Extending Pluto-Style Polyhedral Scheduling with Consecutivity

Sven Verdoolaege[1] Alexandre Isoard[2]

[1]KU Leuven and Polly Labs

[2]Xilinx

January 23, 2018

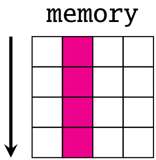
Outline

Outline

Consecutivity Concept

- Temporal Locality
  Consecutive operations access
  the same memory element

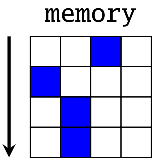
  ⇒ reuse of data in cache or registers



Consecutivity Concept

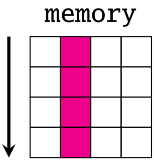
- Spatial Locality
  Consecutive operations access
  neighboring memory elements

  $\Rightarrow$ reuse of cache lines



- Temporal Locality
  Consecutive operations access
  the same memory element

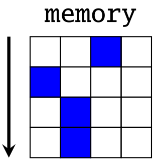
  $\Rightarrow$ reuse of data in cache or registers



Consecutivity Concept

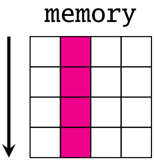
- Spatial Locality
  Consecutive operations access
  neighboring memory elements

  ⇒ reuse of cache lines



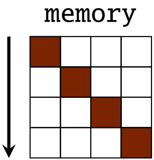
- Temporal Locality
  Consecutive operations access
  the same memory element

  ⇒ reuse of data in cache or registers



- Consecutivity
  Consecutive operations access
  consecutive memory elements

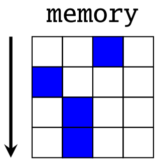
  ⇒ vectorization
  ⇒ hardware cache prefetcher
  ⇒ burst accesses, e.g., on FPGA (Xilinx)



Consecutivity Concept

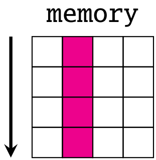
- Spatial Locality
  Consecutive operations access
  neighboring memory elements

  ⇒ reuse of cache lines



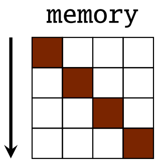
- Temporal Locality
  Consecutive operations access
  the same memory element

  ⇒ reuse of data in cache or registers



- Consecutivity
  Consecutive operations access
  consecutive memory elements

  ⇒ vectorization
  ⇒ hardware cache prefetcher
  ⇒ burst accesses, e.g., on FPGA (Xilinx)



Burst Accesses (Sketch)

```
for (int i = 0; i < N; ++i) {

  for (int j = 0; j < M; ++j) {
                         C[j][i]  =
                          A[i]  *
                          B[j] ;
  }

}
```

Burst Accesses (Sketch)

```
AA = burst_read_start(A, N);
for (int i = 0; i < N; ++i) {
  BB = burst_read_start(B, M);
  for (int j = 0; j < M; ++j) {
                              C[j][i]  =
      burst_read_iter(AA, &A[i]) *
      burst_read_iter(BB, &B[j]);
  }
  burst_read_end(BB, M);
}
burst_read_end(AA, N);
```

Burst Accesses (Sketch)

```
AA = burst_read_start(A, N);
for (int i = 0; i < N; ++i) {
  BB = burst_read_start(B, M);
  for (int j = 0; j < M; ++j) {
                          C[j][i]  =
      burst_read_iter(AA, &A[i]) *
      burst_read_iter(BB, &B[j]);
  }
  burst_read_end(BB, M);
}
burst_read_end(AA, N);
```

No burst accesses on C

Burst Accesses (Sketch)

```
for (int i = 0; i < N; ++i) {
  for (int j = 0; j < M; ++j) {
                                C[j][i]  =
                                 A[i]  *
                                 B[j] ;
  }
}
```

No burst accesses on C

Burst Accesses (Sketch)

```
for (int j = 0; j < M; ++j) {
  for (int i = 0; i < N; ++i) {
                                    C[j][i]  =
                                     A[i]  *
                                     B[j] ;
  }
}
```

