Exploiting automatic vectorization to employ SPMD on SIMD registers

Stefan Sprenger (sprengsz@informatik.hu-berlin.de)
Steffen Zeuch (steffen.zeuch@dfki.de)
Ulf Leser (leser@informatik.hu-berlin.de)

HardBD & Active '18
April 16, 2018

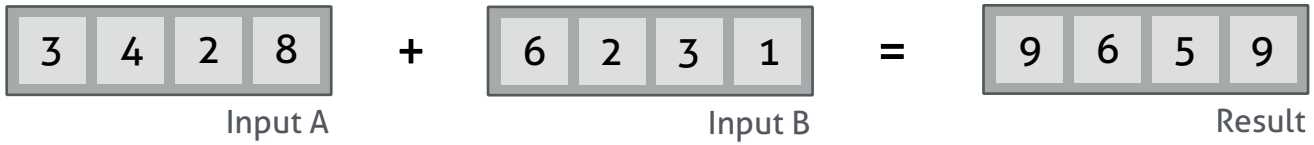
Agenda

- SIMD and SPMD
- Automatic Vectorization vs. Intrinsics
- Intel SPMD Program Compiler
- Case Study: Column Scan

Agenda

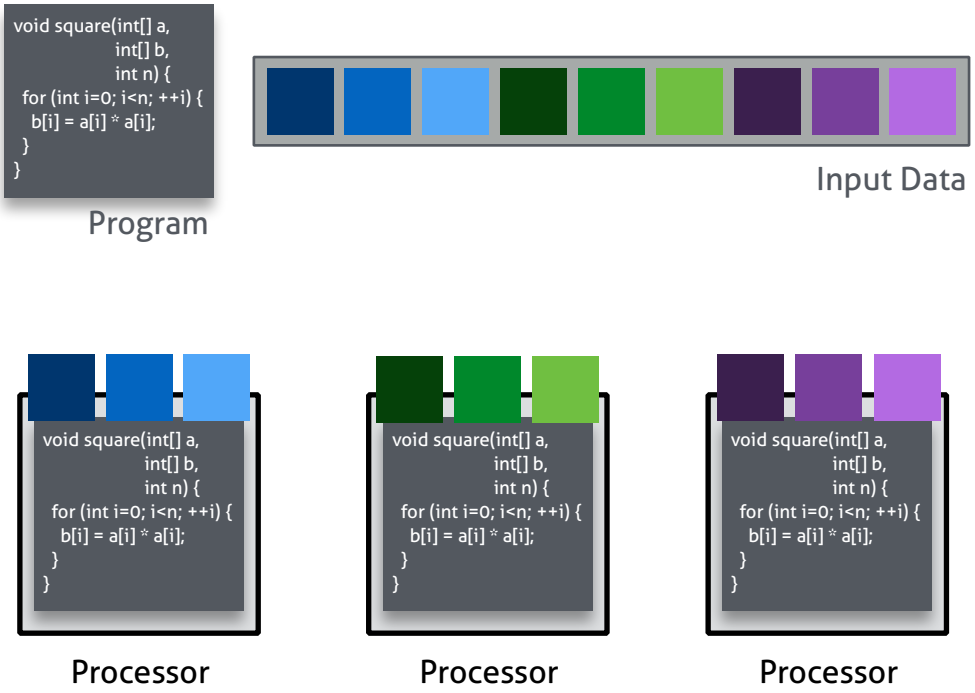
- SIMD and SPMD
- Automatic Vectorization vs. Intrinsics
- Intel SPMD Program Compiler
- Case Study: Column Scan

Single Instruction Multiple Data (SIMD)



- Process multiple data elements with one instruction
- Modern CPUs offer dedicated instructions executed on extra-wide registers
- Different instruction set architectures, e.g., SSE (128 Bits), AVX (256 Bits), AVX-512 (512 Bits)
- Degree of parallelism of a SIMD instruction depends on how many data elements fit into one register, e.g., eight 32-bit ints fit into one 256-bit register
- Developers can use SIMD instructions through intrinsics or rely on compiler-based automatic vectorization

Single Program Multiple Data (SPMD)



A single program that appears to be serial is deployed onto multiple independent processing units (processors).

The program instances are concurrently executed on different subsets of the data.

