CUDA Support for KDevelop IDE

Matthew Suozzo
Zuokun Yu

Warp Speed Introduction to CUDA

What is CUDA?

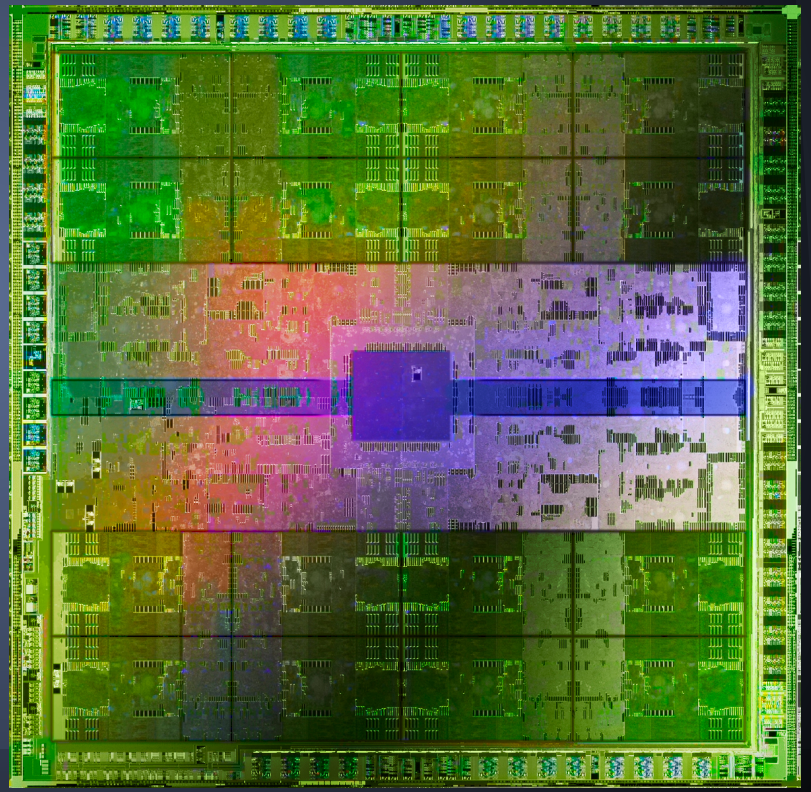
- A Parallel Computing Platform
  - Based on C/C++
  - Leverages massively parallel graphics architecture
- Proprietary to NVIDIA Hardware
- Industry Standard
  - High-Performance Computing (HPC)
  - Simulation (Physics, Fluid Dynamics)

What is CUDA?


Theoretical Peak Performance, Single Precision
CPUs, Intel
GPUs, NVIDIA
GPUs, AMD
MIC, Intel
GFLOP/sec
End of Year
2007
2008
2009
2010
2011
2012
2013
GeForce 8800 GTS
Radeon HD 3870
Radeon HD 4870
GeForce GTX 280
Radeon HD 5870
GeForce GTX 285
Radeon HD 6970
GeForce GTX 580
Radeon HD 6970
GeForce GTX 580
Radeon HD 7970 GHz Ed.
GeForce GTX 680
Radeon HD 8970
GeForce GTX Titan
Xeon X5482
Xeon X5492
Xeon W5590
Xeon X5680
Xeon X5690
Xeon E5-2690
Xeon Phi X7120X

What is CUDA?

