Having It Both Ways:

Eclipse Parallel Tools Platform (PTP) on Desktop and Cluster

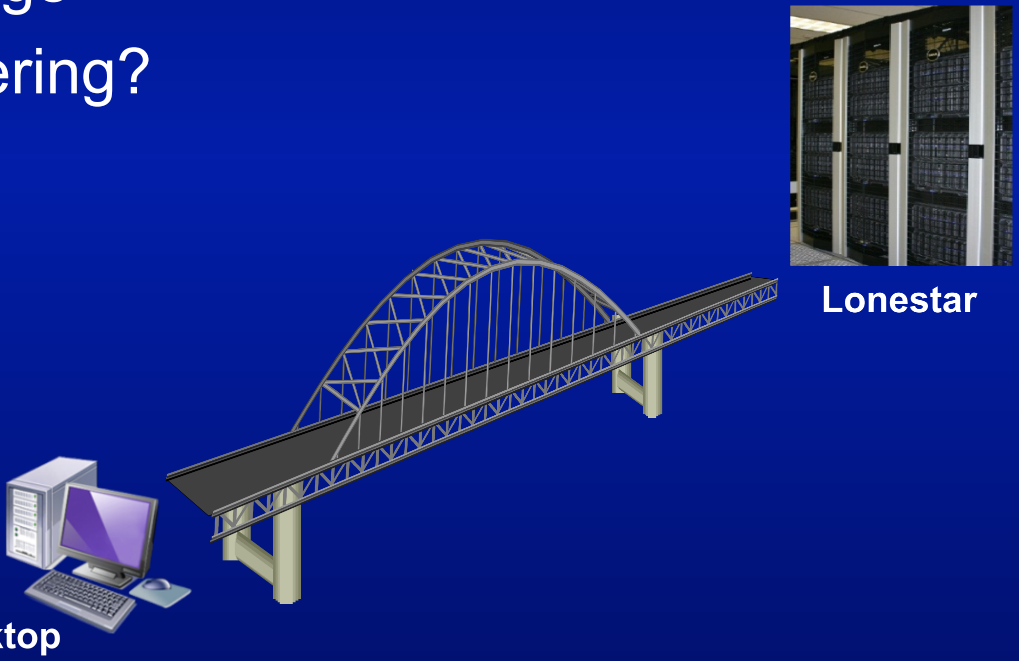
Doug James

Texas Advanced Computing Center

17 Dec 2012

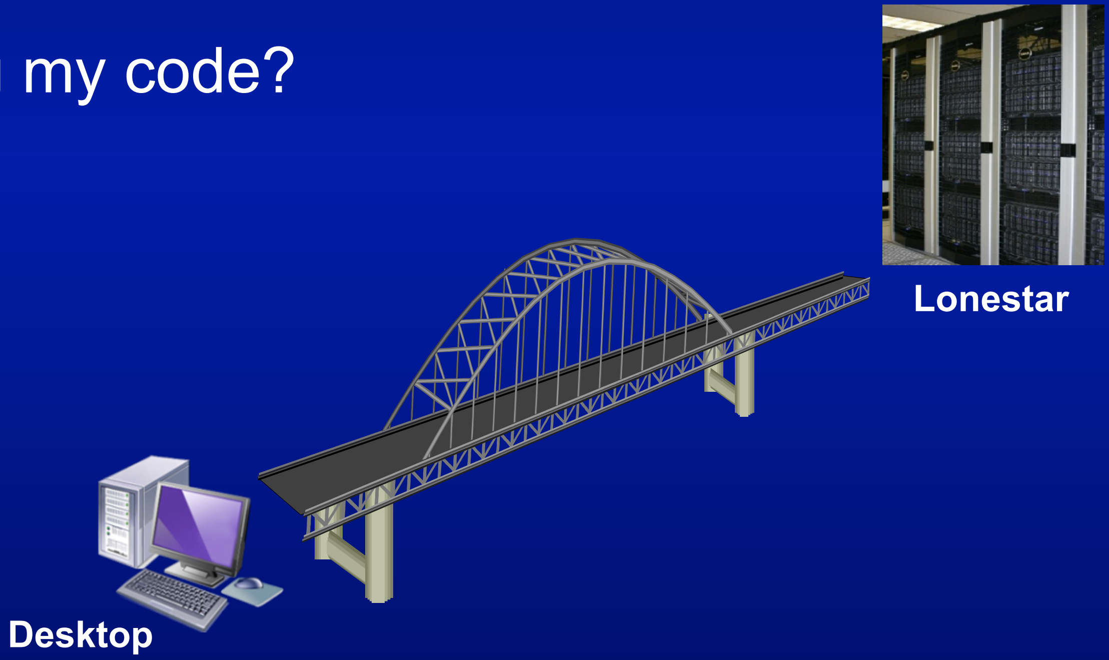
The Workflow Problem

