

## Performance-Profiling komplexer Netzwerkanwendungen

# Das Nadelöhr finden

Eine Netzwerkanwendung soll vielen Anwendern bei gleichzeitigem Zugriff zufriedenstellende Antwortzeiten bieten. Beim Belastungstest fällt die Anwendung aber mit Pauken und Trompeten durch. Wo liegt das Nadelöhr? Für diese Detektivarbeit gibt es entsprechend spezialisierte Werkzeuge, zum Beispiel den Application Expert von Compuware.

**E**ine neu entwickelte Anwendung ist funktional getestet und der Termin für die Auslieferung rückt unaufhaltsam näher. Als Teil der Abnahme müssen in einem Lasttest die vereinbarten Antwortzeiten beziehungsweise Benutzerzahlen überprüft werden. Sollte sich bei diesem Lasttest herausstellen, dass die Anwendung nicht den Anforderungen standhält, ist das Projekt gefährdet.

Der Artikel zeigt an einem Beispiel, welche Probleme auftreten können und wie Sie diese Risiken mit präventiven Maßnahmen in der Entwicklungsphase einer Anwendung minimieren können.

## Beim Lasttest durchgefallen

Die Beispielanwendung ist ein mit .NET-Technologie entwickelter Webshop. Zwei Wochen vor der Auslieferung werden in einem Lasttest die geforderten Service Level Agreements (SLA) überprüft:

## Auf einen Blick

### Autor

**Kurt Aigner** ist Consultant für Testing- und Development-Solutions bei Compuware. Nähere Informationen finden Sie unter [www.compuware.de](http://www.compuware.de).



dotnetpro.code  
A0407Profile

**Technik** Performance-Profiling und -Tuning von Multi-Tier-Anwendungen in komplexen Netzen

**Voraussetzungen** Compuware Application Expert, DevPartner Studio Professional Edition

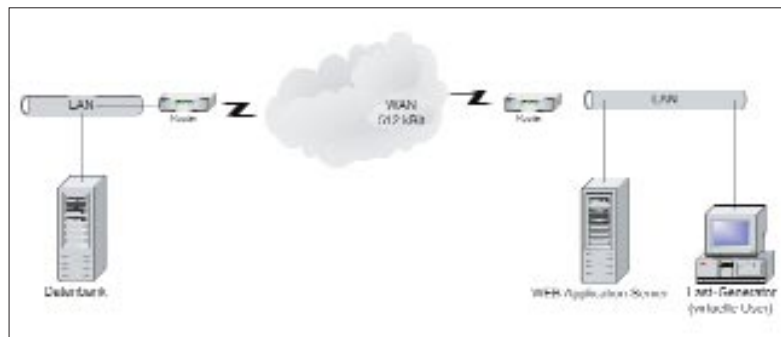


Abbildung 1 Zielinfrastruktur des Webshops und Aufbau des Lasttests.

- 50 Anwender führen gleichzeitig einen definierten Geschäftsprozess (Bestellvorgang) aus.
- Die Gesamtantwortzeit dieses Prozesses darf maximal 100 Sekunden betragen.

Bereits bei 17 Anwendern, das sind 34 Prozent der geforderten Anzahl von insgesamt 50 Anwendern, können die Anforderungen nicht mehr erfüllt werden und die maximale Antwortzeit wird erreicht. Gegen Ende des Lasttests wird die geforderte Antwortzeit um das Fünffache überschritten.

Nach einem solchen Ergebnis stellen sich den Verantwortlichen eine Reihe von Fragen: Löst eine größere Bandbreite die Probleme? Wie sieht es mit den zusätzlichen Kosten aus? Ist der Termin zu halten? Es sollte aber auch danach gefragt werden, ob diese Situation zu vermeiden gewesen wäre, wenn man das Problem frühzeitig erkannt hätte.

## Das Problem einkreisen

Bereits während der funktionalen Tests lässt sich das Antwortzeitverhalten der Software in der künftigen Zielumgebung prüfen. Dies ist beispielsweise mit dem Werkzeug Application Expert von Com-

puware möglich. Mit Application Expert können umfassende Hochrechnungen bezüglich Antwortzeiten und WAN-Auslastung durchgeführt werden. Darüber hinaus liefert es detaillierte Einblicke in das Verhalten verteilter Anwendungen, die für gezielte Fehlersuche und Tuning mehr als hilfreich sind. Um derartige Hochrechnungen zu ermöglichen, sniffert das Werkzeug in der Testumgebung den Netzwerkverkehr eines Clients, der den Musterprozess ausführt.

