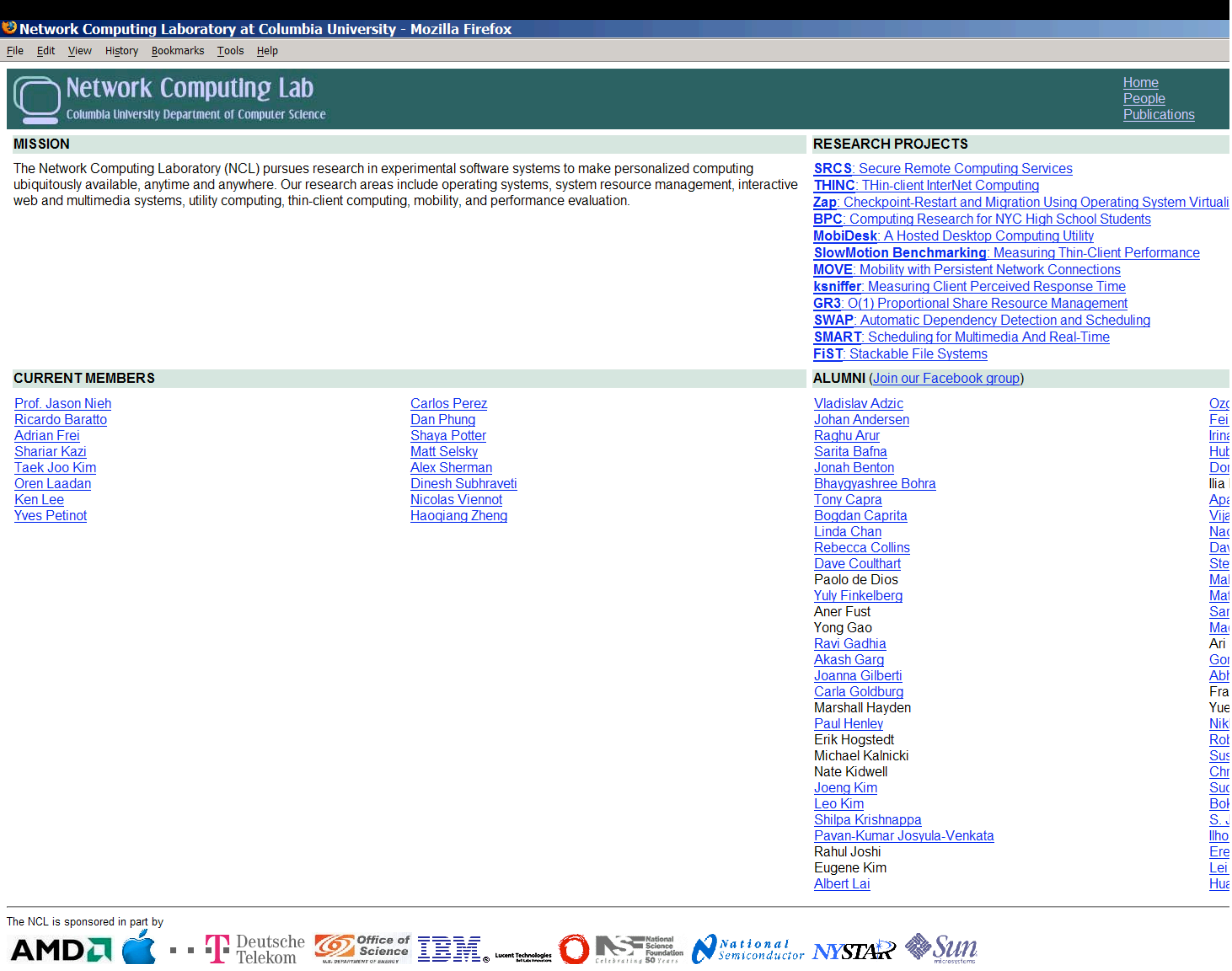
# DejaView: A Personal Virtua
# Computer Recorder

Jason Nieh

Columbia University


Network Computing Laboratory at Columbia University - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help

# Network Computing Lab

Columbia University Department of Computer Science

Home
People
Publications

## MISSION

The Network Computing Laboratory (NCL) pursues research in experimental software systems to make personalized computing ubiquitously available, anytime and anywhere. Our research areas include operating systems, system resource management, interactive web and multimedia systems, utility computing, thin-client computing, mobility, and performance evaluation.

## RESEARCH PROJECTS

**SRCS**: Secure Remote Computing Services
**THINC**: THin-client InterNet Computing
**Zap**: Checkpoint-Restart and Migration Using Operating System Virtuali
**BPC**: Computing Research for NYC High School Students
**MobiDesk**: A Hosted Desktop Computing Utility
**SlowMotion Benchmarking**: Measuring Thin-Client Performance
**MOVE**: Mobility with Persistent Network Connections
**ksniffer**: Measuring Client Perceived Response Time
**GR3**: O(1) Proportional Share Resource Management
**SWAP**: Automatic Dependency Detection and Scheduling
**SMART**: Scheduling for Multimedia And Real-Time
**FiST**: Stackable File Systems

## CURRENT MEMBERS

Prof. Jason Nieh
Ricardo Baratto
Adrian Frei
Shariar Kazi
Taek Joo Kim
Oren Laadan
Ken Lee
Yves Petinot
Carlos Perez
Dan Phung
Shaya Potter
Matt Selsky
Alex Sherman
Dinesh Subhraveti
Nicolas Viennot
Haoqiang Zheng

## ALUMNI (Join our Facebook group)

Vladislav Adzic
Johan Andersen
Raghu Arur
Sarita Bafna
Jonah Benton
Bhaygyashree Bohra
Tony Capra
Bogdan Caprita
Linda Chan
Rebecca Collins
Dave Coulthart
Paolo de Dios
Yuly Finkelberg
Aner Fust
Yong Gao
Ravi Gadhia
Akash Garg
Joanna Gilberti
Carla Goldburg
Marshall Hayden
Paul Henley
Erik Hogstedt
Michael Kalnicki
Nate Kidwell
Joeng Kim
Leo Kim
Shilpa Krishnappa
Pavan-Kumar Josyula-Venkata
Rahul Joshi
Eugene Kim
Albert Lai
Ozc
Fei
Irina
Hub
Dor
Ilia
Apa
Vija
Nac
Dav
Ste
Mal
Mat
Sar
Mac
Ari
Gor
Abh
Fra
Yue
Nik
Rob
Sus
Chr
Suc
Bok
S. J
Ilho
Ere
Lei
Hua

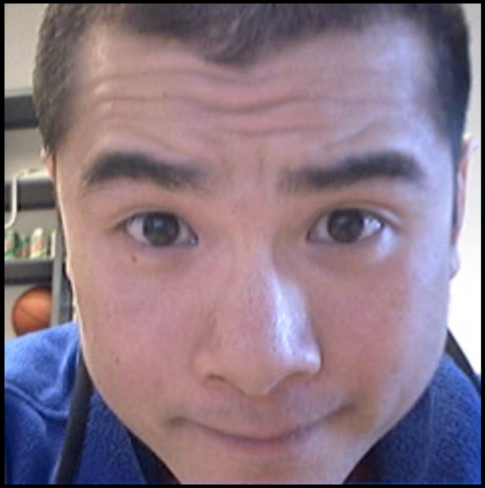
The NCL is sponsored in part by

AMD Deutsche Telekom Office of Science U.S. DEPARTMENT OF ENERGY IBM Lucent Technologies National Science Foundation Celebrating 50 Years National Semiconductor NYSTAR Sun microsystems

Oren
Laadan

Ricardo
Baratto

Dan
Phung



Shaya
Potter

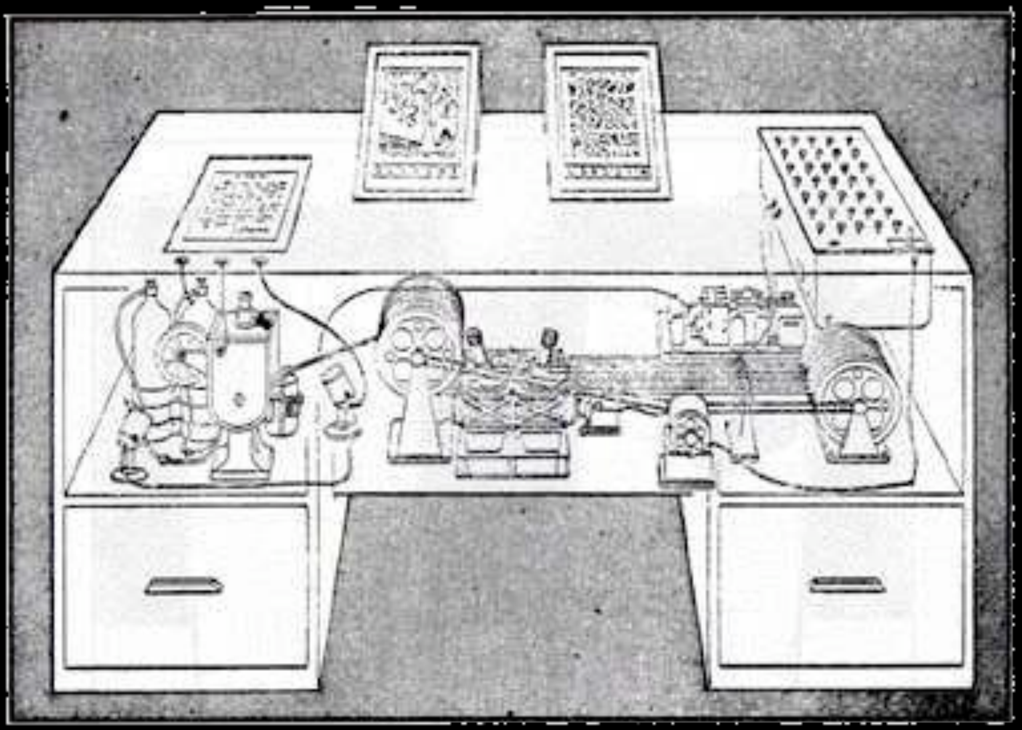
# The MEMEX Vision

“A device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility.”

