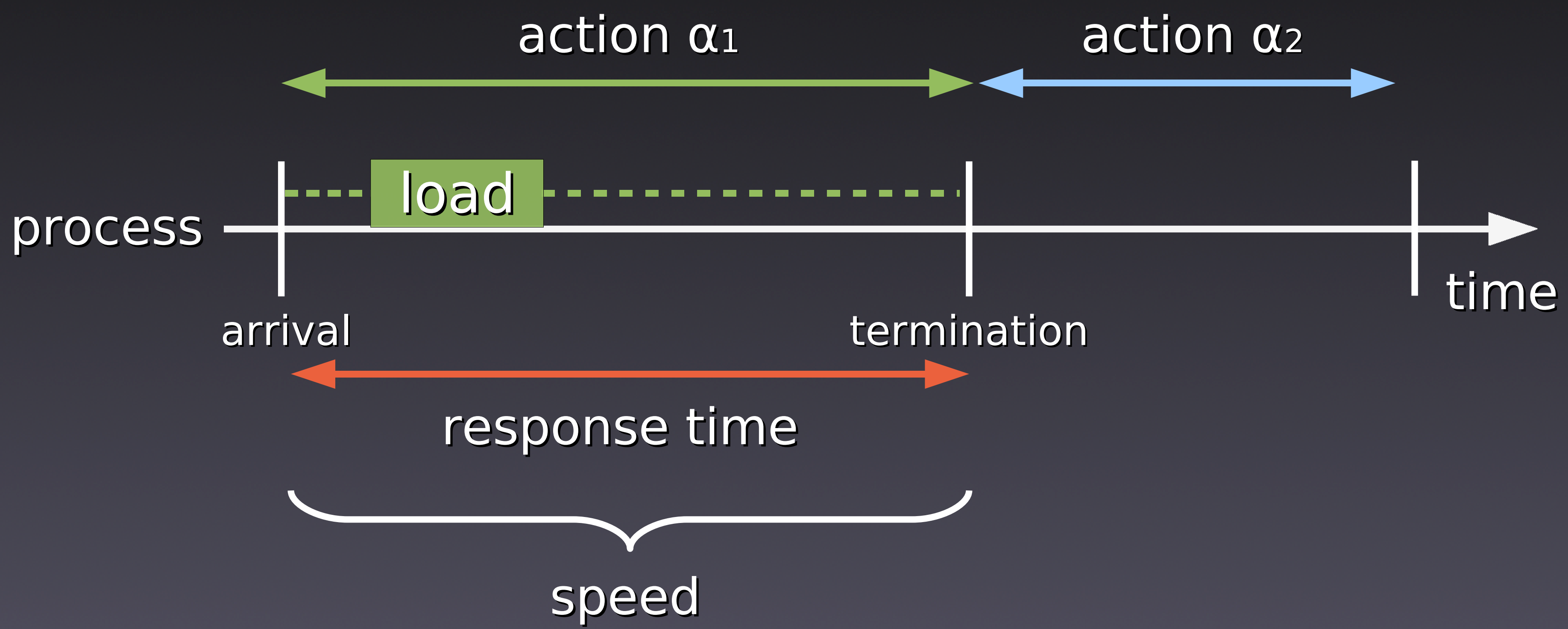
Programmable Temporal Isolation through Variable-Bandwidth Servers

Silviu S. Craciunas

Department of Computer Sciences
University of Salzburg

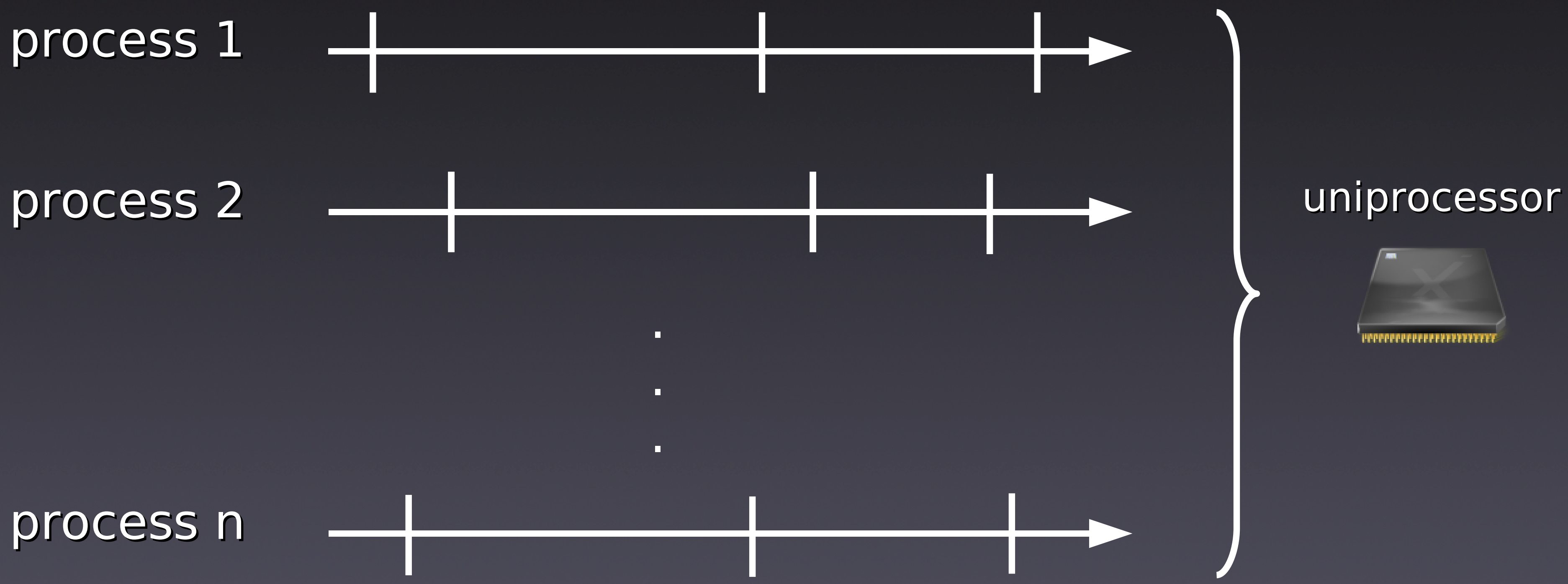
joint work with
C. Kirsch, H. Payer, H. Röck, and A. Sokolova

Process model



- action is a piece of code
- process is a sequence of actions

Problem



schedule the processes so that each of their actions maintains its response time

Goal

- solve the scheduling problem

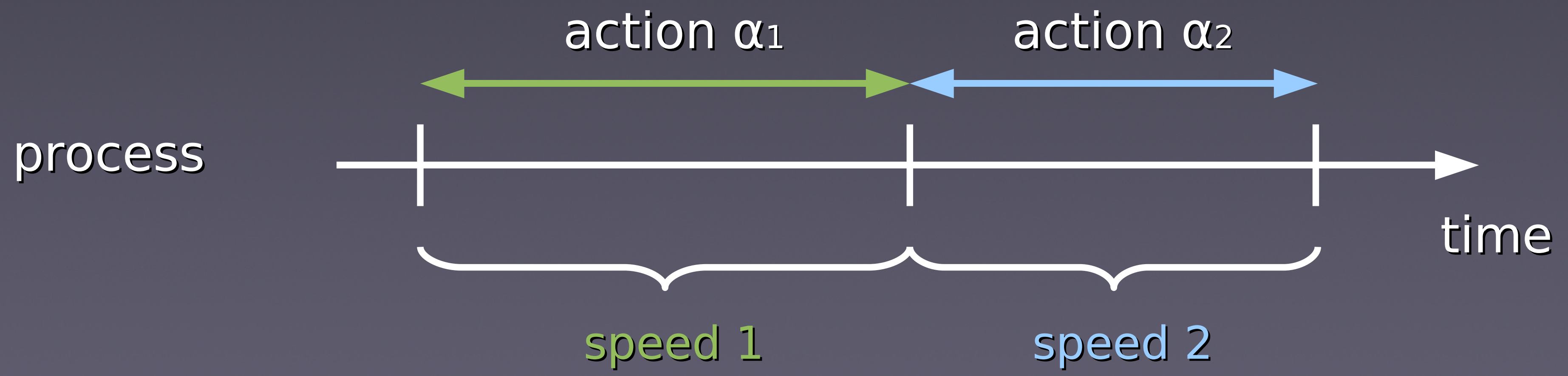
  (temporal isolation)

