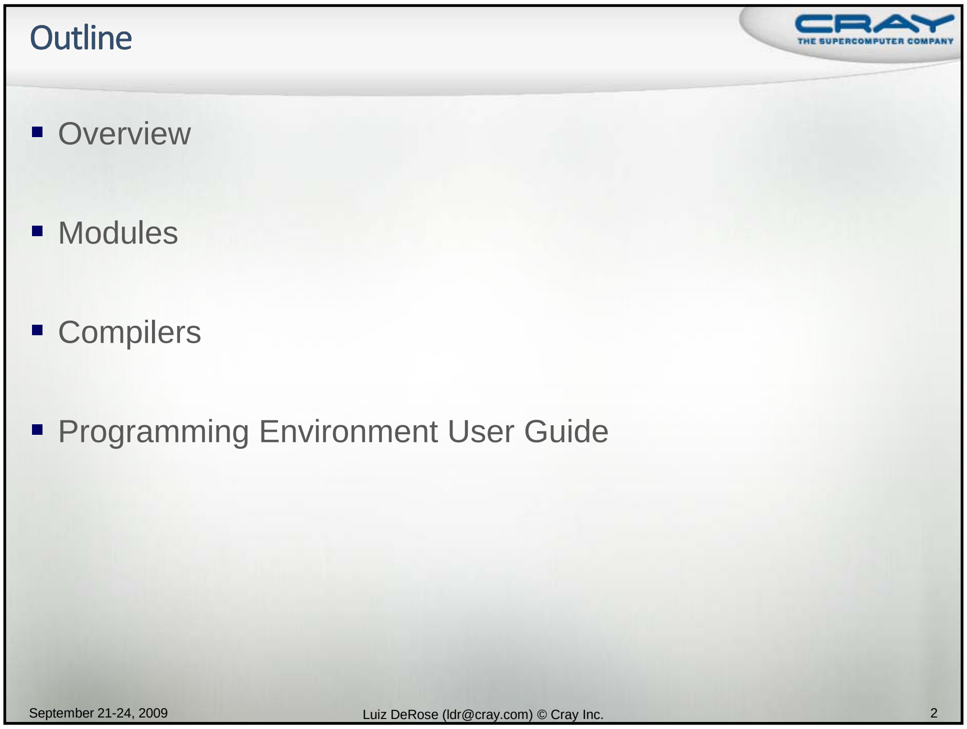
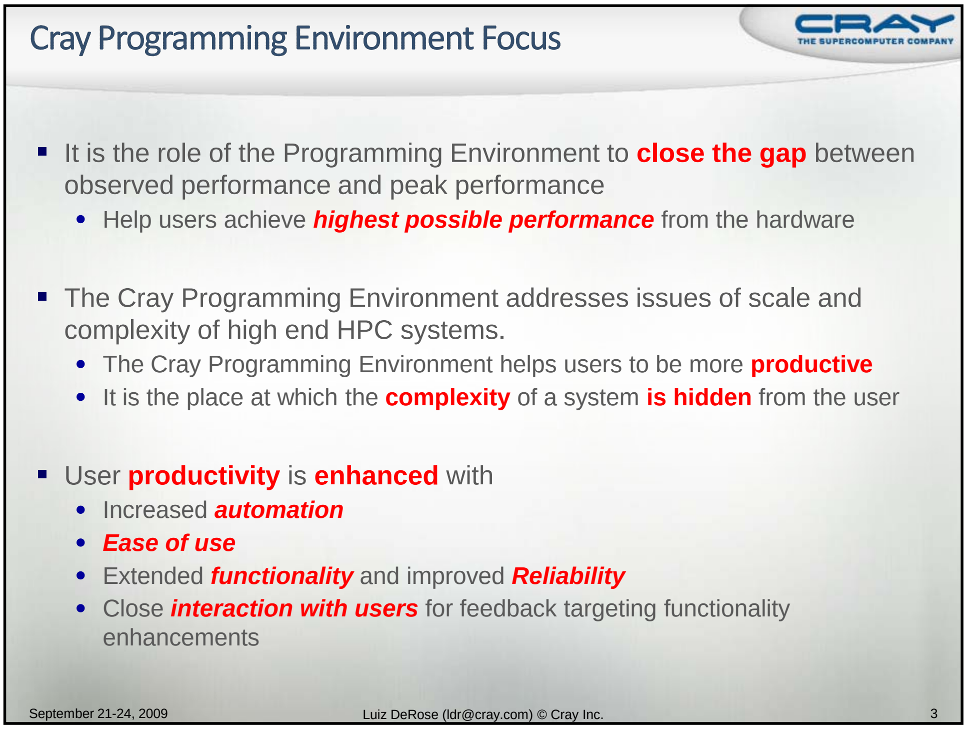
Programming Environment for the Cray XT system

Luiz DeRose
Programming Environments Director
Cray Inc.
ldr@cray.com



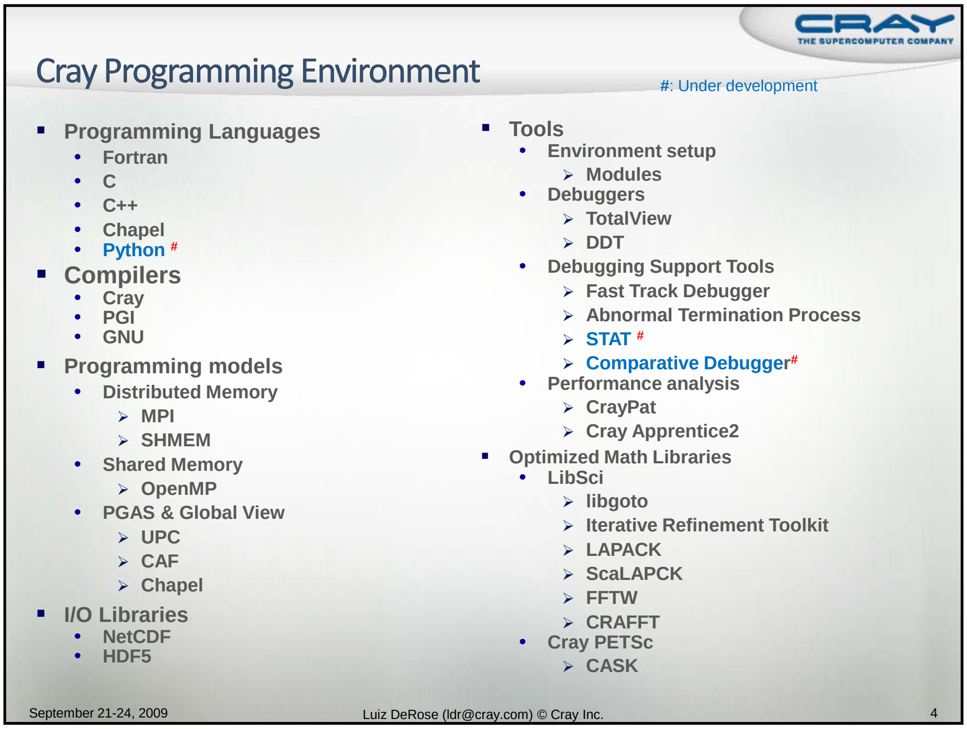
Outline

- Overview
- Modules
- Compilers
- Programming Environment User Guide



Cray Programming Environment Focus

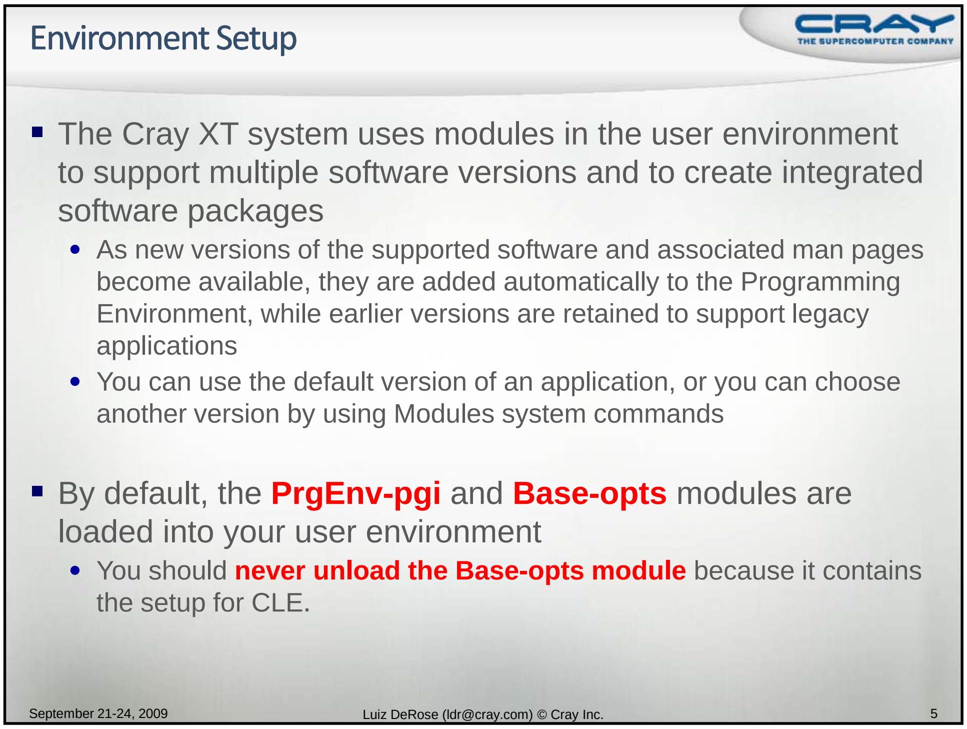
- It is the role of the Programming Environment to **close the gap** between observed performance and peak performance
  - Help users achieve ***highest possible performance*** from the hardware
- The Cray Programming Environment addresses issues of scale and complexity of high end HPC systems.
  - The Cray Programming Environment helps users to be more **productive**
  - It is the place at which the **complexity** of a system **is hidden** from the user
- User **productivity** is **enhanced** with
  - Increased ***automation***
  - ***Ease of use***
  - Extended ***functionality*** and improved ***Reliability***
  - Close ***interaction with users*** for feedback targeting functionality enhancements



