Teaching LLVM's Optimizer to Understand OpenMP

Or, "Why parallel loops might slow down your code"

Hal Finkel
(Featuring work by Johannes Doerfert, see our IWOMP 2018 paper)

Problem 1: variable capturing

Input program:

```
int y = 1337;

#pragma omp parallel for

for (int i = 0; i < N; i++)
  g(y, i);
g(y, y);
```

Optimal program:

```
#pragma omp parallel for

for (int i = 0; i < N; i++)
  g(1337, i);
g(1337, 1337);
```

Clang output:

```
int y = 1337;
call fork_parallel(fn, &y);
g(y, y);
```

GCC output:

```
int y = 1337;
call fork_parallel(fn, &y);
g(1337, 1337);
```

Solution 1: variable privatization

Input program:

```
int y = 1337;

#pragma omp parallel for

for (int i = 0; i < N; i++)
  g(y, i);
g(y, y);
```

Optimized program:

```
int y = 1337; y_p = y;

#pragma omp parallel for

for (int i = 0; i < N; i++)
  g(y_p, i);
g(1337, 1337);
```

Clang output:

```
int y = 1337;
call fork_parallel(fn, &y);
g(y, y);
```

Clang output:

```
int y_p = 1337;
call fork_parallel(fn, &y_p);
g(1337, 1337);
```

Problem 2: (implicit) barriers

```
void copy(float* dst, float* src, int N) {
 #pragma omp parallel for
  for(int i = 0; i < N; i++)
    dst[i] = src[i];
}

void compute_step_factor(int nelr, float* vars,
                         float* areas, float* sf) {
#pragma omp parallel for
  for (int blk = 0; blk < nelr / block_length; ++blk) {
      ...
```

Problem 2: (implicit) barriers (con’t)

```
for (int i = 0; i < iterations; i++) {
  copy(old_vars, vars, nelr * NVAR);
  compute_step_factor(nelr, vars, areas, sf);
  for (int j = 0; j < RK; j++) {
    compute_flux(nelr, ese, normals, vars, fluxes, ff_vars,
                 ff_m_x, ff_m_y, ff_m_z, ff_dnergy);
    time_step(j, nelr, old_vars, vars, sf, fluxes);
  }
}
```

Problem 2: (implicit) barriers (con’t)

```
for (int i = 0; i < iterations; i++) {
  #pragma omp parallel for              // copy
  for (...) { /* write old_vars, read vars */ }
  #pragma omp parallel for              // compute_step_factor
  for (...) { /* write sf, read vars & area */ }
  for (int j = 0; j < RK; j++) {
    #pragma omp parallel for            // compute_flux
    for (...) { /* write fluxes, read vars & ... */ }
    ...
```

Solution 2: region expansion & barrier elimination

```
#pragma omp parallel
for (int i = 0; i < iterations; i++) {
  #pragma omp for nowait          // copy
  for (...) { /* write old_vars, read vars */ }
  #pragma omp for   nowait        // compute_step_factor
  for (...) { /* write sf, read vars & area */ }
  for (int j = 0; j < RK; j++) {
    #pragma omp for               // compute_flux
    for (...) { /* write fluxes, read vars & ... */ }
  ...
```

