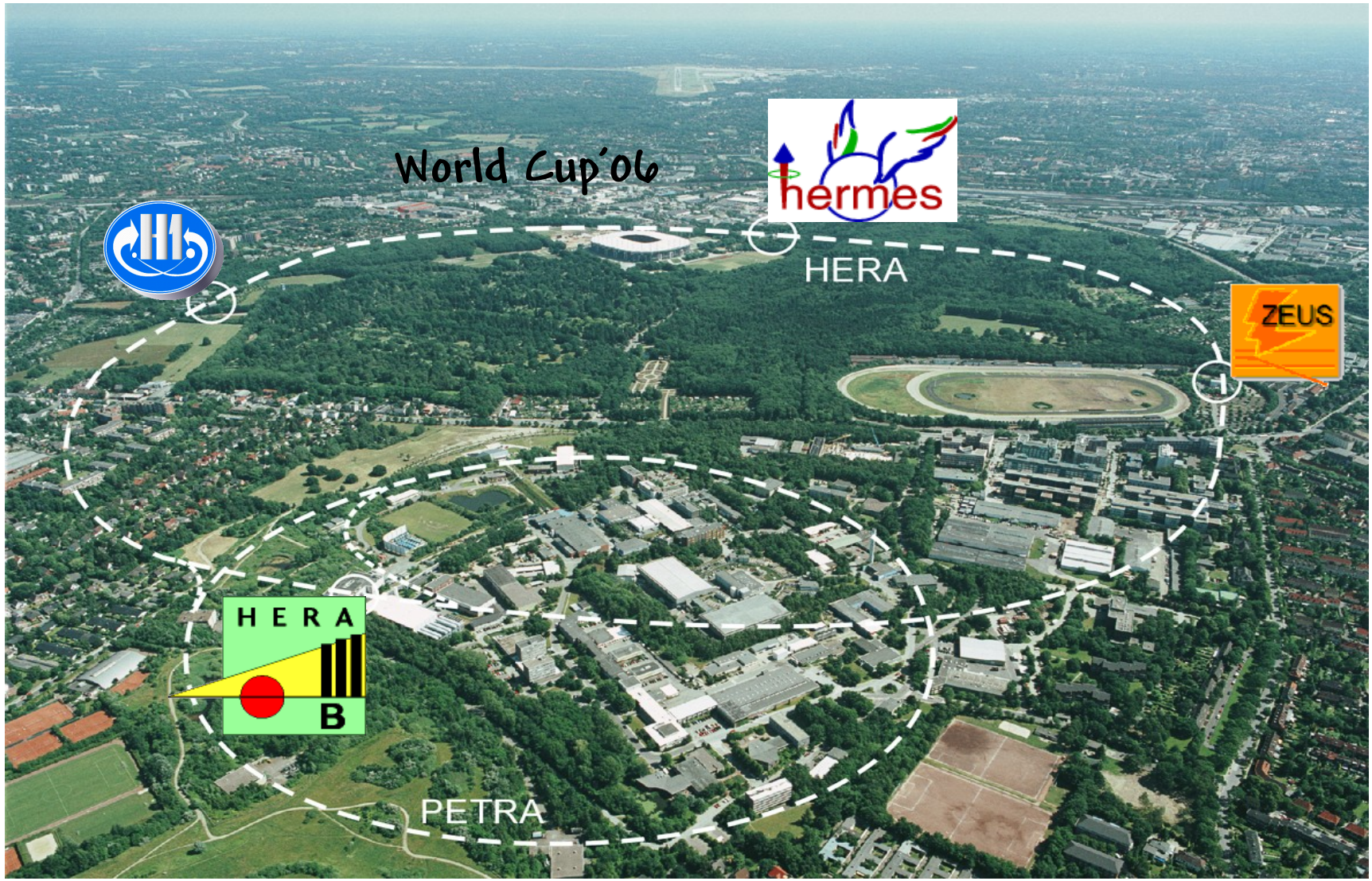
Support of Kerberos 5 Authenticated Environment by TORQUE

Bogdan Lobodzinski

bogdan@mail.desy.de

DESY, H1 Collaboration


World Cup'06
hermes
HERA
ZEUS
H1
HERA
B
PETRA

Computing infrastructure

- Desktops
- Work Group Servers
- File Servers
- GRID (mainly MC production)
- Computing Farm
  - about 200 hosts with Dual CPUs/host: Pentium III 1 Ghz, Xeon 3GHz, Dual Core AMD Opteron 270, 2GHz
  - RAM: 0.5 - 4 GB, local HD: 10 - 300 GB
  - Batch System: TORQUE 2.0.0p7 with modifications

