Orchestrating a brighter world NEC

# Open Source LLVM for SX-Aurora

Simon Moll, NEC Deutschland

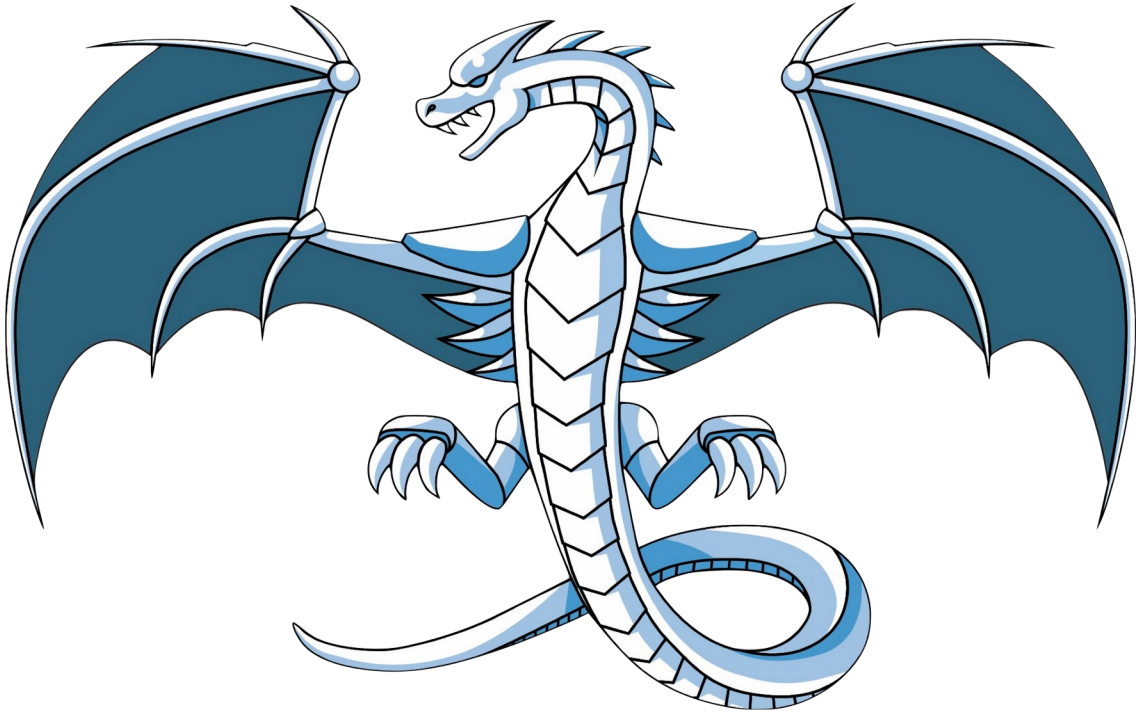
ISC High Performance 2021 Digital, Aurora Forum, 2021.06.30

