

# Medical Imaging Workstation: CR Views

by *Gerrit Muller* Embedded Systems Institute

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## Abstract

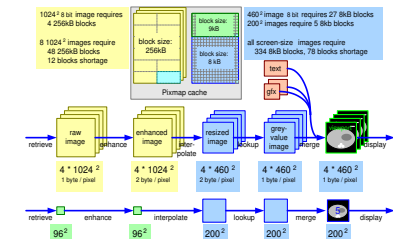
The concepts and realization of the medical imaging workstation are described. The following concepts are described: presentation and processing pipeline, resource management (CPU and memory), including caching and anti-fragmentation strategy, software process decomposition and decomposition rules.

The actual realization figures serve as illustration for the justification of some of the concepts.

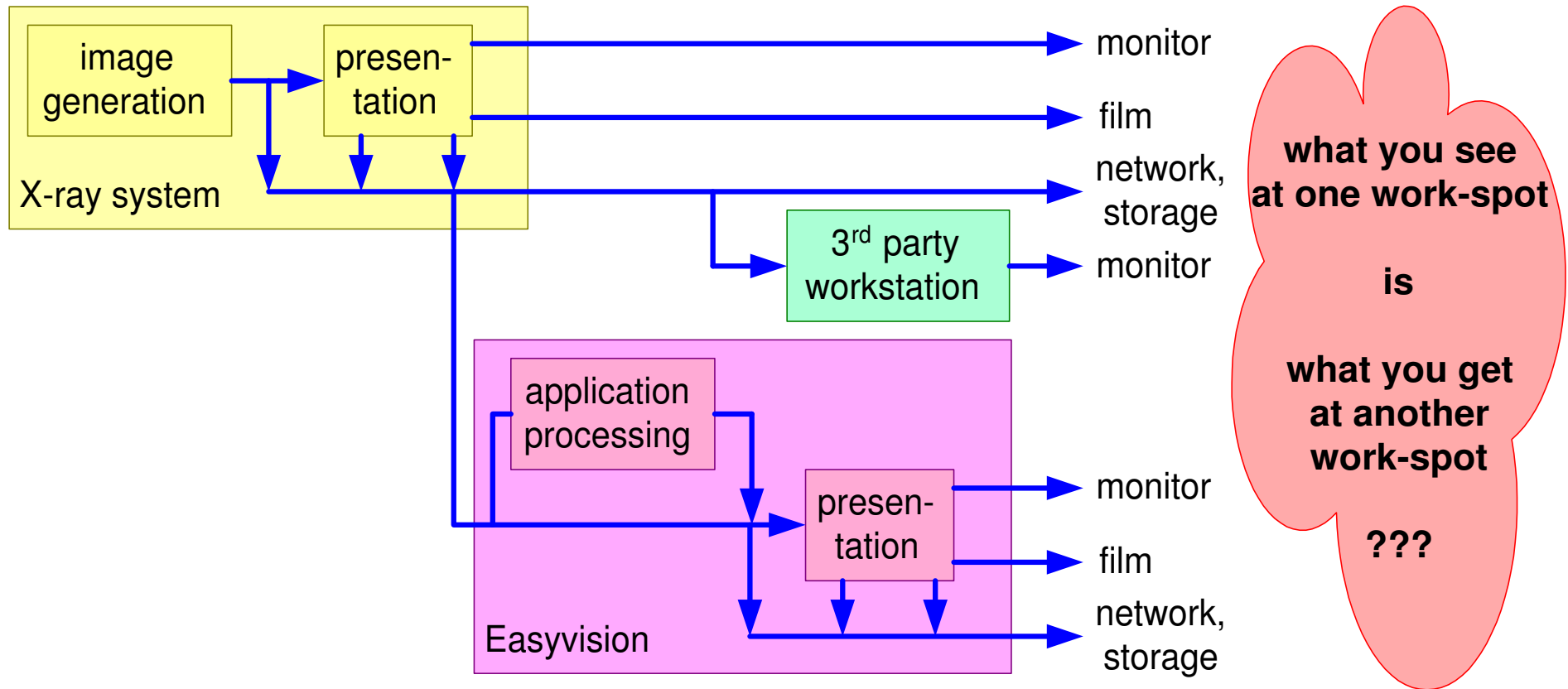
## Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

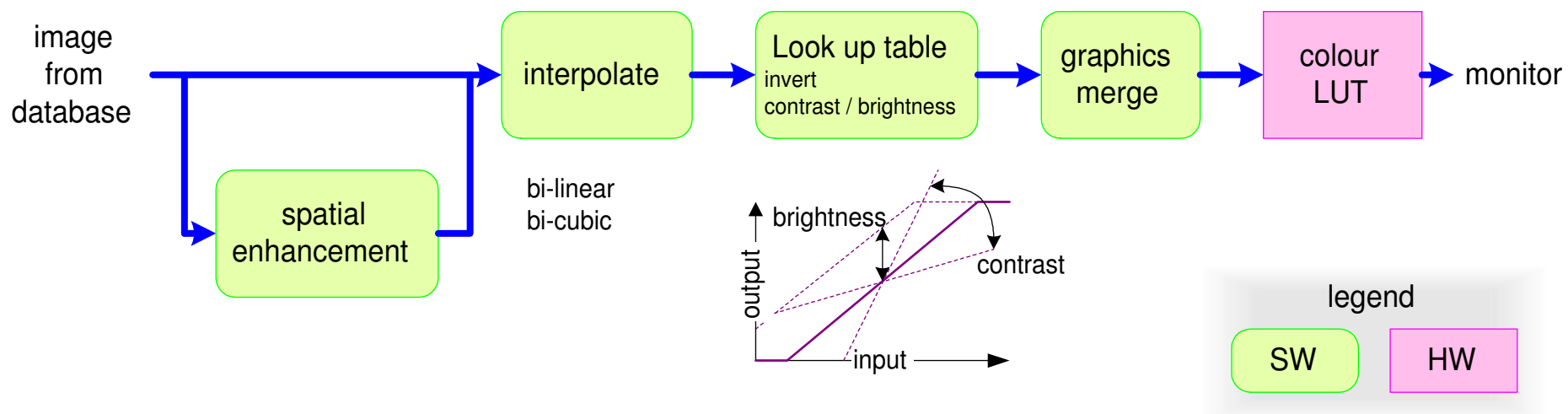
February 10, 2011  
status: finished  
version: 2.7



