Das Morfeus-Grid

Eine kurze Einführung für Benutzerinnen und Benutzer in das HTCondor-Batchsystem

Thomas Bauer / Thomas Wiesehöfer
IVV Naturwissenschaften
Universität Münster
Juni 2016

https://www.uni-muenster.de/NWZ/Angebot/ScientificComputing/Morfeus

https://www.uni-muenster.de/NWZ

https://research.cs.wisc.edu/htcondor/

Inhaltsverzeichnis

1. Vorwort zur aktuellen Ausgabe

Im Rahmen der Umstellung vom NWZnet auf die neue Domäne NWZ haben wir das Morfeus-Grid in einigen Punkten erweitert und die eingesetzte HTCondor-Software auf den neuesten Stand gebracht. Dadurch ergeben sich auch für die Benutzerinnen und Benutzer einige Änderungen.

Eine Neuentwicklung ist die grafische Oberfläche MorfeusConsole, mit deren Hilfe man auch ohne Benutzung der Kommandozeile seine Jobs abschicken und überwachen kann. Das Programm kann auch genutzt werden, um sich ein Grundgerüst für die Jobbeschreibungsdatei „zusammenzuklicken“, das man dann ggf. noch manuell anpassen kann.

Eine weitere wichtige Neuerung betrifft die Linux-Nutzer. Der weitaus größte Teil der Rechner im NWZ läuft unter Windows, so dass Linux-Nutzer ihre Programme bislang nur auf einem Bruchteil der zur Verfügung stehenden Rechner ausführen konnten. Für das NWZ haben wir eine Lösung entwickelt, die es ermöglicht, mit minimalen Änderungen an der Jobbeschreibung Linux-Programme auch auf die Windows-Rechner zu schicken. Dabei wird auf den Windows-Rechenknoten bei Bedarf automatisch eine virtuelle Maschine gestartet, auf der das Linux-Programm dann ausgeführt werden kann.

Bisher gab es im Morfeus-Grid die Möglichkeit, spezielle Prozessortypen auszuwählen (beispielsweise „i7“), um so Programme, die für bestimmte Prozessoren optimiert sind, auch an Geräte mit solchen CPUs schicken zu können. Diese Lösung stieß jedoch an ihre Grenzen, da es inzwischen z.B. vom Core i7 sechs verschiedene Generationen mit unterschiedlicher Unterstützung für erweiterte Befehlssätze gibt. Daher wurde dieses System ersetzt durch eines, bei dem direkt die Verfügbarkeit bestimmter Befehlssätze wie SSE4.1 oder AVX geprüft werden kann.

Auch die neue Version von HTCondor bringt einige Änderungen mit sich, z.B. wird üblicherweise nicht mehr zwischen den verschiedenen Windows-Versionen unterschieden (OpSys ist „WINDOWS“ statt „WINNT60“, „WINNT61“ usw.). Die entsprechenden Kapitel wurden angepasst.

Schließlich wurde noch ein Kapitel ergänzt, das anhand eines Beispiels beschreibt, wie man auch Programme, die sehr lange rechnen, im Morfeus-Grid erfolgreich einsetzen kann.

Münster, im Mai 2016
Thomas Wiesehöfer

Juni 2016: Da jetzt auch parallele Programme im Morfeus-Grid ausgeführt werden können, findet sich auch zu diesem Thema ein neues Kapitel in der Anleitung.

2. Einleitung

- Haben Sie ein Programm, das eine Berechnung durchführt, die möglicherweise einige Stunden dauert?
- Eventuell wollen Sie dieses Programm auch noch mehrmals mit verschiedenen Parametern ausführen?
- Damit ist stundenlange Wartezeit vorprogrammiert!
- Gleichzeitig stehen fast überall Computer, die zeitweise nicht benutzt werden. Besonders nachts geht somit wertvolle Rechenzeit verloren.

Es ist das Ziel des Projektes MORFEUS[1], diese ungenutzte Rechenleistung in der IVV Naturwissenschaften für deren Mitglieder nutzbar zu machen. Besonders Benutzerinnen und Benutzer von Linux und Windows sind angesprochen.

Das Batchsystem HTCONDOR (http://research.cs.wisc.edu/htcondor/) ist ein intelligentes Queueingsystem. Es ist für Umgebungen entwickelt, in denen Computer nicht 24 Stunden am Tag benutzt werden. HTCONDOR eignet sich daher ideal für MORFEUS.

Die Funktionsweise sei an dieser Stelle kurz erläutert. Ein Computer agiert als zentraler Manager. Er verwaltet den Computer-Pool und entscheidet, wann und wo ein Job gestartet wird. Wann ein Job gestartet wird, hängt davon ab, wie hoch die Priorität des Benutzers oder der Benutzerin ist. Wo ein Job gestartet wird, hängt davon ab, ob der Computer gerade vor Ort benutzt wird. Sitzt beispielsweise ein Student im ComputerLab (ehemals CIP-Pool) an einem Computer, so wird auf diesem Computer kein Job gestartet. Falls eine Benutzerin oder ein Benutzer sich auf einem Computer einloggt, auf dem gerade ein Job läuft, so wird der Job sofort angehalten. Wird der Computer wieder frei, setzt HTCONDOR den Job auf diesem Computer fort, und zwar an der Stelle, an der er unterbrochen wurde.

Falls der Computer, von dem aus Sie den Job losgeschickt haben, oder der Computer, auf dem der Job gerechnet wird, neugestartet werden, kann HTCONDOR den Job nicht anhalten und später weiterrechnen. Stattdessen wird der Job abgebrochen und sofort auf einem anderen freien Rechner neugestartet. Es gibt also theoretisch unbegrenzte Rechenzeit, praktisch ist die Rechenzeit jedoch begrenzt, da die Computer in NWZ gelegentlich automatisch aufgrund von Updates neugestartet werden müssen. Wie Sie dieses Problem umgehen können, ist in Abschnitt 4.10 beschrieben.

[1] MORFEUS steht für Multiple Orphaned Resources for Educational Use.

3. Benutzung von HTCondor

Dieses Kapitel soll die grundlegenden Funktionen des HTCondor-Systems erläutern.

HTCondor umfasst keine graphische Benutzeroberfläche, daher werden üblicherweise alle Befehle über die Kommandozeile (Eingabeaufforderung bzw. Shell) eingegeben. Für das Morfeus-Grid der IVV Naturwissenschaften wurde allerdings eine solche Oberfläche, die *MorfeusConsole*, entwickelt. Sie können daher die im Folgenden erläuterten Funktionen entweder über die Kommandozeile oder über die graphische Oberfläche steuern.[2] Sie können dabei auch jederzeit wechseln. Es ist kein Problem, einen Job über die Kommandozeile abzuschicken und über die MorfeusConsole zu überwachen oder umgekehrt.

Abbildung 3.1.: Die Eingabeaufforderung (Windows).

3.1. Voraussetzung

Zunächst benötigen Sie einen Computer, auf dem die HTCondor-Software installiert ist. Falls Sie keinen solchen Computer zur Verfügung haben, können Sie einen Terminalserver der IVV benutzen.
(`https://www.uni-muenster.de/NWZ/Angebot/NWZatHome/Terminalserver/`)
Die Benutzung von HTCondor auf dem Terminalserver ist fast identisch mit der Benutzung auf einem Computer, auf dem Sie sich vor Ort einloggen. Beachten Sie jedoch die Hinweise im Abschnitt 4.6.

3.2. Anmeldung (`condor_store_cred`)

Haben Sie sich für einen Windows-Computer entschieden, von dem Sie Ihre Jobs abschicken wollen, so müssen Sie sich einmalig anmelden und Ihr NWZ-Kennwort für HT-

[2] Momentan ist die MorfeusConsole ausschließlich unter Windows installiert. Unter Linux sind Sie auf die Kommandozeile angewiesen.

Condor hinterlegen. Dies ist nötig, damit HTCondor Ausgabedateien in Ihre Verzeichnisse schreiben kann. Unter Linux ist dieser Schritt nicht erforderlich.

Die MorfeusConsole erkennt automatisch, wenn Sie HTCondor Ihr aktuelles Passwort noch nicht mitgeteilt haben, und wird Sie bei Bedarf danach fragen. Sie brauchen daher keinen gesonderten Anmeldeschritt durchzuführen.

Wenn Sie mit der Kommandozeile arbeiten, führen Sie den folgenden Befehl aus:

```
condor_store_cred add
```

HTCondor fordert Sie nun auf, Ihr Passwort zu hinterlegen. Nach dieser einmaligen Anmeldung können Sie HTCondor nun vollständig nutzen.

WICHTIG:

- Diese Anmeldung müssen Sie auf jedem Windows-Computer durchführen, von dem aus Sie einen Job abschicken wollen.
- Falls Sie Ihr NWZ-Passwort ändern, so müssen sie auch das HTCondor-Passwort ändern. Führen Sie dazu

  ```
  condor_store_cred delete
  ```

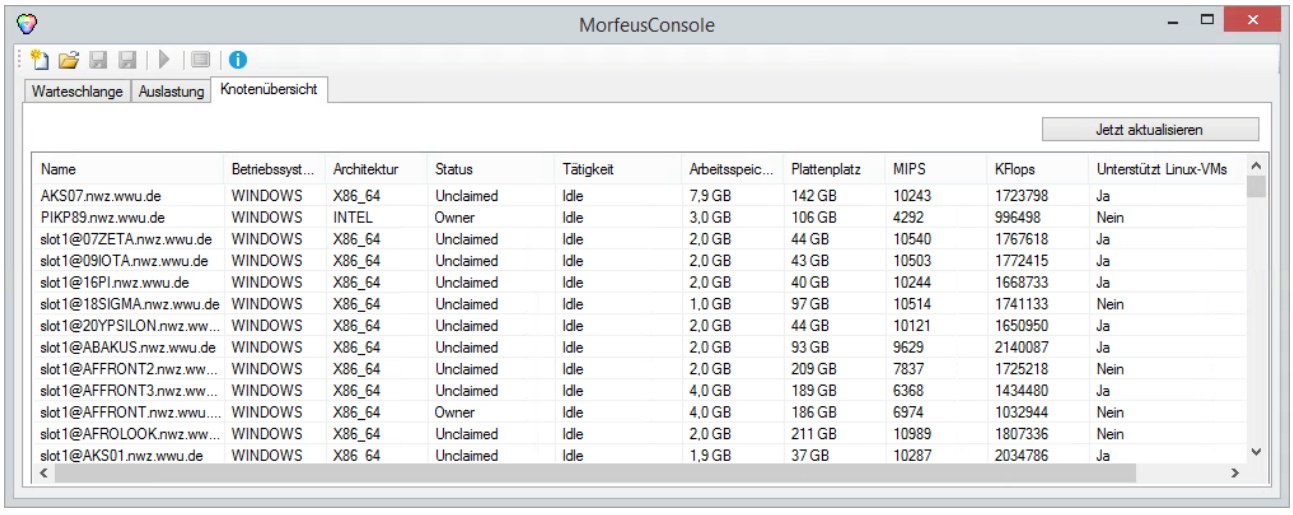
  und danach wieder

  ```
  condor_store_cred add
  ```

  aus.

3.3. Aktueller Status des Pools (`condor_status`)

Für eine Übersicht über alle Knoten des Pools wählen Sie den entsprechenden Tab der MorfeusConsole aus:



Name	Betriebssyst...	Architektur	Status	Tätigkeit	Arbeitsspeic...	Plattenplatz	MIPS	KFlops	Unterstützt Linux-VMs
AKS07.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	7,9 GB	142 GB	10243	1723798	Ja
PIKP89.nwz.wwu.de	WINDOWS	INTEL	Owner	Idle	3,0 GB	106 GB	4292	996498	Nein
slot1@07ZETA.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	44 GB	10540	1767618	Ja
slot1@09IOTA.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	43 GB	10503	1772415	Ja
slot1@16PI.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	40 GB	10244	1668733	Ja
slot1@18SIGMA.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	1,0 GB	97 GB	10514	1741133	Nein
slot1@20YPSILON.nwz.ww...	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	44 GB	10121	1650950	Ja
slot1@ABAKUS.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	93 GB	9629	2140087	Ja
slot1@AFFRONT2.nwz.ww...	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	209 GB	7837	1725218	Nein
slot1@AFFRONT3.nwz.ww...	WINDOWS	X86_64	Unclaimed	Idle	4,0 GB	189 GB	6368	1434480	Ja
slot1@AFFRONT.nwz.wwu....	WINDOWS	X86_64	Owner	Idle	4,0 GB	186 GB	6974	1032944	Nein
slot1@AFROLOOK.nwz.ww...	WINDOWS	X86_64	Unclaimed	Idle	2,0 GB	211 GB	10989	1807336	Nein
slot1@AKS01.nwz.wwu.de	WINDOWS	X86_64	Unclaimed	Idle	1,9 GB	37 GB	10287	2034786	Ja

In der Kommandozeile führen Sie stattdessen den Befehl

```
condor_status
```

aus. Sie erhalten eine Liste wie in Tabelle 3.1 abgebildet. Jeder Knoten verfügt minde-

Name	OpSys	Arch	State	Activity	Mem
slot1@PFT23.nwz	LINUX	X86_64	Owner	Idle	0
slot1_1@PFT23.n	LINUX	X86_64	Claimed	Suspended	4096
slot1@PCTP04.nw	WINDOWS	INTEL	Unclaimed	Idle	4096
slot1@PCPI05.nw	WINDOWS	X86_64	Owner	Idle	4096
slot1@PCFT15.nw	WINDOWS	X86_64	Unclaimed	Idle	5120
slot1_1@PCFT15.	WINDOWS	X86_64	Claimed	Busy	1024
slot1_2@PCFT15.	WINDOWS	X86_64	Claimed	Busy	2048

Tabelle 3.1.: (Gekürzte) Ausgabe von `condor_status`.

stens über den Slot `slot1@computername`, der anfangs alle Ressourcen (RAM, CPUs, Festplattenspeicher) des Knotens enthält. Soll ein Job auf dem Knoten ausgeführt werden, werden zuerst die vom Job angeforderten Ressourcen in einen Unterslot (benannt `slot1_1@computername`, `slot1_2@computername` usw.) verschoben. In diesem Unterslot wird dann der Job ausgeführt. Solange noch genügend freie Ressourcen vorhanden sind, werden bei Bedarf weitere Unterslots angelegt, in denen dann gleichzeitig andere Jobs ausgeführt werden.

