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.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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.

- Linear algebra, stencil code, some graph algorithm, statistics, ...

Kernel Offloading / Scratch Pad Programming / Instruction Level Parallelization

All those techniques involve more or less the same scheme:

- e.g. FPGA, GPU, ASICS
- low throughput
- high latency
- high throughput
- low latency

Scratch Pad Programming

Involves an explicit cache. Transfers from and to the cache are governed by the software through explicit copies.
- e.g. GPU Shared Memory

Instruction Level Parallelization

When enough spare registers are available, increasing the number of parallel instructions improves performances.
- e.g. GPU Private Memory

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. 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 → less memory transfer, bigger local memory
- pipelining with a bigger pipeline depth → better transfer/computation overlap
- data-reuse with a bigger reuse length → less memory transfer, bigger local memory

What is done? / What is next?

- Proof of concept implementation of the exact case analysis is working.
- A basic lattice based memory allocation has been implemented.
- Full implementation into ppcc 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.