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
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
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\cite{3}

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

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{memguard_diagram.png}
\caption{MemGuard Diagram}
\end{figure}

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

```c
1 function periodic_timer_handler ;
2 begin
3     \begin{align}
4         B_{rt} \leftarrow & \text{RT core’s memory usage} ; \\
5         \text{if } B_{rt} > T_{cpu} \text{ then} & \\
6             \text{foreach NRT core } c_i \text{ do} & \\
7                 \quad \text{program } c_i \text{ to throttle at } T_{cpu} ; \\
8             U_{cpu} \leftarrow & \text{CPU’s memory utilization} ; \\
9             T L_{gpu} = & \frac{U_{cpu}}{U_{cpu}^{\max}} \times T L_{\max} ; \\
10            \text{program MC to throttle GPU at } T L_{gpu} ; & \\
11            \text{else} & \\
12             \text{foreach NRT core } c_i \text{ do} & \\
13                 \quad \text{unthrottle } c_i ; \\
14             \text{unthrottle GPU} ;
\end{align}
```
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

Evaluation Setup

- **RT CPU Core**
  - SD-VBS\(^5\)

- **NRT CPU Cores**
  - IsolBench\(^7\)

- **NRT GPU**
  - HeSoC\(^6\)

---


\(^7\) 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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time (s)</th>
<th>Utilization</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>disparity</td>
<td>5.6</td>
<td>.06</td>
<td>793</td>
</tr>
<tr>
<td>sift</td>
<td>5.7</td>
<td>.02</td>
<td>239</td>
</tr>
<tr>
<td>mser</td>
<td>1.5</td>
<td>.03</td>
<td>360</td>
</tr>
<tr>
<td>tracking</td>
<td>1.5</td>
<td>.01</td>
<td>129</td>
</tr>
<tr>
<td>texture_synthesis</td>
<td>41.8</td>
<td>0</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Interference Benchmarks Solo Performance

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Utilization</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA memset</td>
<td>.81</td>
<td>8116</td>
</tr>
<tr>
<td>CUDA memcpy</td>
<td>.82</td>
<td>3980</td>
</tr>
<tr>
<td>bandwidth read</td>
<td>.17</td>
<td>4280</td>
</tr>
<tr>
<td>bandwidth write</td>
<td>.26</td>
<td>3259</td>
</tr>
</tbody>
</table>
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

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

**RT (SD-VBS) Isolation impact**

- 1.9x
- 1.5x

**NRT (CudaMemSet) performance impact**

- 3.3x to 2.2x
Impact of CPU Interference

- BandWatch is highly effective for NRT CPU tasks
Impact of CPU and GPU Interference

- BandWatch and Static still provide RT isolation
Impact of CPU and GPU Interference

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