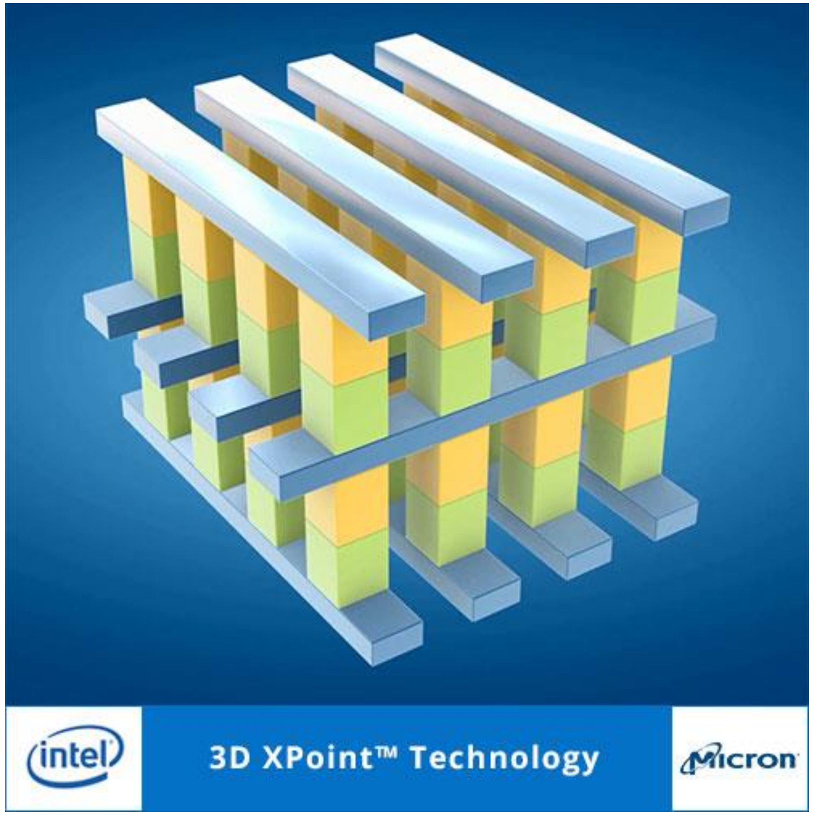
Correct and Fast Persistency Guarantees

Sara Mahdizadeh Shahri
Aasheesh Kolli

PennState

YArch Workshop - February 2019

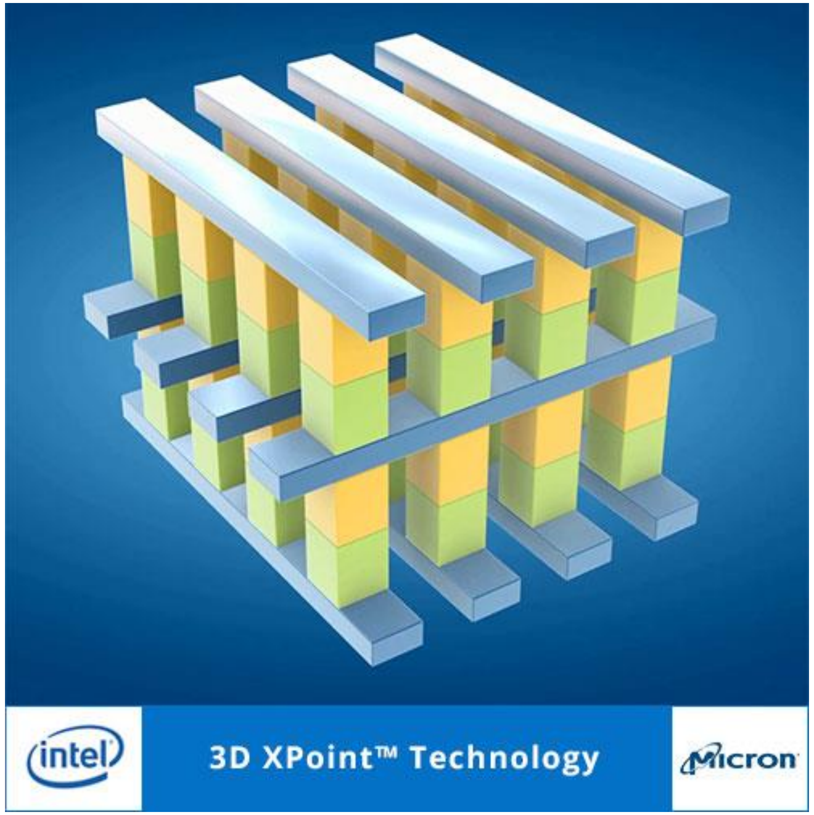
Why Persistent Memory?



PennState

Why Persistent Memory?

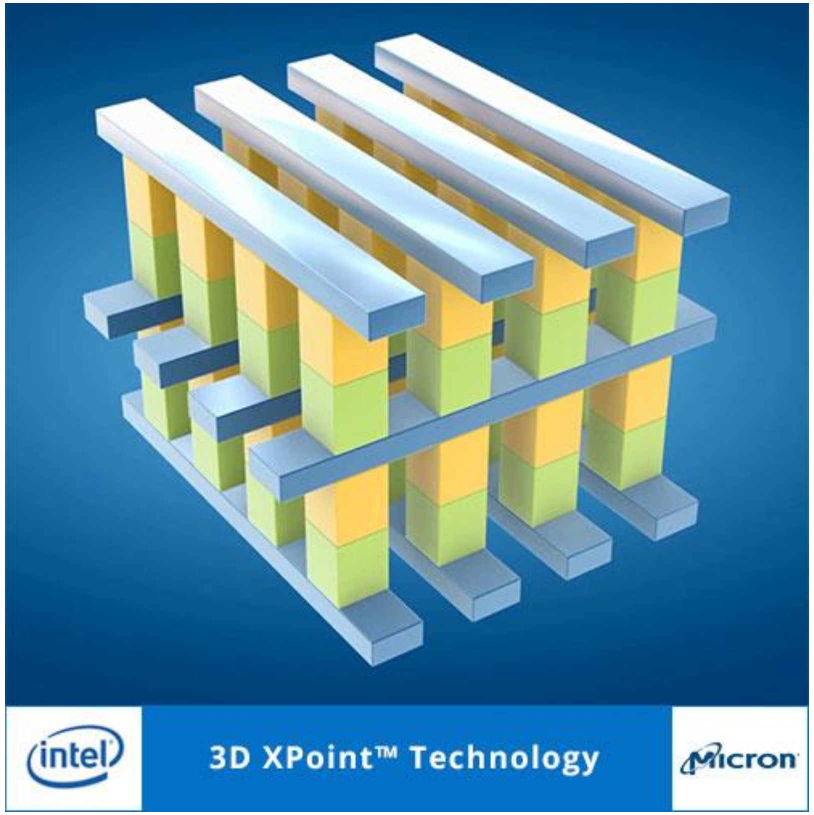
Non-volatility



PennState

Why Persistent Memory?

Non-volatility

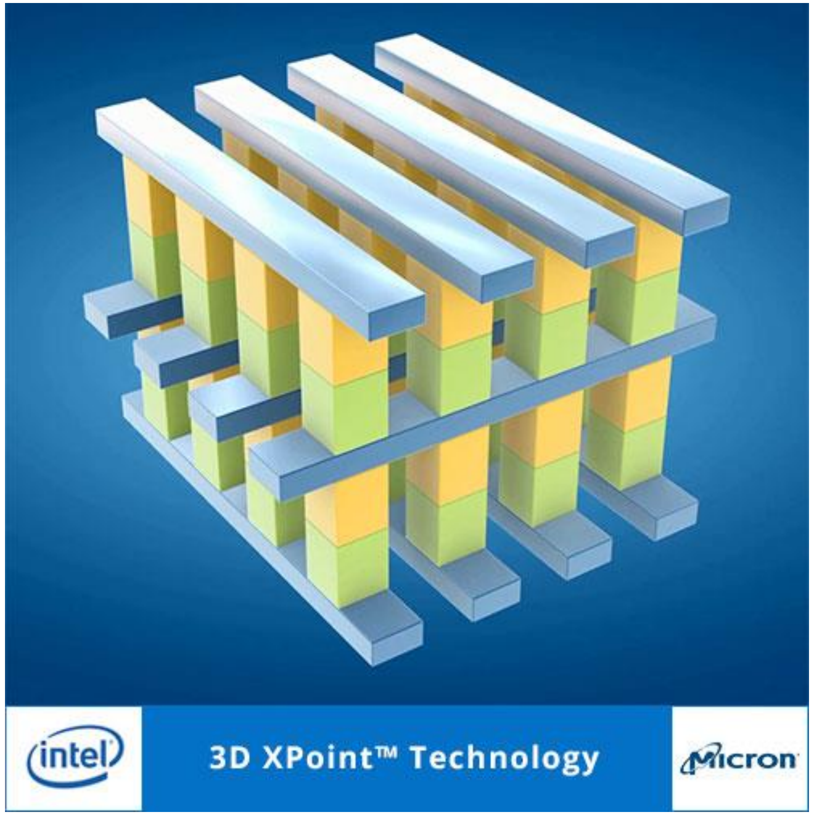


Density

Why Persistent Memory?

