CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Extending the Capabilities of the Cray Programming Environment with Clang-LLVM Framework Integration

Sadaf Alam, Ben Cumming and Ugo Varetto

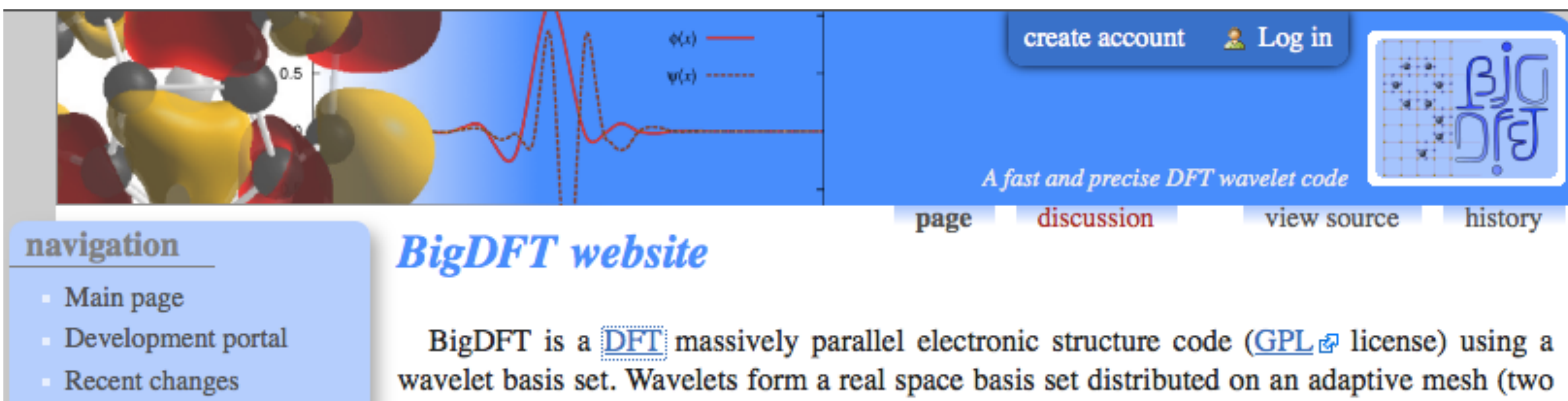
CSCS

CUG 2014

Motivation (I)

