parallel tools platform

oak ridge national laboratory

Greg Watson
IBM Research

Outline

- Eclipse Overview
- Parallel Tools Platform
- Architecture Overview

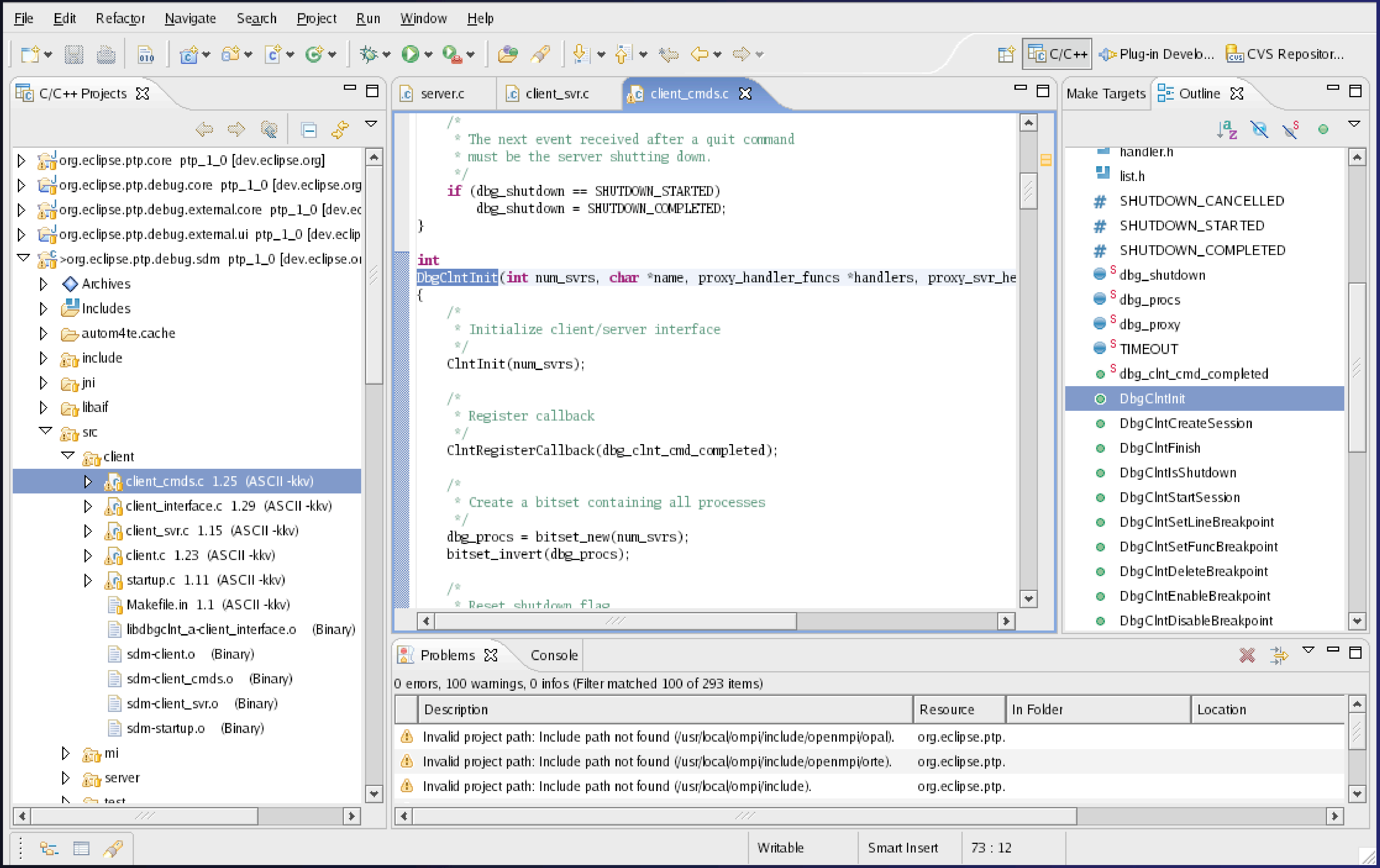
© 2007 by Greg Watson; made available under the EPL 1.0

What is Eclipse?

- Cross-platform open source framework for highly integrated state-of-the-art tools
- Existing tools include project management, advanced editing, automatic build system, revision control, visual debugger, and others...
- Designed to be robust, scalable, commercial quality
- Available for Linux, Unix and Windows
- Multi-language support for Java, C, C++, Fortran, Python, Perl, PHP, and others

© 2007 by Greg Watson; made available under the EPL 1.0

Workbench Features



```
org.eclipse.ptp.core  ptp_1_0 [dev.eclipse.org]
org.eclipse.ptp.debug.core  ptp_1_0 [dev.eclipse.org
org.eclipse.ptp.debug.external.core  ptp_1_0 [dev.ec
org.eclipse.ptp.debug.external.ui  ptp_1_0 [dev.eclip
>org.eclipse.ptp.debug.sdm  ptp_1_0 [dev.eclipse.o
    Archives
    Includes
    autom4te.cache
    include
    jni
    libaif
    src
        client
            client_cmds.c  1.25  (ASCII -kkv)
            client_interface.c  1.29  (ASCII -kkv)
            client_svr.c  1.15  (ASCII -kkv)
            client.c  1.23  (ASCII -kkv)
            startup.c  1.11  (ASCII -kkv)
            Makefile.in  1.1  (ASCII -kkv)
            libdbgclnt_a-client_interface.o   (Binary)
            sdm-client.o   (Binary)
            sdm-client_cmds.o   (Binary)
            sdm-client_svr.o   (Binary)
            sdm-startup.o   (Binary)
        mi
        server
        test
```

```
	/*
	 * The next event received after a quit command
	 * must be the server shutting down.
	 */
	if (dbg_shutdown == SHUTDOWN_STARTED)
		dbg_shutdown = SHUTDOWN_COMPLETED;
}

int
DbgClntInit(int num_svrs, char *name, proxy_handler_funcs *handlers, proxy_svr_he
{
	/*
	 * Initialize client/server interface
	 */
	ClntInit(num_svrs);

	/*
	 * Register callback
	 */
	ClntRegisterCallback(dbg_clnt_cmd_completed);

	/*
	 * Create a bitset containing all processes
	 */
	dbg_procs = bitset_new(num_svrs);
	bitset_invert(dbg_procs);

	/*
	 * Reset shutdown flag
```

```
handler.h
list.h
SHUTDOWN_CANCELLED
SHUTDOWN_STARTED
SHUTDOWN_COMPLETED
dbg_shutdown
dbg_procs
dbg_proxy
TIMEOUT
dbg_clnt_cmd_completed
DbgClntInit
DbgClntCreateSession
DbgClntFinish
DbgClntIsShutdown
DbgClntStartSession
DbgClntSetLineBreakpoint
DbgClntSetFuncBreakpoint
DbgClntDeleteBreakpoint
DbgClntEnableBreakpoint
DbgClntDisableBreakpoint
```

0 errors, 100 warnings, 0 infos (Filter matched 100 of 293 items)

Description	Resource	In Folder	Location
Invalid project path: Include path not found (/usr/local/ompi/include/openmpi/opal).	org.eclipse.ptp.		
Invalid project path: Include path not found (/usr/local/ompi/include/openmpi/orte).	org.eclipse.ptp.		
Invalid project path: Include path not found (/usr/local/ompi/include).	org.eclipse.ptp.		

