Real-Time Java Gateways: When Milliseconds Matter

Gateways are a critical element of the architecture of the Internet of Things. The IoT is inherently a system of systems. There are ele-ments in the "cloud" that run on traditional servers using IT infra-structure and there are elements at the "edge" that use embedded systems technologies. The cloud offers big data analytics, machine learning, and ubiquitous connectivity.

Embedded devices offer the ability to monitor and control almost any physical system in the world. Service gateways, also called M2M gateways or IoT gateways, bridge these two environments to create an integrated network combining the power and features of both. IoT gateways bring storage and computing resources out to the edge of the network where they can be used to aggregate, filter, and respond to data as it is collected and streamed to the cloud. They also provide a means of securely interfacing legacy end-devices to the IoT.

The Value of Java

One of the most useful technologies for implementing IoT gateways is the Java language, runtime, and ecosystem. Java surpasses C++ as a high-level object-oriented language with automatic garbage collection, robust exception handling, built-in threading model, and a wealth of standard and third-party libraries for doing all the things an IoT gateway requires. Programming in Java is safer and more reliable by avoiding common coding errors, such as dereferencing stale pointers and trampling memory. Java is one of the most popular languages for computer science degree programs and there are over nine million Java developers worldwide . Standard frameworks such as the Open Systems Gateway Initiative (OSGi) add capabilities to Java that further enhance its usefulness in the Internet of Things. OSGi bundles can be downloaded, installed, started, stopped, and uninstalled over the network without having to restart the Java Virtual Machine. This can be a huge benefit for systems that must operate 24/7. Real-world IoT gateways are being deployed today running the open source Eclipse Kura OSGi framework, which is derived from the commercial Eurotech Everyware™ Software Framework . Kura and ESF provide services for messaging, I/O, configuration, remote management, networking (including Bluetooth LE and cellular), and web administration.

Traditional Java and Real Time

Something traditional Java cannot handle well is real-time response. Real-time systems have to guarantee that they will respond to an event or input within a deadline. Most Java programmers have experienced the unexpected delays that can occur when a "stop-the-world" garbage collection cycle runs. Depending on the size of heap memory and the fragmentation of live objects, GC pauses can last hundreds of milliseconds up to several seconds . But GC pauses are just the tip of the iceberg when it comes to traditional Java's real-time shortcomings. For example, the Just-in-Time compiler may kick in at an inoppor-tune moment and delay execution of a critical bit of code. Java thread priority is used as little more than a hint to the operating system scheduler, which may unpredictably favor low-priority threads over high-priority threads. Worse yet, a priority inversion can occur which blocks a high priority thread from running indefinitely. Add to this the uncertainties of page faults, preemption by other processes, and the lack of precise timing APIs and you might be convinced that Java could never guarantee a program will ever finish, let alone meet a deadline.

But you'd be wrong. It is possible to build a Java Virtual Machine that supports real-time response – it just doesn't come with standard Java SE. The PTC Perc product has been deployed in real-world, real-time systems since 1998. It is used in telecommunications, industrial control, aerospace, automotive, and defense applications. And now it is taking on the Internet of Things as the go-to technology for gateways that need to react within milliseconds to critical events.

Milliseconds Matter in numerous Scenarios

Think about a wind turbine that detects a spike in gearbox oil temperature. With a traditional IoT gateway, an alarm will be transmitted to the cloud but by the time the command to shut down is returned, the gearbox could be beyond repair and secondary failures or a fire could ensue. An intelligent, real-time Java-based gateway can be programmed to feather the turbine blades and engage the braking system before catastrophic failure. The gateway can be updated with algorithms for early symptom detection and rules for remedial action to take as the IoT cloud applies analytics and machine learning to the data from hundreds of identical turbines around the world.

Real-Time Features

Table 1, shows the features of traditional Java Standard Edition vs. PTC Perc.

Feature	Java SE	PTC Perc
Just-in-Time (JIT) Compiler	Yes	Yes
Ahead-of-Time (AOT) Compiler	No	Yes
Real-Time Garbage Collector (GC)	No	Yes
Priority Inheritance Protocol	No	Yes
Precise Timing APIs	No	Yes
GC monitor and control APIs	No	Yes
Real-Time Scheduling	No	Yes
Page Locking	No	Yes

Table 1.

PTC Perc supports Just-in-Time (JIT) compilation of Java byte codes to native code but adds an Ahead-of-Time (AOT) compiler that allows developers to pre-compile their applications and Java libraries to native code and link them into the PTC Perc Virtual Machine just like a C/C++ toolchain. This avoids unpredictable JIT delays.

PTC Perc uses a patented real-time garbage collection algorithm that allows high priority threads to interrupt GC activity at any time to perform time-critical tasks. When the task is finished, the GC continues running. There are no "stop the world" collection cycles.

PTC Perc implements the Priority Inheritance Proto-col when threads contend for shared locks, thereby preventing the priority inversions that can occur in traditional Java SE.

Additional Timing APIs in PTC Perc permit develop-ers to schedule jitter-free periodic tasks, sleeps, or waits. The new APIs use an absolute uptime-based scheduling mechanism so it is not subject to disruptions caused by a change to the system clock.

The PTC Perc VM lets the user set the priority and timing parameters of the garbage collector to control when it runs and how much CPU time it is allowed to consume in relation to Java threads. On SMP processors, multiple GC worker threads run concurrently with Java threads and the user can assign which cores are to be used for GC, thus reserving cores only for Java.

PTC Perc allows the user to set real-time scheduling policies in Linux to avoid preemption by other processes. The real-time priorities supersede all normal threads.

And the PTC Perc VM can be configured to lock all memory pages into physical RAM rather than wait for page faults to occur. These features and others make PTC Perc a robust, real-time Java solution.

PTC Perc: When Milliseconds Matter

Java is a great language and platform for building IoT gateways. When a gateway or any other embedded system needs to respond quickly and predictably to events, don't think you have to give up all the benefits of Java and switch to C or C++. PTC Perc is the right Java to use when milliseconds matter.

Learn More

If you would like to learn more or evaluate Perc for your real-time application needs, feel free to go to the PTC Perc homepage:

https://www.ptc.com/en/products/developer-tools/perc

J06551–Real Time Java Gateways-WP–EN