An alternative OpenMP Backend for Polly

Michael Halkenhäuser

2019 European LLVM Developers’ Meeting
Polly

- Polyhedral framework on LLVM-IR

- Efficient analyses and transformations

- Code generation

- Example transformations
  - Loop interchange / fission / fusion
  - Strip mining (Vectorization)
  - Automatic parallelization
Polly

Polyhedral framework on LLVM-IR
  Efficient analyses and transformations
  Code generation
Polly

- Polyhedral framework on LLVM-IR
  - Efficient analyses and transformations
  - Code generation

- Example transformations
  - Loop interchange / fission / fusion
  - Strip mining (Vectorization)
Polyhedral framework on LLVM-IR
- Efficient analyses and transformations
- Code generation

Example transformations
- Loop interchange / fission / fusion
- Strip mining (Vectorization)
- Automatic parallelization
Polly – Sample Parallelization

- Automatic parallelization
  - No need for manual annotation
Polly – Sample Parallelization

- Automatic parallelization
  - No need for manual annotation

```
// "matvect" -- Sequential
// (Simplified dependencies)

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

**Input**
Polly – Sample Parallelization

- Automatic parallelization
  - No need for manual annotation

---

// "matvect" -- Sequential
// (Simplified dependencies)

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

---

// "matvect" -- OpenMP parallelized
// Equivalent to the LLVM-IR output

