ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Computersysteme

Cluster Administration Tool

Rolf Spuler

Semesterarbeit

9. März 1999

Zuständiger Professor
Prof. Thomas Stricker

Betreuender Assistent
Felix Rauch

Inhaltsverzeichnis

Kapitel 1

Implementationsbeschreibung

1.1 Einleitung

Im Rahmen dieser Semesterarbeit sollte ein graphisches Tool entwickelt werden, mit dessen Hilfe sich ein Benutzer über den aktuellen Zustand eines Clusters von Rechnern informieren kann. Um an diese Informationen zu gelangen, musste man bis anhin auf Programme wie ping, finger etc. zurückgreifen. Dieses Vorgehen ist bei einer grösseren Zahl von Rechnern nicht mehr praktikabel und wird schnell unübersichtlich. Mit dem Cluster Administration Tool sollen diese Informationen mit einem Blick überschaubar werden. Ausserdem sollte das Tool einige einfache, öfters anfallende Administrationsaufgaben vereinfachen, die sonst von Hand auf jedem Rechner einzeln vorgenommen werden mussten.

1.2 Funktionsumfang des Tools

Das Cluster Administration Tool bietet dem Benutzer folgende Informationen und Funktionen an:

Zustandsinformationen

- Rechner an / aus
- Aktuelles Betriebssystem
- Eingeloggte Benutzer und deren Idle-Time
- Auslastung des Rechners

Administrative Funktionen

- Kopieren von Files vom lokalen Rechner auf andere Rechner im Cluster
- Neustart des Rechners mit einem wählbaren Betriebssystem

- Anzeigen von Logfiles

Dieser Teil der Dokumentation soll den Aufbau der Applikation aus der technischen Perspektive beleuchten und mögliche Alternativen zur realisierten Lösung aufzeigen.

1.3 Programmiersprachen

Der grösste Teil des Tools wurde in Tcl/Tk und Tix entwickelt[1]. Tcl *(Tool command language)* ist eine von John Ousterhout entwickelte Skript-Programmiersprache, welche besonders für das effiziente Entwickeln von einfacheren Anwendungen mit graphischer Benutzeroberfläche geeignet ist. Die Tk *(Toolkit)* Bibliothek stellt die Befehle für das Entwickeln des GUIs zur Verfügung, während Tcl die Grundkonstrukte der Sprache beinhaltet. Ausserdem verfügt die Sprache über ausgezeichnete String-Funktionen, welche sich beim Parsen der Ausgaben von UNIX-Kommandos als sehr nützlich erwiesen. Tix ist eine Erweiterung welche die von der Tk-Library zur Verfügung gestellten Widgets um zusätzliche graphische Elemente bereichert.

Bei der Programmierung des Daemon-Programmes auf Windows NT fand MS Visual C++ 5.0 Verwendung.

1.4 Lösungsansätze für die Client-Server Kommunikation

Die wichtigste Entscheidung war bezüglich der Kommunikation der Rechner innerhalb des Clusters zu treffen. Auf welche Weise sammelt das Tool Informationen von den einzelnen Rechnern? Es standen drei verschiedene mögliche Realisierungen zur Wahl:

1. Ausführen von Kommandos auf den Rechnern über eine *remote shell*.
2. Funktionsaufrufe über *remote procedure calls*.
3. Ein Daemon der über *sockets* mit dem Client kommuniziert.

Die Implementation sollte unabhängig vom Betriebssystem sein, daher war der erste Ansatz zum Scheitern verurteilt, ausserdem dürfte diese Version vermutlich relativ langsam sein. Da das Cluster Administration Tool vorwiegend in Tcl/Tk programmiert ist, stellte sich die Realisierung der Kommunikation über *remote procedure calls* ebenfalls als schwierig heraus, da eine solche Funktionalität von Tcl nicht zu Verfügung gestellt wird. Es existiert zwar eine Erweiterung für Tcl, welche dieses Manko behebt, aber es war nicht genau festzustellen, ob die Kommunikation über verschiedene Betriebssysteme hinweg funktioniert. Der Hauptgrund aber wieso diese Lösung nicht gewählt wurde, war, dass Tcl kein API für *system calls* zur Verfügung stellt. Da der Daemon verschiedene Systeminformationen sammelt, stellt sich Tcl in diesem Sektor als nicht besonders geeignet heraus (auf der Unix-Umgebung ist dies weniger schlimm,

