NVRAMOS2009

# Real-Time Executions of Program Codes in NAND Flash Memory

2009. 10. 20

Seoul National University
Chang-Gun Lee

# Increasing Market of Flash Memory

**Mobile embedded devices → shock resistance**
**Data and Codes are dramatically increasing → high volume**
**Flash Memory: Non-volatile, shock resistance, high volume**

해상도
DIGITAL VIDEO CAMERA DV330
RexPrime

Traditional Embedded Systems

Soft Real-time Embedded Systems

Hard Real-time Embedded Systems

NAND: Data Storage
NOR: Code Storage

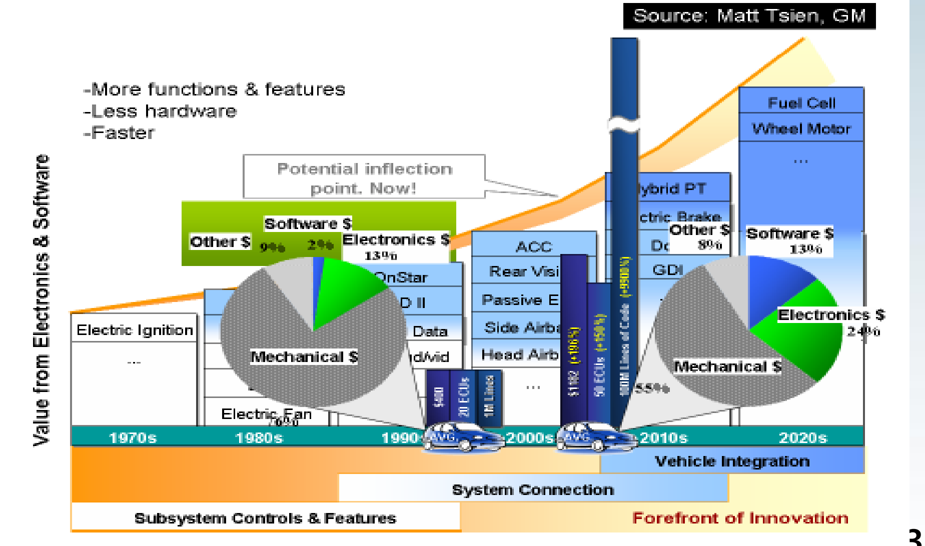
# Why NAND is attractive for codes?

## Increasing SW Complexity → Huge program codes

### Soft Real-Time Embedded Systems (e.g., Multimedia Smart Phones)

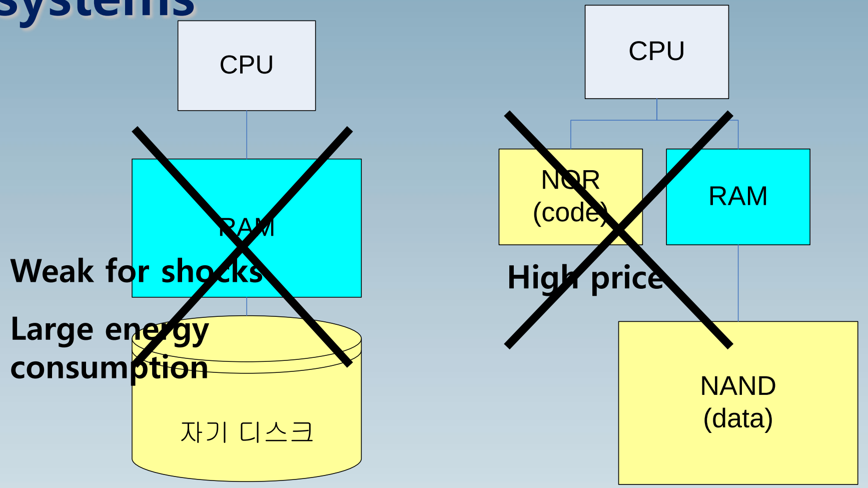
- more than 5M source code lines in a smart phone
- cf. 5M source code lines in banking system

### Hard Real-Time Embedded Systems (e.g., Automotive)



Source: Matt Tsien, GM

### Memory-hierarchy for embedded systems



자기디스크 기반
메모리 계층구조

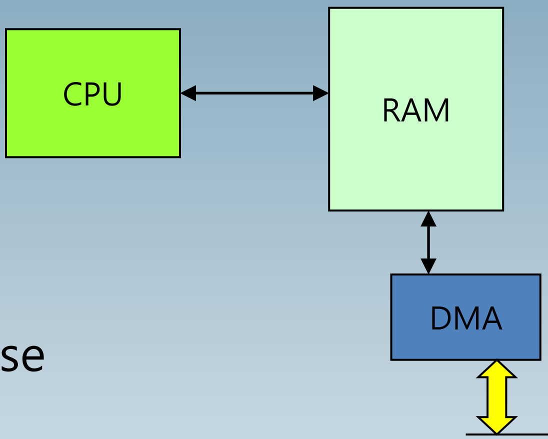
NOR와 NAND를 모두
사용하는 메모리 계층구조

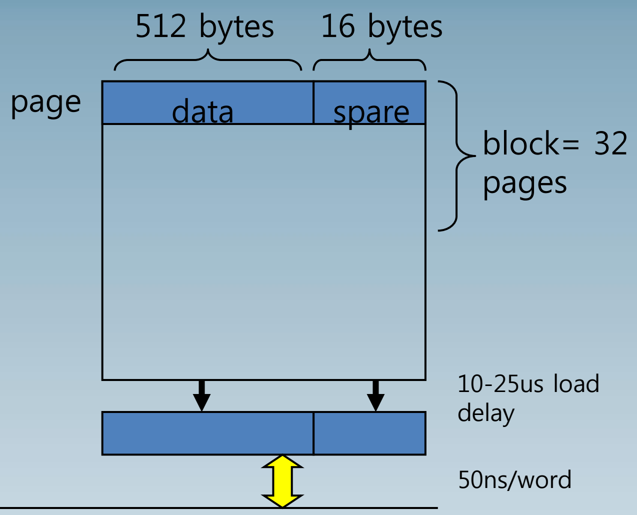
***NAND-based low-cost, large-size Code execution technology***

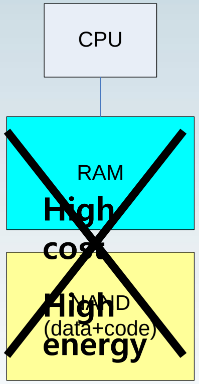
# What is a big challenge of NAND for codes?

## NAND Physical Characteristics

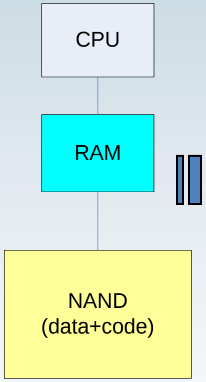
- Page based sequential Read (No Random Access)
  - Read: 130us/page
- Page based write
  - 300us/page
- Block based Erase
  - 2 ms/block
- No overwrite before erase







NAND 기반 Shadowing



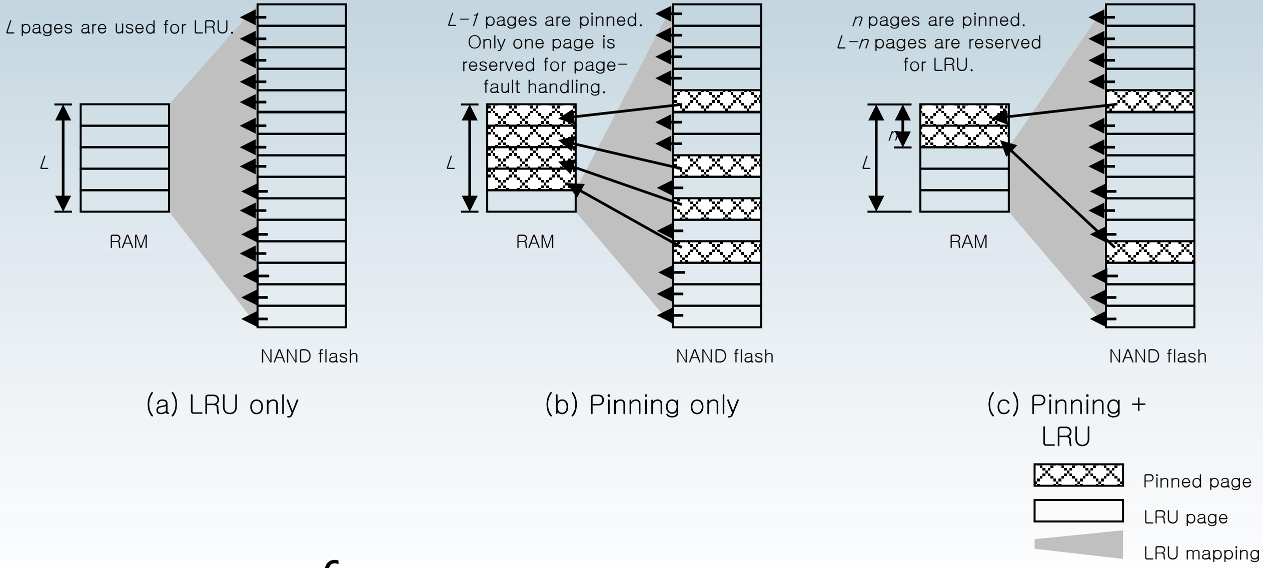
NAND 기반 가상메모리

**How to guarantee Program's real-time execution with smallest RAM?**

- RT-PLRU
  - Soft real-time
  - Single task
- mRT-PLRU
  - Extension to multiple tasks
- HRT-PLRU
  - Extension to hard real-time

