Adaptive Runtime Resource Management of Heterogeneous Resources

Roel Wuyts
Principal Scientist, imec
Professor, KUL (Distrinet)

Carreer Overview

Studies: Licentiaat Informatica (VUB, 1991-1995)

1995	2001	2004	07	08
Doctoral Researcher VUB	Postdoc University of Bern, Switzerland	Chargé de cours ULB	Principal Scientist imec	
				Professor (10%) KUL

Juggling Hats

IMEC
- Embedded devices
- Runtime resource management

ULB
- Object versioning
- AOP

KUL
Language Design

imec

- Research organization located in Leuven
  - world-leading independent research center in nanoelectronics and nanotechnology
  - More Moore research targets semiconductor scaling for the 22nm technology node and beyond.
  - More than Moore research invents technology for nomadic embedded systems, wireless autonomous transducer solutions, biomedical electronics, photovoltaics, organic electronics and GaN power electronics.
- Numbers
  - Budget: ± 200 M€
  - Staff: ± 1700
  - Cleanroom: ± 10,000 m2

The ARES Team


Maja
D'Hondt


Rogier
Baert


Carolina
Blanch


Paul
Coene


Zhe
Ma


Roel
Wuyts

IMEC Ph.D. Students


Narasinga Rao
Miniskar

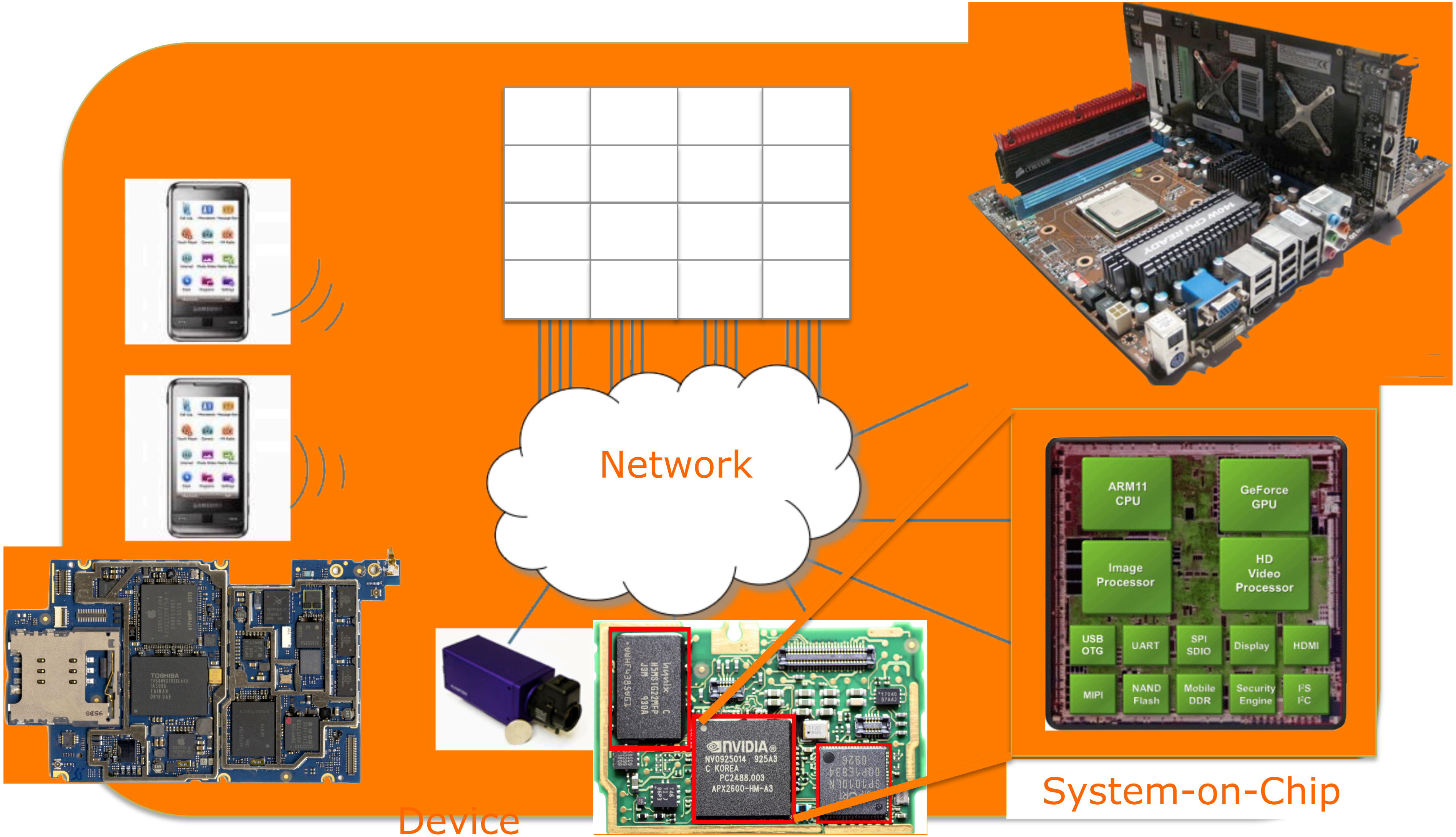

Hengjie
Song

Master Students

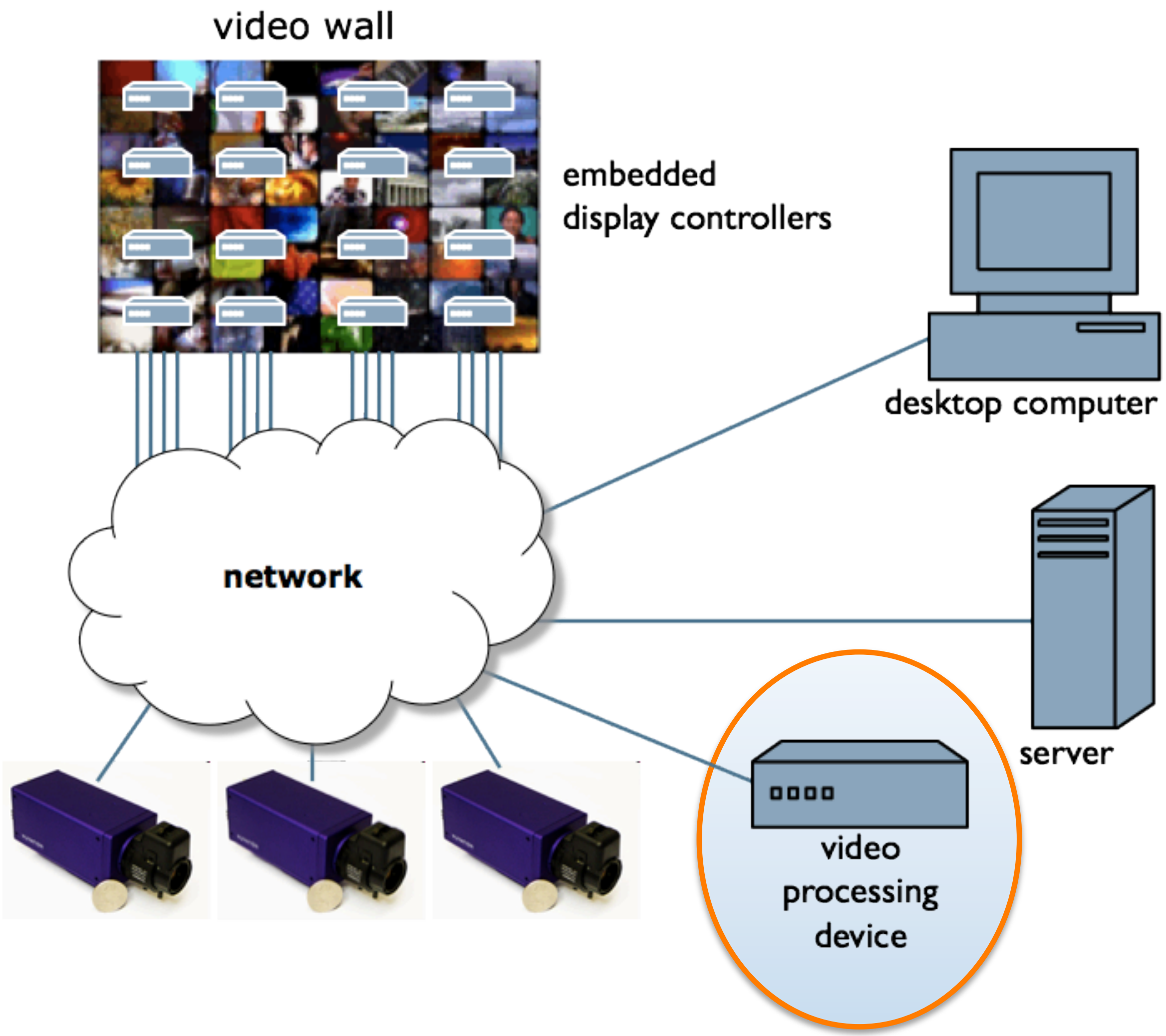

Tipnis
Ameya

ARES General Goal

- Software that takes advantage of heterogeneous platforms is becoming the rule.
- Developing such software is hard because:
  - A decision needs to be made regarding what software components can use what resources,
  - that decision varies at runtime as the application's context changes.
  - moreover the decision needs to result in good performance,
  - And the software needs to run with many possible resource configurations
- ARES solves this problem through adaptive runtime resource management, a solution that monitors applications at runtime and decides the assignment of resources to software components at runtime according to a decision algorithm.

