Towards Scalable and Secure Execution Platform for Embedded Systems

Junji Sakai, INOUE Hiroaki, Masato Edahiro
System Devices Research Laboratories
NEC Corporation

Jan 24, 2007

Agenda

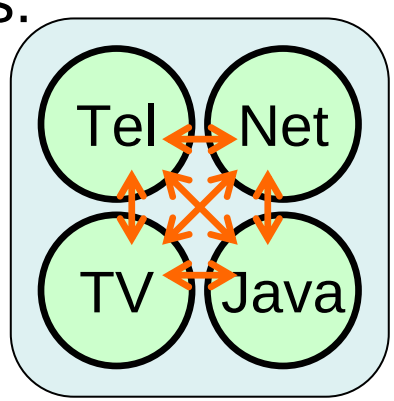
- Background
  - Problems in high-end embedded systems
  - Partitioning using multicore
- Physical partitioning
  - for Performance assurance
  - for Download security
- More flexibility and scalability
  - Virtualization
  - SMP
- Summary and future work

Arising Problems

Needs: “PC-level” functions in embedded systems

Problems:

Multi-Functional

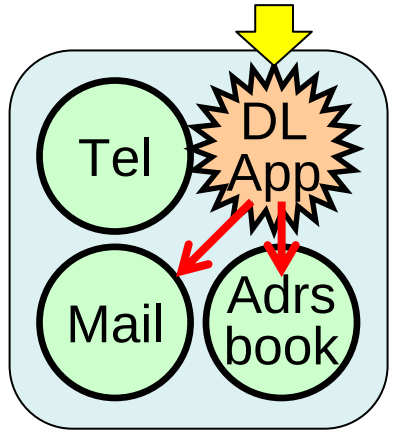


Java disturbs TV stream

(CPU resource consumed)

lack of **performance assurance**

Application Download



DL App attacks Mail App

decline in **robustness**

Reliability degradation

Existing approaches:

Priority Control

Sandbox

insufficient

Improve Reliability with Partitioning

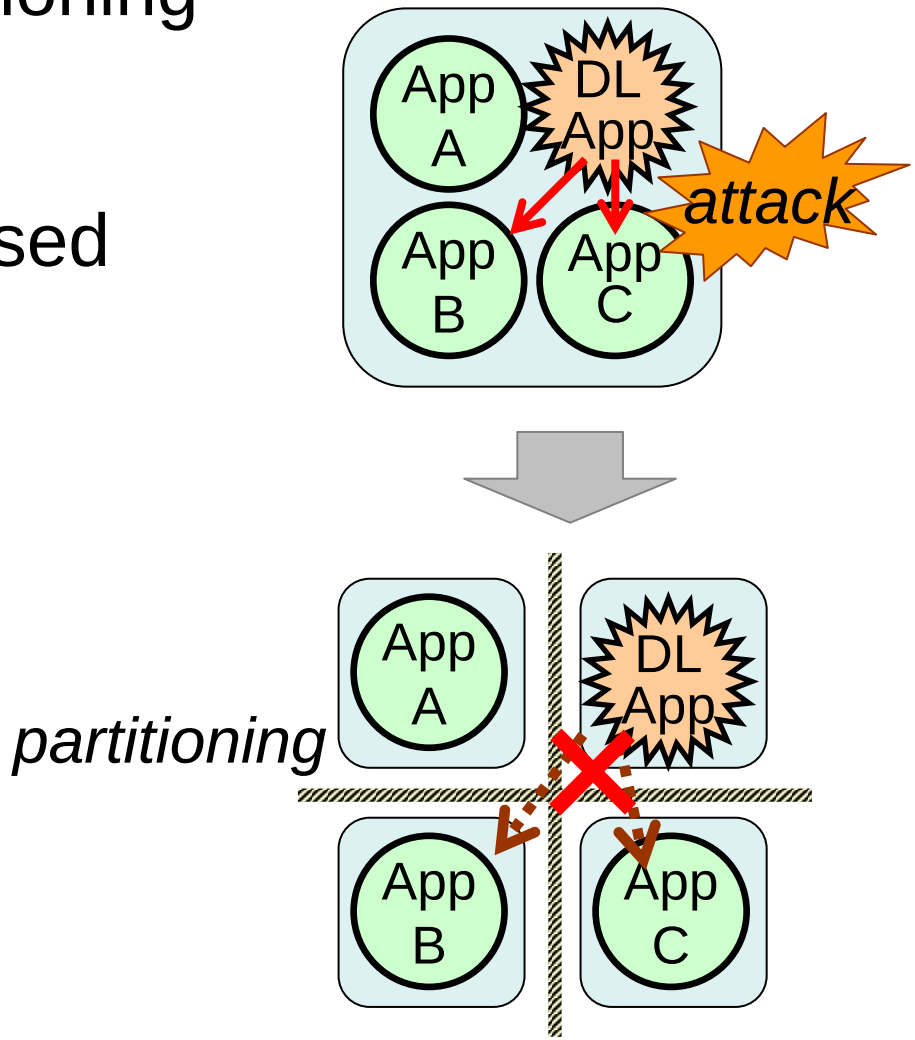
- Separate Apps by Partitioning

- Interference
- Attacks

} suppressed

- Reliability

- Accompanying issue
  = Communication
  →discussed later



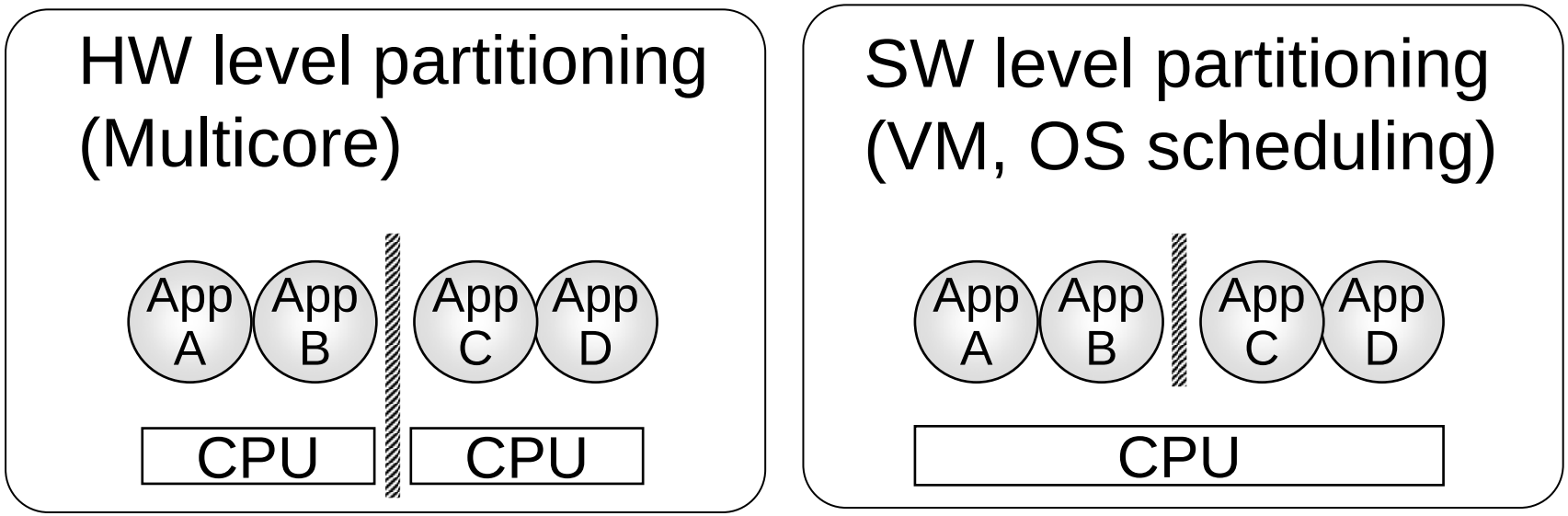
CPU Trend – Multicore anywhere

- In Servers and Desktops
  - Use multicore to reduce heat emission
- Also in Embedded Systems
  - Use multicore to reduce power consumption

and moreover,

- to resolve the arising problems ← our proposal

Two Approaches to Partition



Robust
High Performance

Tradeoff

Flexible
Extendable

Our approach

degree of reliability

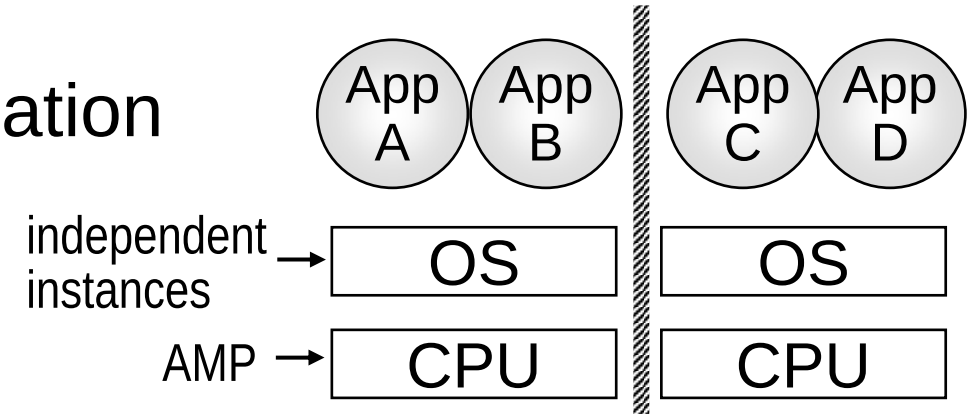
Existing approaches

Physical Partitioning

Our basic strategy:

- CPU level
  and
- OS kernel level

} separation

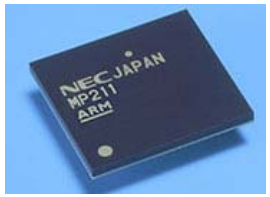


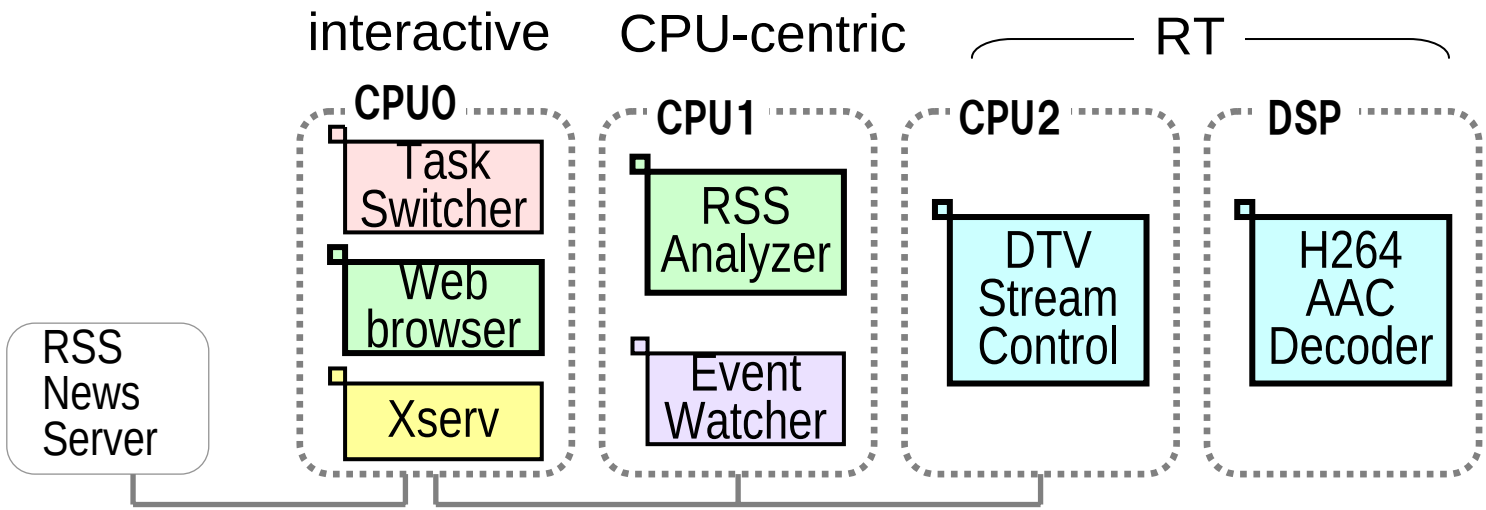
- Why OS kernel level ?
  - Avoid too much dependence on OS reliability
    (OS may be vulnerable)
  - Fast recovery (reboot only the crashed part)
  - Simple and robust using commodity CPU and OS

⇨ 1) performance assurance
2) system robustness

Performance Assurance by Multicore

- App set: DTV + Newsreader + gadgets
- Assign tasks to CPU cores so that mutual interference is minimized
  - RT tasks / CPU-centric tasks / Interactive tasks
  - MP211 (3x ARM9 + 1x DSP)

