The Aurora OS: Revisiting the Single Level Store

Emil Tsalapatis, Ryan Hancock, Tavian Barnes, Ali José Mashtizadeh

RCS Group @ University of Waterloo

HotOS ’21 – June 4, 2021

Persistence Is Hard

- Persistence is difficult to implement
- Subtle bugs persists even for mature systems
  - LevelDB (Chrome, Ethereum) has had multiple[1] [2] [3] [4]
- New apps rebuild persistence from scratch
  - Developers must move data around the storage hierarchy

[1] https://ethereum.stackexchange.com/questions/1159/corruption-on-data-block-while-synchronising

[2] https://github.com/google/leveldb/issues/333

[3] https://forum.syncthing.net/t/panic-leveldb-table-corruption-on-data-block/2526

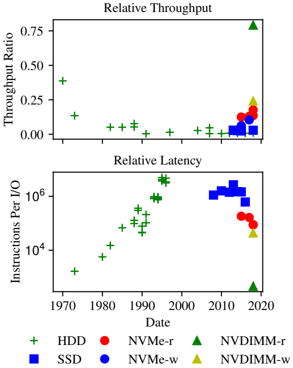
[4] https://bugs.chromium.org/p/chromium/issues/detail?id=261623

Single Level Stores (SLSes)

- Eliminate semantic gap between file IO and in-memory
  - No file IO, no data serialization
- SLS: Applications entirely in memory
- Applications oblivious to system crashes
  - No error handling by the app itself

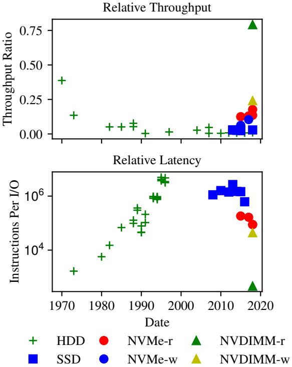
Re-enabling the Single Level Store

Fast Flash Devices

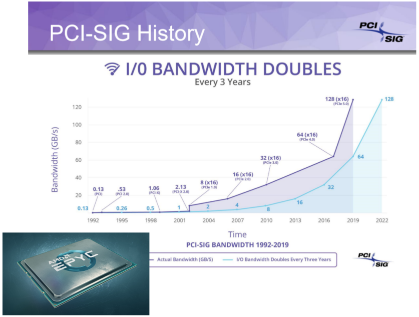


Re-enabling the Single Level Store

Fast Flash Devices

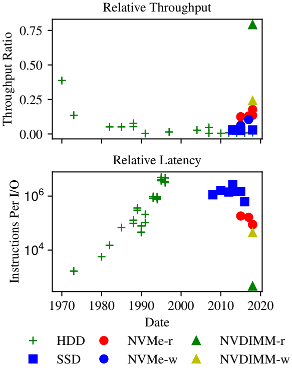


IO Bandwidth $\geq$ Memory

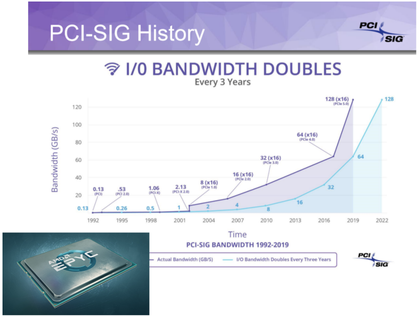


Re-enabling the Single Level Store

Fast Flash Devices



IO Bandwidth $\geq$ Memory



Larger Address Spaces

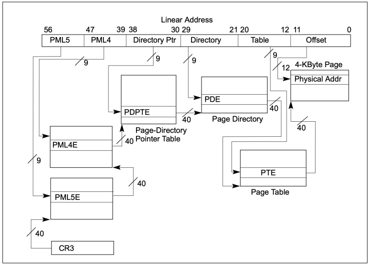
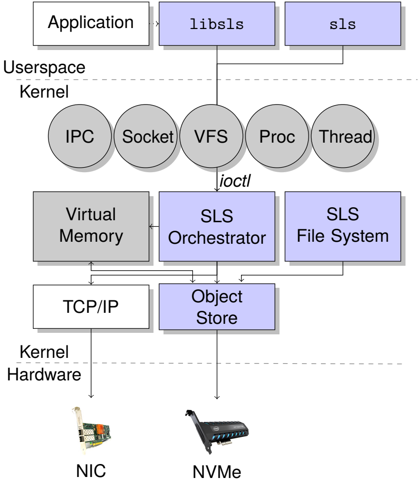


