TASK-AWARE VIRTUAL MACHINE SCHEDULING FOR I/O PERFORMANCE

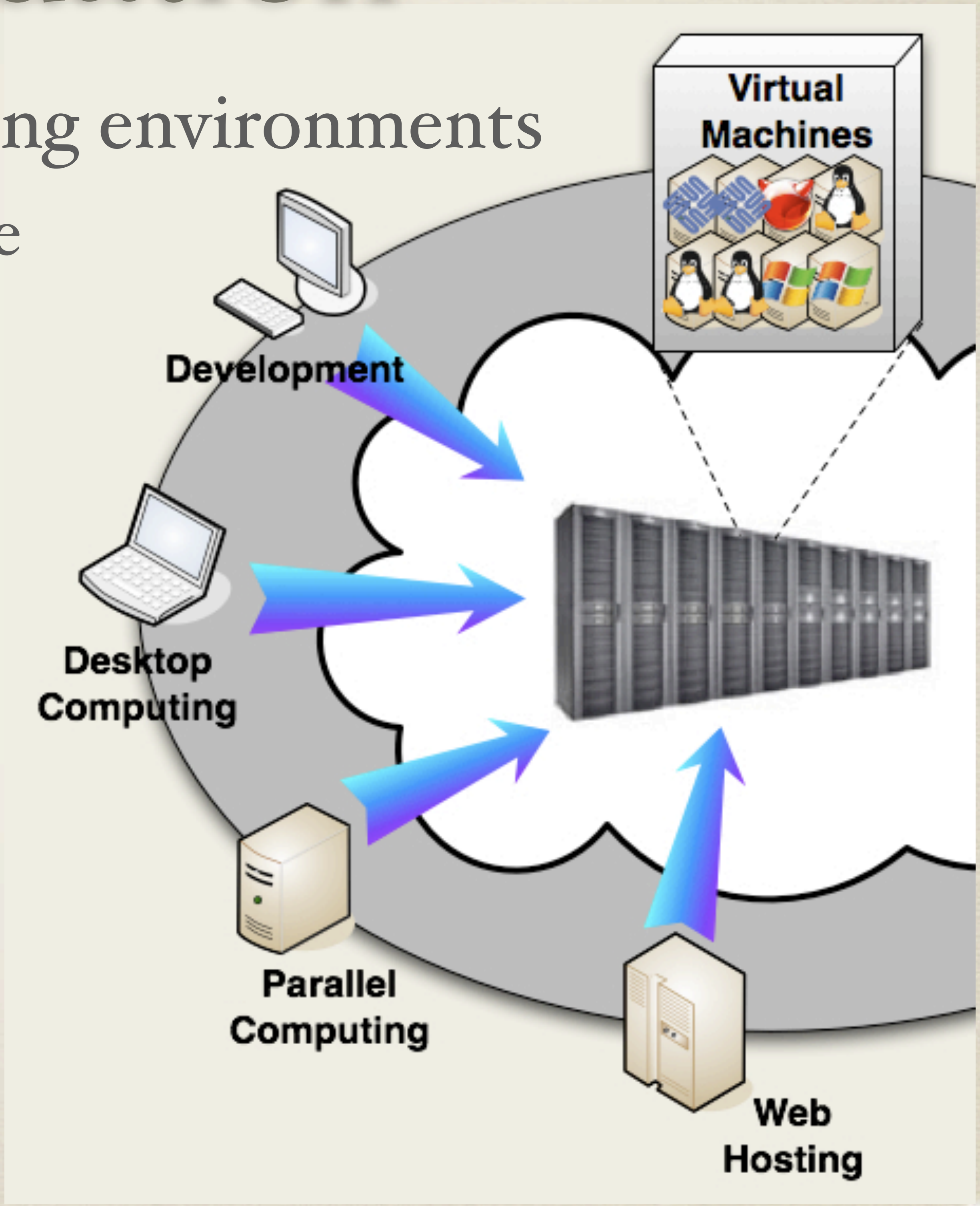
Hwanju Kim, Hyeontaek Lim, Jinkyu Jeong, Heeseung Jo
(*Korea Advanced Institute of Science and Technology*)
Joonwon Lee (*Sungkyunkwan Univ.*)

VEE 2009 March 13

KAIST Computer Science

Virtual Machine Consolidation

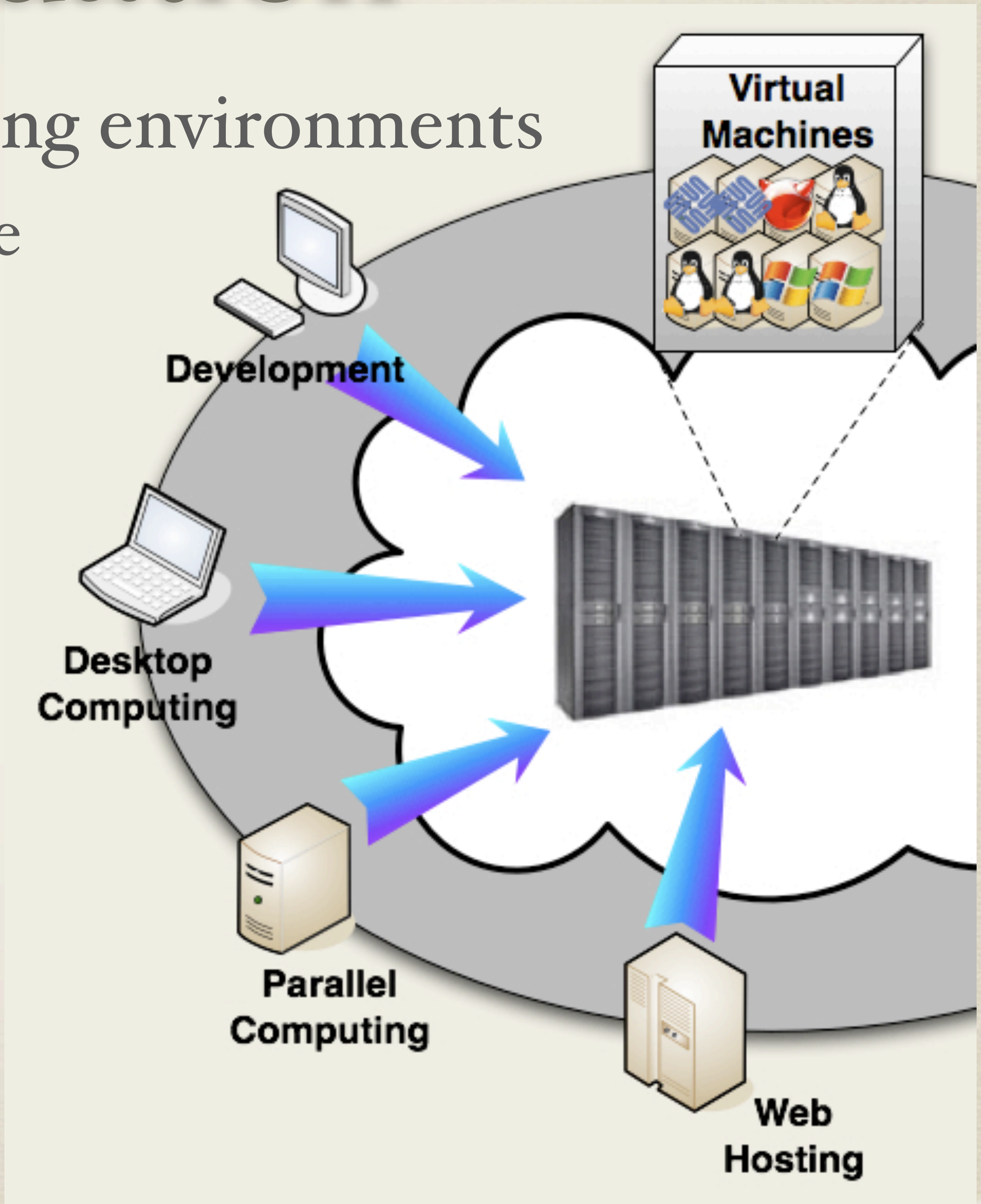
- Centralized various computing environments
  - Virtual desktop infrastructure
    - VMware, Sun, HP, MS
  - Cloud computing
    - Amazon EC2



Virtual Machine Consolidation

- Centralized various computing environments
  - Virtual desktop infrastructure
    - VMware, Sun, HP, MS
  - Cloud computing
    - Amazon EC2

Unpredictable workloads due to the diversity



Virtual Machine Consolidation

- Performance enhancement
  - Paravirtualization
  - Hardware-assisted techniques
    - Intel VT, AMD SVM
  - Optimization

Virtual Machine Consolidation

- Performance enhancement
  - Paravirtualization
  - Hardware-assisted techniques
    - Intel VT, AMD SVM
  - Optimization

↓

High degree of consolidation

Virtual Machine Consolidation

- Performance enhancement
  - Paravirtualization
  - Hardware-assisted techniques
    - Intel VT, AMD SVM
  - Optimization

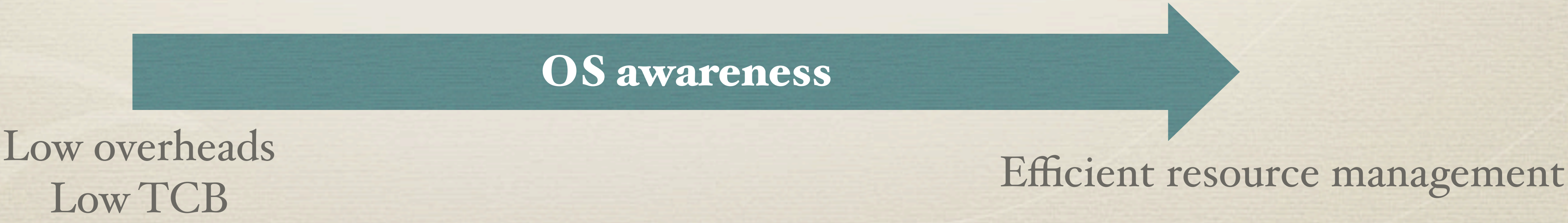
Unpredictable workloads

High degree of consolidation

Intelligent CPU management can improve the performance

Background

- A **semantic gap** between the VMM and a guest OS
  - VMM's lack of knowledge of VM internal
  - No tracking characteristics of guest-level tasks
    - Internal workload-agnostic scheduling
    - Poor decision about **"when"** to schedule a VM
  - Simple design of the VMM



