HYBRID FORTRAN

High Performance, Low Friction GPGPU for Weather Prediction

Michel Müller
RIKEN Advanced Institute for Computational Science
michel@typhooncomputing.com

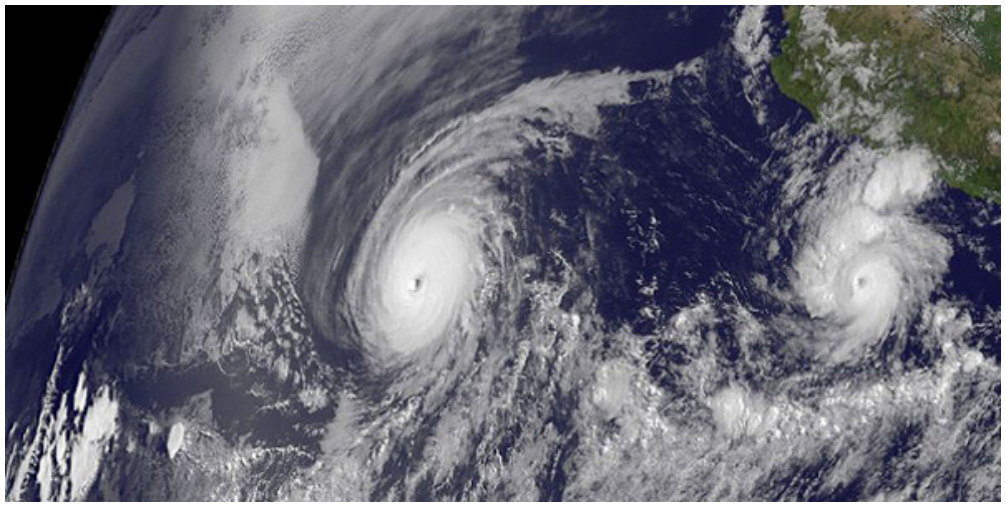
Supervised by
Dr. Takashi Shimokawabe
Tokyo Institute of Technology

Prof. Dr. Takayuki Aoki
Tokyo Institute of Technology

Prof. Dr. Roger Wattenhofer
ETH Zurich

Dr. Tabito Hara
Japan Meteorological Agency

Dr. Naoya Maruyama
RIKEN Advanced Institute for Computational Science



東京工業大学
Tokyo Institute of Technology

GSIC
Global Scientific Information and Computing Center

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Japan Meteorological Agency

RIKEN

HYBRID FORTRAN

High Performance, Low Friction GPGPU for Weather Prediction

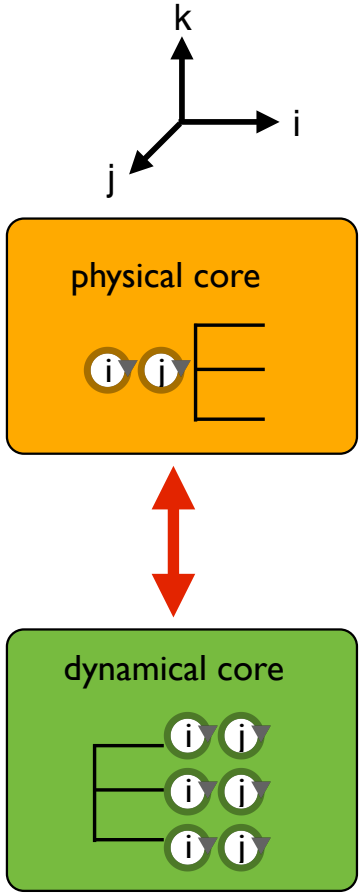
CPU + GPU executable

Topics

1. Motivation
2. Why not OpenACC?
3. Introducing Hybrid Fortran
4. Implementation
5. Results & Discussion
6. Conclusion & Future Work

ASUCA | Motivation

- What is ASUCA? [4]
  - Nonhydrostatic weather prediction model
  - Successor of current JMANHM model
  - Dynamical + physical core
- Goals for ASUCA GPGPU portation of physical core:
  1. Eliminate host-to-device communication
  2. Gain execution time speedups



[1] K. Kawano, J. Ishida, and C. Muroi, “Development of a New Nonhydrostatic Model ASUCA at JMA”, 2010.

