Performance Prediction of Image Processing Algorithms on Heterogenous Architectures

Romain SAUSSARD: PhD student at Renault S.A.S. and U. Paris Sud
Boubker BOUZID, Renault S.A.S.
Marius VASILIU, IEF U. Paris Sud
Roger REYNAUD, IEF U. Paris Sud
Context: Image Processing for ADAS

- Image processing algorithms for ADAS (Advanced Driver Assistance Systems) applications
  - Lane detection
  - Pedestrian detection
  - ...
- High computational cost, but easily parallelizable
- Embeddability?
Context: Embeddability

- Large diversity of computing architecture
  - Heterogeneous SoCs
- Real-time constraints
  - Rate
  - Latency
- Association algorithm – computing architecture
Several processing units with different capabilities
Massively parallel computing unit
Example:
- Nvidia Tegra K1
- Texas Instrument TDA2x
Nvidia Tegra K1

- ARM A15 4 cores @ 1.8 GHz
- GPU Kepler with 192 cuda cores @ 700MHz
- Image Signal Processor (ISP)
- Shared global memory
Texas Instrument TDA2x

- ARM A15 2 cores @ 750 MHz
- Dual Cortex-M4
- C66x DSPs
- EVE (Embedded Vision Engine) cores
Heterogeneous SoC: Kernel mapping

- Heterogeneous SoC
  - Processor 1
  - Processor 2

- Input Image
- Kernel 1
- Kernel 2
- Kernel 3
- Kernel 4

- Output Image

Algorithm: many kernels

Kernel Mapping Optimization
- Execution time of each kernel
- Transfer delays
- Dependencies
- Real-time constraints
Our methodology

➢ Inputs
  ❖ Algorithm to embed
  ❖ Set of SoCs
  ❖ Real-time constraints

➢ Performance prediction
Arithmetic Intensity

- Number of operations for each memory instruction
- $Mi$: set of memory instructions
- $Ci$: set of computing instructions
- Used for bottleneck identification:
  - Low $Ia$: memory bandwidth limitation
  - High $Ia$: computational limitation

$$Ia = \frac{N_{Ci}}{N_{Mi}}$$
SoCs modeling

- Generic test vectors
- Characteristics extraction

Performance Prediction

- Computing time of a kernel
- Memory access delays
- Prediction for multiple kernels
- Use-case: pedestrian detection
Computing Time Prediction of a Kernel

- Each computing instruction \( Ci \) can be divided in 6 classes:
  - \( S\_Int \): simples operations on \( int \)
  - \( M\_Int \): multiplications on \( int \)
  - \( Float \): floating point operations
  - \( Spec \): specific operations (\( sqrt \), \( div \), etc)
  - \( Branch \) (for, if, etc.)
  - \( Address \): []

- \( p_{c,a} \): throughput of computing unit \( a \), for class \( c \in Ci \)

\[
t_{\text{max},a} = \sum_{c \in Ci} \frac{N_c}{p_{c,a}}; \quad t_{\text{min},a} = \max_{\{c \in Ci\}} \left( \frac{N_c}{p_{c,a}} \right) \Rightarrow t_a = [t_{\text{min},a}, t_{\text{max},a}]
\]
Memory Bandwidth and Latency

- A generic benchmark-vectors set is applied on the architecture
- Prediction are based on benchmarks results

![Graph showing memory bandwidth and latency](image)
Prediction for Multiple Kernels

- Predicted execution time = a random variable following a uniform distribution
- Summing two predicted execution times => summing two independent random variables
- Given by convolution of density functions

\[ f(x) \]

\[ \frac{1}{b-a} \]

\[ 0 \quad a \quad b \quad x \]
Application: Histogram of Oriented Gradient

- HOG descriptors are used for pedestrian detection [N. Dalal and B. Triggs]
- Processes in 3 steps:
  - Gradient norm
  - Gradient orientation
  - Histogram construction
- 640x480 image
- Embedded on Tegra K1
Prediction on ARM 4 cores

- High $la$
- Gradient norm: [3,4]
- Gradient orientation: [6.75, 9.5]
- Histogram: [1.25, 1.75]
- $t \in [12, 14.25] \rightarrow 79\%$
- Reality: 2200µs (12.9 ckt/pixel)
Prediction on GPU

- Only gradient norm and orientation
- Low \( I_\alpha \)
- Predicted read time: 230\( \mu s \)
- Predicted write time: 230\( \mu s \)
- Reality: 360\( \mu s \)
Example of Mapping and Execution Pipeline

**GPU**
- Managed Memory
  - **Gradient norm & orientation image** $n-1$
  - Copying image $n$ to GPU memory
  - Executing **Histogram** computation on Image $n-1$
  - Processing time

**ARM**
- Acquisition Image $n$
- Launching GPU kernel on Image $n$
- Execution time

Processing **Gradient** on Image $n$
- Copying image $n+1$ to GPU memory
- **Gradient norm & orientation image** $n$
- Executing **Histogram** computation on Image $n$

7/15/2016
romain.saussard@renault.com
Example of Mapping and Execution Pipeline

- **Histogram on ARM 4 cores:**
  - Prediction: 380μs (memory)
  - Reality: 310μs

- **Gradient on GPU:**
  - Predicted transfer time: 290μs
  - Predicted Execution time: [230, 460]μs
  - Global: [520, 750]μs
  - Reality: ~670μs

- **Global processing time limited by GPU**
Conclusion

- Our approach is able to predict execution times for different computing units

- Global methodology for embeddability:
  - Several pending patents
  - Journal paper (Real-Time Image Processing)

- Publications:
  - Towards an Automatic Prediction of Image Processing Algorithms Performances on Embedded Heterogeneous Architectures (ICPPW 2015)
  - The embeddability of lane detection algorithms on heterogeneous architectures (ICIP 2015)
  - A robust methodology for performance analysis on hybrid embedded multicore architectures (MCSOC 2016)
Thank you for your kind attention

ANY QUESTIONS?