

# TECHNIQUES FOR DIMENSIONING AGGREGATED EVENT STREAMS TO SUSTAIN GLOBAL APPLICATION'S DEMANDS

Lorenzo Di Gregorio  
lorenzo.digregorio@lantiq.com



Lantiq Deutschland GmbH, Neubiberg, Germany

3rd Workshop on Mapping Applications on MPSoCs, 2010

Dimensioning  
Event Streams

Di Gregorio

Motivation

Problem  
Example

Event Streams

Definition  
Modeling  
Bounds over time

Techniques

Empirical  
Analytical

## DISCLAIMER

All opinions expressed here are those of the author individually and are not reflective or indicative of the opinions and positions of the author's employer.

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- Modeling event streams using linear inequalities
- Time and variability in event streams

## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- Modeling event streams using linear inequalities
- Time and variability in event streams

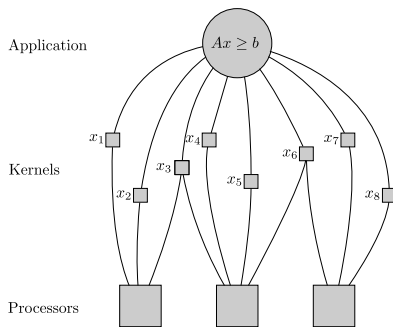
## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## PROBLEM

What level of performance is demanded on every processing element, so that they **jointly** satisfy the global demand?

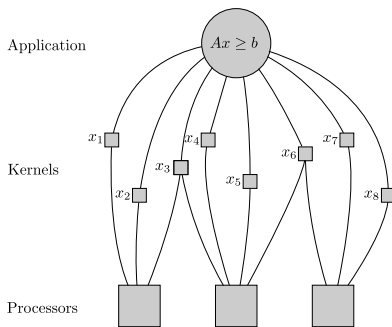
- Application = interworking of **multiple kernels**.
- Demand on kernels for **aggregated streams** of computation events.
  - aggregated = many contributors to one performer.
- Demand on **processing elements** for performance in executing kernels.



## PROBLEM

What level of performance is demanded on every processing element, so that they **jointly** satisfy the global demand?

- Application = interworking of **multiple kernels**.
- Demand on kernels for **aggregated streams** of computation events.
  - aggregated = many contributors to one performer.
- Demand on **processing elements** for performance in executing kernels.



It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
  - sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- Model many problems by parametric linear inequalities.
- Solve the “architectural part” of the problem only once.
- Apply the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

It is a standard industry practice to solve such problems

- by incremental steps and gut feeling,
- by tabling scenarios on spreadsheets,
- by profiling and tweaking the design,
- sometimes, by developing abstract architecture models.

using mathematics is usually considered not feasible for this **complexity** and too cumbersome.

### HOW COMPLEX IS “TOO COMPLEX”?

Some quite generic problems are not as complex as they appear.

- **Model** many problems by parametric linear inequalities.
- **Solve** the “architectural part” of the problem only once.
- **Apply** the “parametric linear solution” to any variable data.

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- **An introductory case study**

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- Modeling event streams using linear inequalities
- Time and variability in event streams

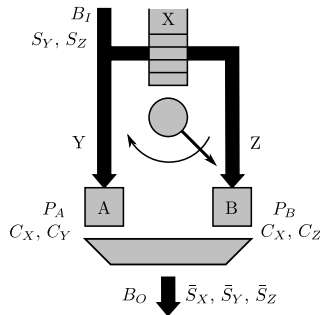
## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## QUESTION

Under sufficient load, how many instructions per unit time,  $P_A$  and  $P_B$ , are demanded from A and B to **guarantee** that the bandwidth  $B_O$  of the output channel is exhausted?

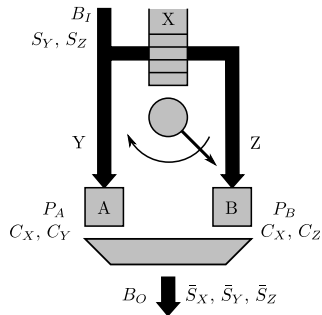
An input channel delivers frame streams Y and Z to two cores A and B, which sustain a fairly distributed traffic flow X from a queue of packets. All the processed streams X, Y and Z are multiplexed into a common output channel.