C/C++/Fortran Features

- C/C++ Development Tools (CDT) adds C and C++ support
- Photran adds Fortran support
- Standard (Makefile) and managed builders
- Outline view
- Advanced searching (types, functions, variables, declaration, reference, etc.)
- Content assist, context sensitive help
- Simple refactoring

© 2007 by Greg Watson; made available under the EPL 1.0

Parallel Tools Platform

- Brings the benefits of an integrated tool platform to parallel programmers
- Tools are designed to specifically address parallel programming problems
- Able to hide much of the parallel system complexity
- Platform for developing new/advanced tools and languages to address petascale issues

© 2007 by Greg Watson; made available under the EPL 1.0

Current status

- Project created in April 2005
- PTP 1.0 released in March 2006
- PTP transitions to Tools Project December 2006
- PTP 1.1 released in February 2007
- PTP 2.0 scheduled for Fall 2007

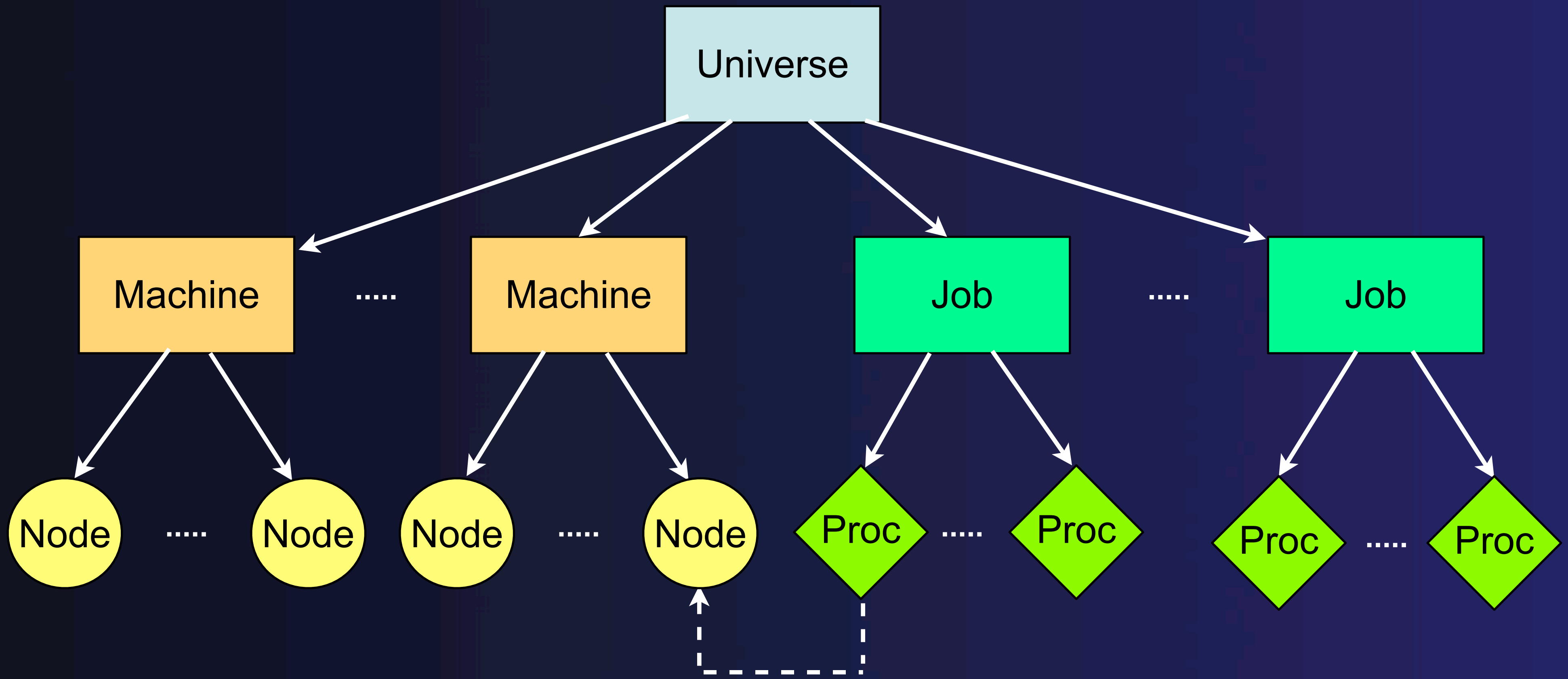
© 2007 by Greg Watson; made available under the EPL 1.0

What's in PTP 1.1

- Attributed parallel system model
  - An abstract representation of a parallel system
  - MVC pattern
- New perspectives
  - Runtime and debugger
- Launch configuration for parallel jobs
- Parallel debugger
- Parallel programming tools
- Fortran development tools

