2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Porto, Portugal

MERLOT: Architectural Support for Energy-Efficient Real-time Processing in GPUs

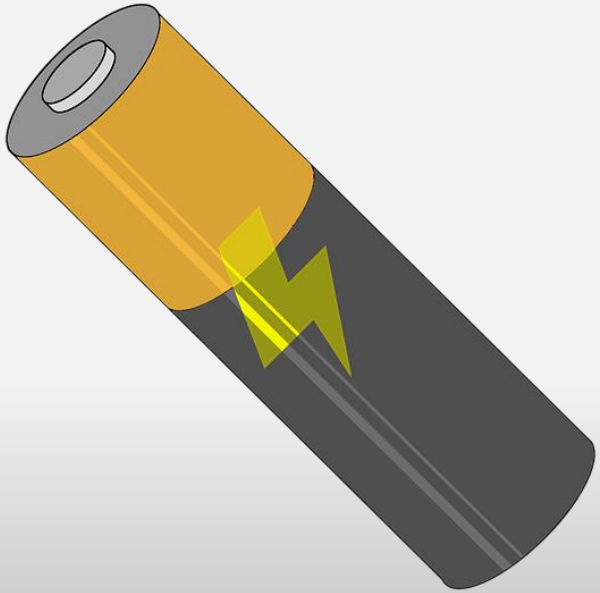
Muhammad Santriaji
santriaji@uchicago.edu

Henry Hoffmann
hankhoffmann@cs.uchicago.edu

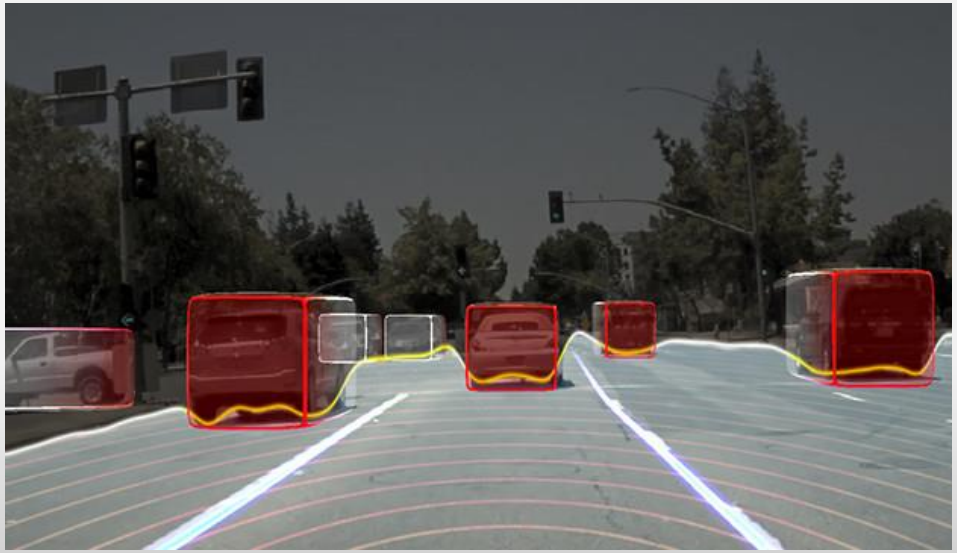
GPU

Importance

Problem:



High power

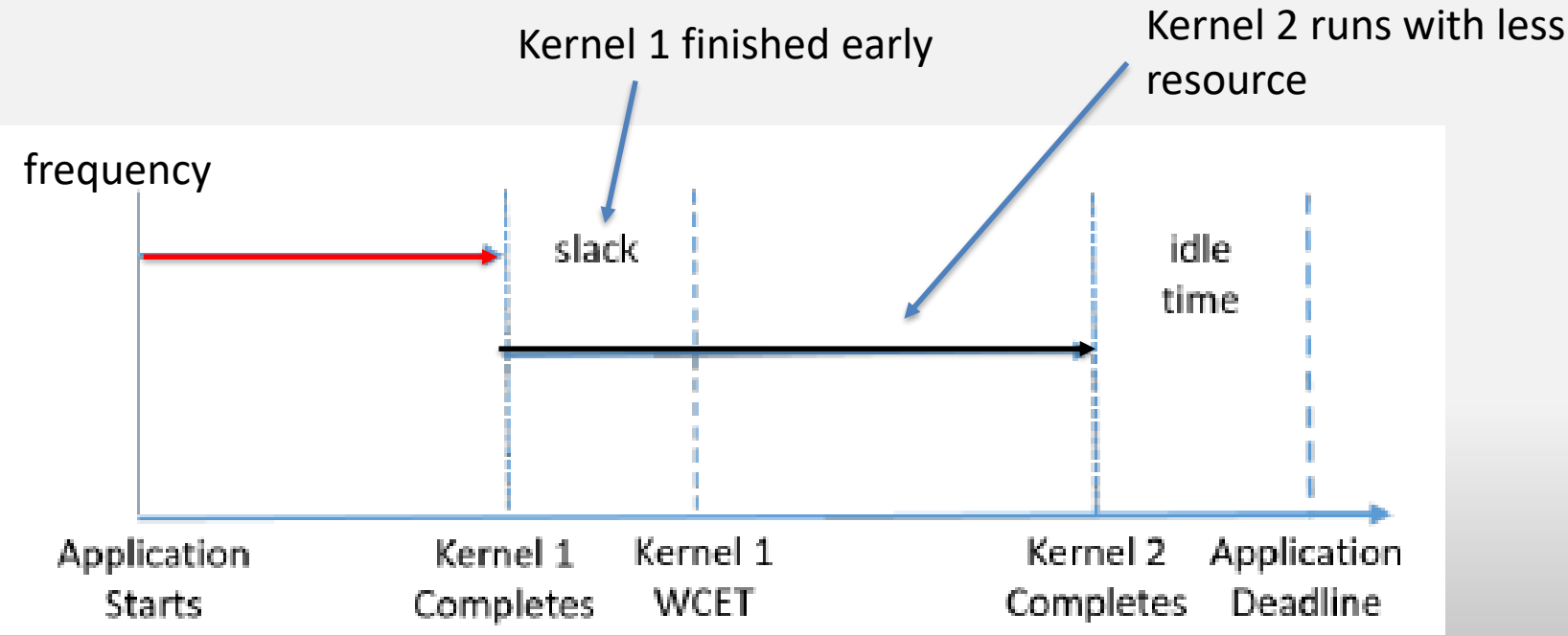


Safety critical application
needs Hard Real Time Guarantee

Challenges in GPU:

