

Dynatrace Diagnostics: Performance-Management und Fehlerdiagnose vereint

Der Pfadfinder

Frank Rometsch, Horst Sauer

Linux oder Windows, Java oder .Net, Open-Source- oder kommerzieller Applikations- und Datenbankserver – bei heutigen Geschäftsanwendungen ist nahezu alles vertreten. Das erschwert die Test- und Diagnostizierbarkeit solcher Systeme erheblich. Dem will Dynatrace mit Diagnostics begegnen.

Betrachtet man die Aufgabe, die Performance und Stabilität einer unternehmenskritischen verteilten Multi-Tier-Applikation sicherzustellen, muss man einerseits den Nachweis über die Einhaltung von Service Level Agreements (SLAs) erbringen, andererseits bei einer Störung schnell reagieren, um die Ursachen zu finden und die Fehler zu beheben. Meist spielt sich das auf einer heterogenen Server-Farm ab, deren Architektur und Dynamik kaum jemand bis auf die Applikationsebene hinauf kennt. Hinzu kommt, dass

sich die Rekonstruktion des Fehlers in den Entwicklungsabteilungen oft als schwierig erweist und meist mehrere Teams mithelfen müssen, was zu unnötig langen Bearbeitungszeiten führt.

Das im Jahr 2005 in Linz gegründete Unternehmen Dynatrace verspricht, mit Diagnostics sowohl das Performance Monitoring im 24 x 7-Betrieb als auch Fehlerdiagnosen durchführen zu können und den gesamten Zyklus erheblich zu beschleunigen. Den Kern und das Alleinstellungsmerkmal des Werkzeugs bildet

Purepath, eine Technik, mit der der Diagnostiker Business-Transaktionen über Server-, Applikations-, Netz- und Betriebssystemgrenzen hinweg ohne Performance-Einbußen protokollieren und wenn nötig bis auf Code-Ebene hinunter verfolgen kann. Die Frage, wie weit das geht, zu beantworten ist Gegenstand einiger Praxistests mit der Version 2.6.0.

Agenten für Clienten mit eigenem Server

Dreh- und Angelpunkt bilden die sogenannten „Diagnostics Agents“, die auf den zu beobachtenden Systemen installiert sein müssen und dort nach der Platzierung der sogenannten „Knowledge Sensors“ in der Applikation deren Rohmessdaten erfassen. Der „Diagnostics Server“ sammelt die Daten aller Agenten, bereitet sie auf und stellt Berechnungen an. Die „Diagnostics Clients“ schließlich bieten eine grafische Oberfläche, auf der die Konfiguration aller Dynatrace-Komponenten und die Visualisierung der Messdaten erfolgen. Agenten, Server und Clients kommunizieren über einen konfigurierbaren TCP/IP-Port, was ein flexibles Deployment und einen verteilten Betrieb auch über das Internet erlaubt. Zusätzlich ist ein Repository für die persistente Speicherung von Messdaten im Lieferumfang enthalten. Alle Komponenten gibt es als Version für Windows und Linux, den Server außerdem für Solaris.

Diagnostics lernt Anwendungsverhalten

Erfreulich einfach geht das Installieren von der Hand, denn für alle Komponenten liegen Installationsroutinen vor. Im Falle der Agenten richten die Prozesse auf den zu beobachtenden Systemen eine einzige .dll bei Windows beziehungsweise eine .so-Datei bei Unix oder Linux ein. Über die Profiler- oder Tool-Schnittstellen der virtuellen Maschine – JVM für Java, CLR für .Net – klinken sich die Agenten mit ihren Sensoren in die jeweilige Laufzeitumgebung ein und erhalten fortan Informationen über alle relevanten Ereignisse der unter der VM laufenden Applikationen sowie über den internen Status der VM selbst (Abbildung 1).

