Condor@HFI

Lars Oergel

12. November 1999

1 Einleitung

Wer Condor im CIP–Pool benutzen möchte, schickt bitte Kai oder mir eine Mail, um ein persönliches Arbeitsverzeichnis unter `/home/condor` zu erhalten. Alle Daten und Programme, die unter Condor–Kontrolle verarbeitet werden sollen, sollten dann in diesem Verzeichnis abgelegt werden.

Wichtig ist außerdem, daß **mit Einführung von Condor Rechenjobs im CIP–Pool nur noch über Condor zu starten sind!!** Da Condor zusätzliche manuell gestartete Jobs nicht „sehen" kann, wäre sofort wieder die Überlastung eines Rechners bzw. einer (Swap-)Platte möglich.

2 Arbeiten mit Condor

Um einen Rechenjob unter Condor–Kontrolle laufen zu lassen sind folgende Schritte durchzuführen:

1. Job–Vorbereitung

   Grundsätzlich ist jeder Job condor–fähig, der im Hintergrund laufen kann. Dazu benutzen wir das sog. *Vanilla–Universe* von Condor. Dabei muß die Netzwerkkommunikation leider über NFS abgewickelt werden. Sollte sich dieses als zu große Netzwerkbelastung herausstellen, haben wir noch die Möglichkeit, das *Standard–Universe* zu benutzen. Die Kommunikation würde dann über effiziente condor–eigene Mechanismen erfolgen, allerdings müßten die Jobs dafür mit einer speziellen Condor–Bibliothek gebunden werden.

2. Absetzen des Jobs

   Zum Absetzen des Jobs an Condor muß ein sog. *Condor–Submit–File* geschrieben werden, das beschreibt, welches Programm in welchem Verzeichnis mit welchen Parametern gestartet werden soll. Außerdem ist zu deklarieren, ob der Job ein „Langläufer" oder ein „Kurzläufer" ist. Der Grenzwert ist zunächst auf 5 Stunden CPU–Zeit eingestellt.

   Momentan ist cip5004 zum Absetzen der Jobs vorgesehen. cip5005 und cip5006 sind für Kurzläufer–Jobs reserviert, der Rest (cip5007 bis cip5015) für Langläufer.

   Condor verwaltet eine Warteschlange für die Rechenjobs, die in chronologischer Reihenfolge des Eingangs der Aufträge abgearbeitet wird.

3. Job–Ablauf unter Condor

   Sobald ein Rechner unbeschäftigt ist (d.h. kein Condor–Prozeß dort läuft) und sich ein seiner Job–Klasse entsprechender Auftrag in der Warteschlange befindet, wird der Rechner von Condor unter Beschlag genommen und der Job dort zur Ausführung gebracht.

4. Beendigung

   Wenige Minuten nach Beendigung eines Jobs (auf „natürlichem" Wege oder nach dem automatischen „Abschießen" eines zu lang gelaufenen Kurzläufer–Jobs) wird der Rechner von Condor als wieder verfügbar erkannt und der nächste anstehende Job dort gestartet. Der Benutzer erhält eine E–Mail mit der Fertigmeldung des Jobs, die u.a. den Exit–Status, die Start- und Endzeit und die CPU–Zeit enthält.

3 Beispiel

Unter `/home/condor/oergel` liegt eine umfassende Condor–Dokumentation in der Datei `condor-V6_0-Manual.ps.gz` und folgendes Beispiel bereit:

`myjob.cc`[1] beinhaltet einen „kleinen" Beispiel–Rechenjob, dessen Speicher- und CPU–Zeit–Bedarf über Konstanten eingestellt werden kann. Das fertige Binary (`make` reicht zum Übersetzen) benötigt zwei Parameter zum Starten: Beide geben Dateinamen an, wobei versucht wird, die erste Datei zum Lesen zu öffnen und ihren Inhalt in die zweite Datei zu schreiben. Außerdem wird die Standardeingabe eingelesen und auf die Standardausgabe geechot.

