03-2016 | Herbst | www.ijug.eu

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

Java aktuell

D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

Java wächst weiter

Microservices
Schnell und einfach implementieren

Container-Architektur
Verteilte Java-Anwendungen mit Docker

Java-Web-Anwendungen
Fallstricke bei der sicheren Entwicklung

4 191978 304903 03

iJUG Verbund

Neues von den letzten Releases

Sich schnell einen Überblick über bestehende Strukturen und deren Beziehungen verschaffen

Daten in unterschiedlichen Formaten in der Datenbank ablegen

Open-Source-Performance-Monitoring mit stagemonitor

Felix Barnsteiner und Fabian Trampusch, iSYS Software GmbH

In jedem Software-Projekt muss man sich früher oder später Gedanken über die Performance machen, um marktfähig zu bleiben und den Erwartungen der Nutzer zu entsprechen. Hierzu ist ein gutes Monitoring-Werkzeug unerlässlich. Kommerzielle Lösungen sind teuer und bestehende Open-Source-Projekte sind oftmals nicht speziell für Web-Anwendungen ausgelegt oder unterstützen Anwendungen im Clusterbetrieb unzureichend.

Genau hier setzt die Open-Source-Performance-Monitoring-Lösung „stagemonitor" an. Geboten werden alle Werkzeuge, die während der Entwicklung, der Qualitätssicherung und in der Produktion nötig sind, um die Performance einer Java-Web-Anwendung überwachen zu können. Die Lösung ist speziell darauf ausgelegt, aktuelle und historische Metriken von Anwendungen im Cluster-Betrieb zu überwachen. Die Plug-in-Architektur ermöglicht es, offizielle Erweiterungen oder Plug-ins Dritter einzubinden. Auch eigene Erweiterungen können entwickelt werden.

stagemonitor während der Entwicklung

In der Praxis realisiert man oftmals erst dann, wie wenig performant eine Anwendung ist, wenn sich ein verärgerter Kunde über eine langsame Seite beschwert. Gerade bei eCommerce-Webseiten kann dies zu erheblichen wirtschaftlichen Schäden führen. Nicht nur, dass ein verärgerter Kunde wahrscheinlich nicht nochmals einen langsamen Shop besucht, viele machen ihrem Unmut auch bei Freunden oder in sozialen Netzwerken Luft. Deshalb ist es entscheidend, Performance-Probleme so früh wie möglich im Entwicklungszyklus zu erkennen und zu beheben.

stagemonitor bietet mit dem In-Browser-Widget ein Werkzeug an, mit dem die Performance während der Entwicklung analysiert und zielgerichtet optimiert werden kann. Zudem wird der Entwickler auf Probleme hingewiesen, die sonst erst in der Produktion sichtbar werden. Das In-Browser-Widget ist ein Monitoring-Dashboard, das automatisch in das HTML von Webapplikationen eingefügt wird. Die wichtigsten Funktionen des In-Browser-Widgets sind:

- *Call Tree*
  Nach dem Öffnen wird ein Call Tree aller Methoden, SQL- und Elasticsearch-Queries angezeigt, die während der Bearbeitung des aktuellen Requests ausgeführt wurden. Hierdurch lassen sich die langsamen Methoden einer Anwendung identifizieren. stagemonitor unterstützt den Entwickler beim Beheben des Problems, indem es den zeitlichen Anteil eines Methodenaufrufs an der Anfrage darstellt. Dadurch kann der langsame Code optimiert werden, ohne dass man als Entwickler ziellos ins Blaue