# LLVMs of SX-Aurora

- **LLVM-VE**

  Open Source

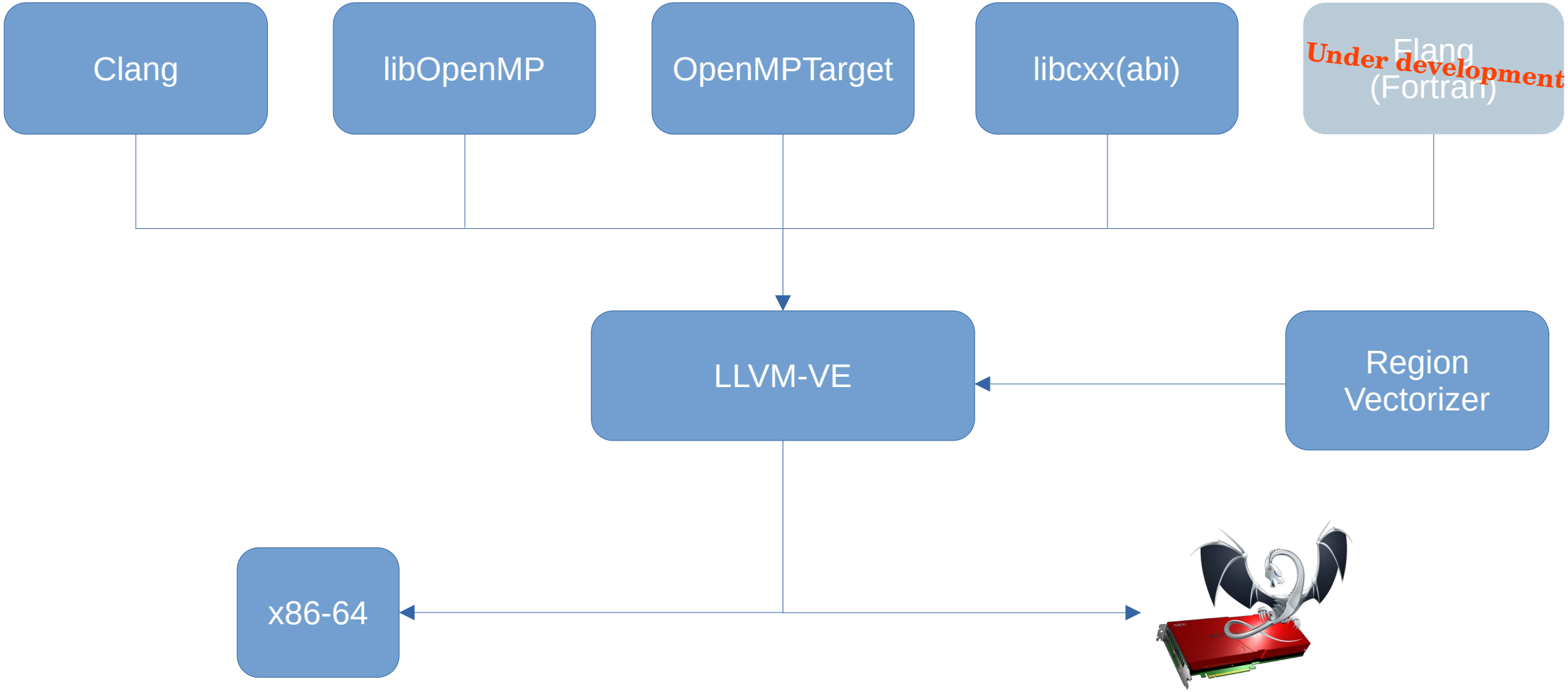
  *Inofficial* compiler for SX-Aurora



- `llvm-vec` NEC LLVM-IR Vectorizer

  Official compiler

# LLVM-VE ecosystem



# Vectorization

# Vectorization in LLVM-VE

- `Clang –target=x86_64-unknown-linux-gnu`
  - Uses LLVM upstream vectorizers (LV)
    - Inner loops only
  - Automatic vectorization

- `Clang –target=ve-linux`
  - Uses the *Region Vectorizer*
    - Outer loop vectorization
  - Automatic vectorization
  - Best controlled with pragmas

# How to vectorize

- `#pragma omp simd [ simdlen(256) | simdlen(512) ]`
  - Vectorize this loop
  - [Optional] hint for normal (256 wide) or packed mode (512 wide) vectorization

- `#pragma omp parallel`
  - May trigger vectorization (details next slides)

- Some unannotated loops - Automatic vectorization
  - Automatic parallel loop detection and vectorization

# Region Vectorizer - Outer-loop vectorization

```
#pragma omp simd
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
    if (A[i] > 42.0)
      <do stuff>
    if (C[j])
      <do that other thing>
```

Vanilla LLVM cannot vectorize this:

- Outer loop
- Control flow (if statements) inside

Region Vectorizer can

- Will retain uniform branch in `C[j]`

# Controlling Vectorization

```
#pragma omp parallel for
for (int i = 0; i < n; ++i)
  [..]
  #pragma omp simd
  for (int j = 0; j < m; ++j)
    [..]
```

Parallel execution

Vectorized

# Controlling Vectorization

```
#pragma omp parallel for
for (int i = 0; i < n; ++i)
  [..]

  for (int j = 0; j < n; ++j)
    [..]
```

Parallel execution

May still vectorize, if

- Loop parallelism detected
- Better score than pragma parallel loop

# Diagnostics

```
clang -Rpass=rv –target=ve-linux -O3 clenshaw.c [..]
```

```
clenshaw.c:6:3: remark: Loop vectorized (width 256) with dynamic VL [-Rpass=rv-loopvec]
  #pragma omp simd
  ^
```

# Adaptive math vectorization

```
void
foo (double x, ..) {
  #pragma omp simd
  for (int i = 0; i < n; ++i)
    A[i] = pow(B[i], x);
}
```

```
double pow(double, double) → double pow_vu(double256, double)
```

# `energy_ia` Compute Kernel

- LLVM-VE + Packed Mode

| `energy_ia` | **LLVM-VE-RV (not packed)** | **LLVM-VE-RV (packed)** | **NCC 3.0.6*** |
|---|---|---|---|
| Runtime [ms] | 0.284 | 0.188 | 0.437 |
| Speedup [rel to ncc] | 35% | 56% | - |

clang –target=ve-linux -fopenmp-simd -O3 -ffast-math

ncc -ffast-math (version 3.0.6)

sx-at-test version of energy_ia

NEC SX-Aurora VE10B – one thread

# OpenMP [Target]

# LLVM OpenMP

- libOpenMP `#pragma omp parallel`
  - Implements parallel loops,barriers and reductions
  - Linked against the device code
    - Either as part of VE-native application
    - Or VE-native kernels

