# Power-aware Scheduling for Real-Time Systems

Silviu Craciunas, Christoph Kirsch, Ana Sokolova
Department of Computer Sciences
University of Salzburg

# Power-Aware Scheduling

# Power-Aware Scheduling

High performance is needed only for a small fraction of time

# Power-Aware Scheduling

High performance is needed only for a small fraction of time

Dynamic Voltage and Frequency Scaling

$$p(f) = c_0 + c_1 f^\alpha$$

# Power-Aware Scheduling

High performance is needed only for a small fraction of time

Dynamic Voltage and Frequency Scaling

$$p(f) = c_0 + c_1 f^\alpha$$

Example: Intel X-Scale

$$p(f) = 1520 f^3 + 80 \text{ mWatt}$$

# Power-Aware Scheduling

High performance is needed only for a small fraction of time

Dynamic Voltage and Frequency Scaling

$$p(f) = c_0 + c_1 f^{\alpha}$$

Example: Intel X-Scale

$$p(f) = 1520 f^3 + 80 \text{ mWatt}$$

5 available speeds: (0.15, 0.4, 0.6, 0.8, 1) GHz

# Power-Aware Scheduling

High performance is needed only for a small fraction of time

Dynamic Voltage and Frequency Scaling

$$p(f) = c_0 + c_1 f^\alpha$$

Example: Intel X-Scale

$$p(f) = 1520 f^3 + 80 \text{ mWatt}$$

5 available speeds: (0.15, 0.4, 0.6, 0.8, 1) GHz

power consumption: (80, 170, 400, 900, 1600) mWatt

# Power-Aware Scheduling

# Power-Aware Scheduling

Scaling the frequency results in modified execution time

# Power-Aware Scheduling

Scaling the frequency results in modified execution time

Scaling to 50% results in double execution time

# Power-Aware Scheduling

Scaling the frequency results in modified execution time

Scaling to 50% results in double execution time

Deadlines remain the same

# Power-Aware Scheduling

Scaling the frequency results in modified execution time

Scaling to 50% results in double execution time

Deadlines remain the same

Main goal: Minimize power while maintaining the real-time properties (deadlines)

# Power-Aware EDF

# Power-Aware EDF

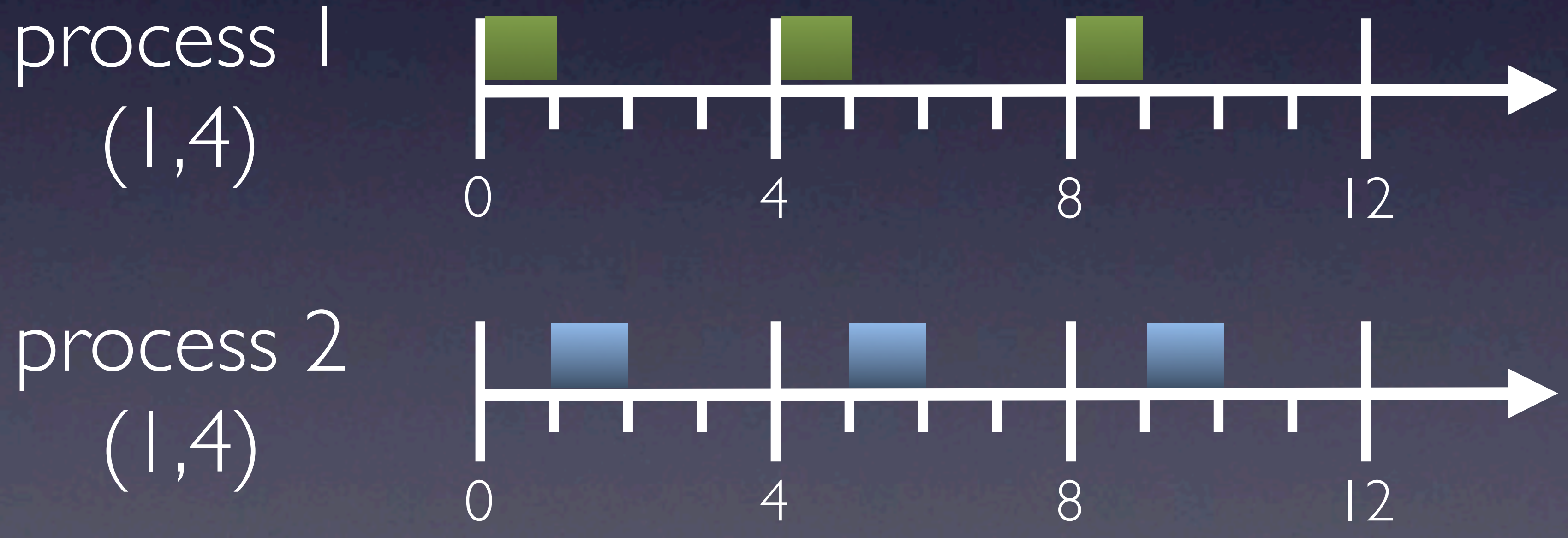
Static frequency scaling

- When CPU utilization is < 100% use idle time

# Power-Aware EDF

Static frequency scaling

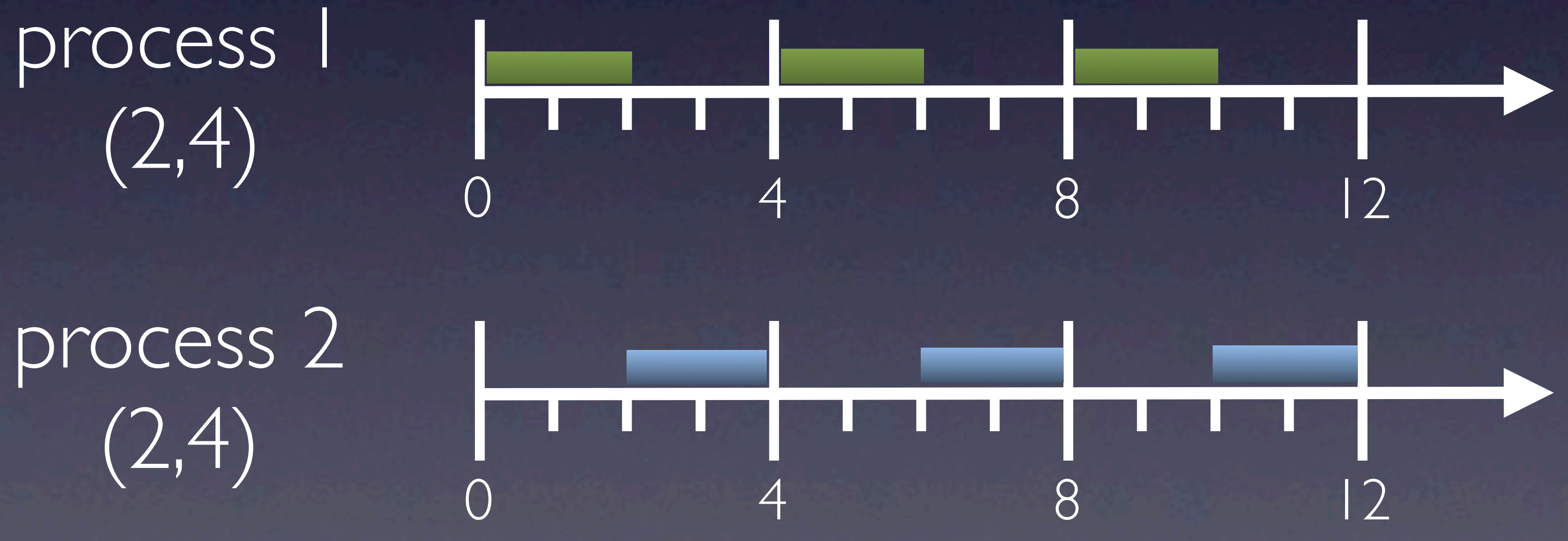
- When CPU utilization is < 100% use idle time



# Power-Aware EDF

Static frequency scaling

When CPU utilization is < 100% use idle time



process 1
(2,4)

process 2
(2,4)

