BandWatch: A System-Wide Memory Bandwidth Regulation System for Heterogeneous Multicore

Eric Seals

Garmin

erjseals@gmail.com

Michael Bechtel

University of Kansas

mbechtel@ku.edu

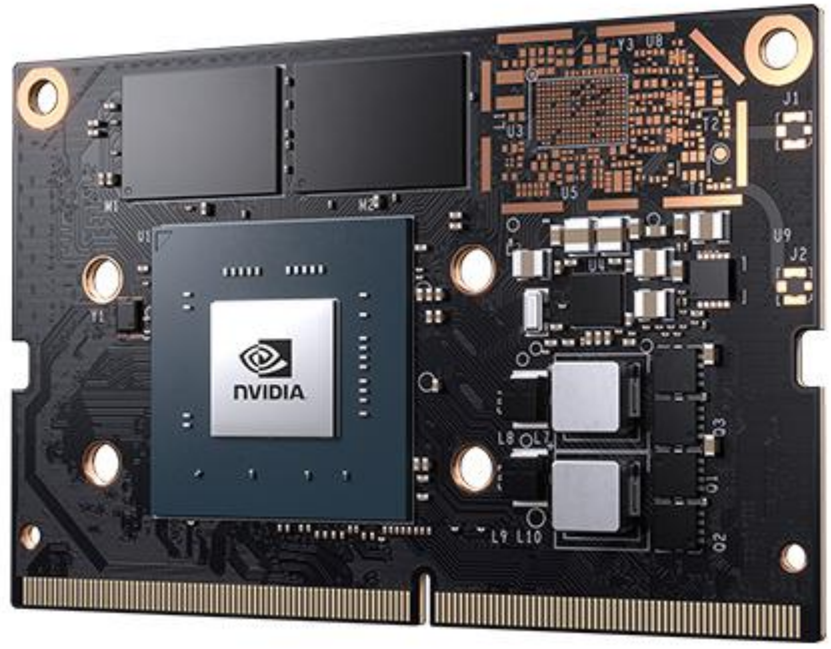
Heechul Yun

University of Kansas

heechul.yun@ku.edu

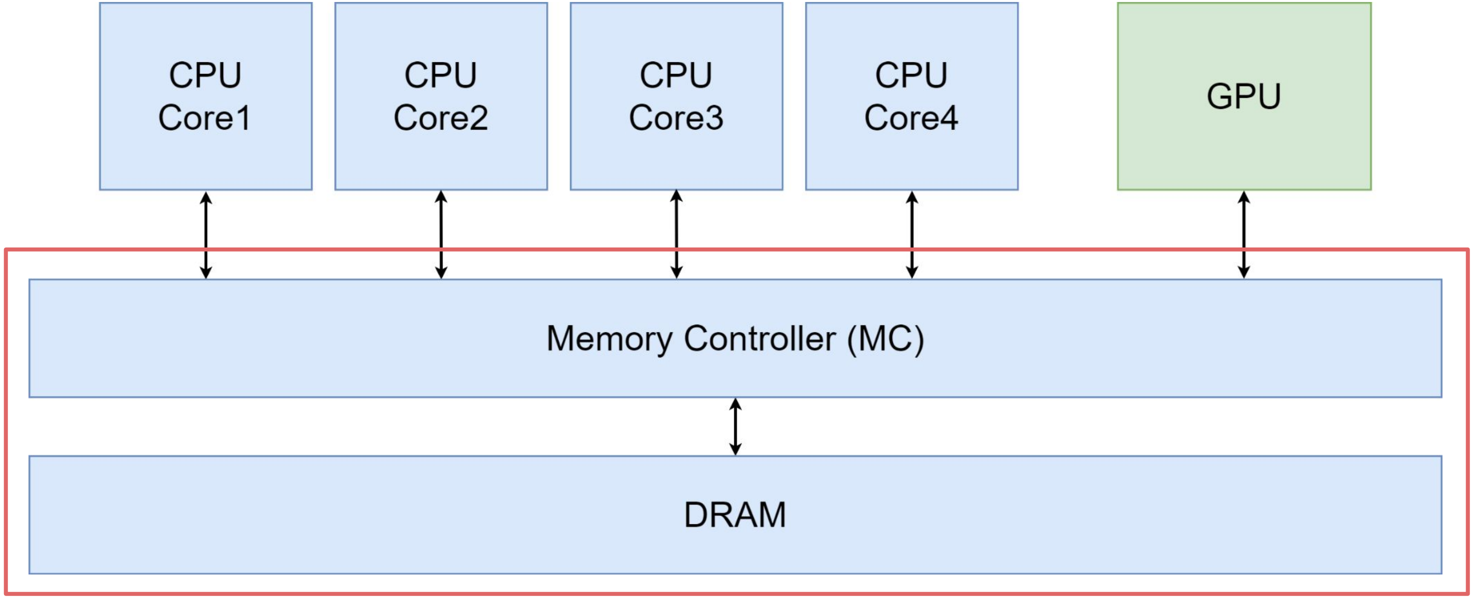
Heterogeneous multicore

- Platforms deliver high throughput
- Shared resource contention can cause major slowdowns
    - CPU's cache
    - DRAM



https://developer.nvidia.com/embedded/jetson-nano

Shared Resource Contention

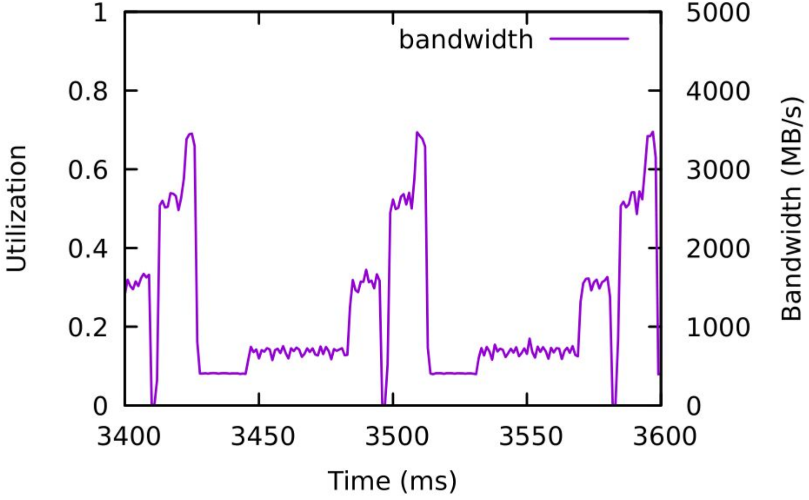


- Memory systems shared by both GPU and CPU
- MC must handle requests from GPU and CPU

