Booklet

nemoART

Real User Monitoring
Methode zur Bestimmung der Qualität von IT-Services mit einem Verfahren zum Erkennen von Ursachen durch die Nutzung von Fehlerbildern

Edition 3.5

©Zott+Co GmbH

| Zott+Co GmbH | Viehmarktplatz 6 | D-82418 Murnau | Tel.: +49 8841 6114-0 |
| E-Mail: info@zott.net | http://www.zott.net

1 nemoART gibt Ihnen Antworten

Wollen Sie wissen, warum Ihre User unzufrieden sind - Sie müssen es wissen !
Wir erkennen das typische Kommunikationsverhalten von IT-Services.
Abweichungen davon werden analysiert und bewertet.

Mit Hilfe der Technologie des Real User Monitoring erfolgt eine Evaluation

- von Antwortzeiten
- von IT-Services
- unabhängig vom Protokoll
- in Realzeit

Die aufgezeichneten Metriken werden zur Ermittlung von Symptomen untersucht. Auffällige Antwortzeit relevante Messwerte werden signalisiert, wenn in der Gesamtbetrachtung der Symptome ein klares Fehlerbild erkannt wurde. Über das Fehlerbild wird ein Rückschluß auf Ursachen gezogen.

In der medizinischen Diagnostik werden vergleichbare Verfahren angewendet:
Es werden unterschiedliche Symptome ermittelt, die zusammen ein Krankheitsbild (Fehlerbild) ergeben und den Rückschluss auf die Krankheit (Ursache) zulassen.

Diese Methode eignet sich zum Beispiel dafür, Ursachen aufzuzeigen, wenn schlechtes Antwortzeit-Verhalten bei der Directory-Synchronisation in einer Cloud oder beim Befüllen eines Warenkorbs auftritt.

2 Verschiedene Rollen - verschiedene Herausforderungen

Es gilt, die Diskrepanz zwischen Netzwerkanalyse und Application-Performance-Analyse zu überwinden.

- Netzwerkverantwortliche:
  möchten beweisen, dass die Ursache für schlechte User-Performance nicht im Netz liegt
- App-Verantwortliche:
  möchten beweisen, dass die Ursache für schlechte User-Performance nicht am Applikationsserver liegt
- Systemverantwortliche:
  möchten wissen, wo die Ursache für schlechte User-Performance liegt

nemoART bietet eine gesamtheitliche Betrachtung und gibt Ihnen Antworten auf die Frage nach den Ursachen. Diese können vielfältig sein, z.B.:

- Netz - Performance zum Client
- Netz - Performance zum Server
- Server - Performance

3 Gegenüberstellung unterschiedlicher Monitorplattformen

Virtuelle Clients mit vordefinierten Transaktionen

Test von Transaktionen benötigt Skripte; diese müssen getestet und an geänderte Applikationen/Oberflächen angepasst werden

- sehr wartungsintensiv
- nur lesende Zugriffe können abgebildet werden
- nicht alle Transaktionen werden getestet

Monitoring der kompletten Dialoge

Sehr hohes Datenaufkommen

- dadurch großes Storagevolumen
- Monitoring nur kurzer Zeiträume möglich
- sehr aufwändig zu handhaben
- datenschutzrechtlich bedenklich

- keine integrierte Aus- und Bewertung der Ergebnisse
- keine Klassifizierung der Ereignisse

Nagios Open-Source-Monitoring
Allgemeine Monitoring-Platform, benötigt Client/Server-Infrastruktur

- dadurch Zugriff auf Systeme nötig, die untersucht werden
- Freigabe von Logins, Installation von Agenten nötig
- wartungsintensiv
- protokollabhängig

nemoART Real-User-Monitoring

- Realzeit-Analyse mit integriertem Verfahren zum Erkennen von Fehlerbildern
- Nur Headerinformationen werden abgelegt
  - hocheffiziente Datenhaltung
  - datenschutzrechtlich unbedenklich
- Retrospektive Analyse und Diagnose über lange Zeiträume
- Funktionen zur automatischen
  - Kalibrierung
  - Aus- und Bewertung von Anomalien und Ergebnissen
  - Klassifizierung von Ereignissen
- Protokollunabhängig

