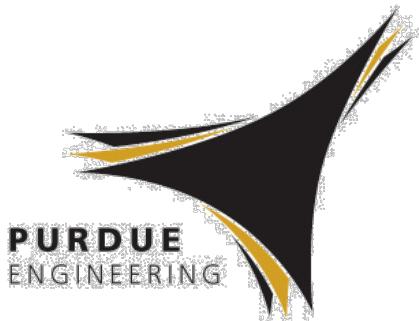


AI, app-defined OS, & the ossified Linux kernel

Felix Xiaozhu Lin 林小竹

<http://xsel.rocks>



Summary

Describe recent trends of OS research

Present three cases of our investigation

Share personal reflection on OS research

Self intro

- 2014 – now. Asst. prof, Purdue ECE
- 2014 PhD in CS. Rice U (advisor: Lin Zhong)
 - Thesis: OS for mobile computing
- 2008 MS + BS. Tsinghua U

Crossroads Systems
Exploration Lab



XSEL & Friends, 2017

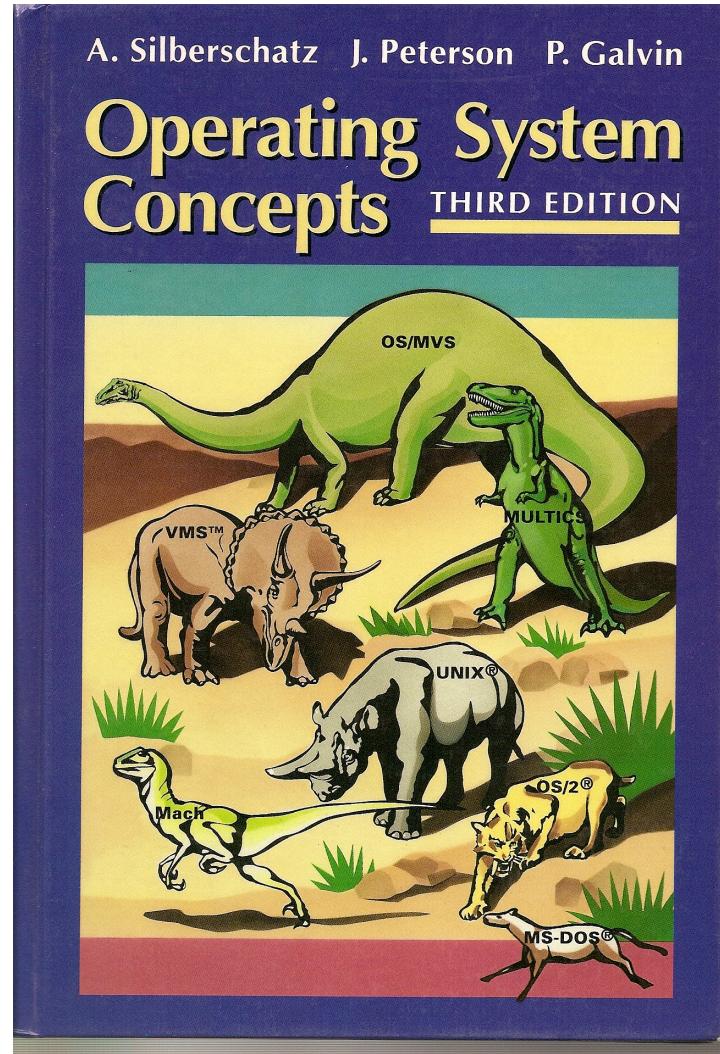
AI + big data: the workloads in our era

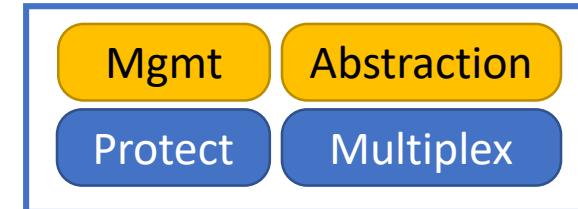


How does OS respond?

OS as in textbooks

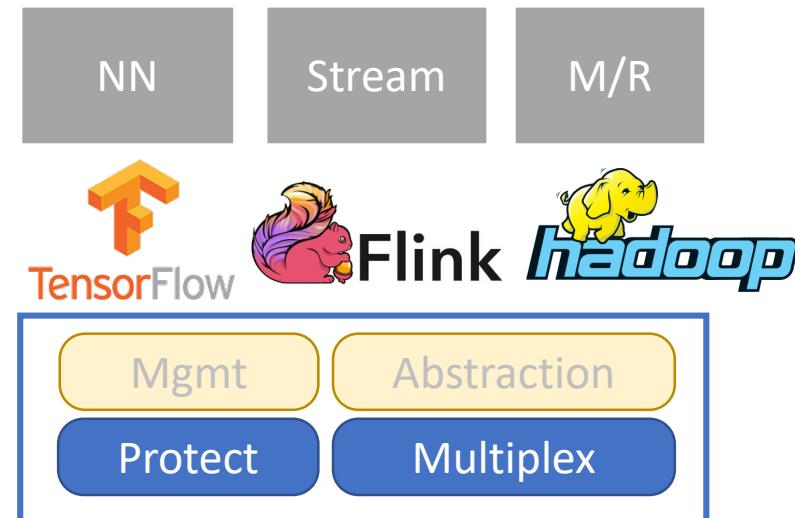
- Resource management
- Abstraction
- Protection
- Multiplexing





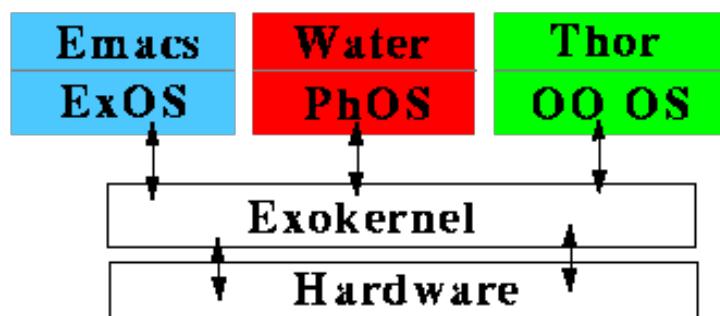
Hardware

OS in 2000



Hardware

OS in 2018



Exokernel: An Operating System Architecture for Application-Level Resource Management

Dawson R. Engler, M. Frans Kaashoek, and James O'Toole Jr.
M.I.T. Laboratory for Computer Science

SOSP 1995

Two premises of our research

1. Important apps define OSes
2. The Linux kernel is the new firmware

Like it or not ...

- Following the model: widely adopted
 - Various ML frameworks
 - Android
 - DPDK
 - RDMA
- Against the mode: less adopted
 - Library OS
 - Multikernel
 - Unikernels
 - (... and all kinds of research proposals)

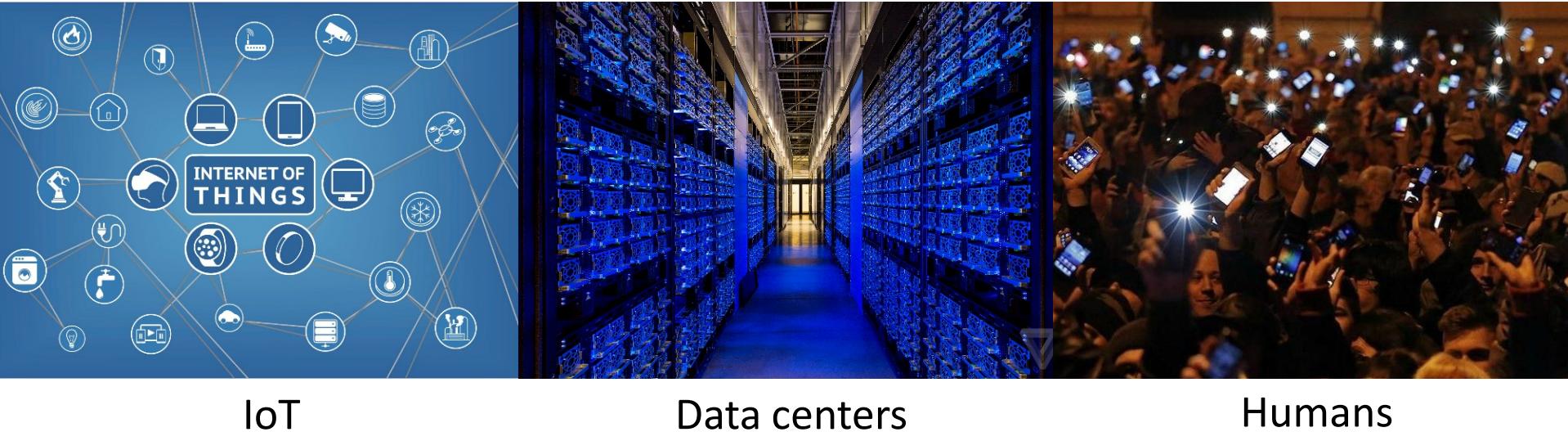
This talk: our exploration in three cases

- App-defined OSes
 - Case 1: Memory mgmt
 - Case 2: Storage
- The ossified Linux Kernel
 - Case 3: Transkernel

Case 1: App-defined memory management

Exploiting hybrid memory for
stream analytics

Background: Stream analytics in 1 minute



IoT

Data centers

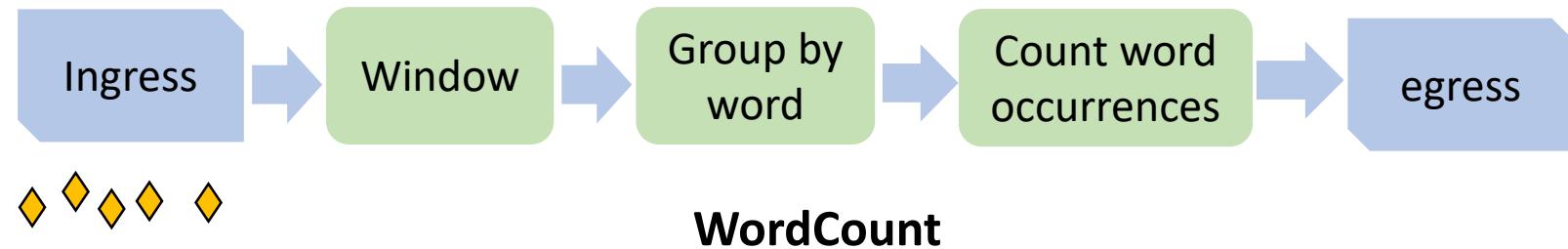
Humans

High input throughput: millions of events per sec

Low delay processing: sub-seconds

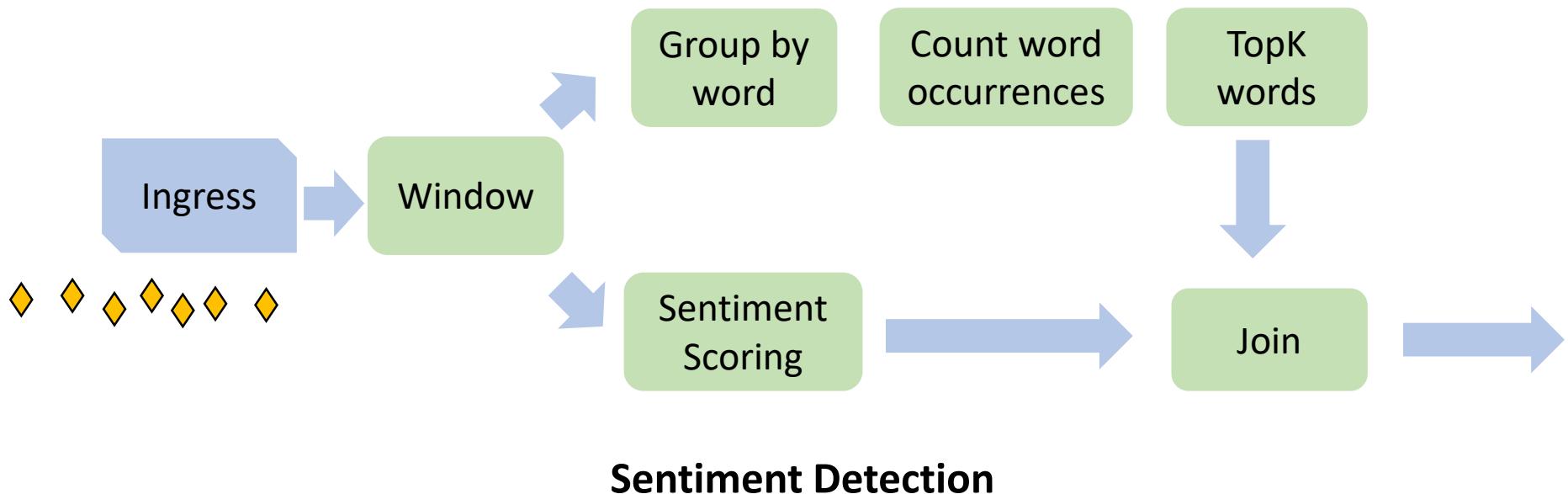
Background: streaming pipeline

Operators Computations consuming & producing streams
Pipeline A graph of operators

