

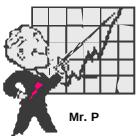
Frischer Wind in die IT

stop on quality

Vortrag Performance-Management

Axel Goldbach
modulo3 gmbh

Karl-Rudolf-Str. 172
40215 Düsseldorf
fon +49 (211) 87672000
fax +49 (211) 87672027
www.modulo3.de



Mr. P

©2001 modulo3 gmbh

Agenda

stop on quality

- **Einflußfaktoren und Analysemethoden**
 - Performance-Testing-Tools
 - Laufzeitschätzung
- **Beispielproblem**
 - Problem und Lösung
 - Client- und Server-Code
- **Langsamer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Ein Märchen**
- **Schnellerer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Asymptotischer Vergleich**

©2001 modulo3 gmbh

PerformanceManagement

Page 2

Agenda

stop on quality

- **Einflußfaktoren und Analysemethoden**
 - Performance-Testing-Tools
 - Laufzeitschätzung
- **Beispielproblem**
 - Problem und Lösung
 - Client- und Server-Code
- **Langsamer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Ein Märchen**
- **Schnellerer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Asymptotischer Vergleich**

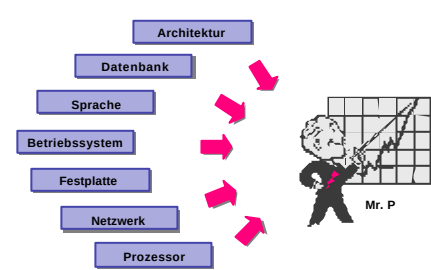
©2001 modulo3 gmbh

PerformanceManagement

Page 3

Einflußfaktoren

stop on quality



Architektur

Datenbank

Sprache

Betriebssystem

Festplatte

Netzwerk

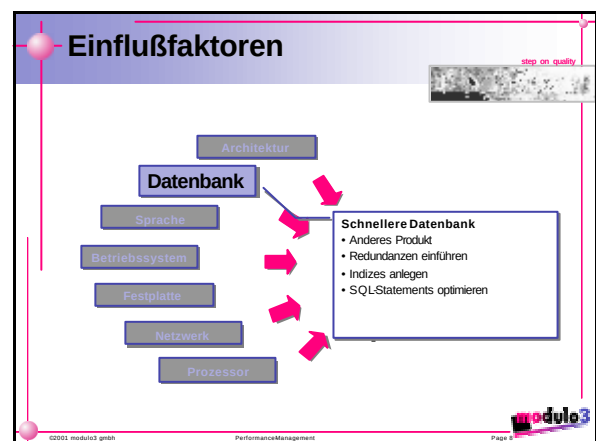
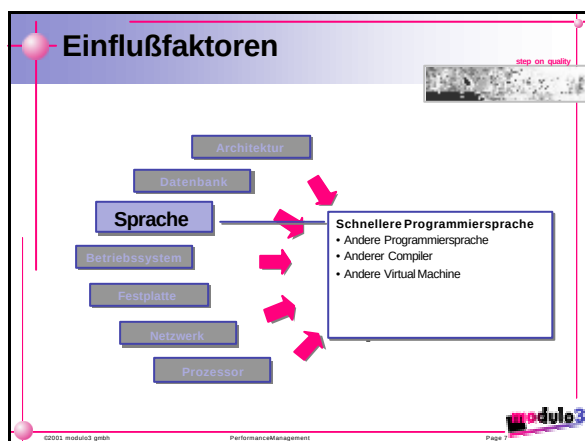
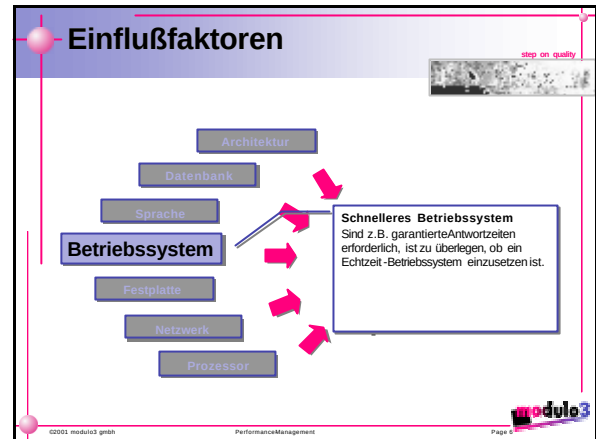
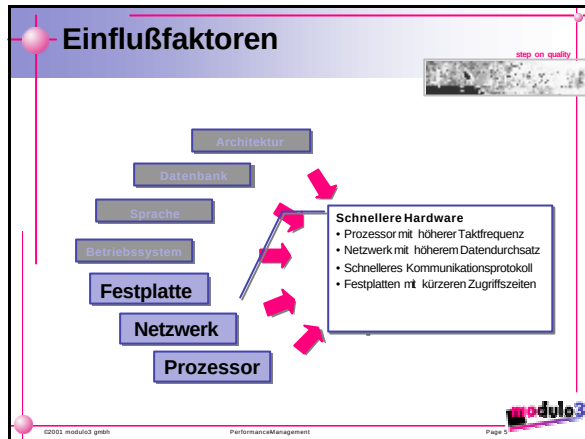
Prozessor