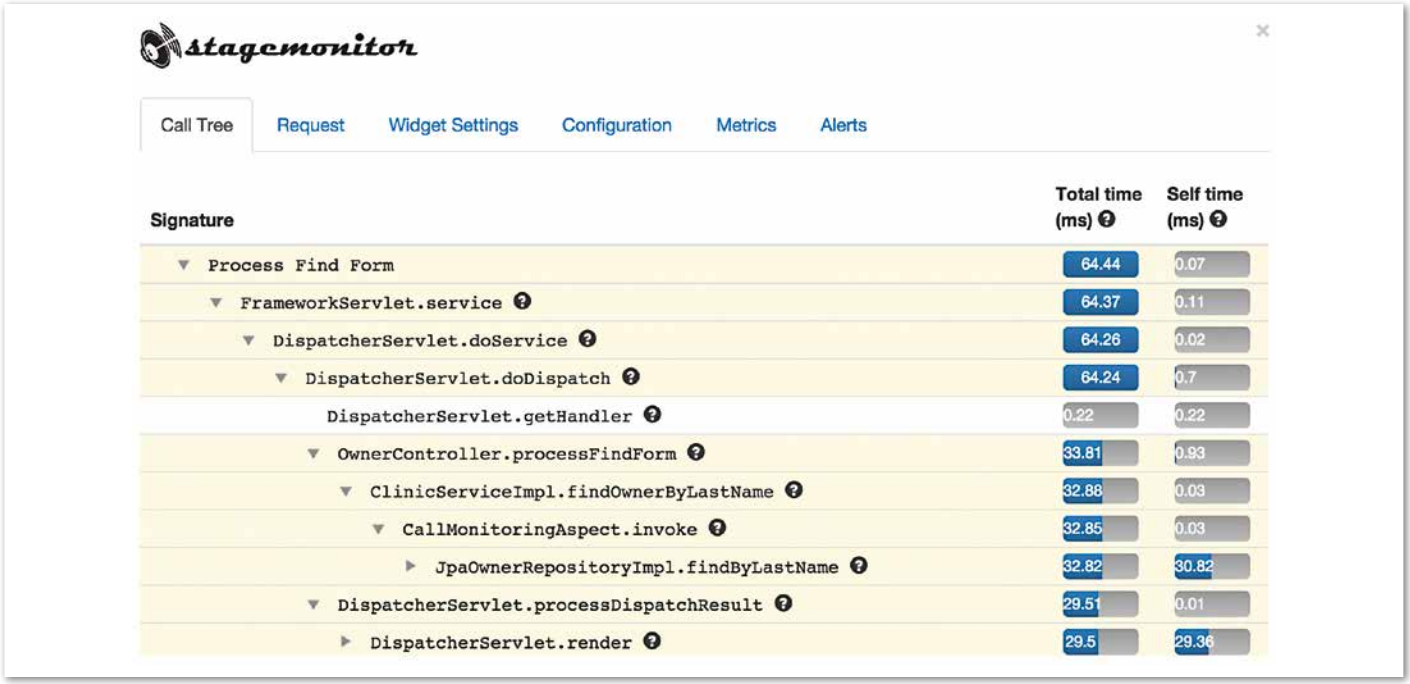

Abbildung 1: In Browser Widget – Call Tree

optimiert. Ajax Requests können ebenfalls analysiert werden *(siehe Abbildung 1)*.

- *HTTP-Request-Informationen*
  Im Request-Tab finden sich allgemeine Informationen zur abgesetzten Anfrage. Diese Informationen umfassen beispielsweise die Größe der Response, die Request-Header sowie Details zum User-Agent (von welchem Gerät und von welchem Betriebssystem die Anfrage abgesetzt wurde). Zudem wird die Ladezeit der Seite in die Kategorien Netzwerk, Server, DOM Processing und Page Rendering aufgeteilt.

- *Konfigurierbare Notifications*
  Es können Grenzwerte für die Seitenladezeit und die Anzahl der SQL-Queries während einer Anfrage definiert werden. Das Widget zeigt eine Notification an, falls diese Grenzwerte nicht eingehalten wurden. Es können so besonders datenbankintensive Requests identifiziert werden.

- *Real-Time-Performance-Graphen*
  Im Metrics-Tab befinden sich Graphen, die in Echtzeit beispielsweise Metriken der JVM, des Betriebssystems und der Antwortzeiten darstellen. Dieser Tab ist durch Plug-ins erweiterbar.

