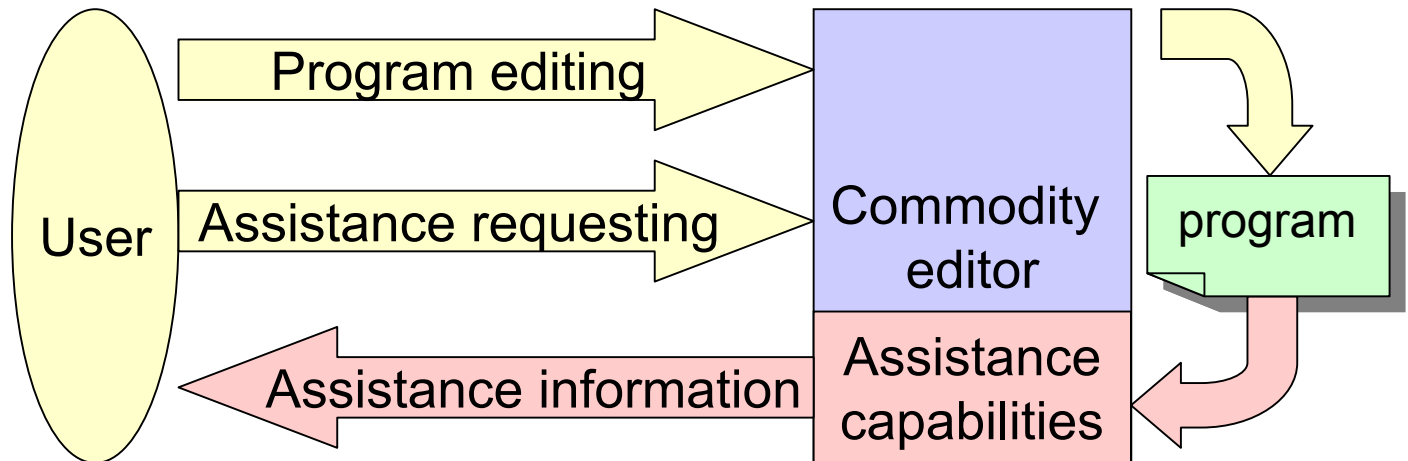
Role assignments

- iPat/OMP:
 - analyzes a program upon request
 - shows the analysis results to the user .



Role assignments

- The user:
 - invokes the program analysis of iPat/OMP
 - carries out:
 - Manual modification of the program
 - Parallelization of the program

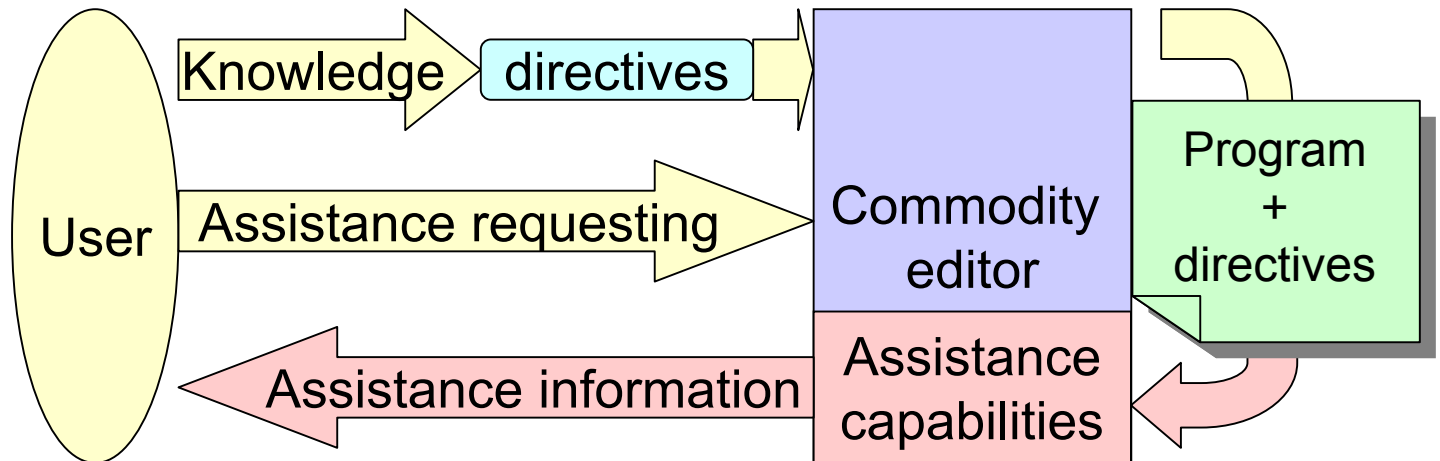


Expected results of role assignments

- Maintaining human-readability of the code
 - The OpenMP program enables the user to understand the source code easily.
- Selecting an appropriate target
 - Selection of the target section by the user makes it possible to prevent unnecessary sections from being parallelized.