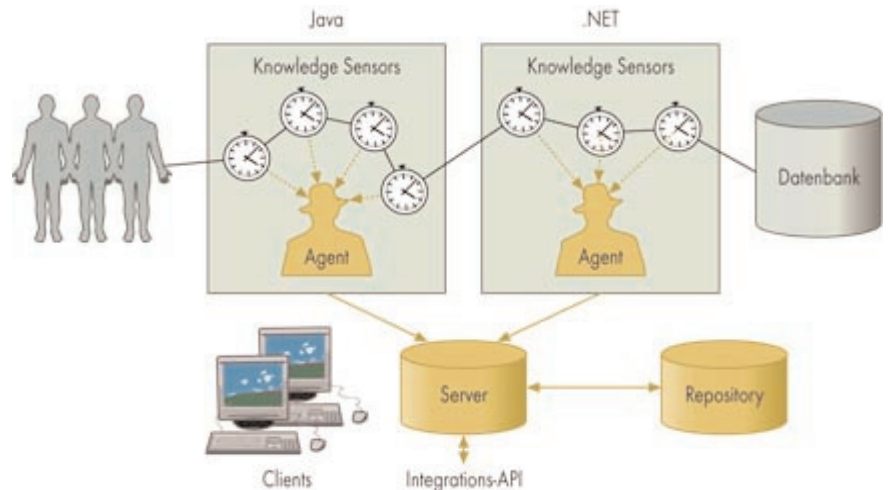
Der Systemingenieur verbindet jeden Agenten, identifiziert über einen

eindeutigen Namen, mit einem Diagnostics Server, an den er die Messdaten liefern soll. Dem ersten Start der Applikation unter „Beobachtung“ von Dynatrace, dem „Discovery Run“, kommt besondere Bedeutung zu: Beim ersten Durchgang „lernt“ Diagnostics die Klassenstruktur und weitere Metainformationen on the fly. Deshalb reagiert die Applikation anfangs schwerfälliger als vorher. Um spätere Messungen nicht durch den Lernvorgang zu verfälschen, sollte man dafür sorgen, dass der Discovery Run möglichst viele Anwendungsfälle und damit die meisten Ausführungspfade durchläuft.

Jetzt kann der Verantwortliche konfigurieren, welche Methoden er beobachten will. Beim „Sensor Placement“ sucht der Agent die ausgewählten Methoden gezielt aus. Dazu fügt er nach Bedarf Instrumentierungs-Code ein, um für jeden Aufruf Daten wie Laufzeit, CPU-Verbrauch, Kontext-Informationen, Methoden-Argumente, Aufrufer oder Exceptions zu protokollieren.

Den dadurch entstehenden Performanceverlust beziffert Dynatrace mit 3 bis 5 %; er kann aber in der Praxis erheblich darüberliegen, denn er hängt von der Laufzeit einzelner Methoden und deren Aufrufhäufigkeit ab. Kurze Methoden, die zu unverhältnismäßigem Instrumentierungs-Overhead führen würden, kann man per Konfiguration ausschließen, für Getter- und Setter-Methoden gibt es sogar eine eigene Option. Damit lassen sich die Einbußen in der Regel auf ein erträgliches Maß reduzieren.

Mit der im Produkt enthaltenen Technik Purepath kann der Analytiker einen frei wählbaren Einstiegspunkt, etwa einen Webservice oder einen normalen Methodenaufruf nebst Trace-Punkten als Sensoren im Gesamtsystem definieren. Purepath erfasst Transaktionen vom Einstiegspunkt durch alle betrof-



Dynatrace Diagnostics besitzt eine einfache, aber dennoch effiziente Struktur (Abb. 1).

fenen Schichten hindurch (siehe Abbildung 2 und Aufmacher).

Virtuelle Maschinen mit Anschlüssen

Recht früh haben die Entwickler der Java Virtual Machines (JVM) die Notwendigkeit einer standardisierten Schnittstelle erkannt, an der Informationen für Entwicklungs- und Monitoring-Tools bereitstehen, etwa für Debugger, Coverage-Tools oder Profiler. Bis Java 1.4 waren es das JVM Profiling Interface (JVMPPI) und das JVM Debug Interface (JVMDI), die ab Java 5 zum JVM Tool Interface (JVMTI) zusammenwuchsen. Entsprechende Schnittstellen sind für Microsofts Common Language Runtime (CLR) vorgesehen. Einstiegspunkte bieten die Profiling-API (insbesondere *ICorProfilerInfo* und *ICorProfilerCallback*) sowie weitere, beispielsweise die Debug- und die MetaInfo-API, die den Zugriff auf weitere Interna der CLR erlauben.

