Inter-program Compilation for Disk Energy Reduction

Jerry Hom and Ulrich (Uli) Kremer

Department of Computer Science

Rutgers University

EEL

Energy Efficiency and Low-Power Lab

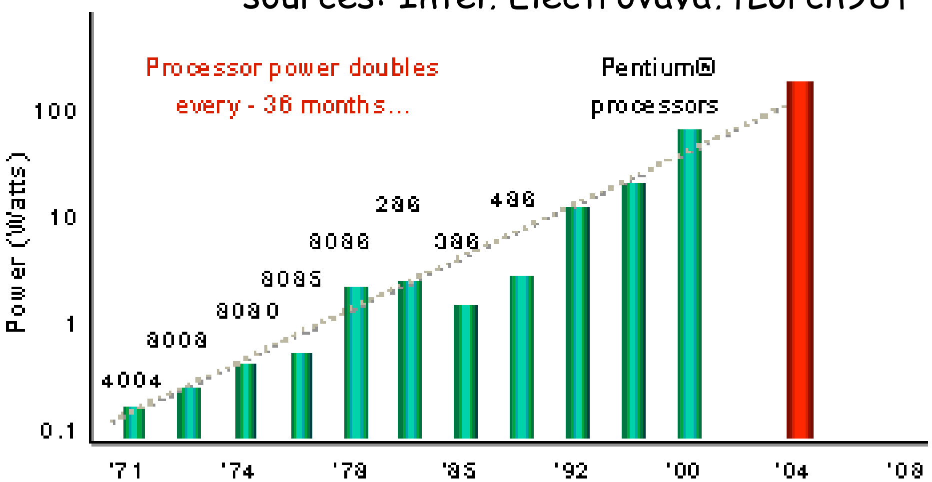
This research is partially supported by NSF

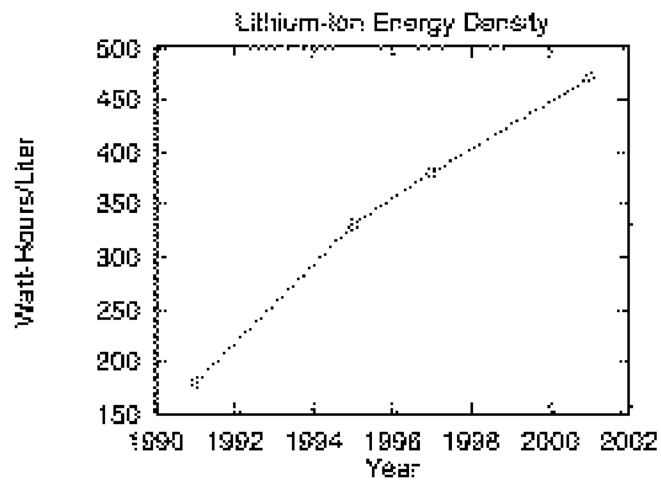
Introduction

Portable Computer System Trends

- Performance
- Power Requirements (CPU, Display, Disk)
- Power Supply

sources: Intel, Electrovaya, [Lorch98]





Introduction

Compiler Optimizations

- Performance
  - Time
  - Space (Resources)

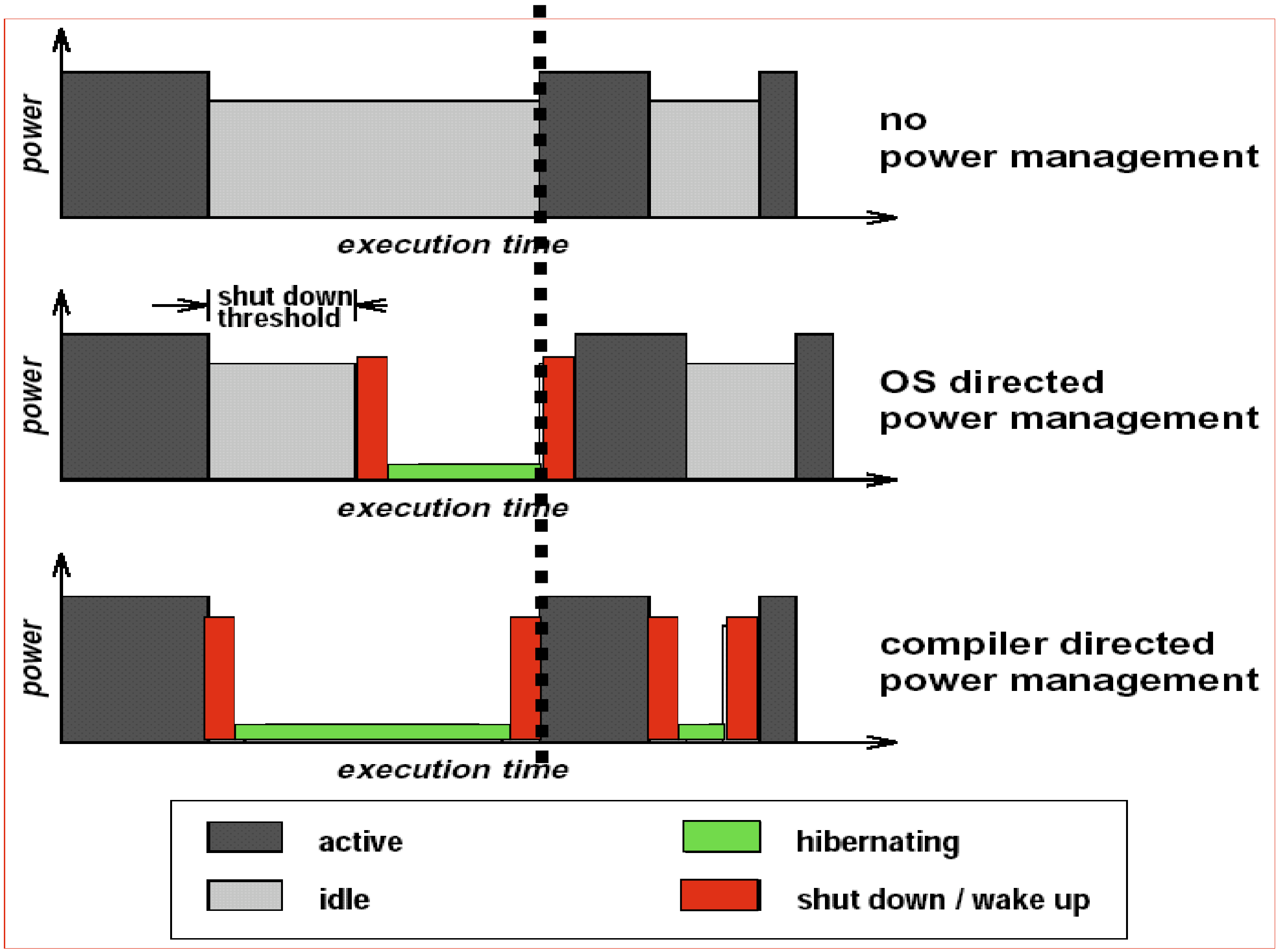
What can compilers do for energy/power?

Introduction

Compiler Optimizations

- Energy/Power [$P \sim fV^2$; $E = P(t)$]
  - Trade-offs with performance
  - Dynamic Frequency/Voltage Scaling
  - Remote Task Execution
  - Resource Management
    - Detect idle resources
    - Increase idleness
    - Direct resources to low power states

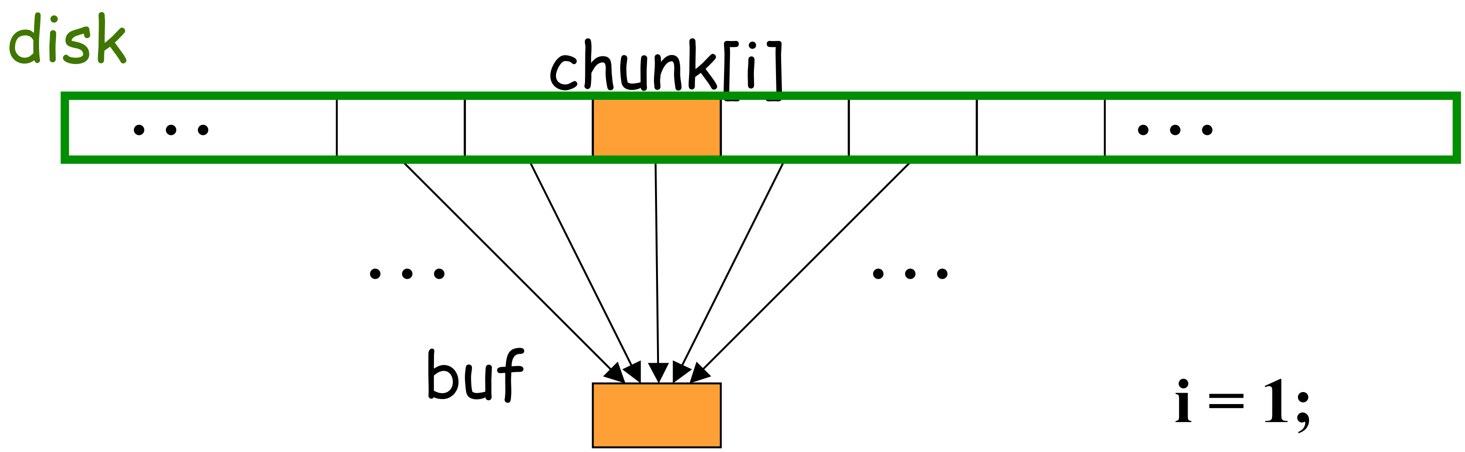
Threshold based OS vs. Compiler Directed Hibernation



