Getting to know Eclipse/CTF

Matthew Khouzam

matthew.khouzam@{polymtl.ca|ericsson.com|gmail.com}

What will we see today

- About me
- A quick overview of LTTng 2.0 / CTF if necessary
- What does Eclipse bring to the party
- Getting Eclipse and Linux Tools
- How to use the java based CTF parser natively (with code!)
- How to use the java based CTF parser with the TMF API (with code!)
- How to create a view in TMF to show raw CTF information (with code!)
- How to read the TMF state systems (with more code!)

LTTng 2.0

- Objectively the single greatest piece of software since Lotus 123, according to some
- Low impact and secure tracer
- Free and open (you can poke its insides)
- Uses common trace format to store traces

Common Trace Format (CTF)

- Self defining file format
- Fast to write
- Efficient storage
- Not all that obvious to read

http://wiki.teamfortress.com/wiki/File:Gamemode_ctf.png

Eclipse TMF

- An easy to use framework for developing new earth shattering algorithms.
- Allows users to not worry about the back-end. (Allows you to do research instead of boilerplate code)
- Pretty

CTF plugin

- Java, can run in Linux, BSD*, Windows, Mac OS*, QNX*, ...
- Made with Antlr parser
- Does not require TMF
- 7+ KLoc, tested, over a year's worth of development. You don't need to reinvent the wheel.

