Packed-Mode Vectorization LLVM for SX-Aurora

Simon Moll, NEC Deutschland

LLVM(s) for SX-Aurora

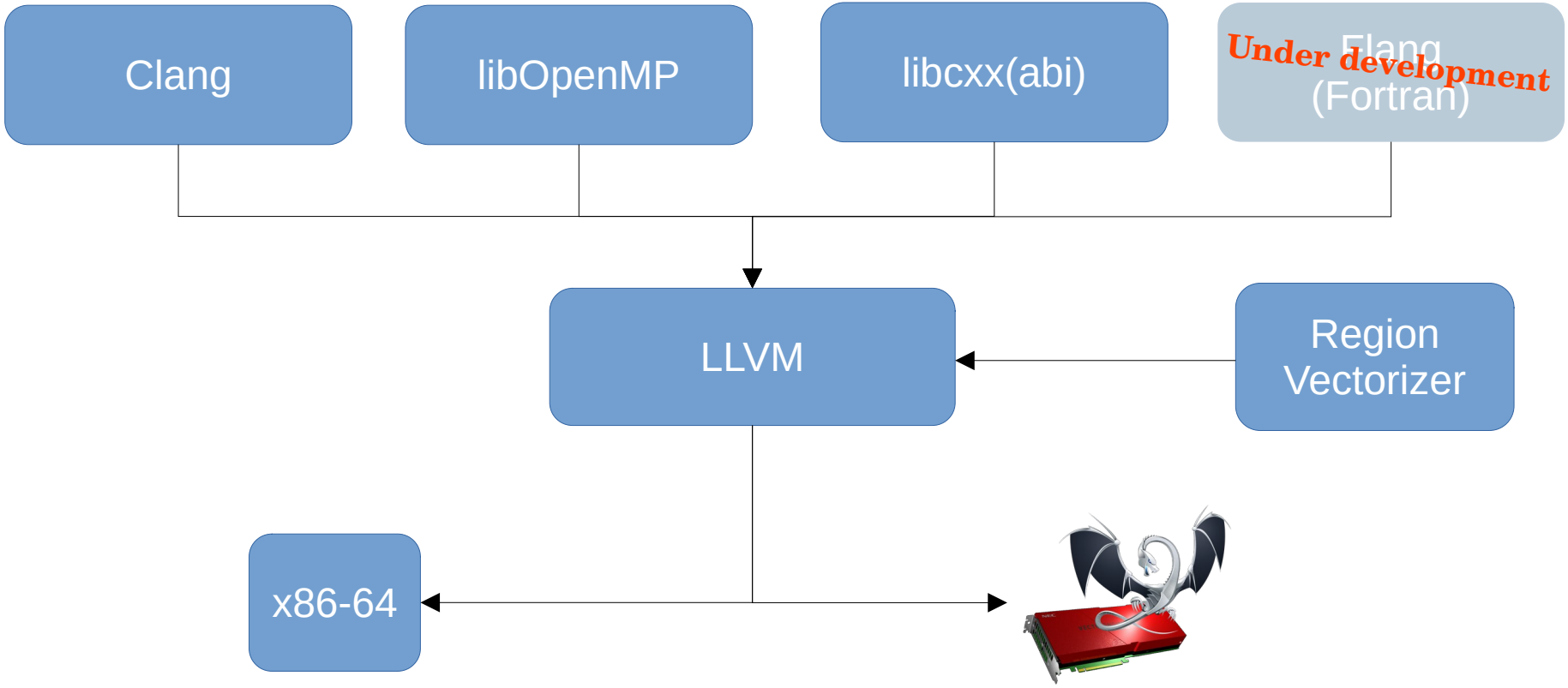
LLVM-VE (DS Labs)