Agenda

- SIMD and SPMD
- Automatic Vectorization vs. Intrinsics
- Intel SPMD Program Compiler
- Case Study: Column Scan

Automatic Vectorization

- Recent versions of compilers support automatic vectorization
- For instance, they accelerate scalar for loops with SIMD instructions
- Works only for simple algorithms
- Lacks support of recent instruction set architectures
- Cannot compete with intrinsics code manually tuned by (experienced) developers

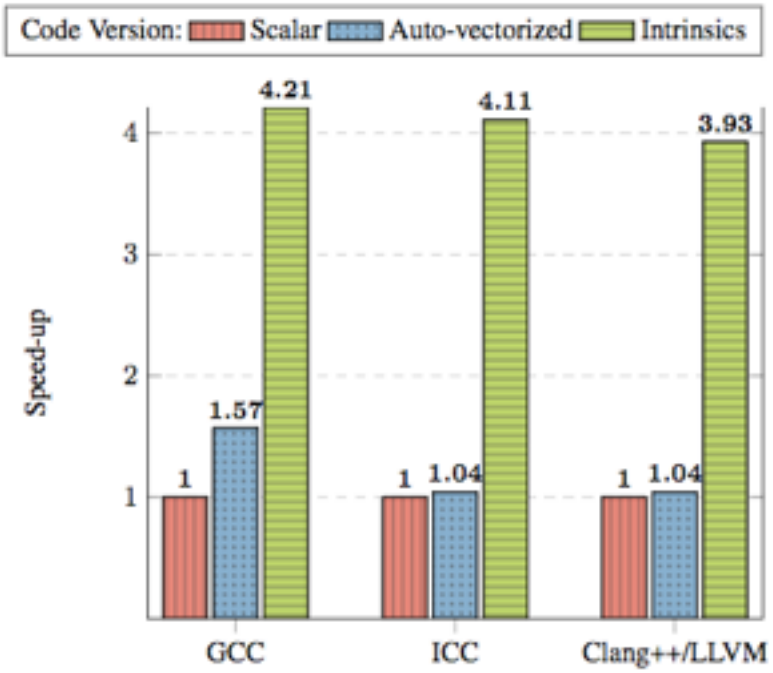


Figure 1. Speed-ups obtained with an auto-vectorized and an intrinsics-based implementation of a real-world HEVC video decoder , shown for the most popular C++ compilers *(4K resolution video decoding, 8 threads on an Intel i7-4770 core with AVX2)*

Figure taken from: Pohl et al.: "An Evaluation of Current SIMD Programming Models for C++" (WPMVP, 2016)

Limitations of SIMD Intrinsics

```
// Broadcast 32-bit floating-point value a to all elements of dst.
__m256 _mm256_set1_ps (float a);
```

- Require low-level hardware knowledge
- Specific to the underlying instruction set architecture, e.g., AVX
- Specific to the processed data type, e.g., float
- Result in hard-to-maintain code when supporting different hardware architectures or data types
- Forward compatibility

