Multiprocessor Kernel Performance Profiling

Alex Mirgorodskii

mirg@cs.wisc.edu

Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706-1685
USA

Kperfmon: Overview

- Specify a *resource*
  - Almost any function or basic block in the kernel
- Apply a *metric* to the resource:
  - Number of entries to a function or basic block
  - Wall clock time, CPU time (virtual time)
  - All Sparc Hardware Counters: cache misses, branch mispredictions, instructions per cycle, ...
- Visualize the metric data in real time

Kperfmon-MP: Goals

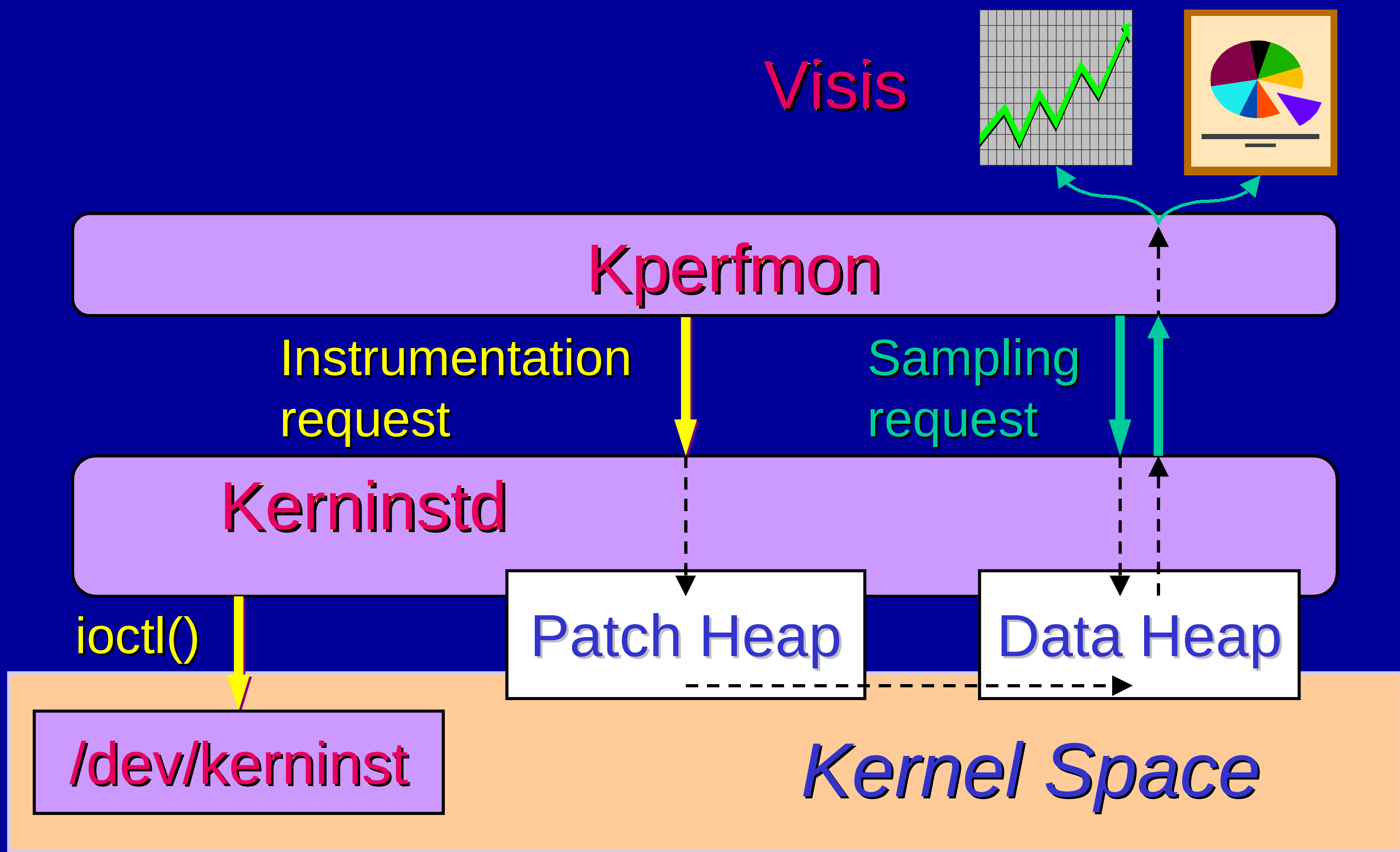
Modify uniprocessor Kperfmon to provide:

- Safe operation on SMP machines
  - Thread safety
  - Migration safety
- New feature: Per-CPU performance data
  - More detailed performance data
  - Reduce cache coherence traffic caused by the tool

Kperfmon: Technology

- Use the *KernInst* framework to:
  - Insert measurement code in the kernel at run time
  - Sample accumulated metric values from the user space periodically
- No need for kernel recompilation
  - Works with stock SPARC Solaris 7 kernels
  - Supports both 32-bit and 64-bit kernels
- No need for rebooting
  - Important for 24 x 7 systems

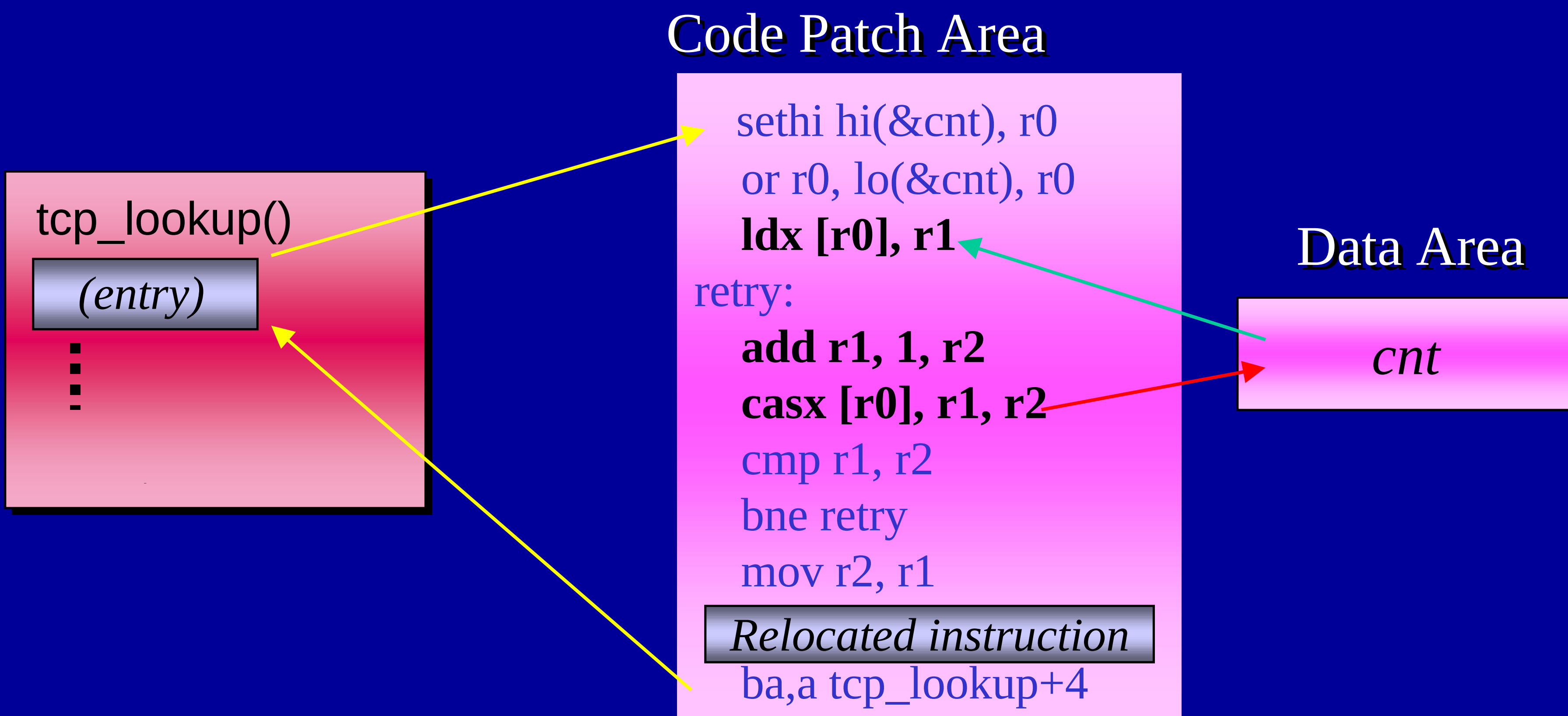
Kperfmon System



Kperfmon instrumentation

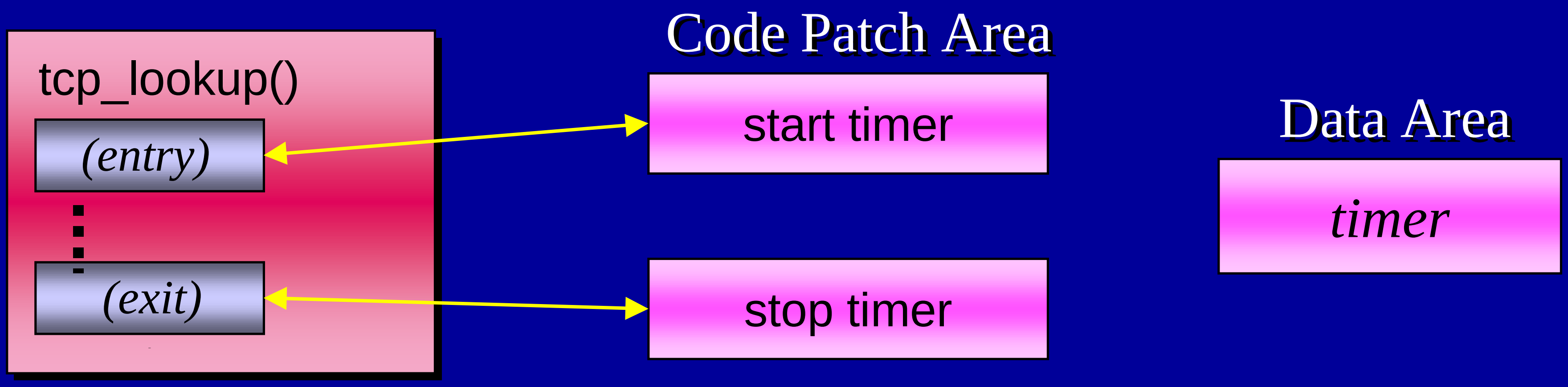
- Counter primitive
  - Number of entries to a function or a basic block
- Wall clock timer primitive
  - Real time spent in a function
- CPU timer primitive
  - Excludes time while the thread was switched-out
  - Can count more than just timer ticks
    - All HW-counter metrics use this mechanism

Non-MP Counter primitive



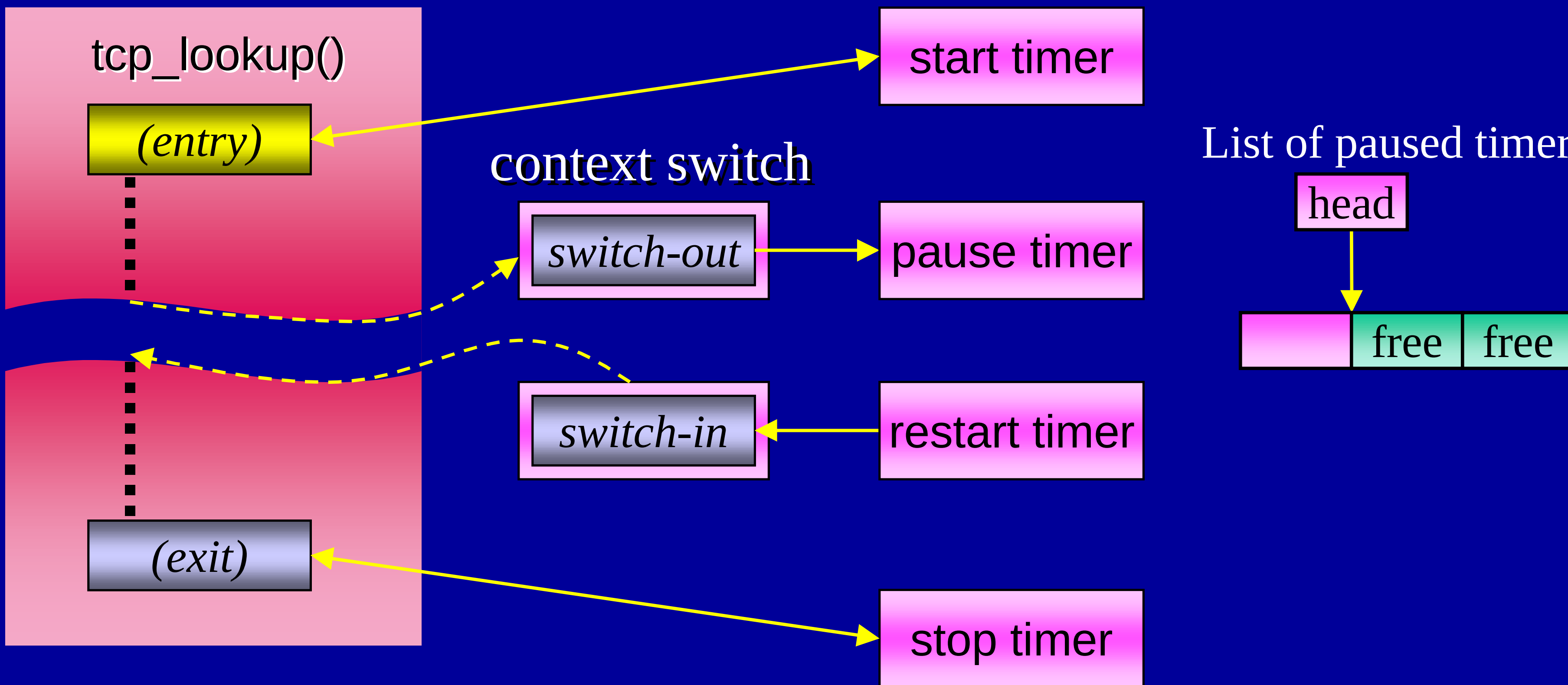
- Atomic, thread-safe update
- Lightweight
- No register save/restore required