Cray Programming Environment

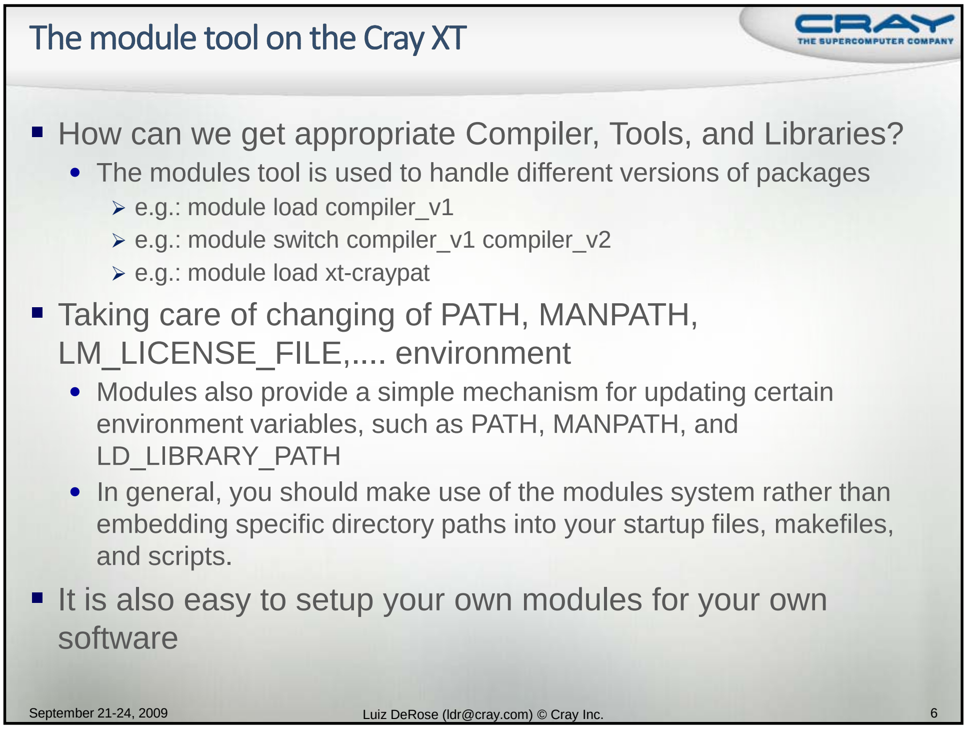
#: Under development

- **Programming Languages**
  - **Fortran**
  - **C**
  - **C++**
  - **Chapel**
  - **Python** #
- **Compilers**
  - **Cray**
  - **PGI**
  - **GNU**
- **Programming models**
  - **Distributed Memory**
    - **MPI**
    - **SHMEM**
  - **Shared Memory**
    - **OpenMP**
  - **PGAS & Global View**
    - **UPC**
    - **CAF**
    - **Chapel**
- **I/O Libraries**
  - **NetCDF**
  - **HDF5**
- **Tools**
  - **Environment setup**
    - **Modules**
  - **Debuggers**
    - **TotalView**
    - **DDT**
  - **Debugging Support Tools**
    - **Fast Track Debugger**
    - **Abnormal Termination Process**
    - **STAT** #
    - **Comparative Debugger**#
  - **Performance analysis**
    - **CrayPat**
    - **Cray Apprentice2**
- **Optimized Math Libraries**
  - **LibSci**
    - **libgoto**
    - **Iterative Refinement Toolkit**
    - **LAPACK**
    - **ScaLAPCK**
    - **FFTW**
    - **CRAFFT**
  - **Cray PETSc**
    - **CASK**



Environment Setup

- The Cray XT system uses modules in the user environment to support multiple software versions and to create integrated software packages
  - As new versions of the supported software and associated man pages become available, they are added automatically to the Programming Environment, while earlier versions are retained to support legacy applications
  - You can use the default version of an application, or you can choose another version by using Modules system commands
- By default, the **PrgEnv-pgi** and **Base-opts** modules are loaded into your user environment
  - You should **never unload the Base-opts module** because it contains the setup for CLE.



The module tool on the Cray XT

- How can we get appropriate Compiler, Tools, and Libraries?
  - The modules tool is used to handle different versions of packages
    - e.g.: module load compiler_v1
    - e.g.: module switch compiler_v1 compiler_v2
    - e.g.: module load xt-craypat
- Taking care of changing of PATH, MANPATH, LM_LICENSE_FILE,.... environment
  - Modules also provide a simple mechanism for updating certain environment variables, such as PATH, MANPATH, and LD_LIBRARY_PATH
  - In general, you should make use of the modules system rather than embedding specific directory paths into your startup files, makefiles, and scripts.
- It is also easy to setup your own modules for your own software

Useful module commands

- Load software
  - module load xt-craypat
- Change software version
  - module swap pgi/7.0.4 pgi/6.1.6
- Load Cray Compiling Environment
  - module swap PrgEnv-pgi PrgEnv-cray

module list

```
users/ldr> module list
Currently Loaded Modulefiles:
  1) modules/3.1.6                10) xt-totalview/8.6.0
  2) xtpe-target-cnl              11) xt-libsci/10.3.9
  3) xt-service/2.2.41            12) xt-mpt/3.4.2
  4) xt-os/2.2.41                 13) xt-pe/2.2.41
  5) xt-boot/2.2.41               14) xt-asyncpe/3.3
  6) xt-lustre-ss/2.2.41_1.6.5    15) PrgEnv-pgi/2.2.41
  7) Base-opts/2.2.41             16) xt-craypat/5.0.0
  8) pgi/9.0.3                    17) apprentice2/5.0.0
  9) totalview-support/1.0.6
```

module show

```
users/ldr> module show cce
-------------------------------------------------------------------
/opt/modulefiles/cce/7.1.3:

setenv           CRAYLMD_LICENSE_FILE /opt/cray/cce/cce.lic
setenv           CRAY_BINUTILS_ROOT /opt/cray/cce/7.1.3/cray-binutils
setenv           CRAY_BINUTILS_VERSION /opt/cray/cce/7.1.3
setenv           CRAY_BINUTILS_BIN /opt/cray/cce/7.1.3/cray-binutils/x86_64-unknown-linux-gnu/bin
setenv           LINKER_X86_64 /opt/cray/cce/7.1.3/cray-binutils/x86_64-unknown-linux-gnu/bin/ld
setenv           ASSEMBLER_X86_64 /opt/cray/cce/7.1.3/cray-binutils/x86_64-unknown-linux-gnu/bin/as
setenv           CRAYLIBS_X86_64 /opt/cray/cce/7.1.3/craylibs/x86-64
setenv           FTN_X86_64 /opt/cray/cce/7.1.3/cftn/x86-64
setenv           CC_X86_64 /opt/cray/cce/7.1.3/CC/x86-64
setenv           CRAY_FTN_VERSION 7.1.3
setenv           CRAY_CC_VERSION 7.1.3
setenv           PE_LEVEL 7.1
prepend-path     FORTRAN_SYSTEM_MODULE_NAMES ftn_lib_definitions
prepend-path     MANPATH
/opt/cray/cce/7.1.3/man:/opt/cray/cce/7.1.3/craylibs/man:/opt/cray/cce/7.1.3/CC/man:/opt/cray/cce/7.
1.3/cftn/man
prepend-path     NLSPATH /opt/cray/cce/7.1.3/CC/x86-
64/nls/En/%N.cat:/opt/cray/cce/7.1.3/craylibs/x86-64/nls/En/%N.cat:/opt/cray/cce/7.1.3/cftn/x86-
64/nls/En/%N.cat
prepend-path     INCLUDE_PATH_X86_64 /opt/cray/cce/7.1.3/craylibs/x86-64/include
prepend-path     PATH /opt/cray/cce/7.1.3/cray-binutils/x86_64-unknown-linux-
gnu/bin:/opt/cray/cce/7.1.3/craylibs/x86-
64/bin:/opt/cray/cce/7.1.3/cftn/bin:/opt/cray/cce/7.1.3/CC/bin
append-path      MANPATH /usr/share/man
-------------------------------------------------------------------
```

module avail

```
(ldr@hawk) 105% module avail
------------------ /opt/cray/xt-asyncpe/3.3.11/modulefiles -------------------
xtpe-barcelona       xtpe-network-seastar xtpe-target-native
xtpe-istanbul        xtpe-quadcore
xtpe-network-gemini  xtpe-shanghai

------------------ /opt/totalview-support/1.0.6/modulefiles ------------------
xt-totalview-mem-debug
------------------------------ /opt/modulefiles ------------------------------
Base-opts/2.0.48.lusrelsave          pathscale/3.2(default)
Base-opts/2.0.49.lusrelsave          pathscale/3.2.orig
Base-opts/2.0.54.lusrelsave          pbs/8.1
Base-opts/2.0.58.lusrelsave          pbs/default
Base-opts/2.0.61.lusrelsave          petsc/3.0.0.1
Base-opts/2.0.62.lusrelsave          petsc/3.0.0.2
Base-opts/2.0.63.lusrelsave          petsc/3.0.0.3
Base-opts/2.1.41HD.lusrelsave        petsc/3.0.0.3.1
Base-opts/2.1.50HD                   petsc/3.0.0.4
Base-opts/2.1.50HD.lusrelsave        petsc/3.0.0.4.2
Base-opts/2.2.24                     petsc/3.0.0.5(default)
Base-opts/2.2.24.lusrelsave          petsc-complex/3.0.0.1
Base-opts/2.2.29                     petsc-complex/3.0.0.2
Base-opts/2.2.29.lusrelsave          petsc-complex/3.0.0.3
Base-opts/2.2.31                     petsc-complex/3.0.0.3.1
Base-opts/2.2.31.lusrelsave          petsc-complex/3.0.0.4
Base-opts/2.2.32DSL3                 petsc-complex/3.0.0.4.2
Base-opts/2.2.32DSL3.lusrelsave      petsc-complex/3.0.0.5(default)
Base-opts/2.2.41(default)            pgi/7.1.6
Base-opts/2.2.41.lusrelsave          pgi/7.2.4
PrgEnv-cray/1.0.0(default)           pgi/7.2.5
PrgEnv-gnu/2.1.50HD                  pgi/8.0.1
PrgEnv-gnu/2.2.24                    pgi/8.0.2
PrgEnv-gnu/2.2.29                    pgi/8.0.3

. . .
```

Release Notes

```
(ldr@hawk) 112% module help xt-craypat

----------- Module Specific Help for 'xt-craypat/5.0.0' -----------
===================================================================
CrayPat  5.0.0
==============
Release Date: August 20, 2009

Purpose:
--------
Differences:
------------
Changes from CrayPat 4.4.1 release to 5.0.0 release
===================================================
CrayPat 4.4.1 release revision: 2380
CrayPat 5.0.0 release revision: 2686
---------------------------------------------------------

CrayPat
-------
modulefile      -DCRAYPAT is added to compilation options when
                the xt-craypat module is loaded. (Add -UCRAYPAT
                to a compiler invocation will undefine the macro.)
pat_hwpc        removed - no longer supported
pat_build       more complete evaluation of DWARF DIEs
pat_build       no longer control tracing symbols in file by write permissions
pat_build       add 'trace-file' directive to control tracing symbols in file

. . .

Bugz closed since 4.4.1 release
-------------------------------
(following fixed in revs <=2686 == 5.0 release)

. . .
```

