

Taking advantage of ARS log files



Carlos Ungil

CERN

RUG Suisse Romande
(Allaman, 9 June 2005)

Agenda

- Remedy at CERN
- Log Files
 - Keep log files
 - they are (almost) for free
 - Look at them in times of trouble
 - the answer might be a few greps away
 - Analyse them regularly
 - catch problems while they are small
- Poor Man's Change Tracking

Remedy at CERN

- Since 1999
 - 4.0, 4.5, 5.0, 5.1 (6.3?)
- Solaris 8
 - Midtier in the same server (now 5.1 and 6.3 in parallel)
- Oracle 9i
 - central database (Sun Cluster)
- Production server, backup server, test+dev servers
 - Remedy Migrator

Remedy at CERN

- 100-120 users/day (200-250 users/month)
- Remedy User Tool available in Terminal Service
 - many Linux users, midtier 6.3 is a big improvement
- 2 main applications
 - Computing Helpdesk
 - Computer Centre Management
- ~ 3000 tickets/week

Reasons to use logs

- Why doesn't the system work?
 - You can enable logs when you want to debug a problem
 - better if always on: you can understand “what happened”
- What can be done to improve performance?
 - Analysing logs you can...
 - tell if the performance is degrading
 - find slow queries (and then add indexes or educate the users)
 - identify obsolete workflow that is never triggered

ARS log files (client)

- Client side workflow (Active Links, Macros)
- It's possible to log in the client the corresponding server workflow (API, SQL, Filters)
 - I've never used this, but I'd love to have the inverse:
collect in the server the logs of client-side workflow

- *arerror.log* is worth looking at from time to time
- other log files can be enabled and disabled at will
 - Alert, Distributed Server, Server Group
 - we don't use alerts, and the others are not even available
 - Arforkd, Plug-in Server, Thread, User
 - Thread and User logs can help to optimise queues and licenses
 - API, Escalation, Filter, SQL
 - very interesting, lots of info... but they can be **huge**

Problem #1

- Performance
 - No noticeable degradation
 - anyway, the first bug resolved thanks to the logs would pay for it
 - (well, maybe when we tried Server Events)
 - (I think so, I don't remember the details)
- There are two options to use in case of problems
 - per-thread logs
 - buffered output

Problem #2

- Space
 - ~2 GB of log files every day!
 - We archive log files every night (50MB compressed)
 - API program used to stop/restart logging
 - Big logs (Filter,API,SQL) are restarted again at 7:00
- If this is a real problem
 - set a maximum size for the logs

□ ARError

- Tue May 31 04:07:31 2005 390603 : : This entry has been marked to deletion, and cannot be modified. (ARERR 555002102)
- Tue May 31 06:15:04 2005 390603 : Failure during SQL operation to the database : ORA-00942: table or view does not exist (ARERR 552)
- Tue May 31 19:49:10 2005 0 : AR System server terminated when a signal/exception was received by the server (ARNOTE 20)
- Tue May 31 19:49:10 2005 15

□ Arforkd

- Wed Jun 1 08:13:36 2005 Received record marker.
- Wed (...) 2005 Spawning child, shell "", command "echo 6/1/2005 8:13:37 AM "--" gbevilla >> /arslogs/login.log".
- Wed (...) 2005 Spawned "echo 6/1/2005 8:13:37 AM "--" gbevilla >> /arslogs/login.log" as pid 14125.
- Wed Jun 1 08:13:36 2005 Awaiting record marker.
- Wed Jun 1 08:13:36 2005 process id 14125 has exited with status 0x0.

□ Thread

- <THRD> /* Fri May 13 2005 10:07:37.0924 */ Thread Trace Log -- ON
- <THRD> /* Fri May 13 2005 10:07:37.2284 */ Limits found: current rlimit=1024 - max rlimit=1024
- <THRD> /* Fri May 13 2005 10:07:37.2288 */ Thread Id 4 (thread number 0) Thread Manager started.
- <THRD> /* Fri May 13 2005 10:08:07.0911 */ Thread Id 36 (thread number 29) on LIST queue started.

□ Plug-in Server

- <PLGN> <TID: 000016> <RPC ID: 0000001167> <Queue: ARFILTERAPI> <Client-RPC: 390695>
> /* Wed Jun 01 2005 21:26:53.4620 */ +CALL ARFilterApiCall -- filter API CERN.ARF.SENDMAIL
<...> /* Wed Jun 01 2005 21:26:54.1283 */ -CALL OK

□ User

- <USER> <TID: 000015> <RPC ID: 0003108934> <Queue: Fast > <Client-RPC: 390620 > <USER: mailer
> /* Mon Jun 06 2005 08:59:06.2871 */ FIXED GRANT WRITE mailer
<USER> <TID: 000010> <RPC ID: 0003109215> <Queue: Fast > <Client-RPC: 390620 > <USER: mailer
> /* Mon Jun 06 2005 08:59:12.7305 */ FIXED RELEASE mailer
<USER> <TID: 000017> <RPC ID: 0003109312> <Queue: Fast > <Client-RPC: 390620 > <USER: cats
> /* Mon Jun 06 2005 09:00:00.1312 */ READ GRANT WRITE cats
<USER> <TID: 000012> <RPC ID: 0003109813> <Queue: Fast > <Client-RPC: 390620 > <USER: akhodaba
> /* Mon Jun 06 2005 09:00:05.7808 */ READ BAD PASSWORD akhodaba
<USER> <TID: 000019> <RPC ID: 0003109946> <Queue: List > <Client-RPC: 390620 > <USER: akhodaba
> /* Mon Jun 06 2005 09:00:13.3965 */ FLOAT GRANT WRITE akhodaba (17 of 55 write)
<USER> <TID: 000019> <RPC ID: 0003109946> <Queue: List > <Client-RPC: 390620 > <USER: akhodaba
> /* Mon Jun 06 2005 09:00:13.3968 */ FLOAT GRANT FULL akhodaba (3 of 10 full)

