

# OFFLINE SCHEDULER BOF

Raz Ben Yehuda  
Linux plumbers conference 2009

## CONCEPT

PARTIAL PARTITIONING vs PURE PARTITIONING

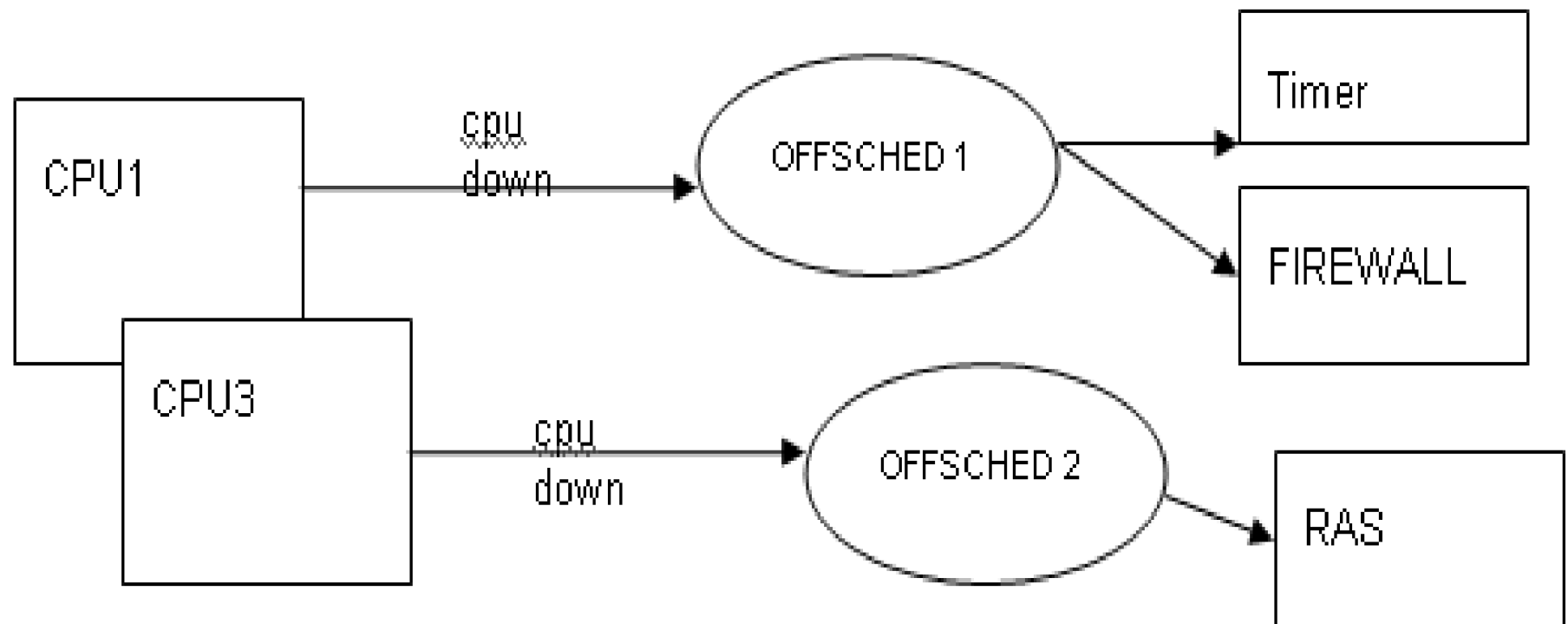
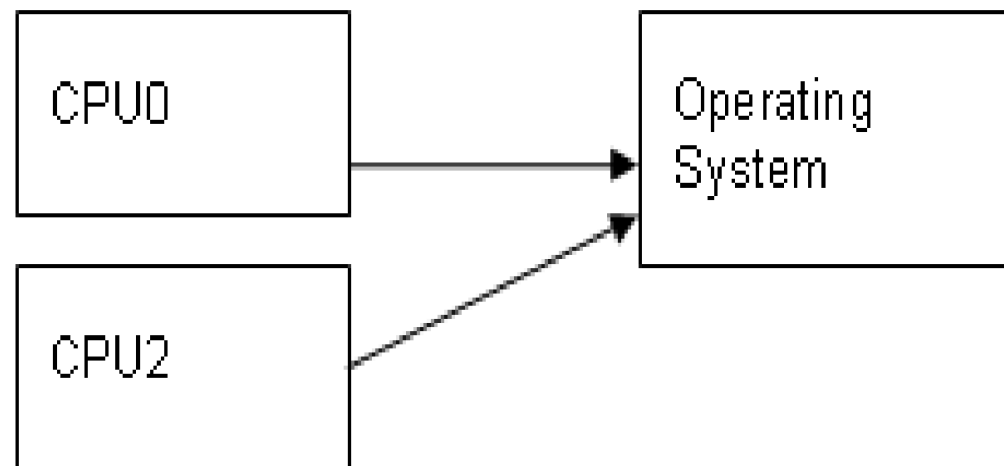
PROCESSOR is a DEVICE vs PROCESSOR is the SYSTEM

