# An alternative OpenMP Backend for Polly

TECHNISCHE UNIVERSITÄT DARMSTADT

**Michael Halkenhäuser, Lukas Sommer**
Embedded Systems and Applications Group (ESA), Technische Universität Darmstadt, Germany
michael.halkenhaeuser@stud.tu-darmstadt.de, sommer@esa.tu-darmstadt.de

Embedded Systems & Applications

## Overview

- *Polly* supports OpenMP auto-parallelization.
- Single user option: *thread count*.
- Its current backend creates runtime scheduled loops only, using an environment var.
- Mandatory support of GNU's OpenMP lib.

**Hence, we want to present our extension:**

- Extra switches: *scheduling* & *chunk size*
- Utilizing LLVM's OpenMP library

```
// "matvect" -- Sequential
// Used as sample input for Polly.
// (Simplified dependencies.)


for (i = 0; i <= n; i++) {
  for (j = 0; j <= n; j++)
    s[i] = s[i] + a[i][j] * x[j];
}
```

**Figure 1:** `matvect` Kernel

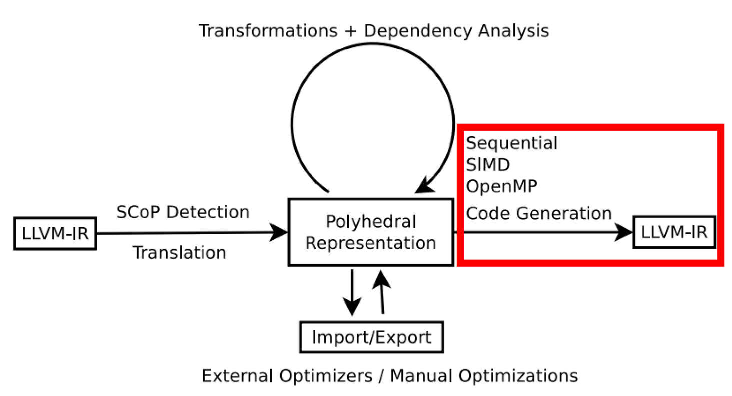


**Figure 2:** Polly workflow schematic [1].

```
// "matvect" -- OpenMP parallelized
// Equivalent to the LLVM-IR
// output of Polly's OpenMP backend.
#pragma omp parallel for [...] \
schedule (dynamic, 1) num_threads(N)
for (i = 0; i <= n; i++) {
  for (j = 0; j <= n; j++)
    s[i] = s[i] + a[i][j] * x[j];
}
```

**Figure 3:** `matvect` Kernel (OpenMP)

### *static* Scheduling

- Even distribution of work among OpenMP threads.
- Work-shares are predetermined.
- Minimal organization overhead.
- Well suited for problems, where each iteration takes similar amounts of time to complete.

### *dynamic* Scheduling

- Work-shares (of *chunk size*) are assigned dynamically.
- After completion of a chunk, the respective worker will request another chunk (if available).
- Potentially high organization overhead.
- Advantageous when load imbalances may occur.

### *guided* Scheduling

- Chunk size starts off large and decreases over time.
- Provided *chunk size* is actually a **minimum** *chunk size*.
- *Lower* organization overhead (than *dynamic*).
- Advantageous when load imbalances may occur and dynamic work distribution turns out to be an issue.

