AutoV: An Automotive Testbed for Real-Time Virtualization

Meng Xu Insup Lee

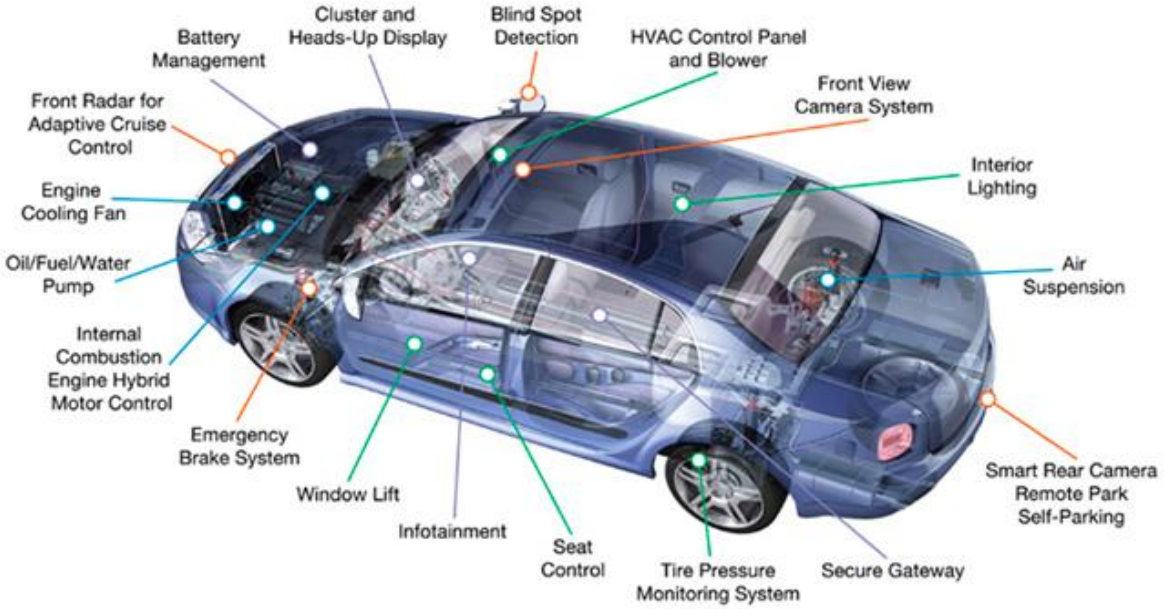
PRECISE Center
University of Pennsylvania

Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

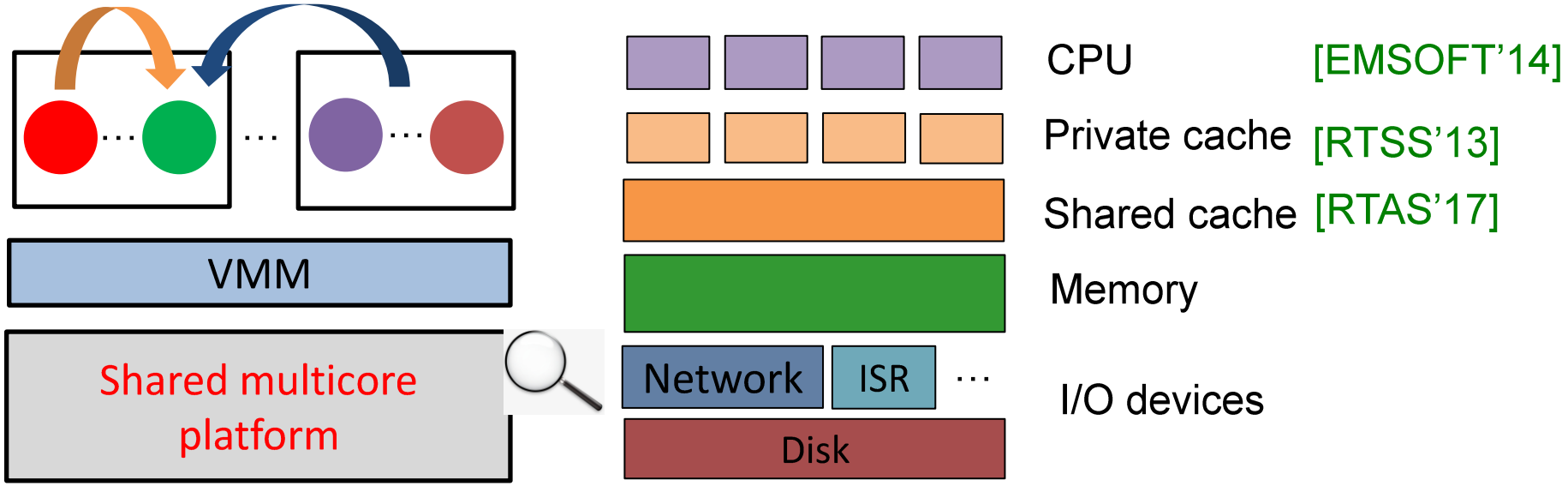
Trend: Multicore & Virtualization

- Automotive systems are becoming more and more complex
  - Requires high performance and strong isolation
- Virtualization on multicore help handle such complexity
  - Increase performance and reduce cost



Real-time virtualization

- **Challenge**: Interference affects the timing isolation
- Shared components are the source of interferences
- **Real-time virtualization** aims to achieve timing isolation
  - e.g., Real-Time Xen (RT-Xen)



