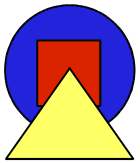
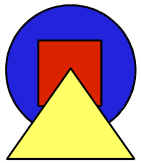


Perl API to AFS Monitoring and Debugging Tools

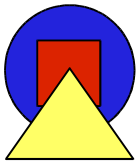
Elizabeth Cassell and
Alf Wachsmann

Stanford Linear Accelerator Center



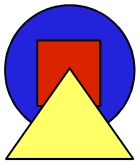
AFS::Debug

- Perl functions for AFS monitoring and debugging tools
- `afsmonitor`
  - Gathers statistics about File Servers and Cache Managers
- `cmdebug`
  - Reports status of cache manager and cache entries on a particular AFS client machine
- `rxdebug`
  - Provides debugging trace of Rx activity
- `scout`
  - Gathers statistics from the File Server processes running on the specified machines
- `udebug`
  - Reports status of Ubik process associated with a database server process
- `xstat_cm_test`
  - Gathers data collections from the Cache Manager
- `xstat_fs_test`
  - Gathers data collections from the File Server process



Implementation…

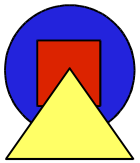
- Like AFS Perl module
- Written in XS
- Code copied (duplicated) from OpenAFS
- Comments to keep track of where each function came from
- Gets rid of ASCII art (see, i.e. scout)
- Got rid of threading: each Perl function returns a snapshot of the current state



...Implementation

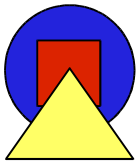
- Input parameters are a hash
- Output of programs is put into Perl hash and returned

```
$result = rxdebug(nodally          => 1,
                  allconnections => 1,
                  peers            => 1,
                  servers          => $servers,
                  port             => $port
                 );
```



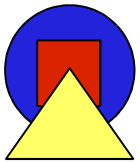
OpenAFS source code quality

- The OpenAFS source code of user space tools is very convoluted and unnecessarily complicated
- Found and fixed several bugs.
  Will be in next OpenAFS release.



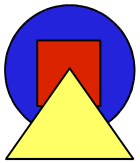
Documentation

- One overall POD file listing all functions and referring to individual POD files
- Each function comes with a separate POD file explaining its input and output parameters
- Everything is derived from IBM's HTML documentation for the functions



Example Code

- For each function, we have packaged one example Perl script that calls the function in many different ways
- We have used these scripts for our code, regression, and compatibility testing



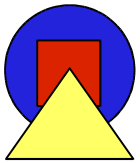
Example 1...

```
% scout -server afs01
```

Scout

Conn	Fetch	Store	Ws		Disk attn: > 95% used
16311	241757090	9067221	3139	afs01	a:17109426
					b:15578438
					c:798753
					d:15824108
					e:39923441
					f:22286173
					g:62531140
					h:30760586