Über die Schnittstellen kommt man an die Zustandsinformationen der virtuellen Maschine heran und erhält die Kontrolle über die dort laufenden Applikationen. Der Ablauf ist einfach: Ein Tool meldet sich als Agent am Interface an und erhält so im Kontext der Applikation Informationen über die gewünschten Ereignisse. Events erzeugt die VM für alle relevanten Laufzeitergebnisse, für den Start und Stopp, das Laden von Klassen, Ein- und Austritte von Methoden, Exceptions, für alle Speicheroperationen wie Heap Allocations oder Objektfreigaben durch den Garbage Collector sowie für das Starten und Stoppen von Threads.

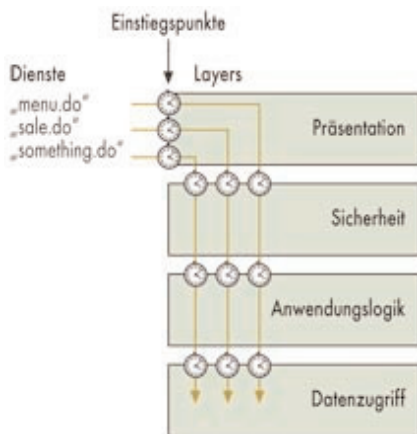
Die Benachrichtigung über Ein- und Austritt einer Methode führt allerdings zu einem hohen Performance-Schwund, vor allem weil die VM für alle Methodenaufrufe bis in Basisklassen hinein ein solches Event generiert. Diesen Overhead bekommt man nur durch gezielte, bedarfsorientierte Methodeninstrumentierung in den Griff. Der richtige Zeitpunkt dafür ist das Laden einer Klasse in den Speicher beziehungsweise der Aufruf des Just-in-time-Compilers.

Inzwischen zählt die Bytecode-Instrumentierung zu den festen Bestandteilen der Tool-APIs für VMs, nicht nur bei JVM und .Net. Bei Java stellt die „Byte Code Engineering Library“ einen Quasi-Standard dar, auf die diverse Profiler, Tracing-Tools und AspectJ zurückgreifen. Microsoft bietet für seine CLR, als Teil der *ICorProfilerInfo*-Schnittstelle, Methoden zur Instrumentierung der Intermediate Language (IL).

Als Testumgebung dienen typische 3-Tier-Architekturen, wie sie im Unter-



- Dynatrace Diagnostics bietet dem Anwender Funktionen sowohl zum Performance-Management als auch zur Fehlerdiagnose.
- Dabei kann Diagnostics den Datenfluss über mehrere Server hinweg sowohl durch Java- als auch durch .Net-Applikationen verfolgen.
- Dazu installiert Diagnostics sogenannte Agents auf den infrage kommenden Systemen, die über ihre Knowledge Sensors in den Applikationen Rohdaten erfassen.
- Über die grafische Oberfläche des Diagnostics Client kann der Anwender die Komponenten konfigurieren sowie die vom Diagnostics Server grafisch und statistisch aufbereiteten Daten analysieren.



Der wahre Weg: Purepath durchläuft die typischen Schichten einer Anwendung (Abb. 2).

nehmensumfeld oft vorzufinden sind. Die Nutzer greifen über Webclients auf die eigentliche Anwendung zu, die aus einem verteilten Web- und Applikationsserver besteht. Oft sind zusätzlich Datenbanken, SAP oder Legacy-Systeme in die Anwendung integriert. Das lässt erahnen, wie heterogen die Umgebungen sind und vor welche komplexen Aufgaben das die Systemadministration stellen kann.

Um Diagnostics erproben zu können, enthielt die Testumgebung typische Störfälle etwa bedingt durch ungünstige Konfigurationen der Anwendungen (zu kleine Pools und Verbindungen), der Software (Thread-Synchronisation, CPU-intensive Methode) und des Netzes (hoher Latenz). Es kamen zwei Architekturen mit passenden Anwendungen für den Test zum Einsatz:

- Die von Microsoft im MSDN für Entwickler angebotene .Net-Anwendung Pet Shop 3.0, bei der alle drei Tiers auf einem Server installiert waren. Der Zugriff der Benutzer erfolgte über Webbrowser.
- Die J2EE-Anwendung Day Trader, verteilt auf einen Applikations- und Webserver unter Linux sowie auf einen MySQL-Datenbankserver unter Windows 2003. Als Clients dienten nativ in Java oder .Net implementierte Programme unter Windows XP.