Byte-addressability

Non-volatility

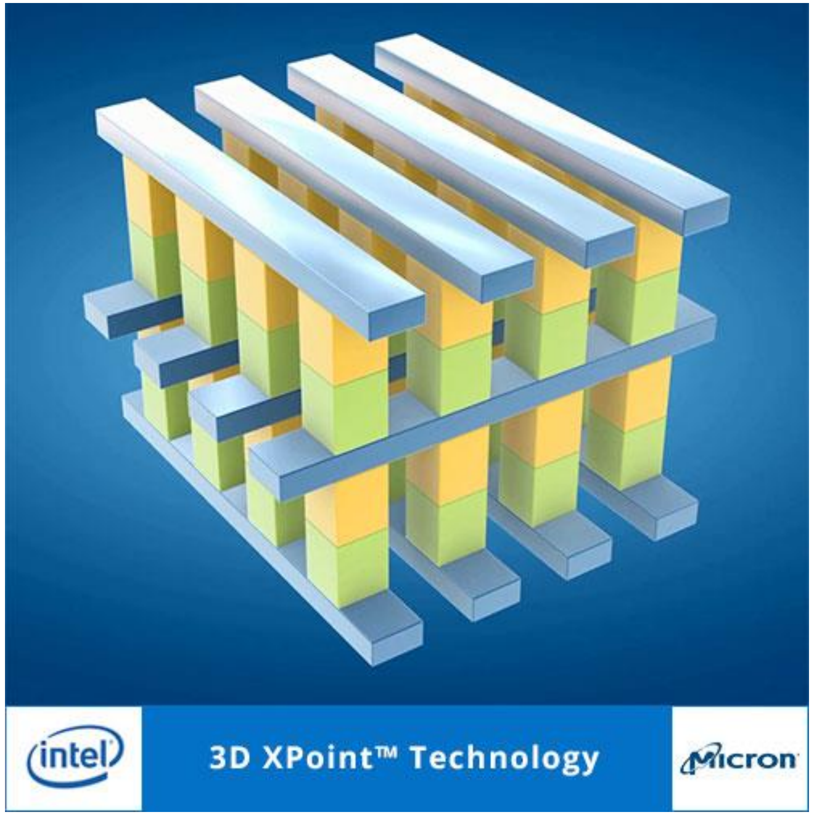


Density

Why Persistent Memory?