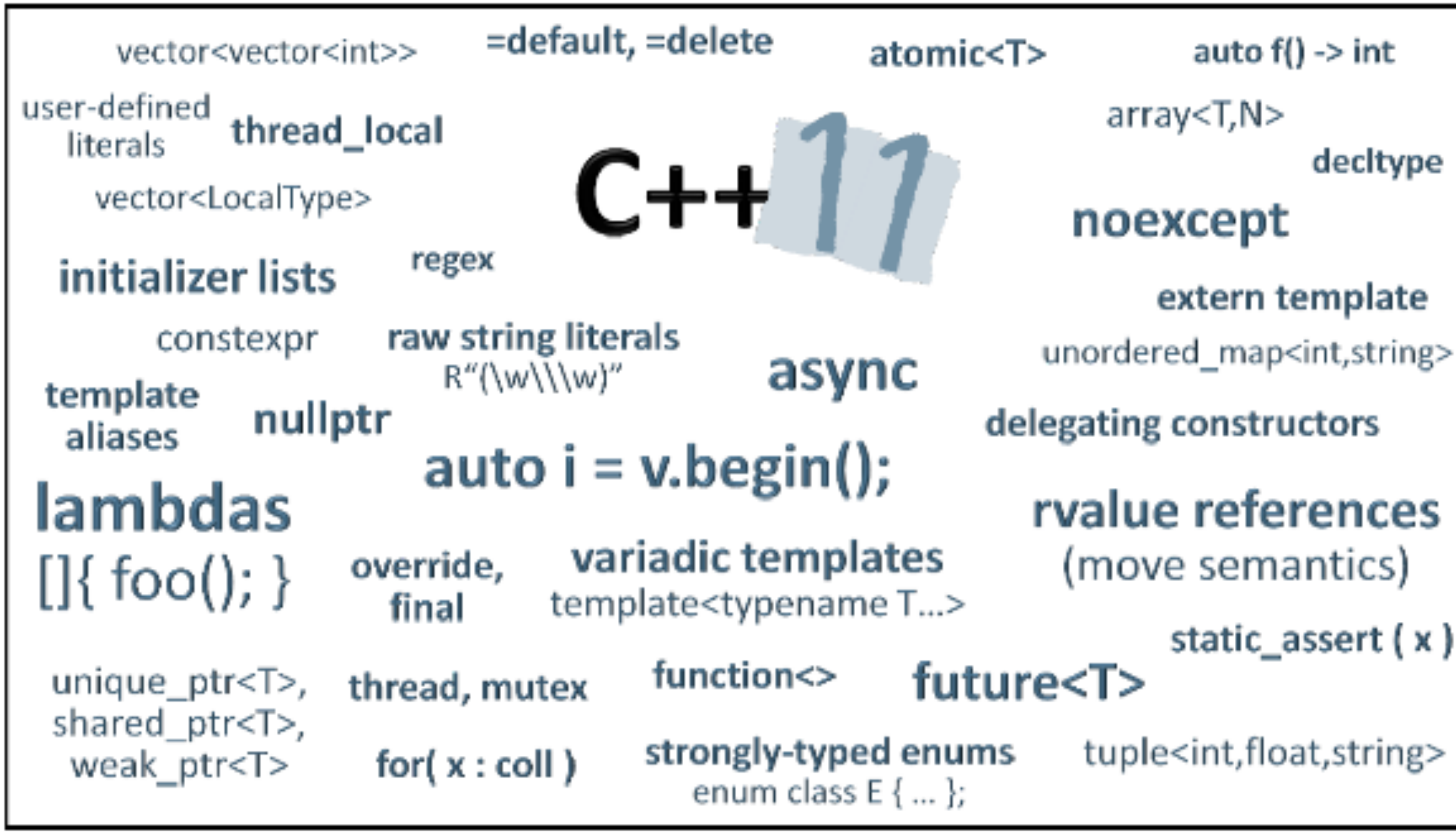
```
> cc hello.c -lOpenCL
> aprun ./a.out
Number of OpenCL GPU devices found = 1
DEVICE_NAME = Tesla K20X
DEVICE_VERSION = OpenCL 1.1 CUDA
DEVICE_VENDOR = NVIDIA Corporation
Hello, World!
…
CL_DEVICE_ADDRESS_BITS: 32
CL_DEVICE_GLOBAL_MEM_SIZE: 1744371712
```

```
Vendor: Intel(R) Corporation
Profile: FULL_PROFILE
Version: OpenCL 1.2 LINUX
Name: Intel(R) OpenCL
…
   Work item sizes: 1024 1024 1024
   Max clock freq: 2600 MHz
   Global memory: 33785212928 bytes
   Local memory: 32768 bytes
```