# RT-PLRU: Soft real-time single task

- Two Important Goals
  - Developer-transparency
  - Probabilistic guarantee of real-time with minimum DRAM
- Solution approach
  - Kernel-level auto-discovery of apps. temporal intension
  - Kernel-level auto-tracing of page reference sequences
  - Kernel-level auto-configuration (optimal) of pinning and LRU (**RT-PLRU**)



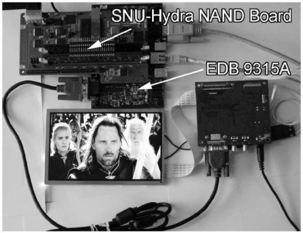
L pages are used for LRU.
L
RAM
NAND flash
L−1 pages are pinned. Only one page is reserved for page−fault handling.
L
RAM
NAND flash
n pages are pinned. L−n pages are reserved for LRU.
n
L
RAM
NAND flash
Pinned page
LRU page
LRU mapping

(a) LRU only

(b) Pinning only

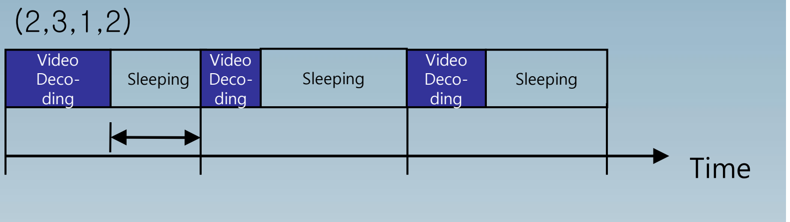
(c) Pinning + LRU

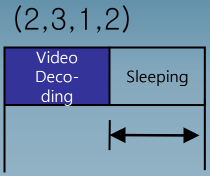
# Overall Design Flow (RT-PLRU)



