TECHNISCHE UNIVERSITÄT DRESDEN

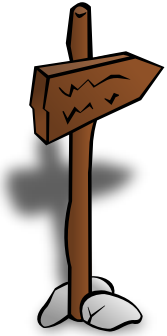
Influential OS Research

Multiprocessors

Jan Bierbaum
Tobias Stumpf

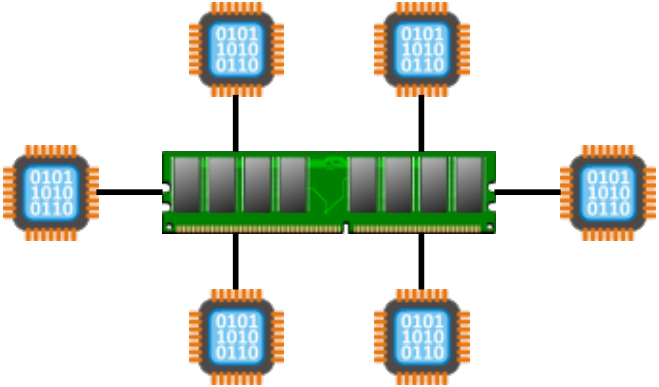
SS 2018

Roadmap



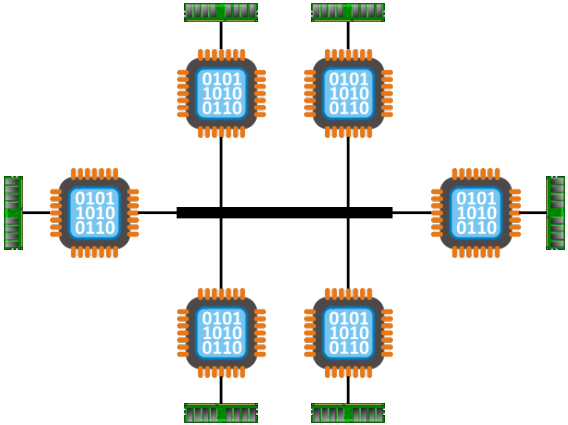
Multiprocessor Architectures

SMP



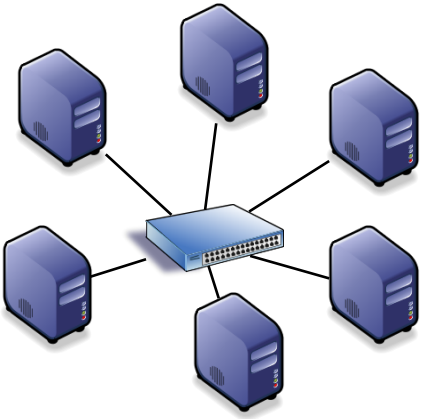
Multiprocessor Architectures

NUMA



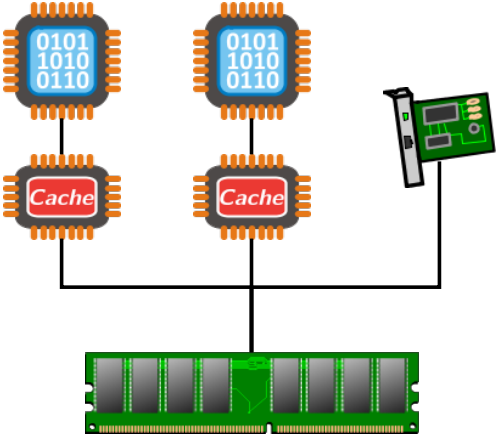
Multiprocessor Architectures

Distributed System



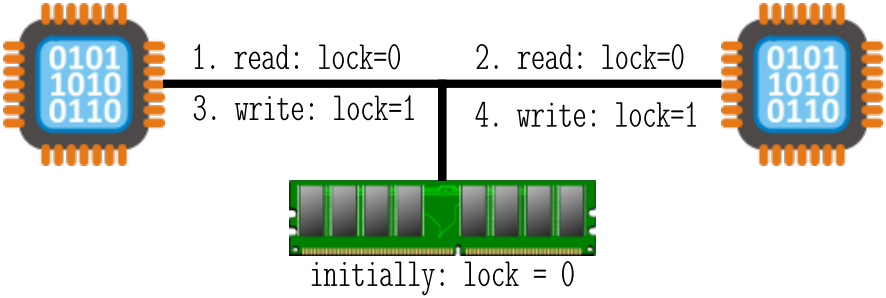
Multiprocessor Architectures

Cache Problematic



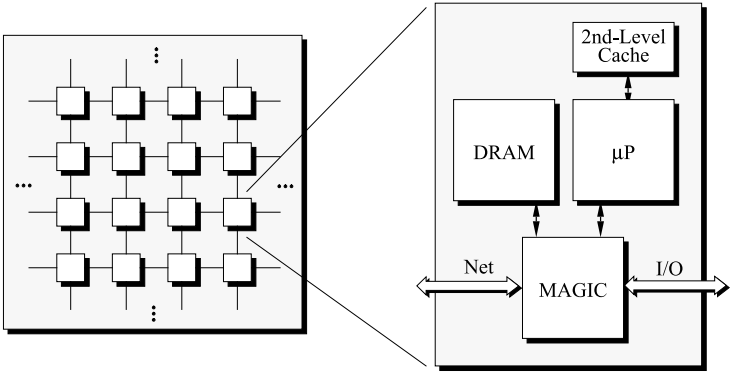
Multiprocessor Architectures

Test-And-Set



Multiprocessor Architectures

The Stanford FLASH Multiprocessor



Source: [KOH+94]

Usage in the Old Days (mid 90s)

