Data-Awareness

Konzepte der Betriebssystementwicklung

05.08.2020

Sven Leykauf

Friedrich-Alexander-Universität Erlangen/Nürnberg

Was ist Data Awareness?

Data Awareness - inoffizielle Definition

Data Awareness beschreibt das Bewusstsein über Form und Bewegung von Daten in Betriebssystemen.

- Welche Art von Daten handhabt das Betriebssystem?
- Wie sieht der Transport aus?
- Welche Rückschlüsse lassen sich aus diesen ziehen?

Hier: Data Awareness über IO-Daten!

CPU vs IO

Inhaltsverzeichnis

Problem Legacy Software

Kontextwechsel

- Leistungsdämpfung durch Kontextwechsel
  - $\Rightarrow$ Systemaufrufe
  - $\Rightarrow$ Interrupts
  - $\Rightarrow$ Andere Prozesse
- Zeitverluste durch Cache-Verschmutzung
- Verminderung in Anzahl - IO-Operation-Merging

Lange Pfade

- Das Durchqueren vieler Softwareschichten nimmt viel Zeit in Kauf
- Trifft auf die verbreitesten Betriebssysteme zu
- Grund liegt oftmals im Wunsch möglichst generisch zu bleiben

Kopierarbeit

- Kopieren zwischen den Hierarchieebenen kostet viel Leistung
- Oft aus Gründen des Pufferns bspw. Page-Cache
- Kopieren oft unnötig, da Daten so wie sie sind genutzt werden oder sowieso verworfen werden
- Es gibt Möglichkeiten Kopieren zu vermeiden
  - $\Rightarrow$ Spezielle Systemaufrufe wie splice(), mmap()
  - $\Rightarrow$ Parameter O_DIRECT bei open()-syscall

Near Data Processing

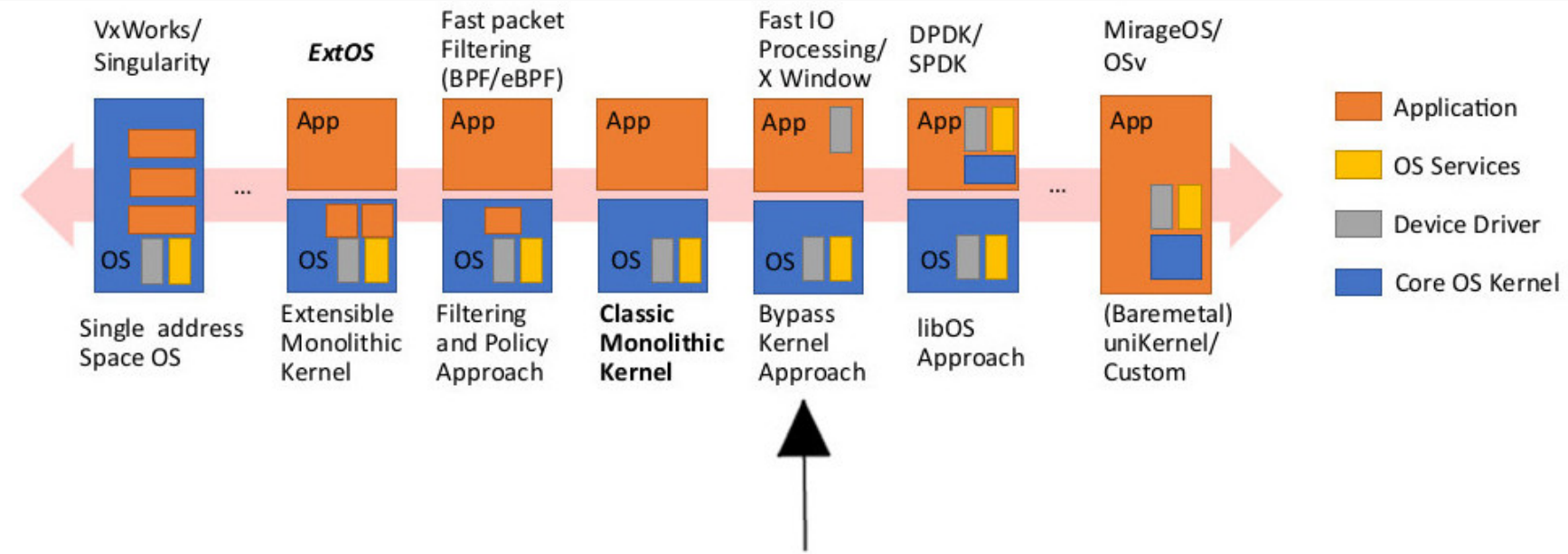
- Spezielle Rechnerarchitekturen, die versuchen die Daten so nah wie möglich an die Recheneinheiten zu bringen
- Teils sehr exotisch
- Oft Nutzung von heterogener Hardware