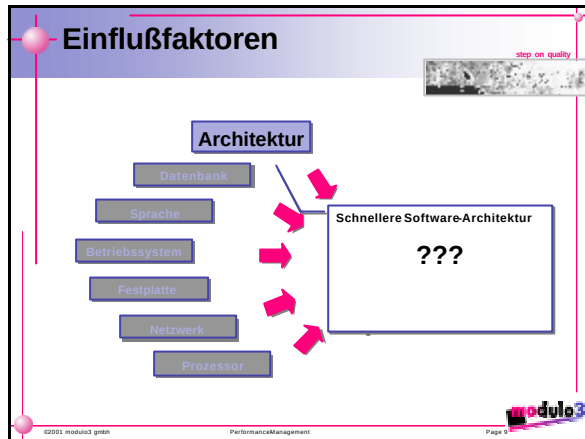
Mr. P

©2001 modulo3 gmbh

PerformanceManagement

Page 4





Analysemethoden

Wie analysiert man Performance-Probleme?

- Performance-Testing-Tools
- Laufzeitabschätzung

©2001 modulo3 gmbh PerformanceManagement Page 10

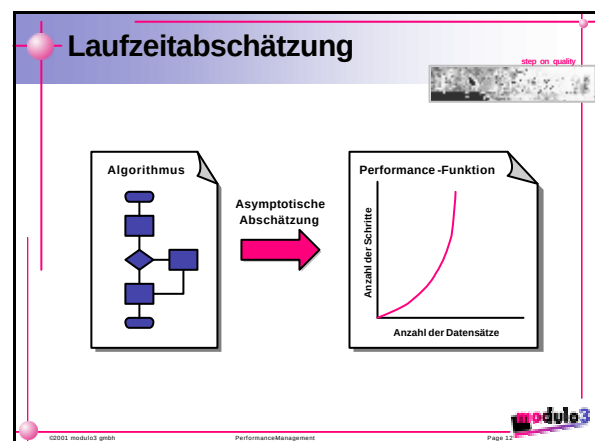
Performance-Testing-Tools

Was machen Performance-Testing-Tools?

- Testen das Laufzeitverhalten einer Applikation
- Die genaue Bestimmung der Laufzeit unter Berücksichtigung der benutzten Programmiersprache, des verwendeten Betriebssystems und der Hardware-Plattform
- Die Betrachtung des Verhaltens eines Programms unter Laufzeitbedingungen

KEINE allgemeingültige Aussage über das Laufzeitverhalten eines Algorithmus

©2001 modulo3 gmbh PerformanceManagement Page 11



Laufzeitabschätzung

Was ist Laufzeitabschätzung?