System Software

Partitioning

- run multiple independent OSes
- 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

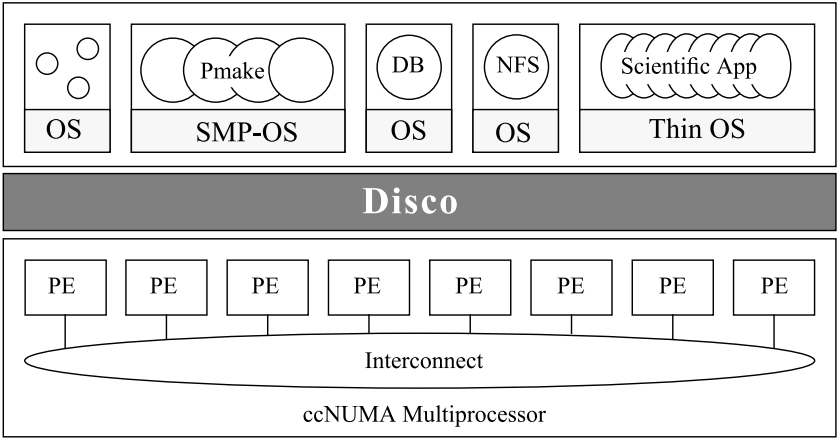
Disco

Running Commodity OSes On Scalable Multiprocessors

Edouard Bugnion, Scott Devine, Kinshuk Govil and Mendel Rosenblum

Disco

Overview



Source: [BDGR97]

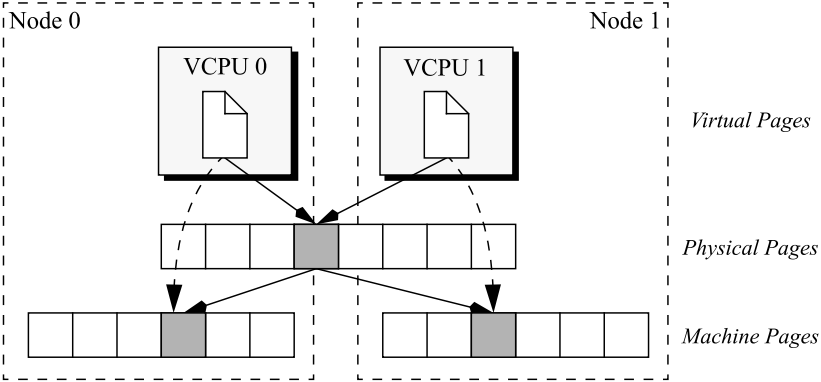
Disco

Virtual Machine Monitor

- implemented as multi-threaded shared-memory program
- virtualises CPU (MIPS R10000), MMU, I/O devices, network
- HAL modified to adapt an OS for Disco (optimisation: paravirtualisation)
- near-uniform address space for non-NUMA aware OSes
- virtual processors can time-share physical cores
- COW disks and shared buffer cache

