NEC and RWTH Aachen University collaboration on SX-Aurora TSUBASA

Christian Terboven, Matthias S. Müller, Tim Cramer, Bo Wang, Daniel Schürhoff

November 12th - NEC Aurora Forum, Dallas, TX, USA

IT Center | RWTH AACHEN UNIVERSITY

Motivation

- Our research on parallel programming
  - We contributed a lot to OpenMP
  - RWTH Aachen is member of the OpenMP ARB and Language Committee

- Our work on performance analysis
  - We contributed to performance tools
  - We contributed to performance analysis methodology and frameworks
  - We support a large technical computing user base

- Our work on productivity and efficiency of simulations targeted for supercomputers
  - Porting of applications to the NEC Aurora Architecture

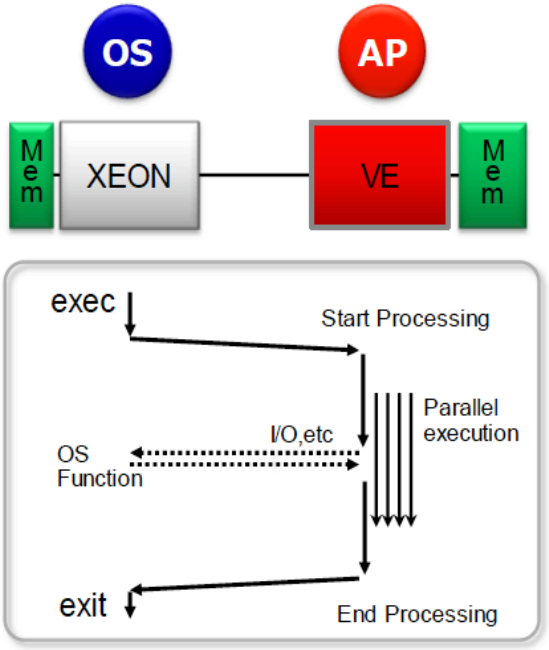
Agenda

- OpenMP Offloading for Aurora
  - Prototype implementation within the LLVM/Clang infrastructure
- Applications for Aurora
  - Screening process
  - Early experiences with applications
- Summary & next steps

OpenMP Offloading for Aurora

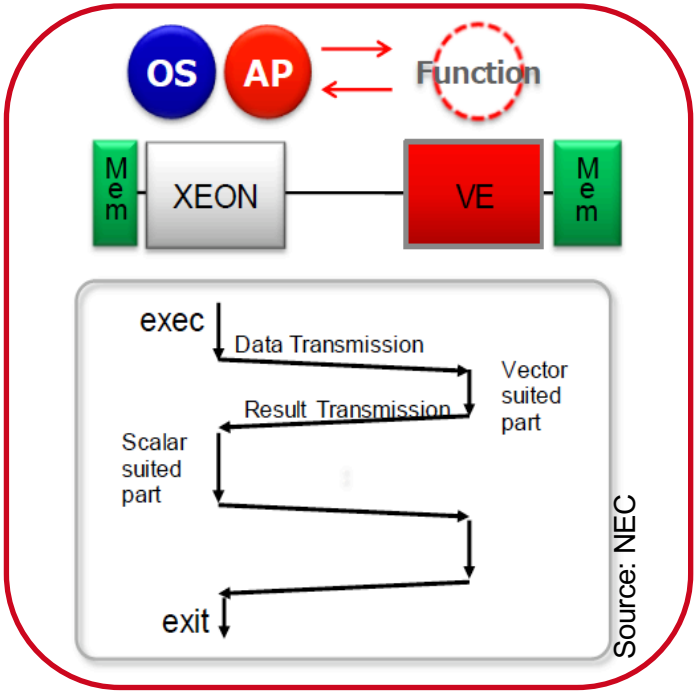
SX-Aurora TSUBASA Execution Models

Aurora *Native* (OpenMP) Execution



- Execute entire (OpenMP) program on Vector Engine
- Good for highly vectorizable applications

Aurora *Offload* (OpenMP) Execution



- Execute scalar suited part of the program on the host processor
- Offload highly parallel parts on the Vector Engine
- RWTH Aachen University is working on a prototype in collaboration with NEC

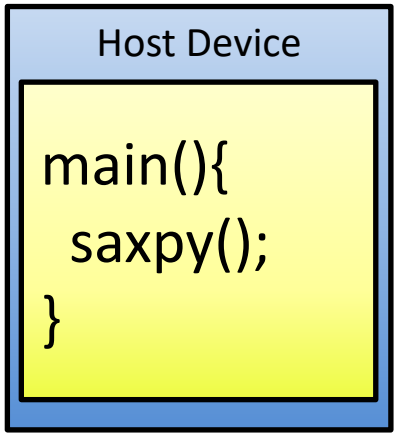
→ Supporting both approaches increases usability