Setting Your Target Architecture

- Before you begin to compile programs, you must verify that the target architecture is set correctly
- The compilers and linker use the target architecture in creating executables to run on compute nodes
- The target architecture is set automatically when you log in
  - The xtpe-target-cnl module should be loaded and the XTPE_COMPILE_TARGET environment variable set to linux

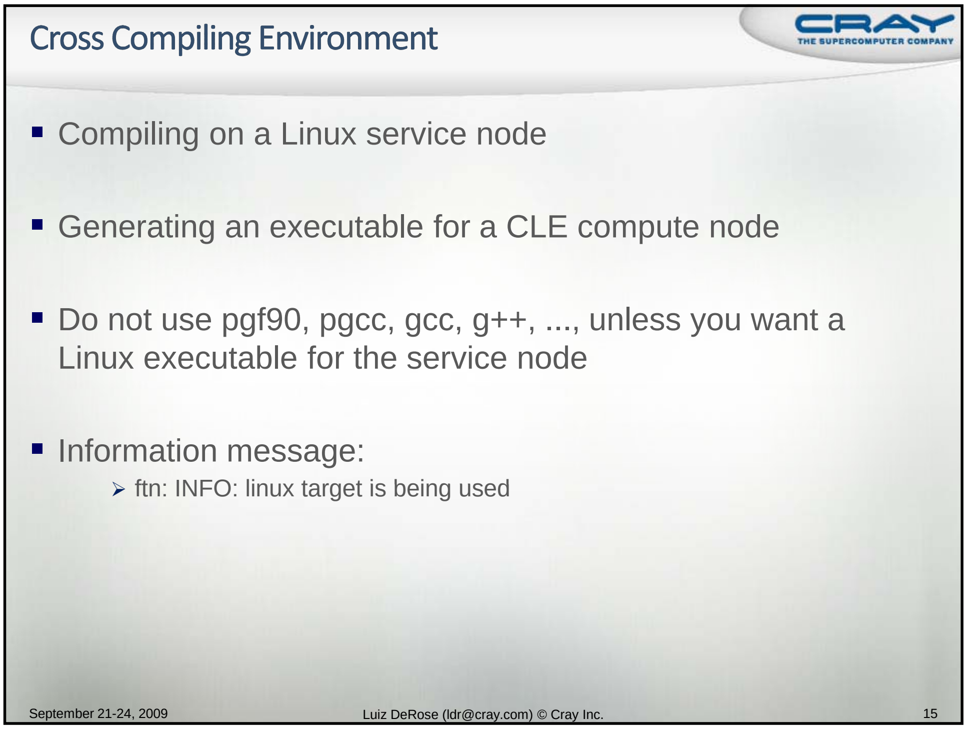
```
users/ldr> module show xtpe-target-cnl
-------------------------------------------------------------------
/opt/modulefiles/xtpe-target-cnl:

conflict       xtpe-target-catamount
conflict       x2pe-target-x2
setenv         XTPE_COMPILE_TARGET linux
-------------------------------------------------------------------
```

Using the Compiler Driver Commands

- You use compiler driver commands to launch all Cray XT compilers
- The syntax for the compiler driver is:
  - cc | CC | ftn [Cray_options | PGI_options | GNU_options] files [-lhugetlbfs]
- For example, to use the PGI Fortran compiler to compile prog1.f90
  - First use the **module list** command to verify that these modules have been loaded:
    - PrgEnv-pgi
    - xtpe-target-cnl
  - Then use this command:
    - % ftn prog1.f90

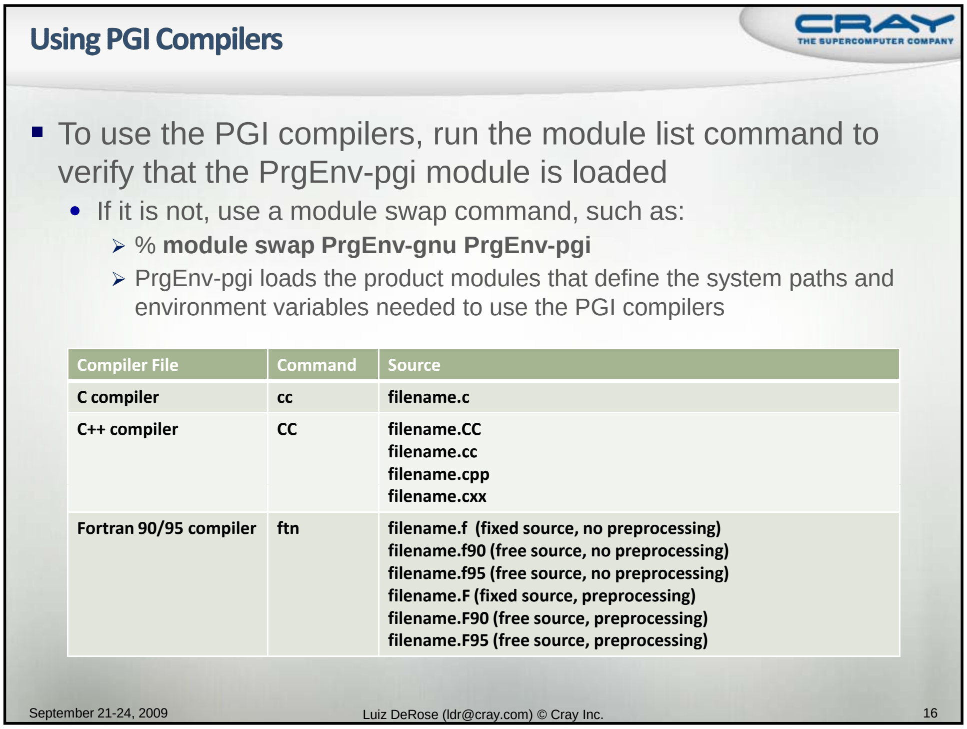
Compiler man Pages

- The cc(1), CC(1), and ftn(1) man pages contain information about the compiler driver commands
- The pgcc(1), pgCC(1), and pgf95(1) man pages contain descriptions of the PGI compiler command options
- The *craycc(1), crayCC(1), and crayftn(1) man pages contain descriptions of the Cray compiler command options*
- The gcc(1), g++(1), and gfortran(1) man pages contain descriptions of the GNU compiler command options
- To verify that you are using the correct version of a compiler, use:
  - -V option on a cc, CC, or ftn command with PGI and CCE
  - --version option on a cc, CC, or ftn command with GNU



Cross Compiling Environment

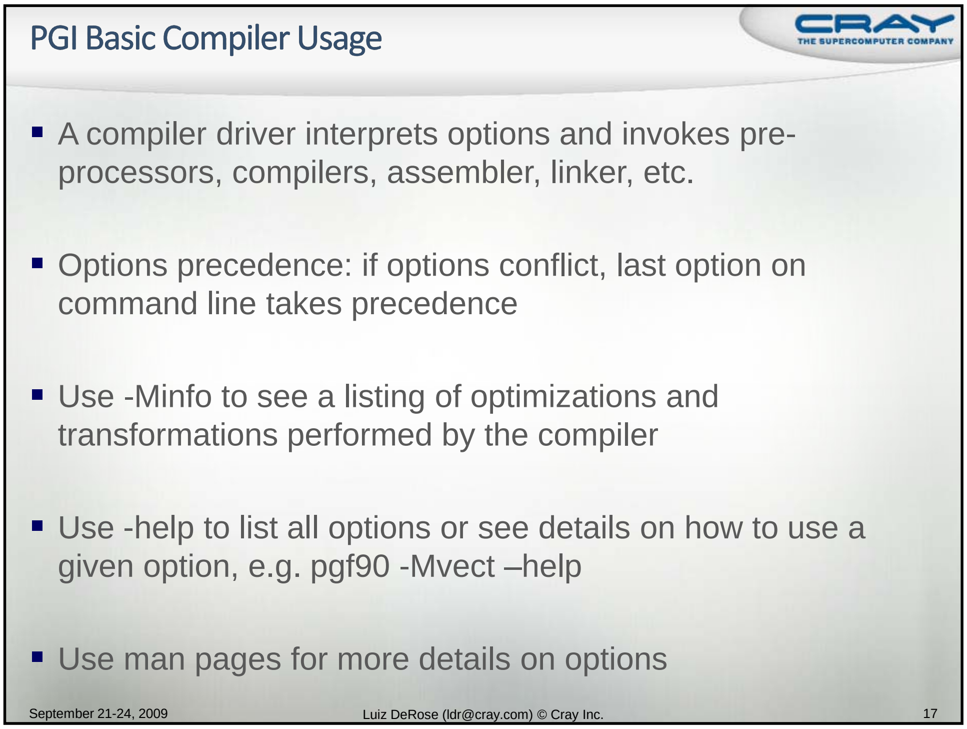
- Compiling on a Linux service node
- Generating an executable for a CLE compute node
- Do not use pgf90, pgcc, gcc, g++, ..., unless you want a Linux executable for the service node
- Information message:
  - ftn: INFO: linux target is being used



Using PGI Compilers

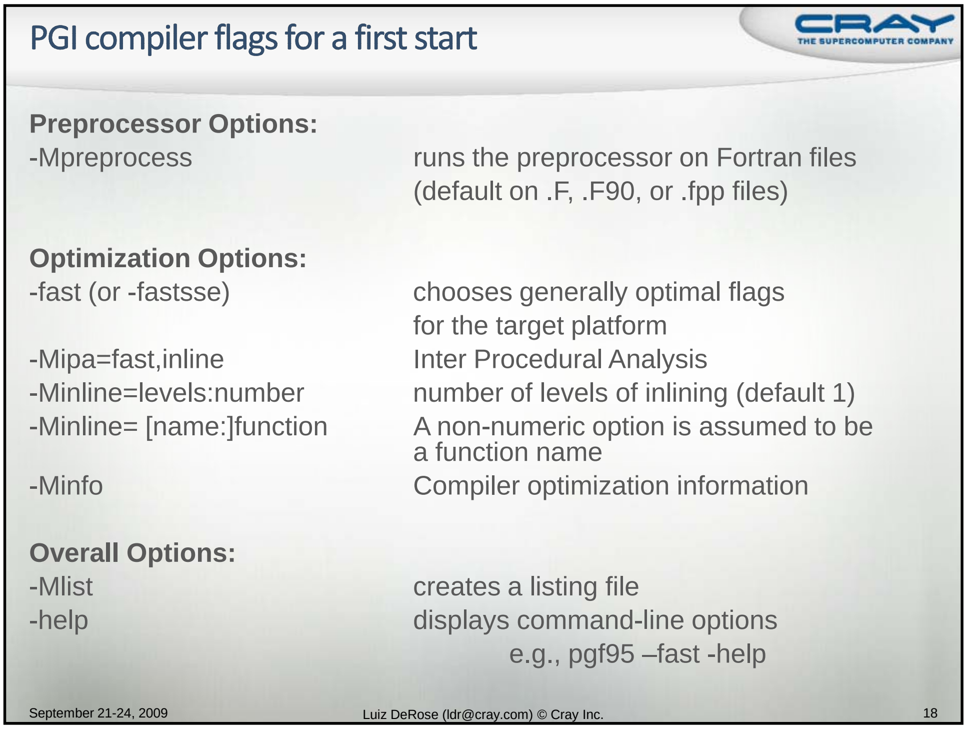
- To use the PGI compilers, run the module list command to verify that the PrgEnv-pgi module is loaded
  - If it is not, use a module swap command, such as:
    - % **module swap PrgEnv-gnu PrgEnv-pgi**
    - PrgEnv-pgi loads the product modules that define the system paths and environment variables needed to use the PGI compilers