1. Reduce Energy.
2. Hard real time.

Slack Transfer

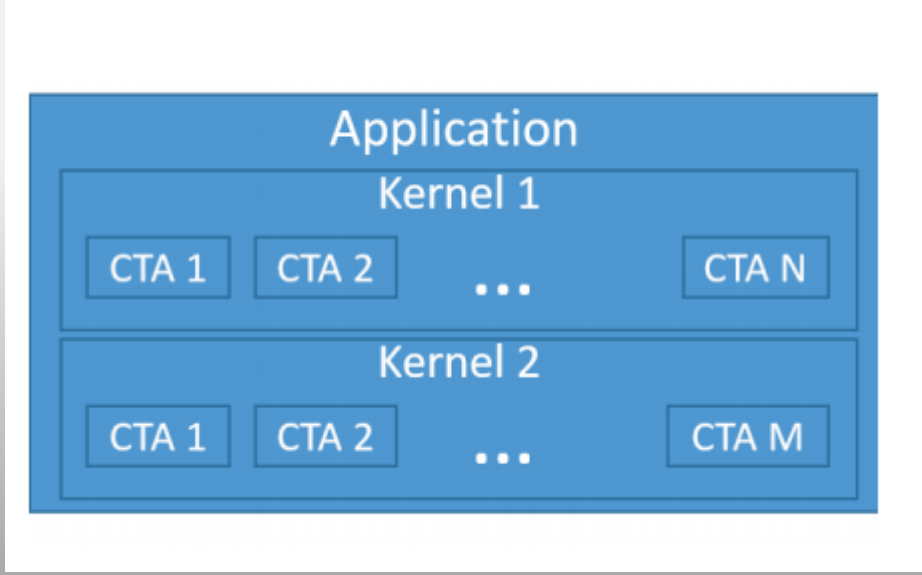


Slack Transfer Problem in Software

- Checkpoints between kernels.
- GPU Application characteristics:
  - Long kernel length.
  - Not enough kernel.
- To reduce memory copy overhead.

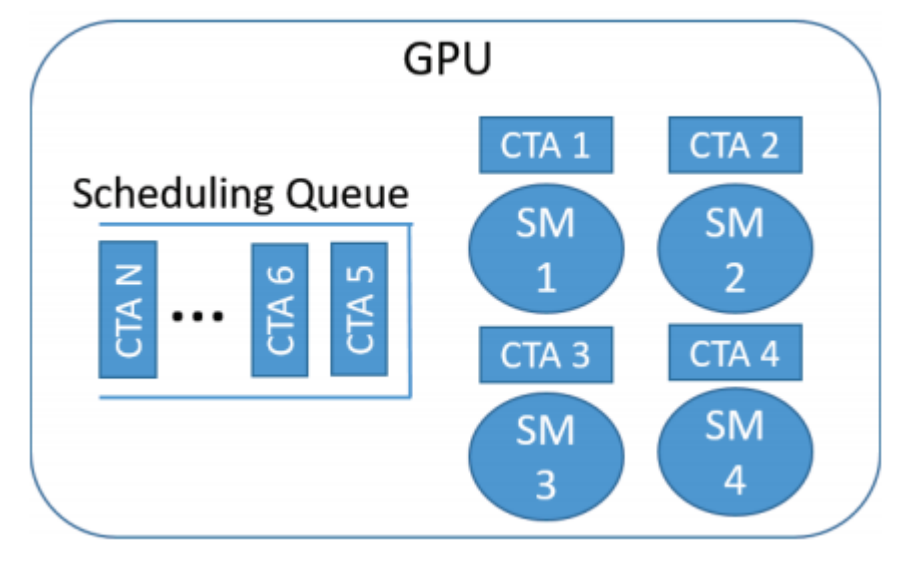
GPU Application

- GPU Applications consist of kernels.
- Kernels consists of CTAs.
- CTA consists of threads.



GPU Application

- CTA is queued because lacks of resources.
- CTA can be used as checkpoints.



