Future Work

- **Finish building VINO.**
  - Networking.
  - Naming.
- **Build applications that use extensions to optimize performance.**
- **Interface design.**
  - What types of extensions actually get used?
  - Revisit flexibility vs. performance trade-off.

E-mail: {chris,keith,margo,yaz}@eecs.harvard.edu

Web site: http://www.eecs.harvard.edu/~vino/vino

Conclusions

- **Possible to build extensible OS.**
- **Extensible OS is a good idea.**
- **Performance trade-off is critical.**
- **Applicable beyond field of operating systems (e.g., to web browsers).**

Performance Summary

- **100–450μs total overhead.**
  - Not cheap.
  - Negligible when savings is disk I/O.
  - Untuned implementation.
  - Not feasible for tiny performance improvement.

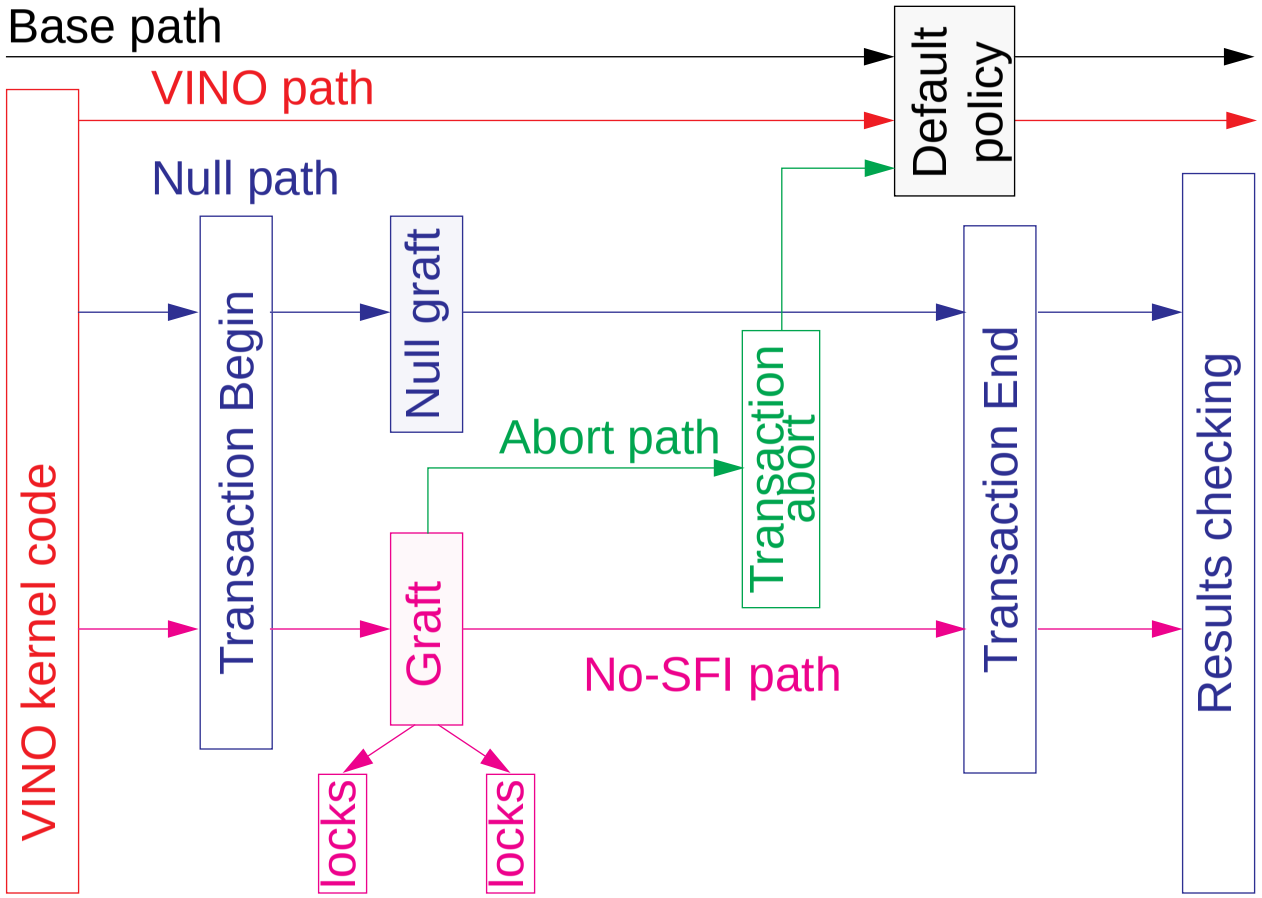
Performance Overhead

	Overhead in μs			
	RA	**VM**	**Sched**	**Encrypt**
Begin	36	52	38	32
Commit	28	34	30	32
Abort	29	27	33	36
Lock	33	34	33	0
Graft	2	160	35	166
Indir	1	1	1	0
SFI	3	26	5	187
Total	**103**	**307**	**142**	**417**

Sample Grafts

- **Measured costs on sample extensions.**
  - VM Page eviction.
    - Keep *important* pages in memory.
  - File read ahead.
    - Support non-sequential, but known access.
  - Process scheduling.
    - Allows group scheduling.
  - Data encryption.
    - Adds new functionality.
    - Filter between user and file system.

