First Fault Performance Problem Resolution

Fix Mainframe Performance Problems Right the First Time

Table of Contents

Tel: +972 3 5220552, www.conicit.biz, info@conicit.biz

Background

ConicIT was founded based on the observation of a fundamental weakness of commonly used performance monitoring systems; too much data is available and there are no accurate analysis tools. We have reached a point where there is too much data for a human to process. This data overload is caused by two factors: the first being the time lag between problem occurrence (when meaningful analyzable factors are available) and problem detection when the data is only symptomatic. The second is that using static thresholds as an alerting mechanism leads to inaccurate alerts and mistrust of the alerting system. It became clear that if it was possible to predict problem occurrence before the data becomes dominated by symptoms, we could mitigate both problems – provide accurate alerts and pinpointed problem data.

There are several factors make that task extremely challenging:

- The problems are rare by definition, which poses a known difficulty to existing statistical prediction algorithms
- The number of potentially relevant features is very high, hence increasing the complexity and the data requirements for obtaining reliable predictions
- The data is non-stationary, hence ruling out most of the existing statistical methodologies
- Extremely noisy data, where the measurement noise often obscures the underlying phenomena of interest.

To address those issues we developed ConicIT for Mainframes, a unique product that uses sophisticated mathematics and algorithms to solve the problem of alerts and root cause analysis for mainframe systems:

- State-of-the art data cleansing algorithms, based on jump-diffusion models, non-linear Monte-Carlo filtering, and dynamic modeling, to address the non-stationary and non-linearity of the monitored signals, among other issues.
- A methodology for generation of composite problem detection criteria, taking into account the inter-correlation between the variables of interest.
- Advanced multi-expert prediction algorithms for look-ahead detection of emerging problems, that greatly facilitate the problem cause analysis.

Extensible performance models

Introduction

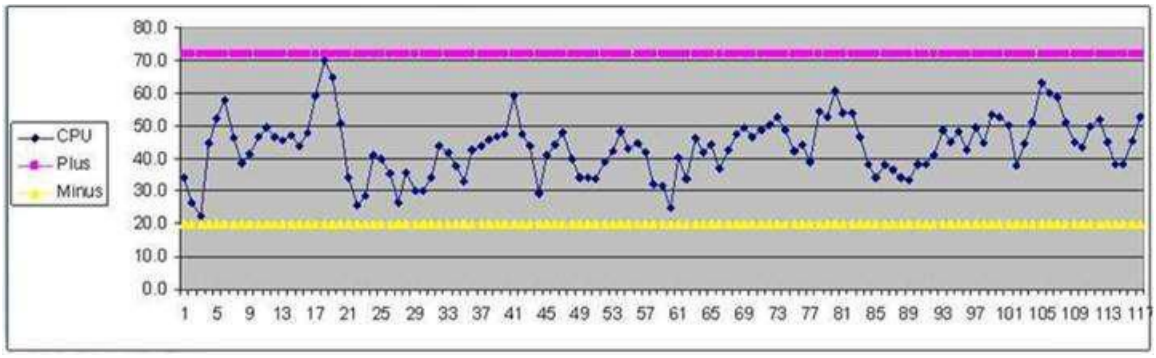
For large enterprises the data center is a critical part of their business infrastructure. New market requirements – such as web based customer self-service, globalization, and increased business agility are leading businesses to create new applications that are heavily reliant on existing legacy transaction computing infrastructure and applications. These new applications place heavy performance requirements on existing infrastructure and applications. This has made system performance even more business critical today than a decade ago. Since almost all of an enterprise's transactions interact with mainframes (which contain 70% of the world's business critical data) during processing, any slowdown has an immediate effect on the business, and a severe, long duration problem can be catastrophic. Over the years, excellent system monitoring tools have evolved but they aren't enough to provide the level of support needed by these new usage paradigms. These new applications and usage paradigms are straining mission critical mainframe applications and IT staff to the limit.