- Scalar code
- Vector intrinsics

LLVM-VE-RV ← *(Talking about this one)*

- Vectorizing LLVM compiler
- Based on LLVM-VE

Project Structure



\Orchestrating a brighter world

NEC

Features

OpenMP

- OpenMP Target Offloading
  - From VH to VE
  - From VE to VH
- #pragma omp parallel
- #pragma omp simd
  - Outer-loop vectorization
- Available for C/C++ now, Fortran coming up

Packed mode

- `#pragma omp simd`
  - Internal heuristics
- `#pragma omp simd simdlen(256)`
  - Use 256 elements (not packed)
- `#pragma omp simd simdlen(512)`
  - .. AVL refers to 256 elements (packs of 2 x f32)

Preview: Some (automatic) Outer-Loop Vectorization

Command line

```
clang -mllvm -rv-autovec -O3
```

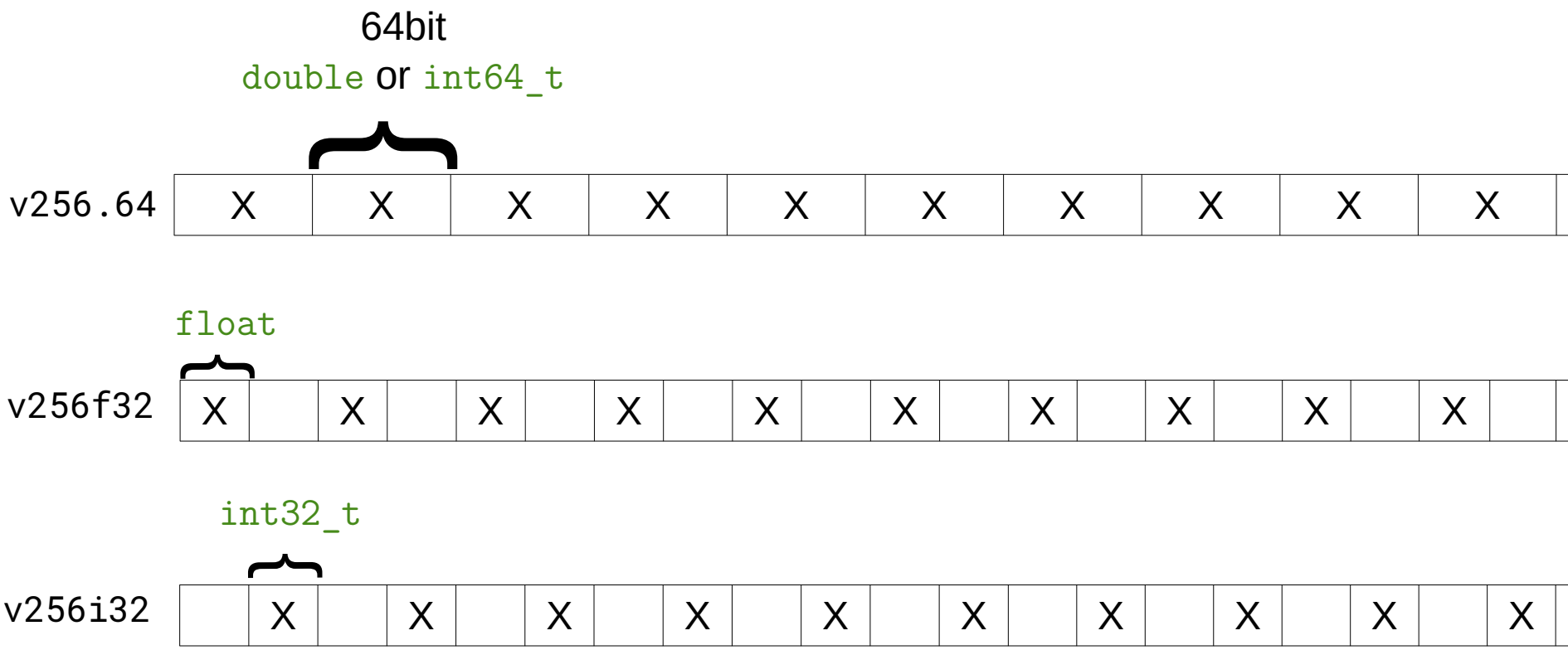
```
#pragma omp simd
for (int64_t i = 0; i < n; ++i) {
  for (int j = 0; j < m; ++j)
    if (A[j] > 42) // <scalar code>
}
```