Vannevar Bush, "As We May Think", July 1945

# The MEMEX Machine

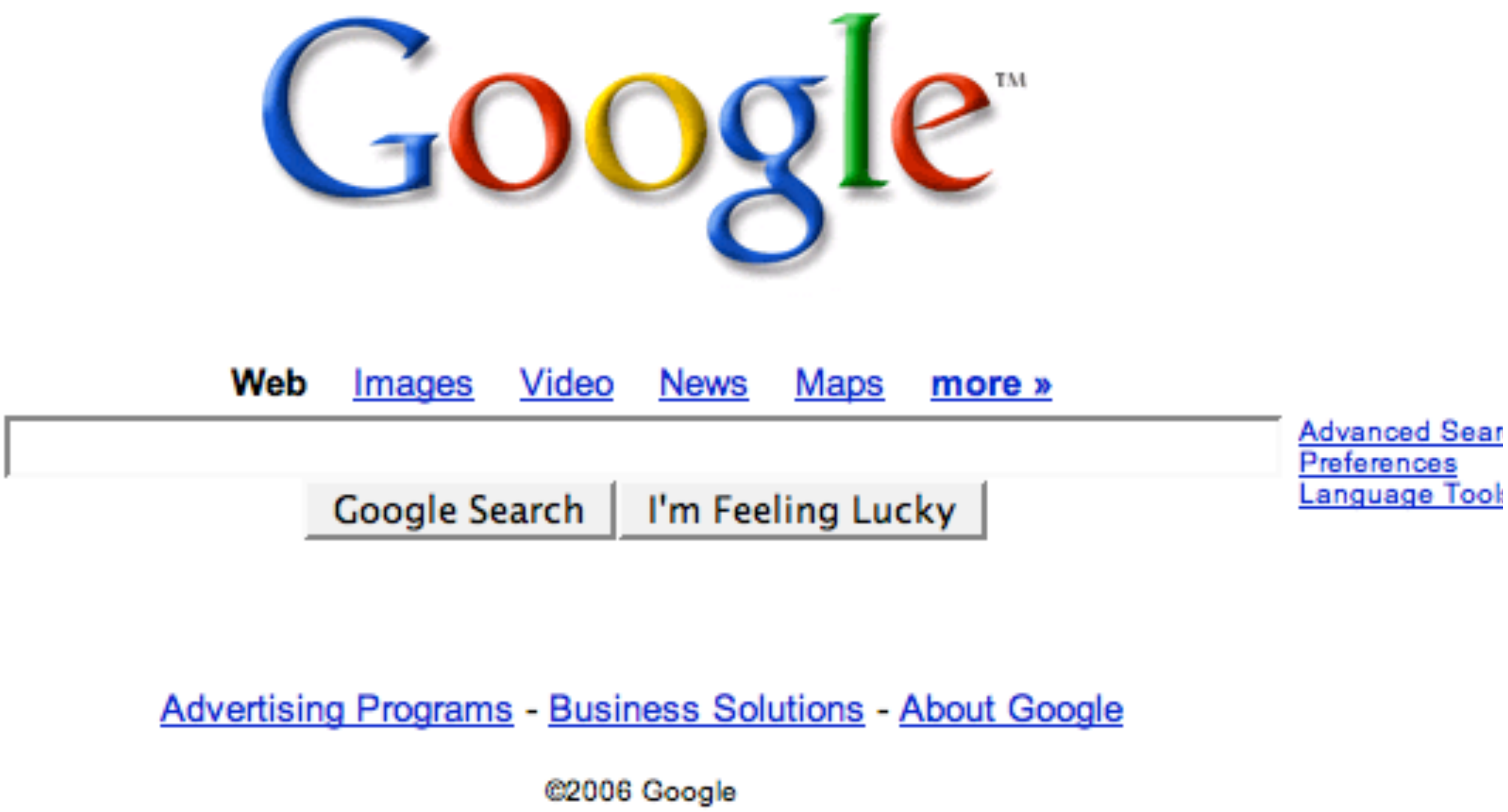
“It is an enlarged intimate supplement to his memory.”



# Today

- It is important to archive, search, view and manipulate what we have **seen**

# Are We There Yet ?



Web Images Video News Maps more »

Google Search I'm Feeling Lucky

Advanced Sear
Preferences
Language Tool

Advertising Programs - Business Solutions - About Google

©2006 Google

# Are We There Yet ?

# DejaView

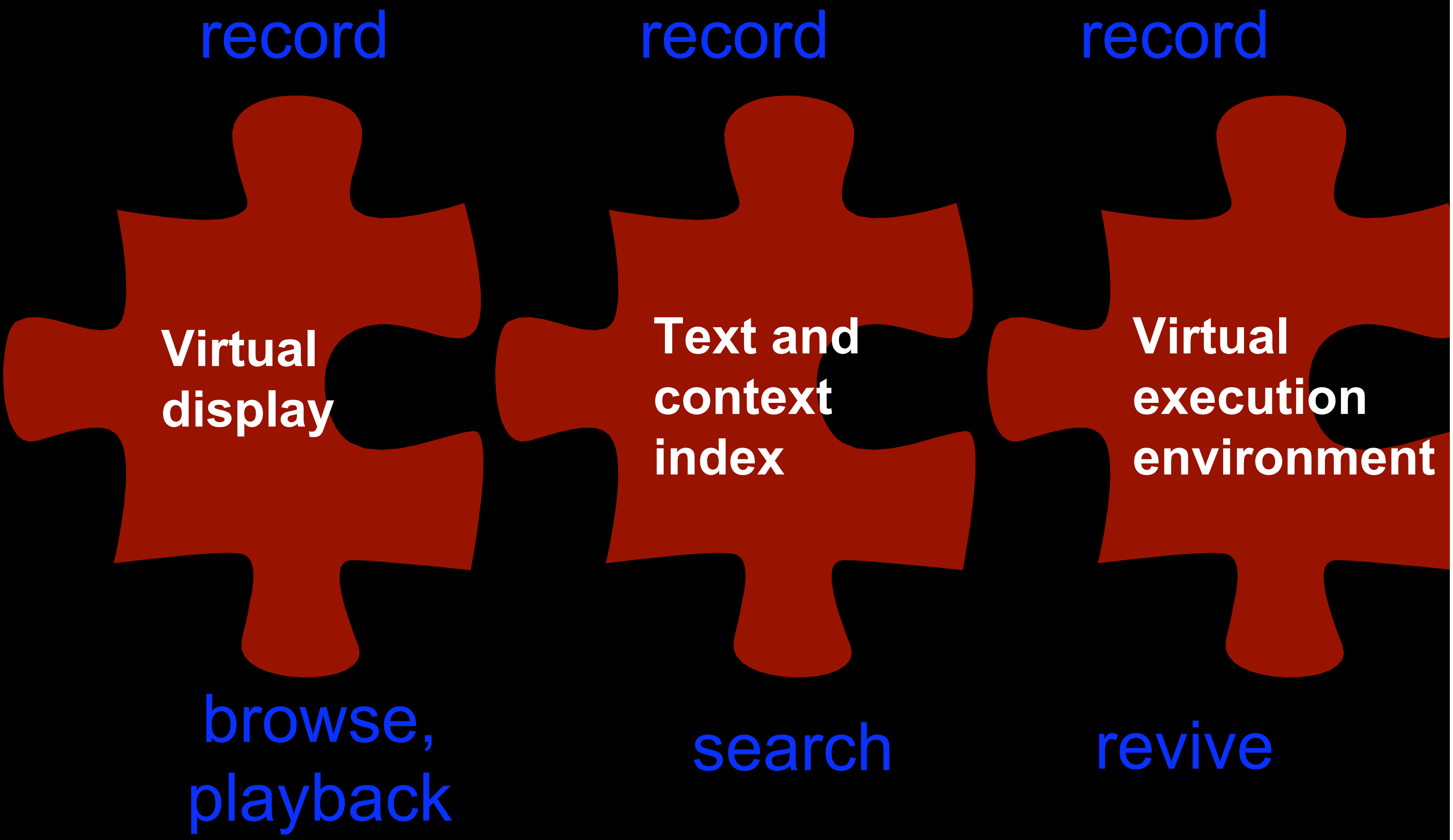
- A Personal Virtual Computer Recorder that provides a complete recording of a desktop computing experience
  - designed for transparency
  - fast enough for interactive use

# DejaView

- Provides a Tivo-like experience for the user's desktop
  - record display
    - to playback, browse, fast-forward, rewind
  - record text and context
    - to use as index to search the display record
  - record execution state
    - to revive and manipulate previous sessions

# DejaView Architecture

# DejaView Architecture



record

Virtual display

browse, playback

record

Text and context index

search

record

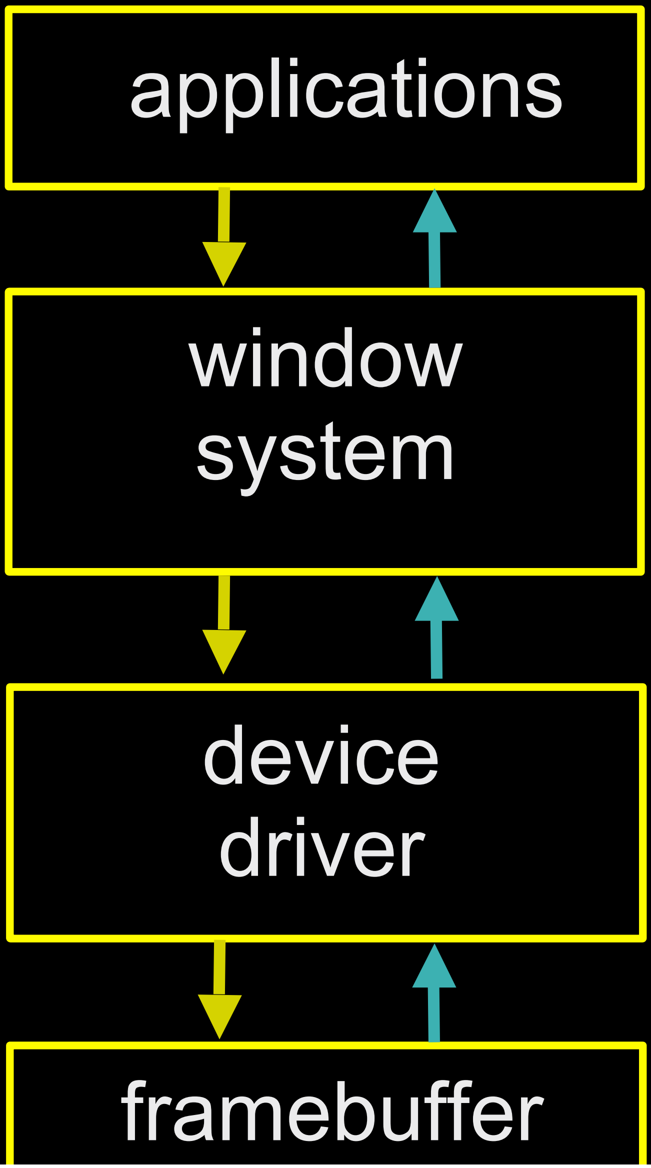
Virtual execution environment

revive

# Display Recording

- Need to record the display ...
  - transparently
  - efficiently
  - at full-fidelity

# Display system



applications

window system

device driver

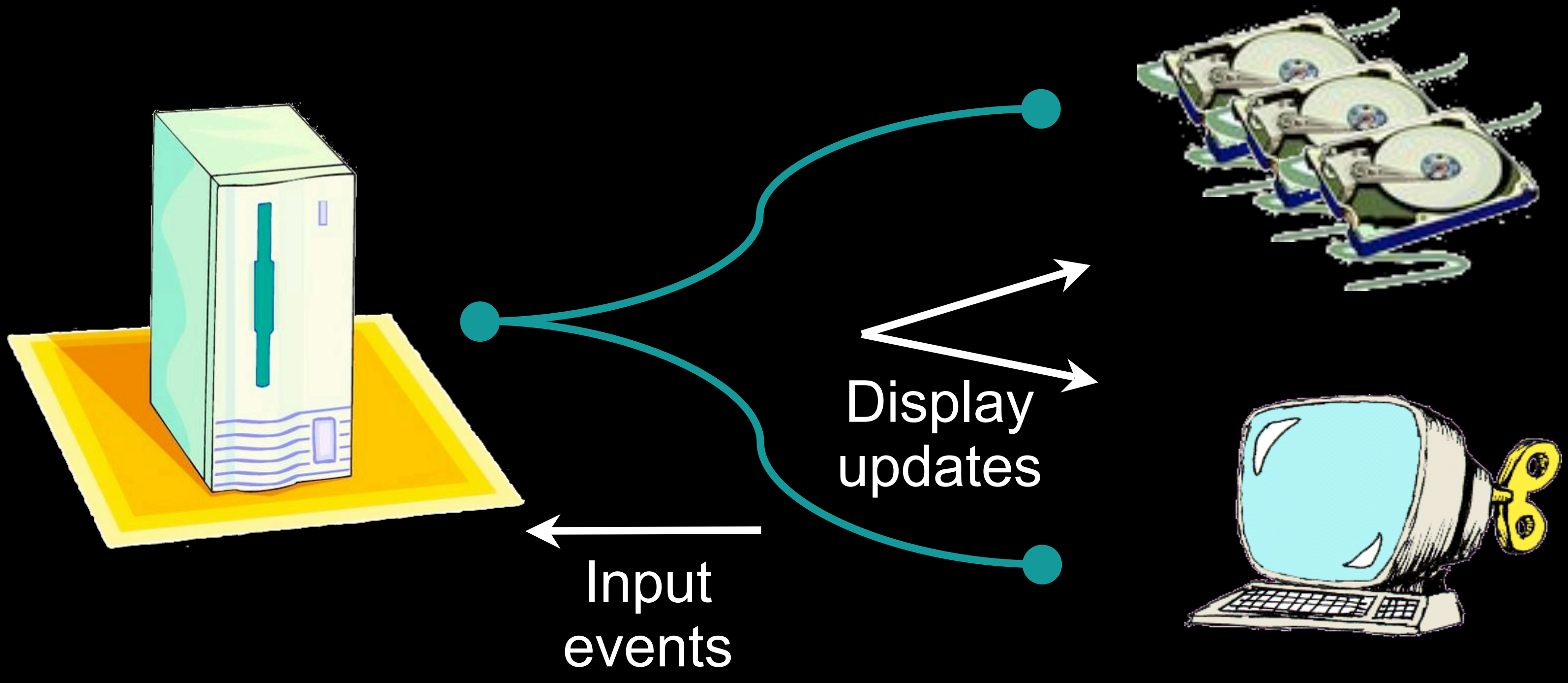
framebuffer

# Possibilities

- Window system commands?
  - complex, non-determinism
  - network limits
- Pixels?
  - high bandwidth
- MPEG?
  - high overhead
  - loss of display fidelity

