Novel Java code in Chronicle

IT Days - Joker 2020

Peter Lawrey

CEO, Chronicle Software

Extending Throwable

Chronicle SOFTWARE

```
public class StackTrace extends Throwable {
    public StackTrace() { this("stack trace"); }

    public StackTrace(String message) { this(message, null); }

    public StackTrace(String message, Throwable cause) {
        super(message + " on " + Thread.currentThread().getName(), cause);
    }

    public static StackTrace forThread(Thread t) {
        StackTrace st = new StackTrace(t.toString());
        st.setStackTrace(t.getStackTrace());
        return st;
    }
}
```

Extending Throwable

Chronicle SOFTWARE

Creating a StackTrace class for holding stack traces is useful for monitoring and reporting. By extending Throwable it can be…

Extending Throwable

Chronicle SOFTWARE

Report where an object was created.

```
private transient volatile StackTrace createdHere =
        Jvm.isResourceTracing() ? new StackTrace("Created Here") : null;
```

```
if (!isClosed()) {
    if (Jvm.isResourceTracing()) {
        Jvm.warn().on(getClass(), "",
                new IllegalStateException("Discarded without closing", createdHere));
    }
    close();
}
```

Extending Throwable

Chronicle SOFTWARE

Report where an object was closed.

```
@Override
public final void close() {
    if (UNSAFE.getAndSetInt(this, CLOSED_OFFSET, 1) != 0) {
        return;
    }
    closedHere = Jvm.isResourceTracing() ? new StackTrace("Closed here") : null;
```

```
public void throwExceptionIfClosed() throws IllegalStateException {
    if (closed != 0)
        throw new IllegalStateException("Closed", closedHere);
```

Chronicle SOFTWARE

Report where threads are still running.

```
allStackTraces = Thread.getAllStackTraces();
for (@NotNull Map.Entry<Thread, StackTraceElement[]> threadEntry
                                : allStackTraces.entrySet()) {
    Thread thread = threadEntry.getKey();
    @NotNull StringBuilder sb = new StringBuilder()
            .append("Thread still running ")
            .append(thread);
    Jvm.trimStackTrace(sb, threadEntry.getValue());

    StackTrace stackTrace = ThreadDump.createdHereFor(thread);
    StackTrace st = new StackTrace(thread.toString(), stackTrace);
    st.setStackTrace(threadEntry.getValue());
    ae.addSuppressed(st);
```

Pass by Name or DynamicEnum

Chronicle SOFTWARE

```
public enum Ecn implements DynamicEnum {

    EBS_LIVE_NYK(8),
    RFX(19),
    PARFX(18),
    EBS_LDN(6),
    // many more

    private final byte priority;
```

Pass by Name or DynamicEnum

Chronicle SOFTWARE

```
configSet: {
  eventId: E1,
  eventTime: 2020-09-09T01:46:40,
  eventName: refData,
  key: PARFX,
  event: !RefData {
    eventId: "",
    data: !Ecn {
      name: PARFX,
      priority: 99
    }
  }
}
```