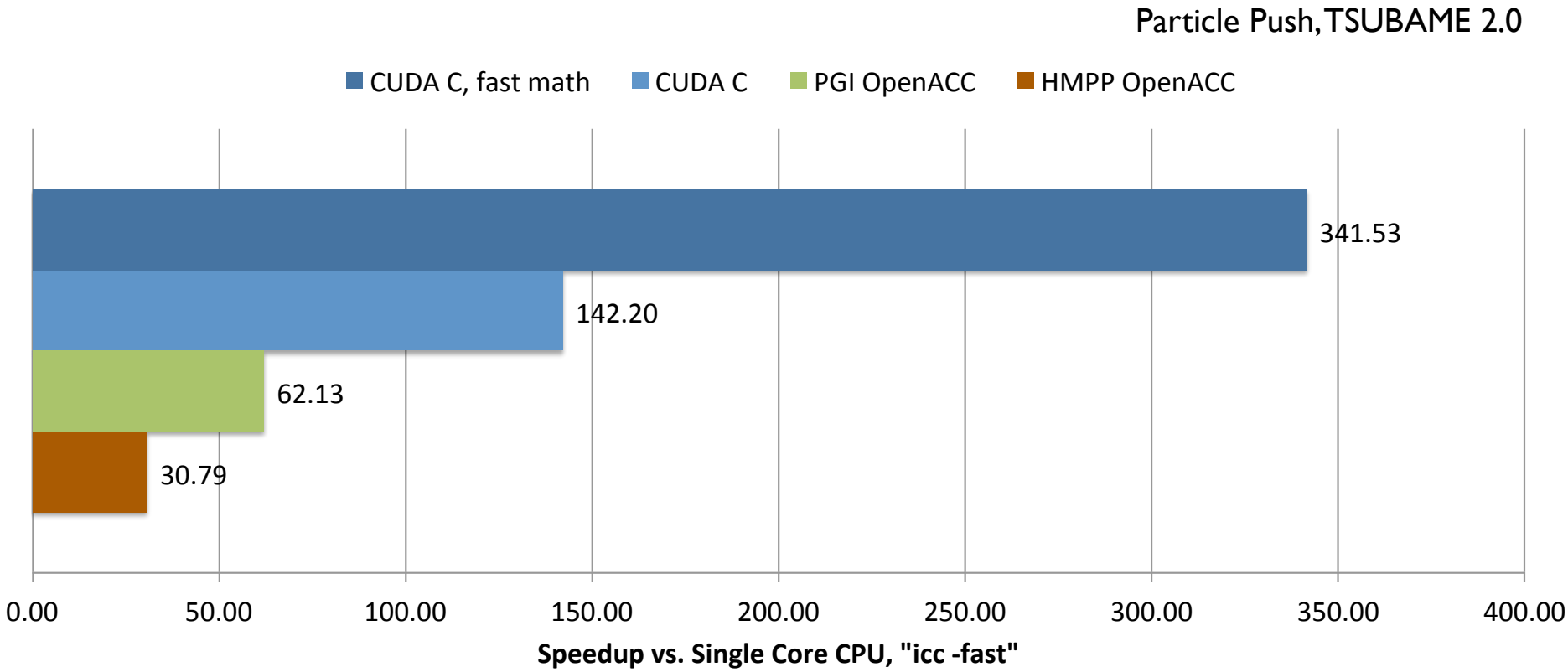
ASUCA: Challenges

- Challenges:
  1. Keep familiar development environment (Fortran 90)
  2. Amount of code changes as small as possible
  3. Keep CPU compatibility
  4. Keep CPU performance and enable high GPU performance
- Scope of this Project:
  - Concentrate on single GPU / single CPU socket

Why not OpenACC?

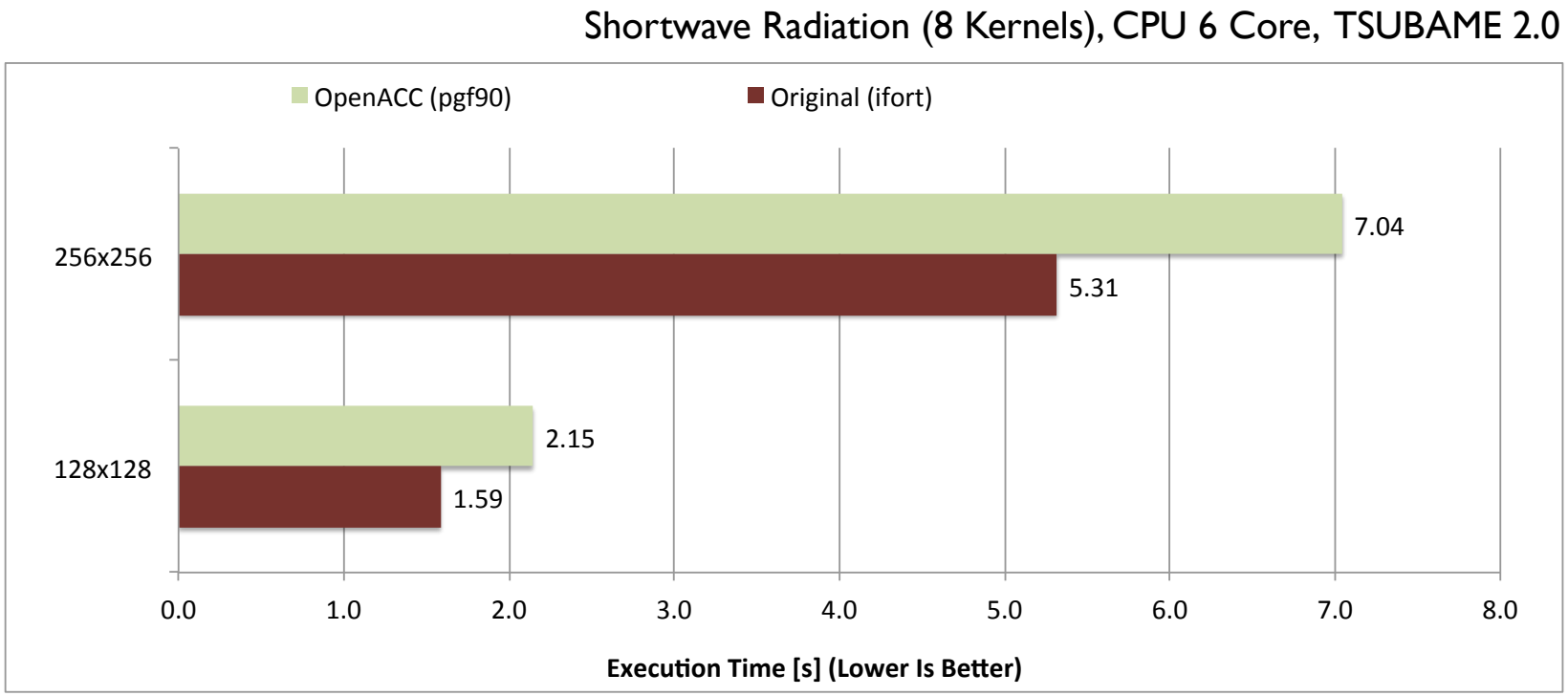
Why not OpenACC?

1. GPU performance for computationally bounded problems



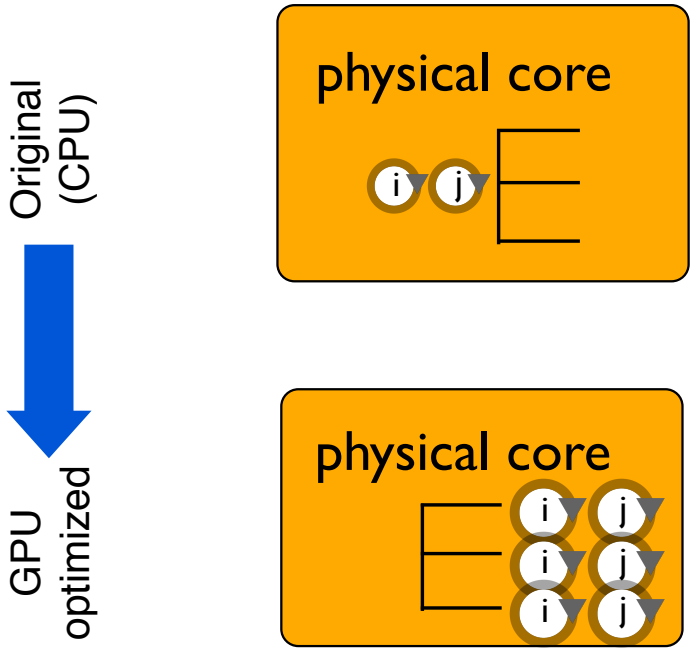
Why not OpenACC?

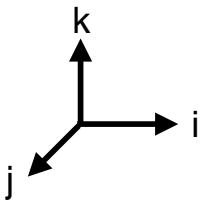
1. GPU performance for computationally bounded problems
2. CPU performance



Why not OpenACC?