# Power-Aware EDF

# Power-Aware EDF

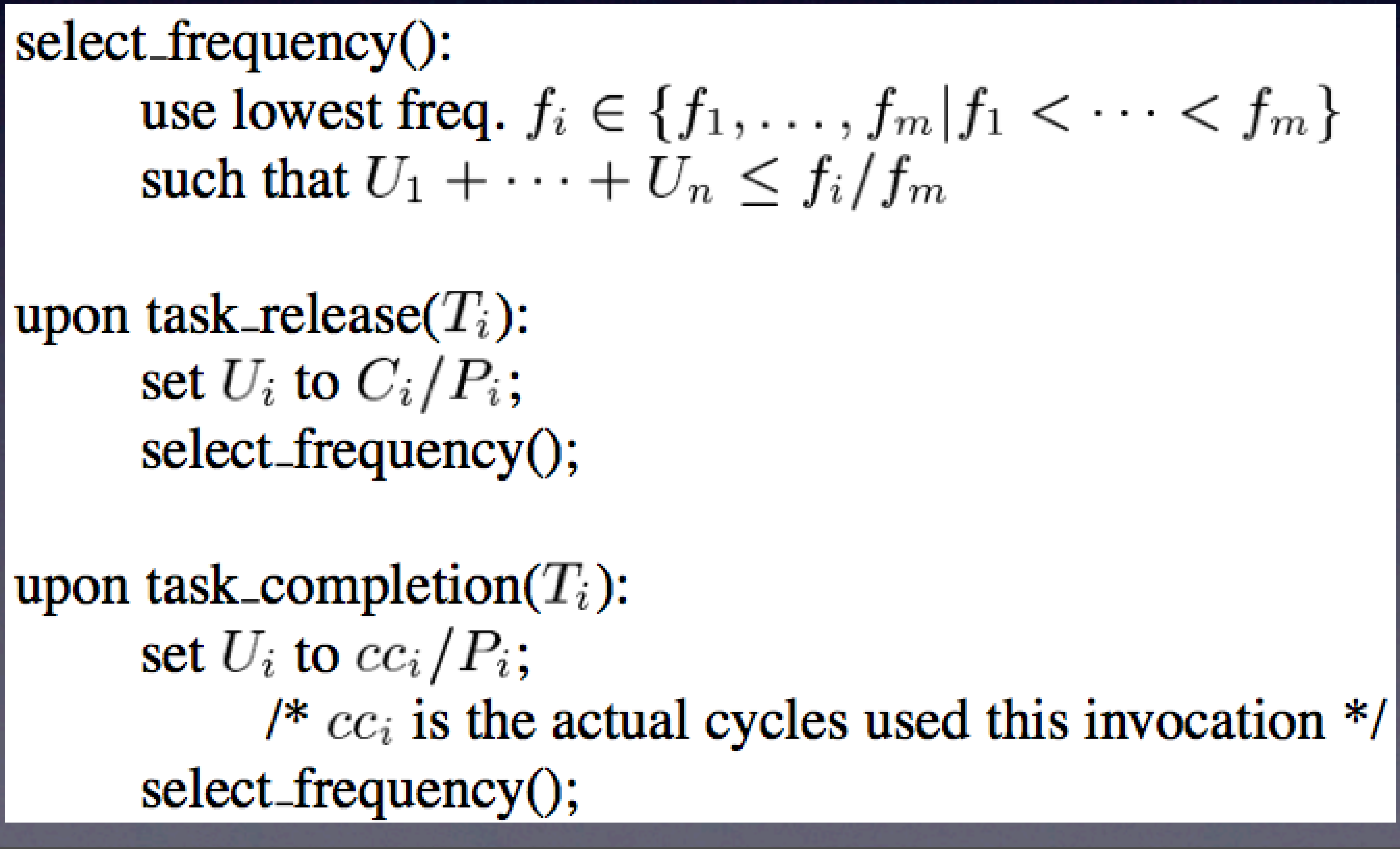
Dynamic frequency scaling

- Processes use much less than their WCET in general

# Power-Aware EDF

Dynamic frequency scaling

- Processes use much less than their WCET in general

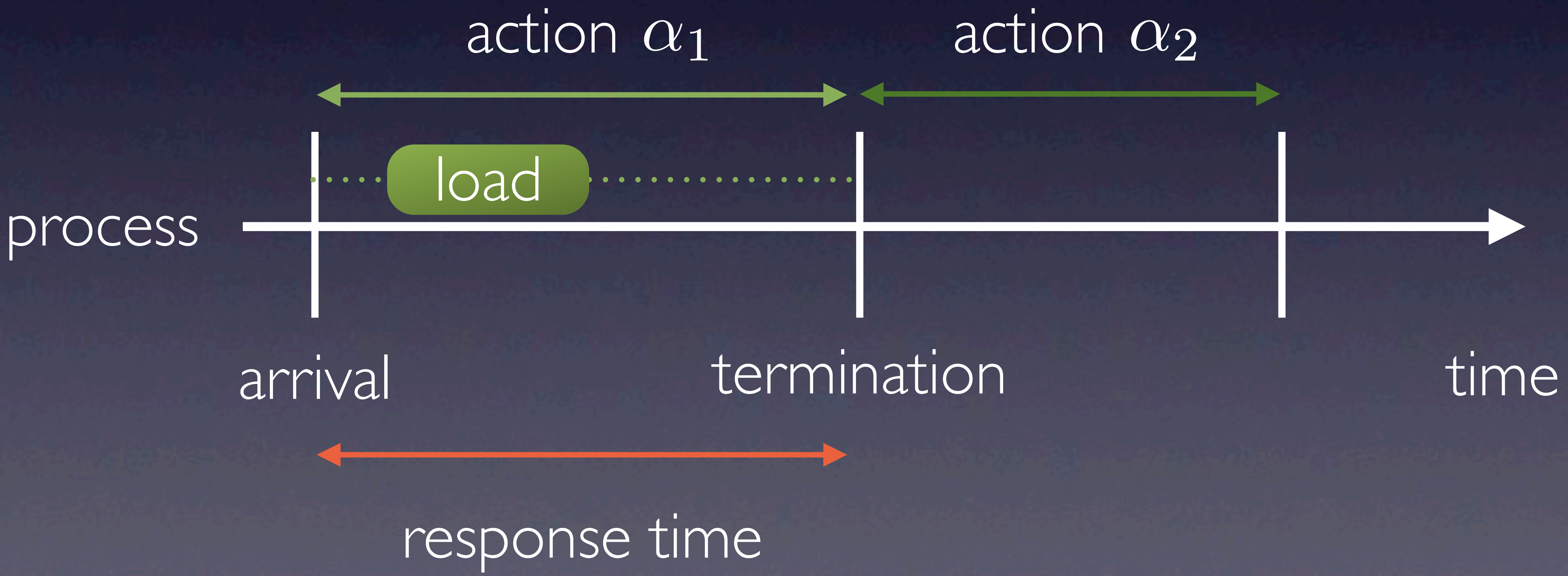


```
select_frequency():
    use lowest freq. $f_i \in \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$
    such that $U_1 + \cdots + U_n \leq f_i / f_m$

upon task_release($T_i$):
    set $U_i$ to $C_i / P_i$;
    select_frequency();

upon task_completion($T_i$):
    set $U_i$ to $cc_i / P_i$;
        /* $cc_i$ is the actual cycles used this invocation */
    select_frequency();
```

# Power-Aware VBS

Maintain VBS properties (temporal isolation, bounds)

We cannot use early completion (different process model)



