To do:
Display the state system data in a view
Test

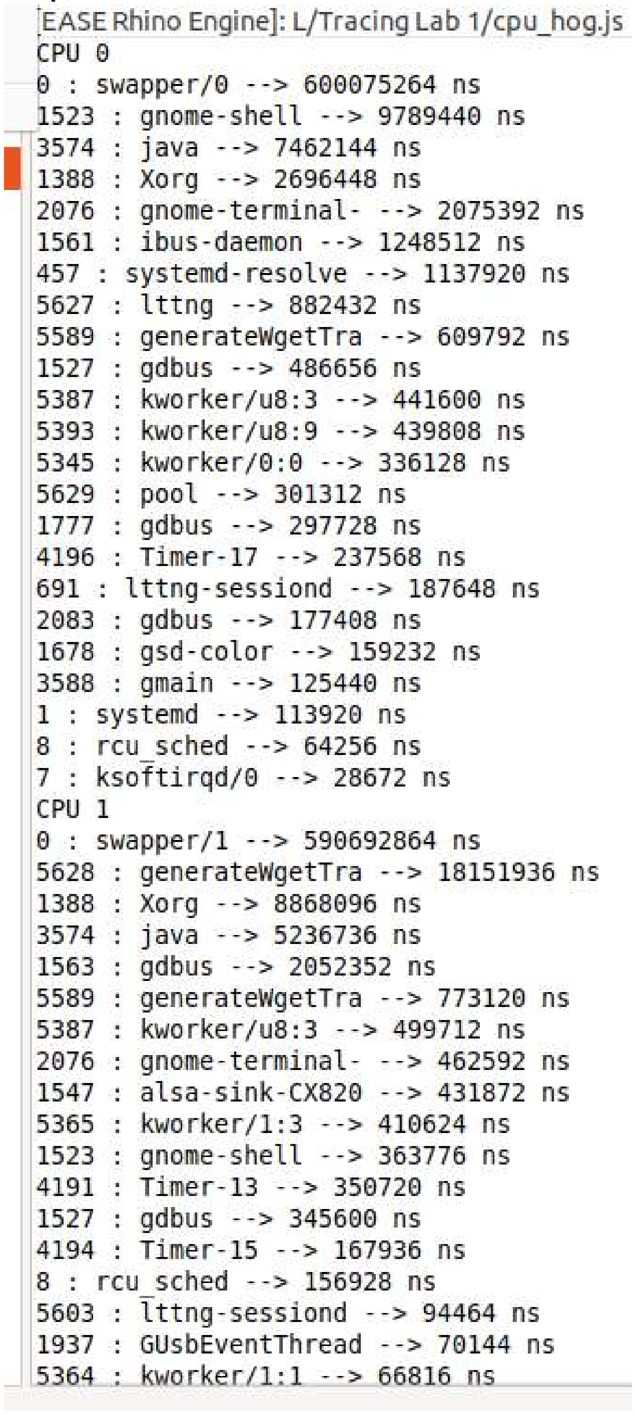
Code:
```
loadModule("/TraceCompass/Trace");
loadModule("/TraceCompass/Analysis");

//get the active trace
var trace = getActiveTrace();
if(trace==null){
	print("No trace is active.");
	exit();
}

//set up the state system
var analysis = createScriptedAnalysis(trace, "cpu_hog_view.js");
var ss = analysis.getStateSystem(false);

//the start and end times for the trace
var startTime = -1;
var endTime = -1;

//this block will create a list that will contain one list for each CPU of the "sched_switch" events
//it also sets the start and end times
var sched_switch_list = [];
var iter = getEventIterator(trace);
var event = null;
while (iter.hasNext()){
	event = iter.next();

	if(startTime==-1) startTime = event.getTimestamp().toNanos();

	var eventName = event.getName();
	var eventCPU = getEventFieldValue(event,"CPU")

	if(eventName=="sched_switch"){
		//create a new CPU list if this is the first event for that CPU
		if(sched_switch_list[eventCPU]==null){
			sched_switch_list[eventCPU] = [];
			sched_switch_list[eventCPU][0] = event;

		//otherwise add the event to the end of the existing CPU list
		}else{
			sched_switch_list[eventCPU][sched_switch_list[eventCPU].length] = event;
		}
	}
}
```

```
endTime = event.getTimestamp().toNanos();

//this block calculates, for each CPU, the time from the 'i'th sched_switch event to the 'i+1'th and
matches that time with the corresponding thread id
var thread_list = [];
for(i=0; i<sched_switch_list.length; i++){
        var new_list = [];
        var prev = startTime;

        for(j=0; j<=sched_switch_list[i].length; j++){
                var new_entry;

                if(j==sched_switch_list[i].length){
                        new_entry = {
                                tid: getEventFieldValue(sched_switch_list[i][j-1], "next_tid"),
                                name: getEventFieldValue(sched_switch_list[i][j-1], "next_comm"),
                                start: prev,
                                end: endTime
                        }
                }else{
                        new_entry = {
                                tid: getEventFieldValue(sched_switch_list[i][j], "prev_tid"),