Other log files

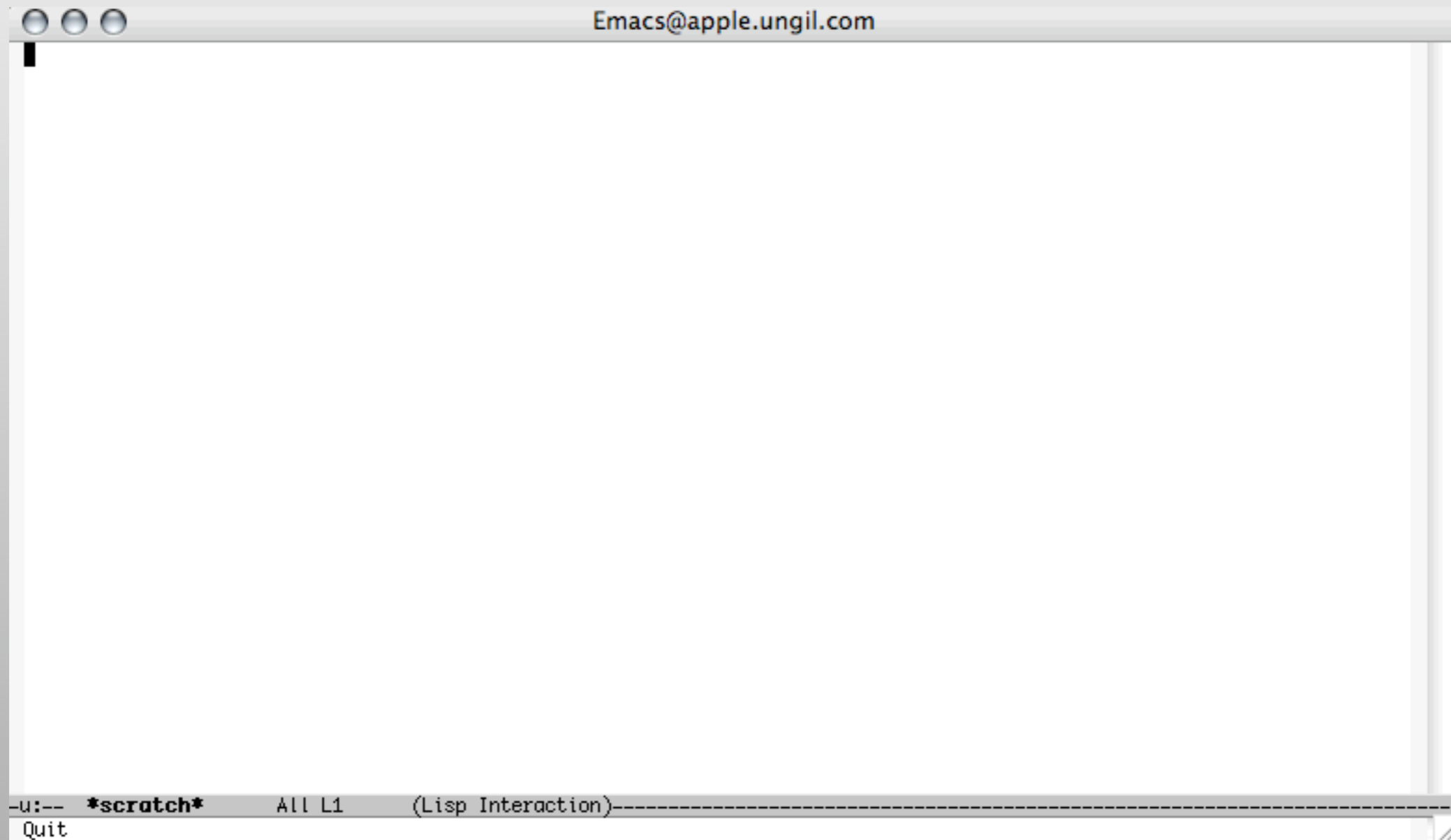
typical daily size	(lines)	(MB)
Filter	6 million	1000
SQL	2.5 million	700
API	2 million	300
Escalation	1 million	150

- These log files are huge and not very readable
- but they contain very useful information
 - after some preprocessing they are easier to inspect, and more adequate to generate graphs or reports from them
 - discarding non-useful bits, grouping related info

Online analysis

- *emacs* library to have easy access to log files
 - see the current log (*cat*) or monitor it how grows (*tail*)
 - look for a particular text, user or rpcid (*grep*)
 - *emacs* connects to the server using ssh, and tailing and grepping is done remotely (only the interesting stuff is transferred)
 - use cases
 - identify the SQL statements executed when a user performs an action, while he describes on the phone what he is doing
 - find when a particular ticket was modified, and then look at the relevant fragment of the filter log for each of these transactions