- ➔ Raw GPU Hardware
  - ◆ NVIDIA Fermi Processor

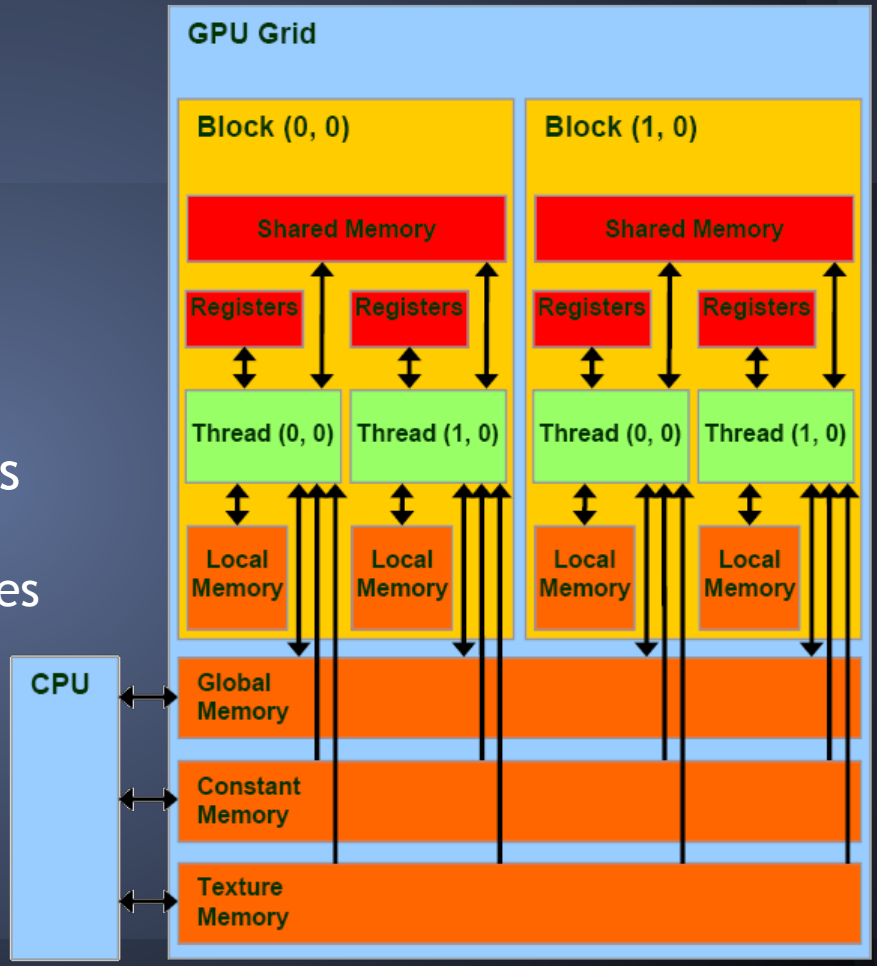


What is CUDA?