In der zweiten Spalte finden Sie das Betriebssystem, das auf dem Knoten läuft. Es folgt die Architektur des Prozessors. Eine komplette Übersicht der im MORFEUS-Grids z. Zt. vorhandenen Betriebssysteme und Prozessorarchitekturen finden Sie in Abschnitt 4.2. In der Spalte „Status“ wird die momentane Verfügbarkeit des Computers angezeigt. `Unclaimed` bedeutet, dass der Rechner gerade frei ist und ein Job durch HTCONDOR auf dem Computer gestartet werden könnte. `Owner` heißt, dass der Computer gerade vor Ort benutzt wird. Der Status `Claimed` bedeutet, dass HTCONDOR bereits einen Job auf dem Computer gestartet hat.

An der Spalte „Tätigkeit“ bzw. `Activity` lässt sich erkennen, ob der Prozessor gerade ausgelastet ist. `Idle` bedeutet, dass der Prozessor gerade nicht besonders beschäftigt wird. `Busy` heißt, dass momentan ein Job durch HTCONDOR auf dem Computer ausgeführt wird. `Suspended` bedeutet, dass zwar ein Job auf dem Computer gestartet wurde, der Job aber zurückgestellt ist, weil der Computer gerade vor Ort benutzt wird.
Weiterhin erhalten Sie Informationen über den installierten Arbeitsspeicher, den verfügbaren Plattenplatz und Informationen über die Geschwindigkeit des jeweiligen Rechners – MIPS (Million Instructions Per Second) und KFLOPS (Kilo Floatingpoint Operations Per Second). Ein höherer Zahlenwert entspricht dabei einem leistungsstärkeren Prozessor.

Wie Sie in Tabelle 3.1 sehen können, fehlen auf der Kommandozeile aus Platzgründen einige Spalten, die in der MorfeusConsole angezeigt werden. Sie können diese aber auch über die Kommandozeile abrufen. Dazu führen Sie das Kommando

```
condor_status -server
```

aus. Sie erhalten, wie in Tabelle 3.2 abgebildet, nun die vorher fehlenden Informationen für jeden Computer.

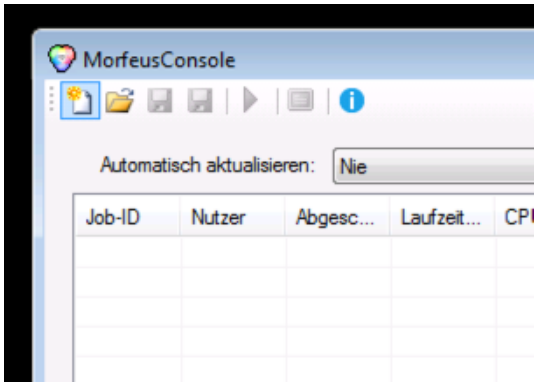
Wie Sie eine knappe Übersicht über die gesamte Auslastung des MORFEUS-Grids anzeigen lassen können, ist in Kapitel 4.1 beschrieben.

Name	OpSys	Arch	Memory	Disk	Mips	KFlops
PFT23.nwz.wwu	LINUX	X86_64	1024	1577756	2555	831774
PCTP04.nwz.ww	WINDOWS	INTEL	512	16071392	2665	826138
PCTP06.nwz.ww	WINDOWS	X86_64	512	15503396	2167	829924
PCPI05.nwz.ww	WINDOWS	X86_64	512	14984136	2623	831124

Tabelle 3.2.: Ausgabe von `condor_status -server`.

3.4. Jobs abschicken (`condor_submit`)

Um einen Job abschicken zu können, müssen Sie zunächst eine Jobbeschreibung anlegen. In der MorfeusConsole funktioniert das über den entsprechenden Befehl in der Symbolleiste:



Wenn Sie mit der Kommandozeile arbeiten möchten, erstellen Sie die Jobbeschreibungsdatei mit einem Texteditor Ihrer Wahl.

Eine Jobbeschreibung enthält alle wichtigen Informationen, die HTCondor benötigt, um den Job ordnungsgemäß verarbeiten zu können. Im Folgenden werden die einzelnen Zeilen einer per Hand angelegten Jobbeschreibungsdatei erläutert. Jede dieser Optionen können Sie auch in der MorfeusConsole konfigurieren.

Die einfachste mögliche Jobbeschreibung enthält nur drei Zeilen:

```
universe = vanilla
executable = job.exe
queue
```

Die erste Zeile wählt dabei das „Vanilla-Universum“. HTCondor unterstützt prinzipiell noch weitere „Universen“, aber im Morfeus-Grid wird lediglich das Vanilla-Universum unterstützt. (In der MorfeusConsole brauchen Sie daher hier nichts zu konfigurieren, das Vanilla-Universum wird automatisch gewählt.)

Weiterhin muss jede Jobbeschreibung festlegen, welches Programm überhaupt ausgeführt werden soll. Das passiert mit der zweiten Zeile. Im Beispiel würde also das Programm `job.exe` abgeschickt.

Die Jobbeschreibungsdatei wird immer mit folgendem Befehl beendet (d.h. auch hier brauchen Sie in der MorfeusConsole nichts einzustellen):

```
queue
```

Alle auf diese Zeile folgenden Einträge werden ignoriert![3]

Neben diesen Pflichteinträgen gibt es noch eine Fülle von weiteren nützlichen Einträgen, von denen an dieser Stelle nur die wichtigsten aufgeführt werden sollen, die Sie alle auch über die graphische Oberfläche konfigurieren können. Weiterführende Informationsquellen sind in Kapitel 6 zu finden.

`log = Logdatei.log` In der angegebenen Datei werden nützliche Informationen über den Programmverlauf angezeigt. Eine Log-Datei sollte immer angelegt werden, da man sonst dem Verlauf des Jobs kaum folgen kann. Eine Erläuterung der Log-Datei erfolgt in Abschnitt 3.6.

`input = Eingabe.in` Haben Sie ein Programm, das Eingabe per Tastatur erfordert („Standardeingabe“), so müssen Sie diese Eingabe schon vorher in einer input-Datei speichern.

`output = Ausgabe.out` In der angegebenen Datei wird die gesamte Ausgabe, die normalerweise auf dem Bilschirm erscheinen würde („Standardausgabe“), abgespeichert.

`error = Fehlermeldungen.err` In diese Datei werden alle Fehlermeldungen, die das Programm ausgibt („Standardfehlerausgabe“), geschrieben.

`transfer_input_files = Datei1,Datei2,Datei3` Falls das Programm zusätzliche Dateien benötigt, müssen diese hier aufgelistet werden.

`request_cpus =` Mit diesem Eintrag geben Sie an, wie viele CPU-Kerne für Ihren Job reserviert werden sollen. Wenn Sie diese Option nicht angeben, wird Ihr Job auf einem einzelnen CPU-Kern ausgeführt. Bei seriellen Jobs können Sie diesen Eintrag also weglassen. Weitere Informationen zu parallelen Jobs finden Sie in Kapitel 4.9.

`request_disk =` Mit diesem Eintrag geben Sie an, wie viel Festplattenspeicher in KB Ihr Job benötigt. (Achtung: Das ist anders als bei `request_memory`, wo die Angabe in MB erfolgt.) So können Sie verhindern, dass der Job an einen Knoten mit weniger Platz geschickt wird und die Berechnung fehlschlägt.

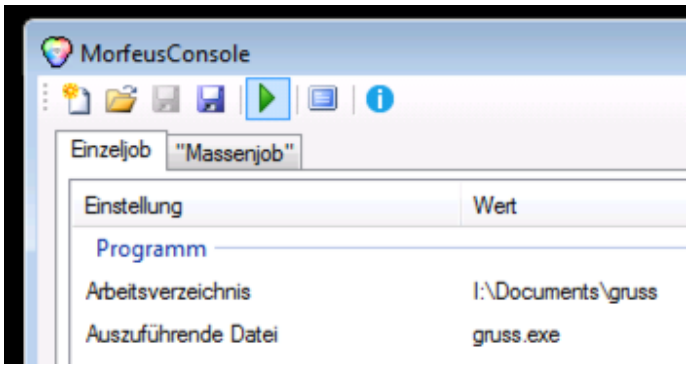
`request_memory =` Mit diesem Eintrag geben Sie an, wie viel Arbeitsspeicher in MB Ihr Job benötigt. (Achtung: Das ist anders als bei `request_disk`, wo die Angabe in KB erfolgt.)

`requirements =` An dieser Stelle können weitere benötigte Ressourcen festgelegt werden. Legen Sie nichts fest, wird der Computer, von dem aus der Job abgeschickt wird, als Referenz genommen. Die genaue Benutzung dieses Eintrages entnehmen Sie bitte dem Abschnitt 3.5.

Alle diese optionalen Einträge müssen vor `queue` stehen, sonst werden sie ignoriert!
Die Dateinamen und Endungen können frei gewählt werden.

Wenn Ihre Jobbeschreibung fertig ist, können Sie den Job abschicken. In der Morfeus-Console klicken Sie den entsprechenden Knopf in der Symbolleiste an:

[3] Es sei denn, es folgt danach noch ein weiterer queue-Befehl, der einen zusätzlichen Job abschickt. Mehr Informationen dazu finden Sie im offiziellen Handbuch zu HTCondor.



In der Kommandozeile nutzen Sie stattdessen den Befehl

`condor_submit` *`job.sub`*

wobei `job.sub` der Name der von Ihnen mit dem Editor erzeugten Jobbeschreibungsdatei ist.

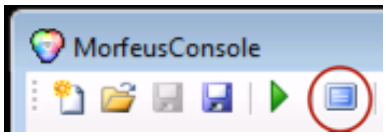
Sobald der Job beendet ist, werden alle Ergebnisse wieder in das Verzeichnis zurückgeschrieben, in dem sich die Jobbeschreibungsdatei befindet.

Nachdem der Job abgeschickt wurde, sollten Sie regelmäßig die Log-Datei (Erläuterungen dazu in Abschnitt 3.6) dahingehend kontrollieren, ob sich der Job wie gewünscht verhält.

An dieser Stelle sei darauf hingewiesen, dass Sie bei Jobs, die länger als 24 Stunden rechnen, unbedingt die Hinweise in Abschnitt 4.10 beachten sollten.

3.4.1. Erstellung einer Jobbeschreibung für die Kommandozeile mit der MorfeusConsole

Wenn Sie in der MorfeusConsole eine Jobbeschreibung erstellt haben, aus dieser aber eine Jobbeschreibung für die Kommandozeile generieren möchten, z.B. um komplexere Einstellungen vornehmen zu können, finden Sie die entsprechende Funktion in der Symbolleiste:



Auf diese Weise können Sie auch jederzeit schauen, wie bestimmte Einstellungen „per Hand“ vorgenommen würden.

3.5. Benötigte Ressourcen angeben (`requirements`)

Es ist nicht zwingend nötig, HTCondor mitzuteilen, welche Ressourcen Sie genau benötigen. Geben Sie nichts an, so nimmt HTCondor automatisch an, dass Ihr Job das Betriebssystem und die Prozessorarchitektur benötigt, von denen aus Sie den Job abschicken. Damit schränken Sie allerdings die in Frage kommenden Rechenknoten häufig zu sehr ein. Wenn Sie z.B. auf einer 32-Bit-Maschine unter Windows ein Programm abschicken, wird HTCondor es ohne weitere Angaben Ihrerseits nie auf einen 64-Bit-Rechner schicken, obwohl diese auch 32-Bit-Programme ausführen können. Umgekehrt würde ein

über eine 64-Bit-Maschine abgeschicktes 32-Bit-Programm nur auf 64-Bit-Rechnern ausgeführt, obwohl es eigentlich auch auf 32-Bit-Rechnern liefe.

In der MorfeusConsole lassen sich die wichtigsten Anforderungen einstellen. Wenn Sie komplexere Anforderungen an den Rechner haben, der Ihren Job ausführen soll, müssen Sie mit der Kommandozeile arbeiten. Sie können aber, wie in Abschnitt 3.4.1 beschrieben, alle grundlegenden Einstellungen in der MorfeusConsole vornehmen und sich dann eine Jobbeschreibung für die Kommandozeile generieren lassen, in der Sie dann nur noch die `requirements`-Zeile anpassen müssen. Wenn Sie eine Einstellungsmöglichkeit in der MorfeusConsole vermissen, schreiben Sie uns doch eine E-Mail, vielleicht können wir die Funktion ja auch kurzfristig ergänzen.

Um in einer Jobbeschreibung für die Kommandozeile eine oder mehrere bestimmte Anforderungen an den Computer festzulegen, der Ihren Job ausführen soll, müssen Sie in der Jobbeschreibungsdatei den Eintrag `requirements` verwenden. Jeder Computer hat sehr viele Attribute (diese nennt man in HTCondor *ClassAds*), die Sie mit

`condor_status -long` *`Computername`*

abfragen können. In Tabelle 3.3 sind nur die wichtigsten aufgelistet. Eine Liste der im Morfeus-Grid verfügbaren Betriebssysteme (`OpSys`), Prozessorarchitekturen (`Arch`) und CPU-Befehlssatzerweiterungen finden Sie in Abschnitt 4.2.

Attribut	Beschreibung	Beispiel
`ARCH`	Prozessorarchitektur	”X86_64”
`Name`	Name des Knotens	”slot1@PCFT15.nwz.wwu.de”
`Machine`	Name des Computers	”PCFT15.nwz.wwu.de”
`OpSys`	Betriebssystem	”WINDOWS”
`KFlops`	Kilo Floating Point Operations Per Second	966574
`Mips`	Million Instructions Per Second	2367

Tabelle 3.3.: Die wichtigsten Attribute (*ClassAds*) der Computer. Beachten Sie, dass manche Attribute in Anführungsstriche gesetzt werden müssen, da es sich hier um String-Variablen handelt. Siehe hierzu auch die Beispiele am Ende dieses Abschnitts.

In Tabelle 3.4 sind die wichtigsten Vergleichsoperatoren zusammengestellt. Mit diesen Operatoren lassen sich Bedingungen (`requirements`) in der Jobbeschreibungsdatei erstellen.

