Research Directions in Operating Systems

Brian Bershad

University of Washington

Dept. of Computer Science and Engineering

Seattle, WA

Composition Technology

- Goal: high performance operating systems
- Support HPCC application requirements
  - parallel processing
  - real time and multimedia
  - distributed processing
  - filing and I/O
- Speed, Speed, Speed
- Flexible, Flexible, Flexible

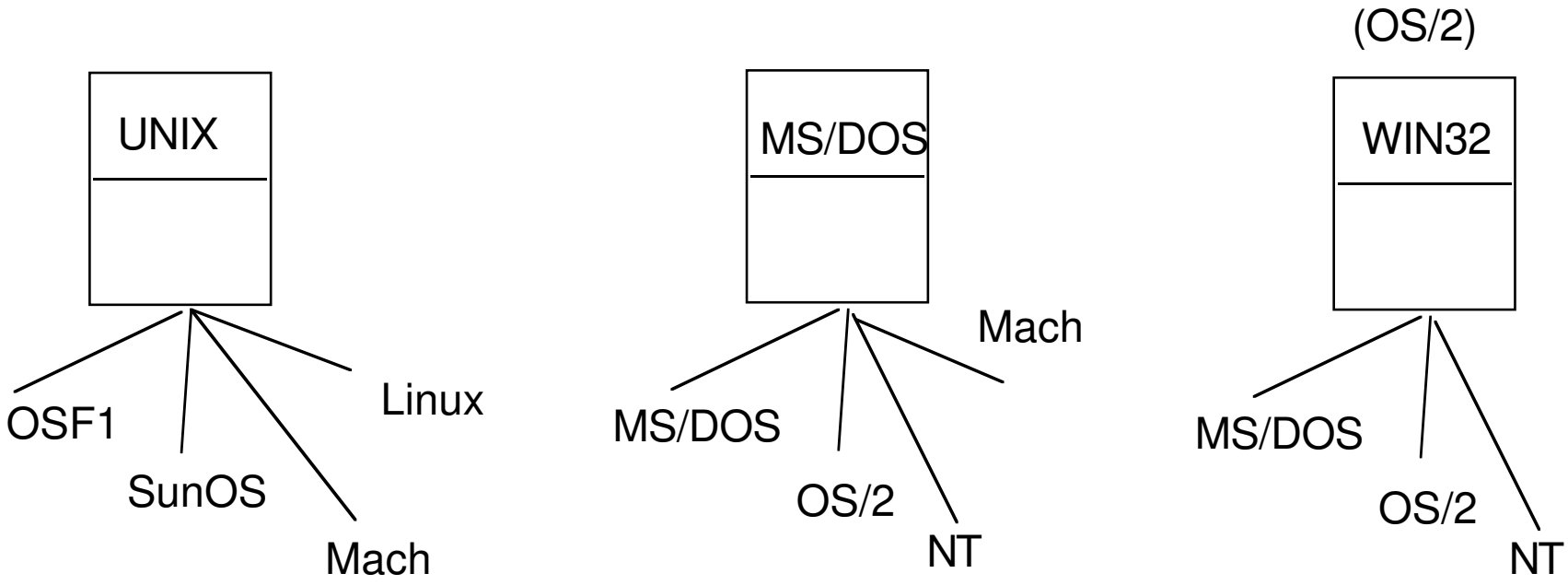
A brief survey of results and impact

- Most major system vendors are selling an OS based on academic research
  - Sun OS came from UC Berkeley
  - Univel (Novell) came from Bell Labs
  - DEC's OSF1 came from OSF/1 came from Carnegie Mellon
  - Microsoft's NT derives borrows from Carnegie Mellon
- Even the "little" companies are benefitting
  - Tera computer OS borrows from University of Washington
  - EO networking comes from the University of Arizona

The Battle of the "API" Has Ended

- The winner is... several
  - Solaris
  - SysV
  - OSF
  - NT
  - WIN-32

The Battle is now in the Implementation



Like what happened in programming languages

- Lots of competing programming languages
  - C, Algol, ADA, Pascal, Lisp, Scheme, APL
  - Language specifications once tied to the language implementation
  - No more. Can select compiler independently of language
  - Choose the "best" compiler for the job
    - does it run on my architecture?
    - does it generate (fast,small,good) enough code?
    - can it interoperate with other languages (compilers)

Operating systems are now commodities

Implication for implementations

- OS implementation is more removed from the hardware
  - increased machine-independence
  - layered implementations
- OS size and complexity has grown

SLOW

BIG

Key Research Problem

- Composition with speed and dignity
- Leverage points are:
  - software engineering for composition technology
  - programming languages and compilers for
    - integration
    - verification
    - efficient code generation

