

# CUDA Support for KDevelop IDE

Matthew Suozzo  
Zuokun Yu



# Warp Speed

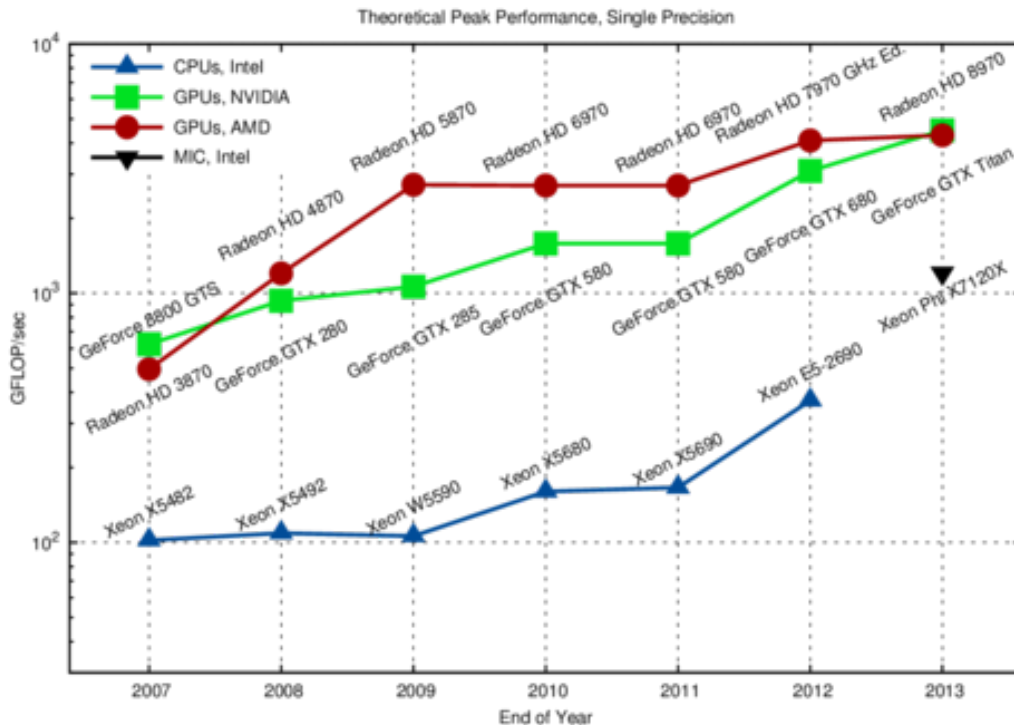
## Introduction to CUDA



# What is CUDA?

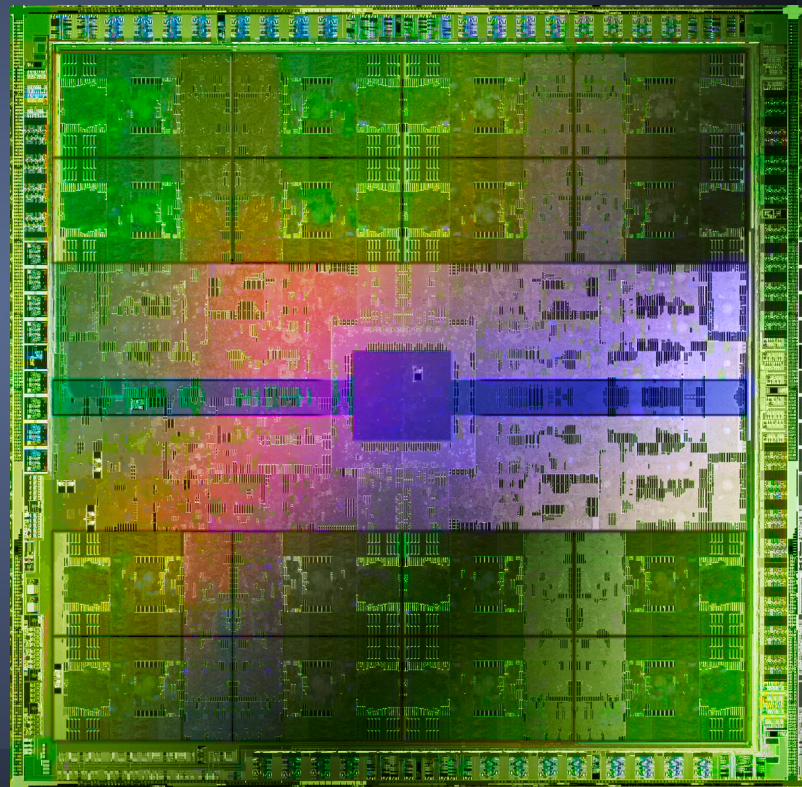
- A Parallel Computing Platform
  - ◆ Based on C/C++
  - ◆ Leverages massively parallel graphics architecture
- Proprietary to NVIDIA Hardware
- Industry Standard
  - ◆ High-Performance Computing (HPC)
  - ◆ Simulation (Physics, Fluid Dynamics)

