LLVM and Clang on the Most Powerful Supercomputer in the World

Hal Finkel

November 7, 2012

The 2012 LLVM Developers' Meeting

What is Argonne National Laboratory

Argonne National Laboratory (ANL), located just outside of Chicago, is one of the U.S. Department of Energy's largest national laboratories for scientific and engineering research.

We have over 1,250 scientists and engineers, and over 300 postdoctoral researchers, in 13 research divisions and 6 national scientific user facilities.

What is the Argonne Leadership Computing Facility

- The Argonne Leadership Computing Facility (ALCF) is one of two leadership computing facilities supported by the U.S. Department of Energy (DOE).
- The ALCF provides the computational science community with computing capability dedicated to breakthrough science and engineering targeting at-scale run campaigns.
- The ALCF is now home to Mira, a 10-petaflop IBM Blue Gene/Q system with 49,152 compute nodes. We also continue to operate the Intrepid, a 557-teraflop IBM Blue Gene/P system.

The Blue Gene/Q (BG/Q)

ALCF's Mira BG/Q system:

The Blue Gene/Q (BG/Q)

The Top500 List of Supercomputing Sites

TOP500 List - June 2012 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

next

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom / 2011 IBM	1572864	16324.75	20132.66	7890.0
2	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
3	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	786432	8162.38	10066.33	3945.0
4	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR / 2012 IBM	147456	2897.00	3185.05	3422.7

The Green500 List of Supercomputing Sites

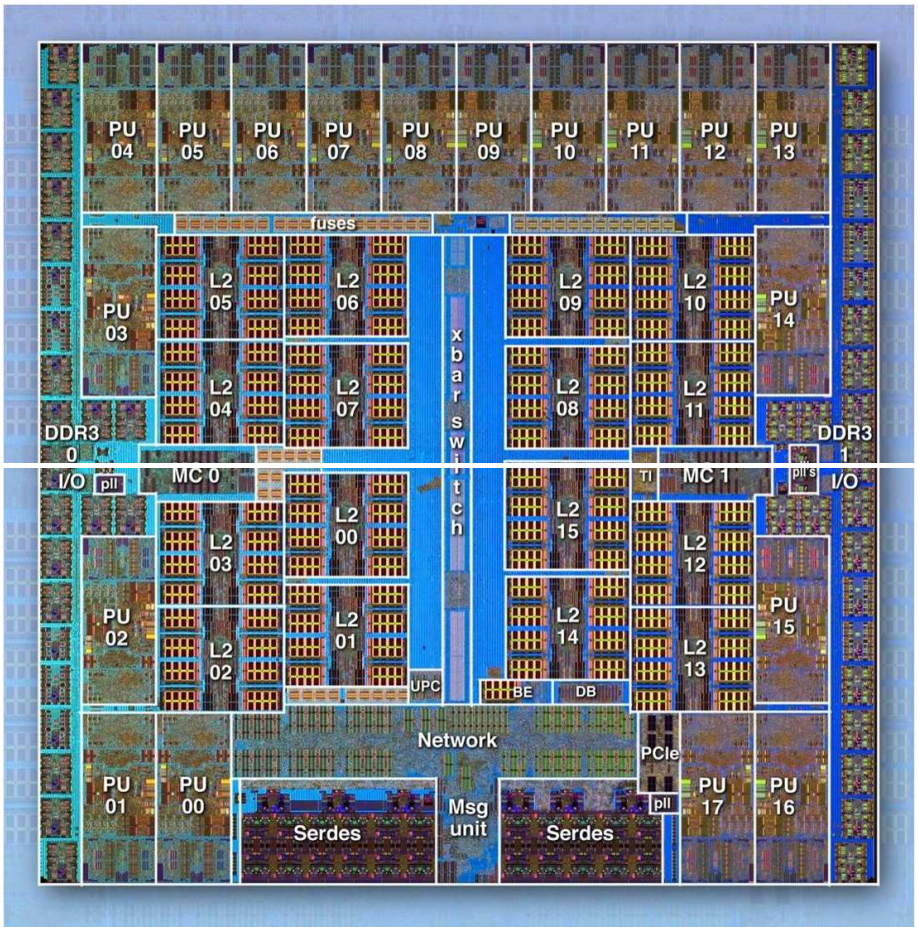
The Green500 List

Listed below are the June 2012 The Green500's energy-efficient supercomputers ranked from 1 to 10.

Green500 Rank	MFLOPS/W	Site*	Computer*
1	2,100.88	DOE/NNSA/LLNL	BlueGene/Q, Power BQC 16C 1.60GHz, Custom
2	2,100.88	IBM Thomas J. Watson Research Center	BlueGene/Q, Power BQC 16C 1.60GHz, Custom
3	2,100.86	DOE/SC/Argonne National Laboratory	BlueGene/Q, Power BQC 16C 1.60GHz,