## QUESTION

Under sufficient load, how many instructions per unit time,  $P_A$  and  $P_B$ , are demanded from A and B to **guarantee** that the bandwidth  $B_O$  of the output channel is exhausted?

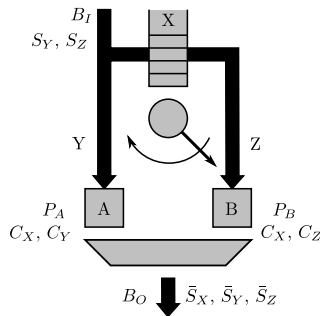
An input channel delivers frame streams Y and Z to two cores A and B, which sustain a fairly distributed traffic flow X from a queue of packets. All the processed streams X, Y and Z are multiplexed into a common output channel.



## QUESTION

Under sufficient load, how many instructions per unit time,  $P_A$  and  $P_B$ , are demanded from A and B to **guarantee** that the bandwidth  $B_O$  of the output channel is exhausted?

- Processing a frame of Y costs **up to**  $C_Y$  **instructions** and changes its **maximum size** from  $S_Y$  to  $\bar{S}_Y$ .
- Similarly a frame of Z costs  $C_Z$  and changes from  $S_Z$  to  $\bar{S}_Z$ , a packet of X costs  $C_X$  and changes from  $S_X$  to  $\bar{S}_X$
- The input channel **guarantees** a bandwidth  $B_I$ .

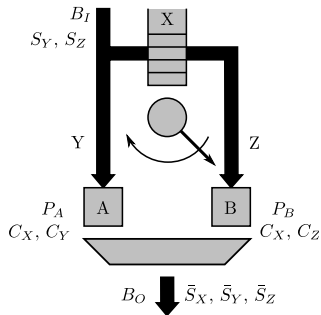


## QUESTION

Under sufficient load, how many instructions per unit time,  $P_A$  and  $P_B$ , are demanded from A and B to **guarantee** that the bandwidth  $B_O$  of the output channel is exhausted?

$$\begin{aligned}
 S_Y x_Y + S_Z x_Z &\geq B_I \\
 C_X x_X + C_Y x_Y &\geq P_A \\
 C_X x_X + C_Z x_Z &\geq P_B \\
 \bar{S}_X x_X + \bar{S}_Y x_Y + \bar{S}_Z x_Z &\geq B_O
 \end{aligned}$$

$x_Y$  and  $x_Z$  represent respectively the total frames per unit time of the streams Y and Z. Similarly,  $x_X$  represents the total packets per unit time of the flow X.



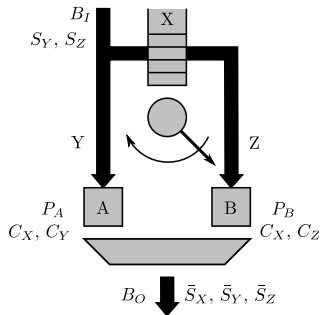
## QUESTION

Under sufficient load, how many instructions per unit time,  $P_A$  and  $P_B$ , are demanded from A and B to **guarantee** that the bandwidth  $B_O$  of the output channel is exhausted?

$$\begin{aligned}
 S_Y x_Y + S_Z x_Z &\geq B_I \\
 C_X x_X + C_Y x_Y &\geq P_A \\
 C_X x_X + C_Z x_Z &\geq P_B \\
 \bar{S}_X x_X + \bar{S}_Y x_Y + \bar{S}_Z x_Z &\geq B_O
 \end{aligned}$$

## ANSWER

we must determine  $P_A$  and  $P_B$  such that the last inequality is satisfied for all  $x_X$ ,  $x_Y$  and  $x_Z$ .



