HW/SW Codesign
HS 2015

Modular Performance Analysis I

09. December 2015

Pengcheng Huang
pengcheng.huang@tik.ee.ethz.ch
Prolixity time?
What?

Virtually, all man-made systems need this…
What?

1. We are interested in system *timing* properties, e.g. end to end latency, throughput, utilization, fairness...
2 With understanding of those properties, we could optimize the resource usage accordingly (memory, communication, computing, energy...).
Timing

To understand this, what do we need?

Timing

Optimization
Timing

To understand this, what do we need?

Abstraction
Formal Analysis

Timing
Optimization
Real-world problems are often big (if not “messy”)...

What is a good (genuine) abstraction of real systems?

What is a good (scalable) analysis?
The hope (principle)

An event processing network

What is a good (genuine) abstraction of real systems?

What is a good (scalable) analysis?

Modular: component → system
The hope (principle)

An event processing network

What is a good (genuine) abstraction of real systems?

What is a good (scalable) analysis?

Modular: component $\rightarrow$ system

Can you find an example that does not fit to this?

What else could be the hope?
This is where RTC comes in

Reusing lecture slides amap without apologies...

Real-time calculus implemented in MPA tool box
An event processing network
Models for one component

Concrete Instance

Abstract Representation

\[ C(t) \]

\[ R(t) \]

\[ R'(t) \]

\[ \alpha(\Delta) \]

\[ \beta(\Delta) \]
Events flow, get processed...

CPU

BUS

DSP

RM

TDMA

GPC

GSC

GPC

GPC

GPC

GPC

GPC

TDMA

σ

β

α

α'

β_{\text{CPU}}

β_{\text{BUS}}

β_{\text{DSP}}
Fairly general, hopefully useful to other fields.
Event Streams to Arrival Curves

Event Stream

number of events in
in \( t=[0 \ .. \ 2.5] \) ms

Arrival Curves \( \alpha = [\alpha^l, \alpha^u] \)

maximum / minimum
arriving events in \textit{any} interval of length 2.5 ms
Resources to Service Curves

Resource Availability

available service in $t=\left[0..2.5\right]$ ms

Service Curves $\beta = [\beta^l, \beta^u]$
If domain conversion is the answer, what is the question?
Greedy Processing Component

Behavioral Description

- Component is triggered by incoming events.
- A fully preemtatable task is instantiated at every event arrival to process the incoming event.
- Active tasks are processed in a greedy fashion in FIFO order.
- Processing is restricted by the availability of resources.
Greedy Processing Component

- **Examples:**
  - computation (event – task instance, resource – computing resource [tasks/second])
  - communication (event – data packet, resource – bandwidth [packets/second])
Greedy Processing Component

If the resource and event streams describe available and requested units of processing or communication, then

\[
\begin{align*}
C(t) &= C''(t) + R'(t) \\
B(t) &= R(t) - R'(t)
\end{align*}
\]

Conservation Laws

\[
R'(t) = \inf_{0 \leq u \leq t} \{R(u) + C(t) - C(u)\}
\]
Greedy Processing

- For all times $u \leq t$ we have $R'(u) \leq R(u)$ (conservation law).
- We also have $R'(t) \leq R'(u) + C(t) - C(u)$ as the output can not be larger than the available resources.
- Combining both statements yields $R'(t) \leq R(u) + C(t) - C(u)$.
- Let us suppose that $u^*$ is the last time before $t$ with an empty buffer. We have $R(u^*) = R'(u^*)$ at $u^*$ and also $R'(t) = R'(u^*) + C(t) - C(u^*)$ as all available resources are used to produce output. Therefore, $R'(t) = R(u^*) + C(t) - C(u^*)$.
- As a result, we obtain

$$R'(t) = \inf_{0 \leq u \leq t} \{R(u) + C(t) - C(u)\}$$
Abstraction

time domain cumulative functions

GPC

\[ R(t) \rightarrow GPC \rightarrow R'(t) \]