## CONS

1. CANNOT RUN USER SPACE TASKS
2. RELEATIVELY HARD TO DEVELOP
3. ONLY FOR SMP/SMT/MC SYSTEMS
4. INDIRECT ACCESS TO VMALLOC AREA
5. NO CONTEXT SWITCH
6. DOES NOT PRESERVE STATES
7. NOT MAINLINE

## PROS

1. PERFORMANCE
2. CONTAINMENT
3. ACCURACY
4. SPEED-UP
5. CONSISTENCY
6. HYBRID SYSTEM
7. ASYMMETRIC PROCESSING
8. INDEPENDENT SYSTEMS
9. BINDING PERIPHERAL TO AN OFFLINE PROCESSOR  
( SMART IO DEVICE)



## OFFLINE REAL TIME

### NMI

- undisturbed ( interrupt-less )
- accurate

### PROGRAM SIZE

- best at many small programs

### ISOLATION

- easier to analyze

### SERIALIZATION

- Full preemption control

### SPEED-UP

- linear – nearly up to BUS contention

### CPU QUIESCE

- No need to walk through a quiesce state

### PATCH OVERHEAD

- Can be used with any Linux kernel that supports processor un-plugging

## OFFLINE REAL TIME - OFFLETS

OFFLET is a context running NMI outside the operating system

OFFLET can be scheduled to a target processor and in a specific point in time.

...

```
int cpu_idx=3;
```

```
offlet x;
```

....

```
offsched_schedule(&x,11,cpu_idx,my_arg);
```

schedule an offlet to run on processor 3 in 11 time units from now.