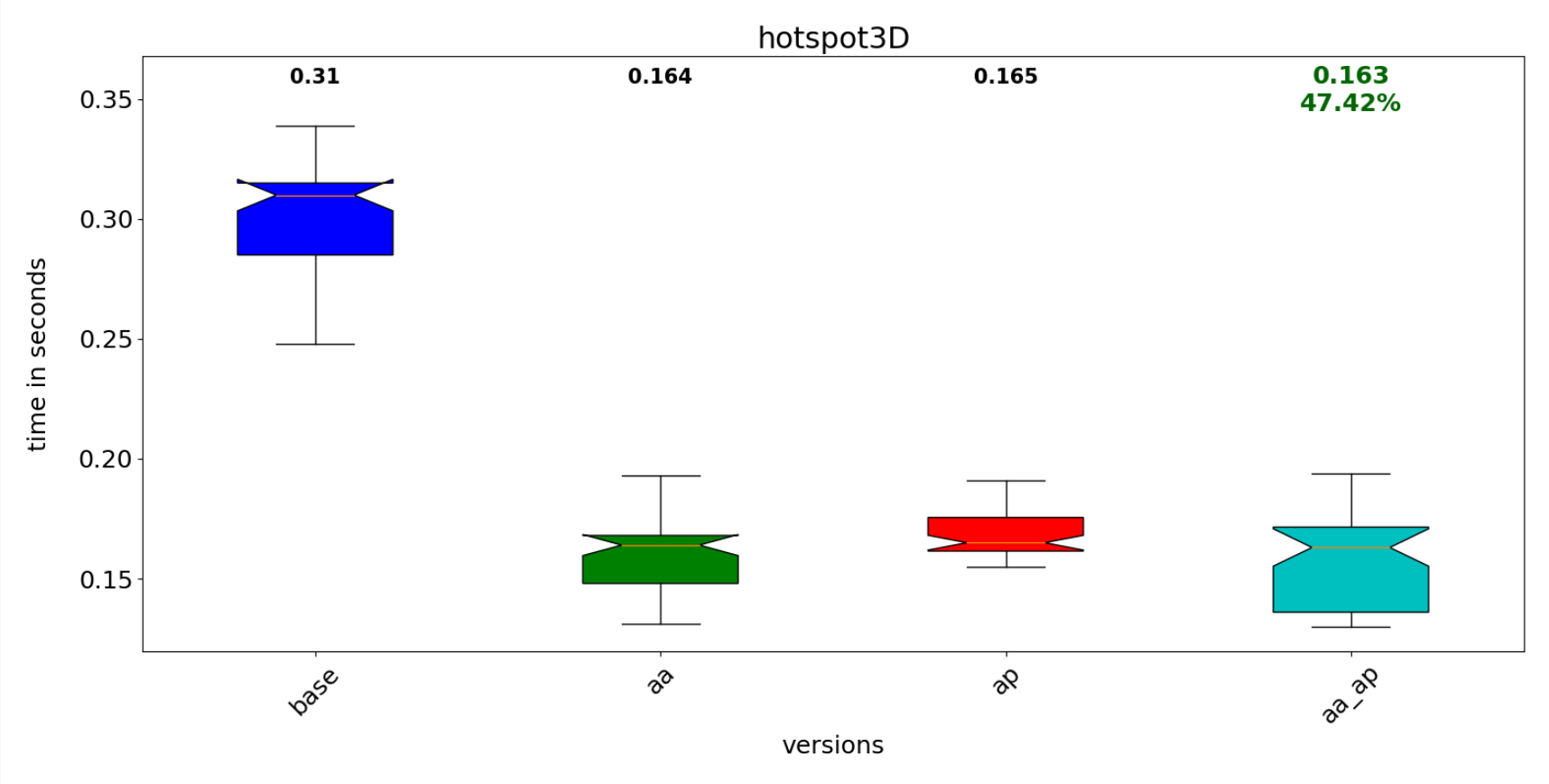
Example 1: Rodinia - hotspot3D

```
#pragma omp parallel
{
  int count = 0;
  float *tIn = In, *tOut = Out;
#pragma omp master
  printf("%d threads running \n", omp_get_num_threads ());
  do {
    int z;
#pragma omp for
    for (z = 0; z < nz; z++) {
      int y;
      for (y = 0; y < ny; y++) {
        int x;
        for (x = 0; x < nx; x++) {
          int c, w, e, n, s, b, t;
          c = x + y * nx + z * nx * ny;
          w = (x == 0) ? c : c - 1;
          e = (x == nx - 1) ? c : c + 1;
          n = (y == 0) ? c : c - nx;
          s = (y == ny - 1) ? c : c + nx;
          b = (z == 0) ? c : c - nx * ny;
          t = (z == nz - 1) ? c : c + nx * ny;
          tOut[c] = cc * tIn[c] + cw * tIn[w] + ce * tIn[e] +
                    cs * tIn[s] + cn * tIn[n] + cb * tIn[b] +
                    ct * tIn[t] + (dt/Cap) * pIn[c] + ct * a;
        }
      }
    }
    float *t = tIn, tIn = tOut;
    tOut = t;
  } while (++count < numiter);
}
```

Example 1:

Rodinia - hotspot3D

./3D 512 8 100 ../data/hotspot3D/power_512x8 ../data/hotspot3D/temp_512x8



Intel core i9, 10 cores, 20 threads, 51 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization

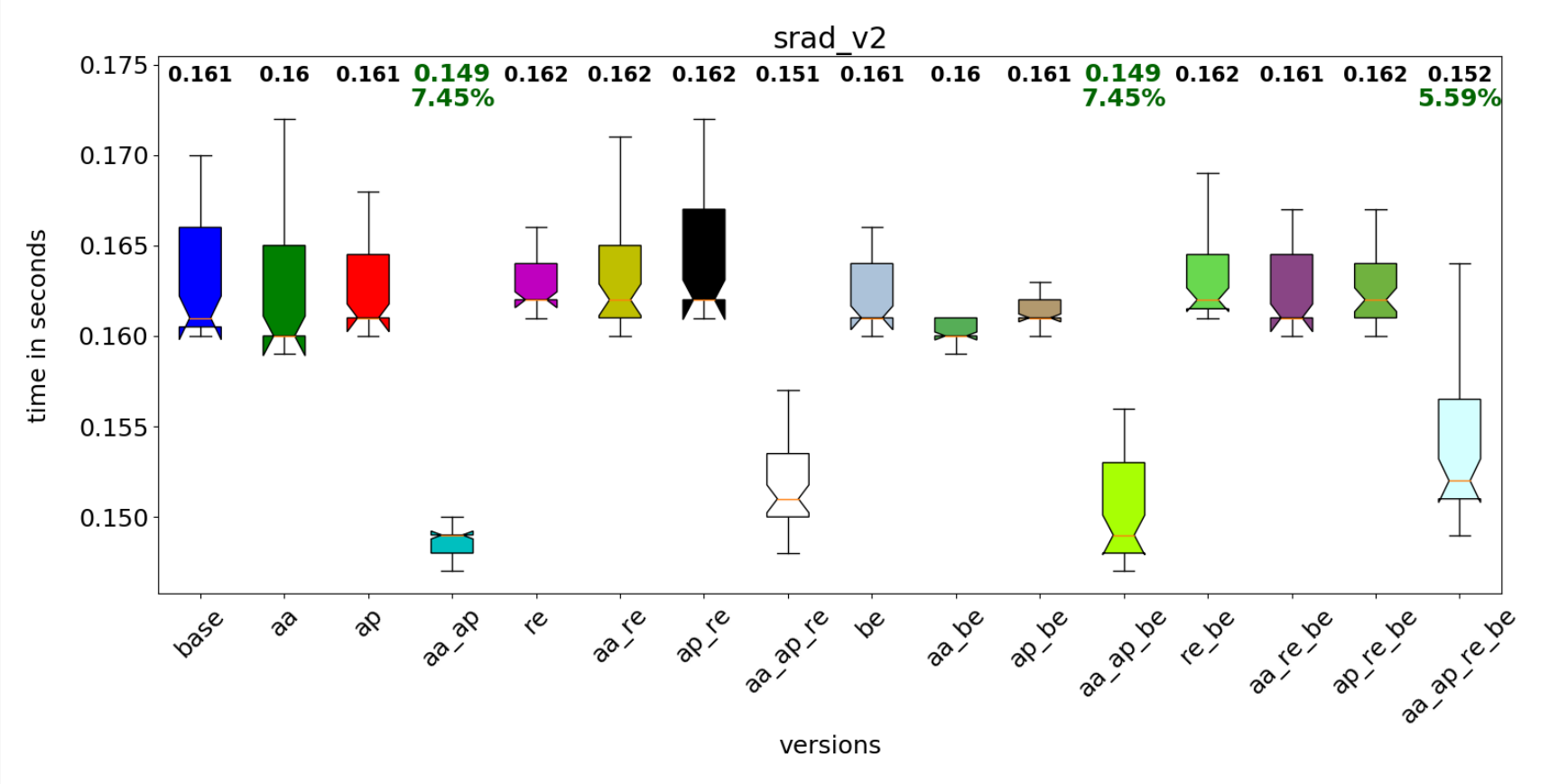
Example 2: Rodinia - srad_v2

```
#pragma omp parallel for shared(J, dN, dS, dW, dE, c, rows,    \
  cols, iN, iS, jW, jE) private(j, k, Jc, G2, L, num, den, qsqr)
for (int i = 0; i < rows; i++) {
  ...
}
#pragma omp parallel for shared(J, c, rows, cols, lambda)
                         private(i, j, k, D, cS, cN, cW, cE)
for (int i = 0; i < rows; i++) {
  ...
}
```

Example 2:

Rodinia - srad_v2

./srad 2048 2048 0 127 0 127 20 0.5 20



Intel core i9, 10 cores, 20 threads, 51 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization
- re => region expansion
- be => barrier elimination

