OpenMP

SC'21 Booth Talk Series

SC21

St. Louis, MO | science & beyond.

Using OpenMP Loop Transformations with Clang

Michael Kruse, Argonne National Laboratory

Outline

Unrolling

Non-Standard Extensions

- Cray
  ```
  #pragma [_CRI] unroll 4
  ```
- icc
  ```
  #pragma unroll 4
  ```
- xlc
  ```
  #pragma unroll(4)
  ```
- HP
  ```
  #pragma UNROLL_FACTOR 4
  ```
- gcc
  ```
  #pragma GCC unroll 4
  ```
- clang
  ```
  #pragma unroll 4
  #pragma clang loop unroll_count(4)
  ```
- msvc

  ???

Full Unrolling

OpenMP 5.1

```
#pragma omp unroll full
for (int i = 0; i < 4; i += 1)
  body(i);
```


Full Unrolling

OpenMP 5.1

```
#pragma omp unroll full
for (int i = 0; i < 4; i += 1)
  body(i);
```

```
body(0);
body(1);
body(2);
body(3);
```


Partial Unrolling

OpenMP 5.1

```
#pragma omp unroll partial(4)
for (int i = 0; i < n; i += 1)
  body(i);
```

```
int i = 0;
for (; i+3 < n; i += 4) {
  body(i);
  body(i + 1);
  body(i + 2);
  body(i + 3);
}
for (; i < n; i += 1)
  body(i);
```


Heuristic Unrolling

OpenMP 5.1

```
#pragma omp unroll
for (int i = 0; i < n; i += 1)
  body(i);
```

or

```
#pragma omp unroll partial
for (int i = 0; i < n; i += 1)
  body(i);
```

```
int i = 0;
for (; i+? < n; i += ?) {
  body(i);
  ...
}
for (; i < n; i += 1)
  body(i);
```

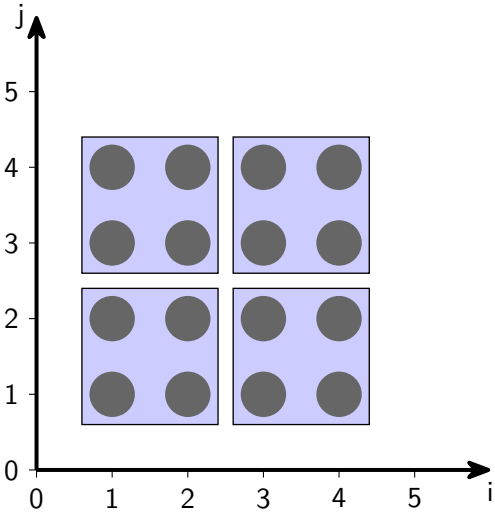

Complete Tiles

OpenMP 5.1

```
#pragma omp tile sizes(2,2)
for (int i = 1; i <= 4; ++i)
  for (int j = 1; j <= 4; ++j)
    body(i,j);
```


Complete Tiles

OpenMP 5.1

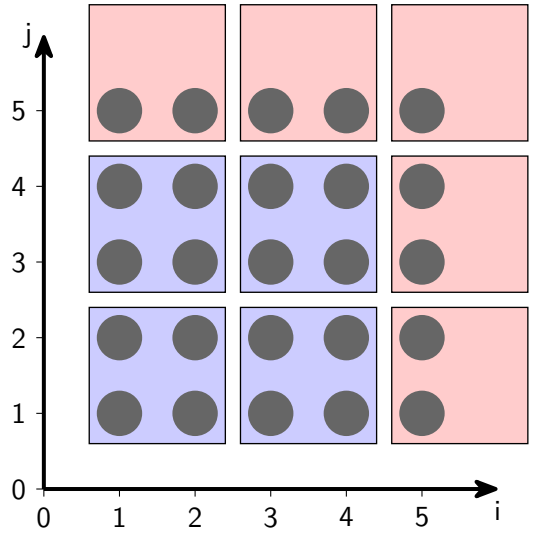



```
#pragma omp tile sizes(2,2)
for (int i = 1; i <= 4; ++i)
  for (int j = 1; j <= 4; ++j)
    body(i,j);
```

```
/* floor loops iterating over tiles */
for (int i1 = 0; i1 < 4; i1 += 2)
  for (int j1 = 0; j1 < 4; j1 += 2)
    /* tile loops over iterations */
    for (int i2 = i1; i2 < i1 + 2; i2 += 1)
      for (int j2 = j1; j2 < j2 + 2; j2 += 1)
        /* an iteration */
        body(1+i2,1+j2);
```


