Programmable Temporal Isolation for Real-Time Systems

Silviu Craciunas
Department of Computer Sciences
University of Salzburg

joint work with Christoph Kirsch and Ana Sokolova

Structure

Structure

VBS process model

varying real-time constraints while maintaining temporal isolation

Structure

VBS process model

varying real-time constraints while maintaining temporal isolation

VBS scheduling result

EDF-based scheduler, response time bounds, queue mechanism

Structure

VBS process model

varying real-time constraints while maintaining temporal isolation

VBS scheduling result

EDF-based scheduler, response time bounds, queue mechanism

Overhead accounting

include scheduler overhead in the schedulability analysis

Structure

VBS process model

varying real-time constraints while maintaining temporal isolation

VBS scheduling result

EDF-based scheduler, response time bounds, queue mechanism

Overhead accounting

include scheduler overhead in the schedulability analysis

Power-aware VBS

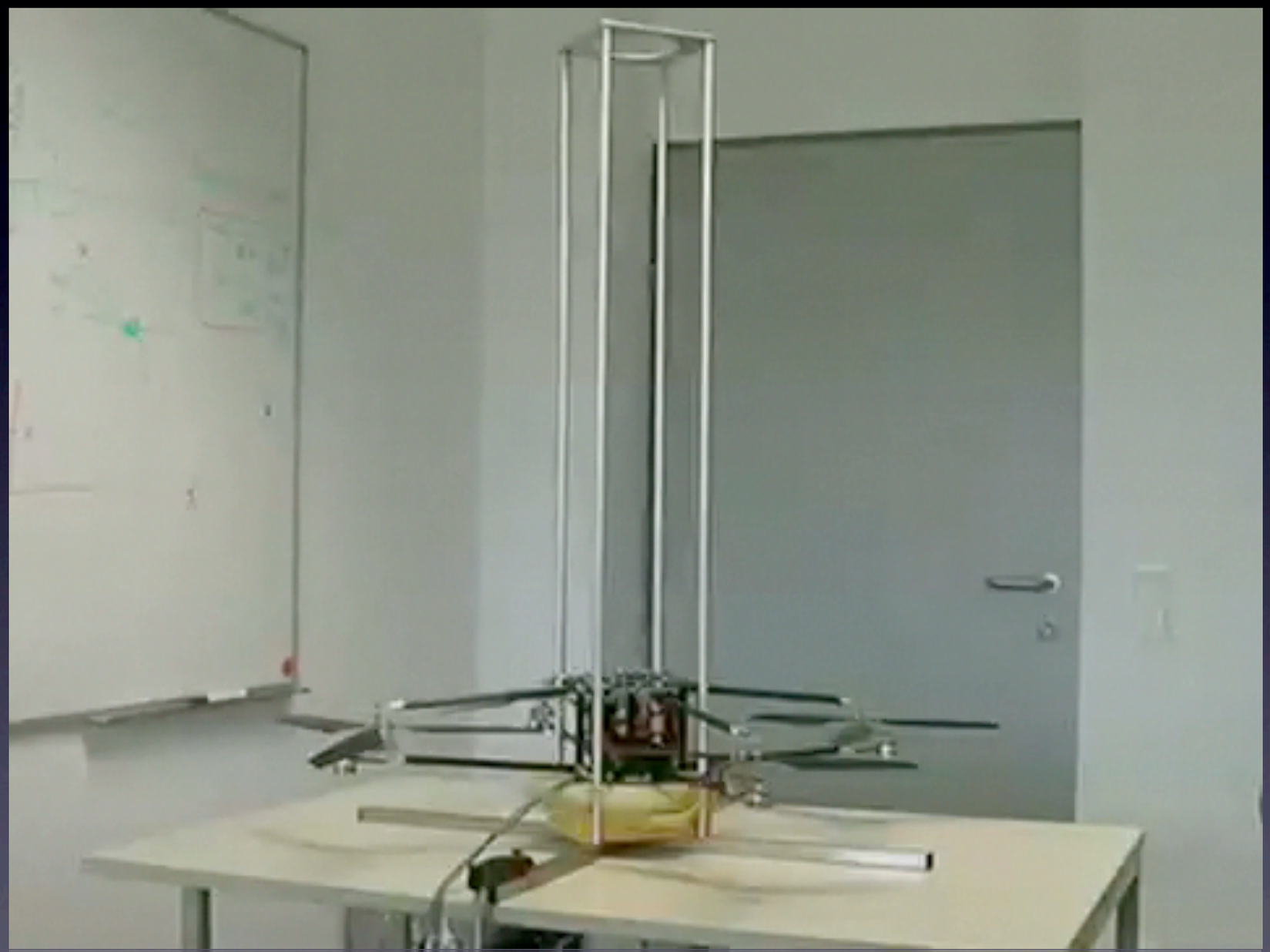
reduce CPU power consumption

Motivation

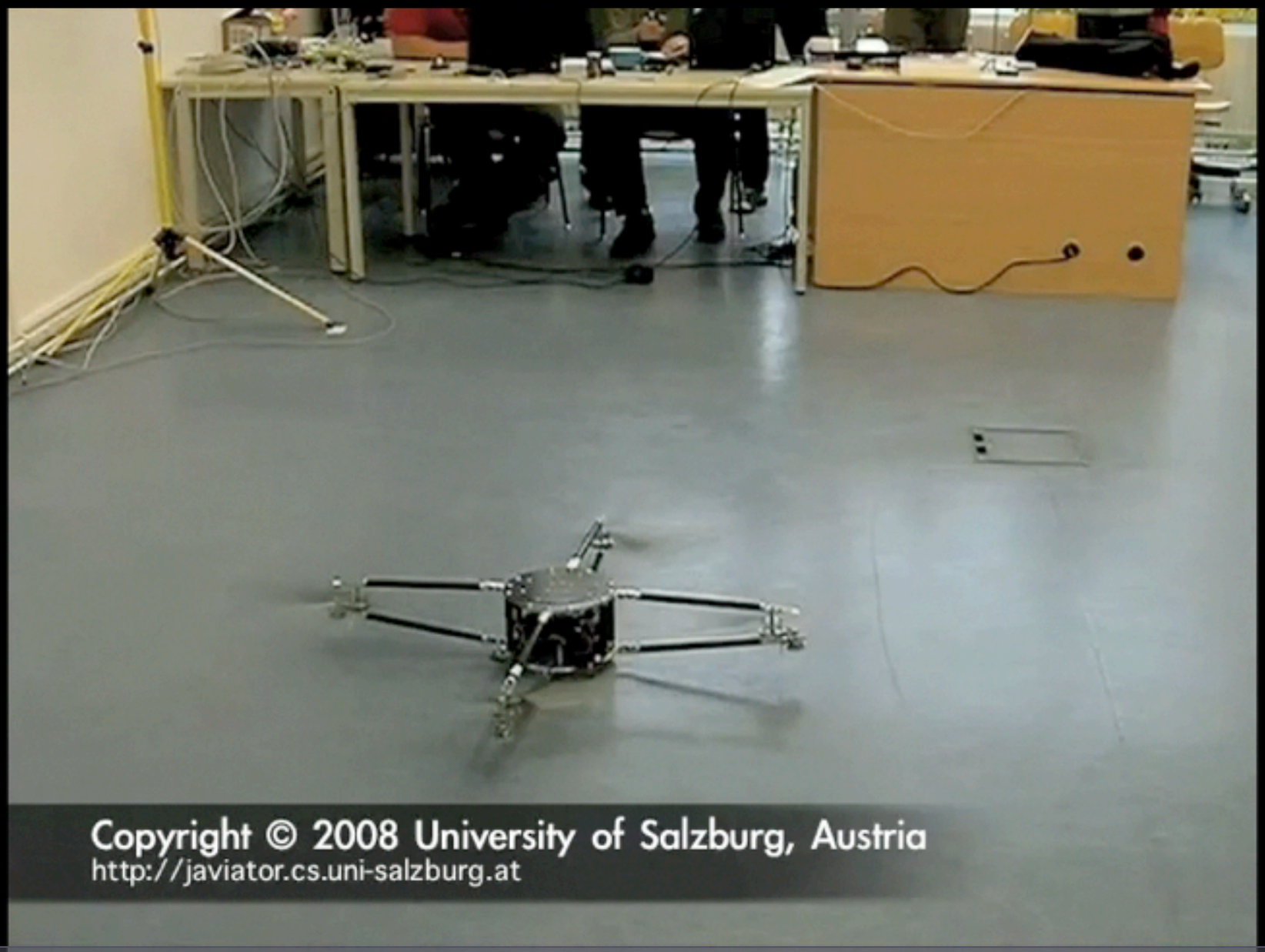
Motivation

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

When things go wrong...



Flying is difficult but fun



Real-time control loop

```
loop {
    read_sensors();
    compute_actuators();
    write_actuators();
    update_state();
    log();
}
```

Real-time control loop

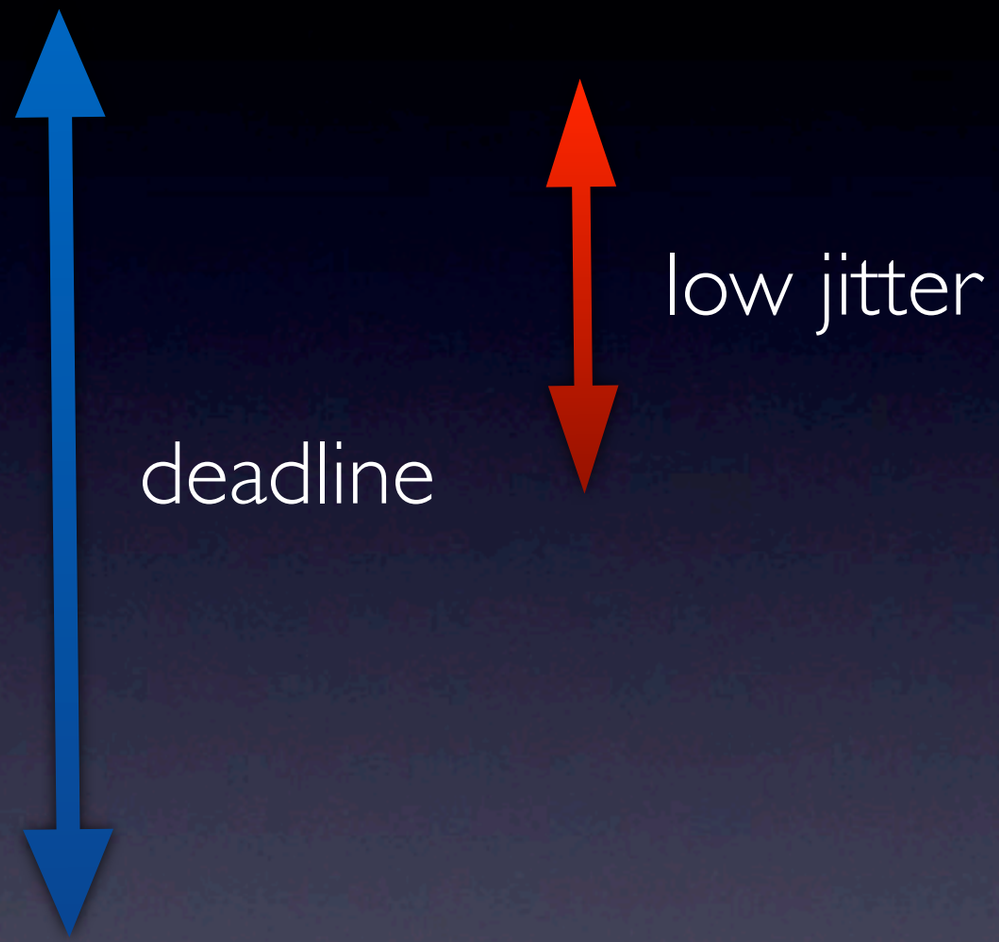
```
loop {
    read_sensors();
    compute_actuators();
    write_actuators();

    update_state();
    log();
}
```

Real-time control loop

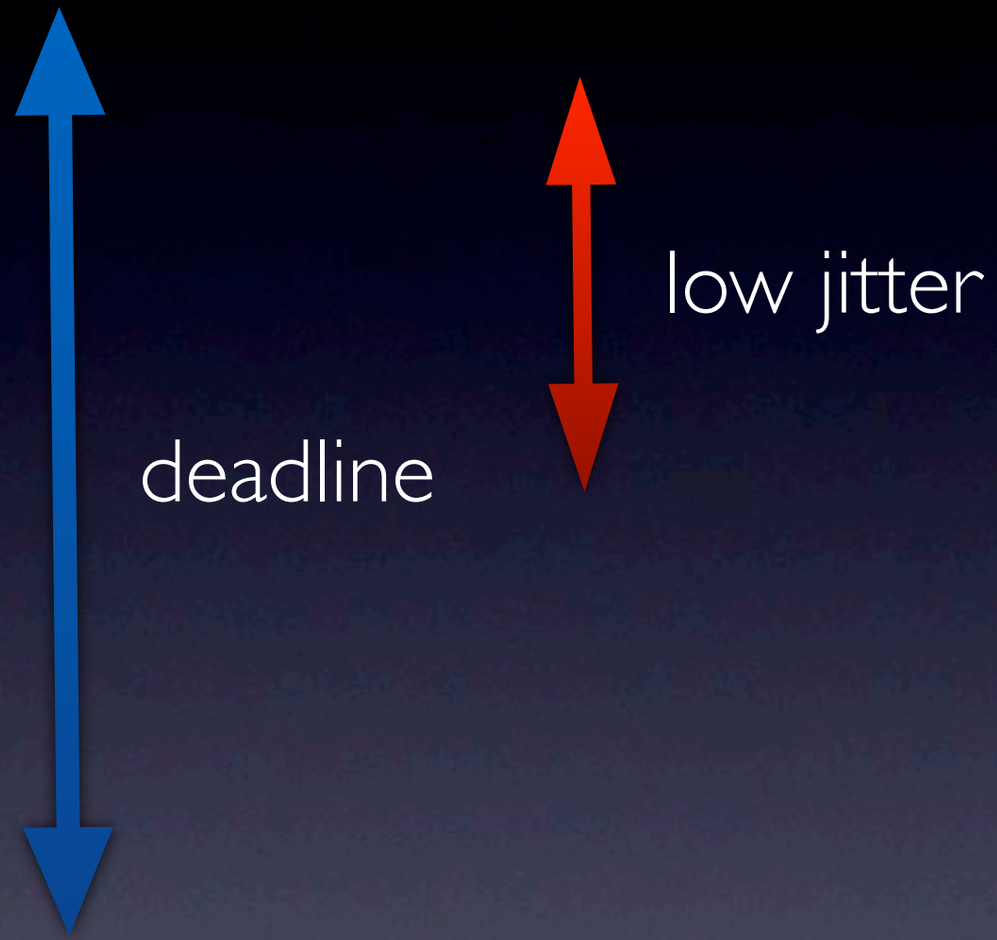
```
loop {
    read_sensors();
    compute_actuators();
    write_actuators();

    update_state();
    log();
}
```



Real-time control loop

```
loop {
    read_sensors();
    compute_actuators();
    write_actuators();
    update_state();
    log();
}
```



- different portions of code have different timing requirements
- temporal behavior should not be affected by other processes (temporal isolation)

Process model

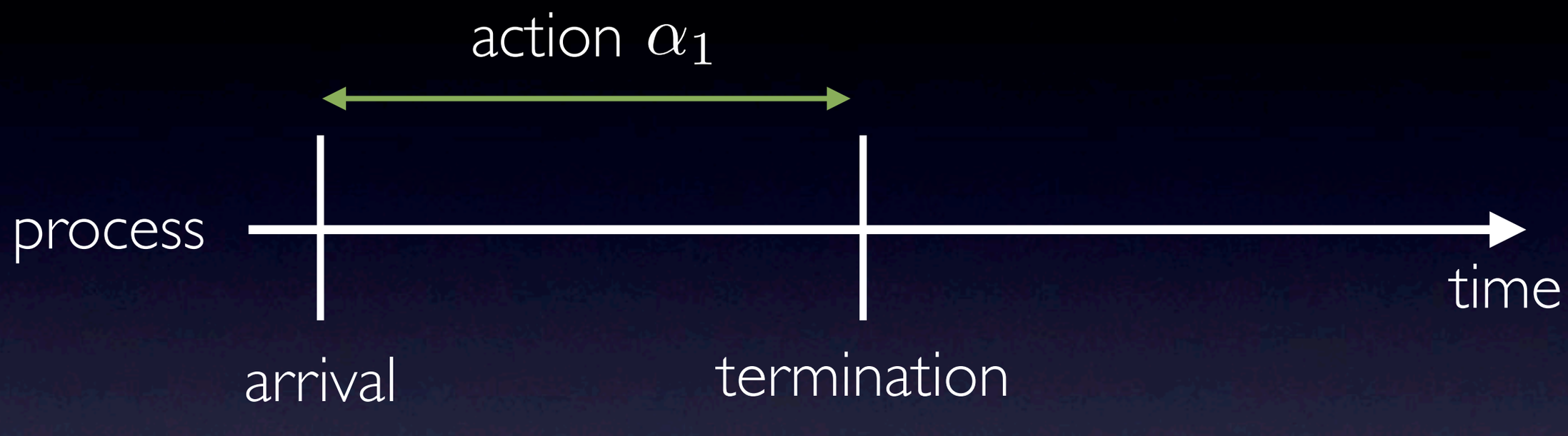
process

time

Process model

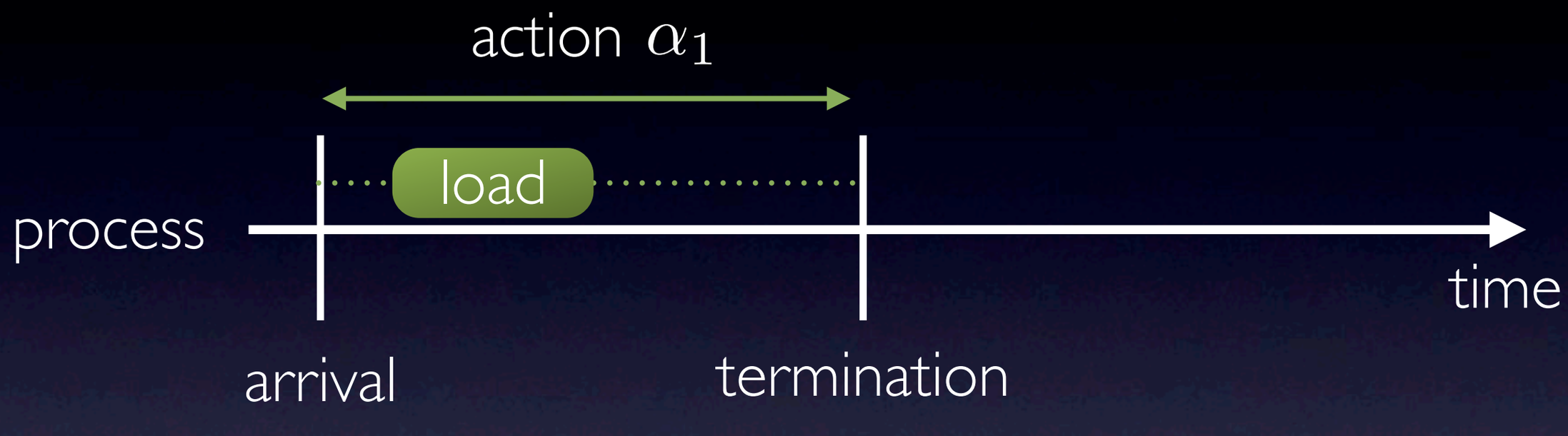
Process model

Process model



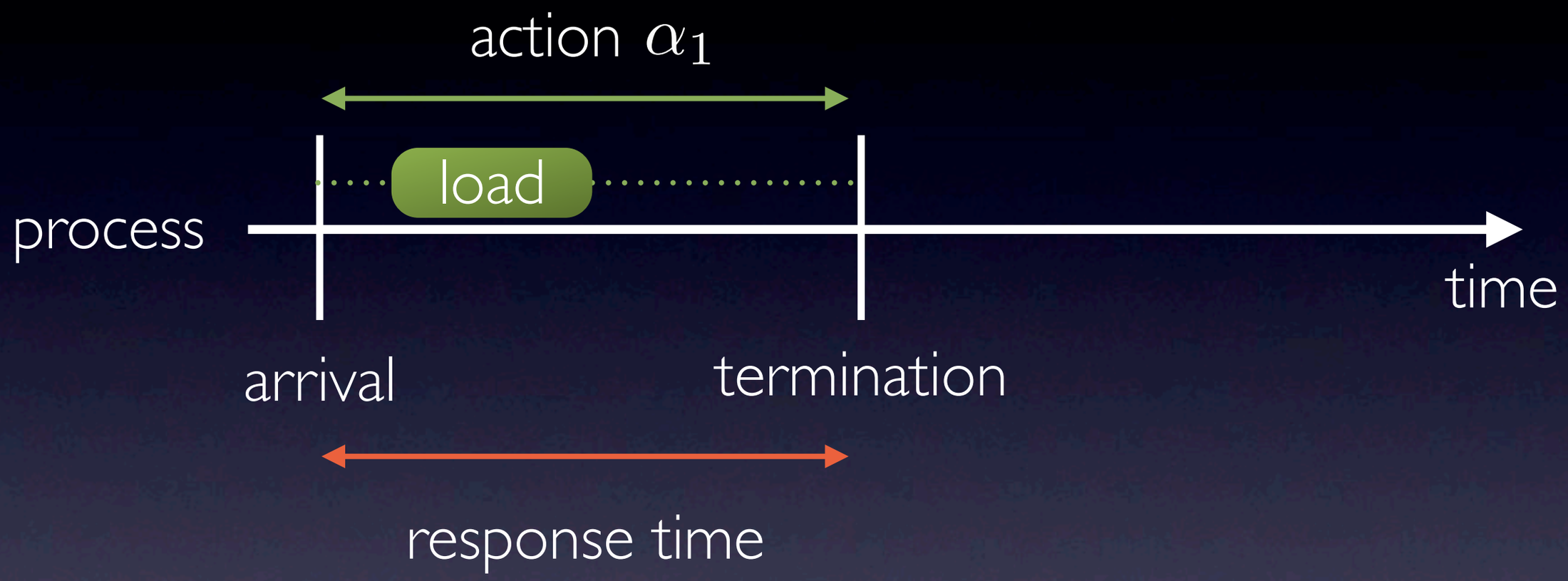
- action is a piece of code

Process model



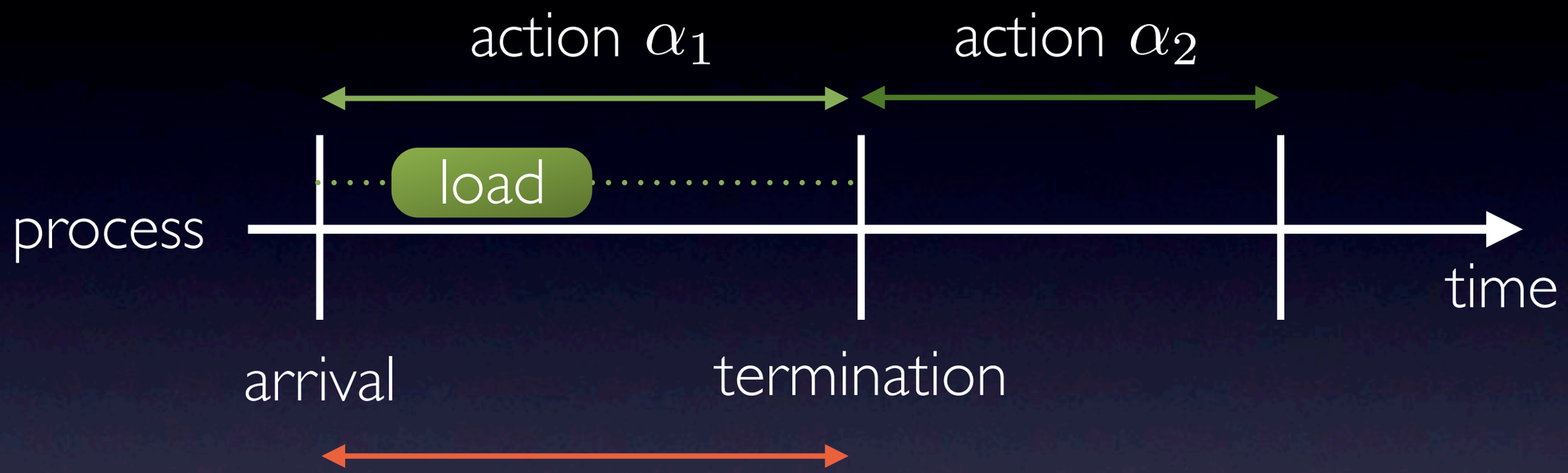
- action is a piece of code

Process model



- action is a piece of code

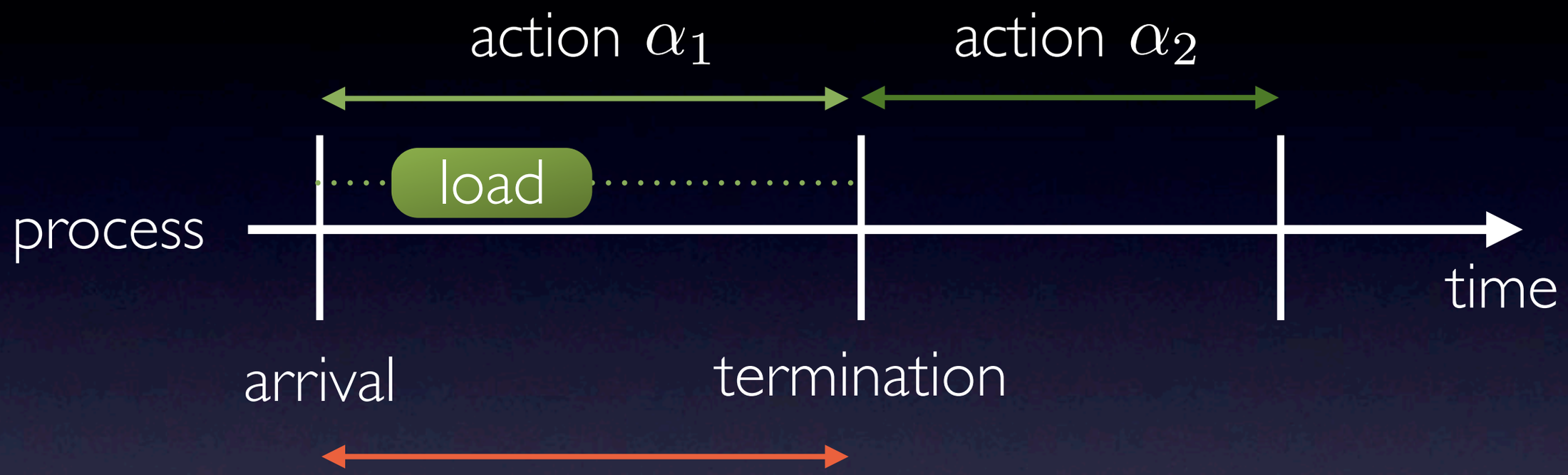
Process model



response time

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

Process model



- action is a piece of code
- process is a sequence of actions
- throughput vs latency of process execution