Zum Starten des Jobs via Condor wird das `submit_file` auf der folgenden Seite benutzt:

- Zunächst muß mit `Executable` der Name des Binaries angegeben werden, relativ zum `Initialdir`.
- Danach erfolgt mit `JobLength` die Klassifizierung des Jobs.
- Anschließend kann mit `Input`, `Output` und `Error` angegeben werden, in welche Dateien die Standardein-, ausgabe und der Fehlerkanal umgeleitet werden soll.
- Kommandozeilenargumente können hinter `Arguments` angegeben werden.
- Damit die E–Mail–Benachrichtigung klappt, muß mit `Notify_user` die E–Mail–Adresse in der Form `<login>@localhost` hinterlegt werden (`Notification = ALWAYS` nicht vergessen).
- Optional kann mit `Log` ein Logfile angegeben werden, in dem Condor dann ein wenig Protokoll führt.
- Zum Schluß muß noch das Vanilla–Universe aktiviert werden, bevor der Job mittels `Queue` in die Condor–Warteschlange eingereiht werden kann.

[1] s. Anhang A

```
# Beispiel-Submit-File
#
# von Lars Oergel
# 12.11.1999

###################
# Pfad zum Binary #
###################
Executable = myjob

###############
# Verzeichnis #
###############
Initialdir = /home/condor/oergel

#######################
# Job-Klassifizierung #
#######################
#+JobLength = "Short"
+JobLength = "Long"

###################################
# Ein-/Ausgabe-/Fehlerumleitungen #
###################################
# Default: /dev/null
Input  = stdin
Output = stdout
Error  = stderr

###########################
# Kommandozeilen-Parameter #
###########################
Arguments = intest outtest

####################
# Benachrichtigung #
####################
# <login>@localhost (ohne localhost gehts nicht!)
Notify_user  = oergel@localhost
Notification = ALWAYS

###########
# Logfile #
###########
Log = log

#############
# Universum #
#############
Universe = vanilla

############
# Und los! #
############
Queue
```


Mittels

```
cip5004:/home/condor/oergel> condor_submit submit_file

Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 39.
```

kann dann der so beschriebene Job in die Condor–Warteschlange gestellt werden.

Diese Queue kann man sich mittels `condor_q` auch jederzeit anzeigen lassen:

```
cip5004:/home/condor/oergel> condor_q

-- Submitter: cip5004.cfd : <192.168.2.4:31280> : cip5004.cfd
 ID     OWNER      SUBMITTED   CPU_USAGE ST PRI SIZE CMD
  39.0  oergel   11/12 23:28  0+00:00:00 R  0   0.0  myjob intest outtest

1 jobs; 0 unexpanded, 0 idle, 1 running, 0 malformed
```

Man sieht, daß sich ein Job mit der ID 39 in der Warteschlange befindet. Dieser gehört dem Benutzer `oergel` und befindet sich im `ST`atus `R`unning. `U`nexpanded heißt, daß sich der Job noch gedulden muß, bis ein Rechner frei wird. Sollte dieser Status eintreten, obwohl sich ein Rechner im Zustand `Unclaimed/Idle` befindet, ist der Job evtl. falsch submittet worden, d.h. man sollte sich sein Submit–File nochmals genau anschauen!

Will man einen Job aus der Queue entfernen, kommt der Befehl `condor_rm` zum Einsatz, dem einfach nur die Job–ID als Parameter zu übergeben ist.

Kommt ein Job zur Ausführung und will man wissen, auf welchem Rechner er gelandet ist, so erfährt man dieses folgendermaßen:

```
cip5004:/home/condor/oergel> condor_status -claimed

Name         Arch    OpSys     LoadAv  RemoteUser     ClientMachine
cip5011.cf   INTEL   LINUX     1.650   oergel@cfd     cip5004.cfd

                     Machines        MIPS      KFLOPS   AvgLoadAvg
         INTEL/LINUX        1         152        7245   1.650
               Total        1         152        7245   1.650
```

Der Job läuft also auf dem Rechner cip5011 und wurde von cip5004 aus submittet.