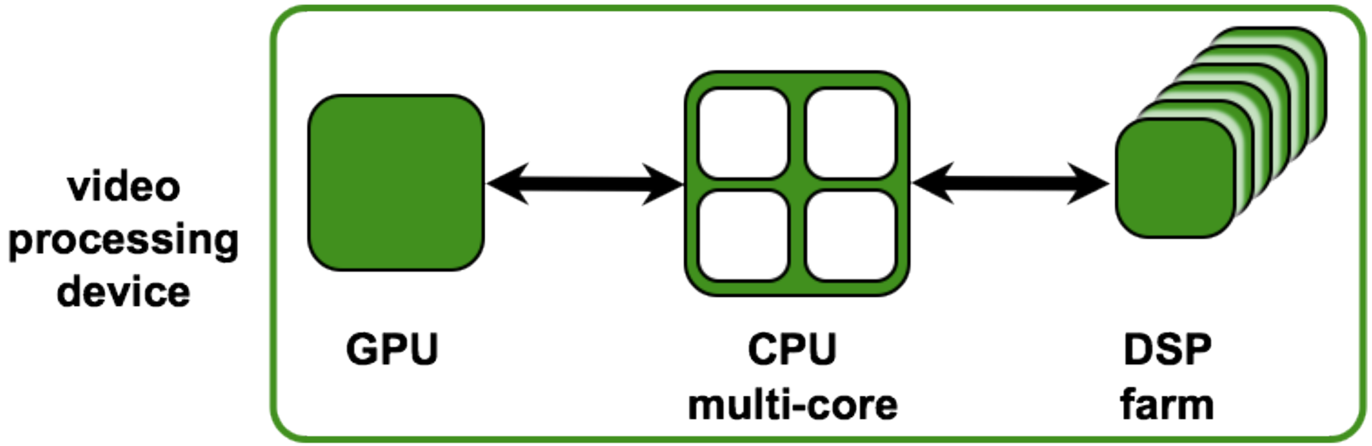

Network
ARM11 CPU
GeForce GPU
Image Processor
HD Video Processor
USB OTG
UART
SPI SDIO
Display
HDMI
MIPI
NAND Flash
Mobile DDR
Security Engine
System-on-Chip
Device

Context: Networked Video Processing



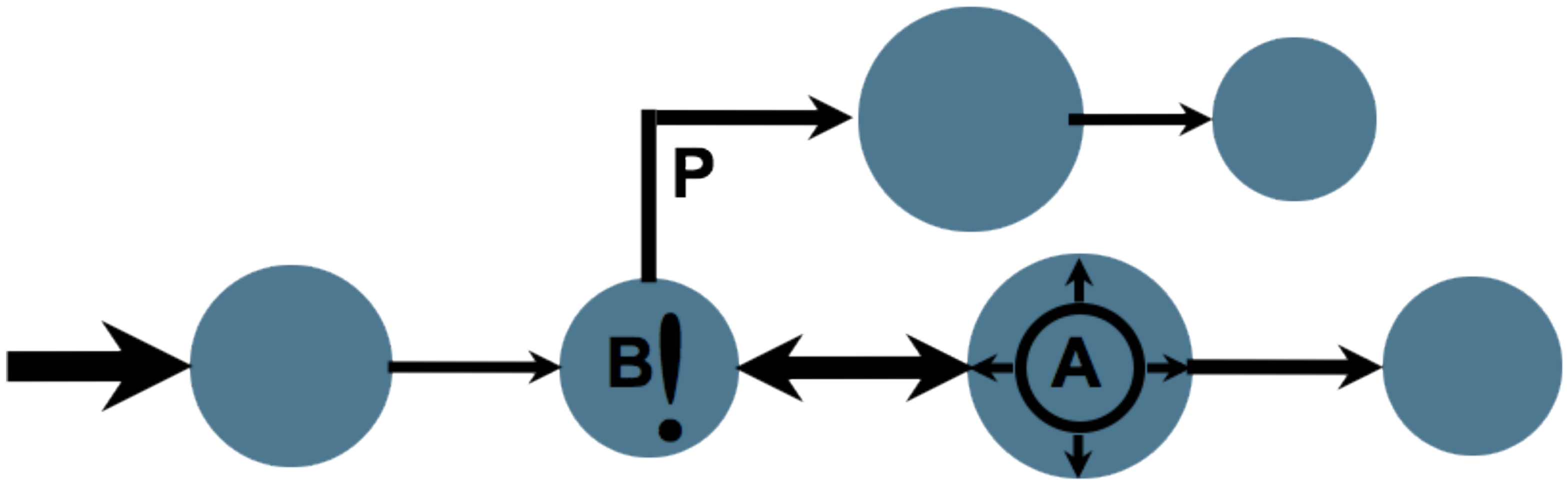
Video Processing Device

- Previously: custom hardware
- Now: Device with off-the-shelf CPU and GPU and optionally DSP-board
  - Many different kinds of CPUs and GPUs -> high variability
  - hardware evolves rapidly -> high variability



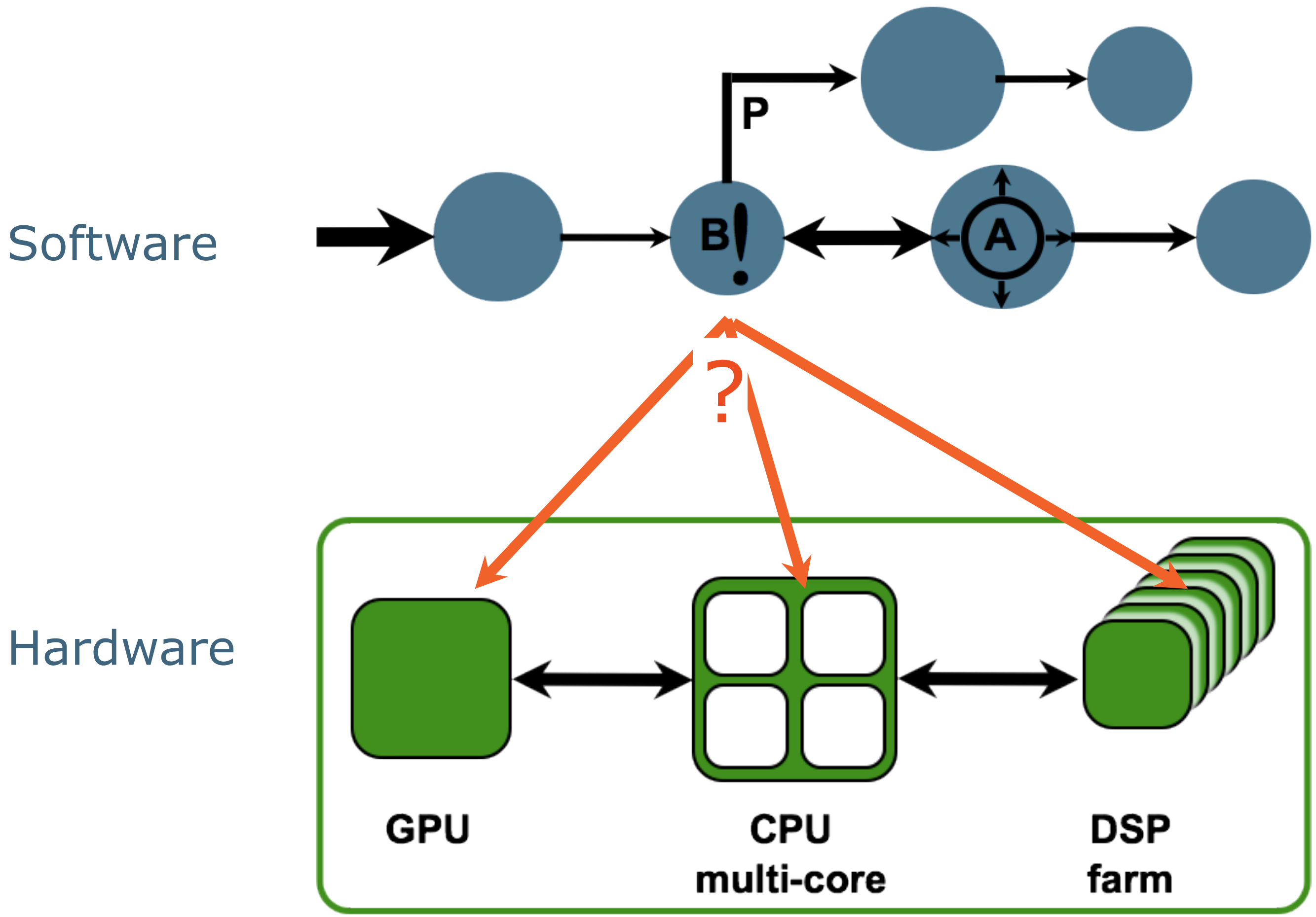
Software Pipelines in Video Processing

- Software to process and analyze video streams
  - encoders, decoders, transcoders, object (e.g. logo) detection, video scalers, color space conversion, ...
- Characteristics
  - Data-dependent: changing workloads (component A in example)
  - User/context interaction: changing pipelines (B triggers pipeline P)



Developing on Heterogeneous Platforms

- Assignment Problem: what runs where when?



