Multithreaded is Hard!
- Bug manifestation is nondeterministic
- Sequential testing techniques do not easily generalize
- Need ways to quickly and easily write debugging tools and analyses for multithreaded code

The Thrille Framework
- Targets multithreaded C/C++ programs that use Pthreads
- Provides explicit access to program concurrency by providing hooks to synchronization functions
- Allows developers to quickly write program analyses for these multithreaded programs
- Tools implemented:
  - Race Detection
  - Active Testing
  - Record and Replay
  - Model Checking

Previous Works
- There are many analysis tools, Thrille is one of the first to specifically target concurrency
- Many analysis tools can be used in conjunction with Thrille, some potential candidates are:
  - Valgrind
  - PIN
  - LLVM
  - Cil

Architecture
- Static and dynamic elements
- Statically instrument program source to track memory accesses, statement id, etc
- Pluggable. We use LLVM et al.
- Dynamically intercept calls to the system’s thread library, run analysis simultaneous with program

Application Highlights
- Race Detection
  - Track memory with an LLVM pass
  - Hybrid detection using vector clocks and locksets
- Active Testing
  - Produces executions exhibiting data races
- Record and Replay
  - Serialize execution and record ordering of synchronization events
  - Extension to add random scheduler “noise” for better stress testing
- Model Checking
  - Record enabled threads at each synchronization point
  - Perform depth first search of interleaving space
  - Iterative Context Bounded Model Checking
    - Prioritize the search of possible interleavings

What’s Next?
- Use Thrille to explore bug finding and verification in the multithreaded setting
  - Semantic Races
  - Model checking optimizations/heuristics
  - Experiment with different methods to statically instrument benchmarks