


# **Information and Control in Gray-Box Systems**



**Andrea C. Arpaci-Dusseau**

**Remzi H. Arpaci-Dusseau**

**Department of Computer Science**

**University of Wisconsin--Madison**

# Motivation



- Problem: Many OS ideas not promptly deployed
- Reason: Modifying OS is difficult or impossible
  - Open-source code
    - | Large, complex body of code
    - | Convince others to trust your modifications
  - Closed-source code
    - | Convince others to implement functionality
    - | Motivating applications do not exist
- How to disseminate OS research without any changes to OS?

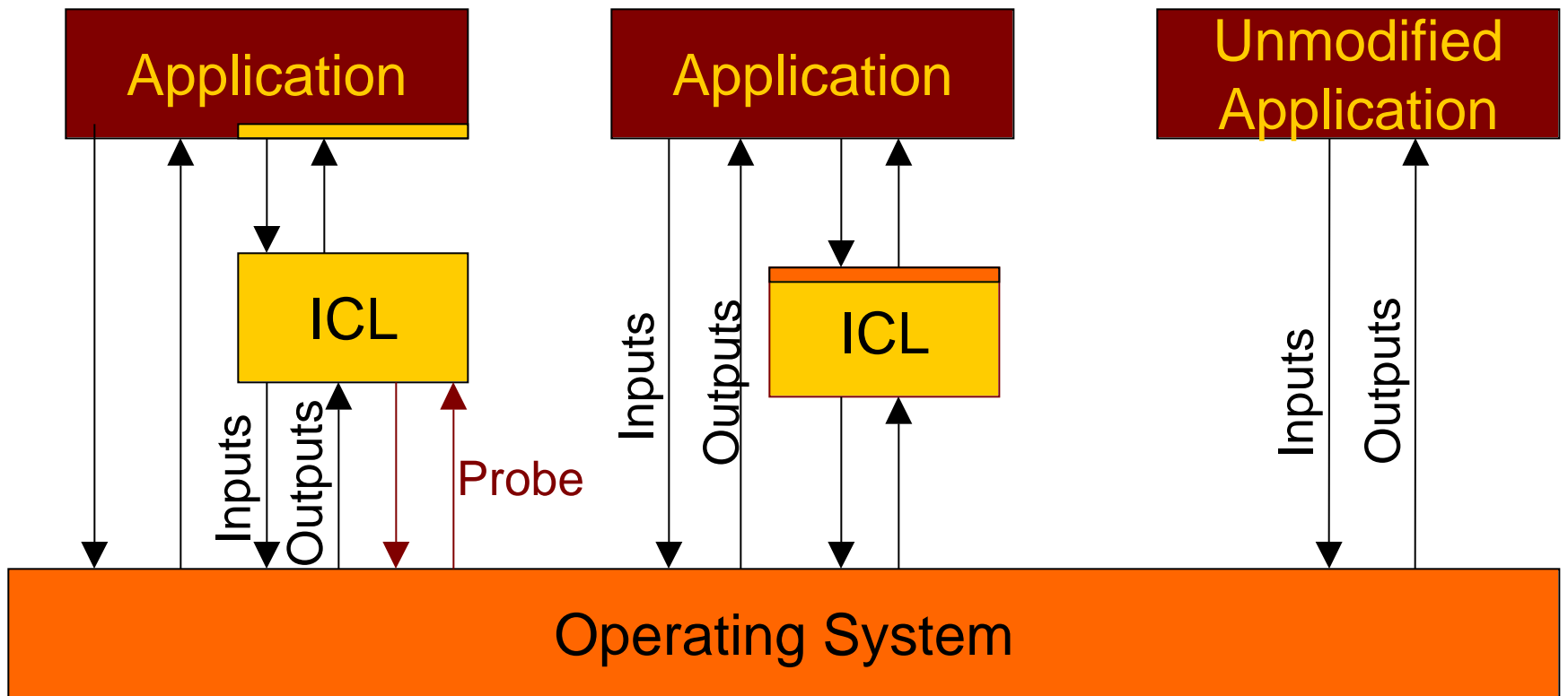
# Thesis



- Large class of “OS-like” services can be provided without modification
  - Can acquire **information** about internal state of OS
  - Can impose **control** on behavior of OS
- Key: Treat OS as **gray-box** component
  - Have knowledge of basic algorithms employed by OS
  - Infer state: Combine knowledge with observations
  - New control: Use knowledge for desired side-effects