## The Minimum Patch

```
#ifdef CONFIG_HOTPLUG_CPU
+void (*hotplug_cpu_dead)(void);
+EXPORT_SYMBOL(hotplug_cpu_dead);
DECLARE_PER_CPU(int, cpu_state);

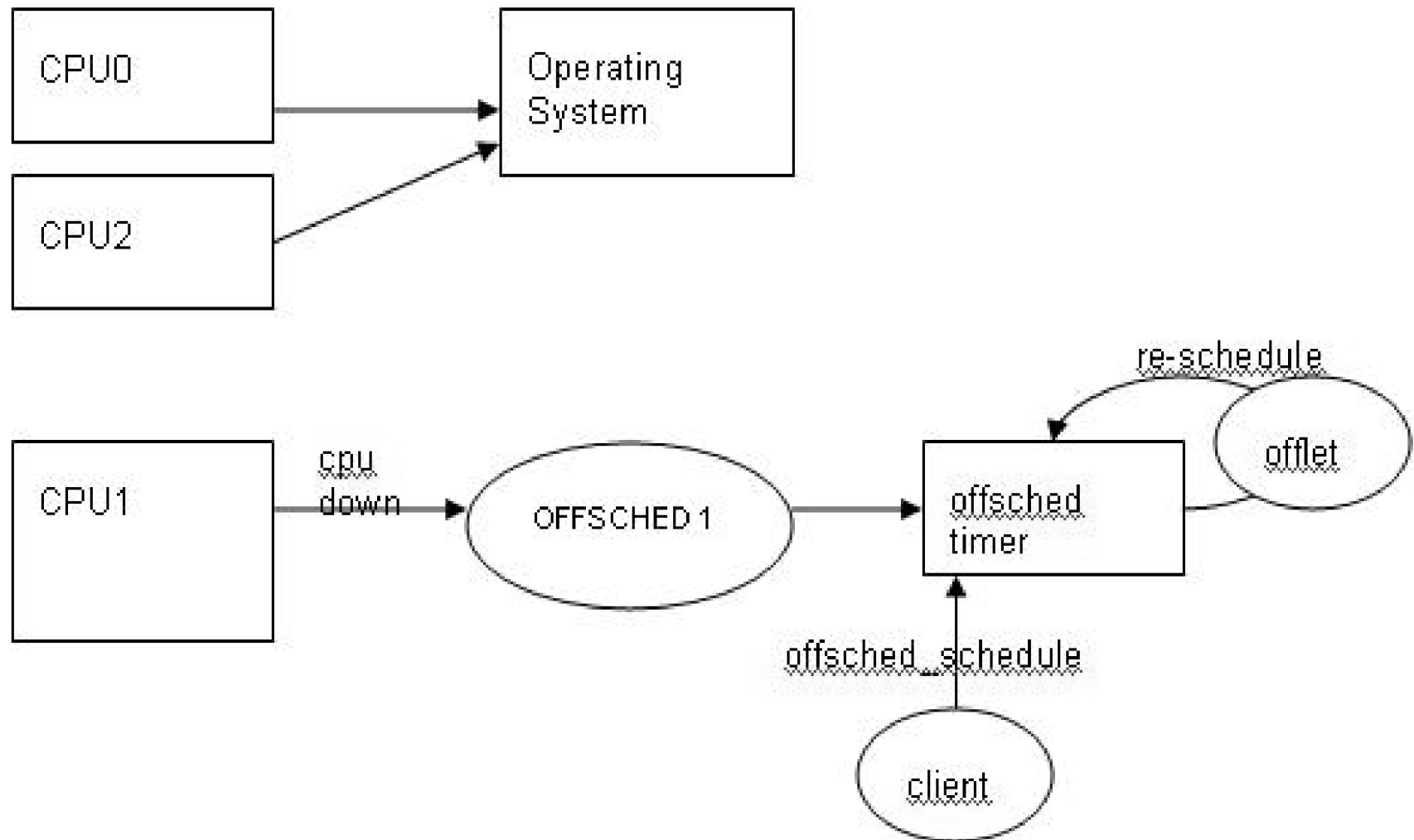
#include <asm/nmi.h>
__get_cpu_var(cpu_state) = CPU_DEAD;

local_irq_disable();
+if (hotplug_cpu_dead)
+hotplug_cpu_dead();
while (1)
    halt();
}
@@ -1265,8 +1265,6 @@
/* They ack this in play_dead by
   setting CPU_DEAD */
if (per_cpu(cpu_state, cpu) == CPU_DEAD) {
-    if (1 == num_online_cpus())
-        alternatives_smp_switch(0);
-        return;
-    }
    msleep(100);
}
```