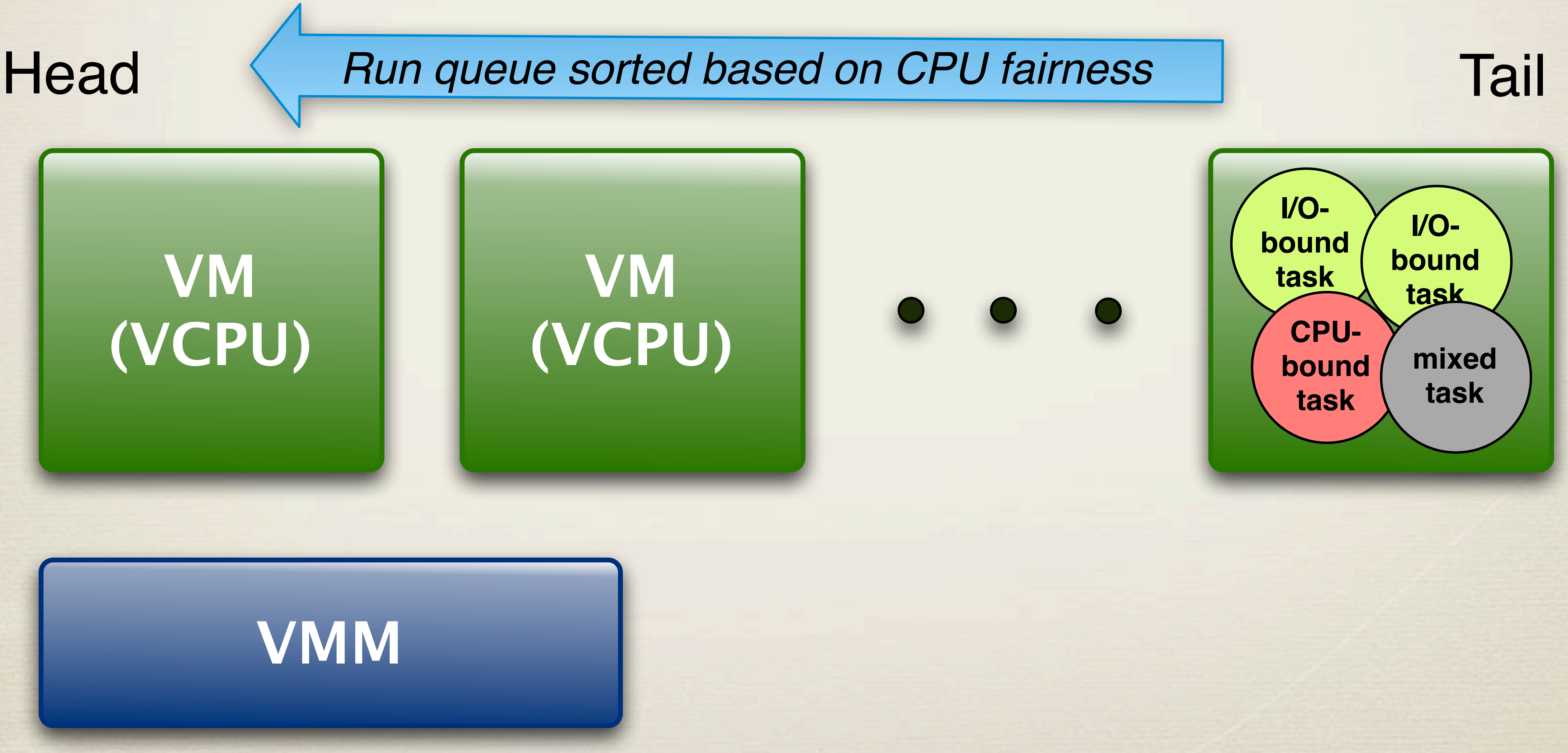
OS awareness

Low overheads
Low TCB

Efficient resource management

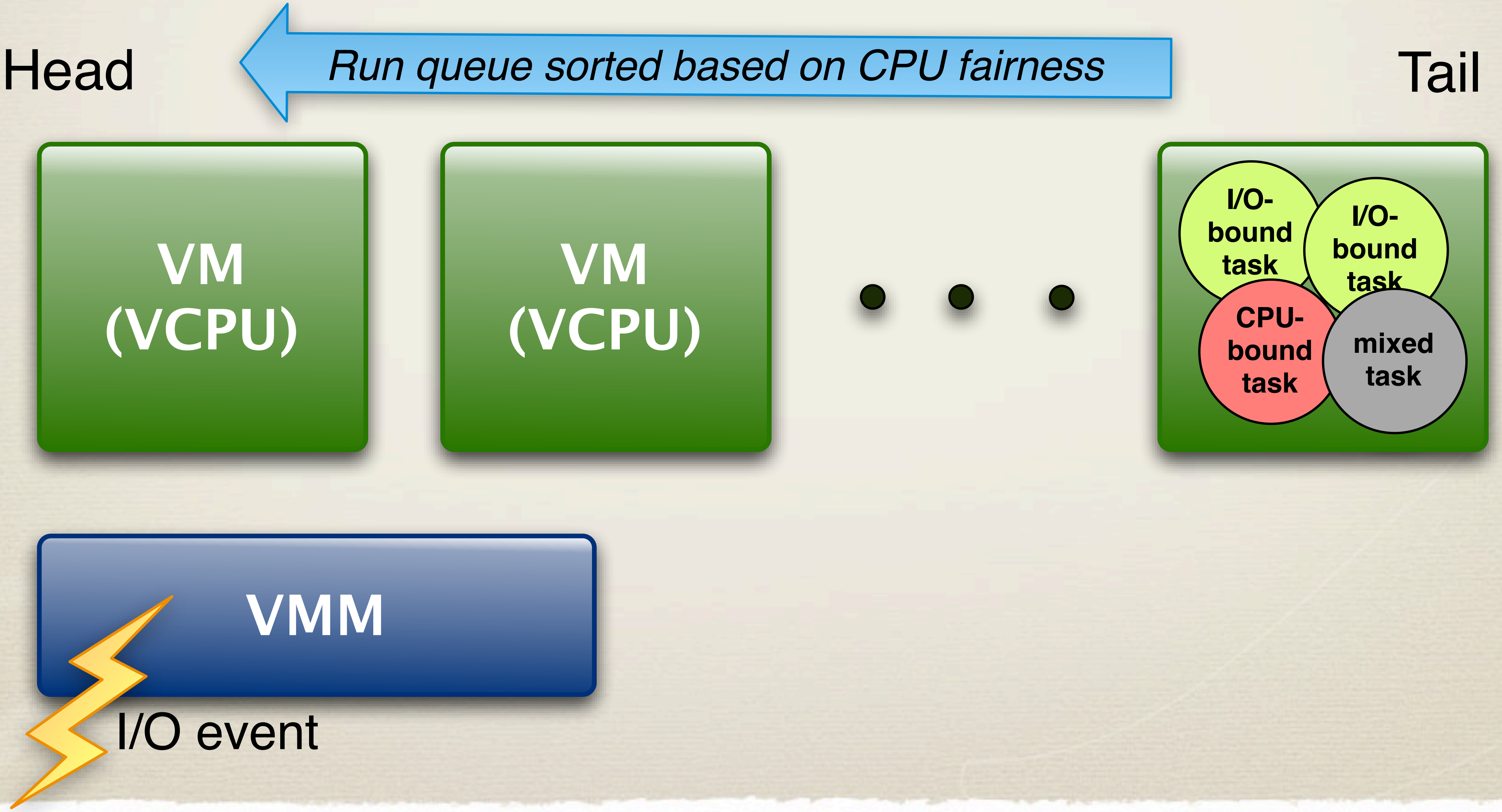
Background

- Task-unawareness leading to poor responsiveness



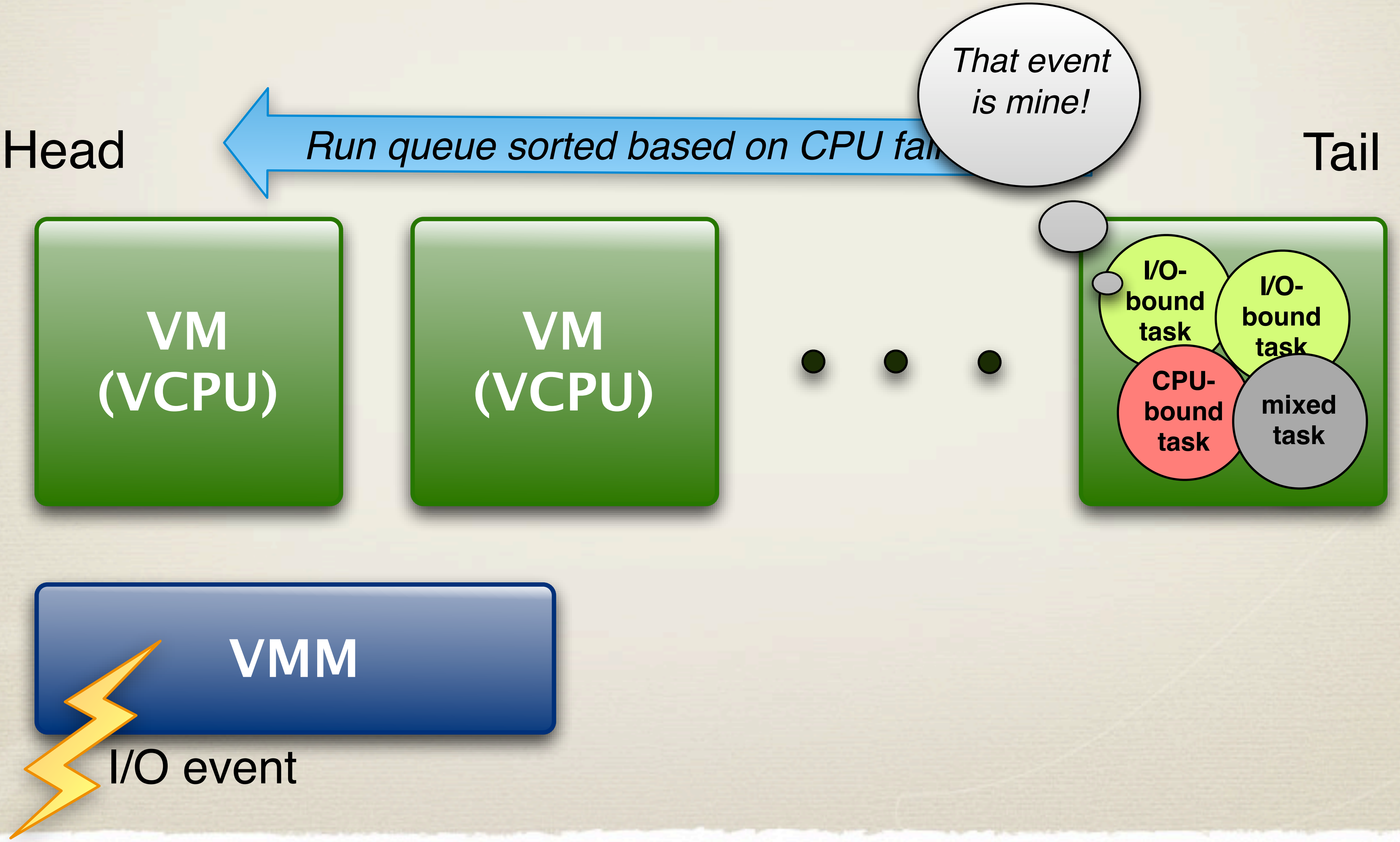
Background

- Task-unawareness leading to poor responsiveness



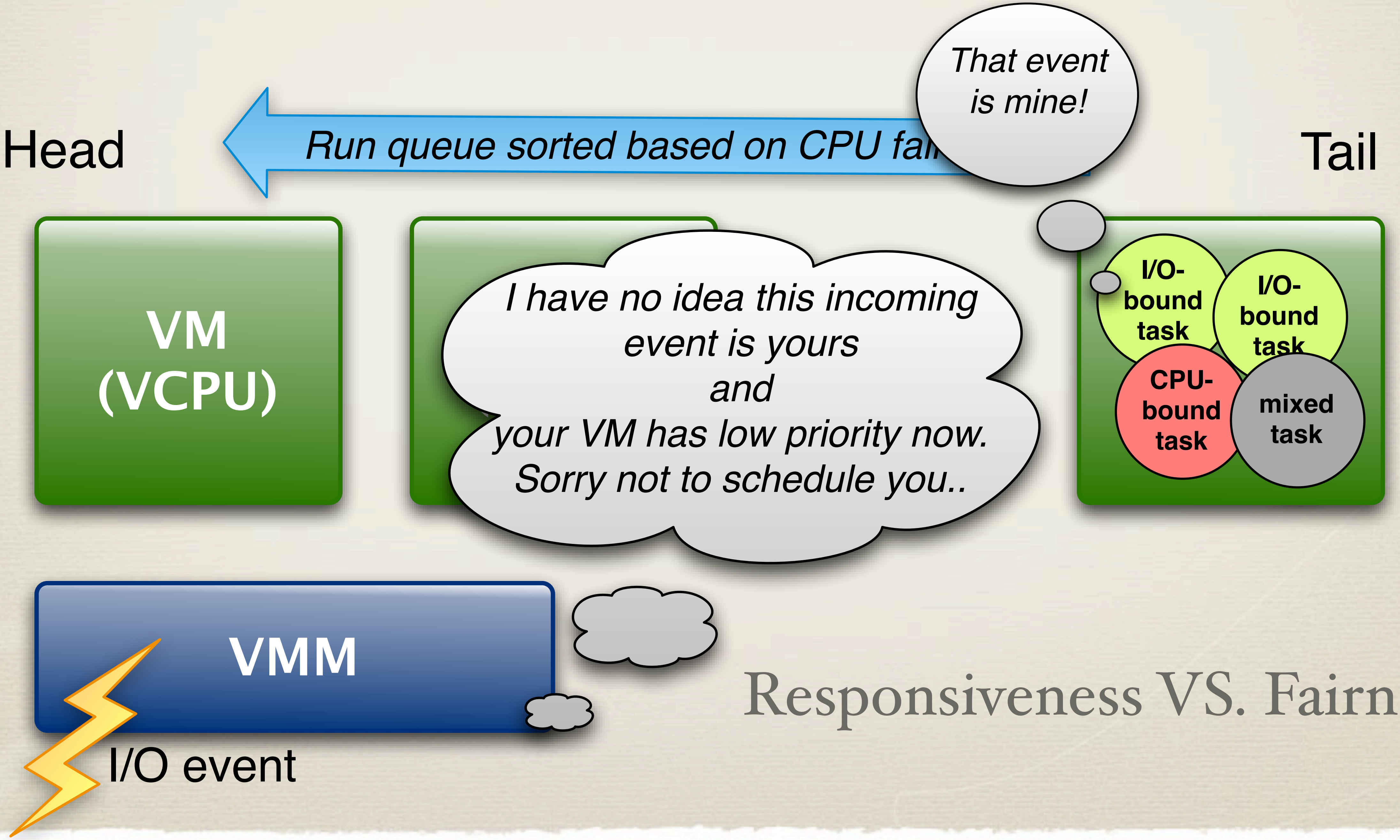
Background

- Task-unawareness leading to poor responsiveness



Background

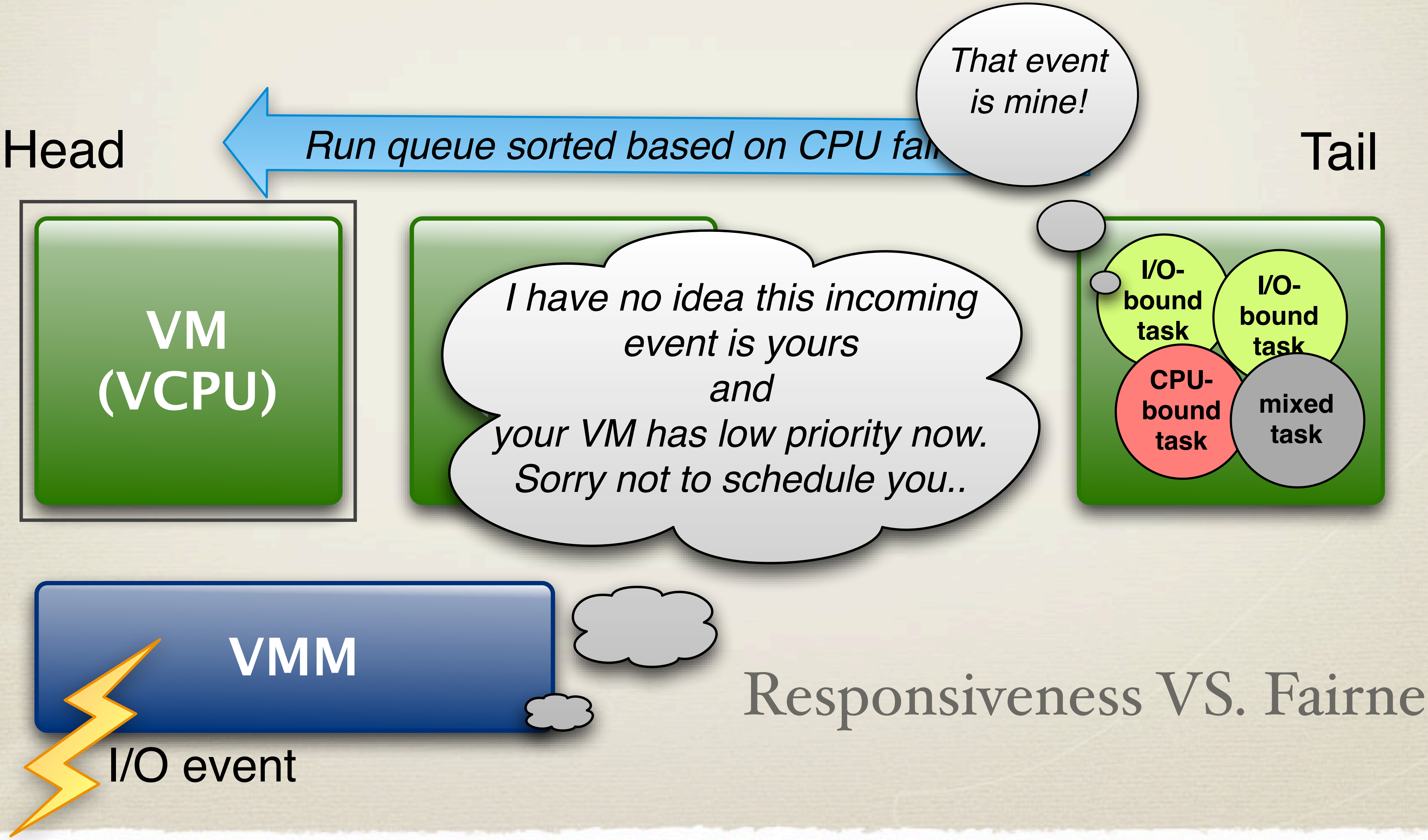
- Task-unawareness leading to poor responsiveness


That event is mine!
Head
Run queue sorted based on CPU fai
Tail
VM (VCPU)
I have no idea this incoming event is yours and your VM has low priority now. Sorry not to schedule you..
I/O-bound task
I/O-bound task
CPU-bound task
mixed task
VMM
I/O event

Responsiveness VS. Fairness

Background

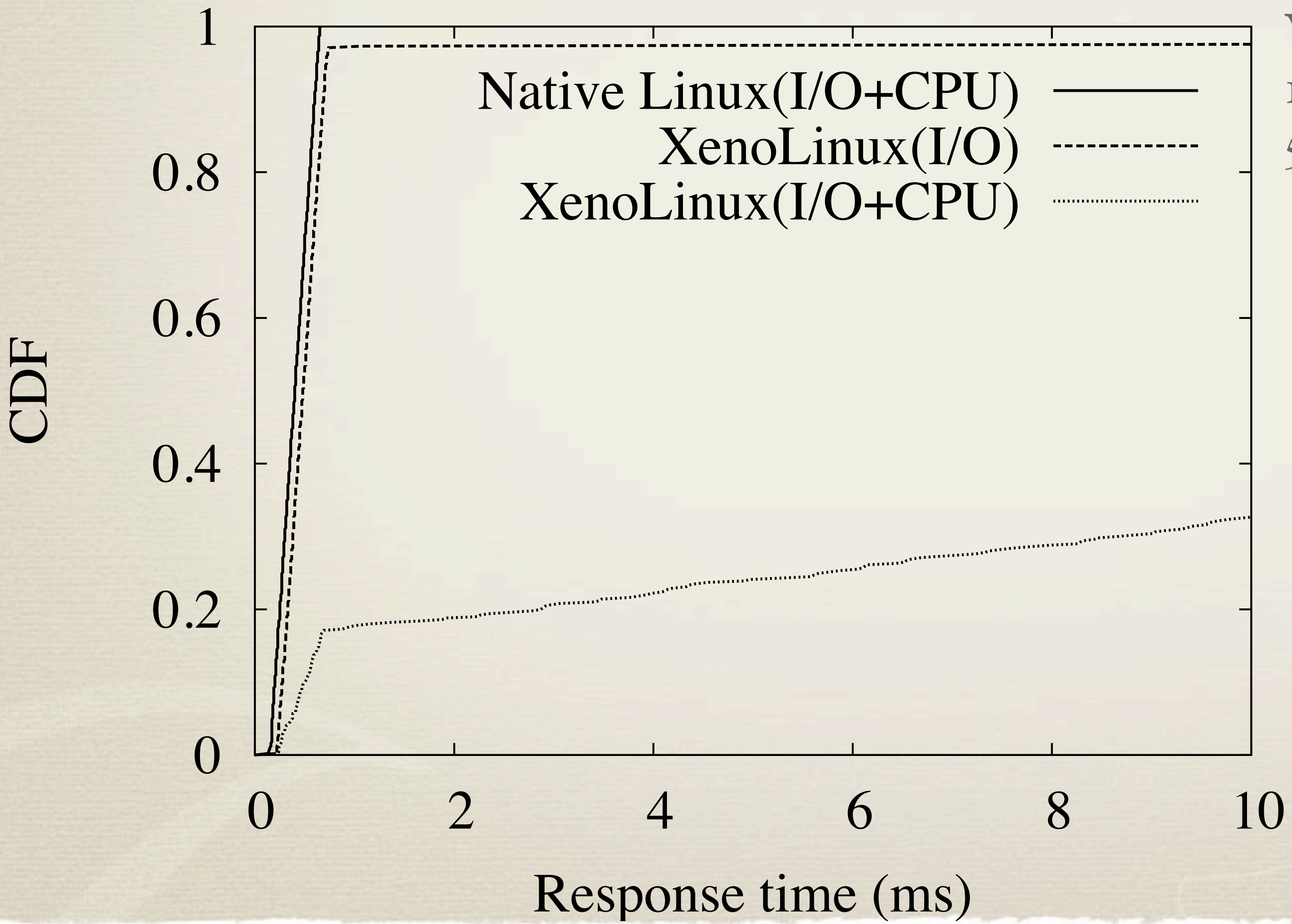
- Task-unawareness leading to poor responsiveness


That event is mine!
Head
Run queue sorted based on CPU fai
Tail
VM (VCPU)
I have no idea this incoming event is yours and your VM has low priority now. Sorry not to schedule you..
I/O-bound task
I/O-bound task
CPU-bound task
mixed task
VMM
I/O event

Responsiveness VS. Fairness

Background

- The worst case example for 6 domains consolidated



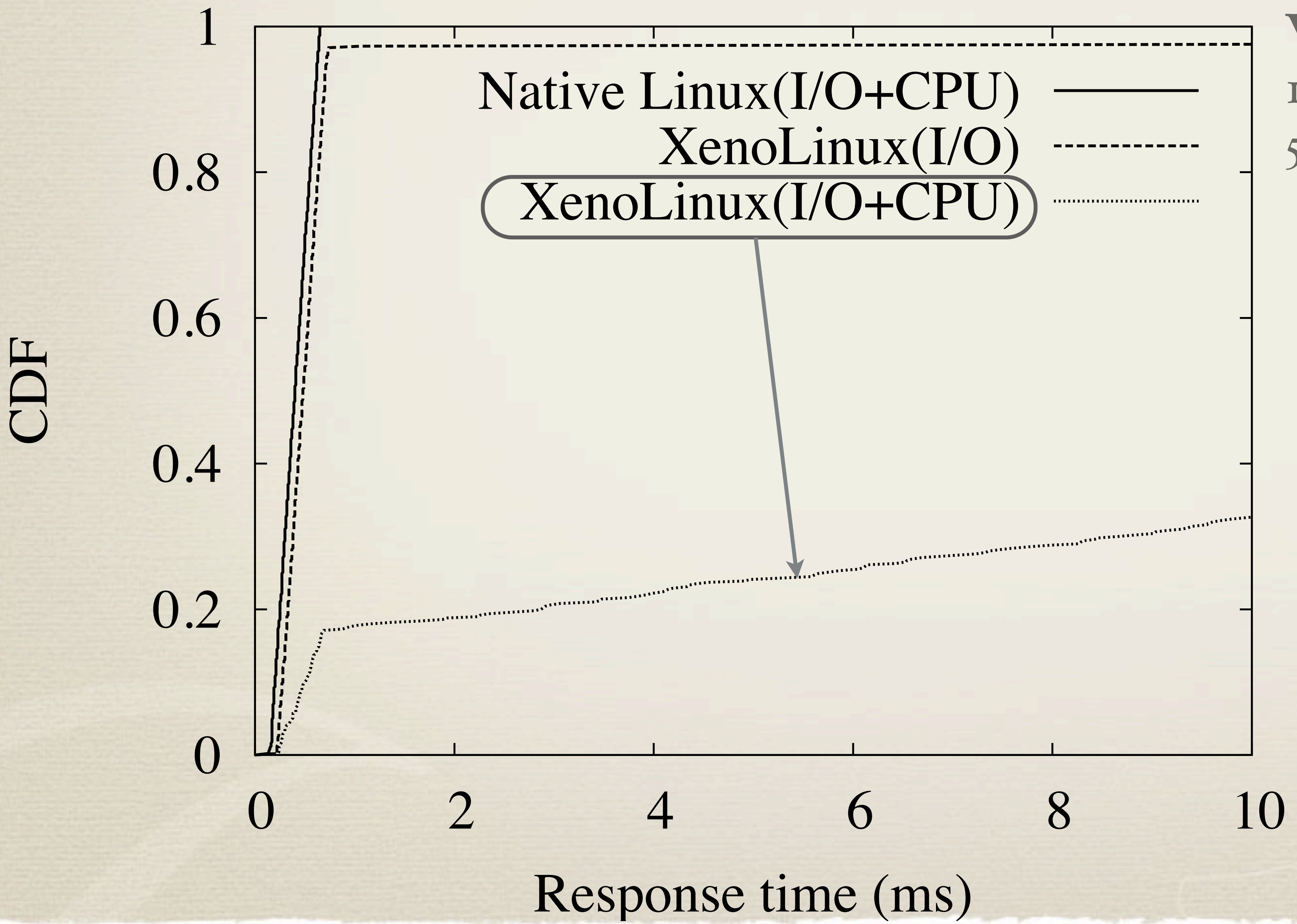
Workloads

1 dom: Server & CPU-bound task

5doms: CPU-bound task

Background

- The worst case example for 6 domains consolidated



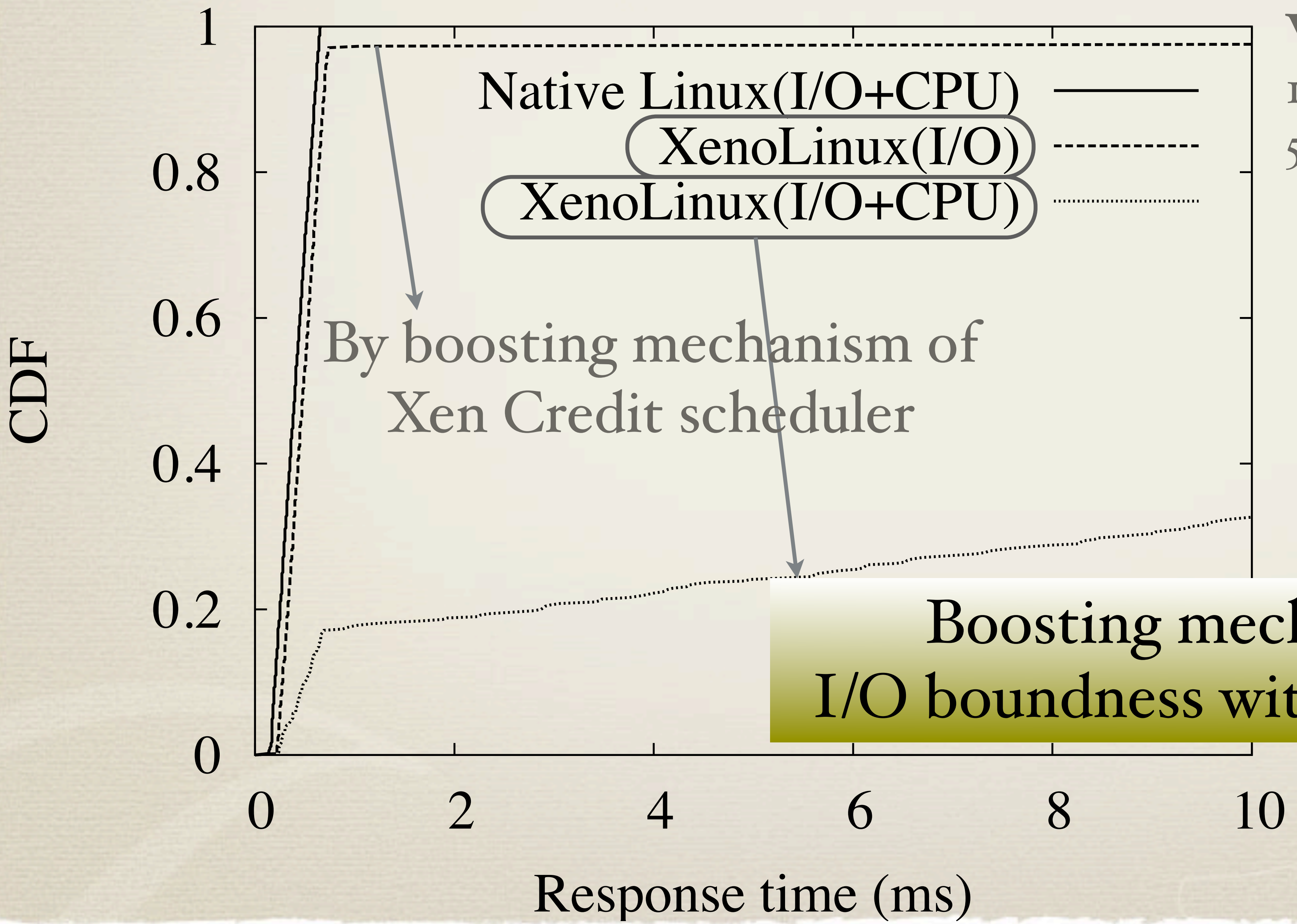
Workloads

1 dom: Server & CPU-bound task

5doms: CPU-bound task

Background

- The worst case example for 6 domains consolidated



By boosting mechanism of Xen Credit scheduler

Workloads

1 dom: Server & CPU-bound task

5doms: CPU-bound task

Boosting mechanism realizes I/O boundness with only VCPU-level

Main Goals

- Improve responsiveness of an I/O-bound task
  - Priority boosting with task-level granularity
    - “**Partial boosting**”
- CPU fairness guarantee
- Transparency
- Low management overheads