Disco

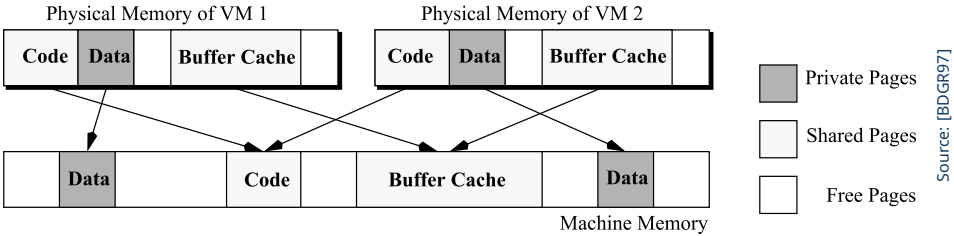
Memory Management



Source: [BDGR97]

Disco

Memory Management



Source: [BDGR97]

Setup

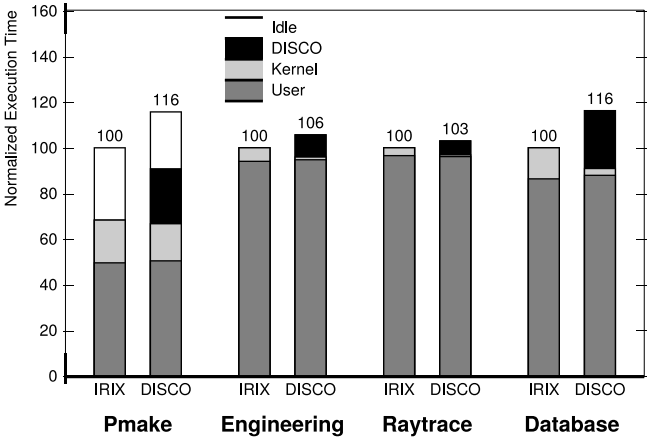
- FLASH processor, modeled in SimOS machine simulator
- 32 KB caches, 1 MB board-level cache
- memory latency: 300 ns local, 900 ns remote

Applications

Pmake	multiprogrammed, short-lived, system and I/O intensive processes (software development)
Engineering	multiprogrammed, long running process (hardware development)
Raytrace	parallel applications (scientific computing)
Database	single, memory intensive process (commercial database)

Disco

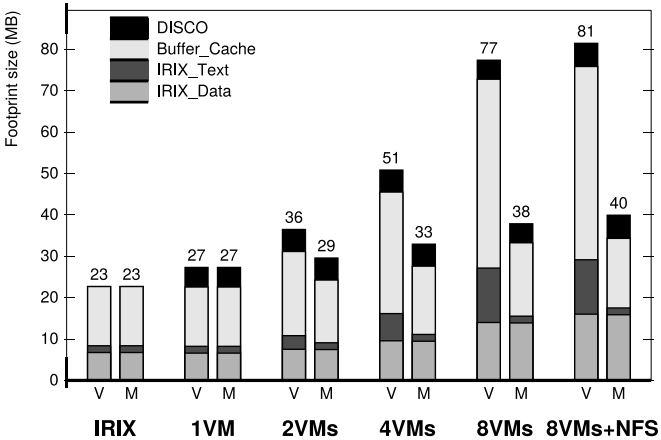
Evaluation — Virtualisation Overhead



Source: [BDGR97]