Partial Tiles

OpenMP 5.1




```
#pragma omp tile sizes(2,2)
for (int i = 1; i <= 5; ++i)
  for (int j = 1; j <= 5; ++j)
    body(i,j);
```

```
/* hot, streamlined code */
for (int i1 = 0; i1 < 4; i1 += 2)
  for (int j1 = 0; j1 < 4; j1 += 2)
    for (int i2 = i1; i2 < i1 + 2; i2 += 1)
      for (int j2 = j1; j2 < j2 + 2; j2 += 1)
        body(i2+1,j2+1);

/* special case code */
for (int i = 1; i < 5; ++i)
  for (int j = 1; j < 5; ++j)
    if (i >= 5 || j >= 5)
      body(i,j);
```


Loop Transformation Composition

- Rule: Loop-associated directives apply to the next line
  1. A base language canonical loop

```
#pragma omp taskloop
for (int i = 0; i < 128; ++i)
  body(i);
```

  2. The output of another loop transformation

```
#pragma omp taskloop
#pragma omp tile sizes(8)
for (int i = 0; i < 128; ++i)
  body(i);
```

```
#pragma omp taskloop
for (int i1 = 0; i1 < 128; i1 += 8)
  for (int i2 = i1; i2 < i1 + 8; i2 += 1)
    body(i2);
```


Multi-Level Tiling

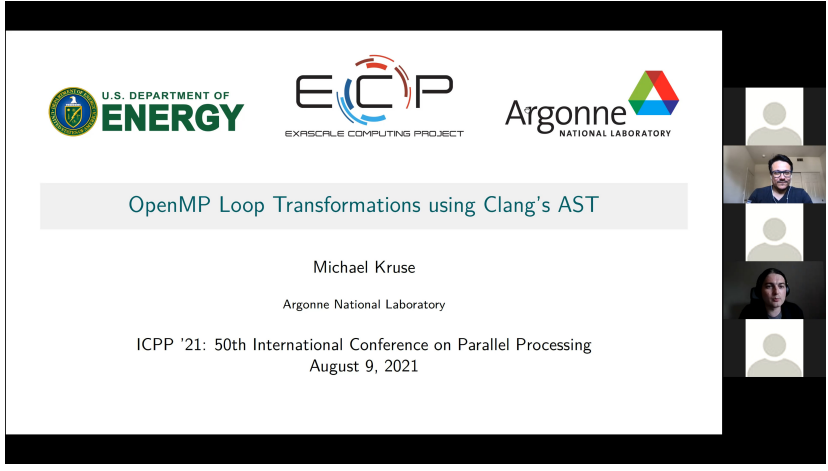
```
#pragma omp tile sizes(4, 4)
#pragma omp tile sizes(5,16)
for (int i = 0; i < 100; ++i)
  for (int j = 0; j < 128; ++j)
    A[i][j] = i*1000 + j;
```

```
#pragma omp tile sizes(4,4)
for (int i1 = 0; i1 < 100; i1+=5)
  for (int j1 = 0; j1 < 128; j1+=16)
    for (int i2 = i1; i2 < i1+5; ++i2)
      for (int j2 = j1; j2 < j1+16; ++j2)
        A[i2][j2] = i2*1000 + j2;
```

```
for (int i11 = 0; i11 < 100; i11+= 5*4)
for (int j11 = 0; j11 < 128; j11+=16*4)
    for (int i12 = i11; i12 < i11+( 5*4); i12+= 5)
    for (int j12 = j11; j12 < j11+(16*4); j12+=16)
        for (int i2 = i12; i2 < i12+ 5; ++i2)
        for (int j2 = j12; j2 < j12+16; ++j2)
            A[i2][j2] = i2*1000 + j2;
```


