Lasttests und Performanceoptimierung

Best-Practice-Ansätze & Anregungen

Wie man den Open Text Delivery Server in den Griff bekommt
oder "Fragen die man vor dem Going-Live beantworten sollte ..."

OPEN TEXT
PREMIER
PARTNER

Wer sind wir?

Aus Erfahrung erfolgreich

- über 300 Online-Projekte bei größeren mittelständischen Unternehmen und Konzernen
- Fokus auf OpenText Produkte und Technologien
- Verantwortung / Mitverantwortung für >10 OTWSM-Infrastrukturen

Key Facts

- inhabergeführtes Unternehmen, gegründet 2003
- 20 feste Mitarbeiter
- RedDot-Partner seit 2003, heute OpenText Partner

Lasttest

Warum Lasttests?

Lasttests bestätigen oder widerlegen die Erwartungshaltung an ein System

- Bestimmung der Leistungsfähigkeit einer Hard- und Softwarekonfiguration vor dem Going-Live
  - Können angestrebte Ziele mit der Infrastruktur umgesetzt werden?
  - Simulation steigende Nutzerzahlen
  - Simulation extreme Zugriffszahlen
- Identifikation von zukünftigen Problemen
  - steigender Ressourcenbedarf
  - schlecht angepasste Software
  - suboptimale Systemkonfiguration
  - unzureichende Leitungskapazitäten

Lasttest

Grundlagen für valide Lasttest

Alte Ingenieurs-Weisheit: Wer misst misst Mist!

- Simulation „realistischer“ Zugriffe auf den Webserver
- WICHTIG
  - funktionierendes Sessionhandling sicherstellen!
  - Realistischer Mix aus anonymen und session-basierenden Zugriffen
- Simulation des produktiven Betriebs mit definierten Bedingungen
  - Top 25 der Seiten laut Website-Statistik
  - Top 10 der performance-hungrigsten Seiten laut Entwicklung
- Protokollierung und Kontrolle der vom Server zurückgegeben Antworten („HTTP 500“ ist auch eine Antwort ;-) !)
- Identifikation kundenindividueller Bottlenecks - Aufspüren von Optimierungspotenzialen durch strukturiertes Analysieren von Auswirkungen
  - geänderter Systemparameter
  - Code-Anpassungen (hier nicht Thema).

Wie kann ein Lasttest aussehen?

- Ein Testplan enthält einen oder mehrere Threads die eine Abfolge von Operationen auf einer Website beschreiben, z.B.
  - Nutzeranmeldungen
  - Logins
  - Suchfunktionen nutzen
  - Formulare verwenden
  - …
- Der Testplan definiert wie Threads bei einem Lasttest abgearbeitet werden sollen
  - Anzahl gleichzeitiger Threads
  - Wiederholungen
  - Ramp-up

