Obliczenia Wielkiej Skali

Condor® vs. Anthill vs. POV-Ray™

Marcin Rociek
Informatyka, IV rok

Spis treści

1. Instalacja i konfiguracja Condora

Moim podstawowym zadaniem było skonfigurowanie na *anthill'*u Condora tak, żeby był on używalny - to znaczy żeby
1. sprawnie działał,
2. mógł być wykorzystywany przez dowolnego użytkownika posiadającego konto na tej maszynie.

1.1. Poprzednia instalacja

Pozostała w spadku poprzednia instalacja Condora pozostawiała wiele do życzenia, gdyż w zasadzie nie dało się tego systemu używać. Podstawowymi problemami było:
1. Program użytkownika (który chciał skorzystać z usług Condora) można było owszem uruchomić (tzn. „wysłać" go poleceniem *condor_submit*), ale w żaden sposób nie dało się uzyskać wyniku działania tego programu. Nie dawało pozytywnych rezultatów ani wypisywanie wyniku na standardowe wyjście, ani zapis do pliku. Jedyną informacją jaką można było uzyskać było to, że program został uruchomiony na jakiejś maszynie i że zakończył działanie.
   W przypadku Condora to gdzie ma się znaleźć wynik standardowego strumienia wyjściowego oraz standardowego strumienia błędów definiujemy w pliku konfiguracyjnym wysyłanego zadania (jeśli nie podamy gdzie te strumienie mają się znaleźć, to wylądują one w `/dev/null`). Z reguły wygląda to w następujący sposób (fragment takiego pliku konfiguracyjnego):

   ```
   output = plik.out
   error  = plik.err
   log    = plik.log
   ```

   Przy próbie wysyłania zadania poleceniem *condor_submit* wyświetlany był radosny komunikat:

   ```
   WARNING: File /home/anthill/mrock/plik.out is not writeable by condor.
   WARNING: File /home/anthill/mrock/plik.err is not writeable by condor.
   ```

   Należy przy tym nadmienić, że:
   a) Pliki `plik.out` i `plik.err` były tworzone (czyli jednak Condor *mógł* je utworzyć), lecz miały długość 0 bajtów.
   b) Plik `plik.log` był tworzony i na dodatek wypełniany danymi, choć znajdował się w tym samym katalogu co pozostałe dwa (!).

   Z pliku z logiem można było odczytać, z której maszyny zadanie zostało wysłane, na której uruchomione, ile to trwało itp. lecz w żaden sposób nie można było uzyskać informacji od samego programu. Nie pomagało nawet samodzielne otwieranie i pisanie do pliku przez program, nawet jeśli plik otwierany był w `/tmp`. Również zmuszanie Condora do tworzenia tych plików wyjściowych w katalogu `/tmp` nic nie dało, gdyż objawiało się takim samym błędem.
2. Nie działał skrypt *condor_compile* umożliwiający programowi skorzystanie z usług zdalnych wywołań funkcji systemowych oraz *checkpointingu*, które to usługi dostępne są dopiero po przelinkowaniu programu z odpowiednimi bibliotekami Condora. Wszelkie próby modyfikacji tego skryptu kończyły się niepowodzeniem i dość bogatą liczbą błędów linkowania pojawiających się przy próbie jego użycia.

Z wyżej wymienionych powodów postanowiłem, że najprostszym rozwiązaniem będzie zainstalowanie Condora zupełnie od zera, co m.in. zostało umożliwione dzięki uprzejmości administratora, który udostępnił *sudo*.

1.2. Instalacja

Instalacja rozpoczęła się od ściągnięcia wersji 6.2.1 systemu z oficjalnej strony Condora[1]. Warto w tym miejscu dodać, że numeracja wersji Condora wzorowana jest na numeracji wersji jądra Linuxa, tzn. wersje stabilne mają cyfrę parzystą na drugiej pozycji.

Normalnie przed rozpoczęciem instalacji należy utworzyć w systemie użytkownika *condor*, lecz w tym przypadku krok ten mógł zostać pominięty, ponieważ użytkownik taki już został utworzony przy okazji poprzedniej instalacji.

