Investigating the performance of LLVM- based Intel Fortran Complier (ifx)

Dhani Ruhela

Table of contents

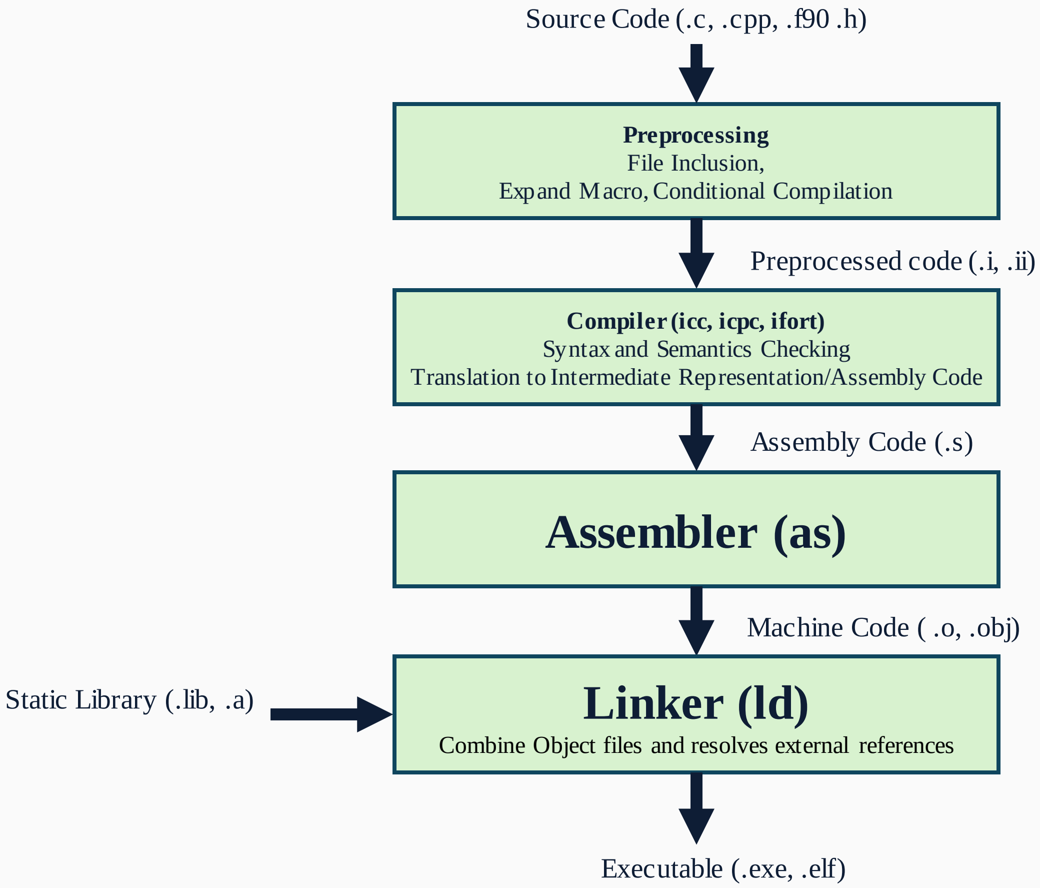
01

Compilation

Compliers

- A compiler is a software library that converts high-level language code into a machine executable format.
- Compilers are very language-specific
    - C Compilers: icc, icx, gcc, pgcc, xlc, ..
    - C++ Compilers : icpc, icpx, g++, xlc++, ..
    - Fortran Compilers: ifort, ifx, gfortran, pgfortran, XL Fortran, ..

Compilation Stages



(Compilation) $ ifort hello.f90 -o hello

Execution

(Execution) $ ./hello (Load and Execute)

Why new compilers are needed?

Existing compilers have been stagnated

How ?