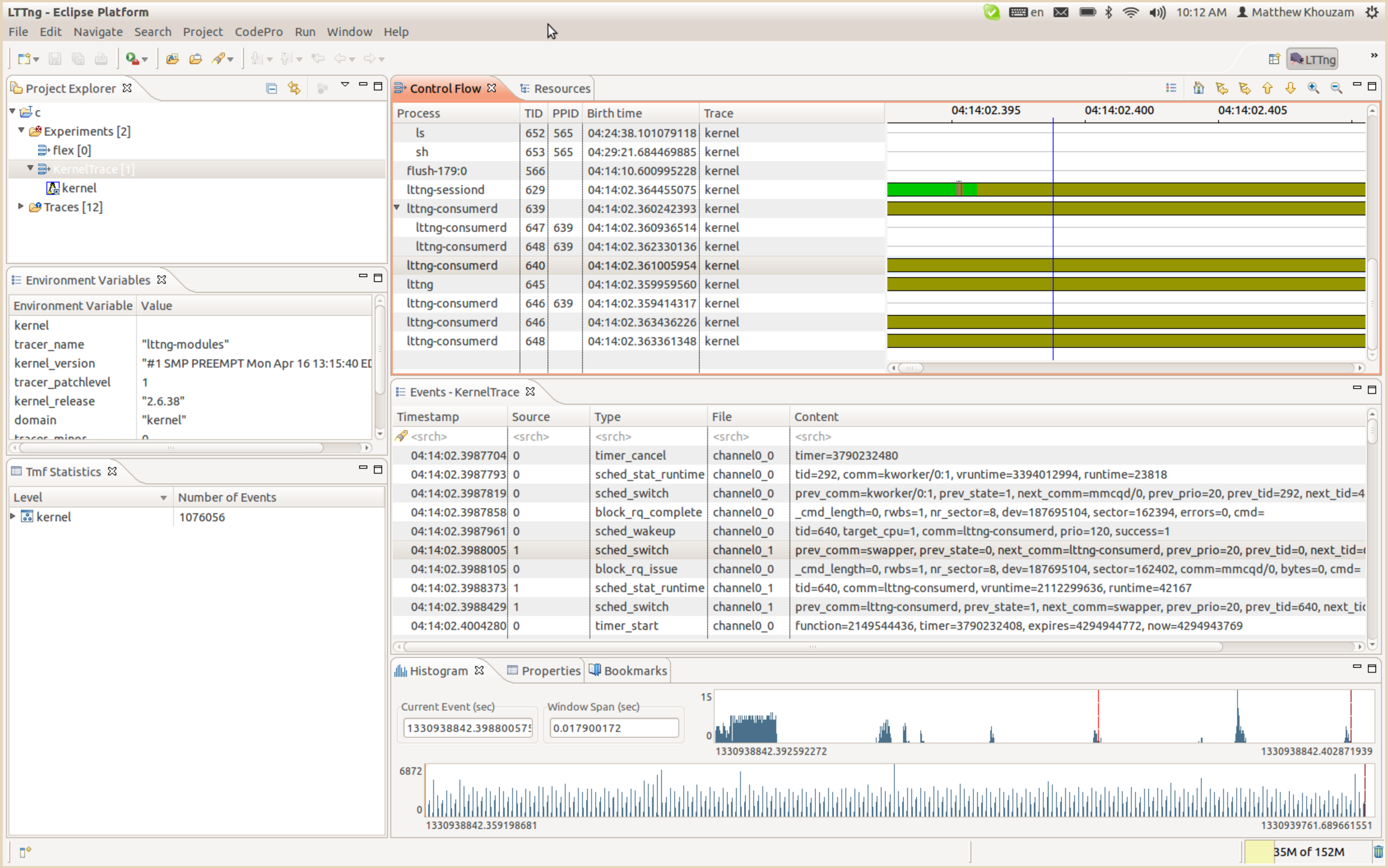
*Not tested

Generic State System

- Persistent on storage
- Generic. You can make a state system for your application, not just the Linux kernel* (Apache anyone?)
- Easy to access data
- Views can access the intervals directly at a pixel perfect resolution

*We still support the Linux Kernel and it is shipped in TMF.

Eclipse TMF View



Process	TID	PPID	Birth time	Trace
ls	652	565	04:24:38.101079118	kernel
sh	653	565	04:29:21.684469885	kernel
flush-179:0	566		04:14:10.600995228	kernel
lttng-sessiond	629		04:14:02.364455075	kernel
lttng-consumerd	639		04:14:02.360242393	kernel
lttng-consumerd	647	639	04:14:02.360936514	kernel
lttng-consumerd	648	639	04:14:02.362330136	kernel
lttng-consumerd	640		04:14:02.361005954	kernel
lttng	645		04:14:02.359959560	kernel
lttng-consumerd	646	639	04:14:02.359414317	kernel
lttng-consumerd	646		04:14:02.363436226	kernel
lttng-consumerd	648		04:14:02.363361348	kernel