[1] Weitere Informationen zur Sprache Tcl / Tk findet man auf dem World Wide Web unter *http://sunscript.sun.com/TclTkCore/*, Tix findet man unter *http://www.xpi.com*

Befehl	Ausgeführte Aktion
getOS	liefert den Namen des aktuellen Betriebssystems
availableOS	liefert eine Liste der installierten Betriebssysteme
getLoad	liefert die Auslastung des Rechners
getUsers	gibt eine Liste der eingeloggten Benutzer zurück
boot:<osIndex>	führt einen Neustart mit einem wählbaren OS aus
watchLog:<lines>	gibt die <lines> letzten Zeilen des Logfiles zurück
copy:<dest>:<pwd>	kopiert ein File an den angegebenen Ort

Tabelle 1.1: Befehlssatz der Daemons

da man durch Ausführen von Standart-Kommandos in den meisten Fällen auch an die benötigte Information kommt).

Aus diesen Gründen fiel die Entscheidung schliesslich auf die Implementation eines eigenen Daemon, welcher über eine Socket-Verbindung mit dem Client-Programm kommuniziert. Dieser Daemon wurde auf der Linux-Umgebung in Tcl und auf Windows NT in C programmiert. Für den Linux-Daemon wurde Tcl gewählt, weil der Umgang mit Sockets und das Parsen von Text einfacher zu realisieren war.

1.5 Kommunikation zwischen Client und Daemon

Der Cluster Administration Tool Client kommuniziert mit den Daemons über die Socket-Schnittstelle. Will der Client von einem Rechner eine Information anfordern, so schickt er dem Daemon, der auf diesem Rechner läuft einen Befehl, welchen der Daemon mit der gewünschten Information beantwortet. Übertragen werden dabei sowohl die Befehle als auch die Antworten in Klartext. Der Befehlssatz den der Windows NT Daemon versteht, ist gegenüber der Linux-Version etwas eingeschränkt. Die Funktionen für das Kopieren von Files und das Anzeigen des Logfiles existieren hier nicht. Ausserdem kann der NT Daemon zur Zeit noch keine *idle time* für die Benutzer liefern. Tabelle 1.1 zeigt eine Übersicht über den Befehlssatz der Daemons. Die Implementation der einzelnen Kommandos wird in Abschnitt 1.6 genauer beschrieben.

Der Ablauf einer Refresh-Phase ist recht einfach und lässt sich am einfachsten verstehen, indem man die involvierten Programmteile auf der Client- und Serverseite betrachtet. In Tabelle 1.2 ist die Refresh-Routine auf der Client-Seite skizziert. Tabelle 1.3 zeigt das Programmgerüst des Daemons.

1.6 Implementation der Daemons

In diesem Abschnitt sollen die beiden Daemons auf Linux und Windows NT etwas genauer betrachtet werden. Dazu wird im folgenden die Implementation der wichtigsten Kommandos aus Tabelle 1.1 beschrieben.

```
# client refresh pseudo algorithm
#
foreach host in HOSTS {
   connect to host
   if connection is up {
      send getOS, save result
      send getLoad, save result
      send getUsers, save result
   }
   disconnect from host
}
update client window
```

Tabelle 1.2: Refreshing Pseudo-Code

```
# daemon pseudo code
#
forever do {
   listen on socket for incoming connect request
   connect to client
   wait for command
   <cmd, argv> := parseCommand
   switch cmd {
      getOS    : call proc getOS
      getUsers : call proc getUsers
      ...
      copy     : call proc copy (argv[1],argv[2])
      ...
   }
}
```

Tabelle 1.3: Daemon Programmgerüst

1.6.1 Linux

Dieser Daemon wurde zum grössten Teil in Tcl geschrieben. Im Nachhinein musste ich allerdings festellen, dass gewisse systemnahe Funktionen nur über den Umweg eines C-Programmes zu realisieren waren. Aus diesem Grund besteht der Linux-Daemon auch aus mehreren Files, aus dem eigentlichen Daemon-Programm und drei Hilfsfunktionen, die in C programmiert wurden. Eine Implementation in C hätte den Vorteil gehabt, dass der Daemon aus einem Stück bestanden hätte.