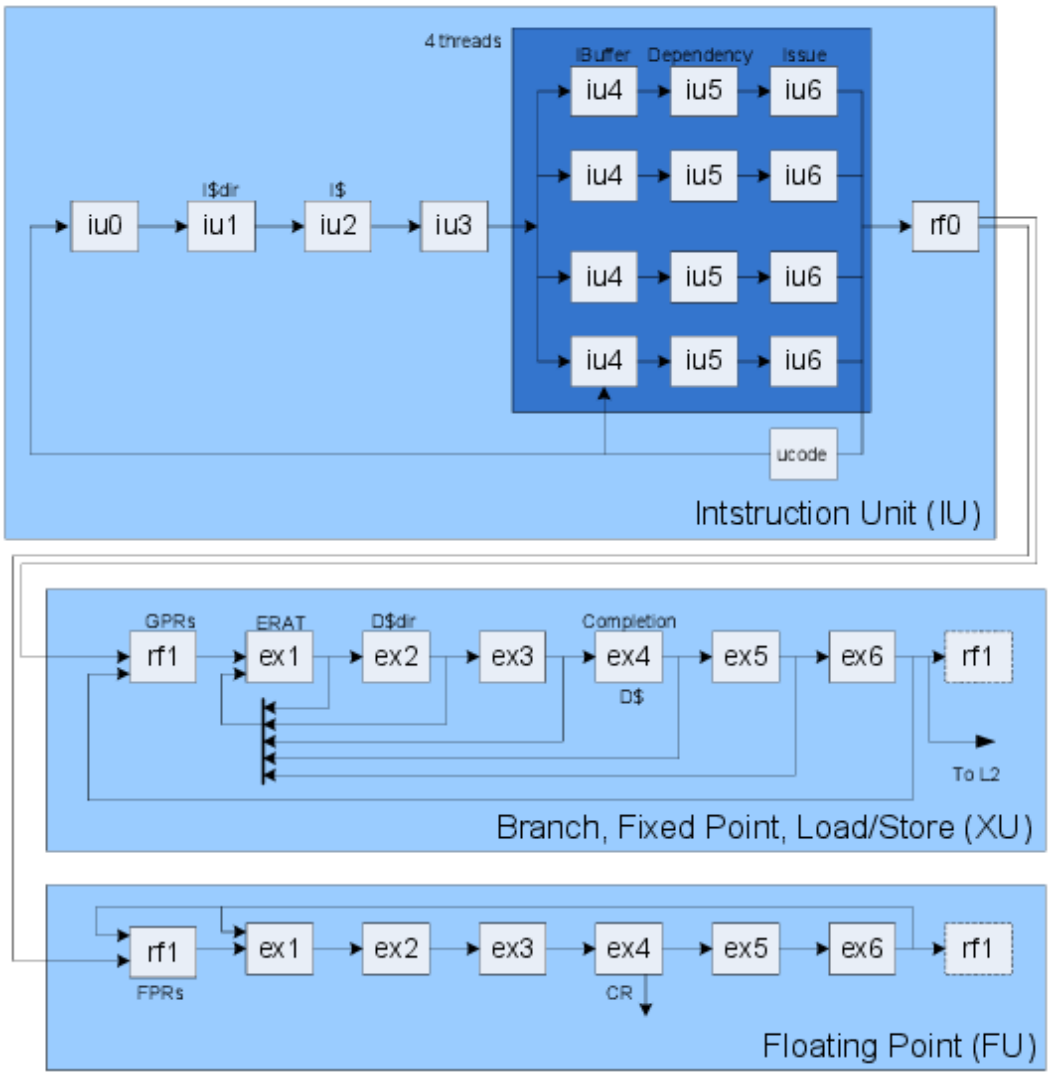
(entries 1 through 20 are all BG/Qs; in 21^{st} place is an Intel Xeon/MIC cluster @ 1,380.67 MFLOPS/W)

The BG/Q Node



- 16 user-accessible cores, 1 system core, and 1 spare @ 1.6 GHz (204.8 peak GFLOPS)
  - 64-bit PowerPC ISA + 4-way double-precision SIMD
  - 16 kB L1 D-cache, 16 kB L1 I-cache
- 32 MB shared L2 cache
  - multiversioned – supports transactional memory and speculative execution
  - built in support for atomic operations
- Dual memory controllers
  - 16 GB external 1.333 GHz DDR3
  - 42.6 GB/s DDR3 bandwidth
- Network with 5D Torus topology
  - 2 GB/s send and 2 GB/s receive
  - DMA with remote put/get and collective operations

The A2 Core



The embedded 64-bit A2 core:

- Two pipelines, one for floating point, one for everything else
- Four hardware threads; one instruction dispatched to each pipeline at each cycle from different threads
- Most instructions, including L1 load and store, don't stall

QPX

$QPR0^0$	$QPR0^1$	$QPR0^2$	$QPR0^3$
$QPR1^0$	$QPR1^1$	$QPR1^2$	$QPR1^3$
...			
...			
$QPR30^0$	$QPR30^1$	$QPR30^2$	$QPR30^3$
$QPR31^0$	$QPR31^1$	$QPR31^2$	$QPR31^3$

0 63 64 127 128 191 192 255

- Quad-Vector Floating-Point (QPX) extends the regular PowerPC floating-point registers to vectors of four: The scalar floating-point registers alias the first element of each corresponding vector register
- A scalar floating-point load splats the loaded value into all vector elements
- Single-precision floating point is supported via instruction variants that round to single precision

QPX Instructions

- vector loads and stores, both single-precision and double-precision. Loading and storing of two "complex" values
- general permutations, alignment-based permutations (to support unaligned accesses), subvector extraction and splat
- vector loads and stores of integer values, float/int conversions (but no other integer operations)
- negate, abs, negative abs, copy sign, rounding (to single precision, to integer in various ways)
- add, sub, mul, recip. est., recip. sqrt est.
- mul-add, mul-sub, negative mul-add, negative mul-sub, various cross mul-adds for complex arithmetic
- compare less-than, compare greater-than, compare equal, test for NaN
- generalized boolean operations, vector select

QPX Booleans

In QPX, floating point values are also booleans:

- Values greater than or equal to ± 0.0 are true
- Values less that 0.0 or NaN are false
- Instructions that produce booleans produce ± 1.0

Instead of providing specific boolean operations, one instruction is provided that takes an arbitrary truth table:

Extended Mnemonic		Equivalent		Function
qvfclr	QRT	qvflogical	QRT,QRT,QRT,0	clear (set as FALSE)
qvfand	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,1	and
qvfandc	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,4	and complement B
qvfctfb	QRT,QRA	qvflogical	QRT,QRA,QRA,5	convert to float-boolean A
qvfxor	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,6	xor
qvfor	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,7	or
qvfnor	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,8	nor
qvfequ	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,9	Boolean equivalent (XNOR)
qvfnot	QRT,QRA	qvflogical	QRT,QRA,QRA,10	not
qvforc	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,13	or complement B
qvfnand	QRT,QRA,QRB	qvflogical	QRT,QRA,QRB,14	nand
qvfset	QRT	qvflogical	QRT,QRT,QRT,15	set (set as TRUE)

Why?