prototype with sample movie

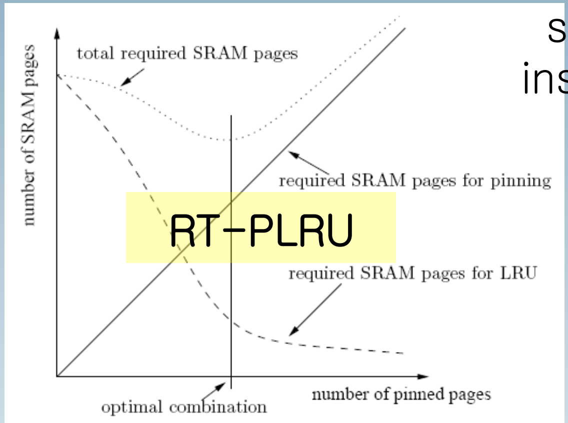
kernel−level auto−tracing



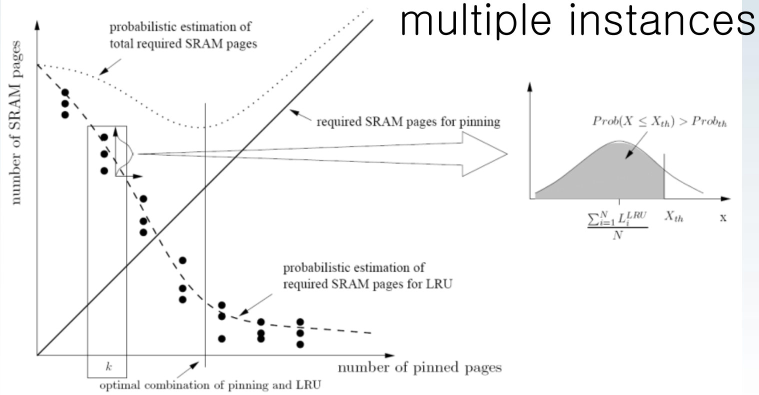


kernel−level auto configuration

single instance

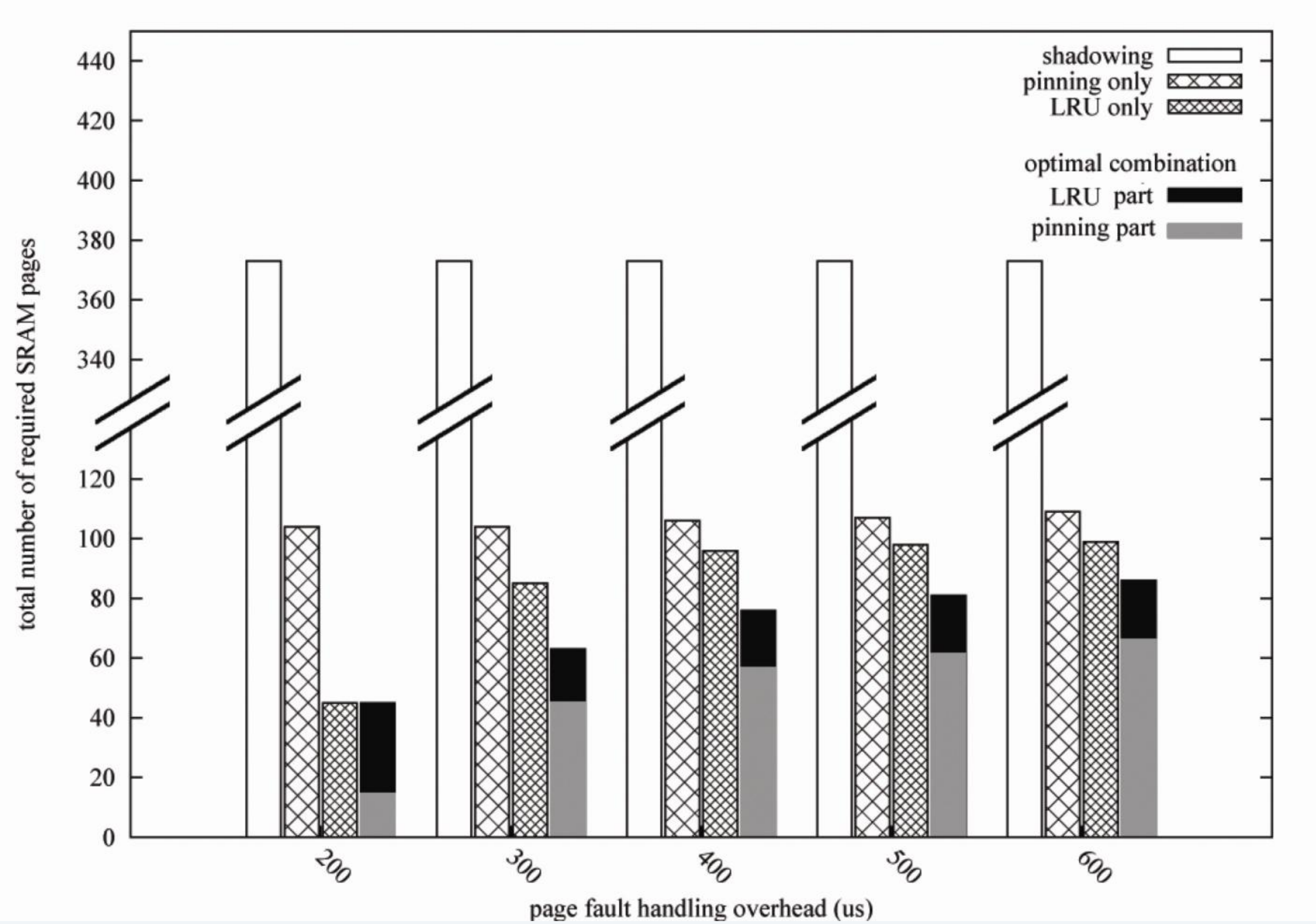


probabilistic extension for multiple instances

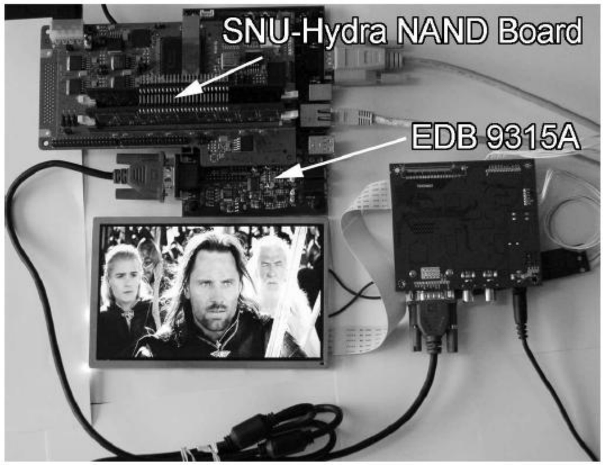


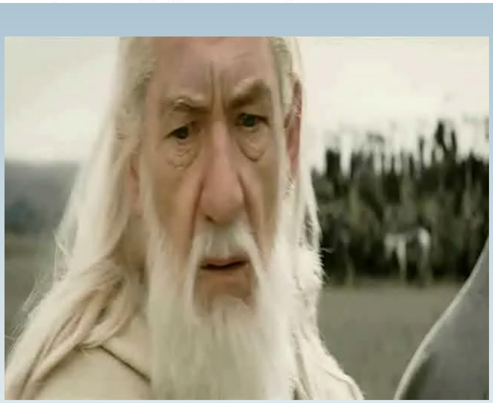
production with RT−PLRU

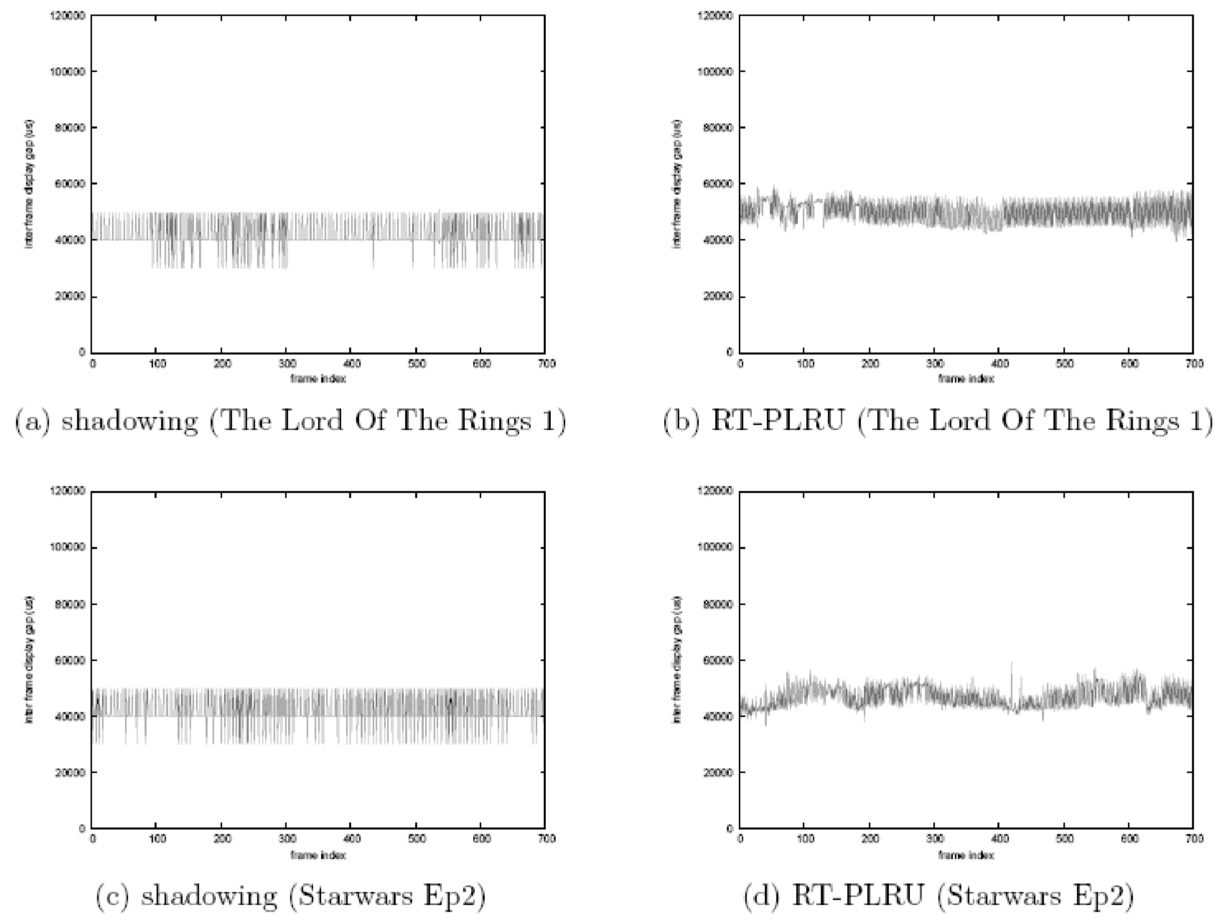
# Comparison of required DRAM sizes



# Implementations







(a) shadowing (The Lord Of The Rings 1)

(b) RT-PLRU (The Lord Of The Rings 1)

(c) shadowing (Starwars Ep2)

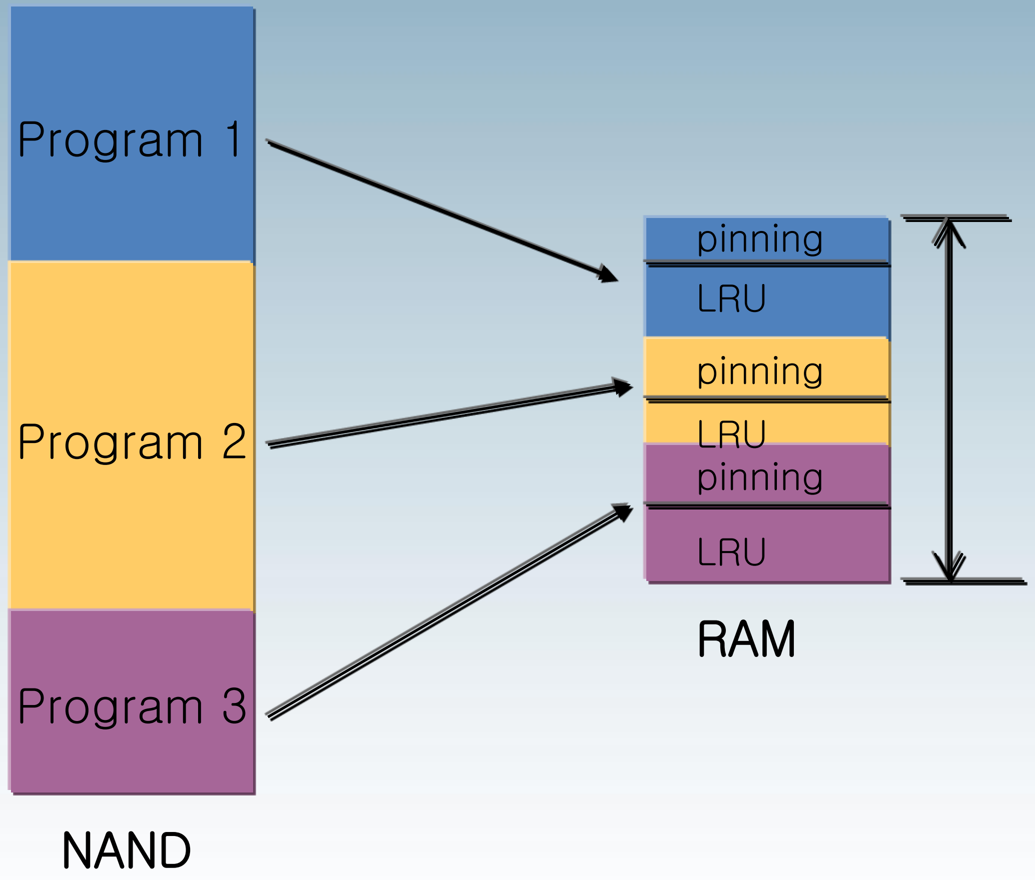
(d) RT-PLRU (Starwars Ep2)

# Demo

- RT-PLRU
  - Soft real-time
  - Single task
- mRT-PLRU
  - Extension to multiple tasks
- HRT-PLRU
  - Extension to hard real-time