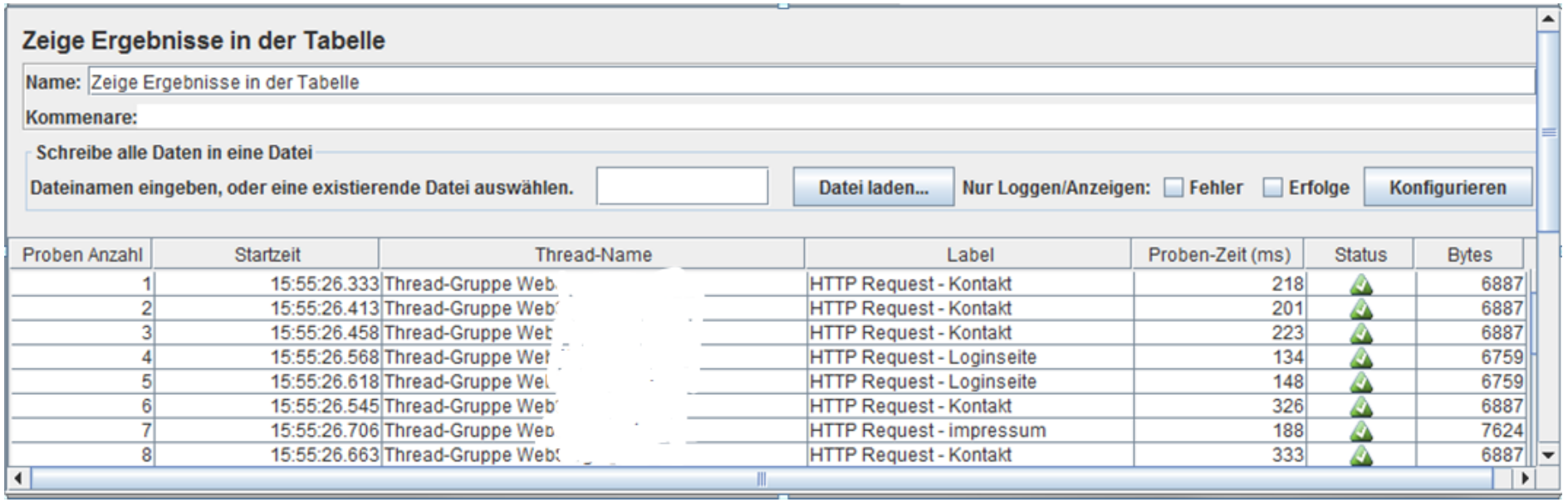
Lasttest

Welche Aussagen liefert der Lasttest?

- Visuelle und tabellarische Darstellung der Ergebnisse (Tabellen & Statistiken)
- Vergleich der Werte mit Hardware und Applikationseinstellungen
- Identifizierung von Bottlenecks durch parallele Beobachtung der Performance-Parameter der betroffenen Systeme

Lasttest

Beispiel: Apache JMeter™



Proben Anzahl	Startzeit	Thread-Name	Label	Proben-Zeit (ms)	Status	Bytes
1	15:55:26.333	Thread-Gruppe Web	HTTP Request - Kontakt	218		6887
2	15:55:26.413	Thread-Gruppe Web	HTTP Request - Kontakt	201		6887
3	15:55:26.458	Thread-Gruppe Web	HTTP Request - Kontakt	223		6887
4	15:55:26.568	Thread-Gruppe Web	HTTP Request - Loginseite	134		6759
5	15:55:26.618	Thread-Gruppe Web	HTTP Request - Loginseite	148		6759
6	15:55:26.545	Thread-Gruppe Web	HTTP Request - Kontakt	326		6887
7	15:55:26.706	Thread-Gruppe Web	HTTP Request - impressum	188		7624
8	15:55:26.663	Thread-Gruppe Web	HTTP Request - Kontakt	333		6887

Lasttest

Wie lässt sich ein Lasttest validieren?

z.B. durch die Verfolgung der Session-ID (SID) im RDE-Log:

143920	2012-07-25	16:55:01.687	(L)	Database: Selecting User by login from database Login=intentive	database	http-9080-21	anonymous	SID-7F80847F-FCDD4E0C	10.101.106.1	iauth			req-468699	rde.log
145497	2012-07-25	17:45:35.561	(L)	Weblet.handleRequest: Requested making-experts/hs.xsl/kontakt.html	exchange_request	http-9080-36	anonymous	SID-FC4B1325-0C86EA0E	10.101.106.1	xchg			req-472677	rde.log
145502	2012-07-25	17:45:35.628	(L)	Weblet.handleRequest: Requested making-experts/hs.xsl/kontakt.html	exchange_request	http-9080-21	anonymous	SID-9DE02706-E834CCDE	10.101.106.1	xchg			req-472681	rde.log
145504	2012-07-25	17:45:35.667	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-FC4B1325-0C86EA0E/making-experts/hs.xsl/kontakt.html, size=6719	log	http-9080-36	anonymous	SID-FC4B1325-0C86EA0E	10.101.106.1				req-472677	rde.log
145505	2012-07-25	17:45:35.667	(L)	exit MasterServlet.service result: ' PSX(0/0) cached: false variants:null 109ms'	log	http-9080-36	anonymous	SID-FC4B1325-0C86EA0E	10.101.106.1				req-472677	rde.log
145509	2012-07-25	17:45:35.693	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-DCBF8BC2-6CB26AAB/making-experts/hs.xsl/kontakt.html, size=-1	log	http-9080-19	anonymous	SID-DCBF8BC2-6CB26AAB	10.101.106.1				req-472727	rde.log
145510	2012-07-25	17:45:35.693	(L)	exit MasterServlet.service result: 'cached PSX(0/0) cached: true variants:1 3ms'	log	http-9080-19	anonymous	SID-DCBF8BC2-6CB26AAB	10.101.106.1				req-472727	rde.log
145511	2012-07-25	17:45:35.701	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-9DE02706-E834CCDE/making-experts/hs.xsl/kontakt.html, size=6719	log	http-9080-21	anonymous	SID-9DE02706-E834CCDE	10.101.106.1				req-472681	rde.log
145512	2012-07-25	17:45:35.701	(L)	exit MasterServlet.service result: ' PSX(0/0) cached: false variants:1 76ms'	log	http-9080-21	anonymous	SID-9DE02706-E834CCDE	10.101.106.1				req-472681	rde.log
145517	2012-07-25	17:45:35.761	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-D179D1F6-3C3DCC9B/making-experts/hs.xsl/kontakt.html, size=-1	log	http-9080-50	anonymous	SID-D179D1F6-3C3DCC9B	10.101.106.1				req-472739	rde.log
145518	2012-07-25	17:45:35.761	(L)	exit MasterServlet.service result: 'cached PSX(0/0) cached: true variants:2 3ms'	log	http-9080-50	anonymous	SID-D179D1F6-3C3DCC9B	10.101.106.1				req-472739	rde.log
145522	2012-07-25	17:45:35.826	(L)	Weblet.handleRequest: Requested making-experts/hs.xsl/login.html	exchange_request	http-9080-36	anonymous	SID-FC4B1325-0C86EA0E	10.101.106.1	xchg			req-472740	rde.log
145523	2012-07-25	17:45:35.834	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-FC4B1325-0C86EA0E/making-experts/hs.xsl/login.html, size=6687	log	http-9080-36	anonymous	SID-FC4B1325-0C86EA0E	10.101.106.1				req-472740	rde.log
145524	2012-07-25	17:45:35.834	(L)	exit MasterServlet.service result: ' PSX(0/0) cached: false variants:1 9ms'	log	http-9080-36	anonymous	SID-FC4B1325-0C86EA0E	10.101.106.1				req-472740	rde.log
145528	2012-07-25	17:45:35.889	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-9DE02706-E834CCDE/making-experts/hs.xsl/login.html, size=-1	log	http-9080-21	anonymous	SID-9DE02706-E834CCDE	10.101.106.1				req-472743	rde.log
145529	2012-07-25	17:45:35.889	(L)	exit MasterServlet.service result: 'cached PSX(0/0) cached: true variants:1 1ms'	log	http-9080-21	anonymous	SID-9DE02706-E834CCDE	10.101.106.1				req-472743	rde.log
145533	2012-07-25	17:45:35.951	(L)	MasterServlet.invokeRequest: remoteAddr=/10.101.106.1 URL=/xchg/SID-DCBF8BC2-6CB26AAB/making-experts/hs.xsl/login.html, size=-1	log	http-9080-19	anonymous	SID-DCBF8BC2-6CB26AAB	10.101.106.1				req-472742	rde.log


Performance-Tuning
Hintergründe & Stellschrauben

Performance-Tuning: Hintergründe

Warum kunden-individuelles Performance-Tuning

- OTWSM ist ein Out-of-the-Box-Produkt aus einer Enterprise Content Management Produkt-Suite
  - Unterstützung diverser Plattformen, Standards, Schnittstellen
  - unterschiedlichste Projekte
  - angebundenen Drittsysteme
- Standard-Software muss an kundenindividuelle Gegebenheiten angepasst werden um
  - Jeweilige Hardwareressourcen voll auszuschöpfen
  - Heterogene System-Anbindungen optimal zu nutzen
  - Spezialisierte Anwendungsfälle performant abzubilden
  - Engpässen bei hoher Last gezielt vorzubeugen

Performance-Tuning: Hintergründe

Optimierungspotenziale suchen

- Bottlenecks durch Lasttests identifizieren
  - Erstellen von „lebensnahen“ Lasttests
  - Analyse und Vergleich von Lasttestergebnissen vor und nach Anpassung
  - Erkenntnisse der Lasttests in konkrete Optimierungen umwandeln
- Systemzustand überwachen
  - Kontinuierliches Monitoring aller performance-relevanten Parameter
  - Wiederholte und angepasste Lasttests (besonders bei Änderungen im System)

Tomcat Application Server

Verarbeitungseigenschaften

- Anpassung des reservierten Speichers für die Anwendungsausführung
  - reservierten Speicher für die Applikation auf einen der Hardware und dem Bedarf angemessenen Wert festlegen
- Für den OTDS kann diese Anpassung in die Datei rdesetenv.sh (/WEB-INF/bin) geschrieben werden

