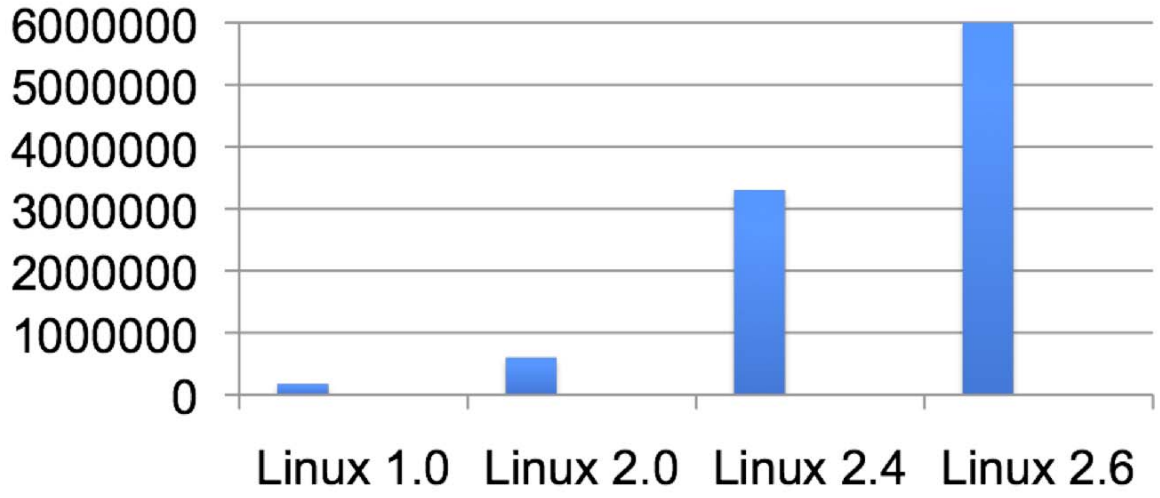
# Hardware Enforcement of Application Security Policies using Tagged Memory

Nickolai Zeldovich^, ***Hari Kannan****, Michael Dalton*, Christos Kozyrakis*

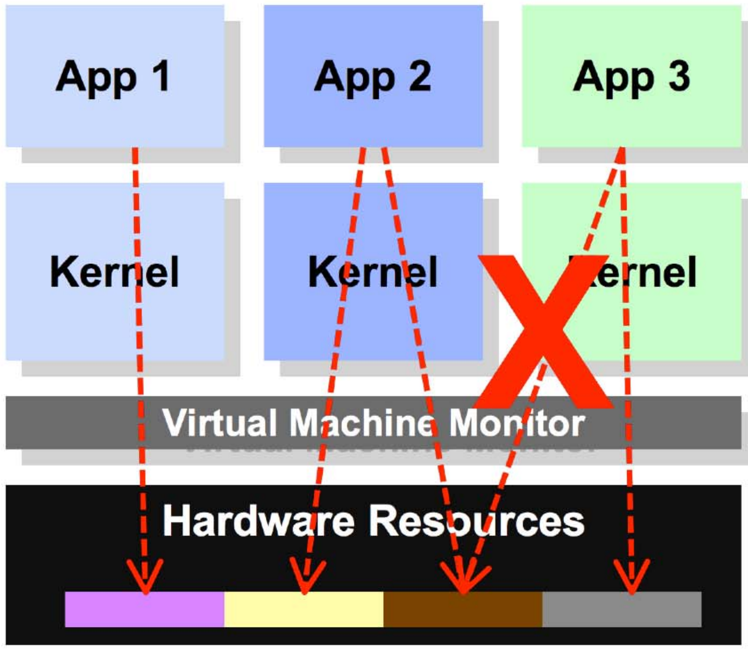
^MIT *Stanford University

# Motivation

- Systems do not provide good abstractions for security
  - Applications forced to build own security mechanisms
  - Software managing security bloats TCB greatly
- TCB for large SW systems is in the millions of LOC
  - Written by multiple developers in different companies
  - Difficult to eliminate bugs or verify correctness



# VMs don't solve the problem




- ❑ Virtualization used for minimizing OS TCB
- ❑ Good for partitioning resources, applications
- ❑ But apps that require sharing must run in same VM
  - Virtualization provides no benefit

# Current HW also inadequate

- Kernel data structures require fine-grained protection
  - struct proc {

```
struct proc {
    struct proc *next;      } All processes can read, but not write
    pid_t pid;
    char proctitle[64];     → Owner can read, write; others can read
    struct inode *cwd;      → Only owner can read and write
}
```

- Page-aligning data structures is complex and expensive
- Paging does not associate policy with **physical** resources
  - Translation mechanisms subject to **aliasing**

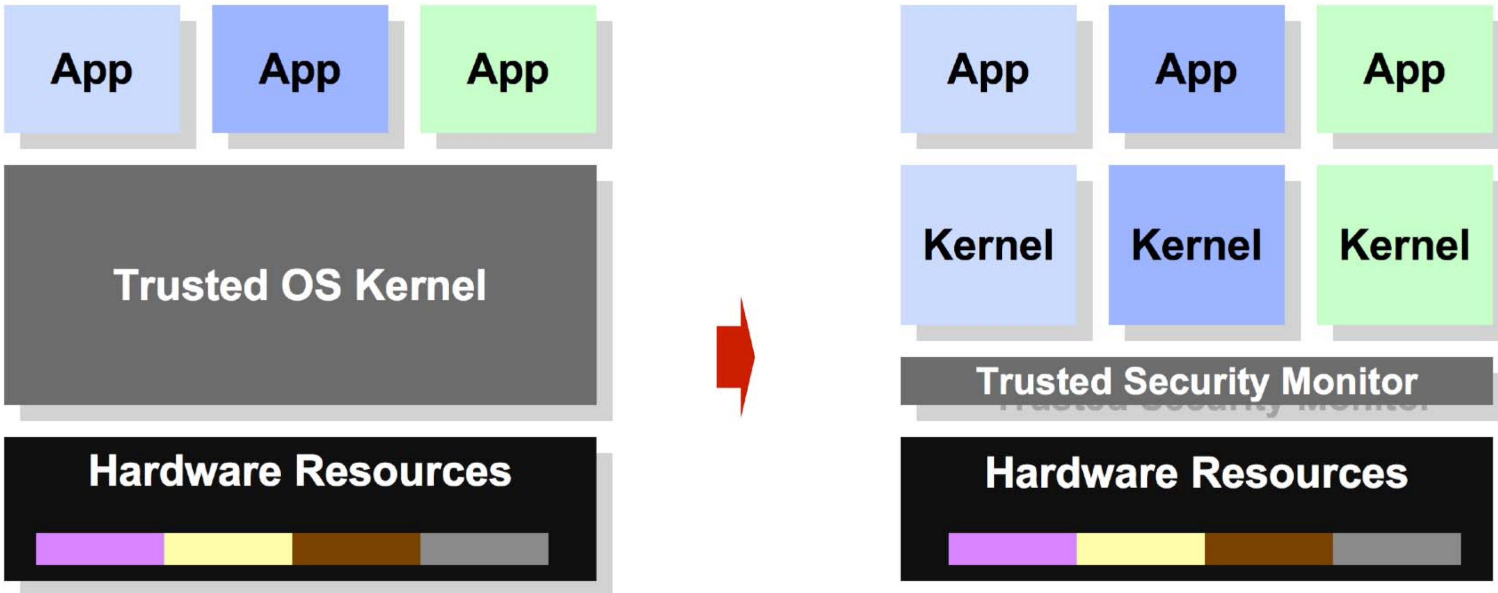
# Our proposal: Tagged Memory

- Tagged memory:
  - Each word of physical memory associated with a 32-bit tag
  - Tags map to access permissions (R/W/X) for prot. domain
  - **Fine-grained** access control
- **Simplifies** security enforcement
  - SW manages tags, but HW enforces security policies
  - Helps maintain security in face of compromised OS
- Ties security policies to **physical** resources
  - Physical resource policies avoid ambiguity
- Allows for a smaller TCB (need to trust less software)

# Outline

- ❑ Motivation & Tagged Memory Overview
- ❑ Software Architecture
  - Enforcing and Managing Security Policies with tags
- ❑ Full-system FPGA Prototype
  - Hardware Design
  - Experiments
- ❑ Conclusion

# High-level system structure



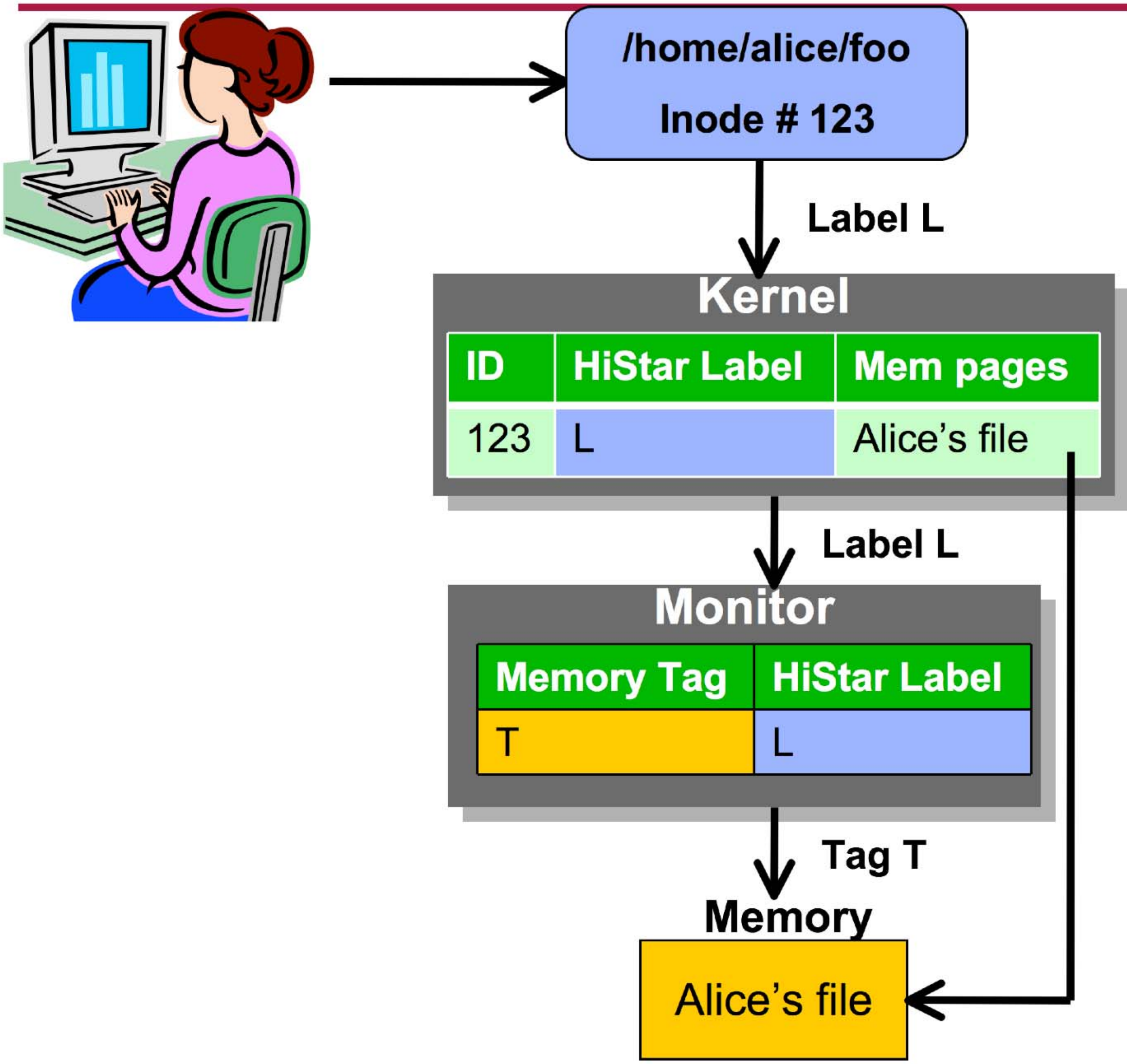

- *Complex kernel* enforces security *and* implements software abstractions

- *Monitor* labels resources
- *Hardware* enforces security
- *Kernel* provides abstractions, but no longer fully trusted

# Prototype: LoStar (Loki + HiStar)

- Based on **HiStar** OS
  - Worst-case scenario: kernel already very small, O(10,000 LOC)
  - Key feature: OS interface consists of few types of **objects**
  - Applications specify security policies in terms of **labels** on objects
  - Kernel uses labels to control **read/write access**
- **Security monitor**: translates HiStar labels into hardware tags
  - Maintains separate tag permissions for every executing context
  - Controls access to shared memory using tags
  - Protects important kernel data structures
- **Loki**: tags for fine-grained permissions on physical memory
  - Tag permission cache in hardware populated by monitor

# LoStar enforces application policies




**Application security policy enforced in hardware**

# Controlled sharing in LoStar

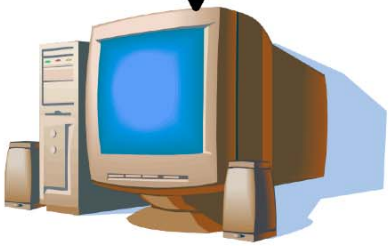
Alice'
s
thread

Bob's
thread

Alice private

Alice
public_html

Alice & Bob
Shared



# Controlled sharing in LoStar

| Tags | Perm |
|---|---|
| | RW |
| | RW |
| | RW |

| Tags | Perm |
|---|---|
| | -- |
| | -- |
| | RW |

Kernel

Kernel

Monitor

Alice private

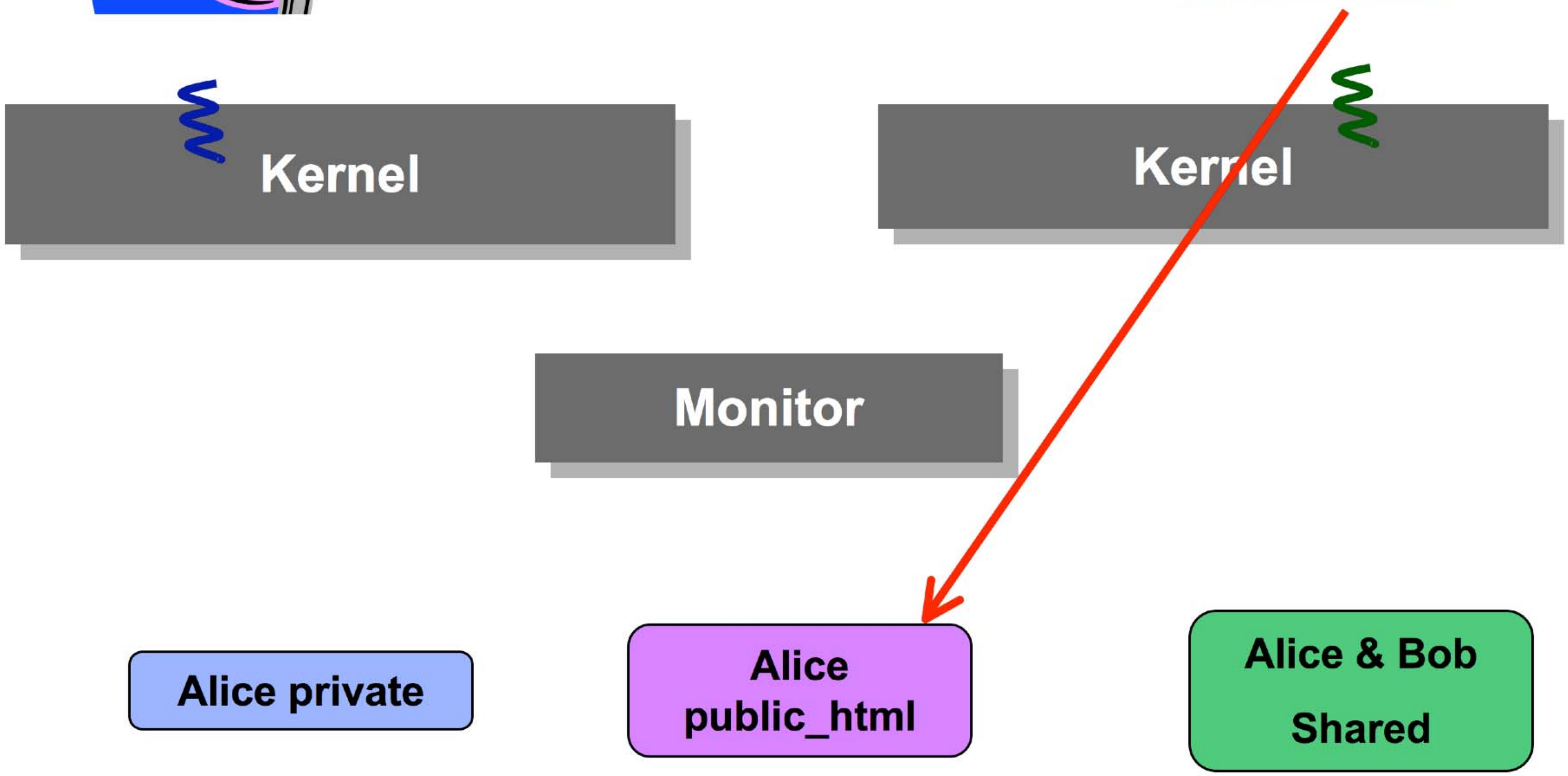
Alice public_html

Alice & Bob Shared

# Controlled sharing in LoStar

| Tags | Perm |
|---|---|
| | RW |
| | RW |
| | RW |

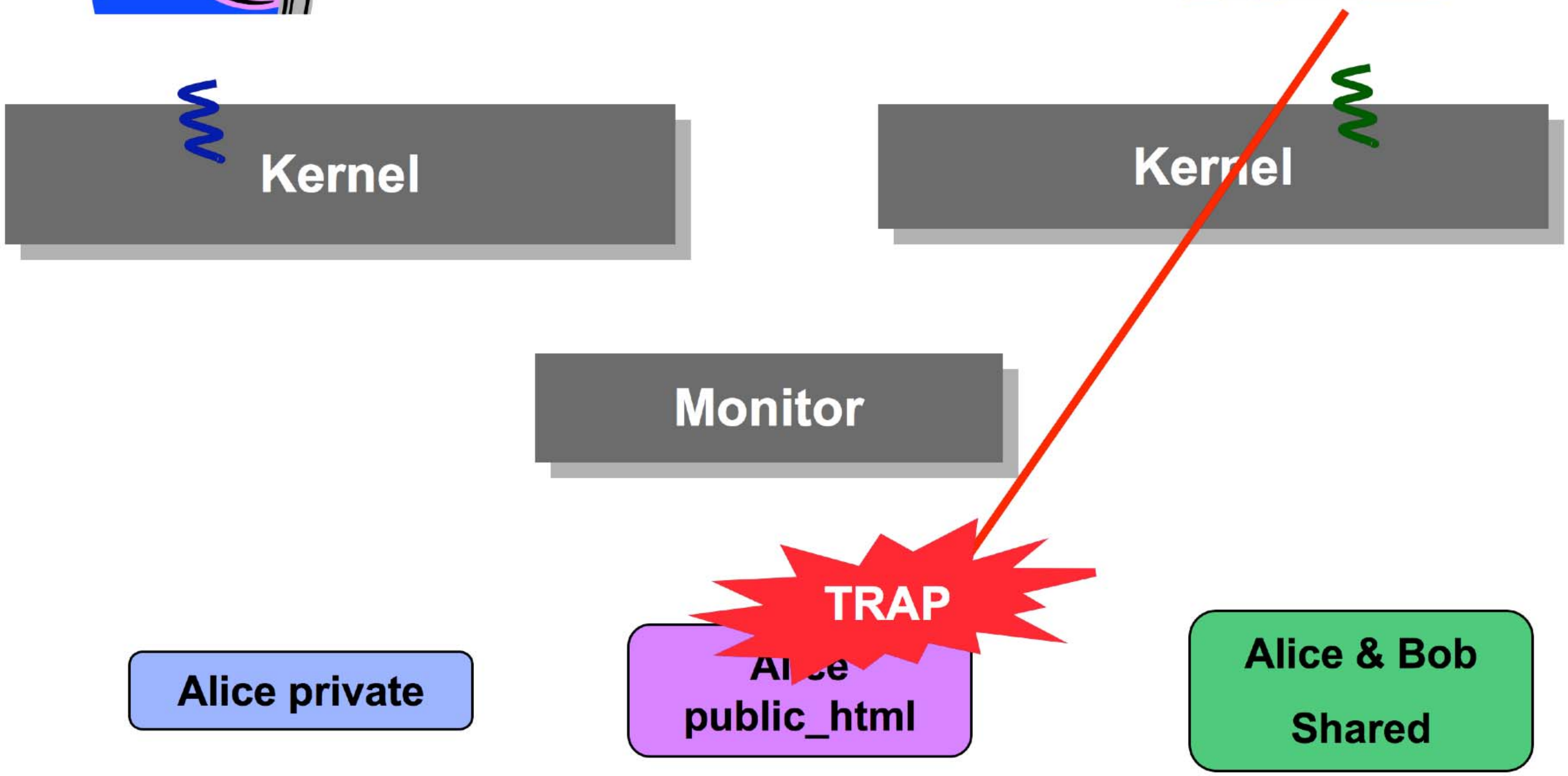
| Tags | Perm |
|---|---|
| | -- |
| | -- |
| | RW |