Probe 1 results



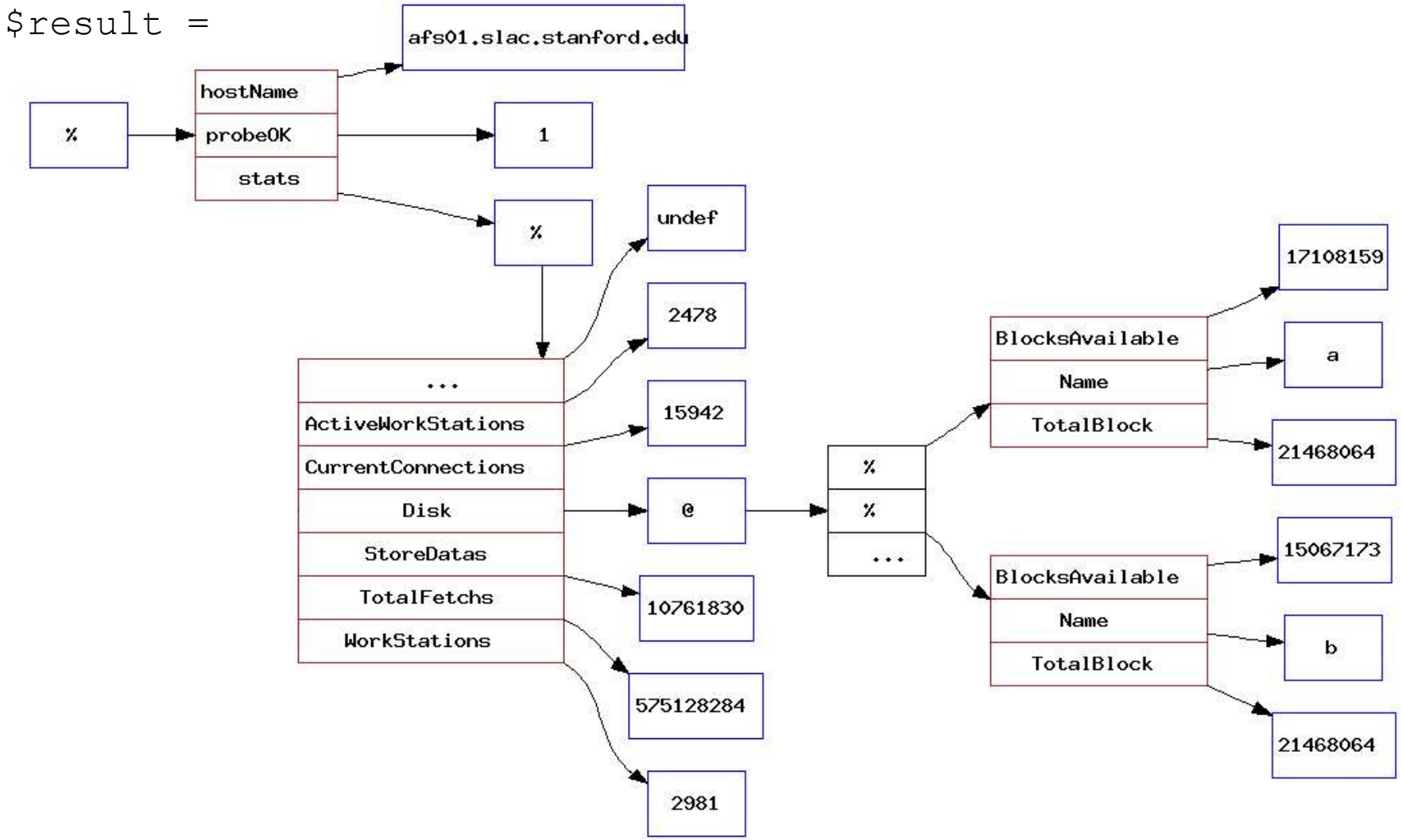
...Example 1

```
use AFS::Debug qw(scout);
my $result = scout(server => “afs01”);
```

Contents of `$result`:

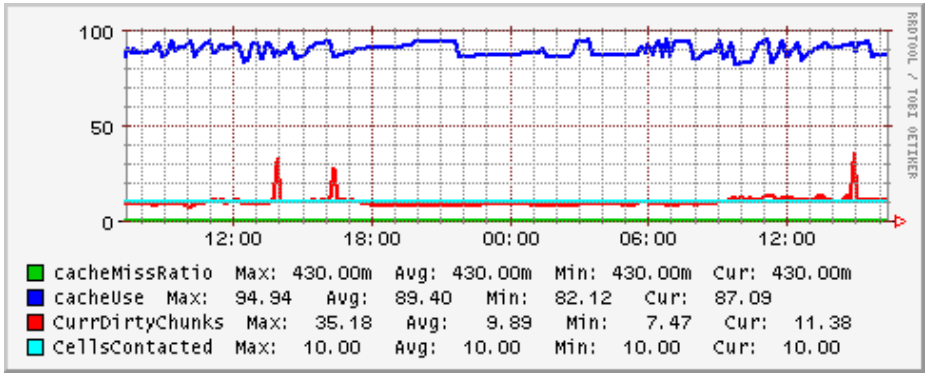
```
$result = [
   {
      hostName => 'afs01.slac.stanford.edu',
      probeOK  => 1,
      stats    => {
                     Disk => [
                        {
                           BlocksAvailable => 17109426,
                           Name => 'a',
                           TotalBlock => 21468064
                        },
                        ...
                     ],
                     CurrentConnections => 16311,
                     StoreDatas => 5864562,
                     TotalFetchs => 241757090,
                     WorkStations => 3139
                     ...
                  }
      ...
   }
]
```