Non-MP Wall clock timer primitive



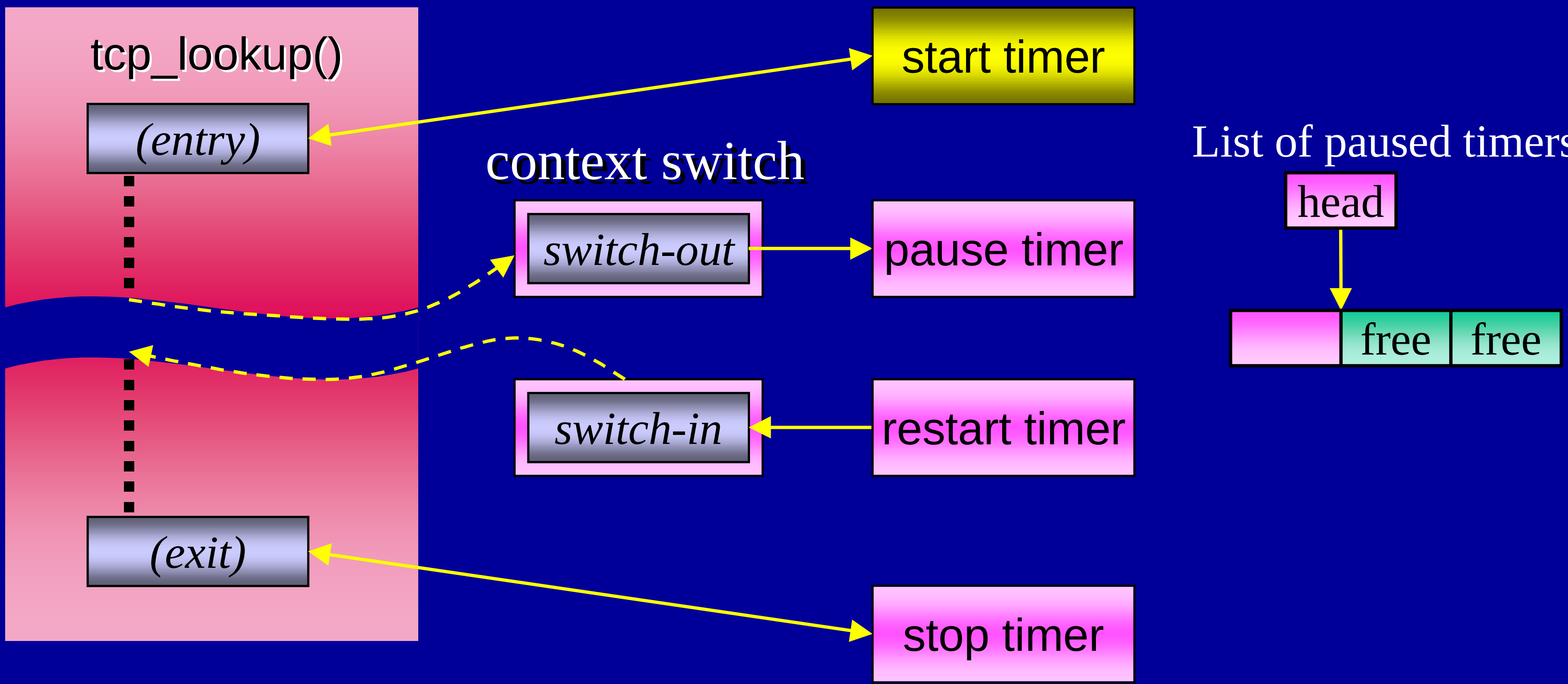
- Inclusive (includes time in callees)
- Keeps accumulating if switched-out

Non-MP CPU timer primitive



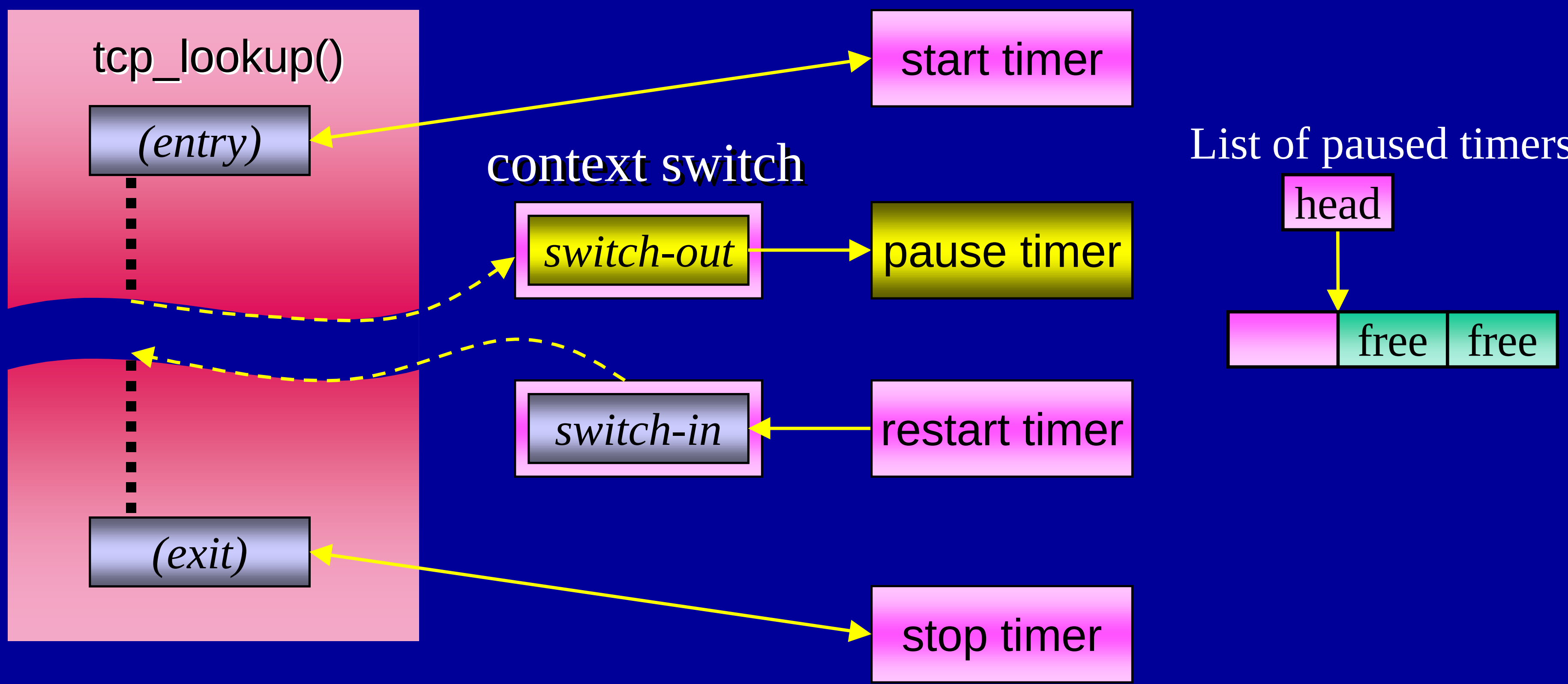
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



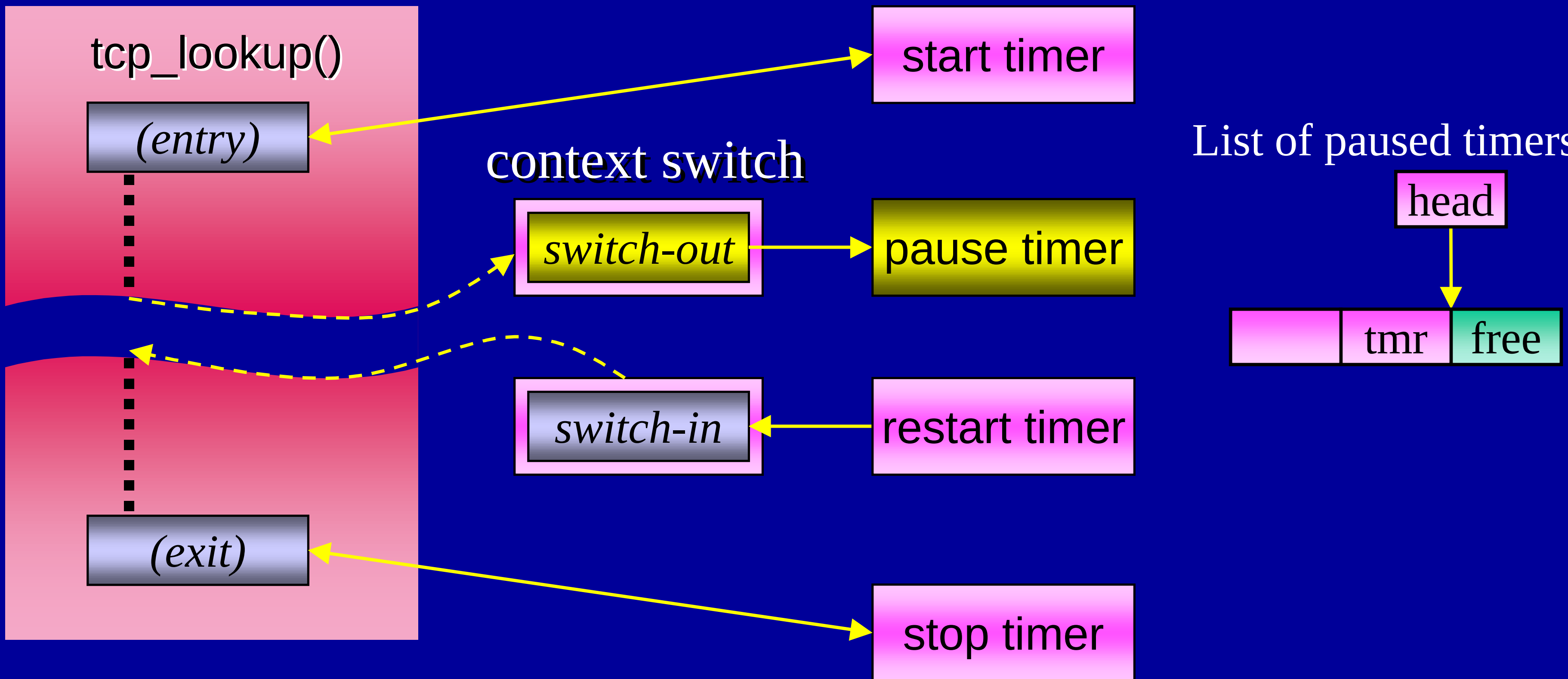
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



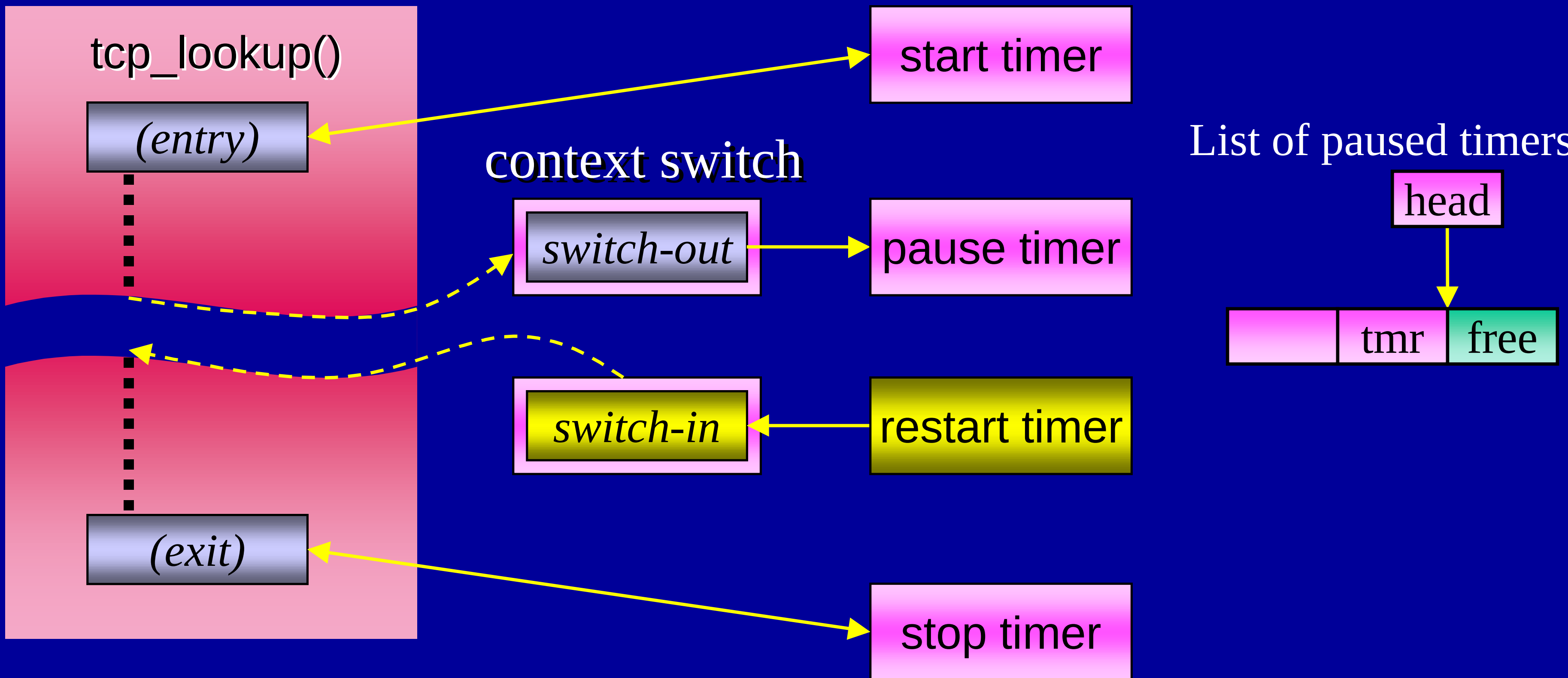
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