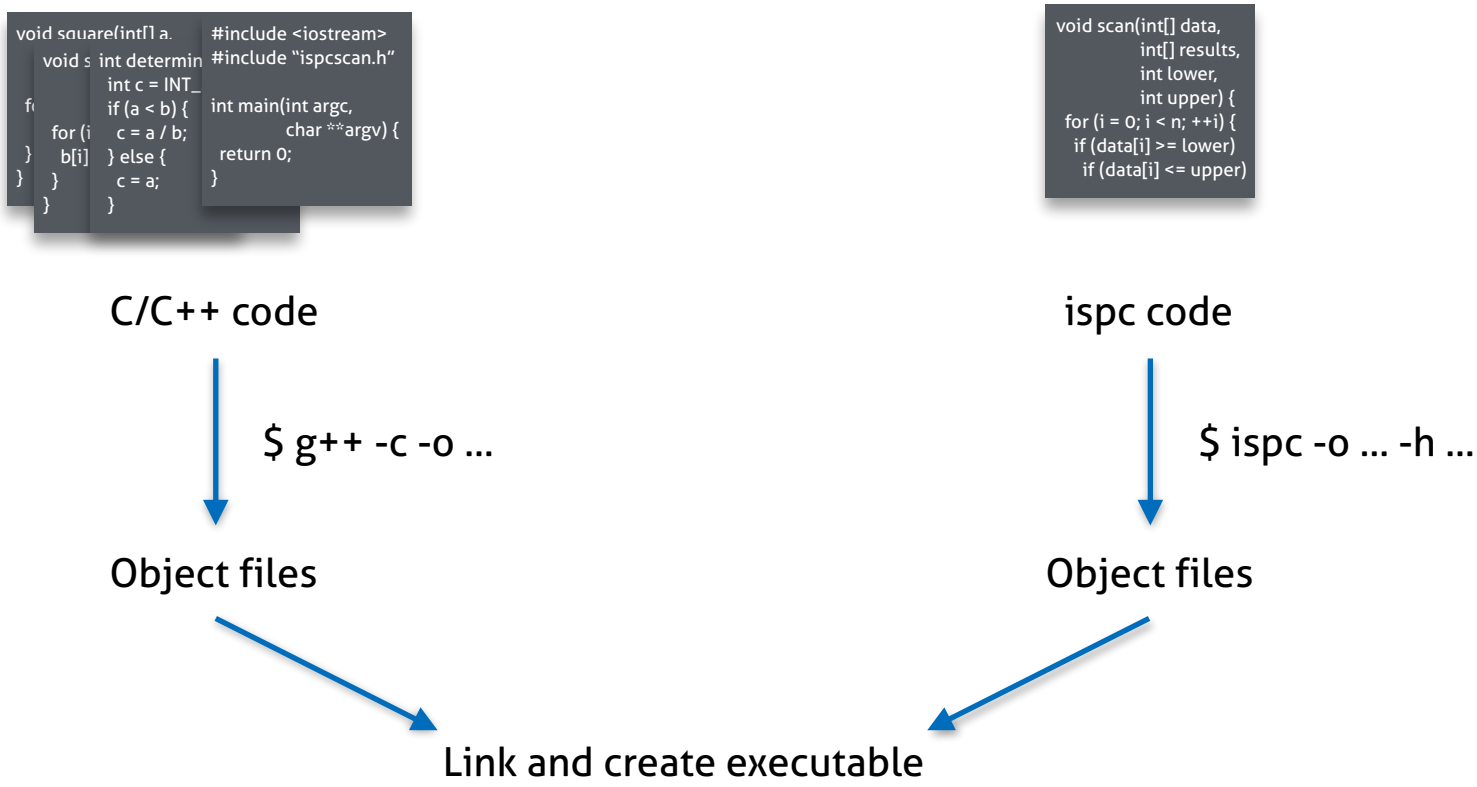
Agenda

- SIMD and SPMD
- Automatic Vectorization vs. Intrinsics
- Intel SPMD Program Compiler
- Case Study: Column Scan

Intel SPMD Program Compiler (ispc)

- Deploys the SPMD execution model on the SIMD registers of modern CPUs
- Program instances are mapped onto SIMD lanes
- Extension of the C programming language with few new features that facilitate writing high-performance SPMD programs
- Programs compiled with ispc can be directly called from C/C++
- Supports current CPU and instruction set architectures
  - x86, x86-64, Xeon Phi, ARM
  - SSE 2/4, AVX, AVX2, AVX512, NEON, ...
- Allows to use multi-threading in addition to SIMD parallelism