Java Virtual Memory

- Standardmäßig der JVM zugesicherter Speicher:
  - JAVA_MEMORY=6144
  - JAVA_MEMORY_MIN=4096
- Mögliche Zielkonfiguration:
  - JAVA_MEMORY=6144
  - JAVA_MEMORY_MIN=6144
- Hier können - entsprechend der Hardwareausstattung - Speicherwerte zugewiesen werden.
- Sofern nicht zwingende Gründe dagegen sprechen, empfiehlt es sich, die Werte identisch zu konfigurieren, da es ansonsten bei der nachträglichen Allokation von RAM zu Verarbeitungspausen kommen kann.

Threads: Parallelisierung durch Multithreading

Die folgenden Anpassungen können in der server.xml vorgenommen (/tomcat/conf)

- Einstellung der Anzahl maximaler Threads

  Es können ggf. mehr parallele Anfragen pro Zeit verarbeitet werden. Hierbei muss mit den passenden Monitoringtools beobachtet werden, welche Threadanzahl für den produktiven Betrieb geeignet ist.

- In der Praxis hat es sich bewährt von 400 Threads auszugehen und diesen Wert auf Basis von Lasttestergebnissen nach oben oder unten anzupassen.

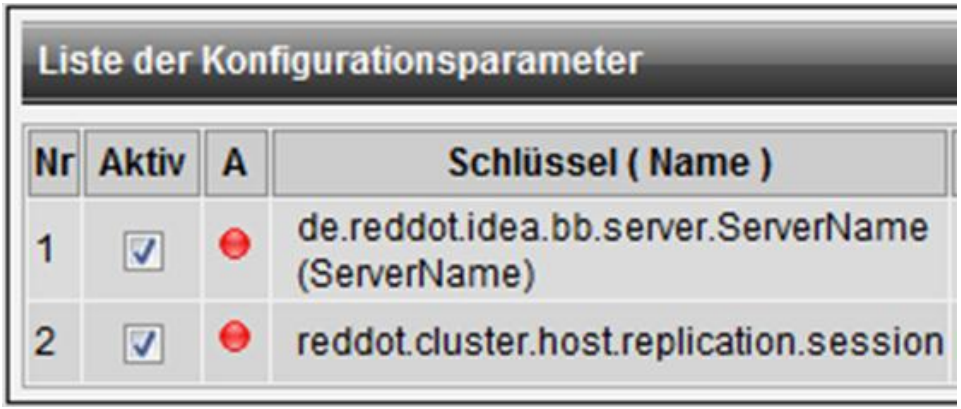
Keep it small and simple!

- Reduzierung der in einer Tomcat-Instanz ausgeführten Applikationen
  - Möglichst nur eine Applikation pro Tomcat
  - Möglichst nur ein Tomcat pro Server
- Größere Anzahl an Threads pro Applikation möglich
- Besser Überschaubarkeit von Bottlenecks und Fehlerpunkten

Delivery Server

Clustering: Ausfallsicherheit und Performance durch Redundanz

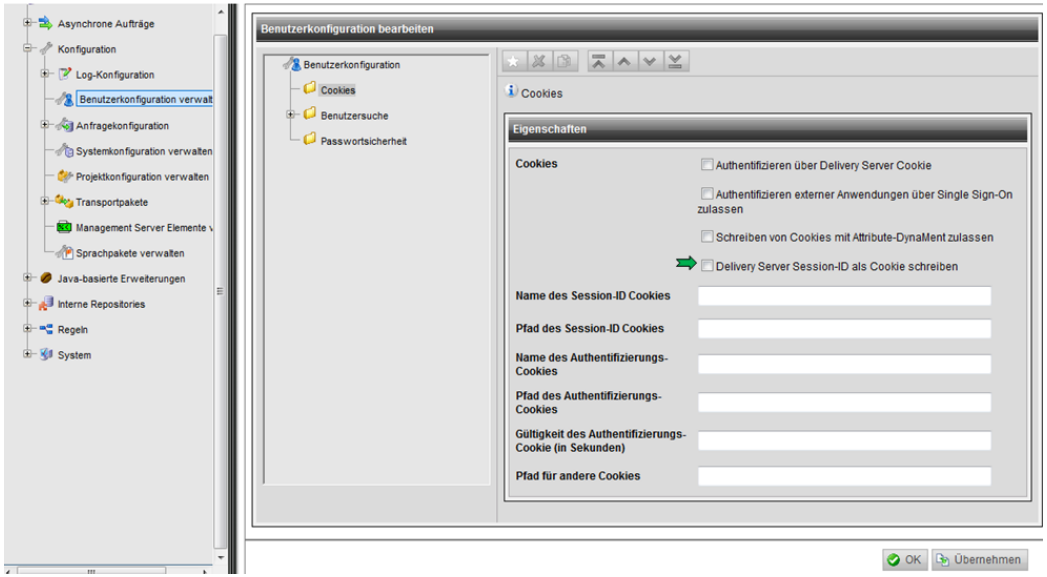
- ABER: OTDS-eigene Session-Replizierung ist langsam, OT empfiehlt die Nutzung nicht in produktiven Umgebungen, bei denen die Session ausfallkritisch ist!
- Angelegt wird hierzu ein lokaler Konfigurationsparameter je Clusterserver, der auf den Replikationspartner referenziert:



Performance-Tuning: Stellschrauben Open Text Delivery Server

Session-Cookies: Verlagerung der Session-Stickyness auf Client-Seite

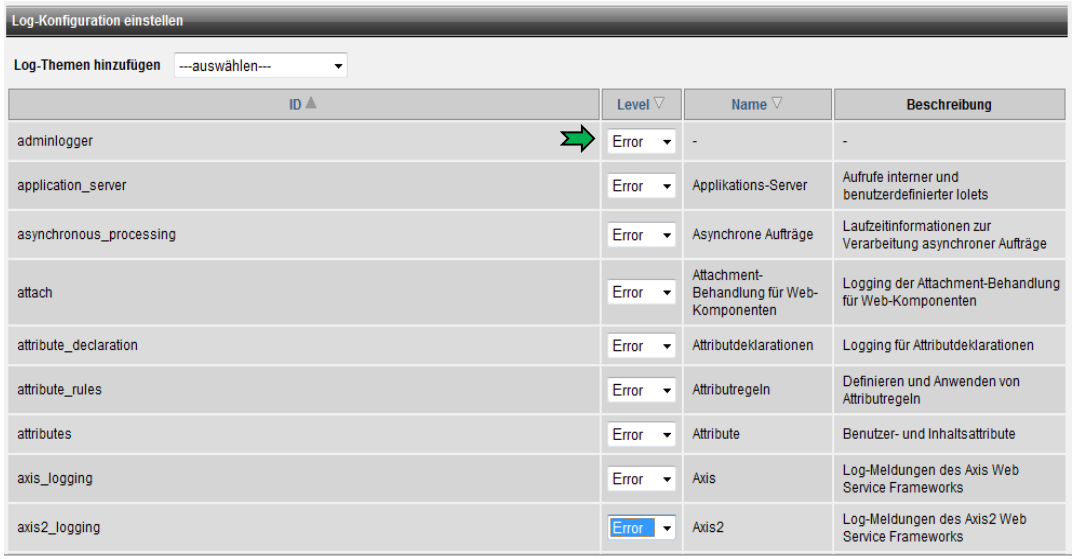
- Ermöglicht persistente Sessions mit dem OTDS bei vorgeschaltetem Loadbalancing ohne Nutzung der Delivery Server-internen (langsamen!) Session-Replizierung
- Server Manager -> Konfiguration -> Benutzerkonfiguration verwalten