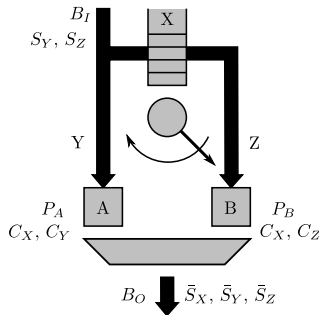
## QUESTION

Under sufficient load, how many instructions per unit time,  $P_A$  and  $P_B$ , are demanded from A and B to **guarantee** that the bandwidth  $B_O$  of the output channel is exhausted?

$$\begin{aligned}
 S_Y x_Y + S_Z x_Z &\geq B_I \\
 C_X x_X + C_Y x_Y &\geq P_A \\
 C_X x_X + C_Z x_Z &\geq P_B \\
 \bar{S}_X x_X + \bar{S}_Y x_Y + \bar{S}_Z x_Z &\geq B_O
 \end{aligned}$$

## ANSWER

we must determine  $P_A$  and  $P_B$  such that the last inequality is satisfied for all  $x_X$ ,  $x_Y$  and  $x_Z$ .



## SOLUTION

The inequality  $\bar{S}_X x_X + \bar{S}_Y x_Y + \bar{S}_Z x_Z \geq B_O$  is **redundant** if:

$$\left[ -K - \frac{\bar{S}_Z}{S_Z}, -K - \frac{\bar{S}_X}{C_X}, K, 1 \right] \begin{bmatrix} B_I \\ P_A \\ P_B \\ B_O \end{bmatrix} \leq 0$$

where

$$K \doteq \frac{\bar{S}_Y - \bar{S}_X \frac{C_Y}{C_X} - S_Y \frac{\bar{S}_Z}{S_Z}}{C_Y + S_Y}$$

## BEFORE PROCEEDING WITH THE MATH ...

What can be modeled by linear inequalities?

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- **What are event streams**
- Modeling event streams using linear inequalities
- Time and variability in event streams

## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## WHAT ARE EVENT STREAMS

- An **event** is the crossing of one control flow through one selected device and instruction. Example: one function call.
- The **type** of an event is its equivalence class with respect to multiple objectives. Example: all lookups in a table.
- A **stream** is a counter over time of events of the same type.

A stream does not need to be localized, events can be anywhere.

## QUESTION

What aggregations can be modeled by linear inequalities?

$$\alpha_1 x_1 + \dots + \alpha_n x_n \geq \beta$$

## WHAT ARE EVENT STREAMS

- An **event** is the crossing of one control flow through one selected device and instruction. Example: one function call.
- The **type** of an event is its equivalence class with respect to multiple objectives. Example: all lookups in a table.
- A **stream** is a counter over time of events of the same type.

A stream does not need to be localized, events can be anywhere.

## QUESTION

What aggregations can be modeled by linear inequalities?

$$\alpha_1 x_1 + \dots + \alpha_n x_n \geq \beta$$

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- **Modeling event streams using linear inequalities**
- Time and variability in event streams

## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

### QUESTION ...

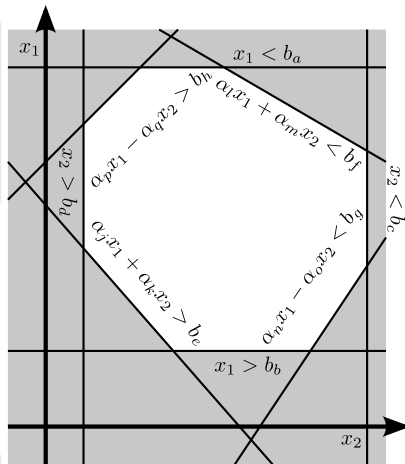
What aggregations can be modeled by linear inequalities?

$$Ax \geq b$$

- It is always an n-dimensional polytope.
- It must be convex, it can be open, it can imply equalities.

### ... TURNS INTO QUESTION

What do facets model?

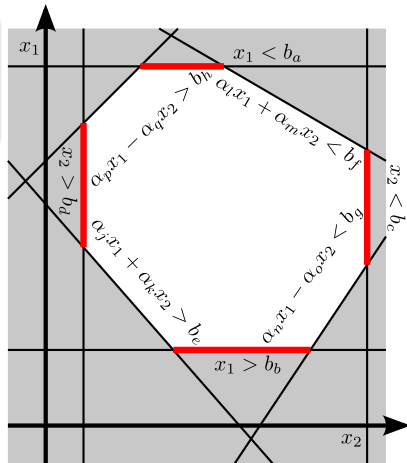


## INDIVIDUAL WORST &amp; BEST

$$b_b < x_1 < b_a$$

$$b_d < x_2 < b_c$$

- Absolute best and worst cases for the event streams.
- Worst cases can be caused by bottlenecks like channel capacity.
- Best cases can be intended “under available workload”.