Compiler File	Command	Source
C compiler	cc	filename.c
C++ compiler	CC	filename.CC filename.cc filename.cpp filename.cxx
Fortran 90/95 compiler	ftn	filename.f (fixed source, no preprocessing) filename.f90 (free source, no preprocessing) filename.f95 (free source, no preprocessing) filename.F (fixed source, preprocessing) filename.F90 (free source, preprocessing) filename.F95 (free source, preprocessing)



PGI Basic Compiler Usage

- A compiler driver interprets options and invokes pre-processors, compilers, assembler, linker, etc.
- Options precedence: if options conflict, last option on command line takes precedence
- Use -Minfo to see a listing of optimizations and transformations performed by the compiler
- Use -help to list all options or see details on how to use a given option, e.g. pgf90 -Mvect –help
- Use man pages for more details on options



PGI compiler flags for a first start

Preprocessor Options:

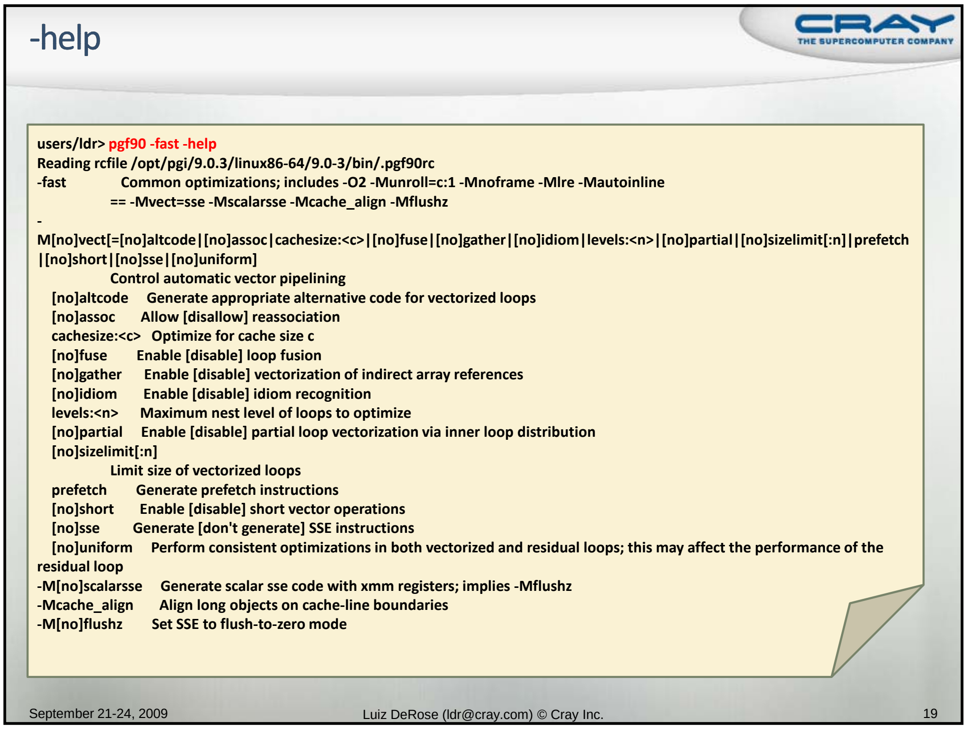
-Mpreprocess	runs the preprocessor on Fortran files (default on .F, .F90, or .fpp files)

Optimization Options:

-fast (or -fastsse)	chooses generally optimal flags for the target platform
-Mipa=fast,inline	Inter Procedural Analysis
-Minline=levels:number	number of levels of inlining (default 1)
-Minline= [name:]function	A non-numeric option is assumed to be a function name
-Minfo	Compiler optimization information

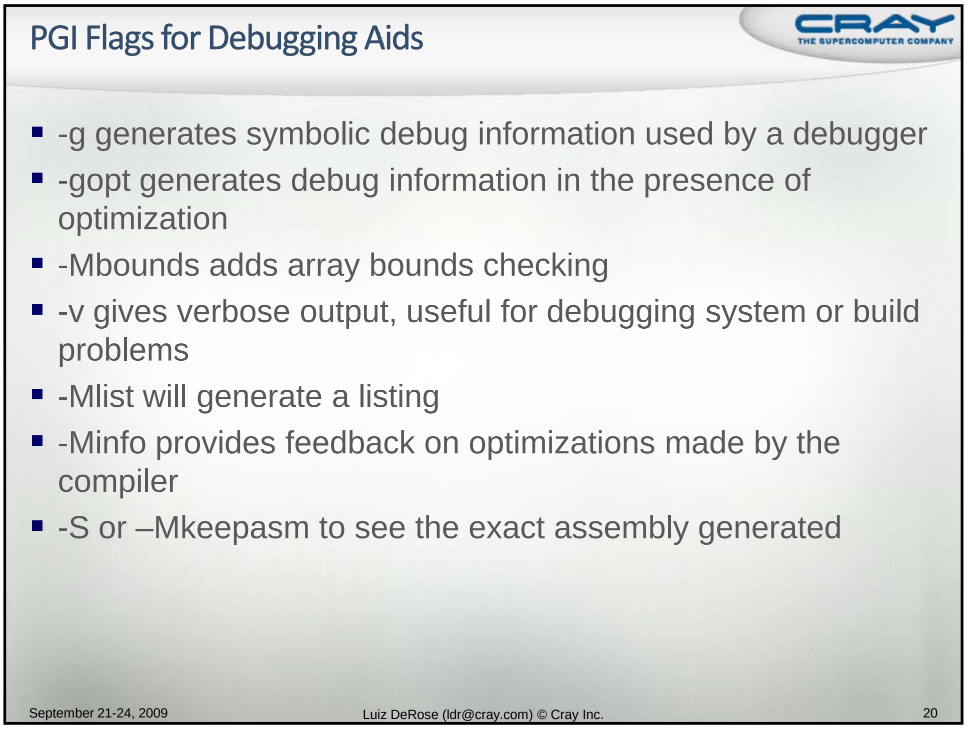
Overall Options:

-Mlist	creates a listing file
-help	displays command-line options e.g., pgf95 –fast -help



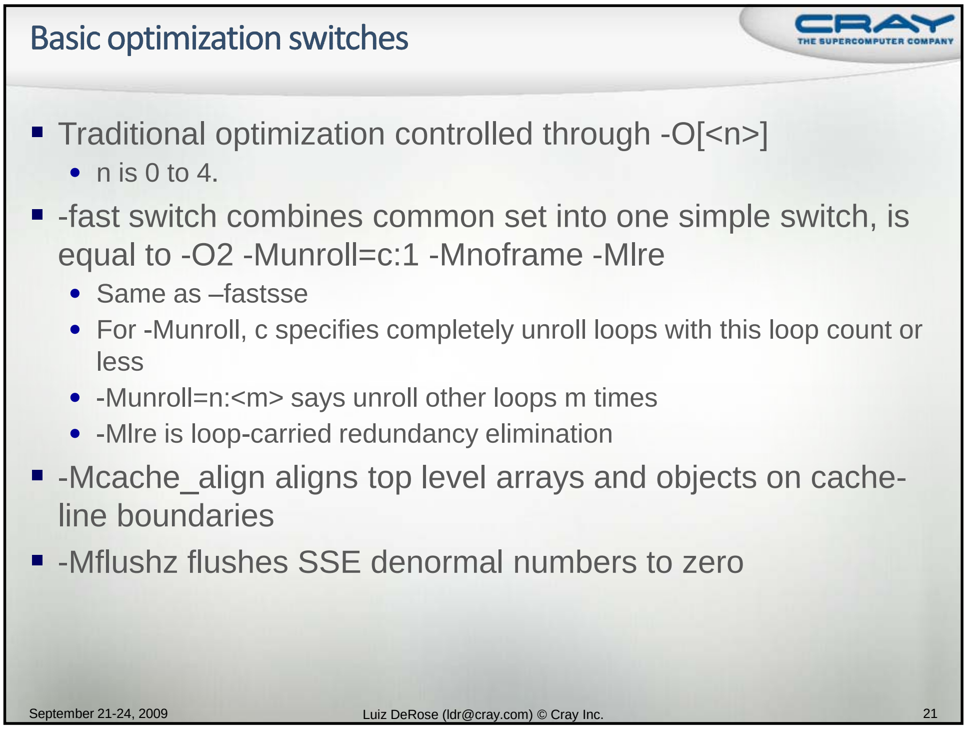
-help

```
users/ldr> pgf90 -fast -help
Reading rcfile /opt/pgi/9.0.3/linux86-64/9.0-3/bin/.pgf90rc
-fast           Common optimizations; includes -O2 -Munroll=c:1 -Mnoframe -Mlre -Mautoinline
                == -Mvect=sse -Mscalarsse -Mcache_align -Mflushz
-
M[no]vect[=[no]altcode|[no]assoc|cachesize:<c>|[no]fuse|[no]gather|[no]idiom|levels:<n>|[no]partial|[no]sizelimit[:n]|prefetch
|[no]short|[no]sse|[no]uniform]
                Control automatic vector pipelining
   [no]altcode     Generate appropriate alternative code for vectorized loops
   [no]assoc      Allow [disallow] reassociation
   cachesize:<c>   Optimize for cache size c
   [no]fuse       Enable [disable] loop fusion
   [no]gather      Enable [disable] vectorization of indirect array references
   [no]idiom      Enable [disable] idiom recognition
   levels:<n>     Maximum nest level of loops to optimize
   [no]partial     Enable [disable] partial loop vectorization via inner loop distribution
   [no]sizelimit[:n]
                Limit size of vectorized loops
   prefetch       Generate prefetch instructions
   [no]short      Enable [disable] short vector operations
   [no]sse       Generate [don't generate] SSE instructions
   [no]uniform     Perform consistent optimizations in both vectorized and residual loops; this may affect the performance of the
residual loop
-M[no]scalarsse    Generate scalar sse code with xmm registers; implies -Mflushz
-Mcache_align      Align long objects on cache-line boundaries
-M[no]flushz      Set SSE to flush-to-zero mode
```