Performance-Tuning: Stellschrauben Open Text Delivery Server

Log-Level: Beschränkung auf das Wesentliche

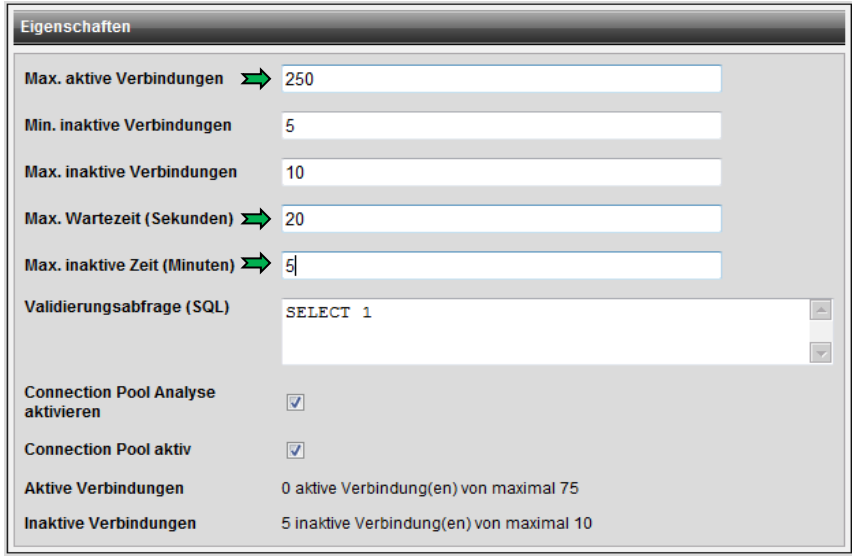
- Das Log-Level sollte in Produktivumgebungen maximal auf „Error“ eingestellt sein
- Dies verringert Schreibzugriffe auf das Log (HDD-Performance!) und hält es klein
- Server Manager -> Konfiguration -> Log-Konfiguration -> Einstellen -> Level



Performance-Tuning: Stellschrauben Open Text Delivery Server

Datenbank-Konnektor: Anpassung des Verbindungspools

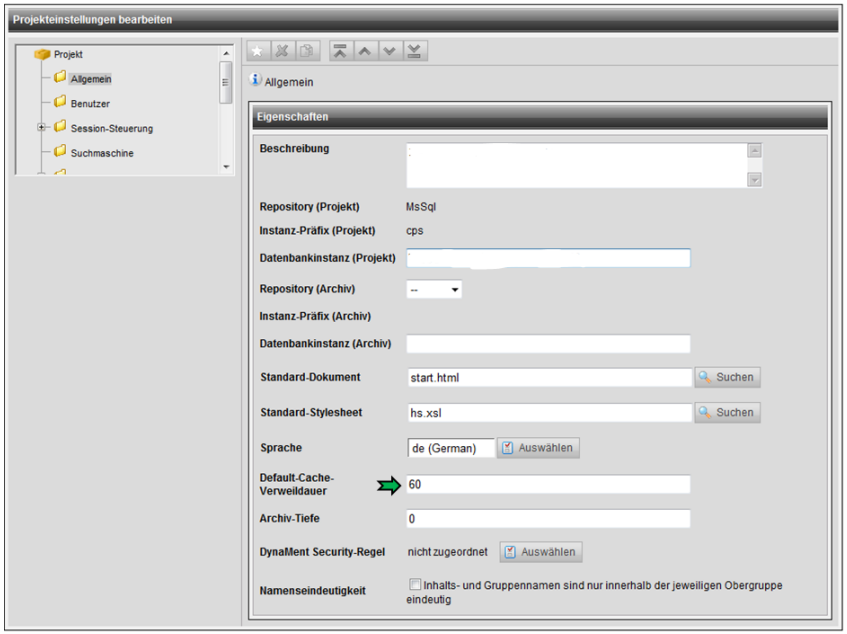
- Bessers Abfrageverhalten zur Datenbankinstanz durch Anpassung der Datenbank-Konnektoren
  - Maximale Verbindungen, inaktive Zeit, inaktive Verbindungen und Wartezeit
  - Minimale Verbindungsanzahl
- Konnektoren -> Verwalten -> Connection-Pool