# Controlled sharing in LoStar

| Tags | Perm |
|---|---|
| | RW |
| | RW |
| | RW |

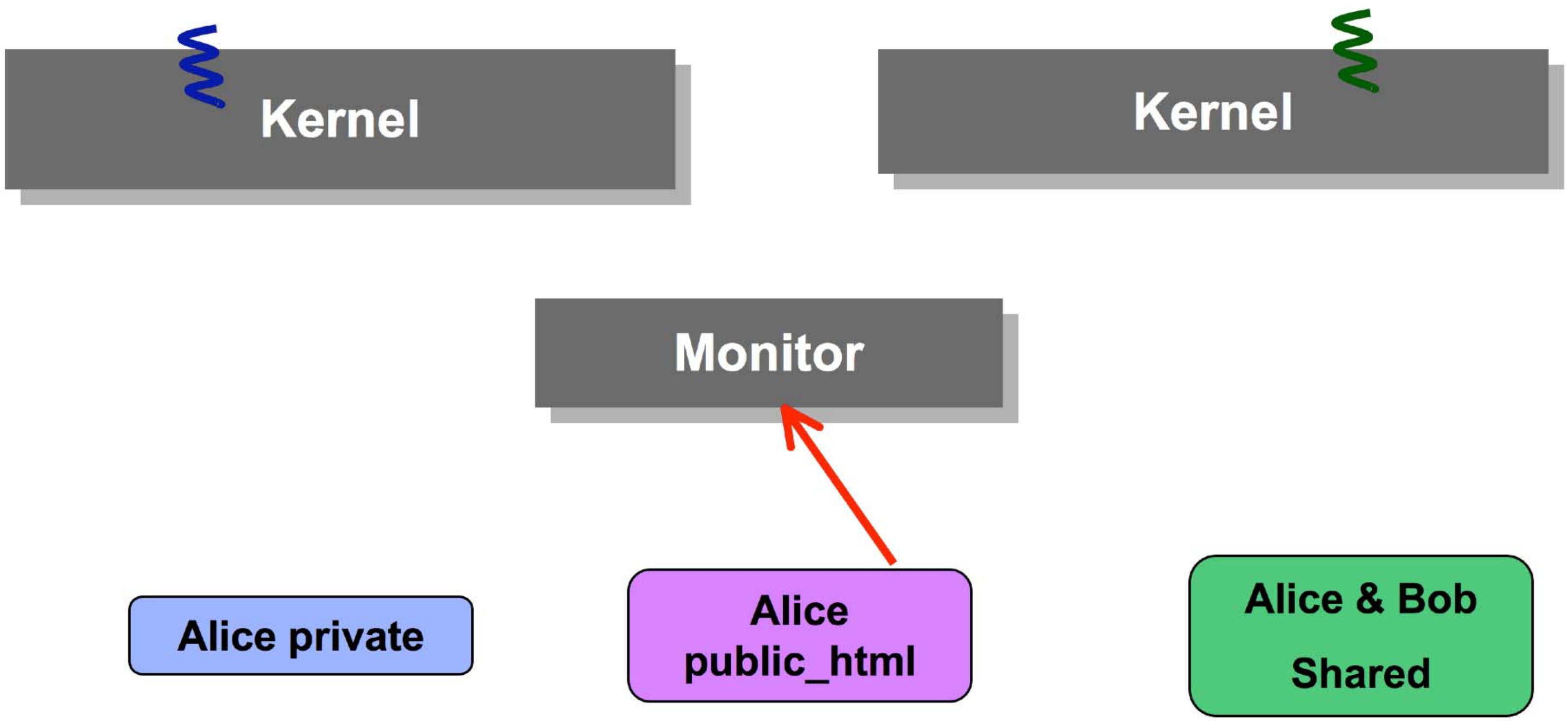
| Tags | Perm |
|---|---|
| | -- |
| | -- |
| | RW |



# Controlled sharing in LoStar

| Tags | Perm |
|---|---|
| | RW |
| | RW |
| | RW |

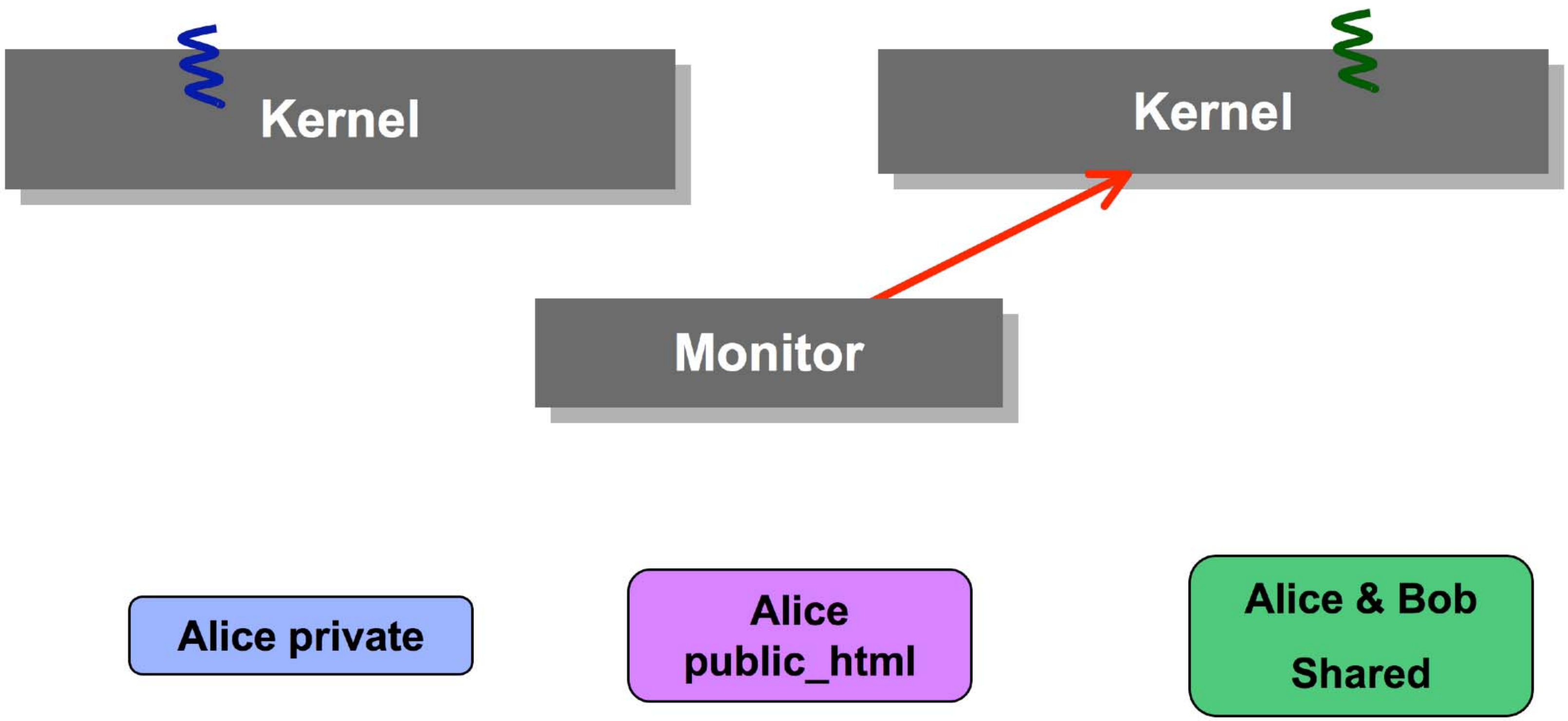
| Tags | Perm |
|---|---|
| | -- |
| | -- |
| | RW |



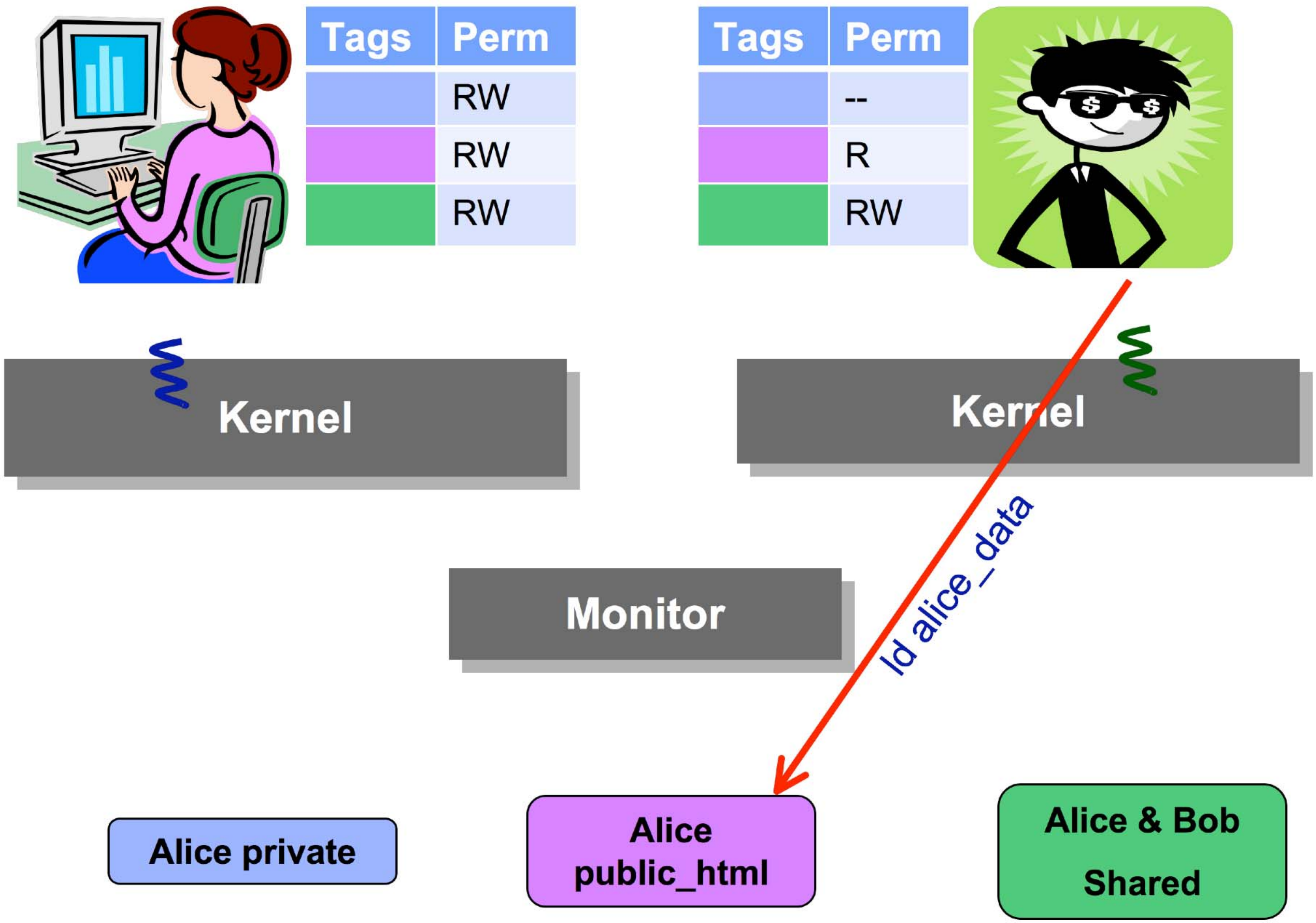
# Controlled sharing in LoStar

| Tags | Perm |
|---|---|
| | RW |
| | RW |
| | RW |

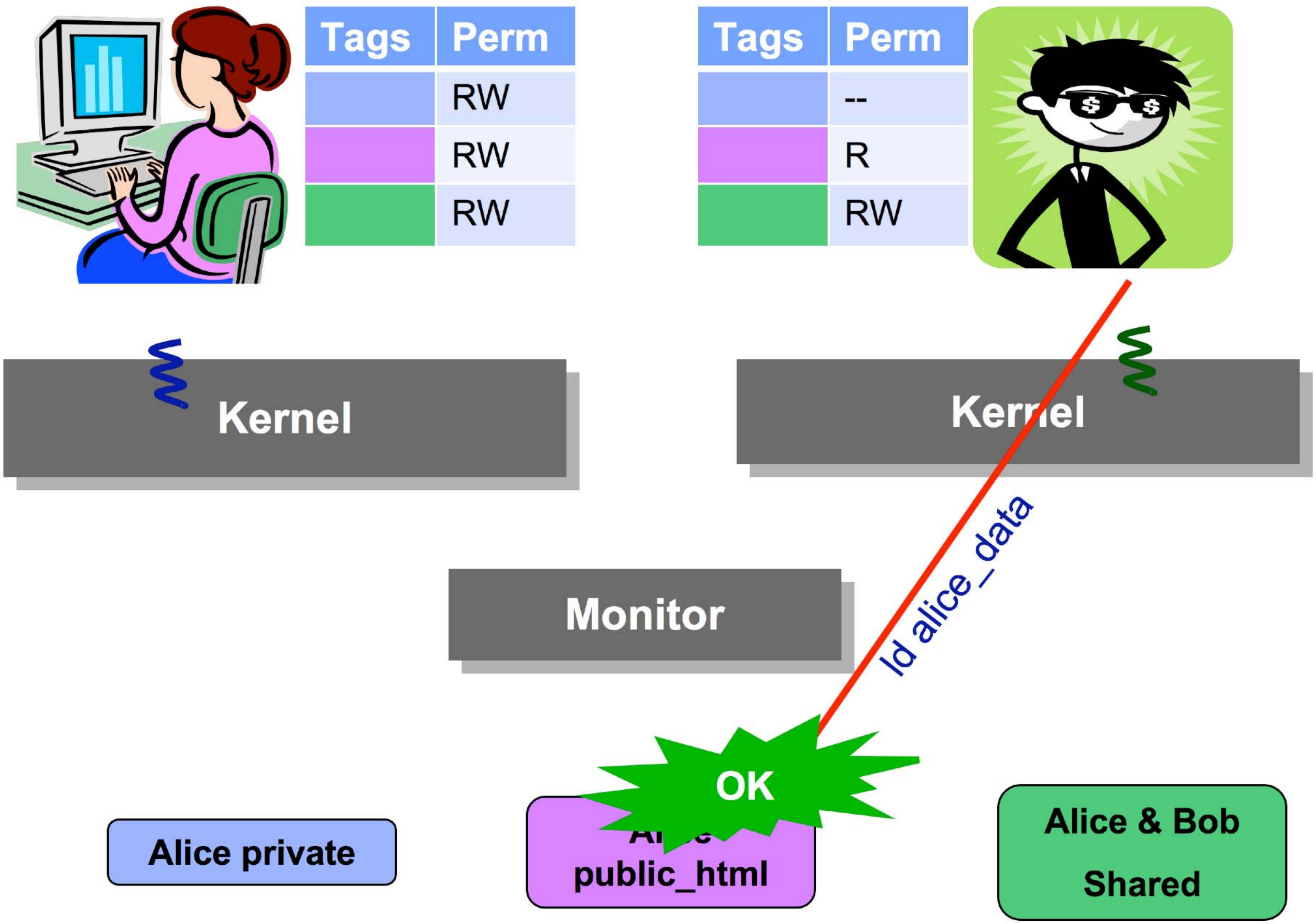
| Tags | Perm |
|---|---|
| | -- |
| | R |
| | RW |



