Introduction to Eclipse III: Debugging MPI and OpenMP code

December 1, 2021

Ge Baolai, *Western University*
SHARCNET | Compute Ontario | Compute Canada

Outline

- Eclipse
- Scope of the talk
- Developing C/C++ and Fortran code
- Debugging serial code
- Debugging parallel MPI code
- Debugging OpenMP code
- Q&A

Compute · Calcul Ontario

SHARCNET™

General Interest Seminars 2021-2022

Western

About Eclipse

About Eclipse

What is Eclipse. An IDE for development and debugging

- C/C++
- Fortran
- Other languages
- MPI, OpenMP, threaded
- Others?

It runs on your local computer (and remote systems), available for Windows, Linux and MacOS.

About Eclipse

What Eclipse is not. It is a front-end, with an editor and code analyzer for C/C++, Fortran, MPI and others, to the underlying compilers and debuggers. It DOES NOT come with

- Compilers
- Debuggers
- Memory analyzers
- Schedulers

About Eclipse

Why Eclipse. Debugging parallel code is not straightforward. Options for debugging parallel, MPI code

- DDT (cost)
- TotalView (cost)
- gdb (free) – command line based, hard to use.
- Eclipse (free) – GUI, reasonably intuitive to use.
- Others?

About Eclipse

ECLIPSE FOUNDATION

Projects Working Groups Members More

Home / Downloads / Packages / Release / Eclipse IDE 2021-06 / R

Eclipse Installer | Eclipse Packages | Eclipse Developer Builds

The Eclipse Installer 2021-06 R now includes a JRE for macOS, Windows and Linux.

Try the Eclipse **Installer** 2021-06 R

The easiest way to install and update your Eclipse Development Environment.

Find out more

2,045,554 Installer Downloads

2,341,599 Package Downloads and Updates

Download

macOS x86_64
Windows x86_64
Linux x86_64 | AArch64

Eclipse IDE 2021-06 R Packages

Eclipse IDE for Java Developers

320 MB | 1,188,534 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

Eclipse IDE for Enterprise Java and Web Developers

517 MB | 850,133 DOWNLOADS

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.

Click here to file a bug against Eclipse Web Tools Platform.
Click here to file a bug against Eclipse Platform.

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

The Eclipse Installer 2021-06 R now includes a JRE for macOS, Windows and Linux.

Get **Eclipse IDE 2021-06**

Install your favorite desktop IDE packages.

Download x86_64

Download Packages | Need Help?

RELATED LINKS

- Compare & Combine Packages
- New and Noteworthy
- Install Guide

About Eclipse

Eclipse IDE for RCP and RAP Developers

322 MB | 10,045 DOWNLOADS

A complete set of tools for developers who want to create Eclipse plug-ins, Rich Client Applications or Remote Application Platform (RCP+RAP), plus Maven and Gradle tooling, and an XML editor. It contains the EGit tooling for accessing Git version control systems, and Eclipse Passage which helps with license management for Eclipse-based products.

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

Eclipse Modeling Tools

442 MB | 9,361 DOWNLOADS

The Modeling package provides tools and runtimes for building model-based applications. You can use it to graphically design domain models, to leverage those models at design time by creating and editing dynamic instances, to collaborate via Eclipse's team support with facilities for comparing and merging models and model instances structurally, and finally to generate Java code from those models to produce complete applications. In addition, via the package's discover catalog, you can easily install a wide range of additional powerful, model-based tools and runtimes to suit your specific needs.

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

Eclipse IDE for Scientific Computing

320 MB | 5,728 DOWNLOADS

Tools for C, C++, Fortran, and UPC, including MPI, OpenMP, OpenACC, a parallel debugger, and remotely building, running and monitoring applications.

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

Eclipse IDE for Scout Developers

293 MB | 3,655 DOWNLOADS

Eclipse Scout is a Java/HTML5 framework to develop business applications that run on the desktop, on tablets and mobile devices. This package includes Eclipse IDE support for Scout developers and source code.

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

About Eclipse

ECLIPSE FOUNDATION

Projects | Working Groups | Members | More

Home / Downloads / Packages / Release / Eclipse IDE 2021-06 / R / Eclipse IDE for Scientific Computing

Eclipse Installer | Eclipse Packages | Eclipse Developer Builds

Eclipse IDE for Scientific Computing

Package Description

Tools for C, C++, Fortran, and UPC, including MPI, OpenMP, OpenACC, a parallel debugger, and remotely building, running and monitoring applications.