OpenCL accelerated application—HP2C project

Motivation (II)—C++11 -> C++1y

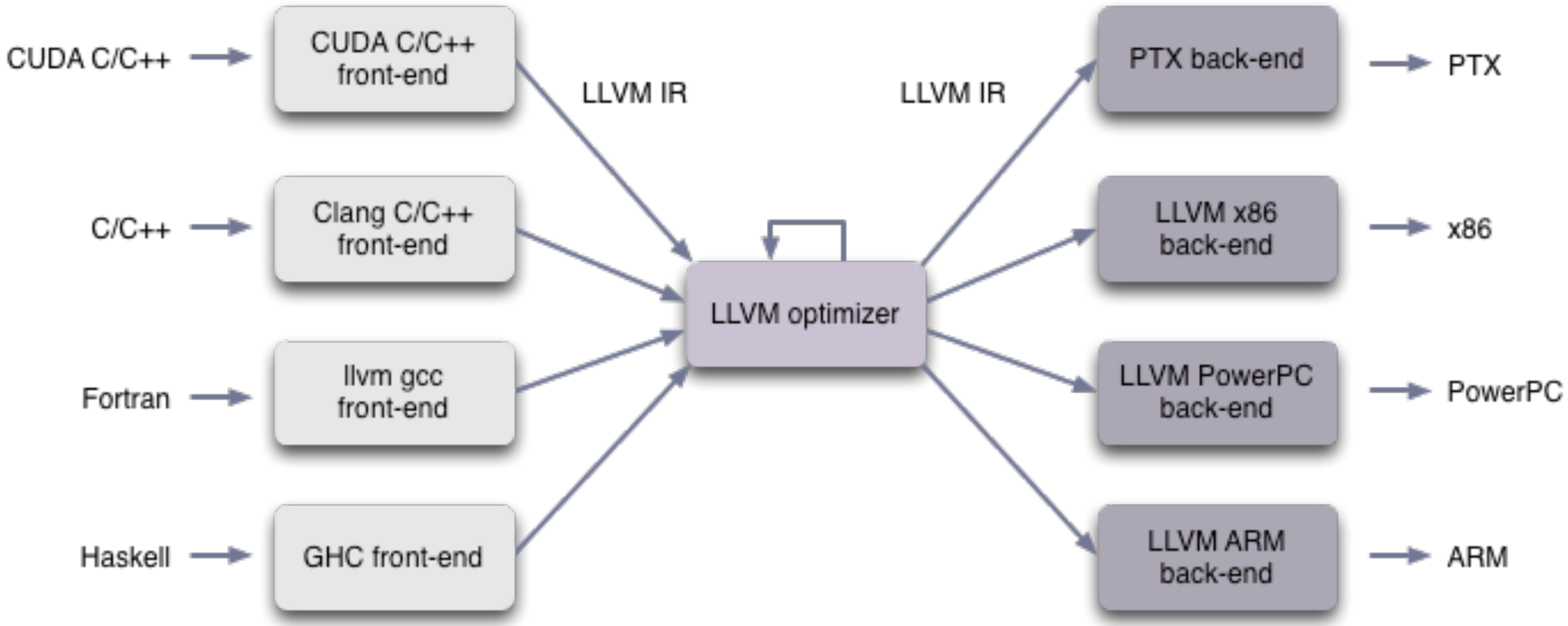


Applications use cases: COSMO, MAQUIS, DCA++, …

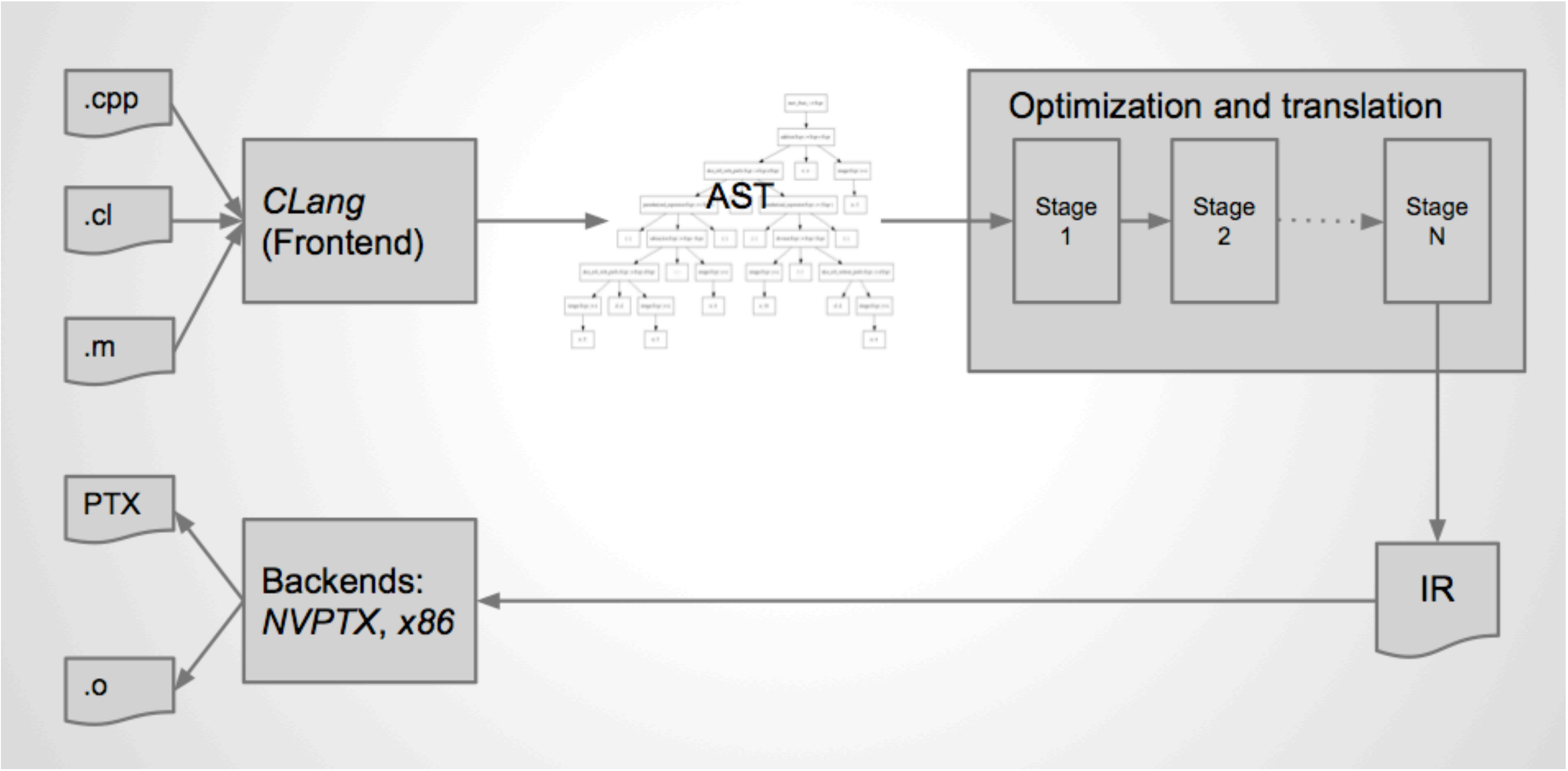
Limitations of the Current Cray PE

- OpenCL
  - Only CUDA SDK version (1.1)
  - CSCS installs CPU version
- C++1y
  - PrgEnv-cray —not up-to-date
  - PrgEnv-gnu —4.9 release made progress
  - PrgEnv-intel —gradual support
  - PrgEnv-pgi —not up-to-date
- Code development tools for OpenCL and C++1y
  - Non-existent—not sure of any roadmaps