Byte-addressability

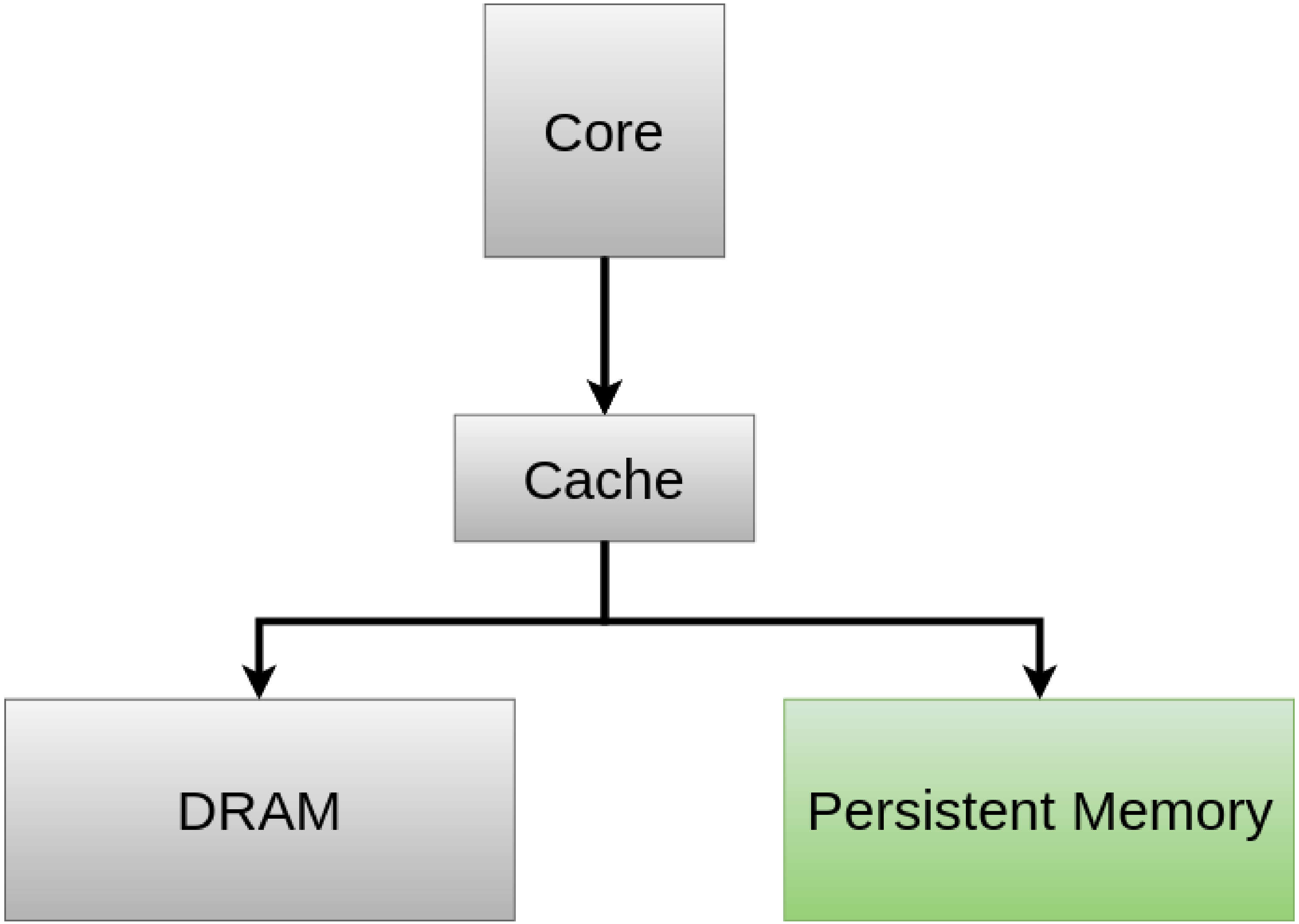
Non-volatility



Performance

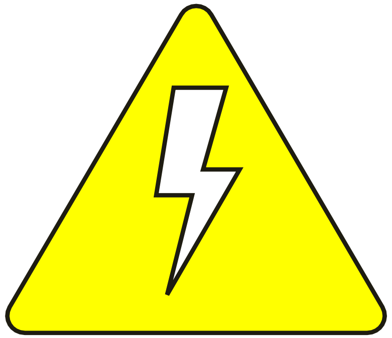
Density

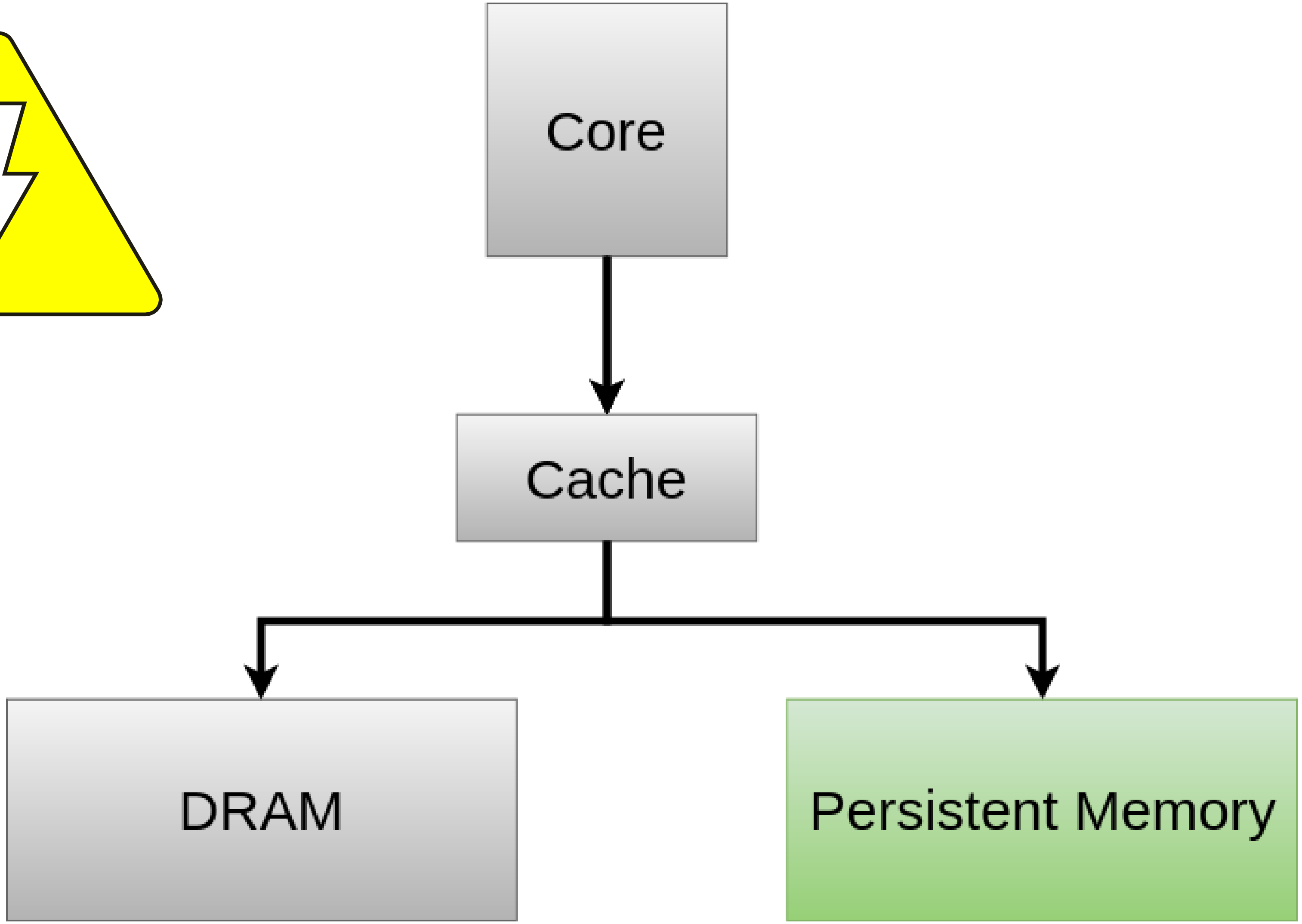
Memory Hierarchy



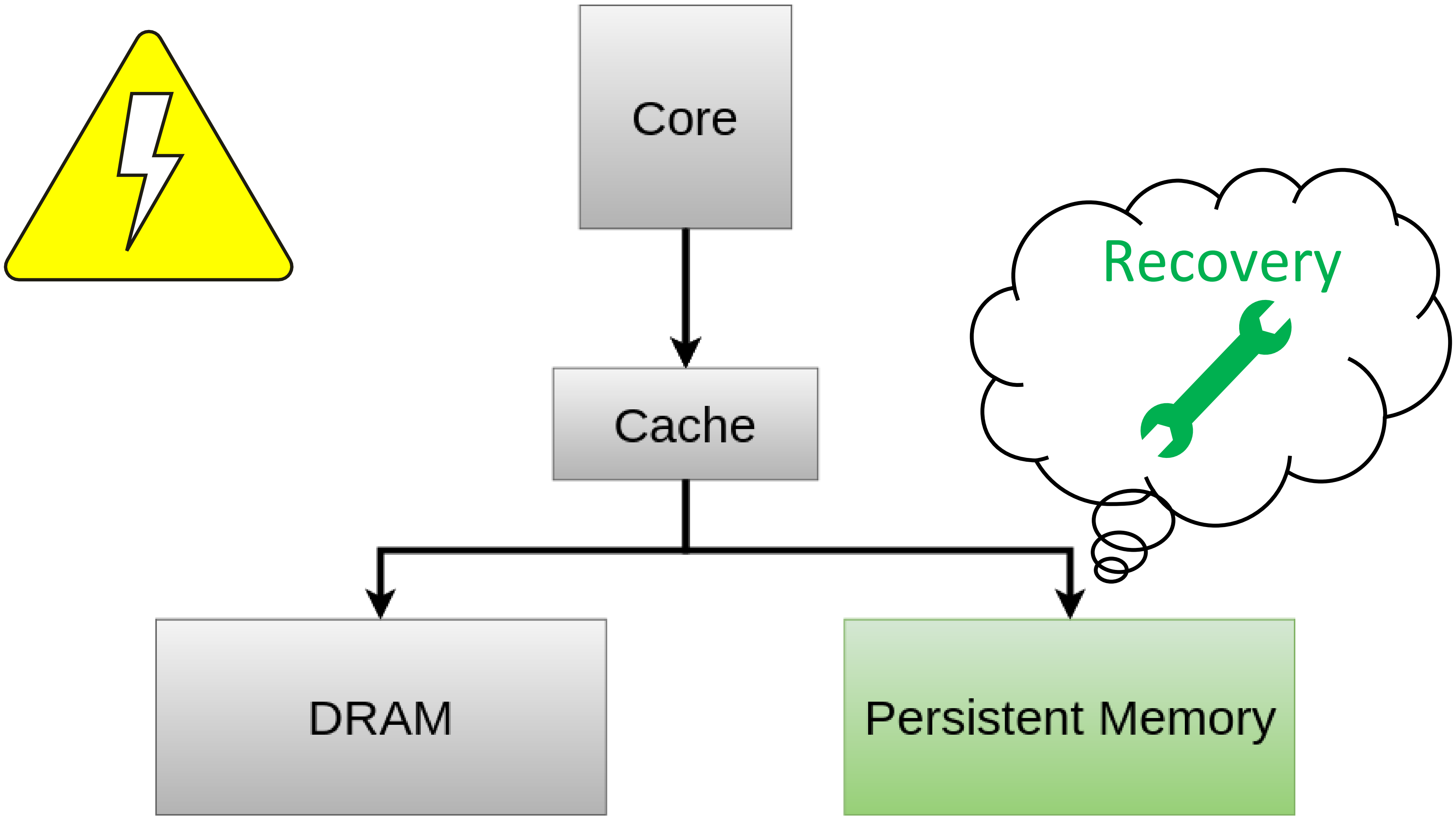
PennState

Memory Hierarchy





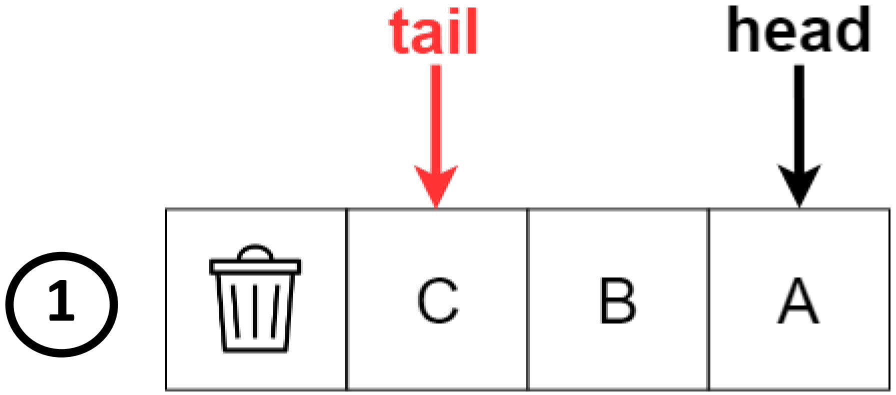
Memory Hierarchy



PennState

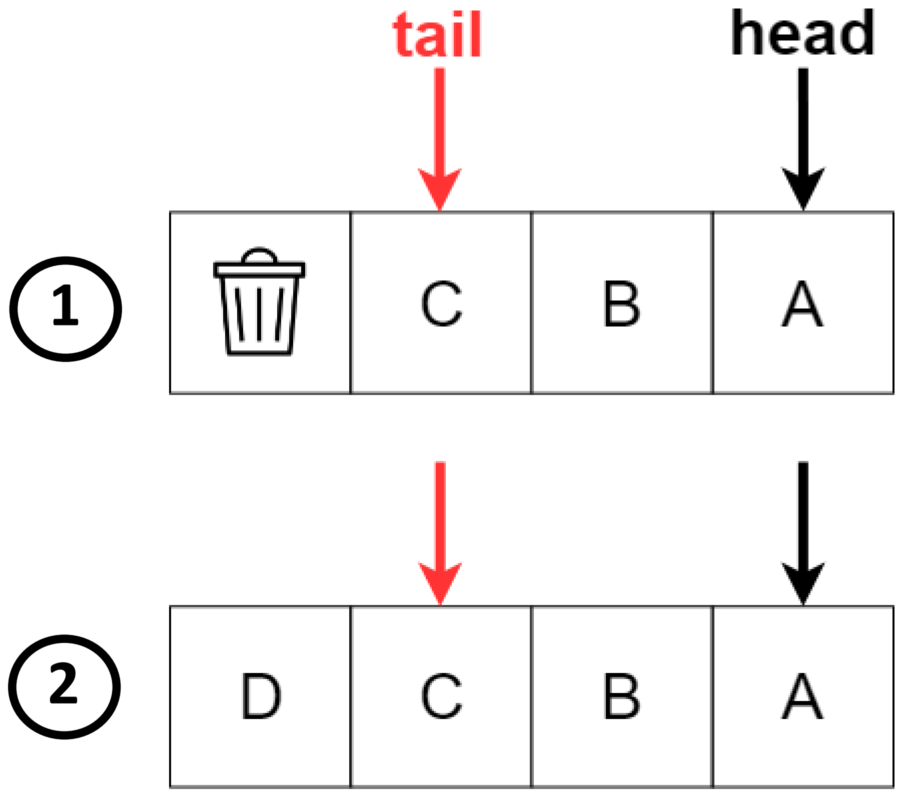