Outline

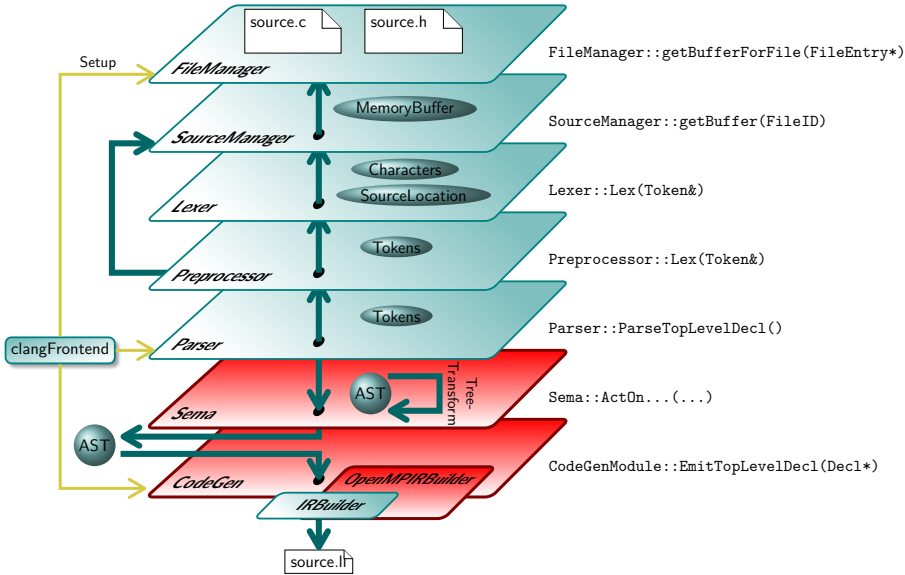
LLVM in Parallel Processing Workshop



https://oaciss.uoregon.edu/icpp21/videos/LLPP_ICPP_21.mp4
Starting at 0:36:50

Clang Layers

clang -cc1



Status

Available in Clang 13

Shadow AST Implementation

- Tiling:
  `https://reviews.llvm.org/D76342`
- Unrolling:
  `https://reviews.llvm.org/D99459`

OpenMPIRBuilder Implementation

Experimental: `-fopenmp-enable-irbuilder`

- `CanonicalLoopInfo`:
  `https://reviews.llvm.org/D90830`
- `collapseLoops`:
  `https://reviews.llvm.org/D93268`
- `unrollLoop`:
  `https://reviews.llvm.org/D107764`
- `tileLoops`:
  `https://reviews.llvm.org/D92974`
- `createWorkshareLoop`:
  `https://reviews.llvm.org/D92476`

Outline

Google Benchmark

Multi-Threading

```
template <int P>
static void benchmark_threads(benchmark::State &state) {
  double *A = aligned_alloc(...);


  for (auto _ : state) {
     #pragma omp parallel for \
                 schedule(static,P)
     for (int i = 0; i < M; ++i)
       ...
     /* implicit barrier */
  }

  free(A);
}
BENCHMARK_TEMPLATE(benchmark_threads, 1)->MeasureProcessCPUTime()->UseRealTime();
BENCHMARK_TEMPLATE(benchmark_threads, 2)->MeasureProcessCPUTime()->UseRealTime();
...
```


Google Benchmark

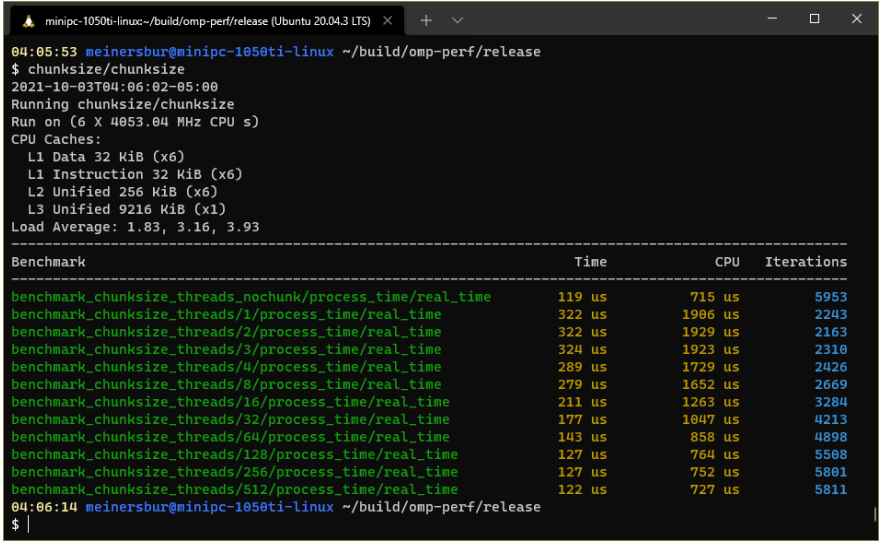
Offloading

```
template <int P>
static void benchmark_target(benchmark::State &state) {
  double *A = aligned_alloc(...);

  #pragma omp target data map(tofrom:A[0..N])
  for (auto _ : state) {
    #pragma omp target teams distribute parallel for \
                dist_schedule(static,P) schedule(static,P)
    for (int i = 0; i < M; ++i)
      ...
    #pragma omp taskwait
  }

  free(A);
}
BENCHMARK_TEMPLATE(benchmark_target, 1)->UseRealTime();
BENCHMARK_TEMPLATE(benchmark_target, 2)->UseRealTime();
...
```