This package includes:

- C/C++ Development Tools
- Git integration for Eclipse
- Parallel Tools Platform
- Eclipse XML Editors and Tools

Detailed features list

Maintained by: Eclipse Packaging Project

Download Links

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

Downloaded 5,728 Times

Checksums...

Bugzilla

Open Bugs: 7

Resolved Bugs: 14

File a Bug on this Package

New and Noteworthy

Eclipse PTP
Eclipse Platform

Testing Details

Package Testers

Greg Watson

ORACLE
Enterprise Pack for Eclipse
OEPE
Download

The Eclipse Installer 2021-06 R now includes a JRE for macOS, Windows and Linux.

Get **Eclipse IDE 2021-06**

Install your favorite desktop IDE packages.

Download x86_64

Download Packages | Need Help?

RELATED LINKS

- Compare & Combine Packages
- New and Noteworthy
- Install Guide

Requirements

- Windows
  - Compilers, debugger, e.g. from cigywin distribution.
  - But **WSL** - a true Linux environment running on top of Windows by Microsoft - is recommended. See a tutorial on enabling WSL on Windows at https://youtube.sharcnet.ca/.
- Linux, Mac OS
  - GCC compilers, gdb.
  - make, cmake.
  - OpenMPI

General advice

- It's always a good practice to install an MPI implementation on your own machine (laptop or desktop), so you have immediate access and can practice and develop code.
- For Windows, we recommend
  - Enable **Windows Subsystem for Linux** (WSL) and
  - Install a Linux distro, e.g. Ubuntu, Debian, etc. on it.
- Install latest GNU compilers and gdb.
- Install **OpenMPI** or **MPICH**.

Scope of The Tutorial

What are covered and what aren't

What to be introduced in this tutorial

- Eclipse on Linux.
- C/C++ and Fortran code.
- MPI code.
- OpenMP code.
- Debugging on local computer.

What NOT to be covered

- OpenACC code.
- Non GCC compilers, debuggers.
- Running and debugging code on remote systems.
- Use of git.

Debugging MPI Code

Stepping through MPI ranks. A simple MPI code to examine variables.

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Comm_size(MPI_COMM_WORLD, &num_tasks);

