Google

MLIR Language Server

Modern IDE features for .mlir

River Riddle

Language Server Protocol

Language Tooling (the Old Way)

	C++	Java	C#
Vim	C++ Vim Plugin	Java Vim Plugin	C# Vim Plugin
Emacs	C++ Emacs Plugin	Java Emacs Plugin	C# Emacs Plugin
VSCode	C++ VSCode Plugin	Java VSCode Plugin	C# VSCode Plugin
Sublime	C++ Sublime Plugin	Java Sublime Plugin	C# Sublime Plugin

Language Tooling (the Old Way)

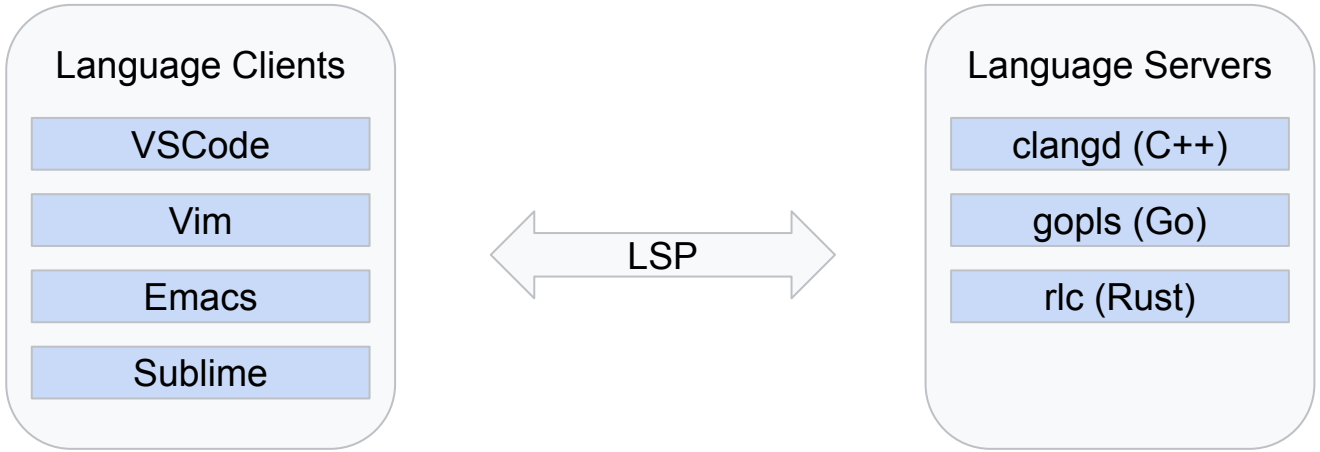
	C++	Java	C#
Vim	C++ Vim Plugin	Java Vim Plugin	C# Vim Plugin
Emacs	C++ Emacs Plugin	Java Emacs Plugin	C# Emacs Plugin
VSCode	C++ VSCode Plugin	Java VSCode Plugin	C# VSCode Plugin
Sublime	C++ Sublime Plugin	Java Sublime Plugin	C# Sublime Plugin

Language Tooling (the Old Way)