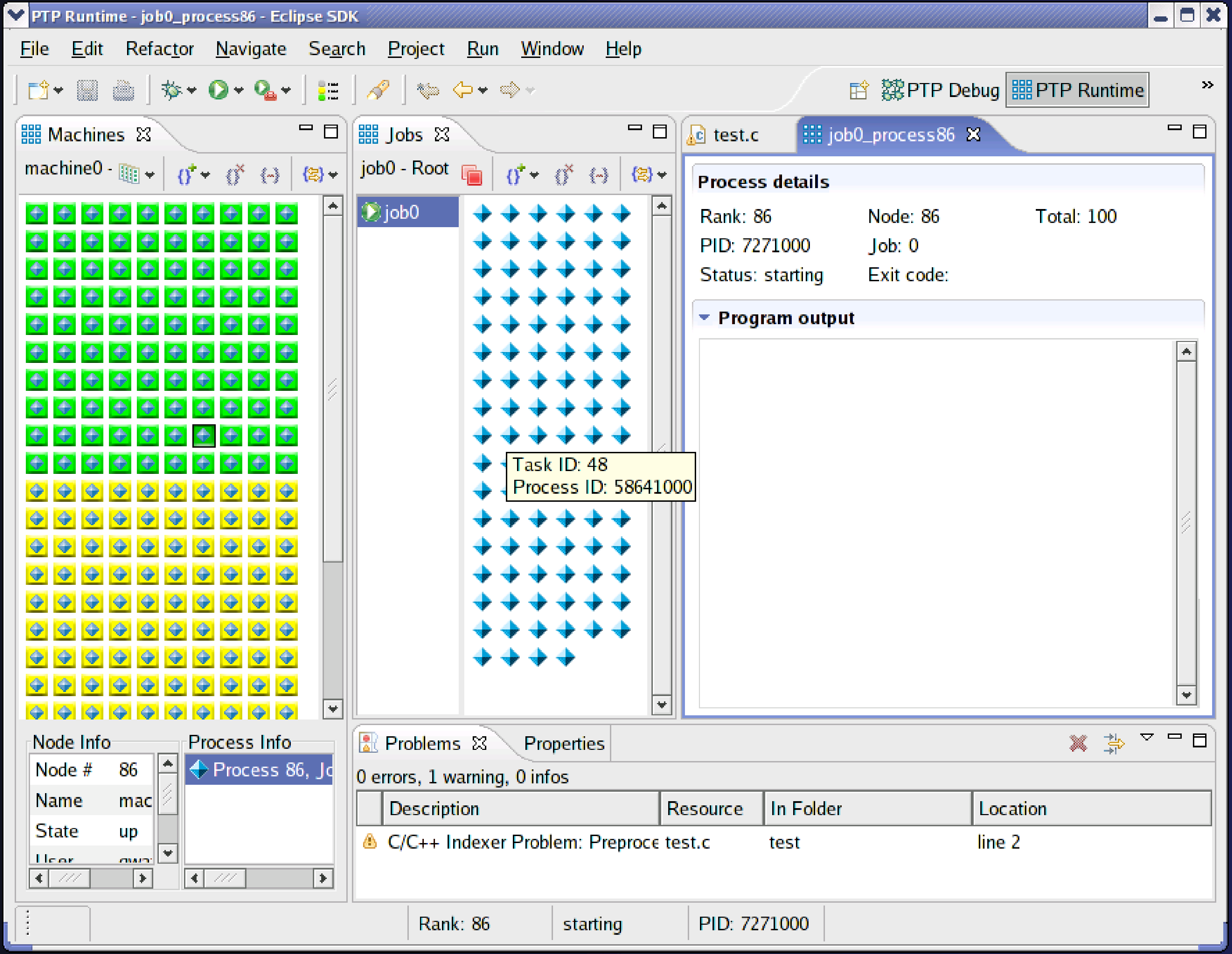
© 2007 by Greg Watson; made available under the EPL 1.0

Attributed parallel system model (1.1)



© 2007 by Greg Watson; made available under the EPL 1.0

PTP runtime perspective



Task ID: 48
Process ID: 58641000

Process details

Rank: 86 Node: 86 Total: 100
PID: 7271000 Job: 0
Status: starting Exit code:

Program output

Node #	86
Name	mac
State	up

0 errors, 1 warning, 0 infos

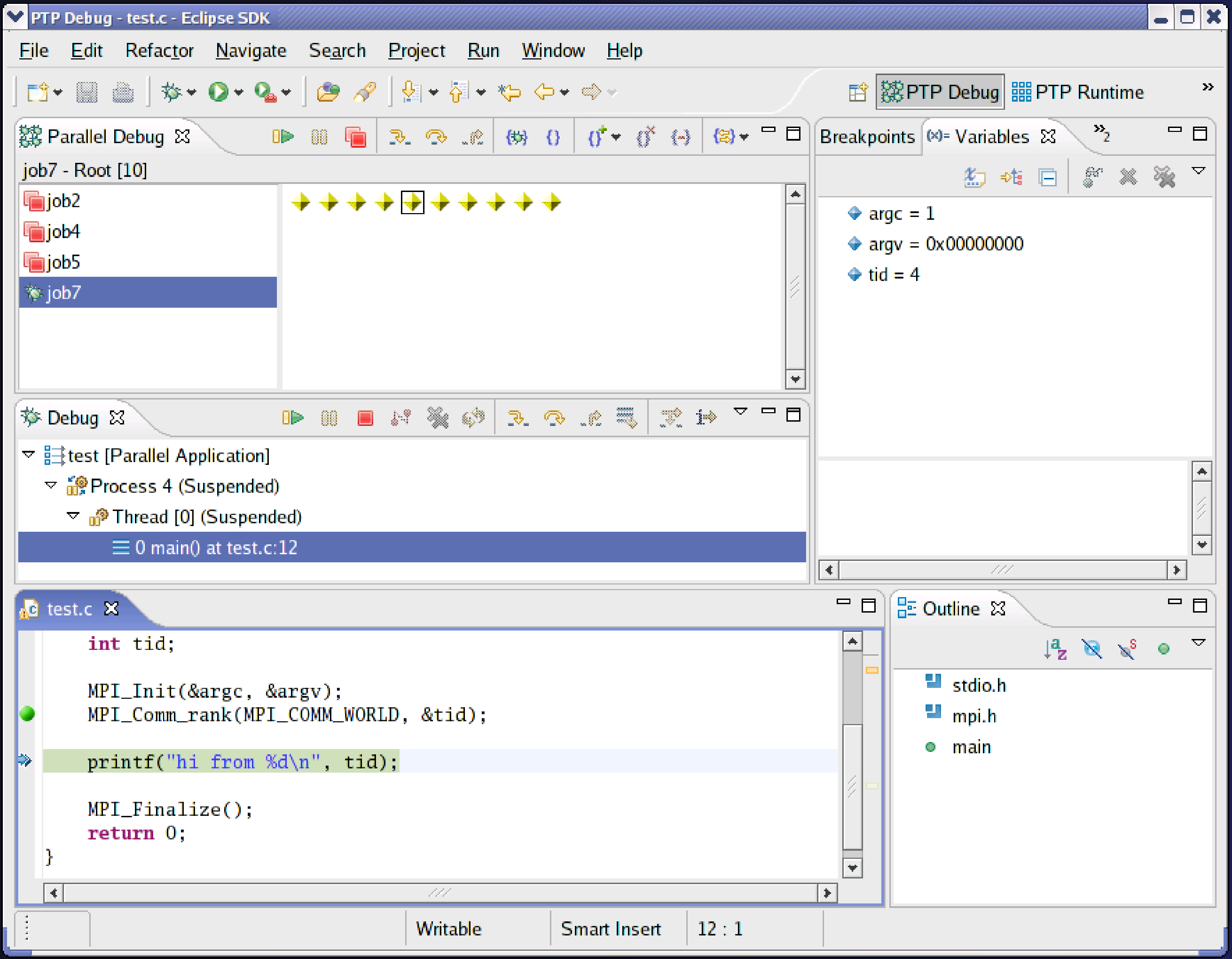
Description	Resource	In Folder	Location
C/C++ Indexer Problem: Preproce	test.c	test	line 2