As a result, existing IT staff is being called upon to provide flawless service response and immediate problem resolution for application issues. Most production service degradation issues are related to unexpected and unplanned interferences between applications and transactions on the production system, making it impossible to provide flawless, 100% problem free operation - no matter how good your staff or monitoring solution. This leaves IT staff scrambling after the fact to discover and analyze the complex interactions that cause performance degradation – in most cases waiting for the problem to surface again before it can be fixed. Instead of after the fact detective work to solve performance problems, IT management and operations should focus on **first fault problem resolution**. That means both alerting IT staff and capturing all the relevant system data during a performance problem so that the IT staff will have a post-mortem detailed view of the system status right before, and during the problem's occurrence. Making this data available to the staff allows them to solve a problem the first time it occurs – providing a dramatic reduction in mean time to repair for transaction slowdowns and performance degradation.

ConicIT System Overview

ConicIT runs on a separate Linux system external to the system being monitored. ConicIT is a completely agentless architecture which doesn't require installation on the system being monitored. It receives data from existing monitors (e.g. Omegamon, TMon, Sysview, Mainview) through their standard interfaces. For mainframes, 3270 emulation enables ConicIT to appear as just another operator to the existing monitor and adds no more load to the monitored system than would adding an additional human operator. Until a problem is predicted ConicIT requests basic monitor information at a very low frequency (about once per minute), but if the ConicIT analysis senses a performance problem brewing, its requests for information increase, but never so much as to effect the monitored system. The maximum load generated by ConicIT is configurable and ConicIT supports all the major mainframe monitors.

The monitor data stream is retrieved by parsing the data from the various data sources. This raw data is first sent to ConicIT's data cleansing component. Data from existing monitors is very "noisy", since various system parameters values can fluctuate widely even when the system is running perfectly. **Graph 1** shows sample CPU utilization measurements over time from a perfectly normal system. Even though the CPU utilization is deviating greatly from the mean – it is not necessarily indicative of a system problem. The job of the data cleansing algorithm is to find meaningful features from the fluctuating data. Without an appropriate data cleansing algorithm it is very difficult or impossible for any useful analysis to take place. Such cleansing is a simple visual task for a trained operator, but is very tricky for an automated algorithm.



Graph 1: Normal CPU Fluctuations

The relevant features found by the data cleansing algorithm are then processed to create appropriate variables. These variables are created by a set of rules that can process the data and apply transformations to the data (e.g. combine single data points into a new synthesized variable, aggregate data points) to better describe the relevant state of the system.

These processed variables are analyzed by models that are used to discover anomalies that could be indicative of a brewing performance problem. Each model looks for a specific type of

statistical anomalies that could predict a performance problem. No single model is appropriate for a system as complex as a large computing system, especially since the workload profile changes over time. So rather than a single model, ConicIT generates models appropriate to the historical data from a large, predefined set of prediction algorithms. This set of active models is used to analyze the data, detect anomalies and predict performance degradation. The active models vote on the possibility of an upcoming problem in order to make sure that as wide a set of anomalies as possible are covered, while lowering the number of false alerts. The set of active models change over time based on the results of an offline learning algorithm which can either generate new models based on the data, or change the weighting of existing models. The learning algorithm is run in the background on a periodic basis.

When a possible performance problem is predicted by the active models, the system takes two actions. It sends an alert to the appropriate consoles and systems, and also instructs the monitor to collect information from the effected systems more frequently. The result is that when IT personnel analyze the problem they have the information describing the state of the system and the effected system components as if they were watching the problem while it was happening. The system also uses the information from the analysis to point out the anomalies that led the system to predict a problem, thereby aiding in root cause analysis of the problem.

ConicIT Alerts

Accurate alerting is key capability for performance monitoring. The standard mechanism for setting alerts is defining thresholds to define the bounds of expected (or normal) performance of system variables like CPU, dispatching wait time, DB buffer usage or the current number of active DB locks. Once thresholds are breached, the system is no longer acting normally – which is usually indicative of a problem that needs to be attended.

