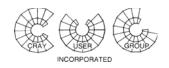
The Supercomputer Company

The BlackWidow Programming Environment

Luiz DeRose Programming Environments Director Cray Inc.

• CUG 2006





This Presentation May Contain Some Preliminary Information, Subject To Change

Outline

- BlackWidow Programming Environment overview
- Programming Models
- Compilers
 - Fortran 2003 standard compliancy status
 - Compiler optimizations
- Scientific Libraries
- Tools
- Conclusions



BlackWidow PE Overview

- Programming Environment goals:
 - Help users achieve highest possible performance from the hardware
 - Provide a natural follow on to the Cray X1 and Cray X1E systems
 - Continue the move towards a common software infrastructure among Cray products
 - Same look and feel between Cray systems
 - Easy of use

Programming Environment Features

- Programming models:
 - Distributed Memory
 - MPI
 - SHMEM
 - Shared Memory
 - OpenMP
 - PGAS
 - Co-Array Fortran (CAF)
 - Unified Parallel C (UPC)
- Compilers
 - Fortran
 - C/C++
 - gcc

- Optimized scientific and math libraries
- Tools:
 - TotalView debugger
 - Performance analysis
 - CrayPat Performance
 Collector
 - Cray Apprentice²
 Performance Analyzer

Outline

- BlackWidow Programming Environment overview
- Programming Models
- Compilers
 - Fortran 2003 standard compliancy status
 - Compiler optimizations
- Scientific Libraries
- Tools
- Conclusions

MPI

- Implementation based on MPICH2 from ANL
- Abstract Device Interfaces(ADI) for BlackWidow
 - ADI for low-latency shared memory on node
 - ADI for distributed shared memory across nodes which is more scalable
- Optimized and enhanced X1 collectives integrated into the MPICH2 framework
- Optimized Remote Memory Access (RMA) fully supported including passive RMA
- Full MPI-2 support with the exception of
 - Dynamic process management (MPI_comm_spawn)
 - Thread safety
 - planned for subsequent releases

Cray SHMEM

- Fully optimized Cray SHMEM library supported
 - Based on highly optimized X1 version
- Enhanced shmem_barrier planned for initial release
 - Additional optimizations in subsequent releases



OpenMP

- OpenMP implementation conforms to the OpenMP 2.0 specification
- BlackWidow implementation will be equivalent to the X1
 - Support is fully integrated in the compilers
 - OpenMP semantics are seen and understood by front ends, optimizer, and code generator
 - OpenMP parallelism cannot span nodes
 - Maximum OMP_NUM_THREADS will be 4



CAF and UPC

- Co-Array Fortran (CAF)
 - Conforms to 1998 specification, plus extensions
 - Will follow Fortran standard once that is codified
- Unified Parallel C (UPC)
 - Conforms to UPC 1.2 specification
- BlackWidow implementations will be equivalent to the X1
 - Support is fully integrated in the compilers
 - CAF and UPC semantics are seen and understood by front ends, optimizer, and code generator
 - BlackWidow remote addressing enhances
 performance

Outline

- BlackWidow Programming Environment overview
- Programming Models
- Compilers
 - C and Fortran 2003 standard compliancy status
 - Compiler optimizations
- Scientific Libraries
- Tools
- Conclusions



Language Standard Conformance

- Fortran ISO/IEC 1539:1-2004 (F2003) compliance planned for FCS
 - Currently Full F95 & subset of F2003
- C standard ISO/IEC 9899:1999 (C99)
 - Minor exceptions to c99 compliance
- C++ standard ISO/IEC 14882:1998 with some exceptions
 - Uses gnu c++ runtime library
- gcc targeting BlackWidow
 - Intended for command builds or for codes that depend on complete gcc compliance
 - Support of dynamic linking of commands



Fortran 2003 Supported Features

- Highlights
 - Allocatable components, dummy arguments and Function results
 - C interoperability
 - Procedure pointers
 - IEEE arithmetic support
 - Parameterized derived types
 - Mixed PUBLIC and PRIVATE component attributes



Planned PE 5.6 F2003 Features

- Object oriented features
 - Extensible derived types
 - Single inheritance type extension
 - Type bound procedures and finalization
 - Polymorphic objects
 - Select type construct
- Assumed and deferred type parameters
- Asynchronous and stream I/O

