Enabling Real-Time Irregular Data-Flow Pipelines on SIMD Devices

Tom Plano & Jeremy Buhler
CNS-1763503
Gamma Ray Burst Detection

- Collect high volume of energetic photon events

- Pipeline process/filter event stream; decide if GRB observed; notify ground stations

- Deadline: < 1 sec per photon

- GPU used to keep up with data volume, but can't guarantee deadlines
Application Model

Item Flow

n₀ → q₁ → n₁ → q₂ → n₂ → q₃ → n₃
Generalized Data-Dependent Streaming

- **Node** – a schedulable unit of computation labeled $n_0$ to $n_{N-1}$

- **Vector Width** – $v_i$ – size of device vector unit for $n_i$

- **Gain** – $g_i$ – Average outputs per input for $n_i$

- **Execution Time** – $t_i$ – runtime of $n_i$ to process $v_i$ or fewer items

- **Inter-arrival period** – $\tau_i$ – time between item arrivals into $n_i$

- **Deadline** – $D$ – time after an item arrival when outputs must be produced
Device Model

- Single core, sequential
- Preemptive
- Hardware vector operations
- User-accessible interrupts

Examples:
- Commodity GPUs (main focus)
- Commodity CPU running only vector inst’s, no SMT, no DFS
Driving Research Question

How do you *safely and efficiently* process items in a SIMD dataflow pipeline that exhibits the following properties:

1. Pipeline nodes may filter or expand inputs during processing due to data irregularity
2. Work items arrive at regular intervals
3. Work items are deadline constrained
Safety and Efficiency Defined

• A **safe** pipeline rarely misses items deadlines
  • Miss fraction can be tuned to be a very low fraction of total inputs

• An **efficient** pipeline minimizes the fraction of the processor it utilizes
  • High vector occupancy leads to lower processor utilization
  • Low utilization means more time for other processes in the system
Key Observations and Hypothesis

• Observations
  • Existing throughput-oriented scheduler does not respect $D$
  • Node in throughput mode may wait arbitrarily long times before firing
  • \textit{System Slack} – the difference between $D$ and the sum of $t_i$’s
    • Amount of time an item can “wait around” before blowing its deadline

• Hypotheses’
  • Augment scheduler by allocating \textit{slack} among various nodes in the pipeline as \textit{waiting time} $w_i$ may enable deadline awareness
  • Enforce that a node must fire every $w_i + t_i$ time
  • Waiting allows a deadline|occupancy trade off
Primary Contributions

An optimization problem that, given inputs

\{N, v, t_0, ..., t_{n-1}, g_0, ..., g_{n-1}, b_0, ..., b_{n-1}, \tau_0, D\}

computes the values of

\{w_0, ..., w_{n-1}\}

that numerically minimizes the processor utilization of the pipeline, while meeting item deadlines
Informal Optimization Problem

An objective function that encodes device utilization
s.t.
1. All wait times must be positive or zero

2. The number of items delivered to $n_i$ per firing of $n_i$ must be less than $v_i$ on average

3. The relationship between two adjacent nodes’ production and consumption must be stable
   • (Stability is tuned by selectable $b_i$ term)

4. Ensure a worst-case execution does not exceed deadline
Simulating

- Developed event driven simulator to model interrupt enabled GPU hardware

- Loads integer valued configuration of pipeline, simulate arrivals

- Nodes outputs controlled by random process, respecting $g_i$’s

- Report stats on device utilization and miss rate
  - Pipeline level and per node level
A Basis For Comparison – Monolith

• Treat a pipeline as a fused set of operations in a black box

• Items can only wait at the head of the pipeline

• Nodes process all available input before transferring control to the next node

• When the last node terminates, pipeline sleeps until enough input has accumulated at its head
Results
## Presented Test Case

<table>
<thead>
<tr>
<th>Node</th>
<th>( t_i ) (cycles)</th>
<th>( g_i )</th>
<th>( b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>287</td>
<td>0.378</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>955</td>
<td>1.920</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>402</td>
<td>0.0332</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>2753</td>
<td>n/a</td>
<td>6</td>
</tr>
</tbody>
</table>
Device Utilization

- Enforced-Wait \( b_i = \{1, 3, 9, 6\} \)
- Monolith \( S = 1 \)

50th International Conference on Parallel Processing (ICPP)
August 9-12, 2021 in Virtual Chicago, IL
Conclusions

• Formulate optimization problem that models real-time behavior of vector devices for pipelines that meet deadlines

• Formalize a competing model that can also meet deadlines

• Demonstrate the improvement region for the enforced-wait solution, in both numerical solutions and simulation
Future Work

• Refine $b_i$ usage, give stronger guarantees about miss fraction

• Explore heuristic real-time schedulers for real GPUs; enable testing on real hardware.
Extras
Enforced-Wait Miss Rate
Enforced-Wait Optimization

Free variables: wait times $w_0 \ldots w_{N-1} \geq 0$.

minimize $T(\vec{w}) = \frac{1}{N} \sum_{i=0}^{N-1} \left( \frac{t_i}{t_i + w_i} \right)$ s.t.

$(t_0 + w_0)\rho_0 \leq \nu$

$(t_i + w_i)g_{i-1} \leq t_{i-1} + w_{i-1}$ for $1 \leq i < N$

$\sum_i b_i(t_i + w_i) \leq D$
Monolithic Optimization

*Free variable:* block size $M > 0$.

\[
\text{minimize } \frac{\rho_0 \bar{T}(M)}{M} \quad \text{s.t.} \quad \bar{T}(M) \leq \frac{M}{\rho_0}
\]

\[
b \frac{M}{\rho_0} + \hat{T}(M) \leq D
\]

where

\[
\bar{T}(M) = \sum_i \left\lfloor \frac{MG_i}{v} \right\rfloor t_i
\]

\[
\hat{T}(M) = S \bar{T}(M)
\]