1. GPU performance for computationally bounded problems
2. CPU performance
3. Insufficient abstraction

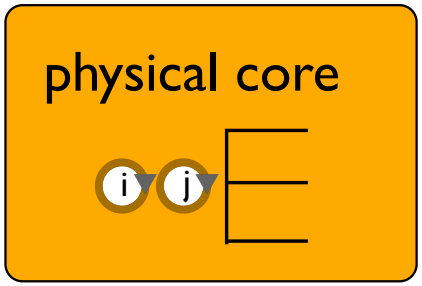


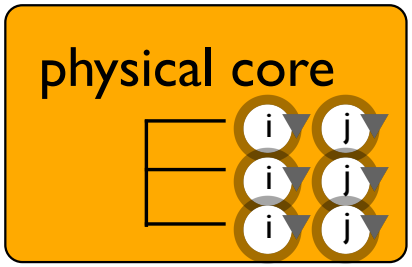


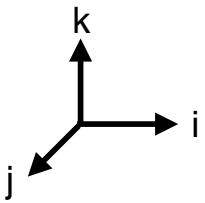
Hybrid Fortran approach

Original (CPU)

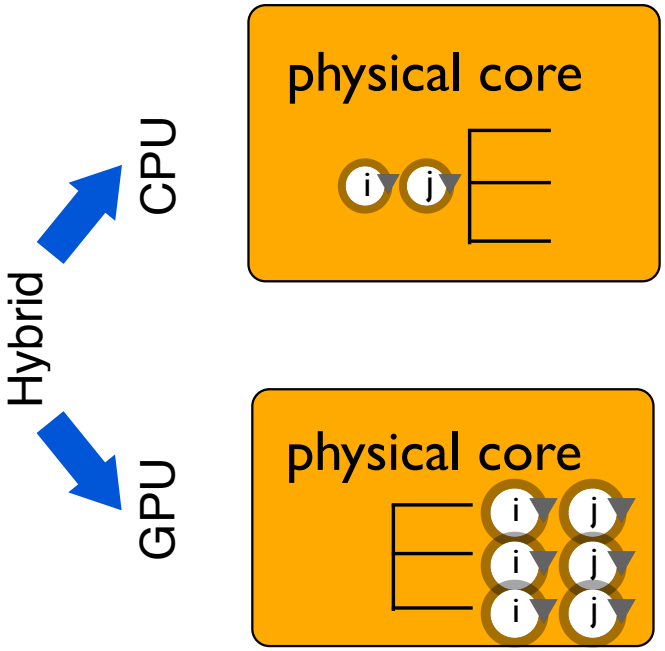
GPU optimized







Hybrid Fortran approach



Example | Hybrid Fortran

```
1  module example
2  contains
3    subroutine wrapper(a, b, c, d)
4      real, intent(in) :: a(DOM(NX, NY, NZ)), b(DOM(NX, NY, NZ))
5      real, intent(out) :: c(DOM(NX, NY, NZ)), d(DOM(NX, NY, NZ))
6      integer(4) :: x, y
7
8      do y=1,NY
9        do x=1,NX
10         call add(a(AT(x,y,:)), b(AT(x,y,:)), c(AT(x,y,:)))
11         call mult(a(AT(x,y,:)), b(AT(x,y,:)), d(AT(x,y,:)))
12       end do
13     end do
14   end subroutine
16   subroutine add(a, b, c)
17     real, intent(in) :: a(NZ), b(NZ)
18     real, intent(out) :: c(NZ)
19     integer :: z
20
21     do z=1,NZ
22       c(z) = a(z) + b(z)
23     end do
34   end subroutine
```

Example | Hybrid Fortran

```
1  module example
2  contains
3    subroutine wrapper(a, b, c, d)
4      real, intent(in) :: a(DOM(NX, NY, NZ)), b(DOM(NX, NY, NZ))
5      real, intent(out) :: c(DOM(NX, NY, NZ)), d(DOM(NX, NY, NZ))
6      integer(4) :: x, y
7
8      do y=1,NY
9      @parallelRegion{appliesTo(CPU), domName(x,y), domSize(NX, NY)}
10         call add(a(AT(x,y,:)), b(AT(x,y,:)), c(AT(x,y,:)))
11         call mult(a(AT(x,y,:)), b(AT(x,y,:)), d(AT(x,y,:)))
12     @end parallelRegion
13     end do
14   end subroutine
16   subroutine add(a, b, c)
17     real, intent(in) :: a(NZ), b(NZ)
18     real, intent(out) :: c(NZ)
19     integer :: z
20
21     do z=1,NZ
22       c(z) = a(z) + b(z)
23     end do
34   end subroutine
```