Burst Accesses (Sketch)

```
CC = burst_write_start(C, M * N);
BB = burst_read_start(B, M);
for (int j = 0; j < M; ++j) {
  AA = burst_read_start(A, N);
  for (int i = 0; i < N; ++i) {
    burst_write_iter(CC, &C[j][i]) =
      burst_read_iter(AA, &A[i]) *
      burst_read_iter(BB, &B[j]);
  }
  burst_read_end(AA, N);
}
burst_read_end(BB, M);
burst_write_end(CC, M * N);
```

Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances

- original schedule (if any) derived from input
- target schedule computed by scheduler

Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances

- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

- instance set: set of schedulable statement instances
- access relations: map instances to memory locations
- dependence relations:
  - ⇒ pairs of instances that need to be executed in order
  - ⇒ derived from access relations and original schedule

Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances

- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

Result (typically):

- multiple (quasi) affine functions on instance set
- hierarchically organized (sequence, tree)

Types:

- Farkas based schedulers (Feautrier 1992)

  $\Rightarrow$ use Farkas to transform dependences into constraints on schedule coefficients

  - Pluto-style schedulers, e.g., `Pluto`, `isl`
    $\Rightarrow$ compute affine functions one by one

- ...

Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances

- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

Result (typically):

- multiple (quasi) affine functions on instance set
- hierarchically organized (sequence, tree)

Types:

- Farkas based schedulers (Feautrier 1992)
  $\Rightarrow$ use Farkas to transform dependences
  into constraints on schedule coefficients
  - Pluto-style schedulers, e.g., `Pluto`, `isl`
    $\Rightarrow$ compute affine functions one by one
  - one-shot schedulers (Vasilache 2007)
    $\Rightarrow$ compute entire schedule as a whole
- ...

Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances

- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

Result (typically):

- multiple (quasi) affine functions on instance set
- hierarchically organized (sequence, tree)

Types:

- Farkas based schedulers (Feautrier 1992)

  ⇒ use Farkas to transform dependences
  into constraints on schedule coefficients

  - Pluto-style schedulers, e.g., `Pluto`, `isl`
    ⇒ compute affine functions one by one
  - one-shot schedulers (Vasilache 2007)
    ⇒ compute entire schedule as a whole
- ...

Pluto-Style Polyhedral Scheduling

Main optimization criteria:

- parallelism
- temporal locality
- permutability $\Rightarrow$ tiling

Pluto-Style Polyhedral Scheduling

Main optimization criteria:

- parallelism
- temporal locality
- permutability $\Rightarrow$ tiling

Remaining freedom (if any)

$\Rightarrow$ `isl` scheduler tends towards lexicographic ordering of instances

Extreme example:

```
for (i=0; i<M; ++i)
  for (j=0; j<N; ++j)
S:  A[i][j] = 0;
```

$$\mathtt{S}[i,j] \to [i,j]$$

consecutive (by chance)

```
for (i=0; i<M; ++i)
  for (j=0; j<N; ++j)
T:  B[j][i] = 0;
```

$$\mathtt{T}[i,j] \to [i,j]$$

not consecutive

Pluto-Style Polyhedral Scheduling

Main optimization criteria:

- parallelism
- temporal locality
- permutability $\Rightarrow$ tiling

Remaining freedom (if any)

$\Rightarrow$ `isl` scheduler tends towards lexicographic ordering of instances

Extreme example:

```
for (i=0; i<M; ++i)
  for (j=0; j<N; ++j)
S:  A[i][j] = 0;
```

$$\mathtt{S}[i,j] \to [i,j]$$

consecutive (by chance)

```
for (i=0; i<M; ++i)
  for (j=0; j<N; ++j)
T:  B[j][i] = 0;
```

$$\mathtt{T}[i,j] \to [i,j]$$

not consecutive

Goal: steer towards consecutivity in case of sufficient freedom

Current implementation in `isl` (roughly):

permutability > parallelism > consecutivity > temporal locality

Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)

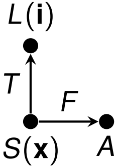
- intra-statement consecutivity ($\Rightarrow$ per statement)
- row-major array layout
- purely affine access function F
- purely affine per-statement schedule T

Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)

- intra-statement consecutivity ($\Rightarrow$ per statement)
- row-major array layout
- purely affine access function F
- purely affine per-statement schedule T



Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

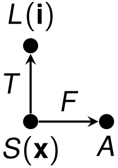
$$[\ldots + 0 i_n] \ldots [\ldots + 0 i_n][\ldots + 1 i_n]$$

Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)

- intra-statement consecutivity ($\Rightarrow$ per statement)
- row-major array layout
- purely affine access function F
- purely affine per-statement schedule T



Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$[\ldots + 0 i_n] \ldots [\ldots + 0 i_n][\ldots + 1 i_n]$$

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)

- intra-statement consecutivity ($\Rightarrow$ per statement)
- row-major array layout
- purely affine access function $F = [G; H]$
- purely affine per-statement schedule $T = [T_1; T_2]$

$$\begin{tikzcd} L(\mathbf{i}) & \\ S(\mathbf{x}) \arrow[u, "T"] \arrow[r, "F"] & A \end{tikzcd}$$

Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$[\ldots + 0 i_n] \ldots [\ldots + 0 i_n][\ldots + 1 i_n]$$

$$F\,T^{-1} = \begin{bmatrix} G \\ H \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

Consecutivity Criterion Reformulation

Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F\,T^{-1} = \begin{bmatrix} G \\ H \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

Consecutivity Criterion Reformulation

Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F\,T^{-1} = \begin{bmatrix} G \\ H \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

- $G\,\mathbf{q} = \mathbf{0}$ (with $\mathbf{q}$ the final columns of T^{-1})

  Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^{\mathrm{t}} & 1 \end{bmatrix}$

  $\Rightarrow$ $\mathbf{q}$ spans $\ker T_1$

  $\Rightarrow$ $\ker T_1 \subseteq \ker G$ (Vasilache et al. 2012)

  That is, rows of G need to be linear combinations of rows of T_1

  $G = A\,T_1$

Consecutivity Criterion and Spatial Locality

- Consecutivity

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

Consecutivity Criterion and Spatial Locality

Spatial Locality

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & x \end{bmatrix}$$

- Consecutivity

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

Consecutivity Criterion and Spatial Locality

Spatial Locality

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & x \end{bmatrix}$$

- Temporal Locality

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 0 \end{bmatrix}$$

- Consecutivity

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

Consecutivity Criterion and Spatial Locality

Spatial Locality

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & x \end{bmatrix}$$

- Temporal Locality

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 0 \end{bmatrix}$$

- Consecutivity

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

in case of innermost temporal locality
$\Rightarrow$ consecutivity on next innermost loop iterator

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} & \mathbf{0} \\ N & 1 & 0 \end{bmatrix}$$

(Kandemir, Ramanujam, and Choudhary 1999)

Related Work on Spatial Locality

Loop nest transformations (not per-statement)

- Wolf and Lam (1991)
  - define temporal ($\ker F$) and spatial ($\ker G$) reuse directions
  - partition original loop iterators
- Kandemir, Ramanujam, and Choudhary (1999)
  - aim: spatial locality
  - criterion more strict than required (ensures consecutivity)
  - incrementally fix elements of T^{-1}
- Kandemir, Ramanujam, Choudhary, and Banerjee (2001)
  - pick (second to) last column of T^{-1} from $\ker G$

Related Work on Spatial Locality

Per-statement schedulers

- Bastoul and Feautrier (2004)
  - pick proto-schedule T orthogonal to element from $\ker G$ (or $\ker F$)
  - construct valid schedule $C\,T$
  - imposing constraints on linear combinations
    $\Rightarrow$ not directly applicable in `isl`
- Vasilache et al. (2012)
  - aim: spatial locality ($\ker T_1 \subseteq \ker G$)
  - one-shot scheduler called multiple times
  - soft constraints encoded in ILP
- `Pluto` (2012) post scheduling intra-tile interchange
- Kong et al. (2013)
  - aim: consecutivity (stride-1 or stride-0)
  - partition original loop iterators
  - soft constraints encoded in ILP
- Zinenko et al. (2018)
  - spatial locality through spatial proximity constraints
  - soft constraints encoded in ILP

Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression
    do *not* appear in outer schedule rows

Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression
    do *not* appear in outer schedule rows

  - consecutivity requires innermost index expression to be equal to innermost schedule row (+ linear combinations of outer schedule rows)
  - how to handle iterators that appear in both?

```
for (int i = 0; i < M; ++i)
  for (int j = 0; j < N; ++j)
S:  A[j][j - i] = f(i, j);
```

Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression
    do *not* appear in outer schedule rows

  - consecutivity requires innermost index expression to be equal to innermost schedule row (+ linear combinations of outer schedule rows)
  - how to handle iterators that appear in both?

```
for (int i = 0; i < M; ++i)
  for (int j = 0; j < N; ++j)
S:  A[j][j - i] = f(i, j);
```

Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression
    do *not* appear in outer schedule rows
  - consecutivity requires innermost index expression to be equal to innermost schedule row (+ linear combinations of outer schedule rows)
  - how to handle iterators that appear in both?

```
for (int i = 0; i < M; ++i)
  for (int j = 0; j < N; ++j)
S:  A[j][j - i] = f(i, j);
```

Other approaches, e.g., using $\mathrm{S}[i,j] \to [j,-i]$:

```
for (int c0 = 0; c0 < N; c0 += 1)
  for (int c1 = -c0; c1 <= 0; c1 += 1)
    A[c0][c0 + c1] = f(-c1, c0);
```

Limitations

- post-schedule interchange
  - does not perform reversal, skewing
  - does not differentiate between statements
  - does not affect shape of schedule (e.g., distribution)

Limitations

- post-schedule interchange
  - does not perform reversal, skewing
  - does not differentiate between statements
  - does not affect shape of schedule (e.g., distribution)

```
void trps(int N, __pencil_consecutive float A[N][N],
   __pencil_consecutive float C[N][N])
{
   float tmp[N][N];
   for (int i = 0; i < N; i++)
      for (int j = 0; j < N; j++) {
S:       tmp[i][j] = A[i][j];
T:       C[j][i] = tmp[i][j];
      }
}
```

  - without consecutivity:
    - ⇒ temporal locality on `tmp` prevents loop distribution
  - with consecutivity:
    - ⇒ consecutivity requires different transformation per statement
    - ⇒ loop distribution

Outline

Consecutivity Criterion Reformulation

Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F\,T^{-1} = \begin{bmatrix} G \\ H \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

- $G\,\mathbf{q} = \mathbf{0}$ (with $\mathbf{q}$ the final columns of T^{-1})

  Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^{\mathrm{t}} & 1 \end{bmatrix}$

  $\Rightarrow$ $\mathbf{q}$ spans $\ker T_1$

  $\Rightarrow$ $\ker T_1 \subseteq \ker G$ (Vasilache et al. 2012)

  That is, rows of G need to be linear combinations of rows of T_1

  $G = A\,T_1$

Consecutivity Criterion Reformulation

Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F\,T^{-1} = \begin{bmatrix} G \\ H \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

- $G\,\mathbf{q} = \mathbf{0}$ (with $\mathbf{q}$ the final columns of T^{-1})

  Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^{\mathrm{t}} & 1 \end{bmatrix}$

  $\Rightarrow$ $\mathbf{q}$ spans $\ker T_1$

  $\Rightarrow$ $\ker T_1 \subseteq \ker G$ (Vasilache et al. 2012)

  That is, rows of G need to be linear combinations of rows of T_1

  $G = A\,T_1$
- $H\,\mathbf{q} = 1$

  $H = T_2 + B\,T_1$

Consecutivity Criterion Reformulation

Transformed access function $F\,T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F\,T^{-1} = \begin{bmatrix} G \\ H \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix}$$

- $G\,\mathbf{q} = \mathbf{0}$ (with $\mathbf{q}$ the final columns of T^{-1})

  Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^{\mathrm{t}} & 1 \end{bmatrix}$

  $\Rightarrow$ $\mathbf{q}$ spans $\ker T_1$

  $\Rightarrow$ $\ker T_1 \subseteq \ker G$ (Vasilache et al. 2012)

  That is, rows of G need to be linear combinations of rows of T_1

  $G = A\,T_1$
- $H\,\mathbf{q} = 1$

  $H = T_2 + B\,T_1$

$\Rightarrow$ H needs to be linearly independent of G

Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

$$F = \begin{bmatrix} G \\ H \end{bmatrix}$$

given

  - H linearly independent of G

Goal:

  - G linear combination of outer schedule rows: $G = A\, T_1$
  - H equal to innermost schedule row : $H = T_2 + B\, T_1$

Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

$$F = \begin{bmatrix} G \\ H \end{bmatrix}$$

  given
  - H linearly independent of G

  Goal:
  - G linear combination of outer schedule rows: $G = A\ T_1$
  - H equal to innermost schedule row : $H = T_2 + B\ T_1$
- multiple references per statement
  $\Rightarrow$ potential conflicts
  Possible resolutions:
  - maximize number of satisfied consecutivity constraints
  - consider constraints in order specified by user

Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

$$F = \begin{bmatrix} G \\ H \end{bmatrix}$$

  given
  - H linearly independent of G

  Goal:
  - G linear combination of outer schedule rows: $G = A\ T_1$
  - H equal to innermost schedule row : $H = T_2 + B\ T_1$
- multiple references per statement
  $\Rightarrow$ potential conflicts
  Possible resolutions:
  - maximize number of satisfied consecutivity constraints
  - consider constraints in order specified by user

Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

$$F = \begin{bmatrix} G \\ H \end{bmatrix}$$

  given
  - H linearly independent of G
  - rows of H linearly independent

  Goal:
  - G linear combination of outer schedule rows: $G = A\, T_1$
  - H equal to innermost schedule rows: $H = T_2 + B\, T_1$
- multiple references per statement
  $\Rightarrow$ potential conflicts
  Possible resolutions:
  - maximize number of satisfied consecutivity constraints
  - consider constraints in order specified by user
    $\Rightarrow$ some constraints may be combined constraints with multi-row H

Multiple References Example: Matrix Multiplication

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

Multiple References Example: Matrix Multiplication

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

$$F_A = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \quad F_B = \left[\begin{array}{ccc} 0 & 0 & 1 \\ \hdashline 0 & 1 & 0 \end{array}\right] \quad F_C = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right]$$

Multiple References Example: Matrix Multiplication

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

$$F_A = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \quad F_B = \left[\begin{array}{ccc} 0 & 0 & 1 \\ \hdashline 0 & 1 & 0 \end{array}\right] \quad F_C = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right]$$

$$F_{BC} = \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right]$$

Multiple References Example: Matrix Multiplication

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

$$F_{\mathtt{A}} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \quad F_{\mathtt{B}} = \left[\begin{array}{ccc} 0 & 0 & 1 \\ \hdashline 0 & 1 & 0 \end{array}\right] \quad F_{\mathtt{C}} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right]$$

$$F_{\mathtt{BC}} = \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right] \qquad F_{\mathtt{ABC}} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 \\ 0 & 1 & 0 \end{array}\right]$$

Multiple References Example: Matrix Multiplication

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

$$F_A = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \quad F_B = \left[\begin{array}{ccc} 0 & 0 & 1 \\ \hdashline 0 & 1 & 0 \end{array}\right] \quad F_C = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right]$$

$$F_{BC} = \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 \end{array}\right] \qquad F_{ABC} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 \\ 0 & 1 & 0 \end{array}\right]$$

List: $F_{ABC}, F_{AC}, F_{AB}, F_{BC}, F_A, F_B, F_C$

Multiple Final Rows

- single final row

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix} \qquad \text{or} \qquad F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} & \mathbf{0} \\ N & 1 & 0 \end{bmatrix}$$

Multiple Final Rows

- single final row

$$F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} \\ N & 1 \end{bmatrix} \qquad \text{or} \qquad F\,T^{-1} = \begin{bmatrix} M & \mathbf{0} & \mathbf{0} \\ N & 1 & 0 \end{bmatrix}$$

- multiple final rows

$$F\,T^{-1} = \begin{bmatrix} & 0 & & 0 & & \cdots & & 0 & \\ & \vdots & & \vdots & & & & \vdots & \\ & 0 & & 0 & & \cdots & & 0 & \\ A & & 0 & & 0 & & 0 & & 0 \\ & 1 & & 0 & & \cdots & & 0 & \\ & & & 1 & & \ddots & & \vdots & \\ & & & & & \ddots & & 0 & \\ & & L & & & & & 1 & \end{bmatrix}$$

  - multiple levels of consecutivity
  - multiple levels of temporal locality (optional)

Constraints on Schedule Coefficients

Affine schedule row:

$$f_S(\mathbf{x}) = \boxed{\quad C_S \quad} \boxed{\mathbf{x}} + \boxed{d_S}$$

Constraints:

- validity: $f_T(\mathbf{y}) - f_S(\mathbf{x}) \ge 0$
  Farkas $\rightarrow$ constraints on C_S and d_S
- proximity (temporal locality): $f_T(\mathbf{y}) - f_S(\mathbf{x})$ small
  Farkas $\rightarrow$ constraints on C_S and d_S
- coincidence (parallelism): $f_T(\mathbf{y}) - f_S(\mathbf{x}) = 0$
  Farkas $\rightarrow$ constraints on C_S and d_S
- linear independence of previous rows ($T_{S,0}$): $C_S \neq Y T_{S,0}$

  $\Rightarrow$ compute orthogonal complement of $T_{S,0}$: $U_S \, T_{S,0}^{t} = \mathbf{0}$

  $\Rightarrow$ impose $U_S \, C_S^{t} \neq \mathbf{0}$

Constraints on Schedule Coefficients for Consecutivity

- G linear combination of outer schedule rows: $G = A\ T_1$
- H equal to innermost schedule rows: $H = T_2 + B\ T_1$

Constraints on Schedule Coefficients for Consecutivity

- G linear combination of outer schedule rows: $G = A\, T_1$
- H equal to innermost schedule rows: $H = T_2 + B\, T_1$

Three stages

1. G is not yet a linear combination of T_0
   $\Rightarrow$ take linear combination of G and T_0
   (heuristic to make progress)
   $\Rightarrow$ but linearly independent of H and T_0

$$C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \wedge C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix}$$

Constraints on Schedule Coefficients for Consecutivity

- G linear combination of outer schedule rows: $G = A\, T_1$
- H equal to innermost schedule rows: $H = T_2 + B\, T_1$

Three stages

1. G is not yet a linear combination of T_0
   $\Rightarrow$ take linear combination of G and T_0
   (heuristic to make progress)
   $\Rightarrow$ but linearly independent of H and T_0

   $$C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \wedge C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix}$$

2. G is linear combination of T_0
   $\Rightarrow$ take C equal to next row of H

   $$C = H_h + X \begin{bmatrix} T_1 \\ H_{<h} \end{bmatrix}$$

Constraints on Schedule Coefficients for Consecutivity

- G linear combination of outer schedule rows: $G = A\ T_1$
- H equal to innermost schedule rows: $H = T_2 + B\ T_1$

Three stages

1. G is not yet a linear combination of T_0 $\qquad C = X\begin{bmatrix} T_0 \\ G \end{bmatrix} \wedge C \neq Y\begin{bmatrix} T_0 \\ H \end{bmatrix}$
   $\Rightarrow$ take linear combination of G and T_0
   (heuristic to make progress)
   $\Rightarrow$ but linearly independent of H and T_0
2. G is linear combination of T_0 $\qquad C = H_h + X\begin{bmatrix} T_1 \\ H_{<h} \end{bmatrix}$
   $\Rightarrow$ take C equal to next row of H
3. all rows of H have been handled
   $\Rightarrow$ no further constraints (final zero columns in $F\ T^{-1}$)

Constraints on Schedule Coefficients for Consecutivity

- G linear combination of outer schedule rows: $G = A\, T_1$
- H equal to innermost schedule rows: $H = T_2 + B\, T_1$

Three stages

1. G is not yet a linear combination of T_0
   $\Rightarrow$ take linear combination of G and T_0
   (heuristic to make progress)
   $\Rightarrow$ but linearly independent of H and T_0

   $$C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \wedge C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix}$$

2. G is linear combination of T_0
   $\Rightarrow$ take C equal to next row of H

   $$C = H_h + X \begin{bmatrix} T_1 \\ H_{<h} \end{bmatrix}$$

3. all rows of H have been handled
   $\Rightarrow$ no further constraints (final zero columns in $F\, T^{-1}$)

At any stage

- C may also be
  linearly independent of T_0, G and H
  (intermediate zero columns in $F\, T^{-1}$)

  $$C \neq Y \begin{bmatrix} T_0 \\ G \\ H \end{bmatrix}$$

- C of lower-dimensional statement may be
  linear combination of T_0

  $$C = X T_0$$

Solving Constraints on Schedule Coefficients (`isl`)