`==`	Gleich	`<`	Kleiner als
`||`	Oder	`>`	Größer als
`&&`	Und	`<=`	Kleiner gleich
`!=`	Ungleich	`>=`	Größer gleich

Tabelle 3.4.: Vergleichsoperatoren

Beispiel 1:

Ich benötige Windows und einen 32- oder 64-Bit-Prozessor. In meiner Jobbeschreibungsdatei muss dann stehen:

```
requirements=(((Arch=="INTEL") || (Arch=="X86_64")) && (OpSys=="WINDOWS"))
```

Beispiel 2:
Ich benötige Windows und einen Prozessor, der AVX beherrscht. In meiner Jobbeschreibungsdatei muss dann stehen:

```
requirements=OpSys=="WINDOWS" && HasFeatureAVX==true
```

Beachten Sie hier unbedingt den Unterschied zwischen dem "vergleichenden Gleich" (==) und dem Gleichheitszeichen (=) beim Setzen der `requirements`!

Haben Sie sehr strikte Anforderungen an die Attribute (*ClassAds*) der Knoten, die Ihren Job ausführen sollen, so können Sie vorher überprüfen, ob und wieviele Knoten im Morfeus-Grid diese Anforderungen überhaupt erfüllen. In diesem Beispiel benötige Ihr Job das Betriebssystem Linux und einen Prozessor, der AVX unterstützt. Führen Sie dazu in der Eingabeaufforderung

```
condor_status -constraint "HasFeatureAVX==true && OpSys==\"LINUX\""
```

aus. Wie man sieht, muss der ganze Ausdruck der benötigten *ClassAds* in Anführungsstriche (") gesetzt werden. Die Werte der String-Variablen müssen ebenfalls in Anführungsstriche gesetzt werden, allerdings mit einem vorangesetzten Backslash (\"). Der Befehl erzeugt eine Liste der Knoten im Morfeus-Grid, die den geforderten Anforderungen entsprechen, so dass man vorher schon überprüfen kann, ob das Morfeus-Grid für die eigenen Jobs geeignet ist.

3.6. Die Log-Datei

Für jeden Job sollte stets eine Log-Datei erstellt werden, da sie wichtige Informationen über den Verlauf des Jobs beinhaltet. Ohne diese Log-Datei ist z.B. eine Fehleranalyse praktisch nicht möglich. Eine typische Log-Datei sieht wie folgt aus:

```
1  000 (1897.000.000) 08/22 16:26:29 Job submitted from host: <128.176.197.24:
2
3  001 (1897.000.000) 08/22 16:26:38 Job executing on host: <128.176.240.82:10
4
5  004 (1897.000.000) 08/22 20:56:55 Job was evicted.
6        (0) Job was not checkpointed.
7              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
8              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
9        0  -  Run Bytes Sent By Job
10       0  -  Run Bytes Received By Job
11
12 001 (1897.000.000) 08/22 20:59:38 Job executing on host: <128.176.240.123:1
13
```

```
14 010 (1897.000.000) 08/23 10:04:01 Job was suspended.
15       Number of processes actually suspended: 1
16
17 011 (1897.000.000) 08/23 11:32:15 Job was unsuspended.
18
19 005 (1897.000.000) 08/24 09:50:49 Job terminated.
20       (1) Normal termination (return value 0)
21             Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
22             Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
23             Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
24             Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
25       7120444  -  Run Bytes Sent By Job
26       3538741  -  Run Bytes Received By Job
27       7120444  -  Total Bytes Sent By Job
28       3538741  -  Total Bytes Received By Job
```

Die Zeile 1 gibt Auskunft über das Abschicken des Jobs. Man erfährt dort die Job-ID (hier `1897`), das Datum und die Uhrzeit, wann der Job abgeschickt wurde, und die IP-Adresse des Computers, von dem aus der Job abgeschickt wurde. In Zeile 3 erfährt man, wann der Job gestartet wurde, und die IP-Adresse des ausführenden Knotens. In der Zeile 5 teilt die Log-Datei mit, dass der Job abgebrochen (`evicted`) wurde. Die Zeilen 6-10 geben noch Auskunft darüber, dass es keinen *Checkpoint* gab. Das *Checkpointing* wird im MORFEUS-Grid nicht unterstützt, da derzeit diese Implementierung für HTCONDOR unter Windows nicht existiert. In Zeile 12 erfährt man, dass der Job ein paar Minuten später auf einem anderen Knoten neu gestartet wurde. In Zeile 14 wird mitgeteilt, dass der Job unterbrochen (`suspended`) wurde, da der Knoten gerade vor Ort benutzt wird. Sobald der Knoten, auf dem der Job gestartet wurde, vor Ort nicht mehr benötigt wird, wird der Job fortgesetzt. In Zeile 17 wird der Job fortgesetzt (`unsuspended`), und zwar genau an der Stelle, an der er vorher angehalten (`suspended`) wurde. In der Zeile 19 ist der Job schließlich beendet (`terminated`) worden. Die Zeile 20 teilt mit, dass es keine Probleme mit dem Job gab, da es sich um eine `normal termination` mit dem `return value 0` handelt. Sollten Sie einen anderen `return value` erhalten haben, ist der Job nicht ordnungsgemäß durchgelaufen. Überprüfen Sie dann unbedingt Ihre Error- und Output-Datei. Ein möglicher Grund kann auch ein fehlerhaft konfigurierter Zielknoten sein, siehe Abschnitt 4.7. Die Zeilen 21-28 geben noch ein paar Informationen zum gesendeten und empfangenen Datenvolumen.

3.7. Abfrage der Warteschlange (`condor_q`)

Wenn Sie die aktuelle Warteschlange ansehen möchten, z.B. um zu schauen, welche Ihrer Jobs bereits berechnet werden oder schon fertig sind, wählen Sie in der MorfeusConsole die entsprechende Registerkarte aus. Sie sehen dort eine Liste mit verschiedenen Informationen zu allen momentan in der Warteschlange stehenden Jobs. Jobs, die bereits abgeschlossen wurden, tauchen hier nicht mehr auf.

Job-ID	N.	Abgeschickt	Laufzeit (bislang)	CPU-Zeit (bislang)	Status	Priorität	Kommandozeile	Ab
4.0	w.	30.04.201...	00:00:00	00:00:00	R (läuft)	0	gruss.exe	W

Wenn Sie nicht nur die Jobs sehen möchten, die von dem Computer aus gestartet wurden, an dem Sie gerade arbeiten, sondern alle Jobs im Morfeus-Grid, kreuzen Sie die Option „global“ an.

In der Kommandozeile führen Sie den Befehl

```
condor_q
```

aus. Möchten Sie die Jobs im gesamten Morfeus-Grid einsehen, so müssen Sie das Kommando

```
condor_q -global
```

ausführen. Sie erhalten in beiden Fällen eine Liste wie in Tabelle 3.5 abgebildet.

ID	OWNER	SUBMITTED	RUN_TIME	ST	PRI	SIZE	CMD
482.0	tombauer	7/8 09:51	0+03:32:25	I	0	57.3	phondos.exe
484.0	falter	7/8 10:55	0+02:28:59	R	0	104.5	strukfit.exe
830.0	tombauer	2/8 13:32	2+10:28:59	H	0	4.5	gorkim.exe

Tabelle 3.5.: Ausgabe von `condor_q`.

Wie Sie sehen, enthält die Ausgabe auf der Kommandozeile im Vergleich zur MorfeusConsole aus Platzgründen weniger Spalten. Wie Sie die fehlenden Spalten anzeigen können, ist weiter unten bei den jeweiligen Erläuterungen beschrieben.

In der ersten Spalte finden Sie die Job-ID, die HTCondor Ihrem Job zugewiesen hat. Diese Job-ID wurde Ihnen beim Abschicken des Jobs direkt mitgeteilt, und sie steht auch noch zusätzlich in der Log-Datei des Jobs. In der nächsten Spalte sehen Sie, wer den Job abgeschickt hat. Es folgt die Angabe, wann der Job abgeschickt wurde. Die Laufzeit bzw. `RUN_TIME` gibt die Zeit an, die der Job bereits im Morfeus-Grid verweilt. Die bislang verbrauchte CPU-Zeit, also die Zeit, die der Job bisher wirklich gerechnet wurde, finden Sie in der Spalte „CPU-Zeit“. In der Eingabeaufforderung müssen Sie die Option `-cpu` verwenden, um diese Spalte anzeigen zu lassen, also:

```
condor_q -cpu
```

Die Spalte „Status“ bzw. `ST` gibt den momentanen Status des Jobs an. `R` bedeutet `Running` und heißt, dass der Job läuft. `I` steht für `Idle` und bedeutet, dass der Job nicht ausgeführt wird. Es kann verschiedene Gründe haben, warum der Job nicht ausgeführt wird. Um die Ursache zu finden, schauen Sie in Abschnitt 3.11. Der Status `H` steht für `Hold`, weitere Hinweise dazu finden Sie in Abschnitt 3.9.
Anhand von „Priorität“ bzw. `PRI` lässt sich eine von Benutzer oder Benutzerin eingestellte Priorität für den Job erkennen (siehe offizielles HTCondor-Manual, Abschnitt 2.6.4).
In der Spalte „Kommandozeile“ bzw. `CMD` sehen Sie, welcher Befehl im Rahmen des Jobs ausgeführt wird.
Schließlich wird Ihnen bei bereits laufenden Jobs noch angezeigt, auf welchem Rechner sie ausgeführt werden. Diese Spalte wird in der Eingabeaufforderung üblicherweise nicht angezeigt, verwenden Sie hier die Option `-run`, also:

```
condor_q -run
```

3.8. Jobs aus der Warteschlange löschen (condor_rm)

Um einen Job aus der lokalen Warteschlange zu löschen, klicken Sie ihn in der Morfeus-Console mit der rechten Maustaste an und wählen Sie den entsprechenden Befehl aus dem Kontextmenü.

In einer Eingabeaufforderung können Sie stattdessen den Befehl:

```
condor_rm ID
```

verwenden. *ID* ist dabei die entsprechende Job-ID, die Sie mit `condor_q` (Abschnitt 3.7) abfragen. Es können nur die eigenen Jobs aus der Warteschlange gelöscht werden!
Falls Sie einen Job aus der Warteschlange eines anderen Computers (z.B. PCFT15) über die Eingabeaufforderung löschen möchten, so müssen Sie das Kommando

```
condor_rm -name PCFT15.nwz.wwu.de ID
```

ausführen.

3.9. Jobs anhalten (condor_hold) und fortsetzen (condor_release)

Manchmal kommt es vor, dass Sie einen Job nicht aus der Warteschlange komplett entfernen, sondern nur anhalten möchten. Beachten Sie jedoch, dass der Job, falls er bereits angelaufen ist, durch das Anhalten vom aktuellen Rechenknoten geworfen wird, d.h. alle bisher durchgeführten Berechnungen des Jobs gehen verloren[4].

Klicken Sie den Job dazu in der Warteschlange der MorfeusConsole mit rechts an und wählen Sie den entsprechenden Befehl aus dem Kontextmenü.

In der Eingabeaufforderung müssen Sie den folgenden Befehl ausführen:

```
condor_hold ID
```

Um einen Job wieder fortzusetzen, nutzen Sie wieder das Kontextmenü oder führen das Kommando

```
condor_release ID
```

aus. Wie bei dem Kommando `condor_rm` aus Abschnitt 3.8 müssen Sie (falls Sie nicht die MorfeusConsole verwenden) hier auch die Option `-name` *Computername* nutzen, wenn Sie Jobs anhalten oder fortsetzen möchten, die Sie von einem anderen Computer aus abgeschickt haben.

Es kann auch vorkommen, dass HTCondor automatisch Jobs anhält, wenn diese zu viele Fehler im Morfeus-Grid verursacht haben. Dies muss nicht zwingend die Schuld Ihres Jobs sein; bei kurzzeitigen Netzwerkproblemen oder fehlerhaft konfigurierten Knoten (siehe Abschnitt 4.7) kann Ihr Job ebenfalls von HTCondor angehalten werden. Geben Sie

[4] vgl. aber Kapitel 4.10

Ihren Job in diesem Fall dann einfach über das Kontextmenü oder mit `condor_release` wieder frei, damit der Job neu gestartet werden kann.

3.10. Priorität abfragen (`condor_userprio`)

Um die Prioritäten der Benutzer und Benutzerinnen des Pools abzufragen, müssen Sie die Kommandozeile benutzen. Führen Sie

```
condor_userprio -all -allusers
```

aus. Es ist zu beachten, dass ein höherer Zahlenwert der Priorität eine geringere Priorität bedeutet. Die höchste Priorität entspricht dem niedrigsten Zahlenwert: 0.5. Jeder Benutzer / jede Benutzerin startet mit einer Priorität von 0.5. Dieser Wert steigt bei Benutzung des Pools an und nimmt bei wenig oder keiner Benutzung wieder ab. Die Jobs der Benutzer und Benutzerinnen mit höheren Prioritäten (= niedrigerem Zahlenwert) werden vor den Jobs der Benutzer und Benutzerinnen mit geringeren Prioritäten gestartet, wenn es einen Mangel an zur Verfügung stehenden Computern gibt.

Wichtig: Diese Priorität ist nicht mit der Priorität zu verwechseln, die Sie für Ihre eigenen Jobs einstellen (Abschnitt 3.7).

3.11. Warum startet mein Job nicht?

Nachdem Sie Ihren Job abgeschickt haben, sollten Sie ca. eine Minute warten und dann in der MorfeusConsole oder mit `condor_q` nachschauen, ob in der Spalte „Status“ bzw. `ST` ein `R` für Running steht.
Steht dort ein `I`, bedeutet das, dass sich der Job momentan im Idle-Status befindet. Um nachzuschauen, warum er nicht startet, sollten Sie dann zuerst eine Jobanalyse anfordern. Das funktioniert in der MorfeusConsole über das Kontextmenü und in der Eingabeaufforderung über den Befehl:

```
condor_q -analyze ID
```

Sie erhalten dann nach ein paar Sekunden eine Liste der folgenden Form:

```
12  are rejected by your job's requirements
10  reject your job because of their own requirements
 5  match, but are serving users with a better priority in the pool
 0  match, but reject the job for unknown reasons
 7  match, but will not currently preempt their existing job
 4  are available to run your job
```

Die meisten dieser Zeilen sind selbsterklärend. An dieser Stelle soll jedoch auf drei Punkte hingewiesen werden:

- Wenn in der letzten Zeile steht, dass Maschinen bereit stehen, den Job zu starten, heißt das, dass Sie sich nur noch einige Sekunden (manchmal Minuten) gedulden müssen, bis HTCondor den Job startet.