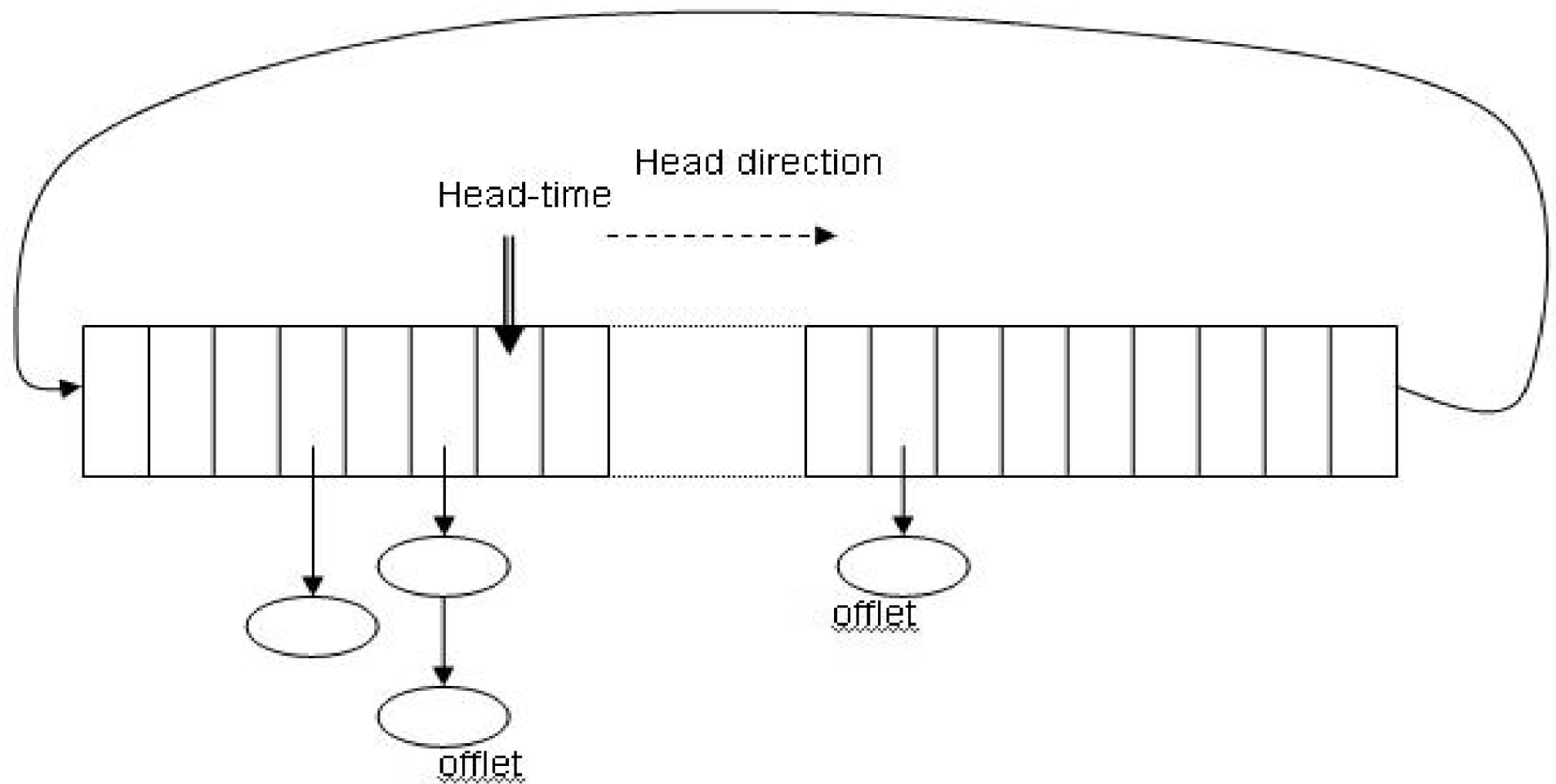
This patch does not include the memory allocation/de-allocation and offline napi.



## Real Time example - 1us timer



## Real Time example - 1us timer



## Real Time example - 1us timer pros and cons

### cons

1. no need in low resolution timers
2. use TICKLESS if can.
3. if the transition from the offlet to the application is too complicated.

### pros

1. the timer interrupt might be nested.
2. amount of work is too much to be handled in interrupt context.
3. Timer deadline varies bellow the defined resolution
4. No HPET available
5. 3% HPET overhead is too much.

## ISOLATION EXAMPLE - RTOP

How do we know what happens in a system when this system is not accessible ?

OFFLINE solution: run a monitoring tool outside the operating system.