# Controlled sharing in LoStar

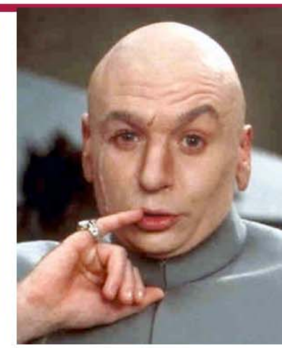


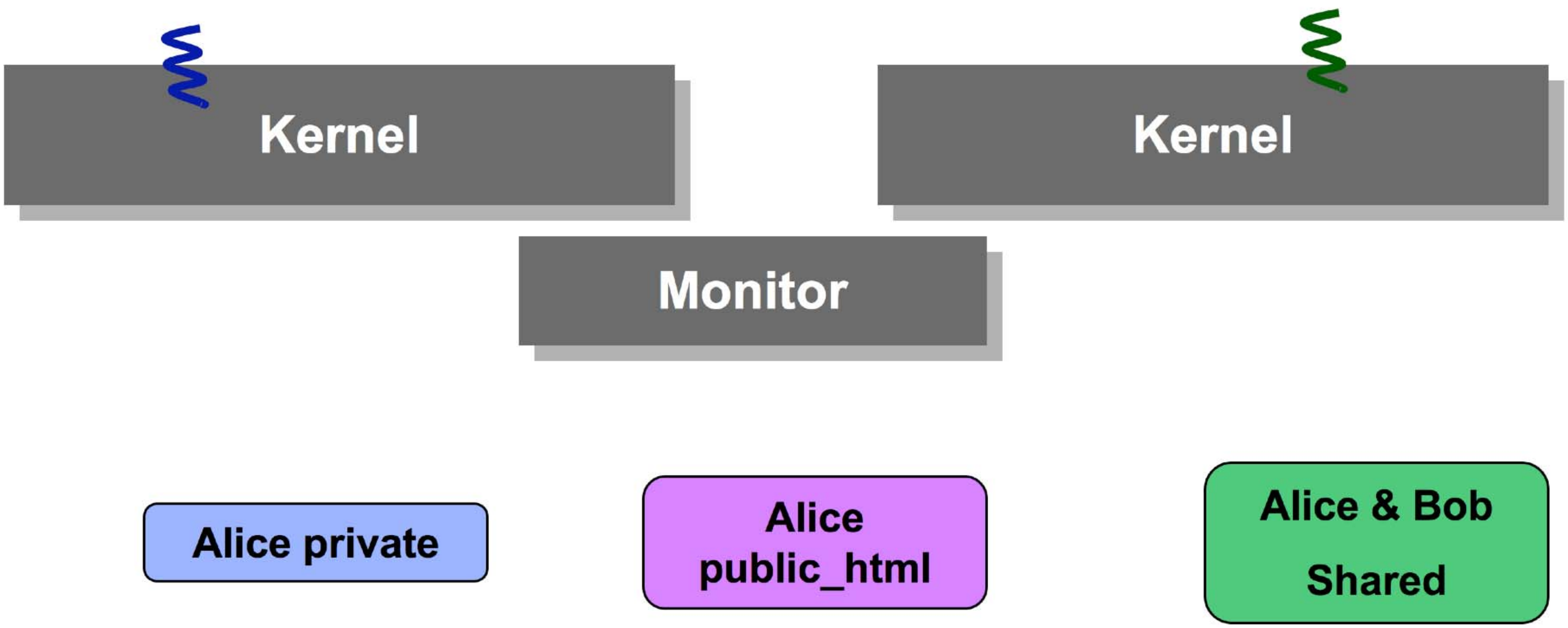
# Controlled sharing in LoStar



# Controlled sharing in LoStar

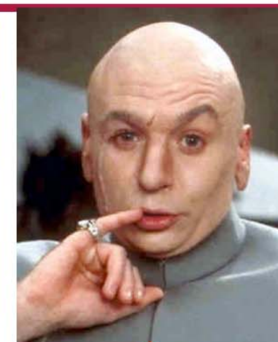
| Tags | Perm |
|---|---|
| (blue) | -- |
| (purple) | R |
| (green) | RW |





# Controlled sharing in LoStar

| Tags | Perm |
| --- | --- |
| | -- |
| | R |
| | RW |



Kernel

Kernel

Monitor

st alice_data

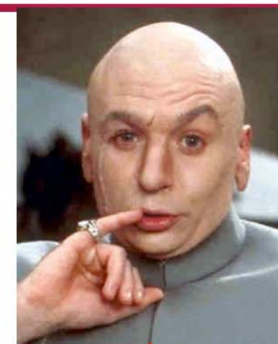
Alice private

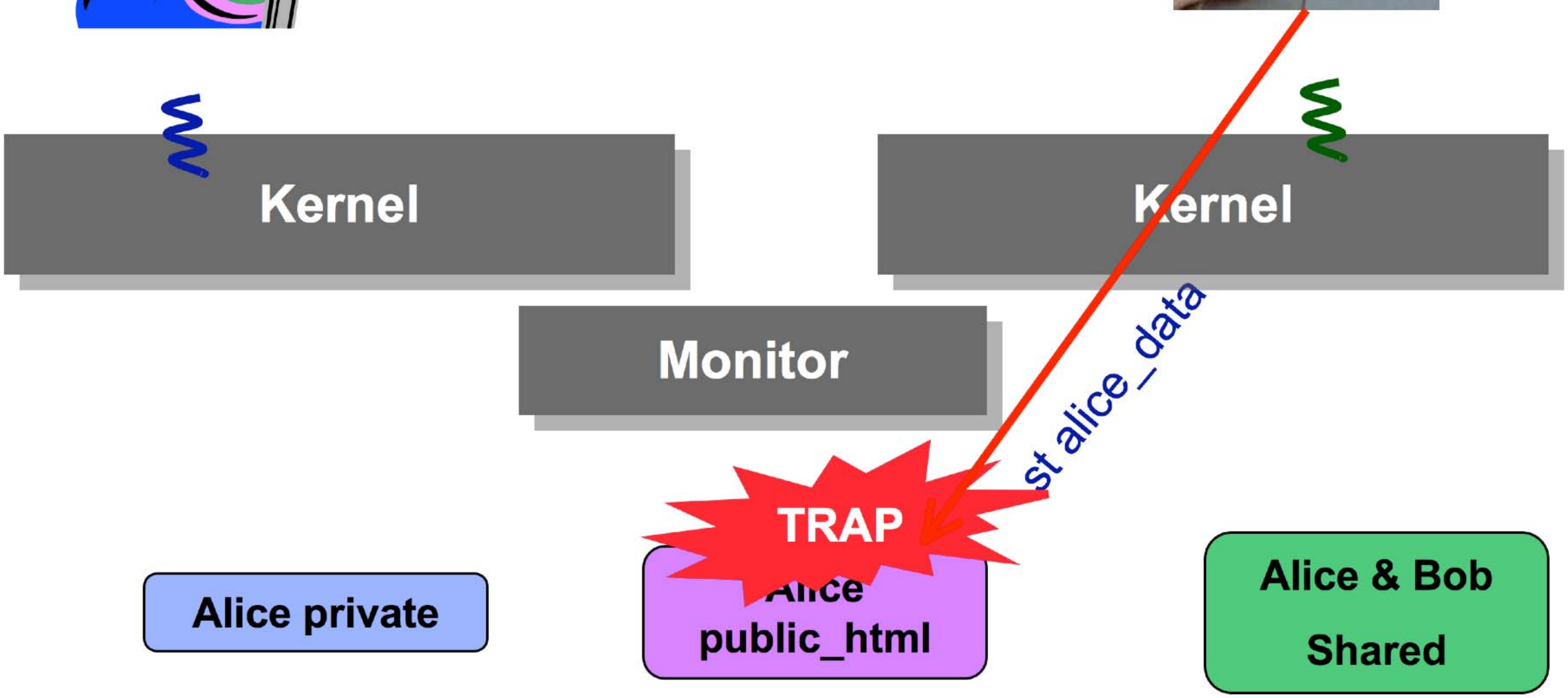
Alice public_html

Alice & Bob Shared

# Controlled sharing in LoStar

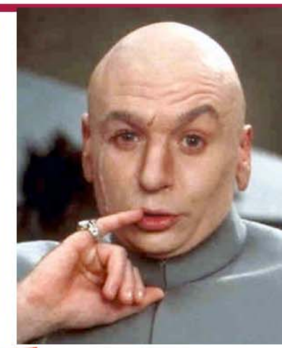
| Tags | Perm |
|---|---|
| | -- |
| | R |
| | RW |

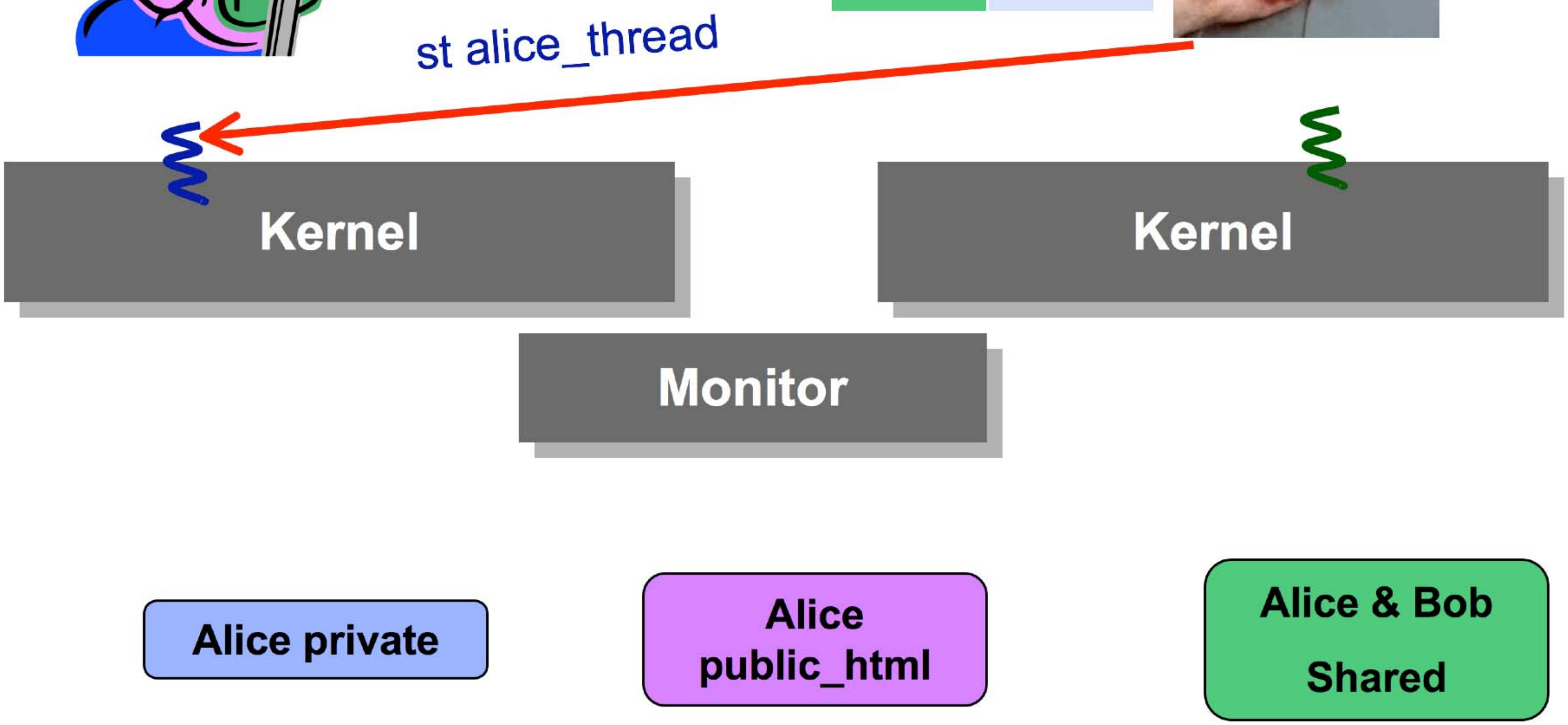




# Controlled sharing in LoStar

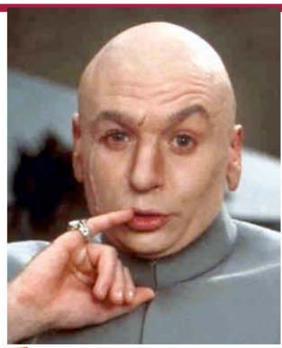
| Tags | Perm |
|---|---|
| | -- |
| | R |
| | RW |