- Steht hingegen in der 4. Zeile eine von Null verschiedene Zahl, so bedeutet dies, dass HTCondor nicht weiß, warum der Job nicht gestartet wird. In diesem Fall müssen Sie Ihre Jobbeschreibungsdatei überprüfen.
  Eine Fehlerursache kann sein, dass Sie Ihr Passwort geändert haben und diese Änderung HTCondor nicht mitgeteilt haben. Näheres finden Sie in Abschnitt 3.2.
- Es kann auch passieren, dass Sie in Ihren `requirements` Anforderungen gestellt haben, die im ganzen Morfeus-Grid nicht verfügbar sind. In diesem Fall steht unter der Tabelle noch der Eintrag:

  ```
  WARNING: Be advised:
  No resources matched request's constraints
  Check the Requirements expression below:
  ```

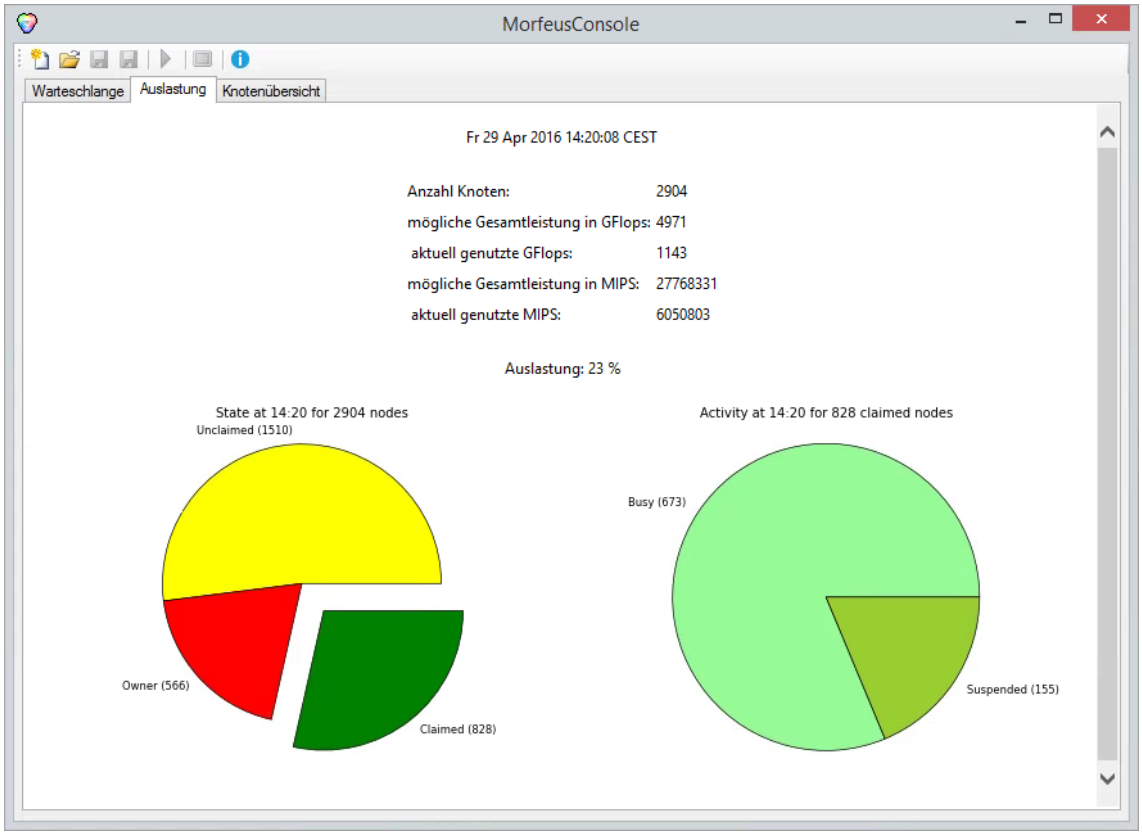
  Der Job wird also niemals starten. Löschen Sie den Job aus der Warteschlange (Abschnitt 3.8) und überprüfen Sie Ihre `requirements`. Starten Sie ihren Job dann erneut (Abschnitt 3.4).

4. Das Morfeus-Grid

Dieses Kapitel beschäftigt sich mit den speziellen Eigenschaften des Morfeus-Grids.

4.1. Auslastung

Wenn Sie sich nur einen schnellen Überblick über die momentane Auslastung des Morfeus-Grids verschaffen möchten, sind die detaillierten Angaben von `condor_status` (s. Abschnitt 3.3) unnötig und nicht leicht zu überblicken. Verwenden Sie dann einfach den Tab „Auslastung“ der MorfeusConsole:



Die Angaben zur Auslastung werden allerdings nur alle 10 Minuten aktualisiert. Wenn Sie nicht die MorfeusConsole verwenden, finden Sie dieselben Informationen auch auf der Morfeus-Homepage der IVV.

Betriebssysteme	`OpSys`
Linux	"LINUX"
Windows	"WINDOWS"

Tabelle 4.1.: Die Betriebssysteme (`OpSys`) im Morfeus-Grid

Prozessorarchitekturen	`Arch`
Intel/AMD 32 Bit (IA32)	"INTEL"
Intel/AMD 64 Bit (Intel64)	"X86_64"

Tabelle 4.2.: Die Prozessorarchitekturen (`Arch`) im Morfeus-Grid

4.2. Ressourcen

Alle Computer des Morfeus-Grids erfüllen zwei Aufgaben. Zum einen können die Benutzer und Benutzerinnen von diesen Computern aus Jobs abschicken, und zum anderen werden diese Computer als Rechenknoten benutzt, solange niemand an dem Computer arbeitet.

Aufgrund der Vielfalt von Betriebssystemen und Prozessorarchitekturen in der IVV Naturwissenschaften sind auch im Morfeus-Grid verschiedene Varianten (siehe Tabelle 4.1 und Tabelle 4.2) vertreten. Die meisten Rechner sind 64-Bit-Intel-Rechner, es gibt jedoch auch noch einige 32-Bit-Maschinen. Andere Architekturen wie PowerPC und Itanium (IA64) sind momentan nicht (mehr) im Morfeus-Grid vertreten. Als Betriebssystem verwenden alle Rechner im Grid Linux oder Windows.

Um eine genauere Spezifikation des benötigten Prozessors zu ermöglichen, wird im Grid auf jedem Rechner[5] ein Testprogramm ausgeführt, das überprüft, welche Erweiterungen des Befehlssatzes durch den Prozessor unterstützt werden. Für jede unterstützte Erweiterung wird ein entsprechendes Attribut definiert (s. Tabelle 4.3).

Die aktuelle Computerliste des Morfeus-Grids ist recht dynamisch, da mit der Zeit immer mehr Computer in den Pool aufgenommen werden. Daher kann an dieser Stelle keine Aufzählung der verfügbaren Ressourcen erfolgen. Informieren Sie sich also immer vor der Benutzung von Morfeus über den aktuellen Status des Grids (Abschnitt 3.3), und schauen Sie, ob die von Ihrem Programm benötigten Ressourcen vorhanden sind.

Falls Ihr Programm beispielsweise unter Windows kompiliert wurde, kann es nicht auf einem Linux-Knoten gerechnet werden. Es wäre aber sehr wohl möglich, diesen Job von einem Linux-Rechner aus an einen Windows-Knoten abzuschicken. Beachten Sie jedoch unbedingt die korrekte Einstellung der `requirements` (Abschnitt 3.5).

4.3. Statisches Linken

Wenn möglich, sollten Sie Ihr Programm statisch linken, damit es alle Bibliotheken enthält, die es benötigt. Dies ist insbesondere unter Linux wichtig, da, anders als bei Win-

[5] Momentan erfolgt diese Überprüfung nur auf Windows-Rechnern. Eine entsprechende Funktion für die Linux-Rechner ist geplant.

Befehlssatz	`Definiertes Attribut`
3DNOW	”HasFeature3DNOW”
3DNOWEXT	”HasFeature3DNOWEXT”
ABM	”HasFeatureABM”
ADX	”HasFeatureADX”
AES	”HasFeatureAES”
AVX	”HasFeatureAVX”
AVX 2	”HasFeatureAVX2”
AVX 512CD	”HasFeatureAVX512CD”
AVX 512ER	”HasFeatureAVX512ER”
AVX 512F	”HasFeatureAVX512F”
AVX 512PF	”HasFeatureAVX512PF”
BMI 1	”HasFeatureBMI1”
BMI 2	”HasFeatureBMI2”
CLFSH	”HasFeatureCLFSH”
CMPXCHG16B	”HasFeatureCMPXCHG16B”
CX8	”HasFeatureCX8”
ERMS	”HasFeatureERMS”
F16C	”HasFeatureF16C”
FMA	”HasFeatureFMA”
FSGSBASE	”HasFeatureFSGSBASE”
FXSR	”HasFeatureFXSR”
HLE	”HasFeatureHLE”
INVPCID	”HasFeatureINVPCID”
LAHF	”HasFeatureLAHF”
LZCNT	”HasFeatureLZCNT”
MMX	”HasFeatureMMX”
MMXEXT	”HasFeatureMMXEXT”
MONITOR	”HasFeatureMONITOR”
MOVBE	”HasFeatureMOVBE”
MSR	”HasFeatureMSR”
OSXSAVE	”HasFeatureOSXSAVE”
PCLMULQDQ	”HasFeaturePCLMULQDQ”
POPCNT	”HasFeaturePOPCNT”
PREFETCHWT1	”HasFeaturePREFETCHWT1”
RDRAND	”HasFeatureRDRAND”
RDSEED	”HasFeatureRDSEED”
RDTSCP	”HasFeatureRDTSCP”
RTM	”HasFeatureRTM”
SEP	”HasFeatureSEP”
SHA	”HasFeatureSHA”
SSE	”HasFeatureSSE”
SSE 2	”HasFeatureSSE2”
SSE 3	”HasFeatureSSE3”
SSE 4.1	”HasFeatureSSE4_1”
SSE 4.2	”HasFeatureSSE4_2”
SSE 4a	”HasFeatureSSE4a”
SSSE 3	”HasFeatureSSSE3”
SYSCALL	”HasFeatureSYSCALL”
TBM	”HasFeatureTBM”
XOP	”HasFeatureXOP”
XSAVE	”HasFeatureXSAVE”

Tabelle 4.3.: Befehlssatzerweiterungen (`HasFeature...`) im Morfeus-Grid

dows, im NWZ verschiedene Distributionen mit unterschiedlichen vorinstallierten Bibliotheken zum Einsatz kommen. Ein sehr häufiges Problem betrifft die Intel-Linux-Compiler: Wenn Sie nicht mindestens die Intel-Laufzeitbibliotheken über die Option `-static-intel` statisch linken, wird Ihr Programm auf keinem Linux-Rechner laufen, auf dem die Intel-Compiler nicht ebenfalls installiert sind. Wenn Sie nicht statisch linken können und Ihr Programm tatsächlich auf einigen Rechnern Probleme bereitet, legen Sie die nötigen Bibliotheken (.dll unter Windows, .so unter Linux) in denselben Ordner wie Ihr Programm und übertragen Sie sie an den ausführenden Rechner, indem Sie sie als zusätzlich benötigte Dateien angeben (per `transfer_input_files`). Unter Linux sollten Sie dann noch dafür sorgen, dass die Umgebungsvariable `LD_LIBRARY_PATH` das aktuelle Verzeichnis enthält, z.B. indem Sie folgendes Shellskript als `executable` festlegen (und das eigentliche Programm als zusätzliche benötigte Datei angeben):

```
1 #!/bin/bash
2 LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
3 ./das_eigentliche_programm
```

Falls Sie Linux-Programme auf Windows-Rechnern ausführen (s. Abschnitt 4.8), brauchen Sie `LD_LIBRARY_PATH` nicht extra zu setzen; das erledigt bereits das Hilfsprogramm für Sie.

4.4. Arbeitsverzeichnis

Das Arbeitsverzeichnis hat mehrere Funktionen. Zum einen lagern Sie dort Ihre Jobbeschreibungsdatei, und zum anderen schreibt HTCondor alle Ergebnisdateien, Log-Dateien, Output-Dateien usw. in genau dieses Verzeichnis zurück, sobald der Job beendet wurde. Bei der Wahl Ihres Arbeitsverzeichnisses sind Sie fast völlig frei. Sie können entweder ein lokales oder ein Netzlaufwerk wählen. Sie sollten sich jedoch gut überlegen, ob Sie wirklich ein lokales Laufwerk als Arbeitsverzeichnis benutzen wollen, besonders wenn Sie keinen eigenen Arbeitsplatzrechner besitzen. Da HTCondor die Ergebnisse wieder in das lokale Verzeichnis zurückschreibt, müssen Sie sich auch genau an diesen Computer begeben, um Ihre Ergebnisse abzuholen.

Wichtig ist auch, dass das Arbeitsverzeichnis während der gesamten Laufzeit des Jobs verfügbar bleibt. Es ist beispielsweise eine ganz schlechte Idee, sich von einem Notebook aus auf dem Terminalserver anzumelden und die Jobs dann aus einem Verzeichnis auf dem Notebook abzuschicken, das über den Citrix-Client auch für die Anwendungen auf dem Terminalserver bereitgestellt wird. Wenn Sie dann nämlich die Verbindung zum Terminalserver trennen, kann dieser nicht länger auf das Arbeitsverzeichnis zugreifen und muss den Job abbrechen, ohne auch nur einen Hinweis auf das Problem in der Logdatei speichern zu können. Sie dürften sich also auf keinen Fall vom Terminalserver abmelden oder die Verbindung trennen, bevor alle Ihre Jobs durchgelaufen sind. Kopieren Sie also die Dateien auf jeden Fall vorher auf ihr persönliches Netzlaufwerk `I:`, und schicken Sie sie von dort aus ab. Dann können Sie sich problemlos vom Terminalserver abmelden und die fertigen Ergebnisse später abholen.

4.5. Pfade

Sind in Ihrem Programm Pfadangaben gemacht worden, so müssen Sie diese entfernen. Öffnet Ihr Programm beispielsweise eine Datei `Q:\PROGRAMS\CALC\PARAMETER.DAT`, so ist der komplette Pfad `Q:\PROGRAMS\CALC\` aus dem Programmcode zu entfernen. Beachten Sie aber, dass diese Datei in Ihrer Jobbeschreibungsdatei als mitzutransferierende Datei mit angegeben wird (Abschnitt 3.5). In diesem Beispiel muss also die Zeile `transfer_input_files = Q:\PROGRAMS\CALC\PARAMETER.DAT` in Ihrer Jobbeschreibungsdatei stehen.