Proces instalacji za pomocą skryptu *condor_install* przebiegł mniej więcej następująco („mniej więcej” - gdyż nie jest to dokładna kopia procesu instalacji, a jego odtworzenie):

```
Would you like to do a full installation of Condor? [yes]
Are you planning to setup Condor on multiple machines? [yes]
Will all the machines share files via a file server? [yes]

What are the hostnames of the machines you wish to setup?
(Just type the hostnames, not the fully qualified names.
Put one machine per line.  When you are done, just hit enter.)
anthill
ant1
ant2
ant3
ant4
ant5
ant6
ant7
ant8

Have you installed a release directory already? [no]
Where would you like to install the Condor release directory?
[/usr/local/condor] /home/anthill/condor

If something goes wrong with Condor, who should get email about it?
[condor] condor@anthill.ki.agh.edu.pl

What is the full path to a mail program that understands "-s" means
you want to specify a subject? [/bin/mail]

Do all of the machines in your pool from your domain ("house")
share a common filesystem? [no] yes

Do all of the users across all the machines in your domain have a unique
UID (in other words, do they all share a common passwd file)? [no] yes

In some cases, even if you have unique UIDs, you might not have all users
listed in the password file on each machine.
Is this the case at your site? [no]

The Condor binaries and scripts are already installed in:
        /home/anthill/condor/bin
If you want, I can create some soft links from a directory that is already
in the default PATH to point to these binaries, so that Condor users do not
have to change their PATH.  Alternatively, I can leave them where they are
and Condor users will have to add /home/condor/bin
to their PATH or explicity use a full pathname to access the Condor tools.
```

[1] http://www.cs.wisc.edu/condor

```
Shall I create links in some other directory? [yes]

Where should I install these files?
[/usr/local/bin]

What is the full hostname of the central manager?
anthill.ki.agh.edu.pl

You have a "condor" user on this machine.  Is the home directory for
this account (/home/condor) shared among all machines in your pool?
[yes]

Do you want to put all the Condor directories for each machine in
subdirectories of /home/condor/hosts? [yes]

Do you want all the machine-specific config files for each host in one
directory? [yes]
What directory should I use? [/home/anthill/condor/etc]

Should I put in a soft link from /home/condor/condor_config to
/home/condor/etc/condor_config [yes]
```

1.3. Konfiguracja

Po zainstalowaniu zostały nadane odpowiednie prawa plikom wykonywalnym, następnie uruchomione zostały daemony *condor_master* na wszystkich maszynach, rozpoczynając od maszyny głównej tj. *anthilla*.

Pierwszym widocznym pozytywnym skutkiem instalacji Condora od zera było bezproblemowe zadziałanie skryptu *condor_compile*. Widoczny jest spory narzut bibliotek Condora, gdyż przelinkowany tym skryptem program powiększa objętość mniej więcej o 2,5MB.

Niestety już pierwsze próby wysłania zadania poleceniem *condor_submit* zakończyły się radosnym komunikatem:

```
WARNING: File /home/anthill/mrock/plik.out is not writeable by condor.
WARNING: File /home/anthill/mrock/plik.err is not writeable by condor.
```

Co najciekawsze, działo się tak nawet w przypadku, gdy główny daemon miał prawa ustawione następująco:

```
-rwsr-x---    1 root     root       825264 Jul 27  2001 condor_master
```

Komenda *ps* pokazywała zaś:

```
condor   18409  0.0  0.4  2776 1048 ?        S    21:11   0:00 ./condor_master
```

W tym momencie proces konfiguracji na dłuższą chwilę zamarł, gdyż sytuacja wydawała się już beznadziejna. Przełom nastąpił dopiero w momencie „odkrycia” faktu, że katalog `/home/anthill` podmontowany jest z opcją `nosuid`:

```
/dev/md8 on /home/anthill type ext2 (rw,nosuid,nodev,usrquota)
```

Należało więc przenieść wykonywalne pliki systemowe Condora z katalogu `/home/anthill/condor/sbin` do `/usr/local/bin`, a w `/home/anthill/condor/sbin` umieścić linki do właściwych plików.