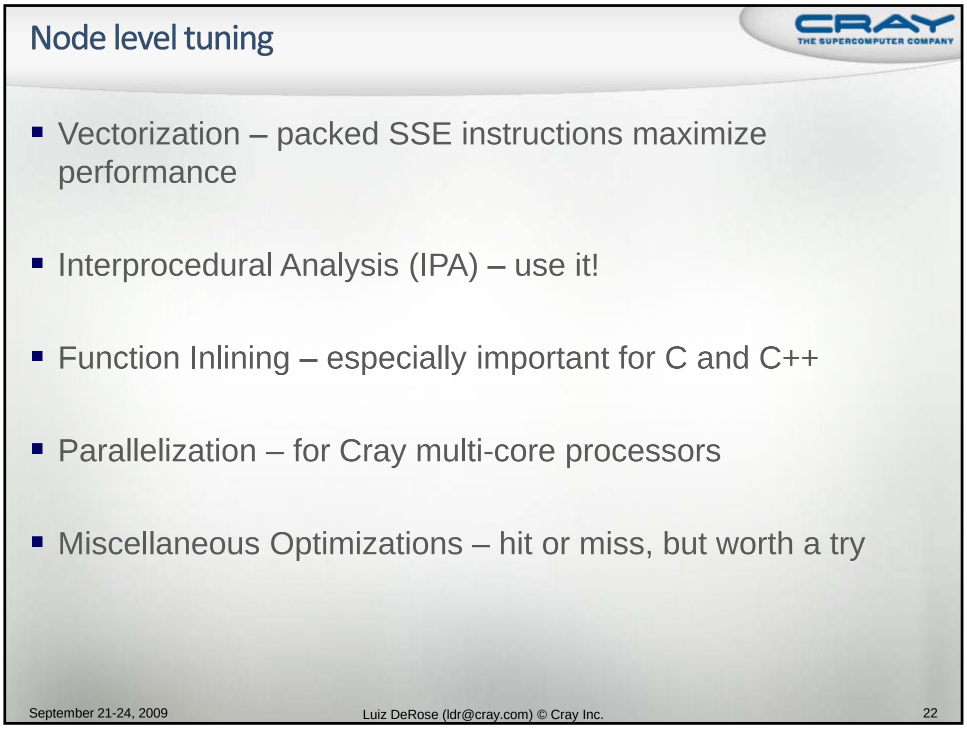
PGI Flags for Debugging Aids

- -g generates symbolic debug information used by a debugger
- -gopt generates debug information in the presence of optimization
- -Mbounds adds array bounds checking
- -v gives verbose output, useful for debugging system or build problems
- -Mlist will generate a listing
- -Minfo provides feedback on optimizations made by the compiler
- -S or –Mkeepasm to see the exact assembly generated



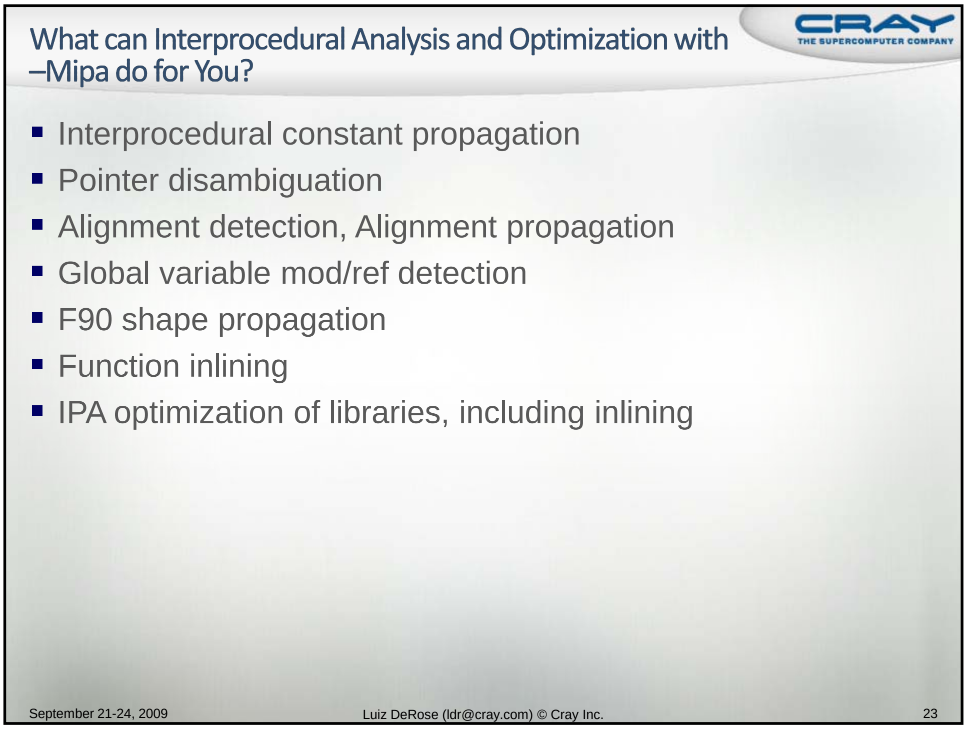
Basic optimization switches

- Traditional optimization controlled through -O[<n>]
  - n is 0 to 4.
- -fast switch combines common set into one simple switch, is equal to -O2 -Munroll=c:1 -Mnoframe -Mlre
  - Same as –fastsse
  - For -Munroll, c specifies completely unroll loops with this loop count or less
  - -Munroll=n:<m> says unroll other loops m times
  - -Mlre is loop-carried redundancy elimination
- -Mcache_align aligns top level arrays and objects on cache-line boundaries
- -Mflushz flushes SSE denormal numbers to zero



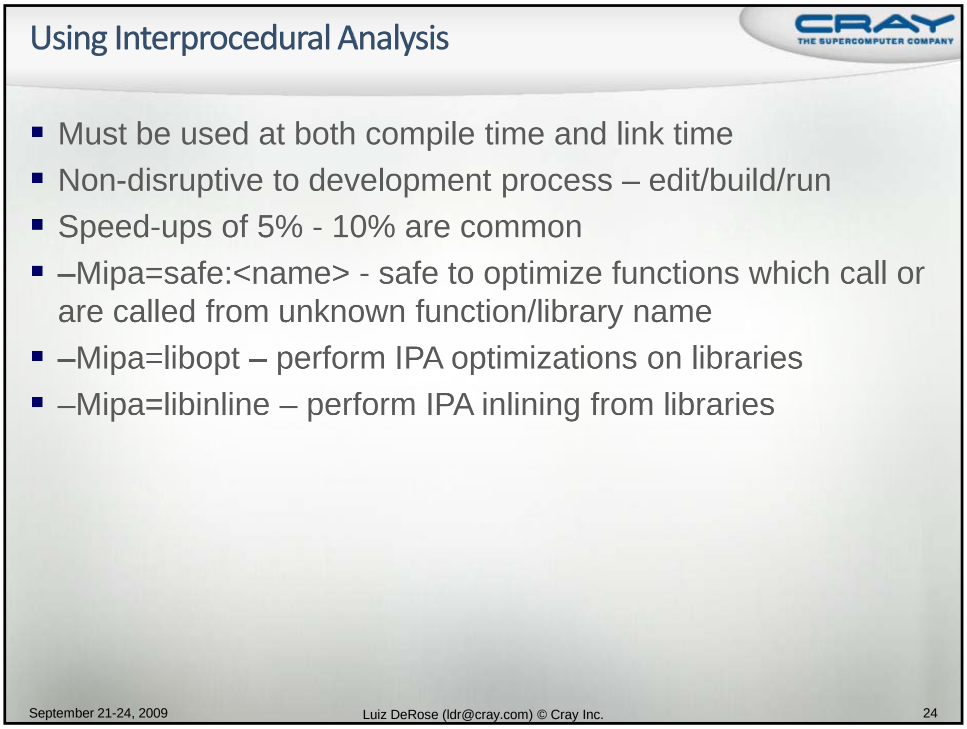
Node level tuning

- Vectorization – packed SSE instructions maximize performance
- Interprocedural Analysis (IPA) – use it!
- Function Inlining – especially important for C and C++
- Parallelization – for Cray multi-core processors
- Miscellaneous Optimizations – hit or miss, but worth a try



What can Interprocedural Analysis and Optimization with –Mipa do for You?

- Interprocedural constant propagation
- Pointer disambiguation
- Alignment detection, Alignment propagation
- Global variable mod/ref detection
- F90 shape propagation
- Function inlining
- IPA optimization of libraries, including inlining



Using Interprocedural Analysis

- Must be used at both compile time and link time
- Non-disruptive to development process – edit/build/run
- Speed-ups of 5% - 10% are common
- –Mipa=safe:<name> - safe to optimize functions which call or are called from unknown function/library name
- –Mipa=libopt – perform IPA optimizations on libraries
- –Mipa=libinline – perform IPA inlining from libraries

–Minline[=[lib:]<inlib> | [name:]<func> | except:<func> | size:<n> | levels:<n>]

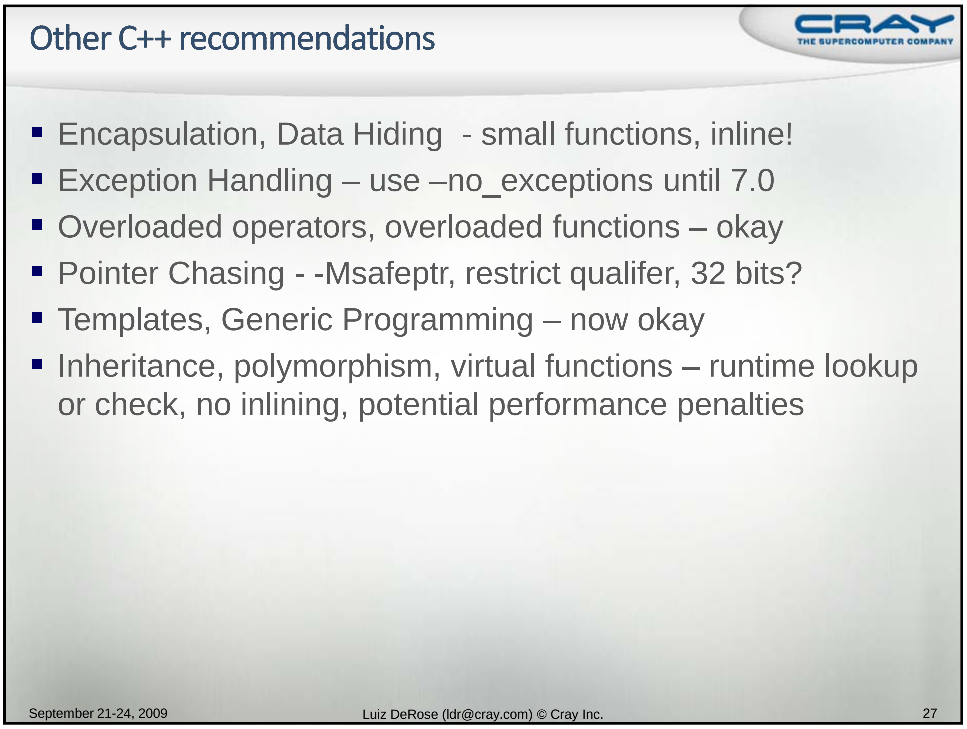
[lib:]<inlib>	**Inline extracted functions from *inlib***
[name:]<func>	**Inline function func**
except:<func>	**Do not inline function func**
size:<n>	**Inline only functions smaller than n statements (approximate)**
levels:<n>	**Inline n levels of functions**

- For C++ Codes, PGI Recommends IPA-based inlining or –Minline=levels:10!