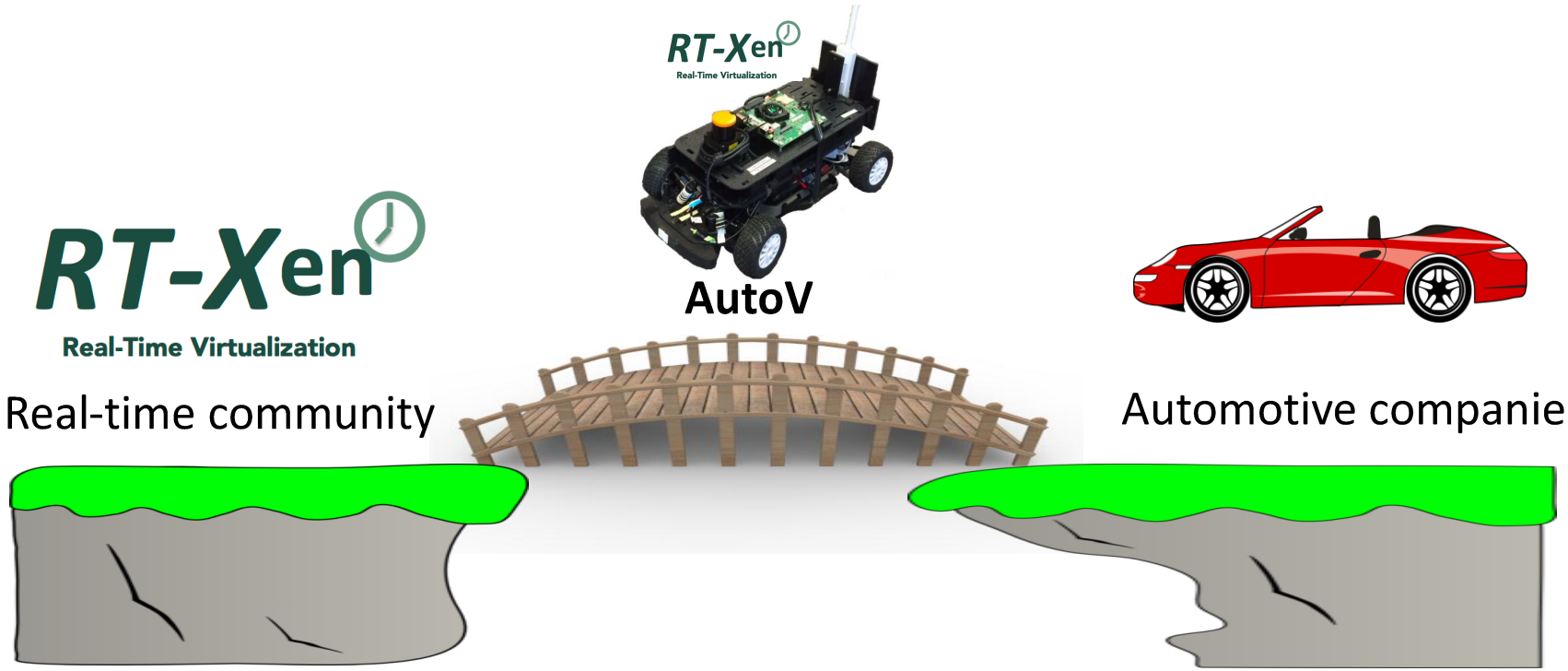
Real-time virtualization

- **Challenge**: Interference affects the timing isolation
- Shared components are the source of interferences
- **Real-time virtualization** aims to achieve timing isolation

Does real-time virtualization provide the timing isolation needed for **automotive systems**?

How to evaluate?

- Real-time community has real-time virtualization, but has no access to
  - Real automotive platforms
  - Real automotive applications



AutoV: An automotive testbed for real-time virtualization

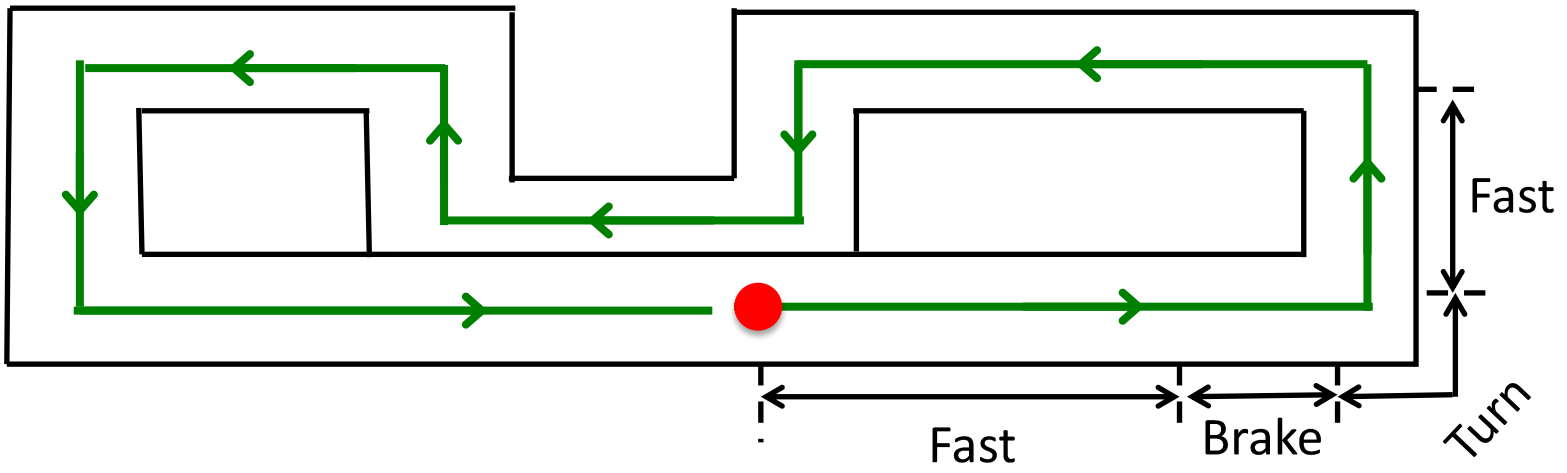
- AutoV
  - Evaluate the timing isolation provided by the real-time virtualization for automotive
- Autonomous racing car
  - Provide automotive applications as the evaluation workloads
- RT-Xen: Real-time Xen
  - Provide state-of-art timing isolation mechanisms for virtualization