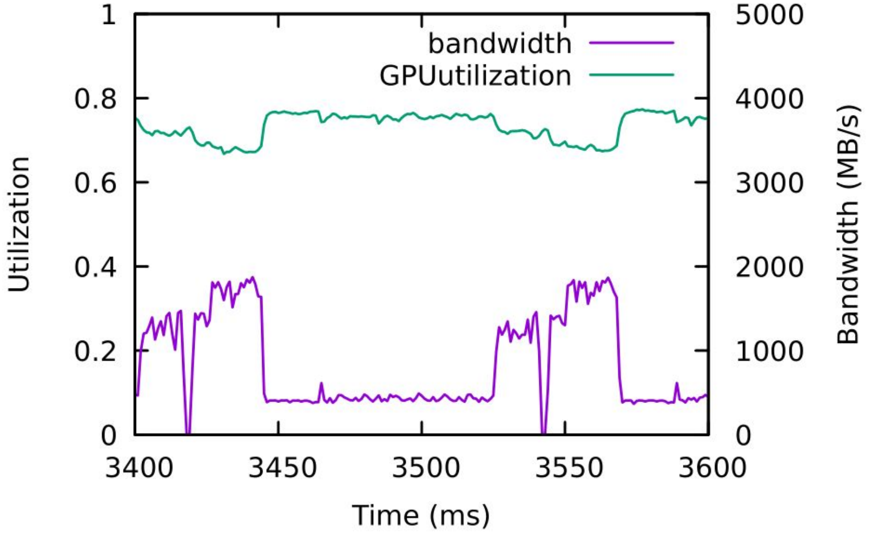
Memory Bandwidth Regulation

- MC must handle requests from GPU and CPU

Solo



w/ GPU Co-runner



BandWatch Contributions

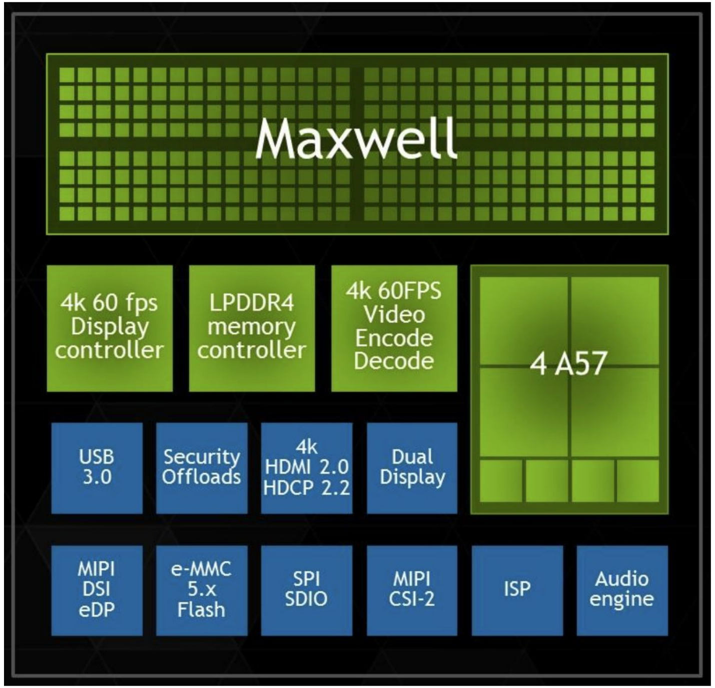
- Holistic bandwidth regulation for heterogeneous multicore systems
- Integrates hardware-software GPU-CPU throttling
- Employs an adaptive strategy
- Extensively tested, ensuring optimal isolation
- Demonstrates improved throughput