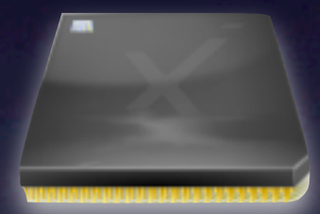
Scheduling problem

process 1

process 2

process n

uniprocessor



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

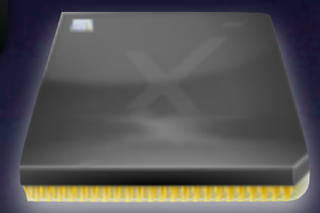
Scheduling problem

process 1

process 2

Solvable with variable-bandwidth servers (VBS)

uniprocessor



process n

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

Scheduling problem

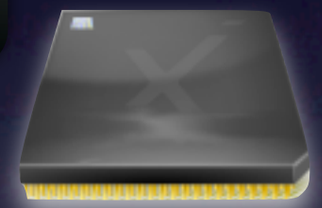
process 1

process 2

process n

Solvable with variable-bandwidth servers (VBS)

uniprocessor



Results:

- constant-time scheduling algorithm
- constant time admission test

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

Resources and VBS

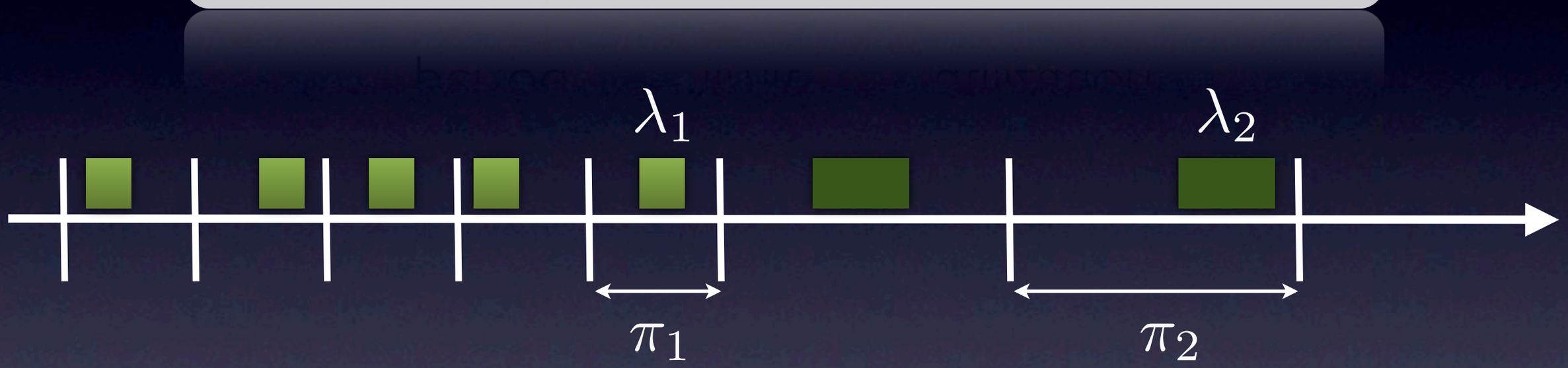
virtual periodic resources

period π limit λ utilization $\frac{\lambda}{\pi}$

Resources and VBS

virtual periodic resources

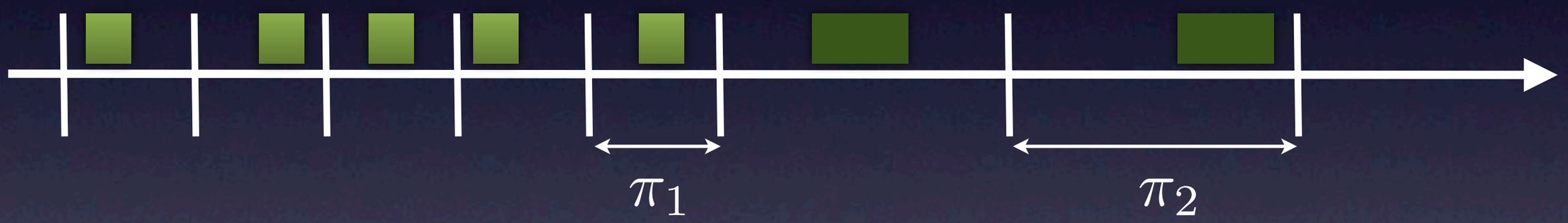
period π limit λ utilization $\frac{\lambda}{\pi}$



Resources and VBS

virtual periodic resources

period π limit λ utilization $\frac{\lambda}{\pi}$



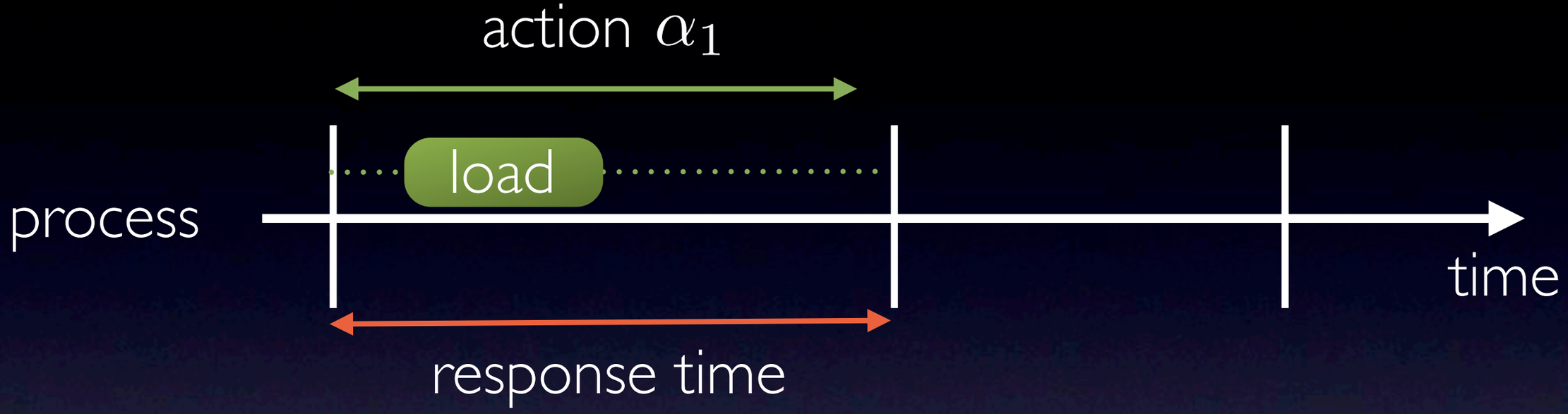
- VBS is determined by a bandwidth cap (u)
- VBS processes dynamically adjust speed (change resources)

$$\frac{\lambda_1}{\pi_1} \leq u \qquad \frac{\lambda_2}{\pi_2} \leq u$$

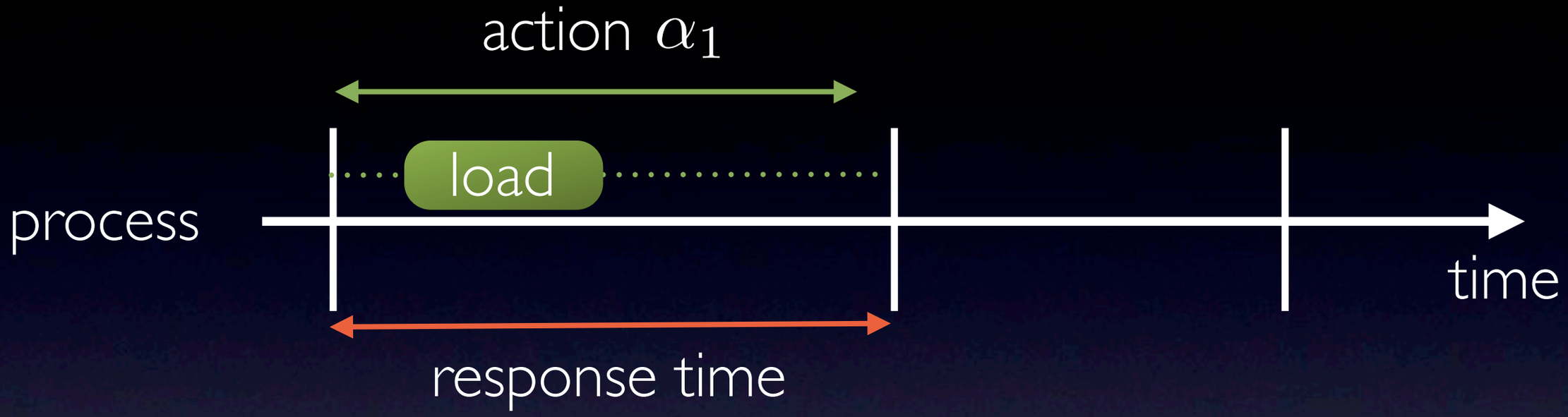
- generalization of constant bandwidth servers (CBS)

[Abeni and Buttazzo 2004]