- Asymptotische Betrachtung der Laufzeit eines Algorithmus
- Basiert z.B. auf der Anzahl der zu verarbeitenden Datensätze
- Betrachtung des allgemeinen Verhaltens von Algorithmen im Vergleich zu anderen Algorithmen
- Problemorientiert, nicht implementationsabhängig

KEINE Aussage über das spezielle Laufzeitverhalten einer Implementation

©2001 modulo3 gmbh

PerformanceManagement

Page 13

modulo3

Agenda

- Einflussfaktoren und Analysemethoden
 - Performance-Testing-Tools
 - Laufzeitabschätzung
- **Beispielproblem**
 - Problem und Lösung
 - Client- und Server-Code
- **Langsamer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Ein Märchen**
- **Schnellerer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Asymptotischer Vergleich**

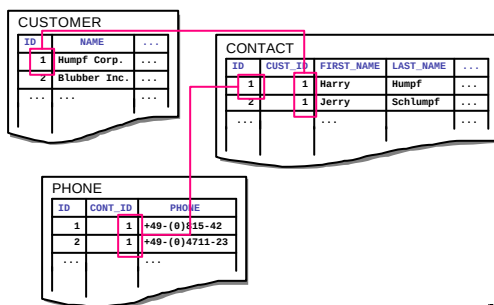
©2001 modulo3 gmbh

PerformanceManagement

Page 14

modulo3

Beispielproblem



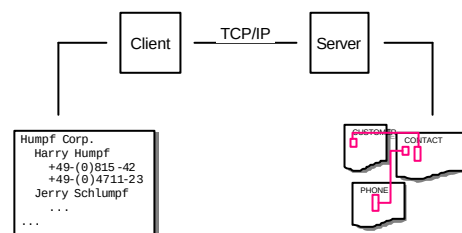
©2001 modulo3 gmbh

PerformanceManagement

Page 15

modulo3

Gesuchte Lösung



©2001 modulo3 gmbh

PerformanceManagement

Page 16

modulo3

Formate

Request-Format

- <TYPE>:<INDEX>
- z.B. CUSTOMER:0

Response-Format

- <NAME> bzw. <PARENT-ID>:<NAME>
- z.B. Customer0 bzw. 0:Contact1 bzw. 1:Phone2

stop on quality

©2001 modulo3 gmbh PerformanceManagement Page 17

Server (Basis)

```

TcpListener listener = new TcpListener ( 4711 );
listener.Start ();
TcpClient server = listener.AcceptTcpClient ();

NetworkStream output = server.GetStream ();
StreamReader input = new StreamReader ( output );

// receive requests and return responses

listener.Stop ();
server.Close ();
    
```

stop on quality

©2001 modulo3 gmbh PerformanceManagement Page 18

Server (Request-Bearbeitung)

```

string request = input.ReadLine ();
while ( ( request != null ) && !request.Equals ( "" ) ) {
    string[] req = request.Split ( new Char[] { ':' } );

    string what = req[0].ToUpper ();
    int index = Convert.ToInt32 ( req[1] );

    string result = null;
    if ( what.Equals ( "CUSTOMER" ) )
        result = customers[index];
    else if ( what.Equals ( "CONTACT" ) )
        result = contacts[index];
    else if ( what.Equals ( "PHONE" ) )
        result = phones[index];

    Byte[] res = Encoding.ASCII.GetBytes ( ( result + "\n" ).ToCharArray () );
    output.Write ( res, 0, res.Length );
    request = input.ReadLine ();
}
    
```

stop on quality

©2001 modulo3 gmbh PerformanceManagement Page 19

Client (Basis)

```

TcpClient client = new TcpClient ( "127.0.0.1", 4711 );

NetworkStream output = client.GetStream ();
StreamReader input = new StreamReader ( output );

DateTime start = DateTime.Now;

if ( slow )
    slowExecution ( input, output, numberOfCustomers );
else
    fasterExecution ( input, output, numberOfCustomers );

DateTime stop = DateTime.Now;

Console.WriteLine ( counter + " requests, duration: " +
    stop.Subtract ( start ).TotalMilliseconds );

client.Close ();
    
```

stop on quality

©2001 modulo3 gmbh PerformanceManagement Page 20

Client (Request senden)

```
public static string[] sendRequest ( StreamReader input,
    NetworkStream output, string request ) {
    counter++;

    // send request
    byte[] req = Encoding.ASCII.GetBytes (
        (request + "\r\n").ToCharArray () );
    output.Write ( req, 0, req.Length );

    // receive response
    string resp = input.ReadLine ();
    return resp.Split ( new char[] {','} );
}
```

©2001 modulo3 gmbh

PerformanceManagement

Page 21



Agenda

- **Einflußfaktoren und Analysemethoden**
 - Performance-Testing-Tools
 - Laufzeitschätzung
- **Beispielproblem**
 - Problem und Lösung
 - Client- und Server-Code
- **Langsamer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- Ein Märchen
- **Schnellerer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- **Asymptotischer Vergleich**

