UNIVERSITY OF CAMBRIDGE

The case for limited-preemptive scheduling in GPUs for real-time systems

Roy Spliet Robert Mullins (first.last@cst.cam.ac.uk)

Department of Computer Science and Technology
University of Cambridge

GPUs in real-time systems

Work complementary to Amert et al.[1]:

- Prior work: scheduling tasks within single context (mainly).
- This work: scheduling properties of different HW contexts.

[1] Amert, T., Otternes, N., Anderson, J.H., Smith, F. D., GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed, December 2017

Outline

Background: context switch mechanisms and preemption models

Experiment: Measure context switch response times

Experiment: Task scheduling on GPUs

Conclusion

Context switch mechanisms and preemption models

Kernel invocation

clProcessImage: $1920 * 1080 = 2{,}073{,}600$ threads

[2] Tanasic et al. "Enabling preemptive multiprogramming on GPUs, 2014"

Context switch mechanisms and preemption models

Kernel invocation

clProcessImage: $1920 * 1080 = 2{,}073{,}600$ threads

Non-preemptive scheduling (current GPUs):

- Finish whole kernel.
- Max blocking: WCRT of kernel.
- Swap: HW+OpenCL configuration.

[2] Tanasic et al. "Enabling preemptive multiprogramming on GPUs, 2014"

Context switch mechanisms and preemption models

Kernel invocation

clProcessImage: 1920 $*$ 1080 $=$ 2,073,600 threads

Non-preemptive scheduling (current GPUs):

- Finish whole kernel.
- Max blocking: WCRT of kernel.
- Swap: HW+OpenCL configuration.

Preemptive scheduling[2]:

- Interrupt anywhere.
- Max blocking: none.
- Swap: HW+OpenCL configuration, register files, local memory.

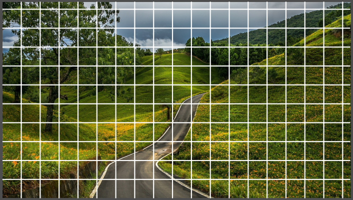
[2] Tanasic et al. "Enabling preemptive multiprogramming on GPUs, 2014"

Context switch mechanisms and preemption models

Kernel invocation

clProcessImage: $1920 * 1080 = 2{,}073{,}600$ threads

$2073600/1024 = 2025$ work-groups



Non-preemptive scheduling (current GPUs):

- Finish whole kernel.
- Max blocking: WCRT of kernel.
- Swap: HW+OpenCL configuration.

Preemptive scheduling[2]:

- Interrupt anywhere.
- Max blocking: none.
- Swap: HW+OpenCL configuration, register files, local memory.

Limited-preemptive scheduling:

- Interrupt on work-group boundary ("SM draining"[2]).
- Max blocking: ∼WCRT of work-group.
- Swap: HW+OpenCL configuration.

[2] Tanasic et al. "Enabling preemptive multiprogramming on GPUs, 2014"

Context switch mechanisms and preemption models

Our claim: SM draining, modelled by limited-preemptive scheduling, provides a good trade-off point for GPUs between:

- Context switching cost, and
- WCRT benefits.

Measure context switch response time

"The Fermi pipeline is optimized to reduce the cost of an application context switch to **below 25 microseconds***."*[3]

[3] C.M. Wittenbrink, E. Kilgariff, and A. Prabhu. Fermi GF100 GPU Architecture. Micro, IEEE 2011

Measure context switch response time

"The Fermi pipeline is optimized to reduce the cost of an application context switch to **below 25 microseconds**.*"*[3]

- Is 25μs an average or worst-case time?
- Is 25μs execution time or response time?
- What is the distribution?

[3] C.M. Wittenbrink, E. Kilgariff, and A. Prabhu. Fermi GF100 GPU Architecture. Micro, IEEE 2011

Measure context switch response time - experiment

Characterise WCRT of hardware (non-preemptive) context switch.

Approach:

1. Modify (nouveau's) context switching firmware to report WCRT.
    - **Excluding** time to finish current kernel execution.
    - Intrusive measurement, max. observed overhead 224ns.

Measure context switch response time - experiment

Characterise WCRT of hardware (non-preemptive) context switch.

Approach:

1. Modify (nouveau's) context switching firmware to report WCRT.
   - **Excluding** time to finish current kernel execution.
   - Intrusive measurement, max. observed overhead 224ns.
2. Write program to read from hardware:
   - Context size,
   - Reported context switch time.

Measure context switch response time - experiment

Characterise WCRT of hardware (non-preemptive) context switch.