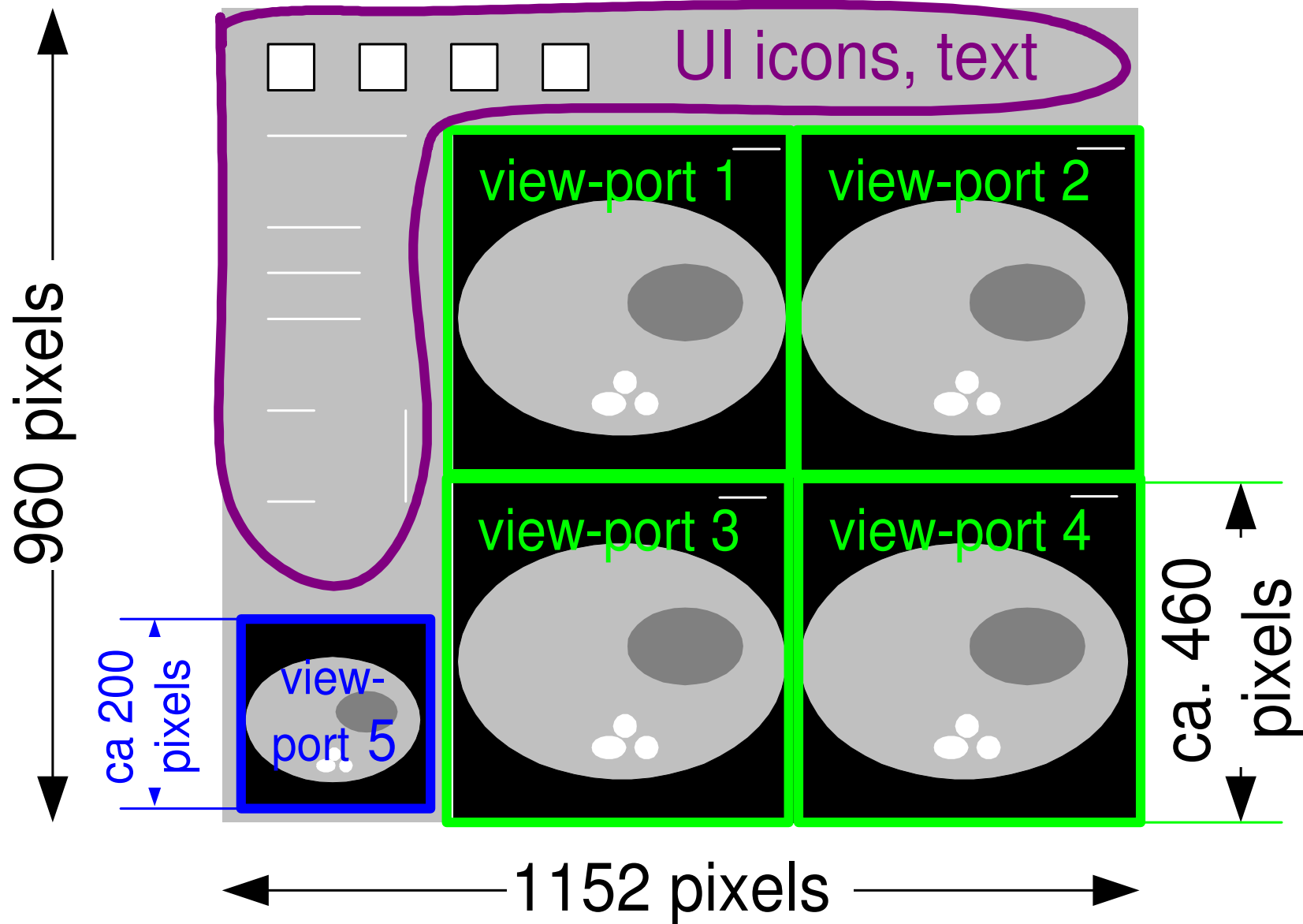
# Image Quality expectation WYSIWYG



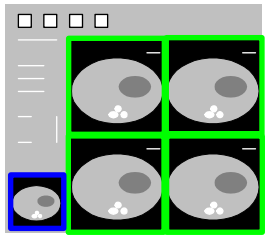
# Presentation pipeline for X-ray images



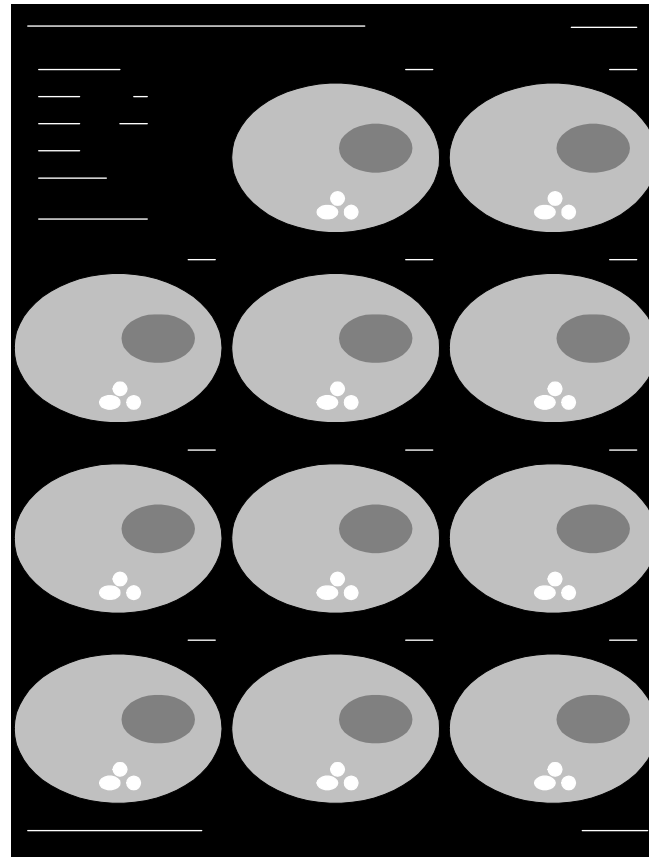
# Quadruple view-port screen layout



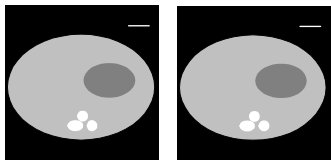
# Rendered images at different destinations