On which side of the bridge should I do my engineering?



The Workflow Problem

Where and how
do I develop/debug my code?

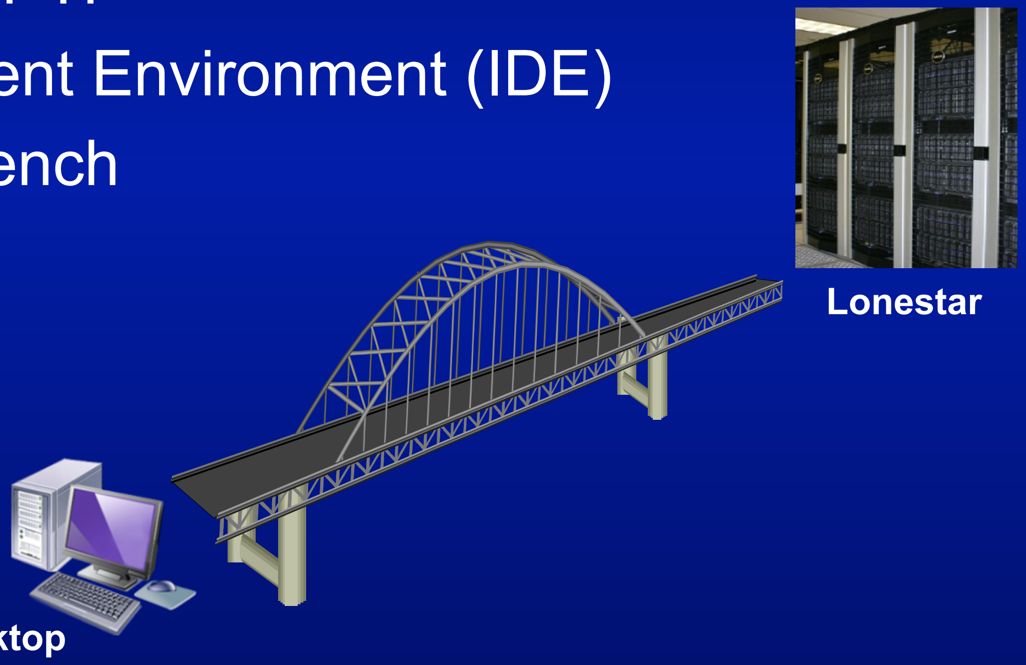


The Workflow Problem

How best to use Eclipse PTP

= Integrated Development Environment (IDE)