- libOpenMPTarget `#pragma omp target`
  - Performs the kernel dispatch, buffer transfers
  - Linked against the host application
  - Plugin mechanism for actual offload (see Tim Cramer's talk)

# LLVM OpenMP

- Generic LLVM OpenMP library compiled for VE

```
#pragma omp parallel
```

- Pro: Mature OpenMP implementation

  Standard OpenMP runtime for x86 for Clang/LLVM

- Con: Not tuned for vector architectures

  Based on pthreads, standard synchronization primitives (futex), not hw features

# OpenMP – EPCC Syncbench

| [µs] | **OpenMP (ncc)** | **LLVM OpenMP (VE)** | **LLVM OpenMP (x86)** |
|---|---|---|---|
| parallel for | 6.77 | 724.4 | 7.27 |
| barrier | 3.74 | 309.8 | 1.87 |
| reduction | 7.01 | 608.5 | 7.5 |

NEC OpenMP is fast  LLVM OpenMP needs tuning

# LLVM OpenMPTarget

```
#pragma omp target
```

- AVEO plugin
  - VH → VE Offloading
- VHCall plugin
  - VE → VH Offloading

- SOLLVE OpenMP Target Verification suite

# OpenMP Target - SOLLVE C

| | **VH → VE** | **VE → VH** | **VH → VE** (sotoc) |
|---|---|---|---|
| Compile Error | 0 | 0 | 7 |
| Runtime Error | 11 | 8 | 11 |
| **Passed** | 98 | 101 | 91 |
| | Conformant LLVM code path | | Custom source-to-source |

# OpenMP Target - SOLLVE C++

| | **VH → VE** | **VE → VH** | **VH → VE** (sotoc) |
|---|---|---|---|
| Compile Error | 0 | 0 | 14 |
| Runtime Error | 1 | 2 | 0 |
| **Passed** | 13 | 12 | 0 |
| | Conformant LLVM code path | | Source-to-source for C only |

# LLVM Test suite

# LLVM Test Suite

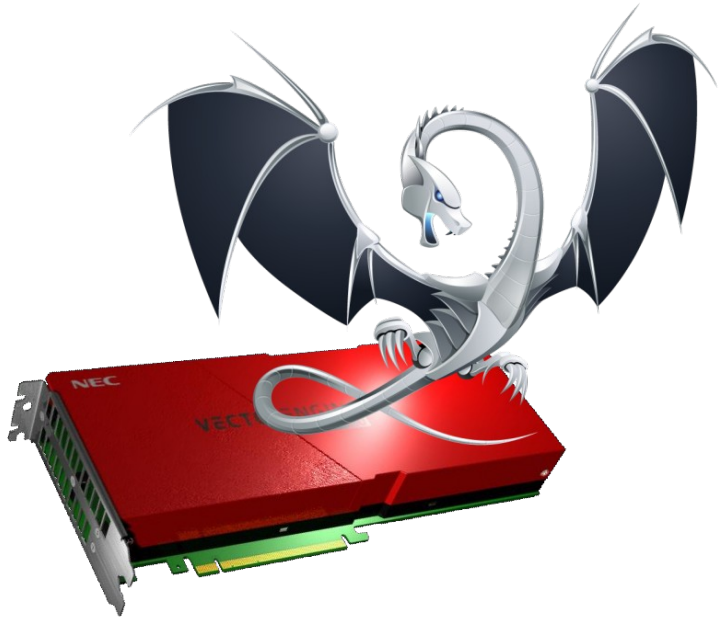
- C, C++ (-O3 , -ffast-math)
  - 432 applicable C/C++ tests (for both NCC and Clang)

| | **LLVM-VE-RV dev** | **NCC 3.0.6*** |
|---|---|---|
| Compile fail | 2 | 100 |
| Compile pass | 430 (99.5 %) | 332 (77 %) |
| Test fail | 57 | 57 |
| **Test pass** | **373 (86%)** | **275 (64%)** |

* translating clang/gcc options into ncc options (wrapper script)

# Future Work

- Improving LLVM's support for vector architectures
- Tuning of OpenMP Runtime
- Better Code Generation
- Making VE an official LLVM Backend




`github.com/sx-aurora-dev/llvm-project`


Orchestrating a brighter world
NEC