# What is CUDA?



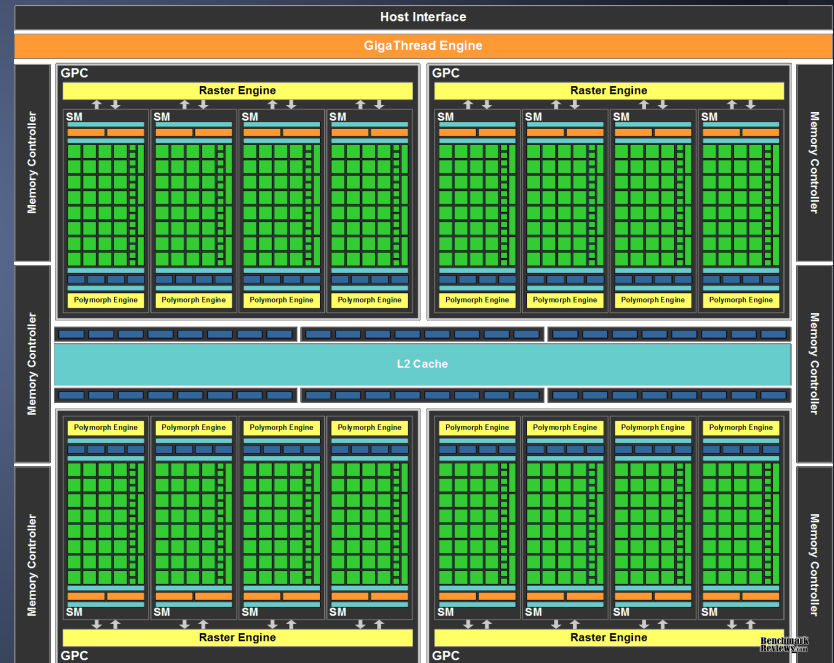
# What is CUDA?

- Raw GPU Hardware
  - ◆ NVIDIA Fermi Processor



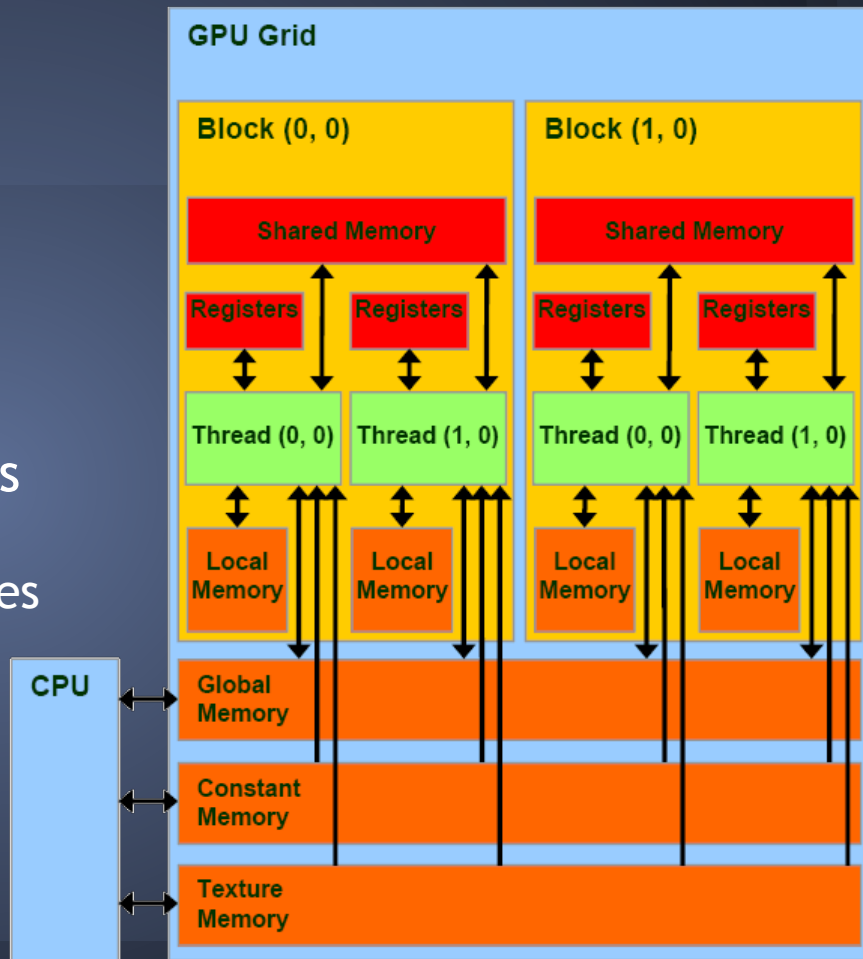
# What is CUDA?

- Take the Parallel Architecture...
- ◆ Streaming Multiprocessors
  - ◆ Dedicated Memory Banks