Outline

- Motivation
- **Background**
- BandWatch
- Evaluation
- Discussion
- Conclusion

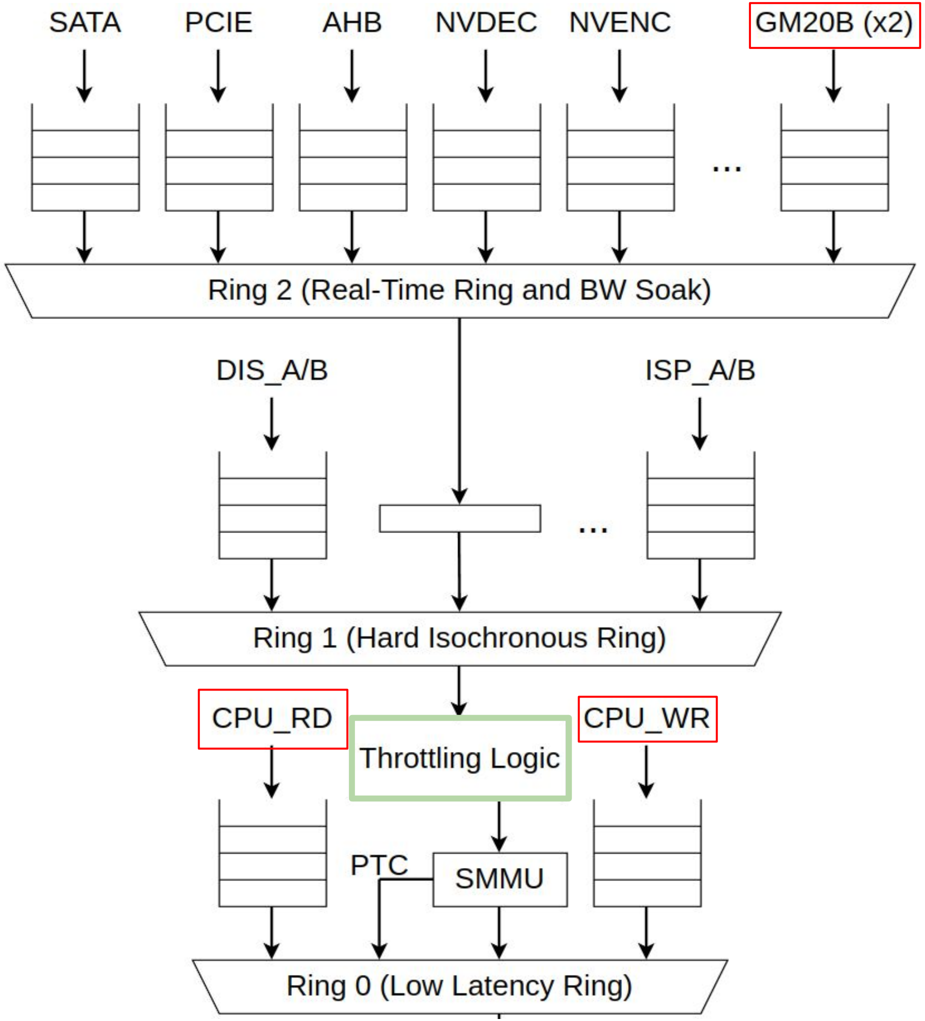
Tegra X1 SoC

- Maxwell GPU
- Quad-core ARM Cortex-A57 CPU
- **Shared Memory Controller**
  - 4 GB LPDDR4, 1600MHz at 25.6 GB/s



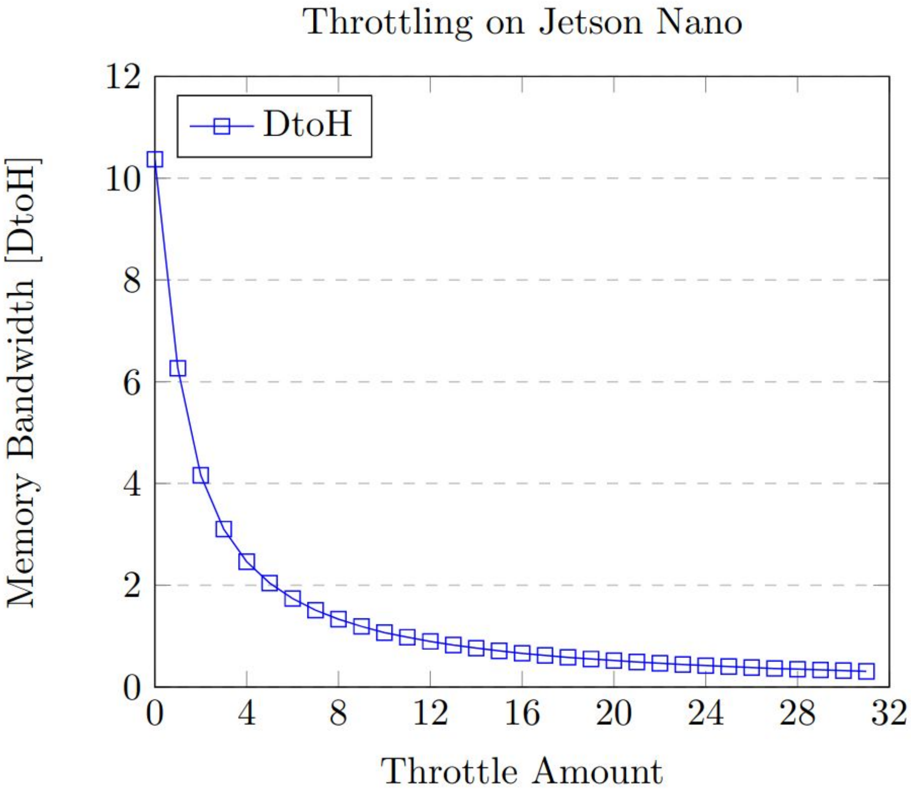
HW Support for Memory Throttling

- Priority Tier Snap Arbiters
- CPU is high-priority
- GPU is low-priority


SATA
PCIE
AHB
NVDEC
NVENC
GM20B (x2)
...
Ring 2 (Real-Time Ring and BW Soak)
DIS_A/B
ISP_A/B
...
Ring 1 (Hard Isochronous Ring)
CPU_RD
Throttling Logic
CPU_WR
PTC
SMMU
Ring 0 (Low Latency Ring)