# DejaView Approach

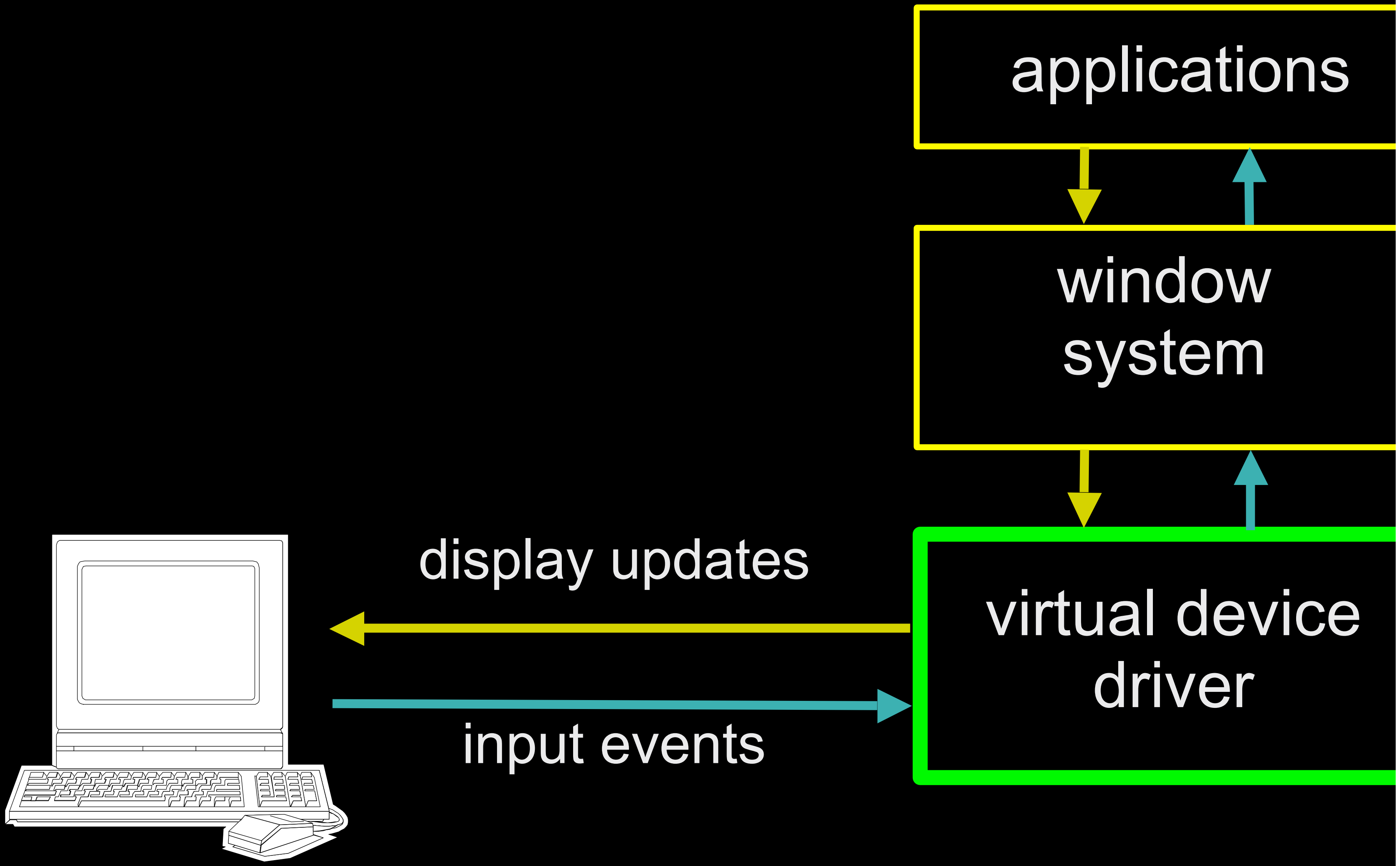
- Virtual display driver
  - no longer tied to a piece of hardware
  - can redirect the display anywhere



Display updates

Input events

# Virtual Display



applications

window system

virtual device driver

display updates

input events

# Virtual Display

- Standard device interface
  - provides full transparency
- Intercepts low level display updates
  - records only changes
  - fast, efficient, optimized for desktop
- Logs all display updates
  - no loss of information

# Text and Context Recording

- Need to record the text and context ...
  - retain semantics
  - transparently
  - efficiently

# Possibilities

- Window system commands?
  - not enough information
- OCR?
  - too slow
  - inaccurate

# DejaView Approach

- Leverage accessibility infrastructure
  - used by screen readers to convert text to speech, for the visually impaired
  - available on most modern desktops
  - incorporated into standard GUI toolkit

**already does what we need !**

# Accessibility Interfaces

- Accessibility infrastructure
  - standard interface – transparent
  - efficient – see evaluation
- Provides useful contextual information about the contents, e.g:
  - name and type of application
  - which window has focus
  - special properties (e.g. menu text)

# Execution Recording

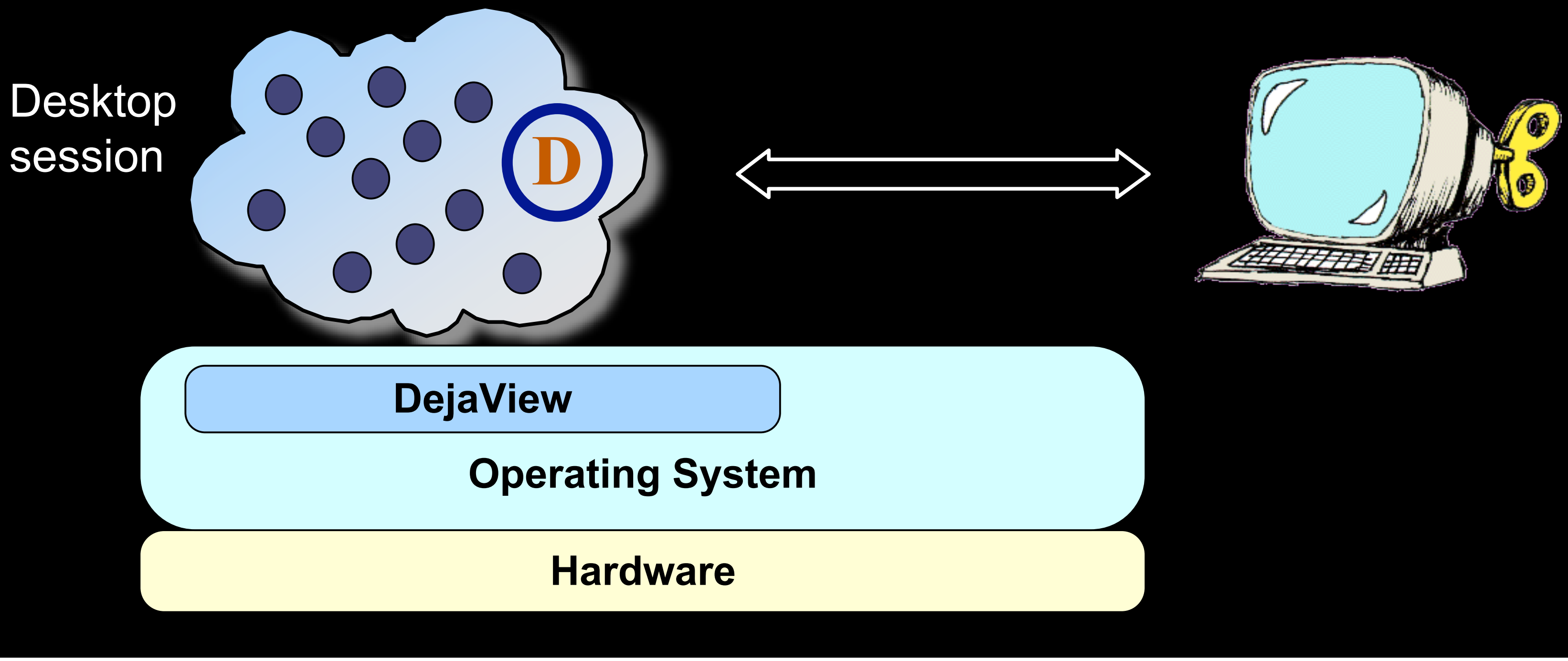
- Need to record execution state...
  - to be able to revive at later time
    - underlying system may change
  - include the entire desktop session
    - not only a single process
  - fast enough to save frequently
  - without degrading user experience

# Possibilities

- Checkpointing using VMMs?
  - too slow
  - too much state
- Log and replay?
  - need to replay from the middle
  - SMP too hard/slow in practice

# DejaView Approach

- Encapsulate only the user's desktop and decouple it from the underlying OS
  - repeatedly checkpoint the desktop session to be able to revive at a later time



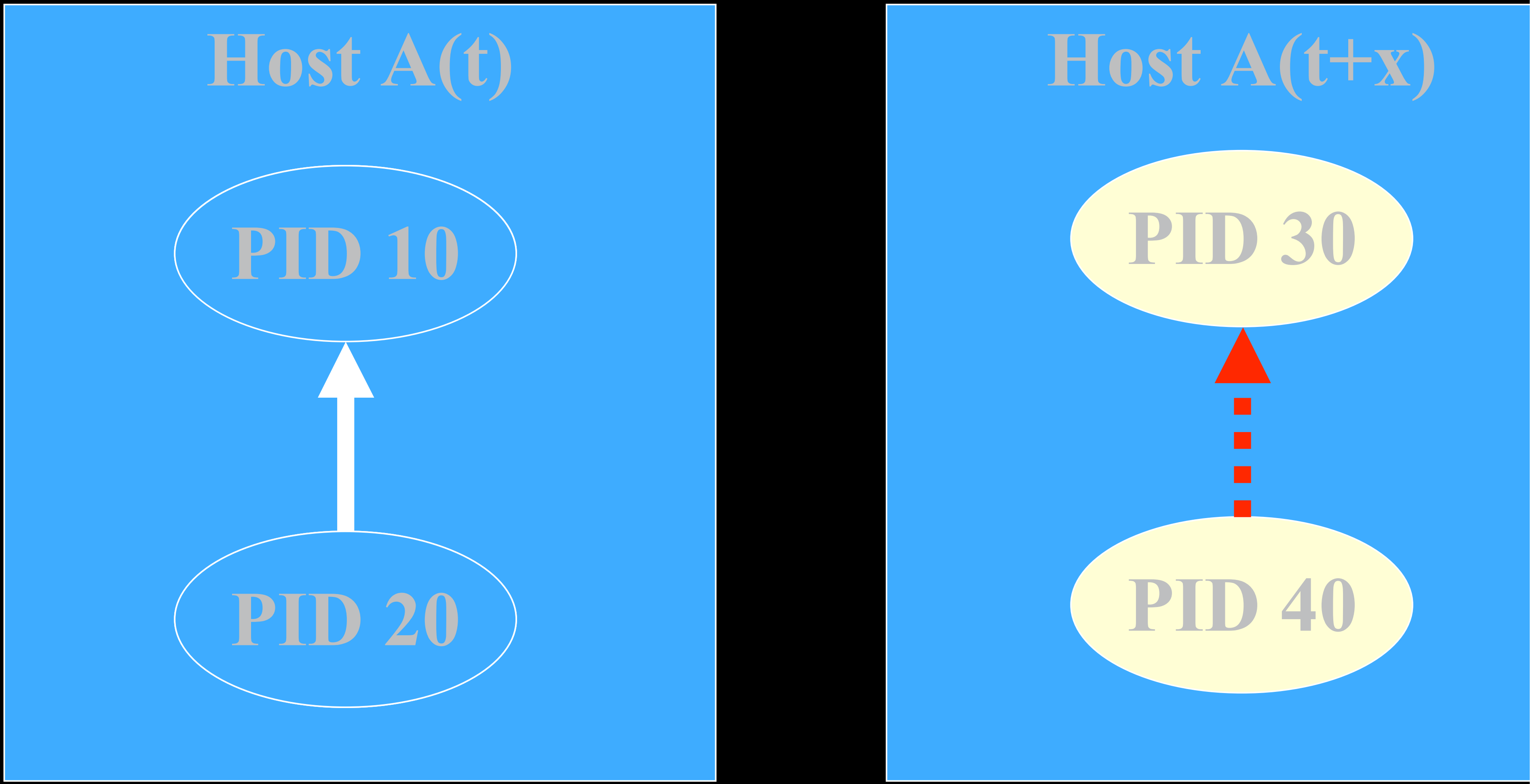
# Challenges

- Desktop consists of  multiple processes
  - processes have dependencies
  - processes are a moving target
  - need to capture globally consistent state
- Need to transparently support large existing installed application base