Issues

- **How to identify an I/O-bound task**

- **How to know an incoming event is for the I/O-bound task**

Approach

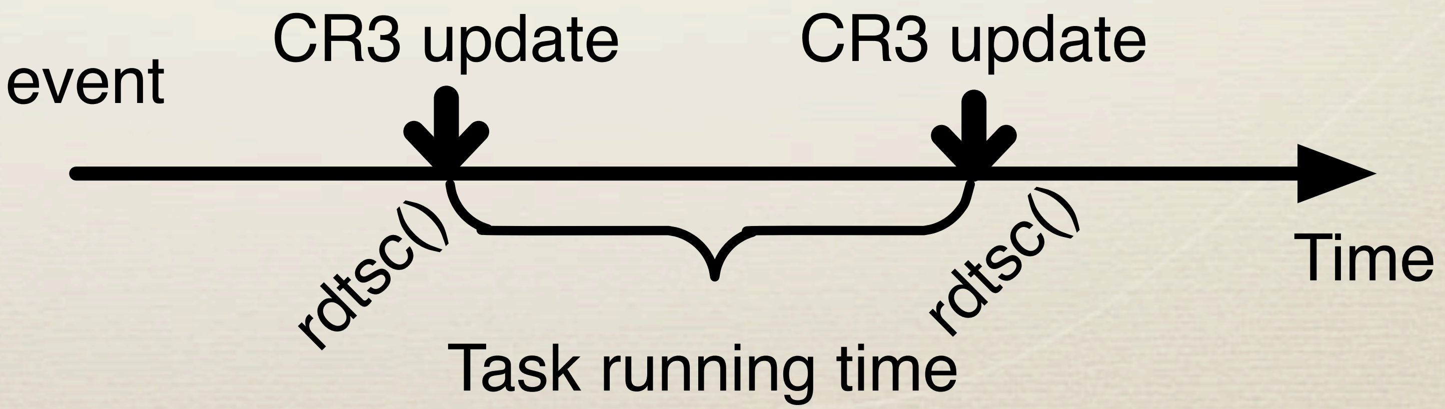
- Non-intrusive approach
  - No guest OS modification
    - No explicit interface to inform I/O-bound task and event data
  - Pros.
    - No additional engineering cost for different OSes
    - Strong trustworthiness
  - Cons.
    - False decision

HOW TO IDENTIFY I/O-BOUND TASKS

Tracking I/O-bound Tasks

- Observable information at the VMM
  - Task switching
    - Monitoring address space changes (Antfarm USENIX'06)
  - CPU time usage
    - Running time of a task

Example (x86)

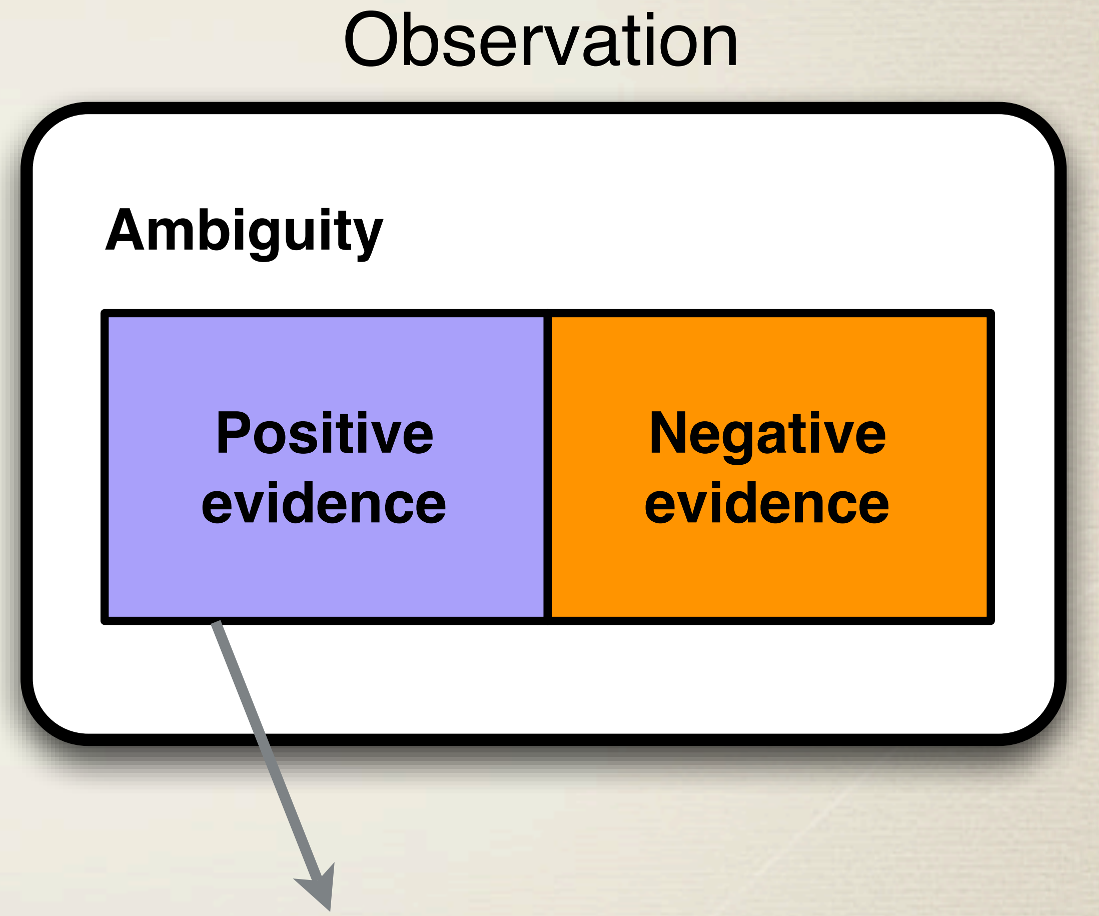


Tracking I/O-bound Tasks

- Inference based on common ***gray-box*** knowledge
    - Kernel policy to improve responsiveness of I/O-bound tasks
        - An I/O-bound task is preemptively scheduled in response to its incoming event
    - Characteristic of I/O-bound tasks
        - Short running time
        - Threshold to decide a short running time: *IOthreashold*

Tracking I/O-bound Tasks

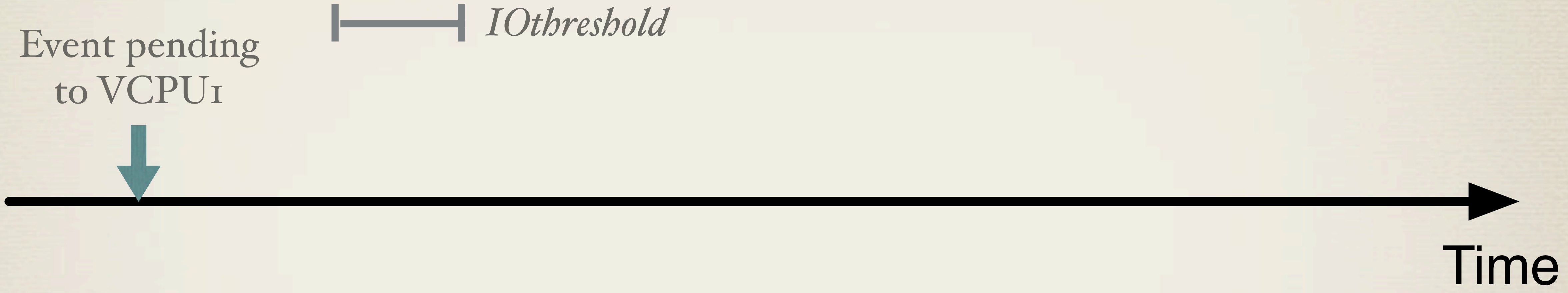
- Three disjoint observation classes based on two gray-box criteria
  - **Positive evidence**
    - supports I/O-boundness
  - **Negative evidence**
    - supports non-I/O-boundness
  - **Ambiguity**
    - No evidence



Preemptively scheduling in response to an event
&
Short running time(< *IOthreshold*)