Content

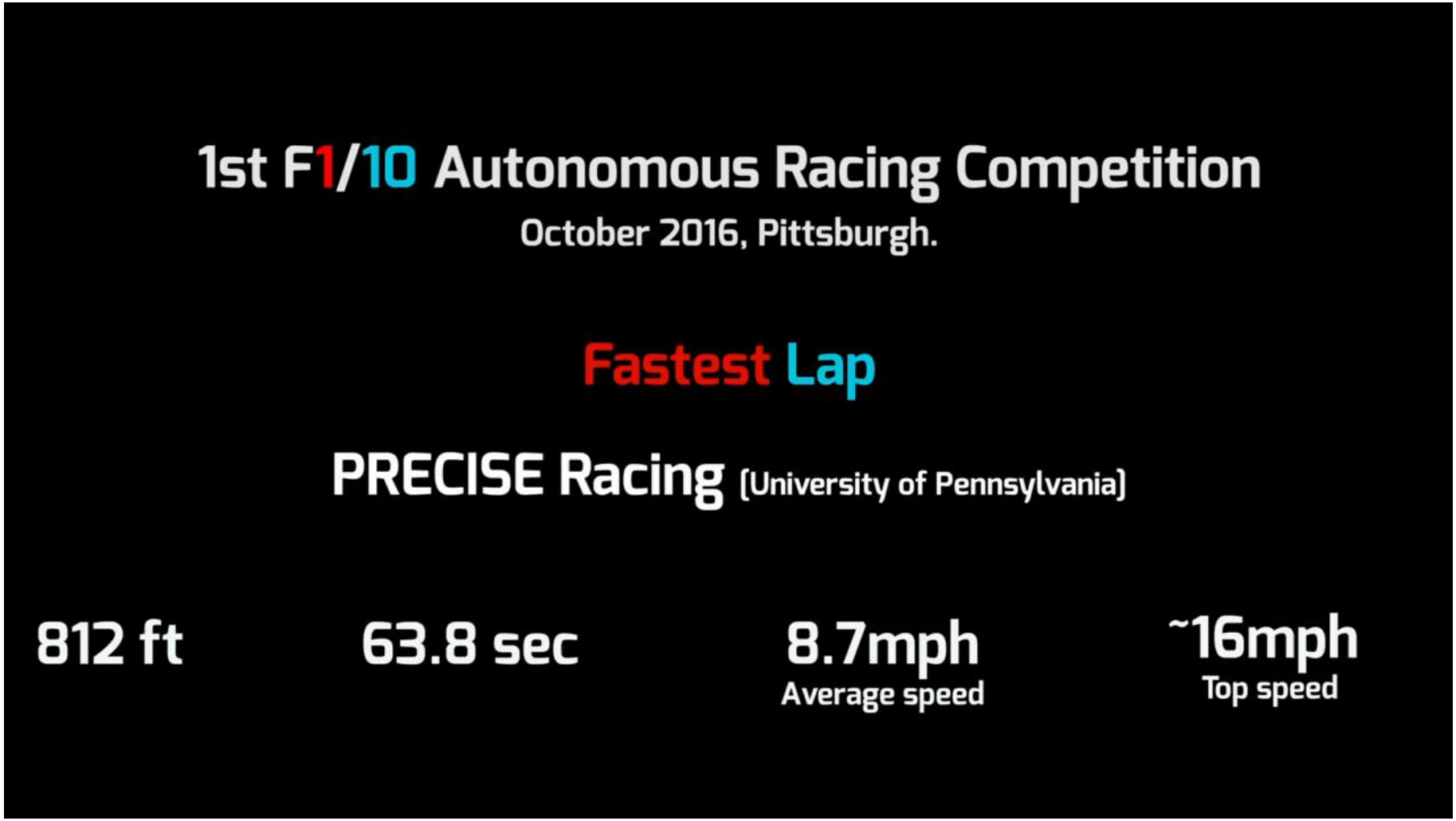
- Introduction
- **Autonomous racing car**
- RT-Xen
- AutoV
- Evaluation

Autonomous racing car

- F1/10 racing car developed by PRECISE Racing
  - Fast mode: Drive straight with PD controller
  - Brake mode: Anti-lock braking
  - Turn mode: Detect the corner and apply a constant steering angle