Related Work

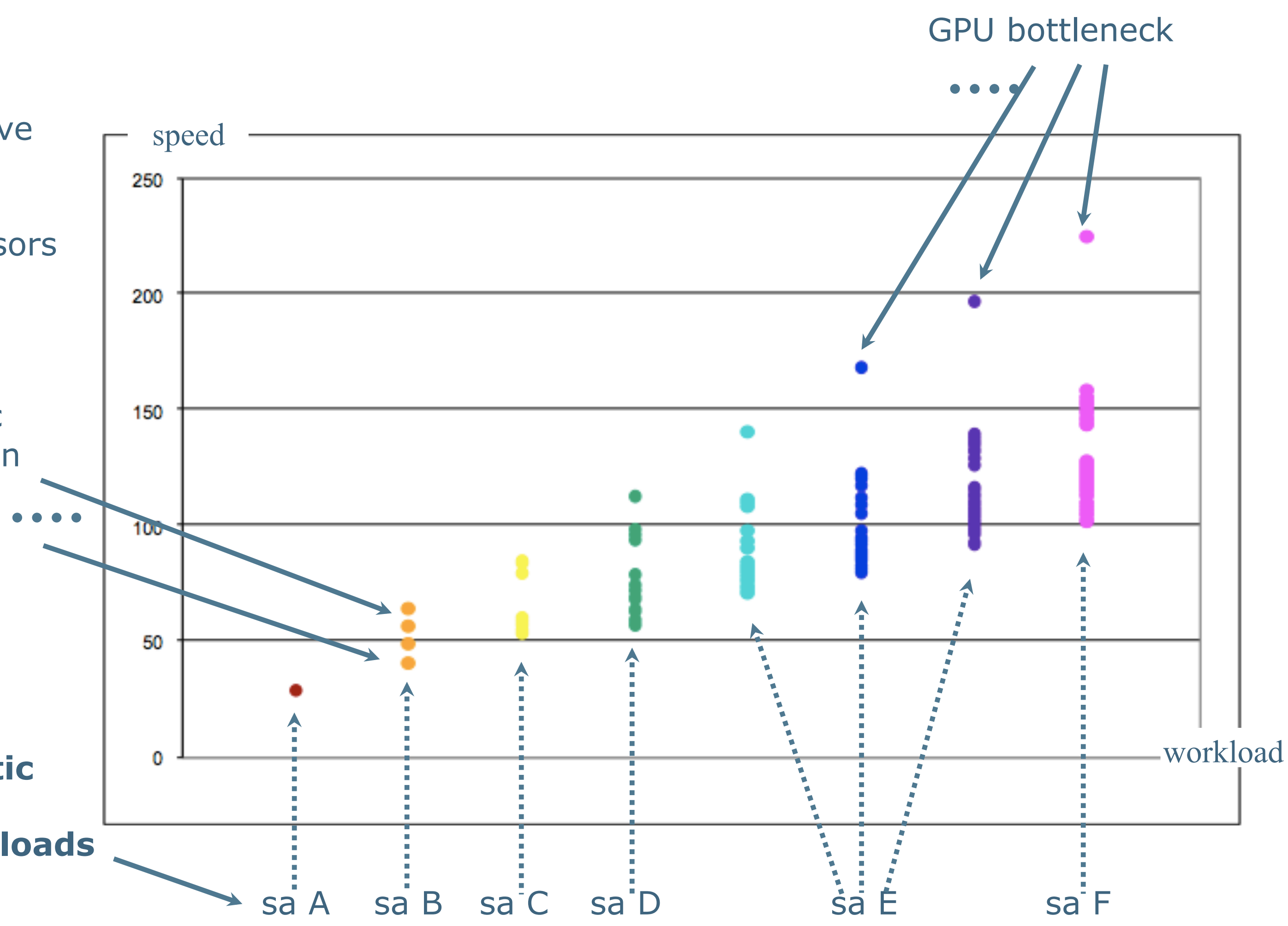
- Practice: (manual) design-space exploration + assumptions

- Task assignment for heterogeneous systems
  - V. J. Jiménez, L. Vilanova, I. Gelado, M. Gil, G. Fursin, and N. Navarro. *Predictive runtime code scheduling for heterogeneous architectures* [HiPEAC '09]
  - Finer-granularity imposing only simple assignment strategies
- Task scheduling on heterogeneous multicore architectures
  - C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier. *StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures* [Euro-par'09].
  - Only list scheduling and without taking data transfer times into account
- Static scheduling heuristics for heterogeneous processors
  - H. Oh and S. Ha. *A static scheduling heuristic for heterogeneous processors* [Euro-Par '96], H. Topcuoglu, S. Hariri, and M.-Y. Wu. *Task scheduling algorithms for heterogeneous processors*. Heterogeneous Computing Workshop, 1999.
  - Formal approaches without implementation, no runtime assignment

Static Assignment Problem 1: which is best?

different workloads have different best static assignments on heterogeneous processors

different static assignments on GPU and CPU per workload ....



6 different best static assignments (sa) for 8 different workloads

Static Assignment Problem 2: scaling

- experiment (previous graph):
  - 1 to 8 streams
  - 2 resolutions
  - 8 different load distributions over GPU and CPU
  - # static assignments ~ 100 (points in the graph)

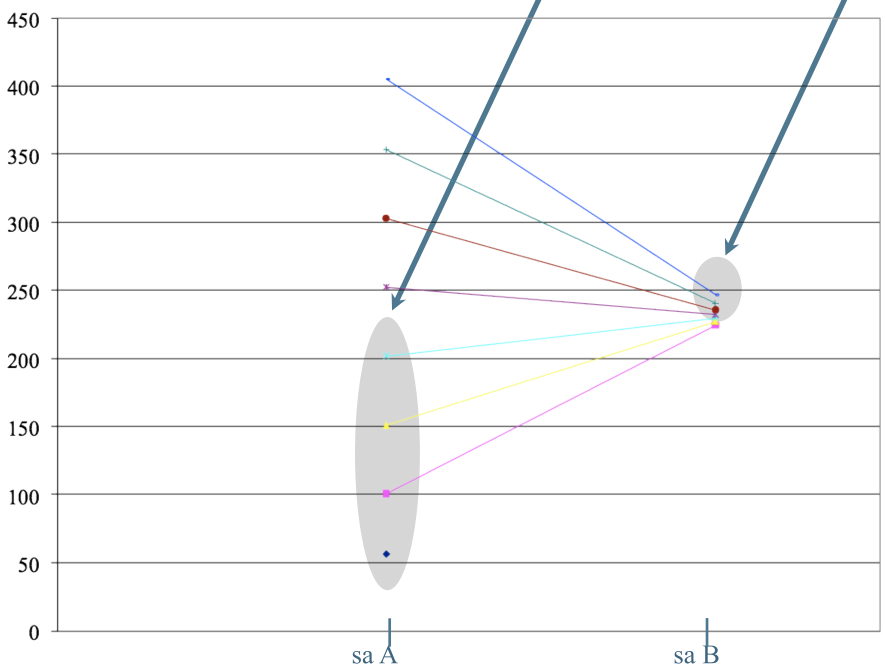
- professional video processing
  - 1 to 64 streams
  - 4 resolutions
  - 64 different load distributions over GPU and CPU
  - # static assignments ~108

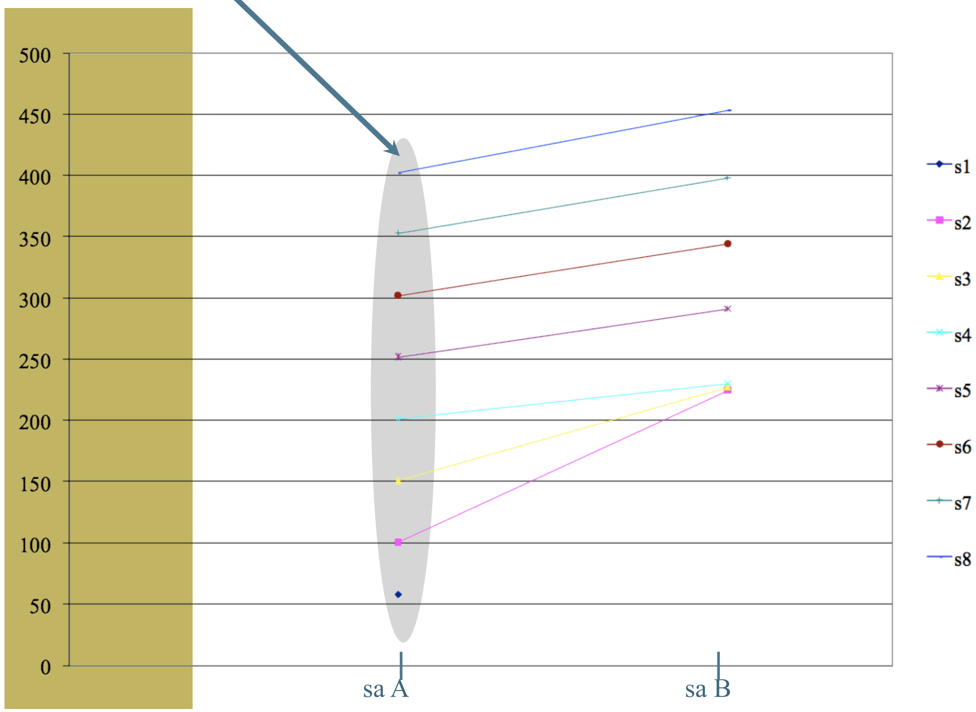
static assignment problem 3: heterogeneity

variations in configurations of processors have different best static assignments for same workloads

best static assigments on **1 GPU and 2x 4-core CPU**:
> 4 tasks: sa A
≤ 4 tasks: sa B