Disco

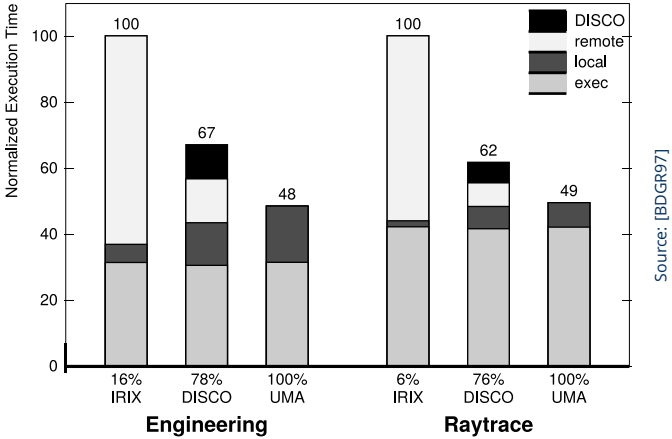
Evaluation — Data Sharing



Source: [BDGR97]

Disco

Evaluation — Page Migration and Replication



Source: [BDGR97]

Present Age Research

State of the Art

- modern OSes designed for ccNUMA and SMP machines
- CPUs as fully interchangeable parts
- OS treats programmable devices like non-programmable I/O devices $\rightarrow$ access via drivers
- general purpose OSes are optimised for common hardware and usage scenarios
- multiprocessor OSes for machines with several hundred processors and $>$ 1 TB of memory already exist

Present Age Research

Why Do We Even Need New OS Designs?

- hardware diversity is increasing: GPUs, FPGAs, programmable controllers
- need for specific hardware optimisations (caches, interconnects, . . .)
- more than one ISA
- cache coherency given up, e.g. between CPU and GPU/NIC

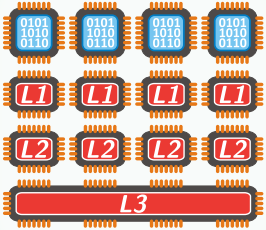
Present Age Research

Challenges of Heterogeneous Hardware

- moving a process between different (types of) cores is difficult
- lack of cache coherence
- execution time depends on the core