## ISOLATION EXAMPLE - RTOP

CentOS-5-32 - VMware Workstation

File Edit View VM Team Windows Help

Home CentOS-5-32

RTOP 0.1 #112 processes

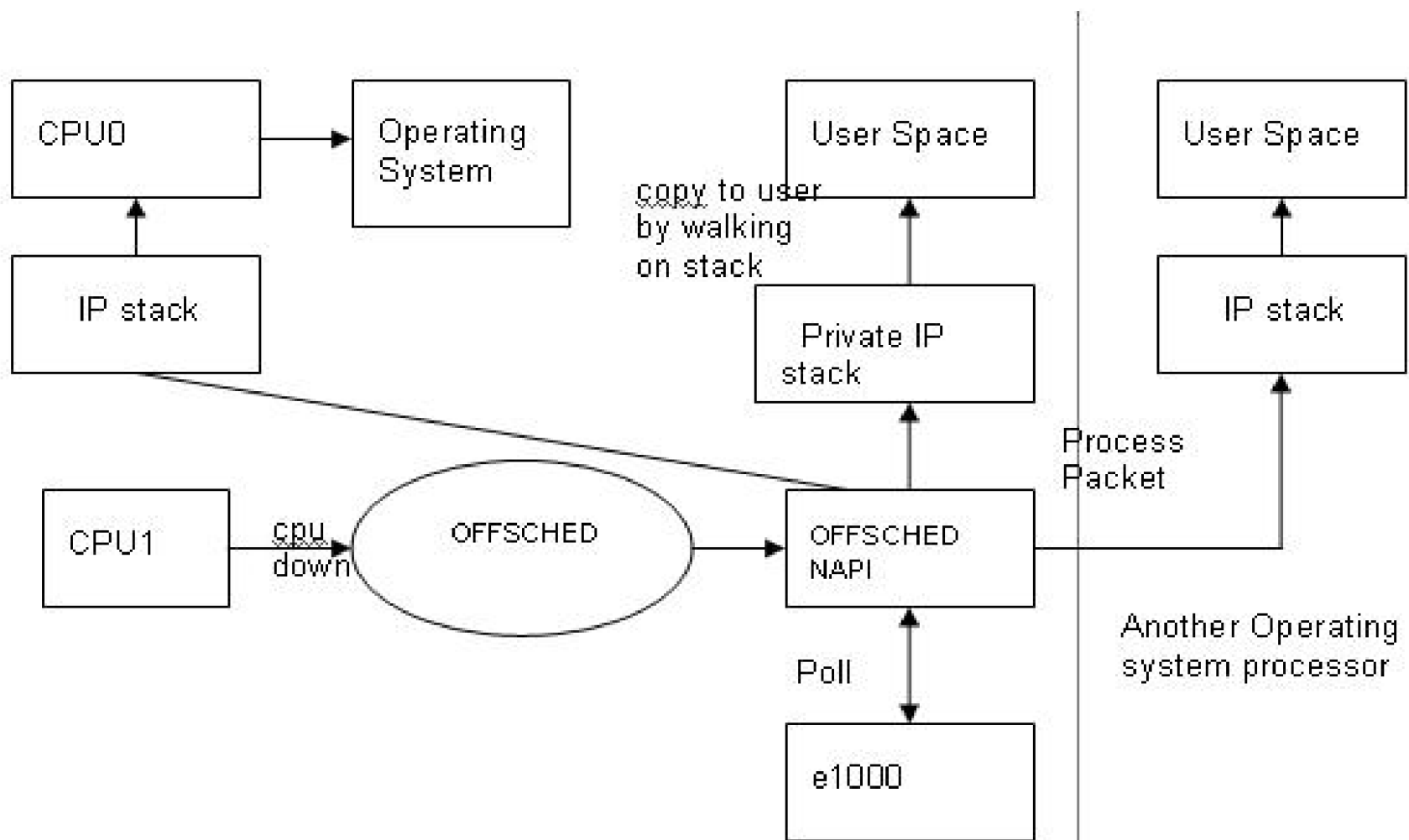
97.68 us, 0.38 sy, 0.00 ni, 4.78 id, 0.00 wa, 0.00 irq, 0.00 st, POLICY

PID	NAME	ST	%CPU	PRIO	CPU	RT	POLICY
8488	malicious	R	97	-100	0	99	2
8858	kded	S	0	20	0	0	0
7834	hald-addon-stor	S	0	20	0	0	0
8878	kicker	S	0	20	0	0	0
7988	Xorg	S	0	20	0	0	0
8868	kdesktop	S	0	20	0	0	0
8192	firefox-bin	S	0	20	0	0	0
5136	screen	S	0	20	0	0	0
8866	kwin	S	0	20	0	0	0
2473	gpm	S	0	20	0	0	0
8338	bash	S	0	20	0	0	0
7813	hald-addon-keyb	S	0	20	0	0	0
9	events/0	S	0	15	0	0	0
8879	kaccess	S	0	20	0	0	0
7671	crond	S	0	20	0	0	0
8863	kurapper	S	0	20	0	0	0
8854	dcopserver	S	0	20	0	0	0
8194	gconfd-2	S	0	20	0	0	0
8281	firefox-bin	S	0	20	0	0	0
8186	run-mozilla.sh	S	0	20	0	0	0
8183	firefox	S	0	20	0	0	0