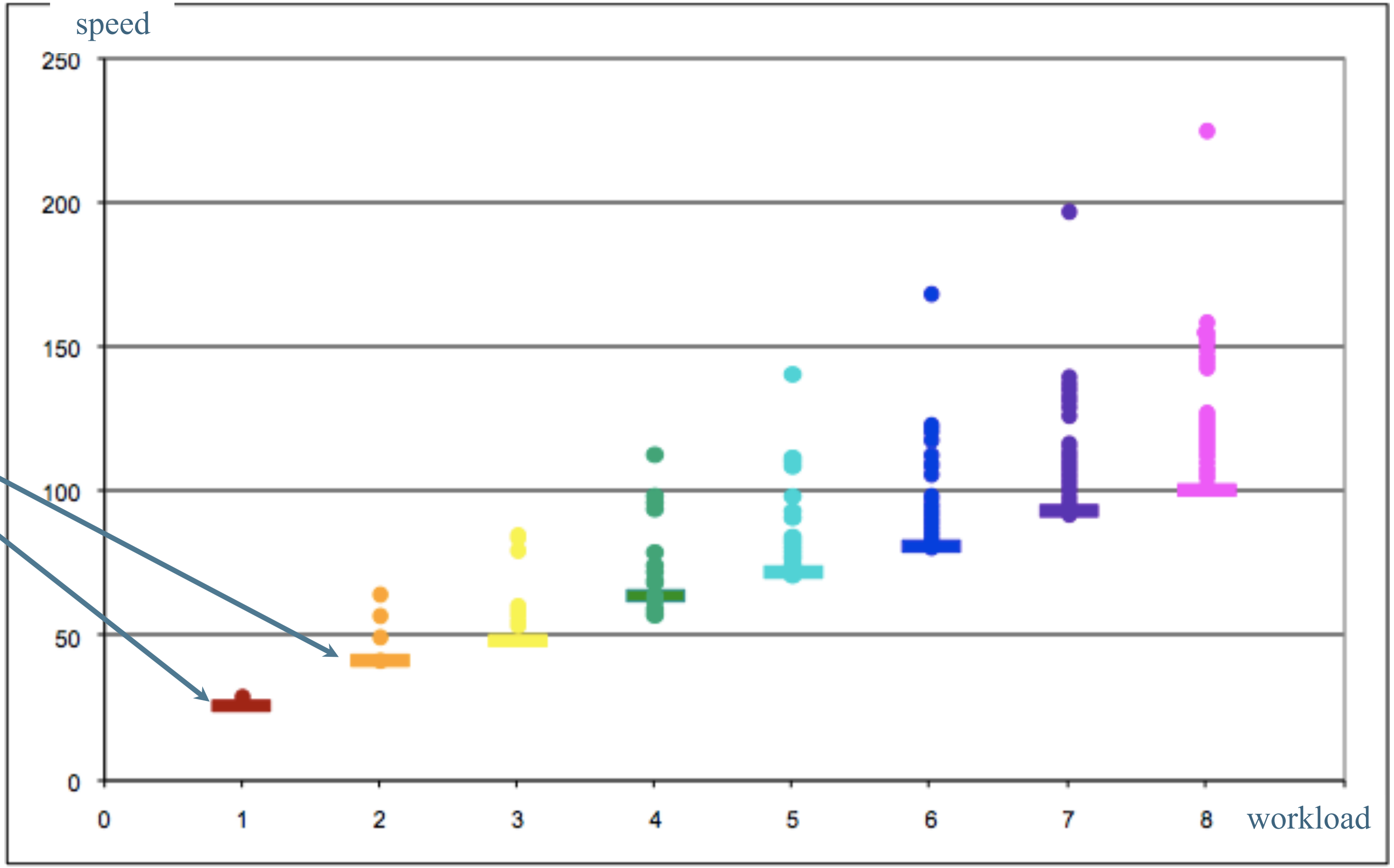
best static assignment on **1 GPU and 4-core CPU**:
sa A





ARES' Adaptive runtime resource management

ARES run-time load balancing for all workloads is almost always better than the best static assignment per workload



ARES' approach is portable across platforms

ARES run-time load balancing is portable to different configurations of heterogeneous processors:

exact same software stack adapts to underlying heterogeneous processors and achieves best performance all the time

(horizontal lines)

run-time load balancing on

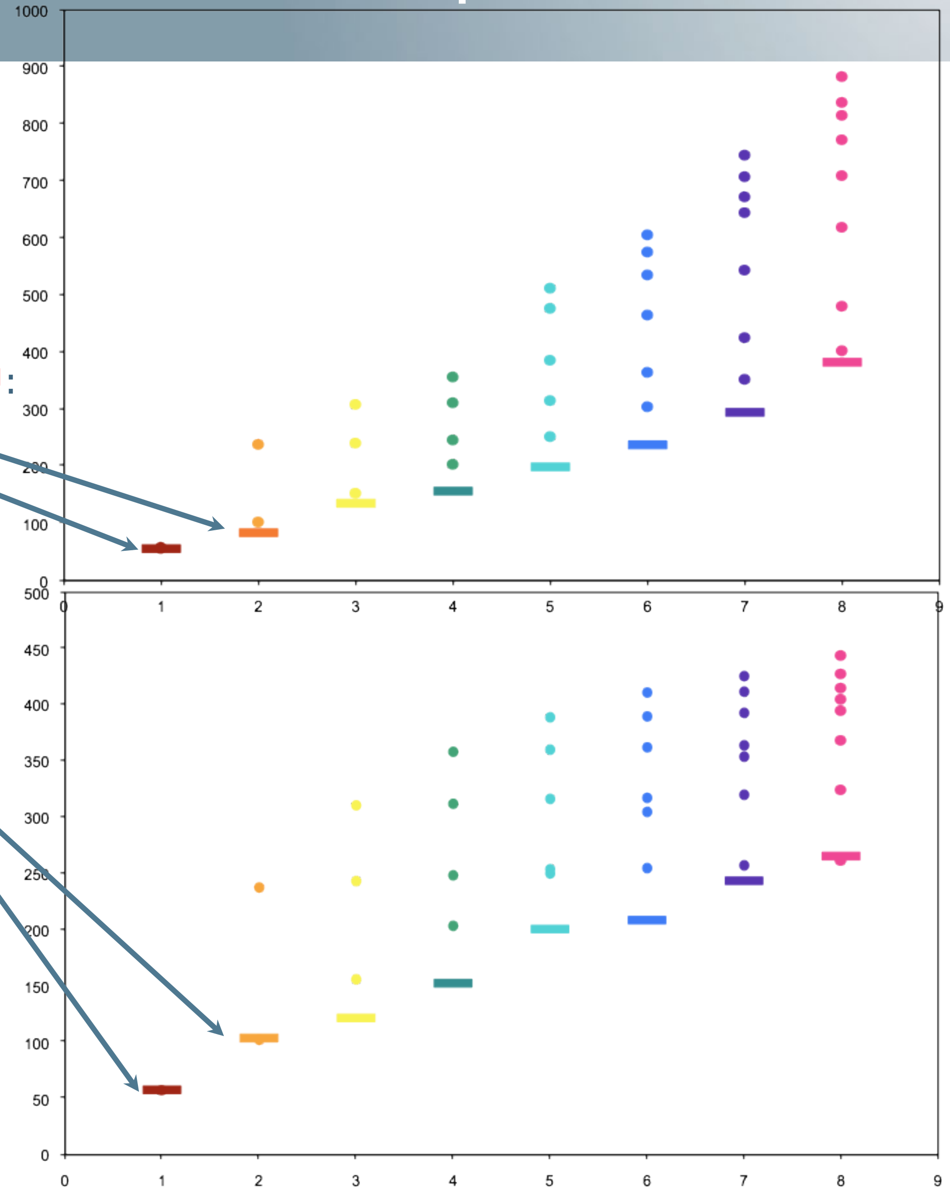
1 GPU and 2x 4-core CPU:

1 to 8 streams

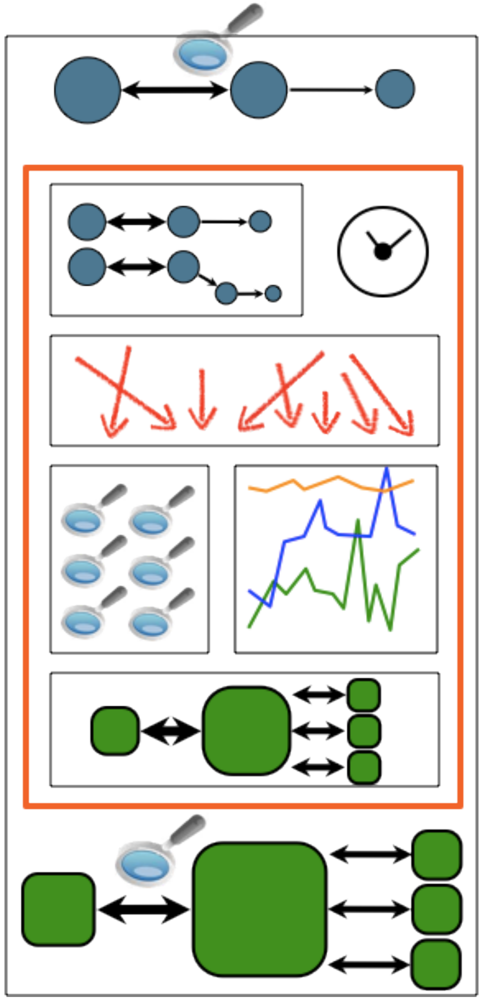
720p resolution

run-time load balancing on

1 GPU and 4-core CPU:



ARES Runtime Resource Management: parts



- Monitor
  resource assignment and usage
- Represent
  monitored information
- Decide assignment at runtime
  - use monitored information
  - predict, learn, adapt, ...
  - Pluggable strategies with different trade-offs