Example: Clustering Disk Accesses

Original program

Heath et al., [PACT'02]

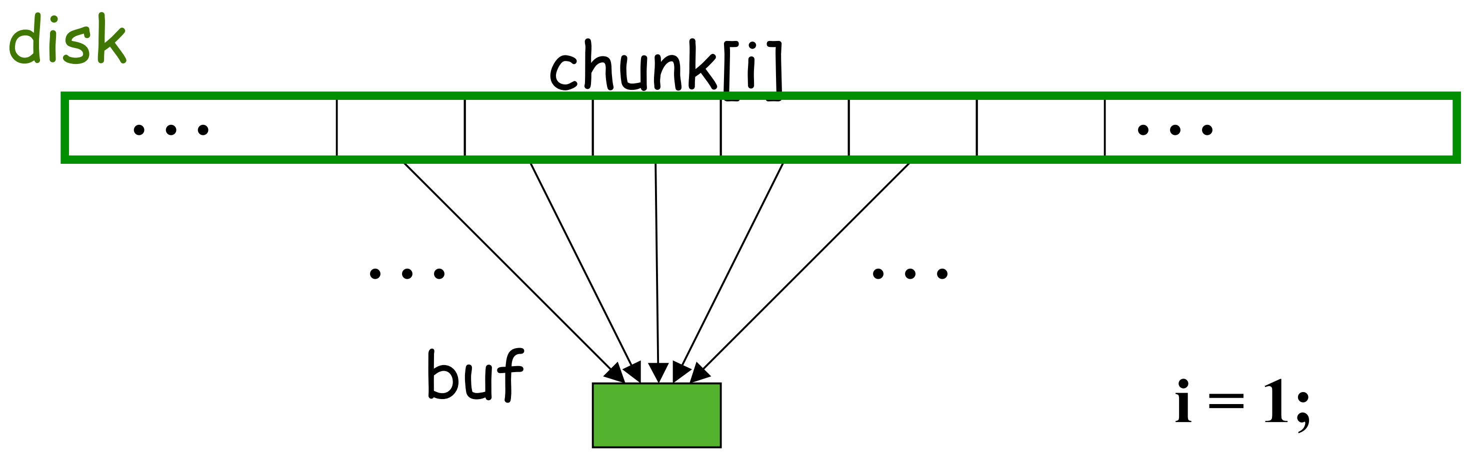



```
i = 1;
while  i <= N {
   read chunk[i] into buf
   compute on buf;
   i := i + 1;
}
```

Example: Clustering Disk Accesses

Original program

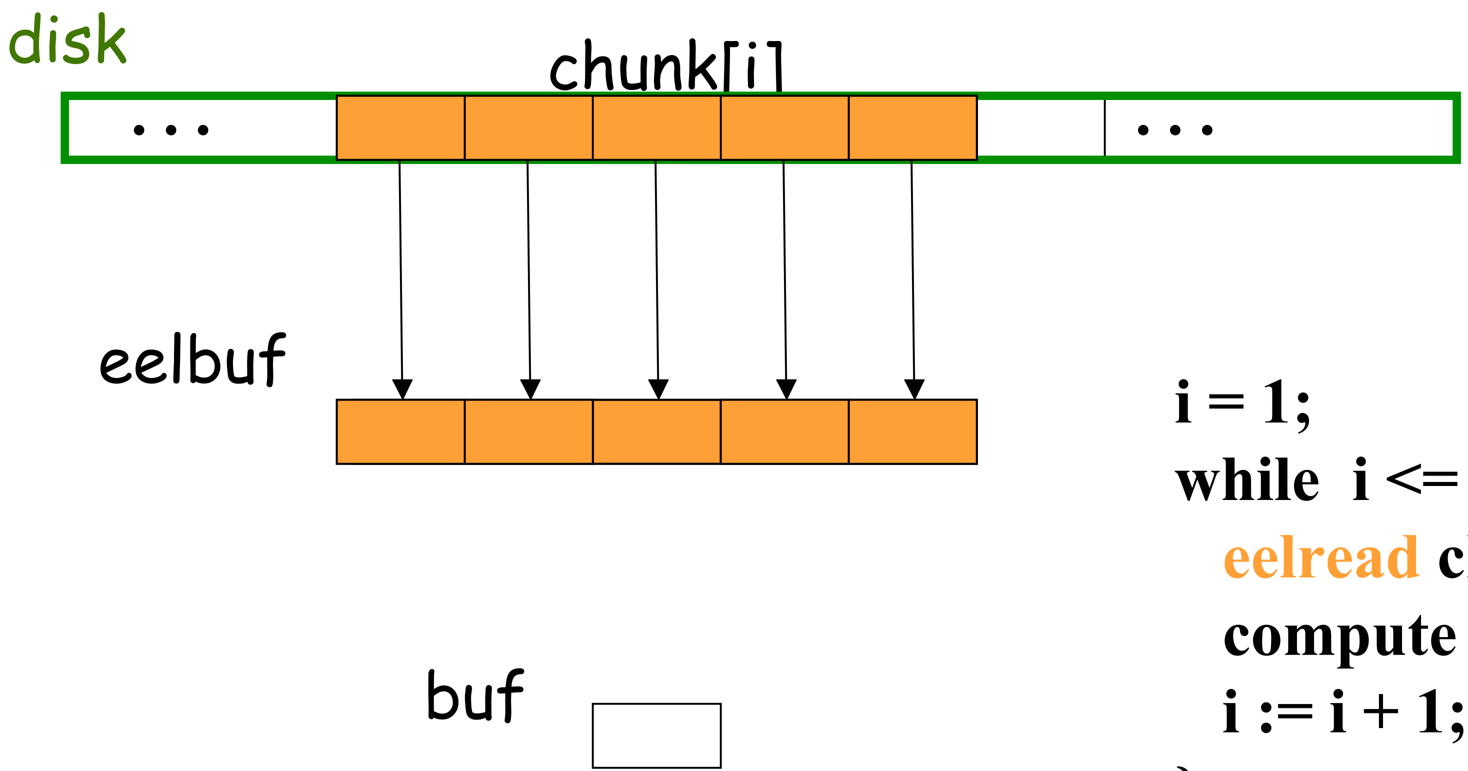
Heath et al., [PACT'02]




```
i = 1;
while  i <= N {
   read chunk[i] into buf
   compute on buf;
   i := i + 1;
}
```

Example: Clustering Disk Accesses

Optimized program — Heath et al., [PACT'02]