Example | Hybrid Fortran

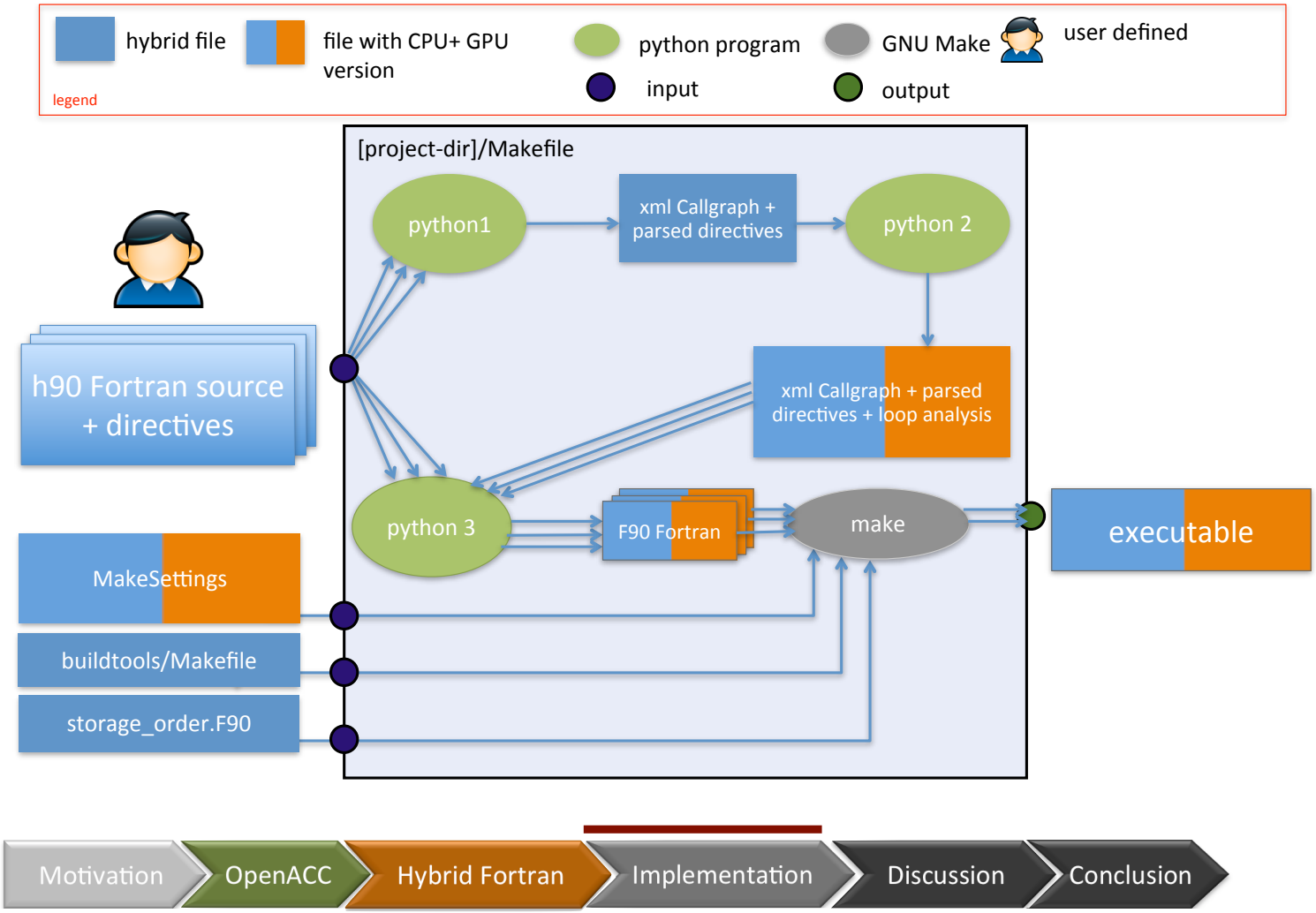
```
1  module example
2  contains
3    subroutine wrapper(a, b, c, d)
4      real, intent(in) :: a(DOM(NX, NY, NZ)), b(DOM(NX, NY, NZ))
5      real, intent(out) :: c(DOM(NX, NY, NZ)), d(DOM(NX, NY, NZ))
6      integer(4) :: x, y
7
8      do y=1,NY
9      @parallelRegion{appliesTo(CPU), domName(x,y), domSize(NX, NY)}
10         call add(a(AT(x,y,:)), b(AT(x,y,:)), c(AT(x,y,:)))
11         call mult(a(AT(x,y,:)), b(AT(x,y,:)), d(AT(x,y,:)))
12     @end parallelRegion
13     end do
14   end subroutine

16   subroutine add(a, b, c)
17     real, intent(in) :: a(NZ), b(NZ)
18     real, intent(out) :: c(NZ)
19     integer :: z
20     @parallelRegion{appliesTo(GPU), domName(x,y), domSize(NX, NY)}
21     do z=1,NZ
22       c(z) = a(z) + b(z)
23     end do
       @end parallelRegion
34   end subroutine
```

Example | Hybrid Fortran

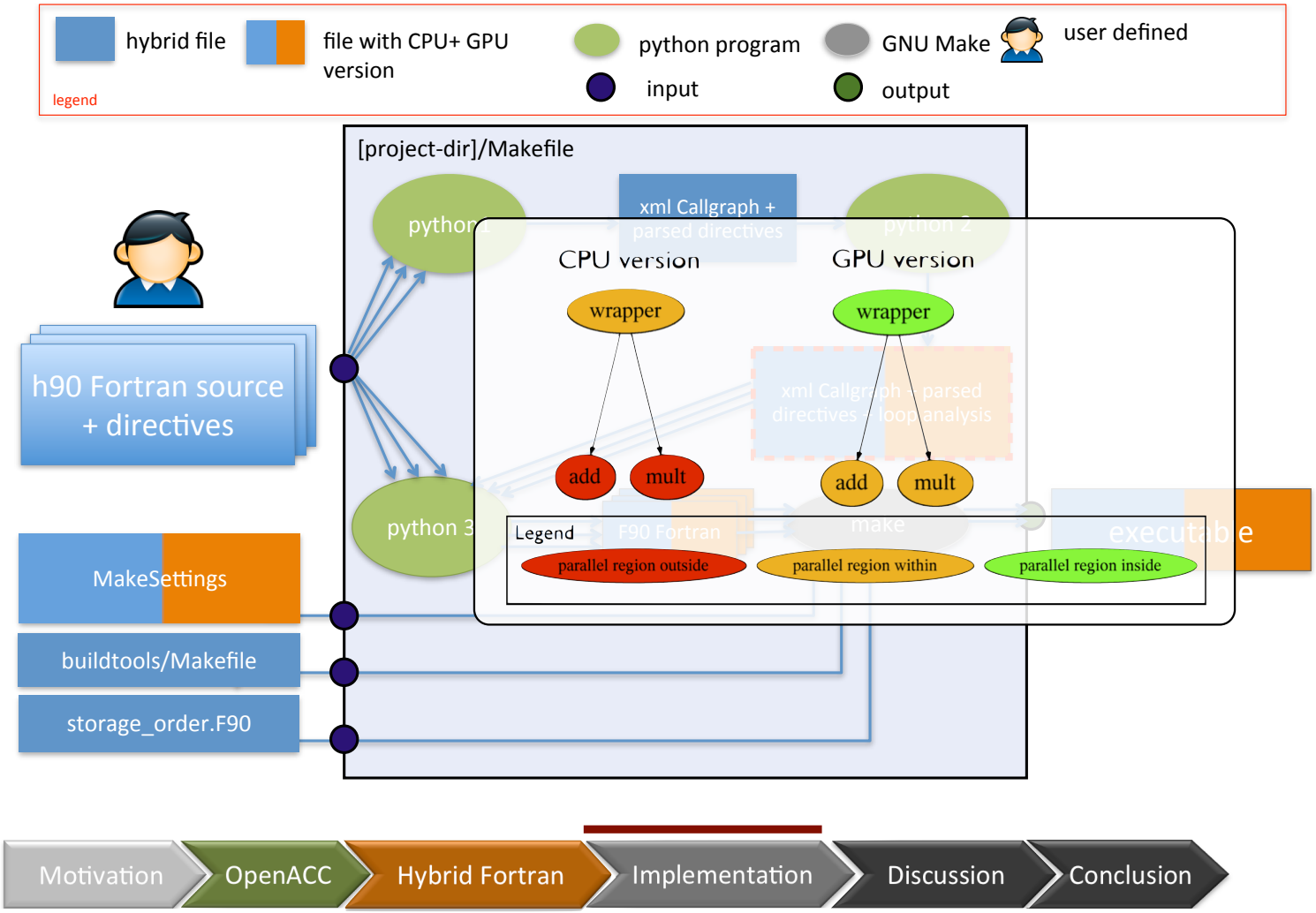
```
1  module example
2  contains
3    subroutine wrapper(a, b, c, d)
4      real, dimension(NZ), intent(in) :: a, b
5      real, dimension(NZ), intent(out) :: c, d
6
7      @domainDependant{domName(x,y,z), domSize(NX,NY,NZ), domPP(DOM),
           accPP(AT)}
8      a, b, c, d
9      @end domainDependant
10
11     @parallelRegion{appliesTo(CPU), domName(x,y), domSize(NX, NY)}
12     call add(a, b, c)
13     call mult(a, b, d)
14     @end parallelRegion
15   end subroutine
```