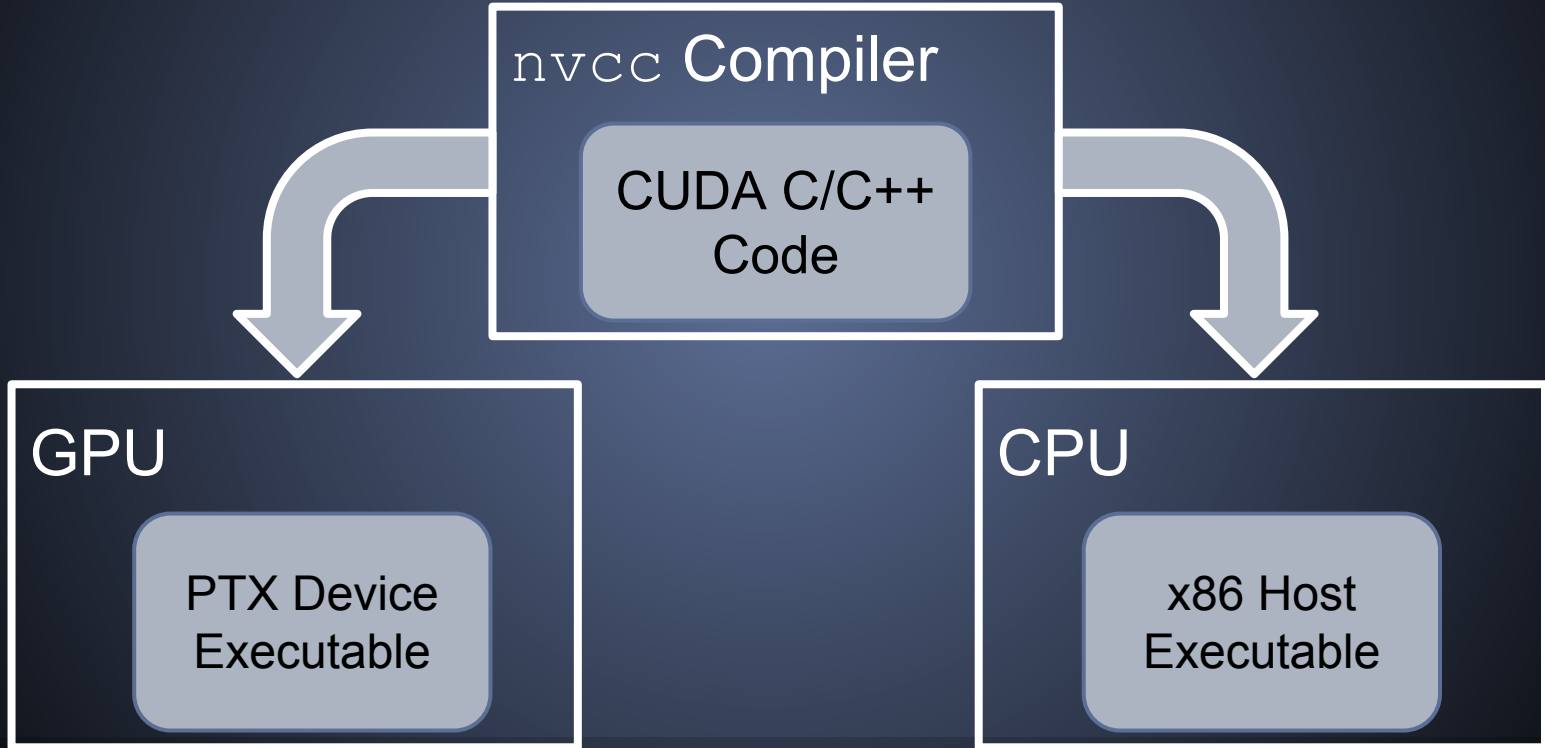


# What is CUDA?

- Useful Programming Abstractions
- ◆ Independent Address Space
  - ◆ Hierarchical Execution Primitives