Kerberos V Ticket (Credential, TGT) handling Mechanisms

- using Authenticated Remote Control tool (ARCv2)
  http://search.cpan.org/dist/ARCv2/
  - ARC server must be running on KDC
- using local server-client RPC connections
  - solution independent of KDC,
  - Kerberos Credential lifetime limited by KDC settings

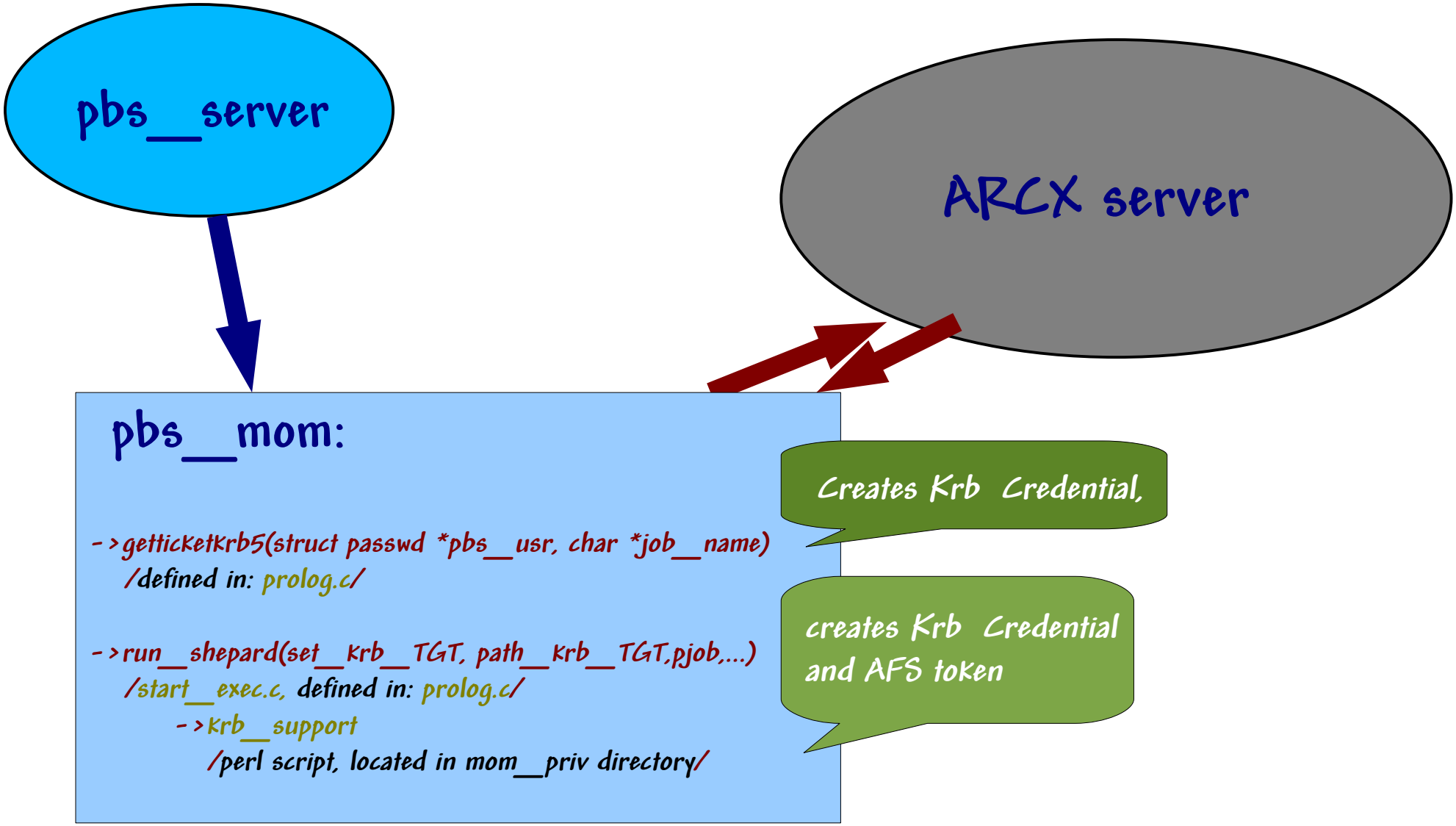
ARCv2 + TORQUE: Why ARCv2 ?