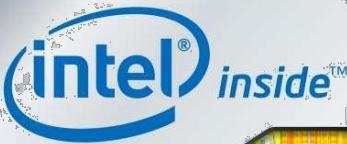
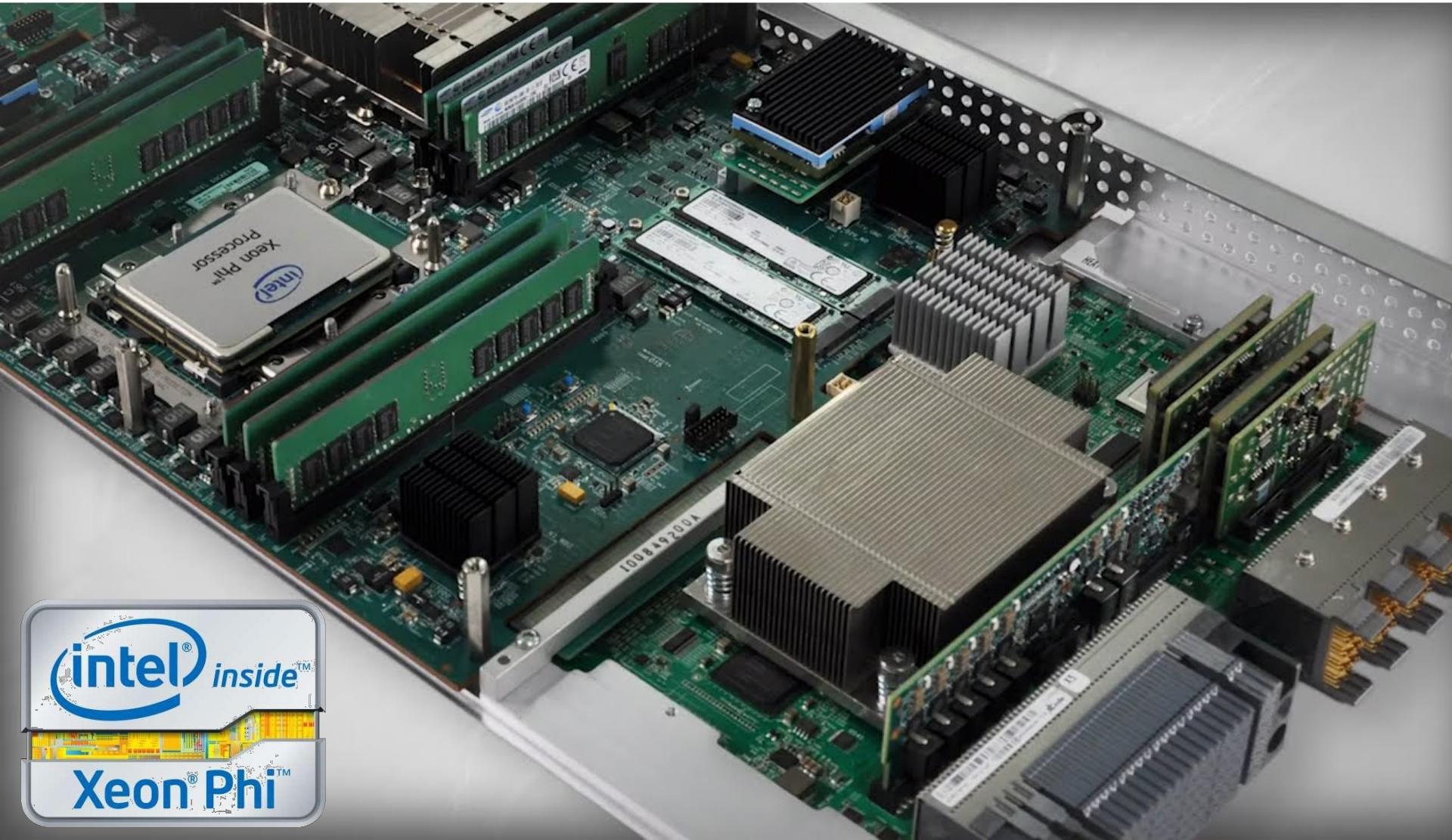


Background: streaming pipeline

Operators Computations consuming & producing streams
Pipeline A graph of operators

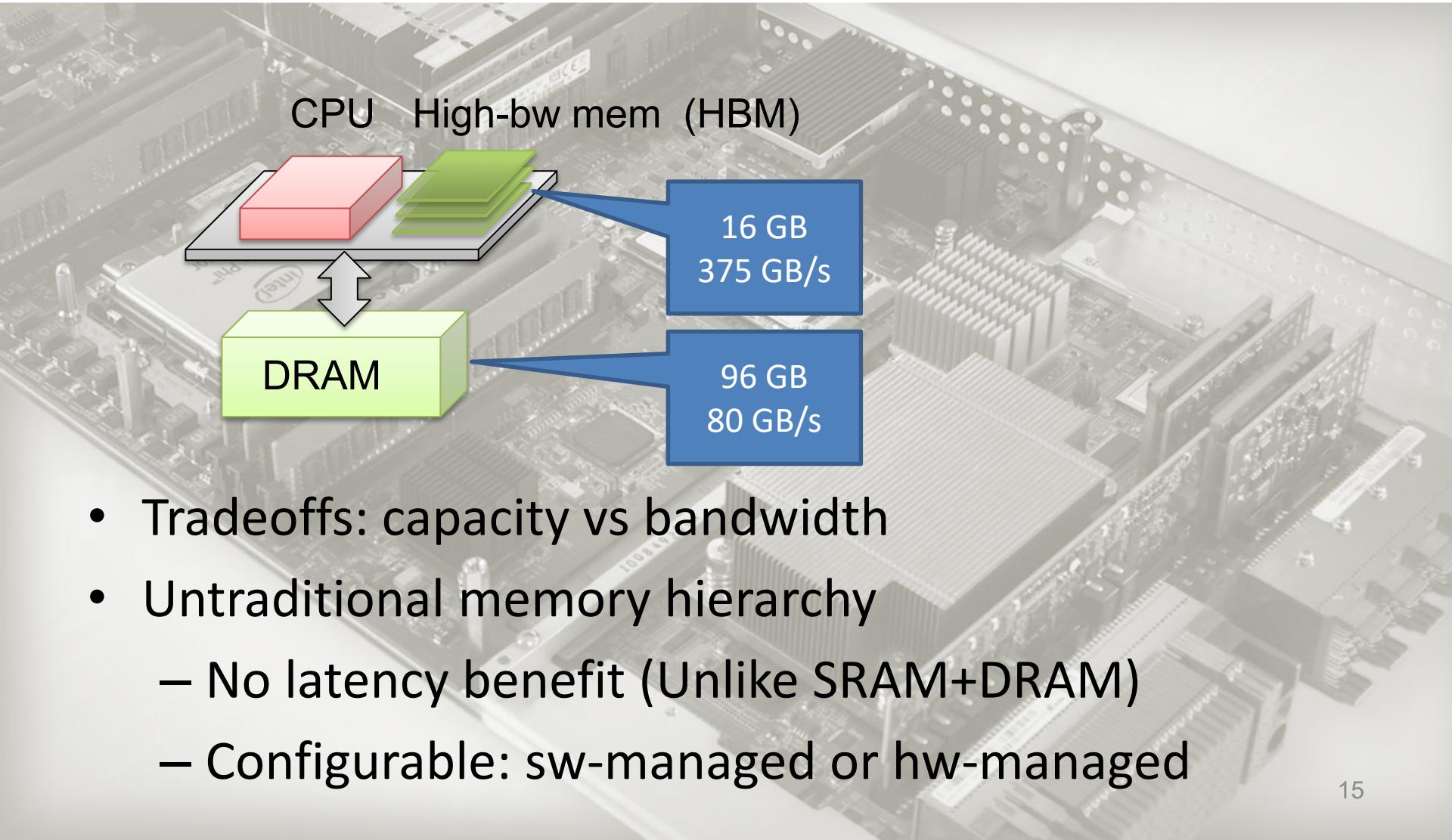


Hybrid high-bandwidth memory



Xeon® Phi™

Hybrid high-bandwidth memory



Accelerate stream analytics with HBM?

- The most expensive stream operators: **grouping**



Grouping traditionally built as Hash
Mismatch HBM!



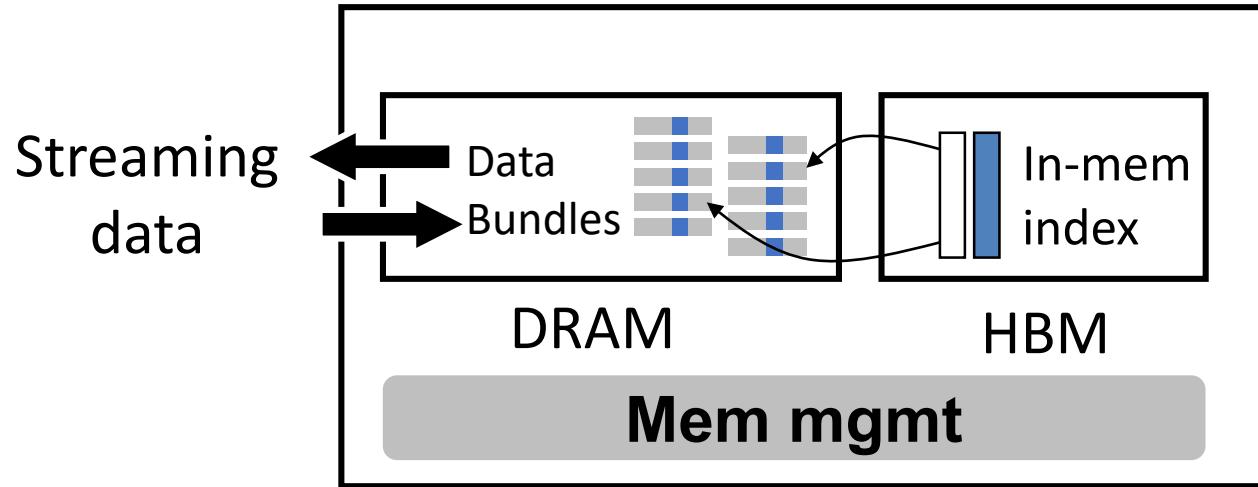
Grouping in Hash	High bandwidth memory
High capacity demand	Capacity limited
Extensive random access	Seq access + parallelism



HBM can accelerate grouping!

