Parallel Application Development With Eclipse

Greg Watson, PTP Lead
Los Alamos National Laboratory

© 2006 by Greg Watson; made available under the EPL v1.0 | March 2006 | LA-UR-06-1917

What do we mean by “parallel”?

- Hardware

Parallel hardware

embedded

small-medium clusters

"big iron" supercomputers

Parallel hardware (cont…)

- Embedded
    - Usually small number of processes (< 8)
    - Tightly coupled (memory bus or equivalent)
- Common off the shelf (COTS) clusters
    - Usually less than 256 processors
    - Sometimes dual core or SMP nodes
    - Loosely coupled with high-speed interconnect (GigE or Infiniband)
- “Big Iron”
    - 256 - 131,072 processors
    - Clusters, MPP, vector, …
    - Myrinet, Infiniband, Quadrics, proprietary interconnects

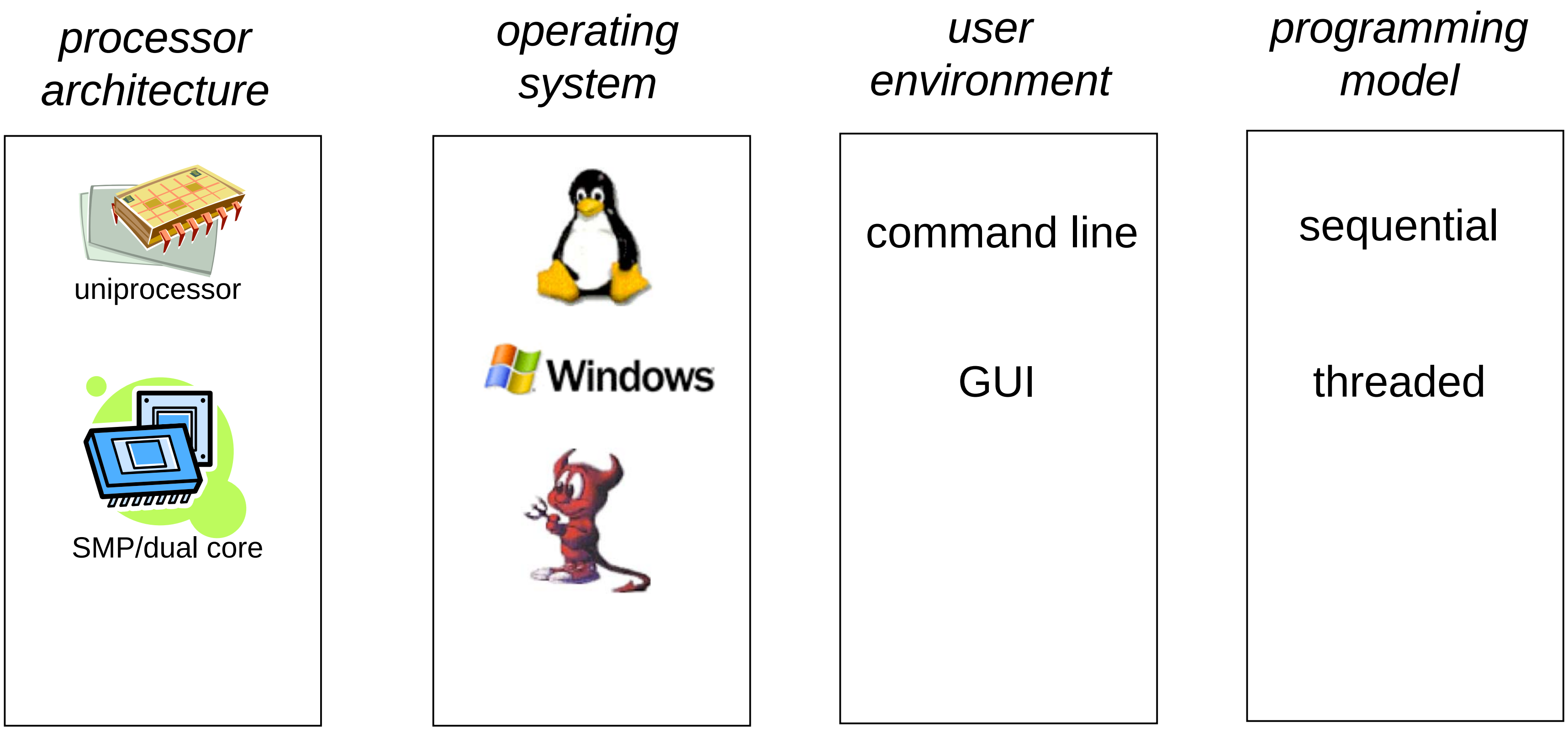
© 2006 by Greg Watson; made available under the EPL v1.0

eclipseCON 2006

What do we mean by “parallel”?

- Hardware
- Software

Traditional environment



processor architecture

uniprocessor

SMP/dual core

operating system

Windows

user environment

command line

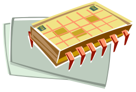
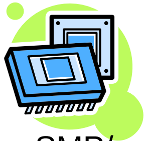
GUI

programming model

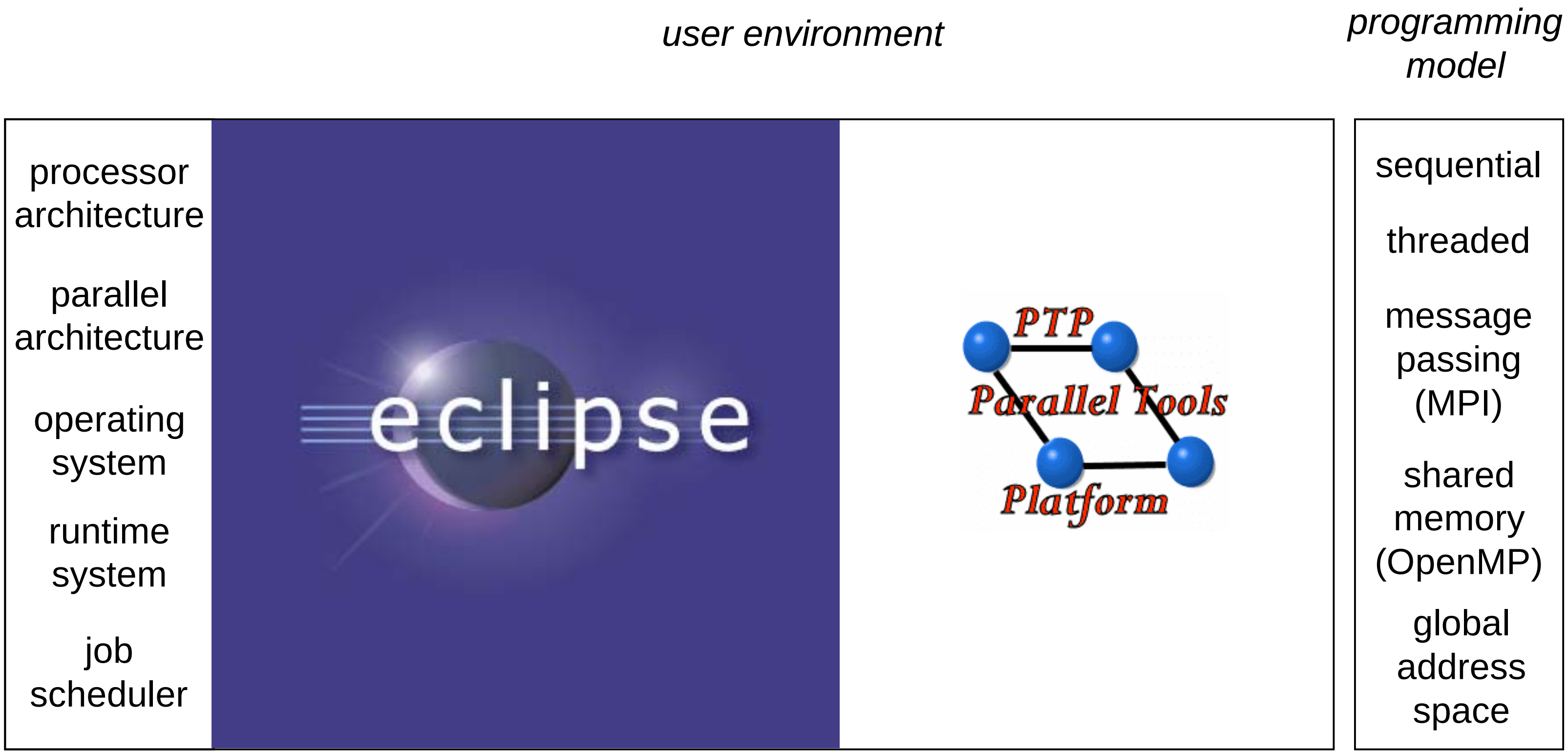
sequential

threaded

Parallel environment

processor architecture	*parallel architecture*	*operating system*	*runtime system*	*job scheduler*	*user environment*	*programming model*
 uniprocessor  SMP/ dual core	cluster MPP shared memory vector proprietary	 Windows	cluster (poe, oscar, rocks, xgrid) single system image mainframe	LSF OpenPBS LoadLeveler SLURM Condor	command line (GUI)	sequential threaded message passing (MPI) shared memory (OpenMP) global address space

PTP environment



user environment

programming model

processor architecture

parallel architecture

operating system

runtime system

job scheduler

sequential

threaded

message passing (MPI)

shared memory (OpenMP)

global address space

How does PTP help?

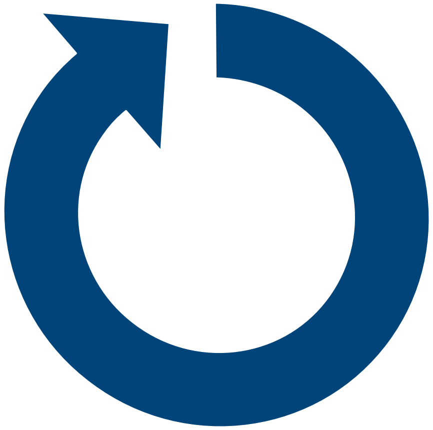
- Provides the benefits of an IDE for parallel programmers
- Hides much of the parallel system complexity
- Simplifies the parallel programming task
- Opportunity for new/advanced tools and languages

PTP ~~1.0~~ 0.9999 components

- Abstract parallel model
  - Machines, nodes, jobs, processes
- Parallel runtime perspective
  - Views: machines, jobs, processes
- Parallel launch
- Parallel debugger
  - Extends and implements CDT CDI
  - External debugger uses high level commands
  - Backend can be gdb, others
- Parallel programming tools (contributed by IBM Research)
  - Currently MPI only
- Fortran development tools