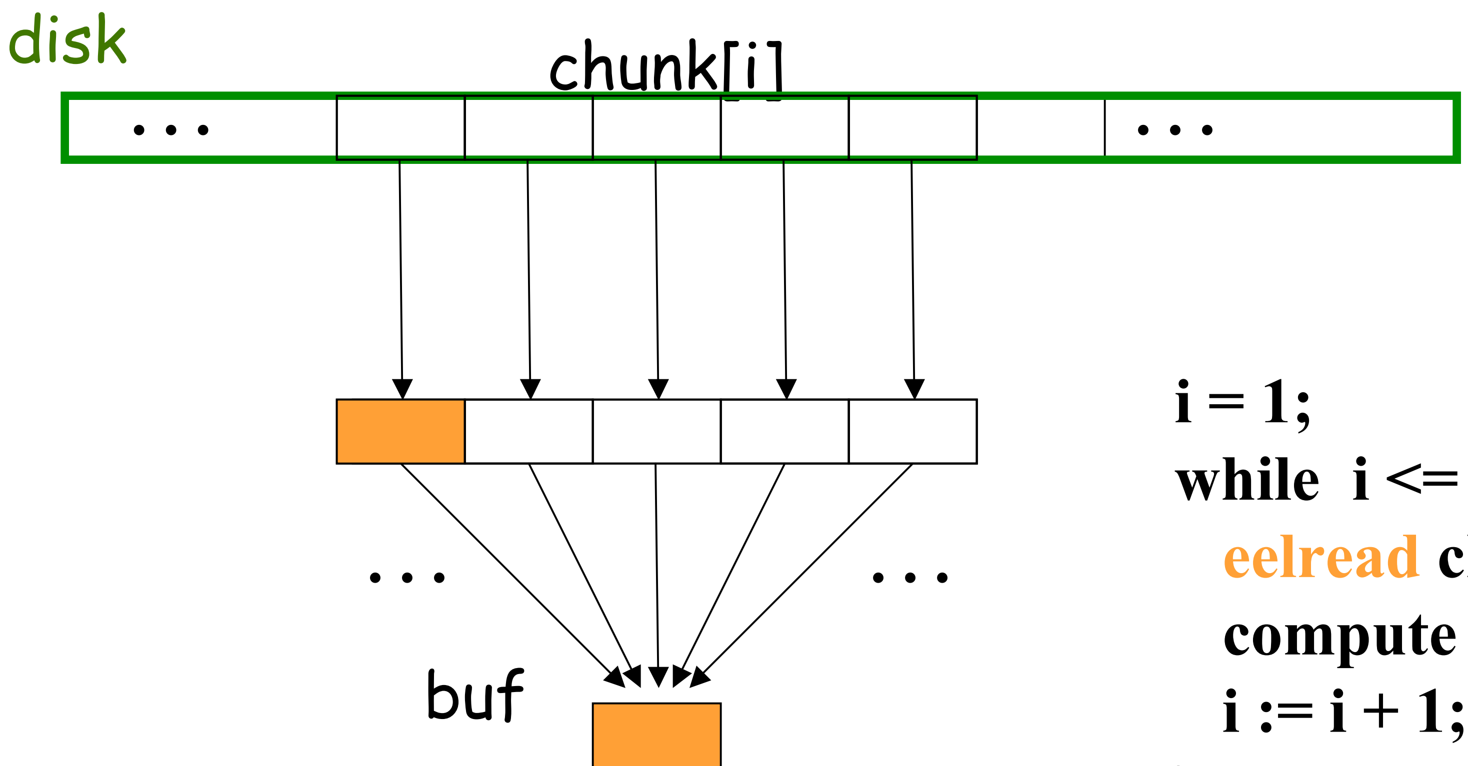



```
i = 1;
while  i <= N {
   eelread chunk[i] into buf
   compute on buf;
   i := i + 1;
}
```

Example: Clustering Disk Accesses

Optimized program

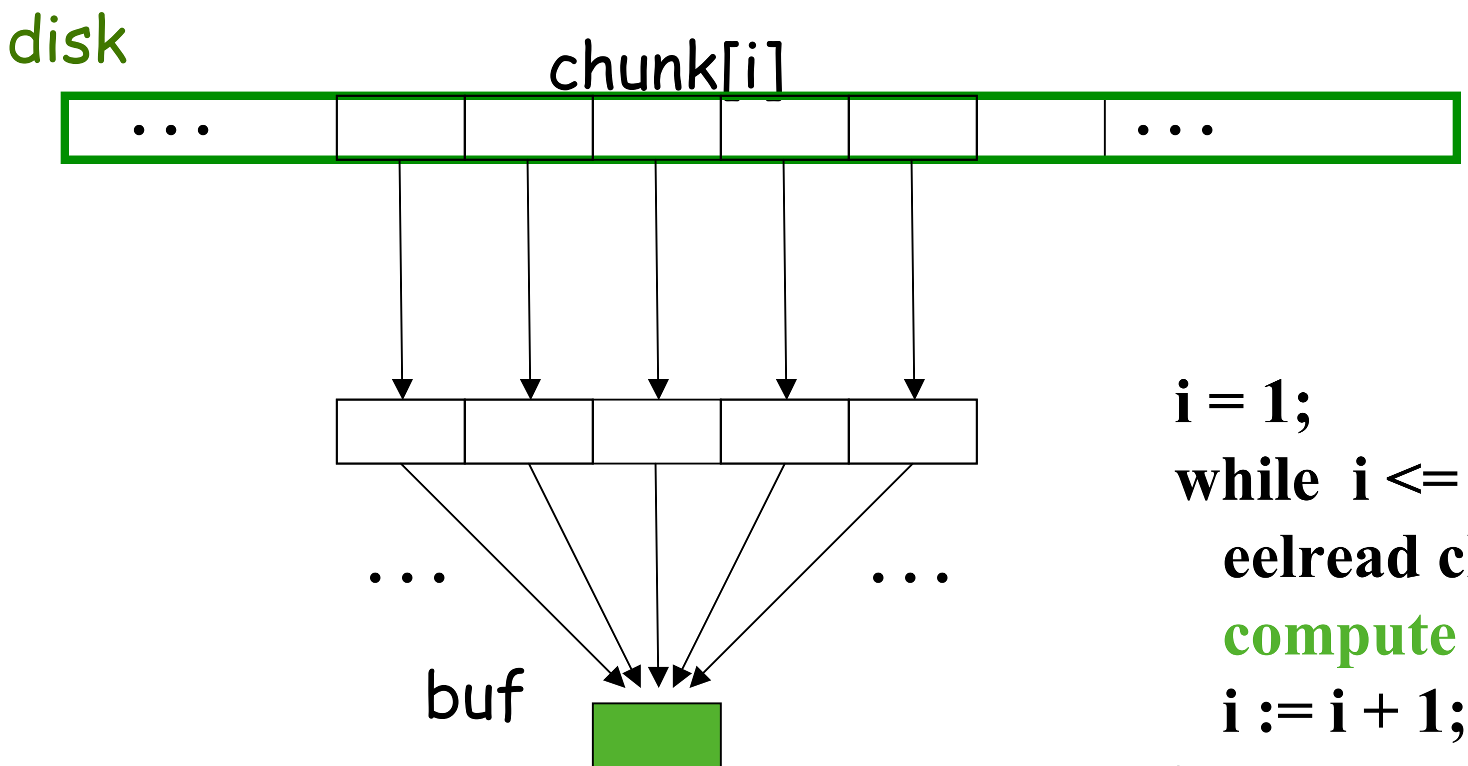
Heath et al., [PACT'02]



i = 1;
while i <= N {
eelread chunk[i] into buf
compute on buf;
i := i + 1;
}

Example: Clustering Disk Accesses

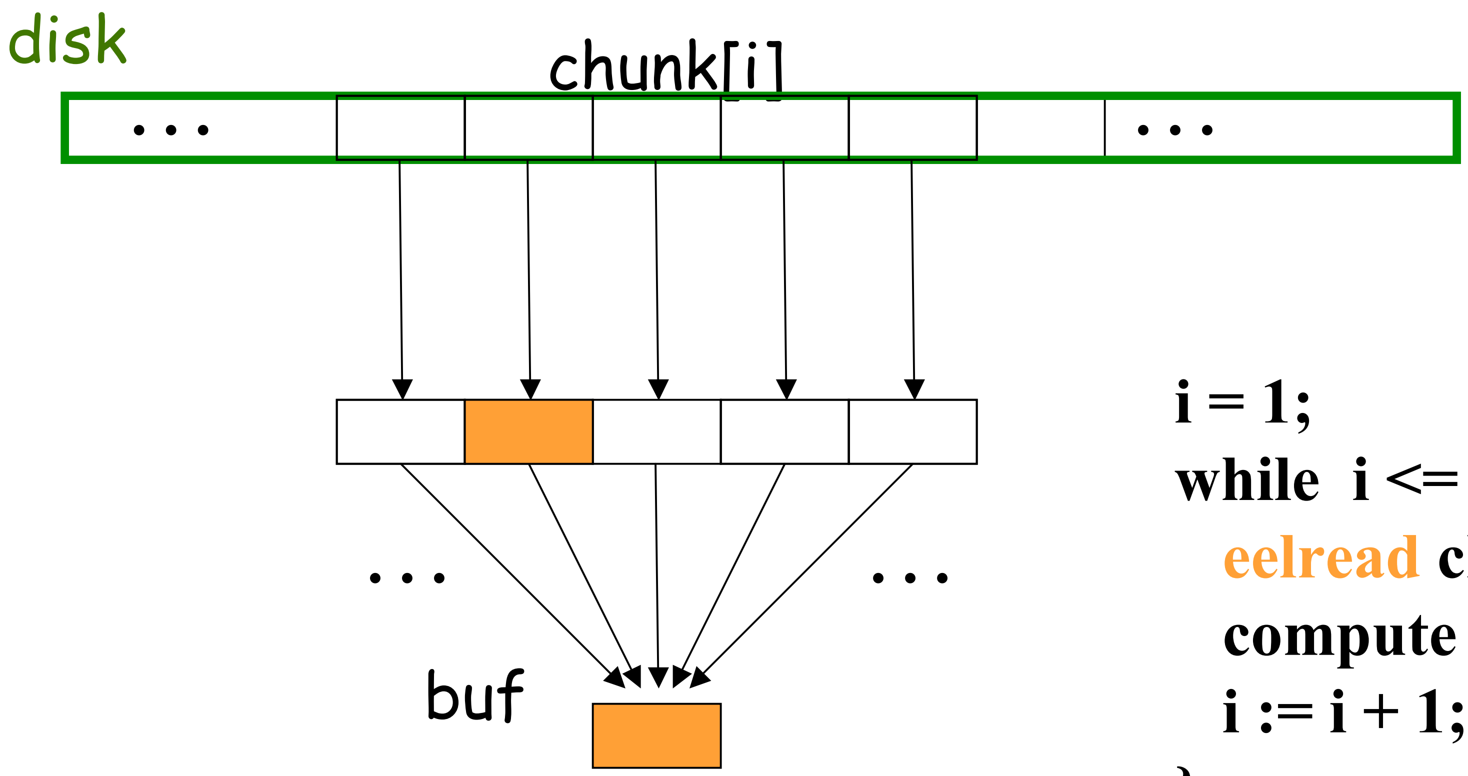
Optimized program   Heath et al., [PACT'02]




```
i = 1;
while  i <= N {
   eelread chunk[i] into buf
   compute on buf;
   i := i + 1;
}
```