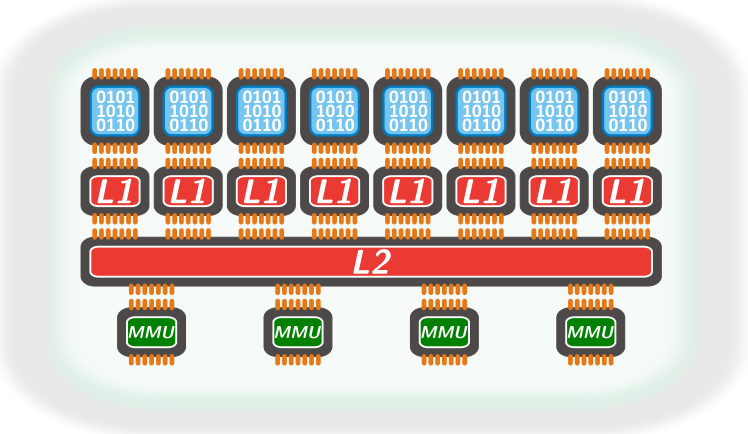
Present Age Research

Example — Multicore



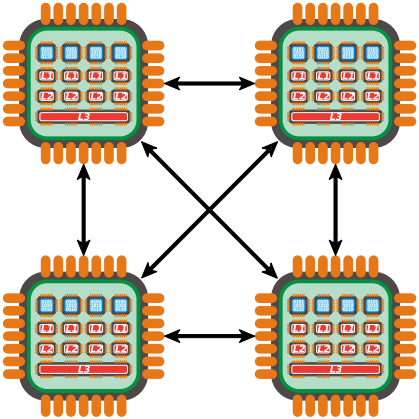
Present Age Research

Example — Multicore



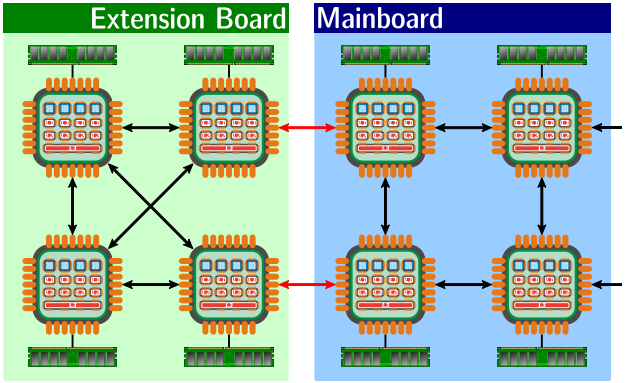
Present Age Research

Interconnect



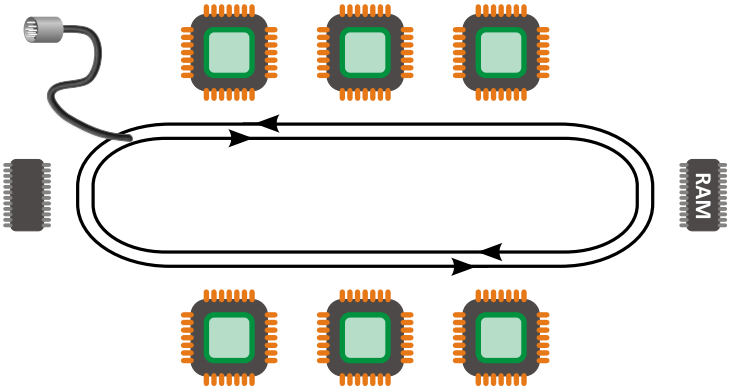
Present Age Research

Interconnect



Present Age Research

Interconnect



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, treating multiprocessor system as a network of independent cores
- traditional OS services become “network services”

The Multikernel

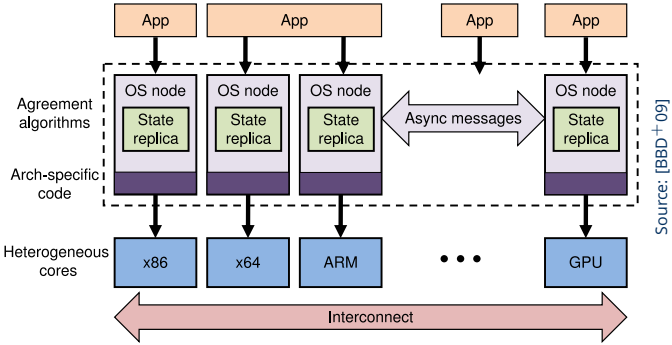
Design Principles

1. explicit inter-core communication
2. hardware-neutral OS structure
3. replicated state instead of shared state

The Multikernel

Design Principles

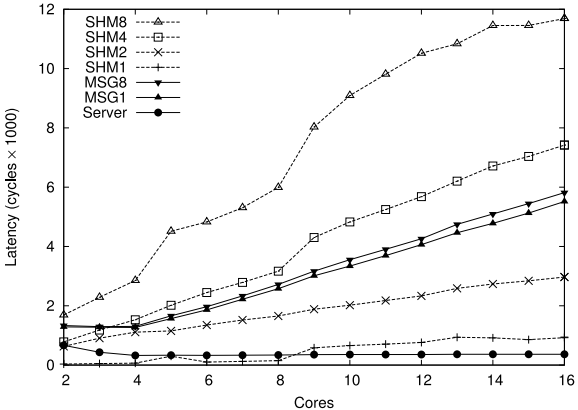
1. explicit inter-core communication
2. hardware-neutral OS structure
3. replicated state instead of shared state



The Multikernel

Shared Memory vs. Message Passing

- cache coherence simplifies memory view but has its price
- message passing less expensive than shared memory



Source: [BBD+09]