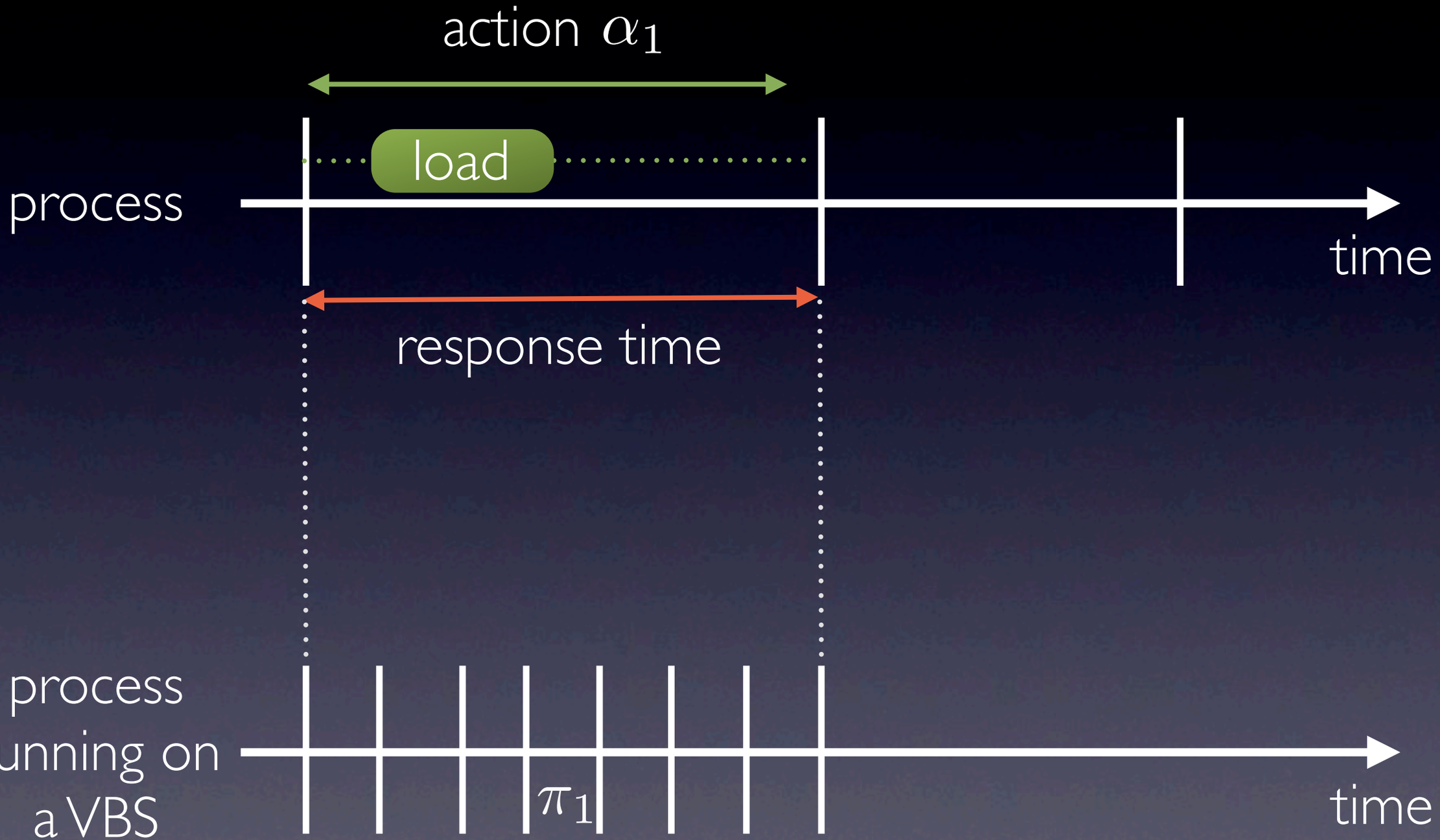
One process on a VBS



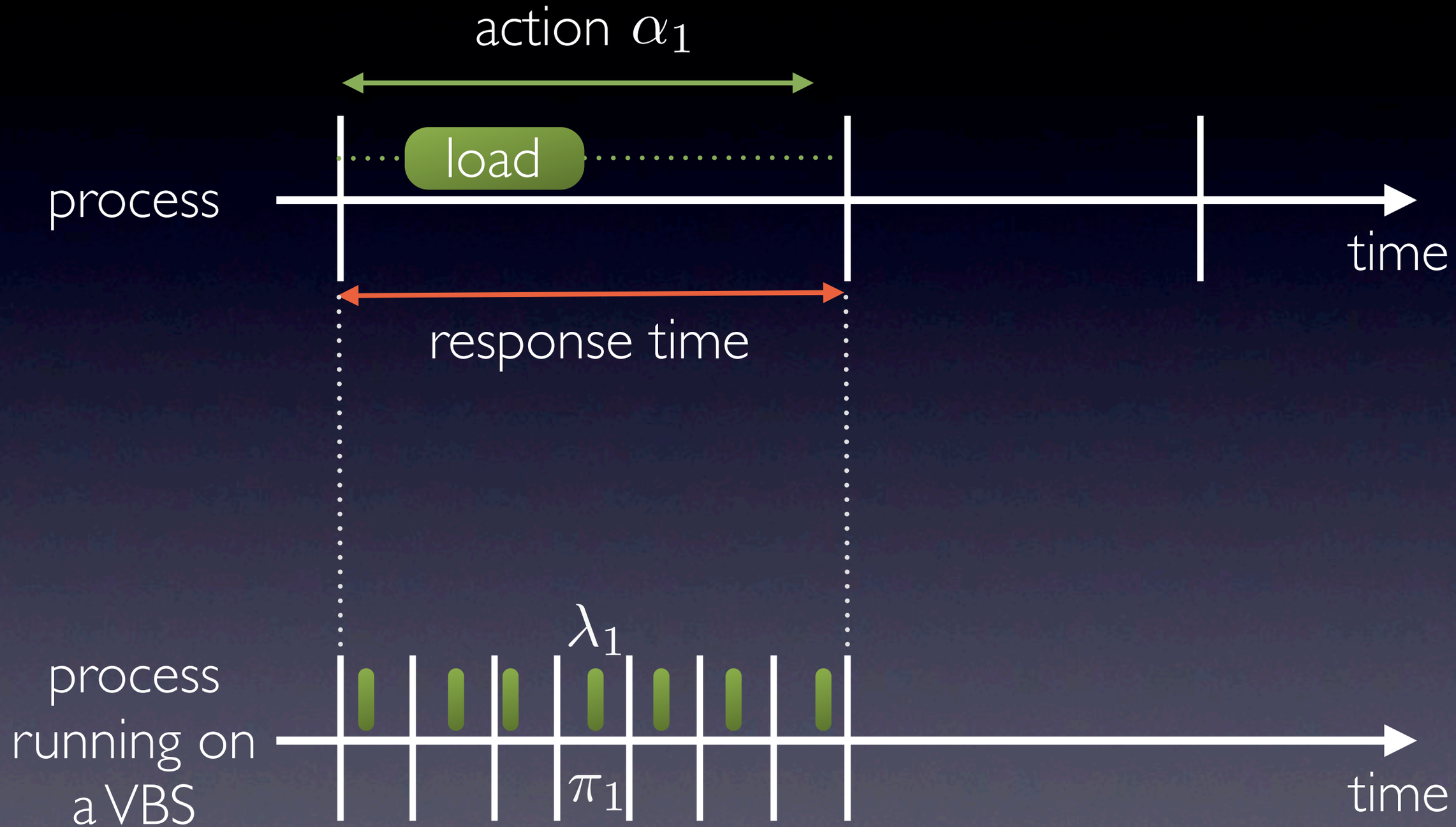
One process on a VBS



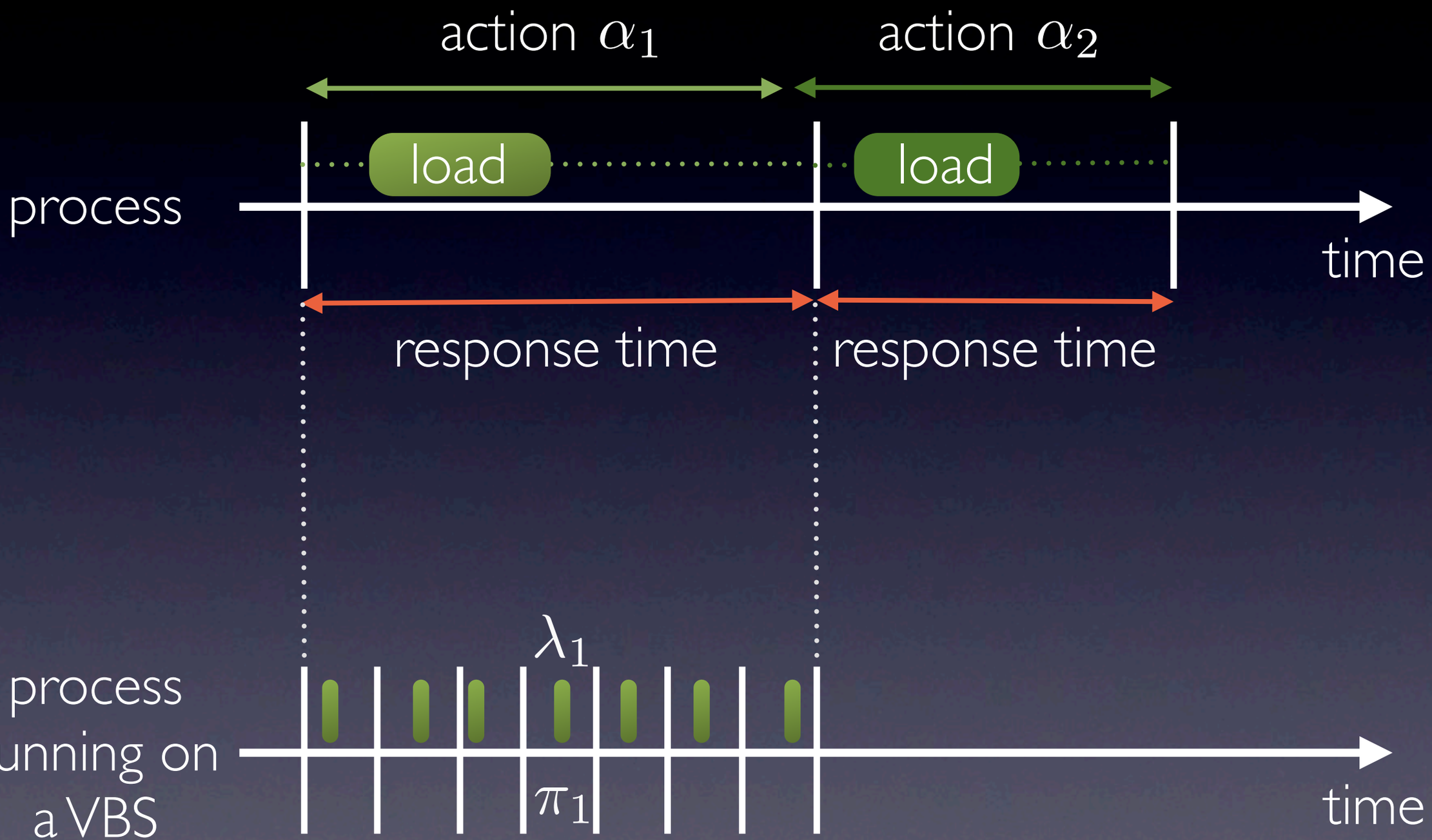
One process on a VBS



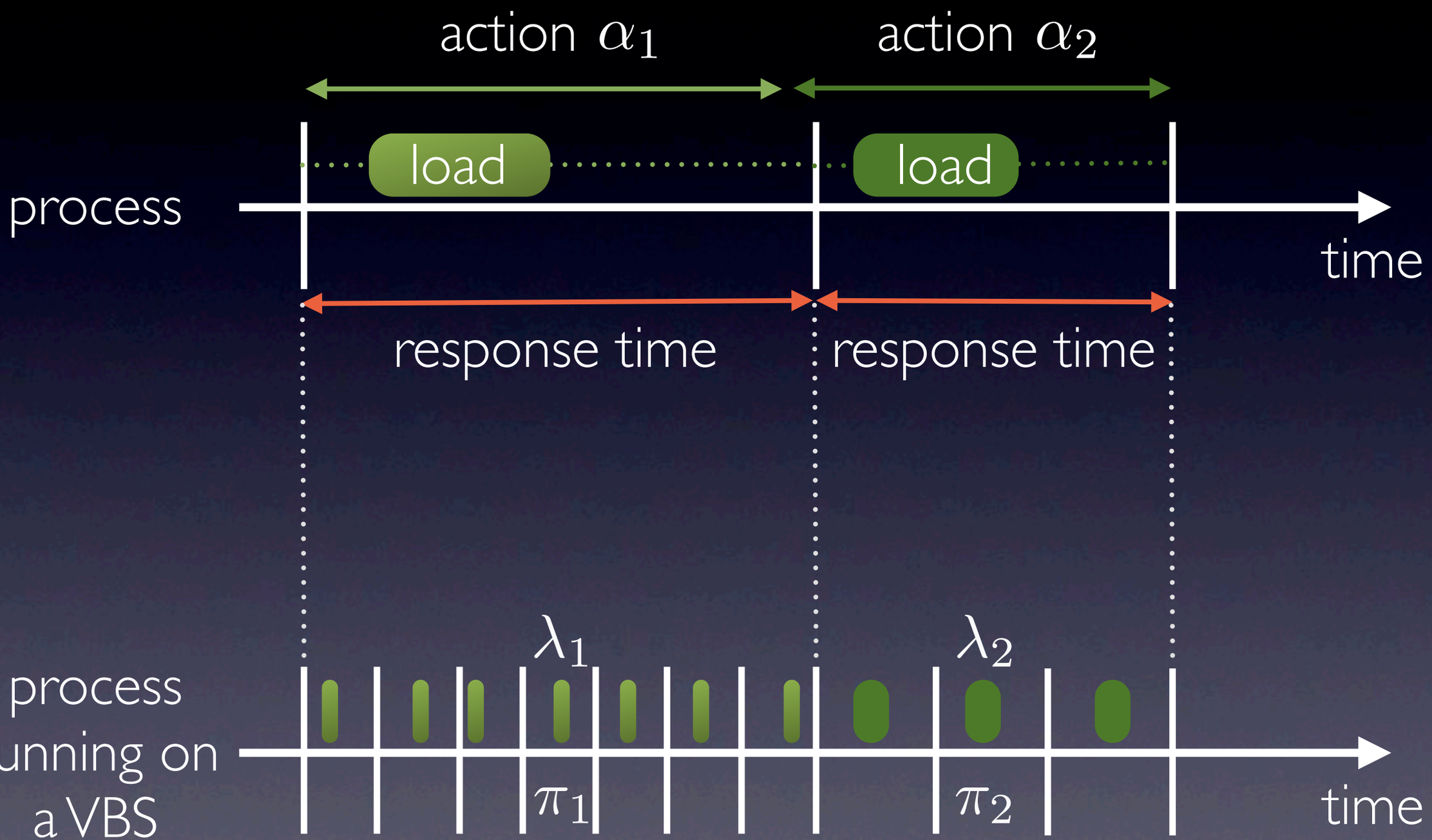
One process on a VBS



One process on a VBS



One process on a VBS

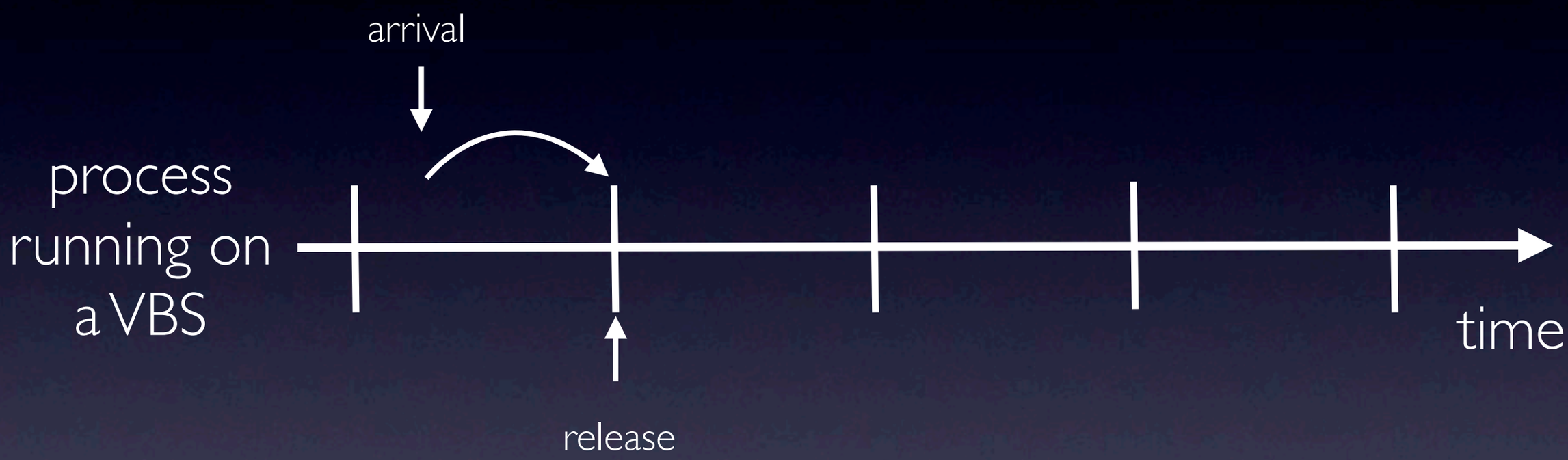


VBS

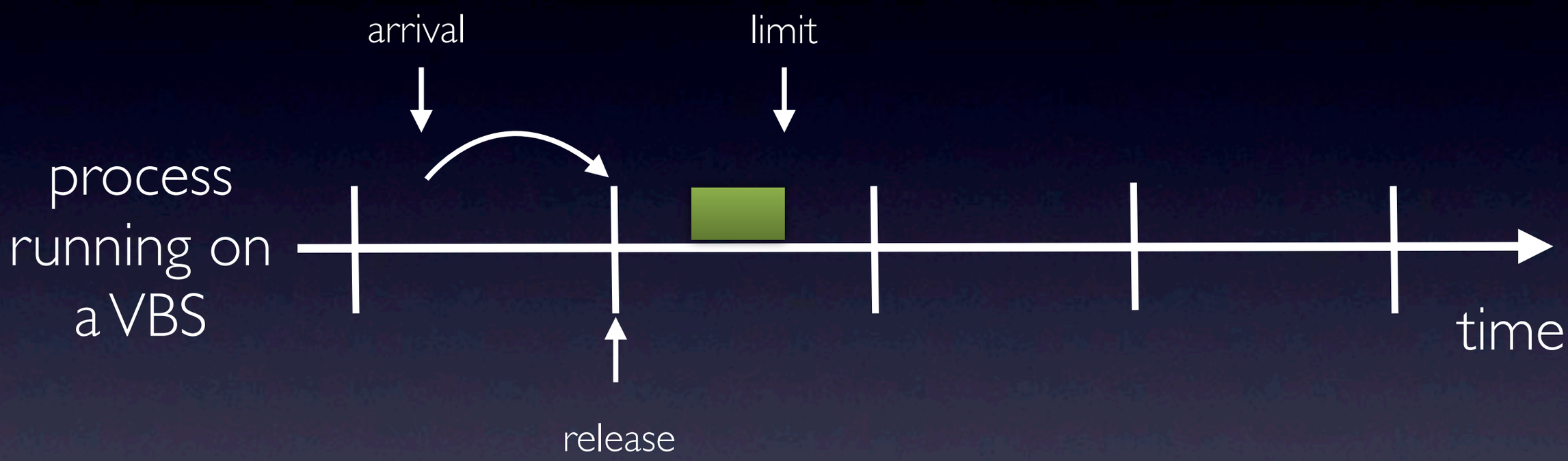
process running on a VBS

VBS

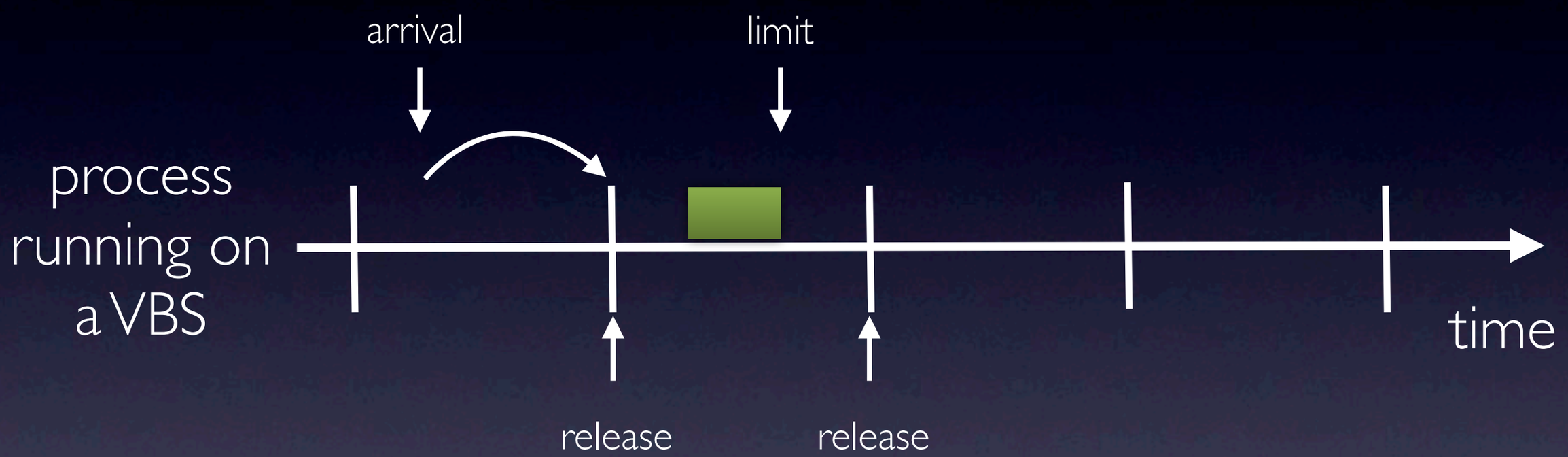
VBS



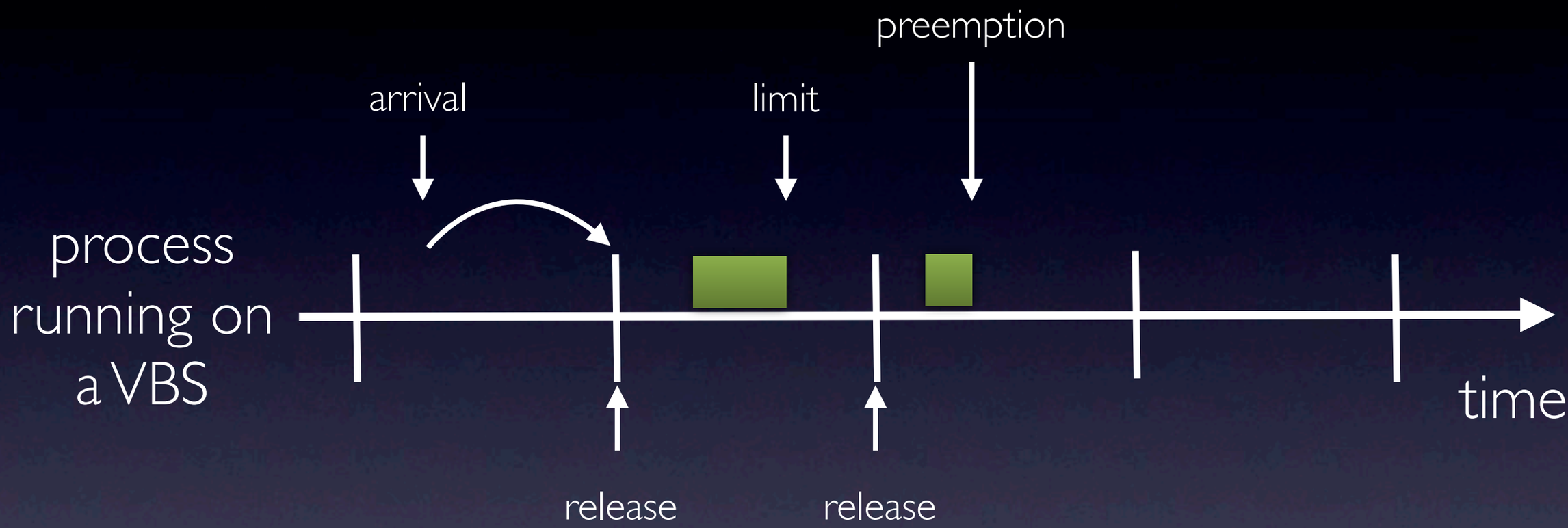
VBS



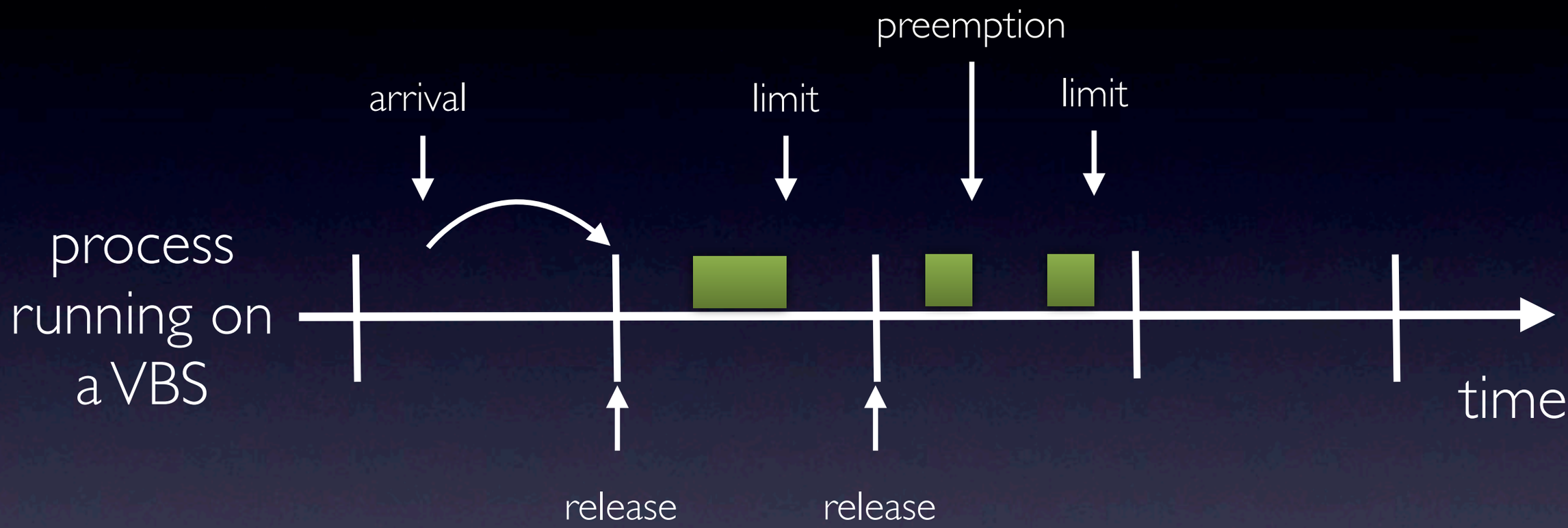
VBS


arrival
limit
process running on a VBS
time
release
release

VBS


preemption
arrival
limit
process running on a VBS
time
release
release

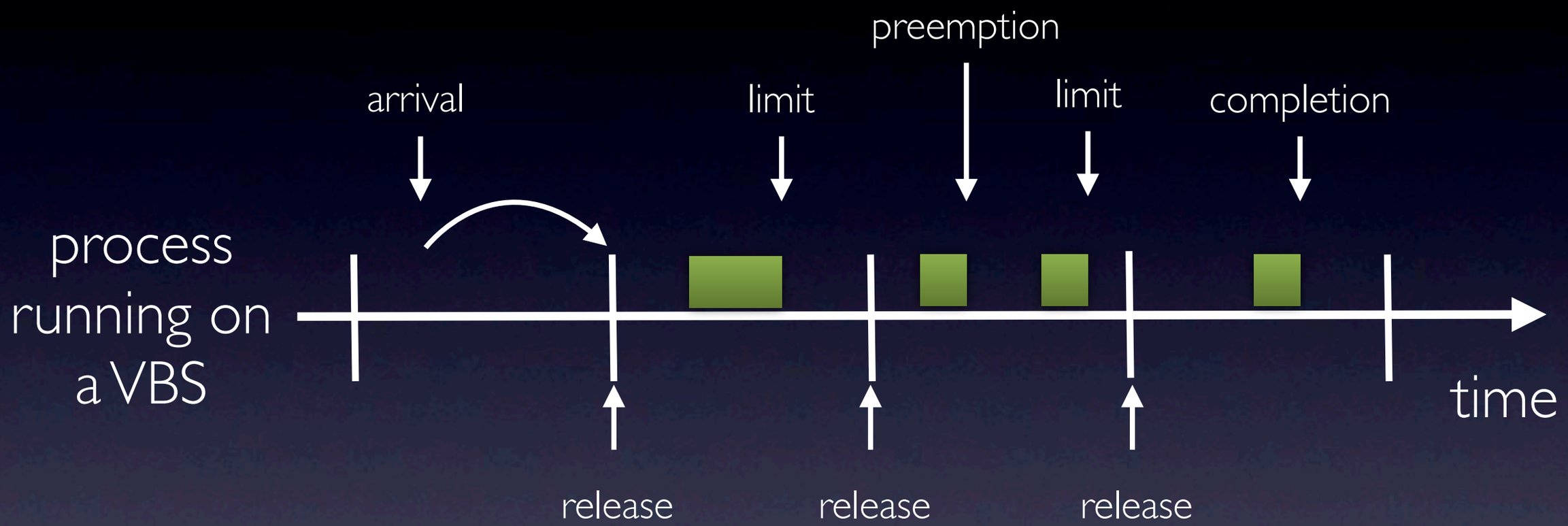
VBS


preemption
arrival
limit
limit
process running on a VBS
time
release
release

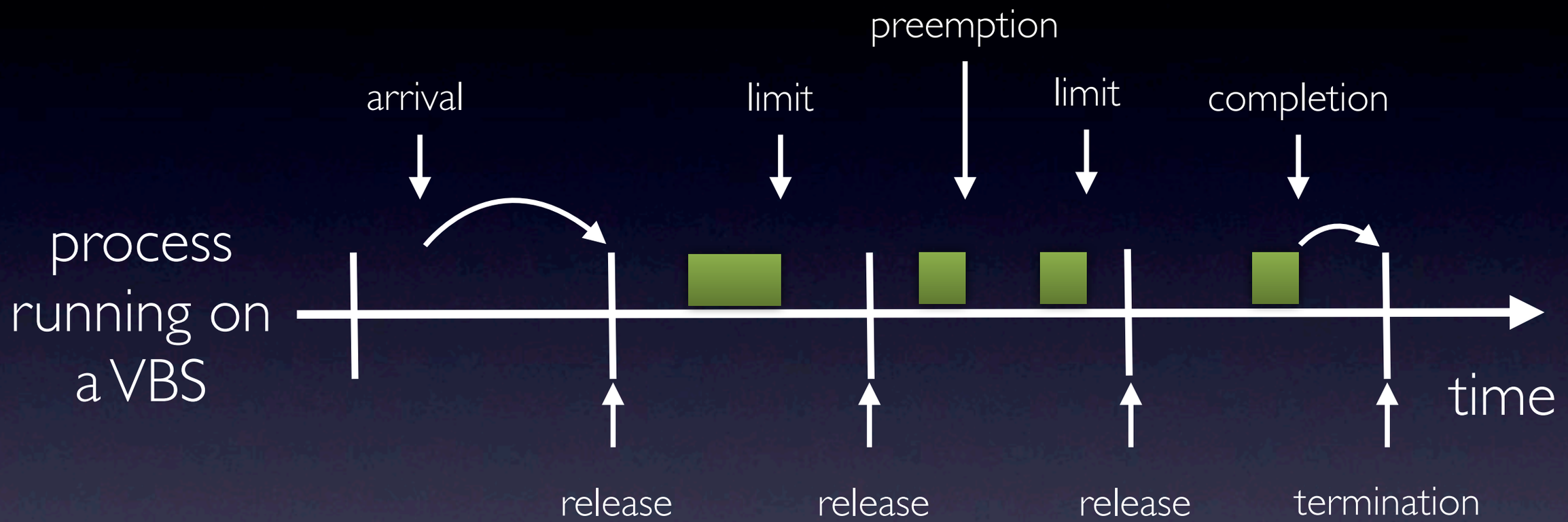
VBS


preemption
arrival
limit
limit
process running on a VBS
time
release
release
release

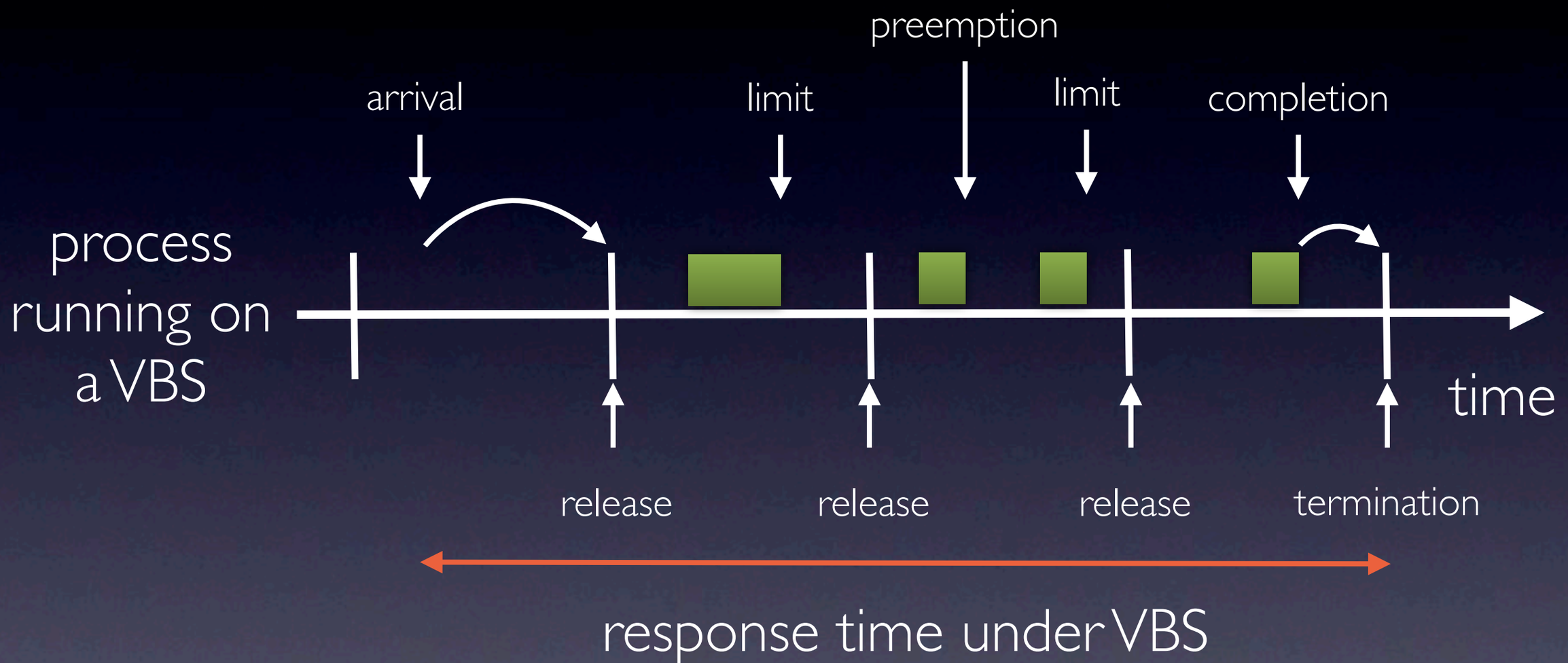
VBS


preemption
arrival
limit
limit
completion
process running on a VBS
time
release
release
release

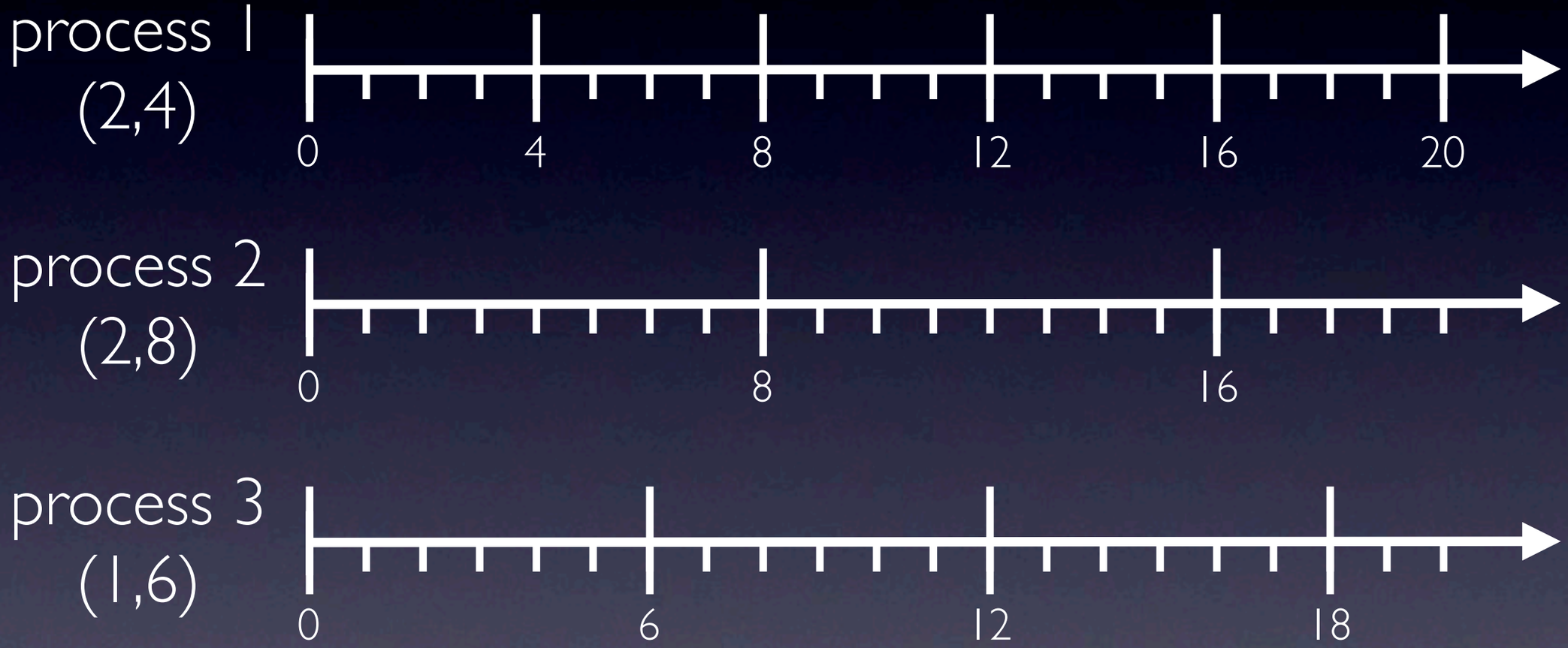
VBS


preemption
arrival
limit
limit
completion
process running on a VBS
time
release
release
release
termination