- Based on decades old code generation technology
- No modern techniques like cross-file optimization and JIT codegen
- Aging code bases: difficult to learn, hard to change substantially
- Can’t be reused in other applications
- Keep getting slower with every release

Source :
LLVM.org

Low Level Virtual Machine (LLVM)

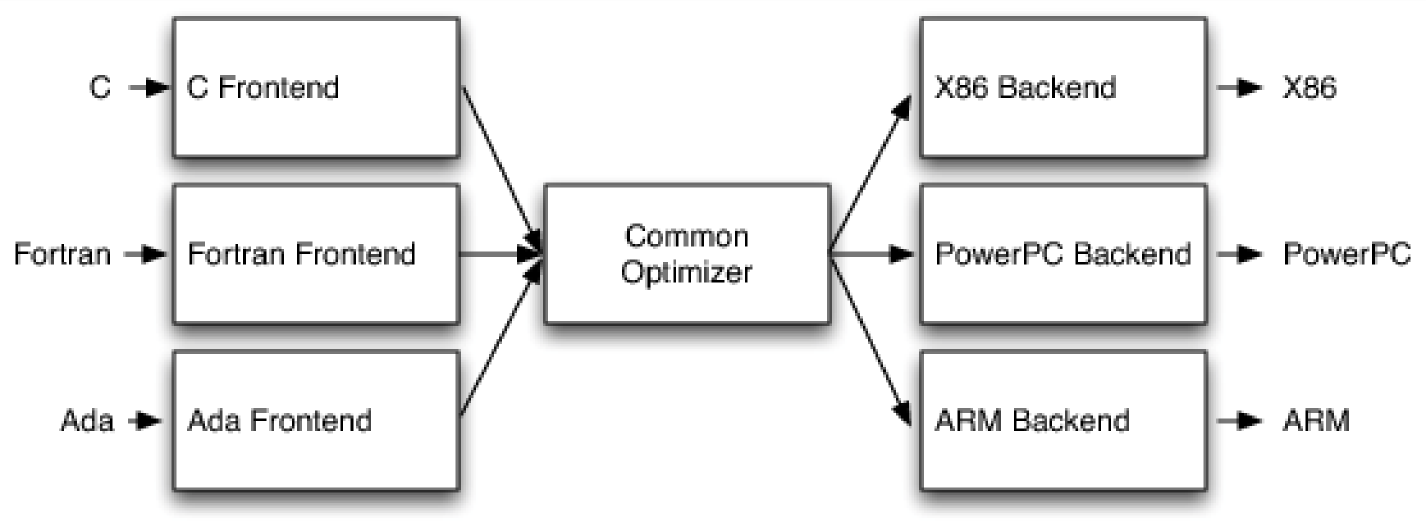
An open-source UIUC project that started in 2000 that provides a compiler frontend for many programming languages and a compiler backend for various instruction set architectures (x86, PowerPC, ARM,..)

A collection of modular compiler and toolchain technologies

Benefits

- Flexibility: Support many different languages and applications
- Consists of components shared across different compilers
- Reduces build time and cost to construct a particular compiler
- Easy maintenance
- Portability: Support many instruction set architectures
- Optimization and Performance

Low Level Virtual Machine (LLVM) - Architecture



Intel Fortran Compilers

Two separate Fortran compilers.

ifort - Named "**Intel® Fortran Compiler Classic**"