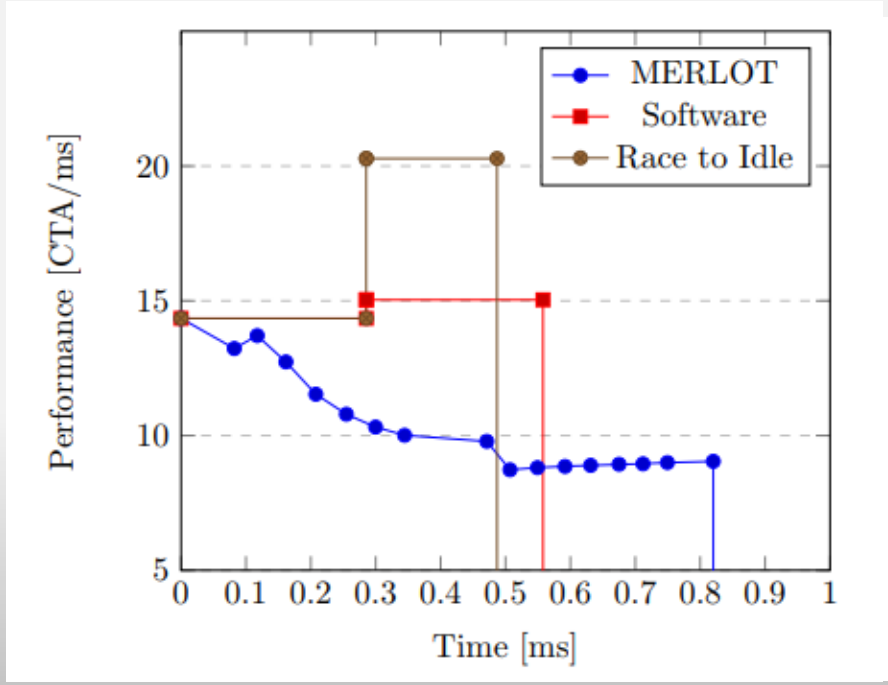
Uniqueness in GPU

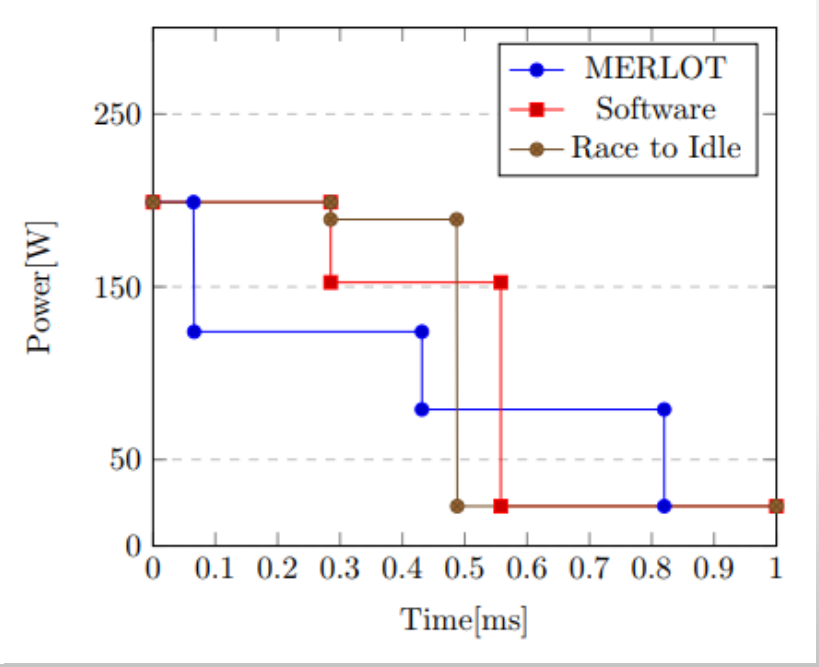
- Hard real time needs checkpoints.
- CTA can be used as checkpoints.
  - A lot of them.
  - Without editing the GPU code.

Scenario

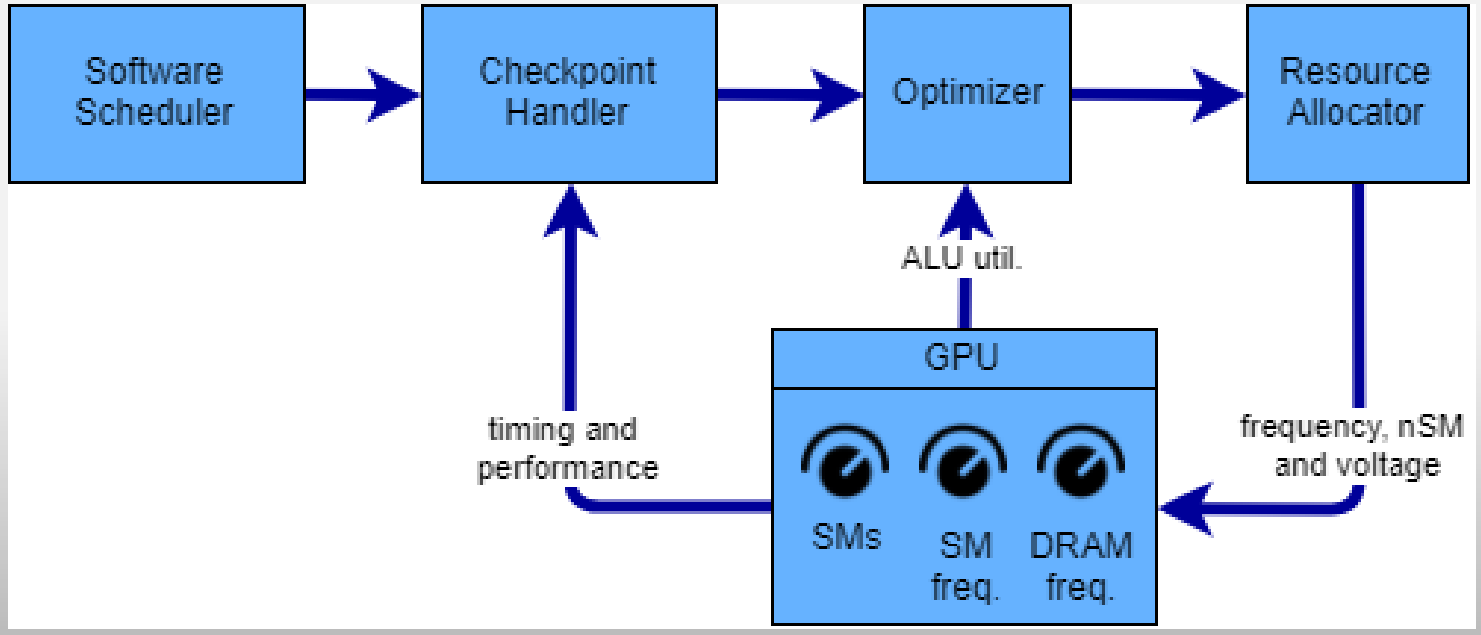
- User gives information about
  - Number of CTA.
  - WCET corresponds to the time to finish the CTA.
- MERLOT configure the resources such that :
  - Energy is decreased
  - Hard real time is guaranteed

