Cellular Disco: Resource management using virtual clusters on shared-memory multiprocessors

Kinshuk Govil, Dan Teodosiu*, Yongqiang Huang, and Mendel Rosenblum

Computer Systems Laboratory, Stanford University

* Xift, Inc., Palo Alto, CA

www-flash.stanford.edu

Motivation

- Why buy a large shared-memory machine?
  - Performance, flexibility, manageability, show-off
- These machines are not being used at their full potential
  - Operating system scalability bottlenecks
  - No fault containment support
  - Lack of scalable resource management
- Operating systems are too large to adapt

Previous approaches

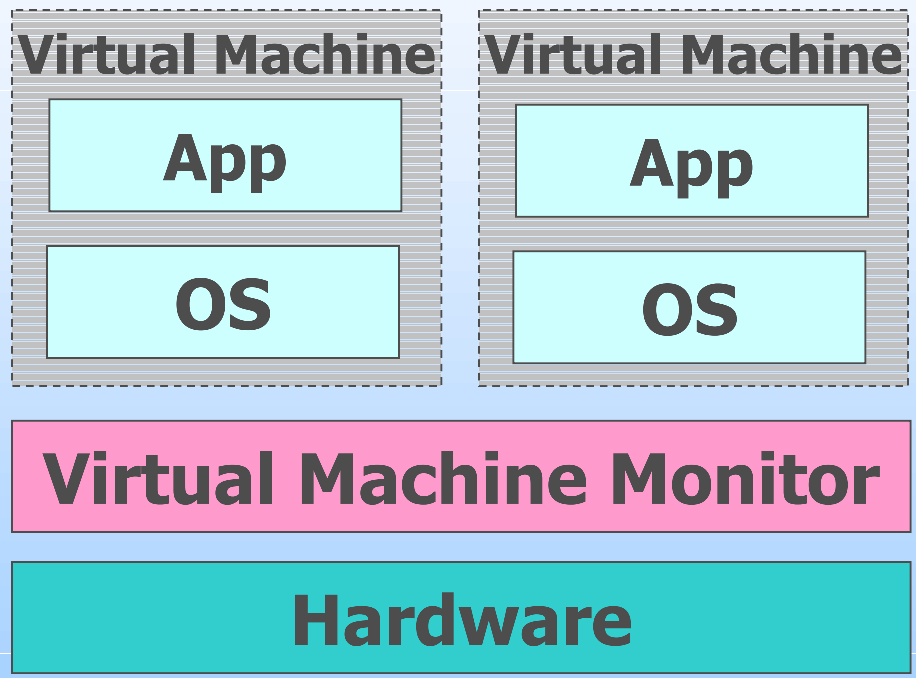
- Operating system: Hive, SGI IRIX 6.4, 6.5
  - \+ Knowledge of application resource needs
  - – Huge implementation cost (a few million lines)
- Hardware: static and dynamic partitioning
  - \+ Cluster-like (fault containment)
  - – Inefficient, granularity, OS changes, large apps
- Virtual machine monitor: Disco
  - \+ Low implementation cost (13K lines of code)
  - – Cost of virtualization

Questions

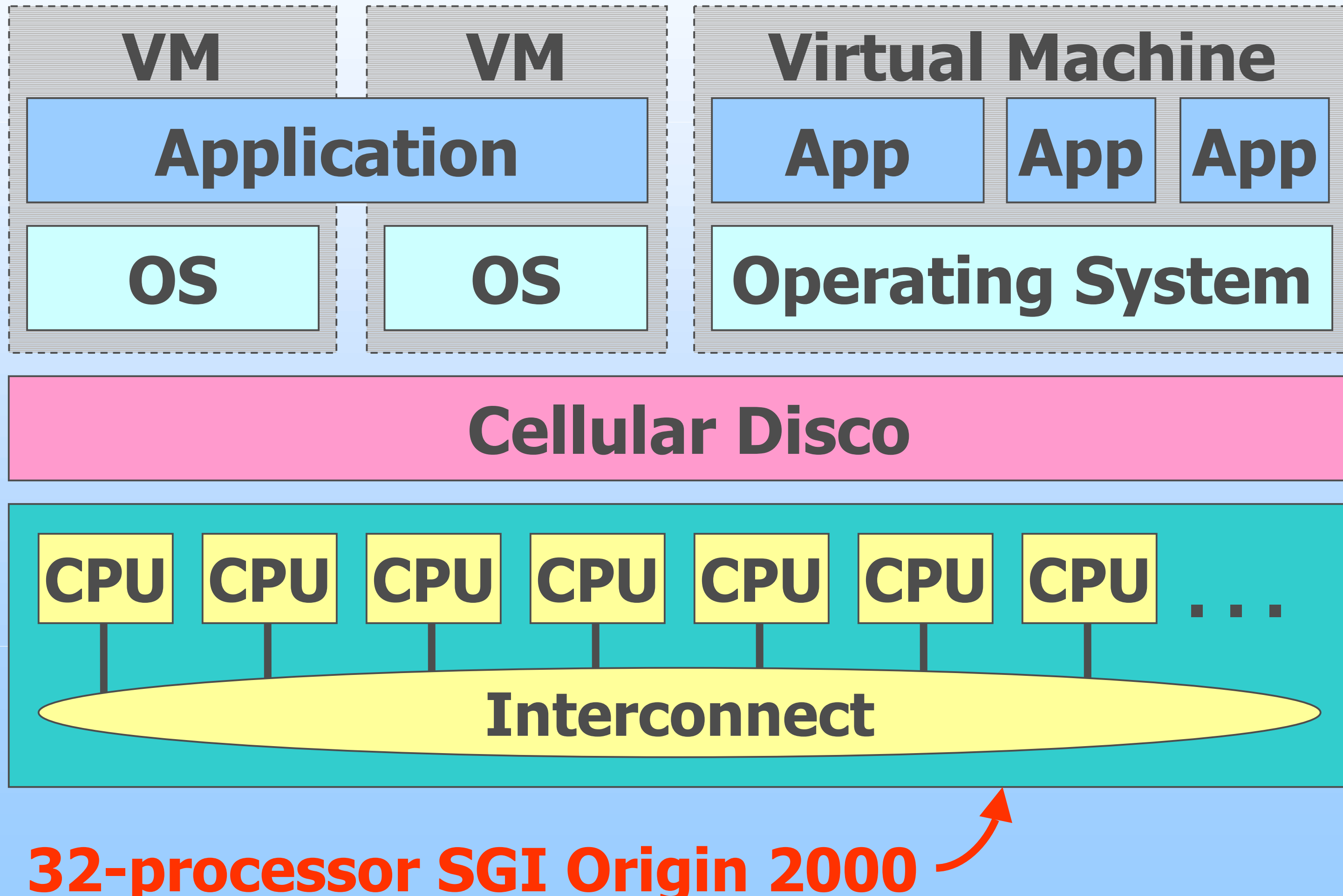
- Can virtualization overhead be kept low?
  - Usually within 10%
- Can fault containment overhead be kept low?
  - In the noise
- Can a virtual machine monitor manage resources as well as an operating system?
  - Yes