# mRT-PLRU: Soft real-time Multiple tasks

- Problems to answer

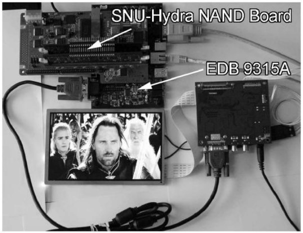


Minimize this size

Such that

All three tasks "probabilistically" guarantee their deadlines

# Step 1: Per-task analysis

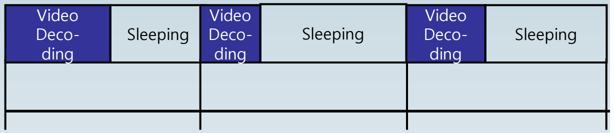


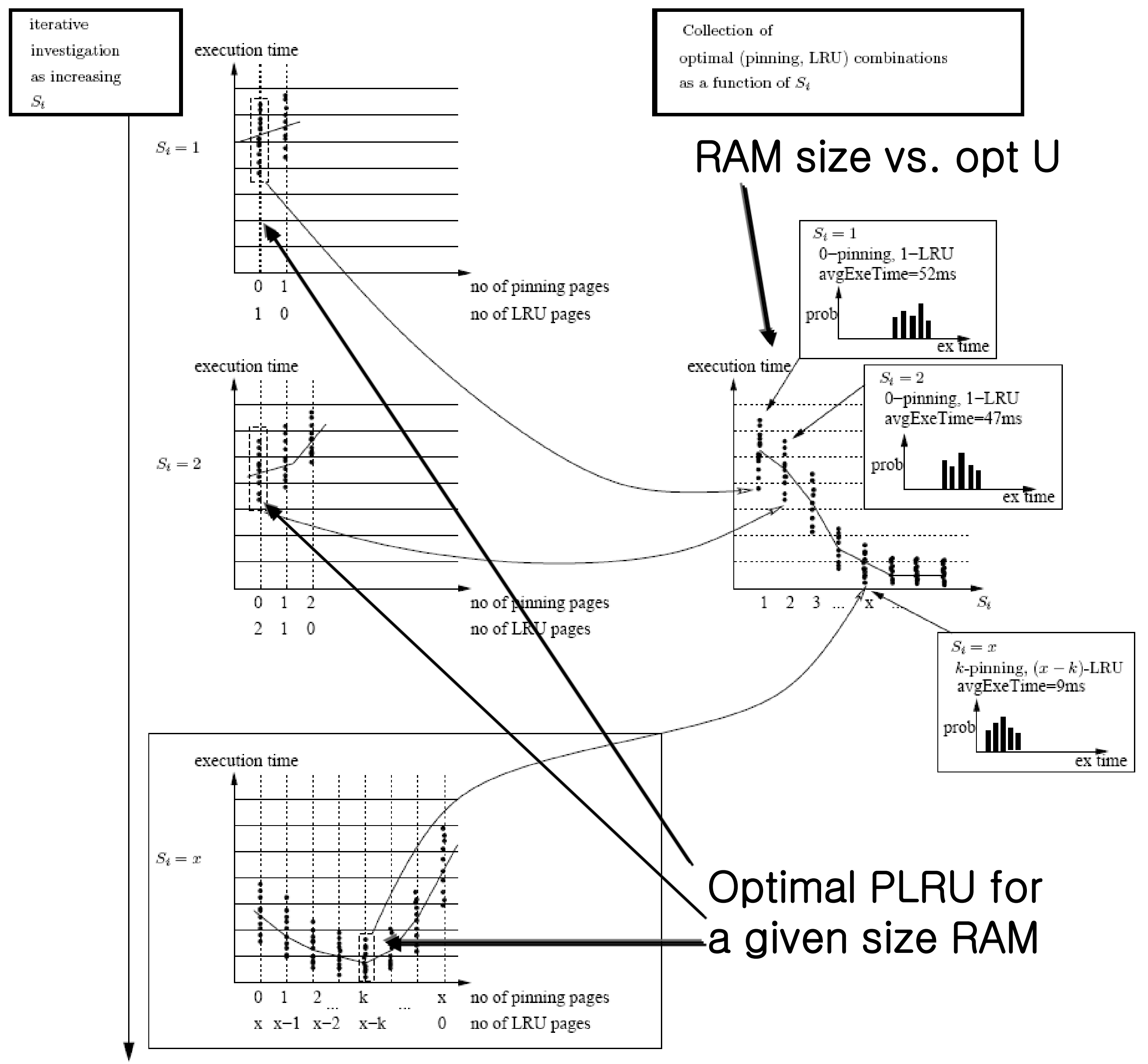
prototype with sample content

kernel−level auto−tracing

⇨

(2,3,1,2)