- ARCv2 is a S(imple) A(uthentication) S(ecurity) L(ayer) based Client/Server application (by Patrick Boettcher).
  - SASL supports various authentication techniques (GSSAPI - Kerberos V, MD5 - for passwords)
- ARCv2 provides generic functions on client and server sites (whoami, uptime, etc)
- ARCv2 allows to add private commands (krbbatchinit)
- another ARCv2 approach:

  "AFS file space administration with ARC version 2" by Wolfgang Friebel

  (http://hepwww.rl.ac.uk/hepix/nesc/friebel.ppt)

ARCv2 + TORQUE: How to - general picture



ARCv2 + TORQUE: how to - more details

- Set up ARCX Server and Client (details are described in docs of ARCv2 package: http://search.cpan.org/dist/ARCv2/)
  - install ARCv2 on KDC (arcxd.conf),
  - add krbbatchinit command,
  - set up KRB V arcx principals,
  - create ARCX Clients (arcx.conf),

ARCv2 + TORQUE: how to - krbbatchinit command

```
package Arc::Command::krbbatchinit;
# written by Patrick Boettcher
use strict;
use warnings;
use File::Temp qw/ :POSIX /;
use Arc::Command;
use vars qw(@ISA $VERSION $kadmin $kinit $fileprefix $accept $deny $subpath $local $pagsh);
use IO::Socket::INET;

$VERSION = "0.02";

# commands:
$kadmin = "/opt/products/sbin/kadmin";
$kinit = "/opt/products/bin/kinit";
$fileprefix = "/x01/pbs/arc";
$subpath = "krbbatchinit";
$accept = "arc.accept";
$pagsh = "/usr/afsws/bin/pagsh";
$local = 1;
$deny = "arc.deny";

@ISA = qw(Arc::Command);

sub members
{
        my $this = shift;
        return { %{$this->SUPER::members},
                # private:
                # protected:
        };
}
```


```
sub Execute
{
        my $this = shift;
        my ($principal, $address) = @_;

        return $this->_SetError("No principal given. (usage <cmdname> <principal> <targethost>)") unless $principal;
        return $this->_SetError("No target host given. (usage <cmdname> <principal> <targethost>)") unless $address;

        my $path = $Arc::ConfigPath."/".$subpath;

        open(FH,"<$path/$accept") or return $this->_SetError("ACL-principal file not found ($path/$accept). Aborting.");
        return $this->_SetError("Access denied (user).") unless grep /^$this->{_username}$/, <FH>; close(FH);

        open(FH,"<$path/$deny") or return $this->_SetError("Deliverable-principals file not found ($path/$deny). Aborting.");
        return $this->_SetError("Access denied (principal).") if grep { chop; $principal =~ /$_/g } <FH>; close(FH);

        $address = gethostbyname($address) or return $this->_SetError("Hostname resolve error: $!");
        $address = inet_ntoa($address) or return $this->_SetError("Hostname resolve error: $!");

        my ($ktfile,$ccfile) = ("",""); $ktfile = tmpnam().$$; $ccfile = $ktfile.".cc";
        # get the key from KRB-DB
        my $l = $local ? "-l" : "-p wfr/admin -K$path/admin_keytab";
        my $ret = system("$kadmin $l ext_keytab -k $ktfile $principal > /dev/null 2>&1");
        return $this->_SetError("Could not ext_keytab for $principal. Kadmin failure.") if $ret;
        # create the TGT file with extracted key:
        $ret = system("$pagsh $kinit -k -t $ktfile -c $ccfile -a IPv4:$address $principal > /dev/null 2>&1");
        return $this->_SetError("Could not kinit a CCACHE for $principal. Kinit failure.") if $ret;

        if (open (FH,"<$ccfile")) { # pass the CC file to the Client
                print <FH>; close(FH);
        } else { $this->_SetError("No ccache file generated?!"); }

        unlink $ccfile; unlink $ktfile;
        1;
}
1;
```

ARCv2 + TORQUE: how to - TORQUE modifications

- modifications are necessary in the files: prolog.c and start_exec.c (+ some header files)
  - prolog.c:

    ->getticketkrb5(struct passwd *pbs_usr, char *job_name)

    create initial KRB Ticket:

    "ARCX -U host/hostname -h arcx_server -s GSSAPI krbbatchinit principal hostname -k host/hostname";

    ->run_shepard(set_krb_TGT,path_krb_TGT,pjob,IO_TYPE_NULL)

    starts the job monitoring script: krb_support

ARCv2 + TORQUE: how to - TORQUE modifications

```
++ prolog.c                2006-08-24 20:25:18.000000000 +0200

@@ ... @@
   return;
   }  /* END pelogalm() */

+/*
+ * run_shepard() - starts wrapper for krb 5 Credential
+ *  creation/prolongation
+ * Wrapper must be owned by root uid,
+*/
+
+int run_shepard(
+  int   which,      /* I (one of PE_*) */
+  char *pelog,      /* I - script path */
+  job  *pjob,       /* I - associated job */
+  int   pe_io_type) /* I */
+{
.....
+  setpag();
+  child = fork();
....
+  if (child == 0) {
....
+    arg[0] = pelog;
+    arg[1] = pjob->ji_qs.ji_jobid;
+    arg[2] = pjob->ji_wattr[(int)JOB_ATR_euser].at_val.at_str;
...
+    execv(pelog,arg);
...
+}  /* END run_shepard() */
```

```
    if (stat(pelog,&sbuf) == -1)
     {
@@ ... @@
     arg[4] = pjob->ji_wattr[(int)JOB_ATR_jobname].at_val.at_str;
+if ( which == PE_PROLOG ) {
+    rc =
getticketkrb5(getpwnam(pjobji_wattr[(int)JOB_ATR_euser].at_val.at_str),
pjob->ji_qs.ji_jobid); }
     /* NOTE:  inside child */
```

ARCv2 + TORQUE: how to - TORQUE modifications

```
+++ start_exec.c          2006-08-24 20:25:18.000000000 +0200

@@ .... @@
 extern       char                        *path_jobs;
 extern       char                        *path_prolog;
+extern  char          *path_krb_TGT;
 extern  char          *path_prologuser;

@@ .... @@
   "PBS_MOMPORT",
   "PBS_NODEFILE",
-  "TMPDIR" };
+  "TMPDIR",
+  "KRB5CCNAME" };

 static        int num_var_else = sizeof(variables_else) / sizeof(char *);

@@ -1108,6 +1116,18 @@
   if (!usertmpdir && TTmpDirName(pjob,buf))
     bld_env_variables(&vtable,variables_else[12],buf);

+    /* setup KRB5CCNAME */
+    sprintf(buf,"%s/%s/%s/krb5cc_%d_%d",
+      PBS_SERVER_HOME,getkrb5cachepath("KRB5_CACHE"),
+      pjob->ji_qs.ji_jobid,pwdp->pw_uid,atol(pjob->ji_qs.ji_jobid) );
+
+    bld_env_variables(&vtable,variables_else[13],buf);
+
   /* passed-in environment for tasks */



    if (envp != NULL)
@@ .... @@

      setwinsize(pts);     /* set window size to qsub's */
```

```
   setwinsize(pts);       /* set window size to qsub's */

+  /* run run_shepard - renewing krb TGT */
+   setpag();
+
+   if ( ( rc = run_shepard(set_krb_TGT,path_krb_TGT,pjob,PE_IO_TYPE_NULL) ) != 0) {
+         log_err(errno,id,"interactive shepard failed (renewing krb TGT)");
+         starter_return(TJE->upfds,TJE->downfds,JOB_EXEC_FAIL2,&sjr);
+   }
+   sleep(1);
+
    /* run prolog - interactive job */

     if (run_pelog(
@@ .... @@
     /* i/o is to slave tty                                    */

+    sjr.sj_session =  getpid();
     close(0);
     dup2(pts,0);

@@ .... @@
     starter_return(TJE->upfds,TJE->downfds,JOB_EXEC_FAIL1,&sjr);
     }

+    /* run run_shepard - renewing krb TGT */
+    setpag();
+
+    if ( ( rc = run_shepard(set_krb_TGT,path_krb_TGT,pjob,PE_IO_TYPE_NULL) ) != 0) {
+          log_err(errno,id,"batch shepard failed (renewing krb TGT)");
+          starter_return(TJE->upfds,TJE->downfds,JOB_EXEC_FAIL2,&sjr);
+    }
+    sleep(1);
+
    /* run prolog - standard batch job */

    if ((j = run_pelog(
```

