Motivation
- Growth in graph data sizes (social networks, scientific computing, biology, etc.)
- Need to process graphs quickly
- What approach to use? Distributed memory, shared memory, disk-based
- Shared memory is the fastest, but limited by memory size
- Cost of renting cloud machines increases with RAM size

Graph Compression
- Format: for each vertex, store differences between consecutive neighbors
- Encode each difference using a k-bit code. Use k-1 bits for data, 1 bit as the “continue” bit
- We use 8-bit (byte) and 4-bit (nibble) codes
- Example: encode “401” using a byte-code
- In binary: 1 1 1 0 0 0 1 0
- Byte-code: 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1

Graph Reordering
- Can run graph reordering (“re-numbering”) algorithms to improve locality and compression (and also performance)
- Goal: have neighbors who have ID’s close to own ID
- Various reordering algorithms: breadth-first search, depth-first search, hybrid BFS/DFS, METIS (based on finding graph separators), and our own separator-based algorithm
- Using best ordering, we get good compression for most graphs

Ligra++
- What about algorithm performance on compressed graphs?
- We implement graph compression and decoding techniques into the Ligra shared-memory graph processing framework

Ligra framework:
- Represents a subset of vertices in a vertexSubset
- edgeMap: applies a function to the outgoing edges of a vertexSubset
- vertexMap: applies a function to the vertices in a vertexSubset

- We modify the edgeMap function to decode each vertex’s compressed edges on-the-fly
- To allow for parallel decoding of high-degree vertices, we split the neighbors into chunks, compress each chunk separately, and decode each chunk in parallel
- Encoding cost is amortized across all future computations on the graph

Performance
- Trade-offs: compressed versions have smaller memory footprint than uncompressed version, but requires time for decoding
- Performance of compressed versions much better in parallel than sequentially
- In parallel, memory bandwidth/contention is more of a bottleneck, and alleviates the cost of decoding!
- In parallel, byte code performance is competitive with uncompressed version

Conclusions
- With Ligra++, we can fit larger graphs than Ligra with the same amount of memory or the same graph with less memory while maintaining performance
- We are exploring techniques that reduce decoding cost to further improve the running time of Ligra++