Module Modeling and Analysis: Inputs and Uncertainties

by *Gerrit Muller* LogoESIbuskerud
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

This module addresses Modeling and Analysis: Inputs and Uncertainties. The input for models comes from different sources: facts obtained from market and technology research, data from measurements, and assumptions. All these sources have uncertainties and may hide unknowns, or may even be wrong. We zoom in on commonly used technology.

March 6, 2021
status: planned
version: 0.6

Module Content

goal of this module

Provide foundation and figures of merit for technology modeling

Provide insight in the inputs of models

Provide measurement fundamentals

content of this module

problem statement

generic layering and block diagrams

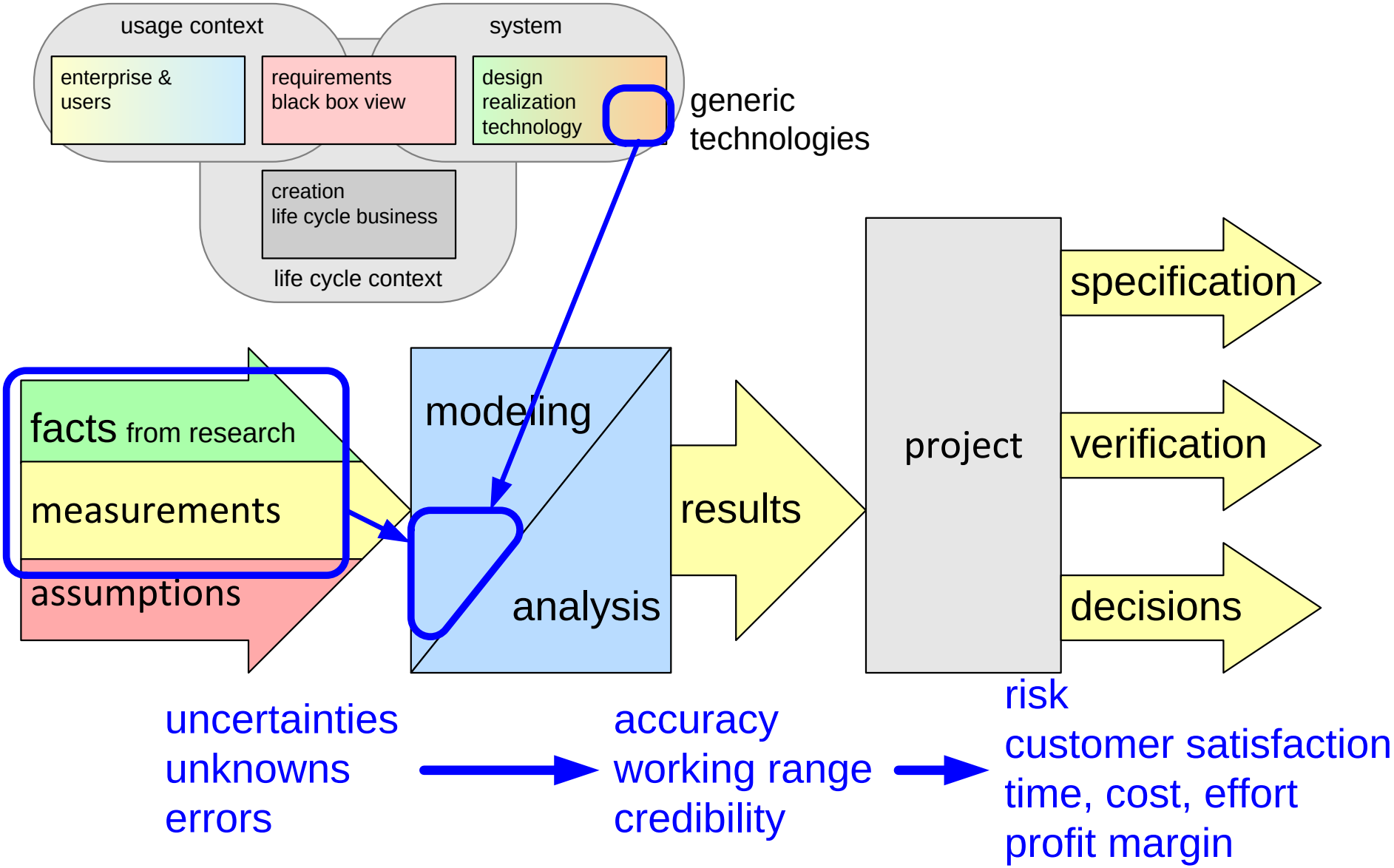
measuring HW and SW

exercise

measurement of loop and file open performance

participants may chose their own programming environment or Python

Where are we in the Course?

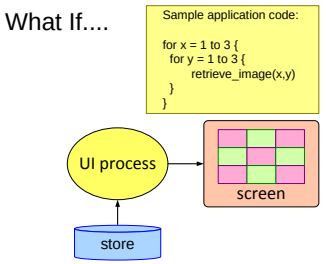


Introduction to System Performance Design

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

What is System Performance? Why should a software engineer have knowledge of the other parts of the system, such as the Hardware, the Operating System and the Middleware? The applications that he/she writes are self-contained, so how can other parts have any influence? This introduction sketches the problem and shows that at least a high level understanding of the system is very useful in order to get optimal performance.



March 6, 2021
status: preliminary draft
version: 0.5

Content of Problem Introduction

content of this presentation

Example of problem

Problem statements

Image Retrieval Performance

application need:

at event 3*3 show 3*3 images instanteneous

design

design

Sample application code:

```
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}
```

or

alternative application code:

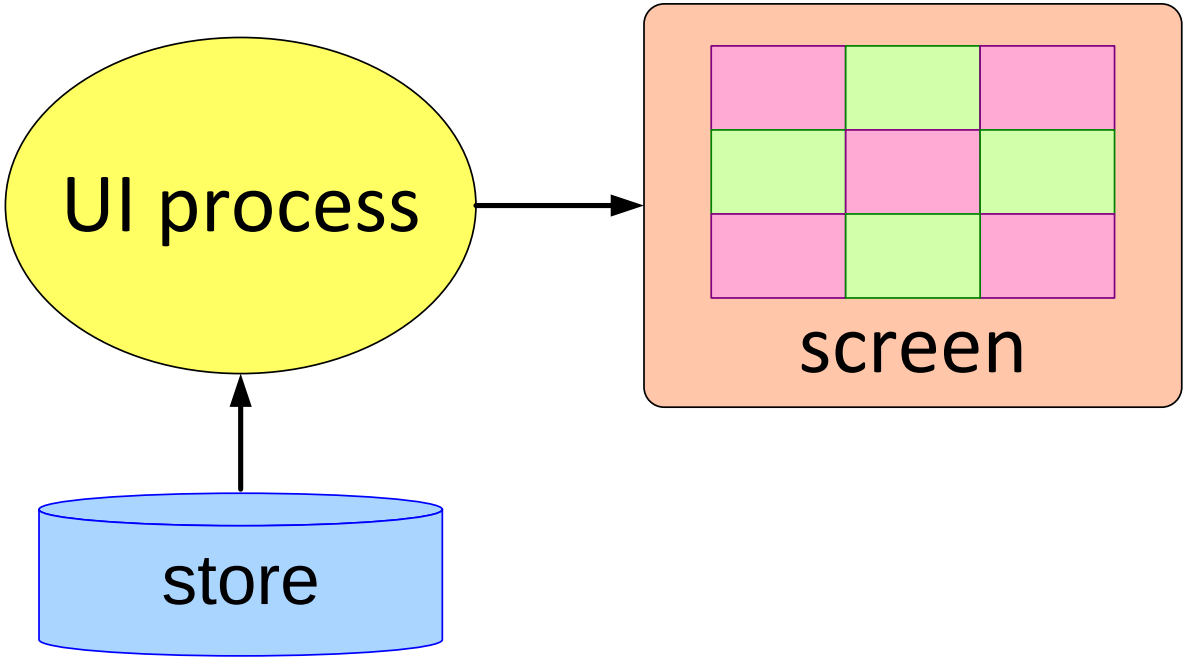
event 3*3 -> show screen 3*3

```
<screen 3*3>
    <row 1>
    <col 1><image 1,1></col 1>
    <col 2><image 1,2></col 2>
    <col 3><image 1,3></col 3>
  </row 1>
  <row 2>
    <col 1><image 1,1></col 1>
    <col 2><image 1,2></col 2>
    <col 3><image 1,3></col 3>
  </row 1>
  <row 2>
    <col 1><image 1,1></col 1>
    <col 2><image 1,2></col 2>
    <col 3><image 1,3></col 3>
  </row 3>
</screen 3*3>
```

What If....

Sample application code:

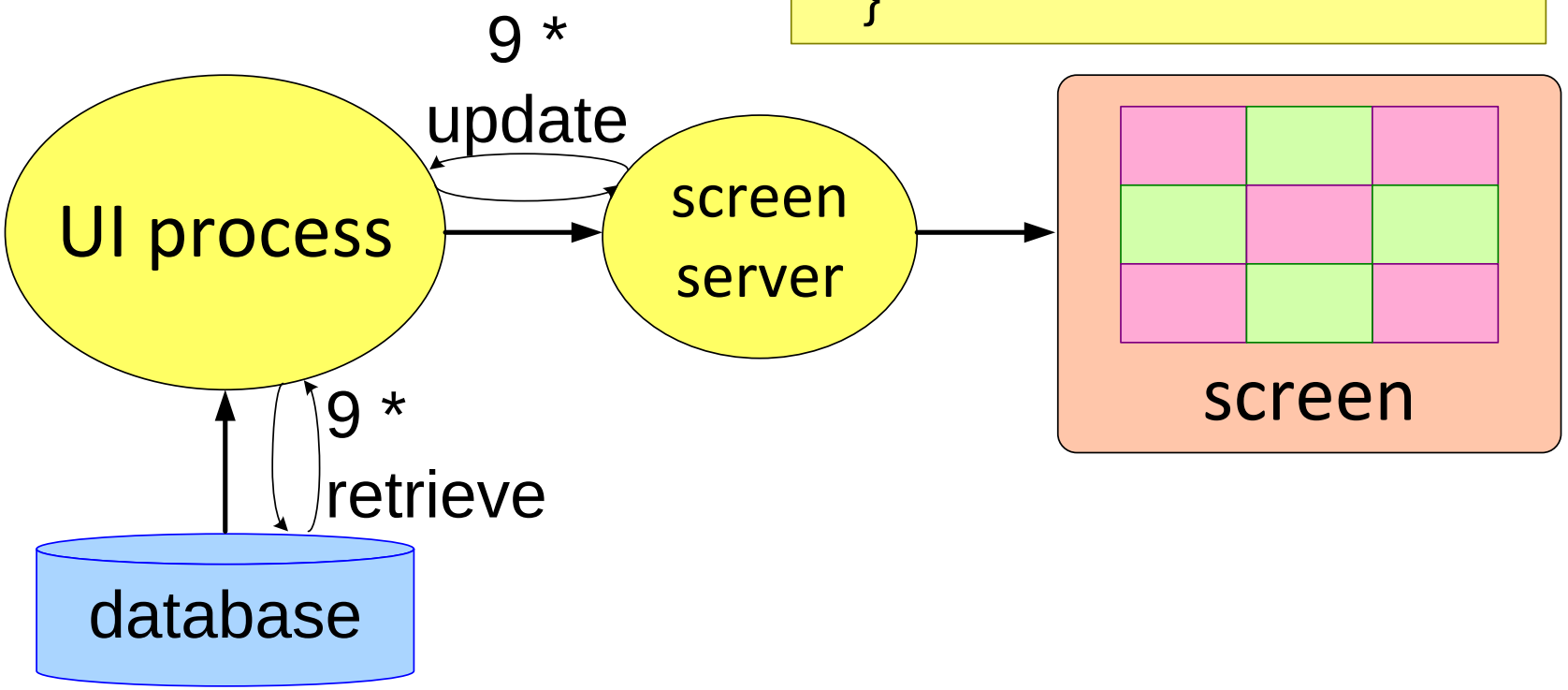
```
for x = 1 to 3 {
  for y = 1 to 3 {
        retrieve_image(x,y)
  }
}
```



What If....

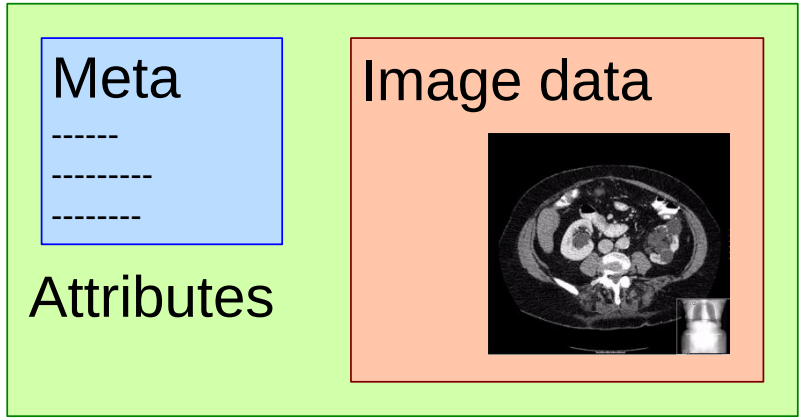
Sample application code:

```
for x = 1 to 3 {
  for y = 1 to 3 {
        retrieve_image(x,y)
  }
}
```



Meta Information Realization Overhead

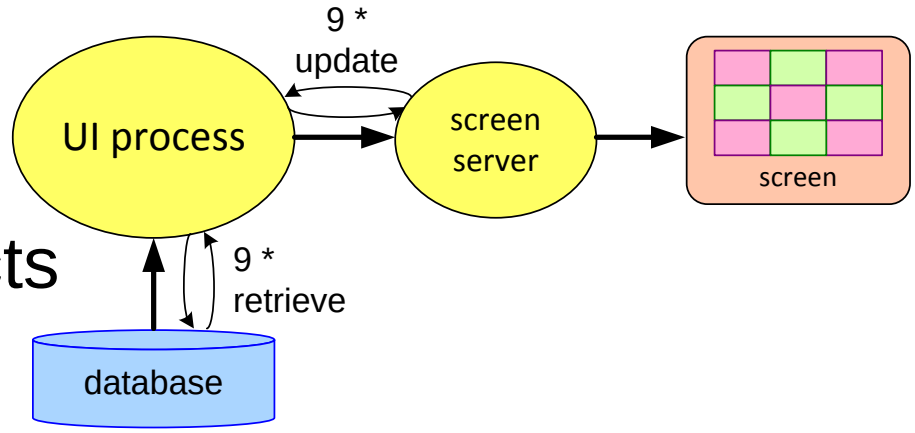
What If....