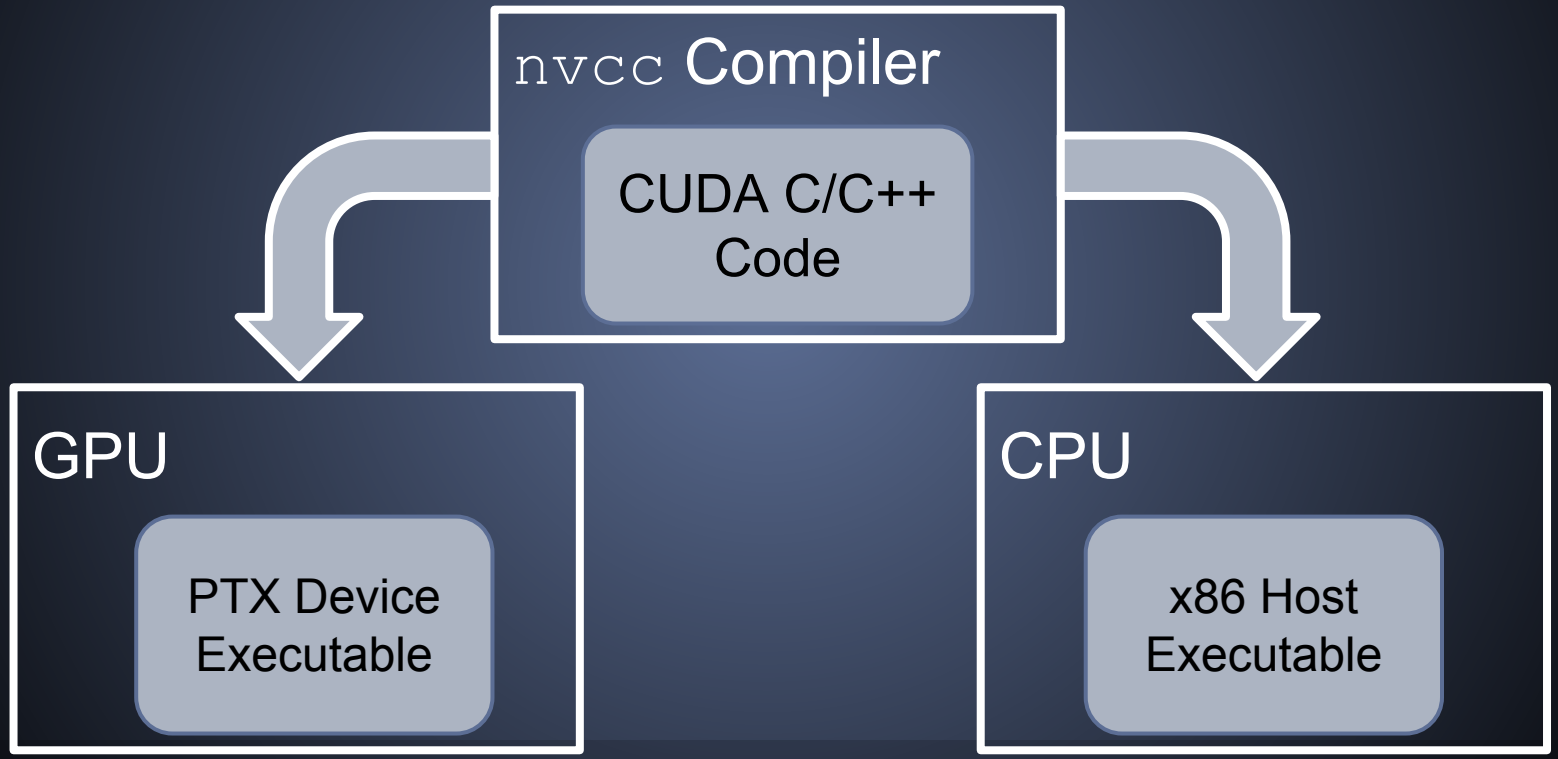
- Take the Parallel Architecture…
  - Streaming Multiprocessors
  - Dedicated Memory Banks

What is CUDA?

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



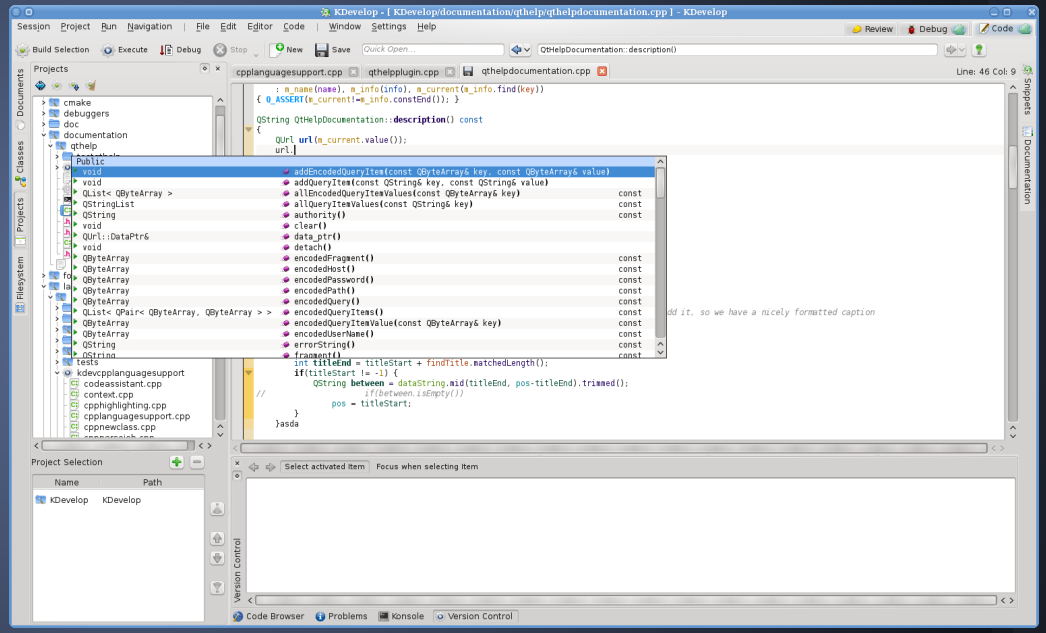
What is CUDA?


