The Ultima Thule* of a DBMS-backed Single-Host Operating System

Kostis Kaffes

*the ancient Greek name of Iceland symbolizing the absolute far-off land/unattainable goal

Revolutionary vs. Evolutionary DBOS

Apiary is a revolution:

- Clean-slate design
- Need to re-write and re-architect applications

Can there be an evolutionary version of DBOS?

- Backwards-compatible with most existing applications
  - Linux runs **90%** of the public cloud workload!
- Function as the kernel/bottom layer for Apiary

Question: How can we bring the DBOS principles to Linux?

Relational Linux APIs I

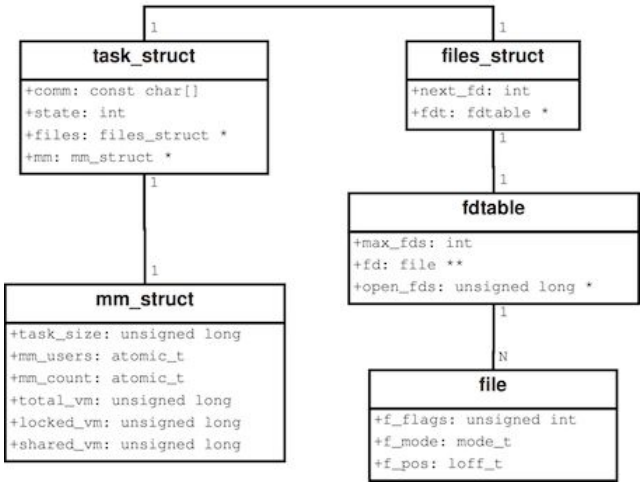
osquery

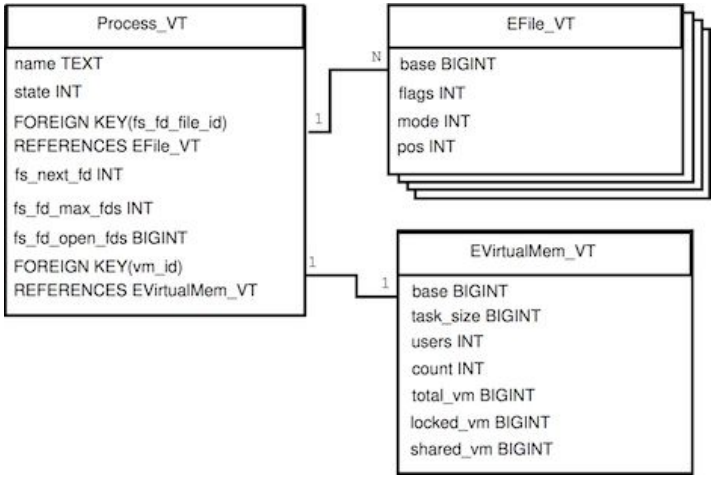
```
SELECT DISTINCT processes.name, listening_ports.port, processes.pid
  FROM listening_ports JOIN processes USING (pid)
  WHERE listening_ports.address = '0.0.0.0';
```

- Relational access to instantaneous information exposed by the kernel
- Read-only
- No historical data

Relational Linux APIs II

PicoQL (Eurosys 2014)





- Exposes kernel data structures as virtual tables
- Read-only and no historical data

Requirements

- Express fundamental OS operations that *alter* system state in short SQL statements, minimizing developer burden
- Capture and log historical state changes, making provenance trivial
- Backwards compatibility

Challenges

- Arcane OS design

What is the best way to integrate a database with the OS?

- **Option 1:** Run a DBMS in the kernel space

Pros:

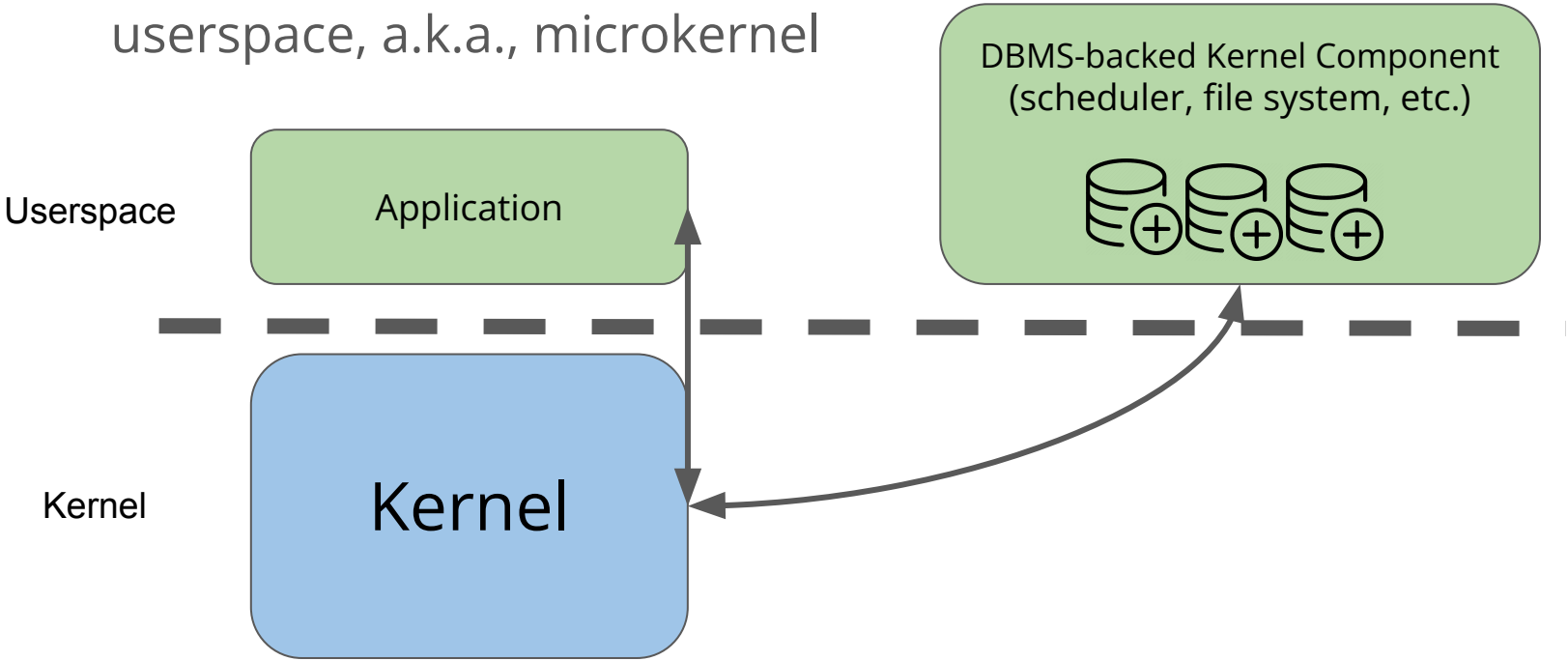
- One DBMS to fit all

Cons:

- One DBMS to fit all
- Kernel-space constraints (floating point, memory management)
- Probably need a new DBMS
- Invasive kernel changes

What is the best way to integrate a database with the OS?

- **Option 2:** Offload kernel functionality to agents running in userspace, a.k.a., microkernel



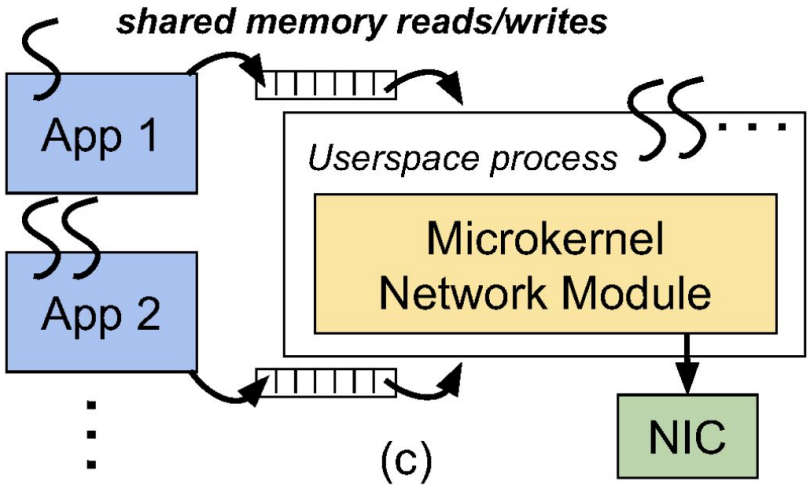
Microkernel DBMS

Pros:

- Gradual lifting of functionality out of the kernel without dealing with the kernel's complexity