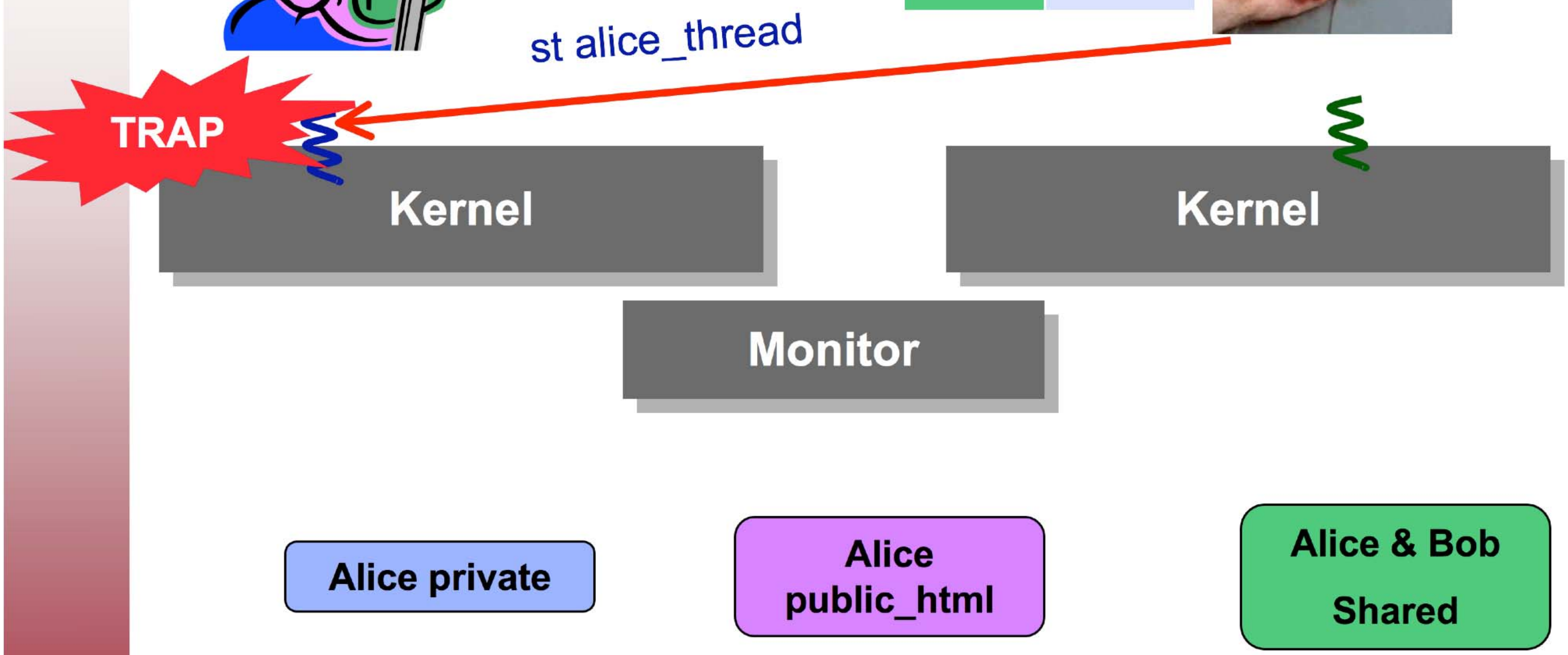




# Controlled sharing in LoStar

| Tags | Perm |
| --- | --- |
| | -- |
| | R |
| | RW |

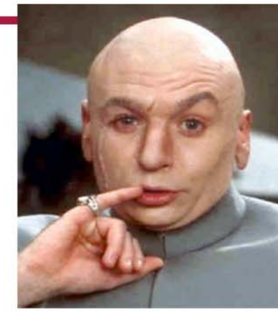


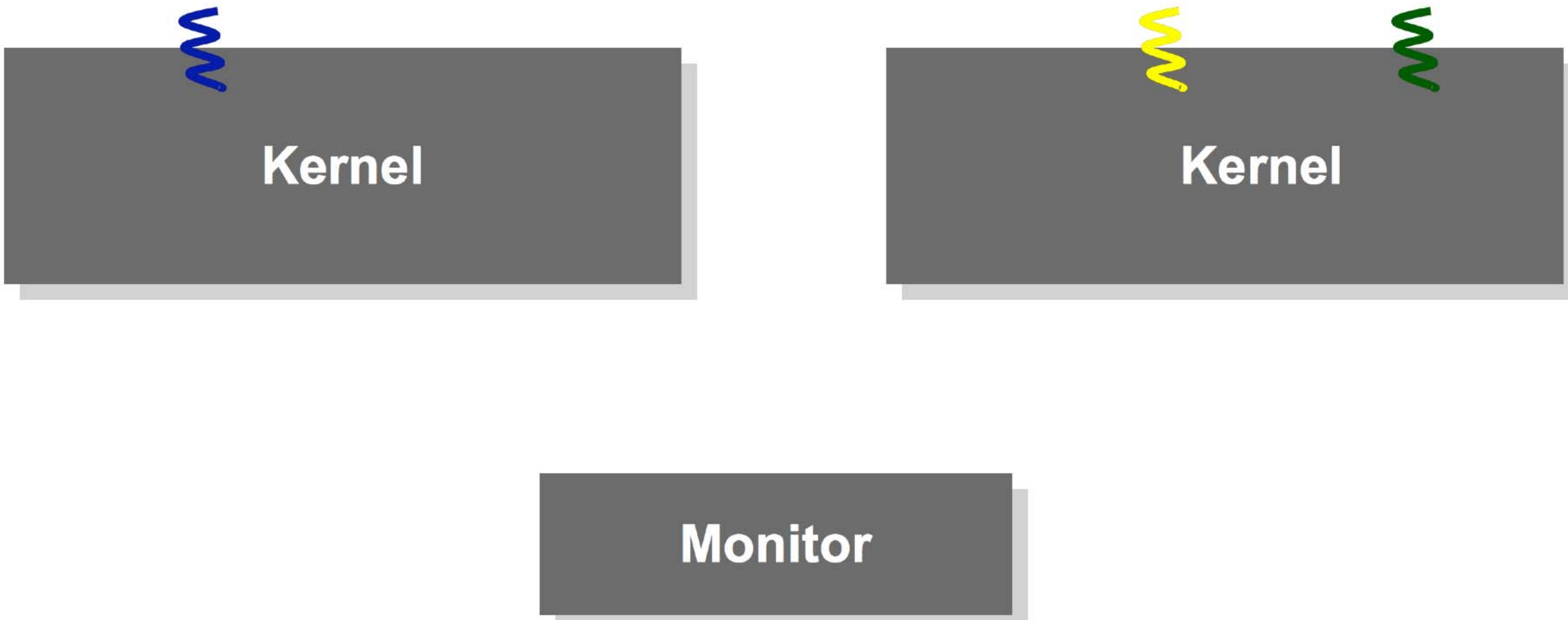



**Monitor provides “yield-to” primitive for thread collaboration**

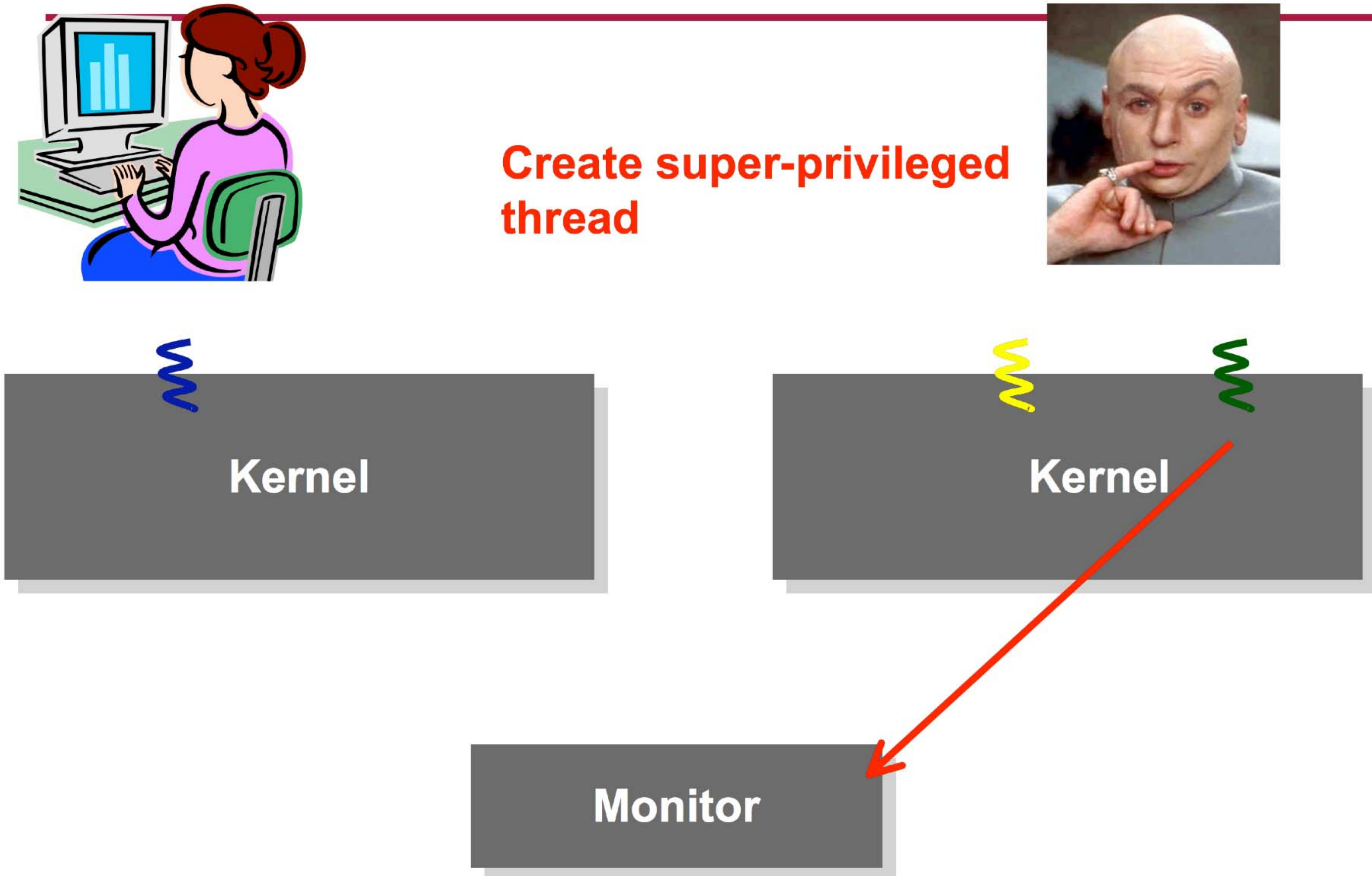
# Controlled sharing in LoStar

Create super-privileged thread



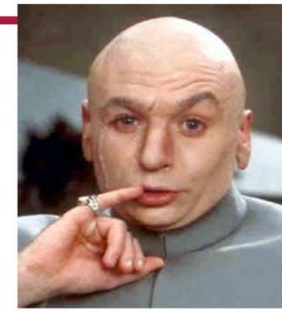


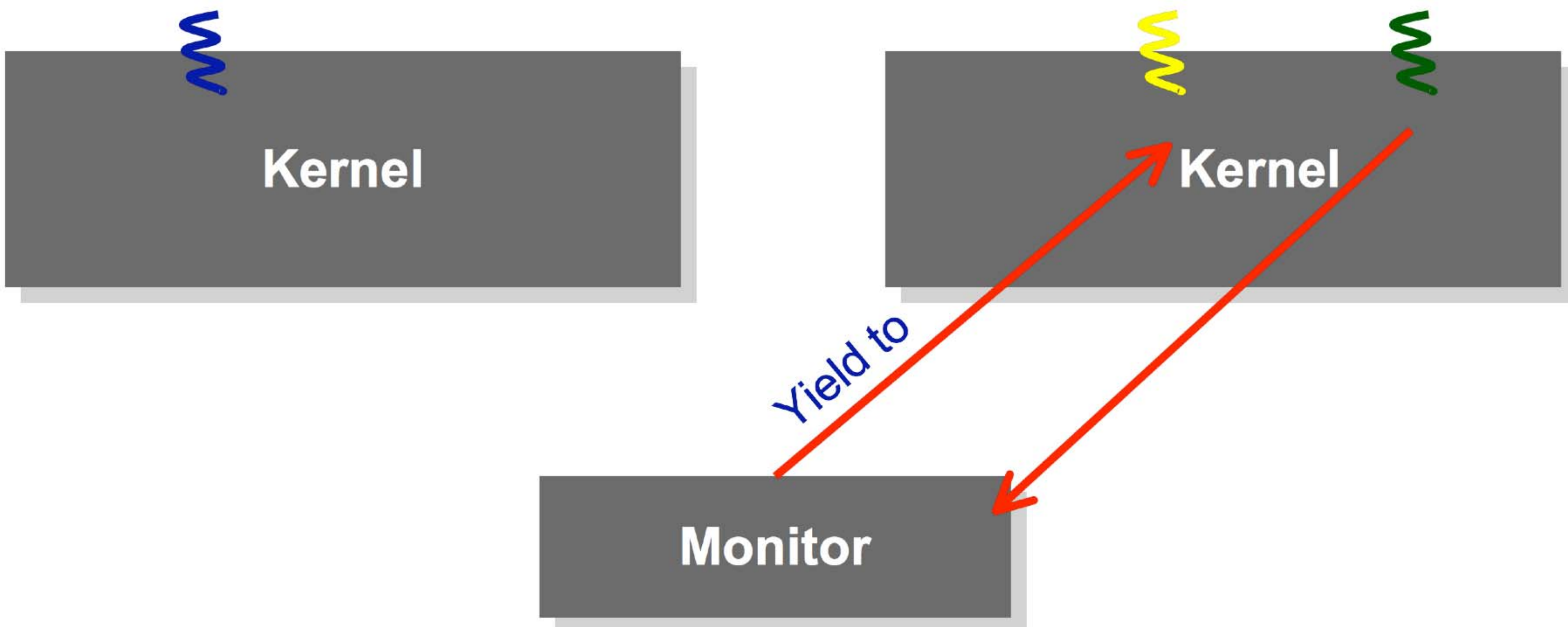
# Controlled sharing in LoStar



# Controlled sharing in LoStar

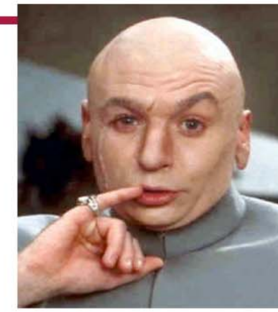
**Create super-privileged thread**

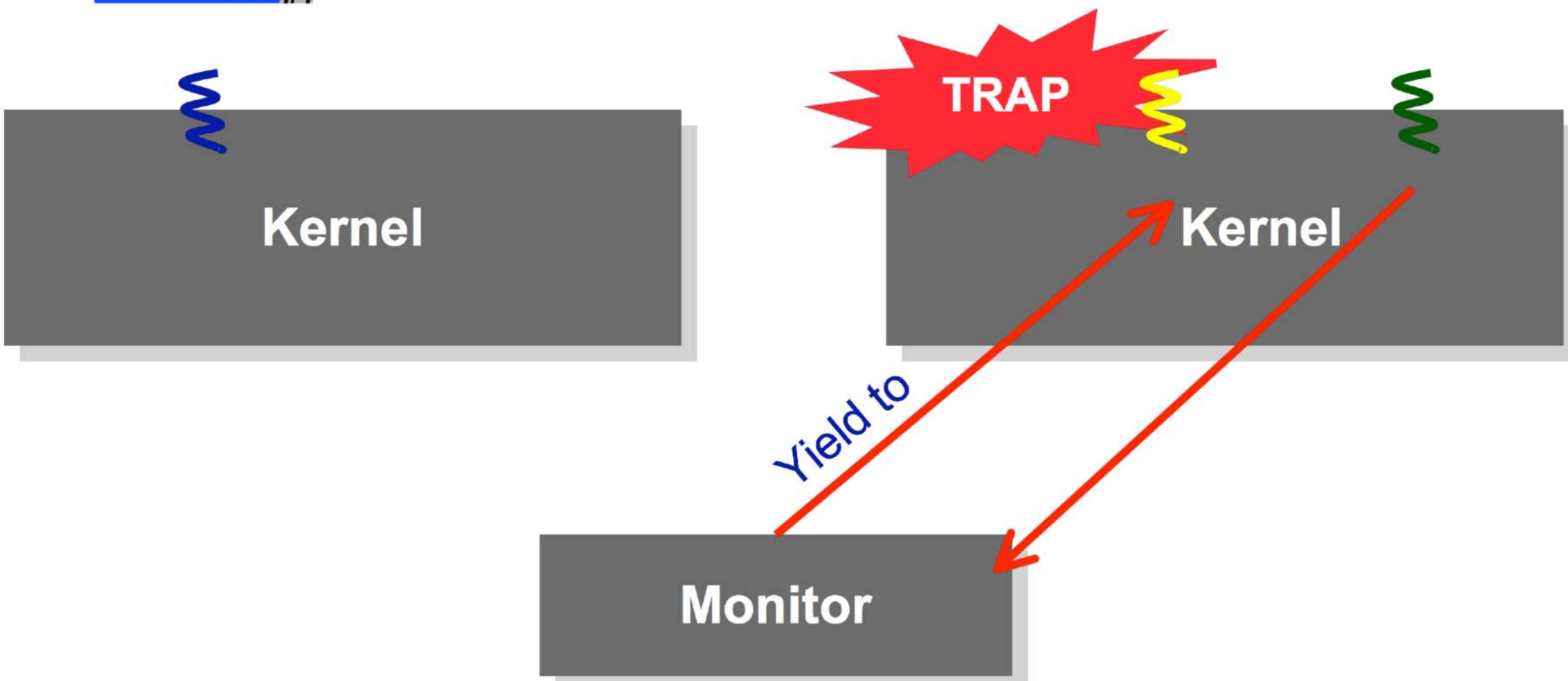




# Controlled sharing in LoStar

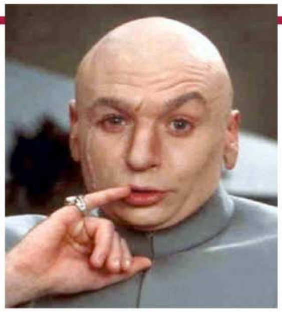
Create super-privileged thread