The Underlying Themes

- Base API is fixed.
- Underlying implementation must be flexible
- Performance is key
  - Previous solutions relied on hard, fixed boundaries
  - Context switching is a serious problem
- Dynamic solutions enable better performance and composition

Some Current Research Projects

- Server co-location at the University of Utah (Flex) and OSF
- SANDBOXING at CMU and UC Berkeley
- The SPIN project at the University of Washington
- The FOX project at Carnegie Mellon University
- The SCOUT project at the University of Arizona

Link-time integration

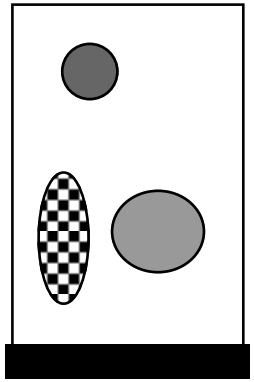
Software segmentation

Application-specific extensibility

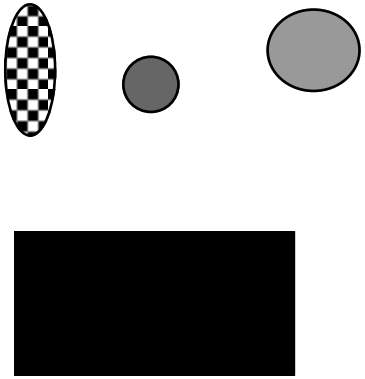
Advanced languages for OS

Resource Allocation

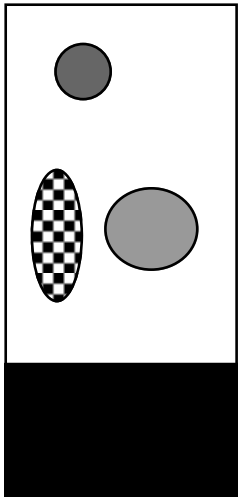
Link-time integration (University of Utah, OSF)



Integrated kernel



Microkernel and servers



The reintegrated microkernel

An example using the primary OS server

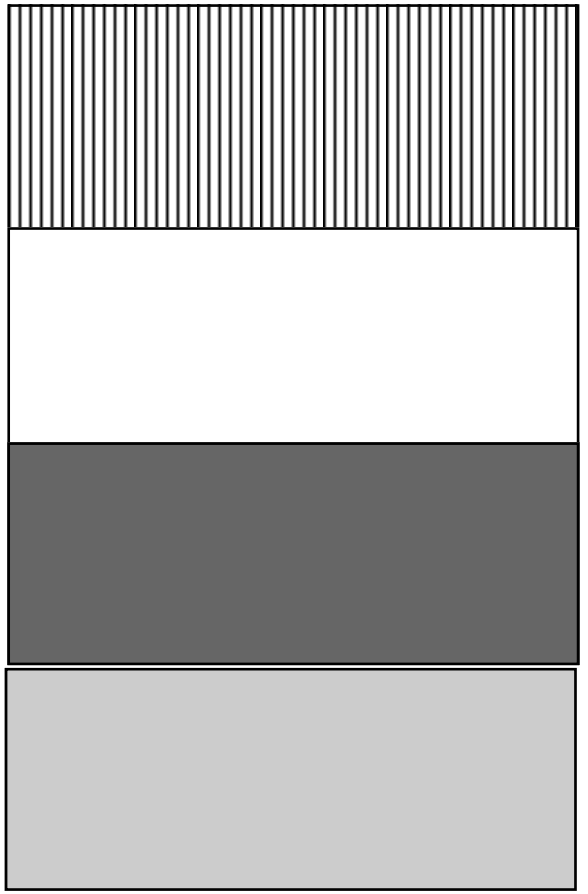
Rationale

- Easy fix for existing server infrastructure
- Legacy software, in the general case
- Can debug in a safe environment
- Can deploy in a fast environment
- System extensibility through slower paths is still possible

Software Segmentation (CMU, UCB)

- Compose multiple independently written modules into the same protection domain
- At run-time, enforce SEGMENTED ADDRESSING onto each SEGMENT.