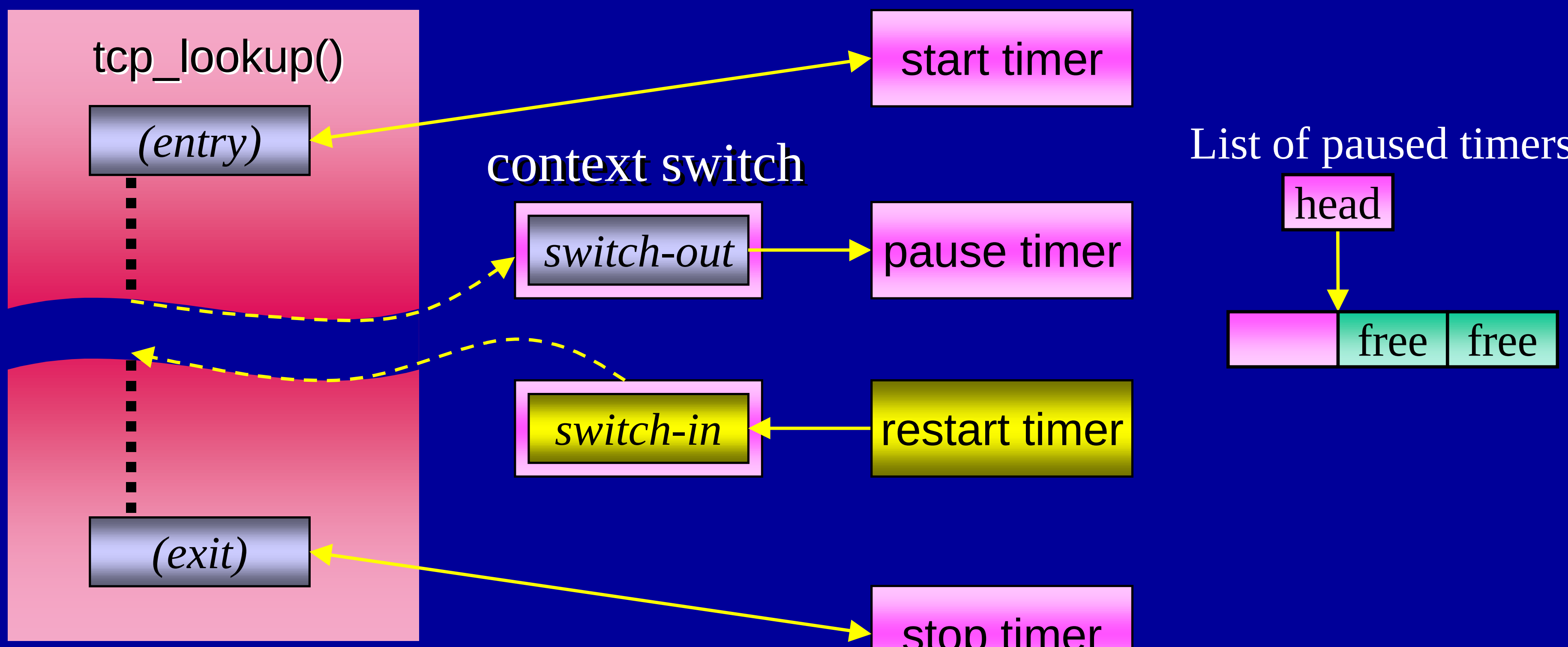
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



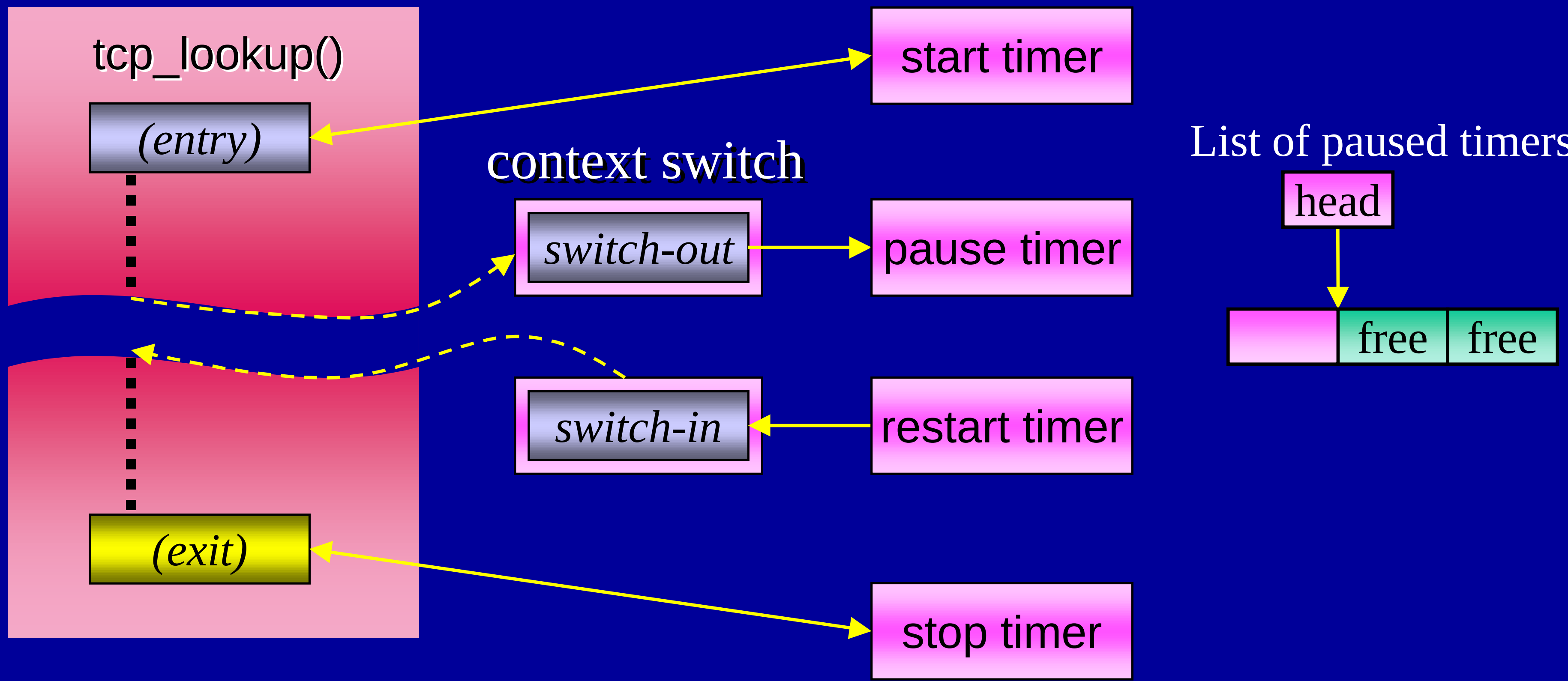
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



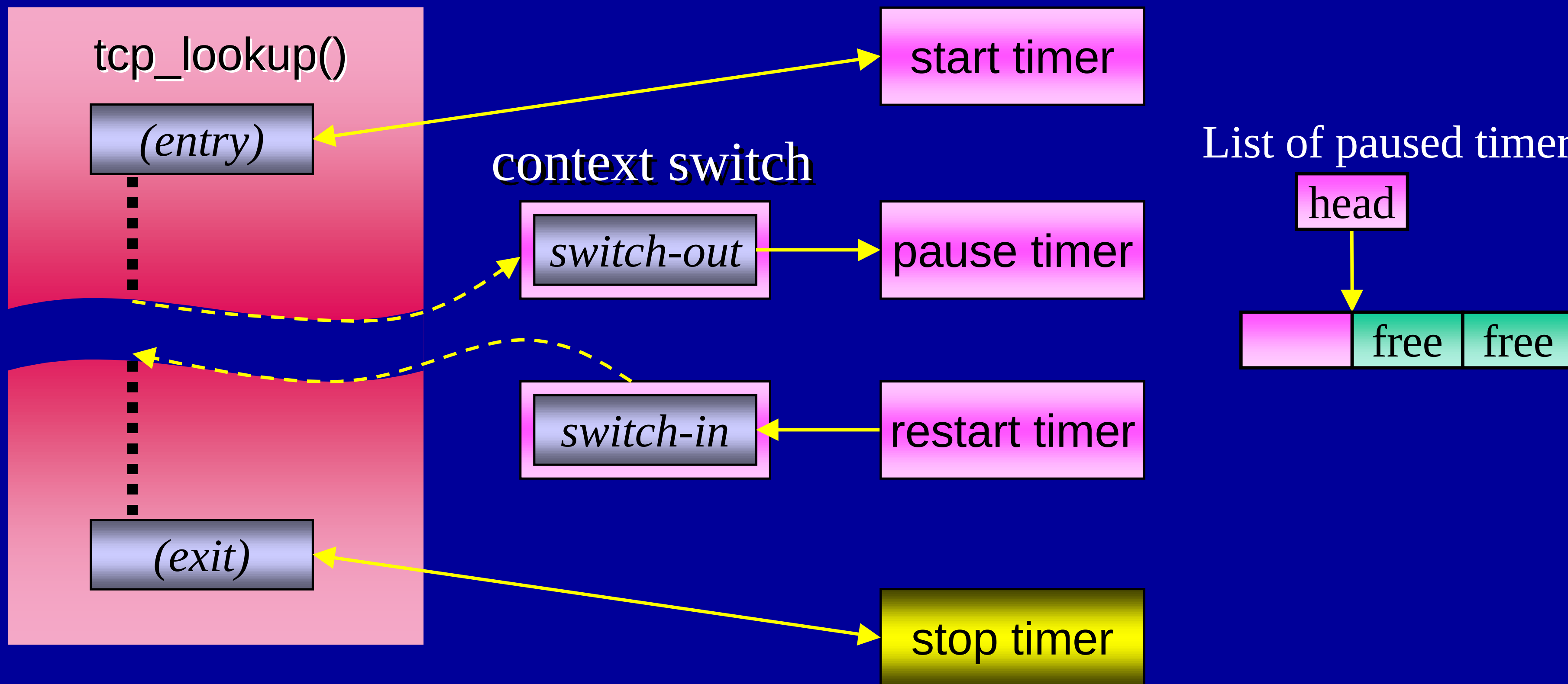
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Non-MP CPU timer primitive



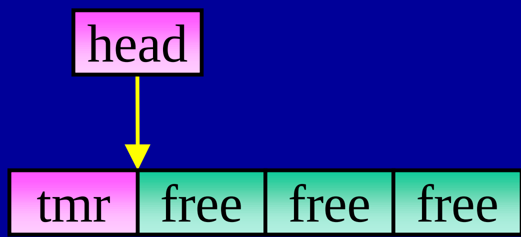
- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism

Kperfmon-MP: Goals

Modify uniprocessor Kperfmon to provide:

- Safe operation on SMP machines
  - Thread safety
  - Migration safety
- New feature: Per-CPU performance data
  - More detailed performance data
  - Reduce cache coherence traffic caused by the tool

Thread Safety

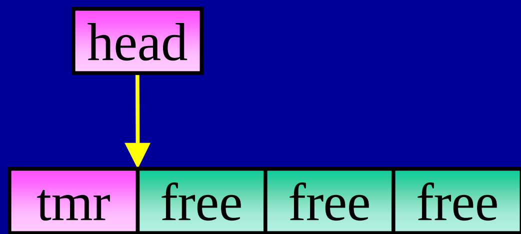


Non-MP timer allocation routine

```
ld [head], R1
add R1, 4, R1
st R1, [head]
```

- Used on switch-out to save the paused timers
- Context switch is serial on uniprocessors
  - No thread safety problems there
- Context switches may be concurrent on SMPs!
  - Multiple threads are being scheduled simultaneously
  - The allocation code is no longer safe

Thread Safety

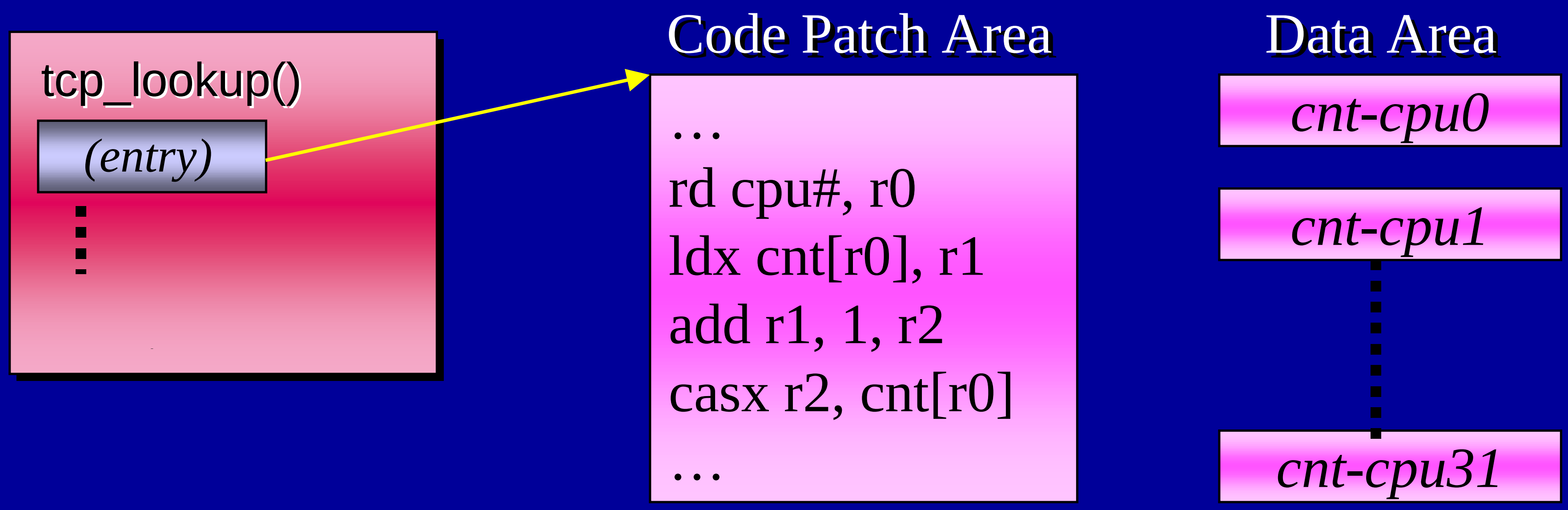


MP timer allocation routine

```
alloc:
    ld [head], R1
    add R1, 4, R2
    cas [head], R1, R2
    cmp R1, R2
    bne alloc
```

