Boosting Performance Optimization with Interactive Data Movement Visualization

Philipp Schaad*, Tal Ben-Nun*, Torsten Hoefler*

* Scalable Parallel Computing Laboratory, ETH Zurich
The Cost of Data Movement

Exploit *spatial locality* and *temporal locality*!
Data Movement Optimization

1. Performance Analysis
2. Reduce Data Movement
Data Movement Optimization
Data Movement Optimization

Performance Analysis

PAPI
Intel Vtune
LIKWID
Perf

[1] Saviankou et al., Cube v4: From Performance Report Explorer to Performance Analysis Tool
[2] Nagel et al., VAMPIR: Visualization and Analysis of MPI Resources
[3] Bell et al., ParaProf: A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis
Data Movement Optimization

Performance Analysis

PAPI
Intel Vtune
LIKWID
Perf

Requires Execution!

[1] Saviankou et al., Cube v4: From Performance Report Explorer to Performance Analysis Tool
[2] Nagel et al., VAMPIR: Visualization and Analysis of MPI Resources
[3] Bell et al., ParaProf: A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis
Data Movement Optimization

Performance analysis without program execution

PAPI
Intel Vtune
LIKWID
Perf

Requires Execution!

[1] Saviankou et al., Cube v4: From Performance Report Explorer to Performance Analysis Tool
[2] Nagel et al., VAMPIR: Visualization and Analysis of MPI Resources
[3] Bell et al., ParaProf: A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis
Data Movement Optimization

Performance Analysis

Reduce Data Movement

Static Dataflow Analysis

Small-Scale Parametric Simulations

Dataflow IR

DaCe [1]

Stateful Dataflow Multigraphs (SDFGs)

[1] Ben-Nun et al., Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures
Data Movement Optimization

Static Dataflow Analysis

Small-Scale Parametric Simulations

Performance Analysis

Reduce Data Movement

Dataflow IR
Data Movement Optimization

[1] Ben-Nun et al., Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures
Data Movement Optimization

Static Dataflow Analysis

Small-Scale Parametric Simulations

Performance Analysis

Reduce Data Movement

Dataflow IR

DaCe [1] Stateful DataFlow multiGraphs (SDFGs)

[1] Ben-Nun et al., Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures
Stateful DataFlow multiGraph (SDFG)

\[ C = A \otimes B \quad A \in \mathbb{R}^N, B \in \mathbb{R}^M, C \in \mathbb{R}^{N \times M} \]

```python
def outer_prod(A, B, C, N, M):
    for i in range(N):
        for j in range(M):
```

Data Containers
Parallel Region (Map)
Computations
Data Movement
State
Static Dataflow Analysis

Data Movement Volume

\[
\begin{align*}
A[i] & \rightarrow B[j] \\
C[i,j] & = A[i] \cdot B[j] \\
C[i,j] & \rightarrow C[i,j] \\
\end{align*}
\]
Static Dataflow Analysis

Data Movement Volume
1. Derive volume for computations
2. Propagate through graph

Arithmetic Operations Count
1. Count operations in AST of computations
2. Propagate through graph

Operational Intensity

Executed $N \times M$ times

$A[i] \rightarrow 1 \rightarrow 1 \rightarrow 1$

$C[i, j] = A[i] \times B[j]$

$N \times M$ Operations $\rightarrow$ Intensity: $\frac{1}{3}$
Static Dataflow Analysis

\[ \begin{align*}
A & \xrightarrow{N \times M} \{i=0:N, j=0:M\} \\
B & \xrightarrow{N \times M} \{i=0:N, j=0:M\} \\
C[i, j] = A[i] \times B[j] & \xrightarrow{1} \{i=0:N, j=0:M\} \\
C & \xrightarrow{N \times M} \{i=0:N, j=0:M\}
\end{align*} \]
Static Dataflow Analysis

\[ N = 8 \]
\[ M = 8 \]
Static Dataflow Analysis

\( N = 8 \)
\( M = 64 \)

Substitute symbols

Change symbol values to perform scaling analysis

\[ C[i, j] = A[i] \times B[j] \]

512 Operations
Intensity: \( \frac{1}{3} \)

1 Operation
Intensity: \( \frac{1}{3} \)

512 Operations
Intensity: \( \frac{1}{3} \)
Visualization

\[ N = 8 \]
\[ M = 64 \]

Visualize data by overlaying a *heatmap*

Low volume  
High volume
Visualization

\[ N = 8 \]
\[ M = 64 \]

Visualize data by overlaying a heatmap

Low operation count

High operation count

\[ C[i, j] = A[i] \times B[j] \]
Visualization

\[ N = 8 \]
\[ M = 64 \]

In-Situ performance data reduces context switching

Visualize data by overlaying a heatmap

Low operation count  High operation count

Low volume  High volume

Low operation count  High operation count

\[ N = 8 \]
\[ M = 64 \]

