YaskSite: Stencil Optimization Techniques Applied to Explicit ODE Methods on Modern Architectures

Christie Alappat, Johannes Seiferth, Georg Hager, Matthias Korch, Thomas Rauber, Gerhard Wellein

International Symposium on Code Generation and Optimization (CGO 2021)
Introduction

System of ordinary differential equations

Parallel Iterated Runge-Kutta (PIRK) method

Implementation?

Solution

Which implementation performs best?

More optimizations? (Tiling, vector folding, …)

YaskSite

/* Implementation F */
// BARRIER
for (int k=0; k<m; ++k) {
    RHS
    // BARRIER
    LC
    // BARRIER
}
// BARRIER
App
// BARRIER
Up

/* Kernel RHS_lj */
for (int j*=0; j*<N; ++j*)
    for (int i=0; i < s; ++s)
        F[i][jx][jy][jz] = %RHS

/* Kernel RHS_ij */
for (int k=0; k<m; ++k) {
    /* Kernel RHS_ij */
    for (int j*=0; j< N; ++j*)
        for (int i=0; i < s; ++s)
            F[i][jx][jy][jz] = %RHS

YaskSite
YASK & YaskSite
YASK - Yet Another Stencil Kit

- Framework to create **high-performance** stencil code.

```
for(int t=1; t<g_t; ++t)
    #pragma omp parallel for collapse(3) schedule(static,1)
    for(int begin_b•=0; begin_b•<g•; begin_b•+=b•)
        for(int•= begin_b•;•< begin_b•+b•;•+=•+ 1)
            out[t+1,x,y,z] = STENCIL(in[t, x, y, z])
```

Spatial tiling

Vector folding

Temporal tiling

Lots of optimizations

Spatial tiling

Vector folding

Temporal tiling

for(int t=1; t<g_t; ++t)
    #pragma omp parallel for collapse(3) schedule(static,1)
    for(int begin_b•=0; begin_b•<g•; begin_b•+=b•)
        for(int•= begin_b•;•< begin_b•+b•;•+=•+ 1)
            out[t+1,x,y,z] = STENCIL(in[t, x, y, z])
YASK - Yet Another Stencil Kit

- Framework to create **high-performance** stencil code.

**Spatial tiling**

**Vector folding**

**Temporal tiling**

Lots of optimizations

**Focus of the paper**

Optimal tile size? Optimal vector folding?

```c
for(int t=1; t<g_t; ++t)
    #pragma omp parallel for collapse(3) schedule(static,1)
    for(int begin_b=0; begin_b<g; begin_b+=b)
        for(int x= begin_b; x< begin_b+b; x+=1)
            out[t+1,x,y,z] = STENCIL(in[t, x, y, z])
```
GRID_ARRAY y_new, y_old, dy;
PARAM h;

y_new = y_old + h * dy;
YASKSITE

- High level code
- Code generation
- YASK
- Code caching
- JIT
- YaskSite
- Stencil properties
- Machine file
- Tuning input
- Analytical spatial tuning
- Analytical vector folding
- Grid coupling
- Data layout change between stencils
- Grid coupling
Performance modeling
Motivation: Why modeling?

Performance of RHS_LC kernel on 1 socket (32 cores) of AMD ROME

Only 25% improvement from spatial tuning?
Analytical performance model

Execution-Cache-Memory* (ECM) performance model

* Stengel et al., 2015. Quantifying Performance Bottlenecks of Stencil Computations Using the Execution-Cache-Memory Model. https://doi.org/10.1145/2751205.2751240
Analytical model

Execution-Cache-Memory (ECM) performance model

2 major components that influence performance:

1) In-core

Static code analysis tools: IACA and OSACA
Analytical model

Execution-Cache-Memory (ECM) performance model

2 major components that influence performance:

1) In-core

2) Data transfer through memory hierarchy
Analytical model

Execution-Cache-Memory (ECM) performance model

2 major components that influence performance:

1) In-core

2) Data transfer through memory hierarchy

\[ T_{L1} \]

\[ T_{L2} \]

\[ T_{L3} \]

\[ T_{MEM} \]

Amount of data

Rate of transfer
Analytical model

Execution-Cache-Memory (ECM) performance model

2 major components that influence performance:

1) In-core

2) Data transfer through memory hierarchy

Amount of data
Rate of transfer

Depends on application
Depends on hardware

Registers

L1

L2

L3

MEM

$T_{L1}$

$T_{L2}$

$T_{L3}$

$T_{MEM}$

Depends on hardware

Depends on application

Rate of transfer

Amount of data

International Symposium on Code Generation and Optimization (CGO 2021), February 27th - March 3rd, 2021
Analytical model