1.6.1.1 getOS, availableOS

Diese beiden Kommandos geben die im Konfigurations-File definierten Betriebssystemnamen zurück. Das Konfigurationsfile wird gelesen, wenn der Daemon gestartet wird.

1.6.1.2 getLoad

Diese Prozedur liefert die Auslastung des Rechners zurück, indem das Unix-Kommando `uptime` ausgeführt wird und aus dessen Ausgabe der gewünschte Wert herausgefiltert wird. Alternativ hätte man diesen Wert auch aus dem */proc* File-System lesen können. Hier stellt sich die Frage, welche Lösung besser portierbar ist. Das */proc* File-System existiert, soweit ich weiss, nur unter Linux, während der Befehl uptime auf allen Unix-Umgebungen existiert. Andererseits sind die Ausgaben bzw. Optionen der Unix-Befehle auch nicht einheitlich gelöst, weshalb das Parsing auf einem anderen Unix-Betriebssystem versagen kann.

1.6.1.3 getUsers

Die Prozedur getUsers gibt die Benutzernamen sowie die *idle time* für jeden Benutzer zurück, der auf dem Rechner eingeloggt ist. Hat ein Benutzer mehrere Terminals geöffnet, so wird die *idle time* des Terminals zurückgegeben, welches zuletzt benutzt wurde. Ich versuchte zuerst, diese Funktion mit Hilfe des Unix-Kommandos `w` zu implementieren, musste aber feststellen, dass die Ausgabe auf unterschiedlichen Systemen nicht einheitlich ist. Ausserdem war das Zeitformat der *idle time* nicht besonders aussagekräftig. Aus diesen Gründen entschied ich mich eine eigenes Programm `userinfo` zu schreiben, welches die Benutzernamen zusammen mit den entsprechenden *idle times* ermittelt.

1.6.1.4 boot:<osIndex>

Dieses Kommando führt einen Neustart des Rechners durch. Durch das Argument <osIndex> wird das Betriebssystem bestimmt, welches gestartet werden soll (der Index bezieht sich auf die in `catd.conf` definierte Variable `OS_AVAIL`). Das Booten funktioniert nur in Verbindung mit *System Commander* als Bootmanager, dessen Konfigurationsfile durch den Daemon direkt verändert wird. Leider liess sich das Format dieses Files nicht herausfinden. Ich versuchte deshalb zu ermitteln, was sich in diesem File ändert, wenn man die verschiedenen Betriebssysteme als Default-Betriebssystem

konfiguriert. Es stellte sich heraus, dass die Änderungen immer an der gleichen Stelle im File auftreten.

Die Veränderungen am Konfigurationsfile (`syscmndr.sys`) werden durch das Programm `scpatch` vorgenommen, welches das vom Benutzer gewählten Betriebssystem in das *System Commander* File einträgt. Diese Lösung ist zwar nicht gerade besonders elegant, aber sie scheint zu funktionieren.

1.6.1.5 watchLog:<lines>

Die Prozedur watchLog liest mittels des `tail` Kommandos die im Argument spezifizierte Anzahl von Zeilen am Ende des Logfiles `/var/log/messages`.

1.6.1.6 copy:<dest>:<pwd>

Diese Funktion ermöglich das Kopieren eines Files vom Rechner auf dem der Client läuft auf einen Rechner auf dem dieser Daemon läuft. Die Funktion ist durch ein Passwort geschützt. Die Überprüfung des Passwortes erfolgt über das Programm `checkpwd`, welches das vom Benutzer eingegebene Passwort mit dem Passwort für den *rcopy*-User vergleicht. Damit auch Programme aus lokalen Verzeichnissen des Client-Rechners kopiert werden können, wird ein File in einem ersten Schritt auf dem Client-Rechner in ein temporäres File kopiert, welches über NFS[2] zugreifbar ist. In einem zweiten Schritt wird das Temporär-File dann vom Daemon an den gewünschten Zielort kopiert.