Po przeniesieniu nadano wszystkim plikom prawa:

```
-rwx------    1 root     root
```

natomiast główny daemon (*condor_master*) otrzymał prawa:

```
-rws------    1 root     root
```

Co ciekawe, po uruchomieniu daemonów komenda *ps* nadal pokazywała:

```
condor   18409  0.0  0.4  2776 1048 ?        S    21:11   0:00 ./condor_master
```

co jak się potem okazało jest wyjaśnione w dokumentacji [2]. Najważniejszy był jednak fakt, że wreszcie można normalnie wysyłać zadania i otrzymywać od nich wyniki.

Podobną operację (tj. przerzucenie plików wykonywalnych do katalogu `/usr/local/bin`) należało wykonać na wszystkich pozostałych maszynach. Przeszkodą w tym był fakt, że *root* nie ma dostępu do katalogów montowanych przez NFS:

```
[root@ant1 bin]# cp /home/anthill/condor/sbin_/condor_* .
cp: cannot open '/home/anthill/condor/sbin_/condor_checkpoint' for reading:
Permission denied
cp: cannot open '/home/anthill/condor/sbin_/condor_collector' for reading:
Permission denied
...
```

Należało więc to zrobić okrężną drogą, kopiując użytkownikiem *condor* pliki do katalogu `/tmp` na danej maszynie, a następnie użytkownikiem *root* z `/tmp` w docelowe miejsce (`/usr/local/bin`).

Dopiero tak skonfigurowany Condor umożliwia wykorzystanie go przez dowolnego użytkownika.

1.3.1. *DAGMan*

Dokumentacja twierdzi, że do wersji 6.2 nie jest dołączony DAGMan, co nie jest do końca prawdą. Po zainstalowaniu rzeczywiście komenda *condor_submit_dag* nie jest dostępna, ale wystarczy w `/usr/local/bin` stworzyć ręcznie linki do plików `condor_submit_dag` i `condor_dagman` znajdujących się w katalogu `/home/anthill/condor/bin`.

2. *POV-Ray*

Do przykładowego przetestowania Condora postanowiłem wykorzystać program *POV-Ray 3.1g*. Po ściągnięciu pliku `povuni_s.tgz` ze strony `http://www.povray.org` wystarczy tylko w jednym miejscu zmodyfikować `makefile` dodając przed komendą dokonującą linkowania polecenie *condor_compile*. Po wydaniu komendy

```
make newunix
```

otrzymujemy dość dużego *POV-Ray'a* (ponad 3MB) którego możemy uruchamiać z pomocą Condora w pełni wykorzystując jego funkcjonalność.

Proces *raytracingu* jest bardzo wdzięczny do zrównoleglenia - można więc pokusić się o wydajniejsze wykorzystanie maszyn. Można by podzielić tworzony obrazek na poziome paski przydzielając każdej z maszyn inny fragment obrazu do przeliczenia. Kiedy wszystkie fragmenty zostaną przeliczone wystarczy jedynie skleić je w całość. Narzuca się tutaj wykorzystanie *DAGMan'a*, który by automatycznie uruchomił program sklejający fragmenty obrazka po zakończeniu ich generowania przez *POV-Ray'a*.

[2] Fragment rozdziału 2.13.4 dokumentacji: *The Unix program ps is not an effective method of determining if Condor is running with root access. When using ps, it may often appear that the daemons are running as the condor user instead of root. However, note that the ps, command shows the current effective owner of the process, not the real owner. [...] In Unix, a process running under the real UID of root may switch its effective UID. [...] For security reasons, the daemons only set the effective uid to root when absolutely necessary (to perform a privileged operation).*

2.1. Wspomaganie zrównoleglenia *raytracingu*

Aby zautomatyzować zadanie zrównoleglenia procesu *raytracingu* przez *POV-Ray'a* napisany został prosty skrypt w Perlu *condor_povray* zarządzający wysyłką zadań do Condora oraz program w C *tga_merge* służący do zlepiania fragmentów obrazu.

