DATA-CENTRIC APPROACH TO MONITORING EMBEDDED COMPUTERS

EXTENDS COVERAGE TO APPLICATIONS, OFFERS INTEROPERABILITY

WHO HAS BEEN MONITORING YOUR EMBEDDED COMPUTER?

by Vincent Chuffart, Senior Business Technology Manager, Kontron

A recent book entitled: “Principles: Work and Life” (Ray Dalio) explains that there are far fewer types of problems than there are actual issues. In the embedded computer domain, thousands of applications designed to solve many use cases eventually operate on only a handful of computer architectures. This is why a computer-centric view of a problem can lead to a powerful and generic solution, whatever the industry segment.

The ultimate mission of a computer system is to serve the application. Regardless of the complexity of the application, all the resources of a computer are accessible by the CPU. With this in mind, the CPU is able to monitor all resources, which in-turn means that by using the basic tools (OS, libraries, drivers) the application has the means to collect accurate and thorough computer health information, as long as it is designed with the right test points.

Classical resources required by embedded applications include disk storage and memory space, CPU computing time, network connections and bandwidth. Add a couple of embedded sensors like temperature and power to complete the picture. Modern OS’s now offer powerful command line utilities to access them, leveraging many years of improvements from the open source community. Deeper and less generic sensors also exist to serve specific requirements (eg: shock, tamper detection, trusted computing, etc) and may be proprietary. These are, however, also accessed from the OS interface thanks to dedicated device drivers and user space programs. In both the generic and specific domain, the main issue lies in the expertise required to use the programs properly and deduce the correct status from their readings. Another problem is to make sure they will always be implemented the same way, operate and at the right frequency.

Imagining the Right Monitoring Approach

How about a system that could guarantee a coherent, repeatable and thorough assessment of a computer’s status? Wouldn’t it be useful if it could access existing commands to capture all the know-how along the way? Could this system also be made customizable and extensible? Would it be able to follow all the changes incurred by the many steps of open architecture computers through integration into systems and solutions? Could this approach offer greater interoperability between components?

Such an ideal solution can only be found with the right level of abstraction. In the case of computer health monitoring, it comes from the data model defined to tackle the problem. Here is the approach in CMON. **Download the CMON White Paper**

Let's define the smallest unit, the health sensor object. It represents a single test point, and everything required to use it. It takes one of three values (Success, Warning, Alarm). The health assessment takes place in the health tree, with the root being the global system status, while each leaf is a test point. Branches and nodes represent a logical view of the status by grouping sensors in logical families (volt, temperature, storage, VPD etc). Health values propagate from the leaves to the root.

Standalone Test Points

The health sensor data includes all the expertise relevant to a test point: how to measure, conditions for alarm or warning, measurement period, information about the test, "what if" recommendations. More properties are also designed for run time information: health value, time stamp of the last reading, reading value, min/max boundaries, and counters.

The sensor object approach allows for many embedded computing use cases with a generic, future-proof health assessment engine and a couple of files. Modified and refined, the collection of sensor objects embarks each experts' knowledge that usually lies deep in test documents. Best tricks discussed at the coffee machine also find their way into the health sensor object. In the end, a single text file contains the summary of the health of this computer.