Example

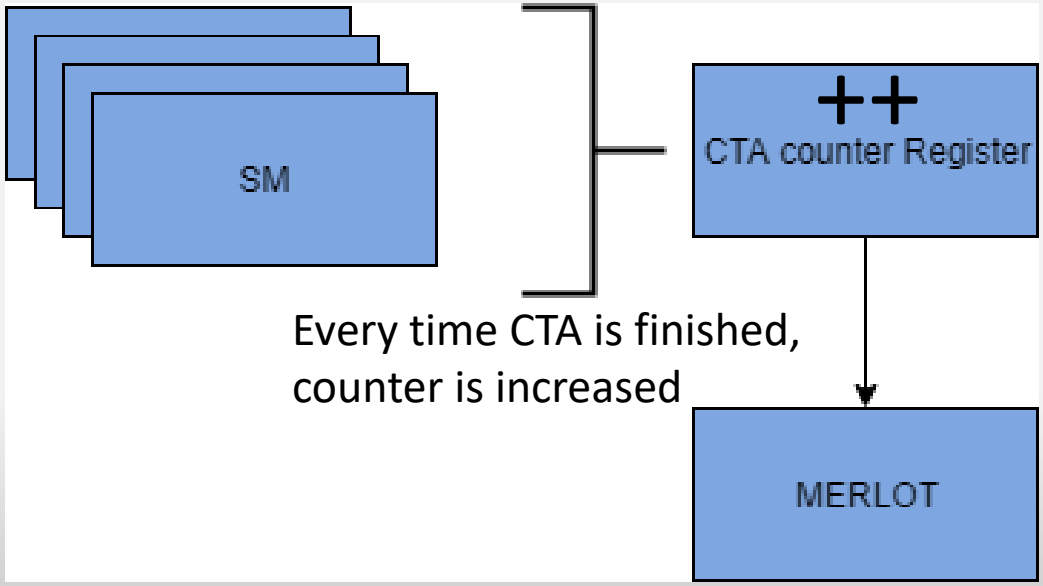

MERLOT
Software
Race to Idle
Performance [CTA/ms]
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time [ms]


MERLOT
Software
Race to Idle
Power [W]
250
150
50
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time[ms]

MERLOT Design



Hardware extension needed



Approach

- Slack transfer in every CTA checkpoint.

procedure CHECKPOINT HANDLER
  if $nCTA_j = CTA_{current}$ **then**
    $t_{ahead} = wCTA_j - t_{current}$
    $t_{remaining} = wCTA_{j+1} - wCTA_j$
    $\eta = \frac{t_{remaining+o}}{t_{remaining}+t_{ahead}+o}$
    $\lambda = \lceil f_{max} \cdot \eta \rceil$
    $j++$

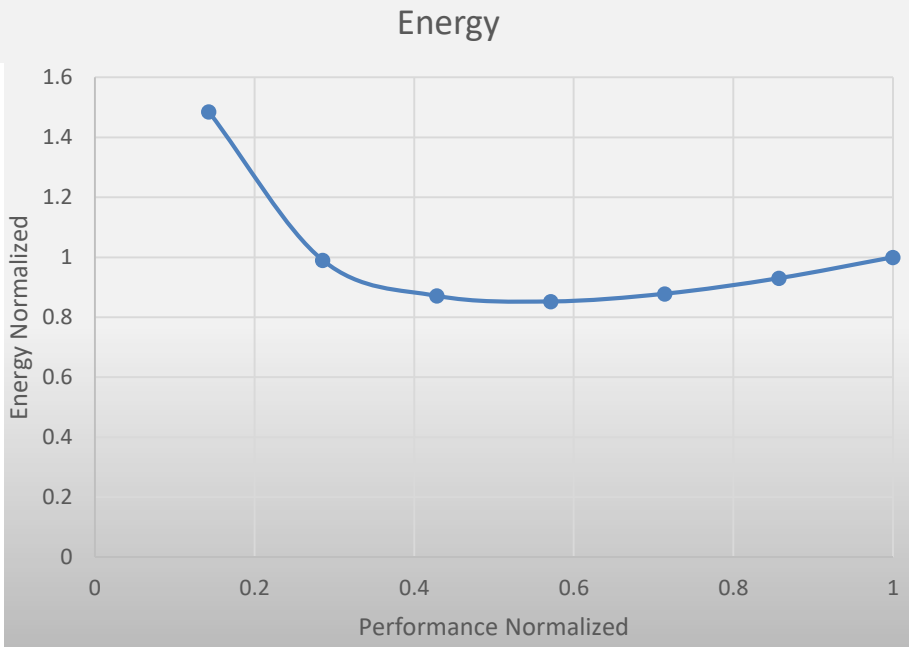
η = slowdown from slack transfer

λ = frequency derived from slowdown

Approach

- Limiting the slowdown (f critical)

```
if λ < f_crit then
    if ALUutilization < th_ALU then
        f_SM ← f_crit − 1
        f_mem ← f_crit
    else
        f_SM ← f_crit
        f_mem ← f_crit − 1
else
    if ALUutilization < th_ALU then
        f_SM ← λ
        f_mem ← λ + 1
    else
        f_SM ← λ
        f_mem ← λ
```



Approach

- Monitoring the workload (ALU utilization)

```
if λ < f_crit then
    if ALUutilization < th_ALU then
        f_SM ← f_crit − 1
        f_mem ← f_crit
    else
        f_SM ← f_crit
        f_mem ← f_crit − 1
else
    if ALUutilization < th_ALU then
        f_SM ← λ
        f_mem ← λ + 1
    else
        f_SM ← λ
        f_mem ← λ
```

Decreased fcrit based on workload.
Compute workload : lower memory frequency
Memory workload : lower SM frequency

Guarantee the hard real time because frequency always equal or above λ

Approach

- Cutoff unused resources

```
nSM = 0
for all sm in the GPU do
    if sm == active then
        nSM = nSM + 1
```

Implementation

- GPGPU-Sim
  - GPU simulator.
  - GTX480.
- GPU-Wattch
  - Acquire power
- FPGA
  - Show it is feasible
  - Get overhead
    - 0.151 microseconds.
    - 0.299 Watts.