Legt Ihr Programm Unterverzeichnisse an, ist zu beachten, dass HTCondor die Unterverzeichnisse und darin angelegte Dateien nach dem Programmende nicht automatisch zurückkopiert. Sie müssen dann über die Angabe `transfer_output_files = ...` angeben, welche Dateien und Ordner zurückkopiert werden sollen. (Das automatische Zurückkopieren ist damit deaktiviert, d.h. sie müssen auch Dateien im Arbeitsverzeichnis angeben, wenn diese kopiert werden sollen.)

4.6. Der Windows-Terminalserver NWZCitrix

Bei dem Server NWZCitrix handelt es sich um einen Windows-Terminalserver. Sie können sich von jedem Rechner aus mit Internetzugang dort einloggen und finden dort dann ihre gewohnte NWZ-Arbeitsumgebung wieder. Da auf NWZCitrix HTCondor installiert ist, können Sie von diesem Server aus Jobs in das Morfeus-Grid schicken. Es ist jedoch eine Besonderheit zu beachten:

Der Server NWZCitrix besteht aus vielen verschiedenen Rechnern. Wenn Sie sich einloggen, entscheidet die Software, auf welchem Rechner Sie eingeloggt werden, um einen optimalen Lastenausgleich zwischen den Rechnern zu erzielen. Da es sich um verschiedene Rechner handelt, existieren dort auch verschiedene Warteschlangen. Es kann nun passieren, dass Sie sich auf NWZCitrix einloggen und vom System z.B. auf den ersten Rechner geschoben werden. Wenn Sie dann einen Job abschicken, wird dieser Job in die Warteschlange des ersten Rechners gestellt. Falls Sie sich danach aus- und wieder einloggen, könnte es sein, dass Sie vom System z.B. auf den zweiten Rechner eingeloggt werden.

Wenn Sie dann mit `condor_q` oder in der MorfeusConsole nach Ihren Jobs schauen, könnte der Eindruck entstehen, dass Ihre Jobs verschwunden sind. In Wirklichkeit stecken Ihre Jobs aber in der Warteschlange des ersten Rechners, so dass Sie Ihre Jobs nur mit dem Kommando `condor_q -global` bzw. durch Ankreuzen von „global“ in der MorfeusConsole sehen können. Wenn Sie Jobs über die Eingabeaufforderung löschen, anhalten oder fortsetzen möchten, müssen Sie dann auch immer die Option `-name` verwenden. Merken Sie sich dazu den Namen des Rechners, in dessen Warteschlange Ihr Job steht. Dieser wird von `condor_q -global` ausgegeben (`- Schedd: abc.nwz.wwu.de`). Beim Entfernen von Jobs (`condor_rm`) und dem Anhalten (`condor_hold`) bzw. Fortsetzen (`condor_release`) von Jobs müssen Sie dann die Option `-name abc.nwz.wwu.de` angeben.

In der MorfeusConsole können Sie Jobs immer ganz normal über das Kontextmenü steuern, egal auf welchem Rechner sie laufen.

Beachten Sie unbedingt auch die Hinweise in Abschnitt 4.4.

4.7. Fehlerhaft konfigurierte Knoten

Das Morfeus-Grid setzt sich aus Rechnern der IVV Naturwissenschaften, also dem NWZ, zusammen. Da die Administration dieser Rechner sehr dezentral verläuft, kann es vorkommen, dass einzelne Knoten im Morfeus-Grid nicht funktionieren, wenn z.B. sehr strikte Einstellungen an der Firewall vorgenommen wurden. Für den Administrator des Morfeus-Grids ist es unmöglich, alle diese Knoten zu identifizieren und aus dem Grid auszuschließen; daher müssen die Benutzer und Benutzerinnen bei Problemen selbst aktiv werden.

Als erstes sollten Sie prüfen, ob sich Ihre Probleme durch statisches Linken (vgl. Abschnitt 4.3) beheben lassen.

Wenn man sich sicher ist, dass ein ganz bestimmter Rechner den eigenen Jobs Probleme bereitet, ist es am einfachsten, diesen Rechner direkt in den `requirements` der Jobbeschreibungsdatei auszuschließen. Stellen Sie beispielsweise in Ihrer Log-Datei fest, dass der ausführende Knoten mit der IP-Adresse `128.176.198.40` Ihren Job immer abbricht, müssen Sie zuerst den Namen des Rechners herausbekommen. Führen Sie dazu

```
nslookup 128.176.198.40
```

in der Konsole aus. Sie erhalten als Ausgabe:

```
Name: PCFT01.UNI-MUENSTER.DE
Address: 128.176.198.40
```

Um zu verhindern, dass in Zukunft Jobs auf diesem Knoten gestartet werden, müssen Sie ihn in Ihren `requirements` ausschließen. Es ist allerdings zu beachten, dass die meisten Rechner im NWZ sich unter der Domäne `NWZ.WWU.DE` anmelden. Schauen Sie also am besten zuerst nach, unter welchem Namen der Rechner tatsächlich im Morfeus-Grid angemeldet ist:

```
condor_status PCFT01.UNI-MUENSTER.DE
```

Wenn hier kein Rechner gefunden wird, versuchen Sie:

```
condor_status PCFT01.NWZ.WWU.DE
```

Hier sollte der Rechner dann angezeigt werden. Schließlich tragen Sie den richtigen Namen in Ihre `requirements` ein:

```
requirements = Machine != "PCFT01.NWZ.WWU.DE"
```

Normalerweise wird jeder Job nach Beendigung aus der Warteschlange gelöscht, egal ob der Rückgabewert[6] gleich oder ungleich Null ist. Wenn Sie aber zum Beispiel nur Jobs aus der Warteschlange gelöscht haben möchten, wenn der Rückgabewert Null ist (der Job also ordnungsgemäß durchgelaufen ist), aber noch einmal neugestartet werden soll, wenn der Rückgabewert ungleich Null ist, so können Sie das mit dem Eintrag

```
on_exit_remove = (ExitCode == 0)
```

[6] errorlevel, exit code

in der Jobbeschreibungsdatei steuern. Wenn Sie außerdem noch möchten, dass der Job nicht noch einmal auf dem gleichen Knoten gestartet wird, so tragen Sie die Zeile

```
requirements = Name=!=LastRemoteHost
```

in die Jobbeschreibungsdatei ein. Beachten Sie hier unbedingt das "`=!=`". Dieser Vergleichsoperator vergleicht nur, wenn beide ClassAds auch definiert sind. Das ist hier wichtig, da beim ersten Starten `LastRemoteHost` noch nicht definiert ist.

Bitte verwenden Sie diese Optionen aber nur, wenn Sie sich sicher sind, dass tatsächlich fehlerhaft konfigurierte Knoten an den Abbrüchen schuld sind und Ihr Programm prinzipiell ordnungsgemäß arbeitet. Wenn der Job nämlich wegen eines Programmierfehlers immer wieder abstürzt und dann nicht aus der Warteschlange entfernt wird, sondern wieder und wieder neugestartet wird, verschwenden Sie unnötig Rechenzeit (und ruinieren sich nebenbei Ihre Priorität im MORFEUS-Grid, s. Abschnitt 3.10).

4.8. Linux-Programme auf Windows-Rechnern

Der größte Teil der Rechner im MORFEUS-Grid arbeitet mit dem Betriebssystem Windows. Viele der im NWZ entwickelten wissenschaftlichen Programme sind jedoch für Linux geschrieben, was nicht zuletzt deshalb sinnvoll ist, weil die meisten Großrechner (darunter auch die der IVV und des ZIV an der Universität Münster) mit Linux arbeiten. Wenn diese Programme im MORFEUS-Grid ausgeführt werden sollten, mussten sie entweder mit dem kleinen Teil der unter Linux laufenden Maschinen auskommen oder auf Windows portiert werden.

Wenn eine Portierung sehr schnell möglich ist, z.B. durch einfaches Neukompilieren der Software auf einem Windows-Rechner (für Linux-Nutzer bietet sich der Terminalserver an), ist dies nach wie vor die beste Lösung. Wenn sich die Portierung aber als schwierig erweist, z.B. weil die eingesetzten Bibliotheken unter Windows nicht verfügbar sind, gibt es jetzt auch die Möglichkeit, die Linux-Programme in unveränderter Form an Windows-Rechner zu schicken, wo sie auf einer virtuellen Maschine ausgeführt werden.[7] Die Ausführung der Jobs wird dabei um wenige Minuten verzögert, was durch das Erstellen und Hochfahren der virtuellen Maschine bedingt ist. Um keine Rechenzeit zu verschwenden, sollten Sie daher nur Jobs an virtuelle Maschinen senden, die wenigstens eine Viertelstunde rechnen. Kürzer laufende Jobs können Sie sicherlich auch in größerer Zahl direkt auf den im MORFEUS-Grid zur Verfügung stehenden „echten“ Linux-Computern berechnen lassen.

Im Vergleich zum üblichen Abschicken an andere Linux-Rechner sind nur wenige Änderungen an der Jobbeschreibungsdatei nötig:

Die Zeile `executable` wird ersetzt durch vier andere Zeilen. Aus

```
2 executable=a.out
```

wird:

[7] Da im MORFEUS-Grid keine Linux-Rechner mit nur 32 Bit mehr vorhanden sind, wurde diese Möglichkeit nur für 64-Bit-Rechner implementiert.

```
2 executable=\\nwzmorfeus\linshare\MorfeusLinuxWrapper.exe
3 transfer_executable=false
4 transfer_input_files=a.out
5 environment = "linuxExecutable=a.out"
```

Da das Programm nicht direkt auf dem Windows-Rechner läuft, muss ein Hilfsprogramm zwischengeschaltet werden, das die virtuelle Linux-Maschine verwaltet. Dieses Programm wird also als `executable` angegeben. Da es über das Netzwerk von jedem Windows-Rechner im NWZ erreichbar ist, braucht es nicht kopiert zu werden, was in Zeile 3 festgelegt wird. Das eigentliche auszuführende Programm (hier `a.out`) dagegen muss nach wie vor kopiert werden, was jetzt aber nicht mehr automatisch passiert. Daher wird über Zeile 4 per Hand festgelegt, dass es kopiert werden soll. (Wenn bereits andere Dateien per `transfer_input_files` zur Übertragung vorgemerkt werden, fügt man die ausführbare Datei dort einfach an.) Schließlich muss auch noch festgelegt werden, welches Linux-Programm gestartet werden soll, da die Zeile `executable` ja nur noch das Hilfsprogramm spezifiziert. Das passiert in der letzten Zeile durch das Festlegen einer Umgebungsvariable.

Als nächstes muss die Zeile mit den `requirements` angepasst werden. Zum einen soll das Programm ja nun an Windows-Rechner geschickt werden, zum anderen sollten diese Rechner unbedingt über (aktivierte) Hardware-Unterstützung für Virtualisierung verfügen, da das Programm ansonsten deutlich langsamer arbeiten würde als ohne zwischengeschaltete virtuelle Maschine. Daher machen wir aus

```
7 requirements=((OpSys=="LINUX") && (ARCH=="X86_64"))
```

die Zeile:

```
10 requirements=((OpSys=="WINDOWS") && (ARCH=="X86_64") &&
     (HasHwVirt == True))
```

Zuletzt müssen wir noch eine Zeile hinzufügen, die dafür sorgt, dass auf dem Windows-Rechner ein Profil[8] geladen wird, da ansonsten die virtuelle Maschine nicht eingerichtet werden kann.

```
11 load_profile=True
```

Weitere Änderungen sind nicht erforderlich, das Hilfsprogramm erledigt den Rest. Im Folgenden finden Sie noch einmal die vollständige Beispiel-Jobbeschreibung, zuerst in der normalen Variante, durch die der Job an einen anderen Linux-Rechner geschickt wird:

```
1 universe=vanilla
2 executable=a.out
3 arguments="--all -n 10"
4 log = beispiel.log
5 output = beispiel.out
6 error = beispiel.err
7 requirements=((OpSys=="LINUX") && (ARCH=="X86_64"))
8 queue
```

[8] Es handelt sich dabei um ein leeres Profil, das HTCondor speziell für diesen Job anlegt und anschließend wieder löscht, nicht um Ihr persönliches Nutzerprofil.

Und hier die Version, in der der Job an einen Windows-Rechner zur Ausführung auf einer virtuellen Maschine geschickt wird:

```
universe=vanilla
executable=\\nwzmorfeus\linshare\MorfeusLinuxWrapper.exe
transfer_executable=false
transfer_input_files=a.out
environment = "linuxExecutable=a.out"
arguments="--all -n 10"
log = beispiel.log
output = beispiel.out
error = beispiel.err
requirements=((OpSys=="WINDOWS") && (ARCH=="X86_64") &&
        (HasHwVirt == True))
load_profile=True
queue
```

Beachten Sie bei der Formulierung Ihrer Jobbeschreibungsdatei, dass die virtuelle Maschine und das darin gestartete Linux ca. 150 MB Arbeitsspeicher benötigen, d.h. wenn Ihr Job z.B. 1,5 GB Arbeitsspeicher nutzt, sollten Sie 1,65 GB anfordern. Weiterhin sollten Sie mindestens doppelt so viel Festplattenspeicher anfordern, wie Ihr Programm für die Ausgabedateien benötigt, da die Dateien nach Beendigung Ihres Programms aus der virtuellen Maschine „herauskopiert“ werden und daher kurzzeitig doppelt existieren.

Ansonsten gelten für Rechnungen auf virtuellen Maschinen genau dieselben Regeln wie für andere Rechnungen auch: Sie werden unterbrochen, wenn der PC, auf dem sie ausgeführt werden, vor Ort benutzt wird, abgebrochen (und dann neu gestartet), wenn der Rechner, auf dem sie laufen, neugestartet wird usw.

Auch die Hinweise zu fehlerhaft konfigurierten Knoten im vorigen Abschnitt und zum statischen Linken in Abschnitt 4.3 können hier nützlich sein.

