Automating Compiler-Directed Autotuning for Phased Performance Behavior

Tharindu Rusira†, Mary Hall†, Protonu Basu*

†School of Computing, University of Utah
* Lawrence Berkeley National Laboratory
Outline

• GMG and phased performance behavior
• Domain-specific optimizations
• Superscripts
• Autotuning
• Results
• Future work
GMG

• V-cycle – Hierarchical linear solver with varying grid resolutions

![Diagram of V-cycle](image)

GMG and stencils

• miniGMG\textsuperscript{[1]}, Geometric Multigrid benchmark

• Domain decomposition

• Higher order stencils (13pt, 27pt, 125pt)

Motivation

• Each GMG level has different performance requirements

• Compiler has to decide which optimizations to apply

• Many performance factors
  • Architecture
  • GMG level (input size)
  • GMG operator (computation)
  • Stencil
  • Interaction between operators
Smooth operator (Jacobi)

- Performance dominant operation in V-cycle (multiple applications of ‘smooth’ at each level, usually communication-bound)

- $S_0$, $S_1$, $S_2$ involve stencil computations and memory updates of two grids temp and phi

```c
// statement S0
temp[k][j][i] = b * h2inv * (beta_1[k][j][i+1] * phi[k][j][i+1] - phi[k][j][i])
-beta_1[k][j][i] * (phi[k][j][i] - phi[k][j][i-1])
+beta_1[k][j][i] * (phi[k][j+1][i] - phi[k][j][i])
-beta_1[k][j][i] * (phi[k][j][i] - phi[k][j-1][i])
+beta_1[k+1][j][i] * (phi[k+1][j][i] - phi[k][j][i])
-temp[k][j][i] = a * alpha[k][j][i] * phi[k][j][i] * temp[k][j][i];
```

```c
// statement S2
phi[k][j][i] = phi[k][j][i] - lambda[k][j][i] * (temp[k][j][i] - rhs[k][j][i]);
```
Optimizations

• Loop fusion

• Wavefront computation
  • With deep ghost zones (halo regions)

• Partial sums

• Nested parallelism (OpenMP)
Loop fusion

Reduces vertical (across memory hierarchy) memory traffic

```c
if (smooth_sweep % 2 == 0) {
  for (k=0;k<N;k++)
    for (j=0;j<N;j++)
      for (i=0;i<N;i++)
        even_S0(); // Laplace
  for (k=0;k<N;k++)
    for (j=0;j<N;j++)
      for (i=0;i<N;i++)
        even_S1(); // Hemholtz
  for (k=0;k<N;k++)
    for (j=0;j<N;j++)
      for (i=0;i<N;i++)
        even_S2(); // Jacobi relaxation
} else {
  for (k=0;k<N;k++)
    for (j=0;j<N;j++)
      for (i=0;i<N;i++)
        odd_S0();
  for (k=0;k<N;k++)
    for (j=0;j<N;j++)
      for (i=0;i<N;i++)
        odd_S1();
  for (k=0;k<N;k++)
    for (j=0;j<N;j++)
      for (i=0;i<N;i++)
        odd_S2();
}
```
Wavefronts

Reduces vertical memory traffic by simultaneously running multiple smooths

Partial sums

2D 9pt stencil

```
for (j=0; j<N; j++)
    for (i=0; i<N; i++)
        out[j][i] = w1*(
            in[j-1][i-1] + in[j-1][i] + in[j-1][i+1] +
            in[j  ][i-1] + in[j  ][i+1] +
            in[j+1][i-1] + in[j+1][i] + in[j+1][i+1]
        ) +
        w2*(
            in[j-1][i-1] + in[j-1][i-1] + in[j-1][i+1] +
            in[j  ][i-1] + in[j  ][i+1] +
            in[j+1][i-1] + in[j+1][i] + in[j+1][i+1]
        ) +
        w3*(in[j  ][i  ]);
```
Nested Parallelism (Collaborative Threading)
Code variants and Search space

• Search space for 125pt stencil

  Wavefront size = \{2,4,8\}
  Partial sums = \{ON, OFF\}
  Collaborative threading = \{1x12, 2x6, 3x4, 4x3, 6x2, 12x1\}
  Smooth variants for a level (\(>4^3\)) = 3 \times 2 \times 6 = 36
  Smooth variants for bottom layer (4^3) = 2 \times 2 \times 6 = 24
  Total smooth variants for the V-cycle = 36^4 \times 24 = 40, 310, 784