Sample application code:

```
for x = 1 to 3 {
  for y = 1 to 3 {
        retrieve_image(x,y)
  }
}
```

Attribute = 1 COM object
100 attributes / image
9 images = 900 COM objects
1 COM object = 80µs
9 images = 72 ms



What If....

Sample application code:

```
for x = 1 to 3 {
  for y = 1 to 3 {
        retrieve_image(x,y)
  }
}
```

- I/O on line basis (512^2 image)

$$9 * 512 * t_{I/O}$$

$$t_{I/O} \sim= 1ms$$

- . . .

Non Functional Requirements Require System View

Sample application code:

```
for x = 1 to 3 {
  for y = 1 to 3 {
        retrieve_image(x,y)
  }
}
```

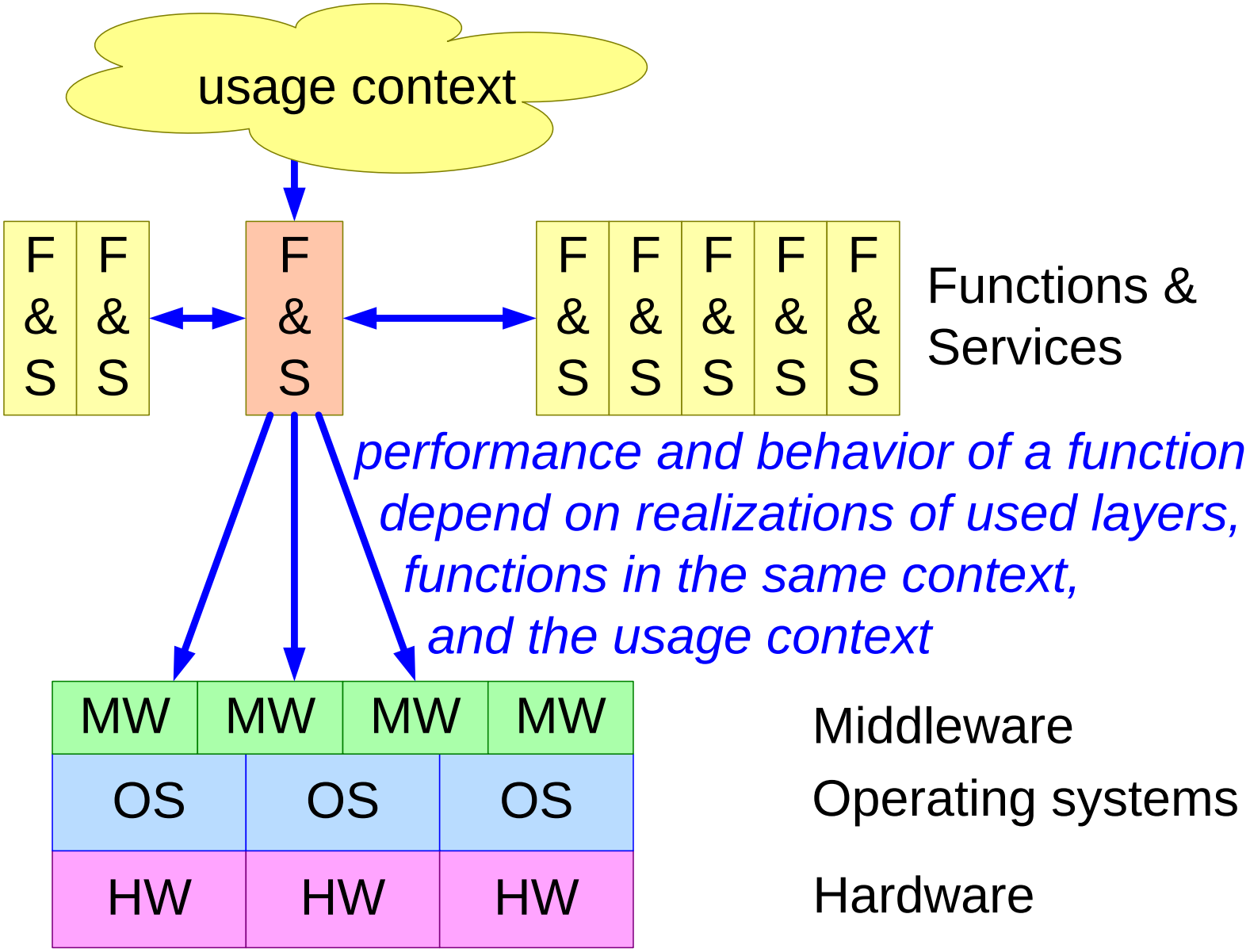
can be:

- fast, but very local
- slow, but very generic
- slow, but very robust
- fast and robust
- ...

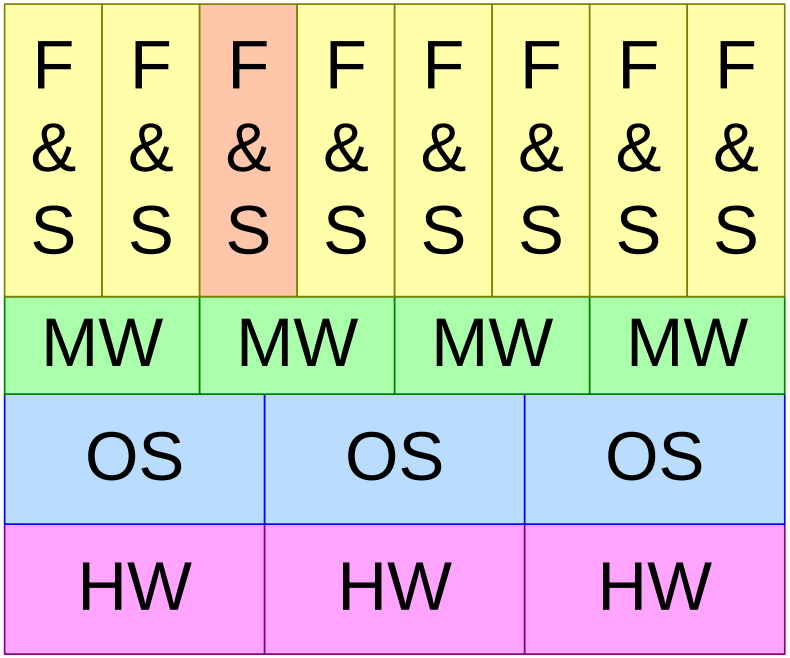
The emerging properties (behavior, performance) cannot be seen from the code itself!

Underlying platform and neighbouring functions determine emerging properties mostly.

Function in System Context



Challenge



Functions & Services

Middleware

Operating systems

Hardware

Performance = Function (F&S, other F&S, MW, OS, HW)
MW, OS, HW >> 100 Manyear : very complex

Challenge: How to understand MW, OS, HW
with only a few parameters

Summary of Problem Introduction

Summary of Introduction to Problem

Resulting System Characteristics cannot be deduced from local code.

Underlying platform, neighboring applications and user context:

have a big impact on system characteristics

are big and complex

Models require decomposition, relations and representations to analyse.

Intermezzo: So Why to Model and Analyse?

Why do we model?

- what are indicators that modeling and analysis beyond "business as usual" architecture is needed.

- What questions trigger Modeling and Analysis.

The answer to the question from business side is *not evident*

The answer is business *critical* (e.g. poor performance -> unusable service). We did not discuss business value for this case.

Past experience shows that design choices have big impact on the outcome, in other words this part of the design is *critical*

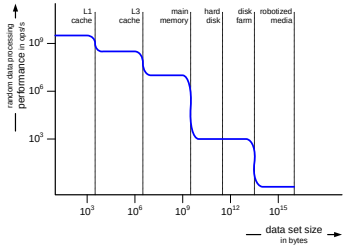
Modeling and Analysis Fundamentals of Technology

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

This presentation shows fundamental elements for models that are ICT-technology related. Basic hardware functions are discussed: storage, communication and computing with fundamental characteristics, such as throughput, latency, and capacity. A system is build by layers of software on top of hardware. The problem statement is how to reason about system properties, when the system consists of many layers of hardware and software.