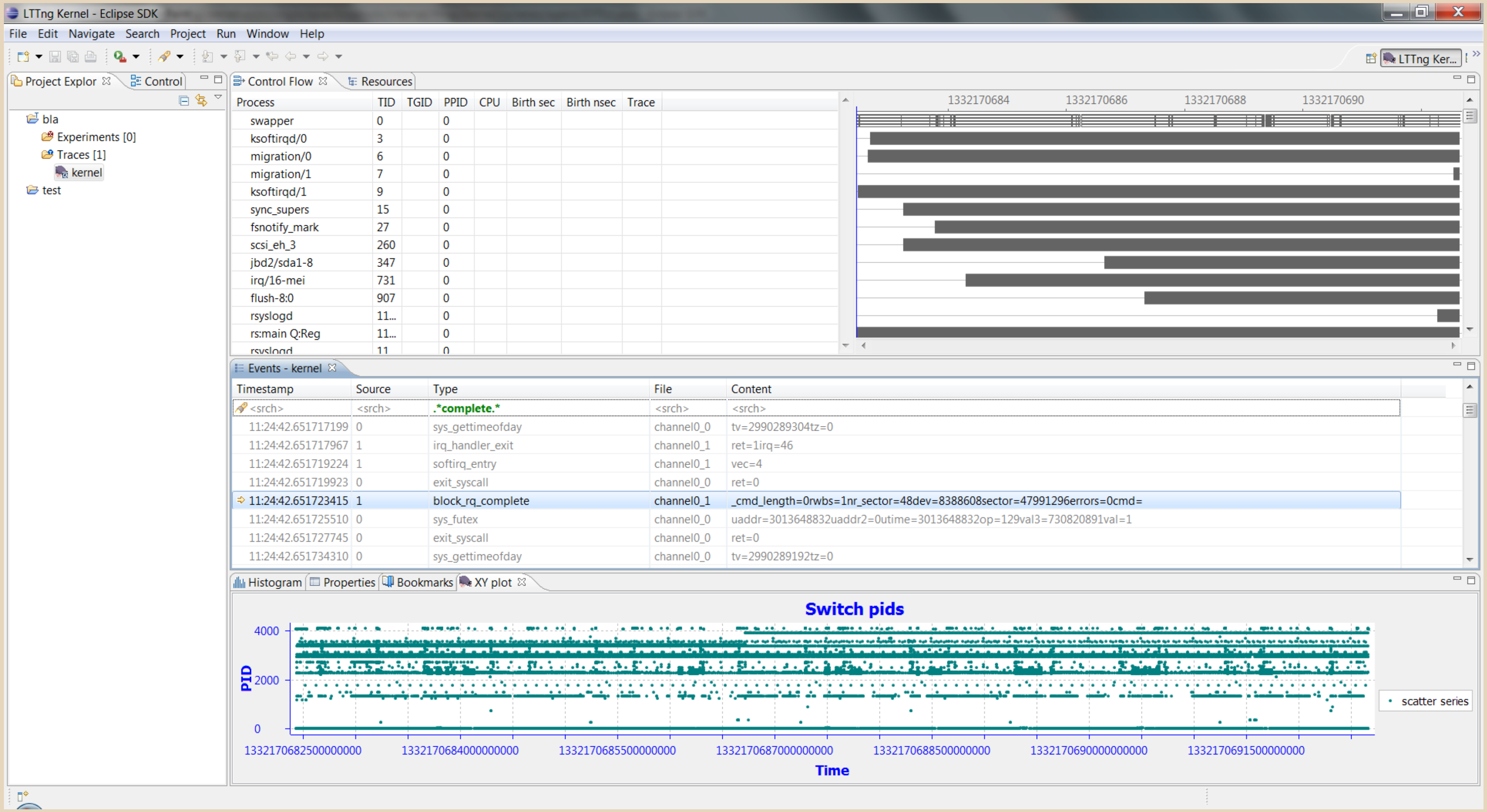
04:14:02.395 04:14:02.400 04:14:02.405

Environment Variable	Value
kernel	
tracer_name	"lttng-modules"
kernel_version	"#1 SMP PREEMPT Mon Apr 16 13:15:40 ED
tracer_patchlevel	1
kernel_release	"2.6.38"
domain	"kernel"
tracer_minor	0

Level	Number of Events
kernel	1076056

Timestamp	Source	Type	File	Content
<srch>	<srch>	<srch>	<srch>	<srch>
04:14:02.3987704	0	timer_cancel	channel0_0	timer=3790232480
04:14:02.3987793	0	sched_stat_runtime	channel0_0	tid=292, comm=kworker/0:1, vruntime=3394012994, runtime=23818
04:14:02.3987819	0	sched_switch	channel0_0	prev_comm=kworker/0:1, prev_state=1, next_comm=mmcqd/0, prev_prio=20, prev_tid=292, next_tid=4
04:14:02.3987858	0	block_rq_complete	channel0_0	_cmd_length=0, rwbs=1, nr_sector=8, dev=187695104, sector=162394, errors=0, cmd=
04:14:02.3987961	0	sched_wakeup	channel0_0	tid=640, target_cpu=1, comm=lttng-consumerd, prio=120, success=1
04:14:02.3988005	1	sched_switch	channel0_1	prev_comm=swapper, prev_state=0, next_comm=lttng-consumerd, prev_prio=20, prev_tid=0, next_tid=
04:14:02.3988105	0	block_rq_issue	channel0_0	_cmd_length=0, rwbs=1, nr_sector=8, dev=187695104, sector=162402, comm=mmcqd/0, bytes=0, cmd=
04:14:02.3988373	1	sched_stat_runtime	channel0_1	tid=640, comm=lttng-consumerd, vruntime=2112299636, runtime=42167
04:14:02.3988429	1	sched_switch	channel0_1	prev_comm=lttng-consumerd, prev_state=1, next_comm=swapper, prev_prio=20, prev_tid=640, next_tid
04:14:02.4004280	0	timer_start	channel0_0	function=2149544436, timer=3790232408, expires=4294944772, now=4294943769

Eclipse TMF views (Prototype!?!)