Da der Netzwerkverkehr passiv gemessen wird, bleibt das Laufzeitverhalten der Anwendung unverändert.

Zuerst prüft Application Expert, wie sich die Antwortzeit eines Anwenders in der Zielumgebung verhalten wird. Dafür rechnet das Werkzeug den aufgezeichneten Netzwerkverkehr hoch. Gemessen wird in einer „idealen Welt“. Das heißt, alle Maschinen sind an einem Hub angeschlossen und ein Anwender führt den Geschäftsprozess aus. Anschließend werden dem Werkzeug die künftigen Bandbreiten und Latenzzeiten bekannt gegeben. Client und Web Application Server sind im selben LAN-Segment (100 MBit/s). Die Datenbank ist über eine 512-KBit/s-Leitung angebunden, wie es Abbildung 1 zeigt.

Der obere Balken von Abbildung 2 zeigt die tatsächlich gemessene Antwort-

zeit in der Testumgebung für einen Anwender. Der Balken in der Mitte ist die hochgerechnete Antwortzeit unter der Annahme, dass die 512-Kbit/s-Leitung zwischen Web Application Server und Datenbank eine Latenzzeit von 30 Millisekunden hat. Das ist ein guter Wert für ein WAN. Ganz unten wurde zusätzlich angenommen, dass der Client über eine ISDN-Leitung angebunden ist.

Die ermittelten Werte sind in Bezug auf die geforderte maximale Antwortzeit sehr wichtig. Die 17,1 Sekunden, die der Anwender in der Testumgebung erreicht hat, ist die kleinstmögliche Zeit, denn es gibt keine besseren Bedingungen, als dass ein Benutzer im LAN die Transaktion völlig allein ausführt. Die Hochrechnung ergibt unter idealen Bedingungen in der Zielumgebung eine Antwortzeit von 46 Sekunden. Das ist die schnellstmögliche Zeit in der Produktionsumgebung.

Der Entwickler kennt damit das Laufzeitverhalten seiner Anwendung in der geplanten Zielumgebung. Und er weiß, dass sich die Antwortzeit fast verdreifachen wird.

Das ist eine sehr wichtige Erkenntnis. Entscheidend ist aber, wie sich das Ganze verhalten wird, wenn 50 Anwender diesen Prozess parallel ausführen. Um diese Berechnung anzustellen, wird im Application Expert ein Benutzerprofil erstellt, in dem hinterlegt ist, welche und vor allem wie viele Transaktionen ausgeführt werden. Im Beispiel wird davon ausgegangen, dass die 50 Anwender den Prozess jeweils 40-mal pro Stunde ausführen.

Das Werkzeug ermittelt die Auslastung im WAN: In der einen Richtung besteht eine Überlastung von 108,2 Prozent und in der Gegenrichtung sogar von über 150 Prozent.

Das WAN-Provisioning nimmt das Ergebnis des Lasttests vorweg. Bereits zu einem wesentlich früheren Zeitpunkt lässt sich mit geringerem Aufwand feststellen, dass die Leitung überlastet sein wird. Eine mögliche Lösung wäre, die Bandbreite zu erhöhen. Bevor dies geschieht, wird in der Regel überprüft, ob Optimierungspotenzial vorhanden ist. Dazu wird der Netzwerkverkehr eines Anwenders in der Testumgebung genauer untersucht.

### Den Netzwerkverkehr analysieren

Abbildung 3 zeigt ein so genanntes Bounce-Diagramm. Hier ist auf einen Blick zu erkennen, wo zwischen zwei Tiers viel oder wenig Kommunikation

