Implementation of Real-Time Java using KURT

Dinesh Selvarajan

Project Defense
Master of Science (Computer Engineering)
University of Kansas
Oct 21, 2003

Committee
Dr. Jerry James (Chair)
Dr. Douglas Niehaus
Dr. John Gauch

University of Kansas
Talk Content

- Need for Real-Time Java (RTJ)
- Implementation
  - RTJ library
  - Driving example
- Evaluation
- Conclusion
- Future Work
Need for RTJ

- Enable Java for real-time systems
- What is real-time?
- Proven predictable behavior
- Useful in military, aerospace and commercial process control systems
- Java does not suit time-aware systems
- Real-Time Specification for Java (RTSJ) – from Real-Time Java Expert Group
Basic Differences – RT & non-RT

- High resolution timers (micro- and nanosecond precision)
- Scheduling of events is guaranteed to take place at the exact time specified.
Framework

- JVM 1.4.1
- Real-time Operating System (RTOS) – KURT Linux
- RTJ Library
- Driving example
Areas of focus

- Under the `javax.realtime` package,
  - Real-time Clocks
    - `HighResolutionTime`
    - `AbsoluteTime`
    - `RelativeTime`
    - `RationalTime`
  - Real-time Timers
    - `OneShotTimer`
    - `PeriodicTimer`
  - Real-time Threads
    - `RealtimeThread`
  - Real-time Scheduling
    - `Scheduler`
- Communicate with the RTOS (KURT) through native calls linked by Java Native Interface (JNI)
Bouncing Ball
RT threads & Scheduling

- Native calls – directly interact with KURT
- Using JNI
- Make threads real-time
- Schedule those threads
  - Explicit Plan Schedule
  - Dynamic Scheduling
Data Streams with JNI

- Timing and schedule of the threads need to be checked using high-precision timers
- DSUI – tool to collect instrumentation data during execution
- DSUI made possible using JNI
- Instrumentation points – family, events and counters (contained in namespace file)
  - APPLICATION (family) 5
  - EVENT_OPEN 5
  - EVENT_START 4
  - EVENT_SUBMIT_SCHED 3
  - EVENT_SUSPEND 2
  - EVENT_WAKEUP 1
  - EVENT_EXIT 0
Customized JVM

- JVM spawns nine threads
- System threads that are non-deterministic
  - Garbage Collector (GC)
    - Finalizer thread
    - Reference Handler thread
  - Signal Dispatcher thread
  - Compile Thread
- Needs to be removed for testing
Results

testbed55 [6] # java Bounce2 10
kurtdev is: 20
Real-time ID# (as assigned by KURT) are:
ball1: 255
ball2: 254
ball3: 253
ball4: 252
ball5: 251
ball6: 250
ball7: 249
ball8: 248
ball9: 247
Results (cont…)

DSUI Output Log (xml format):

...<ENTITY number="1" time_stamp="6791678943066" tag="0" type="Event">
  <EVENT name="EVENT_OPEN" family="APPLICATION" id="69" />
</ENTITY>

<ENTITY number="2" time_stamp="6791679225486" tag="0" type="Counter">
  <COUNTER name="COUNTER_BOUNCE_SPEED_CONST" family="APPLICATION" id="64"
    count="10" first_updatetime="6791679167887"
    last_updatetime="6791679167887"/>
</ENTITY>

<ENTITY number="3" time_stamp="6838384115559" tag="0" type="Event">
  <EVENT name="EVENT_START" family="APPLICATION" id="68" />
</ENTITY>

<ENTITY number="4" time_stamp="6838384832808" tag="0" type="Event">
  <EVENT name="EVENT_SUBMIT_SCHED" family="APPLICATION" id="67" />
</ENTITY>

<ENTITY number="5" time_stamp="6838384854671" tag="0" type="Event">
  <EVENT name="EVENT_SUSPEND" family="APPLICATION" id="66" />
</ENTITY>

<ENTITY number="6" time_stamp="6838399036229" tag="0" type="Event">
  <EVENT name="EVENT_WAKEUP" family="APPLICATION" id="65" />
</ENTITY>

...
Results (cont…)

Status of the running real-time threads:

- **kurt_status** program provided by KURT – lists all the current real-time processes and their status:

```
... handled late dropped invalid wdog baddog1 baddog2 baddog3
1316  504    0    0    0    0    0    0
rt_id  pid   woken missed rtSusp aborts nonrtSusp switches
 247 10890    3    0    4    0    0    0    0
 248 10889   19    0   20    0    0    0    0
 249 10888   30    0   31    0    0    0    0
 250 10887   47    0   48    0    0    0    0
 251 10886   63    0   64    0    0    0    0
 252 10885   98    0   99    0    0    0    0
 253 10884  151    0  152    0    0    0    0
 254 10883  195    0  196    1    0    0    0
 255 10882  416    0  417    0    0    0    0
...```
Results (cont…)

Scheduling of events – timing information, based on timestamp counters:
(Processor speed = 1399.380 MHz; which means 1399380000 cycles per second)

<table>
<thead>
<tr>
<th>Timestamp counter of the wakeup events</th>
<th>Diff. bet. 2 wakeup events</th>
<th>Diff. in milliseconds = (diff./cycles per second)* 1000</th>
<th>Deviation (in ms) from the expected 10ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>6838399036229</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6838413029334</td>
<td>13993105</td>
<td>9.9995</td>
<td>0.0004</td>
</tr>
<tr>
<td>6838427395794</td>
<td>14366460</td>
<td>10.2663</td>
<td>0.266</td>
</tr>
<tr>
<td>6838441011820</td>
<td>13616026</td>
<td>9.7300</td>
<td>0.269</td>
</tr>
<tr>
<td>6838455003761</td>
<td>13991941</td>
<td>9.9986</td>
<td>0.0013</td>
</tr>
</tbody>
</table>
Conclusion

- javax.realtime package is built
- Gaps in Java have been bridged
- Widens the scope of Java
- Limitations
  - Overhead due to JNI
  - Not for hard real-time systems
- Other commercial versions of RTJ – not freely available to all
Other Challenges

- Memory Management
- Garbage Collection (GC)
- Contribute to unpredictable behavior
Other Challenges (cont…)

Memory Management
- JVM allocates memory from heap
- Does not use a specific allocation algorithm – leads to non-deterministic results
- Solution – create No Heap Real-Time (NHRT) threads
- Dynamic checking (at every read & write) – so that NHRT threads do not access any location into the GC heap
- Limitation – overhead due to this kind of dynamic checking for memory access for the NHRT threads
Other Challenges (cont…)

- Garbage Collector (GC)
  - GC acts unpredictably
  - NHRT threads should be made to preempt GC
  - Wider perspective – Sun’s Java HotSpot uses a compacting mark & sweep algorithm for GC
  - Objects are grouped into following generations:
    - Nurseries
    - Older generation
  - for Nurseries – generational copying collector
  - for Older generation – mark-compact collection algorithm
Thanks for your (real) time!