VMware Tools is not installed in this guest. Choose "Install VMware Tools" from the VM menu.

start CentOS-5-32-... 2.59%, Taine... Screen captu... root - root@1... 20/20 HE Desktop 10:11 PM

## ISOLATION EXAMPLE – OFFLINE NAPI



## ISOLATION EXAMPLE – OFFLINE NAPI

### Pros

1. RX disabling latency
2. IRQ masking latency
3. Rotting Packet
4. SMP IRQ affinity

### Cons

cost a processor

## USES

1. HIGH VOLUME DEVICES
2. DELICATE LATENCY IN KERNEL SPACE
3. INTEL I/OAT DMA LIKE



# Acknowledgments

Jeff Roberson for creating the first implementation of the offline scheduler.

# OFFLINE SCHEDULER BOF

Raz Ben Yehuda  
Linux plumbers conference 2009

## CONCEPT

PARTIAL PARTITIONING vs PURE PARTITIONING

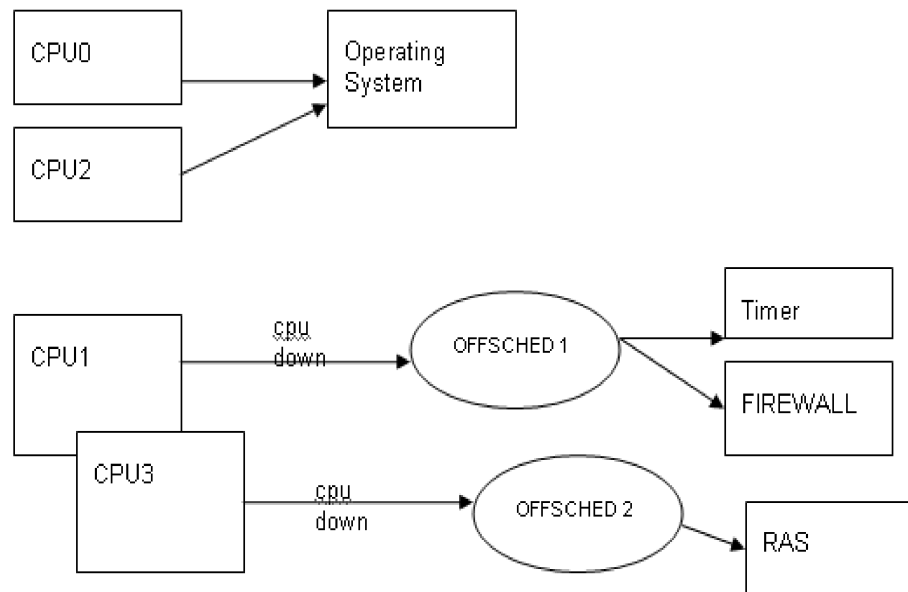
PROCESSOR is a DEVICE vs PROCESSOR is the SYSTEM

## CONS

1. CANNOT RUN USER SPACE TASKS
2. RELEATIVELY HARD TO DEVELOP
3. ONLY FOR SMP/SMT/MC SYSTEMS
4. INDIRECT ACCESS TO VMALLOC AREA
5. NO CONTEXT SWITCH
6. DOES NOT PRESERVE STATES
7. NOT MAINLINE

## PROS

1. PERFORMANCE
2. CONTAINMENT
3. ACCURACY
4. SPEED-UP
5. CONSISTENCY
6. HYBRID SYSTEM
7. ASYMMETRIC PROCESSING
8. INDEPENDENT SYSTEMS
9. BINDING PERIPHERAL TO AN OFFLINE PROCESSOR  
( SMART IO DEVICE)



## OFFLINE REAL TIME

### NMI

- undisturbed ( interrupt-less )
- accurate

### PROGRAM SIZE

- best at many small programs

### ISOLATION

- easier to analyze

### SERIALIZATION

- Full preemption control

### SPEED-UP

- linear – nearly up to BUS contention

### CPU QUIESCE

- No need to walk through a quiesce state

### PATCH OVERHEAD

- Can be used with any Linux kernel that supports processor un-plugging

## OFFLINE REAL TIME - OFFLETS

OFFLET is a context running NMI outside the operating system  
OFFLET can be scheduled to a target processor and in a specific point in time.

```
...  
int cpu_idx=3;  
offlet x;  
....
```

```
offsched_schedule(&x,11,cpu_idx,my_arg);
```

schedule an offlet to run on processor 3 in 11 time units from now.