Approach:

1. Modify (nouveau's) context switching firmware to report WCRT.
   - **Excluding** time to finish current kernel execution.
   - Intrusive measurement, max. observed overhead 224ns.
2. Write program to read from hardware:
   - Context size,
   - Reported context switch time.
3. For several Kepler GPUs (2012-2014) gather 20M samples each.
   - 1600x1200 X.org/XFCE desktop,
   - 1024x768 OpenArena windowed timedemo.

Measure context switch response time - results

NVIDIA GeForce	SM	Cores MHz	Max bw GiB/s	State KiB	Time (μs) Min	Avg	Max	Avg. bw GiB/s	Util.
GT 710	1	953	14.4	63.9	9.2	21.5	80.1	2.83	19.6%
GT 640	2	901	28.5	68.2	13.6	26.5	43.7	2.45	8.6%
GTX 650	2	1058	80.0	68.2	12.7	23.2	36.0	2.71	3.4%
GTX 780	12	992	288.4	268.6	9.7	20.0	28.6	13.76	4.8%

- What is the average context switch time? $20.0 - 26.5\mu$s.
- What is the worst-case context switch time? $> 28.6\mu$s.

Measure context switch response time - results

NVIDIA GeForce	SM	Cores MHz	Max bw GiB/s	State KiB	Time (μs) Min	Avg	Max	Avg. bw GiB/s	Util.
GT 710	1	953	14.4	63.9	9.2	21.5	80.1	2.83	19.6%
GT 640	2	901	28.5	68.2	13.6	26.5	43.7	2.45	8.6%
GTX 650	2	1058	80.0	68.2	12.7	23.2	36.0	2.71	3.4%
GTX 780	12	992	288.4	268.6	9.7	20.0	28.6	13.76	4.8%

- What is the average context switch time? $20.0 - 26.5\mu s$.
- What is the worst-case context switch time? $> 28.6\mu s$.
- Execution time or response time?
  - Ex. time: Average context switch time not strictly memory bound.

Measure context switch response time - results

NVIDIA GeForce	SM	Cores MHz	Max bw GiB/s	State KiB	Time (μs) Min	Avg	Max	Avg. bw GiB/s	Util.
GT 710	1	953	14.4	63.9	9.2	21.5	80.1	2.83	19.6%
GT 640	2	901	28.5	68.2	13.6	26.5	43.7	2.45	8.6%
GTX 650	2	1058	80.0	68.2	12.7	23.2	36.0	2.71	3.4%
GTX 780	12	992	288.4	268.6	9.7	20.0	28.6	13.76	4.8%

- What is the average context switch time? $20.0 - 26.5\mu s$.
- What is the worst-case context switch time? $> 28.6\mu s$.
- Execution time or response time?
  - Ex. time: Average context switch time not strictly memory bound.
  - Resp. time: Worst case overhead due to interference on DRAM bus from display scan-out.

Measure context switch response time - results

NVIDIA GeForce	SM	Cores MHz	Max bw GiB/s	State KiB	Time (μs) Min	Avg	Max	Avg. bw GiB/s	Util.
GT 710	1	953	14.4	63.9	9.2	21.5	80.1	2.83	19.6%
GT 640	2	901	28.5	68.2	13.6	26.5	43.7	2.45	8.6%
GTX 650	2	1058	80.0	68.2	12.7	23.2	36.0	2.71	3.4%
GTX 780	12	992	288.4	268.6	9.7	20.0	28.6	13.76	4.8%

- What is the average context switch time? $20.0 - 26.5\mu s$.
- What is the worst-case context switch time? $> 28.6\mu s$.
- Execution time or response time?
  - Ex. time: Average context switch time not strictly memory bound.
  - Resp. time: Worst case overhead due to interference on DRAM bus from display scan-out.
- Distribution (GT 710): 0.3% of samples in $[23.6, \infty]$.
  - see paper for plot.

Preemption models

Our claim: SM draining, modelled by limited-preemptive scheduling, provides a good trade-off point for GPUs between:

- √ Context switching cost, and
- WCRT benefits.

Task scheduling on GPUs - experiment

Study WCRT implications of scheduling models under context switching constraints, through overhead-aware schedulability experiment.

Approach:

1. Determine feasible parameters/ranges for
   - Context switch overheads for different scheduling policies,
   - (Periodic) task sets.
2. Compare schedulability of random task sets.

