Advanced Operating Systems

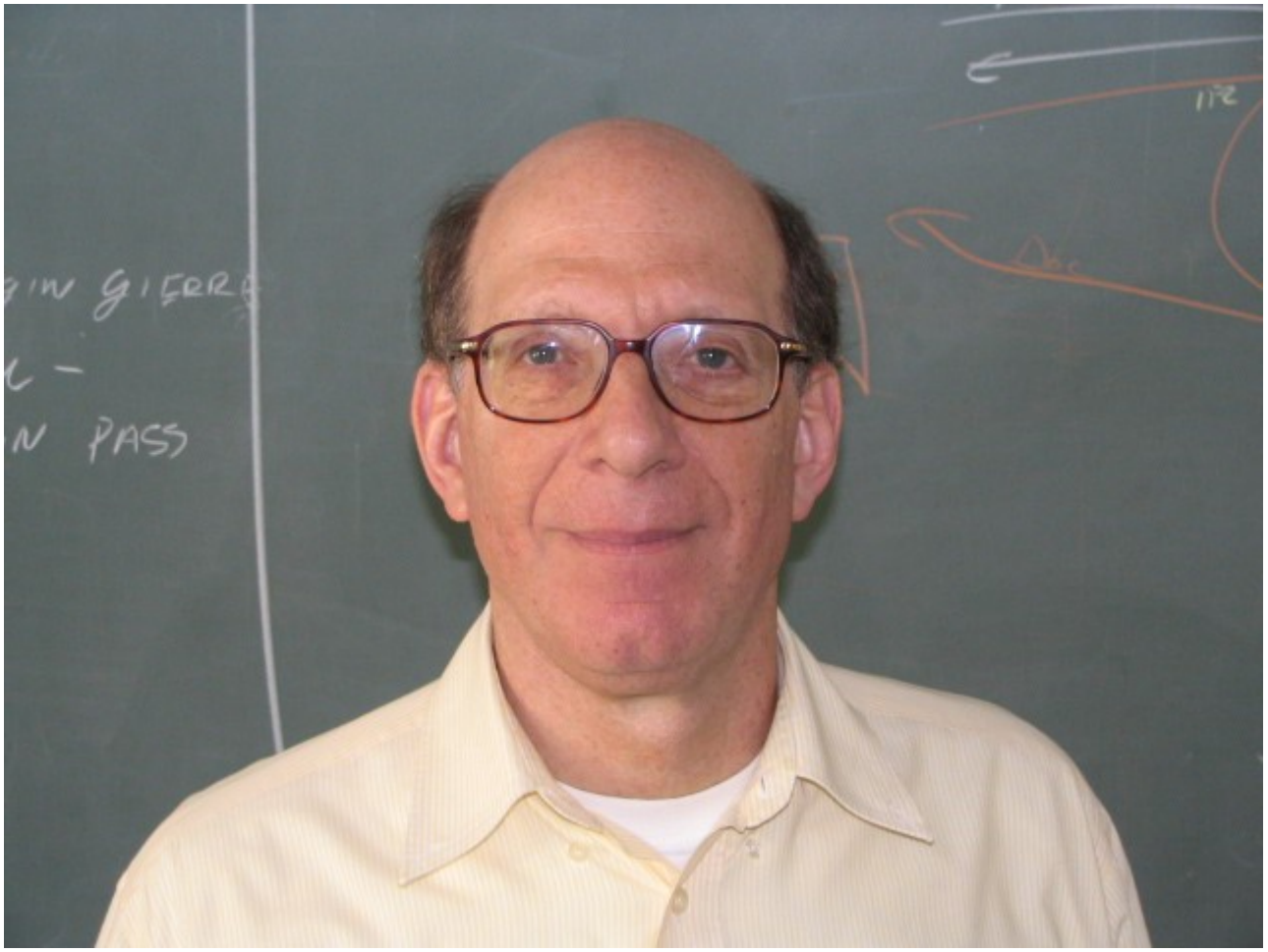
- lecture series introduction -

Petr Tůma

Department of Distributed and Dependable Systems

