



#### **Extending the LLVM/Clang Framework for OpenMP Metadirective Support**

#### Alok Mishra<sup>1</sup>, Abid M. Malik<sup>2</sup> and Barbara Chapman<sup>1,2</sup>

<sup>1</sup>Stony Brook University - USA, <sup>2</sup>Brookhaven National Laboratory - USA





- The mission of the OpenMP ARB (Architecture Review Board) is to standardize directive-based multilanguage high-level parallelism that is performant, productive and portable.
- De-facto portable parallel programming since 1997
  - o Compiler directives
  - Data, task, SIMD parallelism
  - o Multicores, GPUs
  - User specifies the strategy, not the details

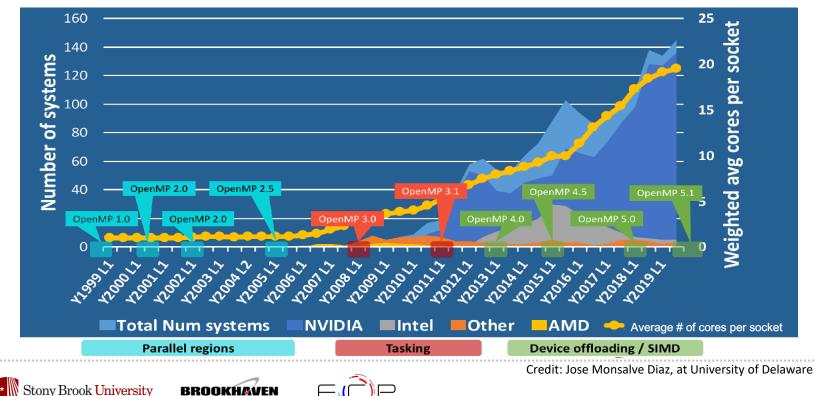




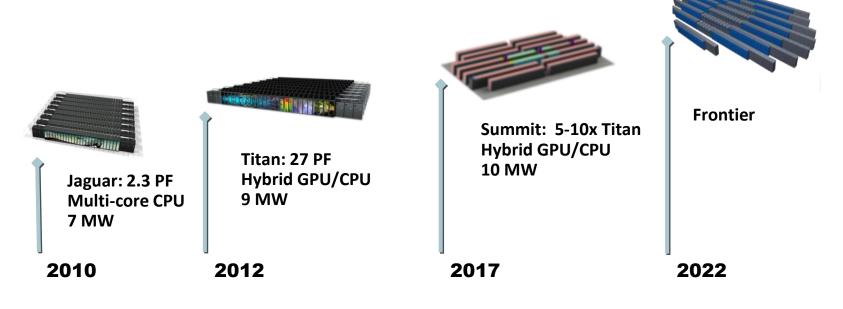


## **OpenMP Evolution**

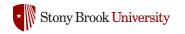
(www.top500.org)



## **Roadmap to Exascale (ORNL View)**



#### **Exascale Computing Project (ECP)**

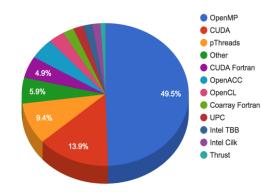






## Growing Relevance of OpenMP: NERSC Example

2015: OpenMP is about 50%, out of all choices of X



Update late 2016: 75% of codes use OpenMP

Courtesy of NERSC







- NERSC announced that OpenMP would be main on-node programming interface for future Perlmutter platform
- Desired features (mostly in OpenMP 5.0):
  - Programming memories (HBM, user-managed caches, unified NUMA memory)
  - Optimization of complex data structure motion (e.g., deep copy) Improved C++/object-oriented programming support
  - Accelerator offloading
  - o Tasks: priorities, groups, better affinity, improved scheduling
  - Task / thread affinity
  - Variant directives
    - declare variant
    - metadirective

#### **Metadirective**

The syntax of a metadirective takes one of the following forms:

#pragma omp metadirective [clause[[,]clause]...] new-line

or

#pragma omp begin metadirective [clause[[,]clause]...] new-line
 stmt(s)
#pragma omp end metadirective

where *clause* is one of the following:

when (context-selector-specification: [directive-variant])
default (directive-variant)

An executable directive that conditionally resolves to another directive at compile time by selecting from multiple directive variants based on traits that define an OpenMP condition or context.





C / C++



#### **Context Selectors**

The syntax to define a *context-selector-specification* is the following:

```
trait-set-selector[, trait-set-selector[,...]]
```

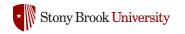
```
trait-set-selector:
    trait-selector-name={trait-selector[, trait-selector[, ...]]}
```

```
trait-selector:
```

```
trait-selector-name[([trait-score: ] trait-property[, trait-property[, ...]])]
```

trait-score:

```
score (score-expression)
```







#### **Trait Selectors**

#### Trait Selectors

- o construct
  - target, teams, parallel, for and simd
- $\circ$  device
  - kind, isa, arch
- $\circ$  implementation
  - vendor, extension
- o user







# Compatibility

✤ A given context selector is compatible with a given OpenMP context if:

- All user selectors are true\*
- All selectors in the context selector appear in the corresponding OpenMP context
- Properties for each context selector, are a subset of the corresponding trait of the OpenMP context
- Selectors in the construct set of the context selector appear in the same relative order as their corresponding traits in the construct trait set of the OpenMP context



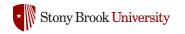




## **Metadirective Example**

```
for (idev=0; idev < omp_get_num_devices(); idev++)
#pragma omp target device(idev)
#pragma omp metadirective \
    when( implementation={vendor(nvidia)}, device={arch("kepler")}: \
        teams num_teams(512) thread_limit(32) ) \
    when( implementation={vendor(amd)}, device={arch("fiji")}: \
        teams num_teams(512) thread_limit(64) ) \
    default(teams)
#pragma omp distribute parallel for
    for (i=0; i<N; i++)
        work_on_chunk(idev,i);</pre>
```

#### Compile time selection of hardware support







## **Related Work**

- Some compilers provide OpenMP 5.0 with limitations
- No compiler provides metadirective implementation
  - $\circ$  Cray
  - o GNU
  - o Intel
  - o LLVM
  - Rose (partial support)
    - > Yan et. al. Extending OpenMP Metadirective Semantics for Runtime Adaptation







## **Dynamic User Condition**

- Extend metadirective to resolve a user condition at runtime.
- Create an if-then-else statement
  - Condition will be evaluated and resolved during execution

Code written using metadirective in OpenMP 5.0

Stony Brook University

```
#pragma omp metadirective \
    when(user={condition(N>1000)}: target teams distribute parallel for) \
    when(user={condition(N>100 && N<=1000)}: parallel for) \
    default()
for( int i=0; i<N; i++)
    compute();</pre>
```

### **Dynamic Metadirective Example**

			<b>—</b>	
if(N<10) {		if(N<10) {		
// Block 1 Serial		// Block 1 Serial		
<pre>for(int i=0; i<n; i++)<="" pre=""></n;></pre>		<pre>for(int i=0; i<n; i++)<="" pre=""></n;></pre>		
<pre>for(int j=0; j<n; j++)<="" pre=""></n;></pre>		<pre>for(int j=0; j<n; j++)<="" pre=""></n;></pre>		
<pre>for(int k=0; k<n; k++)<="" pre=""></n;></pre>		<pre>for(int k=0; k<n; k++)<="" pre=""></n;></pre>		
C[i][j] += A[i][k] * B[k][j];			C[i][j] += A[i][k] * B[k][j];	
} else if (N<100) {		} else if (N<100) {		
// Block 2 CPU Parallel		<pre>// Block 2 CPU Parallel</pre>		
<pre>#pragma omp parallel for collapse(2)</pre>		<pre>#pragma omp parallel for co</pre>	<pre>#pragma omp parallel for collapse(2)</pre>	
<pre>for(int i=0; i<n; i++)<="" pre=""></n;></pre>		<pre>for(int i=0; i<n; i++)<="" pre=""></n;></pre>	<pre>for(int i=0; i<n; i++)<="" pre=""></n;></pre>	
<pre>for(int j=0; j<n; j++)<="" pre=""></n;></pre>		<pre>for(int j=0; j<n; j++)<="" pre=""></n;></pre>	<pre>for(int j=0; j<n; j++)<="" pre=""></n;></pre>	
<pre>for(int k=0; k<n; k++)<="" pre=""></n;></pre>		<pre>for(int k=0; k<n; k++<="" pre=""></n;></pre>	<pre>for(int k=0; k<n; k++)<="" pre=""></n;></pre>	
C[i][j] +=	אויוווין א דוווין.		╋ B[k][j];	
} else {	<pre>#pragma omp metadirective \</pre>			
// Block 3 GPU off.			present	
#ifdef NVPTX	target teams distribute parallel for) $\setminus$			
<pre>#pragma omp target</pre>			64")}: \	
#else	parallel for collapse(2)) $\setminus$		ute parallel for) $\setminus$	
#pragma omp parall	default()		lapse(2))	
#endif	<pre>for(int i=0; i<n; i++)<="" pre=""></n;></pre>			
<pre>for(int i=0; i<n< pre=""></n<></pre>	<b>for</b> (int j=0; j <n; j++)<="" td=""><td></td></n;>			
<pre>for(int j=0; j</pre>	<pre>for(int k=0; k<n; k++)<="" pre=""></n;></pre>			
<pre>for(int k=0;</pre>			* B[k][j];	
C[i][]j] +	}			
}				
L			-	
Stony Brook University		)		
<b></b>	NATIONAL LABORATORY			

EXASCALE COMPUTING PROJECT

\*

#### Program Transformation for automatic GPU-Offloading with OpenMP

- Parallel regions detection
  - $\circ$  Loops, functions
- Patterns / Data Analysis
  - Suggest and evaluate several variants resulting from tiling, interleaving, collapsing of kernel
- Cost model
  - o ML based to determine profitability of kernel
- Code Generation
  - Insert pertinent OpenMP directives

Compiler framework to automatically offload profitable regions of code using OpenMP