Pass by Name or DynamicEnum

Chronicle SOFTWARE

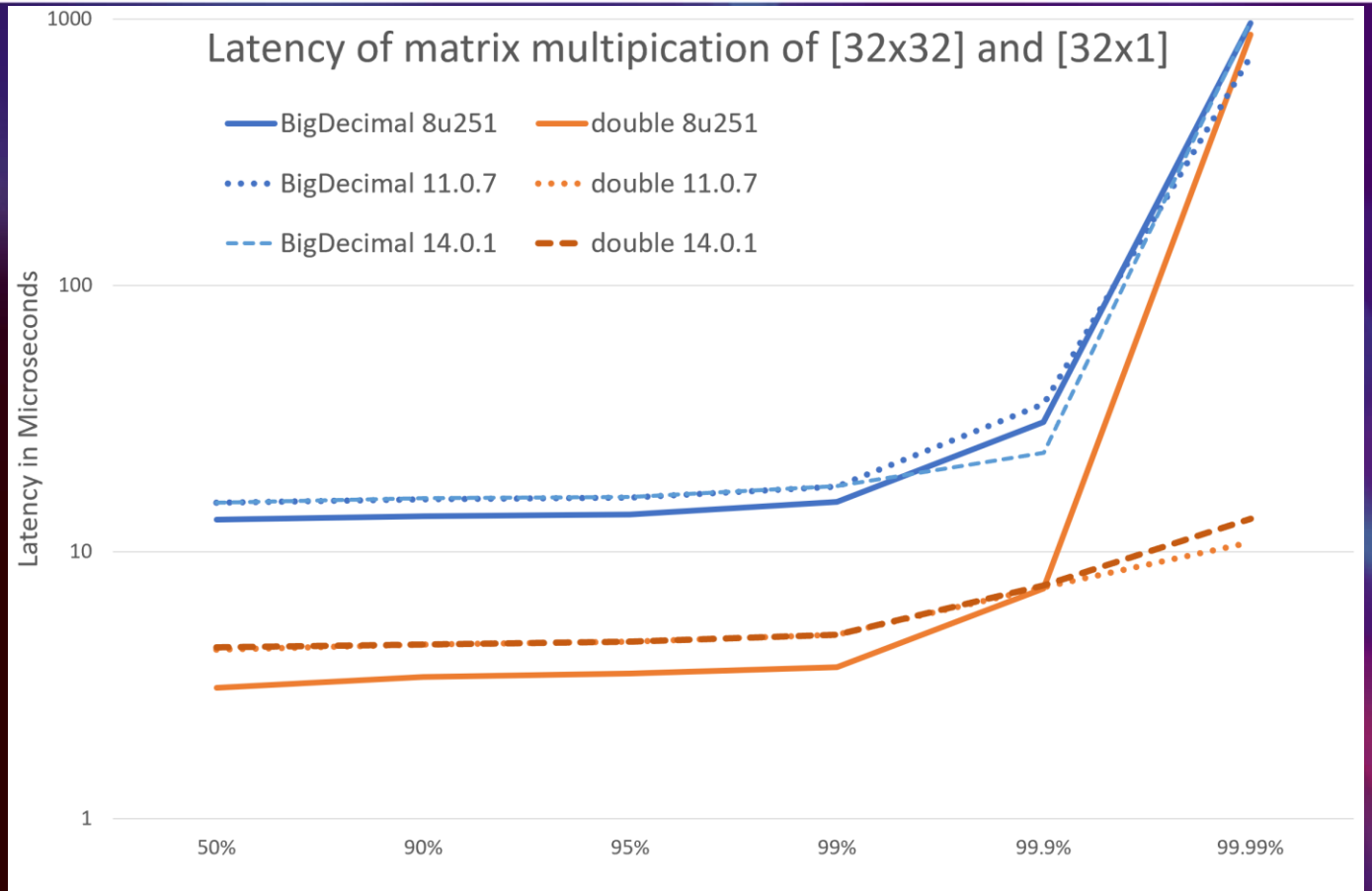
```
newOrderSingle: {
  eventId: E1,
  eventTime: 2020-09-09T01:46:40,
  ecn: PARFX,
  // more data
}
```

Chronicle SOFTWARE

Faster, GC free fixed precision

```
public static double round2(double d) {
    final double factor = 1e2;
    return Math.abs(d) < WHOLE_NUMBER / factor
            ? (long) (d < 0 ? d * factor - 0.5 : d * factor + 0.5) / factor : d;
}
```

Fixed precision rounding



Cleaning ThreadLocals

Chronicle SOFTWARE

```
public class CleaningThread extends Thread {
    public CleaningThread(Runnable target) {  super(target);  }

    public CleaningThread(Runnable target, String name) {
        super(target, name);
    }

    @Override
    public void run() {
        try {
            super.run();
        } finally {
            performCleanup(this);
        }
    }
}
```

Cleaning ThreadLocals

Chronicle SOFTWARE

```
public static void performCleanup(Thread thread) {
    WeakReference[] table = Jvm.getValue(thread, "threadLocals/table");
    if (table != null) {
        for (WeakReference reference : table) {
            Object key = reference != null ? reference.get() : null;
            if (key instanceof CleaningThreadLocal) {
                Object value = Jvm.getValue(reference, "value");
                if (value != null) {
                    CleaningThreadLocal ctl = (CleaningThreadLocal) key;
                    ctl.cleanup(value);
                }
            }
        }
    }
}
```