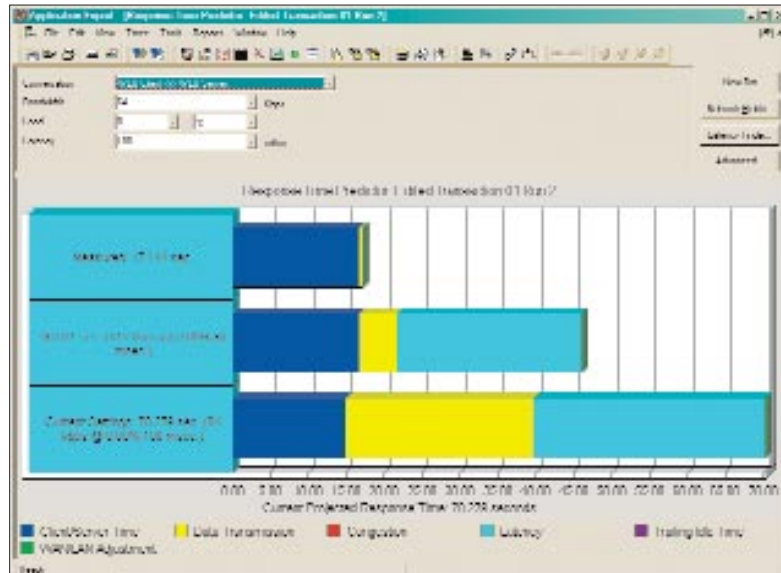


Abbildung 2 Hochrechnung der Antwortzeit in der Zielumgebung für einen Anwender.

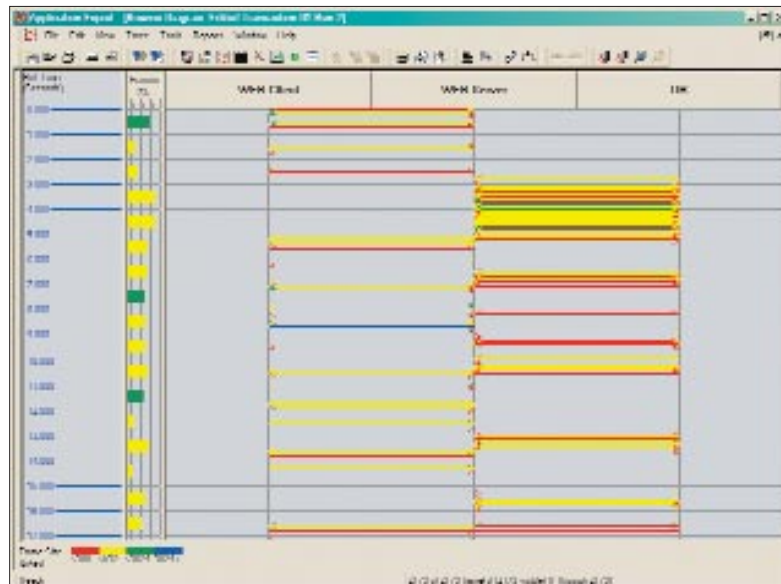


Abbildung 3 Bounce-Diagramm eines Anwenders in der Testumgebung. Jeder dargestellte Pfeil ist ein Netzwerkpaket, das zwischen den Tiers ausgetauscht wurde.

stattfindet. Nicht nur die Anzahl der Pakete ist zu sehen, sondern auch die Richtung und ihre Größe. Gelbe und rote Pakete sind sehr klein, das heißt, sie haben wenig Dateninhalt. Die grünen und blauen Pakete hingegen transportieren eine wesentlich größere Datenmenge.

Am Beginn der Transaktion fällt ein Bereich zwischen Sekunde 3 und 5 auf, in dem der Web Application Server sehr viele kleine Pakete mit der Datenbank austauscht. Das sind schlechte Vorausset-

zungen für ein WAN, da sich zu jedem Paket die Latenzzeit addiert. Eine Analyse der einzelnen Pakete würde Aufschluss darüber bringen, was gerade an dieser Stelle der Anwendung passiert. Das ist aber nicht nur sehr mühsam, sondern auch unnötig. Das Werkzeug ist nämlich in der Lage, den Netzwerkverkehr zu decodieren und zu so genannten Threads zusammenzusetzen. Das macht es für den Entwickler einfach, die Anwendung aus Netzwerksicht zu „lesen“.