- Provide a high-performance **and** up-to-date C/C++ compiler over the lifetime of the machine (the fact that Clang has excellent diagnostics and a static-analysis framework makes this even more appealing)
- Provide access to other languages that use LLVM has a backend (such as Intel's ISPC, and various scripting languages)
- Provide a platform for compiler research supporting the BG/Q
  - Autovectorization
  - Parallelization (including transactional memory and speculative execution)
  - Communication-related optimizations and distributed systems

Tagged-Type Diagnostics

The tagged-type diagnostics, developed by Dmitri Gribenko, are an important new feature in Clang that will greatly benefit the HPC community. To my knowledge, no other compiler can produce these kinds of warnings, and these will be extremely valuable to our users.

```
-bash-4.1$ /home/projects/llvm/mpi/clang/bin/mpicc -Wall mpit2.c
mpit2.c:18:17: warning: argument type 'char *' doesn't match specified 'MPI'
      type tag that requires 'double *' [-Wtype-safety]
  rc = MPI_Send(&outmsg, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
                ^~~~~~~     ~~~~~~~~~~
1 warning generated.
```

The wider community can also benefit by applying the relevant attributes to, for example, some POSIX APIs.

The "Easy" Parts

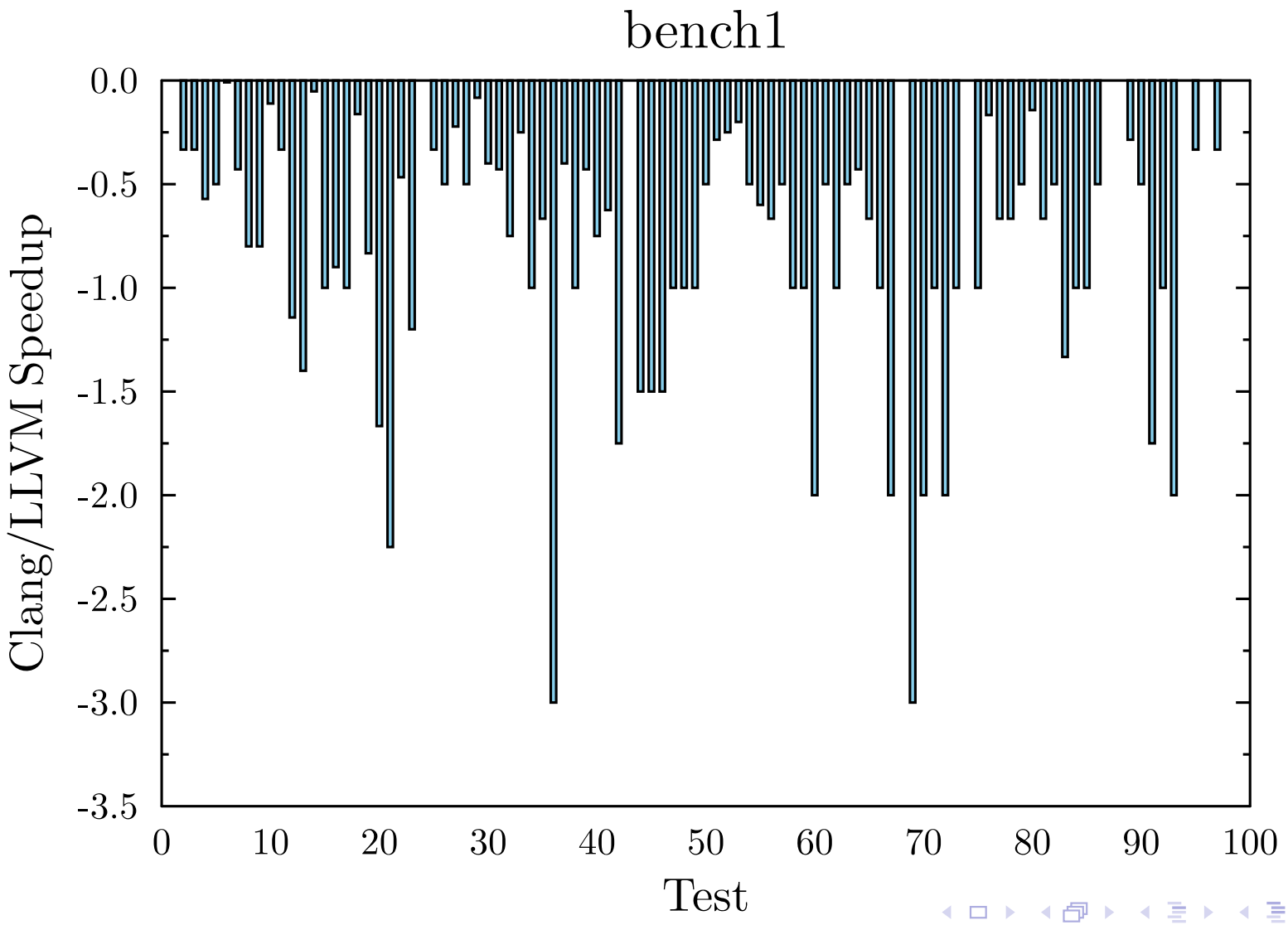
- Modifications to the PowerPC backend: Register definitions, calling conventions, declaring instruction legality, patterns for basic arithmetic operations, loads and stores, etc.
- Developing an itinerary for the A2 core based on IBM's documentation, and making associated modifications to the subtarget code
- Adapting the Hexagon hardware loops pass to do the equivalent thing for PowerPC
- Adding support for QPX intrinsics in Clang (and developing a header file compatible with the corresponding vendor-compiler intrinsics)