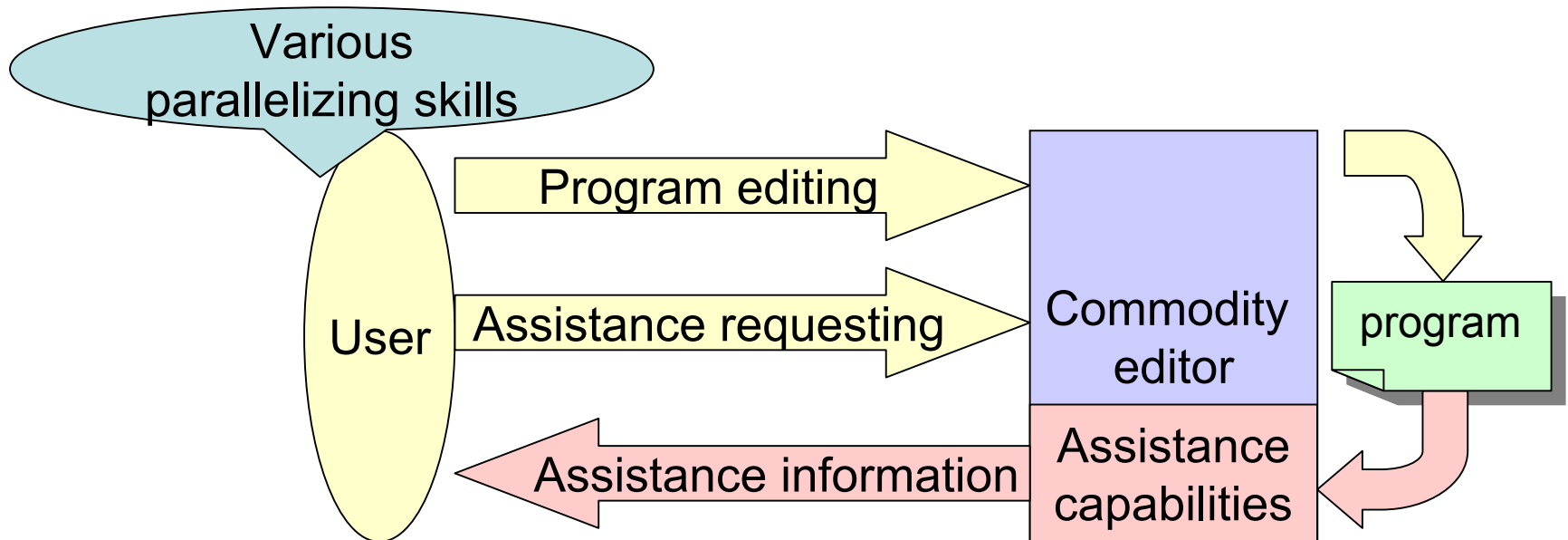
Interaction with the user

- iPat/OMP obtains knowledge of the program properties from the user.
- The user provides the knowledge via directives.
 - iPat/OMP analyzes the program based on this knowledge.



Target users

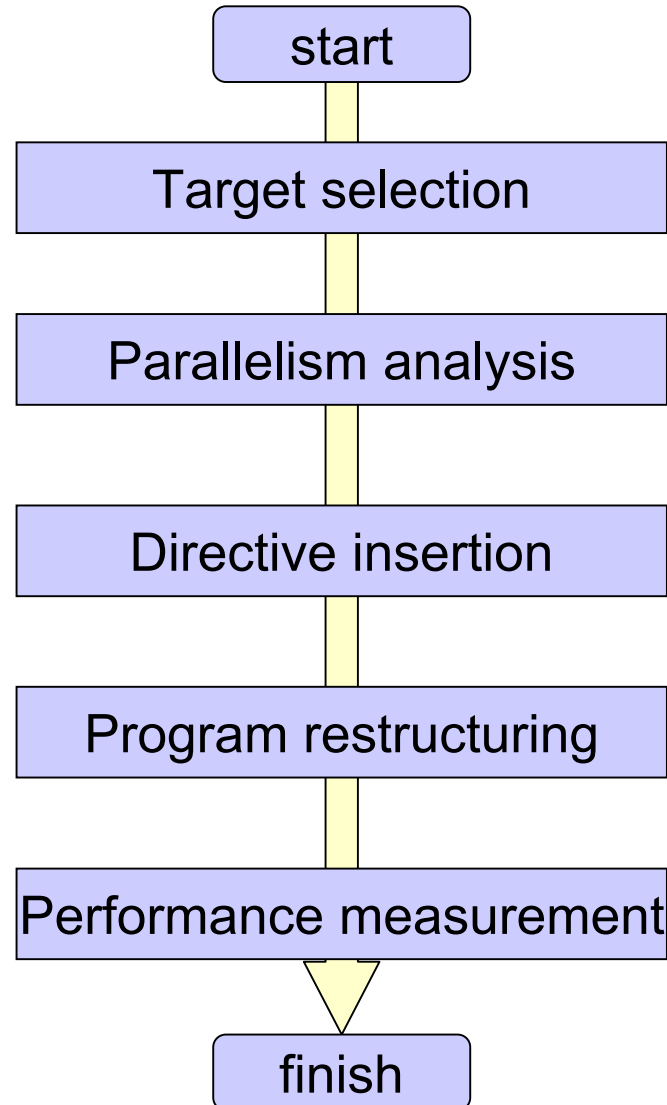
- iPat/OMP assists users of various skill levels:
 - either parallel programming beginners or experts
- iPat/OMP assists the user according to the parallel programming skill of the user.