1.6.2 Windows NT

Der Daemon für Windows NT wurde mit MS Visual C++ entwickelt. Der Begriff Daemon stammt ursprünglich aus der Unix-Welt, in Windows NT bezeichnet man solche Programme als Service[3]. Die Idee, die hinter diesen Begriffen steckt, ist aber die selbe — ein Programm welches permanent im Hintergrund irgend eine Arbeit verrichtet, solange das System läuft. Ein Service ist aufwendiger zu Programmieren als ein Unix-Daemon, da er sich zuerst im System registrieren lassen muss, bevor er gestartet werden kann.

Die Implementation des NT Services bietet keine Funktionen watchLog und copy und gibt zum Teil nicht die gleichen Angaben zurück als der Linux-Daemon.

1.6.2.1 getOS, availableOS , boot:<os Index>

Diese Prozeduren entprechen denjenigen des Linux-Daemon

1.6.2.2 getLoad

Die Funktion getLoad liefert die Auslastung des Prozessors in Prozent. Zwar existieren verschiedene Performance-Daten, die in Windows NT abgefragt werden können,

[2] Network File System

[3] Der Einfachheit halber wird in diesem Dokument der Begriff Daemon für beide Betriebssysteme benutzt.

aber ich konnte keine mit der Load in Unix vergleichbare Metrik finden. Das Ermitteln der Prozessorauslastung erfolgt mit Hilfe der Performance Data Helper Library (`pdh.dll`), welche man auf der Microsoft Win32 SDK CD findet.

1.6.2.3 getUsers

Diese Prozedur gibt die Namen der eingeloggten Benutzer zurück. Die Benutzernamen werden über die Systemfunktion `NetWkstaUserEnum` ermittelt. Zur Zeit existiert noch keine Ausgabe der *idle time*, diese Funktion konnte aus Zeitmangel nicht mehr realisiert werden.

1.7 Erweitern der Daemons

Die beiden Daemons lassen sich ohne allzu grossen Aufwand um weitere Funktionalität erweitern. Es müssen aber einige Besonderheiten bei der Kommunikation mit dem Client beachtet werden. Will man dem Daemon eine neue Funktion hinzufügen, so schreibt man eine Prozedur deren Schnittstelle folgendermassen aussieht:

```
proc NewProcedure {sock argument1 argument2 ...}
```

In der Prozedur `doCommand` muss ein entsprechender Aufruf ergänzt werden. Wie man das macht, ist im Source-Code dokumentiert. Das Kommando kann dann vom Clienten durch

```
call NewProcedure:argument1:argument2:...
```

aufgerufen werden. Es ist zu beachten, dass die Argumente keine Doppelpunkte enthalten dürfen , da diese als Trennzeichen zwischen den einzelnen Argumenten dienen.

Der Client erwartet vom Daemon nach dem Senden eines Kommandos eine Antwort. Dies kann entweder eine angeforderte Information, eine Fehlermeldung oder eine Bestätigung für die Ausführung einer Operation sein. Auf jeden Fall muss der Daemon bei jeder Funktion eine Antwort senden, da der Client solange blockiert, bis er eine Antwort erhält (bzw. bis die Socket-Verbindung zusammenbricht).

Kapitel 2

Installation und Konfiguration

Auf den folgenden Seiten wird beschrieben, wie man das Cluster Administration Tool installiert, welche Sytemvoraussetzungen erfüllt sein müssen und wie man die einzelnen Komponenten an eine gegebene Systemumgebung anpasst.

2.1 Programm-Files

Das Tool besteht aus mehreren Files, welche zu drei verschiedenen Programmen gehören. Tabelle 2.1 zeigt eine Uebersicht aller Files und deren Zugehörigkeit zu den einzelnen Programmen.

2.2 Cluster Administration Tool Client

2.2.1 Installation

Das Client-Programm wurde vollständig in Tcl geschrieben und setzt voraus, dass Tcl 7.6 / Tk 4.2 sowie Tix 4.1 auf dem Rechner installiert ist. Das Programm wurde für Linux geschrieben, sollte aber ohne grosse Anpassungen auch auf anderen Betriebssystemen laufen, für welche die oben genannten Programmiersprachen verfügbar sind.