VBS

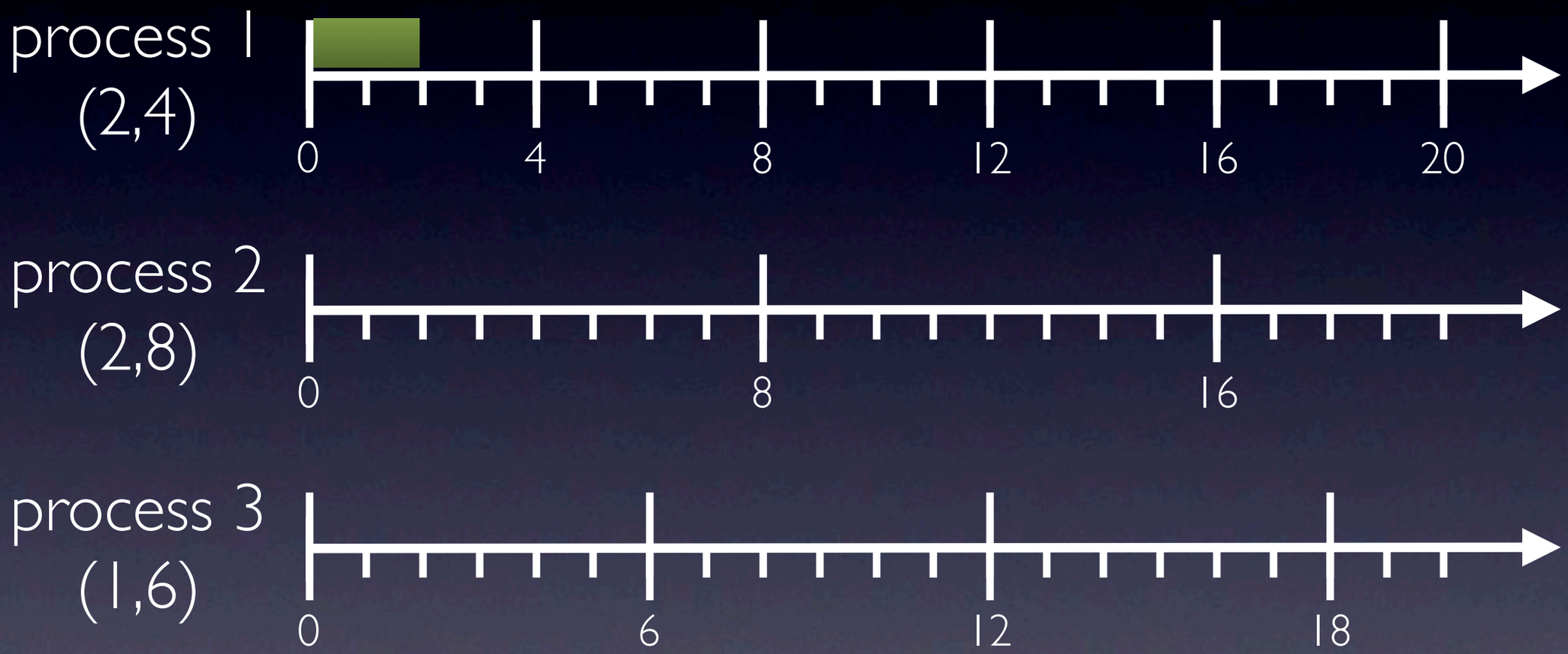

preemption
arrival
limit
limit
completion
process running on a VBS
time
release
release
release
termination
response time under VBS

VBS



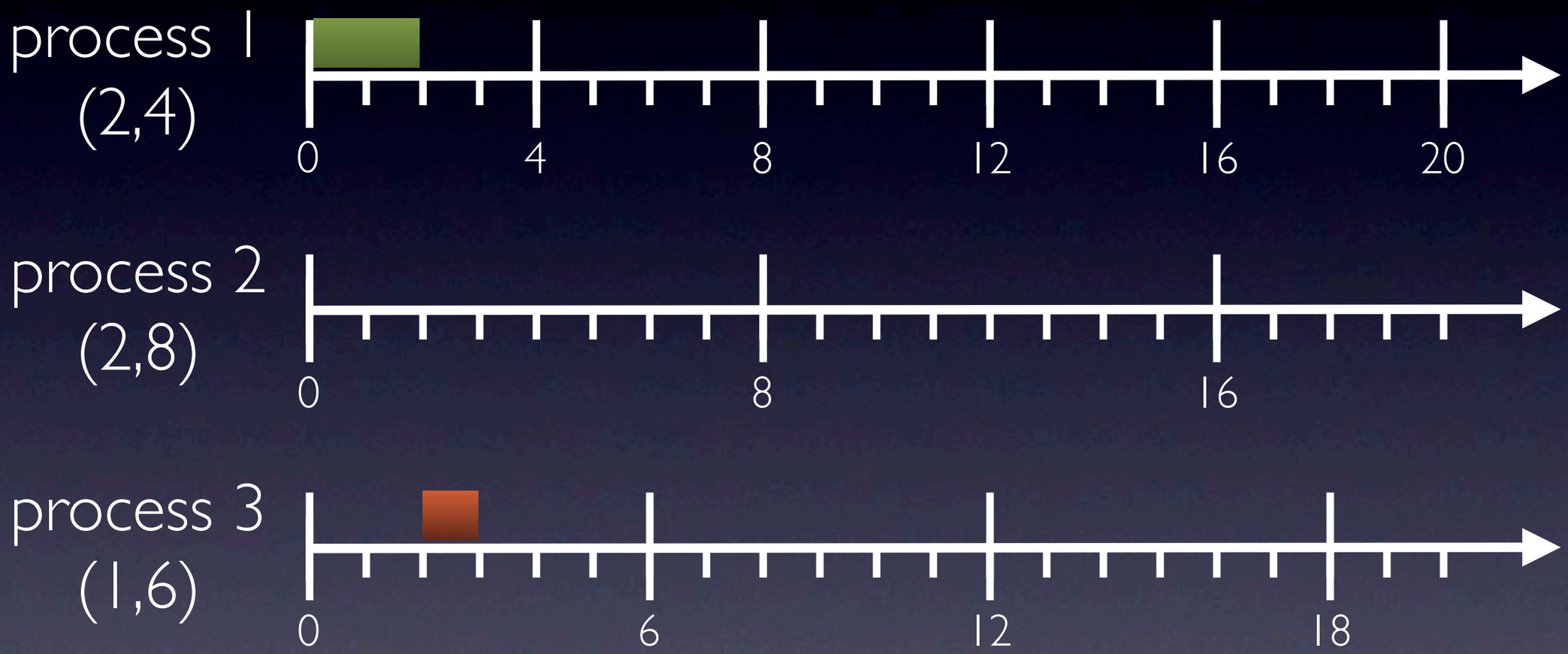
multiple processes are EDF-scheduled

VBS



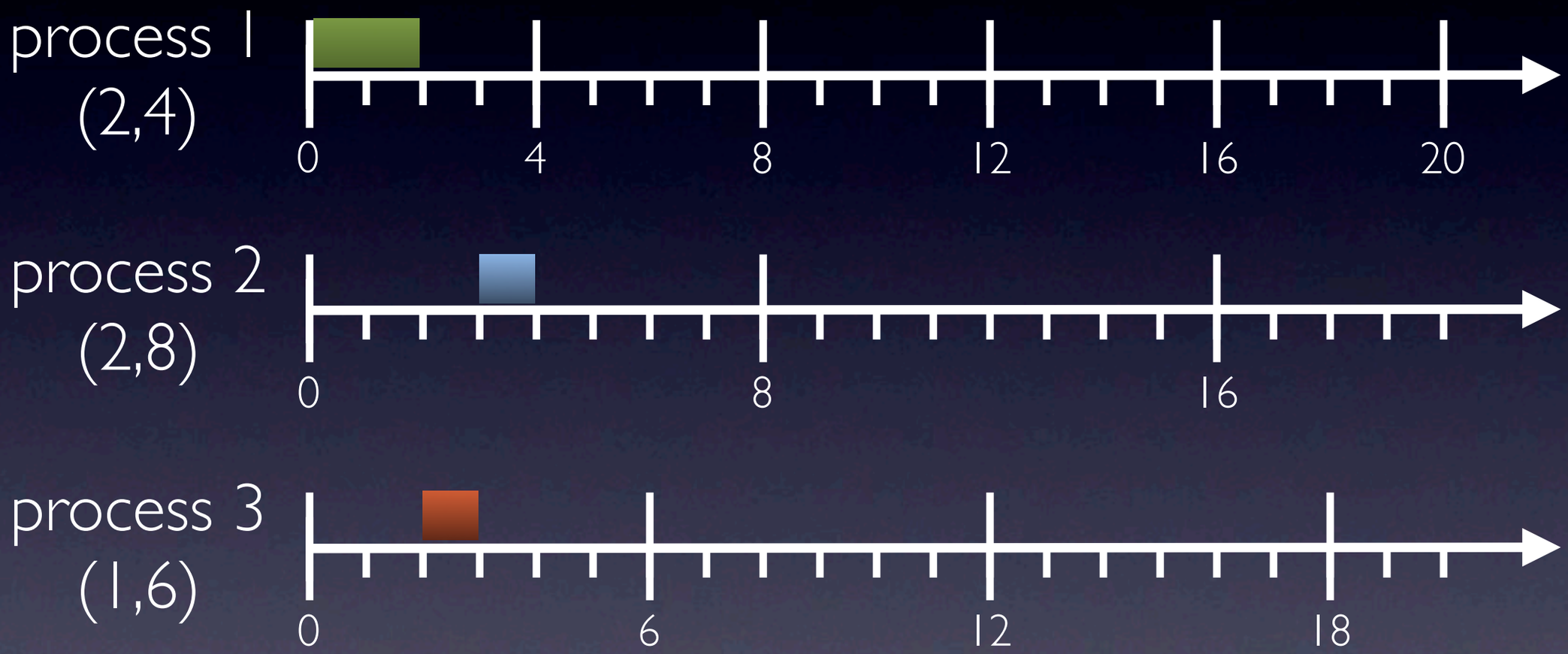
multiple processes are EDF-scheduled

VBS



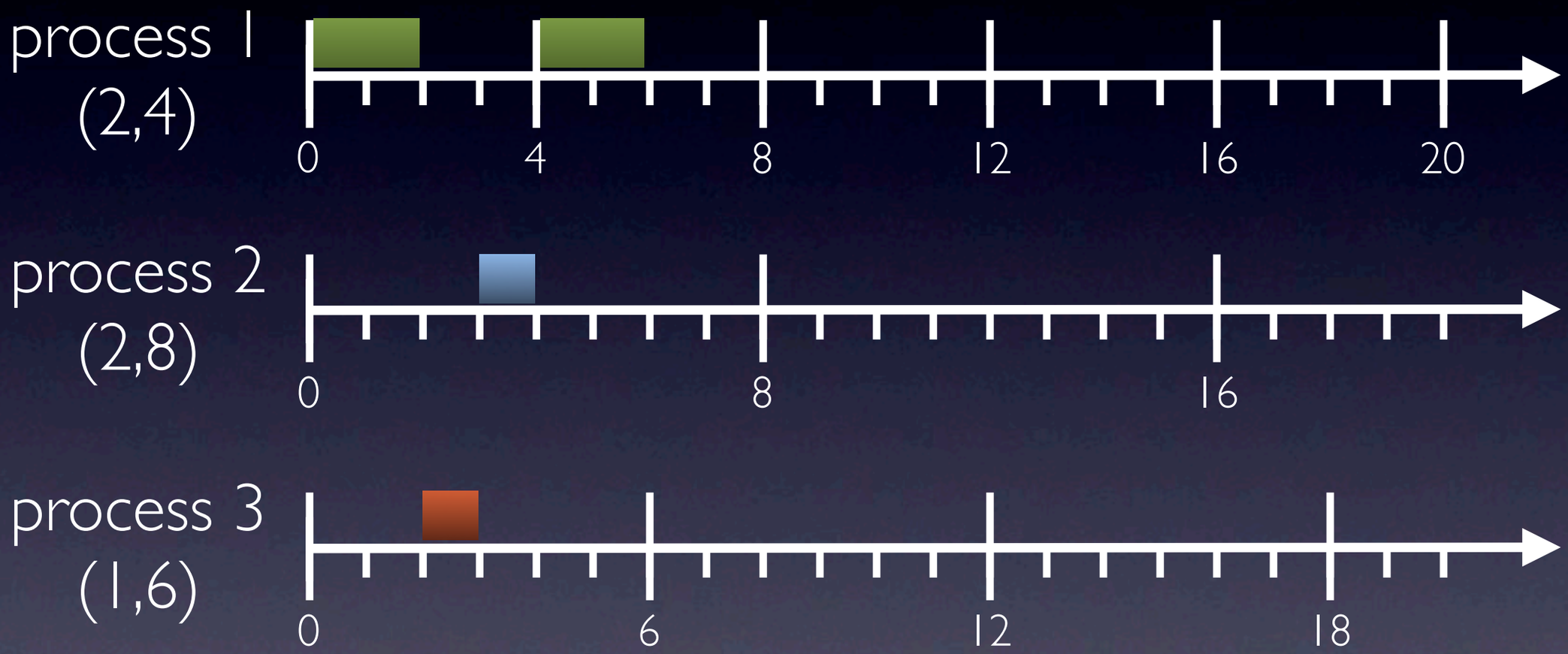
multiple processes are EDF-scheduled

VBS



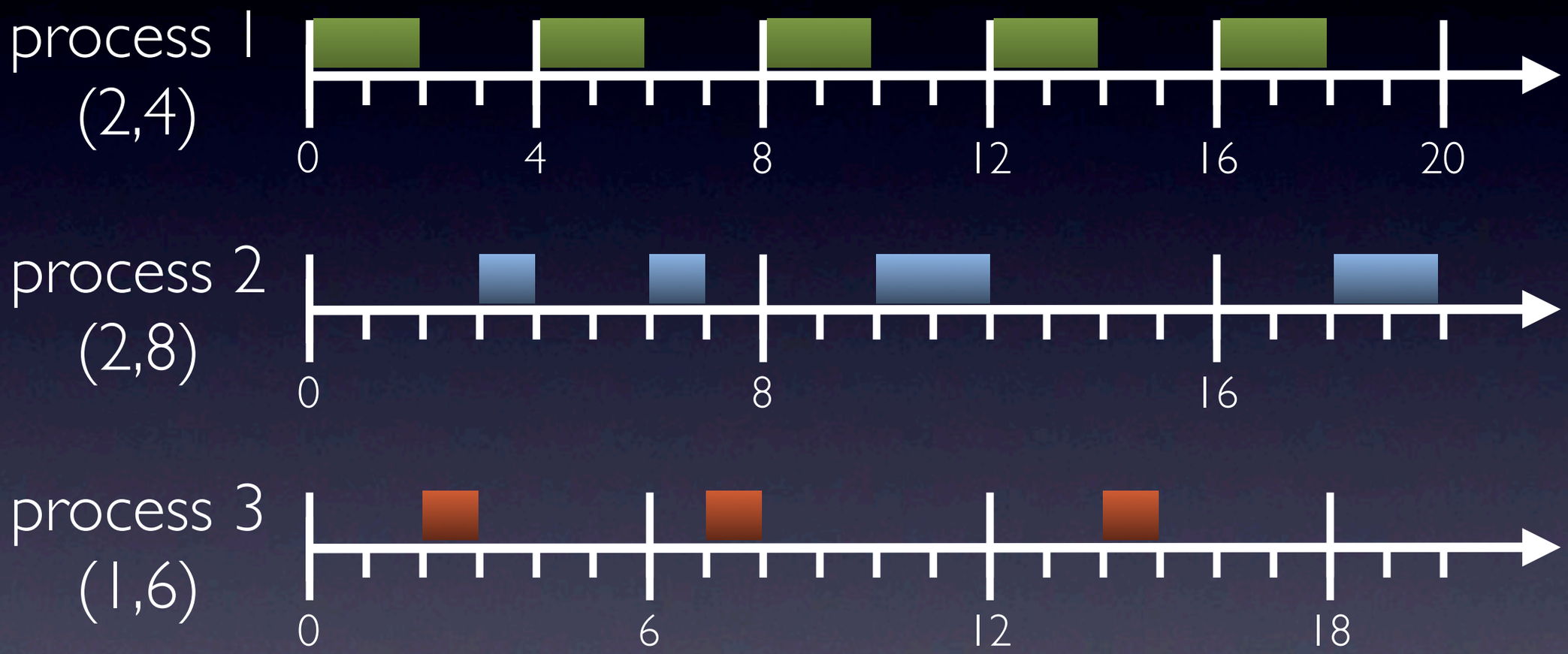
multiple processes are EDF-scheduled

VBS



multiple processes are EDF-scheduled

VBS



multiple processes are EDF-scheduled

Scheduling result and bounds

Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$ are schedulable if $\sum_{i=1}^{n} u_i \leq 1$

Scheduling result and bounds

Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$ are schedulable if $\sum_{i=1}^{n} u_i \leq 1$

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

- upper response-time bound $\left\lceil \frac{load}{\lambda} \right\rceil \pi + \pi - 1$
- lower response-time bound $\left\lceil \frac{load}{\lambda} \right\rceil \pi$
- jitter $\pi - 1$

Scheduling result and bounds

Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$ are schedulable if $\sum_{i=1}^{n} u_i \leq 1$

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

- upper response-time bound $\left\lceil \frac{load}{\lambda} \right\rceil \pi + \pi - 1$
- lower response-time bound $\left\lceil \frac{load}{\lambda} \right\rceil \pi$
- jitter $\pi - 1$

temporal isolation

Programmable temporal isolation

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

smaller π $\Rightarrow$

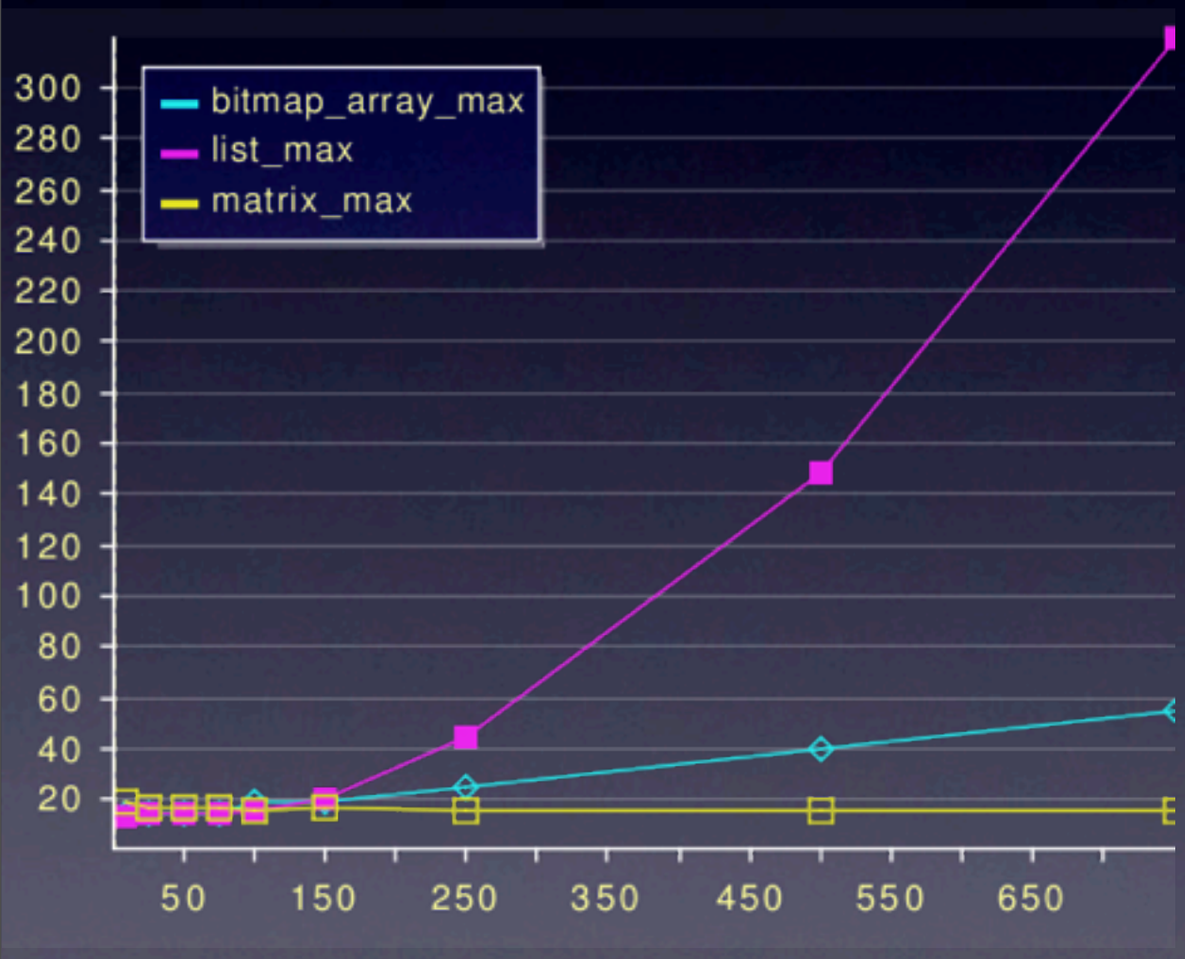
+ smaller jitter

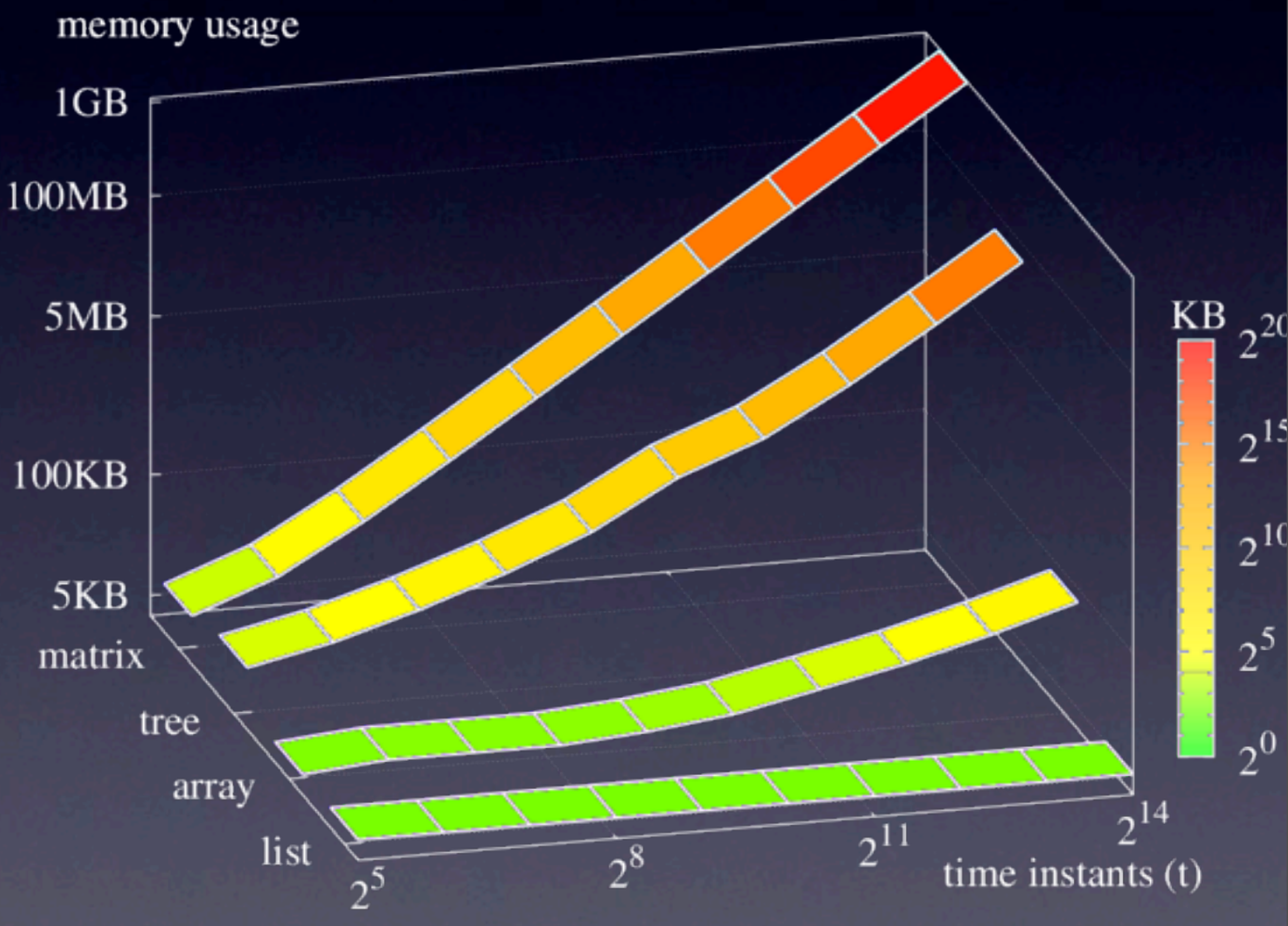
+ VBS response time closer to „ideal" response time

