**Background**

Set intersection is a functional primitive used in:
- Data query processing
- Social network analytics

Data parallelism is a common feature in the architectures of modern processors. And it has the trend of becoming wider and wider.

**Approaches for set intersection**

Example: hash, tree-based data structures for set intersections

- ✔️ Good runtime complexity
- ✗ Hard to leverage vectorization features on modern processors, therefore cannot achieve speedups in practice

**Vectorizations for set intersection**

Merge-based intersection has irregular patterns of computations. It is non-trivial on how to leverage SIMD acceleration.

**Step 1: Bitmap-level intersection**

1. Bitwise-AND on bitmaps e.g., vandps
2. Segment transformation
   - e.g., pcmpeqw, pcmpeqq, with s = segment size
   - the output grouped by the segment size, e.g., 0xFFFF (s=16)
3. Non-zero segment index extraction e.g., pextrb

**Step 2: Segment-level intersection**

A segmented-bitmap data structure, and a two-step intersection approach: (1) the bitmaps are used to filter out unmatched elements, and (2) a segment-by-segment comparison is conducted to compute the final set intersection using specialized SIMD kernels.

**Performance on varying input size**

Results are shown in two Intel architectures: Haswell and Skylake. The relative performance of these methods remains consistent as we increase the input size. FESIA outperforms other scalar and SIMD set intersection methods.

**Real-world datasets**

We study the performance on two real-world tasks: (1) a database query task, and (2) a triangle counting task in graph analytics.