Performance Testing von Webanwendungen: Eine kurze Einführung

fukami @ DevHouse Cologne 2007

Agenda

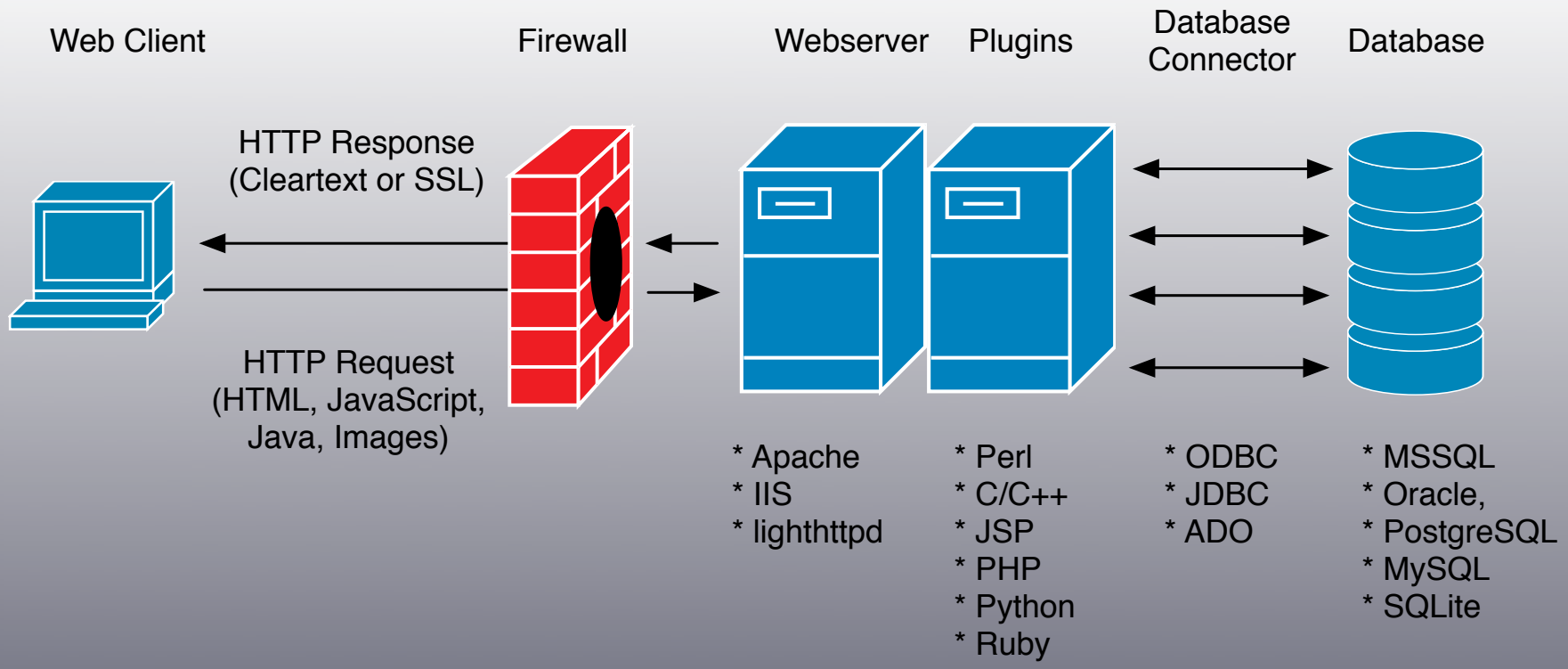
- Vorstellung des Vortragenden
- Typischer Development Cycle
- Typische Probleme
- Tools für Load und Analyse
- Skalierung/Verteilung
- Diskussion

Der Referent

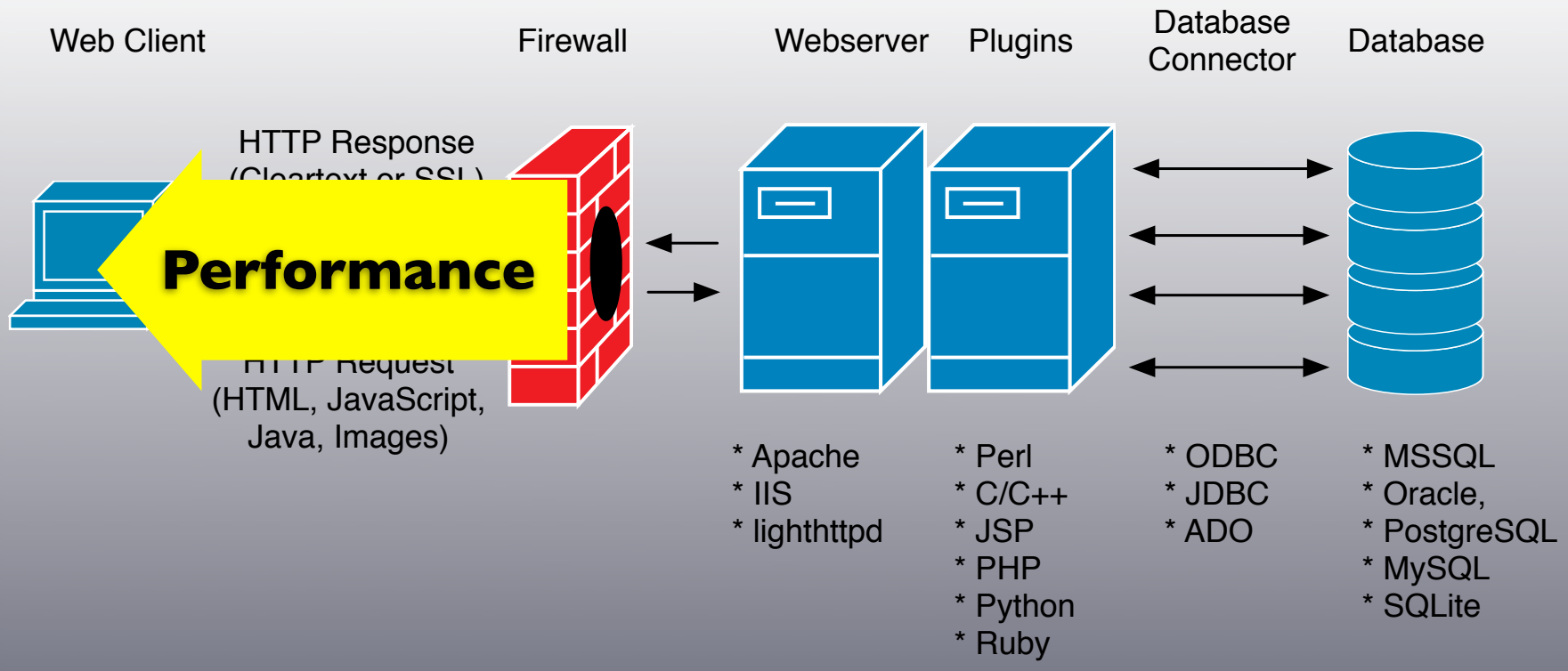
- arbeitet zur Zeit als Entwickler und Security Consultant bei Media Ventures (ab 1.9. bei SektionEins).
- beschäftigt sich seit ca. 1998 mit Sicherheit im Web (mit Unterbrechungen).
- ist bei Zone-H, freebox Security and Development und dem CCC aktiv.