if (my_rank !=0) // Non root ranks
{
   sprintf(message, "Hello MPI World from process %d!", my_rank);
   message[strlen(message)] = '\0';
   MPI_Send(message,strlen(message)+1,MPI_CHAR,0,tag,MPI_COMM_WORLD);
}
else // Root
{
   printf("Hello MPI World from process 0: Num of processes: %d\n", num_tasks);
   for (p=1; p<num_tasks; p++) {
      MPI_Recv(message,100,MPI_CHAR,MPI_ANY_SOURCE,tag,MPI_COMM_WORLD, &status);
      printf("From process %d: %s\n", status.MPI_SOURCE, message);
   }
}
```

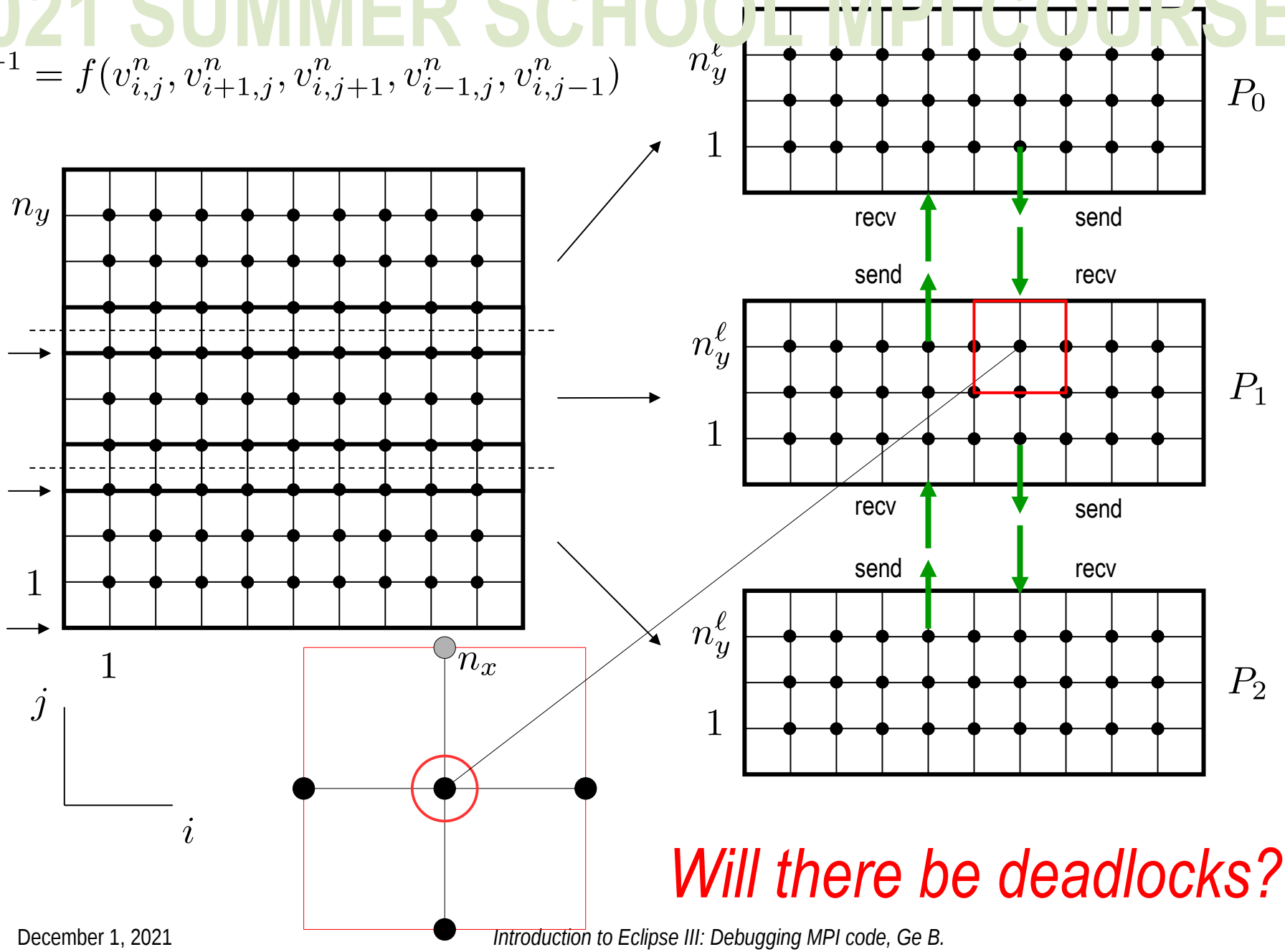
MPI code with segmentation fault.

```
#define N_src 10000
#define N_dest 100
#define TAG 99

int main(int argc, char* argv[])
{
   int num_tasks, rank, root, dest;
   int sbuf[N_src], rbuf[N_dest];
   MPI_Status status;

   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &num_tasks);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   root = 0;
   dest = 1;
   if (0 == rank)
      MPI_Send(sbuf, N_src, MPI_INT, dest, TAG, MPI_COMM_WORLD);
   else
      MPI_Recv(rbuf, N_dest, MPI_INT, root, TAG, MPI_COMM_WORLD, &status);

