INFLUENTIAL OS RESEARCH

Multiprocessors

Tobias Stumpf

SS 2014
Roadmap

Multiprocessor Architectures
Usage in the Old Days (mid 90s)
Disco
Present Age Research
The Multikernel
Helios
Three Papers in one Slide
References
MULTIPROCESSOR ARCHITECTURES

SMP
MULTIPROCESSOR ARCHITECTURES
NUMA
MULTIPROCESSOR ARCHITECTURES
Distributed System
MULTIPROCESSOR ARCHITECTURES
Cache Problematic
MULTIPROCESSOR ARCHITECTURES
Test-And-Set

1. read: lock=0
2. read: lock=0
3. write: lock=1
4. write: lock=1

initially: lock = 0
MULTIPROCESSOR ARCHITECTURES
The Stanford FLASH Multiprocessor

Source: [KOH+ 98]
**Usage in the Old Days (Mid 90s)**

System Software

**Partitioning**
- run multiple independent OS
- communicate like distributed systems
- e.g. Sun Enterprise10000, Digital’s Galaxies OS

**Large OS**
- single OS controls all resources
- OS creates resource partitions, which can communicate
- e.g. Hive, Hurriance, Cellular-IRIX
DISCO
Running Commodity OSes On Scalable Multiprocessors

Edouard Bugnion, Scott Devine, Kinshuk Govil and Mendel Rosenblum
DISCO
Overview

Source: [BDGR97]
DISCO
Virtual Machine Monitor

- Disco emulates CPU (MIPS R10000), MMU, I/O devices, network
- virtual hardware specification to tune an OS for Disco
- creates a uniform address space for non-NUMA aware OS
- implemented as multi-threaded shared memory program
- virtual processors can be time shared across the physical cores
- support affinity scheduling to increase data locality
DISCO
Memory Management

Source: [BDGR97]
DISCO
Memory Management

Physical Memory of VM 1

Physical Memory of VM 2

Machine Memory

Source: [BDGR97]
DISCO
Memory Management

Source: [BDGR97]
DISCO
Memory Management

Source: [BDGR97]

IOS - Multiprocessors
slide 16
DISCO
Test Applications

- Pmake: multiprogrammed, short-lived, system and I/O intensive processes
- Engineering: multiprogrammed, long running process
- Splash: parallel applications
- Database: single memory intensive process
DISCO
Overhead

- 3% - 16% overhead
- higher overhead for applications with high TLB miss rate
- virtualization costs for some OS services can exceed native costs
DISCO
Overhead

Source: [BDGR97]
PRESENT AGE RESEARCH
State of the Art

- OSes are designed for ccNUMA and SMP machines
- CPUs are treated as interchangeable parts
- OSes treats programmable devices like non-programmable I/O devices
- General purpose OSes are optimized for common hardware
- Multiprocessor OS for machines with several hundred processors and more than one Tera byte of memory already exist
Present Age Research

Why are New OS Designs Necessary?

- hardware diversity is increasing (GPUs, FPGAs, programmable controllers)
- need for specific hardware optimizations
- more than one instruction set
- cache coherency is given up (for instance between CPU and GPU/NICs)
Challenges of Heterogeneous Hardware

- moving a process between different cores is difficult
- lack of cache coherence
- execution time depends on the core
PRESENT AGE RESEARCH
Example - Multicore
PRESENT AGE RESEARCH
Example - Multicore
PRESENT AGE RESEARCH
Interconnection
PRESENT AGE RESEARCH
Interconnection
PRESENT AGE RESEARCH

Interconnection
The Multikernel
A New OS Architecture for Scalable Multicore Systems

Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach and Akhilesh Singhania

Idea

- new OS structure, trading multiprocessor system as a network of independent cores
- no inter-core sharing, provides traditional OS services as network services
THE MULTIKERNEL
Design Principles

- make all inter-core communication explicit
- make OS structure hardware-neutral
- view state as replicated instead of shared

Source: [BBD⁺ 09]
The Multikernel

Shared Memory vs. Message Passing

- cache coherence is expensive, but simplifies memory view
- message passing costs less than shared memory

Source: [BBD+09]
THE MULTIKERNEL
Cache Coherence is not a Panacea

Pros

• simplifies the programmers life

Cons

• with increasing core count cache coherency protocols become expensive
• cache coherence restricts the ability to scale up to around 80 cores
• NICs, GPUs maintain no cache coherence with CPUs
THE MULTIKERNEL
Communication