nvcc Compiler
CUDA C/C++ Code
GPU
PTX Device Executable
CPU
x86 Host Executable

Even More Warp Speed Intro to KDevelop

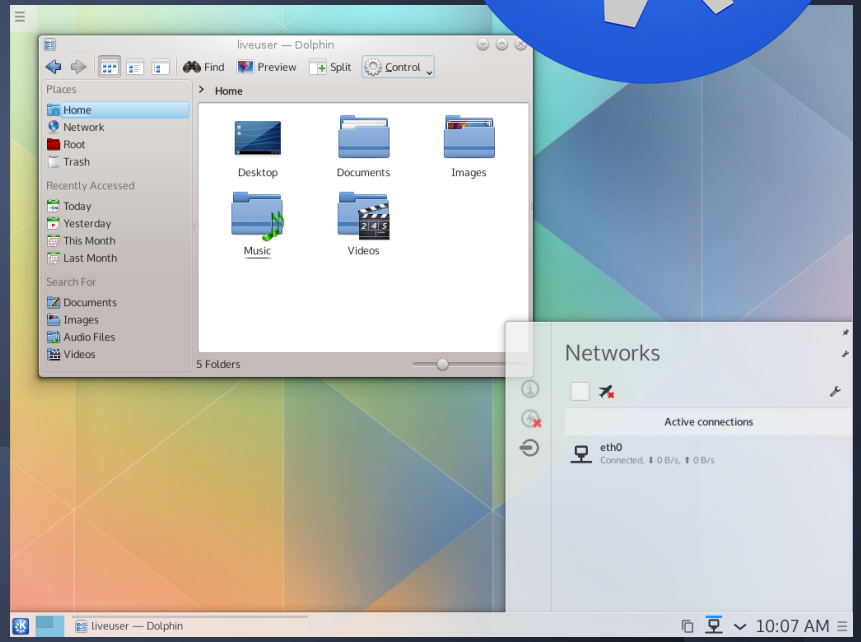
What is KDevelop

- Open Source IDE
- Written in C++
- Part of the KDE Community



KDE 5

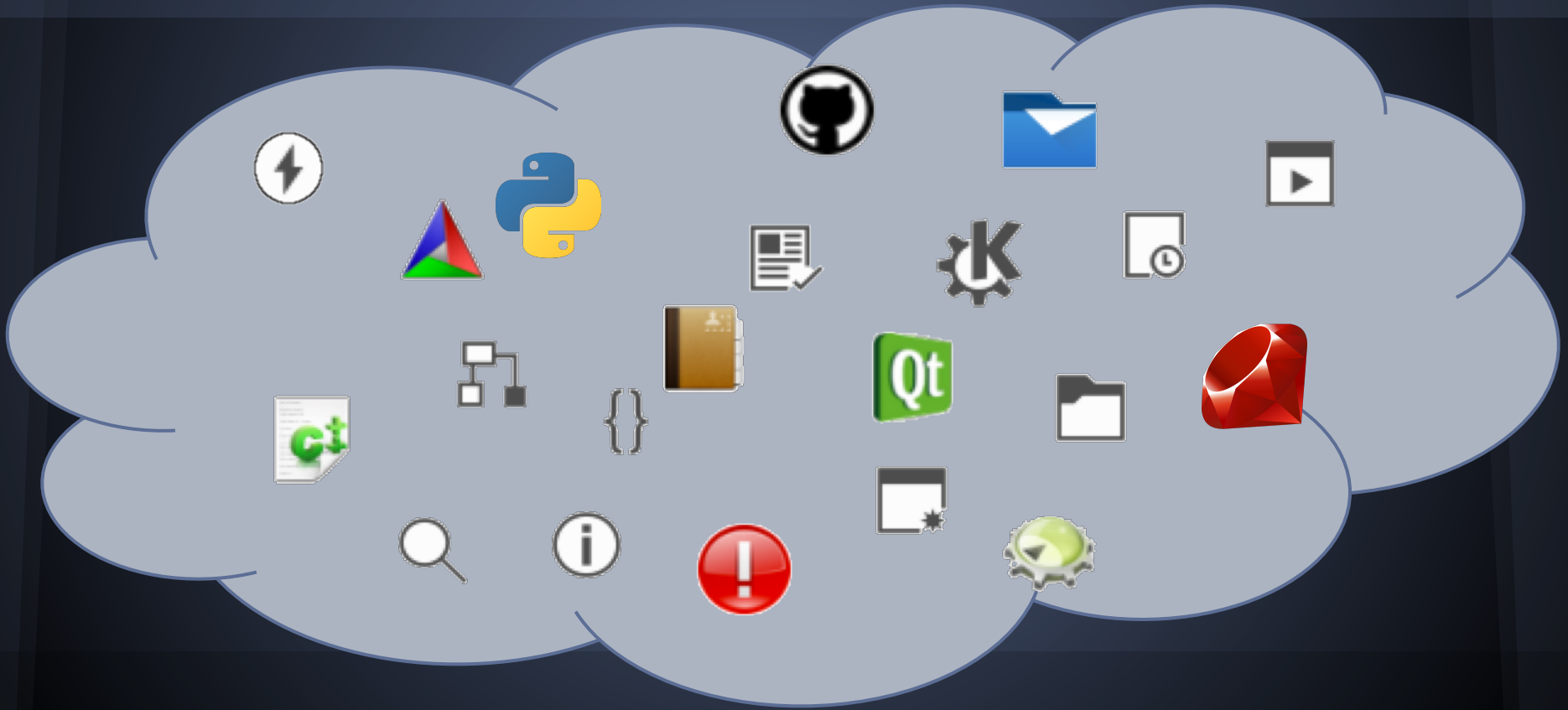
- Based on Qt Framework
  - ಠ_ಠ
- Application Suite
- Plasma Desktop Environment