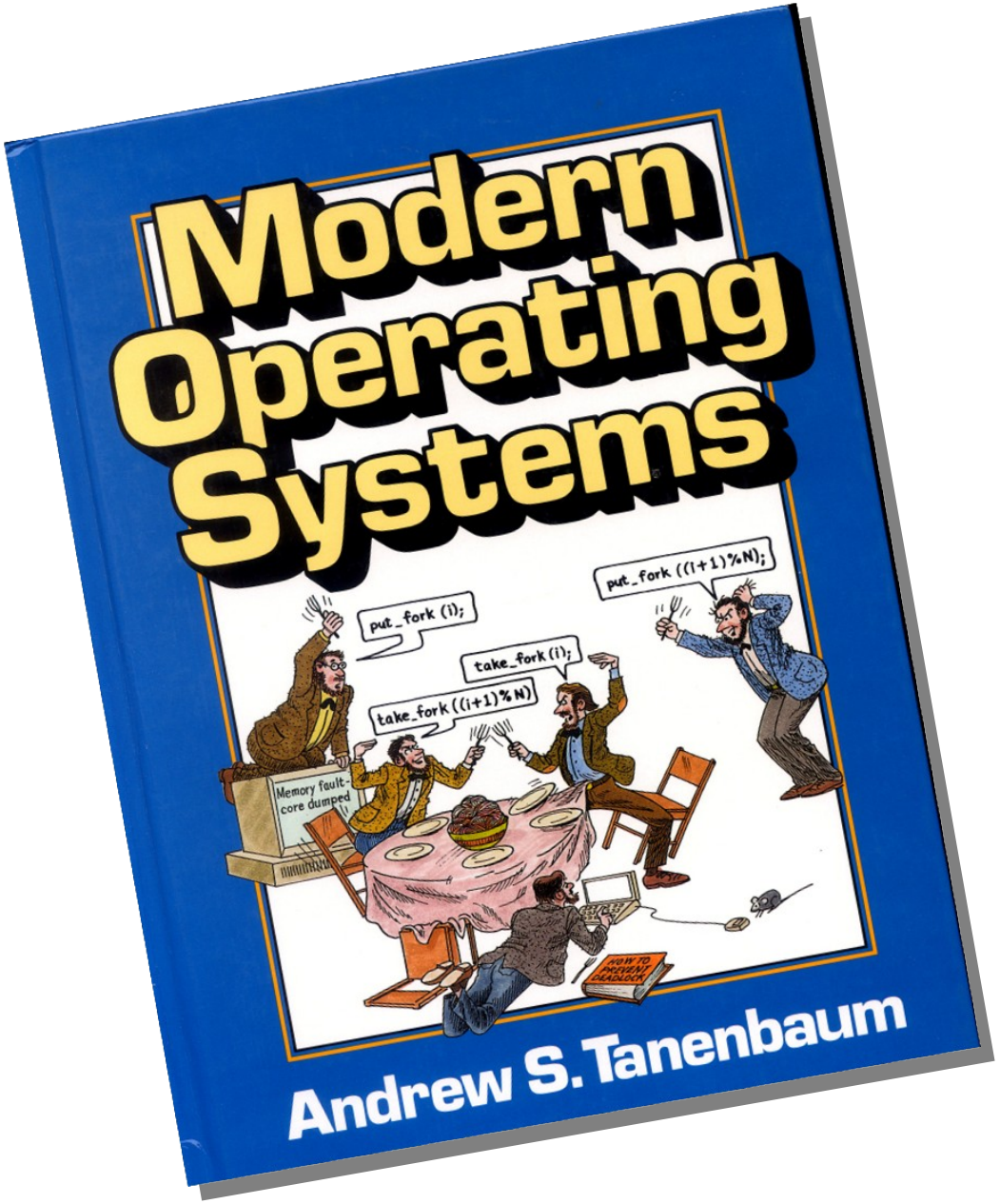
FACULTY OF MATHEMATICS AND PHYSICS
CHARLES UNIVERSITY IN PRAGUE

Do you know this professor ?