Bei der Durchführung eines Lasttests der Anwendungen im Unternehmensumfeld simuliert man eine große Anzahl paralleler Client-Zugriffe und misst Größen wie Antwortzeiten und Transaktionsdurchsatz. Die dazu verwendeten Hilfsmittel, beispielsweise HPs Loadrunner, Empirix eLoad oder Borlands Silkperformer, führen dazu Blackbox-Tests durch. Ist die Leistung

des Systems nicht zufriedenstellend, geben sie keinen Hinweis auf die Ursachen, ein Monitoring der Komponenten CPU, Disk- oder Netzwerkauslastung der beteiligten Rechner ist damit aber fast immer realisierbar. Da Diagnostics in der vorliegenden Version 2.6.0 (noch) keine Systemmonitoring-Daten liefert und sich auf die Applikationen beschränkt, wäre es hilfreich, die Daten beider Werkzeugarten kombinieren zu können, damit man einen optimalen Gesamtüberblick bekommt. Dynatrace Diagnostics bietet ebenso wenig wie alle anderen kommerziellen Tools hierfür Schnittstellen an. Somit muss man in der Analysephase immer wieder hin- und herspringen.

Einstiegspunkt per Lastanalyse finden

Einen ersten Gesamtüberblick der Durchsatzraten, Antwortzeiten und Systemauslastung aus Nutzersicht liefert ein Lasttest-Werkzeug. Auf Basis dieser Daten rückt man den Engpässen mit Dynatrace Diagnostics zu Leibe. Im Falle der Webanwendung im Test ist der Einstieg über den View „Tagged Web Requests“ naheliegend, in dem in einer tabellarischen Ansicht die „Einstiegspunkte“ der Clients in die Anwendung aufgelistet sind. Hat man in seine Testskripte passende Tags für den Loadrunner eingebaut, tauchen sie unter ihren Namen in den in Diagnostics gelisteten Webabfragen wieder auf.

Zu jedem Request findet man einige Messwerte wie die übertragene Datenmenge, CPU-Zeiten oder insgesamt dafür verbrauchte Zeit. Das reicht oft, eine typische Schwäche im System wie

inakzeptable Antwortzeiten für den Nutzer zu entdecken. Für die weitere Analyse bis hinunter auf die Codeebene kommt Purepath ins Spiel. Der Analytiker kann an jedem Punkt entscheiden, ob er weiter ins Detail gehen (über Method-Breakdown beziehungsweise Purepath) oder, falls er beim Detail angekommen ist, die Summe über alle Methoden-APIs auf dieser Ebene bekommen will.

Im Test fielen die veränderten Methoden der Applikationen sofort auf, die Dinge simulieren sollten wie rechenintensive Prozeduren oder mangelnde Synchronisation. Hat man nach erfolgreicher Identifikation eine potenzielle Verbesserung im Code vorgenommen, kann man in den Darstellungen des Diagnostic Client Testläufe miteinander vergleichen. Softwarearchitekten erkennen leicht, an welchen Stellen es tatsächlich messbare Veränderungen gab. Dabei hilft ihnen Diagnostics, indem es ihnen den direkten Sprung an die entsprechende Stelle im Quellcode der aufgeführten Packages, Klassen oder Methoden erlaubt, sofern die Entwickler Eclipse oder Visual Studio genutzt haben.

Für Auswertungen etwa mit ANOVA oder DoE (Design of Experiment) aus der „Six Sigma Welt“ fehlt eine Exportfunktion für die Einzelmessungen zur Analyse mit Programmen wie Minitab oder Fathom. Bis dato müssen sich Statistiker mit im 10-Sekunde-Raster generierten Min-, Max- oder Mittelwerten begnügen. Exportieren kann man in Diagnostics über CSV und HTML oder die JMX-Schnittstelle.