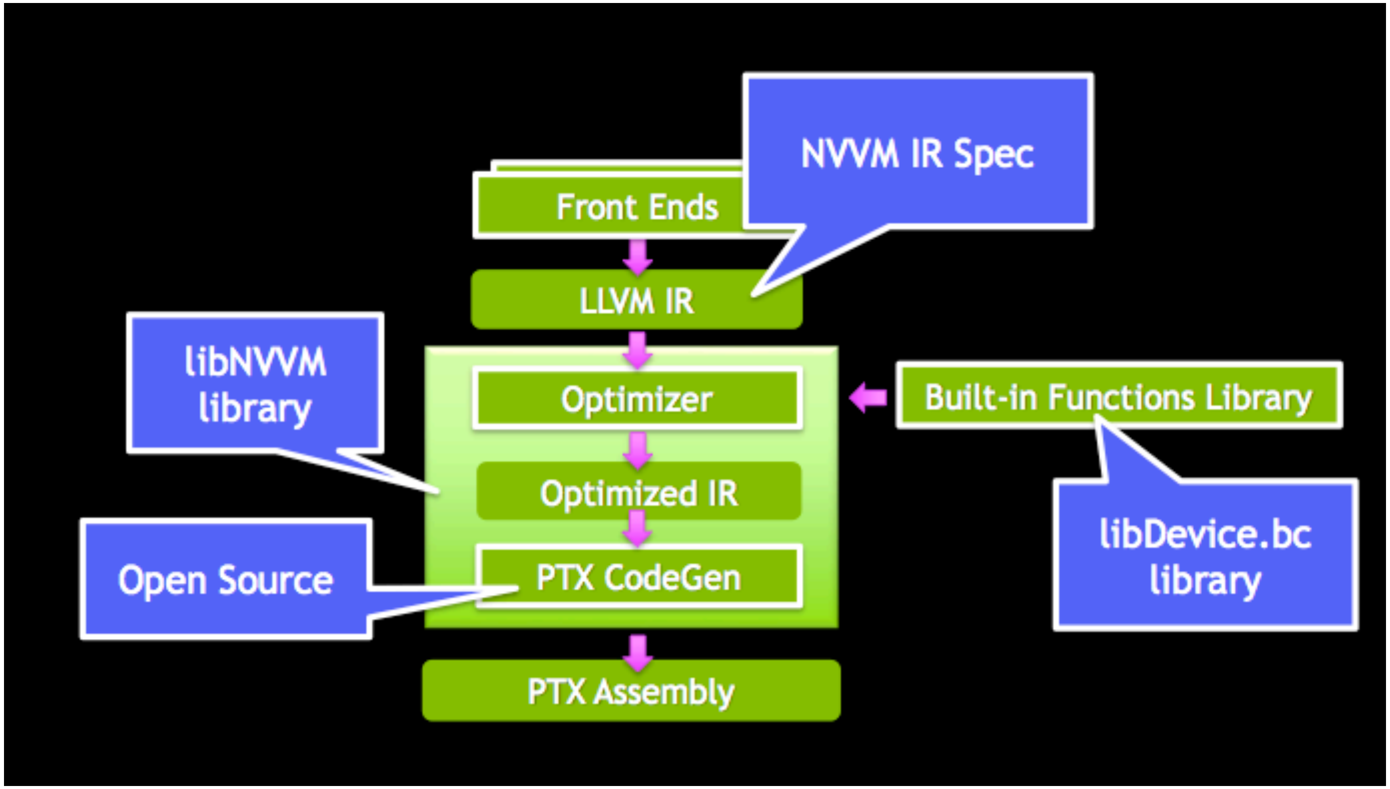
Clang-LLVM Solution



Code Generation in Clang-LLVM

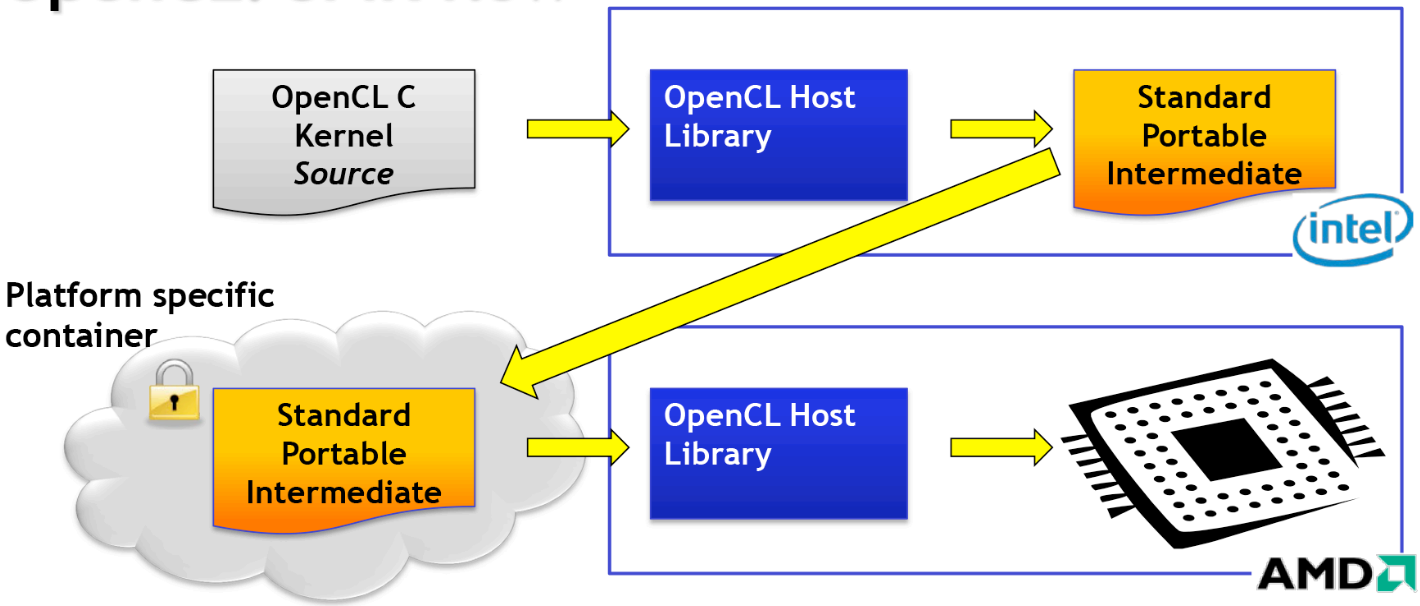


An LLVM Based Compiler Example (I)



OpenCL to IL → SPIR

OpenCL: SPIR flow



- ISV ships kernels in SPIR form
- Customer runs application on platform of their choice

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Who is using SPIR?