Performance-Tuning: Stellschrauben Open Text Delivery Server

Seiten-Caching aktivieren

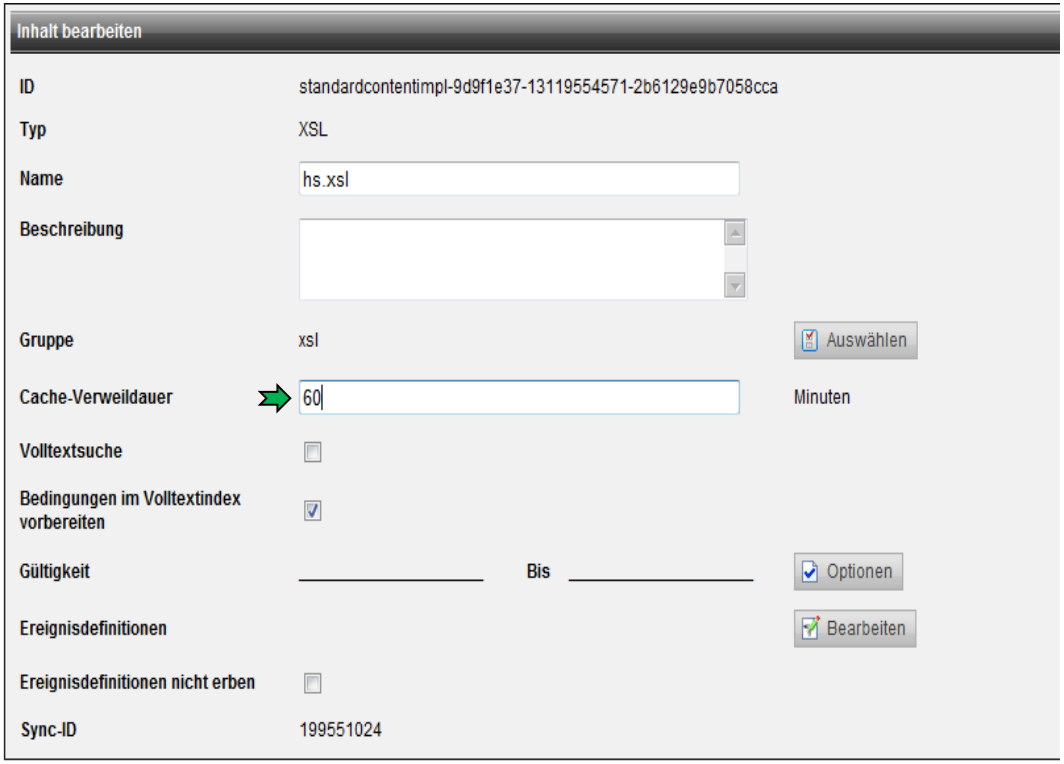
- Das Caching auf Seiten, XSLs und URLs führt zu schnelleren Zugriffe auf häufig abgerufene Inhalten
- Standard-Caching einfach einrichtbar – individuelles Tuning sinnvoll
- Projekt -> Projekteinstellungen bearbeiten -> Default-Cache-Verweildauer



Open Text Delivery Server

XSL-Caching aktivieren

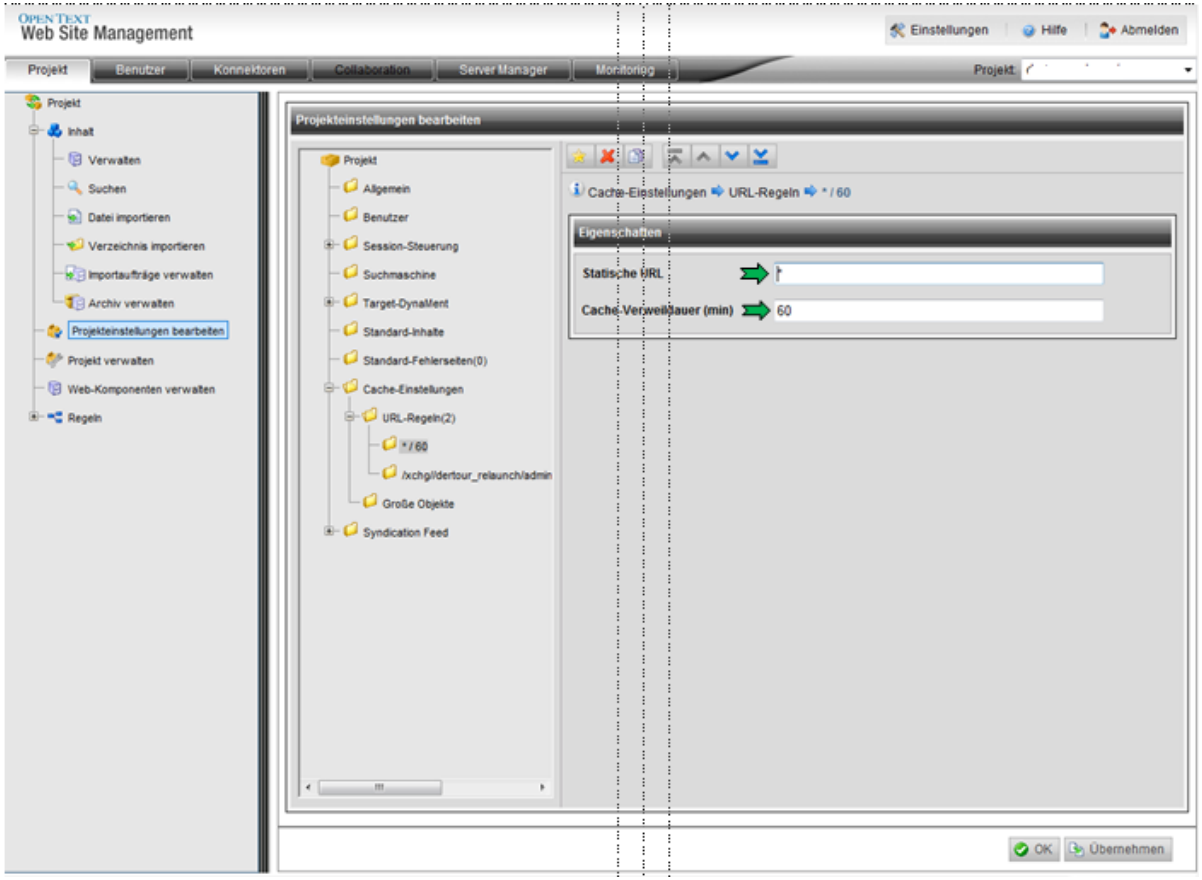
- Muss für jedes XSL-File konfiguriert werden
- Projekt -> Inhalt -> Verwalten -> XSL -> *.xsl -> Inhalt -> Cache-Verweildauer