- *Alerts*
  Im Alerts-Tab können Grenzwerte für alle gesammelten Metriken wie beispielsweise die Antwortzeit festgelegt werden. Exemplarisch können hiermit E-Mail-Benachrichtigungen versendet werden, falls die durchschnittliche Antwortzeit der Suchfunktion im Produktivsystem einen bestimmten Wert überschreitet. Dieser Tab ist Bestandteil des Alerting-Plug-ins und zeigt, wie die Tabs durch Plug-ins erweiterbar sind.

stagemonitor in der Produktion

Die Performance während der Entwicklungsphase überwachen und verbessern zu können, ist schön und gut, aber viel wichtiger ist das Monitoring in der Produktion. Wenn eine Anwendung mit einem kleinen Testdaten-Bestand und einem aktiven Nutzer schnell ist, bedeutet das nicht, dass die Anwendung auch mit mehreren Nutzern und einer größeren Datenbasis skaliert. stagemonitor hat sich für den Einsatz in der Produktion bewährt, ohne dass hierzu ein weiteres Tool notwendig ist. Damit vereinfacht sich auch die Kommunikation zwischen Entwicklern und Administratoren, da beide dieselbe Daten-Grundlage und Auswertungsmöglichkeiten haben. Bei der Entwicklung von stagemonitor lag ein Augenmerk darauf, dass die Anfragebearbeitung in der Produktion nicht spürbar verlangsamt werden darf.

Da das In-Browser-Widget in der Produktion nicht aktiv ist und nur die Daten eines Servers anzeigen kann, gibt es für den produktiven Betrieb spezielle Dashboards, in denen die Informationen für alle Server zusammengefasst dargestellt sind. Die Dashboards basieren auf den Open-Source-Projekten „Kibana" und „Grafana"; sie lassen sich individuell anpassen. Dabei können die Daten wahlweise aggregiert (Übersicht über alle Server) oder einzeln dargestellt werden. Es ist auch möglich, historische Daten abzurufen. Deshalb ist es notwendig, eine Datenbank aufzusetzen, in der die Daten persistiert werden.

Analyse von Anfragen

Ein momentan weitverbreitetes Vorgehen, um mehr Einsicht in die Anfragen zu bekommen, die eine Applikation abarbeitet, ist die Analyse von Log-Dateien mit dem sogenannten „ELK-Stack". Hierzu werden zunächst Logstash-Access-Logs, beispielsweise von Apache, ausgelesen und mithilfe von regulären Ausdrücken Informationen extrahiert, beispielsweise die angefragte URL oder die Verarbeitungszeit. Anschließend werden die Daten in der Key-Value-Datenbank Redis zwischengespeichert. Der nächste Schritt ist die Ablage der Daten in Elasticsearch, um eine einfache Durchsuchbarkeit und Analyse zu ermöglichen. Im letzten Schritt muss noch ein Kibana-Dashboard erstellt werden, mit dessen Hilfe die Informationen visualisiert werden können.

Setzt man stagemonitor ein, so entfallen die meisten Schritte. stagemonitor befindet sich innerhalb der Applikation und hat so

direkten Zugriff auf Informationen wie die URL, die Verarbeitungszeit, aufgetretene Exceptions und viele weitere. Diese Informationen müssen nicht erst mühsam aus Logs extrahiert werden. Zudem sind nicht gleich vier neue Systeme zu installieren und zu warten, sondern nur Elasticsearch und Kibana. Um auch diesen Schritt zu vereinfachen, wird eine Imagedocker-compose.yml Datei angeboten, in dem alle benötigten Systeme bereits aufgesetzt und vorkonfiguriert sind.

stagemonitor sendet die gesammelten Informationen auf Wunsch selbstständig zu einem Elasticsearch-Server und bietet sogar ein vorkonfiguriertes Kibana-Dashboard an, das jedoch angepasst werden kann. Durch die dynamischen Analysemöglichkeiten von Kibana können vielseitige Fragestellungen beantwortet werden:

- Welche Fehler treten besonders häufig auf und durch welche Requests werden sie verursacht?
- Wie können die Fehler reproduziert werden?
- Warum war der Request eines Kunden zu einer bestimmten Zeit langsam?
- Welchen Browser und welchen Gerätetyp setzen meine Kunden am häufigsten ein?

Diese Informationen sind für den Betrieb und für das Marketing Gold wert, um Fehler oder das Nutzerverhalten nachzuvollziehen.