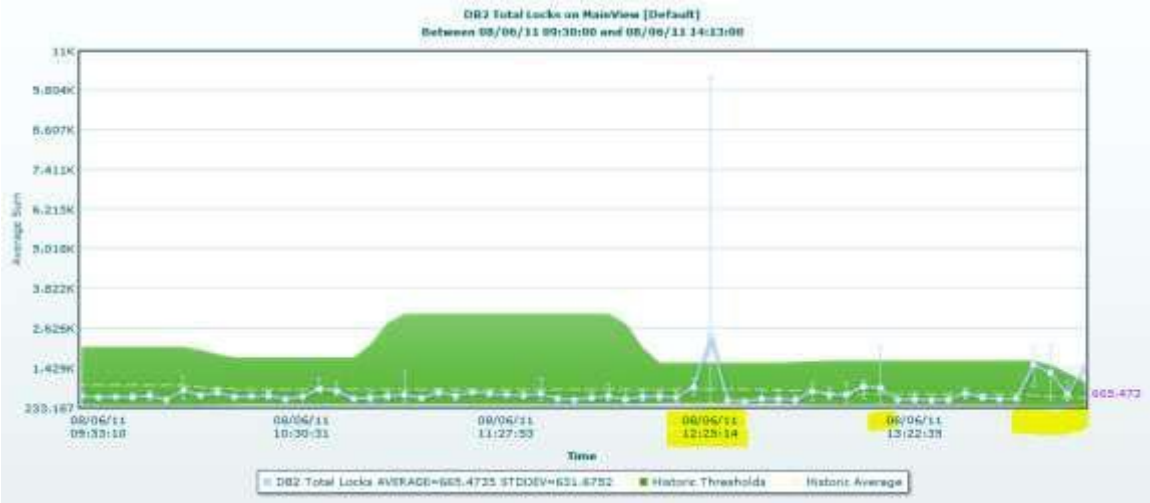
Thresholds are usually defined as fixed thresholds which are easy to define manually, but suffer from over generalization – not taking into account a variable's (values) behavior changes over time. Fixed threshold breaches are a lagging indicator - identifying an extreme failure (like 100% CPU utilization) after it has already happened – making them almost useless for real world performance alerts and management. Using fixed threshold systems leads to either setting the thresholds too low and receiving too many alerts (causing technical staff to ignore these alerts), or setting thresholds too high and missing real problems.

Dynamic thresholds solve the issues associated with fixed thresholds by using a behavior profile describing the dynamic nature of normal behavior (e.g. how a variable is different at different times of the day, days of the week, and special days like the first of the month). Dynamic thresholds provide the basis for accurate real time alerts by providing an unlimited set of context relevant thresholds. The reason that dynamic thresholds aren't in widespread use is that they can't be calculated manually. For example to define the dynamic threshold for variable like DB2-locking-rate or the CPU usage would require at least 8700 thresholds (one per hour) for a

single system for single year. Just consider how difficult it would be to define the same for the hundreds and thousands of values of each system variable, manually.

While dynamic thresholds are much better than fixed thresholds, on their own they still aren't accurate enough for performance alerts. Not every breach of a dynamic threshold is a reason for an alert – that is where performance models come into play. These models understand semantics of system and application performance, and can differentiate between a threshold breach that should generate an alert and a threshold breach that should be ignored in the current context. The best performance models are layered models (e.g. generic performance models, virtualization performance models, mainframe performance models) where the models are combined to provide the most accurate alerts possible for a specific installation.

ConicIT automatically (without any user involvement) calculates dynamic thresholds defining a Dynamic Behavior Profile for each variable, and constantly updates it to adapt to changes in the system. These Dynamic Behavior Profiles define the dynamic thresholds needed for accurate, meaningful real time alert candidates. These alert candidates are then fed into the appropriate models which decide whether an alert should be issued, or if the anomaly can be ignored.



For example: in the graph shown, the green area represents the predicted normal behavior of the number of DB2 locks. Each point on the graph actually represents an average of the number of DB2 locks held during the time the point represents (the vertical line displays the min and max number of locks held). From the graph, it is clear that there is one unusual peak outside of the predicted behavior boundaries at around 12:25 a.m. and two less pronounced peaks (around 13:30 and 14:00). Using a fixed threshold with the threshold set too low would cause too many alerts to be issued (at 12:25, 13:30 and 14:00, which means 66% of the alerts are incorrect), while setting the threshold too high would cause the alert on the real problem to be missed.

ConicIT's dynamic threshold would automatically define different thresholds for 12:00-13:00 and 13:00-14:00 and alert candidates would be generated only when a real change in behavior has occurred which may be indicative of a real problem (i.e. only at 12:25). These alert candidates would be passed the performance models which would decide whether this change in behavior warrants issuing a real alert or not, given the current context.

System Components

As shown in **Figure 1**, ConicIT's system architecture consists of 3 major components:

1. A data cleansing component
2. A rule analysis component
3. An expert system and learning component

Figure 2 shows in more detail how the components interact, and the data flow between the components.