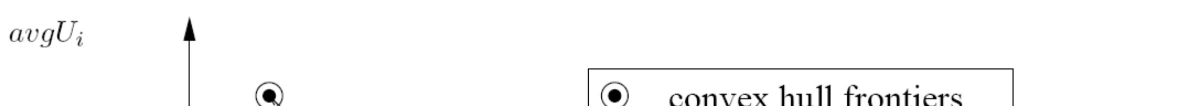


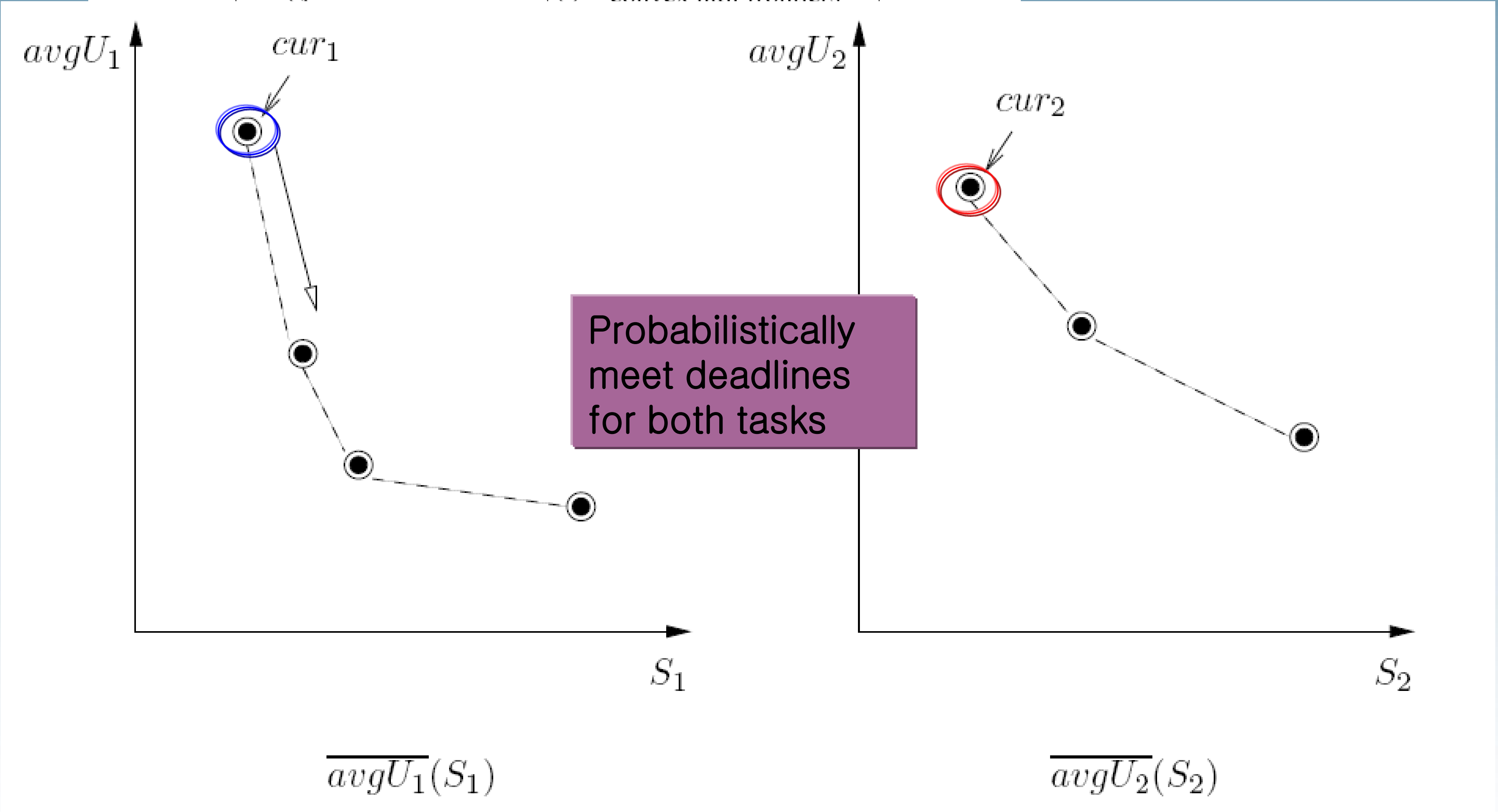


RAM size vs. opt U

Optimal PLRU for a given size RAM

# Step 2: Convex optimization



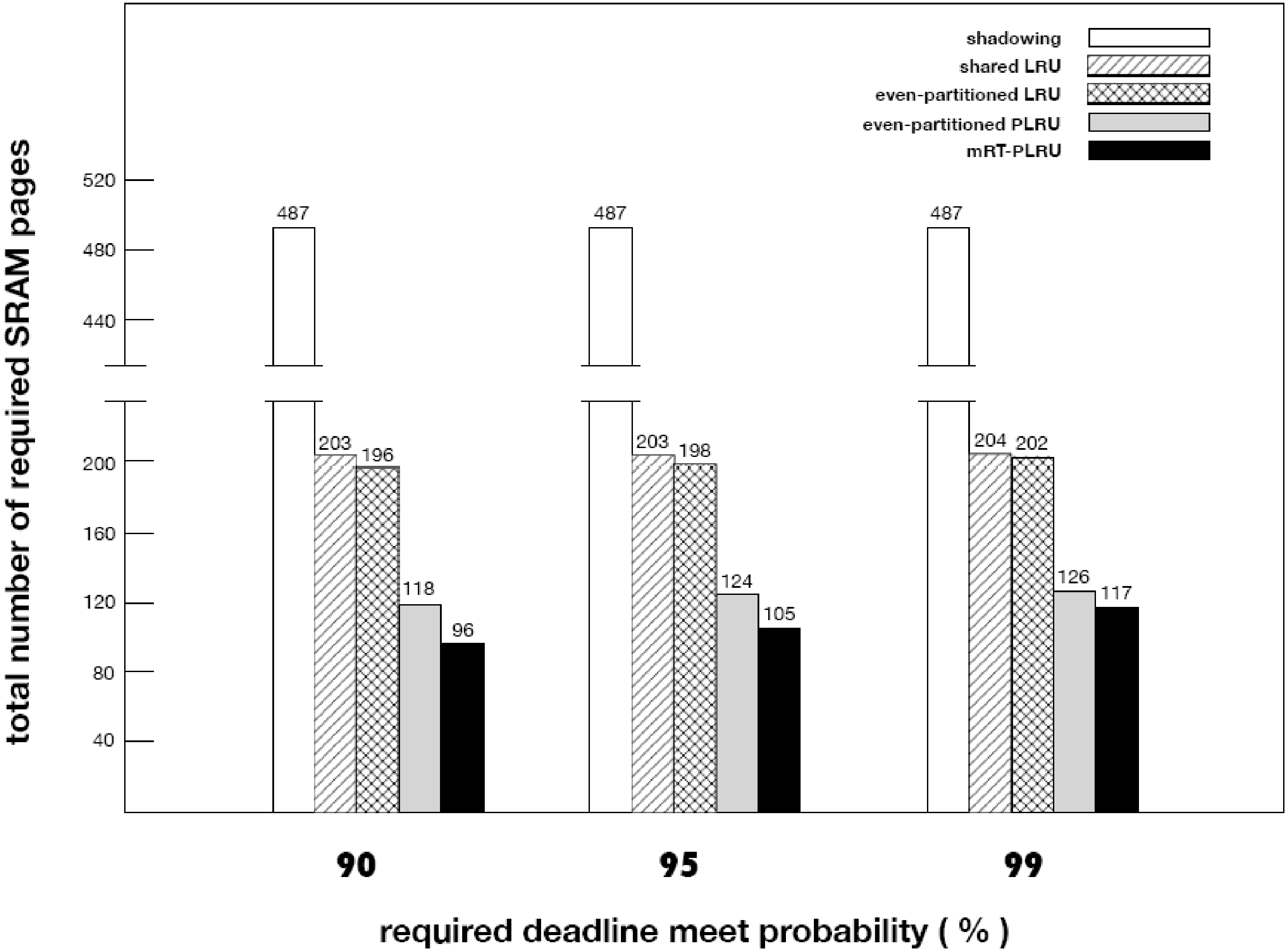


$\overline{avgU_1}(S_1)$

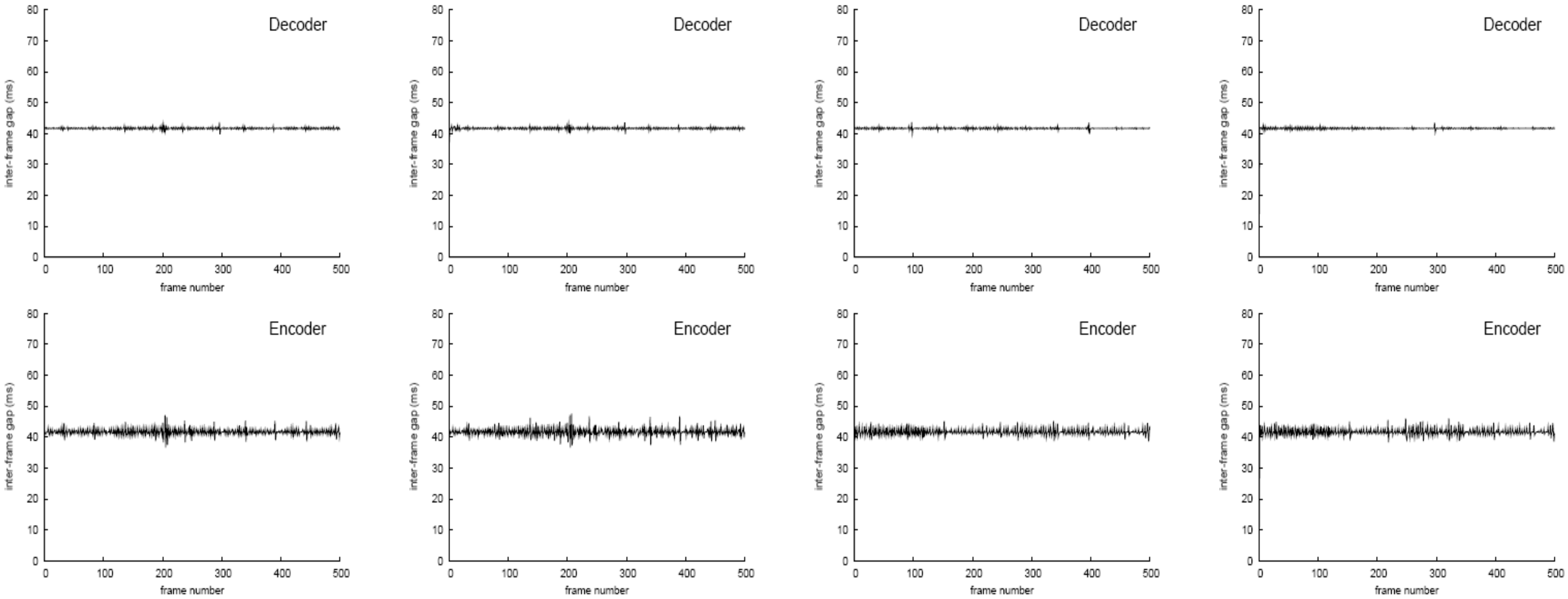
$\overline{avgU_2}(S_2)$

Probabilistically meet deadlines for both tasks