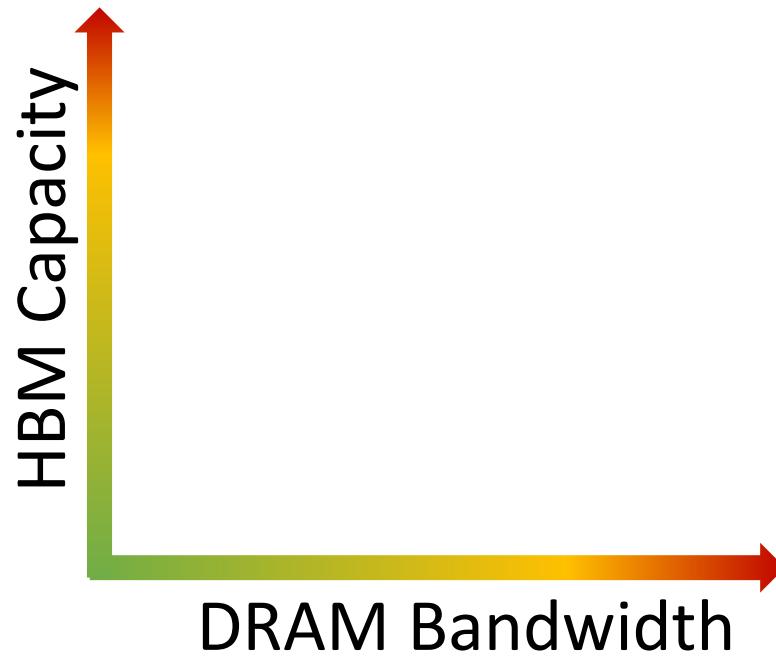
- ... with an unconventional algorithm choice
- Grouping: ~~hash~~ → sort
- Sort is **worse** than Hash on algorithmic complexity
 - $O(N \log N)$ vs. $O(N)$
- Yet, Sort **beats** Hash with ...
 - Abundant mem bw
 - High task parallelism
 - Wide SIMD (avx512)

A bold design: only use HBM for in-mem index of parallel sorting



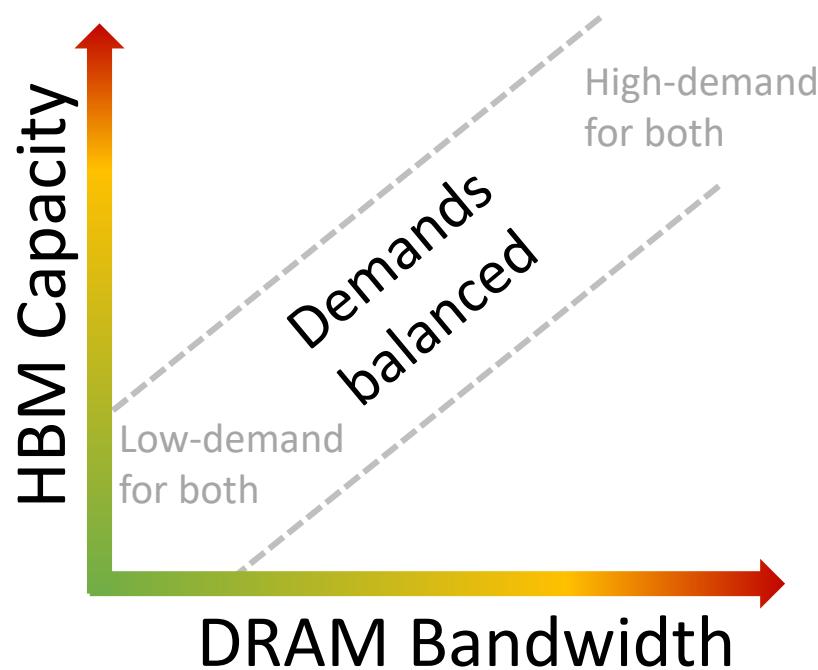
Memory allocator: balancing two limited resources

- Prevent either from becoming the bottleneck
- A single knob: where to allocate new index: HBM or DRAM?



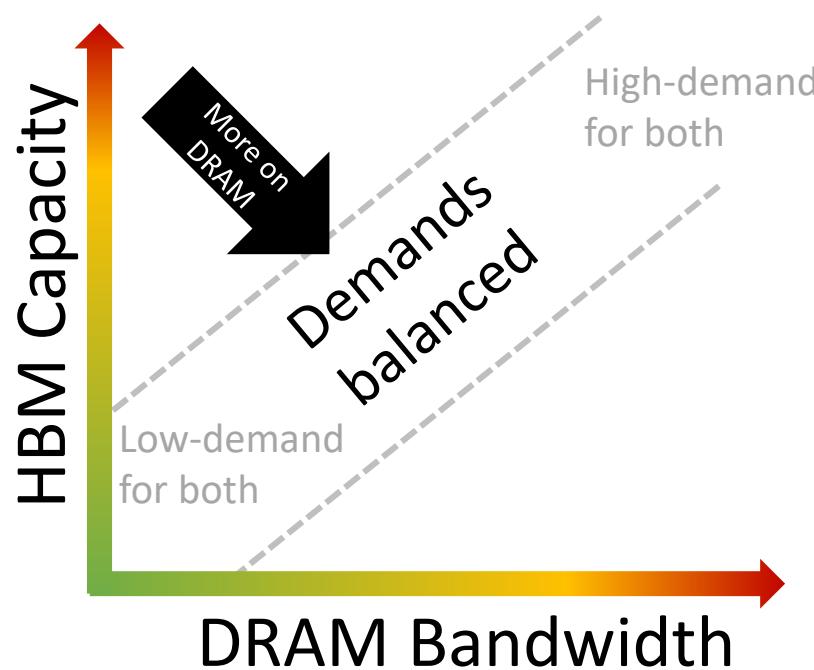
Memory allocator: balancing two limited resources

- Prevent either from becoming the bottleneck
- A single knob: where to allocate new index: HBM or DRAM?



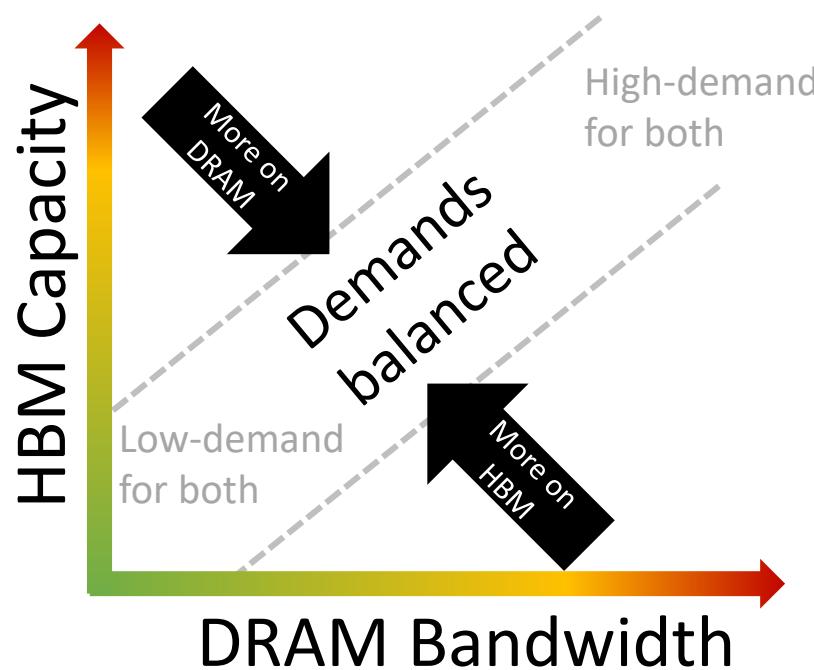
Memory allocator: balancing two limited resources

- Prevent either from becoming the bottleneck
- A single knob: where to allocate new index: HBM or DRAM?



Memory allocator: balancing two limited resources

- Prevent either from becoming the bottleneck
- A single knob: where to allocate new index: HBM or DRAM?



Case 1: First streaming engine optimized for hybrid memory

- Works on real hardware
- Best stream analytics performance on a single machine
- Generic memory allocators are inadequate!



This talk: our exploration in three cases

- App-defined OSes
 - Case 1: Memory mgmt
 - Case 2: Storage
- The ossified Linux Kernel
 - Case 3: Transkernel

Case 2: App-defined storage

A data store for video analytics



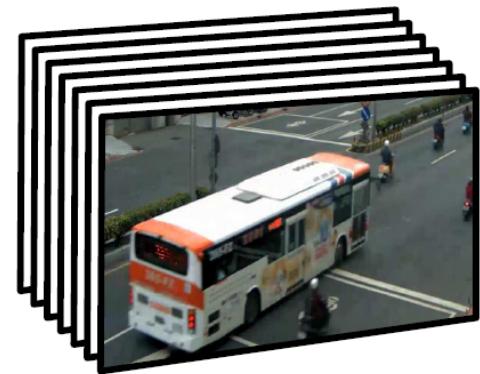
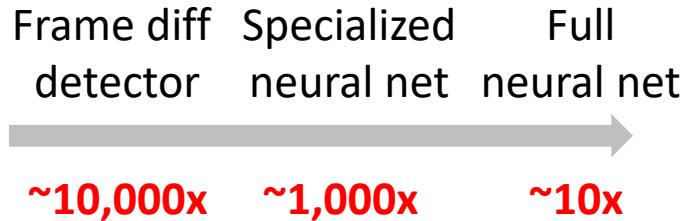
Pervasive cameras. Large videos.

- 130M surveillance cameras shipped per year
- Many institutions run > 200 cameras 24x7
- A single camera produces 24 GB video per day



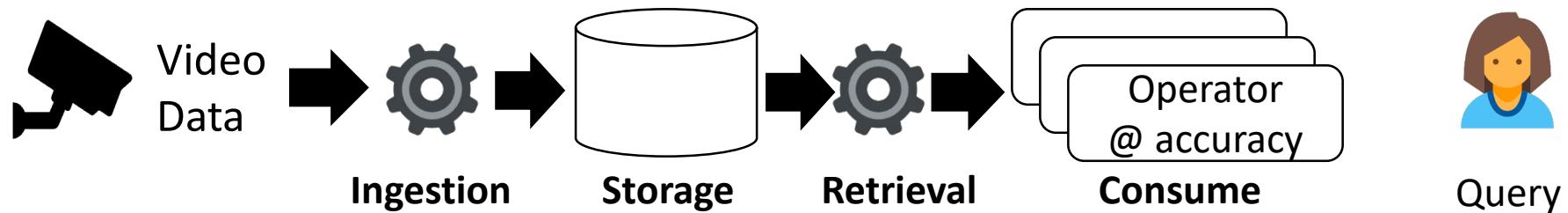
Retrospective video analytics in 30 seconds

- A query: “find all white buses appeared yesterday”
- A cascade of operators



- Query picks operator accuracies
 - Lower accuracy → lower cost → faster execution

Need for a video store for retrospective analytics



“Aren’t there many video databases already?”

Yes.

But they are for **human** consumers.

Not for **algorithmic** consumers.

Central design issue: choose storage formats catering to operators

- Video format: resolution, frame rate, crop, compression level, color channel... **Many knobs!**

Central design issue: choose storage formats catering to operators

- Video format: resolution, frame rate, crop, compression level, color channel... **Many knobs!**
- Higher accuracy → richer format & higher cost



200p 1FPS
(~100KB/s)



720p 30FPS
(~400KB/s)

