IBM Power Systems Compiler Roadmap

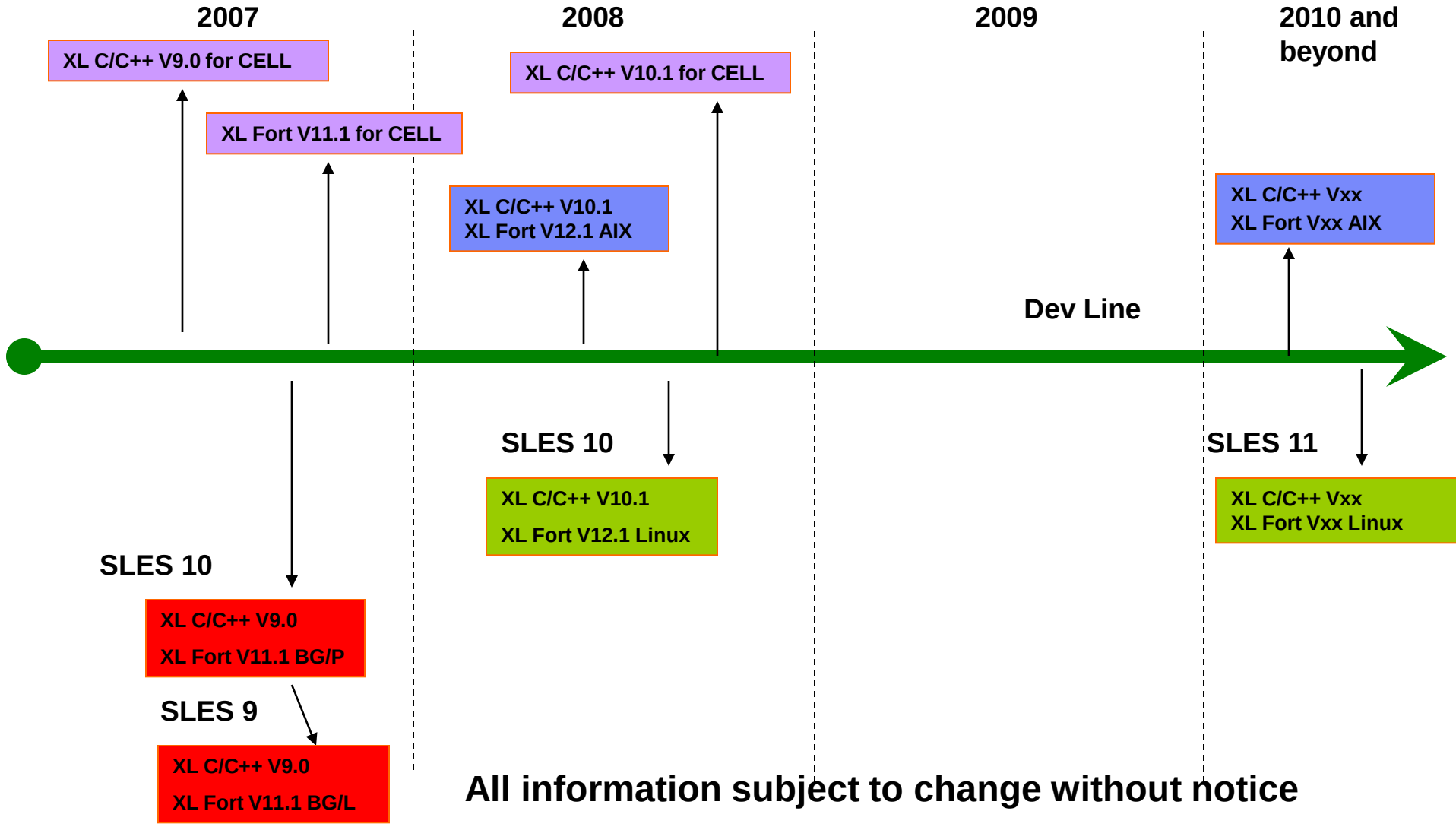
Roch Archambault
IBM Toronto Laboratory
archie@ca.ibm.com

IBM Rational Disclaimer

Agenda

- Overall Roadmap
- The Power Systems Compiler Products
- Detailed Roadmaps
  - Common Features & Compiler Architecture
  - XL Fortran
  - XL C/C++
  - XL Compilers for Blue Gene
  - XL Compilers for Cell
  - XL UPC Compiler
- Examples of Tasks in OpenMP 3.0
- New –qstrict suboptions
- Online Documentation
- Q&A

Roadmap of XL Compiler Releases



All information subject to change without notice

The Power Systems Compiler Products: Previous Versions

- **All POWER4, POWER5, POWER5+ and PPC970 enabled**
  - XL C/C++ Enterprise Edition V8.0 for AIX
  - XL Fortran Enterprise Edition V10.1 for AIX
  - XL C/C++ Advanced Edition V8.0 for Linux (SLES 9 & RHEL4)
  - XL Fortran Advanced Edition V10.1 for Linux (SLES 9 & RHEL4)
  - XL C/C++ Advanced Edition V8.0.1 for Linux (SLES 10 & RHEL4)
  - XL Fortran Advanced Edition V10.1.1 for Linux (SLES 10 & RHEL4)
  - XL C/C++ Enterprise Edition for AIX, V9.0 **(POWER6 enabled)**
  - XL Fortran Enterprise Edition for AIX, V11.1 **(POWER6 enabled)**
  - XL C/C++ Advanced Edition for Linux, V9.0 **(POWER6 enabled)**
  - XL Fortran Advanced Edition for Linux, V11.1 **(POWER6 enabled)**

The Power Systems Compiler Products: Latest Versions

- **All POWER4, POWER5, POWER6 and PPC970 enabled**
  - XL C/C++ for AIX, V10.1 (July 2008)
  - XL Fortran for AIX, V12.1 (July 2008)
  - XL C/C++ for Linux, V10.1 (September 2008)
  - XL Fortran for Linux, V12.1 (September 2008)
- **Blue Gene (BG/L and BG/P) enabled**
  - XL C/C++ Advanced Edition for BG/L, V9.0
  - XL Fortran Advanced Edition for BG/L, V11.1
  - XL C/C++ Advanced Edition for BG/P, V9.0
  - XL Fortran Advanced Edition for BG/P, V11.1
- **Cell/B.E. cross compiler products:**
  - XL C/C++ for Multicore Acceleration for Linux on Power Systems, V10.1 (4Q2008)
  - XL C/C++ for Multicore Acceleration for Linux on x86 Systems, V10.1 (4Q2008)
  - XL Fortran for Multicore Acceleration for Linux on System p, V11.1