Rank: 86 starting PID: 7271000

- Machines view
  - node status, node details, and processes running on the node
- Jobs view
  - jobs launched, processes in a job, and process status
- Process details view
  - more detailed process information and standard output from individual processes

© 2007 by Greg Watson; made available under the EPL 1.0

Parallel debugger



```
argc = 1
argv = 0x00000000
tid = 4
```

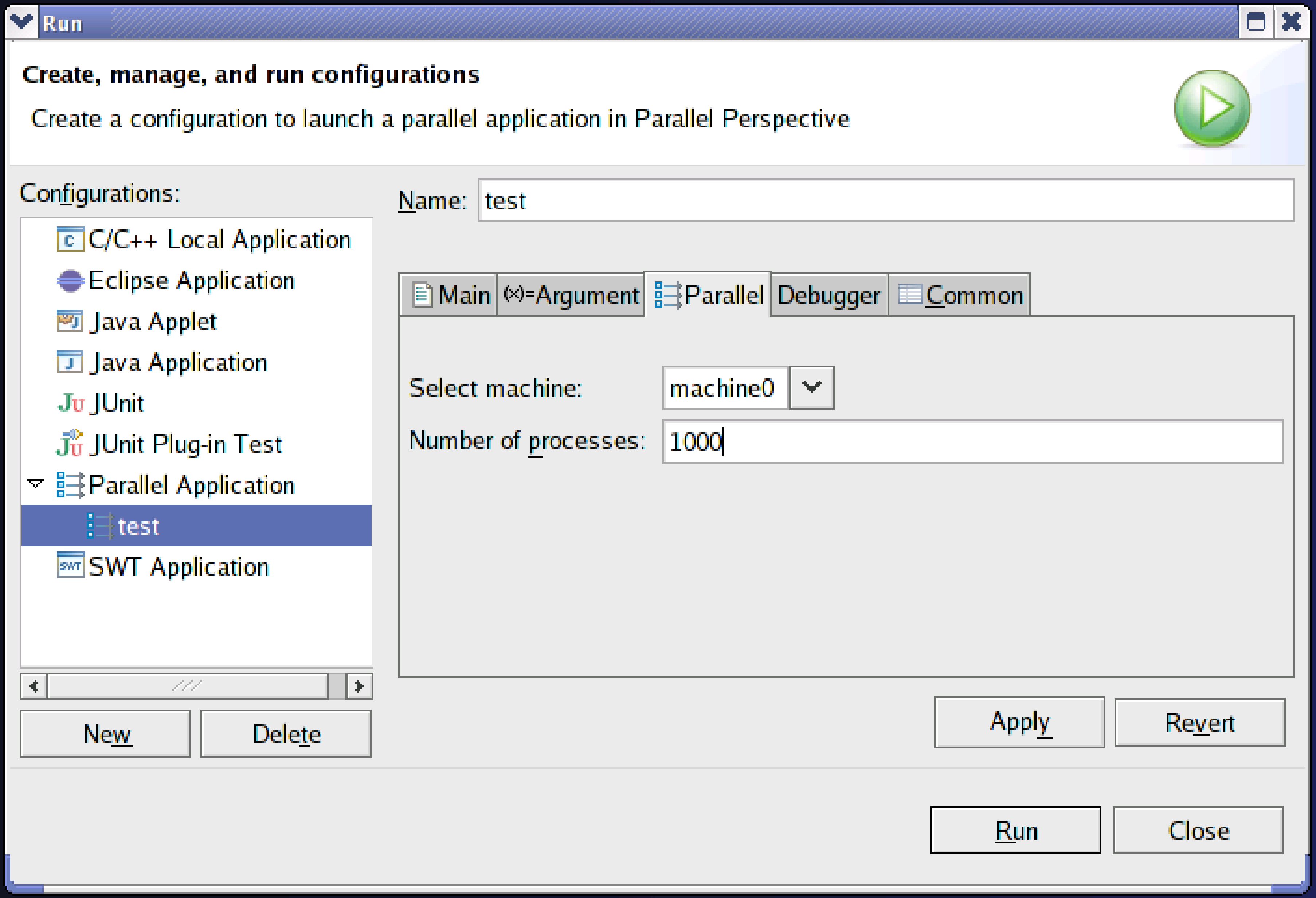
```
test [Parallel Application]
  Process 4 (Suspended)
    Thread [0] (Suspended)
      0 main() at test.c:12
```

```
    int tid;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &tid);

    printf("hi from %d\n", tid);

    MPI_Finalize();
    return 0;
}
```

- Parallel debug view
  - debug jobs launched, processes in a job, and process status
  - process sets
  - registered processes
  - tooltip display
- Extended breakpoints and location markers
  - breakpoint color shows which set associated with the breakpoint
  - multiple simultaneous location markers