                                name: getEventFieldValue(sched_switch_list[i][j], "prev_comm"),
                                start: prev,
                                end: sched_switch_list[i][j].getTimestamp().toNanos()
                        }
                }

                prev = new_entry.end;
                new_list[j] = new_entry;
        }

        thread_list[i] = new_list;
}

//this block creates a new list that will hold the total duration on the CPU for each thread
var duration_list = [];
for(i = 0; i < thread_list.length; i++){
        var new_list = [];
        var p = 0;

        for(j=0; j<thread_list[i].length; j++){
                var exists = false;
                for(k=0; k<new_list.length; k++){
                        //if the thread is already in the new list, add the additional duration to the
existing duration
                        if(thread_list[i][j].tid == new_list[k].tid){
```

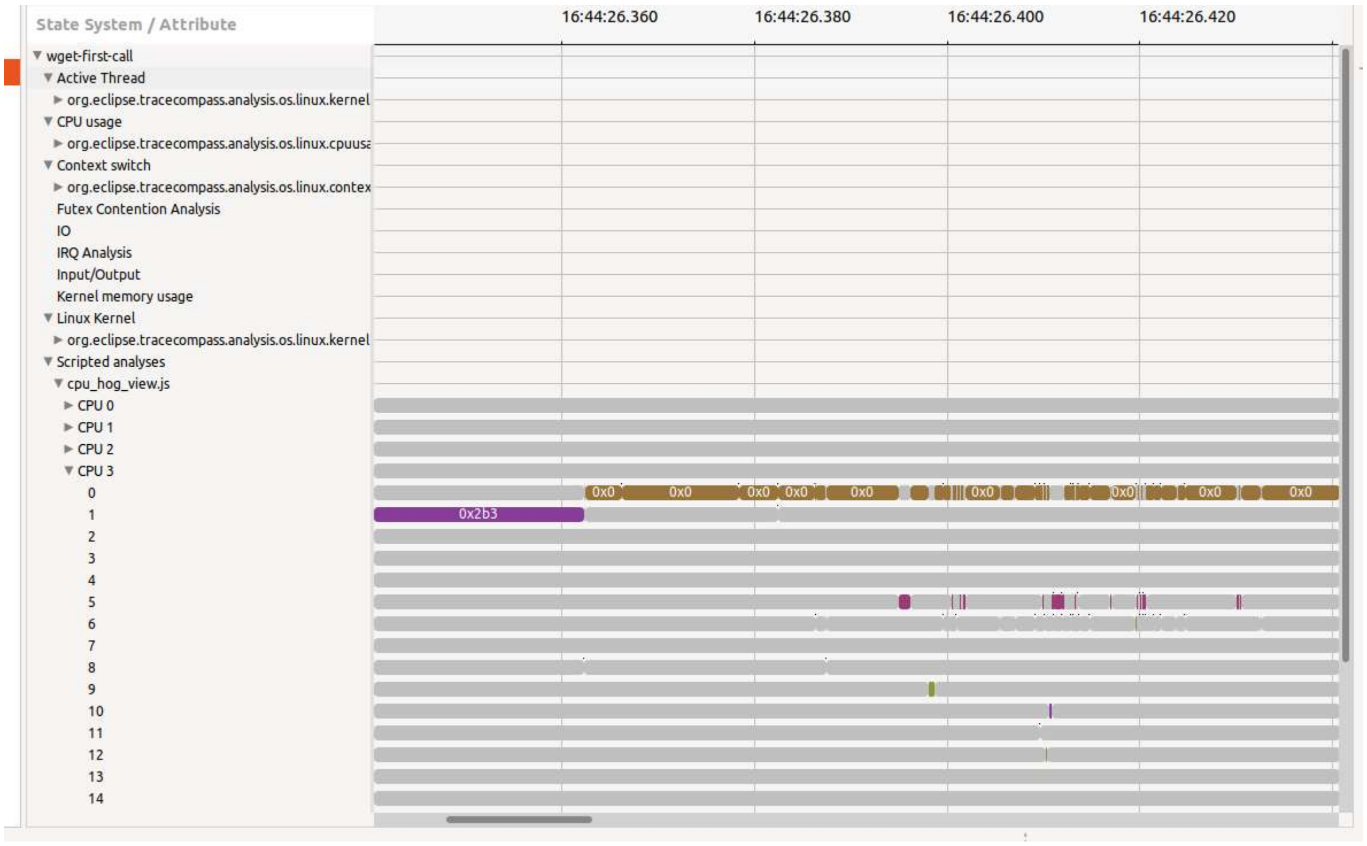
```
					new_list[k].duration = new_list[k].duration + (thread_list[i][j].end -
thread_list[i][j].start);
					exists = true;
				}
			}

			//if the thread is not yet represented in the new list, add it
			if(!exists){
				var new_entry = {
					tid: thread_list[i][j].tid,
					name: thread_list[i][j].name,
					duration: thread_list[i][j].end - thread_list[i][j].start
				};
				new_list[p] = new_entry;
				p++;
			}
		}

		duration_list[i] = new_list;
}

//sort the entries by duration: highest to lowest
for(i = 0; i < duration_list.length; i++){
		duration_list[i].sort(function(a,b){return b.duration - a.duration});
		printCPU(i,duration_list[i]);
}

//this block saves the attributes to the state system
for(i = 0; i < duration_list.length; i++){
		for(j = 0; j < duration_list[i].length; j++){
			quark = ss.getQuarkAbsoluteAndAdd("CPU " + i, j);
			for(k = 0; k < thread_list[i].length; k++){
				if(thread_list[i][k].tid==duration_list[i][j].tid){
					ss.modifyAttribute(thread_list[i][k].start, thread_list[i][k].tid, quark);
					ss.removeAttribute(thread_list[i][k].end, quark);
				}
			}
		}
}

//done
ss.closeHistory(endTime);
print("Done");

//this function prints the data to the console