By GerardM - Own work, CC BY 2.5
https://commons.wikimedia.org/w/index.php?curid=635930

Do you know this book ?



Department of
Distributed and
Dependable
Systems

Table of contents

Department of Distributed and Dependable Systems

Table of contents

2. Processes and Threads

1962/1963 Dijkstra: Semaphores

1966 MIT: Processes and threads
1967 IBM OS/360: Multiprogramming

3. Memory Management

Address translation
- 1959 University of Manchester
- 1960s IBM 360, CDC 7600 ...
- 1970s IBM 370, DEC VMS ...
- 1985 Intel 80386

Memory caches
- 1968 IBM 360

4. File Systems

Hierarchical directories
- 1965 MIT & Bell Labs: Multics

Remote file access
- 1960s MIT: ITS

What is happening ?

selection of topics
browsing Linux Weekly News

Interesting architectures

ARM

- Memory management and virtualization
- Support for big.LITTLE architectures
- Everything Android :-)

DSP Processors

- Qualcomm Hexagon added 2011 removed 2018
- Imagination META added 2013 removed 2018

IoT Devices

- How to shrink the kernel ?

Memory management

Huge Pages and Friends

- Compaction
- Multiple huge page sizes
- Huge pages in page cache

IPC and Sealed Files

Memory Hotplugging

Compressed Memory Swap

Cache Partitioning Support

Userspace Page Fault Handling

Concurrency and scheduling

Using C11 Atomics (or Not)

- Really mind bending examples :-)

Futex Optimizations

Concurrent Resizable Hash Table

Userspace Restartable Sequences

- Processor local optimistic code sequence
- Restarted if sequence interrupted before commit

Tickless Kernel

Scheduler Aware Frequency Scaling

C11 atomics in kernel ?

```
if (x) y = 1;
else   y = 2;
```

Can we change this to the following ?

```
y = 2;
if (x) y = 1;
```

Why ?

- Can save us a branch
- Is valid for single thread
- But how about atomics ?

After ~250 messages involving names like Paul McKenney and Torvald Riegel some people are still not quite sure ...

Will Deacon, Paul McKenney, Torvald Riegel, Linus Torvalds, Peter Zijlstra et al.

gcc mailing list https://gcc.gnu.org/ml/gcc/2014-02/msg00052.html

Department of Distributed and Dependable Systems

Block devices

SSDs Everywhere

- Block cache SSD layer
- SSD journal for RAID 5 devices
- Flash translation layer in software

Atomic Block I/O

Large Block Sizes

Inline Encryption Devices

Error Reporting Issues

- Background writes can still (?) fail silently

Better Asynchronous I/O Interfaces

Multiple Queues Support

Filesystems

NVMM Is Coming

- Zero copy filesystem support
- Log structured filesystem

statx

overlayfs

Extensions to copy_file_range

Filesystem Level Event Notification

Generic Dirty Metadata Pages Management

Network Filesystem Cache Management API

Networking

Extended BPF

- JIT for extended BPF
- Tracepoints with extended BPF
- Extended BPF filters for control groups

Accelerator Offload

Shaping for Big Buffers

WireGuard VPN Merge

Security

Spectre and Meltdown and ... ?

Kernel Hardening

- Reference count overflow protection
- Hardened copy from and to user
- Kernel address sanitizer
- Syscall fuzzing
- Control flow enforcement via shadow stacks

Full Memory Encryption

File Integrity Validation

Live Kernel Patching

Department of Distributed and Dependable Systems

... and more !

Kernel Documentation with Sphinx

Continuous Integration

API for Sensors

Better IPC than D-Bus

Error Handling for I/O MMU

The 2038 Problem (or Lack Thereof)

Plus things outside kernel

- Systemd ? Wayland ? Flatpak ? CRIU ?