# How much RAM saved?



# Really work?



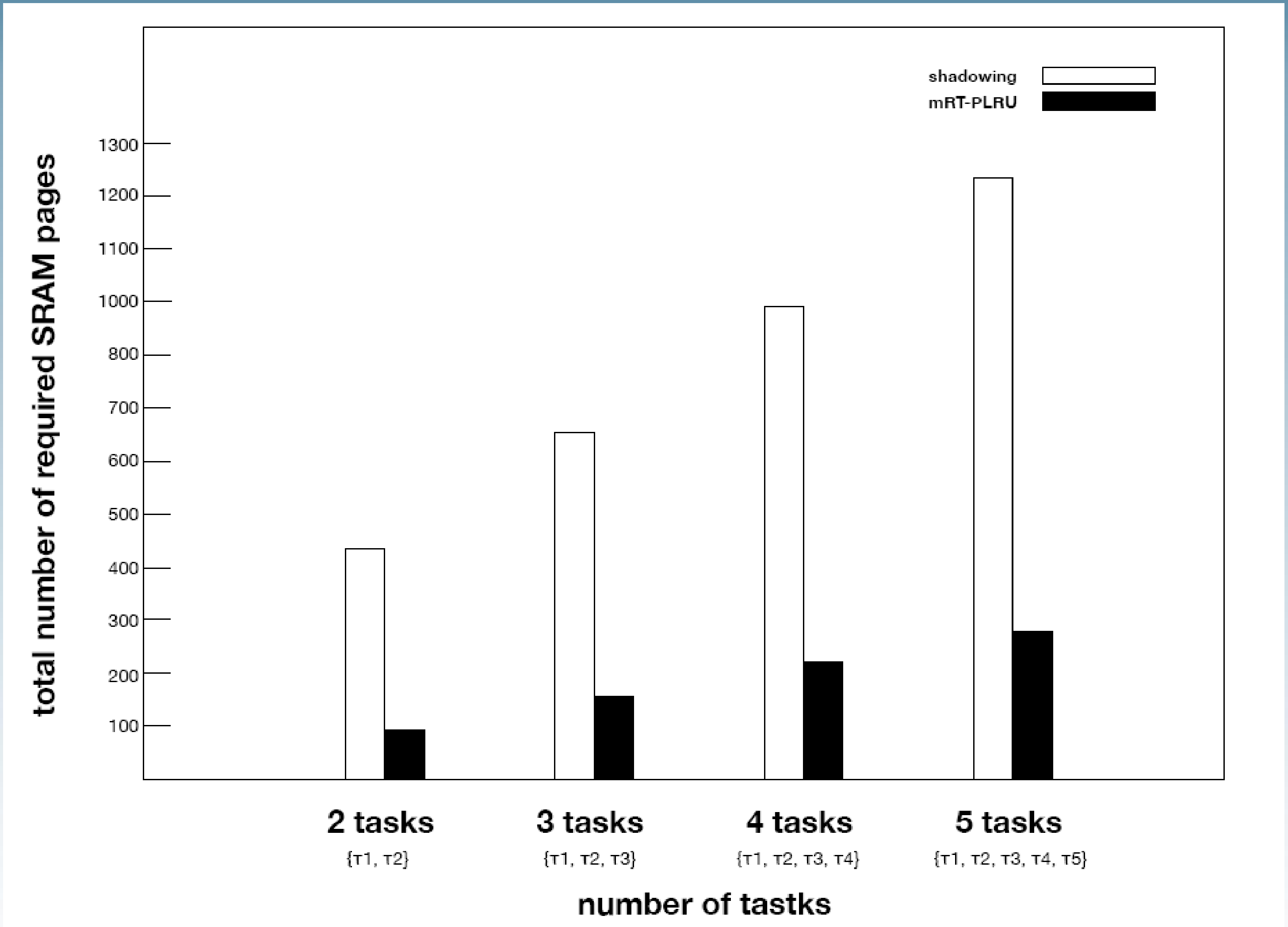
(a) shadowing ("Content A")

(b) mRT-PLRU ("Content A")

(c) shadowing ("Content B")

(d) mRT-PLRU ("Content B")

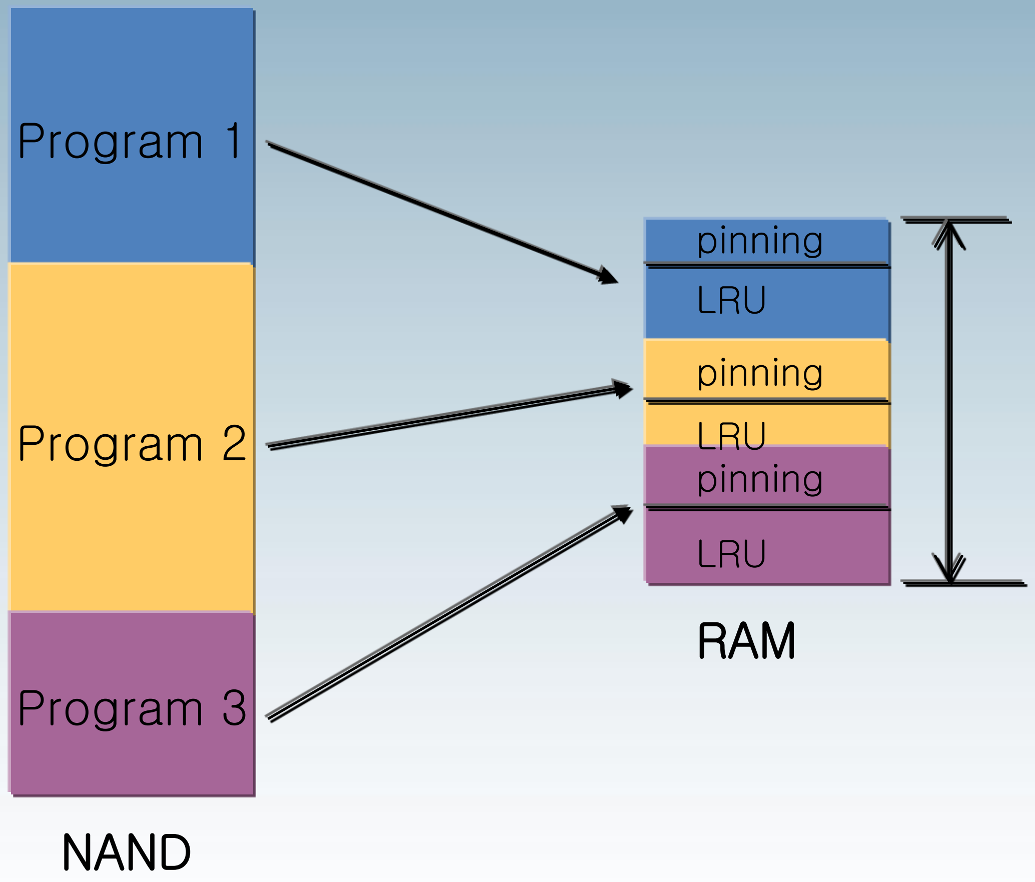
# More than two tasks?



- RT-PLRU
  - Soft real-time
  - Single task
- mRT-PLRU
  - Extension to multiple tasks
- HRT-PLRU
  - Extension to hard real-time