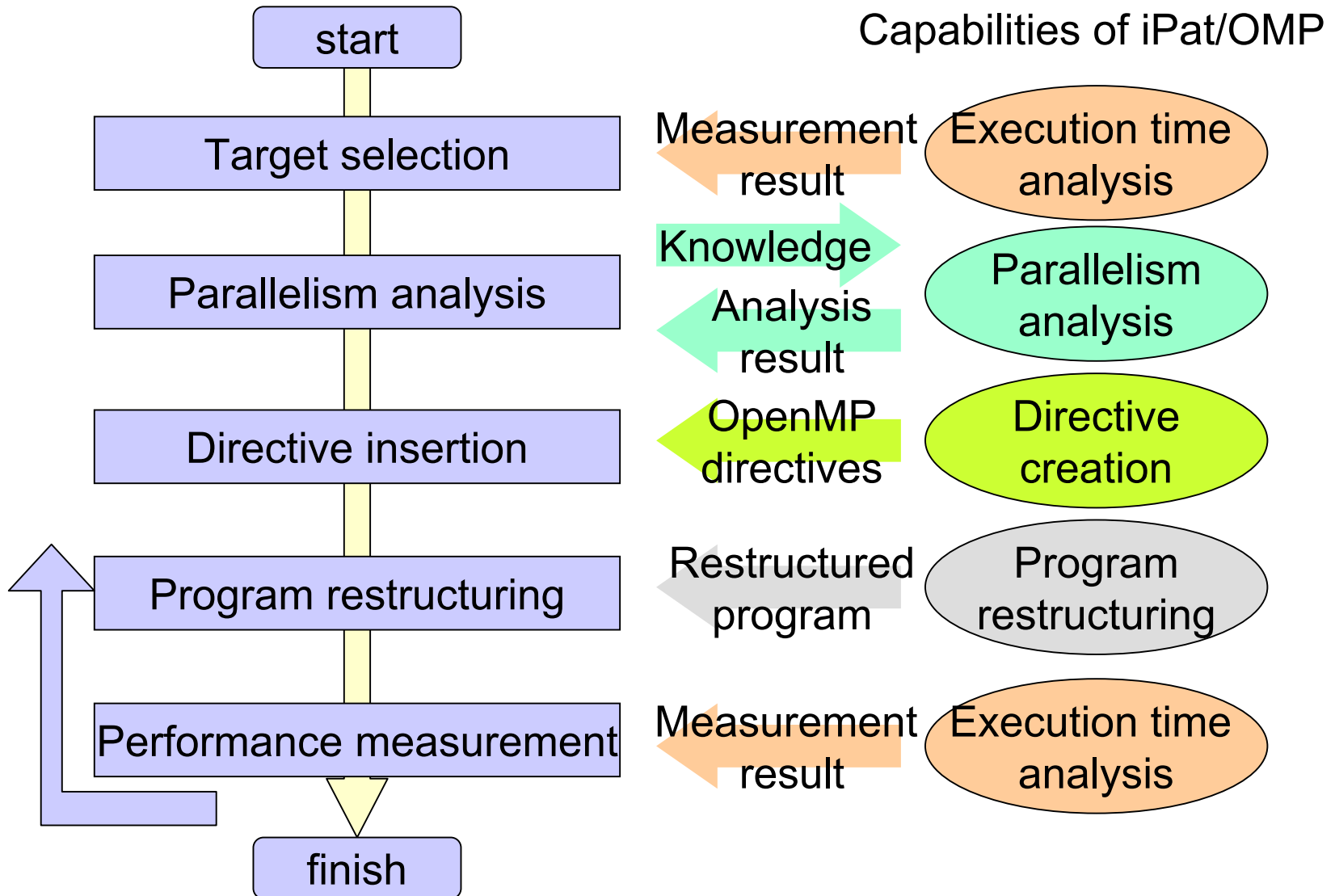
Portability and extendability

- Portability:
 - iPat/OMP is implemented without libraries that are limited to a particular OS environment.
- Extendability:
 - A developer can add a new capability easily.

Parallelization step

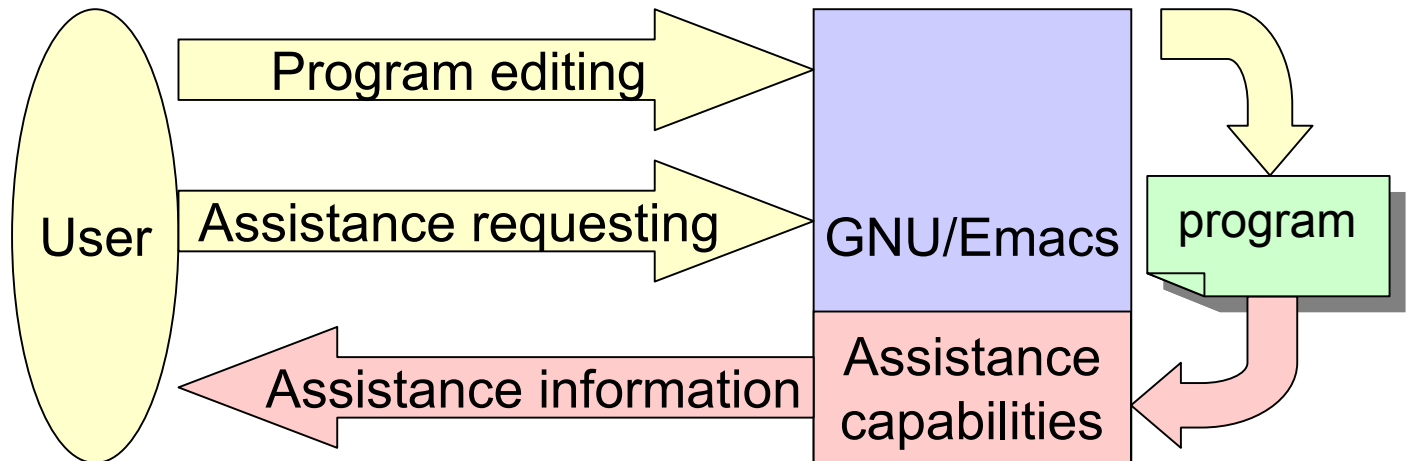


Parallelization step



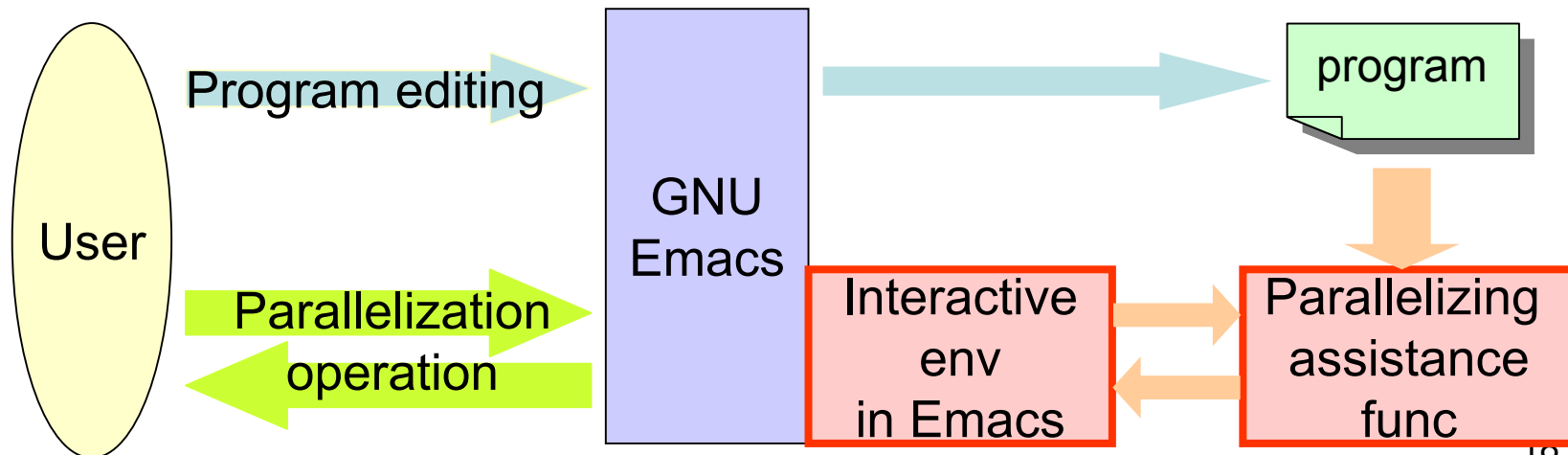
Implementation

- Beneficial features of GNU/Emacs:
 - Widely used free-software
 - Powerful editing functions
 - High extendability by using Emacs Lisp