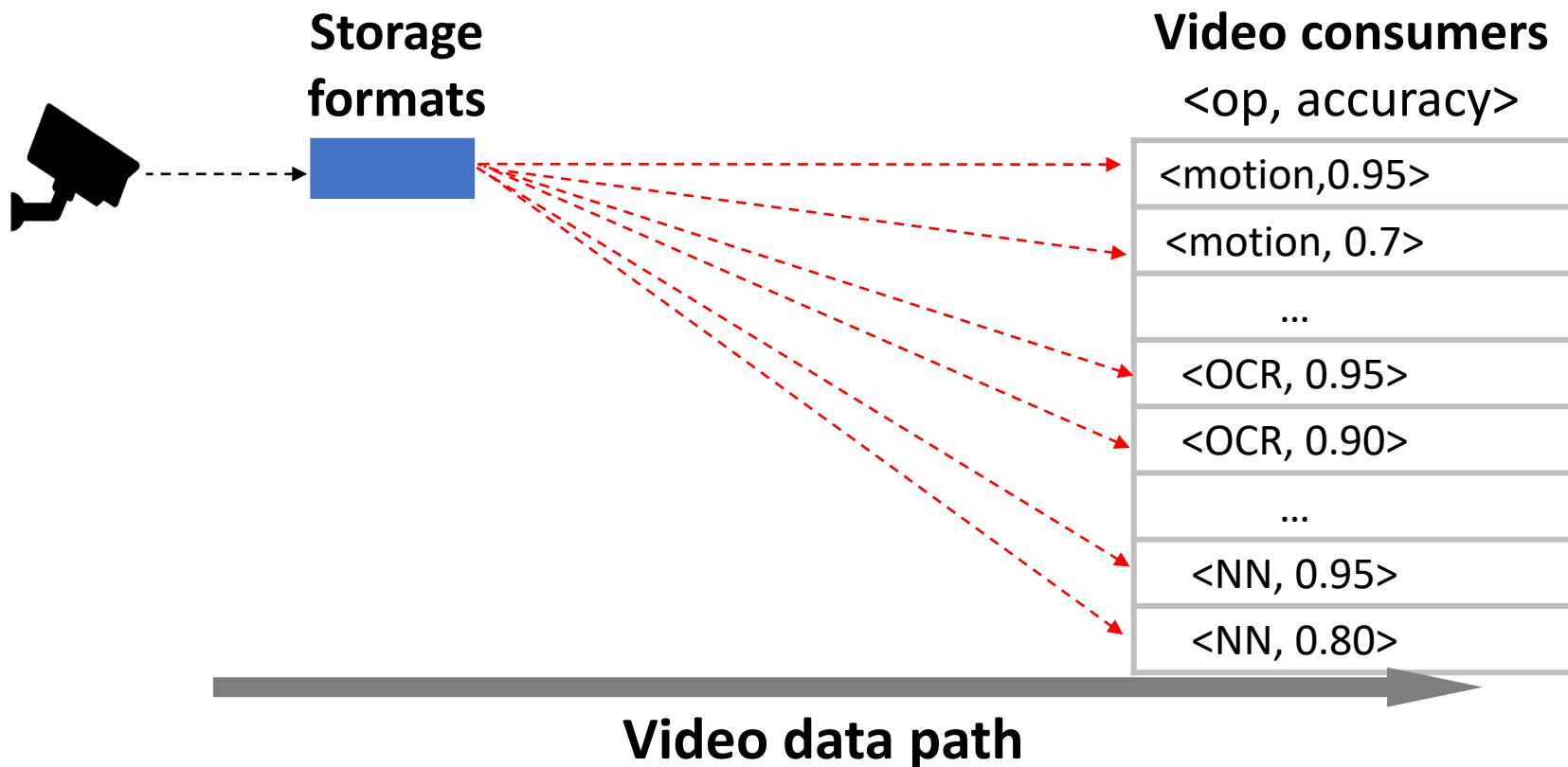
Actual tuning of formats is complex!

Key problem: What video formats to store?

1. Retrieving formats must be fast enough to feed operators
2. Formats must be rich enough to satisfy all accuracy needs

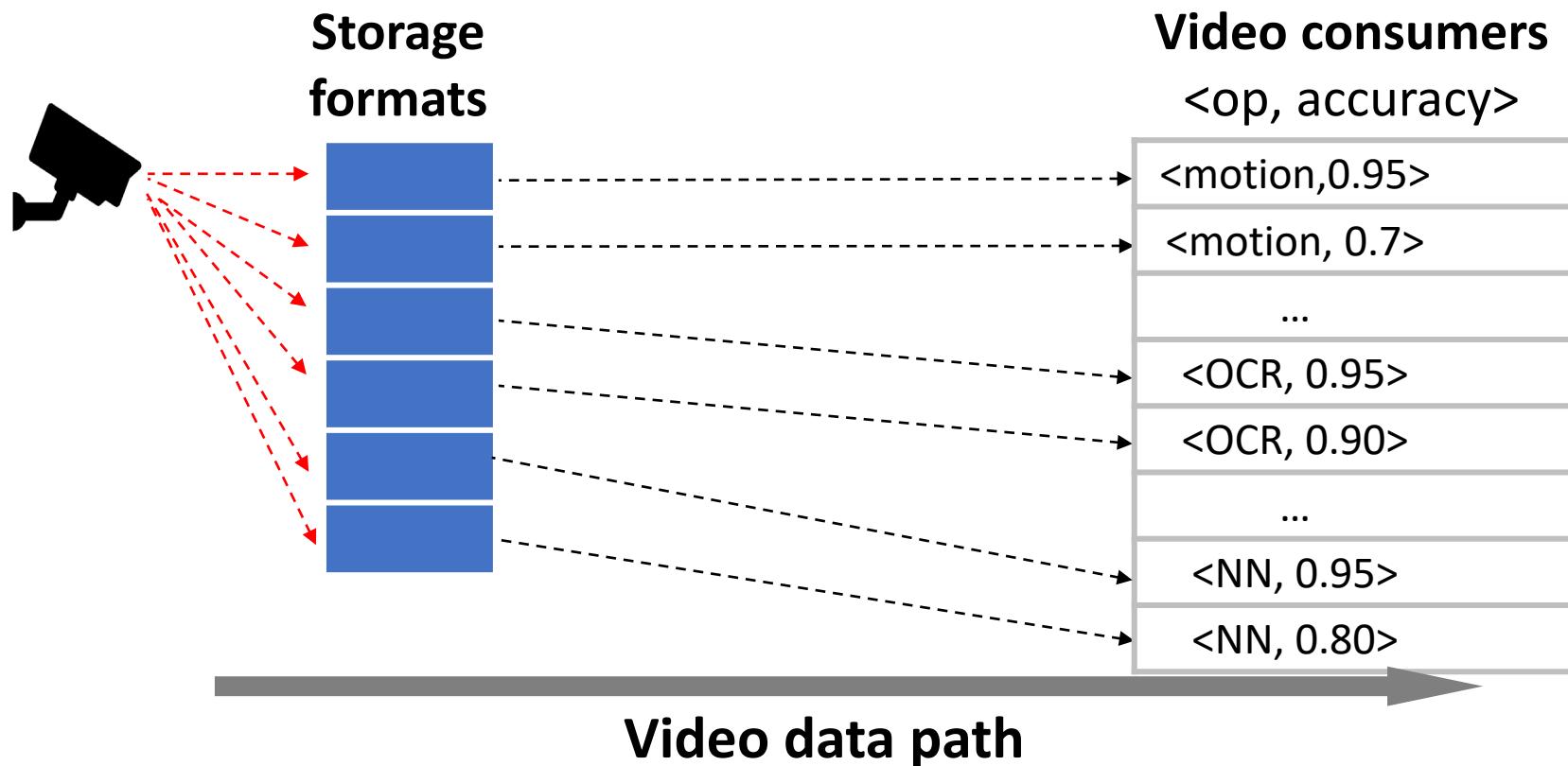
Storing one **unified** format?

Retrieving & decoding slows down operators 😞



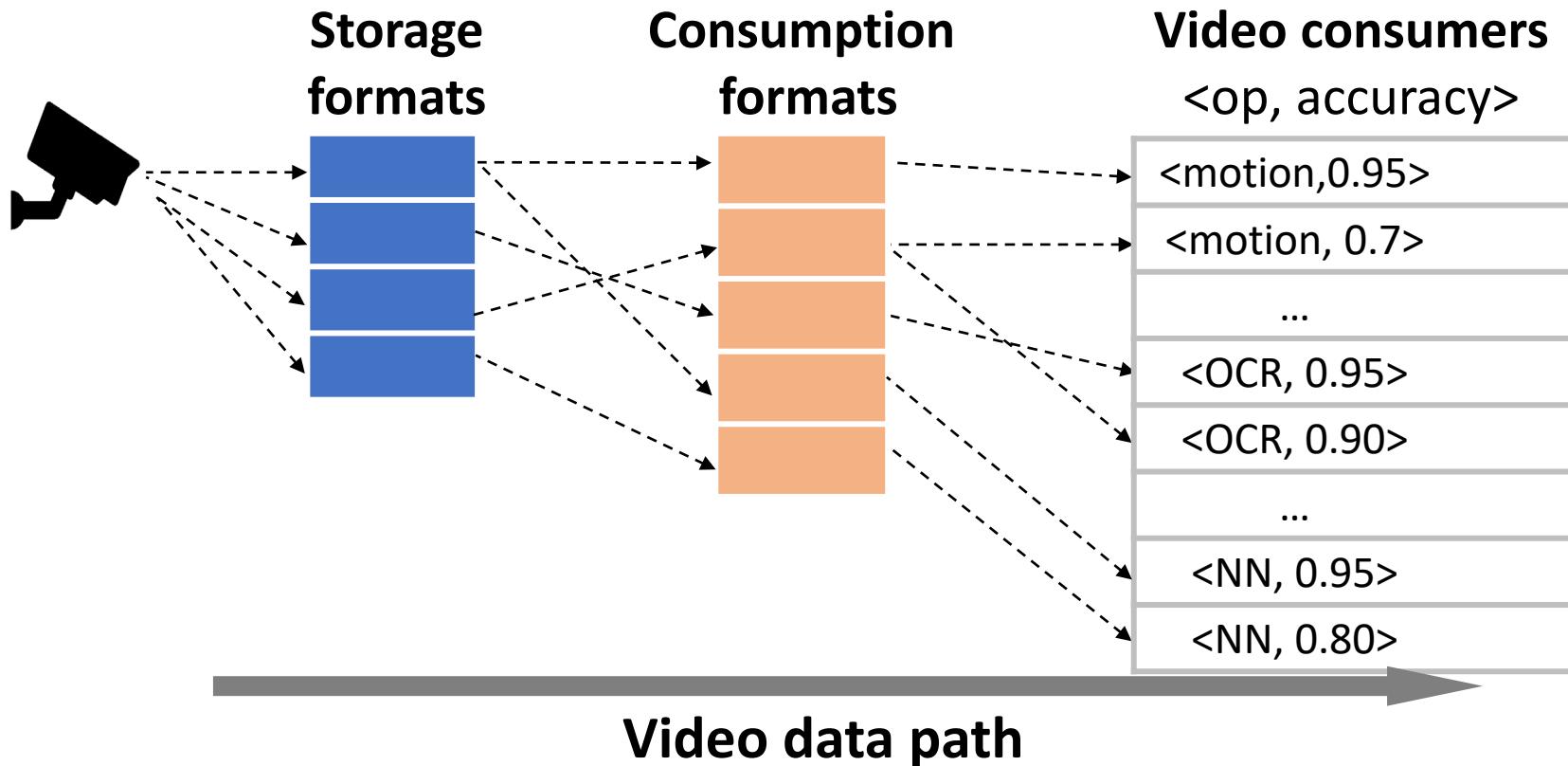
Storing **all** needed formats?

Ingestion & storage are expensive 😞





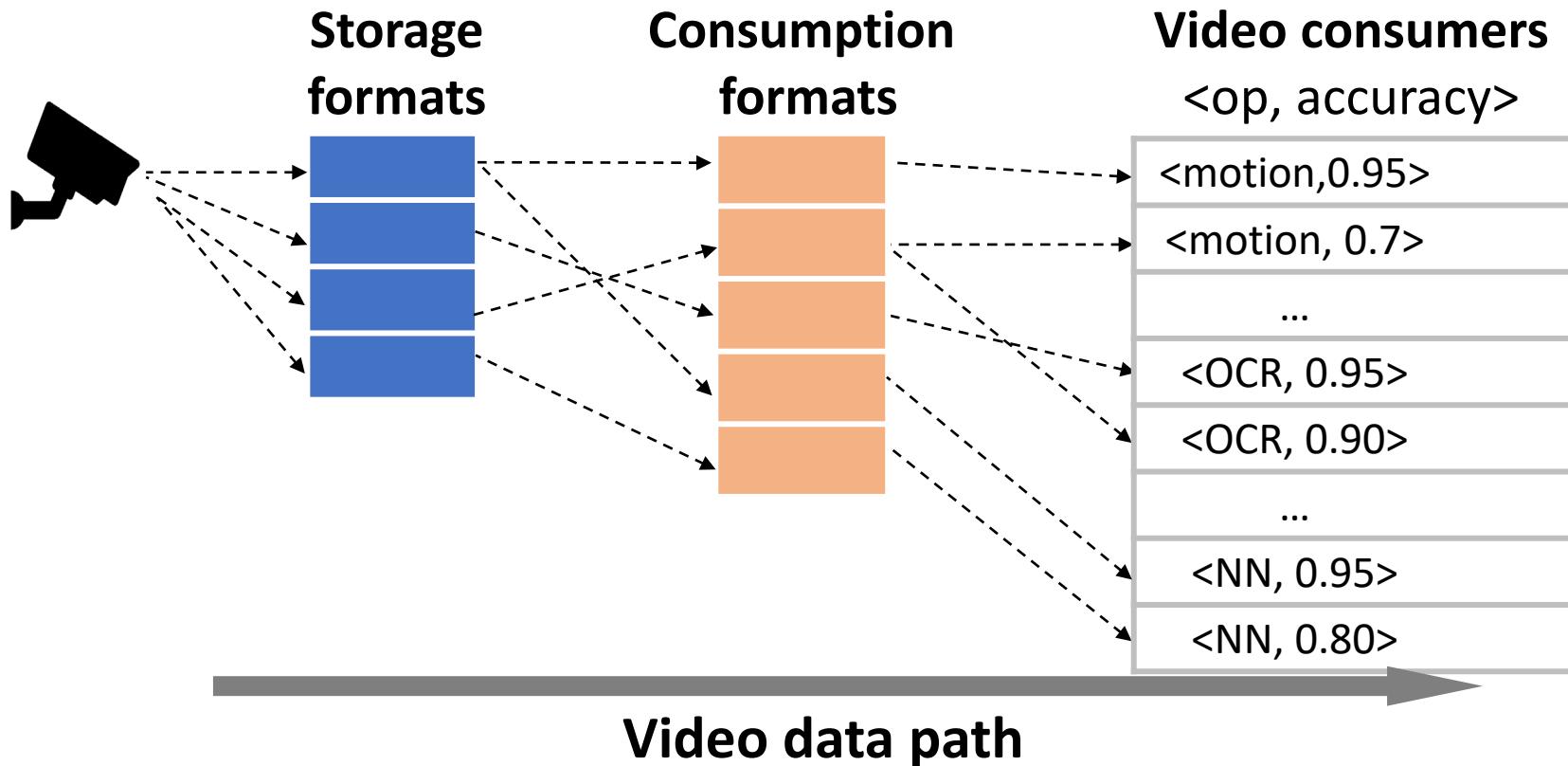
Proposal: Store minimum necessary formats