- hat sehr viel Spass am Gerät :-)

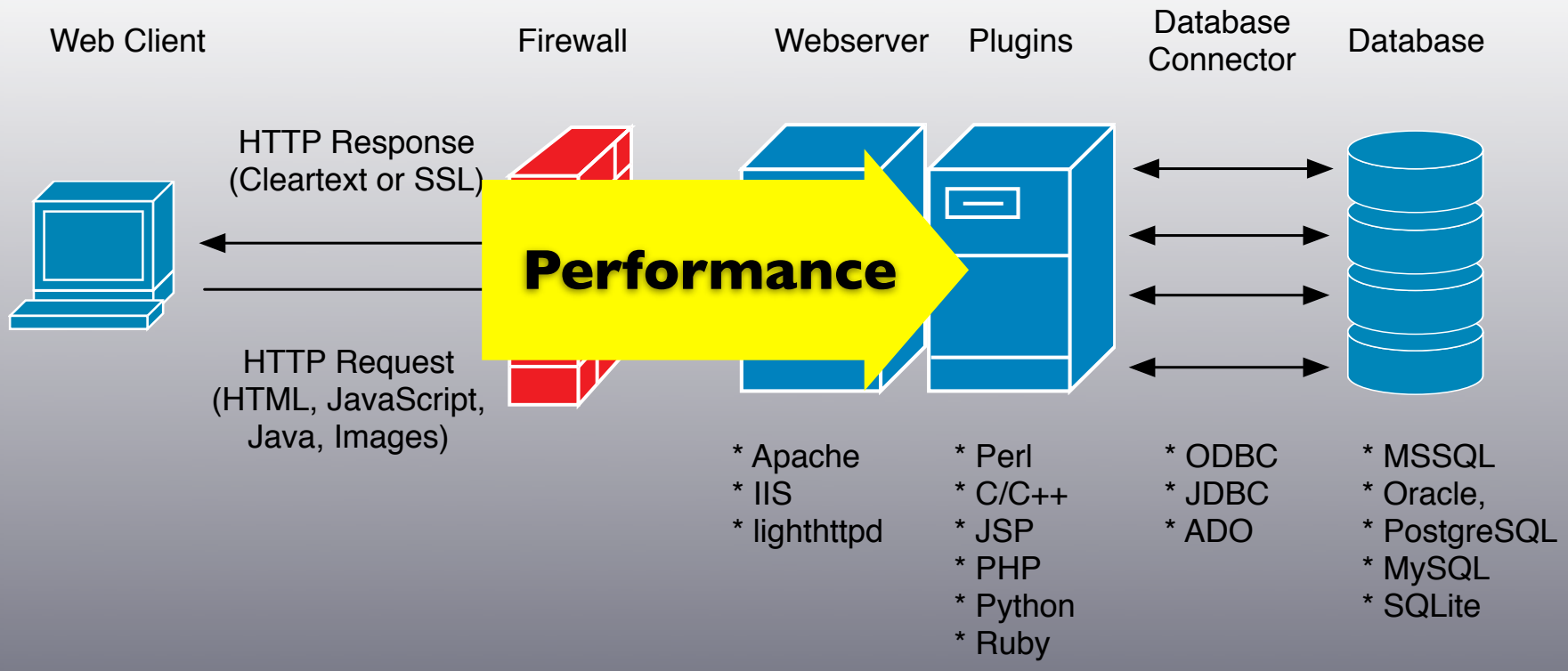
Typische Webanwendung



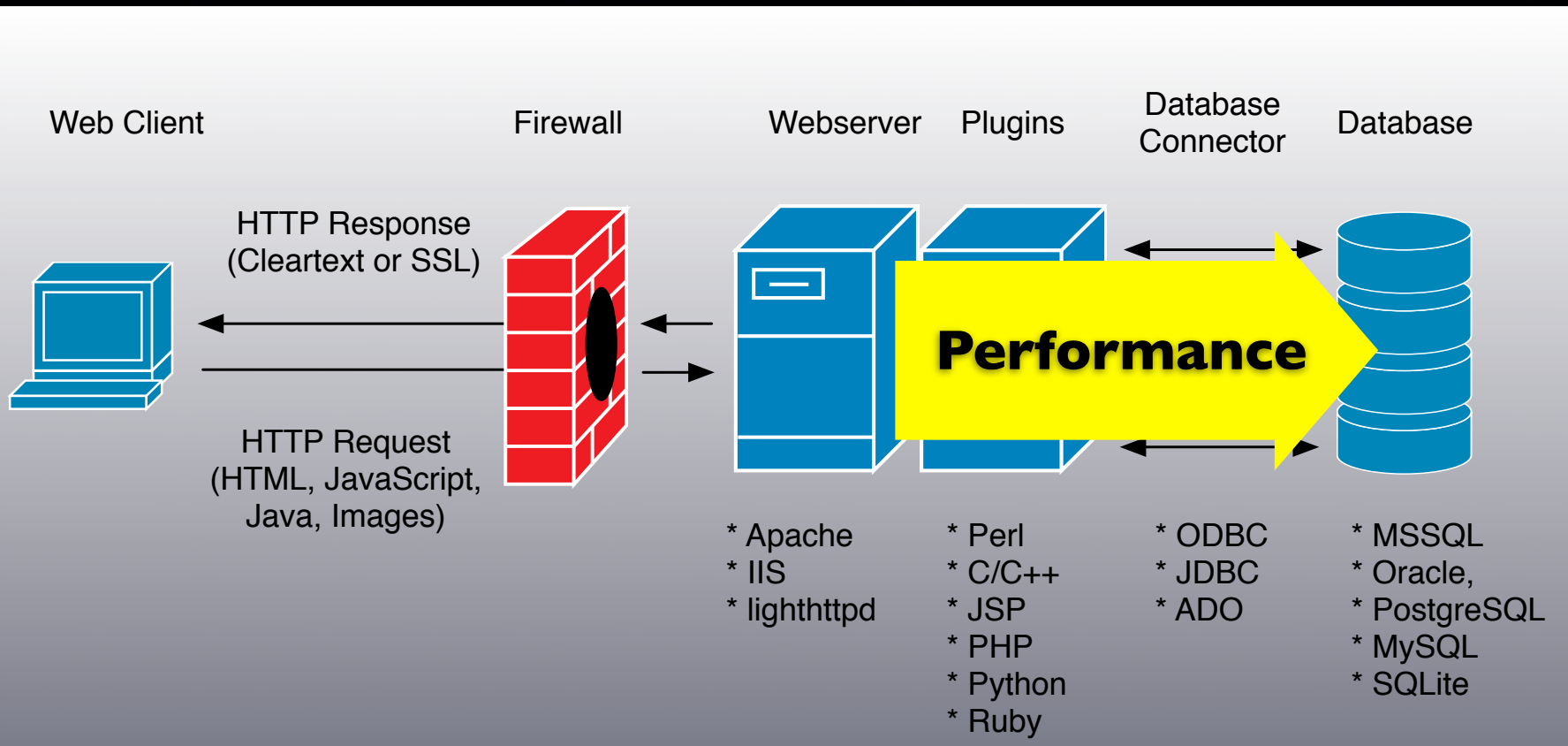
Browser: JavaScript



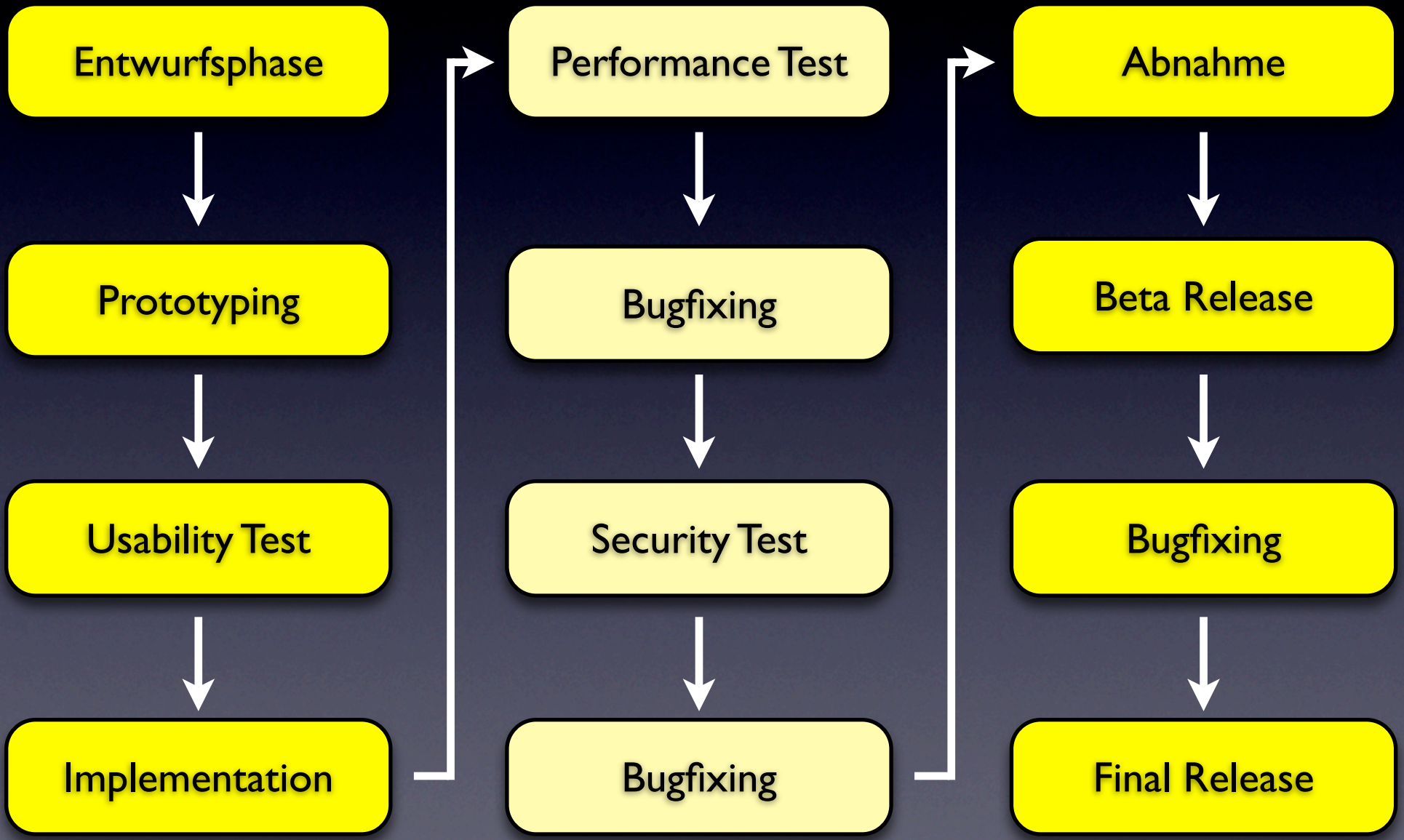
Plugins/Frameworks: Optimierung/Ausführung



Datenbank, File System: Queries, IO, Memory



Development Cycle



Load und Performance Tests

- Was wird gemacht:
  - Simulation vieler gleichzeitiger User in realistischem Szenario (inklusive Sessions, Click Path usw.)
  - Messung von Responds und Resourcen-Verbrauch
- Ziel:
  - finden der Performance-Engpässe einzelner Teile der Applikation und deren Behebung
  - realistische Aussage zum Hardwarebedarf
  - Verhalten in verteilter Installation ermitteln

Load Testing Tools: Einige Beispiele

- **httperf:** Vergleichsweise einfache CLI-Anwendung, die nur HTTP unterstützt und auch nicht sinnvoll verteilt eingestzt werden kann. Reicht aber für einfache Anwendungsfälle.
- **Tsung:** Erlang-basiertes verteiltes Loadtesting Werkzeug mit Unterstützung für HTTP, SOAP, Jabber/XMPP, PostgreSQL.
- **JMeter:** Komplexe Java-GUI und CLI-Anwendung mit Möglichkeit zu sehr genauer Definition der verschiedenen (auch verteilbaren) Workloads. Unterstützt: HTTP(S), FTP, SOAP, LDAP ...)

Load Testing: httperf

```
$ httperf --server priboi --port 80 --uri /test.html  \
     --rate 150 --num-conn 27000 --num-call 1 --timeout 5
```

Total: connections 27000 requests 26701 replies 26701 test-duration 179.996 s
Connection rate: 150.0 conn/s (6.7 ms/conn, <=47 concurrent connections)
Connection time [ms]: min 1.1 avg 5.0 max 315.0 median 2.5 stddev 13.0
Connection time [ms]: connect 0.3