Overview of virtual machines

- IBM 1960s
- Trap privileged instructions
- Physical to machine address mapping
- No/minor OS modifications



Avoiding OS scalability bottlenecks

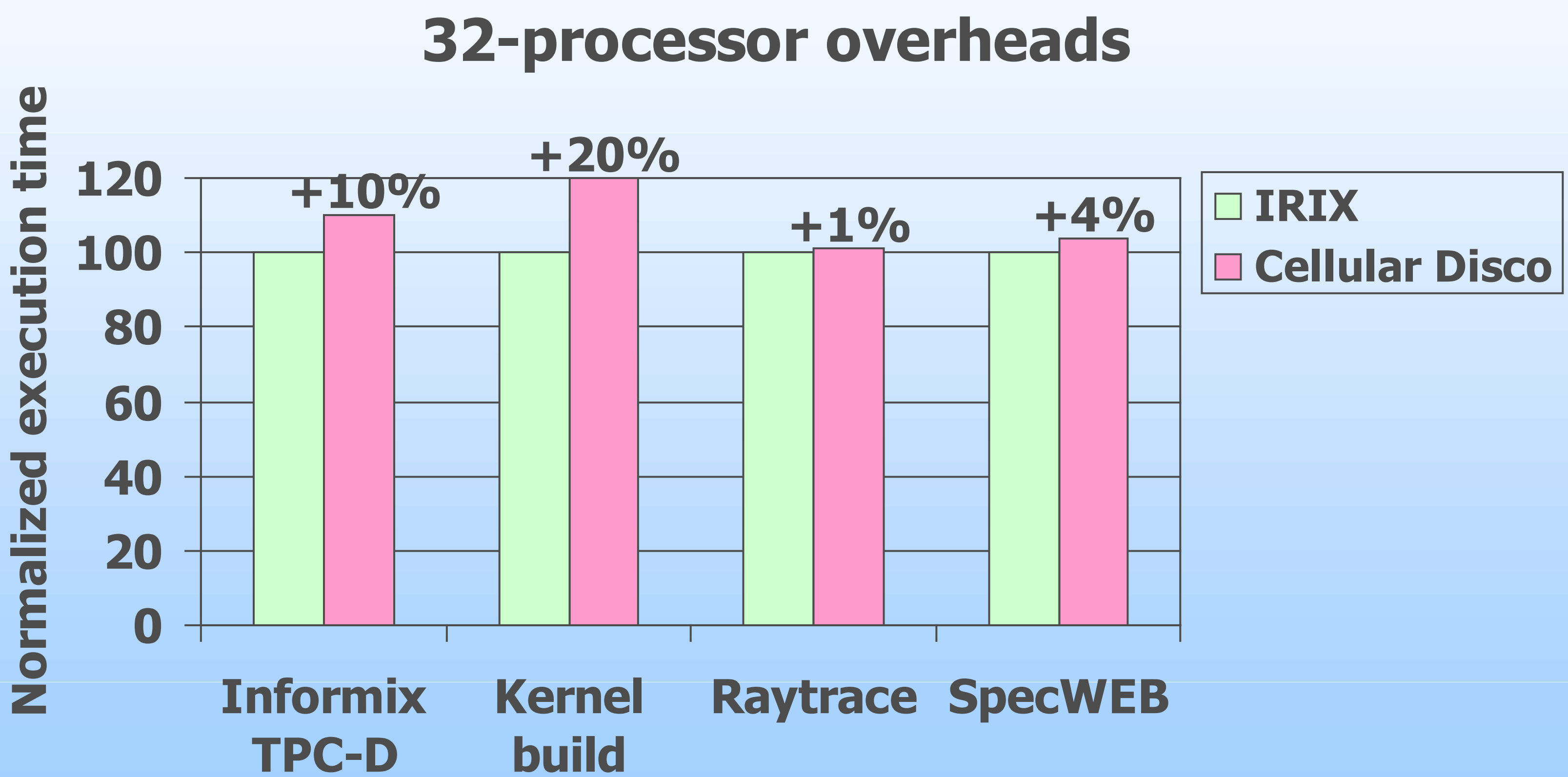


Experimental setup

IRIX 6.2	vs.	IRIX 6.4
Cellular Disco		
32P Origin 2000		32P Origin 2000

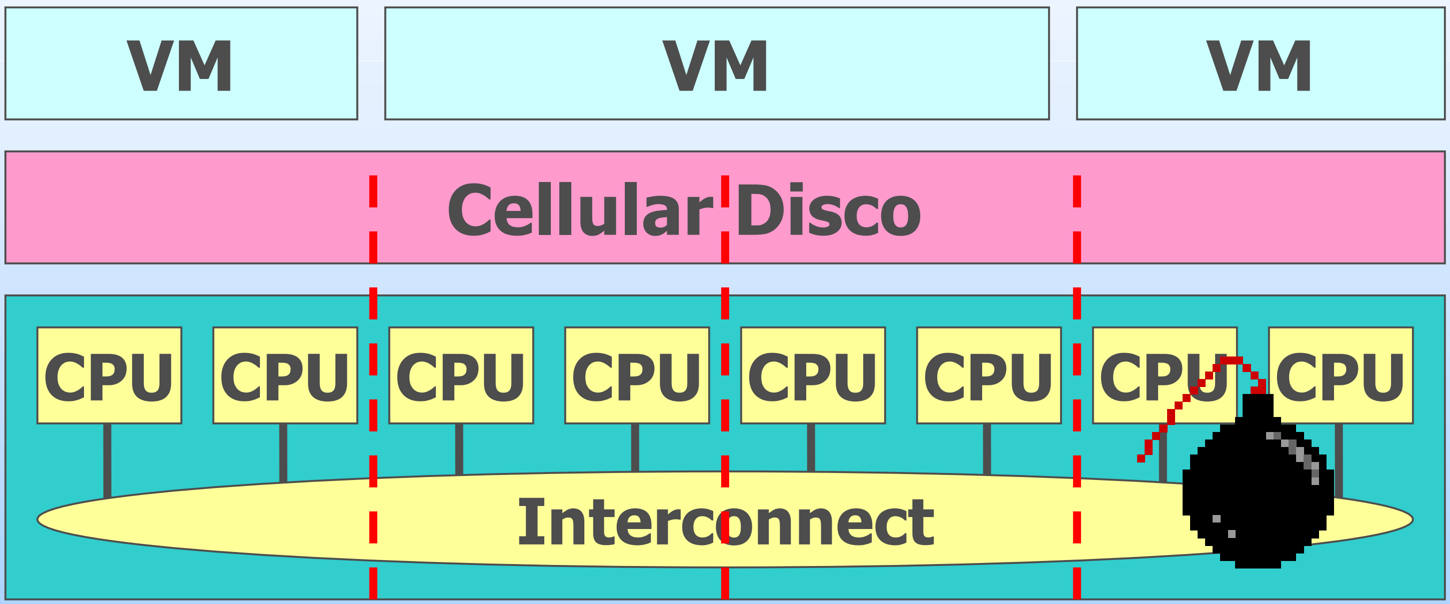
- Workloads
  - Informix TPC-D (Decision support database)
  - Kernel build (parallel compilation of IRIX5.3)
  - Raytrace (from Stanford Splash suite)
  - SpecWEB (Apache web server)