Monitoring and Representation Examples

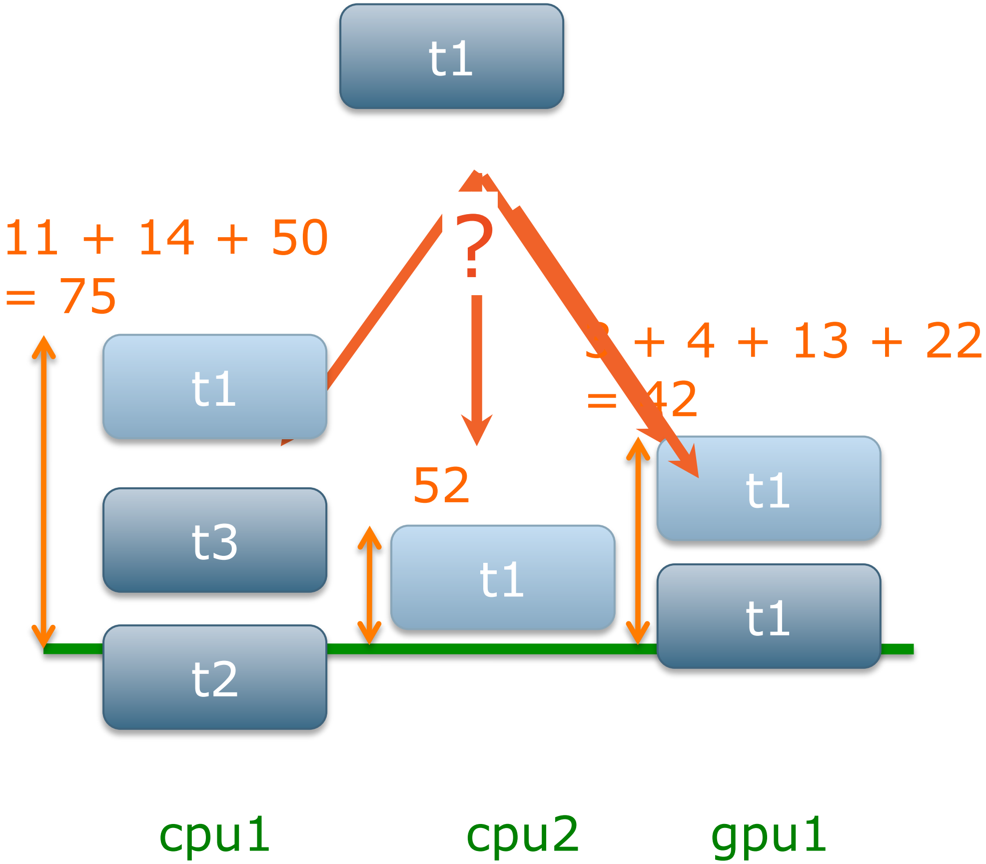
- We monitor:
  - execution time of a component on a processing element
  - data transfer times between two connected components executing on different type of processing element
- We represent:
  - Average time + standard deviation per component and per processing element

Task	PE	time	Dev.
t1	cpu1	50	9,3%
t1	cpu2	52	5,6%
t1	gpu1	4	4,1%
t1	->gpu	13	12,8%
t1	gpu->	32	8,4%
t2	cpu1	134	3,6%

Assignment Strategies

- Can use the following information:
  - Hardware metadata: static and runtime
  - Software metadata: static and runtime
- Have to respond to assignment requests
  - Fast response is required
- Different algorithms are possible
  - Static (up-front) decision: no runtime adaptation (SoA)
  - Generic: fastest available, first finished
  - Domain-specific: prefer-GPU-sequence
  - Machine learning

Example: First Finish Strategy



Task	PE	time	Dev.
t1	cpu1	50	9,3%
t1	cpu2	52	5,6%
t1	gpu1	4	4,1%
t1	->gpu	13	12,8%
t1	gpu->	32	8,4%
t2	cpu1	134	3,6%
t3	cpu1	14	6,2%
…			

Implementation

- ARES Runtime resource manager implementation:
  - Dynamic library for Unix (Linux, OS-X) and Windows
  - C and C++ header for integrating with applications
  - Uses Boost shared memory to store values
  - Low-overhead (0,01%)
- Used with:
  - AVC Encoder (CUDA-accelerated motion estimation)
  - GStreamer applications
  - Imec in-house multimedia framework in .Net on Windows

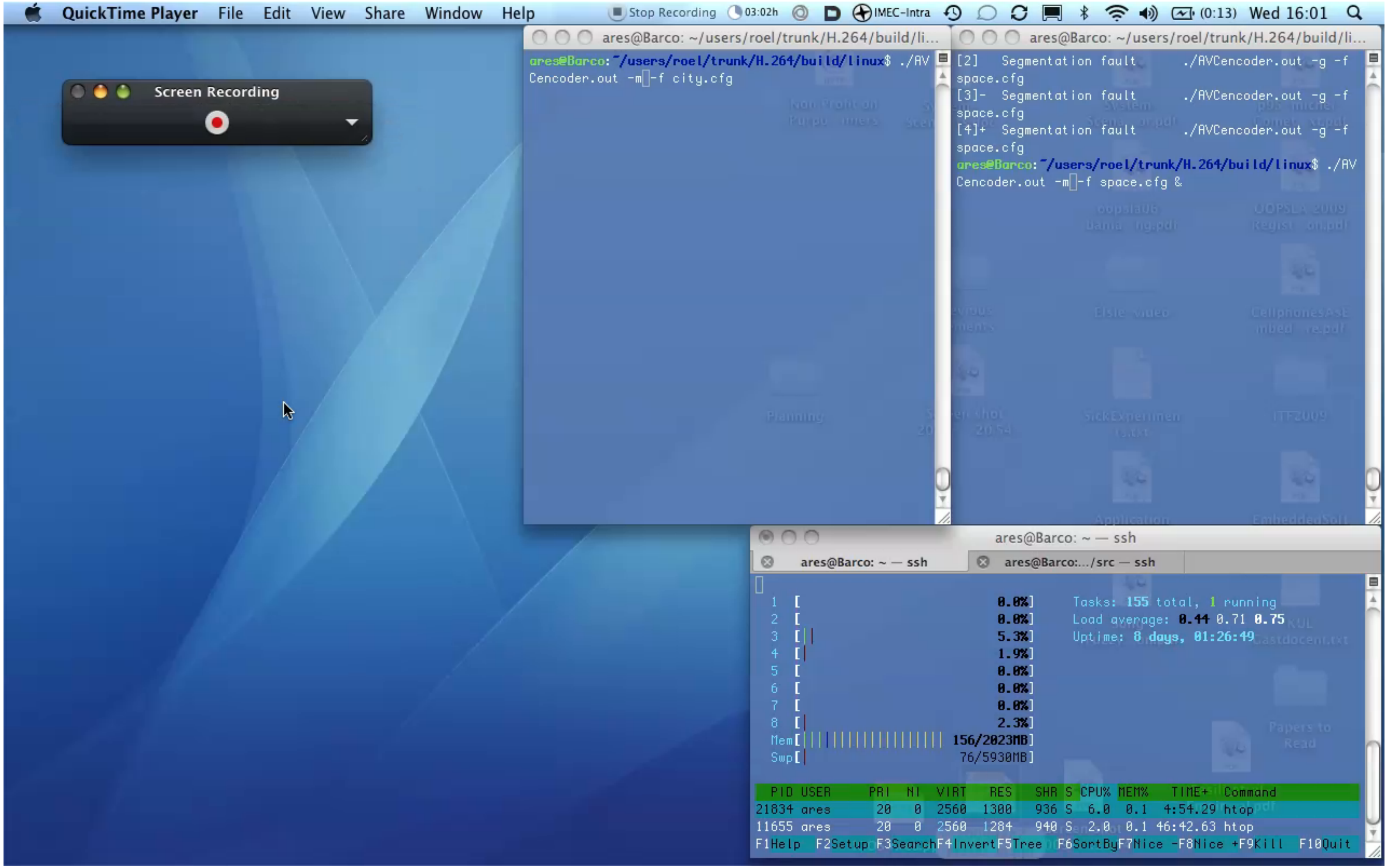
Making the AVC Encoder runtime managed?

```
...
//ask RRM to decide between GPU or CPU
proc_type  = rrm_get_processor(encID);
if (  RRM_PROC_TYPE(proc_type) == RRM_PROC_GPU  )
     cuda_me = 1;
else
     cuda_me = 0;

start2 = RDTSC ();
if (cuda_me == 1) {
  start = RDTSC ();
  GPUinit();
  cuda_motion_estimation();
  GPUExit();
  g_total_MEtime = (RDTSC () - start);
}
else {
  ((ARMVCM4P10_MESpec *)encInfo.params.meSpec)->no_gpu_data();
}
...

//update RRM execution time
update_kernel_timing(encID, proc_type, g_total_MEtime);
...
```

2 lines

1 line

City + Space: runtime managed



But...

- … what runtime management strategy works best for my application ?

- … will my existing application benefit from runtime management ?
- … will my new application benefit from runtime resource management?
- … what if my clients use a dualcore CPU and 2 GPU’s ?

