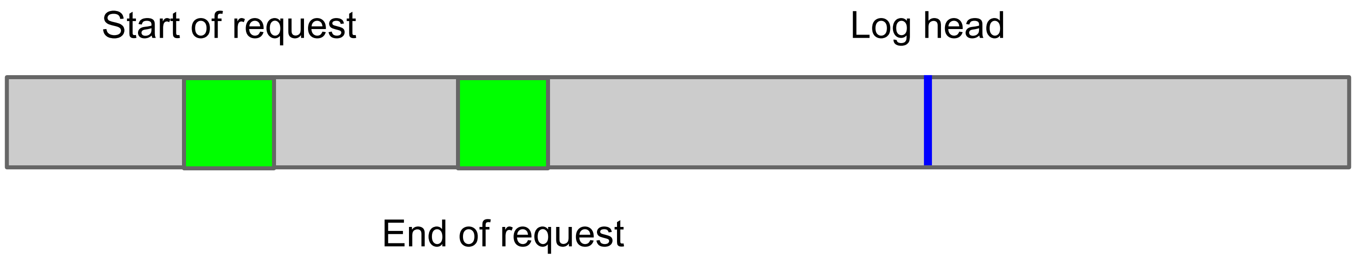
Logging in Varnish 4

Martin Blix Grydeland
Varnish Software

Logging in Varnish

- Shared memory
- Always log everything (no debug switches)
- Utilities to inspect the log

Shared memory log layout



varnishlog in 3.0

- Groups log records into transactions
- -i and -x controls output
- -m <tag:regex> allows querying

Problems with varnishlog in 3.0

- Only regex matching
- Only AND operator
- -i/-x and -m don’t work together
- No req/bereq relationship
- Always copying
- Varnish 4 doesn’t log by fd

Logging goals for Varnish 4

- Transactions and transaction groups
- Query language
- Output controls
- Zero copy in the common case

Varnish transactions

- A transaction is one work item in Varnish
- Share a single VXID
- Types of transactions
    - Client request
    - Backend request
    - ESI sub-request
    - Session

Transaction groups …

- … group together transactions
- Grouping modes
  - VXID
  - Request
  - Session
  - Raw
- Names subject to change?

Transaction groups are hierarchical

Cache miss with request grouping

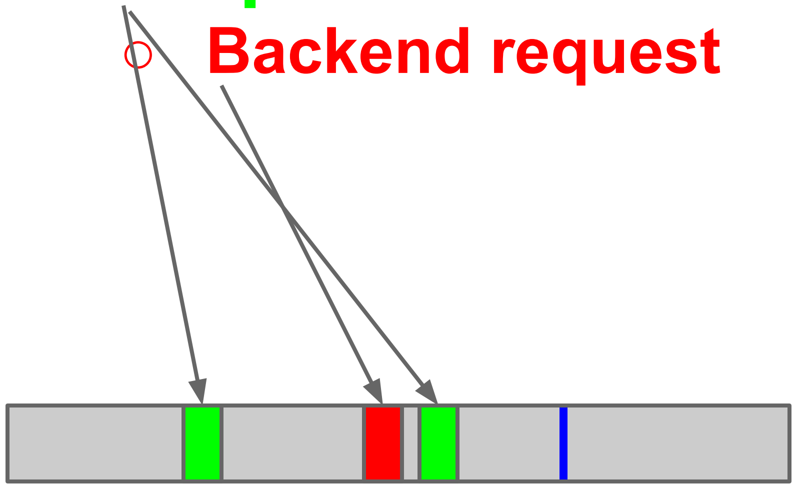
- Request
    - Backend request

ESI included miss with session grouping

- Session
    - Request
        - ESI sub-request
            - Backend request
        - ESI sub-request
        - Backend request
    - Request
        - Backend request

Cache miss with request grouping

- Request
  - Backend request



```
martin@friday:~$ varnishlog -g request
*    << Request  >> 32770
-    Begin          req 32769
-    ReqMethod      GET
-    ReqURL         /
-    ReqProtocol    HTTP/1.1
-    ReqHeader      TE: deflate,gzip;q=0.3 ...
-    Link           bereq 32771
-    VCL_call       DELIVER
-    VCL_return     deliver
-    RespProtocol   HTTP/1.1
-    RespStatus     200
-    RespResponse   OK
-    ReqEnd         1385330985.979025126
1385330985.978960991 -0.001315594 0.001251459
-0.001315594
-    End
**   << BeReq    >> 32771
--   Begin          bereq 32770
--   VCL_call       BACKEND_FETCH
--   VCL_return     fetch
--   BackendOpen    18 default(127.0.0.1,::1,8020)
127.0.0.1 45989
--   Backend        18 default default(127.0.0.1,::1,8020)
--   BereqMethod    GET
--   BereqURL       /
--   BereqEnd       1385330985.979187250
1385330985.980367422 0.000082792 0.000496101 0.000326045
0.000822146
--   End
```

varnishlog invocation

```
varnishlog -g <grouping-mode> -q <query-expression>
```