= development workbench



Overview

Eclipse PTP

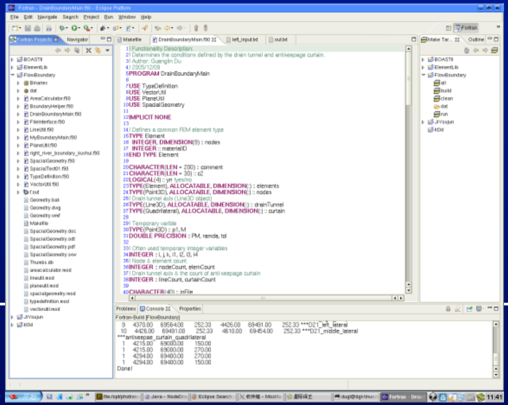
Potential Workflows

Demo: Synchronized Projects

Eclipse

- A framework for language-specific IDEs
- Provides the foundation/hooks/API to build IDEs
- Open source, Java app, multi-platform, generic
- Dozens of language-specific Eclipse IDEs available

www.eclipse.org



```fortran
! Functionality Description:
! Determines the conditions defined by the drain tunnel and antiseepage curtain.
! Author: Guanglin Du
! 2005/12/09
PROGRAM DrainBoundaryMain

USE TypeDefinition
USE VectorUtil
USE PlaneUtil
USE SpacialGeometry

IMPLICIT NONE

! Defines a common FEM element type
TYPE Element
  INTEGER, DIMENSION(9) :: nodes
  INTEGER :: materialID
END TYPE Element

CHARACTER(LEN = 200) :: comment
CHARACTER(LEN = 30) :: c2
LOGICAL(4) :: yn !yes/no
TYPE(Element), ALLOCATABLE, DIMENSION(:) :: elements
TYPE(Point3D), ALLOCATABLE, DIMENSION(:) :: nodes
! Drain tunnel axis (Line3D object)
TYPE(Line3D), ALLOCATABLE, DIMENSION(:) :: drainTunnel
TYPE(Quadrilateral), ALLOCATABLE, DIMENSION(:) :: curtain

! Temporary varible
TYPE(Point3D) :: p1, M
DOUBLE PRECISION :: PM, ramda, tol

! Often used temporary integer variables
INTEGER :: i, j, k, i1, i2, i3, i4
! Node & element count
INTEGER :: nodeCount, elemCount
! Drain tunnel axis & the count of anti-seepage curtain
INTEGER :: lineCount, curtainCount

CHARACTER(40) :: inFile
```

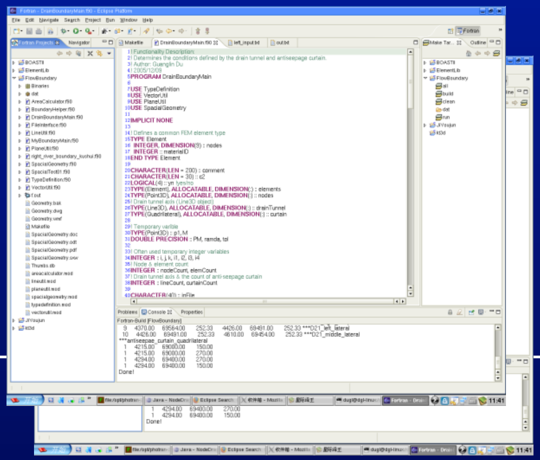
```
Fortran-Build [FlowBoundary]
  9   4370.00   69564.00    252.33    4426.00   69491.00    252.33 ***D21_left_lateral
 10   4426.00   69491.00    252.33    4610.00   69454.00    252.33 ***D21_middle_lateral
***antiseepae_curtain_quadrilateral
  1   4215.00   69000.00    150.00
  1   4215.00   69000.00    270.00
  1   4294.00   69400.00    270.00
  1   4294.00   69400.00    150.00
Done!
```

Eclipse C/C++ Dev Tooling (CDT)