Analyse von Metriken

Ein weiteres Kerngebiet von stagemonitor ist die Analyse von Metriken verteilter Anwendungen. Zwar bietet das In-Browser-Widget bereits eine Möglichkeit, die Metriken eines Servers in Graphen darzustellen, jedoch reicht dies nicht immer aus. Will man die Metriken der Vergangenheit ansehen, interessiert man sich für die Metriken mehrerer Server oder möchte man eigene Dashboards erstellen, so benötigt man die von stagemonitor bereitgestellten Grafana-Dashboards. Grafana ist ein Visualisierungstool für Zeitreihen-Daten und unterstützt mehrere Zeitreihen-Datenbanken:

- Das Request-Dashboard gibt Auskunft über das Antwortzeitverhalten und den Durchsatz der Anwendung. Zudem werden die langsamsten und fehleranfälligsten Business Transactions (wie „Suche" oder „Artikeldetail ansehen") hervorgehoben.
- Das JVM-Dashboard enthält Informationen über die Heap-Auslastung, das Garbage-Collection-Verhalten und die CPU-Auslastung der JVM.
- Darüber hinaus gibt es Dashboards für EhCache-, Logging-, Application-Server-, Betriebssystem-Metriken und Datenbank-Abfragen.

Mit diesen Statistiken fällt es leicht, die Frage zu beantworten, ob einzelne Caches das Antwortverhalten der Anwendung tatsächlich verbessern oder ob die Cache-Hit-Rate zu gering ist, um einen tatsächlichen Mehrwert zu bieten. Jedes Dashboard bietet die Wahl, welche Applikation, welcher Host und welche Instanz betrachtet werden soll. Somit lassen sich wahlweise entweder einzelne Server oder der gesamte Cluster im Überblick betrachten.

stagemonitor unterstützt die Zeitreihen-Datenbanken Elasticsearch, InfluxDB und Graphite. Empfohlen ist der Einsatz von Elasticsearch, da hierbei nur eine Datenbank installiert werden muss, um sowohl die Request-Traces als auch die Metriken abzuspeichern. Dies macht die Installation und den Betrieb von stagemonitor besonders einfach. Zudem kann Elasticsearch gut mit großen Datenmengen umgehen, da es sich einfach skalieren lässt.

Der Unterschied zu vielen anderen Lösungen besteht darin, dass nicht bei jedem eingehenden Request Daten übermittelt werden müssen. Die statistischen Daten über die Antwortzeit, beispielsweise Standardabweichung, Mittelwert und verschiedene Perzentile, werden im Server berechnet und periodisch (standardmäßig jede Minute) an die Zeitreihen-Datenbank übermittelt. Damit ist es egal, wie viele Anfragen der Server bearbeiten muss. Die anfallende Datenmenge bleibt immer konstant und wird nur vom Übertragungsintervall bestimmt. Auch die Datenstruktur, die die Statistiken generiert, ist unabhängig von der Anzahl eingehender Anfragen, da immer nur eine repräsentative Menge von rund tausend

Abbildung 2: Request Dashboard

Antwortzeiten (pro Business Transaction) im Speicher gehalten wird *(siehe Abbildung 2)*.

stagemonitor ausprobieren

Nachfolgend sind drei Möglichkeiten beschrieben, um stagemonitor schnell und einfach auszuprobieren. Für eine Live-Demo steht unter „www.stagemonitor.org" ein Link zur „Spring PetClinic" mit stagemonitor bereit. Das In-Browser-Widget wird durch das Icon in der unteren rechten Ecke aktiviert.

Um stagemonitor in eigener Umgebung auszuprobieren, steht als zweite Möglichkeit auf GitHub *(siehe „https://github.com/stagemonitor/spring-petclinic#running-petclinic-locally-with-stagemonitor-enabled")* ein Beispielprojekt auf Basis der Beispiel-Applikation „Spring PetClinic" bereit. Um die Anwendung zu starten, sind lediglich drei Befehle auf der Konsole erforderlich *(siehe Listing 1)*.

```
git clone https://github.com/stagemonitor/spring-petclinic.git
cd spring-petclinic
mvn tomcat7:run
```

Listing 1

Das Projekt wird durch die Befehle per „git" aus GitHub geklont und mithilfe von Maven gebaut. Anschließend wird ein Embedded-Tomcat-Server mit der Anwendung gestartet. Unter *„http://localhost:9966/petclinic"* steht nun die Beispiel-Applikation „Spring PetClinic" mit aktiviertem stagemonitor-In-Browser-Widget zur Verfügung.