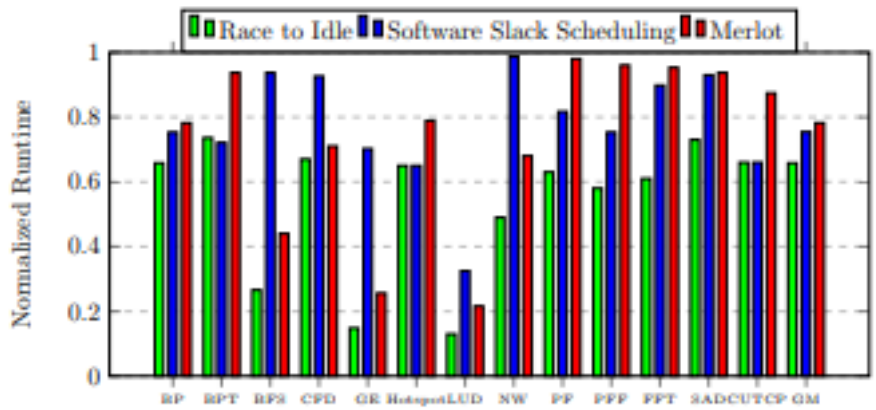
Simulation Setup

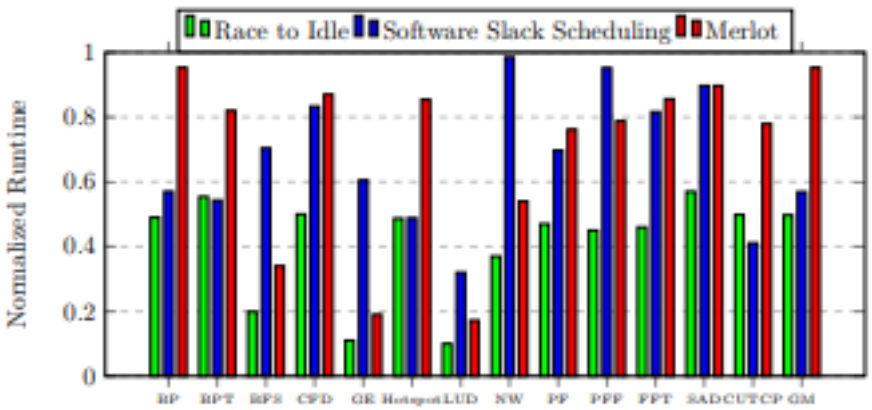
- Benchmark
  - Rodinia Benchmark
  - Parboil Benchmark
- WCET
  - Measure the WCET time by MERLOT (dynamic).
  - Set the deadline = WCET.
    - Varying the WCET from default to 1.5X, 2X, 5X, 10X, 15X.
- DVFS with 23.3 W idle power.

SM frequency		*DRAM frequency*		Normalized Voltage
f_{SM}	frequency	f_{mem}	frequency	
7	700 MHz	7	924 MHz	1
6	600 MHz	6	792 MHz	0.9
5	500 MHz	5	660 MHz	0.83
4	400 MHz	4	528 MHz	0.76
3	300 MHz	3	396 MHz	0.7
2	200 MHz	2	264 MHz	0.62
1	100 MHz	1	132 MHz	0.55

Results

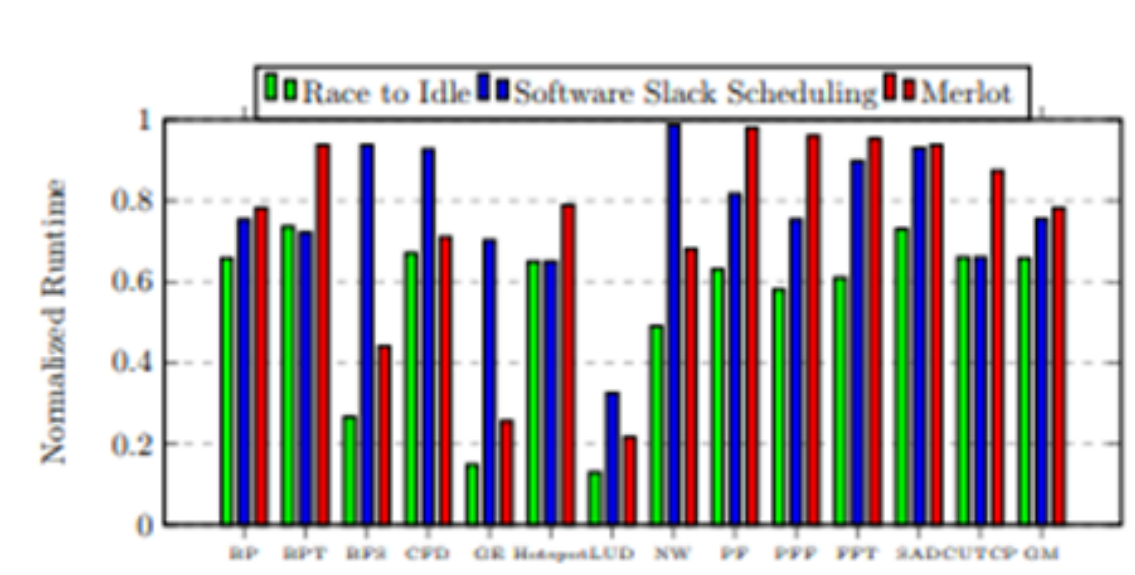
- Timing, never miss deadline.
- Left 1.5X WCET, Right 2X WCET