©2001 modulo3 gmbh

PerformanceManagement

Page 22



Client (langsam)

```
public static void slowExecution ( StreamReader input, NetworkStream output,
    int numberOfCustomers ) {
    string[] resp = null;
    for ( int i = 0; i < numberOfCustomers; i++ ) {
        // get customer i
        Console.WriteLine ( sendRequest ( input, output, "CUSTOMER:" + i )[0] );

        for ( int j = 0; j < numberOfCustomers * 2; j++ ) {
            // get contact j
            resp = sendRequest ( input, output, "CONTACT:" + j );
            int contactCustomerIndex = Convert.ToInt32 ( resp[0] );
            if ( contactCustomerIndex == i )
                Console.WriteLine ( " " + resp[1] );
        }

        for ( int k = 0; k < numberOfCustomers * 4; k++ ) {
            // get phone k
            resp = sendRequest ( input, output, "PHONE:" + k );
            if ( ( contactCustomerIndex == i ) && Convert.ToInt32 ( resp[0] ) == j )
                Console.WriteLine ( " " + resp[1] );
        }
    }
}
```

©2001 modulo3 gmbh

PerformanceManagement

Page 23



Performance-Messung

Au Backe !!!

Anzahl Kunden	Requests	Dauer (ms)
5	1.055	411
10	8.210	2.674
15	27.465	8.542
20	64.820	20.099
25	126.275	38.926
30	217.830	67.086

©2001 modulo3 gmbh

PerformanceManagement

Page 24



Vereinfachung der Analyse

stop on quality

Elimination „uninteressanter“ Befehle

- Zuweisungen, Vergleiche, Unterfunktionsaufrufe

Diese Befehle benötigen
„konstante“ Zeit und sind relativ schnell
=> für asymptotische Analyse „uninteressant“

©2001 modulo3 gmbH PerformanceManagement Page 25

Vereinfachung der Analyse

stop on quality

Betrachtung „interessanter“ Befehle

- Netzwerkzugriffe, Schleifen

Diese Befehle sind relativ langsam
oder erhöhen Komplexität
=> für asymptotische Analyse „interessant“

©2001 modulo3 gmbH PerformanceManagement Page 26

Langsamer Algorithmus

stop on quality

```

for ( int i = 0; i < numberOfCustomers; i++ ) {
    // get customer i                !!! network access !!!
    for ( int j = 0; j < numberOfCustomers * 2; j++ ) {
        // get contact j                !!! network access !!!
        // if contact j is contact of customer i
        // add contact j to customer i
        for ( int k = 0; k < numberOfCustomers * 4; k++ ) {
            // get phone k                !!! network access !!!
            // if phone k is phone of contact j
            // add phone k to contact j
        }
    }
}
    
```

©2001 modulo3 gmbH PerformanceManagement Page 27

Langsamer Algorithmus

stop on quality

```

for ( int i = 0; i < numberOfCustomers; i++ ) {
    // get customer i                !!! network access !!!
    for ( int j = 0; j < numberOfCustomers * 2; j++ ) {
        // get contact j                !!! network access !!!
        // if contact j is contact of customer i
        // add contact j to customer i
        for ( int k = 0; k < numberOfCustomers * 4; k++ ) {
            // get phone k                !!! network access !!!
            // if phone k is phone of contact j
            // add phone k to contact j
        }
    }
}
    
```

©2001 modulo3 gmbH PerformanceManagement Page 28

Definitionen

stop on quality

Sei

- a die Anzahl aller Kunden
- b die Anzahl aller Ansprechpartner aller Kunden
- c die Anzahl aller Telefonnummern aller Ansprechpartner