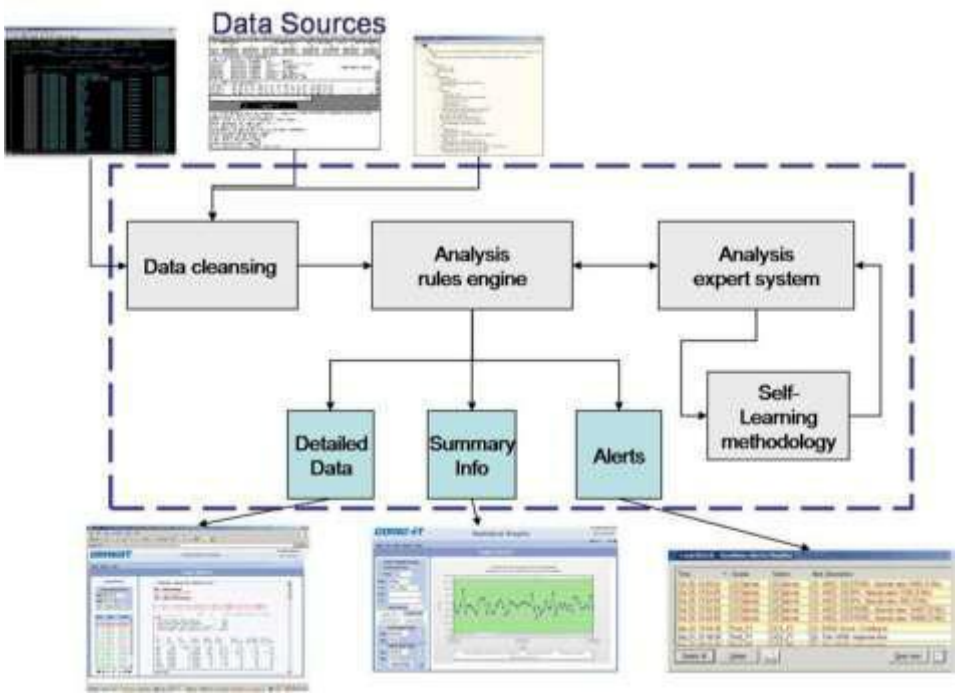


Figure 1: ConicIT Major Component Overview

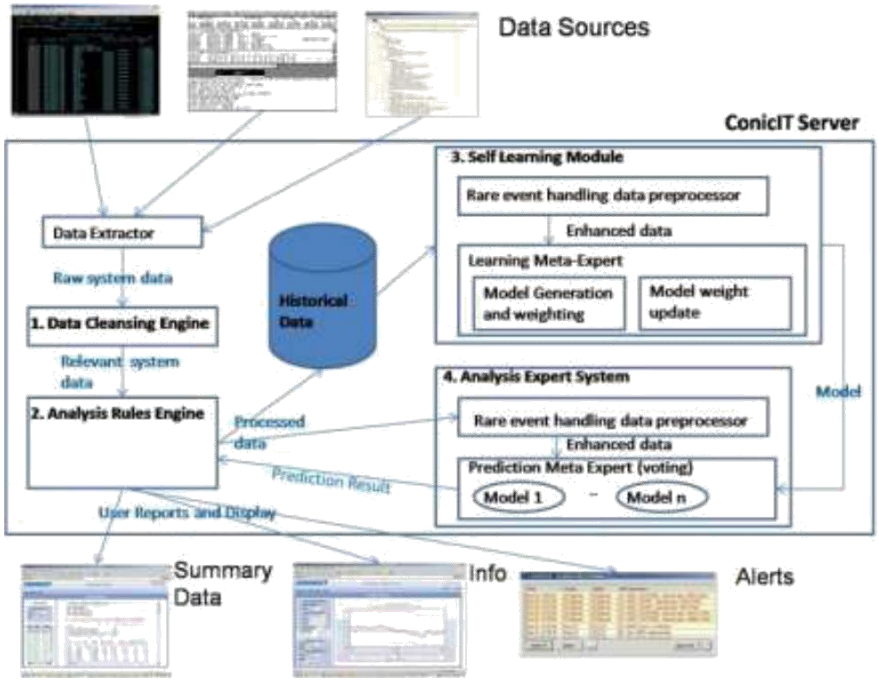


Figure 2: ConicIT Component Detail and Data Flow Overview

Data Extraction and Cleansing (1)

The system interfaces with the system monitors and other data sources through standard (3270) emulation. The systems appears as just another user to the monitors, and requests data from monitors on a regular basis. This data is then parsed into a form usable by the data cleansing component.

