RC24799 (W0906-011) June 2, 2009
Other

IBM Research Report

End-to-End Automated Analytics for CBM+

Nathaniel Mills
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

End-to-End Automated Analytics for CBM+

Abstract

End-to-End Condition Based Maintenance Plus (CBM+) requires careful balances between data generation, filtering, preprocessing, compression, secure communications, cleansing/validation, persistence, categorization, classification, analysis, summarization, reporting and distribution. Often, the conditions themselves demand flexibility within the CBM+ framework to adapt for data volumes, traffic priority, and critical impact on safety and operations. Intelligent services distributed across disparate operational environments and varying data schemas enable collaborative applications to strike the balance needed for effective operation of CBM+ systems. Open standards in a service oriented architecture enable cooperating processes to provide synergy and expand opportunities for discovery and insight to diagnostic procedures and predictive maintenance.

Too often, CBM+ has focused on the data structures and detailed component structure models resulting in overly rigid designs that are labor intensive and have difficulty accommodating change brought on by varying platform configurations and new insights from analysis. This paper discusses how automating data analysis for CBM+ is possible in a distributed, open standards based SOA environment from the embedded applications on monitored platforms through intermediate processing / communications gateways to the enterprise repository and among collaborative service providers and consumers. In particular, we draw from experience gained in US Army and commercial automotive applications to highlight challenges and possible solutions. We discuss data synchronization strategies, distributing cooperative analytics, data compression and summarization strategies, how automated analysis of CBM data can improve OLTP and OLAP management and performance strategies for CBM+, and how to simplify SOA implementations for quick adoption while retaining open standards and flexibility. We also discuss driving factors for CBM+ framework requirements and their affect on selecting among alternative implementation strategies.

CBM Goals

The simplified goal of Condition Based Maintenance (CBM) is to adapt service schedules to individual components based on their needs, as opposed to elapsed time. Goals that are more aggressive include evaluating the conditions of actual and predicted use to plan for preventative maintenance. Regardless, the desire to tailor individual maintenance schedules can reduce costs while improving availability by avoiding unwarranted repairs. If done well, and with enough foresight, it does not preclude improving the logistics supply chain, maintenance resource scheduling, parts distribution strategies and equipment availability. However, being able to glean what conditions drive necessary maintenance is more complicated than purely time-scheduled maintenance as they require usage models and fault impact analysis.

Correlation

Sensors monitoring equipment operations need to include enough information to supplement direct measurements (observations) with correlated associative or model-based derived measurements describing the regime at the time the direct

measurements were observed. This may require only capturing enough hooks for late binding with other data sources or to enable algorithms to deduce the regime based on systematic review of multiple data sources and behavior models. Correlation of direct measurements is an important aspect of CBM as it provides the basis for deriving operational states, higher-level activities (events), health assessments, alerts, and generation of actionable advice. Correlation often involves time sequencing of related measurements or commonly observed events.

Time as a Correlator

One important correlator for CBM involves time of day, relative time, or sequencing. Time of day measurements facilitate late binding with other data sources. However, unless a common, reasonably accurate timing source as is found in GPS measurements or from time synchronization infrastructures (e.g., NNTP) this measure will only provide loose correlation at best. It may still be suitable for late binding with temperature, pressure and/or weather measurements where change occurs relatively slowly. Relative time measurements are more important for correlation across multiple sensors / multiple measurements provided they share a common timing source. This may require additional communications among sensors to ensure their timing sources do not drift beyond an acceptable distance. Timing made relative to a common event may also require less storage than calendar date / time, as only 8 bytes may be sufficient to hold high precision timestamps for a long enough period to cover the recording period. Depending on the environment, time may not be able to be synchronized across sensors. However, sequence counters may still enable correlation across measurements if they are made frequently enough and there is an abundance of shared event activity between different sensors to allow for periodic alignment. Sequences can also help to differentiate gross level sets of measurements. For example, when time is maintained by a battery backed time of day clock, and the battery fails or other error causes the clock to be reset to the beginning of its epoch, sequence numbers can still position one set of measurements relative to others. Later analysis and best guess estimates may allow for reasonable recovery of the measurement period.

Events as Correlators

Often, sensors monitor pieces of equipment that communicate with other each other on an ongoing basis. An event may be shared between the equipment (e.g., as conveyed in a communications message sent from one to the other) and be seen by both sensors. The propagation delay for communications across a bus or LAN may not be significant or at least may be constant enough to permit alignment of two independently operating sensors output. At the very least, relative sequencing should be possible to enable analytics to glean which sensor readings occurred after others, or between "significant" events.