*Screen:*  
low resolution  
fast response

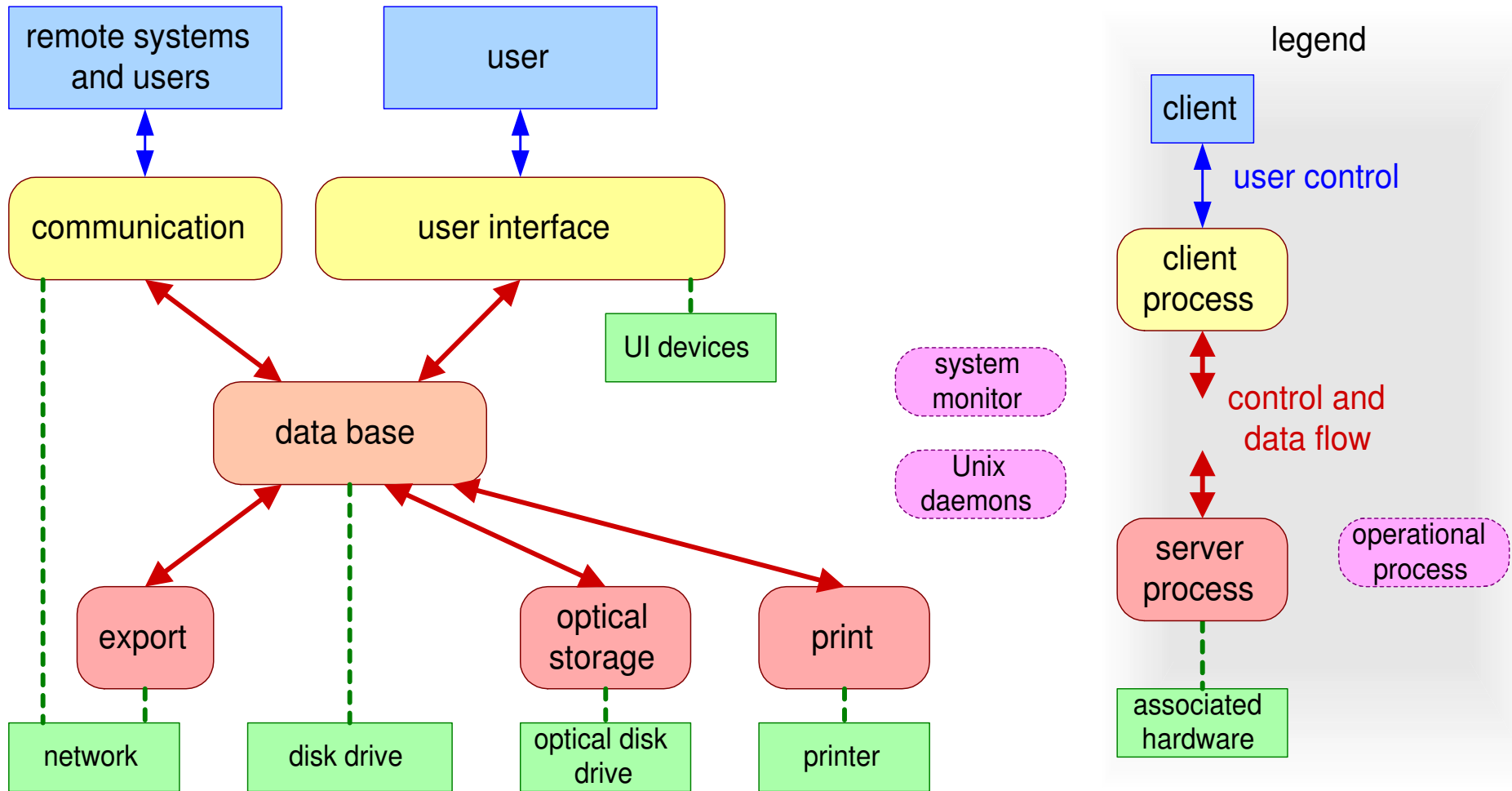


*Film:*  
high resolution  
high throughput



*Network:*  
medium resolution  
high throughput

# Concurrency via software processes

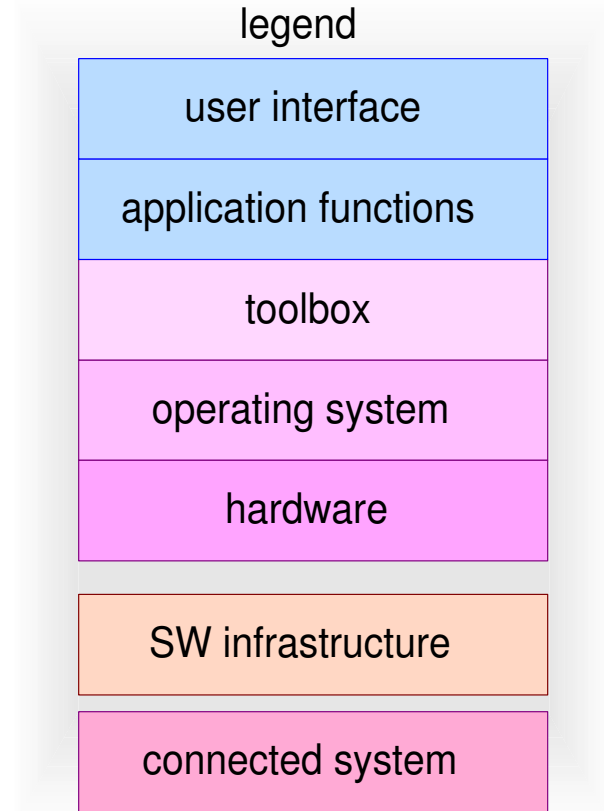
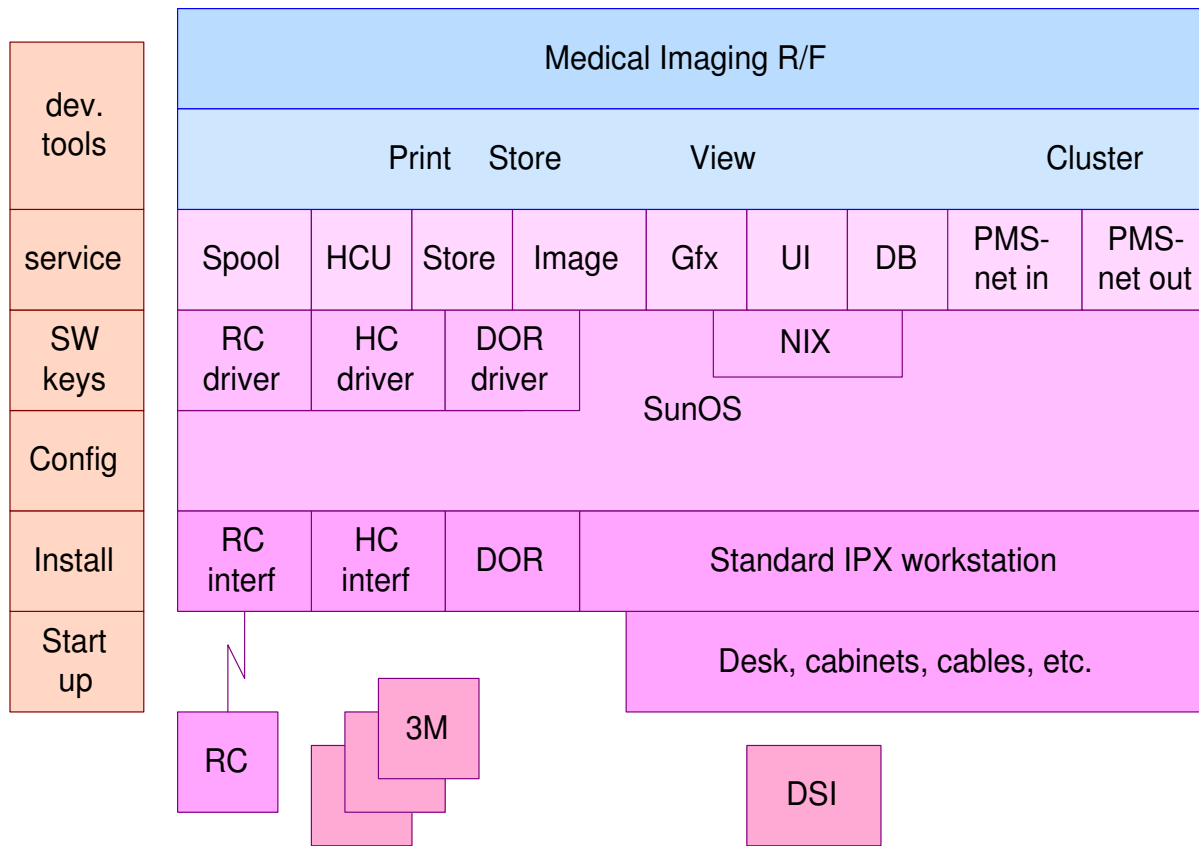


# Criteria for process decomposition

---

- management of concurrency
- management of shared devices
- unit of memory budget (easy measurement)
- enables distribution over multiple processors
- unit of exception handling: fault containment and watchdog monitor

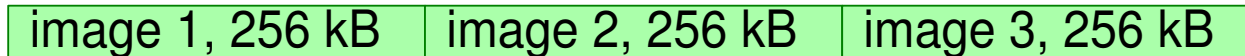
# Simplified layering of the software



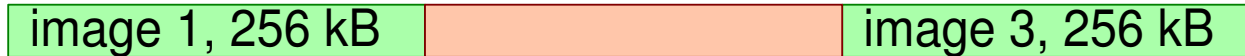
# Memory budget of Easyvision RF R1 and R2