Effect of IPA on the WUPWISE Benchmark

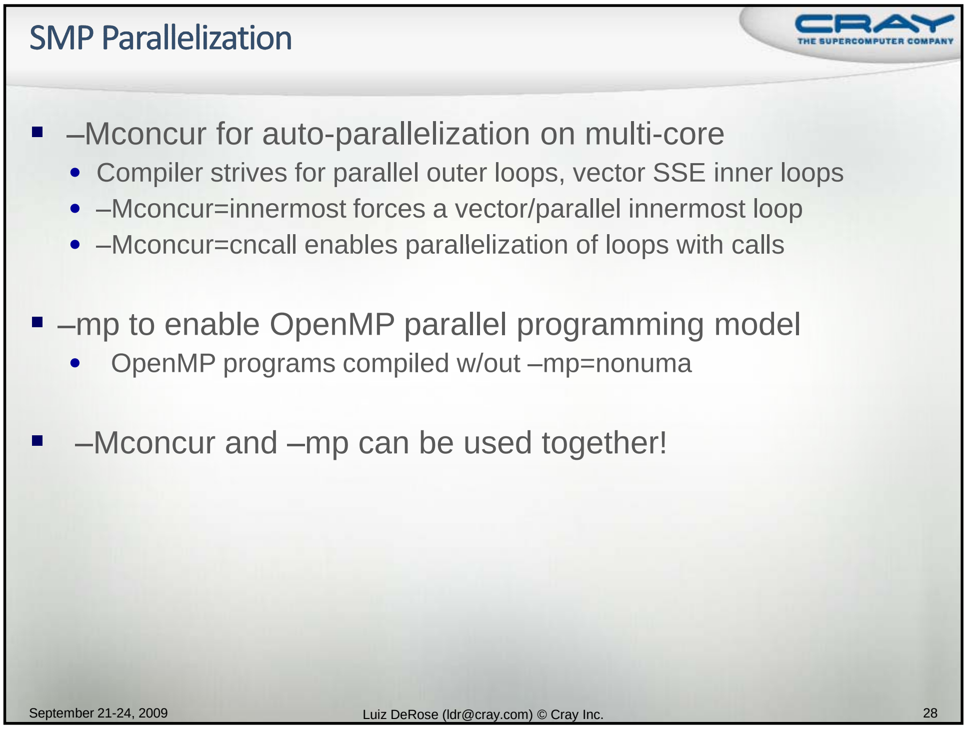
PGF95 Compiler Options	Execution Time in Seconds
–fastsse	156.49
–fastsse –Mipa=fast	121.65
–fastsse –Mipa=fast,inline	91.72

- –Mipa=fast => constant propagation =>
  - compiler sees complex matrices are all 4x3 => completely unrolls loops
- –Mipa=fast,inline => small matrix multiplies are all inlined



Other C++ recommendations

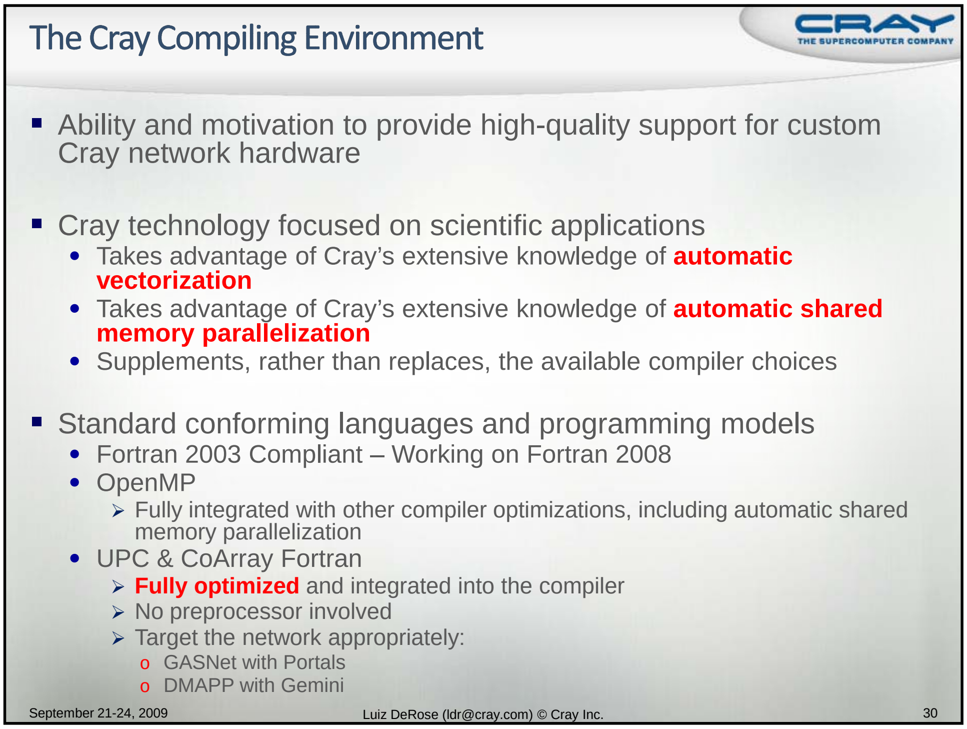
- Encapsulation, Data Hiding - small functions, inline!
- Exception Handling – use –no_exceptions until 7.0
- Overloaded operators, overloaded functions – okay
- Pointer Chasing - -Msafeptr, restrict qualifer, 32 bits?
- Templates, Generic Programming – now okay
- Inheritance, polymorphism, virtual functions – runtime lookup or check, no inlining, potential performance penalties



SMP Parallelization

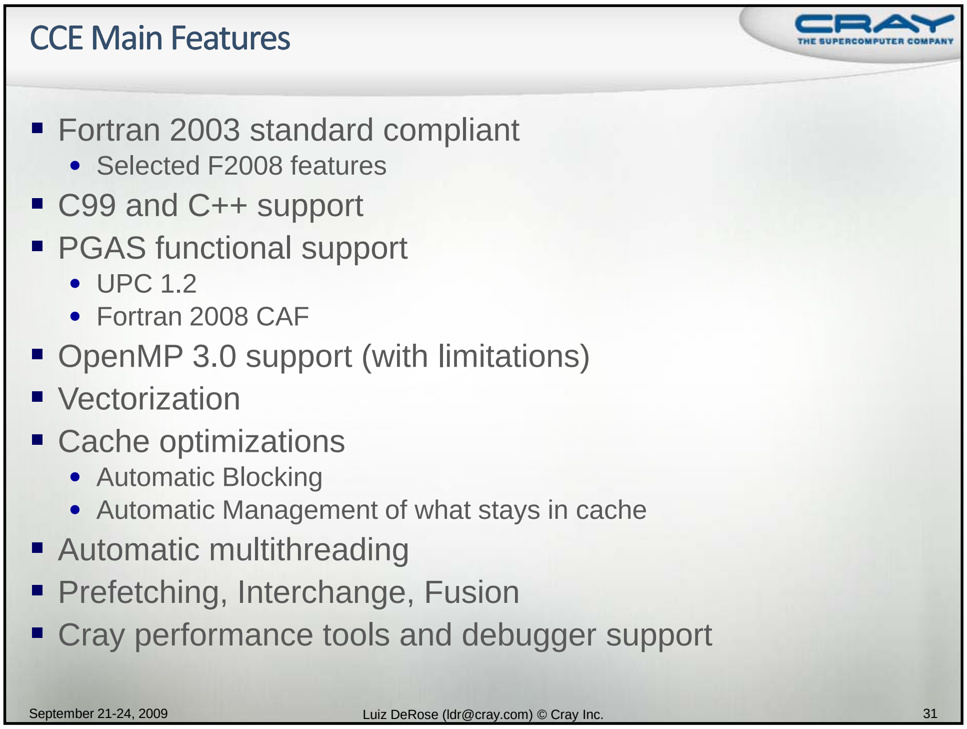
- –Mconcur for auto-parallelization on multi-core
  - Compiler strives for parallel outer loops, vector SSE inner loops
  - –Mconcur=innermost forces a vector/parallel innermost loop
  - –Mconcur=cncall enables parallelization of loops with calls
- –mp to enable OpenMP parallel programming model
  - OpenMP programs compiled w/out –mp=nonuma
- –Mconcur and –mp can be used together!