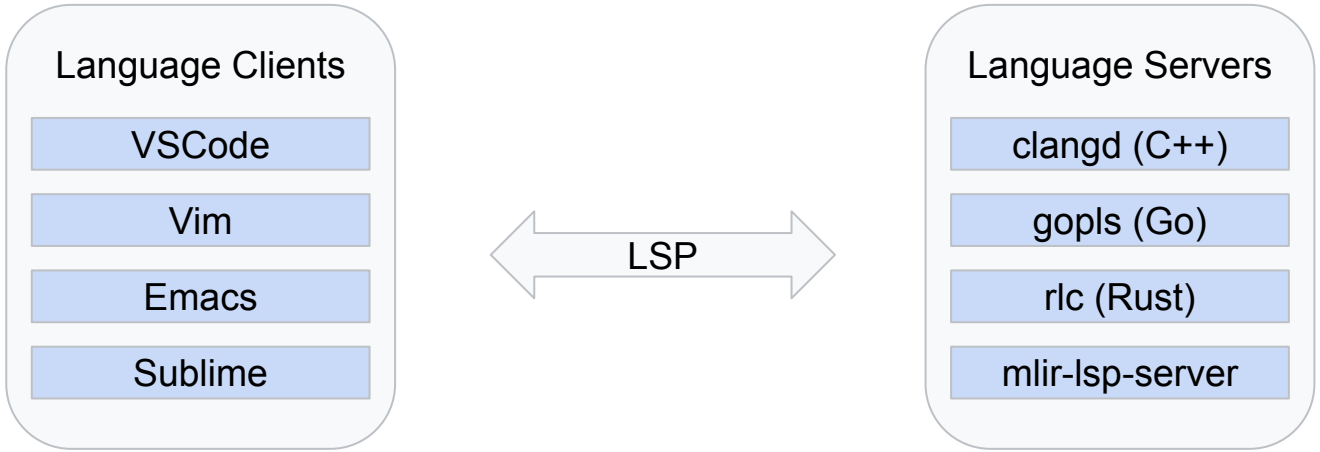
	C++	Java	C#
Vim	C++ Vim Plugin	Java Vim Plugin	C# Vim Plugin
Emacs	C++ Emacs Plugin	Java Emacs Plugin	C# Emacs Plugin
VSCode	C++ VSCode Plugin	Java VSCode Plugin	C# VSCode Plugin
Sublime	C++ Sublime Plugin	Java Sublime Plugin	C# Sublime Plugin

Language Server Protocol (LSP)

Language Server Protocol (LSP)



Language Server Protocol (LSP)



mlir-lsp-server

Why a language server for MLIR?

Why a language server for MLIR?

- MLIR is a programming language

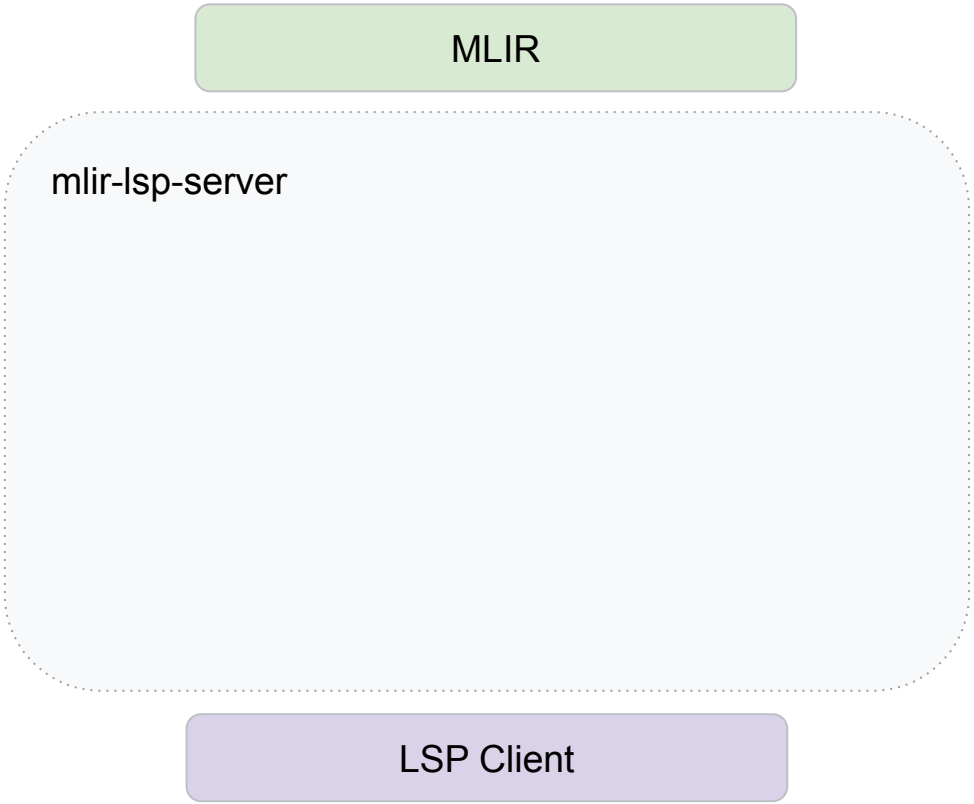
Why a language server for MLIR?