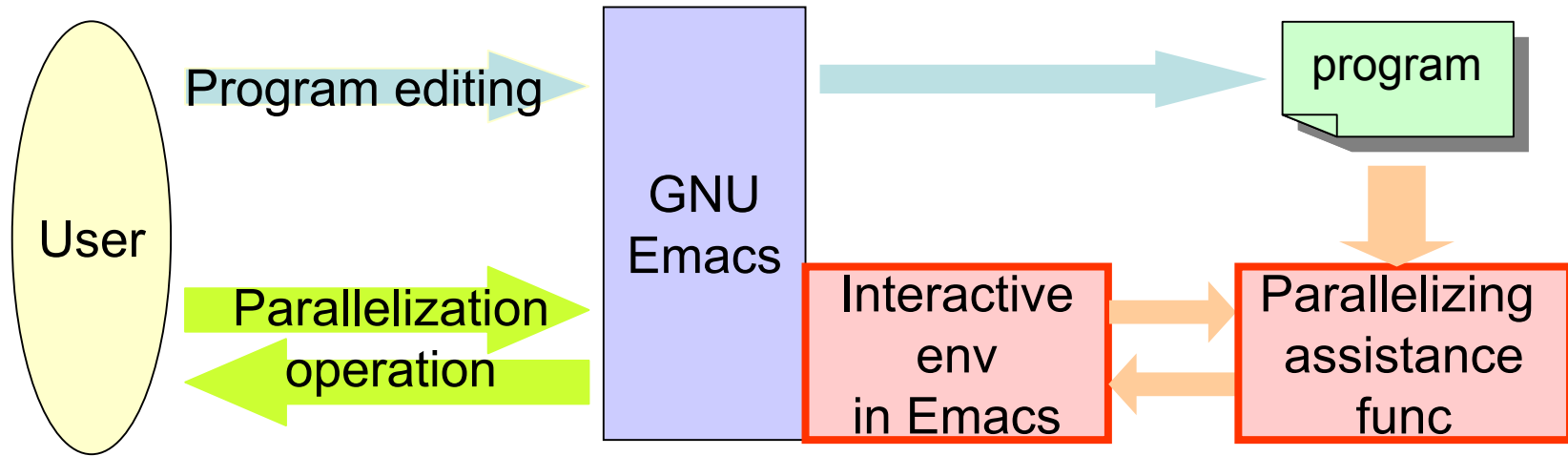


Components

- iPat/OMP consists of two sub-components:
 - Parallelizing assistance functionalities:
 - analyzing the program and showing the results
 - Interactive environment in Emacs
 - implemented using Emacs Lisp
 - enables the user to invoke parallelizing assistance capabilities on Emacs interactively