# Approach

## ■ Information and Control Layers (ICLs)



# Outline



- *Motivation*
- Gray-Box Techniques
- ICL Case Studies
  - File-Cache-Content Detector (FCCD)
  - File-Layout-Detector and Controller (FLDC)
  - Memory-based Admission Controller (MAC)
- Conclusions and Future Work

# Gray-Box Techniques: Information



- Obtain knowledge of gray-box OS
  - Algorithmic: Code, documentation, or experience
  - Parametric: Micro-benchmark OS
- Monitor inputs and outputs of OS
  - Observe covert channels (e.g., time operation takes)
  - Insert probes (i.e., requests solely to observe output)
- Infer current OS state
  - Model or simulate OS given observations
  - Use simple statistics

# Gray-Box Techniques: Control



- Move OS to known state
  - Difficult to infer current state
  - Easier to model given known initial state
- Reinforce behavior via feedback
  - Difficult given arbitrary application behavior
  - Use ICL to manage interactions

# Outline



- *Motivation*
- *Gray-Box Techniques*
- **Case Studies**
  - File-Cache-Content Detector (FCCD)
  - File-Layout-Detector and Controller (FLDC)
  - Memory-based Admission Controller (MAC)
- **Conclusions and Future Work**



# ICL #1: File-Cache Content Detector (FCCD)

## ■ Goal

- Reorder I/O requests to first use data in file cache
- Similar to SLEDs

## ■ Proposed interface and usage

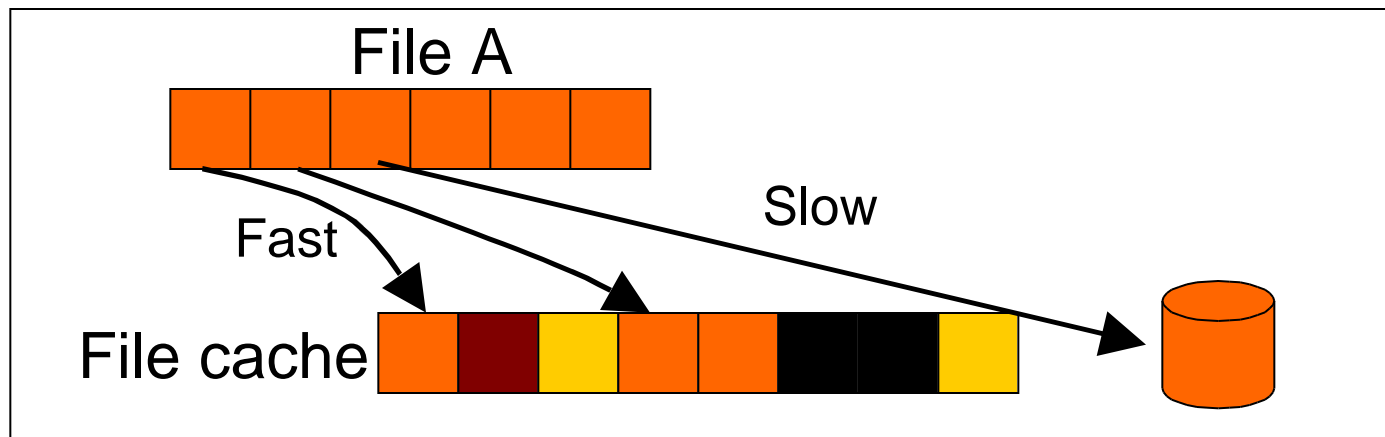
- Application specifies file or list of files to be accessed
- ICL returns list of (offset, length) pairs for in-cache data
- Application reorders accesses accordingly

## ■ Desired OS state information

- Which data blocks are in file cache?