- MLIR is a programming language
- When looking at a .mlir snippet, have you ever wondered:
  - “What is the type of this value?”
  - “Where is this value/function/block defined?”
  - “What is the generic form of this operation?”

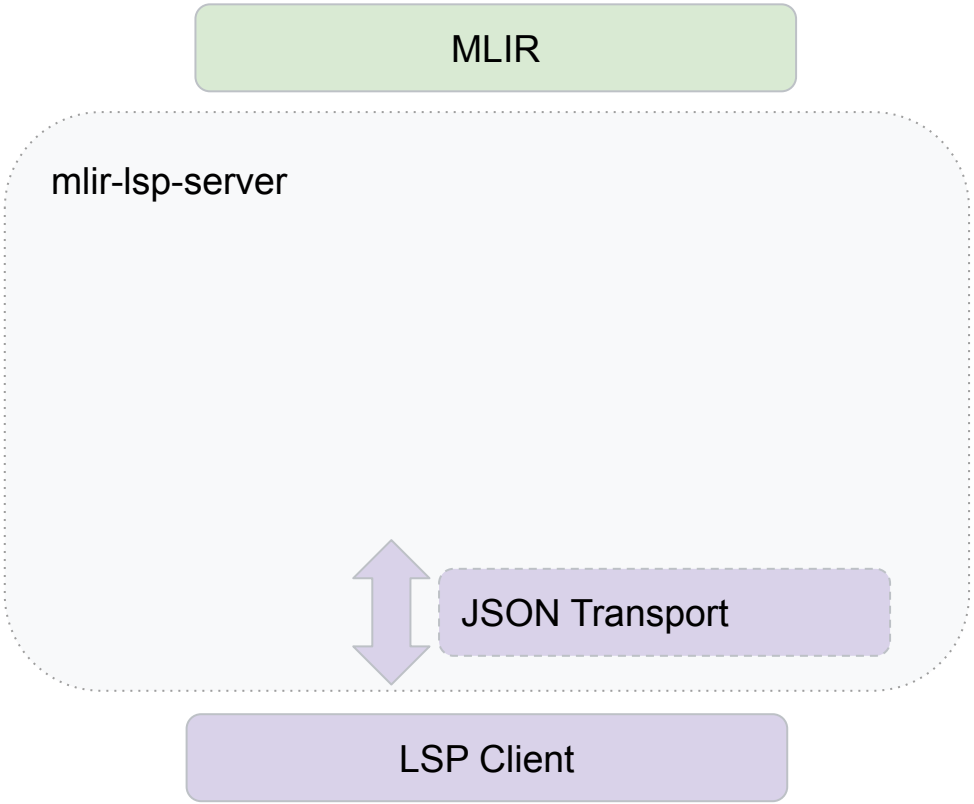
Demo

mlir-lsp-server: Design

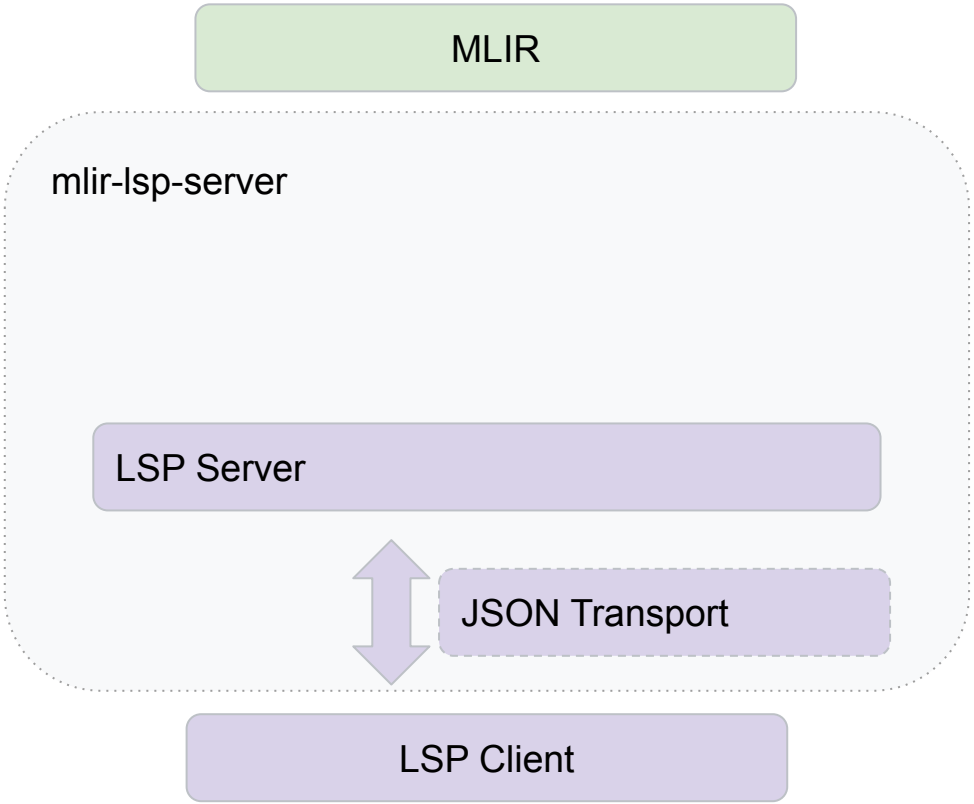
Design