Outline

- BlackWidow Programming Environment overview
- Programming Models
- Compilers
 - Fortran 2003 standard compliancy status
 - Compiler optimizations
- Scientific Libraries
- Tools
- Conclusions

CRAY

BlackWidow Compiler Optimizer

- Derived from the production Cray X1 optimizer
- Changes from the Cray X1:
 - BlackWidow is little-endian
 - No multistreaming processor on BlackWidow
 - Compiler generates single-processor code with automatic vectorization
 - Makes use of the new and improved vector capabilities of the BlackWidow architecture
 - Vector atomic memory operations allow for more aggressive vectorization

CRAY

BlackWidow Enhancements

- Support is complete for instruction set differences between the X1 and BlackWidow, including:
 - Inclusive-or bit matrix multiplication
 - Same optimization characteristics as standard bit matrix multiplication
 - Full-speed bit matrix operations
 - Full-speed vector compress
 - 32-bit vector compress and gather/scatter
 - Vector registers contain 128 elements
 - Vector masks are 128 bits wide
 - 32 vector mask registers
 - Branch prediction hints
 - Vector atomic memory operations



BlackWidow Branch With Hint

BN,P Ai,imm20 BN,P Si,imm20

- Backward jump, predict branch taken
- Forward jump, predict branch not taken
- Compiler automatically sets the hint information
- Information from the *probability* and *loop_info* compiler directives can influence the branch hint

Vector Atomic Memory Operations

• Instructions of the form:

[Aj,Vk]Vi,Mm,AADD,hint [Aj,Ak]Vi,Mm,AXOR,hint

- 64-bit integer add, and, or, and xor
- Strided and indirect reference versions
- Indirect versions accept 32 and 64 bit indices
- Most of the instruction happens away from the main processor, in the memory system
- Used for update operations
- Used for simple operations on non-local memory
- Stores to 8- and 16-bit arrays are vectorized using two vector AMOs

Vector AMOs (continued)

 Examples of statements that use vector atomic memory operations:

> a(ix(i)) = a(ix(i)) + <expr>; a(ix(i)) = a(ix(i)) & <expr>; a(ix(i)) = a(ix(i)) | <expr>; a(ix(i)) = a(ix(i)) ^ <expr>; a(ix(i)) = ~a(ix(i)); a(ix(i)) = -a(ix(i)); // 64-bit floating point

 Non-indirect memory references are also mapped to vector AMOs, especially when striding over a CAF or UPC dimension

Outline

- BlackWidow Programming Environment overview
- Programming Models
- Compilers
 - Fortran 2003 standard compliancy status
 - Compiler optimizations
- Scientific Libraries
- Tools
- Conclusions