512 Operations
Intensity: \( \frac{1}{3} \)

1 Operation
Intensity: \( \frac{1}{3} \)

512 Operations
Intensity: \( \frac{1}{3} \)
Optimizing BERT Transformer Encoder

Data movement heatmap
Optimizing BERT Transformer Encoder

Loops with similar bounds
Optimizing BERT Transformer Encoder
Optimizing BERT Transformer Encoder
Optimizing BERT Transformer Encoder

Operational intensity heatmap
Optimizing BERT Transformer Encoder
Optimizing BERT Transformer Encoder

30.2x Speedup

16-core Intel Xeon Gold 6130 at 2.1 GHz, 1.5 TB RAM
Close-Up Reuse Analysis

Simulate data reuse behavior

\[ C = A \otimes B \quad A \in \mathbb{R}^N, B \in \mathbb{R}^M, C \in \mathbb{R}^{N \times M} \]
Close-Up Reuse Analysis

$C = A \otimes B \quad A \in \mathbb{R}^N, B \in \mathbb{R}^M, C \in \mathbb{R}^{N \times M}$
Close-Up Reuse Analysis

\[ C = A \otimes B \quad A \in \mathbb{R}^3, B \in \mathbb{R}^4, C \in \mathbb{R}^{3\times4} \]

Simulate data reuse behavior

Specify program region

Specify small example input parameters

\[ N = 3 \]
\[ M = 4 \]
Close-Up Reuse Analysis

\[ C = A \otimes B \quad A \in \mathbb{R}^3, B \in \mathbb{R}^4, C \in \mathbb{R}^{3 \times 4} \]

Specify small example input parameters

\[ N = 3 \]
\[ M = 4 \]

Specify program region

Simulate data reuse behavior
Visualizing High-Dimensional Data

\[ w \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K_Y \times K_X} \]

- \( C_{\text{out}} = 2 \)
- \( C_{\text{in}} = 3 \)
- \( K_Y = 4 \)
- \( K_X = 4 \)
Visualizing High-Dimensional Data

\[ w \in \mathbb{R}^{C_{out} \times C_{in} \times K_X \times K_Y} \]

- \( C_{out} = 2 \)
- \( C_{in} = 3 \)
- \( K_X = 4 \)
- \( K_Y = 4 \)
Access Pattern Simulation

Convolution operation
Access Pattern Simulation

Visually play back access pattern

\[ y[i, j, k, l] = x[i, m, k+ky, l+kx] \cdot w[j, m, ky, kx] \]
Access Pattern Simulation

Flatten time dimension with heatmap
Access Pattern Simulation

\[ C[i, j] += A[i, k] \times B[k, j] \]
Data Layout Visualization

- Exposes data layout
- Data layout & access pattern → spatial locality

float64 / double → element size = 8 bytes

Determine cache line using strides, line size, and element size

C[i, j] += A[i, k] * B[k, j]
Temporal Locality

Stack distance, cache line granularity

Accesses to *unique* addresses since last reference

```
C[i, j] += A[i, k] * B[k, j]
```
Cache Misses

1. Cold miss
   Access with stack distance = \( \infty \)

2. Capacity miss
   Access with stack distance \( \geq t_d \), stack distance threshold

3. Conflict miss
   Not counted in fully-associative cache
   Calculations generalizeable [1][2]

\[ C[i, j] = A[i, k] \times B[k, j] \]

Physical data movement = \#misses x cache line size

\( t_d = 5 \)

[1] McKinley and Temam, Quantifying Loop Nest Locality Using SPEC’95 and the Perfect Benchmarks
[2] Beyls and D’Hollander, Reuse distance as a metric for cache behavior
Stencil Optimization

\[ I = 8 \]
\[ J = 8 \]
\[ K = 5 \]

Accesses spread over non-contiguous dimension

Cache line shows \( K \) as contiguous dimension

Original sizes:
\[ I = 256 \]
\[ J = 256 \]
\[ K = 160 \]
Scaling Factor x32

\[ I = 8 \]
\[ J = 8 \]
\[ K = 5 \]

\[ \text{in\_field}[I+4, J+4, K] \]

\[ \text{coeff}[I, J, K] \]

\[ \text{out\_field}[I, J, K] \]

\[ \text{hdiff} \]
Stencil Optimization

$I = 8$
$J = 8$
$K = 5$

Reshape data containers

Poor use of cache

Better use of spatial locality

$\mathbf{I} = 8$$
$\mathbf{J} = 8$$
$\mathbf{K} = 5$
Stencil Optimization

\[ I = 8 \]
\[ J = 8 \]
\[ K = 5 \]

Iterates over contiguous dimension

Reorder loops
Stencil Optimization

16-core Intel Xeon Gold 6130 at 2.1 GHz, 1.5 TB RAM

Predicted 10.3x reduction in data movement

138x Speedup
9.6x Reduction in cache misses
Conclusion

Global Data Movement

Fine-Grained Data Reuse

in_field [i=3, j=4, k]

ocelf [i, j, k]

bdfj

out_field [i, j, k]
Where Next?

Automatic Optimization

Hardware Modelling

Educational Tool

Loop reordering

Reg
L1
L2
L3
Main Memory
Disk
Network
Thank you!

https://github.com/spcl/dace-vscode