# FCCD Approach: Probe to infer cache state

- Read byte from each requested block
- Measure time of access
  - Fast probe --> Block in cache
  - Slow probe --> Block not in cache

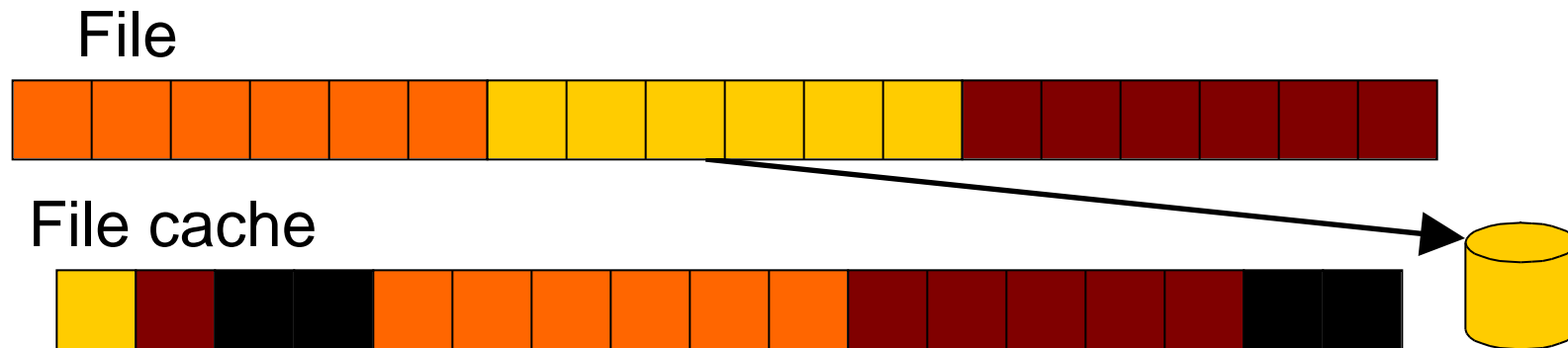


- Probes to disk have high overhead
- Probes are destructive (Heisenberg effect)

# File-Cache Probes:

## Low overhead, high accuracy

- Probe state must correlate w/ neighbors
- Algorithmic knowledge
  - Applications access files w/ temporal & spatial locality
  - Replacement policies are locality based



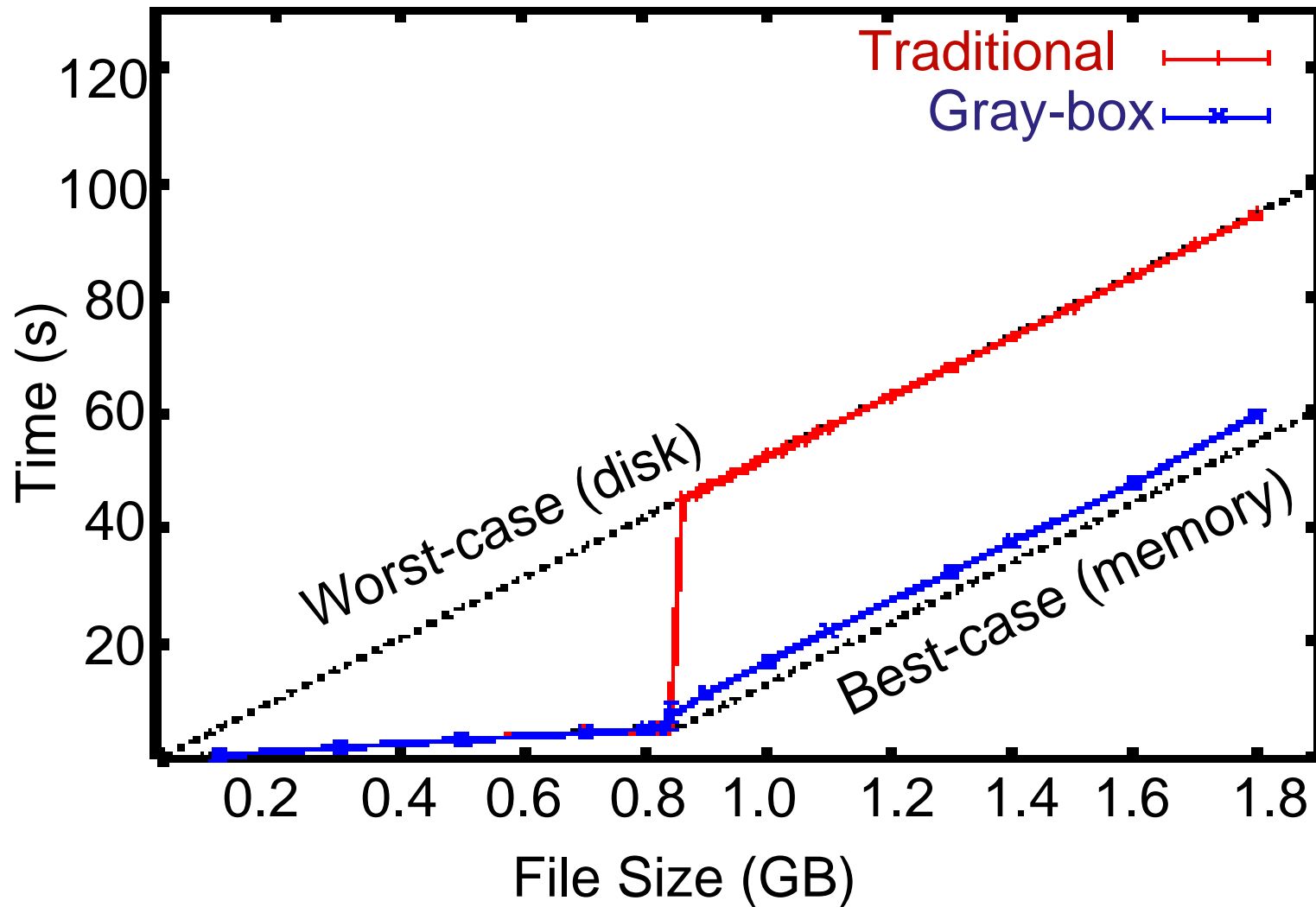
- Blocks of file replaced in contiguous regions
- A few probes predict state of entire region