The Power Systems Compiler Products: Latest Versions

- **Technology Preview currently available from alphaWorks**

  XL UPC language support on AIX and Linux

  Download: **http://www.alphaworks.ibm.com/tech/upccompiler**

  XL C/C++ for Transactional Memory for AIX

  Download: **http://www.alphaworks.ibm.com/tech/xlcstm**

  CDT for AIX

  Download: http**://www.alphaworks.ibm.com/tech/cremoteide**

- **IBM Debugger for AIX (with Fortran support)**

  Download: **https://www.ibm.com/services/forms/preLogin.do?source=swerpsw**

The Power Systems Compiler Products: Future Versions

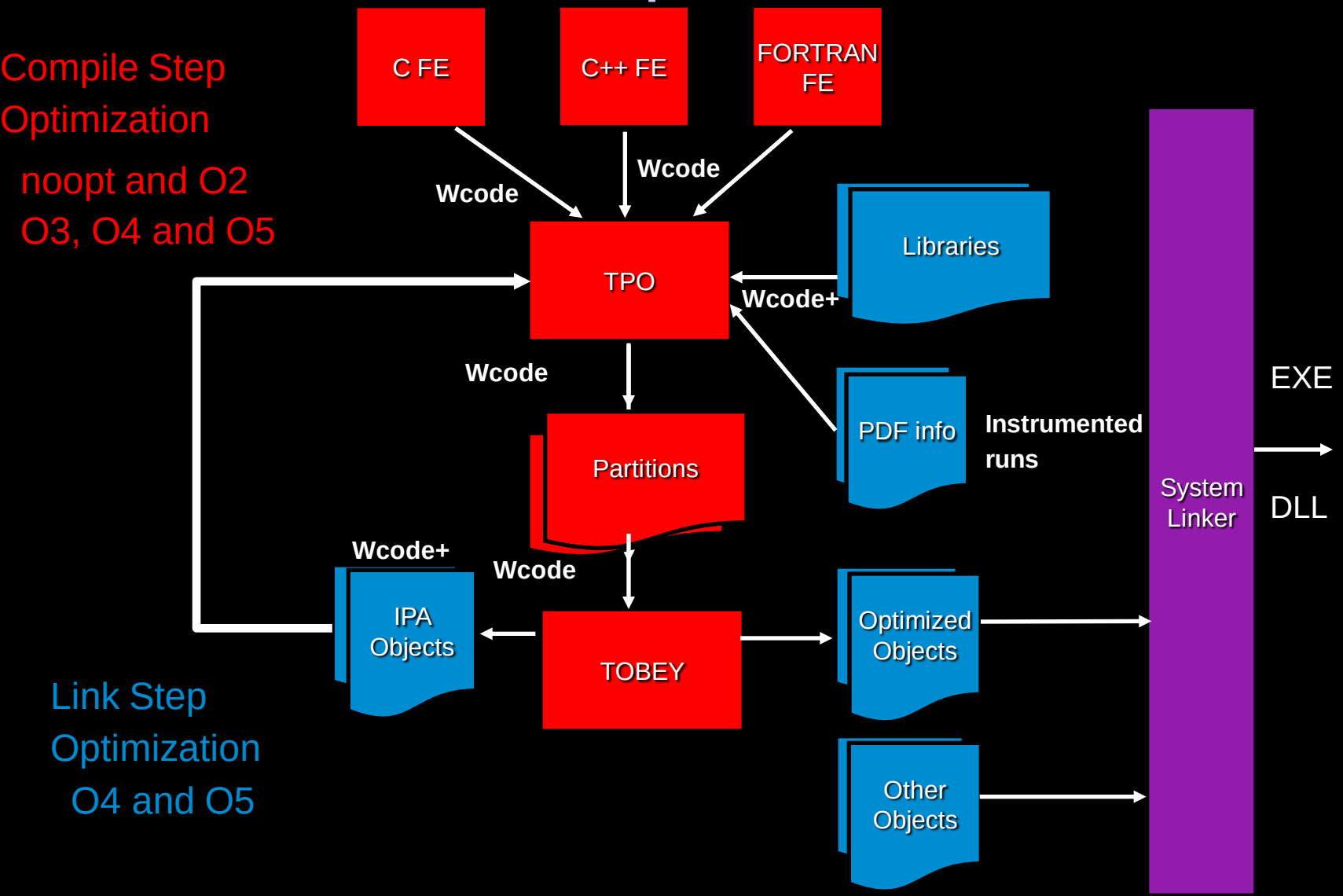
- POWER7 support
  - XL C/C++ for AIX, Vxx (2010 and beyond)
  - XL Fortran for AIX, Vxx (2010 and beyond)
  - XL C/C++ for Linux, Vxx (2010 and beyond)
  - XL Fortran for Linux, Vxx (2010 and beyond)

All information subject to change without notice

Common Fortran, C and C++ Features

- Linux (SLES and RHEL) and AIX, 32 and 64 bit
- Debug support
  - Debuggers on AIX:
    - Total View (TotalView Technologies), DDT (Allinea), IBM Debugger and DBX
  - Debuggers on Linux:
    - TotalView, DDT and GDB
- Full support for debugging of OpenMP programs (TotalView)
- Snapshot directive for debugging optimized code
- Portfolio of optimizing transformations
  - Instruction path length reduction
  - Whole program analysis
  - Loop optimization for parallelism, locality and instruction scheduling
  - Use profile directed feedback (PDF) in most optimizations
- Tuned performance on POWER3, POWER4, POWER5, PPC970, PPC440, PPC450, POWER6 and CELL systems
- Optimized OpenMP

IBM XL Compiler Architecture



XL Fortran Roadmap: Strategic Priorities

- **Superior Customer Service**
  - Continue to work closely with key ISVs and customers in scientific and technical computing industries
- **Compliance to Language Standards and Industry Specifications**
  - OpenMP API V2.5 (Full) and OpenMP API V3.0 (Partial)
  - Fortran 77, 90 and 95 standards
  - Fortran 2003 Standard
- **Exploitation of Hardware**
  - Committed to maximum performance on POWER4, PPC970, POWER5, POWER6, PPC440, PPC450, CELL and successors
  - Continue to work very closely with processor design teams

XL Fortran Version 12.1 for AIX/Linux – July/September 2008

New features since XL Fortran Version 10.1:

- Continued rollout of Fortran 2003
- Compliant to OpenMP V2.5
- Perform subset of loop transformations at –O3 optimization level
- Tuned BLAS routines (DGEMM and DGEMV) are included in compiler runtime (libxlopt)
- Recognize matrix multiply and replace with call to DGEMM
- Runtime check for availability of ESSL
- Support for auto-simdization and VMX intrinsics (and data types) on AIX
- Inline MASS library functions (math functions)
- Partial support for OpenMP V3.0
- Fine grain control for –qstrict option
- Improved compile/link time
- More Interprocedural data reorganization optimizations

XL C/C++ Roadmap: Strategic Priorities

- **Superior Customer Service**
- **Compliance to Language Standards and Industry Specifications**
  - ANSI / ISO C and C++ Standards
  - OpenMP API V3.0
- **Exploitation of Hardware**
  - Committed to maximum performance on POWER4, PPC970, POWER5, PPC440, POWER6, PPC450, CELL and successors
  - Continue to work very closely with processor design teams
- **Exploitation of OS and Middleware**
  - Synergies with operating system and middleware ISVs (performance, specialized function)
  - Committed to AIX Linux affinity strategy and to Linux on pSeries
- **Reduced Emphasis on Proprietary Tooling**
  - Affinity with GNU toolchain

XL C/C++ Version 10.1 for AIX/Linux – July/September 2008

New features since XL C/C++ Version 8.0:

- Exploit “restrict” keyword in C 1999
- Partial compliance to C++ TR1 libraries and Boost 1.34.1
- Support for -qtemplatedepth which allows the user to control number of recursive template instantiations allowed by the compiler.
- Exploit DFP and VMX on Power6.
- Improved inline assembler support
- Full support for OpenMP V3.0
- Fine grain control for –qstrict option
- Improved compile/link time
- More Interprocedural data reorganization optimizations

Blue Gene Compilers

XL C/C++ Advanced Edition for BG/P, V9.0 and XL Fortran Advanced Edition for BG/P, V11.1

- Support for OpenMP, automatic parallelization and dynamic linking
- Performance improvements: SIMD and other general optimizations
- MASS/MASSV performance improvements
- FEN (Front End Node) is SLES10
- March 2009 Fortran PTF available:
  http://www.ibm.com/support/docview.wss?rs=43&uid=swg24022541
- March 2009 C/C++ PTF available:
  http://www.ibm.com/support/docview.wss?rs=2239&uid=swg24020540

XL C/C++ Advanced Edition for BG/L, V9.0 and XL Fortran Advanced Edition for BG/L, V11.1

- Same code base as BG/P release except FEN is SLES 9
- GA is one month after BG/P GA
- April 2009 Fortran PTF available:
  http://www.ibm.com/support/docview.wss?rs=43&uid=swg24022947
- April 2009 C/C++ PTF available:
  http://www.ibm.com/support/docview.wss?rs=2239&uid=swg24022952

Cell/B.E. Compilers

- **Cross compilers products:**
  - Hosted on RHEL5U1 (Red Hat) and F7 (Fedora)
  - Hosted on x86 and PPC (separate products)
  - Support SDK 3.0 interfaces
  - Targets QS20, QS21 and QS22 Blades
  - IBM XL C/C++ for Multicore Acceleration for Linux, V9.0
  - IBM XL Fortran for Multicore Acceleration for Linux, V11.1 (Hosted on PPC only)

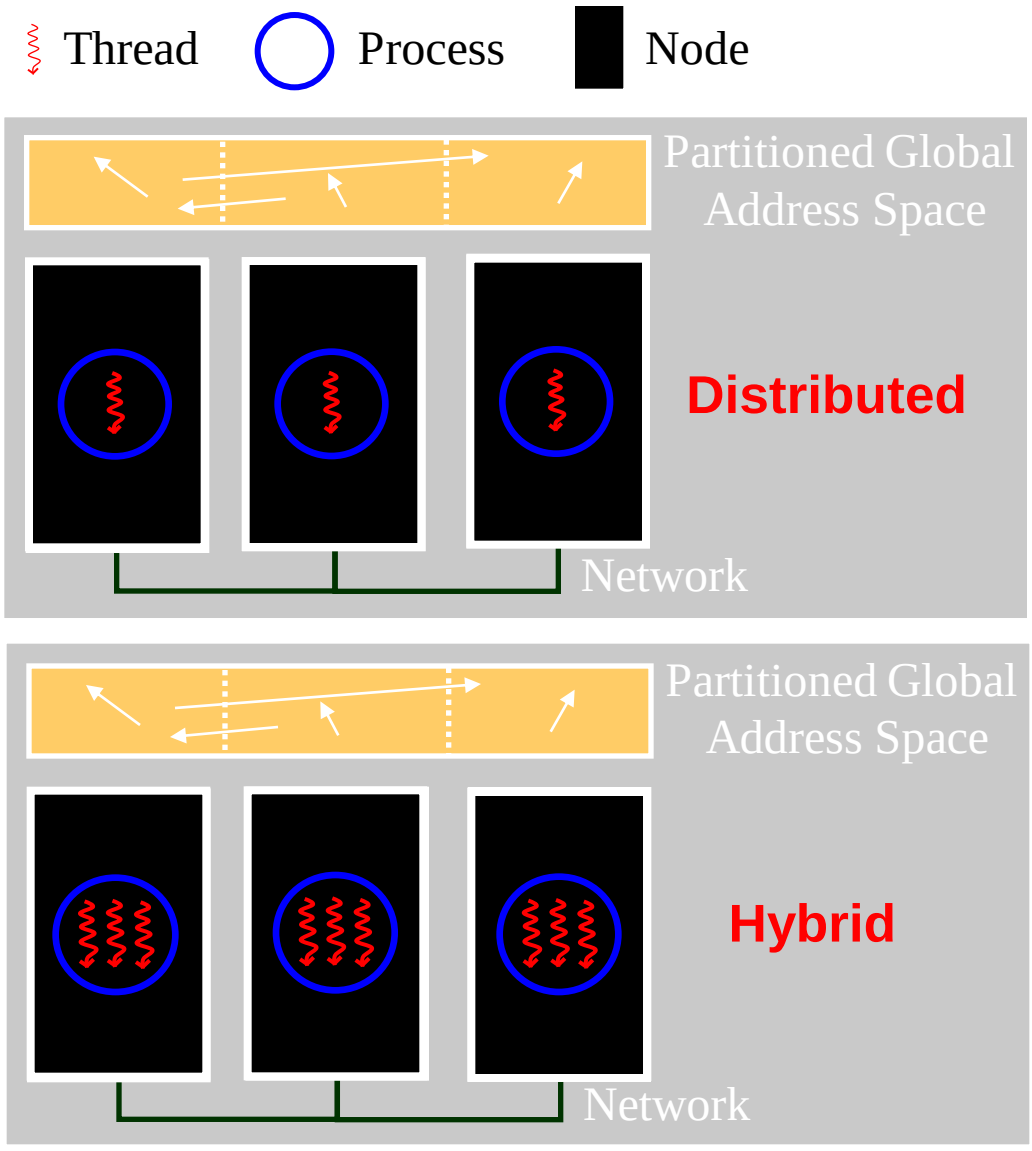
- **Latest cross compiler products:**
  - Hosted on RHEL5U2
  - Support SDK 3.1 interfaces
  - User directed single source compiler (using OpenMP)
  - IBM XL C/C++ for Multicore Acceleration for Linux, V10.1 (Hosted on x86 and PPC)

XL UPC Compiler

- **Tech preview on alphaWorks**
  - Based on XL C V10.1 compiler
  - Compiler generated interface to the runtime system is identical for shared and distributed memory implementations
  - Optimizations take advantage of system architecture knowledge
- **On AIX**
  - Shared Memory (pthreads)
  - Distributed (LAPI)
- **On Linux**
  - Shared Memory (pthreads)
  - Distributed (LAPI)
- **On BG/L**
  - BG Message Layer (based on XLC V9.1 compiler)
- **Using approximately 1000 test scenarios:**
  - GWU UPC test suite
  - UPC version of NAS benchmarks
  - Berkeley UPC test suite
  - MTU UPC test suite
  - HPC Challenge suite