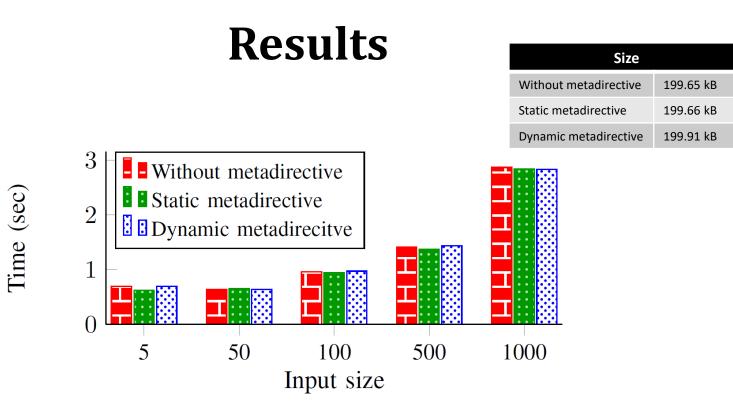




# **Experimental Setup**

- SOLLVE compiler (LLVM-12.0)
- Summit Supercomputing Cluster
  - o 2 IBM POWER9 CPUs
  - $\circ$   $\,$  6 NVIDIA Volta (V100) GPUs  $\,$
  - $\circ$  GCC version6.4.0
  - CUDA version 10.1.105
- SeaWulf computational cluster
  - o 2 Intel Xeon E5-2683v3 CPUs
  - o 4 Nvidia Tesla (K8 0) GPUs
  - $\circ$  GCC version 6.5.0
  - o CUDA version 9.1.185





Execution time (in sec) of matrix multiplication for 3 different implementations







### **Benchmarks**

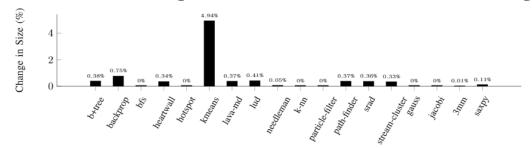
Modified 18 benchmark applications from the Rodinia Benchmark Suite

- $\circ \quad \text{use our dynamic metadirective implementation} \\$
- diversity of the domains in which each of its applications falls
- ✤ 4 micro benchmarks
  - Matrix Multiplication, Gauss Seidel Method, Laplace Equation and SAXPY
- Help the application developers learn how to use dynamic metadirective in real applications from different domains.
- Built application for :
  - Summit compute capability 7.0 for Volta V100
  - SeaWulf compute capability 3.5 for Tesla K80

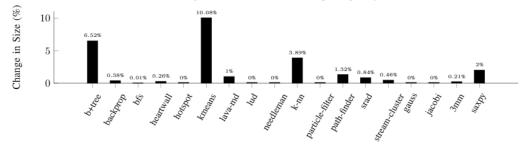




## **Results (Executable Size)**



a: Summit using cuda version 10.1.105 and compute capability 7.0



b: Seawulf using cuda version 9.1.85 and compute capability 3.5

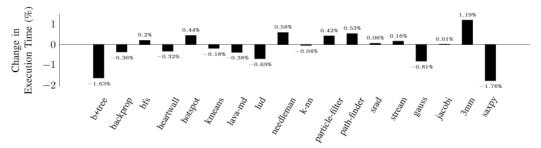
Percentage increase in size of the executable built with our dynamic metadirective implementation in LLVM 12.0.0



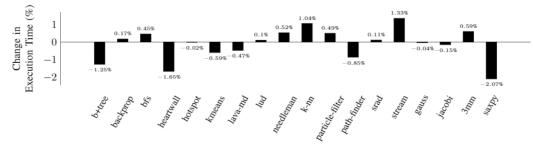




## **Results (Running Time)**



a: A Summit node with 2 IBM Power9 CPUs (128 threads) and NVIDIA Volta V100 GPU



b: A Seawulf node with 2 Intel Xeon E5-2683v3 CPUs (28 threads) and NVIDIA Tesla K80 GPU

Percentage change in running time of the executable built with our dynamic metadirective implementation







### **Future Work**

- Finalize implementation testing and upstream the patch to LLVM
- Integrate the patch with SOLLVE
- Research aimed at automatic code generation for heterogeneous devices
- Explore complex user-defined conditions
- Adding template definition to metadirective for type manipulation
- Explore potential pitfalls for downstream code generation







## Conclusion

- OpenMP 5.0 introduces the metadirective directive that conditionally resolves to another directive at compile time.
- Extension of user-defined contexts to allow directive variant selection at runtime.
- Modified the open source LLVM compiler.
- Modifications to the Rodinia Benchmark Suite enabled us to explore the impact of dynamic metadirective in an OpenMP context.
- Provides a guideline to the end users to help them apply these features in real applications.
- Minimal or no overhead to the user application.













# **Thank You!**

### **Questions?**

You can find our work at:

https://github.com/almishra/metadirective.git

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. The authors would like to thank Stony Brook Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University for access to the SeaWulf computing system, which was made possible by a \\$1.4M National Science Foundation grant (\#1531492).