This raw stream of data enters the data cleansing component as a set of simple and complex variables obtained by simple preprocessing of the 3270 data stream. The data cleansing algorithm then uses various sophisticated mathematical models to decide what data should be passed the next stage for analysis. The enormous fluctuations in systems data make standard filters and statistical analysis irrelevant. For example, looking at the data in **Graph 1** it is clear that thresholding won't work – it will cause either too few or too many alerts, depending on how the threshold is set. Using the mean is equally uninformative.

ConicIT uses a proprietary data cleansing algorithm that is fast enough to analyze the real time data stream while ensuring that the analysis engine is not swamped with irrelevant data.

Analysis Rule Engine (2)

The analysis rule engine contains the ConicIT performance models. This component takes relevant data provided by the data cleansing component and uses it to create specific variables appropriate for performance analysis. This allows the system to easily support new monitors, since when adding a new monitor only the preprocessing needs to be changed, not the data cleansing component. ConicIT configuration set up also allows for easy mapping between the 3270 data stream and variables needed by expert system, and it also enables the use of multivariate data (creating a synthetic variable that represents the merged behavior of more than just a single measured variable) which enables the models to use correlated variables to find anomalies.

The analysis rule engine is also responsible for data manipulation related to end user needs (e.g. data aggregation for user display) and for creating all of the user notifications and reports. This engine uses the relevant data it obtains from the data cleansing component and results from the expert system to create end user screens, reports and alerts.

It is implemented as rules in an Automatic Rule Generator (ARG) translated to PERL commands and can be easily extended.

Self Learning Module and Analysis Expert System (3,4)

These are the modules that are responsible for the actual prediction of possible performance degradation. They receive as input the processed data (variables) from the analysis rules engine and use it as input for both the prediction and learning algorithms.

The first step for both the expert systems and self-learning module is a rare event preprocessor. Since system anomalies are statistically rare there is a need to process them before they are appropriate for use by the prediction algorithms and models. This component takes the processed data (variables) and transforms it to enhanced data appropriate for the prediction of rare events.

Self Learning Module (3)

The learning mechanism is an offline algorithm that uses historical data to periodically check the prediction algorithms in order to select the most appropriate set of active models for the current state of the monitored system. The self learning module applies the historical data to candidate prediction algorithms to create prediction models and a weight for each model. The set of active models cannot be too large since they need to be calculated in real time against the feature set provided by the data cleansing component.

ConicIT comes out of the box with a large set of prediction algorithms appropriate for most mainframe installations. Achieving optimal performance for a specific installation requires about a week of site specific professional services tailoring and a equivalent automated learning period.

Analysis Expert System (4)

This is the run time component used for predicting possible performance degradation. It uses the current enhanced data, and uses a voting algorithm to generate a score defining the probability of possible service degradation. The voting algorithm examines uses all of the active models and their weights to provide a numerical prediction of performance degradation and it also provide a vector of all the variables that led to the decision (to be used for root cause analysis).

Summary

As opposed to the assumptions of the 2000's - mainframes are here to stay and mainframe usage continues to grow and evolve. Performance issues continue to plague even the most efficient mainframe installations, and the costs associated with mainframe performance issues are of great concern to CIOs. There are more than 200 billion lines of mainframe code running today's businesses worldwide. Fortune 500 companies maintain 35 million lines of COBOL code and add 10 percent more new lines of mainframe code annually just to keep up with business needs, changing conditions, and regulations. These applications still manage and process most customer, product, supply-chain, and critical business data and most are a mix of screen processing, data I/O, and the core of the application—the enterprise's business rules.

The performance and reliability requirements from these mainframe applications are growing while the number of developers who truly know the applications and systems is decreasing. Even though mainframe systems have robust monitoring and management capabilities, mainframe IT personnel are buckling under the strain of supporting all the business IT requirements, and there is a growing gap between the number of mainframe personnel needed and those available. The existing IT personnel need tools that augment existing management and monitoring tools to support them in the analysis and repair of system performance issues in less time and the first time they happen.

ConicIT's technology is uniquely positioned to solve these problems.