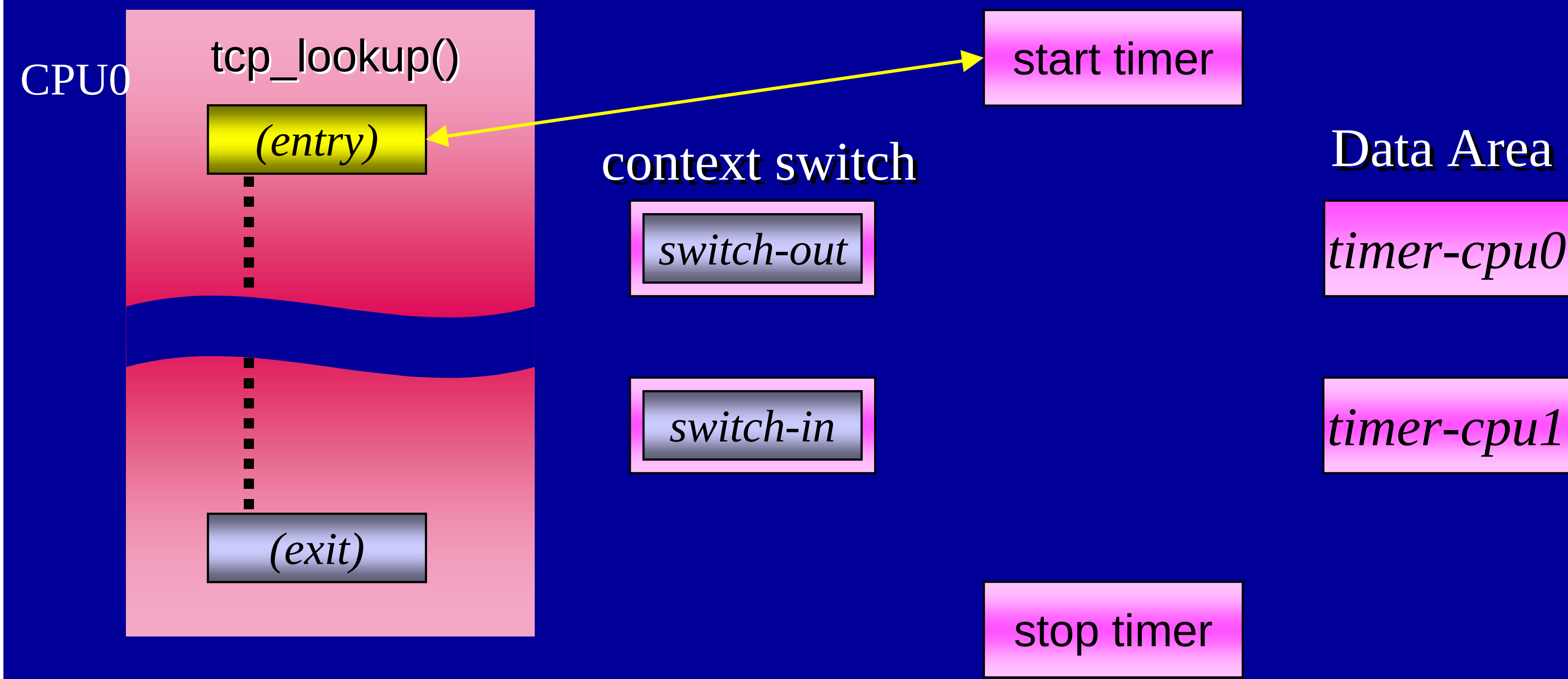
- Context switches may be concurrent on SMPs
- Use the atomic cas instruction to ensure safety

Per-CPU performance data



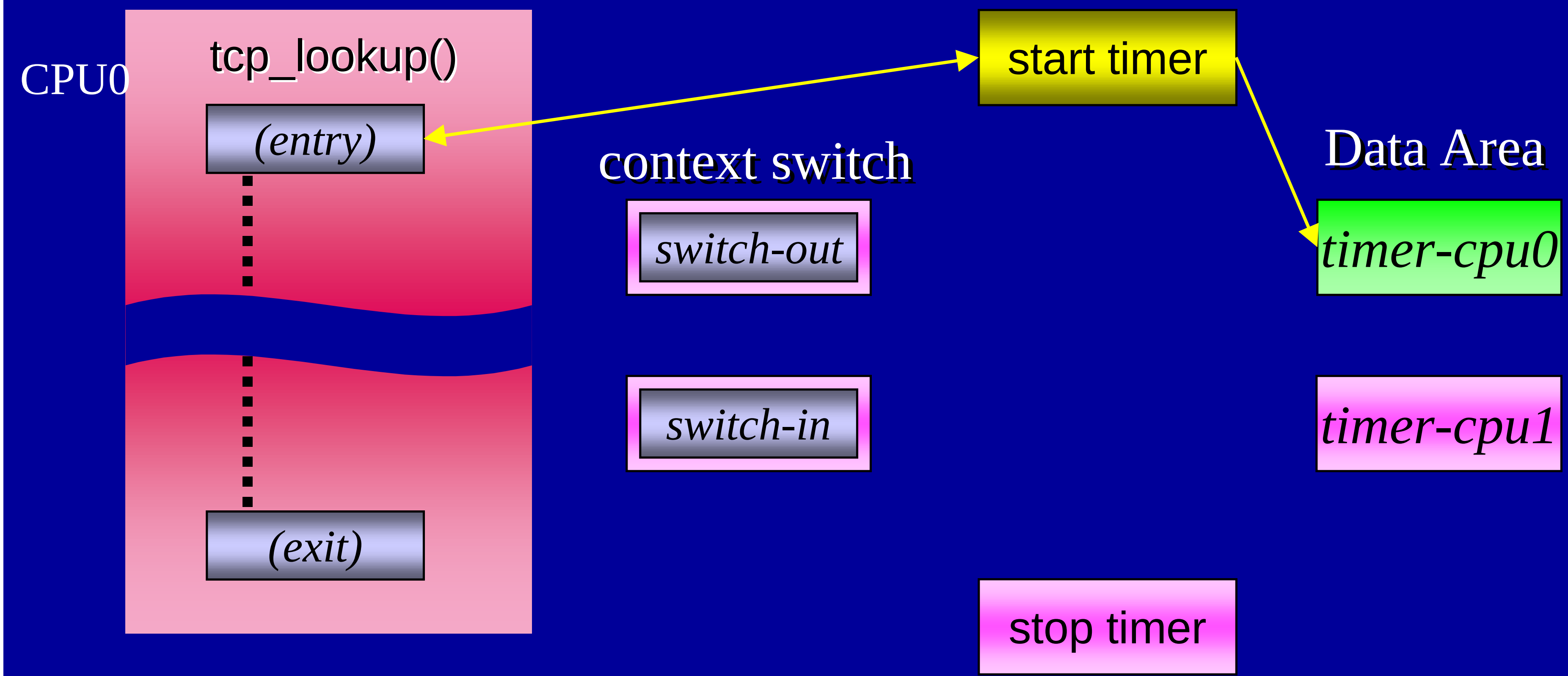
- Instrumentation code is shared by all CPUs
- Per-CPU copies of the primitive's data
  - Two copies are never placed in the same cache line

Migration Between Primitives



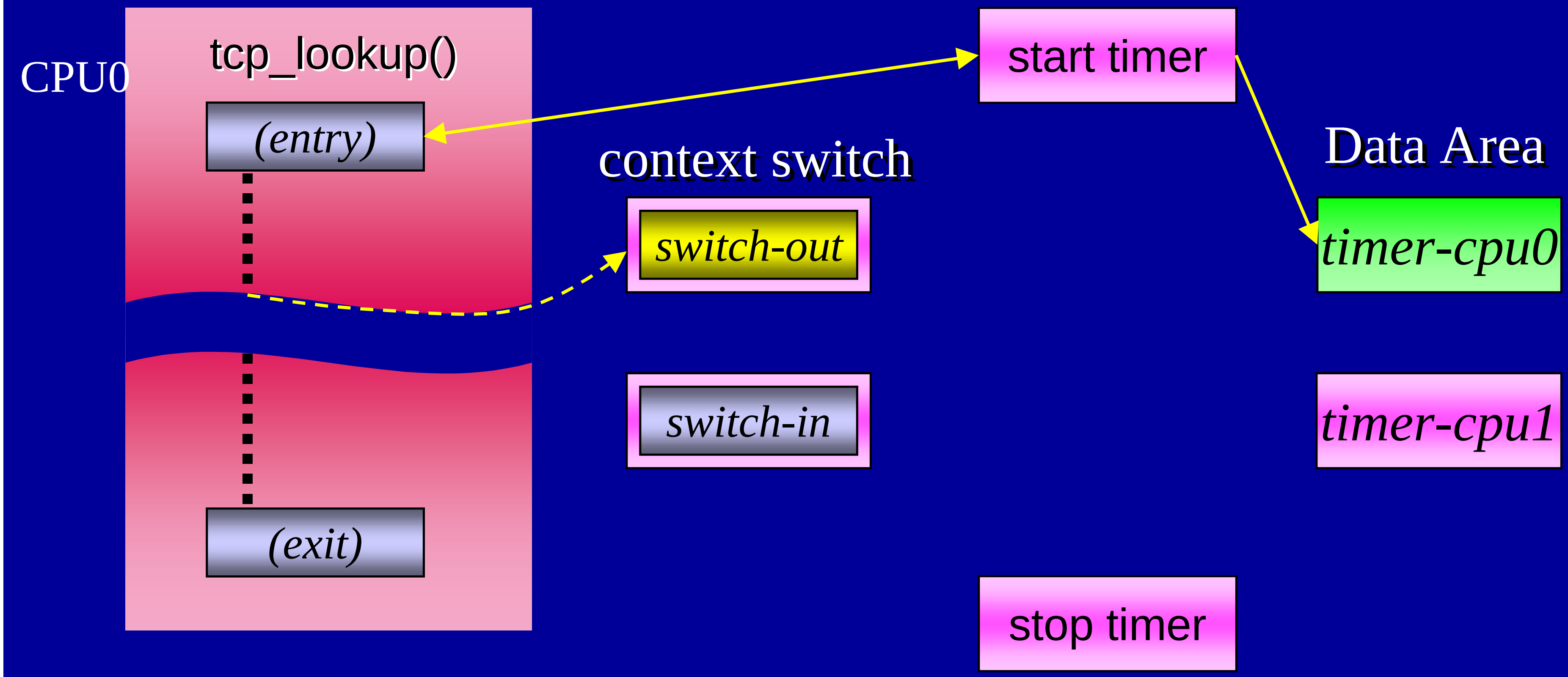
- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected

Migration Between Primitives



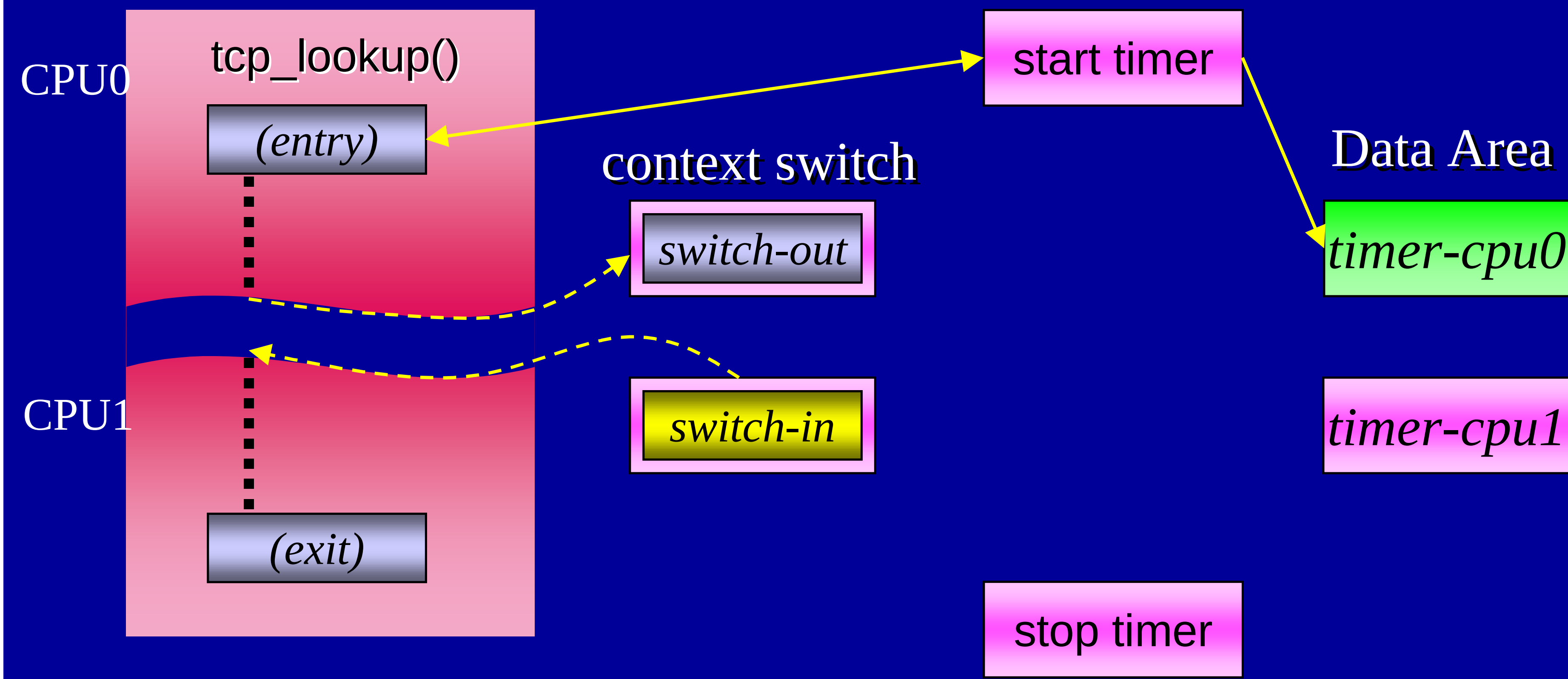
- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected

Migration Between Primitives



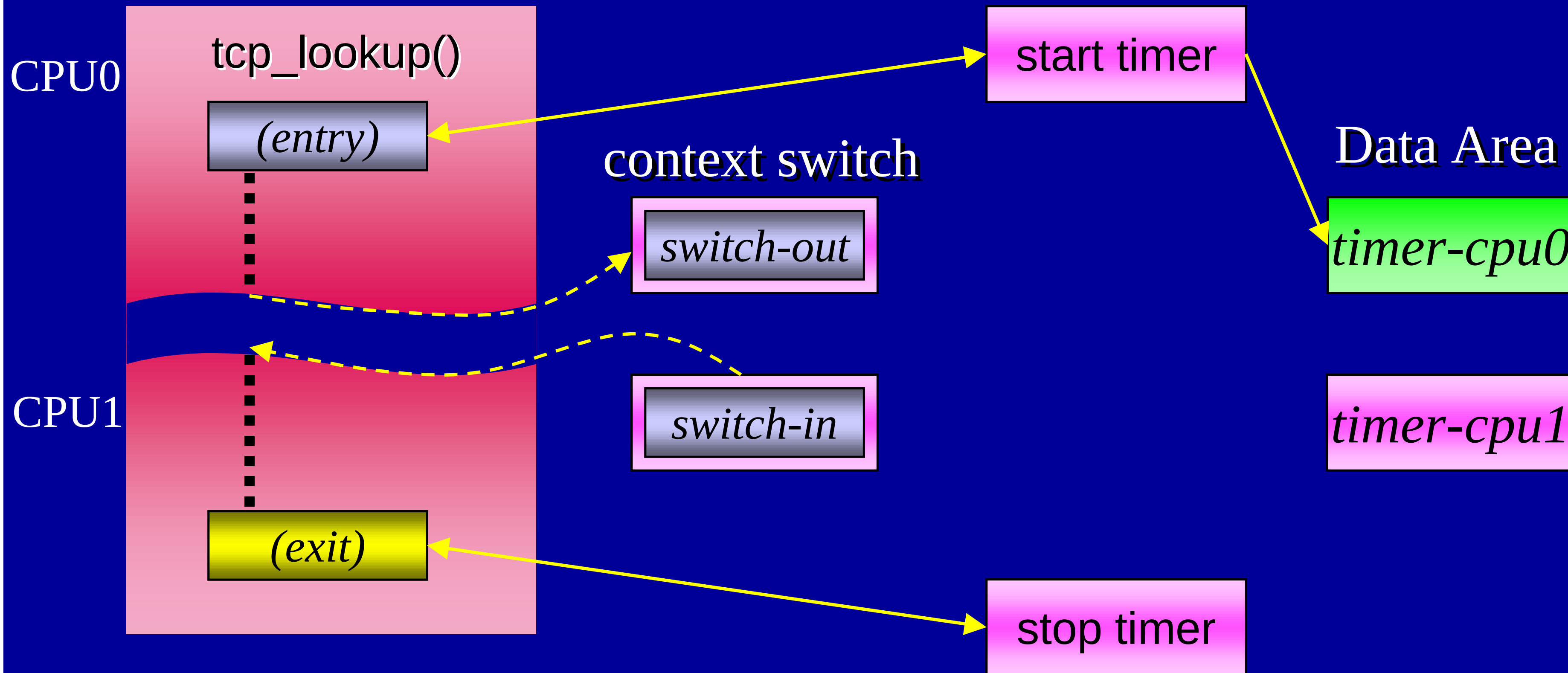
- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected

Migration Between Primitives



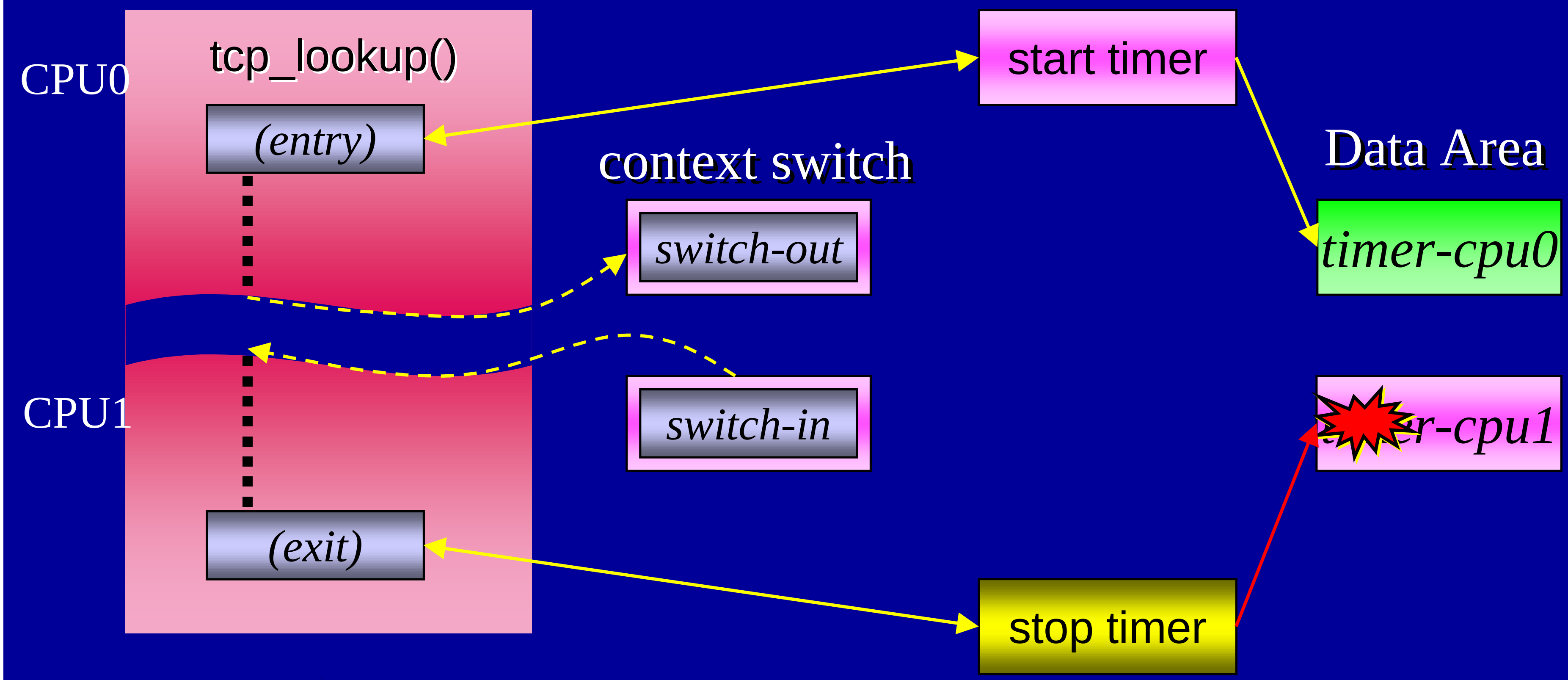
- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected

Migration Between Primitives



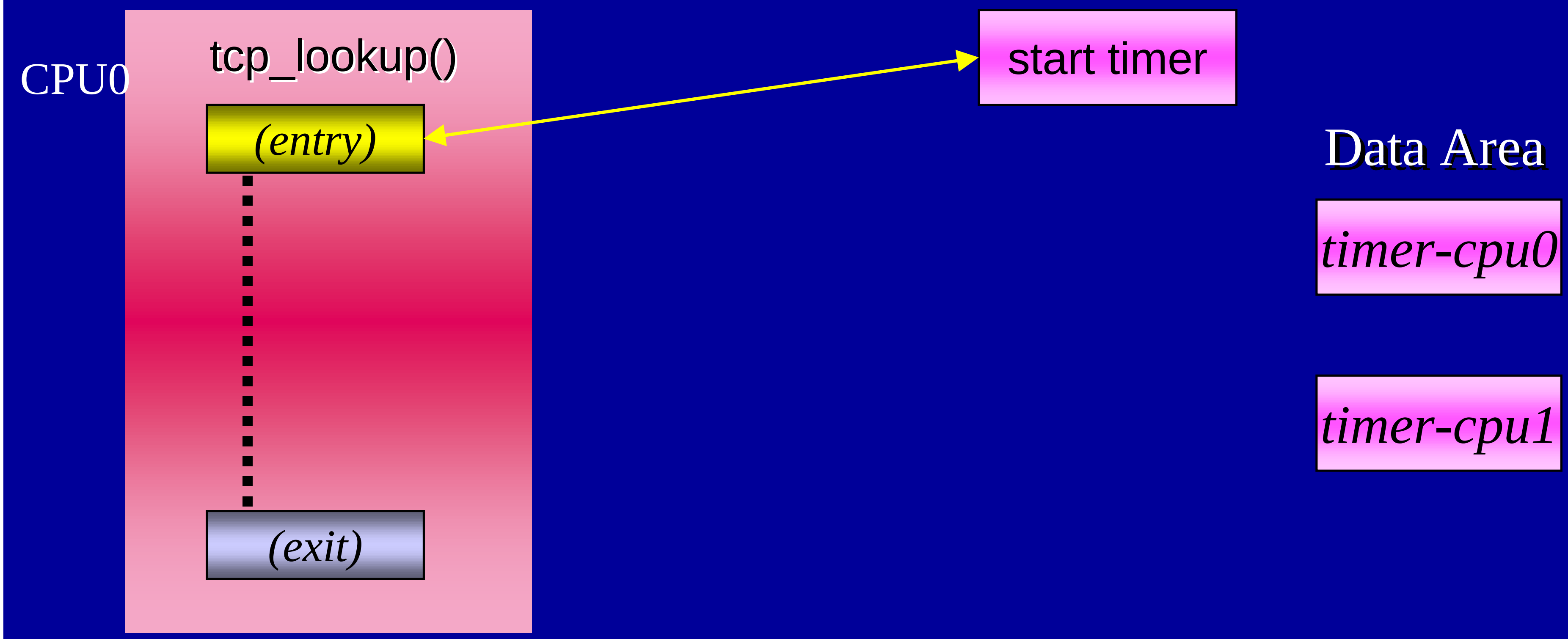
- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected

Migration Between Primitives



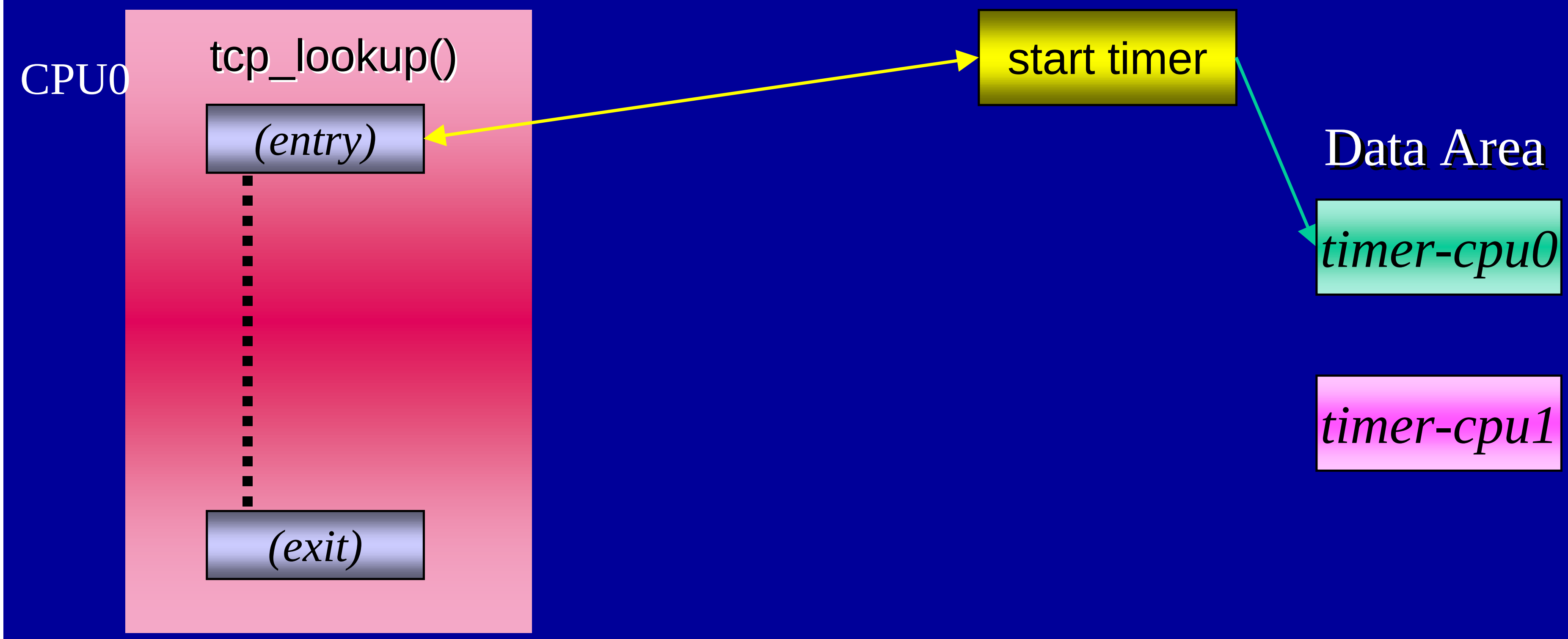
- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected

Solution: virtualization



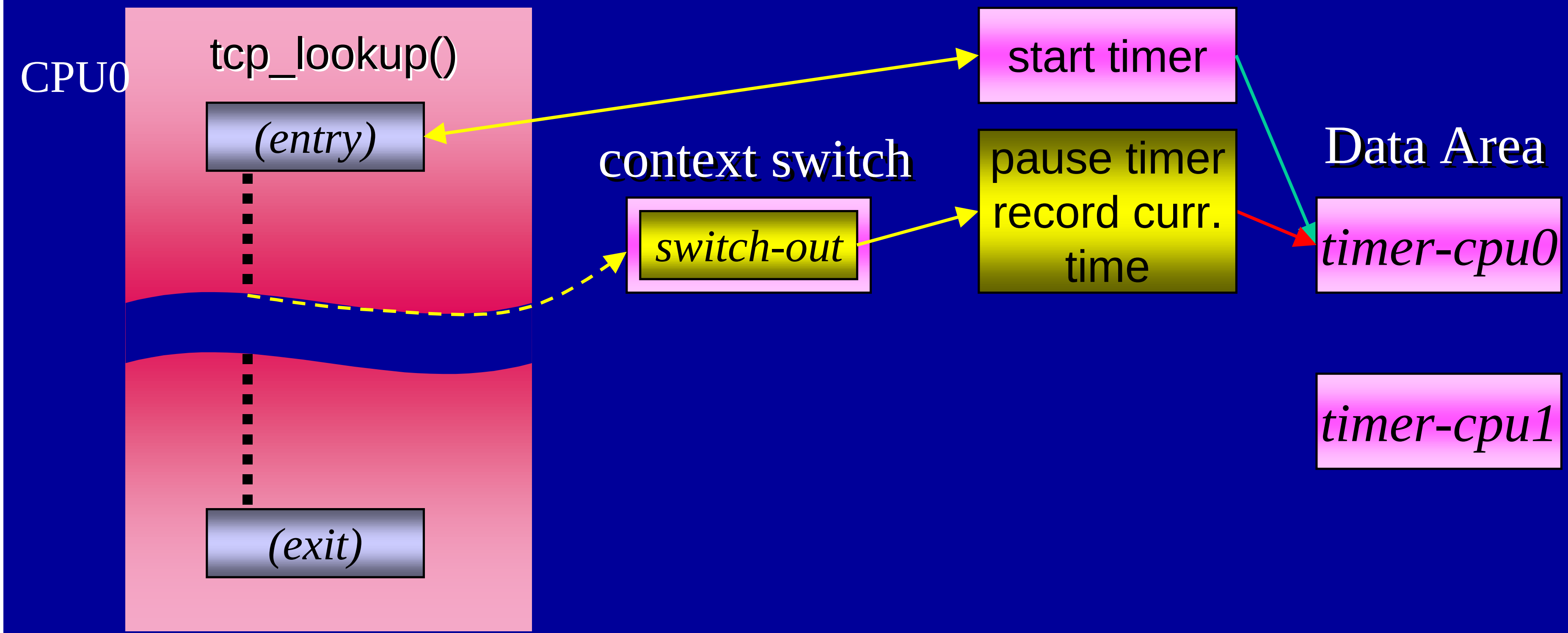
- Implement wall timers on top of CPU timers!