## The Minimum Patch

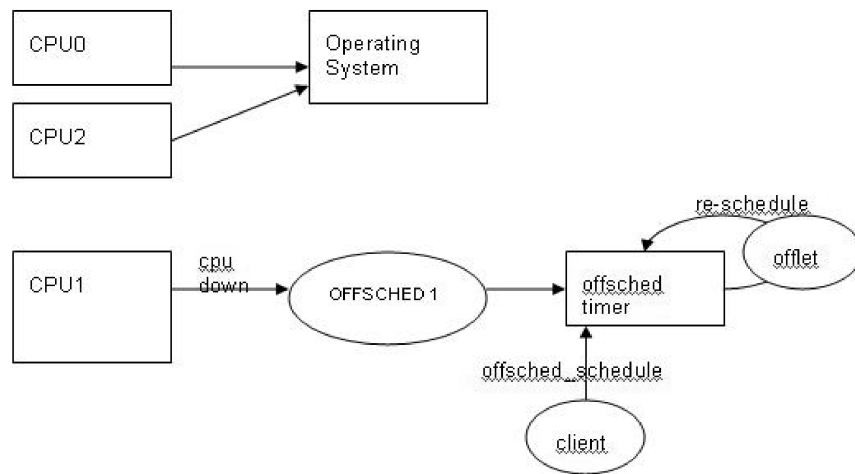
```
#ifdef CONFIG_HOTPLUG_CPU
+void (*hotplug_cpu_dead)(void);
+EXPORT_SYMBOL(hotplug_cpu_dead);
DECLARE_PER_CPU(int, cpu_state);

#include <asm/nmi.h>
__get_cpu_var(cpu_state) = CPU_DEAD;

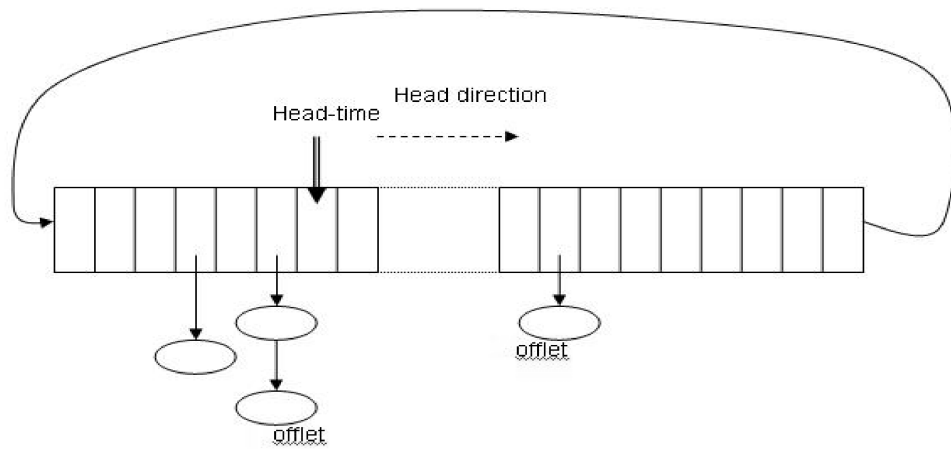
local_irq_disable();
+if (hotplug_cpu_dead)
+hotplug_cpu_dead();
while (1)
    halt();
}
@@ -1265,8 +1265,6 @@
/* They ack this in play_dead by
   setting CPU_DEAD */
if (per_cpu(cpu_state, cpu) == CPU_DEAD) {
-   if (1 == num_online_cpus())
-       alternatives_smp_switch(0);
-       return;
}
msleep(100);
```

This patch does not include the memory allocation/de-allocation and offline napi.