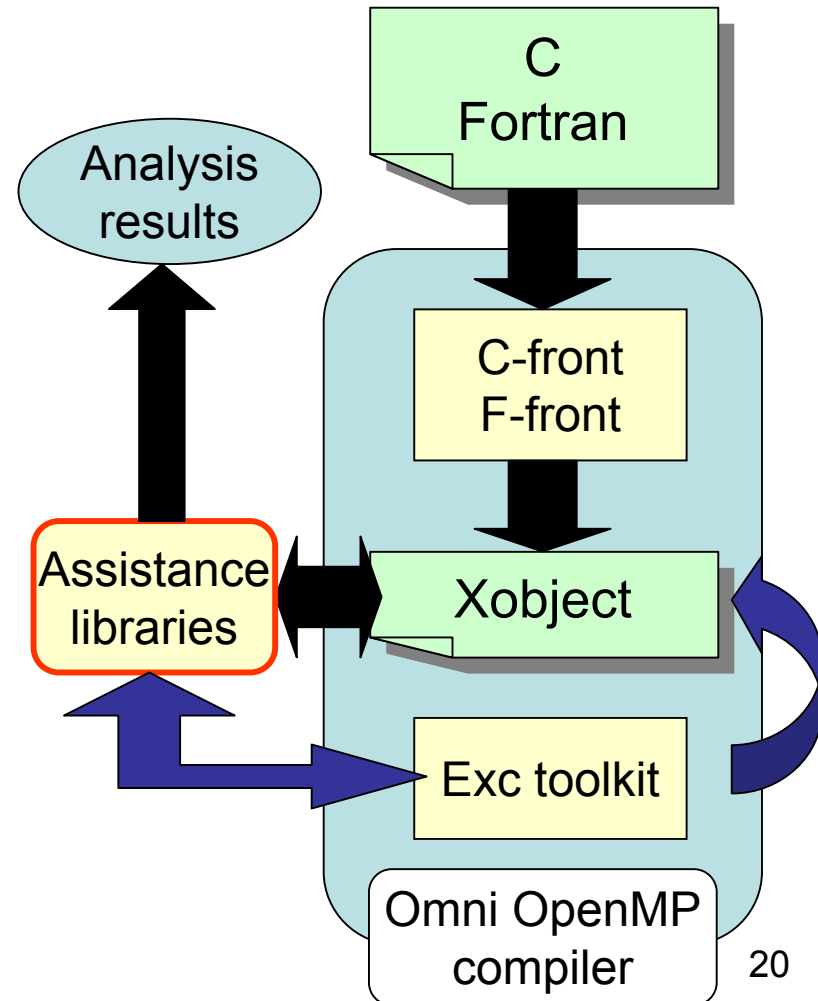


Parallelizing assistance functionalities



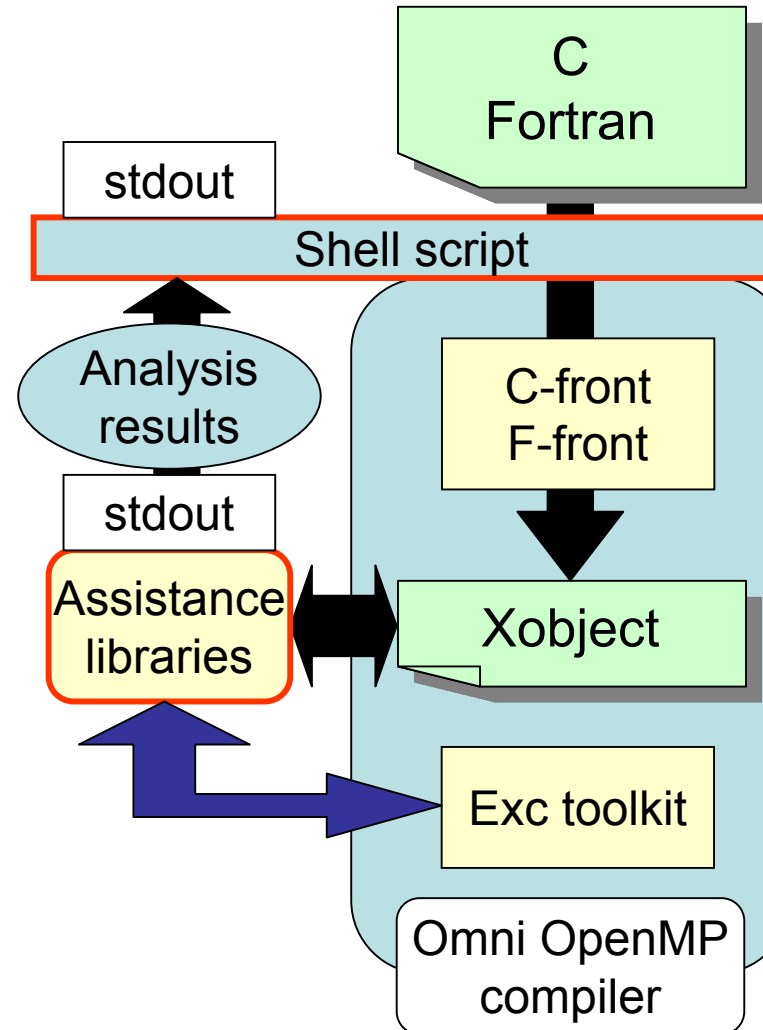
Parallelizing assistance functionalities

- iPat/OMP uses the Omni OpenMP compiler.
- The three components of the Omni OpenMP compiler are:
 - C/F-front
 - Xobject
 - Exc toolkit
- The assistance libraries use the Exc toolkit.



Parallelizing assistance functionalities

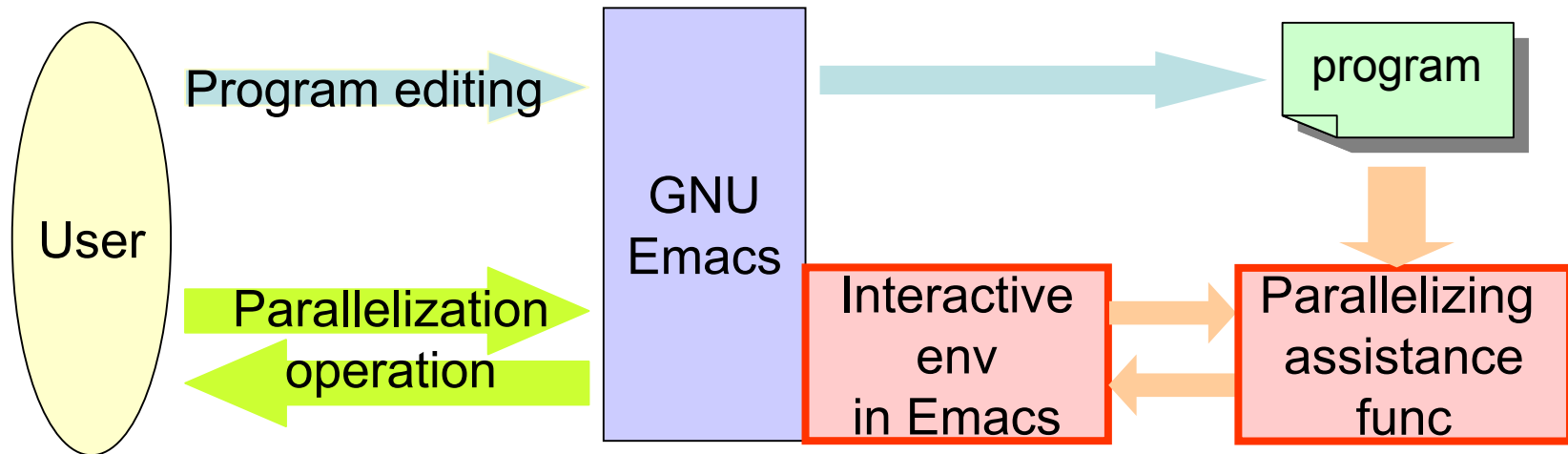
- Shell script manages:
 1. Translating a source program with the C/F-front
 2. Analyzing the Xobject code with the assistance libraries
 3. Showing the results onto the standard output



Assistance libraries

- Loop parallelism analysis function:
 - Analyzes data dependence relations in a for-loop and shows the results
 - Uses *PowerTest*
- OpenMP directive creation function:
 - Creates the OpenMP directives required for parallelization and shows them

Interactive environment of iPat/OMP in Emacs

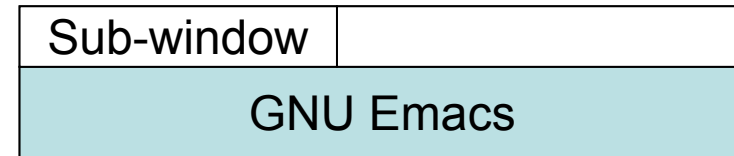


Interactive environment of iPat/OMP in Emacs

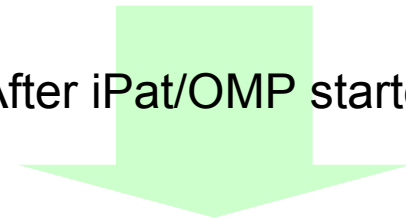
- iPat/OMP mode creation
 - Mode: the environment for the particular operations
 - A mode-name
 - Key-binds
 - Functions
 - Four groups of iPat/OMP Emacs Lisp functions
 - Boot function
 - Assistance command functions
 - Message management function
 - Target selection function

Boot function

- Embeds the assistance functionalities into Emacs
- Splits the Emacs window into two sections:
 1. Main-window:
 - used for editing the program
 2. Sub-window
 - used for the interactive processing of assistance functionalities