4 nemoART Technologie

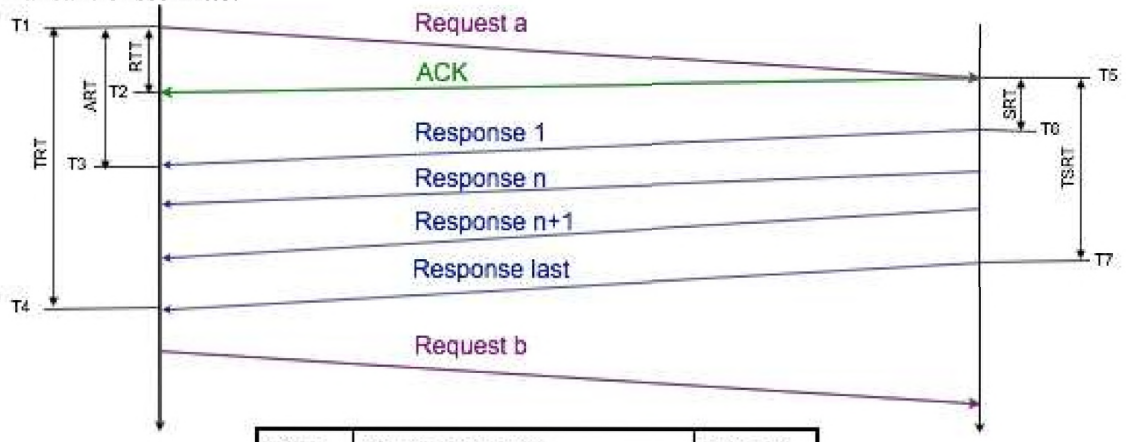
- Die Erfassung der Metriken erfolgt vollkommen unabhängig von der Auswertung: Metriken werden an Interfaces von Servern erfasst, an denen nemo-Datenkollektoren installiert sind.
- Die Aus- und Bewertung dieser Metriken wird durch unterschiedliche Standard-Auswertungen als Actions, in Form von Stored-Procedures (Analyseprogramme in der Datenbank) durchgeführt.
- Auch applikationsspezifische Auswertungen können integriert werden.

- Externe ITSM-Systeme wie z.B. BMC-Remedy oder Nagios werden mit den Ergebnissen bedient.
- Die integrierte Web-Oberfläche ermöglicht umfangreiche Analysen und Diagnosen.

4.1 Transaktionsschritte

Eine Transaktion, in der Terminologie von nemoART, ist eine Abfolge von Kommunikations-Operationen, die durch Requests und Responses repräsentiert werden.

Transaktionsschritte:



RTT	Round-Trip-Time	T2 - T1
ART	Application-Response-Time	T3 - T1
TRT	Total-Response-Time	T4 - T1
SRT	System-Reaction-Time	T6 - T5
TSRT	Total-System-Reaction-Time	T7 - T5

Die von nemoART zu analysierenden und zu signalisierenden Metriken sind ART/TRT/RTT

4.2 Analyse-Methode

- nemoART arbeitet nach dem Ursache-Wirkung-Prinzip:
  Auffälligkeiten aus Antwortzeit relevanten Metriken (Symptome=Wirkung) werden ermittelt und signalisiert, wenn in der Gesamtbetrachtung aus diesen und verschiedenen anderen Faktoren ein klares Fehlerbild erkannt wurde. Über das Fehlerbild wird ein Rückschluß auf Ursachen gezogen.

- Auf Basis der Evaluations - Ergebnisse können über die Expertenoberfläche über einen gewünschten Bereich eine Detailanalyse oder eine Serverzugriffsanalyse mit Informationen zur legalen und illegalen Nutzung von IT-Services durchgeführt und Diagnoseunterlagen erstellt werden.

- Der gewünschte Bereich kann aufgrund eines hocheffizienten Datenhaltungskonzepts auch über Wochen/ Monate rückwirkend gewählt werden.

4.3 Analyse von Metriken

- Ein hoher Wert bei einer Metrik macht noch keine Unregelmäßigkeit und selbst wenn es eine sein sollte, hat man kaum eine Chance diese zu finden

- nemoART ermittelt permanent eine ganze Reihe von Metriken, von denen ART (Applikation Response Time), TRT (Total Response Time) und RTT (Round Trip Time), die vom Benutzer gefühlte Antwortzeit reflektieren

- Wird bei einer dieser drei Metriken (ART/TRT/RTT) der Schwellwert überschritten, analysiert nemoART automatisch alle anderen relevanten Metriken und erzeugt einen qualifizierten Alert, wenn in der Gesamtbetrachtung ein klares Fehlerbild erkannt wurde

- Da die Schwellwerte zu ART/TRT/RTT bewirken, daß die Überschreitung einer Zeitmetrik zur Signalisierung führt, werden sie justiert zu jedem Untersuchungs-Objekt. Das geschieht automatisch und treffsicher.