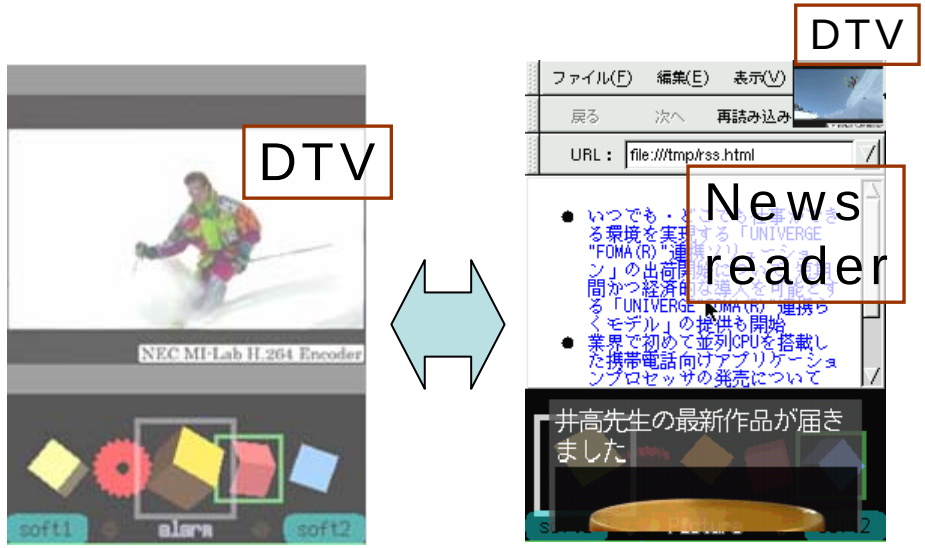




→ Browser, Java, Xserver, RT stream control

Performance Assurance -- Results

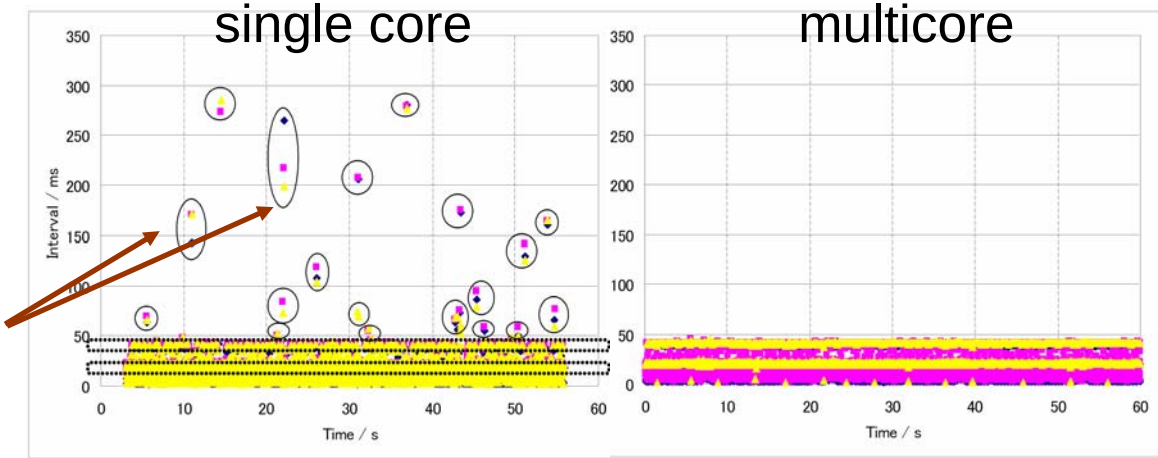
- All applications smoothly run on multicore.
  ←→ Single core:
  short interrupts
  of sound stream



Jitter of a periodical task interval (DTV stream control)

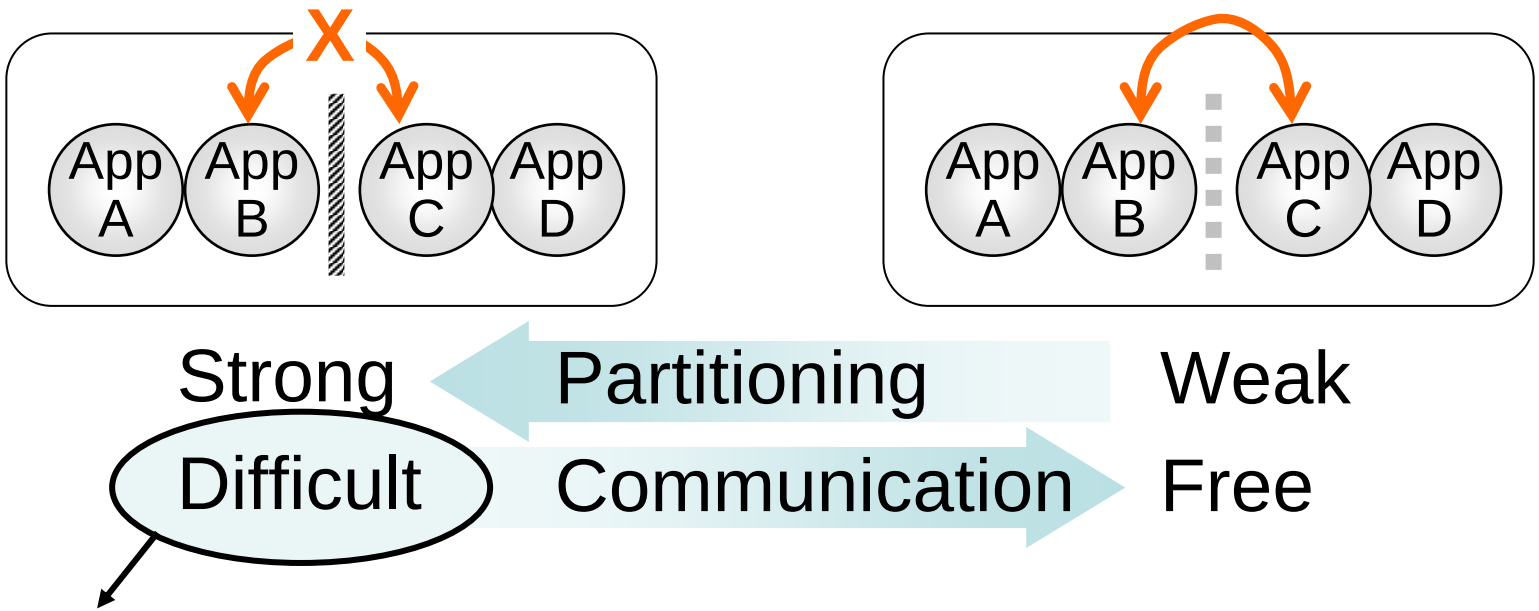
No jitter in multicore
→stable execution