Bypass Kernel

Allgemeines Konzept

- Vermeidung des Kernels durch eigene Implementierungen im User-Space
- Zugriff auf Geräte muss im User-Space gewährleistet werden
  - $\Rightarrow$ Unter Zuhilfenahme einer IOMMU
  - $\Rightarrow$ Mittels Virtualisierung
- Dadurch lassen sich Großteile der Legacy Probleme entschärfen

Monolithic Kernel OS Design Space - Bypass Kernel



Bedeutung für die Anwendung

- Eigene Implementierung zur Hardwaresteuerung nun notwendig
- Polling kann jetzt vorteilhaft sein
- User-Level-Thread-Scheduling
- Zusätzlicher Schutz durch Treiber-Auslagerung
- Arbeiten auf gemeinsamen Daten jedoch erschwert

Ansätze unter Linux

UIO - User Space I/O

Schnittstelle, welche mmap() für Gerätezugriffe nutzt. Wird vor allem in VMs verwendet, da sie selbst nicht die IOMMU-Schnittstelle nutzt.

VFIO - Virtual Function I/O

Schnittstelle, die IOMMU-Schnittstelle nutzt und somit auch DMA verwenden darf.

DPDK & SPDK

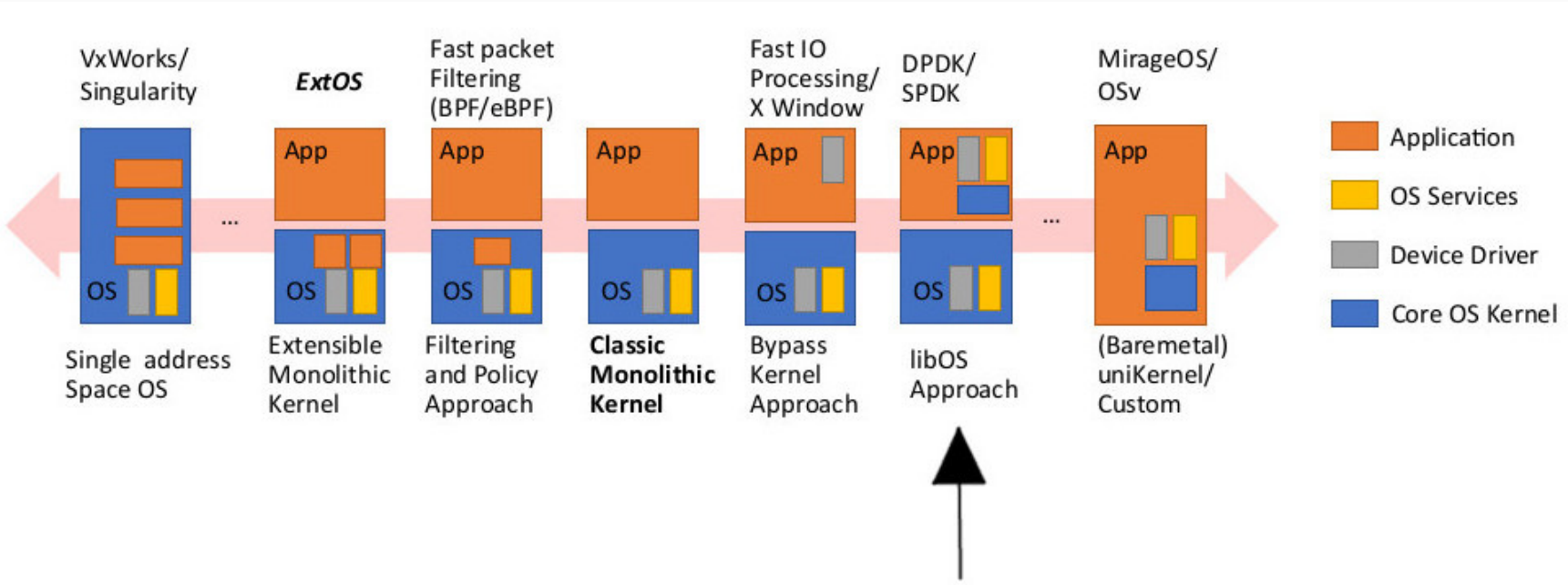
Data Plane Development Kit

Framework zur Entwicklung von Netzwerk Bypass-Kernel-Anwendungen.