Example: Clustering Disk Accesses

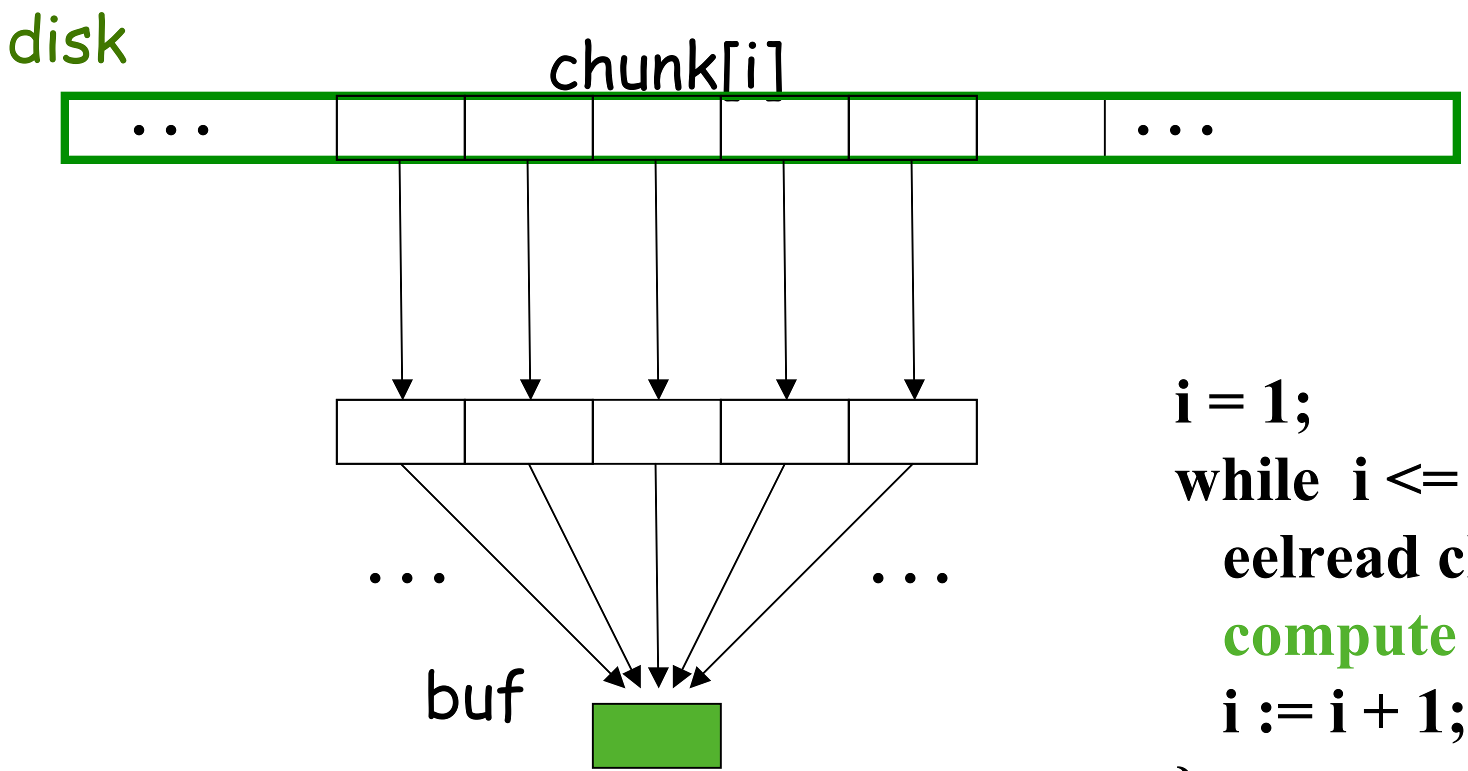
Optimized program Heath et al., [PACT'02]




```
i = 1;
while  i <= N {
   eelread chunk[i] into buf
   compute on buf;
   i := i + 1;
}
```

Example: Clustering Disk Accesses

Optimized program   Heath et al., [PACT'02]




```
i = 1;
while  i <= N {
   eelread chunk[i] into buf
   compute on buf;
   i := i + 1;
}
```

Contributions

Idea: Cluster resource accesses (disk accesses) across multiple programs

Contributions:

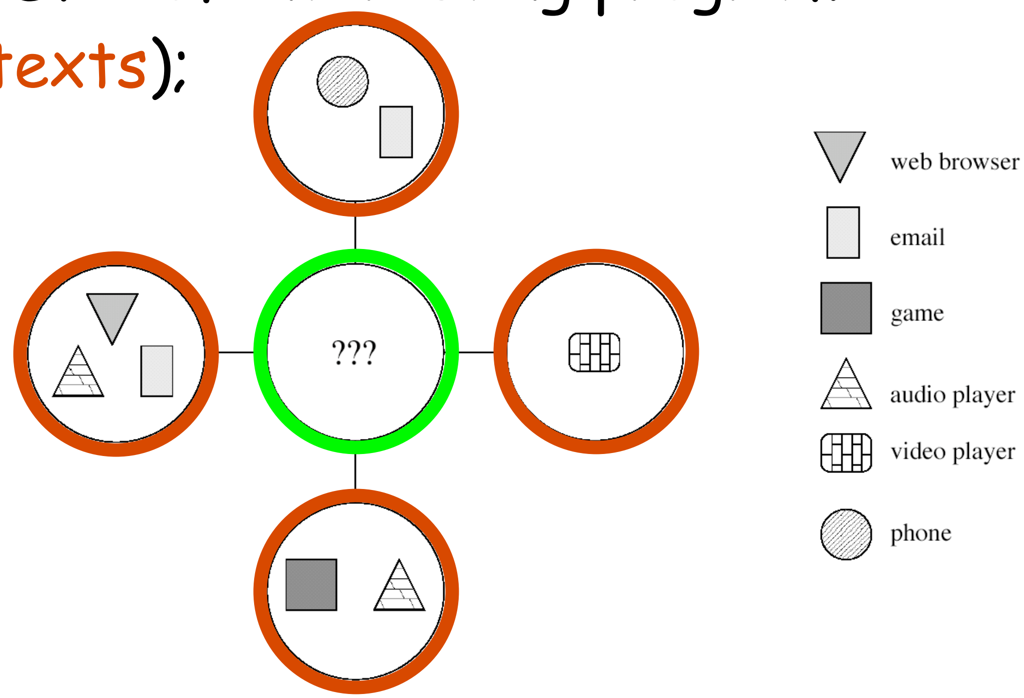
- How to synchronize resource accesses
- How to keep execution context information
- Initial benefit analysis

Talk Outline

- Assumptions
- Barrier and inverse-barrier synchronization
- Experimental results based on hand-simulations (mpeg video, mpeg audio, sftp)
- Proposed compilation framework
- Recent results
- Related work
- Summary and Future work

Assumptions

1. Primary target environment: handheld PCs, with small groups of programs at any point in time
2. All programs in a group fit into main memory
3. Groups may be determined through profiling and benefit analyses _ DFA of interesting program subsets (execution contexts); transitions due to program execution and termination events