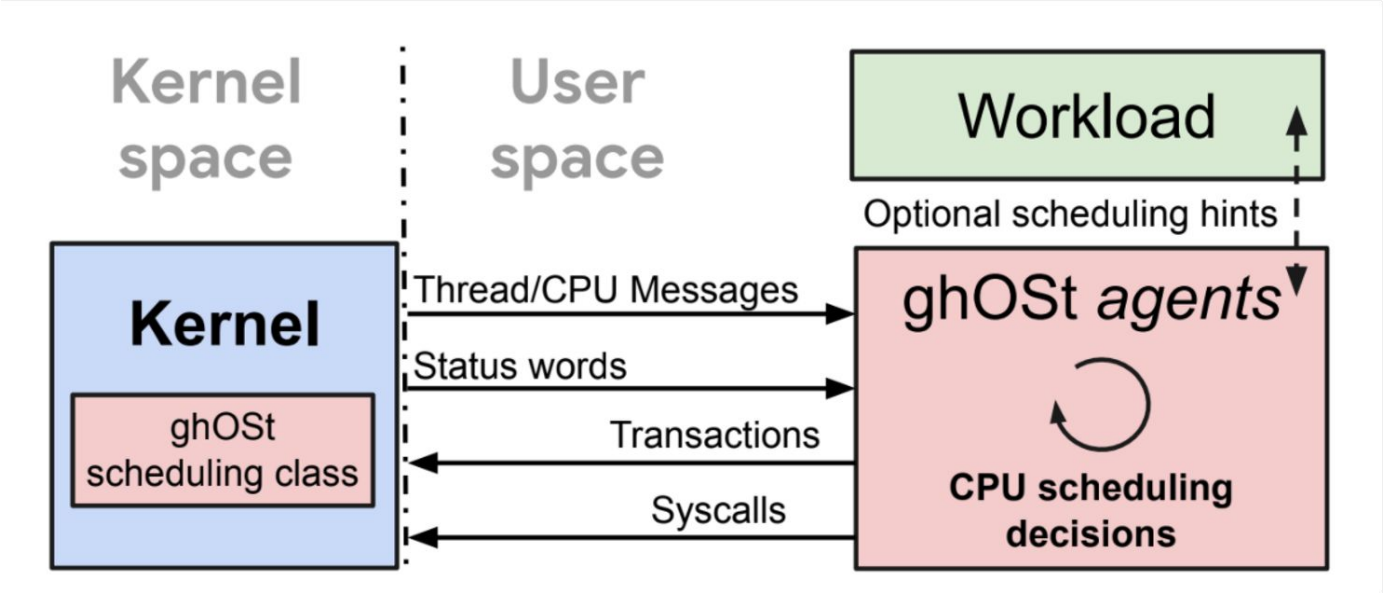
Microkernel Example 1: Networking

Snap [SOSP 2019]: Google's host **networking** stack running in userspace



Microkernel Example 2: Scheduler

ghOSt [SOSP 2021]**:** Kernel **scheduler** running in userspace



Microkernel DBMS

Pros:

- Gradual lifting of functionality out of the kernel without dealing with the kernel's complexity
- Can use any DBMS for any subcomponent without any constraints
- Agile development in userspace

Microkernel DBMS

Pros:

- Gradual lifting of functionality out of the kernel without dealing with the kernel's complexity
- Can use any DBMS for any subcomponent without any constraints
- Agile development in userspace

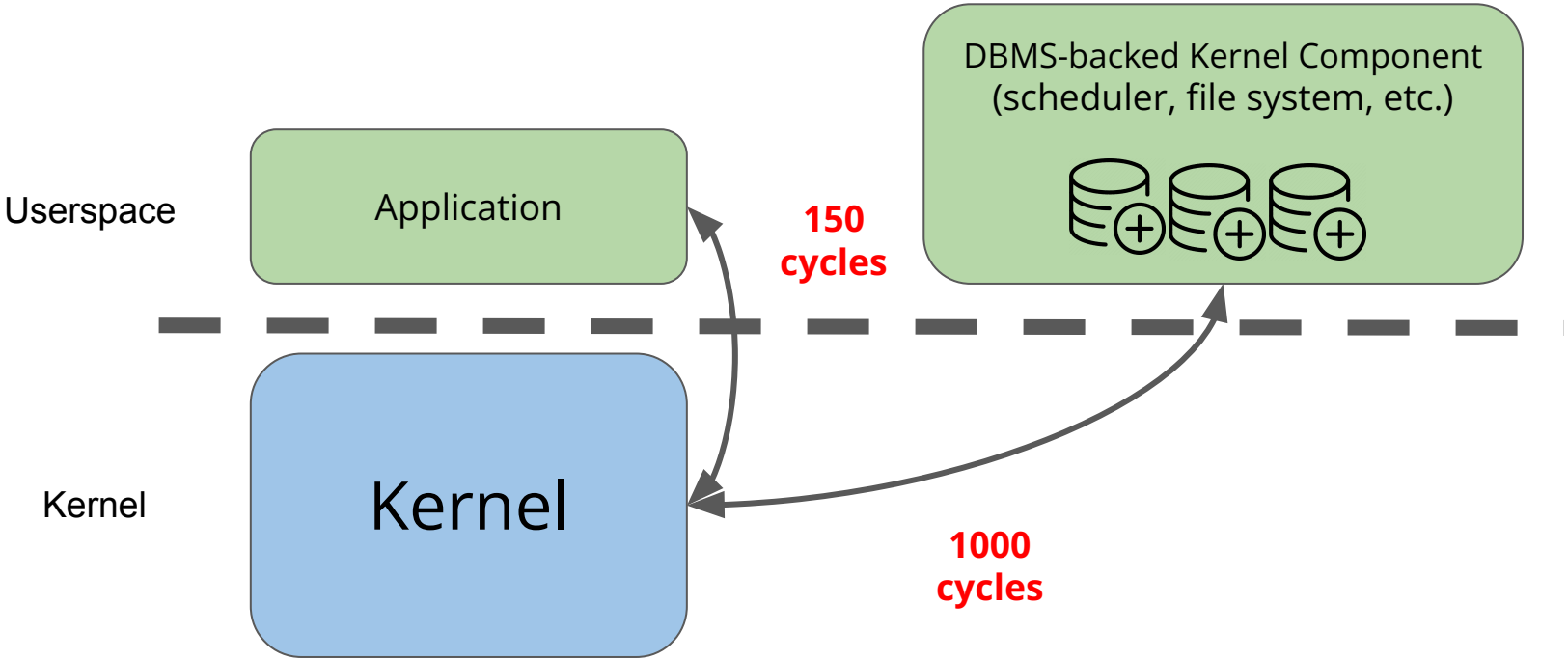
Question:

- Isn't delegating to userspace too slow?

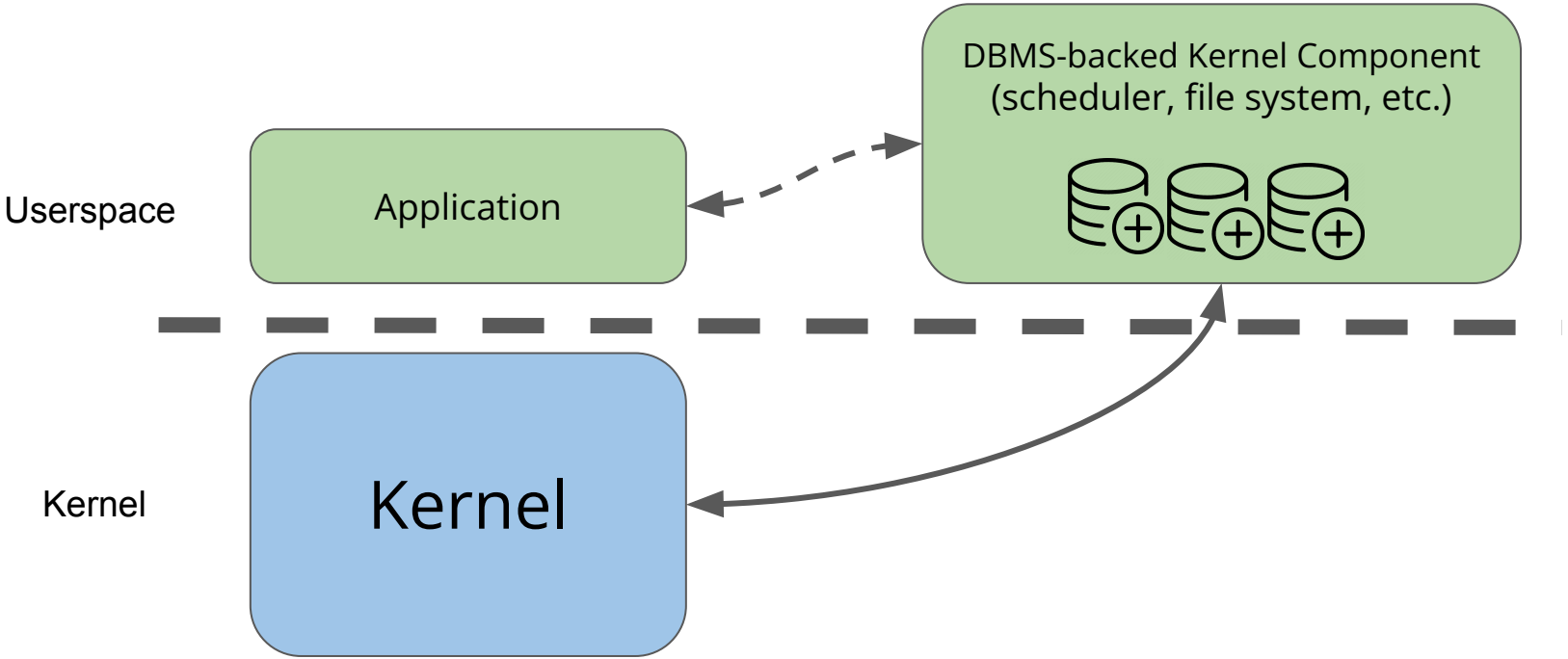
Isn’t delegating to userspace too slow? NO!!!!

