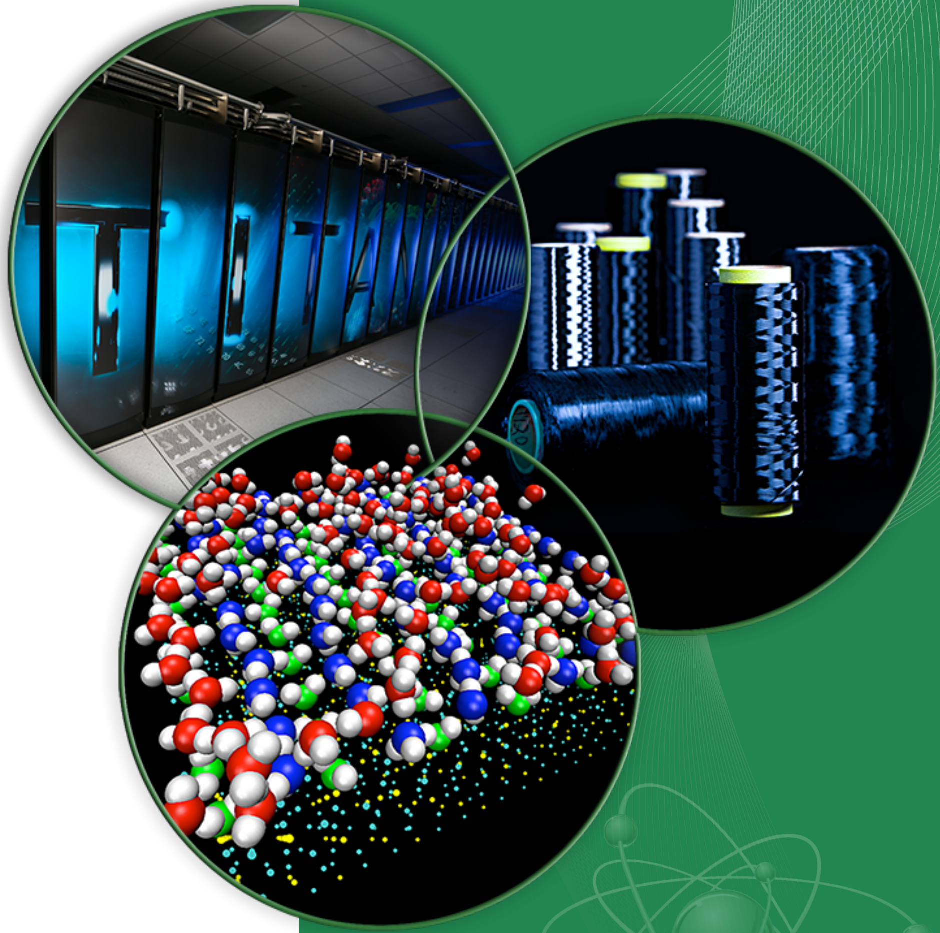
Early Experiences in using OpenMP 4 for SPEC ACCEL



Presented by:

Oscar Hernandez (ORNL)

Kalyan Kumaran (ANL)

Arpith Jacob (IBM)

Alexander Bobyr (Intel)

Graham Lopez (ORNL)

David Bernholdt (ORNL)

Work done in collaboration with SPEC/HPG group.
This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE
National Laboratory

SPEC HIGH PERFORMANCE GROUP (HPG)

- Develops benchmarks that represent high-performance computing applications for standardized, cross-platform performance evaluation.
- Current Benchmarks
  - SPEC OMP2012, SPEC MPI2007, SPEC ACCEL 1.0, 1.1
- **Working toward OpenMP 4 SPEC ACCEL 1.2**
  - Portable across architectures (host, GPUs, XeonPhi)
  - Works with at least two compilers
- Active members:
  - NVIDIA*, SGI, Intel*, IBM*, AMD, Argonne*, ORNL*, HZDR, Oracle, University of Delaware, University of Virginia, RWTH Aachen University, University of Illinois, Indiana University, TU Dresden

*Present at the DOE workshop

OpenMP 4.0 – Performance Portability (Meeting in Berlin)