Google Benchmark



SU3 Matrix-Vector Multiplication

Quantum Chromodynamics

```
#pragma omp unroll full
for (int j = 0; j < 3; ++j) {
  #pragma omp unroll full
  for (int k = 0; k < 3; ++k)
    C[i].v[j] += A[i].v[k] * B[i].c[k].v[j];
}
```

- $128 \cdot 1024$ $(= 2^{17})$ elements in `A`/`B`/`C`
- Single precision (`float`)

SU3 Matrix-Vector Multiplication

```
$ clang -O3 -fno-exceptions -march=native \
  -fopenmp -fopenmp-version=51 \
  -fno-unroll-loops -fno-vectorize \
  -Rpass=loop-unroll
```


SU3 Matrix-Vector Multiplication

```
$ clang -O3 -fno-exceptions -march=native \
  -fopenmp -fopenmp-version=51 \
  -fno-unroll-loops -fno-vectorize \
  -Rpass=loop-unroll
```

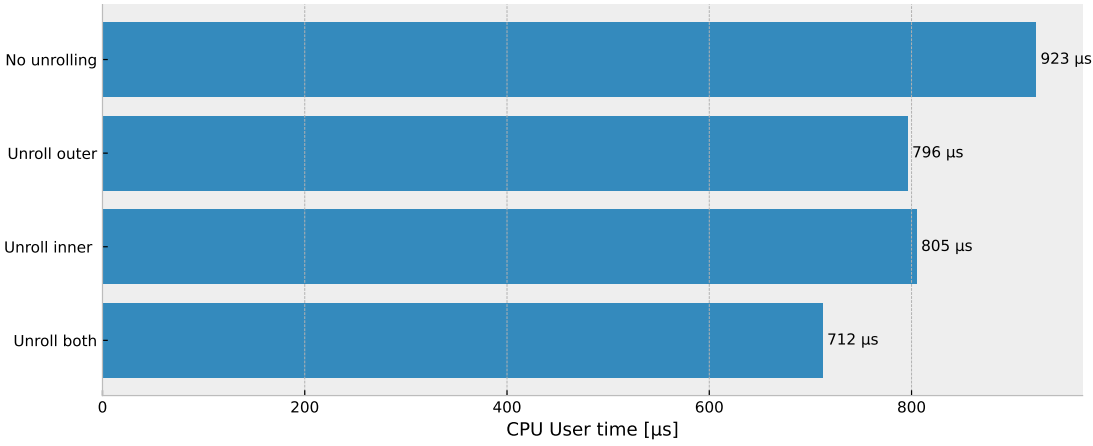

SU3 Matrix-Vector Multiplication

```
$ clang -O3 -fno-exceptions -march=native \
  -fopenmp -fopenmp-version=51 \
  -fno-unroll-loops -fno-vectorize \
  -Rpass=loop-unroll
```

```
remark: su3.cpp:217:15: completely unrolled loop with 3 iterations [-Rpass=loop-unroll]
remark: su3.cpp:215:7: completely unrolled loop with 3 iterations [-Rpass=loop-unroll]
```