The following projects are known to use SPIR:

Project	Description	Project owner
Clang 3.2 patched	OpenCL C 1.2 to SPIR compiler	Khronos Group Inc
SPIR Tools	SPIR module verifier, etc.	Khronos Group Inc
PGI OpenACC	OpenACC to SPIR compiler	Portland Group Inc
Multicoreware C++AMP (on Bitbucket)	C++AMP compiler	Multicoreware Inc

SPIR portability and versioning

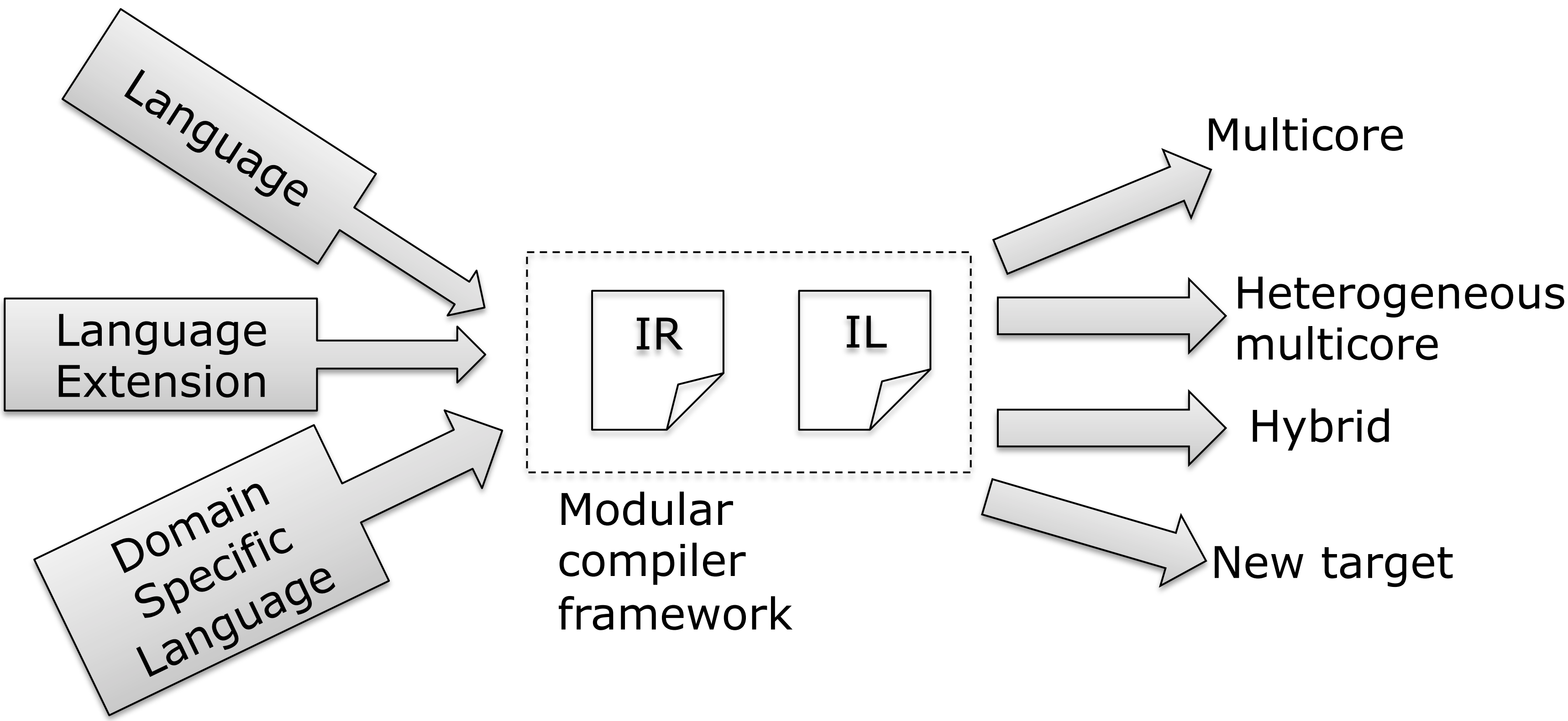
Can I generate a SPIR instance with compiler X and run it on platform Y?