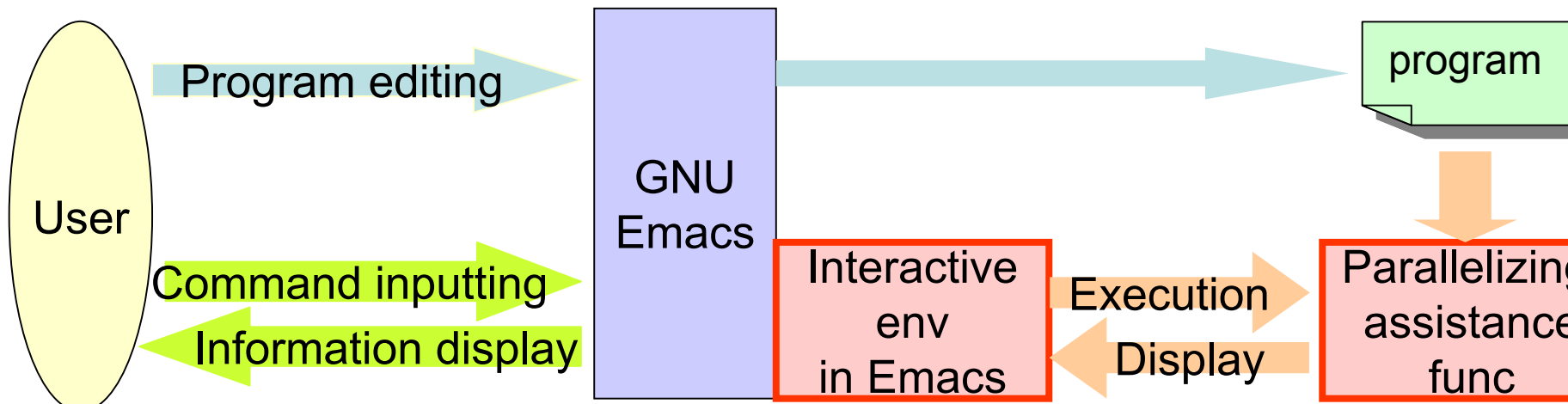


After iPat/OMP started



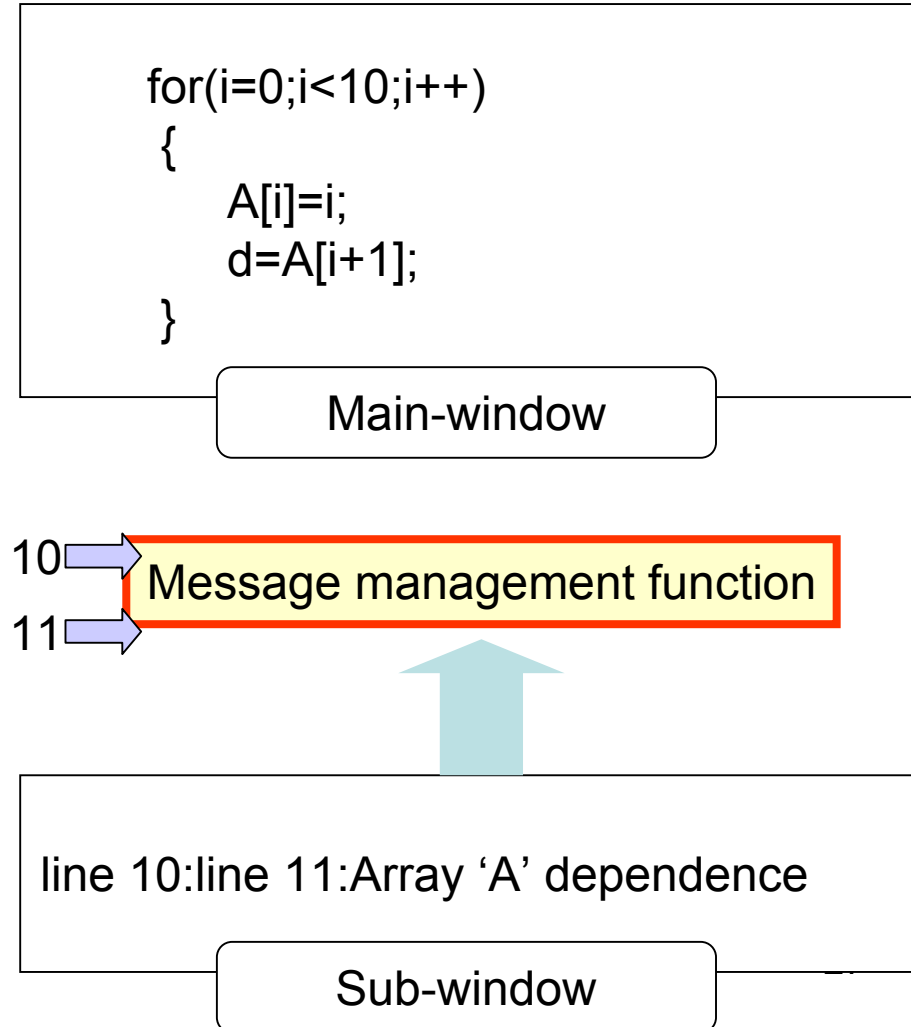
Assistance command functions

- Carry out assistance functionalities upon request in the following order:
 1. The user requests the parallelizing assistance.
 2. The assistance command function invokes a shell script, which manages the assistance libraries and the Omni OpenMP compiler.
 3. The assistance command function sends the information to the standard output of the shell script for display.