March 6, 2021
status: preliminary draft
version: 0.5



content of this presentation

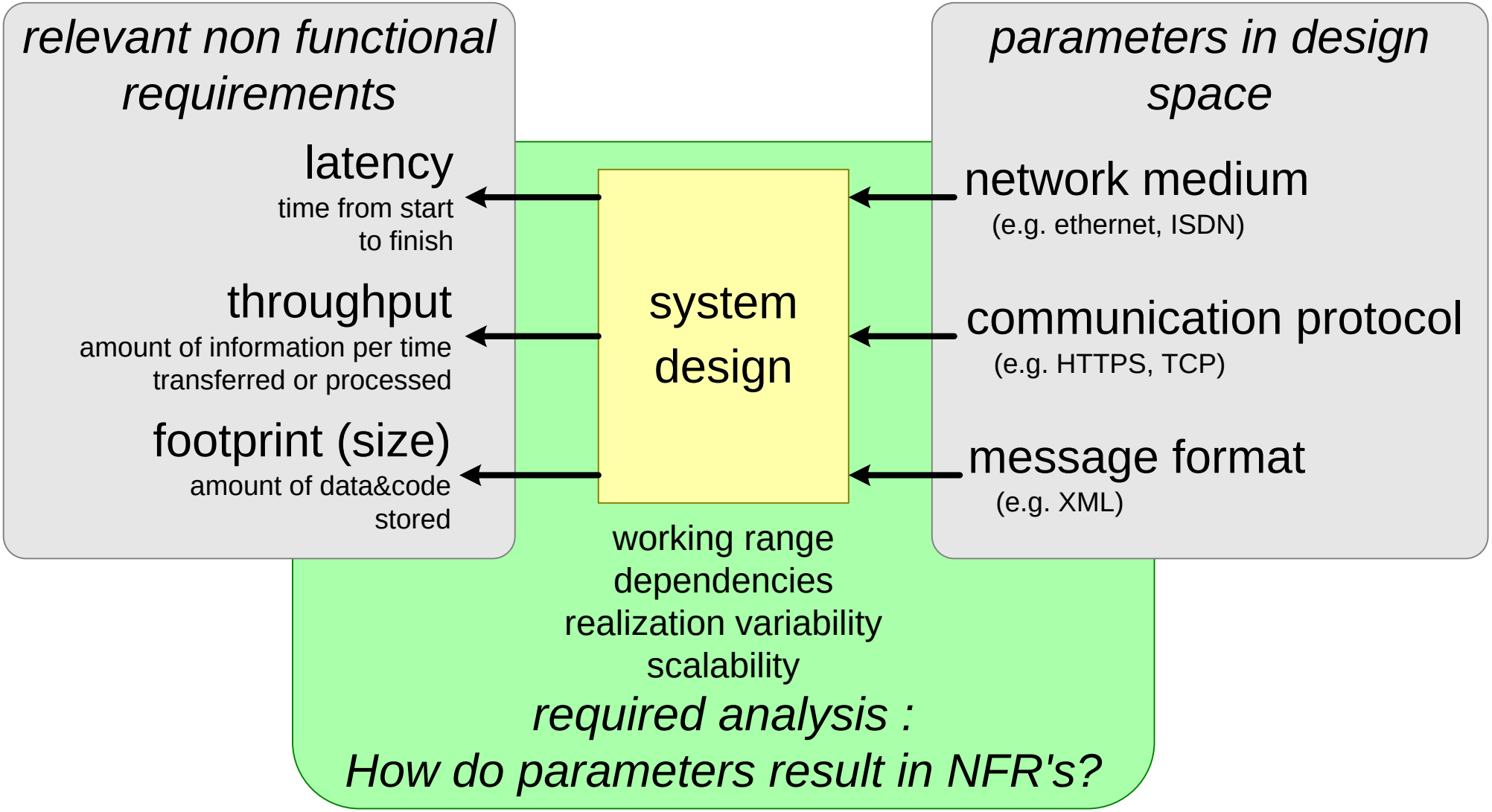
generic layering and block diagrams

typical characteristics and concerns

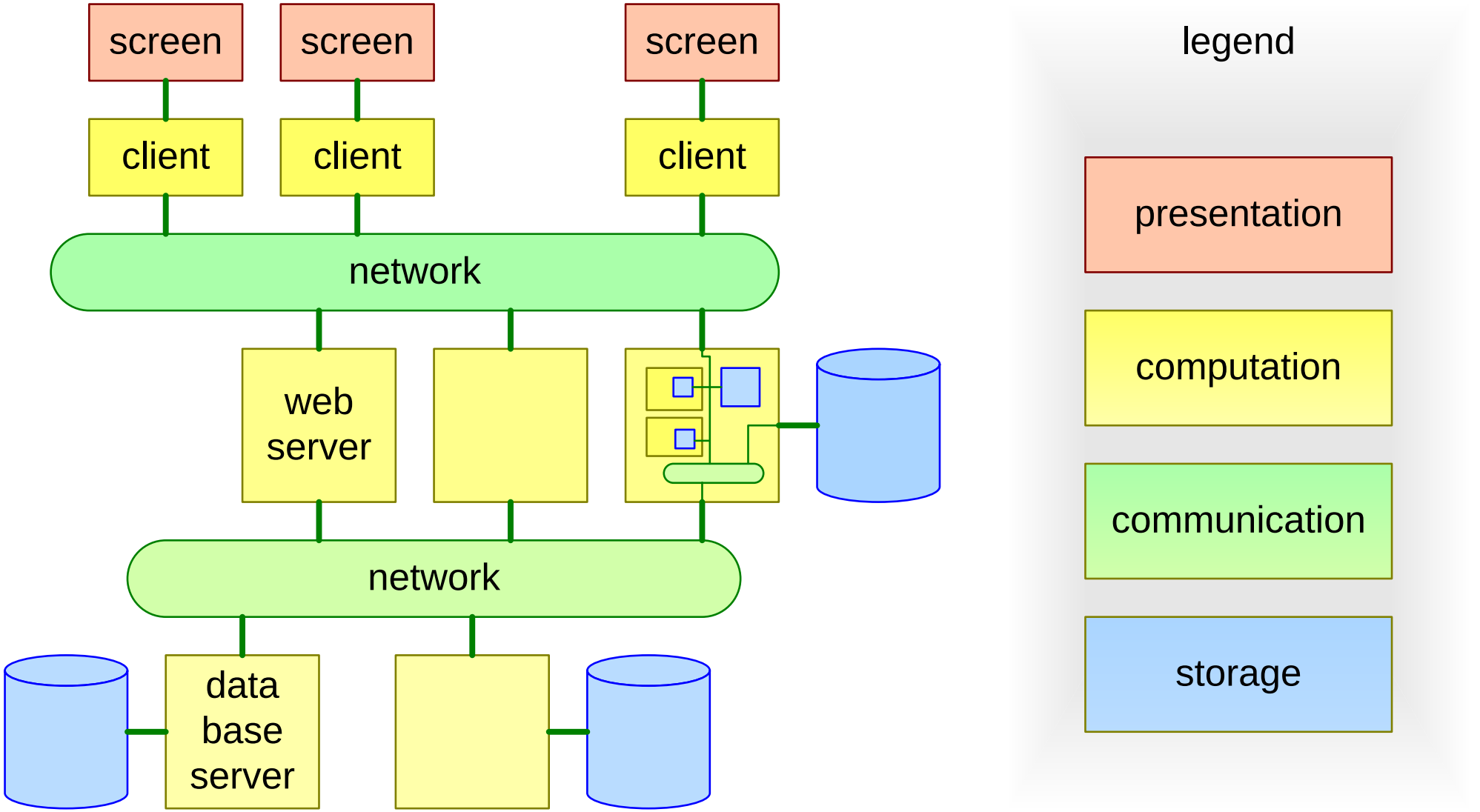
figures of merit

example of picture caching in web shop application

What do We Need to Analyze?



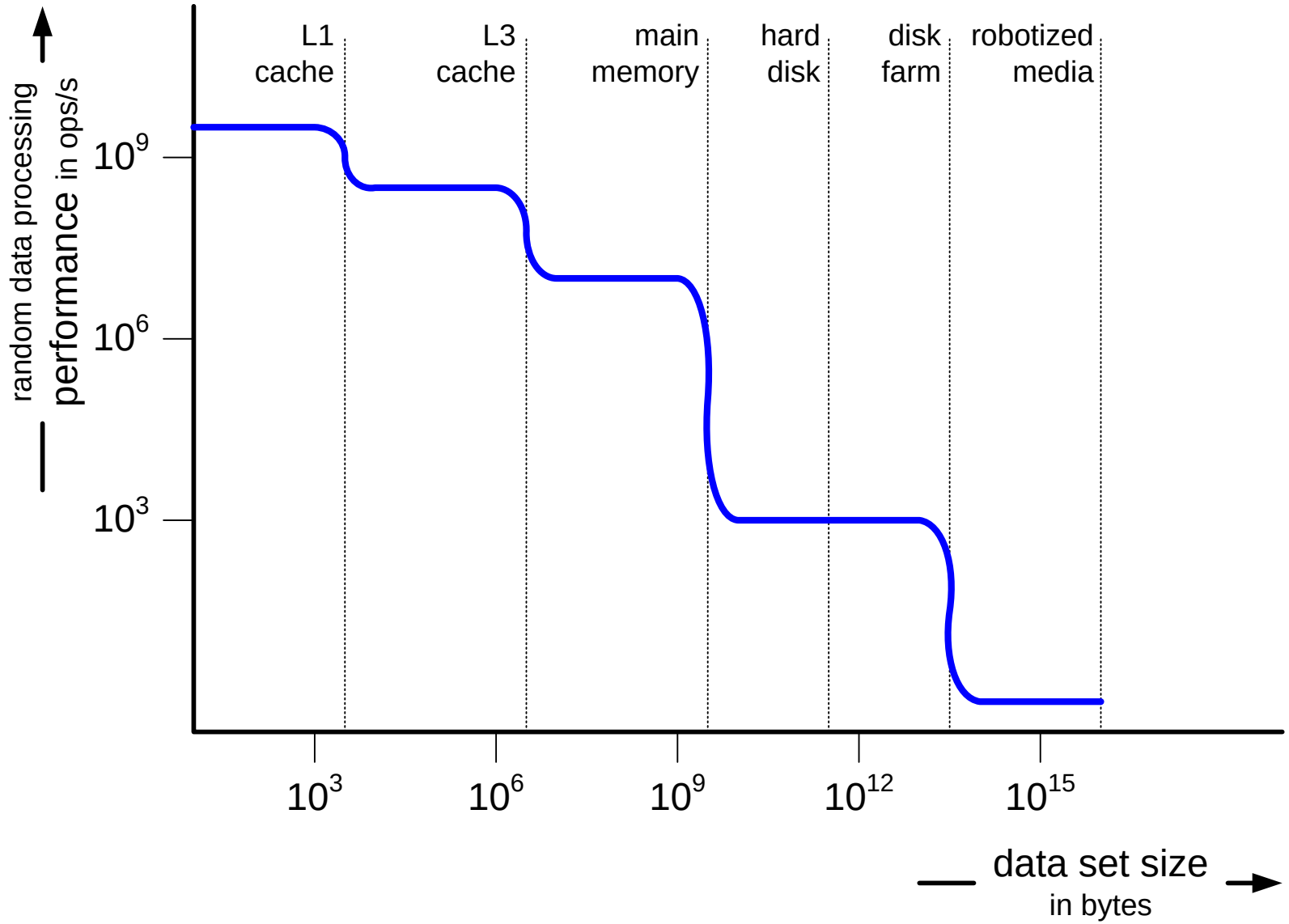
Typical Block Diagram and Typical Resources



Hierarchy of Storage Technology Figures of Merit

		latency	capacity
processor cache	*L1 cache*	sub ns	n kB
	L2 cache		
	L3 cache	ns	n MB
fast volatile	*main memory*	tens ns	n GB
persistent	*disks*		n*100 GB
	disk arrays	ms	
	disk farms		n*10 TB
archival	*robotized*		
	optical media	>s	n PB
	tape		

Performance as Function of Data Set Size



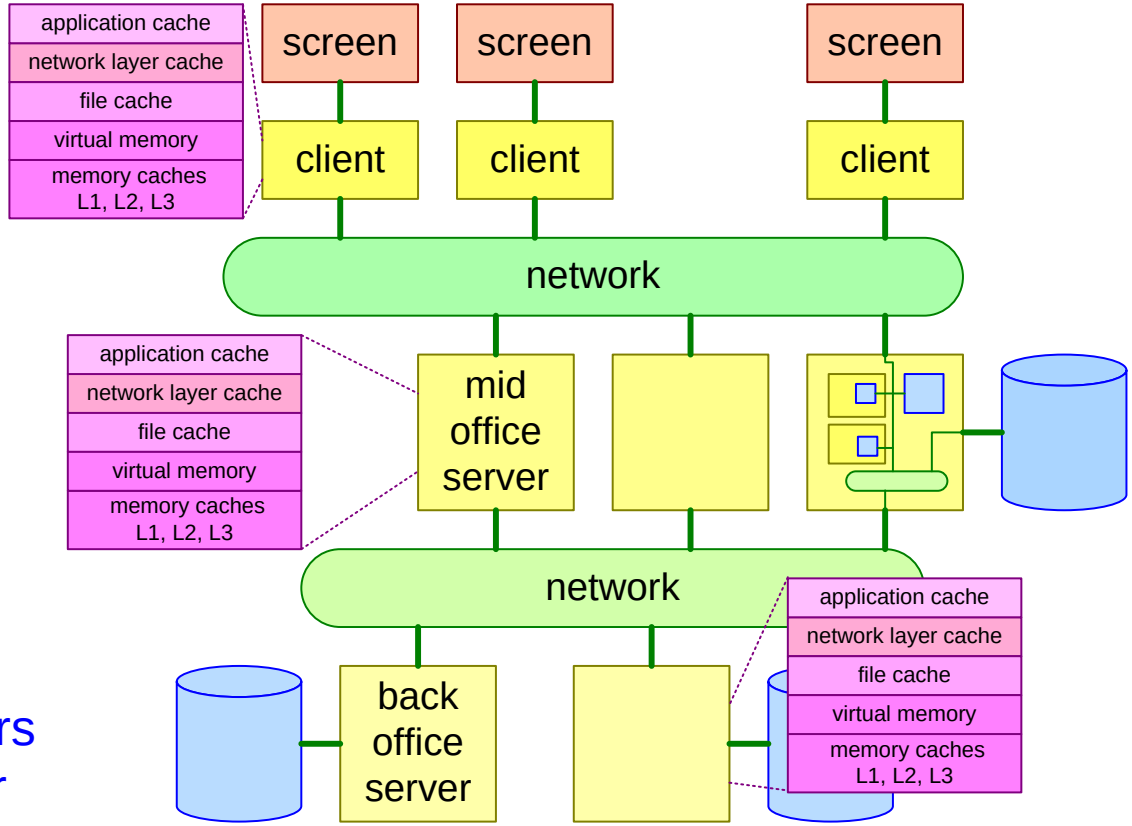
Communication Technology Figures of Merit

		latency	frequency	distance
on chip	*connection*	sub ns	n GHz	n mm
	network	n ns	n GHz	n mm
PCB level		tens ns	n 100MHz	n cm
Serial I/O		n ms	n 100MHz	n m
network	*LAN*	n ms	100MHz	n km
	WAN	n 10ms	n GHz	global

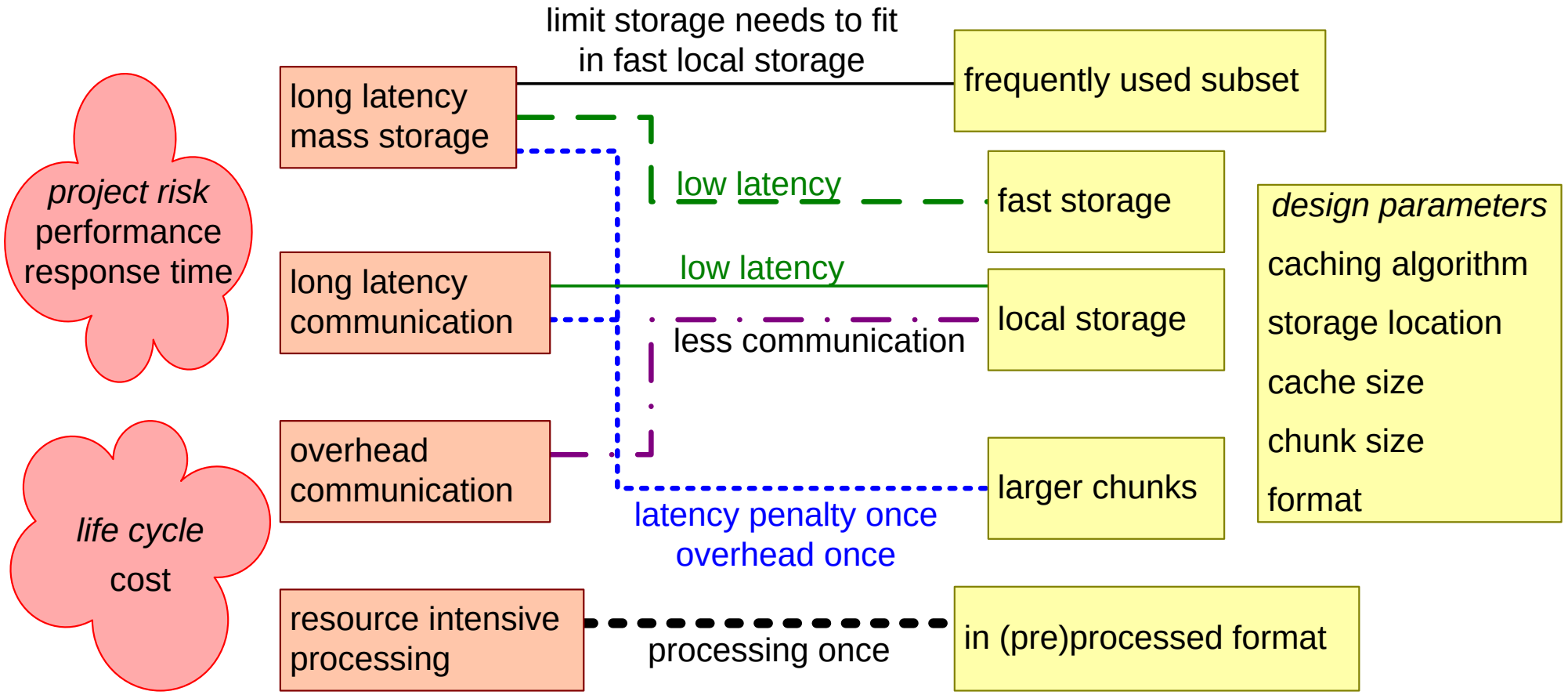
Multiple Layers of Caching

	cache miss penalty	cache hit performance
application cache	1 s	10 ms
network layer cache	100 ms	1 ms
file cache	10 ms	10 µs
virtual memory	1 ms	100 ns
memory caches L1, L2, L3	100 ns	1 ns

