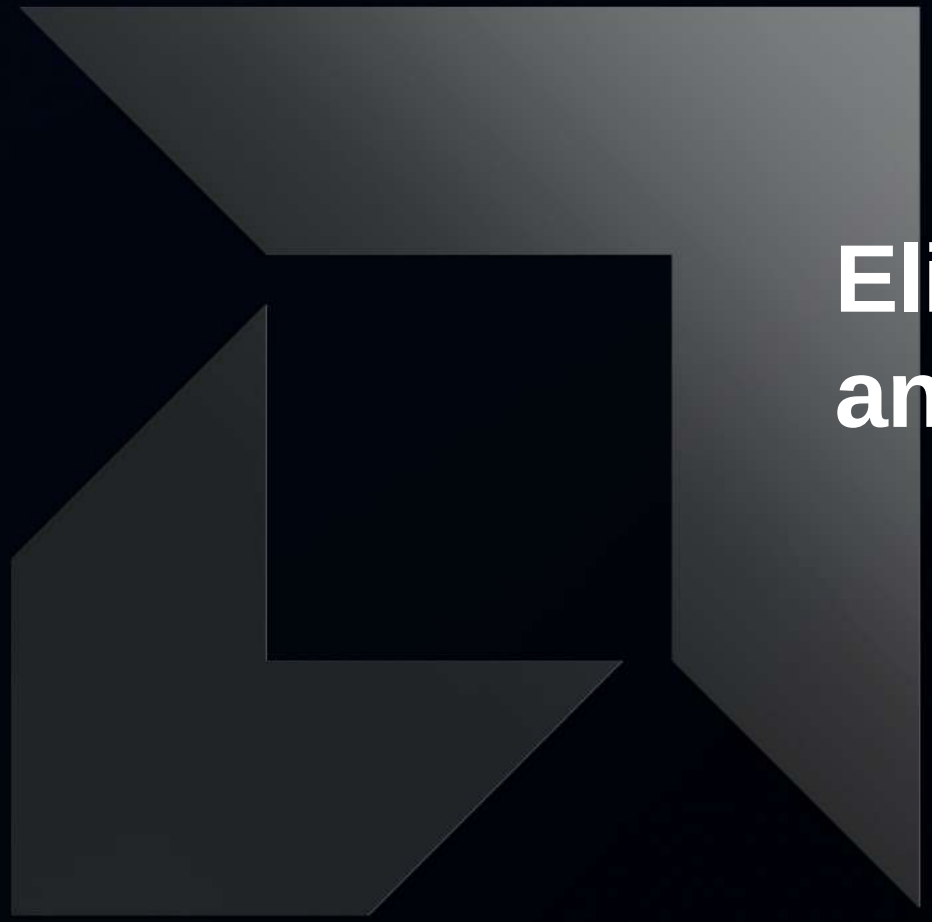
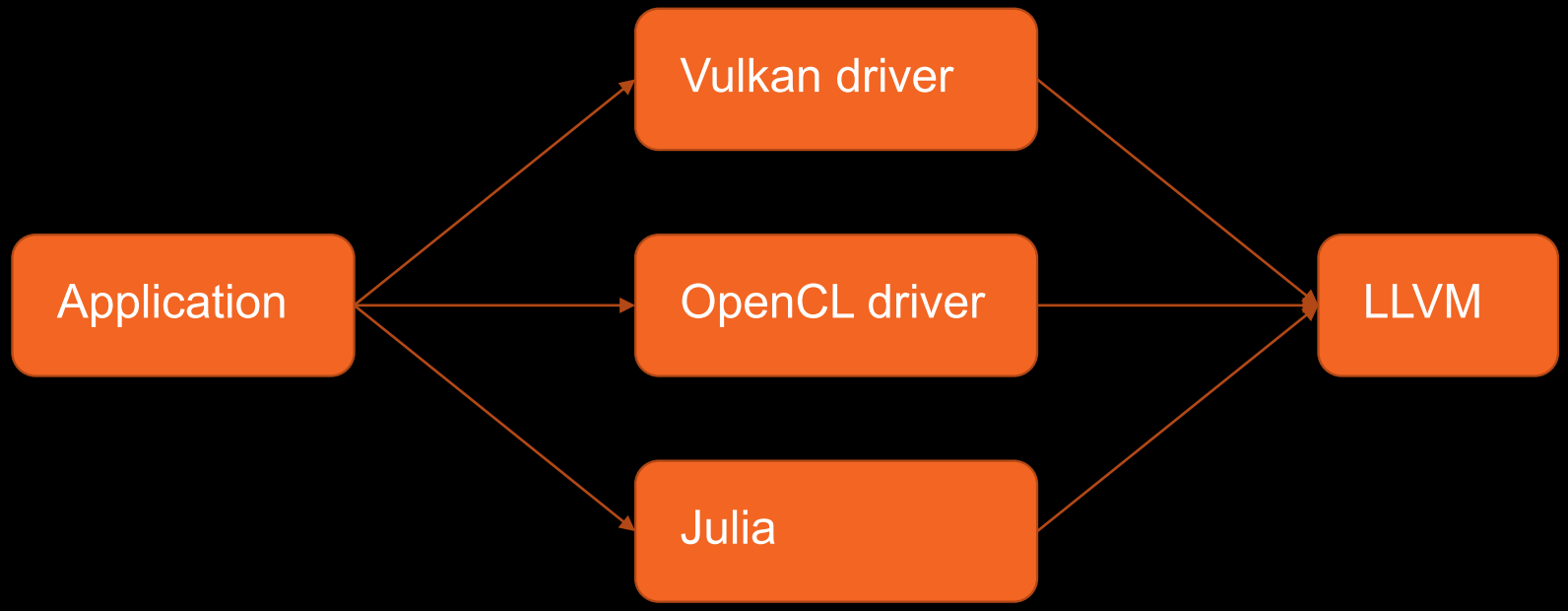