Resource-Centric Synchronization

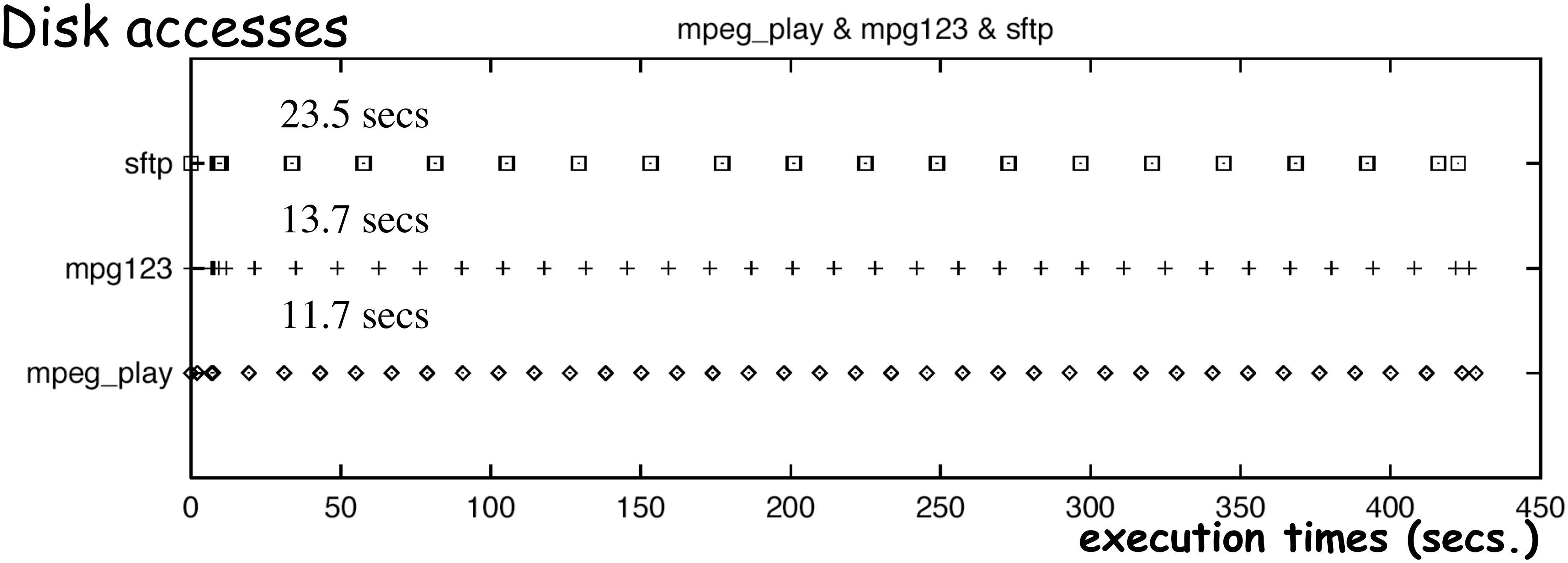
Barrier synchronization: Delay resource access until all members in the group are ready to use it

Inverse Barrier: Initiate resource access in all other members once a single member has accessed the resource

Note: Barrier synchronization may lead to problems if program group contains "real-time" application(s); Example: audio player and editor

Resource-Centric Synchronization

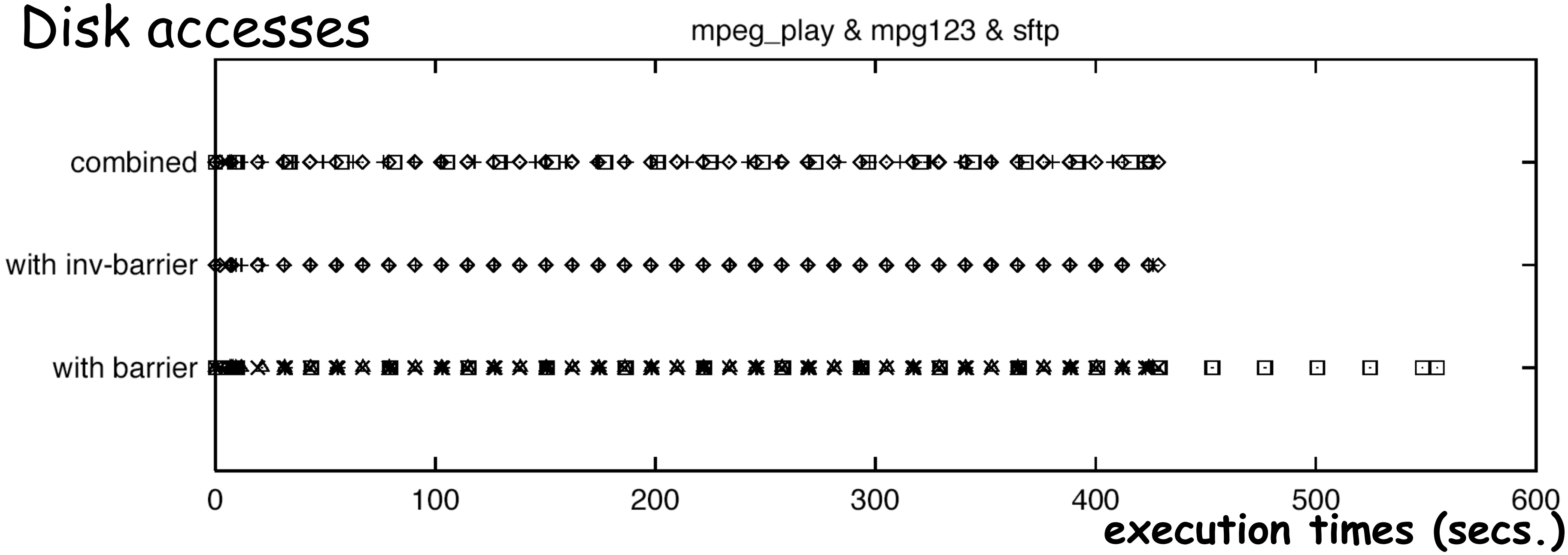
Disk accesses



- each program optimized for maximal buffer size
- CPU – enough capacity
- OS - immediate de-activation and pre-activation
- Fujitsu disk: 10 secs. idle time to break even in "standby"

Resource-Centric Synchronization

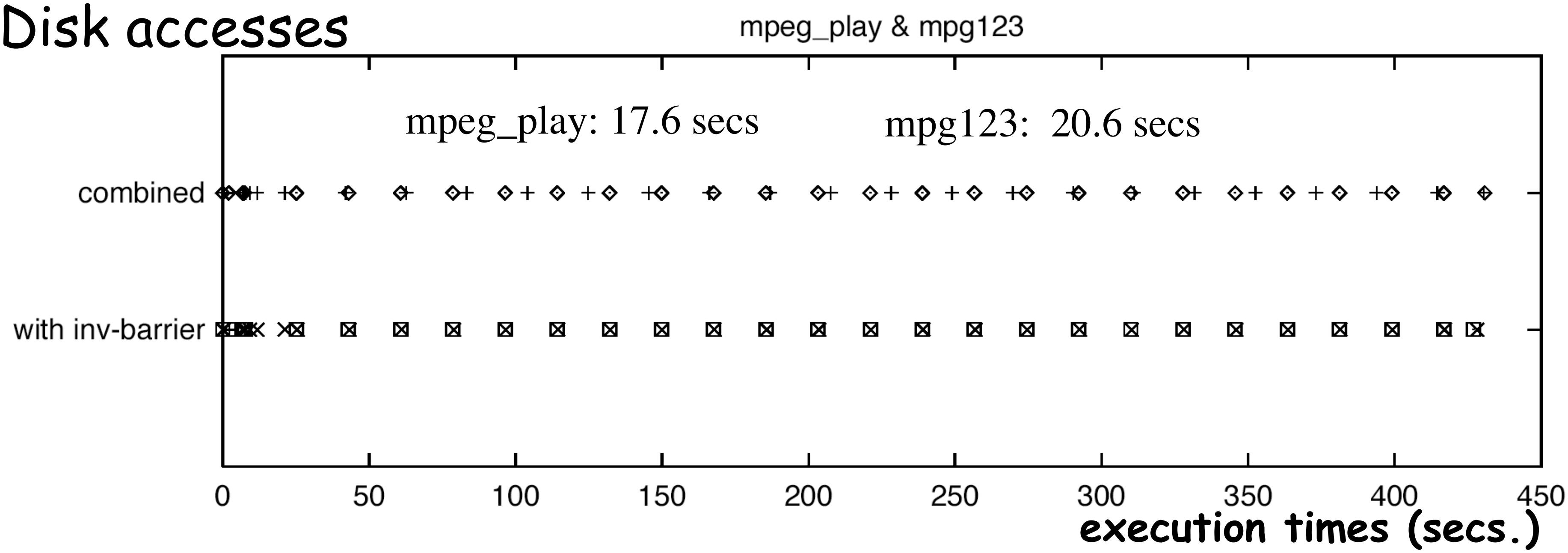
Disk accesses



- All Inverse Barrier: saves 5% energy; no performance penalty
- Both mpeg applications Inverse Barrier, ftp Barrier:
  2.4% more energy --- 31.4% performance penalty (sftp)