4.9. Parallele Programme

Sie können im Morfeus-Grid auch parallele Programme ausführen. Dabei können Sie jedoch nur mehrere CPU-Kerne desselben Knotens einsetzen.[9] Um z.B. zwei Kerne zu verwenden, fügen Sie Ihrer Jobbeschreibungsdatei die folgende Zeile hinzu:

```
request_cpus = 2
```

4.9.1. OpenMP

HTCondor setzt automatisch die Variable `OMP_NUM_THREADS` auf die Anzahl der reservierten CPUs. Wenn Ihr Programm mit Hilfe von OpenMP parallelisiert ist, brauchen Sie

[9] Technisch wären auch MPI-Programme realisierbar, die wie in einem Cluster viele Knoten gleichzeitig nutzen, aber das wäre im Morfeus-Grid ineffizient, da ein solcher Job angehalten werden müsste, sobald auch nur einer der eingesetzten Knoten vor Ort verwendet würde und daher zeitweilig nicht für Rechnungen zur Verfügung stünde.

also in der Regel außer der Zeile `request_cpus` keine weiteren Einstellungen vorzunehmen.

4.9.2. MPI

Es ist auch möglich, mit MPI parallelisierte Programme im Morfeus-Grid zu verwenden. Allerdings sind die nötigen MPI-Bibliotheken auf den wenigsten Rechnern im Grid installiert, daher ist die einfachste Möglichkeit, die in Kapitel 4.8 beschriebene Technik zu verwenden, bei der eine virtuelle Maschine zum Einsatz kommt. Kompilieren Sie dazu Ihr Programm unter Linux mit der Intel-MPI-Bibliothek. Verwenden Sie die beiden folgenden Optionen, um sowohl die normalen Intel-Bibliotheken als auch die MPI-Bibliotheken statisch zu linken (die erste Option ist nicht nötig, wenn Sie nicht den Intel-Compiler verwenden):

```
-static-intel -static_mpi
```

Dann geben Sie als ausführbares Programm `mpirun` an und als Argumente:

```
-n 2 ./das_mpi_programm
```

Dabei sollte 2 der Anzahl der angeforderten CPUs entsprechen, und `./das_mpi_programm` ist das auszuführende mit MPI parallelisierte Programm. Insgesamt könnte damit Ihre Jobbeschreibungsdatei wie folgt aussehen (vgl. Kapitel 4.8):

```
1  universe=vanilla
2  executable=\\nwzmorfeus\linshare\MorfeusLinuxWrapper.exe
3  transfer_executable=false
4  transfer_input_files=das_mpi_programm
5  environment = "linuxExecutable=mpirun"
6  arguments="-n 4 ./das_mpi_programm"
7  log = beispiel.log
8  output = beispiel.out
9  error = beispiel.err
10 request_cpus = 4
11 requirements=((OpSys=="WINDOWS") && (ARCH=="X86_64") &&
           (HasHwVirt == True))
12 load_profile=True
13 queue
```

4.10. Laufzeiten von Jobs

Im Morfeus-Grid gibt es keine festgelegte Beschränkung für die Laufzeit Ihrer Programme. Theoretisch können Sie also auch Programme in das Grid schicken, die wochenlang rechnen. Falls ein Knoten, der Ihr Programm ausführt, zwischendurch vor Ort gebraucht wird, ist das kein Problem: Das Programm wird nur angehalten und rechnet später einfach weiter, sobald der Computer wieder frei wird. Um zu verhindern, dass einige Benutzer und Benutzerinnen mit einer Vielzahl von extrem lange laufenden Jobs den Pool für andere Benutzer und Benutzerinnen unbenutzbar machen, wurde jedoch ein Mechanismus

eingebaut, der Jobs unterbrechen kann. Falls die Priorität (siehe Abschnitt 3.10) eines Benutzers oder einer Benutzerin mit einem Job, der momentan von HTCondor auf einem Knoten gerechnet wird, um eine Größenordnung (Faktor 10) geringer ist als die Priorität eines Benutzers oder einer Benutzerin, der oder die einen Job neu in die Warteschlange schickt, so wird der Job des Benutzers oder der Benutzerin mit der geringeren Priorität gestoppt und erst wieder neu gestartet, sobald wieder genug Ressourcen zur Verfügung stehen. Außerdem wird die maximale Laufzeit in der Praxis dadurch beschränkt, dass die Computer im NWZ von Zeit zu Zeit wegen der Installation von Updates oder anderer Software neugestartet werden. Beim Neustart eines Knotens werden alle darauf laufenden Jobs abgebrochen. Sie werden dann zwar auf einem anderen Knoten gleich wieder gestartet, aber alle bereits ausgeführten Berechnungen sind verloren, und das Programm muss wieder von vorne beginnen.

Die Erfahrung hat gezeigt, dass bereits Programme, die länger als 48 Stunden rechnen, nur noch geringe Chancen haben, in akzeptabler Zeit fertig zu werden.

Um dieses Problem zu umgehen, müssen Sie Ihr Programm so schreiben, dass es regelmäßig (oder spätestens dann, wenn es ein Signal zum Beenden erhält) die bisher berechneten Ergebnisse auf der Platte sichert und beim Starten zuerst prüft, ob Zwischenergebnisse auf der Platte gespeichert sind, mit denen es weiterrechnen kann. Sie können dann HTCondor anweisen, bei einem Abbruch des Programms die gespeicherten Daten nicht wegzuwerfen, sondern auf den nächsten Knoten zu kopieren, so dass Ihr Programm nach dem Neustart die Zwischenergebnisse finden und einlesen kann. In diesem Fall kann der Programmfortschritt nur noch in den seltenen Fällen verloren gehen, in denen HTCondor z.B. wegen wegen eines plötzlichen Stromausfalls oder Netzwerkstörungen keine Chance mehr hat, die geschriebenen Zwischenergebnisse noch zu sichern.

Im Folgenden finden Sie ein ganz einfaches Beispiel, das diese Technik verdeutlicht. Die Kreiszahl π soll mit Hilfe eines Monte-Carlo-Verfahrens berechnet werden (s. Abbildung 4.1). Das ursprüngliche Fortran-Programm sieht so aus:

```
1  Program MonteCarloPi
2  Implicit None
3  Integer(8) :: i,treffer
4  Real(8) :: zufall(2)
5  Integer :: time
6  Integer :: seed(20)
7
8  ! Zufallszahlengenerator initialisieren
9  Call System_Clock(time)
10 seed = time
11 Call Random_Seed(Put=seed)
12
13 treffer = 0
14
15 Do i=1, 1000000000000_8
16    Call Random_Number(zufall)
17    If (zufall(1)**2+zufall(2)**2 <= 1.0) treffer = treffer + 1
18 End Do
19
```

```
20 ! Ergebnis
21 Write (*,*) "Pi␣~=␣", treffer*4.0d0/1000000000000_8
22
23 End Program
```

Um wenigstens 5 Nachkommastellen von π korrekt zu berechnen, werden ungefähr eine Billion Zufallszahlen benötigt. Auf einem zur Zeit aktuellen Rechner benötigt dieses Programm für seine Berechnungen mehrere Stunden. Es soll jetzt so erweitert werden, dass es zwischendurch abgebrochen und neugestartet werden kann, ohne von vorne beginnen zu müssen. Dazu ändern wir die Schleife so ab, dass sie alle 10 Mrd. Zufallszahlen, also alle paar Minuten, eine Backup-Datei ausgibt, in der gespeichert wird, wie viele „Treffer“ es bislang gab und wie weit das Programm gerade ist. Man könnte die Datei auch häufiger schreiben, um im Falle eines Abbruchs noch weniger zu verlieren, aber irgendwann verliert man dann durch das dauernde zeitaufwendige Schreiben der Backup-Datei mehr Zeit als man im Falle eines Abbruchs gewinnt. Die Datei wird zuerst unter dem Namen `Backup.tmp` gespeichert und dann umbenannt, damit es nicht passieren kann, dass das Programm ausgerechnet zwischen dem Überschreiben der bisherigen Zwischenergebnisse durch den `Open`-Befehl in Zeile 34 und dem Schreiben der neuen Zwischenergebnisse abgebrochen wird und die Backup-Datei dann leer ist. In diesem Programm ist das äußerst unwahrscheinlich, kann aber in richtigen Programmen mit deutlich größeren zu schreibenden Datenmengen durchaus vorkommen. (Der Rename-Befehl ist übrigens nicht Teil des Fortran-Standards, wird aber von fast allen Compilern unterstützt.) Dann muss nur noch beim Starten des Programms geprüft werden, ob eine Backup-Datei vorhanden ist, und ggf. werden die Zwischenergebnisse von dort eingelesen.

```
1  Program MonteCarloPi
2  Implicit None
3  Integer(8) :: i,j,jstart,treffer
4  Logical :: backup_vorhanden
5  Real(8) :: zufall(2)
6  Integer :: time
7  Integer :: seed(20)
8
9  ! Zufallszahlengenerator initialisieren
10 Call System_Clock(time)
11 seed = time
12 Call Random_Seed(Put=seed)
13
14 ! Normalerweise von vorne anfangen
15 jstart = 1
16 treffer = 0
17
18 ! Wenn es aber eine Backup-Datei gibt,
19 ! dann wird sie gelesen
20 Inquire(File="Backup.dat", Exist=backup_vorhanden)
21 If (backup_vorhanden) Then
22    Open(1000, File="Backup.dat", Status="Old")
23    Read(1000, *) treffer, jstart
```

```
24     Close(1000)
25  End If
26
27  Do j=jstart, 100_8
28     Do i=1, 10000000000_8
29        Call Random_Number(zufall)
30        If (zufall(1)**2+zufall(2)**2 <= 1.0) treffer = treffer + 1
31     End Do
32     ! Alle 10.000.000.000 Durchläufe (~ 3 Minuten) die
33     ! Backup-Datei schreiben
34     Open(1000, File="Backup.tmp", Status="Replace")
35     Write(1000, *) treffer, j+1
36     Close(1000)
37     Call Rename("Backup.tmp", "Backup.dat")
38  End Do
39
40  ! Ergebnis
41  Write (*,*) "Pi␣~=␣", treffer*4.0d0/1000000000000_8
42
43  End Program
```

Die neue Version des Programms können Sie jetzt jederzeit mit Ctrl-C unterbrechen und wieder neu starten. Es können dadurch maximal 3 Minuten Rechenzeit verloren gehen. Dasselbe gilt auch, wenn das Programm im MORFEUS-Grid abgebrochen und auf einen neuen Knoten geschoben wird. Sie müssen HTCondor lediglich noch mitteilen, dass es die bislang geschriebenen Daten (also die Datei `Backup.dat`) auch beim „Herunterwerfen“ des Programms von einem Knoten kopieren soll. Das passiert mit der Angabe

```
when_to_transfer_output = ON_EXIT_OR_EVICT
```

in der Jobbeschreibungsdatei.

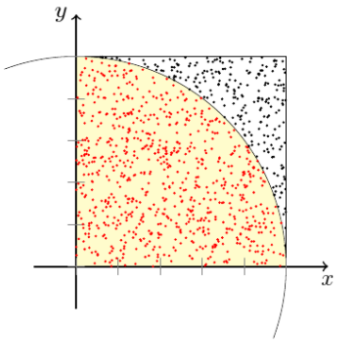


Abbildung 4.1.: Das Monte-Carlo-Verfahren zur Berechnung von π: π entspricht genau der vierfachen Wahrscheinlichkeit dafür, dass ein innerhalb des Quadrats zufällig gewählter Punkt in den Kreis fällt. Quelle: Springob / de.wikipedia.org (CC BY-SA 3.0)

4.11. Zitieren

Die HTCondor-Software, die das Morfeus-Grid erst möglich gemacht hat, ist komplett kostenlos. Die Programmierer bitten jedoch darum, dass HTCondor zitiert wird (siehe Kapitel 7), wenn diese Software zur Berechnung der Ergebnisse eingesetzt wurde. Eine mögliche Art HTCondor zu zitieren ist z.B.:

The calculations were carried out on the computers of the Morfeus Grid at the Westfälische Wilhelms-Universität Münster, with the use of HTCondor[1].

In die Zitatenliste fügt man dann noch das folgende Zitat ein:

[1] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience" Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

Das Zitieren von HTCondor ist freiwillig; es liegt bei Ihnen, ob Sie dem Wunsch der Entwickler nachkommen möchten oder nicht.

5. Beispiele

5.1. Ein einführendes Beispiel

Anhand dieses Beispiels sollten eigentlich schon alle wichtigen Schritte zum Starten eines Jobs auf dem Condor-Pool geklärt werden.

Der Job

Als Beispielprogramm verwenden wir hier ein ganz simples Gruß-Programm, geschrieben in Fortran. Natürlich sollte man in Wirklichkeit nur solche Programme in das MORFEUS-Grid schicken, die viel Rechenzeit benötigen (und auch etwas Sinnvolles tun...).

gruss.f90:

```
1 program gruss
2    Implicit None
3    Character(Len=80) :: name
4    Write (*,*) "Bitte␣geben␣Sie␣Ihren␣Namen␣ein:"
5    Read (*,*) name
6    Write (*,*) "Hallo,␣", trim(name)
7 end
```

Das Programm wurde unter Windows ganz normal mit dem 32-Bit-Compiler kompiliert und liegt jetzt unter `I:\Documents\gruss\gruss.exe` . Von dort aus könnten wir es jetzt wie üblich lokal ausführen. Stattdessen wollen wir aber das MORFEUS-Grid zum Einsatz bringen.

Schritt 1 - Einen geeigneten Rechner zum Abschicken wählen

Zum Abschicken des Jobs kann man jeden Rechner verwenden, der Teil des MORFEUS-Grids ist, also auch den eigenen Arbeitsplatzrechner. Wichtig ist aber, dass der Rechner so lange läuft, bis die Jobs abgeschlossen sind. Schaltet man den Rechner ab, werden auch die von dort abgeschickten Jobs abgebrochen, auch wenn sie auf einem ganz anderen Rechner laufen. Gegebenenfalls sollte man sich also einen geeigneteren Rechner suchen.