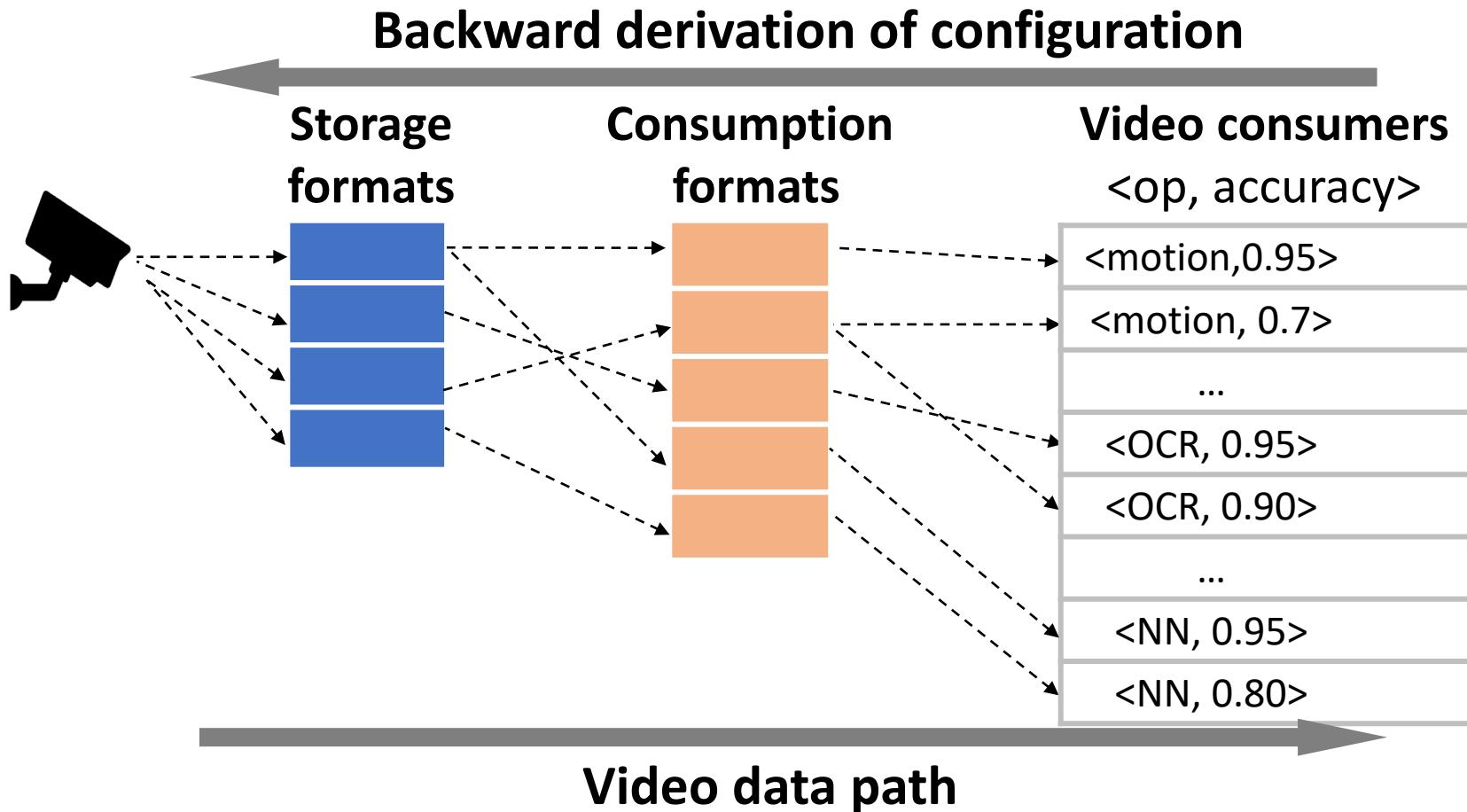
Proposal: Store minimum necessary formats

Key Challenge: Too many knobs to tune!

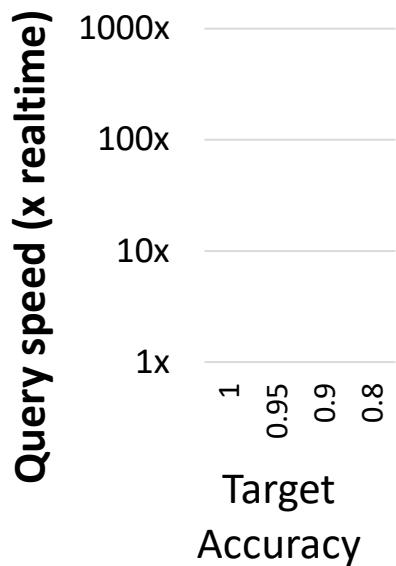




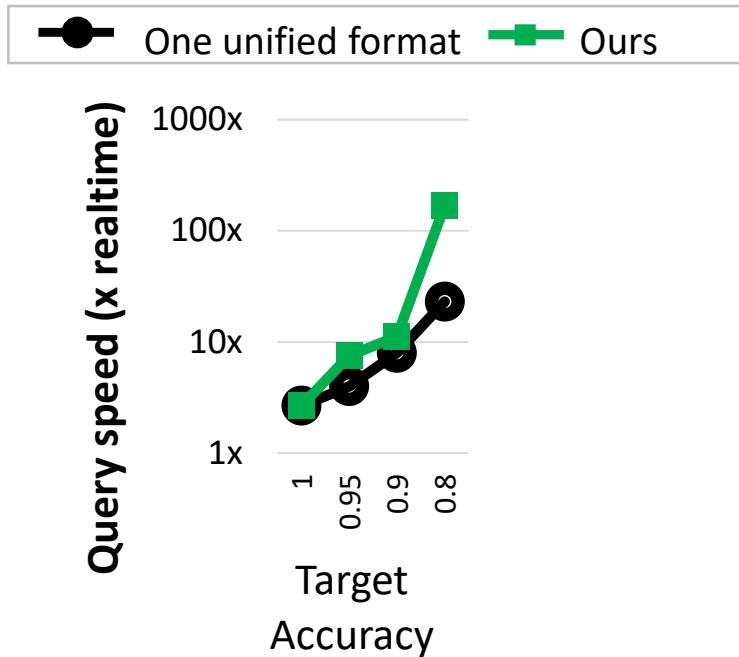
Proposal: Store minimum necessary formats & derive them automatically!



Benefit: faster analytics



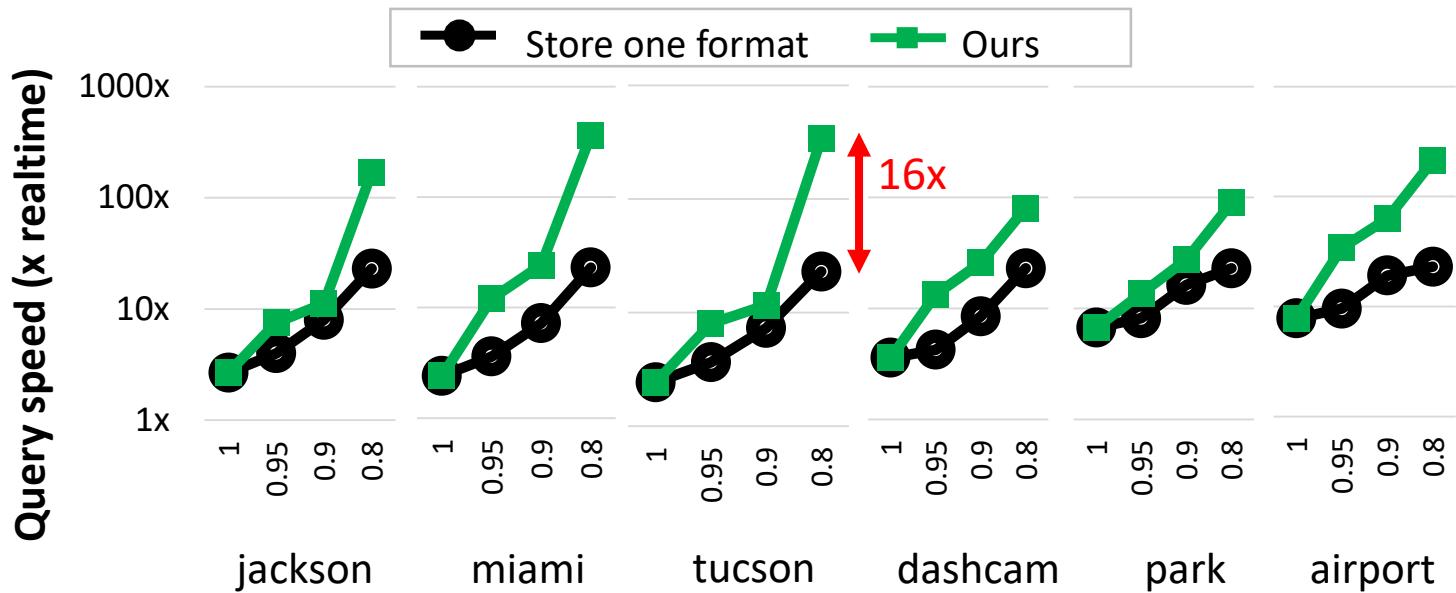
Benefit: faster analytics



Why?

- Lower accuracy → analytics tolerates cheaper video formats
- Video retrieval should be proportionally cheaper!

Benefit: faster analytics



Why?

- Lower accuracy → analytics tolerates cheaper video formats
- Video retrieval should be proportionally cheaper!

Case 2: First data store for retro. video analytics



Baking analytics demands into storage decisions

- Showing clear advantage over generic file systems or storage layers

This talk: our exploration in three cases

- App-defined OSes
 - Case 1: Memory mgmt
 - Case 2: Storage
- The ossified Linux Kernel
 - Case 3: Transkernel

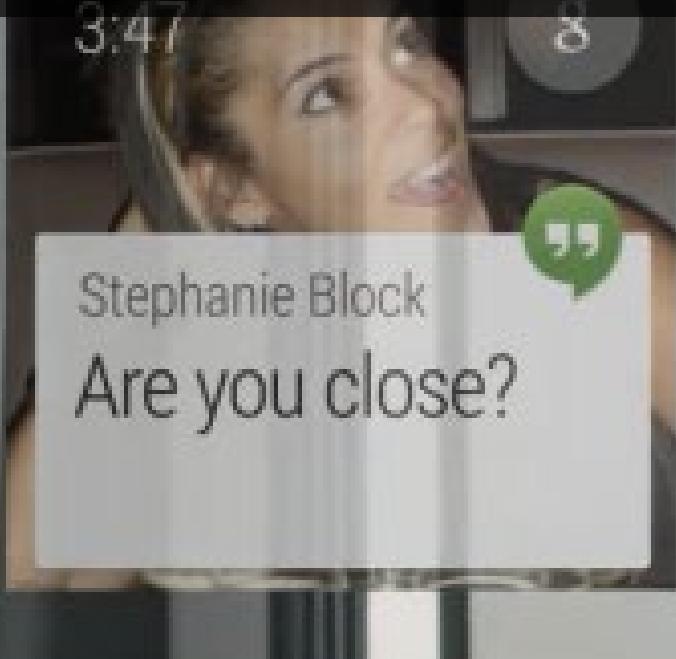
Case 3: enhancing the ossified Linux kernel

A new OS structure for
device driver offloading

A commodity kernel still irreplaceable

- The major reason: **device drivers**
 - Linux: a common environment for all drivers
 - >70% Linux kernel source are device drivers
- All other OS functions have alternatives!