The Cray Compiling Environment



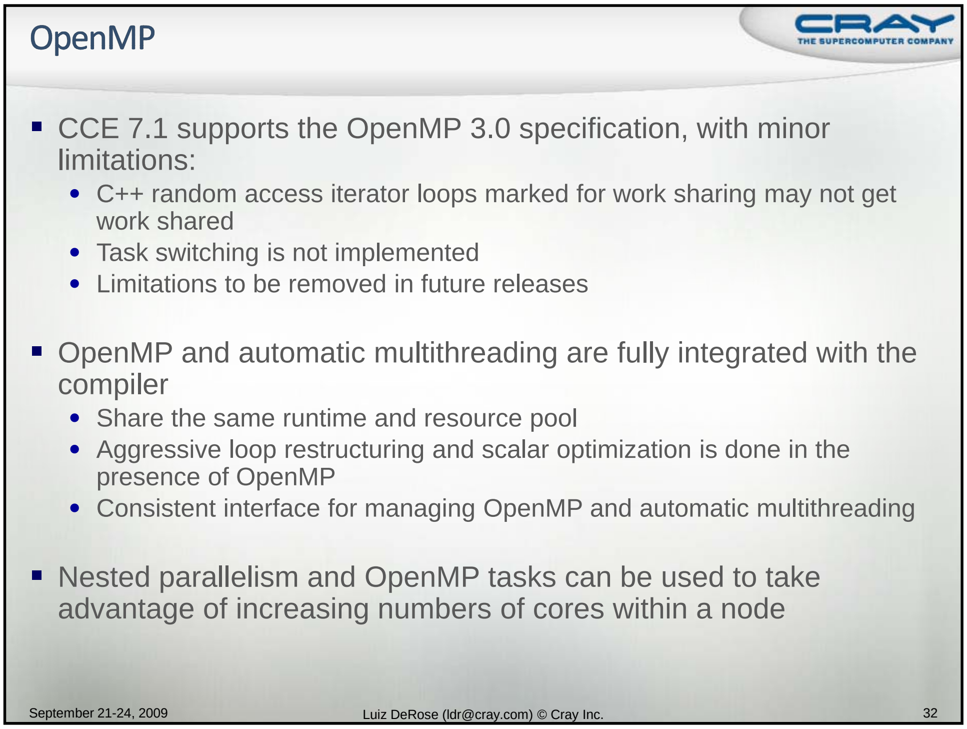
The Cray Compiling Environment

- Ability and motivation to provide high-quality support for custom Cray network hardware
- Cray technology focused on scientific applications
  - Takes advantage of Cray's extensive knowledge of **automatic vectorization**
  - Takes advantage of Cray's extensive knowledge of **automatic shared memory parallelization**
  - Supplements, rather than replaces, the available compiler choices
- Standard conforming languages and programming models
  - Fortran 2003 Compliant – Working on Fortran 2008
  - OpenMP
    - Fully integrated with other compiler optimizations, including automatic shared memory parallelization
  - UPC & CoArray Fortran
    - **Fully optimized** and integrated into the compiler
    - No preprocessor involved
    - Target the network appropriately:
      - GASNet with Portals
      - DMAPP with Gemini



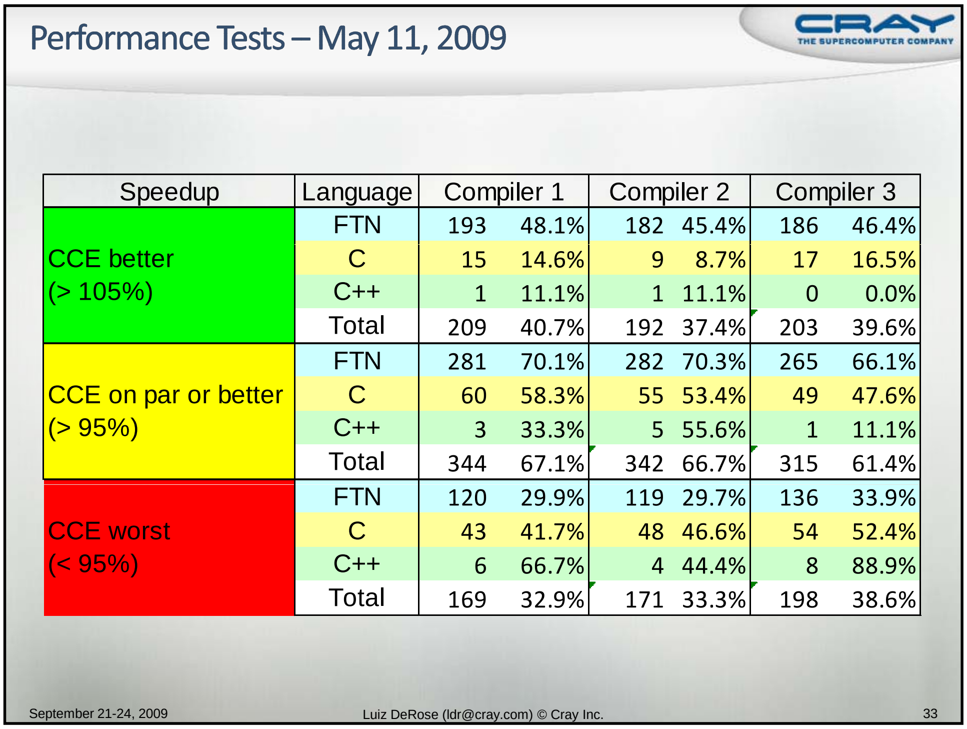
CCE Main Features

- Fortran 2003 standard compliant
  - Selected F2008 features
- C99 and C++ support
- PGAS functional support
  - UPC 1.2
  - Fortran 2008 CAF
- OpenMP 3.0 support (with limitations)
- Vectorization
- Cache optimizations
  - Automatic Blocking
  - Automatic Management of what stays in cache
- Automatic multithreading
- Prefetching, Interchange, Fusion
- Cray performance tools and debugger support



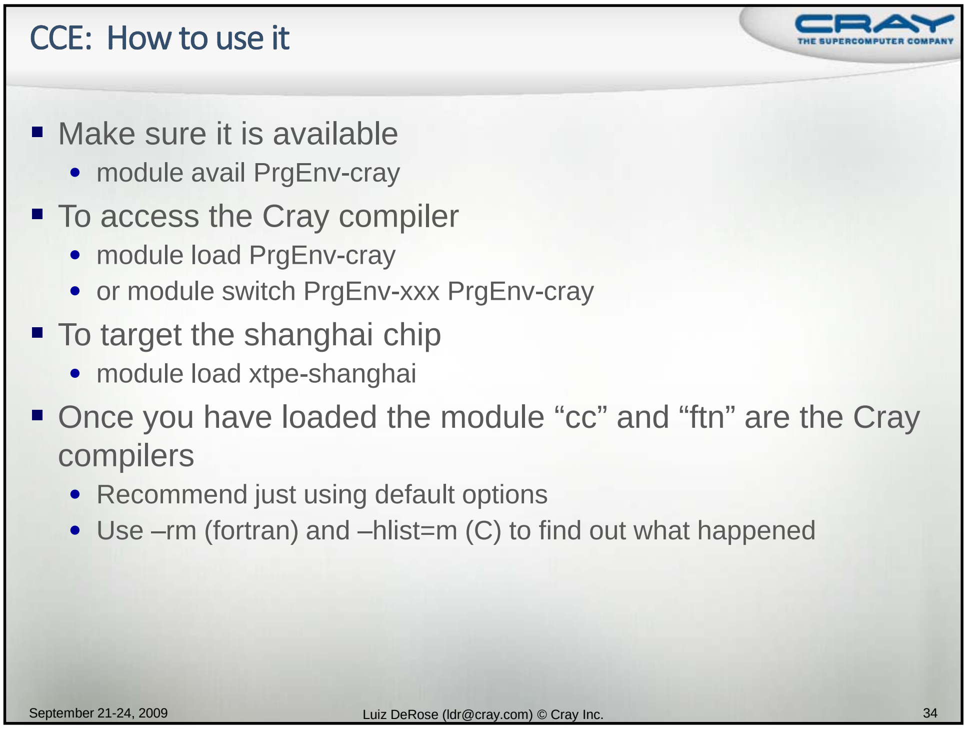
OpenMP

- CCE 7.1 supports the OpenMP 3.0 specification, with minor limitations:
  - C++ random access iterator loops marked for work sharing may not get work shared
  - Task switching is not implemented
  - Limitations to be removed in future releases
- OpenMP and automatic multithreading are fully integrated with the compiler
  - Share the same runtime and resource pool
  - Aggressive loop restructuring and scalar optimization is done in the presence of OpenMP
  - Consistent interface for managing OpenMP and automatic multithreading
- Nested parallelism and OpenMP tasks can be used to take advantage of increasing numbers of cores within a node



Performance Tests – May 11, 2009

Speedup	Language	Compiler 1		Compiler 2		Compiler 3	
CCE better (> 105%)	FTN	193	48.1%	182	45.4%	186	46.4%
	C	15	14.6%	9	8.7%	17	16.5%
	C++	1	11.1%	1	11.1%	0	0.0%
	Total	209	40.7%	192	37.4%	203	39.6%
CCE on par or better (> 95%)	FTN	281	70.1%	282	70.3%	265	66.1%
	C	60	58.3%	55	53.4%	49	47.6%
	C++	3	33.3%	5	55.6%	1	11.1%
	Total	344	67.1%	342	66.7%	315	61.4%
CCE worst (< 95%)	FTN	120	29.9%	119	29.7%	136	33.9%
	C	43	41.7%	48	46.6%	54	52.4%
	C++	6	66.7%	4	44.4%	8	88.9%
	Total	169	32.9%	171	33.3%	198	38.6%



CCE: How to use it

- Make sure it is available
  - module avail PrgEnv-cray
- To access the Cray compiler
  - module load PrgEnv-cray
  - or module switch PrgEnv-xxx PrgEnv-cray
- To target the shanghai chip
  - module load xtpe-shanghai
- Once you have loaded the module “cc” and “ftn” are the Cray compilers
  - Recommend just using default options
  - Use –rm (fortran) and –hlist=m (C) to find out what happened

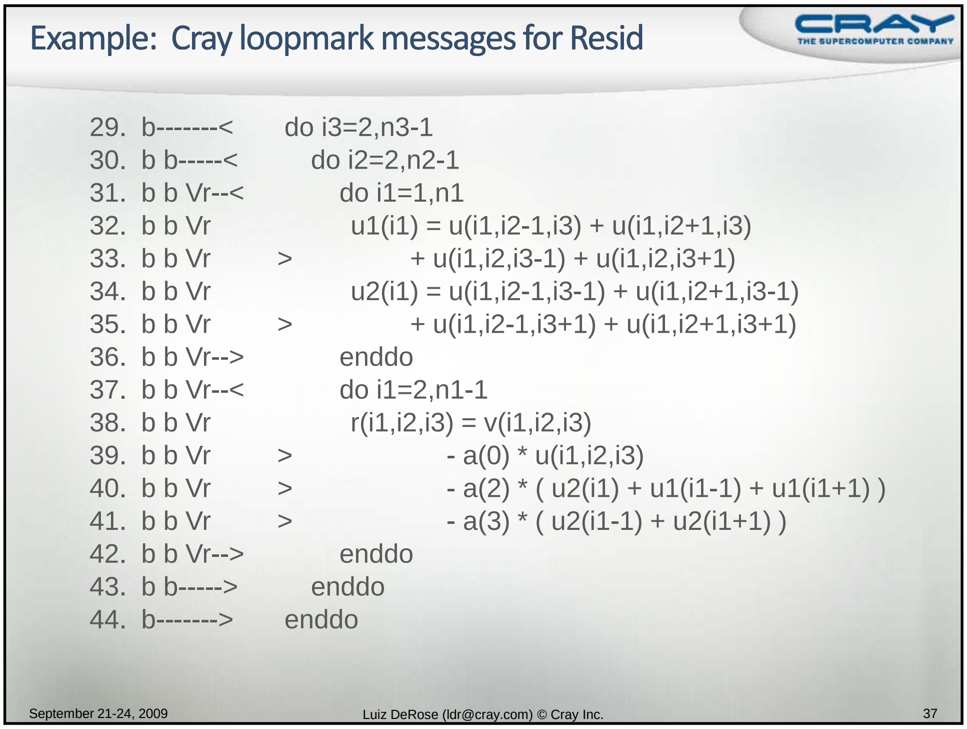
Cray Opteron Compiler: Directives

- Cray compiler supports a full and growing set of directives and pragmas
  - !dir$ concurrent
  - !dir$ ivdep
  - !dir$ interchange
  - !dir$ unroll
  - !dir$ loop_info [max_trips] [cache_na] ... Many more
  - !dir$ blockable
- See
  - **man directives**
  - **man loop_info**