The Multikernel

Cache Coherence is not a Panacea

Pros

- simplifies the programmer's life

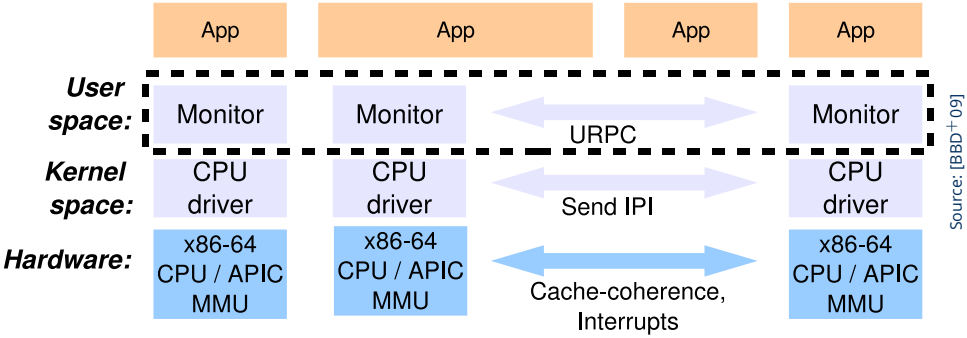
Cons

- with increasing core count cache coherency protocols become expensive
- cache coherence restricts the ability to scale up the number of cores
- NICs, GPUs maintain no cache coherence with CPUs

The Multikernel

System Structure (Barrelfish)

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



Process Structure

- each process is represented by one dispatcher object per core it may run on
- communication between dispatchers
- dispatching objects scheduled by the local CPU driver via upcall

Process Structure

- each process is represented by one dispatcher object per core it may run on
- communication between dispatchers
- dispatching objects scheduled by the local CPU driver via upcall

Communication

- critical for a Multikernel
- user-level RPC mechanism
- explicit network-like message passing
- local messages via CPU driver
- shared memory areas to transfer cache-line-sized message between cores
- receiving via polling . . . for some time (cp. spin lock)

Memory Management

- capability based system to decentralise resource allocation
- virtual memory management entirely at user level
- most memory management operations need global coordination

Memory Management

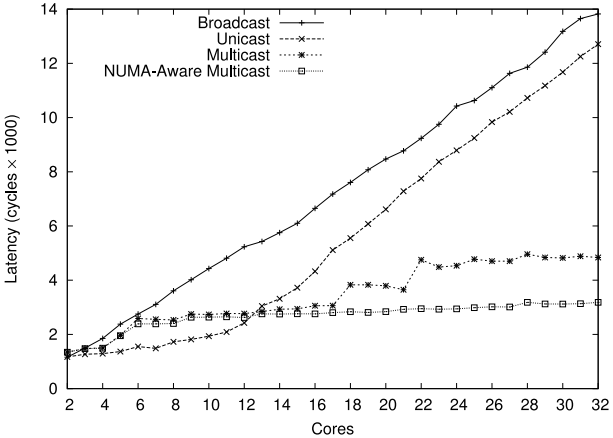
- capability based system to decentralise resource allocation
- virtual memory management entirely at user level
- most memory management operations need global coordination

System Knowledge Base (SKB)

- information about the underlying hardware
- populated during boot (hardware discovery), runtime (measurements) and pre-asserted facts
- used for optimisation

The Multikernel

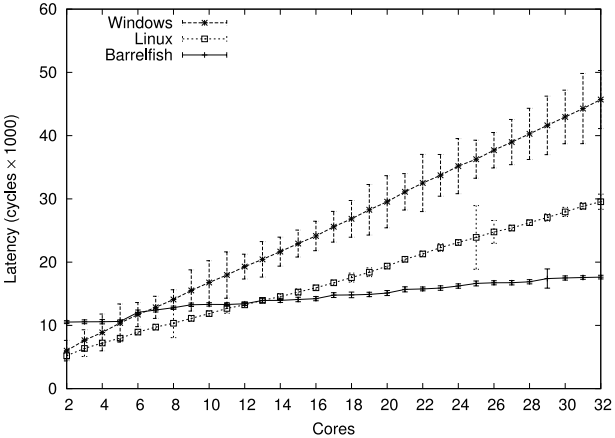
Evaluation — TLB Shootdown



Source: [BBD+09]

