

# LLVM/clang on the BG/Q

Hal Finkel

Argonne National Laboratory

March 21, 2012

# Why LLVM?

LLVM is used as the compiler backend for many tools and languages.

- clang (a good modern C++ frontend)
- clang's static analyzer
- Address Sanitizer (and Thread Sanitizer)
- JIT (OpenJDK, PyPy, Mono, Julia, etc.)
- OpenCL: clang + (libclc or pocl or clover)
- Many others, see: <http://llvm.org/Users.html> and <http://llvm.org/ProjectsWithLLVM/>

LLVM and clang have been ported to the BG/Q

- A2 instruction scheduling in LLVM
- QPX support in LLVM
- Basic-Block Autovectorization in LLVM (accepted upstream)
- QPX intrinsics in clang
- Some improvements to the PowerPC backend

# Where Is It?

There are now wrapper scripts:

- `/home/projects/llvm/bin/bgclang`
- `/home/projects/llvm/bin/bgclang++`

These are not in their final form. Currently:

- Autovectorization is turned on with `-O3` (set `BGCLANG_VECTORIZE=no` to turn this off)
- Static linking is turned on by default (set `BGCLANG_STATIC_LINKING=no` for dynamic linking)

In the future, these will turn into command-line options.

```
#if !defined(__xlc__) && !defined(__xlc__)  
#include <qpxintrin.h>  
#endif
```

This header file defines the IBM-style QPX intrinsics (vector4double, vec\_ld, vec\_add, etc.) in terms of the clang QPX intrinsics. All QPX intrinsics are supported. Also, this header file defines \_\_dcbt so that you can use the same prefetch intrinsic with both clang and the IBM compilers.

In my personal experience...

On simple vectorizable loops:

- LLVM/clang code is 2-3x faster than code from IBM's gcc port
- xlc code is 0-3x faster than the LLVM/clang code (LLVM/clang is rarely faster)

On heavily-templated C++ code:

- LLVM/clang code is often 0-30% faster than xlc++ code