Weil der Analyseschwerpunkt von Dynatrace auf der Softwareseite liegt – was in Entwicklungsabteilungen si-

Editionen

Dynatrace Diagnostics

Workstation Edition: erlaubt die Analyse des Applikationsverhaltens einer Anwendung während der Softwareentwicklung in der lokalen Entwicklungsumgebung.

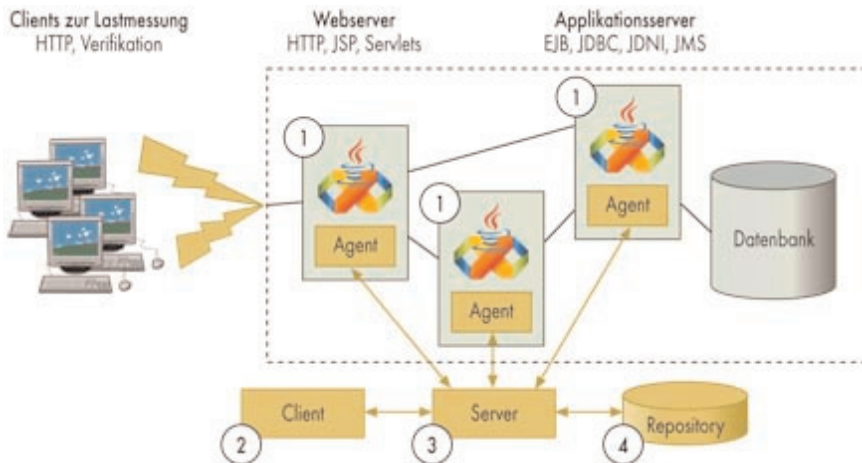
Professional Edition: bietet zusätzlich die Überwachung und Diagnose von zwei produktiven Applikationen tagtäglich rund um die Uhr.

Enterprise Edition: Überwachen, Diagnostizieren und Performance-Management sämtlicher Applikationen im Betrieb inklusive Java-Anwendungen auf dem Mainframe, Zugriff auf Diagnosedaten und Aktionen unternehmensweit.

Preise: auf Anfrage, www.dynatrace.com

⚡-Wertung

- ⊕ Echtzeitanalyse von Business-Transaktionen mittels Purepath
- ⊕ systemübergreifender Einsatz
- ⊕ Analyse im gesamten Entwicklungszyklus
- ⊖ .Net-Unterstützung hinkt der von Java hinterher
- ⊖ keine Einbindung von älteren und nativ implementierten Komponenten
- ⊖ beschränkte Exportfunktion der Messdaten



Testaufbau: Die an den Messpunkten (1) von den Agenten gesammelten Daten erscheinen vom Server (3) grafisch aufbereitet auf dem Client (2) und können im Repository (4) hinterlegt werden (Abb. 3).

cherlich gefragter ist – ließen sich andere im Test bewusst eingebaute Bremsen, etwa eine limitierte Anzahl von gleichzeitig erlaubten Browser-Verbindungen zum Webservice, besser durch die Analyse mit dem Lasttest-Tool identifizieren. Das hat Dynatrace wohl erkannt und für die kommende Version 3.0 Erweiterungen für das System-Monitoring angekündigt.

In den Tiefen der Applikation

Ob man als Performance Engineer Bottlenecks identifizieren will oder als Entwickler auf Fehlersuche ist, man profitiert von Diagnostic. Aber auch Systembetreuern hilft es, wenn sie das dynamische Verhalten eines in Betrieb befindlichen Systems beobachten, verstehen und verifizieren wollen. Sie erhalten durch die Echtzeit-Analyse eine Unterstützung, bei der die Messwerte des „System under Diagnose“ (SUD) mit geringer Verzögerung ständig beim Diagnostics Client eintreffen. JMX erlaubt für einige Messvorgänge die Anbindung an Frameworks wie IBMs Tivoli oder CA Unicenter.