# Problem

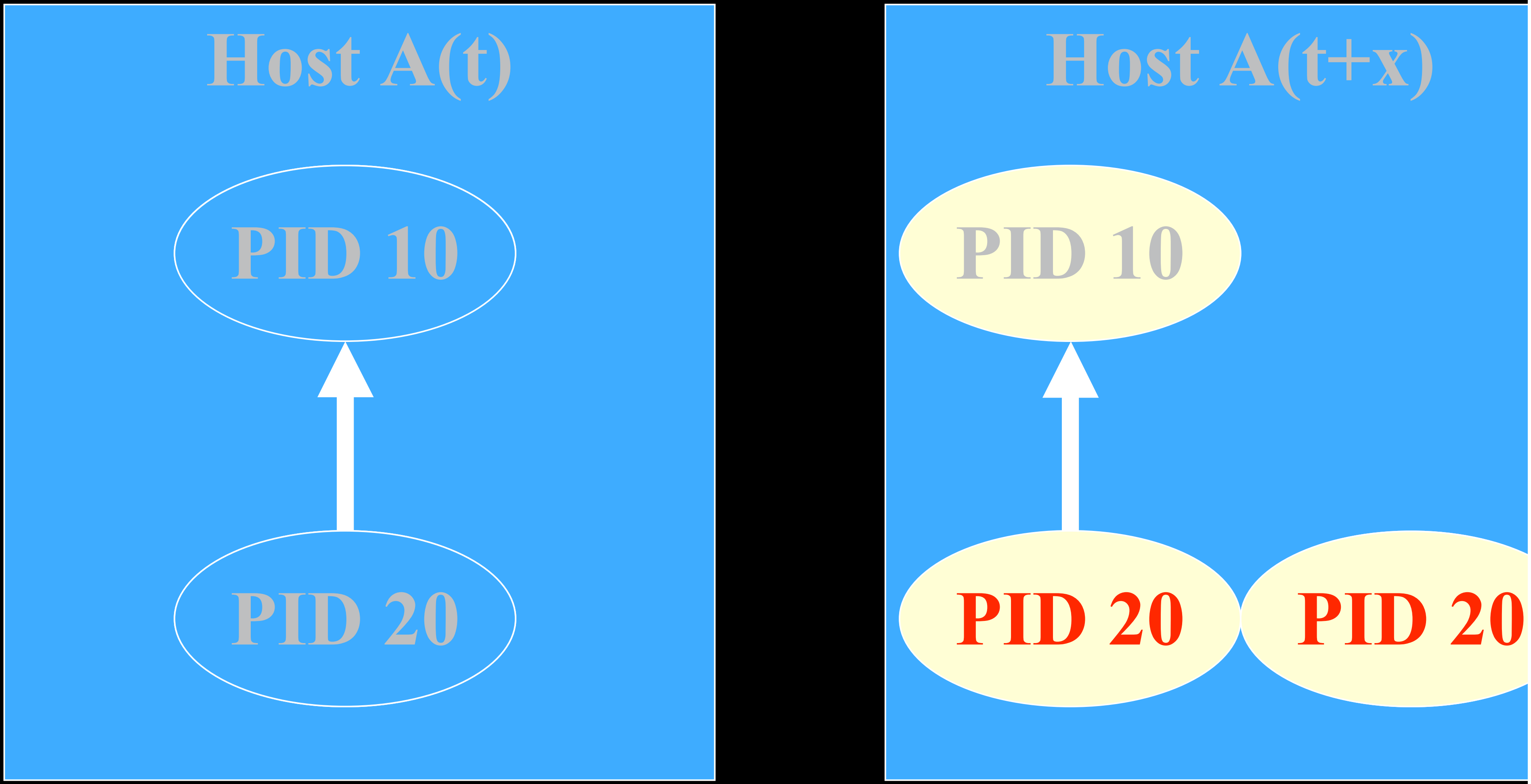
```
int iChildPID;

if (iChildPID=fork()) {
   /* parent does some work */
   waitpid(iChildPID);
} else {
   /* child does some work */
   exit(0);
}
```

# Resource consistency problem



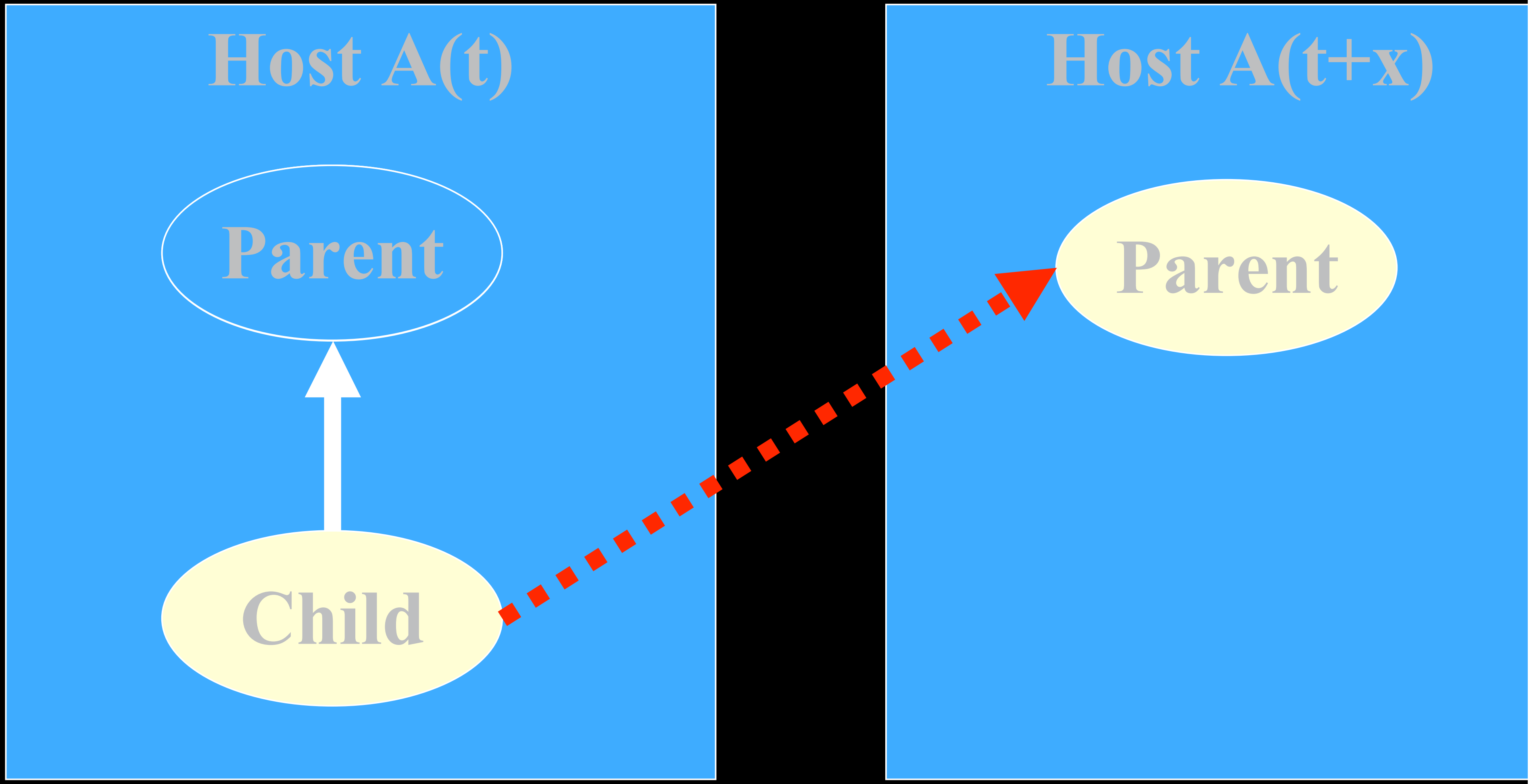
**Parent invoked waitpid(20)**

# Resource conflict problem



**Resources May Conflict With Other Processe**

# Resource dependency problem



**Parent and child depend on each other**

# Problem recap

resource consistency

- names can’t change

resource conflict

- names can’t be duplicates

resource dependency

- checkpoint must be complete

# Pod solution

- POD (PrOcess Domain)
- can contain any number of processes
- migrated as a unit
- ***private virtual*** namespace

# PID and IPC key virtualization

- create unique namespace for the pod
- names are virtualized
- when entering a system call, replace pod virtual identifiers with real ones
- when exiting a system call, replace real return values with pod virtual ones
- mask out identifiers that do not belong to the pod

# Memory virtualization

- like IPC, create unique shared memory namespace
- modern architectures support virtual memory

# Desktop POD

- Desktop PrOcess Domain (POD)
  - encapsulate user's desktop
- Private, virtual namespace
  - level of indirection
  - isolated, self-contained

# Virtual Execution Environment

- Interpose on operating system API
  - transparent, lightweight
- Operating system virtualization
  - confine dependencies among processes
  - remove dependencies on OS instance