Figure 2-1. Linear-Address Translation Using 5-Level Paging

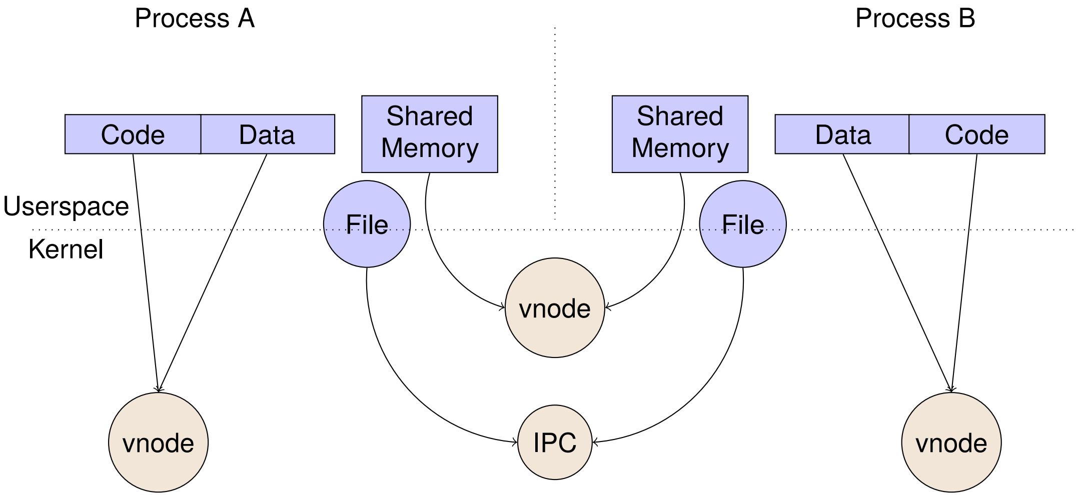
Demo

Aurora's Architecture

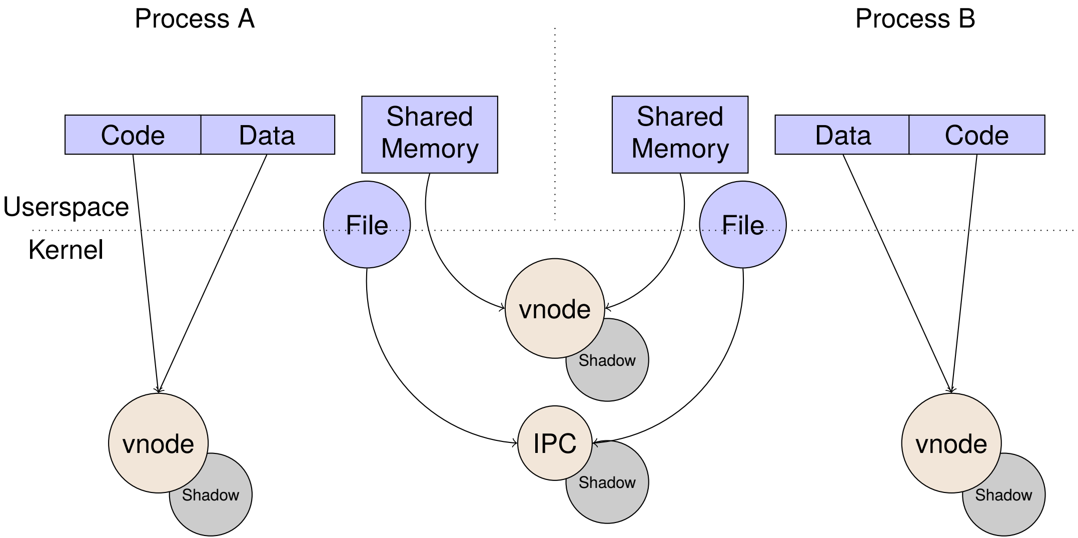
- SLS orchestrator gathers state from subsystems
- Bundles persistent objects into checkpoints
- Checkpoints held in high frequency COW store
  - Default frequency: 100 Hz
  - No NVDIMMs necessary!



Two Key Insights: POSIX Object Model

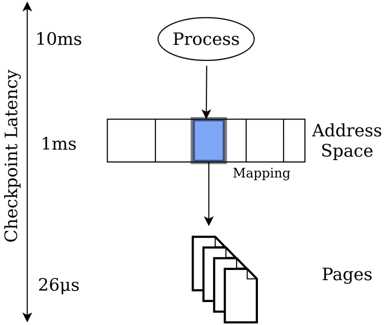


Two Key Insights: System Shadowing



Aurora for Developers

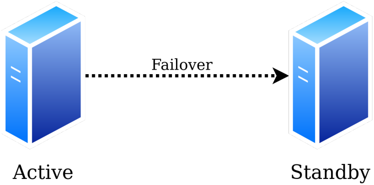
- Custom persistence schemes
  - General primitives with explicit guarantees
- Checkpoint an application, a region, a page...
  - Single page latency: 26 µs
- Custom restore handler for post crash fixups



Manipulating Execution State

Mobility and HA

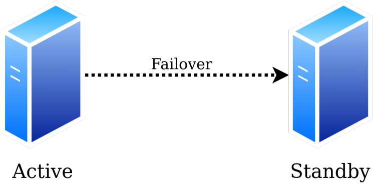
- Fault tolerance
- Application migration



Manipulating Execution State

Mobility and HA

- Fault tolerance
- Application migration



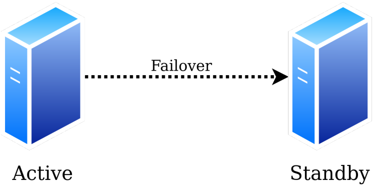
Debugging

- Time Travelling Debugging
- Optimizing record/replay

Manipulating Execution State

Mobility and HA

- Fault tolerance
- Application migration

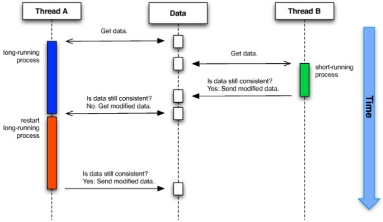


Debugging

- Time Travelling Debugging
- Optimizing record/replay

Speculative Execution

- Speculative Execution
- Rollbacks
- OS Transactions


Thread A
Data
Thread B
Get data.
long-running process
Get data.
short-running process
Is data still consistent? Yes: Send modified data.
Is data still consistent? No: Get modified data.
restart long-running process
Is data still consistent? Yes: Send modified data.
Time

Applications: Serverless Computing

- Cold starts dominate execution time
- Images partially overlap
  - High density in memory, on disk
- Shared data overlaps between images
  - Improves startup times

Conclusion

- The time has come for SLSes to make a comeback
  - Modern hardware makes transparent persistence possible
- We can - and should - offer persistence at the OS level
- Persistent processes are a flexible and powerful abstraction