Eliminating ManagedStatic and llvm_shutdown

AMD
together we advance_

Using LLVM as a (shared) library



- We want this to Just Work – but sometimes it fails
- Global objects are in the way

Global objects in LLVM

- Command-line options (cl::opt and friends)
  - Options set for component A could cause confusion (up to miscompilation) for component B
  - New developments should prefer IR attributes over pass-specific options
  - Logical isolation will eventually be needed anyway, but that is not our topic today
- On-demand generated tables (e.g., SelectionDAG EVTs)
  - Effectively read-only
  - No real conflict between components
- Various debugging odds and ends (llvm::dbgs(), timers for profiling, …)
  - Cleaning those up could be quite painful
  - Turn a blind eye because they aren’t needed for “production” purposes?
- This talk: General problem of global object **lifetime**
  - Applies even to “read-only” tables

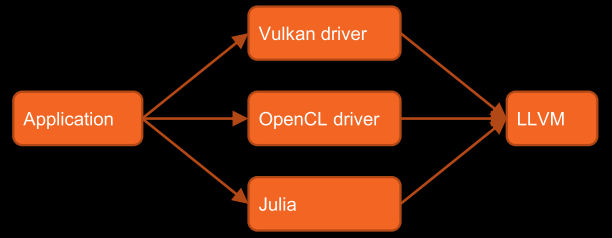
ManagedStatic and llvm_shutdown

- ManagedStatic is used to construct global objects only when they're first used


```
// Lazy-initialized global instance of options controlling the command-line
// parser and general handling.
static ManagedStatic<CommandLineCommonOptions> CommonOptions;
```

- Once constructed, objects are added to a global linked list
- llvm_shutdown frees those objects in reverse order

- Q: Should a driver (plugin etc.) call llvm_shutdown when it is unloaded?
- There is no answer!
  - If it calls llvm_shutdown, other components may be corrupted
  - If it doesn't call llvm_shutdown, it may leak memory



Solution: Remove ManagedStatic

- All uses of ManagedStatic can be replaced with a “function-scope static variable” pattern:

```
static CommandLineCommonOptions &getCommonOptions() {
  // Lazy-initialized global instance of options controlling the command-line
  // parser and general handling.
  static CommandLineCommonOptions CommonOptions;
  return CommonOptions;
}
```

- The C++ runtime destructs these objects for us when libLLVM.so is unloaded (or at process exit)

Pattern: Pack related globals into a struct

```
// All global objects associated to the DebugCounter, including the DebugCounter
// itself, are owned by a single global instance of the DebugCounterOwner
// struct. This makes it easier to control the order in which constructors and
// destructors are run.
struct DebugCounterOwner {
  DebugCounter DC;
  DebugCounterList DebugCounterOption{
      "debug-counter", cl::Hidden,
      cl::desc("Comma separated list of debug counter skip and count"),
      cl::CommaSeparated, cl::location(DC)};
  cl::opt<bool> PrintDebugCounter{
      "print-debug-counter", cl::Hidden, cl::init(false), cl::Optional,
      cl::desc("Print out debug counter info after all counters accumulated")};

  DebugCounterOwner() {…

  // Print information when destroyed, iff command line option is specified.
  ~DebugCounterOwner() {…
};

} // anonymous namespace

void llvm::initDebugCounterOptions() { (void)DebugCounter::instance(); }

DebugCounter &DebugCounter::instance() {
  static DebugCounterOwner O;
  return O.DC;
}
```

- Take special note of the idea of registering sets of command-line options together using this pattern

Status of ManagedStatic removal

- I've been slowly landing patches to remove ManagedStatic from LLVM
  - Stack on Phabricator: https://reviews.llvm.org/D129134
  - Discourse: https://discourse.llvm.org/t/making-llvm-play-nice-r-when-used-as-a-shared-library-in-a-plugin-setting/63306/
- Some of the changes are subtle and revealed "fun" issues
- Latest piece of "fun":
  - TableGen tools link against libLLVMSupport both statically and dynamically
  - This leads to globals appearing twice, with conflicts between them, in some build configurations
  - I don't know why these conflicts didn't cause bugs earlier
  - Proposed solution is to stop treating libLLVMTableGen specially
    - Stack on Phabricator: https://reviews.llvm.org/D138278
    - Discourse: https://discourse.llvm.org/t/rfc-cleaning-up-how-we-link-tablegen-tools/66678
- I will continue to slowly push on this as a background task beside my real job ☺
- Please help by following best practice and avoid/remove ManagedStatic in your corner(s) of the world

Thank you!


AMD