MP virtualization overheads



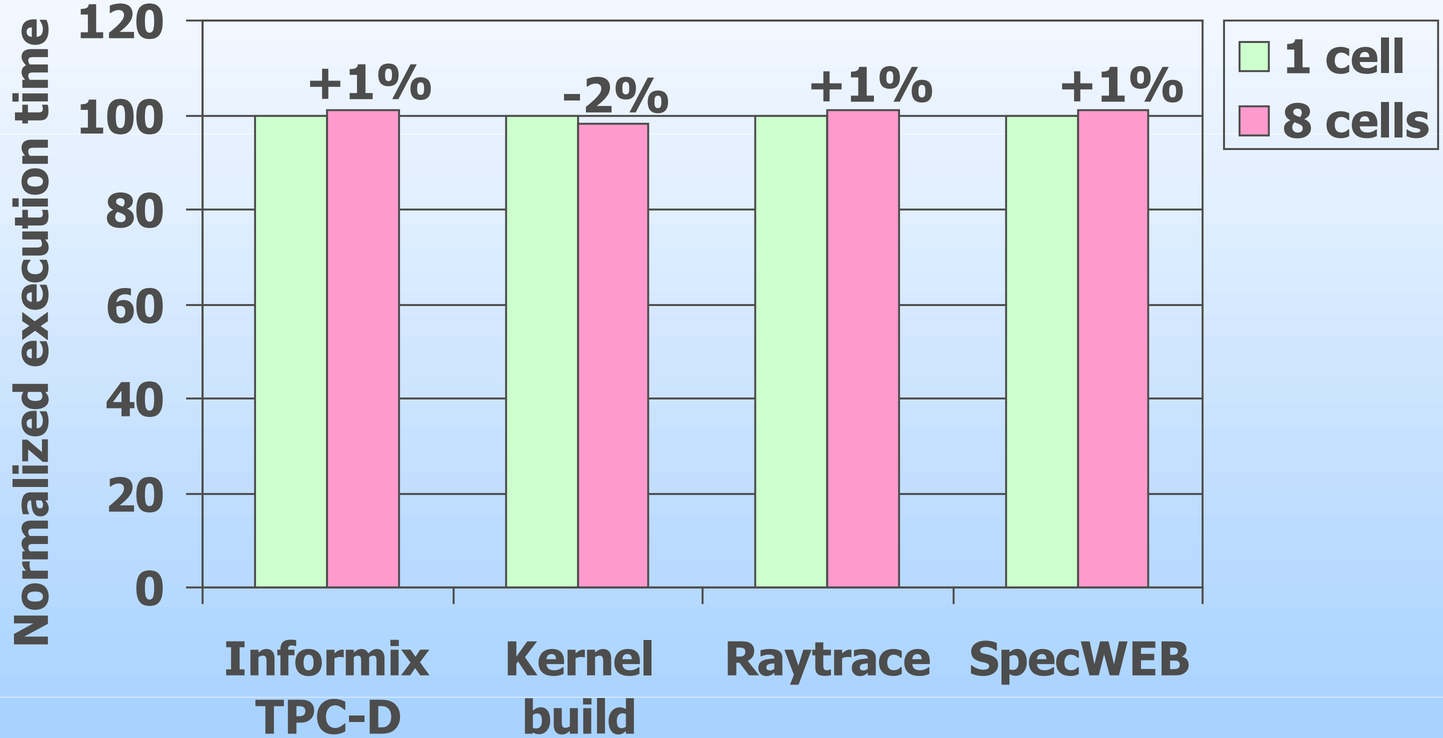
- Worst case uniprocessor overhead only 9%