© 2007 by Greg Watson; made available under the EPL 1.0

Parallel launch



Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

Name: test

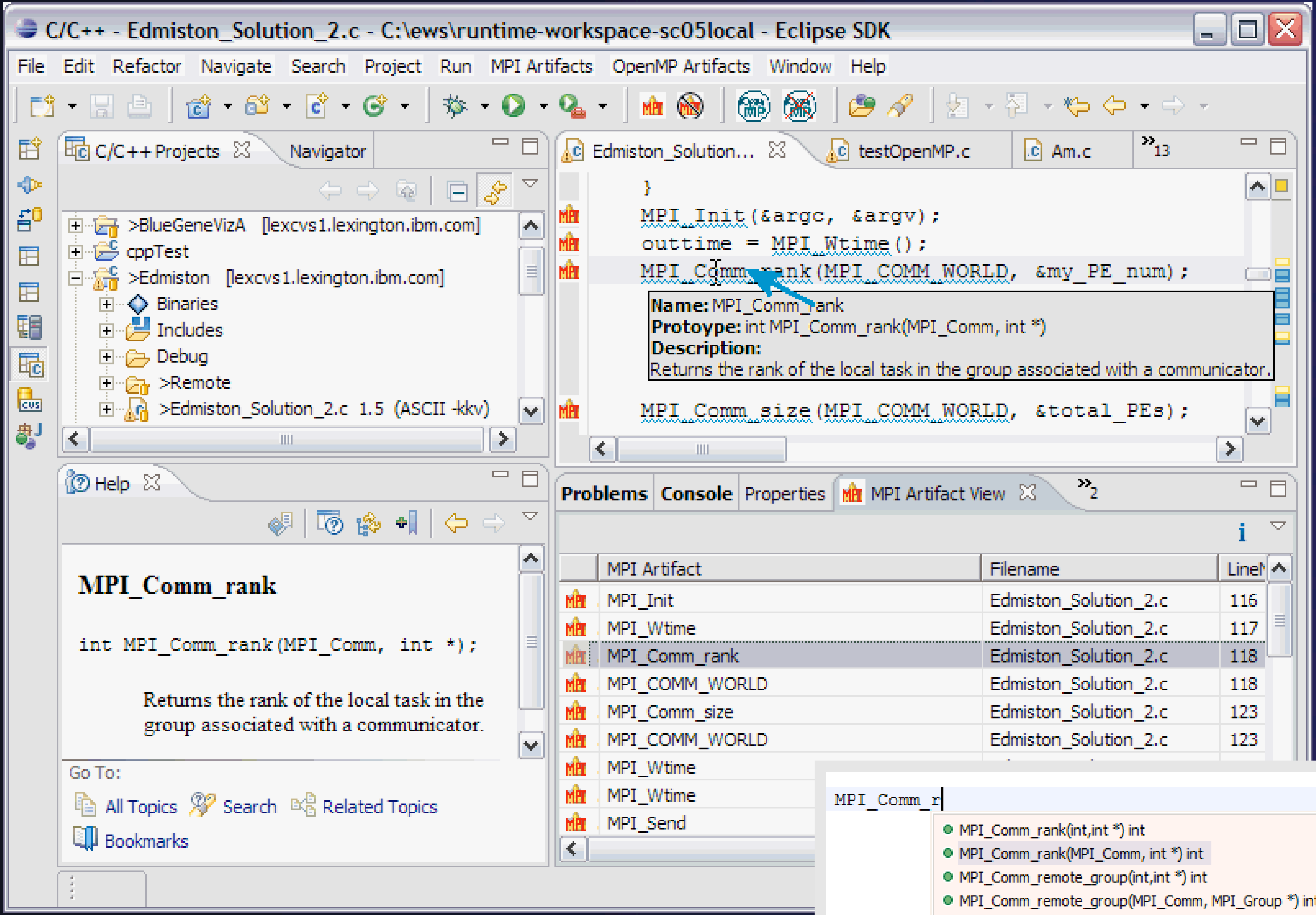
Select machine: machine0

Number of processes: 1000

- User can specify the machine and number of processes to launch
- Configure parallel debug launch

© 2007 by Greg Watson; made available under the EPL 1.0

Parallel programming tools


C/C++ - Edmiston_Solution_2.c - C:\ews\runtime-workspace-sc05local - Eclipse SDK
File Edit Refactor Navigate Search Project Run MPI Artifacts OpenMP Artifacts Window Help
C/C++ Projects
Navigator
Edmiston_Solution...
testOpenMP.c
Am.c
Help
Problems
Console
Properties
MPI Artifact View

```
>BlueGeneVizA  [lexcvs1.lexington.ibm.com]
cppTest
>Edmiston  [lexcvs1.lexington.ibm.com]
    Binaries
    Includes
    Debug
    >Remote
    >Edmiston_Solution_2.c  1.5  (ASCII -kkv)
```

```
    }
    MPI_Init(&argc, &argv);
    outtime = MPI_Wtime();
    MPI_Comm_rank(MPI_COMM_WORLD, &my_PE_num);

    MPI_Comm_size(MPI_COMM_WORLD, &total_PEs);
```

Name: MPI_Comm_rank
Protoype: int MPI_Comm_rank(MPI_Comm, int *)
Description:
Returns the rank of the local task in the group associated with a communicator.

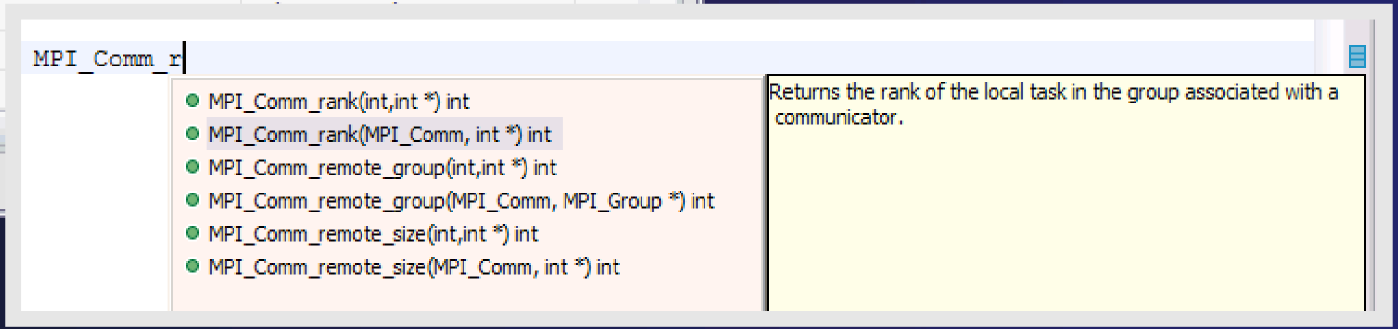
MPI_Comm_rank

```
int MPI_Comm_rank(MPI_Comm, int *);
```

Returns the rank of the local task in the group associated with a communicator.