Correct Recovery

Use Case: Insertion in Queue



Correct Recovery

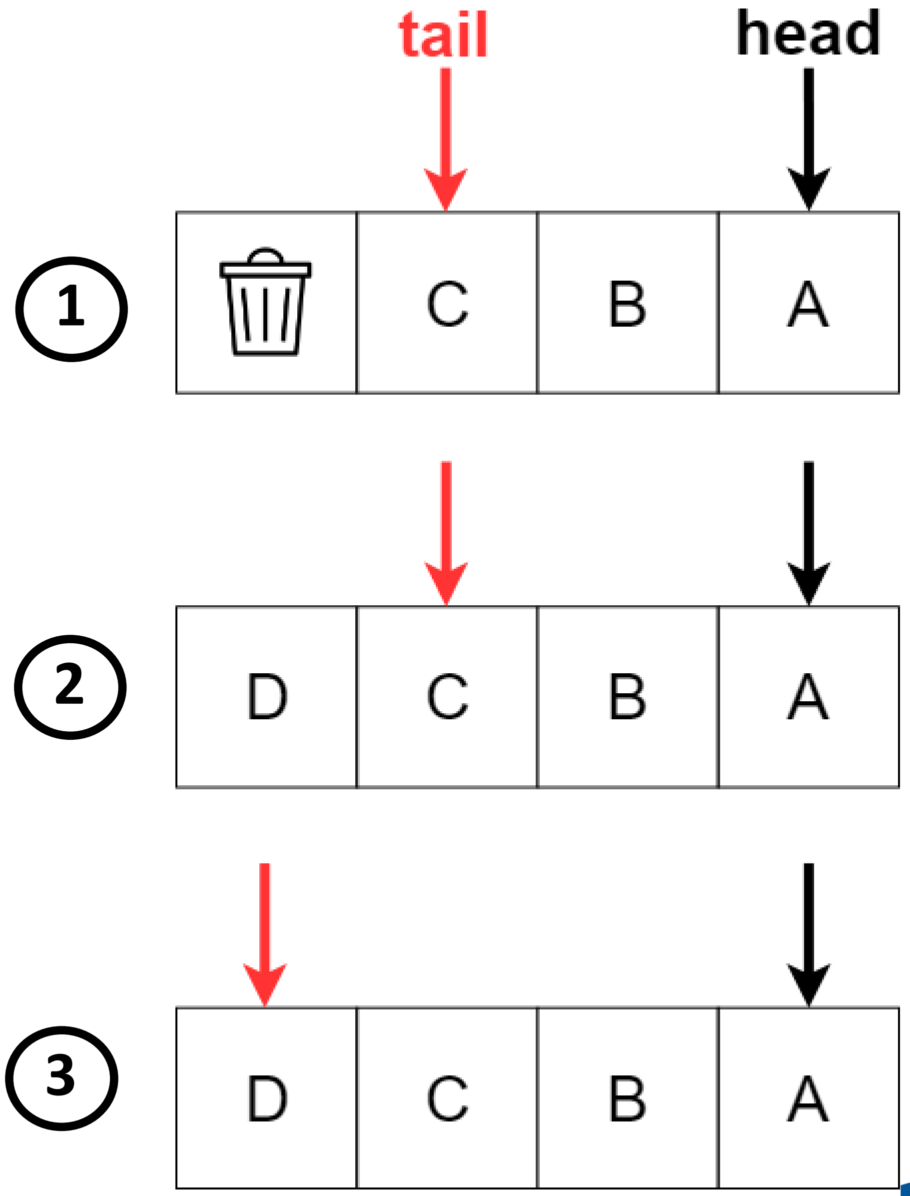
Use Case: Insertion in Queue



PennState

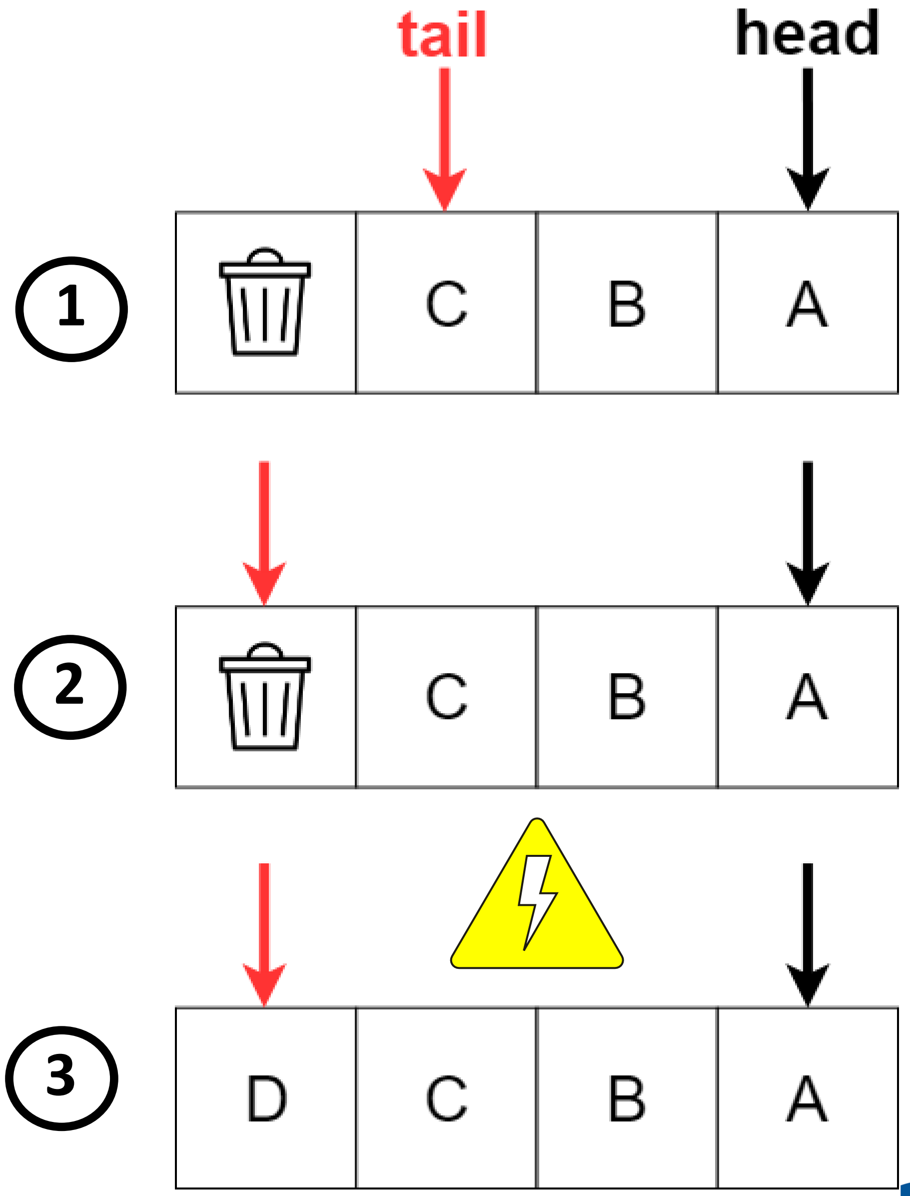
Correct Recovery

Use Case: Insertion in Queue



Correct Recovery

Use Case: Insertion in Queue

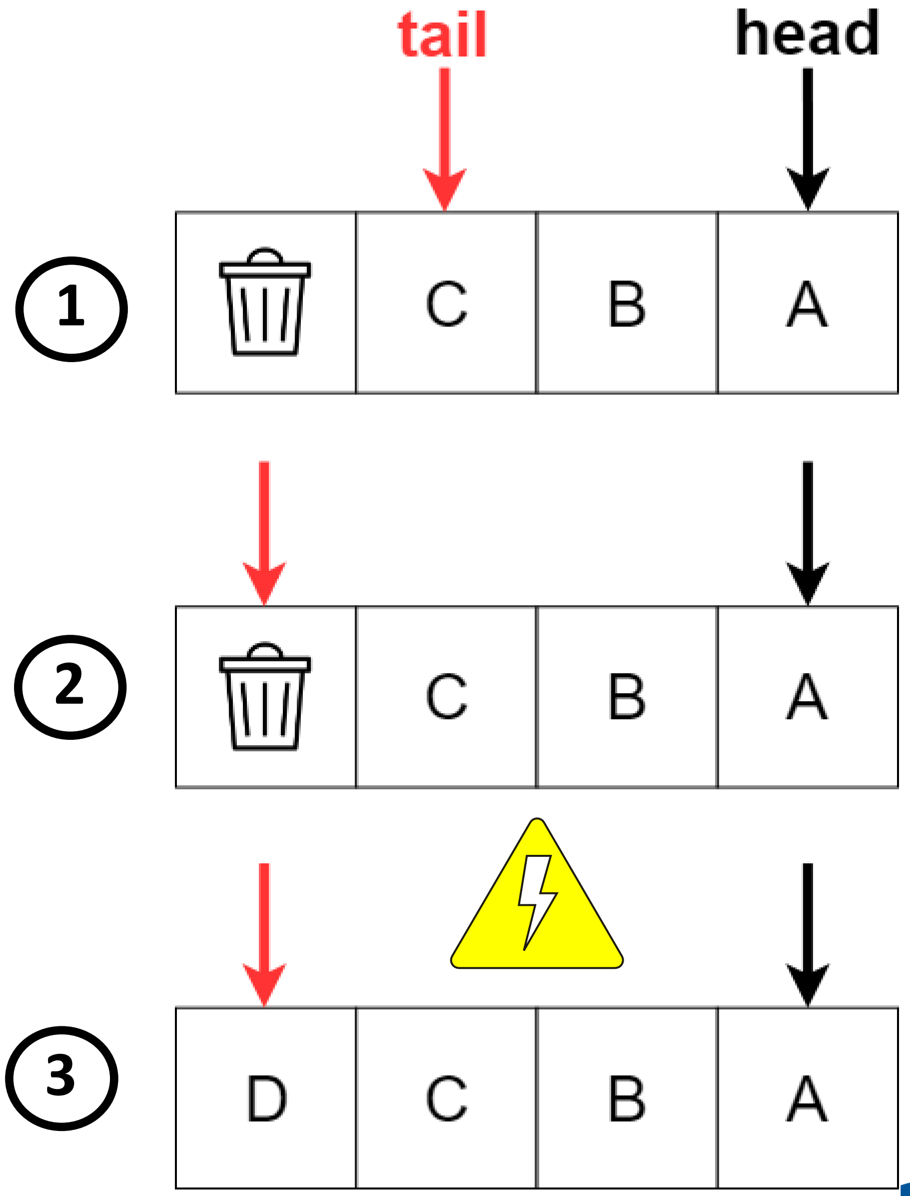


PennState

Correct Recovery

Use Case: Insertion in Queue