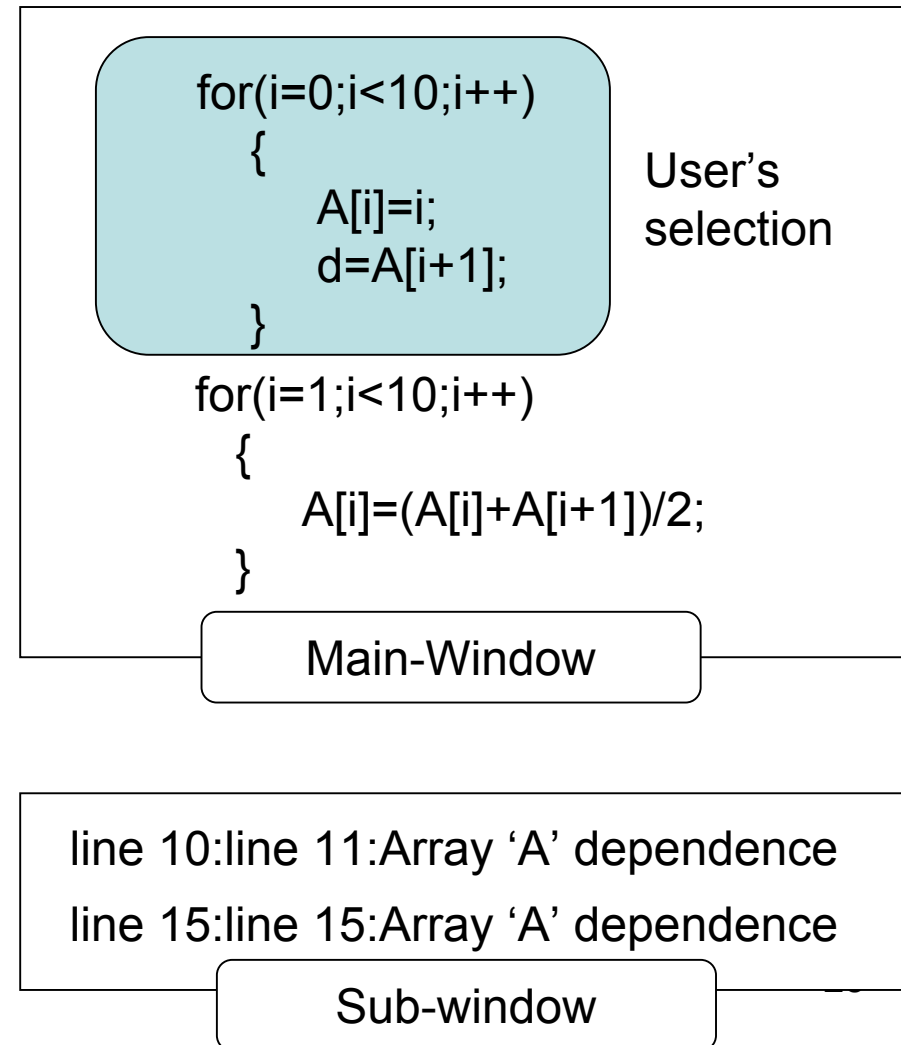
Message management function

- Relates messages in the sub-window to appropriate program sections in the main window by:
 1. Analyzing each message in the sub-window
 2. Visually pointing to the indicated section of the program in the main window



Target selection function

- Enables the user to parallelize a specific part of a program:
 1. The user selects a specific part of the program.
 2. iPat/OMP shows analysis results of the selected section.
 3. iPat/OMP does not allow the editing of other sections until the user finishes or cancels the current selection.



Demonstration

```
emacs@optiplex3
File Edit Options Buffers Tools C Help
[Icons: File, Folder, X, Disk, Printer, Undo, Cut, Copy, Paste, Find, Print, Checkmark, Question Mark]

double time1,time2;
double second();

double u[1000 +2][1000 +2],uu[1000 +2][1000 +2];

main()
{
    int x,y,k;
    double sum;
    for(x = 1; x <= 1000; x++)
    {
        for(y = 1; y <= 1000; y++)
        {
            u[x][y] = sin((double) (x-1)/1000*3.1415927) + cos((double) (y-1)/1000*3.1415927);
        }
    }

    for(x = 0; x < (1000 +2); x++)
    {
        u[x][0] = 0.0;
        u[x][1000 +1] = 0.0;
        uu[x][0] = 0.0;
    }
}

--:-- lapexam.c (C CVS-1.2 Abbrev)--L1--Top-----
```

```
emacs@optiplex3
File Edit Options Buffers Tools Minibuf Help
[Icons: File, Folder, Close, Save, Print, Undo, Cut, Copy, Paste, Find, Print, Checkmark, Help]

double time1,time2;
double second();

double u[1000 +2][1000 +2],uu[1000 +2][1000 +2];

main()
{
    int x,y,k;
    double sum;
    for(x = 1; x <= 1000; x++)
    {
        for(y = 1; y <= 1000; y++)
        {
            u[x][y] = sin((double) (x-1)/1000*3.1415927) + cos((double) (y-1)/1000*3.1415927);
        }
    }

    for(x = 0; x < (1000 +2); x++)
    {
        u[x][0] = 0.0;
        u[x][1000 +1] = 0.0;
        uu[x][0] = 0.0;
    }
}

--:-- lapexam.c (C CVS-1.2 Abbrev)--L1--Top-----
M-x iPat-OMP
```



```
double time1,time2;
double second();

double u[1000 +2][1000 +2],uu[1000 +2][1000 +2];

main()
{
    int x,y,k;
    double sum;

    for(x = 1; x <= 1000; x++)
    {
        for(y = 1; y <= 1000; y++)
        {
```

---:-- **lapexam.c** (C CVS-1.2 Abbrev)--L1--Top-----

---:-- ***messages*** (iPat/OMP mode)--L1--All-----



```
emacs@optiplex3
File Edit Options Buffers Tools C Help
[Icons: File, Folder, Close, Save, Undo, Cut, Copy, Paste, Find, Print, Refresh, Help]

for(k = 0; k < 100; k++)
{
  for(x = 1; x <= 1000; x++)
  {
    for(y = 1; y <= 1000; y++)
    {
      uu[x][y] = u[x][y];
    }
  }
  for(x = 1; x <= 1000; x++)
  {
    for(y = 1; y <= 1000; y++)
    {

```

--:-- **lapexam.c** (C CVS-1.2 Abbrev) --L39--63%

--:-- ***messages*** (iPat/OMP mode) --L1--All

```
emacs@optiplex3
File Edit Options Buffers Tools Help
[Icons: File, Folder, Close, Save, Print, Undo, Cut, Paste, Find, Print, Copy, Help]
[] for(x = 1; x <= 1000; x++)
  {
    for(y = 1; y <= 1000; y++)
      {
        uu[x][y] = u[x][y];
      }
  }

--:-- lapexam.c (C CVS-1.2 Abbrev Narrow) --L1--All-----
lapexam.c:39:#pragma omp parallel for shared(u,uu) lastprivate(x,y)
lapexam.c:41:#pragma omp parallel for shared(u,uu) firstprivate(x) lastprivate(y)
completed

--:%% *messages* (iPat/OMP mode) --L1--All-----
[Icon]
```

```
emacs@optiplex3
File Edit Options Buffers Tools C Help
[Icons: File, Folder, Close, Save, Print, Undo, Cut, Paste, Find, Print, Stop, Help]
39-> for(x = 1; x <= 1000; x++)
    {
        for(y = 1; y <= 1000; y++)
            {
                uu[x][y] = u[x][y];
            }
    }

---:-- lapexam.c (C CVS-1.2 Abbrev Narrow) --L1--All-----
lapexam.c:39:#pragma omp parallel for shared(u,uu) lastprivate(x,y)
lapexam.c:41:#pragma omp parallel for shared(u,uu) firstprivate(x) lastprivate(y)
completed

---:%% *messages* (iPat/OMP mode) --L1--All-----
[Icons: Refresh] insert OK?(y or n)
```



```

    for(x = 1; x <= 1000; x++)
    {
        for(y = 1; y <= 1000; y++)
43->    {
            uu[x][y] = u[x][y];
        }
    }
    for(x = 1; x <= 1000; x++)
    {
        for(y = 1; y <= 1000; y++)
50->    {
            u[x][y] = (uu[x-1][y] + uu[x+1][y] + uu[x][y-1] + uu[x][y+1])/4.0;
        }
    }

