Algebraic Tiling

Clément Rossetti   Philippe Clauss

ICube ICPS, Inria Camus, University of Strasbourg, France

Impact 2023
January 16th, 2023

Loop Tiling

A powerful & well-known loop optimising transformation

- to improve data locality
- to adjust the grain of parallelism

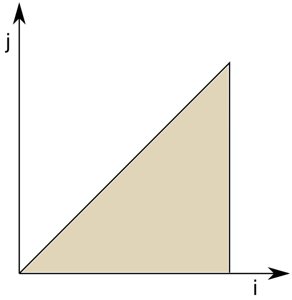
However

- What about load balancing among threads/cores?
- Tile sizes have huge impact over performance

Loop Tiling

- Workload imbalance and partial tiles issues
  Example : program syr2k (polybench) onto 5 parallel threads

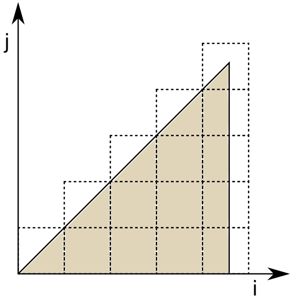
```
for(i = 0; i < N;  i++)
  for(j = 0; j <= i; j++)
    for(k = 0; k < M; k++)
      C[i][j] += A[j][k]*alpha*B[i][k]
                + B[j][k]*alpha*A[i][k];
```



Loop Tiling

- Load imbalance & partial tiles issues
  Example : syr2k (polybench) among 5 parallel threads

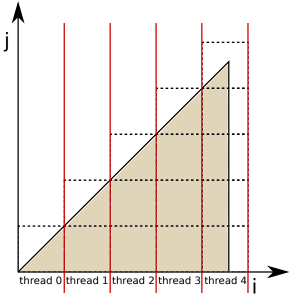
```
for (it=0;it<=(N-1)/32;it++) {
 for (jt=0;jt<=it;jt++) {
  for (kt=0;kt<=(M-1)/32;kt++) {
   for (i=32*it;
        i<=min(N-1,32*it+31);i++) {
    for (j=32*jt;
         j<=min(i,32*jt+31);j++) {
     for (k=32*kt;
          k<=min(M-1,32*kt+31);k++) {
      C[i][j] += A[j][k]*alpha*B[i][k]
               + B[j][k]*alpha*A[i][k];
```



Loop Tiling

- Load imbalance & partial tiles issues
  Example : syr2k (polybench) among 5 parallel threads

```
#pragma omp parallel for \
      private(jt,kt,i,j,k)
 for (it=0;it<=(N-1)/32;it++) {
  for (jt=0;jt<=it;jt++) {
   for (kt=0;kt<=(M-1)/32;kt++) {
    for (i=32*it;
           i<=min(N-1,32*it+31);i++) {
     for (j=32*jt;
          j<=min(i,32*jt+31);j++) {
      for (k=32*kt;
           k<=min(M-1,32*kt+31);k++) {
       C[i][j] += A[j][k]*alpha*B[i][k]
                  + B[j][k]*alpha*A[i][k];
```



Algebraic Tiling

- New approach based on tiles **volumes** (vs sizes)

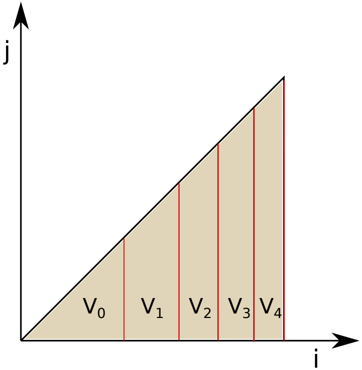
$\iff$ dynamic tile sizes to reach a given volume

- Promotes load balancing **and** data locality
  - adapted for nested parallelism: load-balancing at each depth

- Subject to the same conditions as classic tiling
  (data dependencies, vectorization, ...)

Algebraic Tiling

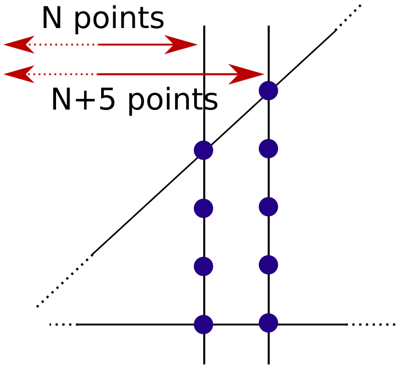
- Slices of quasi-equal volumes
  Example : syr2k (polybench) among 5 parallel threads



$$V_0 \simeq V_1 \simeq V_2 \simeq V_3 \simeq V_4$$

Algebraic Tiling

- Why "$\simeq$" ?



- If targeted slice of N+k $<$ N+5, select either N or N+5

Ehrhart polynomial of a polytope P

Exact number of points in the intersection of...