Tracking I/O-bound Tasks

- Example



Event pending to VCPU1

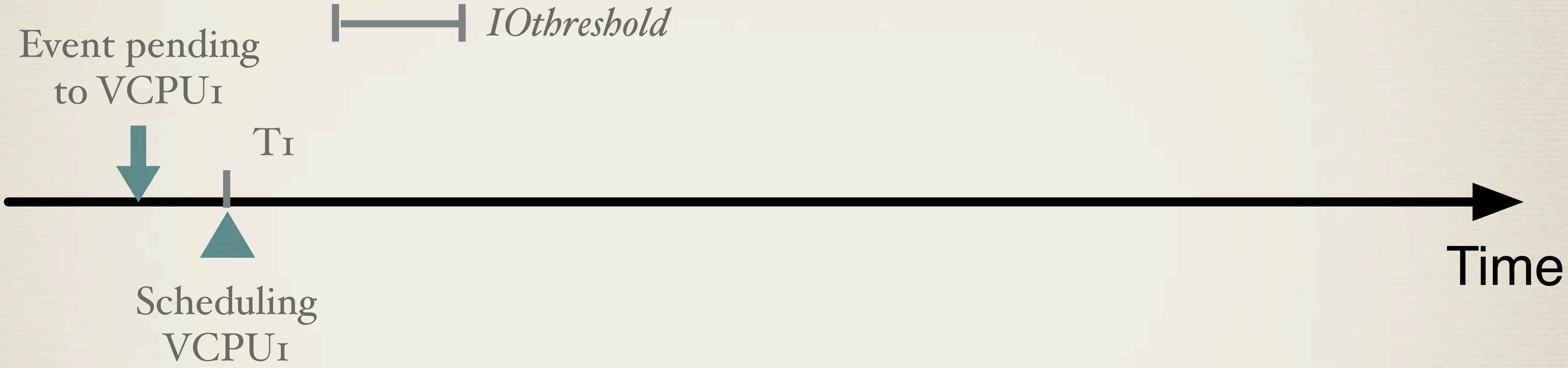
IOthreshold

Time

Positive	Negative	Ambiguity

Tracking I/O-bound Tasks

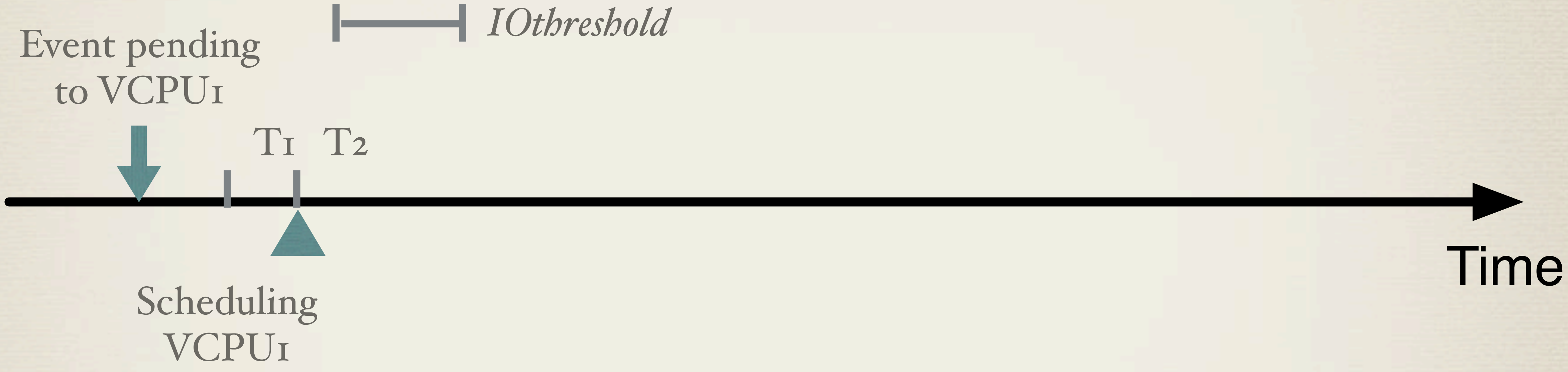
- Example



Positive	Negative	Ambiguity

Tracking I/O-bound Tasks

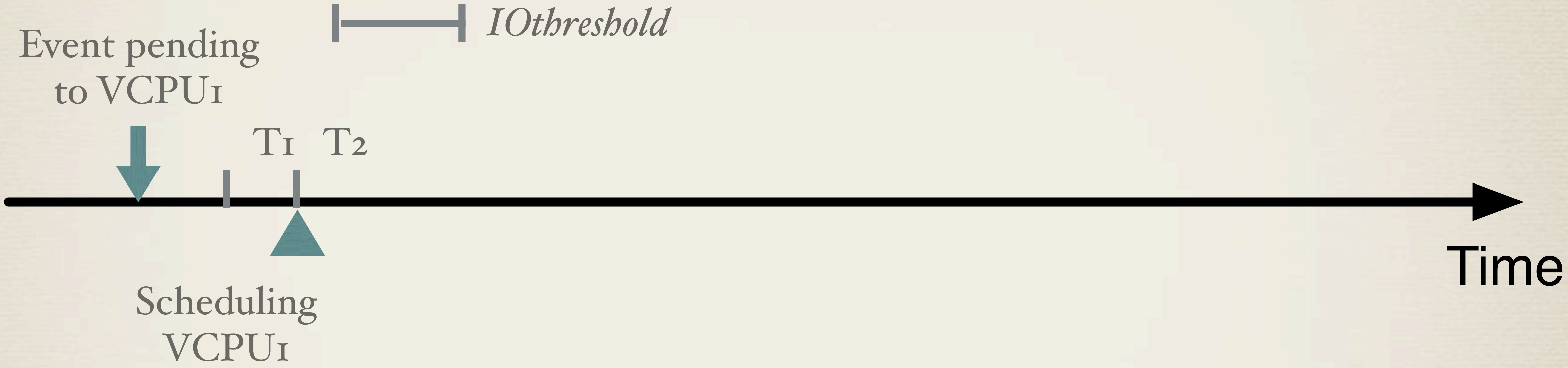
- Example



Positive	Negative	Ambiguity

Tracking I/O-bound Tasks

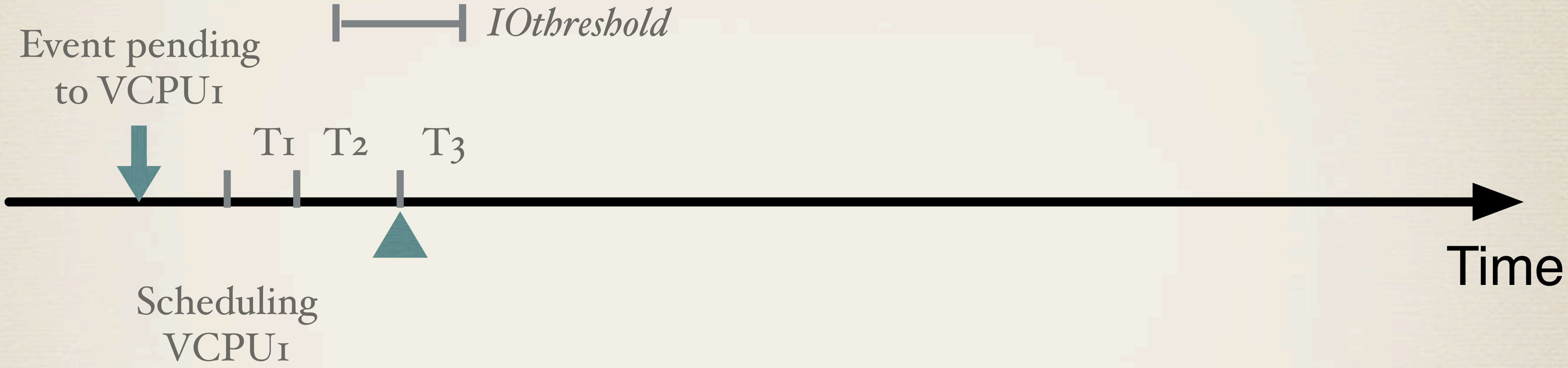
- Example



Positive	Negative	Ambiguity
		T1

Tracking I/O-bound Tasks

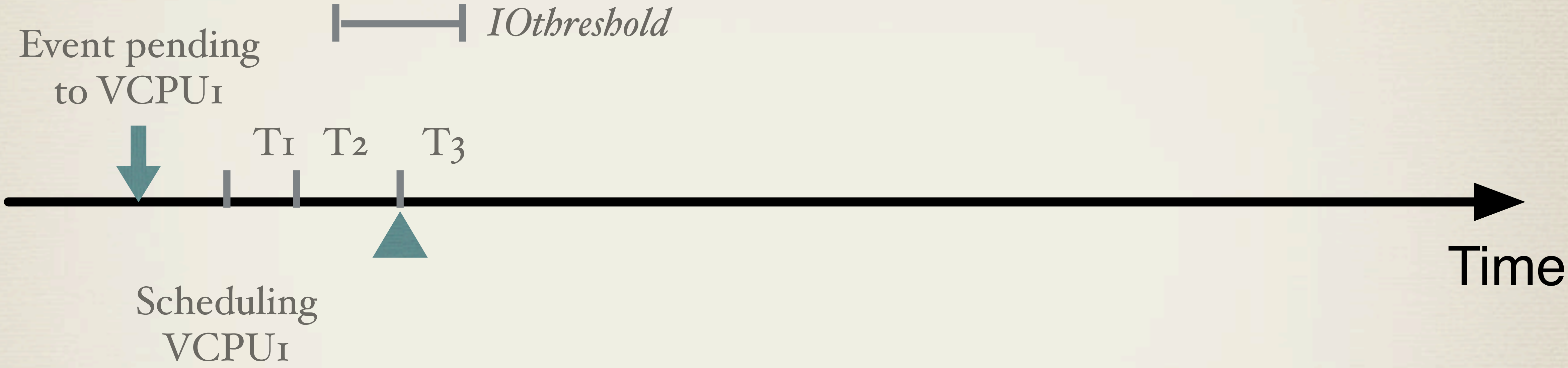
- Example



Positive	Negative	Ambiguity
		T1

Tracking I/O-bound Tasks

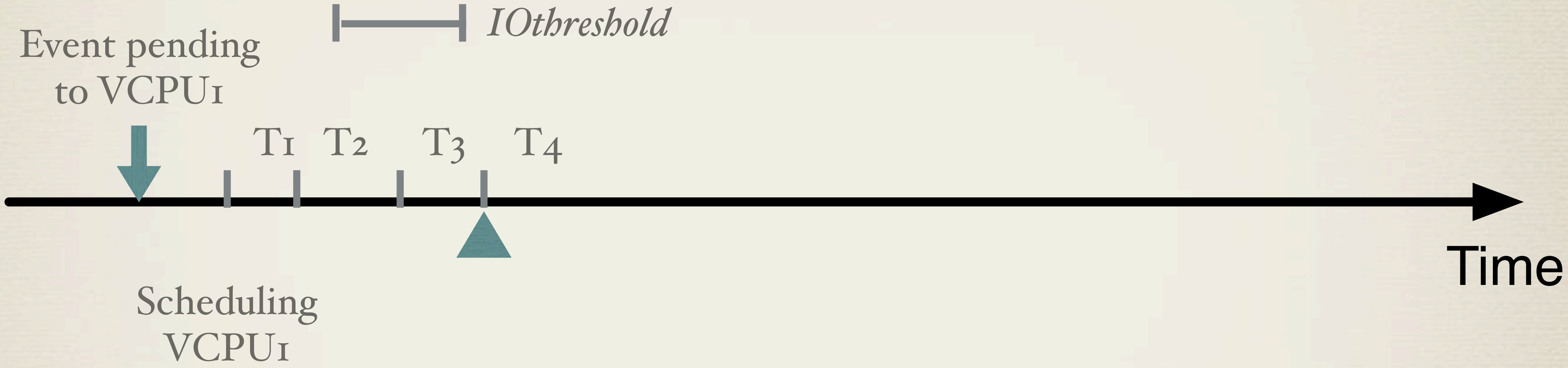
- Example



Positive	Negative	Ambiguity
T2		T1

Tracking I/O-bound Tasks

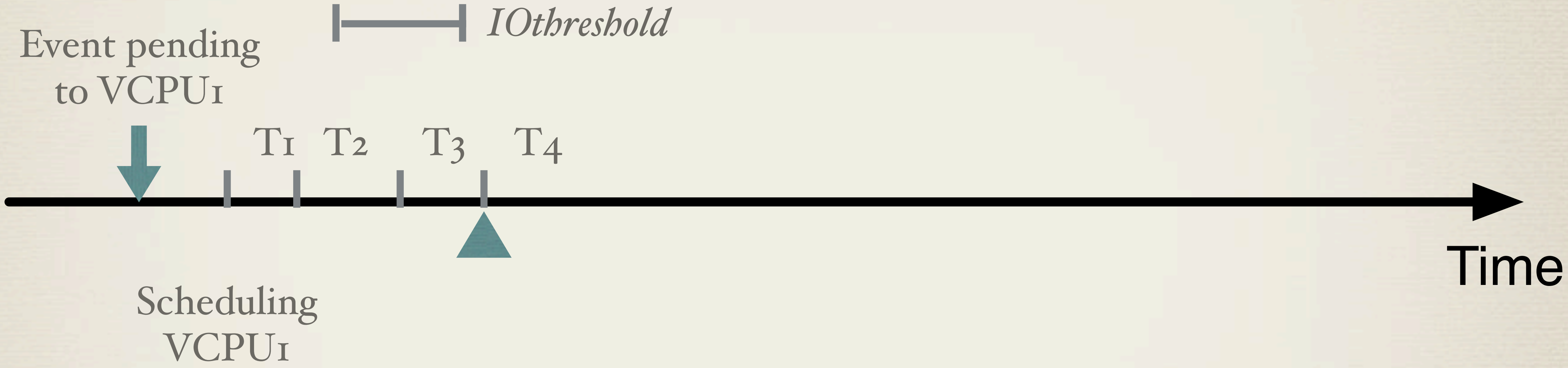
- Example



Positive	Negative	Ambiguity
T2		T1

Tracking I/O-bound Tasks

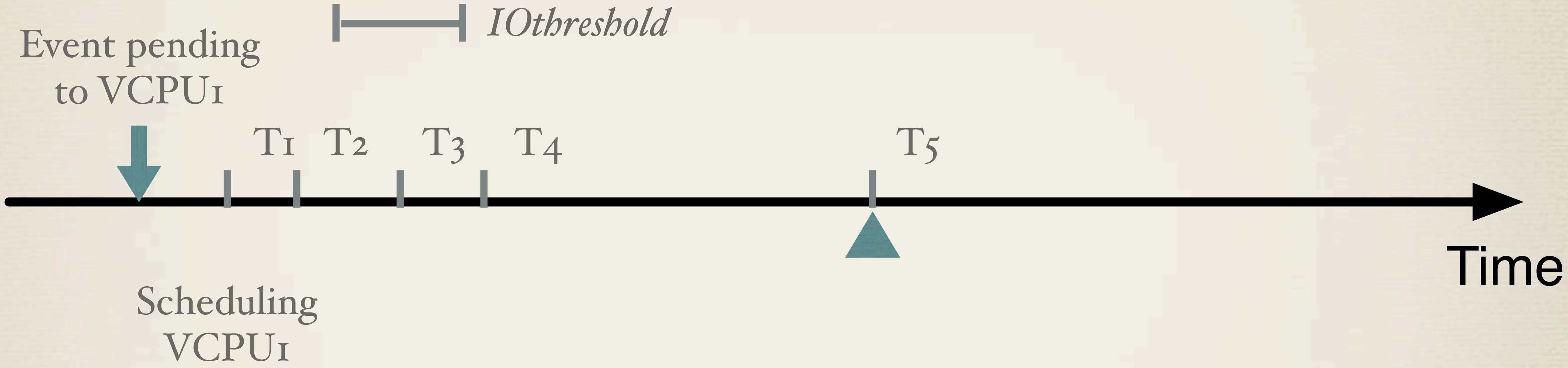
- Example



Positive	Negative	Ambiguity
T2 T3		T1

Tracking I/O-bound Tasks

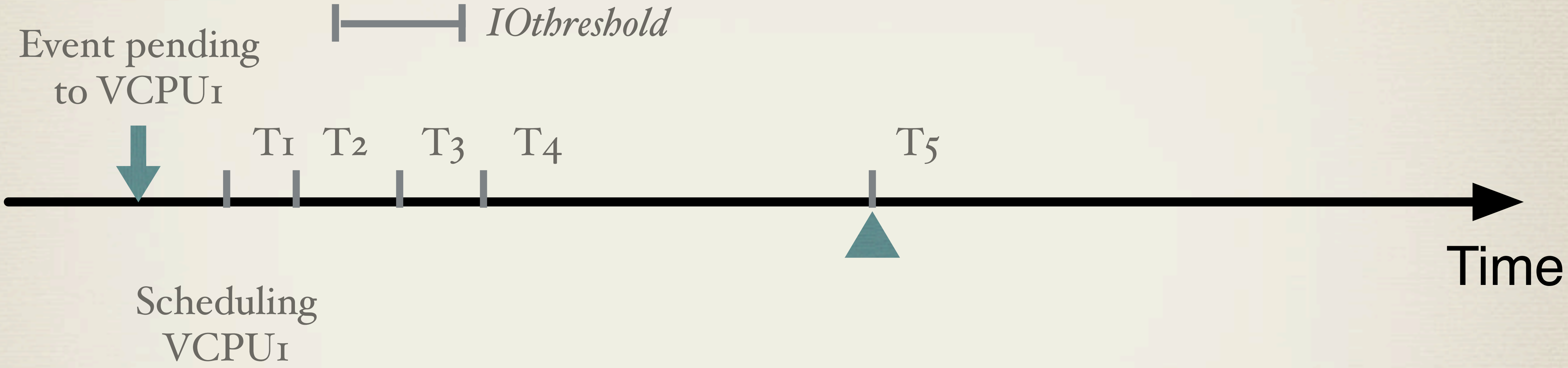
- Example



Positive	Negative	Ambiguity
T2 T3		T1

Tracking I/O-bound Tasks

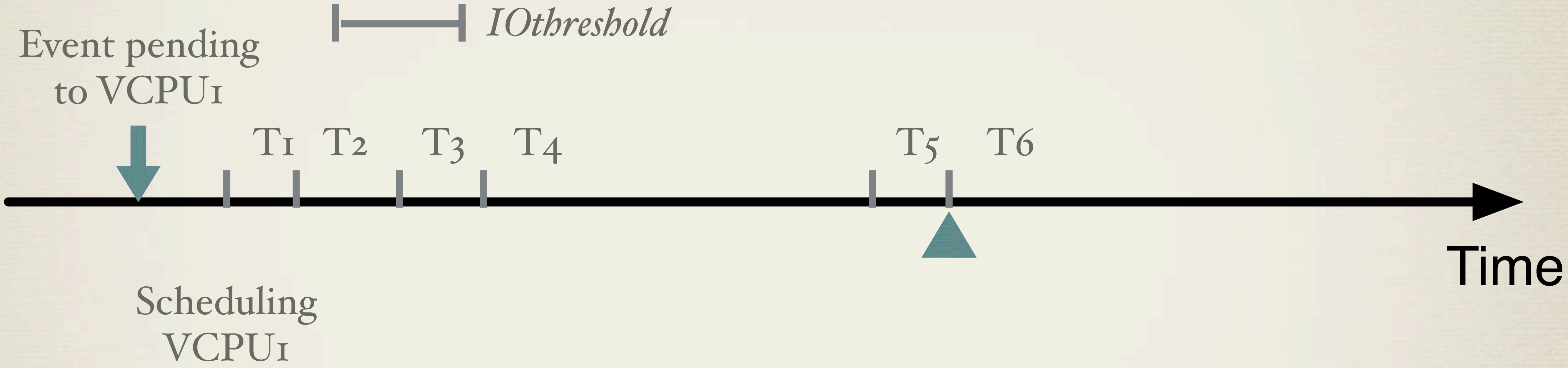
- Example



Positive	Negative	Ambiguity
T2 T3	T4	T1

Tracking I/O-bound Tasks

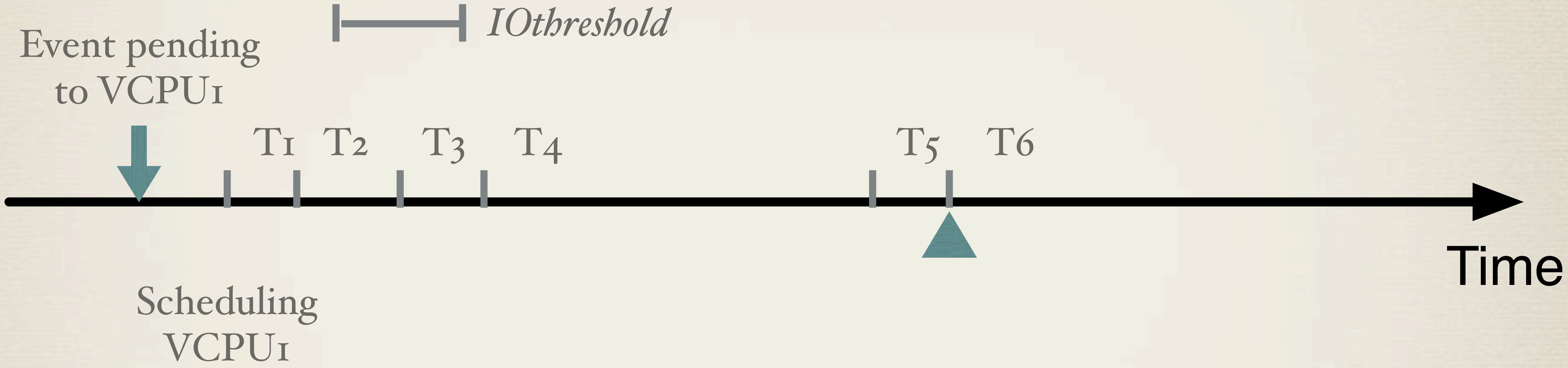
- Example



Positive	Negative	Ambiguity
T2 T3	T4	T1

Tracking I/O-bound Tasks

- Example



IOthreshold

Event pending to VCPU1

T1 T2 T3 T4 T5 T6

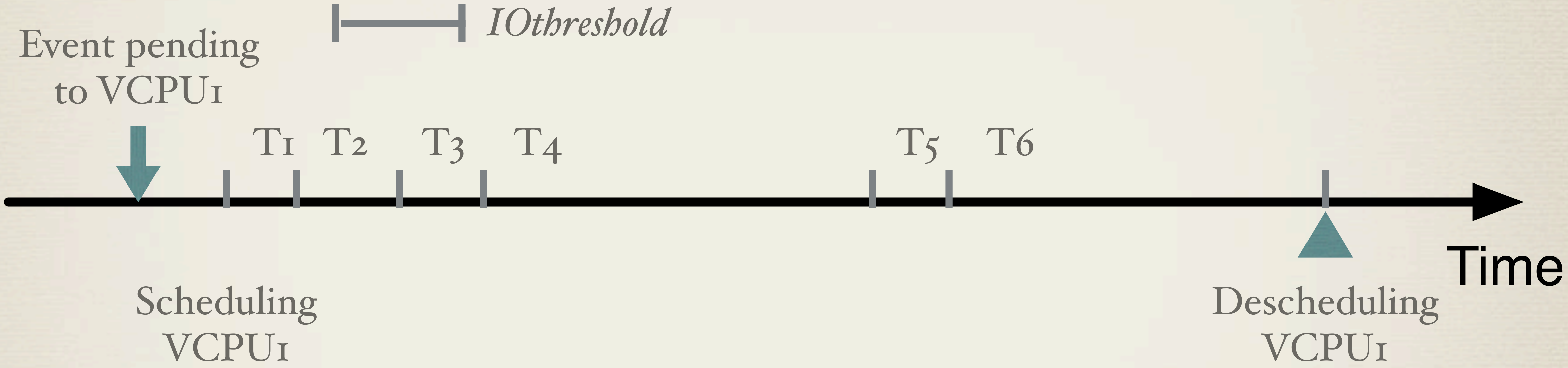
Time

Scheduling VCPU1

Positive	Negative	Ambiguity
T2 T3	T4	T1 T5

Tracking I/O-bound Tasks

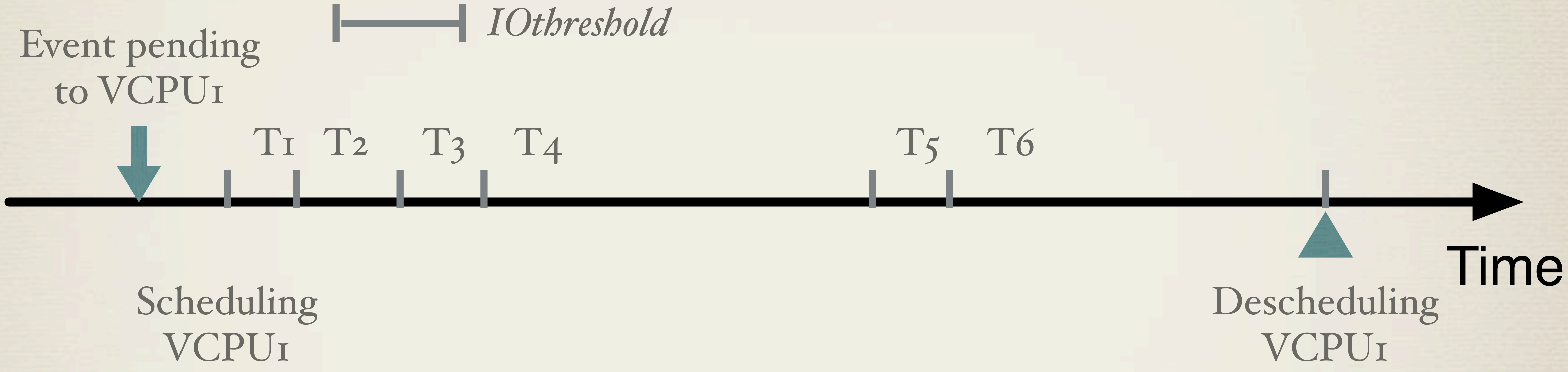
- Example



Positive	Negative	Ambiguity
T2 T3	T4	T1 T5

Tracking I/O-bound Tasks

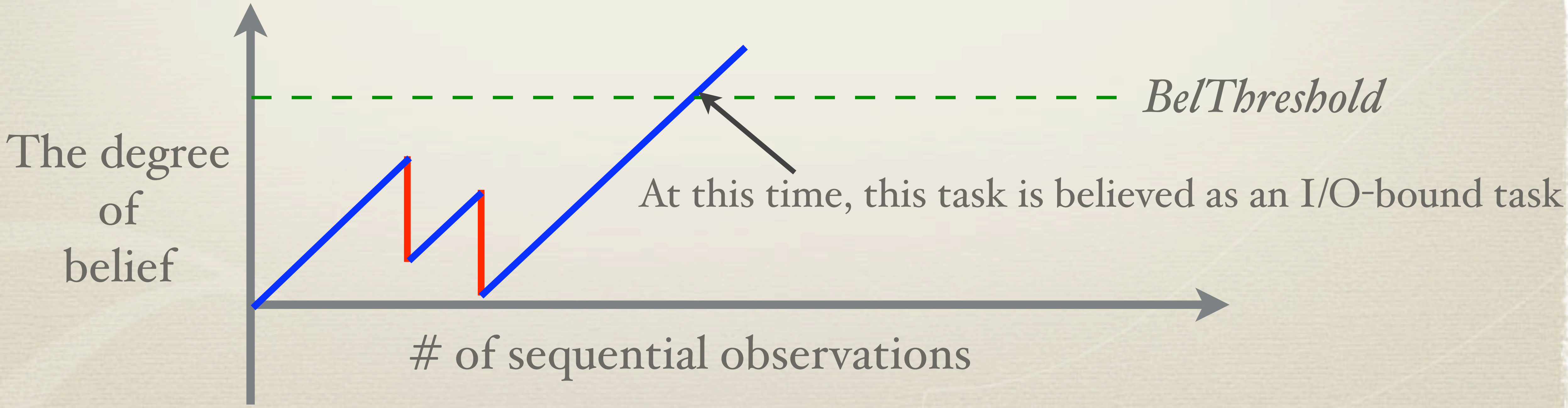
- Example



Positive	Negative	Ambiguity
T2 T3	T4 T6	T1 T5

Tracking I/O-bound Tasks

- Weighted evidence accumulation
  - The degree of belief to reinforce the inference
  - Weight of positive evidence < Weight of negative evidence
    - More penalize for negative evidence



At this time, this task is believed as an I/O-bound task

HOW TO KNOW AN INCOMING EVENT IS FOR AN I/O-BOUND TASK

Correlation Mechanism

- To distinguish an incoming event for I/O-bound task

- **Block I/O**
  - Block read
- **Network I/O**
  - Packet reception

Correlation Mechanism
- Block I/O -

- Request/response style

If T1 requests for reading B1 **and** T1 is I/O-bound
Completion event for B1 is for I/O-bound task

- How to decide “T1 read B1” at the VMM

Correlation Mechanism
- Block I/O -

- Request/response style

If T1 requests for reading B1 **and** T1 is I/O-bound
Completion event for B1 is for I/O-bound task

- How to decide "T1 read B1" at the VMM
  - When the VMM observes a read event, it checks whether the current task is I/O-bound

