OpenMP on the NEC SX-Aurora Vector Engine

Erich Focht, Simon Moll, NEC Deutschland

OpenMPCon 21
Vector Engine Card

**Air Cooled Card**
- Two types of packages

**Passive Cooling**
For Servers

**Active Cooling**
For Tower/Workstation

**Water Cooled Card**
- Direct liquid cooling
- Hot water cooling available

Direct Liquid Cooling
For Supercomputer

40°C/104°F water
# VE20 Processor Specifications

## VE20 Specifications

<table>
<thead>
<tr>
<th>Feature</th>
<th>Type 20A</th>
<th>Type 20B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Version</td>
<td>Type 20A</td>
<td>Type 20B</td>
</tr>
<tr>
<td>Cores/processor</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Core performance</td>
<td>307GF (DP)</td>
<td>614GF (SP)</td>
</tr>
<tr>
<td>Processor performance</td>
<td>3.07TF (DP)</td>
<td>2.45TF (DP)</td>
</tr>
<tr>
<td></td>
<td>6.14TF (SP)</td>
<td>4.91TF (SP)</td>
</tr>
<tr>
<td>Cache capacity</td>
<td>16MB</td>
<td></td>
</tr>
<tr>
<td>Cache bandwidth</td>
<td>3TB/s</td>
<td></td>
</tr>
<tr>
<td>Cache Function</td>
<td>Software Controllable</td>
<td></td>
</tr>
<tr>
<td>Memory capacity</td>
<td>48GB</td>
<td></td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td>1.53TB/s</td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>~300W (TDP)</td>
<td>~200W (Application)</td>
</tr>
</tbody>
</table>

Diagram showing the processor design with HBM2 memory and core configurations.
Long Vector = SIMD + Pipelining

SIMD

Example: 
\[ \text{do } i=1, N \\
\text{c}(i) = a(i) + b(i) \\
\text{enddo} \]

Vector

Example: 
\[ \text{do } i=1, N \\
\text{c}(i) = a(i) + b(i) \\
\text{enddo} \]
Vector Engine Core

Vector Processing Unit (VPU)
- Powerful
  - 307.2GFLOPS DP / 614.4GFLOPS SP performance
- High bandwidth memory access
  - 409.6GB/sec Load and Store

Scalar Processing Unit (SPU)
- Provides basic processor functions
  - Fetch, decode, branch, add, exception handling, etc...
- Controls the status of complete core

Address translation and data forwarding crossbar
- To support contiguous vector memory access
  - 16 elements/cycle vector address generation and translation, 17 requests/cycle issuing
  - 409.6GB/sec load and 409.6GB/sec store data forwarding
Vector Processing Unit

- Four pipelines, each 32-way parallel
  - FMA0: FP fused multiply-add, integer multiply
  - FMA1: FP fused multiply-add, integer multiply
  - ALU0/FMA2: Integer add, multiply, mask, FP FMA
  - ALU1/Store: Integer add, store, complex operation

- Doubled SP performance by 32bit x 2 packed vector data support

- Vector register (VR) renaming with 192 physical VRs
  - 64 architectural VRs are renamed
    - Enhanced preload capability
    - Avoidance of WAR and WAW dependencies

- OoO scheduling

- Dedicated complex operation pipeline to prevent pipeline stall
  - Vector sum, divide, mask population count, etc.
Vector Engine Card

Standard PCIe card

- Full-length full-height card
- Dual slot
- <300W power

High sustained performance

- High B/F ratio (0.62 for VE20B), good balance
- ~6% HPCG performance efficiency

TCO reduction

- Low power consumption: long vectors
- Increased productivity (programming, code maintenance)
SX-Aurora TSUBASA: VE Native Mode

- Programs run on VE in a Linux environment
  - Accelerator that doesn’t behave like an accelerator

Aurora architecture

VE OS is “offloaded”

VEOS features:
- VE process management
- VE memory management
- VE program loading
- System call handling
- Signal handling
- OS commands support
  - gdb, ps, free, top, sar etc.