- higher administrative overhead
(more scheduler invocations)

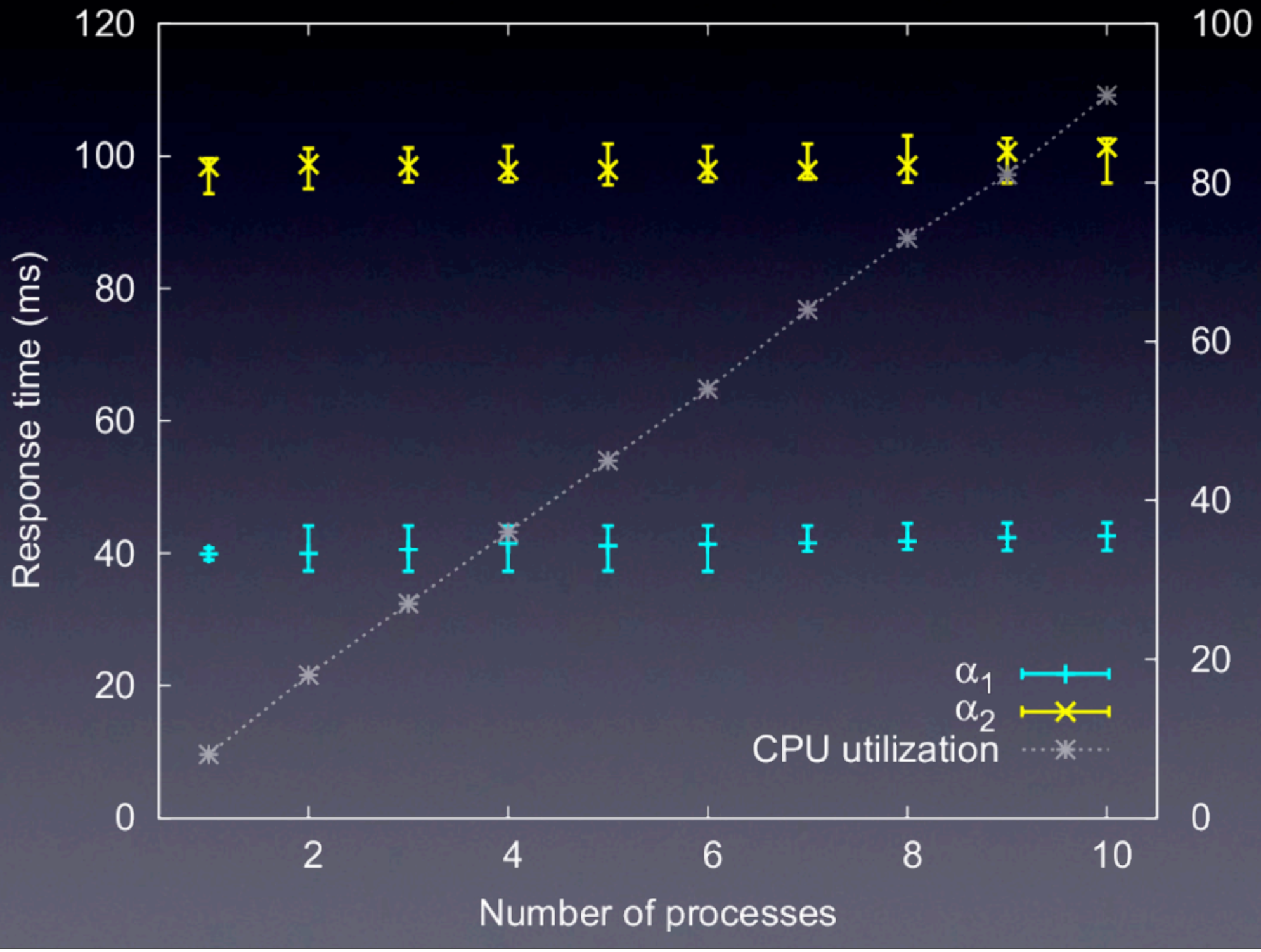
Finding the right λ,π is difficult (server design problem).

Scheduler overhead

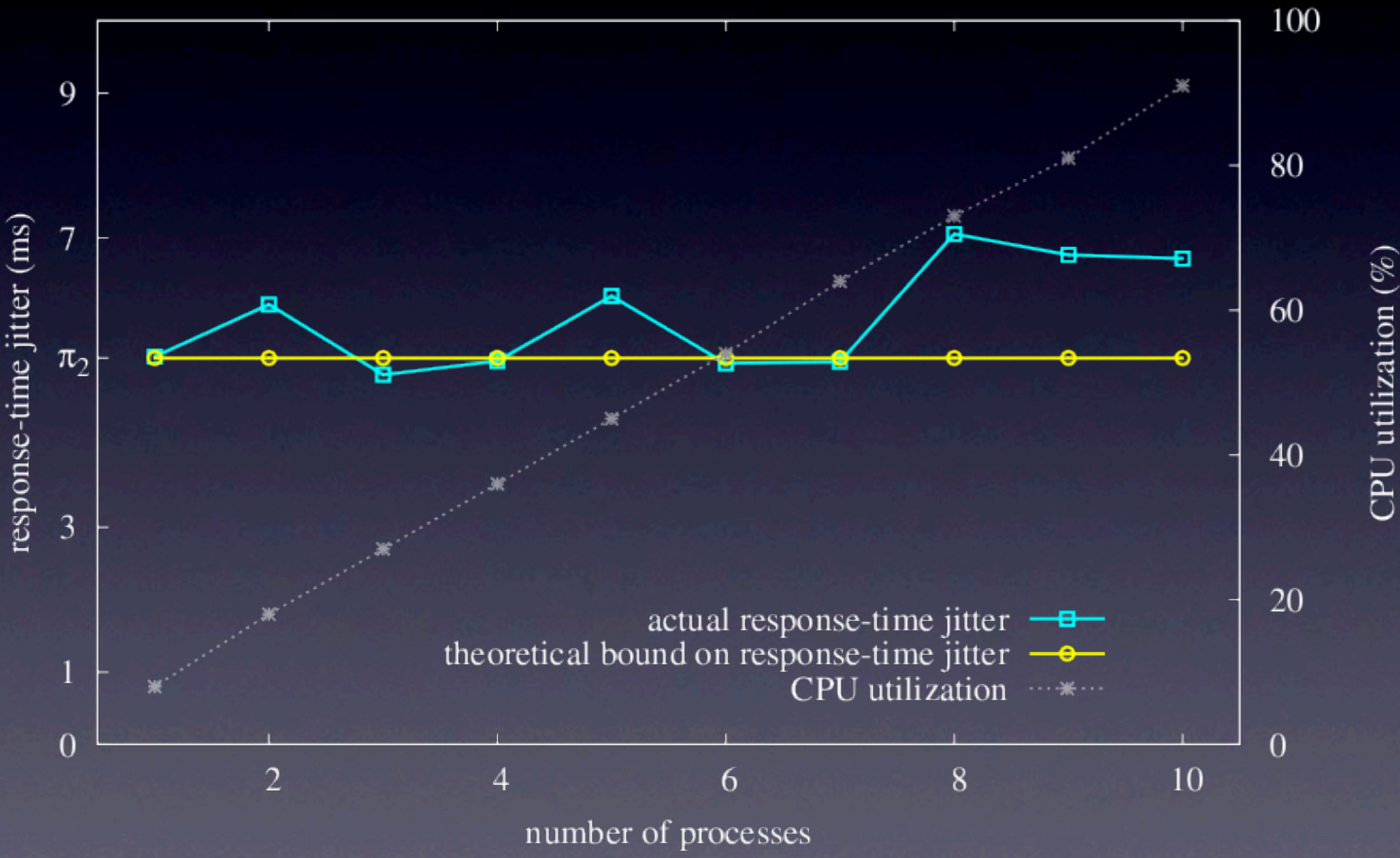




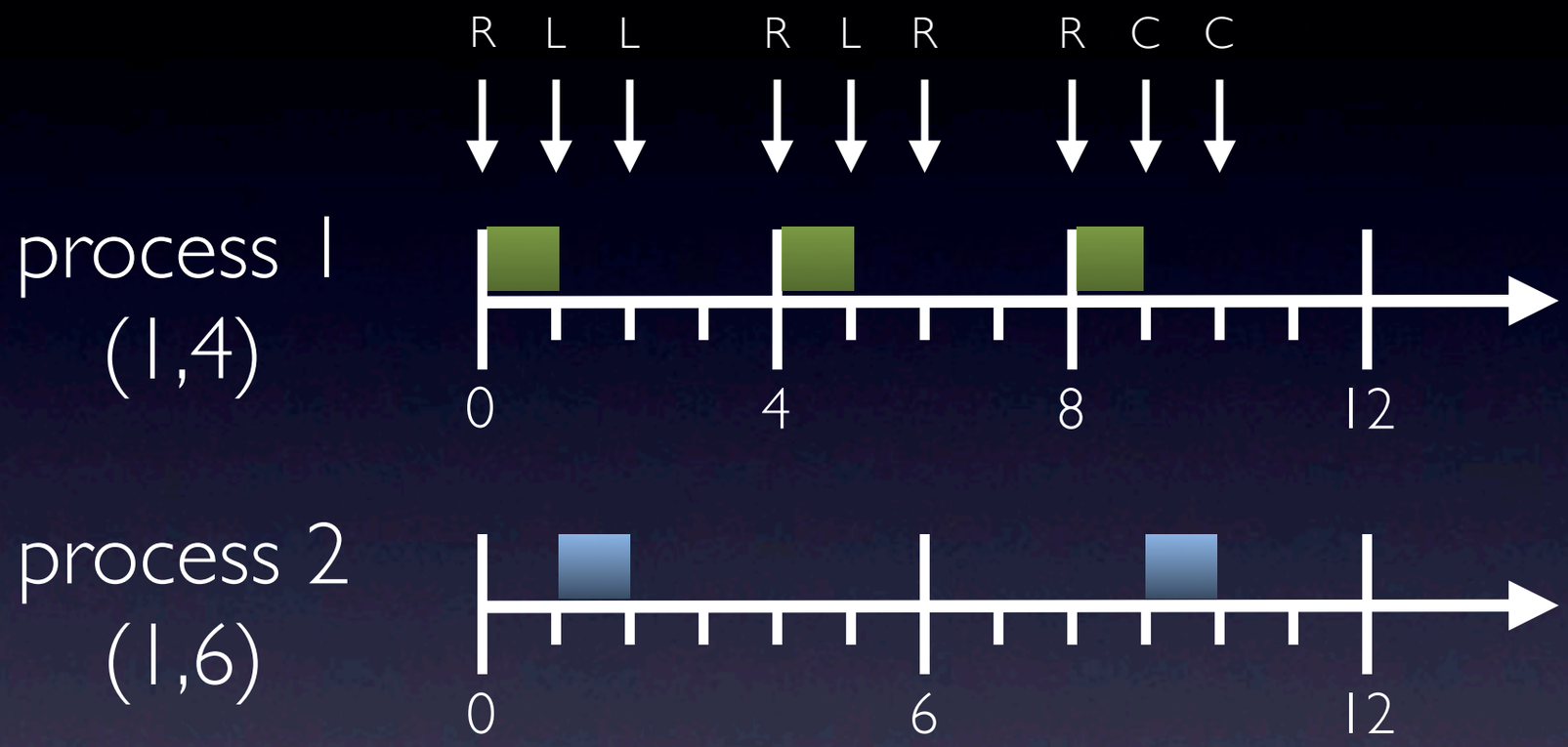
Bare-metal experiment



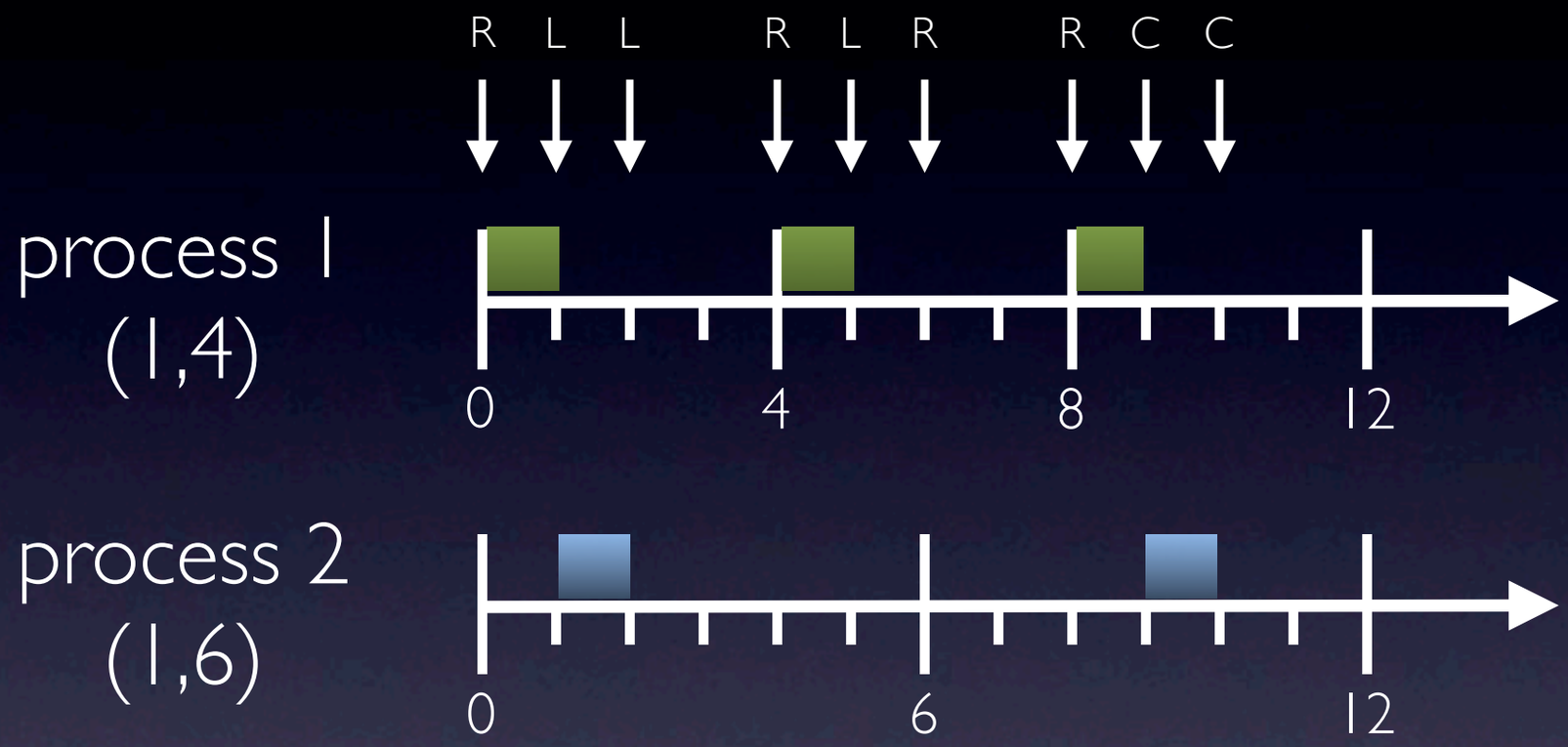
Bare-metal experiment



Overhead accounting

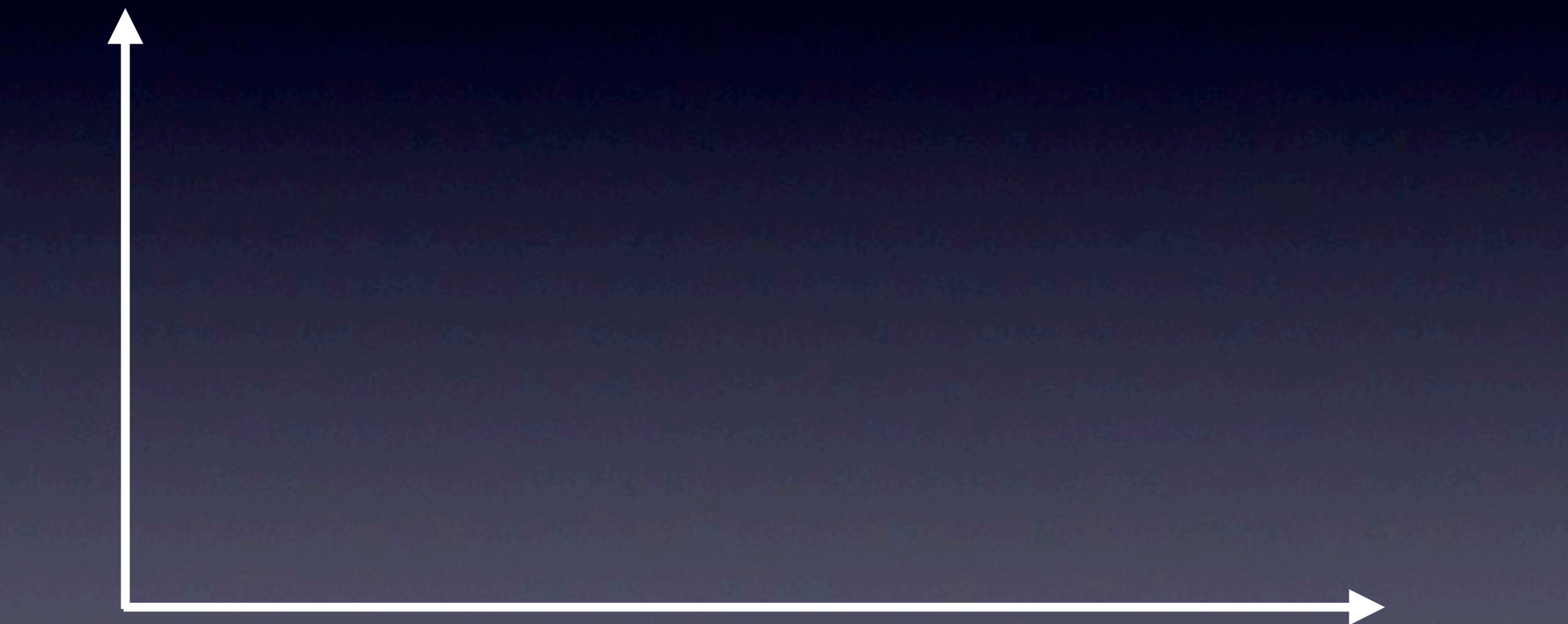


Overhead accounting



$$N = \left\lceil \frac{\pi}{\gcd(\text{all periods})} \right\rceil + 1$$

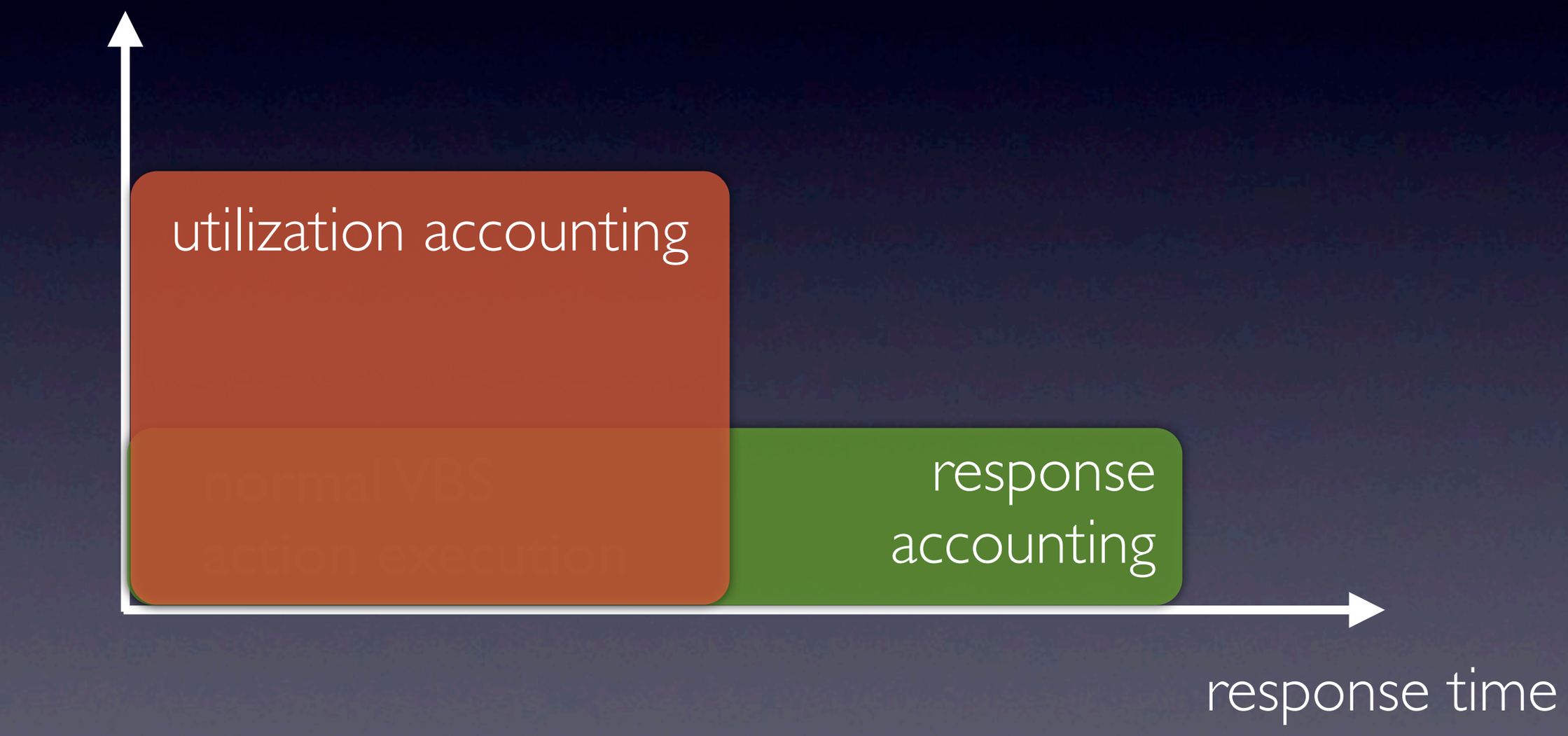
Overhead accounting



Overhead accounting

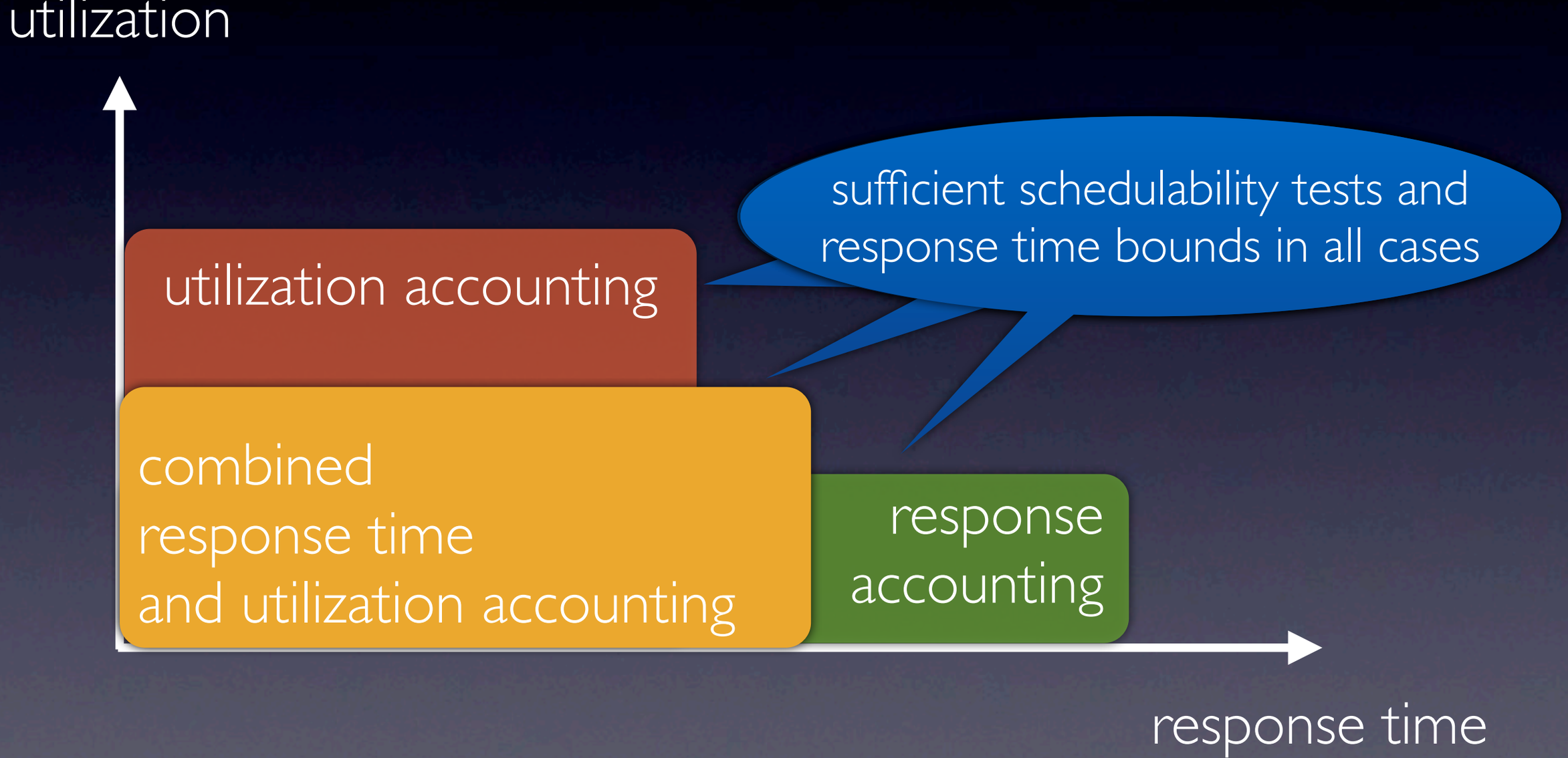
Overhead accounting

Overhead accounting



Overhead accounting

Overhead accounting



Without overhead

Without overhead

utilization

test: $\sum_i u_i \leq 1$

bounds:

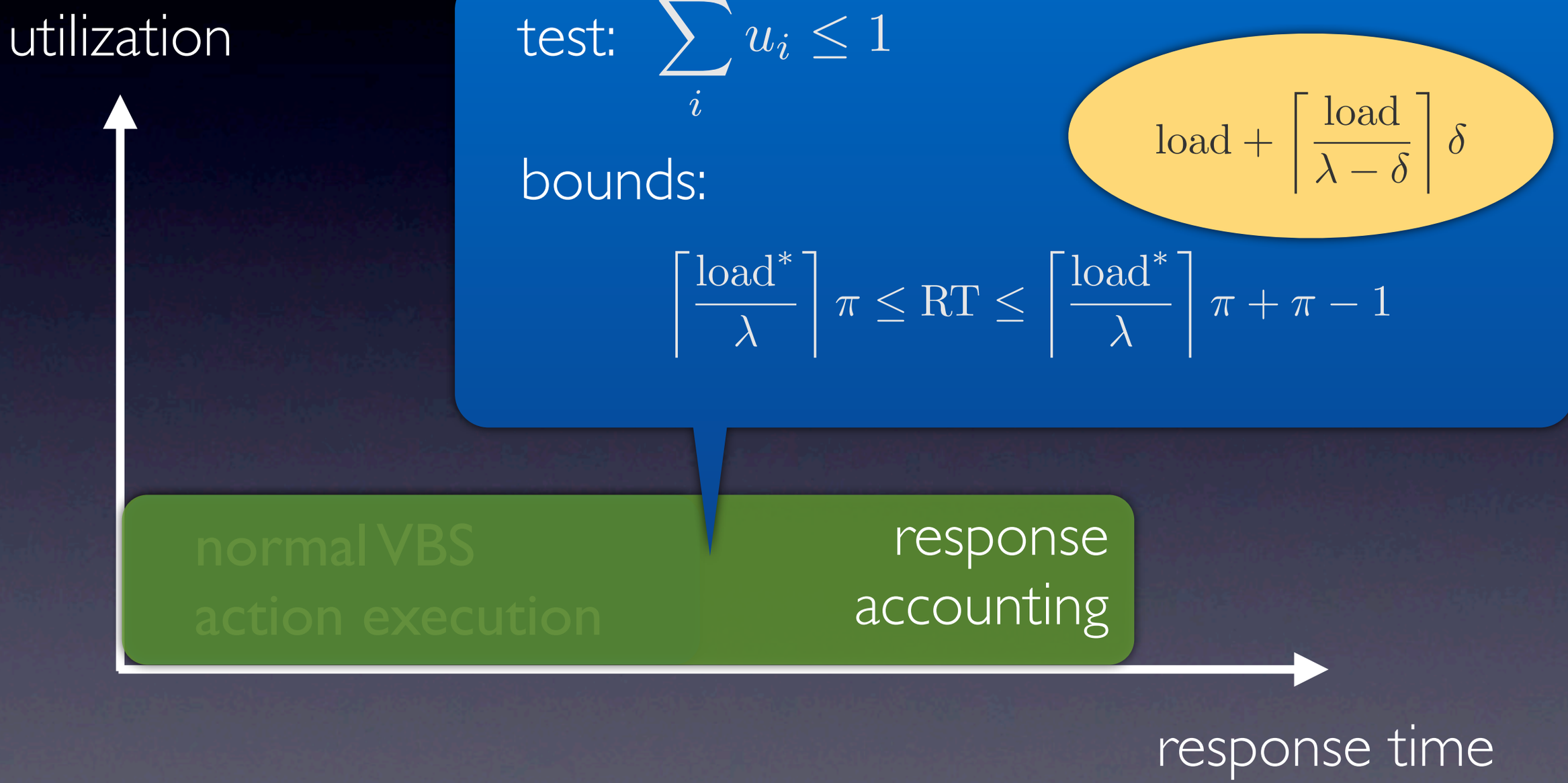
$$\left\lceil \frac{\text{load}}{\lambda} \right\rceil \pi \leq \text{RT} \leq \left\lceil \frac{\text{load}}{\lambda} \right\rceil \pi + \pi - 1$$

normal VBS
action execution

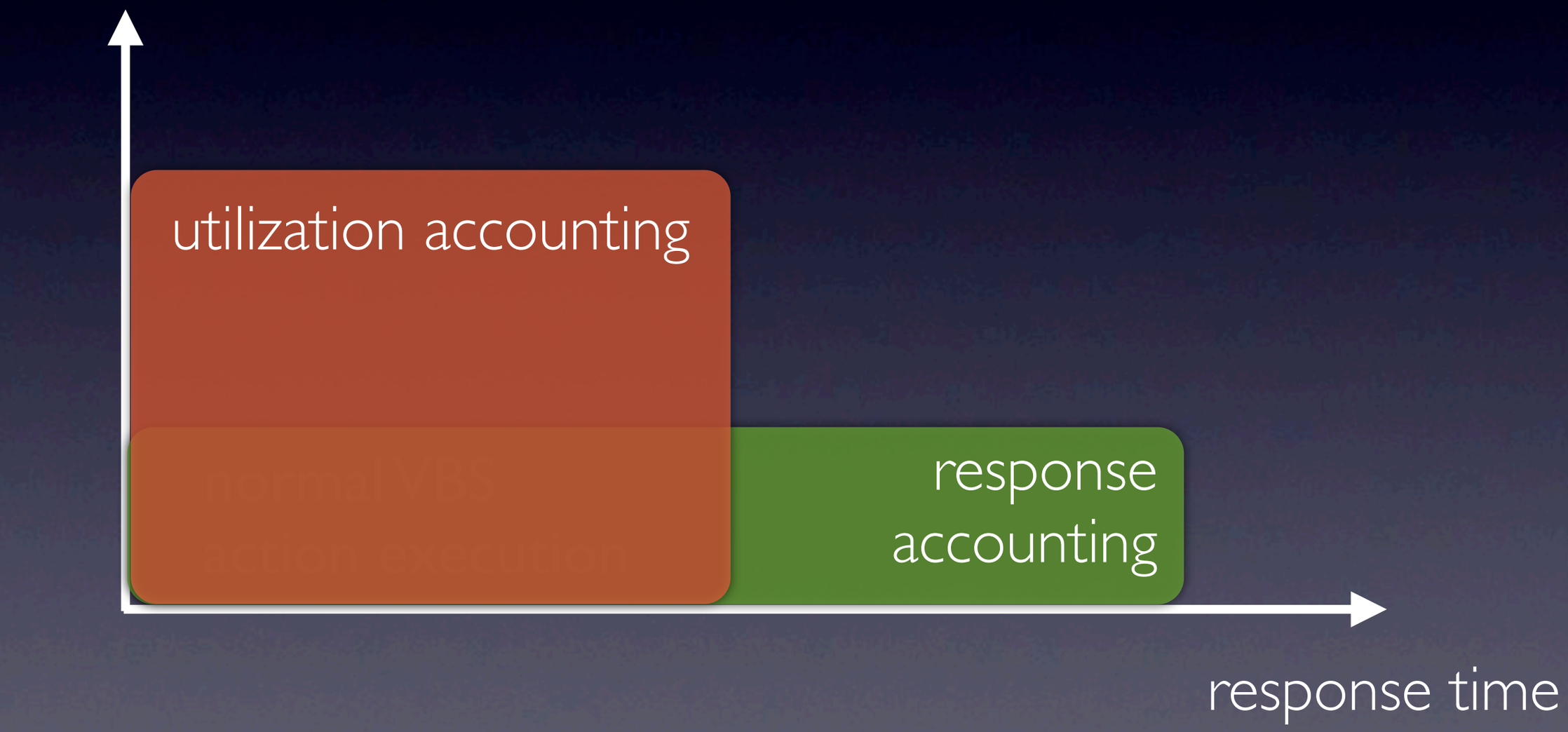
response time

Response accounting

Response accounting



Utilization accounting



Utilization accounting

test: $\sum_i \max_j \frac{\lambda_{i,j} + \delta_{i,j}}{\pi_{i,j}} \leq 1$

bounds:

$$\left\lceil \frac{\text{load}}{\lambda} \right\rceil \pi \leq \text{RT} \leq \left\lceil \frac{\text{load}}{\lambda} \right\rceil \pi + \pi - 1$$

utilization

utilization accounting

normal VBS action execution

response accounting

response time

Utilization accounting

Utilization accounting

utilization

utilization accounti

combined
response time
and utilization accounting

response
accounting

response time

test: $\sum_i \max_j \frac{\lambda_{i,j} + \delta^u_{i,j}}{\pi_{i,j}} \leq 1$

bounds:

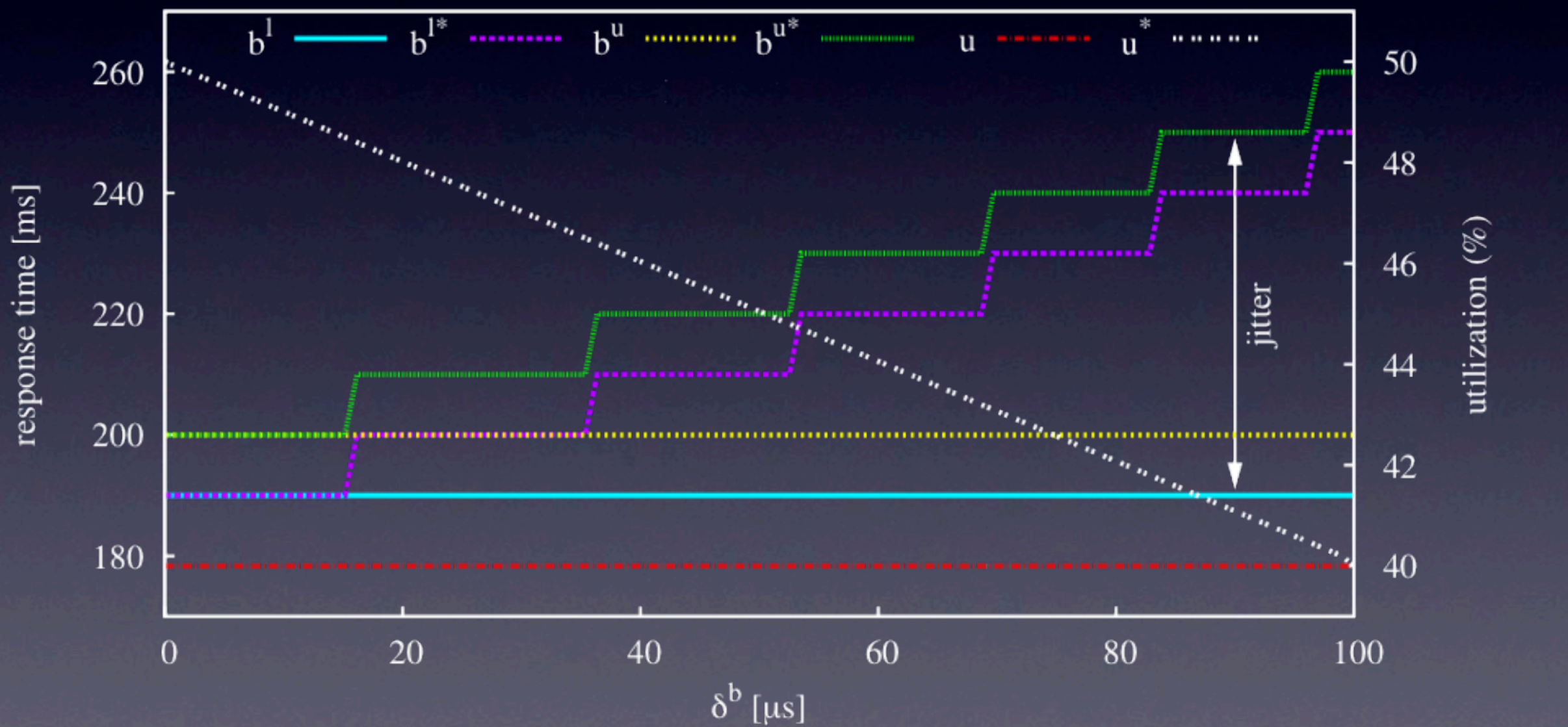
$$\left\lceil \frac{\text{load}^*}{\lambda} \right\rceil \pi \leq \text{RT} \leq \left\lceil \frac{\text{load}^*}{\lambda} \right\rceil \pi + \pi - 1$$

$$\text{load}' + \left\lceil \frac{\text{load}'}{\lambda} \right\rceil \delta^u$$

$$\text{load} + \left\lceil \frac{\text{load}}{\lambda - \delta^b} \right\rceil \delta^b$$

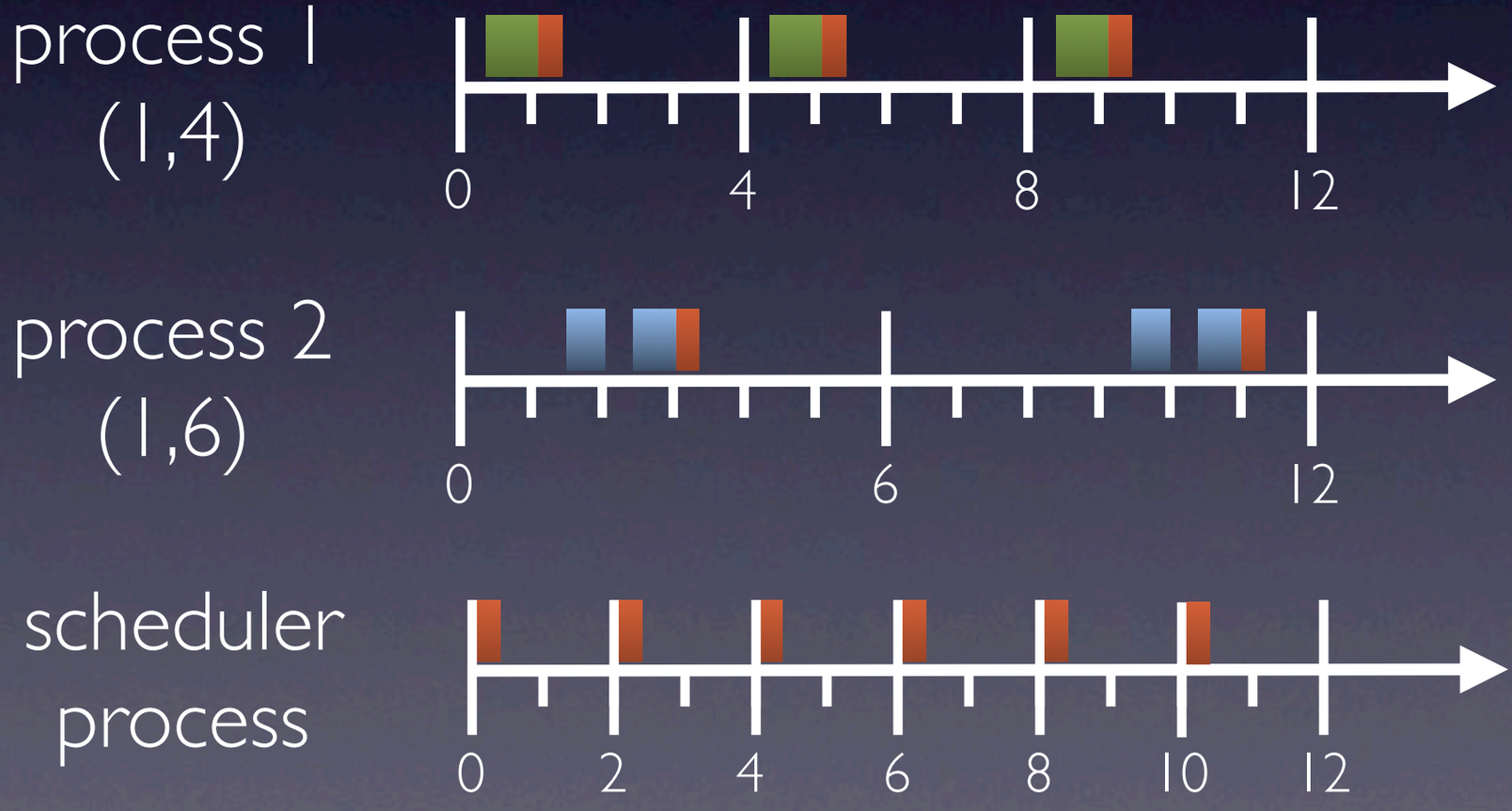
$$\delta = \delta^u + \delta^b$$

Overhead accounting

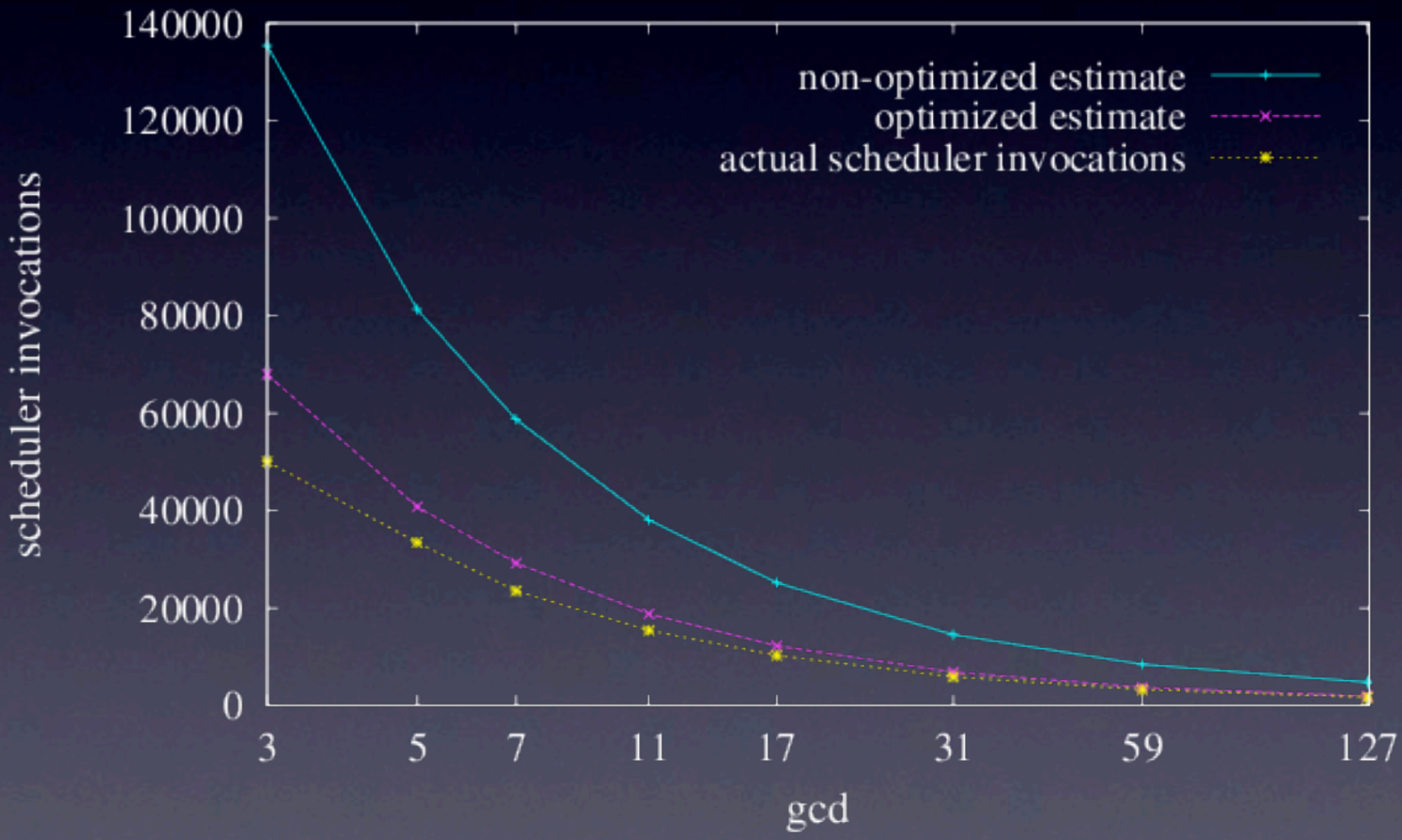


Optimization of the estimate

Scheduling invocations due to release can be considered as a separate process



Experiments



Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

process 1
(1,4)

process 2
(1,4)

Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

process 1
(2,4)

process 2
(2,4)

Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

process 1
(2,4)

process 2
(2,4)

Possible whenever there is slack in the system

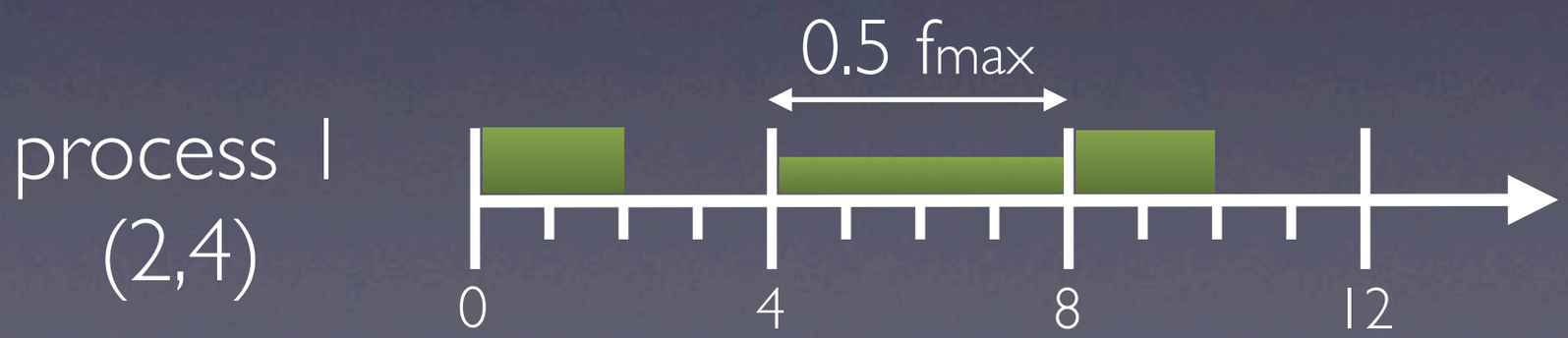
Power-Aware VBS

EDF frequency scaling result:

An EDF-schedulable set of tasks is still schedulable if the processor frequency in between any two release times is set to at least

$$Uc \cdot fmax$$

current total utilization of all released tasks in the considered interval of time between two releases