- Mature, robust C/C++ IDE; add Photran to get Fortran
- Everything you'd expect: syntax-aware editing, debugging, code completion, refactoring, version control, team tools…
- Hides makefile details, or supply your own
- You supply compilers (e.g. gcc); CDT integrates them

www.eclipse.org/cdt

www.eclipse.org/photran



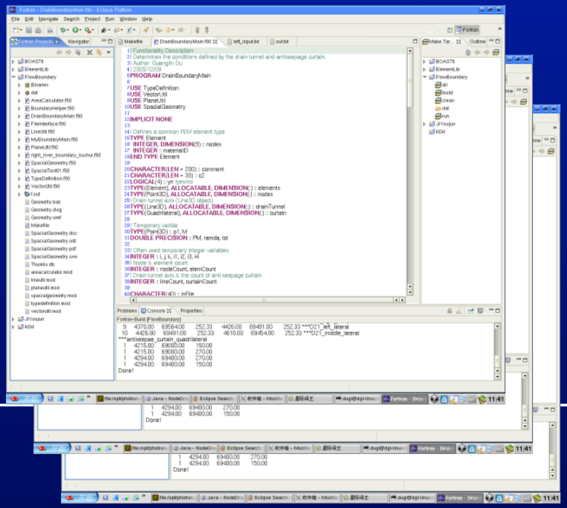
```
! Functionality Description:
! Determines the conditions defined by the drain tunnel and antiseepage curtain.
! Author: Guanglin Du
! 2005/12/09
PROGRAM DrainBoundaryMain

USE TypeDefinition
USE VectorUtil
USE PlaneUtil
USE SpacialGeometry

IMPLICIT NONE

! Defines a common FEM element type
TYPE Element
  INTEGER, DIMENSION(9) :: nodes
  INTEGER :: materialID
END TYPE Element

CHARACTER(LEN = 200) :: comment
CHARACTER(LEN = 30) :: c2
LOGICAL(4) :: yn !yes/no
TYPE(Element), ALLOCATABLE, DIMENSION(:) :: elements
TYPE(Point3D), ALLOCATABLE, DIMENSION(:) :: nodes
! Drain tunnel axis (Line3D object)
TYPE(Line3D), ALLOCATABLE, DIMENSION(:) :: drainTunnel
TYPE(Quadrilateral), ALLOCATABLE, DIMENSION(:) :: curtain

! Temporary varible
TYPE(Point3D) :: p1, M
DOUBLE PRECISION :: PM, ramda, tol

! Often used temporary integer variables
INTEGER :: i, j, k, i1, i2, i3, i4
! Node & element count
INTEGER :: nodeCount, elemCount
! Drain tunnel axis & the count of anti-seepage curtain
INTEGER :: lineCount, curtainCount

CHARACTER(40) :: inFile
```

```
Fortran-Build [FlowBoundary]
  9    4370.00    69564.00     252.33     4426.00    69491.00     252.33 ***D21_left_lateral
 10    4426.00    69491.00     252.33     4610.00    69454.00     252.33 ***D21_middle_lateral
***antiseepae_curtain_quadrilateral
  1    4215.00    69000.00     150.00
  1    4215.00    69000.00     270.00
  1    4294.00    69400.00     270.00
  1    4294.00    69400.00     150.00
Done!
```

Eclipse Parallel Tools Platform (PTP)

- Built on CDT/Photran: supports C/C++ and Fortran
- MPI, OpenMP, and other protocols
- Built-in parallel debugging
- Supports variety of analysis tools (e.g. TAU, gprof, valgrind)
- Ready out of the box: extract archive and run
- Tight integration with XSEDE clusters