ARCv2 + TORQUE: how to - krb_support

```
#!/opt/products/bin/perl -w

use strict;
use AFS qw(setpag);

setpag;

my $BASE_PATH = "/x01/pbs/var/spool";
my ($uid, $jobid, $jobname, $hostname, $command, $principal, $left, $right, $rec, $index, $timeleft, $krbinit);
my $arcx_host = "arcx_server";

my $CAT = "/bin/cat -A";
my $LESS = "/usr/bin/less";
my $QSTAT = "/h1/torque/i586_linux24/bin/qstat";
my $delay = 1;
my $ARCX = "/x01/arcv2/arcx"
my $KINIT = "/opt/products/bin/kinit";

my $checkinterval = 1000; # in sec;
# the time difference between "NOW" and the expiration date of the specific credential
# when the renew procedure will start:
my $timeset = 2400;       # in sec;

$jobname = $ARGV[0];
$principal = $ARGV[1];

$jobid = (split(/\./, $jobname))[0];
$hostname = `hostname -f`;
chop $hostname;

# set environment:
$uid    = getpwnam($principal) if defined($principal);
my $ccname = "$BASE_PATH/$jobname/krb5cc_".$uid."_".$jobid;
$ENV{'KRB5CCNAME'} = $ccname;
```

ARCv2 + TORQUE: how to - krb_support

```
# find the main job process:
my $pid = -1;
while ( $pid < 0 ) {
        $pid = FindJobChild($jobid);
        exit if ( $pid == -2 );
        sleep 1;
}

# get group ID:
my @job_groups = FindGroupID($pid);

my $setgroups_list;
foreach my $item ( @job_groups ) {
        $setgroups_list .= $item . " " . $item . " ";
}

# set groups:
$) = $setgroups_list;
```