# What is CUDA?





# Even More Warp Speed Intro to KDevelop

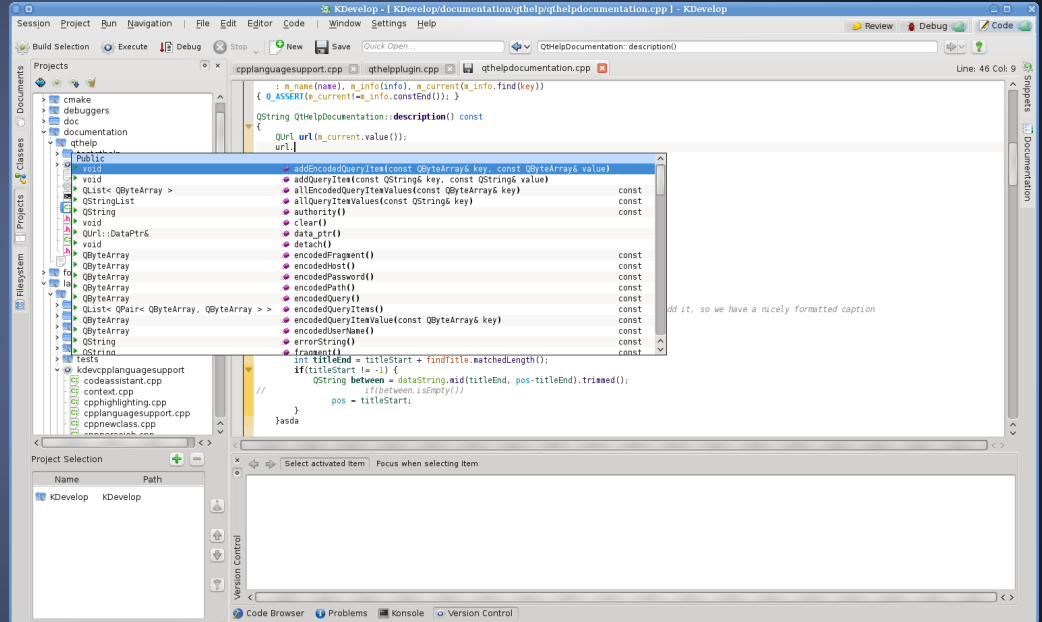


# What is KDevelop

→ Open Source IDE

→ Written in C++

→ Part of the KDE  
Community



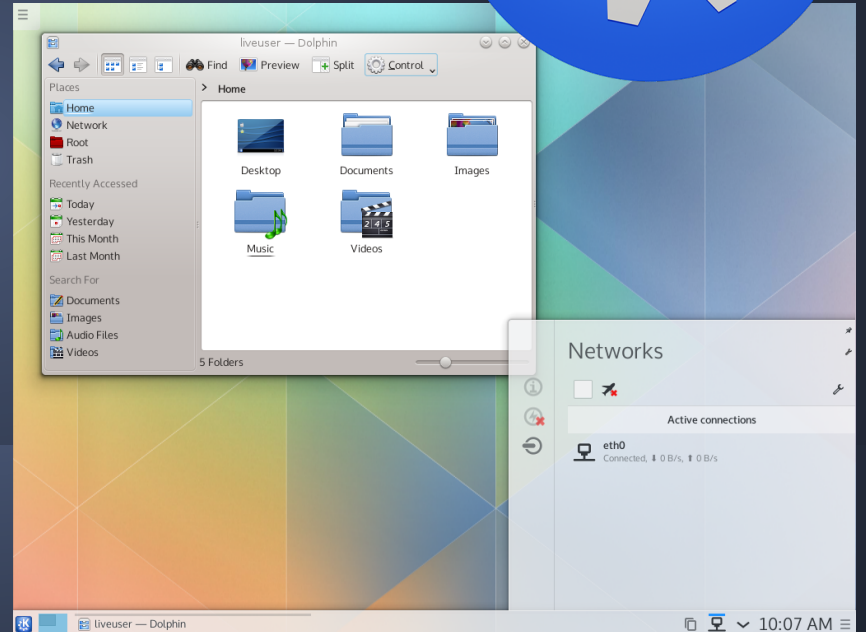
# KDE 5

→ Based on Qt Framework



→ Application Suite

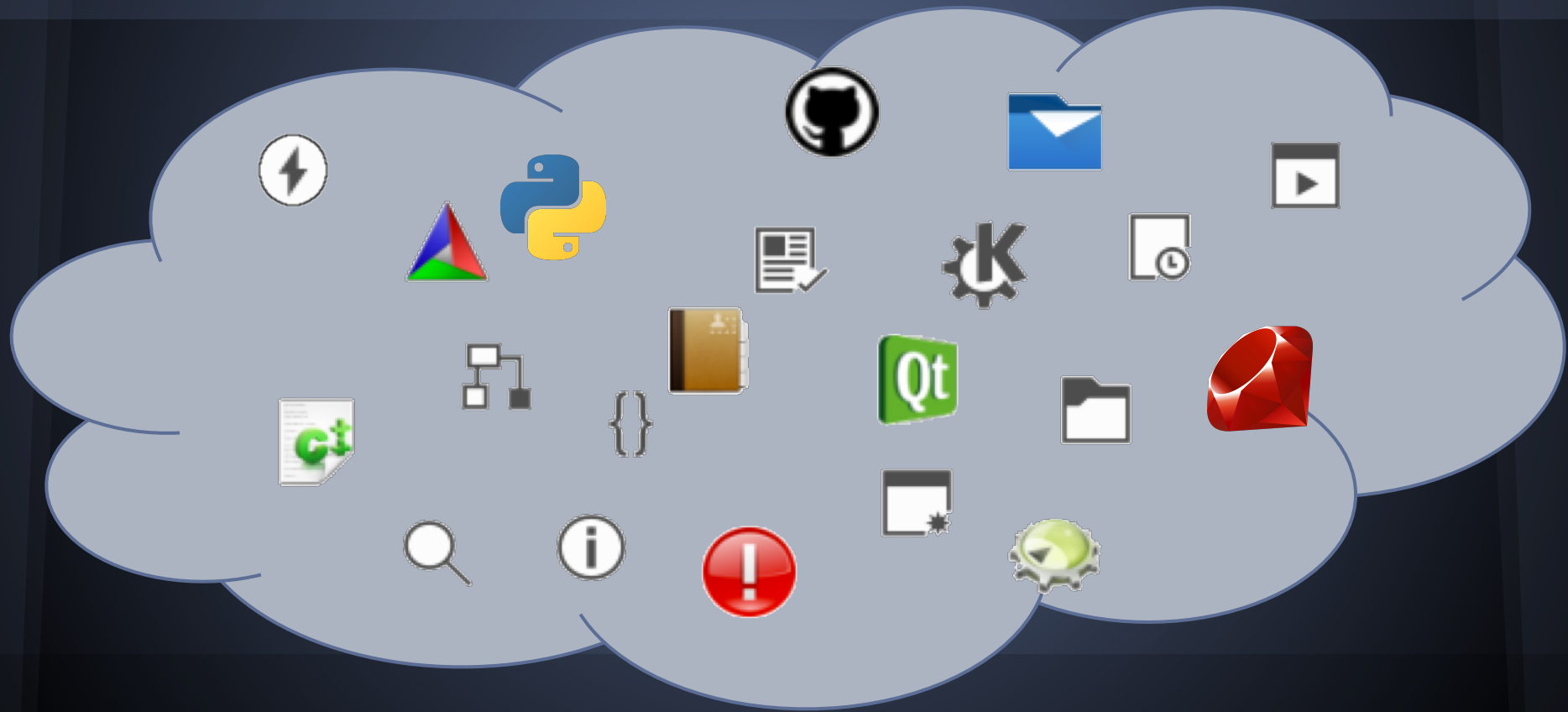
→ Plasma Desktop Environment



# Why KDevelop

- Very Low Memory Usage
  - ◆ 3x less than Eclipse
- It's Fast
  - ◆ 2-3x faster than Eclipse
- Robust Plugin Ecosystem
  - ◆ Something for Every Language