**Monitor provides operations to**
- Create protection domain
- Change privileges

# Controlled sharing in LoStar

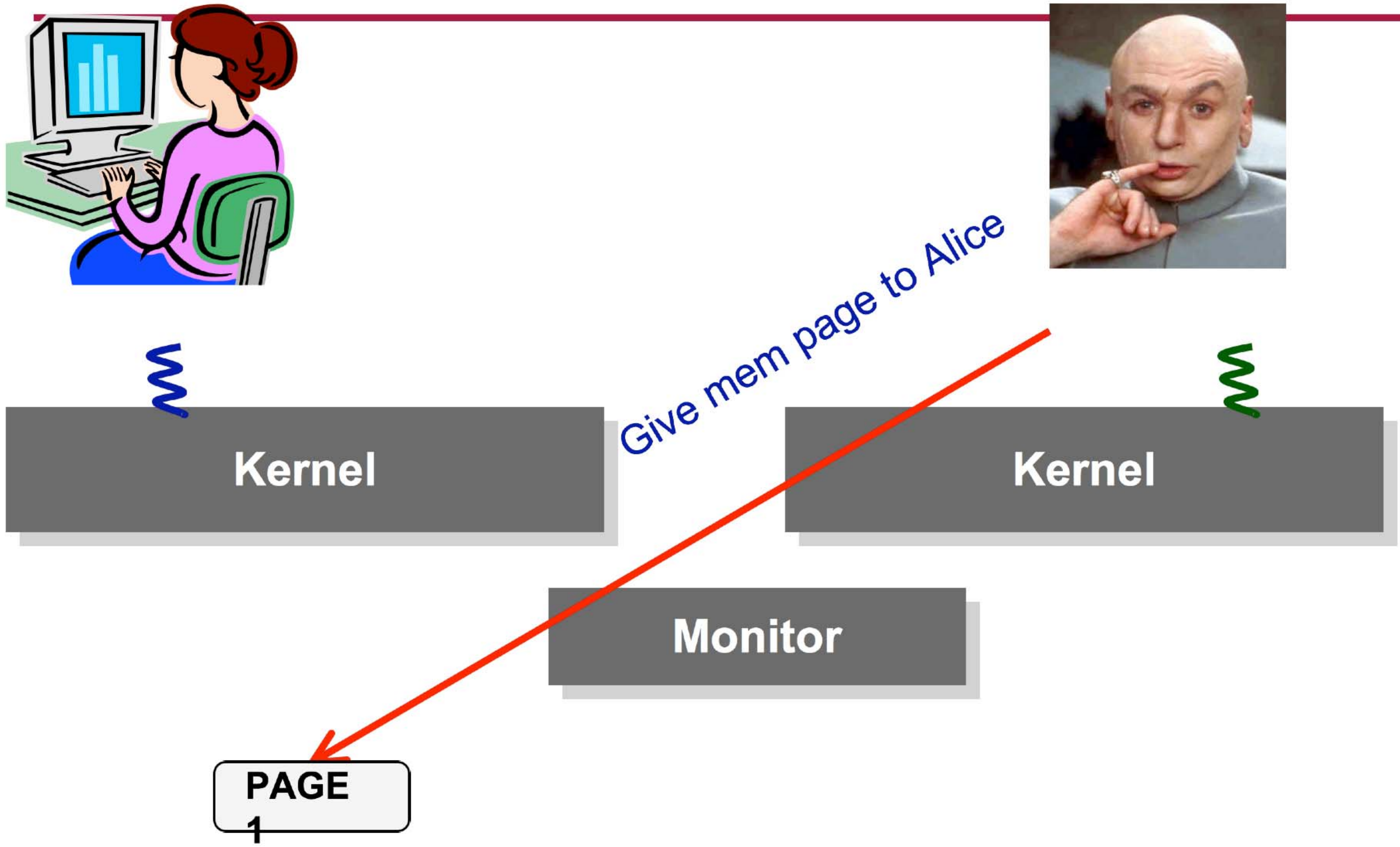


Kernel

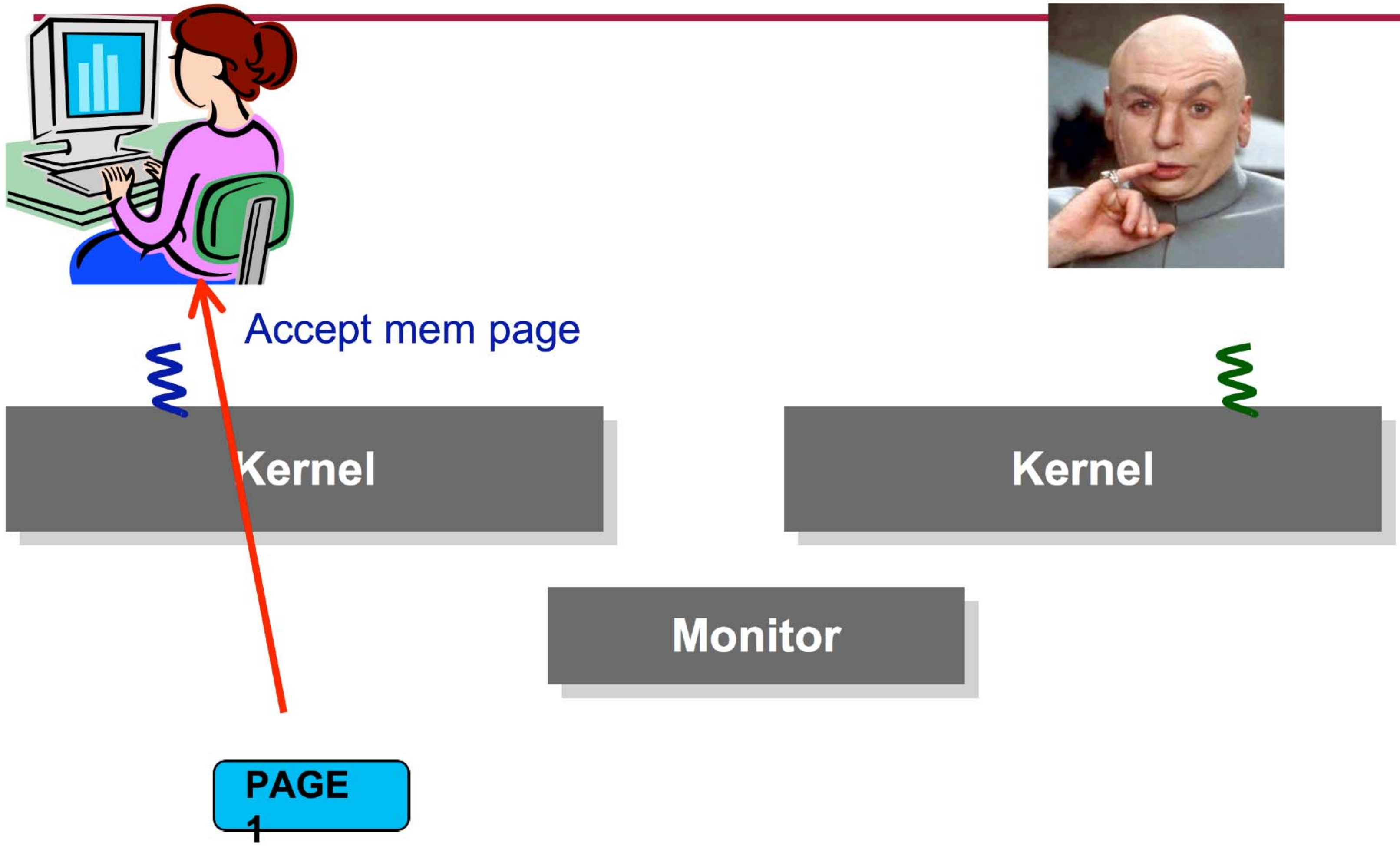
Kernel

Monitor

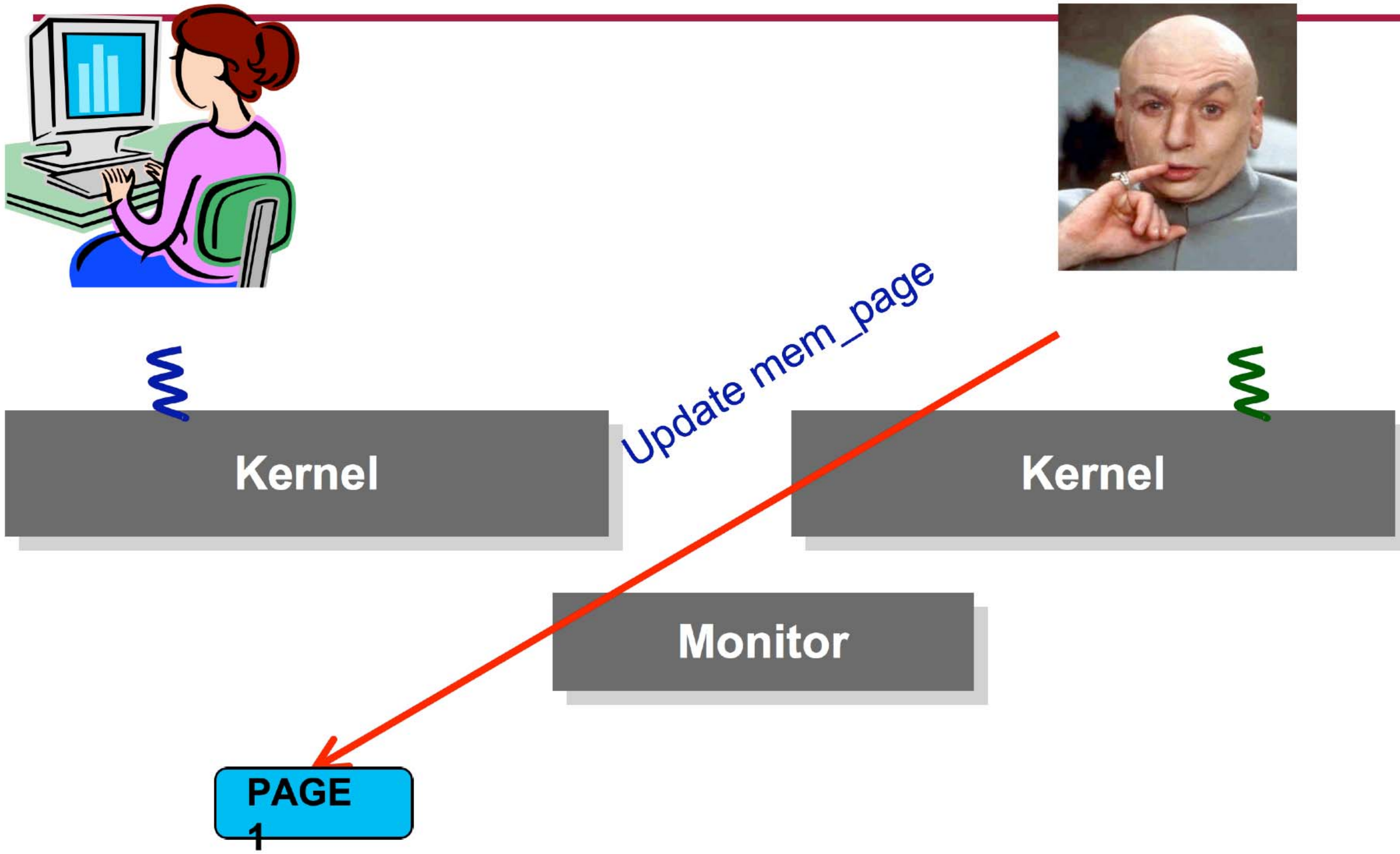
# Controlled sharing in LoStar



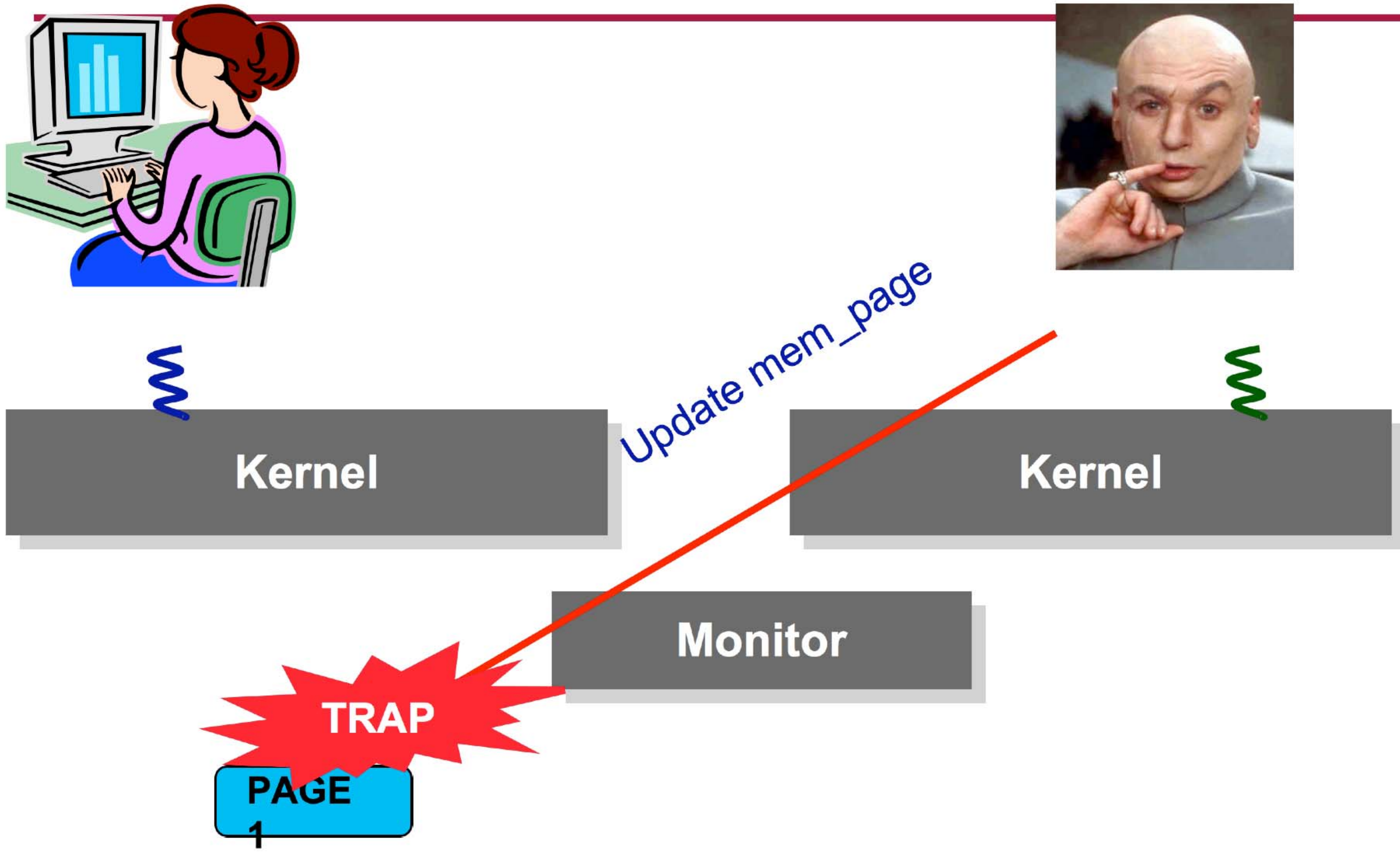
# Controlled sharing in LoStar



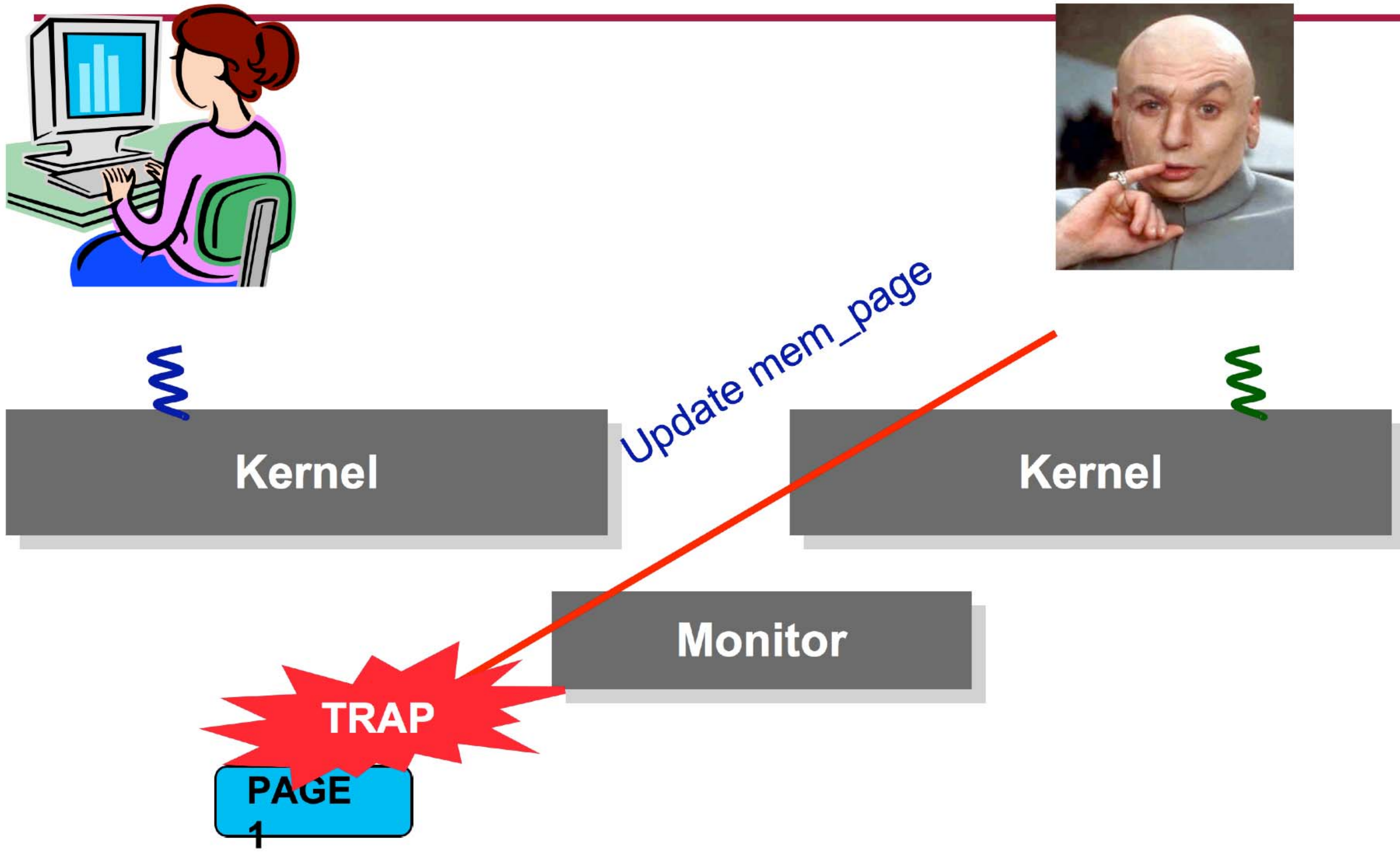
# Controlled sharing in LoStar



# Controlled sharing in LoStar

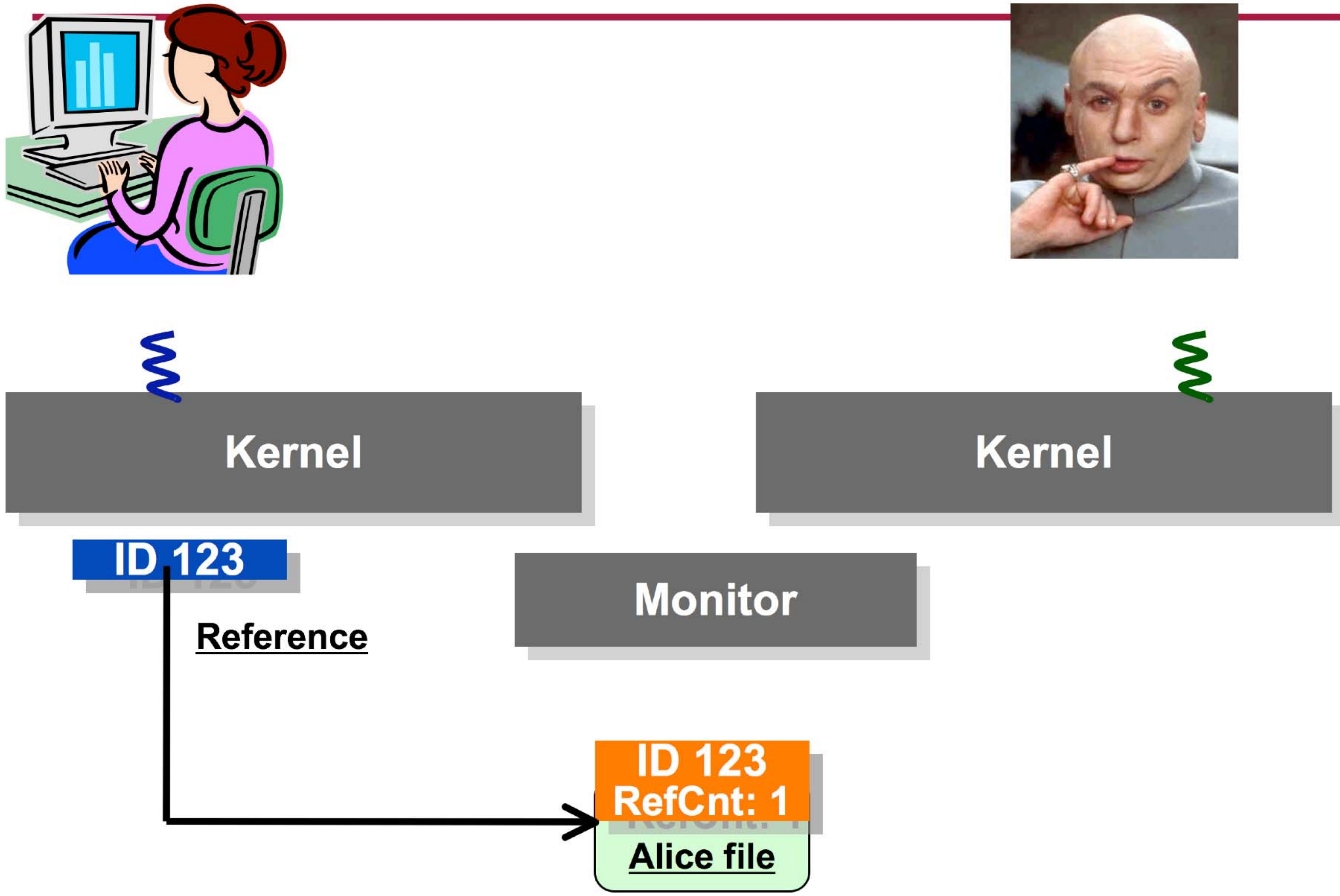


# Controlled sharing in LoStar

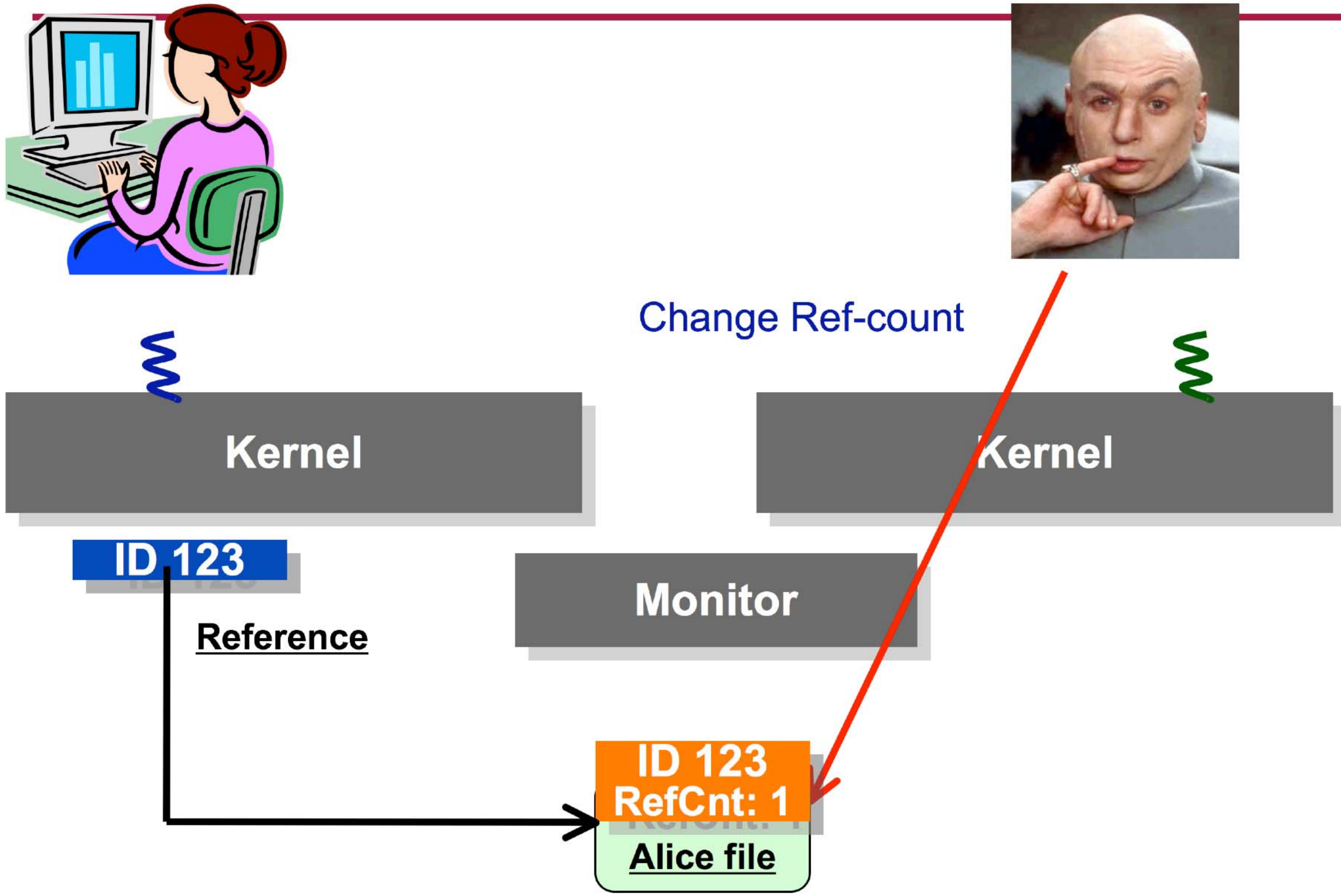



**Alice's memory allocator sets tags on physical memory**

# Controlled sharing in LoStar

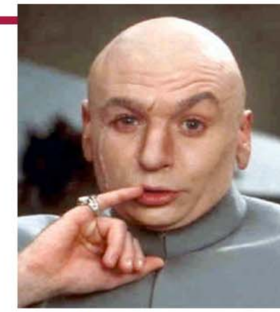


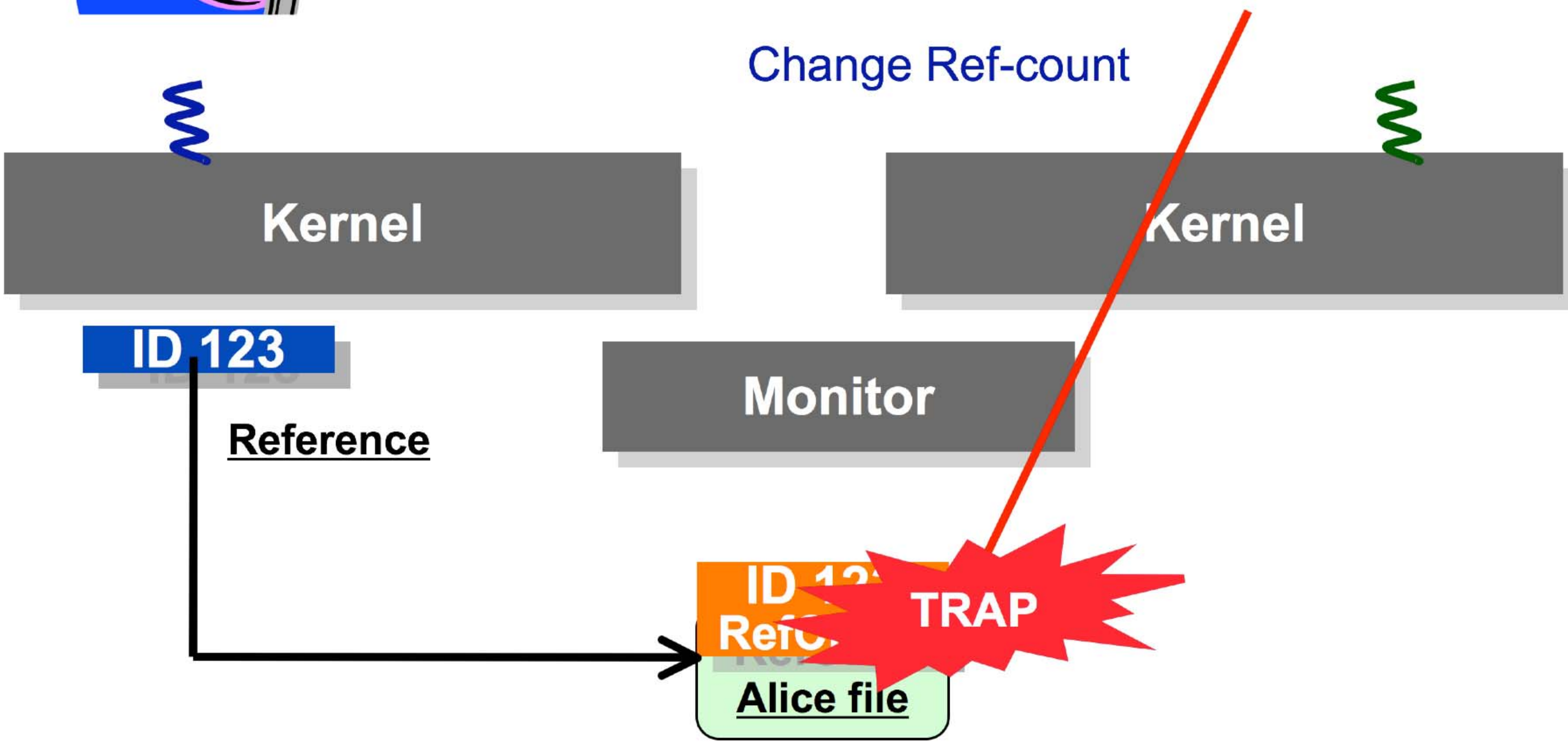
# Controlled sharing in LoStar



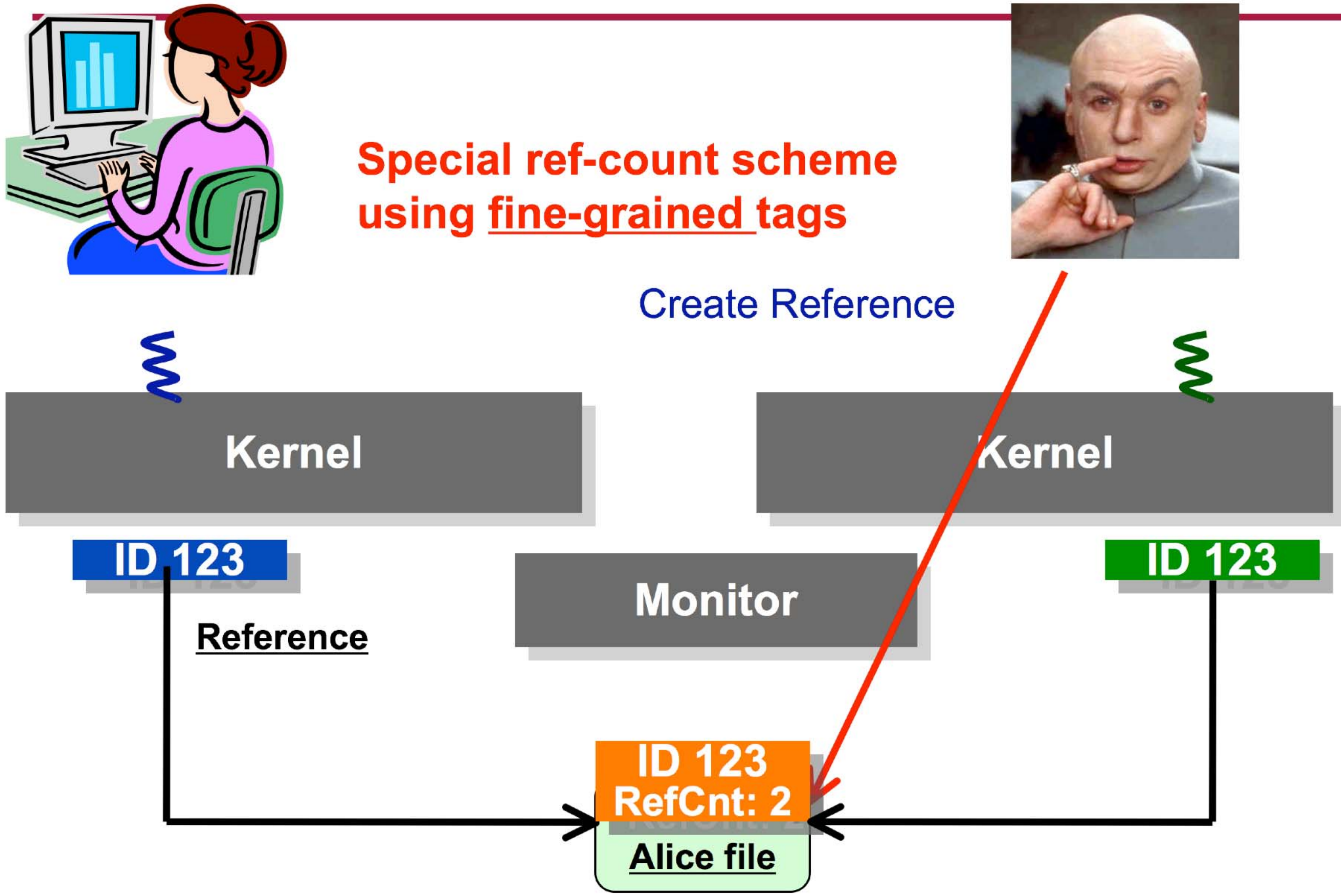