Parallel application development lifecycle

Design & Develop



Deploy & Log

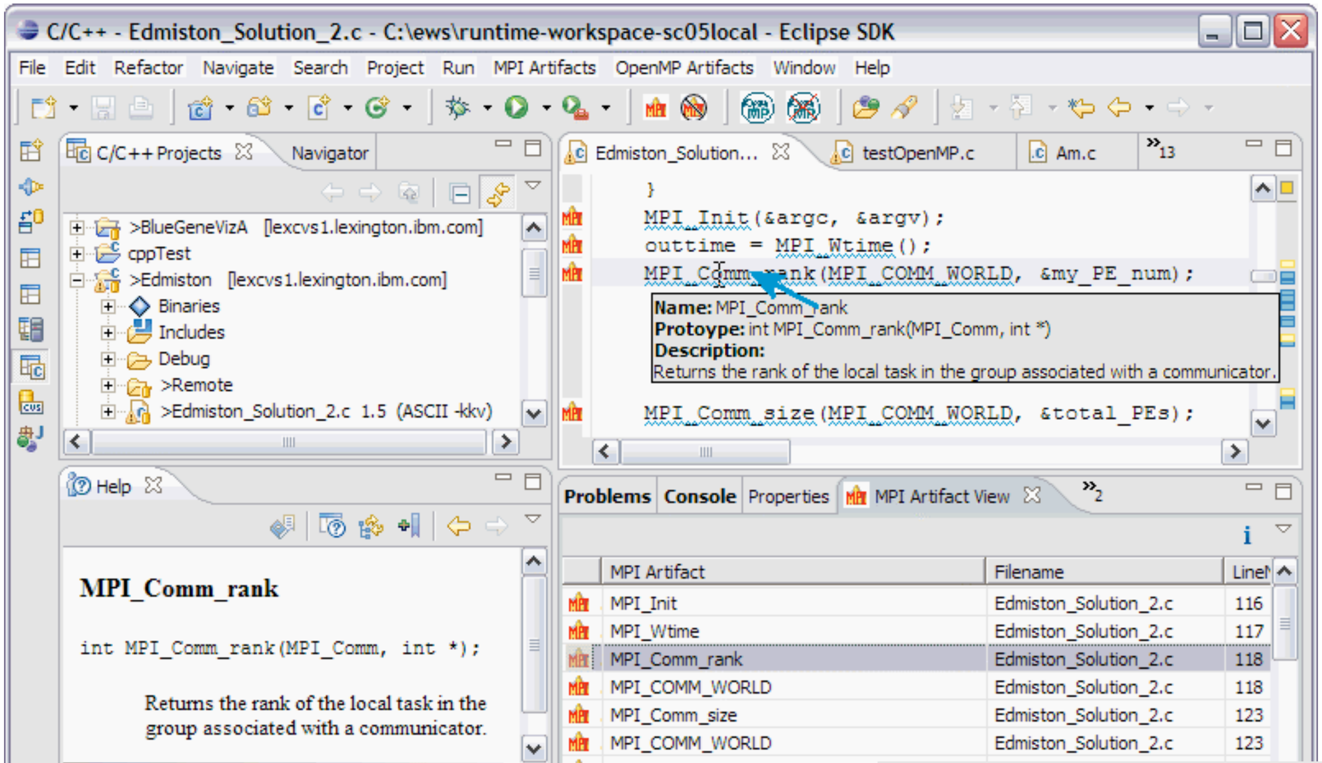
Debug & Test

Performance Analysis & Tuning

Program Development

- Embarrassingly parallel
  - Single program multiple data (SPMD)
- Message passing (MPI)
  - Send/receive
  - Collective opertions (e.g. broadcast, gather)
- C/C++ (CDT)
  - Refactoring is improving…
- Fortran (Photran)
  - No refactoring (yet)

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
>BlueGeneVizA [lexcvs1.lexington.ibm.com]
cppTest
>Edmiston [lexcvs1.lexington.ibm.com]
Binaries
Includes
Debug
>Remote
>Edmiston_Solution_2.c 1.5 (ASCII -kkv)
Edmiston_Solution...
testOpenMP.c
Am.c
Help
Go To:
All Topics
Search
Related Topics
Bookmarks
Problems
Console
Properties
MPI Artifact View

```
}
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

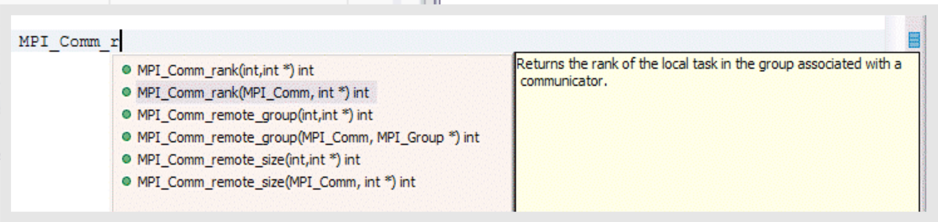
`int MPI_Comm_rank(MPI_Comm, int *);`

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

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		

- Identifies MPI artifacts
- Navigates to source code locations
- Help: hover, content assist, F1
- Currently C/C++ only

Content Assist

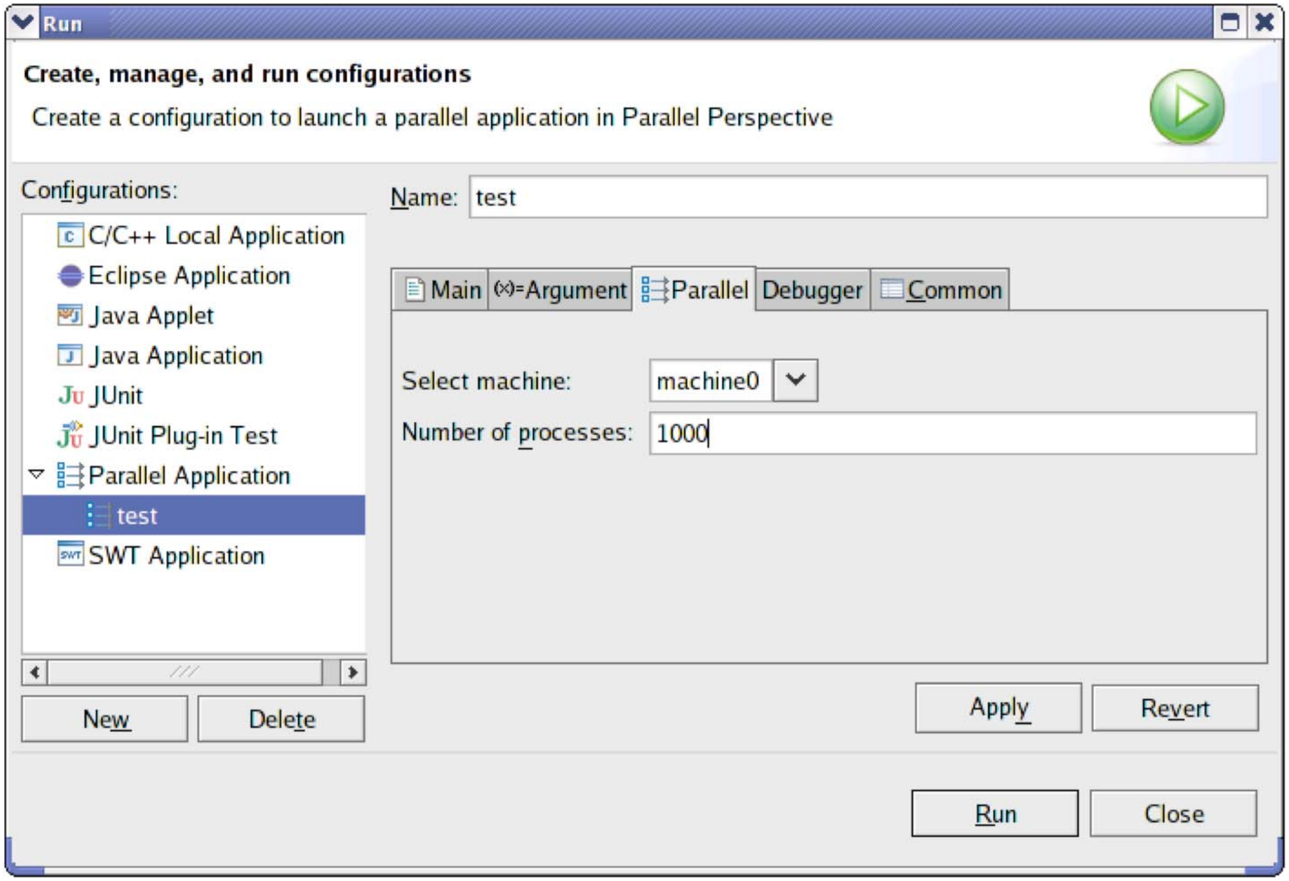