\Orchestrating a brighter world

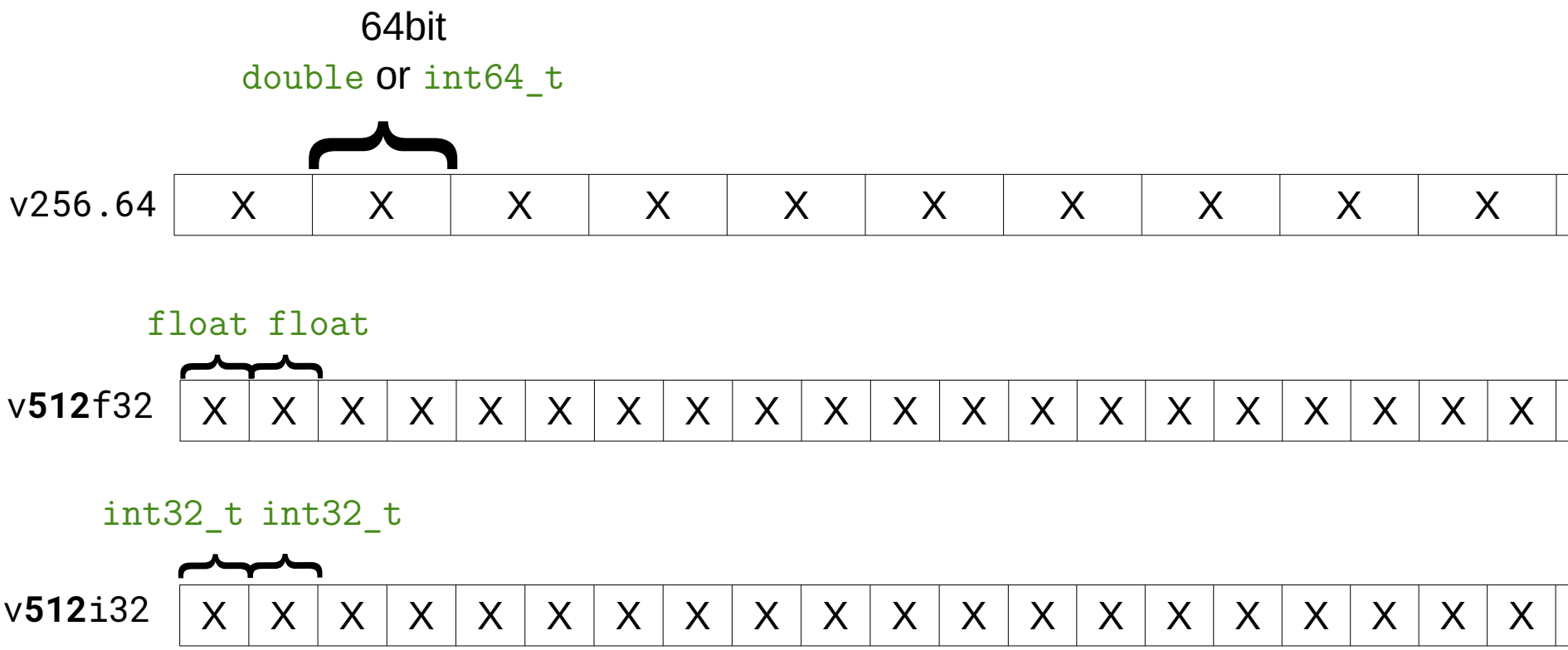
NEC

Packed Mode

Data in Vector Registers



Packed Data in Vector Registers



Packed Mode Irregularity

VE Vector ISA is 64bit centric

- Dynamic VL step size is 2x32bit
- No (real) Packed Vector Load / Store
- Splitting for missing packed instructions

...