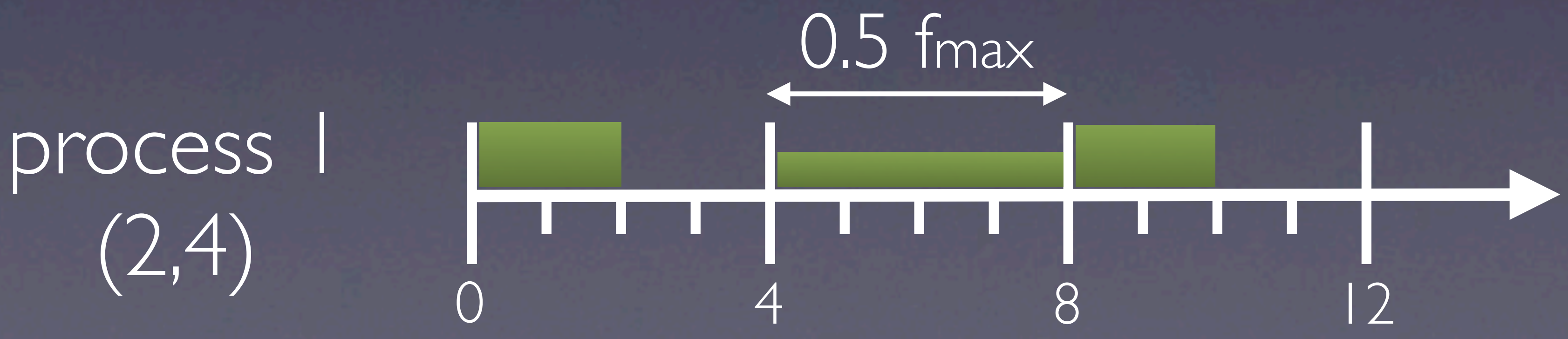
# Power-Aware VBS

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

$$U_c \cdot f_{max}$$

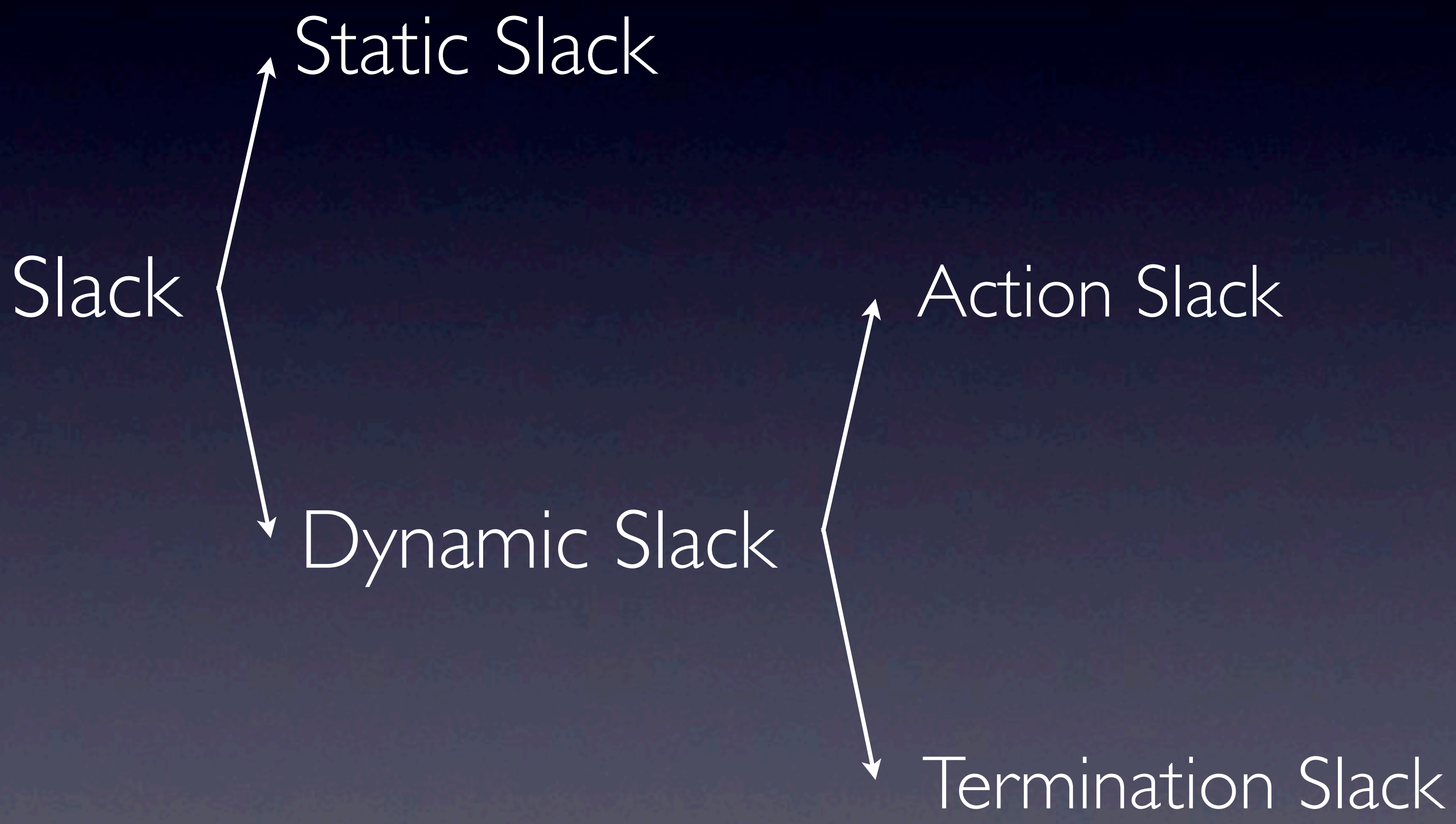
current total utilization of all released tasks in the considered interval of time between two releases