Correlation Mechanism
- Block I/O -

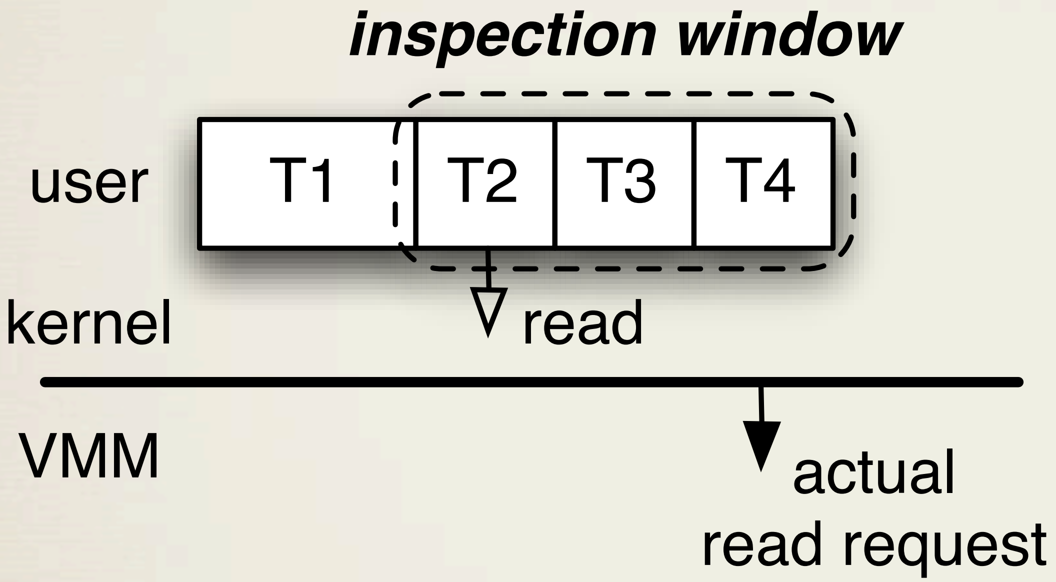
- Request/response style

If T1 requests for reading B1 **and** T1 is I/O-bound
Completion event for B1 is for I/O-bound task

- How to decide "T1 read B1" at the VMM
  - When the VMM observes a read event, it checks whether the current task is I/O-bound
  - But, how about "**delayed read event**" ?
    - Guest OS dependent (e.g. block I/O scheduler)

Correlation Mechanism
- Block I/O -

- Inspection window



If an I/O-bound task **in** inspection window
The actual read request is for I/O-bound task

- False positive VS. False negative

Correlation Mechanism - Network I/O -

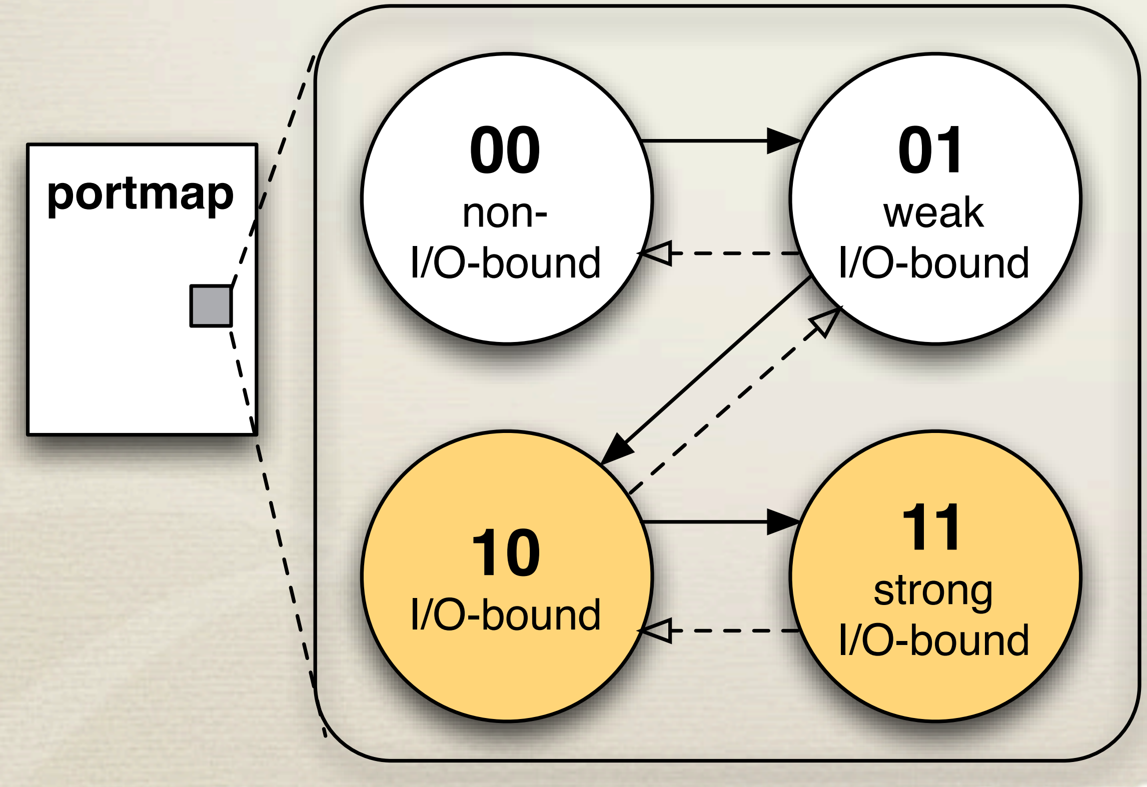
- Event identification
  - Socket-like information is too heavy for the VMM
  - **Destination port number** for TCP/IP communication
    - Most specific to a recipient task
- Asynchronous packet reception
  - No prior information about incoming packets
  - ***History-based prediction mechanism***

Correlation Mechanism - Network I/O -

- History-based prediction mechanism
    - Inference
        - "If an incoming packet is for I/O-bound task, this packet makes the I/O-bound task to be preemptively scheduled"
    - Monitoring <u>the first woken task</u> in response to an incoming packet

Correlation Mechanism
- Network I/O -

- History-based prediction mechanism (cont')
    - *Portmap*
        - An entry for each destination port number
        - Each entry is an N-bit saturating counter



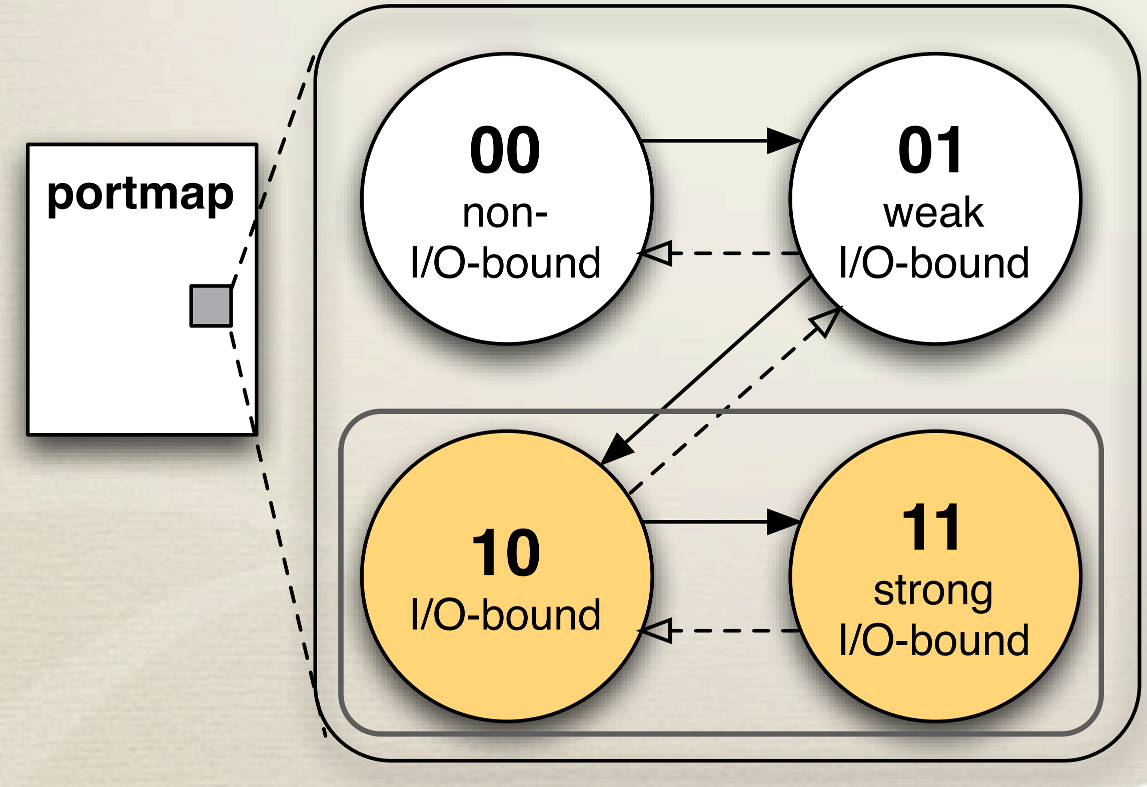
Example (2-bit counter)

→ If the first woken task is I/O-bound

⇢ Otherwise

Correlation Mechanism
- Network I/O -

- History-based prediction mechanism (cont')
  - *Portmap*
    - An entry for each destination port number
    - Each entry is an N-bit saturating counter



Example (2-bit counter)

⟶ If the first woken task is I/O-bound

- - -▷ Otherwise

If portmap counter's MSB is set, this packet is for I/O-bound

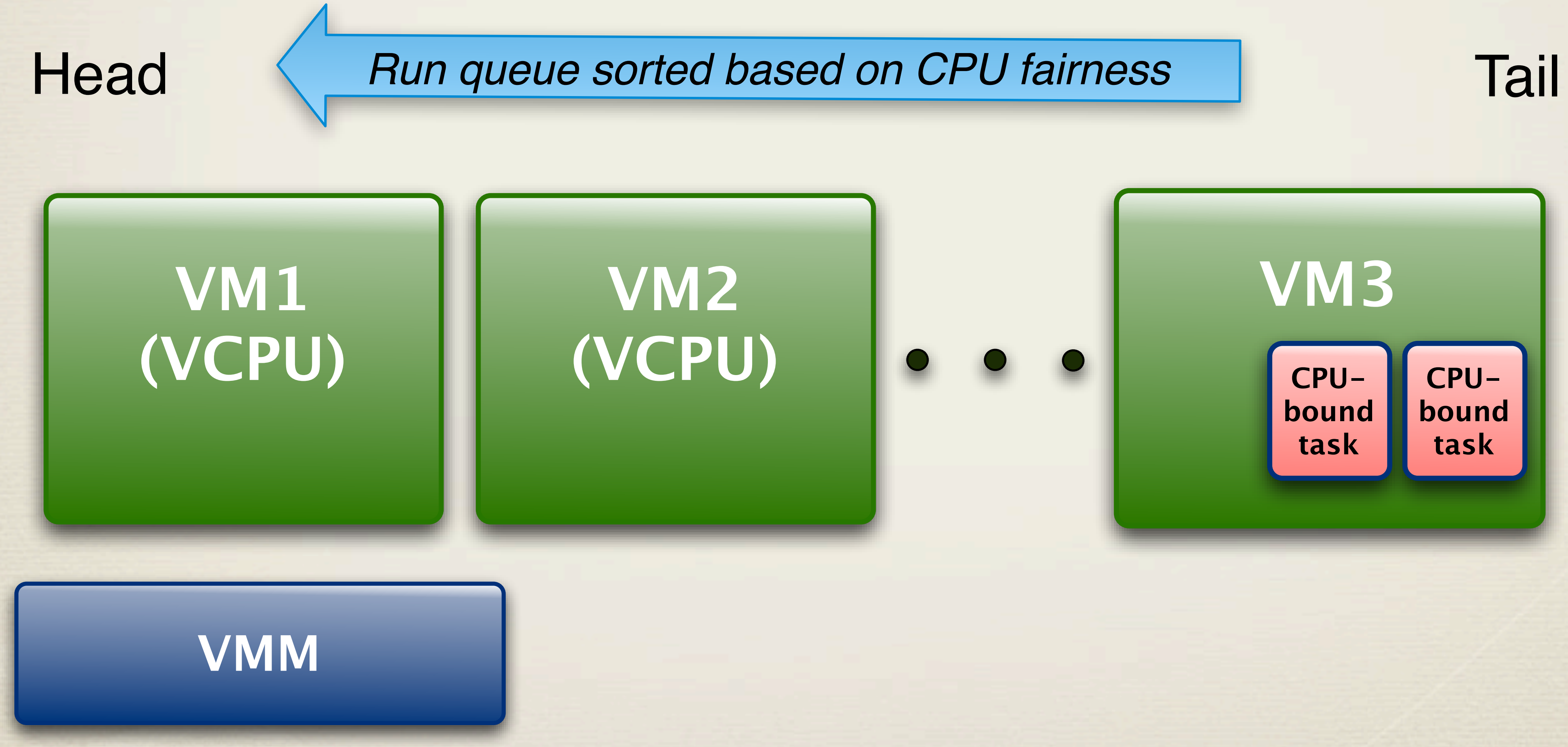
PARTIAL BOOSTING

Partial Boosting

- Priority boosting with task-level granularity
  - Priority boosting lasts during the run of an I/O-bound task
- Why?
  - To prevent CPU-bound tasks in a boosted VCPU from compromising CPU fairness

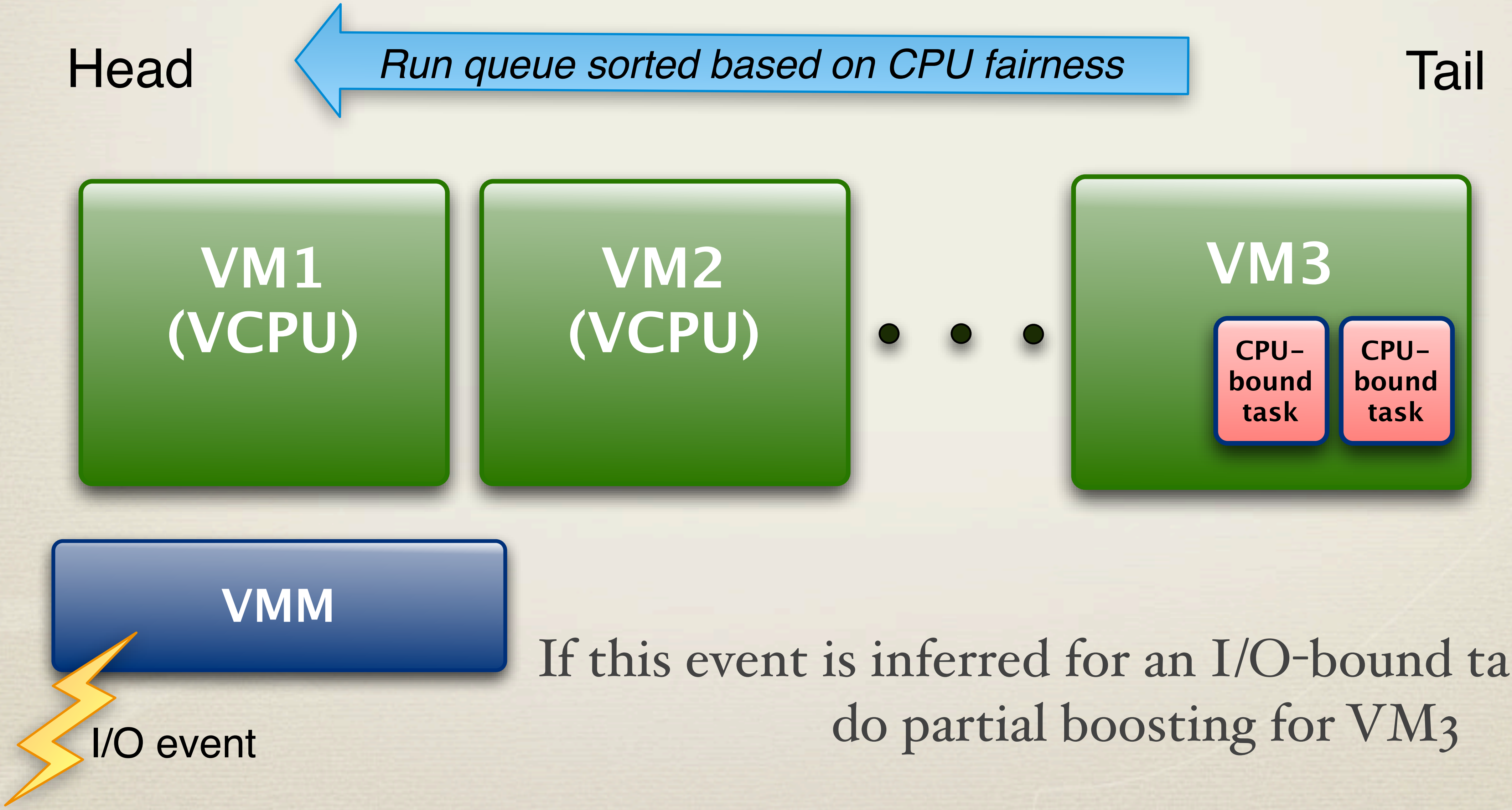
Partial Boosting

- Procedure



Partial Boosting

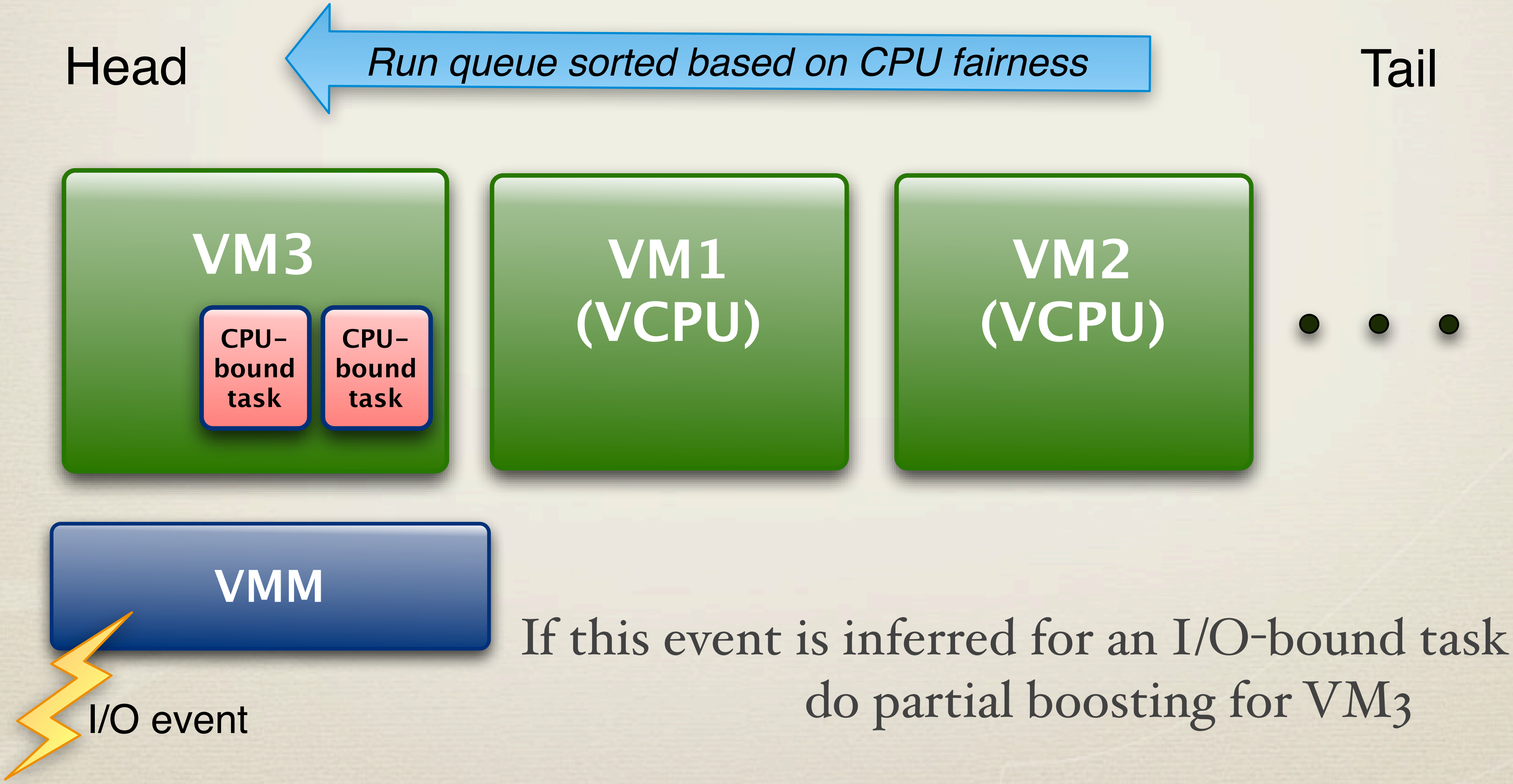
- Procedure



If this event is inferred for an I/O-bound task in VM3, do partial boosting for VM3