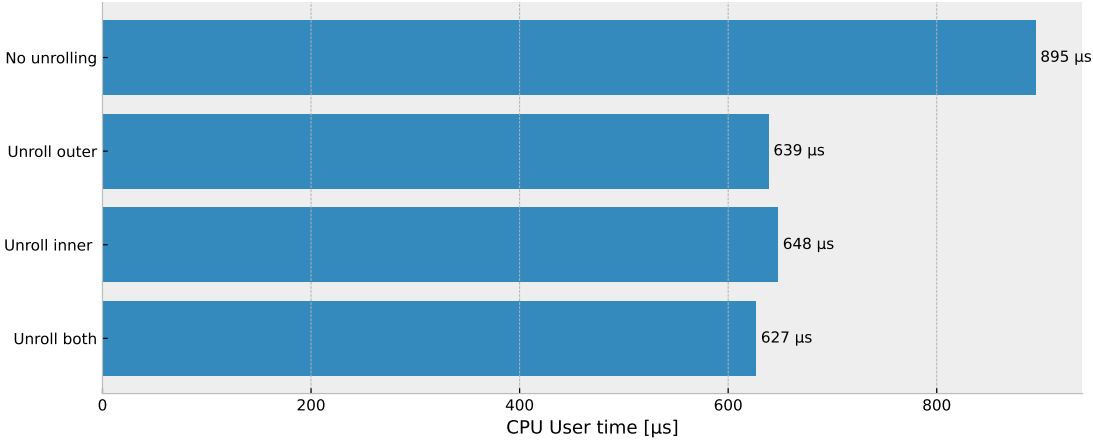

SU3 Single-Thread Performance

Intel(R) Xeon(R) Gold 6152 CPU @ 2.10GHz



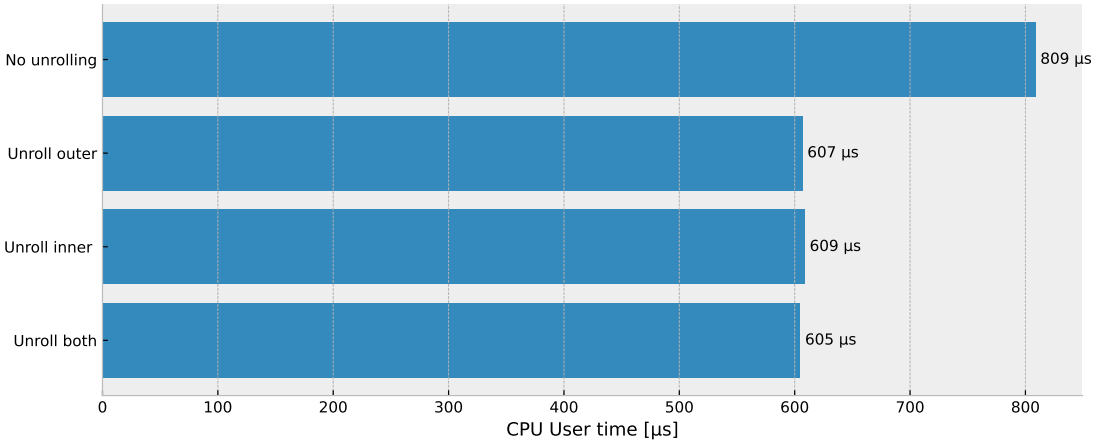
SU3 Single-Thread Performance

Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz



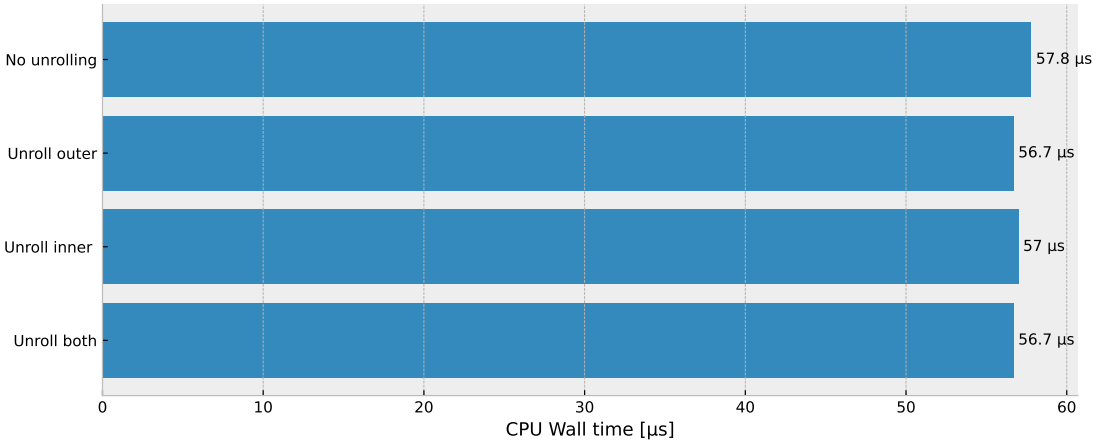
SU3 Single-Thread Performance

AMD EPYC 7532 32-Core Processor



SU3 Offloading Performance

NVIDIA Tesla V100-SXM2-32GB



Output Verification

```
$ clang -save-temps
```

su3-openmp-nvptx64.s

```
    mov.u64     %rd70, %rd28;
LBB0_7:
    .pragma "nounroll";
    add.s64         %rd65, %rd67, %rd71;
    ld.global.f32   %f4, [%rd65];
    ld.global.f32   %f5, [%rd70];
    mul.rn.f32      %f6, %f4, %f5;
    add.rn.f32      %f7, %f7, %f6;
    st.global.f32   [%rd29], %f7;
    add.s64         %rd71, %rd71, 4;
    cvt.u32.u64     %r21, %rd71;
    add.s64         %rd70, %rd70, 12;
    setp.eq.s32     %p4, %r21, 12;