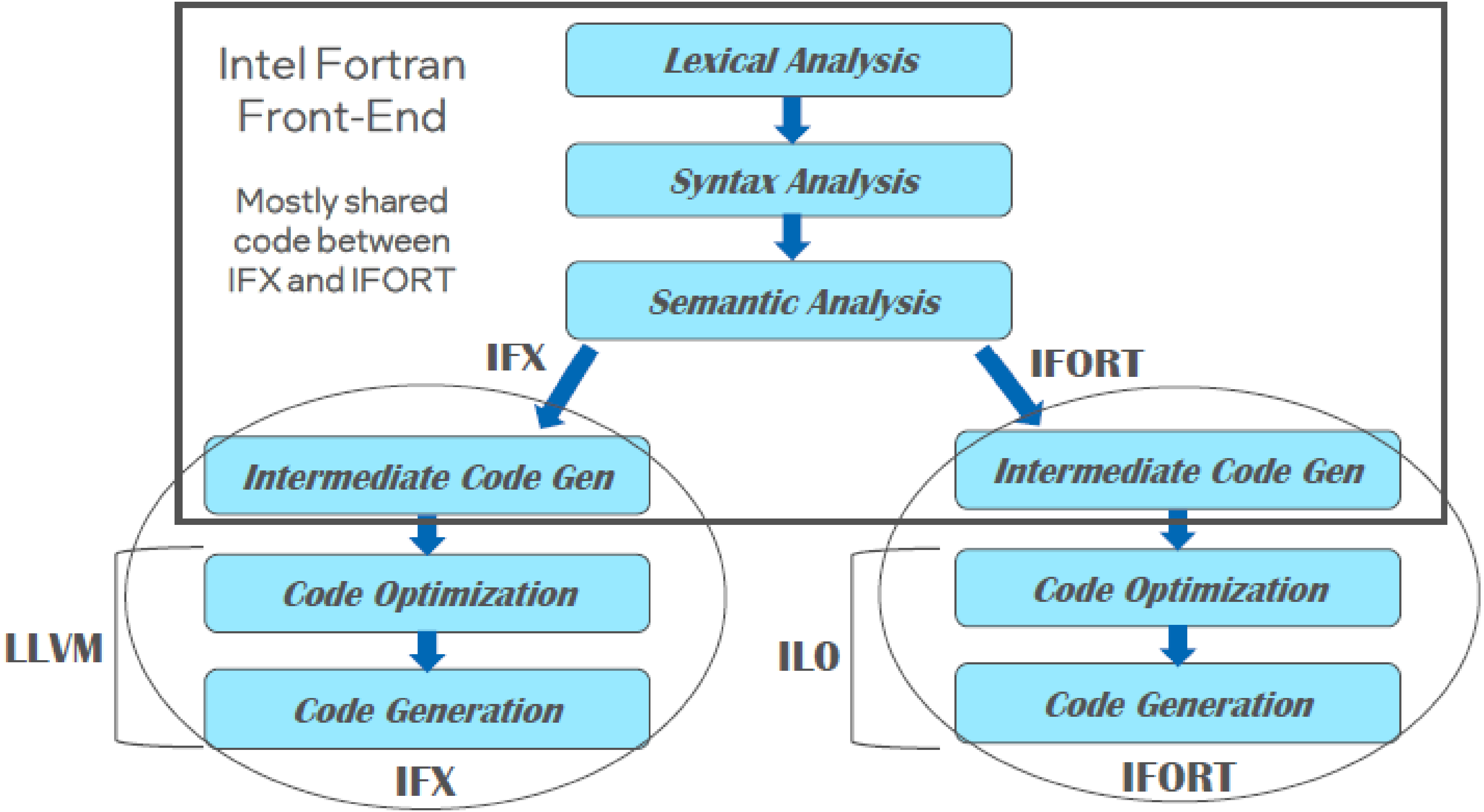
- Ifort Intel Fortran parser/analyzer + Intel optimizer/code generation (ILO)
- CPU only classic compiler. NO OFFLOAD TO GPU
- Support up to F2018 language standards

ifx – Named "**Intel® Fortran Compiler**"

- Ifort Frontend parser/analyzer + LLVM optimizer and code generation **(with Intel enhancements)**
- Supports OpenMP Offload to Intel GPUs Features!
  - *No need to call C/C++ or proprietary APIs for GPU acceleration!*
  - *An open, portable Standard to maintain your software investment*
- FORTRAN 77, Fortran 90/95, Fortran 2003, Fortran 2008 and Fortran 2018 language standards
- Binary compatible with Codes compiled with DPCPP, ICX, ICC, IFORT compilers

How IFX Relates to IFORT

Source : Intel



Research Goals

1. Performance with threads

How do the performance of binaries generated by compilers perform with increasing OMP threads?

2. Performance in benchmarks

How does binaries compiled with Legacy and new Intel Fortran compiler perform with SPEC OMP benchmark suite?

How does Intel Compilers compare with GNU Fortran Compilers

3. Performance in architectures

How does the performance of each Fortran compiler vary with Intel architecture?