# KDevelop: Plugins



**Development**

# Our Tasks

- Basic Support for CUDA Files
  - ◆ KDevelop Recognizing .cu Source
- CUDA Syntax Highlighting
- CUDA Code Parsing

# KDevelop Quirks

## → “Switch-to-Buddy”

- ◆ Able to Switch Between .cu and .cuh Files
- ◆ Handy Feature for Any C-like Language

## → “Find-in-Files”

- ◆ Ability to search for text in .cu/.cuh files



# Syntax Highlighting



KATE the Woodpecker

→ KATE Editor  
◆ Part of KDE

→ XML Syntax Definition  
◆ Keywords  
◆ Types  
◆ Language Constructs  
◆ CUDA APIs  
◆ Typeahead

```
vim          vagrant@vagrant-ubuntu-tr...  vim          ssh          Python
<list name="CUDAtypes">
  <item> uint </item>
  <item> int1 </item>
  <item> uint1 </item>
  <item> int2 </item>
  <item> uint2 </item>
  <item> int3 </item>
  <item> uint3 </item>
  <item> int4 </item>
  <item> uint4 </item>
  <item> float1 </item>
  <item> float2 </item>
  <item> float3 </item>
  <item> float4 </item>
  <item> char1 </item>
  <item> char2 </item>
  <item> char3 </item>
  <item> char4 </item>
  <item> uchar1 </item>
  <item> uchar2 </item>
  <item> uchar3 </item>
  <item> uchar4 </item>
  <item> short1 </item>
  <item> short2 </item>
  <item> short3 </item>
  <item> short4 </item>
  <item> dim1 </item>
  <item> dim2 </item>
  <item> dim3 </item>
  <item> dim4 </item>
</list>

<contexts>
<context attribute="Normal Text" lineEndContext="#stay" name="Normal">
  <DetectSpaces />
</context>
<context attribute="Preprocessor" context="Outscoped" String="#s*ifs+0" beginRegion="Outscoped" firstNonSpace="true" />
  <DetectChar attribute="Preprocessor" context="Preprocessor" char="#" firstNonSpace="true" />
  <StringDetect attribute="Region Marker" context="Region Marker" String="//BEGIN" beginRegion="Region1" firstNonSpace="true" />
  <StringDetect attribute="Region Marker" context="Region Marker" String="//END" endRegion="Region1" firstNonSpace="true" />
<keyword attribute="Keyword" context="#stay" String="keywords"/>
<keyword attribute="Data Type" context="#stay" String="types"/>

<keyword attribute="CUDA Keyword" context="#stay" String="CUDAkeywords"/>
<keyword attribute="CUDA Data Type" context="#stay" String="CUDAtypes"/>

<keyword attribute="CUDA Automatic Variable" context="#stay" String="CUDAautomaticvariable"/>
<keyword attribute="CUDA Device Function" context="#stay" String="CUDAdevicefunction"/>
<keyword attribute="CUDA Atomic Function" context="#stay" String="CUDAatomicfunction"/>

<keyword attribute="CUDA Runtime API Function" context="#stay" String="CUDAruntimeAPIfunction"/>
<keyword attribute="CUDA Driver API Function" context="#stay" String="CUDAdriverAPIfunction"/>

<StringDetect attribute="CUDA Kernel Launch" context="CUDAKernelLaunch" String="&lt;&lt;&lt;"/>
```

# Code Parsing

## → KDevelop <= 5.2

- ◆ Used Custom C++ Parser
- ◆ ~80,000 Lines of Code

## → KDev-Clang

- ◆ Pet Project of Milian Wolff (a Primary Maintainer)
- ◆ Thin Abstraction Layer Over `clang-c` API
- ◆ ~15,000 Lines of Code

## → Why Clang?

- ◆ Fast
- ◆ Reliable
- ◆ Actively Developed



wyvern the Dragon (LLVM)