# HRT-PLRU: Hard real-time Multiple tasks

- Problems to answer

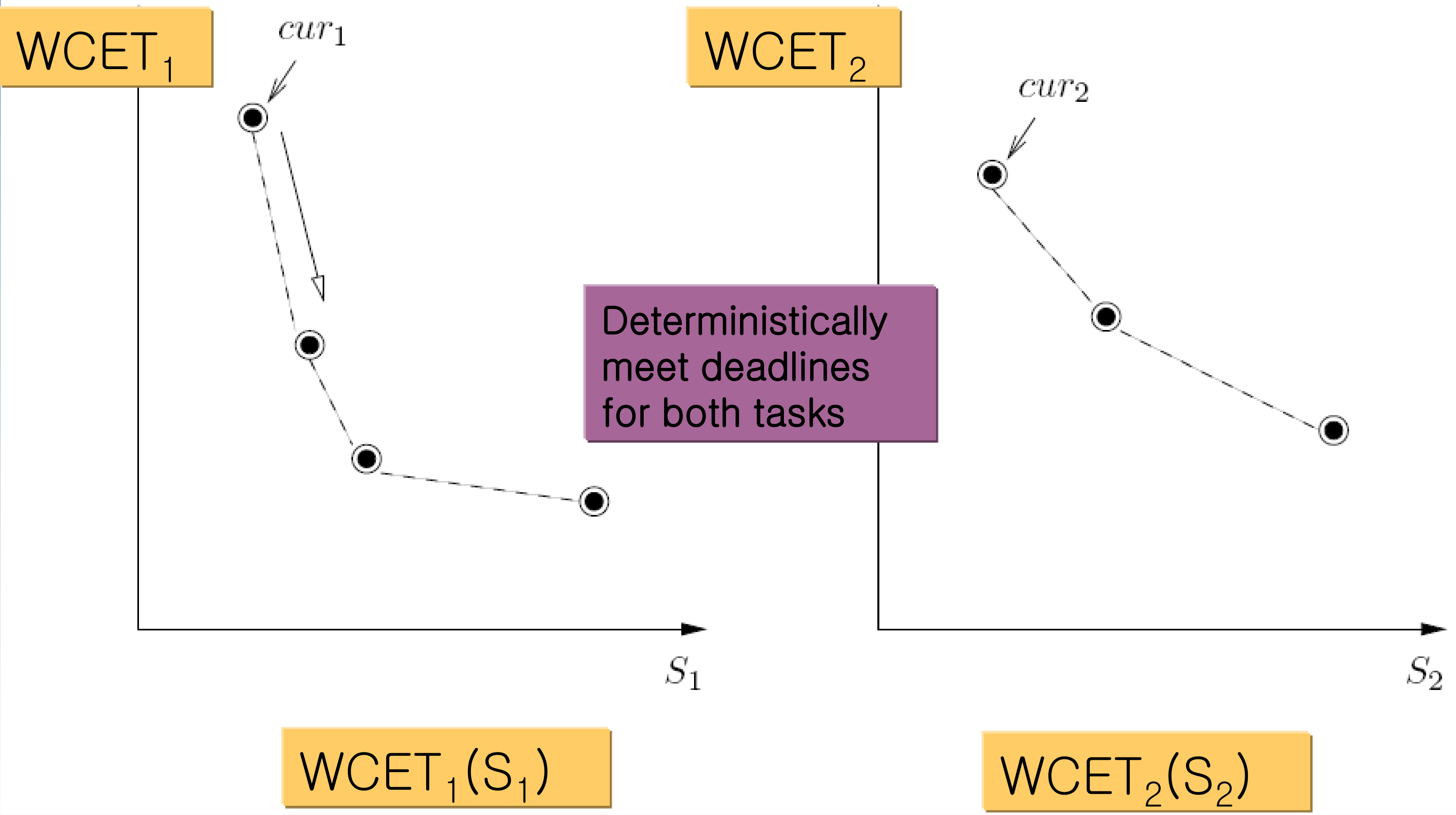


Minimize this size

Such that

All three tasks "deterministically" guarantee their deadlines

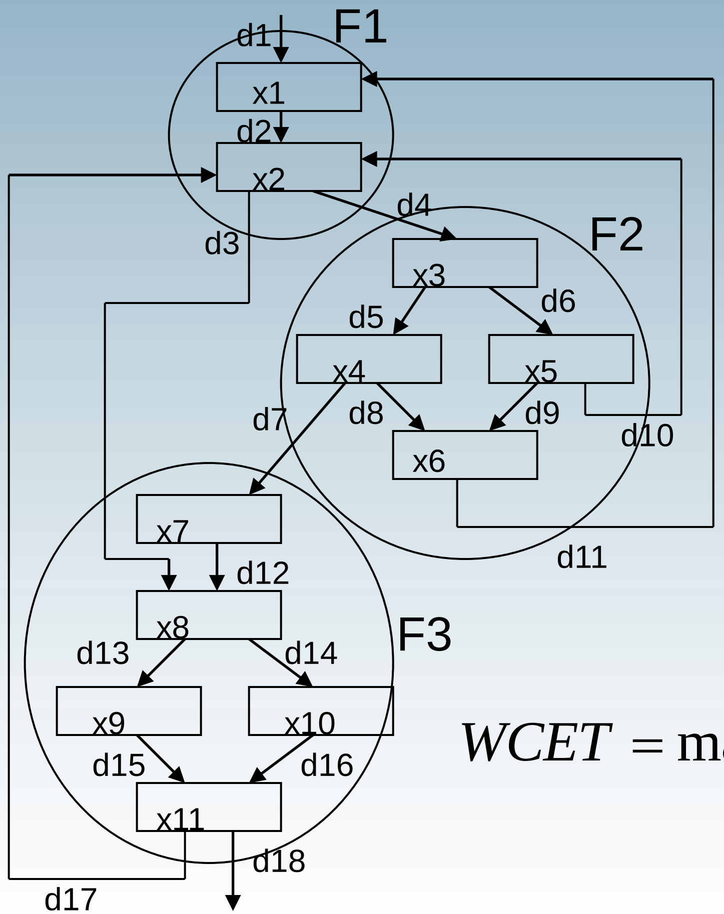
# Per-task analysis and Convex optimization



Deterministically meet deadlines for both tasks

# Step 1: Per-task analysis

- WCET for a PLRU combination

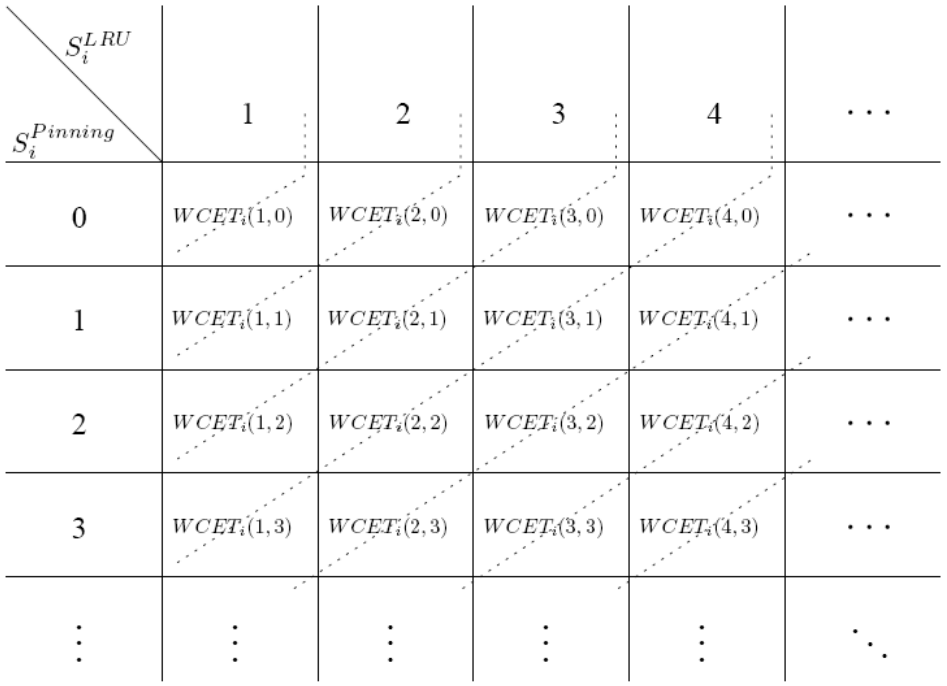


ILP can solve this!

$$WCET = \max\left(\left(\sum_{i=1}^{11} e_i x_i\right) + \left(\sum_{j \in PageTransition} d_j^{miss}\right) \cdot PageFaultDelay\right)$$