Problem: suspend/resume in ephemeral tasks



- Mobile/IoT run frequent, ephemeral tasks
 - Android smartwatch: each minute
 - Each task is short-lived
 - Smartwatch: 10 secs
 - Background task: < 1 sec

Under the hood...

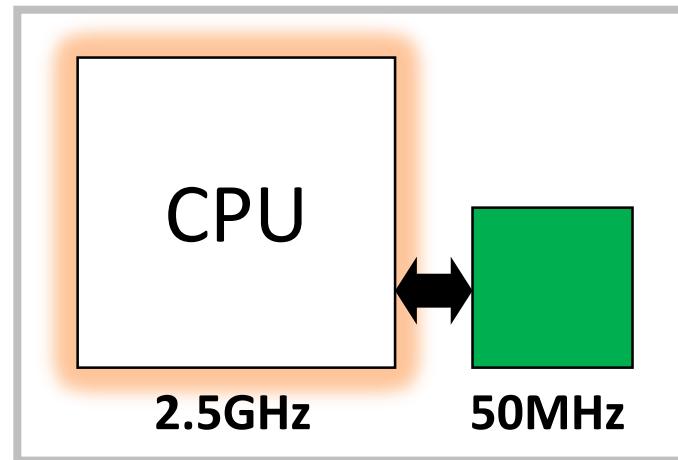


Suspend/Resume Is Slow

	Nexus 5	Gear	Note 4	Panda
Suspend	119 ms	191 ms	231 ms	262 ms
Resume	88 ms	159 ms	316 ms	492 ms

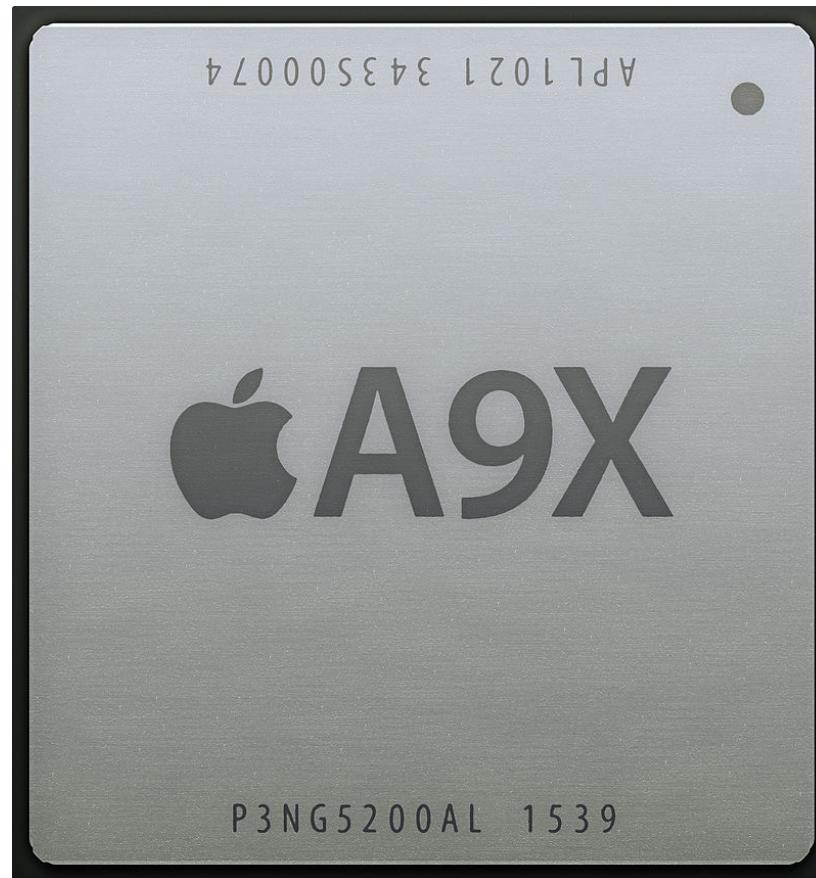
Why? IO devices are slow

Proposal: run these driver paths
on a low-power tiny core

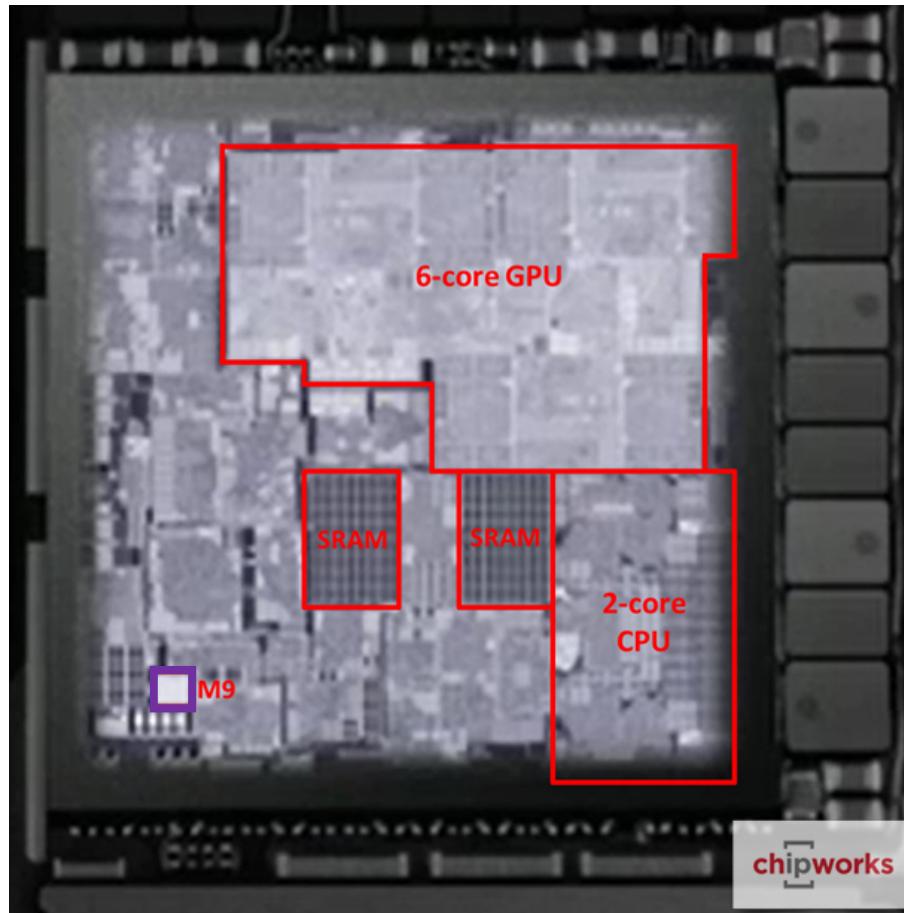


What are these tiny cores?

What are these tiny cores?



What are these tiny cores?

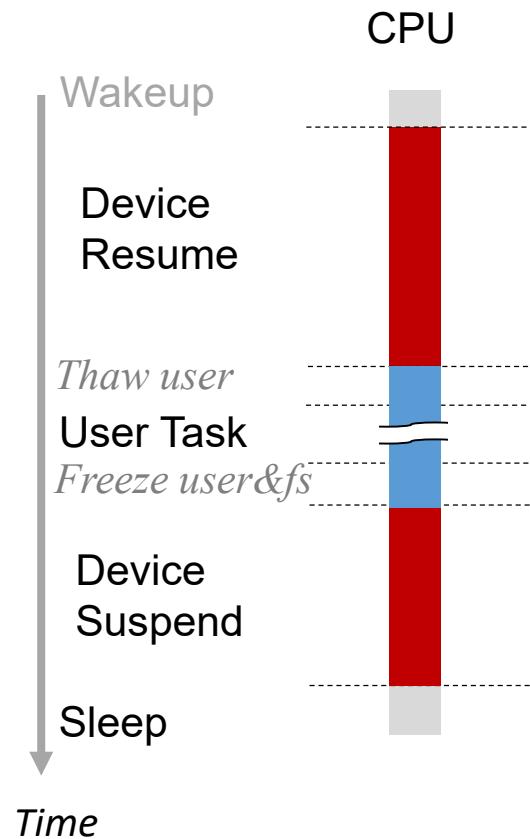


How can a tiny core help?

- Idling more efficiently
 - 3 mW vs 30 mW
- Executing kernel more efficiently
 - Small code working set
 - Less predictable control flow

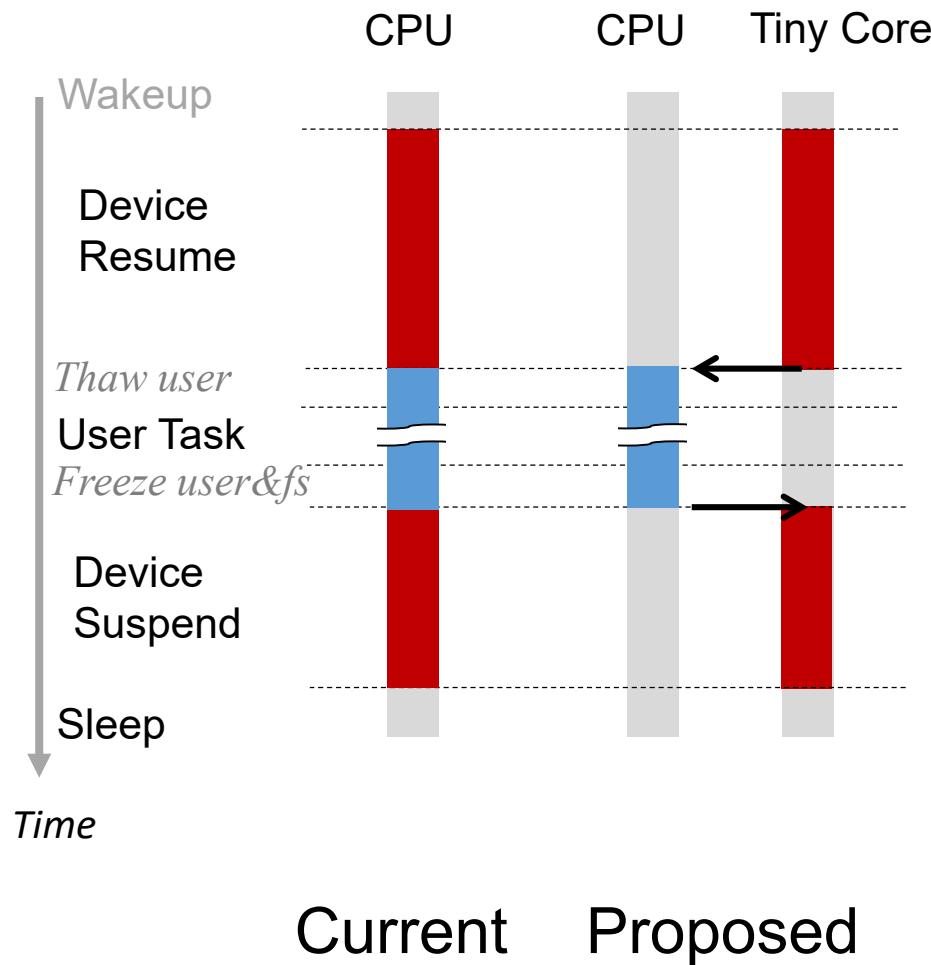
1. J. Mogul, J. Mudigonda, N. Binkert, P. Ranganathan, and V. Talwar. Using asymmetric single-isa cmps to save energy on operating systems. Micro, IEEE, 2008