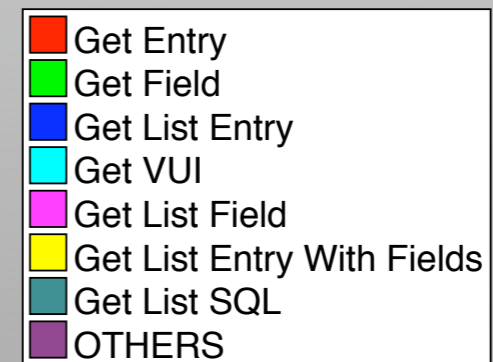
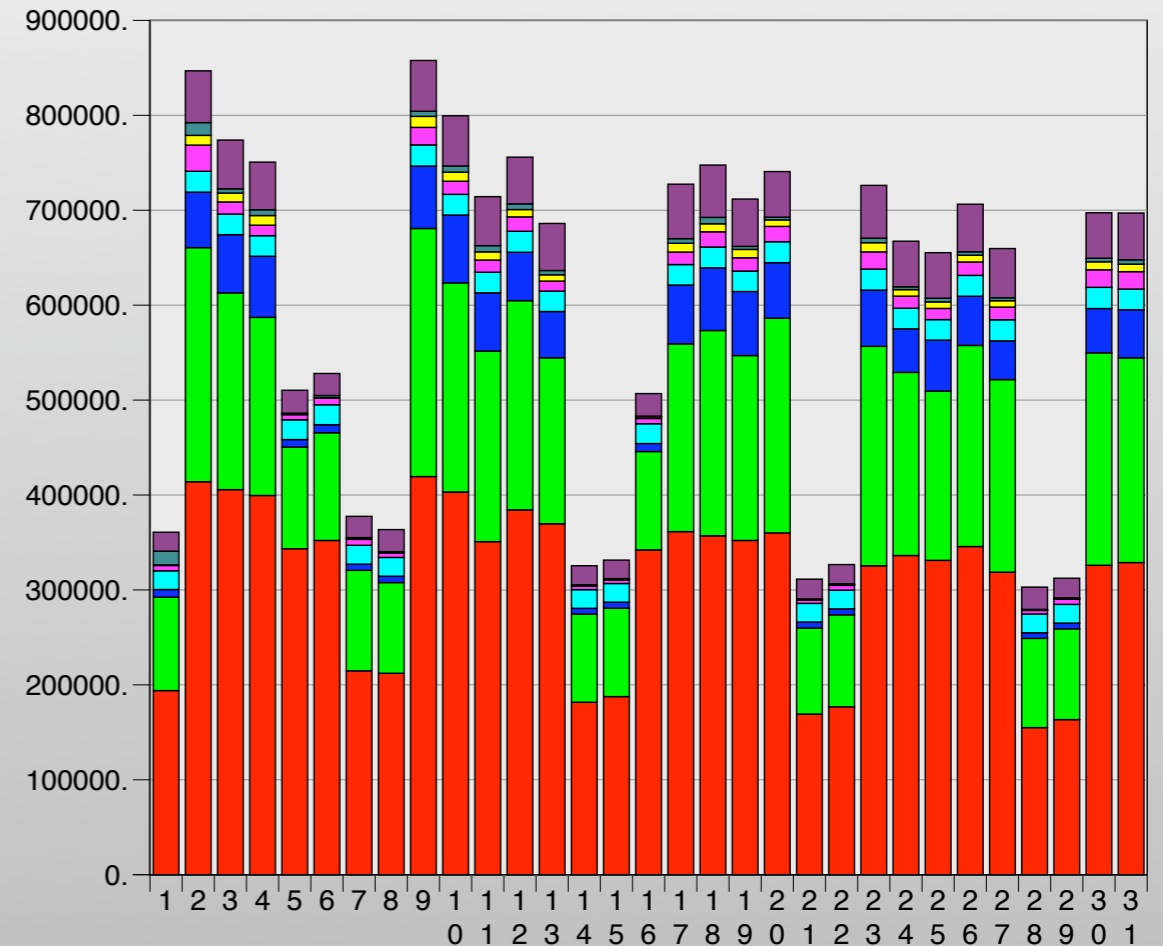
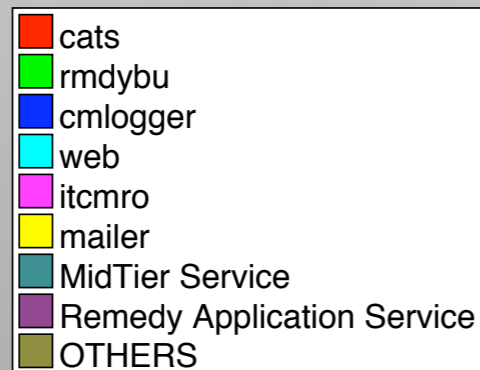
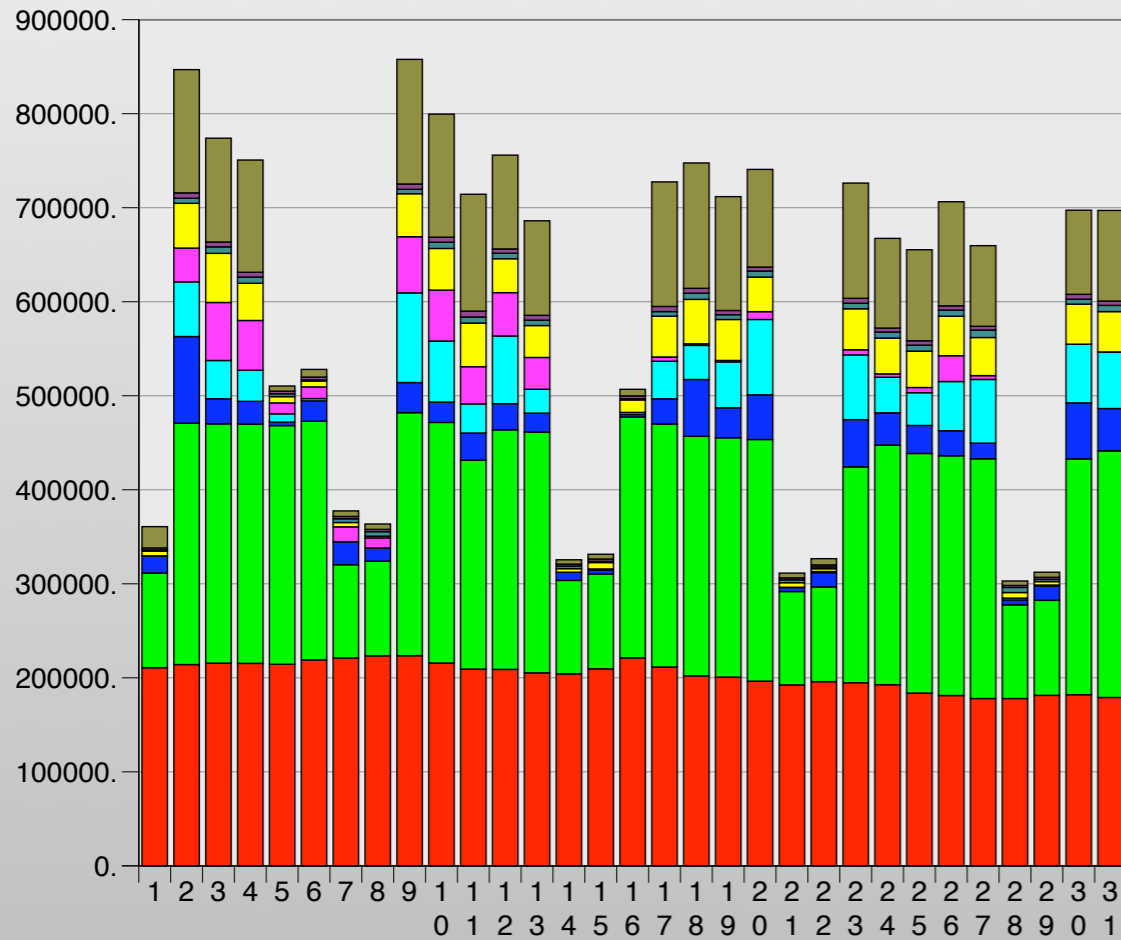
(demo)