- explicit network communication between two cores
- shared-memory areas only for message buffers
- applications can use shared memory
- separation provides resource isolation and management
THE MULTIKERNEL

System Structure

- separate OS components
- easily adaptable to new hardware
- hardware independent message passing protocols

Source: [BBD+09]

IOS - Multiprocessors
THE MULTIKERNEL
Implementation Details

Process structure

- one process is represented by several dispatching objects
- one dispatching object per possible core
- dispatching objects are scheduled by the local CPU driver
- communication is between dispatchers
Memory Management

- capability based system to decentralize resource allocation
- virtual memory management at user level
- most memory management operations need global coordination
Inter-core communication

- critical for a multikernel
- using message passing
- Barrelfish uses a user-level RPC mechanism
- memory regions are shared to transfer cache-line sized messages
- receiving uses polling
The Multikernel
Implementation Details

Hardware Knowledge Base

- information about the underlying hardware
- used for optimization
THE MULTIKERNEL
Evaluation - TLB Shutdown

Source: [BBD+09]
THE MULTIKERNEL
Evaluation - Memory Unmap

Source: [BBD+ 09]
THE MULTIKERNEL
Evaluation - Compute-bound Workloads

Source: [BBD+09]
HELIOS
Heterogeneous Multiprocessing with Satellite Kernels

*Edmund B. Nightingale, Orion Hodson, Ross McIlroy, Chris Hawblitzel, and Galen Hunt*

- OS for heterogeneous platforms
- providing a single, uniform set of OS abstractions across different cores
- OS services (e.g. file system access) provided via remote message passing (like distributed systems)
- simplified application deployment and tuning
Satellite Kernels

- satellite kernels export a single, uniform set of OS abstractions across CPUs
- a satellite kernel is composed of a scheduler and a memory manager and has to coordinate remote communication
- Helios’s satellite kernels trade a NUMA architecture as a shared nothing multiprocessor
- kernel code is replicated instead of shared
- one satellite kernel is the coordinator, which launches the other kernels
HELIOS
Satellite Kernels - Requirements

1. avoid unnecessary remote communication
2. require minimal hardware primitives
3. require minimal hardware resources
4. avoid unnecessary local IPC
HELIOS Communication

- message-passing interface
- local / remote communication is transparent for the applications
- zero-copy protocol for local messages
- implementation for remote communication is hardware dependent
- during boot-up the controller makes a point-to-point connection to all the satellite kernels
HELIOS

Time (Microseconds)

Message size (bytes)

LMP
RMP XScale
RMP NUMA

Source: [NHM⁺ 09]
• per Satellite Kernel
• no resource sharing between two kernels
• a process can exist only on one satellite kernel
• all threads of one process have to be executed on the same kernel
Helios is based on a modified Singularity RDK
Singularity applications are type and memory safe
Memory protection is ensured by software isolation
Only one address space and all applications run with the highest privileges
HELIOS Namespace

- only component which needs a centralized control
- managed by the coordinator kernel
- an application makes a service available by registration
- a second application can bind to this service
- coordinator kernel sends a message to create a new channel
- de-registration by an explicit remove message or closing the channel to the namespace
HELIOS
Placement of Applications

- placement is performance critical
- Helios makes automatic placement decisions depending on a affinity metric

Affinity Metric
Positive hint that two components benefit from a fast message passing or the execution of one component on a specific core

Neutral no affect (standard case)

Negative expresses an interference between two components
HELIOS
Placement Example

Coordinator kernel
x86

Satellite kernel
XScale Programmable Device

Source: [NHM+ 09]

IOS - Multiprocessors
HELIOS
Evaluation - Benefits of Performance Isolation

Scheduling

Mail Server

Source: [NHM⁺ 09]

IOS - Multiprocessors
THREE PAPERS IN ONE SLIDE

- Disco uses virtualization to partition the hardware and run different OSes which are not necessary multiprocessor aware.
- Barrelfish is a multikernel approach which treats the machine as a network of independent cores with minimal inter-core sharing of OS data.
- Helios uses one control kernel and several satellite kernels, with smallest possible interference to provide scalability.
References

Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania.

The multikernel: A new os architecture for scalable multicore systems.

Edouard Bugnion, Scott Devine, Kinshuk Govil, and Mendel Rosenblum.

Disco: Running commodity operating systems on scalable multiprocessors.
REFERENCES

The stanford flash multiprocessor.

Edmund B. Nightingale, Orion Hodson, Ross McIlroy, Chris Hawblitzel, and Galen Hunt.
Helios: Heterogeneous multiprocessing with satellite kernels.