The Multikernel

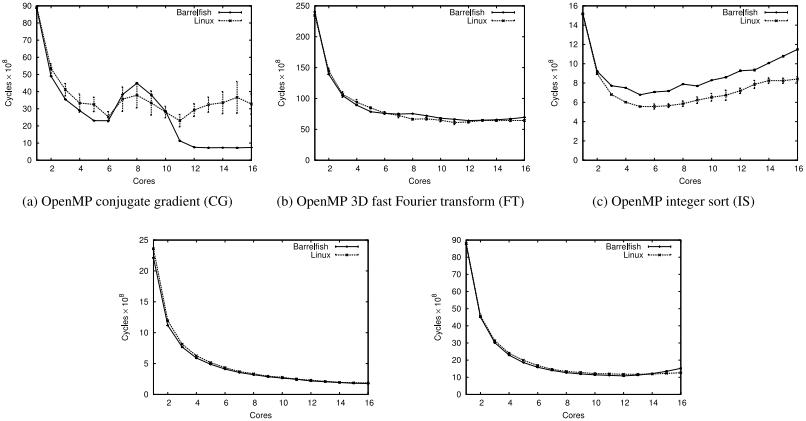
Evaluation — End-to-end Memory Unmap



Source: [BBD+09]

The Multikernel

Evaluation — Compute-bound Workloads



(a) OpenMP conjugate gradient (CG)

(b) OpenMP 3D fast Fourier transform (FT)

(c) OpenMP integer sort (IS)

(d) SPLASH-2 Barnes-Hut

(e) SPLASH-2 radiosity

Source: [BBD+09]

Helios

Heterogeneous Multiprocessing with Satellite Kernels

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

Idea

- OS for heterogeneous platforms
- simplify application writing, deploying and tuning
- single, uniform set of OS abstractions across different ISAs
- OS services (e.g. file system) provided via remote message passing, cp. distributed systems

Helios

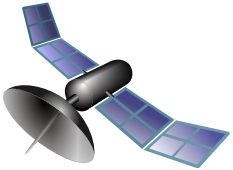
Satellite Kernels

- manages “its” hardware: scheduler + memory manager + remote communication primitives
- Helios treats a NUMA architecture as a shared-nothing multiprocessor
- kernel code is replicated instead of shared
- boot-up kernel launches other satellite kernels and acts as coordinator

Helios

Satellite Kernels — Design Principles

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



Helios

MMU-less Memory Protection

- Helios is based on a modified Singularity RDK
- Singularity application are type and memory safe
- memory protection is ensured by software isolation
- single address space per kernel, all applications run with the highest privileges

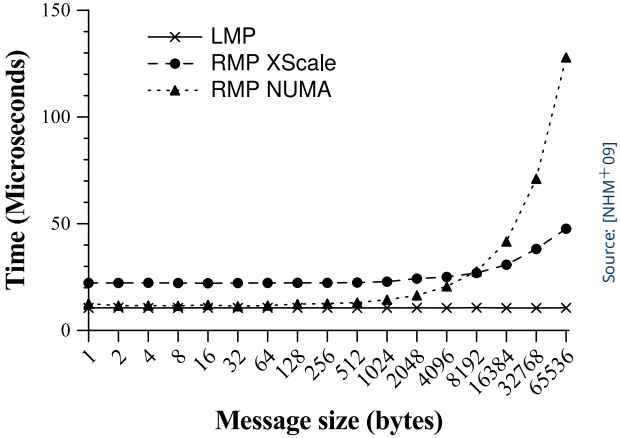
Helios

Communication

- message-passing interface
- local/remote communication is transparent for the applications
- zero-copy protocol (see Singularity [FAH+06]) for local messages
- hardware-dependent implementation for remote communication
- during boot-up the controller kernel establishes point-to-point connections between all satellite kernels

Helios

Evaluation — Communication



Source: [NHM+09]

Helios

Resource Management

- no resource sharing between satellite kernels
- each process exists only on one satellite kernel
- no migration between kernels $\rightarrow$ initial placement is final

Helios

Namespace

- managed by the coordinator kernel
- only component which uses a centralised control instance
- applications make services available via registration
- other application can bind to this service, i.e. request to use it
  $\rightarrow$ coordinator kernel forwards the message
  $\rightarrow$ service creates new, direct channel
- de-registration by an explicit remove message or closing the channel to the namespace