2.1.1. Skrypt *condor_povray*

Skrypt ten ma za zadanie stworzyć odpowiednie pliki dla *DAGMan'a* i docelowo ma być jedynym stykiem pomiędzy użytkownikiem a *POV-Ray'em* uruchamianym przez Condora.

Wywoływany jest w sposób następujący:

```
condor_povray n +Iplik.pov +Oplik.tga +Wszer +Hwys +inneopcje
```

gdzie:

`n` - ilość części, na które ma być podzielony obraz;
`plik.pov` - wejściowy plik;
`plik.tga` - docelowy plik wynikowy;
`szer,wys` - szerokość i wysokość docelowego pliku;
`inneopcje` - inne opcje *POV-Ray'a.*

Ponieważ z założenia operujemy tylko na plikach TGA, do opcji *POV-Ray'a* automatycznie dołączana jest opcja `+FT` (zapisywanie TGA).

Skrypt tworzy pliki opisujące zadania, tj. uruchomienie *POV-Ray'a* z różnymi opcjami w zależności od tego, która część obrazu ma być stworzona. Ponadto tworzony jest plik opisujący zadanie sklejenia fragmentów obrazków (za pomocą opisanego niżej programu *tga_merge*). Wreszcie tworzony jest plik konfiguracyjny dla *DAGMan'a* odnoszący się do wszystkich uprzednio stworzonych plików zadań i z tym plikiem jako parametrem uruchamiany jest *DAGMan* poleceniem *condor_submit_dag.*

2.1.2. Program *tga_merge*

Program ten skleja zestaw fragmentów obrazka w jedną całość. Uruchomienie wygląda następująco:

```
tga_merge n +Oplik.tga
```

gdzie:

`n` - ilość części, z których ma być sklejony obraz;
`plik.tga` - docelowy plik wynikowy;

Na podstawie ilości części, z których się składa obrazek, program szuka odpowiednich plików zawierających fragmenty i zlepia je w całość.

Program ten ma bardzo prostą konstrukcję i w zasadzie kompletnie nie analizuje wejściowych plików TGA pod względem tego czy faktycznie są plikami TGA. Jego powstanie było spowodowane problemami ze zlepianiem plików PNG wymuszającym stosowanie biblioteki LIBPNG do odczytu i zapisu plików.

2.2. Przykład

Przykładowo testowaną sceną jest `diffract.pov` dostarczana jako jeden z przykładów wraz z *POV-Ray'em.*

Wysłanie zadań wygląda następująco:

```
$ ./condor_povray 8 +Idiffract.pov +Odiffract.tga +W1024 +H768 +A0.3 +AM2 +R2

Checking your DAG input file and all submit files it references.
This might take a while...
Done.
-----------------------------------------------------------------------
File for submitting this DAG to Condor   : tmp.diffract.tga.povray.dag.condor.sub
Log of DAGMan debugging messages         : tmp.diffract.tga.povray.dag.dagman.out
Log of Condor library debug messages     : tmp.diffract.tga.povray.dag.lib.out
Log of the life of condor_dagman itself  : tmp.diffract.tga.povray.dag.dagman.log

Condor Log file for all jobs of this DAG : tmp.diffract.tga.povray.log
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 272.
-----------------------------------------------------------------------
```

Można sprawdzić, że w kolejce został umieszczony *DAGMan* wraz z odpowiednimi zadaniami:

```
$ condor_q

-- Submitter: anthill.ki.agh.edu.pl : <149.156.100.60:39895> : anthill.ki.agh.edu.pl
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 272.0   mrock           6/23 17:07   0+00:00:03 R  0   0.8  condor_dagman -f -
 273.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 274.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 275.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 276.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 277.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 278.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 279.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr
 280.0   mrock           6/23 17:07   0+00:00:00 I  0   3.4  povray +FT +Odiffr

9 jobs; 8 idle, 1 running, 0 held
```