<i>memory budget in Mbytes</i>	code		object data		bulk data		total	
	R1	R2	R1	R2	R1	R2	R1	R2
shared code	6.0	11.0					6.0	11.0
UI process	0.2	0.3	2.0	3.0	12.0	12.0	14.2	15.3
database server	0.2	0.3	4.2	3.2		3.0	4.4	6.5
print server	0.4	0.3	2.2	1.2	7.0	9.0	9.6	10.5
DOR server	0.4	0.3	4.2	2.0	2.0	1.0	6.6	3.3
communication server	1.2	0.3	15.4	2.0	10.0	4.0	26.6	6.3
UNIX commands	0.2	0.3	0.5	0.2			0.7	0.5
compute server		0.3		0.5		6.0		6.8
system monitor		0.3		0.5				0.8
application total	8.6	13.4	28.5	12.6	31.0	35.0	66.1	61.0
UNIX							7.0	10.0
file cache							3.0	3.0
total							76.1	74.0

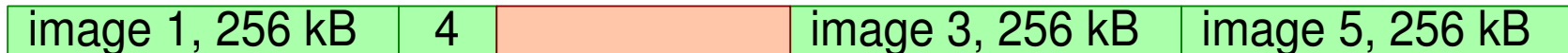
# Memory fragmentation



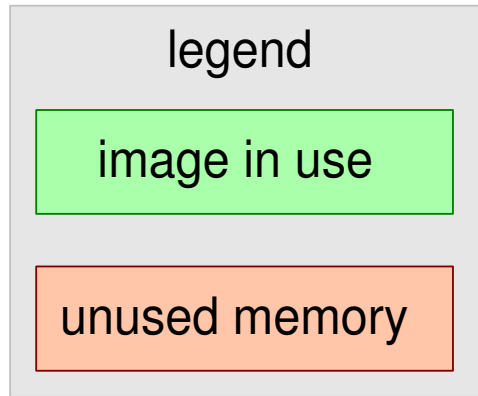
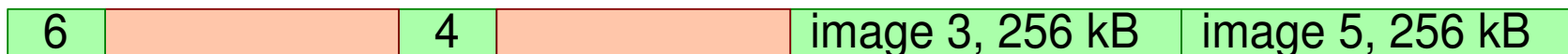
1. replace image 3 by image 4