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 />
  <RegExpr attribute="Preprocessor" context="Outscoped" String="#\s*if\s+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

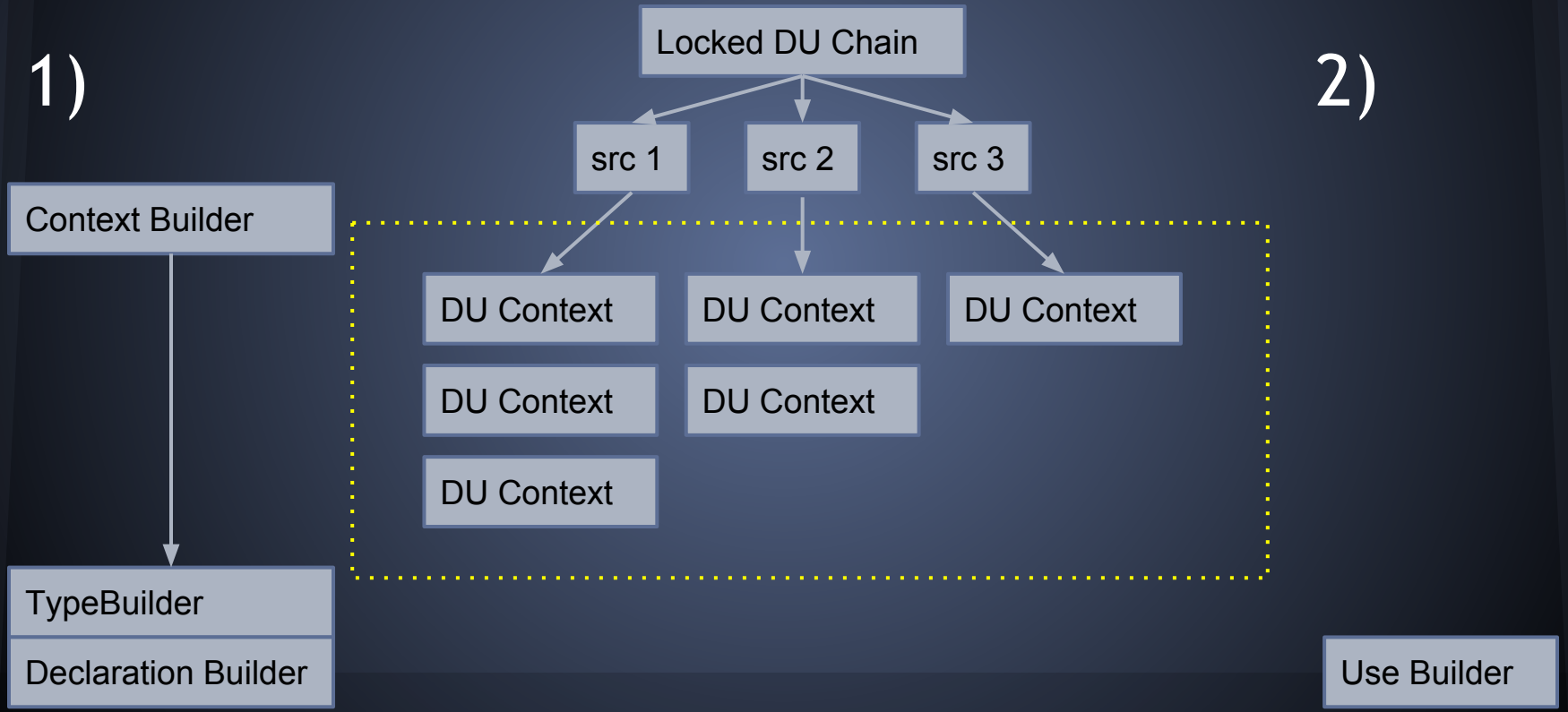
- KDevelop <= 5.2
  - Used Custom C++ Parser
  - ~80,000 Lines of Code
- KDev-Clang
  - Pet Project of Milian Wolff (a Primary Maintainer)
  - Thin Abstraction Layer Over `clang-c` API
  - ~15,000 Lines of Code
- Why Clang?
  - Fast
  - Reliable
  - Actively Developed