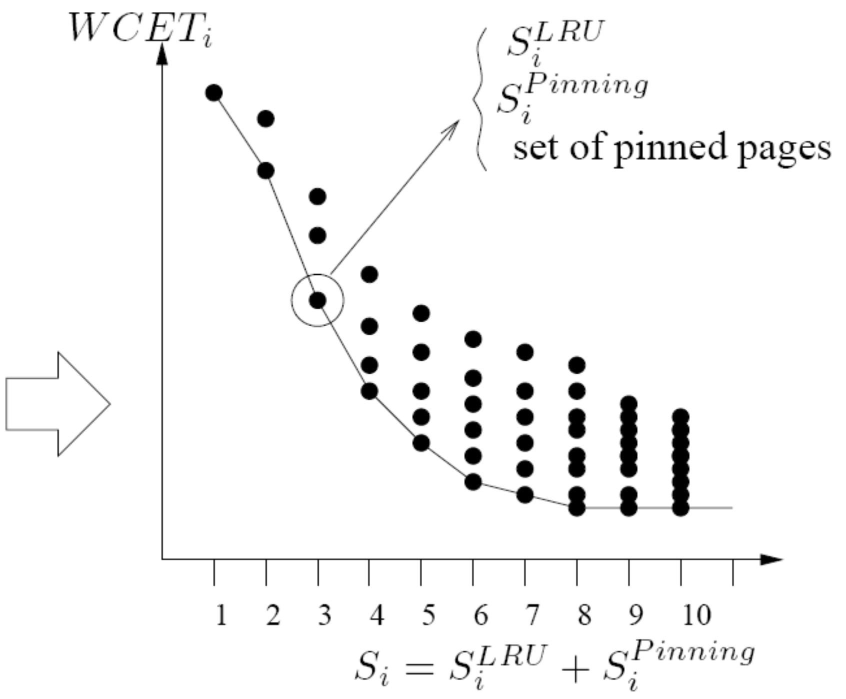
# Step 1: Per-task analysis

- RAM size vs. opt PLRU in terms of WCET



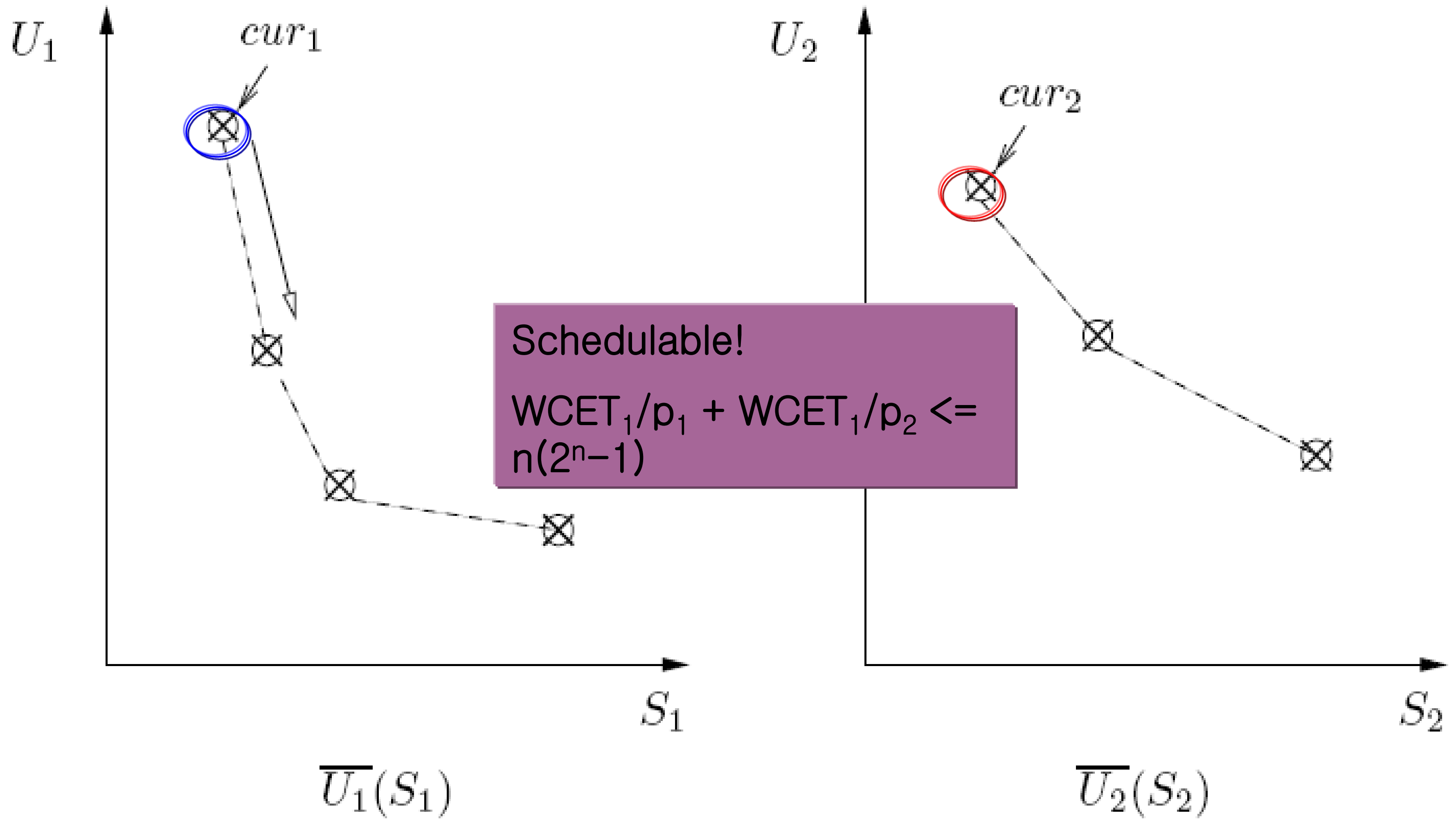
| $S_i^{LRU}$ / $S_i^{Pinning}$ | 1 | 2 | 3 | 4 | $\cdots$ |
|---|---|---|---|---|---|
| 0 | $WCET_i(1,0)$ | $WCET_i(2,0)$ | $WCET_i(3,0)$ | $WCET_i(4,0)$ | $\cdots$ |
| 1 | $WCET_i(1,1)$ | $WCET_i(2,1)$ | $WCET_i(3,1)$ | $WCET_i(4,1)$ | $\cdots$ |
| 2 | $WCET_i(1,2)$ | $WCET_i(2,2)$ | $WCET_i(3,2)$ | $WCET_i(4,2)$ | $\cdots$ |
| 3 | $WCET_i(1,3)$ | $WCET_i(2,3)$ | $WCET_i(3,3)$ | $WCET_i(4,3)$ | $\cdots$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ |

(a) $WCET_i(S_i^{LRU}, S_i^{Pinning})$ table



(b) RAM size $S_i$ vs. WCET relation

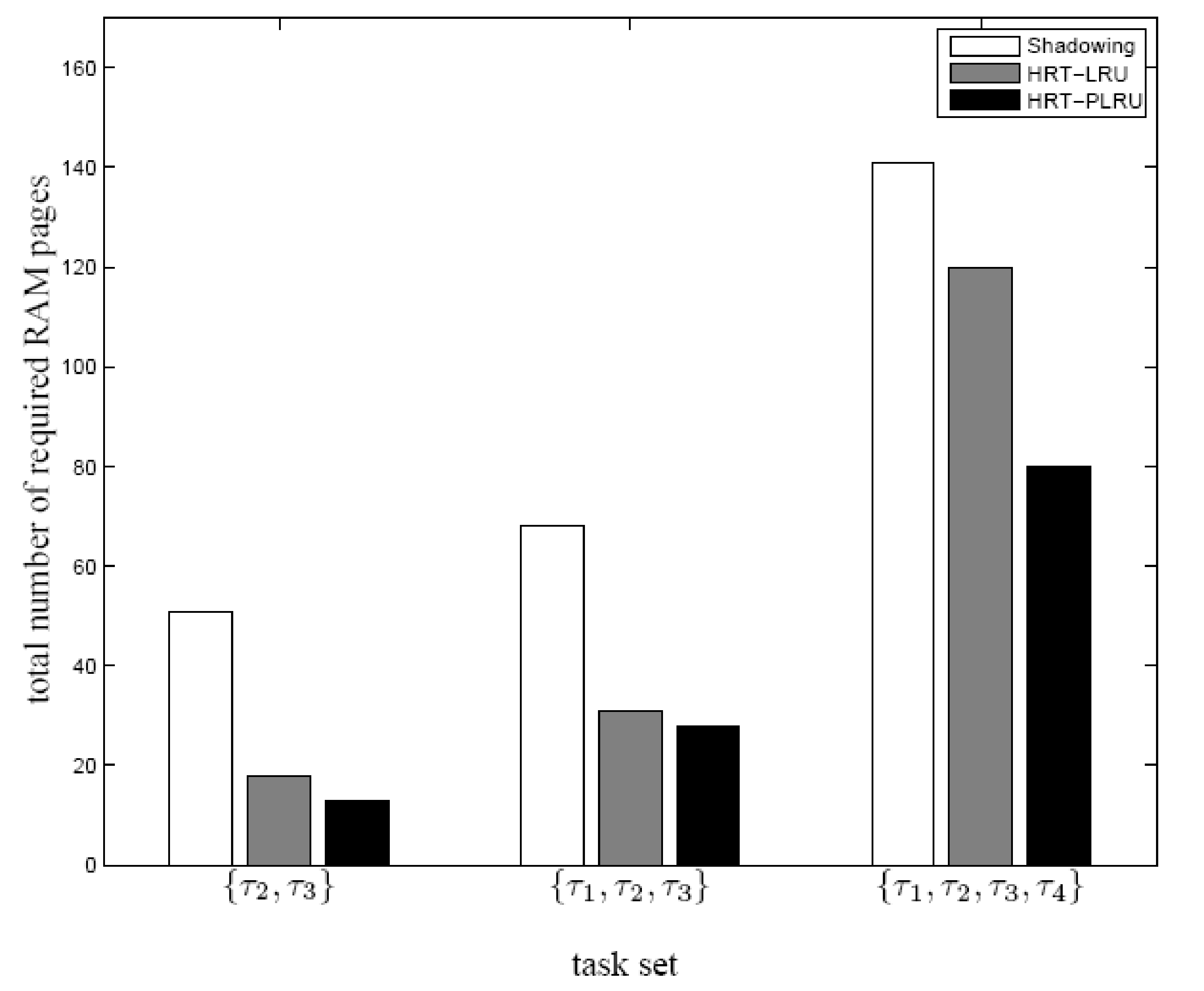
# Step 2: Convex optimization



Schedulable!

$WCET_1/p_1 + WCET_1/p_2 <= n(2^n-1)$

# How much RAM saved?



# Conclusion

- RT-PLRU for
  - Soft real-time single task → RT-PLRU
  - Soft real-time multiple tasks → mRT-PLRU
  - Hard real-time multiple tasks → HRT-PLRU
- It provides a potential to use NAND for code executions of real-time applications
- More study needed for practical applications
  - Trade-off between RAM cost and energy consumption
  - System bus conflict problems
  - etc.