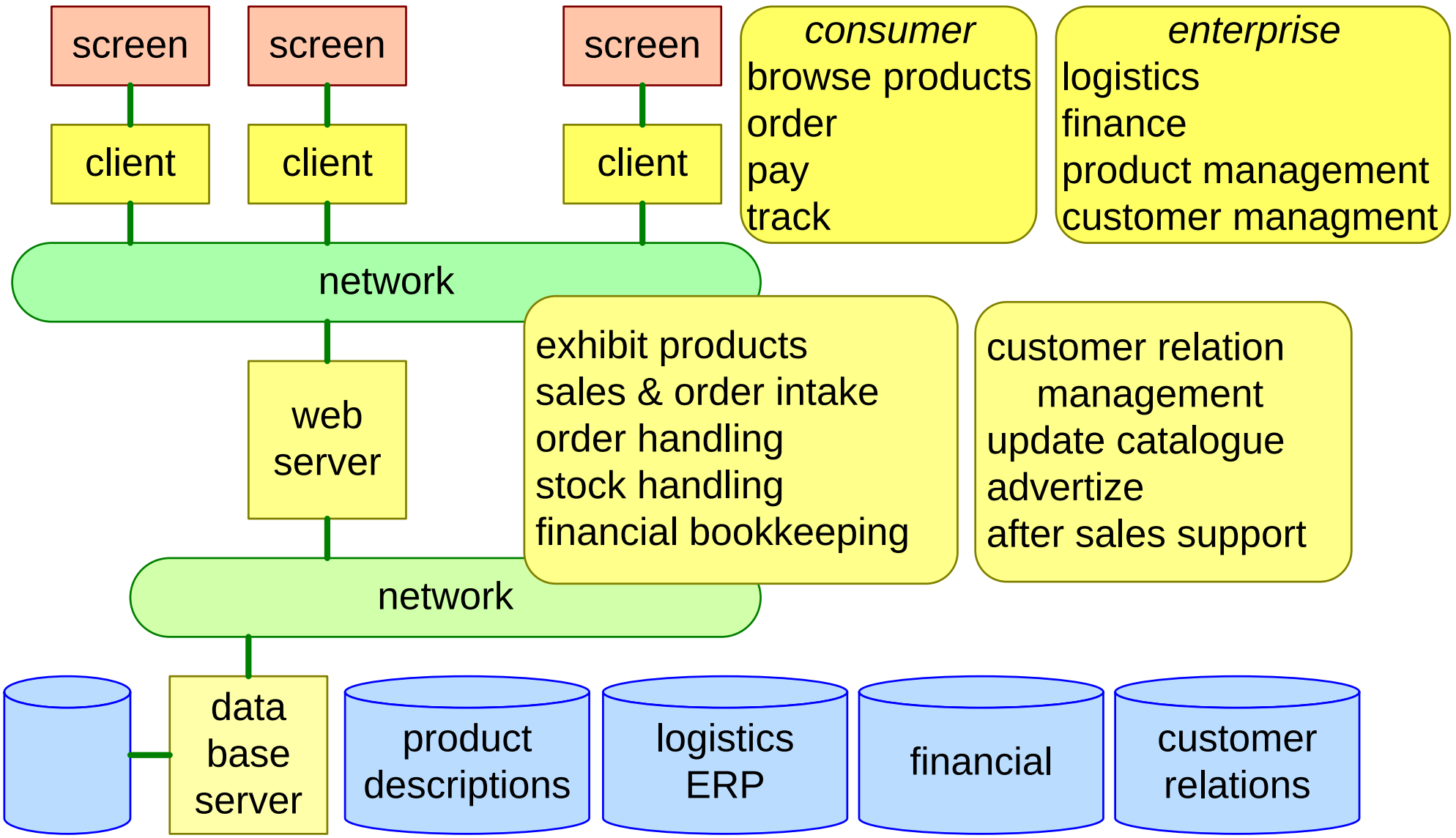
typical cache 2 orders of magnitude faster



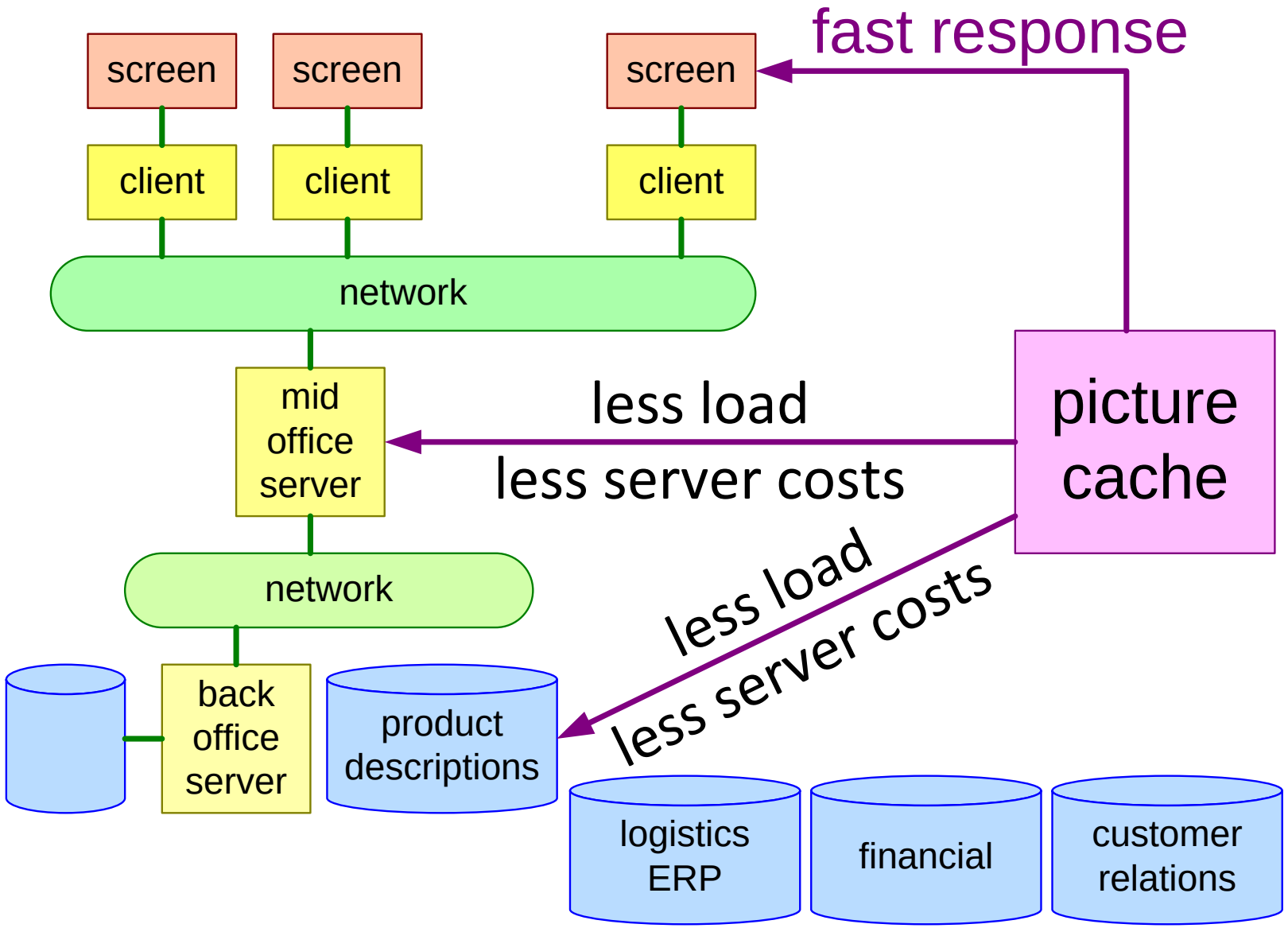
Why Caching?


limit storage needs to fit
in fast local storage
long latency
mass storage
frequently used subset
low latency
fast storage
project risk
performance
response time
long latency
communication
low latency
local storage
less communication
design parameters
caching algorithm
storage location
cache size
chunk size
format
overhead
communication
larger chunks
latency penalty once
overhead once
life cycle
cost
resource intensive
processing
processing once
in (pre)processed format

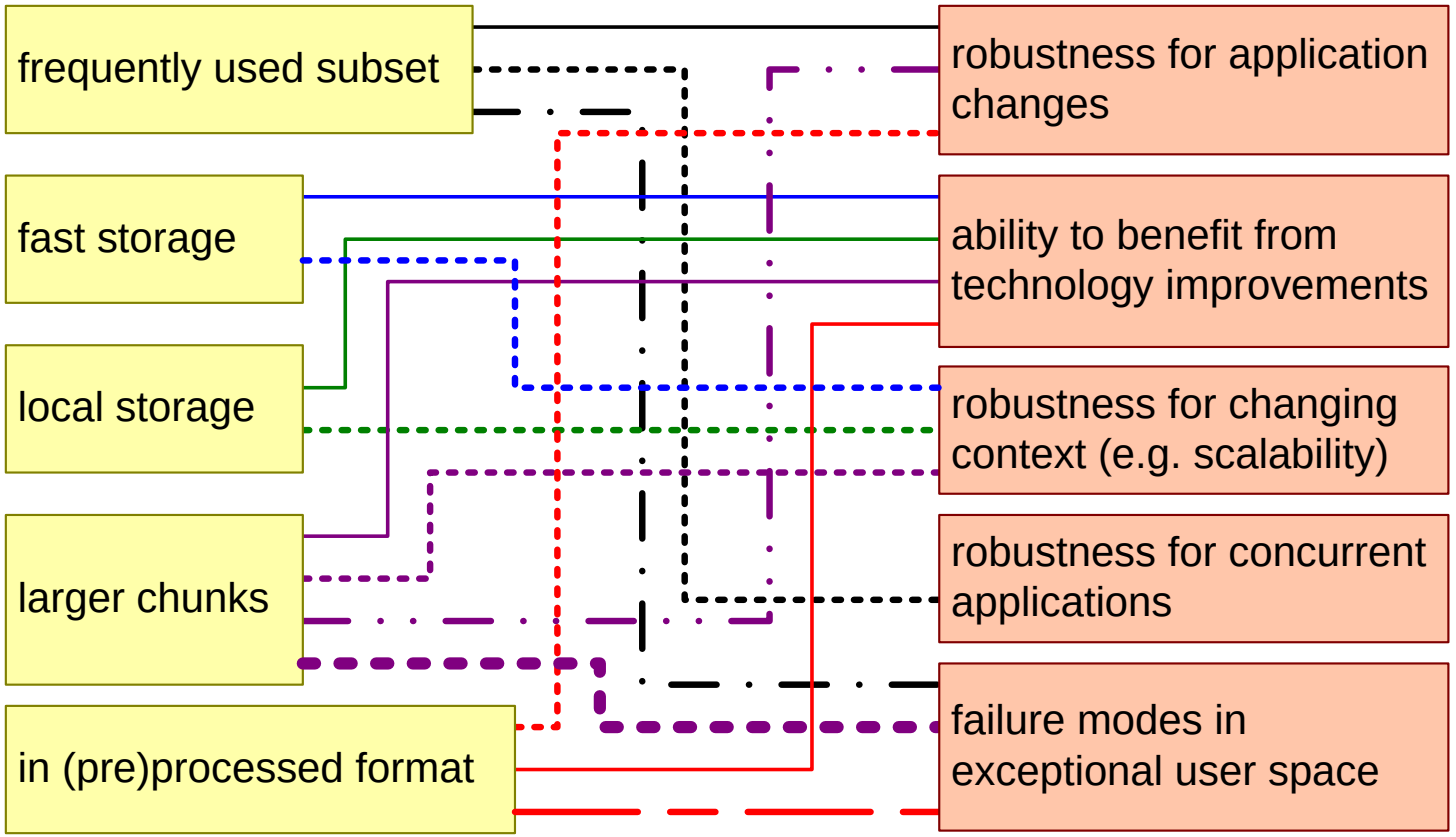
Example Web Shop



Impact of Picture Cache



Risks of Caching



life cycle
cost
effort

project risk
cost
effort
performance

Summary

Conclusions

Technology characteristics can be discontinuous

Caches are an example to work around discontinuities

Caches introduce complexity and decrease transparancy

Techniques, Models, Heuristics of this module

Generic block diagram: Presentation, Computation, Communication and Storage

Figures of merit

Local reasoning (e.g. cache example)

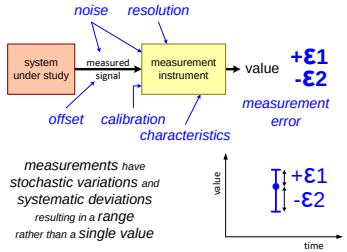
Modeling and Analysis: Measuring

by *Gerrit Muller* University of South-Eastern Norway-SE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

This presentation addresses the fundamentals of measuring: What and how to measure, impact of context and experiment on measurement, measurement errors, validation of the result against expectations, and analysis of variation and credibility.