Example 3: Rodinia - cfd

```
for (int i = 0; i < iterations; i++) {
  copy(old_vars, vars, nelr * NVAR);
  compute_step_factor(nelr, vars, areas, sf);
  for (int j = 0; j < RK; j++) {
    compute_flux(nelr, ese, normals, vars, fluxes, ff_vars,
                 ff_m_x, ff_m_y, ff_m_z, ff_dnergy);
    time_step(j, nelr, old_vars, vars, sf, fluxes);
  }
}
```

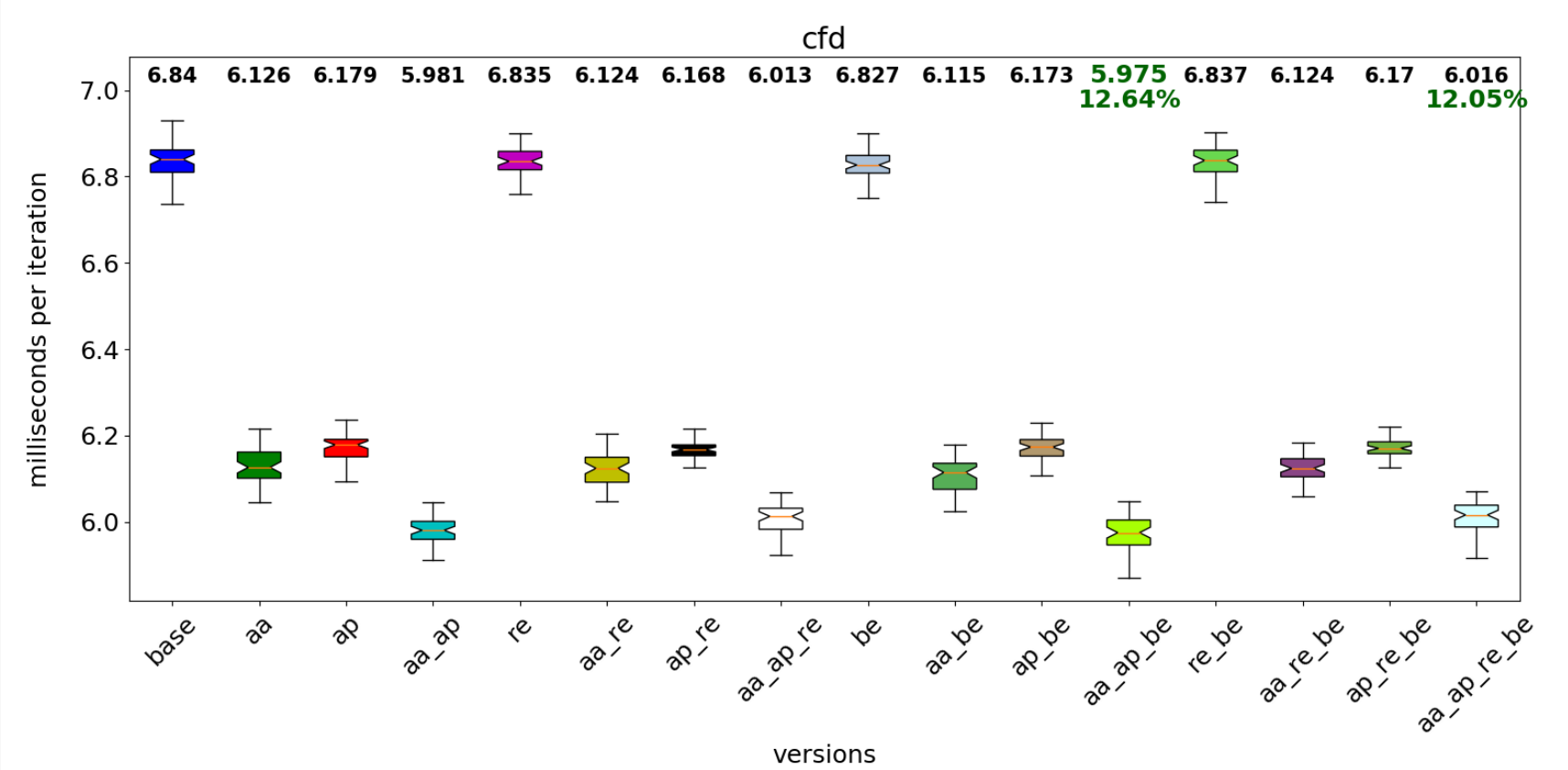
Example 3: Rodinia - cfd

```
#pragma omp parallel
for (int i = 0; i < iterations; i++) {
  #pragma omp for nowait                    // copy
  for (...) { /* write old_vars, read vars */ }
  #pragma omp for nowait                    // compute_step_factor
  for (...) { /* write sf, read vars & area */ }
  for (int j = 0; j < RK; j++) {
    #pragma omp for                         // compute_flux
    for (...) { /* write fluxes, read vars & ... */ }
  ...
```

