Clacc 2019: An Update on OpenACC Support for Clang and LLVM

Joel E. Denny, Seyong Lee, Jeffrey S. Vetter

Future Technologies Group, ORNL

https://ft.ornl.gov/   dennyje@ornl.gov

April 8, 2019 EuroLLVM

U.S. DEPARTMENT OF **ENERGY**

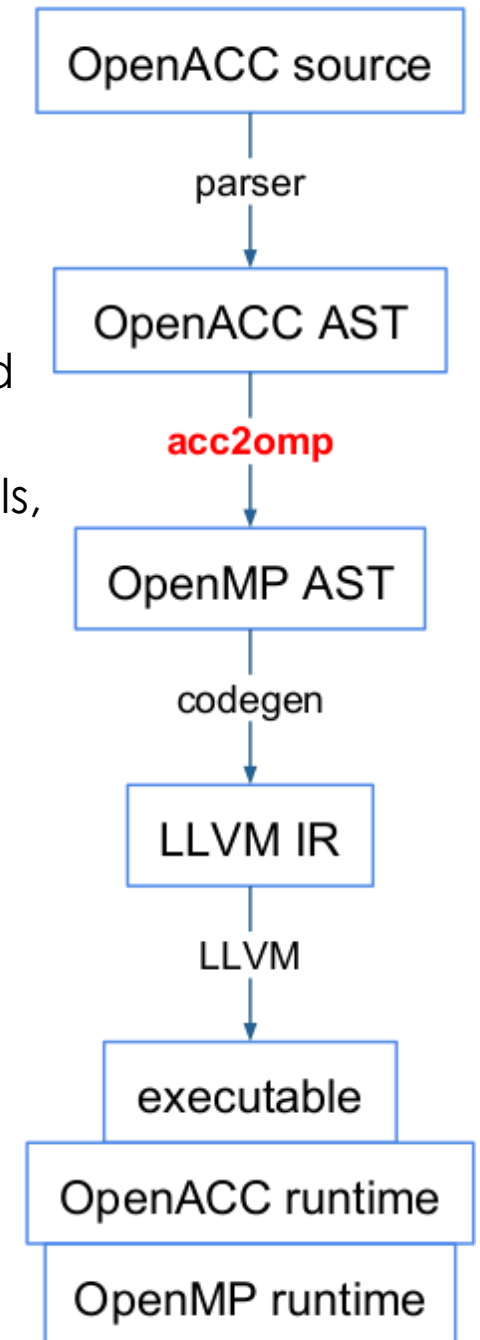# Clacc Overview

Clacc Background

OpenACC

- Launched 2010 as portable directive-based programming model in C, C++, Fortran for heterogeneous accelerators

- Best known for NVIDIA GPU; implementations have targeted AMD GCN, multicore CPU, Intel Xeon Phi, FPGA

- Compared to OpenMP
  - Descriptive vs. Prescriptive
  - Many features ported to OpenMP
  - Specification less complex

- OpenACC 2.7 released in Nov, 2018

Clacc

- US Exascale Computing Project (ECP)

- Goal: Open-source, production-quality, standard-conforming OpenACC compiler support for Clang and LLVM

- Why?
  - Needed for HPC app development and OpenACC adoption and evolution
  - GCC is only open-source, production-quality compiler supporting OpenACC

- Design: Translate OpenACC to OpenMP to build on OpenMP support in Clang

OAK RIDGE
National Laboratory

Clacc Current Design

- Need AST transformation
  - OpenACC AST for source-level tools: pretty printers, analyzers, lint tools, and debugger and editor extensions, etc.
  - OpenMP AST for source-to-source: reuse OpenMP implementation and tools, automatically port apps, etc.