What is happening ?

selection of topics
browsing ACM Symposium
on Operating System Principles

2011

Securing Malicious Kernel Modules

- Enforce module API integrity at runtime

Virtualization Support

- Better isolation
- Better security

Deterministic Multithreading

- For debugging and postmortem purposes

GPU as First Class Citizen

2013

Peer to Peer Replicated File System

- Opportunistic data synchronization with history

Replay for Multithreaded Apps with I/O

Compiler for Heterogeneous Systems

- CPU, GPU, FPGA

In Kernel Dynamic Binary Translation

- Translate (virtualize) running kernel code

Detecting Optimization Unstable Code

- Compiler plugin to identify unstable patterns

Optimization unstable code ?

```
char *buf = ...;
char *buf_end = ...;
unsigned int len = ...;
if (buf + len >= buf_end) return;
   /* len too large */
if (buf + len < buf) return;
   /* overflow, buf+len wrapped around */
```

What if your compiler is (too) smart ?

- Pointer arithmetic overflow is undefined
- So ignoring the second branch is correct behavior

Wang et al.: Towards Optimization-Safe Systems
http://dx.doi.org/10.1145/2517349.2522728

Department of Distributed and Dependable Systems

2015

File System Stability Work

- Formally proven crash recovery correctness
- Formal model driven testing

Hypervisor Testing and Virtual CPU Validation

Casual Profiling

- To identify concurrent optimization opportunities

From RCU to RLU

- With multiple concurrent readers and writers

Software Defined Batteries

2017

Filesystem Innovations

- High throughput filesystem for manycore machines
- Cross media filesystem (NVMM, SSD, HDD)
- Fault tolerant NVMM filesystem

Nested Virtualization Hypervisor for ARM

Unikernel Based Lightweight Virtualization

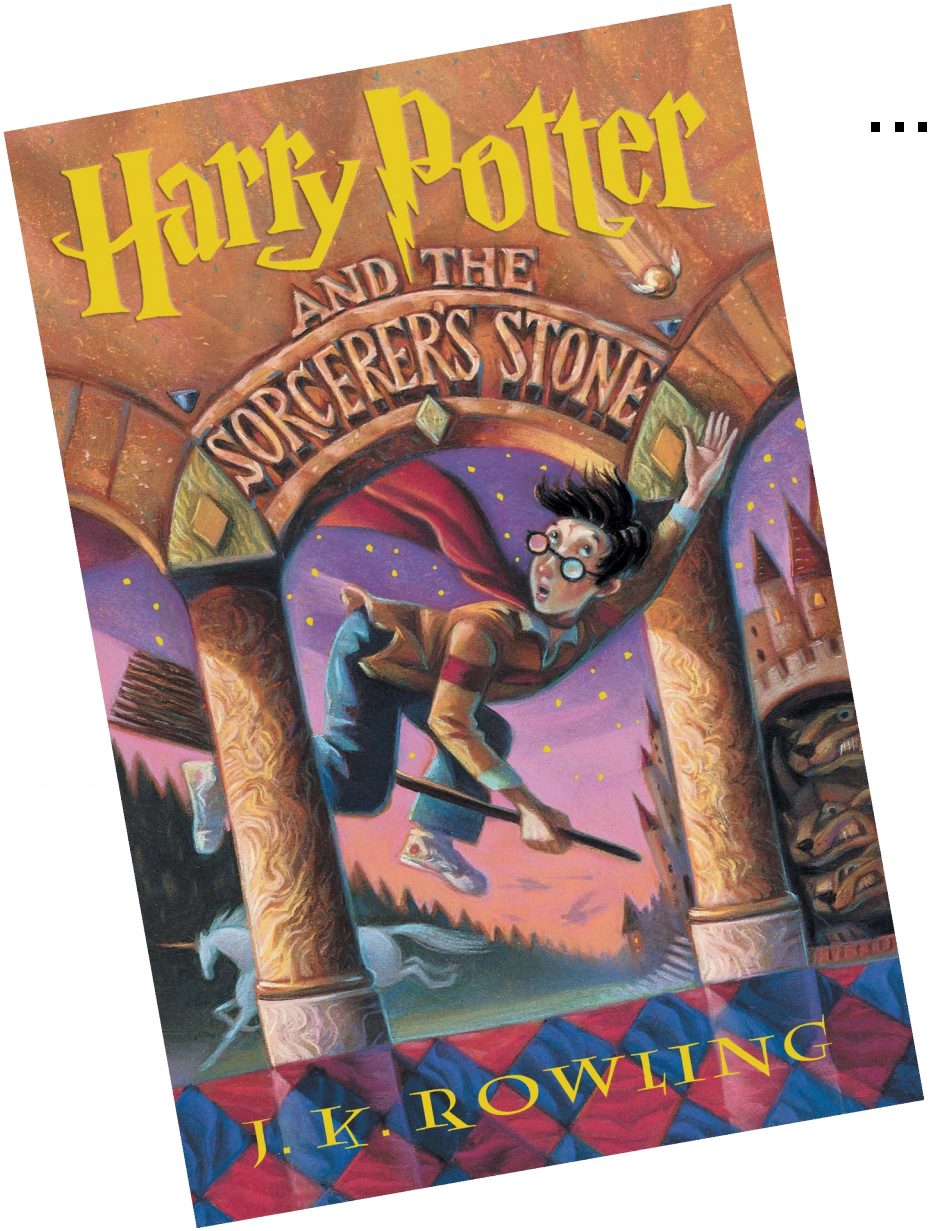
Operating System for Low Power Platforms