March 6, 2021
status: preliminary draft
version: 1.2



Presentation Content

content

What and How to measure

Impact of experiment and context on measurement

Validation of results, a.o. by comparing with expectation

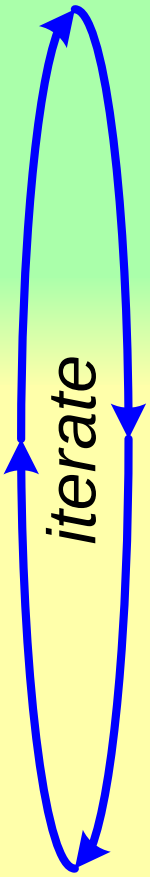
Consolidation of measurement data

Analysis of variation and analysis of credibility

Measuring Approach: What and How

what

1. What do we need to know?	
2. Define quantity to be measured.	initial model
3. Define required accuracy	purpose
4A. Define the measurement circumstances	fe.g. by use cases
4B. Determine expectation	historic data or estimation
4C. Define measurement set-up	
5. Determine actual accuracy	uncertainties, measurement error
6. Start measuring	
7. Perform sanity check	expectation versus actual outcome



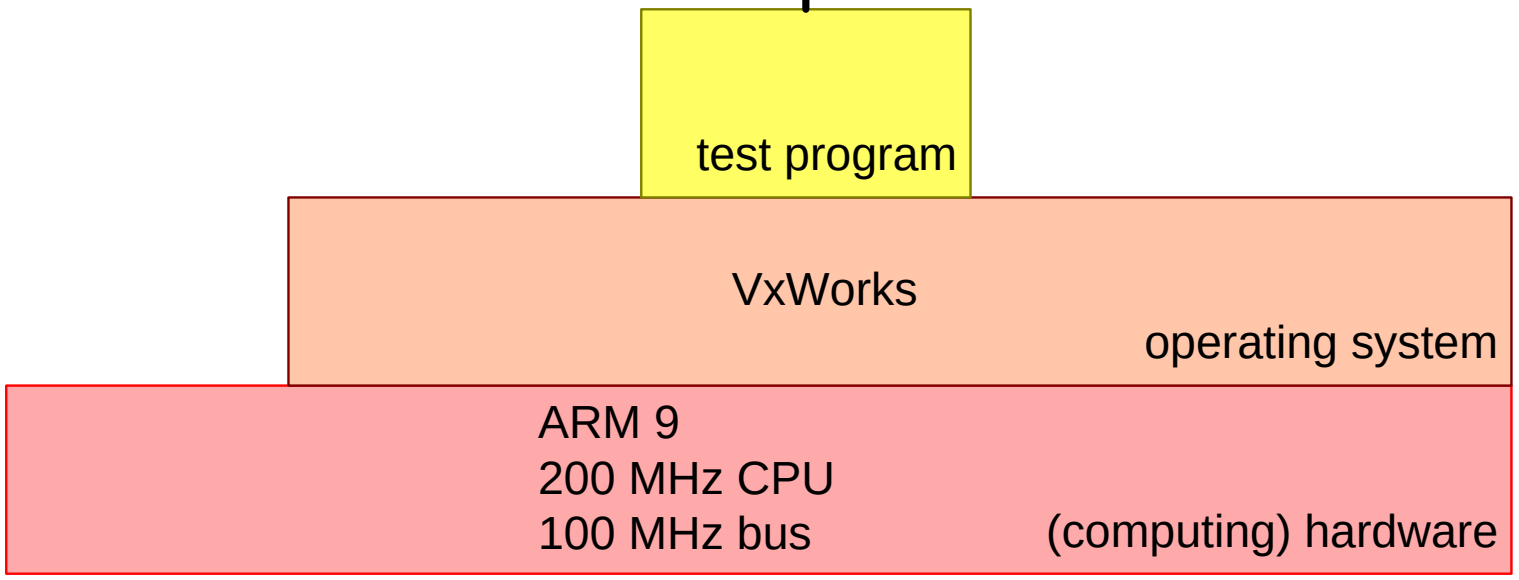
how

1. What do We Need? Example Context Switching

guidance of concurrency design and task granularity

estimation of total lost CPU time due to context switching

What:
context switch time of VxWorks running on ARM9

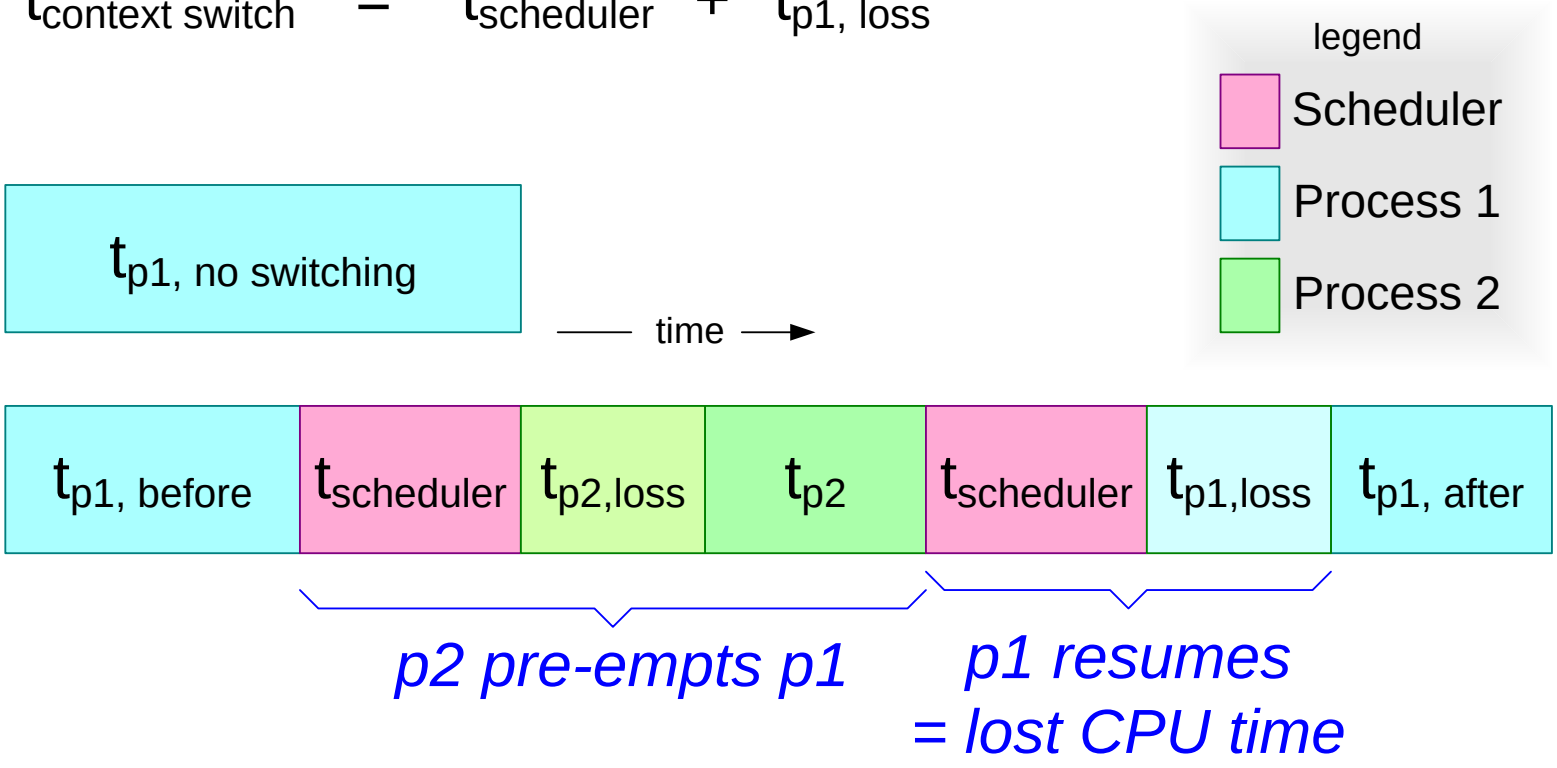


2. Define Quantity by Initial Model

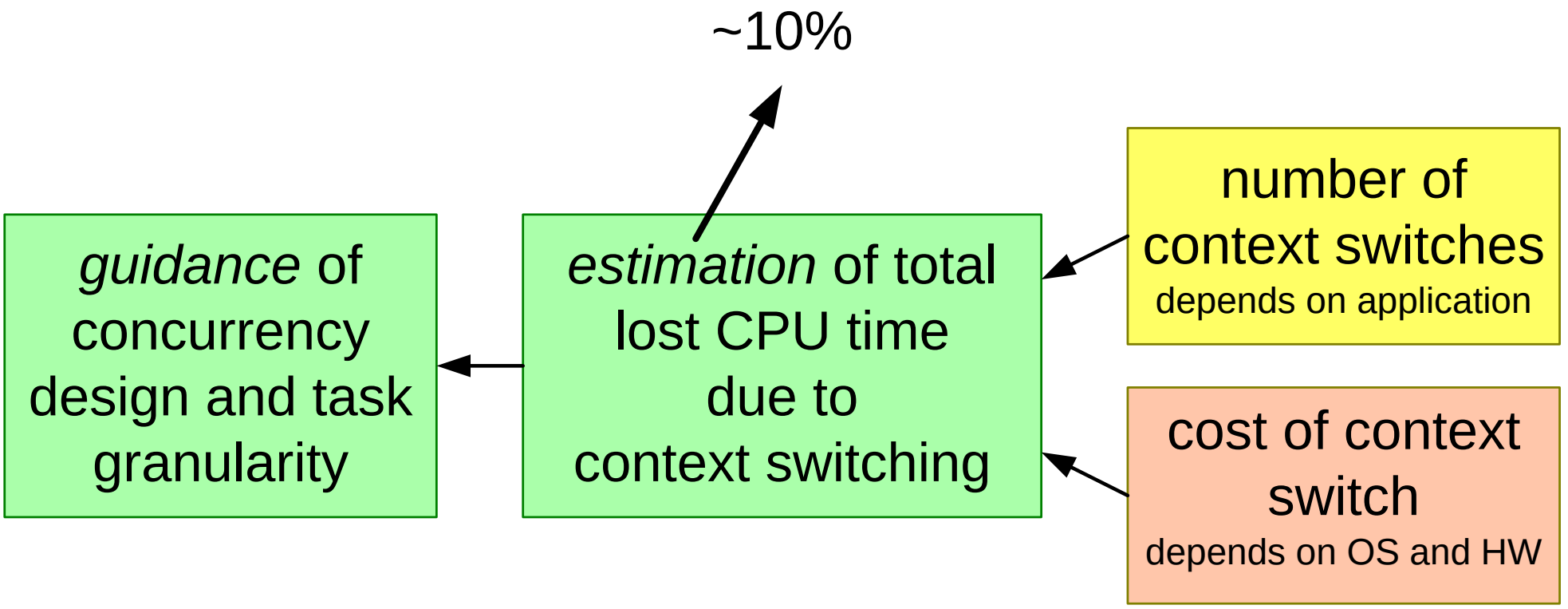
What (original):
context switch time of
VxWorks running on ARM9

What (more explicit):
The amount of lost CPU time,
due to context switching on
VxWorks running on ARM9
on a heavy loaded CPU