Grouping mode is one of:

- vxid (default)
- request
- session
- raw

Query language

- Operates on transaction groups
- Expr true if it matches one or more records
- Expr false if it matches no records

```
$ varnishlog -q 'respstatus < 500'
```

Matching on transaction groups

```
$ varnishlog -g request \
    -q 'ReqURL eq "/"'
```

```
$ varnishlog -g request \
    -q 'Backend ~ default'
```

```
*   << Request  >> 32770
-   Begin          req 32769
-   ReqMethod      GET
-   ReqURL         /
-   ReqProtocol    HTTP/1.1
-   ReqHeader      TE: deflate,gzip;q=0.3 ...
-   Link           bereq 32771
-   VCL_call       DELIVER
-   VCL_return     deliver
-   RespProtocol   HTTP/1.1
-   RespStatus     200
-   RespResponse   OK
-   ReqEnd         1385330985.979025126
1385330985.978960991 -0.001315594 0.001251459
-0.001315594
-   End
**  << BeReq    >> 32771
--  Begin          bereq 32770
--  VCL_call       BACKEND_FETCH
--  VCL_return     fetch
--  BackendOpen    18 default(127.0.0.1,::1,8020)
127.0.0.1 45989
--  Backend        18 default default(127.0.0.1,::1,8020)
--  BereqMethod    GET
--  BereqURL       /
--  BereqEnd       1385330985.979187250
1385330985.980367422 0.000082792 0.000496101 0.000326045
0.000822146
--  End
```

Usual suspects of operators

- String matching
  ```
  RespProtocol eq "HTTP/1.1"
  ```
- Regular expressions
  ```
  ReqMethod ~ "GET|POST"
  ```
- Integer matching
  ```
  RespStatus == 200
  RespStatus >= 200
  ```
- Float matching
  RespStatus == 200.

Boolean operators

- and

```
RespStatus >= 500 and RespStatus < 600
```

- or

```
ReqURL == "/" or ReqURL == "/index.html"
```

- Group terms using parenthesis
- Negate using `not`

Other useful tricks

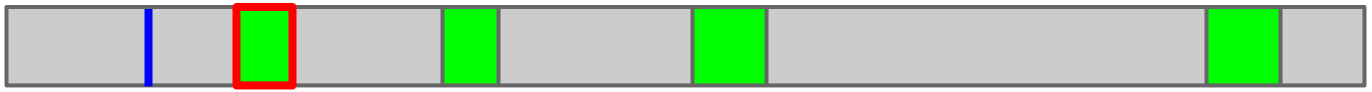
- Response time exceeds ½ second
  ```
  ReqEnd[5] >= 0.5
  ```
- Client requests connection closed
  ```
  ReqHeader:connection ~ close
  ```
- ESI miss (-g request)
  ```
  {3+}Begin ~ Bereq
  ```

Output controls

- Applied last, doesn’t affect queries
- -i <taglist> / -I <taglist:regex>
- -x <taglist> / -X <taglist:regex>
- Taglists supports globbing (e.g. Req*)

Zero copy

- Everything done on shared memory
- Except when needed
- Transparently make copies
- Copy only what’s needed
- Max time / number knobs



varnishncsa in 4

- Query language
- Zero copy
- Format string parsed once
- Built for speed
- Feature parity with 3.0
    - Backend traffic logging planned

Status of the other utilities

- varnishtop
- varnishhist
- varnishreplay

Not yet reimplemented using the new API

API

- All of the magic happens in the API
- Better varnishd exceptional handling
- Easy integration

API cont.

```
/* See varnish-cache/include/vapi/vsl.h for full documentation */


/* Create new query */

struct VSLQ *VSLQ_New(struct VSL_data *vsl, struct VSL_cursor **cp,
    enum VSL_grouping_e grouping, const char *query);


/* API processing */

int VSLQ_Dispatch(struct VSLQ *vslq, VSLQ_dispatch_f *func, void *priv);


/* Callback function definition */
typedef int VSLQ_dispatch_f(struct VSL_data *vsl,
    struct VSL_transaction * const trans[], void *priv);
```

API cont.

```
struct VSL_transaction {
    unsigned                    level;
    int32_t                     vxid;
    int32_t                     vxid_parent;
    enum VSL_transaction_e      type;
    struct VSL_cursor               *c;
};
struct VSL_cursor {
    struct VSLC_ptr             rec;
    const void                  *priv_tbl;
    void                        *priv_data;
};

/* Traverse log entries */
int VSL_Next(struct VSL_cursor *c);
```

Thanks for listening