# FCCD Challenges

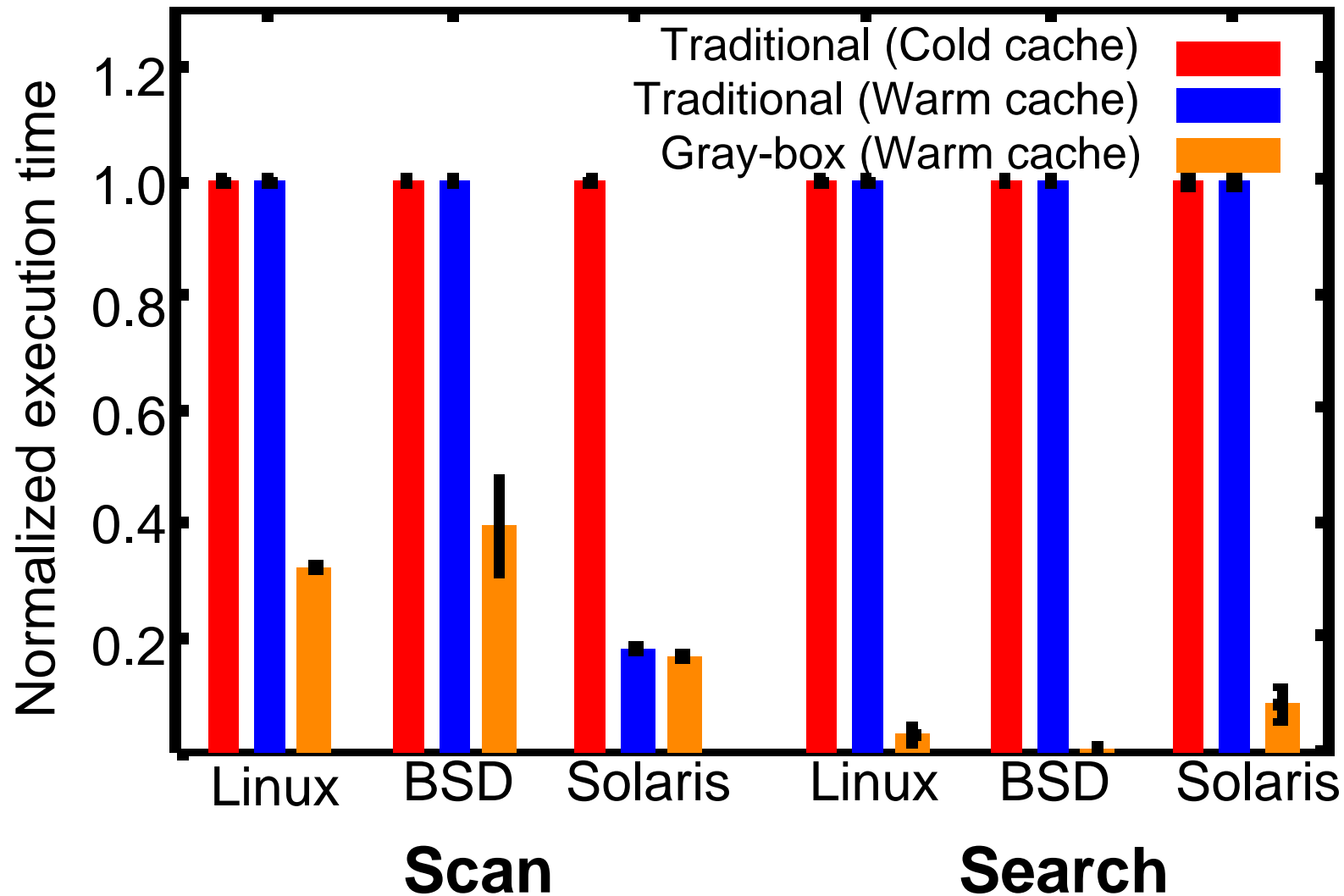


- #1: What is access unit of application?
  - Size of contiguous region    access unit of application
- Solution: Reinforce behavior with feedback
  - ICL returns blocks in unit of access
  
- #2: How to differentiate file cache hit from miss?
  - Want platform independence
- Solution: Sort by probe time
  - Handles multiple levels of storage hierarchy

# FCCD Evaluation: Single-File Scan on Linux



# FCCD Evaluation: Multi-Platform Performance



# FCCD Summary: Gray-Box Techniques



- Leverage algorithmic knowledge
  - State of block correlated with state of neighbors
- Insert probes
  - Measure time to read byte from page
  - Limit number due to Heisenberg effect
- Reinforce behavior with feedback
  - ICL determines access pattern of application

Works well on all 3 OSes

Limitation: Cannot predict very small files

# Outline



- *Motivation*
- *Gray-Box Techniques*
- **Case Studies**
  - *File-Cache-Content Detector (FCCD)*
  - **File-Layout-Detector and Controller (FLDC)**
  - **Memory-based Admission Controller (MAC)**
- **Conclusions and Future Work**



# ICL #2: File Layout Detector & Controller (FLDC)

- Goal: Reorder I/O requests to minimize seeks
  - Determine location of blocks on disk
- Gray-Box Techniques
  - Leverage detailed FFS algorithmic knowledge
    - Directory in cylinder group; Layout matches creation order
  - Insert probes
    - Call stat() to obtain I-node number, sort by I-node number
  - Move system to known state
    - Periodically refresh directory layout
- z Works well on 3 OS's, Composes with FCCD
- z Limitations: FFS-specific, Overhead of control

# ICL #3: Memory-based Admission Controller (MAC)

- Goal: Avoid thrashing memory system
  - Determine and allocate available memory
- Gray-Box Techniques
  - Leverage only high-level algorithmic knowledge
    - Page replaced when physical memory full
  - Insert probes
    - Measure time to write to increasing size
  - Move system to known state
    - Probe first to make pages resident, probe again to check
- z Handles multiple memory and I/O-intensive apps
- z Limitations: Only Linux, High probe overhead

# Conclusions and Future Work



- Gray-box approach
  - Migration path for new ideas
  - Combine gray-box knowledge with observations
  - Promising initial case studies
- Future Work
  - Understand limits of gray-box techniques
    - Evaluate more OS platforms and applications
    - Develop additional ICL case studies
  - Allow others to use gray-box techniques
    - Provide Gray Toolbox
    - Explore advanced gray-box techniques