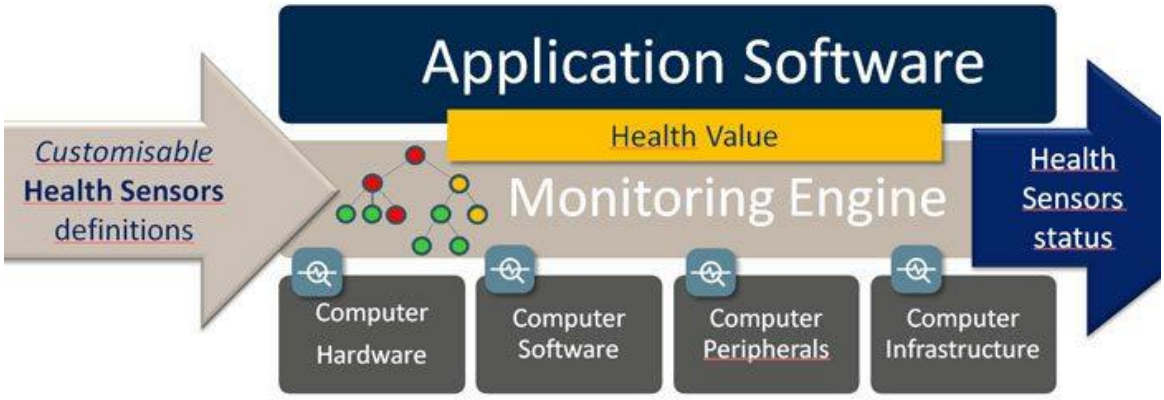

Figure 1: CMON Health Assessment Engine

The job of the health assessment engine then boils down to reading the input file, processing objects in memory (executing the sensor command and updating the run time data accordingly), and writing the result file. In the case of a CBIT, this file is written each time the health tree is traversed. This time depends on the most demanding health sensor (the one with the shortest measurement period, which can be changed).

Using Monitoring data

In many embedded computing problems, the global assessment of the platform will be used by the main application, in order to make educated operational decisions, depending on the mission. This is done with the health engine library linked into the main application code. Everything happens within the application which also schedules the health assessment, avoiding mission critical periods.

In most cases the CBIT service is used. Made from a simple loop around the health engine, it maintains computer health data in memory, while keeping the CMON result file continuously up to date. From this file and thanks to CMON data model, displaying role specific dashboards is just a transformation away.

Figure 2: Building dashboards from CMON data using XSLT transformation

CMON files include the complete picture of a system content and health. As simple text files, they are useful in FAI reports, acceptance tests, and conformity assessment. Used as input to the health engine, they become the most critical companion of software engineers, using techniques from the best experts in the field to keep you informed on your computer status as you work.

Processing Monitoring data

The current implementation uses XML format for sensor information enforced by a XTD grammar. Both input and output files use a unique syntax, which offers interesting possibilities (resuming an interrupted long session by feeding the last result file before reboot, borrowing an old result file to use as a reference years later, etc).

In the last decade the Big Data industry has developed powerful techniques to process data whether it is a continuous flow, or at rest in data lakes. Thanks to a simple and generic data model, all such techniques can be used to process CMON monitoring data, either locally, in "air gapped" use cases or with services running in the cloud (data storage, analytics).

Another vibrant industry is cloud computing, and its health assessment problem is not entirely unlike our embedded computing domain (no one is sitting in front of the computer either). This booming segment has created a large market for APM (Application Performance Monitoring) solutions addressing the monitoring of thousands of virtual machines. Feeding CMON data to these frameworks takes only a few lines of python and offers a unique point of view to the most intricate details of your computers small or large fleets.

Even if it can only be used during non-operational testing (when your infrastructure can be connected to the outside world), this money saving approach needs to be considered seriously.