- A finite convex polyhedron (polytope) and a regular grid of points, dilated by a factor N (classic math point of view)
- A polytope depending linearly on an unknown parameter N and the grid of integer points of $\mathbb{Z}^p$

$\Rightarrow$ Ehrhart polynomial of one variable N

Multi-variate Ehrhart polynomial

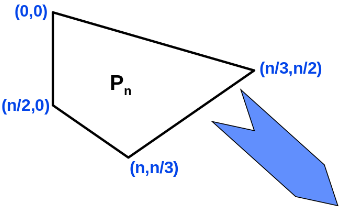
Extension to any number of parameters $N_1, N_2, ...$

- Intersection of a polytope depending linearly on several parameters $N_1, N_2, ...$ and the grid of integer points of $\mathbb{Z}^p$

$\Leftrightarrow$ Exact number of integer points that are inside a polytope which depends linearly on several parameters

$\Rightarrow$ Multi-variate Ehrhart polynomial of variables $N_1, N_2, ...$

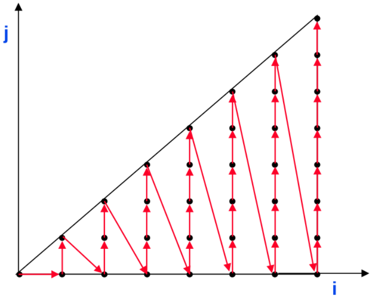
Ehrhart Polynomials



$$\frac{5}{18}n^2 + [\frac{1}{2}, \frac{4}{9}, \frac{7}{18}]n + [1, \frac{5}{18}, \frac{1}{9}, 1, -\frac{2}{9}, \frac{1}{9}]$$

Position (or rank) of an integer point

- Inside a polytope
- whose integer points are ranked following *the lexicographic order*



$$(i,j) \leq_{lex} (i',j') \text{ if } \begin{cases} i < i' \\ \text{or} \\ i = i' \text{ and } j \leq j' \end{cases}$$

Trahrhe Expressions

Inverse problem

- Given a rank of an integer point in $\mathbb{P}$, what are its integer coordinates?

$\Rightarrow$ Inverting the ranking polynomial

- Let p be a rank and $R(I)$ be the ranking polynomial
  $\Rightarrow$ solve the equation $R(I) = p$ whose unknown is I
- Find the inverse function $R^{-1}(p) = T(p)$

$=$ a sequence of algebraic expressions $t_1(p), t_2(p), \ldots$

$=$ *Trahrhe expressions*

Algebraic Tiling

- `trahrhe_i(pc)` ? Computation of a *Trahrhe Expression*

= value i for which there exist j and k such that:
at iteration (i, j, k), pc iterations have been run

- `trahrhe_i(pc)` *expression for syr2k loop nest:*

$$\left\lfloor \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M} \right\rfloor$$

Algebraic Tiling

- `trahrhe_i(pc)` ? Computation of a *Trahrhe Expression*

= value i for which there exist j and k such that:
at iteration (i, j, k), pc iterations have been run

- `trahrhe_i(pc)` *expression for syr2k loop nest:*

$$\left\lfloor \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M} \right\rfloor$$

- floating-point operations

Algebraic Tiling

- `trahrhe_i(pc)` ? Computation of a *Trahrhe Expression*

= value i for which there exist j and k such that:
at iteration (i, j, k), pc iterations have been run

- `trahrhe_i(pc)` *expression for syr2k loop nest:*

$$\left\lfloor \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M} \right\rfloor$$

- floating-point operations
- requiring sufficient precision

Algebraic Tiling

- `trahrhe_i(pc)` ? Computation of a *Trahrhe Expression*

= value i for which there exist j and k such that:
at iteration (i, j, k), pc iterations have been run

- `trahrhe_i(pc)` *expression for syr2k loop nest:*

$$\left\lfloor \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M} \right\rfloor$$

- floating-point operations
- requiring sufficient precision
- in order for the floor to be correct

Algebraic Tiling

- `trahrhe_i(pc)` ? Computation of a *Trahrhe Expression*

= value i for which there exist j and k such that:
at iteration (i, j, k), pc iterations have been run

- `trahrhe_i(pc)` *expression for syr2k loop nest:*

$$\left\lfloor \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M} \right\rfloor$$

- floating-point operations
- requiring sufficient precision
- in order for the floor to be correct
  - at least `long double` variables

Algebraic Tiling

- `trahrhe_i(pc)` ? Computation of a *Trahrhe Expression*

$=$ value i for which there exist j and k such that:
at iteration (i, j, k), pc iterations have been run

- `trahrhe_i(pc)` *expression for syr2k loop nest:*

$$\left\lfloor \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M} \right\rfloor$$

- floating-point operations
- requiring sufficient precision
- in order for the floor to be correct
  - at least `long double` variables
  - multi-precision arithmetic for some cases