Integrating ispc into your C/C++ project



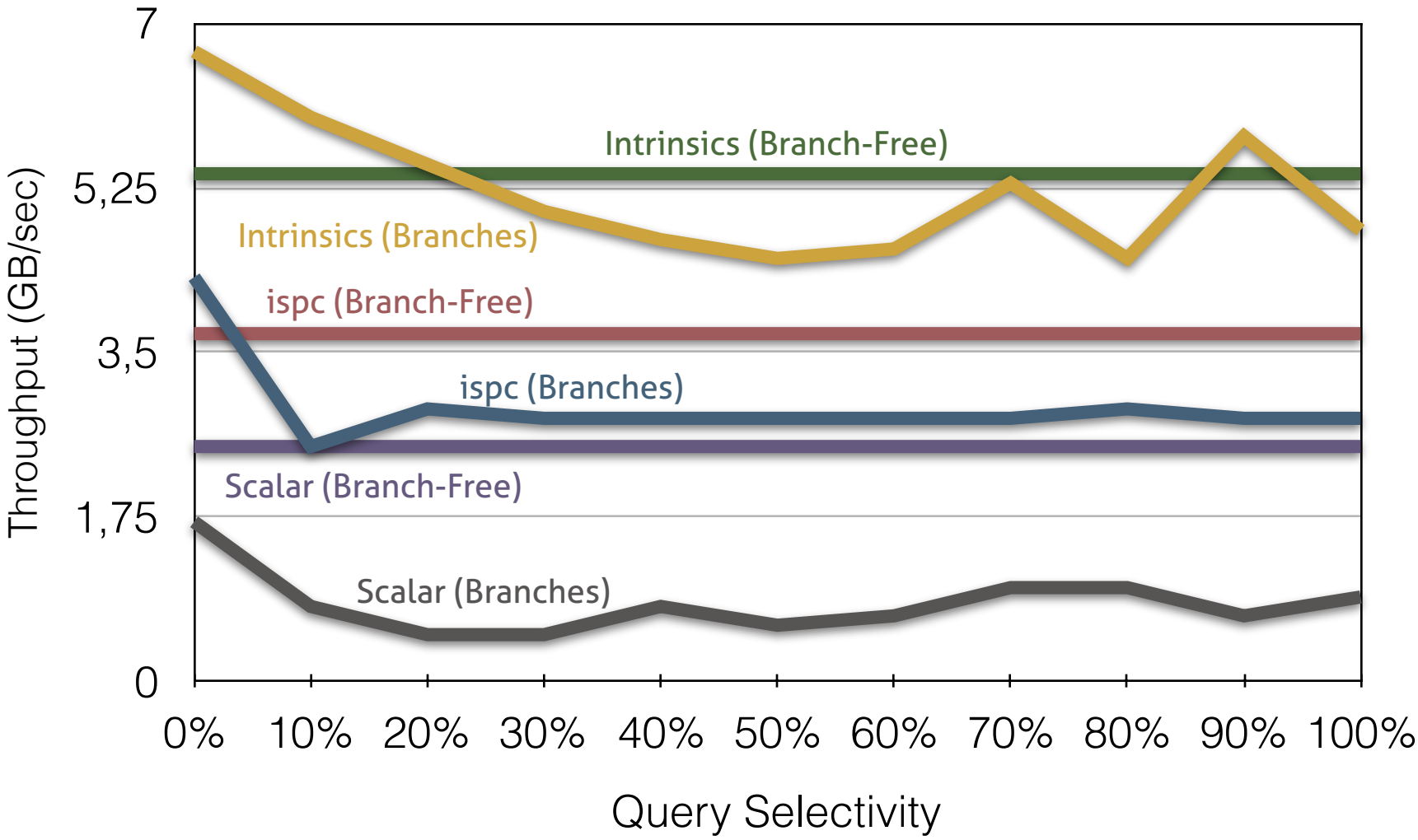
Agenda

- SIMD and SPMD
- Automatic Vectorization vs. Intrinsics
- Intel SPMD Program Compiler
- Case Study: Column Scan