Po jakimś czasie zaczynają napływać listy od Condora informujące o zakończeniu wykonywania poszczególnych zadań. Poniżej umieszczona jest przykładowa treść takiego listu:

```
This is an automated email from the Condor system
on machine "anthill.ki.agh.edu.pl".  Do not reply.

Your condor job /home/anthill/mrock/condor_povray/povray +FT +Odiffract.tga.000
+SR1 +ER100  +Idiffract.pov +W1024 +H768 +A0.3 +AM2 +R2 exited with status 0.


Time:
        Submitted at:        Sun Jun 23 17:07:28 2002
        Completed at:        Sun Jun 23 17:08:21 2002

        Real Time:           0 00:00:53
        Run Time:            0 00:00:29
        Committed Time:      0 00:00:29

        Remote User Time:    0 00:00:05
        Remote System Time:  0 00:00:00
        Total Remote Time:   0 00:00:05

        Local User Time:     0 00:00:00
        Local System Time:   0 00:00:00
```

```
        Total Local Time:    0 00:00:00

        Virtual Image Size:  3443 Kilobytes

Network:
        3.5 MB read
        333.1 KB written

Buffer Configuration:
        512.0 KB max buffer space per open file
        32.0 KB buffer block size

Total I/O:

        12.6 KB/s effective throughput
        106 files opened
        21 reads totaling 63.5 KB
        212 writes totaling 303.1 KB
        104 seeks

I/O by File:

 /home/anthill/mrock/condor_povray/tmp.diffract.tga.povray.000.err
        opened 1 times
        0 reads totaling 0.0 B
        112 writes totaling 3.0 KB
        0 seeks

 /home/anthill/mrock/condor_povray/diffract.tga.000
        opened 101 times
        0 reads totaling 0.0 B
        100 writes totaling 300.0 KB
        100 seeks

 /home/anthill/mrock/condor_povray/diffract.pov
        opened 1 times
        3 reads totaling 6.5 KB
        0 writes totaling 0.0 B
        1 seeks

 /home/anthill/mrock/condor_povray/woods.inc
        opened 1 times
        7 reads totaling 22.6 KB
        0 writes totaling 0.0 B
        1 seeks

 /home/anthill/mrock/condor_povray/woodmaps.inc
        opened 1 times
        7 reads totaling 22.6 KB
        0 writes totaling 0.0 B
        1 seeks

 /home/anthill/mrock/condor_povray/colors.inc
        opened 1 times
        4 reads totaling 11.8 KB
        0 writes totaling 0.0 B
        1 seeks
```

```
Remote System Calls:
        CONDOR_report_file_info_new      116
        CONDOR_register_syscall_version    1
        CONDOR_get_std_file_info           3
        CONDOR_get_file_info             110
        CONDOR_get_buffer_info             1
        CONDOR_register_opsys              1
        CONDOR_register_arch               1
        CONDOR_lseekwrite                212
        CONDOR_lseekread                   4
        CONDOR_register_fs_domain          1
        CONDOR_register_uid_domain         1
        CONDOR_get_a_out_name              1
        CONDOR_file_info                   1
        CONDOR_get_ckpt_name               1
        CONDOR_get_iwd                     1
        CONDOR_startup_info_request        1
        CONDOR_get_file_stream             1
        CONDOR_reallyexit                  1
        CONDOR_getwd                       1
        CONDOR_chdir                       1
        CONDOR_close                     105
        CONDOR_fstat                     205
        CONDOR_fsync                       2
        CONDOR_lseek                     214
        CONDOR_open                      109


-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Questions about this message or Condor in general?
Email address of the local Condor administrator: condor@anthill.ki.agh.edu.pl
The Official Condor Homepage is http://www.cs.wisc.edu/condor
```

Jak widać zawiera on szczegółowe informacje na temat:

- czasu wykonania zadania;
- ilości danych przesyłanych poprzez sieć;
- ilości danych zapisywanych / odczytywanych z plików wykorzystywanych przez program (pliki wejściowe i wyjściowe *POV-Ray'a*;
- ilości wywołanych zdalnych funkcji systemowych (*Remote System Calls*).