The OpenCL cl_khr_spir runtime API extension assumes you can deliver a binary SPIR IR instance to the clCreateProgramWithBinary API.

The target environment may impose additional requirements before such an application can execute. For example, a target environment may require application pre-verification and cryptographically strong application signing. See the definition of Deployment Ecosystem in the Glossary below.

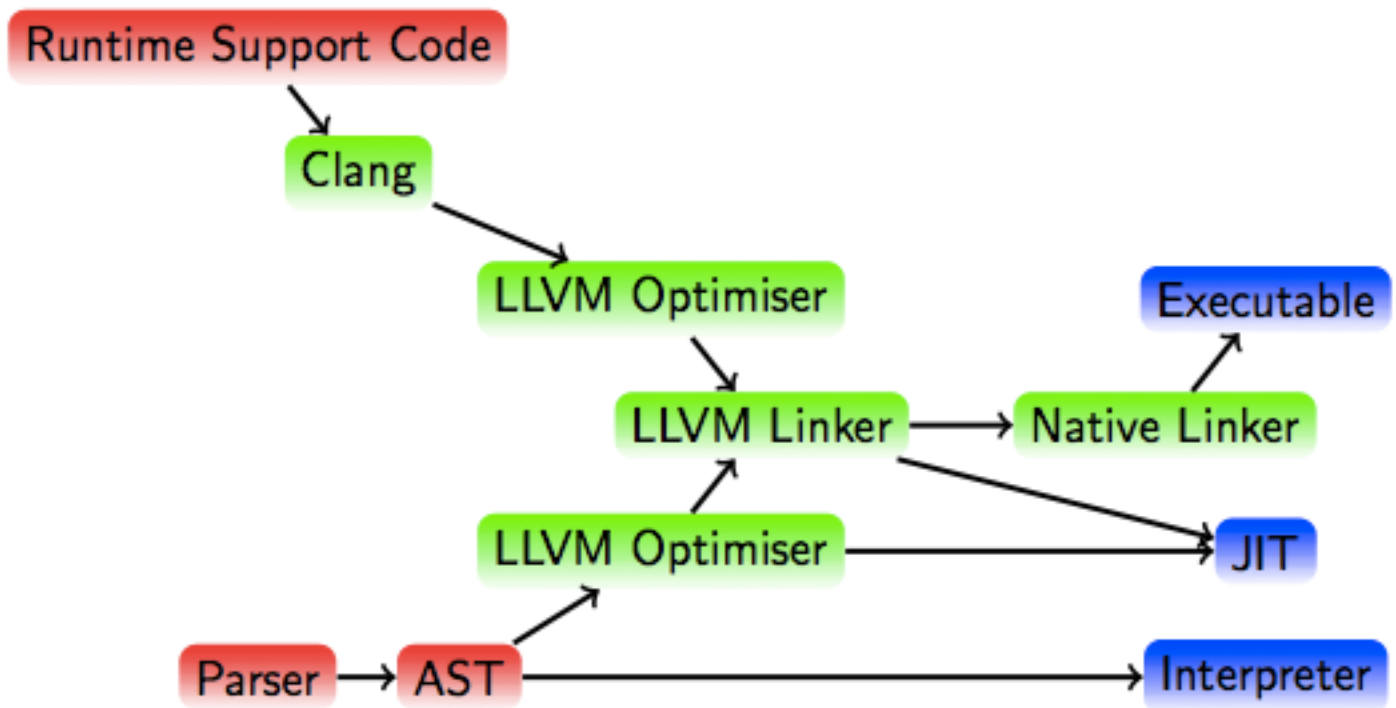
http://www.khronos.org/faq/spir

An Adaptive Environment



Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

A typical DSL implemetation



FOSDEM2012 talk by David Chisnel on
"Implementing Domain Specific Languages with LLVM"

http://cs.swan.ac.uk/~csdavec/FOSDEM12/DSLsWithLLVM.pdf

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

OpenCL 1.2 Code Generation (I)