Go To:
All Topics Search Related Topics
Bookmarks

MPI Artifact	Filename	Line
MPI_Init	Edmiston_Solution_2.c	116
MPI_Wtime	Edmiston_Solution_2.c	117
MPI_Comm_rank	Edmiston_Solution_2.c	118
MPI_COMM_WORLD	Edmiston_Solution_2.c	118
MPI_Comm_size	Edmiston_Solution_2.c	123
MPI_COMM_WORLD	Edmiston_Solution_2.c	123
MPI_Wtime		
MPI_Wtime		
MPI_Send		



```
MPI_Comm_r
```

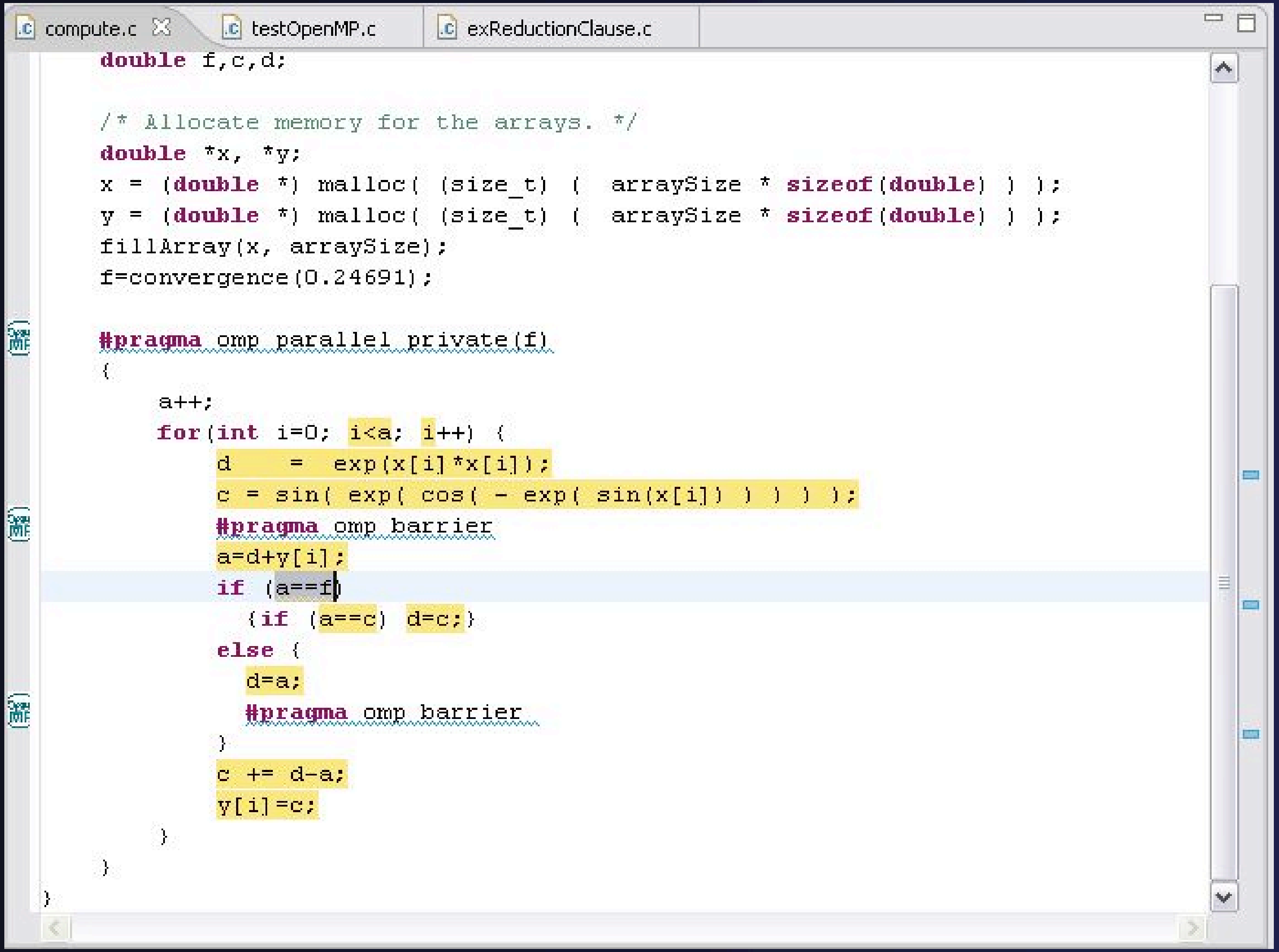
- MPI_Comm_rank(int,int *) int
- MPI_Comm_rank(MPI_Comm, int *) int
- MPI_Comm_remote_group(int,int *) int
- MPI_Comm_remote_group(MPI_Comm, MPI_Group *) int
- MPI_Comm_remote_size(int,int *) int
- MPI_Comm_remote_size(MPI_Comm, int *) int

Returns the rank of the local task in the group associated with a communicator.

- Identifies MPI artifacts
- Navigates to source code locations
- Help:
  - hover
  - content assist
  - F1

© 2007 by Greg Watson; made available under the EPL 1.0

Parallel programming tools (cont...)