# Frequency-scaling VBS

Slack

# Frequency-scaling VBS



Slack

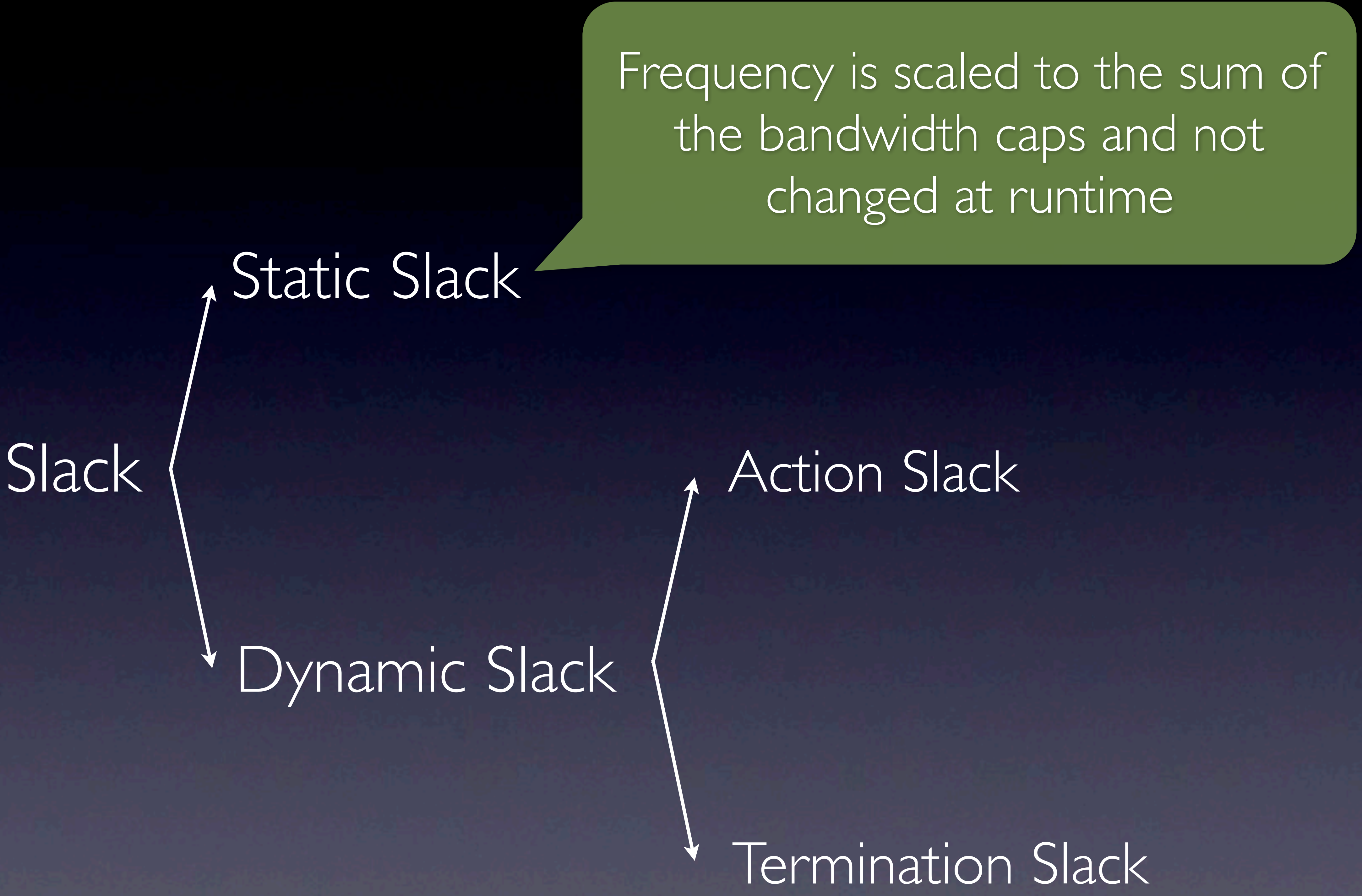
Static Slack

Dynamic Slack

Action Slack

Termination Slack

# Frequency-scaling VBS



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

Slack

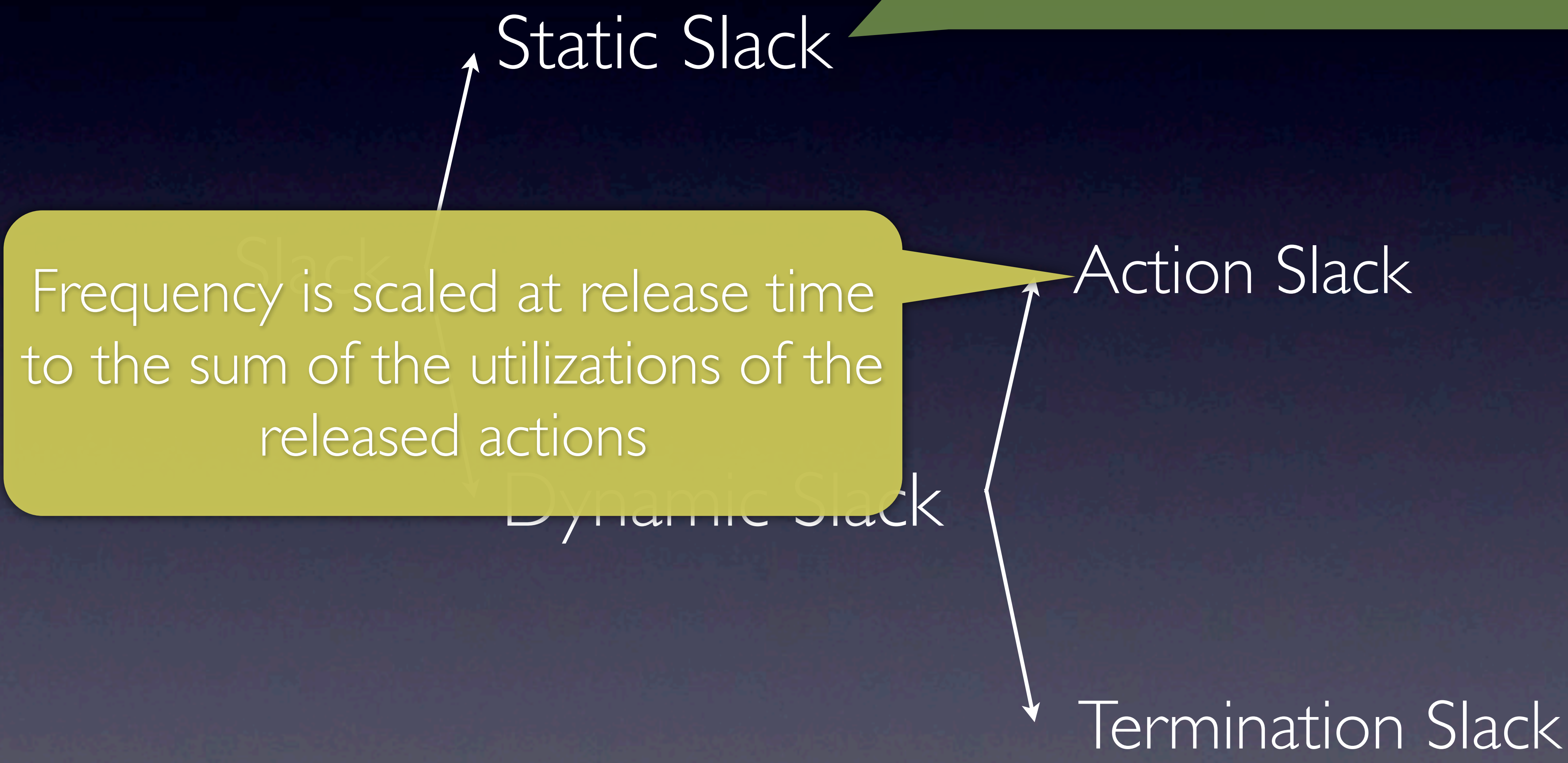
Static Slack

Dynamic Slack

Action Slack

Termination Slack

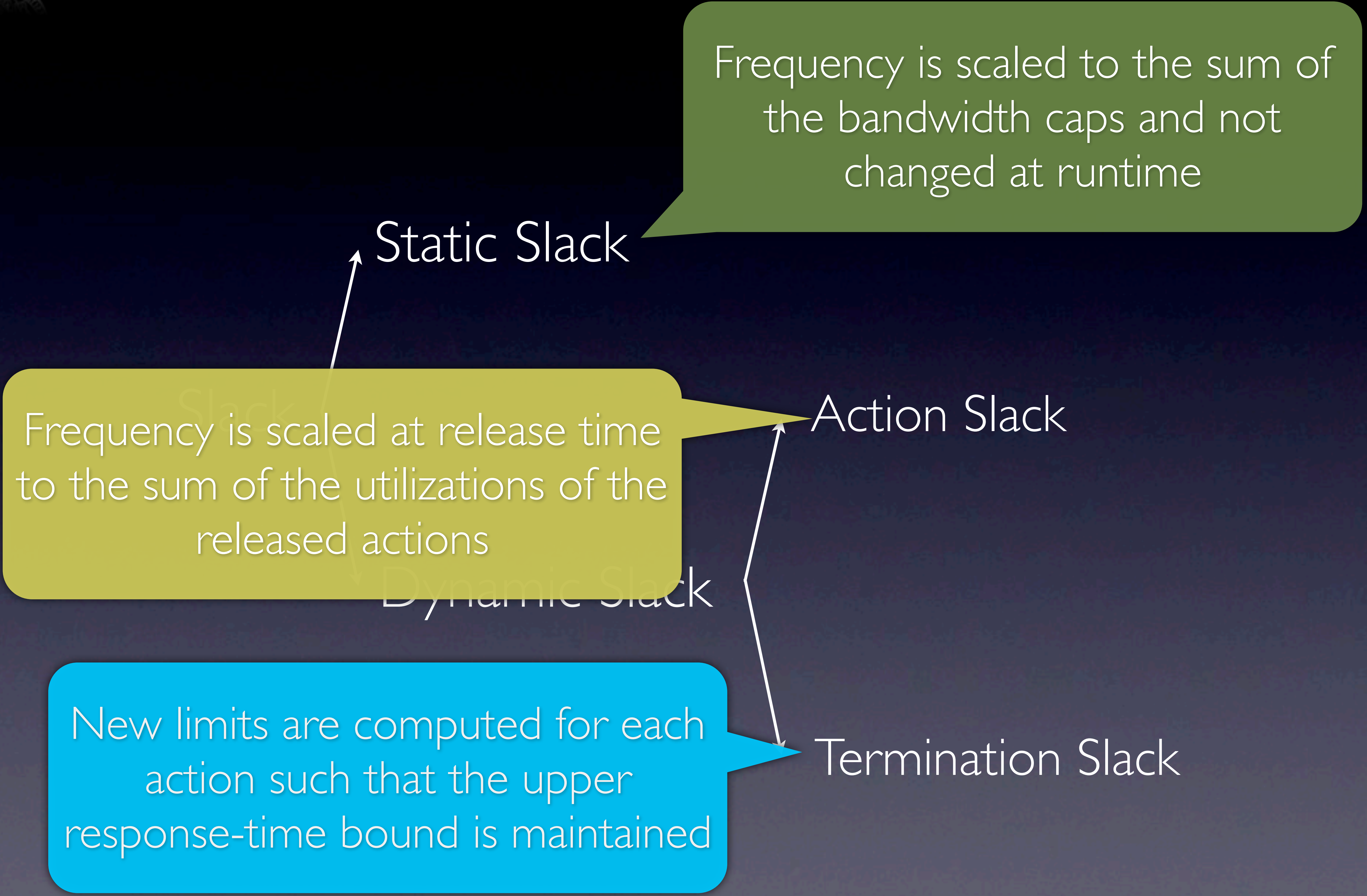
# Frequency-scaling VBS



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

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

# Frequency-scaling VBS



Slack

Static Slack

Dynamic Slack

Action Slack

Termination Slack

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