Process	TID	TGID	PPID	CPU	Birth sec	Birth nsec	Trace
swapper	0		0				
ksoftirqd/0	3		0				
migration/0	6		0				
migration/1	7		0				
ksoftirqd/1	9		0				
sync_supers	15		0				
fsnotify_mark	27		0				
scsi_eh_3	260		0				
jbd2/sda1-8	347		0				
irq/16-mei	731		0				
flush-8:0	907		0				
rsyslogd	11...		0				
rs:main Q:Reg	11...		0				
rsyslogd	11		0				

Timestamp	Source	Type	File	Content
<srch>	<srch>	.*complete.*	<srch>	<srch>
11:24:42.651717199	0	sys_gettimeofday	channel0_0	tv=2990289304tz=0
11:24:42.651717967	1	irq_handler_exit	channel0_1	ret=1irq=46
11:24:42.651719224	1	softirq_entry	channel0_1	vec=4
11:24:42.651719923	0	exit_syscall	channel0_0	ret=0
11:24:42.651723415	1	block_rq_complete	channel0_1	_cmd_length=0rwbs=1nr_sector=48dev=8388608sector=47991296errors=0cmd=
11:24:42.651725510	0	sys_futex	channel0_0	uaddr=3013648832uaddr2=0utime=3013648832op=129val3=730820891val=1
11:24:42.651727745	0	exit_syscall	channel0_0	ret=0
11:24:42.651734310	0	sys_gettimeofday	channel0_0	tv=2990289192tz=0

Getting Eclipse and Linux Tools

www.eclipse.org
To develop plugins get the git.

Clone
git://git.eclipse.org/gitroot/linuxtools/org.eclipse.linuxtools.git
ssh://git.eclipse.org/gitroot/linuxtools/org.eclipse.linuxtools.git
http://git.eclipse.org/gitroot/linuxtools/org.eclipse.linuxtools.git

CTF Parser

/lttng/org.eclipse.linuxtools.ctf.core.tests/src/org/eclipse/linuxtools/ctf/core/tests/headless/ReadTrace.java

```
public class ReadTrace {
    public static void main(String[] args) {
        CTFTrace trace = null;
        try {
           trace = new CTFTrace("tracedir");
        } catch (CTFReaderException e) {
          return;
        }
        System.out.println("Event, " + " Time, " + " type, " + " CPU ");
        CTFTraceReader traceReader = new CTFTraceReader(trace);
        while (traceReader.hasMoreEvents()) {
          EventDefinition ed = traceReader.getCurrentEventDef();
          System.out.println(traceReader.getIndex() + ", "
                    + ed.timestamp + trace.getOffset() + ", "
                 + ed.getDeclaration().getName()
                    + ", " + ed.getCPU());
          traceReader.advance();
        }
    }
}
```

CTF TMF adapter

- Much easier to use
- Requires TMF

CTF Tmf Adapter Code!

/lttng/org.eclipse.linuxtools.tmf.core.tests/src/org/eclipse/linuxtools/tmf/core/tests/ctfadaptor/headless/Benchmark.java

```
public class ReadTrace {
    public static void main(String[] args) {
        CTFTmfTrace trace = new CtfTmfTrace();
        try {
          trace.initTrace(null, "tracedir", CtfTmfEvent.class);
        } catch (CTFReaderException e) {
          return;
        }
        System.out.println("Event, " + " Time, " + " type, " + " CPU ");
        final CtfIterator traceReader = (CtfIterator) trace.seekEvent(0);
     CtfTmfEvent current = traceReader.getCurrentEvent();
        while (current != null) {
           System.out.println("Event " + traceReader.getRank()
                                + " Time " + current.getTimestamp()
                                + " type " + current.getType()
                                + " on CPU " + current.getCPU());
           traceReader.advance();
           current = traceReader.getCurrentEvent();
        }
    }
}
```

Making a TMF view

A picture is worth 1024 words...
yet it fits into 2" by 2" in a publication.

We will make a table to display time deltas between the first 8 events.