```
MPI_Comm_r
```

- 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.

Parallel launch



Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

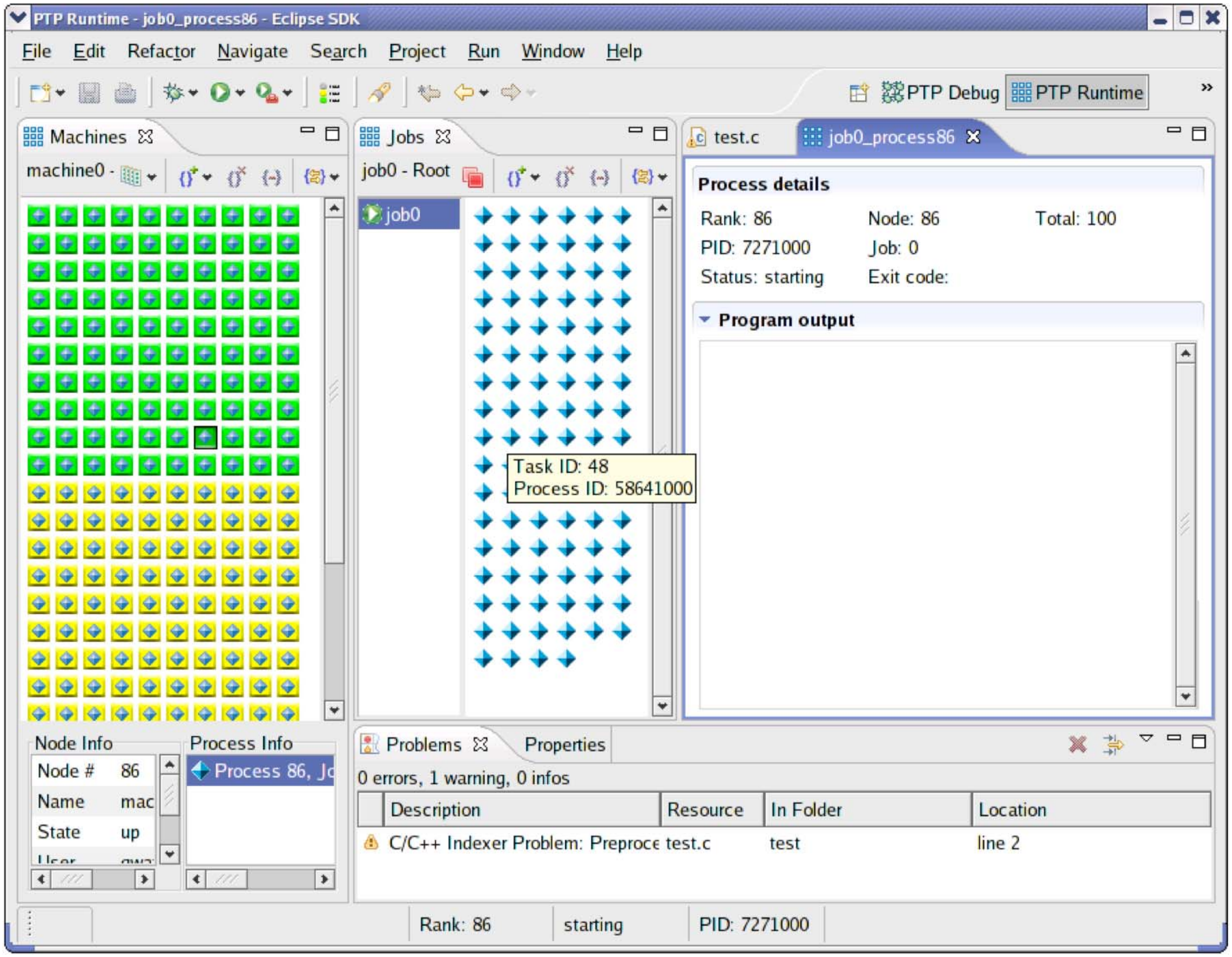
Name: test

Select machine: machine0

Number of processes: 1000

- Allows user to specify machine and number of processes to launch
- Can also configure parallel debug launch

Parallel runtime perspective



Task ID: 48
Process ID: 58641000

Process details

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

Node #	86
Name	mac
State	up
User	gwa

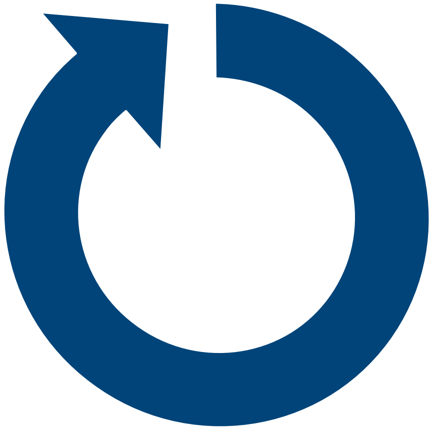
0 errors, 1 warning, 0 infos

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 nodes
- Jobs view
  - jobs launched, processes in a job and process status
  - Terminate job button
- Process details view
  - more detailed process information and standard output from process

Parallel application development lifecycle (cont…)



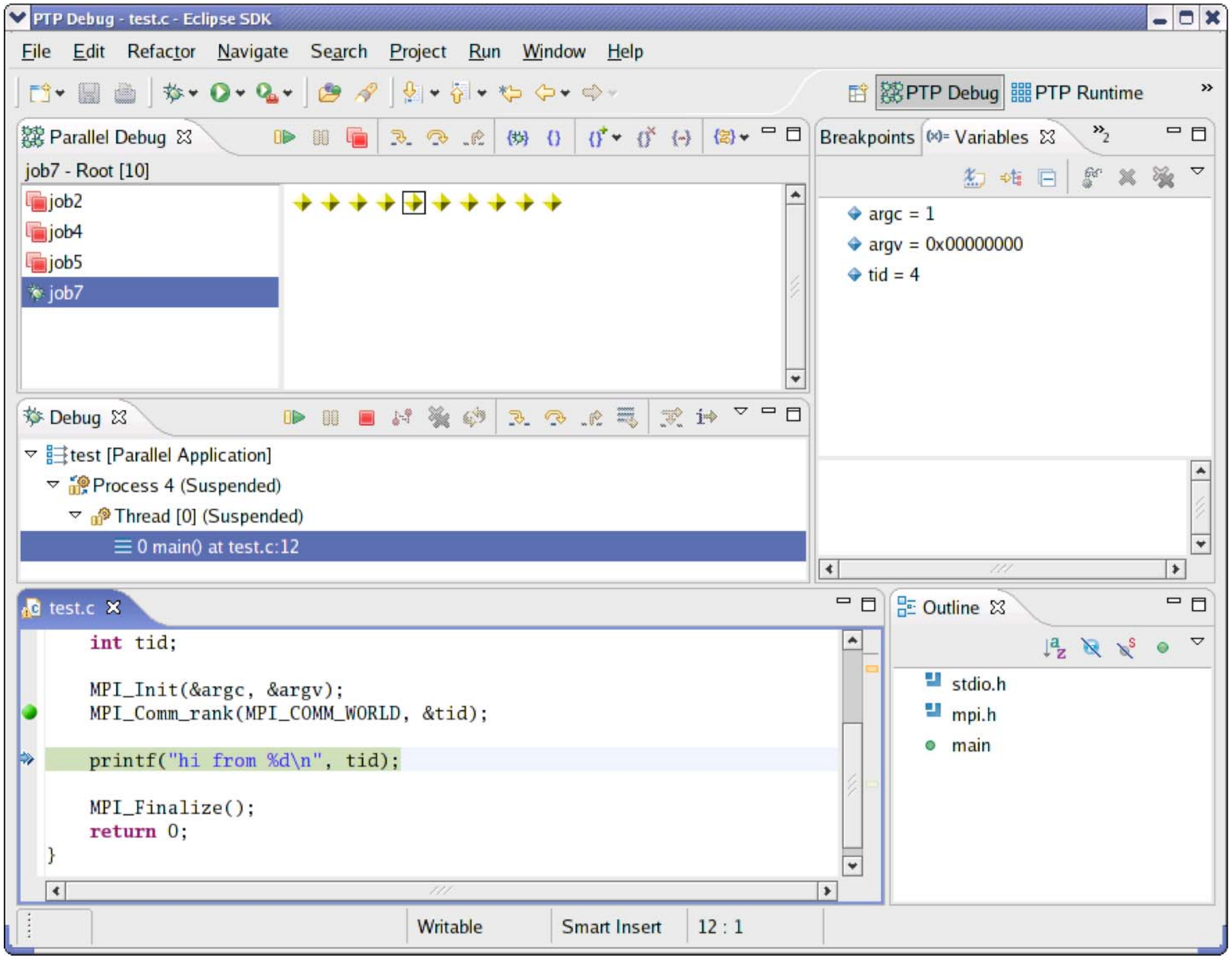
Design & Develop

Deploy & Log

Debug & Test

Performance Analysis & Tuning

Parallel debug perspective



job7 - Root [10]

- job2
- job4
- job5
- job7

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

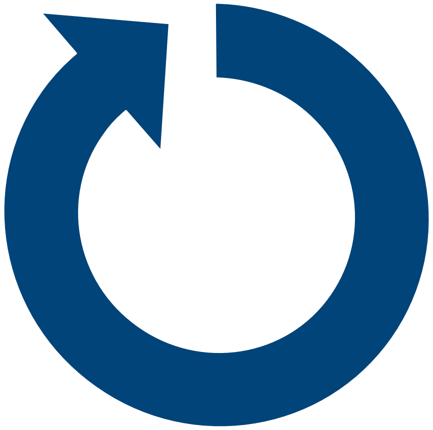
```
test [Parallel Application]
  Process 4 (Suspended)
    Thread [0] (Suspended)
      0 main() at test.c:12
```

```
    int tid;