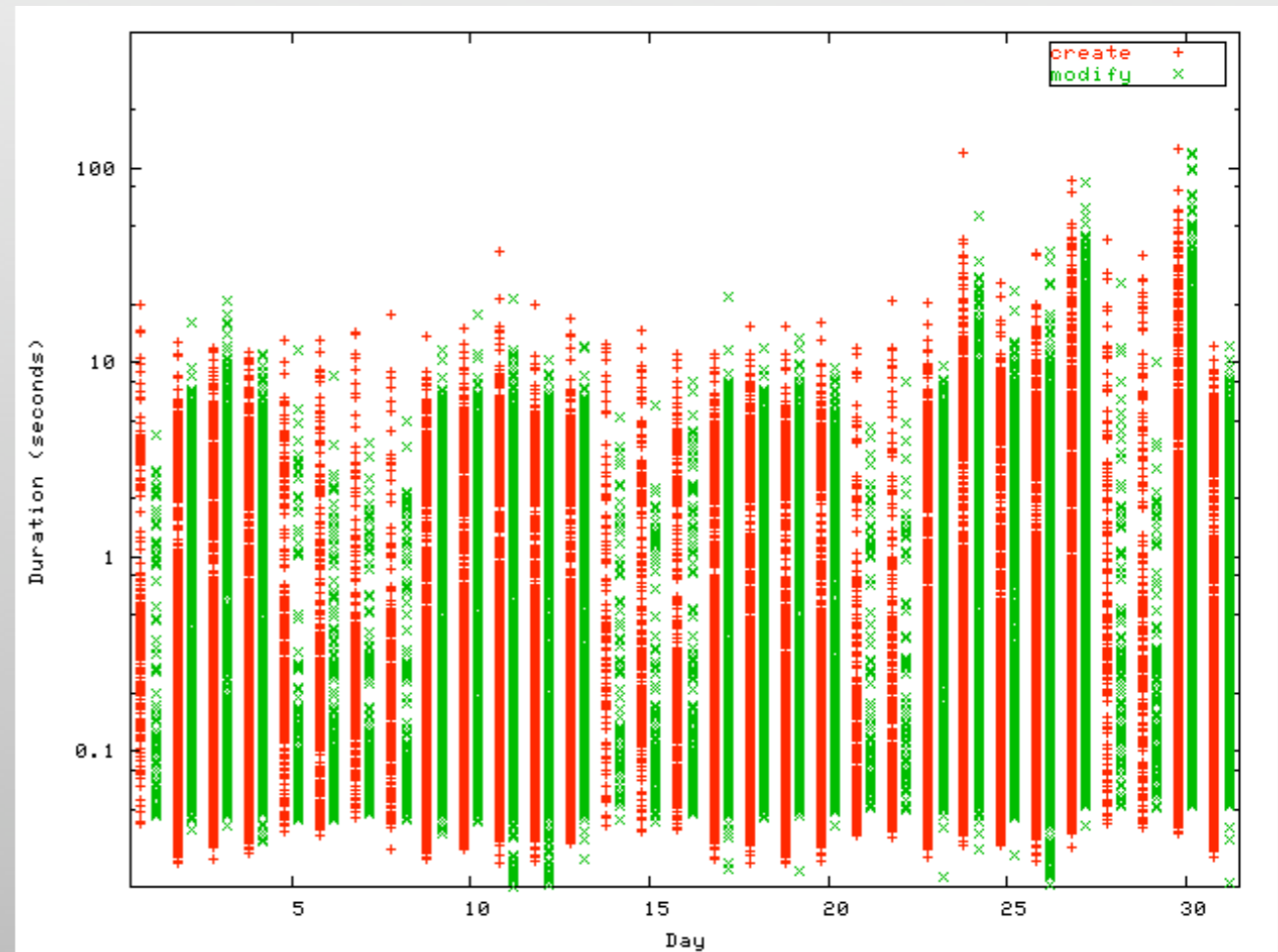
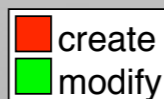
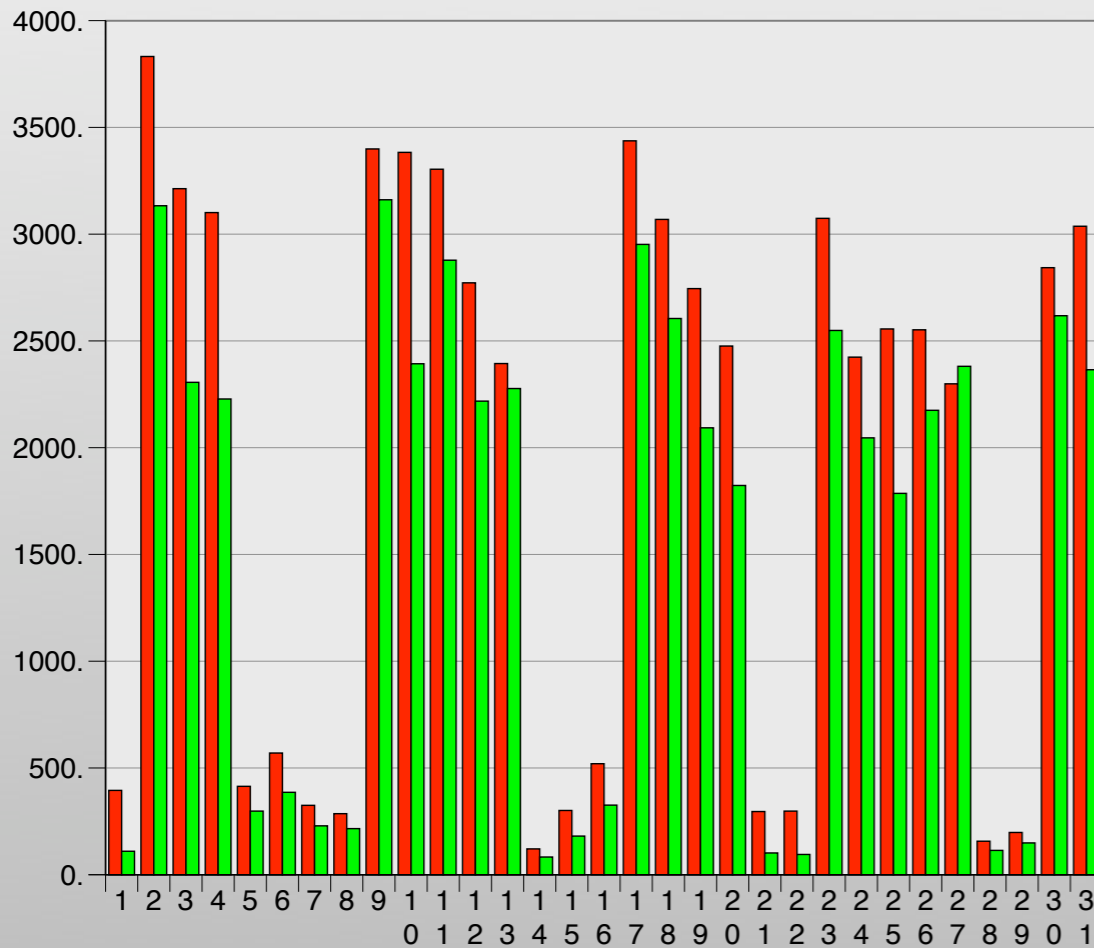
API log

- ```
<API > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast > <Client-RPC: 390620 > <USER: web >
/* Tue May 31 2005 09:27:00.8096 */ +GSF ARGetField -- schema PRMS:ProblemMgmt fieldId 7
<API > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast > <Client-RPC: 390620 > <USER: web >
/* Tue May 31 2005 09:27:00.8278 */ -GSF OK
```
- ```
( 0.0182 "Tue May 31 2005 09:27:00.8096" GSF      590263 "web"
"ARGetField -- schema PRMS:ProblemMgmt fieldId 7" "OK")
```
- This one is trivial to process
 - each call consists of two lines, the second OK or FAIL
 - the only issue is that the calls overlap
- But it takes 1.5 hours per day
 - almost 1 million API calls, 10MB files (compressed)

