Formal Modeling and Analysis of Stream Processing Systems

Linh T.X. Phan

March 2009
High performance requirements

Highly optimized systems
Overview

Stream Processing

Real-Time Embedded System

Methods discussed here are applicable to general real-time embedded systems
Inside the box...

Application

RTOS-APIs

RTOS

Drivers

Core

I/O

Int

Timers

Bus

Input Streams

Output Streams
A complete system

Target Platform
1. Streaming application tasks

- **Application**

![Diagram of streaming application tasks]

- **Color Conversion**
- **Luminance**
- **Chrominance**
- **DCTQ**
- **Run-Level Encoding**
- **Variable Length Coding**
- **IQDCT**

**Acronyms**:
- **DCT**: Discrete Cosine Transform and Quantisation
- **IQDCT**: Inverse Quantisation Discrete Cosine Transform
2. Heterogeneous computing and memory resources
3. Heterogeneous RTOS scheduling and synchronization protocols

- Proportional Share
- EDF
- TDMA
- FCFS
- Dynamic Fixed Priority
- Static
4. Heterogeneous communication resources

- Topology (ring, mesh, star)
- Switching strategies (packet, circuit)
- Routing strategies
  (static, dynamic, reconfigurable)
- Arbitration policies (dynamic, TDM, CDMA)
The Design Problem

Build a system from subsystems that satisfy the application’s requirements and resource constraints.
The Design Process

Application → Allocation
Allocation → Mapping
Mapping → Scheduling
Scheduling → Performance Analysis
Performance Analysis → Design Space Exploration
Design Space Exploration → Architecture

Application → Architecture
The Performance Analysis Problem

Compute/verify the performance properties of the system model
e.g. Architecture of a Picture-in-Picture App.

- Maximum fill-level (backlog) of the buffers?
- Maximum end-to-end delay of the stream?
- Characteristics of the output stream?
- Characteristics of the remaining resource?
Key Challenges: Complex Event Streams

- Infinite sequence of items (events)
- Highly bursty
- Events of multiple types interleaving
- Varied memory and execution demands
- Historically dependent or dynamically controlled
Key Challenges: Complex Tasks & Architectures

• Complex processing semantics
  – fill-level of the buffers
  – execution of an internal automaton
  – synchronization between different streams

• Heterogeneous computing and communication resources

• Various scheduling policies
  – EDF, Fixed-Priority, TDMA, etc.
  – complex state-dependent scheduling schemes
Complex Trade-offs

Computational Demand

Throughput

Memory Size
Two Categories of Performance Analysis

Simulation

An input trace → Simulation → An output trace

Formal Analysis

An abstract input model → Formal Analysis → Analysis bound

Hybrid of Simulation & Formal Analysis

Analysis bound

e.g. A set of input traces
Simulation vs Formal Analysis

- Actual worst case
- Actual best case

Our focus!
Formal Analysis Overview

Concrete System
Formal Analysis Overview

How event streams arrive and its characteristics
Formal Analysis Overview

Concrete System
Performance Model

How much and when the resource are available
Formal Analysis Overview

How events are scheduled and processed
Formal Models and Analysis Methods

1 Standard Event Models (SEM)
   - periodic, periodic with jitter/burst and variations

👍 Simple, easy to analyze

👎 Unrealistic assumptions

👎 Too restrictive

👎 Overly pessimistic results
2 Real-Time Calculus (RTC)

- streams & resources: count-based abstraction
- analysis: (min,+) algebra

👍 Capture burstiness of streams & resource availability

👍 Highly efficient

👎 Cannot model state-dependencies
Timed Automata

- model streams at very detailed level
  - capture exact arrival time of each event

👍 Models state-dependencies

👍 Highly accurate

👎 Too detailed → large models

👎 Large systems → inefficient
4 Event Count Automata (ECA)

- syntax: similar to Timed Automata
- semantics: count-based abstraction
  - capture #events in an interval of time

👍 Models state-dependencies

👍 Highly accurate

👎 Large systems → inefficient
Hybrid Models and Methods

- RTC + SEM
- RTC + ECA
- Multi-Mode RTC
- ...

👍 Good accuracy-efficiency trade-off
The rest of the talk…

Formal Analysis using Real-Time Calculus (RTC)
RTC Background

• Originated from Network Calculus in computer networks domain
  – extended for real-time embedded systems

• Worst-case deterministic formal analysis
  – variant of classical queuing theory

• Abstract models: count-based abstraction

• Analysis: min-plus / max-plus algebra
Recall...
RTC Performance Model

Service Functions
(input)

Arrival Functions
(input)

Algebraic Formulation

Service Functions
(output)

Arrival Functions
(output)
An Arrival Pattern

$R(t) = \text{number of events that } \text{arrive} \text{ in } [0,t)$

#numbers that arrive in $[t, t+\Delta)$ is: $R(t+\Delta) - R(t)$
Count-based Abstraction

<table>
<thead>
<tr>
<th>Δ Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

sliding window size

A set of arrival patterns

Δ = 1

concrete time instant
Load Model: Arrival Functions
\[ \alpha = (\alpha^l, \alpha^u) \]

\[
\begin{array}{|c|c|c|}
\hline
\Delta & \alpha^l(\Delta) & \alpha^u(\Delta) \\
\hline
1 & 1 & 4 \\
2 & 3 & 6 \\
\vdots & \vdots & \vdots \\
\hline
\end{array}
\]

A set of arrival patterns

An arrival pattern \( R(t) \) satisfies \( \alpha \) iff
\[ \alpha^l(\Delta) \leq R(t+\Delta) - R(t) \leq \alpha^u(\Delta) \]
A Service Pattern

\[ C(t) = \text{number of events that can be processed in } [0,t) \]

#events that can be processed in \([t, t+\Delta)\) is: \( C(t+\Delta) - C(t) \)
Service Model: Service Functions

\[ \beta = (\beta^l, \beta^u) \]

<table>
<thead>
<tr>
<th>( \Delta )</th>
<th>( \beta^l(\Delta) )</th>
<th>( \beta^u(\Delta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A service pattern \( C(t) \) satisfies \( \beta \) iff

\[ \beta^l(\Delta) \leq C(t+\Delta) - C(t) \leq \beta^u(\Delta) \]
Units of Arrival and Service Functions

- \([R(t), \alpha(\Delta)]\) and \([C(t), \beta(\Delta)]\) can also be specified in terms of the number of resource units
  - processor cycles, transmitting bit, etc.

- Should always convert to the same unit before performing analysis