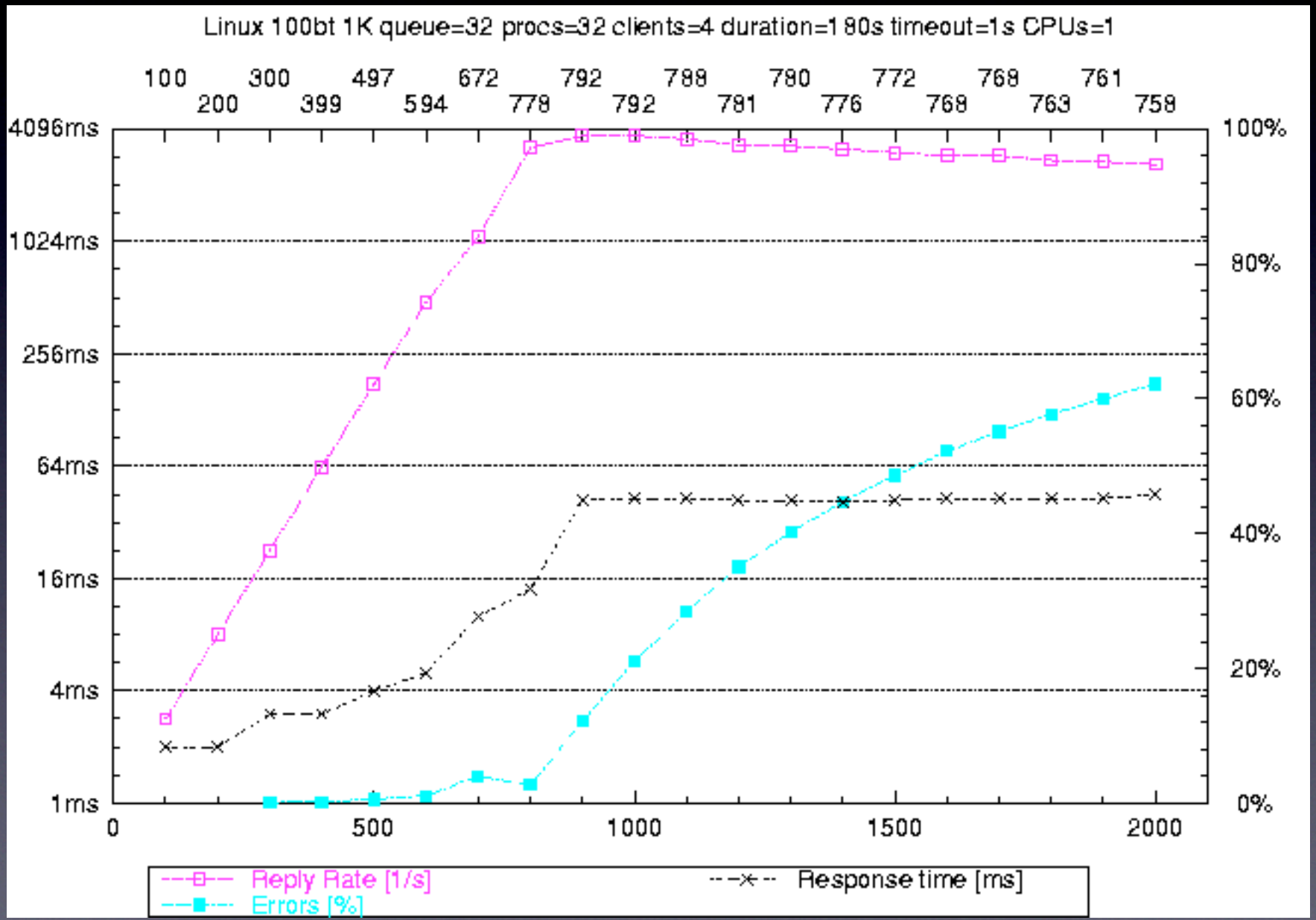
Request rate: 148.3 req/s (6.7 ms/req)
Request size [B]: 72.0

Reply rate [replies/s]: min 139.8 avg 148.3 max 150.3 stddev 2.7 (36 samples)
Reply time [ms]: response 4.6 transfer 0.0
Reply size [B]: header 222.0 content 1024.0 footer 0.0 (total 1246.0)

Reply status: 1xx=0 2xx=26701 3xx=0 4xx=0 5xx=0
CPU time [s]: user 55.31 system 124.41 (user 30.7% system 69.1% total 99.8%)
Net I/O: 190.9 KB/s (1.6*10^6 bps)

Errors: total 299 client-timo 299 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0

Load Testing: httperf



Load Testing: Tsung

Konfigurationsbeispiel:

```
<clients>
  <client host="priboi" weight="1" maxusers="800">
    <ip value="192.168.20.6"></ip>
    <ip value="192.168.20.7"></ip>
  </client>
  <client host="wostok" weight="3" maxusers="600" cpu="2">
    <ip value="192.168.22.4"></ip>
  </client>
</clients>
<servers>
  <server host="192.168.2.6" port="8080" type="tcp"></server>
</servers>
```

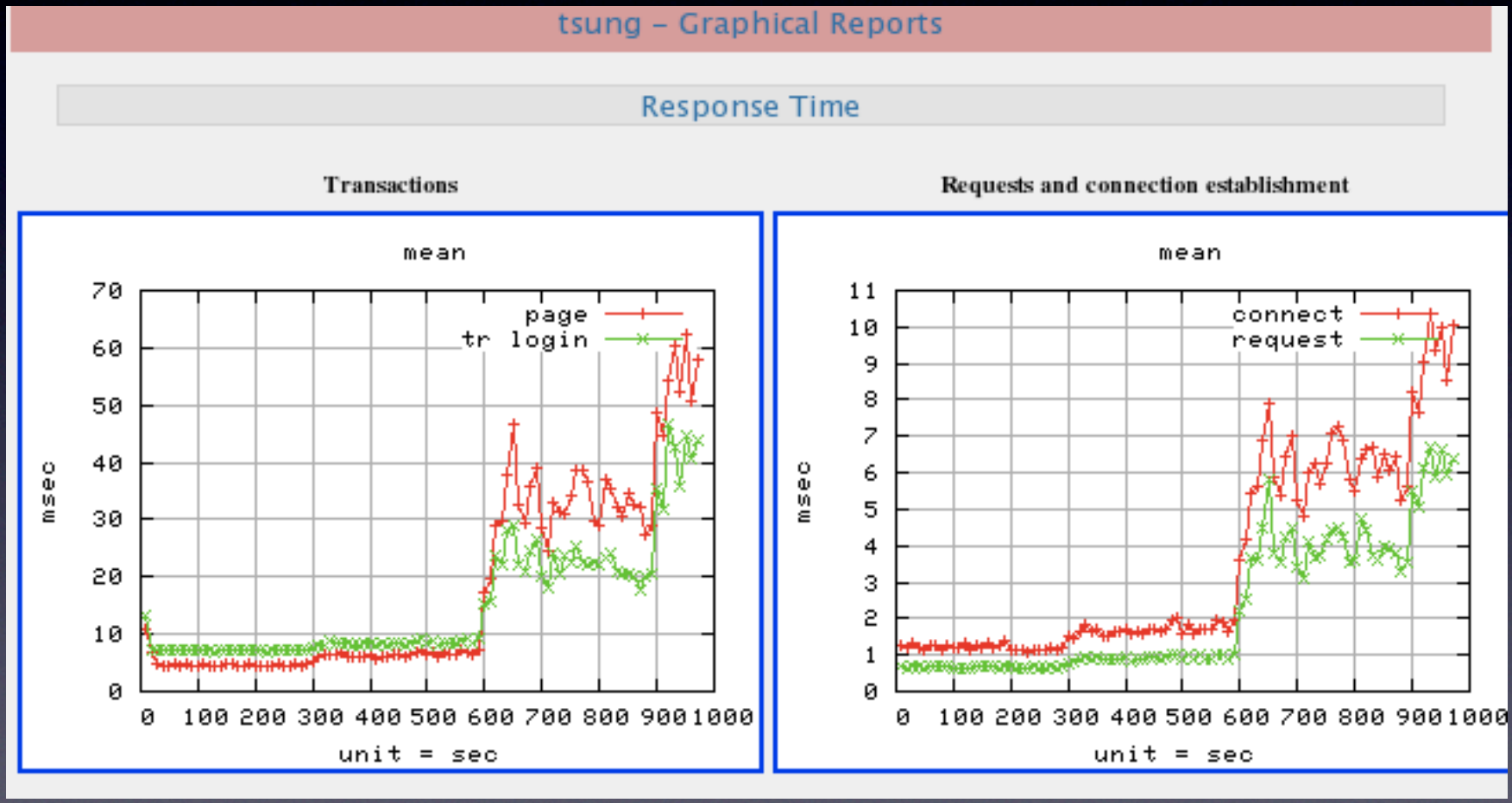
Starten von Tsung:

```
$ tsung start
```