Partial Boosting

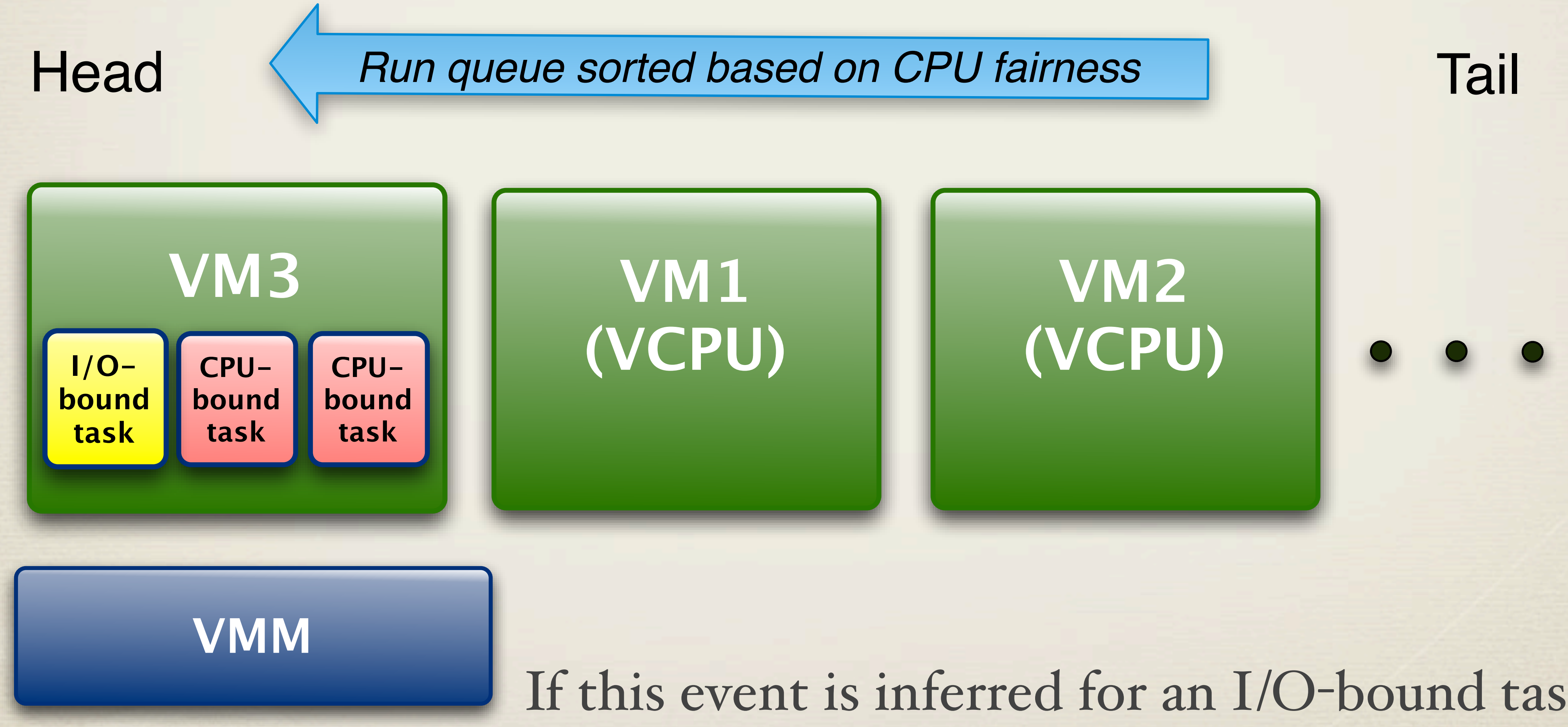
- Procedure



If this event is inferred for an I/O-bound task in VM3, do partial boosting for VM3

Partial Boosting

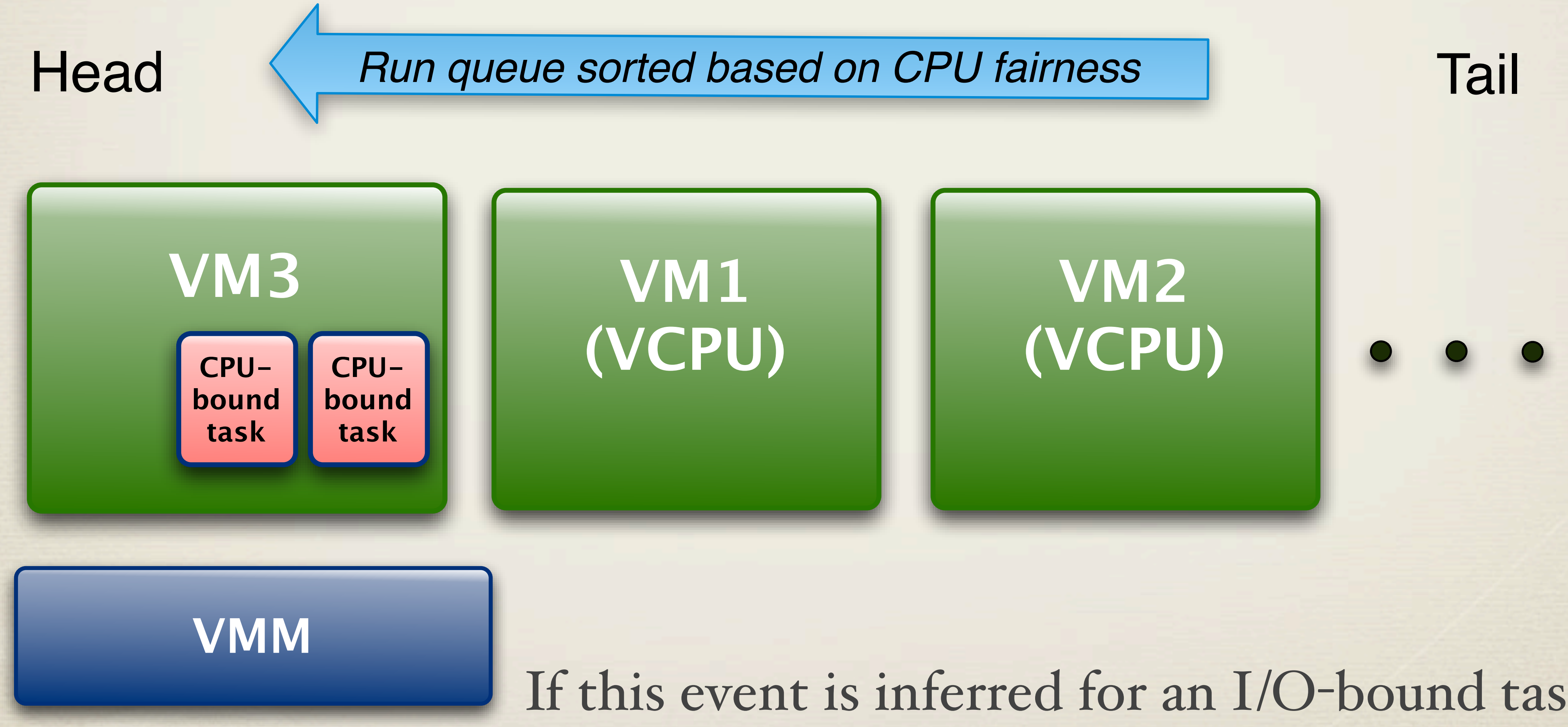
- Procedure



If this event is inferred for an I/O-bound task in VM3, do partial boosting for VM3

Partial Boosting

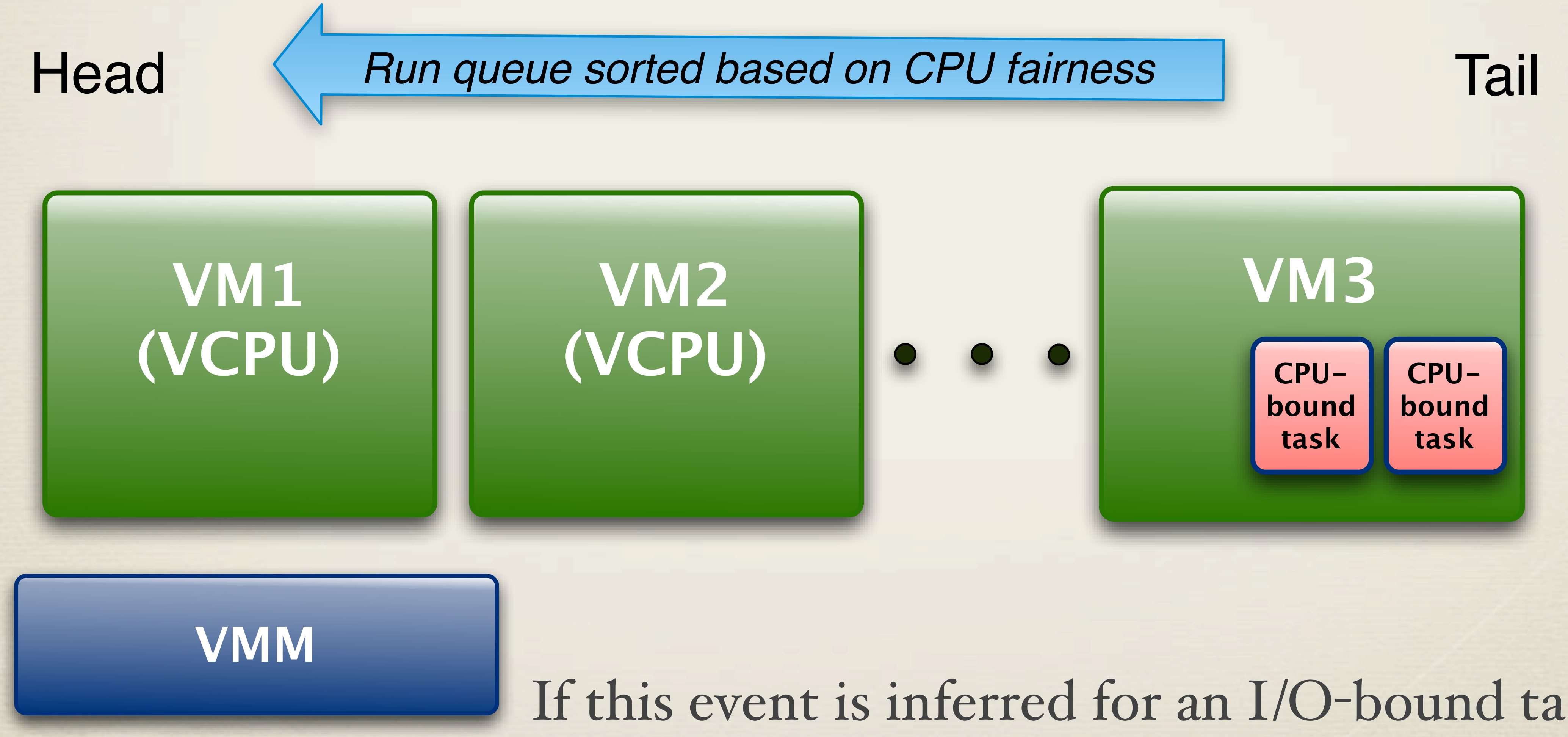
- Procedure



If this event is inferred for an I/O-bound task in VM3, do partial boosting for VM3

Partial Boosting

- Procedure



If this event is inferred for an I/O-bound task in VM3, do partial boosting for VM3

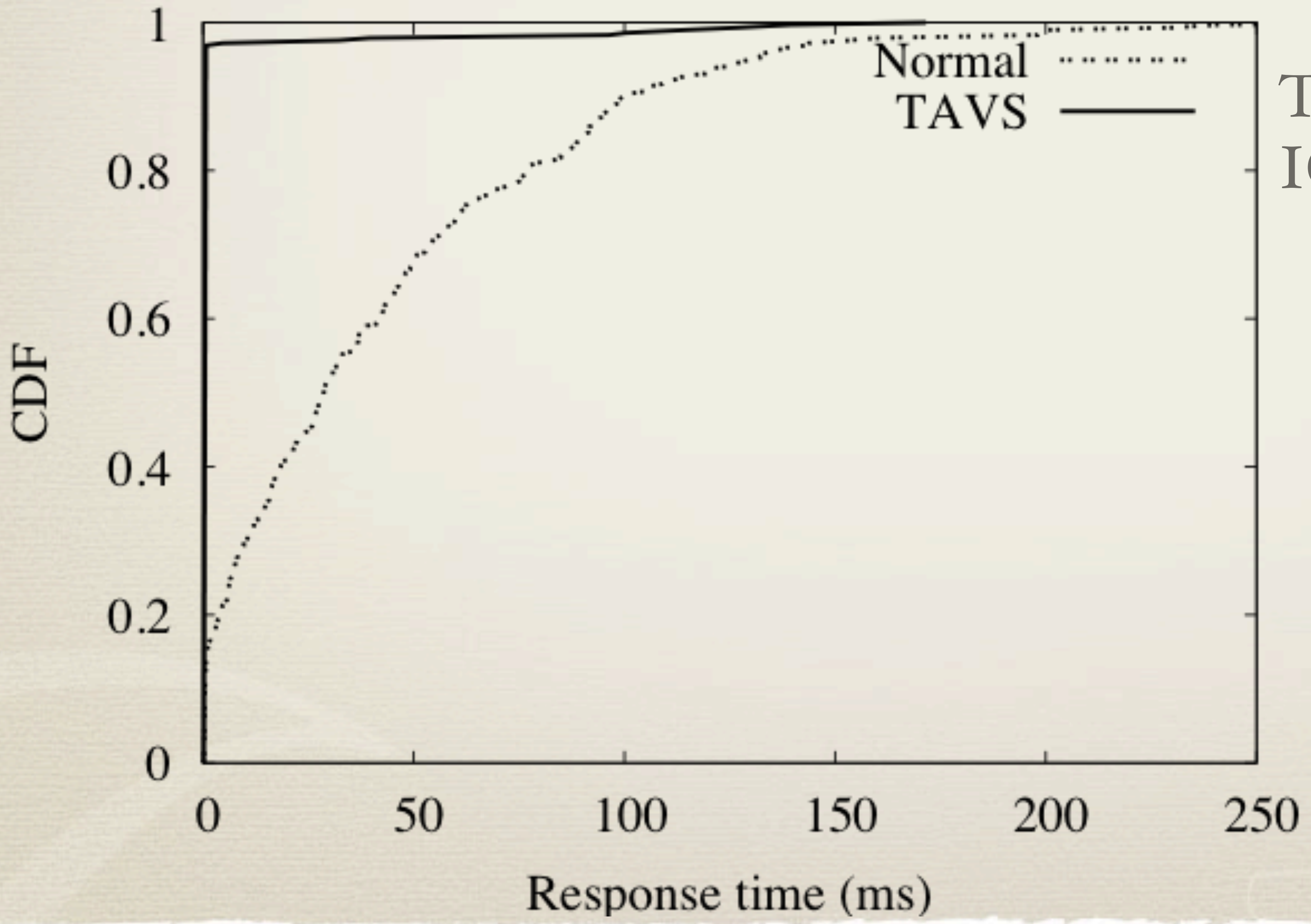
IMPLEMENTATION & EVALUATION

Implementation

- Based on Credit scheduler in Xen 3.2.1
- Task information maintained by hash
  - Limited number of tasks maintained
  - Remove of a task with infrequent I/O
- Correlation
  - Block I/O : using grant table in Xen
  - Network I/O : supported by network backend driver
- No consideration of multiple VCPUs

Evaluation

- Interactive workload
  - Packet request-response



Worst case scenario
1 dom: Server & CPU-bound task
5doms: CPU-bound task

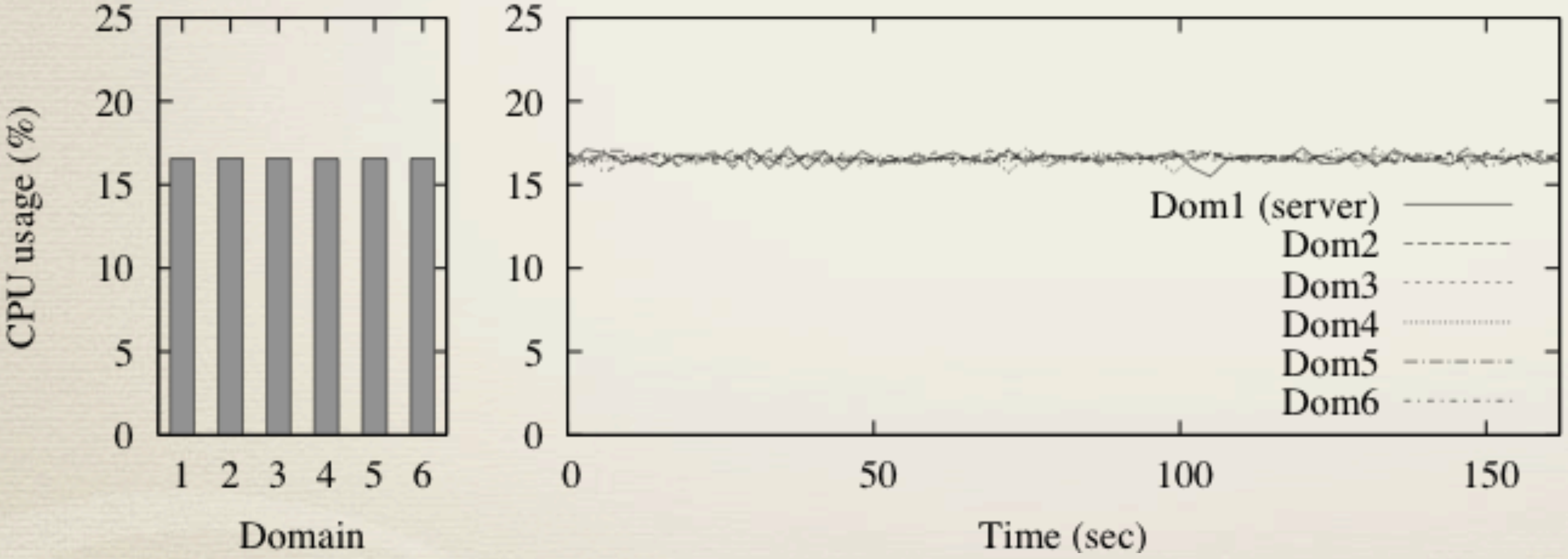
Think time: 100 ~ 1000 ms
IOthreshold = 0.5 ms

Evaluation

- Interactive workload
  - Packet request-response

Worst case scenario
1 dom: Server & CPU-bound task
5doms: CPU-bound task

Think time: 100 ~ 1000 ms
IOthreshold = 0.5 ms



Evaluation

- Correlation evaluation
  - Partial boosting hit ratio (PBHR)

$$\texttt{PBHR (\%)} = \frac{\sum h}{The\ number\ of\ partial\ boostings} \times 100$$