- change execution speed of processes

  (programmable)

- solve admission problem

  (changeable set of processes)



Goal

- solve the scheduling problem

lation)

- cha

Solvable with variable bandwidth servers (VBS)

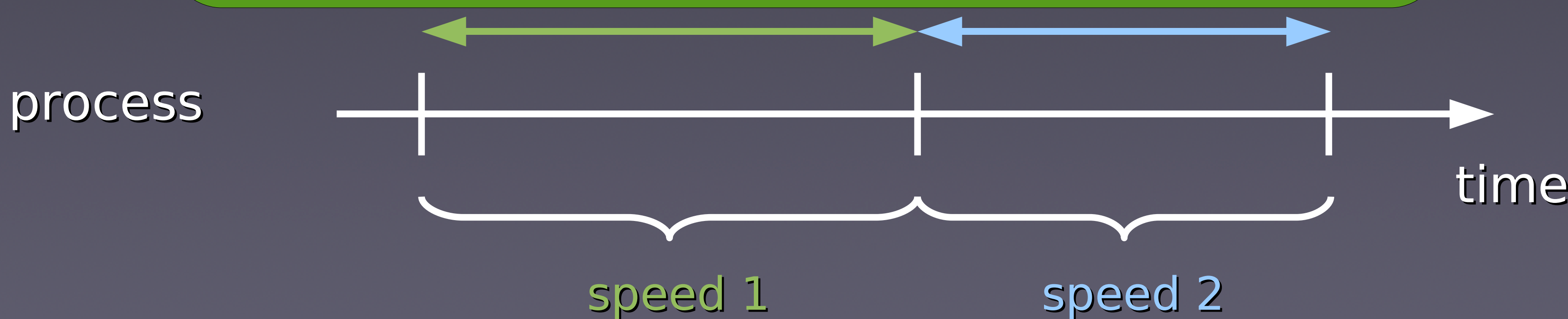
programmable)

- solv

ses)

Results:

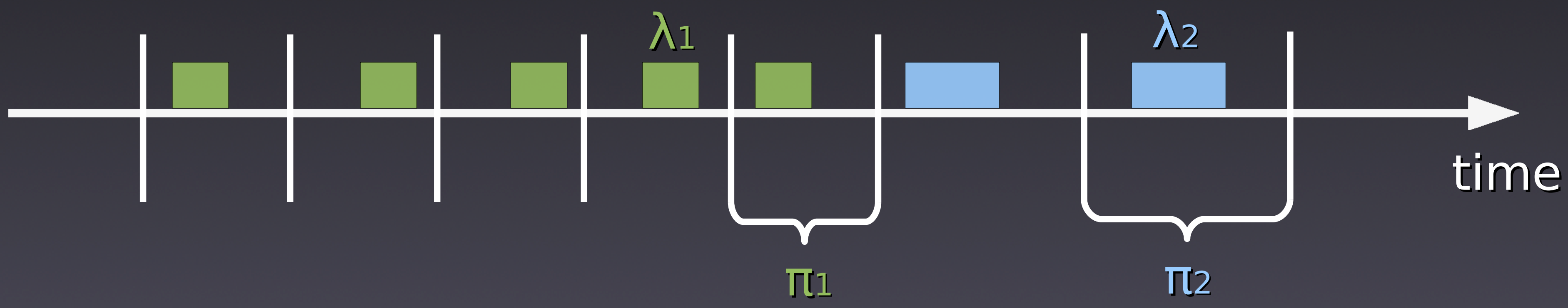
- a constant-time scheduling algorithm
- a constant-time admission test



Resources and VBS

virtual periodic resources

period π limit λ utilization λ/π



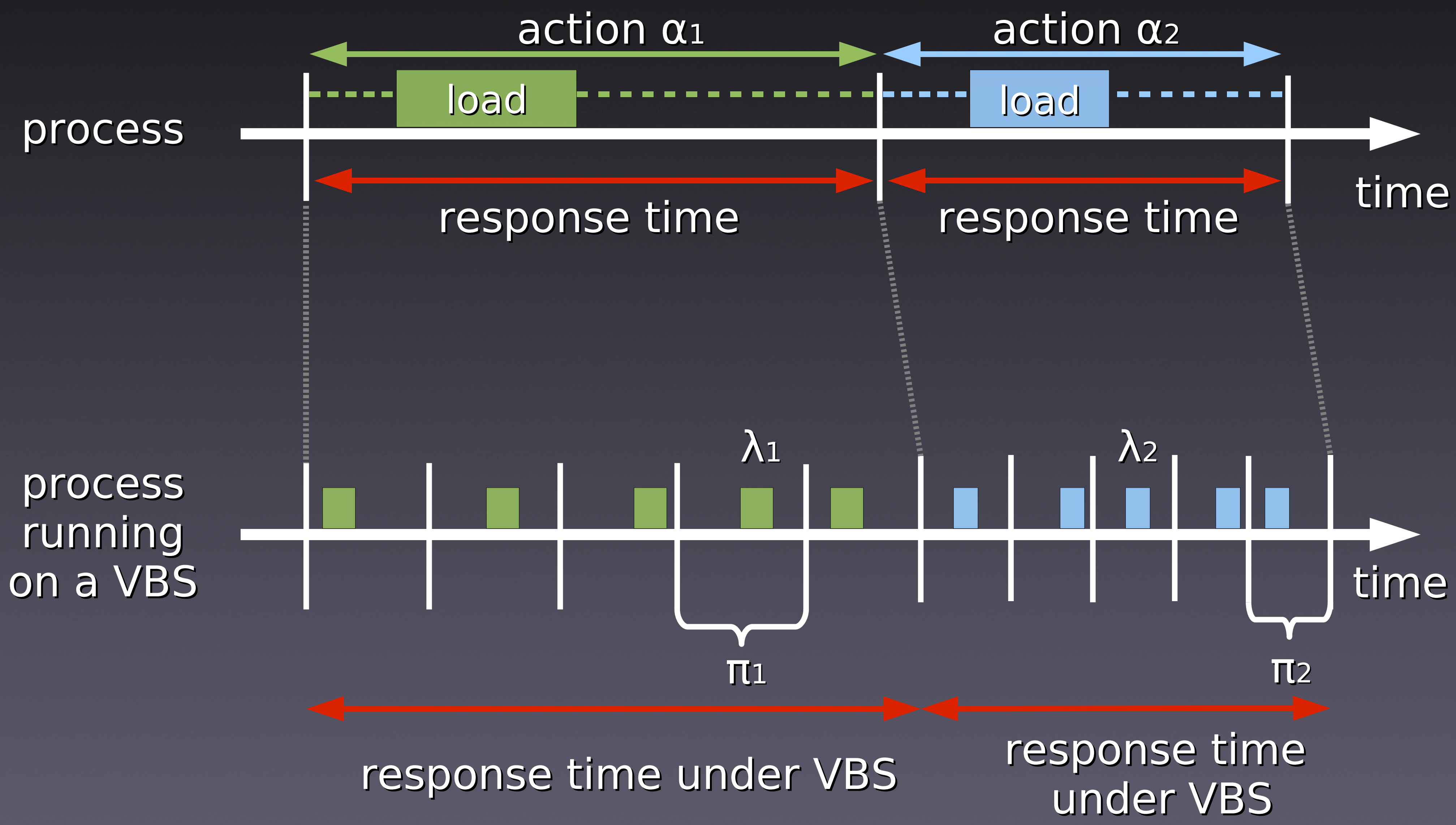
- VBS is determined by a bandwidth cap (u)
- VBS processes dynamically adjust speed (resource)