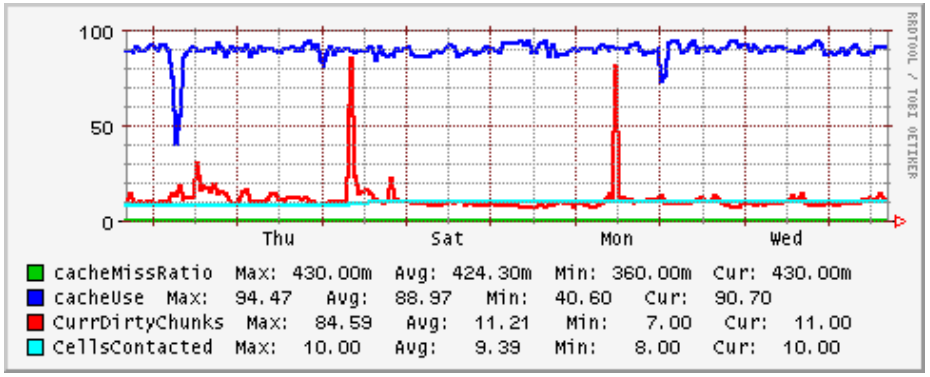
...Example 1

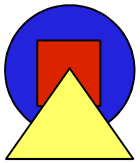

$result =
%
hostName
probeOK
stats
afs01.slac.stanford.edu
1
%
...
ActiveWorkStations
CurrentConnections
Disk
StoreDatas
TotalFetchs
WorkStations
undef
2478
15942
@
10761830
575128284
2981
%
%
...
BlocksAvailable
Name
TotalBlock
17108159
a
21468064
BlocksAvailable
Name
TotalBlock
15067173
b
21468064

Applications...

- Monitoring (Nagios)
- `check_AFScm.pl`:
  - AFS Cache Manager (aka AFS Client) tester
  - uses `afsmonitor()`



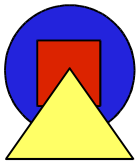




...Applications...

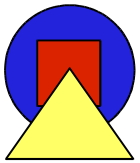
- **Core of** `check_AFScm.pl`:

```
($fsinfo, $cminfo) = afsmonitor(cmhosts => [$opt_H],
                                cmshow  => ["PerfStats_section"]);

if ($cminfo && $cminfo->[0]->{probeOK}) {

  $BlocksTotal = $cminfo->[0]->{PerfStats_section}-> \
                 {PerfStats_group}->{cacheBlocksTotal}->{value};

  $BlocksInUse = $cminfo->[0]->{PerfStats_section}-> \
                 {PerfStats_group}->{cacheBlocksInUse}->{value};

  $percentage = sprintf("%.2f", $BlocksInUse/$BlocksTotal*100);


  $CurrDirtyChunks = $cminfo->[0]->{PerfStats_section}-> \
                 {PerfStats_group}->{cacheCurrDirtyChunks}->{value};

  print "AFS $status - ";
  print "cacheUse=$percentage,CurrDirtyChunks=$CurrDirtyChunks\n";
}
```



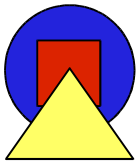
...Applications

- `check_ubik.pl`:
  - AFS database server ubik tester:
    - Is sync site still sync site?
    - Are non-sync sites still non-sync sites?
    - Is recovery state still "0x1f"?
  - uses `udebug()`
  - see next slide
- `Meltdown.pl`:
  - analyze problems on AFS servers
  - uses `rxdebug()`
  - derived `check_AFSfs.pl` to monitor number of idle threads on AFS file servers



Example 2: check_ubik.pl

```
#!/usr/bin/perl
use AFS::Debug qw(udebug);

$result = udebug(server => $opt_H, port => 7003);
$status = 'CRITICAL' if ($AFS::CODE);

my $sync_site_name = 'afsdb1'; # IP: 134.79.18.25
my $afsdb_server_pattern = 'afsdb';

if ($result) {
  if ($opt_H eq $sync_site_name) {
    $status = 'WARNING' unless $result->{amSyncSite};
    $status = 'WARNING' unless ($result->{recoveryState}
                                == 0x1f);
  } elsif ($opt_H =~ /$afsdb_server_pattern/) {
    $status = 'WARNING' if $result->{amSyncSite};
    $status = 'WARNING' unless ($result->{lastYesHost}
                                eq '134.79.18.25')
  }
}

print "AFS $status\n";
exit $ERRORS{$status};
```

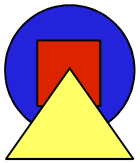


Compatibility

- Tested on the following operating systems:

OS	@sys	Compiler
Solaris	sun4x_58	gcc-2.95.3
	sun4x_59	gcc-2.95.3
Red Hat 7.2	i386_linux24	gcc-2.95.3
Red Hat Enterprise Linux 3	i386_linux24	gcc-3.2.3
SuSE Linux 8.2	i386_linux24	gcc-3.3.3
Mac OS X	ppc_darwin_70	gcc-3.3

- Perl 5.6.1 or newer
- IBM/Transarc AFS v3.6 or OpenAFS 1.2.x



Availability

- Have heard from too few people who have tried it
- Source code can be downloaded via http://www.slac.stanford.edu/~alfw/AFS-Debug/
- It will go on CPAN once more people/sites have tested it