Task scheduling on GPUs - parameters

Scheduling policy	State (KiB) Ctx	Reg	Local	Total	Time (μs) Avg	Max	Preempt /job[4]
Full preemptive (EDF)	68.2	512	96	676.2	263	434	×2
SM draining (lpEDF)	68.2	0	0	68.2	27	44	×2
Non-preemptive (npEDF)	68.2	0	0	68.2	27	44	×1

(Based on GeForce GT 640 (2×SM), resembling Tegra K1)

[4] A. Burns, K. Tindell, and A. Wellings. Effective analysis for engineering real-time fixed priority schedulers. IEEE Trans. on Software Engineering, May 1995.

Task scheduling on GPUs - parameters

Scheduling policy	State (KiB) Ctx	Reg	Local	Total	Time (μs) Avg	Max	Preempt /job[4]
Full preemptive (EDF)	68.2	512	96	676.2	263	434	×2
SM draining (lpEDF)	68.2	0	0	68.2	27	44	×2
Non-preemptive (npEDF)	68.2	0	0	68.2	27	44	×1

(Based on GeForce GT 640 (2×SM), resembling Tegra K1)

Linear correlation state size ↔ context switch time

[4] A. Burns, K. Tindell, and A. Wellings. Effective analysis for engineering real-time fixed priority schedulers. IEEE Trans. on Software Engineering, May 1995.

UNIVERSITY OF CAMBRIDGE

Task scheduling on GPUs - parameters

Scheduling policy	State (KiB) Ctx	Reg	Local	Total	Time (μs) Avg	Max	Preempt /job[4]
Full preemptive (EDF)	68.2	512	96	676.2	263	434	×2
SM draining (lpEDF)	68.2	0	0	68.2	27	44	×2
Non-preemptive (npEDF)	68.2	0	0	68.2	27	44	×1

(Based on GeForce GT 640 (2×SM), resembling Tegra K1)

Linear correlation state size ↔ context switch time

Inflate task cost with n × context switch time

[4] A. Burns, K. Tindell, and A. Wellings. Effective analysis for engineering real-time fixed priority schedulers. IEEE Trans. on Software Engineering, May 1995.

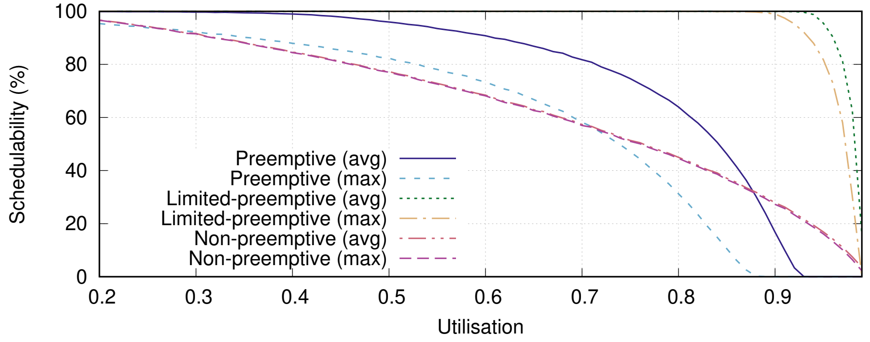
Preemptive GPU scheduling

Compare schedulability of random task sets:

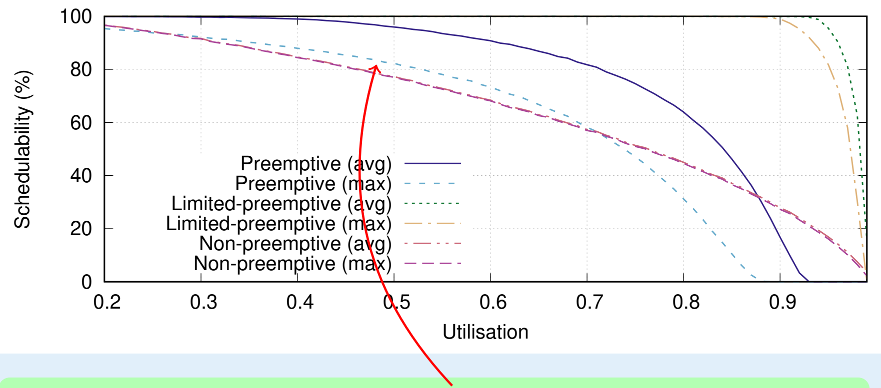
Task set:

- Uniprocessor EDF scheduling policy.
- $U = \{0.2, 0.21, \ldots, 1.0\}$
- 100,000∗81 = 8.1M random task sets (UUniFast).
- Task set: two tasks, 1,000μs $\leqslant P_i <$15,000μs.
- lpEDF: max blocking $q = \frac{c}{random(2{,}500)}$, 2-500 WGs per SM.