Schritt 2 - Eine Eingabedatei schreiben

Falls der Job üblicherweise Eingaben von der Tastatur erwartet, so wie unser Programm oben, schreibt man diese stattdessen in eine Textdatei und legt sie in dasselbe Verzeichnis

wie die Programmdatei. Wir legen also eine Datei `I:\Documents\gruss\eingabe.txt` mit dem Inhalt

```
Fritzchen
```

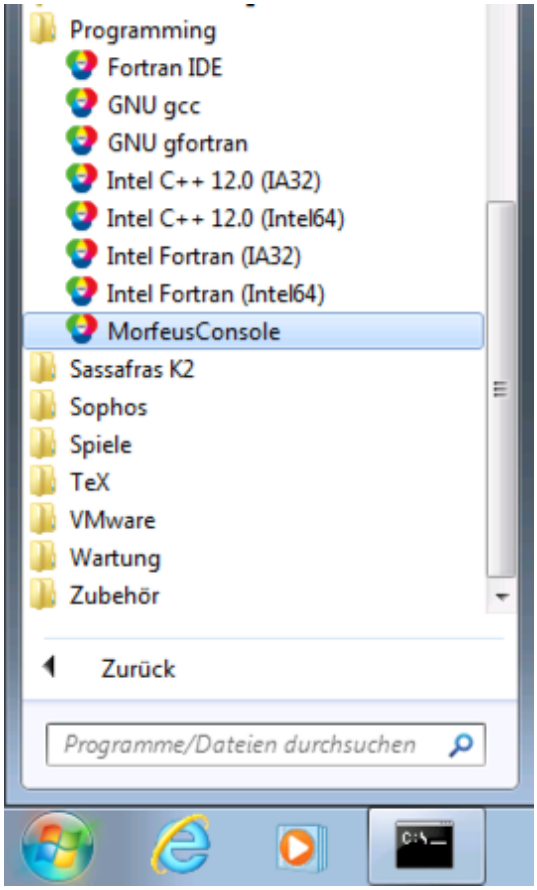
an. Liest das Programm seine Eingaben ohnehin aus einer Datei oder benötigt gar keine Eingaben, fällt dieser Schritt weg. Es ist nicht notwendig, eine leere Datei anzulegen.

Ab hier gibt es zwei Möglichkeiten: Die Nutzung einer graphischen Oberfläche oder die Steuerung von Morfeus über die Kommandozeile.

Möglichkeit 1 (Nutzung von Morfeus über eine graphische Oberfläche)

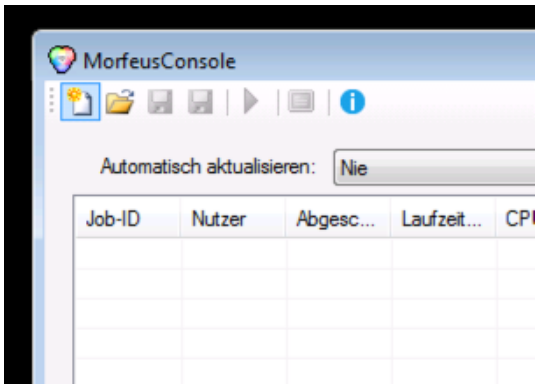
Schritt 3 - MorfeusConsole starten

Die Jobs im MORFEUS-Grid lassen sich mit dem Programm „MorfeusConsole“verwalten. Wir starten das Programm aus dem Start-Menü (Unterpunkt „Programmierung“):



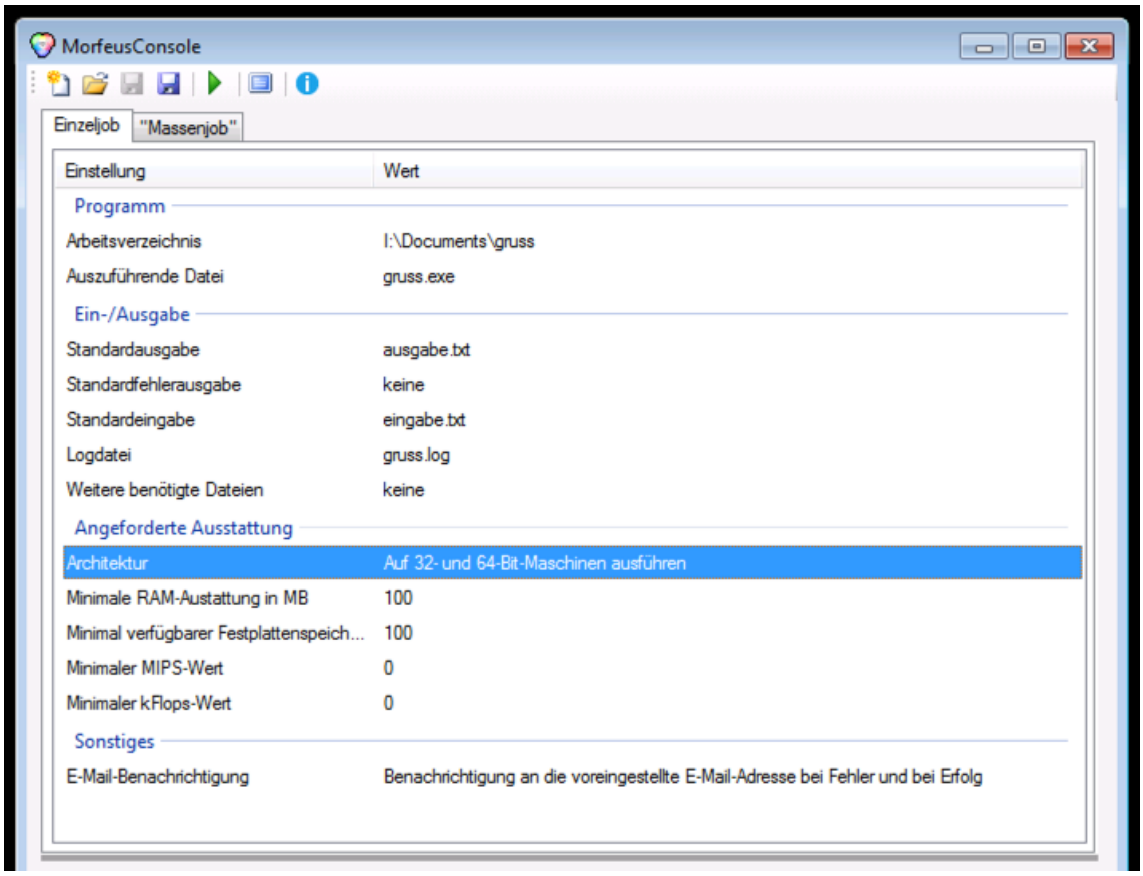
Schritt 4 - Jobbeschreibung anlegen

Wir legen eine neue Jobbeschreibung an und wählen als Arbeitsverzeichnis das Verzeichnis, in dem unser Programm liegt (`I:\Documents\gruss`):



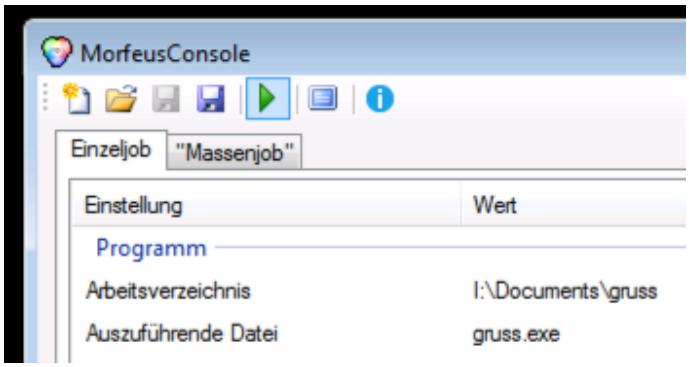
Schritt 5 - Einstellungen vornehmen

Wir müssen jetzt noch einige Einstellungen vornehmen. Der Name der auszuführenden Datei stimmt bereits, aber die Eingabedatei, die wir in Schritt 2 erstellt haben, muss noch angegeben werden. Dazu doppelklicken wir auf die Zeile „Standardeingabe“ und geben dann den Dateinamen „eingabe.txt“ ein bzw. wählen ihn über den Knopf „Auswählen“ aus. Die Ausgabe des Programms soll nicht unter „gruss.out“, sondern unter „ausgabe.txt“ gespeichert werden, also ändern wir auch diesen Eintrag per Doppelklick. Da wir ein 32-Bit-Programm erstellt haben, das sowohl auf 32-, als auch auf 64-Bit-Maschinen lauffähig ist, können wir zum Schluss noch die Einstellung zur Architektur entsprechend ändern. Unser simples Programm hat keine weiteren Anforderungen an Arbeits- oder Plattenspeicher.



Schritt 6 - Job abschicken

Jetzt muss der Job nur noch per Klick auf den „Job starten“-Knopf ins Grid übertragen werden:



Wenn Sie zum ersten Mal von diesem Rechner aus einen Job abschicken oder kürzlich Ihr Kennwort geändert haben, wird die MorfeusConsole Sie noch nach Ihrem Kennwort fragen, da dieses benötigt wird, um die Jobs ins Grid zu übertragen.

Schritt 7 - Status des Jobs überprüfen

Per Klick auf „Jetzt aktualisieren“ können wir jetzt den Status des Jobs überprüfen. Je nachdem, wie sehr das Grid ausgelastet ist, wird der Job sich zuerst im Zustand „I (wartet)“ befinden. Sobald eine passende Maschine für den Job gefunden wurde, wechselt der Status zu „R (läuft)“. Sobald der Job abgeschlossen ist, verschwindet er aus der Liste. Sofern Sie dies nicht in Schritt 5 abgeschaltet haben, erhalten Sie zu diesem Zeitpunkt auch eine E-Mail.

Job-ID	N.	Abgeschickt	Laufzeit (bislang)	CPU-Zeit (bislang)	Status	Priorität	Kommandozeile	Ab
4.0	w.	30.04.201...	00:00:00	00:00:00	R (läuft)	0	gruss.exe	W

Schritt 8 - Ausgabedatei anschauen

Wenn wir jetzt zurück in unser Arbeitsverzeichnis gehen, finden wir dort die Ausgabedatei `ausgabe.txt` mit dem Inhalt:

```
Bitte geben Sie Ihren Namen ein:
Hallo, Fritzchen
```

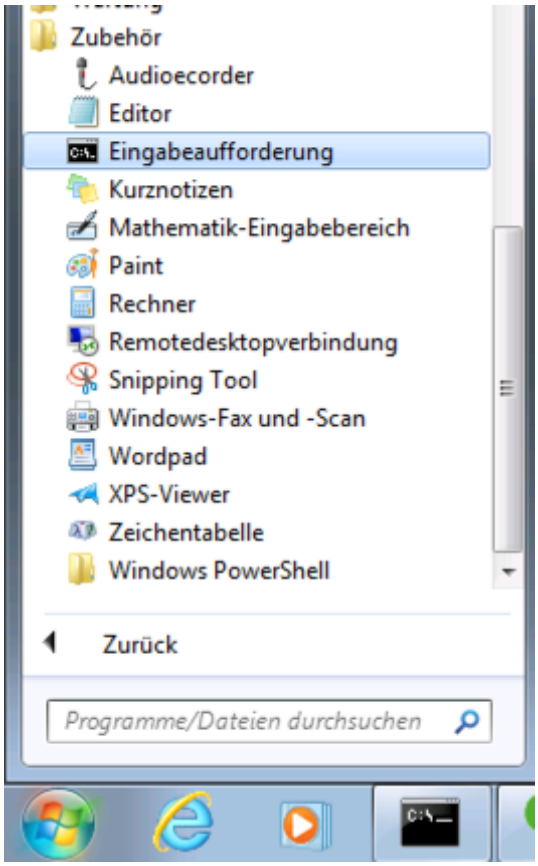
Das war schon alles!

Möglichkeit 2 (Nutzung von Morfeus über die Kommandozeile):

Schritt 1 und 2 s.o.

Schritt 3 - Konsole öffnen

Wir starten zuerst eine Eingabeaufforderung:



Schritt 4 - Jobbeschreibung anlegen

Wir wechseln in das Verzeichnis, in dem unser Programm liegt (`I:\Documents\gruss`) und erstellen dort eine Datei `gruss.sub`:

```
cd I:\Documents\gruss
notepad gruss.sub
```

Schritt 5 - Einstellungen vornehmen

In diese Datei schreiben wir Folgendes:

```
1 universe = vanilla
2 executable = gruss.exe
3 input = eingabe.txt
4 output = ausgabe.txt
5 log = gruss.log
6 notification = Complete
7 request_disk = 102400
8 request_memory = 100
9 requirements =  ((ARCH=="X86_64")||(ARCH=="INTEL"))
```

```
10
11 queue 1
```

Die erste Zeile muss in jeder Jobbeschreibung enthalten sein. Andere „Universen“ werden im MORFEUS-Grid nicht unterstützt. Die zweite Zeile gibt den Namen unseres Programms an. Die dritte Zeile spezifiziert, dass unsere Eingabedatei `eingabe.txt` heißt. Die Ausgaben des Programms sollen in eine Datei `ausgabe.txt` gespeichert werden. Es soll außerdem eine Log-Datei namens `gruss.log` angelegt werden, in der man später Informationen darüber finden kann, wann und wo der Job gerechnet wurde. Zeile 6 aktiviert den Versand von E-Mails, in denen man informiert wird, wenn der Job beendet wurde oder aufgrund eines Fehlers abgebrochen werden musste. Die beiden `request`-Zeilen verlangen für unseren Job mindestens 100MB Plattenspeicher (angegeben in KB) und Arbeitsspeicher (in MB). Da wir ein 32-Bit-Programm erstellt haben, das sowohl auf 32-, als auch auf 64-Bit-Maschinen lauffähig ist, geben wir das in der nächsten Zeile an. Wir möchten nur einen Job rechnen, daher schließen wir die Beschreibungsdatei mit `queue 1` ab.

Schritt 6 - Job abschicken

Jetzt muss der Job nur noch ins Grid übertragen werden. Das funktioniert mit dem Befehl

```
condor_submit gruss.sub
```

Wenn Sie zum ersten Mal von diesem Rechner aus einen Job abschicken oder kürzlich Ihr Kennwort geändert haben, wird es eine Fehlermeldung geben, weil Condor Ihr Kennwort noch nicht kennt. Führen Sie dann zuerst den Befehl

```
condor_store_cred add
```

aus, geben Sie Ihr Kennwort ein, und schicken Sie den Job dann erneut ab.