   printf("Done.\n");
   MPI_Finalize();
   return 0;
}
```

Original Author: Sergey Maschenko

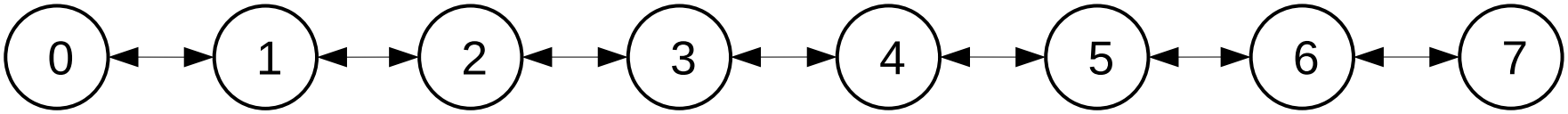
Parallel Algorithm

$$v_{i,j}^{n+1} = f(v_{i,j}^{n}, v_{i+1,j}^{n}, v_{i,j+1}^{n}, v_{i-1,j}^{n}, v_{i,j-1}^{n})$$

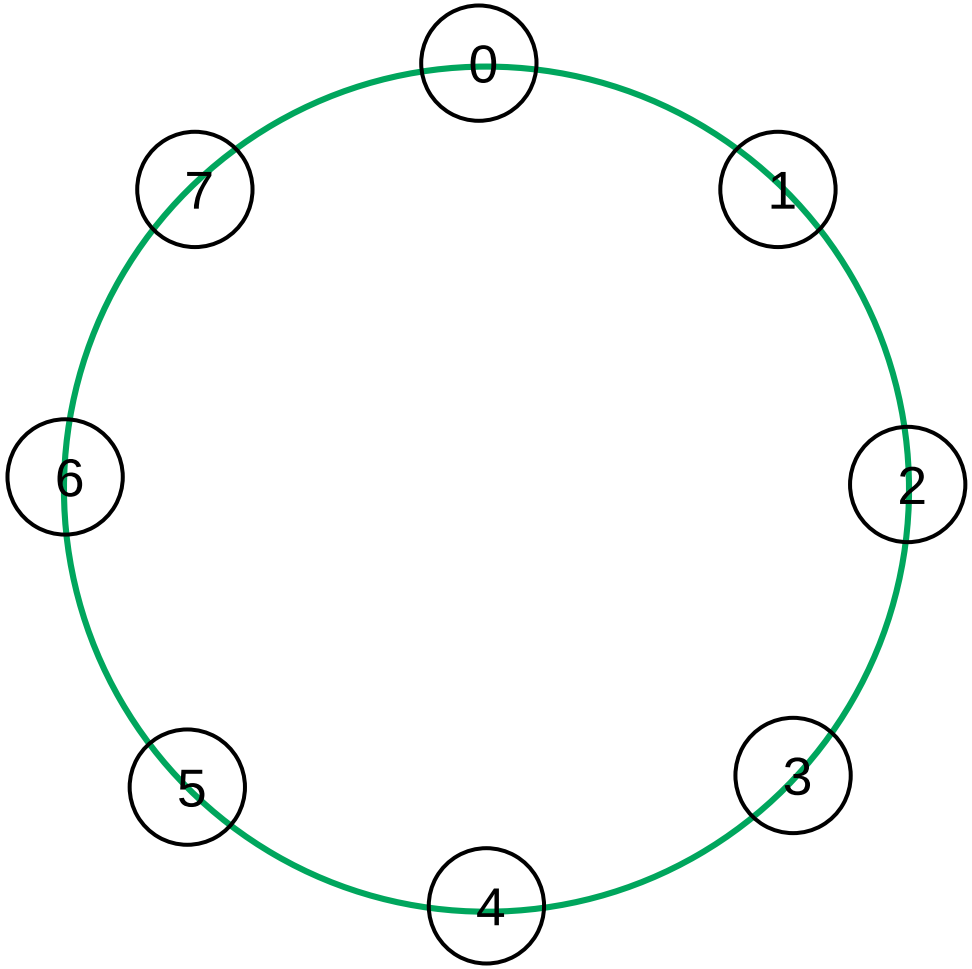


Will there be deadlocks?

Domain decomposition with boundaries



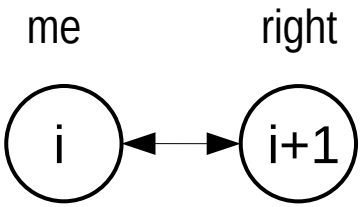
Domain decomposition with periodic conditions



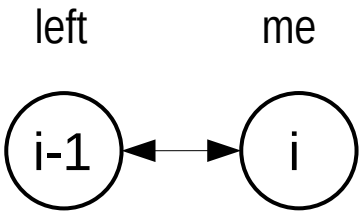
Blocking vs Non-blocking: Deadlock

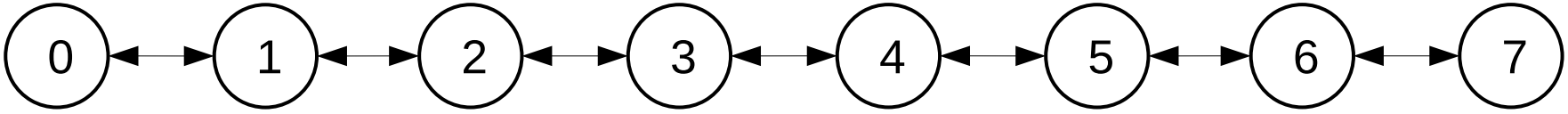
Example: Think about writing a code that involve N processes in order in a chain. Each one sends and receives a message to/from both its left and right neighbours. The key code segment is sketched below. Will there be a deadlock?