Fault containment



- Requires hardware support as designed in FLASH multiprocessor

Fault containment overhead ≅ 0%

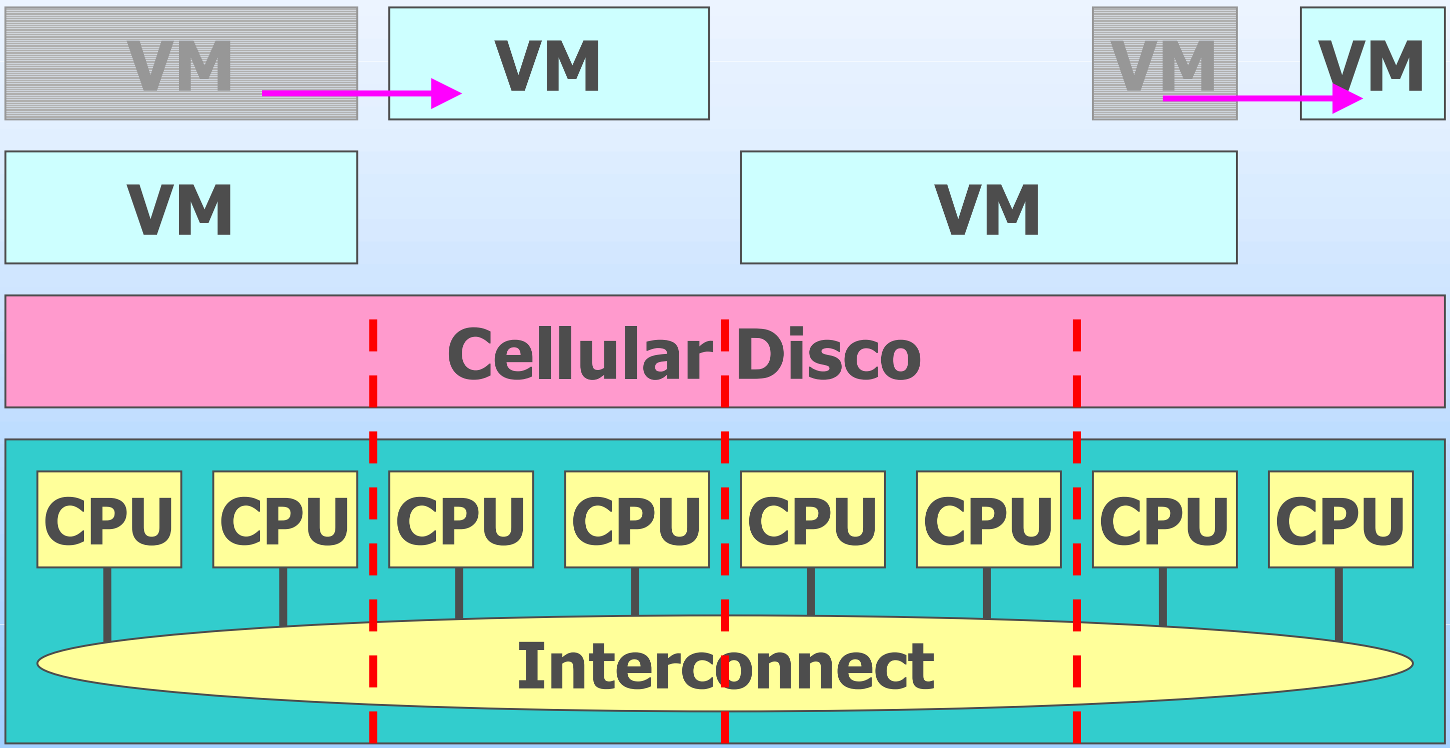


- 1000 fault injection experiments (SimOS): 100% success

Resource management challenges

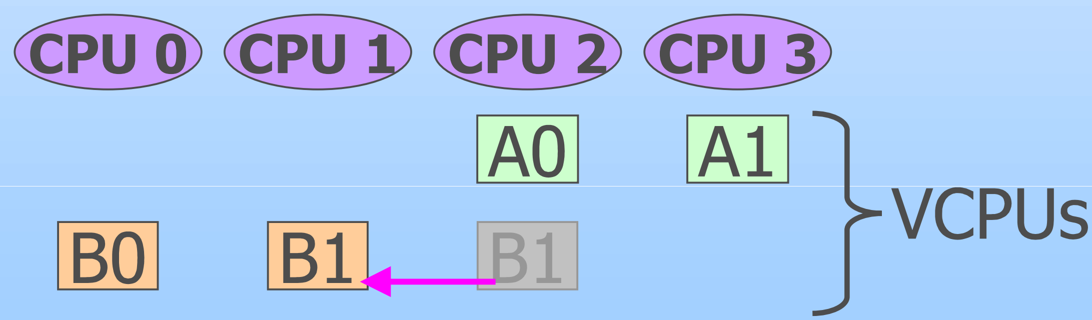
- Conflicting constraints
  - Fault containment