\[ C(t) \]

\[ C'(t) \]

time-interval domain variability curves

GPC

\[ \alpha(\Delta) \rightarrow GPC \rightarrow \alpha'(\Delta) \]

\[ \beta(\Delta) \]

\[ \beta'(\Delta) \]
Tighter Bounds

The greedy processing component transforms the variability curves as follows:

\[ \alpha^{u'} = [(\alpha^u \otimes \beta^u) \otimes \beta^l] \land \beta^u \]
\[ \alpha^{l'} = [(\alpha^l \otimes \beta^u) \otimes \beta^l] \land \beta^l \]
\[ \beta^{u'} = (\beta^u - \alpha^l) \overline{\lor} 0 \]
\[ \beta^{l'} = (\beta^l - \alpha^u) \overline{\lor} 0 \]

Without proof ... .
**Delay and Backlog**

\[
B = \sup_{t \geq 0} \{ R(t) - R'(t) \} \leq \sup_{\lambda \geq 0} \{ \alpha^u(\lambda) - \beta^l(\lambda) \}
\]

\[
D = \sup_{t \geq 0} \{ \inf \{ \tau \geq 0 : R(t) \leq R'(t + \tau) \} \}
= \sup_{\Delta \geq 0} \{ \inf \{ \tau \geq 0 : \alpha^u(\Delta) \leq \beta^l(\Delta + \tau) \} \}
\]
System Composition

How to interconnect service?

Scheduling!

\( \beta_{CPU} \)

\( \beta_{BUS} \)

\( \beta_{DSP} \)

\( \alpha \)

\( \alpha' \)
System Composition

\[ \beta_{CPU} \]
\[ \beta_{BUS} \]
\[ \beta_{DSP} \]

\[ \alpha \]
\[ \alpha' \]

\[ \text{GPC} \]
\[ \text{GPC} \]
\[ \text{GPC} \]
\[ \text{GPC} \]

\[ \text{RM} \]
\[ \text{TDMA} \]
Extending the Framework

- New HW behavior
- New SW behavior
- New scheduling scheme
- ...

Find new relations:
\[ \alpha'(\Delta) = f_{\alpha}(\alpha, \beta) \]
\[ \beta'(\Delta) = f_{\beta}(\alpha, \beta) \]

This is the hard part...!
We are calling
Massively Parallel Architectures

Inventing supercomputing on a chip™

The new generation of C/C++ processors for
- Extreme computing
- Low power
- Real-time applications

Up to 1024 processing cores!
- Amazing performance/power ratio
- VLIW core
- C/C++/OpenCL
- High speed (predictable) network on chip
- Low latency processing
Massively Parallel Architectures

Exciting applications!

Single cloud on a chip
• data analytics
• Image processing
• High performance computing

Industrial applications
• Flight control / management systems
• Smart car systems
Massively Parallel Architectures

Challenges!

Single cloud on a chip
- Performance optimization
- Programmability
- Energy / reliability / thermal issues
- ...

Industrial applications
- Strict predictability
- Extreme dependability
- High efficiency
- ...

Swiss Federal Institute of Technology
Massively Parallel Architectures

We are calling!

• Benchmarking / modeling / understanding
  – Safety-critical applications
  – Is the behavior of all resources predictable?
  – Can machine learning help us reverse engineer h/w resources?

• Programming / Optimization
  – Model driven design
  – Scheduling
  – Energy efficiency…
Massively Parallel Architectures

We are calling!

- Benchmarking / modeling / understanding
  - Safety-critical applications
  - Is the behavior of all resources predictable?
  - Can machine learning help us reverse engineer h/w resources?

- Programming / Optimization
  - Model driven design
  - Scheduling
  - Energy efficiency…
Massively Parallel Architectures

We are calling!

Visit our webpage!

http://www.tec.ethz.ch/theses.php
Your turn now
That is all, folks!

• See you next week