$$\lambda_1/\pi_1 \leq u \text{ and } \lambda_2/\pi_2 \leq u$$

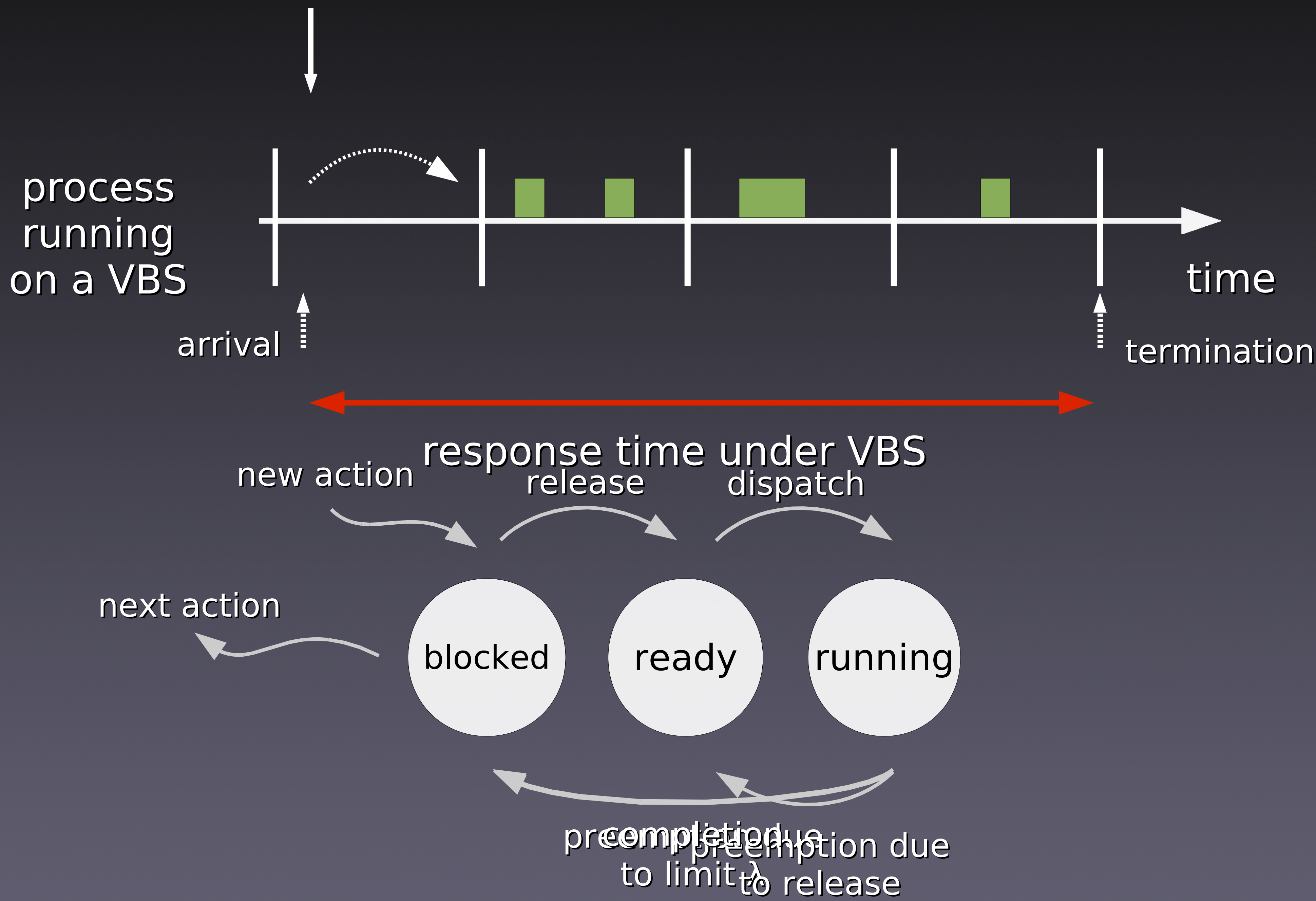
- generalization of constant bandwidth servers (CBS)

[Abeni and Buttazzo 2004]

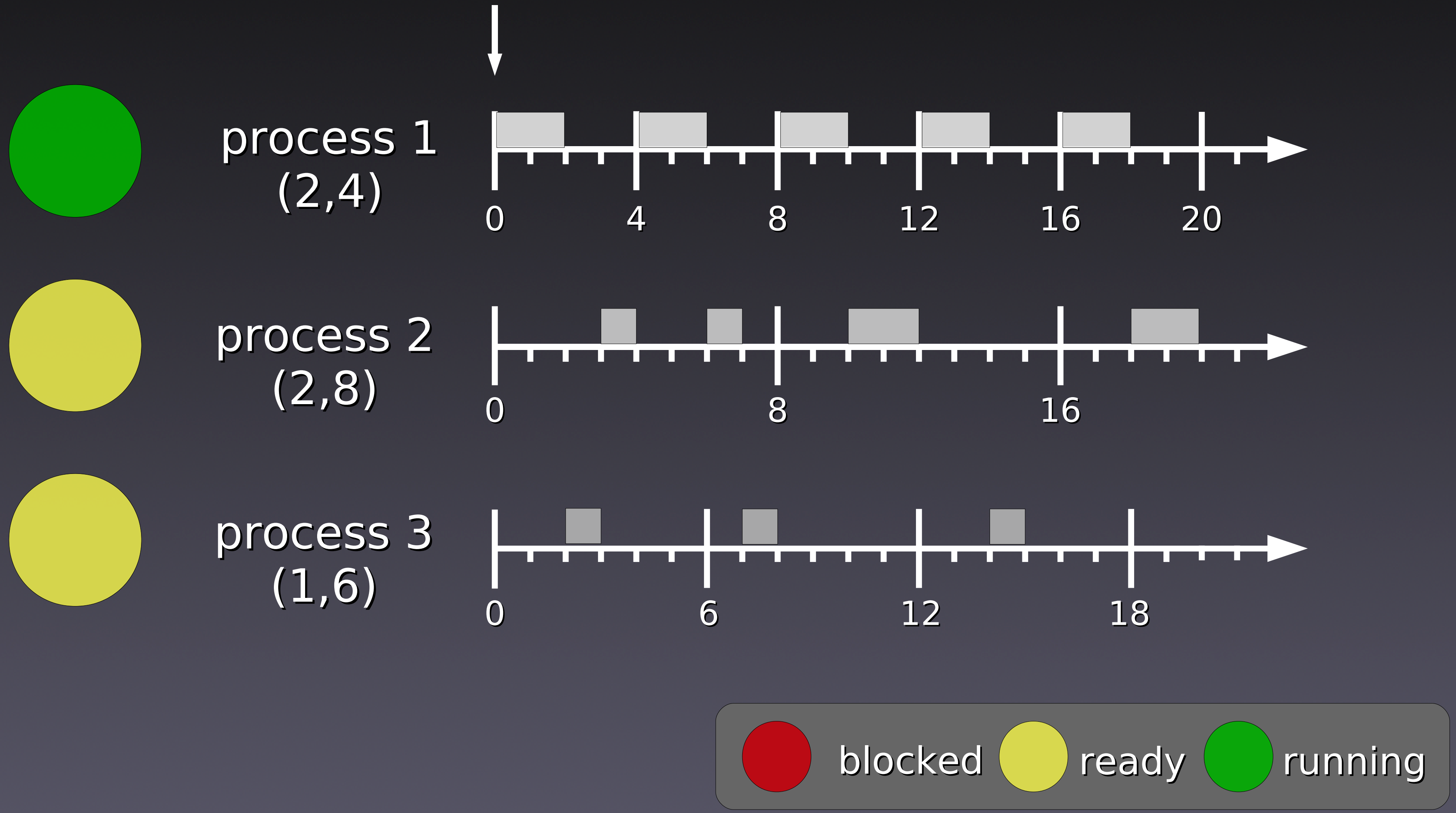
One process on a VBS



VBS


process running on a VBS
time
arrival
termination
response time under VBS
new action
release
dispatch
next action
blocked
ready
running
preemption due to limit λ
completion
preemption due to release