Exploration Tool

- Compare different runtime resource management strategies
- How?
  - Model software at high-level (connected components).
  - Decorate nodes with timing information:
    - Average execution times per processing element supported;
    - Data transfer Times between different processing elements.
    - These timings come from the runtime manager, from other profiling tools, from experience, or even from guestimates.
  - Model kind and number of processing elements.
  - Select the strategies you want to compare.
- Result?
  - Exploration tool simulates the execution for each strategy and outputs information that can be plotted (dropped frames, late frames, platform utilization)

Exploration Tool Input

Application 1

Source
Execution time:
CPU: 0.011ms

Demux
Execution time:
CPU: 0.00163ms

Scaler
Execution time:
CPU: 0.115ms
GPU: 0.0129ms
Data Transfer Time:
CPU->GPU: 0.079ms
GPU->CPU: 0.17ms

Sink
Execution time:
CPU: 0.51ms

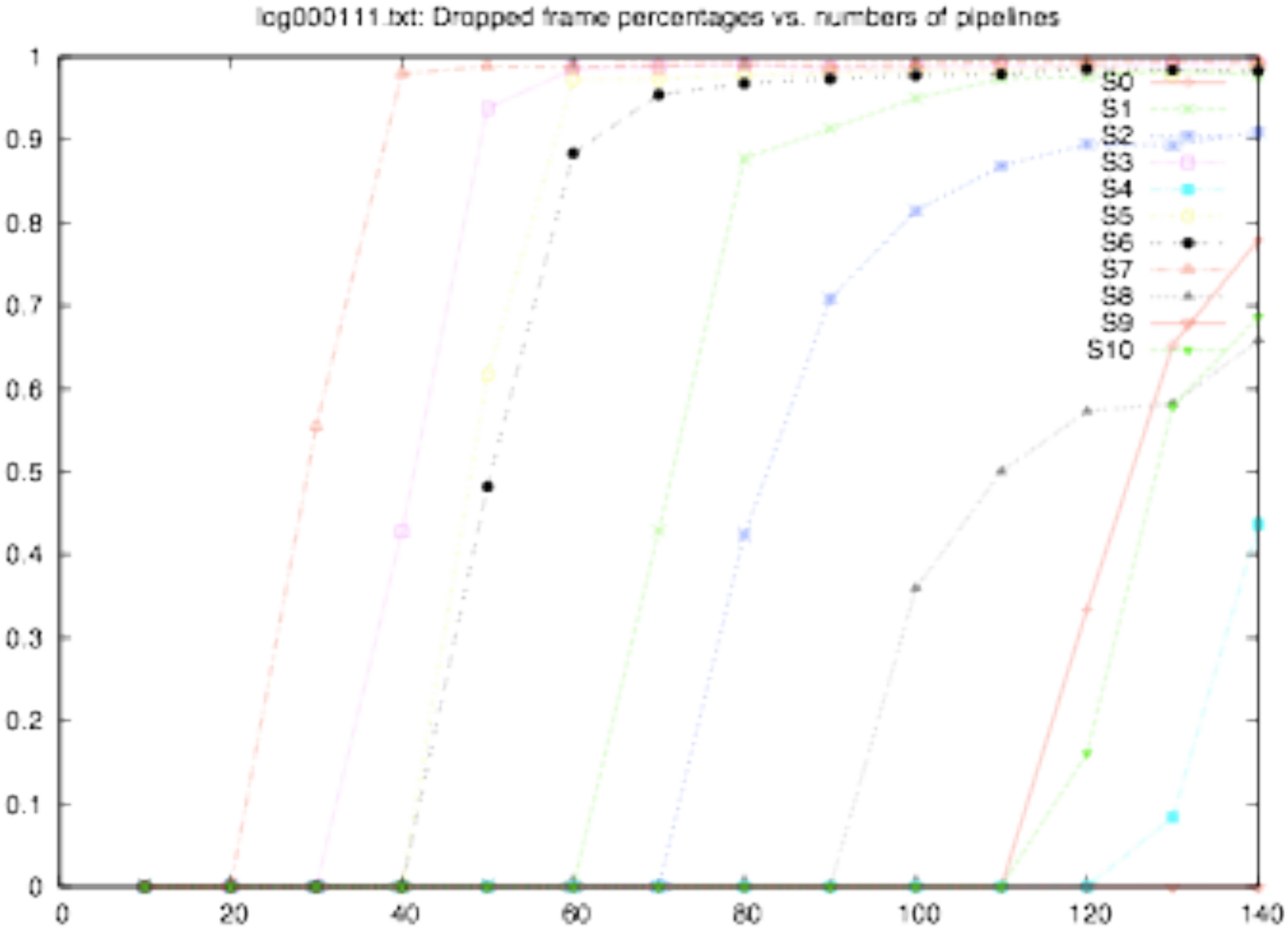
Application 2

Encoder
Execution time:
CPU: 0.43ms
GPU: 0.074ms
Data Transfer Time:
CPU->GPU: 0.085ms
GPU->CPU: 0.46ms