Implementation

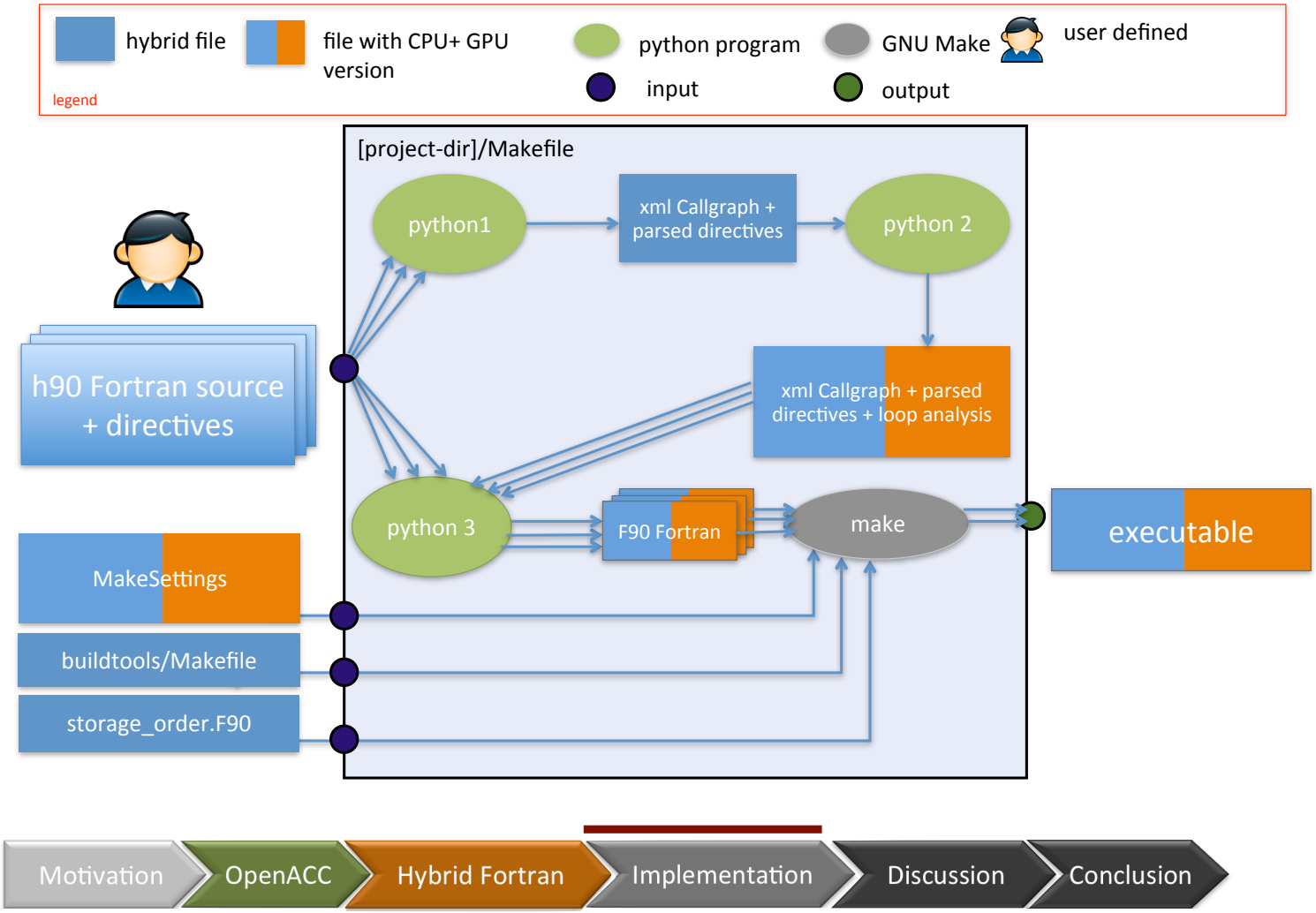
Build System | Implementation



Build System | Implementation

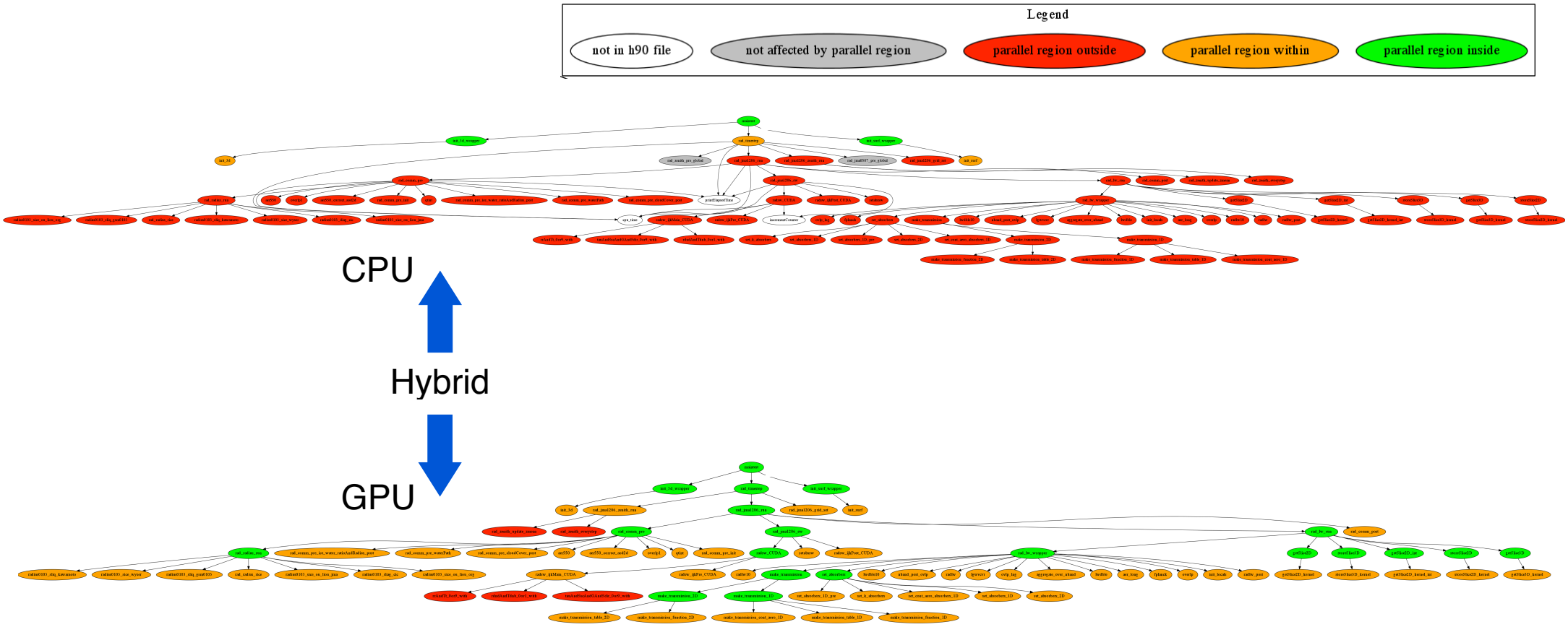


Build System | Implementation

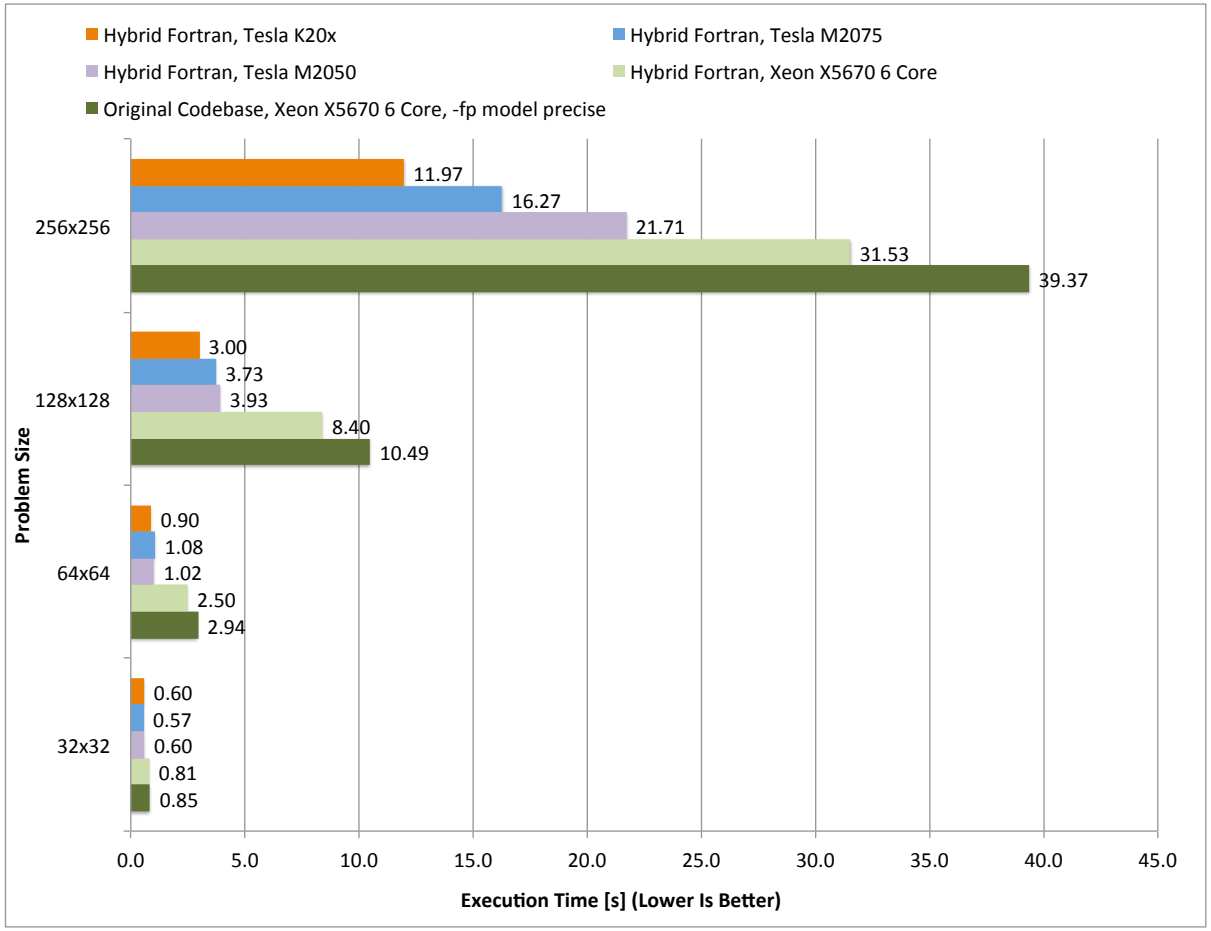


Results & Discussion

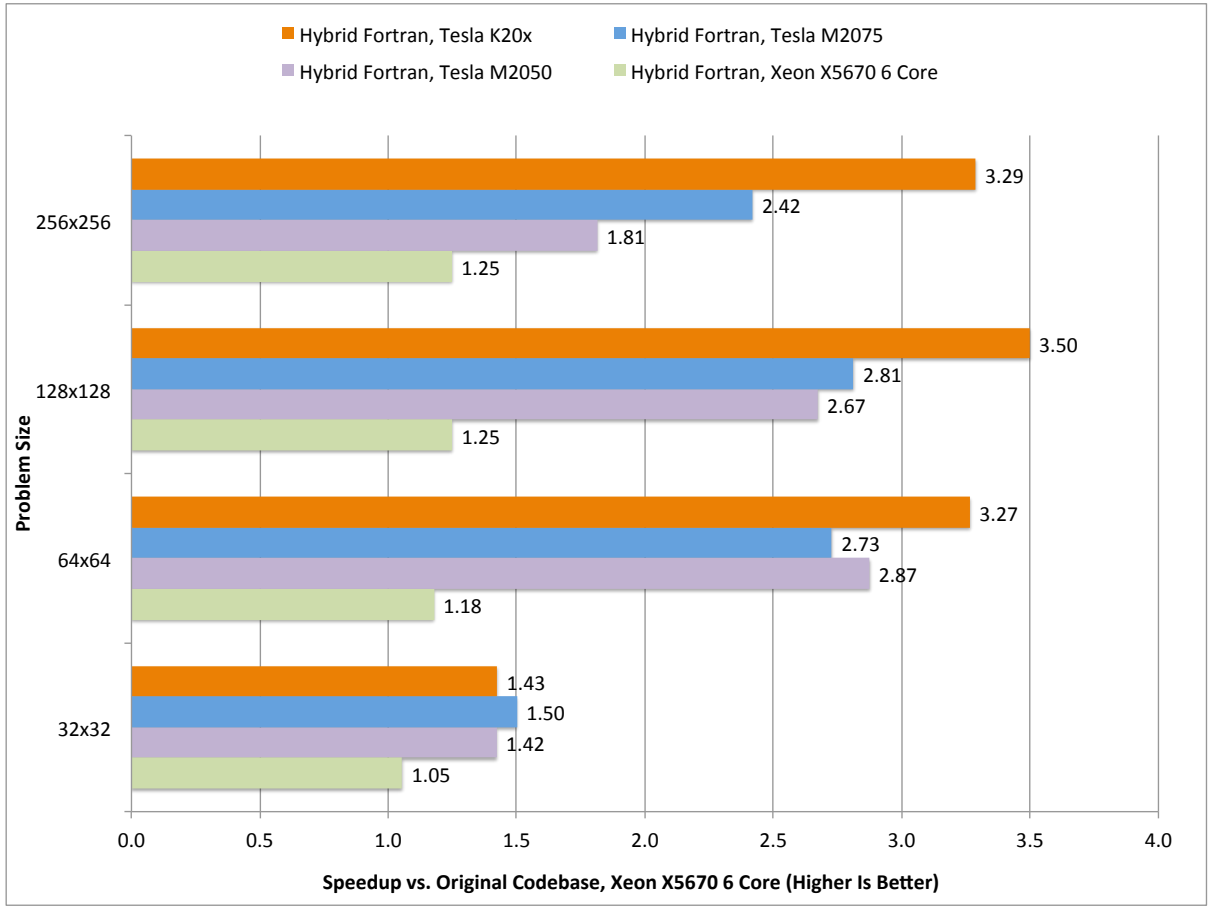
Scope of Current Implementation



Results: Execution Time



Results: Speedup



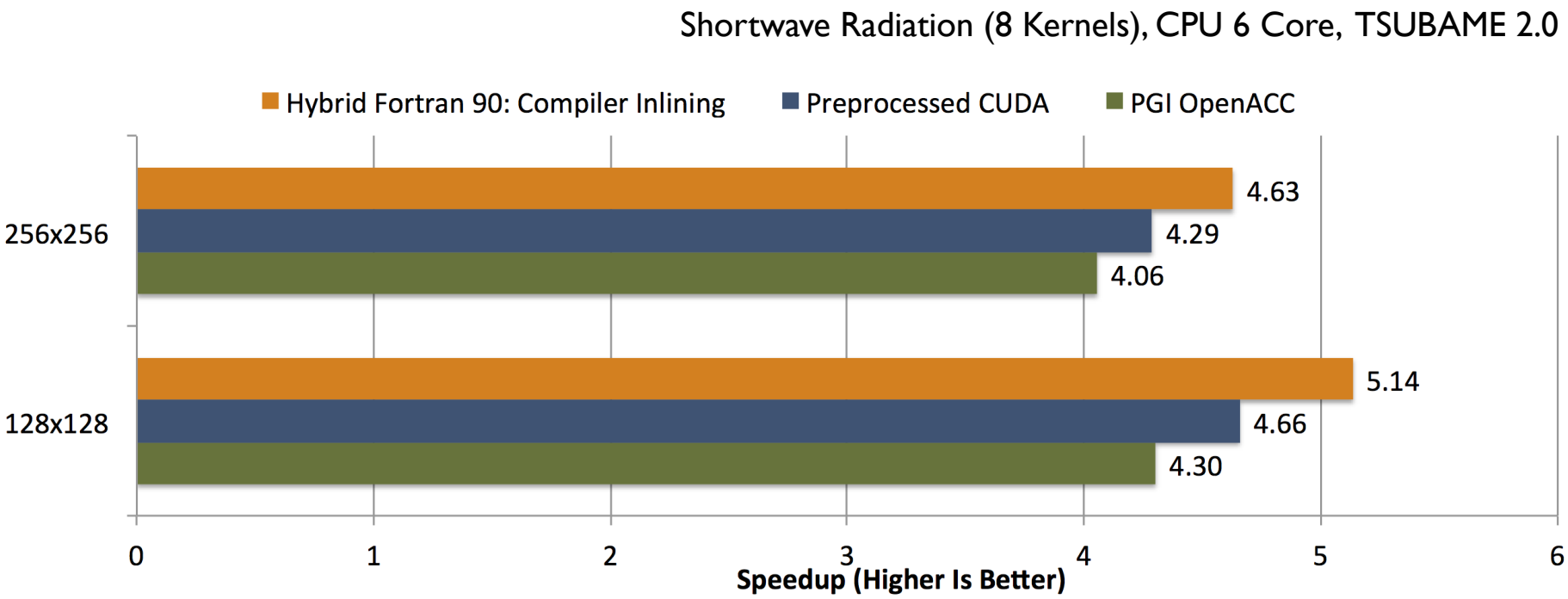
Expected Speedup | Results & Discussion

Tesla M2050 vs. Xeon X5670 | Tesla M2050 vs. Xeon X5670

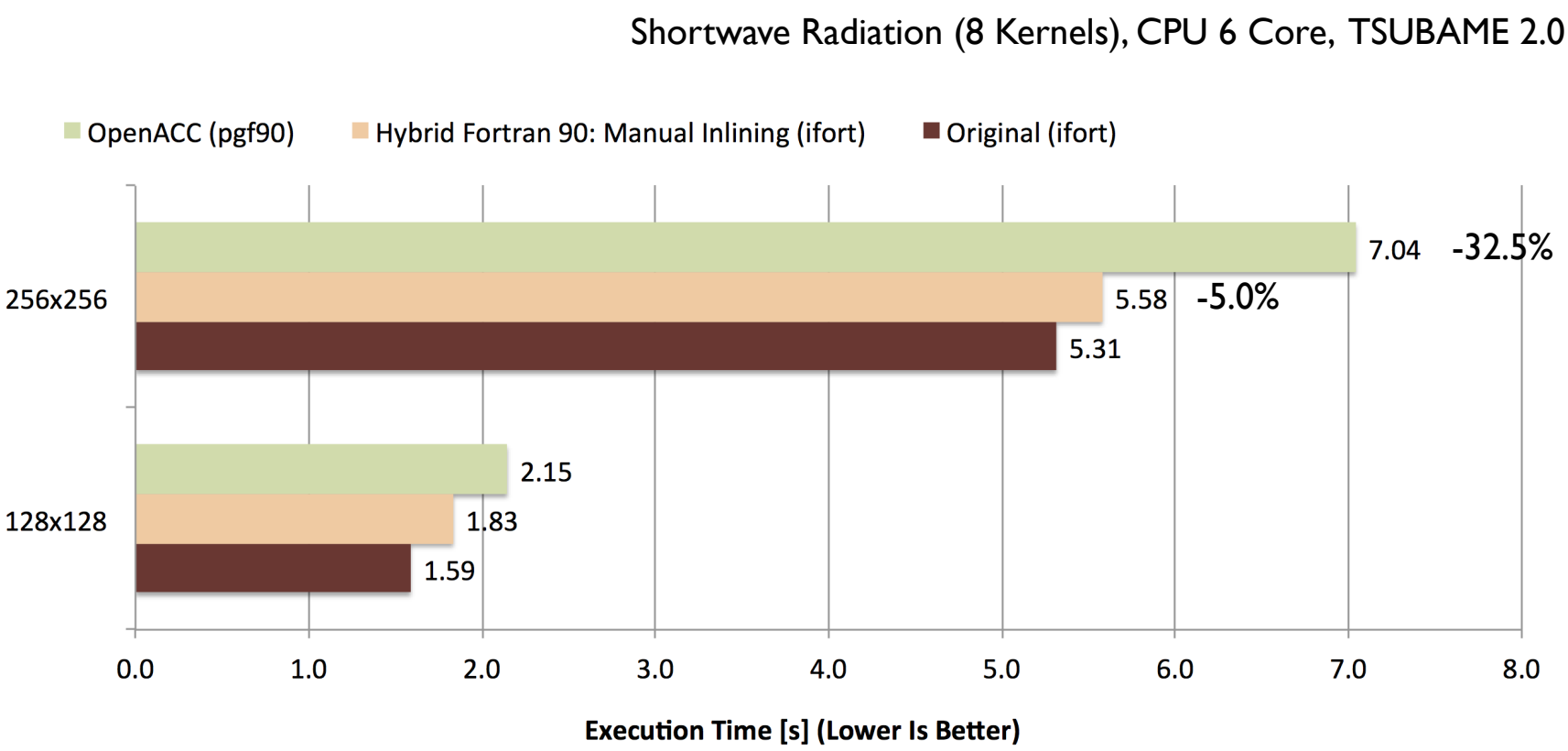
Memory Bandwidth Bounded Algorithms *	11.1 x	5.3 x