No OS jitter on Vector Engine
VEOS works completely on Linux/x86.
VE Native Programming Model

- **C / C++ / Fortran**
  - With any syscalls
  - Proprietary: NEC ncc, nc++, nfort

- **OpenMP parallelization**
  - Inside one VE: 8-10 cores
  - With NUMA mode: 4-5 cores

- **MPI**
  - Infiniband, EDR or HDR
  - PeerDirect VE to VE

- **Hybrid MPI**
  - Same program compiled for VE and VH, with NEC MPI
NEC ncc/nc++/nfort OpenMP

◆ Compliant to OpenMP 4.5
  - With restrictions
    • eg. Cancellation constructs, thread affinity control, array sections, certain clauses
  - Environment variables with VE_prefix take precedence
    • VE_OMP_NUM_THREADS overrides OMP_NUM_THREADS
    • i.e. control threads of host process and VE process separately.

◆ Biggest limitation: no Device Constructs
  - No OpenMP Target for accelerator use!
  - Remember: VE primarily aimed at native programming
  - NEC ncc/nc++/nfort are pure VE cross compilers
    • No code generation capabilities for the host CPU
But... VE Is Actually an Accelerator!

- While code can be ported as if we’d be using a general purpose CPU
  - Some functions won’t vectorize
  - Or have too short vector length
But... VE Is Actually an Accelerator!

- Run main program on VH and offload VE kernels to VE...
  - Like OpenCL or CUDA
  - No mechanism available on OS-less device

- VEOffloading
  - Born 2015, evolved to AVEO in 2020.

- Core mechanism for accelerator style programming

- VEDA: Vector Engine Device API
  - Implements CUDA alike API
  - VEDA: https://docs.nvidia.com/cuda/cuda-driver-api/
  - VERA: https://docs.nvidia.com/cuda/cuda-runtime-api/
OpenMP Target: First Steps

- Cooperation project with RWTH Aachen
  - Started 2017
  - Tim Cramer, Manoel Römmer, Boris Kosmynin, Marius Behrens. Erich Focht

- Leveraging LLVM/Clang infrastructure
  - Source-to-source transformation, C language support
  - Target compiler independent
  - Plugin for host – VE communication using VEO

First evaluation results: Parallel Processing and Applied Mathematics, PPAM 2019: OpenMP Target Device Offloading for the SX-Aurora TSUBASA Vector Engine, Tim Cramer et al.
LLVMs of SX-Aurora

◆ LLVM-VE
  - Started 2018. Open Source!
  - *Inofficial* compiler for SX-Aurora
    • Experimental → upstreaming!
  - Highlights
    • Vector Intrinsics
    • Region Vectorizer: outer loop!
    • OpenMP Target

◆ llvm-vec NEC LLVM-IR Vectorizer
  - Based on proprietary NEC compiler
  - First test version: May 2021

[github.com/sx-aurora-dev/llvm-project]
LLVM-VE ecosystem

Clang
libOpenMP
OpenMPTarget
libcxx(abi)

Region Vectorizer

x86-64
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

```c
#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

```c
#pragma omp parallel for
double sum = 0;
for (int i = 0; i < n; ++i)
    sum += a[i];
#pragma omp parallel for
for (int j = 0; j < m; ++j)
    sum += b[j];
```

Parallel execution

```
#pragma omp simd
for (int j = 0; j < m; ++j)
    sum += b[j];
```

Vectorized
Controlling Vectorization

```c
#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
Adaptive math vectorization

```cpp
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)
```
OpenMP [Target]
```c
int data[N];
#pragma omp target data map(tofrom: data)
{
    #pragma omp target
    for (int i = 0; i < N; i++)
        data[i] *= 2;

    #pragma omp target update from(data)
    // ...

    #pragma omp target
    for (int i = 0; i < N; i++)
        data[i] += 5;
}
```

- Where does main() run?
  - Vector host (VH)
  - Vector engine (VE)

- Which compiler
  - For offloaded part
  - For main() part
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
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, not hw features
## OpenMP – EPCC Syncbench