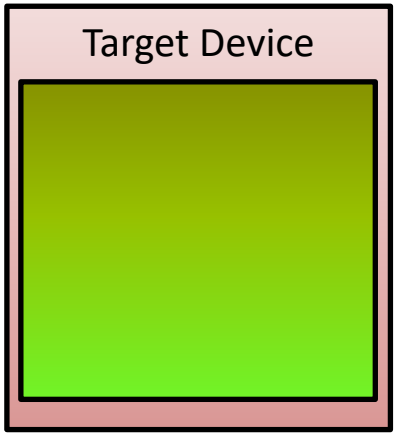
OpenMP Offloading

Target Device Offloading


```
void saxpy(){
  int n = 10240; float a = 42.0f; float b = 23.0f;
  float *x, *y;
  // Allocate and initialize x, y
  // Run SAXPY

  #pragma omp parallel for
  for (int i = 0; i < n; ++i){
    y[i] = a*x[i] + y[i];
  }
}
```



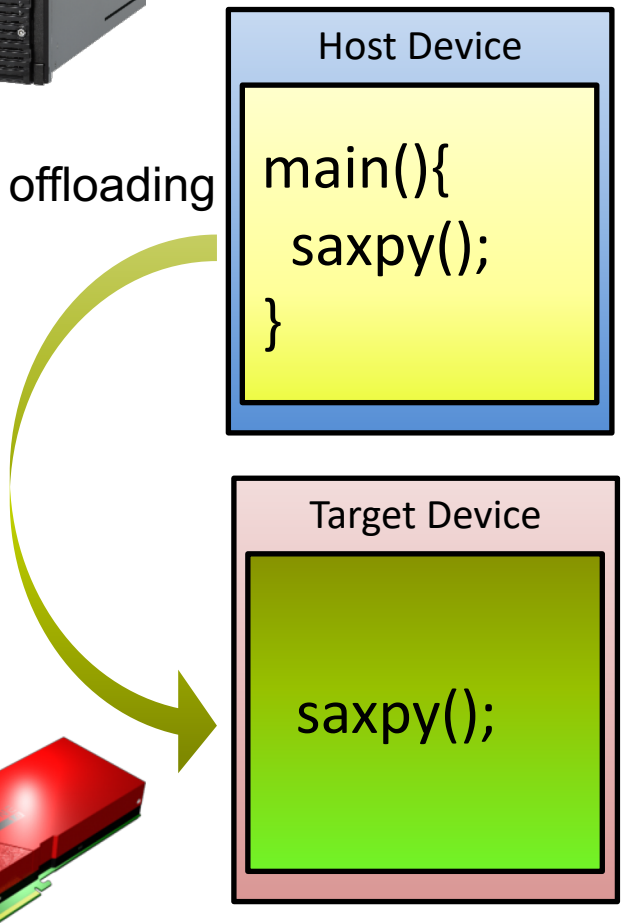


OpenMP Offloading

Target Device Offloading


```
void saxpy(){
  int n = 10240; float a = 42.0f; float b = 23.0f;
  float *x, *y;
  // Allocate and initialize x, y
  // Run SAXPY

  #pragma omp target
  #pragma omp parallel for
  for (int i = 0; i < n; ++i){
    y[i] = a*x[i] + y[i];
  }
}
```