☑ **Order of Updates**

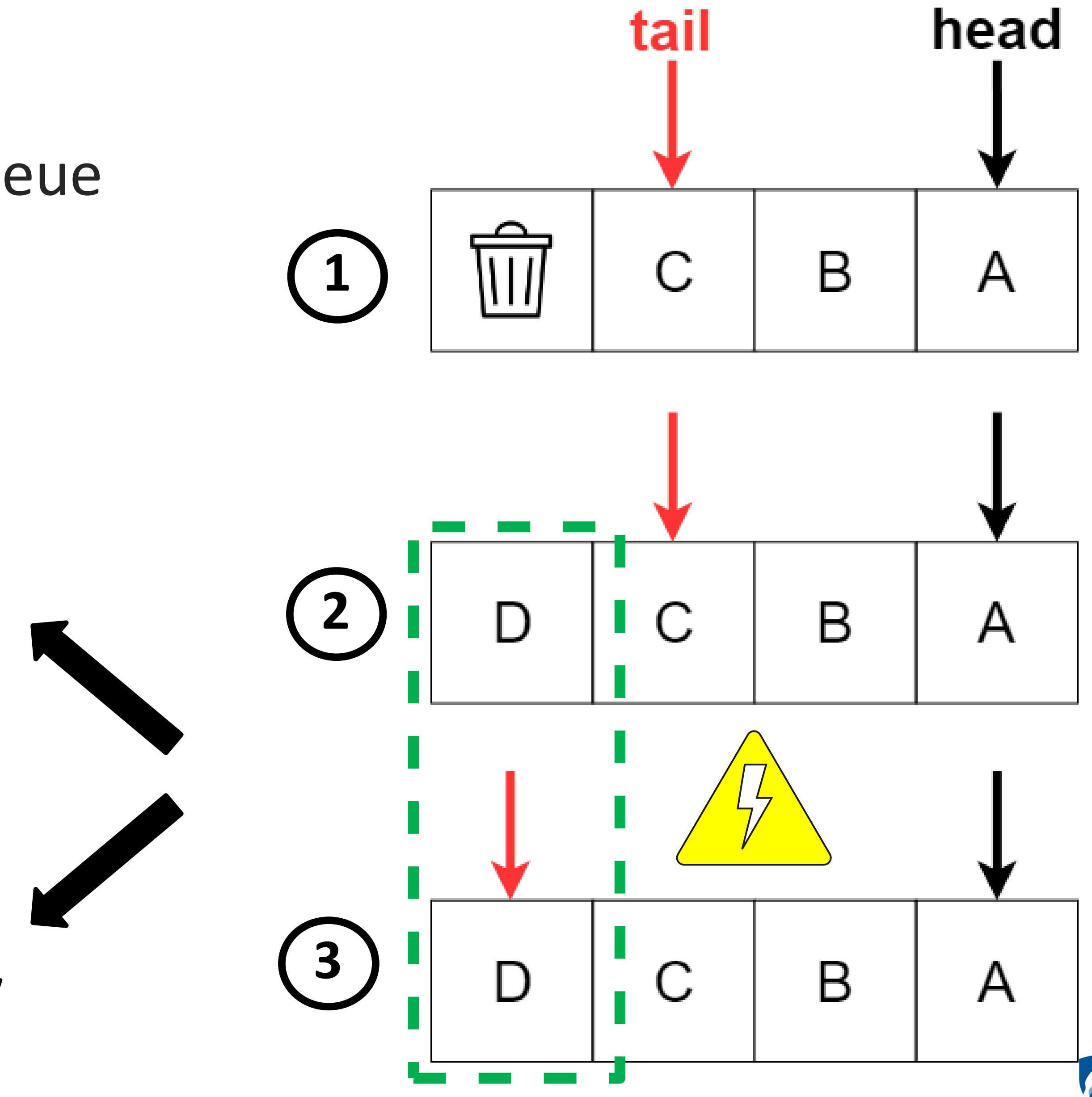


Correct Recovery

Use Case: Insertion in Queue

☑ **Order of Updates**

☑ **Failure Atomicity**



Failure Atomicity Semantics

- Providing Failure Atomicity for Granularity Beyond Persist Access

Failure Atomicity Semantics

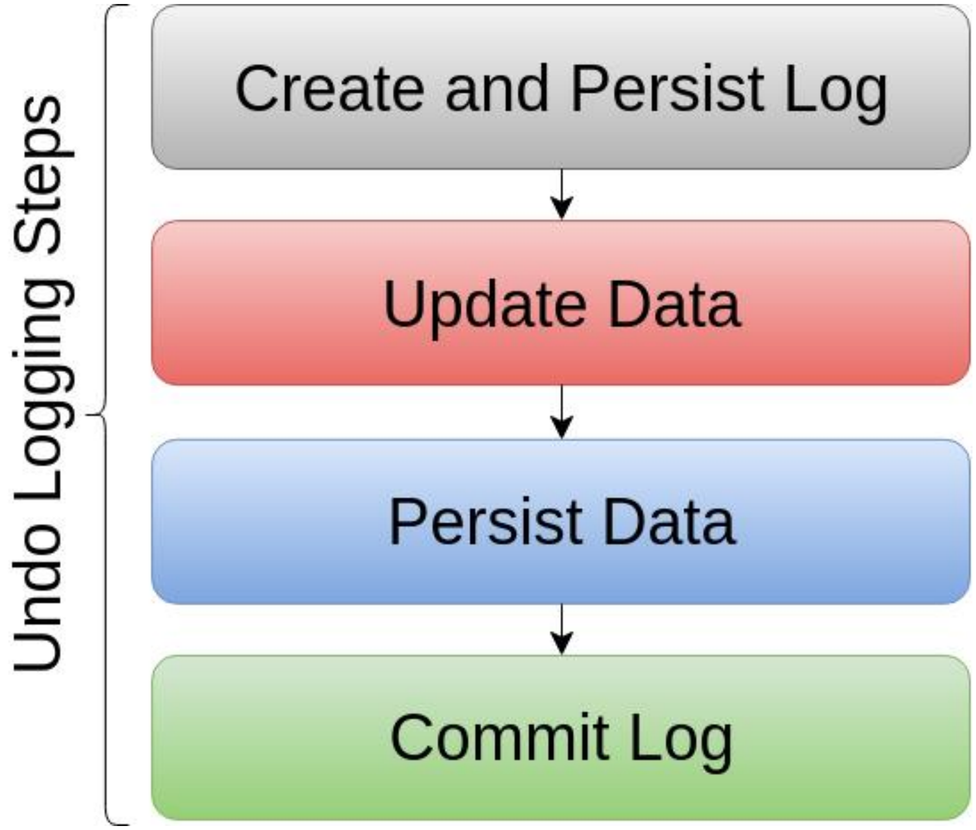
- Providing Failure Atomicity for Granularity Beyond Persist Access