Storage Performance Development Kit

Framework zur Entwicklung von Speicher Bypass-Kernel-Anwendungen.

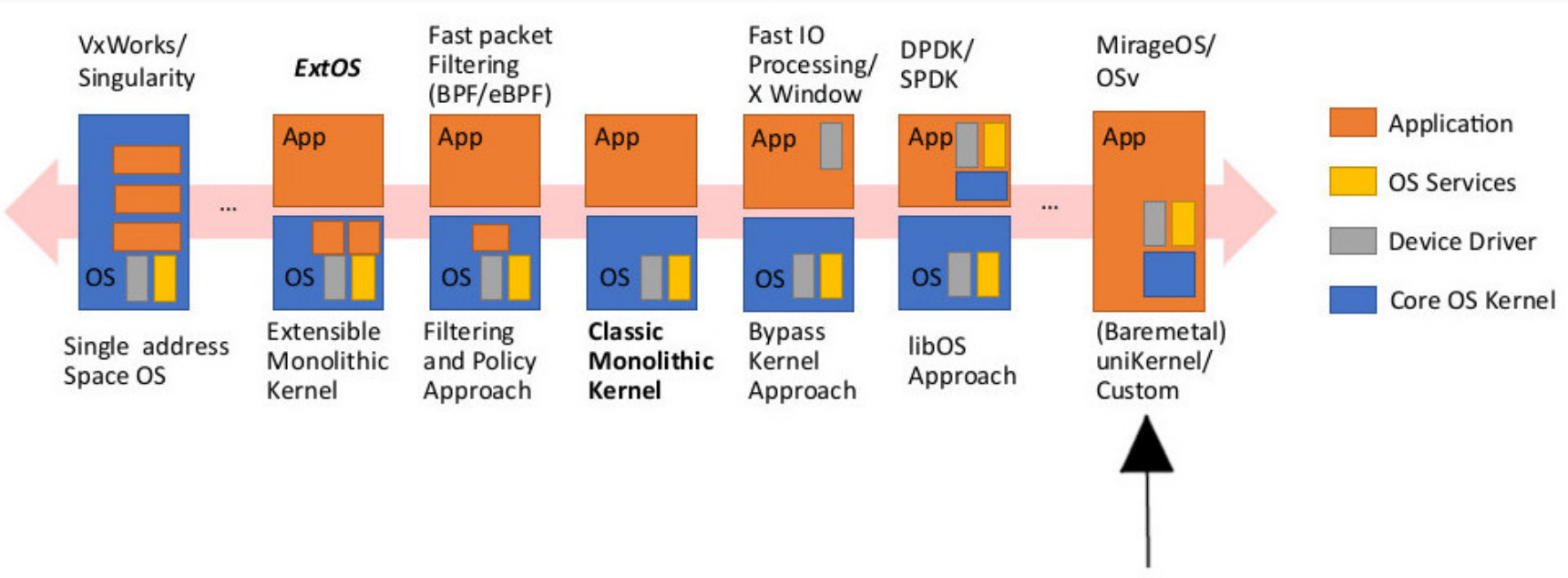
Monolithic Kernel OS Design Space - DPDK & SPDK



Exokernel/Tailored-OS

- Steigerung des Bypass Kernel Konzepts führen zu Exokernel/Tailored-OS
- Nur grundlegendste Funktionalitäten werden vom Kernel erledigt
- Ermöglichen das Entwickeln von hochoptimierten Anwendungen
- So gut wie alles muss jedoch selbst implementiert werden