©2001 modulo3 gmbh PerformanceManagement Page 29



Analyse langsamer Algorithmus

stop on quality

3. Schleife

- 1 Netzwerkzugriff
- c Wiederholungen

2. Schleife

- 1 Netzwerkzugriff
- Aufruf der 3. Schleife
- b Wiederholungen

1. Schleife

- 1 Netzwerkzugriff
- Aufruf der 2. Schleife
- a Wiederholungen

©2001 modulo3 gmbh PerformanceManagement Page 30



Analyse langsamer Algorithmus

stop on quality

3. Schleife

- 1 Netzwerkzugriff
- c Wiederholungen

$c \cdot 1 = c$

2. Schleife

- 1 Netzwerkzugriff
- Aufruf der 3. Schleife
- b Wiederholungen

$b \cdot (1 + <3. \text{ Schleife} >) = b \cdot (1 + c)$

1. Schleife

- 1 Netzwerkzugriff
- Aufruf der 2. Schleife
- a Wiederholungen

$a \cdot (1 + <2. \text{ Schleife} >) = a \cdot (1 + b \cdot (1 + c))$

©2001 modulo3 gmbh PerformanceManagement Page 33

Analyse langsamer Algorithmus

stop on quality

Insgesamt $a \cdot (1 + b \cdot (1 + c))$ Netzwerkzugriffe

- Für $a=100$, $b=200$, $c=400$:
 $100 \cdot (1 + 200 \cdot 401) = 8.020.100$ Zugriffe

Mit einer Request-Dauer von 2ms Zugriffszeit braucht der Algorithmus also

$8.020.100 \cdot 2\text{ms} = 16.040.200\text{ms} = 4,45\text{h}$

©2001 modulo3 gmbh PerformanceManagement Page 34

Agenda

stop on quality

- Einflußfaktoren und Analysemethoden
 - Performance-Testing-Tools
 - Laufzeitschätzung
- Beispielproblem
 - Problem und Lösung
 - Client- und Server-Code
- Langsamer Algorithmus
 - Performance-Messung
 - Performance-Analyse
- Ein Märchen**
- Schnellerer Algorithmus
 - Performance-Messung
 - Performance-Analyse
- Asymptotischer Vergleich

©2001 modulo3 gmbh PerformanceManagement Page 35

Ein Märchen

stop on quality

Da sagt plötzlich einer:

Ich kenne jemanden, der hat eine Netzwerktechnologie für Dich, die den doppelten Datendurchsatz hat.

Dann halbiert sich die Zeit, die Dein Algorithmus benötigt.

©2001 modulo3 gmbh PerformanceManagement Page 36

Ein Märchen

stop on quality



Da hat er recht:

$8.020.100 * 1\text{ms} =$
 $8.020.100\text{ms} = 2,23\text{h}$
(ca. $\frac{1}{2}$ von 4,45h)

©2001 modulo3 gmbh PerformanceManagement Page 31

modulo3

Ein Märchen

stop on quality



Und weiter:

Da wir nun den doppelten Datendurchsatz haben, können wir in derselben Zeit, die die alte Netzwerktechnologie benötigt hat, die doppelte Menge an Datensätzen verarbeiten.

©2001 modulo3 gmbh PerformanceManagement Page 31

modulo3

Ein Märchen

stop on quality



Arrrgrrrh...

©2001 modulo3 gmbh PerformanceManagement Page 32

modulo3

Ein Märchen

stop on quality



**Halbieren wir die Zugriffszeit
Verdoppeln wir die Anzahl der aller Datensätze**

- Sei also $a=200$, $b=400$, $c=800$ und die Zugriffszeit 1ms
- Daraus ergibt sich:
 $200 * (1 + 400 * 801) = 64.080.200$ Netzwerkzugriffe

Der Algorithmus benötigt also
 $64.080.200 * 1\text{ms} = 17,8\text{h}$
(ca. 8 mal 2,23h)

©2001 modulo3 gmbh PerformanceManagement Page 32

modulo3

Ein Märchen

stop on quality

Trotz Halbierung der einzelnen Zugriffszeit ergibt sich bei Verdoppelung aller Datensätze fast eine Verachtfachung der Gesamtzugriffszeit !!!

Wir brauchen also einen schnelleren Algorithmus !