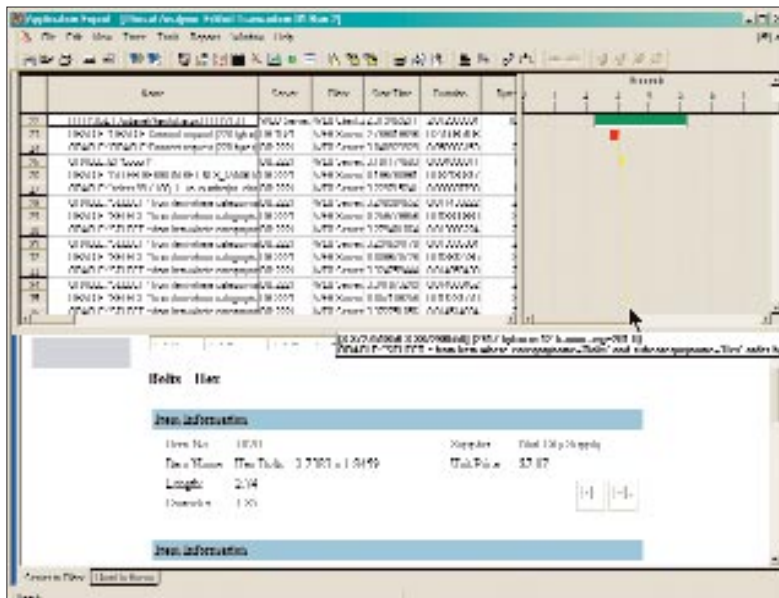


Abbildung 4 Decodierter Netzwerkverkehr eines Anwenders.

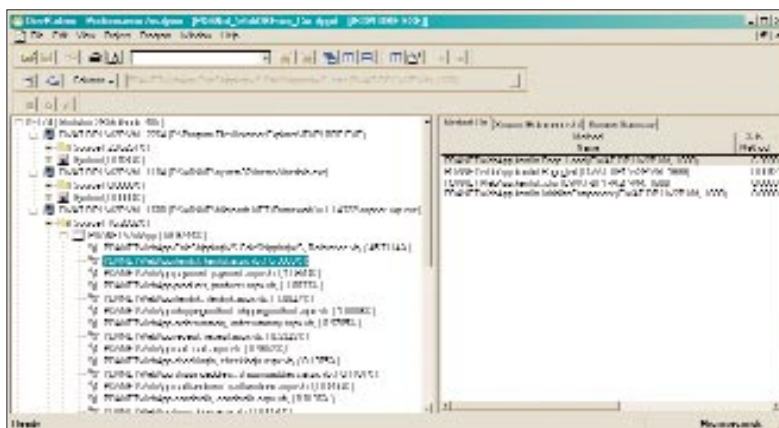


Abbildung 5 Performance-Messung der CPU-Zeit auf Methodenebene.

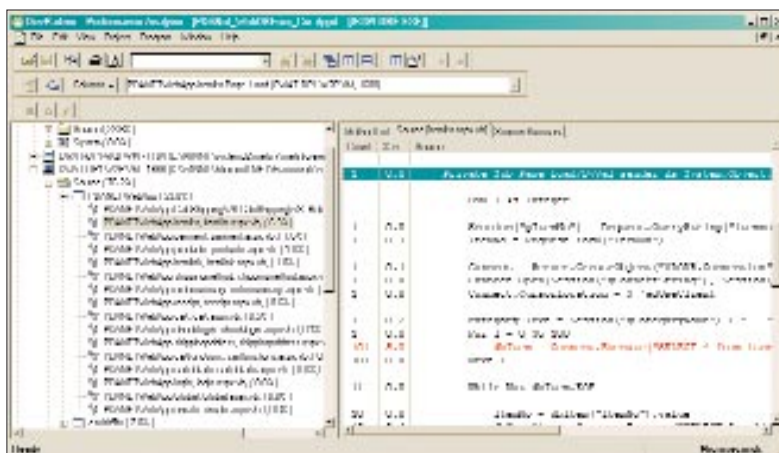


Abbildung 6 Ergebnisse der Performance-Messung auf Sourcecode-Ebene.

Abbildung 4 zeigt den decodierten Netzwerkverkehr. In der oberen Hälfte sieht man die einzelnen Threads. In diesem Fall sind das HTTP- und Oracle-Statements, die nach ihrer Startzeit sortiert sind. Unten wird der Inhalt des markierten HTTP-GET-Statements *Item-List.aspx* dargestellt.

Der langsamste Thread ist identifiziert. Es handelt sich um ein HTTP-GET der Seite *ItemList.aspx*. Auch die Datenbankstatements werden decodiert. Parallel zu *ItemList.aspx* werden unzählige Selects auf die Datenbank abgesetzt. Das Problem ist nun schon deutlich eingegrenzt. Um es endgültig zu lösen, wird der Sourcecode von *ItemList.aspx* geprüft.