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

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

    MPI_Finalize();
    return 0;
}
```

- stdio.h
- mpi.h
- main

- Adds parallel debug view
  - Debug jobs launched, processes in a job and process status
  - Used to specify process sets
  - Registered processes displayed in Debug view
  - Tooltip displays variable values
- Extended breakpoints and current location markers
  - Breakpoint color shows which set the breakpoint applies to
  - Multiple simultaneous current location markers
- Tested to 1024 processes
  - Model tested to 90,000
  - UI tested to 400,000

Parallel application development lifecycle (cont…)



Design & Develop

Debug & Test

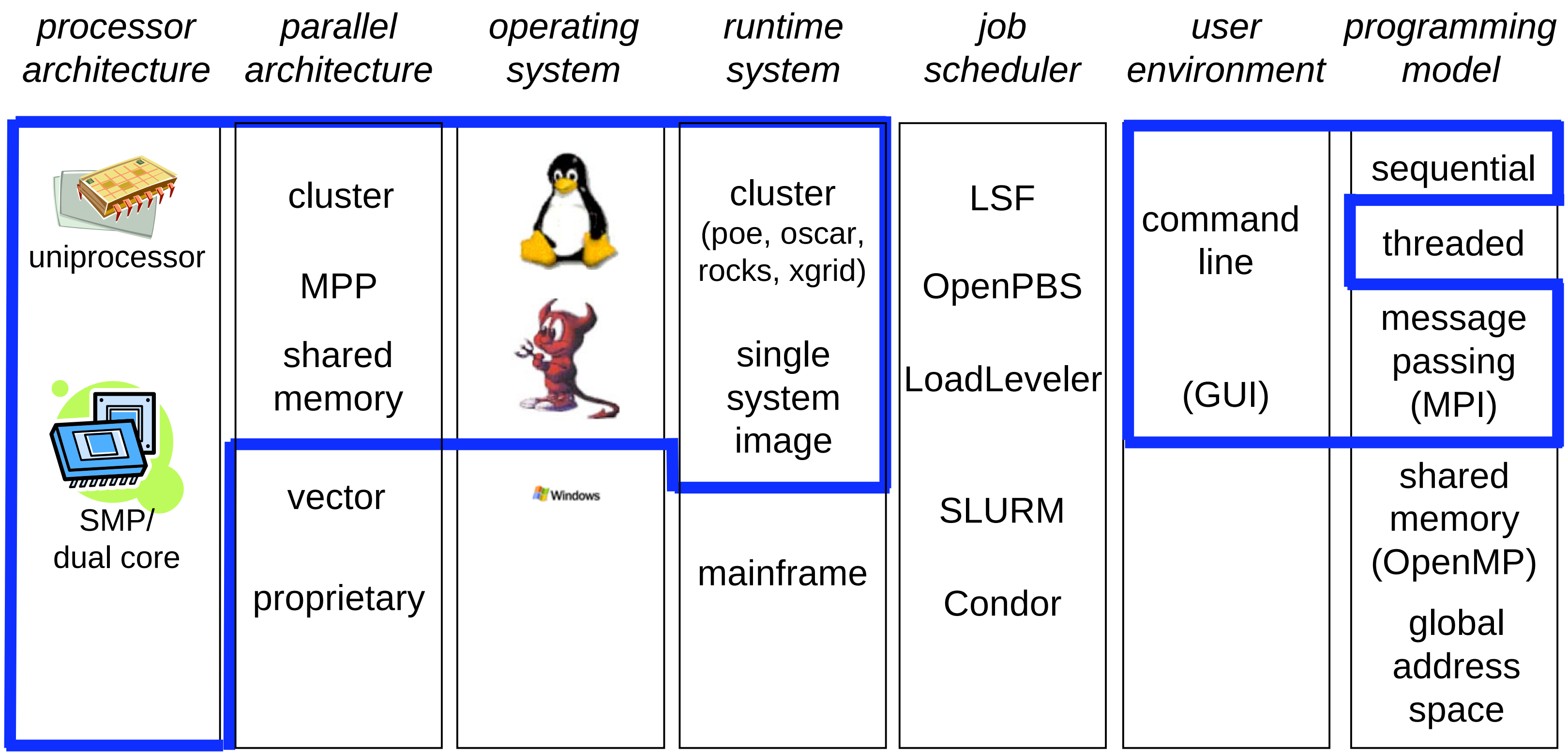
Performance Analysis & Tuning

Deploy & Log

Performance analysis tools

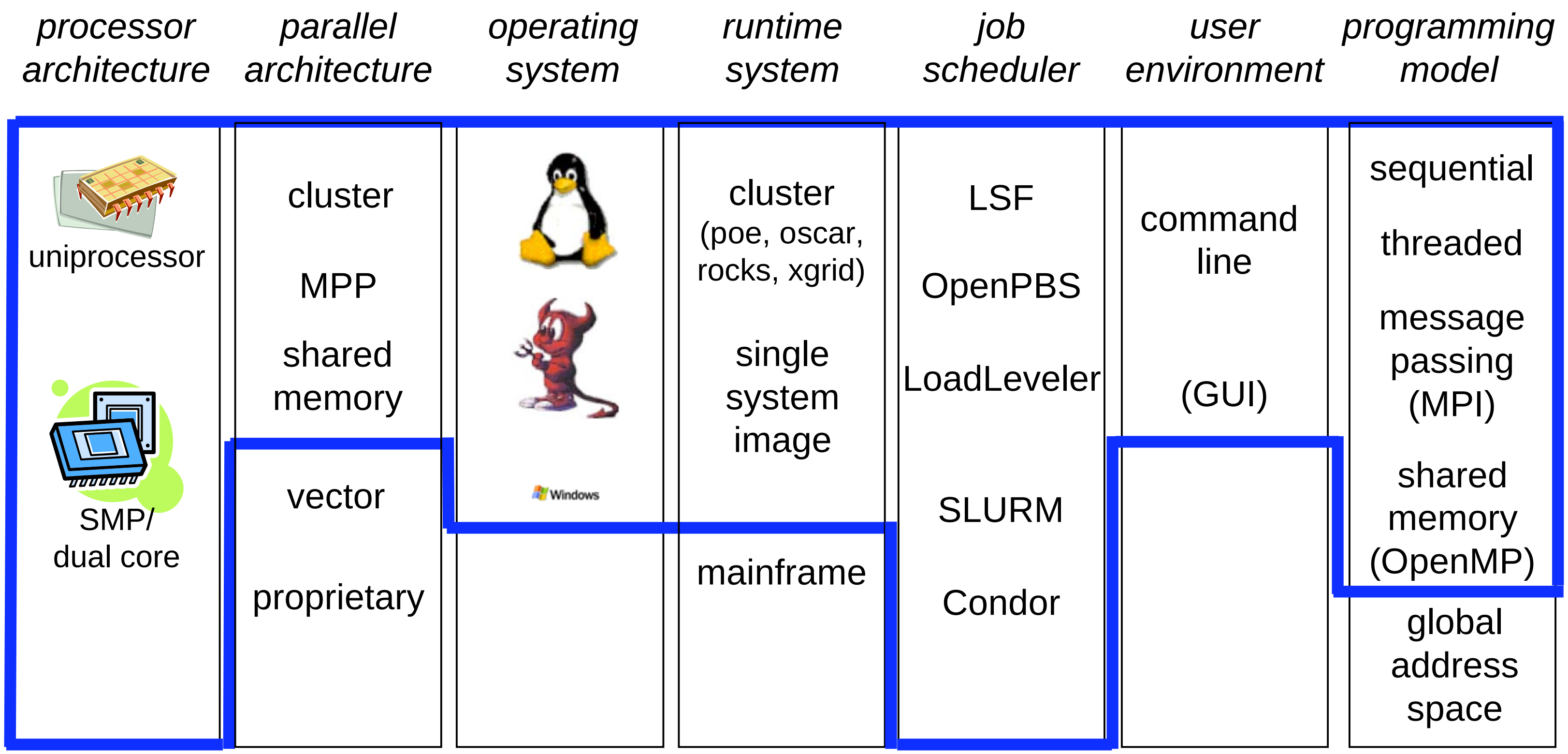
- Test and Performance Tools Platform (TPTP)
  - for sequential applications
- Testing and Analysis Utilities (TAU) stage 1 integration
  - Integration of existing TAU tools allows performance analysis of projects within the PTP
  - C/C++ and Fortran projects can be automatically instrumented and compiled with TAU libraries from within Eclipse
  - Performance data output is automatically organized
  - The Paraprof tool can be launched automatically to visualize profile output

PTP 1.0 support



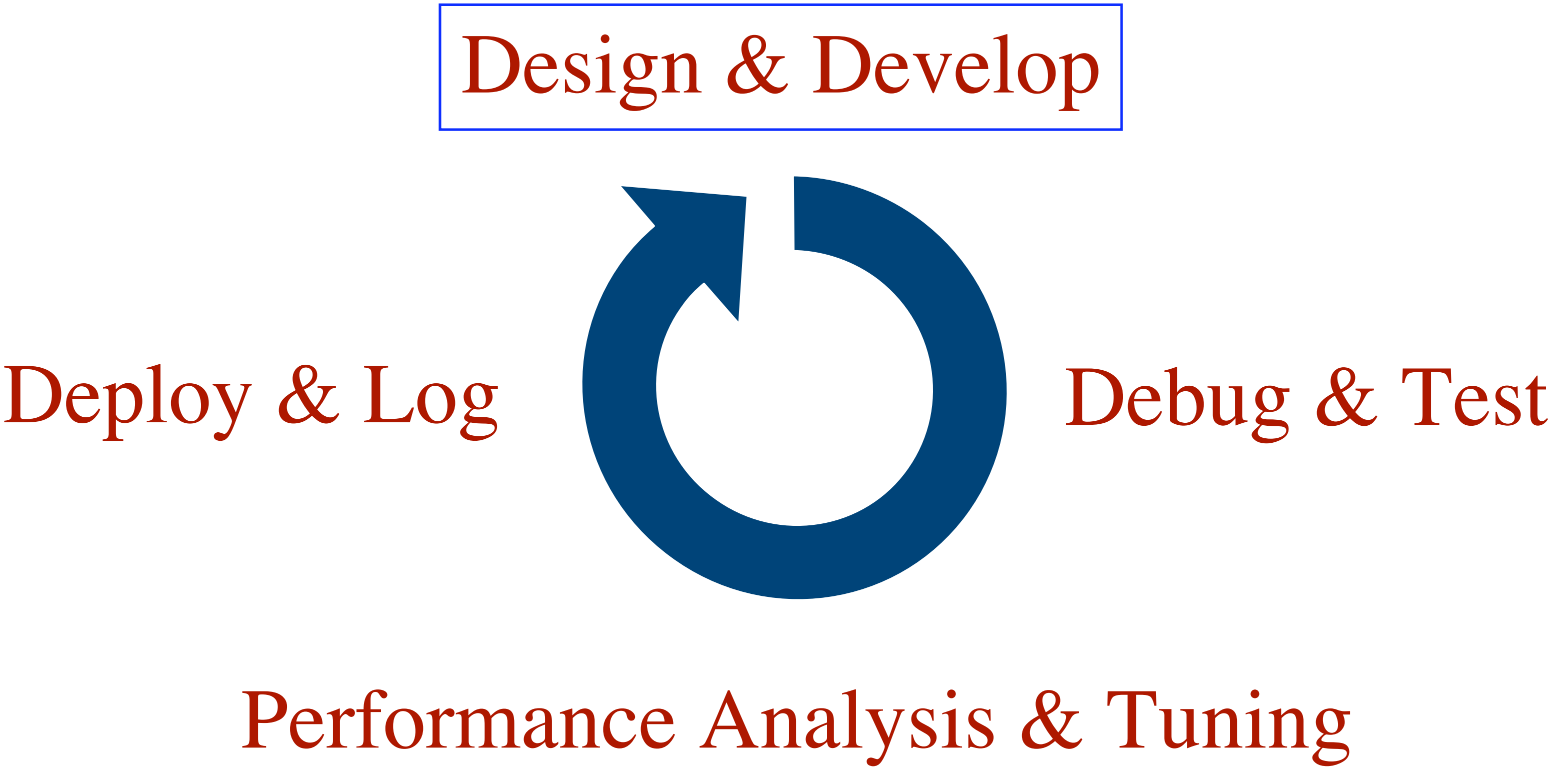
processor architecture	*parallel architecture*	*operating system*	*runtime system*	*job scheduler*	*user environment*	*programming model*
uniprocessor SMP/ dual core	cluster MPP shared memory vector proprietary	Windows	cluster (poe, oscar, rocks, xgrid) single system image mainframe	LSF OpenPBS LoadLeveler SLURM Condor	command line (GUI)	sequential threaded message passing (MPI) shared memory (OpenMP) global address space

PTP 2.0 support



processor architecture	*parallel architecture*	*operating system*	*runtime system*	*job scheduler*	*user environment*	*programming model*
uniprocessor SMP/ dual core	cluster MPP shared memory vector proprietary	Windows	cluster (poe, oscar, rocks, xgrid) single system image mainframe	LSF OpenPBS LoadLeveler SLURM Condor	command line (GUI)	sequential threaded message passing (MPI) shared memory (OpenMP) global address space

Future plans



Design & Develop

Debug & Test

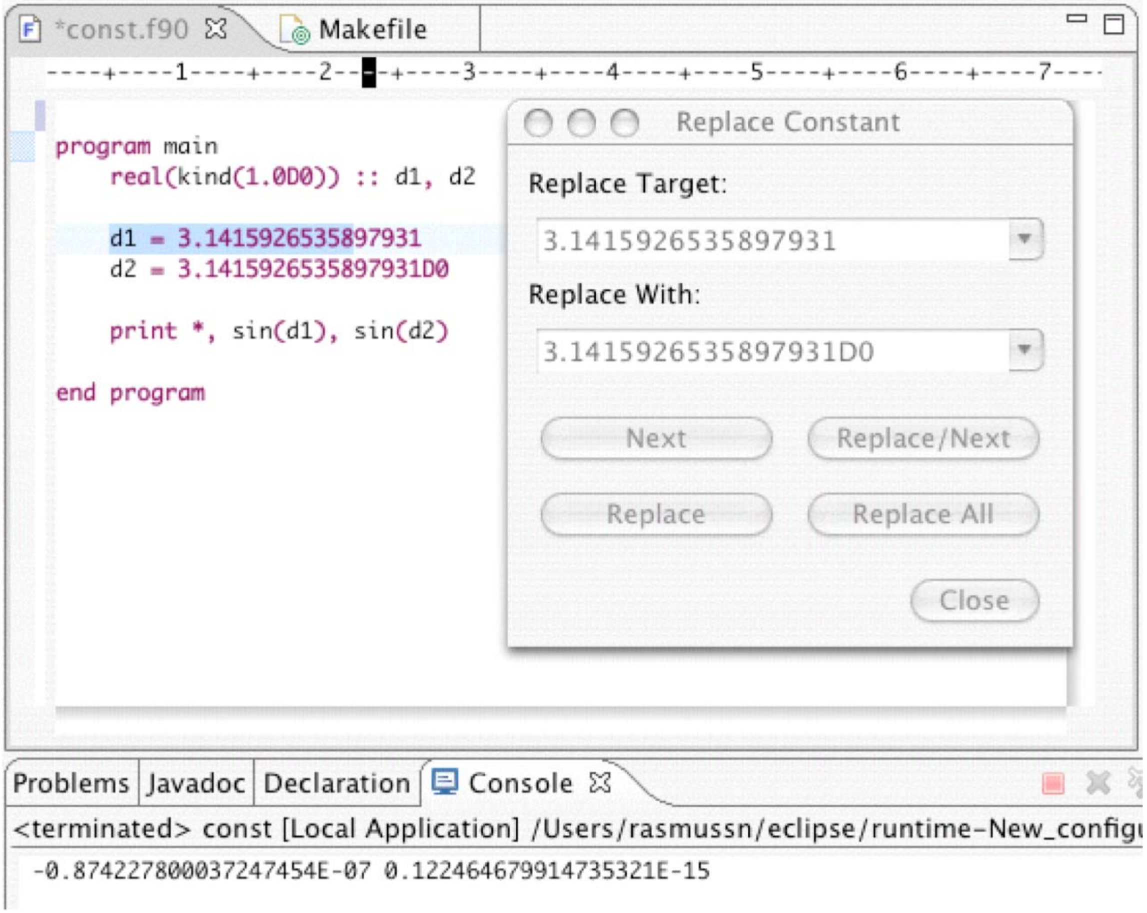
Performance Analysis & Tuning

Deploy & Log

Development capabilities

- Static analysis infrastructure
  - Embedded C++ and Fortran parsers
  - External parsers
- Refactoring for scientific computing
  - Language interoperability
  - Fortran: constant replacement and type promotion
  - Semi-automatic parallelization (OpenMP)
- Program graphs (e.g. call tree graphs)
- Error checking
- Assisted documentation generation (Doxygen)
- Support for new parallel languages
  - Co-array Fortran, UPC

Enhanced Fortran tools