VBS



multiple processes are EDF-scheduled

Scheduling result and bounds

Processes P1,P2, ... ,Pn on VBSs u1,u2, ... ,un, are schedulable

if $\sum u_i \leq 1$

For any action α on a resource (λ,π) we have

upper response time bound
$\lceil load / \lambda \rceil \pi + \pi - 1$

lower response time bound
$\lceil load / \lambda \rceil \pi$

jitter
$\pi - 1$

Programmable temporal isolation

the “speed“ of an action is programmable
(influencing response time and jitter)

smaller π ⇒

+ smaller jitter

+ VBS response time closer to „ideal“ response time

- higher administrative overhead

(more scheduler invocations)

Finding the right λ,π is difficult.

server design problem

Real-world example

```
loop {

  sensor_data = read(sensors);
  actuator_data=compute(sensor_data);
  write(actuator_data);

  log(actuator_data);
  update_internal_state();

} until (done);
```

low latency

less stringent

control-loop period

Real-world example

```
loop {

  sensor_data = read(sensors);
  actuator_data=compute(sensor_data);
  write(actuator_data);

  log(actuator_data);
  update_internal_state();

} until (done);
```

action 1

action 2

control-loop period

different throughput and latency requirements
for different portions of code

Implementation

- constant-time scheduling algorithm
- different queue management plugins

(lists, arrays, matrices, trees)

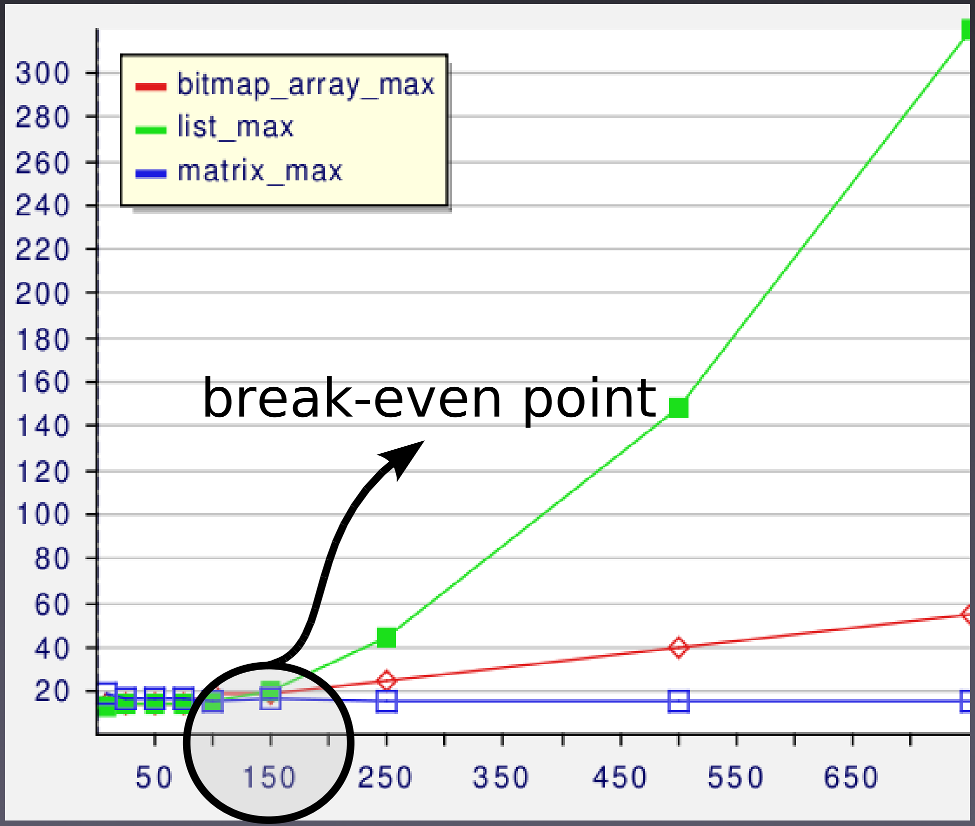
trade off time and space complexity

	list	array	matrix/tree
time	$O(n^2)$	$O(\log(t)+n\log(t))$	$\Theta(t)$
space	$\Theta(n)$	$\Theta(t+n)$	$O(t^2+n)$

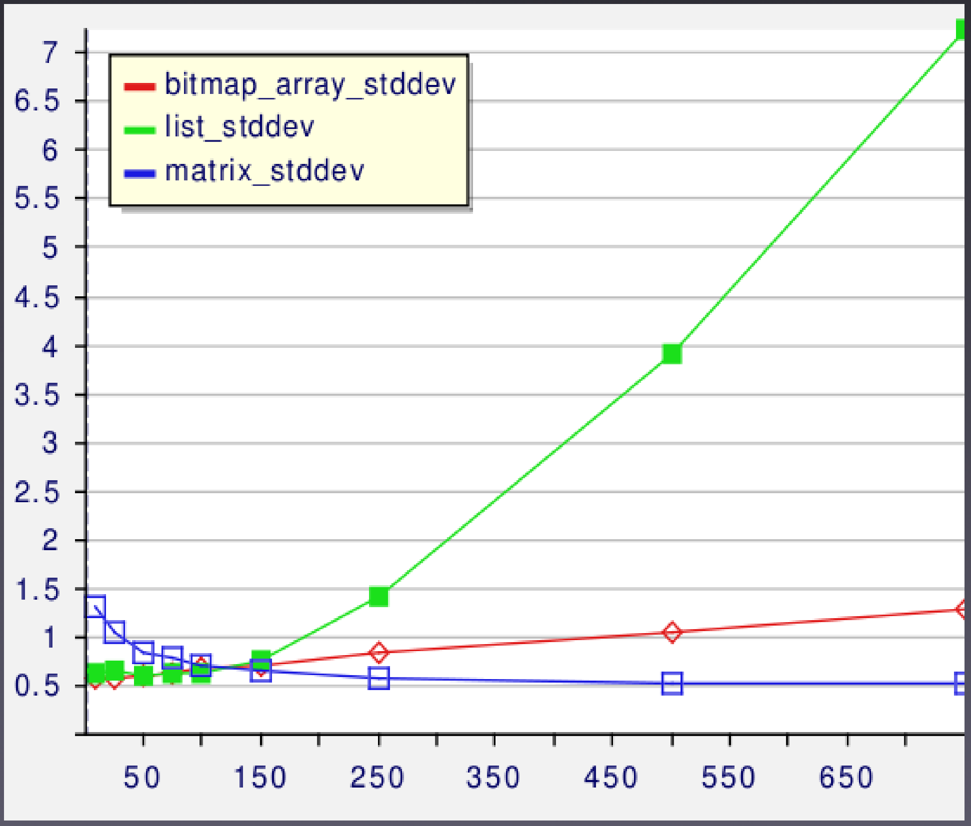
n – number of processes t – number of time instants