Der Client besteht aus den Files `CAT, main.tcl und remote.tcl`, welche sich im gleichen Verzeichnis befinden müssen. Die Programmeinstellungen werden im File `.catrc` im Home-Verzeichnis des Benutzers gespeichert. Dieses File wird erzeugt, wenn das Tool zum ersten mal gestartet wird.

2.2.2 Konfiguration

Grundsätzlich gibt es zwei verschiedene Möglichkeiten, das Tool an die eigenen Bedürfnisse anzupassen. Einige der Einstellungen lassen sich im Settings-Fenster über die graphische Benutzeroberfläche verändern, die weniger oft gebrauchten Einstellungen muss man durch Editieren des Files `.catrc` ändern. Dieses File ist ein normales

CAT Client	
CAT	Graphische Benutzeroberfläche
main.tcl	Event Handling Routinen
remote.tcl	Kommunikations-Prozeduren
.catrc	Client Konfigurations-File
Linux Daemon	
catd	Daemon Programm
userinfo	liefert Benutzerinformationen
scpatch	verändert syscmndr.sys
checkpwd	Passwort-Check für das Kopieren
catd.conf	Daemon Konfigurations-File
NT Daemon	
catd.exe	Daemon Programm
pdh.dll	Performance Data Helper Library
catd.cfg	Daemon Konfigurations-File

Tabelle 2.1: Liste aller Files

Tcl-Skript, daher müssen sich die Einträge darin an die Syntax von Tcl halten. Eine Variable wird mit `set varName value` gesetzt. Der Variable `HOSTS` muss eine Liste zugewiesen werden. Listen werden in geschweiften Klammern angegeben, wobei die einzelnen Listenelemente durch Leerzeichen getrennt werden.

```
Beispiel: set HOSTS {hostname1 hostname2 hostname3}
```

Tabelle 2.2 erklärt alle konfigurierbaren Variablen

Variable	Bedeutung
HOSTS	Hostnamen der Rechner im Cluster
COLOR_ALIVE	Farbe für Hosts deren Daemon läuft
COLOR_NO_RESPONSE	Farbe wenn der Daemon nicht läuft
COLOR_DEAD	Farbe falls der Rechner nicht erreichbar
RFSH_DELAY	Zeit zwischen Refreshs in sek.
RFSH_ENABLE	Autorefresh yes/no
HOSTNAME_COL_WIDTH	Breite der Spalte Hostname
LOAD_COL_WIDTH	Breite der Spalte Load
OS_COL_WIDTH	Breite der Spalte OS
USERS_COL_WIDTH	Breite der Spalte Users

Tabelle 2.2: Einstellungen im File `.catrc`

2.3 Daemons

2.3.1 Installation

Zur Zeit existieren zwei verschiedene Daemons, einer für Linux und einer für Windows NT.

2.3.1.1 Linux

Der Linux-Daemon besteht aus den Programmen `catd, userinfo, scpatch und checkpwd`, welche sich im selben Verzeichnis befinden müssen, sowie einem File `catd.conf`, welches standartmässig aus dem Verzeichnis `/etc` gelesen wird[1]. Damit der Daemon automatisch vom System gestartet werden kann, muss ein Startfile im Verzeichnis `/etc/rc.d` erstellt werden. Der Daemon muss mit Root-Privilegien laufen, da sonst einige Funktionen (Booten, Kopieren von Files) nicht möglich sind.

2.3.1.2 Windows NT

Der Daemon `catd.exe` benötigt die Performance Data Helper Library `pdh.dll`. Diese Bibliothek enthält Funktionen, die das Auslesen von Performance-Daten vereinfachen. Der Daemon ist als NT Service implementiert und wird folgendermassen installiert:

- In der MS-DOS Eingabeaufforderung wird der Service mit `catd -install` installiert (Entfernt wird er analog mit `catd -remove`). Der Service ist jetzt installiert, dem System muss aber noch mitgeteilt werden, dass es den Service jeweils beim Aufstarten aktivieren soll.
- Dazu öffnet man das *Service Control Panel* (Arbeitsplatz -> Control Panel -> Services) und aktiviert den automatischen Startup für den *Cluster Admininstration Tool Service* (Abbildung 2.1). Auch hier muss darauf geachtet werden, dass der Service mit Administrator-Rechten gestartet wird (Standarteinstellung).

