Operating Systems
Hot Topics

http://d3s.mff.cuni.cz

Martin Děcký
deky@d3s.mff.cuni.cz

HelenOS

CHARLES UNIVERSITY IN PRAGUE
faculty of mathematics and physics
Who Am I?

- Passionate programmer and operating systems enthusiast for many years
- HelenOS developer since 2005
- Computer science researcher
  - Distributed and component systems
  - Formal verification of operating system correctness
Reliability
Robustness
Dependability
A fatal exception OE has occurred at 0028:C562F1B7 in VXD ctpci9x(05) + 00001853. The current application will be terminated.

* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue
Reliability

- Some monolithic operating systems from 1990s infamous for their unreliability
  - Promise of microkernel multiserver systems to provide fundamentally better reliability
    - Smart design, simple code
    - Not enough momentum for large “market share”
- Time was good for the monolithic systems
  - Linus' Law (by Eric Raymond):
    “Given enough eyeballs, all bugs are shallow.”
  - Jermář's Law:
    “Given enough eyeball-years, all bugs are shallow.”
138 Days, 1 Hours, 19 Minutes, 41 Seconds
Robustness

- **Record uptimes are no longer considered cool**
  - Kernel bugs happen and they need to be patched
  - New kernel features are sometimes needed
    - Promise of microkernel systems for a feature-complete kernel

- **Jon Corbet:** "*Linux has no longer any formal regression tracking process.*"
  - "How do we know the kernel is getting better over time?"
  - Promise of microkernel multiserver systems for run-time component upgrade and replacement
IEEE definition

“Dependability is a measurable and provable degree of system's availability, reliability and its maintenance support”

In other words

Formal verification of correctness and quality of service with respect to predefined specification/criteria
**Dependability (2)**

- **Practical ends**
  - (Static) Driver Verifier
    - SLAM (Software, Language, Analysis and modeling) model checker
    - WHQL
  - Verifying C Compiler (VCC)
    - Invariants, pre-, postconditions using theorem prover
    - Object ownership and concurrency properties in Hyper-V
  - Promise of microkernel multiserver systems for a system-wide verification of correctness
Semantic information in source code
Dependability (3)

Semantic information in source code

Architecture and behavior specification
Dependability (3)

- Architecture and behavior specification
- Semantic information in source code
- Architecture models
Dependability (3)

- Architectures and behavior specification
- Semantic information in source code
- Architecture models
- Extra-functional properties
Dependability (3)

- Compiler checks
  - Static analyzers
  - Abstract interpretation

- Semantic information in source code

- Architecture and behavior specification

- Extra-functional properties

- Architecture models

- Compatibility checks
  - Compliance checks
  - Code generation

- Model checking
  - Use case analysis

- Performance modeling
  - Schedulability analysis
HelenOS sources
C99 with GNU extensions

Architecture Description
ADL/CDL with extensions

Behavior Description
{T|E|-}BP with preprocessing

Use cases
domain-limited English

Functional properties
Frama-C
pre/post-conditions checking
invariant checking

Extra-functional properties
timing properties
interface properties checking
(correct sequencing,
live/deadlock freedom)

Behavior Description
{T|E|-}BP with preprocessing
behavior compatibility
substituability

Use cases
domain-limited English
compliance checking
extraction

Architecture Description
ADL/CDL with extensions
stubs, skeletons,
connectors generation
run-type interface type checks

HelenOS sources
C99 with GNU extensions
interface properties checking
(correct sequencing,
live/deadlock freedom)
Industry Driven Dependability

- **Secure computing**
  - End-to-end digitally signed trusted code
    - From firmware (UEFI), over boot loaders, the kernel, kernel modules, device drivers, to any user space privileged code

- **LLVM/clang as a new unifying compiler toolchain**
  - FreeBSD, Mac OS X, HelenOS – on par with GCC
  - Linux, MINIX 3, others – solid support
  - Integration into IDEs, flexibility for verification tools
    - Detection of undefined behaviour (University of Illinois, Urbana-Champaign)
    - Arithmetic overflow checking (University of Utah)
Multicores
Manycores
• **Moore's Law still applies**
  - The number of *transistors* on integrated circuits doubles every two years (or so)