Amount of data: YASK application

Determine working-set size → where data resides

\[
\begin{align*}
\text{for} (i = 0; i < n; ++i) & \quad // \quad n \times n = N \\
\text{for} (j = 0; j < n; ++j) & \\
b[i][j] &= a[i-1][j] + a[i][j] + a[i+1][j]
\end{align*}
\]

\[2^{\cdot}8 \cdot N \text{ [B]}\]

Calculate minimum data traffic

\[8 + 16 \text{ [B/it]}\]

Account for stencil reuse

Layer condition analysis

Assume fits in L3

Assume L1 cannot reuse
Analytical model \(\rightarrow\) Tuning

Layer condition analysis*

3D star stencil example with radius \(r\)

\[(N_s(2r + 1) + (N_r - N_s)) \times b_z b_y d < CS\]

\(\text{If satisfied, reuse in } x \text{ (outermost) dimension.}\)

Tunable parameters

Analytical spatial tuning:
Solve for these parameters \((b_z, b_y)\) to satisfy a certain cache (input) subject to constraints.

\[T = \max(T_{ol}, T_{L1} + T_{L2} + T_{L3} + T_{MEM})\]

Test bed

Intel Xeon Gold 6248 (SNC-off mode)

AMD EPYC 7452 – ROME (NPS1 mode)
Intel CLX

Tuning time in seconds

Wave3D (r=1)

Wave3D (r=2)

Wave3D (r=4)
Analytical tuning

**AMD ROME**

**Tuning time in seconds**

<table>
<thead>
<tr>
<th>GLUP/s</th>
<th>Wave3D (r=1)</th>
<th>Wave3D (r=2)</th>
<th>Wave3D (r=4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
<tr>
<td>analytical</td>
<td>38.45</td>
<td>40.78</td>
<td>46.00</td>
</tr>
<tr>
<td>GD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vector folding reduces traffic between L1 and registers. The folding dimension is a parameter for vector folding.

- Change in data layout
- Change in layer condition analysis

Incorporate into performance model for analytical performance prediction and tuning
Yet another benefit of modeling

Performance of RHS_LC kernel on 1 socket (32 cores) of AMD ROME

- ECM model

![Graph showing performance comparison between plain and spatial tuning]
Yet another benefit of modeling

Performance of RHS_LC kernel on 1 socket (32 cores) of AMD ROME
Offsite: PIRK ranking
- Tuning framework for explicit ODE methods
- YaskSite integrated as alternative ECM backend
Use-case: Tuning of PIRK variants

Predicted and measured performance of different PIRK variants for Wave3D (r=2) on AMD ROME

- **Plain variants**
  - A, 1:1:4
  - A, 2:2:1
  - F, 1:1:4
  - F, 2:2:1
  - Prediction

- **Spatial variants**
  - A, 1:1:4
  - A, 2:2:1
  - F, 1:1:4
  - F, 2:2:1
  - Prediction
Use-case: Tuning of PIRK variants

Predicted and measured performance of different PIRK variants for Wave3D (r=2) on AMD ROME

![Graph showing performance of Plain variants](image)

**Plain variants**

![Graph showing performance of Spatial variants](image)

**Spatial variants**

Offsite’s ranking quality when using YaskSite as backend considering 16 different PIRK variants

<table>
<thead>
<tr>
<th>Mean deviation (%)</th>
<th>Maximum perf. loss</th>
<th>Mean perf. loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0 %</td>
<td>21.1 %</td>
<td>4.4 %</td>
</tr>
</tbody>
</table>

\[
\text{Offsite’s ranking quality} = \frac{\text{best-selected}}{\text{selected}} \times 100
\]
Conclusion

- YaskSite combines YASK with the analytical ECM model to successfully predict and tune the performance of stencil codes
  - ECM model’s layer condition analysis was extended to include vector folding and victim caches
  - First time application of the ECM model to AMD Rome
- Demonstrated YakSite’s usefulness to detect bottlenecks and guide performance optimization
- YaskSite can be integrated with external tools to tune more complex applications.
Thank you for your interest!

For more information please contact:
Christie Alappat Christie.alappat@fau.de, Johannes Seiferth Johannes.seiferth@uni-bayreuth.de

This work is supported by the German Ministry of Science and Education (BMBF) under the project SeASiTe.

https://github.com/seasite-project

This presentation and recording belong to the authors. No distribution is allowed without the authors' permission.