Example 3:

Rodinia - cfd

cfd fvcorr.domn.193K



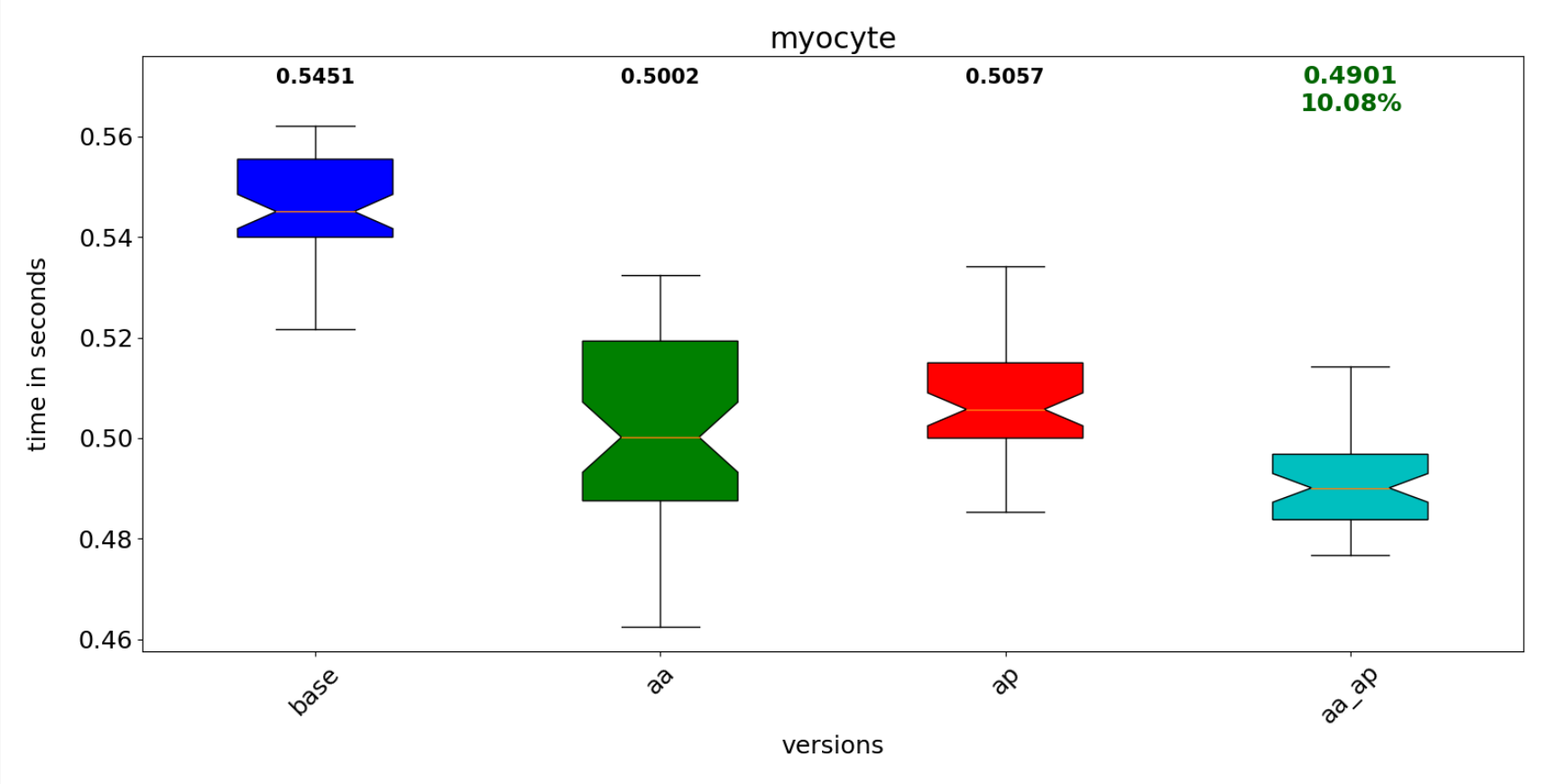
Intel core i9, 10 cores, 20 threads, 51 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization
- re => region expansion
- be => barrier elimination