### CPU-Zeiten analysieren

Damit eine fundierte Aussage getroffen werden kann, misst man die CPU-Zeiten aller beteiligten Tasks, den Client in Form des Webbrowsers (*Explore.exe*), den Webserver (*Inetinfo.exe*) und das ASP.NET-Framework (*Aspnet\_ws.exe*). Da bereits bekannt ist, dass das Problem seine Ursache auf der Seite *ItemList.aspx* hat, wird gezielt im Task *Aspnet\_ws.exe* gesucht. Die Daten werden mit DevPartner Studio Professional Edition von Compuware gemessen, einer Lösung, die sowohl integriert in Visual Studio arbeitet als auch im verteilten Umfeld im selbstständigen Betrieb.

Im Beispiel laufen alle beteiligten Komponenten auf einer Maschine, mit Ausnahme der Datenbank. Der Internet Explorer, der Internet Information Server und das .NET Framework werden für eine Performance-Messung konfiguriert. Dabei wird DevPartner mitgeteilt, was bis zu welcher Tiefe gemessen werden soll. Danach kann der Geschäftsprozess ausgeführt werden. Im Gegensatz zur Analyse des Netzwerkverkehrs handelt es sich hier um ein aktives Messverfahren. Die Ergebnisse werden dadurch aber nicht beeinflusst, da ein Treiber direkt an der CPU für jeden Taktzyklus bestimmt, welcher Task im Augenblick läuft. Das garantiert konsistente Messergebnisse, egal welche anderen Programme oder Systemtasks parallel zur Messung laufen. Um die Messung zu kontrollieren, steht für jeden Task ein eigenes Steuerelement zur Verfügung.

Nach dem Starten des Webservers und Laden des Webshops werden bei allen Tasks die Daten gelöscht, damit nur der Geschäftsprozess gemessen wird. Würde

man dies nicht tun, wären auch die CPU-Zeiten für Start und Initialisierung in den Ergebnissen enthalten.

Die DevPartner-Performance-Analyse in Abbildung 5 misst die CPU-Zeiten der einzelnen Komponenten. In der linken Fensterhälfte sind die beteiligten Tasks dargestellt. Die Prozentangaben beziehen sich auf die CPU-Zeit, wobei 100 % der gesamten CPU-Zeit während der Messung entspricht. Im rechten Bereich sind die beteiligten Methodenaufrufe aufgelistet. Es zeigt sich, wie hoch der Einfluss des Netzwerkes auf die Gesamt-Performance einer verteilten Anwendung ist.

Der identifizierte Flaschenhals *ItemList.aspx* verbraucht nur 6,3 Prozent der CPU-Zeit. Viel größeres Potenzial lässt der Web Service *CalcShippingWS* vermuten, der über 45 Prozent der CPU-Zeit konsumiert.

In der rechten Fensterhälfte in Abbildung 6 ist der Sourcecode der Methode *ItemList.Page\_Load* dargestellt. Die Spalte *Count* zeigt, wie oft jede Zeile ausgeführt wurde. Die Prozentzahl dahinter entspricht wieder dem Anteil an der verbrauchten CPU-Zeit. Die rote Zeile ist die langsamste in dieser Methode.

Damit ist die Ursache für die verstopfte Leitung offensichtlich. Die rote Zeile zeigt ein SQL-Statement, das in einer *For*-Schleife 101-mal ausgeführt wurde. Um die Tragweite dieser Codezeile bezüglich der Anforderungen zu verdeutlichen, folgt eine Berechnung, die den Durchschnitt der Select-Statements pro Sekunde liefert:

50 Anwender x 101 Selects / 100 Sekunden = 50 Selects / Sekunde.

Diese eine Zeile sorgt dafür, dass im Echtbetrieb im Durchschnitt 50 Select-Statements pro Sekunde abgearbeitet und über das WAN übertragen werden müssen. Im Beispiel handelt es sich um eine sinnlose Schleife, die zu Demonstrationszwecken bewusst eingebaut wurde. Aus der Praxis ist bekannt, dass die Architektur beziehungsweise der Inhalt von Schleifen sehr oft einen Performance-Engpass darstellen. Solche Schleifen lassen sich durch Tuning des Codes leicht optimieren.