```
program main
    real(kind(1.0D0)) :: d1, d2

    d1 = 3.1415926535897931
    d2 = 3.1415926535897931D0

    print *, sin(d1), sin(d2)

end program
```

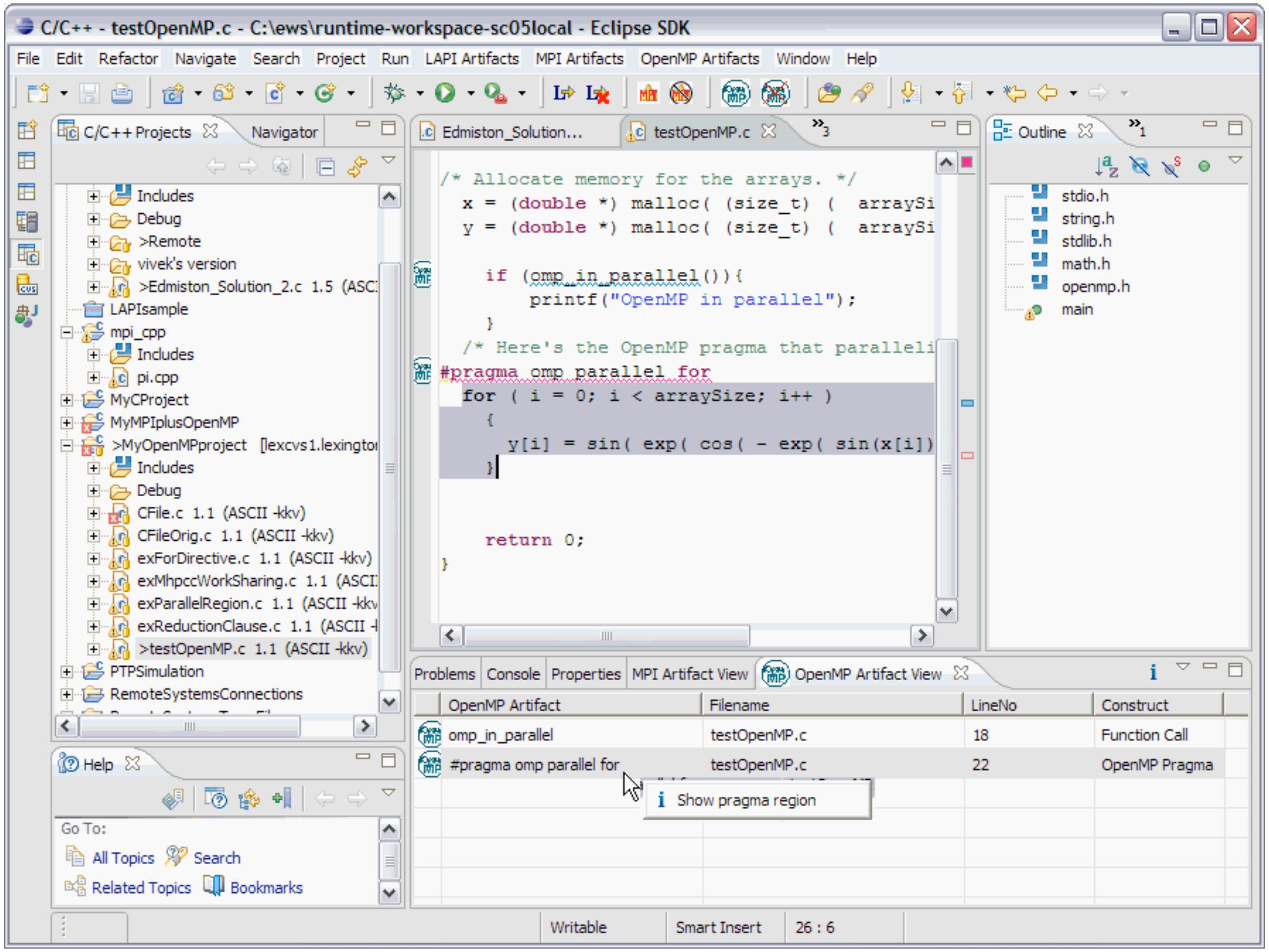
Replace Target: 3.1415926535897931

Replace With: 3.1415926535897931D0

```
<terminated> const [Local Application] /Users/rasmussn/eclipse/runtime-New_configu
 -0.874227800037247454E-07 0.122464679914735321E-15
```

- Constant replacement refactoring
- Other refactorings for scientific computing

New parallel language programming tools

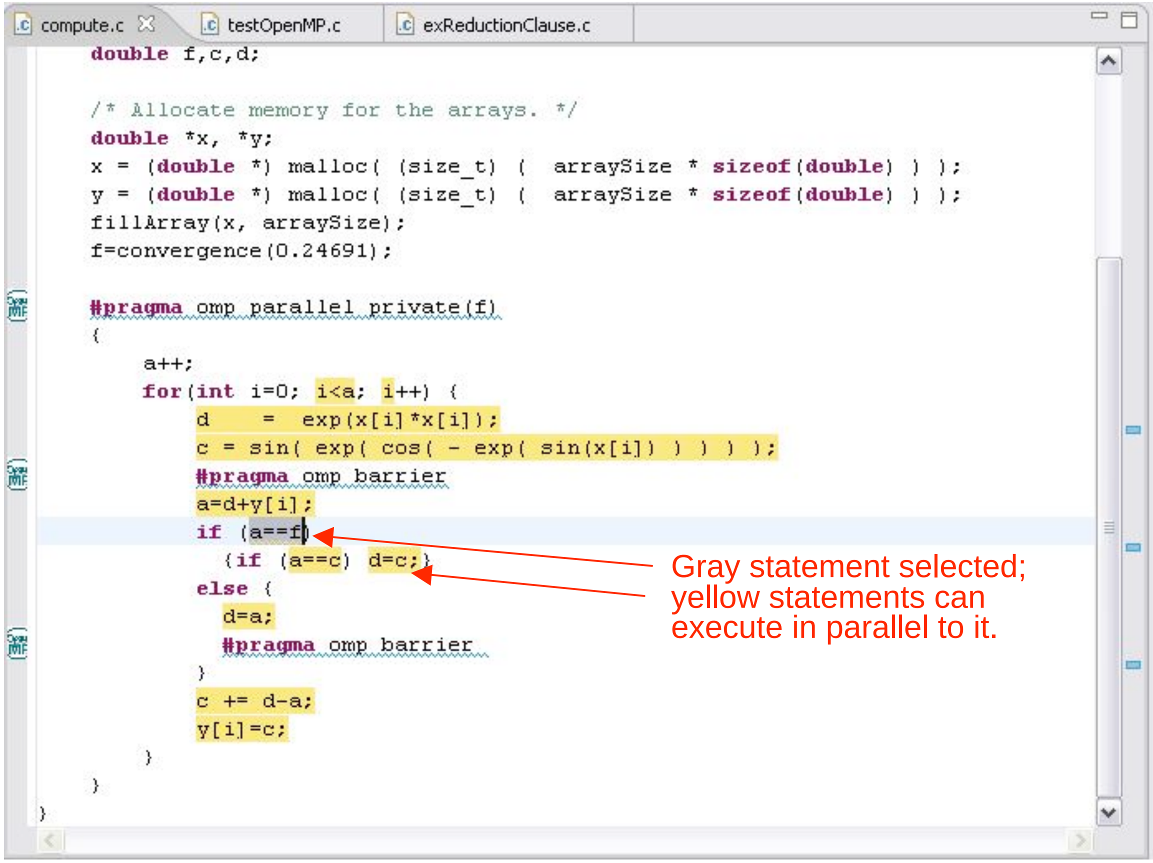

C/C++ - testOpenMP.c - C:\ews\runtime-workspace-sc05local - Eclipse SDK
File Edit Refactor Navigate Search Project Run LAPI Artifacts MPI Artifacts OpenMP Artifacts Window Help
C/C++ Projects
Navigator
Includes
Debug
>Remote
vivek's version
>Edmiston_Solution_2.c 1.5 (ASC
LAPIsample
mpi_cpp
Includes
pi.cpp
MyCProject
MyMPIplusOpenMP
>MyOpenMPproject [lexcvs1.lexingto
Includes
Debug
CFile.c 1.1 (ASCII -kkv)
CFileOrig.c 1.1 (ASCII -kkv)
exForDirective.c 1.1 (ASCII -kkv)
exMhpccWorkSharing.c 1.1 (ASCI
exParallelRegion.c 1.1 (ASCII -kkv
exReductionClause.c 1.1 (ASCII -
>testOpenMP.c 1.1 (ASCII -kkv)
PTPSimulation
RemoteSystemsConnections
Help
Go To:
All Topics
Search
Related Topics
Bookmarks
Edmiston_Solution...
testOpenMP.c
Outline
stdio.h
string.h
stdlib.h
math.h
openmp.h
main
Problems
Console
Properties
MPI Artifact View
OpenMP Artifact View
Show pragma region
Writable
Smart Insert
26 : 6

```
/* Allocate memory for the arrays. */
  x = (double *) malloc( (size_t) (  arraySi
  y = (double *) malloc( (size_t) (  arraySi

    if (omp_in_parallel()){
        printf("OpenMP in parallel");
    }
  /* Here's the OpenMP pragma that paralleli
#pragma omp parallel for
  for ( i = 0; i < arraySize; i++ )
    {
      y[i] = sin( exp( cos( - exp( sin(x[i])
    }


    return 0;
}
```

OpenMP Artifact	Filename	LineNo	Construct
omp_in_parallel	testOpenMP.c	18	Function Call
#pragma omp parallel for	testOpenMP.c	22	OpenMP Pragma

- Identifies OpenMP artifacts
- Show #pragma region
- Help: hover, content assist, F1

OpenMP concurrency analysis

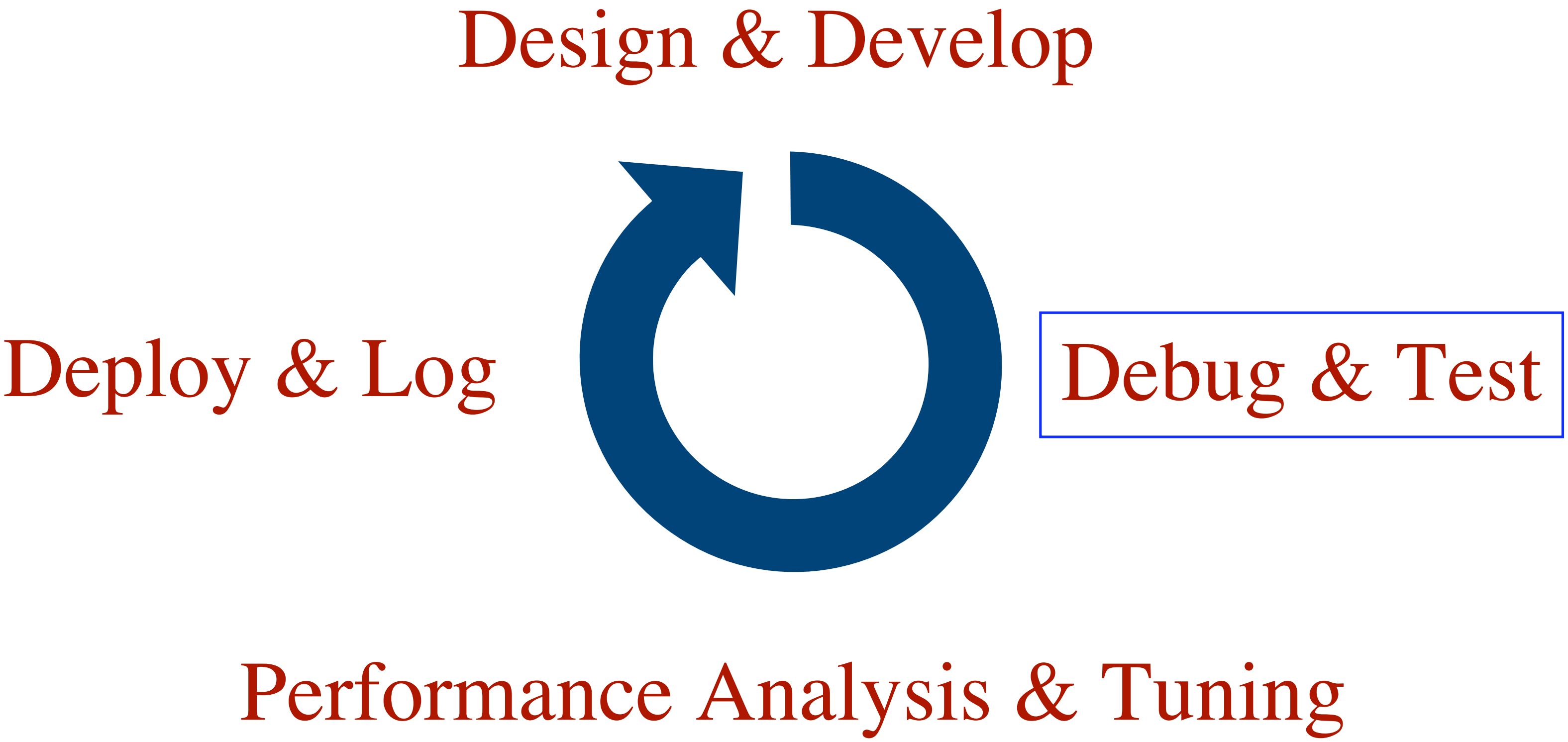


```
    double f,c,d;