Example 4:

Rodinia - myocyte

./myocyte 100 100 0 8



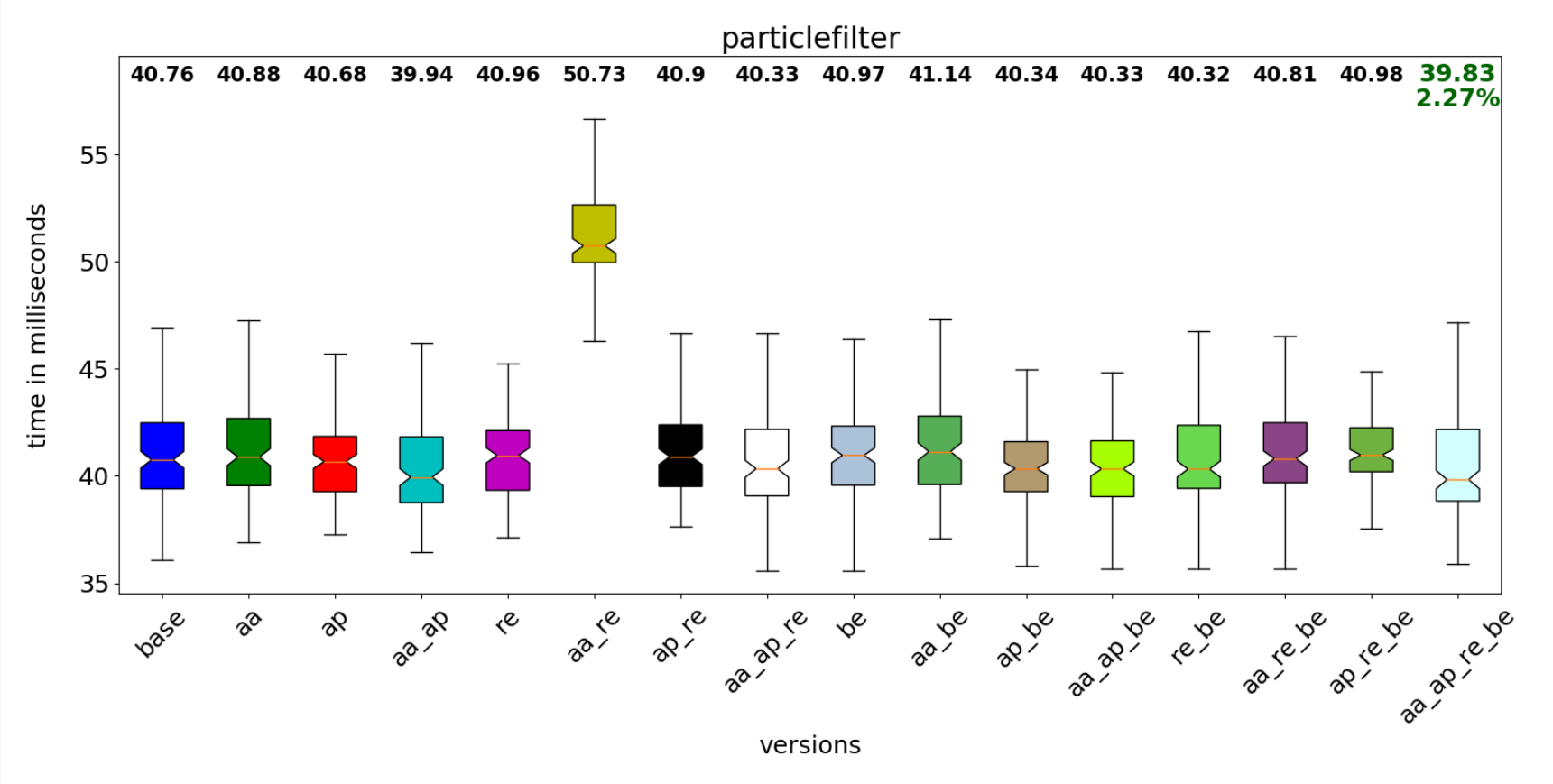
Intel core i9, 10 cores, 20 threads, 51 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization
- re => region expansion
- be => barrier elimination