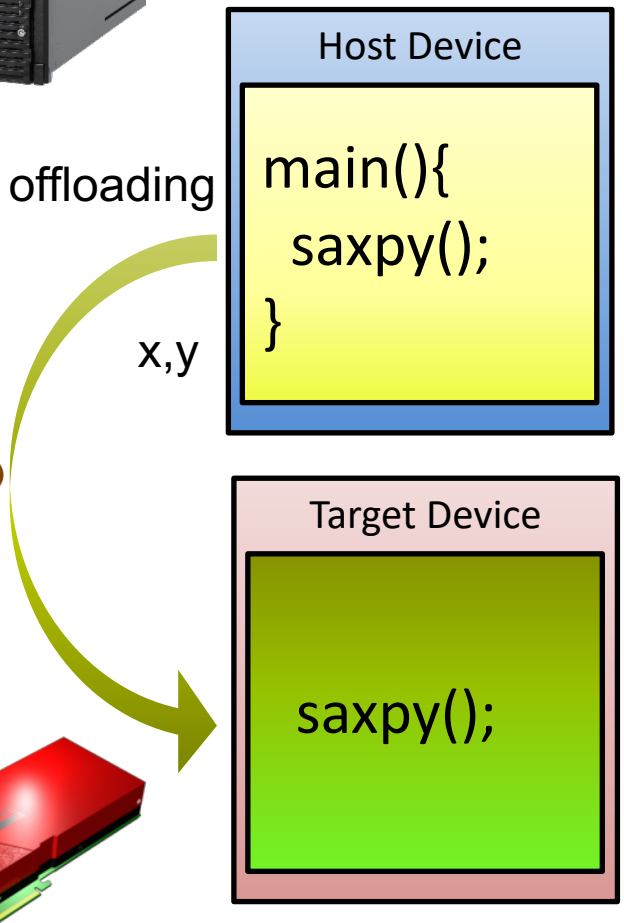


OpenMP Offloading

Target Device Offloading


```
void saxpy(){
  int n = 10240; float a = 42.0f; float b = 23.0f;
  float *x, *y;
  // Allocate and initialize x, y
  // Run SAXPY

  #pragma omp target map(to:x[0:n]) map(tofrom:y[0:n])
  #pragma omp parallel for
  for (int i = 0; i < n; ++i){
    y[i] = a*x[i] + y[i];
  }
}
```



OpenMP Offloading

Target Device Offloading

- How it works: usage of OpenMP Offloading via new target-triple
  - `$ clang -fopenmp -fopenmp-targets=`**`aurora-nec-veort-unknown`** `input.c`
  - Clang driver calls NEC compiler for vector code generation

- Features
  - Full LLVM Integration
  - Adoption for OpenMP 5.0 (already in Clang)
  - Support for most combined constructs
  - Support for macros
  - Compilation of the SPEC Accel benchmarks

- Next Step
  - Release of open source code
  - Evaluation with SPEC Accel benchmarks

Applications for SX-Aurora

Candidate applications at RWTH

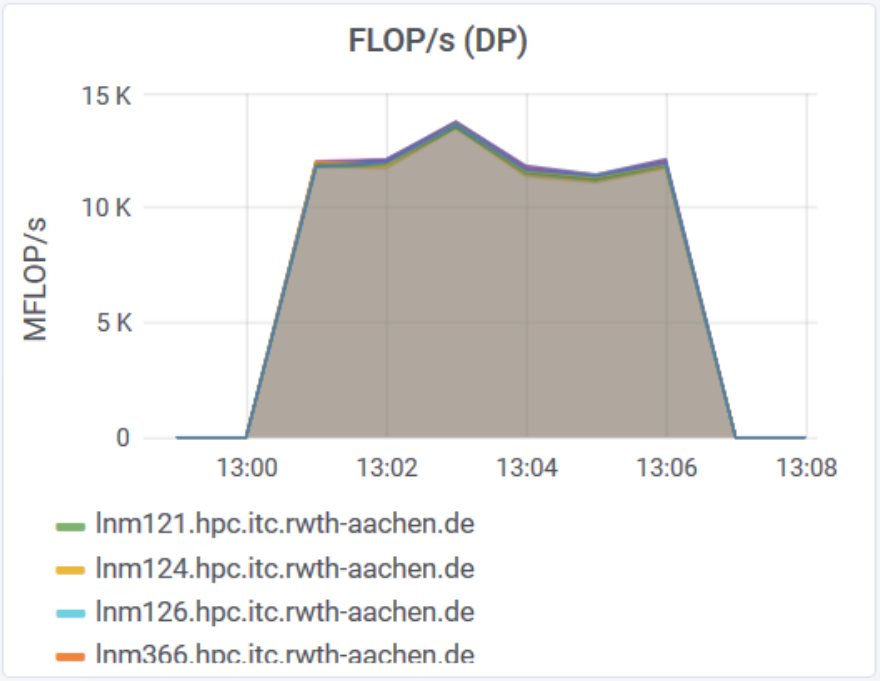
- RWTH Aachen used a representative job mix with (user) codes for procurement
  - Users provided data sets for each code
  - Typical size: 288 MPI processes
    - Reference System: 2x Intel Xeon E5-2650 v4 @ 2.20GHz
  - Total-cost-of-ownership evaluation in the procurement