One virtual domain



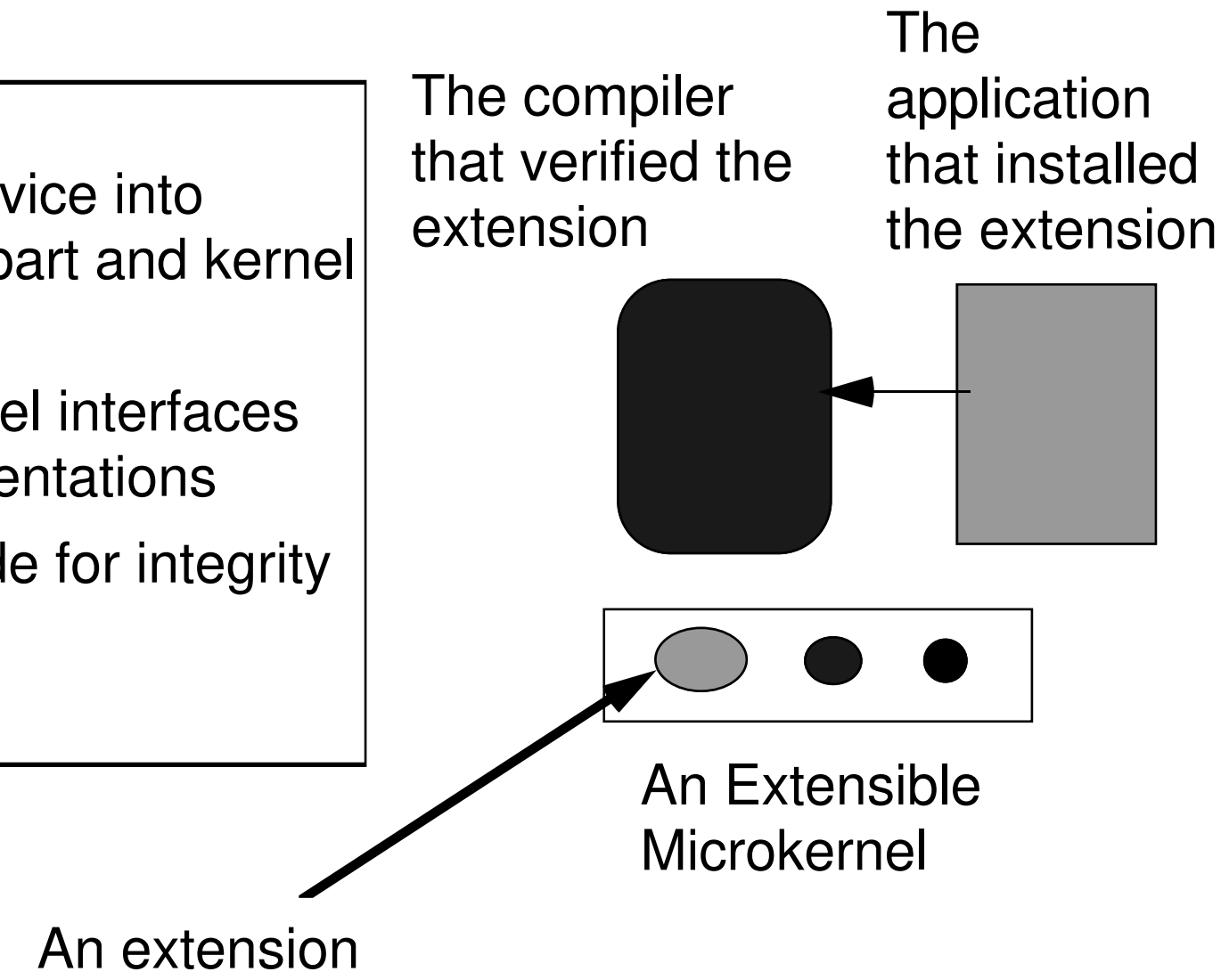
Many "virtual virtual" domains

Rationale

- Easy fix for existing server infrastructure
- Dynamic protection at low cost
- Modules do not need to be trusted

Application-specific extensibility (University of Washington)

- Partition service into application part and kernel part
- Extend kernel interfaces and implementations
- Validate code for integrity



Advanced languages for system development (CMU)