Loopmark: Compiler Feedback

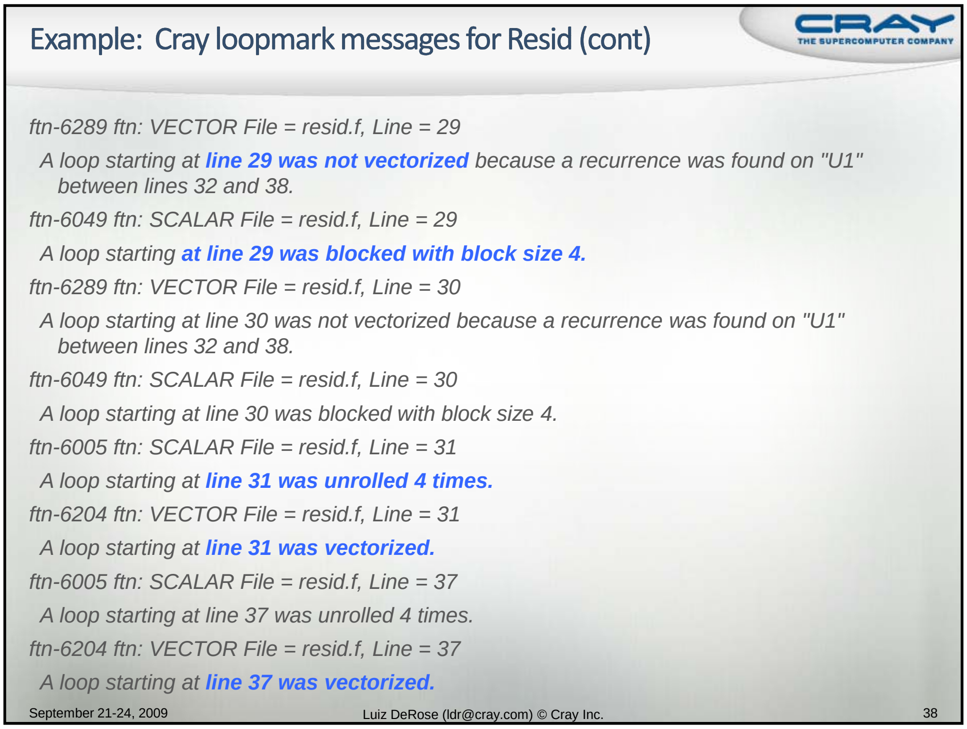
- Compiler can generate a “filename.lst” file
  - ftn –rm … or cc –hlist=m
  - Contains annotated listing of your source code with letter indicating important optimizations

```
%%%    Loopmark Legend    %%%
   Primary Loop Type        Modifiers
   ------- ---- ----        ---------
                             a - vector atomic memory operation
  A  - Pattern matched       b – blocked
  C  - Collapsed             f – fused
  D  - Deleted               i – interchanged
  E  - Cloned                m - streamed but not partitioned
  I  - Inlined               p - conditional, partial and/or computed
  M  - Multithreaded         r – unrolled
  P  - Parallel/Tasked       s – shortloop
  V  - Vectorized            t - array syntax temp used
  W  - Unwound               w - unwound
```



Example: Cray loopmark messages for Resid

```
29.  b-------<      do i3=2,n3-1
30.  b b-----<         do i2=2,n2-1
31.  b b Vr--<            do i1=1,n1
32.  b b Vr                 u1(i1) = u(i1,i2-1,i3) + u(i1,i2+1,i3)
33.  b b Vr        >               + u(i1,i2,i3-1) + u(i1,i2,i3+1)
34.  b b Vr                 u2(i1) = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
35.  b b Vr        >               + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
36.  b b Vr-->            enddo
37.  b b Vr--<            do i1=2,n1-1
38.  b b Vr                 r(i1,i2,i3) = v(i1,i2,i3)
39.  b b Vr        >                   - a(0) * u(i1,i2,i3)
40.  b b Vr        >                   - a(2) * ( u2(i1) + u1(i1-1) + u1(i1+1) )
41.  b b Vr        >                   - a(3) * ( u2(i1-1) + u2(i1+1) )
42.  b b Vr-->            enddo
43.  b b----->         enddo
44.  b------->      enddo
```



Example: Cray loopmark messages for Resid (cont)

ftn-6289 ftn: VECTOR File = resid.f, Line = 29

A loop starting at ***line 29 was not vectorized*** *because a recurrence was found on "U1" between lines 32 and 38.*

ftn-6049 ftn: SCALAR File = resid.f, Line = 29

A loop starting ***at line 29 was blocked with block size 4.***

ftn-6289 ftn: VECTOR File = resid.f, Line = 30

A loop starting at line 30 was not vectorized because a recurrence was found on "U1" between lines 32 and 38.

ftn-6049 ftn: SCALAR File = resid.f, Line = 30

A loop starting at line 30 was blocked with block size 4.

ftn-6005 ftn: SCALAR File = resid.f, Line = 31

A loop starting at ***line 31 was unrolled 4 times.***

ftn-6204 ftn: VECTOR File = resid.f, Line = 31

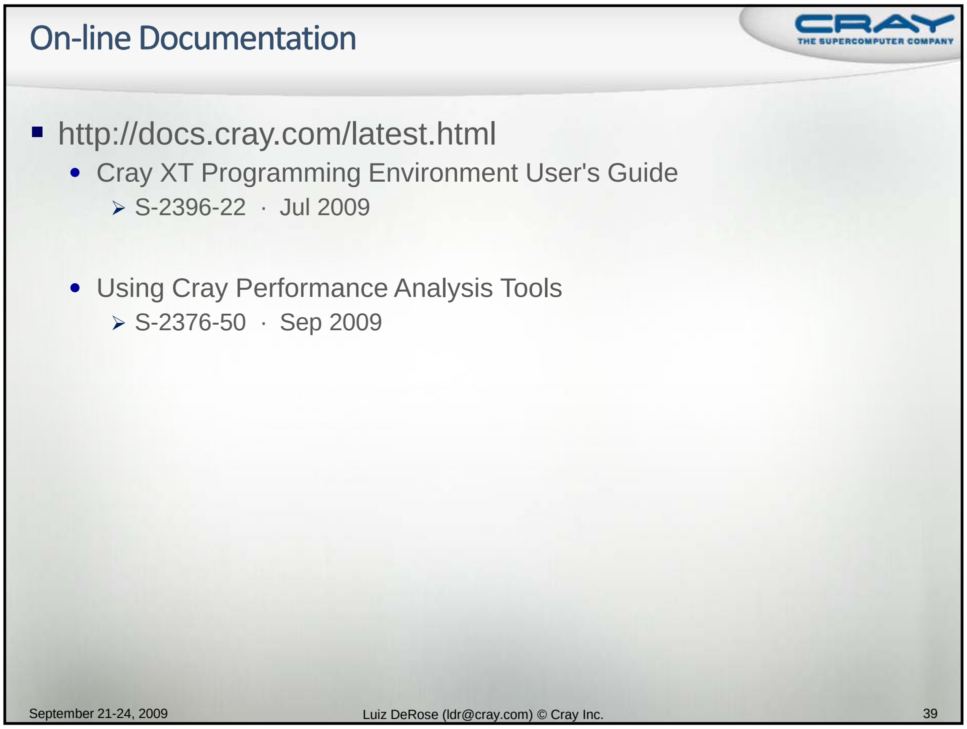
A loop starting at ***line 31 was vectorized.***

ftn-6005 ftn: SCALAR File = resid.f, Line = 37

A loop starting at line 37 was unrolled 4 times.

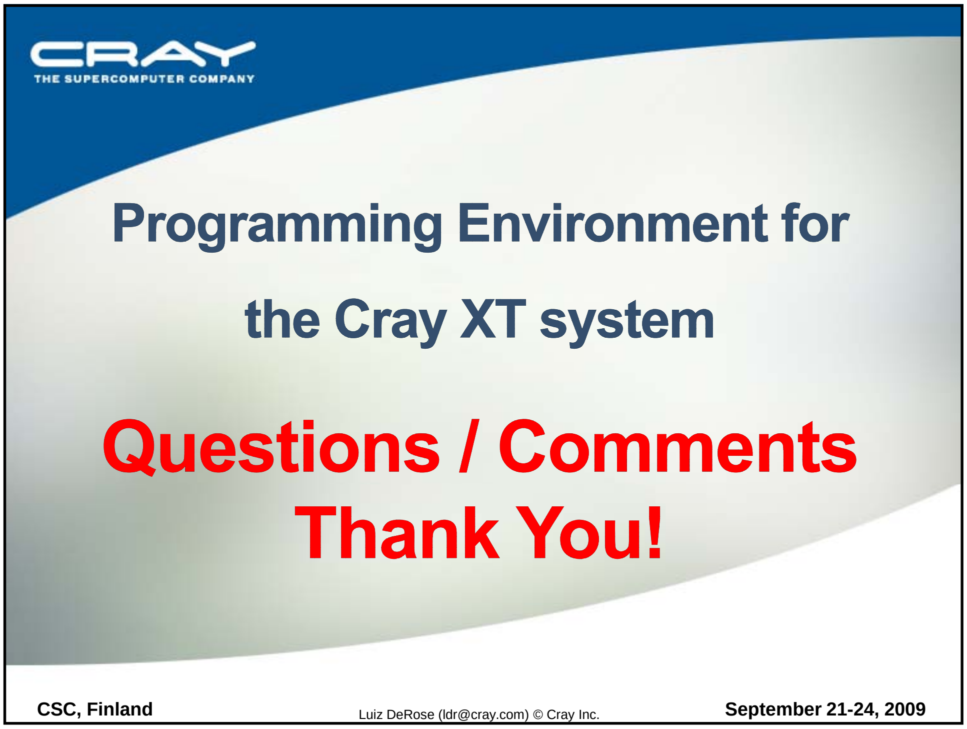
ftn-6204 ftn: VECTOR File = resid.f, Line = 37

A loop starting at ***line 37 was vectorized.***



On-line Documentation

- http://docs.cray.com/latest.html
  - Cray XT Programming Environment User's Guide
    - S-2396-22 · Jul 2009
  - Using Cray Performance Analysis Tools
    - S-2376-50 · Sep 2009



Programming Environment for the Cray XT system

Questions / Comments

Thank You!