- Problem
  - Clang AST is immutable by design

- Solution
  - Add hidden OpenMP subtree for each OpenACC subtree
  - Using Clang's TreeTransform facility
  - TreeTransform nicely encapsulates reuse of Sema implementations
  - TreeTransform uses CRTP to be extensible

OpenACC source

| parser

OpenACC AST

| acc2omp

OpenMP AST

| codegen

LLVM IR

| LLVM

executable

OpenACC runtime

OpenMP runtime

OAK RIDGE
National Laboratory

Clacc Roadmap

2019 and earlier

- Focus on C

- Focus on behavioral correctness
  - Prescriptive OpenACC interpretation
  - Many-to-one mapping to OpenMP

- Propose fixes to OpenACC spec

- Upstreaming mutually beneficial improvements to Clang and LLVM

2020 and later

- Extend to C++

- Focus on performance
  - Descriptive OpenACC interpretation
  - Analyses for best mapping to OpenMP
  - Investigate advanced LLVM analyses (e.g., autotuning, polly, LLVM IR extensions)

- Upstreaming OpenACC support

OAK RIDGE
National Laboratory
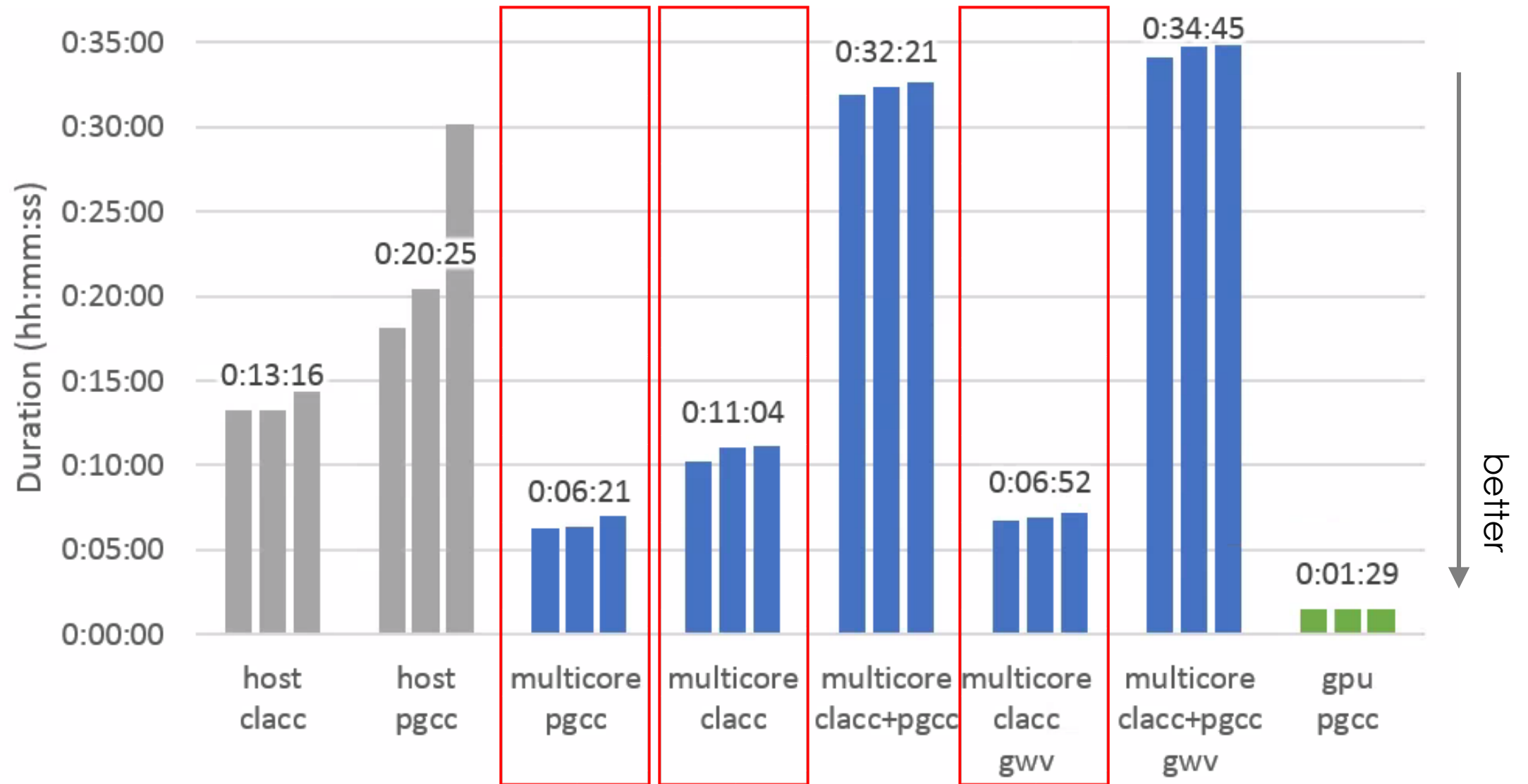
Early Performance Results

