STATUS UPDATE DES MEKONG-PROJEKTS

MODELING PERFORMANCE AND ENERGY AT COMPILATION TIME FOR IMPROVED SCHEDULING DECISIONS

Holger Fröning, Lorenz Braun, Simon Gawlok, Sotirios Nikas, Vincent Heuveline
Heidelberg University, Germany
http://www.gpumekong.org
holger.froening@ziti.uni-heidelberg.de

BMBF HPC Statuskonferenz, 08.11.2018, RRZE Erlangen
MEKONG’S BASIC IDEA

Automatically transform a single-device CUDA program into a multi-device program

- No user intervention
- Key: automated partitioning and creation of communication tasks
- Initial target: one multi-GPU node, but not limited in principle

Code analysis/code generation at compile time

- Minimize run-time overhead
- Partitioning along CTA boundaries
- => Analysis inter-CTA, not intra-CTA (e.g., no shared memory analysis)

Key for good data partitioning is memory access pattern

Received a Google Faculty Research Award in 2014
UPDATE ON COMPILER PROTOTYPE - RESULTS

Proxy app: stencil code

- No residual, manually defined number of iterations
- CUDA driver overhead omitted
- No overlap exploitation (yet)

8x NVIDIA K80

- 16 discrete GPUs total

![Hotspot Graph](image)

**Grid Size**
- 4096
- 8192
- 16384
- 23200
- 28384
- 32768

**Speedup** vs **GPUs used**
TODAY: NEED FOR PREDICTIONS

Execution time: scheduling (overlap, scalability, GPU class)
Power: power provisioning, heterogeneity (multiple GPU classes, CPUs)
Main problem: time for prediction << time for execution

Related work documents many successful approaches, most based on measured performance counters

Nice survey in [1], most recent work focuses on pre-processing and neural networks [2][3], one compile-time analytical model (limited to certain apps) [4]

Results suggest that ML techniques outperform analytical models


PERFORMANCE MODELING
CONCEPT

Compile Time

CUDA code
Computational Kernels

Compiler
Code Analysis

Executable
x86/PTX

Run Time

Launch

Command line
parameters

Inference
Predictions

Models
Device-specific

Time and power
Per-kernel

Scheduling

Execution

Input Features
Total instructions executed
Integer instructions
Load/Store instructions
Float32 instructions
Control flow instructions
Misc instructions
Grid configuration
INPUT FEATURES AND GROUND TRUTH

Input feature acquisition
- Analyze code features per code block
- Block frequency: prediction at compile time
- Note: block frequency currently done by profiling at execution time

Data set
- parboil-2.5, polybench-gpu-1.0, rodinia-3.1, shoc (selected apps)

Ground truth: performance counters and execution time via nvprof
MODEL BUILDING

Preprocessing
For each application and input data: list of kernel executions
Each kernel execution: kernel launch configuration, execution time, performance counter set, power consumption
Remove unsuitable kernels: performance counter overflows, crashes when profiling
148 samples remain

Data analysis
Total execution time: histogram shows that vast majority of kernels have less than 10% of maximum execution time
Instructions per cycle: histogram shows more uniform distribution

Measures to improve data quality
All features scaled to [0;100%], based on maximum values
Output feature total execution time scaled using log function
EARLY RESULTS - DETAIL
POWER MODELING

Resolution of power measurement is about 50ms
  Only 7 kernels run longer than that

Solutions
  Automated kernel repetition, e.g. using power profiles [1]
  Other measurement hardware (PowerMon with up to 1kHz, or plain nvprof)

Use same concept as presented before, but new output feature power

SUMMARY

Concept to model performance and power at compile time
   Code features per code block and block frequency - currently based on nvprof
   148 kernels used for training
   ANN-based inference of execution time shows promising results
   The same concept should be applicable to predict power consumption

Mekong’s first compiler prototype
   Runs for application proxies: mmult, hotspot, n-body
   Results indicate near-zero run-time overhead

Next
   Finish performance and power modeling work
   Use predictions for overlap of compute and communication tasks, and scalability predictions

http://www.gpumekong.org