Measurements vs. Events

Events may also be deduced through modeled behavior and algorithms identifying a sequence or pattern. While an event may map directly to an observation, in this context we differentiate derived events as representing an observation about the system's state and may or may not be directly observable by an individual sensor. For example, the determination that a misfire has occurred, or when the system has transitioned from a running state to an idle state, or inappropriate operation when shifting gears may all reflect events requiring assessment of multiple observable

measurements and take some subjective analysis into account to determine when the event has occurred.

Data Inundation

Initially, CBM suffered for lack of observed measurements. There were not enough strategically placed sensors to provide insight to system behavior. Luckily, as more software-based systems came on-line, the measures needed by the embedded software to make decisions also provide data for CBM analysis. Today, unless the system is antiquated and/or still relies heavily on pure physical linkages for operation, there is no dearth of operational data. In many cases, the opposite problem exists with too many low-level data packets, and the problem is how to archive, filter, consolidate, and analyze them for actionable intelligence.

CBM systems design involves maximizing coverage of system operations through strategic placement of sensors, minimizing data sizes and transmission times, and distributing analysis for tactical (e.g., safety, availability, mission readiness) and strategic (e.g., mission planning, parts availability) benefit.

Sensing Sensor Behavior

Having many sensors presents a new problem -- how to know when to trust the sensors' observations. Derived events, based on modeled behavior can play an important role in determining whether sensors are behaving properly. While sensors may be thought to be reporting observed states, they can break, overheat, or misbehave when inappropriate voltages or vibrations are present. We have observed broken temperature sensors that reported valid but inaccurate readings, and airflow sensors that became fouled over time and reported false airflow problems. We have also observed inaccurate readings due to unforeseen circumstances when mice had eaten the air filter and paper crumbs had been sucked into the airflow preheat chamber affecting the heating elements effect on circulating air. To help determine whether to trust sensor readings, operational models may be used to derive expected states for comparison against sensor readings. For example, a comparison of the actual and demand torque readings from the engine may provide a check and balance against sensed airflow readings. If the engine is able to provide the torque demanded, then it may be safe to assume airflow is not restricted and the sensor should be checked. Ideally, such checks and balances should be performed in close proximity to the sensors to alert operators so mechanics can affect repairs.

When / Where to Perform Analysis

Data analysis may not always be possible on-platform, and needs to be deferred until more time, CPU power, or other information is available. For example, it may be desirable to review every possible message that could flow across the bus, but to do so would involve soliciting data from embedded controllers that might overload the bus bandwidth and block crucial messages from flowing. Constant interrogation of the embedded controllers may affect their operation by using CPU to reply instead of doing expected work. This identifies three other CBM system design challenges:

- How should the "monitoring" infrastructure interact with the "natural" system so it does not interfere (can additional probing be performed for short bursts that would otherwise not be supported on a continuing basis)?

- How close should filtering be performed to the generation source to reduce storage / communications bandwidth requirements without hiding needed detail from downstream analytics?
- How should analysis of operational and/or derived data be distributed to achieve minimal impact on the system and maximum awareness of problems, in a manner that balances these other two considerations?

Contextual, Temporal Data

CBM typically requires access to historic data for comparison against current operations in order to determine degrading conditions. Consider monitoring fuel consumption over time to infer a blocked / dirty fuel filter, or dirty / worn out spark plugs, or a weakened fuel pump. These subtle changes may not be easy to notice within a short time window or small set of measurements. However, by comparing current consumption against history spanning days or weeks, one could identify a downward performance trend. Even better, accurate assessment is more likely when considering the regime so comparisons are made for similar stresses on the system. Historic comparative analysis requires data persistence for lengthy periods. Unless the domain being monitored is relatively stable and well understood, it is likely that sensor software, as well as the models and rules used for filtering, collation and correlation will change. To ensure fair comparisons are made, the data recorded for historic comparisons will need to be tagged with the version of logic used to produce them.

Parameter Set Definitions

The set of parameter definitions of measurements provides other utility. When data are managed over lengthy periods, one may need to transform the strategy used on earlier "recordings" to match current strategies. For example, a change from kilometers per hour to miles per hour, or Fahrenheit to Celsius is simple, but still required to enable accurate comparison. In cases where data need to be normalized to be fed into algorithms like neural networks the parameter definitions for the minimum and maximum allowed values can be used for this transformation.