The workflows



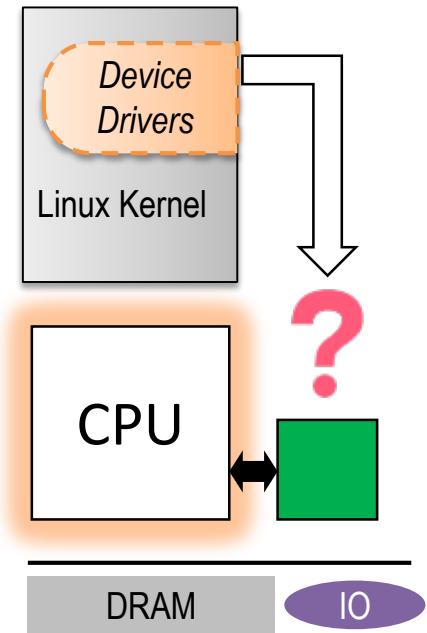
Current

The workflows



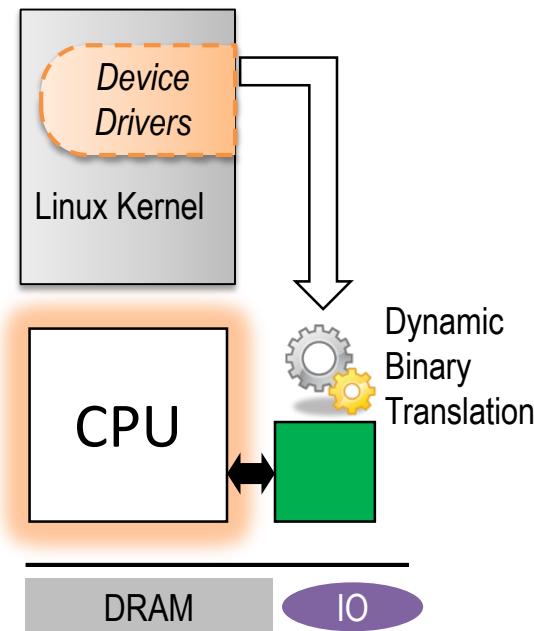
How?

- The tiny core is very wimpy
 - Essentially a microcontroller
- Has a different ISA
 - More aggressive than big.LITTLE
- Re-engineering the kernel?
- “Linux kernel is the new firmware”



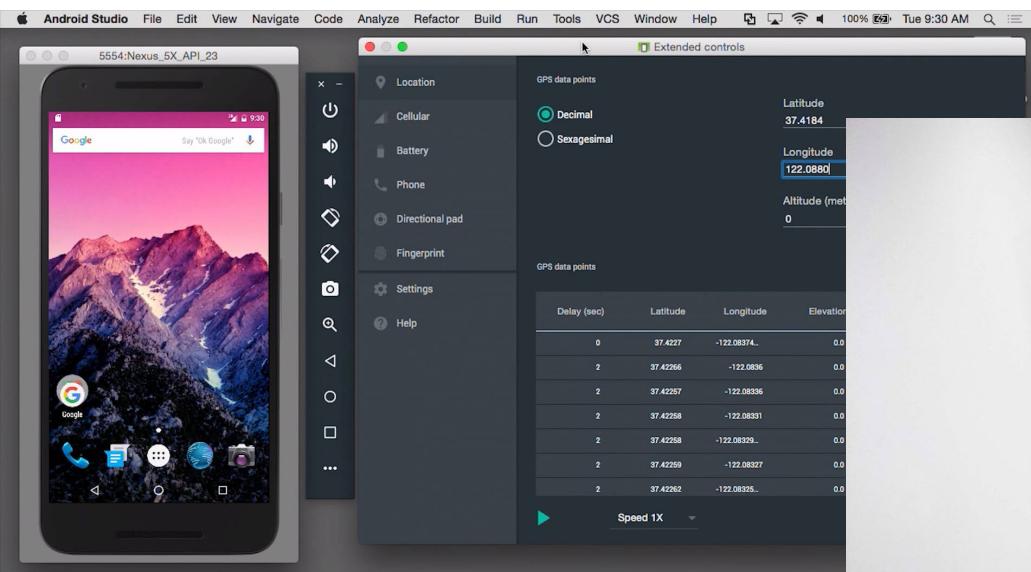
Proposal: use dynamic binary translation

- Empower the tiny core to execute *unmodified* kernel binaries



Dynamic binary translation in 30 sec

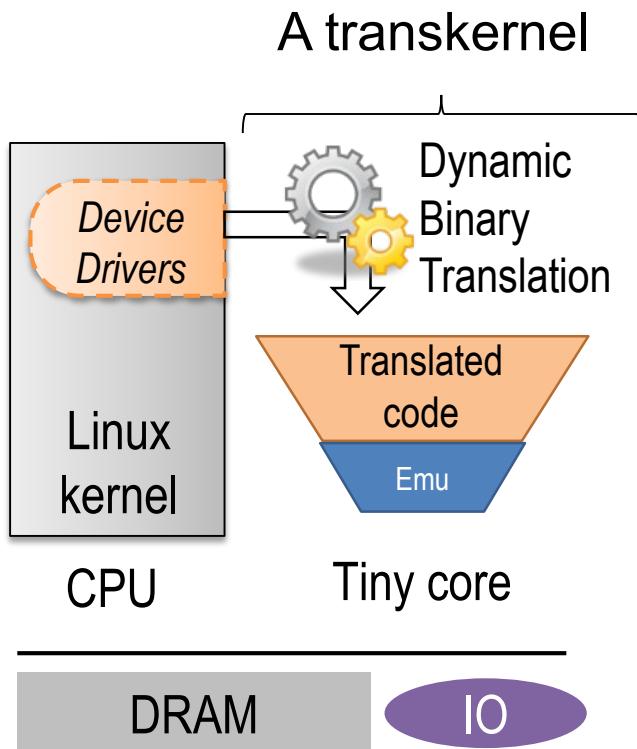
- The technology behind QEMU and a PS emulator



“This sounds impractical”

- Dynamic bin translation (DBT) is known expensive
 - > 10x overhead
- No one runs DBT on microcontrollers
 - Always weak over strong

Solution: the Transkernell model

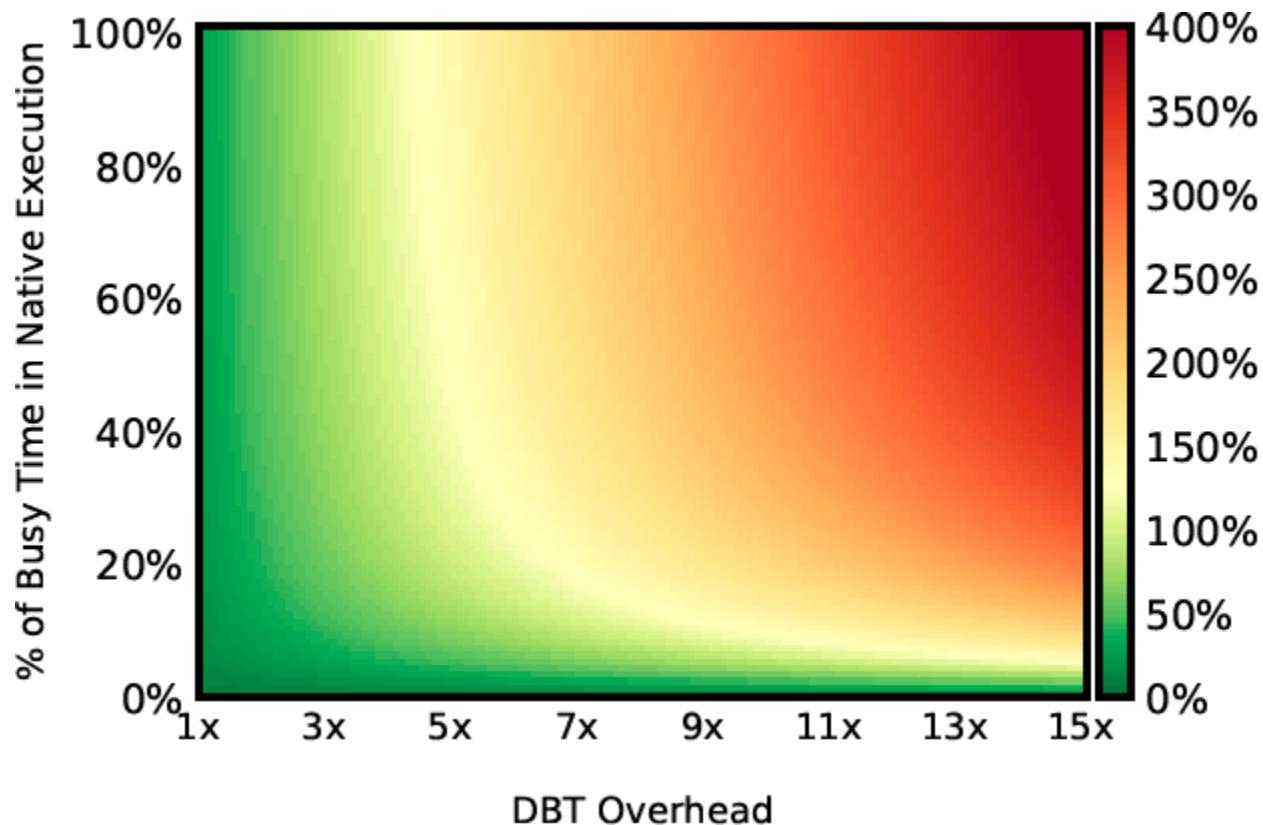


Key ideas:

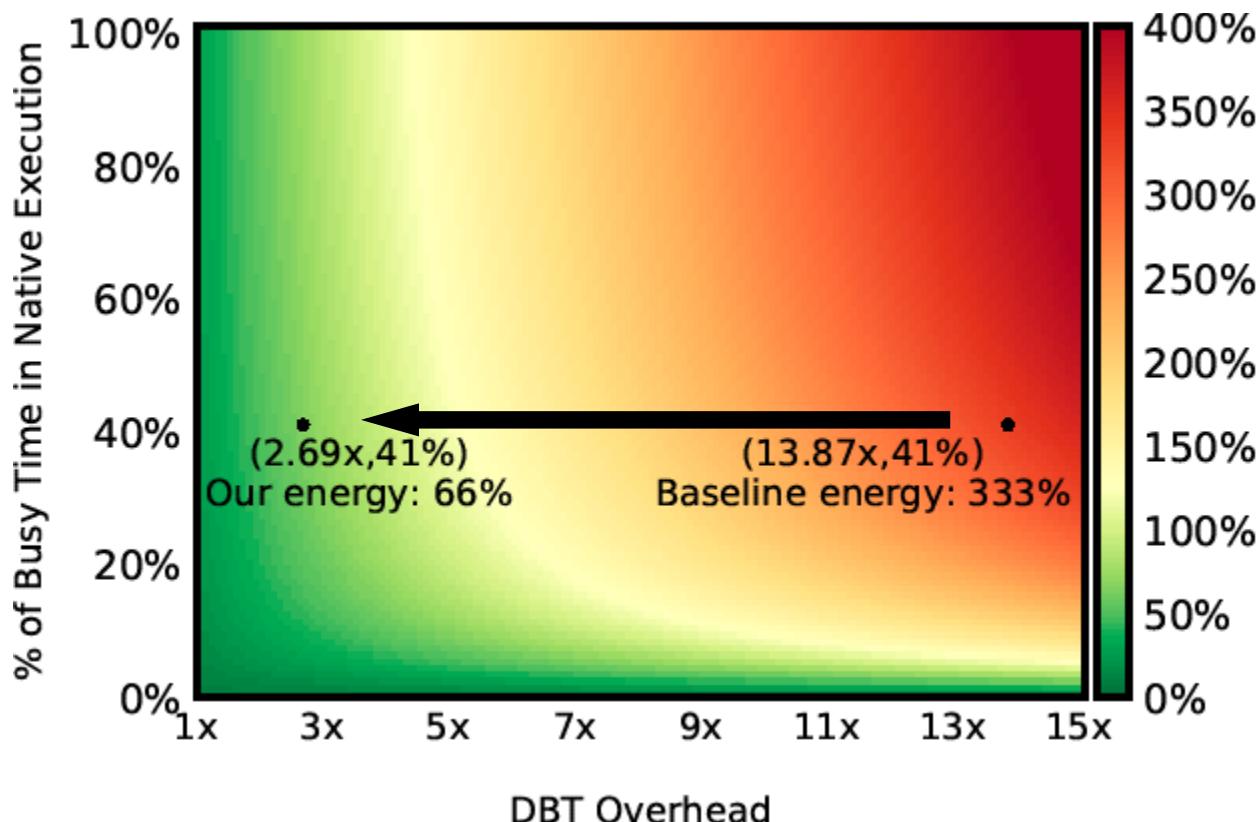
1. Translate most of driver code
2. Emulate the remaining kernel infrastructure

Essentially a tiny VM for drivers!