# Real Time example - 1us timer



# Real Time example - 1us timer



## Real Time example - 1us timer pros and cons

### cons

1. no need in low resolution timers
2. use TICKLESS if can.
3. if the transition from the offlet to the application is too complicated.

### pros

1. the timer interrupt might be nested.
2. amount of work is too much to be handled in interrupt context.
3. Timer deadline varies below the defined resolution
4. No HPET available
5. 3% HPET overhead is too much.

## ISOLATION EXAMPLE - RTOP

How do we know what happens in a system when this system is not accessible ?

OFFLINE solution: run a monitoring tool outside the operating system.

## ISOLATION EXAMPLE - RTOP

The screenshot shows a VMware Workstation window titled "CentOS-5-32 - VMware Workstation". Inside the window, a terminal window titled "CentOS-5-32" displays the output of the 'rtop' command. The terminal shows system statistics at the top, followed by a table of processes. The 'malicious' process is highlighted with a mouse cursor.

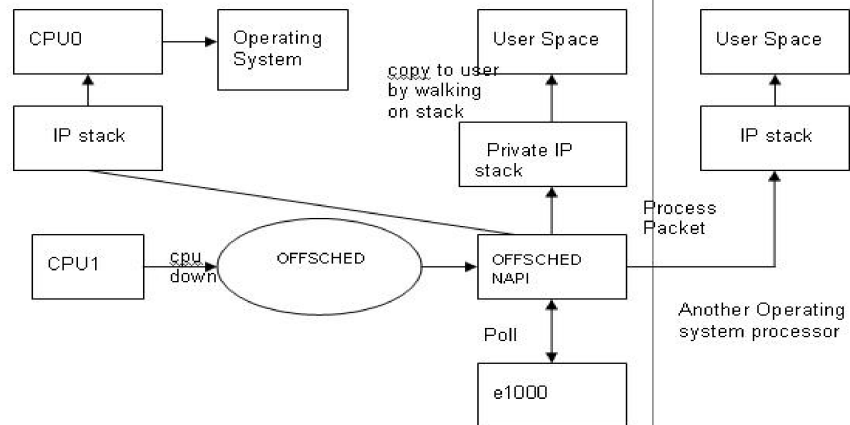
System Statistics:

```
rtop 0.1 #112 processes
97.68 us, 0.38 sy, 0.00 ni, 1.78 id, 0.00 wa, 0.00 irq, 0.00 st
```

PID	NAME	ST	%CPU	PRI	CPU	RT	POLICY
8488	malicious	R	97	-108	0	99	2
8858	kdcd	S	0	20	0	0	0
7834	hald-addon-stor	S	0	20	0	0	0
8878	kicker	S	0	20	0	0	0
7988	Xorg	S	0	20	0	0	0
8868	kdcd	S	0	20	0	0	0
8192	firefox-bin	S	0	20	0	0	0
5136	screen	S	0	20	0	0	0
8866	kuin	S	0	20	0	0	0
2473	gpm	S	0	20	0	0	0
8338	bash	S	0	20	0	0	0
7813	hald-addon-keyb	S	0	20	0	0	0
9	events/0	S	0	15	0	0	0
8879	kaccess	S	0	20	0	0	0
7671	crond	S	0	20	0	0	0
8863	kuapper	S	0	20	0	0	0
8854	dcopserver	S	0	20	0	0	0
8194	gconfd-2	S	0	20	0	0	0
8281	firefox-bin	S	0	20	0	0	0
8186	run-mozilla.sh	S	0	20	0	0	0
8183	firefox	S	0	20	0	0	0

VMware Tools is not installed in this guest. Choose "Install VMware Tools" from the VM menu.

## ISOLATION EXAMPLE – OFFLINE NAPI



## ISOLATION EXAMPLE – OFFLINE NAPI

### Pros

1. RX disabling latency
2. IRQ masking latency
3. Rotting Packet
4. SMP IRQ affinity

### Cons

cost a processor



## USES

1. HIGH VOLUME DEVICES
2. DELICATE LATENCY IN KERNEL SPACE
3. INTEL I/OAT DMA LIKE

## Acknowledgments

Jeff Roberson for creating the first implementation of the offline scheduler.