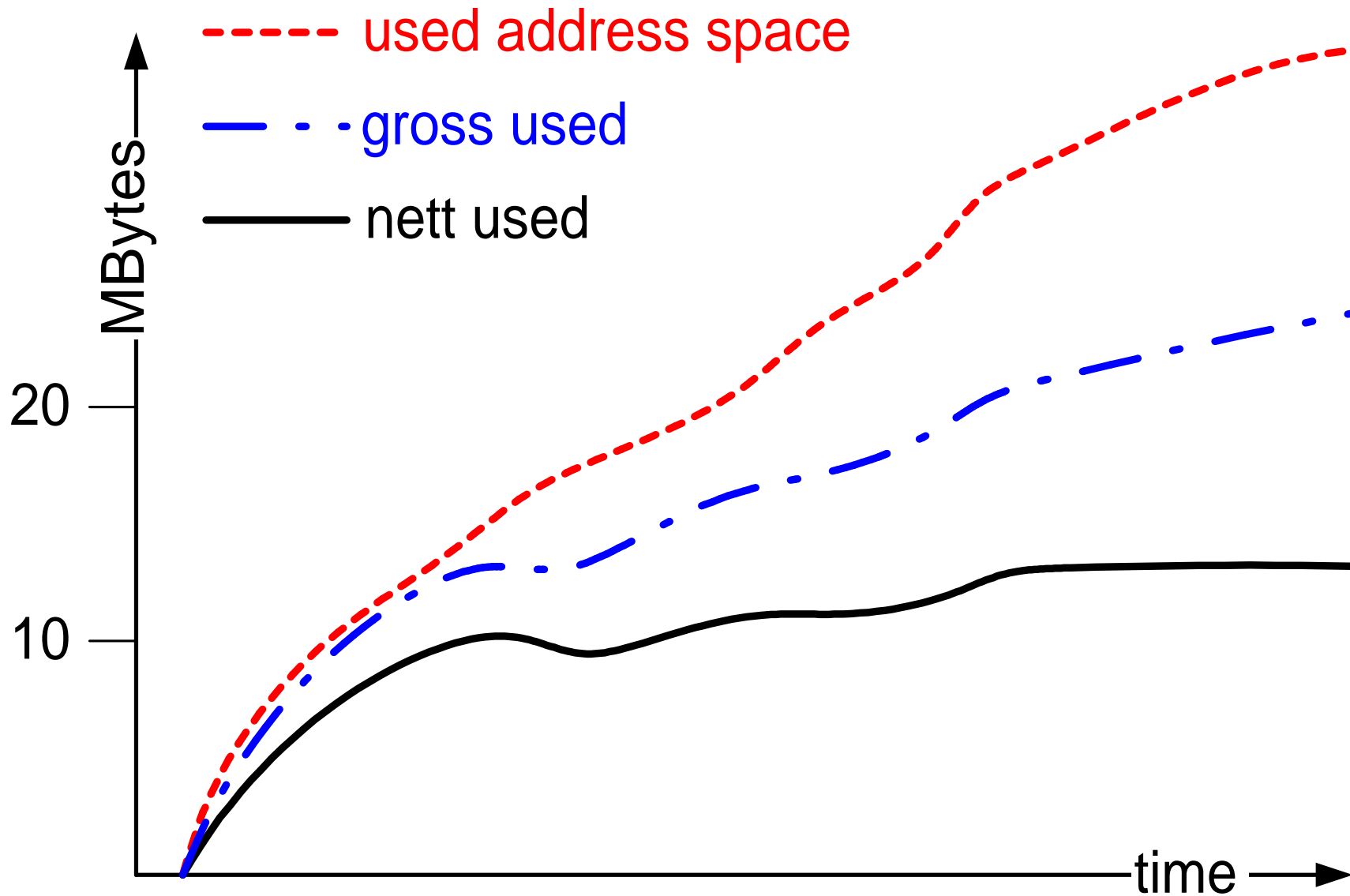
2. add image 5



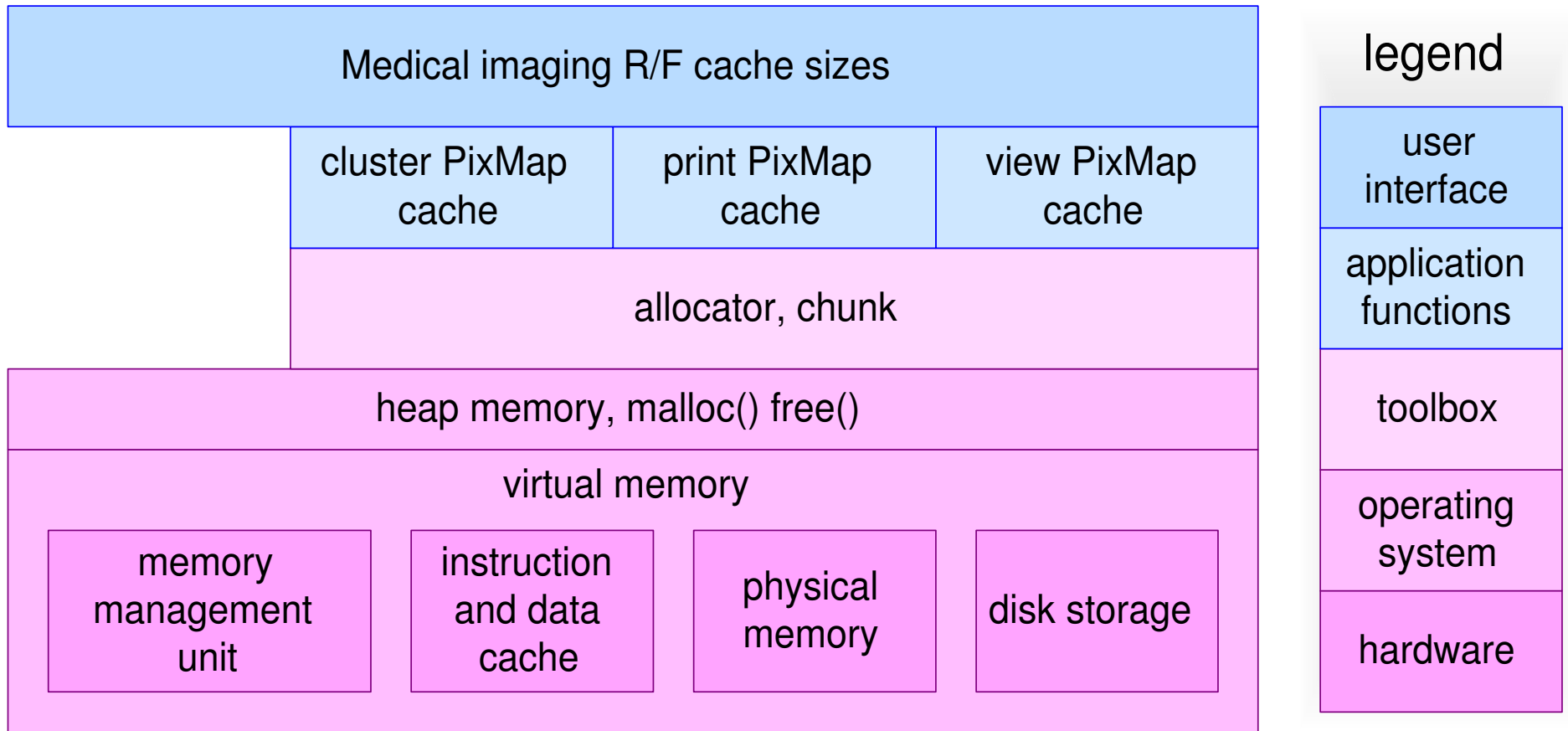
3. replace image 1 by image 6



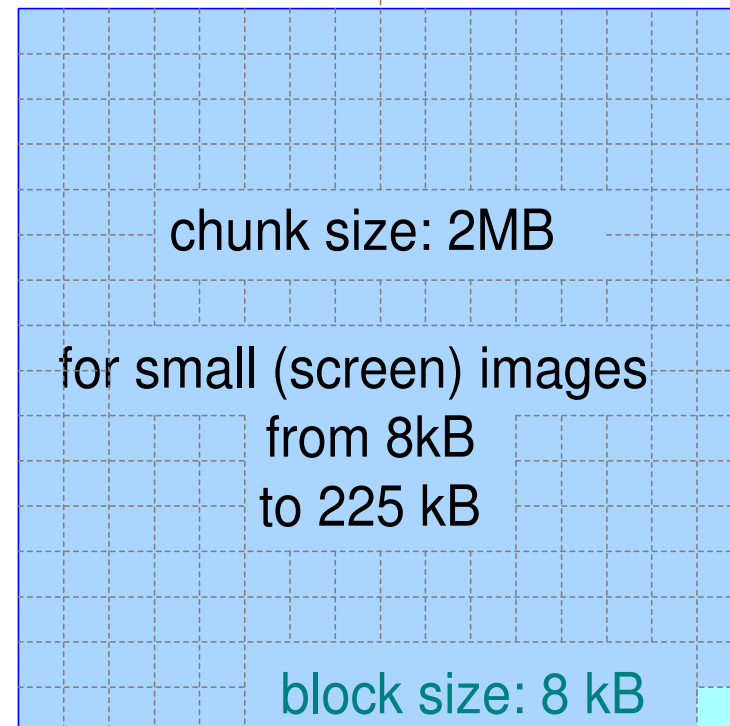
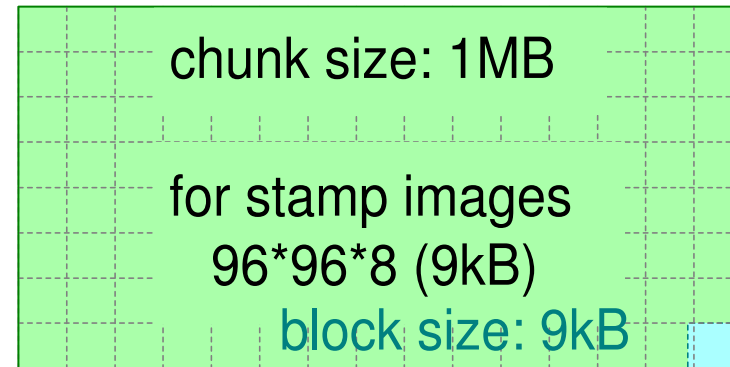
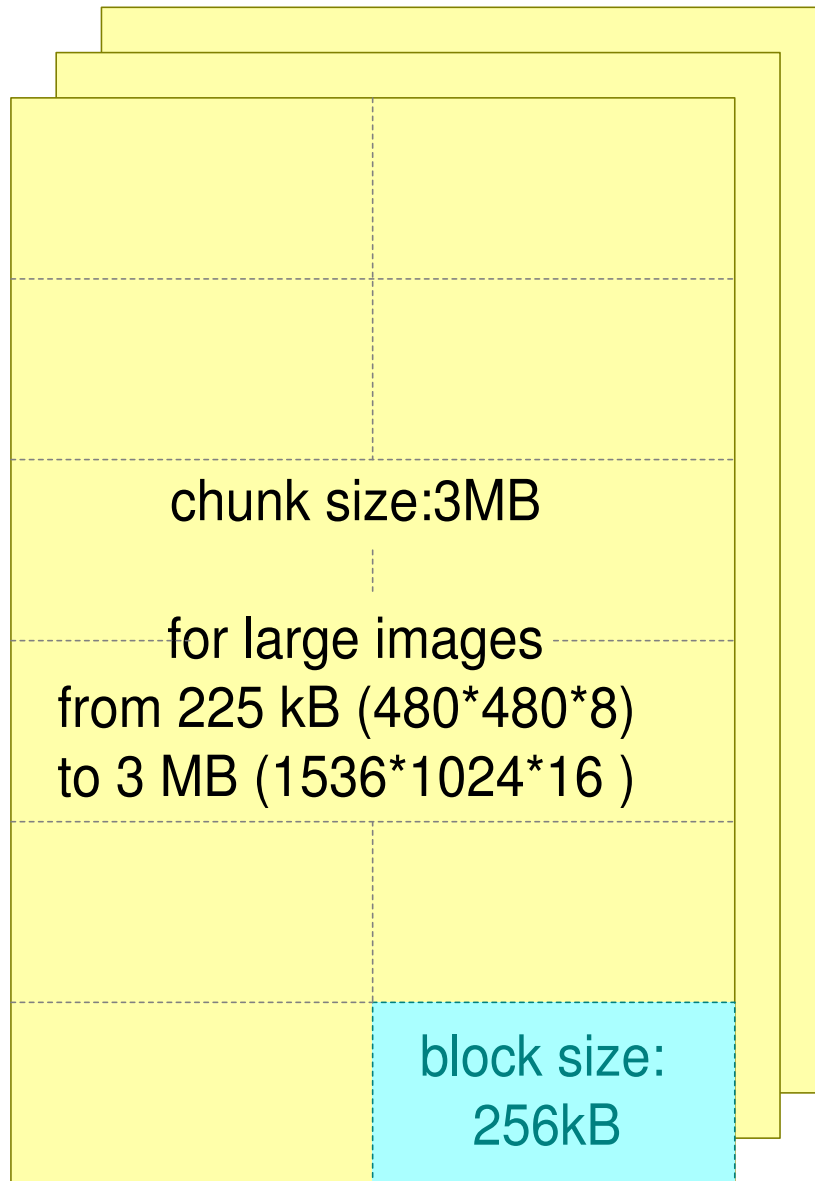
# Memory fragmentation increase