- Platform 64 kB SRAM, 512 kB Flash ROM
- System ~12 kB RAM, 87 kB Flash ROM
- Concurrent processes with hardware protection

And my point is ...

In standard lectures we miss all of the fun !

Sidetracking a bit ...



... Imagine this book is just out

... Sold in a kit with
a working magic wand

... Would you come here to have me read it to you ?

Architectures - Microkernels
IPC - Capabilities

Jakub Jermář
Senior Software Engineer, Kernkonzept

Department of
Distributed and
Dependable
Systems

Operating system architectures

Famous debate Tanenbaum vs Torvalds

“MINIX is a microkernel-based system …
LINUX is a monolithic style system …
This is a giant step back into the 1970s …
To me, writing a monolithic system
in 1991 is a truly poor idea.”

… so who was right ?

Operating system architectures

How to imagine a monolithic kernel ?

- Quite big (Linux ~20M LOC) multifunction library
- Written in an unsafe programming language
- Linked to potentially malicious applications
- Subject to heavily concurrent access
- Executing with high privileges

It (obviously) works but some things are difficult

- Guaranteeing stability and security
- Supporting heterogeneous systems
- Scaling with possibly many cores
- Doing maintenance

Security Enhanced Linux

Lukáš Vrabec
Software Engineer, RedHat

Department of
Distributed and
Dependable
Systems

MAC vs DAC

Discretionary Access Control

- System gives users tools for access control
- Users apply these at their discretion

Mandatory Access Control

- System defines and enforces access control policy

SELinux is NSA made MAC for Linux

How hard can it be ?

