Performance Analysis and Automatic Tuning of Hash Aggregation on GPUs

Viktor Rosenfeld
sebastian.bress@tu-berlin.de
Steffen Zeuch
steffen.zeuch@dfki.de
Tilmann Rabl
tilmann.rabl@hpi.de
Volker Markl
volkar.markl@tu-berlin.de

Introduction

Hash aggregation:
• Used to implement GROUP BY and DISTINCT.
• Can be significantly accelerated on GPUs.

Example:
• Query: SELECT G, sum(V) FROM R GROUP BY G

Previous work:
• Analyzed influence of parallelization strategy and thread configuration on aggregation performance.
• Formulated heuristics based on analysis of single NVIDIA Kepler GPU.

Our contributions:
1. Performance analysis on six diverse AMD and NVIDIA GPUs.
2. Automatic tuning of execution parameters at runtime.
3. Number of threads determines execution runtime.
4. Performance plateaus: Adjacent thread configurations have similar runtimes.
5. AMD GPUs exhibit high degree of runtime variation.

Thread configuration search space

• Number of threads determines execution runtime.
• Performance plateaus: Adjacent thread configurations have similar runtimes.
• AMD GPUs exhibit high degree of runtime variation.

(3) Single local minimum when we account for runtime variation.

Optimization algorithm

1. Start with initial thread configuration.
2. Follow gradient in search space to local minimum.
3. Branch search path at performance plateaus.
4. Prune slow branches.
5. Stop at minimum when there are no more branches.

(1) Heuristics to select execution parameters, which are derived from analyzing a single GPU, are not generalizable to other GPUs.

(2) We can optimize execution parameters by following the gradient in the search space and branching the search at performance plateaus.

Main takeaways

Parallelization strategies

(1) INDEPENDENT aggregation is not competitive on newer GPUs which implement fast atomic on local memory.
(2) Aggregation throughput limited by global GPU memory latency (and not transfer bandwidth) when hash table exceeds L2 cache.
(3) Work items per work group
Formulated heuristics based on analysis of single NVIDIA Kepler GPU.

Performance penalty

A configuration optimized for a specific GPU is executed on another GPU.

(4) The optimal thread configuration is highly GPU dependent.
(5) Suboptimal configurations lead to large performance penalties.

Evaluation

Quality of found execution parameters

Branch at performance plateaus

Optimization costs

Optimization effort

0% 10% 20% 30% 40% Fraction of full evaluation
0 100 200 300 400 Number of blocks