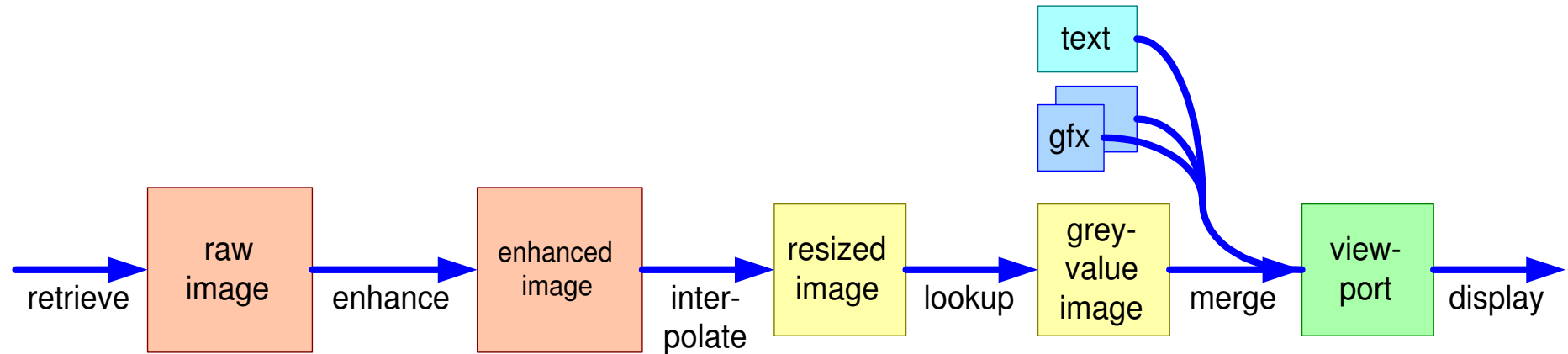
# Cache layers



# Bulk data memory management memory allocators



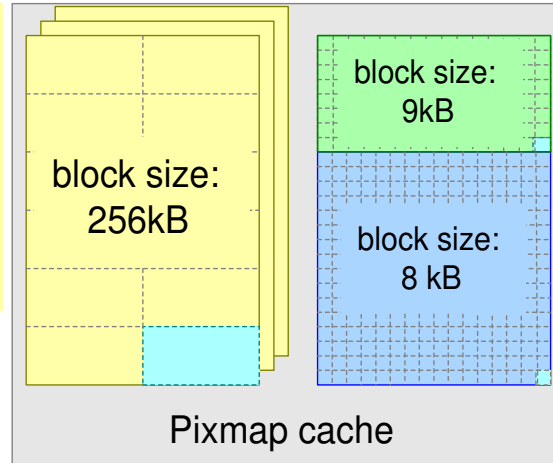
# Cached intermediate processing results



# Example of allocator and cache use

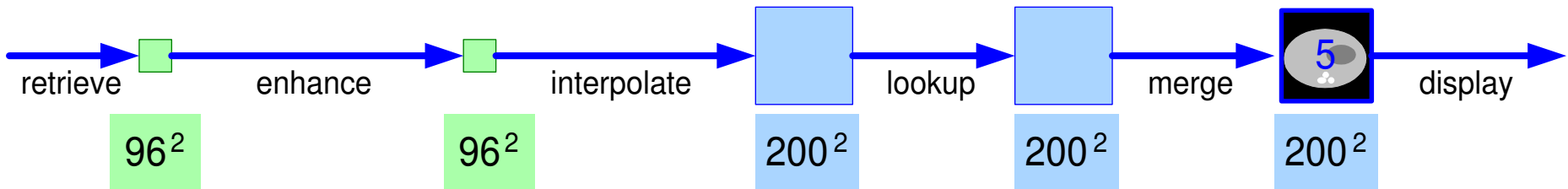
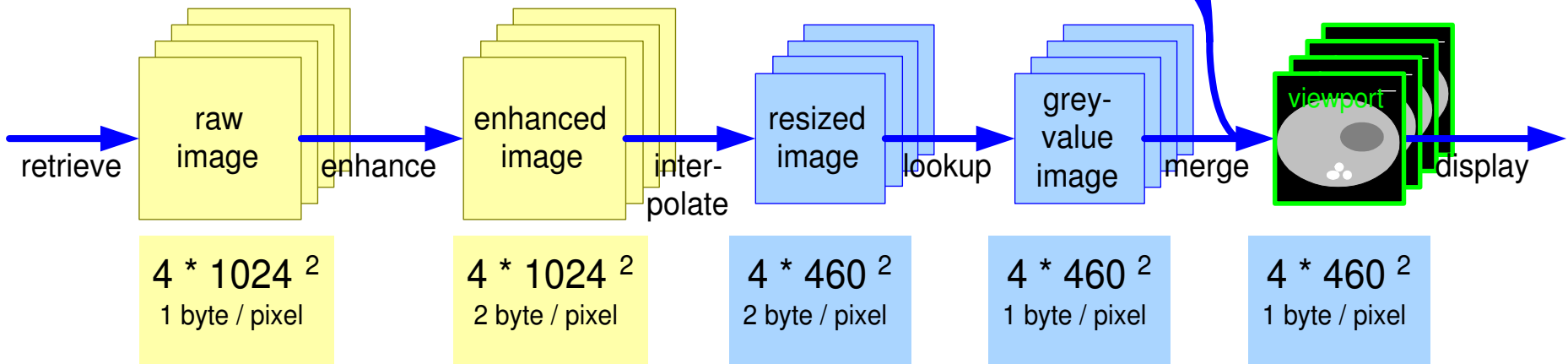
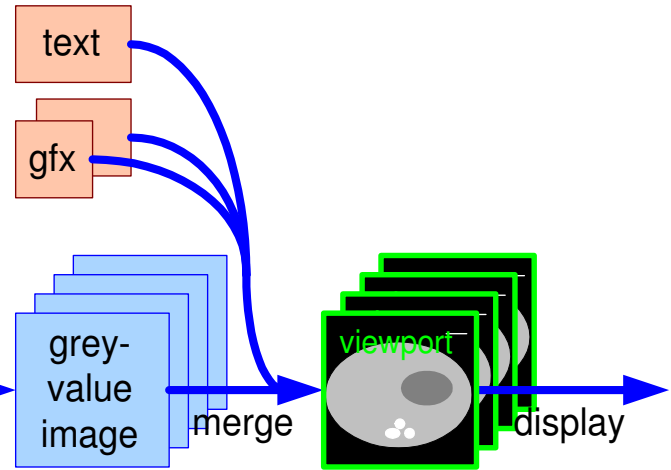
1024<sup>2</sup> 8 bit image requires  
4 256kB blocks