## Die Lösung

Um die Folgen des Tunings zu überprüfen, werden Messung und Analyse des Netzwerkverkehrs in der Testumgebung

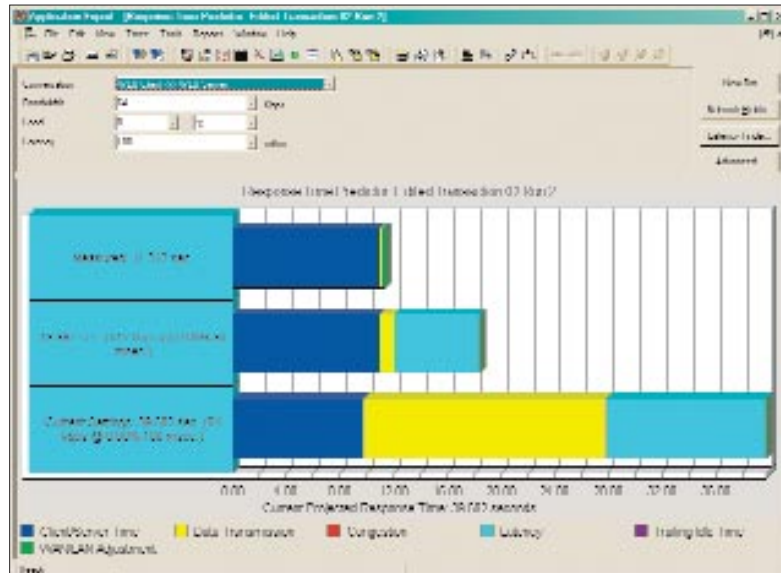


Abbildung 7 Hochrechnung der Antwortzeit in die Zielumgebung, zweite Messung.

wiederholt. Abbildung 7 zeigt das Ergebnis.

Wie beim ersten Mal zeigt der obere Balken die tatsächlich gemessene Antwortzeit in der Testumgebung für einen Anwender. Der Balken in der Mitte ist die hochgerechnete Antwortzeit unter der Annahme, dass die 512-Kbps-Leitung zwischen Web Application Server und Datenbank eine Latenzzeit von 30 Millisekunden hat. Ganz unten wurde zusätzlich angenommen, dass der Client über eine ISDN-Leitung angebunden ist.

Tabelle 1 zeigt, dass das Tuning den erwarteten Effekt zeigt. In der Testumgebung wird die Antwortzeit um 34 Prozent verkürzt. Deutlich größer sind die Auswirkungen in der Zielumgebung. Gegenüber der ersten Messung kann die Antwortzeit um 60 Prozent unterboten werden.

Zum Abschluss berechnet das Werkzeug die Auslastung des WANs. Auch hier ist eine deutliche Verbesserung gegenüber der ersten Messung zu erwarten.

Die maximale Auslastung der 512-KBit/s-Leitung liegt nur noch bei 35 Pro-

zent, wenn 50 Anwender den Geschäftsprozess gleichzeitig ausführen. Das heißt, der Verkehr konnte um über 75 Prozent verringert werden. Der abschließende Lasttest in der Zielumgebung erfüllt alle geforderten Anforderungen. Die Service Level Agreements werden eingehalten.

## Fazit

Die Einbeziehung der Zielumgebung zum frühestmöglichen Zeitpunkt ist nicht nur in Bezug auf die Performance und die geforderten Antwortzeiten ein wichtiger Aspekt. Auch die Bandbreitenbestimmung von WAN-Strecken ist so frühzeitig möglich. Die Darstellung des decodierten Netzwerkverkehrs in so genannten Threads ist für den Entwickler ein äußerst nützliches Werkzeug bei der Analyse von verteilten Anwendungen. Sie gibt ihm die Möglichkeit eines Reverse Engineerings aus Netzwerksicht. Natürlich werden Lasttests dadurch nicht überflüssig. Sie sollten allerdings nur die Bestätigung für eine hohe Software-Qualität sein. |||||

## Tabelle 1

### Erfolgreiches Tuning.

	Anforderung	Testumgebung	Zielumgebung	Client mit ISDN
Antwortzeit Messung 1	< 100 Sek.	≥ 17,1 Sek.	≥ 46 Sek.	≥ 70,3 Sek.
Antwortzeit Messung 2	< 100 Sek.	≥ 11,3 Sek.	≥ 18,4 Sek.	≥ 39,7 Sek.
Performance-Steigerung		34 %	60 %	45 %