Results

scheduler overhead



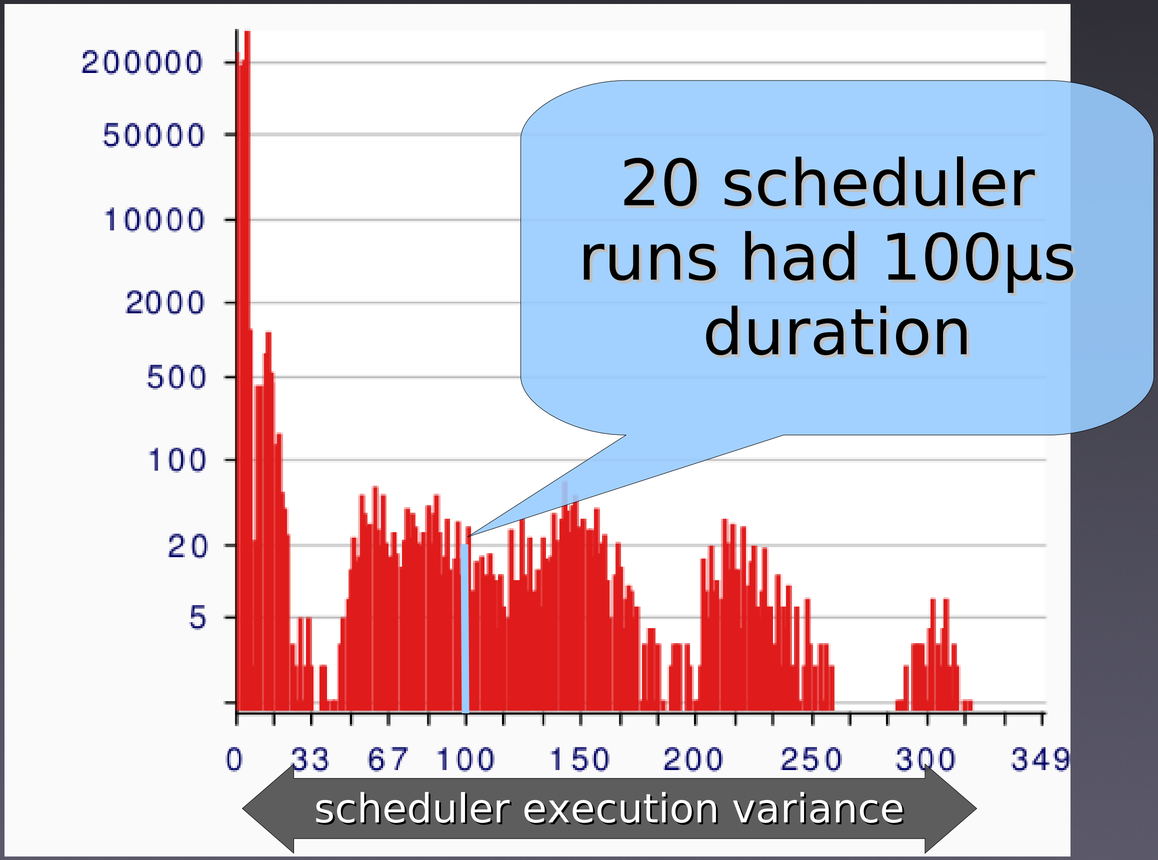
Maximum duration(µs) for increasing number of processes



Standard deviation for increasing number of processes

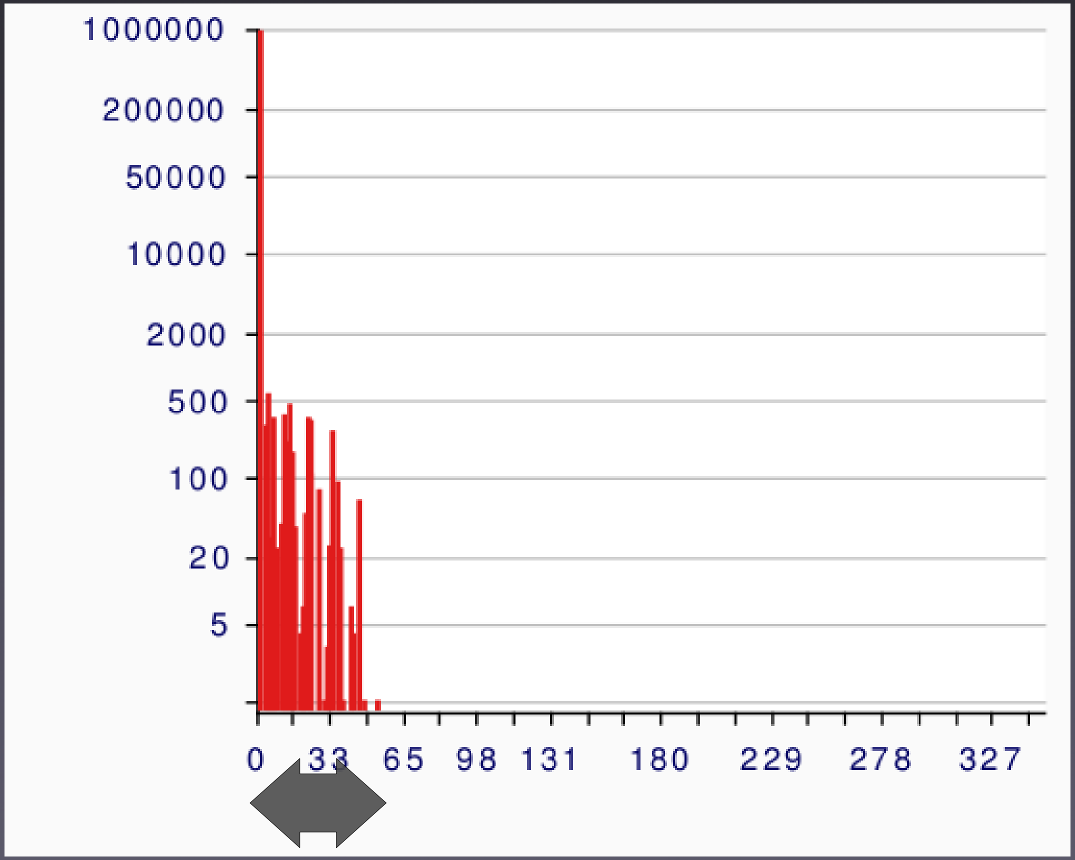
Results

scheduler overhead (list)