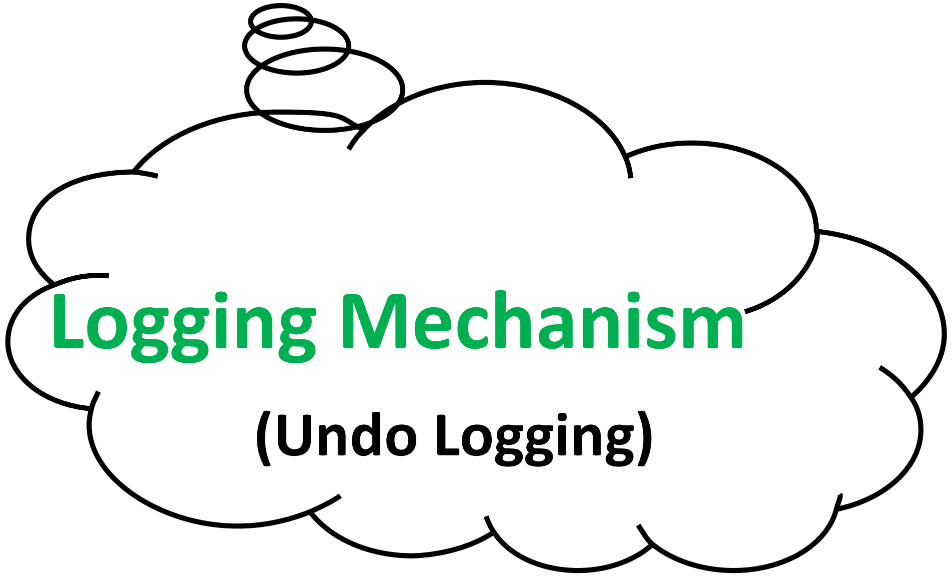
Logging Mechanism

(Undo Logging)

Failure Atomicity Semantics

- Providing Failure Atomicity for Granularity Beyond Persist Access





PennState

Semantics for Delivering Persistency
Challenge #1: Correctness

Prior Work

Atlas [Chakrabarti' 14]
Coupled/Decoupled SFR [Gogte' 18]

Semantics for Delivering Persistency
Challenge #1: Correctness

Prior Work

Atlas [Chakrabarti' 14]
Coupled/Decoupled SFR [Gogte' 18]

Detect Persist Updates at **Compile Time**

PennState

Semantics for Delivering Persistency
Challenge #1: Correctness

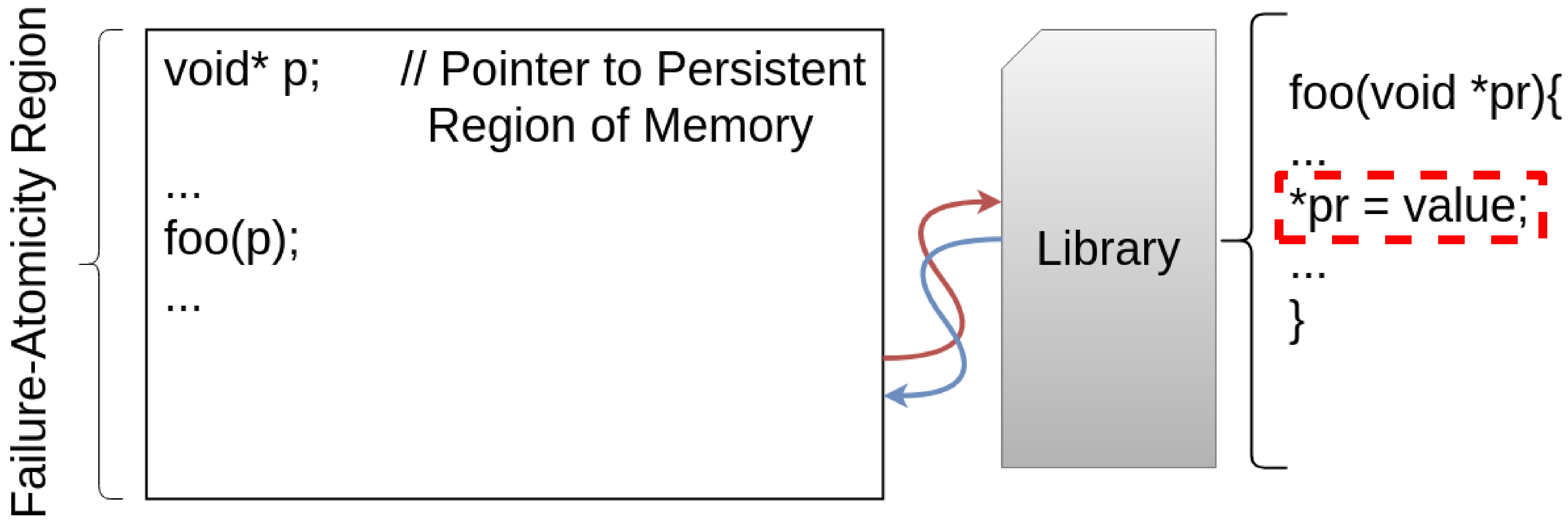
Prior Work

Atlas [Chakrabarti' 14]
Coupled/Decoupled SFR [Gogte' 18]

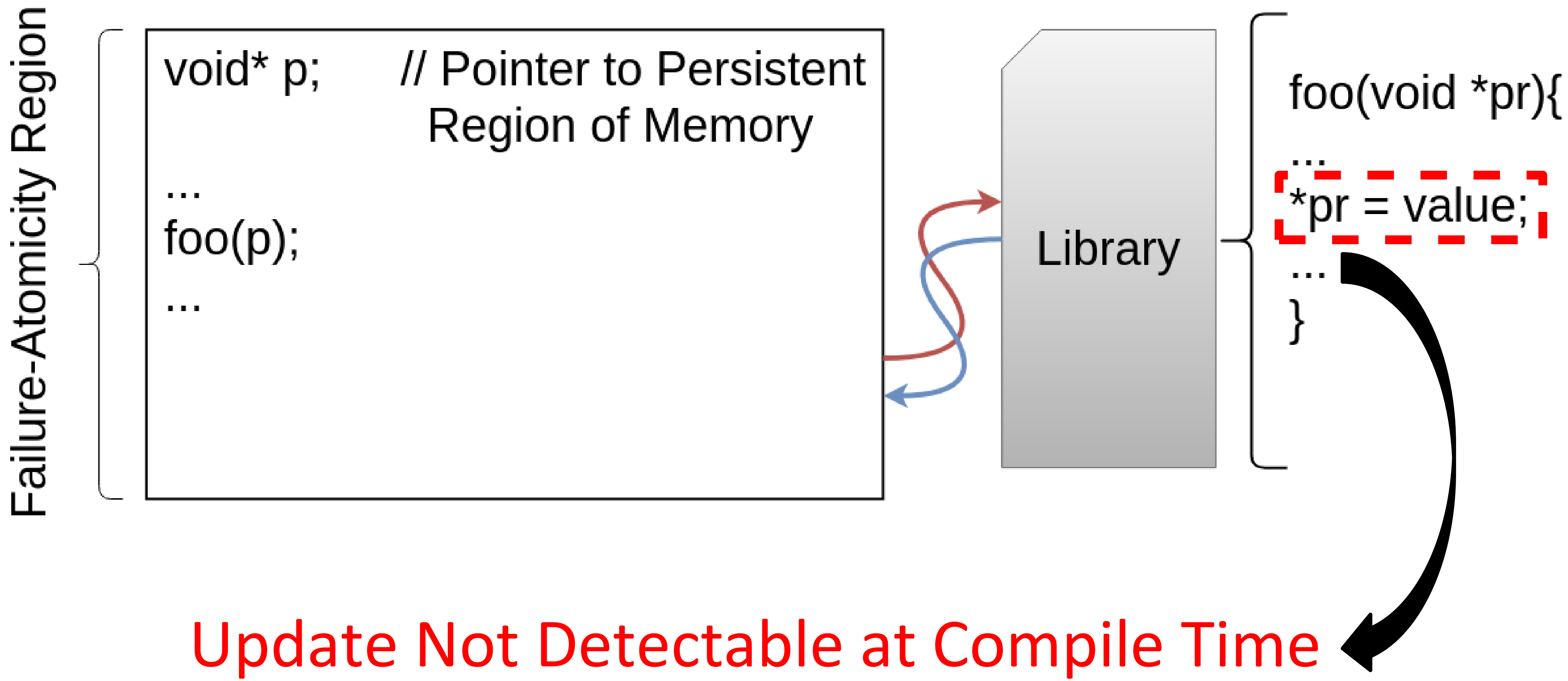
Detect Persist Updates at **Compile Time**

What Can Go Wrong?