  - Resource load balancing
- Scalability
- Decentralized control
- Migrate VMs without OS support

CPU load balancing



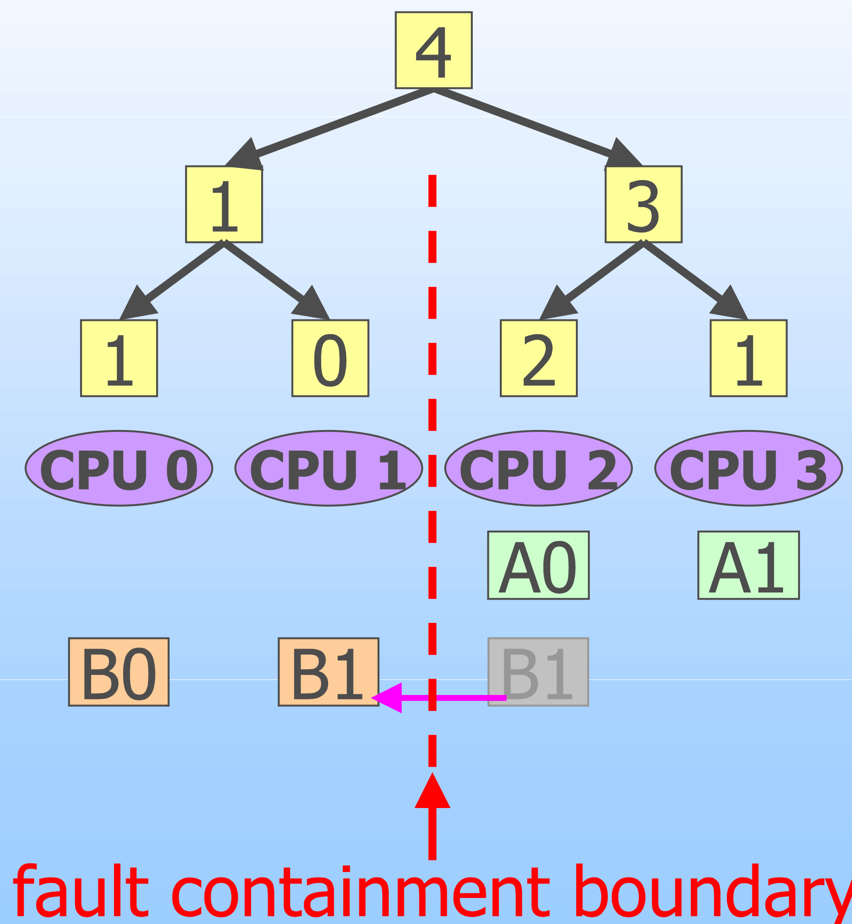
Idle balancer (local view)

- Check neighboring run queues (intra-cell only)
- VCPU migration cost: 37µs to 1.5ms
  - Cache and node memory affinity: > 8 ms
- Backoff
- Fast, local