    /* 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;
        }
    }
}
```

Gray statement selected; yellow statements can execute in parallel to it.

- Concurrency analysis
    - Compute and visualize statement/expression level concurrency in OpenMP parallel regions
- Analysis to assist in code parallelization
    - For shared/private variable determination, e.g. race detection
    - For restructuring code into parallel regions

Future plans (cont…)



Design & Develop

Deploy & Log

Debug & Test

Performance Analysis & Tuning

Parallel debugging

- Scalability improvements
- Array viewer and distributed array viewer
- Debug event hooks
- New (non-gdb) backend debugger support
- Data-centric debugging
- MPI-specific debugging

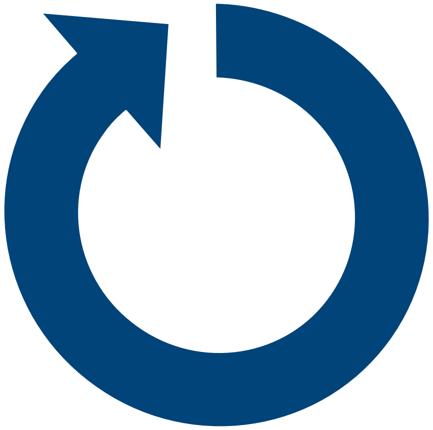
Parallel testing

- Provide integrated test environment
  - Deployment and execution of tests
  - Execution history analysis and reporting
- Automated unit test stub generation
  - Fortran and C/C++
  - Output results to spreadsheet (for example)
- Integrate Verification and Validation
  - Logging

© 2006 by Greg Watson; made available under the EPL v1.0

Future plans (cont…)

Design & Develop



Deploy & Log

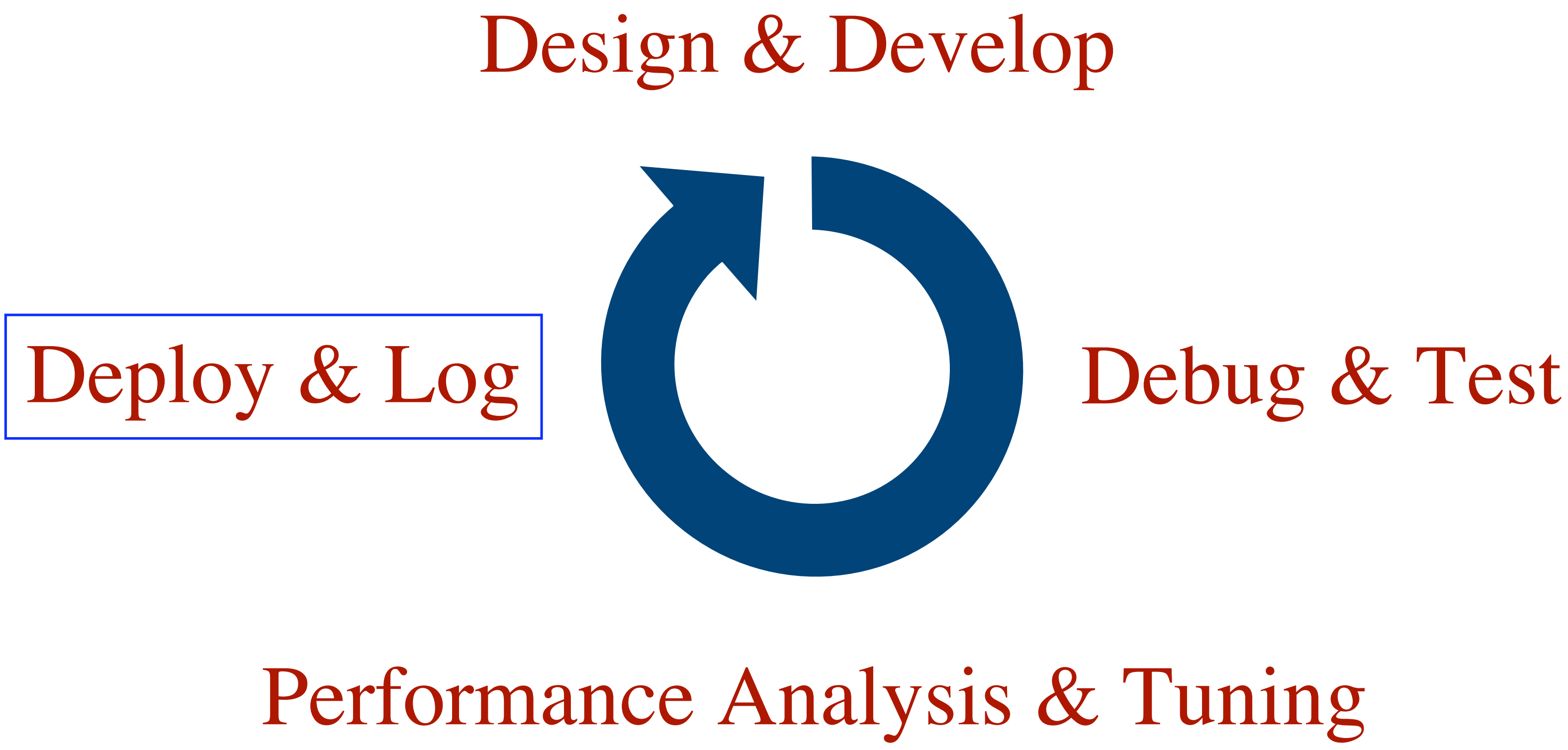
Debug & Test

Performance Analysis & Tuning

Future performance analysis and tuning

- Provide framework for performance analysis tools
  - Automatic instrumentation
  - Data collection
  - Data analysis