Schritt 7 - Status des Jobs überprüfen

Rufen Sie jetzt den Befehl

```
condor_q
```

auf, um zu sehen, ob der Job schon läuft:

```
  ID   OWNER    SUBMITTED  RUN_TIME    ST  PRI  SIZE  CMD
  4.0  account  4/30 6:10  0+00:00:00  I   0    57.3  gruss.exe
```

Je nachdem, wie sehr das Grid ausgelastet ist, wird der Job sich zuerst im Zustand (Spalte `ST`) `I` (wartet) befinden. Sobald eine passende Maschine für den Job gefunden wurde, wechselt der Status zu `R` (läuft). Sobald der Job absgeschlossen ist, verschwindet er aus der Liste. Sofern Sie dies nicht in Schritt 5 abgeschaltet haben, erhalten Sie zu diesem Zeitpunkt auch eine E-Mail.

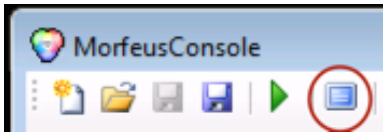
Schritt 8 - Ausgabedatei anschauen

Wenn wir jetzt zurück in unser Arbeitsverzeichnis gehen, finden wir dort die Ausgabedatei `ausgabe.txt` mit dem Inhalt:

```
Bitte geben Sie Ihren Namen ein:
Hallo, Fritzchen
```

Das war schon alles!

Übrigens gibt es in der graphischen Oberfläche „MorfeusConsole“ auch einen Knopf, mit dem Sie sich eine .sub-Datei für die Kommandozeile generieren lassen können:



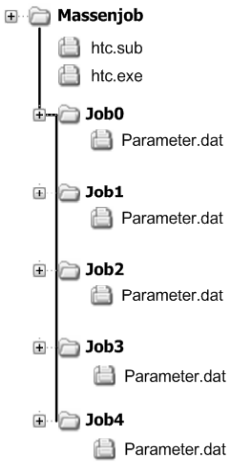
Auf diese Weise können Sie jederzeit schauen, wie die jeweiligen Einstellungen „per Hand“ vorgenommen würden.

5.2. Beispiel eines Massenjobs

Es kommt vor, dass Sie ein Programm haben, welches mit mehreren, verschiedenen Parametersätzen gestartet werden soll. Natürlich könnten Sie jeden Job einzeln mit dem entsprechenden Parametersatz starten, jedoch gibt es dafür einen besseren Weg.

In diesem Beispiel wird ein Programm `htc.exe` benutzt, welches einige Parameter über die Standardeingabe (also üblicherweise die Tastatur) einliest und das Ergebnis auf der Standardausgabe (also normalerweise auf den Bildschirm) ausgibt. Dieses Programm soll für fünf Parametersätze auf dem MORFEUS-Grid gerechnet werden.

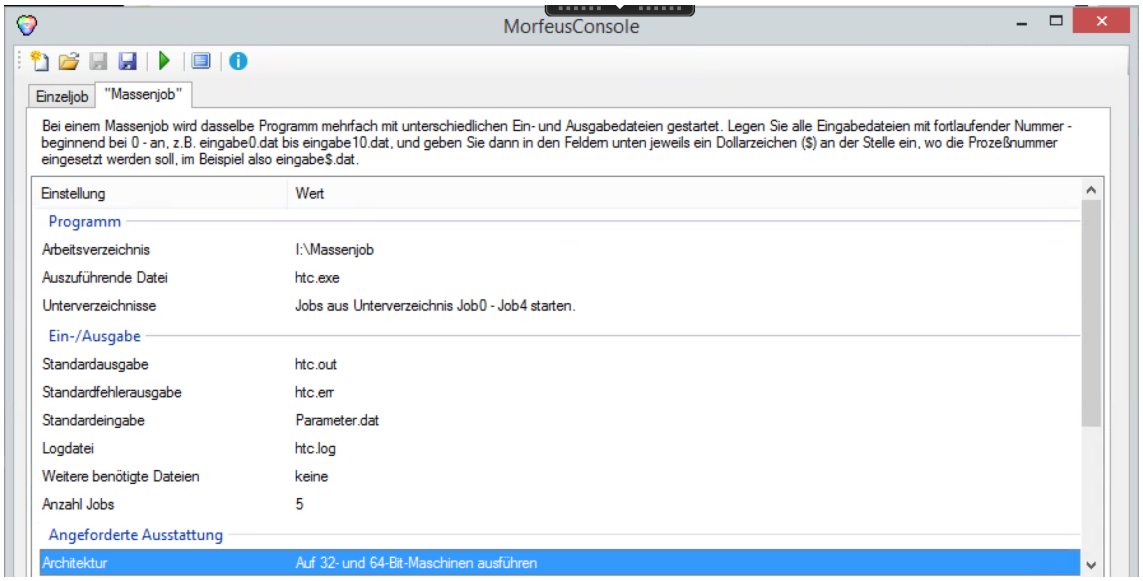
Es gibt mehrere Möglichkeiten, einen Massenjob auszuführen. Eine ist, die Ein- und Ausgabedateien durchnummerieren zu lassen, also die Eingabedateien z.B. `Parameter0.dat`, `Parameter1.dat` usw. zu nennen und die Ausgabedaten dann `htc0.out`, `htc1.out` usw. Eine meistens übersichtlichere Variante arbeitet mit Unterverzeichnissen:



Nachdem ein Arbeitsverzeichnis (hier z.B.: `I:\Massenjob`) angelegt und die Datei `htc.exe` dort hineinkopiert wurde, werden zusätzlich noch fünf Unterverzeichnisse `I:\Massenjob\Job0`,

`I:\Massenjob\Job1`, `I:\Massenjob\Job2`, `I:\Massenjob\Job3` und `I:\Massenjob\Job4` angelegt. In jedes dieser Verzeichnisse wird jeweils eine Datei `Parameter.dat` kopiert, die den entsprechenden Parametersatz enthält.

Danach öffnet man die MorfeusConsole und erstellt eine neue Jobbeschreibung. Das Arbeitsverzeichnis ist `I:\Massenjob`. Es wird die Registerkarte „Massenjob“ ausgewählt, die ausführbare Datei htc.exe eingestellt und beim Punkt „Unterverzeichnisse“ aktiviert, dass für jeden Job ein eigenes Verzeichnis `Job$` verwendet werden soll. Das Dollarzeichen steht dabei für die Nummer des jeweiligen Jobs, angefangen bei 0. Dann brauchen wir nur noch wie bei einem üblichen Einzeljob die Ein- und Ausgabedateien und die Anforderungen des Jobs einzustellen. Schließlich wählen wir noch aus, dass wir (für unsere vorher angelegten Verzeichnisse `Job0-Job4`) 5 Jobs starten möchten.



Da ja in unseren fünf Verzeichnissen unterschiedliche Parameter-Dateien `Parameter.dat` liegen, wird das Programm auf fünf Knoten mit fünf unterschiedlichen Parametersätzen gestartet. Außerdem wird für jeden Job eine separate Log-, Output-, und Error-Datei in dem entsprechenden Unterverzeichnis angelegt.

Wenn man den Massenjob lieber über die Kommandozeile abschicken möchte, geht das mit der folgenden Jobbeschreibungsdatei, die man als `htc.sub` im Verzeichnis `I:\Massenjob` anlegt:

```
1 universe = vanilla
2 executable = htc.exe
3 initialdir = Job$(PROCESS)
4 requirements =  ((ARCH=="X86_64")||(ARCH=="INTEL"))
5 input = Parameter.dat
6 output = htc.out
7 log = htc.log
8 error = htc.err
9 queue 5
```

Die Angaben, die sich hier von einem Einzeljob unterscheiden, sind Zeile 3, in der mit Hilfe der Variablen `$(PROCESS)`, die immer von 0 bis zur Anzahl der Jobs minus 1 läuft,

festgelegt wird, dass das Unterverzeichnis für den ersten Job `Job0`, für den zweiten `Job1` usw. heißt, und die letzte Zeile, in der 5 Jobs auf einmal abgeschickt werden.

6. Links

Zum Schluss dieser Dokumentation sei noch auf weiterführende Informationsquellen verwiesen.

Offizielle Homepage des Morfeus-Projektes

`https://www.uni-muenster.de/NWZ/Angebot/ScientificComputing/Morfeus`

Offizielle Homepage der Terminalserver

`https://www.uni-muenster.de/NWZ/Angebot/NWZatHome/Terminalserver`

Offizielle Homepage des HTCondor-Batchsystems:

`https://research.cs.wisc.edu/htcondor/`

Beiträge der HTCondor-Mailinglist:

`https://research.cs.wisc.edu/htcondor/mail-lists/`

Offizielles Manual zu HTCondor

`http://research.cs.wisc.edu/htcondor/manual/index.html`

7. Lizenz für HTCondor

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an ÄS IS"BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

Definitions.

“License” shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

“Licensor” shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

“Legal Entity” shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, controlmeans (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50outstanding shares, or (iii) beneficial ownership of such entity.

“You” (or “Your”) shall mean an individual or Legal Entity exercising permissions granted by this License.

“Source” form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

“Object” form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

“Work” shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

“Derivative Works” shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

“Contribution” shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, ßubmittedmeans any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving

the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as 'Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a NOTICEtext file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an ÄS IS"BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any

warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Notices

- This product includes software developed by and/or derived from the Globus Project (http://www.globus.org/) to which the U.S. Government retains certain rights. Copyright (c) 1999 University of Chicago and The University of Southern California. All Rights Reserved.
- This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org). Complete conditions and disclaimers for OpenSSL can be found at http://www.openssl.org/source/license.html
- Some distributions of HTCondor include a compiled, unmodified version of the GNU C library. The complete source code to GNU glibc can be found at http://www.gnu.org/software/libc/.
- Some distributions of HTCondor include software developed by the Info-ZIP Project (http://www.info-zip.org/). Complete conditions and disclaimers for Info-ZIP can be found at http://www.info-zip.org/doc/LICENSE
- Some distributions of HTCondor include MAKEMSI software developed by Dennis Bareis (http://dennisbareis.com/makemsi.htm). Complete conditions and disclaimers for MAKEMSI can be found at http://makemsi-manual.dennisbareis.com/disclaimer.htm
- Part of the software embedded in this product is gSOAP software. Portions created by gSOAP are Copyright (C) 2001-2004 Robert A. van Engelen, Genivia inc. All Rights Reserved. THE SOFTWARE IN THIS PRODUCT WAS IN PART PROVIDED BY GENIVIA INC AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A. Anhang

A.1. Begriffe

An dieser Stelle sind die wichtigsten Begriffe des Morfeus-Grids bzw. der HTCondor-Software aufgelistet.

- **Abschicken:** Als Abschicken bezeichnet man den Vorgang, wenn ein Benutzer oder eine Benutzerin einen Job in die Queue des HTCondor-Pools stellt.
- **ClassAds:** Der Begriff ClassAds steht für *Classified Advertisements*, was soviel wie klassifizierte Werbung bedeutet. Jeder Knoten besitzt eine Vielzahl von Attributen (CPU, RAM, Festplattenplatz, ...), die in diesen ClassAds stehen, und sendet sie in regelmäßgien Abständen an den zentralen Manager. Der zentrale Manager kann dadurch entscheiden, auf welchen Knoten ein Job mit bestimmten Anforderungen gestartet werden kann.
- **Condor:** Condor ist der frühere Name der HTCondor-Software.
- **HTCondor-Pool:** Als *HTCondor-Pool* bezeichnet man die Menge aus dem zentralen Manager und allen Knoten. Der HTCondor-Pool der IVV Naturwissenschaften heißt Morfeus-Grid.
- **Evict:** Wenn ein Job *evicted* wurde, so bedeutet das, dass er vom Rechenknoten geworfen wurde und im Normalfall alle bisherigen Rechnungen dieses Jobs verloren sind.[10]
- **Job:** Ein Benutzer oder eine Benutzerin schickt einen Job in die Warteschlange, wenn er oder sie eine Berechnung im HTCondor-Pool durchführen möchte.
- **Kern:** Die meisten heutigen Prozessoren (selbst in Smartphones) bestehen aus mehreren Kernen. Grob kann man sagen, dass jeder Kern eine eigene CPU bildet und damit auch einen eigenen Knoten im HTCondor-Pool.
- **Knoten:** Bei den Knoten handelt es sich um die einzelnen CPUs im HTCondor-Pool.
- **Morfeus-Grid:** siehe HTCondor-Pool.
- **Node:** siehe Knoten.
- **Pool:** Siehe *HTCondor-Pool*.
- **Suspend:** Wenn ein Knoten, auf dem ein Job gestartet wurde, vor Ort benötigt wird, so wird der Job unterbrochen (suspended).

[10] vgl. aber Abschnitt 4.10

- **Unsuspend:** Ein Job bleibt solange *suspended*, bis der ausführende Node vor Ort nicht mehr benutzt wird. Der Job wird dann an der Stelle fortgesetzt, an der er unterbrochen wurde. Diesen Vorgang nennt man unsuspend.
- **Warteschlange:** Wenn ein Benutzer oder eine Benutzerin einen Job abgeschickt hat, wird dieser zunächst in die Warteschlange gestellt. Der zentrale Manager entscheidet dann anhand der Auslastung des Pools und der Priorität des Benutzers bzw. der Benutzerin, wann der Job gestartet wird.
- **Zentraler Manager:** Der zentrale Manager ist der zentrale Server eines jeden HTCondor-Pools. Er kennt alle Ressourcen im Pool und entscheidet, wann und wo ein Job gestartet wird.

A.2. Kommandoreferenz

An dieser Stelle sind die wichtigsten Befehle der HTCondor-Software aufgelistet. Für eine vollständige Kommandoreferenz sei auf das offizielle Handbuch von HTCondor verwiesen.

- **condor_store_cred:** Einmalige Anmeldung auf dem Rechner, von dem der Job abgeschickt wird, siehe Abschnitt 3.2.
- **condor_userprio:** Abfragen der Benutzer/-innen-Prioritäten, siehe Abschnitt 3.10.
- **condor_status:** Aktueller Status des Morfeus-Grids, siehe Abschnitt 3.3.
- **condor_submit:** Jobs in das Morfeus-Grid schicken, siehe Abschnitt 3.4.
- **condor_q:** Abfrage der Warteschlange, siehe Abschnitt 3.7.
- **condor_rm:** Jobs aus der Warteschlange löschen, siehe Abschnitt 3.8.
- **condor_hold:** Jobs anhalten, siehe Abschnitt 3.9.
- **condor_release:** Angehaltene Jobs wieder freigeben, siehe Abschnitt 3.9.