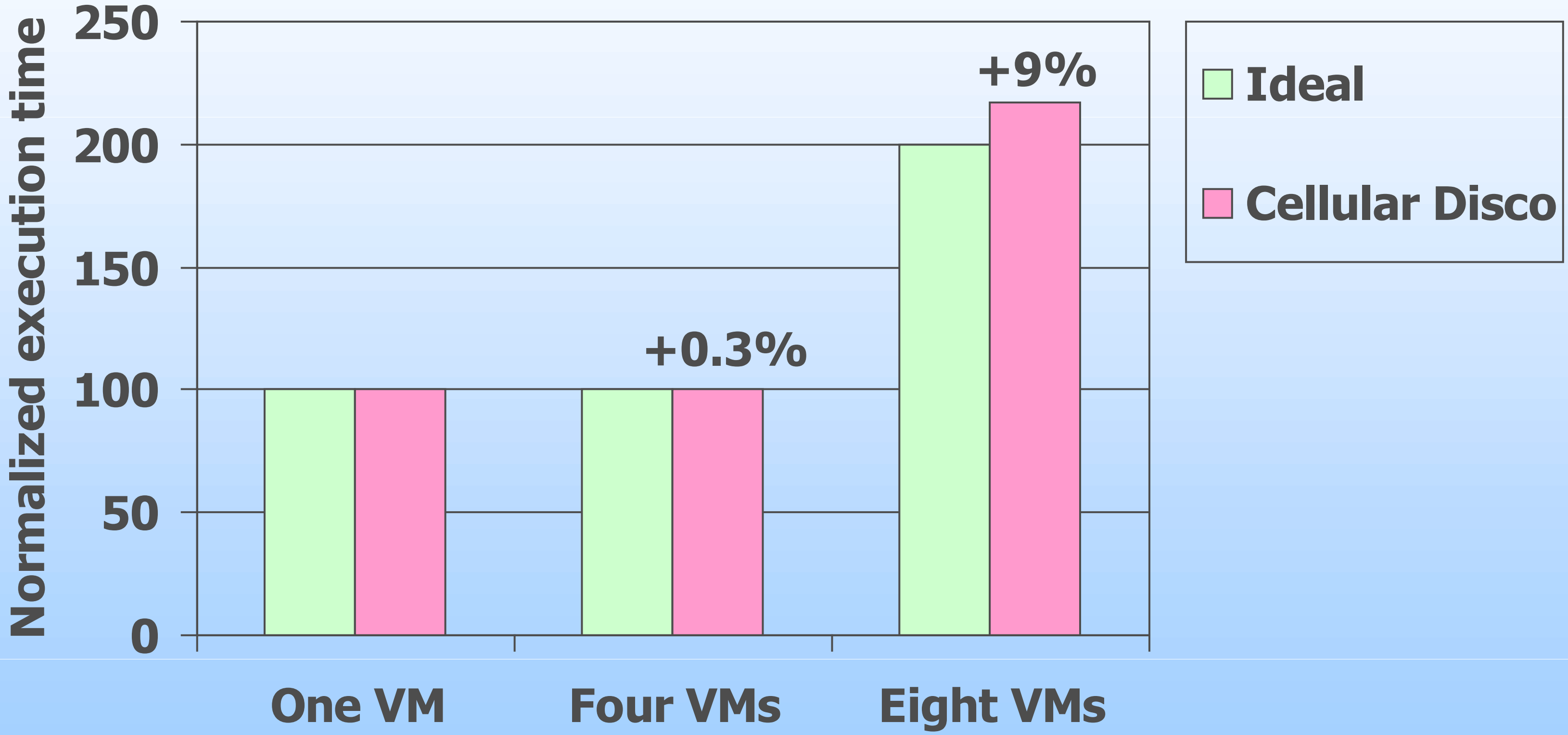


Periodic balancer (global view)

- Check for disparity in load tree
- Cost
  - Affinity loss
  - Fault dependencies