Example 5:

Rodinia - particlefilter

./particlefilter -x 128 -y 128 -z 10 -np 10000

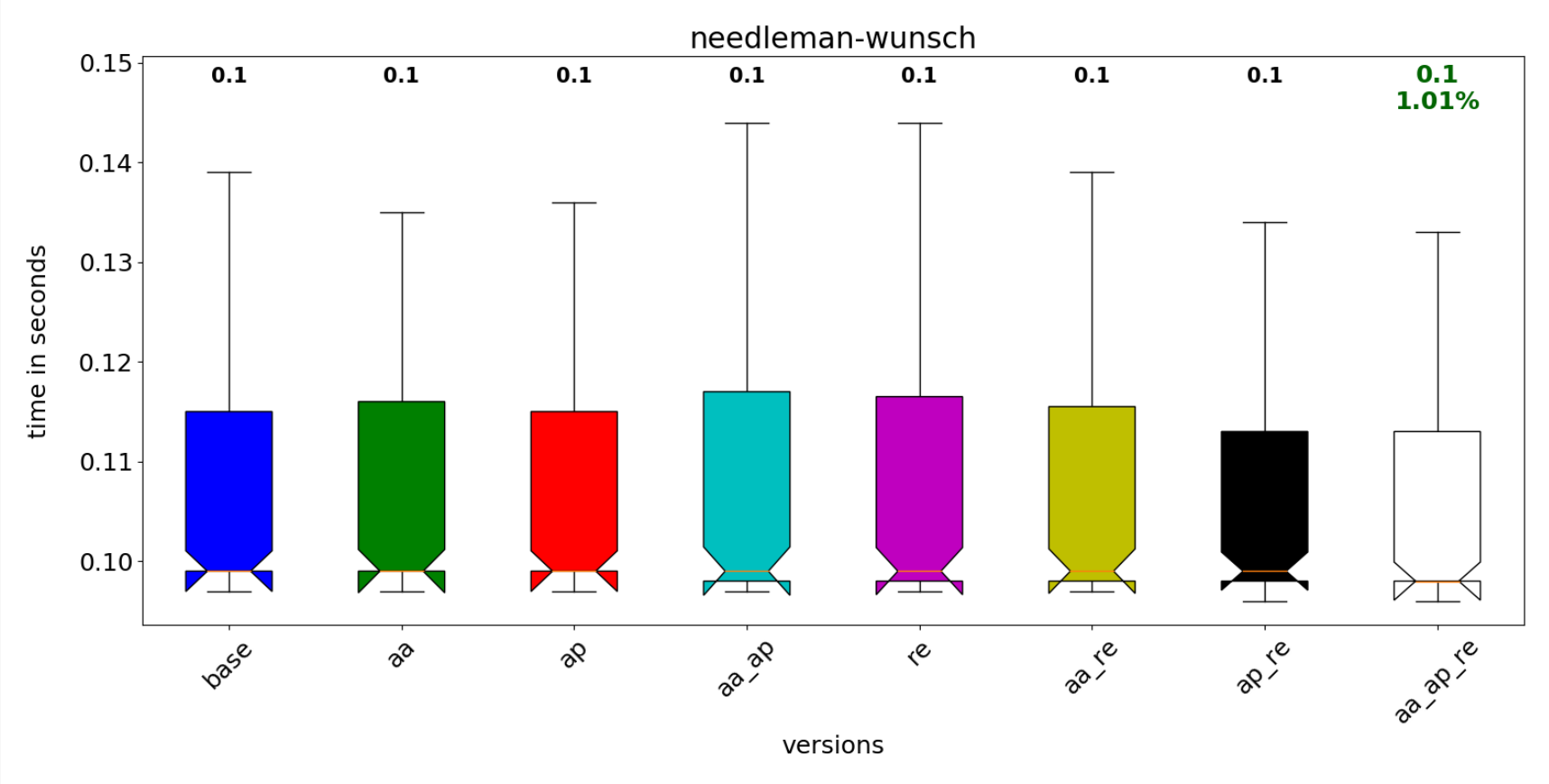


Intel core i9, 10 cores, 20 threads, 151 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization
- re => region expansion
- be => barrier elimination

