TECHNISCHE UNIVERSITÄT DRESDEN

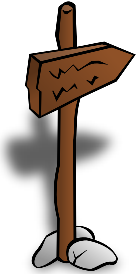
INFLUENTIAL OS RESEARCH

Multiprocessors

Tobias Stumpf

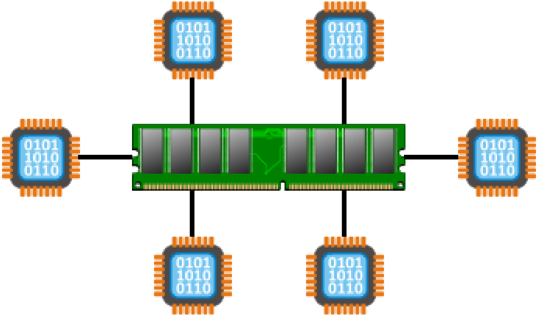
SS 2014

Roadmap



TECHNISCHE UNIVERSITÄT DRESDEN

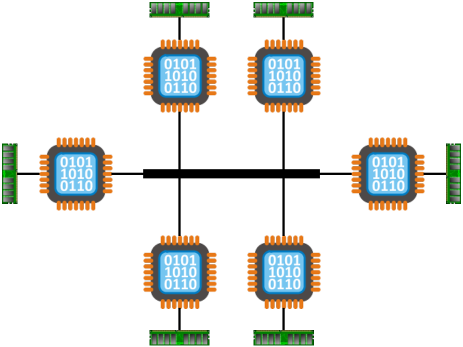
MULTIPROCESSOR ARCHITECTURES
SMP



TECHNISCHE UNIVERSITÄT DRESDEN

MULTIPROCESSOR ARCHITECTURES

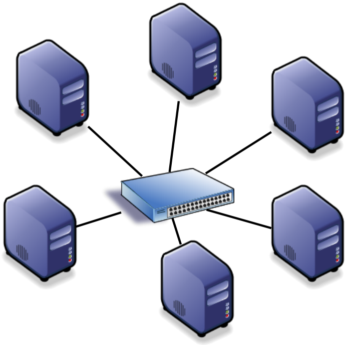
NUMA



TECHNISCHE UNIVERSITÄT DRESDEN

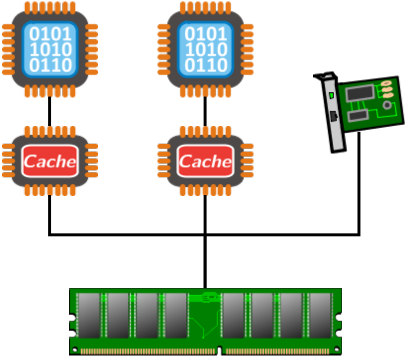
Multiprocessor Architectures

Distributed System



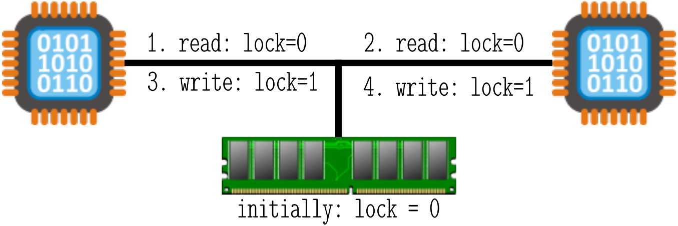
MULTIPROCESSOR ARCHITECTURES

Cache Problematic



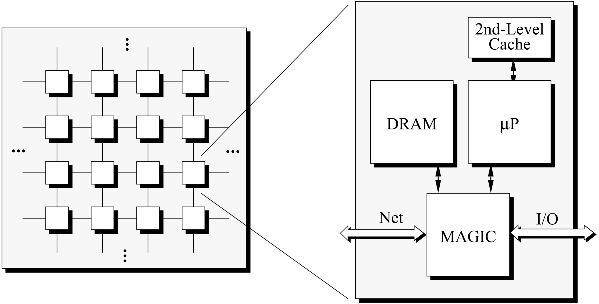
Multiprocessor Architectures

Test-And-Set