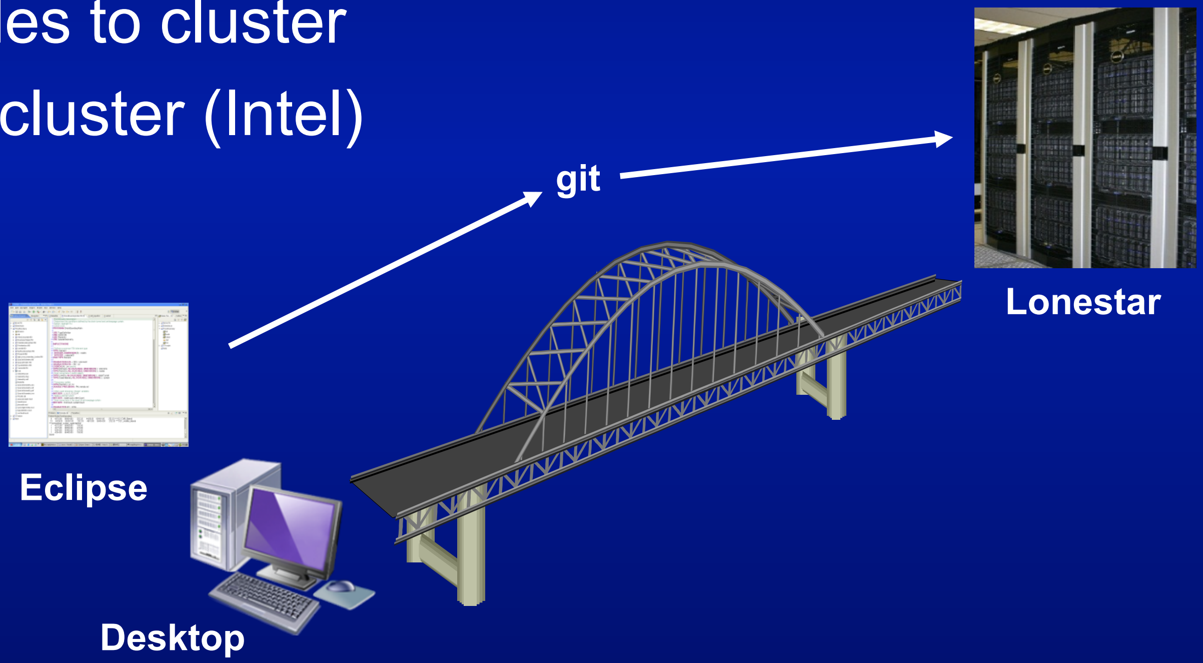
www.eclipse.org/pdt



TACC

Basic Workflow

- Run Eclipse on desktop
- Build/debug on desktop (gnu)
- Use git to push files to cluster
- Build and run on cluster (Intel)



Stampede Changes Everything

```
#ifdef MIC_AWARE  // we've got a MIC-aware compiler
    #define ADD_DECL_TARGET_MIC      __declspec( target(mic:0) )
    #define ADD_SHARED                _Cilk_shared
    #define ADD_OFFLOAD               _Cilk_offload
    #define ADD_OFFLOAD_TO            _Cilk_offload_to(0)
    #define NEW(X)                     new( _Offload_shared_malloc( X )  )
    #define ALIGNED_NEW(X)             new( _Offload_shared_aligned_malloc( X, 8 )  )
    #define MALLOC(X)                 _Offload_shared_malloc( X )
    #define DELETE(X)                 _Offload_shared_free( X )
    #define ALIGNED_MALLOC(X)         _Offload_shared_aligned_malloc( X, 8 )
    #define ALIGNED_DELETE(X)         _Offload_shared_aligned_free( X )
    #define VECTOR(XTYPE)              vector< XTYPE, __offload::shared_allocator< XTYPE > >
#else  // compiler is not MIC-aware
    #define ADD_DECL_TARGET_MIC
    #define ADD_SHARED
    #define ADD_OFFLOAD
    #define ADD_OFFLOAD_TO
    #define NEW(X)                     new
    #define ALIGNED_NEW(X)             new
    #define MALLOC(X)                  malloc( X )
    #define DELETE(X)                  delete X
    #define ALIGNED_MALLOC(X)          malloc( X )
    #define ALIGNED_DELETE(X)          delete( X )
    #define VECTOR(XTYPE)              vector< XTYPE >
#endif
```

