GPrioSwap: Towards a Swapping Policy for GPUs

Jens Kehne, Jonathan Metter, Martin Merkel, Marius Hillenbrand, Marc Rittinghaus, Frank Bellosa
10th ACM International Systems and Storage Conference
Motivation

- GPUs are widespread in computing
  - Unprecedented performance for some applications
  - Very energy efficient

- GPUs are moving to the cloud
  - Cost effective through oversubscription

- Can safely share computational power
  - Even have fairness to some degree

- But what about memory?
Our Approach: GPUSwap (VEE '15)
Our Approach: GPUSwap (VEE '15)
Our Approach: GPUSwap (VEE '15)

Achieves both fairness and good utilization
GPUSwap: Swapping Policy

- Choose app with most GPU memory ("The Victim")
  - Achieves fairness
- Choose chunk of memory from victim’s AS

- How do we find the right chunk?
- No reference bit on current GPUs!

- Original implementation: Random
Results: Runtime Overhead (lud)

![Graph showing runtime overhead vs. GPU memory]
Analysis: Methodology

- No easy way to count page accesses
  - No reference bit
  - No page faults
- Performance counters only count entire application

Idea: Separate single page from rest of AS

- Accurate access count for each page
- Must rerun application once per page
Analysis: Results (bfs)

- Large variance between application buffers
- Little variance within each buffer
- Not shown: Large stack buffer, close to zero accesses
- Similar results for other applications

➤ Finding the right buffer to swap is probably enough
GPriorSwap

- Operates in two steps

  - Offline step
    - Profile application
    - Assign a priority to each buffer

  - Online step (on memory pressure)
    - Find set of chunks with lowest priority from victim‘s AS
    - Select one chunk from set at random
Offline Step

- Profile application as before
  - Re-run once per buffer rather than per page
- Calculate avg. number of accesses per page
- Assign buffer priorities based on averages

- Pass priorities as parameter during allocation
  - Requires changes to application code
Swapping Policy (Online Step)

- Select victim (application with most GPU memory)

![PRIORITY LIST]

- Find all chunks with lowest priority
Swapping Policy (Online Step)

- Select victim (application with most GPU memory)

```
1 5 3 2 5 4 2 3 1 5 2 1 4
```

- Find all chunks with lowest priority
- Select one low-priority chunk at random

- Repeat until enough chunks have been selected
Swapping Policy (Online Step)

- Select victim (application with most GPU memory)

1 5 3 2 5 4 2 3

5 2 1 4

- Find all chunks with lowest priority
- Select one low-priority chunk at random

- Repeat until enough chunks have been selected
- Swap all selected chunks
- Service allocation request
Results: Backprop

![Graph showing runtime vs GPU memory for different swapping policies (Random, Best random, Prio).](image-url)
Results: Heartwall

- 2 out of 9 applications:

![Graph showing performance comparison between Random, Best random, and Prio for GPU Memory vs. Runtime]
Conclusion

- We can efficiently swap GPU data at runtime
- But we do not yet know what to swap

- Importance of pages varies by buffer
- Profile applications, assign buffer priorities
- Swap from low-priority buffers first