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

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

# 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

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

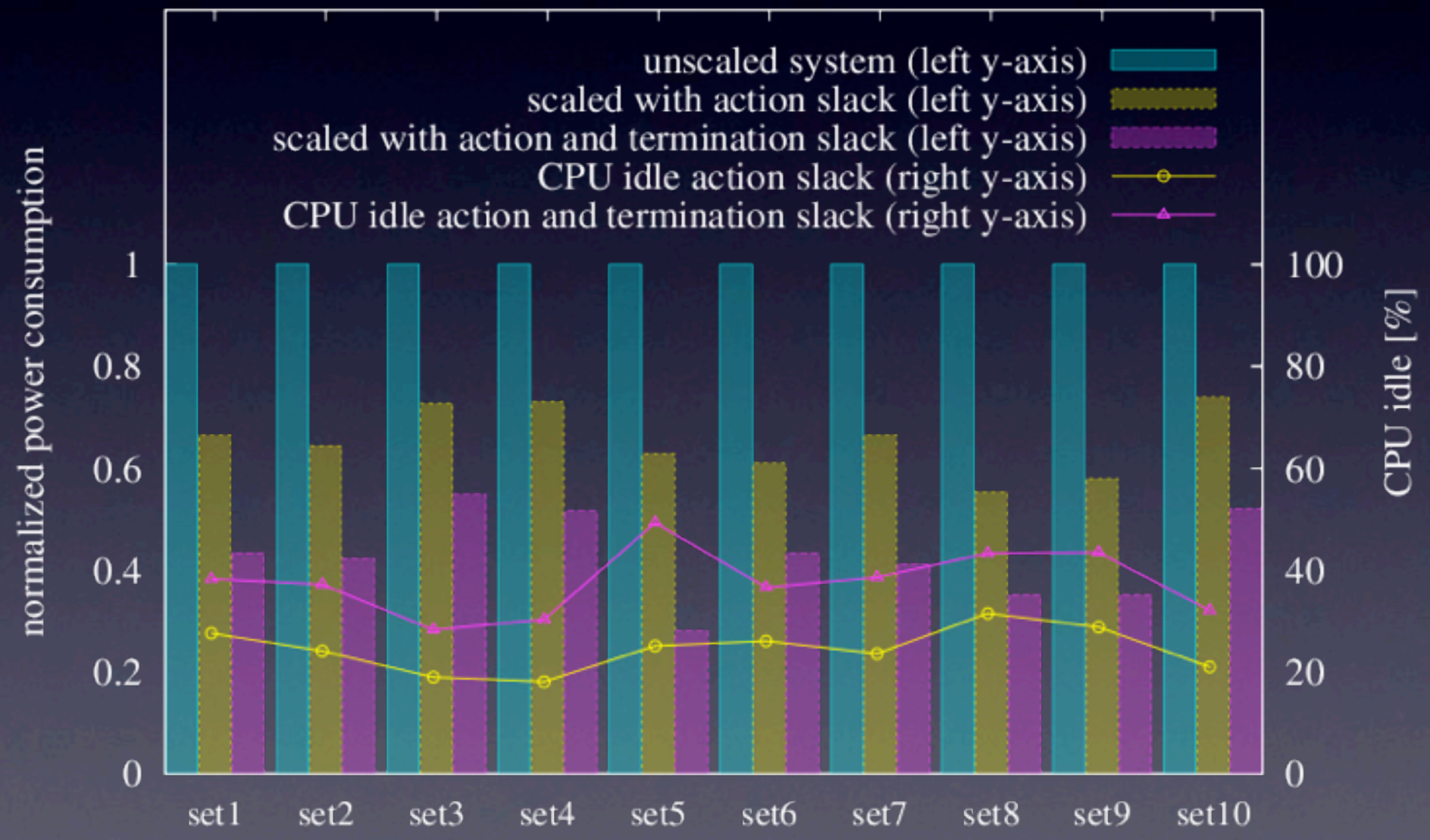
Termination slack

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

Termination and action slack can be used separately or together

# Power-Aware VBS

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



# 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

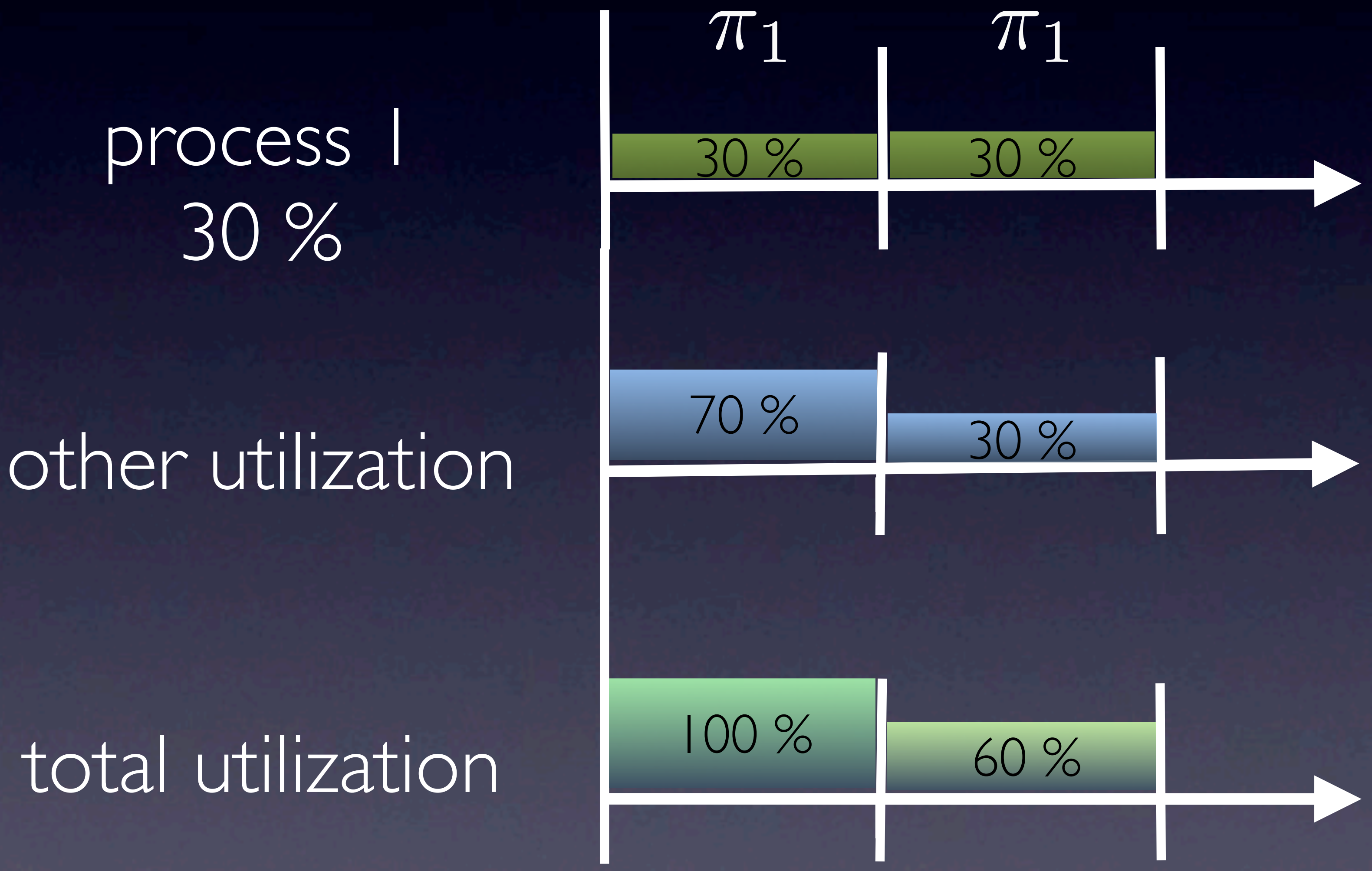
May help to handle:

more complex power models

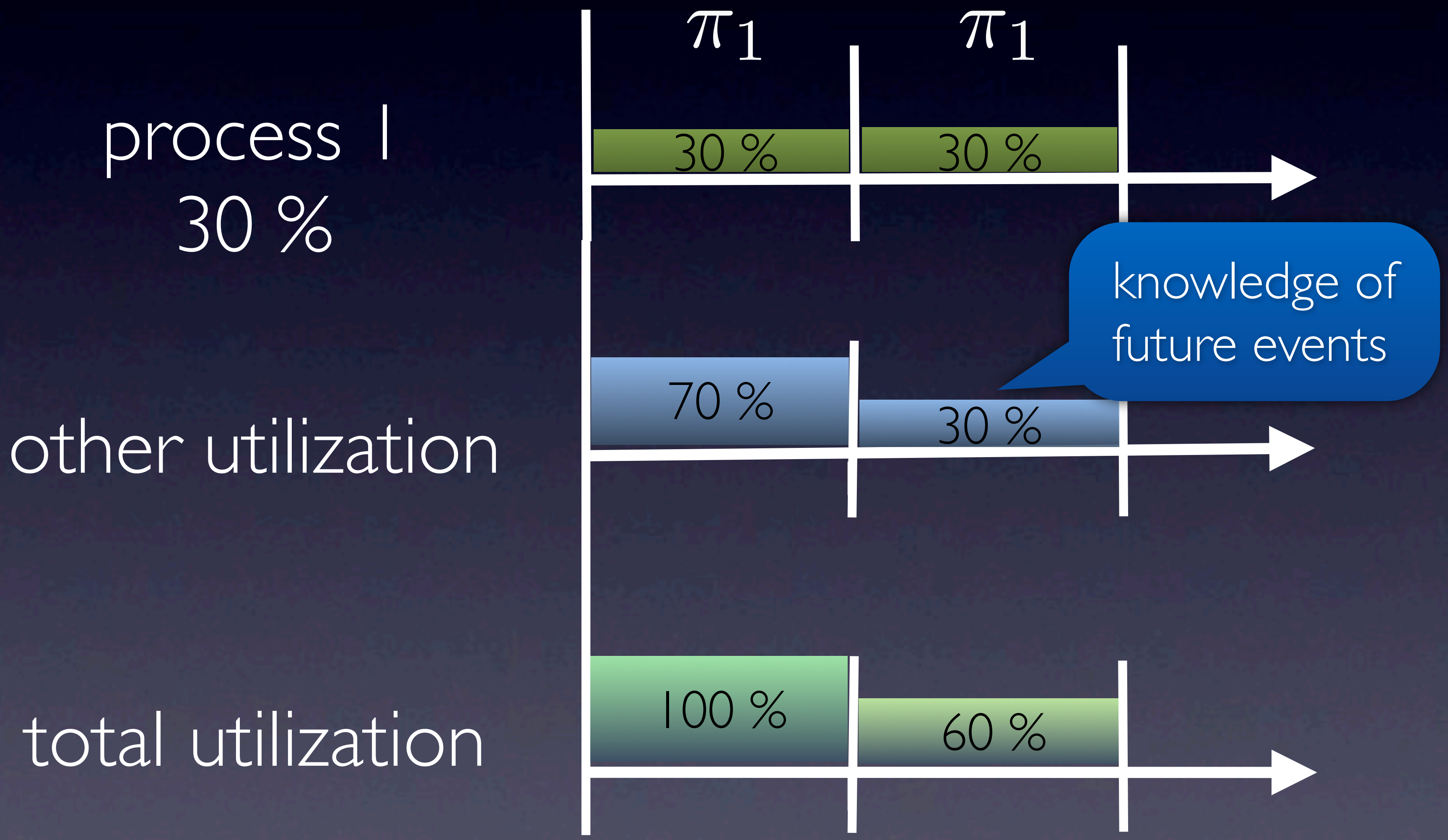
frequency switching cost (time and power)