Monolithic Kernel OS Design Space - Tailored OS



Extensible Monolithic OS

Allgemeine Idee

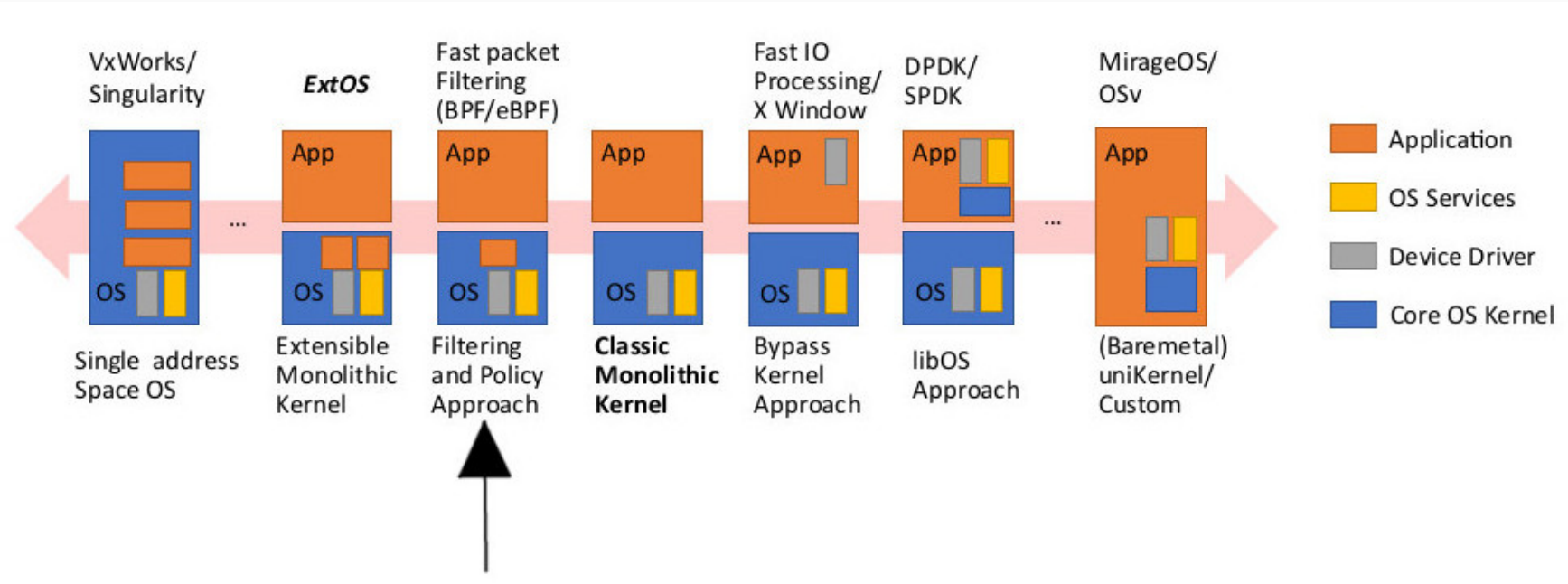
- Es soll Code der Anwendung in den Kernel eingespielt werden
- Hilft der Vermeidung von Kopierarbeiten
  - $\Rightarrow$ Anwendung kann dem Kernel sagen wie dieser mit den Daten umzugehen hat
  - $\Rightarrow$ Weiterreichen in den User-Space entfällt dadurch für einfache Operationen
- Möglichkeiten zur Optimierung, da System über Anwendungen bescheid weiß

BPF/eBPF

(extended)Berkley Packet Filter

- Virtuelle Maschine mit eigener ISA
- Ermöglicht das Einspielen von Anwendungscode
- Dient als Packet-Filter des Netzwerkstacks