Als Einstieg empfiehlt sich ein Discovery Run, bei dem die Applikation möglichst realistische Anwendungsfälle durchläuft. In typischen Enterprise-Architekturen kann man mit dem mitgelieferten Sensor Pack von Dynatrace, das standardmäßig alle gängigen Techniken unterstützt, Analysen durchführen. In der Grobarchitektur Server, Applikationen und Software-Layer tritt über die aufgezeichneten Purepaths das dynamische Zusammenspiel der

Komponenten/Layer und deren Kommunikationsschnittstellen zutage.

Damit erlangt man in kürzester Zeit einen Überblick, selbst bei komplexen oder wenig bekannten Architekturen. Der Schritt kann zu überraschenden Erkenntnissen vor allem bei besonders fantasievollen Implementierungen führen. Positiv fällt auf, dass Systemgrenzen oder Herstellereigenheiten weitgehend transparent bleiben und Dynatrace Tools die Analyse von Transaktionen nicht behindern.

Wer tiefer in die Applikation eintauchen will, kann im Dynatrace-Client eigene erweiterte Konfigurationen vornehmen und auf dem Server persistent hinterlegen. Der Anwender der Werkzeuge darf im während des Discovery Run „erlernten“ Klassenbaum auf Methodenebene Sensoren an- oder abwählen.

Alternativ zum eben beschriebenen Weg „Outside-in“, unterstützt Dynatrace ein „Inside-out“: Ausgehend von Exceptions oder identifizierten Engpässen auf Methodenebene, kann man die zugehörigen Business-Transaktionen identifizieren und damit die eventuell finanzielle Auswirkung verstehen.

Obwohl Dynatrace alle gängigen Techniken und Hersteller unterstützt, stößt man im praktischen Einsatz an Grenzen, insbesondere wenn ältere oder nativ (in C/C++) implementierte Komponenten wesentlicher Bestandteil des Systems sind. Analysen von Vorgängen auf dem Datenbankserver (etwa bei Stored Procedures) oder dem SAP-Backend sind nur über proprietäre Tools durchführbar. Diagnostics erfasst aber immerhin die Aufrufsstelle einschließlich ihrer Parameter,

was für eine erste Analyse ausreichen dürfte.

Trotz der breiten Zielgruppe, die Dynatrace ansprechen möchte, bleibt die Benutzbarkeit nicht auf der Strecke. Sowohl das Konfigurieren als auch das Erfassen und Auswerten von Messwerten nebst Traces ist intuitiv durchführbar und erlaubt flüssiges Arbeiten. Bei komplexen Systemen kann man zwar durch die Vielzahl der Fenster leicht die Übersicht verlieren, kommt aber über das Cockpit schnell wieder zurück zum Ausgangspunkt, egal ob das ein Purepath, ein Web Request oder eine Exception war. Ohne Hindernisse funktionierte das verteilte Arbeiten mit Dynatrace, das gerade im Enterprise-Umfeld ein Muss ist. Der Client regiert auch aus der Ferne selbst über schmalbandige Verbindungen schnell genug. Daten stehen sowohl online als auch „aus der Konserve“ offline zur Analyse zur Verfügung. Das ist besonders praktisch, wenn die Auswertung durch ein anderes Team erfolgt oder sogar über mehrere Zeitzonen geht.

Fazit

Wer jemals versucht hat, in verteilten Systemen ein rudimentäres Messsystem zu implementieren, das es erlaubt, die beteiligten Komponenten bei Performance- und Stabilitätsproblemen synchronisiert für eine Auswertung unter einen Hut zu bekommen, für den dürfte Dynatrace Diagnostics eine spürbare Erleichterung sein. Auch wenn es andere spezifische Tools nicht vollkommen ersetzen kann, bietet es wegen seiner Systemunabhängigkeit und dem breiten Anwendungsspektrum vor allem für Teams eine passende Analysegrundlage. Bei der Untersuchung der Anwendung erlaubt Dynatrace Purepath-Technik einen tiefen und frei wählbaren Einblick. (rh)

FRANK ROMETSCH

ist Senior Engineer bei Siemens Corporate Technology. Seine Themenschwerpunkte sind Softwaretest und Performance Engineering.

HORST SAUER

ist Senior Engineer bei Siemens Corporate Technology. Seine Themenschwerpunkte sind Softwaretest, Testbarkeit und Diagnose.