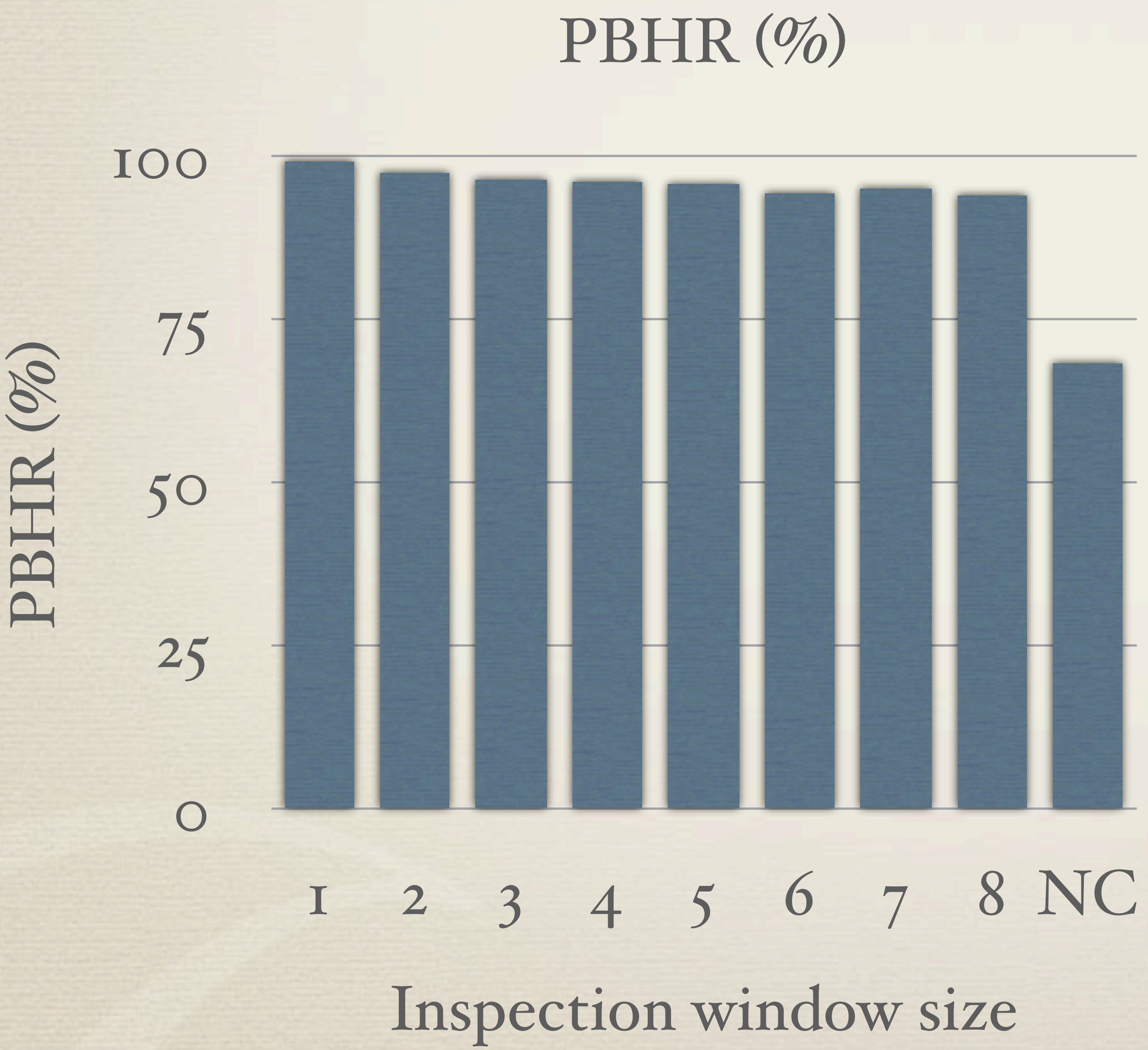
where

$$h = \begin{cases} 1 & \text{, if an I/O-bound task awakes during partial boosting.} \\ 0 & \text{, otherwise.} \end{cases}$$

  - defined as true positive ratio
    - False positive ratio = (100 - PBHR) %

Evaluation

- Correlation evaluation: Block I/O



Workloads

1 dom: 8 tasks

    1 task: I/O-bound task

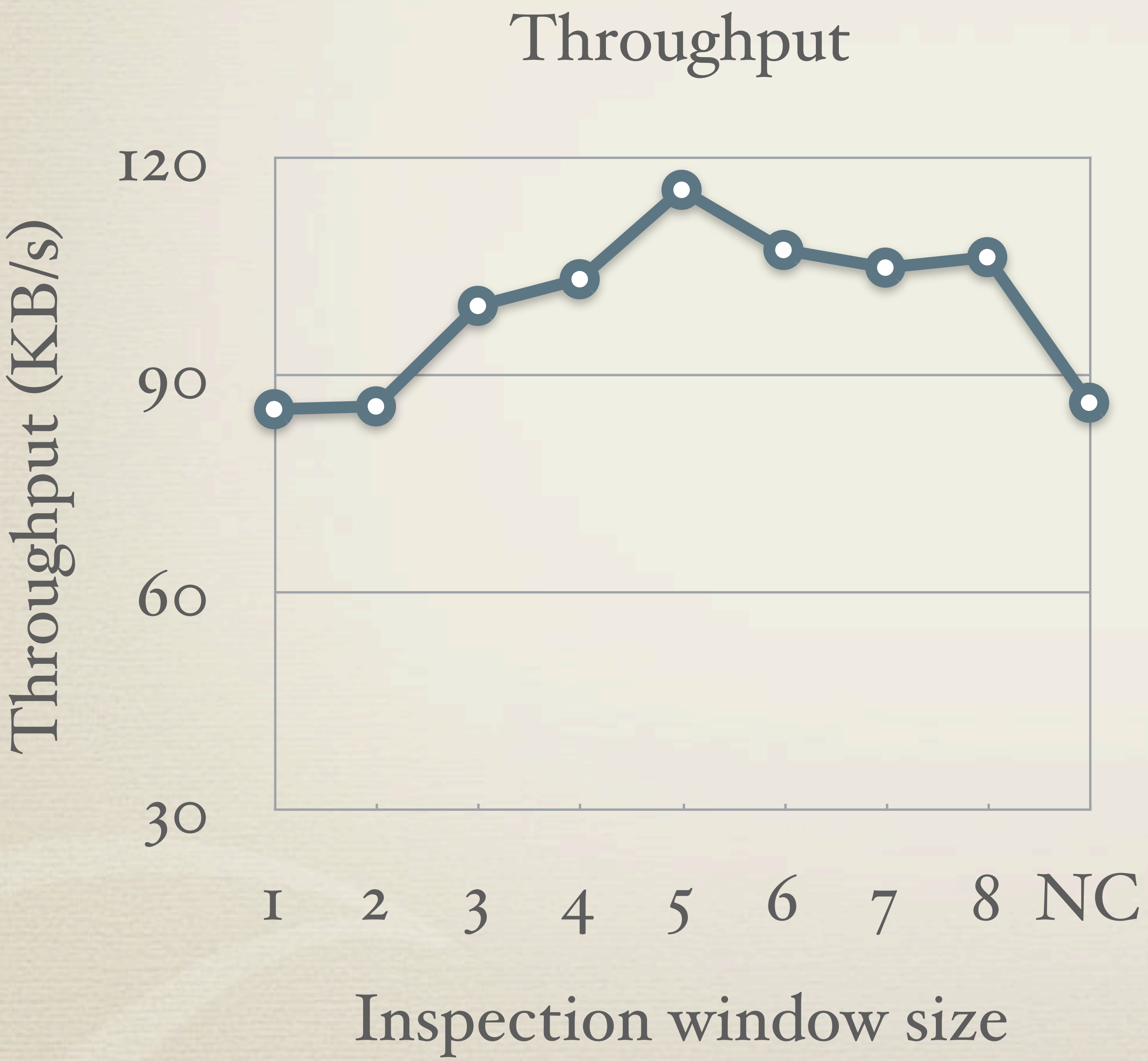
    7 tasks: I/O+CPU task

    (CPU usage 1-300ms between IOs)

5doms: CPU-bound task

Evaluation

- Correlation evaluation: Block I/O



Workloads

1 dom: 8 tasks

1 task: I/O-bound task

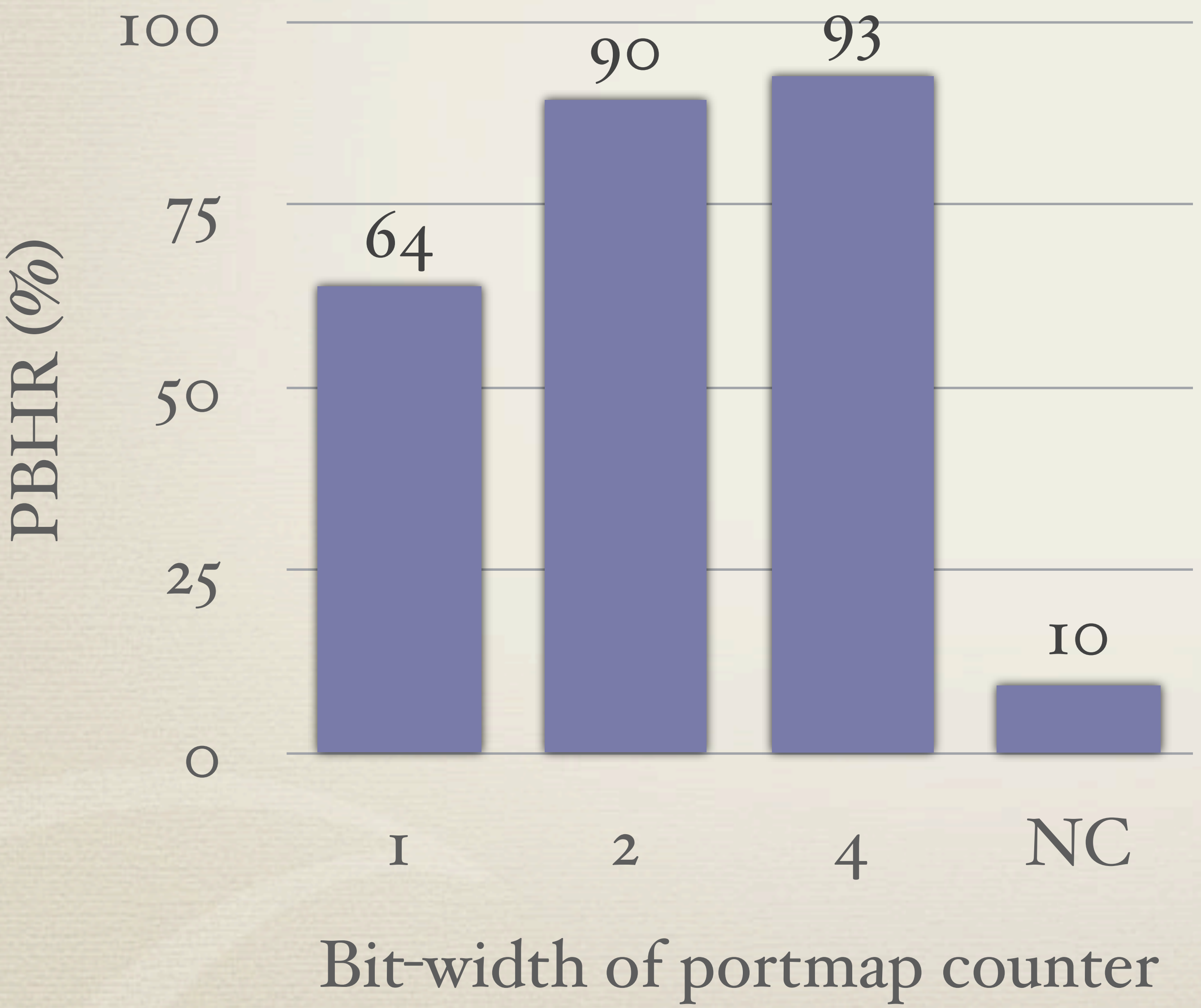
7 tasks: I/O+CPU task

(CPU usage 1-300ms between IOs)

5doms: CPU-bound task

Evaluation

- Correlation evaluation: Network I/O



Workloads

1 dom: 8 tasks

1 task: I/O-bound task

7 tasks: I/O+CPU task

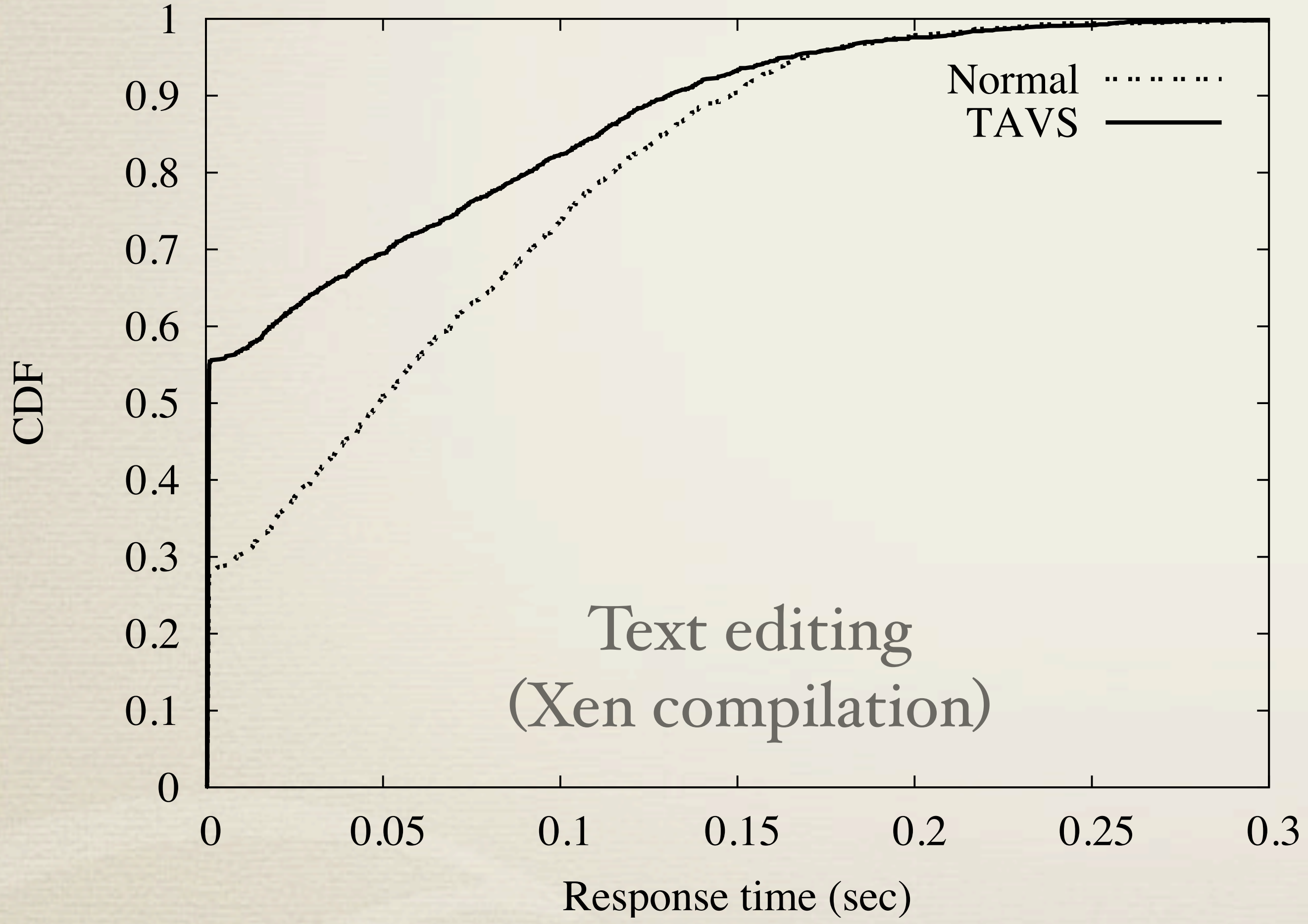
(CPU usage 1-300ms between IOs)