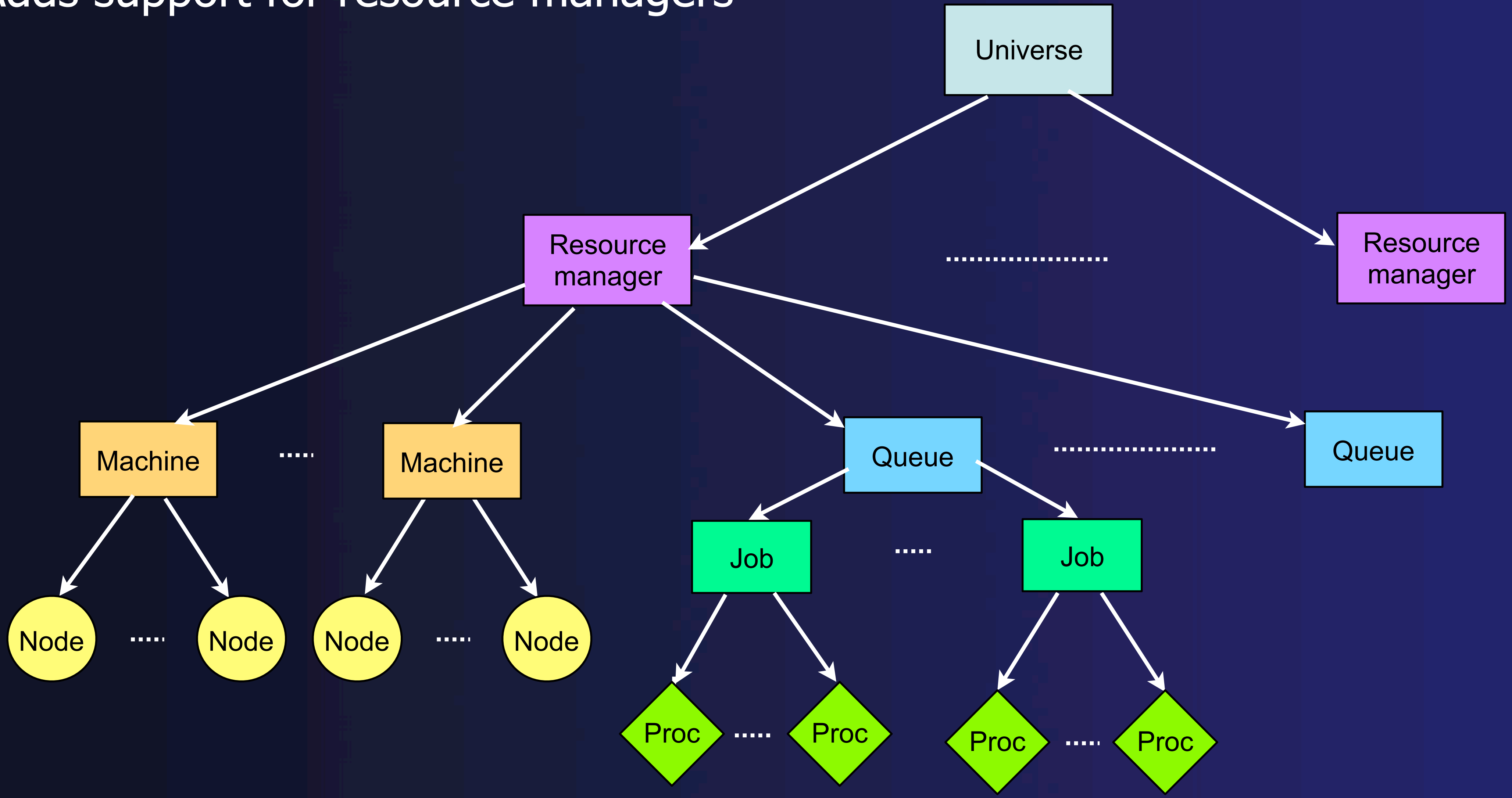


```
    double f,c,d;

    /* Allocate memory for the arrays. */
    double *x, *y;
    x = (double *) malloc( (size_t) (  arraySize * sizeof(double) ) );
    y = (double *) malloc( (size_t) (  arraySize * sizeof(double) ) );
    fillArray(x, arraySize);
    f=convergence(0.24691);

    #pragma omp parallel private(f)
    {
        a++;
        for(int i=0; i<a; i++) {
            d     =  exp(x[i]*x[i]);
            c = sin( exp( cos( - exp( sin(x[i]) ) ) ) );
            #pragma omp barrier
            a=d+y[i];
            if  (a==f)
              {if  (a==c)  d=c;}
            else {
              d=a;
              #pragma omp barrier
            }
            c += d-a;
            y[i]=c;
        }
    }
}
```

- Concurrency analysis shows which statements will be executed in parallel with highlighted statement
- Advanced error analysis detects if directives have been placed in incorrect locations

What’s coming in PTP 2.0

- New parallel model elements
  - Adds support for resource managers



© 2007 by Greg Watson; made available under the EPL 1.0

What's coming in PTP 2.0 (cont...)

- Job scheduler support
  - New resource manager system will allow job submission to multiple job schedulers from a single Eclipse session
  - Viewing of job status and job control will also be supported
  - Initial implementations for LSF, MOAB and SLURM
- Remote services
  - Allows Eclipse to run on local machine
  - Job submission, program launch, and program debugging on remote hosts

© 2007 by Greg Watson; made available under the EPL 1.0

What’s coming in PTP 2.0 (cont…)

- Parallel debugger enhancements
  - Scalability improvements
  - Support for non-gdb backend debuggers
  - New user interface features, including multi-variable viewer and array viewer
- Redesigned runtime system interface
  - Will allow installation via software update
- Parallel language tool enhancements
  - MPI analysis and checking tools
  - Fortran support