# Execution Checkpoint

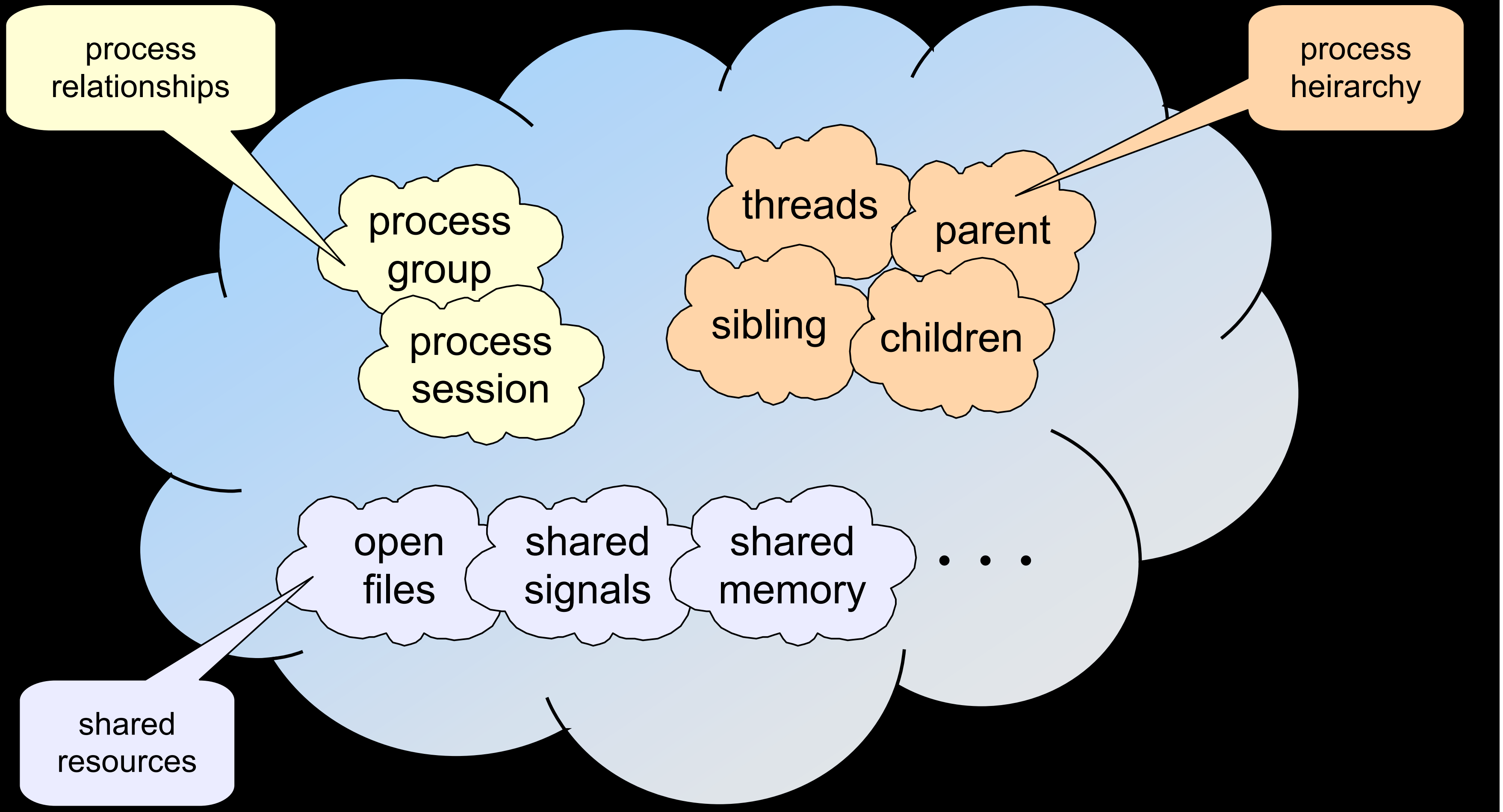
- Auxiliary checkpoint process
- Consistent checkpointing
  - (1) quiesce session
  - (2) save execution state
  - (3) save file system state (snapshot)
  - (4) let session resume

# Quiescing the POD

- Freeze processes
  - ensure global consistency
- Put processes in a known state
  - easy to restore
- Use native SIGSTOP
  - forced known state with minimal stack
  - synchronization handled natively
  - watch out for visible side-effects

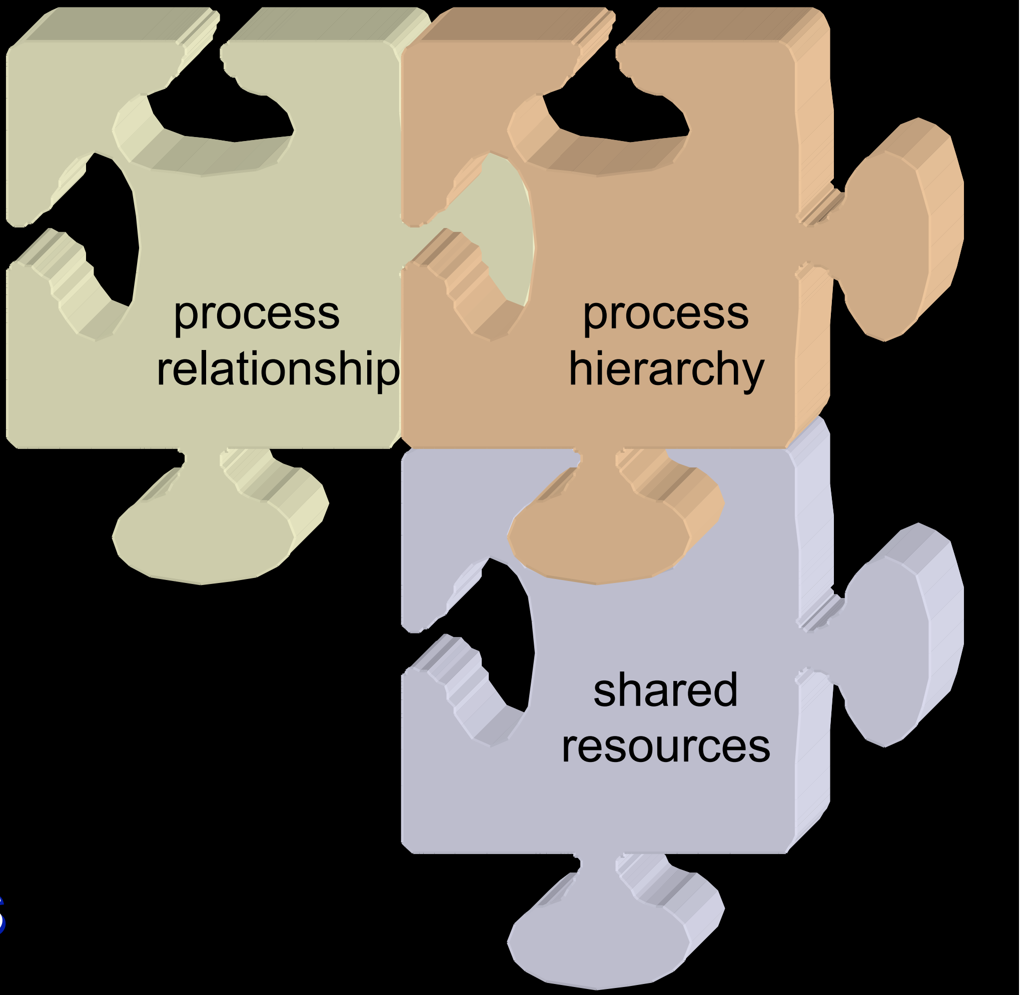
# Save Execution State

## Process Dependencies



# The Process Forest

- A-priori
  - parent-child
  - session
  - threads
- A-posteriori
  - process group
  - shared resources



# DumpForest Algorithm

- The algorithm records the state of the process forest in a consistent manner
- Goal: find creator, not just parent
- Input: available state at the time of the checkpoint
  - no logging or replay of events
- Output: a table that will hold a set of instructions to recreate the forest

# Save File System State

- Leverage log-structured file system
  - every transaction results in a snapshot

# Optimize for Interactivity

- Remove work from critical path:
  - pre-quiesce
  - pre-snapshot
  - incremental checkpoint
  - copy-on-write
  - deferred write-back

# Checkpoint Policy

- Only checkpoint on display updates
  - this is what interests the user
- Only when there are enough updates
  - skip unnecessary checkpoints to reduce storage requirements
- Limit checkpoint rate
  - so runtime overhead is manageable

# Reviving Execution

- Revive to a previous checkpoint
  - (1) restore file system state
  - (2) restore execution state
  - (3) let session resume

# Restore file system state

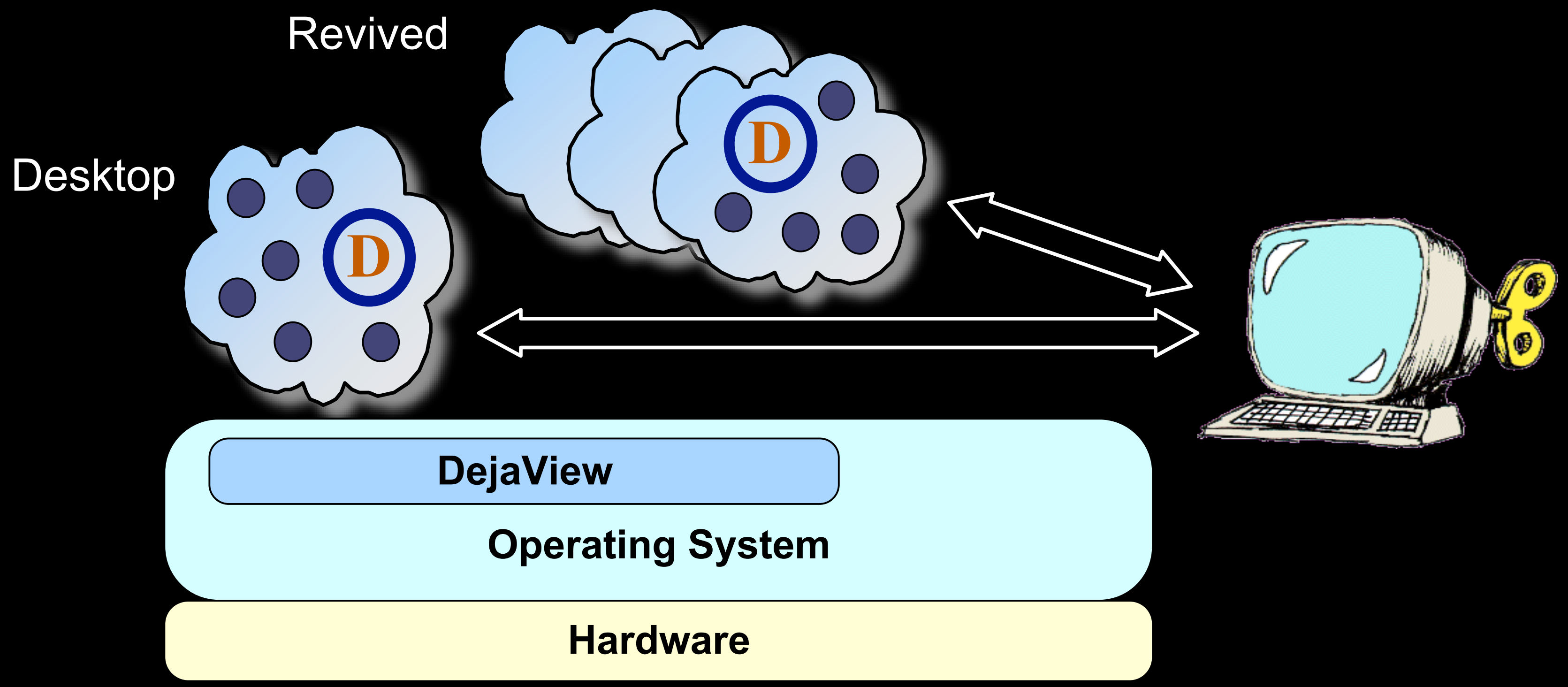
- Leverage union file system
  - combine the read-only snapshot with a fresh read-write file system layer on top

# Restore execution state

- In-context
- Restore process forest
  - leverage existing process creation functionality
- Restore process state
- Resume session

# Parallel Worlds

- Revived session has own environment
- Multiple sessions can run concurrently



# DejaView Performance

# Implementation

- X windows virtual display driver
- GNOME accessibility infrastructure
- Tsearch with PostgreSQL
- User-space utilities and Linux kernel module