Computationally Bounded Algorithms [2, p. 4], [3], [5]	44 x	7.4 x

* Tesla sustained bandwidth according to CUDA SDK bandwidth test; CPU bandwidths according to [2, p. 4], [4, p. 7].

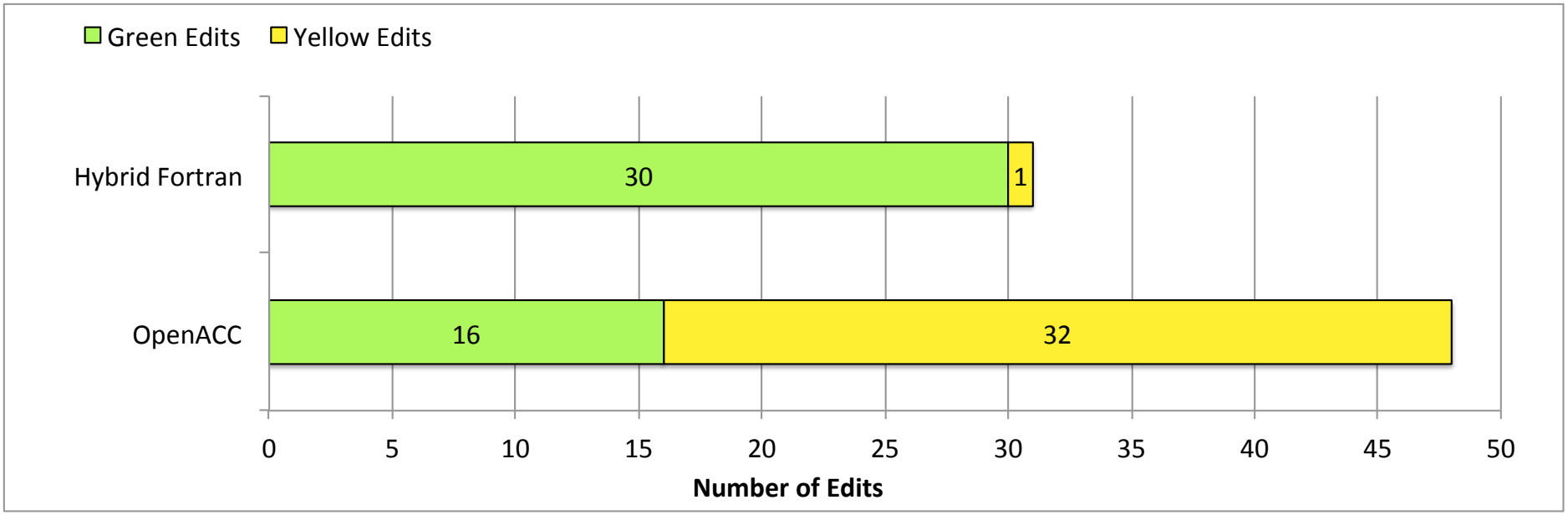
Tuned Results: GPU



Tuned Results: CPU



Usability | Results & Discussion



Conclusion

1. Dynamic scripting languages can be successfully used to enhance the functionality of a compiled language.
2. Hybrid Fortran allows us to get GPU compatibility with baseline performance quickly, without compromising CPU performance.
3. We still have all the flexibility for performance tuning.

Future Work

- Full scale ASUCA Physical Core implementation