delay occurrences



Communication and Partitioning

Comm. and partitioning are tradeoff

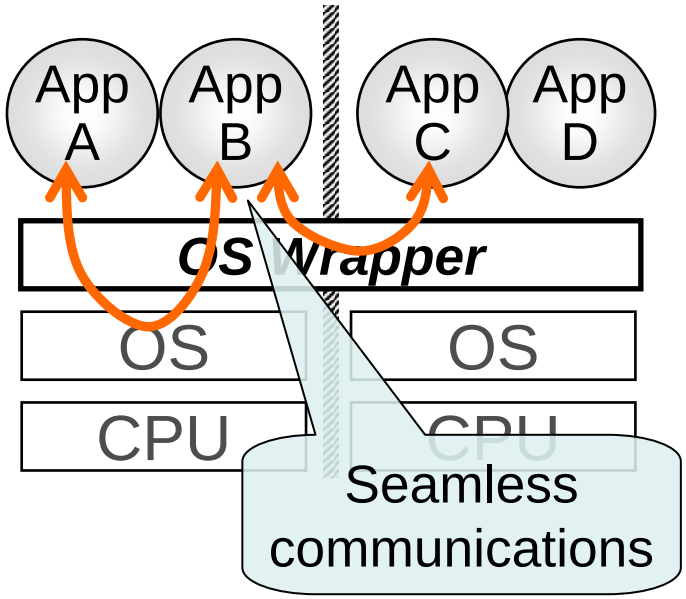


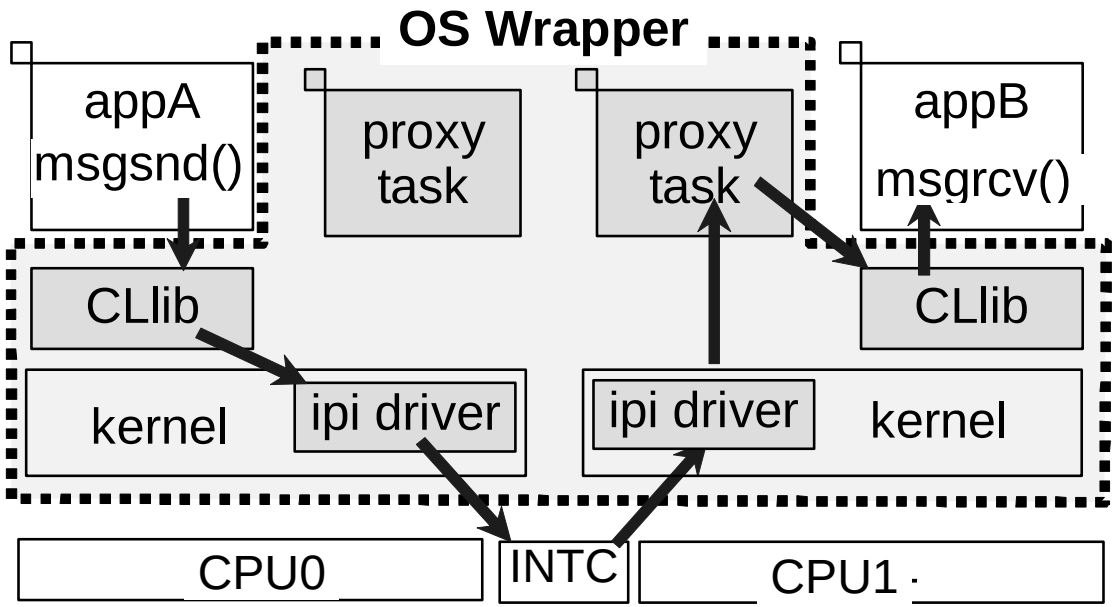
In our case:

- OS standard APIs cannot talk to other core.
- Rewriting applications should be avoided.
  (No special communication APIs welcomed)

OS Wrapper

- Provides seamless APIs
  for inter-core and intra-core communications
  - *No* source code modifications in the Demo set
- Hooks OS service calls and dispatches to destination
- Mostly user land implementation (→can be applied to other OSs)

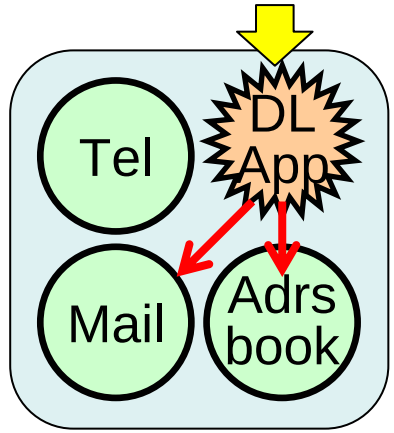




Downloading and Security

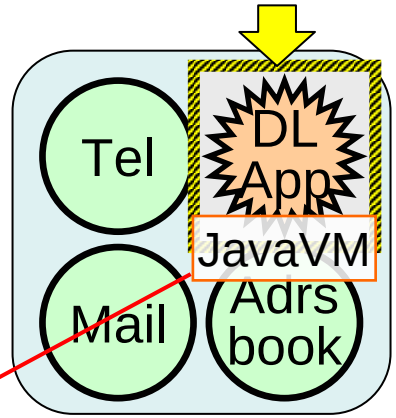
- Downloading becomes popular in embedded systems
  - Expand functions
- Security issue:
  - Unauthorized access
  - Malicious attacks

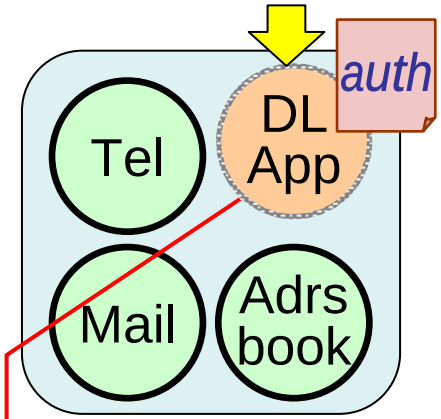
  } against core functions



- Existing approaches:
  - Sandbox (Java)
  - Authorization (BREW)