```
#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];
}
```

---

Input

Output
Polly – Parallelization Scheme

- Polly detects parallelizable code regions
  - Moved into an *outlined function*
  - Executed using OpenMP API
Motivation for an alternative OpenMP Backend

- Limited influence on OpenMP execution
  - Increase number of user options
  - Improve fine-tuning possibilities
- Dependent on GNU OpenMP API
- Expand the scope of application
- LLVM OpenMP implementation available
- Enable direct use of LLVM's OpenMP runtime
- Support automated testing
Motivation for an alternative OpenMP Backend

- Limited influence on OpenMP execution
  - Increase number of user options
  - Improve fine-tuning possibilities

- Dependent on GNU OpenMP API
  - Expand the scope of application
Motivation for an alternative OpenMP Backend

- Limited influence on OpenMP execution
  - Increase number of user options
  - Improve fine-tuning possibilities

- Dependent on GNU OpenMP API
  - Expand the scope of application

- LLVM OpenMP implementation available
  - Enable direct use of LLVM’s OpenMP runtime
  - Support automated testing
LLVM OpenMP Backend

- Extension of the preexisting backend
  - Reused common functionalities
    - Moved into abstract base class
LLVM OpenMP Backend

- Extension of the preexisting backend
  - Reused common functionalities
    - Moved into abstract base class
  - API-specific call creation and placement
    - Implemented in derived class per backend

User may choose backend via CL switch, similar to number of threads and additional options.
LLVM OpenMP Backend

- Extension of the preexisting backend
  - Reused common functionalities
    - Moved into abstract base class
  - API-specific call creation and placement
    - Implemented in derived class per backend

- User may choose backend
  - Via CL switch, similar to
    - Number of threads
LLVM OpenMP Backend

- Extension of the preexisting backend
  - Reused common functionalities
    - Moved into abstract base class
  - API-specific call creation and placement
    - Implemented in derived class per backend

- User may **choose backend**
  - Via CL switch, similar to
    - *Number of threads*
  - Additional options
    - *Scheduling type*
    - *Chunk size*
LLVM OpenMP Backend – Options

▶ *Scheduling type* determines work distribution

<table>
<thead>
<tr>
<th>static</th>
<th>dynamic</th>
<th>guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predetermined, uniform distribution of iterations</td>
<td>Threads request work shares of <em>chunk size</em></td>
<td>Hybrid scheduling of <em>static</em> and <em>dynamic</em>, using a minimum <em>chunk size</em></td>
</tr>
</tbody>
</table>
LLVM OpenMP Backend – Options

- **Scheduling type** determines work distribution

<table>
<thead>
<tr>
<th></th>
<th>static</th>
<th>dynamic</th>
<th>guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Balancing</td>
<td>−</td>
<td>+</td>
<td>○</td>
</tr>
<tr>
<td>Organization Overhead</td>
<td>+</td>
<td>−</td>
<td>○</td>
</tr>
</tbody>
</table>
Scheduling type determines work distribution

<table>
<thead>
<tr>
<th></th>
<th>static</th>
<th>dynamic</th>
<th>guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Balancing</td>
<td>−</td>
<td>+</td>
<td>○</td>
</tr>
<tr>
<td>Organization Overhead</td>
<td>+</td>
<td>−</td>
<td>○</td>
</tr>
</tbody>
</table>

- **static** suited for constant computational demands
- **dynamic** suited for shifting computational demands
- **guided** suited for "both"
Experimental Methodology

PolyBench\(^1\)
- Provides multiple datasets
- Triggers auto-parallelization in 18 benchmarks

\(^1\)https://sourceforge.net/projects/polybench/
Experimental Methodology

- PolyBench\(^1\)
  - Provides multiple datasets
  - Triggers auto-parallelization in 18 benchmarks

- Runtime results
  - Average from 50 out of 60 runs (10% trimmed-mean)
  - Utilized CPU: AMD R5 1600X

---

\(^1\)https://sourceforge.net/projects/polybench/
Experimental Methodology

- PolyBench\(^1\)
  - Provides multiple datasets
  - Triggers auto-parallelization in 18 benchmarks

- Runtime results
  - Average from 50 out of 60 runs (10\% trimmed-mean)
  - Utilized CPU: AMD R5 1600X

- Plots show *relative speedup*
  - \[ \text{speedup} = \frac{\text{runtime of baseline}}{\text{runtime of competitor}} \]

---

\(^1\)https://sourceforge.net/projects/polybench/
Performance Impact of **chunk size**

LLVM OpenMP Chunk Size Comparison

- Large Dataset · No Vectorization · Dynamic Scheduling · 12 Threads · Baseline: Chunk Size 1
Performance Impact of **scheduling type**

LLVM OpenMP Scheduling Comparison
No Vectorization · 12 Threads · Baseline: Dynamic Scheduling

- Guided Scheduling · Large Dataset
- Static Scheduling · Large Dataset

![Graph showing speedup comparison between different scheduling types for PolyBench benchmarks.](image-url)
Intermezzo – Customization Options

▶ **Chunk size**
  ▶ 1 is usually a reasonable choice
  ▶ Very beneficial in particular cases
    ▶ More than 3× speedup possible
Intermezzo – Customization Options

▶ **Chunk size**
  ▶ 1 is usually a reasonable choice
  ▶ Very beneficial in particular cases
    ▶ More than 3× speedup possible

▶ **Scheduling type**
  ▶ *Dynamic*: Good overall performance
  ▶ *Guided*: Performs at least as good as *dynamic*
  ▶ *Static*: Problem-dependent
    ▶ May achieve 8× speedup compared to *dynamic*
Backend Comparison
LLVM versus GNU OpenMP Backend

GNU & LLVM Backend Comparison
Large Dataset · No Vectorization · 4 Threads · Baseline: GNU Backend

Achieved Speedup

PolyBench-Benchmark

2019-04-08 | Embedded Systems and Applications Group | Michael Halkenhäuser | 13 / 19
Backend Comparison
LLVM versus GNU OpenMP Backend

GNU & LLVM Backend Comparison
Large Dataset · No Vectorization · 12 Threads · Baseline: GNU Backend

Achieved Speedup

PolyBench-Benchmark
Using the maximum number of available threads

Our "LLVM" backend
- Achieves comparable performance
- Performs significantly faster than "GNU" in seven cases
- Reaches up to $1.6 \times$ speedup
Using the maximum number of available threads

Our "LLVM" backend
  - Achieves comparable performance
  - Performs significantly faster than "GNU" in seven cases
  - Reaches up to $1.6 \times$ speedup

GNU backend
  - Only a single, considerable lead
Using the maximum number of available threads

Our "LLVM" backend
- Achieves comparable performance
- Performs significantly faster than "GNU" in seven cases
- Reaches up to $1.6 \times$ speedup

GNU backend
- Only a single, considerable lead

Additional switches
- Allow problem-specific adjustments
- ... without depending on env. variable
General Comparison

LLVM OpenMP Backend versus clang

clang Comparison

Large Dataset · With Vectorization · Baseline: clang-8 -O3

Achieved Speedup

PolyBench-Benchmark
Conclusion

- Our "LLVM" OpenMP backend for Polly
  - Represents a superior alternative
Conclusion

- Our "LLVM" OpenMP backend for Polly
  - Represents a superior alternative
  - Acts as drop-in replacement
Conclusion

▶ Our "LLVM" OpenMP backend for Polly
  ▶ Represents a superior alternative
  ▶ Acts as drop-in replacement
  ▶ Provides more customization options
Conclusion

- Our "LLVM" OpenMP backend for Polly
  - Represents a superior alternative
  - Acts as drop-in replacement
  - Provides more customization options
  - Carries no clear drawbacks, but instead ...
    - Reaches up to $1.6 \times$ speedup
Conclusion

▶ Our "LLVM" OpenMP backend for Polly
  ▶ Is publicly available
    ▶ Review accepted on March 19th
      https://reviews.llvm.org/D59100
    ▶ Currently on Polly's master branch
      https://github.com/llvm/llvm-project/commit/89251ed

▶ References:
  ▶ Title graphic: https://polly.llvm.org/images/header-background.png
  ▶ 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
Questions?

▶ Ask them now, or ...

▶ Find me tomorrow, at the poster session
  ▶ 09:00 am - 10:00 am (Foyer)