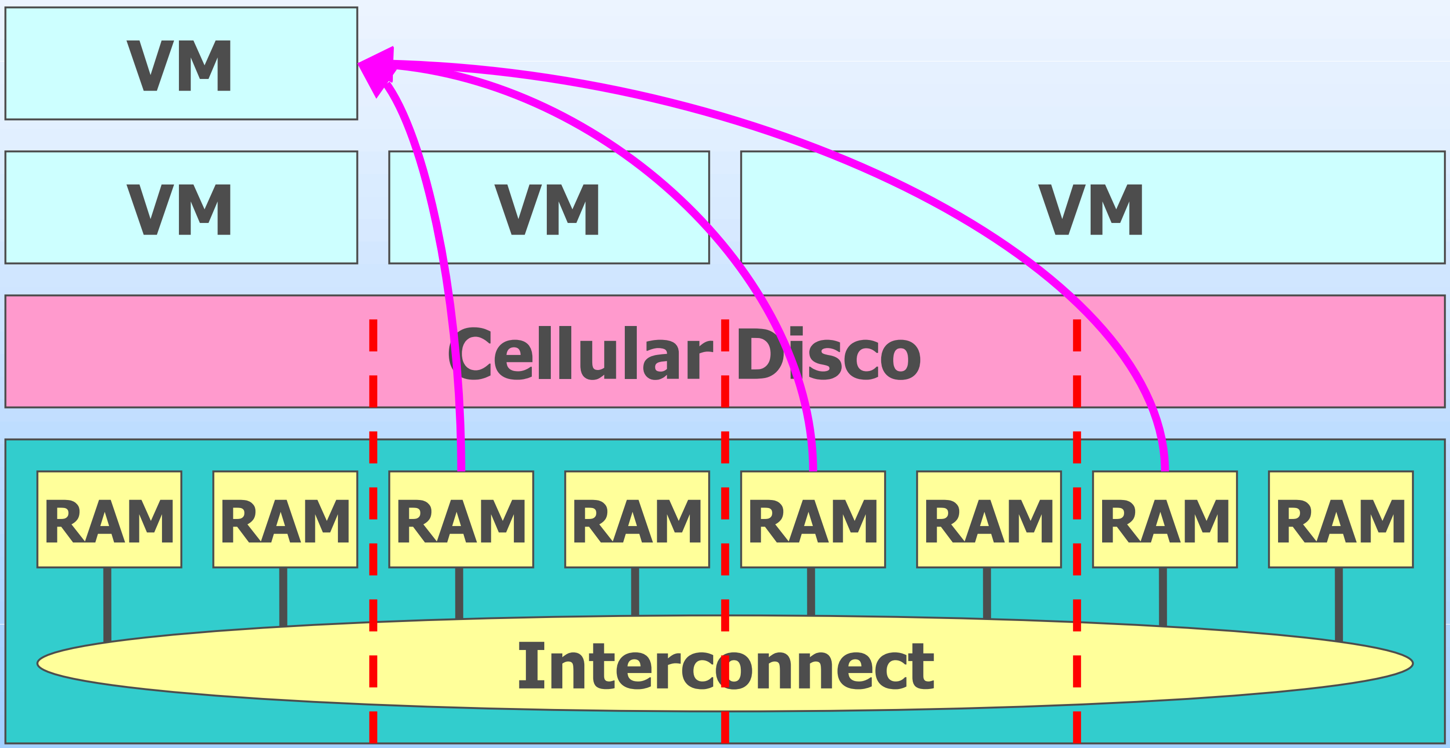


CPU management results



- IRIX overhead (13%) is higher

Memory load balancing

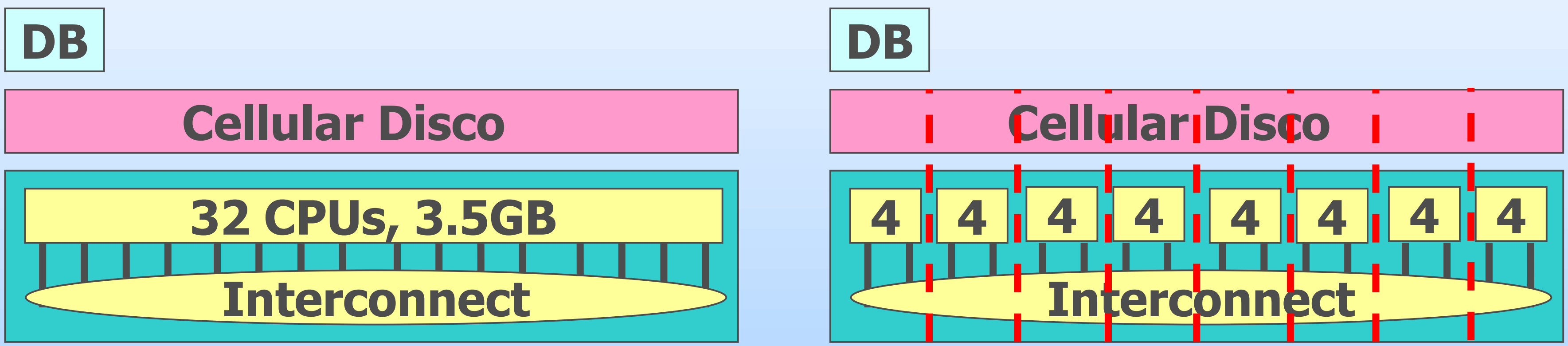


Memory load balancing policy

- Borrow memory before running out
- Allocation preferences for each VM
- Borrow based on:
  - Combined allocation preferences of VMs
  - Memory availability on other cells
  - Memory usage
- Loan when enough memory available