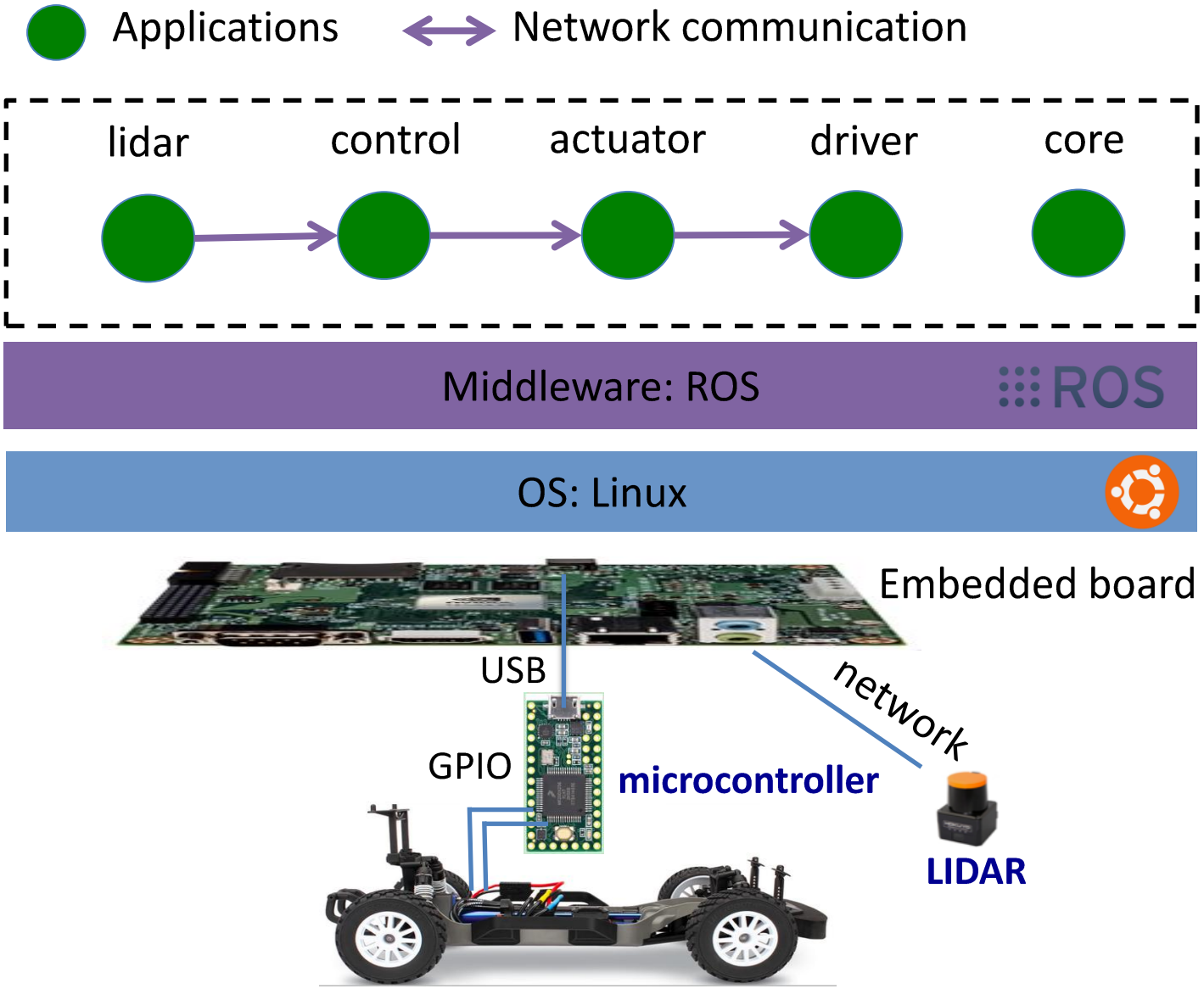


F1/10 racing car from PRECISE Racing



This video was captured by F1/10 organizer in ESWeek' 16.