```
left = rank - 1;
if (left < 0) left = MPI_PROC_NULL;
right = rank + 1;
if (right == num_procs) right = MPI_PROC_NULL;

MPI_Send(rsmsg,n,MPI_DOUBLE,right,rtag,MPI_COMM_WORLD);
MPI_Recv(rrmsg,n,MPI_DOUBLE,right,rtag,MPI_COMM_WORLD,&rstat);
```




```
MPI_Send(lsmsg,n,MPI_DOUBLE,left,ltag,MPI_COMM_WORLD);
MPI_Recv(lrmsg,n,MPI_DOUBLE,left,ltag,MPI_COMM_WORLD,&rstat);
```





Debugging MPI code

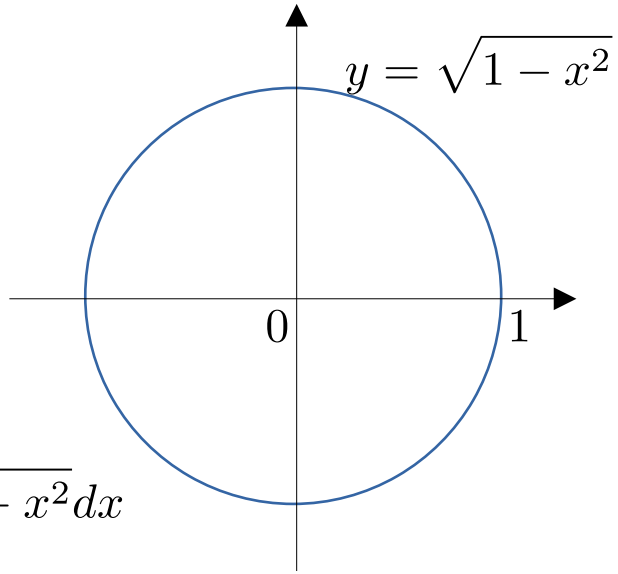
MPI code causing deadlock. We are looking at a simplest scenario: sending and receiving messages to/from the same peer.

```
// This example requires two ranks involved
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &num_tasks);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

peer = 1 – rank;

// The following is unsafe, there is a good chance  that it blocks
MPI_Send(sbuf,n,MPI_FLOAT,peer,TAG,MPI_COMM_WORLD);
MPI_Recv(rbuf,n,MPI_FLOAT,peer,TAG,MPI_COMM_WORLD,&stat);
printf("Rank %d send, receive calls completed.\n", rank);
```

Debugging OpenMP Code

Computing pi – race condition

```
for (int i=0; i<n; i++)
{
    x = (i + 0.5)/n; // Midpoint rule
    s += sqrt(1 = x*x);
}

pi_est = 4.0/n*s;
```



$$\pi = 4 \int_0^1 \sqrt{1 - x^2} dx$$

The OpenMP code in omp-pi.c is simple but buggy. We first consider a rather simple problem: add natural numbers from 1 to N in parallel.

Parallel regions – deadlock

```
flag=0;
#pragma omp parallel shared(flag)
{
  int id = omp_get_thread_num();
  #pragma omp sections
  {
    #pragma omp section
    {
      while (flag != 0) // If 0 then break, otherwise spin forever
      {
      }
      printf("Section 1\n");
      flag = 1;
    }
    #pragma omp section
    {
      while (flag != 1) // If 0, 2 or 3, then break
      {
      }
      printf("Section 2\n");
      flag = 2;
    }
    #pragma omp section
    {
      while (flag != 2) // If 0, 1 or 3, then break
      {
      }
      printf("Section 3\n");
      flag = 3;
    }
  }
}
```

Original Author: Sergey Maschenko

Q&A