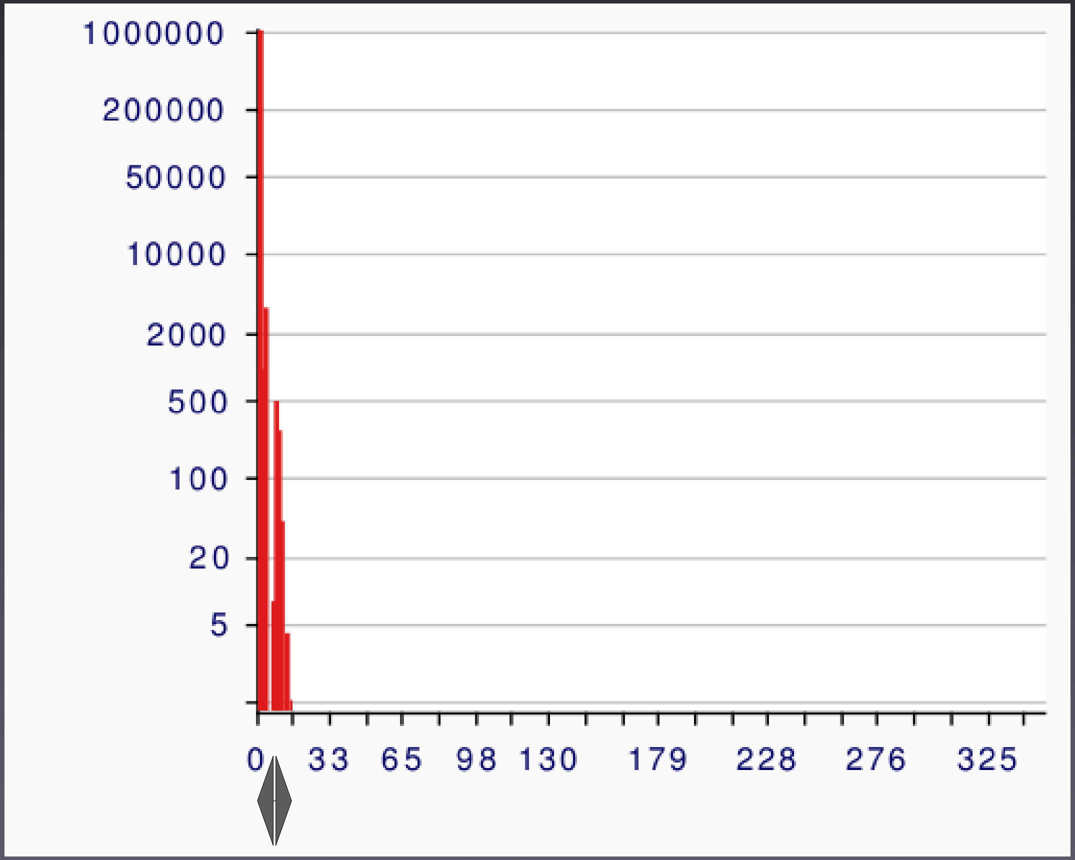
Results

scheduler overhead (array)



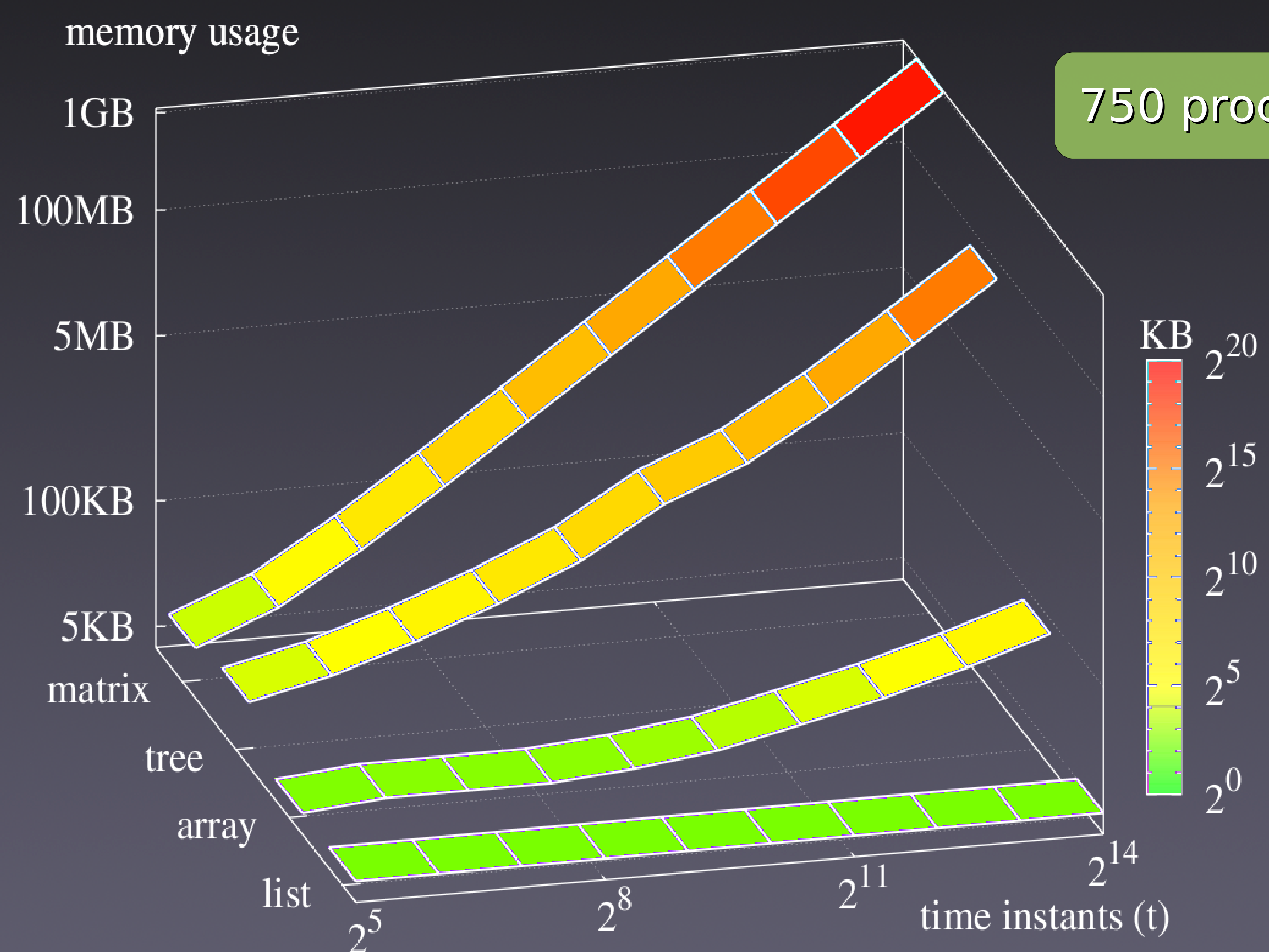
Results

scheduler overhead (matrix)