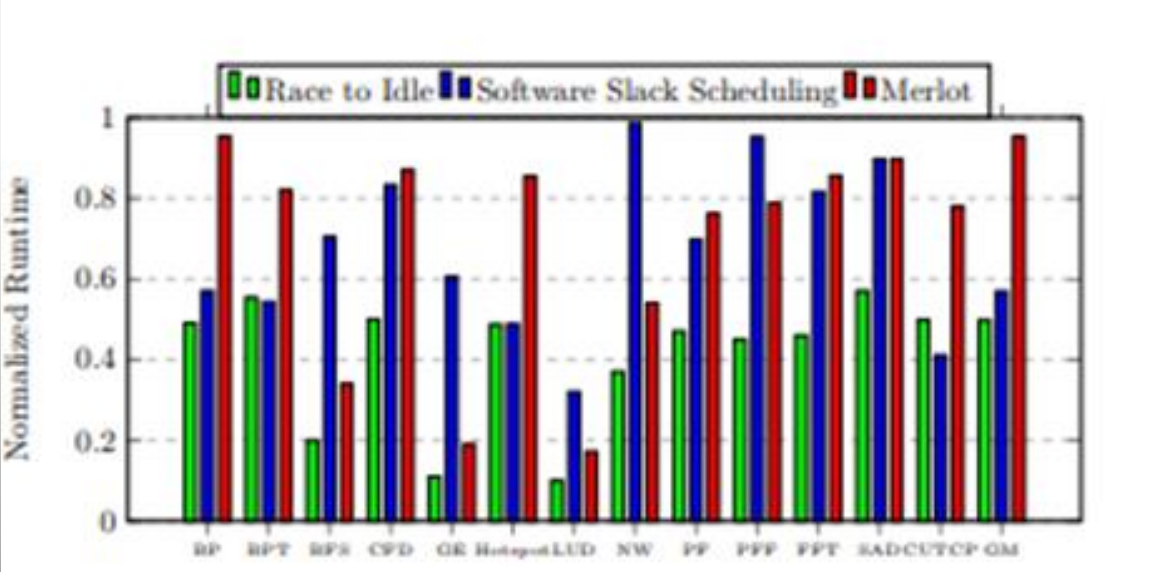
Results 1.5X WCET

- Race to Idle (default speed)
  - 34% faster.
- Software (only reconfigure between kernels)
  - 24 % faster.
- MERLOT (slack transfer in CTA)
  - 22% faster



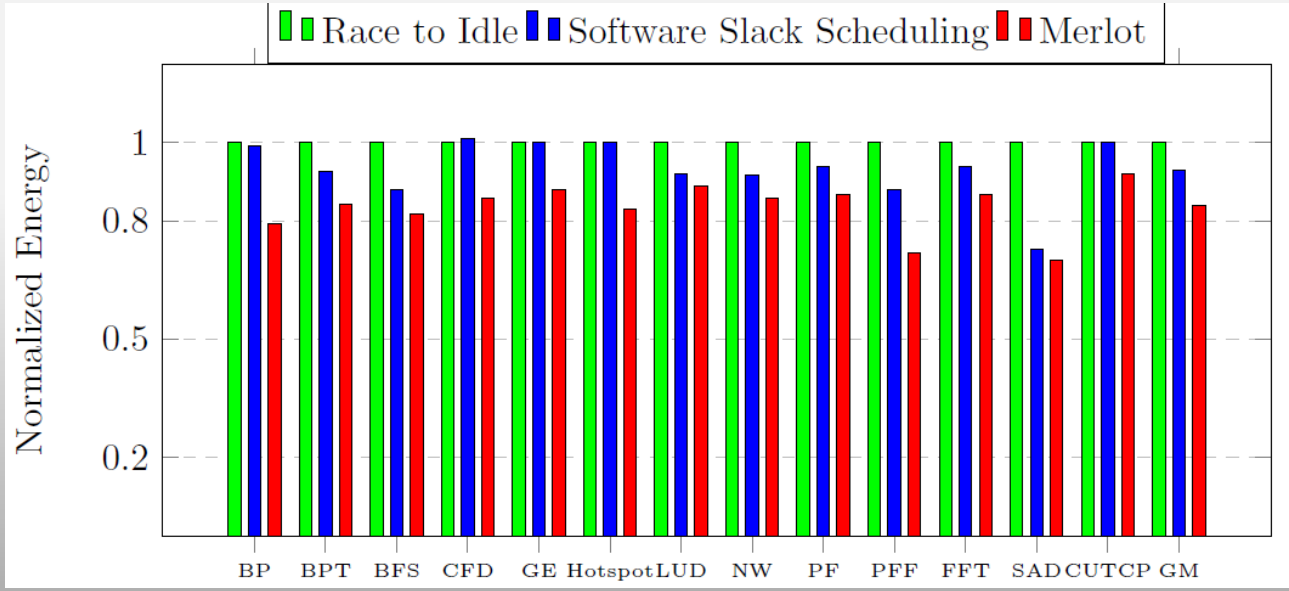
Results 2X WCET

- Race to Idle (default speed)
  - 50% faster.
- Software (only reconfigure between kernels)
  - 43 % faster.
- MERLOT (slack transfer in CTA)
  - 5% faster



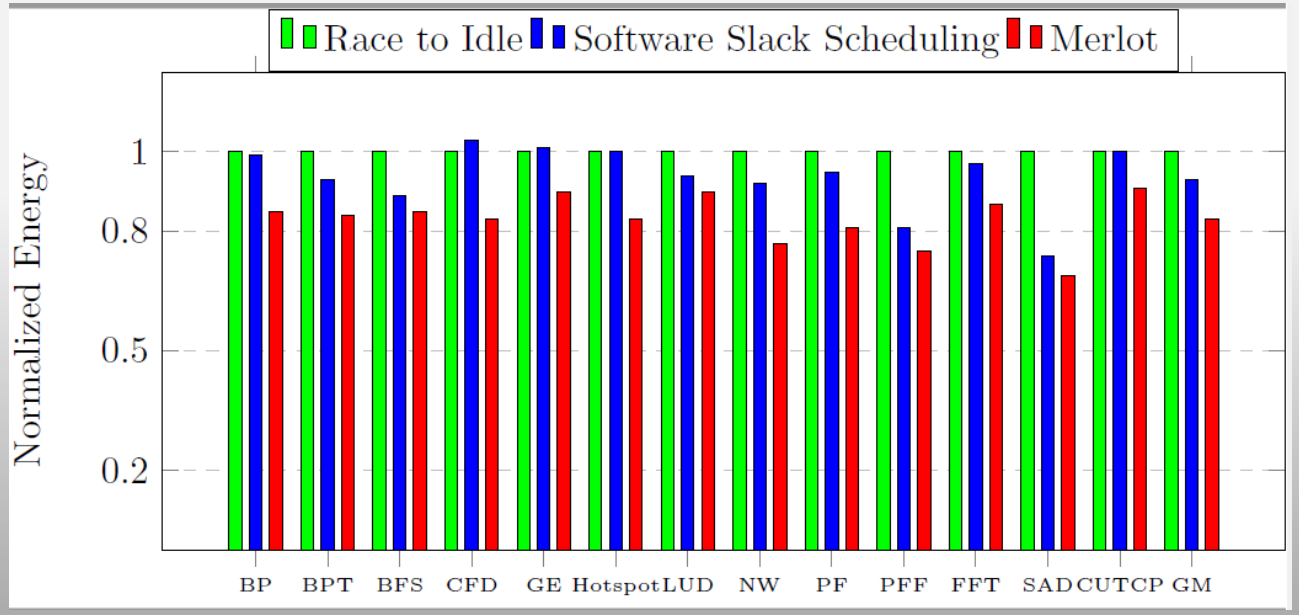
Energy Results (1.5X)