  - Visualization
- Framework for performance tuning using empirical methods
  - Launch code
  - Gather performance data
  - Analyze performance data
  - Code/Compiler optimization
  - Repeat (search space)

Future plans (cont…)



Design & Develop

Deploy & Log

Debug & Test

Performance Analysis & Tuning

Deployment and logging

- Deployable packages
  - Autoconf based
  - Explicit listing of dependencies on other tools
  - Package version
  - List of interfaces (for components)
- Web-based deployment
  - Automatic configure/build/install on download
- Provide history through logging
  - Build tools (compilers and options, libraries, versions, …)
  - Test history
  - Run history (traces, profiles, …)

Conclusion

- PTP 1.0 available for download
  - Supports small set of architectures/platforms
  - Can be used for real C/C++/Fortran parallel applications
  - Parallel debugger
- Ongoing development
  - New new architectures and platforms
  - Better Fortran tools
  - New/improved language tools
  - Parallel performance analysis and tuning tools

- Demo of PTP on a real cluster (Wed 2:00pm)
- Come to BOF (Wed 6:30pm)

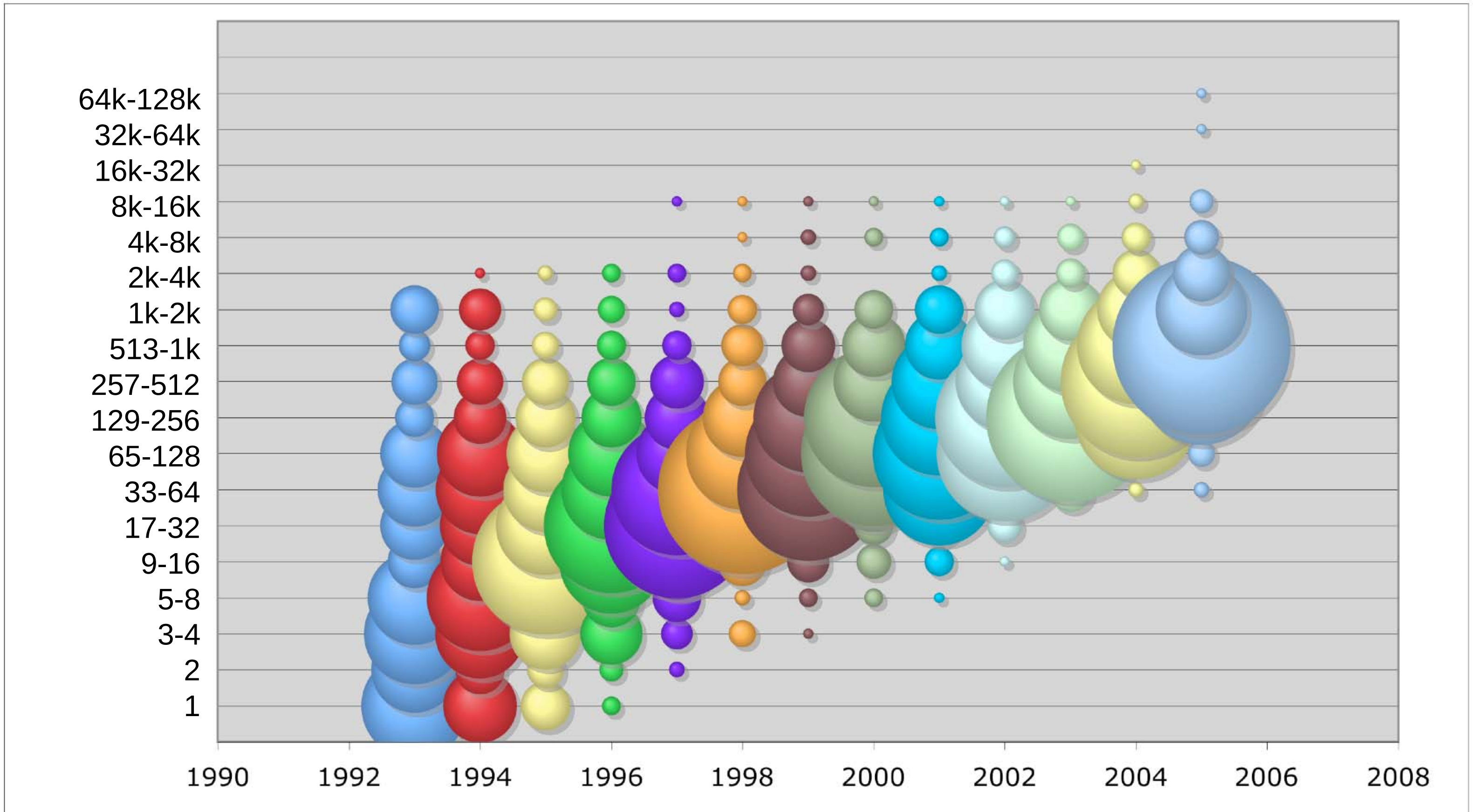
What makes a parallel supercomputer?

- Lots of processors (thousands)
- Lots of memory (terabytes)
- Lots of disk (terabytes)
- High bandwidth & low latency network (GB/s & < 10us)

As of November 2005, #1:
280,600 GFlops = 274 TFlops
(BlueGene/L)

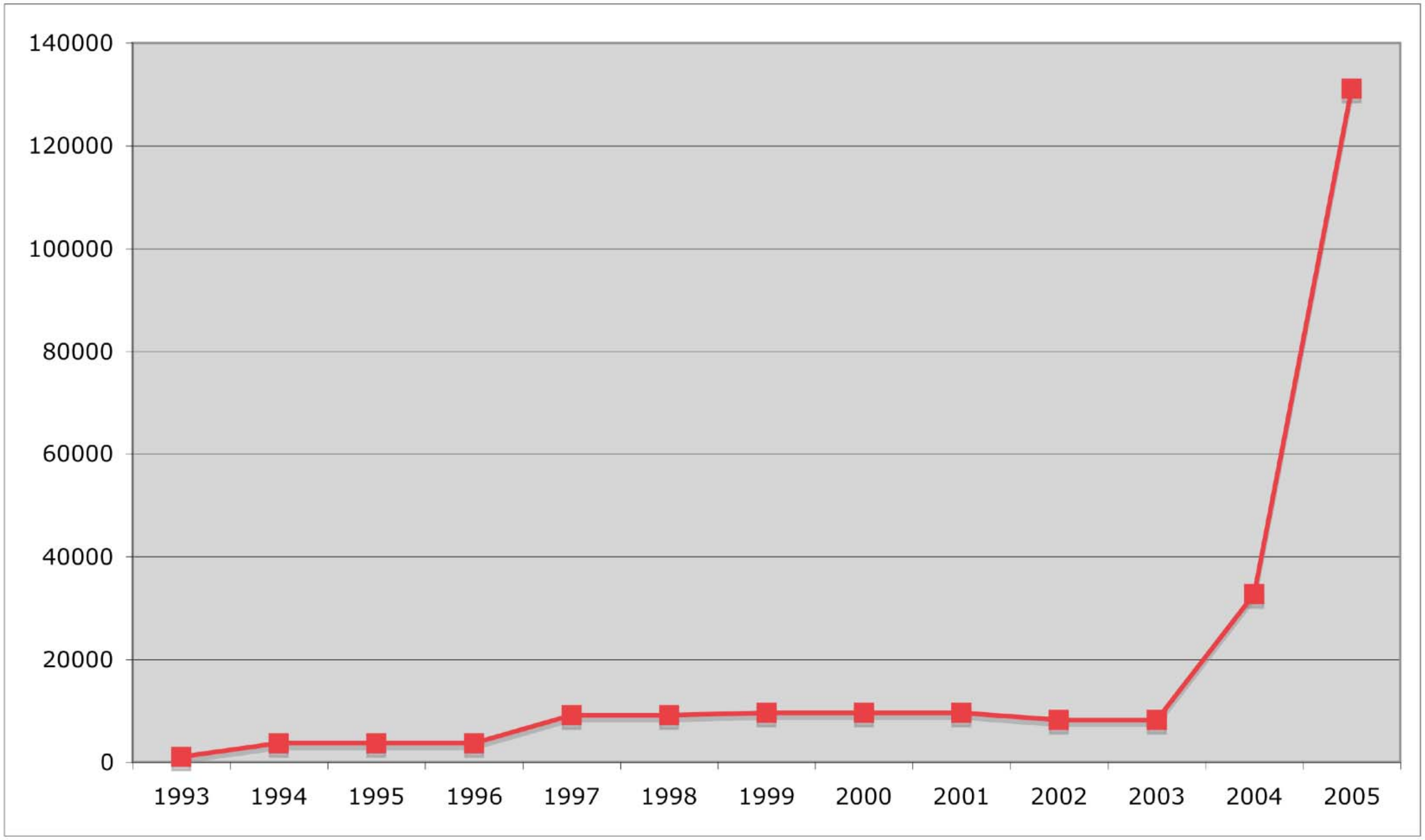
