Kernel Partitioning of Streaming Applications: A Statistical Approach to an NP-complete Problem

Petar Radojković, Paul M. Carpenter, Miquel Moretó, Ales Ramírez, Francisco J. Cazorla

Motivation

Stream programming languages
- Programming of multithreaded applications is difficult
- A possible solution: Expose the parallelism to the compiler
  - The application is presented as a stream graph
  - Suitable for applications that process long sequences of data: voice, image, multimedia, Internet and communication traffic, etc.

Problem

How to (optimally) partition kernels into software threads?
- The compilation problem
  - Color the nodes of the graph
- Exact solution (Garey and Johnson, 1979)
- Vast exploration space (e.g. 1016 possible kernel partitions)
- NP-complete (Garey and Johnson, 1979)

Our proposal

Estimate the performance of the optimal kernel partition

Step 1: Execute random (i.i.d.) kernel partitions

Step 2: Measure the performance of each of them

Step 3: Estimate the performance of the optimal partition

The optimal performance ranges from \( \frac{1}{10} \) to \( \frac{1}{9} \) [confidence level = 0.9, 0.95; 0.99]

Results

Can we apply EVT to the KP problem?
- The samples should be uniformly distributed

Is the estimation precise?
- Few 1000 KPs are sufficient for a precise estimation

Application of Extreme Value Theory

Process scheduling for MT CPUs

Is the estimation accurate?
- The estimation is accurate

Civil engineering

Numerous real-life problems

Can random sampling find a good kernel partition (KP)?

- Total number of KPs: Finite but vast (e.g. 10^16)
- Random sample of 1000 KPs
  - Probability (capture the best KP) = 0
  - Probability (capture one out of the best 1% of KPs) = 99.999% *


High pressure on the compiler

Source code
- Explicit dependencies

Compiler
- Complex code analysis and optimizations

(Optimal) Multithreaded executable

The importance of a good kernel partitioning
- StreamIt 2.1.1 benchmark suite
- Exactly four software threads
- Observe the performance of good and bad KPs
  - The performance difference ranges from 2.4x to 3.9x

The performance of the optimal kernel partitioning is unknown

State of the art approaches are based on heuristics
- Try to find a good kernel partition

Kernel partitioning (KP) is an intractable problem

Can random sampling find a good kernel partition (KP)?

Performance

State of the art
New KP method

Optimal kernel partition

Are we close to the optimal?

Is the estimation accurate?
- The estimation is accurate

Can random sampling find a good KP?
- Random sampling provides very good results