04

Benchmarks

SPEC OMP 2012 Benchmarks

Benchmark	Language	Application domain
350.md	Fortran	Physics: Molecular Dynamics
351.bwaves	Fortran	Physics: Computational Fluid Dynamics (CFD)
352.nab	C	Molecular Modeling
357.bt331	Fortran	Physics: Computational Fluid Dynamics (CFD)
358.botsalgn	C	Protein Alignment
359.botsspar	C	Sparse LU
360.ilbdc	Fortran	Lattic Boltzmann
362.fma3d	Fortran	Mechanical Response Simulation
363.swim	Fortran	Weather Prediction
367.imagick	C	Image Processing
370.mgrid331	Fortran	Physics: Computational Fluid Dynamics (CFD)
371.applu331	Fortran	Physics: Computational Fluid Dynamics (CFD)
372.smithwa	C	Optimal Pattern Matching
376.kdtree	C++	Sorting and Searching

Hardware Setup

Sapphire Rapids : Intel Xeon Scalable 9480 Processor @ 1.9 Ghz (SPR-DDR5) (112 cores, 224 hw threads)
: 251 GB DDR5 Memory

Sapphire Rapids : Intel Xeon Max Processors (HBM) @ 1.9 Ghz (112 cores, 224 hw threads)
: 125 GB HBM Memory, NO DDR optional memory

Cascade Lake : Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz (56 Cores per node, 56 hw threads)
: 192 GB DDR4 Memory

Binaries generated on **Native Machines** with Flags (Refer paper for details):

IFORT: -O3 -qopenmp -ipo1 -xCORE-AVX512 -qopt-zmm-usage=high -shared-intel

IFX : -O3 -qopenmp -ipo1 -xCORE-AVX512 -mprefer-vector-width=512 -shared-intel -axSAPPHIRERAPIDS

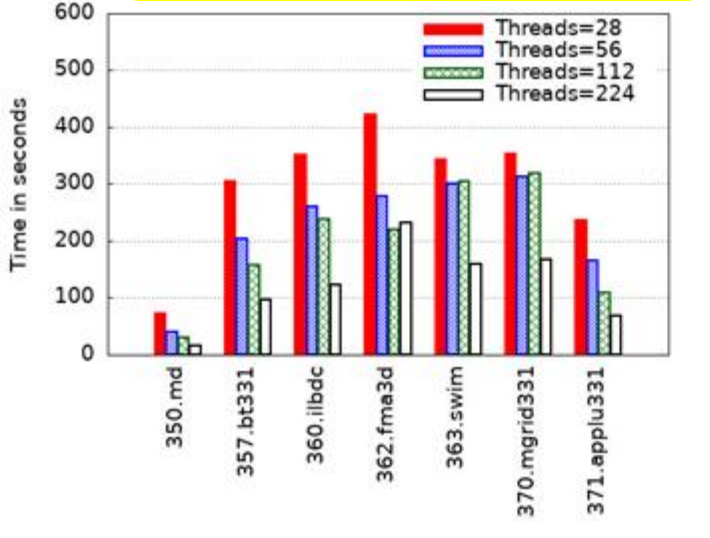
Gfortran : -Ofast -fopenmp -funroll-loops -march=native -mtune=native