Helios

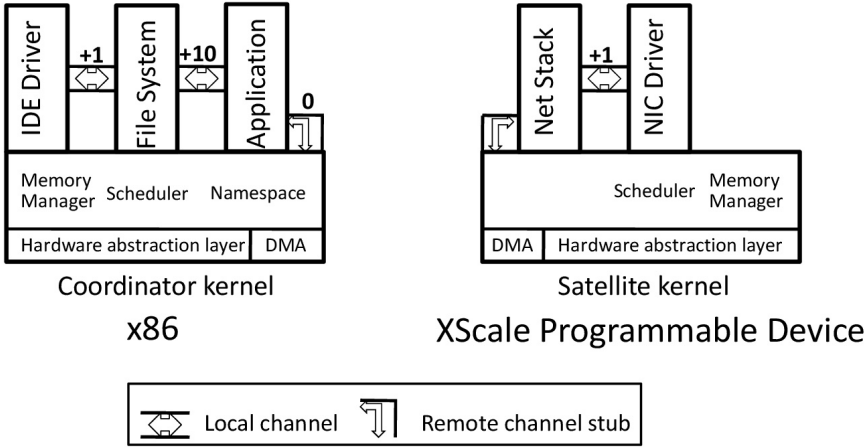
Placement of Applications

- placement is performance critical
- Helios makes automatic placement decisions considering a user-specified affinity metric

Positive	two components benefit from a fast message passing or component prefers execution on a specific ISA
Neutral	no affect (standard case)
Negative	expresses an interference between two components

Helios

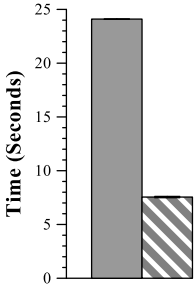
Placement Example



Source: [NHM+09]

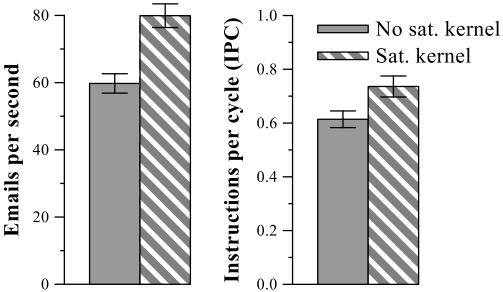
Helios

Evaluation — Benefits of Performance Isolation



Scheduling

Source: [NHM+09]



Mail Server

Three Papers in one Slide

Disco

- virtualisation to manage/partition the hardware
- run different OSes which are not necessarily multiprocessor-aware

Multikernel/“Barrelfish”

- machine as a network of independent cores
- minimal inter-core sharing of OS data: replication + agreement

Helios

- single control kernel and several satellite kernels
- affinity for simplified tuning

References

[BBD+09] 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.
In *Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles*, pages 29–44. ACM, 2009.

[BDGR97] Edouard Bugnion, Scott Devine, Kinshuk Govil, and Mendel Rosenblum.
Disco: Running commodity operating systems on scalable multiprocessors.
ACM Transactions on Computer Systems, 15(4):412–447, 1997.

[FAH+06] Manuel Fahndrich, Mark Aiken, Chris Hawblitzel, Orion Hodson, Galen Hunt, Jim Larus, and Steven Levi.
Language support for fast and reliable message-based communication in Singularity OS.
In *Proceedings of the EuroSys 2006 Conference*, page 177–190. Association for Computing Machinery, Inc., April 2006.

[KOH+94] Jeffrey Kuskin, David Ofelt, Mark Heinrich, John Heinlein, Richard Simoni, Kourosh Gharachorloo, John Chapin, David Nakahira, Joel Baxter, Mark Horowitz, et al.
The Stanford FLASH Multiprocessor.
In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 302–313. IEEE, 1994.

[NHM+09] Edmund B. Nightingale, Orion Hodson, Ross McIlroy, Chris Hawblitzel, and Galen Hunt.
Helios: Heterogeneous multiprocessing with satellite kernels.
In *Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles*, pages 221–234. ACM, 2009.