Is this practical?



Only through careful optimization!



Case 3: Transkernel: tiny VMs for device driver paths

First DBT engine running on a microcontroller-like core!

Demonstrated:

- One can use DBT for efficiency **gain**
- The ossified kernel can be tamed!

This talk: our exploration in three cases

- App-defined OSes
 - Case 1: Memory mgmt for stream analytics
 - Case 2: Storage for retro. video analytics
- The ossified Linux Kernel
 - Case 3: Transkernel

Reflection: a tale of two inquiries

- App-defined OS
 - Brave new world
 - Usefulness depends on the target apps
 - OS as a tool: serving AI and big data
- Kernel as the new firmware
 - Still lots of opportunities
 - Unique constraints. Require unconventional techniques
 - Like fixing an engine of an airplane in the air
 - OS as a craft. Fun!

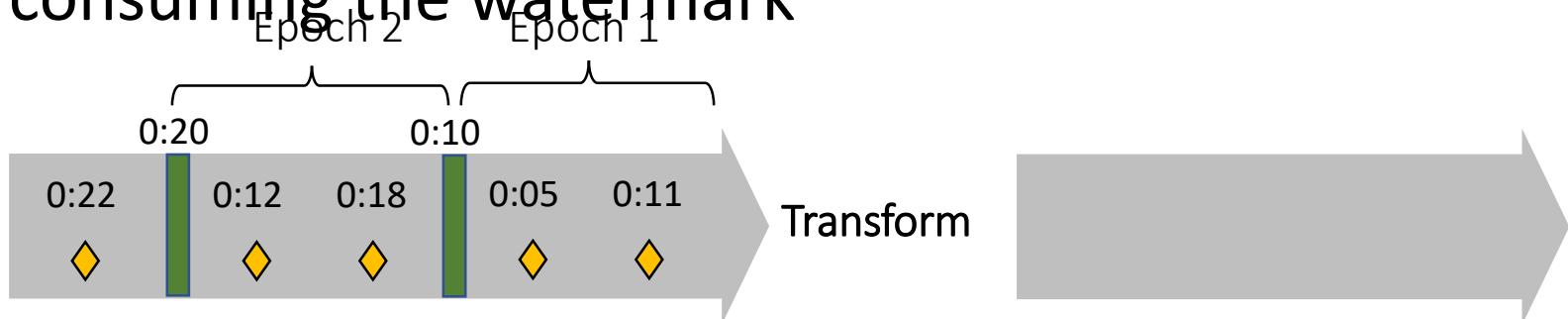
Reflection: a tale of two inquiries

- App-defined OS
 - OS as a tool: serving AI and big data
 - Large design space
 - Utility triumphs over other concerns
 - Staying relevance
- Kernel as the new firmware
 - OS as an art in its own right
 - Craft and beauty triumphs utility
 - A lot of fun
 - Privilege as academics

Invariant 1 Watermark ordering

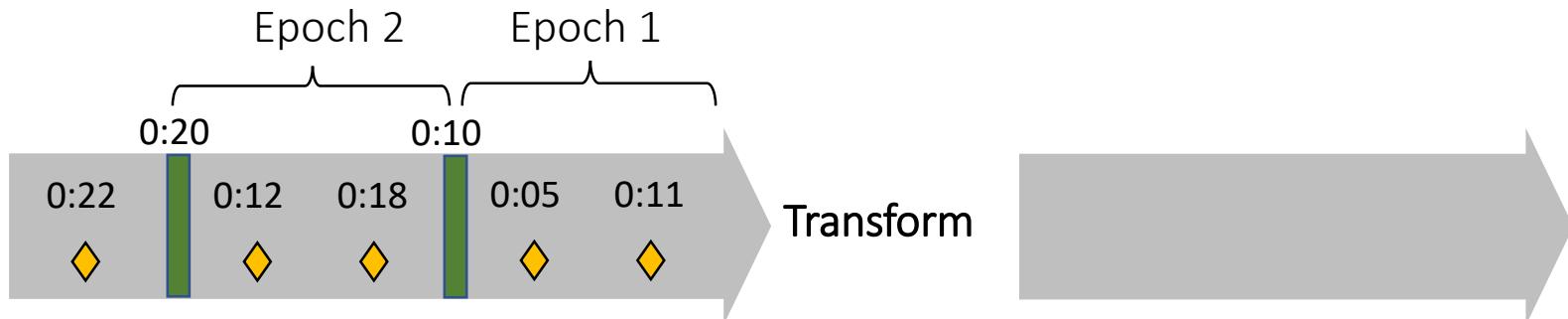
Transforms consume watermarks in order

Transforms consume all records in an epoch before consuming the watermark



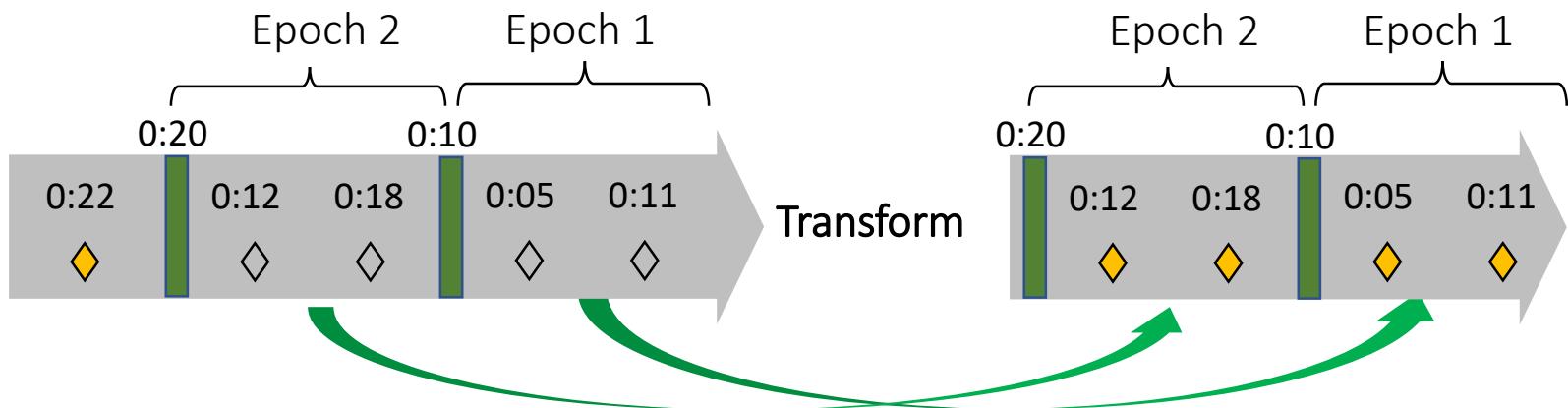
Invariant 2 Respect epoch boundaries

Once a transform assigns a record an epoch, the record never changes epochs



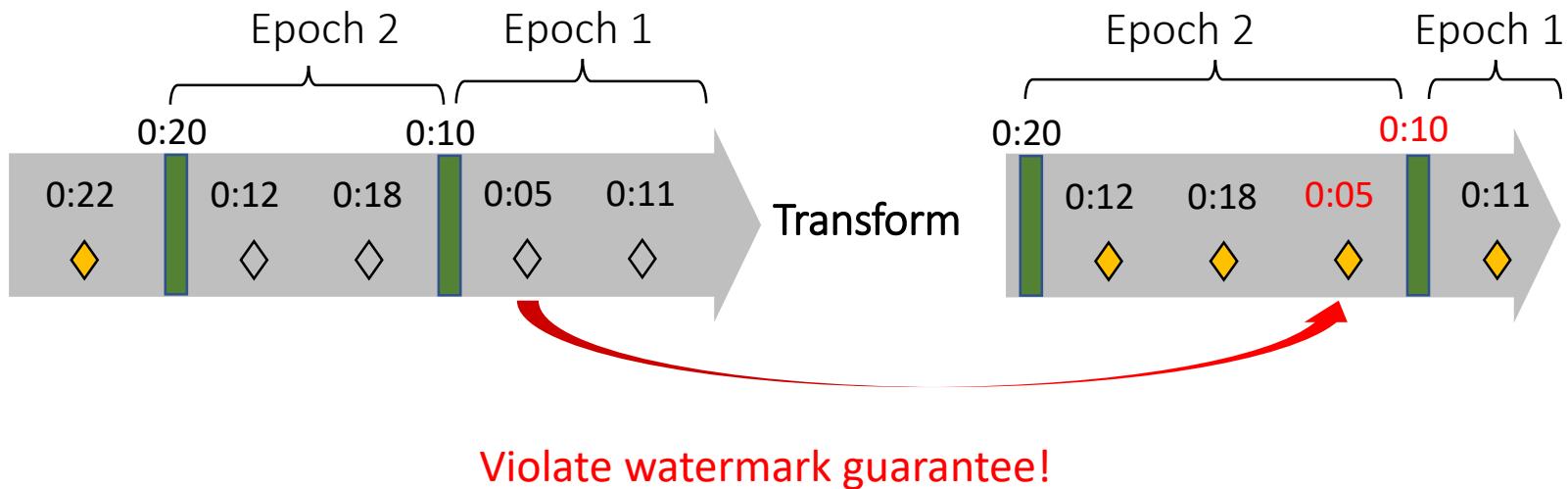
Invariant 2 Respect epoch boundaries

Once a transform assigns a record an epoch, the record never changes epochs



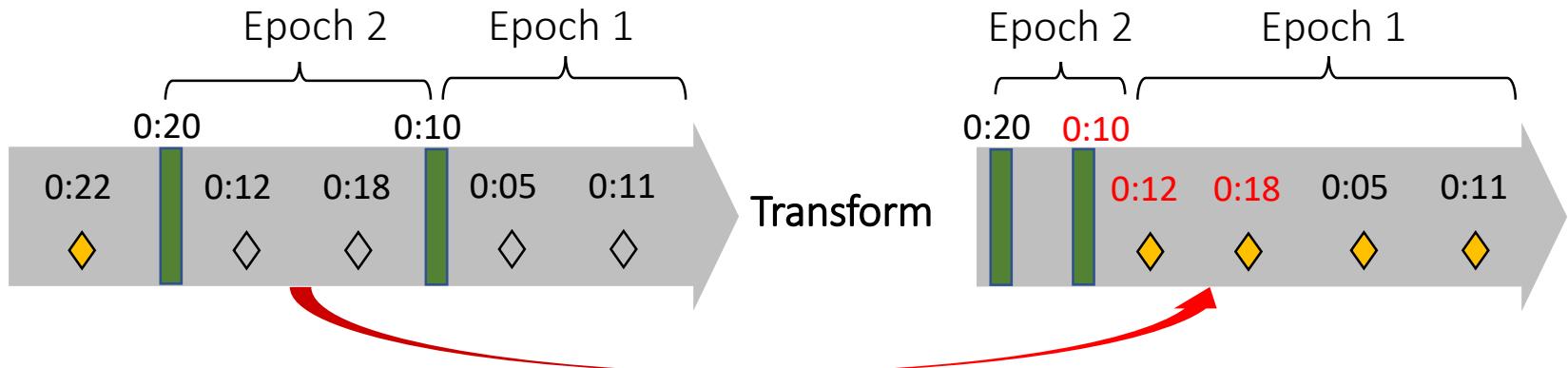
Invariant 2 Respect epoch boundaries

What if a record changes to a **later** epoch?



Invariant 2 Respect epoch boundaries

What if records change to an **earlier** epoch?



Relax watermark, and delay window completion!