# Controlled sharing in LoStar

**Special ref-count scheme using fine-grained tags**



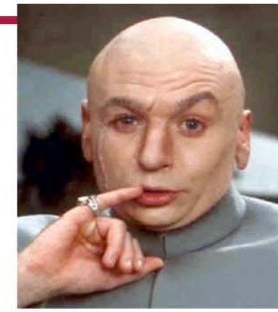


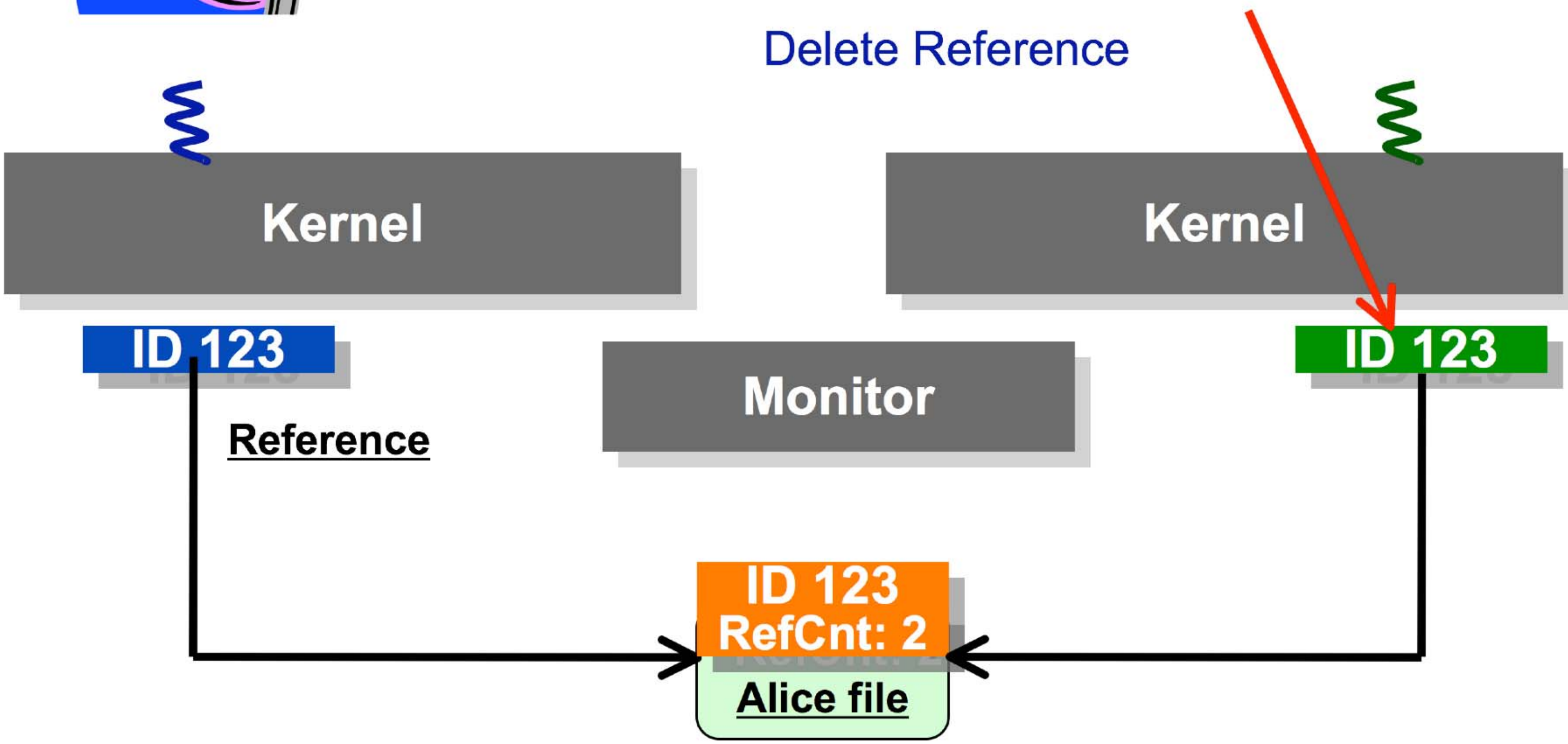
# Controlled sharing in LoStar



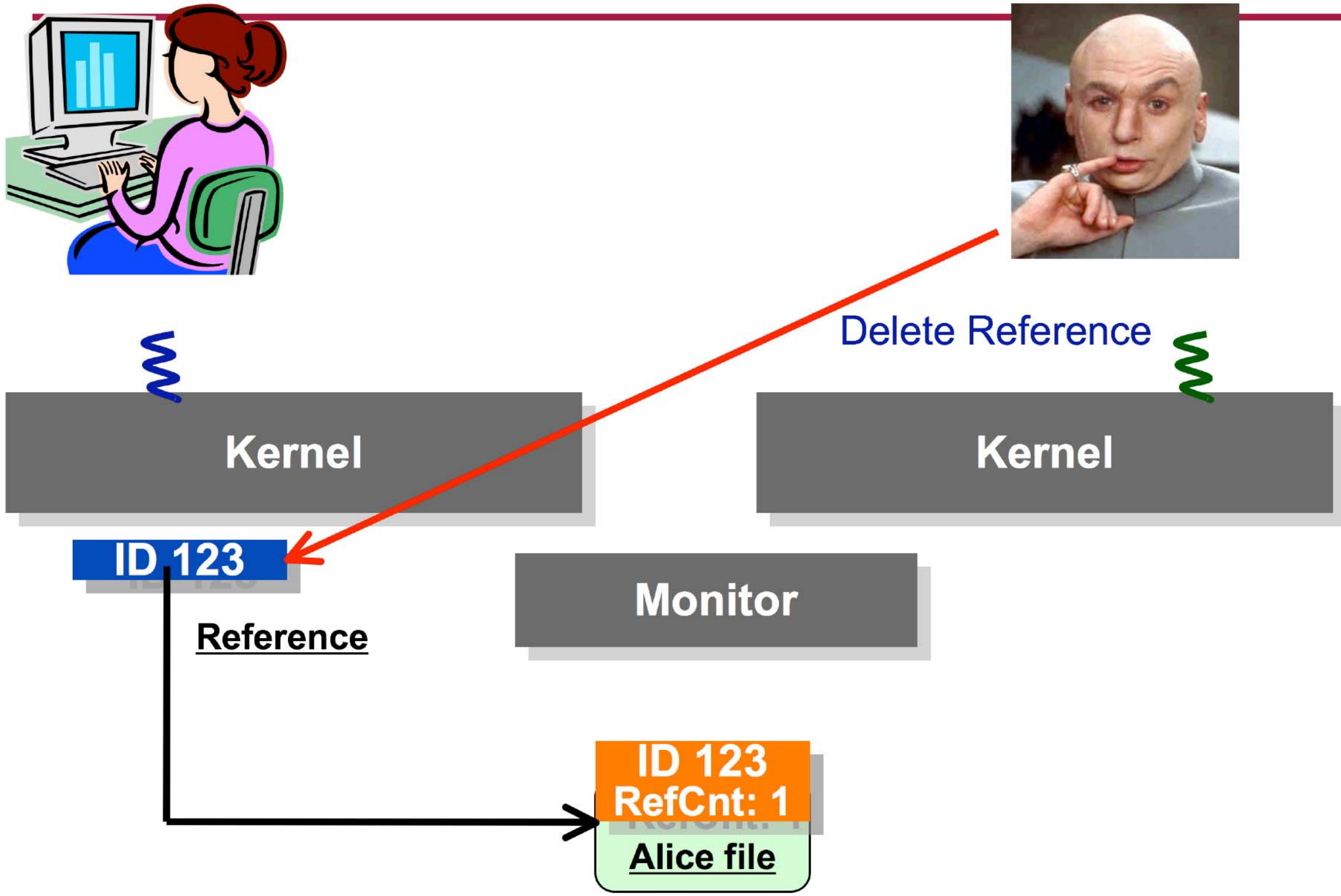
# Controlled sharing in LoStar

**Special ref-count scheme using fine-grained tags**

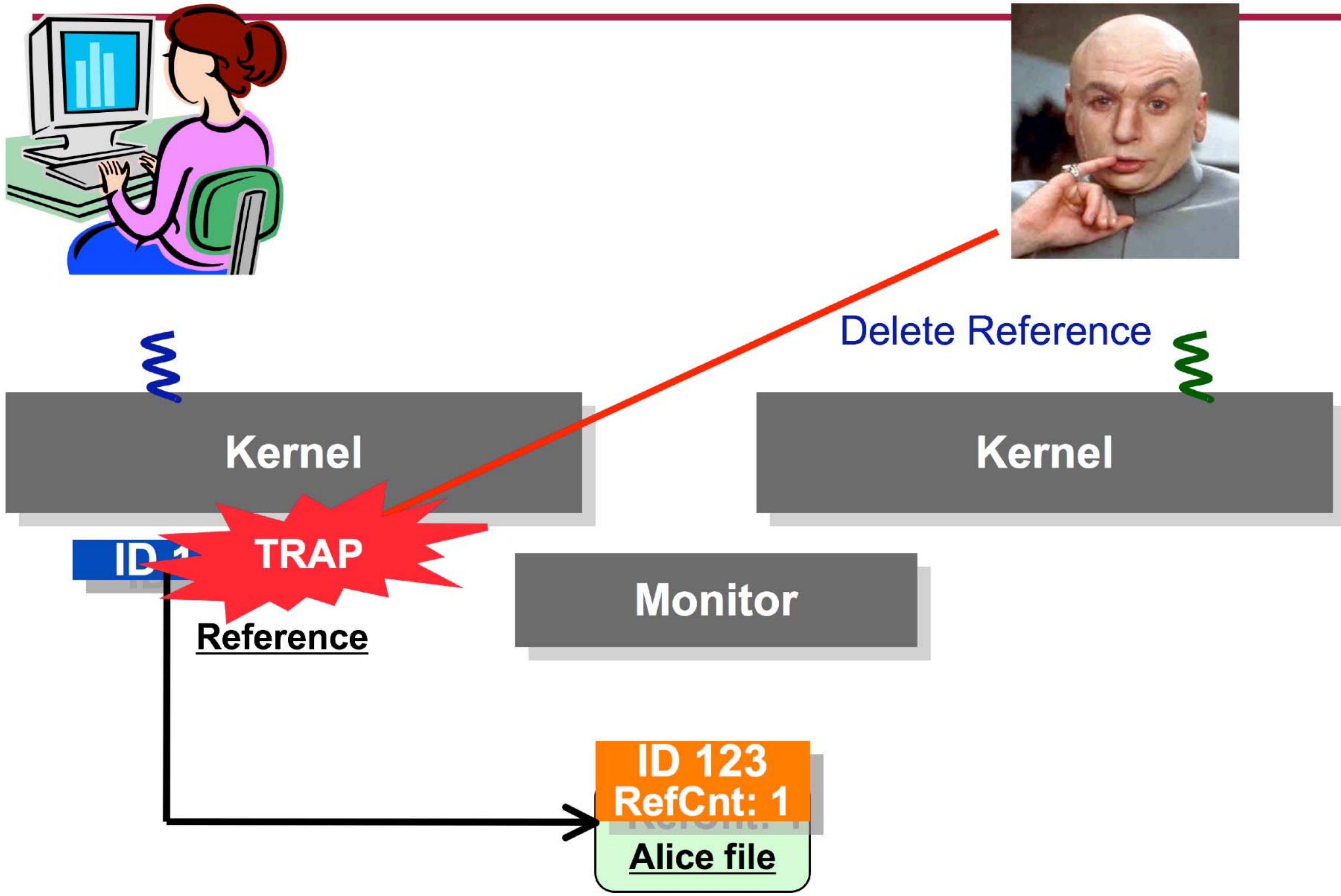




# Controlled sharing in LoStar

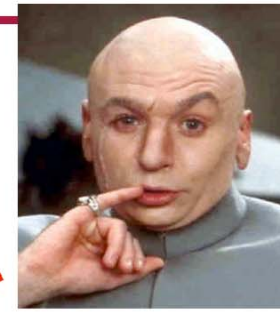


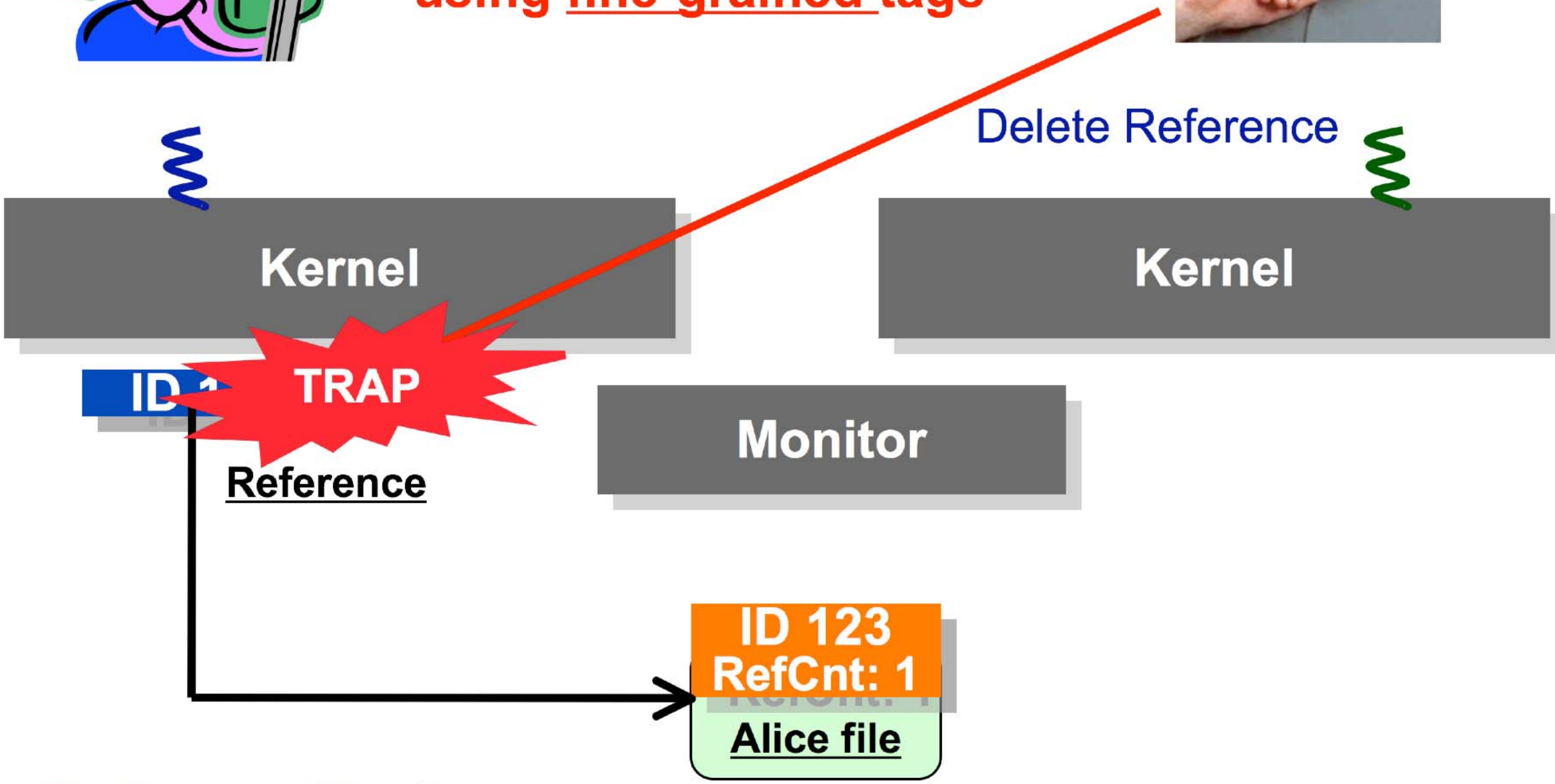
# Controlled sharing in LoStar



# Controlled sharing in LoStar

References protected using fine-grained tags

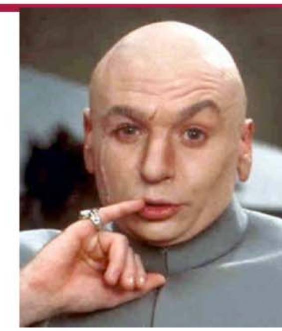


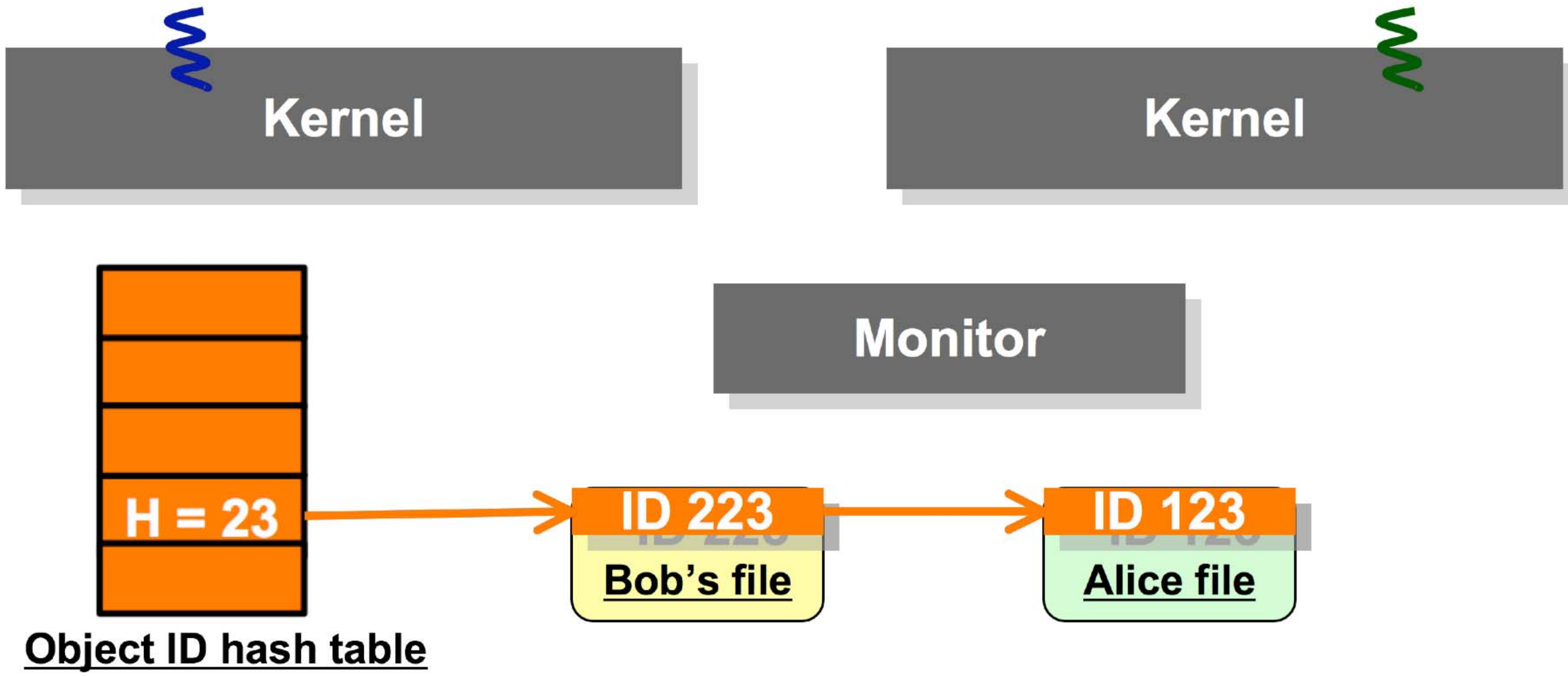



Garbage collection:

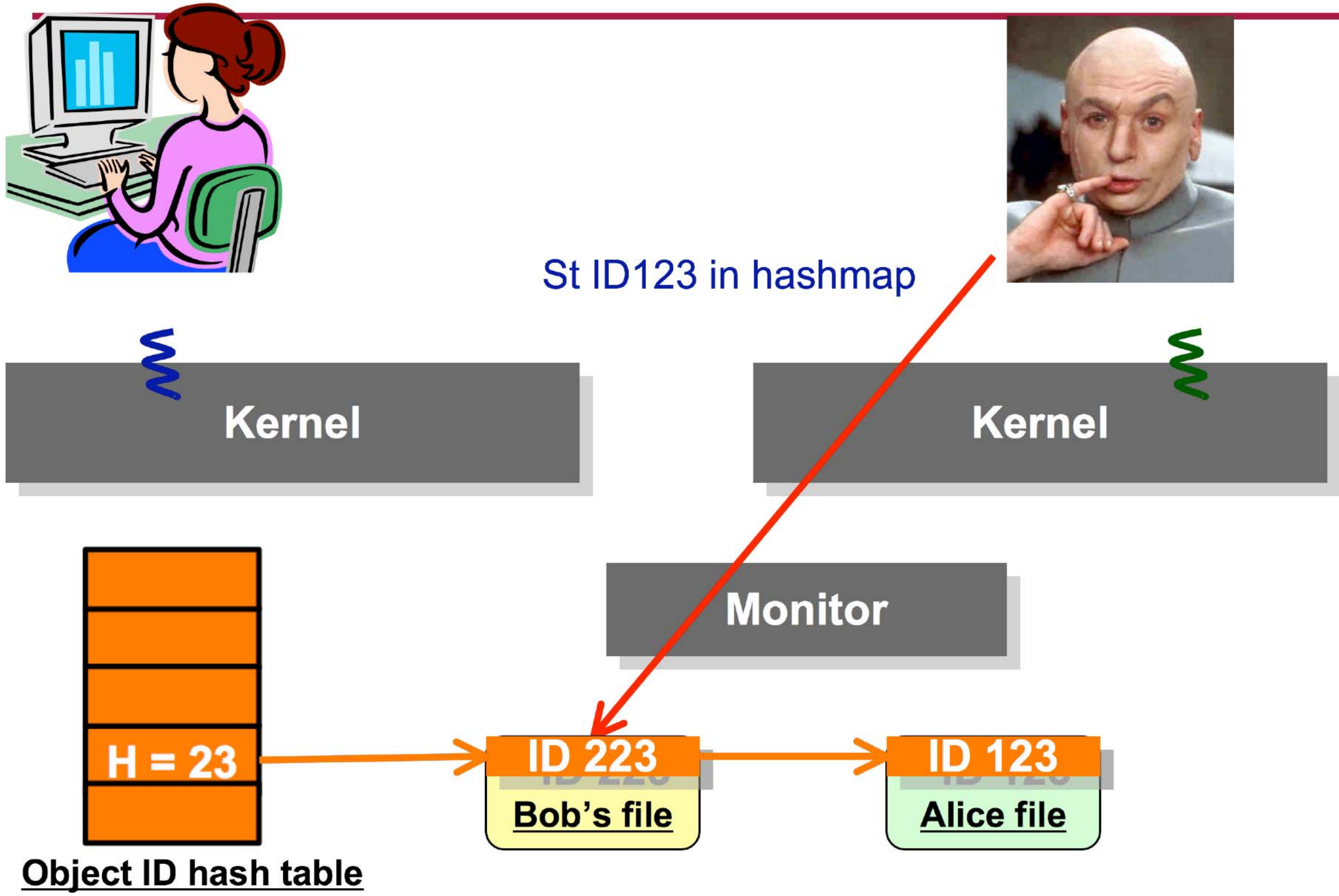