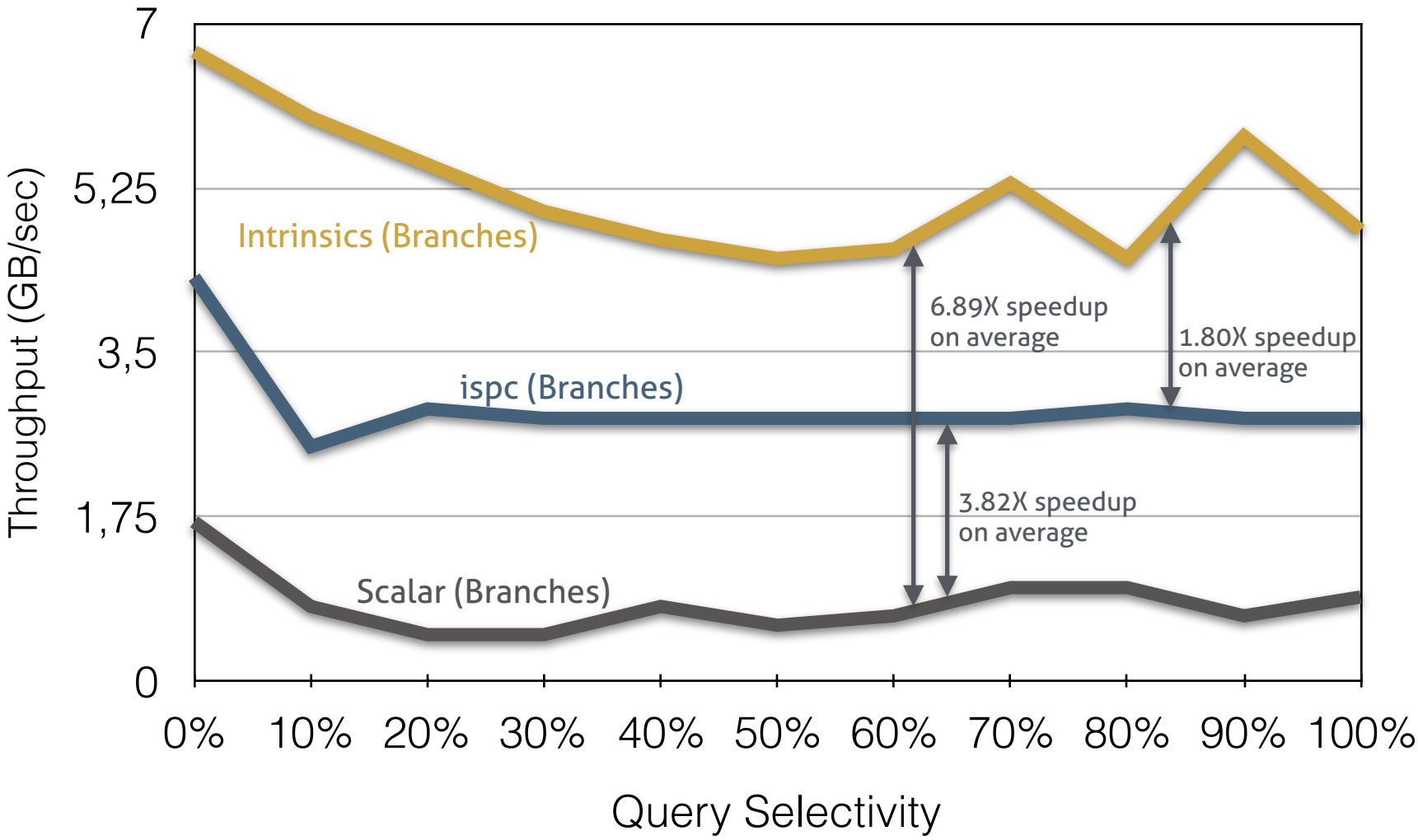
Experimental Setup

- Scalar, Intrinsics-based, and ispc-based column scan
- Branching and branch-free scan variants
- 1GB of synthetic keys generated with std::rand()
- Synthetic range scans of varying selectivity
  - lower bound: random, existing key
  - upper bound: lower bound + selectivity * domain
- Server machine equipped with Intel Xeon E5-2620 (2 GHz clock rate, 256-bit wide SIMD registers, AVX) and 32 GB of main memory

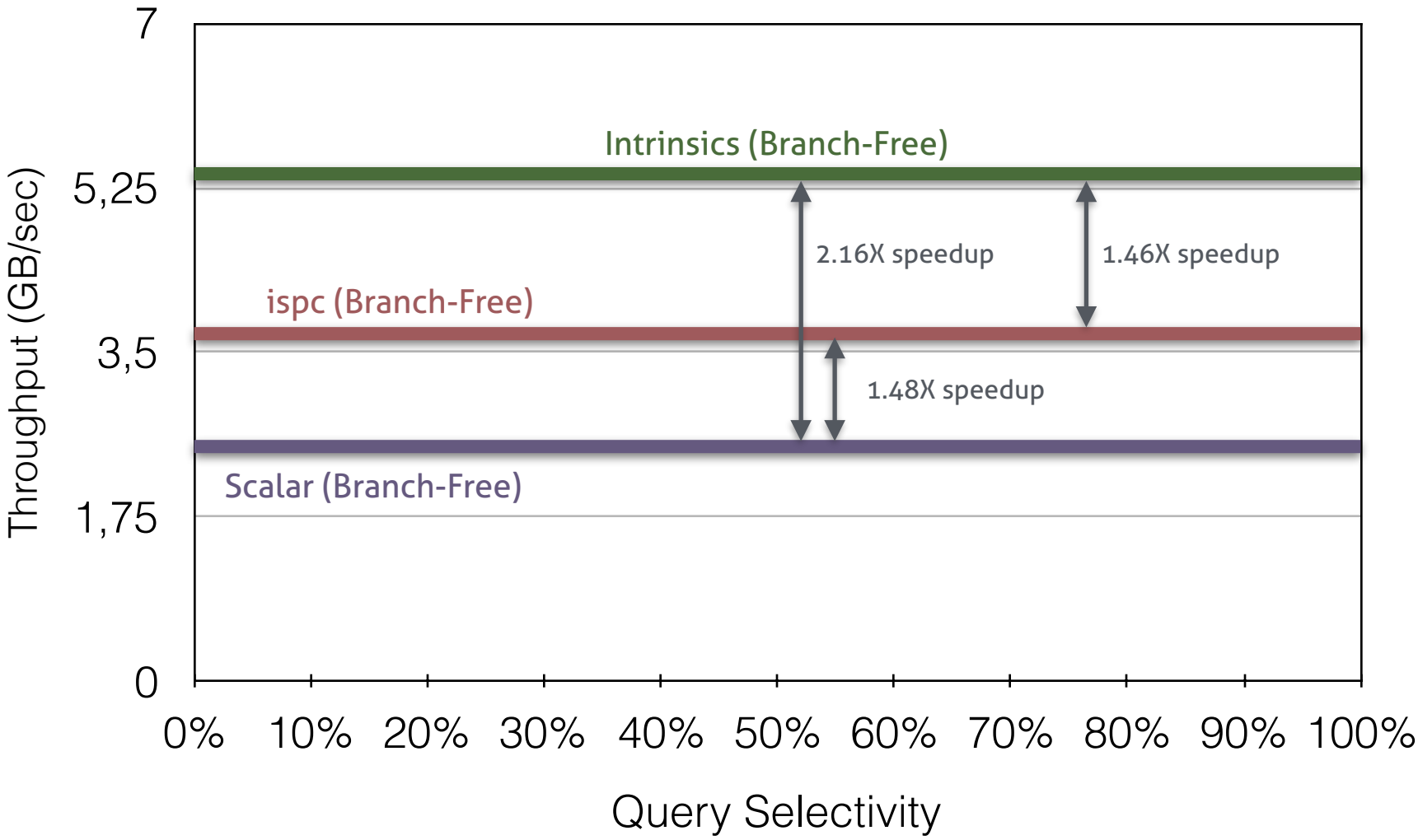
ispc vs. Intrinsics vs. Scalar (4-byte unsigned int keys)