# Code Parsing

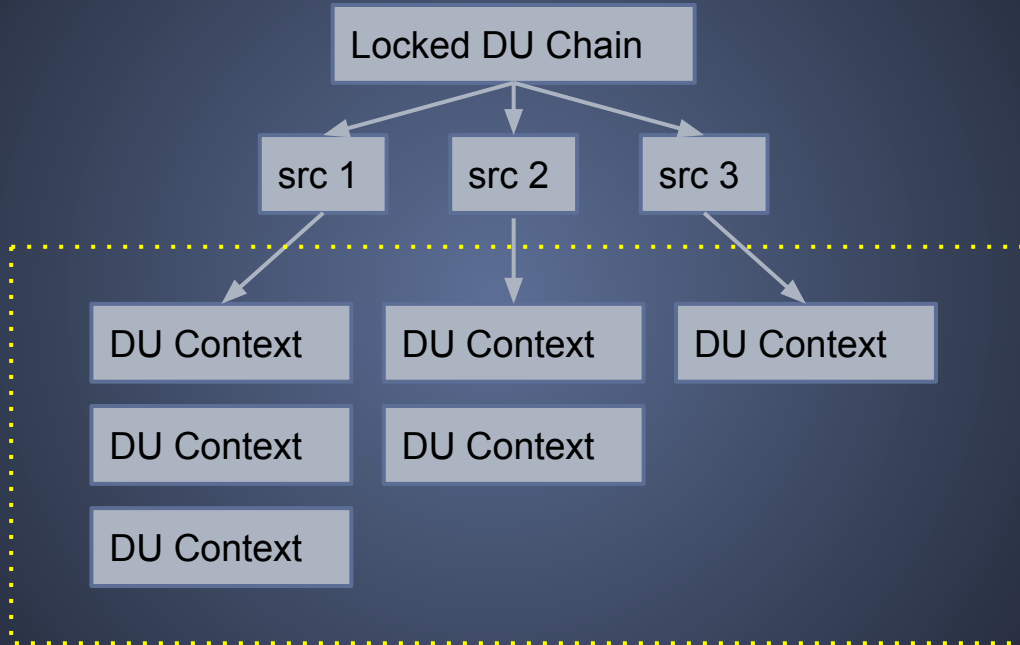
- Developed on the KDev-Clang Plugin
- All Parsing Done through the “Definition-Use Chain”
  - ◆ Language-Agnostic Program Representation
  - ◆ Used for Syntax Checking, Identifier Indexing, Autocomplete, etc.
- Modified the Clang DUChain Builder
  - ◆ Essentially: Changed Conversion from Clang AST to KDevelop AST

# The Definition Use Chain

1)



2)