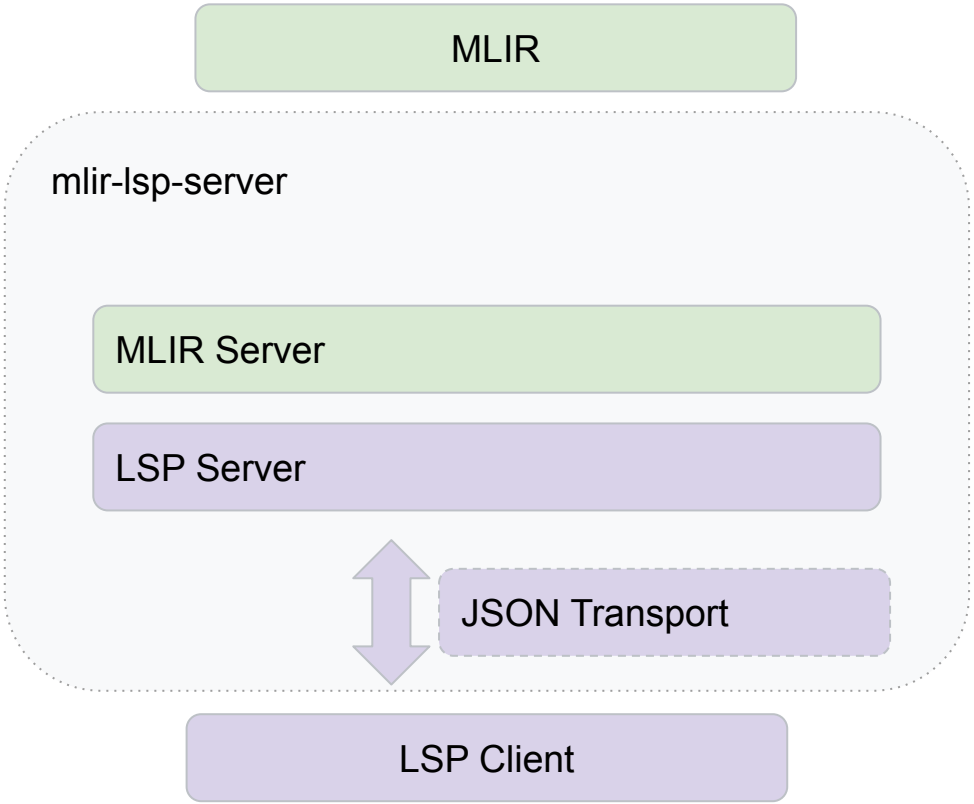
Design



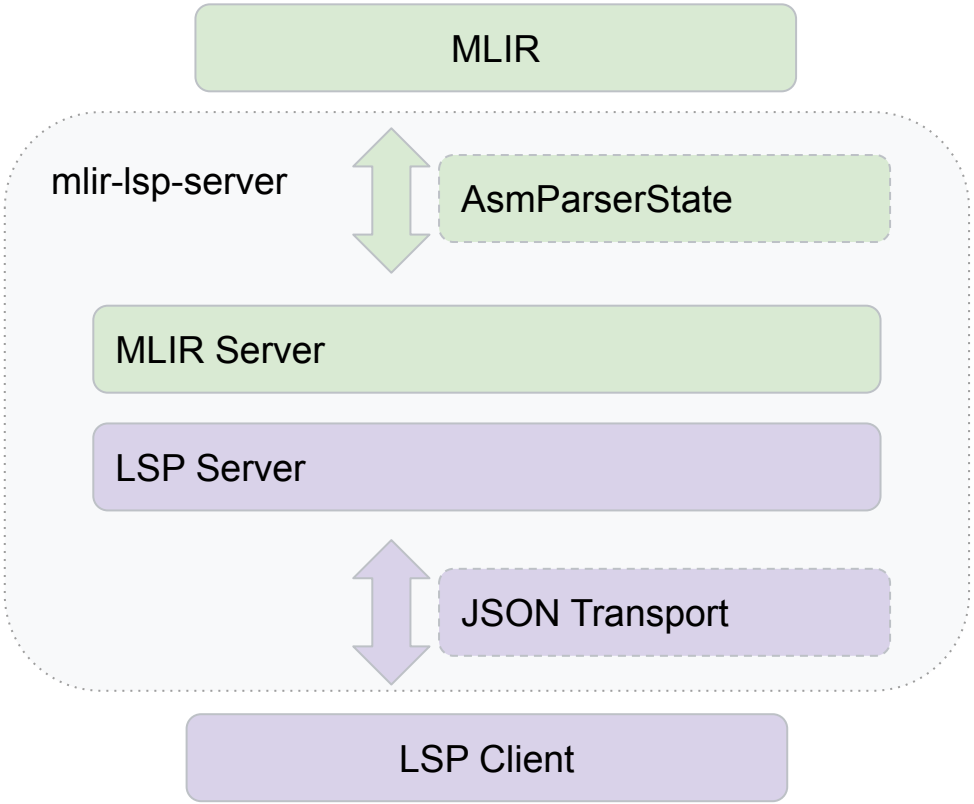
Design



Design



Design



Custom Dialects

Custom Dialects

```
#include "mlir/Tools/mlir-lsp-server/MlirLspServerMain.h"

int main(int argc, char **argv) {
  mlir::DialectRegistry registry;
  registerMyDialects(registry);
  return mlir::failed(mlir::MlirLspServerMain(argc, argv, registry));
}
```

Status

Current:

- Support for many core LSP features
  - Diagnostics, Hover, Cross-References, Navigation
- VSCode Extension
  - mlir-lsp-server binary specified via extension setting

Future:

- Add new LSP features as needed
- Add non-lsp MLIR specific utilities to VSCode extension:
  - Run passes, auto-generate CHECK lines, show diagram of CFG, etc.

Thanks!