## Experimental results

### Chunk size

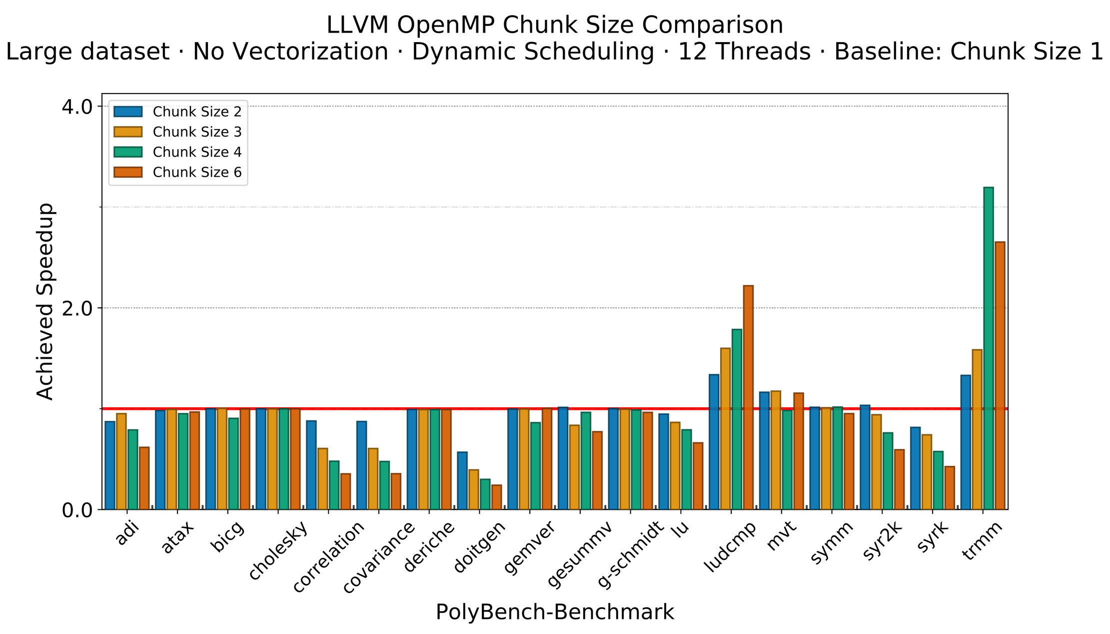


**Figure 4:** Impact of different chunk sizes.

- Variation of the chunk size may improve performance.
- But: An optimal value is problem-dependent.

### Scheduling type

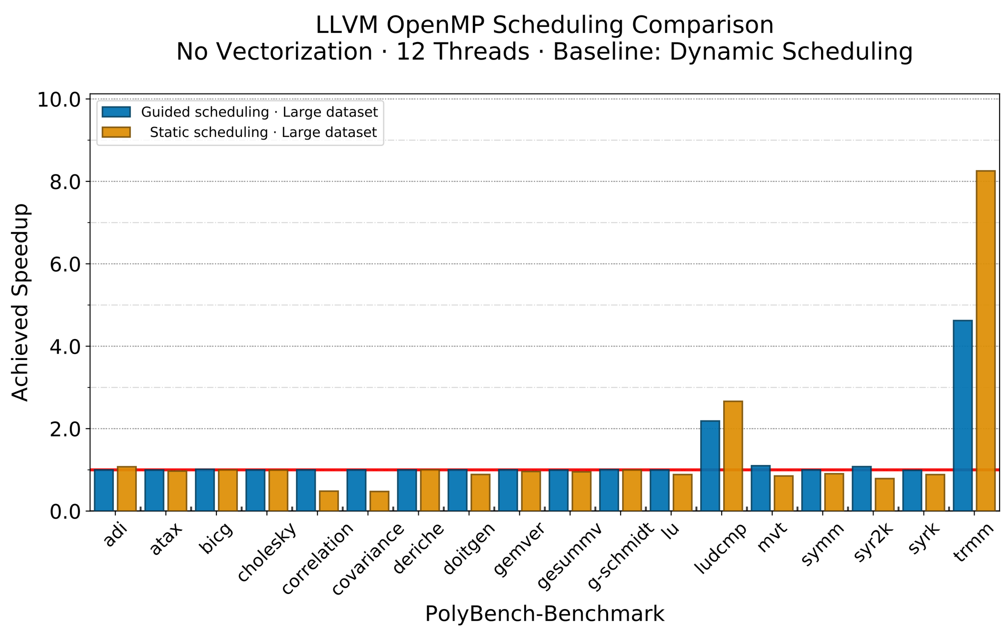


**Figure 5:** Impact of different scheduling types.

- Choosing an appropriate scheduling type may yield high speedups.