All information subject to change without notice

Hybrid Execution Environment



Execution Environments

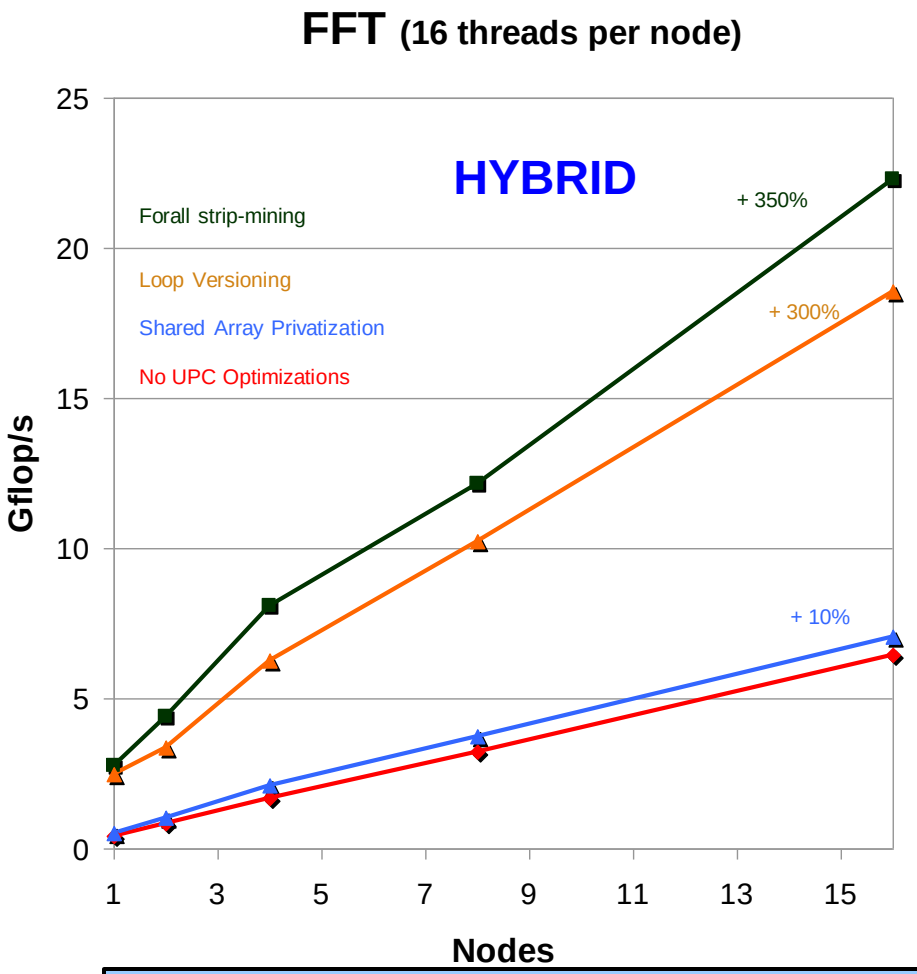
- Distributed: multiple nodes, one (or more) process per node running one thread
- Hybrid: multiple nodes, one (or more) processes per node running multiple threads

Memory Locality

- shared memory physically located on the node where the thread is running has lower latency than memory located on another node (affinity concept)
- compiler can exploit memory affinity information when thread mapping and cluster topology is known (static threads, num. nodes known)

All information subject to change without notice

FFT - Compiler Optimizations




```
typedef struct { double re, im; }  complex_t;
typedef shared [BF] complex_t   ComplexArray_t [N*N];

void transpose2 (ComplexArray_t A, ComplexArray_t B) {
   // tile exchange (communication)
   upc_barrier;
   // local transpose
   upc_forall(i = 0; i < N; i += bsize; &B[i*N])
     for (j = 0; j < N; j += bsize)
       for (ArrayIndex_t k = 0; k < bsize - 1; k++)
         for (ArrayIndex_t l = k + 1; l < bsize; l++) {
           complex_t c = B[(i+k)*N+(j+l)];
           B[(i+k)*N+(j+l)] = B[(i+l)*N+(j+k)];
           B[(i+l)*N+(j+k)] = c;
         }
```

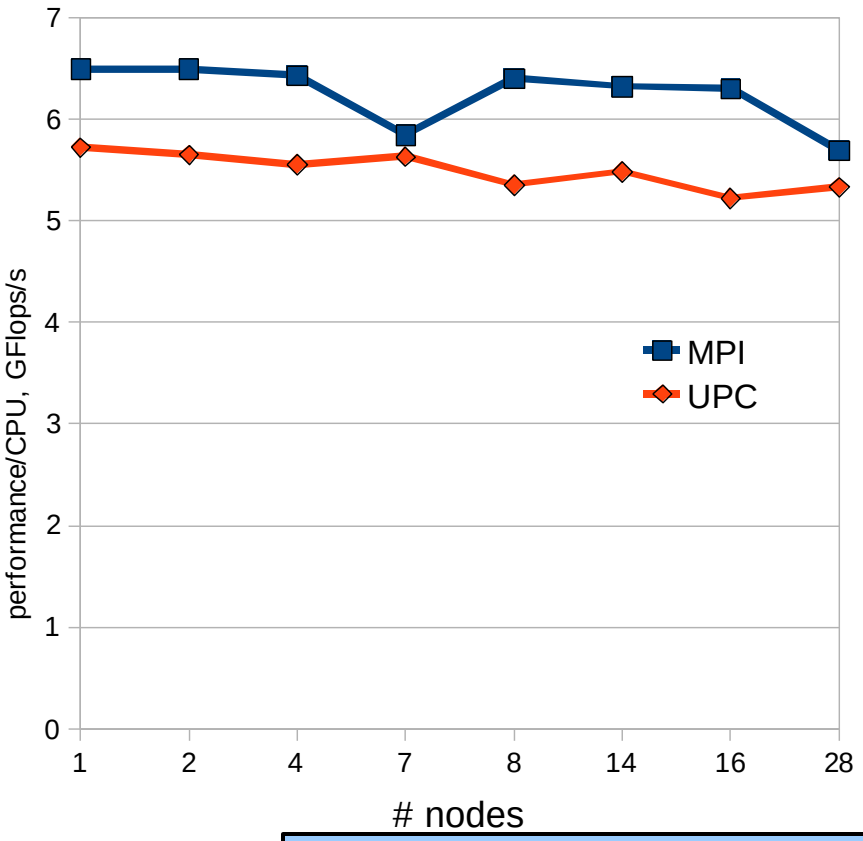
Privatized

1. version the upc_forall, this allows the analysis to reason about the locality of shared accesses done through pointers-to-shared
2. accesses in the local transpose are then recognized as local and privatized.