Architecture of the racing car

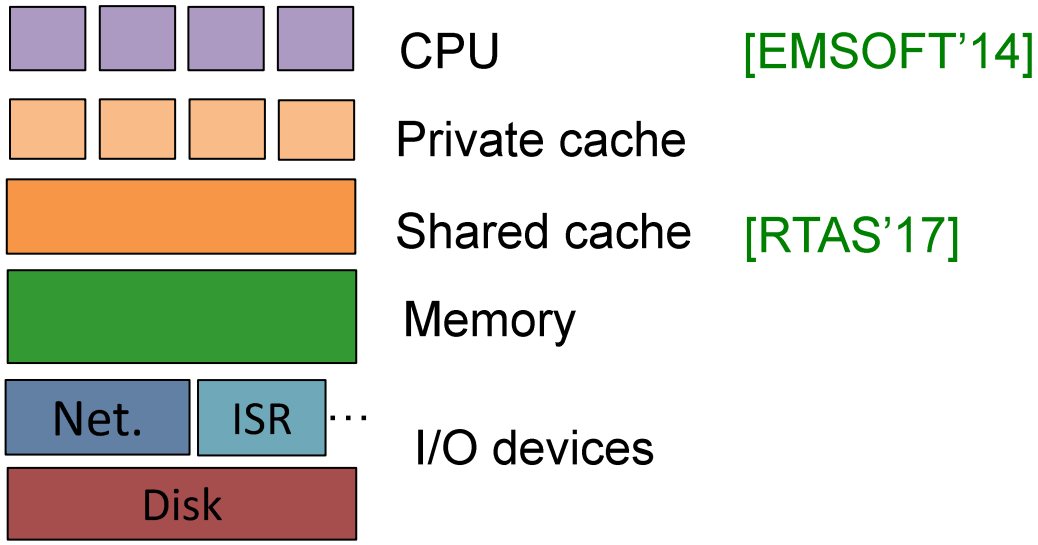


Content

- Introduction
- Autonomous racing car
- **RT-Xen**
- AutoV
- Evaluation

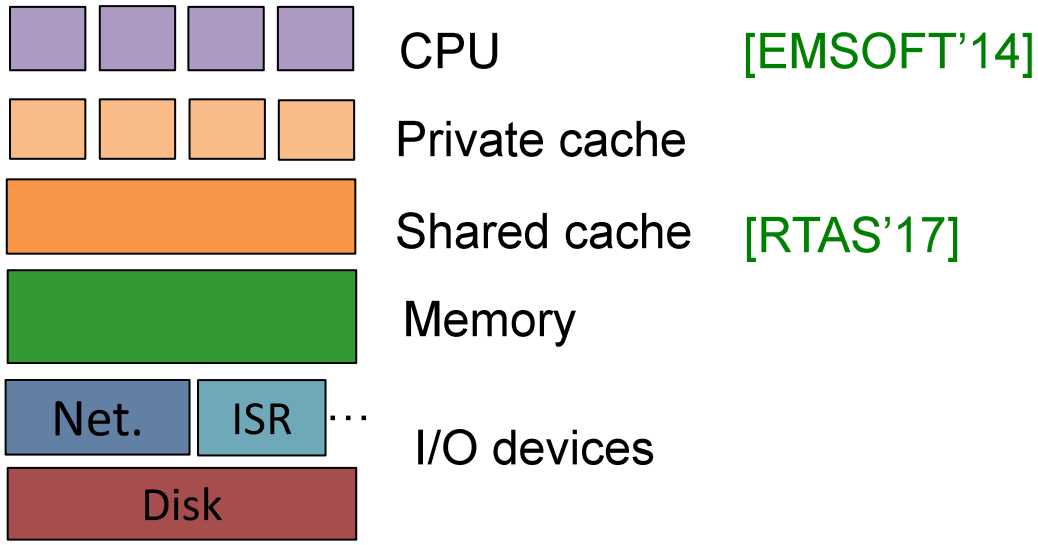
Real-time capabilities of RT-Xen

- Real-time CPU scheduling [EMOSFT' 14]
  - Global / Partitioned, EDF / RM
  - Global EDF: RTDS scheduler since Xen 4.5
- Dynamic shared cache management [RTAS' 17]
  - Strong isolation in last level cache (LLC)
  - Dynamically reconfigure LLC when demand changes
  - Based on Intel Cache Allocation Technology hardware



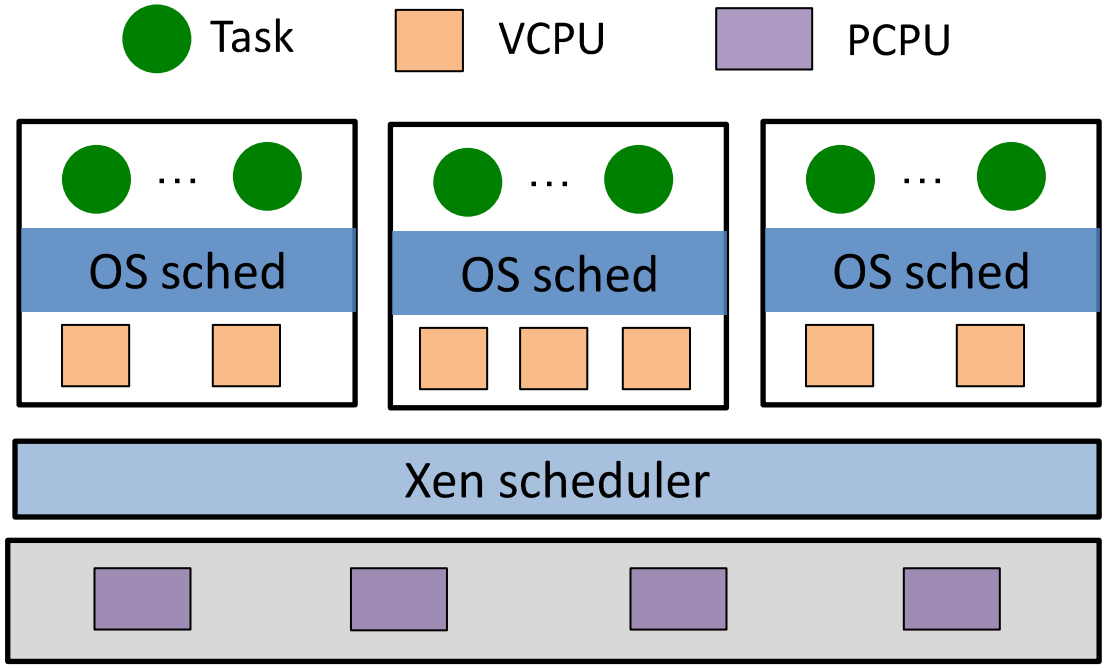
Real-time capabilities of RT-Xen

- **Real-time CPU scheduling [EMOSFT' 14]**
  - Global / Partitioned, EDF / RM
  - Global EDF: RTDS scheduler since Xen 4.5
- Dynamic shared cache management [RTAS' 17]
  - Strong isolation in last level cache (LLC)
  - Dynamically reconfigure LLC when demand changes
  - Based on Intel Cache Allocation Technology hardware



Xen scheduler

- Xen scheduler
  - VM runs on VCPUs
  - Xen schedules VCPUs on PCPUs
- Credit scheduler: Round-robin with proportional share
  - No real-time guarantee for VCPUs