### Thread count

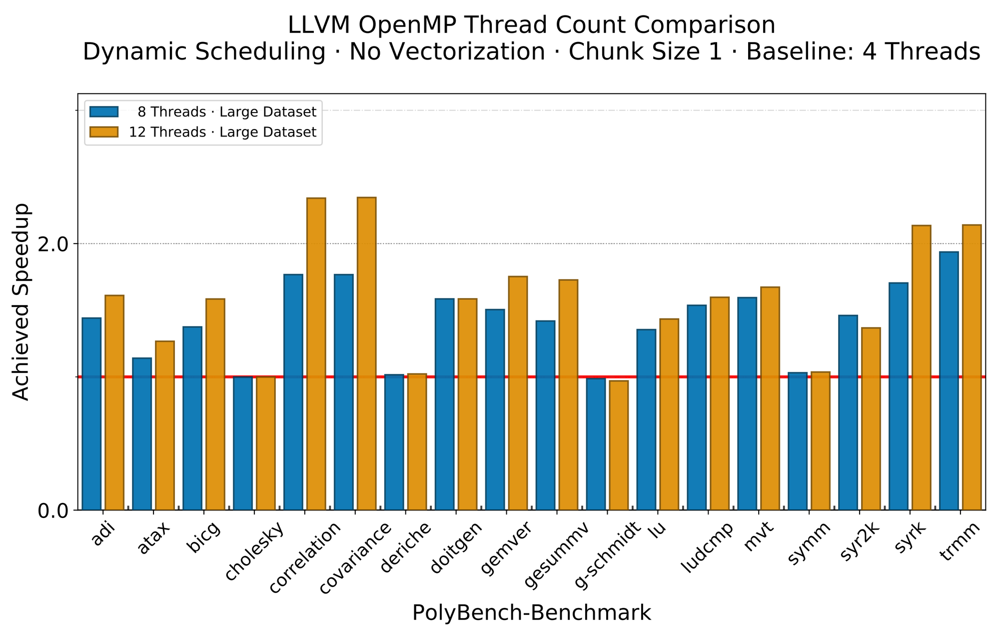


**Figure 6:** Evaluation of different thread counts.

- Higher thread counts may offer more processing power.
- However, the setup of threads has to be amortized.