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

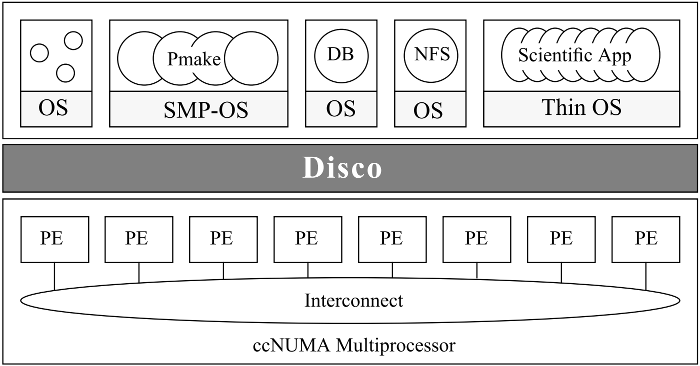
- single OS controls all resources
- OS creates resource partitions, which can communicate
- e.g. Hive, Hurriance, Cellular-IRIX

DISCO

Running Commodity OSes On Scalable Multiprocessors

Edouard Bugnion, Scott Devine, Kinshuk Govil and Mendel Rosenblum

Disco
Overview



Source: [BDGR97]

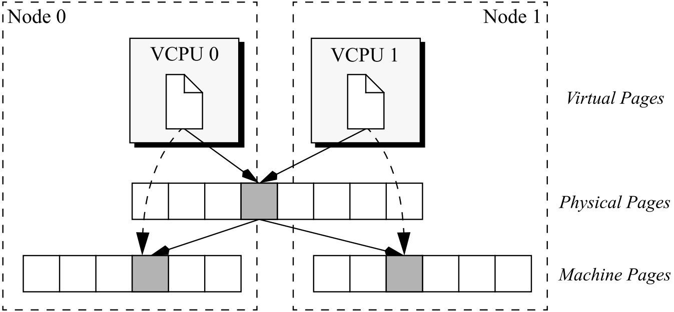
Disco

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

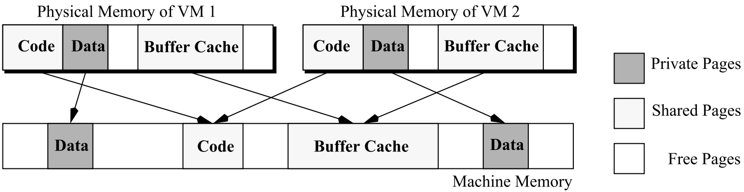
Memory Management



Source: [BDGR97]

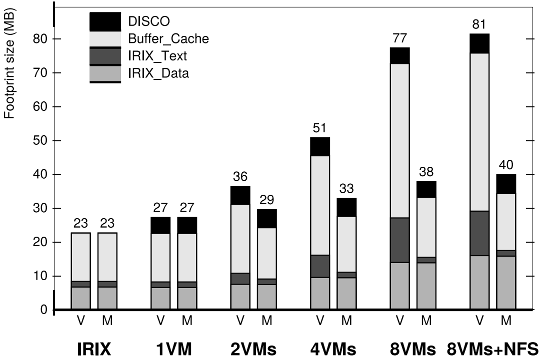
DISCO

Memory Management



Source: [BDGR97]

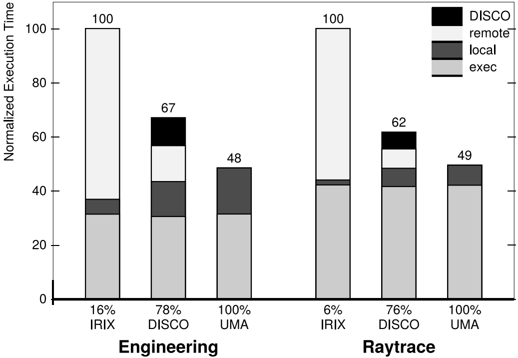
DISCO
Memory Management



Source: [BDGR97]