Clacc: Translating OpenACC to OpenMP in Clang, Joel E. Denny, Seyong Lee, and Jeffrey S. Vetter, 2018 IEEE/ACM 5th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC), Dallas, TX, USA, (2018).
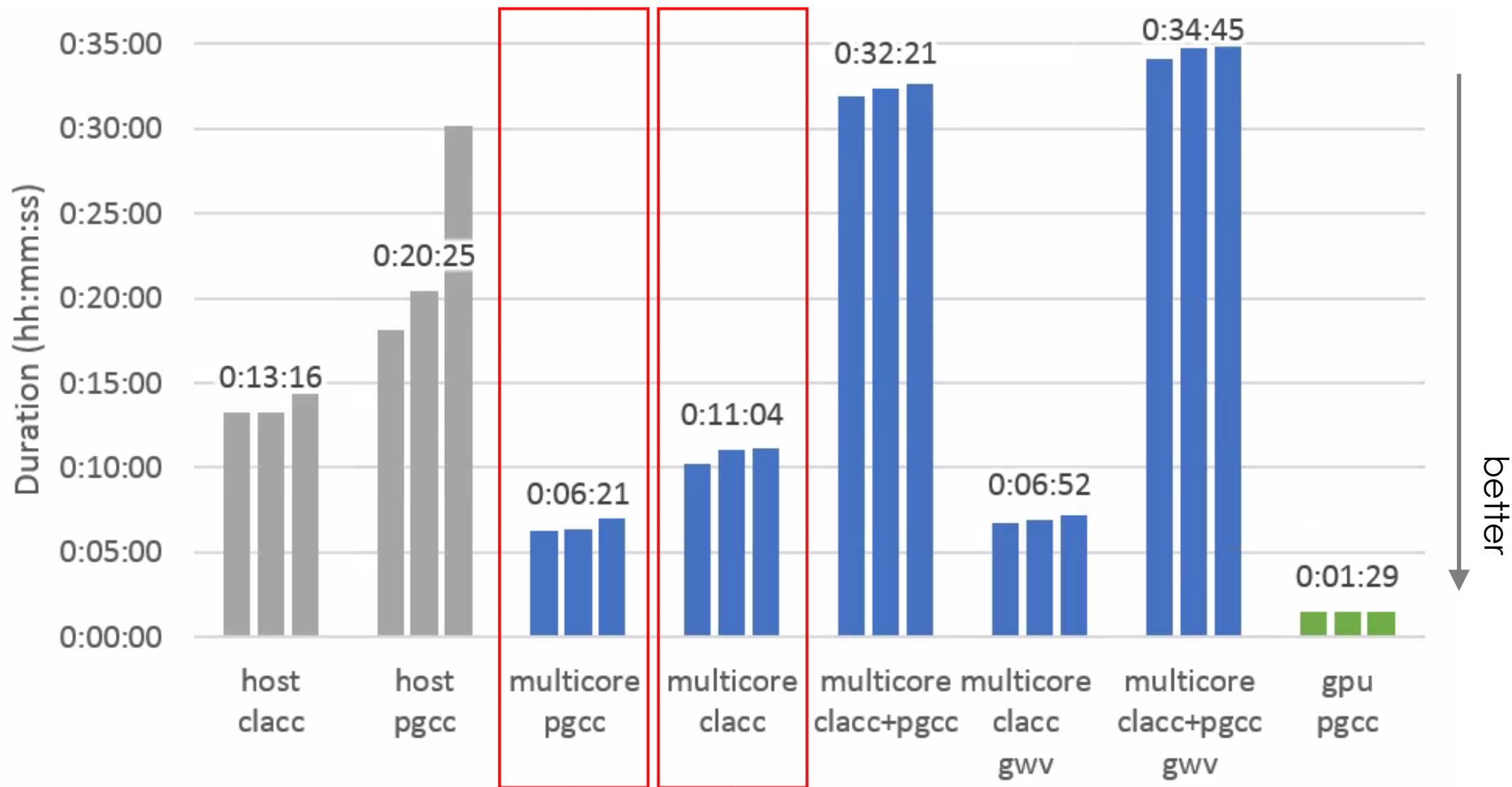
SPEC ACCEL 303.ostencil