API - usage



API - performance



```

set terminal png; set output "create-set.png"
set pointsize 1; set logscale y
set key box; set xlabel "Day"; set ylabel "Duration (seconds)"
set xrange [0.5:31.5]; set yrange [0.02:500]
plot "create.dat" using ($1)-0.2:2 title 'create' with points pt 2,
      "set.dat" using ($1)+.2:2 title 'modify' with points pt 4
    
```

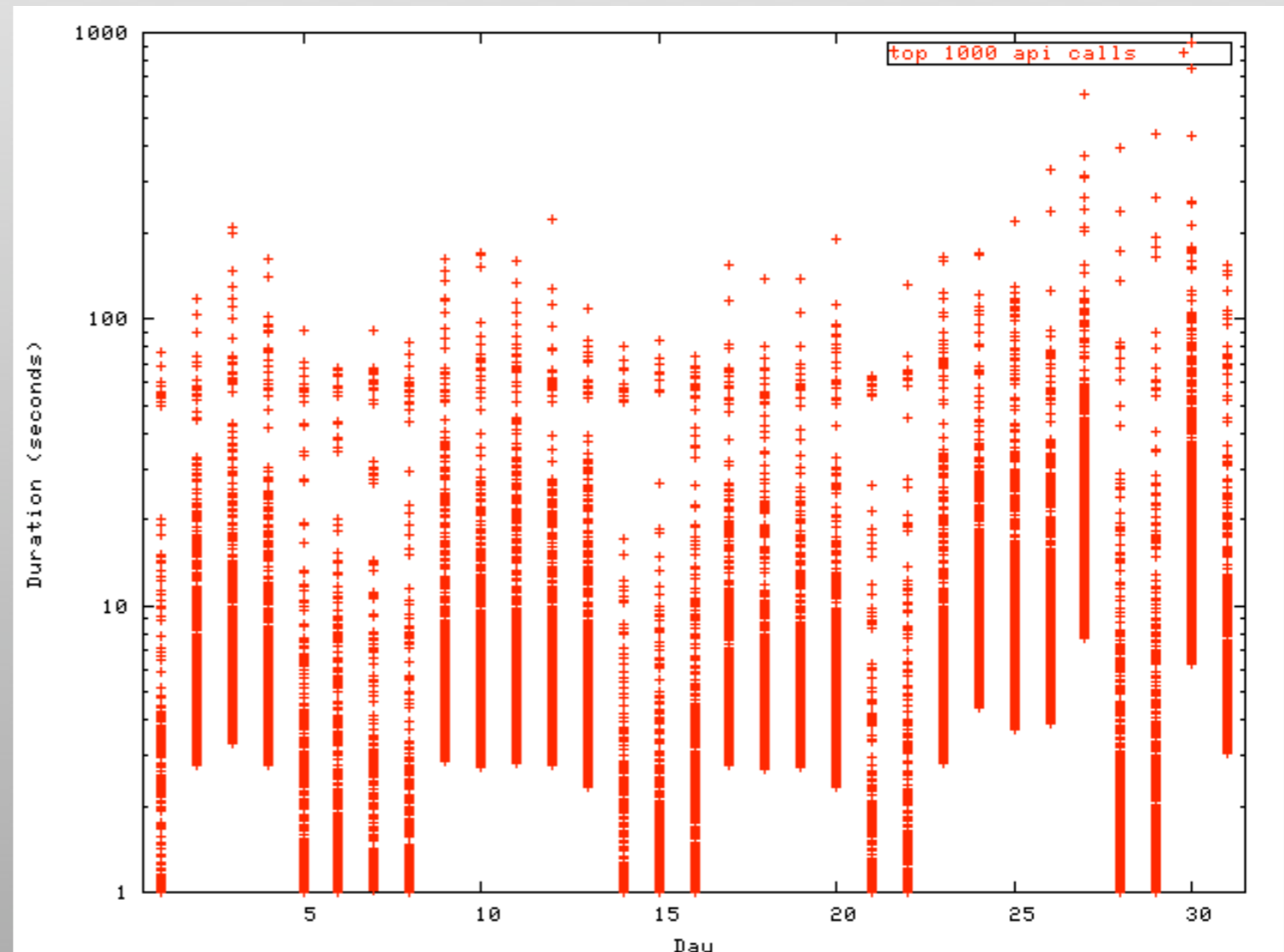
API - top 1000

- for day in 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31;
do gzcat api-2005-05-\$day.out.gz | sort -nr | head -1000 | tr -d '(' | awk '{print "'\$day'" " " \$1}';
done > api-top1000.dat
- (0.1328 "Mon May 30 2005