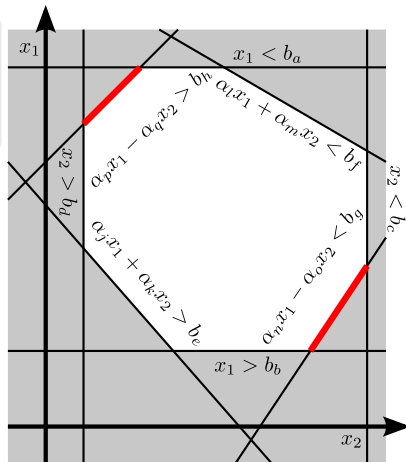


## MAX &amp; MIN SHIFT

$$\alpha_n x_1 - \alpha_o x_2 < b_g$$

$$\alpha_p x_1 - \alpha_q x_2 > b_h$$

- One stream cannot be increased above a given limit, without increasing also the other one.
- Reasons can be data hazards, but also workload characteristics like frame types in an MPEG stream.

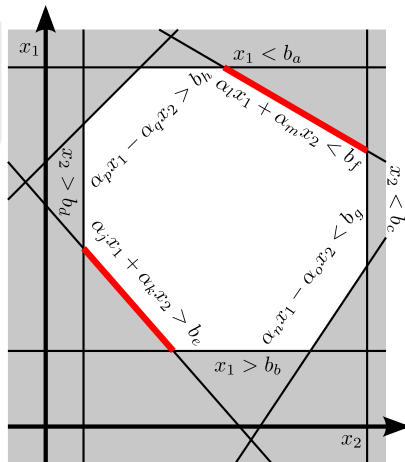


### COMMON WORST & BEST

$$\alpha_j x_1 + \alpha_k x_2 > b_e$$

$$\alpha_l x_1 + \alpha_m x_2 < b_f$$

- Upper limit of the system's performance under peak load and improvements over the individual worst cases.
- The system will process more events of one type if it less loaded from events of the other type.



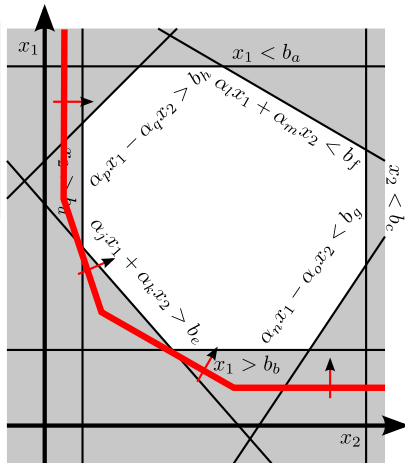
### GUARANTEE & LIMITATION

$$Ax \geq b \rightsquigarrow D^+ x \leq r^+$$

$$Ax \geq b \rightsquigarrow D^\pm x \geq r^+$$

$\rightsquigarrow$  means “is a dependency set for the infeasibility of”

- The demand is guaranteed:  $D^+ x \geq r^+$  is always satisfied.
- The limitation is never violated:  $D^\pm x \leq r^+$  is always satisfied.



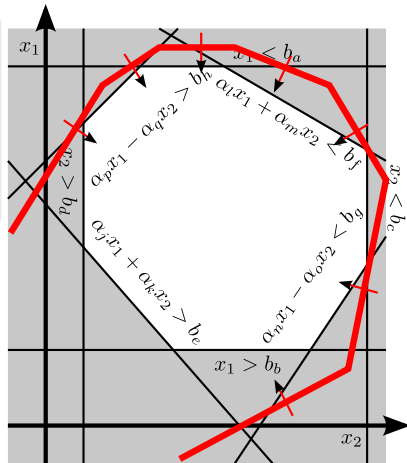
### GUARANTEE & LIMITATION

$$Ax \geq b \rightsquigarrow D^+ x \leq r^+$$

$$Ax \geq b \rightsquigarrow D^\pm x \geq r^+$$

$\rightsquigarrow$  means “is a dependency set for the infeasibility of”

- The demand is guaranteed:  $D^+ x \geq r^+$  is always satisfied.
- The limitation is never violated:  $D^\pm x \leq r^+$  is always satisfied.



## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- Modeling event streams using linear inequalities
- Time and variability in event streams

## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## PERFORMANCE OVER TIME

Average performance at module level is not indicative.

- “phase behaviors” play a major role in programs.
- Applications settle down on these phases.

Disregarding such effects leads to grossly wrong estimations: the bounds for the event counters must be **time dependent**.

## PERFORMANCE UNDER VARIABILITY

Effects difficult to model in simulations:

- Same code paths run at different performance levels.
- Same initial states lead to different code paths.

Do not use averages, use **confidence intervals**.

## PERFORMANCE OVER TIME

Average performance at module level is not indicative.

- “phase behaviors” play a major role in programs.
- Applications settle down on these phases.

Disregarding such effects leads to grossly wrong estimations: the bounds for the event counters must be **time dependent**.

## PERFORMANCE UNDER VARIABILITY

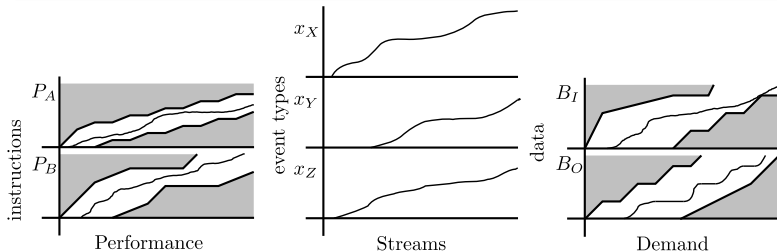
Effects difficult to model in simulations:

- Same code paths run at different performance levels.
- Same initial states lead to different code paths.

Do not use averages, use **confidence intervals**.

### FUNDAMENTAL QUESTION

Which performance bounds satisfy the demand bounds?



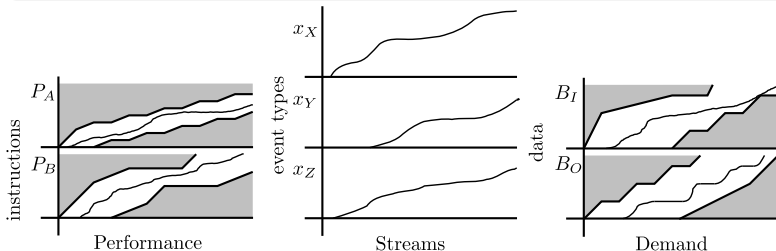
### NETWORK CALCULUS BOUNDS

Performance is characterized within a time dependent confidence interval by network calculus bounds.

- generalization to generic events: **real-time calculus**.

## FUNDAMENTAL QUESTION

Which performance bounds satisfy the demand bounds?



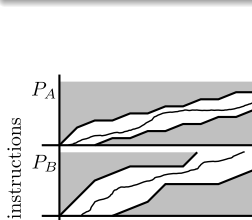
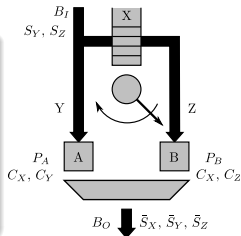
## NETWORK CALCULUS BOUNDS

Performance is characterized within a time dependent confidence interval by network calculus bounds.

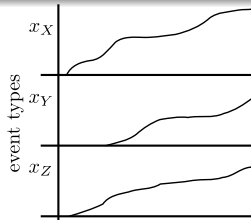
- generalization to generic events: **real-time calculus**.