Dynamic VL

Dynamic VL step size is 64bit

```
lea %s1, 17
lvl %s1                    // Sets VL to 17
VFADD %vr, %vx, %vy        // f64 vector add
```

Computes %vr[0] to %vr[16] // 17 elements

Packed Dynamic VL

Dynamic VL step size is ~~64bit~~ 2x32bit

```
lea %s1, 17
lvl %s1                    // Sets VL to 17
PVFADD %vr, %vx, %vy       // Packed f32 vector add
```

Computes %vr[0] to %vr[33] // 2 x 17 elements

=> Workarounds for odd-iteration count loops

No (versatile) Packed VLD / VST

Abusing VST for (non-existent) PVST

PVST i64:%BasePtr, v512f32:%Data, v512i1:%Mask

(does not exist)

VST i64:%BasePtr, v512f32:%Data, **v256i1**:%Mask

How do you do a packed vector store then?

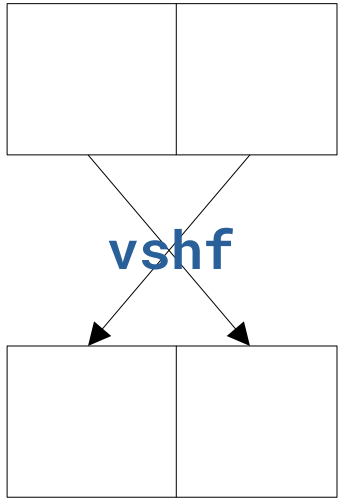
If stars don’t align => split it aka you don’t

Splitting Overhead

```
pvfdiv.s %v0, %v1, %v2, %vm
(does not exist)
```

Hypothetical
Packed float vector division

```
vfdiv.s %v0, %v1, %v2, %vm_hi
vshf %v1, %v1, %v1, 4
vshf %v2, %v2, %v2, 4
vfdiv.s %v1, %v1, %v2, %vm_lo
vshf %v0, %v1, %v2, 8
```



Splitting Overhead with odd VL

```
            // %s0 holds the (packed) element count
add %s1, %s0, 1
srl %s0, %s0, 1
srl %s1, %s1, 1
lvl %s1
vfdiv.s %v0, %v1, %v2, %vm_hi
lvl %s0
vshf %v1, %v1, %v1, 4
vshf %v2, %v2, %v2, 4
vfdiv.s %v1, %v1, %v2, %vm_lo
vshf %v0, %v1, %v2, 8
```

10 Ops instead of 1

5 Vops instead of 1
(7 if `lvl` counts as vop)

(!!!)

Details, Details, Details

(Ab)using VLD/VST for packed mode

- VST/VLD alignment is 8 Bytes
  - What if we load/store `float*` ?

- Byte-order mismatch between `int64_t` and `int32_t[2]`
  ABI <> Memory Layout

- No masked VLD (not only in packed mode)

\Orchestrating a brighter world

NEC

Results

Packed Mode Results

Re: Thursday's Part by Erich Focht

LLVM-VE-RV + Outer-Loop + 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

Evaluation - Correctness

LLVM Test suite (-O3, -ffast-math)

- C/C++ tests:
  - DoE Proxy Apps (C/C++)
  - Benchmarks (scimark, fftbench, TSVC, ..)
  - Codecs (jpeg, mpeg, ogg, ..)
  - Applications (ClamAV, SPASS, ..)
  - UnitTests (Clang regression tests)

Passing Rate

LLVM Test suite (-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)

\Orchestrating a brighter world

NEC

Outlook

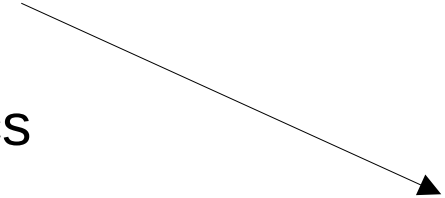
Merging (DONE as of 2021-03-18)