Preemptive GPU scheduling



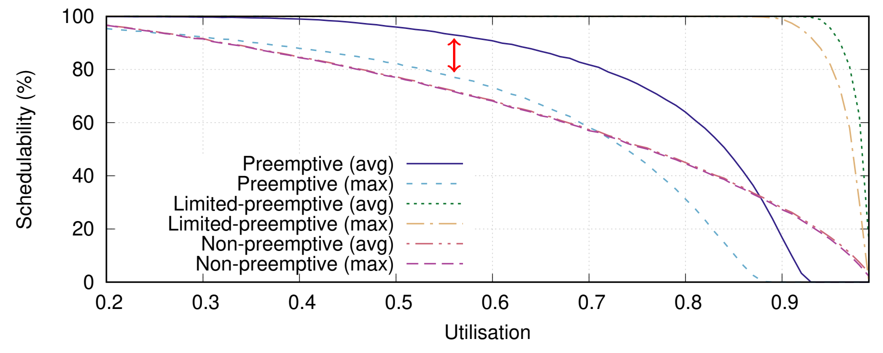
Preemptive GPU scheduling



For $0.25 \leqslant U \leqslant 0.72$ full-preempt beneficial

UNIVERSITY OF CAMBRIDGE

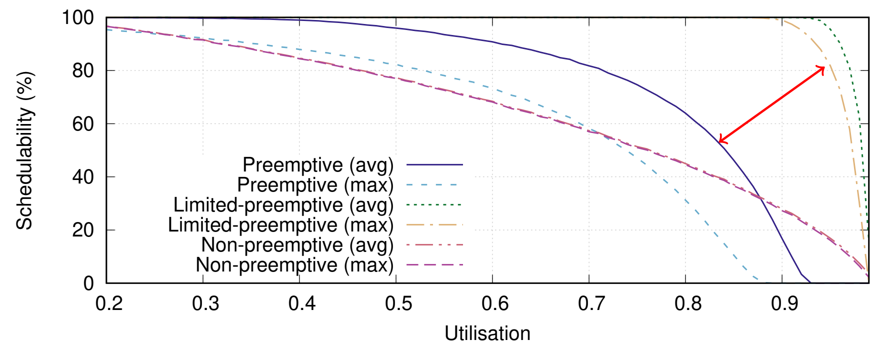
Preemptive GPU scheduling



For $0.25 \leqslant U \leqslant 0.72$ full-preempt beneficial

Reduce preemptive ctxswitch overhead → higher schedulability.

Preemptive GPU scheduling



Limited-preemption far outperforms other models!

UNIVERSITY OF CAMBRIDGE

Conclusion

Limited-preemptive scheduling (SM draining) provides a good trade-off point for GPUs between context switching cost and WCRT benefits.

- Current GPUs: context switch $20 - 26.5\mu s$ on average.
- Overhead-aware schedulability experiment demonstrates advantage of SM draining model.

Conclusion

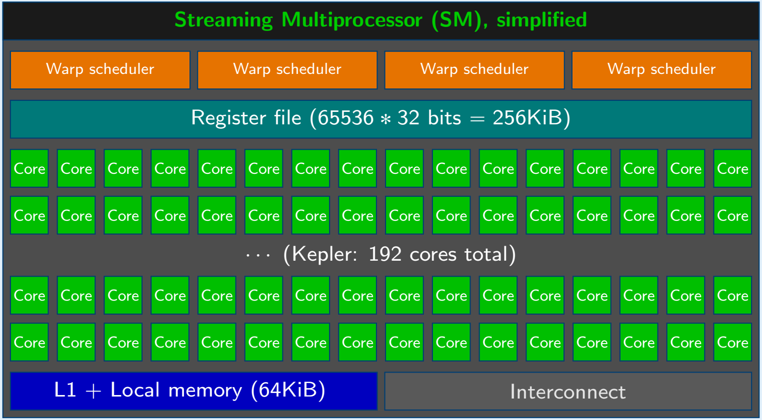
Limited-preemptive scheduling (SM draining) provides a good trade-off point for GPUs between context switching cost and WCRT benefits.

- Current GPUs: context switch $20 - 26.5\mu s$ on average.
- Overhead-aware schedulability experiment demonstrates advantage of SM draining model.