Hardware Description

8 CPU
1 GPU
2 DMA

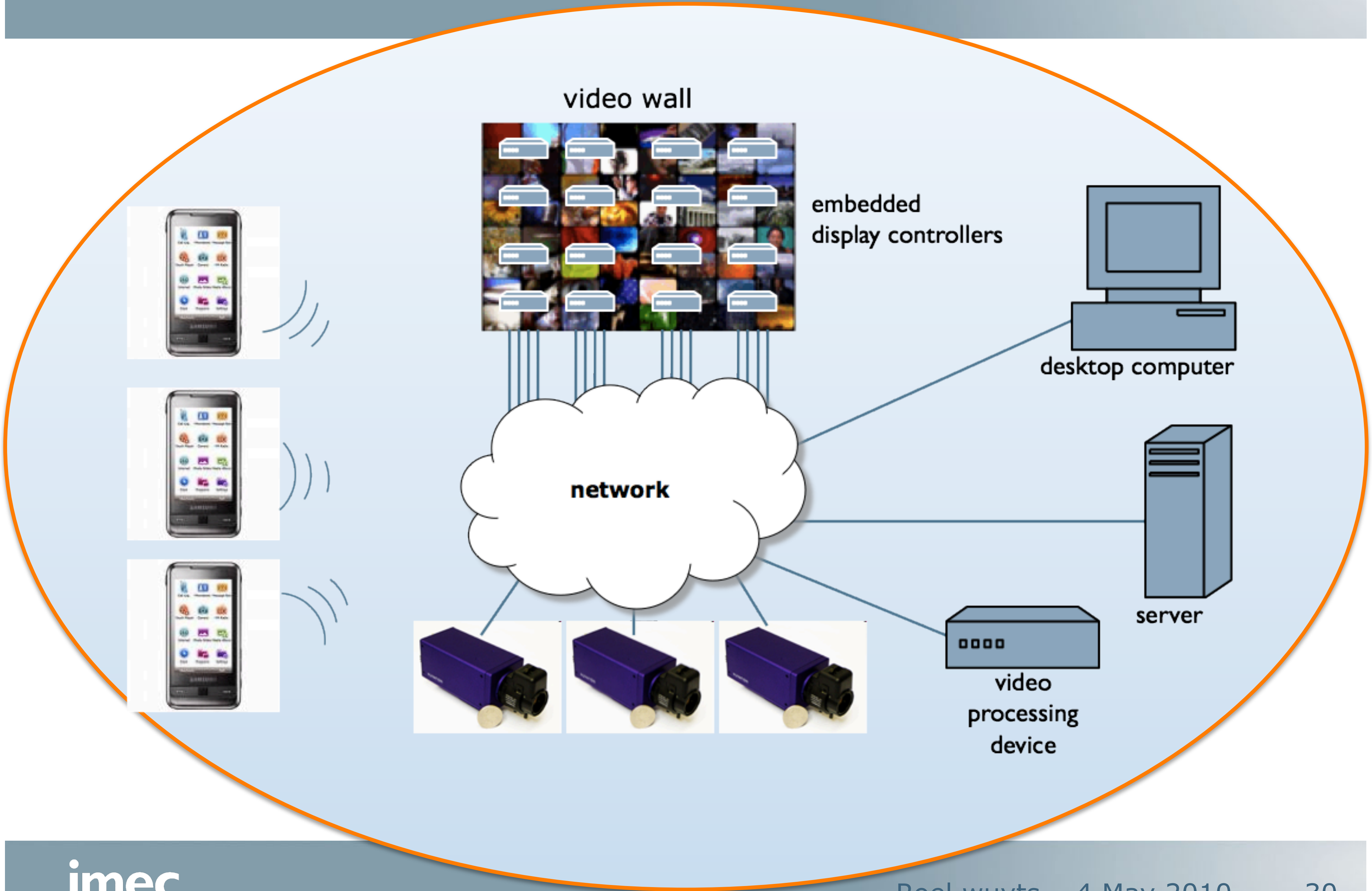
Plotted outputresult



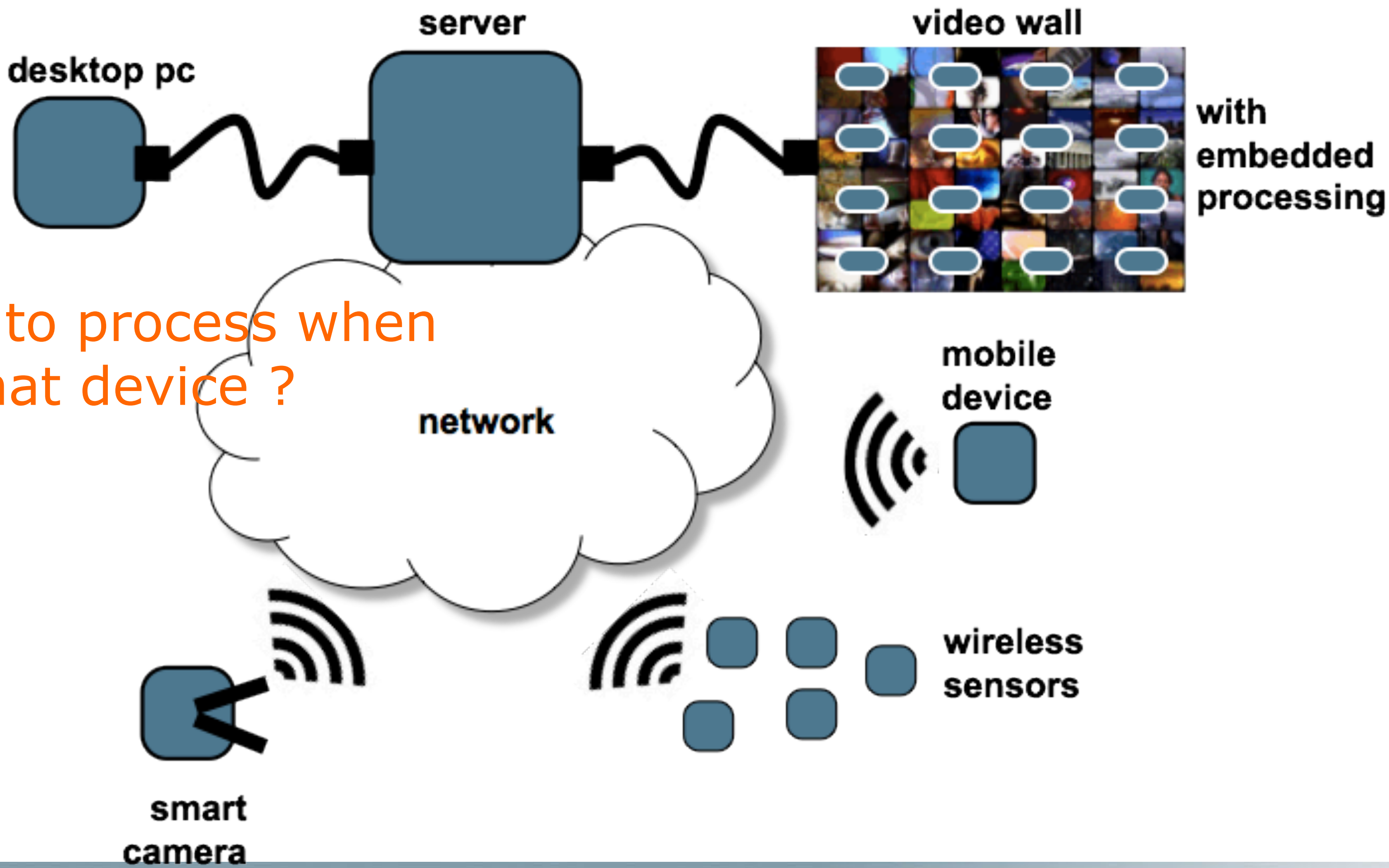
Device-level adaptive resource management

- Static assignments exhibit problems
  - Different solutions for different workloads or other runtime variability
  - Do not scale (exploration space explosion)
  - Different solutions for different platforms
- Runtime resource managed solution adapts to different conditions
  - Runtime variability
  - Heterogeneous platforms

Context: Networked Video Processing

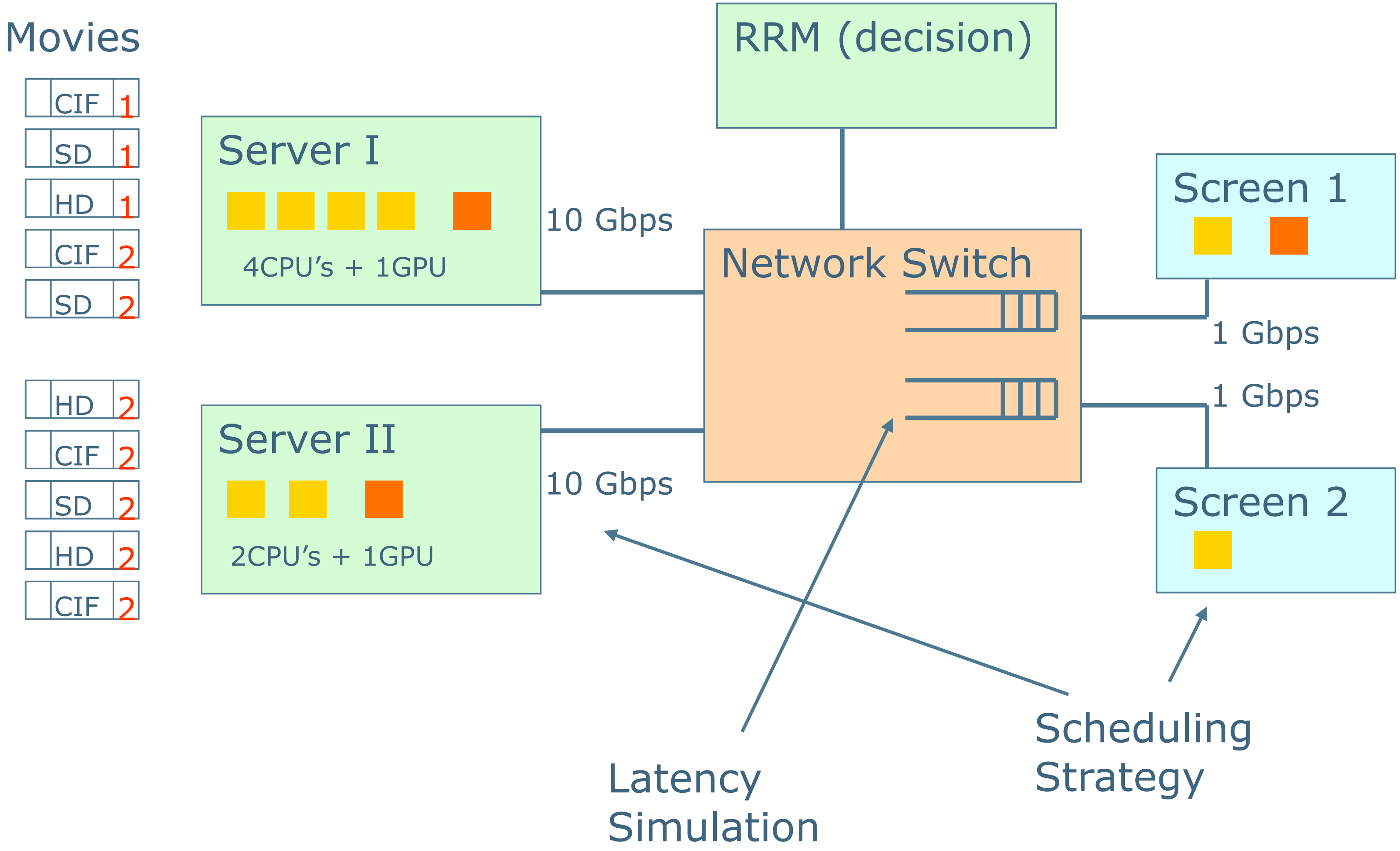


Network: Connected heterogeneous devices

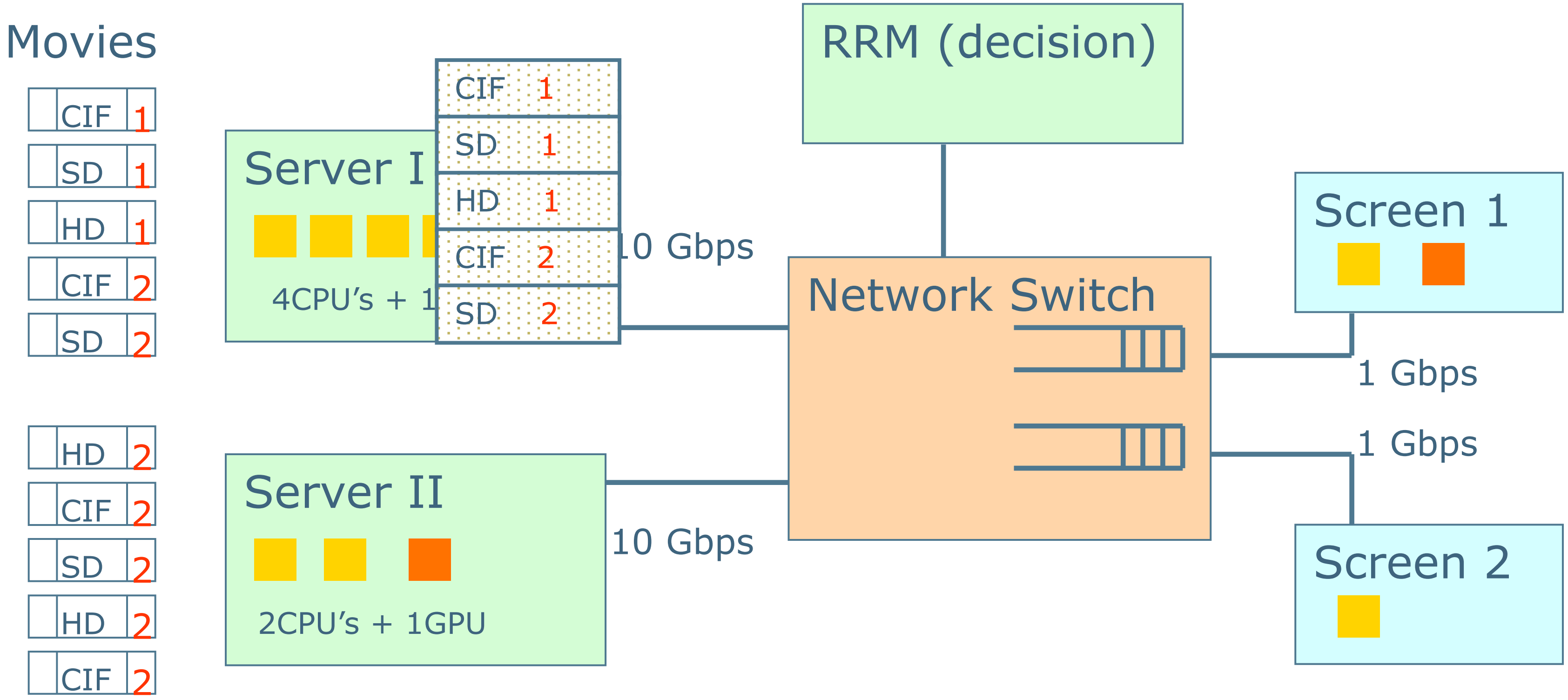


What to process when
on what device ?