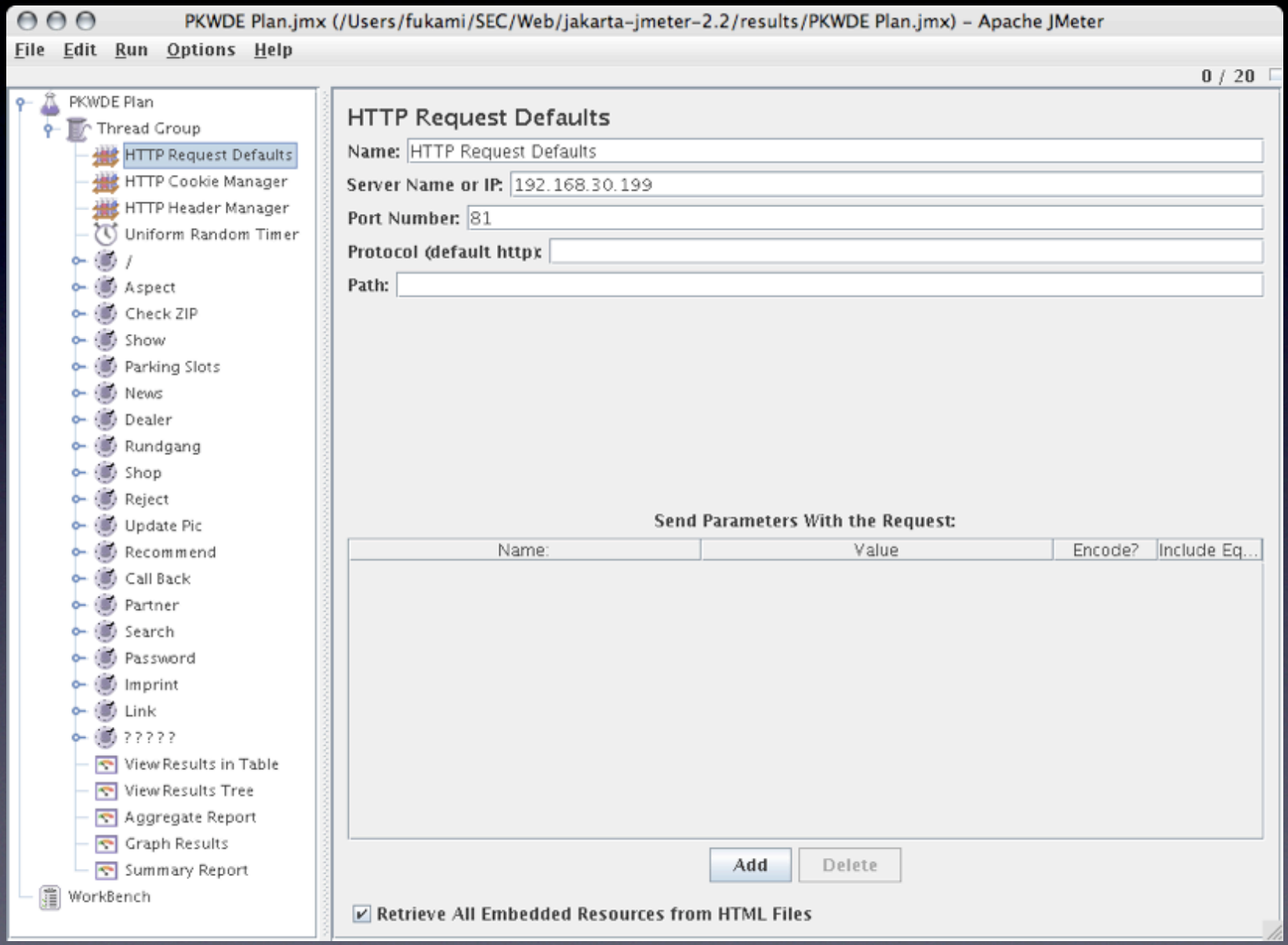
Monitoring der Aktivität:

```
$ tsung status
```

Load Testing: Tsung



Load Testing: JMeter



Load Testing: JMeter

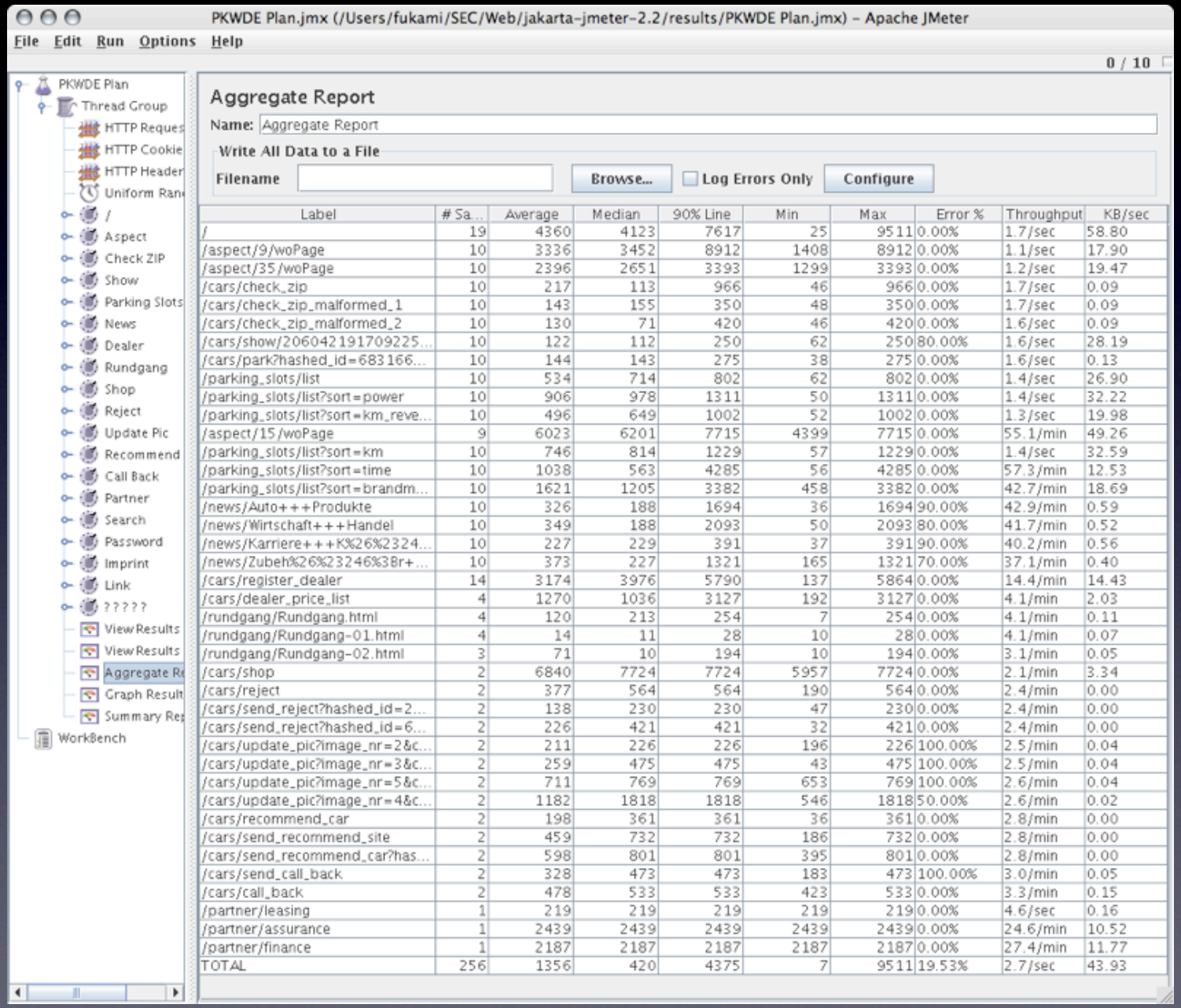
Start von Kommandozeile:

```
$ ./jmeter -t pkw.de.Plan.jmx -n -l pkw-out.log
```