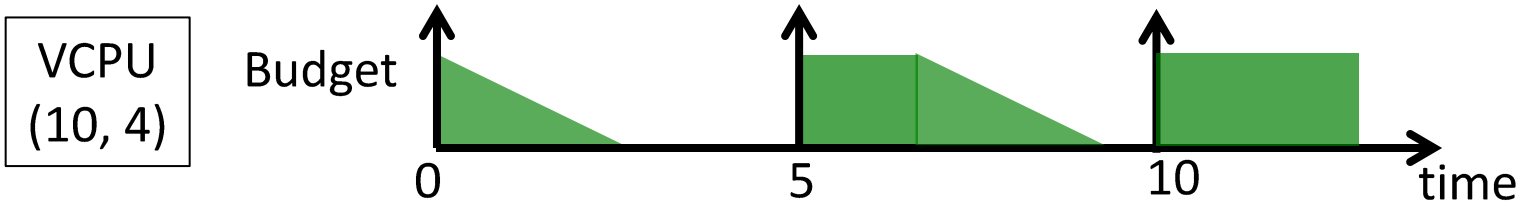


RT-Xen scheduler

- A set of real-time schedulers
  - Earliest Deadline First (EDF) and Rate Monotonic (RM) algorithm
  - Global and partitioned scheduling
- Global EDF has been incorporated in Xen as the RTDS scheduler since 2015
  - RTDS: Real-Time Deferrable Server
- RTDS scheduler is **re-written** for Xen 4.7
  - Time-driven scheduling Event-driven scheduling
  - Less implementation overhead

RTDS scheduler

- A VCPU is specified as (period, budget)
  - The VCPU will get budget time at the beginning of every period
  - The VCPU's budget decreases when a task runs on it

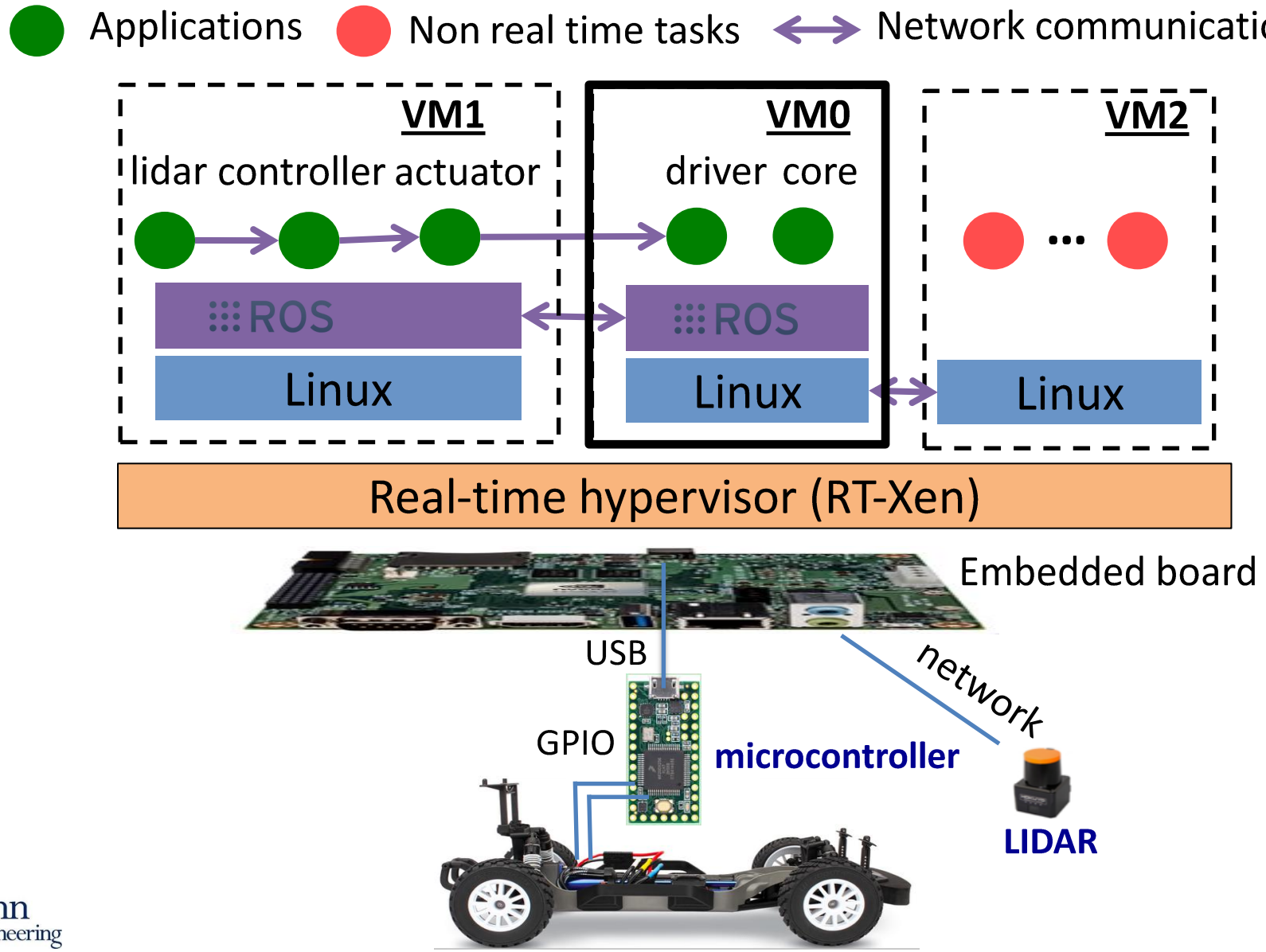


- The scheduler schedules VCPUs based on priority
  - The VCPU with earlier deadline has higher priority
  - Ready queue holds all VCPUs with budget
  - Depleted queue holds VCPUs that run out of budget in the period