Solution: virtualization



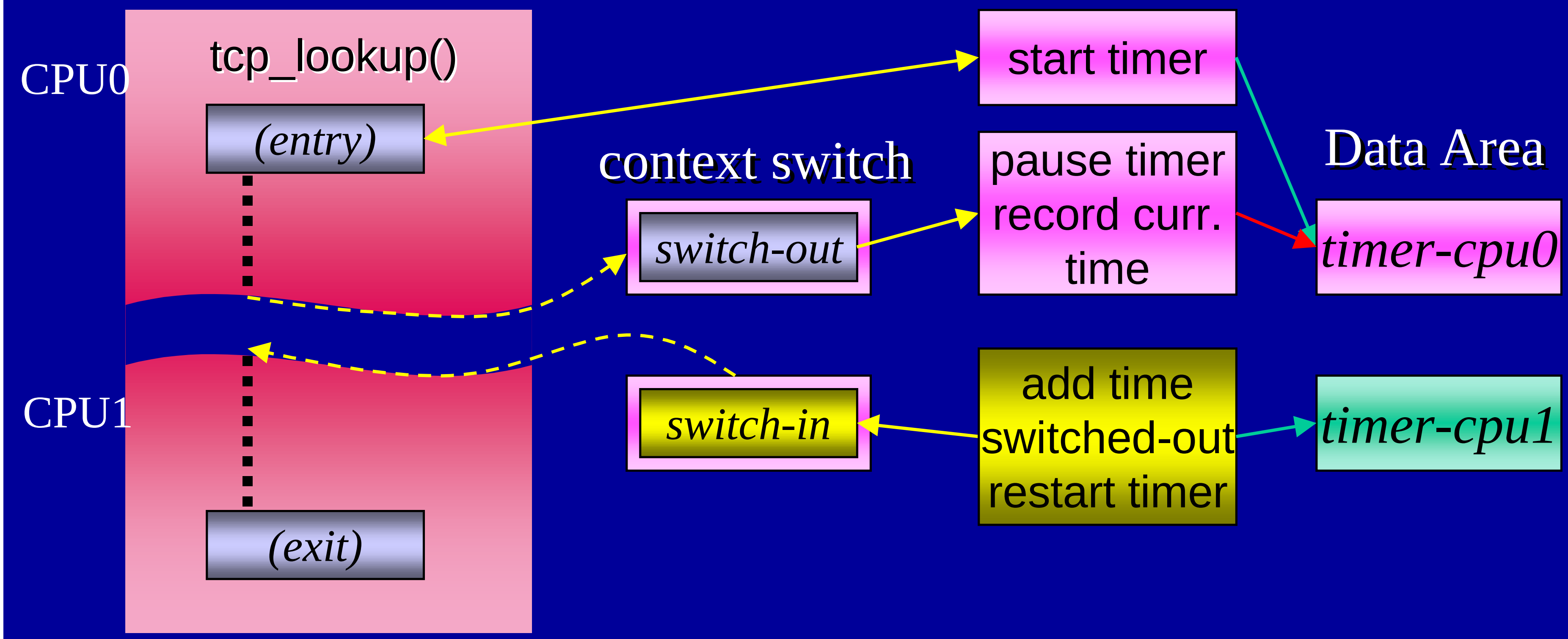
- Implement wall timers on top of CPU timers!

Solution: virtualization



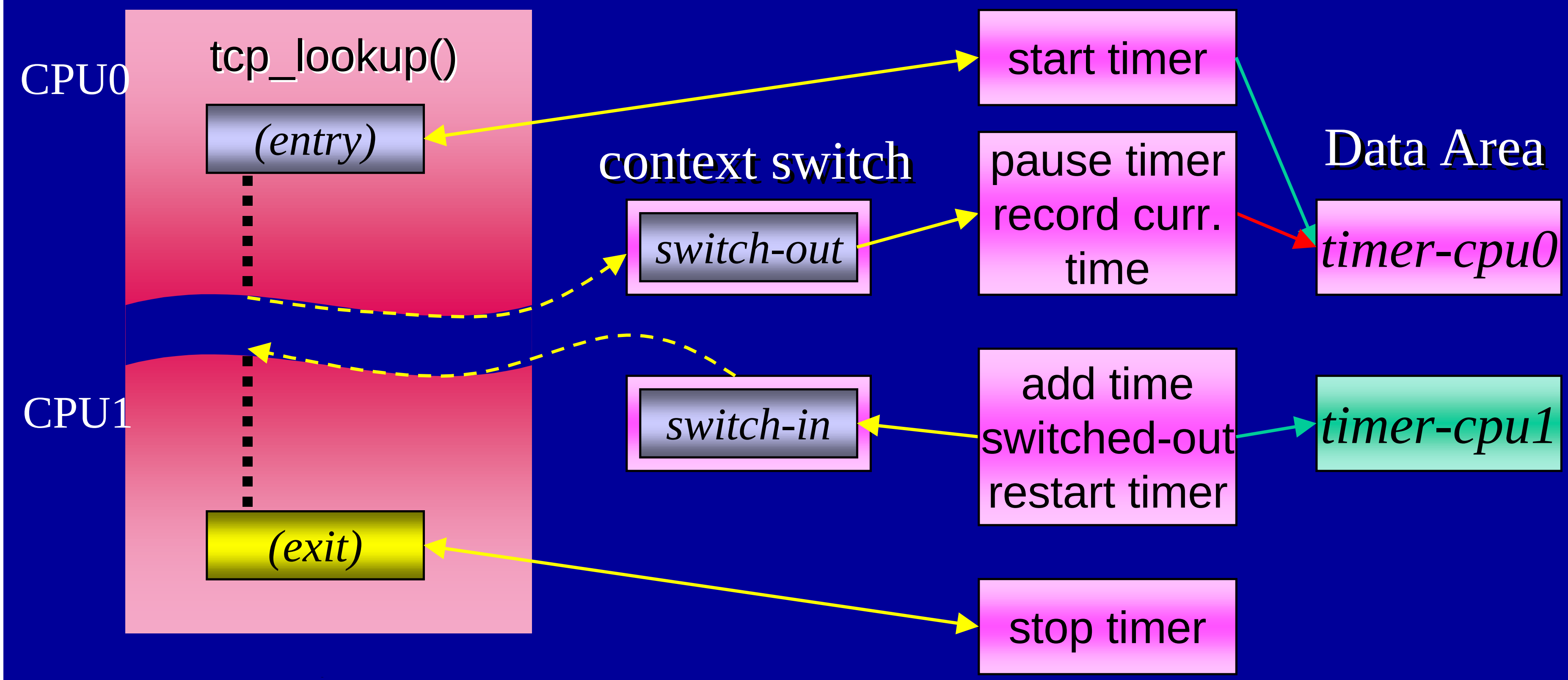
- Implement wall timers on top of CPU timers!

Solution: virtualization



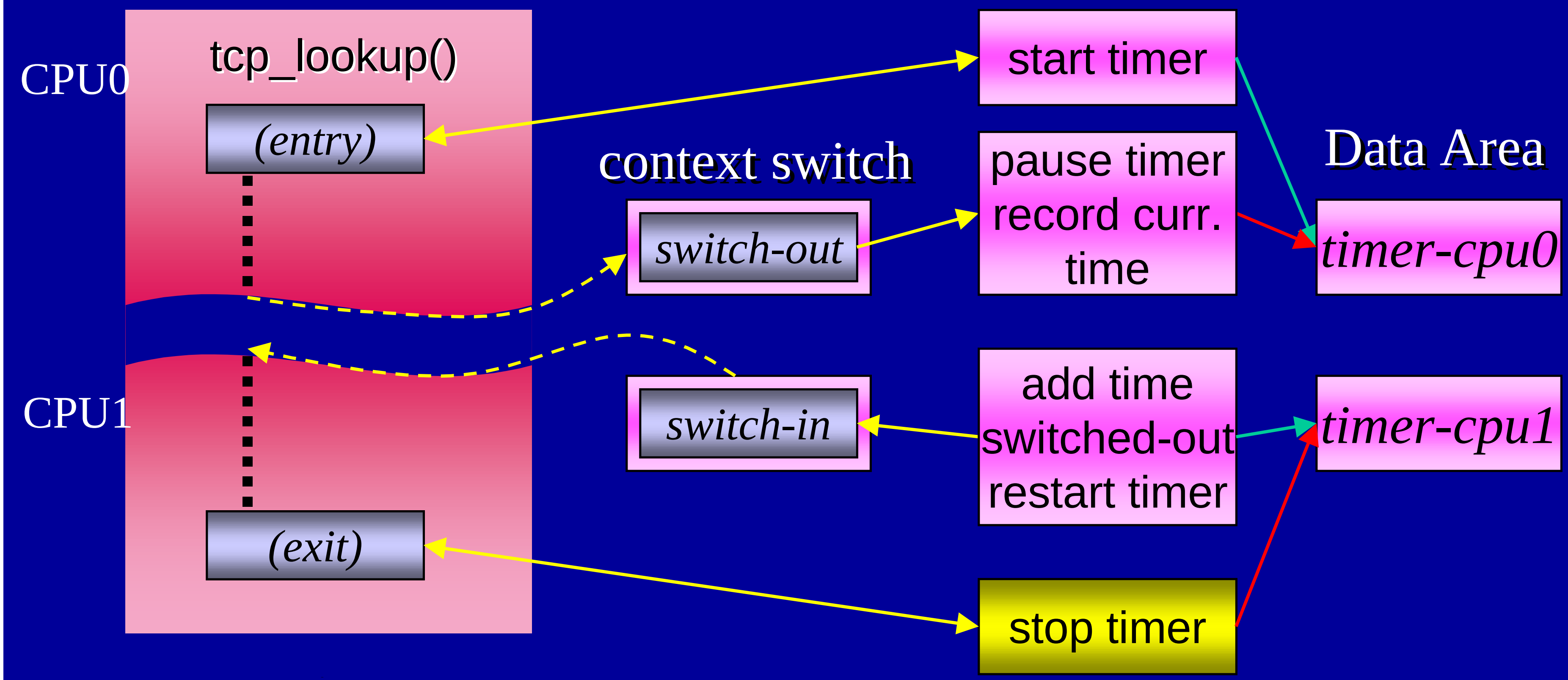
- Implement wall timers on top of CPU timers!