30 0.1328

- gnuplot <<EOF
set terminal png
set output "top1000.png"
set pointsize 1
set logscale y
set key box
set xlabel "Day"
set ylabel "Duration (seconds)"
set xrange [0.5:31.5]
plot "< grep -v '0.' < api-top1000.dat"
title 'top 1000 api calls'
with points pt 2
EOF



```
<API > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast      > <Client-RPC: 390620  > <USER: web      >
/* Tue May 31 2005 09:27:00.8096 */ +GSF      ARGetField -- schema PRMS:ProblemMgmt fieldId 7
<API > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast      > <Client-RPC: 390620  > <USER: web      >
/* Tue May 31 2005 09:27:00.8278 */ -GSF              OK

<SQL > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast      > <Client-RPC: 390620  > <USER: web      >
/* Tue May 31 2005 09:27:00.8150 */ SELECT helpText FROM field WHERE (schemaId = 413) AND (fieldId = 7)
<SQL > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast      > <Client-RPC: 390620  > <USER: web      >
/* Tue May 31 2005 09:27:00.8214 */ SELECT changeDiary FROM field WHERE (schemaId = 413) AND (fieldId = 7)
<SQL > <TID: 000013> <RPC ID: 0000590263> <Queue: Fast      > <Client-RPC: 390620  > <USER: web      >
/* Tue May 31 2005 09:27:00.8263 */ COMMIT WORK
```

- several statements can be executed for the same RPCID
 - calculate the time from each one to the next
 - stop looking for that pattern after the commit (or rollback)
 - or after some time, not all the sequences end with a commit (or rollback)
- statements can include line breaks
 - converted to [newline]

SQL queries

- Create indexes on selected fields
- Use them properly
 - Avoid *anywhere* in Query By Example
 - Use $>$ and $<$ in enumeration fields, not $!=$
 - *Status* $<$ "Closed", instead of *Status* $!=$ "Closed"
 - \geq and \leq are even better than $>$ and $<$
 - Isolate the indexed field
 - *Date* $<$ $\$TSS\-3600 , instead of $\$TSS\$-Date$ $>$ 3600

SQL queries (I)

- ```
SELECT T411.C1,C3,C536870924,C7,C536870918,C536871040,C4,C536870935,C2,C536870941,C536870920,T411.C1,T411.C1 FROM T411
WHERE ((T411.C536870921 LIKE 'CC-SYSADMIN%') AND (T411.C536870918 LIKE '%\lxfS6134%')) ORDER BY 13 DESC
```
- ```
SELECT * FROM ITCM_CALLMANAGEMENT WHERE
((DOMAIN LIKE 'CC-SYSADMIN%') AND (HOST LIKE '%\lxfS6134%'))
```
- QBE property of **host** is set to match *anywhere*
 - *leading* would be much faster (HOST LIKE 'lxfS6134%')
 - the user could add a % at the beginning to match everywhere

SQL queries (II)

SELECT T411.C1,C3,C536870924,C7,C536870918,C536871040,C4,C536870935,C2,C536870941,C536870920,T411.C1,T411.C1 FROM T411 WHERE ((T411.C536870921 LIKE 'CC-SYSADMIN%') AND (((T411.C7 != 6) AND (T411.C7 != 5)) AND (T411.C7 != 4))) ORDER BY 13 DESC

SELECT * FROM ITCM_CALLMANAGEMENT WHERE ((DOMAIN LIKE 'CC-SYSADMIN%') AND (((STATUS != 6) AND (STATUS != 5)) AND (STATUS != 4)))

□

□ *Status:* 0) New 2) In Hand 4) Rejected 6) Closed
 1) Assigned 3) Suspended 5) Completed

- The query *Status < 'Rejected'* is equivalent to the old one
 - this is a search written by a user
 - the only way to improve it is user education

Escalation log

- <ESCL> <TID: 000007> <RPC ID: 0000036689> <Queue: Escalation> <Client-RPC: 390603 > <USER: AR_ESCALATOR >
/* Tue May 03 2005 23:57:34.0013 */ Start escalation processing -- Operation
- <...> Checking CleanupAlertEvents (disabled) on Alert
- <...> Checking PRMS:PM-AutoClose (enabled) : going to fire in 178 seconds on PRMS:ProblemMgmt
- <...> Checking PRMS:SLA-Timers (enabled) : ready to fire now on PRMS:SLA-Timers
<...> --> Failed qualification
- <...> Checking PRMS:SendReminder (enabled) : ready to fire now on PRMS:Reminders
<...> --> Passed -- perform actions
<...> 0000000000004270
<...> 0: Set Fields
<...> zTmpNotifyNow (250000010) = 1
- <...> /* Tue May 03 2005 23:57:40.0529 */ Stop escalation processing
<...> set new firetime of 60 seconds
- (0000036689 START "Tue May 03 2005 23:57:34.0013")
(0000036689 FAILED "PRMS:SLA-Timers" "PRMS:SLA-Timers")
(0000036689 PASSED "PRMS:SendReminder" "PRMS:Reminders")
(0000036689 "0000000000004270")
(0000036689 "0: Set Fields")
(0000036689 "zTmpNotifyNow (250000010) = 1")
(0000036689 STOP "Tue May 03 2005 23:57:35.0960")

