gem5 GPU Accuracy Profiler (GAP)

Charles d'Andrea Jamieson, Anushka Chandashekar, Ian McDougall, Matthew D. Sinclair
University of Wisconsin-Madison, AMD Research
sinclair@cs.wisc.edu
Prior CPU-GPU Support in gem5

- Execution-driven, cycle-level
  - Accurately models complex CPUs & GPUs
  - Rapid prototyping of new features
  - Validate simulation with execute-in-execute

- Runs ROCm 4.0 user stack
- Simulates HIP applications
Improving Register Allocation Support

- Simple dependence tracking – only 1 wavefront/CU at a time
  - Even if sufficient registers are available for more WFs
- Issue: unrealistic relative to real GPUs
- Solution: add dynamic register allocator [Bruce et al. ISPASS ‘20]
  - If enough registers available, schedule additional WFs concurrently/CU
  - Potentially can utilize all WF slots depending on register requirements
  - More complex, higher performance designs possible

Intuition: Dynamic allocator significantly improves accuracy
Dynamic Register Allocator Performance

![Graph showing normalized speedup comparison between static and dynamic register allocation across various benchmarks.]

Reality: dynamic register allocator 6% worse than simple – why?
Issue: Dependence Tracking

- GPU model did not track dependencies well → many stalls
  - Result: optimizing register allocation in isolation was insufficient
- Issue: Proprietary GPU dependence checking sols unknown
- Solution: simple, in-order scoreboard
  - Bit per register to track use status
  - Cleared on instruction completion
  - Checks for RAW/WAW hazards

Result: up to 44% reduction in stalls
How Does GAP Work?

- **Issue:** need standard approach for evaluating new configurations
- **Solution:** gem5 GPU Accuracy Profiler (GAP)

Using GAP to iteratively refine gem5 GPU models
How to Run GAP?

• Need three things:
  1. Executables to run and their input args (for both GPU & gem5)
  2. gem5 settings (i.e., input args to simulator)
  3. Metrics to collect from real GPU and gem5

• 1 & 2: configuration input file; 3: separate metrics input file

• To run: python3 gap.py -c <configFile>
Example Setup (Vega 20 GPU)

• Example configuration file (1/2)
  • Tells GAP what benchmark(s) to run

  # List of executables to collect metrics from
  # Include any executable specific arguments

  [EXECUTABLES]
  bench1 = gem5-resources/src/gpu/square/bin/square
  bench2 = BabelStream/hip-stream -n 2 -s 16777216 --triad-only
  ...
Example Setup (Vega 20 GPU)

Example configuration file (2/2)

• Settings: tell gem5 what and where to run

```
[SETTINGS]

gem5 = /path/to/gem5  # path to the directory containing gem5
output_dir = ./outdir  # output destination directory
docker_image = gcr.io/gem5-test/gcn-gpu:v21-1  # docker image used for gem5
gem5_script = gem5/configs/example/apu_se.py  # gem5 python configuration file

# Any flags input to the gem5 config file (optional)
script_flags = --num-compute-units=60 --cu-per-sa=15 --num-gpu-complex=4 --reg-alloc-policy=dynamic --barriers-per-cu=16 --num-tccs=8 --bw-scalor=8 --num-dirs=64 --mem-size=16GB --mem-type=HBM_1000_4H_1x64 --vreg-file-size=16384 --sreg-file-size=800

# rocprof input file (specifies profiler behavior and which metrics to collect)
rocprof_metrics = metric.txt
```
Example Setup (Vega 20 GPU)

• Example GPU metrics file

```
pmc : FetchSize VALUInsts Wavefronts FlatVMemInsts
pmc : WriteSize SALUInsts LDSInsts VWriteInsts
pmc : MemUnitBussy MemUnitStalled TCC_HIT_sum LDSBankConflict
pmc : LDSInsts TCC_MISS_sum VALUUtilization
```

• “pmc”: tells rocprof which HW counters to collect per run
  • May need multiple lines because limited HW counters on GPU
  • rocprof will run application once per “pmc” line on real GPU
Example Results (Vega 20 GPU) (1/2)

- Example Output file (for square):

  Metric, rocprof measurement, gem5 measurement, Absolute difference, % Difference
  VALUInsts,120,120,0,0.0
  FlatVMemInsts,15,268,253,1686.6666666666667
  Wavefronts,2048,2048,0,0.0
  SALUInsts,19,19,0,0.0
  LDSInsts,0,0,0,0
  VWriteInsts,0,0,0,0
  LDSBankConflict,0,0,0,0
  TCC_HIT_sum,184,3002034,3001850,1631440.2173913044
  LDSInsts,0,0,0,0
  VALUUtilization,99,100,1,1.0101010101010102
  TCC_MISS_sum,63401,583906,520505,820.9728553177393
Stats well correlated (<= 1% difference) for vector ALU stats, but poor for memory

Need to isolate stats for specific components to properly fix this
Next Step: Additional Microbenchmarks

• Goal: isolate behavior of different components
  • Via GAP, can refine latencies, bandwidths, etc.

• Current tests (handwritten HIP assembly kernels)
  • L1 I$ size & latency
  • L1 D$ size, latency, & bandwidth
  • LDS (scratchpad) latency & bandwidth
  • L2 $ latency & bandwidth
  • Main memory latency & bandwidth
  • GPU STREAM peak bandwidth
  • max FLOPs, Arithmetic latency for various operations, …
Conclusions & Future Work

• Having validated gem5 models is important
  • Existing GPU model does not always behave intuitively
  • Point solutions insufficient

• Solution: More automated framework (GAP)
  • Results: memory system seems to need the most attention
  • Potentially can be applied to other, non-GPU models

• Goals:
  • Use microbenchmarks to tune for minimum absolute error in GPU model
  • Release tool (& GPU improvements) publicly
  • Performance regression testing integrated into gem5
Backup
Frequently Asked Questions (1/2)

- What do I need to use GAP?
  - A computer with a (supported in gem5) AMD GPU

- What mode(s) work with GAP?
  - Currently only SE mode support

- How does GAP run things on hardware?
  - Uses rocprof (AMD’s GPU profiling tool) to run application(s)
    - User must specify metric(s) they want to compare (input file to GAP)
    - Can add a new metric (hardware counter) by updating input file
  - Compares hardware counter results to gem5 stats
    - We have created a 1-1 mapping between GPU stats and gem5 stats
Frequently Asked Questions (2/2)

• What is output format?
  • CSV

• What if gem5 has a metric rocprof doesn’t (or vice-versa)?
  • GAP cannot support these at the moment
  • Could gather results, but could not compare them
More GAP Setup Info

- To run GAP user must first specify a few items for gem5
  - Must be placed under the [SETTINGS] header
  - All fields are required unless specified.

<table>
<thead>
<tr>
<th>Config File Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gem5</td>
<td>Absolute path to gem5 directory</td>
</tr>
<tr>
<td>output_dir (optional)</td>
<td>Relative path to directory to put output in</td>
</tr>
<tr>
<td>docker_image (optional)</td>
<td>Docker image to run (e.g., with gem5 GPU model)</td>
</tr>
<tr>
<td>gem5_config_script</td>
<td>Path to gem5 Python configuration script</td>
</tr>
<tr>
<td>script_flags</td>
<td>Flags to pass into gem5 configuration script</td>
</tr>
<tr>
<td>rocprof_metrics</td>
<td>File that specifies metrics rocprof should collect</td>
</tr>
<tr>
<td>rocprof_flags (optional)</td>
<td>Flags to pass into rocprof</td>
</tr>
</tbody>
</table>