- validity, proximity, coincidence
  - $\Rightarrow$ encoded in ILP
- linear independence

$$C \neq YT_0 \qquad \rightarrow U\,C^{\mathrm{t}} \neq \mathbf{0}$$

  - $\Rightarrow$ not linear
  - $\Rightarrow$ backtracking search (in `isl`): $U_i C^{\mathrm{t}} \geq 1$ or $U_i C^{\mathrm{t}} \leq -1$

Solving Constraints on Schedule Coefficients (isl)

- validity, proximity, coincidence
  $\Rightarrow$ encoded in ILP
- linear independence
  $$C \neq Y T_0 \qquad \to U C^{\mathrm{t}} \neq \mathbf{0}$$
  $\Rightarrow$ not linear
  $\Rightarrow$ backtracking search (in isl): $U_i C^{\mathrm{t}} \geq 1$ or $U_i C^{\mathrm{t}} \leq -1$
- consecutivity
  $$C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \qquad \to U' C^{\mathrm{t}} = \mathbf{0} \qquad \text{linear}$$
  $$C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix} \qquad \to U'' C^{\mathrm{t}} \neq \mathbf{0} \qquad \text{backtracking}$$
  Note:
  - extra rows $H \Rightarrow$ fewer rows in $U'' \Rightarrow$ fewer backtracking cases
  - no extra ILP variables, but possibly more backtracking

Solving Constraints on Schedule Coefficients (isl)

- validity, proximity, coincidence
  $\Rightarrow$ encoded in ILP
- linear independence

  $$C \neq YT_0 \qquad \rightarrow U\,C^{\mathrm{t}} \neq \mathbf{0}$$

  $\Rightarrow$ not linear
  $\Rightarrow$ backtracking search (in `isl`): $U_i C^{\mathrm{t}} \geq 1$ or $U_i C^{\mathrm{t}} \leq -1$
- consecutivity

  $$C = X\begin{bmatrix} T_0 \\ G \end{bmatrix} \qquad \rightarrow U' C^{\mathrm{t}} = \mathbf{0} \qquad \text{linear}$$

  $$C \neq Y\begin{bmatrix} T_0 \\ H \end{bmatrix} \qquad \rightarrow U'' C^{\mathrm{t}} \neq \mathbf{0} \qquad \text{backtracking}$$

  Note:
  - extra rows $H \Rightarrow$ fewer rows in $U'' \Rightarrow$ fewer backtracking cases
  - no extra ILP variables, but possibly more backtracking

  Differences with linear independence handling:
  - optional
  - fixed part that applies in each backtracking case
  - disjunctive (independent or dependent rows)
  - conditional (multiple consecutivity constraints)

Outline

Inter-Statement Consecutivity

Input:

```
for (int i = 0; i < N; i += 2)
    for (int j = 0; j < M; j += 2) {
        B[j + 0][i + 0] = A[i + 0][j + 0];
        B[j + 1][i + 0] = A[i + 0][j + 1];
        B[j + 0][i + 1] = A[i + 1][j + 0];
        B[j + 1][i + 1] = A[i + 1][j + 1];
    }
```

Output (try and obtain distances 0 and 1):

```
for (int c0 = 0; c0 < M - 1; c0 += 2) {
  for (int c1 = 0; c1 < N - 1; c1 += 2) {
    B[c0][c1] = A[c1][c0];
    B[c0][c1 + 1] = A[c1 + 1][c0];
  }
  for (int c1 = 0; c1 < N - 1; c1 += 2) {
    B[c0 + 1][c1] = A[c1][c0 + 1];
    B[c0 + 1][c1 + 1] = A[c1 + 1][c0 + 1];
  }
}
```

Outline

Local Rescheduling

Consecutivity usually only important inside tiles

1. compute schedule without consecutivity (or lower priority)
2. tile
3. recompute schedule inside tile with consecutivity

Local Rescheduling

Consecutivity usually only important inside tiles

1. compute schedule without consecutivity (or lower priority)
2. tile
3. recompute schedule inside tile with consecutivity