4.4 Signalisierung von Ursachen nach erkannten Fehlerbildern

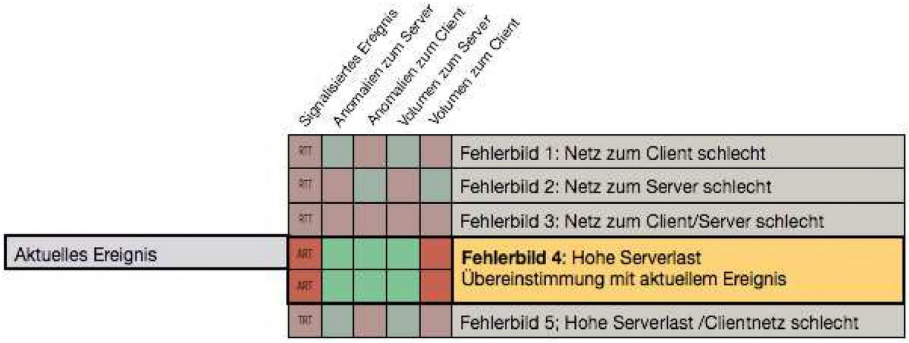
- Die Signalisierung basiert auf dem Überschreiten von vordefinierten Schwellwerten von Antwortzeit-Indizes. Die Signalisierung ist hierbei das Vergeben des Status Warning/Critical für die Objekte/Connnections und die auslösende Action.

- Parallel zu den Zeitmetriken (ART/TRT/RTT) werden permanent Ereignisse, Anomalien und Volumen-Informationen erfasst
- Auffällige Metriken (Symptome) werden als Indikatoren zur Bewertung von Ereignissen benutzt

- Diese Indikatoren werden analysiert und mittels definierter Zustände eine Klassifizierung vorgenommen
- Aus der Kombinationen dieser Indikatoren werden Muster gebildet, die zur Identifikation von Alerts, die zu einer Signalisierung geführt haben, verwendet werden

4.5 Methode zur Identifikation von Fehlerbildern

- Um das Ereignis, das zu einer Signalisierung geführt hat, einer Klasse zuordnen zu können, wird aus den einzelnen Indikatoren ein Muster (Fehlerbild) gebildet, das in den bekannten Indikatorklassen (Fehlerbildern) gesucht wird und bei Übereinstimmung identifiziert werden kann.
- Wenn ein Muster keiner Klasse zugeordnet werden kann, wird daraus eine neue Klasse definiert.
- Dadurch ist nemoART in der Lage, in der Signalisierung konkrete Hinweise auf Zustände in Objekten (Connections gruppiert in logische Einheiten [z.B. Services]) zu liefern, für die das nemoART-Monitoring aktiv ist.



5 Vorteile von nemoART

- Mit Hilfe einer permanenten, lückenlosen und flächendeckenden Netzwerk-Performance-Analyse der gesamten Kommunikation (protokollunabhängig) werden auffällige Systemreaktionszeiten erkannt und deren Ursachen analysiert
- Plug und Play Appliance mit integrierter MariaDB Datenbank
- Aufgrund eines hocheffizienten Datenhaltungskonzepts ist auch eine Langzeitdiagnose über Wochen/ Monate möglich