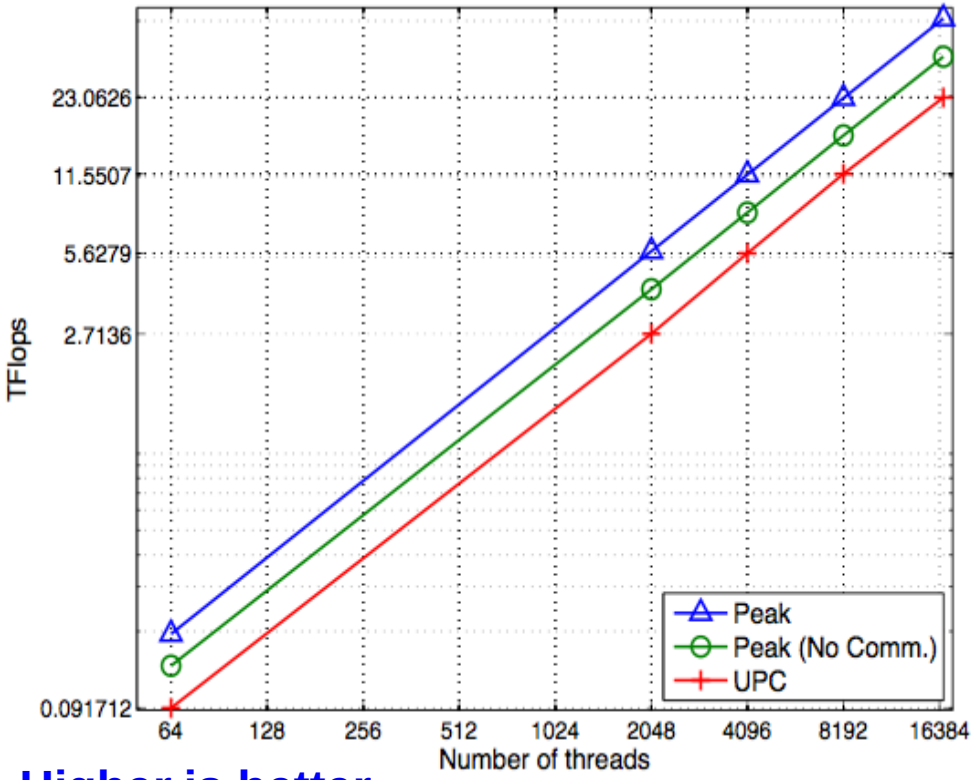
High Performance Linpack – MPI vs UPC

MPI vs UPC HPL

Power5 + HPS, 28 nodes, 16 processes/node



UPC HPL (Blue Gene/L, 16k nodes)



Higher is better

Performance: almost equivalent, lags **10%** behind MPI
Complexity: UPC code **1,430** lines, MPI code **30,744** lines

OpenMP 3.0: Examples of Task

- Pointer chasing

```
#pragma omp parallel
{
  #pragma omp single
  {
    while(p) {
      #pragma omp task
        process(p)
      p=p->next;
    }
  }
}
```

- Recursive algorithm

```
int fib(int n) {
  int x, y;
  if (n<2)
    return n;
  {
    #pragma omp task shared(x)
      x=fib(n-1);
    #pragma omp task shared(y)
      y=fib(n-2);
  }
  #pragma omp taskwait
  return x+y;
}
```

General –qstrict Suboptions

- **-qstrict=all** **-- be strict or**
- **-qstrict=none** **-- relaxed about all changes**

- **-qstrict=precision** **-- be strict or**
- **-qstrict=noprecision** **-- relaxed about changing precision**

- **-qstrict=exceptions** **-- be strict or**
- **-qstrict=noexceptions** **-- relaxed about changing exceptions (whether more or less or moved)**

[no]exceptions does not control *everything* that could produce different results, so different exceptions are possible even with –qstrict=noexceptions.

- **Each general suboption controls decisions itself, and also controls nested suboptions.**

General –qstrict Suboptions

- **-qstrict=ieeefp -- be strict or**
- **-qstrict=noieeefp -- relaxed about violating IEEE 754**

  [no]ieeefp controls individual operations defined by IEEE 754, not how operations interact or are ordered.

  Detail suboptions allow controlling specific aspects of [no]ieeefp. Most operations are affected by multiple detail suboptions, and also by [no]exceptions.

- **-qstrict=order -- be strict or**
- **-qstrict=noorder -- relaxed about operation order**

  [no]order controls the order between operations, as defined by language semantics, not how each operation is implemented.

  Detail suboptions allow controlling specific aspects of [no]order.

Examples

- **-qstrict=all:nooperationprecision:noreductionorder**
  - **Be strict about everything,**
  - **except about operationprecision,**
    part of what's needed to allow x / loop_constant => x * (1 / loop_constant),
    which is needed to allow faster MOD
    (note this allows other changes too),
  - **and about reductionorder,**
    to allow recognizing dot product and similar reductions
    and generating faster parallelized code,
    without allowing other reordering.

Examples

do i = 1, n
a(i) = b(i) / x
end do

-qstrict=operationprecision\
:exceptions:zerosigns

```
  LFL     fp0=x(gr31,0)

    . . .
CL.30:
  LFL     fp2=b[](gr4,8)
  AI      gr3=gr3,8
  STFL    a[](gr3,0)=fp1
  DFL     fp1=fp2,fp0,fcr   =b(i)/x
  AI      gr4=gr4,8
  BCT     ctr=CL.30
```

divide by x

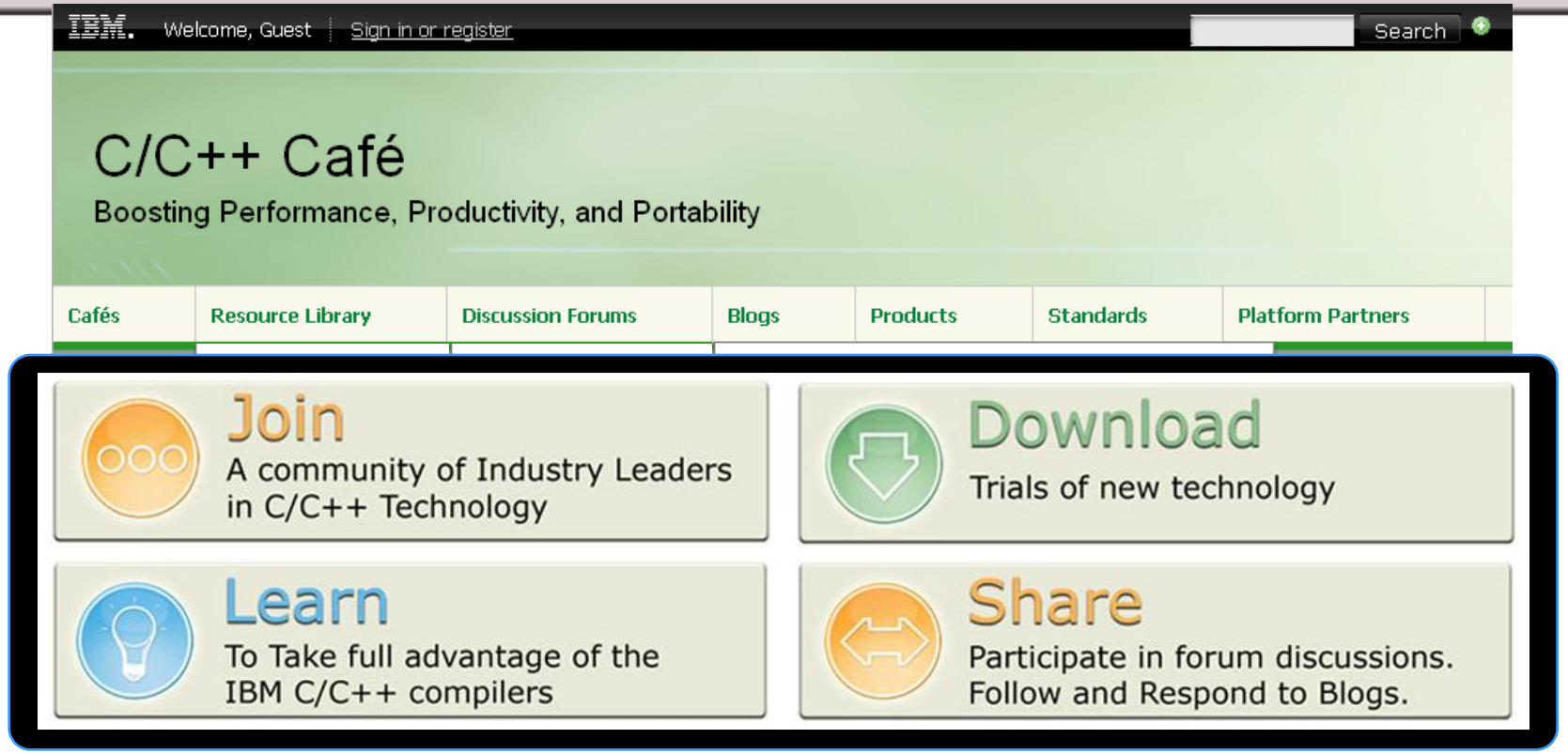
-qstrict=nooperationprecision\
:noexceptions:nozerosigns

```
  LFS     fp0=+CONSTANT_AREA(gr5,4)  =1
    . . .
  LFL     fp2=x(gr31,0)
  RCPFL   fp0=fp0,fp2,fcr            =1/x
    . . .
CL.30:
  AI      gr3=gr3,8
  LFL     fp2=b[](gr4,8)
  AI      gr4=gr4,8
  STFL    a[](gr3,0)=fp1
  MFL     fp1=fp2,fp0,fcr   =b(i)*(1/x)
  BCT     ctr=CL.30
```

