Data-reuse Optimizations for Pipelined Tiling with Parametric Tile Sizes
Alexandre Isoard

To cite this version:
Alexandre Isoard. Data-reuse Optimizations for Pipelined Tiling with Parametric Tile Sizes. 23rd International Conference on Parallel Architectures and Compilation Techniques (PACT’14), Aug 2014, Edmonton, Canada. hal-01111393

HAL Id: hal-01111393
https://inria.hal.science/hal-01111393
Submitted on 30 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

Copyright
Data-reuse Optimizations for Pipelined Tiling with Parametric Tile Sizes

Alexandre Isoard
ENS de Lyon, France
Ph.D advisor: Alain Darte

Polyhedral Model

Symbolic representation of the semantic of affine kernels.
Allows arbitrary nesting/sequence of for loops and if conditionals, data manipulation exclusively through (possibly multidimensional) arrays. Requires all predicates (loop bounds and conditional statements) as well as all array accesses to be affine expressions of parameters or surrounding loop counters. Thus insuring the full static analyzability of the kernel using integer linear programming.

Matrix Multiply

\[
\begin{align*}
&\text{for}(i = 0; i < n; ++i) \{ \\
&\quad \text{for}(j = 0; j < n; ++j) \{ \\
&\quad\quad \text{C}[i][j] = 0; \\
&\quad\quad \text{for}(k = 0; k < n; ++k) \\
&\quad\quad\quad \text{C}[i][j] += \text{A}[i][k] * \text{B}[k][j]; \\
&\quad\quad \} \\
&\quad \} \\
\end{align*}
\]

Optimal Schedule

\[
\begin{align*}
&\text{Domain} = \{ k \rightarrow (i \geq 0) \land (j \geq 0) \land (k \geq 0) \}\ \\
&\text{Read} = \{ k \rightarrow \text{A}[i][k] \land \text{B}[j][k] \}\ \\
&\text{Write} = \{ k \rightarrow \text{C}[i][j] \} \\
&\text{Schedule} = \{ k \rightarrow (i \geq 0) \land (j \geq 0) \land (k \geq 0) \}
\end{align*}
\]

Data-reuse Optimizations for Pipelined Tiling with Parametric Tile Sizes

Parametric Tiling

Tiling is a common technique involving reordering of computation into blocks. The objective is to increase the locality of the data accesses. It usually requires a rescheduling of the loop-nest prior to blocking to ensure correctness but also to improve the final locality. Selection and application of a good schedule is made possible and fully automatic by current polyhedral techniques.

When the size of the generated tiles is a parameter instead of a static constant we speak of parametric tiling.

Parametric Tiling is problematic as it breaks the polyhedral model by introducing a division or a modulo between unknowns. Problem which is usually dealt by applying the transformation at the very end of the optimization process, during code generation. Thus forbidding any further optimization involving inter-tiles analysis.

Execution of the given tile will require to:
- load 3 cells of A
- load 6 cells of B
- load/store 4 cells of C.

Data-Reuse / Pipelining

Executing the kernel tile by tile and holding and transferring only necessary data allows to control the local memory footprint and the amount of communication as memory transfers are coalesced before and after each tile and benefit from the good locality.

Pipelining of the computation consists in loading the data for the current tile during the computation of the previous ones and in storing back the results during the computation of the next ones. Pipelining requires a full data-reuse on at least the same length to avoid incorrectly loading data while it is being produced by running tiles.

As a rule of thumb:
- coalescing on bigger tiles → memory transfer, bigger local memory
- pipelining with a bigger pipeline depth → better transfer/computation overlap

Main trick for Parametric Analysis:
The set of points to the left of the green line and the set of points to the right of the red line have a computable affine expression! Knowing the tile in which those points actually live is not required!

What is done? / What is next?

- Theory for the exact and approximated case is published:
- Proof of concept implementation of the exact case analysis is working.
- A basic lattice based memory allocation has been implemented.
- Full implementation into ppce is underway
- Cost model to actually choose the tile size remains to do.
- Designing a general scheme for approximation of data sets might be interesting.
- Improved lattice based memory allocation.