Measuring Performance



Performance

- **Allowing extensibility has costs.**
  - Extra levels of indirection.
  - Transaction overhead.
  - Validation of return value(s).
  - Cost of graft code.
  - Software fault isolation.
  - Abort cost.

Transaction Implementation

- **Extensions invoked through wrapper.**
  - Begin a transaction.
  - Switch stacks.
  - Calls extension.
  - Commits transaction.
- **State changes must be logged.**
  - State changes made by accessor methods.
  - Accessor methods write log records.
  - Log can be transient.
  - Implemented as a call stack of undo functions.
- **If extension fails, abort transaction.**
  - Jump to abort call stack.
  - Return through each “undo” function.

Transactions

- **Why?**
  - Guarantee atomicity.
  - Single mechanism to enforce consistency.
  - Generally useful tool.
  - Allows nested extension calls.
- **How?**
  - Returns kernel to pre-extension state on failure.
  - Ensures that other threads do not depend on interim extension state.

Handling Failure

- **Remove extension from kernel.**
- **Undo changes to kernel state made by extension.**
  - Free memory.
  - Release locks.

Interface Abuse

- **Misusing legal interface functions.**
  - Fail to release locks.
  - Fail to free resources (e.g., memory).
- **Operating system must detect these problems.**
  - Time-out contested locks.
  - Resource limits.
- **Trade-off between interface flexibility and potential for abuse.**
  - Disallow locks; require lock-do-unlock interface.
  - Allow locks; support lock, do, ..., do, unlock interface.

Protecting the Kernel

- **Extension accesses forbidden memory.**
  - Software fault isolation (VINO).
  - Safe language (e.g., Java, Modula-3 [SPIN]).
- **Extension returns invalid data.**
  - Validate return values.
  - Time-out long running extensions.
- **Extension calls forbidden functions.**
  - Static check at download time.
  - Software fault isolation checks indirect jumps.
  - Check security—extensions have privileges of application that installed them.

Extensibility Challenges

- **Three interfaces between extension and kernel.**
- **All three interfaces can be abused.**

  Interface: Kernel and extension share memory.

  Problem: Extension reads/writes private kernel memory.

  Interface: Kernel calls extension.

  Problem: Extension returns invalid data (or doesn't return).

  Interface: Extension can call other kernel functions.

  Problem: Extension calls forbidden kernel functions.

VINO Implementation

- **New kernel design and implementation.**
- **Use NetBSD device drivers and locore.**
- **Object-oriented design (C++).**
- **Design for per-method extensibility.**
  - Highly (overly?) modularized.
  - Encapsulate every policy decision in a method.
  - Two extension techniques:

    Replace or extend methods.

    Specify event handler.

Extensibility in VINO

- **Working assumptions**
  - The OS frequently does ***almost*** the correct thing.
  - Often minor tweaks can fix major problems.
  - Minimize effort to modify kernel behavior.
- **Design principles**
  - Extensibility should be fine-grain (e.g., function call).
  - Extensions should look just like kernel code.
  - Extensions should be able to call kernel functions.

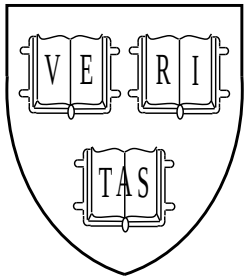
Why Extensibility?

- **Systems optimize for the common case.**
- **Some important cases are uncommon.**
- **Phenomenon appears in many places.**
  - Database servers.
    - Download queries.
    - Download new data types.
  - Web browsers.
    - Download applets.
  - Operating systems.
    - Download drivers.
    - Download entire subsystems.
    - Download minor modifications.

Outline

- **Why extensibility?**
- **Extensibility in VINO.**
- **Challenges in extensibility.**
- **Performance.**

Dealing with Disaster: Surviving Misbehaving Kernel Extensions



Margo Seltzer, Yasuhiro Endo, Chris Small, and Keith Smith

Harvard University
Division of Engineering and Applied Sciences

October 31, 1996