function printCPU(number, threads){
		print("CPU " + number);
```

```
        for(num_threads = 0; num_threads < threads.length; num_threads++){
                print(threads[num_threads].tid + " : " + threads[num_threads].name + " --> " +
threads[num_threads].duration + " ns");
        }
}
```

Output:



```
[EASE Rhino Engine]: L/Tracing Lab 1/cpu_hog.js
CPU 0
0 : swapper/0 --> 600075264 ns
1523 : gnome-shell --> 9789440 ns
3574 : java --> 7462144 ns
1388 : Xorg --> 2696448 ns
2076 : gnome-terminal- --> 2075392 ns
1561 : ibus-daemon --> 1248512 ns
457 : systemd-resolve --> 1137920 ns
5627 : lttng --> 882432 ns
5589 : generateWgetTra --> 609792 ns
1527 : gdbus --> 486656 ns
5387 : kworker/u8:3 --> 441600 ns
5393 : kworker/u8:9 --> 439808 ns
5345 : kworker/0:0 --> 336128 ns
5629 : pool --> 301312 ns
1777 : gdbus --> 297728 ns
4196 : Timer-17 --> 237568 ns
691 : lttng-sessiond --> 187648 ns
2083 : gdbus --> 177408 ns
1678 : gsd-color --> 159232 ns
3588 : gmain --> 125440 ns
1 : systemd --> 113920 ns
8 : rcu_sched --> 64256 ns
7 : ksoftirqd/0 --> 28672 ns
CPU 1
0 : swapper/1 --> 590692864 ns
5628 : generateWgetTra --> 18151936 ns
1388 : Xorg --> 8868096 ns
3574 : java --> 5236736 ns
1563 : gdbus --> 2052352 ns
5589 : generateWgetTra --> 773120 ns
5387 : kworker/u8:3 --> 499712 ns
2076 : gnome-terminal- --> 462592 ns
1547 : alsa-sink-CX820 --> 431872 ns
5365 : kworker/1:3 --> 410624 ns
1523 : gnome-shell --> 363776 ns
4191 : Timer-13 --> 350720 ns
1527 : gdbus --> 345600 ns
4194 : Timer-15 --> 167936 ns
8 : rcu_sched --> 156928 ns
5603 : lttng-sessiond --> 94464 ns
1937 : GUsbEventThread --> 70144 ns
5364 : kworker/1:1 --> 66816 ns
```


State System / Attribute
16:44:26.360
16:44:26.380
16:44:26.400
16:44:26.420
▼ wget-first-call
▼ Active Thread
► org.eclipse.tracecompass.analysis.os.linux.kernel
▼ CPU usage
► org.eclipse.tracecompass.analysis.os.linux.cpuusa
▼ Context switch
► org.eclipse.tracecompass.analysis.os.linux.contex
Futex Contention Analysis
IO
IRQ Analysis
Input/Output
Kernel memory usage
▼ Linux Kernel
► org.eclipse.tracecompass.analysis.os.linux.kernel
▼ Scripted analyses
▼ cpu_hog_view.js
► CPU 0
► CPU 1
► CPU 2
▼ CPU 3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0x0
0x2b3