- time overhead included using overhead accounting

# Look-ahead FS-VBS



# Look-ahead FS-VBS



process 1
30 %

other utilization

total utilization

$\pi_1$ $\pi_1$

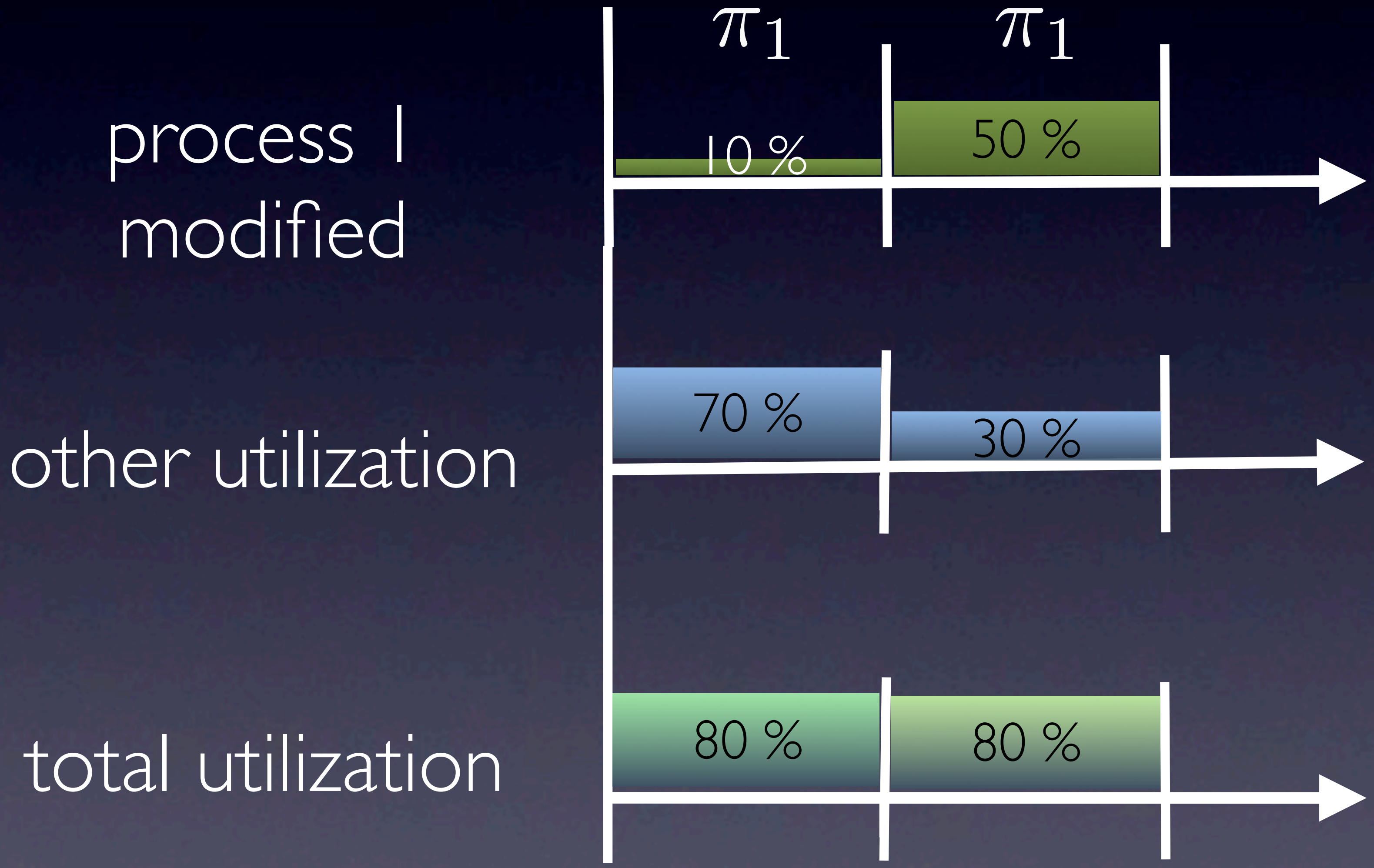
30 % 30 %

70 % 30 %

100 % 60 %

knowledge of future events

# Look-ahead FS-VBS



$\pi_1$ $\pi_1$

process 1 modified