Challenge #1: One Possible Scenario



Challenge #1: One Possible Scenario



Update Not Detectable at Compile Time

PennState

Semantics for Delivering Persistency
Challenge #2: Performance

High Logging Overheads (Atlas and Coupled SFR)

PennState

Semantics for Delivering Persistency
Challenge #2: Performance

High Logging Overheads (Atlas and Coupled SFR)

Decoupled SFR:
Background Threads for
Persisting Data and
Committing Log

Semantics for Delivering Persistency
Challenge #2: Performance

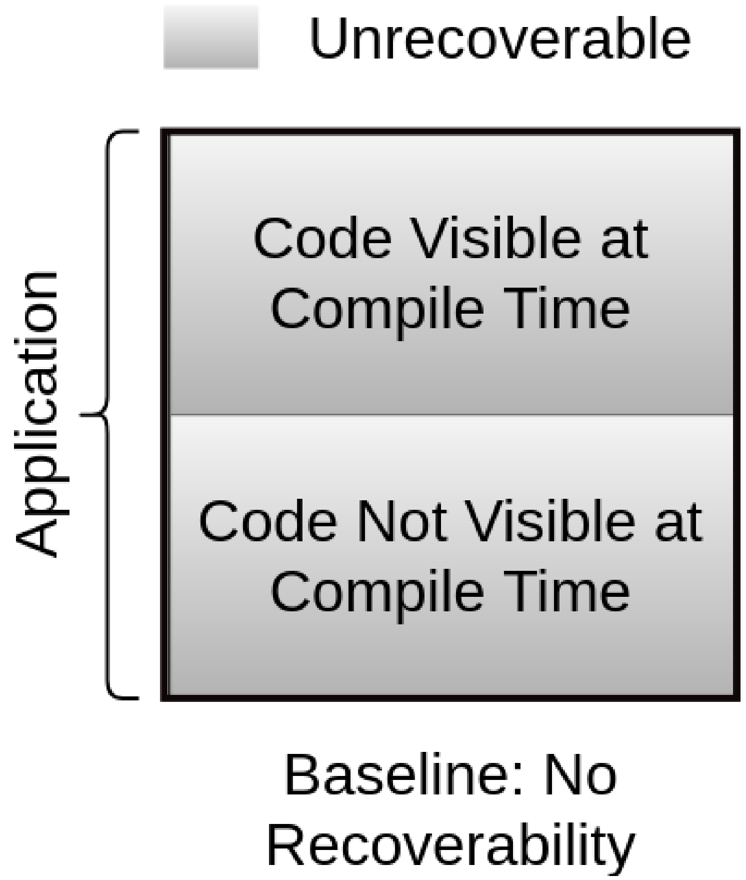
High Logging Overheads (Atlas and Coupled SFR)