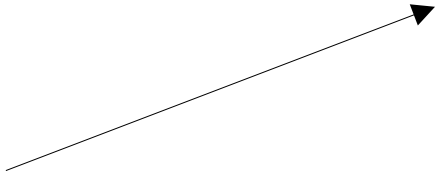
LLVM-VE (DS Labs)

- Scalar code
- Vector intrinsics

LLVM-VE-RV

- Vectorizing LLVM compiler
- Based on LLVM-VE





LLVM for SX-Aurora

Merging means

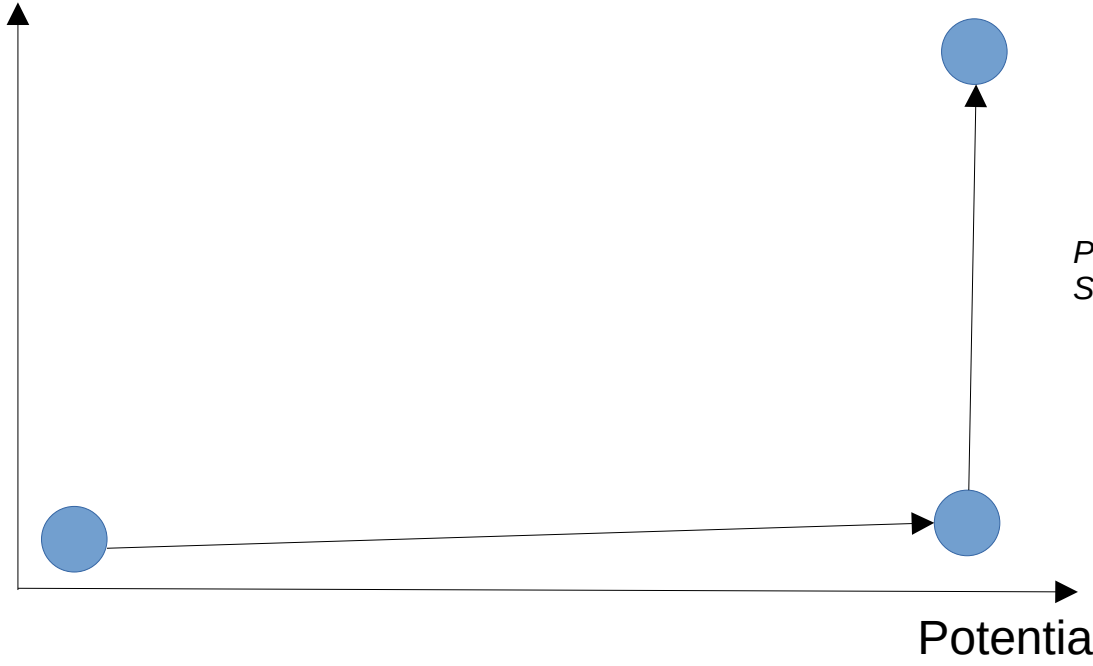
- joint forces

- less politics

Development Strategy LLVM-VE-RV

Do Things Fast

Get Speedup of xN on benchmark Y



Do Things Right

Packed-mode Support

Pass the LLVM test suite

Loop nest Transformations

Stuff I didn’t talk about

- **Upstreaming VE target to LLVM**
- **LLVM-VP (aka make LLVM a better place for vector)**
  - **Interest picking up.. SiFive , IBM, ..**
- **On-going work for loop nest transformations**
  - **Stefanos Baziotis (intern/subcontractor)**
- **Uncertainty of LLVM-VE development**
  - **LLVM for SX-Aurora is *inofficial***

Frontends

- SYCL
- Julia
- OpenCL / SPIR-V / VulkanCompute
- MLIR

Conclusion

- LLVM stack keeps improving
- Packed Mode is the default
- Many opportunities
- Give it a try!
  https://github.com/sx-aurora-dev/llvm-project

Questions?