- Software (only reconfigure between kernels)
  - 6.38% energy saving.
- MERLOT (slack transfer in CTA)
  - 16.43% energy saving.



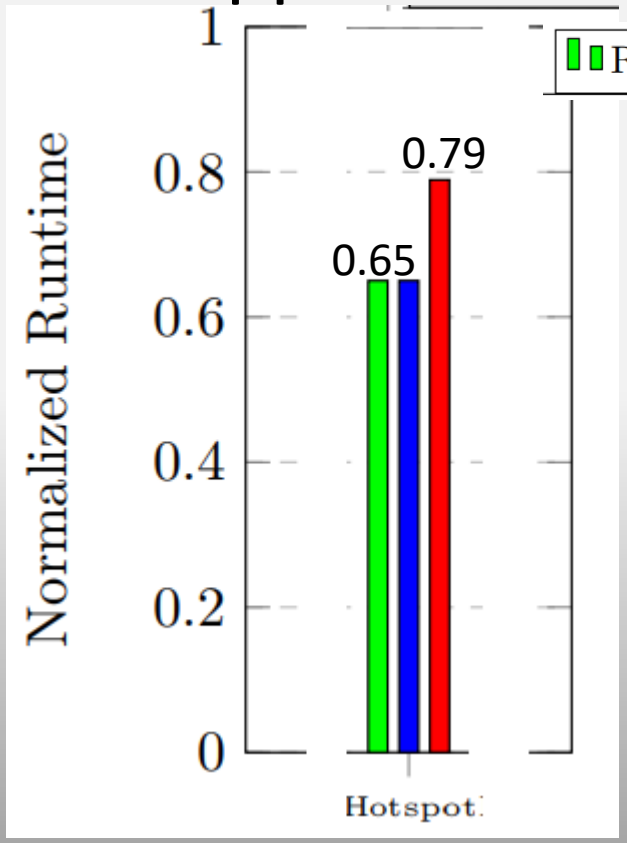
Energy Results (2X)

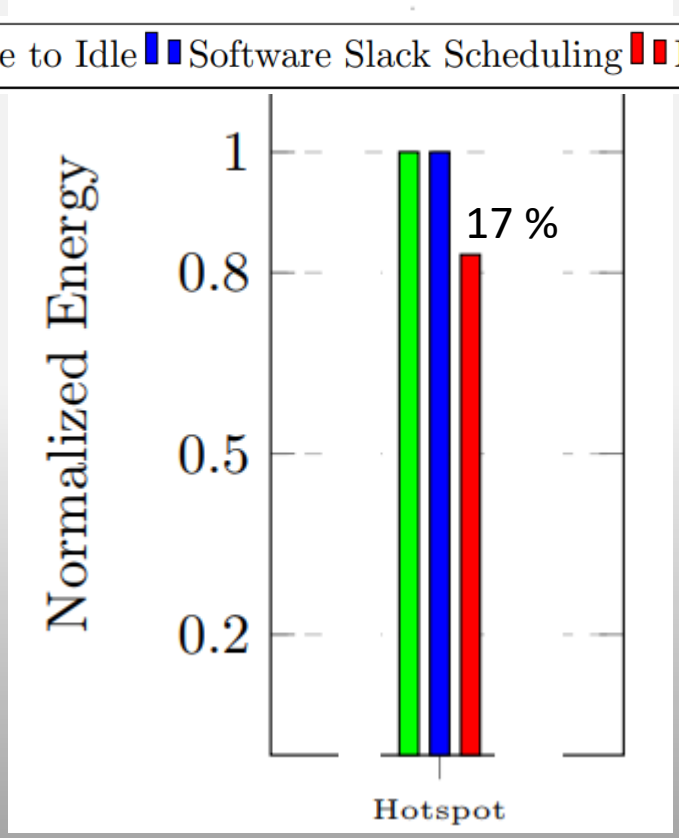
- Software (only reconfigure between kernels)
  - 6.67% energy saving.
- MERLOT (slack transfer in CTA)
  - 17.03% energy saving.