mic offload requires code that gcc doesn't understand…

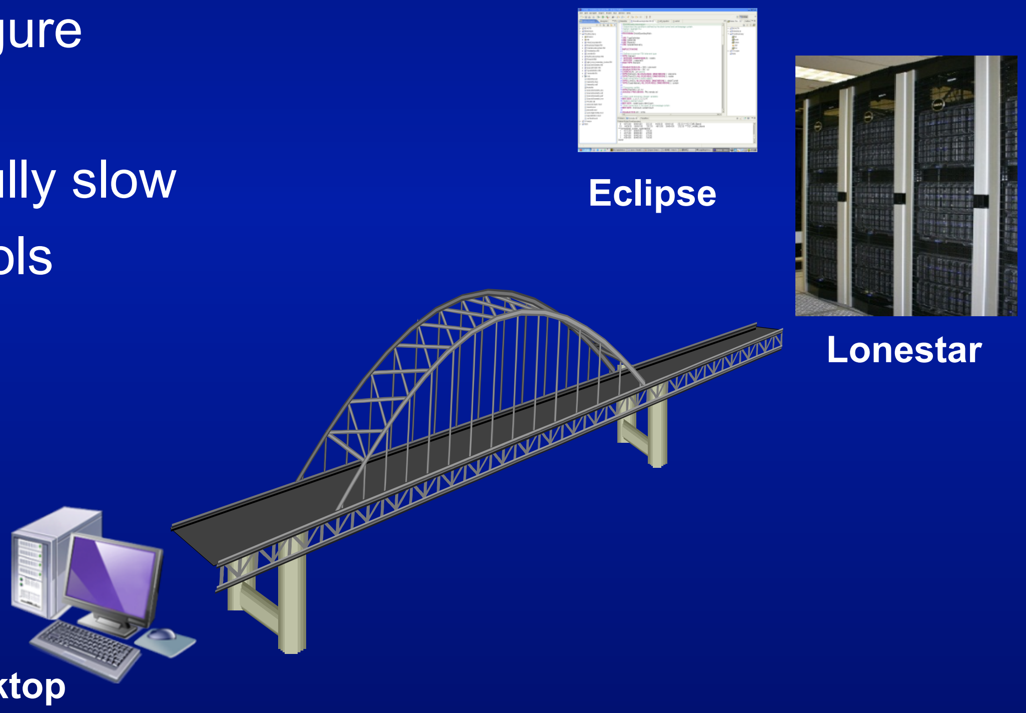
Stampede Changes Everything

```
#ifdef MIC_AWARE
    #include <offload.h>
#endif

typedef VECTOR( double ) ShReadyDoubleVec;
...
ADD_SHARED ShReadyDoubleVec  sadv1;
ADD_SHARED ShReadyDoubleVec* ADD_SHARED pDVec;
ADD_SHARED ShReadyDoubleVec::iterator sd1, sd2;
...
void ADD_SHARED manipByPtrArg( ShReadyDoubleVec ADD_SHARED *pV );
...
pDVec = NEW( sizeof( ShReadyDoubleVec ) ) ADD_SHARED ShReadyDoubleVec(size);
ADD_OFFLOAD manipByPtrArg( pDVec );
```