Logoutput:

```
1184940361604,68,/,200,OK,Thread Group 1-4,text,true
1184940361762,13,/,200,OK,Thread Group 1-10,text,true
1184940361816,3868,/,200,OK,Thread Group 1-5,text,true
1184940361861,3842,/,200,OK,Thread Group 1-3,text,true
1184940361677,4035,/,200,OK,Thread Group 1-2,text,true
1184940361734,4263,/,200,OK,Thread Group 1-7,text,true
1184940361850,4165,/,200,OK,Thread Group 1-1,text,true
1184940366022,87,/,200,OK,Thread Group 1-1,text,true
1184940362645,4265,/,200,OK,Thread Group 1-9,text,true
1184940361995,5095,/,200,OK,Thread Group 1-10,text,true
1184940361835,5622,/,200,OK,Thread Group 1-8,text,true
1184940361979,5558,/,200,OK,Thread Group 1-4,text,true
1184940365726,2589,/,200,OK,Thread Group 1-5,text,true
1184940365783,1118,/dealers/dealer_search?
page=2&sort=dealer_company,500,Internal Server Error,Thread
Group 1-10,text,false
```

Load Testing: JMeter



Label	# Sa...	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	19	4360	4123	7617	25	9511	0.00%	1.7/sec	58.80
/aspect/9/woPage	10	3336	3452	8912	1408	8912	0.00%	1.1/sec	17.90
/aspect/35/woPage	10	2396	2651	3393	1299	3393	0.00%	1.2/sec	19.47
/cars/check_zip	10	217	113	966	46	966	0.00%	1.7/sec	0.09
/cars/check_zip_malformed_1	10	143	155	350	48	350	0.00%	1.7/sec	0.09
/cars/check_zip_malformed_2	10	130	71	420	46	420	0.00%	1.6/sec	0.09
/cars/show/206042191709225...	10	122	112	250	62	250	80.00%	1.6/sec	28.19
/cars/park?hashed_id=683166...	10	144	143	275	38	275	0.00%	1.6/sec	0.13
/parking_slots/list	10	534	714	802	62	802	0.00%	1.4/sec	26.90
/parking_slots/list?sort=power	10	906	978	1311	50	1311	0.00%	1.4/sec	32.22
/parking_slots/list?sort=km_reve...	10	496	649	1002	52	1002	0.00%	1.3/sec	19.98
/aspect/15/woPage	9	6023	6201	7715	4399	7715	0.00%	55.1/min	49.26
/parking_slots/list?sort=km	10	746	814	1229	57	1229	0.00%	1.4/sec	32.59
/parking_slots/list?sort=time	10	1038	563	4285	56	4285	0.00%	57.3/min	12.53
/parking_slots/list?sort=brandm...	10	1621	1205	3382	458	3382	0.00%	42.7/min	18.69
/news/Auto+++Produkte	10	326	188	1694	36	1694	90.00%	42.9/min	0.59
/news/Wirtschaft+++Handel	10	349	188	2093	50	2093	80.00%	41.7/min	0.52
/news/Karriere+++K%26%2324...	10	227	229	391	37	391	90.00%	40.2/min	0.56
/news/Zubeh%26%23246%3Br+...	10	373	227	1321	165	1321	70.00%	37.1/min	0.40
/cars/register_dealer	14	3174	3976	5790	137	5864	0.00%	14.4/min	14.43
/cars/dealer_price_list	4	1270	1036	3127	192	3127	0.00%	4.1/min	2.03
/rundgang/Rundgang.html	4	120	213	254	7	254	0.00%	4.1/min	0.11
/rundgang/Rundgang-01.html	4	14	11	28	10	28	0.00%	4.1/min	0.07
/rundgang/Rundgang-02.html	3	71	10	194	10	194	0.00%	3.1/min	0.05
/cars/shop	2	6840	7724	7724	5957	7724	0.00%	2.1/min	3.34
/cars/reject	2	377	564	564	190	564	0.00%	2.4/min	0.00
/cars/send_reject?hashed_id=2...	2	138	230	230	47	230	0.00%	2.4/min	0.00
/cars/send_reject?hashed_id=6...	2	226	421	421	32	421	0.00%	2.4/min	0.00
/cars/update_pic?image_nr=2&c...	2	211	226	226	196	226	100.00%	2.5/min	0.04
/cars/update_pic?image_nr=3&c...	2	259	475	475	43	475	100.00%	2.5/min	0.04
/cars/update_pic?image_nr=5&c...	2	711	769	769	653	769	100.00%	2.6/min	0.04
/cars/update_pic?image_nr=4&c...	2	1182	1818	1818	546	1818	50.00%	2.6/min	0.02
/cars/recommend_car	2	198	361	361	36	361	0.00%	2.8/min	0.00
/cars/send_recommend_site	2	459	732	732	186	732	0.00%	2.8/min	0.00
/cars/send_recommend_car?has...	2	598	801	801	395	801	0.00%	2.8/min	0.00
/cars/send_call_back	2	328	473	473	183	473	100.00%	3.0/min	0.05
/cars/call_back	2	478	533	533	423	533	0.00%	3.3/min	0.15
/partner/leasing	1	219	219	219	219	219	0.00%	4.6/sec	0.16
/partner/assurance	1	2439	2439	2439	2439	2439	0.00%	24.6/min	10.52
/partner/finance	1	2187	2187	2187	2187	2187	0.00%	27.4/min	11.77
TOTAL	256	1356	420	4375	7	9511	19.53%	2.7/sec	43.93