- **Underbridge [ATC 20]:** 109 cycles RTT
- **Skybridge [Eurosys ‘19]:** 400 cycles RTT
- **ghOSt [SOSP ‘21]:** ~1000 cycles RTT

DBMS Microkernel Overheads



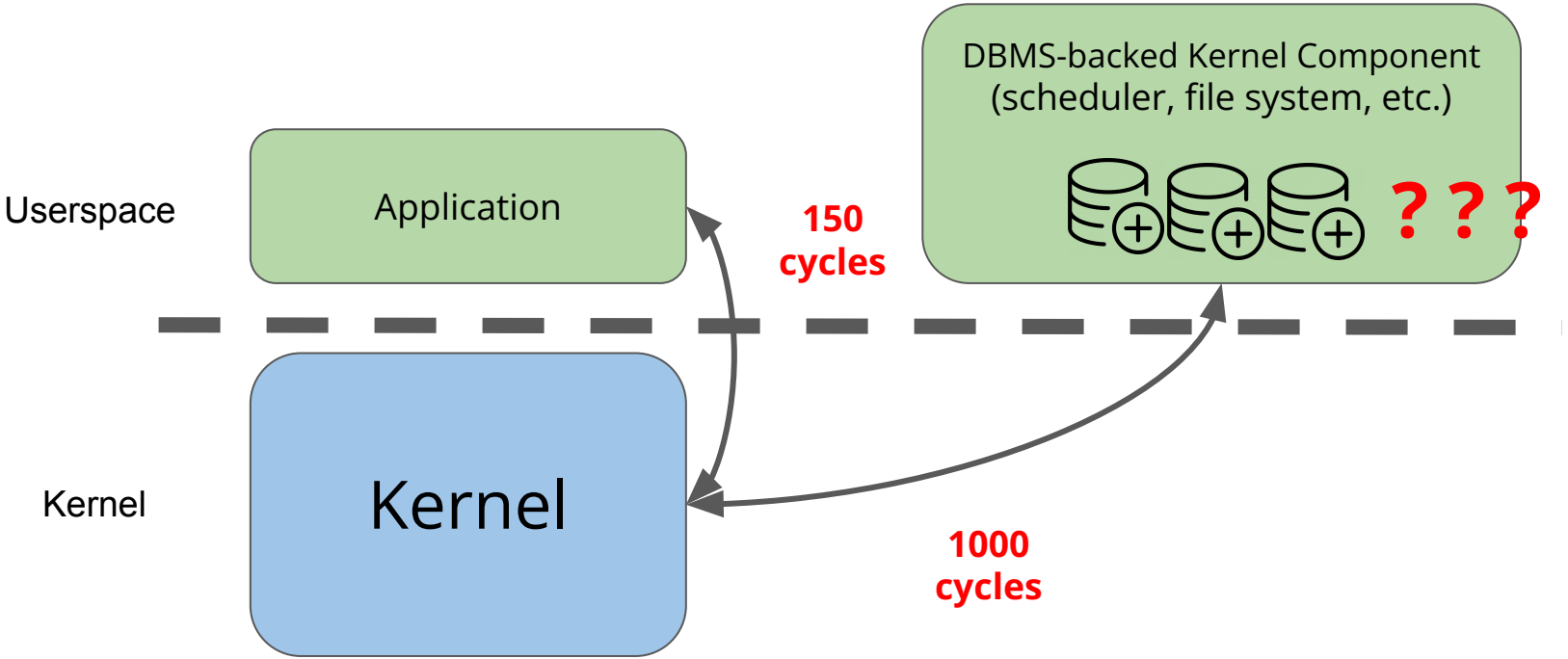
DBMS Microkernel Overheads



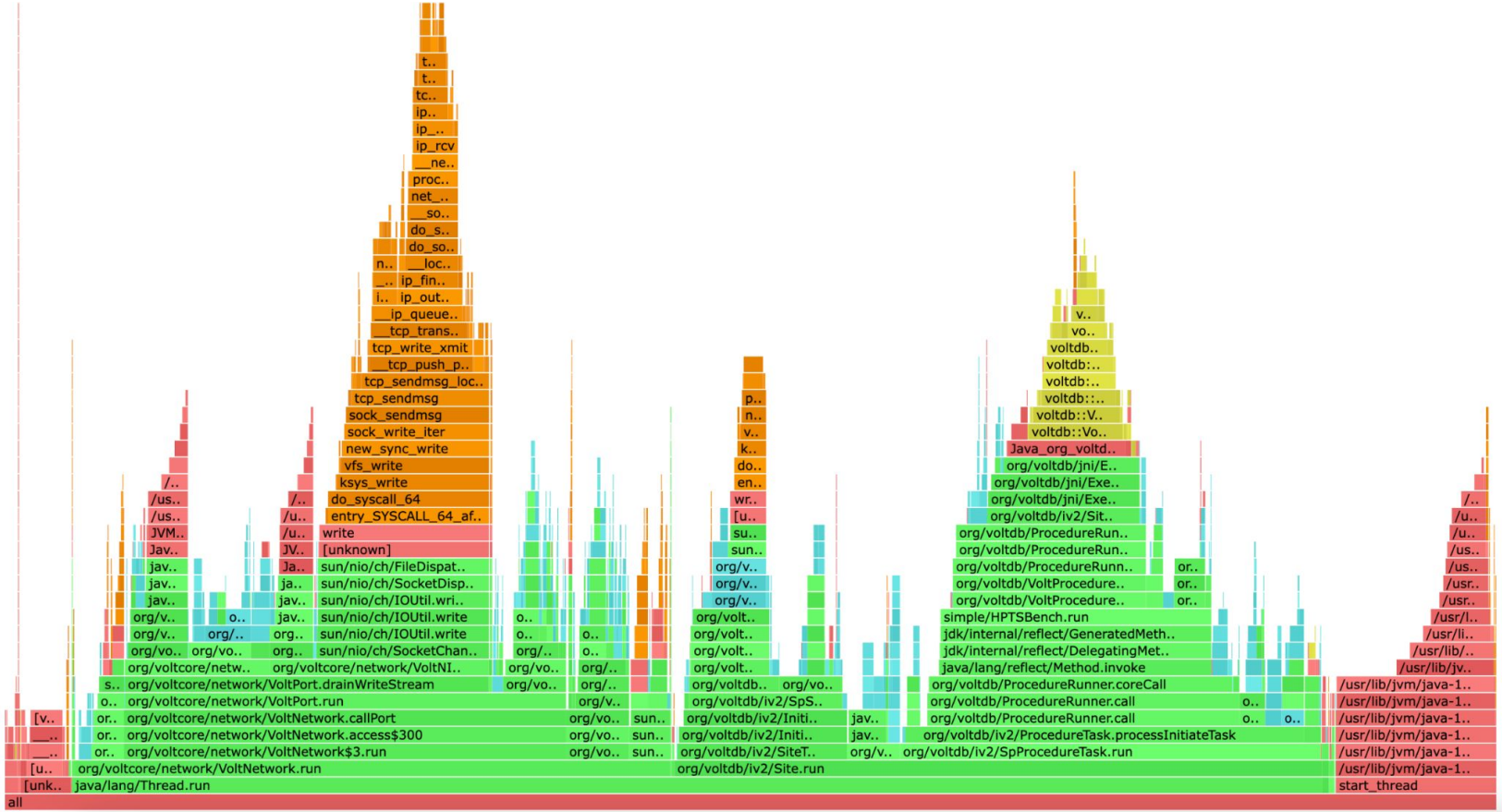
Challenges

- Arcane OS design 
- Database overhead

