An alternative OpenMP Backend for Polly

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

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

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

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

Scheduling type

- Choosing an appropriate scheduling type may yield high speedups.

Thread count

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

OpenMP library comparison

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

General comparison

- 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