• **The golden era is over**
  - The raw single-core (sequential) performance does not double every two years (or so)
  - Parallel algorithms and concurrency are more and more important
Empowering Manycores

- **Single chip cloud computing**
  - Individual microkernel running on each core
    - Multikernel distributed system
    - Core-to-core and node-to-node communication treated as equal
    - Asynchronous messaging and state replication
    - *Barrelfish (ETH Zürich, Microsoft Research Cambridge)*
Barrelfish Architecture

The multikernel model

- Agreement algorithms
- Arch-specific code
- Heterogeneous cores
- OS node
- State replica
- App
- Interconnect
- Async messages
- [2]
Empowering Manycores (2)

- **Non-Symmetric Multiprocessing, Retargetable CPUs/hardware**
  - Utilizing a massive number of specialized co-processors
    - GPUs, big.LITTLE
    - Transparency vs. utilization
  - Dynamically reprogramming CPU cores on FPGAs
  - *ReconOS (University of Paderborn)*
Empowering Manycores (3)

- **Hardware Transactional Memory**
  - Intel Haswell microarchitecture
    - Extension to the instruction set
  - How does it relate to synchronization problems
    - *Paul McKenney: Lock elision and HTM*
  - How does it relate to synchronization methods
    - *Paul McKenney: Read-Copy-Update using HTM*
Big Data
WHAT IS BIG DATA?

VOLUME

Large amounts of data.

VELOCITY

Needs to be analyzed quickly.

VARIETY

Different types of structured and unstructured data.

WHAT ARE THE VOLUMES OF DATA THAT WE ARE SEEING TODAY?

- Facebook: 30 billion pieces of content were added to Facebook this past month by 600 million plus users.
- Zynga: Zynga processes 1 petabyte of content for players every day, a volume of data that is unmatched in the social game industry.
- YouTube: More than 2 billion videos were watched on YouTube yesterday.
- LOL: The average teenager sends 4,762 text messages per month.
- Twitter: 32 billion searches were performed last month on Twitter.

WHAT DOES THE FUTURE LOOK LIKE?

Worldwide IP traffic will quadruple by 2015.

By 2015, nearly 3 billion people will be online, pushing the data created and shared to nearly 8 zettabytes.

Everyday business and consumer life creates 2.5 quintillion bytes of data per day.

90% of the data in the world today has been created in the last two years alone.

Key questions enterprises are asking about Big Data:

- How to store and protect big data?
- How to backup and restore big data?
- How to organize and catalog the data that you have backed up?
- How to keep costs low while ensuring that all the critical data is available when you need it?

A new IDC study says the market for big technology and services will grow from $3.2 billion in 2012 to $16.9 billion in 2019. That’s a growth of 40% CAGR.

58% of respondents expect their companies to increase spending on server backup solutions and other big data-related initiatives within the next three years.

2/3rds of surveyed businesses in North America said big data will become a concern for them within the next five years.
Big Data File Systems

- Integration of previously distinct components
  - Volume management + redundancy (RAID) + silent data corruption detection + file API + transactions API
    - Copy-on-write design, log-structured
    - ZFS (Oracle)
      - btrfs (Oracle, Red Hat et al.)
      - Loris (VU Amsterdam)
      - HAMMER2 (Matthew Dillion)
  - Fault-tolerant, seamlessly replicated distributed file systems
    - Ceph (University of California, Santa Cruz)
      - HekaFS (community driven, venture capital)
Bleeding Edge Microkernel Ideas
HelenOS In-Progress Features

- **Split of mechanism and policy** design principle
  - User space driven system-wide scheduler
  - User space driven SMP management

- Rethinking the file system paradigms
  - Using capabilities for real-life user stories
    - If you cannot see it, you cannot access it
HelenOS Research Projects

• **New RCU algorithms**
  - AP-RCU (highly portable, decently scalable PaR)
  - AH-RCU (highly scalable, microkernel-friendly)

• **Implicitly shared resources management**
  - De-duplicated caching, future usage prediction (read-ahead), resource pressure evaluation (out-of-memory conditions)
Q&A
www.helenos.org
References

[1] Intel Press Kit