On `trps`:

```
float tmp[N][N];
for (int c0 = 0; c0 < N; c0 += 32)
  for (int c1 = 0; c1 < N; c1 += 32) {
    for (int c2 = c0; c2 <= min(N - 1, c0 + 31); c2 += 1)
      for (int c3 = c1; c3 <= min(N - 1, c1 + 31); c3 += 1)
        tmp[c2][c3] = A[c2][c3];
    for (int c2 = c1; c2 <= min(N - 1, c1 + 31); c2 += 1)
      for (int c3 = c0; c3 <= min(N - 1, c0 + 31); c3 += 1)
        C[c2][c3] = tmp[c3][c2];
  }
```

Outline

Conclusions and Future Work

Conclusions:

- slightly generalized criterion for consecutivity
- combining multiple references per statement
- approach for integration in Pluto-style scheduler
- implementation in `isl`/`PPCG` (branch `consecutivity_CW_709`)

Future work:

- experiment and fine-tune

References I

Bastoul, Cédric and Paul Feautrier (2004). “More Legal Transformations for Locality”. In: *Euro-Par’10 International Euro-Par conference*. Vol. 3149. Lecture Notes in Computer Science. Pisa, pp. 272–283. DOI: 10.1007/978-3-540-27866-5_36.

Bondhugula, Uday, Albert Hartono, J. Ramanujam, and P. Sadayappan (2008). “A practical automatic polyhedral parallelizer and locality optimizer”. In: *Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation*. PLDI ’08. Tucson, AZ, USA: ACM, pp. 101–113. DOI: 10.1145/1375581.1375595.

Feautrier, Paul (1992). “Some Efficient Solutions to the Affine Scheduling Problem. Part I. One-dimensional Time”. In: *International Journal of Parallel Programming* 21.5, pp. 313–348. DOI: 10.1007/BF01407835.

Kandemir, Mahmut T., J. Ramanujam, and Alok N. Choudhary (1999). “Improving Cache Locality by a Combination of Loop and Data Transformation”. In: *IEEE Transactions on Computers* 48.2, pp. 159–167. DOI: 10.1109/12.752657.

References II

Kandemir, Mahmut T., J. Ramanujam, Alok N. Choudhary, and Prithviraj Banerjee (2001). "A Layout-Conscious Iteration Space Transformation Technique". In: *IEEE Transactions on Computers* 50.12, pp. 1321–1335. DOI: 10.1109/TC.2001.970571.

Kong, Martin, Richard Veras, Kevin Stock, Franz Franchetti, Louis-Noël Pouchet, and P. Sadayappan (2013). "When polyhedral transformations meet SIMD code generation". In: *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*. PLDI '13. Seattle, Washington, USA: ACM, pp. 127–138. DOI: 10.1145/2491956.2462187.

V., Sven (2010). "isl: An Integer Set Library for the Polyhedral Model". In: *Mathematical Software - ICMS 2010*. Ed. by Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama. Vol. 6327. Lecture Notes in Computer Science. Springer, pp. 299–302. DOI: 10.1007/978-3-642-15582-6_49.

References III

V., Sven, Juan Carlos Juega, Albert Cohen, José Ignacio Gómez, Christian Tenllado, and Francky Catthoor (2013). "Polyhedral parallel code generation for CUDA". In: *ACM Trans. Archit. Code Optim.* 9.4, p. 54. DOI: 10.1145/2400682.2400713.

Vasilache, Nicolas (2007). "Scalable Program Optimization Techniques in the Polyhedral Model". PhD thesis. Université Paris Sud XI, Orsay.

Vasilache, Nicolas, Benoît Meister, Muthu Baskaran, and Richard Lethin (2012). "Joint Scheduling and Layout Optimization to Enable Multi-Level Vectorization". In: *IMPACT-2: 2nd International Workshop on Polyhedral Compilation Techniques*. Paris, France.

Wolf, Michael E. and Monica S. Lam (1991). "A Data Locality Optimizing Algorithm". In: *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '91. Toronto, Ontario, Canada: ACM, pp. 30–44. DOI: 10.1145/113445.113449.

References IV

Zinenko, Oleksandr, Sven V., Chandan Reddy, Jun Shirako, Tobias Grosser, Vivek Sarkar, and Albert Cohen (2018). “Modeling the Conflicting Demands of Multi-Level Parallelism and Temporal/Spatial Locality in Affine Scheduling”. In: *Proceedings of the 27th International Conference on Compiler Construction*. CC 2018. accepted.