Frequency-scaling VBS

Slack

Frequency-scaling VBS

Slack

- Static Slack
- Dynamic Slack
  - Action Slack
  - Termination Slack

Frequency-scaling VBS

Frequency is scaled to the sum of the bandwidth caps and not changed at runtime

Static Slack

Slack

Dynamic Slack

Action Slack

Termination Slack

Frequency-scaling VBS

Frequency is scaled to the sum of the bandwidth caps and not changed at runtime

Static Slack

Frequency is scaled at release time to the sum of the utilizations of the released actions

Action Slack

Dynamic Slack

Termination Slack

Frequency-scaling VBS

Frequency is scaled to the sum of the bandwidth caps and not changed at runtime

Static Slack

Frequency is scaled at release time to the sum of the utilizations of the released actions

Action Slack

Dynamic Slack

New limits are computed for each action such that the upper response-time bound is maintained

Termination Slack

Frequency-scaling VBS

Frequency-scaling VBS

Static slack

$$f = \sum_{i=1}^{n} u_i \cdot f_{max}$$

Frequency-scaling VBS

Static slack

$$f = \sum_{i=1}^{n} u_i \cdot f_{max}$$

Action slack

$$f = \sum_{i=1}^{n} \frac{\lambda_{i,j}}{\pi_{i,j}} \cdot f_{max}$$

Frequency-scaling VBS

Static slack

$$f = \sum_{i=1}^{n} u_i \cdot f_{max}$$

Action slack

$$f = \sum_{i=1}^{n} \frac{\lambda_{i,j}}{\pi_{i,j}} \cdot f_{max}$$

Termination slack

$$f = \sum_{i=1}^{n} \frac{\lambda^*_{i,j}}{\pi_{i,j}} \cdot f_{max} \qquad \lambda^*_{i,j} = \left\lceil \frac{l_{i,j}}{n_{i,j}} \right\rceil \qquad n_{i,j} = \left\lceil \frac{l_{i,j}}{\lambda_{i,j}} \right\rceil$$

Frequency-scaling VBS

Static slack

$$f = \sum_{i=1}^{n} u_i \cdot f_{max}$$

Action slack

$$f = \sum_{i=1}^{n} \frac{\lambda_{i,j}}{\pi_{i,j}} \cdot f_{max}$$

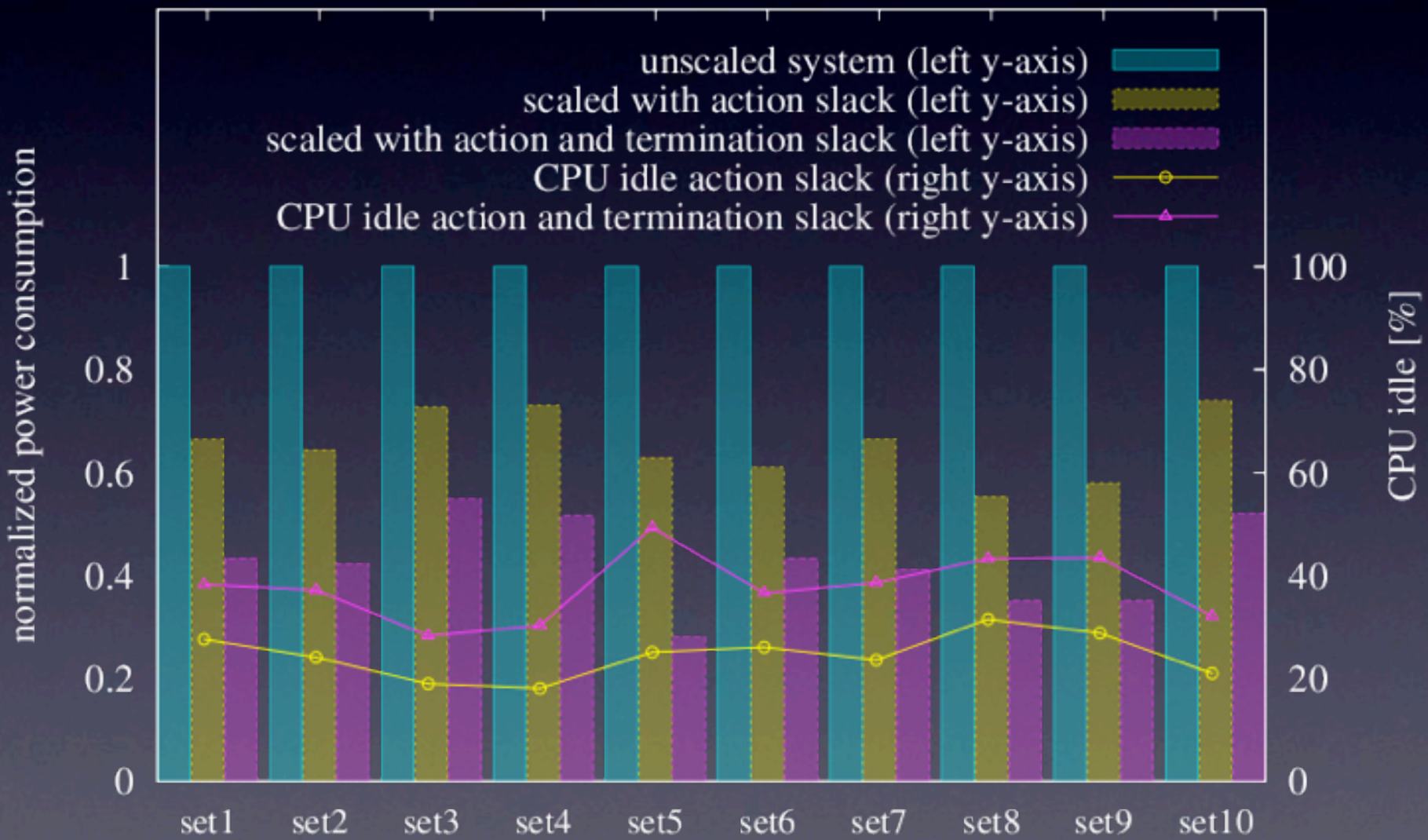
Termination slack

$$f = \sum_{i=1}^{n} \frac{\lambda^*_{i,j}}{\pi_{i,j}} \cdot f_{max} \qquad \lambda^*_{i,j} = \left\lceil \frac{l_{i,j}}{n_{i,j}} \right\rceil \qquad n_{i,j} = \left\lceil \frac{l_{i,j}}{\lambda_{i,j}} \right\rceil$$

Termination and action slack can be used separately or together

Power-Aware VBS

Assuming a simple power model ($P \propto V^2$)



Look-ahead FS-VBS

Look-ahead FS-VBS

With knowledge of future events:

- redistribute computation time between periods

Look-ahead FS-VBS

With knowledge of future events:

- redistribute computation time between periods
- optimal offline method

Look-ahead FS-VBS

With knowledge of future events:

- redistribute computation time between periods
- optimal offline method
- feasible online method

Look-ahead FS-VBS

With knowledge of future events:

redistribute computation time between periods

optimal offline method

feasible online method

May help to handle:

more complex power models

Look-ahead FS-VBS

With knowledge of future events:

- redistribute computation time between periods
- optimal offline method
- feasible online method

May help to handle:

- more complex power models
- frequency switching cost (time and power)

Look-ahead FS-VBS

With knowledge of future events:

- redistribute computation time between periods
- optimal offline method
- feasible online method

May help to handle:

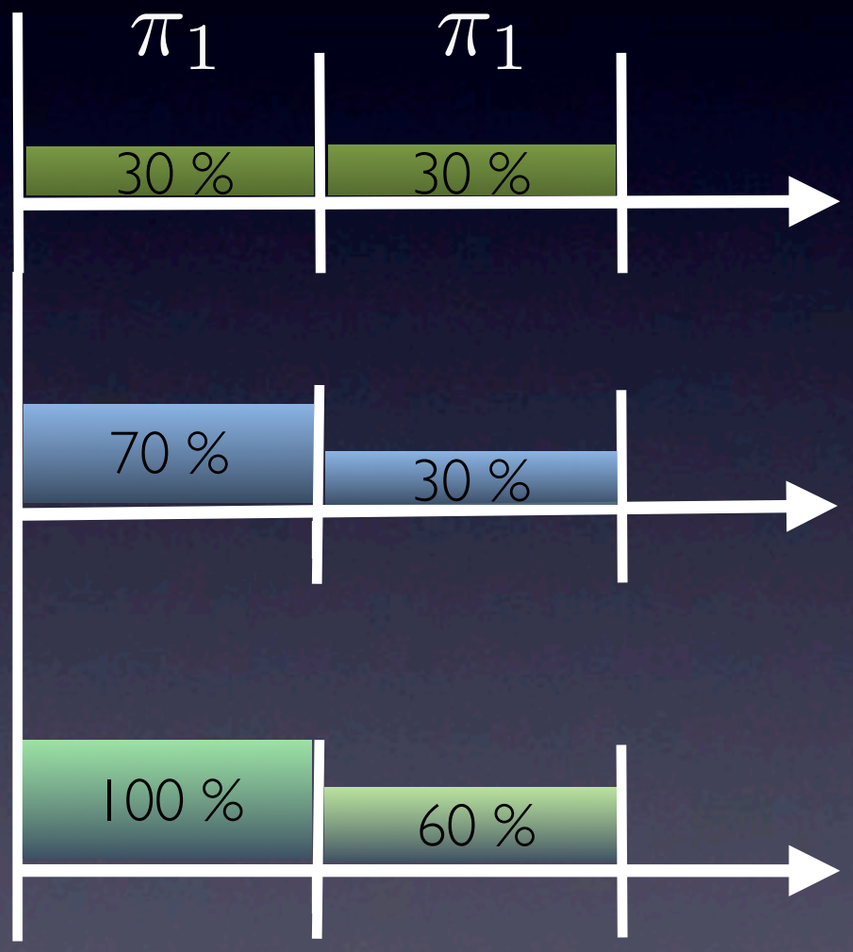
more complex power models

frequency switching cost (time and power)

- time overhead included using overhead accounting

Look-ahead FS-VBS

process 1
30 %



other utilization

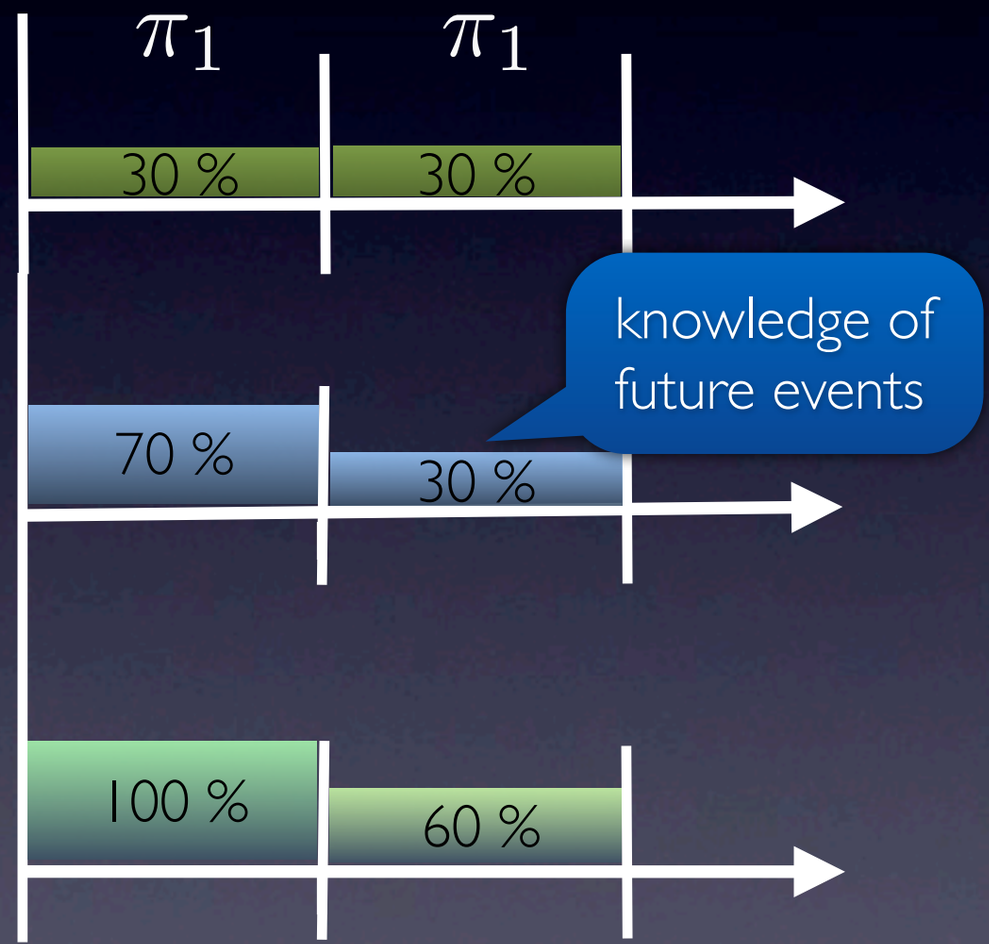
total utilization

Look-ahead FS-VBS

process 1
30 %

other utilization

total utilization

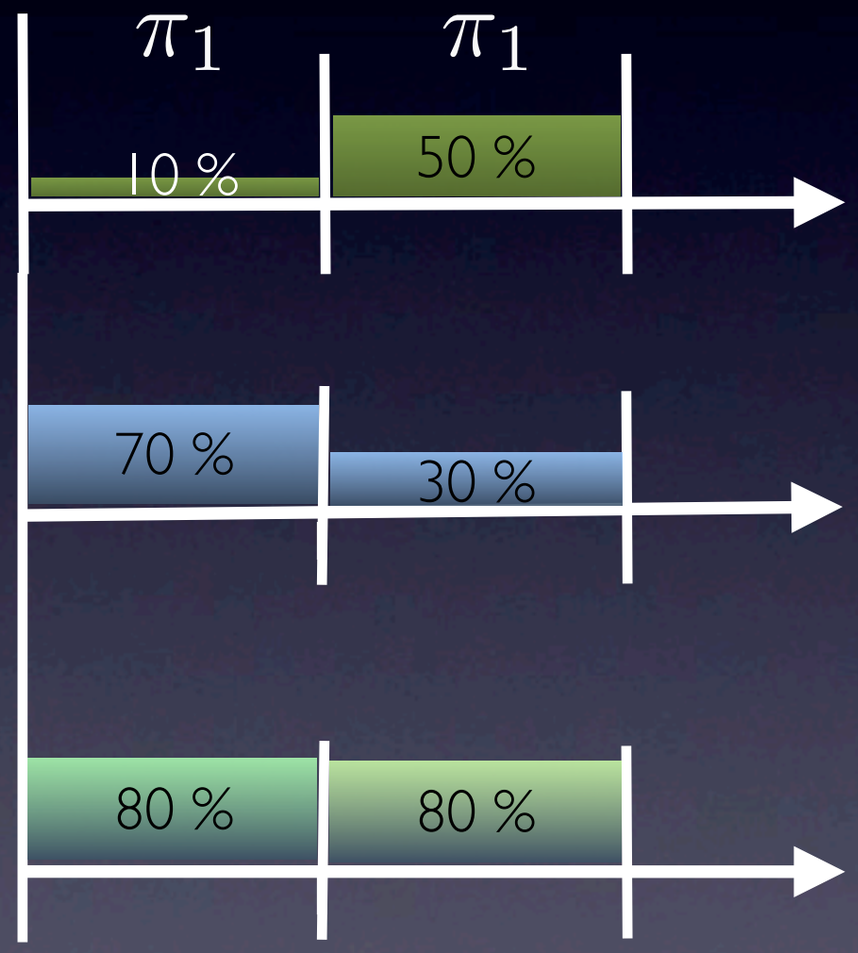


Look-ahead FS-VBS

process 1 modified

other utilization

total utilization

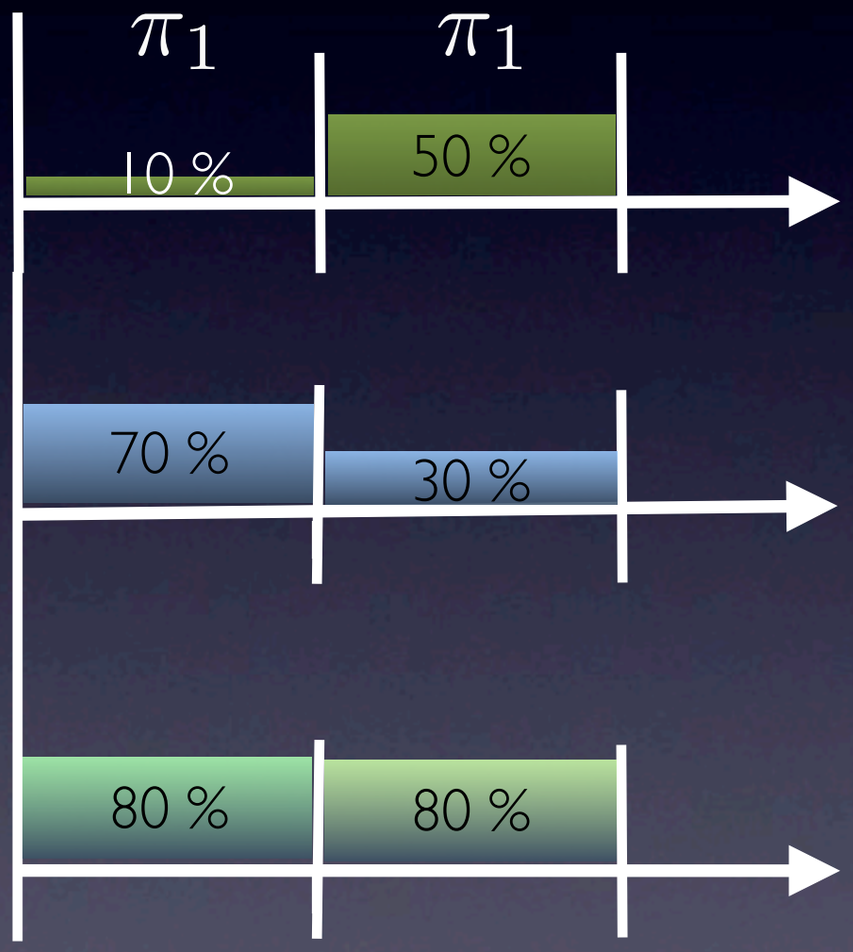


Look-ahead FS-VBS

process 1 modified

other utilization

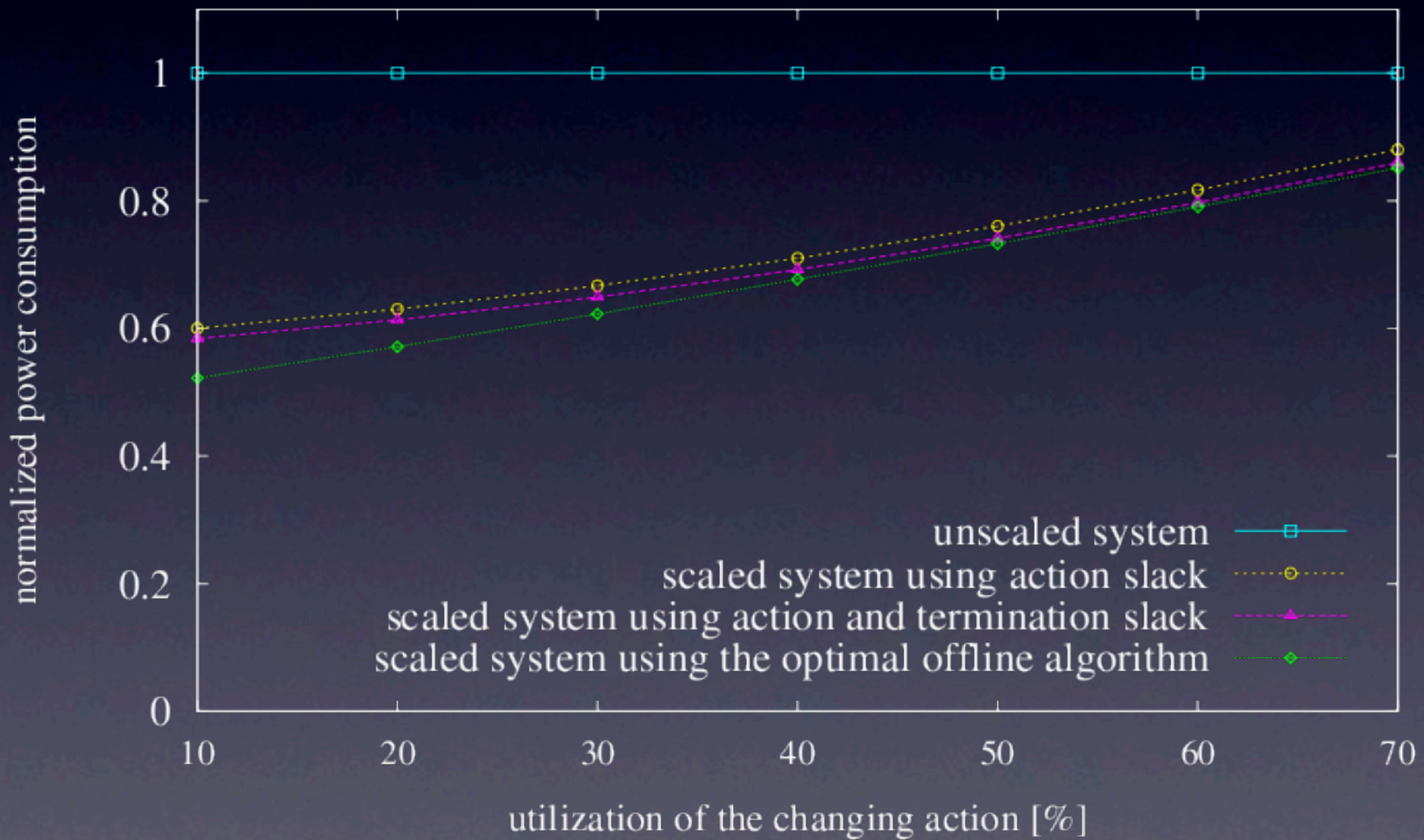
total utilization



actual improvement depends on the power model

Look-ahead FS-VBS

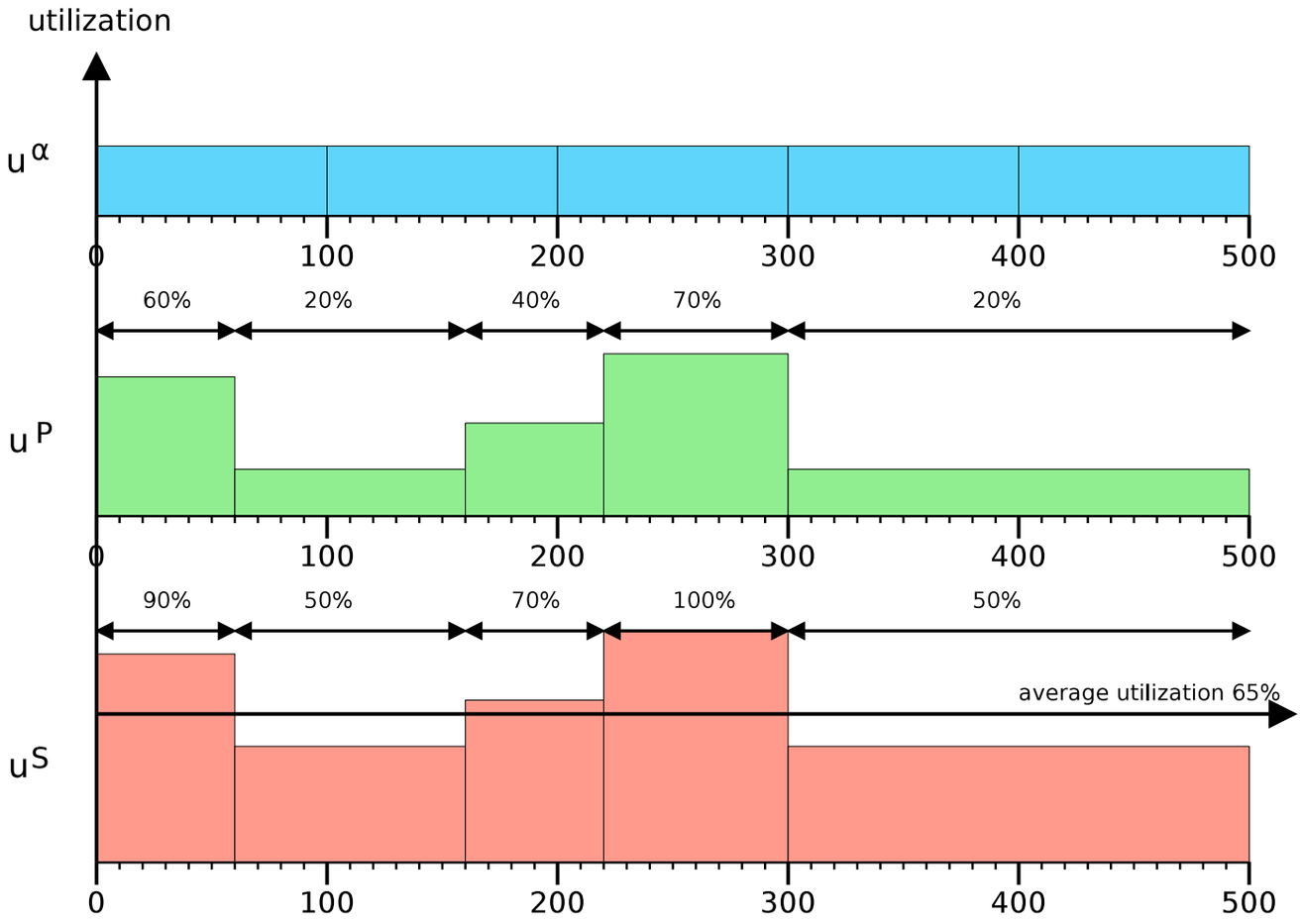
Assuming a simple power model ($P \propto V^2$)