DISCO

Memory Management



Source: [BDGR97]

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

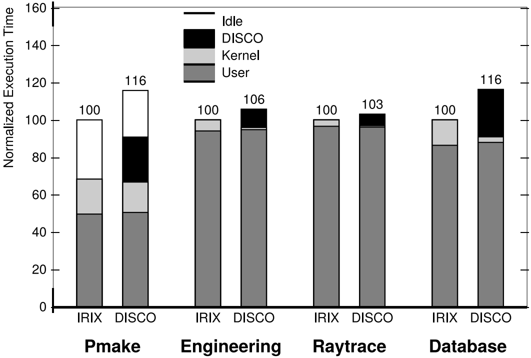
DISCO

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?

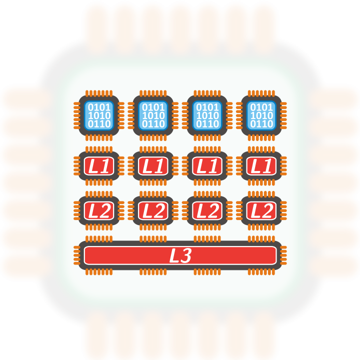
- hardware diversity is increasing (GPUs, FPGAs, programmable controllers)
- need for specific hardware optimizations
- more than one instruction set
- cache coherency is given up (for instance between CPU and GPU/NICs)

Present Age Research

Challenges of Heterogeneous Hardware

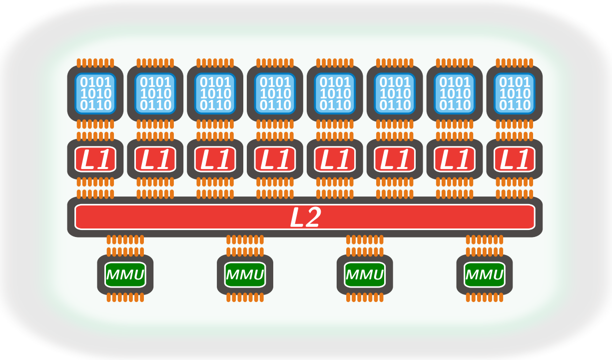
- moving a process between different cores is difficult
- lack of cache coherence
- execution time depends on the core