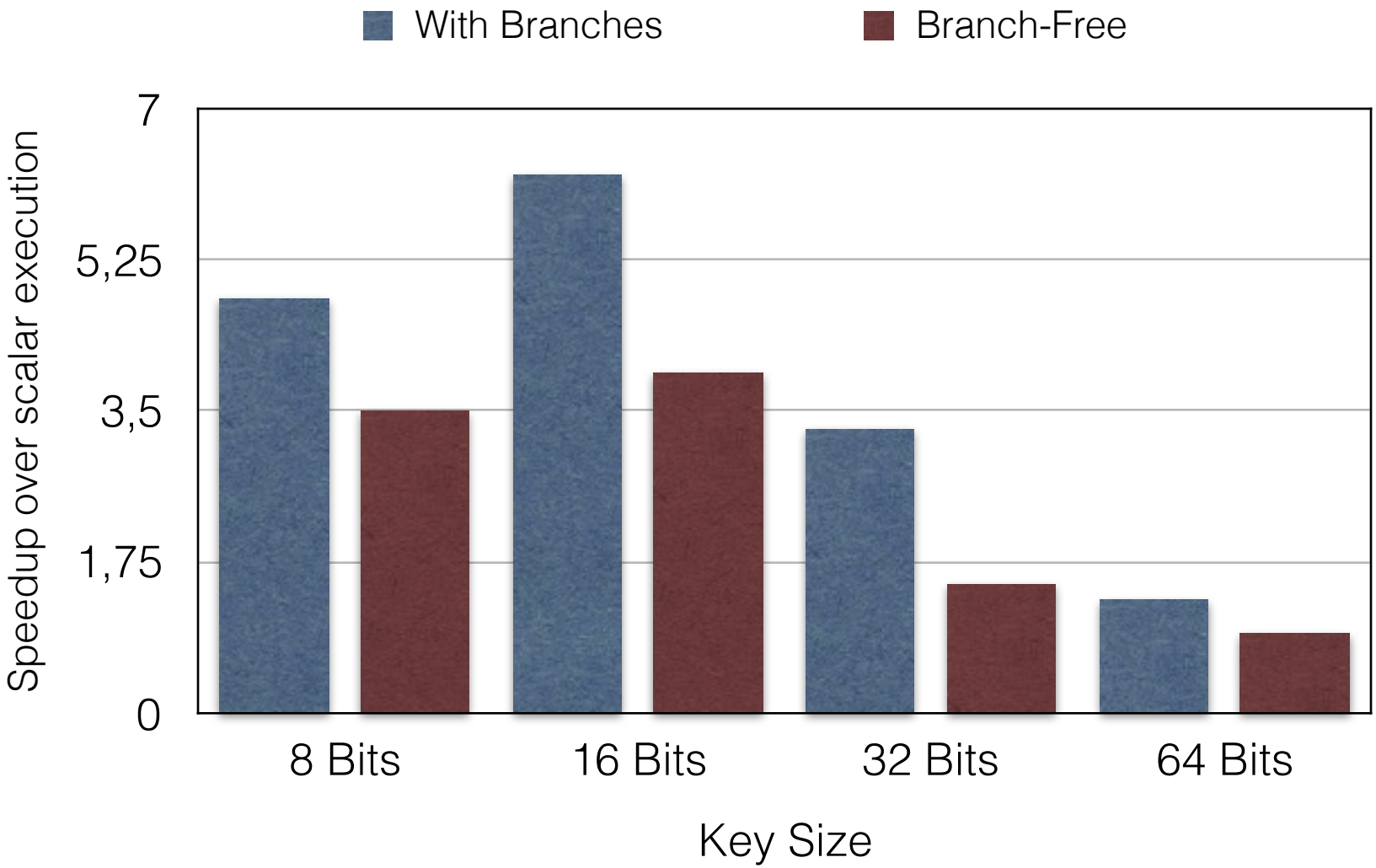
ispc vs. Intrinsics vs. Scalar (4-byte unsigned int keys)