GPU Throttling Evaluation

- 32 degrees of throttling
- Throttles bandwidth from 11GB/s to 0.1GB/s

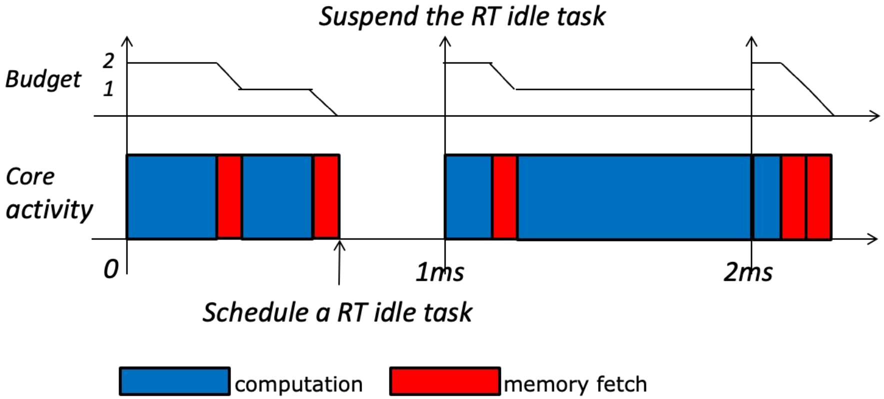


Memory Controller Utilization Monitoring

- Tegra X1 Activity Monitors
  - MC-ALL (total memory events)
  - MC-CPU (CPU memory events)
- Utilization
  - U_{all} : total memory utilization using MC-ALL
  - U_{cpu}: CPU's memory utilization using MC-CPU data
  - U_{gpu} : GPU's memory utilization from $U_{all} - U_{cpu}$

CPU Bandwidth Throttling: MemGuard[3]

- **MemGuard manages individual CPU cores**
- Assigns each core a fraction of total allowed bandwidth
- Stalls CPU core if it exceeds the bandwidth budget



[3] Yun, Heechul, et al. “MemGuard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multi-core Platforms”

Outline

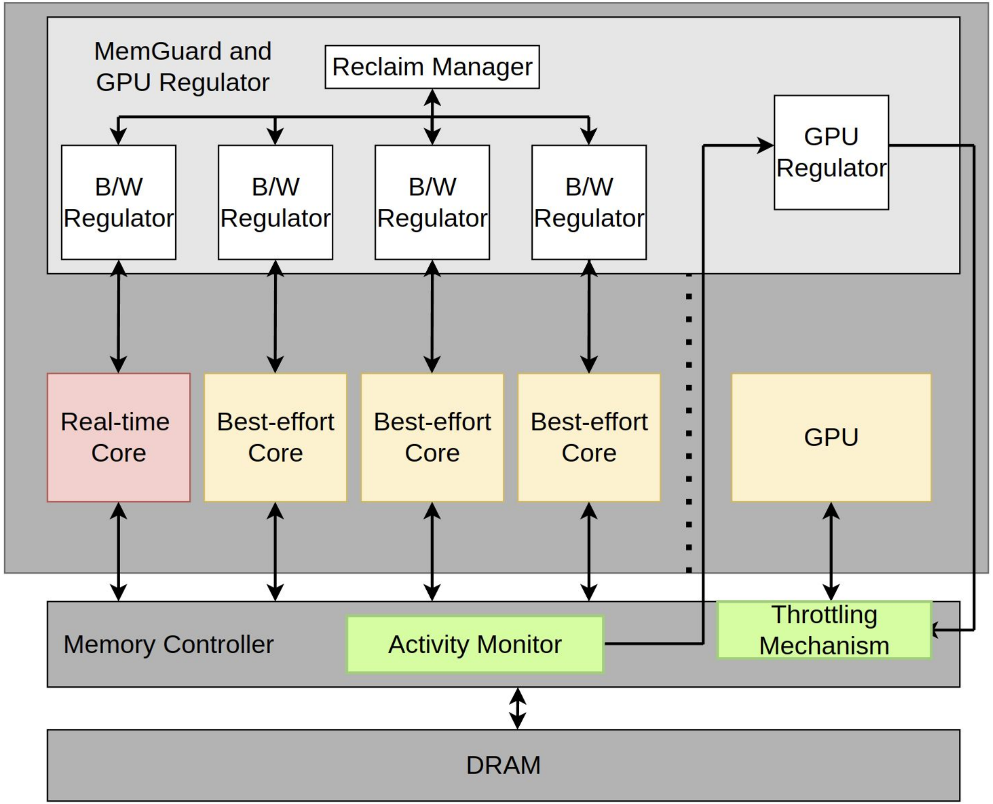
- Motivation
- Background
- **BandWatch**
- Evaluation
- Discussion
- Conclusion

System Model

- Multicore processor with shared DRAM
- Partition between RT and NRT tasks
- One CPU core typically reserved for RT
- Flexible partitioning schemes supported
- BandWatch: isolate RT, maintain NRT performance

BandWatch

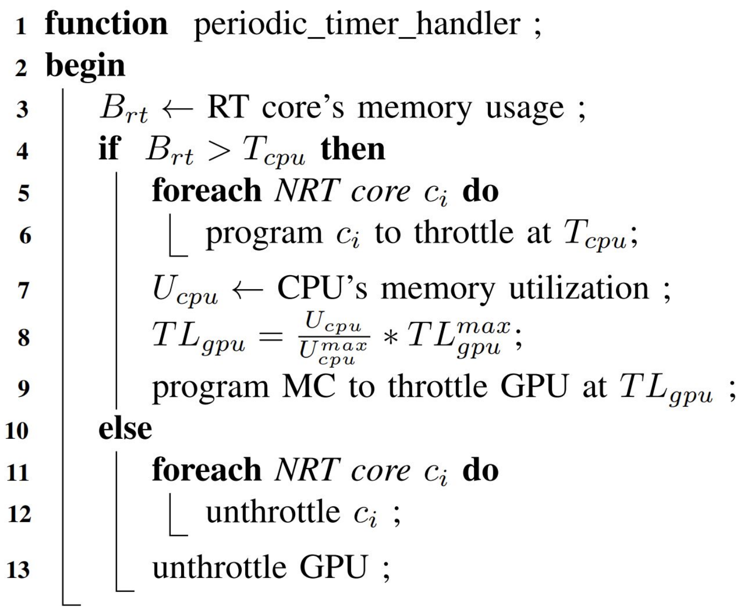
- Activity Monitor provides MC utilization
- Hardware-assisted GPU bandwidth throttling
- MemGuard regulates CPU bandwidth