    @%p4 bra         LBB0_8;
    bra.uni         LBB0_7;
```


Polybench 4.2.1b atax

Linear Algebra

```
for (int i = 0; i < m; i++) {
  tmp[i] = 0;

  #pragma omp unroll partial(P)
  for (int j = 0; j < n; j++)
    tmp[i] = tmp[i] + A[i][j] * x[j];

  #pragma omp unroll partial(P)
  for (int j = 0; j < n; j++)
    y[j] = y[j] + A[i][j] * tmp[i];
}
```

- `LARGE_DATASET` (default)
  - $\mathtt{m} = 1900$
  - $\mathtt{n} = 2100$
- `DATA_TYPE_IS_INT`

atax Unroll Heuristic

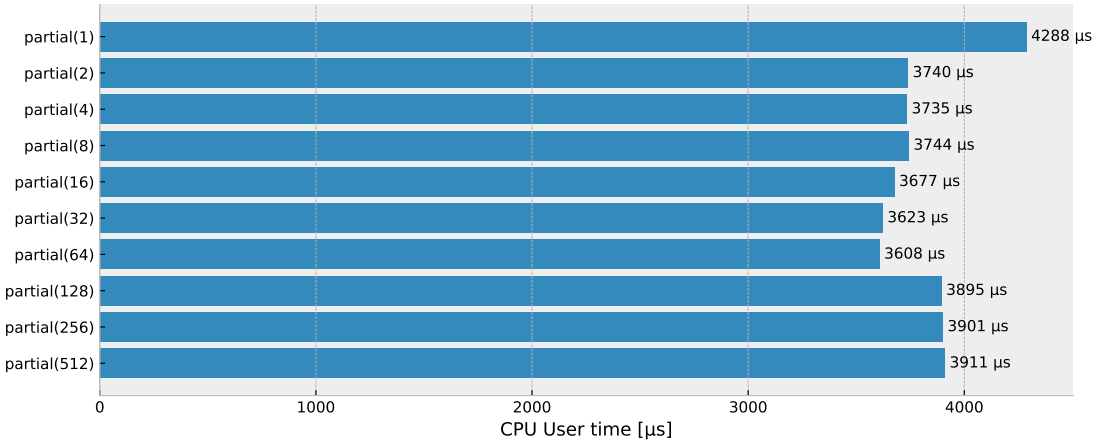
```
$ clang -O3 -fno-exceptions -march=native \
  -fopenmp -fopenmp-version=51 \
  -fno-unroll-loops -fno-vectorize \
  -Rpass=loop-unroll
```

```
#pragma omp unroll
```

```
remark: atax.cpp:947:5: unrolled loop by a factor of 8 with run-time trip count [-Rpass=loop-unroll]
remark: atax.cpp:951:5: unrolled loop by a factor of 8 with run-time trip count [-Rpass=loop-unroll]
```