SPEC ACCEL 303.ostencil

Manually added *prescriptive* gang/worker/vector
(like OpenMP distribute/parallel for/simd)
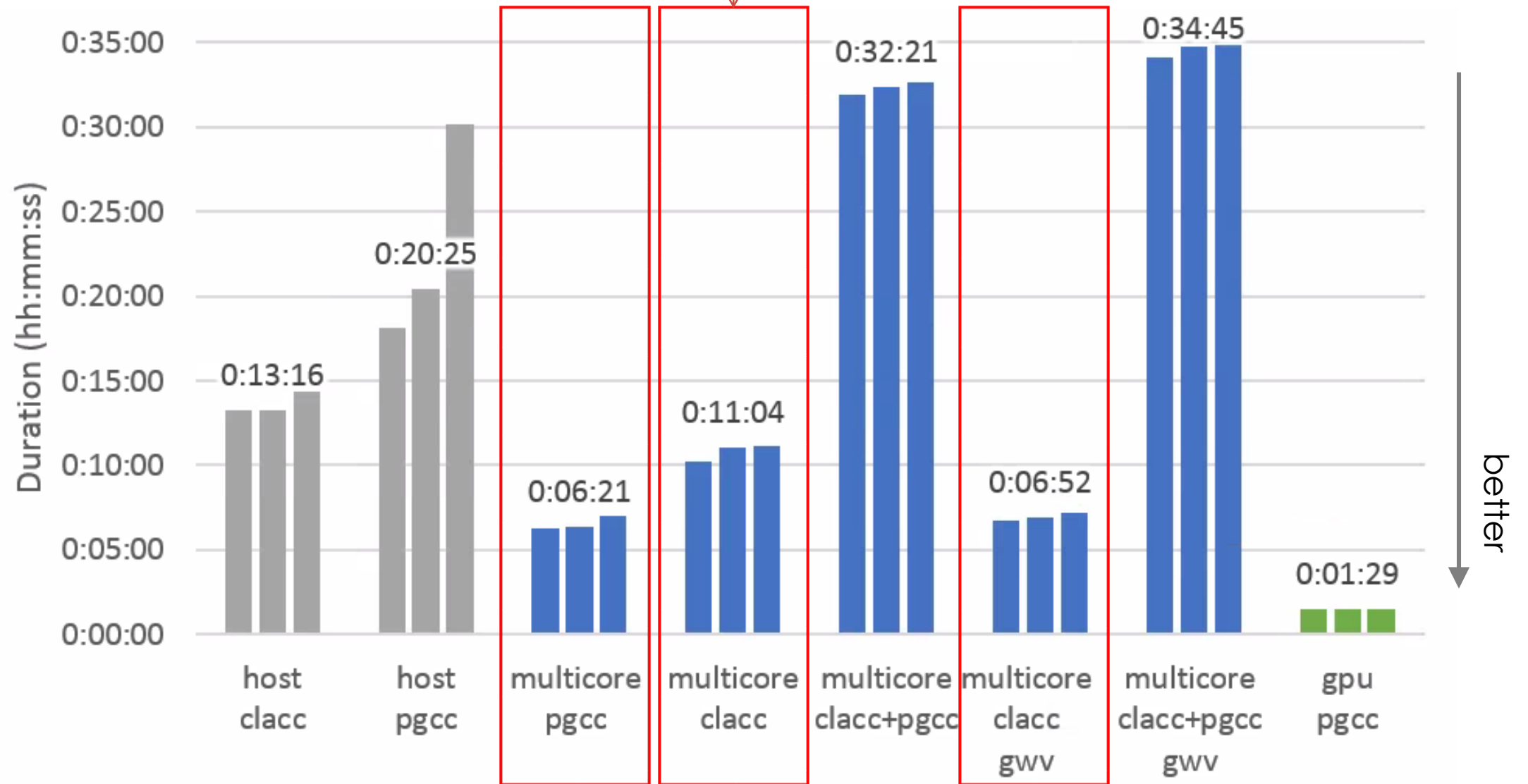
SPEC ACCEL 303.ostencil