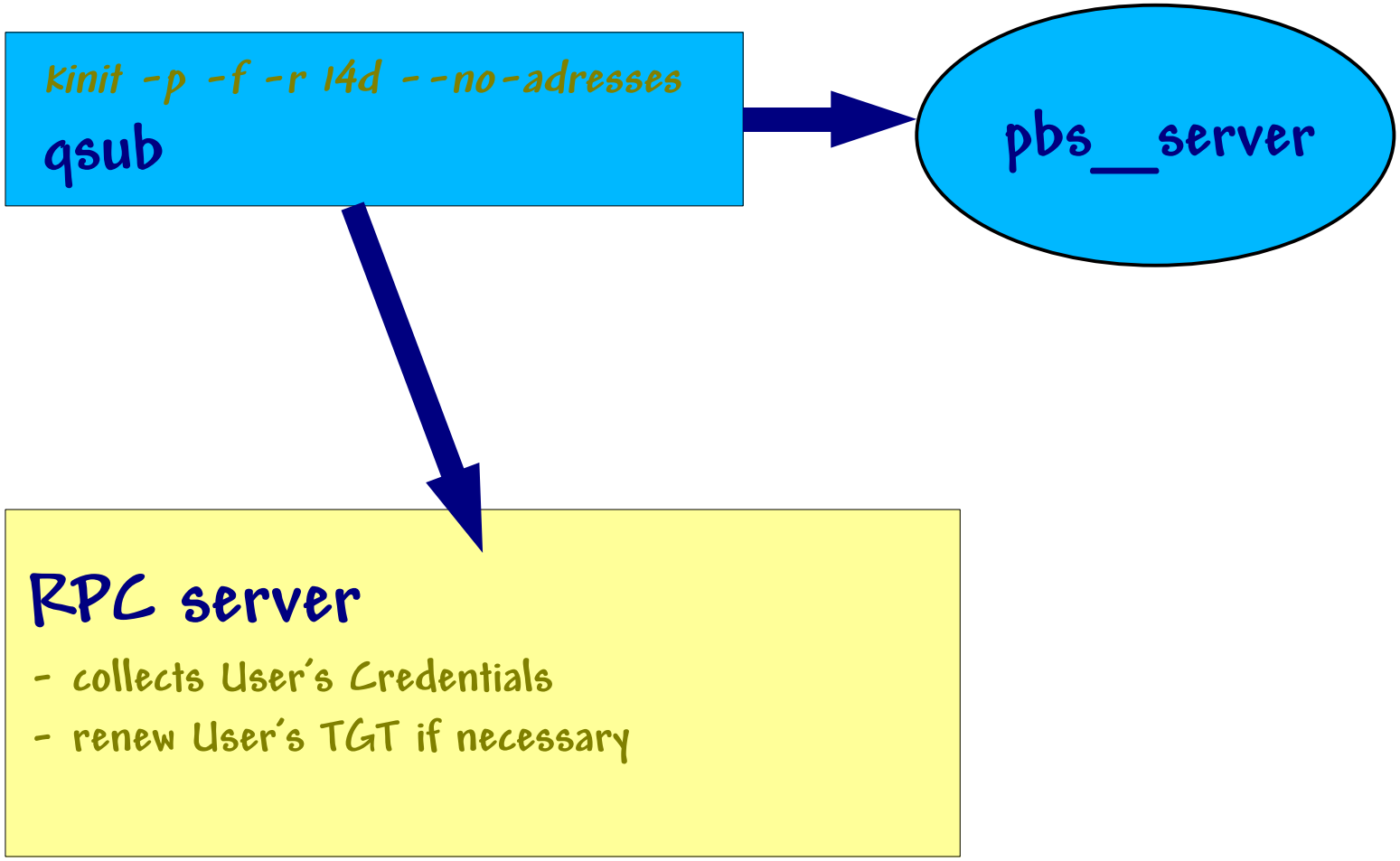
ARCv2 + TORQUE: how to - krb_support

```
# keep process running as long as the job is running / and process state is not ZOMBI
my $ret = 1; $index = 0;
while ( $ret == 1 ) {
        sleep $delay;
        $ret = CheckJobProcess($pid);
        $index++;

        # determine the difference between Expiration time of krb5 ticket and present time:
        if ( ( $index % $checkinterval ) == 0 ) {
                $timeleft = gettimeleft($ccname);

                # create krb5 ticket and afs token before expiration time:
                if ( $timeleft < $timeset ) {
                        # prepare krbbatchinit call
                        $krbinit = "$ARCX -U host/$hostname -h  $arcx_host -s GSSAPI krbbatchinit $principal $hostname";
                        # get a ticket to be able to use krbbatchinit service
                        system("$KINIT -k host/$hostname");
                        # issue a ticket for the user via krbbatchinit
                        system($krbinit." >".$ccname);
                        # chown the krb5cc
                         chown scalar getpwnam($principal) , 0, $krbccfile;
                        # get AFS token:
                        $command = "/bin/su $principal -c \"/opt/products/bin/kinit -R -f -p \" ";
                        system($command);
                }
                $index = 0;
        }

}
```