The More-Difficult Parts

- Developing a basic-block autovectorizer (see the presentation from the 2012 Euro-LLVM conference, and come to the BoF later at this meeting)
- Modifications to the SelectionDAG builder to loosen the critical-chain restrictions
- Adding support for v4i1 booleans to support QPX logical operations (still in progress)
- Cleaning up, modernizing and fixing bugs in the PowerPC backend (IBM has recently started to help with this)
- Fixing bugs elsewhere in LLVM exposed by the new target code

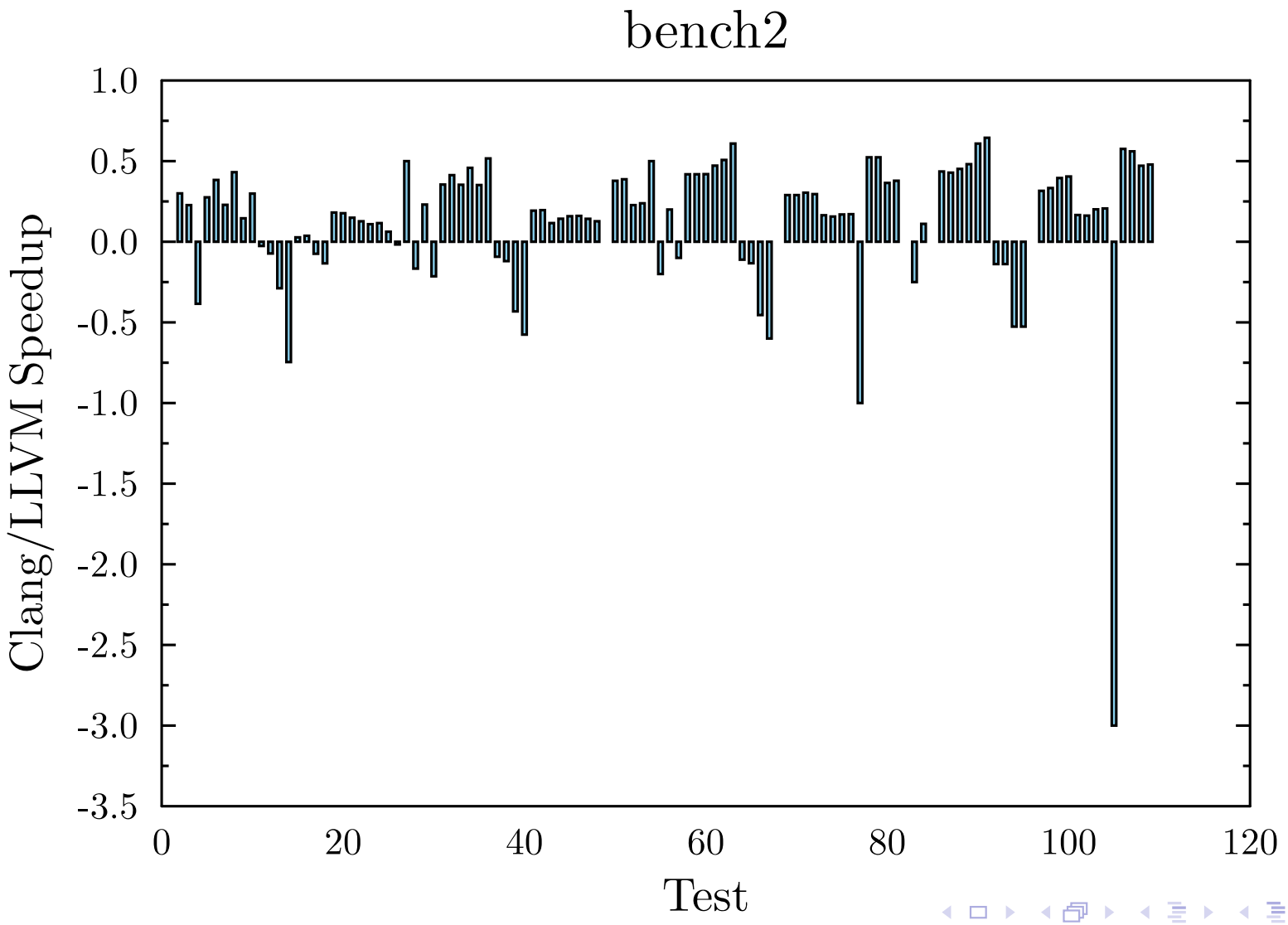
Boost uBlas: Benchmark 1

Dense matrix and vector operations:



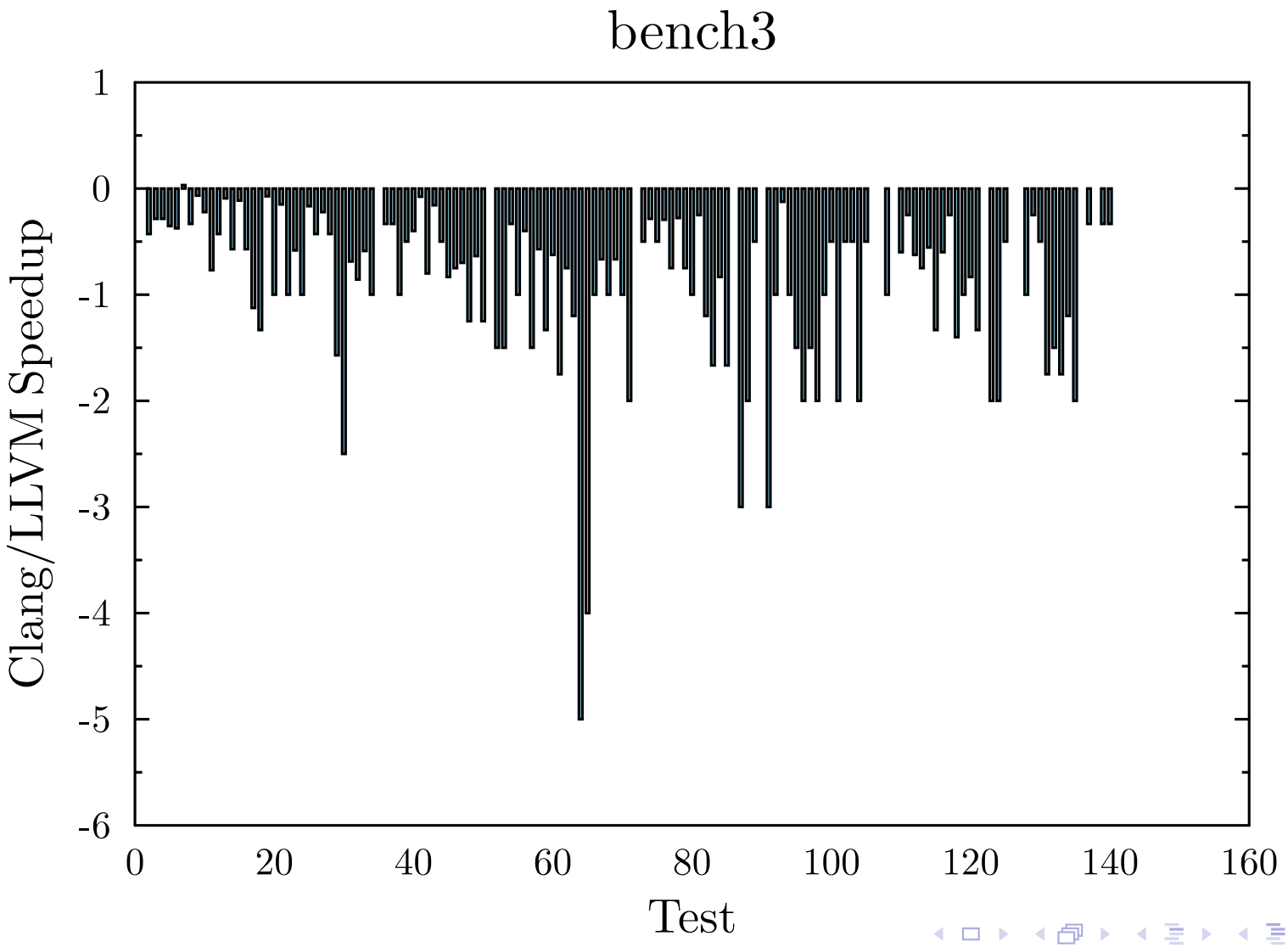
Boost uBlas: Benchmark 2

Sparse matrix and vector operations:



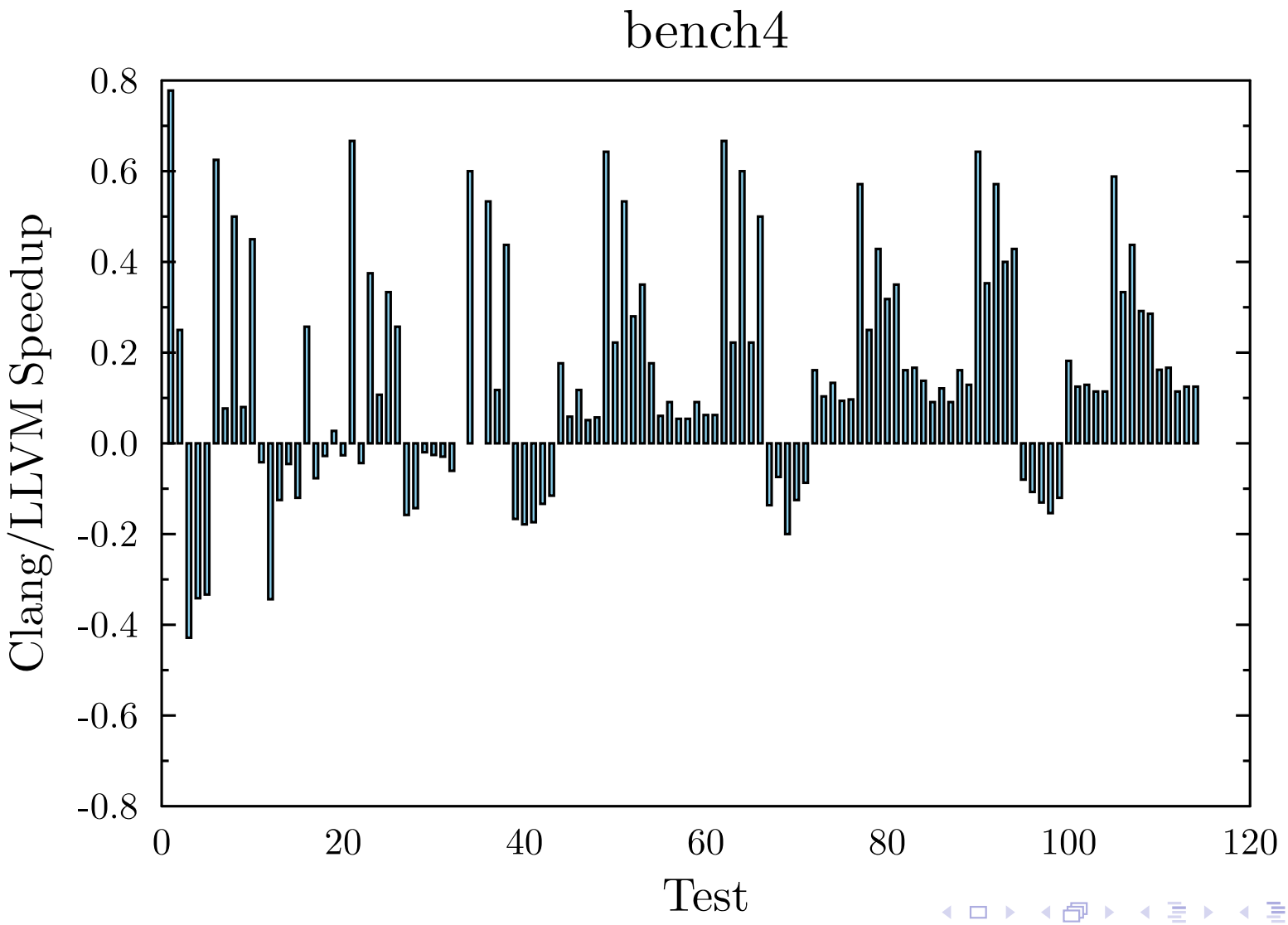
Boost uBlas: Benchmark 3

Vector and matrix proxy's operations:



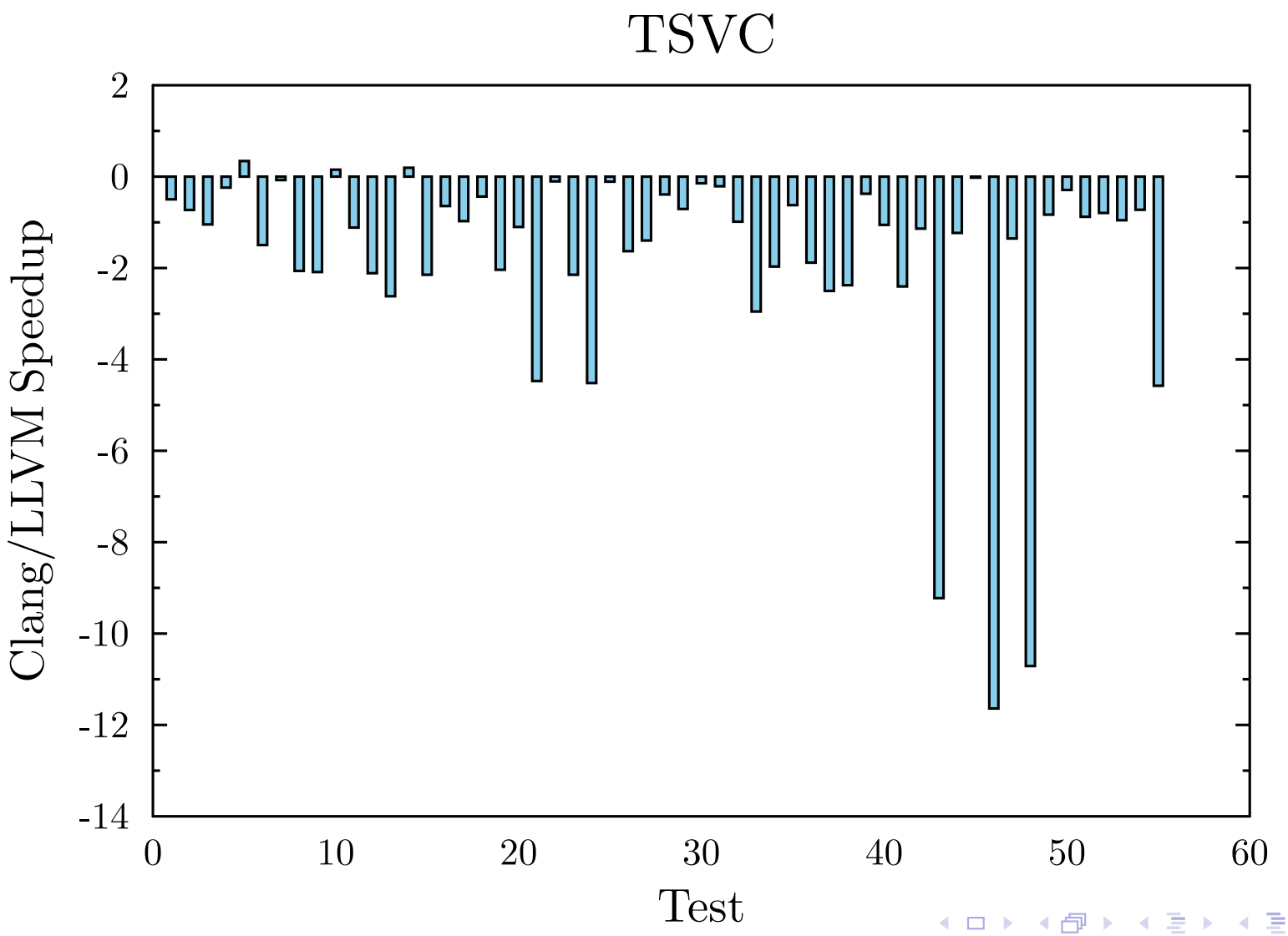
Boost uBlas: Benchmark 4

Dense matrix and vector operations with boost::numeric::interval:



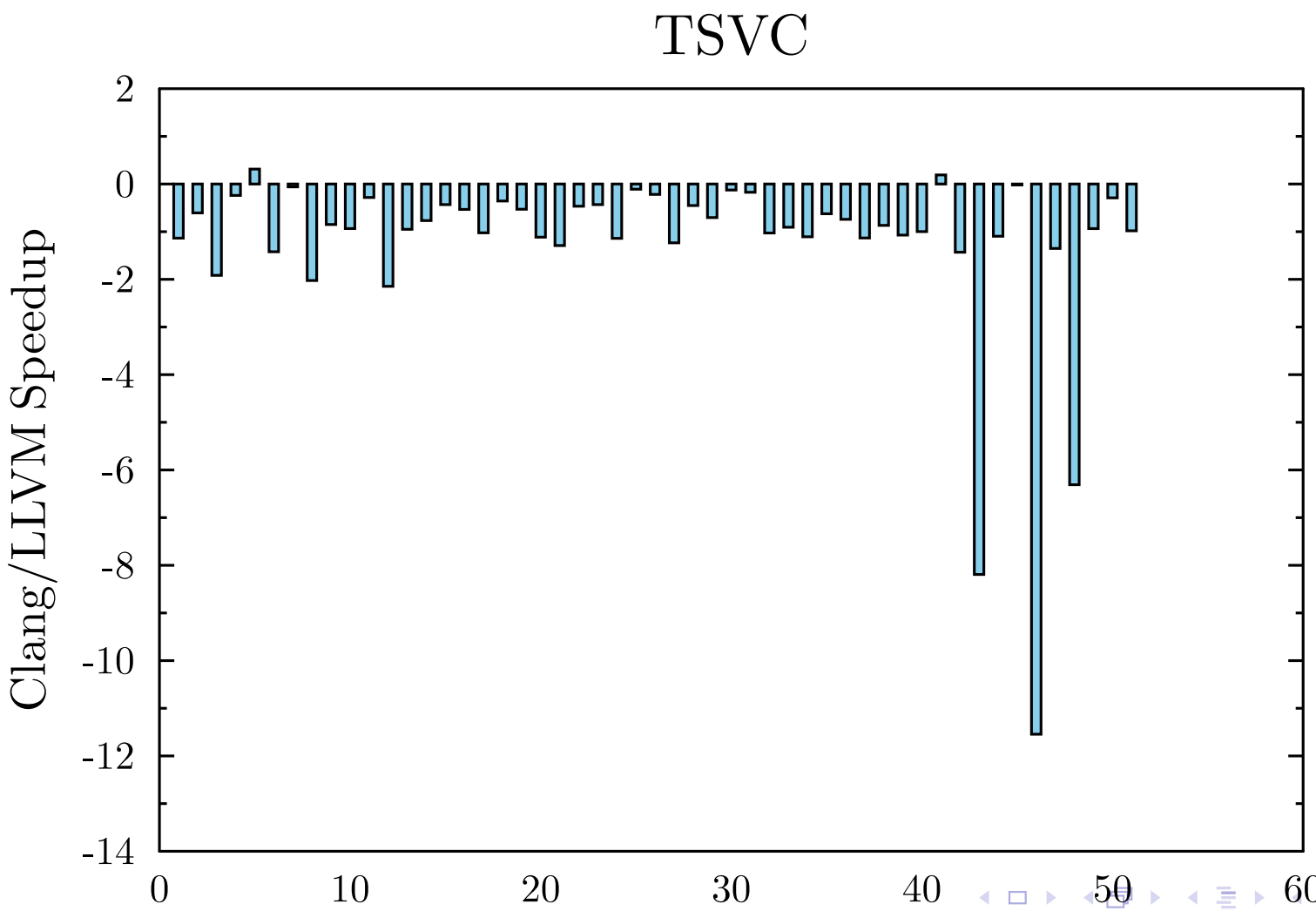
TSVC

The first few tests from the TSVC autovectorization benchmark:



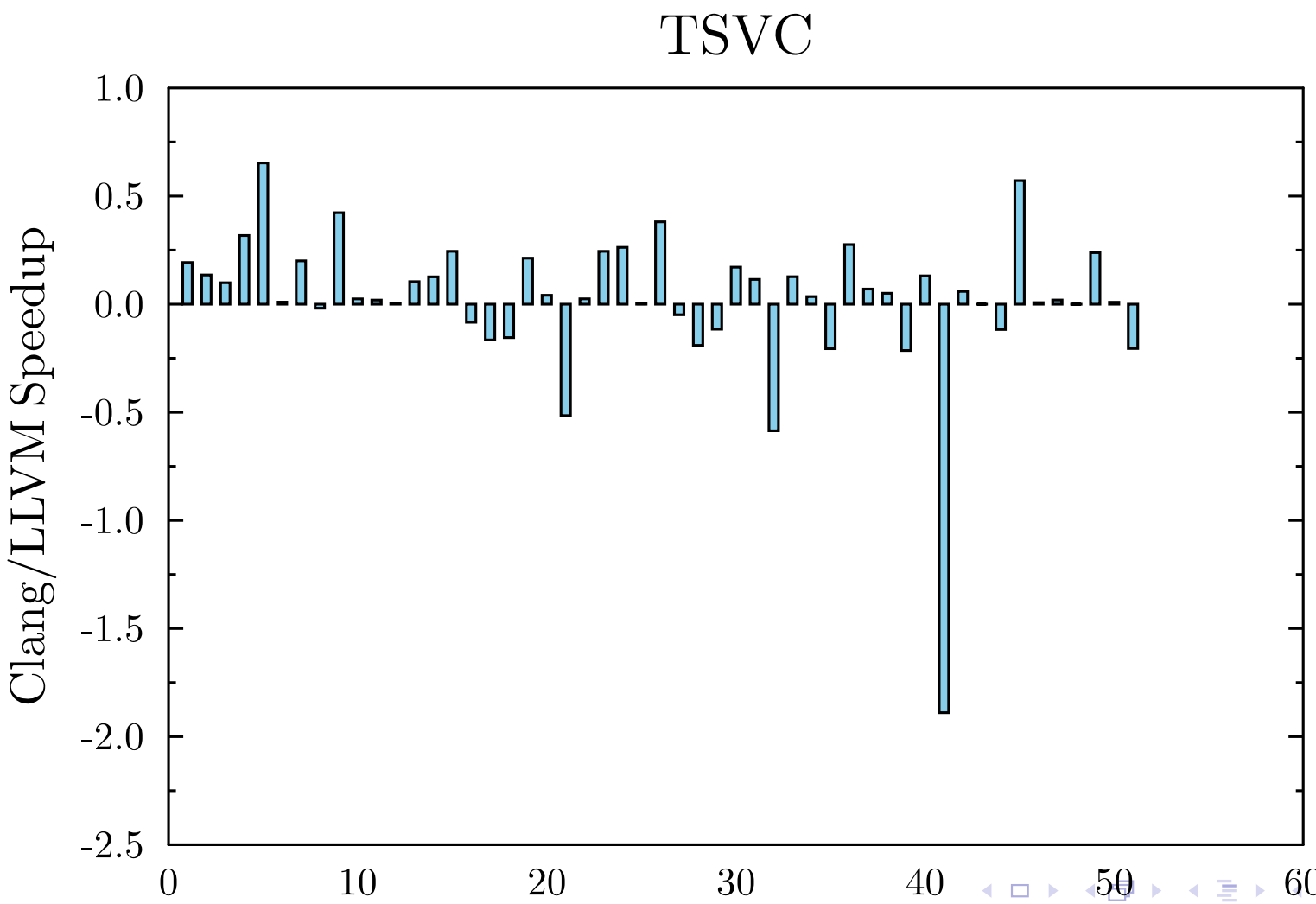
TSVC

The first few tests from the TSVC autovectorization benchmark (without vectorization):