Po zakończeniu wykonywania się wszystkich zadań (czyli również po wykonaniu się procesu zlepiającego fragmenty obrazu) otrzymujemy gotowy plik wynikowy `diffract.tga`. Niepotrzebne są już wszystkie stworzone pliki pomocnicze `tmp*` oraz fragmenty obrazka (pliki `diffract.tga.00*`).

2.3. Przykład z *checkpointingiem*

Aby przetestować mechanizm *checkpointingu* i migracji zadań posłużyłem się również skryptem `condor_povray` tym razem jednak nie dzieląc obrazka na części (aby jedno zadanie było wykonywane maksymalnie długo).

Uruchomienie zadania odbyło się w sposób standardowy, tj.:

```
$ ./condor_povray 1 +Idiffract.pov +Odiffract.tga +W1024 +H768 +A0.3 +AM2 +R2
```

Następnie, po sprawdzeniu za pomocą polecenia *condor_status*, że zadanie wykonuje się na maszynie *ant1* zalogowałem się na tą maszynę powodując tym samym aktywność użytkownika, którą

Condor wykrył i zaprzestał wykonywania zadania. Rzeczywiście, polecenie *condor_status* pokazywało już, że zadanie przeszło w stan *Suspended*.

```
$ condor_status

Name          OpSys       Arch   State      Activity   LoadAv Mem   ActvtyTime

ant1.cluster  LINUX       INTEL  Claimed    Suspended  0.000    60  0+00:00:04
ant2.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+00:00:30
ant3.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+00:07:29
ant4.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+00:08:04
ant5.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+00:09:55
ant6.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+00:10:55
ant7.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+00:07:53
ant8.cluster  LINUX       INTEL  Unclaimed  Idle       0.000    60  0+02:10:04
anthill.ki.ag LINUX       INTEL  Owner      Idle       2.380   249  9+02:45:20

                     Machines Owner Claimed Unclaimed Matched Preempting

         INTEL/LINUX        9     1       1         7       0          0

               Total        9     1       1         7       0          0
```

Po dłuższym czasie utrzymywania się stanu zajętości maszyny przez użytkownika Condor przeprowadził *checkpointing* zadania i zakończył jego wykonywanie na innej maszynie. Fakt dokonania *checkpointingu* zadania zaznaczony jest już w liście otrzymanym od Condora dotyczącym wykonania zadania. W liście tym znalazły się dwie dodatkowe linijki:

```
Checkpoints written: 1
Checkpoint restarts: 1
```

Wędrówkę zadania można zaś prześledzić w logach, które prezentują się następująco:

```
000 (283.000.000) 06/23 17:19:52 Job submitted from host: <149.156.100.60:39895>
...
001 (283.000.000) 06/23 17:19:58 Job executing on host: <192.168.0.11:2589>
...
006 (283.000.000) 06/23 17:41:40 Image size of job updated: 4540
...
004 (283.000.000) 06/23 17:41:42 Job was evicted.
        (1) Job was checkpointed.
                Usr 0 00:00:24, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
        1743055  -  Run Bytes Sent By Job
        3722199  -  Run Bytes Received By Job
...
001 (283.000.000) 06/23 17:44:58 Job executing on host: <192.168.0.12:3464>
...
005 (283.000.000) 06/23 17:52:20 Job terminated.
        (1) Normal termination (return value 0)
                Usr 0 00:02:18, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
                Usr 0 00:02:42, Sys 0 00:00:00  -  Total Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
        1935223  -  Run Bytes Sent By Job
        4847541  -  Run Bytes Received By Job
        3678278  -  Total Bytes Sent By Job
        8569740  -  Total Bytes Received By Job
...
```

Zadanie zostało przeniosione z maszyny *ant1* <192.168.0.11> na maszynę *ant2* <192.168.0.12>. Najwięcej czasu zostało „zmarnowane” przez Condora na oczekiwaniu, czy być może użytkownik łaskawie zwolni zajętą maszynę *ant1*.