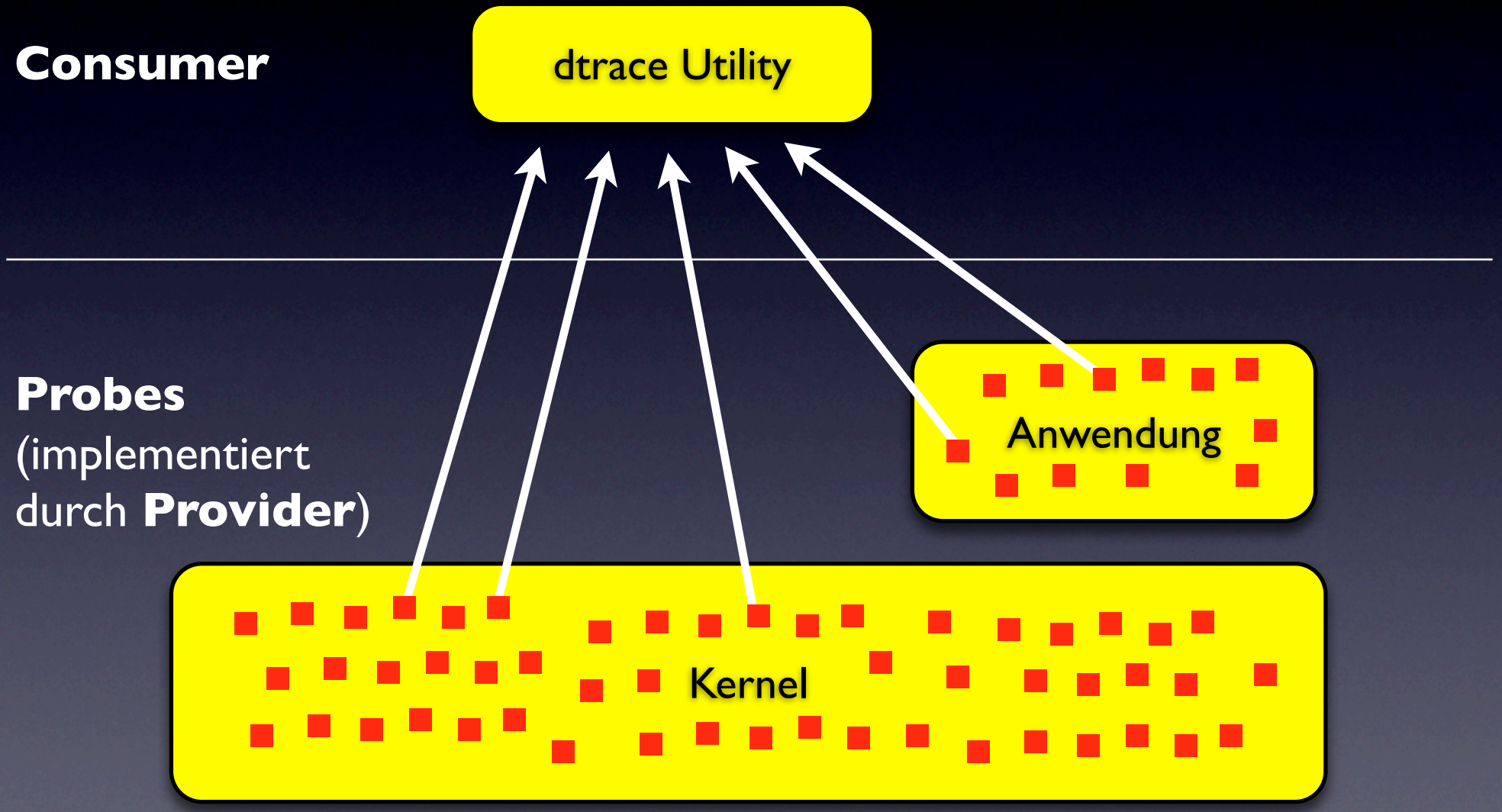
Tracing Tools

- programmiersprachen-nahe Profiler:
  - Rails hat Built-Ins für Profiling
  - PHP: Xdebug (Analyse z.B. mit KCacheGrind)
  - Python: qPyProfiler, Modul profile/pstats
- systemnahe Profiler:
  - DTrace (Solaris, FreeBSD, MacOS X)
  - SystemTap (Linux)

DTrace

- DTrace wurde entwickelt von Sun Microsystems für Solaris 10 (ab 10.5 in MacOS X: XRay, beta für FreeBSD)
- DTrace steht für "Dynamic Tracing"
- hat Tausende von Probes in OS und Anwendungen
- Provider sind die Implementierung von Sonden und in C geschrieben
- Skripte für DTrace-Client werden in der Skriptsprache D geschrieben (C-ähnliche Syntax)

DTrace Architektur



DTrace “Hallo Welt!”

Als Oneliner:

```
# dtrace -n 'BEGIN {trace("Hallo Welt!"); exit(0);}' -n END

dtrace: description 'BEGIN ' matched 1 probe
dtrace: description 'END' matched 1 probe
CPU     ID                    FUNCTION:NAME
  1      1                           :BEGIN   Hallo Welt!
  1      2                             :END
```

Als Script:

```
# cat hallo.d

#!/usr/sbin/dtrace -s
BEGIN
{
  trace("Hallo Welt!");
  exit(0);
}
END {}
```

Die Sprache D

- D ist C-ähnlich
- D unterstützt alle ANSI C Operatoren und erlaubt Zugriff auf native Kernel Typen und globale Variablen
- D hat Support für verschiendene Typen von User-definierten Variablen, inklusive globaler, clause- und thread-local Variablen und assoziativer Arrays
- D-Programme werden in DIF (D Intermediate Format) compiliert und dann im Speicher als Objectfile für das D-Framework zugreifbar gemacht

DTrace Oneliner

Neue Prozesse (mit Argument):

```
dtrace -n 'proc:::exec-success { trace(curpsinfo->pr_psargs); }'
  CPU     ID                    FUNCTION:NAME
    0   5117          exec_common:exec-success   ls
    0   5117          exec_common:exec-success   df -kh
```

Syscall Anzahl pro Prozess ID:

```
dtrace -n 'syscall:::entry { @num[pid,execname] = count(); }'
      683  httpd                8
      409  mysqld              12
     7700  sshd                15
     9816  bash                20
```

Geschriebene Bytes pro Prozessname:

```
dtrace -n 'sysinfo:::writech { @bytes[execname] = sum(arg0); }'
  dtrace                        1
  bash                         90
  df                          986
  sshd                       3307
```

DTrace Provider

```
# dtrace -l|wc -l
   72814

# dtrace -l|grep ruby|wc -l
   55

# dtrace -l|grep ruby
72607  ruby3681  ruby   rb_call0 function-entry
72608  ruby3681  ruby   rb_call0 function-return
72609  ruby3681  ruby   garbage_collect gc-begin
72610  ruby3681  ruby   garbage_collect gc-end
72611  ruby3681  ruby   rb_eval line
72612  ruby3681  ruby   rb_obj_alloc object-create-done
[...]

# dtrace -l|grep mysql
72475  pid409  mysqld __1cLmysql_parse6FpnDTHD_pcI_v_          entry
72482  pid409  mysqld __1cVmysql_execute_command6FpnDTHD__b_ entry
72483  pid409  mysqld __1cVmysql_execute_command6FpnDTHD__b_ return
72495  pid409  mysqld __1cLmysql_parse6FpnDTHD_pcI_v_          return
```