Results

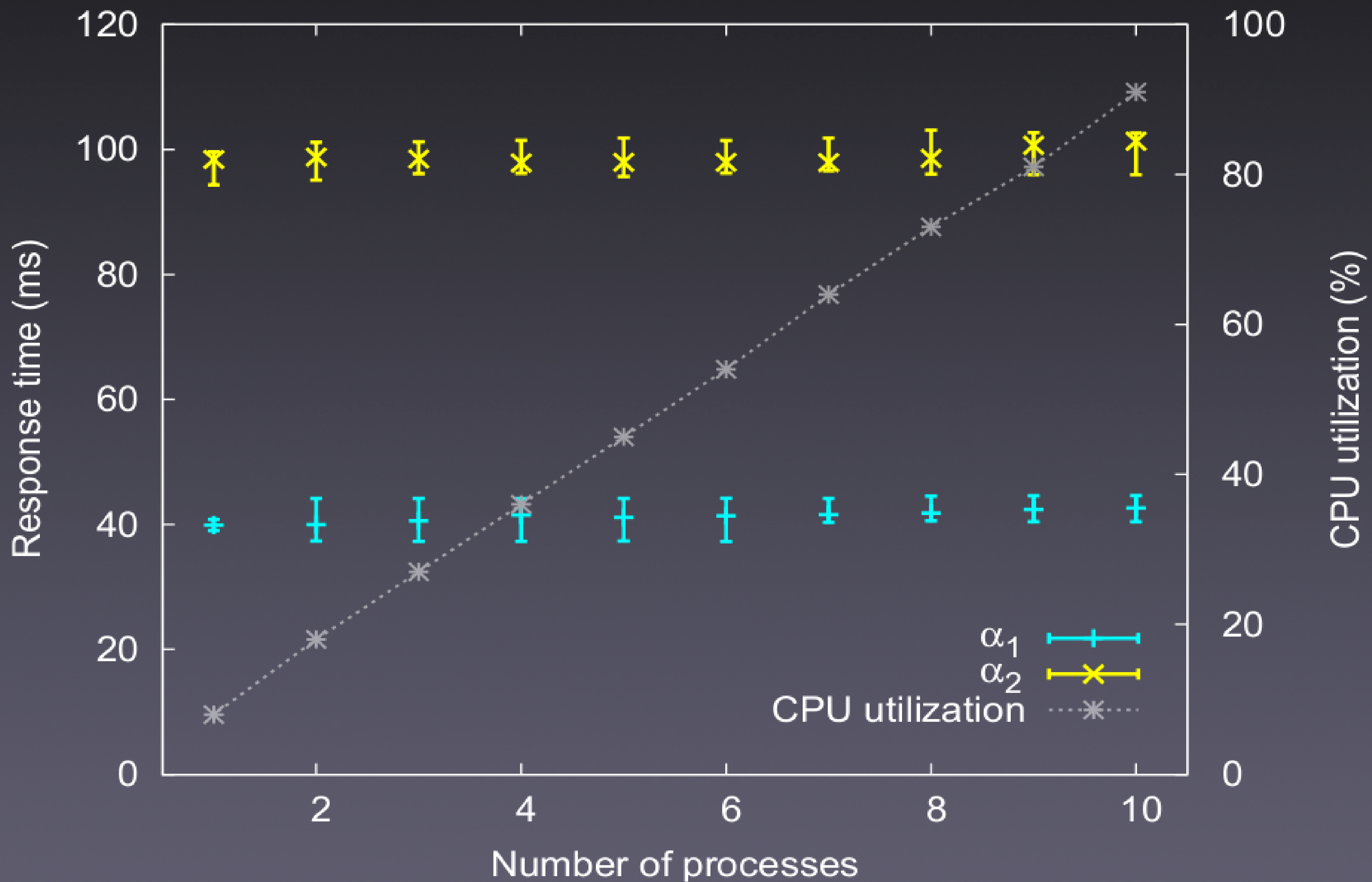
space complexity



750 processes

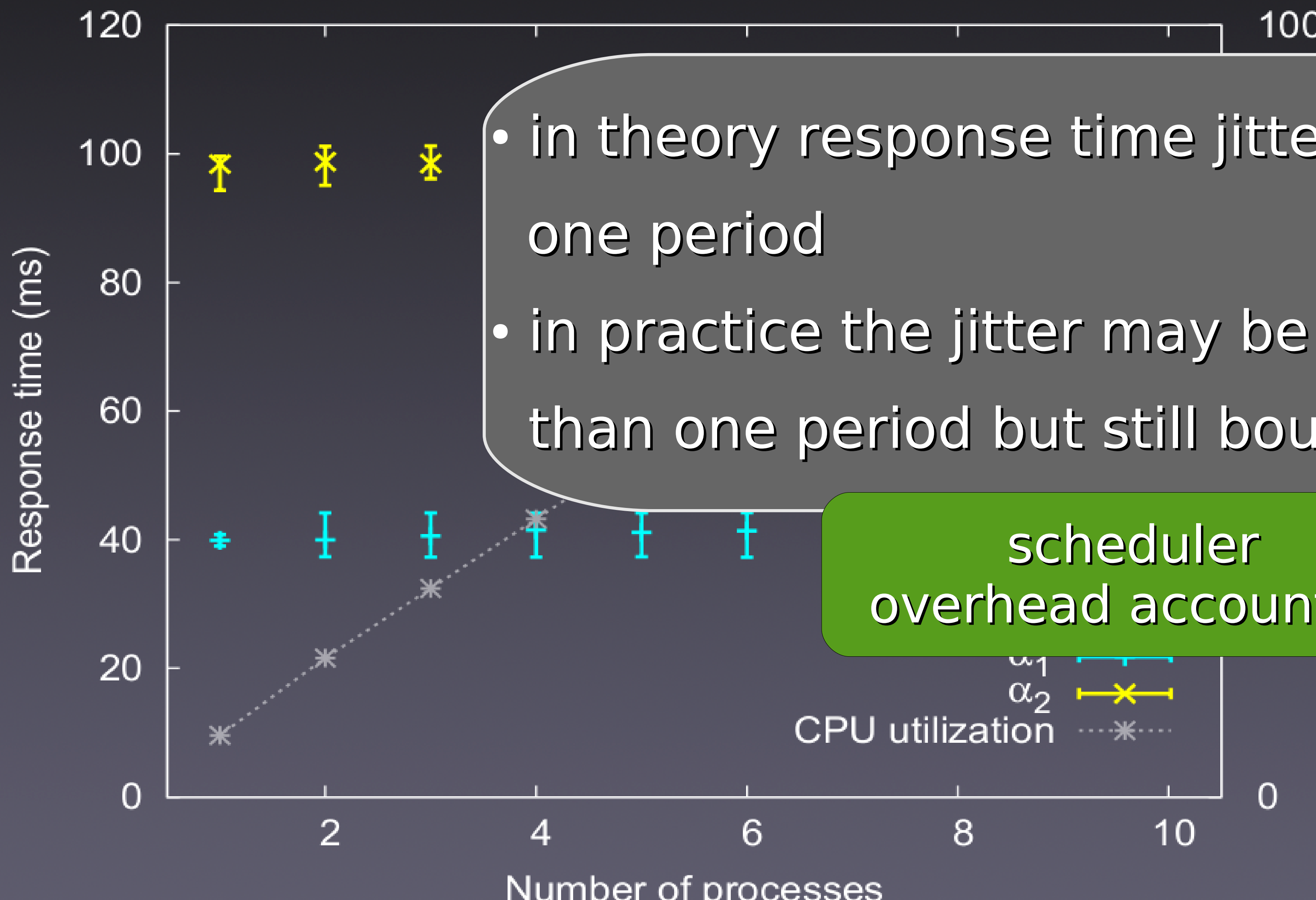
Results

bare-metal experiment



Results

bare-metal experiment



- in theory response time jitter is one period
- in practice the jitter may be more than one period but still bounded

scheduler overhead accounting

Conclusion

VBS scheduling enables:

- temporal isolation
- trading off throughput and latency
- controlling the response-time jitter of individual process actions
- trading off space and time complexity of the scheduling overhead

http://tiptoe.cs.uni-salzburg.at/

Conclusion

VBS scheduling enables:

- temporal isolation
- trading
- controll … of individu

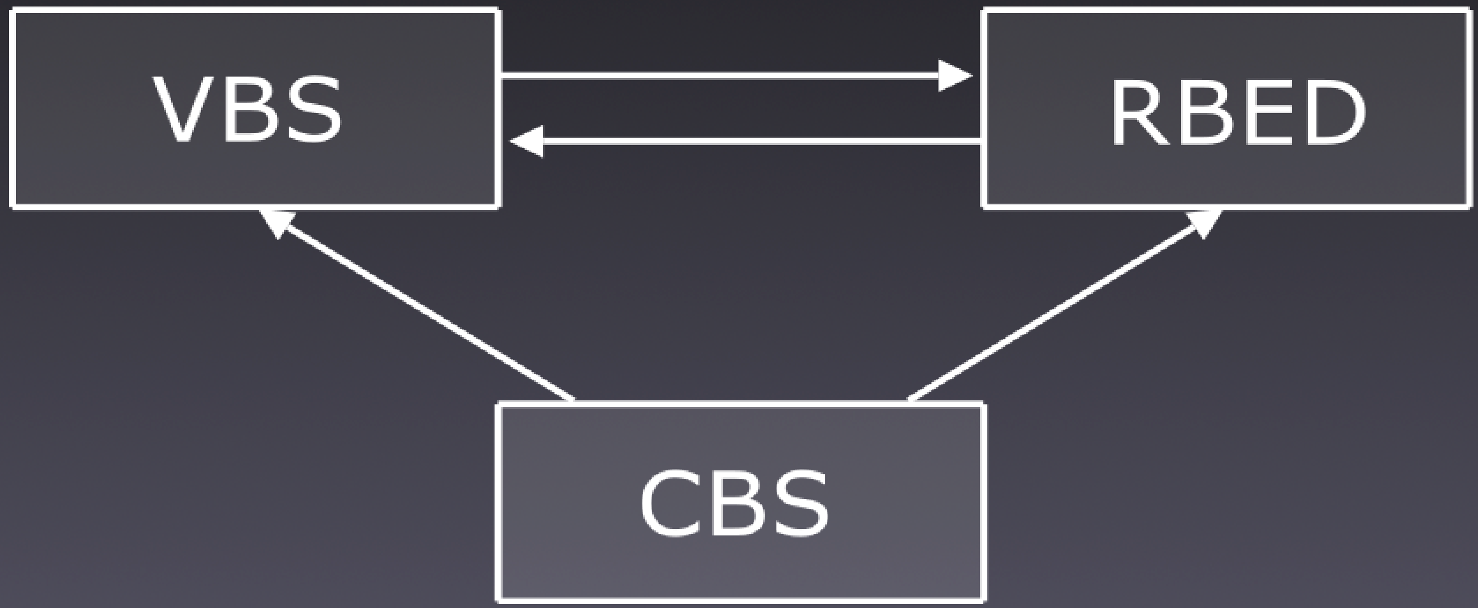
Thank you!