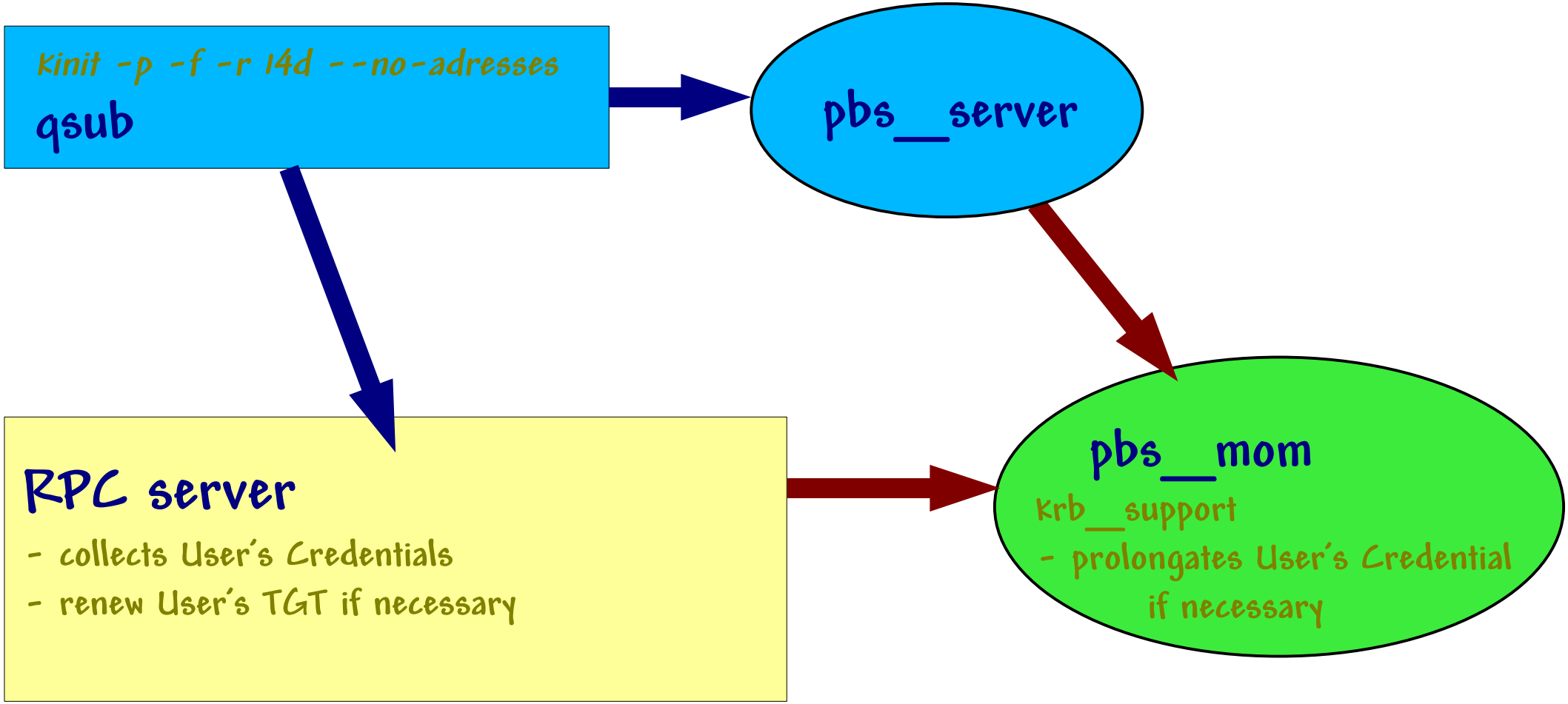
RPC Client-Server + TORQUE:
How to - general picture (I)
Preparation

```
kinit -p -f -r 14d --no-adresses
```

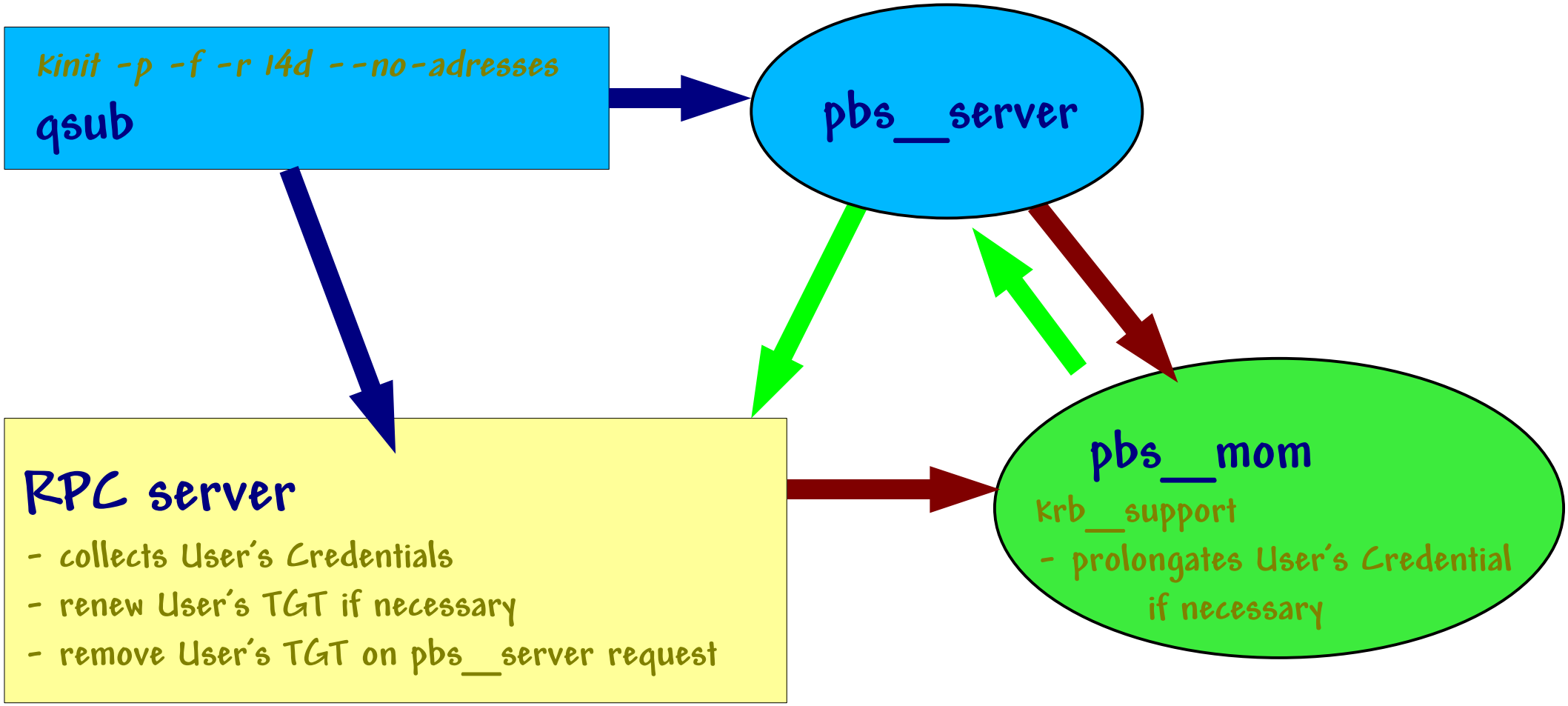
RPC Client-Server + TORQUE:
How to - general picture (2)
batch job submission



