Crossing Timezones in the TimeTrial Performance Monitor

Joseph M. Lancaster
Roger D. Chamberlain

{lancaster, roger}@wustl.edu
Dept. of Computer Science and Engineering

Washington University in St. Louis

Funding: NIH, NSF
The Problem

• Applications deployed onto diverse architectures generally execute in different clock domains
• How does one measure performance metrics (e.g. latency) that span multiple time domains?

Contribution

• Developed a method for normalizing across different clock domains (Timezones)
Outline

• **Background**
  – Diverse computing systems & apps.
  – TimeTrial Performance System

• **Crossing Timezones**
  – Transforming to virtual time
  – Measuring latency

• **Experiments and Results**
  – Mercury BLASTN
  – Latency of virtual queues

• **Summary**
Diverse (heterogeneous) systems

- XtremeData, Mercury, Nallatech, etc.
Diverse Applications

- Split
- Sort
- Merge

CPU
- Core
- Core

FPGA

GPU
TimeTrial System

- TimeTrial (TT) helps developers by providing feedback of app. performance
- Instruments, profiles and validates runtime performance of streaming data applications
Features of TimeTrial

• Developer specifies types and location of measurements for streaming apps. using the TimeTrial language
  – “measure mean of rate @ edge”
• Agents monitor & summarize performance
  – reduces communication overhead
• Aggregate results are computed over user-specified data frames
  – e.g. mean rate for 10 MB of data consumed
• Lower monitoring impact by use of idle resources
Extensible Networking Platform - Crossing Timezones in TimeTrial

Monte Carlo - Collect

Rand

Signal monitors

% Utilization

Queue Monitor 0

Cycle Count

Queuing Occupancy

CDF
Outline

• Background
  – Diverse computing systems & apps.
  – TimeTrial Performance System

• Crossing Timezones
  – Transforming to virtual time
  – Measuring latency

• Experiments and Results
  – Mercury BLASTN
  – Latency of virtual queues

• Summary
Crossing Timezones

- Communication latency is the time delay from transmitting an element to a receiver reading it.
Unifying Time

- Must determine relative time of events to monitor communication latency

- TimeTrial discovers timezone differences via a calibration phase

- Cycle counts from each timezone are then normalized into virtual time
Transformation to Virtual Time

- For CPU:
  \[ t_V = s_P t_P + b_P \]
- \( t_P \), \( t_F \) are cycle counter values
- \( s_P \), \( s_F \) are multipliers to normalize slope
- \( b_P \), \( b_F \) are cycle offset conversion factors

- For FPGA:
  \[ t_V = s_F t_F + b_F \]
Virtual Time Example

- For CPU:
  \[ t_V = s_P t_P + b_P \]
- For FPGA:
  \[ t_V = s_F t_F + b_F \]
- How to calculate offsets?
- Set \( b_P \) to an arbitrary value
- Determine \( b_F \) experimentally

*Crossing Timezones in TimeTrial*
1. Send a short message from the CPU to the FPGA and timestamp in virtual time ($t_{V,1}$)
2. FPGA agent sends the FPGA cycle counter value ($t_{F,2}$) back to CPU
3. CPU records virtual time upon receipt ($t_{V,3}$)
Determining remaining offset

- Given causality:
  \[ t_{V,1} < t_{V,2} < t_{V,3} \]

- Assume FPGA timestamp is half-way between the CPU timestamps:
  \[ t_{V,2} = \frac{(t_{V,1} + t_{V,3})}{2} \]

- Substituting for \( t_V \) and setting \( b_P = 0 \):
  \[ b_F = s_P t_{P,1} - s_F t_{F,2} \]
Calibration Uncertainty

• Round-trip latency bounds the virtual time uncertainty

• Repeat calibration cycle to lower uncertainty
  – use only the lowest round-trip time

• In experimental system, minimum round-trip time was ~4 µs

• Uncertainty in virtual time stamps is +/- 4 µs
  – less if comm. delay is symmetric
Outline

• Background
  – Diverse computing systems & apps.
  – TimeTrial Performance System

• Crossing Timezones
  – Transforming to virtual time
  – Measuring latency

• Experiments and Results
  – Mercury BLASTN
  – Latency of virtual queues

• Summary
Experiments

- Mercury BLASTN is a diverse application that finds similarities between two DNA strands
- Two parts mapped to CPU, one on FPGA
- Experiment:

  Determine latency through *virtual queues*
Instrumentation of Mercury BLASTN

- Four event taps were instrumented on each side of a timezone boundary
- Record latencies of first and last elements for each run
Box-Whisker Plots
Extensible Networking Platform

Crossing Timezones in TimeTrial
Summary and Ongoing Work

• Algorithm transforms cycle counts in any timezone to virtual time
• Enables TimeTrial to measure time of events independent of timezone
  – latency through virtual queues
  – any pair of performance events
  – round-trip latency bounds inaccuracy
• Measured virtual queues in Mercury BLASTN

• Verify method accuracy using two FPGAs
• Validate method on systems utilizing GPUs
Questions?

Thank You!

[ERSA 2010, “Better languages for more effective designing”]