Solution: virtualization

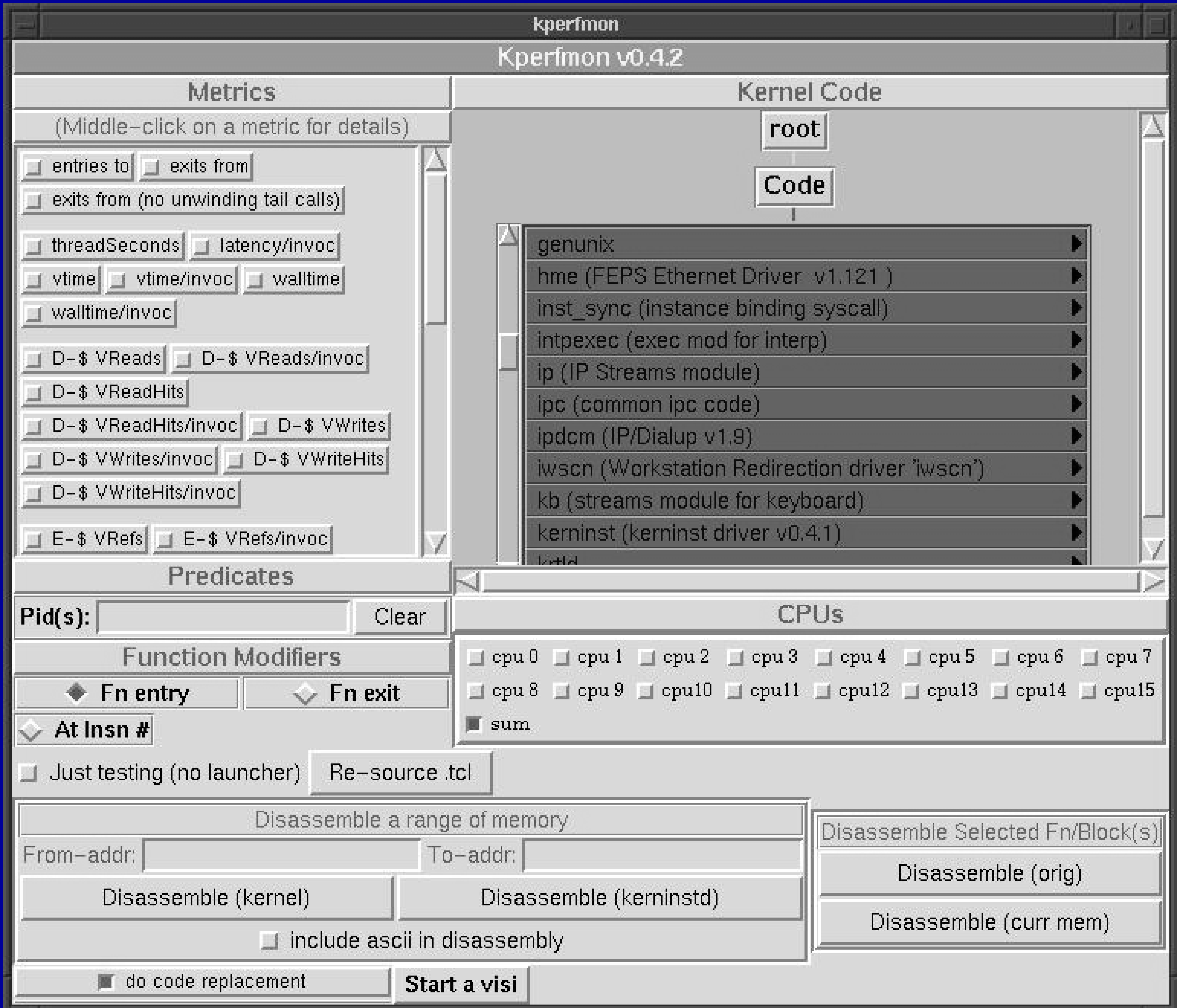


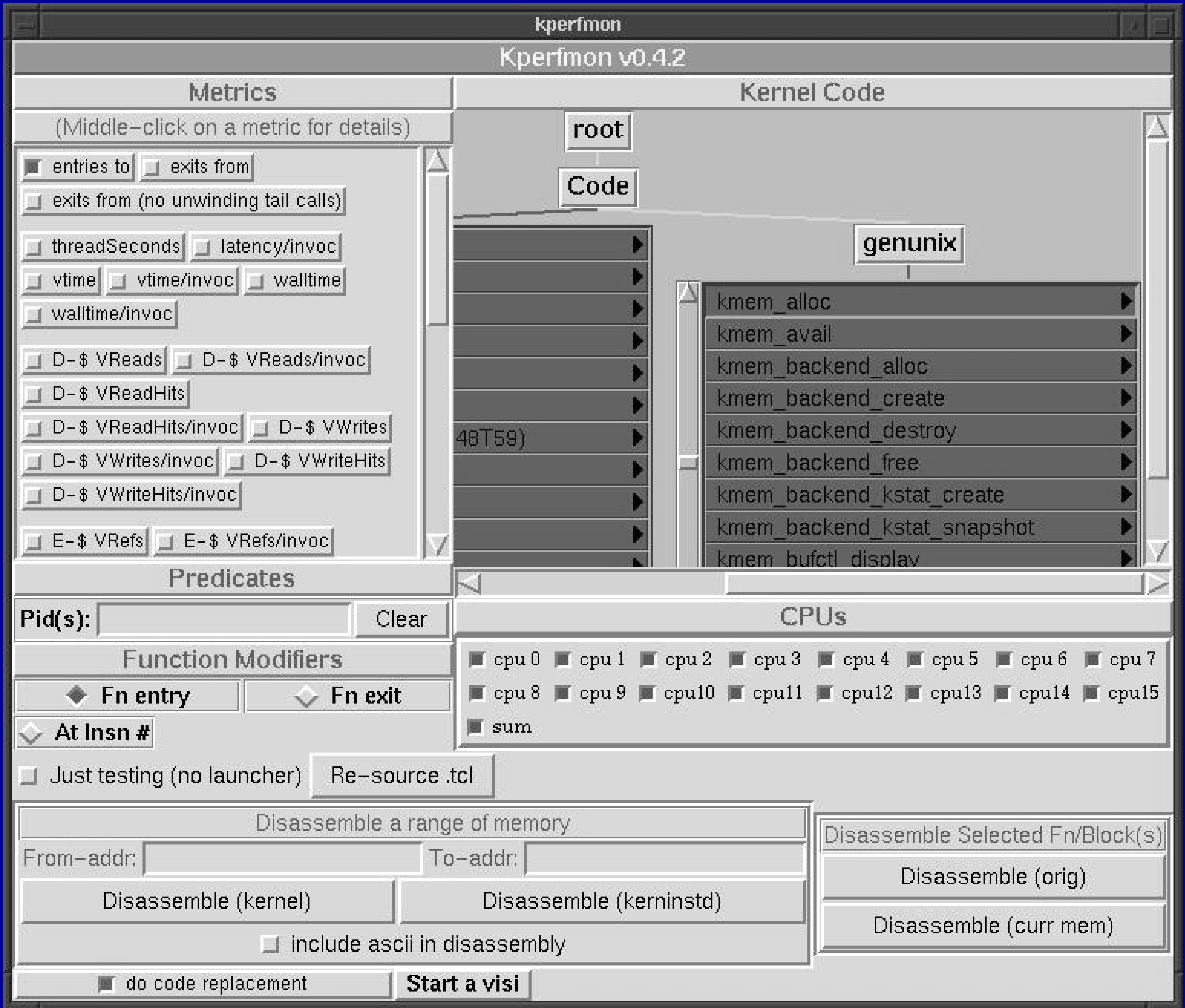
- Implement wall timers on top of CPU timers!

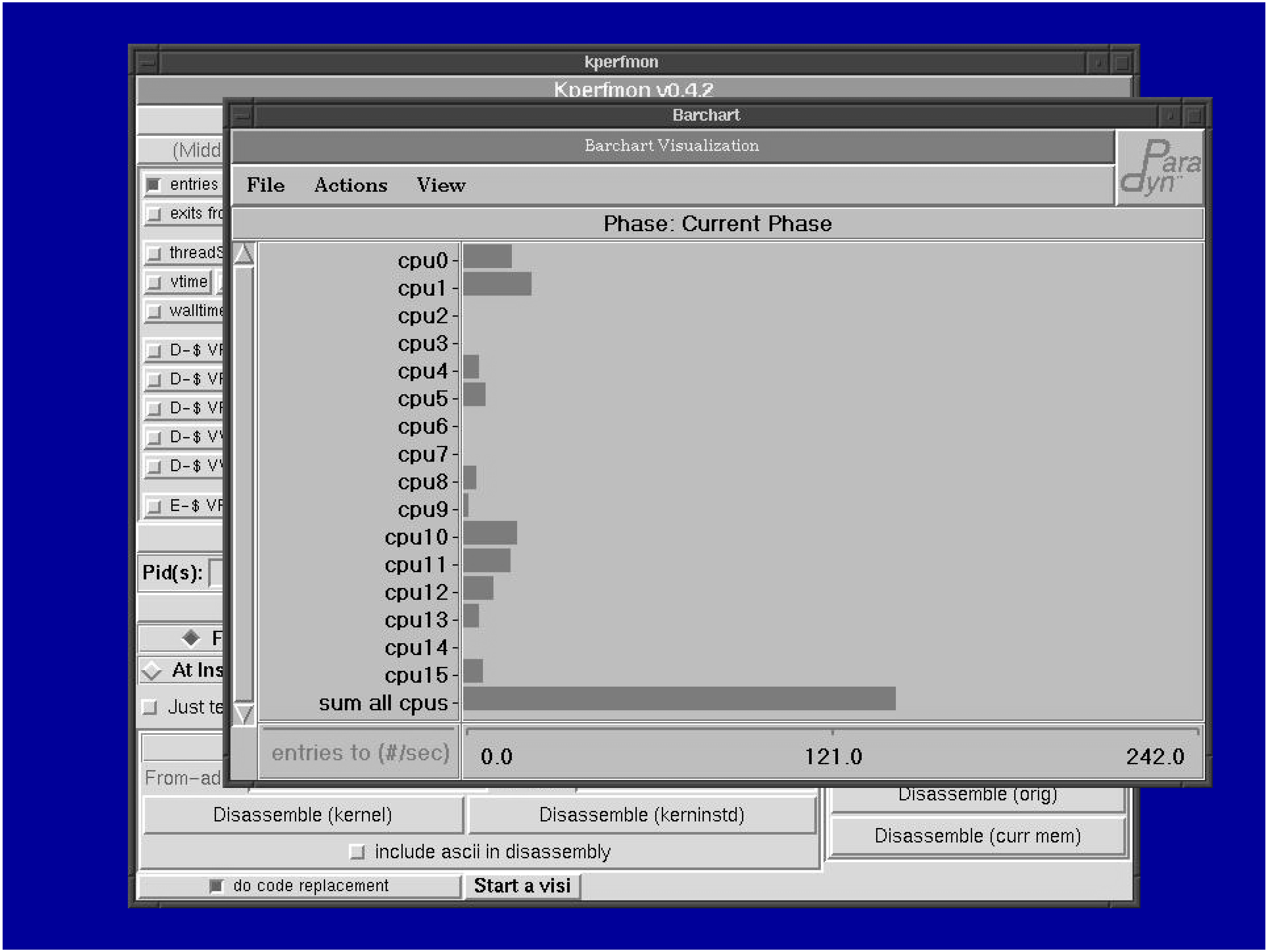
Solution: virtualization

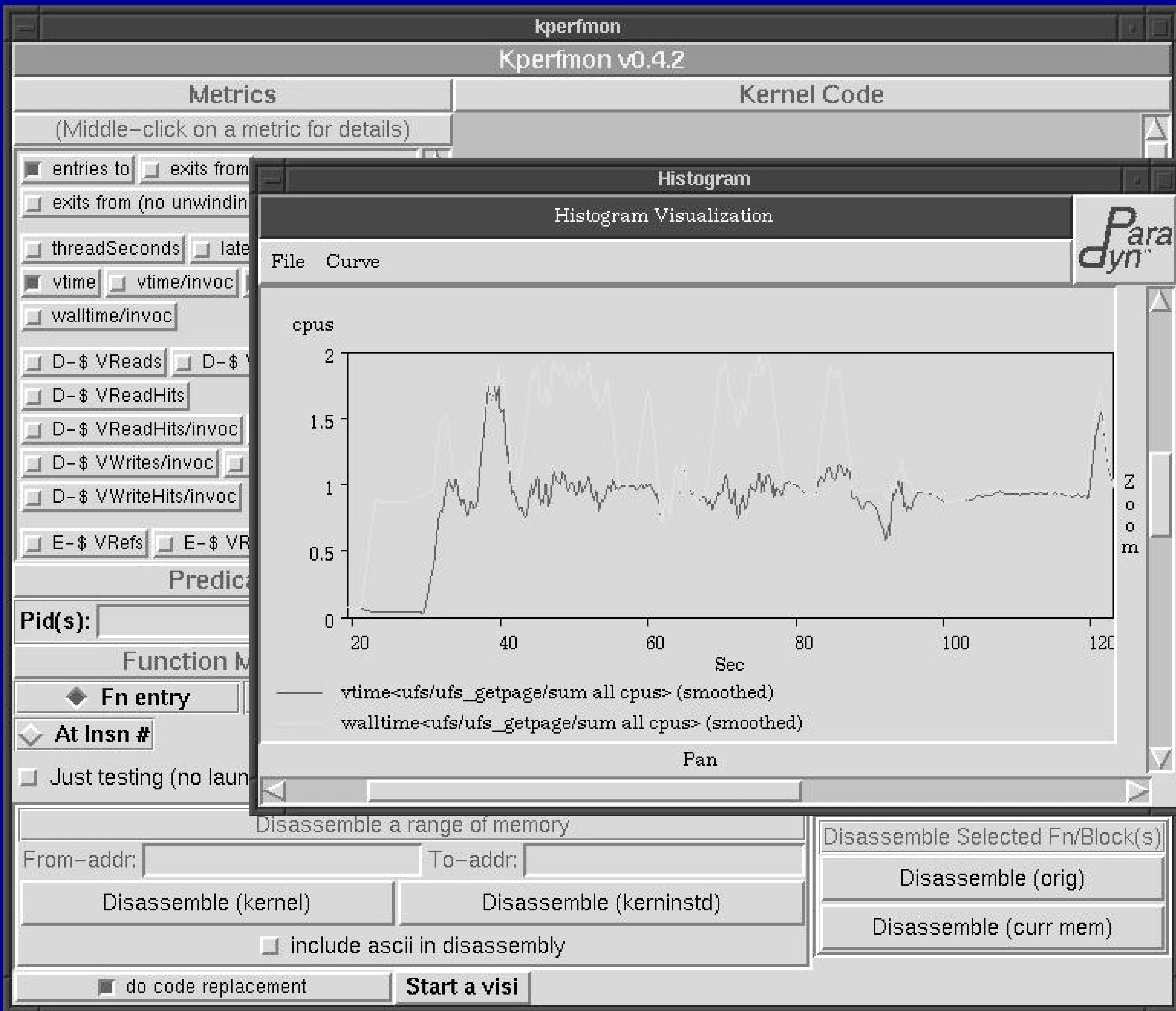


- Implement wall timers on top of CPU timers!