→ *apply multicore to security issue*





- Performance overhead

- Validation cost to ensure safety
- Recovery time

FIDES: Robustness by Multicore

* FIDES means “trust” in Latin

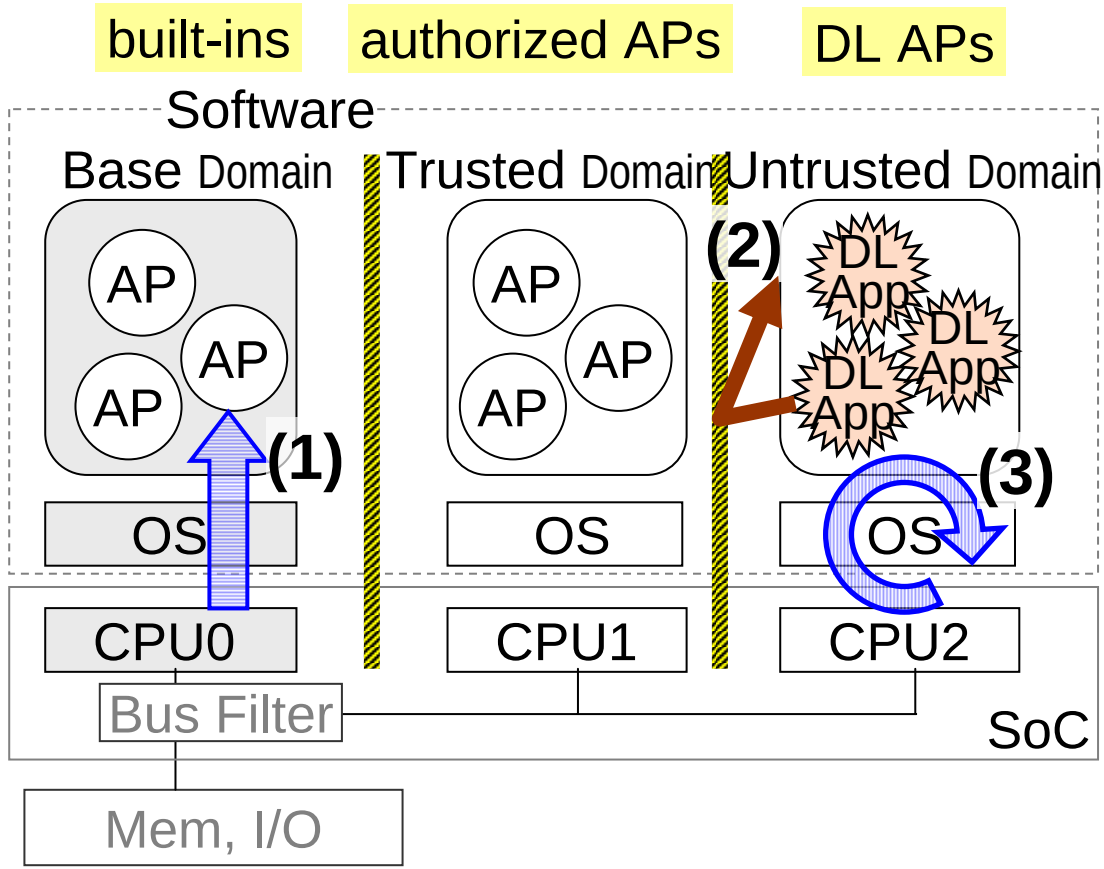
- Domains (CPU core + OS) corresponding to Trust levels

Benefit:

(1)No sandbox
→lower overhead

(2)Physical partitioning
→block attacks

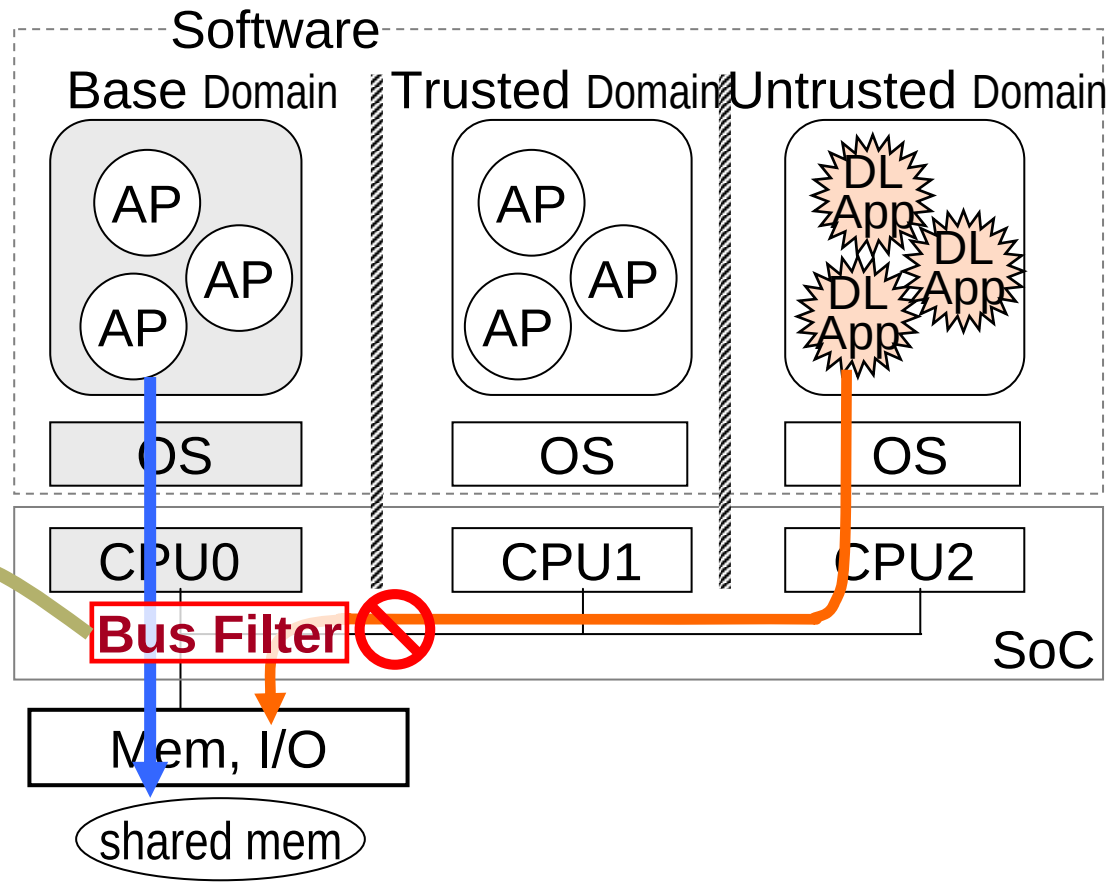
(3)Separate OSs
→quick recovery



Bus Filter

- Another security risk: caused through shared devices

- Bus Filter :
  a firewall on the bus
  → block illegal access
  from DL Apps

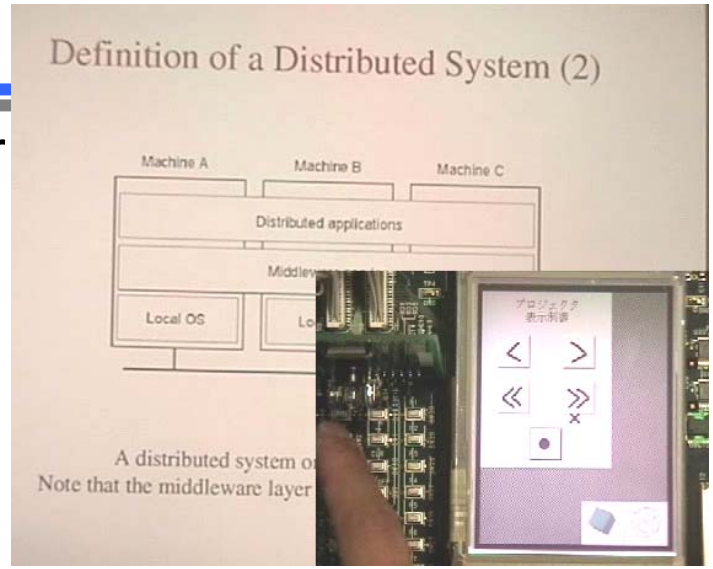


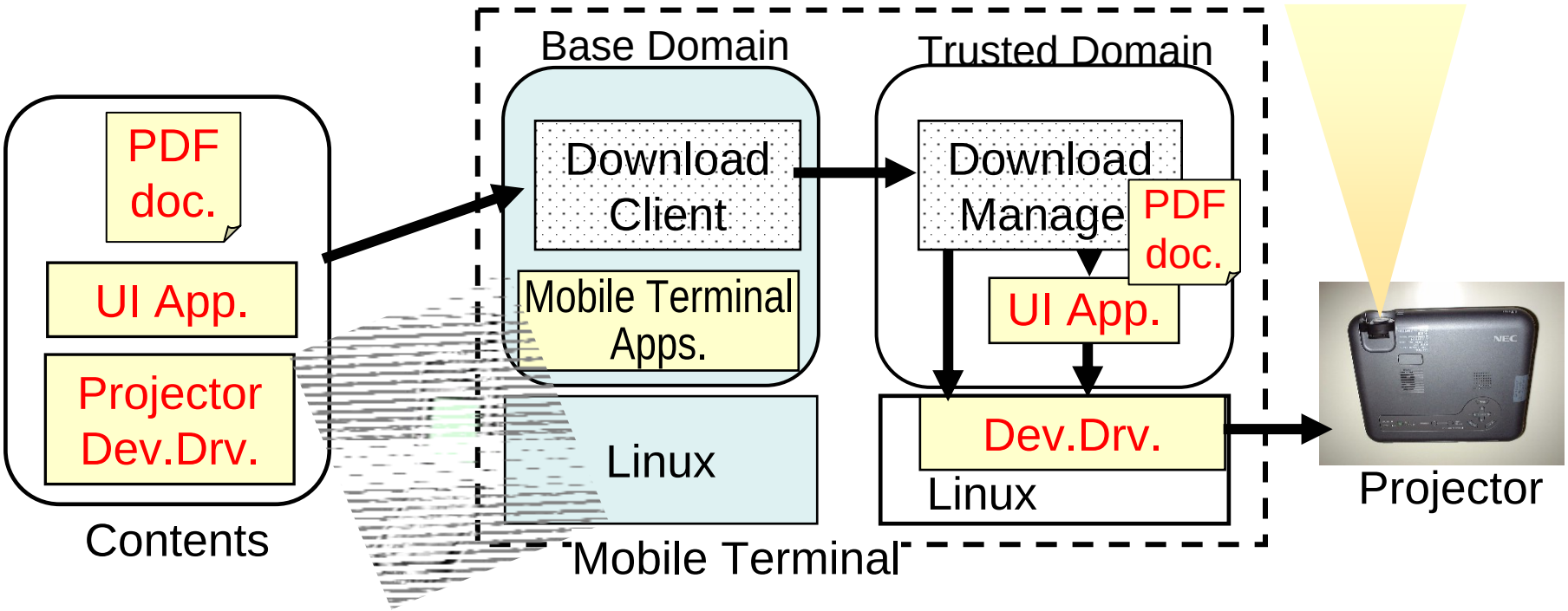
Access Control

Resources	CPU0	CPU1,2
LCD	R/W	R
I/Os	R/W	R
CPU0 Mem	R/W	
CPU1,2 Mem	R/W	R/W

FIDES sample

- Connect a mobile terminal to a projector
- Download a document with a device driver for the projector
- Install the driver ***into the kernel !***
- If the driver crashes – just reboot the domain. No harm in the base domain.

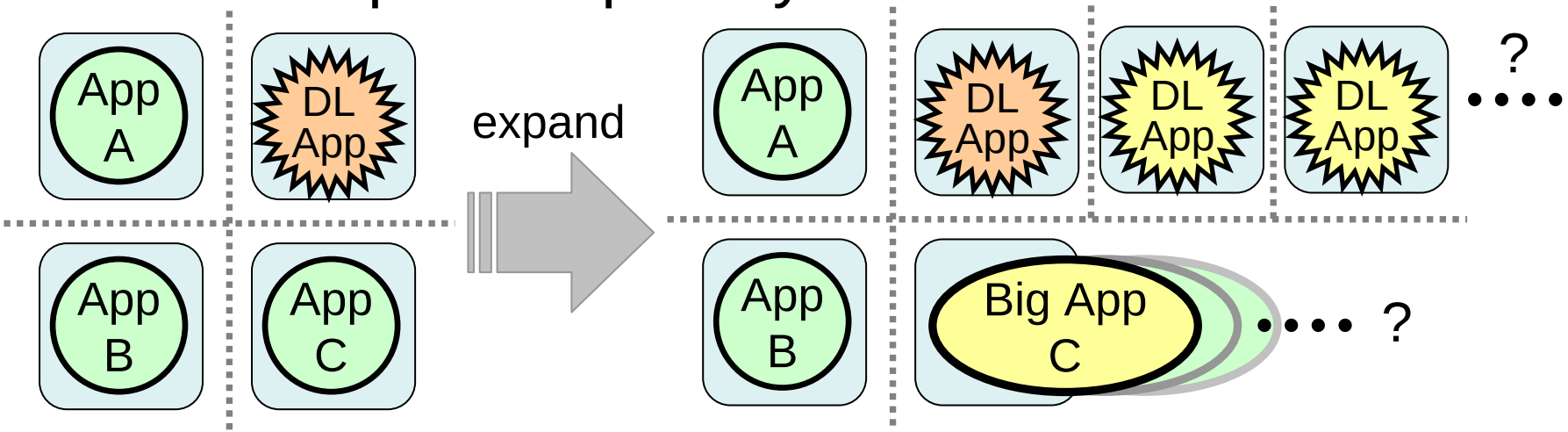




New Problems

- Physical partitioning ***is*** powerful.
  - Completely removes interference
  - Makes system quite robust
- But, deeply bound to multicore configuration
  - # of domains
  - Performance of each domains

  } limited
- How to expand capability ?



Chip Manufacturing Aspect

- Small quantity of customized LSI becomes difficult

  Rising cost of chip..

  – Design
  – Verification
  – Manufacturing

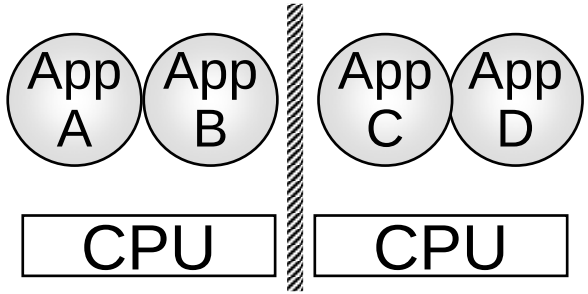
- Partitioning mechanism should be more flexible, scalable