Fehlermeldungen des Services werden im EventLog von Windows NT festgehalten und lassen sich über den EventViewer abfragen.

2.3.2 Konfiguration

Der Cluster Administration Tool Daemon benötigt zwei Angaben über die Umgebung, in welcher er seine Arbeit verrichtet. Erstens muss er wissen, auf welchem Betriebssystem er läuft und zweitens welche Betriebssysteme auf dem Rechner installiert sind[2]. Diese Information wird für jeden Daemon durch ein Konfigurations-File festgelegt, in welchem zwei Variablen gesetzt werden:

OS_NAME Der Name des Betriebssystems, auf welchem der Daemon läuft.

[1] Es ist auch möglich, das Konfigurations-File als Argument beim Start von `catd` anzugeben

[2] Diese Konfiguration wird im Zusammenhang mit der Boot-Funktion benötigt, welche voraussetzt, dass auf den Rechnern *System Commander* als Bootmanager installiert ist.

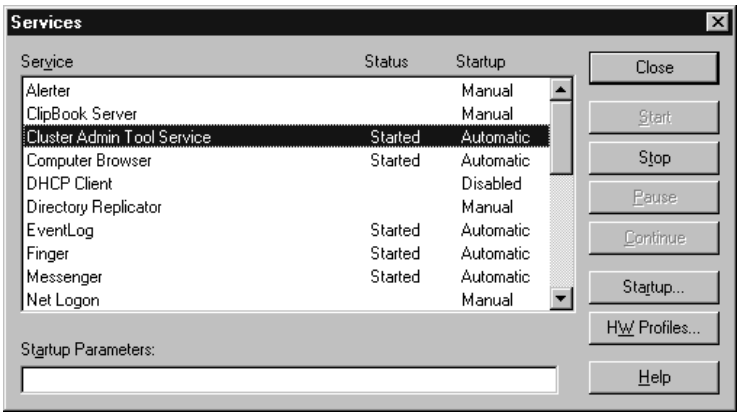

Abbildung 2.1: Das Service Control Panel

OS_AVAIL Eine Liste der Betriebssysteme, welche auf dem Rechner installiert sind. Die Namen müssen in der gleichen Reihenfolge aufgelistet werden, wie sie im *System Commander* Menu erscheinen.

2.3.2.1 Linux

Das Konfigurationsfile hat den Namen `catd.conf` und befindet sich im Normalfall im `/etc` Verzeichnis. Das Konfigurationsfile ist ein Tcl-Skript und muss sich an die bereits erwähnte Syntax halten. Es könnte zum Beispiel folgendermassen aussehen:

```
# Cluster Administration Tool Daemon Configuration
# Linux example
#
set OS_NAME Linux
set OS_AVAIL {Linux Solaris “Windows NT”}
```

2.3.2.2 Windows NT

Das Konfigurationsfile mit dem Namen `CATD.CFG` muss sich im gleichen Verzeichnis befinden, in welchem System Commander installiert wurde. Hier sieht das Format der Variablenzuweisung folgendermassen aus:

```
varName=value
```

Falls `value` eine Liste von Werten ist, werden die einzelnen Elemente durch Leerzeichen voneinander getrennt. Das oben aufgeführte Beispiel würde hier wie folgt aussehen:

```
# Cluster Administration Tool Daemon Configuration
```

```
# Windows NT example
#
OS_NAME=”Windows NT”
OS_AVAIL=Linux Solaris “Windows NT”
```

Kapitel 3

Bedienungsanleitung

In diesem Kapitel soll die Bedienung des Cluster Administration Tools erläutert werden. Das Programm verfügt über eine grafische Benutzeroberfläche und sollte daher recht einfach zu handhaben sein. Man startet das Tool in der Shell mit `CAT`.