### OpenMP library comparison

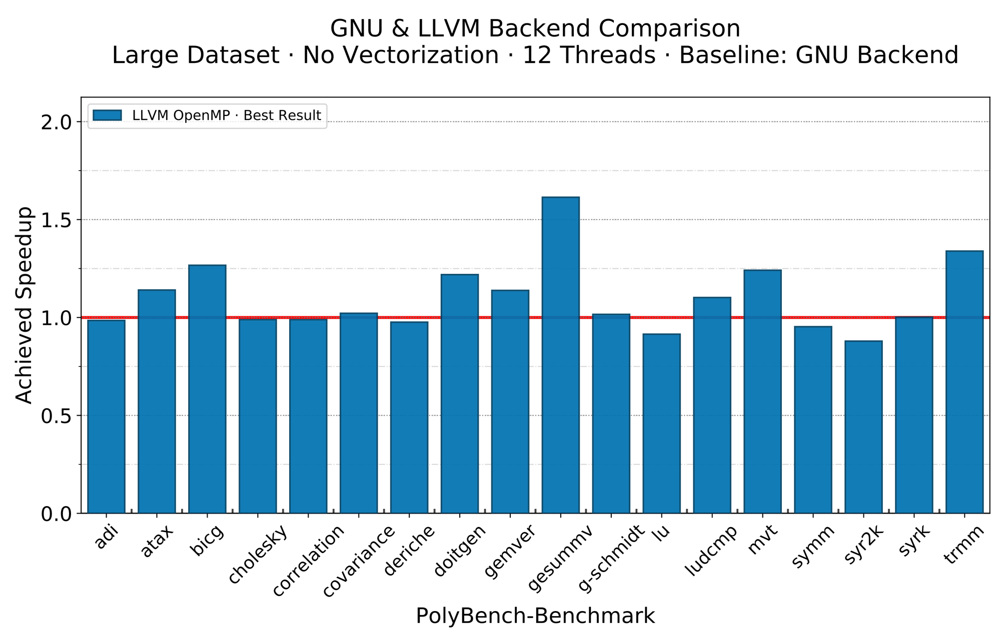


**Figure 7:** Comparison of both OpenMP backends.

- The LLVM OpenMP library achieves comparable results.
- Additionally, there are several cases where our backend achieves up to 1.6× relative speedup.

### General comparison

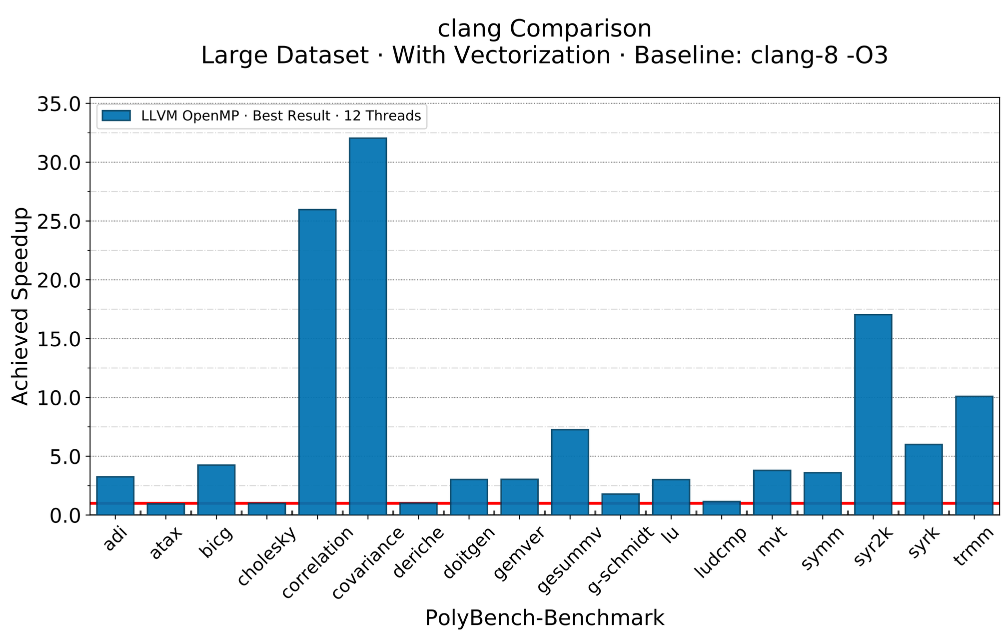


**Figure 8:** Clang versus our LLVM OpenMP backend.

- Large problem sizes benefit from thread level parallelism.
- Our alternative backend remains competitive overall.

## Conclusion

- Introduced switches carry no clear drawback:
  - Only in three considered cases, the GNU backend is able to achieve a lead of at least three percent.
- Not every benchmark will be parallelized w.r.t. the measured section (but the initialization).
- *Chunk size* offers problem-dependent customization but will decrease performance in unfavorable settings.
- *Scheduling* types may also be used to account for peculiarities of a computation and our results emphasize the advantage of a corresponding switch.
- In seven cases our backend gains significant speedups, when compared to the existing GNU-based backend.

## References

[1] T. Grosser, H. Zheng, R. Aloor, A. Simbürger, A. Größlinger, and L.-N. Pouchet, "Polly - polyhedral optimization in llvm," in *Proceedings of the First International Workshop on Polyhedral Compilation Techniques (IMPACT)*, vol. 2011, 2011, p. 1.

**Polly – LLVM OpenMP Backend Source**

The LLVM OpenMP backend extension was committed to Polly:
`https://github.com/llvm/llvm-project/commit/89251ed`