10 % 50 %

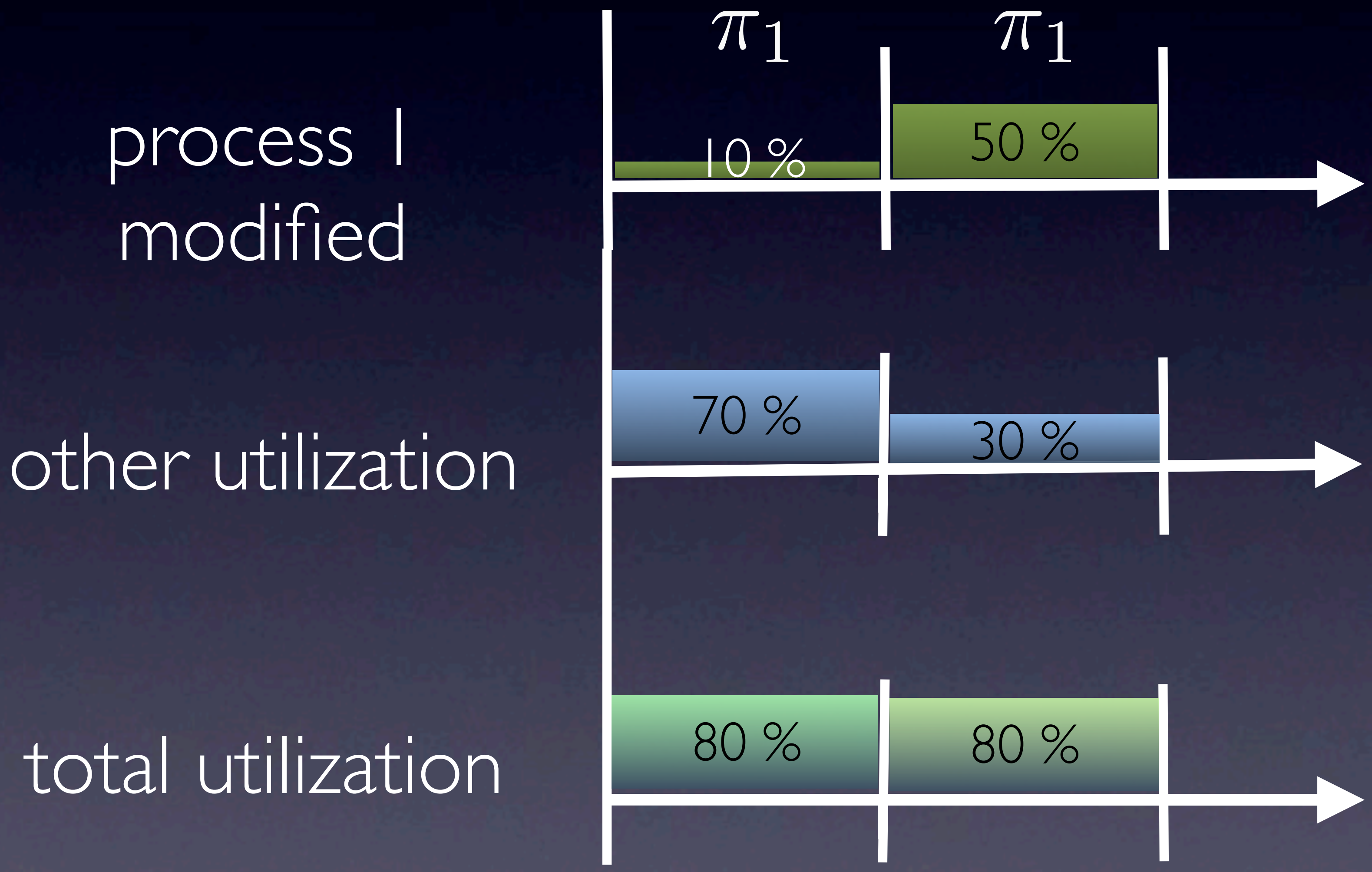
other utilization

70 % 30 %

total utilization

80 % 80 %

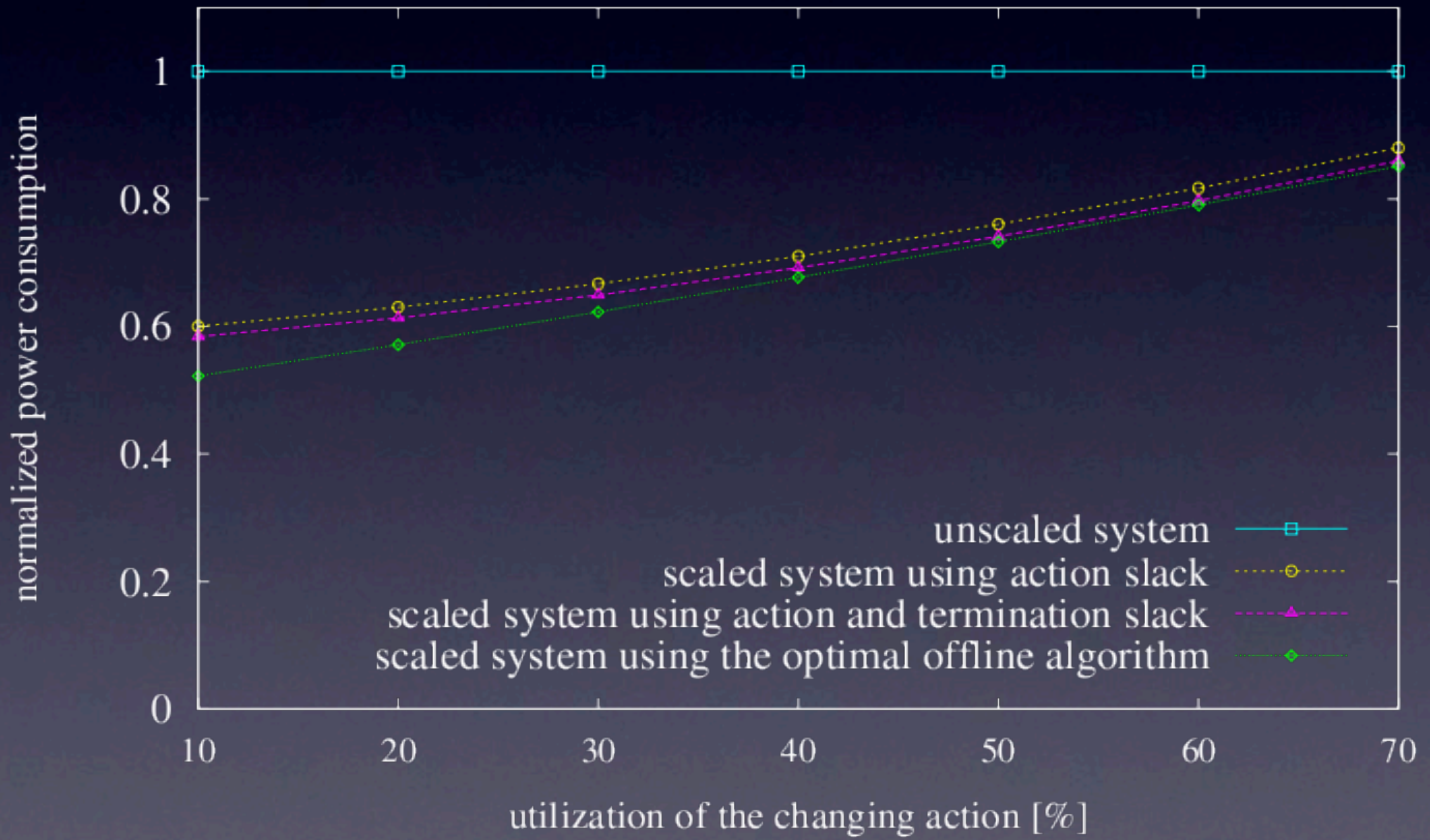
# Look-ahead FS-VBS



actual improvement depends
on the power model

# Look-ahead FS-VBS

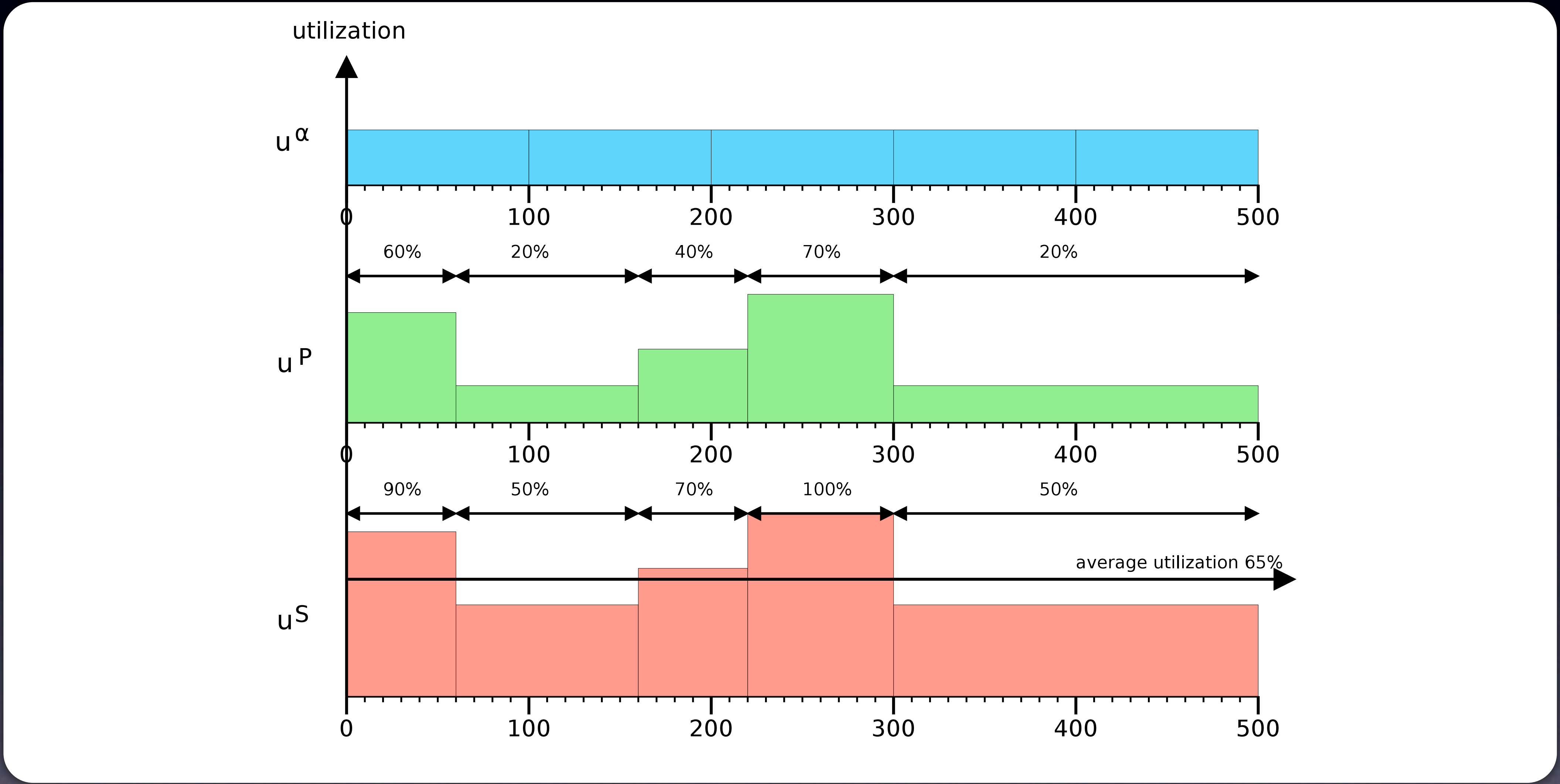
Assuming a simple power model ($P \propto V^2$)