Analysis

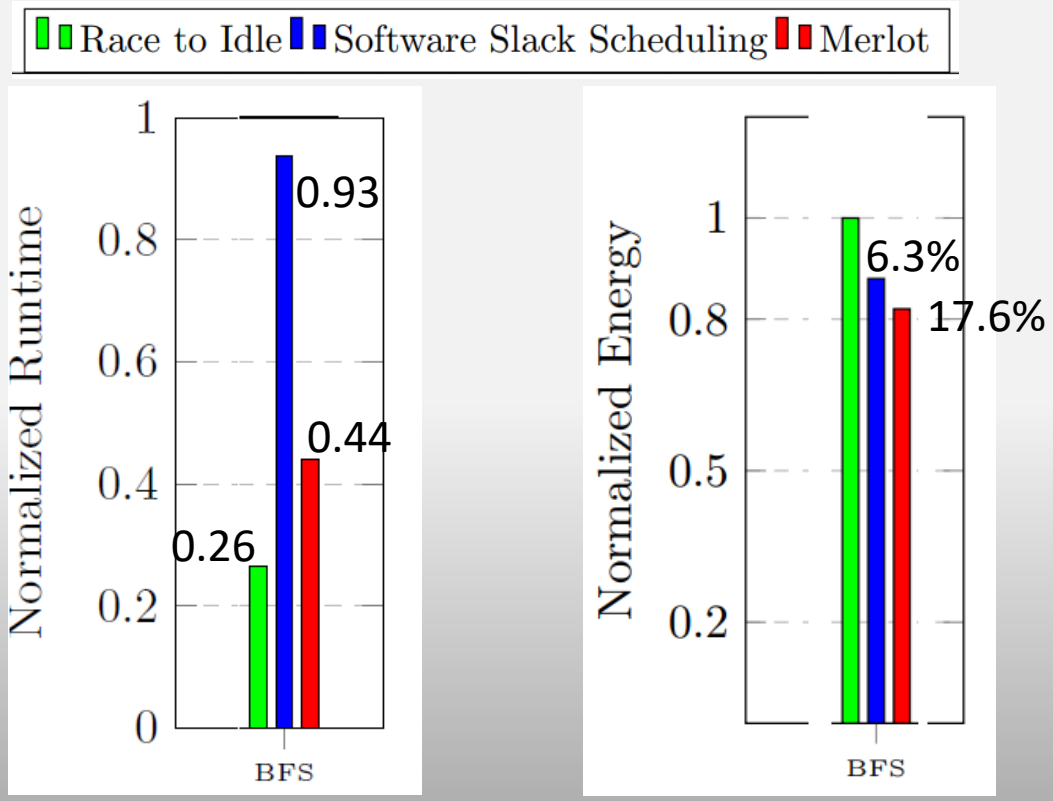
- One application one kernel (Hotspot) in 1.5X.





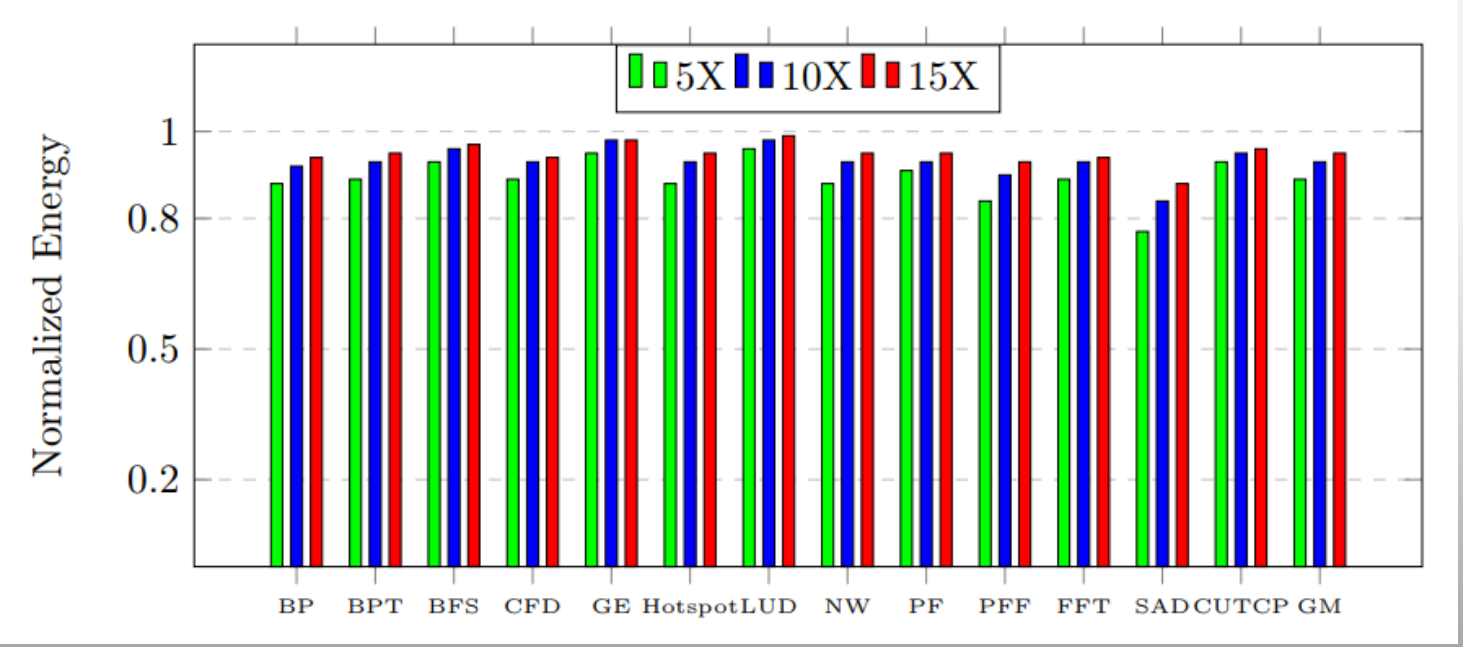
Analysis

- Too much slowdown in software (BFS).
- For (256){
  - BFS()}



MERLOT sensitivity in huge WCET

- 5X saves 10.74%
- 10X saves 6.91%
- 15X saves 5.19%



MERLOT sensitivity in huge WCET

- Decreased because slowdown is limited by frequency critical.

- Energy saving = 1 - $\frac{MERLOT\ energy + idle\ time * idle\ power}{default\ energy + idle\ time * idle\ power}$

Sensitivity to new technology

- Idle power is decreased to 6.8W.
- Better power and performance tradeoff.
- Higher static/leakage power.

Results

- New technology

	Energy Savings	
Deadline	**GTX 580 41.9W**	**GTX 580 83.8W**
1.5	33.93 %	29.76 %
2	34.92 %	30.61 %
5	30.09 %	26.98 %
10	24.54 %	23.08 %
15	20.89 %	20.31 %

MERLOT

- Slack transferring in CTA level.
- Reduce Energy with hard real time guarantee.
- Light overhead without editing GPU code.
- Check the code at https://github.com/santriaji/MERLOT.