Rules that define security policy

- `allow ssh_t sshd_key_t:file read_file_perms;`
- About 150k rules for default targeted policy

Tons of places in the kernel checking that policy

- `security_file_permission (file, MAY_WRITE);`

Originally multiple policy packages

- Strict
  - Everything denied by default
  - Known programs granted privileges
- Targeted
  - Everything permitted by default
  - Known (sensitive) programs restricted

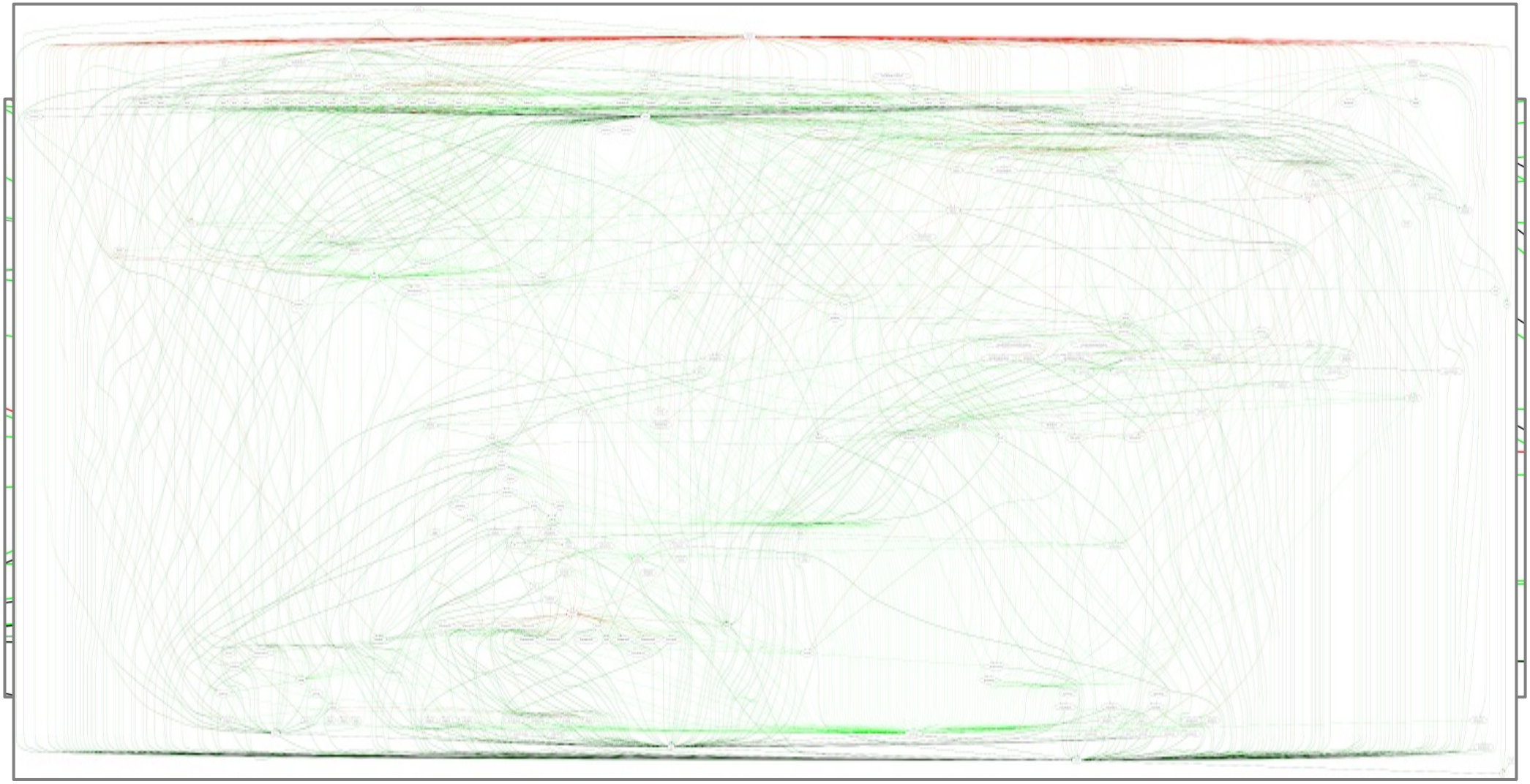
Service Management – systemd
Also OpenRC – upstart – SMF

Michal Sekletár
Senior Software Engineer, RedHat

Department of
Distributed and
Dependable
Systems

Services ? What services ?

```
> systemd-analyze dot
```



Tracing – ptrace
Profiling – SystemTap – eBPF

Michal Sekletár
Senior Software Engineer, RedHat

Department of
Distributed and
Dependable
Systems

How can we debug a process ?