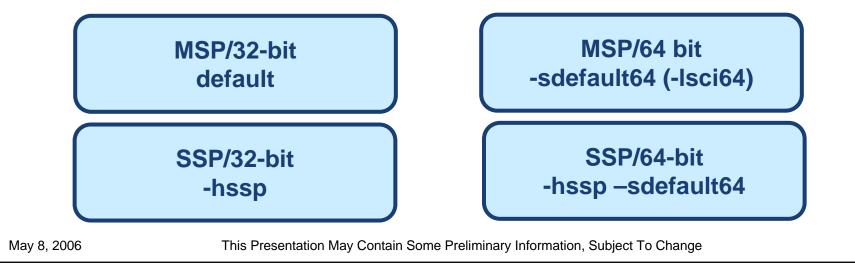


BlackWidow LibSci

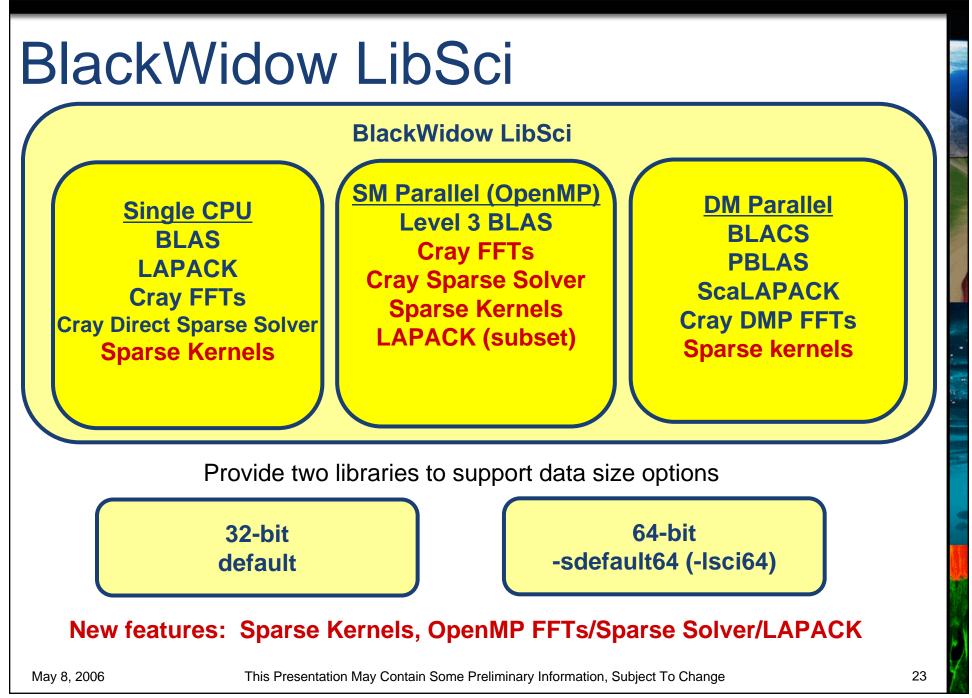
- Extension of the X1/X1E LibSci
 - Additional OpenMP support:
 - FFTs
 - Some LAPACK solvers
 - Cray's direct sparse solver
 - Sparse BLAS
- Tuned for
 - BlackWidow memory model
 - One-sided communication



Provide four libraries to support streaming and data size options







FFTs

- Cache Tuned
- Added OpenMP capability
- Distributed memory parallel FFTs are hybrid
 - Tuned with one-sided communication across nodes
 - CAF and/or SHMEM
 - OpenMP within a node

Sca/LAPACK

- Added OpenMP versions of LAPACK routines:
 - LU
 - Symmetric Tridiagonalization
 - Cholesky
 - QR
- Collaboration with LBNL to vectorize MRRR eigensolver
- Co-Array Fortran PBLAS

Sparse Computation Support

- Extension of work started for XT line
- Provide sparse BLAS routines to support iterative solvers in
 - PETSc
 - Trilinos
 - User-defined
- OpenMP direct sparse solvers

Outline

- BlackWidow Programming Environment overview
- Programming Models
- Compilers
 - Fortran 2003 standard compliancy status
 - Compiler optimizations
- Scientific Libraries
- Tools
- Conclusions



The Cray Tools Strategy

- Must be easy to use
- Integrated performance tools solution
- Strategy based on the three main steps normally used for application optimization and tuning:
 - 1. Debug application
 - Nobody really cares how fast the program can compute the wrong answer
 - 2. Single processor and vector optimization
 - Make the common case fast
 - 3. Parallel processing and I/O optimization
 - Communication / barriers / etc



Debuggers

- Cray maintains a strong interaction with Etnus
 - TotalView is our debugger of choice on all platforms
 - TotalView is recognized by the community as the state of the art parallel visual debugger
 - BlackWidow TotalView port is done at Cray
 - Started with TotalView 7 base

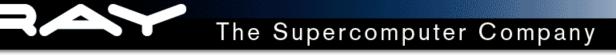


Cray Performance Tools

- PAT_HWPC
 - Hardware Counters based measurement for whole application
- CrayPat Performance Collector
 - Automatic program instrumentation
 - Data collection
 - Performance file generation
- Cray Apprentice² Performance Analyzer
 - Performance visualization

Conclusions

- BlackWidow Programming Environment was built on top of the X1 software base, extending and optimizing its functionality
- Common software infrastructure provides the same look and feel on all Cray Platforms
- Easy to use programming environment targeted to help users to get highest possible performance from the hardware



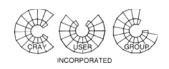
The BlackWidow Programming Environment

Thanks!

Questions / Comments

• CUG 2006





This Presentation May Contain Some Preliminary Information, Subject To Change