Use Builder

# Clang AST

## → Basic AST Format

- ◆ Type
- ◆ Decl
- ◆ Stmt

## → Leverage Clang's API Features

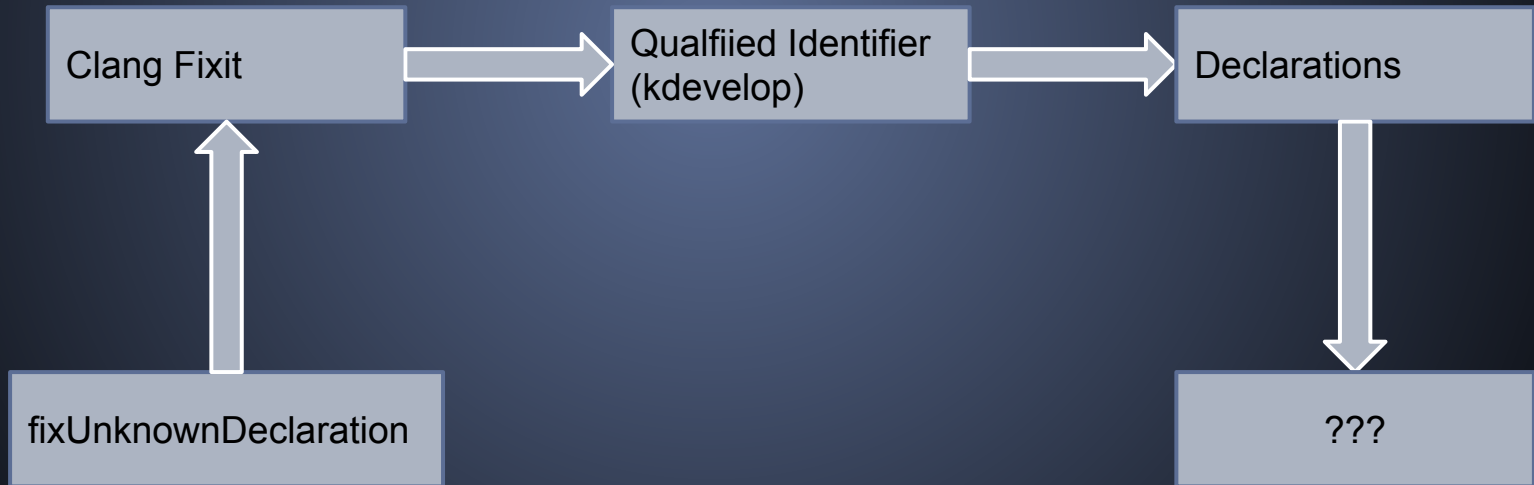
- ◆ CUDAKernelCallExpr
- ◆ CUDAGlobalAttr

```
|-PasmVazDecl 0x244cb60 <Line:272:3, col:12> devPtr 'T **'  
|-PasmVazDecl 0x244cbdb0 <Line:273:3, col:12> pitch 'size_t *'  
|-PasmVazDecl 0x244cc40 <Line:274:3, col:12> width 'size_t', unsigned long'  
|-PasmVazDecl 0x244ccb0 <Line:275:3, col:12> height 'size_t', unsigned long'  
-CompoundStmt 0x244d660 <Line:277:1, line:279:1>  
-ReturnStmt 0x244d640 <Line:278:3, col:69>  
-CallExpr 0x244d500 <col:10, col:69> 'cudaError_t' {enum cudaError'  
-ImplicitCastExpr 0x244d510 <col:10> 'cudaError_t (*)'(void **, size_t *, size_t, size_t) <'FunctionToPointerDecay'  
-DeclRefExpr 0x244d4510 <col:10> 'cudaError_t (void **, size_t *, size_t, size_t) lvalue Function 0x237d520 'cudaMalloc'  
-CStyleCastExpr 0x244cf00 <col:126, col:41> 'void **' <Dependent>  
-CStyleCastExpr 0x244cf90 <col:34, col:41> 'void *' <Dependent>  
-DeclRefExpr 0x244cf58 <col:41> 'T **' lvalue PasmVar 0x244cb60 'devPtr' 'T **'  
-ImplicitCastExpr 0x244d5f8 <col:49> 'size_t' <'ValueToRValue'  
-DeclRefExpr 0x244d510 <col:49> 'size_t *' lvalue PasmVar 0x244cbdb0 'pitch' 'size_t *'  
-ImplicitCastExpr 0x244d610 <col:56> 'size_t', unsigned long' <'ValueToRValue'  
-DeclRefExpr 0x244d020 <col:56> 'size_t', unsigned long' lvalue PasmVar 0x244ccb0 'width' 'size_t', unsigned long'  
-ImplicitCastExpr 0x244d628 <col:63> 'size_t', unsigned long' <'ValueToRValue'  
-DeclRefExpr 0x244d048 <col:63> 'size_t', unsigned long' lvalue PasmVar 0x244ccb0 'height' 'size_t', unsigned long'  
-FunctionDecl 0x244d560 <Line:271:1, line:276:1> cudaMallocPitch 'cudaError_t (void **, size_t *, size_t, size_t) inline  
-TemplateArgument type 'void'  
|-PasmVazDecl 0x244d1e0 <Line:272:3, col:12> devPtr 'void **'  
|-PasmVazDecl 0x244d240 <Line:273:3, col:12> pitch 'size_t *'  
|-PasmVazDecl 0x244d2a0 <Line:274:3, col:12> width 'size_t', unsigned long'  
|-PasmVazDecl 0x244d300 <Line:275:3, col:12> height 'size_t', unsigned long'  
-FunctionDecl 0x244d760 <foo.cu:4:1, col:46> my_kernel 'void (int *)'  
|-PasmVazDecl 0x244d6a0 <col:40, col:45> A 'int *'  
-CUDAGlobalAttr 0x244d810 <col:16>  
-FunctionDecl 0x244d8f0 pdev 0x244d760 <Line:5:1, col:50> my_kernel 'void (int *)'  
|-PasmVazDecl 0x244d800 <col:45> A 'int *'  
-CompoundStmt 0x244d9e0 <col:148, col:50>  
-CUDAGlobalAttr 0x244d9a0 <col:16>  
-FunctionDecl 0x244db50 <Line:7:10, line:9:1> main 'int (int, char **)  
|-PasmVazDecl 0x244da10 <Line:7:19, col:23> argc 'int'  
|-PasmVazDecl 0x244da80 <col:19, col:36> argv 'arg **'  
-CompoundStmt 0x244e500 <col:142, line:9:1>  
-CUDAKernelCallExpr 0x244e580 <Line:8:3, col:23> 'void'  
-ImplicitCastExpr 0x244e568 <col:3> 'void (*)'(int *) <'FunctionToPointerDecay'  
-DeclRefExpr 0x244d000 <col:3> 'void (int *)' lvalue Function 0x244d8f0 'my_kernel' 'void (int *)'  
-CallExpr 0x244dd00 <col:12, col:18> 'cudaError_t' {enum cudaError'  
-ImplicitCastExpr 0x244dd00 <col:12> 'cudaError_t (*)'(dim3, dim3, size_t, cudaStream_t) <'FunctionToPointerDecay'  
-DeclRefExpr 0x244dd68 <col:12> 'cudaError_t (dim3, dim3, size_t, cudaStream_t) lvalue Function 0x23ef590 'cudaConfig'  
-CXXConstructExpr 0x244e378 <col:15> 'dim3', struct dim3 'void (const struct dim3 & throw())' elidable  
-MaterializeTemporaryExpr 0x244e128 <col:15> 'const struct dim3' lvalue  
-ImplicitCastExpr 0x244e110 <col:15> 'const struct dim3' <NoOp>  
-ImplicitCastExpr 0x244dd00 <col:15> 'dim3', struct dim3 <'ConstructorConversion'  
-CXXConstructExpr 0x244df88 <col:15> 'dim3', struct dim3 'void (unsigned int, unsigned int, unsigned int)'  
-IntegerLiteral 0x244dc28 <col:15> 'int' 1  
-CXXDefaultArgExpr 0x244df48 <<invalid sloc>> 'unsigned int'  
-CXXDefaultArgExpr 0x244df68 <<invalid sloc>> 'unsigned int'  
-CXXConstructExpr 0x244e4e0 <col:17> 'dim3', struct dim3 'void (const struct dim3 & throw())' elidable  
-MaterializeTemporaryExpr 0x244e480 <col:17> 'const struct dim3' lvalue  
-ImplicitCastExpr 0x244e468 <col:17> 'const struct dim3' <NoOp>  
-ImplicitCastExpr 0x244e458 <col:17> 'dim3', struct dim3 <'ConstructorConversion'  
-CXXConstructExpr 0x244e408 <col:17> 'dim3', struct dim3 'void (unsigned int, unsigned int, unsigned int)'  
-IntegerLiteral 0x244dc48 <col:17> 'int' 1  
-CXXDefaultArgExpr 0x244e3c8 <<invalid sloc>> 'unsigned int'  
-CXXDefaultArgExpr 0x244e3e8 <<invalid sloc>> 'unsigned int'  
-CXXDefaultArgExpr 0x244e4d8 <<invalid sloc>> 'size_t', unsigned long'  
-CXXDefaultArgExpr 0x244e4f8 <<invalid sloc>> 'cudaStream_t', struct CUDastream_st *'  
-ImplicitCastExpr 0x244e5b8 <col:22> 'int **' <'NullToPointer'  
-IntegerLiteral 0x244e518 <col:22> 'int' 0  
ms4249@damascus /home/ms4249 $
```