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

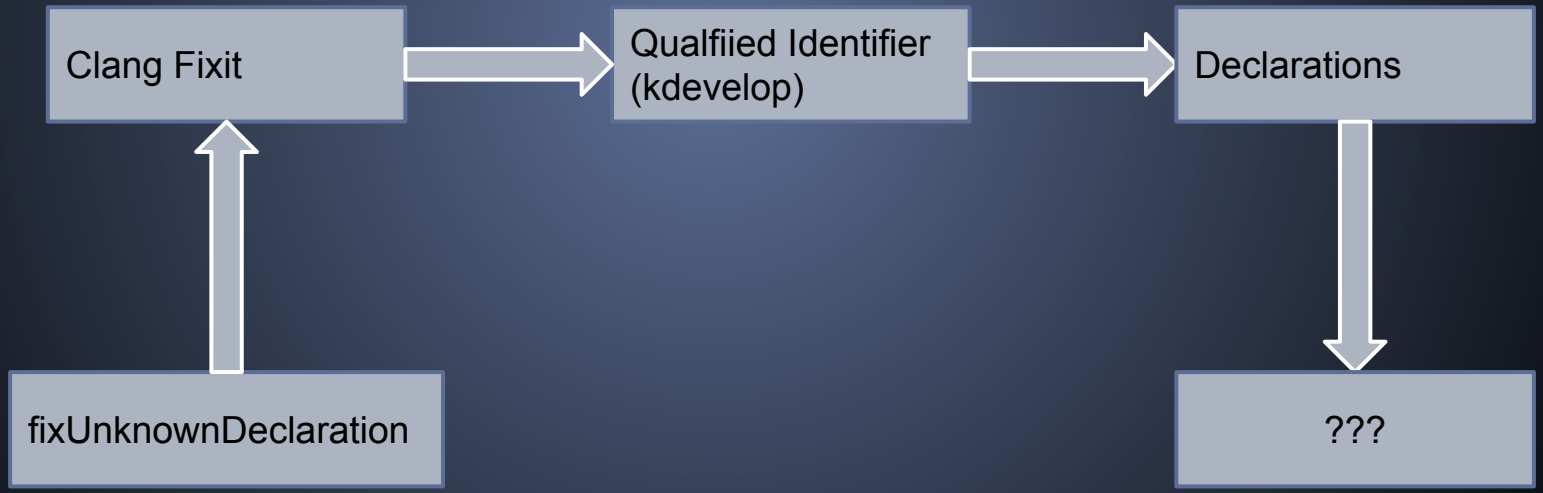


Clang AST

- Basic AST Format
  - Type
  - Decl
  - Stmt
- Leverage Clang's API Features
  - CUDAKernelCallExpr
  - CUDAGlobalAttr