Bibliografia

[1] Condor Team, University of Wisconsin-Madison; *Condor® Version 6.3.1 Manual*.
`http://www.cs.wisc.edu/condor/manual/v6.3/index.htm`

[2] Peter Couvares and Todd Tannenbaum; *Condor Tutorial, First EuroGlobus Workshop, June 2001*.
`http://www.cs.wisc.edu/condor/slides/euroglobus-tutorial/euroglobus-tutorial.html`
`http://www.cs.wisc.edu/condor/slides/euroglobus-tutorial/euroglobus-tutorial.ppt`

[3] POV-Team™ ; *POV-Ray™ Version 3.1 User’s Documentation.*

A. Skrypt *condor_povray*

```
#!/usr/bin/perl

use POSIX;

$notify_user = "mrock\@netart.com.pl";

$count = 10;
$width = 640;
$height = 480;
$filename = "test.tga";

$count = $ARGV[0];

$options = "";

for ($i = 1; $i <= $#ARGV; $i++)
{
  $arg = $ARGV[$i];
  $option = substr $arg, 1, 1;
  $value = substr $arg, 2;

  if ($option =~ "W") { $width = int($value); }
  if ($option =~ "H") { $height = int($value); }
  if ($option =~ "O") { $filename = $value; }
  if ($option !~ "O") { $options .= " " . $arg; }
}

$rows = ceil($height/$count);
$rows = 10*int(ceil(0.1*$rows));

$dest_row = 0;

$tmpprefix = "tmp.$filename";

$dagfilename = "$tmpprefix.povray.dag";
open(DAG, "> $dagfilename");

for ($i = 0; $i < $count; $i++)
{
  if ($i == $count - 1)
  {
    $rows = $height - ($count - 1)*$rows;
  }

  $tempfilename = sprintf("%s.%03d", $filename, $i);
  $sr = $dest_row + 1;
  $er = $dest_row + $rows;

  $suffix = sprintf("%03d", $i);

  $cmdfilename = "$tmpprefix.povray.$suffix.cmd";
  open(CMD, "> $cmdfilename");
  print CMD "executable      = povray\n";
  print CMD "output          = $tmpprefix.povray.$suffix.out\n";
  print CMD "error           = $tmpprefix.povray.$suffix.err\n";
  print CMD "log             = $tmpprefix.povray.log\n";
  print CMD "arguments       = +FT +O$tempfilename +SR$sr +ER$er $options\n";
  print CMD "notify_user     = $notify_user\n";
  print CMD "queue\n";
  close CMD;

  print DAG "Job A$suffix $cmdfilename\n";

  $dest_row += $rows;
}

open(CMD, "> $tmpprefix.tga_merge.cmd\n");
print CMD "executable      = tga_merge\n";
print CMD "output          = $tmpprefix.tga_merge.out\n";
print CMD "error           = $tmpprefix.tga_merge.err\n";
```

```
print CMD "log             = $tmpprefix.povray.log\n";
print CMD "arguments       = $count +O$filename\n";
print CMD "notify_user     = $notify_user\n";
print CMD "queue\n";
close CMD;

print DAG "Job B $tmpprefix.tga_merge.cmd\n";

print DAG "PARENT ";

for ($i = 0; $i < $count; $i++)
{
  $suffix = sprintf("%03d", $i);
  print DAG "A$suffix ";
}
print DAG "CHILD B\n";
close DAG;

system("condor_submit_dag $dagfilename");
```

B. Program *tga_merge*

```
/*
 * tga_merge.c
 */

#include <stdio.h>

void main(int argc, char **argv)
{
  char filename[256], infilename[256];
  int count = 10, i, c;
  FILE *outfile, *infile;

  count = atoi(argv[1]);
  for (i = 2; i < argc; i++)
  {
    if (argv[i][1] == 'O')
      strcpy(filename, &argv[i][2]);
  }
  outfile = fopen(filename, "wb");
  for (i = 0; i < count; i++)
  {
    sprintf(infilename, "%s.%03d", filename, i);
    infile = fopen(infilename, "rb");
    if (i > 0)
      fseek(infile, 18, SEEK_SET);
    while ((c = fgetc(infile)) != EOF)
      fputc(c, outfile);
    fclose(infile);
  }
  fclose(outfile);
}
```