6 nemoART's Nutzen

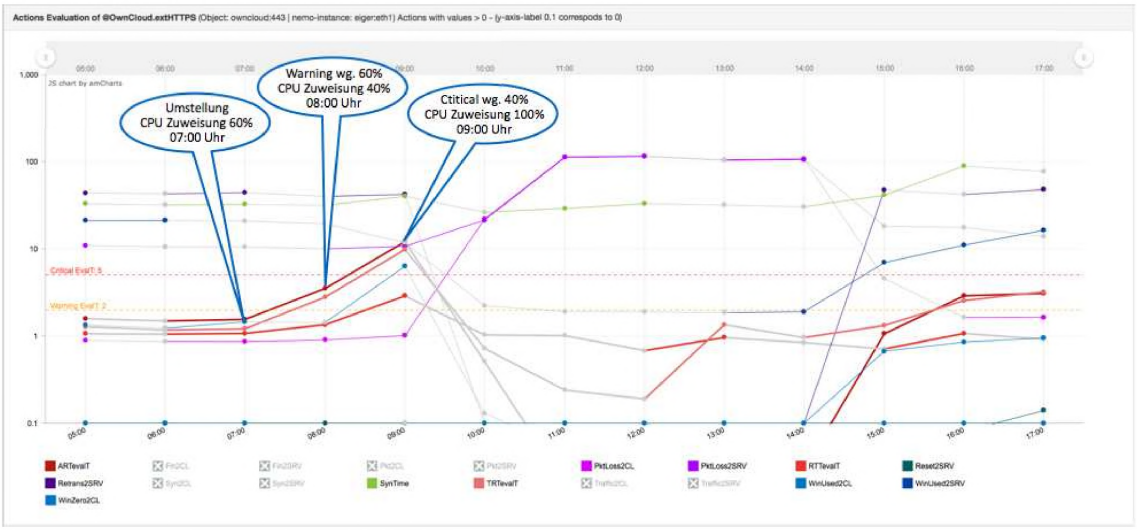
- Das Response-Time-Tracking liefert Systemreaktions-, Netzlauf- und Antwortzeiten und erkennt Anomalien in der Kommunikation.Diese Symptome werden analysiert, Fehlerbilder werden erstellt und daraus Ursachen erkannt.
  nemoART läuft passiv, flächendeckend im realen Betrieb mit und benötigt keine Agenten auf Clients oder Servern.
- Das SLA-Tracking versetzt sowohl Betreiber von Services als auch die Nutzer in die Lage, auch rückwirkend, zu analysieren, wann, in welchem Umfang und mit welcher Qualität die zur Verfügung gestellten Services genutzt werden konnten.
- Über das Access-Tracking erhält man Informationen, wer, von wo, wie und wann auf die IT Services zugreift. Damit können Quellen und Umfang von Attacken analysiert und rückwirkend verfolgt werden.

7 Fallbeispiel

Die einzelnen Services wie z.B. die Directory-Synchronisation in einer Cloud oder HTTP/HTTPS, über das z.B. auch das Befüllen eines Warenkorbes abgehandelt wird, werden anhand des Kommunikationsverhaltens auf Auffälligkeiten überprüft.

Die Verbindungen zu einer als 'ownCloud' konfigurierten Cloud waren zunächst völlig unauffällig. Dann wurden in einer HTTPS-Verbindung zu dieser Cloud gezielt Auffälligkeiten provoziert durch die Zuweisung geringerer CPU Nutzung zu bestimmten Zeiten:

CPU Zuweisung 60% um 07:00 Uhr	=>	Warning um 08:00 Uhr
CPU Zuweisung 40% um 08:00 Uhr	=>	Critical um 09:00 Uhr
CPU Zuweisung 100% um 09:00 Uhr	=>	Verbindung ab 10:00 Uhr unauffällig
CPU Zuweisung 60% um 15:00 Uhr	=>	Warning ab 16:00 Uhr



Graphische Darstellung des Ablaufes

Bei jedem Critical Zustand einer Verbindung werden bei entsprechender Konfiguration folgende E-Mails versandt:

NR	Date	AlertType	Object	Connection	Action	Value	Reference	Rating	Warning	Critical	Last 24 Hours
7	19.02.16/09:00:00	SRVPerf	owncloud:443	@OwnCloud.extHTTPS	TRTevalT	491,24	50	9,82	2	5	09 08 07 06 05 04 03 02 01 00 23 22 21 20 19 18 17 16 15 14 13 12 11 10
8	19.02.16/09:00:00	SRVPerf	owncloud:443	@OwnCloud.extHTTPS	ARTevalT	481,72	40	12,04	2	5	09 08 07 06 05 04 03 02 01 00 23 22 21 20 19 18 17 16 15 14 13 12 11 10

Signalisierung des Critical Alerts um 09:00 Uhr

'SRVPerf': Server-Performance-Auffälligkeit, die im vorliegenden Beispiel durch Manipulation an der CPU-Nutzung entstanden ist