Memory management results

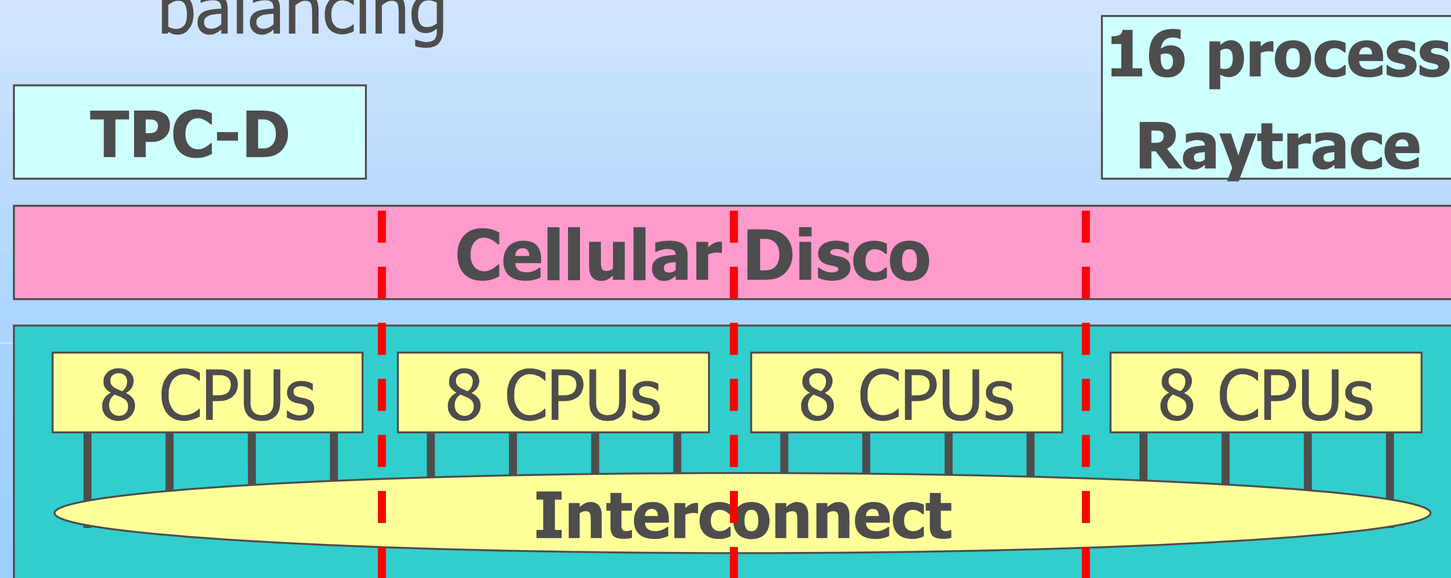
Only +1% overhead



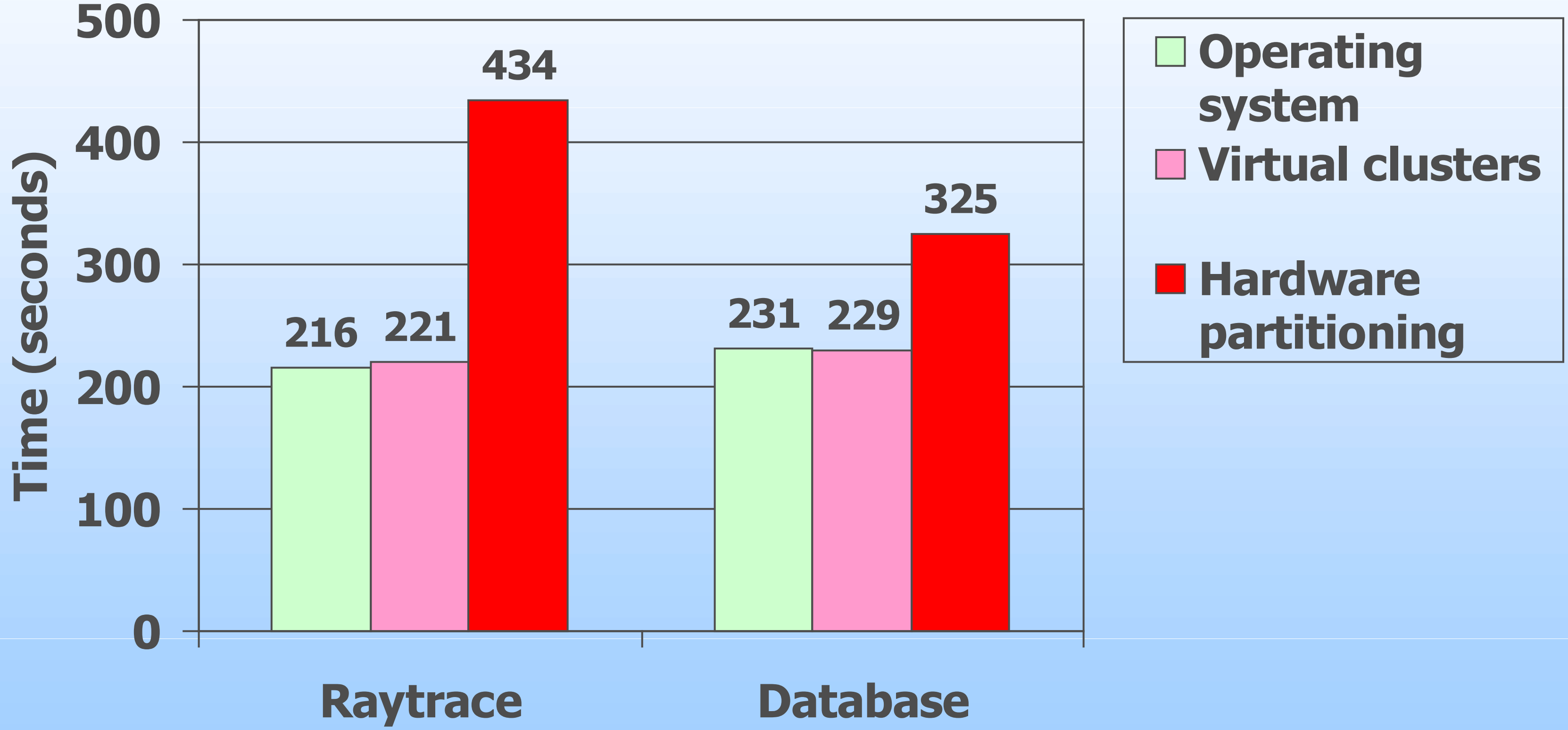
- Ideally: same time if perfect memory balancing

Comparison to related work

- Operating system (IRIX6.4)
- Hardware partitioning
  - Simulated by disabling inter-cell resource balancing



Results of comparison



- CPU utilization: 31% (HW) vs. 58% (VC)

Conclusions

- Virtual machine approach adds flexibility to system at a low development cost
- Virtual clusters address the needs of large shared-memory multiprocessors
  - Avoid operating system scalability bottlenecks
  - Support fault containment
  - Provide scalable resource management
  - Small overheads and low implementation cost