Ebenso läßt sich auch eine Gesamtansicht des Condor–Pools anzeigen:

```
cip5004:/home/condor/oergel> condor_status

Name       Arch   OpSys  State      Activity  LoadAv Mem  ActvtyTime

cip5005.cf INTEL  LINUX  Unclaimed  Idle      1.100  63   0+01:23:39
cip5006.cf INTEL  LINUX  Unclaimed  Idle      1.000  63   0+01:23:43
cip5007.cf INTEL  LINUX  Unclaimed  Idle      1.390  63   0+00:00:-3
cip5008.cf INTEL  LINUX  Unclaimed  Idle      1.400  63   0+00:00:-6
cip5009.cf INTEL  LINUX  Unclaimed  Idle      1.200  63   0+00:00:-8
cip5010.cf INTEL  LINUX  Unclaimed  Idle      1.240  63   0+00:00:-5
cip5011.cf INTEL  LINUX  Owner      Idle      1.550  63   0+00:00:-6
cip5012.cf INTEL  LINUX  Claimed    Busy      1.510  63   0+00:00:05
cip5013.cf INTEL  LINUX  Owner      Idle      1.580  63   0+00:00:-5
cip5014.cf INTEL  LINUX  Unclaimed  Idle      1.370  63   0+00:00:-7
cip5015.cf INTEL  LINUX  Unclaimed  Idle      1.260  63   0+00:00:-6

                 Machines Owner Claimed Unclaimed Matched Preempting

     INTEL/LINUX       11     2       1         8       0          0
           Total       11     2       1         8       0          0
```

Einträge in der Spalte `State` können dabei drei verschiedene Werte aufweisen:

- `Unclaimed`

  besagt, daß Condor diesen Rechner zur Zeit nicht benutzt.

- `Claimed`

  meint das Gegenteil: Dieser Rechner ist zur Zeit für Condor aktiv.

- `Owner`

  ist bei uns nur ein Zwischenzustand, der minutenweise beim Übergang von `Claimed` nach `Unclaimed` auftritt.

Ich hoffe, daß diese Informationen ausreichen, um Jobs unter Condor–Kontrolle im CIP–Pool laufen lassen zu können. Sollten Fragen oder Probleme auftauchen bitte eine E–Mail an Kai oder mich schicken!

A `myjob.cc`

```
// Testjob fuer Condor
//
// von Lars Oergel
// 12.11.1999
//
// Parameter: <infile> <outfile>
//
// Das Programm oeffnet infile und schreibt den Inhalt in outfile.
// Anschliessend wird die Standardeingabe auf die Standardausgabe
// geechot, danach wird eine einstellbare Speichermenge belegt und
// eine bestimmte CPU-Zeit verbraucht.

const int memory  = 20; // in Megabyte
const int minuten = 3; // bemessen an einem P200

#include <fstream.h>

void main (int argc, char *argv[])
{
    if(argc != 3)
    {
        cerr<<argv[0]<<": Usage: "<<argv[0]<<" <infile> <outfile>\n";
        exit(1);
    }

// Dateien oeffnen und Inhalt von infile in outfile schreiben
    char b;
    ifstream infile (argv[1]); ofstream outfile (argv[2]);
    for(infile.get(b); infile.good(); infile.get(b))
        outfile.put(b);
    infile.close(); outfile.close();

// Standardeingabe auf Standardausgabe echoloten :-)
    for(cin.get(b); cin.good(); cin.get(b))
        cout.put(b);

// Speicher anfordern und belegen
    for(long x=0; x<memory; x++)
    {
        char *p=new char[1024*1024];
        for(long i=0; i<1024*1024; i++)
            *(p+i)='\0';
    }

// CPU-Zeit mampfen
    for(long x=0; x<10*minuten; x++)
        for (long y=0; y<200000000; y++)
            ;

    return 0;
}
```