Resource-Centric Synchronization

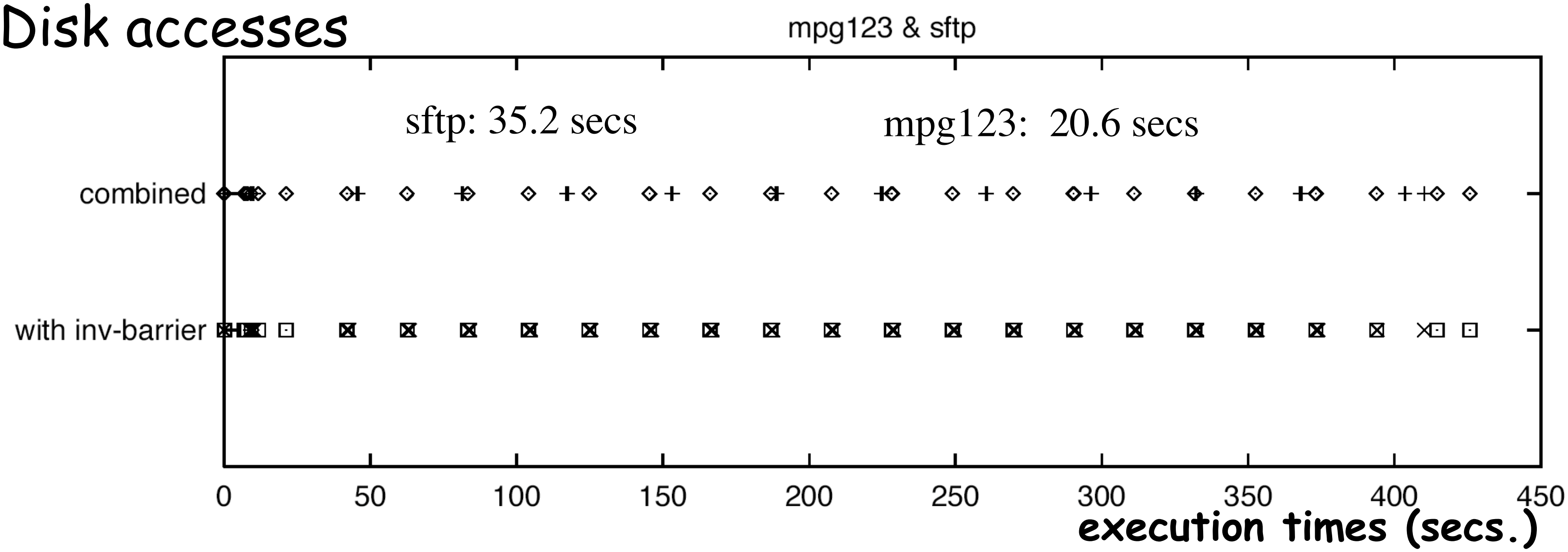
Disk accesses



Inv-Barrier: saves 15.9% energy, no performance penalty

Resource-Centric Synchronization

Disk accesses



Inv-Barrier: saves 9.8% energy, no performance penalty

Proposed Compilation Framework

Synchronization and communication between programs through signals and signal handlers; handlers decide when to refill buffers

Inverse Barriers:

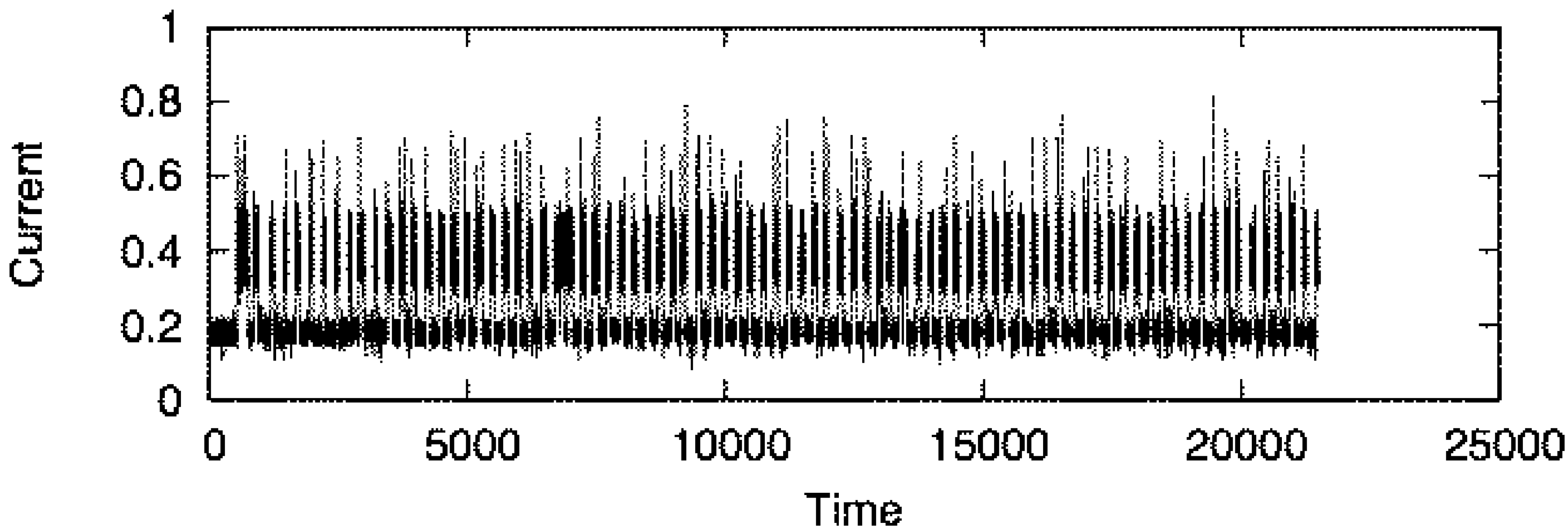
- program accesses disk, then signal other applications
- program receives disk access signal; handler implements policy whether to refill buffer or not

State Transitions:

- program begins execution or terminates: inform other programs to initiate transition events
- program receives its initial state from other programs