### ANALYTICAL MODEL

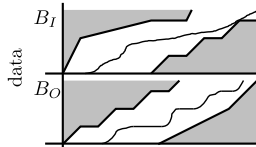
$$\left[ -K - \frac{\bar{S}_Z}{S_Z}, -K - \frac{\bar{S}_X}{C_X}, K, 1 \right] \begin{bmatrix} B_I \\ P_A \\ P_B \\ B_O \end{bmatrix} \leq 0$$



Performance



Streams



Demand

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- Modeling event streams using linear inequalities
- Time and variability in event streams

## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## SLIDING WINDOW UNDER LOAD

For every time window, determine

- the highest increment: arrival curve
- the lowest increment **under load**: service curve

## COMPLEX DETERMINATION

- must update all sliding windows at every simulation step
- must identify and skip idle conditions

## PIECEWISE LINEAR PROFILES

Rather than cycle by cycle, characterize event by event and assume linear growth between events.

- calculate and round intersections between linear pieces
- avoid accumulating rounding errors during incremental steps

## 1 OVERVIEW AND MOTIVATION

- The problem of sustaining the performance demand
- An introductory case study

## 2 THE ACHIEVABLE PERFORMANCE OF EVENT STREAMS

- What are event streams
- Modeling event streams using linear inequalities
- Time and variability in event streams

## 3 EMPIRICAL AND ANALYTICAL TECHNIQUES

- Empirical determination of network calculus bounds
- Analytical determination of sustained properties

## PROBLEM

$$\begin{aligned}
 S_Y X_Y + S_Z X_Z &\geq B_I \\
 C_X X_X + C_Y X_Y &\geq P_A \\
 C_X X_X + C_Z X_Z &\geq P_B \\
 \bar{S}_X X_X + \bar{S}_Y X_Y + \bar{S}_Z X_Z &\geq B_O
 \end{aligned}$$

## SOLUTION

$$\left[ -K - \frac{\bar{S}_Z}{S_Z}, -K - \frac{\bar{S}_X}{C_X}, K, 1 \right] \begin{bmatrix} B_I \\ P_A \\ P_B \\ B_O \end{bmatrix} \leq 0$$

## POLAR CONE OF THE DUAL SPACE OF THE INCONSISTENT PROBLEM

$$\begin{bmatrix} 0 & C_X & C_X & \bar{S}_X \\ S_Y & C_Y & 0 & \bar{S}_Y \\ S_Z & 0 & C_Z & \bar{S}_Z \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{bmatrix} = 0, \forall \pi_1 \geq 0, \pi_2 \geq 0, \pi_3 \geq 0, \pi_4 < 0$$

- the dual space is a line: 3 equations in four variables
- the polar cone is a halfspace, the parameters vector must remain out of that halfspace: in this case just sign inversion.

## A CONVEX HULL PROBLEM

Determining the **halfspace representation** of the prohibited regions for the parameters is a **convex hull** problem.

- simplified by the null space, but still complex.
- solved by double description method (CDD by Fukuda).
- performance strictly depending on the problem formulation.
- as a rule of thumb, in general no more than 20 variables.

## INVERSE PROBLEM: WHICH DEMAND CAN BE SUSTAINED?

Determine the space of redundant inequalities.

- employ the subsumption cone by Lassez
- solve by Fourier-Motzkin elimination (complex!)

Example on the next slide.

## TWO THREADS WITH OVERHEAD

Thread 1 guarantees a service  $b_1$ , thread 2 guarantees a service  $b_2$ . Either thread 1 or 2 is active, so  $b_3 > b_1 + b_2$ .

Which demand  $\alpha_1 x_1 + \alpha_2 x_2 \geq \beta$  can be sustained?

## AS INEQUALITY SYSTEM $Ax \geq b$

$$\begin{array}{rcl} & +x_1 & \geq b_1 \\ x_2 & & \geq b_2 \\ x_2 + x_1 & \geq b_3 \end{array}$$

Subsumption cone:

$$\{\pi A - \alpha = 0, \pi b - \beta \geq 0, \pi \geq 0\}$$

## SUBSUMPTION CONE

$$\begin{array}{rcccccl} & \pi_2 & +\pi_3 & -\alpha_1 & = & 0 \\ \pi_1 & & +\pi_3 & -\alpha_2 & = & 0 \\ b_1 \pi_1 & +b_2 \pi_2 & +b_3 \pi_3 & -\beta & \geq & 0 \\ \pi_1 & & & & \geq & 0 \\ & \pi_2 & & & \geq & 0 \\ & & \pi_3 & & \geq & 0 \end{array}$$

## FOURIER-MOTZKIN ELIMINATION

$$\begin{array}{rcl} \pi_3(b_3 - b_2 - b_1) + \alpha_1 b_2 + \alpha_2 b_1 - \beta & \geq & 0 \\ & -\pi_3 & \geq \alpha_1 \\ & -\pi_3 & \geq -\alpha_2 \\ & \pi_3 & \geq 0 \end{array}$$

The coefficient of  $\pi_3$  is  $b_3 - b_2 - b_1 \geq 0$  per construction.