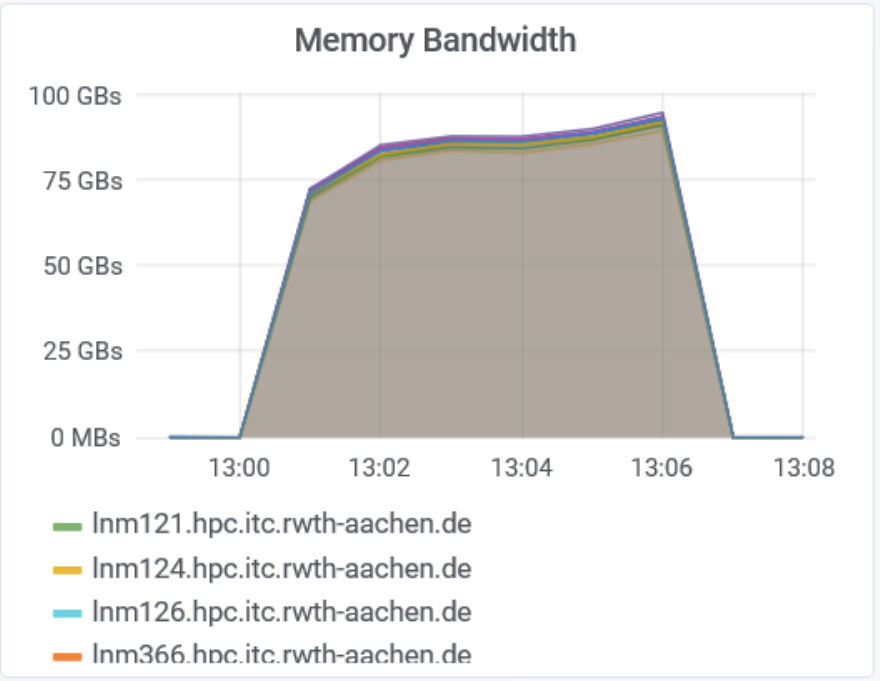
→ Candidates come from RWTH Job Mix

- Semi-automatic screening of production HPC system for more candidates

Example from our screening work

- Job-specific performance analysis framework developed within ProPE
  - University of Erlangen, TU Dresden, RWTH Aachen University





- Code: XNS ~13 GFLOP/s per node (P_peak = 844 GFLOP/s)
- 95 GB/s (B_peak = 120 GB/s)

→ Memory bus almost saturated -> might be a candidate for Aurora

Example of application work: Quantum Espresso (QE)

- What is QE?
  - A well-known code for electronic-structure calculations and materials modeling
  - One of the most resource-consuming codes at RWTH Aachen University

- Porting to Aurora
  - Replace config files using files supplied by NEC
  - Rewrite config files in order to link in NEC libraries
  - FFTW, BLAS, SCALAPACK

  → Compiler issues during beta stage, but **now builds without any bigger issues** ☺

QE Performance Measurement (Baseline)

- Reference machine: CLAIX-2016
  - 2 x Intel Xeon E5-2680 V4, 12 cores at 2.2 GHz
  - Different data sets than during procurement due to memory limitations (one VE only)
  - Data sets
    - SMALL: ~ 9 GB
    - Medium: ~ 20 GB

	CLAIX 24P [s]	Aurora 8P [s]	Speedup
Small	360	629	0.57
Medium	1297	1089	1.19

→ Speedup on SX-Aurora not impressive so far, but still nice for a baseline (no tuning work done so far done)

Make_pointlist

```
DO ir= , ,
   DO i= , ,
      DO j= , ,
         DO k= , ,

            DO ipol= , ,
            ENDDO

            DO iat= , ,
               distance = SQRT()
               IF
                  GOTO 10
               ELSE
                  GOTO 10
               ENDIF
            ENDDO

         ENDDO
      ENDDO
   ENDDO
   10 CONTINUE
ENDDO
```

execute many times,
not the performance issue her

prevents vectorization

Eliminating the gotos increases the performance significantly.

→ Good candidate for SX-Aurora TSUBASA

Functions Performance Issues

- Medium data set

	CLAIX 24P(s)	Aurora 8P(s)	Speedup
Make_pointlist	5.17	267	0.019
H_pi	457.41	304.06	1.50
Mix_rho	283.34	94.39	3

- Execution of Make_pointlist is especially inefficient
  - Executed only once for initialization
    - One possible solution: VHCall: Execute this function on the host
  - H_pi, Mix_rho might be called more often depending on the number of iterations (convergence criteria)

- Other known issues
  - MPI get_wtime() has a high execution overhead
  - sqrt() is very expensive

Summary & next steps

Conclusion

- OpenMP Offloading
  - Functional prototype complete
  - Validation with SPEC Accel benchmarks and publication of results
- Applications for Aurora
  - Promising results with QE and some other codes
  - Semi-automatic screening of candidate applications
  - Continuous extension of our performance analysis methodology
- Outreach
  - Installation of NEC SX-Aurora system at RWTH
  - Accessibility as a production system
  - *Vector Programming* workshop at RWTH in Q1/2019

Thank you for your attention.

Vielen Dank für Ihre Aufmerksamkeit.