RPC Client-Server + TORQUE:
How to - general picture (3)
batch job execution



RPC Client-Server + TORQUE:
How to - general picture (4)
the end of batch job execution



RPC Client-Server + TORQUE: How to - details

- First version of RPC Server/Client has been written by Tigran Mkrtchyan as a base for AFS token handling mechanism in openpbs (release from 2000)
- RPC Server/Client is based on RPC/XDR library
- modified version of TORQUE (2.0.0p7) and RPC server is available from bogdan@mail.desy.de

RPC Client-Server + TORQUE: How to - details

- *initial Kerberos V Ticket must be:*
  forwardable, proxiable, renewable, without address *and with*
  renewable-lifetime *long enough.*

```
% kinit -p -f -r 14d --no-addresses
bogdan@DESY.DE's Password:

% klist -v
Credentials cache: FILE:/tmp/krb5cc_4058_bxoNUQ
        Principal: bogdan@DESY.DE
    Cache version: 4

Server: krbtgt/DESY.DE@DESY.DE
Ticket etype: des-cbc-crc, kvno 11
Auth time:  Oct  4 21:06:06 2006
Start time: Oct  4 21:06:07 2006
End time:   Oct  5 21:06:07 2006
Renew till: Oct 18 21:06:06 2006
Ticket flags: forwardable, proxiable, renewable, transited-policy-checked
Addresses:
....
```

Kerberos V + TORQUE

- *for both methods (ARCv2, RPC Client/Server) running processes on the execution host look like:*

```
|-pbs_mom,root  -p
  |-zsh,kcerny
  |   |-krb_support,root  /pbs/var/mom_priv/krb_support  251945.h1farm03.desy.de  kcerny
  |   `-sh  /pbs/var/mom_priv/jobs/251945.h1fa.SC
  |       `-main  -f  data.39.steer
  `-zsh,knowak
      |-251321.h1fa.SC /pbs/var/mom_priv/jobs/251321.h1fa.SC
      |   `-oosubset  -f  subset.WSpacal.Data04.steer
      `-krb_support,root  /pbs/var/mom_priv/krb_support  251321.h1farm03.desy.de  knowak
```

Kerberos V Ticket handling Mechanisms: Summary

- ARCv2 tool is easy to integrate (and updates) with any release of TORQUE (++)
- ARCv2 requires access to core of the security infrastructure (-)
- Integration (updates) of RPC Client/Server with TORQUE is highly complicated (--)
- Usage of RPC Client/Server does not depend on security infrastructure (++)
- Implementation of RPC Client/Server requires weaker Kerberos Security (option: --no-addresses) (--)

THANK YOU FOR YOUR ATTENTION !!