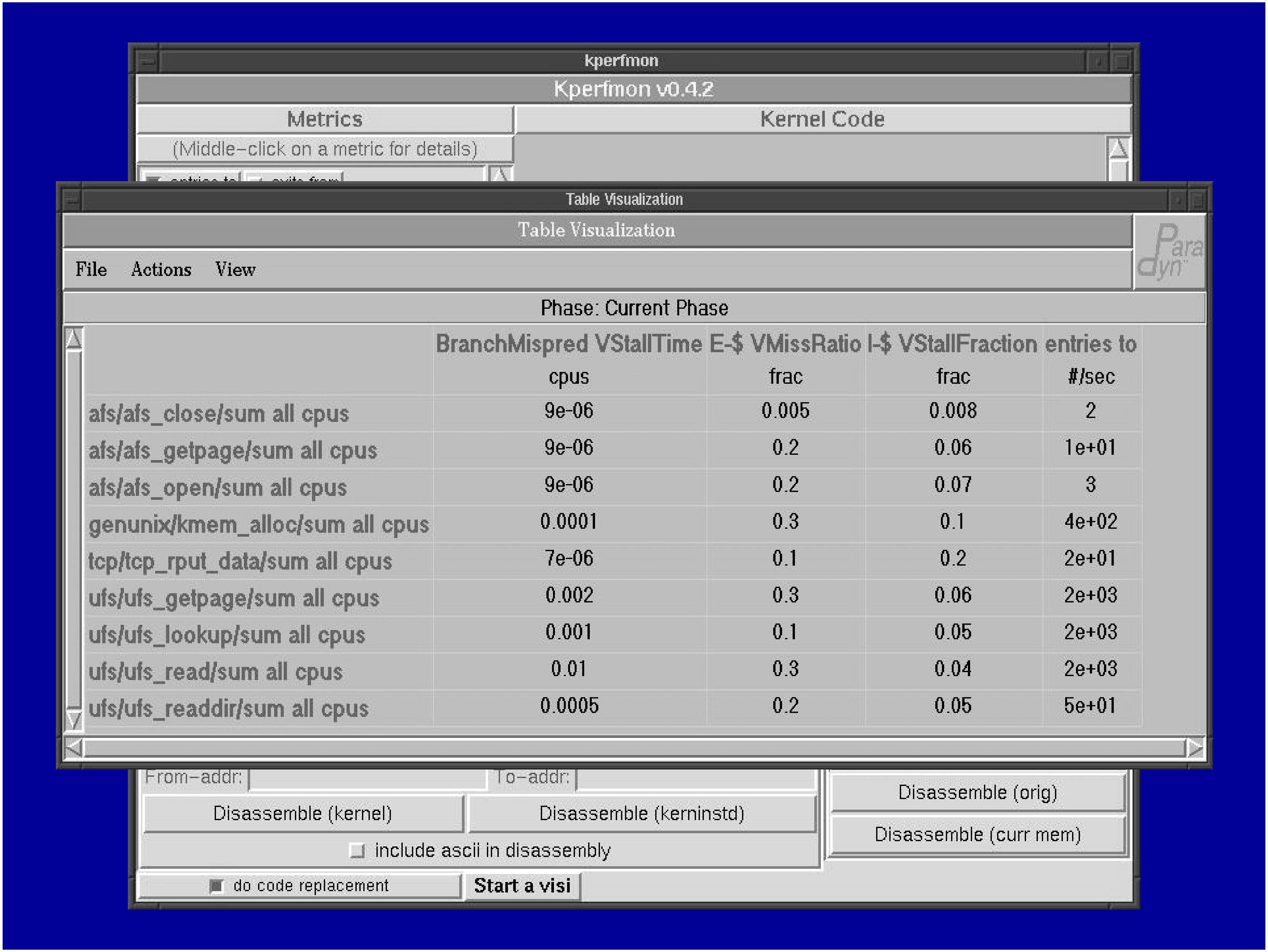

kperfmon
Kperfmon v0.4.2
Metrics
(Middle-click on a metric for details)
entries to
exits from
exits from (no unwinding tail calls)
threadSeconds
latency/invoc
vtime
vtime/invoc
walltime
walltime/invoc
D-$ VReads
D-$ VReads/invoc
D-$ VReadHits
D-$ VReadHits/invoc
D-$ VWrites
D-$ VWrites/invoc
D-$ VWriteHits
D-$ VWriteHits/invoc
E-$ VRefs
E-$ VRefs/invoc
Kernel Code
root
Code
genunix
hme (FEPS Ethernet Driver v1.121)
inst_sync (instance binding syscall)
intpexec (exec mod for interp)
ip (IP Streams module)
ipc (common ipc code)
ipdcm (IP/Dialup v1.9)
iwscn (Workstation Redirection driver 'iwscn')
kb (streams module for keyboard)
kerninst (kerninst driver v0.4.1)
Predicates
Pid(s):
Clear
CPUs
cpu 0
cpu 1
cpu 2
cpu 3
cpu 4
cpu 5
cpu 6
cpu 7
cpu 8
cpu 9
cpu10
cpu11
cpu12
cpu13
cpu14
cpu15
sum
Function Modifiers
Fn entry
Fn exit
At Insn #
Just testing (no launcher)
Re-source .tcl
Disassemble a range of memory
From-addr:
To-addr:
Disassemble (kernel)
Disassemble (kerninstd)
include ascii in disassembly
Disassemble Selected Fn/Block(s)
Disassemble (orig)
Disassemble (curr mem)
do code replacement
Start a visi


kperfmon
Kperfmon v0.4.2
Metrics
(Middle-click on a metric for details)
entries to
exits from
exits from (no unwinding tail calls)
threadSeconds
latency/invoc
vtime
vtime/invoc
walltime
walltime/invoc
D-$ VReads
D-$ VReads/invoc
D-$ VReadHits
D-$ VReadHits/invoc
D-$ VWrites
D-$ VWrites/invoc
D-$ VWriteHits
D-$ VWriteHits/invoc
E-$ VRefs
E-$ VRefs/invoc
Kernel Code
root
Code
genunix
48T59)
kmem_alloc
kmem_avail
kmem_backend_alloc
kmem_backend_create
kmem_backend_destroy
kmem_backend_free
kmem_backend_kstat_create
kmem_backend_kstat_snapshot
kmem_bufctl_display
Predicates
Pid(s):
Clear
CPUs
cpu 0
cpu 1
cpu 2
cpu 3
cpu 4
cpu 5
cpu 6
cpu 7
cpu 8
cpu 9
cpu10
cpu11
cpu12
cpu13
cpu14
cpu15
sum
Function Modifiers
Fn entry
Fn exit
At Insn #
Just testing (no launcher)
Re-source .tcl
Disassemble a range of memory
From-addr:
To-addr:
Disassemble (kernel)
Disassemble (kerninstd)
include ascii in disassembly
Disassemble Selected Fn/Block(s)
Disassemble (orig)
Disassemble (curr mem)
do code replacement
Start a visi


kperfmon
Kperfmon v0.4.2
Barchart
Barchart Visualization
File
Actions
View
Phase: Current Phase
cpu0
cpu1
cpu2
cpu3
cpu4
cpu5
cpu6
cpu7
cpu8
cpu9
cpu10
cpu11
cpu12
cpu13
cpu14
cpu15
sum all cpus
entries to (#/sec)
0.0
121.0
242.0
Disassemble (kernel)
Disassemble (kerninstd)
Disassemble (orig)
Disassemble (curr mem)
include ascii in disassembly
do code replacement
Start a visi
Pid(s):



kperfmon
Kperfmon v0.4.2
Metrics
Kernel Code
(Middle-click on a metric for details)
entries to
exits from
exits from (no unwindin
threadSeconds
late
vtime
vtime/invoc
walltime/invoc
D-$ VReads
D-$ VReadHits
D-$ VReadHits/invoc
D-$ VWrites/invoc
D-$ VWriteHits/invoc
E-$ VRefs
Predica
Pid(s):
Function M
Fn entry
At Insn #
Just testing (no laun
Histogram
Histogram Visualization
File
Curve
Paradyn
cpus
Sec
Zoom
vtime<ufs/ufs_getpage/sum all cpus> (smoothed)
walltime<ufs/ufs_getpage/sum all cpus> (smoothed)
Pan
Disassemble a range of memory
From-addr:
To-addr:
Disassemble (kernel)
Disassemble (kerninstd)
include ascii in disassembly
Disassemble Selected Fn/Block(s)
Disassemble (orig)
Disassemble (curr mem)
do code replacement
Start a visi


kperfmon
Kperfmon v0.4.2
Metrics
Kernel Code
(Middle-click on a metric for details)
Table Visualization
Table Visualization
File
Actions
View
Phase: Current Phase
BranchMispred VStallTime E-$ VMissRatio I-$ VStallFraction entries to
cpus frac frac #/sec
afs/afs_close/sum all cpus 9e-06 0.005 0.008 2
afs/afs_getpage/sum all cpus 9e-06 0.2 0.06 1e+01
afs/afs_open/sum all cpus 9e-06 0.2 0.07 3
genunix/kmem_alloc/sum all cpus 0.0001 0.3 0.1 4e+02
tcp/tcp_rput_data/sum all cpus 7e-06 0.1 0.2 2e+01
ufs/ufs_getpage/sum all cpus 0.002 0.3 0.06 2e+03
ufs/ufs_lookup/sum all cpus 0.001 0.1 0.05 2e+03
ufs/ufs_read/sum all cpus 0.01 0.3 0.04 2e+03
ufs/ufs_readdir/sum all cpus 0.0005 0.2 0.05 5e+01
From-addr:
To-addr:
Disassemble (orig)
Disassemble (kernel)
Disassemble (kerninstd)
Disassemble (curr mem)
include ascii in disassembly
do code replacement
Start a visi

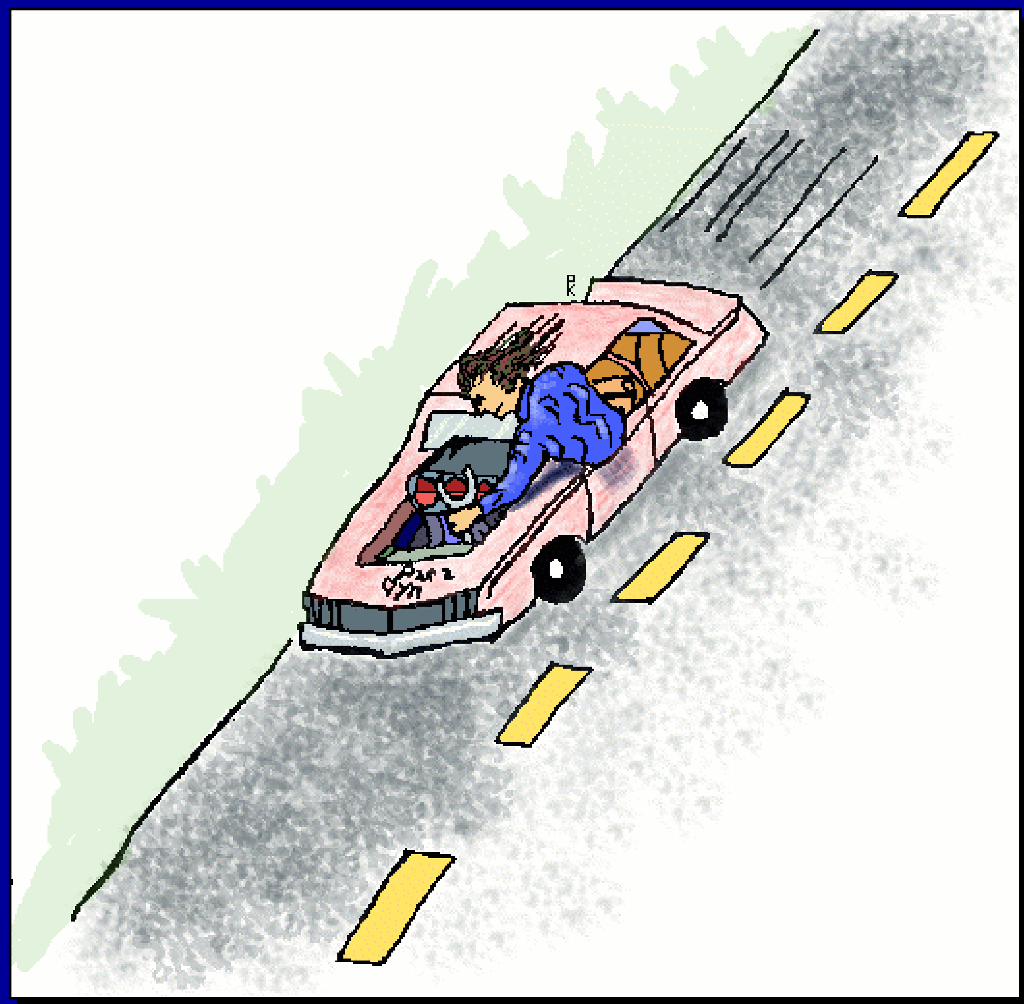
Conclusion

- Techniques for correct MP profiling:
  - Atomic memory updates to ensure thread safety
  - Virtualized timers to handle thread migration
- Per-CPU data collection is important
  - Provides detailed performance information
  - Introduces fewer coherence cache misses

Future Work

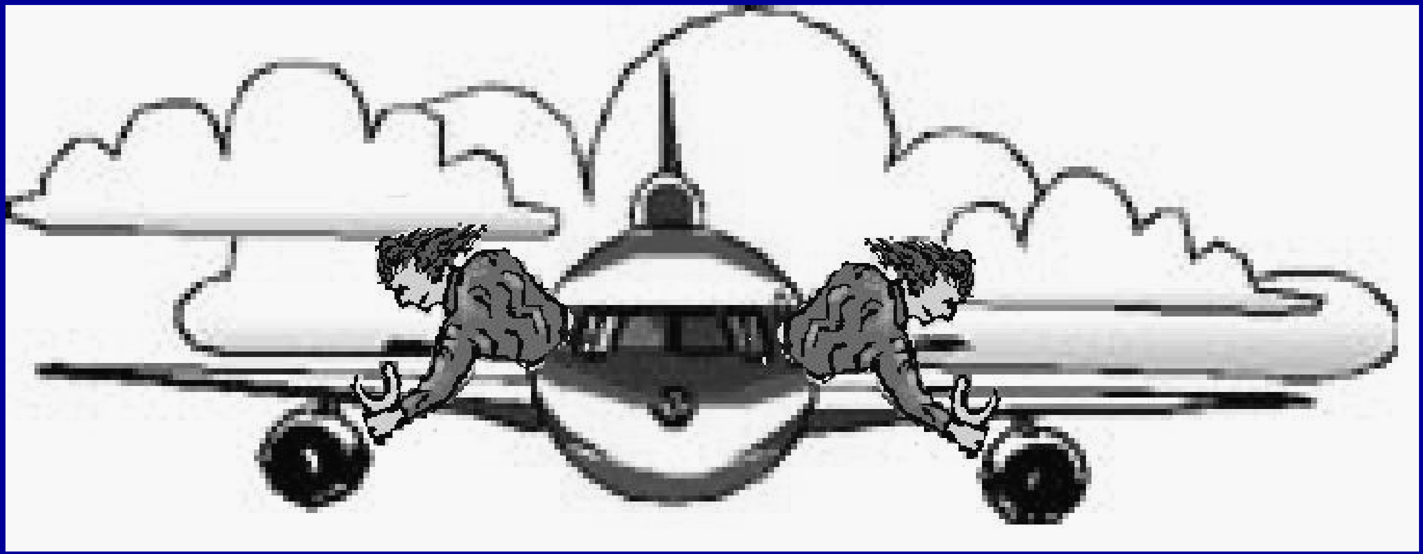
- New metrics
  - Locality of CPU assignments
  - Per-thread performance data
- Formal verification of instrumentation code for migration/preemption problems
- Ports to other architectures and OS'es

The Big Picture



http://www.cs.wisc.edu/paradyn

The Big Picture



- Demo: Wednesday, Room 6372
- Available for download on request
  - mailto: mirg@cs.wisc.edu
  - Public release in April