© 2007 by Greg Watson; made available under the EPL 1.0

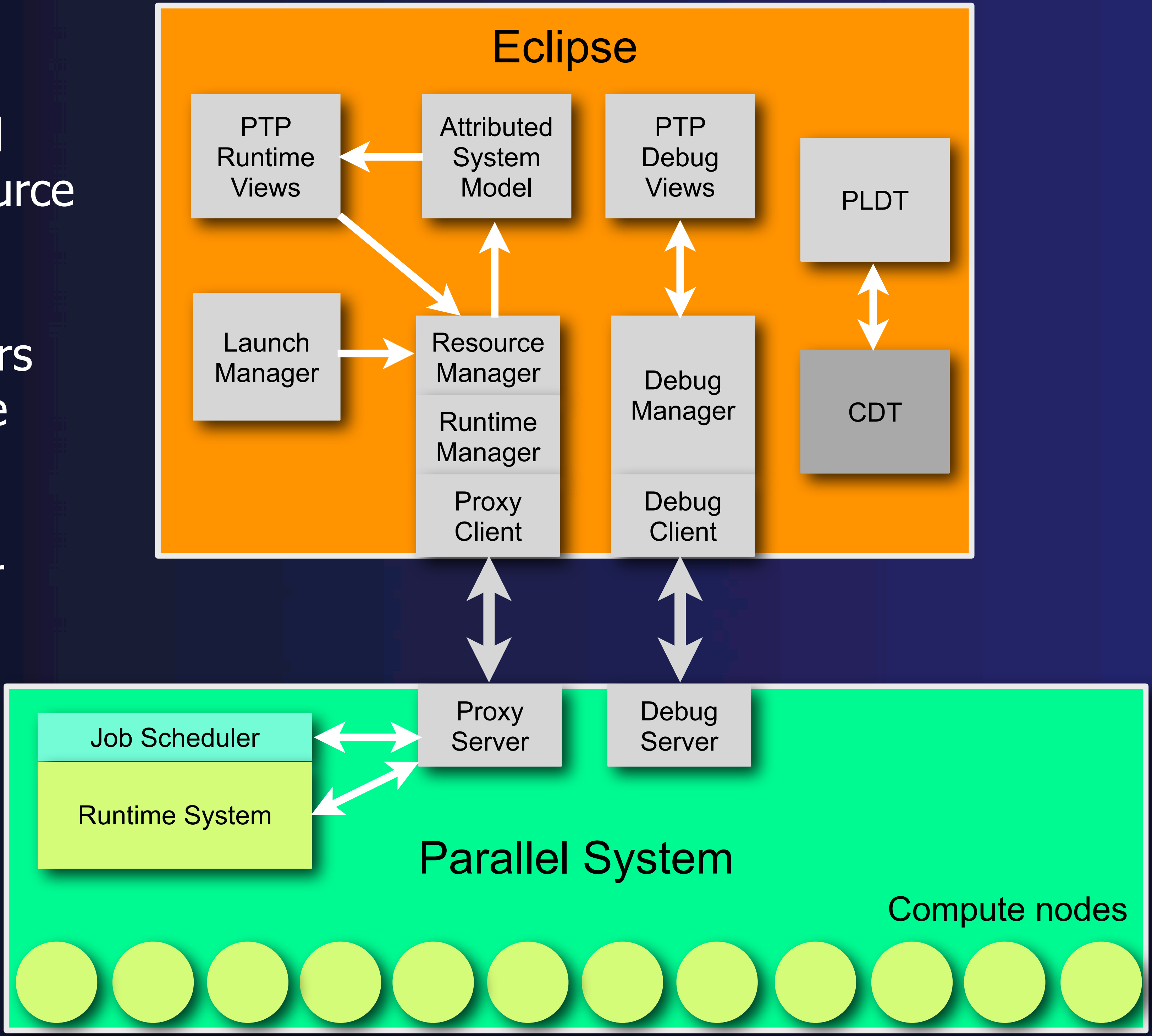
Architecture Overview

- Client/server model
  - In PTP 1.1 both client and server must run on same machine
  - In PTP 2.0 client will run on desktop, server on remote machine
- Client resides in Eclipse, controls internal model
- Server can be any language, manages interaction with runtime system, job scheduler, etc.
- Client and server communicate with simple, extensible text-based command/event protocol

© 2007 by Greg Watson; made available under the EPL 1.0

PTP Architecture Details

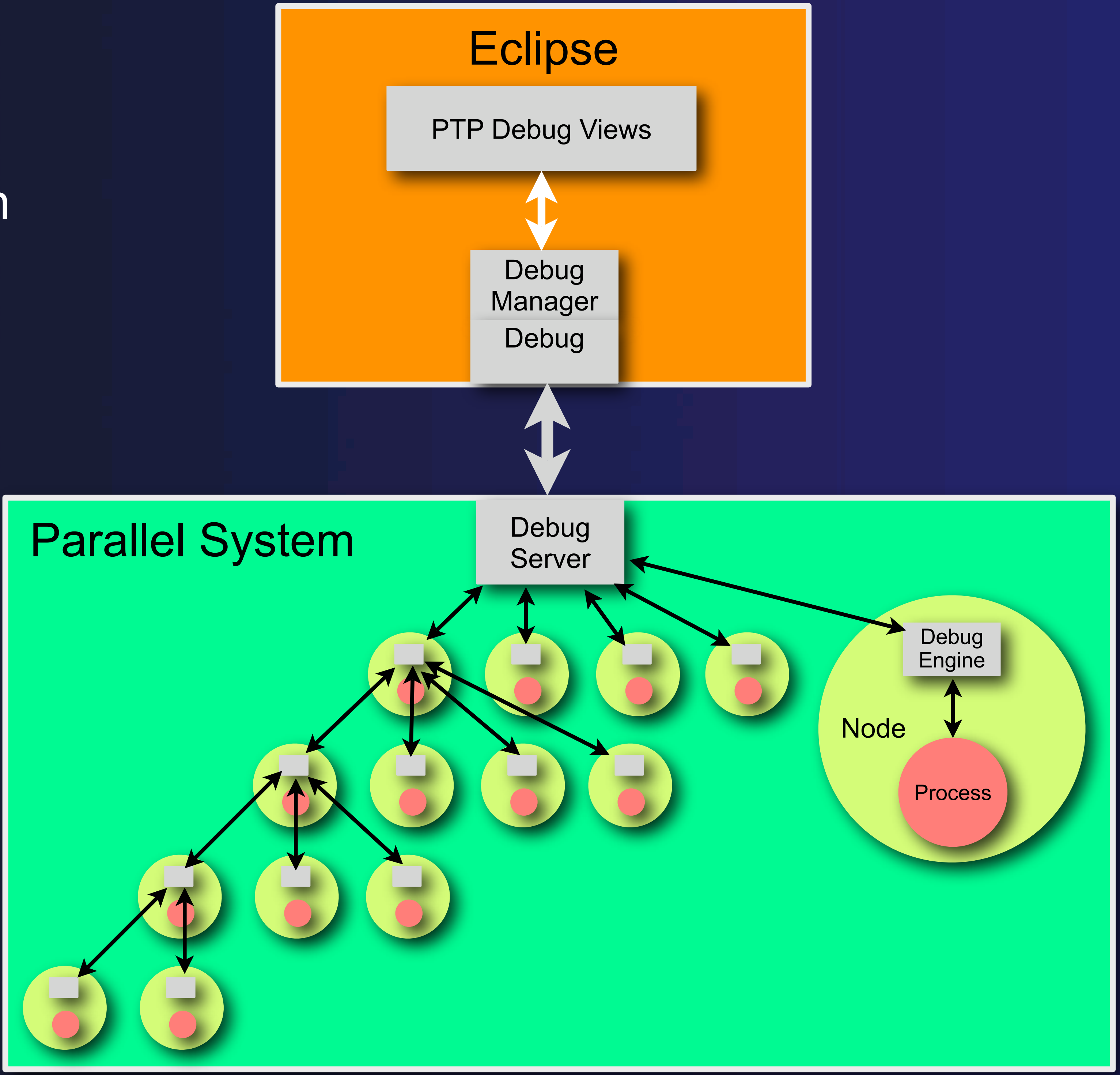
- Proxy and debug client/ servers use same protocol
- Support for different resource managers and runtime systems added as plugins
- Multiple resource managers can be in operation at one time
- SSH authentication for launching proxy/debugger



© 2007 by Greg Watson; made available under the EPL 1.0

PTP Debugger Details

- Debug server is an MPI program
- Debug engines are started on each node, one per process
- Debug engines act as message forwarders/ aggregators
- High level debug API allows replacement of debug server
- GDB currently used for low-level debug operations



© 2007 by Greg Watson; made available under the EPL 1.0

Conclusion

- PTP project has demonstrated steady progress over the last 2 years
- Community support and participation has continued to grow
- Eclipse is now being used in many government labs
- Great opportunity to take a leadership role in the project
- Many exciting opportunities for improving parallel development

© 2007 by Greg Watson; made available under the EPL 1.0

Resources

- PTP Project
  - http://eclipse.org/ptp
- OpenMPI
  - http://open-mpi.org
- MPICH2
  - http://www.mcs.anl.gov/mpi/mpich2
- OpenMP
  - http://www.openmp.org

© 2007 by Greg Watson; made available under the EPL 1.0