Measurement Types

Knowing the reporting frequency for a parameter, and whether a measurement is one of the following is useful when needing to generate values for curve fitting or time alignment:

- **Enumeration:** where state transitions or sequence cannot be inferred (e.g., colors, or circuit breaker state (e.g., on, off, tripped) or the gear name)
- **Discrete:** where measurements are integers (e.g., cyclic counters)
- **Continuous:** where measurements are rational numbers (e.g., RPM, voltages, temperatures)

Filtering

While not long ago, the focus was on whether or not sensors existed to produce measurements, today the problem is managing too many low level details. Not only is this a storage issue when memory is constrained, but also effects transmission times and operations used to collect data. Filtering can begin as early as the initial visibility of the data presented to sensors. For example, monitoring messages on a communications bus may employ filters to determine whether messages have changed enough to warrant further analysis by the sensor. Filters can mask off the

portions of messages expected to change (e.g., a timestamp, or sequence counter) and may mask off insignificant changes in precision. Another type of filter may compare the current against past measurements to report only significant changes.

Band Filters

While similar to precision filters, banding can be used to track significant changes in data. Imagine measurements for revolutions per minute (RPM). Unless the RPM are being analyzed to deduce vibration requiring relatively high precision, it may be suitable to track changes that are a distance of +/- 5 (or possibly even 10 or 25). This distance is considered the filter band width, and the center of the band will move when an observed measurement occurs outside the current band's perimeter. In other words, values observed within the same band are considered not to have changed significantly enough to warrant recording them. Band filters differ from precision filters because precision bands are spaced equally, and separate the possible range of measures into fixed bands, one on top of the other. Band filters are set based on using the first value observed outside the current band as the center of the new band. Using distribution counts for predefined value ranges (that may not be evenly distributed (e.g., using a logarithmic scale)) is another way to reduce the data storage and reporting requirements.

Time Frequency Alignment

In order to align different measures occurring on different reporting frequencies or that are reported sparsely due to filtering on a common timeline, it might be necessary to generate representative values when none were reported, or that were previously filtered. In these cases, the parameter definitions and knowledge of the filters employed are useful to know what strategy should be used to generate missing values. For example, when precision or band filtering is employed, it may be best to consider a value retains its last known value until one reporting period before a new value is reported. For values within a reporting period, linear interpolation, or polynomial splines can be used. The candidate generated value needs to take into account the parameter measurement type so generated values make sense (e.g., you would not want to report that the gear was 2.5).

Parameter Semantics

Managing the semantic namespace for parameters is important when similar but different machinery is to be compared. Attempting to use the same semantic meaning from one parameter set to another can enable common references used by rules or algorithms to evaluate states of operation. For example, the state of being at idle may differ from one type of engine to another, but each can achieve this state. For higher-level analysis, it may be necessary to know whether the engine is idling (e.g., has achieved a desired state) when evaluating or correlating other measurements. Idle definitions may differ depending on the context. For example, while a range for RPM may be defined for engine idle, it may be important to consider the gear position, throttle position, load from other equipment (e.g., air conditioning), and the period over which the RPM remains within the idle range. These nuances will often be refined over time based on the development of more advanced analytics used to derive health or status measures. The possible change in meaning as knowledge about the measurements becomes more refined, demands that the context under which measurements are produced be recorded along with the measurements themselves. Tracking the semantics of the logic under which data is

being produced along with the data is the only way to ensure the CBM data warehouse can be used for historic comparison and trend analysis.

Fault Tree Diagnostic Model

Too often, diagnostics use a singular approach based on a fault tree to resolve issues. This approach is based on first identifying the configuration of parts assembled to comprise the whole system. Then, each part is reviewed to identify each of the failure modes it can produce. The diagnostic process becomes an iterative testing process where each failure mode is examined to determine if the part is working as designed. When a failure mode is detected, the part becomes a candidate as the cause of the problem being researched (part of an ambiguity group). Finally, review of the set of failed parts and their failure modes is performed to try to reduce the ambiguity group to the final set of causal failures, and parts are repaired or replaced. This diagnostic approach makes sense given the history of physical interactions of hardware, where linkages can be traced from one activity to another. It is very time consuming to construct these models because each part's failure modes must be identified, and then diagnostic procedures to determine if a failure has occurred must be developed. For systems that do not change very often, and whose behavior is largely tied to the physical aspects of its construction, the time investment may be justified. However, with the advent of software-controlled functions becoming pervasive in complex systems, the rate of change and/or flexibility in operating models makes fault tree maintenance very labor intensive. Alternative approaches must be considered to reduce the time needed to build and test the fault tree model and to accommodate systematic problems that result from the logical connections between physical parts that are not easily associated with single part fault modes. For example, when an operator shifts from forward to reverse when the vehicle is still moving forward and/or the engine RPM's are excessive, the system can be compromised. In this case, the "fault" is with the operator, not a specific part. This problem may present as wear or cracks on the gears without any causal explanation without appropriate analytics.