No application, window system, or base kernel changes

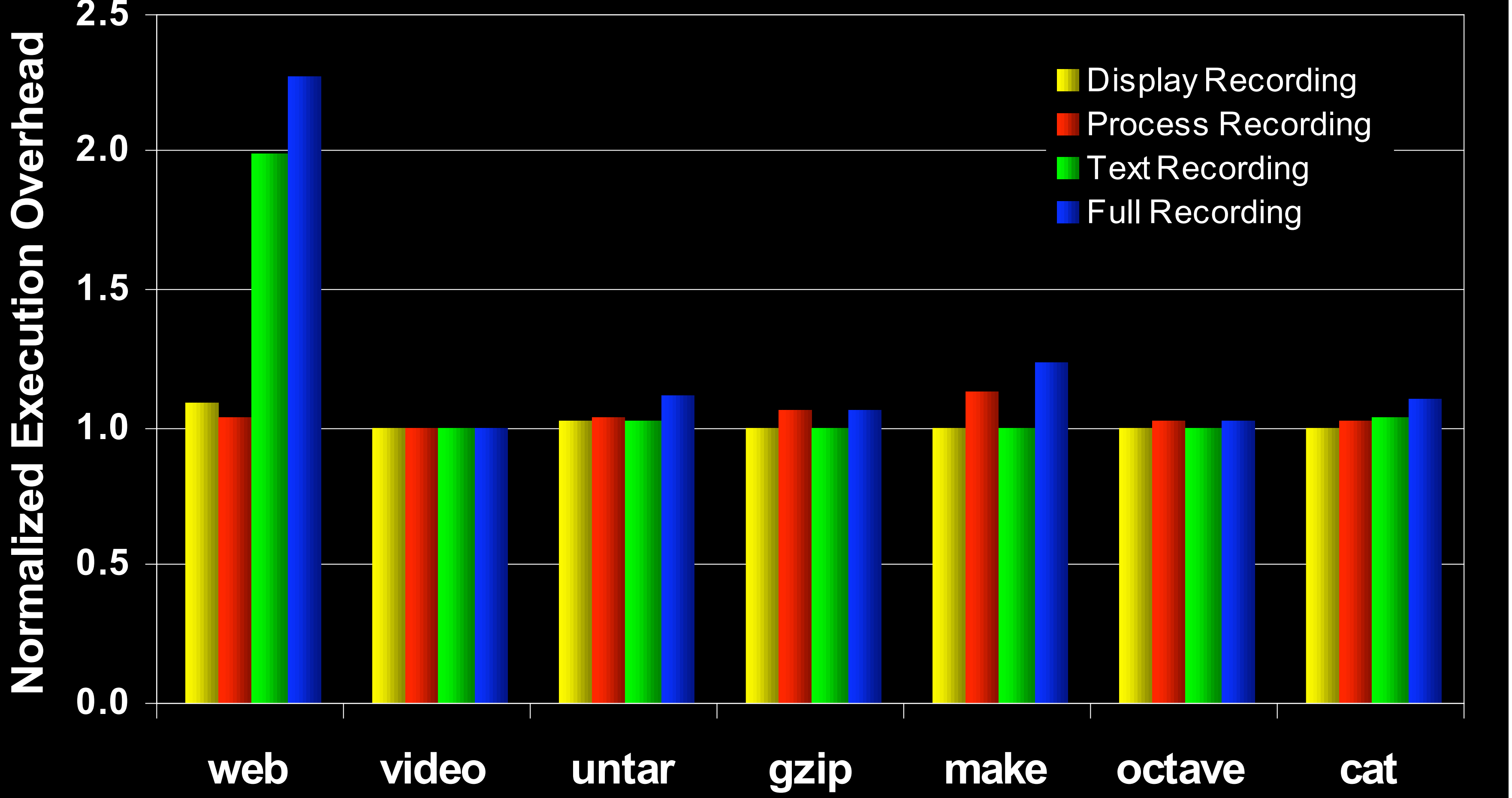
# Performance Evaluation

- System overhead:
  - runtime overhead of recording
  - impact on system interactivity
  - storage requirements
- Access to data:
  - search and browse latency
  - playback speed
  - session revive latency

# Application Scenarios

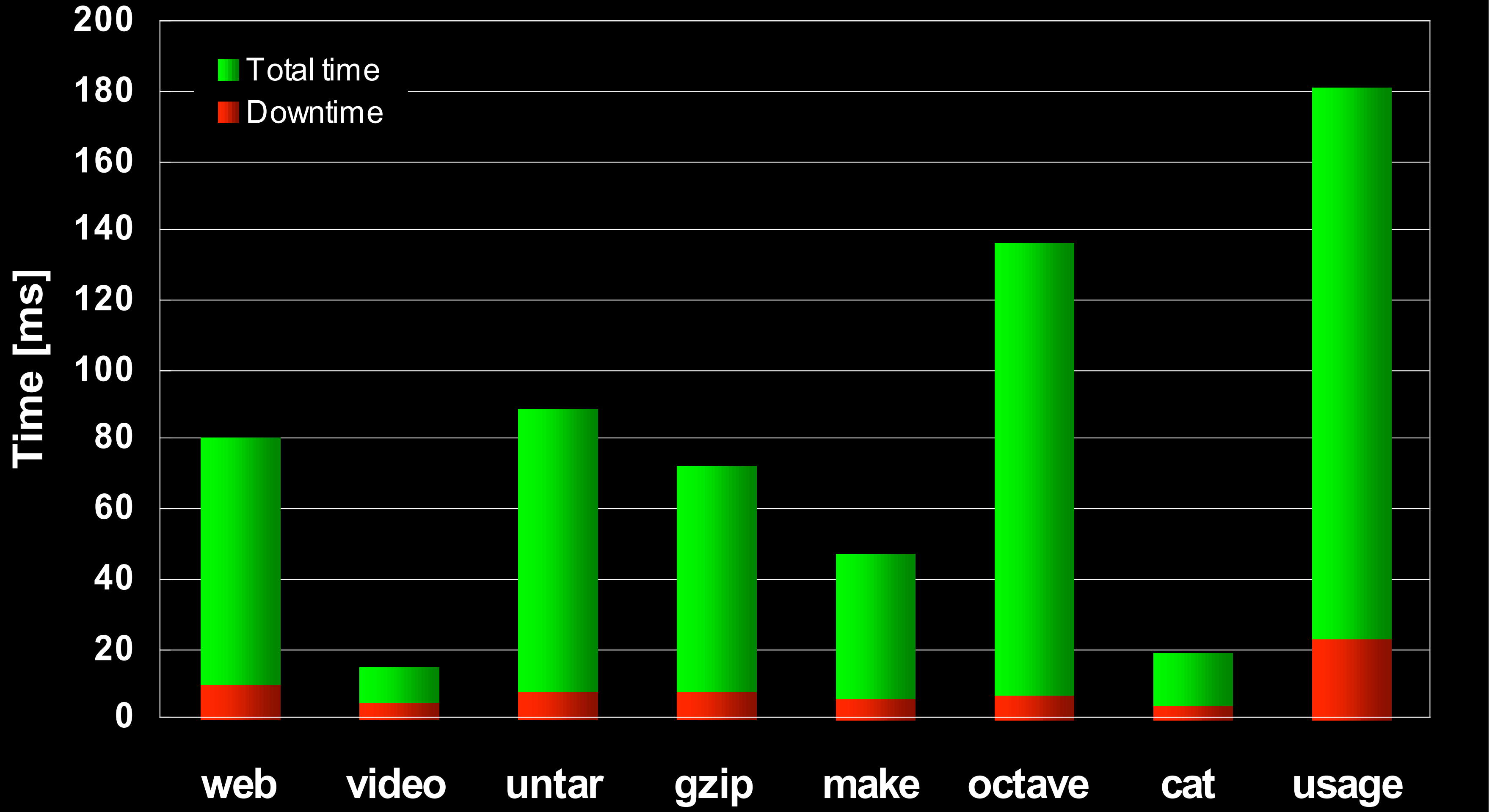
- Benchmarks
  - web - rapid-fire browsing
  - video - full screen playback
  - untar - untar of kernel source files
  - gzip - compress kernel source tar file
  - make - kernel make
  - octave - matlab clone calculation
  - cat - cat of a large file to screen
  - **usage - real desktop usage**

# Recording Runtime Overhead



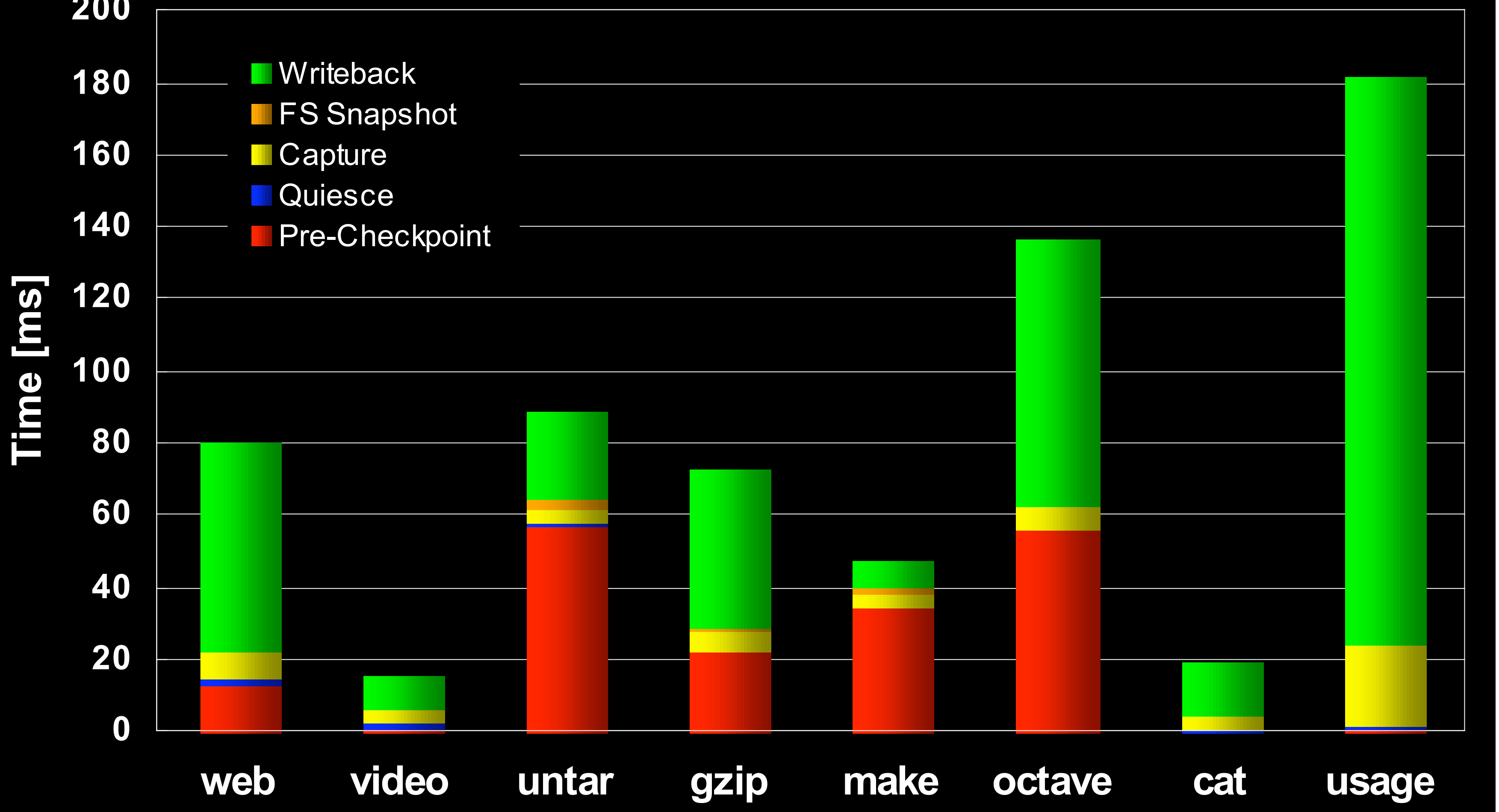
- Display and execution recording overhead is low