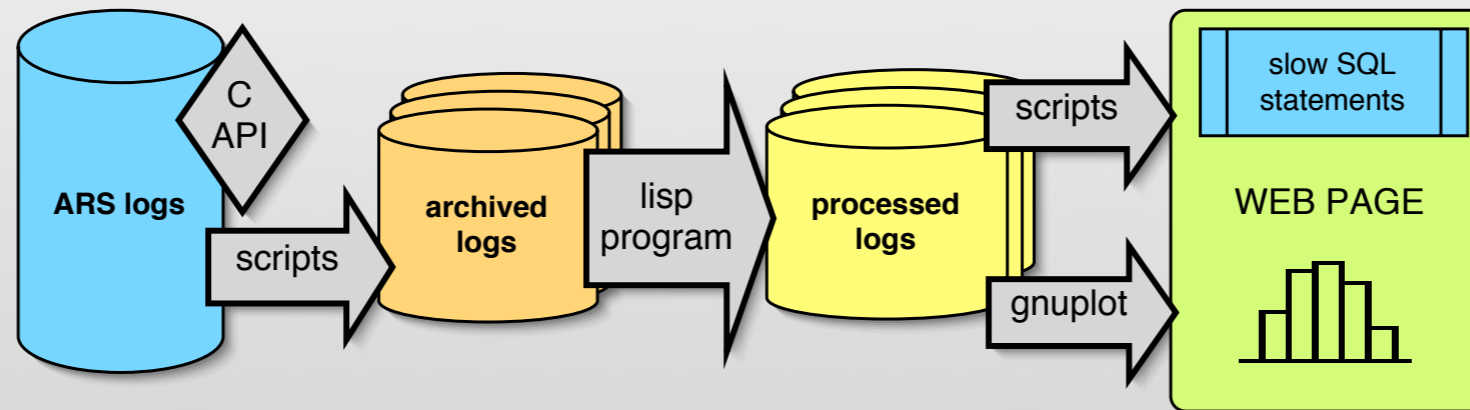
Escalation log

- Escalation log processing
 - takes around 15 minutes
 - daily log ~150MB, ~1 million lines
 - eliminates 75% of the lines (final size is 10%)
 - only keeps the entry when the escalation is actually run
 - the output is much more readable
 - the name of the escalation is in the same line as result
- The filter log is similar, but more complex

- Processing the filter log takes around 5 hours
- We split the output, entries for *GET* not interesting
 - we don't have workflow firing on *GET*
 - typically we get from 7 Mlines to 3 (+2 for *GET*)

- ```
(0003442251 "ungil" SET "Tue May 03 2005 10:39:49.7080")
(0003442251 "ungil" "Remedy User Preference" "0000000000000001")
(0003442251 "ungil" PASSED "RemedyUserPrefWarnMessage (500)")
(0003442251 "ungil" "0: Message")
(0003442251 "ungil" "You have modified your preferences using Remedy User Preferences
form. You need to log in again for these changes to take effect. [newline]")
(0003442251 "ungil" END-1 "Tue May 03 2005 10:39:49.7199")
(0003442251 "ungil" RESTART-3 "Tue May 03 2005 10:39:50.8116")
(0003442251 "ungil" STOP "Tue May 03 2005 10:39:50.8118")
```

# Implementation



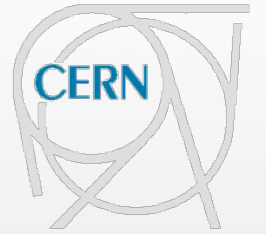
- The program processing the logs is written in Lisp
  - the LoGS framework is used, with rules for ARS logs
  - but a similar program could be written in Perl
- The log processing could be done online
  - currently runs nightly, not available until the next day



# LoGS

Super-Elite Enterprise Edition 2005

0.0.3



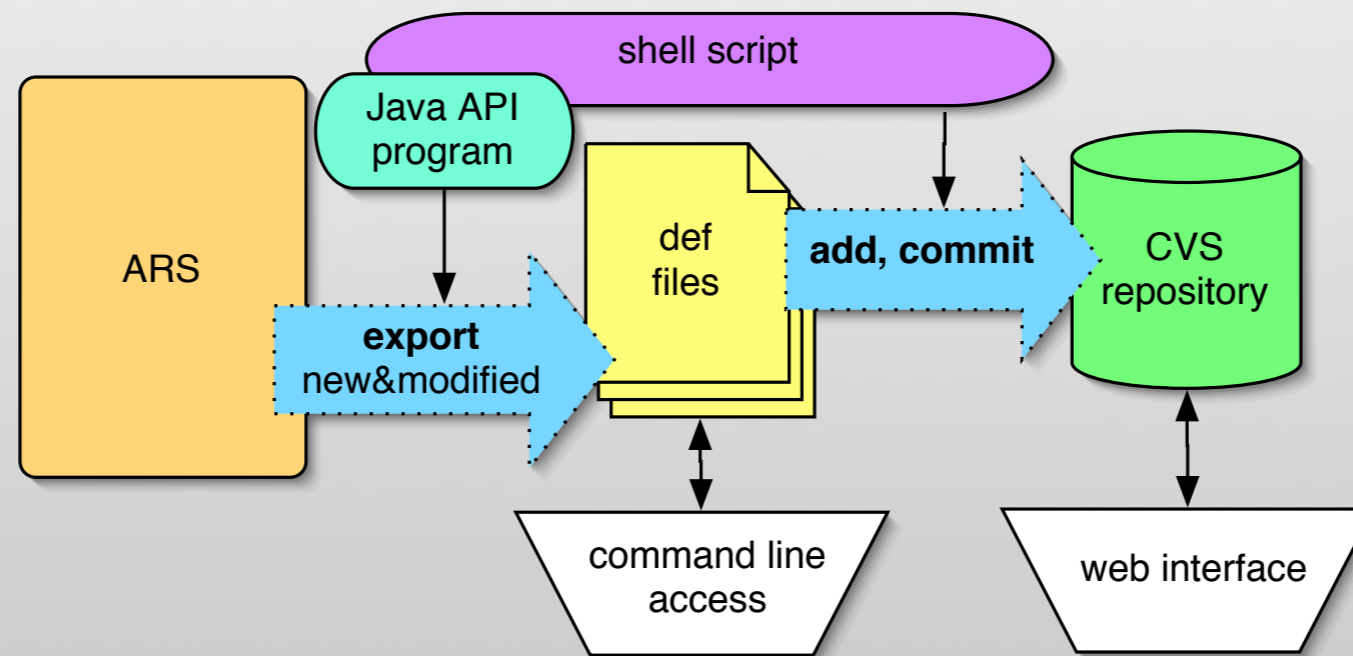
- LoGS is a dynamic, extendable log analysis engine that attempts to be useful for very large systems (...)
- Features:
  - Dynamic Ruleset
  - Extendable (even at run-time!)
  - Powerful configuration language (Common Lisp)
  - Fast Regexp Engine (CL-PPCRE)
  - Nest-able rulesets (allowing for greater efficiency than flat rulesets)
- <http://www.hpc.unm.edu/~download/LoGS/>