…gcc portability results in code that’s hard to read

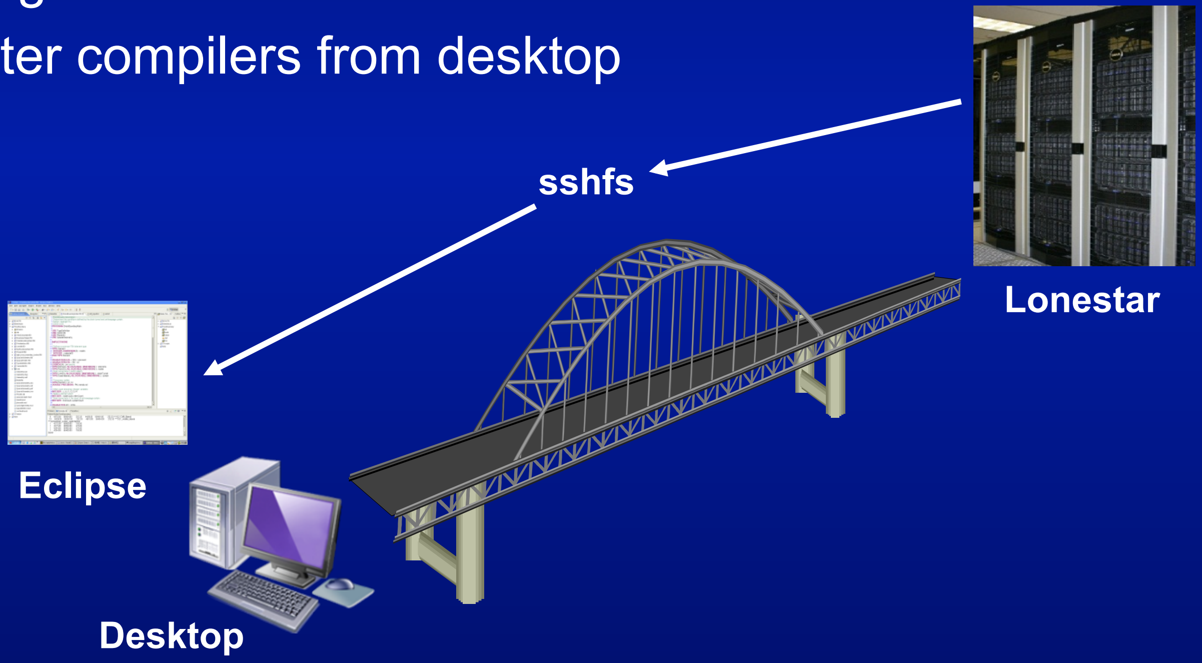
Alternative Workflows

- Run Eclipse on cluster
  - Easy to install and configure
  - All files on cluster
  - X interface can be painfully slow
  - No access to desktop tools