Needs *descriptive* interpretation

Manually added *prescriptive* gang/worker/vector
(like OpenMP distribute/parallel for/simd)

OAK RIDGE
National Laboratory

Upstream Contributions

- Mutually beneficial contributions

- Not OpenACC-specific yet

- Big thanks to reviewers and others in the LLVM community!

Clang and OpenMP Improvements

- OpenMP Parse and Sema fixes

- Clang -ast-print fixes
  - Affects Clacc in source-to-source mode

- Attribute handling fixes

- Debian/Ubuntu nvidia-cuda-toolkit support fixes
  - Affects OpenMP/OpenACC offloading support

- Add libraries to Clang-dedicated directories
  - Avoids incorrect linking of libomp*.so
  - In progress

OAK RIDGE
National Laboratory

Testing Infrastructure Improvements

- clang -cc1 -verify=<prefixes>
  - like FileCheck -check-prefixes=<prefixes>
  - // expected-error {{message}}
  - // your-prefix-error {{message}}

- lit -vv shows line number for failed RUN command
  - Some lit tests have hundreds of RUN commands

- FileCheck CHECK-DAG behavior cleanup
  - Most notably, matches are now non-overlapping
  - More intuitive and less error-prone
  - Enables checking unordered, non-unique strings (e.g., from parallel program)

OAK RIDGE
National Laboratory

FileCheck Debugging

- FileCheck -v and -vv
  - Traces matches

- FileCheck -color
  - Forces color output through lit/ninja

- FileCheck -dump-input=always|fail|never
  - Dumps input annotated with diagnostics

- FILECHECK_OPTS environment var
  - Passes command-line options through lit/ninja
  - FILECHECK_OPTS='-vv –color –dump-input=fail' ninja check-clang-openmp

OAK RIDGE
National Laboratory

FileCheck normal diagnostic

Input

```
1: store i64 %0, i64* %1
2: store i64 %2, i64* %3
3: store i64 %4, i64* %5
4: store i64 %6, i64* %7
5: store i32 %8, i32* %9
6: store i64 %10, i64* %11
7: ret i32 %8
```

Checks

```
1: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
2: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
3: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
4: CHECK: store i32 %{{[0-9]+}}, i32* %{{[0-9]+}}
5: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
6: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
```

Error at CHECK on line 6?

```
check:6:8: error: CHECK: expected string not found in input
CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
       ^
<stdin>:7:1: note: scanning from here
ret i32 %8
^
```

OAK RIDGE
National Laboratory

FileCheck -v -dump-input=fail

Input

```
1: store i64 %0, i64* %1
2: store i64 %2, i64* %3
3: store i64 %4, i64* %5
4: store i64 %6, i64* %7
5: store i32 %8, i32* %9
6: store i64 %10, i64* %11
7: ret i32 %8
```