- trading off space and time complexity of the scheduling overhead

http://tiptoe.cs.uni-salzburg.at/

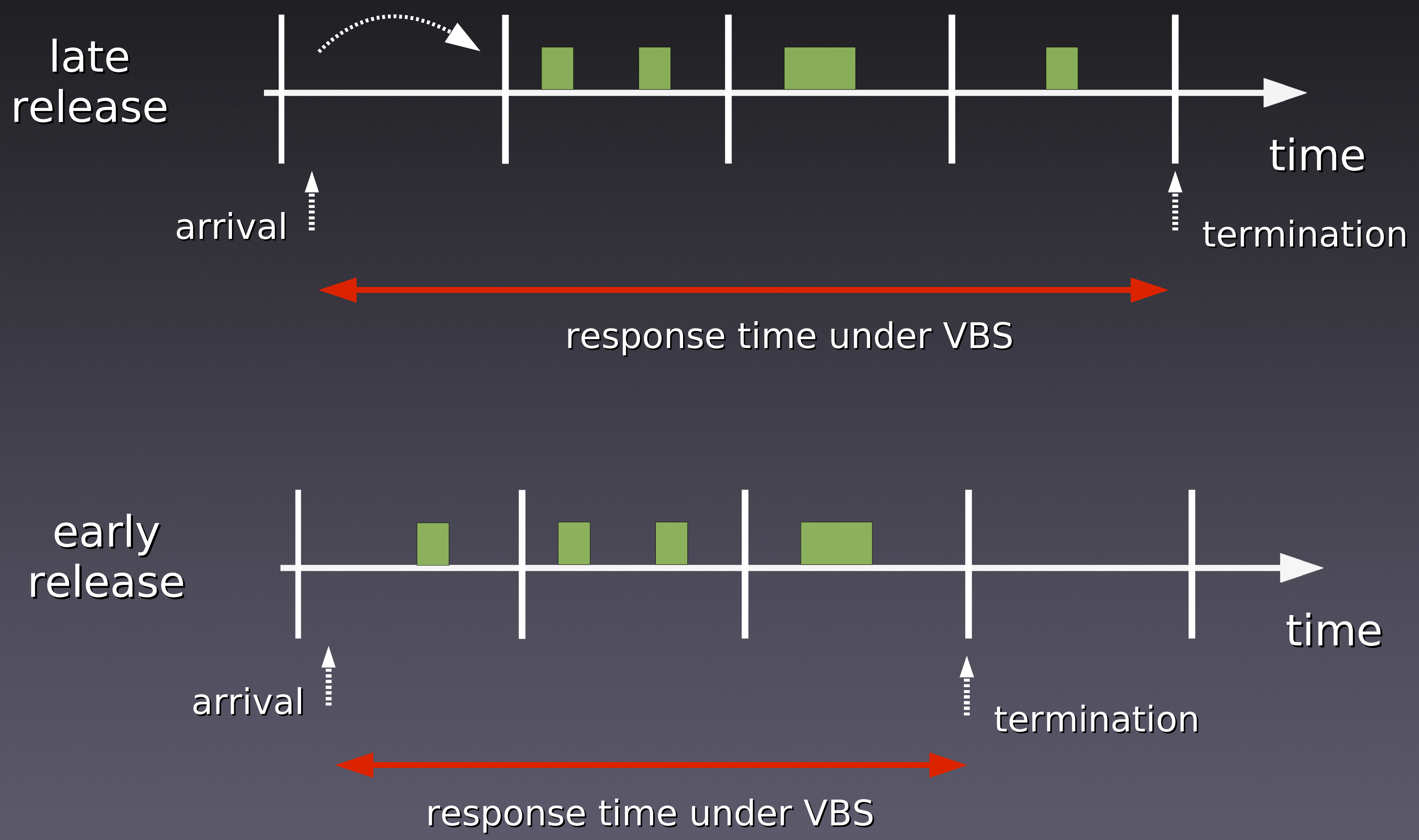
VBS

selected related work

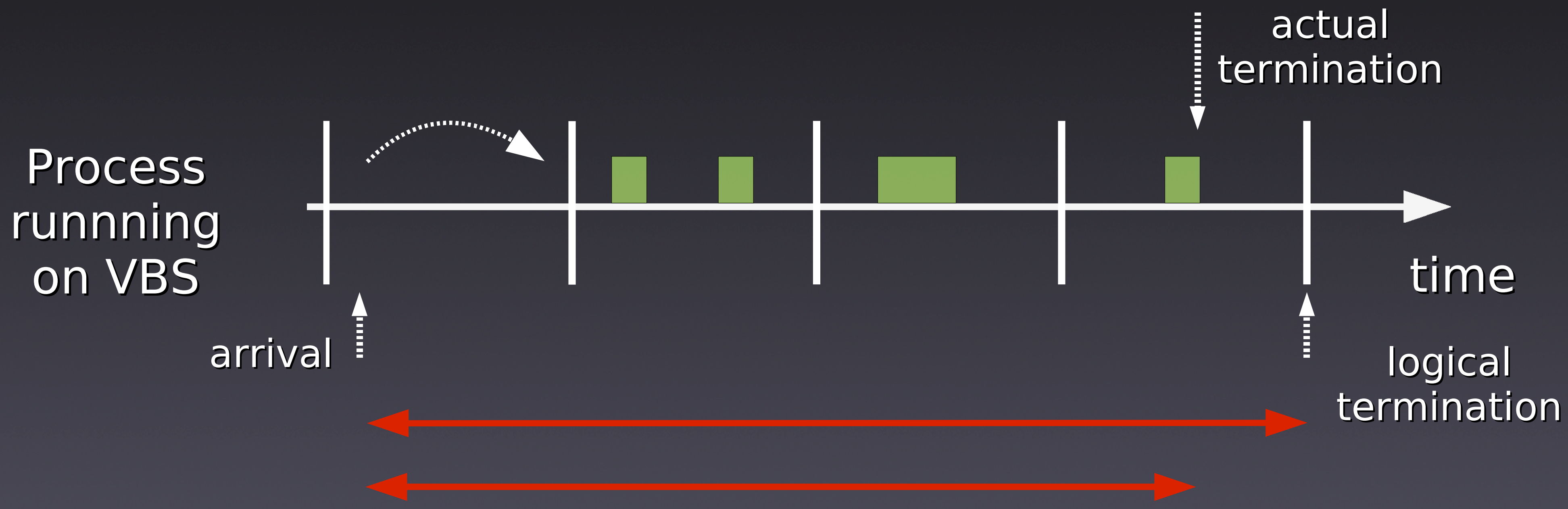


- CBS does not allow changing of the period and limit
- RBED and VBS differ on the level of abstractions provided

Release strategy



Logical response time jitter



jitter

$2(\pi - 1)$