# Ongoing Development

## → Type inference in Clang

### ◆ Clang `fixit`



# Additional Clang Work

## → C Preprocessor Code

- ◆ Not Migrated to Clang API
- ◆ Fixes, Refactors, Tests

## → Additional Unit Tests

- ◆ Easy Way to Get on a Maintainers' Good Sides

# Post-Mortem



# Roadblocks: The Bleeding Edge

## → KDE 5

- ◆ Initial Released in July 2014
- ◆ VERY Sparse Documentation

## → Clang Parser

- ◆ Substantial Backend Transition

## → Maintaining the Environment

- ◆ Development Toolchains Break

# Roadblocks: Code Sprawl

## → Massive Project

- ◆ KDevelop: ~160k Lines of Code
- ◆ KDevPlatform: ~200k Lines of Code
- ◆ KDev-Clang: ~15k Lines of Code (+ Clang API)

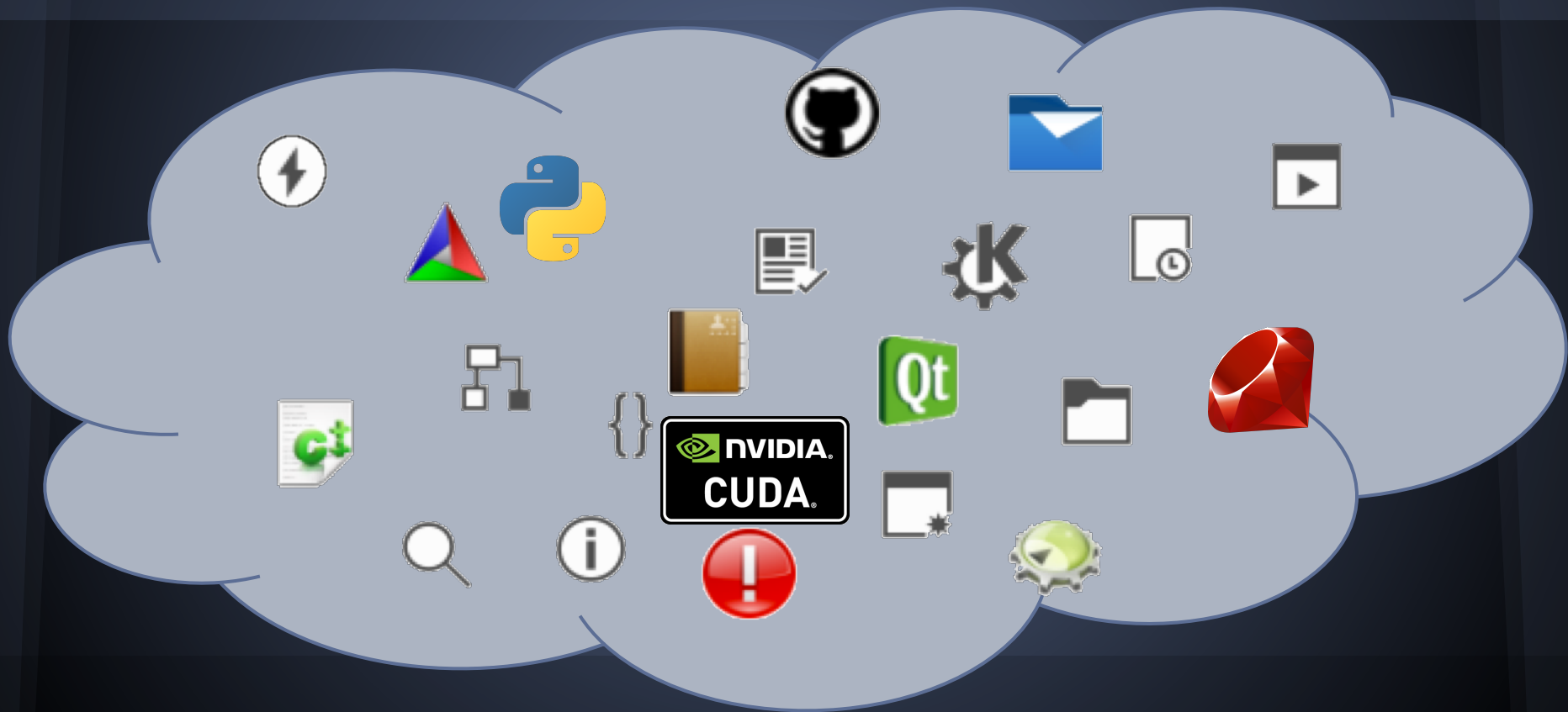
## → Chicken & Egg

- ◆ If the IDE isn't working, hard to navigate the code
- ◆ Hard to fix the IDE if the you can't navigate the code

# Where is Our Code

- Vanilla KDevelop Focused on C++
  - ◆ Additional languages added as plugins
  
- CUDA as a Plugin
  - ◆ Initial steps towards full-fledged support

# CUDA Plugin?



# Questions?

Thank you to Alex Dymo, Adam, and Jae.