$t_{context\ switch} = t_{scheduler} + t_{p1,\ loss}$

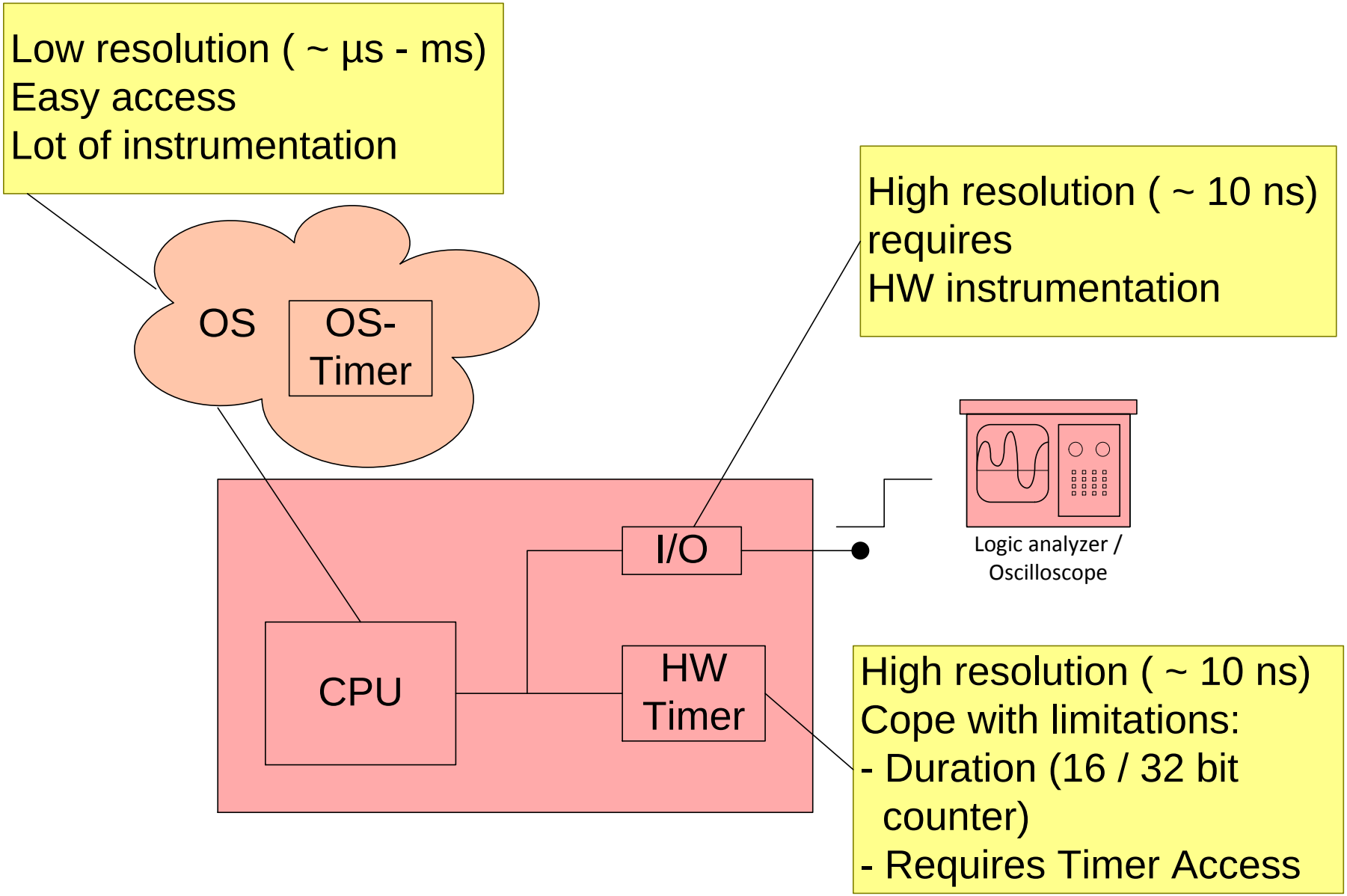


3. Define Required Accuracy



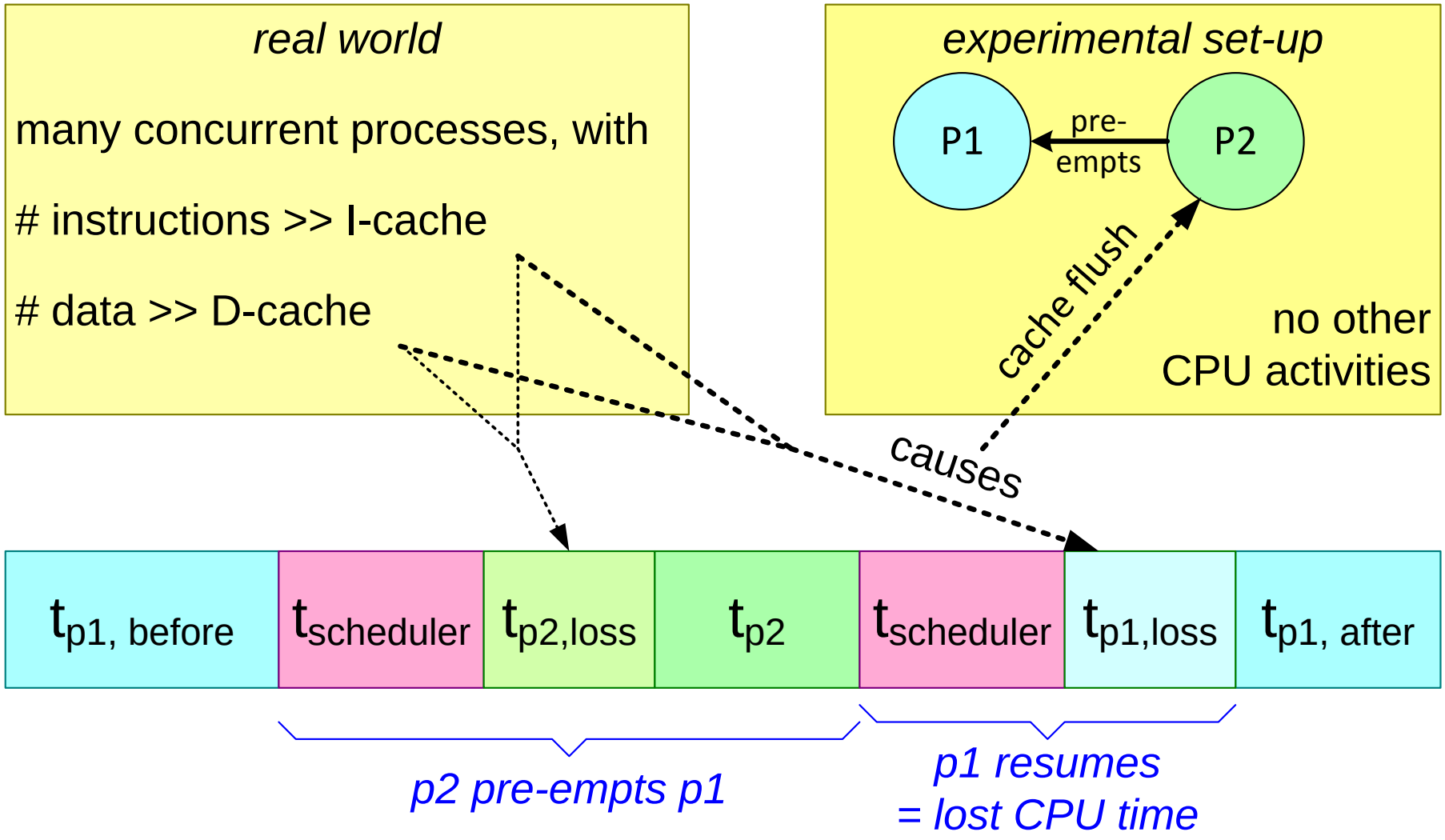
purpose drives required accuracy

Intermezzo: How to Measure CPU Time?

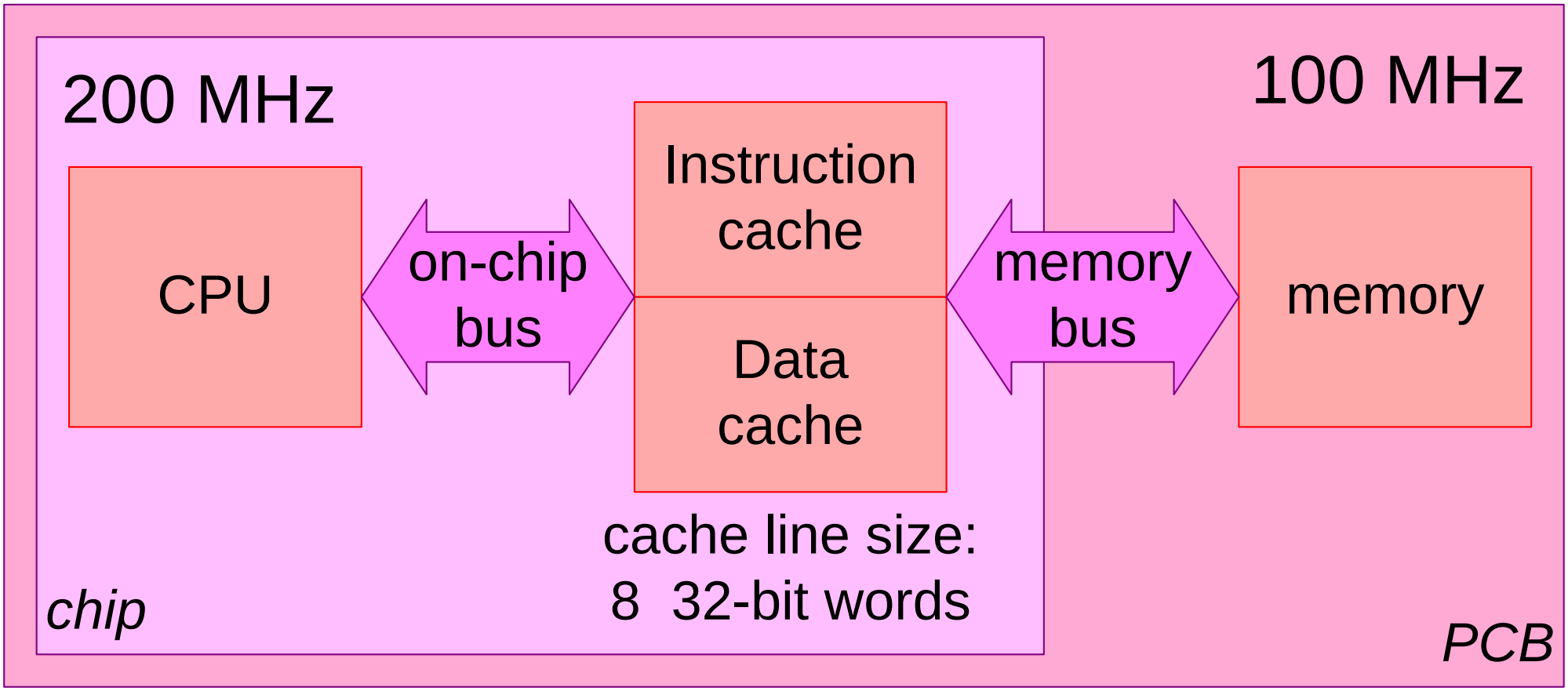


4A. Define the Measurement Set-up

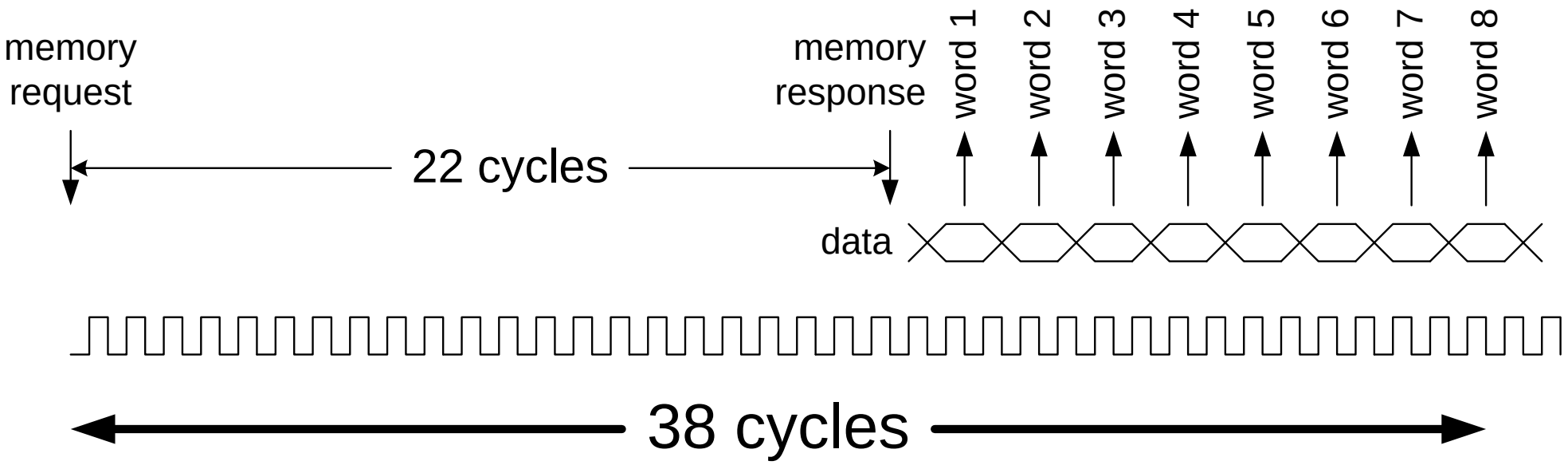
Mimick relevant real world characteristics