- Separate idle "GC" protection domain for each object
- GC code thus runs outside of the monitor (TCB)

# Controlled sharing in LoStar

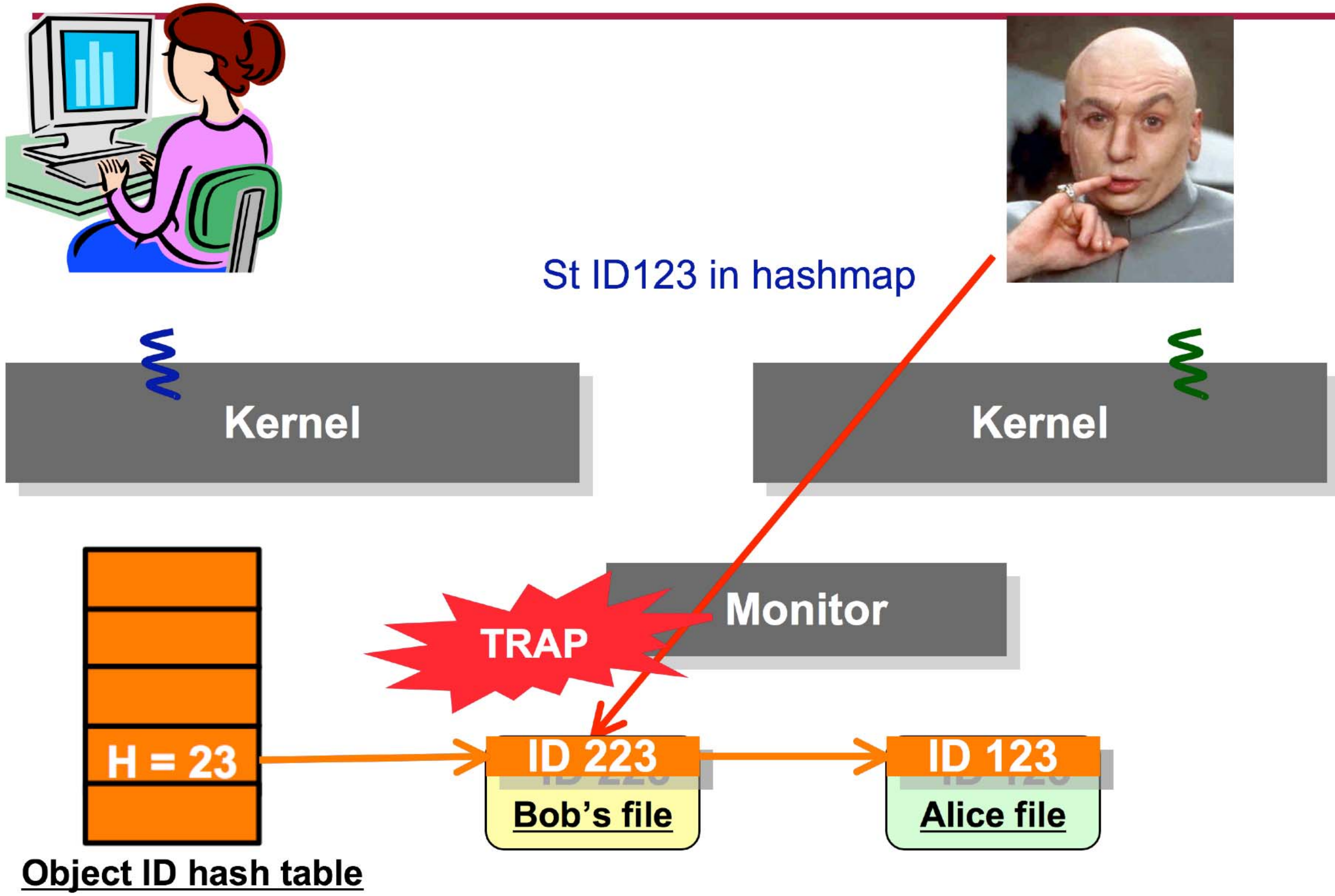




# Controlled sharing in LoStar

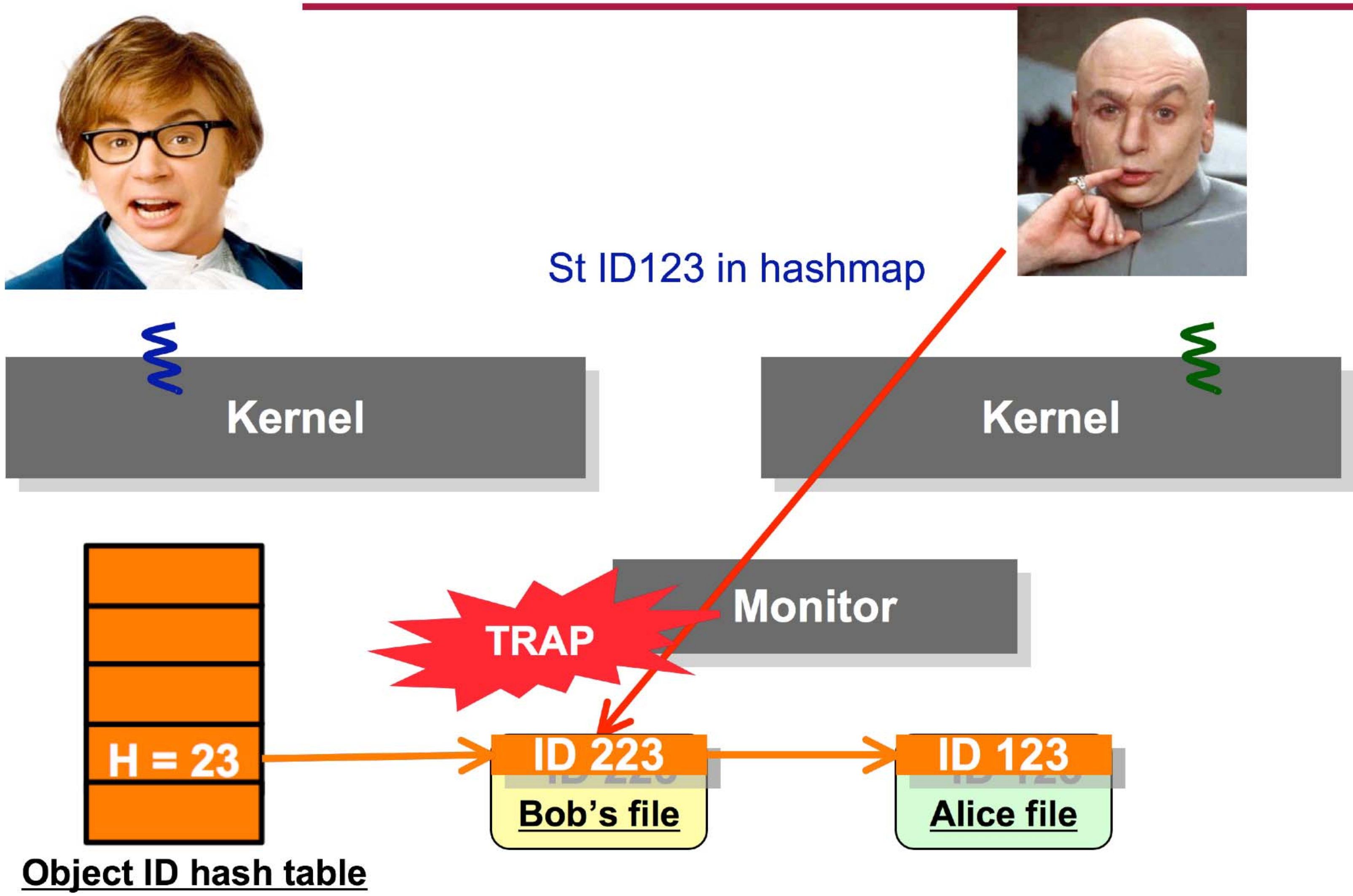


# Controlled sharing in LoStar




**Monitor sets read-only tags on objectID, and linked list pointers**
**Need fine-grained protection**

# Controlled sharing in LoStar




**Monitor sets read-only tags on objectID, and linked list pointers**
**Need fine-grained protection**

# Monitor vs. Kernel

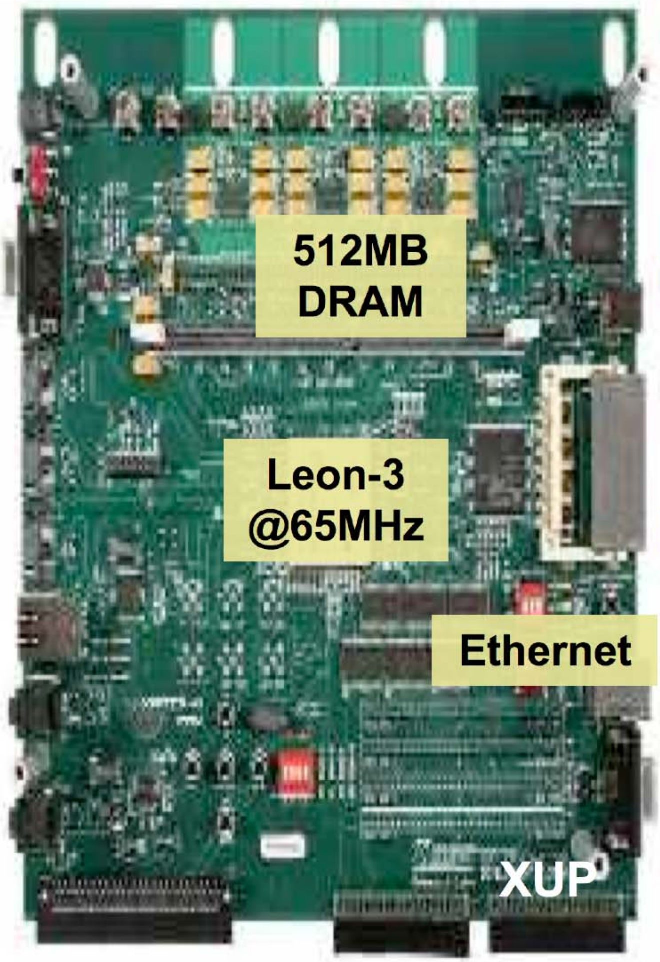
- Monitor operations:
  - Uses tags for enforcing security
  - Manipulating protection domains
  - Context-switching between protection domains
  - Manipulating memory

- Kernel still in charge of scheduling, translation etc.