DTrace mit MySQL

Bisher nur rudimentäre MySQL-Probes erhältlich. Jenny Chen von Sun hat aber weitere Probes fertig. Aus einer Mail von ihr:

> *Although we can't release the DTrace patch for mysql currently, I would share you the info on DTrace probes we have integrated with MySQL so far:*
>
> *Query execution: query_execute_start, query_execute_finish*
>
> *Query parse: query_parse_start, query_parse_finish*
>
> *Query cache hit/miss: query_cache_miss, query_cache_hit*
>
> *The above probes have the arguments including "db_name, hostname, username, query", so that we can get the info locally for each db/host/user/query*

DTrace mit MySQL

Der Oneliner (parallel dazu: Login MySQL und Abfrage der existenten Datenbanken):

```
# dtrace -qn 'pid$target::*mysql_parse*:entry \
  { printf("%Y   %s\n", walltimestamp, copyinstr(arg1)); }'\
  -p `pgrep -x mysqld`

2007 Jul 19 14:42:31   select @@version_comment limit 1
2007 Jul 19 14:42:35   show databases
2007 Jul 19 14:42:39   SELECT DATABASE()
2007 Jul 19 14:42:39   show databases
2007 Jul 19 14:42:39   show tables

^C
```

DTrace Ruby Functions

Demo

DTrace Rails und MySQL I

```
#pragma D option quiet

self string uri; self string newuri;

BEGIN
{ start = timestamp; }

syscall::read:entry
/execname == "mongrel_rails" && self->uri == NULL/
{ self->fd = arg0; self->buf = arg1; self->size = arg2; }

syscall::read:return
/self->uri == NULL && self->buf != NULL && (strstr(this->str = copyinstr(self->buf,
    self->size), "GET ") == this->str || strstr(this->str, "POST ") == this->str)/
{
  this->head = strtok(this->str, " ");
  self->newuri = this->head != NULL ? strtok(NULL, " ") : NULL;
}

syscall::read:return
/self->newuri != NULL/
{
  self->uri = self->newuri; self->newuri = NULL;
  printf("################################\n");
  printf("# %3d.%03d  =>  URI  %s\n", (timestamp - start) / 1000000000,
     ((timestamp - start) / 1000000) % 1000,self->uri);
}
```

DTrace Rails und MySQL 2

```
pid$1::*mysql_parse*:entry
{
  printf("  %3d.%03d  ->  SQL  %s\n", (timestamp - start) / 1000000000,
    ((timestamp - start) / 1000000) % 1000,copyinstr(arg1));
}

syscall::read:return
/self->buf != NULL/
{ self->buf = NULL; }

syscall::write:entry
/self->uri != NULL && arg0 == self->fd && strstr(this->str =copyinstr(arg1, arg2),
"HTTP/1.1") == this->str/
{
  printf("# %3d.%03d  <=  URI  %s\n", (timestamp - start) / 1000000000,
    ((timestamp - start) / 1000000) % 1000,self->uri);
  printf("####################################\n");
  self->uri = NULL;
}
```

DTrace Rails und MySQL 3

```
 # ./rsnoop.d `pgrep -x mysqld`

##################################
#  13.450  =>  URI  /
   17.366  ->  SQL  SET NAMES 'UTF8'
   17.618  ->  SQL  SET SQL_AUTO_IS_NULL=0
   26.892  ->  SQL  SELECT * FROM users WHERE (users.`fake_id` = [...]
   27.081  ->  SQL  SELECT min(price_customer),max(price_customer),min(mileage), [...]
   28.988  ->  SQL  SELECT id FROM cars WHERE (cars.deleted_at IS NULL) AND [...]
   37.837  ->  SQL  SELECT id FROM cars WHERE (cars.deleted_at IS NULL) AND [...]
   37.122  ->  SQL  SELECT cars.*, addresses.zip address_zip,addresses.city [...]
   37.882  ->  SQL  SELECT cars.*, addresses.zip address_zip,addresses.city [...]
   38.254  ->  SQL  SELECT * FROM brands
   38.536  ->  SQL  SELECT * FROM models
#  47.116  <=  URI  /
##################################

[...]
```

DTrace Firefox JavaScript Functions I

```
<HTML>
<HEAD>
<TITLE>Clock, JavaScript</TITLE>
<SCRIPT type="text/javascript">
function padZero(i)
{
  if (i < 10) return "0" + i;
  return i;
}
function startTime()
{
  var now = new Date;
  var time = padZero(now.getHours()) + ":" +
  padZero(now.getMinutes()) + ":" +
  padZero(now.getSeconds());
  document.getElementById('clock').innerHTML = "time: " + time + "<br>";
  var timeout = setTimeout('startTime()', 1000);
}
</SCRIPT>
</HEAD>
<BODY onload="startTime()">
<DIV id="clock"></DIV>
</BODY>
</HTML>
```

DTrace Firefox JavaScript Functions 2

```
# cat js_funccalls.d

#!/usr/sbin/dtrace -Zs
#pragma ident   "@(#)js_funccalls.d     1.3     07/03/26 SMI"
#pragma D option quiet
dtrace:::BEGIN
{
  printf("Tracing... Hit Ctrl-C to end.\n");
}
*mozilla*:::js_function-entry
{
  @funcs[basename(copyinstr(arg0)), copyinstr(arg2)] = count();
}
dtrace:::END
{
  printf(" %-32s %-36s %8s\n", "FILE", "FUNC", "CALLS");
  printa(" %-32s %-36s %@8d\n", @funcs);
}
```

DTrace Firefox JavaScript Functions 3

```
# ./js_funccalls.d
Tracing... Hit Ctrl-C to end.
^C

 FILE                                 FUNC                                 CALLS
 nsMicrosummaryService.js             GetSources                               1
 nsMicrosummaryService.js             MSS__getBookmarks                        1
 nsMicrosummaryService.js             MSS__updateMicrosummaries                1
 nsMicrosummaryService.js             getPref                                  1
 nsMicrosummaryService.js             getPrefType                              1
 nsMicrosummaryService.js             getService                               1
 nsMicrosummaryService.js             hasMoreElements                          1
 nsMicrosummaryService.js             max                                      1
 nsMicrosummaryService.js             now                                      1
 nsMicrosummaryService.js             GetResource                              2
 nsMicrosummaryService.js             MSS__resource                            2
 clock.html                           getElementById                           3
 clock.html                           getHours                                 3
 clock.html                           getMinutes                               3
 clock.html                           getSeconds                               3
 clock.html                           setTimeout                               3
 clock.html                           startTime                                3
 clock.html                           padZero                                  9
```

Referenzen

- JMeter
  http://jakarta.apache.org/jmeter/index.html
- Tsung
  http://tsung.erlang-projects.org/
- httperf
  http://www.hpl.hp.com/research/linux/httperf/