Example 6: Rodinia - needleman-wunsch

./nw 8192 10 8



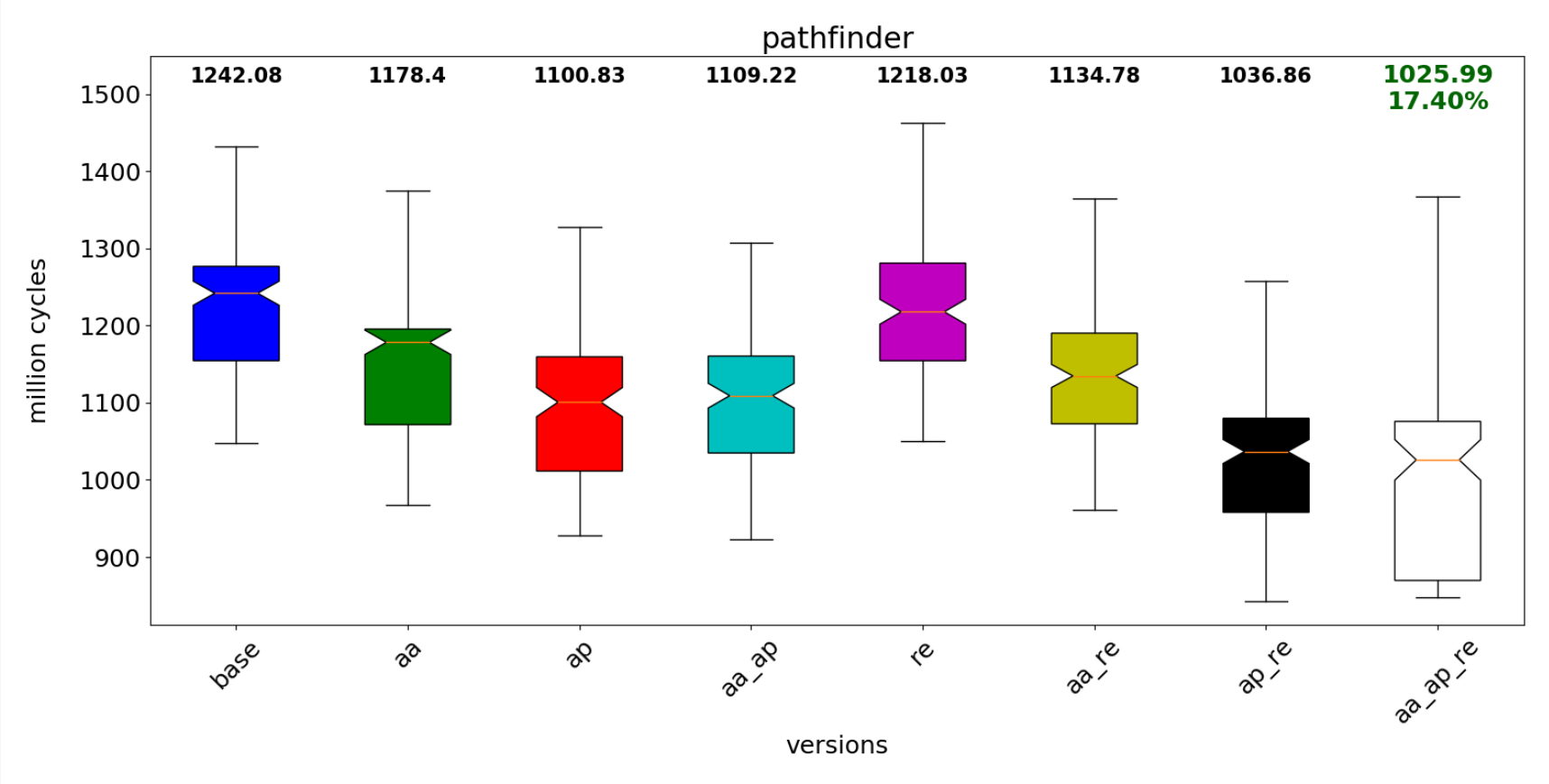
Intel core i9, 10 cores, 20 threads, 151 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization
- re => region expansion
- be => barrier elimination

Example 7:

Rodinia - pathfinder

./pathfinder 40000 40000



Intel core i9, 10 cores, 20 threads, 151 runs, with and without

- aa => alias attribute propagation
- ap => argument privatization
- re => region expansion
- be => barrier elimination

Acknowledgments

- The LLVM community (including our many contributing vendors)
- ALCF, ANL, and DOE
- ALCF is supported by DOE/SC under contract DE-AC02-06CH11357

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.