Step # 1: Conversion of OpenCL code to LLVM IR using the Clang compiler

```
clang -Dcl_clang_storage_class_specifiers -isystem libclc/
generic/include -include clc/clc.h -target nvptx64-nvidia-cuda -
xcl kernel.cl -emit-llvm -S -o kernel.ll
```

Step # 2: Optional

```
llvm link libclc/built_libs/nvptx64   nvidiacl.bc kernel.ll o
kernel.linked.bc
```

OpenCL 1.2 Code Generation (II)

Step # 3: IR to PTX

```
llc -mcpu=sm_35 kernel.ll -o kernel.ptx
```

With the built-in functions:

```
clang -target nvptx64-nvidia-cuda  kernel.linked.bc -S -o
kernel.nvptx.s
```

Step # 4: Write the driver code

```
CC  sample.cpp  -o  sample  -O2  -g  -I/opt/nvidia/cudatoolkit/
5.5.20-1.0501.7945.8.2/include  -L  /opt/nvidia/cudatoolkit/
5.5.20-1.0501.7945.8.2/lib64/ -lcudart

> aprun ./sample
Using CUDA Device [0]: Tesla K20X
Device Compute Capability: 3.5
Launching kernel …
```

Integration into the Cray PE

- **Works with Cray MPI, perftools & libraries (examples in paper)**
- **Full automation tricky—hard coded paths not always discoverable by scripts**
  - E.g. LD_LIBRARY_PATH include paths
- **Binaries for compiler wrappers**
- **Must be straightforward for Cray to come up with PrgEnv-clang?**

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Other Opportunities—Developers tools

- **ClangFormat – formatting tool for C & C++ (integrated in editors)**
- **ClangCheck – performs basic error and warning checks on code snippets. For example**

```
 > cat test.c void foo() { int a = 3 }
$ clang-check test.c --
/scratch/santis/bcumming/test.c:1:23: error: expected ';' at end
of declaration
void foo() { int a = 3 }                ^
```

- **ClangModernize –automatically converts C++ code to C++11, with support for features such as converting for loops to range-based loops and using the auto keyword.**

Other Opportunities—Debug tools

- **Clang Static Analyzer – find bugs without running the code**
- **`$ scan-build make`**

```
int main(void) {
   int *ptr = nullptr;
   ptr[10]++;
}
```

```
test.cc:3:5: warning: Array access
(from variable 'ptr') results in a null
pointer dereference
   ptr[10]++;
      ^~~~~~~~~~
1 warning generated.
scan-build: 1 bugs found.
```

Other Opportunities—Debug tools

```
int main(void) {
    int *ptr = nullptr;
```

1 'ptr' initialized to a null pointer value →

```
    ptr[10]++;
```

2 ← Array access (from variable 'ptr') results in a null pointer dereference

```
}
```

Other Opportunities—Sanitizers

- **AddressSanitizer—fast memory error detector (`-fsanitize=address`)**
  - Out-of-bounds accesses to heap, stack and globals
  - Use-after-free
  - Use-after-return (to some extent)
  - Double-free, invalid free
  - Memory leaks (experimental)

- **MemorySanitizer—detects uninitialized reads (`-fsanitize=memory`)**

- **ThreadSanitizer—detects race conditions (`-fsanitize=thread`)**

RFE for Cray & collaboration opportunities with other sites

- **PrgEnv-Clang, please!**
- **OpenACC to OpenCL SPIR via Intel**
- **After PrgEnv-Clang, solution for OpenCL on GPU, CPU & beyond …**
- **Domain specific languages using LLVM**
- **OpenCL tools for parallel computing**
- **DSL (e.g. poloyglot: http://www.exmatex.org/prog-models.html)**
- **Incremental development on Cray (Co-design summer school: http://codesign.lanl.gov/summer-school/)**
  – Reduce development to deployment time on Cray systems

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Thank you