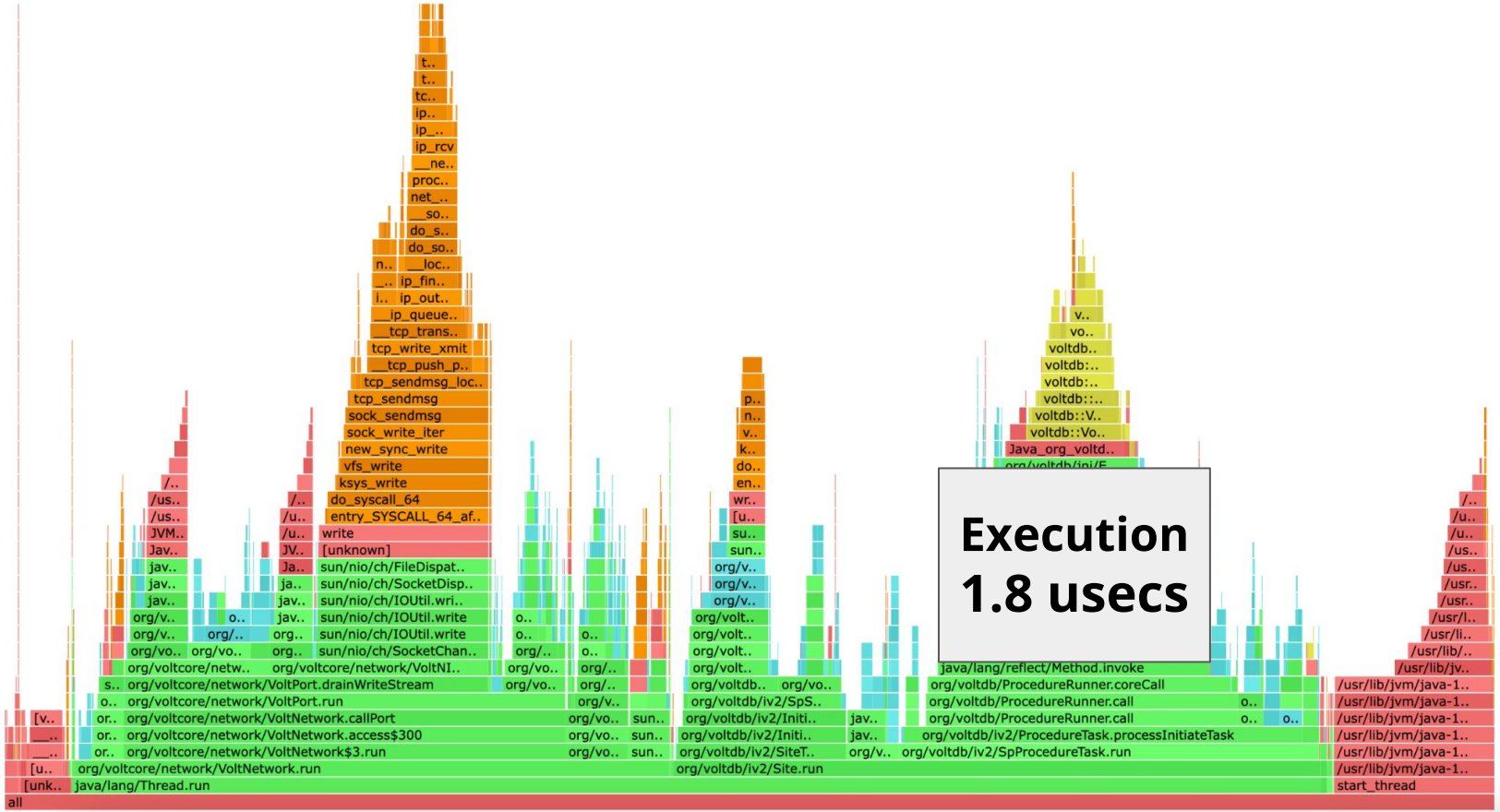
DBMS Microkernel Overheads



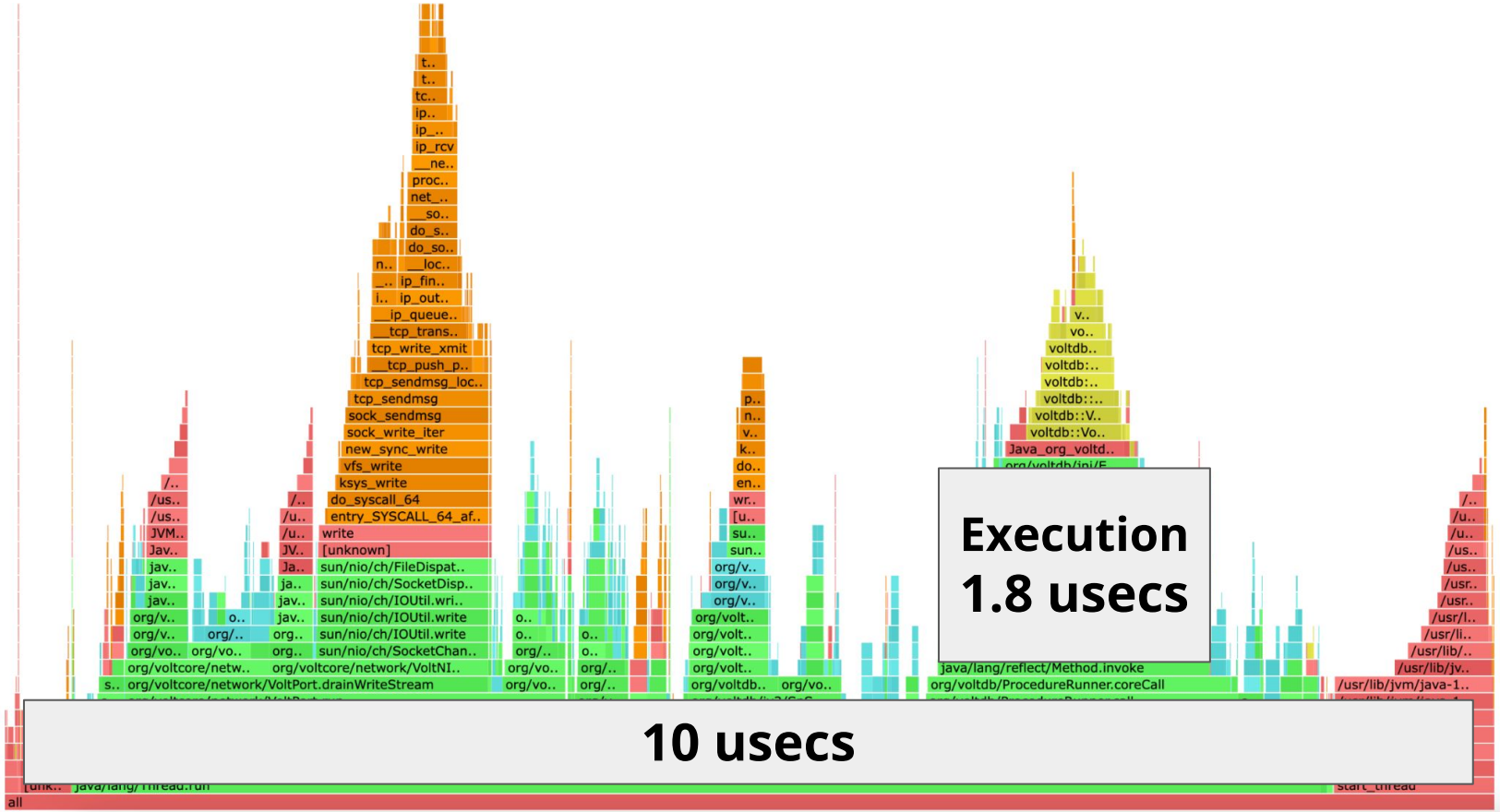
VoltDB Point Update Overhead



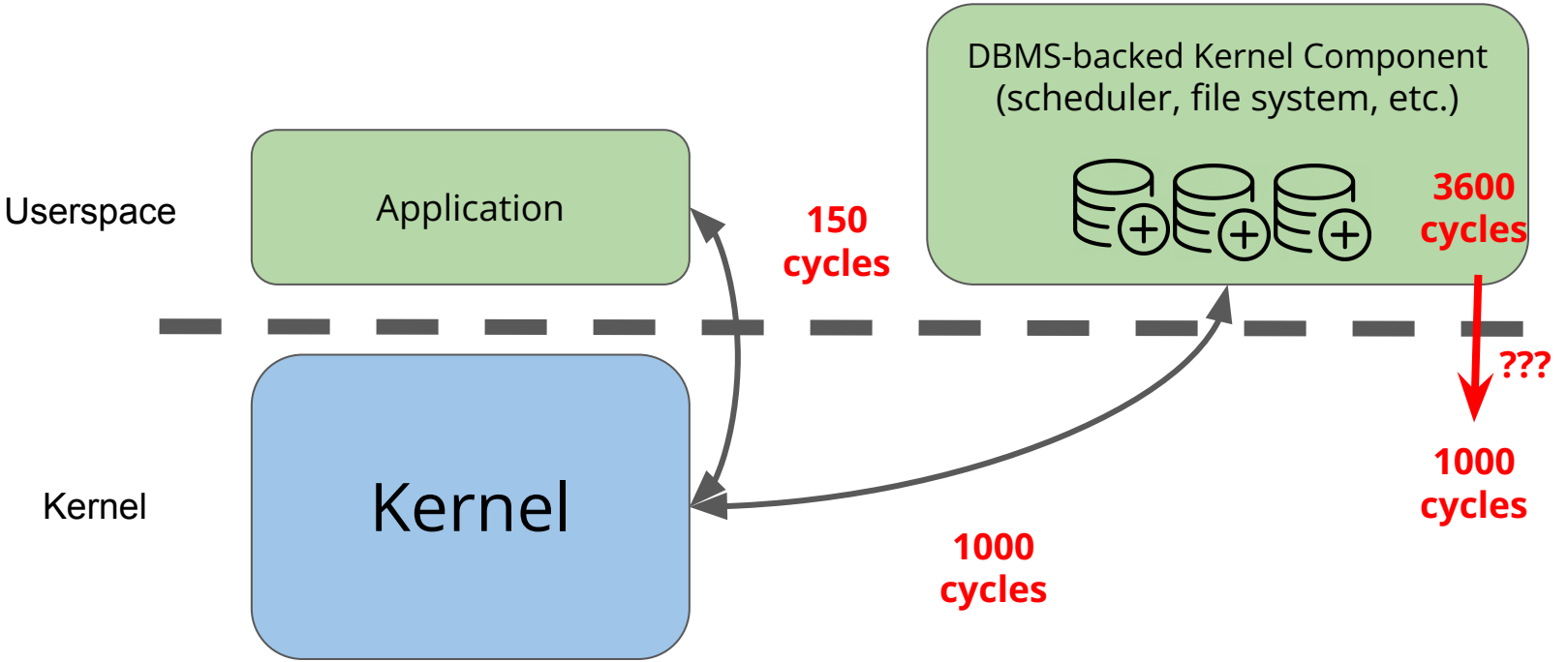
VoltDB Point Update Overhead



VoltDB Point Update Overhead



DBMS Microkernel Overheads



Challenges

- Arcane OS design 
- Database overhead