MICROKERNELS IN THE ERA OF DATA-CENTRIC COMPUTING

Martin Děcký
martin.decky@huawei.com

February 2018
Who Am I

- **Passionate programmer and operating systems enthusiast**
  - With a specific inclination towards multiserver microkernels
- **HelenOS developer since 2004**
- **Research Scientist since 2006**
  - Charles University (Prague), Distributed Systems Research Group
- **Senior Research Engineer since 2017**
  - Huawei Technologies (Munich), Central Software Institute
MOTIVATION
Memory Barrier

Performance

© 2003 Elsevier Science

CPU

Memory

Year


100,000 10,000 1000 100 10 1

LaMarca, Ladner (1996)

- **Quick Sort**
  - $O(n \times \log n)$ operations

- **Radix Sort**
  - $O(n)$ operations

![Graph showing the comparison between Quick Sort and Radix Sort](graph.png)

- **Quick Sort**
- **Radix Sort**

<table>
<thead>
<tr>
<th>Thousands of items</th>
<th>Instructions / Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1200</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>16</td>
<td>800</td>
</tr>
<tr>
<td>32</td>
<td>600</td>
</tr>
<tr>
<td>64</td>
<td>400</td>
</tr>
<tr>
<td>128</td>
<td>200</td>
</tr>
<tr>
<td>256</td>
<td>100</td>
</tr>
<tr>
<td>512</td>
<td>50</td>
</tr>
<tr>
<td>1024</td>
<td>25</td>
</tr>
<tr>
<td>2048</td>
<td>12</td>
</tr>
<tr>
<td>4096</td>
<td>6</td>
</tr>
</tbody>
</table>

---

LaMarca, Ladner (1996)

- **Quick Sort**
  - $O(n \times \log n)$ operations

- **Radix Sort**
  - $O(n)$ operations

![Graph showing the comparison between Quick Sort and Radix Sort](graph.png)

<table>
<thead>
<tr>
<th>Thousands of items</th>
<th>Instructions / Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1200</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>16</td>
<td>800</td>
</tr>
<tr>
<td>32</td>
<td>600</td>
</tr>
<tr>
<td>64</td>
<td>400</td>
</tr>
<tr>
<td>128</td>
<td>200</td>
</tr>
<tr>
<td>256</td>
<td>100</td>
</tr>
<tr>
<td>512</td>
<td>50</td>
</tr>
<tr>
<td>1024</td>
<td>25</td>
</tr>
<tr>
<td>2048</td>
<td>12</td>
</tr>
<tr>
<td>4096</td>
<td>6</td>
</tr>
</tbody>
</table>
LaMarca, Ladner (1996)

- **Quick Sort**
  - $O(n \times \log n)$ operations

- **Radix Sort**
  - $O(n)$ operations

![Graph comparing Quick Sort and Radix Sort](image)
LaMarca, Ladner (1996)

- **Quick Sort**
  - $O(n \times \log n)$ operations

- **Radix Sort**
  - $O(n)$ operations

![Graph showing performance comparison between Quick Sort and Radix Sort](image)
The Myth of RAM

Accessing a random memory location requires $O(1)$ operations.

Accessing a random memory location takes $O(1)$ time units.
Accessing a random memory location requires \( O(1) \) operations.

Accessing a random memory location takes \( O(1) \) time units.
The Myth of RAM

Accessing a random memory location requires \( O(1) \) operations.

Accessing a random memory location takes \( O(\sqrt{n}) \) time units.
BREAKING THE BARRIER
Von Neumann Forever?

Von Neumann architecture diagram:
- **Data**: Blue arrows
- **Control**: Red arrows
- **Status**: Green arrows

Components:
- **Input peripheral**
- **ALU**
- **Output peripheral**
- **Controller**

Diagram shows the flow of data and control signals through the components of a Von Neumann architecture.
Von Neumann Forever?

- New emerging memory technologies
  - Bridging the gap between volatile and non-volatile memory
    - No longer necessary to keep the distinction between RAM and storage (peripherals)
      - Single-level memory (universal memory)
      - See also the talk by Liam Proven (*The circuit less traveled*), Janson, Sat 13:00
  - Many technologies in development
    - Magnetoresistive random-access memory (MRAM)
    - Racetrack memory
    - Ferroelectric random-access memory (FRAM)
    - Phase-change memory (PCM)
    - Nano-RAM (Nanotube RAM)
Less Radical Solution

**Near-Data Processing**

- Moving the computation closer to the place where the data is
  - Not a completely new idea at all
    - Spatial locality in general
    - GPUs processing graphics data locally
  - Breaking the monopoly of CPU on data processing even more
    - CPUs are fast, but also power-hungry
    - CPUs can only process the data already fetched from the memory/storage
  - The more data we avoid moving from the memory/storage to the CPU, the more efficiently the CPU runs
Near-Data Processing

**Benefits**

- Decreased latency, increased throughput
- Not necessarily on an unloaded system, but improvement under load


Figure 8. Performance of SQL queries on lineitem table.
Near-Data Processing

Benefits (2)
- Decreased energy consumption

Table 5: Comparison of normalized energy consumption.

