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
AI + big data: the workloads in our era

How does OS respond?
OS as in textbooks

- Resource management
- Abstraction
- Protection
- Multiplexing
OS in 2000

OS in 2018

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

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

---

**Ingress** → **Window** → **Group by word** → **Count word occurrences** → **egress**

**WordCount**
Background: streaming pipeline

**Operators**  Computations consuming & producing streams

**Pipeline**  A graph of operators

- Ingress
- Window
- Group by word
- Count word occurrences
- TopK words
- Sentiment Scoring
- Join

Sentiment Detection
Hybrid high-bandwidth memory
Hybrid high-bandwidth memory

- Tradeoffs: capacity vs bandwidth
- Untraditional memory hierarchy
  - No latency benefit (Unlike SRAM+DRAM)
  - Configurable: sw-managed or hw-managed
Accelerate stream analytics with HBM?

• The most expensive stream operators: grouping

<table>
<thead>
<tr>
<th>Grouping in Hash</th>
<th>High bandwidth memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>High capacity demand</td>
<td>Capacity limited</td>
</tr>
<tr>
<td>Extensive random access</td>
<td>❤ Seq access + parallelism</td>
</tr>
</tbody>
</table>
HBM can accelerate grouping!

- ... with an unconventional algorithm choice
- Grouping: hash $\rightarrow$ 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?

![Diagram showing HBM capacity and DRAM bandwidth with demand levels]

- Low-demand for both
- High-demand for both
- Demands balanced
- More on DRAM
Memory allocator: balancing two limited resources

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

![Diagram showing the balance between HBM Capacity and DRAM Bandwidth with two demand levels for both resources.](22)
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 campuses run > 200 cameras 24x7
- A single camera produces 24 GB video per day
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

Frame diff detector Specialized neural net Full neural net

~10,000x ~1,000x ~10x

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

Image credits: NoScope: Optimizing Neural Network Queries over Video at Scale, Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, Matei Zaharia, VLDB 2017
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

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 😞

![Diagram showing the video data path and storage formats](image_url)

Video data path

Storage formats

Video consumers

- <op, accuracy>
- <motion, 0.95>
- <motion, 0.7>
- ...
- <OCR, 0.95>
- <OCR, 0.90>
- ...
- <NN, 0.95>
- <NN, 0.80>
Storing all needed formats?

Ingestion & storage are expensive 😞
Proposal: Store minimum necessary formats

Video data path

Storage formats

Consumption formats

Video consumers

<op, accuracy>

<motion, 0.95>

<motion, 0.7>

...

<OCR, 0.95>

<OCR, 0.90>

...

<NN, 0.95>

<NN, 0.80>
Proposal: Store minimum necessary formats

Key Challenge: Too many knobs to tune!
Proposal: Store minimum necessary formats & derive them automatically!

Backward derivation of configuration

Storage formats

Consumption formats

Video data path

Video consumers

<op, accuracy>

<motion, 0.95>

<motion, 0.7>

...

<OCR, 0.95>

<OCR, 0.90>

...

<NN, 0.95>

<NN, 0.80>
Benefit: faster analytics

Query speed (x realtime)

<table>
<thead>
<tr>
<th>Target Accuracy</th>
<th>1000x</th>
<th>100x</th>
<th>10x</th>
<th>1x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>0.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Benefit: faster analytics

Why?
- **Lower** accuracy $\rightarrow$ analytics tolerates cheaper video formats
- Video retrieval should be proportionally cheaper!
Benefit: faster analytics

Why?

- **Lower** accuracy $\rightarrow$ analytics tolerates cheaper video formats
- Video retrieval should be proportionally cheaper!
Case 2: First data store for retro. video analytics

缥 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!

1. Understanding Modern Device Drivers, Asim Kadav and Michael Swift. ASPLOS’12.
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...

Time
Sleep
Device Suspend

Freeze user&fs
User Task
Thaw user

Resume
Device
Wakeup
Suspend/Resume Is Slow

<table>
<thead>
<tr>
<th></th>
<th>Nexus 5</th>
<th>Gear</th>
<th>Note 4</th>
<th>Panda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspend</td>
<td>119 ms</td>
<td>191 ms</td>
<td>231 ms</td>
<td>262 ms</td>
</tr>
<tr>
<td>Resume</td>
<td>88 ms</td>
<td>159 ms</td>
<td>316 ms</td>
<td>492 ms</td>
</tr>
</tbody>
</table>

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

The workflows
The workflows

Current   Proposed

Wakeup
Device Resume
Thaw user
User Task
Freeze user&fs
Device Suspend
Sleep

Time

CPU   CPU   Tiny Core
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 Transkernel 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!

% of Busy Time in Native Execution

DBT Overhead

- Our energy: 66%
- Baseline energy: 333%

(2.69x, 41%)
(13.87x, 41%)
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!