BandWatch Runtime Regulation Algorithm

High-Level:

- Check RT core memory traffic
- Skip if RT core has low memory usage
- For high RT activity, NRT CPU and GPU are throttled
- Dynamic throttling
  - NRT CPU limited to 75 MB/s
  - GPU proportional to CPU memory usage



```
function periodic_timer_handler ;
begin
    $B_{rt} \leftarrow$ RT core's memory usage ;
    if $B_{rt} > T_{cpu}$ then
        foreach NRT core $c_i$ do
            program $c_i$ to throttle at $T_{cpu}$;
        $U_{cpu} \leftarrow$ CPU's memory utilization ;
        $TL_{gpu} = \frac{U_{cpu}}{U_{cpu}^{max}} * TL_{gpu}^{max}$;
        program MC to throttle GPU at $TL_{gpu}$ ;
    else
        foreach NRT core $c_i$ do
            unthrottle $c_i$ ;
        unthrottle GPU ;
```

Outline

- Motivation
- Background
- BandWatch
- **Evaluation**
- Discussion
- Conclusion

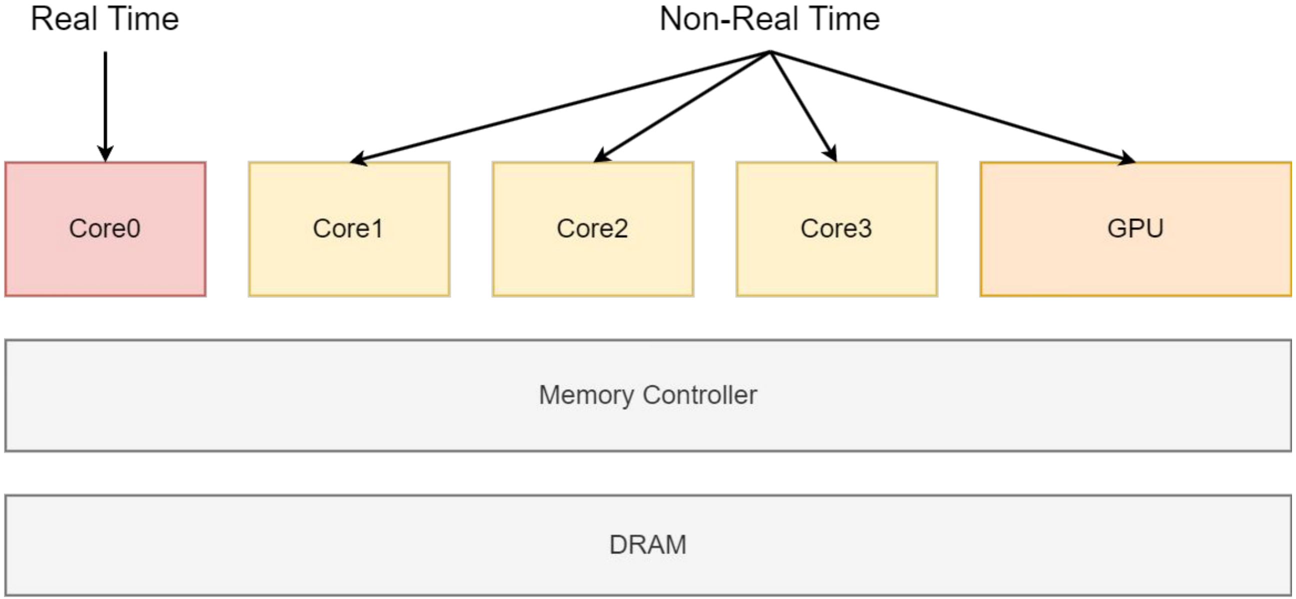
Evaluation

- NVIDIA's Jetson Nano
- Quad-core ARM Cortex-A57s
- 128-core Maxwell based GPU
- 32KB L1 cache per core
- 2MB L2 cache shared
- Memory controller max clock 1.6GHz

[4] "JetsonNano-DevKit" https://developer.nvidia.com/embedded/jetson-nano-developer-kit Accessed 7 June, 2022

Evaluation Setup

- RT CPU Core
  - SD-VBS[5]
- NRT CPU Cores
  - IsolBench[7]
- NRT GPU
  - HeSoC[6]

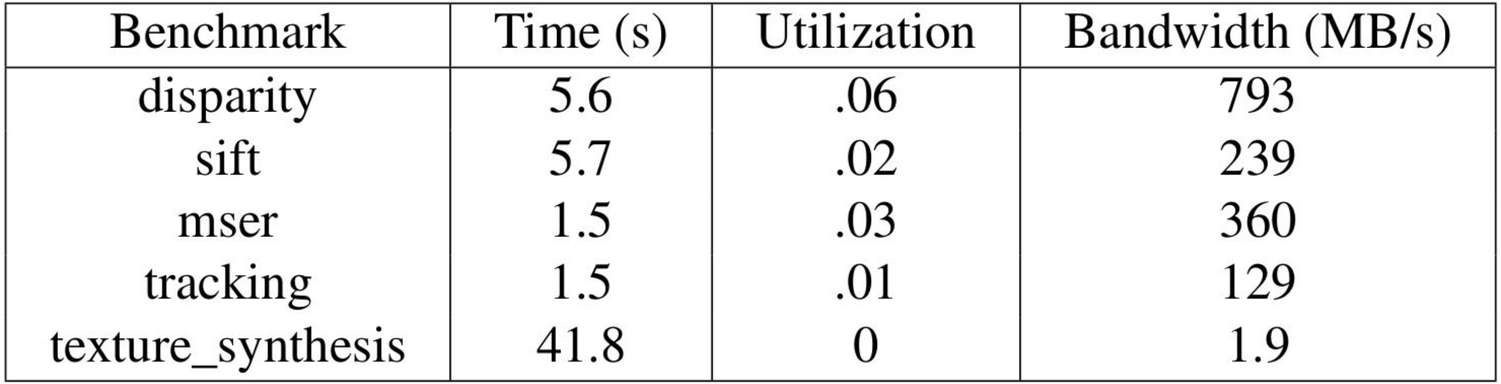


[5] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor. SD-VBS: The San Diego Vision Benchmark Suite

[6] N. Capodieci, R. Cavicchioli, I. S. Olmedo, M. Solieri, and M. Bertogna. Contending Memory in Heterogeneous SoCs: Evolution in NVIDIA Tegra Embedded Platforms.

