Every cloud has a silver lining and
What we can learn from it

Jakub Jermář
Introduction

Who is Jakub

HelenOS developer since 2001
Solaris kernel engineer between 2006 and 2011
Software engineer at AVAST since 2011

Am I in a wrong devroom? This talk's got to be about clouds...

No way!
It's mostly about learning from mistakes

Microkernels and Component-based OS devroom, FOSDEM 2013
Where mistakes were made

Memory management
- kernel identity mappings only
- over-synchronized page tables

Device drivers
- in-kernel little brother drivers
- platform-specific drivers
- ad-hoc drivers

IPC
- all messages must be answered
Where mistakes were made

Memory management
  kernel identity mappings only
  over-synchronized page tables

Device drivers
  in-kernel little brother drivers
  platform-specific drivers
  ad-hoc drivers

IPC
  all messages must be answered
Kernel identity mappings only

RAM sizes in 32-bit computers around 2005 did not exceed 2G
Legacy or virtual devices only

Map physical memory to virtual 2G identically & life will be great

\[
PA2KA(\text{addr}) = \text{addr} + 2G \\
KA2PA(\text{addr}) = \text{addr} - 2G
\]

! Device registers in high physical memory
! 32-bit systems started to have more than 2G RAM
Kernel identity mappings only

Microkernels and Component-based OS devroom, FOSDEM 2013
Kernel identity mappings only

- Using locked TLB entries
- Waste a physical frame and use its virtual address
- Limit RAM to 2G – `const`, use the residue for non-identity

Road block for full-fledged userspace drivers
Need to access IO registers from IRQ pseudocode (in-kernel)

Real solution merged on Dec 31\textsuperscript{st}, 2011
Kernel identity / kernel non-identity split, VA allocator
Kernel identity mappings only

Microkernels and Component-based OS devroom, FOSDEM 2013
Kernel identity mappings only

Keep the Moore's law in mind early during the design phase. Do not place arbitrary integer limits on resources carelessly. Consider also the real-life scenarios.
In-kernel little brother drivers

Early versions of HelenOS featured PoC userspace drivers
Mostly counterparts of in-kernel input / output drivers

Let the kernel driver do all the setup & life will be great
device enumeration, device resources, parea whitelisting,
IRQ enabling, IRQ dispatching

! People want to be able to write purely userspace drivers
In-kernel little brother drivers

Kernel

Userspace

kbd

fb

e1000

ohci hc
In-kernel little brother drivers

device enumeration
device resources
handled by the userspace DDF

\textit{parea} whitelisting

concept abandoned for drivers

IRQ enabling

userspace interrupt controller drivers (still not part of DDF)

IRQ dispatching

empower IRQ pseudocode to claim the interrupt

drivers use physical addresses for IO registers
In-kernel little brother drivers

Avoid dependencies of production code on debug features. Kernel should preferably keep out of any device driver business. Do not put the kernel in charge of purely userspace namespace.
Over-synchronized page tables

HelenOS sometimes needs to walk the page tables
Page tables is a shared data structure

When walking the page tables, take a lock & life will be great

!? Hardware vs. software walked page tables
!? 4-level page tables vs. page hash table

! Problems surfaced with the advent of real userspace drivers
Over-synchronized page tables
Over-synchronized page tables

Discrepancy between hardware and software walked PTs
Assertions hit when PT mutex locked in the interrupt context

The fix involved transition to lock-free PT search
Search vs. Insert vs. Remove

Easy with 4-level page tables
Dependence on list implementation in case of page hash table
WiP: GSoC project to deliver scalable resizing CHT
Over-synchronized page tables

The variety of supported architectures can give hints. Kernel assertions can hint on new unexpected problems. Too much synchronization spoils the kernel.
Conclusion

Keep the Moore's law in mind early during the design phase
Do not place arbitrary integer limits on resources carelessly
Consider also the real-life scenarios
Avoid dependencies of production code on debug features
Kernel should preferably keep out of any device driver business
Do not put the kernel in charge of purely userspace namespace
The variety of supported architectures can give hints
Kernel assertions can hint on new unexpected problems
Too much synchronization spoils the kernel
Thank you!

http://www.helenos.org