# Kernel security mechanisms

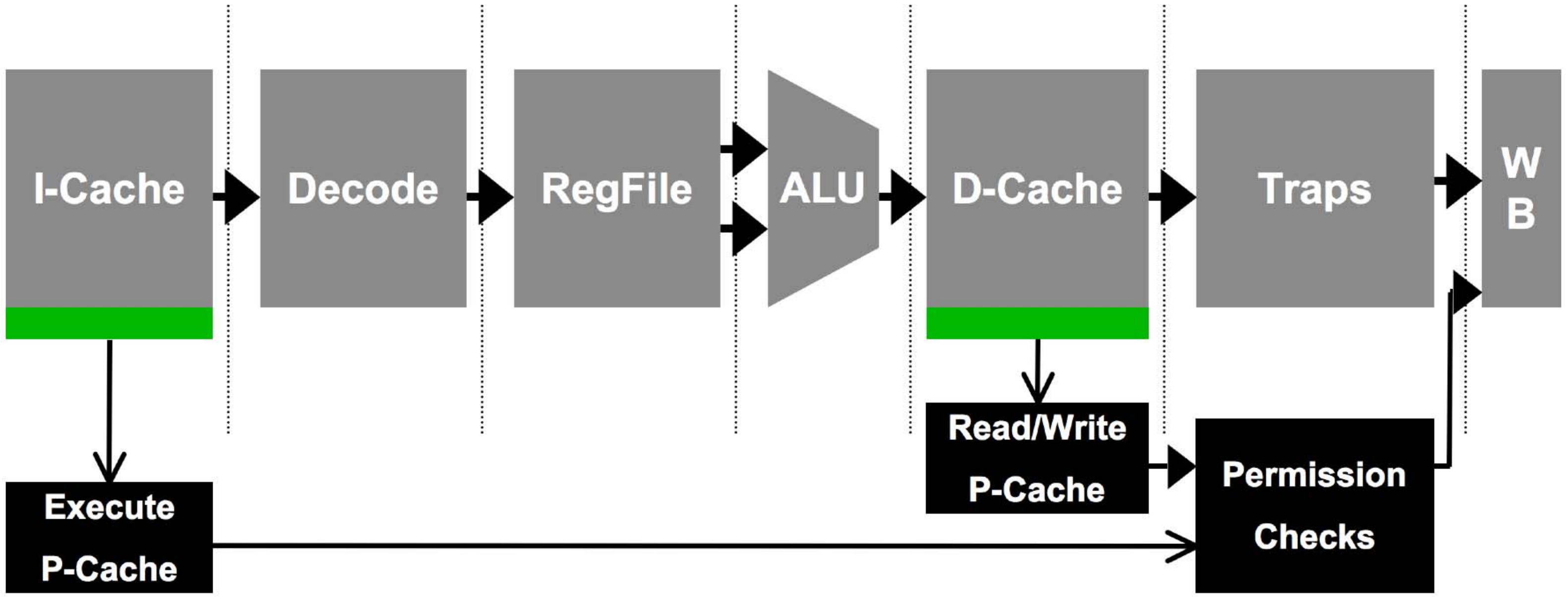
- So what does the kernel do?
  - Monitor provides read-write memory protection
  - However, monitor cannot ensure liveness
  - Monitor cannot prevent covert channels
    - In LoStar, HiStar's kernel can provide stronger security
    - Uses information flow to prevent covert channels

# LoStar Prototype System




- HW: modified SPARC processor (Leon3)
  - In-order, 7 stage pipeline
  - Mapped to FPGA board
- Loki logic overhead: 19%
  - Much lower fraction in more complex CPUs
- Loki clock frequency overhead: none

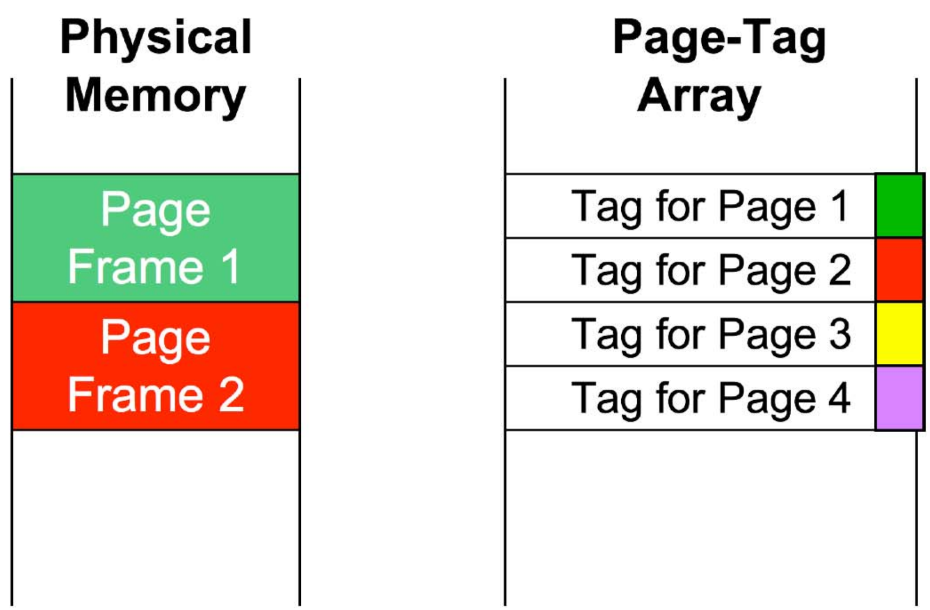
# Loki Architecture




- Instruction/data caches extended to hold tag bits
- Permissions cache accessed in two cases
  - On instruction fetch using the instruction tag
  - On loads/stores using the data tag
- Exceptions invoke security monitor

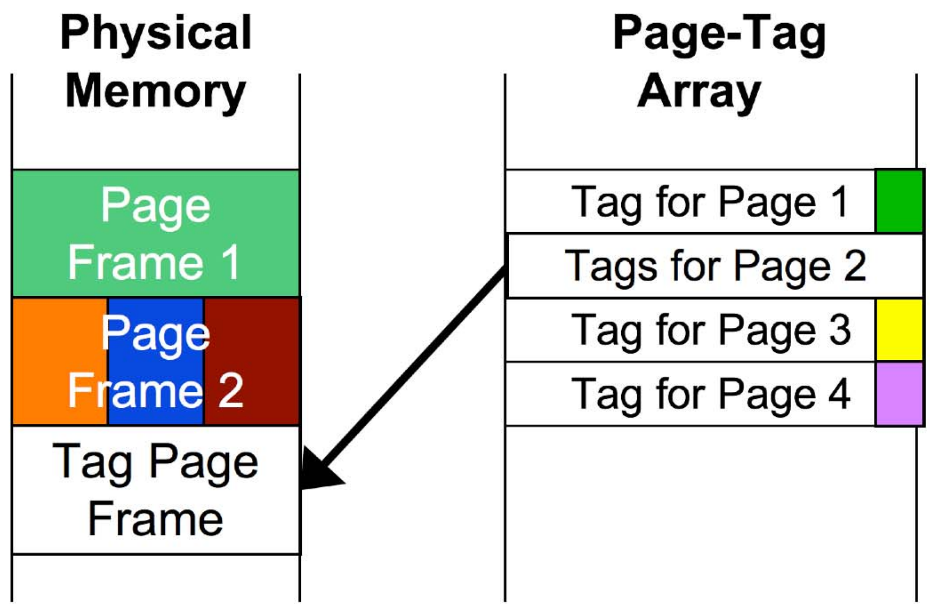
# Loki Tag Storage

- Fine-grained permission for physical memory
- Simple approach: +32 bits/word in caches and memory
  - 100% storage overhead
- Multi-granular tag storage scheme [Suh'04]
  - Exploit tag similarity to reduce storage overhead
  - Page-level tags ⇒word-level tags



# Loki Tag Storage

- Fine-grained permission for physical memory
- Simple approach: +32 bits/word in caches and memory
  - 100% storage overhead
- Multi-granular tag storage scheme [Suh'04]
  - Exploit tag similarity to reduce storage overhead
  - Page-level tags ⇒ word-level tags



# Monitor Mode

- Updates of tags and permission cache entries
  - Requires new operating mode

| | |
|---|---|
| **User** | Limited instructions; limited address ranges; VM |
| **Kernel** | Access to all instructions & address ranges; VM/PM |
| **Monitor** | Access to all instructions & address ranges; PM |

# Monitor Mode

- Updates of tags and permission cache entries
  - Requires new operating mode

| | |
|---|---|
| **User** | Tags are transparent |
| **Kernel** | Can read tags |
| **Monitor** | Can read/write tags |

- On permission check failure/permission cache miss
  - Switch to monitor mode & jump to security monitor
  - Disable processor's MMU
    - This helps remove MMU handling code from the OS' TCB

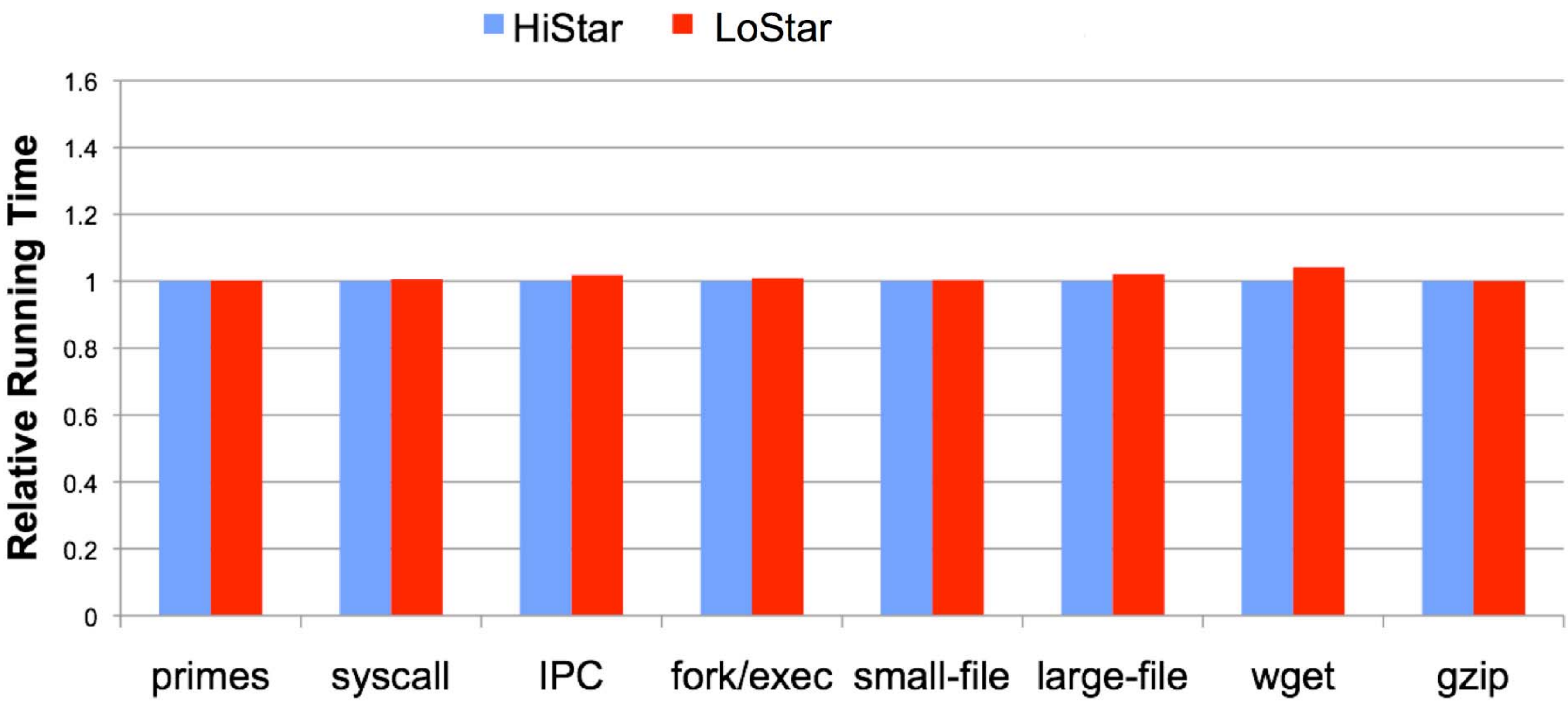
# TCB Reduction

- TCB reduction

| LOC | HiStar | LoStar |
|---|---|---|
| Kernel | 11,600 (**trusted**) | 12,700 (**untrusted**) |
| Bootstrapping | 1,300 | 1,300 |
| Security Monitor | N/A | 5,200 (**trusted**) |
| **TCB size: Trusted code** | **11,600** | **5,200** |

- **TCB reduction: from 12K LOC to 5K LOC**
- Currently RAMDisk system
  - Could use flash to store tags for disk sectors

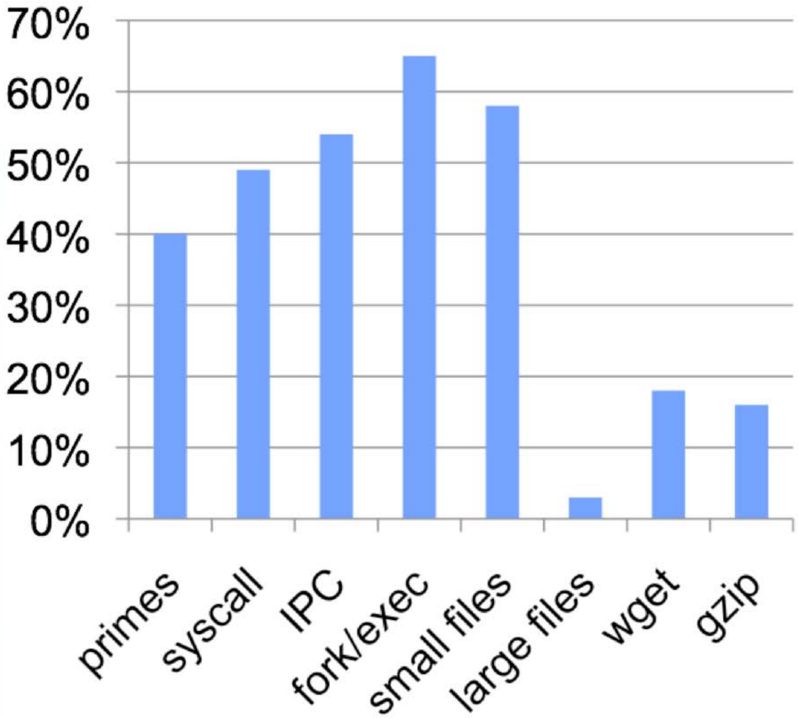
# LoStar Performance Evaluation



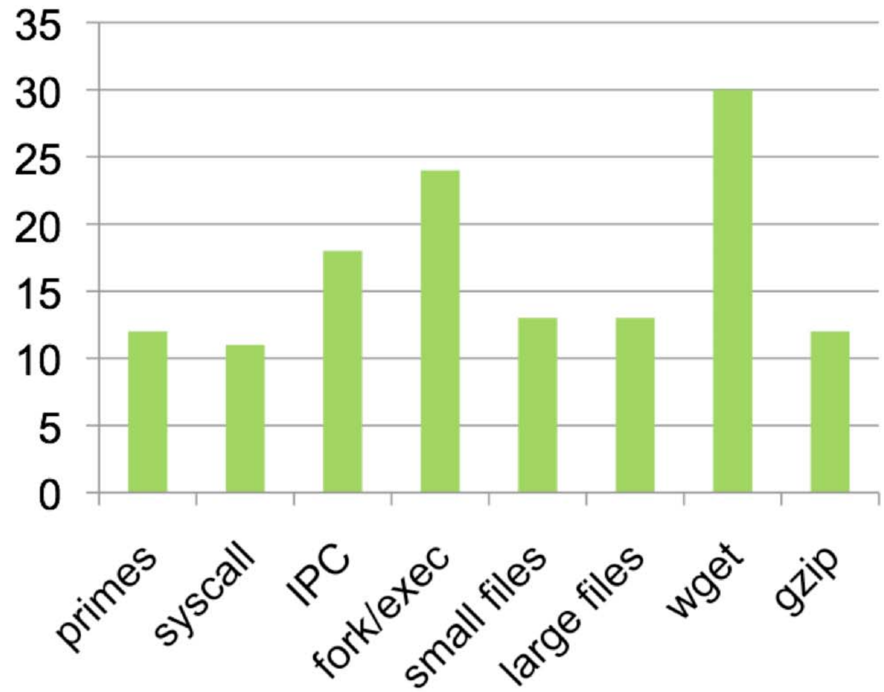

- Negligible performance impact

# Need for fine-grained tags

***Fraction of mem pages with word-level tags***




***Max number of concurrently accessed tags***




- Fine-grained tags are necessary
- Need many tags to protect users, processes, file descriptors

# Conclusions

- Loki hardware architecture using tagged memory
  - Enforces application security policies on data

- LoStar = Loki + HiStar
  - Provides fine-grained access control in hardware
    - Many independent protection domains
  - Protects user data from other malicious kernels
  - Reduces the TCB of HiStar by over a factor of 2
  - Provides strong security guarantees and good performance

# Questions?

- Want to use LoStar?
  - Let us know …
  - Loki port to Xilinx XUP board
    - $300 for academics
    - $1500 for industry
  - Full RTL + HiStar-SPARC distribution