

Dynatrace Diagnostics: Performance management and troubleshooting combined

The Pathfinder

Frank Rometsch, Horst Sauer

Linux or Windows, Java or .Net, Open-Source or commercial application and data base server - you can find almost everything in today's business applications. That makes the testability and diagnosability of such system extremely difficult. This is what Dynatrace wants to help you overcome.

If you are tasked with guaranteeing the performance and stability of a business-critical distributed multi-tier application, you must establish that SLAs (Service Level Agreements) are observed on one hand, and on the other ensure that a quick response to failure is possible to find its causes and eliminate the errors. Most of the time, this occurs on a heterogeneous server farm, whose architecture and dynamics few understand on the application level.

What's more, the reconstruction of errors by development is often very dif-

ficult. Often, several teams have to assist, which causes unnecessarily long delays in handling time.

Dynatrace, founded in Linz, Austria, in 2005, provides 24x7-operation performance monitoring as well as troubleshooting with Dynatrace – and thus enables considerable acceleration of the entire application lifecycle. The core and stand-alone characteristic of the tool is Purepath, a technology with which a diagnostician can record business transactions without any performance loss, beyond the limits of

servers, applications, networks and operation systems, down to code level if required.

How can this be realized? This is the focus of some practical tests of dynaTrace version 2.6.0.

Agents for clients with their own server

Central point are the so-called „Diagnostic Agents“ which must be installed on the monitored systems and which record raw application data after the placement of so-called „Knowledge Sensors“ in said application. The Diagnostics server collects the data of all agents, processes them and makes calculations. And finally the „Diagnostics Client“ offers a graphical interface on which the configuration of all Dynatrace components and the visualization of the measured data are carried out. Further, agents, server and clients communicate via configurable TCP/IP-port which allows for flexible deployment and distributed operation via internet. Additionally, dynaTrace offers a repository for the persistent storage of measured data. All components are available as Windows and Linux versions; the server is also available for Solaris.

Diagnostics learns application behavior

Installation is pleasantly simply as there are installation routines for all components. Regarding agents, the processes on the systems to be monitored create a single .dll for Windows, respectively one .so-file for Unix or Linux. The agents log onto the runtime environment with their sensors via profiler or tool interface of the virtual machine - JVM for Java, CLR for .Net – and receive information on all relevant results of applications running in the virtual machine as well as about the internal state of the virtual machine itself (figure 1).

The system engineer connects all agents, identifiable by their unique names, with a Diagnostic Server to which the measured data is sent. When starting the application under „Monitoring“ by Dynatrace for the first time, the „Discovery Run“ has a special meaning: During the first run, Diagnostics „learns“ the class structure and other meta information on the fly. This is

why the application is much slower at first (especially with .Net) than it was before. To prevent any falsification caused by the learning process, care should be taken that Discovery Run runs through as many application cases and thus as many execution paths as possible.

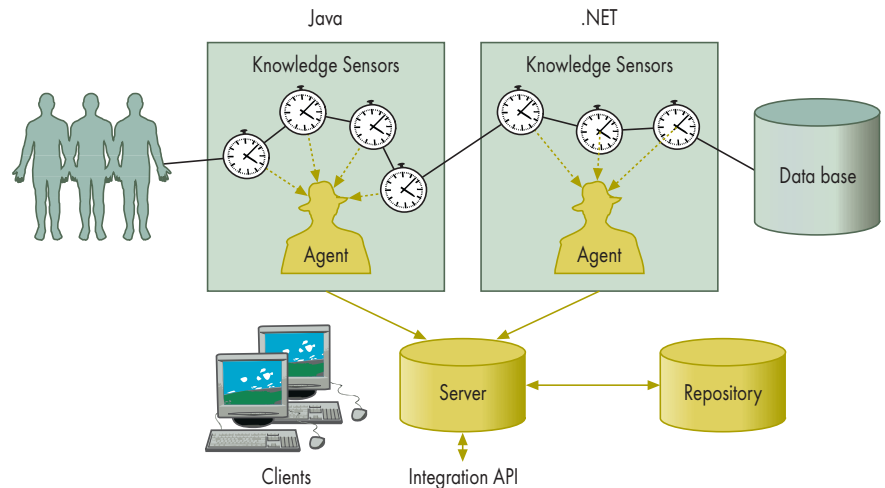
Now the person responsible can configure which methods she or he wants to monitor. When applying „Sensor Placement“, the agent picks the chosen methods selectively. For this, the agent adds instrumentation code as required to log data like run time, CPU usage, context information, method arguments, caller or exceptions for each call.