Checks

```
1: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
2: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
3: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
4: CHECK: store i32 %{{[0-9]+}}, i32* %{{[0-9]+}}
5: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
6: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
```

```
Full input was:
<<<<<<
          1: store i64 %0, i64* %1
check:1      ^~~~~~~~~~~~~~~~~~~~~~
          2: store i64 %2, i64* %3
check:2      ^~~~~~~~~~~~~~~~~~~~~~
          3: store i64 %4, i64* %5
check:3      ^~~~~~~~~~~~~~~~~~~~~~
          4: store i64 %6, i64* %7
          5: store i32 %8, i32* %9
check:4      ^~~~~~~~~~~~~~~~~~~~~~
          6: store i64 %10, i64* %11
check:5      ^~~~~~~~~~~~~~~~~~~~~~~~
          7: ret i32 %8
check:6      X~~~~~~~~~~ error: no match found
>>>>>>
```

OAK RIDGE
National Laboratory

FileCheck -v -dump-input=fail

Input

```
1: store i64 %0, i64* %1
2: store i64 %2, i64* %3
3: store i64 %4, i64* %5
4: store i64 %6, i64* %7
5: store i32 %8, i32* %9
6: store i64 %10, i64* %11
7: ret i32 %8
```

Checks

```
1: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
2: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
3: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
4: CHECK: store i32 %{{[0-9]+}}, i32* %{{[0-9]+}}
5: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
6: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
```

**Ah! Problem actually occurred earlier:
Line 4 never matched!**

```
Full input was:
<<<<<<
            1: store i64 %0, i64* %1
check:1        ^~~~~~~~~~~~~~~~~~~~~~~
            2: store i64 %2, i64* %3
check:2        ^~~~~~~~~~~~~~~~~~~~~~~
            3: store i64 %4, i64* %5
check:3        ^~~~~~~~~~~~~~~~~~~~~~~
            4: store i64 %6, i64* %7
            5: store i32 %8, i32* %9
check:4        ^~~~~~~~~~~~~~~~~~~~~~~
            6: store i64 %10, i64* %11
check:5        ^~~~~~~~~~~~~~~~~~~~~~~~
            7: ret i32 %8
check:6        X~~~~~~~~~~ error: no match found
>>>>>>
```

🌿 **OAK RIDGE**
National Laboratory

FileCheck -v -dump-input=fail

Input

```
1: store i64 %0, i64* %1
2: store i64 %2, i64* %3
3: store i64 %4, i64* %5
4: store i64 %6, i64* %7
5: store i32 %8, i32* %9
6: store i64 %10, i64* %11
7: ret i32 %8
```

Checks

```
1: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
2: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
3: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
4: CHECK: store i32 %{{[0-9]+}}, i32* %{{[0-9]+}}
5: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
6: CHECK: store i64 %{{[0-9]+}}, i64* %{{[0-9]+}}
```

reversed directives

```
Full input was:
<<<<<<
            1: store i64 %0, i64* %1
check:1     ^~~~~~~~~~~~~~~~~~~~~~~~
            2: store i64 %2, i64* %3
check:2     ^~~~~~~~~~~~~~~~~~~~~~~~
            3: store i64 %4, i64* %5
check:3     ^~~~~~~~~~~~~~~~~~~~~~~~
            4: store i64 %6, i64* %7
            5: store i32 %8, i32* %9
check:4     ^~~~~~~~~~~~~~~~~~~~~~~~
            6: store i64 %10, i64* %11
check:5     ^~~~~~~~~~~~~~~~~~~~~~~~~~
            7: ret i32 %8
check:6     X~~~~~~~~~~ error: no match found
>>>>>>
```

**Ah! Problem actually occurred earlier:
Line 4 never matched!**

OAK RIDGE
National Laboratory

Clacc Takeaways

- Overview
  - Objective: Production-quality OpenACC compiler support for Clang and LLVM
  - Design: Translate OpenACC to OpenMP to build on existing OpenMP support in Clang

- Roadmap
  - <= 2019: C, correctness, upstream mutually beneficial improvements
  - >= 2020: C++, performance, upstream OpenACC support

- Join Us
  - Future Technologies Group, Oak Ridge National Laboratory
  - Hiring interns, postdocs, research and technical staff
  - External collaborators welcome

      https://ft.ornl.gov/
      dennyje@ornl.gov

OAK RIDGE
National Laboratory