Analytic Semantics

Where the fault tree approach provides a single model for physical behavior, other semantics used for analysis allow behavior to be expressed using multiple, concurrent models. By approaching a problem from different perspectives, it may be possible to simplify the complex system into a subset of related functions: communications, electrical, power, physical, safety, comfort, replenishment, and operations. Sensors and analytics can be developed for individual behavior models and their output / results can be fed into other models, or into higher-level systemic models. By using semantics to present sensor and derived data from a particular perspective allows domain experts helping to define and develop diagnostics to use familiar reference labels. Any logic embodied within the diagnostic rules can also present their results from the perspective of the semantic model, yet these results may be cast using other reference labels in other semantic models, and become useful within other diagnostic domains. Allowing multiple views of the data adds flexibility, can focus studies on a smaller, related set of data, and simplify communications and interactions with non-IT oriented diagnosticians. Using semantics also allows a layer of indirection from the raw data values, and enables experimentation with new models that would not otherwise interfere with existing

semantic models. This flexibility enables diagnostics to be built in short order for the well understood and/or frequently encountered problems, while more advanced, complicated diagnostics can be developed and enhanced over time. This relieves the need to build the system configuration completely, then overlaying the monitoring locations for sensor or derived data, identifying the fault modes and defining the diagnostics to determine whether these faults have occurred. The semantic diagnostics system can begin to produce benefit much sooner while still enabling the fault tree diagnostics to be built and used where appropriate.

Distributing Analysis

It may not be possible to perform all analysis on measured data depending upon the environment where measurements are produced. It may be that the CPU has limited processing capability, or that storage is limited due to constraints prohibiting less expensive, spinning recording media due to temperature or vibration. When analysis is distributed and communications are required to move data from one analysis location to another, there are operational considerations. For example, is there a stable, available network for on-demand transmission of data from one location to the other, or is network connectivity intermittent? What is the available bandwidth for data transmission compared with the data production rates? Can analysis be performed by nodes within the communications network on data destined for the CBM data warehouse?

Opportunistic Data Collection

To move large amounts of data to more suitable analysis environments involves picking suitable communications paths, and additional processing to reduce the transmission size and time as much as possible. When mobile assets are being monitored, communications may not be available 100% of the time, or operations may not allow extended connectivity periods to allow large bulk transfers to occur. In this case, sufficient storage should be planned for the data generator balanced with sufficient contact with data collectors to allow accumulated data to be collected. We have experienced 200MB recordings collected over several days, and transmission over wireless communications were limited to 3-5 minute intervals to avoid interrupting normal operations. To operate under these conditions, we needed to provide multiple data collection points and to break apart these large recordings into smaller, self-contained units of work that can be moved during the collection interval. Frequent collections of partial data introduces its own set of problems as transfers can be interrupted and either need to be resumed with the same collector, or cleaned up if a second collector successfully retrieves the content. However, allowing for parallel transmission and collection paths provides a more robust net-centric approach to data retrieval and can allow speedier processing of the data.

Fragments and Chunking

Breaking a large recording or document into smaller, more manageable pieces can be as simple as picking a number of bytes that would normally be able to be sent within the collection time interval. However, these "chunks" of the larger "whole" require that the recipient has collected all the chucks to reassemble the whole before it can process the content. Large zipped binary files, or XML documents are examples of content where chunking may apply. However, if the data to be transferred can be broken into transactions or other self-contained units of work

containing header information to describe how each "fragment" relates to the whole, then each fragment can be sent ahead and be processed upon receipt before waiting for the entire collection of fragments to arrive. Additional accounting logic needs to be employed to ensure no duplicate posting occurs, and to track missing chunks or fragments for subsequent retrieval. Note that chunking and fragmenting can be performed as a background task while waiting for communications to become available.

In network situations where content may be significantly delayed or lost (e.g., a collection node is destroyed), chunked content and its "all or nothing" requirement is less flexible than fragments because the latter allows for some of the information to be processed while attempting to request retransmission of the missing chunks or fragments. In either case, bidirectional communication between the intended recipient and the generation or staging source is needed to ensure content is retained for a reasonable period to support retransmission requests. Once the intended recipient has processed the content, it can send a "receipt acknowledgement" allowing the original data to be deleted if desired to reclaim space for subsequent data generation needs.