multiply by reciprocal of x

The IBM Rational C/C++ Café

ibm.com/rational/cafe/community/ccpp



Feature Request

- Request for a feature to be supported by our compilers
- C/C++ feature request page:

  http://www.ibm.com/support/docview.wss?uid=swg27005811
- Fortran feature request page:

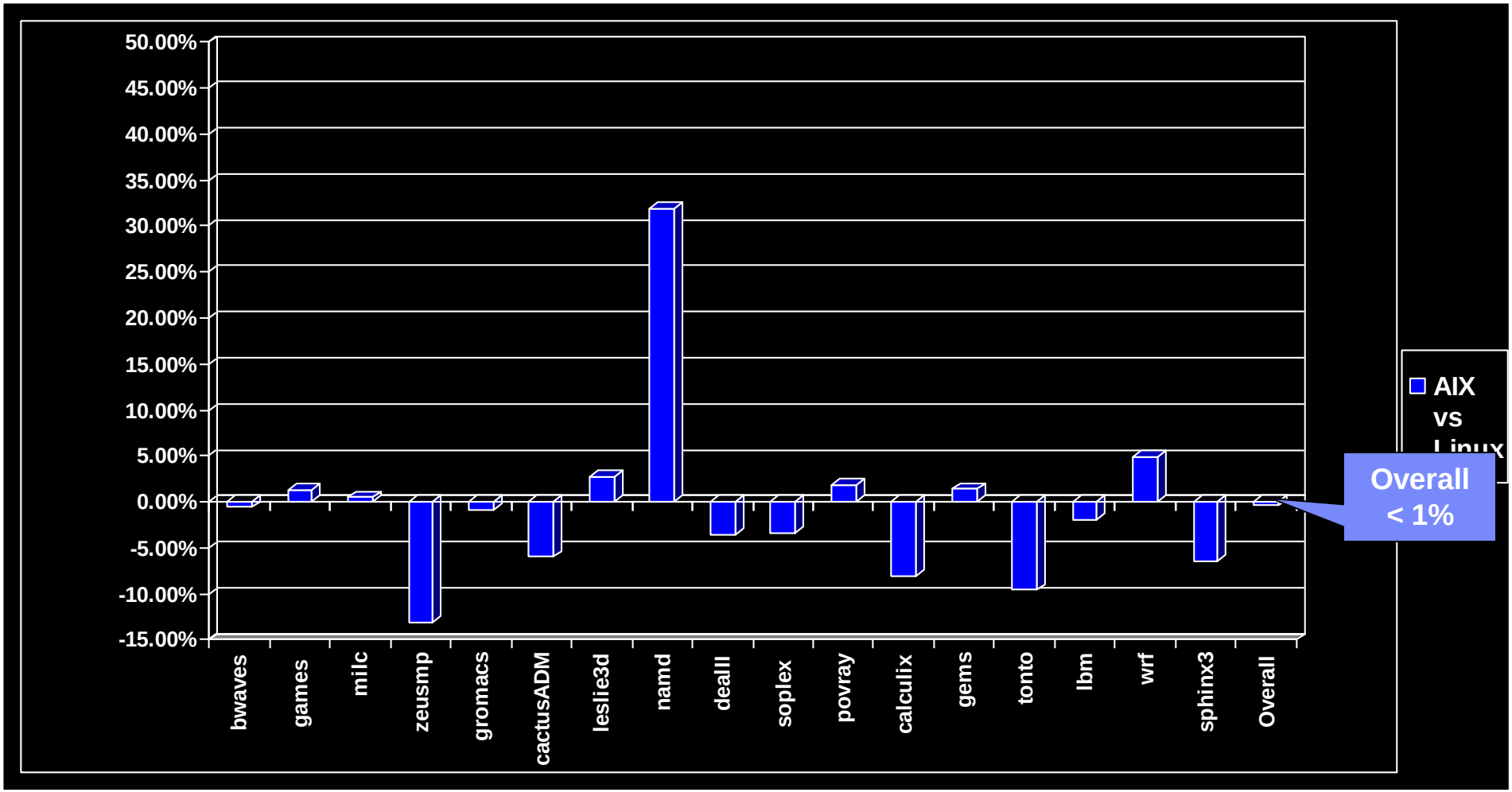
  http://www.ibm.com/support/docview.wss?uid=swg27005812
- Or send e-mail to **xl_feature@ca.ibm.com**

Documentation

- An information center containing the documentation for the **XL Fortran V12.1** and **XL C/C++ V10.1** versions of the AIX compilers is available at: **http://publib.boulder.ibm.com/infocenter/comphelp/v101v121/index.jsp**
- An information center containing the documentation for the **XL Fortran V11.1** and **XL C/C++ V9.0** versions of the AIX compilers is available at: **http://publib.boulder.ibm.com/infocenter/comphelp/v9v111/index.jsp**
- **Optimization and Programming Guide** for XLF V12.1 is now available online at: **http://publib.boulder.ibm.com/infocenter/comphelp/v101v121/index.jsp**
- New whitepaper “Overview of the IBM XL C/C++ and XL Fortran Compiler Family” available at: **http://www.ibm.com/support/docview.wss?uid=swg27005175**
- This information center contains all the html documentation shipped with the compilers. It is completely searchable.
- Please send any comments or suggestions on this information center or about the existing C, C++ or Fortran documentation shipped with the products to **compinfo@ca.ibm.com.**