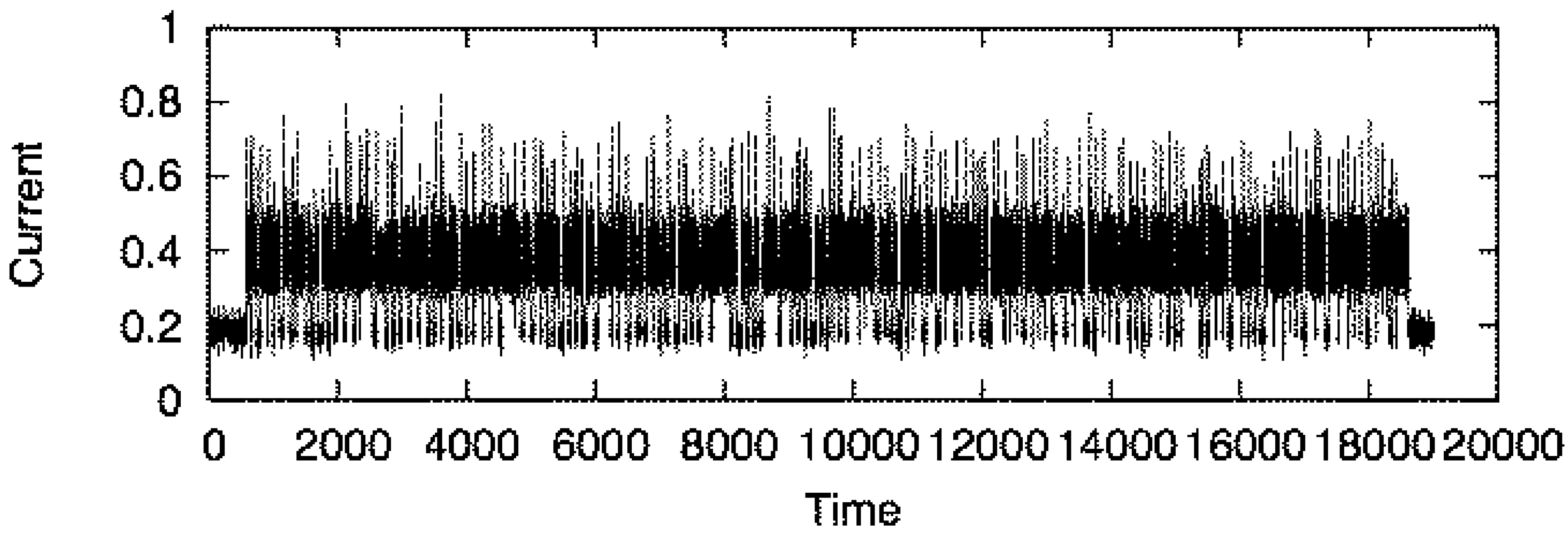
Recent Results

Audio UM



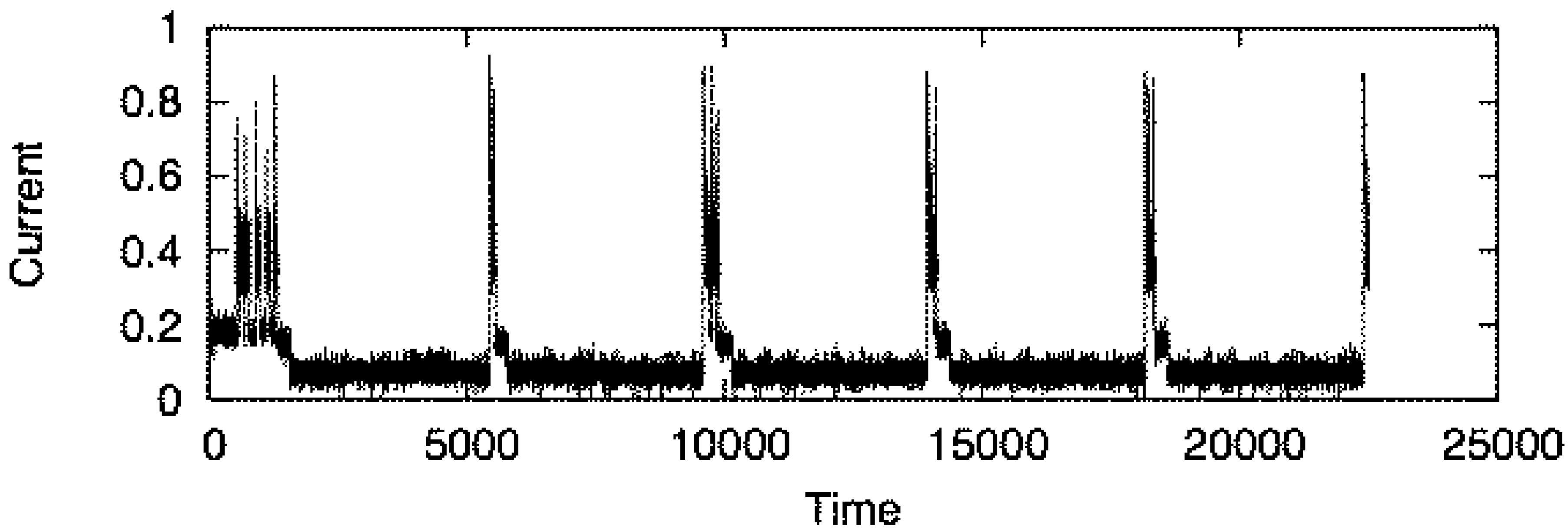
Recent Results

Video UM



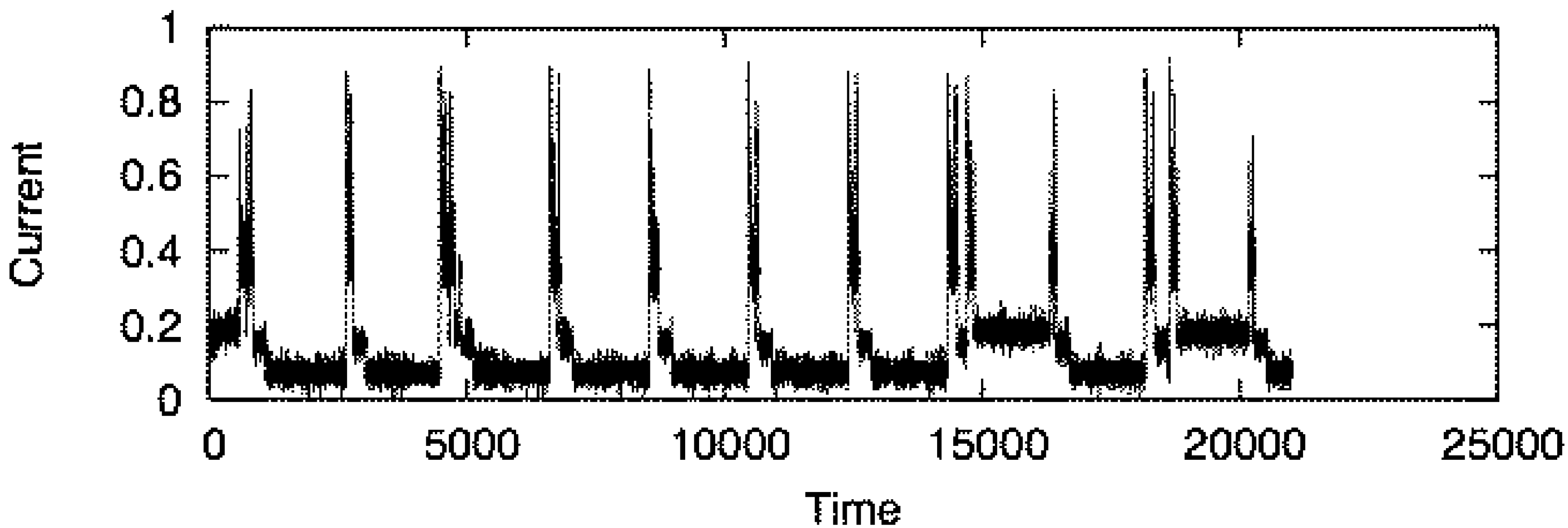
Recent Results

Audio CO



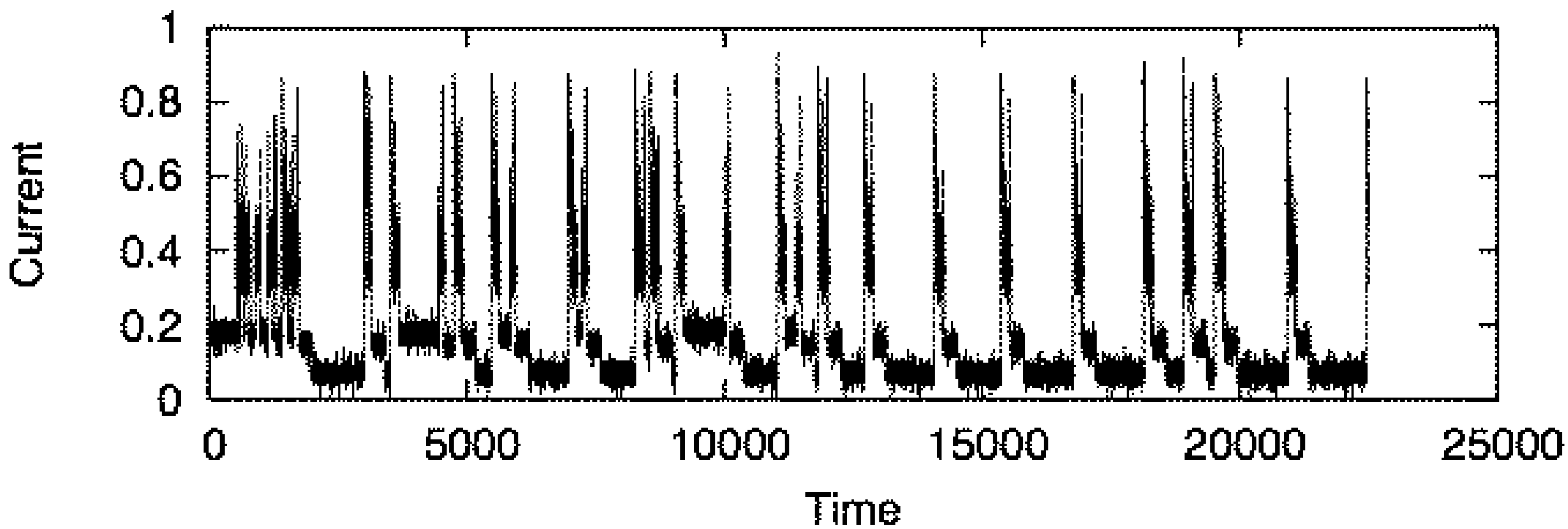
Recent Results

Video CO



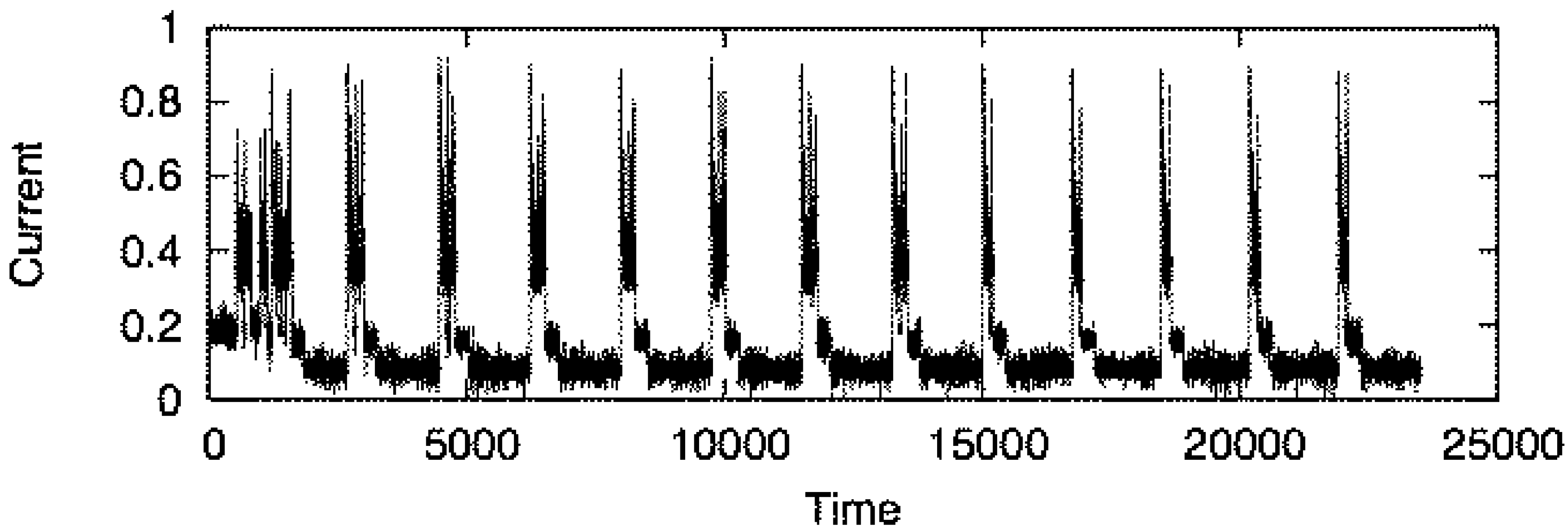
Recent Results

AV CO



Recent Results

AV INV



Related Work

- Application Transformations [PACT02]