PRESENT AGE RESEARCH
Example - Multicore



Present Age Research

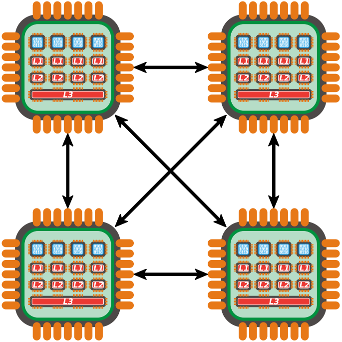
Example - Multicore



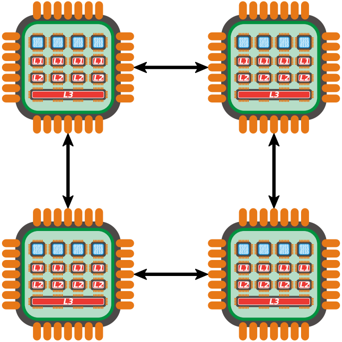
TECHNISCHE UNIVERSITÄT DRESDEN

Present Age Research

Interconnection

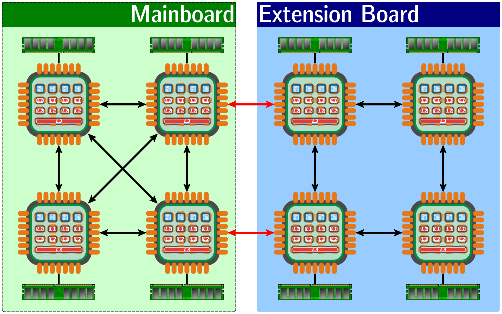


PRESENT AGE RESEARCH
Interconnection



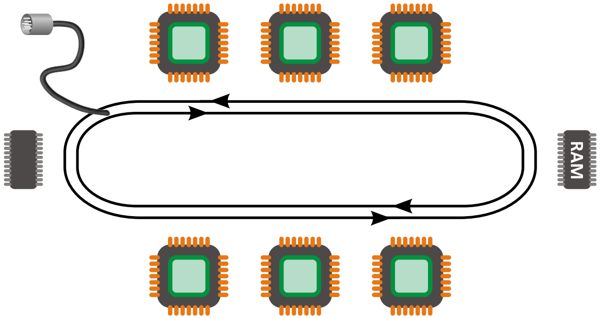
Present Age Research

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

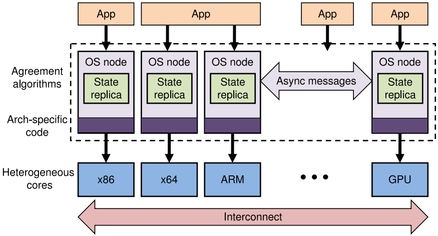
Idea

- 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

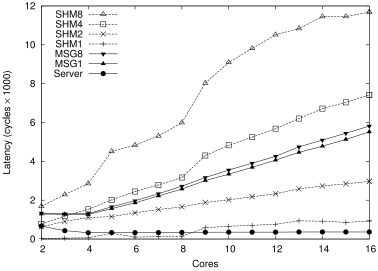
- make all inter-core communication explicit
- make OS structure hardware-neutral
- view state as replicated instead of shared



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

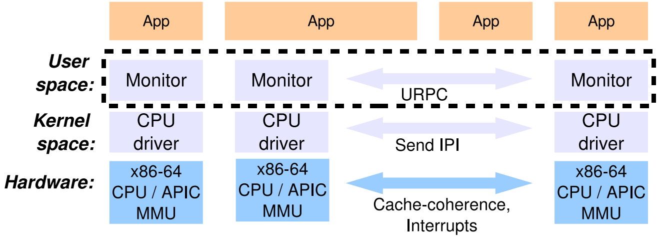
Communication

- 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]

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

The Multikernel
Implementation Details

Memory Management

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

The Multikernel
Implementation Details

Inter-core communication

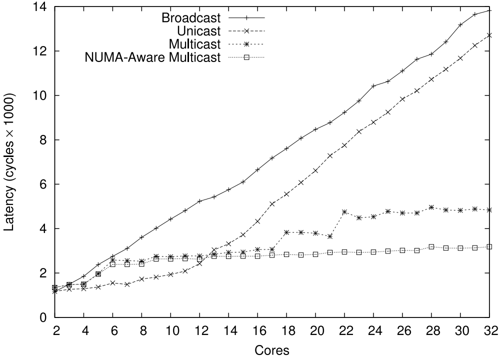
- critical for a multikernel
- using message passing
- Barrelfish uses a user-level RPC mechanism
- memory regions are shared to transfer cache-line sized messages
- receiving uses polling

THE MULTIKERNEL
Implementation Details

Hardware Knowledge Base

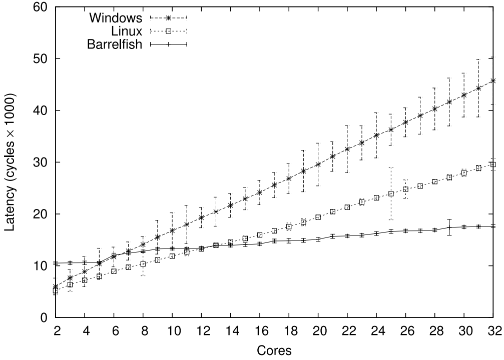
- information about the underlaying hardware
- used for optimization

THE MULTIKERNEL
Evaluation - TLB Shutdown



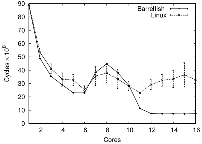
Source: [BBD+09]

THE MULTIKERNEL
Evaluation - Memory Unmap

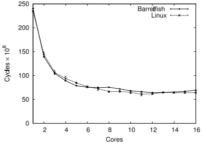


Source: [BBD+09]