• Question! How do we explore the search space?
CHiLL loop transformations

```
original()
skew ([0,1,2,3,4,5], 2, [3,1])
permute ([2,1,3,4])

partial_sums(0)
partial_sums(5)

fuse ([0,1,2,3,4,5,6,7,8,9], 4)

omp_par_for(4,3)
```


CHiLL superscripts

```python
original()
skew ([0,1,2,3,4,5], 2, [3,1])
permute ([2,1,3,4])

partial_sums(0)
partial_sums(5)

fuse ([0,1,2,3,4,5,6,7,8,9], 4)
omp_par_for(4,3)
```

```python
original()
skew ([0,1,2,3,4,5], 2, [3,1])
permute ([2,1,3,4])

partial_sums(0)
partial_sums(5)

fuse ([0,1,2,3,4,5,6,7,8,9], 4)

@begin_param_region
param(x, enum, [1,2,3,4,6,12])
param(y, enum, [1,2,3,4,6,12])
constraint (x*y==12)
omp_par_for(x,y)
@end_param_region
```

import chillsuper as cs
cs.generate_parameter_domain('superscript')

https://github.com/TharinduRusira/CHiLLsuper
Autotuning

```python
wav = ['wav0', 'wav1', 'wav2', 'wav3', 'wav4']
collab = ['c0', 'c1', 'c2', 'c3', 'c4']
ps = ['ps0', 'ps1', 'ps2', 'ps3', 'ps4']
def manipulator(self):
    manipulator = ConfigurationManipulator()
    for x in collab:
        manipulator.add_parameter(EnumParameter(x, ['1x12', '2x6', '3x4', '4x3', '6x2', '12x1']))
    for y in wav:
        if y is 'wav4':
            manipulator.add_parameter(EnumParameter(y, [2, 4]))
        else:
            manipulator.add_parameter(EnumParameter(y, [2, 4, 8]))
    for z in ps:
        manipulator.add_parameter(EnumParameter(z, ['on', 'off']))
    return manipulator
```
Code variant evaluation

• Total ‘smooth’ time

\[ f_{cost} = \sum_{i=0}^{\text{levels}} t_s^i \]

• Customizable cost function
Tuned configurations

<table>
<thead>
<tr>
<th>TABLE I: Tuned configuration for 13pt stencil, # of tests=1454</th>
</tr>
</thead>
<tbody>
<tr>
<td>box size</td>
</tr>
<tr>
<td>fused</td>
</tr>
<tr>
<td>wavefront</td>
</tr>
<tr>
<td>partial sums</td>
</tr>
<tr>
<td>collab. threading</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II: Tuned configuration for 27pt stencil, # of tests=1518</th>
</tr>
</thead>
<tbody>
<tr>
<td>box size</td>
</tr>
<tr>
<td>fused</td>
</tr>
<tr>
<td>wavefront</td>
</tr>
<tr>
<td>partial sums</td>
</tr>
<tr>
<td>collab. threading</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III: Tuned configuration for 125pt stencil, # of tests=1078</th>
</tr>
</thead>
<tbody>
<tr>
<td>box size</td>
</tr>
<tr>
<td>fused</td>
</tr>
<tr>
<td>wavefront</td>
</tr>
<tr>
<td>partial sums</td>
</tr>
<tr>
<td>collab. threading</td>
</tr>
</tbody>
</table>
Results

NERSC’s Edison (8 MPI tasks, 4 nodes), ICC with -O3 -xAVX --D__COLLABORATIVE_THREADING=12 --D__MPI
GMG configuration: $512^3$ full domain distributed among 8 tasks ($256^3$ subdomain each). 5 GMG levels with $64^3$ at the top level ($64^3,32^3,16^3,8^3,4^3$).
Effect of autotuning on phased performance

Comparison of speedups by each GMG level \((64^3, 32^3, 16^3, 8^3, 4^3)\) and overall speedup of the tuned configuration against corresponding V-cycle levels in default configurations.
Related Work


Future work

• Leverage support for arbitrary HPC applications

• Loop transformation sequences

• Better search space pruning with domain-specific information

• Guided search for fast convergence using analytical models/domain-specific heuristics