T. Heath, E. Pinheiro, J. Hom, U. Kremer, R. Bianchini

- Collective Compilation [IASTED01]

I. Kadayif, M. Kandemir, U. Sezer

- Cooperative I/O [OSDI02]

A. Weissel, B. Beutel, F. Bellosa

- Implicit Co-Scheduling [SIGMETRICS98]

A. Arpaci-Dusseau, D. Culler, A. Mainwaring

- Barrier

Summary and Conclusions

- promising new technique
- inverse barrier – an interesting synchronization paradigm
- simulation shows disk energy savings between 5% and 16% on specific program groups
- analytical model shows upper bound of energy savings as 58% (two applications)
- qualitative validation of inverse-barrier via signaling on physical disk traces

Future Work

- Generate code of compilation framework by hand and perform benefit analysis based on physical measurements

- Implement an OS oriented approach
  - OS maintains buffer for each active process and implements refill policies
  - OS keeps track of states
  - OS takes hints from compiler

- Investigate techniques to identify interesting program groups

Thank You

Energy Efficiency and Low-Power Lab

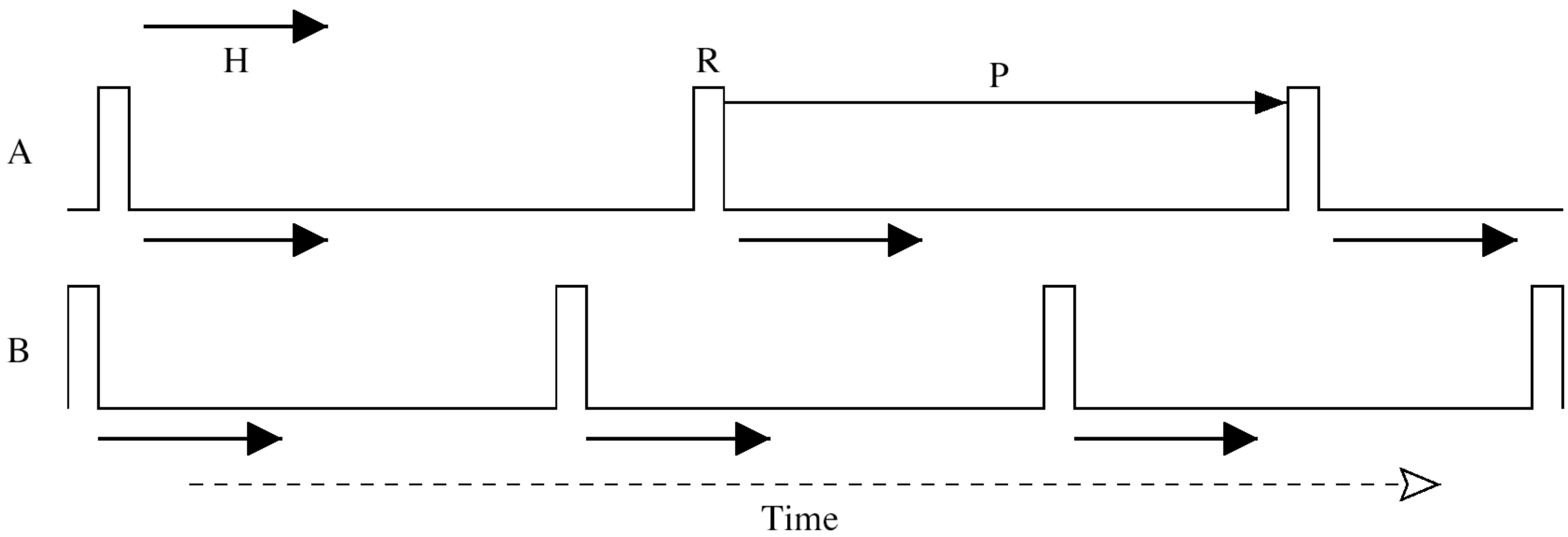
http://www.cs.rutgers.edu/~uli/eel

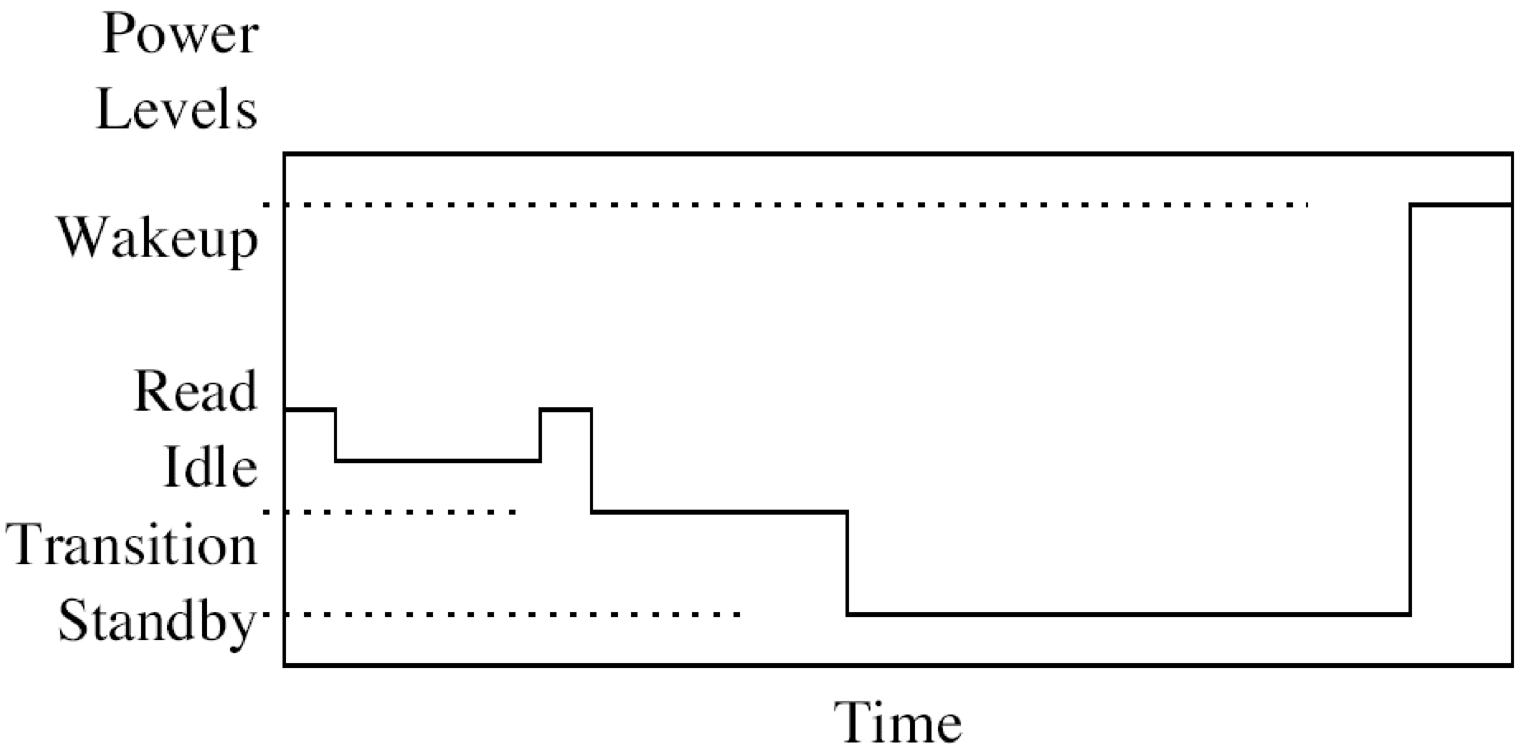
Simplified Fujitsu Disk Parameters

Disk States	Power (W)	Time (s)
Wakeup	3.0	1.6
Read	1.8	
Idle	0.9	
Transition	0.7	5.0
Standby	0.2	
Threshold for Standby: 10.0 secs		

Disk Access Intervals (s)

	1-app	2-apps	3-apps
mpeg_play	35.2	17.6	11.7
mpg123	41.2	20.6	13.7
sftp	70.4	35.2	23.5


H
R
P
A
B
Time


Power
Levels
Wakeup
Read
Idle
Transition
Standby
Time