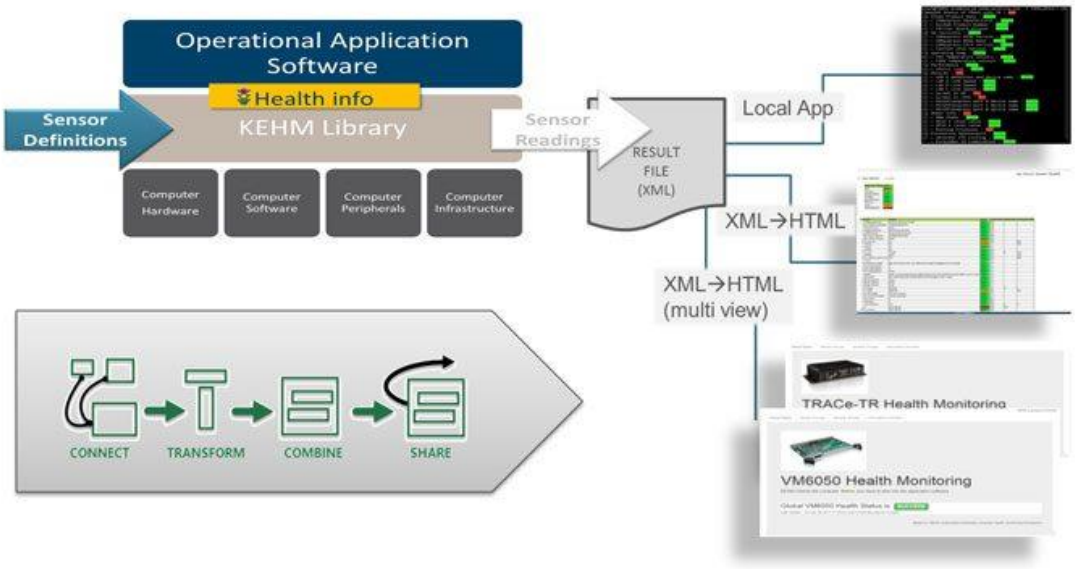

Figure 3: Monitoring data flow benefits from industry tools and techniques

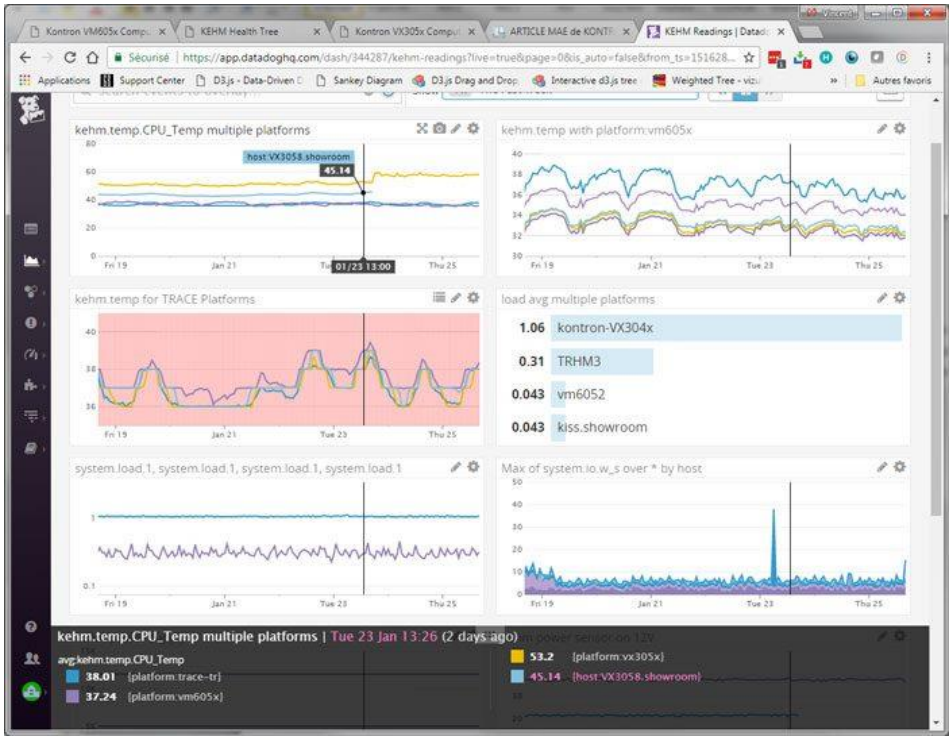

Figure 4: Sending CMON data flow in cloud APM service (Datadog)

Conclusions

A data centric approach to computer monitoring can accumulate computer expertise from all actors into a continuous health data flow, ready for immediate or delayed processing. The CMON data model is designed to fulfill most of the open architecture systems use cases, covering from the design phase, through customization and integration into deployment and support. The rest is data plumbing…

References :

- ***CMON WHITE PAPER**, CMON Monitoring Users Guide*
- Examples of cloud based APM (Datadog, New Relic) .
- One place to learn data transformation techniques from scratch (www.w3schools.com/)

To learn more about Kontron's CMON products please visit
https://www.kontron.com/products/solutions/control-and-monitoring

Contact Us

- Contact Us Form
- sales@kontron.com