Content

- Introduction
- Autonomous racing car
- RT-Xen
- **AutoV**
- Evaluation

Architecture of the racing car on RT-Xen



Content

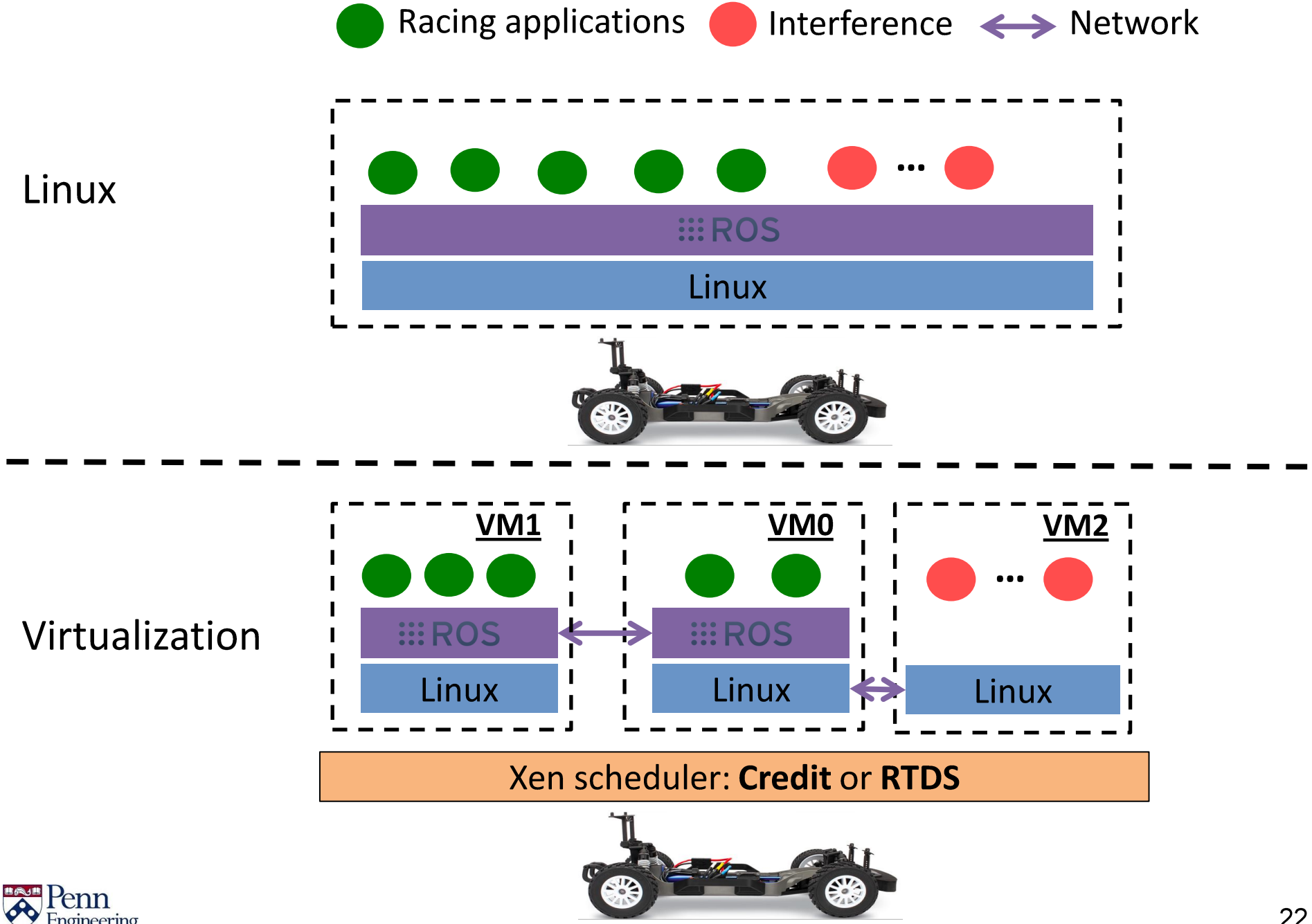
- Introduction
- Autonomous racing car
- RT-Xen
- AutoV
- **Evaluation**