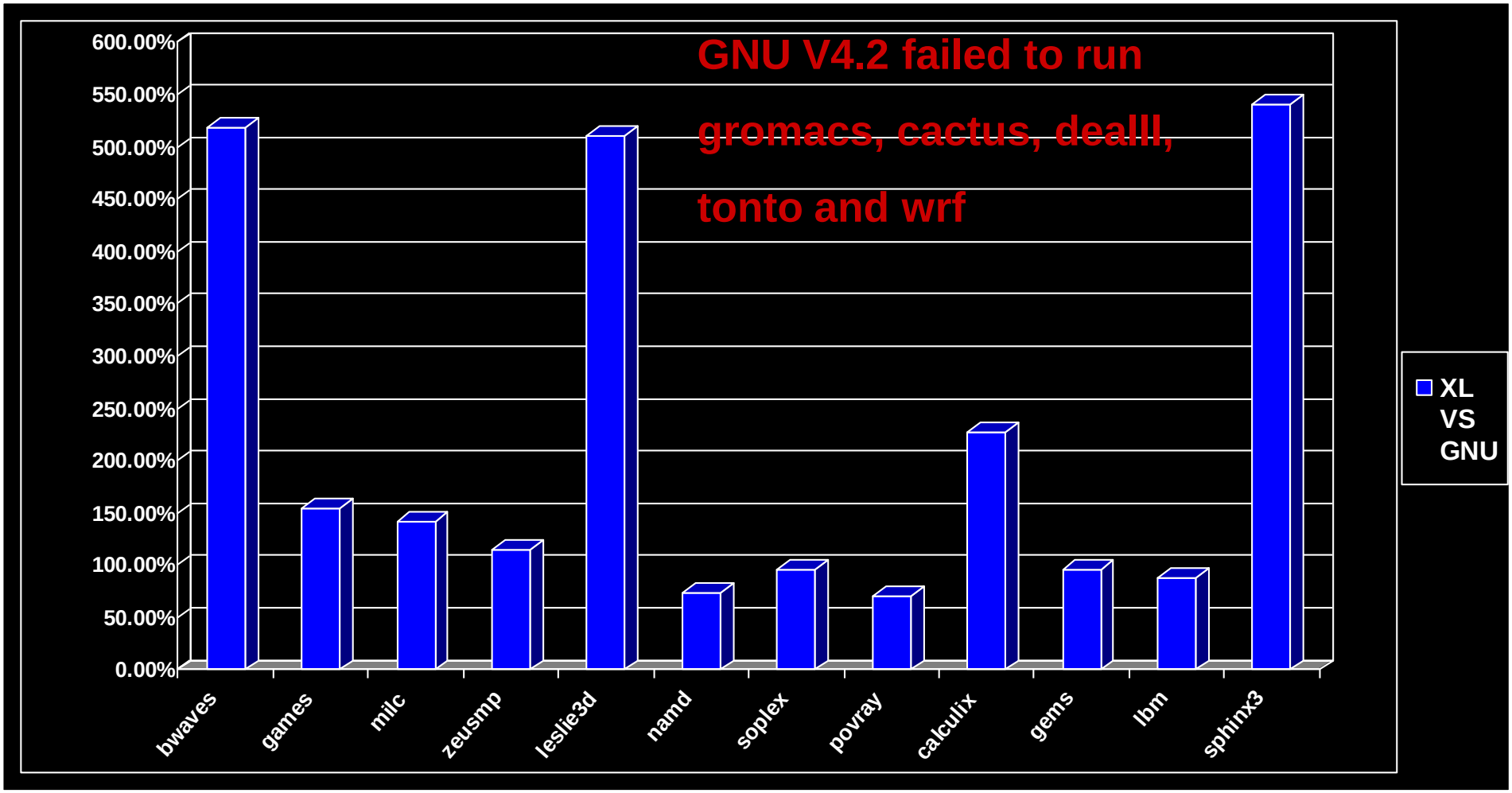
BACKUP SLIDES

SPEC2006 FP Comparison Between AIX and Linux on Power6



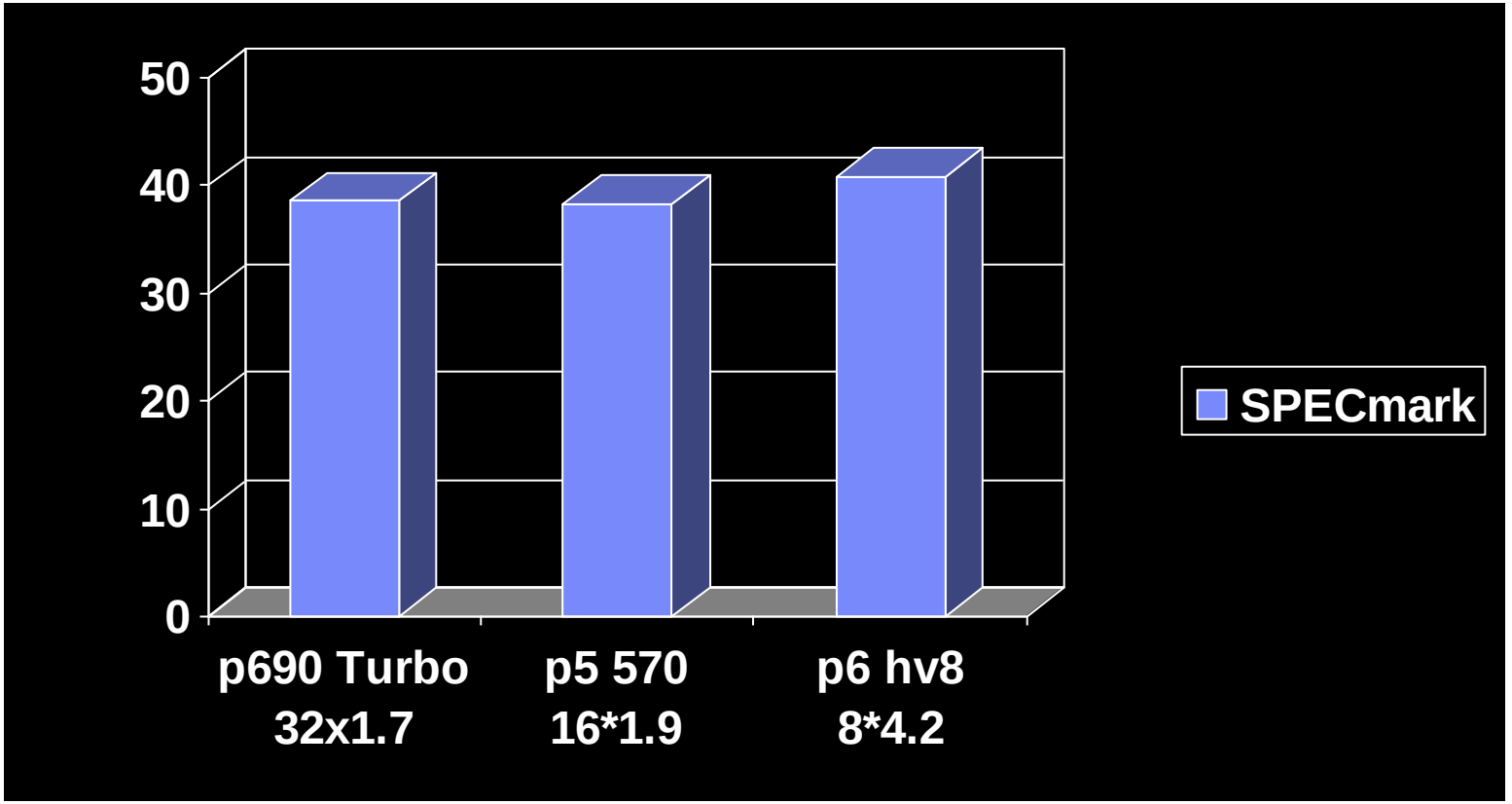
Using peak options from spec.org (positive means AIX is faster than Linux)