This performance loss amounts to 3 to 5 %, but it can actually be much higher as it depends on the runtime of individual methods and their call frequency. Short methods which would lead to a disproportionately high instrumentation overhead can be excluded in the configuration. There is even an own option for getter- and setter methods. This means the loss can usually be restricted to a bearable extent.

With the Purepath technology monitoring the product, the analyst can define a user-definable entry point, like a web service or a normal method call with trace points as sensors in the overall system. Purepath captures transactions from the entry point through all affected layers (see figure 2 and lead).

Virtual machines with interfaces

The developers of the Java Virtual Machines (JVM) realized the necessity of a standardized information interface for development and monitoring tools



Dynatrace Diagnostics has a simple but efficient structure (fig. 1).

(such as debuggers, coverage tools and profilers) at a very early stage. Until Java 1.4 these were the JVMPi (JVM Profiling Interface) and the JVMDI (JVM Debug Interface) and from Java 5 on they were combined in the JVMTI (JVM Tool Interface). Corresponding interfaces are planned for Microsoft's CLR (Common language Runtime). Entry points can be the profiling-API (especially *ICorProfilerInfo* and *ICorProfilerCallback*) as well as others, for instance the debug- and the meta-info-API, which allow the access to further CLR information.

You can get status information about the virtual machine via the interfaces and, by that, a certain control over the applications running on these VMs. The process is simple: A tool logs into the interface as agent and receives information about the requested events in the context of the applications. The VM creates events for all relevant run time occurrences, for starts

and stops, loading classes, entries and exits of methods, exceptions, for all storage operations such as heap allocations or object releases by the garbage collector and for starting and stopping threads.

The notification about entry and exit of a method however leads to a high performance loss, because the VM creates an event for all method calls down to the base classes. This overhead can only be managed by targeted, demand-orientated method instrumentation. The right time for this is when a class is loaded into the memory respectively when the just-in-time compiler is called.

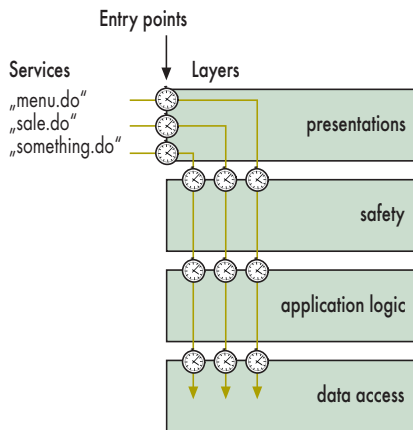
By now, the byte code instrumentation is an inherent part of the tool-APIs for VMs, not only for JVM and .Net. The „Byte Code Engineering Library“ is practically a standard for Java, and is used by several profilers, tracing tools and AspectJ. Microsoft offers methods for the instrumentation of the intermediate language (IL) for its CLR as part of the *ICorProfilerInfo*-interface.

3-tier-architectures, typical for enterprise environments, serve as test environment. The users access the actual application, which consists of a distributed web and application server, via web clients. Often additional databases, SAP or legacy systems are integrated into the application. This gives an idea how heterogeneous the environments are and how complicated the tasks can be for the system administration.

To be able to test Diagnostics, the test environment included typical dis-



- Dynatrace Diagnostics offers the user functions for performance-, management- and error analysis
- the data flow can be traced across multiple servers, in Java and .net-applications.
- For this, Diagnostics installs so-called agents on the systems in question which capture raw data in the applications via their Knowledge Sensors.
- On the graphic user interface of Diagnostics Client the user can configure the components and analyze the data evaluated statistically and represented graphically by Diagnostics Server.



The right path: Purepath runs through the typical layers of an application (fig.2).

ruptions such as the ones caused by an inappropriate configuration of the applications (pools too small, not enough links), the software (thread synchronization, CPU-intensive method) and the network (high latency). Two kinds of architectures with appropriate applications were used for the test:

- The .Net application Pet Shop 3.0 offered by Microsoft in the MSDN for developers, for which all three tiers are installed on one server. The user accesses it via web browser.

- The J2EE application Day Trader, distributed on an application and a web server under Linux as well as on a MySQL database server under Windows 2003. Programs natively implemented in Java or .Net under Windows XP served as clients.