```

```
--:** lapexam.c (C CVS-1.2 Abbrev Narrow)--L7--13%
```

```

lapexam.c:43:50: array 'u' : flow or anti dependence.
lapexam.c:43:50: array 'uu' : flow or anti dependence.
lapexam.c:43:50: array 'uu' : flow or anti dependence.
lapexam.c:43:50: array 'uu' : flow or anti dependence.
lapexam.c:43:50: array 'uu' : flow or anti dependence.
lapexam.c:39:#pragma omp parallel for shared(u,uu) lastprivate(x,y)
lapexam.c:41:#pragma omp parallel for shared(u,uu) firstprivate(x) lastprivate(y)
lapexam.c:46:#pragma omp parallel for shared(uu,u) lastprivate(x,y)

```

```
--:%% *messages* (iPat/OMP mode)--L1--Top
```

Summary

- We proposed a parallelizing assistance tool, called iPat/OMP.
 - Its design concepts enhance the maintenance of human-readable source code.
 - It does not rely on any particular OS, consisting of a combination of an Emacs mode, Java functions, and shell scripts, making it highly portable.
 - It allows the addition of new assistance capabilities using Emacs Lisp, making it extendable.

Future work

- Improvements in assistance library capabilities:
 - directive creation and optimization
 - program restructuring
 - execution time analysis
- Improvement in the iPat/OMP environment in Emacs to make it more useful and user-friendly
- Addition of the capability to customize the style of assistance according to individual skill in parallel programming

Future work

- Evaluation of the usability of iPat/OMP
 - Using bench-mark programs
 - Comparing iPat/OMP with:
 - Other parallelizing assistance tools
 - Automatic parallelizing compilers
- Collection of advice through the release of a beta version of iPat/OMP with Omni OpenMP distribution
- Application of this advice in the release of a formal version of the program

References

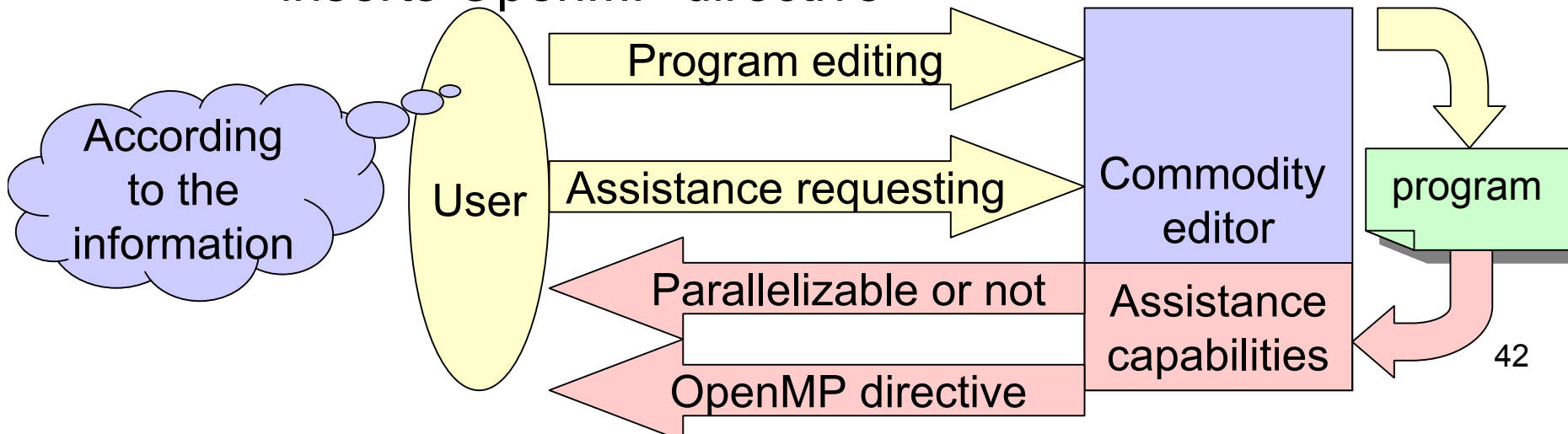
- [1] <http://www.openmp.org/>
- [2] M. Sato, S. Satoh, K. Kusano and Y. Tanaka, Design of OpenMP Compiler for an SMP Cluster, EWOMP '99, pp.32-39, October 1999.
- [3] U. Banerjee, Loop Transformations for Restructuring Compilers The foundations, Kluwer Academic Publishers, January 1993.
- [4] M. Wolfe and C.-W Tseng, The Power Test for Data Dependence, IEEE Transactions on Parallel and Distributed Systems pp.591-601, September 1992.
- [5] NASA, CAPO(Computer-Aided Parallelizer and Optimizer). <http://www.nas.nasa.gov/Tools/CAPO/>

References

- [6] K. Kennedy, K. S. McKinley, and C.-W. Tseng, Analysis and Transformation in the ParaScope Editor. In Proc. ACM Int'l. Conf. Supercomputer. pp.433-447, June 1991.
- [7] Parallel Software Products Inc, ParaWise
<http://www.parallels.com/>
- [8] I. Park, M. J. Voss, S. W. Kim and R. Eigenmann, Parallel Programming Environment for OpenMP, Scientific Programming, Vol. 9, No. 2/3, pp.143-161, 2001.
- [9] R. Eigenmann and P. McClaughry, Practical Tools for Optimizing Parallel Programs. 1993 SCS Multiconference, March 1993.

For beginners

- iPat/OMP
 - indicates whether or not a program section is parallelizable
 - shows OpenMP directives if the program is parallelizable
- The user
 - follows the information from iPat/OMP
 - inserts OpenMP directive



For experts

- iPat/OMP shows more detailed information.
 - If the program is parallelizable : OpenMP directives
 - If the program is not parallelizable : information about the obstacles to parallelization
- The user parallelizes the program based on this information.