Machine Unique Microsecond timestamp

Chronicle SOFTWARE

```
public enum MappedUniqueMicroTimeProvider implements TimeProvider {
    INSTANCE;
    private static final int LAST_TIME = 128;
    private final MappedFile file;
    private final Bytes bytes;
    private TimeProvider provider = SystemTimeProvider.INSTANCE;
    MappedUniqueMicroTimeProvider() {
        String user = System.getProperty("user.name", "unknown");
        file = MappedFile.mappedFile(OS.TMP + "/.time-stamp." + user + ".dat", OS.pageSize(), 0);
        bytes = file.acquireBytesForWrite(mu ReferenceOwner.temporary("mumtp"), 0);
        bytes.append8bit("&TSF\nTime stamp file uses for sharing a unique id\n");
    }
    @Override public long currentTimeMicros() {
        long time = provider.currentTimeMicros();
        while (true) {
            long time0 = bytes.readVolatileLong(LAST_TIME);
            if (time0 >= time)
                time = time0 + 1;
            if (bytes.compareAndSwapLong(LAST_TIME, time0, time))
                return time;
            Jvm.nanoPause();
        }
    }
```

A collection without explicit thread safety

Chronicle SOFTWARE

```
public class StringInterner {
    @NotNull
    protected final String[] interner;

    @Nullable
    public String intern(@Nullable CharSequence cs) {
        if (cs == null)
            return null;
        if (cs.length() > interner.length)
            return cs.toString();
        int hash = Maths.hash32(cs);
        int h = hash & mask;
        String s = interner[h];
        if (StringUtils.isEqual(cs, s))
            return s;
```

Checking documentation of unchecked exceptions

Chronicle SOFTWARE

```
public class IllegalStateException extends Exception {
    public IllegalStateException(String message) {
        super(message);
    }

    public IllegalStateException(Throwable cause) {
        super(cause);
    }

    public IllegalStateException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Chronicle SOFTWARE

Caching meta data for classes

```
static final ClassValue<Method> READ_RESOLVE = ClassLocal.withInitial(c -> {
    try {
        Method m = c.getDeclaredMethod("readResolve");
        Jvm.setAccessible(m);
        return m;
    } catch (NoSuchMethodException expected) {
        return null;
    } catch (Exception e) {
        throw new AssertionError(e);
    }
});
```

Chronicle SOFTWARE

Vacuous exceptions

```
public static <T extends Throwable>
        RuntimeException rethrow(Throwable throwable) throws T {
    throw (T) throwable; // rely on vacuous cast
}
```

```
@FunctionalInterface
public interface ThrowingConsumer<I, T extends Throwable> {
    void accept(I in) throws T;
}
```

Vacuous exceptions

```
static <I, T extends Throwable>
Consumer<I> asConsumer(@NotNull ThrowingConsumer<I, T> function) {
    return in -> {
        try {
            function.accept(in);
        } catch (Throwable t) {
            throw Jvm.rethrow(t);
        }
    };
}
```

Chronicle SOFTWARE

Add a profiling point before or after a key operation or lock

```
public static void safepoint() {
    if (IS_JAVA_9_PLUS)
        Safepoint.force();
    else
        Compiler.enable();
}
```

https://stackoverflow.com/questions/62550828/is-there-a-lightweight-method-which-adds-a-safepoint-in-java-9

```
static class Safepoint {
    private static volatile int one = 1;

    public static void force() {
        // trick only works from Java 9+
        for (int i = 0; i < one; i++) ;
    }
}
```

Chronicle SOFTWARE

Add a hint for busy waiting loop

```
for (; ; Jvm.nanoPause()) {
```

```
public static void nanoPause() {
    if (onSpinWaitMH == null) {
        safepoint();
    } else {
        try {
            onSpinWaitMH.invokeExact();
        } catch (Throwable throwable) {
            throw new AssertionError(throwable);
        }
    }
}
```

Web exception handler

Chronicle SOFTWARE

```
public class WebExceptionHandler implements ExceptionHandler {
    private final Properties properties = new Properties();
    private final ExceptionHandler fallBack;
    private final String baseURI;

    @Override
    public void on(@NotNull Class clazz, @Nullable String message, @NotNull Throwable t) {
        try {
            if (Jvm.isDebug() && Desktop.isDesktopSupported())
                Desktop.getDesktop().browse(new URI(uri));
            else
                fallBack.on(clazz, message, t);

        } catch (Exception e) {
            fallBack.on(WebExceptionHandler.class, "Failed to open browser", e);
        }
    }
```

Questions

Chronicle SOFTWARE

https://chronicle.software

Linkedin: Chronicle Performance Engineers

@ChronicleSoftw

sales@chronicle.software