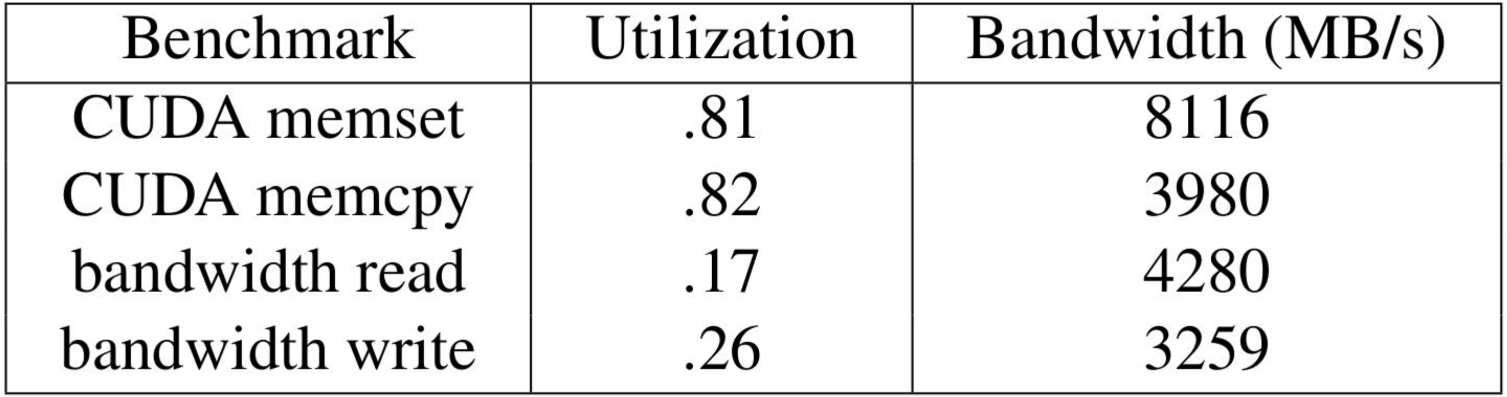
[7] P. K. Valsan, H. Yun, and F. Farshchi. Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems.

SD-VBS Benchmark Solo Performance



Benchmark	Time (s)	Utilization	Bandwidth (MB/s)
disparity	5.6	.06	793
sift	5.7	.02	239
mser	1.5	.03	360
tracking	1.5	.01	129
texture_synthesis	41.8	0	1.9

Interference Benchmarks Solo Performance

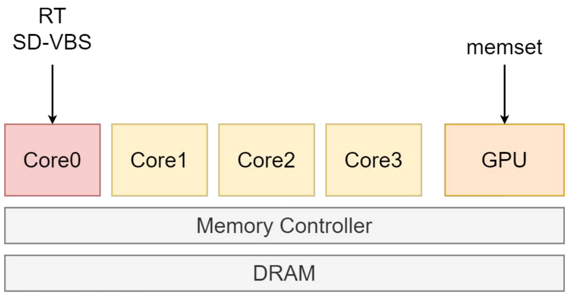


Benchmark	Utilization	Bandwidth (MB/s)
CUDA memset	.81	8116
CUDA memcpy	.82	3980
bandwidth read	.17	4280
bandwidth write	.26	3259

Comparison

- Unregulated
  - Both RT and NRT tasks run w/o any regulation
- Static regulation
  - NRT cores are throttled at a fixed level to achieve less than 10% RT core slowdown via exhaustive offline searching of all possible throttling configurations
- Dynamic regulation (BandWatch)
  - NRT cores are throttled dynamically in response to CPU and GPU memory utilization according to BandWatch runtime regulation algorithm