System-wide Resource Allocation

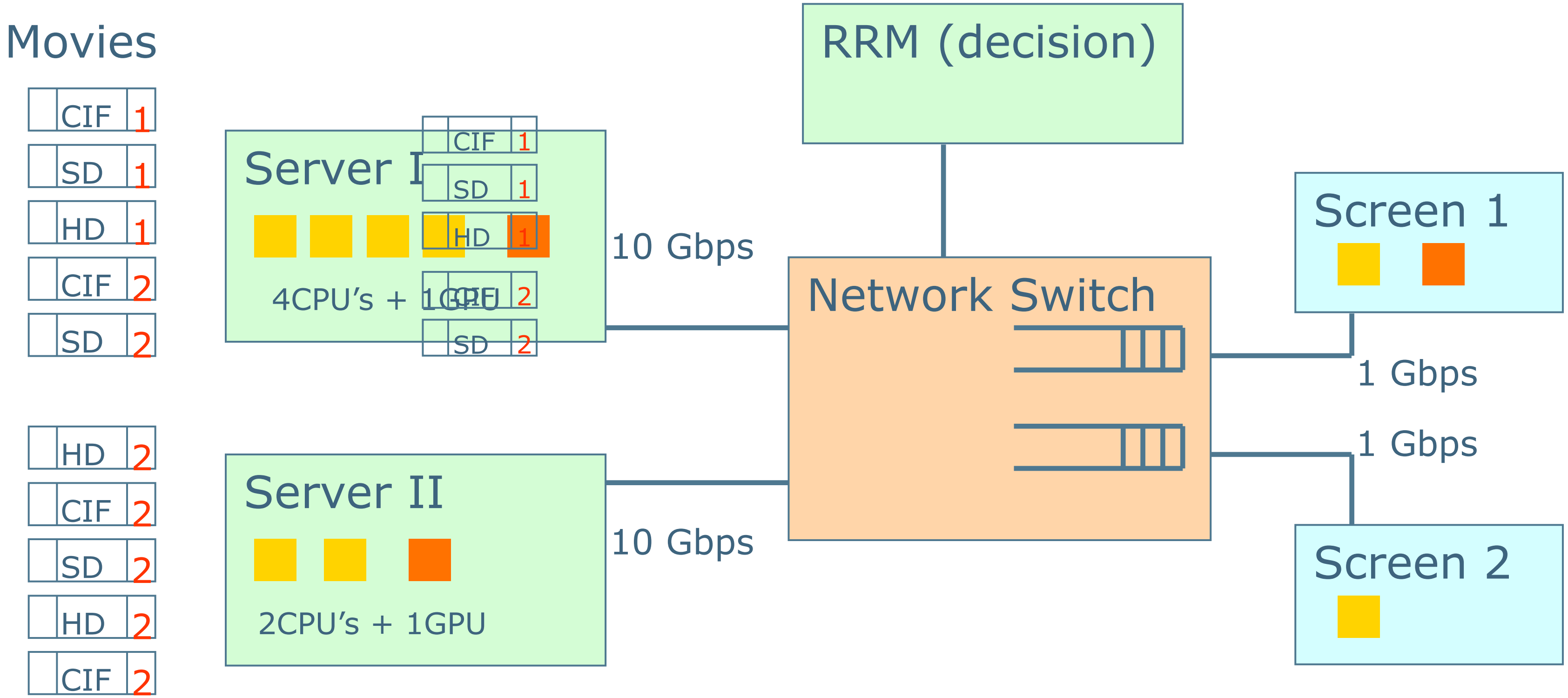


Possibility: Everything decoded at server, raw data to client



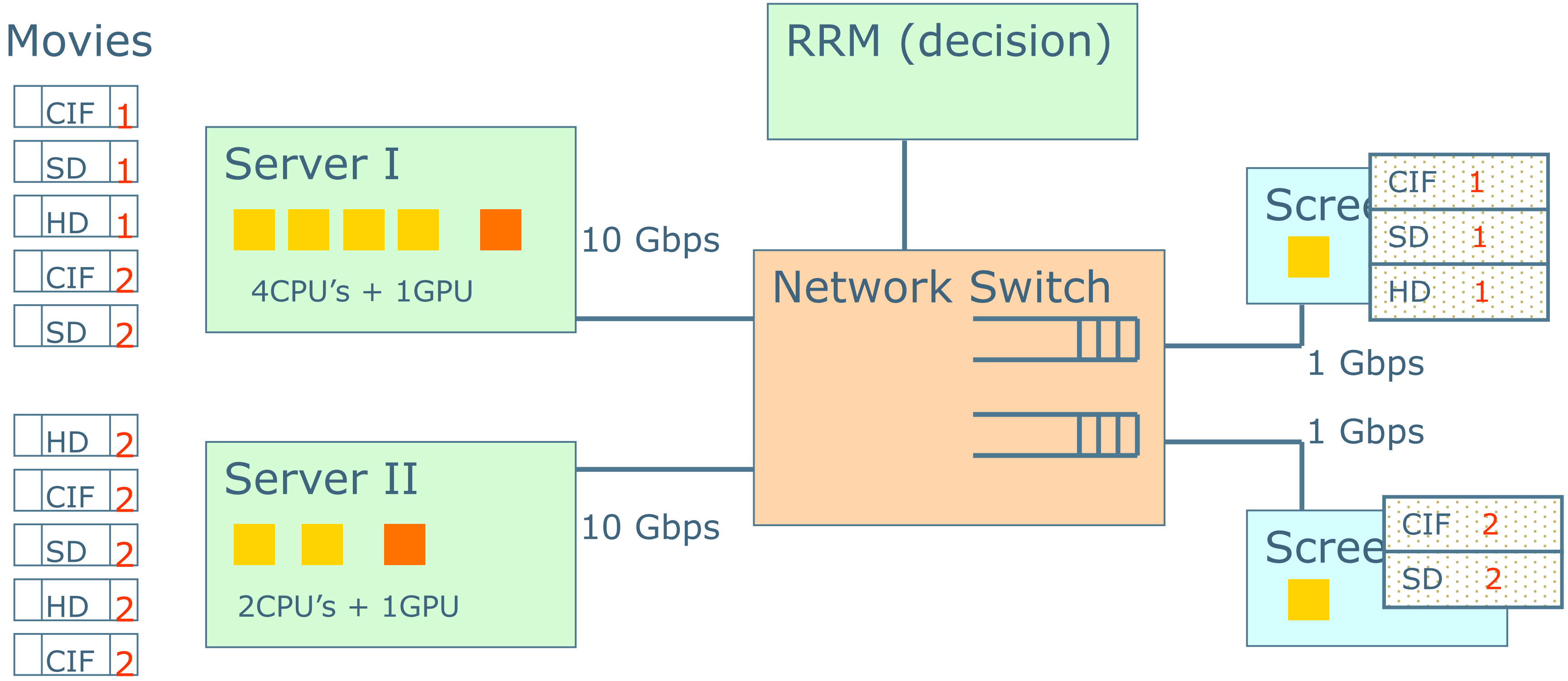
(-) Increases Bandwidth and network latency
(+) No processing cost at client

Possibility: Everything trans-coded at server, lower resolution sent to client



(+) Reduces Bandwidth and network latency
(+) Lower processing cost at client
(-) Increases processing cost at server

Possibility 4: Everything fully decoded at client



(-) All processing at client, might miss deadlines
(+) No BW or latency increase

network-level: first results

- distributed processing of video processing applications on server or client
- trade-off between processing at server, processing at client or transcoding to lower quality
- adaptive run-time resource management - using a mixture of the above - gives good results:

	Processed at Server	Processed at Client	Transcode	Mixed Processing
Resolution/Quality	High	High	Low	Medium-High
Missed Streams	6 (limited bw)	5	0	0
BW (Gbps)	2.7	0.6	0.08	0.95
Latency (ms)	92	10	2.9	33

Networked Video Processing: Future Work

- Discover Distributed Processing Strategies
  - Trading of bandwidth, processing power and quality
- Implement
  - Currently extending the device-level manager

Conclusion

- Problem: how to develop software that runs on heterogeneous devices
  - At SoC level
  - At Device level
  - AtNetwork Level
- Solution: runtime decision strategies decide what software component uses what resource

- Meta Remark: versatility of your studies make you valuable assets
- Meta Meta Remark: Choose according to Flexibility versus Pay