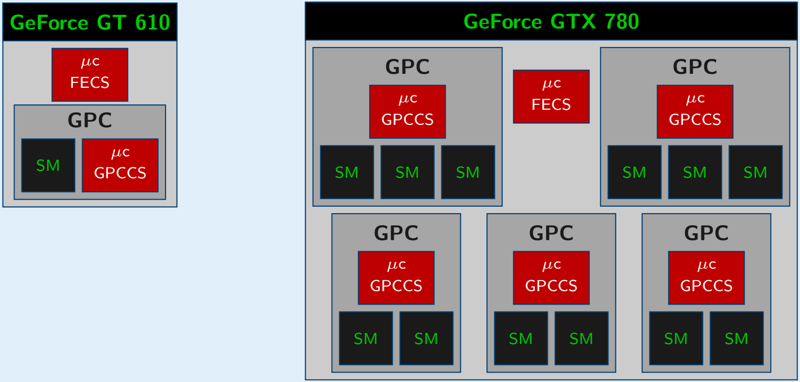
In the paper:

- Histogram of context switch times GeForce GT 710.
- Demonstration of interference context switch ↔ scan-out.
- Schedulability experiment with 3-task systems.

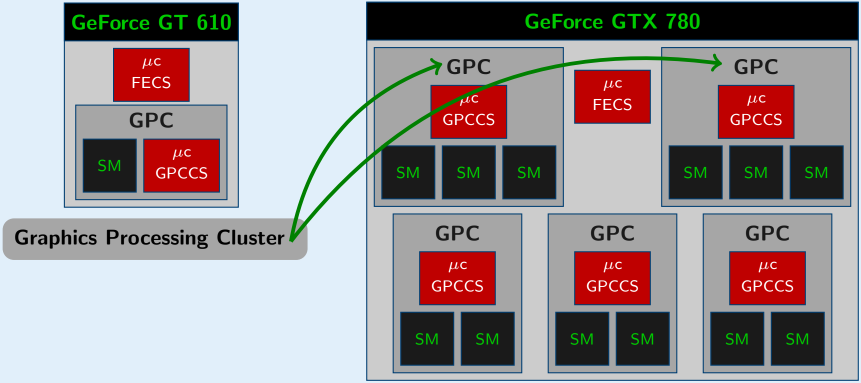
NVIDIA GPU architecture - streaming multiprocessor



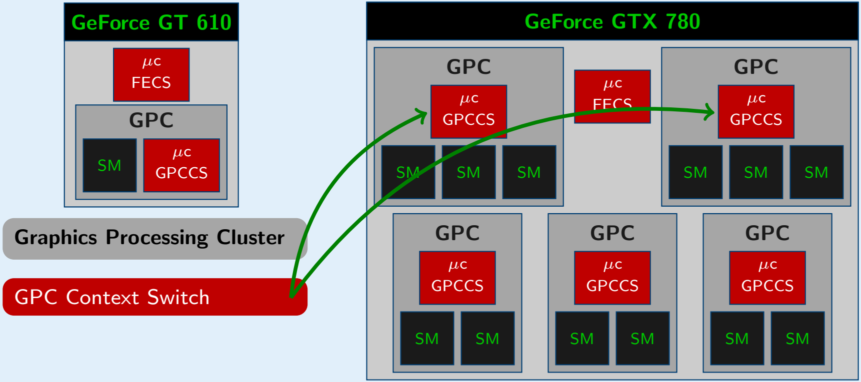
NVIDIA GPU architecture - FECS and GPCCS



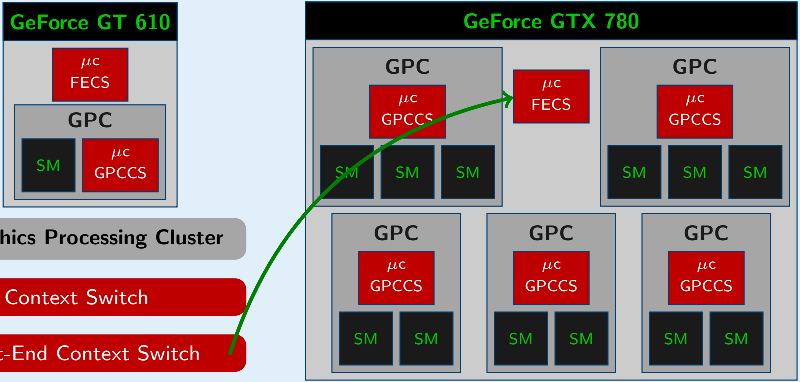
NVIDIA GPU architecture - FECS and GPCCS



NVIDIA GPU architecture - FECS and GPCCS



NVIDIA GPU architecture - FECS and GPCCS



Graphics Processing Cluster

GPC Context Switch

Front-End Context Switch