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

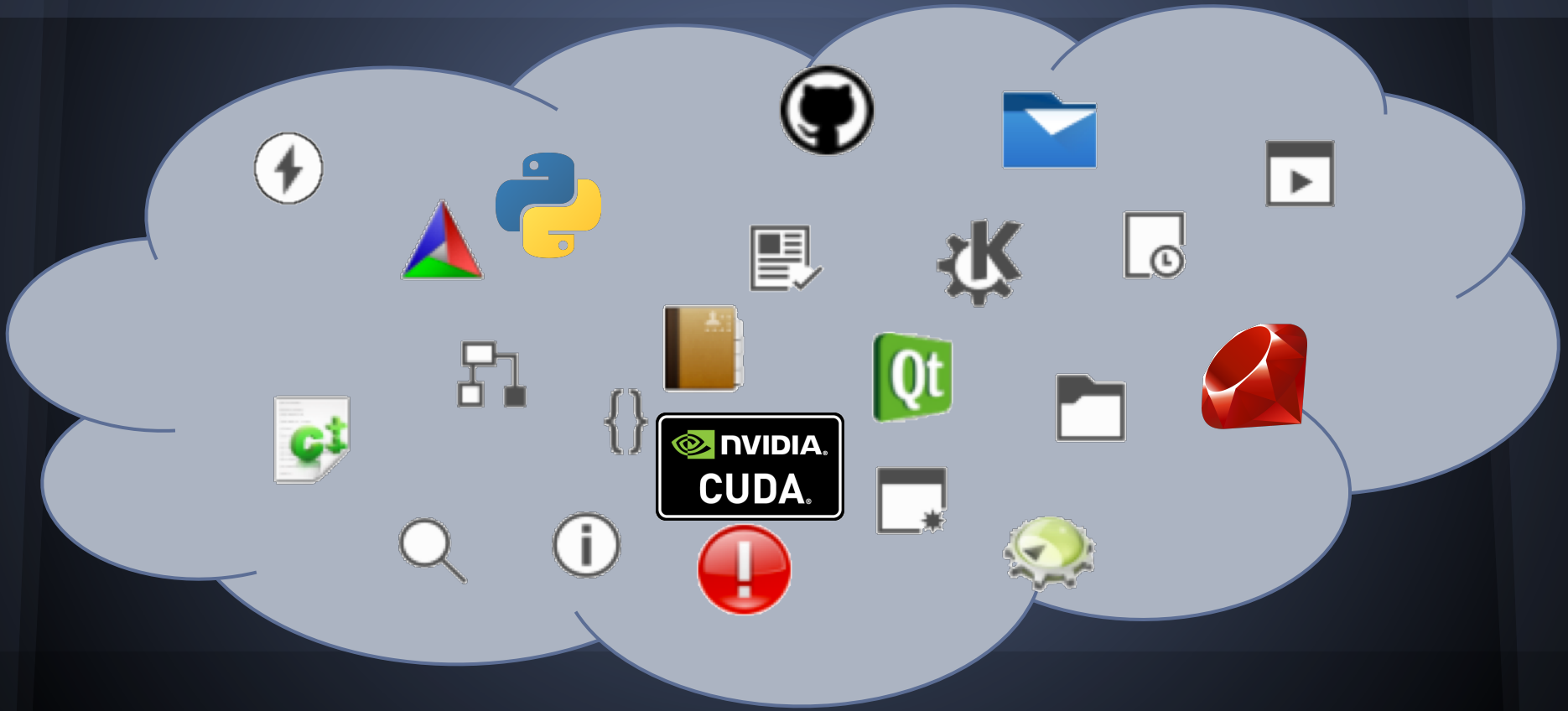
Roadblocks: Code Sprawl

- 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.