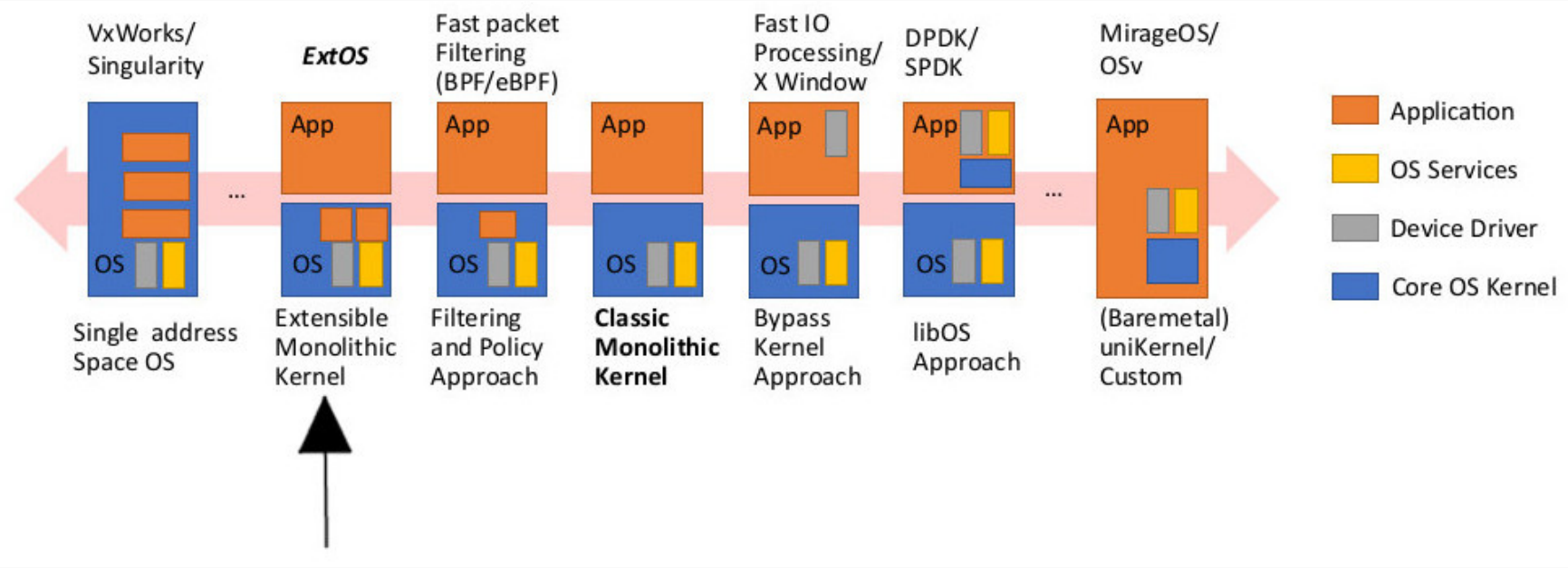
Monolithic Kernel OS Design Space - BPF/eBPF



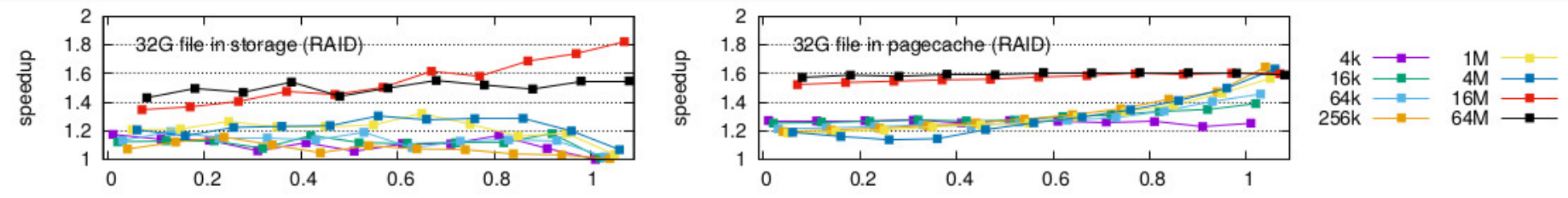
ExtOS

- Ein auf Linux basiertes OS welches BPF/eBPF nutzt und erweitert
- Versucht Erweiterbarkeit auf alle IO-Subsysteme auszuweiten - Fileoperations
- Granularität der Erweiterung auf dynamischem Level bis runter auf einzelne Prozesse
- Implementierung eines Privilegiensystems

Monolithic Kernel OS Design Space - ExtOS



Leistungssteigerung in ExtOS



Experimente zeigen, dass mittels der von ExtOS angewandten Erweiterungen des Kernels ein positiver Speedup erreicht wird.

Kontrolle des fremden Codes

- Fremdcode, der im Kernel ausgeführt wird, birgt große Gefahr für das System
- Wird JIT-compiliert
- Es benötigt Möglichkeiten diese zu beherrschen
  - $\Rightarrow$ Unterstützng für einspielbaren Code nur für bestimmte Programmiersprachen
  - $\Rightarrow$ Dadurch Möglichkeiten der Verifikation
  - $\Rightarrow$ Überprüfung dynamisch oder statisch
  - $\Rightarrow$ Software Fault Isolation (SFI)

Fazit

- Heutige Betriebssysteme kränkeln zunehmend an alter Legacy-Software
- Es lässt sich zeigen, dass durch Konzepte der Data-Awareness Leistungsdefiziete zwischen CPU und IO maßgeblich verbessert werden
- Da das Problem eher zunehmen als abnehmen wird, lohnt es sich in diese Konzepte Forschung zu investieren

Fragen?