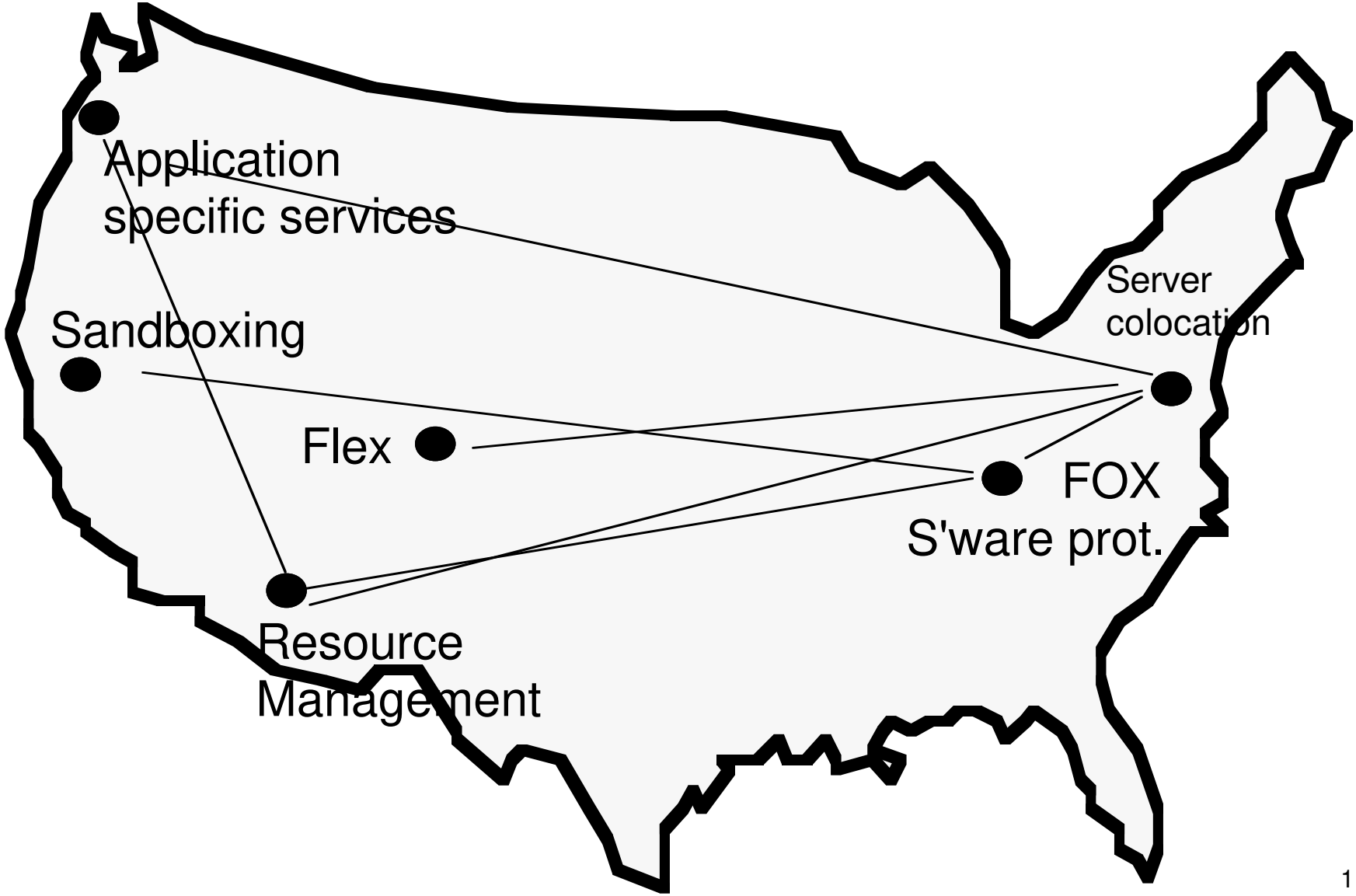
- Write the entire operating system in an advanced language
- An advanced language is one that isn't C.

ML

Rationale

- Rich module structure for composition
- Strict type checking for safety
- Mostly-functional for efficient compilation
- Theoretical foundations for correctness

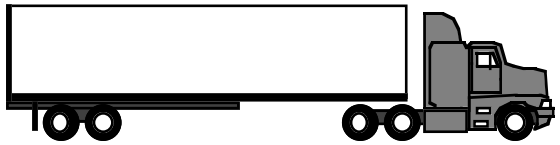
Some Current Research Projects

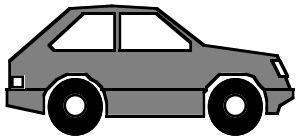


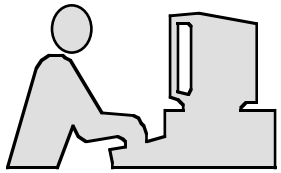
The Challenges

- compiler's aren't as good as we want
- cyclic research dependencies
- the architectures are a moving target
- application domains are rapidly changing

Expected Beneficiaries







- The big software players
  - efficient, integrable extensions
  - verification
- The little software players
  - documented entry points
  - increased programmer efficiency
- The users
  - faster systems
  - cheaper systems
- The researchers
  - more flexible research platforms
  - no performance hit
  - correctness can be ensured

Summary

- Fast flexible systems are still on the horizon.
- Use compiler/linker technology to realize goals