MORE CODE!!! - Boiler plate (1/3)

```
public class TmfDeltaView extends TmfView {
    public static final String ID = "org.eclipse.linuxtools.tmf.ui.views.delta"; //$NON-
NLS-1$
    private TmfExperiment<?> fExperiment;
    private Table fTable;
    final private String fTitlePrefix;
    private Composite fParent;

    public TmfDeltaView() {
        super("Deltas"); //$NON-NLS-1$
        fTitlePrefix = getTitle();
    }


    @Override
    public void setFocus() {
        fTable.setFocus();
    }

    @Override
    public void dispose() {
        if (fTable != null) {
            fTable.dispose();
        }
        super.dispose();
    }
}
```

Even more code - UI stuff (2/3)

```
@Override
@SuppressWarnings("unchecked")
public void createPartControl(Composite parent) {
    fParent = parent;
    TableItem ti[];
    // If an experiment is already selected, update the table
    TmfExperiment<ITmfEvent> experiment = (TmfExperiment<ITmfEvent>)
            TmfExperiment.getCurrentExperiment();
    if (experiment == null) return;
    fTable = new Table(parent, SWT.BORDER|SWT.FILL);
    CtfTmfTrace ctfTrace;
    for (ITmfTrace trace : experiment.getTraces()) {
       if (trace instanceof CtfTmfTrace) {
           ctfTrace = (CtfTmfTrace) trace;
    }  }
    CTFIterator iter = ctfTrace.seek(0);
    long prevTS = 0;
    fTable.setItemCount(8);
    ti = fTable.getItems();
    for(int i = 0; i < 8; i++){
        ti[i].setText(iter.getCurrentEvent.getTimestamp() - prevTS);
        prevTS = iter.getCurrentEvent.getTimestamp();
        iter.advance();
    }
    fTable.setHeaderVisible(true);
    fTable.pack();
    parent.layout();
}
```

Zounds! Code! - Signal Handler (3/3)

```
@SuppressWarnings("unchecked")
@TmfSignalHandler
public void experimentSelected(TmfExperimentSelectedSignal<TmfEvent> signal) {
    // Update the trace reference
    TmfExperiment<TmfEvent> exp = (TmfExperiment<TmfEvent>) signal.getExperiment();
    if (!exp.equals(fExperiment)) {
        fExperiment = exp;
        if (fTable != null) {
            fTable.dispose();
        }
        createPartControl( fParent );
        fParent.layout();
    }
}
```

Easy as cake!

CTF using the State History Tree(1/2)

```
public static void main(String[] args) {

IStateSystemBuilder ss;
IStateChangeInput input;
File newStateFile;
IStateHistoryBackend backend;
HistoryBuilder builder;

try {
     // Read a trace and build the state system
     input = new CtfKernelStateInput(CtfTestFiles.getTestTrace());
     newStateFile = new File("testHistory.ht");
     backend = new HistoryTreeBackend(newStateFile, input.getStartTime());
     builder = new HistoryBuilder(input, backend);
} catch (Exception e) {
     e.printStackTrace();
     return;
}
builder.run();
ss = builder.getStateSystemBuilder();
builder.close(); // Waits for the construction to finish

requestExample();
}
```

CTF using the State History Tree(2/2)

```
public static void requestExample(IStateSystemBuilder ssb) {
     try {
          // Request the current thread executing on each CPU
          List<Integer> currentThreadByCPUS =
                         ssb.getQuarks(Attributes.CPUS, "*", Attributes.CURRENT_THREAD);
          for (Integer pid : currentThreadByCPUS) {
          List<ITmfStateInterval> stateIntervals =
          ssb.queryHistoryRange(pid, ssb.getStartTime(),ssb.getCurrentEndTime());
          // Output formatting
          String output = "Attribute :" +ssb.getFullAttributePath(currentThread)+"\n";
          for (ITmfStateInterval stateInterval : stateIntervals) {
               // Print the interval "[begin,end]"
               output += "[" + String.valueOf(stateInterval.getStartTime());
               output += "," + String.valueOf(stateInterval.getEndTime()) + "]";
               // Print the attribute value
               output += " = " + (stateInterval.getStateValue().unboxInt()) + "\n";
          }
          System.out.println(output);
          }
     } catch (Exception e) {
          e.printStackTrace();
          return;
     }
   }
}
```

Thank you!

Questions?

Demo?

Questions about demos?

Demos about questions?