atax Single-Thread Performance

AMD EPYC 7532 32-Core Processor



Polybench 4.2.1b heat-3d

Stencil

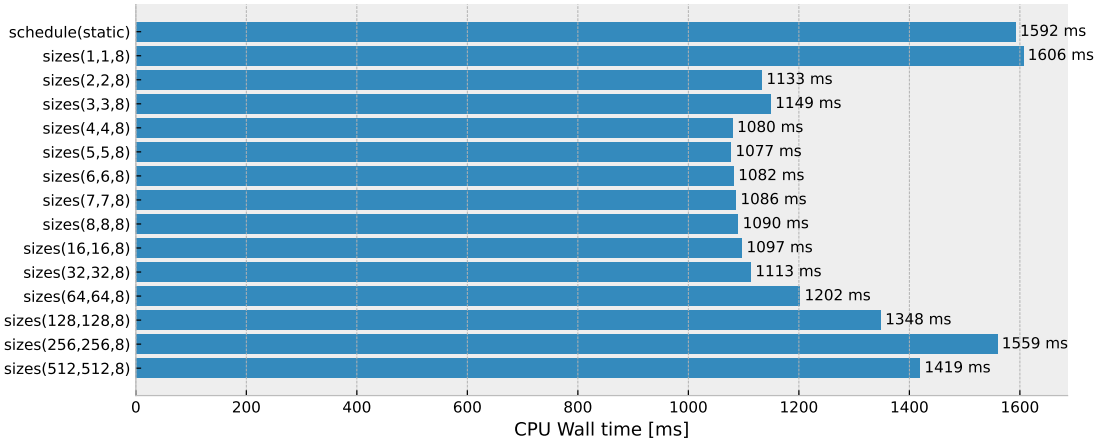
```
#pragma omp parallel
for (int t = 1; t <= steps; t++) {
  #pragma omp for collapse(3)
  #pragma omp tile sizes(P,P,8)
  for (int i = 1; i < n-1; i++)
    for (int j = 1; j < n-1; j++)
      for (int k = 1; k < n-1; k++)
        B[i][j][k] =  A[i][j][k] +
             0.125 * (A[i+1][j][k] - 2.0 * A[i][j][k] + A[i-1][j][k]) +
             0.125 * (A[i][j+1][k] - 2.0 * A[i][j][k] + A[i][j-1][k]) +
             0.125 * (A[i][j][k+1] - 2.0 * A[i][j][k] + A[i][j][k-1]);

  /* again with A and B swapped */
}
```

- steps $= 6$
- n $= 400$
- double precision

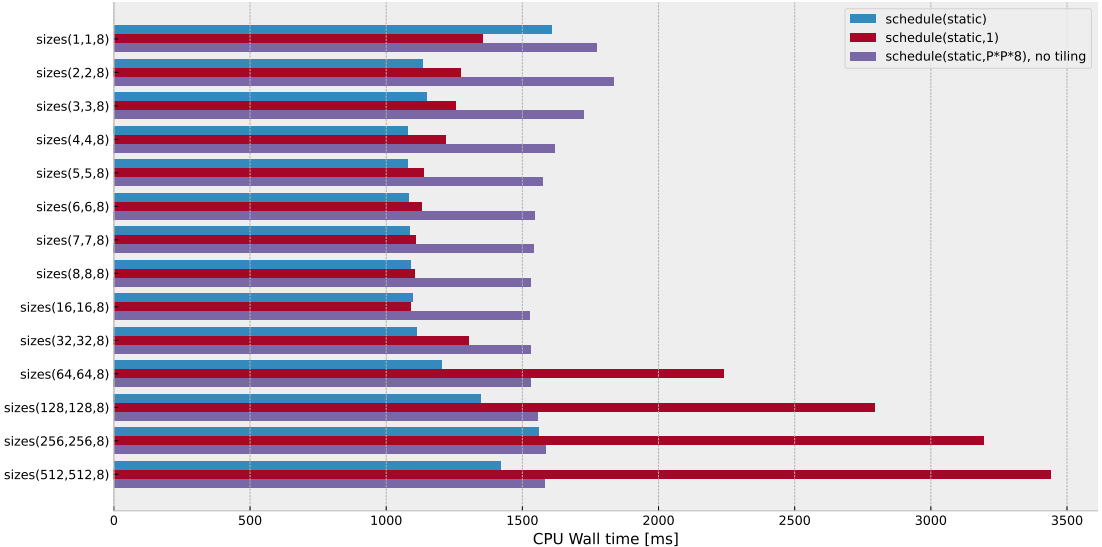
heat-3d Multi-Thread Performance

Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz (6 cores, 6 threads)