Performance-Tuning: Stellschrauben Open Text Delivery Server

URL-Caching aktivieren

- Projekt -> Projekteinstellungen bearbeiten -> Cache-Einstellungen -> URL-Regeln



Datenbankserver

Hardware-Dimensionierung überprüfen

Die Datenbank ist das Rückgrat der Applikation!

- Angemessene RAM-Ausstattung ist Grundlage für performante Datenbankabfragen (Index und häufig genutzte Daten sollten im RAM liegen können) – hoher Festplatten-I/O deutet auf zu geringe RAM-Ausstattung hin
- Bereits geringfügig erhöhte Zugriffszeiten der HDDs kumulieren sich aufgrund hoher Anzahl SQL-Queries pro ausgelieferter Seite schnell zu signifikanten Wartezeiten – Perfmon-Parameter „Disk-Queue“ ist wichtiger Indikator
- Bereits geringe Netzwerklatenzen kumulieren sich aufgrund hoher Anzahl SQL-Queries pro ausgelieferter Seite schnell zu signifikanten Wartezeiten - OT empfiehlt dedizierte Gigabit-Netzwerkverbindung zwischen Application-Server und DB-Server

Individuelle Indizes gezielt anlegen

- Eine vollständige Indizierung häufig genutzter Tabellen kann die Zugriffsgeschwindigkeit deutlich erhöhen
- Ein überlasteter DB-Server bremst das gesamte OTDS-Cluster aus!
- Gründliche Tests sind erforderlich
  - Gibt es Fehler und Wechselwirkungen?
  - Haben die Indizes einen positiven Effekt?

Tuning von Datenbankservern ist ein Spezialisten-Thema!

- Die Absprache mit den Datenbankadministratoren ist unabdingbar, da diese als Spezialisten Ihr System und dessen Optionen am besten bewerten können.
- Eine Auswertung von Lasttestergebnissen kann die Datenbankadministratoren bei der Identifizierung und Behebung von Bottlenecks in Ihrem Bereich unterstützen.

Webserver

Apache Webserver / IIS via mod_jakarta vor den Tomcat Applicationserver schalten

- Grundsätzliche Empfehlung des BSI zur Erhöhung der Applikationssicherheit
  - https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/isi_web_server_studie_pdf
- Caching-Funktionen des Webservers können für statische Inhalte (z. B. Bilder) genutzt werden
- Messungen bei verschiedenen Kunden haben Leistungssteigerung im zweistelligen Prozentbereich ergeben

Loadbalancer

Verteilung der Zugriffe im Cluster

Herausforderung: persistente Sessions

- Zur Erinnerung:
  „OTDS-eigene Session-Replizierung ist langsam, OT empfiehlt das nicht in produktiven Umgebungen, bei denen die Session nicht ausfallkritisch ist!“
- Persistente Sessions sollten im Loadbalancer verwaltet werden, nicht im DS!
- Verwendung von Session-Cookies i.d.R. unerlässlich, da SID in URL von den wenigsten Loadbalancern erkannt wird.
- Aktivierung im OTDS erforderlich „ Write Delivery Server session ID as cookie”


Haben Sie noch Fragen?