1 PFlop within 5-10 years

Supercomputer evolution (processors)



Source: http://www.top500.org

Supercomputer evolution (max processors)

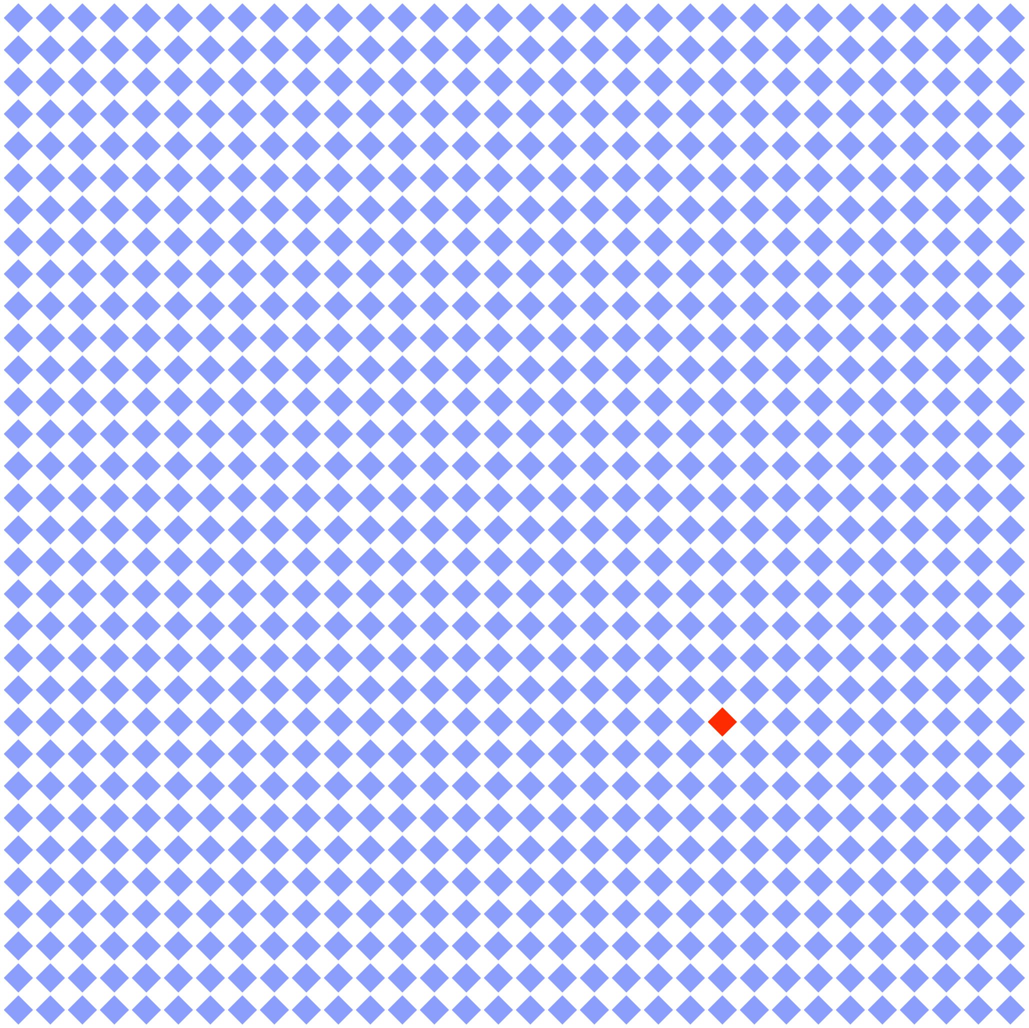


Source: http://www.top500.org

Eclipse is a visual medium

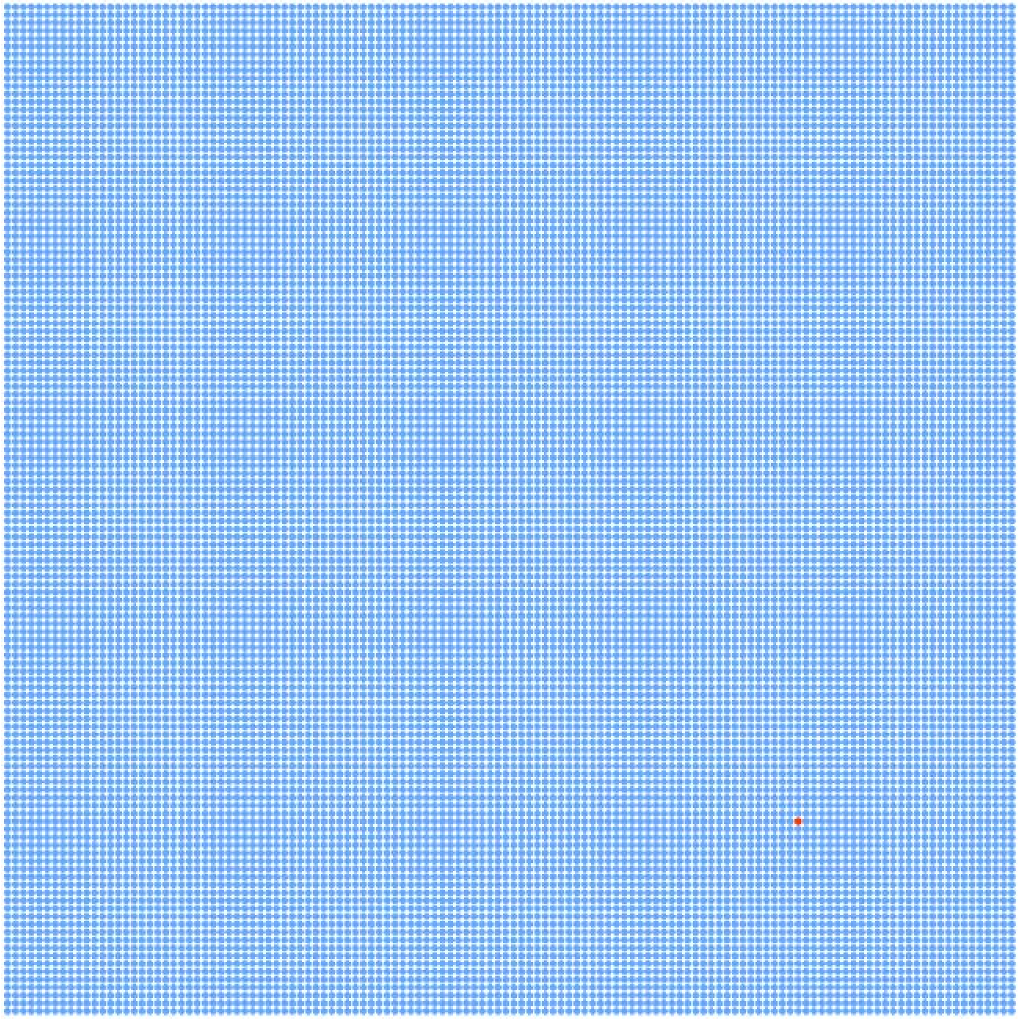
- Parallel computing is dealing with large numbers of “things”, such as :
  - Processors
  - Processes
  - Threads
  - Messages
  - Etc.
- How are we going to represent lots of “things”?

Visualizing “things” (1,024)



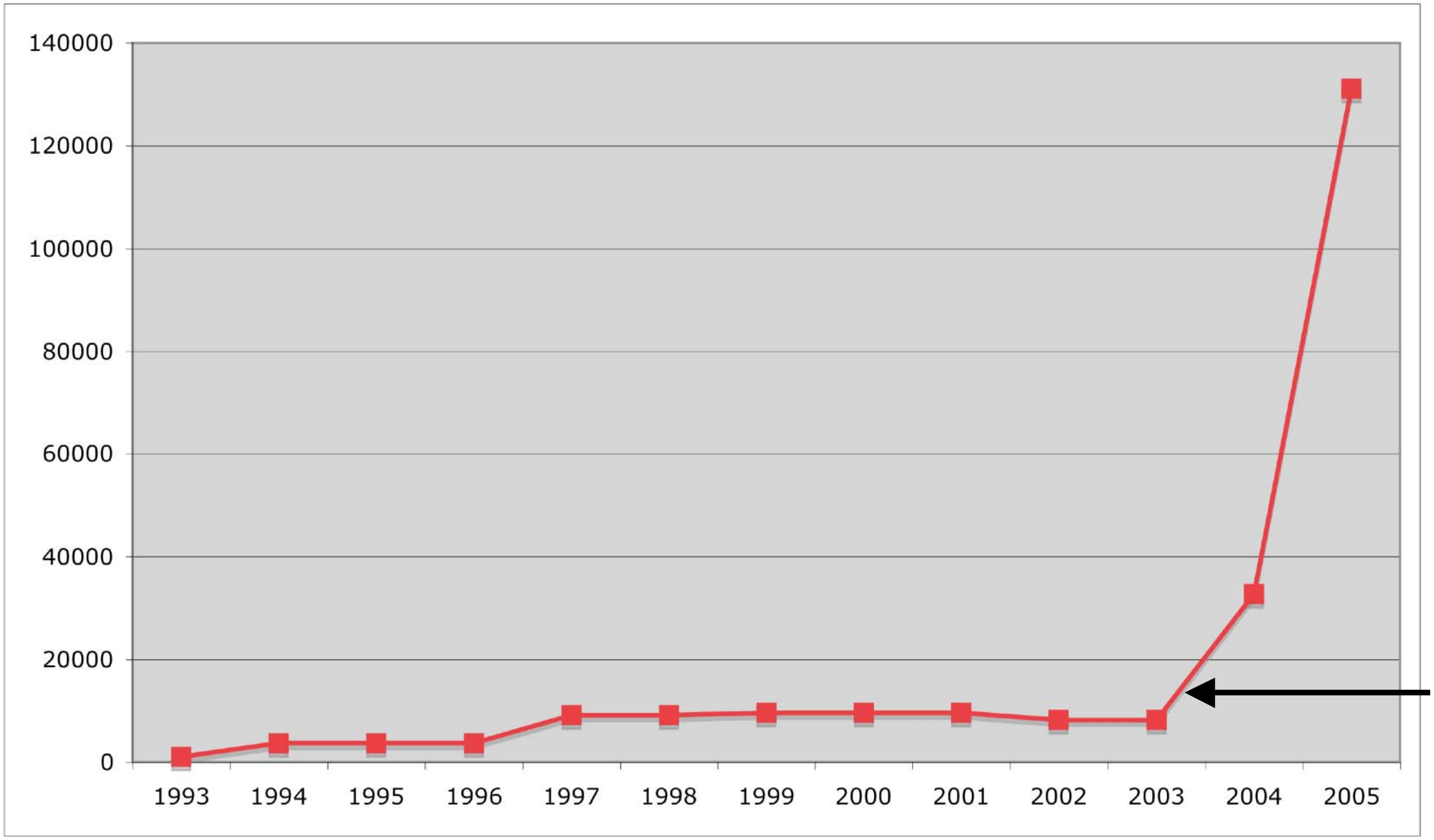
© 2006 by Greg Watson; made available under the EPL v1.0

Visualizing “things” (16,384)



© 2006 by Greg Watson; made available under the EPL v1.0

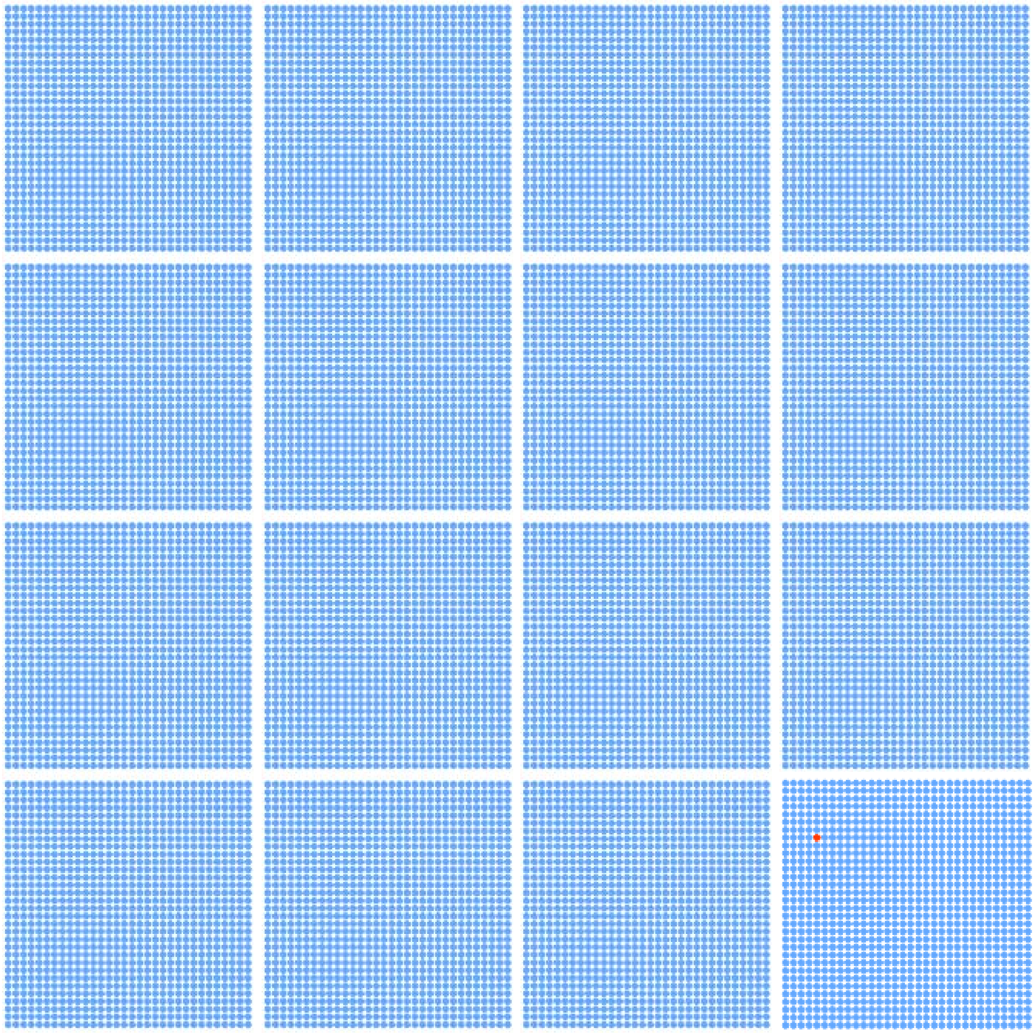
Where are 16,384 “things” (= processors)?



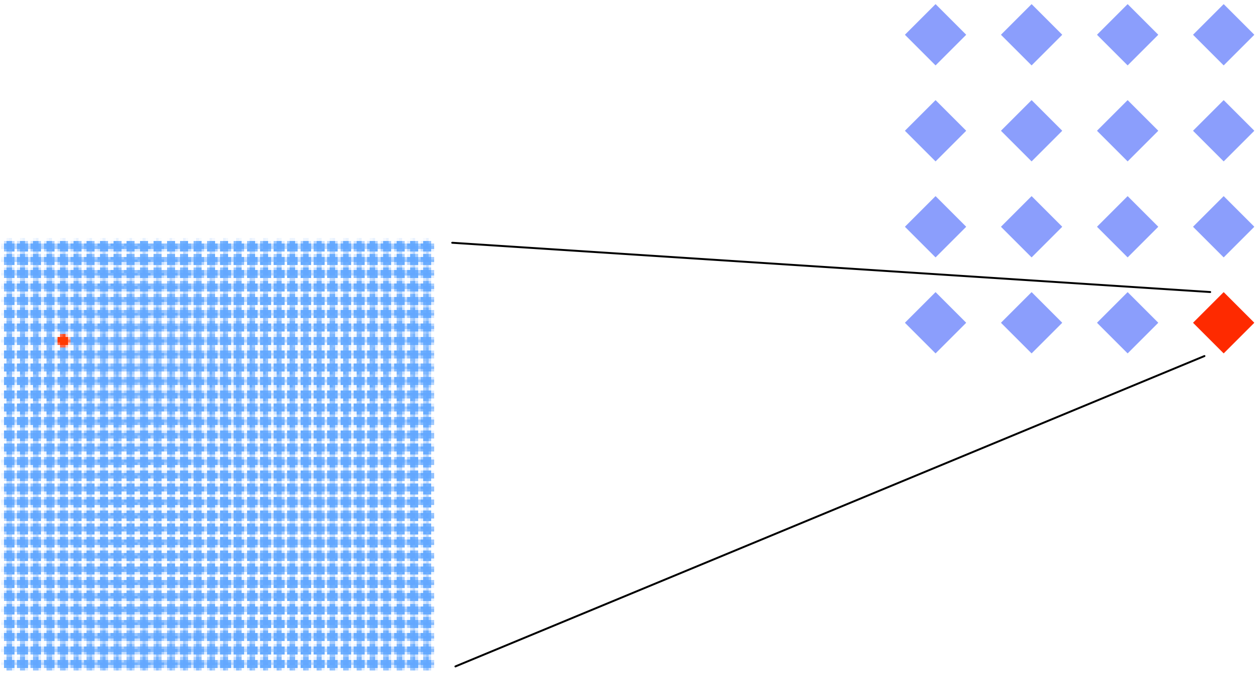
16384

Source: http://www.top500.org

Visualizing things (16,384)

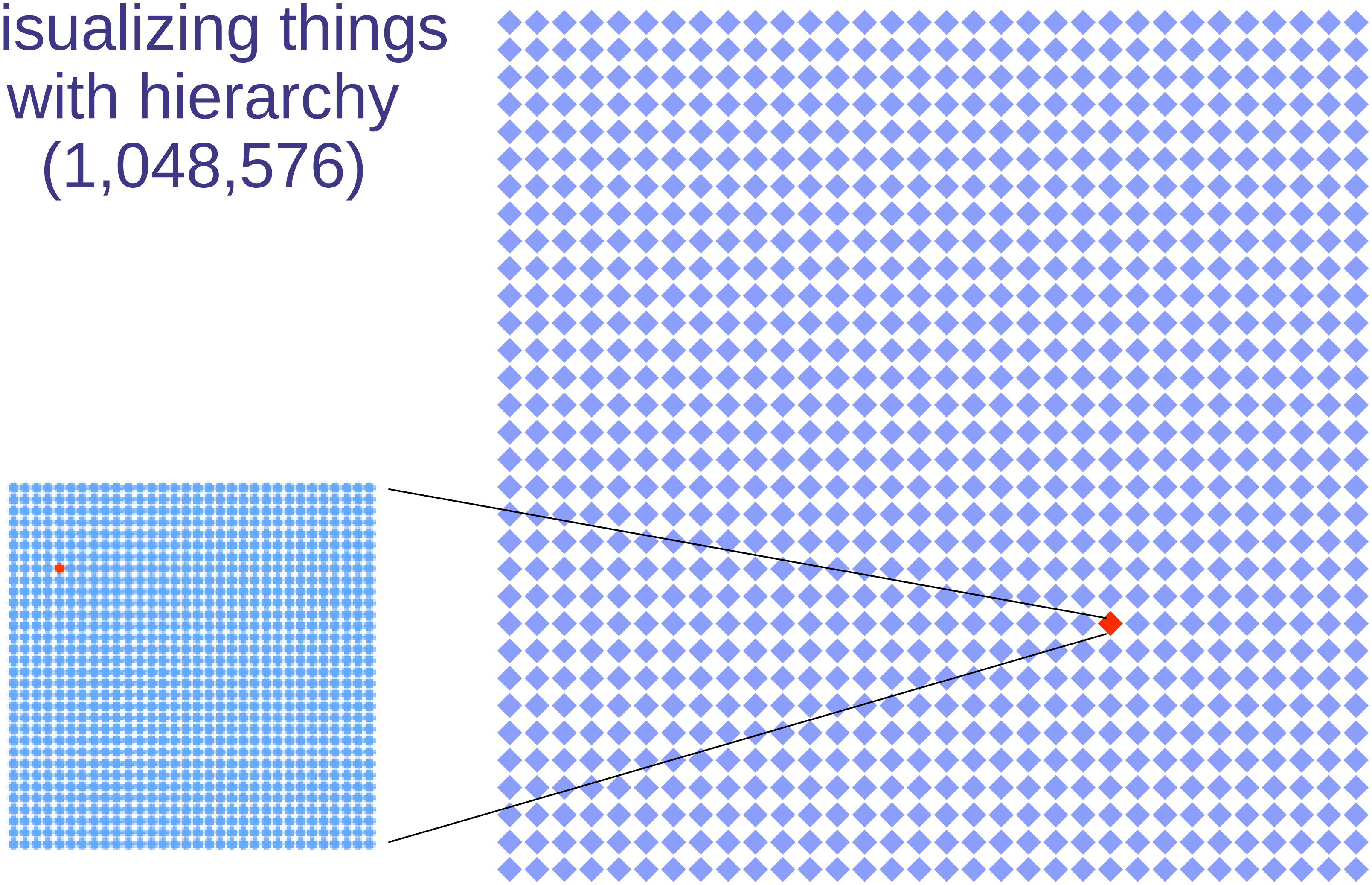


Visualizing things with hierarchy (16,384)



© 2006 by Greg Watson; made available under the EPL v1.0

Visualizing things with hierarchy (1,048,576)



© 2006 by Greg Watson; made available under the EPL v1.0

(not so) Future design issues

- With just two levels of hierarchy, we can handle 1,048,576 “things” relatively easily
- Even a PFlop machine will likely only have ~500,000 processors (4 x 131,072)

BUT, what if each process has 10 threads?

We need to get away from “process-centric” runtime
and debugger views