Code:

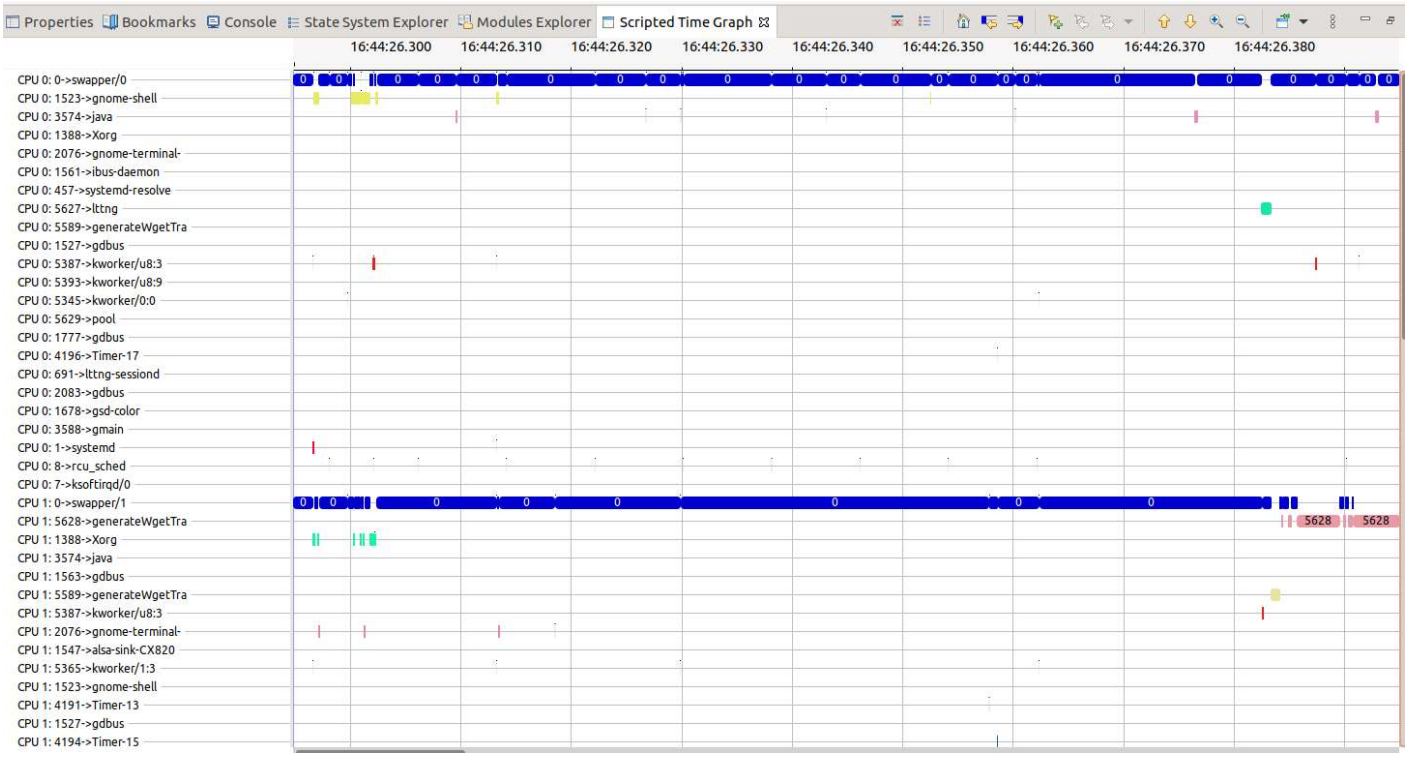
```
loadModule("/TraceCompass/Trace");
loadModule("/TraceCompass/Analysis");
loadModule("/TraceCompass/DataProvider");
loadModule("/TraceCompass/View");
loadModule('/TraceCompass/Utils');

//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");
if(analysis==null){
	print("The analysis could not be created.");
	exit();
}
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, duration_list[i][j].tid, quark);
                                ss.removeAttribute(thread_list[i][k].end, quark);
                        }
                }
        }
}

ss.closeHistory(endTime);

//this block sets up the time graph provider for the time graph view by creating an entries list from the
state system
var entries = createListWrapper();
```

```
for(i = 0; i <duration_list.length; i++){
	quarks = ss.getQuarks("CPU "+i,"*");
	for (j = 0; j < quarks.size(); j++) {
		quark = quarks.get(j);
		entry_name = "CPU " + i + ": " + duration_list[i][j].tid + "->" + duration_list[i][j].name;
		entry = createEntry(entry_name, {'quark' : quark});
		entries.getList().add(entry);
	}
}

//the function used to get the entries for the provider
function getEntries(parameters) {
	return entries.getList();
}

//create the time graph provider and view
provider = createScriptedTimeGraphProvider(analysis, getEntries, null, null);
if (provider != null) {
	openTimeGraphView(provider);
}

//Script finished.
//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:


Properties
Bookmarks
Console
State System Explorer
Modules Explorer
Scripted Time Graph
16:44:26.300
16:44:26.310
16:44:26.320
16:44:26.330
16:44:26.340
16:44:26.350
16:44:26.360
16:44:26.370
16:44:26.380
CPU 0: 0->swapper/0
CPU 0: 1523->gnome-shell
CPU 0: 3574->java
CPU 0: 1388->Xorg
CPU 0: 2076->gnome-terminal-
CPU 0: 1561->ibus-daemon
CPU 0: 457->systemd-resolve
CPU 0: 5627->lttng
CPU 0: 5589->generateWgetTra
CPU 0: 1527->gdbus
CPU 0: 5387->kworker/u8:3
CPU 0: 5393->kworker/u8:9
CPU 0: 5345->kworker/0:0
CPU 0: 5629->pool
CPU 0: 1777->gdbus
CPU 0: 4196->Timer-17
CPU 0: 691->lttng-sessiond
CPU 0: 2083->gdbus
CPU 0: 1678->gsd-color
CPU 0: 3588->gmain
CPU 0: 1->systemd
CPU 0: 8->rcu_sched
CPU 0: 7->ksoftirqd/0
CPU 1: 0->swapper/1
CPU 1: 5628->generateWgetTra
CPU 1: 1388->Xorg
CPU 1: 3574->java
CPU 1: 1563->gdbus
CPU 1: 5589->generateWgetTra
CPU 1: 5387->kworker/u8:3
CPU 1: 2076->gnome-terminal-
CPU 1: 1547->alsa-sink-CX820
CPU 1: 5365->kworker/1:3
CPU 1: 1523->gnome-shell
CPU 1: 4191->Timer-13
CPU 1: 1527->gdbus
CPU 1: 4194->Timer-15
5628
5628