## TWO THREADS WITH OVERHEAD

Thread 1 guarantees a service  $b_1$ , thread 2 guarantees a service  $b_2$ . Either thread 1 or 2 is active, so  $b_3 > b_1 + b_2$ .

Which demand  $\alpha_1 x_1 + \alpha_2 x_2 \geq \beta$  can be sustained?

## AS INEQUALITY SYSTEM $Ax \geq b$

$$\begin{array}{rcl} & +x_1 & \geq b_1 \\ x_2 & & \geq b_2 \\ x_2 + x_1 & \geq & b_3 \end{array}$$

Subsumption cone:

$$\{\pi A - \alpha = 0, \pi b - \beta \geq 0, \pi \geq 0\}$$

## SUBSUMPTION CONE

$$\begin{array}{rcccccc} & & \pi_2 & +\pi_3 & -\alpha_1 & = & 0 \\ \pi_1 & & & +\pi_3 & -\alpha_2 & = & 0 \\ b_1 \pi_1 & +b_2 \pi_2 & +b_3 \pi_3 & & -\beta & \geq & 0 \\ \pi_1 & & & & & \geq & 0 \\ & \pi_2 & & & & \geq & 0 \\ & & \pi_3 & & & \geq & 0 \end{array}$$

## FOURIER-MOTZKIN ELIMINATION

$$\begin{array}{rcl} \pi_3(b_3 - b_2 - b_1) + \alpha_1 b_2 + \alpha_2 b_1 - \beta & \geq & 0 \\ & -\pi_3 & \geq \alpha_1 \\ & -\pi_3 & \geq -\alpha_2 \\ & \pi_3 & \geq 0 \end{array}$$

The coefficient of  $\pi_3$  is  $b_3 - b_2 - b_1 \geq 0$  per construction.

## TWO THREADS WITH OVERHEAD

Thread 1 guarantees a service  $b_1$ , thread 2 guarantees a service  $b_2$ . Either thread 1 or 2 is active, so  $b_3 > b_1 + b_2$ .

Which demand  $\alpha_1 x_1 + \alpha_2 x_2 \geq \beta$  can be sustained?

AS INEQUALITY SYSTEM  $Ax \geq b$ 

$$\begin{array}{rcl} & +x_1 & \geq b_1 \\ x_2 & & \geq b_2 \\ x_2 & +x_1 & \geq b_3 \end{array}$$

Subsumption cone:

$$\{\pi A - \alpha = 0, \pi b - \beta \geq 0, \pi \geq 0\}$$

## SUSTAINABLE DEMAND SPACE

$$-\alpha_2 b_1 - \alpha_1 (b_3 - b_1) + \beta \leq 0$$

$$-\alpha_2 (b_3 - b_2) - \alpha_1 b_2 + \beta \leq 0$$

$$\alpha_1 \geq 0$$

$$\alpha_2 \geq 0$$

Dimensioning  
Event Streams

Di Gregorio

Motivation

Problem  
Example

Event Streams

Definition  
Modeling  
Bounds over time

Techniques

Empirical  
Analytical

# Thank you!

Lorenzo Di Gregorio  
lorenzo.digregorio@lantiq.com



Komei Fukuda and Alain Prodon.

The double description method revisited.



Harvey J. Greenberg.

Consistency, redundancy, and implied equalities in linear systems.

*Annals of Mathematics and Artificial Intelligence*, vol. 17, pp. 37–83, 1996



Alaa R. Alameldeen and David A. Wood.

Variability in Architectural Simulations of Multithreaded Workloads

*Proceedings of the Ninth IEEE Symposium on High-Performance Computer Architecture*, 2003