<table>
<thead>
<tr>
<th>Processing method</th>
<th>Energy consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISP (modified firmware)</td>
<td>0.142</td>
</tr>
<tr>
<td>IHP (conventional)</td>
<td>1.000</td>
</tr>
</tbody>
</table>

In-Memory Near-Data Processing

- **Adding computational capability to DRAM cells**
  - Simple logical comparators/operators that could be computed in parallel on individual words
    - Filtering based on bitwise pattern
    - Bitwise operations
  - Making use of the inherent parallelism
    - Avoiding moving unnecessary data out of DRAM
    - Avoiding linear processing of independent words of the data in CPU
Dynamic RAM
Dynamic RAM with NDP
In-Storage Near-Data Processing

- Adding computational capability to SSD controllers
  - Again, making use of inherent parallelism of flash memory
  - But SSD controllers also contain powerful embedded cores
    - Flash Translation Layer, garbage collection, wear leveling
    - Thus computation is not limited to simple bitwise filtering and operations
  - Complex trade-offs between computing on the primary CPU and offloading to the SSD controller
Our Prototype

- Based on OpenSSD
  - http://openssd.io/
  - Open source (GPL) NVMe SSD controller
    - Hanyang University (Seoul), Embedded and Network Computing Lab
  - FPGA design for Xilinx Xynq-7000
    - ONFI NAND flash controller with ECC engine
    - PCI-e NVMe host interface with scatter-gather DMA engine
  - Controller firmware
    - ARMv7
    - Flash Translation Layer, page caching, greedy garbage collection
Our Prototype (2)

**NVMe NDP extensions**

- NDP module deployment
  - Static native code so far, moving to safe byte-code (eBPF)
- NDP datasets
  - Safety boundaries for the NDP modules (for multitenancy, etc.)
- NDP Read / Write
  - Extensions of the standard NVMe Read / Write commands
  - NDP module executed on each block, transforms/filters data
- NDP Transform
  - Arbitrary data transformations (in-place copying, etc.)
  - Flow-based computational model
Our Prototype (3)

- **Fast prototyping**
  - QEMU model of the OpenSSD hardware (for running the OpenSSD firmware on ARMv7)
  - Connected to a second host QEMU/KVM (as a regular PCI-e NVMe storage device)

- **Planned evaluation**
  - Real-world application proof-of-concept
    - Custom storage engine for MySQL with operator push-down
    - Ceph operator push-down
    - Key-value store
    - Generic file system acceleration
  - Still a very much work-in-progress
    - Preliminary results quite promising
HOW MICROKERNELS FIT INTO THIS?
Future Vision

- Not just near-data processing, but data-centric computing
  - As opposed to CPU-centric computing
    - Running the computation dynamically where it is the most efficient
    - Not necessarily moving the data to the central processing unit
      - The CPU is the orchestrator
  - Massively distributed systems
    - Within the box of your machine (desktop, laptop, smartphone)
    - Outside your machine (edge cloud, fog, cloud, data center)
  - Massively heterogeneous systems
    - Different ISAs
    - Fully programmable, partially programmable, fixed-function
Future Vision & Microkernels

- application
- network stack
- security server
- naming server
- location server
- device multiplexer
- device driver
- file system multiplexer
- file system driver
- memory mgmt
- scheduler
- IPC
- microkernel
- CPU

privileged mode
unprivileged mode
Future Vision & Multi-Microkernels

- Distributed application
- Security server
- Device multiplexer
- File system multiplexer
- Naming server
- Location server
- Device driver server
- File system driver server
- ... (network stack)
- ... (unprivileged mode)
- Compositor

Microkernels in the Era of Data-Centric Computing
Microkernels for Data-Centric Computing

- **Multiserver microkernels**
  - Naturally support distribution and heterogeneity
    - IPC is a well-defined mechanism to communicate across isolation boundaries
      - Communicating between multiple (heterogeneous) computing cores is just a different transport
      - Message passing as the least common denominator
        - But also memory coherency emulation is possible
    - Multiserver designers are used to think in terms of fine-grained components
      - Composing an application from multiple components
      - Missing piece: Framework for automatic workload migration
Why not multikernels?

- Yes, why not!
- Many multikernels (i.e. Barrelfish) are essentially microkernels to the extreme
  - Each core running its independent microkernel
  - No memory coherency assumed

Why not unikernels?

- OK, if you insist
- A microkernel running a single fixed payload (e.g. a virtual machine) could be considered a unikernel
Road Forward

- **As usual, incremental steps**
  - Start with just the basic workload off-loading to the SSD, NIC, etc.
  - Gradually transform the run-time for the workload off-loading into a proto-microkernel
  - Consider the dynamic workload migration framework
  - Gradually move to a full-fledged multi-microkernel
ENGINEERING FLOWCHART

DOES IT MOVE?

NO

SHOULD IT?

NO

NO PROBLEM

YES

SHOULD IT?

YES

YES PROBLEM

NO

NO PROBLEM
Summary

- There are potential ways of abandoning the von Neumann architecture
  - Revolutionary: Single-level (universal) memory
  - Evolutionary: Near-Data Processing
- Near-Data Processing is beneficial in terms of performance and energy consumption
- Data-centric computing leads to massively distributed and heterogeneous systems
  - Multi-microkernels are a natural match for such architectures
Acknowledgements

- Huawei Technologies in-storage NDP team
  - Antonio Barbalace
  - Martin Decky
  - Anthony Iliopoulos
  - Javier Picorel
  - Dmitry Voytik
  - Hangjun Zhao

- References & Inspiration
  - Mark Silberstein (Technion)
  - Biscuit framework (Samsung Electronics)
Q&A
THANK YOU!