Decoupled SFR: Background Threads for Persisting Data and Committing Log

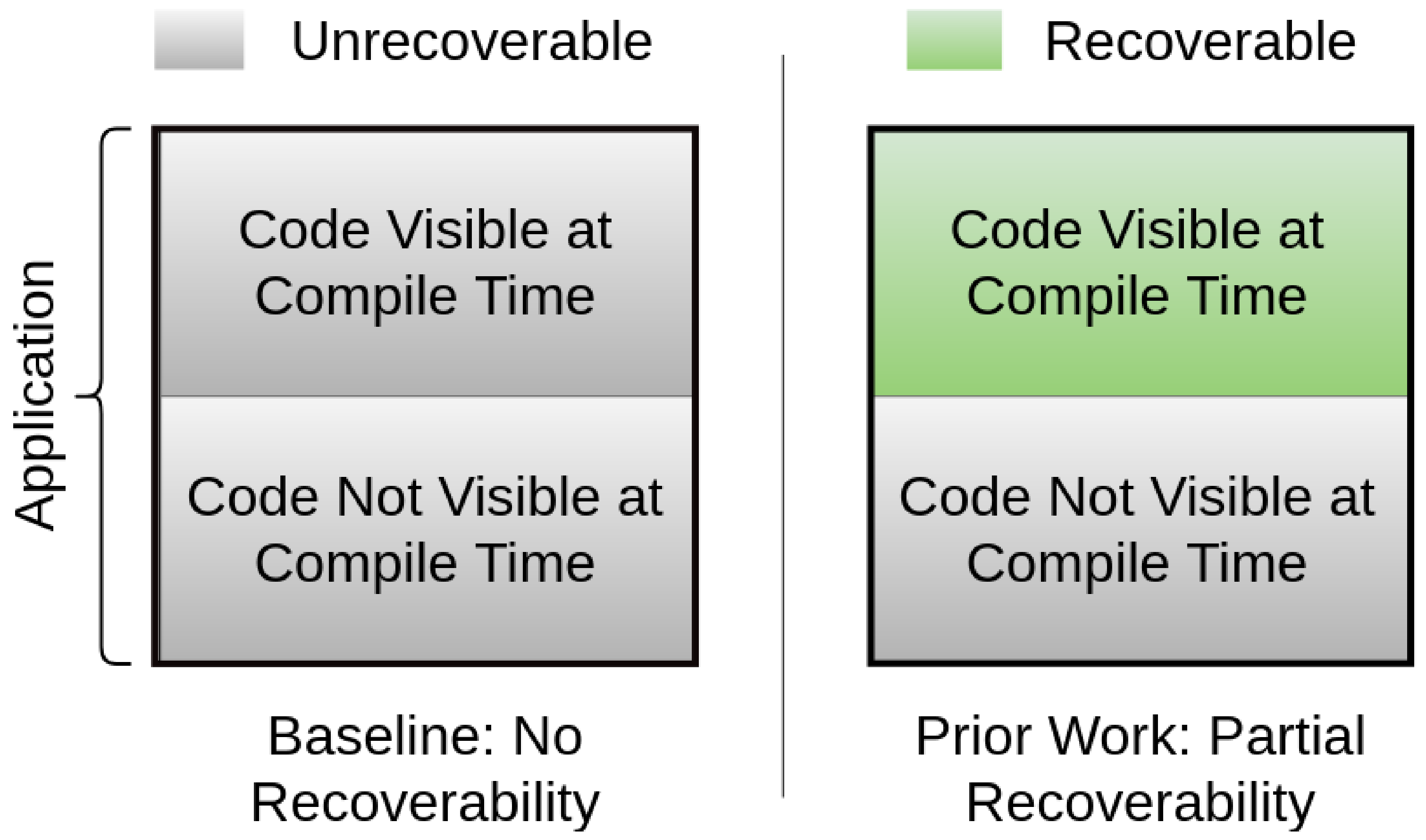
✖ **Utilization**

PennState

Idea #1: Persistency for the Whole Program

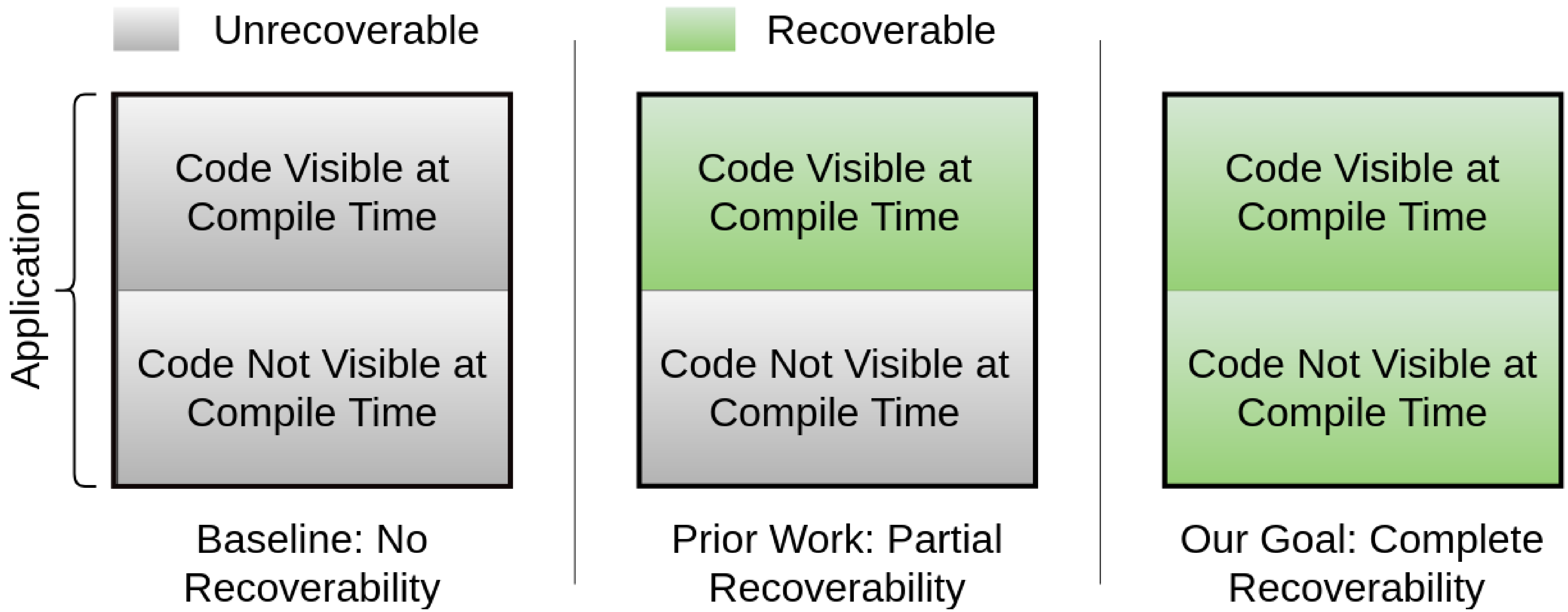


Idea #1: Persistency for the Whole Program

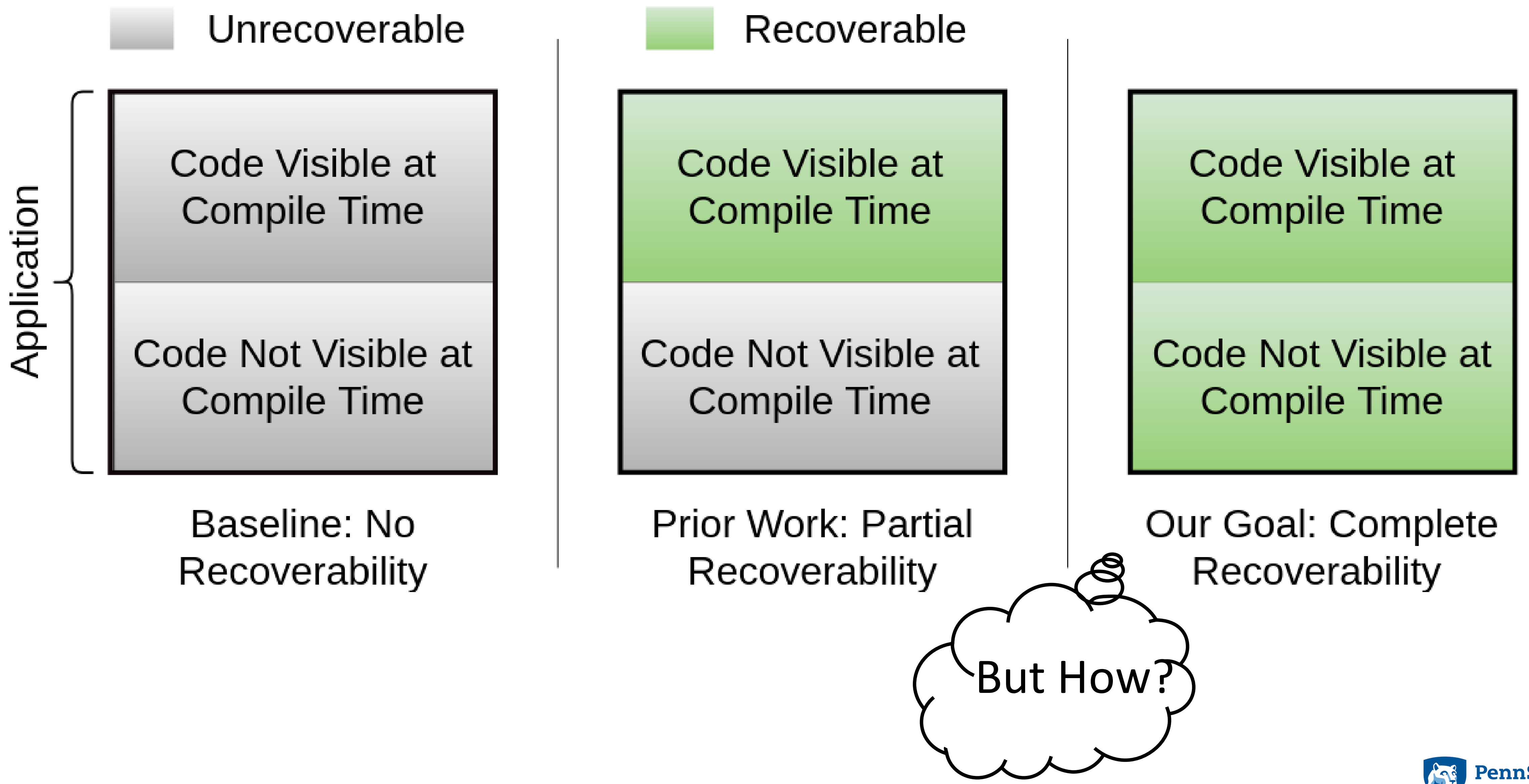


PennState

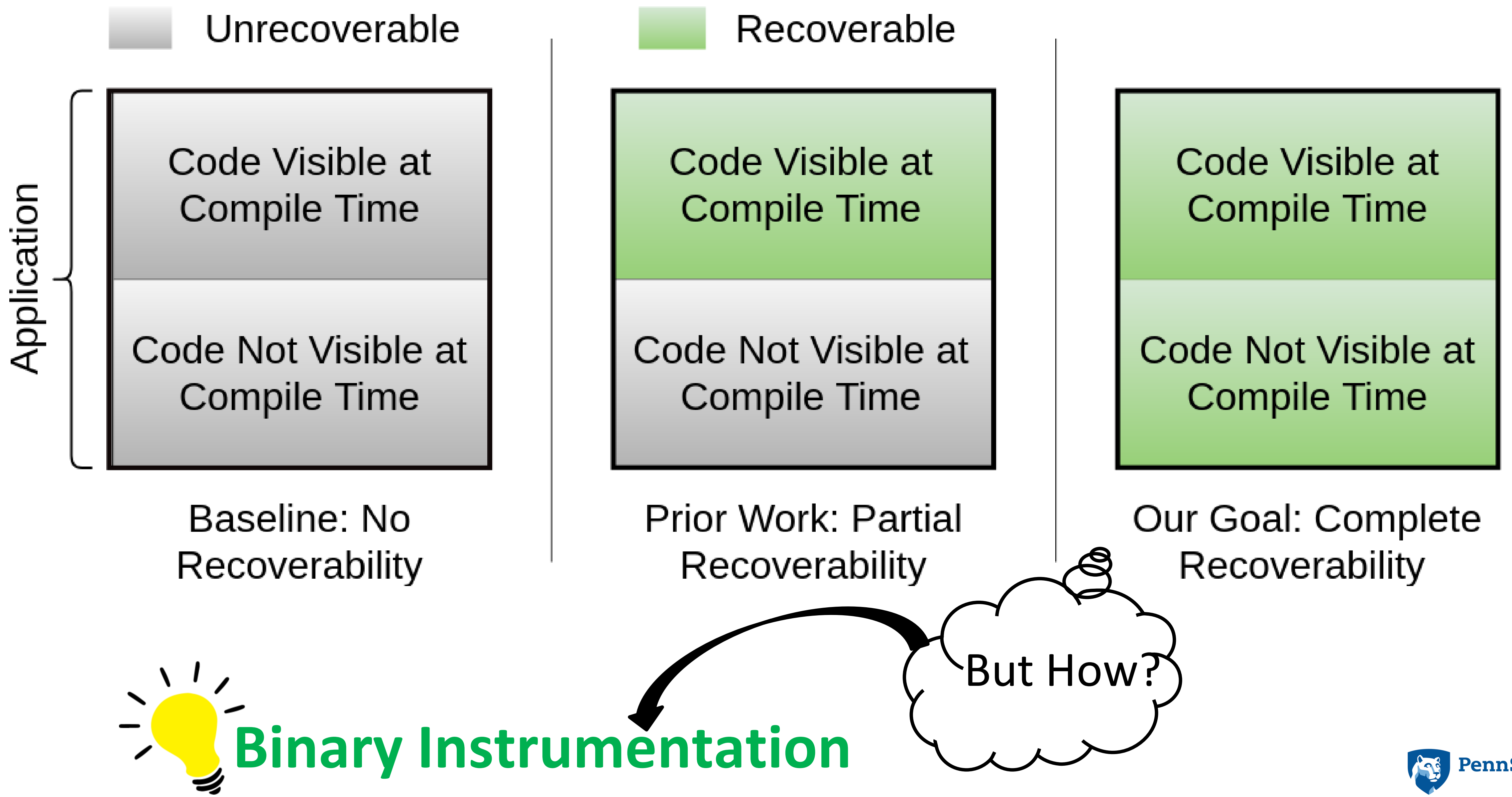
Idea #1: Persistency for the Whole Program



Idea #1: Persistency for the Whole Program



Idea #1: Persistency for the Whole Program



Idea #2: More Optimization in Logging

Store A

Store B

Store C

Baseline: No Logging

U_1=createLog(&A, A)
flush(U_1)
FENCE
St A
flush(A)

U_2=createLog(&B, B)
flush(U_2)
FENCE
St B
flush(B)

U_3=createLog(&C, C)
flush(U_3)
FENCE
St C
flush(C)

Prior Work: Serial Logging

U_1=createLog(&A, A)
U_2=createLog(&B, B)
U_3=createLog(&C, C)
flush(U_1)
flush(U_2)
flush(U_3)
FENCE
St A
flush(A)
St B
flush(B)
St C
flush(C)

Our Goal: Batch Logging

Idea #2: More Optimization in Logging

Baseline: No Logging

Store A

Store B

Store C

Prior Work: Serial Logging

U_1=createLog(&A, A)
flush(U_1)
FENCE
St A
flush(A)

U_2=createLog(&B, B)
flush(U_2)
FENCE
St B
flush(B)

U_3=createLog(&C, C)
flush(U_3)
FENCE
St C
flush(C)

Our Goal: Batch Logging

U_1=createLog(&A, A)
U_2=createLog(&B, B)
U_3=createLog(&C, C)
flush(U_1)
flush(U_2)
flush(U_3)
FENCE
St A
flush(A)
St B
flush(B)
St C
flush(C)

Can Hardware Help Us?

PennState

Evaluation Methodology

Tools:

- LLVM [Lattner' 04]
- DynamoRIO [Bruening' 12]

Benchmarks:

- WHISPER [Nalli' 17]
- Real-word Persistence Applications

Correct and Fast Persistency Guarantees

Sara Mahdizadeh Shahri
Aasheesh Kolli

PennState

YArch Workshop - February 2019

Backup Slides

Prior work

Atlas

Ensure Atomicity Within

Outermost Critical Section

Coupled SFR

Ensure atomicity within

Synchronization Free Regions