05

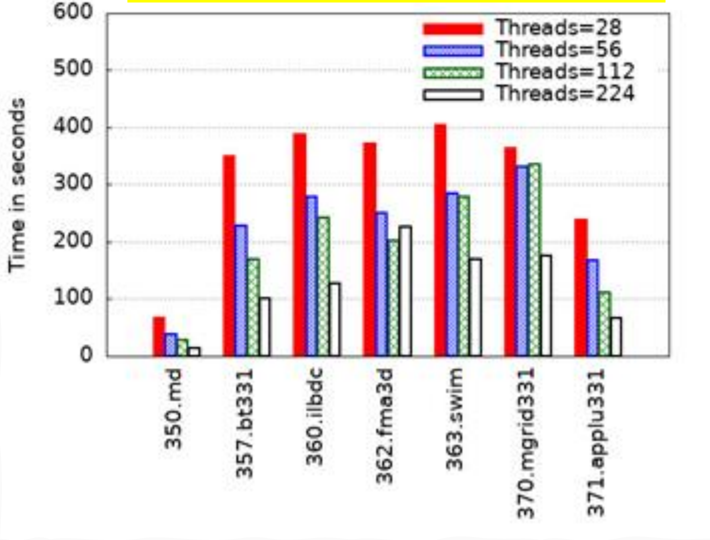
Experiments

Scalability with OMP Threads

SPR + DDR5 Architecture



SPR + HBM Architecture



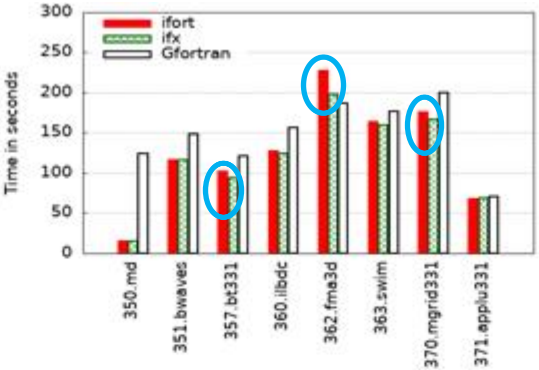
Conclusions :

Most of the benchmarks scale well in performance with increasing OMP threads.
Exceptions :

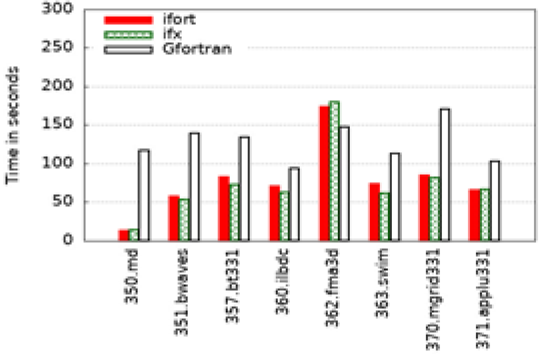
- 362.fma3d shows overheads at 224 OMP threads
- 363.swim and 370.MGRID331 shows no change in performance at 112 threads.

Comparing ifort, ifx, Gfortran

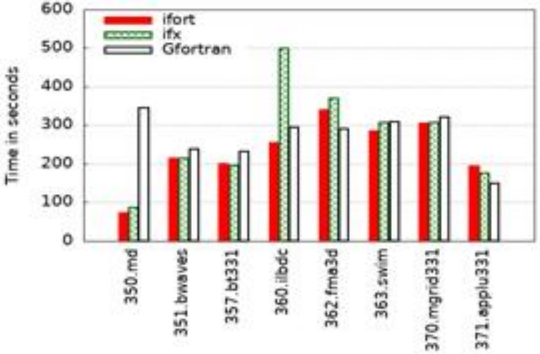
SPR + DDR5 (224 threads)



SPR + HBM (224 threads)



Cascade Lake (56 threads)

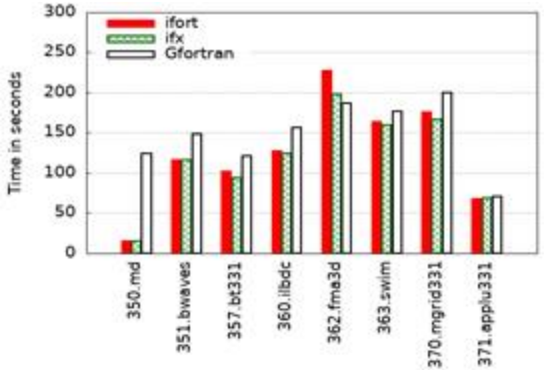


Conclusions :

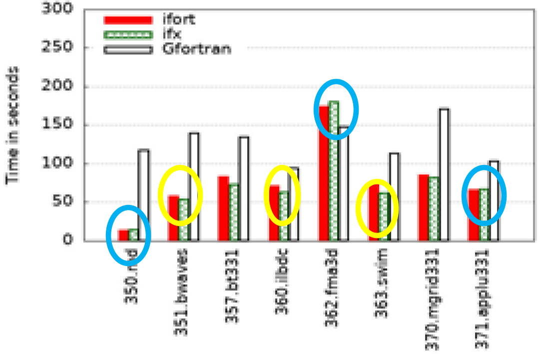
- SPR+DDR5: Ifx perform 13% better than Ifort (bt.331, fma.3d, mggrid.331).
- Gfortran 30% to 7.3 times slower.