Data Compression

Whether sending the original content, chunks or fragments, it makes sense to compress the content to save transmission time. As when partitioning large data into smaller pieces, the type of content should be evaluated when choosing the compression algorithm. Zip algorithms may work very well for binary data, while specialized Base64 encoding may work well for textual representation of numeric data, and XML can be highly compressed when considering its schema. If compression is part of the infrastructure, take care that previously compressed data is not being compressed again as this rarely results in improvement (it may actually add to the size) and wastes CPU cycles.

Data Security

When working with CBM data, there are security concerns. How secure are the communications and/or the data? Who is allowed to access the data? At what point does unclassified data require increased security due to aggregation? For example, knowing where a particular vehicle is located, or its health status may not be classified, but knowing the health and/or location of all military vehicles in a Battalion or Brigade could convey effective or potential force which is classified.

Data Synchronization

Moving data from the generation location to the warehouse should focus on the interoperability services used by communicating nodes, and not on the underlying data store schema. By providing a public interface to access and manage the data, implementers of software can have freedom to select the appropriate storage paradigms for their environments. Exposing the data store for direct access is often against Information Assurance (IA) regulations as this introduces denial of service attacks, and requires the data consumer to have intricate knowledge of the schema. This design is brittle as it requires heavy software blocking to introduce changes in the data persistence model, whereas having the indirection layer of the public service interface allows multiple interaction models to coexist. The public service layer also permits changes for new data structures or for performance reasons with

minimal disruption of service. Finally, the service interfaces can permit disparate environments to communicate without placing explicit demands on development languages or implementation details. The freedom of well-designed SOA helps to ensure data consumers and providers can be added to existing systems with relative ease.

Automated Analysis

Data collection strategies will vary depending on the amount of data, frequency of access, and use cases for data consumption needs. Using opportunistic data collection and data partitioning using fragments and chunks requires that data arriving out of sequence can be processed. This means a transactional data store at the final destination is best suited to persisting incoming data so partial updates can occur with minimal churn to retrieve / update / store data. However, for longer-term use of the data, maintaining access to individual records is inefficient as volumes can easily reach trillions of records. Instead, transitioning from an OLTP based schema to an OLAP schema where data are organized according to descriptive attributes should be performed as soon as practical. By using the service interface, public requests from data need not know whether the data resides in the OLTP or OLAP schema. To facilitate migration of data from OLTP to OLAP, automated analysis of the high volume, time series operational data can be performed to derive the lower volume, semantic attributes used to classify and describe aspects of the data. Automated analysis typically includes routing select related transactional data (e.g., records from the same, recorded trip or mission) through a battery of analytics that determine whether certain conditions were present. When conditions like inappropriate operation, or faults, or out of spec measurements are identified, these become attributes of the trip itself. The actual details can be stored as files or BLOBs and only need to be retrieved for in-depth studies. The majority of analysis can be performed on the metadata derived through automated analysis. Ad hoc analysis performed on through reporting against these attributes allows for further refinement of knowledge about the study being performed and may cause temporary expansion of the archived details. This is true for data mining use cases where a flatter schema is used to expose common attributes of select data content but does not need to be saved beyond the mining exercise.

Summary

End-to-End Condition Based Maintenance Plus (CBM+) requires careful balances between data generation, filtering, preprocessing, compression, secure communications, cleansing/validation, persistence, categorization, classification, analysis, summarization, reporting and distribution. We have attempted to surface various areas for consideration in designing and implementing CBM+ systems. Undoubtedly, the initial design will need significant revision due to unforeseen performance problems, interrupted and unstable communications, and the need for more detailed data, or smarter analytics to reduce data volumes. Focus the design on the public interfaces in a service oriented architecture for maximum flexibility to accommodate these changes and allow the existing system to remain functional as change is introduced. CBM+ is an iterative process much like data mining. The more you experience, the more you will want to refine and improve.

Author

Mr. Mills is the Chief Architect and Lead Engineer for Vehicle Health Management at IBM Research. Mr. Mills has won Outstanding Technical Achievement, and Research Accomplishment awards for the design and development of the Parametric Analysis Center, and Page Detailer commercial assets. Mr. Mills provides technical leadership for IBM service engagements for Condition Based Maintenance and Vehicle Health Management in both government and commercial / industrial sectors.

Author: Nathaniel Mills
Senior Technical Staff Member
IBM T. J. Watson Research Center
1101 Kitchawan Road, Route 134
Yorktown Heights, NY, 10598
wnm3@us.ibm.com

Direct mailing address:
16 Deer Hill Lane
Coventry, CT, 06238
860 760 3764 (voice and fax)