- Provided by Remedy/BMC
  - unsupported
  - Windows-only
  - calculates duration of actions in API/SQL logs
    - much faster (10x)
    - the format is not changed (new lines are added around blocks)
  - complete SQL transactions only (not individual queries)
    - *MidTier Service* GMAL are 1000's of queries (without commit)
    - when there is a commit in that thread, minutes are reported

# Change Tracking

- Often I was exporting to a file to search on it
  - v.g. to find external processes or SQL command
- I wrote a program to export each object to a .def file
  - it runs twice per hour, updating changed definitions
- Definitions are in CVS to keep track of changes
  - sometimes the diff shows clearly what has changed
  - old def files can be recovered and imported by hand
    - in principle, I've never tried

# How does it work?



- The Java API is used to export modified objects
  - one file per object
  - `java ArchiveDefs server timestamp dir`
- The scripts keeps the timestamp and commits in CVS

**defs\_sunar01**

Current directory: [\[remedy\]](#) / defs\_sunar01  
Files shown: 4041

[File](#)

- [filter HMS.SMSInstallClear](#)
- [filter HMS.SMSInstallSet](#)
- [filter HMS.SMSMoveClear](#)
- [filter HMS.SMSMoveSet](#)
- [filter HMS.SMSRepairClear](#)
- [filter HMS.SMSRepairSet](#)
- [filter HMS.SMSRetireClear](#)
- [filter HMS.SMSRetireSet](#)
- [filter HMS.RegInstallITCM](#)
- [filter HMS.RegInstallITCM-nostress](#)
- [filter HMS.RegInstallITCM-stress](#)
- [filter HMS.RegInstallITCM2](#)
- [filter HMS.RegInstallITCM3](#)
- [form PRMS](#)

**CVS log for defs\_sunar01/HMS:SMSInstallClear**

Up to [\[remedy\]](#) / [defs\\_sunar01](#)

[Request diff between arbitrary revisions](#) / [Display revisions](#)

Default branch: MAIN  
Bookmark a link to: [HEAD](#) / [\(download\)](#)  
Current tag: MAIN

Revision [1.5](#) / [\(view\)](#) - [annotate](#) - [\[select for diffs\]](#), Mon Jun 27 14:15:02 2005 (by *rmidybu*)  
Branch: [MAIN](#)  
CVS Tags: [HEAD](#)  
Changes since 1.4: +3 -3 lines  
Diff to [previous 1.4](#)

Revision [1.4](#) / [\(view\)](#) - [annotate](#) - [\[select for diffs\]](#), Sun Jun 27 15:15:01 2005 (by *rmidybu*)  
Branch: [MAIN](#)  
Changes since 1.3: +3 -3 lines  
Diff to [previous 1.3](#)

Revision [1.3](#) / [\(view\)](#) - [annotate](#) - [\[select for diffs\]](#), Tue Jun 27 13:15:07 2005 (by *ungiladm*)  
Branch: [MAIN](#)  
Changes since 1.2: +6 -3 lines  
Diff to [previous 1.2](#)

Revision [1.2](#) / [\(view\)](#) - [annotate](#) - [\[select for diffs\]](#), Fri Sep 2 14:15:02 2005 (by *ungiladm*)  
Branch: [MAIN](#)

**defs\_sunar01/active link ITCM:CM:WEB:HideTabs - diff - 1.3**

Return to [active link ITCM:CM:WEB:HideTabs](#) CVS log

Up to [\[remedy\]](#) / [defs\\_sunar01](#)

Diff for /defs\_sunar01/active link ITCM:CM:WEB:HideTabs between version 1.2 and 1.3

| version 1.2, 2005/05/27 12:15:07         | version 1.3, 2005/05/27 13:15:07                              |
|------------------------------------------|---------------------------------------------------------------|
| <b>Line 1</b>                            | <b>Line 1</b>                                                 |
| #                                        | #                                                             |
| # File exported Fri May 27 14:15:02 2005 | # File exported Fri May 27 15:15:01 2005                      |
| #                                        | #                                                             |
| begin active link                        | begin active link                                             |
| name : ITCM:CM:WEB:HideTabs              | name : ITCM:CM:WEB:HideTabs                                   |
| timestamp : 1117195707                   | timestamp : 1117198926                                        |
| export-version : 6                       | export-version : 6                                            |
| owner : ungil                            | owner : ungil                                                 |
| last-changed : ungiladm                  | last-changed : ungiladm                                       |
| <b>Line 13</b>                           | <b>Line 13</b>                                                |
| actlink-mask : 279056                    | actlink-mask : 279056                                         |
| enable : 1                               | enable : 1                                                    |
| permission : 0                           | permission : 0                                                |
| actlink-query : 4\1\2\1\22\2\2\9\        | actlink-query : 1\4\1\2\1\22\2\2\9\4\6\2\1\16\2\4\8\Sysadmin\ |
| action {                                 | action {                                                      |
| display-prop : 1\4\6\0\                  | display-prop : 1\4\6\0\                                       |
| id : 999999037                           | id : 999999037                                                |

Legend:

- Removed from v. 1.2
- changed lines
- Added in v. 1.3

Colored Diff

[CERN Central CVS service](#)

Powered by [ViewCVS 0.9.2](#)

# If you want to try...

- ...you can get it here *<http://cern.ch/ungil/remedy>*
- There are some issues
  - an object name could contain “/”
    - which is not a valid character in a filename
  - if an object is removed we don't notice
    - the system can be extended, using another API program to list the objects in the server and compare with the existing files
    - files referring to removed objects could then be removed (after committing a last change to CVS, with “deleted” as comment)

# Questions?



*Optimization and Troubleshooting Guide (since 5.1)*