©2001 modulo3 gmbh PerformanceManagement Page 41

Agenda

stop on quality

- Einflußfaktoren und Analysemethoden
 - Performance-Testing-Tools
 - Laufzeitschätzung
- Beispielproblem
 - Problem und Lösung
 - Client- und Server-Code
- Langsamer Algorithmus
 - Performance-Messung
 - Performance-Analyse
- Ein Märchen
- Schnellerer Algorithmus**
 - Performance-Messung
 - Performance-Analyse
- Asymptotischer Vergleich

©2001 modulo3 gmbh PerformanceManagement Page 42

Client (schneller)

stop on quality

```

public static void fasterExecution ( StreamReader input, NetworkStream output,
int numberOfCustomers ) {
    string[] resp = null;
    for ( int i = 0; i < numberOfCustomers; i++ ) {
        // get customer i
        Console.WriteLine ( sendRequest ( input, output, "CUSTOMER:" + i )[0] );

        for ( int j = 0; j < numberOfCustomers * 2; j++ ) {
            // get contact j
            resp = sendRequest ( input, output, "CONTACT:" + j );
            int contactCustomerIndex = Convert.ToInt32 ( resp[0] );
            if ( contactCustomerIndex == i ) {
                Console.WriteLine ( " " + resp[1] );
            }

            for ( int k = 0; k < numberOfCustomers * 4; k++ ) {
                // get phone k
                resp = sendRequest ( input, output, "PHONE:" + k );
                if ( Convert.ToInt32 ( resp[0] ) == j )
                    Console.WriteLine ( " " + resp[1] );
            }
        }
    }
}
    
```

©2001 modulo3 gmbh PerformanceManagement Page 43

Schnellerer Algorithmus

stop on quality

```

for ( int i = 0; i < numberOfCustomers; i++ ) {
    // get customer i
    // !!! network access !!!

    for ( int j = 0; j < numberOfCustomers * 2; j++ ) {
        // get contact j
        // !!! network access !!!

        if ( contact j is contact of customer i ) {
            // add contact j to customer i

            for ( int k = 0; k < numberOfCustomers * 4; k++ ) {
                // get phone k
                // !!! network access !!!

                // if phone k is phone of contact j
                // add phone k to contact j
            }
        }
    }
}
    
```

©2001 modulo3 gmbh PerformanceManagement Page 44

Performance-Messung

Schon besser !!!

Anzahl Kunden	Langsam		Schneller		Beschleunigung	
	Requests	Dauer (ms)	Requests	Dauer (ms)	Requests	Dauer
5	1.055	411	255	140	? 4	? 3
10	8.210	2.674	1.010	401	? 8	? 7
15	27.465	8.542	2.265	851	? 12	? 10
20	64.820	20.099	4.020	1.402	? 16	? 13
25	126.275	38.926	6.275	2.193	? 20	? 18
30	217.830	67.086	9.030	3.405	? 24	? 20

©2001 modulo3 gmbh PerformanceManagement Page 43

Analyse schnellerer Algorithmus

3. Schleife

- 1 Netzwerkzugriff
- c Wiederholungen

2. Schleife

- 1 Netzwerkzugriff
- b Wiederholungen
- Im worst-case hat jeder Kunde die gleiche Anzahl von Ansprechpartnern, also wird bei jedem d-ten Durchlauf die 3. Schleife aufgerufen, d ist die durchschnittliche Anzahl der Ansprechpartner pro Kunde

©2001 modulo3 gmbh PerformanceManagement Page 44

Analyse schnellerer Algorithmus

3. Schleife

- 1 Netzwerkzugriff
- c Wiederholungen

$c \cdot 1 = c$

2. Schleife

- 1 Netzwerkzugriff
- b Wiederholungen
- Im worst-case hat jeder Kunde die gleiche Anzahl von Ansprechpartnern, also wird bei jedem d-ten Durchlauf die 3. Schleife aufgerufen, d ist die durchschnittliche Anzahl der Ansprechpartner pro Kunde

©2001 modulo3 gmbh PerformanceManagement Page 45

Analyse schnellerer Algorithmus

3. Schleife

- 1 Netzwerkzugriff
- c Wiederholungen

$c \cdot 1 = c$

2. Schleife

- 1 Netzwerkzugriff
- b Wiederholungen
- Im worst-case hat jeder Kunde die gleiche Anzahl von Ansprechpartnern, also wird bei jedem d-ten Durchlauf die 3. Schleife aufgerufen, d ist die durchschnittliche Anzahl der Ansprechpartner pro Kunde