TSVC

The first few tests from the TSVC autovectorization benchmark (without vectorization, compared to gcc for the BG/Q):



Future Work

In addition to general LLVM work:

- Full support for generating all QPX instructions (will require further enhancement to the BB vectorizer and the loop vectorizer, more target-level DAG combiner enhancements, and support for the single-precision-rounded variants) and support for vendor-supplied vectorized math libraries
- More PowerPC backend enhancements: For example, better handling of condition registers (especially how they are spilled)
- **Parallelization support** (especially, but not limited to, OpenMP)
- Higher level loop transformations (using Polly)
- MPI-specific optimizations

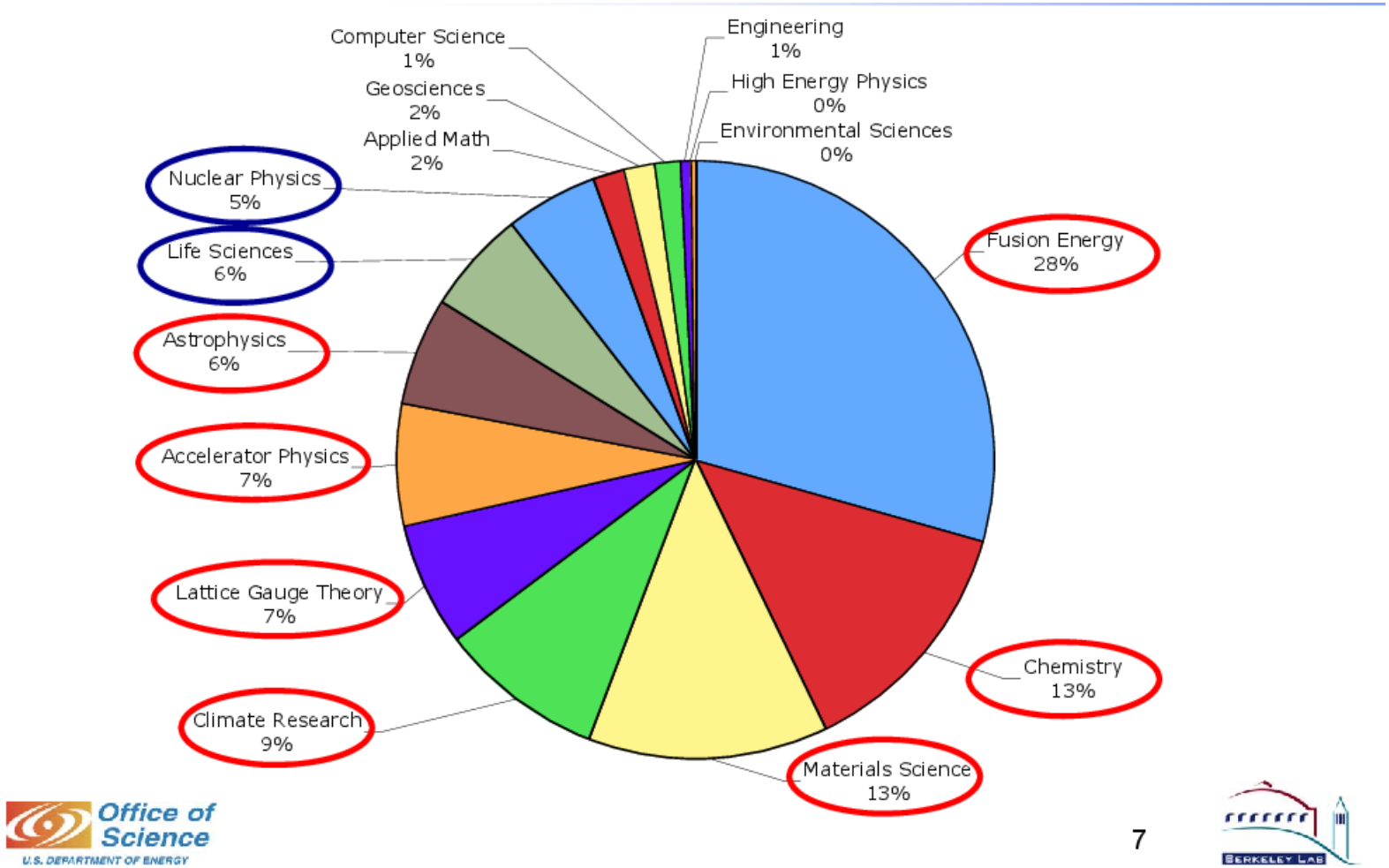
Make LLVM (with Clang) a powerful force in HPC!

Acknowledgements

From idea to production-stable, high performance, autovectorizing compiler for the BG/Q is a long road (and we're not quite there yet), and I've had a lot of help:

- Roman Divacky, Tobias von Koch, and others who have contributed to the PowerPC backend (now including several people from IBM's LTC: Bill Schmidt, Will Schmidt, Ulrich Weigand, Adhemerval Zanella, and Peter Bergner)
- Tobi Grosser, Nadav Rotem, and others who have helped with development of the vectorizer
- Andy Trick, Jakob S. Olesen (for answering many questions on scheduling, register allocation, etc.)
- The LLVM and Clang development communities
- ALCF, ANL and DOE

Science Areas



(science areas at NERSC – from a 2009 presentation by John Shalf)