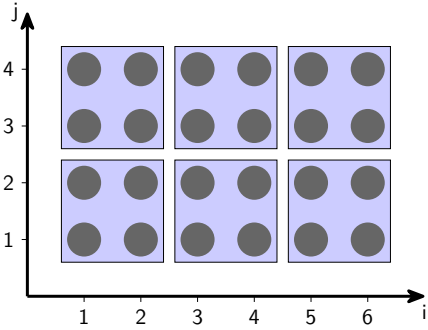
heat-3d Multi-Thread Performance

Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz (6 cores, 6 threads)

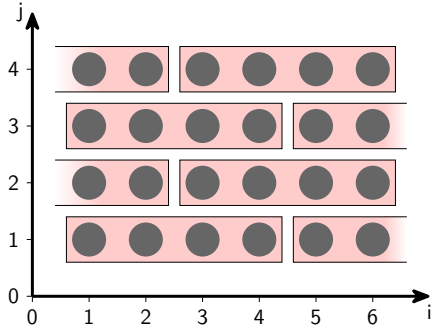


Complete Tiles

```
#pragma omp for schedule(static,1) collapse(2)
#pragma omp tile sizes(2,2)
for (int j = 1; j <= 4; ++j)
  for (int i = 1; i <= 6; ++i)
```

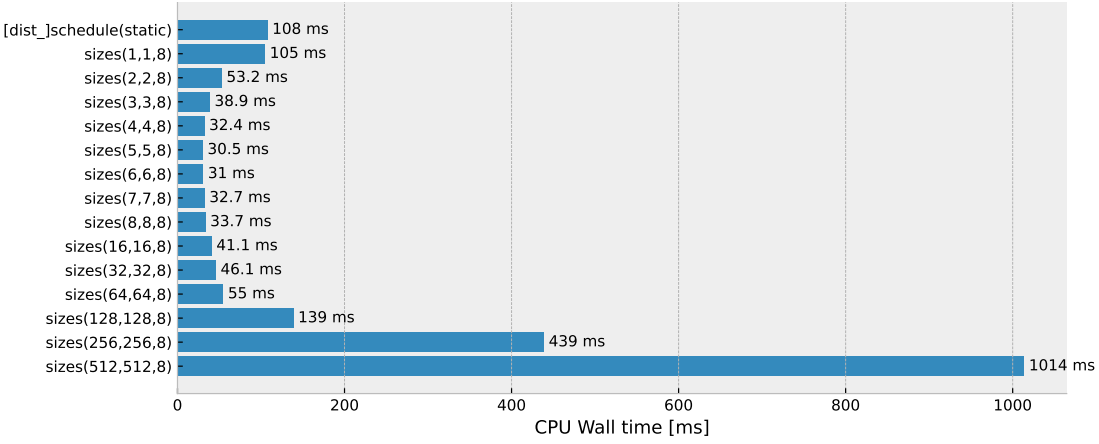



```
#pragma omp for schedule(static,4) collapse(2)
for (int j = 1; j <= 4; ++j)
  for (int i = 1; i <= 6; ++i)
```



heat-3d Offloading Performance

NVIDIA Tesla V100-SXM2-32GB



Outline

Summary

- New in OpenMP 5.1: Loop Transformations
- Available in Clang 13
- Google Benchmark for Microbenchmarking
  - Full unrolling to remove loop overhead
  - Partial unrolling to reduce loop overhead
  - Tiling for memory access locality
- Outcome depends on processor architecture

Possible Extensions for OpenMP 6.0

Additional Transformations

- Loop interchange
- Loop fission/fusion
- Peeling
- Loop unswitching
- Space-filling curves

Auxiliary Transformations

- Nestify
- Rectangify
- Collapse

More Clauses

- Control over remainder loops
- Enable safety checks

Compose Non-Outermost Generated Loops

- Loop identifiers
- `apply` clause

```
#pragma omp tile apply(floor: parallel for) \
        apply(tile : simd)
for (int i = 0; i < n; ++i)
    Body(i);
```

Semantics/Optimization Hints

- Assumptions:
  "ivdep", parallelizable, interchangeable, ...
- Expectations:
  loop trip count, hot/cold/dead, ...

OpenMP

SC'21 Booth Talk Series

openmp.org OpenMP API specs, forum, reference guides, and more

link.openmp.org/sc21 Videos and PDFs of OpenMP SC'21 presentations

Acknowledgments

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, in particular its subproject SOLLVE.

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

This material was based in part upon funding from the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.