When performing load tests of applications in a business environment, a huge number of parallel client accesses are simulated and values like response time and transaction throughput are measured. The tools used for this, for example HP's Loadrunner, Empirix eLoad or Borland's SilkPerformer, also perform black box tests. If the system performance is not satisfactory, this might not give any hints regarding the causes, but most of the time it is possible to monitor the average CPU utilization of components with this. As Diagnostics does not (yet) provide system monitoring data and concentrates on applications alone in the current version 2.6.0, it would be helpful to combine the data of both tool types to get an optimum overview. Dynatrace Diagnostics does not offer any interfaces for this, just like most other com-

mercial tools. This means you have to sort of jump between tools during the analytic phase.

Find the entry point via load analysis

A load test tool offers a first overview of throughput rates, response times and system utilization from the user's point of view. Based on this data, bottlenecks can be analyzed by Dynatrace Diagnostics. In the case of the tested web application it would make sense to start with the view „Tagged Web Requests“. Here, the „entry points“ of the clients (test scripts) into the application under test are listed in a table. If you have integrated appropriate tags for LoadRunner in your test scripts, they reappear under their names in the web queries listed in Diagnostics.

You can find measured values for each request such as data amount, CPU time or the total time used. Often this is enough to identify a typical weakness of the system like unacceptable response times for the user. For all other analyses down to code level, Purepath is the tool of choice. An analyst can decide at every point he wants to drill down deeper (via method breakdown and Purepath respectively) or, when the analysis is detailed enough, would like to receive the sum of all method APIs on this level.

During the tests, we immediately noticed the modified methods of applications that had to simulate things like computationally intensive procedures or lack of synchronization. Whenever you have made a potential improvement in your code after the successful

identification of a problem, it is possible to compare test runs in the visualization of the Diagnostic Client. Software architects can easily see in which part of the code they actually accomplished a measurable change. Diagnostics helps them with this as it allows them to directly jump to the corresponding place in the source code of the listed packages, classes or methods, if the developers used Eclipse or Visual Studio.

For evaluations with ANOVA or DoE (Design of Experiment) from the „Six Sigma World“ there is no export function available for the individual measurements to be analysed with programs like Minitab or Fathom.

Statisticians have to content themselves with minimum, maximum or average values generated in a 10 second pattern. Export in Diagnostics is possible via CSV or HTML or the JMX interface.

Because the focus of the analysis performed by Dynatrace is on the software side - which is certainly more in demand in development departments - other integrated „stop points“ in the test, like a limited number of simultaneously allowed browser connections to the web server, can be analyzed better with the load test tool. Obviously, Dynatrace has realized this and announced extensions for system monitoring for the upcoming version 3.0.

In the depths of the application

No matter if you are a performance engineer and want to identify bottlenecks or if you are a troubleshooting

Editions

Dynatrace Diagnostics

Workstation Edition: Allows the analysis of the application behavior during software development in the local development environment.

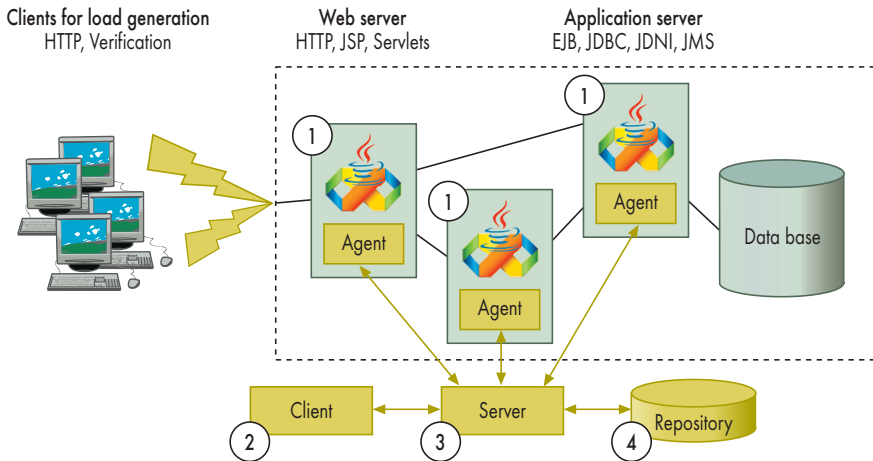
Professional Edition: Offers additional monitoring and diagnosis of two productive applications day-to-day, 24 hours.