Als dritte Möglichkeit wird stagemonitor in das Spring-Beispielprojekt PetClinic *(siehe „https://github.com/stagemonitor/spring-petclinic")* integriert. Hierfür muss die „pom.xml" um den Dependency-Eintrag für stagemonitor ergänzt werden *(siehe Listing 2)*.

```
<dependencies>
  <dependency>
     <groupId>org.stagemonitor</groupId>
     <artifactId>stagemonitor-web</artifactId>
     <version>0.22.0</version>
  </dependency>
  [....]
</dependencies>
```

Listing 2

Projekte, die Gradle oder Ant verwenden, können ähnlich gebaut werden. Hierfür sei auf die Dokumentation von stagemonitor *(„https://github.com/stagemonitor/stagemonitor/wiki/Step-1:-Prerequisites")* verwiesen. Zuletzt wird unter „src/main/resources" die Datei „stagemonitor.properties" angelegt. In dieser Datei können diverse Konfigurations-Parameter *(siehe https://github.com/stagemonitor/stagemonitor/wiki/Configuration-Options")* für stagemonitor hinterlegt sein, die beim Start der Anwendung aktiviert sind. Die meisten lassen sich jedoch auch zur Laufzeit über das In-Browser-Widget anpassen. Als minimale Konfiguration ist „stagemonitor.instrument.include=org.springframework.samples.petclinic" empfohlen. Der Konfigurationsschalter erlaubt die Angabe von kommagetrennten Package- und Klassennamen, die instrumentiert werden sollen. Damit ist die Integration abgeschlossen.

Das Maven-Target „tomcat7:run" baut und startet nun die Anwendung. Sie ist unter *„http://localhost:9966/petclinic"* erreichbar. In der rechten unteren Ecke befindet sich das stagemonitor-Icon für das In-Browser-Widget zum Anklicken.

Fazit

stagemonitor stellt eine praxiserprobte Lösung für das Monitoring der Anfragen und Einblicke in das interne Performance-Verhalten der Anwendung dar. Es unterstützt dabei den gesamten Lebenszyklus der Anwendung, von der Entwicklung bis zum Betrieb. Mit dem Call-Tree des In-Browser-Widgets steht ein mächtiges und einfaches Werkzeug zur Verfügung, um gezielte Optimierungen des serverseitigen Codes zu ermöglichen. Das Aufzeichnen der Request-Statistiken bietet detaillierte Informationen, um auch bei umfangreichen Web-Anwendungen langsame Funktionalitäten zu identifizieren und den langfristigen Erfolg der Optimierung messbar zu machen. Auch Fehler lassen sich mit den gesammelten Daten einfacher reproduzieren.

Auf der Roadmap steht ein Ausbau der Analyse-Funktionen, um eine leichtgewichtige Alternative zu Google Analytics oder Piwik bieten zu können. In Zukunft wird stagemonitor unter anderem Unterstützung für automatisiertes Baselining bieten. Dies soll es ermöglichen, konkrete Kennzahlen zur Performance der Anwendung in Bezug auf unterschiedliche Releases zu liefern. Auch automatisierte Anomalie-Detektion oder Rückschlüsse auf Fehlerursachen sollen so möglich werden.Getreu dem Zitat des Informatik-Pioniers Donald Knuth „We should forget about small efficiencies, say about 97 percent of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3 percent" hilft stagemonitor dabei, sich auf diese drei Prozent zu konzentrieren. stagemonitor steht kostenlos unter der Apache License 2.0 unter *„http://www.stagemonitor.org"* zur Verfügung.

Felix Barnsteiner
f.barnsteiner@isys-software.de

Felix Barnsteiner ist Software Engineer bei der iSYS Software GmbH. Er ist der Entwickler des Open-Source-Performance-Monitoring-Projekts stagemonitor. Zudem beschäftigt er sich mit der Entwicklung von eCommerce-Systemen.

Fabian Trampusch
f.trampusch@isys-software.de

Fabian Trampusch ist Software Engineer bei der iSYS Software GmbH. Er entwickelt moderne Web-Anwendungen mit Haupt-Augenmerk auf Qualität und Benutzerkomfort. Neben der Arbeit studiert er den Master „Software Engineering" an der Hochschule München.