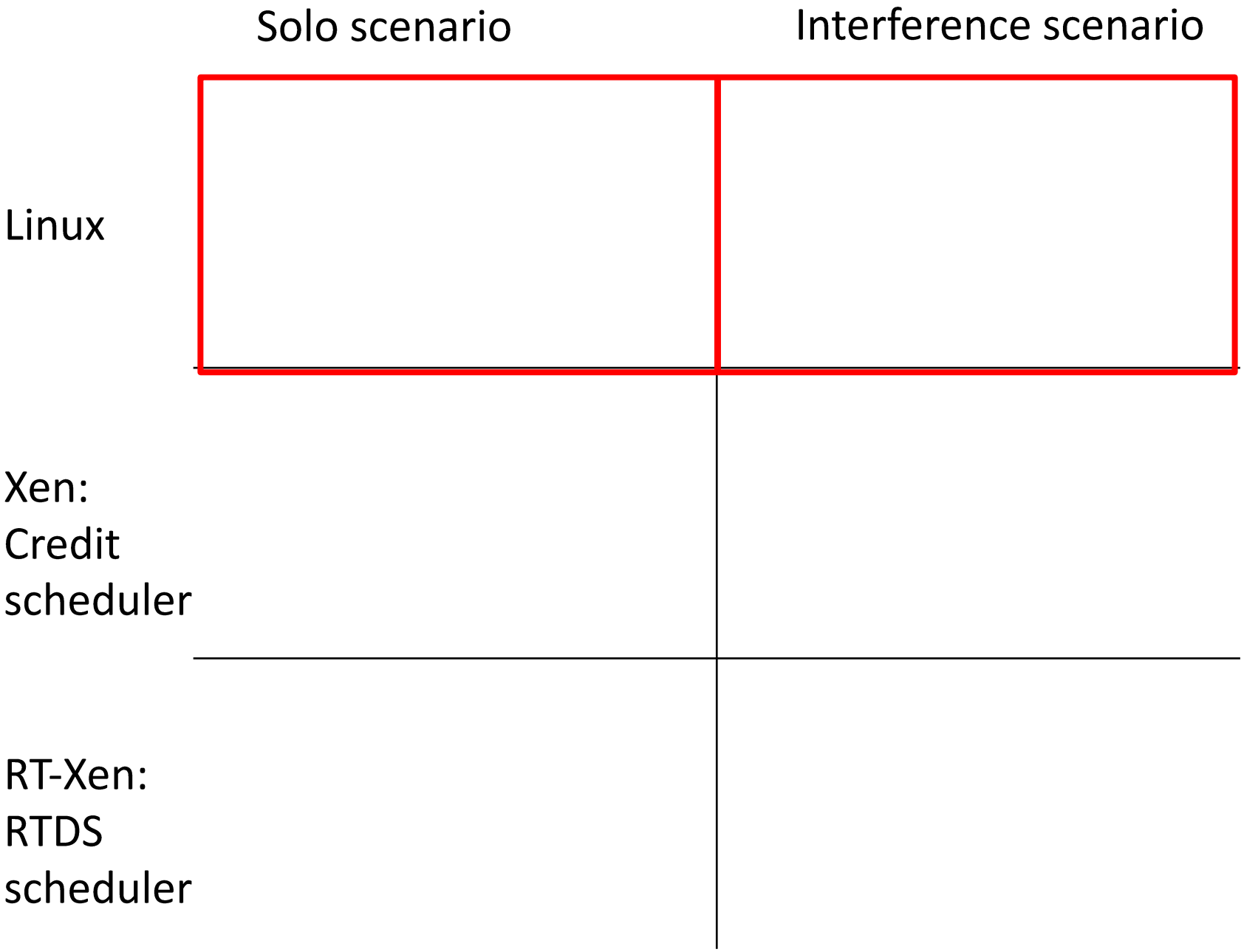
Evaluation goal

- Can safety-critical automotive applications run in virtualization environment?
- Can the interference jeopardize the safety of automotive applications?
- Can RT-Xen eliminate the interference and guarantee the safety?

Evaluation setup

- Straight hallway
- Expected car behavior
  - Drive straight for 5 seconds and brake
- Two types of workload
  - Racing car application
  - Interference tasks
- Interference task
  - CPU-intensive tasks
  - Future work: Consider other types of interferences (e.g., cache and memory)
- Two scenarios
  - Solo: When only the racing car application runs
  - Interference: When both the racing car application and the interference tasks run


Racing applications
Interference
Network
Linux
ROS
Linux
Virtualization
VM1
VM0
VM2
ROS
ROS
Linux
Linux
Linux
Xen scheduler: Credit or RTDS


Solo scenario
Interference scenario
Linux
Xen:
Credit
scheduler
RT-Xen:
RTDS
scheduler

Solo scenario

Interference scenario

Linux

Penn Engineering

	Solo scenario	Interference scenario
Linux		
Xen: Credit scheduler		
RT-Xen: RTDS scheduler		

Solo scenario

Interference scenario

RT-Xen:
RTDS
scheduler

Penn Engineering

	Solo scenario	Interference scenario
Linux		
Xen: Credit scheduler		
RT-Xen: RTDS scheduler		

Evaluation goal

- Can safety-critical automotive applications run in virtualization environment?
  - **Yes**
- Can the interference jeopardize the safety of automotive applications?
  - **Yes**
- Can RT-Xen eliminate the interference and guarantee the safety?
  - **Yes**

Conclusion

- AutoV: An Automotive Testbed for Real-Time Virtualization
- Evaluation demonstrates
  - Automotive systems' integration requires the timing isolation
  - Real-time virtualization is a promising technique to achieve the timing isolation
- Challenges
  - How much evaluation should be done to claim that the real-time virtualization can be used for automotive systems?
  - How to manage the other resources, e.g., I/O, network?
- Future work
  - Evaluate other mechanisms of RT-Xen, e.g., dynamic cache management
  - Make it available to others so that our community can develop more complex applications

Acknowledgement

- Autonomous racing car collaborators
  - University of Pennsylvania: Radoslav Ivanov, Nguyen Hung, Bipeen Acharya, Kyoungwon Kim, Sarvesh Patkar, James Weimer

- RT-Xen collaborators
  - University of Pennsylvania: Linh Thi Xuan Phan, Oleg Sokolsky
  - Washington University in St. Louis: Sisu Xi, Chenyang Lu, Chris Gill

- AutoV collaborators
  - University of Pennsylvania: Karthik Methuku, Nitesh Singh

Check out AutoV
https://sites.google.com/site/autovtestbed/

Thank You