- Pending Improvements for **Hybrid Fortran**
  1. General Stencil compatibility
  2. Support for more implementations and hardware, e.g. OpenCL, Xeon Phi
  3. Data handling simplifications

References

[1] K. Kawano, J. Ishida, and C. Muroi, “Development of a New Nonhydrostatic Model ASUCA at JMA”, 2010.

[2] Intel Corporation, “Specifications Intel Xeon Processor X5670, *http: //ark.intel.com/products/47920/Intel-Xeon-Processor-X5670-% 2812M-Cache-2_93-GHz-6_40-GTs-Intel-QPI%29*, 2010. [Online; accessed October 19, 2012]

[3] A. Vladimirov, “Arithmetics on Intel’s Sandy Bridge and Westmere CPUs”, *http://research.colfaxinternational.com/file.axd?file=2012%2F4%2FColfax_FLOPS.pdf* 2012. ; [Online, accessed October 19, 2012].

[4] NVIDIA, “Specifications NVIDIA Tesla C2050 GPU computing processor”, *http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C2050_C2070_jul10_lores.pdf*, 2010. [Online; accessed September 26, 2012]

[5] S. Williams, A. Waterman, and D. Patterson, “Roofline: An Insightful Visual Performance Model for Multicore Architectures”, 2009.

Next Steps + Questions

- Hybrid Fortran will be available under LGPL license by this weekend!
- E-Mail me for GitHub-link, thesis, slides, further questions..

  michel@typhooncomputing.com