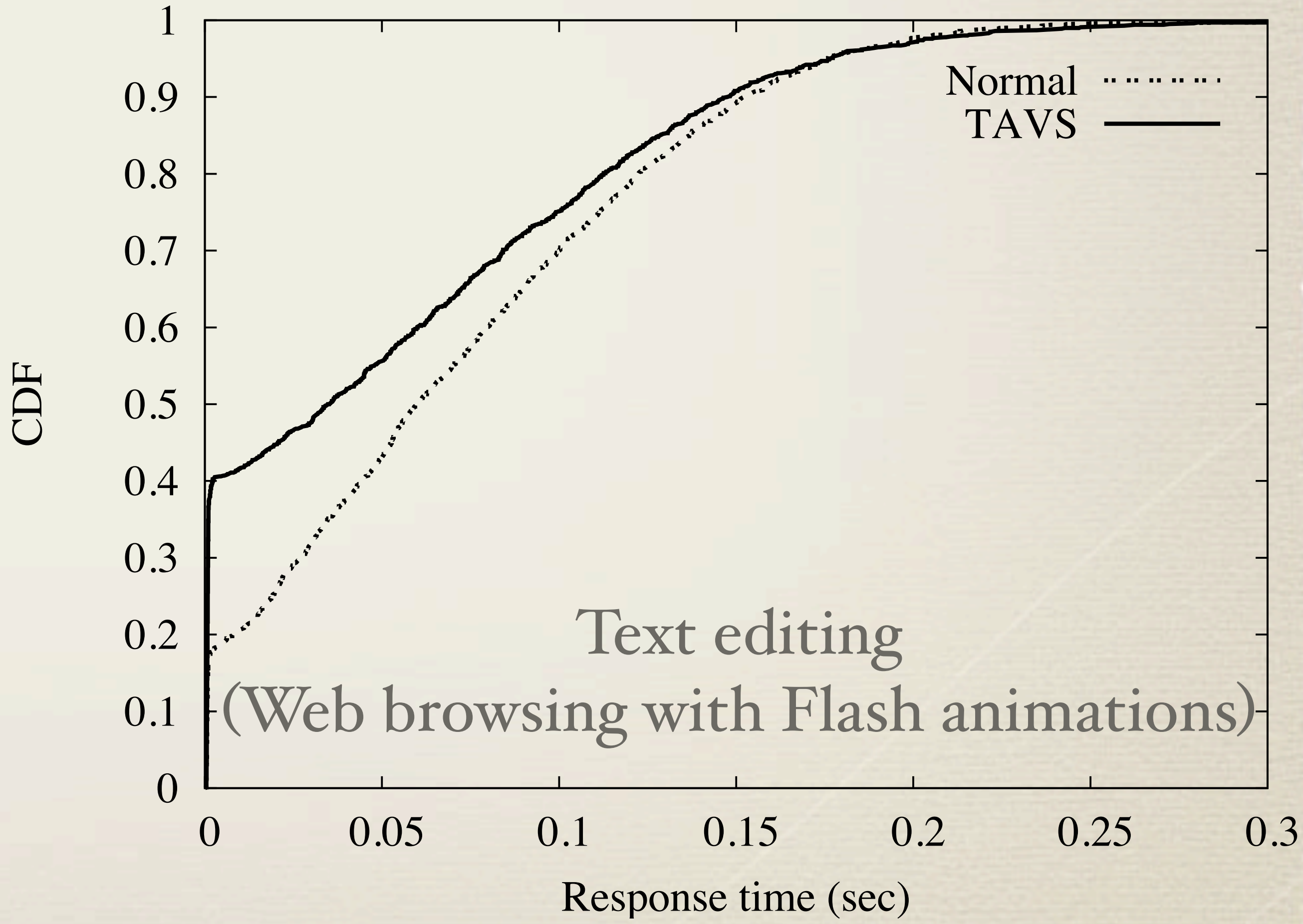
5doms: CPU-bound task

2-bit is reasonable
in terms of space overheads

Evaluation

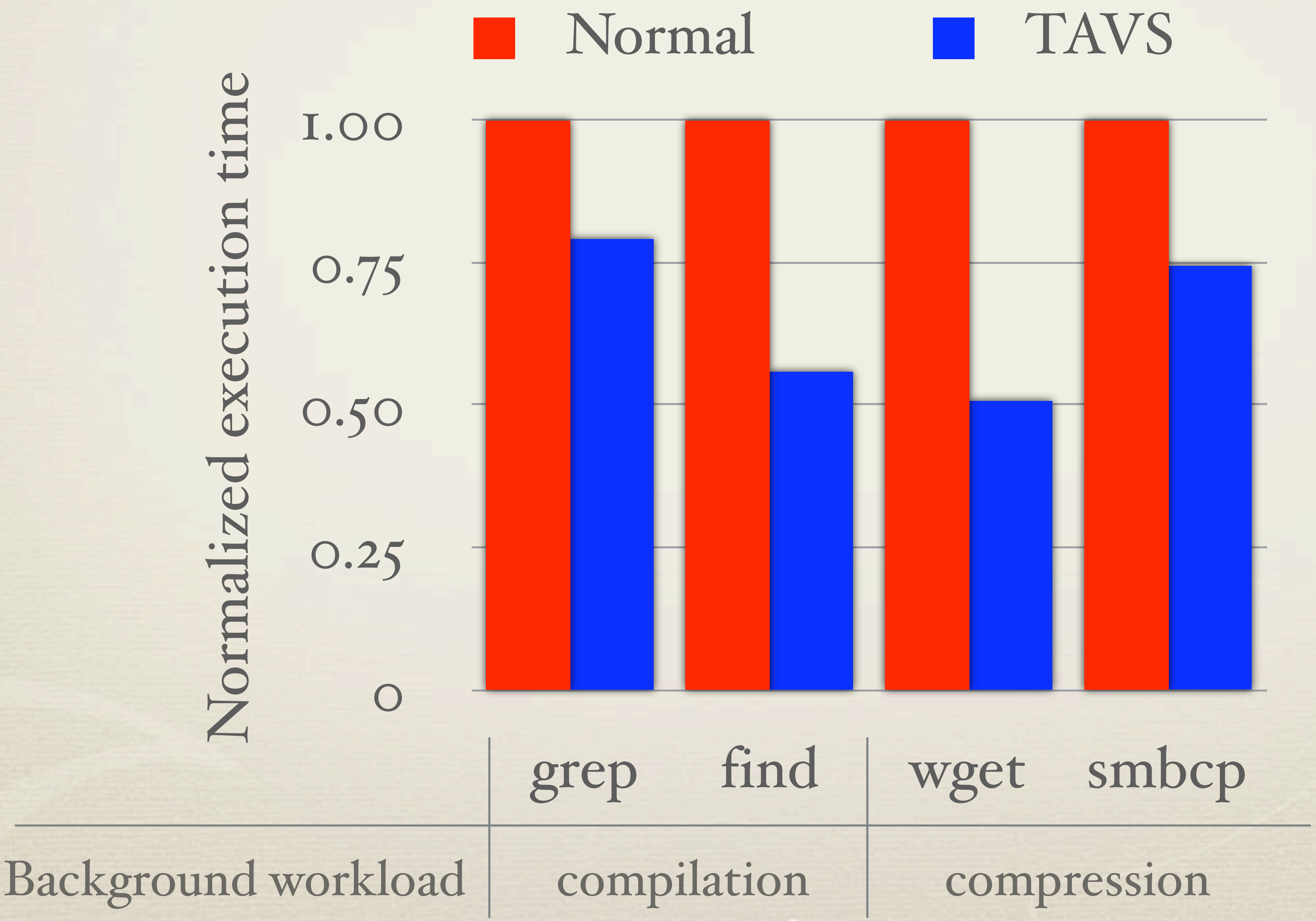
- Response time for text editing during CPU-intensive workload





Evaluation

- Execution time of I/O-bound tasks with CPU-intensive workloads



PBHR = 99%

smbcp: copy files from remote samba server

Evaluation

- Overheads
  - Task tracking overheads: 0.06%
  - No overhead for inspecting incoming packets
  - Increased network throughput : decreased CPU throughput = 48 : 1
  - Space overhead of N-bit portmap
    - N * 8KB for each VM
    - e.g. 2-bit portmaps for TCP and UDP: 32KB for each VM

Conclusions

- Task-aware VM scheduling
  - Bridging the semantic gap in CPU management
  - Transparency by VMM-level inference
    - Gray-box technique
  - Low overheads

Future Work

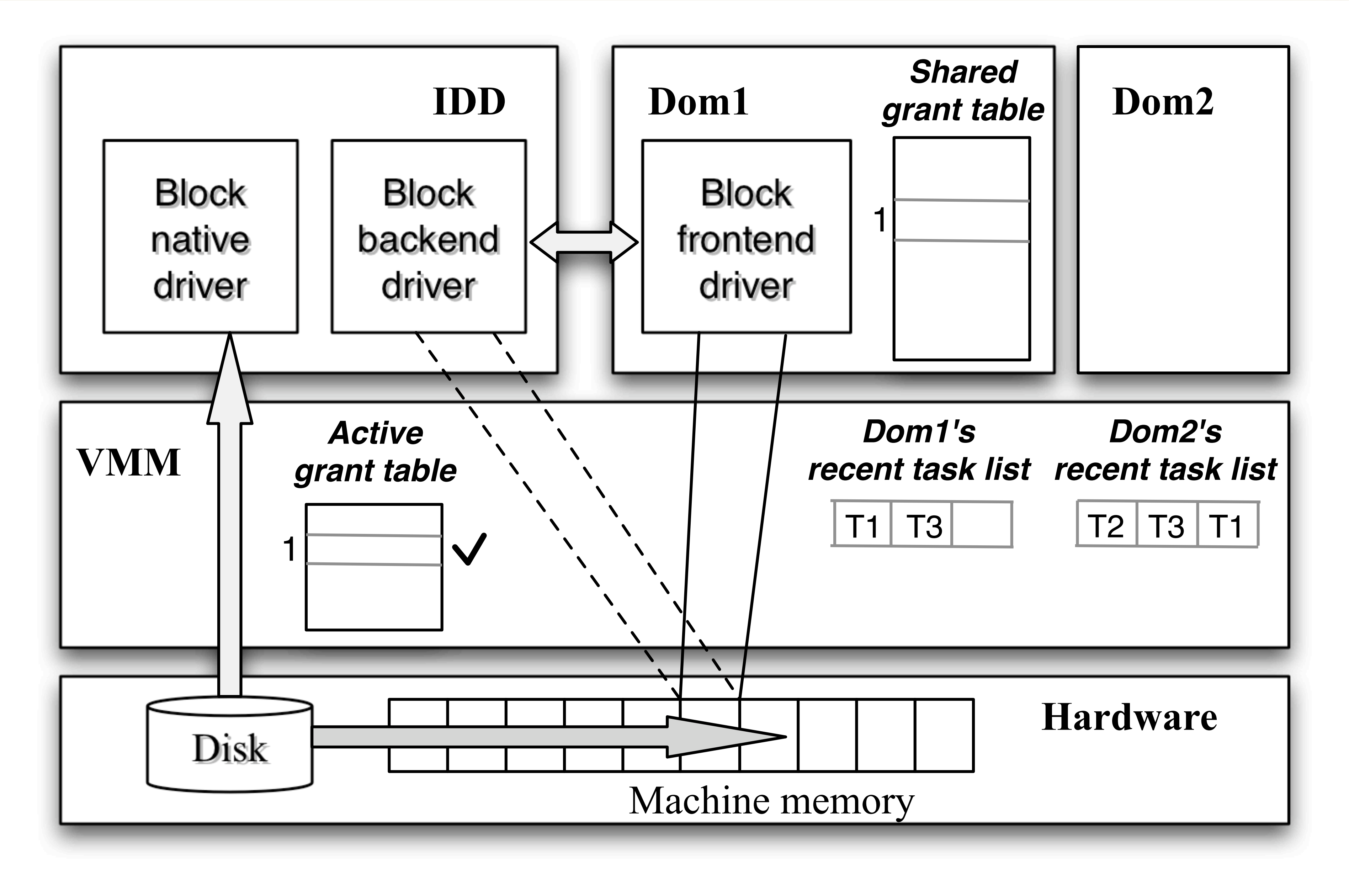
- Extension on multicore system
- Simulation-based analysis for more intelligent scheduling
- Evaluation for more various workloads

THANK YOU!

BACKUP SLIDES

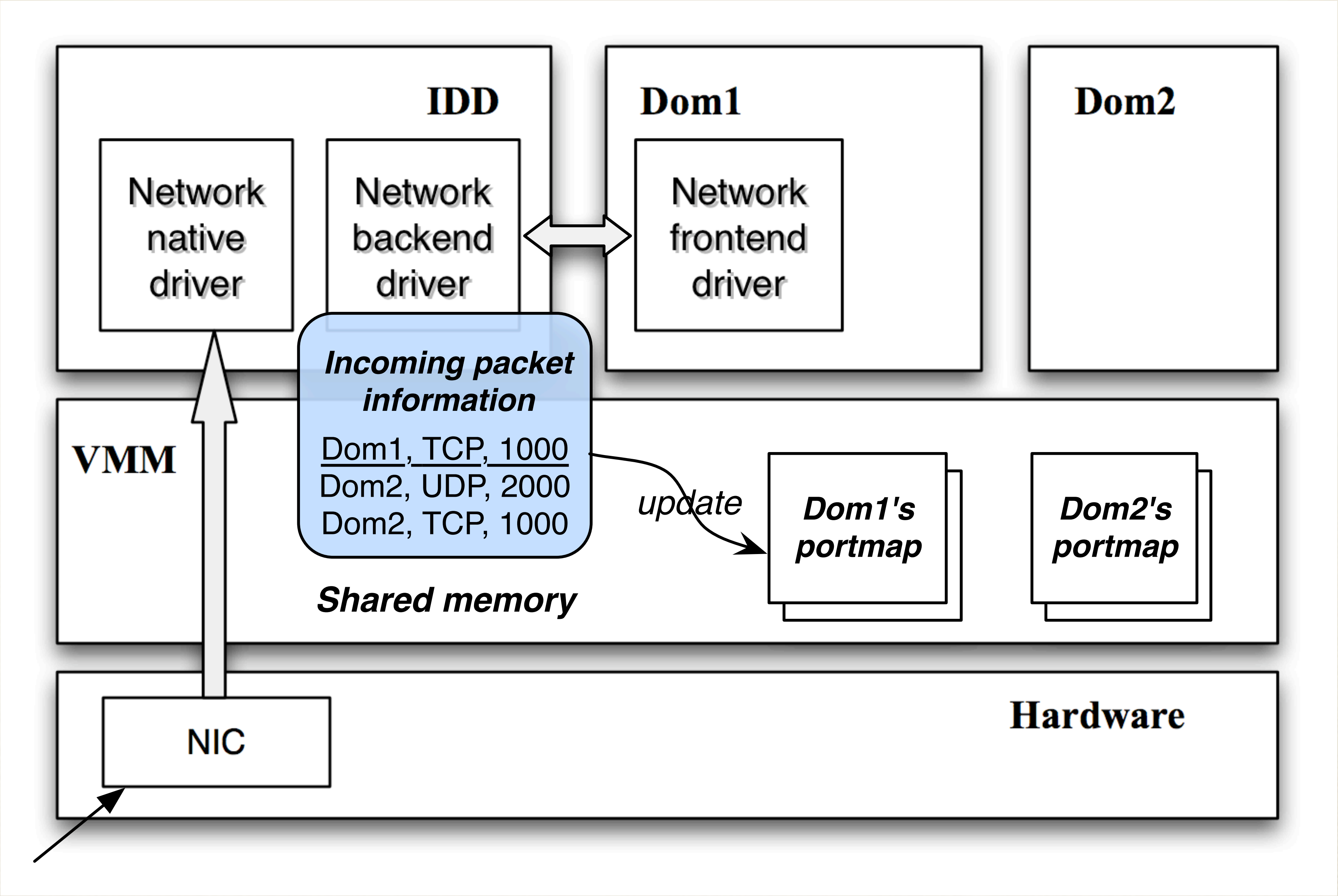
Implementation

- Block correlation

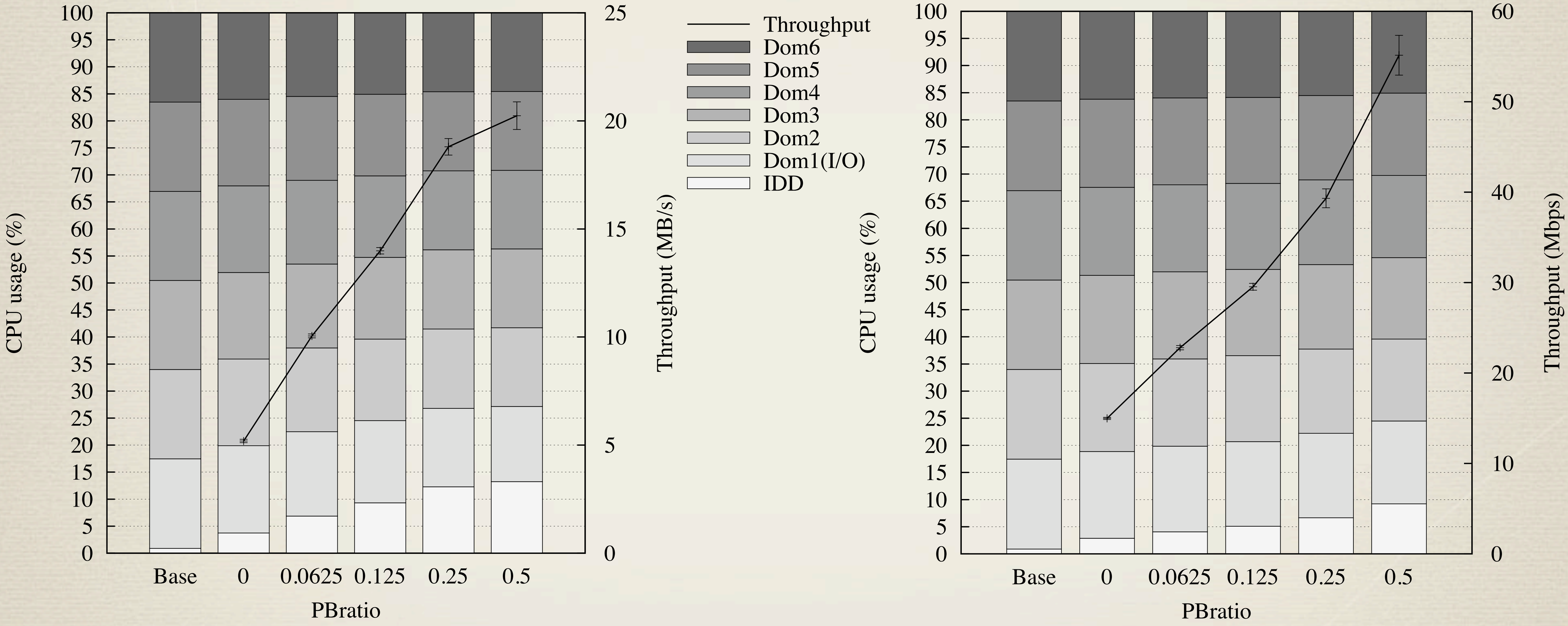


Implementation

- Network correlation



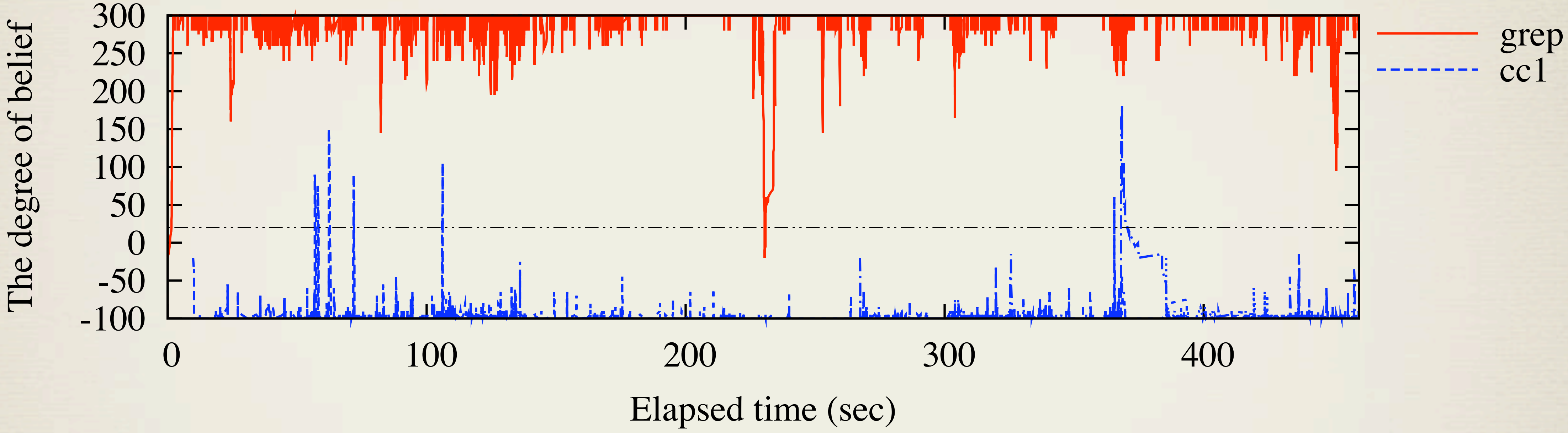
Throughput

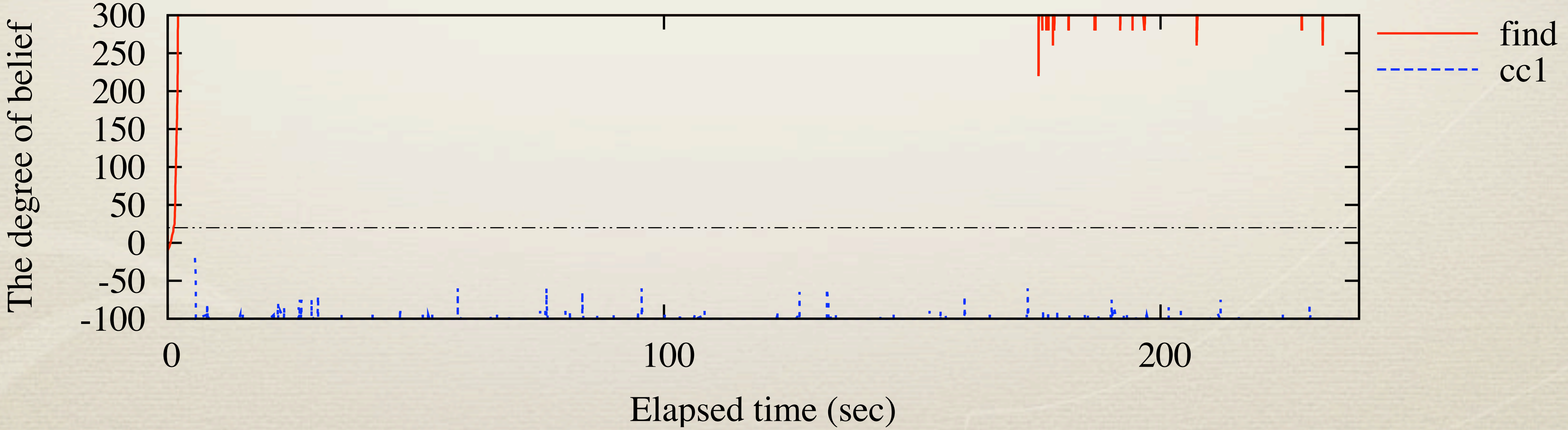


$$\texttt{PBratio} = \frac{Allowed\ CPU\ usage\ for\ partial\ boosting}{Total\ CPU\ usage}$$

Degree of Belief

- Degree of belief (grep, find + compilation)


The degree of belief
Elapsed time (sec)
grep
cc1


The degree of belief
Elapsed time (sec)
find
cc1