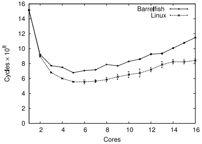
THE MULTIKERNEL
Evaluation - Compute-bound Workloads



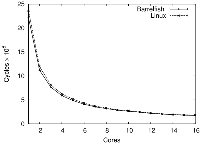
(a) OpenMP conjugate gradient (CG)



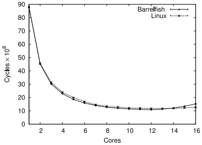
(b) OpenMP 3D fast Fourier transform (FT)



(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

- 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

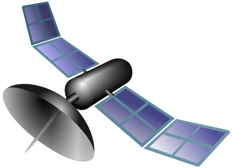
HELIOS
Satellite Kernels

- 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

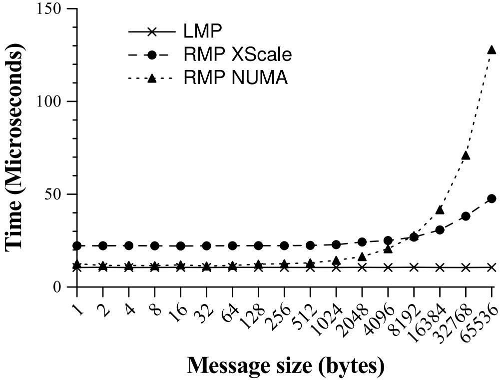


HELIOS

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



Source: [NHM+09]

HELIOS

Ressource Management

- 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
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
- 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

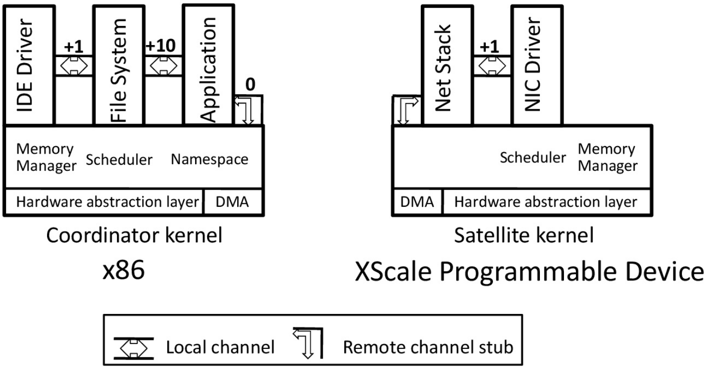
Placement of Applications

- 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

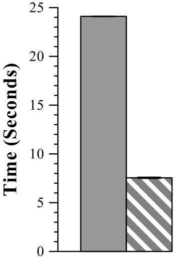
HELIOS
Placement Example



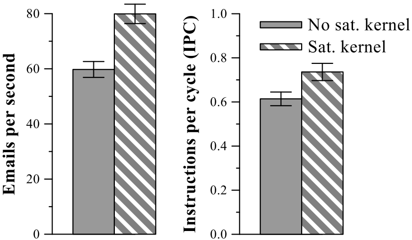
Source: [NHM+09]

HELIOS

Evaluation - Benefits of Performance Isolation



Scheduling



Mail Server

Source: [NHM+09]

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.

  In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 29–44, New York, NY, USA, 2009. ACM.

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

  Disco: Running commodity operating systems on scalable multiprocessors.

  ACM Trans. Comput. Syst., 15(4):412–447, November 1997.

TECHNISCHE UNIVERSITÄT DRESDEN

REFERENCES

Jeffrey Kuskin, David Ofelt, Mark Heinrich, John Heinlein, Richard Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy.
The stanford flash multiprocessor.
In *25 Years of the International Symposia on Computer Architecture (Selected Papers)*, ISCA '98, pages 485–496, New York, NY, USA, 1998. ACM.

Edmund B. Nightingale, Orion Hodson, Ross McIlroy, Chris Hawblitzel, and Galen Hunt.
Helios: Heterogeneous multiprocessing with satellite kernels.
In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 221–234, New York, NY, USA, 2009. ACM.