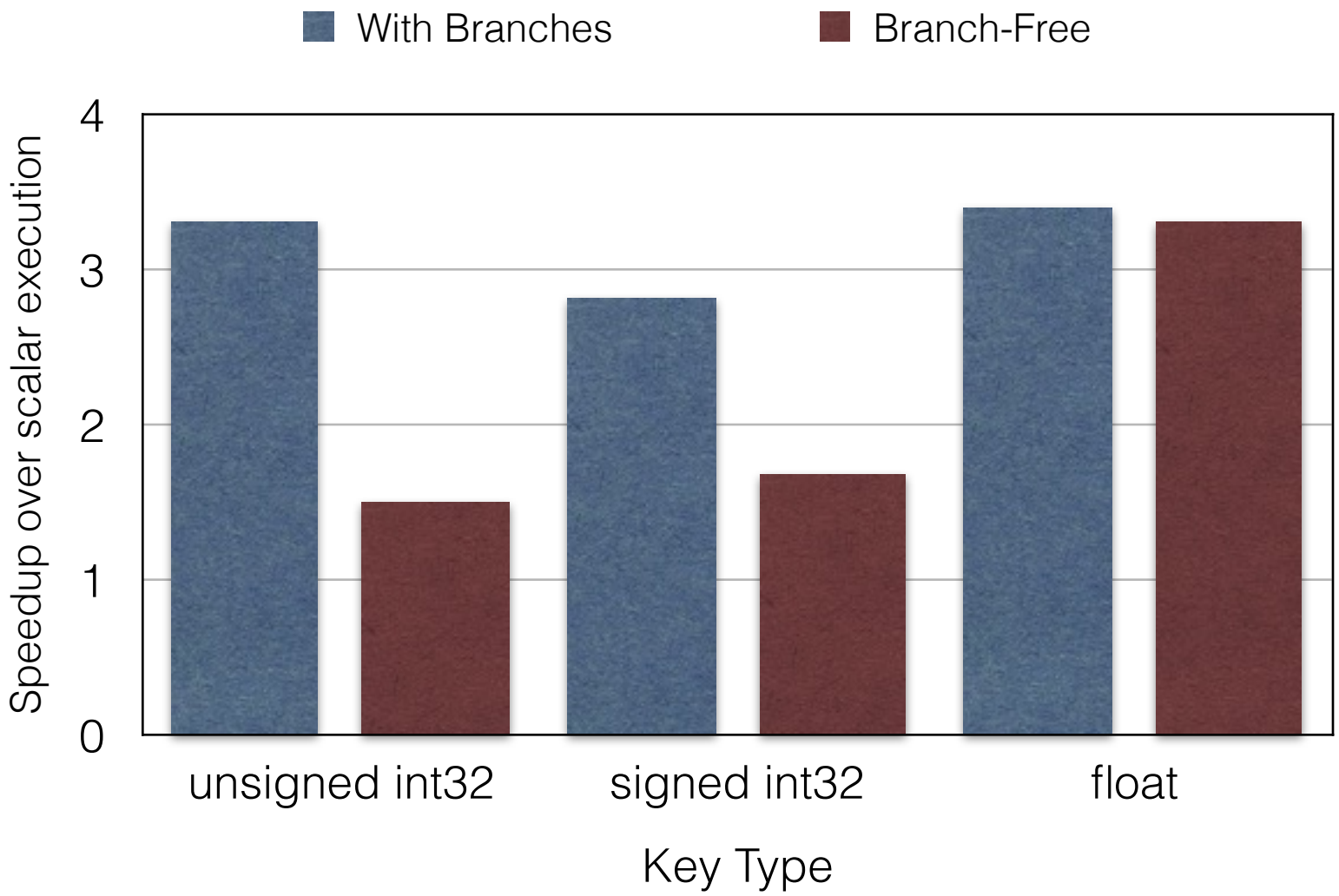
ispc vs. Intrinsics vs. Scalar (4-byte unsigned int keys)