Enterprise Edition: Monitoring, diagnosis and performance management of all applications in operation including Java applications on the mainframe, access to diagnosis data and actions company-wide.

Prices: On request www.dynatrace.com

X-Rating

- ⊕ Real time analysis of business transactions via Purepath
- ⊕ Cross-system usage
- ⊕ Analysis during the complete development phase
- ⊖ .Net- support somewhat behind that for Java
- ⊖ No integration of older or natively implemented components
- ⊖ Restricted export function of measured data



Test setup: The data collected by the agents at the measuring points (1) are displayed on the client (2), put in graphs by the server (3) and can be stored in the repository (4) (fig.3).

developer – you will benefit from Diagnostics. It will also help system administrators to monitor, understand and verify the dynamic behavior of a system in operation. With the real time analysis they get a support with measured data from the “system under diagnosis” (SUD) constantly coming in to the Diagnostics Client with a little delay. JMX allows plug-in to some frameworks like IBM's Tivoli or CA Unicenter for some measuring processes.

Discovery Run can be recommended as start point; the application runs through use cases which are as realistic as possible. In typical enterprise architectures you can carry out analyses with the Sensor Pack provided with Dynatrace, supporting all standard frameworks. Recorded PurePaths clearly show within the high-level architecture the dynamic interplay between the components/layers, and their communication interfaces.

With this, you get an overview very quickly – even with some complex or less known architectures. The step can lead to surprising insights, particularly for especially “exotic” implementations. A positive aspect is how transparent system limits or characteristics of certain vendors become and another is that Dynatrace does not influence the analyses of transactions.

Anyone who wants to have a deeper look into the applications can make customized, enhanced configurations and provide them on the server persistently. The user of the tools can select or deselect sensors on method level in the class tree “learned” during the Discovery Run.

As an alternative to the above-described „Outside-in“ process, Dynatrace also supports an „Inside-out“ process: Based on exceptions or identified bottlenecks on method level, the corresponding business transactions can be identified and thus the (potential) financial effects can be understood.

Though Dynatrace supports all popular technologies and vendors, the limits might become obvious during practical use, especially if older or natively (in C/C++) implemented components are an essential part of the system. Analyses of procedures on the database server (i.e. for stored procedures) or the SAP backend can only be realized with proprietary tools. However Diagnostics captures the call interface including their parameters which should be sufficient for a first analysis.

Despite the broad target group that Dynatrace addresses, its usability does not suffer. Configuration, capturing and evaluating the measured data including traces can be done intuitively and allows for a good workflow. You might not have the best overview whenever numerous windows are opened (for complex systems), but via the cockpit you always get back fast to your starting point, no matter if this was a Purepath, a web request or an exception. Working remotely, which is indispensable especially in an enterprise environment, works flawlessly with Dynatrace. The client reacts fast enough, even over small bandwidth connections. Data is available for analyses online and offline. This might be an advantage whenever the evaluation is done by another team or even across several time zones.

Conclusion

Anyone who has ever tried to implement a rudimentary measuring system in distributed systems, allowing the synchronization of components for an evaluation of performance and stability issues, will see Dynatrace as a real relief. It may not be a substitute for some more specific tools but it is an appropriate base for analyses due to its system independence and broad range of analysis, especially for teams. When analyzing an application, Dynatrace's Purepath technology allows a deep, freely configurable insight. (rh)

FRANK ROMETSCH

is Senior Engineer at Siemens Corporate Technology. His main areas are software tests and performance engineering.

HORST SAUER

is Senior Engineer at Siemens Corporate Technology. His main areas are software tests, testability and diagnosis.



dynaTrace
software

USA

dynaTrace software Inc.
200 West Street, Waltham, Massachusetts
02451, USA
Tel.: +1 (781) 466 8082

UK

dynaTrace software UK Ltd.
Argentum, 2 Queen Caroline Street,
Hammersmith London, W6 9DT, UK
Tel.: +44 (203) 178 6086

Netherlands

dynaTrace software GmbH
Graadt van Roggenweg 328-334,
Blok D, 3531 AH Utrecht, Netherlands
Tel.: +31 (30) 2982257

Germany:

dynaTrace software Germany GmbH
Landsberger Strasse 155,
80687 München, Germany
Tel.: +49 (89) 57959-157