$b \cdot 1 + d \cdot c < 3 \cdot \text{Schleife} \Rightarrow b + d \cdot c$

©2001 modulo3 gmbh PerformanceManagement Page 46

Analyse schnellerer Algorithmus

stop on quality

1. Schleife

- 1 Netzwerkzugriff
- Aufruf der 2. Schleife
- a Wiederholungen

©2001 modulo3 gmbh PerformanceManagement Page 43

Analyse schnellerer Algorithmus

stop on quality

1. Schleife

- 1 Netzwerkzugriff
- Aufruf der 2. Schleife
- a Wiederholungen

$a \cdot (1 + c \cdot \text{Schleife}) = a \cdot (1 + b + d \cdot c)$

©2001 modulo3 gmbh PerformanceManagement Page 54

Analyse schnellerer Algorithmus

stop on quality

Insgesamt $a \cdot (1 + b + d \cdot c)$ Netzwerkzugriffe

- Für $a=100$, $b=200$, $c=400$ und $d=2$ sind also $100 \cdot (201 + 2 \cdot 400) = 100.100$ Netzwerkzugriffe
- Für eine Netzwerk mit 2ms Zugriffszeit braucht der Algorithmus also $100.100 \cdot 2\text{ms} = 200.200\text{ms} = 3,33\text{min}$

©2001 modulo3 gmbh PerformanceManagement Page 55

Agenda

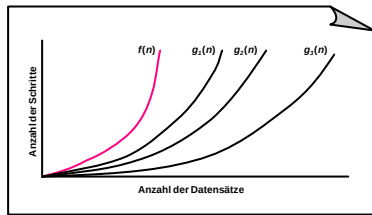
stop on quality

- Einflussfaktoren und Analysemethoden
 - Performance-Testing-Tools
 - Laufzeitschätzung
- Beispielproblem
 - Problem und Lösung
 - Client - und Server-Code
- Langsamer Algorithmus
 - Performance-Messung
 - Performance-Analyse
- Ein Märchen
- Schnellerer Algorithmus
 - Performance-Messung
 - Performance-Analyse
- Asymptotischer Vergleich

©2001 modulo3 gmbh PerformanceManagement Page 56

Asymptotik

Ein Bild zur Verdeutlichung



Asymptotischer Vergleich

Vereinfachung der Formel (langsamer Algorithmus)

Sei $n \geq \max\{a, b, c\}$

$$a \cdot n + b \cdot n + c \leq n^2 + n$$

Für große n gilt: $n^2 + n \approx n^2$

$$a \cdot n + b \cdot n + c \leq n^3$$

Asymptotischer Vergleich

Vereinfachung der Formel (schnellerer Algorithmus)

Sei $d \geq \frac{b}{a}, a \geq 0$

$$a \cdot n + b \cdot d \leq a \cdot n^2 + b \cdot \frac{b}{a} \leq a \cdot n^2 + bc$$

Sei $n \geq \max\{a, b, c\}$

$$a \cdot n + b \cdot d \leq n^2 + 2n^2$$

Entwicklung von n^3 vs. $n+2n^2$

n	n^3	$n+2n^2$	Beschleunigung
400	64.000.000	320.400	≈ 200
800	512.000.000	1.280.800	≈ 400
1.600	4.096.000.000	5.121.600	≈ 800
3.200	32.768.000.000	20.483.200	≈ 1.600
6.400	262.144.000.000	81.926.400	≈ 3.200

In eigener Sache

stop on quality

Kontakt

axel.goldbach@modulo3.de

Aktueller Vortrag und C#-Sourcen

<http://www.modulo3.de/vortraege/vortraege.html>

©2001 modulo3 gmbh PerformanceManagement Page 51

Weitere Informationen

stop on quality

modulo3

www.modulo3.de

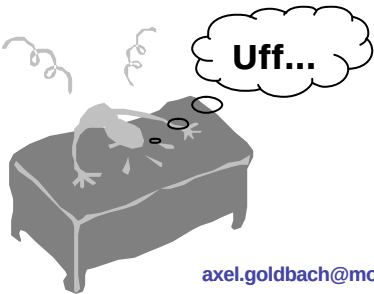
QualiMagic

www.qualimagic.de

©2001 modulo3 gmbh PerformanceManagement Page 52

Fragen ?

stop on quality



axel.goldbach@modulo3.de

©2001 modulo3 gmbh PerformanceManagement Page 53