Comparing ifort, ifx, Gfortran

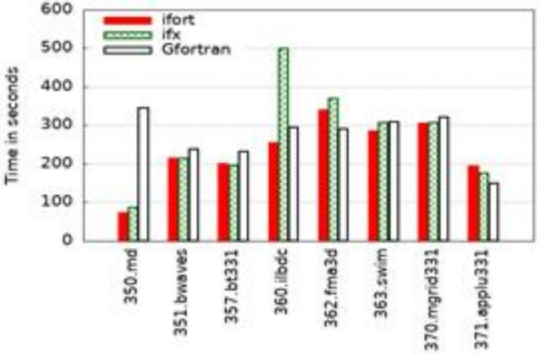
SPR + DDR5 (224 threads)



SPR + HBM (224 threads)



Cascade Lake (56 threads)



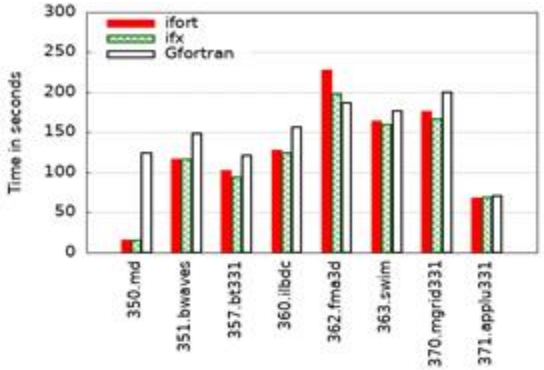
Conclusions :

- SPR+DDR5: Ifx perform 13% better than Ifort (bt.331, fma.3d, mggrid.331).
- Gfortran 30% to 7.3 times slower.

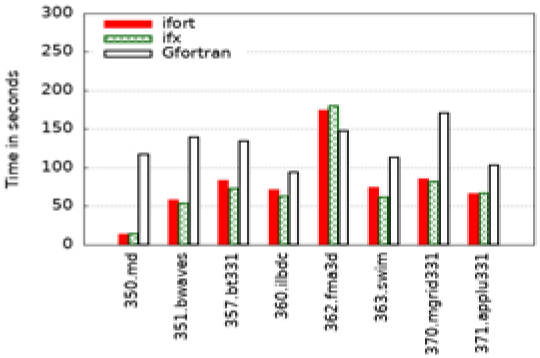
- SPR+HBM: Ifx perform 17% better than Ifort (bwaves, ibdc, swim) and 3% slower for (md, fma3d, applu331) due to slightly higher memory latency of SPR-HBM (130 ns) node vs SPR-DDR5 (110 ns) node.
- Gfortran 1.5 times to 7.6 times slower.

Comparing ifort, ifx, Gfortran

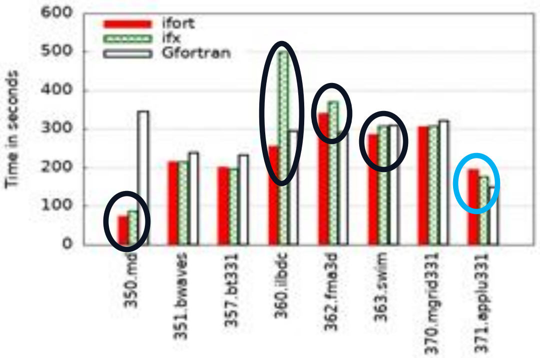
SPR + DDR5 (224 threads)



SPR + HBM (224 threads)