Impact of GPU Interference



RT SD-VBS

memset

Core0

Core1

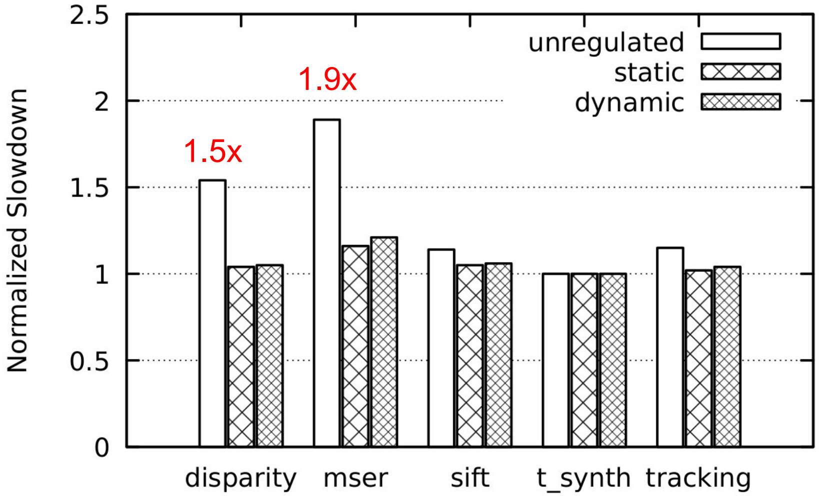
Core2

Core3

GPU

Memory Controller

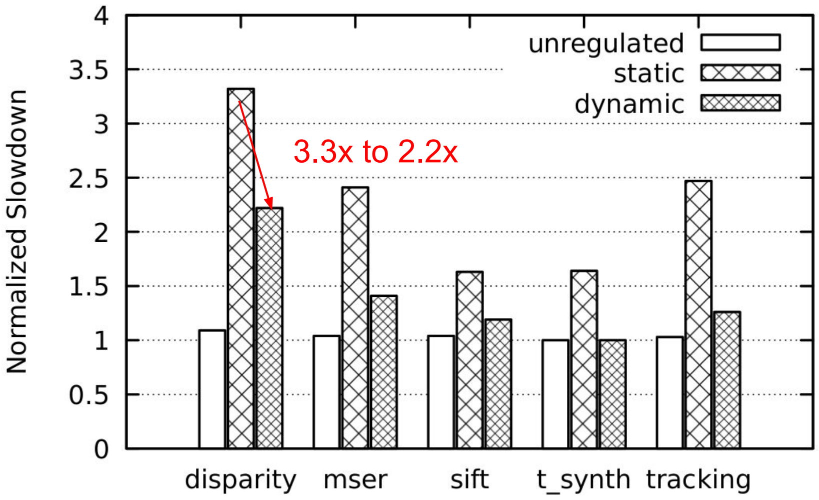
DRAM



1.5x

1.9x

RT (SD-VBS) Isolation impact

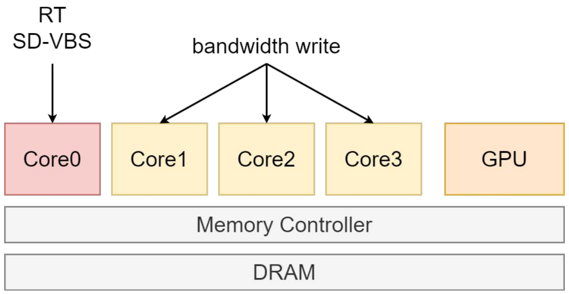


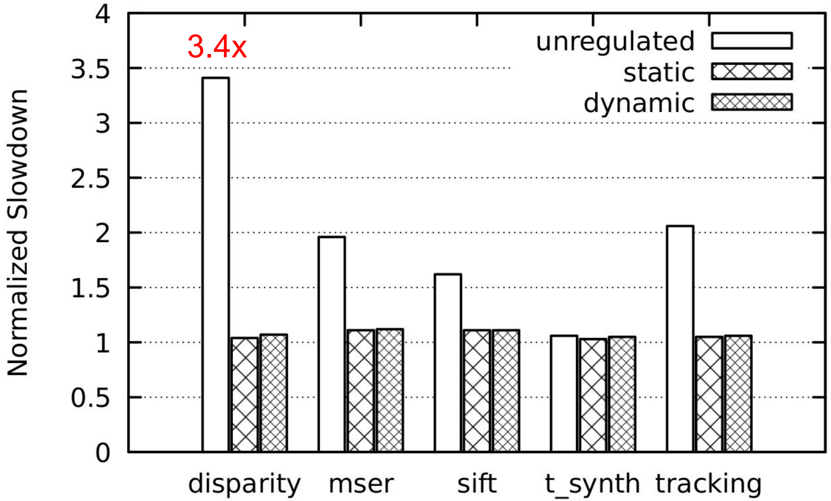
3.3x to 2.2x

NRT (CudaMemSet) performance impact

- BandWatch achieves RT isolation at a lower NRT slowdown vs. *static*

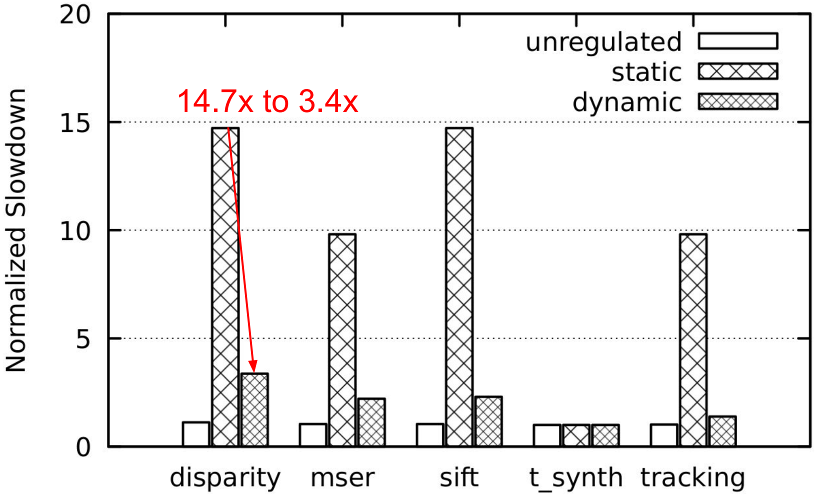
Impact of CPU Interference





3.4x

RT (SD-VBS) Isolation impact

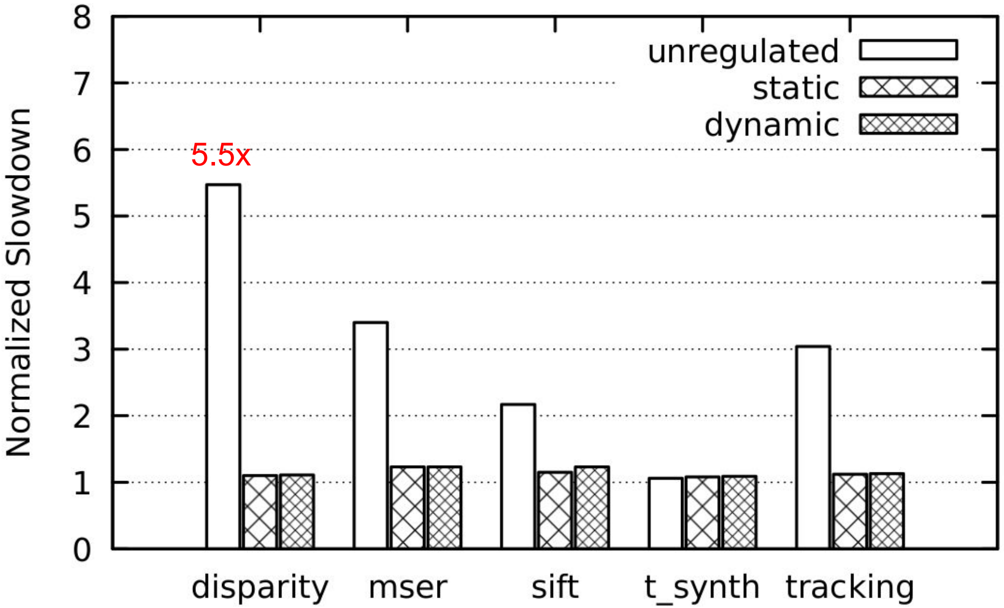


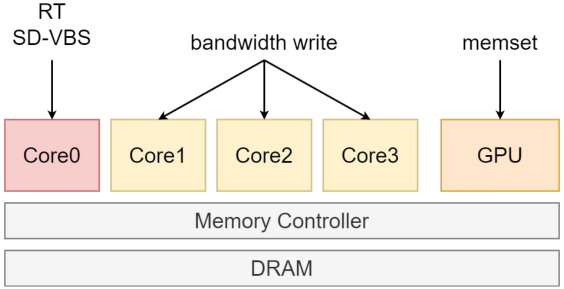
14.7x to 3.4x

NRT (bandwidth write) bandwidth impact

- BandWatch is highly effective for NRT CPU tasks

Impact of CPU and GPU Interference

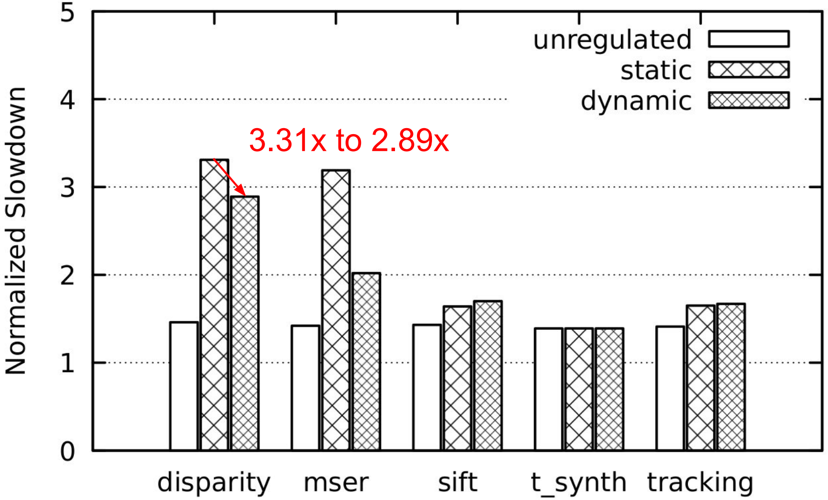




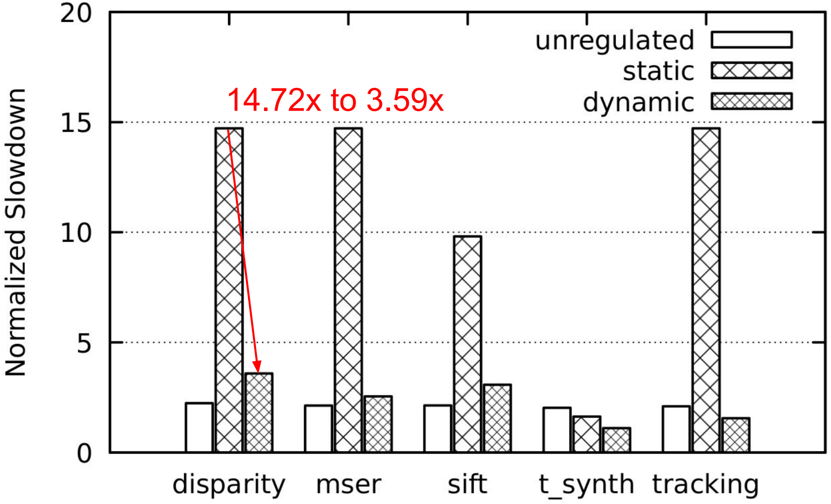
RT (SD-VBS) Isolation impact

- BandWatch and Static still provide RT isolation

Impact of CPU and GPU Interference



NRT GPU bandwidth impact



NRT CPU bandwidth impact

- BandWatch improves performance of both NRT CPU and GPU tasks.

Discussion

- Applicability
  - We exploit Tegra X1 SoC's throttling and monitoring capabilities, which can limit applicabilities on other SoCs
  - But many current/future SoCs already or will have QoS features (e.g,. ARM MPAM) needed support BandWatch
- Execution model
  - BandWatch's model currently focuses on one RT CPU core
  - Extendable to multi-core or iGPU RT tasks are possible and left as future work

Conclusion

- BandWatch is a holistic, adaptive bandwidth management framework for heterogeneous CPU+GPU platforms
  - Provide strong isolation for RT core
  - Minimize performance degradation of NRT co-runners
  - Practical and effective adaptive throttling approach based on CPU and GPU memory utilization
  - Implemented on NVIDIA Tegra X1 SoC

https://github.com/erjseals/bandwatch

Thank you

Disclaimer:

This research was supported in part by NSF grant CNS1815959, CPS-2038923, and NSA Science of Security initiative contract no. H98230-18-D-0009