The ptrace system call

- Attach to another process
- Pause, resume, single step execution
- Inspect and modify process state
  - Register content
  - Memory content
  - Signal state
- ...

How can we observe our system ?

Many tools at our disposal

- Dynamic event interception points
  - Kernel function tracer
  - Kernel probes
  - User level probes
- Event data collection buffers
- Event data processing
  - SystemTap scripts
  - Extended BPF filters

SystemTap probe script

```
global packets

probe netfilter.ipv4.pre_routing {
  packets [saddr, daddr] <<< length
}

probe end {
  foreach ([saddr, daddr] in packets) {
    printf ("%15s > %15s : %d packets, %d bytes\n",
            saddr, daddr,
            @count (packets [saddr,daddr]),
            @sum (packets [saddr,daddr]))
  }
}
```

Debugging in kernel kdump – crash - oops

Vlastimil Babka
Linux Kernel Developer, SUSE

Department of Distributed and Dependable Systems

Beyond kernel panic

Salvaging system state

- How to do that when your kernel is not safe to use ?
- What information can be salvaged

Analyzing system state

- So you have your dump …
- But what data to look at ?

Kernel Memory Management

Michal Hocko
Team Lead, Linux Kernel Developer, SUSE

Department of Distributed and Dependable Systems
D3S

Bits and pieces

Transparent Huge Pages

- Multiple memory page sizes (4 kB, 2 MB, 1 GB)
- Larger sizes make some things more efficient
  - Reduce TLB entry use
  - Reduce page table size
- Transparent use for applications ?

NUMA

memcg

NVDIMM

Advanced File Systems journaling – ZFS

Jan Šenolt
Principal Software Engineer, Oracle

Department of Distributed and Dependable Systems
D3S

Journaling for consistency

Filesystem operations are not atomic

- Operations can be interrupted by crash
- What happens when operation only half done ?

What if we knew what was the operation ?

- Note operations into journal
- Recovery with journal replay
- But how to do that and be fast ?
- And do we need standard data when we have journal ?

Virtualization – Containers

Adam Lackorzynski
Security and Systems Architect, Kernkonzept

Department of
Distributed and
Dependable
Systems

Hardware virtualization support

Very basic support

- Reliably intercepting privileged operations
  - Operations modifying state
  - Operations querying state

Required for efficiency

- Virtualized memory management
- DMA protection domains and DMA remapping
- Direct device and virtual function assignment for I/O

Networking
Linux Network Stack Design

Jiří Benc
Linux Kernel Developer, RedHat

Department of
Distributed and
Dependable
Systems

Live Kernel Patching

Miroslav Beneš
Linux Kernel Developer, SUSE

Department of
Distributed and
Dependable
Systems

How to patch executing program ?

Locating code to replace

- Function entry points known
- Think about compiler optimizations

Replacing function code

- Trampolines because code cannot be shifted easily
- What if function is currently executing ?

Can we deal with state too ?

Real Time Operating Systems Certification

Roman Kápl
Software Developer, SYSGO

Tomáš Martinec
Verification Engineer, SYSGO

Department of Distributed and Dependable Systems
D3S

Realtime is a different world !

Bounded latency of all operations

What can go wrong in a standard kernel ?

- Synchronized access to shared resources
  - Even simple malloc typically locks something
- Inaccurate process time accounting
  - Interrupts run on behalf of interrupted process
- Interference from noisy neighbors
  - Memory access latencies with caches
  - I/O latencies with queues and broken locality
- …

And can you convince other people ?

Department of Distributed and Dependable Systems
D3S

Security Exploits

Jiří Kosina
Director, Distinguished Engineer
Linux Kernel Developer, SUSE

Department of
Distributed and
Dependable
Systems