4B. Case: ARM9 Hardware Block Diagram

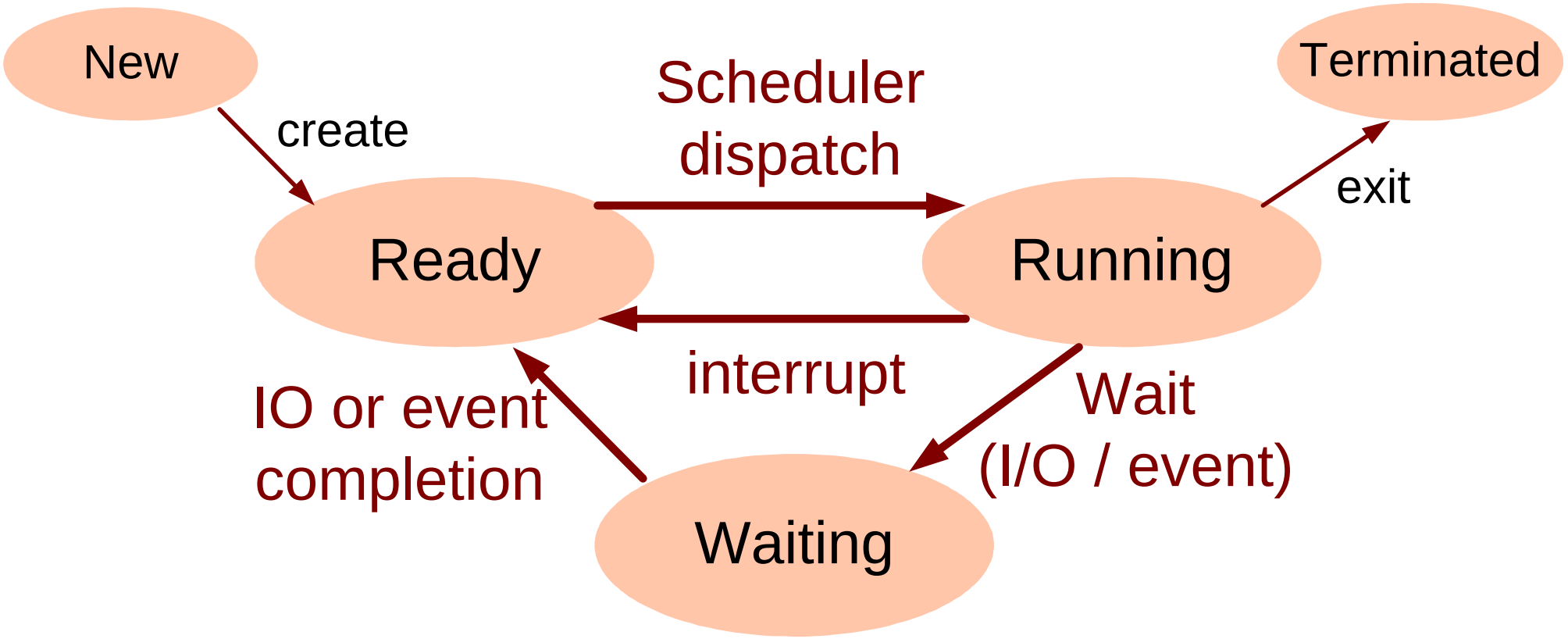


Key Hardware Performance Aspect



memory access time in case of a cache miss
200 Mhz, 5 ns cycle: 190 ns

OS Process Scheduling Concepts



Determine Expectation

simple SW model of context switch:

- save state P1
- determine next runnable task
- update scheduler administration
- load state P2
- run P2

Estimate how many

instructions and memory accesses

are needed per context switch

input data HW:

$t_{ARM\ instruction}$ = 5 ns

$t_{memory\ access}$ = 190 ns

Calculate the estimated time

needed per context switch

Determine Expectation Quantified

instructions	memory accesses	
10	1	
50	2	
20	1	
10	1	
10	1	+
100	6	

simple SW model of context switch:

- save state P1
- determine next runnable task
- update scheduler administration
- load state P2
- run P2

Estimate how many instructions and memory accesses are needed per context switch

500 ns	
1140 ns	+
1640 ns	

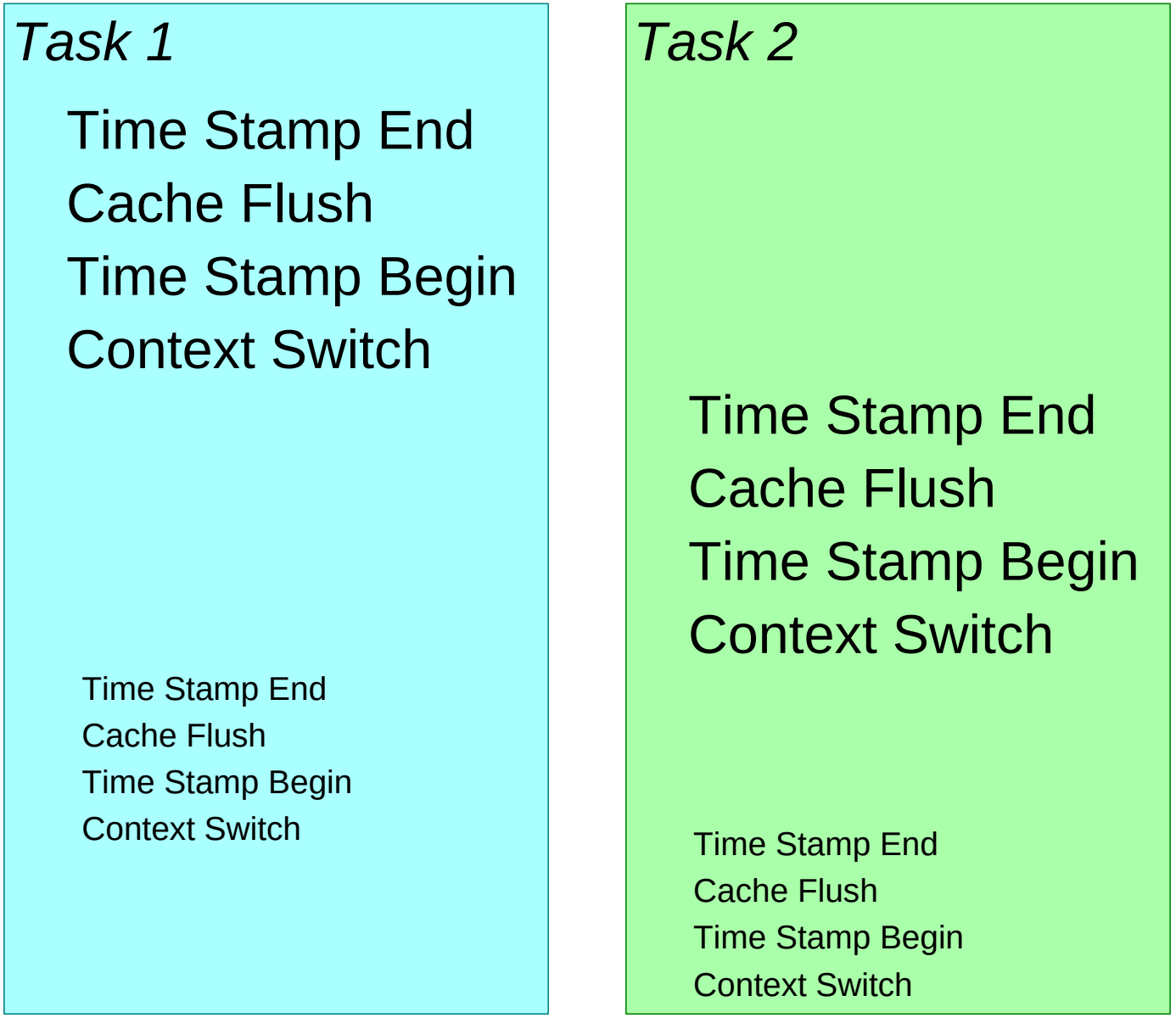
input data HW:

- $t_{ARM\ instruction}$ = 5 ns
- $t_{memory\ access}$ = 190 ns

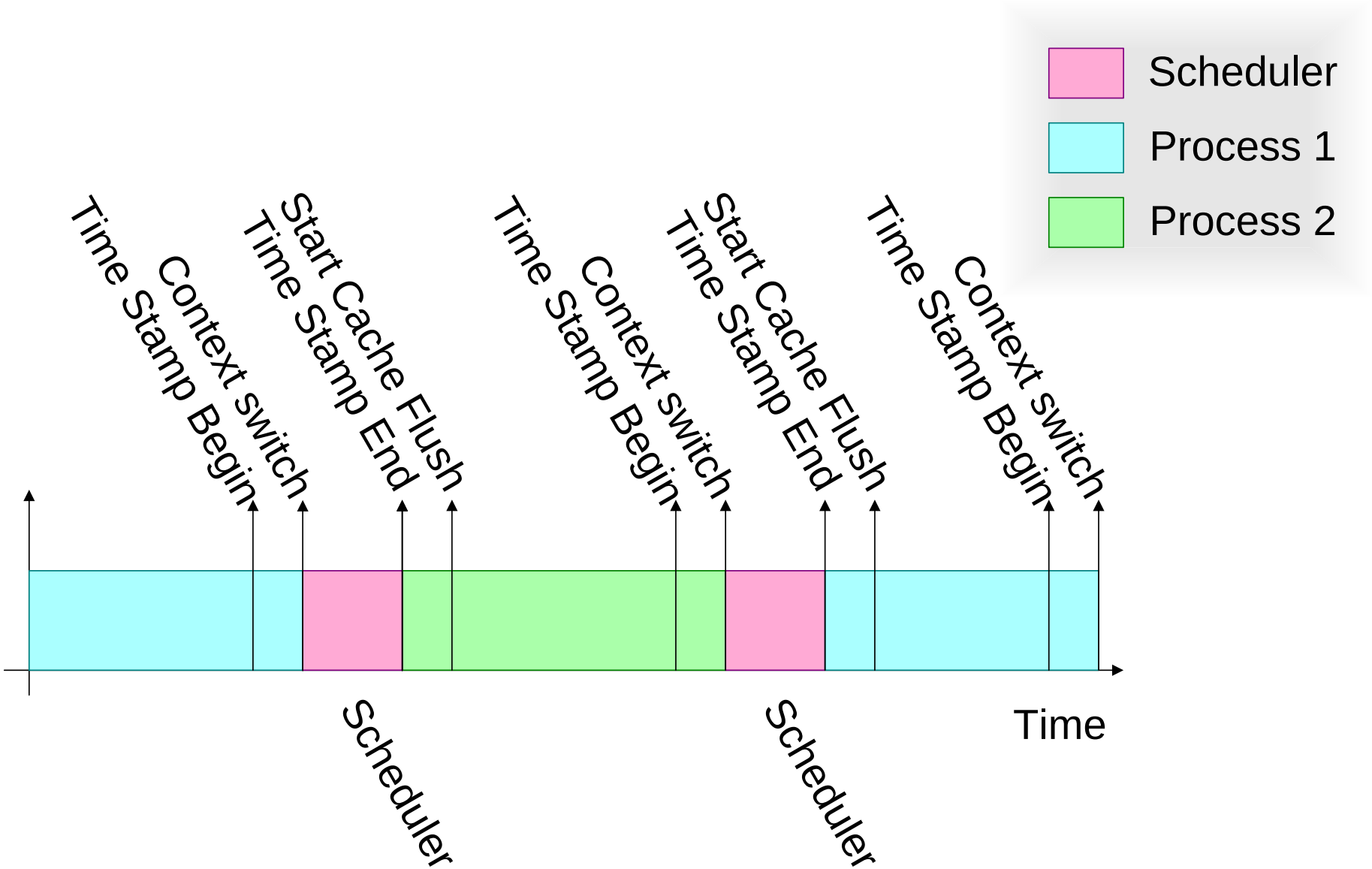
Calculate the estimated time needed per context switch