8 1024<sup>2</sup> images require  
48 256kB blocks  
12 blocks shortage

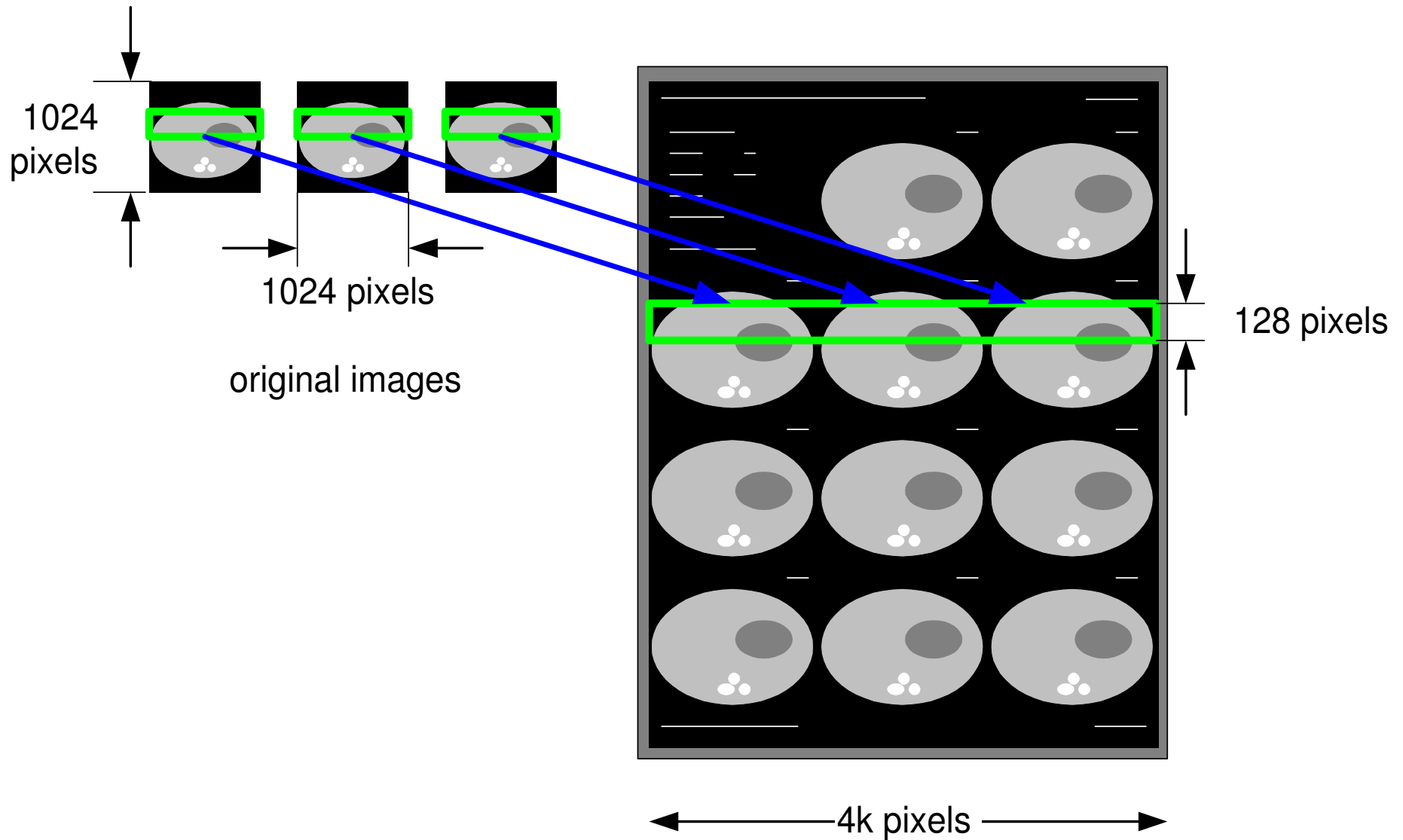


460<sup>2</sup> image 8 bit requires 27 8kB blocks  
200<sup>2</sup> images require 5 8kb blocks

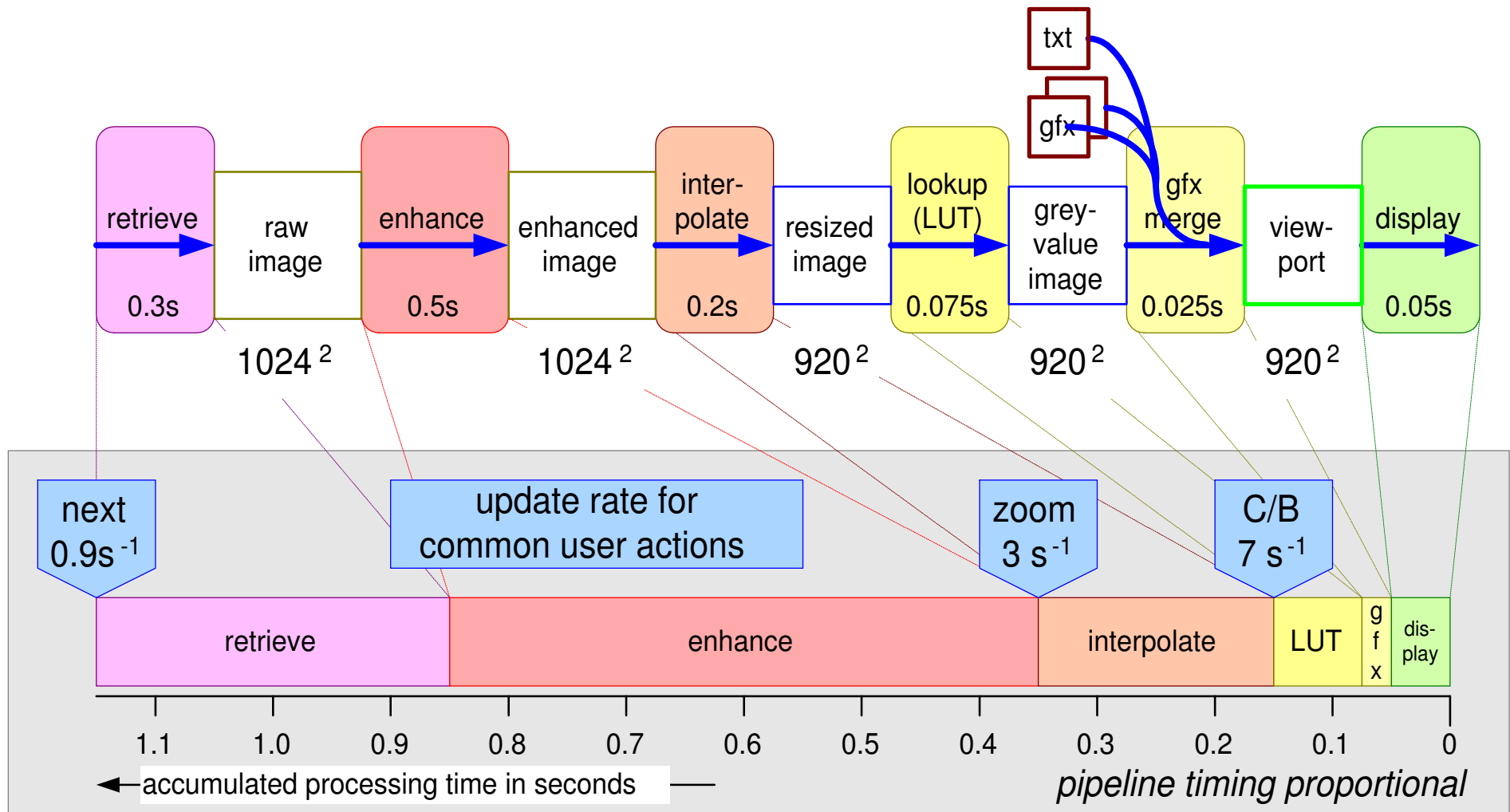
all screen-size images require  
334 8kB blocks, 78 blocks shortage