<table>
<thead>
<tr>
<th></th>
<th>OpenMP (ncc)</th>
<th>LLVM OpenMP (VE)</th>
<th>LLVM OpenMP (x86)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel for</td>
<td>6.77</td>
<td>724.4</td>
<td>7.27</td>
</tr>
<tr>
<td>barrier</td>
<td>3.74</td>
<td>309.8</td>
<td>1.87</td>
</tr>
<tr>
<td>reduction</td>
<td>7.01</td>
<td>608.5</td>
<td>7.5</td>
</tr>
</tbody>
</table>

NEC OpenMP is fast  LLVM OpenMP needs tuning
LLVM OpenMPTarget

```c
#pragma omp target
```

- AVEO plugin
  - VH $\rightarrow$ VE Offloading
- VHCall plugin
  - VE $\rightarrow$ VH Offloading
- SOLLVE OpenMP Target Verification suite
## OpenMP Target - SOLLVE C

<table>
<thead>
<tr>
<th></th>
<th>VH $\rightarrow$ VE</th>
<th>VE $\rightarrow$ VH</th>
<th>VH $\rightarrow$ VE (sotoc)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compile Error</strong></td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td><strong>Runtime Error</strong></td>
<td>11</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td><strong>Passed</strong></td>
<td>98</td>
<td>101</td>
<td>91</td>
</tr>
</tbody>
</table>

- Conformant LLVM code path
- Custom source-to-source
## OpenMP Target - SOLLVE C++

<table>
<thead>
<tr>
<th></th>
<th>VH $\rightarrow$ VE</th>
<th>VE $\rightarrow$ VH</th>
<th>VH $\rightarrow$ VE (sotoc)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compile Error</strong></td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td><strong>Runtime Error</strong></td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><strong>Passed</strong></td>
<td>13</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

Conformant LLVM code path

Source-to-source for C only
## OpenMP Target: SPEC ACCEL

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>x86→SX (NEC 3.0.8)</th>
<th>x86→V100 (LLVM 12)</th>
<th>x86→V100 (GCC 9)</th>
<th>x86→x86</th>
<th>SX→x86</th>
</tr>
</thead>
<tbody>
<tr>
<td>503.postencil</td>
<td>21.3*</td>
<td>16.8</td>
<td>145</td>
<td>103</td>
<td>137</td>
</tr>
<tr>
<td>504.polbm</td>
<td>18.8</td>
<td>36.7</td>
<td>60.1</td>
<td>RE</td>
<td>94.4</td>
</tr>
<tr>
<td>514.pomrq</td>
<td>321*</td>
<td>31.4</td>
<td>RE</td>
<td>11822</td>
<td>11976</td>
</tr>
<tr>
<td>552.pep</td>
<td>1923</td>
<td>71.1</td>
<td>140</td>
<td>639</td>
<td>667</td>
</tr>
<tr>
<td>554.pcg</td>
<td>91.6</td>
<td>88.6</td>
<td>80.2</td>
<td>124</td>
<td>240</td>
</tr>
<tr>
<td>557.pcs</td>
<td>81.6</td>
<td>101</td>
<td>232</td>
<td>167</td>
<td>253</td>
</tr>
<tr>
<td>570.pbt</td>
<td>19.6</td>
<td>486</td>
<td>122</td>
<td>114</td>
<td>154</td>
</tr>
</tbody>
</table>
Conclusion

- Vector Engine is different from other accelerators
  - Can behave like a normal CPU
  - Normal loop based programming in standard languages
- OpenMP is an important tool for parallelizing inside one VE
- LLVM is crucial for OpenMP device offloading
  - Two approaches, two target compilers
  - Two offload directions
- LLVM libomp for VE needs tuning
NEC SX-Aurora TSUBASA : From tower model to DLC

Built with SX-Aurora Vector Engine

- NEC vector processor
- Long vector units: 256 * 64 bit
- 32-way SIMD and 8 cycle deep pipelining
- Huge memory bandwidth
Vector Engine Based Heterogeneous HPC Examples

NIFS : Fusion Science
National Institute for Fusion Science

DWD : Weather / Climate
Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Tohoku University : Academic

JAMSTEC : Earth Science

Photo of the previous system