# Checkpoint Latency

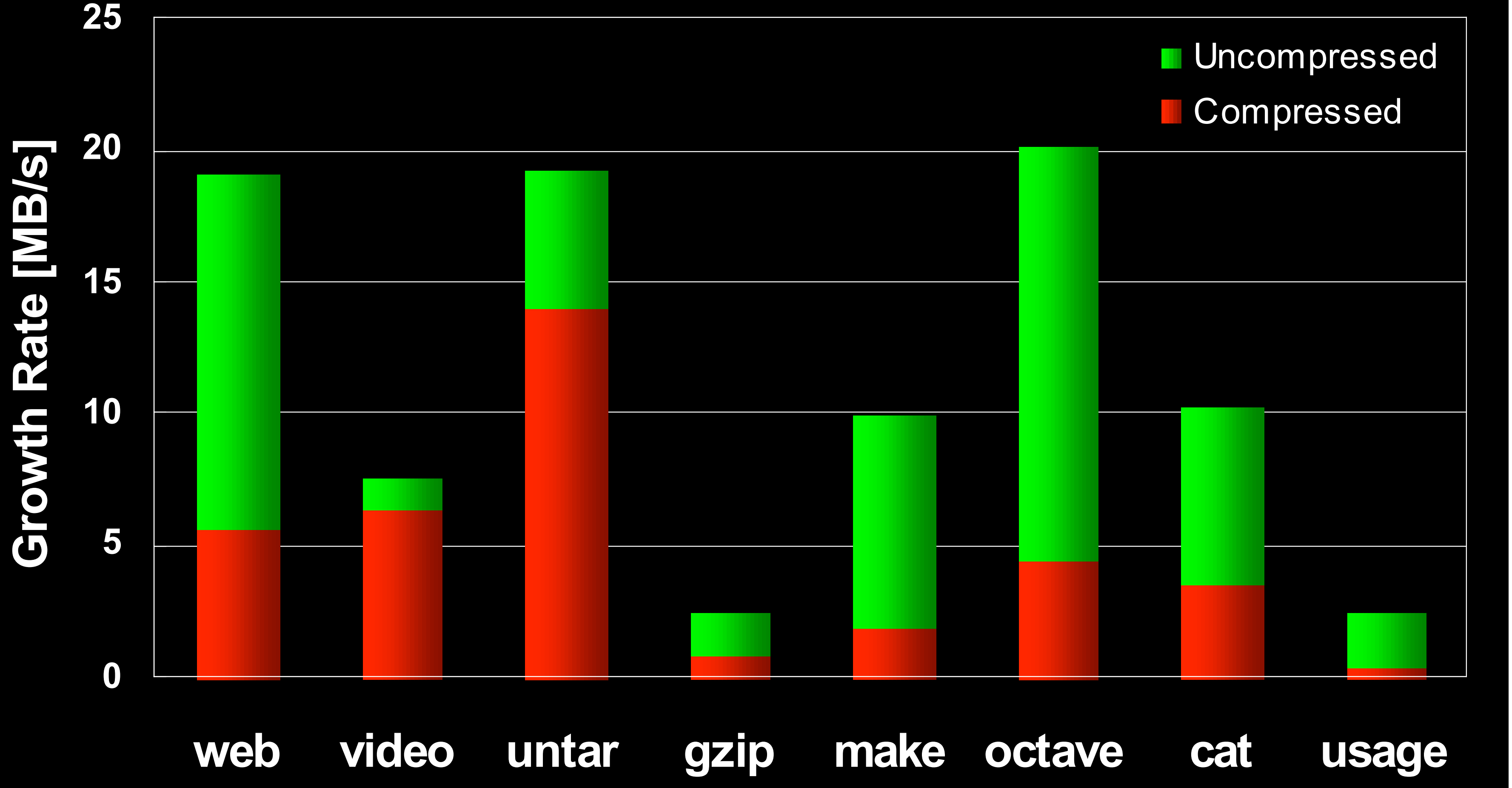


- Downtime low enough for interactive usage
- Total time low enough for frequent checkpoints