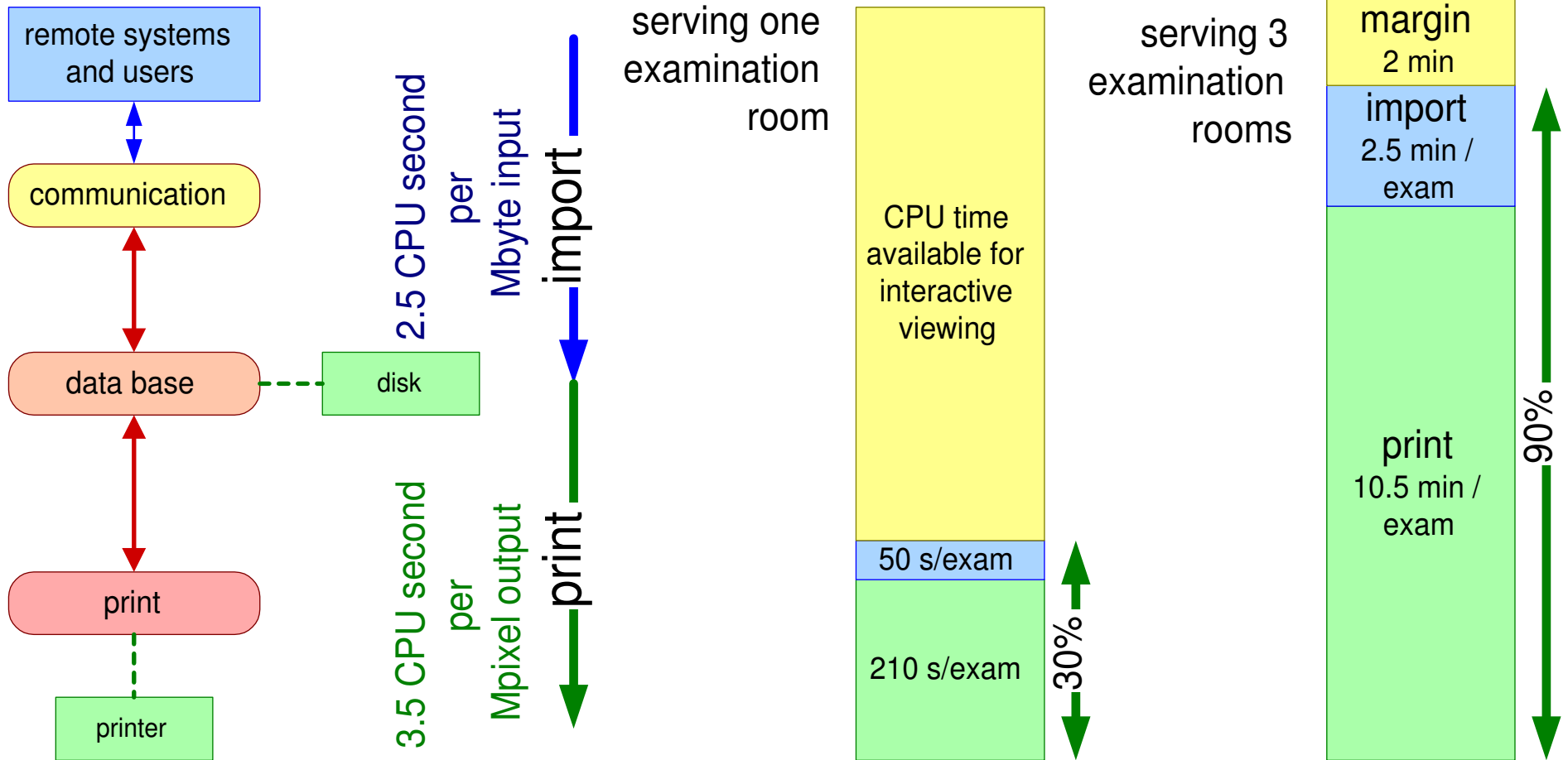
# Print server is based on banding



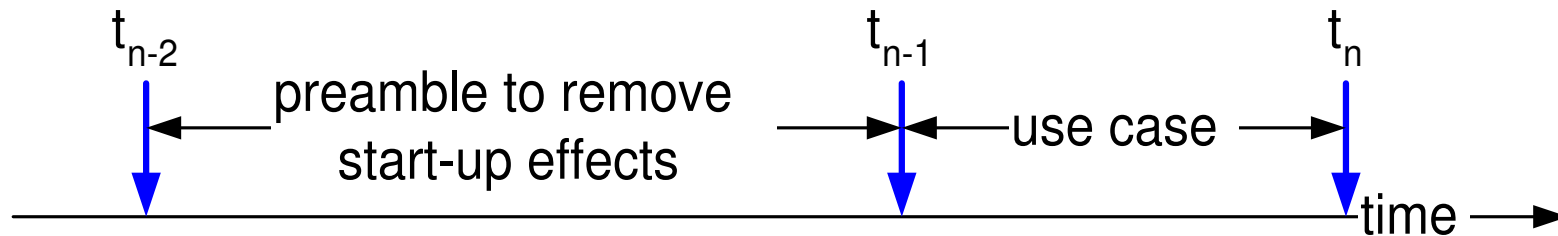
# CPU processing times and viewing responsiveness



# Server CPU load



# Resource measurement tools



oit

△ object instantiations  
heap memory usage

ps  
vmstat  
kernel resource  
stats

kernel CPU time  
user CPU time  
code memory  
virtual memory  
paging

heapviewer (visualise fragmentation)

# Object Instantiation Tracing

class name	current nr of objects	deleted since $t_{n-1}$	created since $t_{n-1}$	heap memory usage
AsynchronousIO	0	-3	+3	
AttributeEntry	237	-1	+5	
BitMap	21	-4	+8	
BoundedFloatingPoint	1034	-3	+22	
BoundedInteger	684	-1	+9	
BTreeNode1	200	-3	+3	[819200]
BulkData	25	0	1	[8388608]
ButtonGadget	34	0	2	
ButtonStack	12	0	1	
ByteArray	156	-4	+12	[13252]

# Overview of benchmarks and other measurement tools

	test / benchmark	what, why	accuracy	when
<i>public</i>	SpecInt (by suppliers)	CPU integer	coarse	new hardware
	Byte benchmark	computer platform performance OS, shell, file I/O	coarse	new hardware new OS release
<i>self made</i>	file I/O	file I/O throughput	medium	new hardware
	image processing	CPU, cache, memory as function of image, pixel size	accurate	new hardware
	Objective-C overhead	method call overhead memory overhead	accurate	initial
	socket, network	throughput CPU overhead	accurate	ad hoc
	data base	transaction overhead query behaviour	accurate	ad hoc
	load test	throughput, CPU, memory	accurate	regression

# Coverage of submethods of the CR views

## C

### Conceptual

*construction decomposition*  
*functional decomposition*  
*designing with multiple decompositions*  
*execution architecture*  
*internal interfaces*  
*performance*  
*start up*  
*shutdown*  
*integration plan*

**work breakdown**  
**safety**

reliability  
security

## R

### Realization

*budget*  
*benchmarking*  
*performance analysis*  
*granularity determination*

**value and cost**

safety analysis  
reliability analysis  
security analysis

legend    *explicitly addressed*    **addressed only implicitly**    not addressed

coverage based on documentation status of first product release

## disclaimer

The case material is based on actual data, from a complex context with large commercial interests. The material is ***simplified*** to increase the accessibility, while at the same time ***small changes*** have been made to remove commercial sensitivity. Commercial sensitivity is further reduced by using relatively ***old*** data (between 5 and 10 years in the past). Care has been taken that the illustrative value is maintained