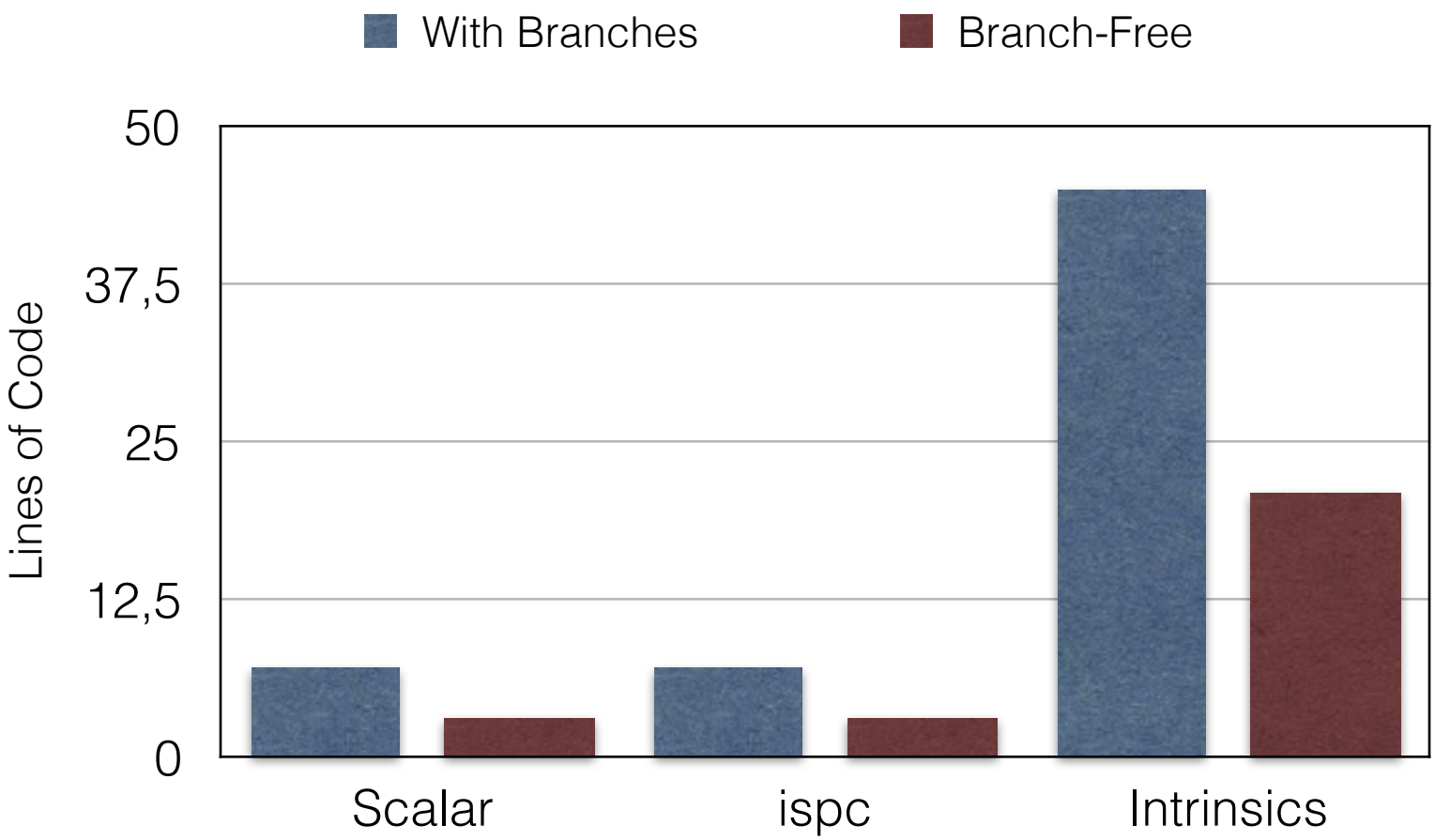
Impact of Key Size on Performance of ispc-based scan



Impact of Key Type on Performance of ispc-based scan



Code Complexity



Next Steps

- Investigate more complex database algorithms, e.g., joins, hashing, or bloom filters
- Run experiments on many-core CPUs (70+ cores, 4-way hyperthreading, AVX-512) and compare performance to modern GPUs
- Compare to other approaches to automatic vectorization, e.g., OpenCL, CilkPlus, and OpenMP

Summary

- ispc overcomes the limitations of SIMD Intrinsics
- We compared branch-free and branching variants of a SPMD-based column scan with a scalar implementation and manually-tuned Intrinsics code
- ispc achieves notable speedups over scalar implementations, however manually tuned Intrinsics code is still slightly more efficient