# Checkpoint Latency

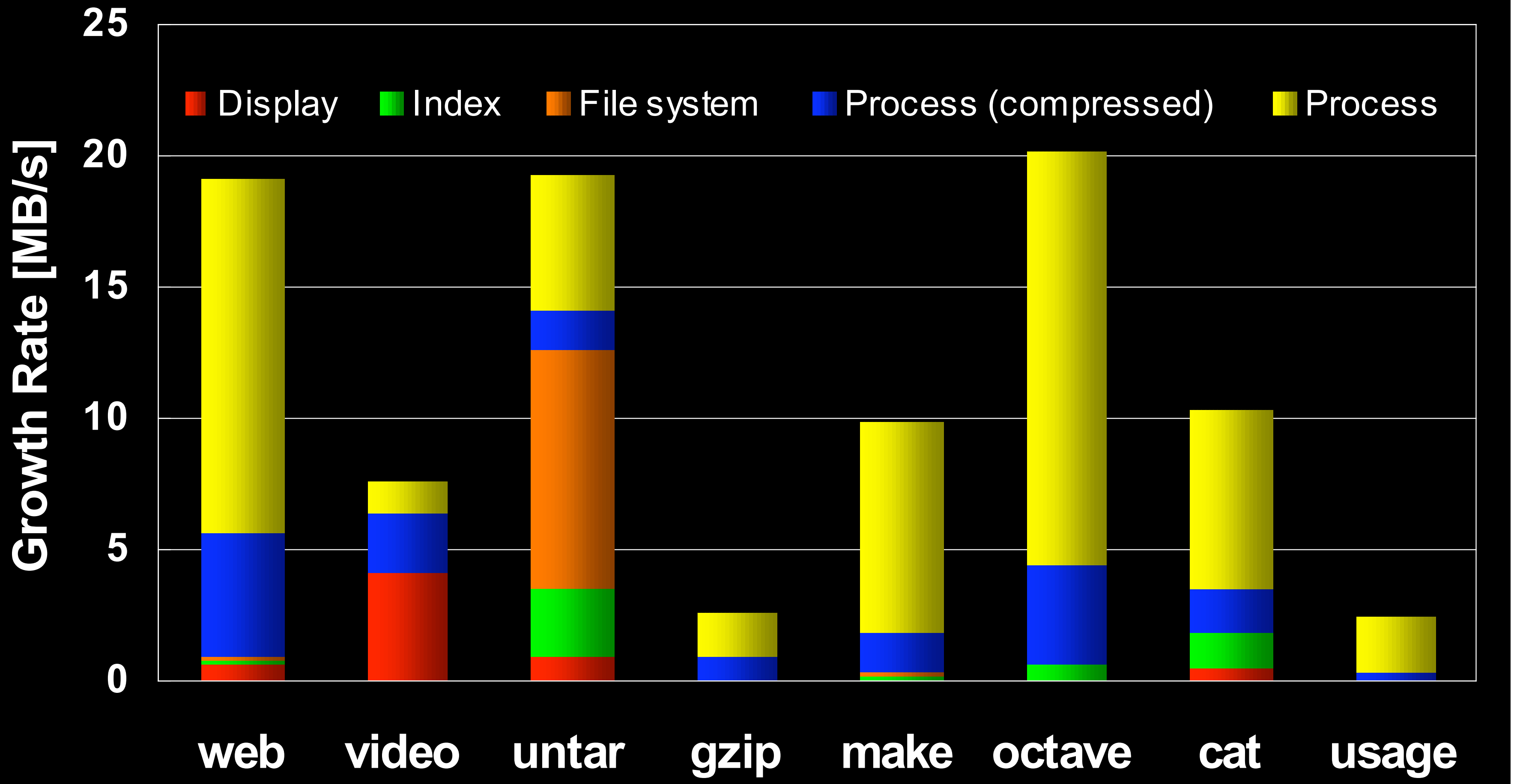


# Recording Storage Growth

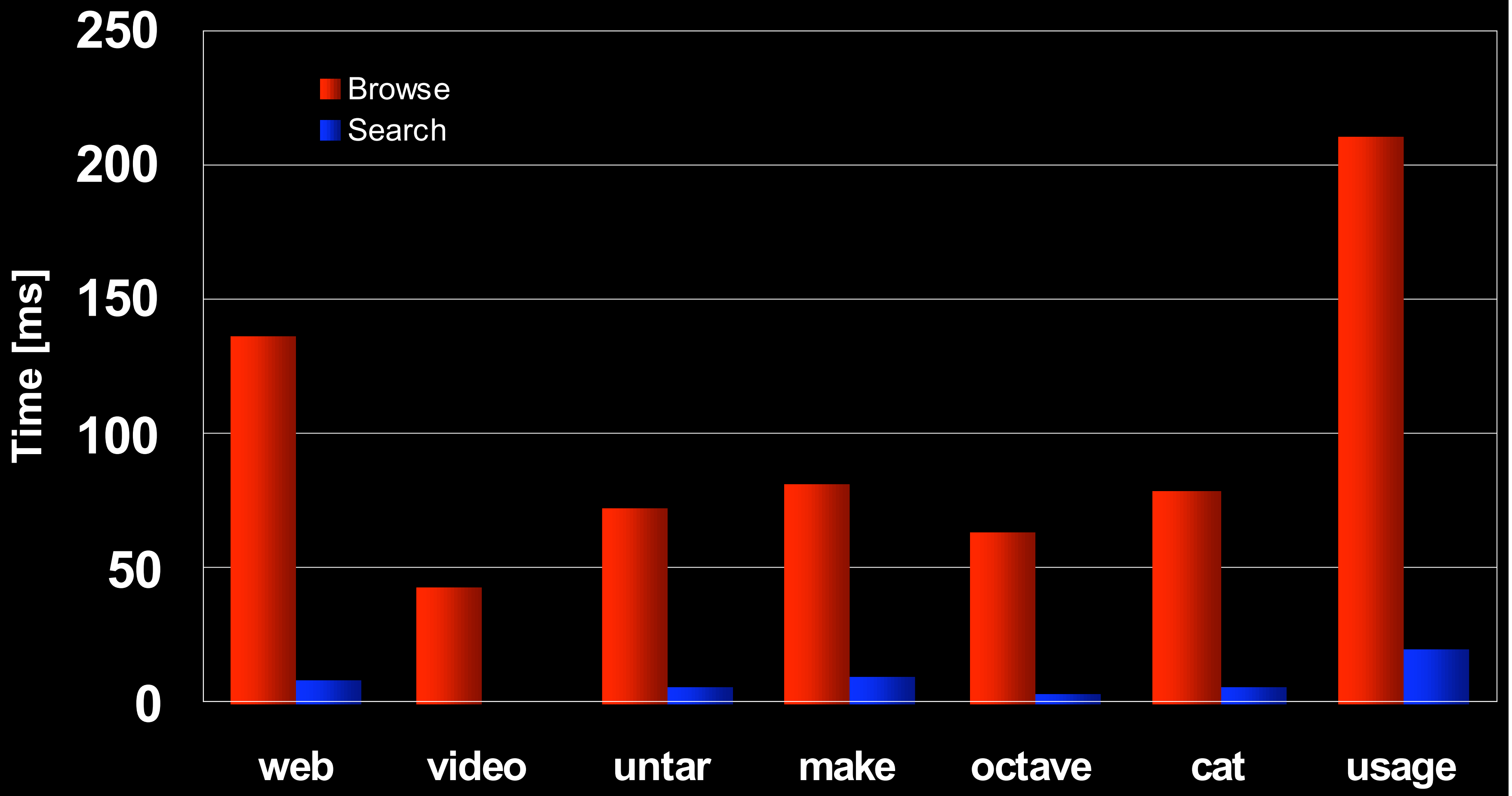


- Storage requirements are lower than PVR with equivalent display resolution

# Recording Storage Growth

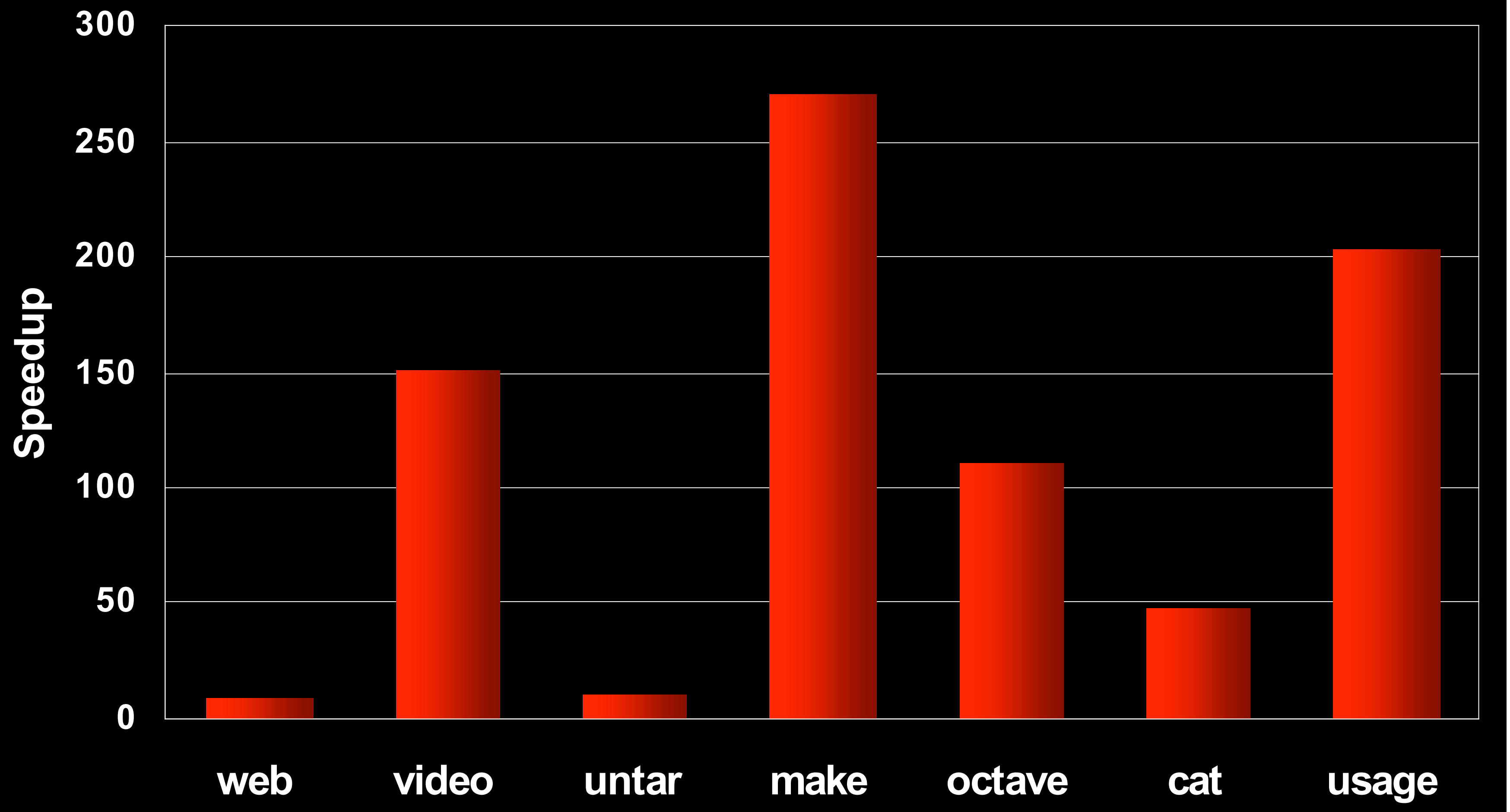


# Browse and Search Latency



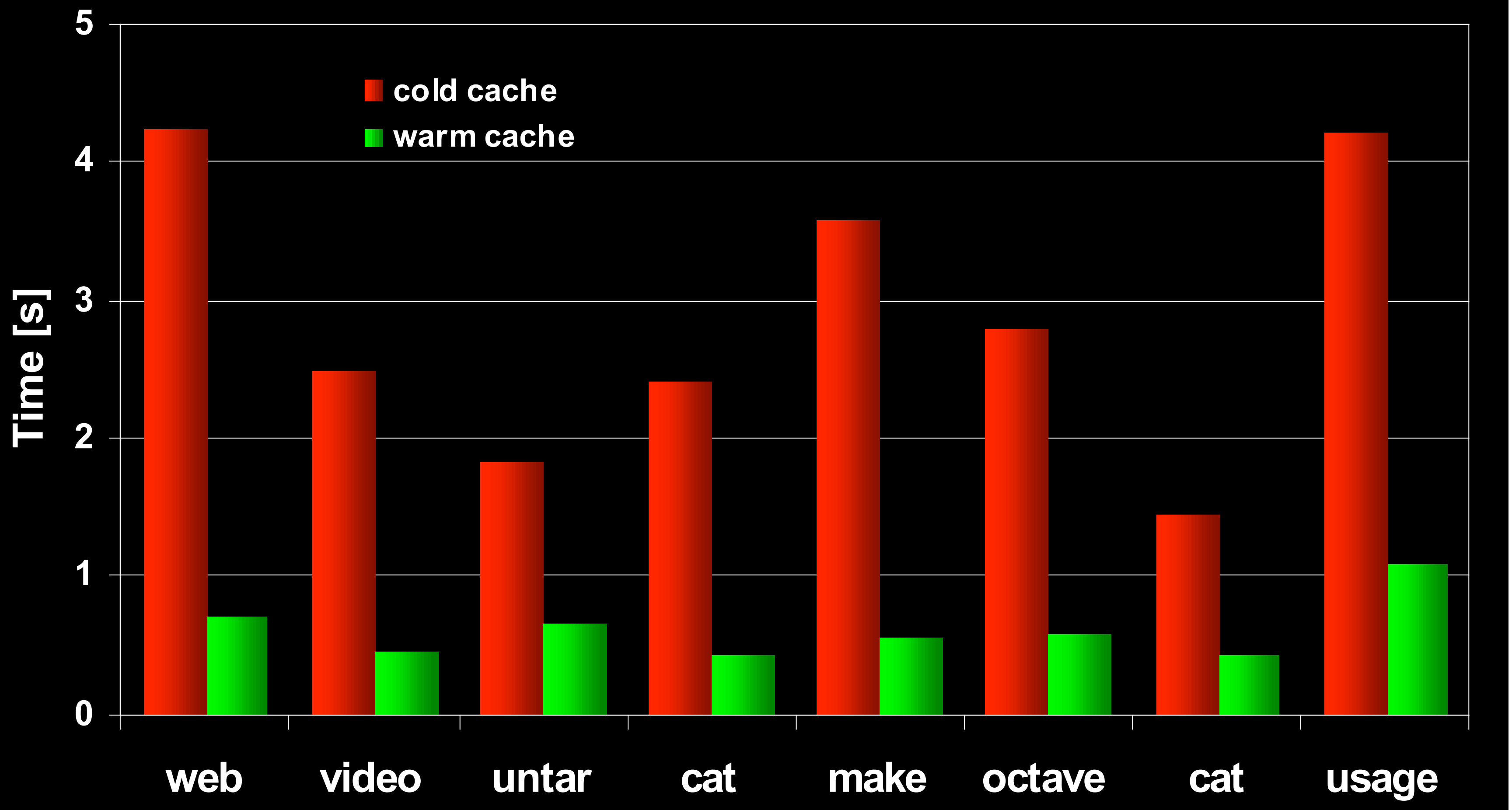
- Searching and browsing are fast enough to support interactive use

# Playback Speedup



- Faster than real-time visual search through the display recording

# Session Revive



- Latency to revive a session (from cold cache) is within a few seconds

# Conclusions

- DejaView: a new personal virtual computer recorder model
  - novel use of virtual display, virtual execution environment and accessibility
  - users can find, access and manipulate data they have previously seen
  - allows recording, playback, browsing, searching, and reviving live desktop
  - modest performance overhead, fast enough for interactive use

# Future Work

- A new paradiagm for desktop search
  - how to determine relevance?
  - relationship to desktop file search?
  - user interface issues
- Collaborative DejaView

# More Info

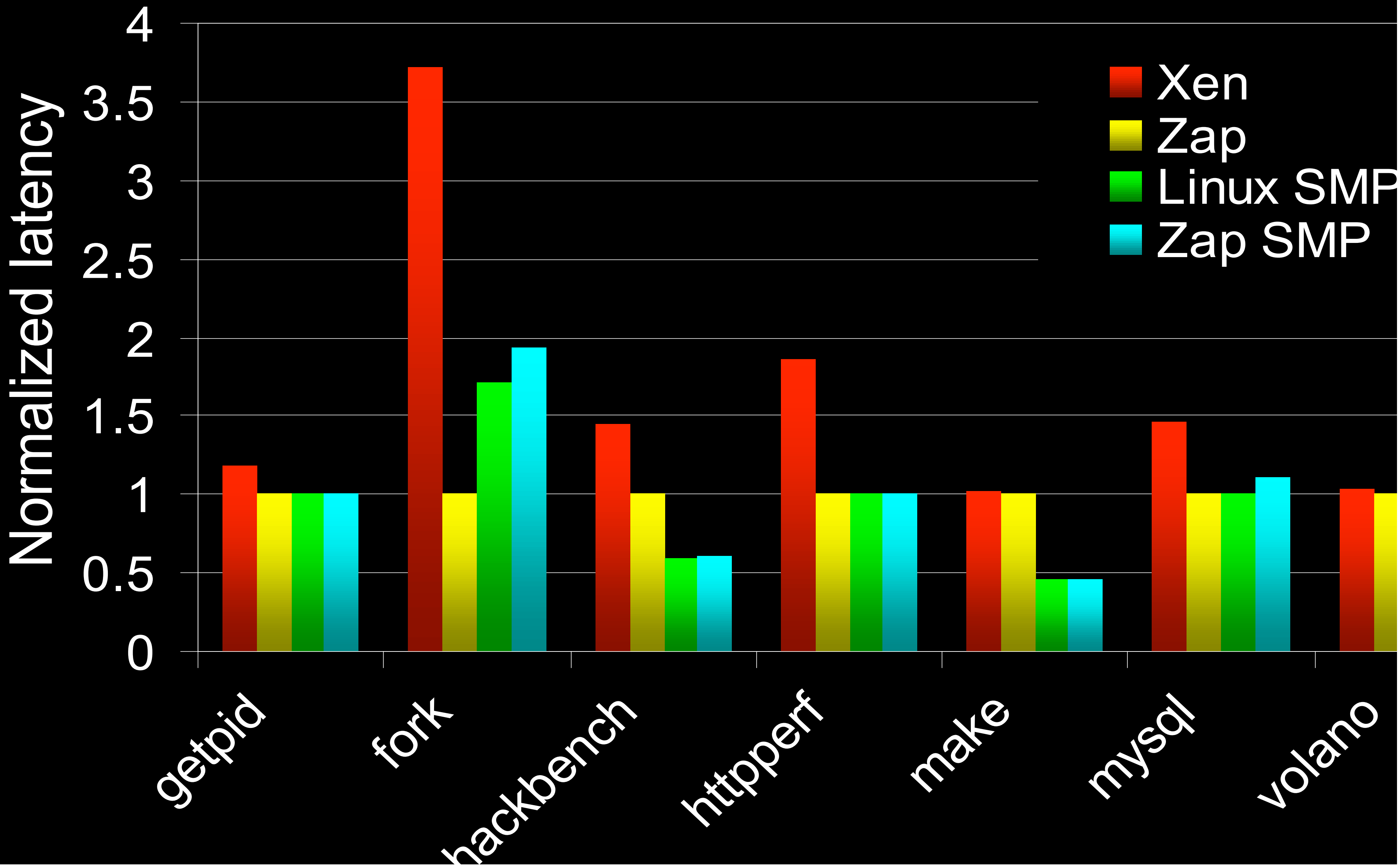
Network Computing Laboratory

http://ncl.cs.columbia.edu

# Reviving the Network

- What is the network state after revive ?
  - like resuming a hibernated laptop
  - stateful protocols: drop all connections
  - stateless protocol: don't care
- What about network access ?
  - disabled by default
  - enable per application, or globally

# Virtualization


Normalized latency
getpid
fork
hackbench
httperf
make
mysql
volano
🟥 Xen
🟨 Zap
🟩 Linux SMP
🟦 Zap SMP