- We had a meeting and discussed a strategy on how to write “performance portable” style in OpenMP 4
  - Initially members had different views.
  - We agreed on some “guidelines” on how to write portable code
  - We used these “guidelines” and successfully parallelized the 16 benchmarks with OpenMP 4

SPEC ACCEL: OpenMP 4 Candidates*

* (SPEC/HPG – Confidential)

OpenACC Benchmarks	Language	Origin	Domain
503.ostencil	C	Parboil, University of Illinois	Thermodynamics
504.olbm	C	Parboil, University of Illinois	CFDm Lattice Boltzmann
514.omriq	C	Rodinia, University of Virginia	Medicine
550.md	Fortran	Indiana University	Molecular Dyn.
551.palm	Fortran	Leibniz University of Hannover	Large-eddy sim.
552.ep	C	NAS Parallel Benchmarks (NPB)	Embarrassing P.
553.clvrleaf	C, Fortran	Atomic Weapons Establishments	Hydrodynamics
554.cg	C	NPB	Conjugate Grad.
555.seismic	Fortran	GeoDynamics.org	Seismic Wave Modeling (PDE)
556.sp	Fortran	NPB	Scalar Peta-d solv
557.csp	C	NPB	Scalar Peta-d solv
559.miniGhost	C, Fortran	Sandia National Laboratory	Finite difference
560.ilbdc	Fortran	SPEC OMP2012	Fluid Mechanics
563.swim	Fortran	SPEC OMP2012	Weather
570.bt	C	NPB	BTS 3D PDE

Guidelines – To write OpenMP 4 "Performance Portable Style"

- Use OpenMP 4 "Accelerator Model"
- Do not specify:
  - # of teams
  - # thread_limit,
  - # of threads – in parallel regions
  - SIMD length
  - dist_schedule – in distribute
  - loop schedules – in parallel do
- Compiler implementers should pick these values to enable performance portability

Guidelines – To write OpenMP 4 “Performance Portable Style”

- For level-1 loopnest
    - #pragma target teams distribute parallel for simd
- For perfectly nested loops
    - Use the following nesting of parallelism

```
#pragma omp target teams distribute parallel for collapse(N)
    for(i=0;….)
      for(j=0;….)
    #pragma omp simd
        for(k=0;…)
```

- Parallelize the inner loops always with SIMD
- Do not collapse inner loops

Guidelines for OpenMP 4

- Reductions
  - Reduction variables need to be mapped to/from

```
#pragma omp target map(tofrom:sum)
#pragma omp teams distribute parallel for reduction(+:sum)
   for(…. )
        sum = sum + ….
```

- Privatization
  - We should only privatize only at a nesting level

```
#pragma omp teams distribute parallel for  // private(yy, zz)
for(i= …. )
   for(j= … )
#pragma omp simd private(yy,zz)
      for(z= …
              yy =
              zz =
```

Guidelines for OpenMP 4

- Don’t merge target regions if they have dependences across loopnests (otherwise do)

```
#pragma omp target teams distribute parallel for
    for(i=…)
        a[i] =
#pragma omp target teams distribute parallel for
    for(i=…)
       b[i] =
```

- To:

```
#pragma omp target teams
#pragma omp distribute
        for(i=…)
           a[i] =
#pragma omp distribute
      for(i=…)
         b[i] =
```

Example – jacobi.f – Portable OpenMP

```
!$omp target map(tofrom: error)
!$omp teams distribute parallel do reduction(+:error)
      do j = 2,m-1
!$omp simd private(resid)
        do i = 2,n-1
          resid = [computes resid from I,j-arrays]
*         error = error + resid*resid
        end do
      enddo
!$omp end teams distribute parallel do
!$omp end target
```

Preliminary results are showing

- If you want performance portability in your codes across platforms:
  - USE OPENMP 4.0 “Accelerator Model”
  - This includes:
    - GPUs
    - Xeon Phi (self-hosted)
    - CPUs
- Compilers should tune and pick code for a given architecture – unless you want to auto-tune.
- Compilers are still working on their OpenMP implementations and few support multiple architectures for OpenMP 4.0 accelerator model