  → Use the same chip for wider applications

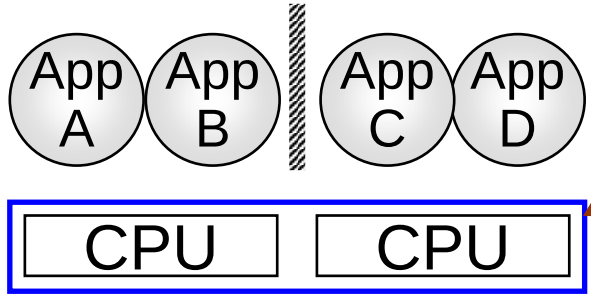
Shift to Logical Partitioning

(2) SMP technology

HW level partitioning



SW level partitioning



Robust
High Performance
Depend on # of Cores

Tradeoff

Flexible
Extendable

(1) more flexible partitioning

Our next approach

our first approach

Gradual shift
to logical partitioning
with general multicore

VIRTUS : Processor Virtualization

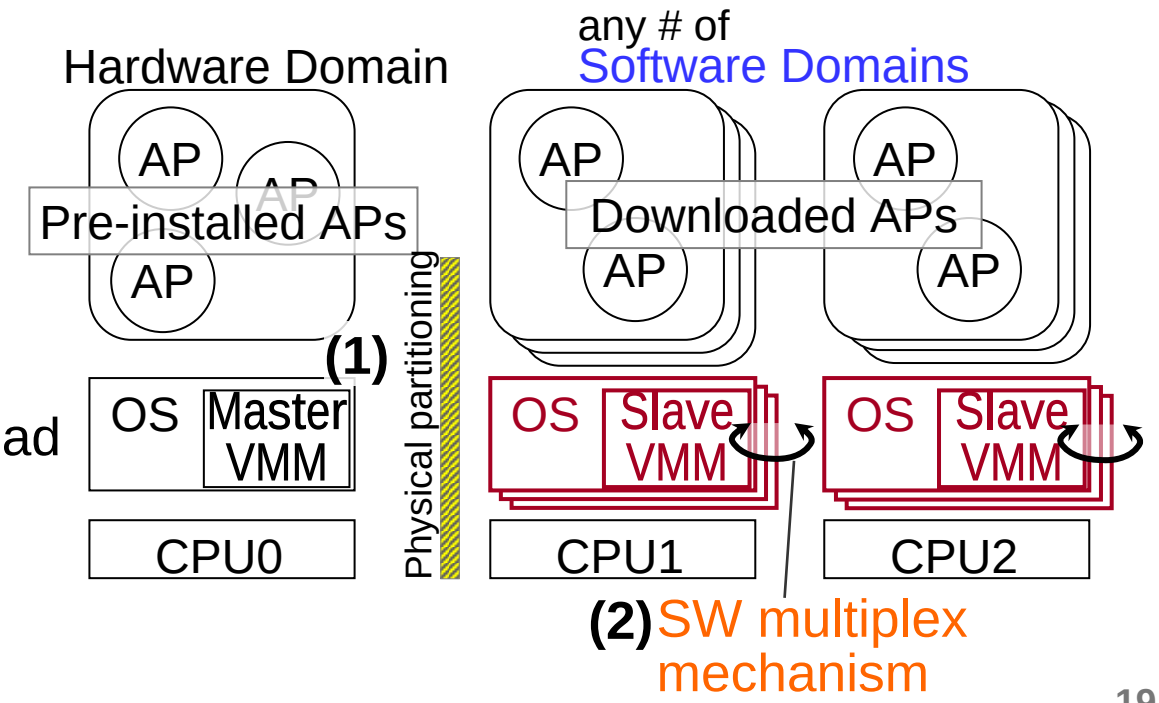
- Combines physical partitioning with virtualization technique

virtus = “virtual” in Latin

(1)Most important part = HW partition
→ physically protected

(2)Download Apps = SW partition using multiplex (VMMs) on other CPU cores
→ any # of domains

Also FIDES features:
- Lower performance overhead
- Block attacks
- Quick recovery

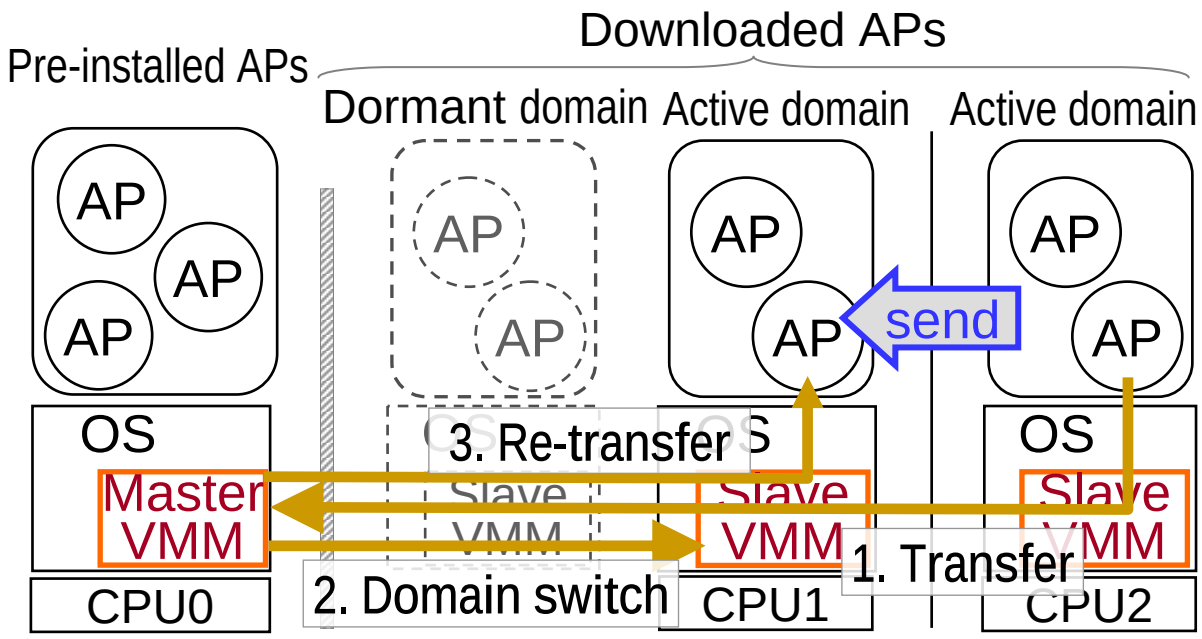


Asymmetric VMM

- Master VMM on CPU0 manages other slave VMMs
  - Communications through the Master VMM
  - Domain switch when talking to a dormant domain