round up (as margin) gives expected $t_{context\ switch}$ = 2 µs

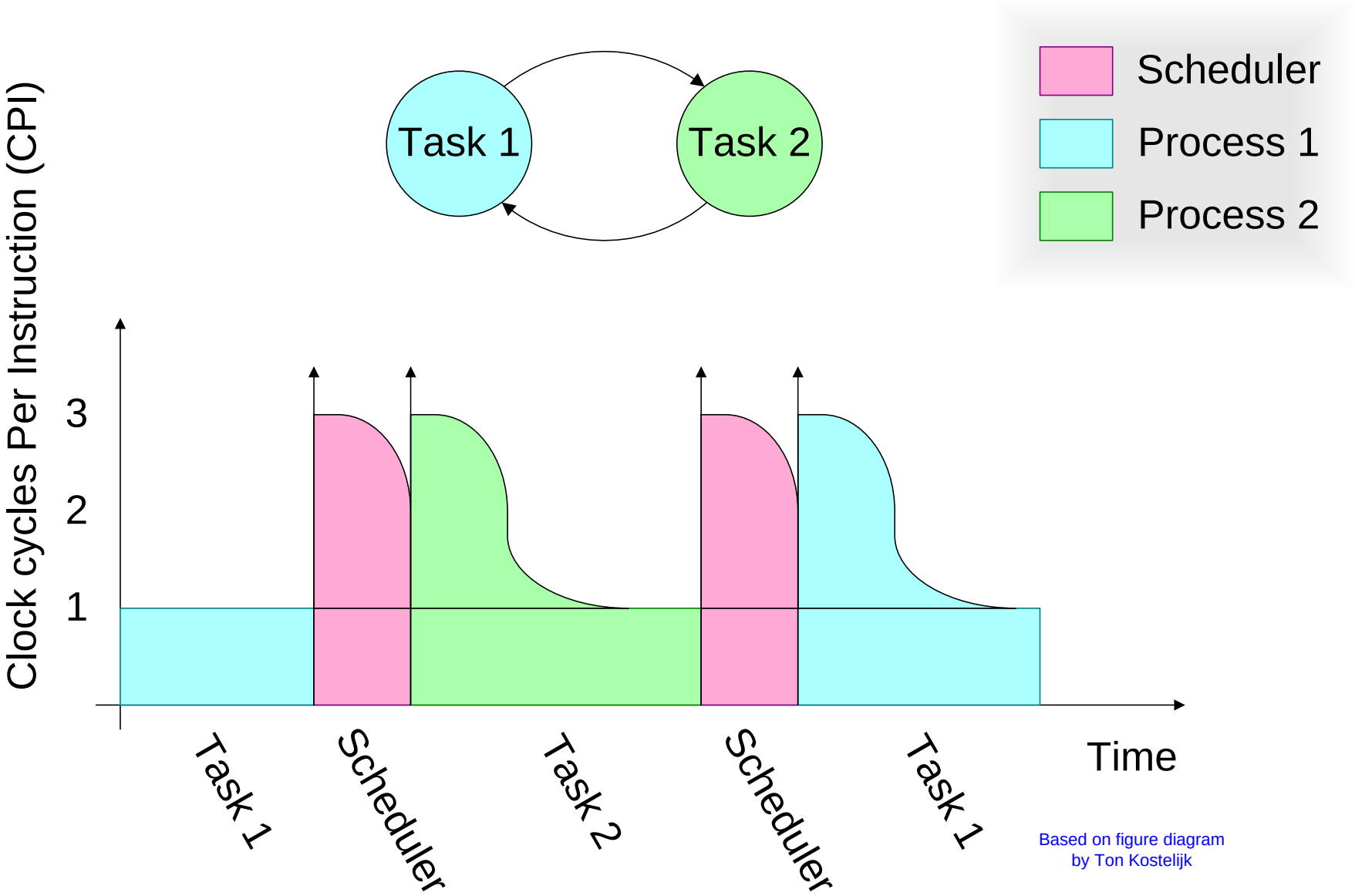
4C. Code to Measure Context Switch



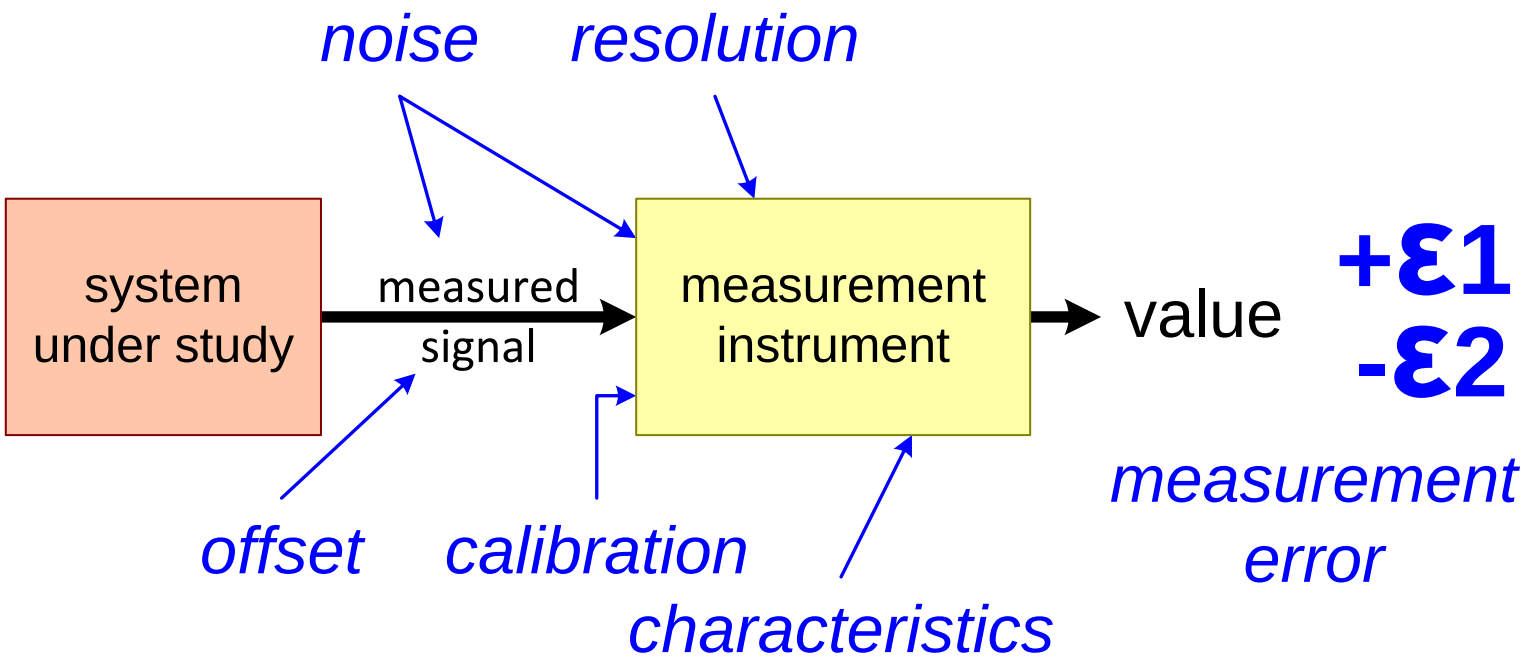
Measuring Task Switch Time



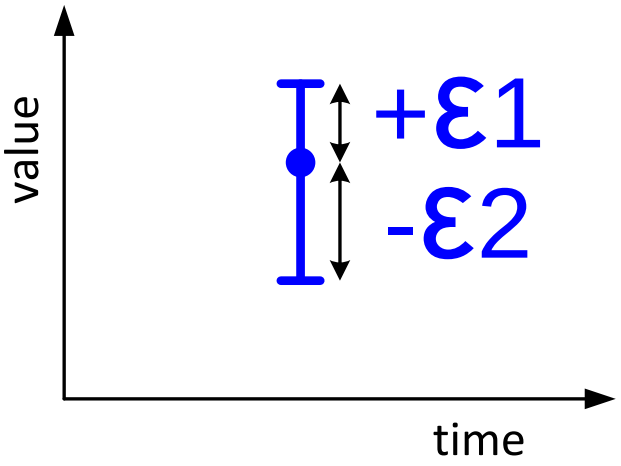
Understanding: Impact of Context Switch



5. Accuracy: Measurement Error



measurements have stochastic variations and systematic deviations resulting in a range rather than a single value



Accuracy 2: Be Aware of Error Propagation

$t_{duration} = t_{end} - t_{start}$

$t_{start} = 10 +/- 2\ \mu s$

$t_{end} = 14 +/- 2\ \mu s$

$t_{duration} = 4 +/- ?\ \mu s$

systematic errors: add linear

stochastic errors: add quadratic

Intermezzo Modeling Accuracy

Measurements have

stochastic variations and systematic deviations

resulting in a range rather than a single value.

The inputs of modeling,

"facts", assumptions, and measurement results,

also have stochastic variations and systematic deviations.

Stochastic variations and systematic deviations

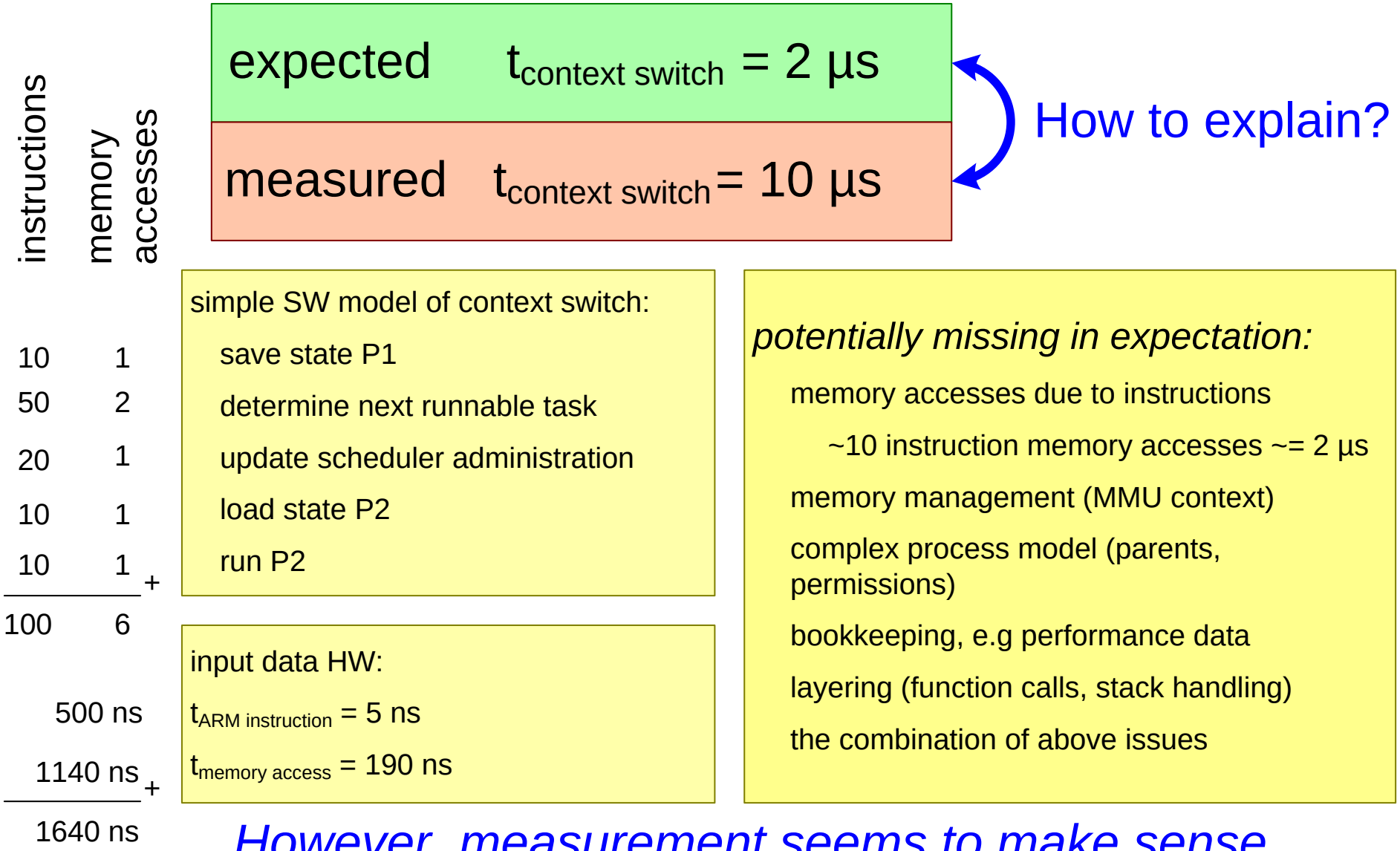
propagate (add, amplify or cancel) through the model

resulting in an output range.

ARM9 200 MHz $t_{context\ switch}$ as function of cache use

cache setting	$t_{context\ switch}$
From cache	2 µs
After cache flush	10 µs
Cache disabled	50 µs

7. Expectation versus Measurement



However, measurement seems to make sense

Conclusion Context Switch Overhead

$$t_{overhead} = n_{context\ switch} * t_{context\ switch}$$

$n_{context\ switch}$ (s^{-1})	$t_{context\ switch}$ = 10µs		$t_{context\ switch}$ = 2µs	
	$t_{overhead}$	CPU load overhead	$t_{overhead}$	CPU load overhead
500	5ms	0.5%	1ms	0.1%
5000	50ms	5%	10ms	1%
50000	500ms	50%	100ms	10%

Summary Context Switching on ARM9

goal of measurement

Guidance of concurrency design and task granularity

Estimation of context switching overhead

Cost of context switch on given platform

examples of measurement

Needed: context switch overhead ~10% accurate

Measurement instrumentation: HW pin and small SW test program

Simple models of HW and SW layers

Measurement results for context switching on ARM9

Summary Measuring Approach

Conclusions

Measurements are an important source of factual data.

A measurement requires a well-designed experiment.

Measurement error, validation of the result determine the credibility.

Lots of consolidated data must be reduced to essential understanding.

Techniques, Models, Heuristics of this module

experimentation

error analysis

estimating expectations

Colophon

This work is derived from the EXARCH course at CTT developed by *Ton Kostelijk* (Philips) and *Gerrit Muller*.

The Boderc project contributed to the measurement approach. Especially the work of

Peter van den Bosch (Océ),

Oana Florescu (TU/e),

and *Marcel Verhoef* (Chess)

has been valuable.

ASP Python Exercise

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

A simple measurement exercise is described. Purpose of this exercise is to build up experience in measuring and its many pitfalls. The programming language Python is used as platform, because of its availability and low threshold for use.

March 6, 2021
status: preliminary draft
version: 0

logo TBD

Case for Exercise

Select a programming environment,

where loop overhead and file open

can be measured in 30 minutes.

If this environment is not available,

then use Python.

Python download and information

Active State Python (Freeware distribution, runs directly)

http://www.activestate.com/Products/ActivePython/

Python Language Website

http://www.python.org/

Python Reference Card

http://admin.oreillynet.com/python/excerpt/PythonPocketRef/examples/python.pdf

Python example

```
import time

for n in (1,10,100,1000,10000,100000,1000000):
    a = 0
    tstart = time.time()
    for i in xrange(n):
        a = a+1
    tend=time.time()

    print n, tend-tstart, (tend-tstart)/n

def example_filehandling():
    f = open("c:\\temp\\test.txt")
    for line in f.readlines():
        print line
    f.close()

tstart = time.time()
example_filehandling()
tend=time.time()
print "file open, read & print, close: ",tend-tstart,"s"
```

```
>>>
1 0.0 0.0
10 0.0 0.0
100 0.0 0.0
1000 0.0 0.0
10000 0.00999999046326 9.99999046326e-007
100000 0.039999961853 3.9999961853e-007
1000000 0.44000005722 4.4000005722e-007
test line 1

line 2

line 3

file open, read, close:  0.039999961853 s
```

Exercise

- Perform the following measurements
  1. loop overhead
  2. file open
- Determine for every measurement:
  - What is the expected result?
  - What is the measurement error?
  - What is the result?
  - What is the credibility of the result?
  - Explain the result.
  - (optional) What is the variation? Explain the variation.

Reflection on Exercise

+ measuring is easy
+ measuring provides data and understanding

~ result and expectation often don't match

- sensible measuring is more difficult