SPEC2006 FP Comparison Between XL Compilers and GNU Compilers on Power6

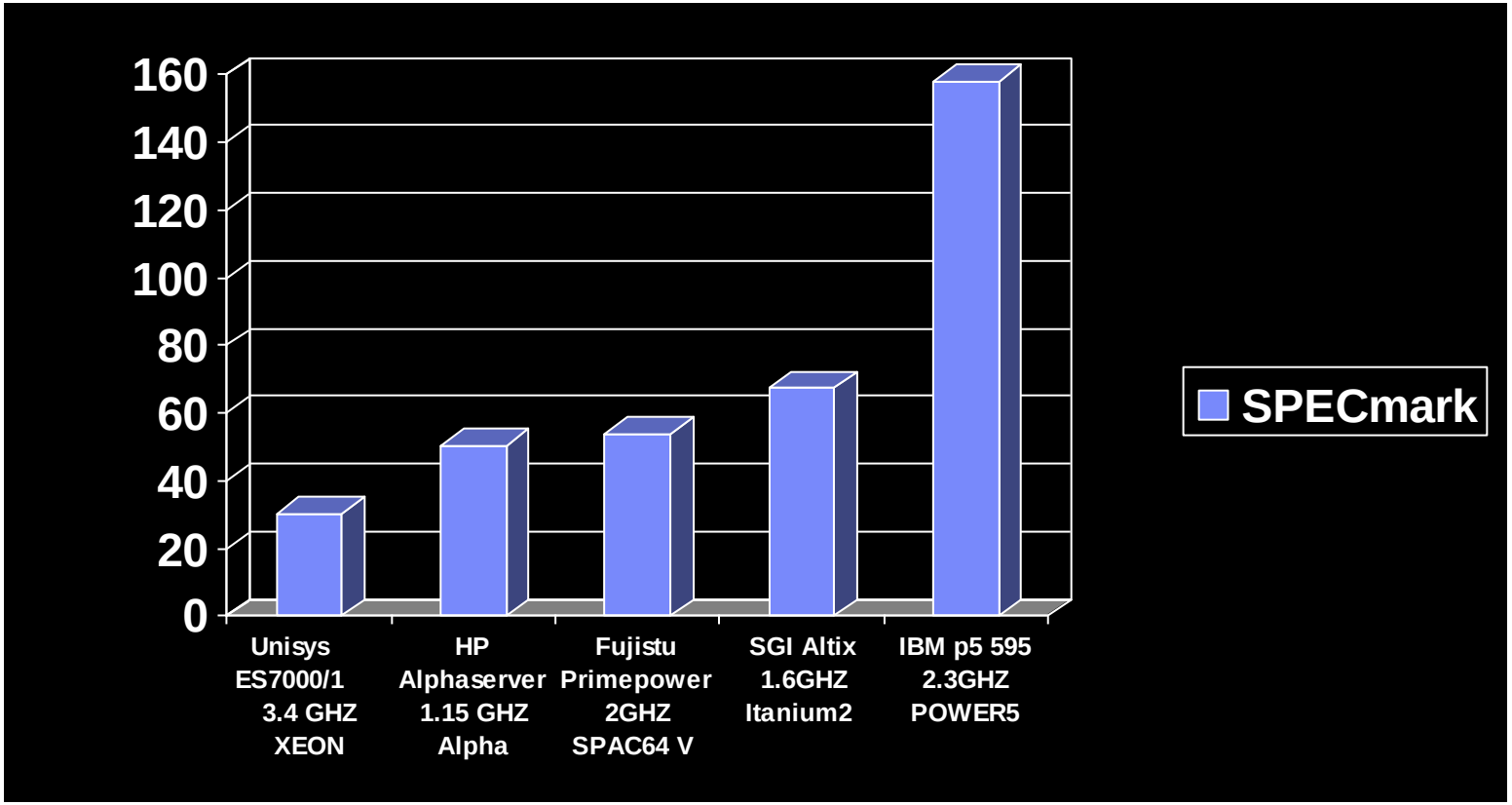


Using peak options with latest XL compilers and GNU compilers V4.2

SPECOMPM2001 Performance – IBM PPC Generations



SPECOMPM2001 Performance – 64-way Competition



History Of Compiler Improvement On Power4

Compilers	2001 V5/V7.1.1	2002 V6/V8.1	2003 V6/V8.1.1	2004 V7/V9.1	2005 V8/V10.1	Compound Over 4 Years	CAGR Rate
SpecINT	baseline	21%	0%	3%	7%	34%	7.6%
SpecFLOAT	baseline	12%	5%	18%	5%	46%	9.9%

Note: SPEC2000 base options improvements from www.spec.org

History Of Compiler Improvement On Power5

Compilers	2004 V7/V9.1	2005 V8/V10.1	2007 V9/V11.1	Compound Over 3 Years
SpecINT	baseline	4.3%	6.4%	11%
SpecFLOAT	baseline	5.4%	1.8%	7.3%

Note: SPEC2000 base options improvements from www.spec.org

Installation of Multiple Compiler Versions

- Installation of multiple compiler versions is supported
- The vacppndi and xlfndi scripts shipped with VisualAge C++ 6.0 and XL Fortran 8.1 and all subsequent releases allow the installation of a given compiler release or update into a non-default directory
- The configuration file can be used to direct compilation to a specific version of the compiler

  Example: xlf_v8r1 –c foo.f

  May direct compilation to use components in a non-default directory
- Care must be taken when multiple runtimes are installed on the same machine (details on next slide)

Coexistence of Multiple Compiler Runtimes

- **Backward compatibility**
  - C, C++ and Fortran runtimes support backward compatibility.
  - Executables generated by an earlier release of a compiler will work with a later version of the run-time environment.
- **Concurrent installation**
  - Multiple versions of a compiler and runtime environment can be installed on the same machine
  - Full support in xlfndi and vacppndi scripts is now available
- **Limited support for coexistence**
  - LIBPATH must be used to ensure that a compatible runtime version is used with a given executable
  - Only one runtime version can be used in a given process.
  - Renaming a compiler library is not allowed.
  - Take care in statically linking compiler libraries or in the use of *dlopen* or *load* .
  - Details in the compiler FAQ
    http://www.ibm.com/software/awdtools/fortran/xlfortran/support/
    http://www.ibm.com/software/awdtools/xlcpp/support/

A Unified Simdization Framework