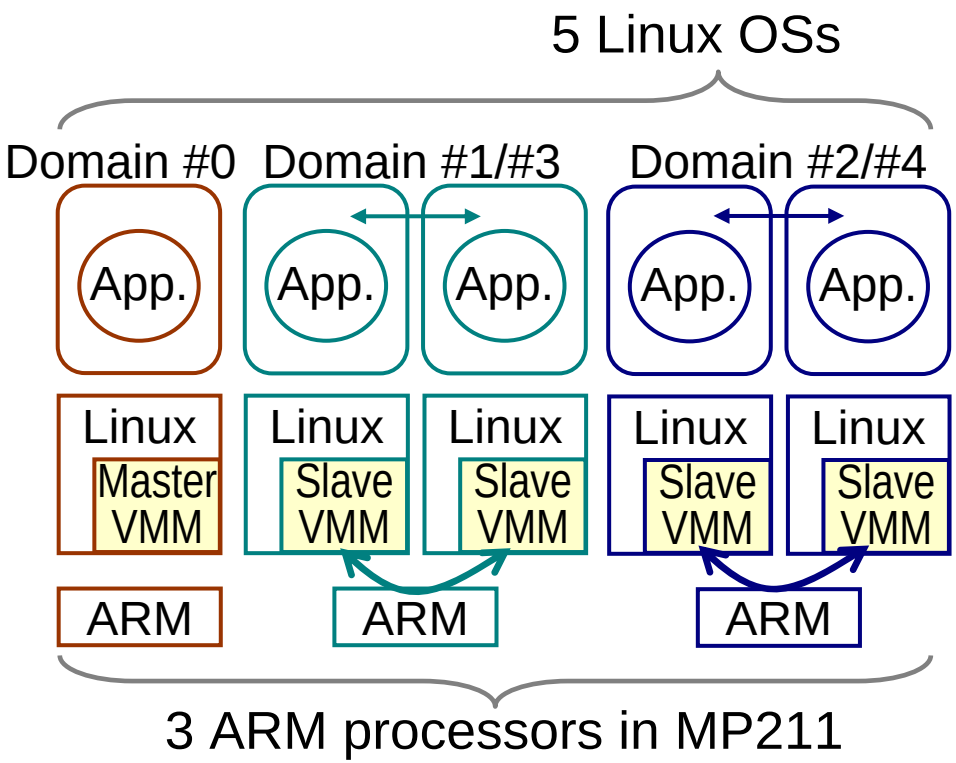
AVMM also avoids system freeze:

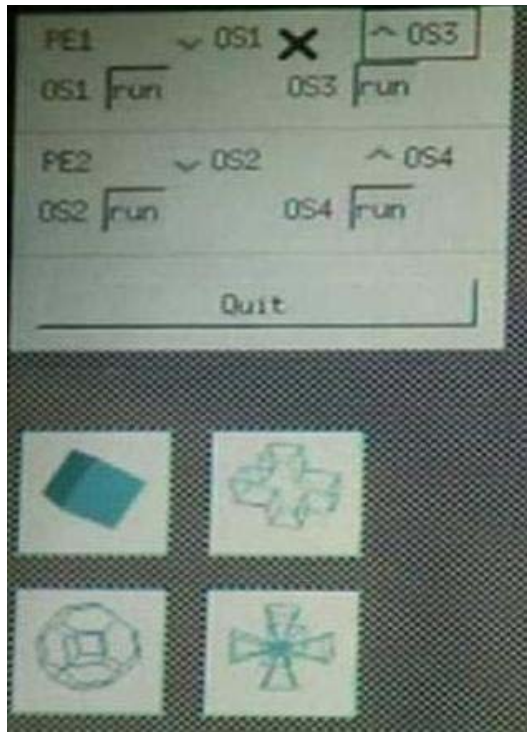
Access domain data via Master VMM
→ Never hang up while holding locks



VIRTUS Screenshot

- Creating 5 domains on 3 CPU cores
  - 1x Base domain
  - 4x untrusted domains for downloaded Apps
  - MP211 (3x ARM9)





SMP-type Multicore

- Embedded SMP chips appeared
  - MPCore (ARM/NEC Electronics)
  - SH-X3 (Renesas)

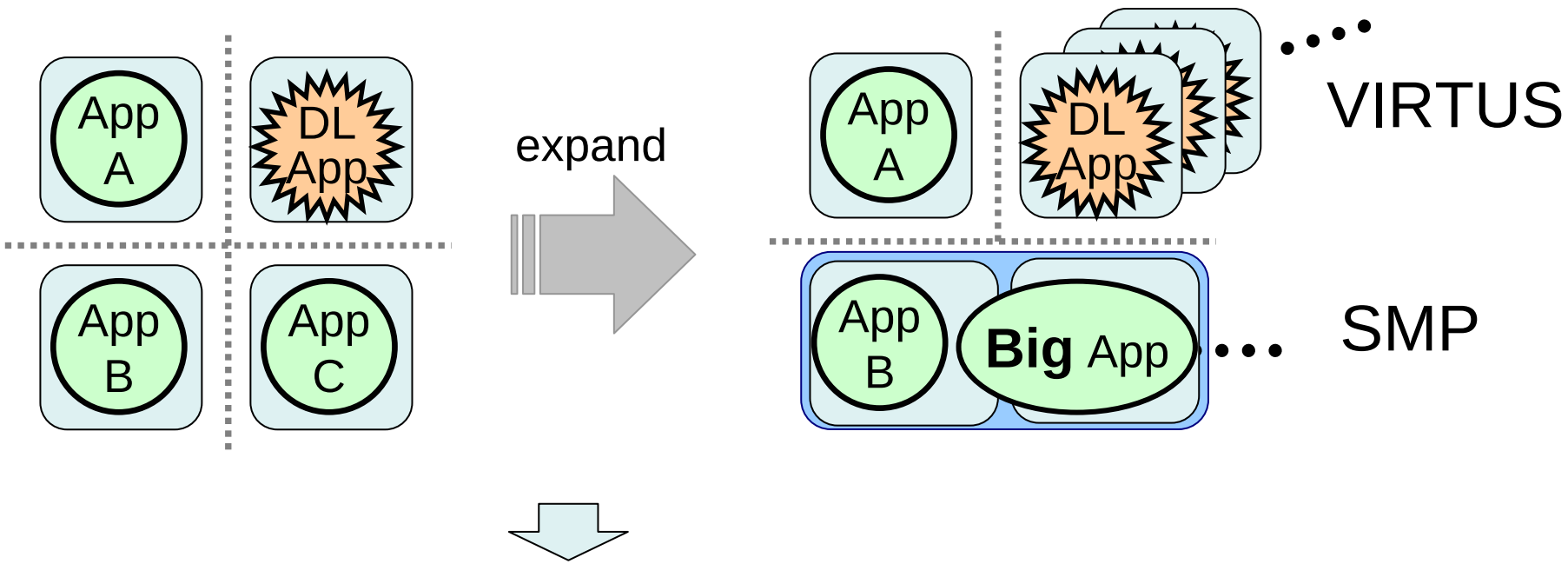
- SMP merits:
  - Performance scalability
  - Automatic load-balancing

  } aim to higher throughput

- demerits:
  - Poor performance assurance (→use affinity)
  - Poor robustness ← No partitioning

→ Introduce partitioning features to SMP architecture

Combination of VIRTUS and SMP



- Any # of domains
- Performance scalability

on the same multicore architecture

Summary & Future work