Cascade Lake (56 threads)



Conclusions :

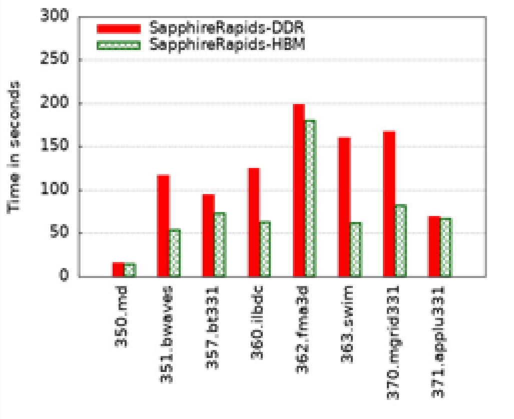
- SPR+DDR5: Ifx perform 13% better than Ifort (fma.3d, bt.331, mggrid.331).
- Gfortran 30% to 7.3 times slower.

- SPR+HBM: Ifx perform 17% better than Ifort (swim, ibdc, waves) and 3% slower for md, fma3d, applu331due to higher memory latency of SPR-HBM (130 ns) node vs SPR-DDR5 (110 ns) node.
- Gfortran 1.5 times to 7.6 times slower.

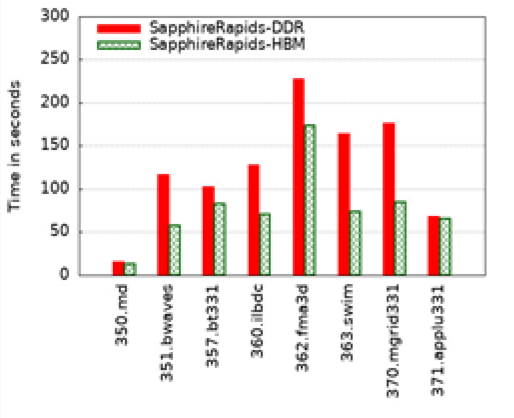
- Cascade Lake: Ifx significantly slower than Ifort (-95%: ibdc -21%:md -9%:fma3d -8%:swim except +9%applu331.
- Gfortran perform much better on Cascade Lake in comparison to Saphire Rapids (Slowness:3.8%:md, <17%:bt,ibdc,bwaves,swim,mgrid, Fastness:23%applu331,23%fma3d)

Finding benefits of HBM Memory

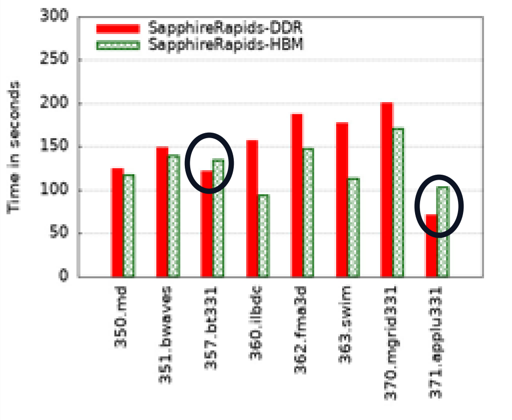
Ifort



Ifx



Gfortran



Conclusions :

- General Trend : HBM memory improves performance for majority of applications
- Ifx/Ifort : Most benchmarks run up to 3%-60% faster at nodes with HBM memory compared to nodes with DDR5 memory as the memory bandwidth for nodes with HBM memory is 3.5 times higher for SPR-H nodes.
- Binaries compiled with Gfortran, bt331 and applu331 shows contrary trend.

06

Conclusions

Conclusions

- Experimented the performance of legacy and LLVM-based Fortran compiler on three Intel architectures and compare it with popular GNU Fortran compiler (gfortran).
- Presented up to 17% improvement by Ifx in running time of SPEC OpenMP 2012 Fortran applications.
- Presented up to 60% improvement in running time of the benchmarks on Sapphire Rapids with HBM memory compared to DDR5 memory.
- Indicated the need of tuning LLVM based Ifx compiler on previous Intel architecture.

Thank you for your time!