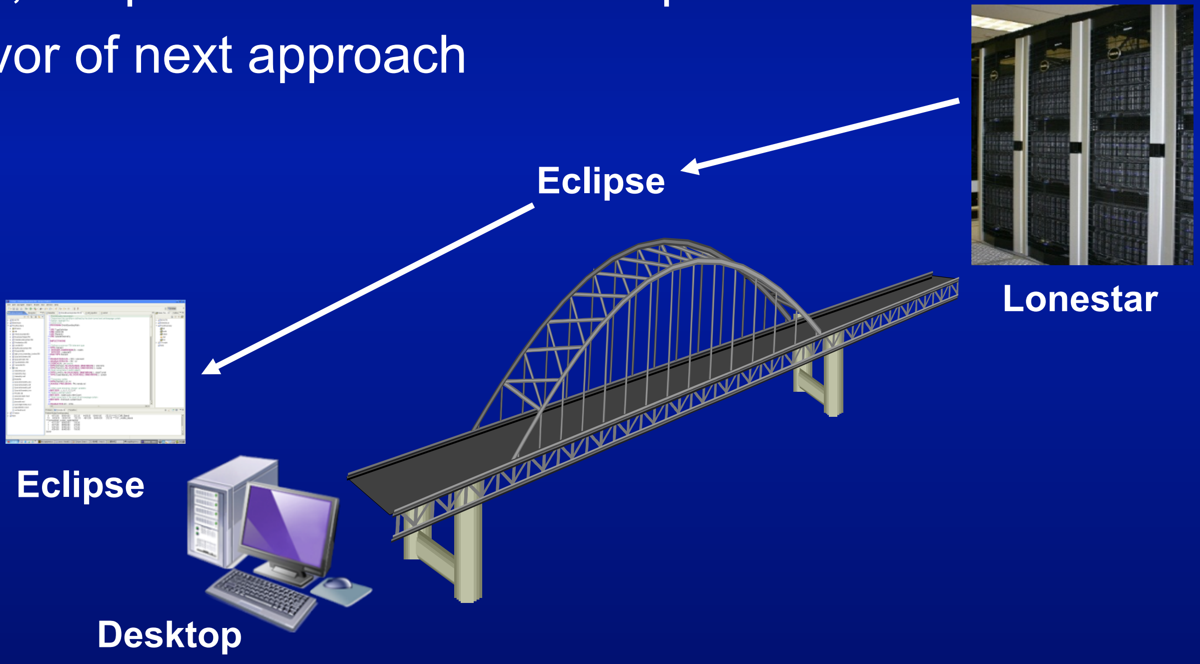
Alternative Workflows

- Mount cluster file system on desktop (sshfs)
  - Very easy to configure
  - Can't exploit cluster compilers from desktop



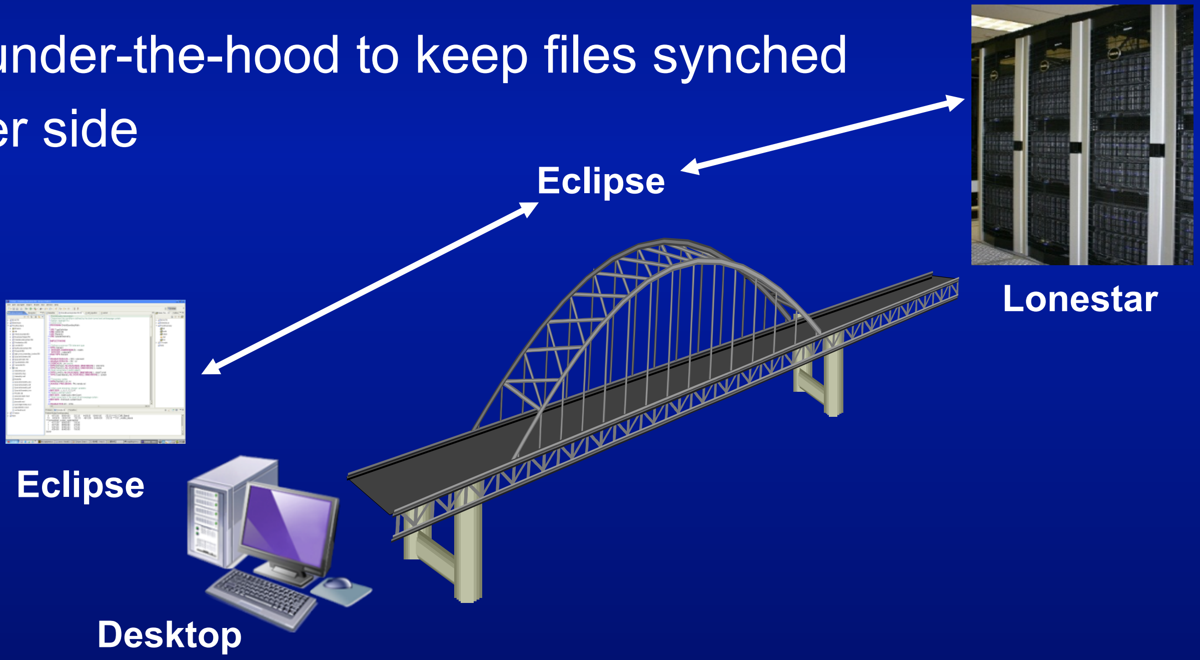
Alternative Workflows

- PTP Remote Project
  - All files on cluster; Eclipse sees files and compilers
  - Deprecated in favor of next approach



Alternative Workflows

- PTP Synchronized Project
  - Files reside on both sides
  - Eclipse uses git under-the-hood to keep files synched
  - Build/run on either side



Conclusions

PTP maturing nicely – I'm impressed

PTP dramatically improves my productivity

But I still need Intel 13 on my desktop

Doug James

djames@tacc.utexas.edu

512-475-9411

www.tacc.utexas.edu