# Look-ahead online FS-VBS

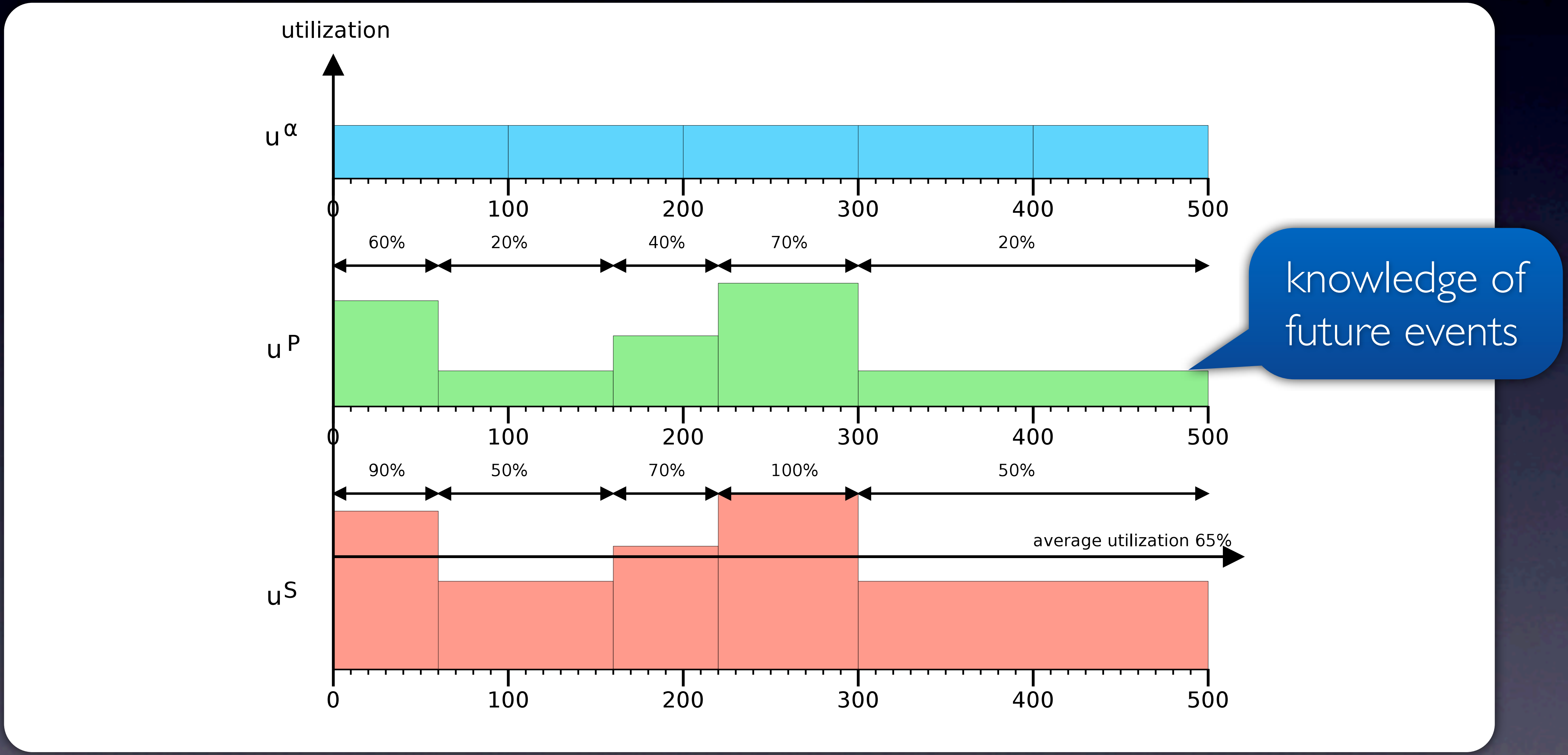
# Look-ahead online FS-VBS

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



# Look-ahead online FS-VBS

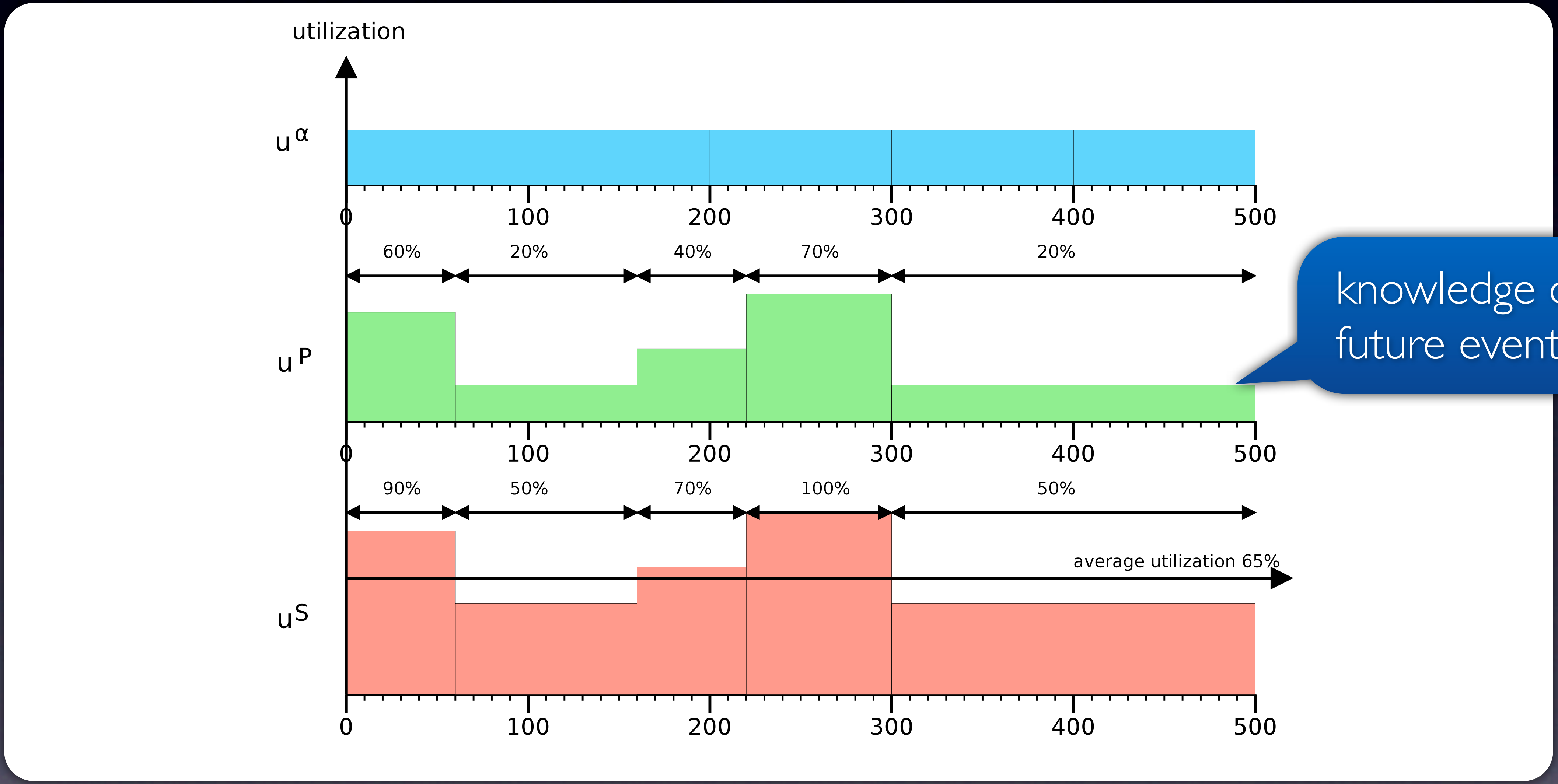
## Assume a simple power model ($P \propto V^2$)



knowledge of future events

# Look-ahead online FS-VBS

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

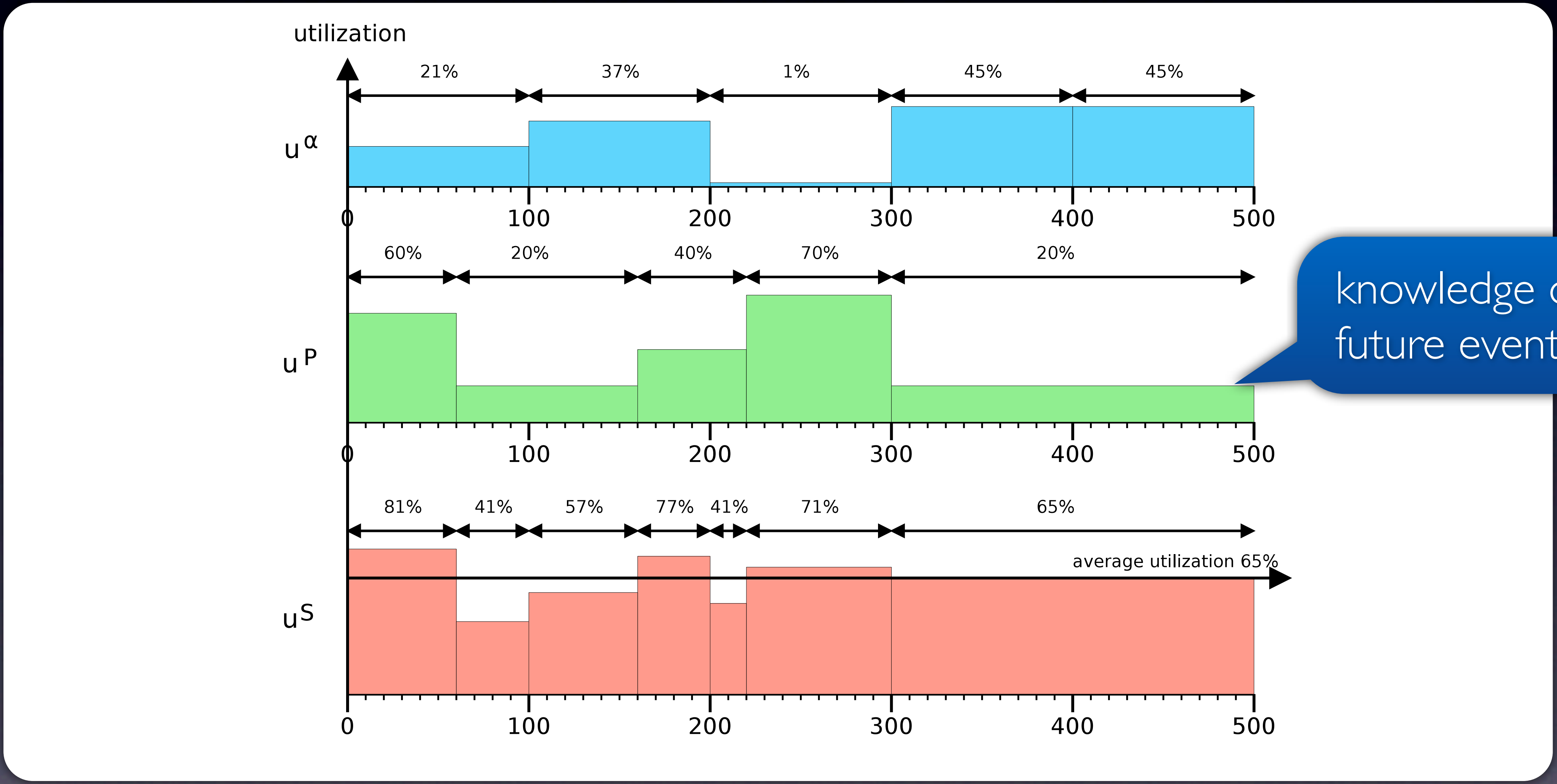


knowledge of future events

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

# Future work

# Future work

Important aspects: time, space and power isolation

# Future work

Important aspects: time, space and power isolation

Temporal isolation through VBS

# Future work

Important aspects: time, space and power isolation

Temporal isolation through VBS

Spatial isolation through memory management

# Future work

Important aspects: time, space and power isolation

Temporal isolation through VBS

Spatial isolation through memory management

What about power isolation? Is power consumption compositional?

# Future work

Important aspects: time, space and power isolation

Temporal isolation through VBS

Spatial isolation through memory management

What about power isolation? Is power consumption compositional?

Yes, if there is no frequency scaling, and if scheduling and context switching cost is accounted for

# Future work

Important aspects: time, space and power isolation

Temporal isolation through VBS

Spatial isolation through memory management

What about power isolation? Is power consumption compositional?

Yes, if there is no frequency scaling, and if scheduling and context switching cost is accounted for

Time and power isolation with frequency scaling?
Problem: non-linear relationship of power consumption and processor frequency