Look-ahead online FS-VBS

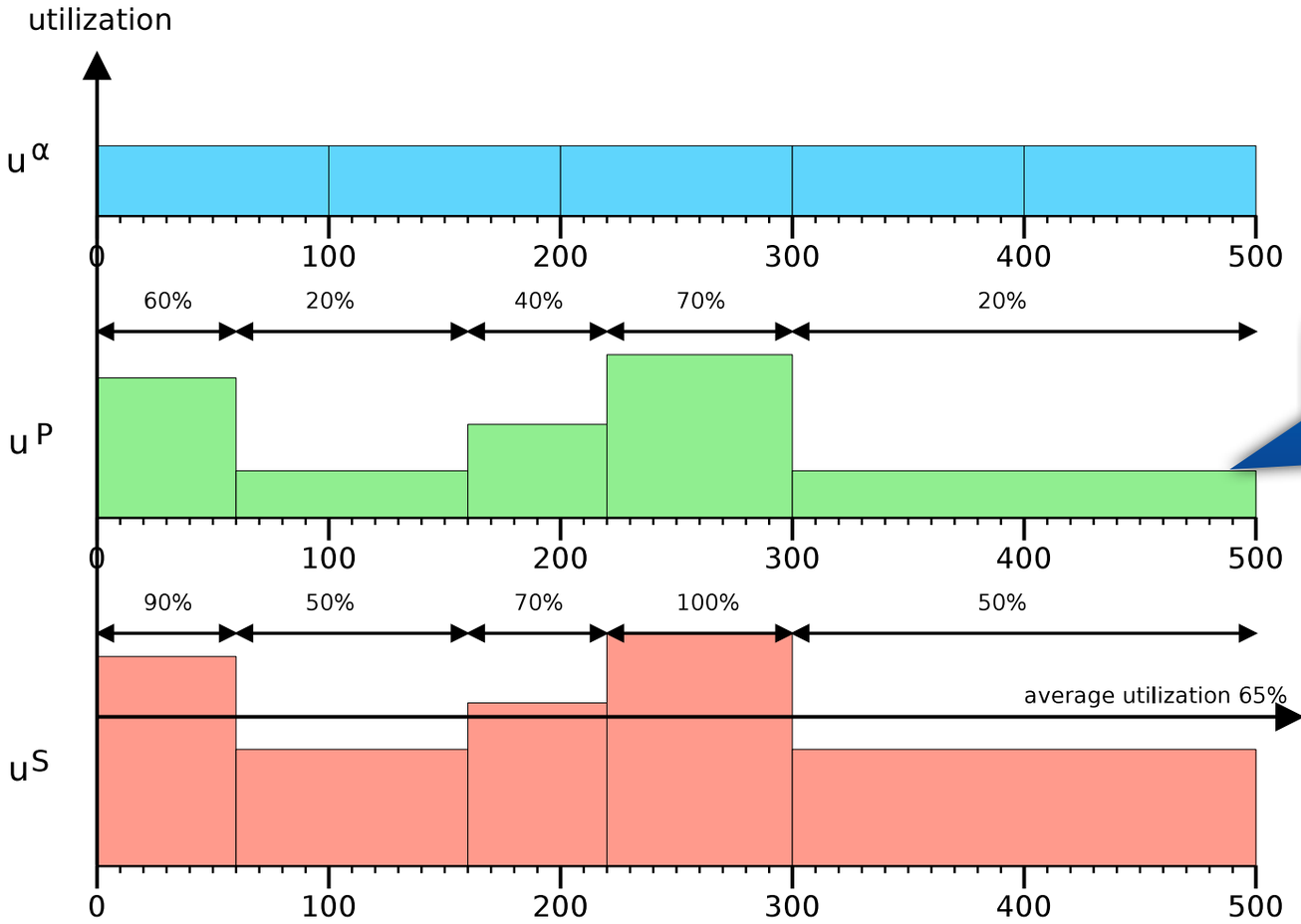
Look-ahead online FS-VBS

Assume a simple power model ($P \propto V^2$)



Look-ahead online FS-VBS

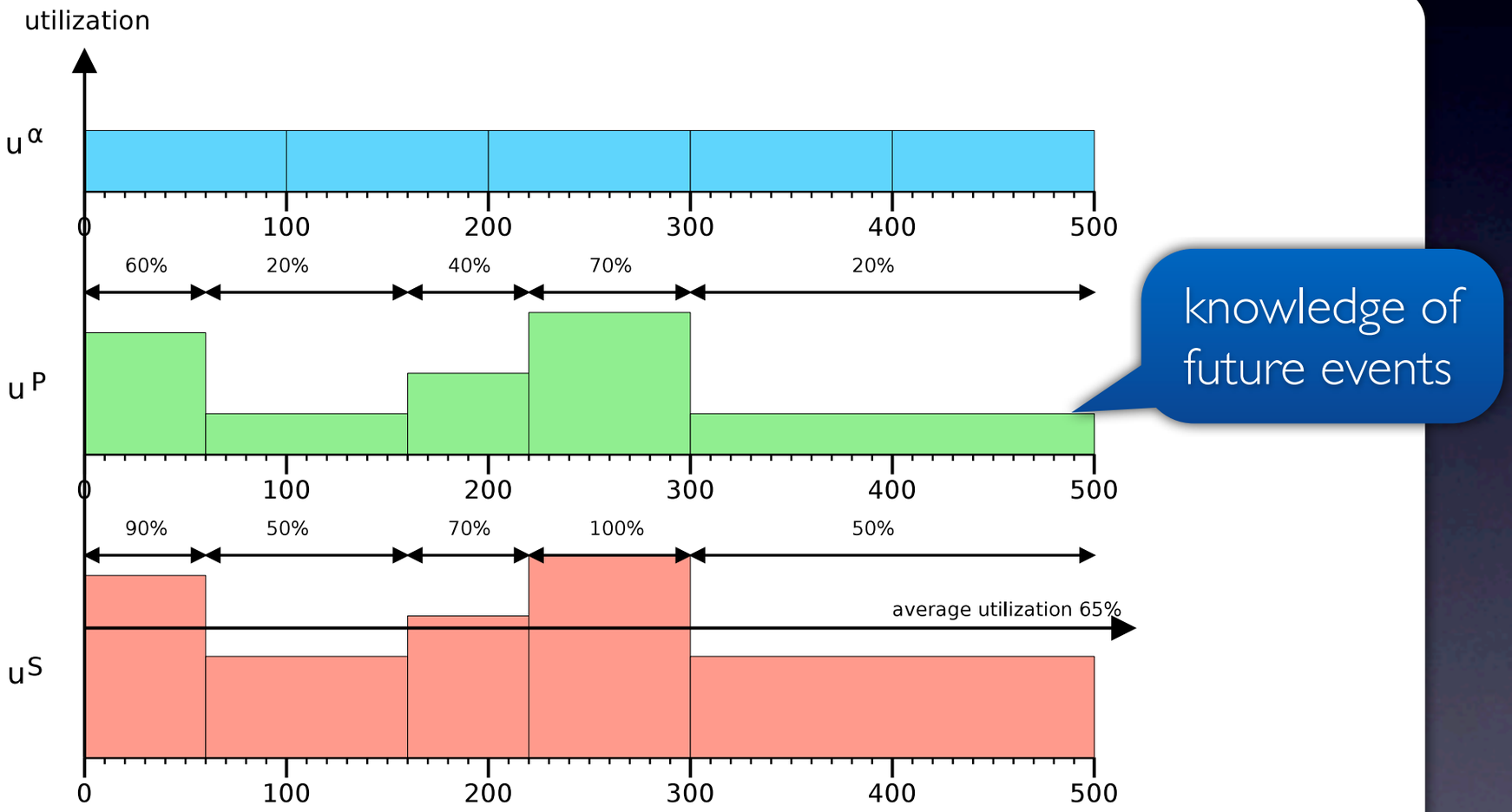
Assume a simple power model ($P \propto V^2$)



knowledge of future events

Look-ahead online FS-VBS

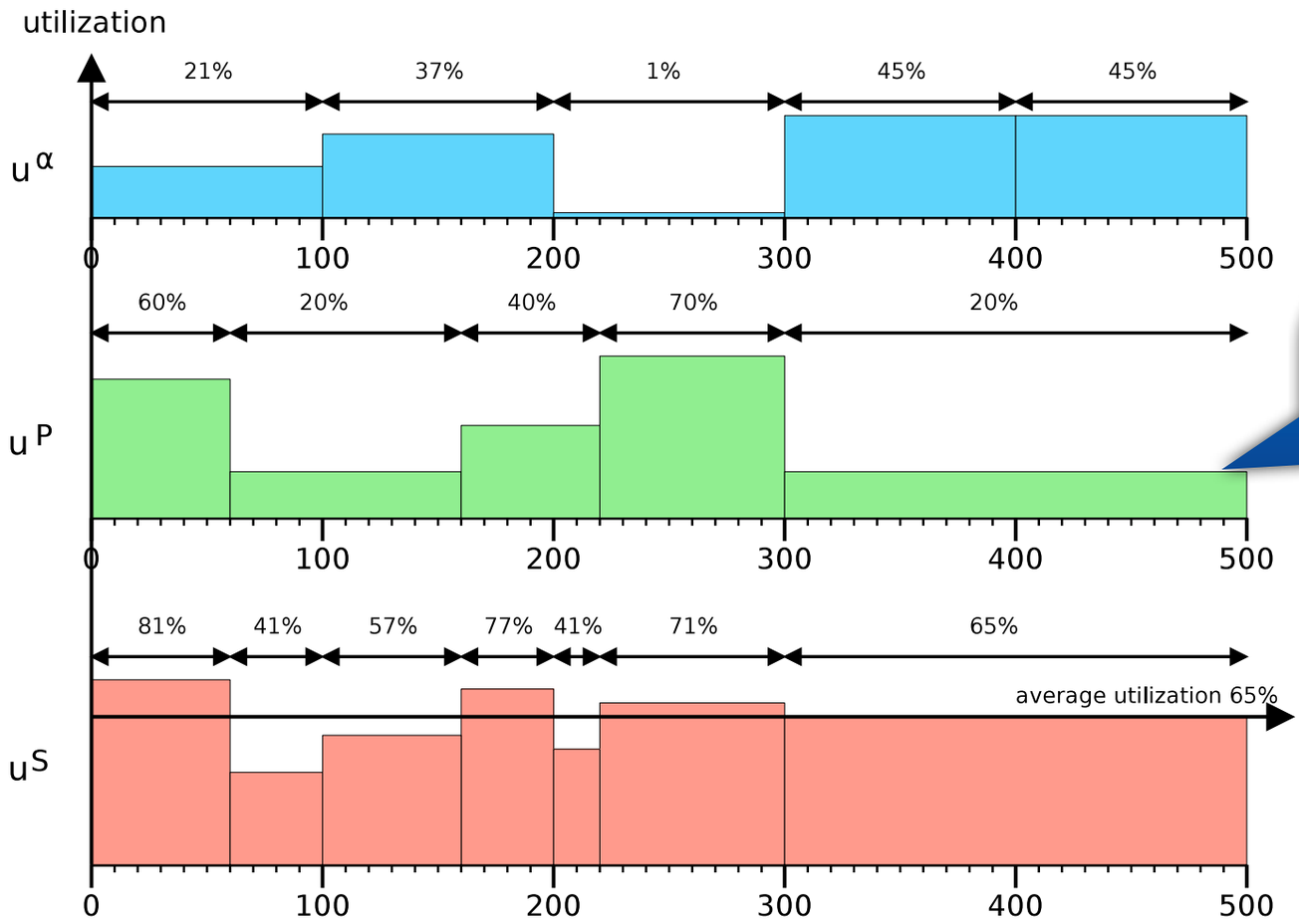
Assume a simple power model ($P \propto V^2$)



Modify the limits in each period (whenever possible) s.t. the utilization approximates the average utilization

Look-ahead online FS-VBS

Assume a simple power model ($P \propto V^2$)



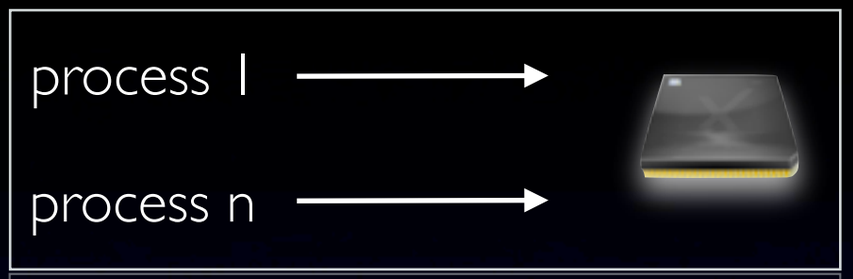
knowledge of future events

Modify the limits in each period (whenever possible) s.t. the utilization approximates the average utilization

Conclusions

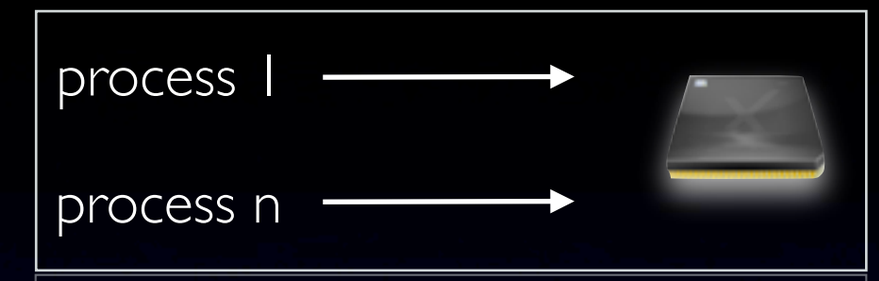
Conclusions

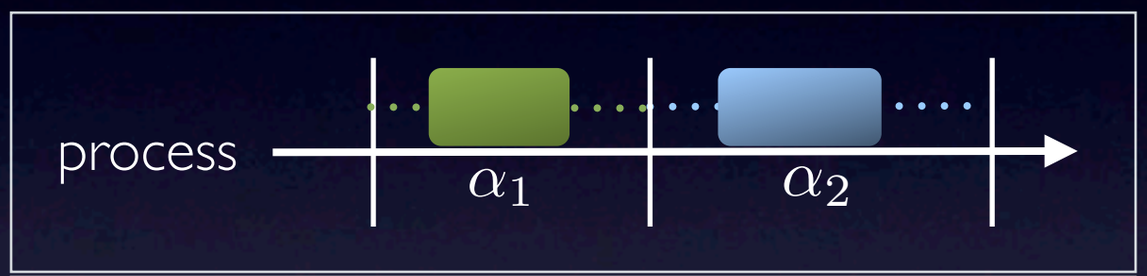
- Reservation-based scheduling for temporal isolation



Conclusions

- Reservation-based scheduling for temporal isolation

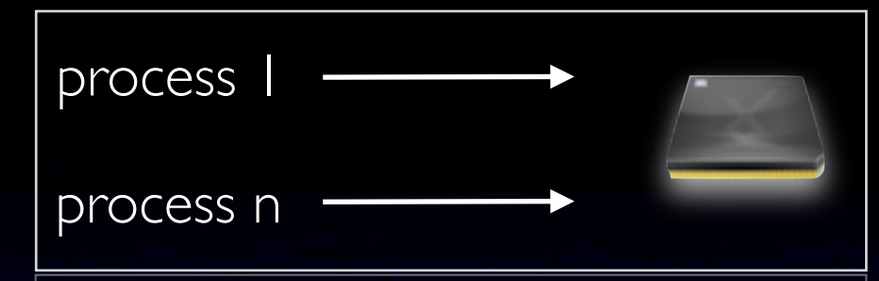


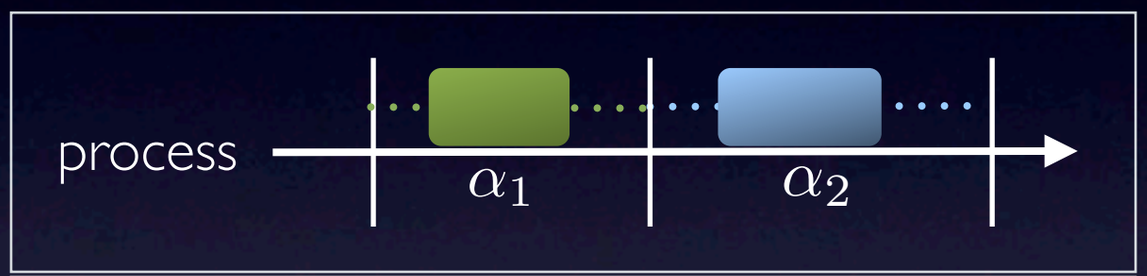


- VBS enables variable execution speed

Conclusions

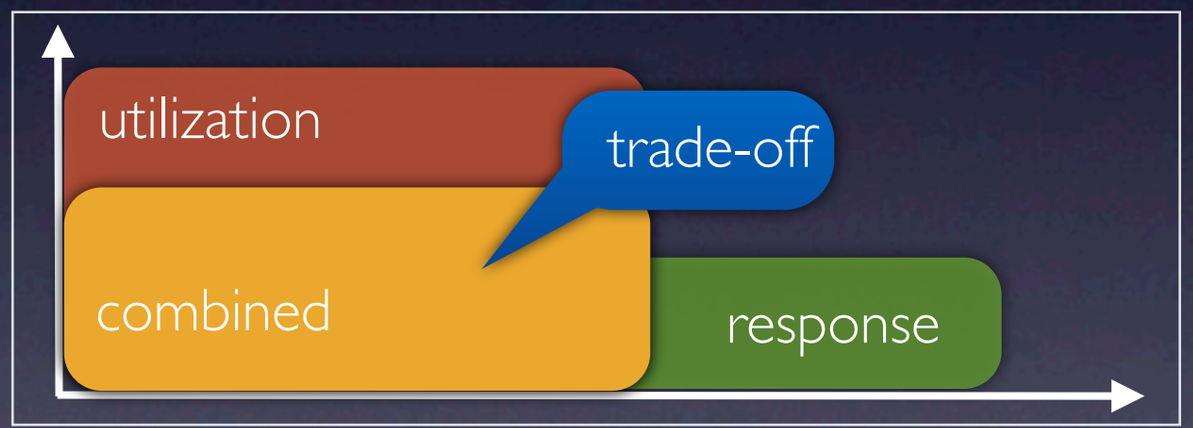
- Reservation-based scheduling for temporal isolation





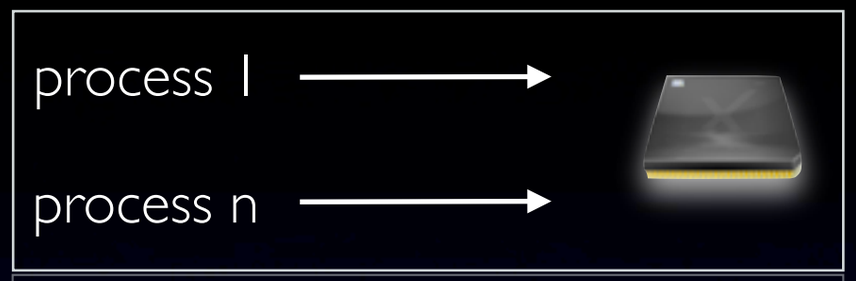
- VBS enables variable execution speed

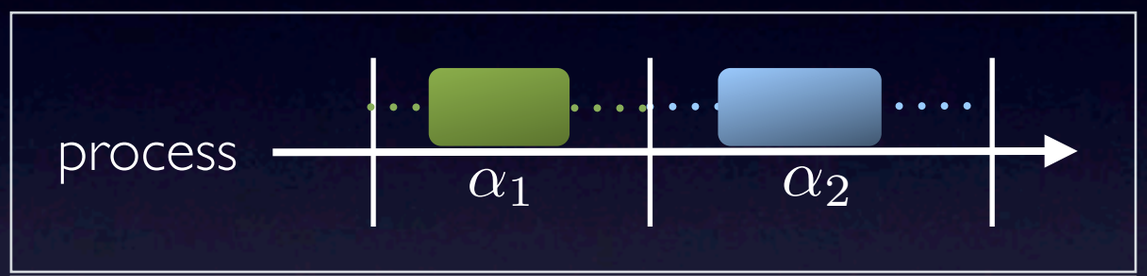
- Overhead accounting



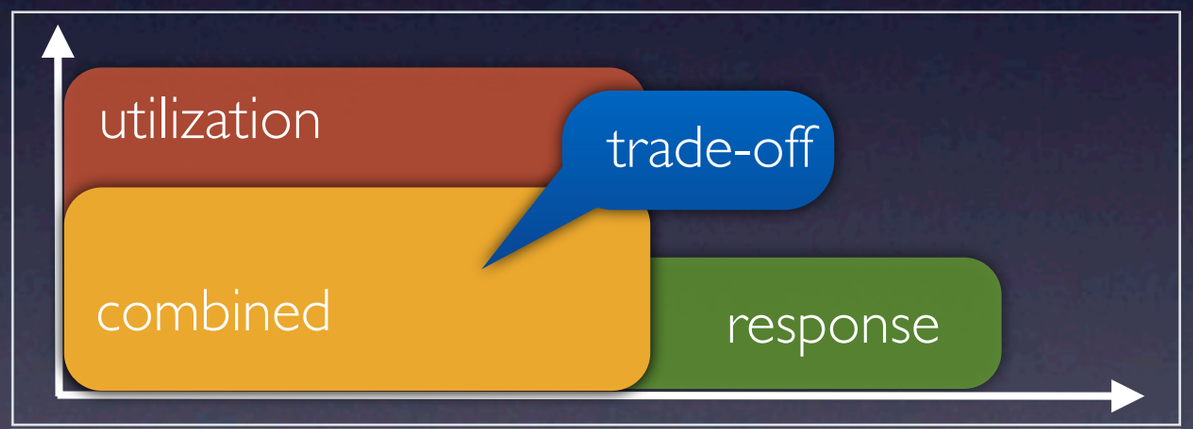
Conclusions

- Reservation-based scheduling for temporal isolation

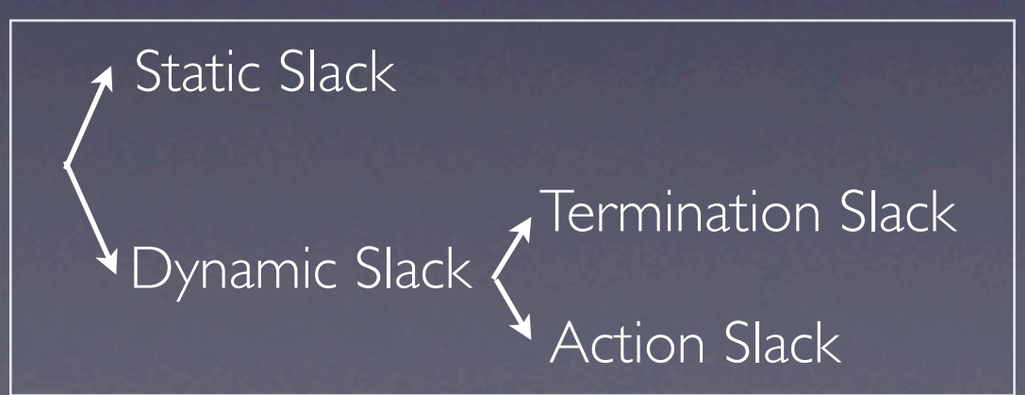




- VBS enables variable execution speed



- Overhead accounting



- Power-aware VBS

Thank you

S.S. Craciunas, C.M. Kirsch, and A. Sokolova - Power-Aware Temporal Isolation with Variable-Bandwidth Servers. In International Conference on Embedded Software (EMSOFT), ACM, 2010.

S.S. Craciunas, C.M. Kirsch, and A. Sokolova - Response Time versus Utilization in Scheduler Overhead Accounting. In Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 291-300, IEEE, 2010.

S.S. Craciunas, C.M. Kirsch, H. Payer, H. Röck, and A. Sokolova - Programmable Temporal Isolation through Variable-Bandwidth Servers. In IEEE Symposium on Industrial Embedded Systems (SIES), pp. 171-180, IEEE, 2009.