Algebraic Tiling
syr2k (polybench)

```
i_pcmax = i_Ehrhart(N, M);  /* Loop Trip Count */
TILE_VOL_L1 = i_pcmax / DIV1;  /* Outermost Slices' Targeted Volume */
ubit = DIV1 - 1;  /* Number of Outermost Slices */

#pragma omp parallel for firstprivate(i_pcmax, TILE_VOL_L1) \
             private(i, j, k, lbi, ubi, lbj, ubj, jt, ubjt, j_pcmax, TILE_VOL_L2)
for (it = 0; it <= ubit; it++) {  /* Loop Scanning the Outermost Slices */
 lbi = trahrhe_i(max(it*(TILE_VOL_L1+1),1),N, M);  /* bounds of the it^th slice */
 ubi = trahrhe_i(min((it+1)*(TILE_VOL_L1+1),i_pcmax),N, M) - 1;
 if (it == ubit) ubi = N-1;  /* last slice adjustment */

 j_pcmax = j_Ehrhart(N, M, lbi, ubi);  /* Loop Trip Count of the current outermost slice */
 TILE_VOL_L2 = j_pcmax/DIV2;  /* Tiles' Targeted Volume */
 ubjt = DIV2 - 1;  /* Number of Tiles in the current slice */
 for (jt = 0; jt <= ubjt; jt++) {  /* Loop Scanning the Tiles */
  lbj = trahrhe_j(max(jt*(TILE_VOL_L2+1),1),N,M, lbi, ubi);  /* bounds of the jt^th tile */
  ubj = trahrhe_j(min((jt+1)*(TILE_VOL_L2+1),j_pcmax),N,M, lbi, ubi) - 1;
  if (jt == ubjt) ubj = ubi;  /* last tile adjustment */
  for (i = max(0,lbi); i <= min(N - 1,ubi); i++) {  /* inner tile loop */
    for (j = max(0,lbj); j <= min(i, ubj); j++) {  /* bounded by the tile bounds */
      for (k = 0; k <= M - 1; k++) {
      C[i][j] += A[j][k]*alpha*B[i][k] + B[j][k]*alpha*A[i][k];
      }
    }
  }
 } /* end for jt */
} /* end for it */
```

- Trahrhe software
  - Bash script and few C parsers
  - Intensive use of
    - *Maxima* : a free computer algebra system
    - *iscc* : the interactive interface to the barvinok counting library
  - Input : Iteration domain using isl syntax

    e.g., `[N, M] -> {[i, j, k]:  0 <= i <= N-1 and 0 <= j <= i and 0 <= k <= M-1}`
  - Compute trahrhe expressions and generate C header file containing the required functions

Implementation
Pluto extension

- Experimental pluto integration:
  - New `--atiling` option
  - Algebraic bounds are described as parameters of the iteration domain
  - Add tiling statements
  - Calls trahrhe software
  - Transparent, works like standard tiling

Experiments

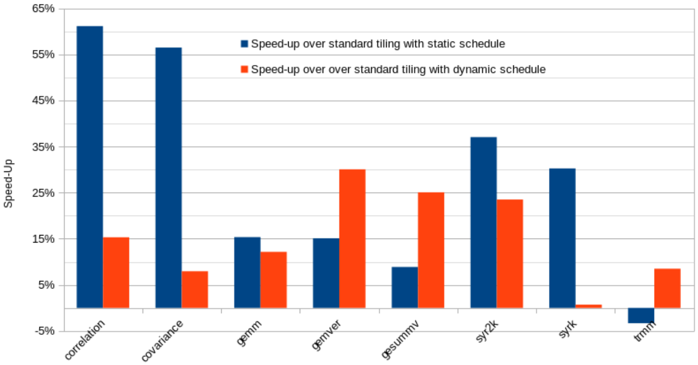


Figure: Speed-ups resulting from comparing algebraic tiling vs standard tiling with non-vectorized codes (64 threads)

Experiments

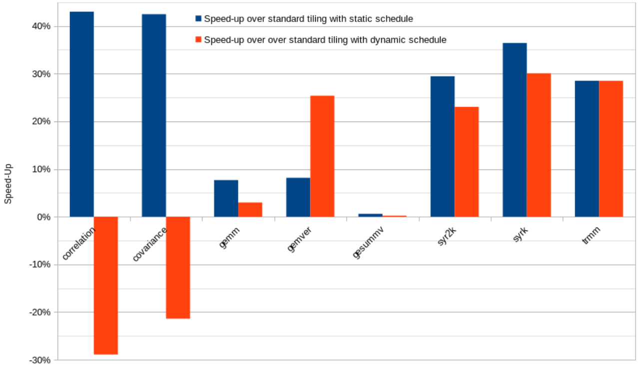


Figure: Speed-ups resulting from comparing algebraic tiling vs standard tiling with vectorized codes (64 threads)

Conclusion

- General comments:
  - Load balancing is crucial
  - Dynamic tile sizes provide significant performance improvement
  - Dynamic schedule becomes useless for polyhedral loops
- Further developments:
  - Automatic number of slices per depths using an analytic model of memory accesses
  - Investigate other applications of trahrhe expressions

Thank you!