3.1 Das Tool Interface

Auf dem Bildschirm erscheint das Hauptfenster des Tools. Es zeigt verschiedene Informationen, die über den Zustand der einzelnen Rechner im Cluster Auskunft geben. In der rechten unteren Ecke des Fensters sieht man einen Button der mit > bezeichnet ist. Damit lässt sich das Fenster in eine erweiterte Ansicht umschalten, in welcher zusätzliche Informationen dargestellt werden. Diese Ansicht ist in Abbildung 3.1 dargestellt. Die ”zugeklappte” Version hat den Vorteil, dass sie weniger Platz auf dem Desktop benötigt und bei dieser Ansicht noch genug Freiraum für andere Fenster vorhanden ist. Mit der rechten Maustaste lässt sich jederzeit ein Menu aufrufen, das Zugang zu den restlichen Funktionen des Tools bietet. Die einzelnen Spalten im Hauptfenster haben folgende Bedeutung:

3.1.1 Hostname

Der Hostname bezeichnet den Rechner, von welchem die Informationen in der entsprechenden Zeile stammen. Wie man die Hostnamen der zu überwachenden Rechner konfiguriert, wird im Abschnitt 2.2 der Installationsanleitung beschrieben. Ein Host kann in einem von drei Zuständen sein, die durch unterschiedliche Farben gekennzeichnet sind.

rot Der Host mit diesem Namen ist nicht erreichbar. Mögliche Ursachen dafür sind, dass der Rechner ausgeschaltet wurde, oder dass der Rechner zwar läuft, aber keine Netzverbindung aufgebaut werden konnte.[1]

orange Es kann eine Netzwerkverbindung mit dem Host etabliert werden, aber der *Cluster Administration Tool Daemon* auf diesem Rechner antwortet bzw. läuft

Abbildung 3.1: Cluster Administration Tool Hauptfenster

nicht.

blau Der Normalzustand. Der Rechner läuft und der CAT Daemon antwortet auf eingehende Requests. In diesem Zustand zeigen auch die restlichen Felder in der Zeile Informationen an.

3.1.2 Load

Aus dieser Spalte lässt sich ersehen, wie die einzelnen Rechner ausgelastet sind. Wie diese Auslastung zu deuten ist, hängt vom Betriebssystem ab, welches auf dem jeweiligen Computer läuft. Es existieren zwei unterschiedliche Metriken für Windows NT und Linux. Unter Linux besagt die Auslastung, wieviele Prozesse (gemittelt über die letzte Minute) darauf warten, vom Prozessor bedient zu werden. Dieser Wert gibt recht aussagekräftig darüber Auskunft, wie hoch der Rechner ausgelastet ist. Auf der Windows NT Seite wird die prozentuale Auslastung des Prozessors gemessen. Dieser Wert ist meistens relativ hoch, wenn jemand am Rechner arbeitet, da auch nur ein simples Verschieben eines Fensters den Prozessor stark beansprucht. Anhand der Load soll der Benutzer festellen können, welche Rechner zur Zeit gerade stark belastet sind und wo noch freie Rechenkapazitäten zur Verfügung stehen.

Die verschiedenen Metriken werden in unterschiedlichen Farben dargestellt und als absolute bzw. prozentuale Werte beschriftet. Unter Linux wird die Auslastung mit einem blauen, unter Windows NT mit einem gelben Balken dargestellt.

3.1.3 Operating System (OS)

Diese Spalte zeigt an, welches Betriebssystem gerade auf den jeweiligen Rechnern läuft. Eine Unterscheidung ob Windows NT oder ein Unix-OS läuft, lässt sich auch im kleineren Hauptfenster anhand der Farben der Load-Anzeige erkennen.

[1]Die Erreichbarkeit eines Hosts wird mittels *ping* getestet

3.1.4 Users

Die Users-Spalte gibt darüber Auskunft, welche Benutzer auf welchen Rechnern eingeloggt sind. Angezeigt werden deren Login-Namen. Die Zahl hinter jedem Namen ist die *idle time* des entsprechenden Benutzers und besagt, wie lange er nichts mehr auf dem Rechner über die Tastatur eingegeben hat[2].

3.2 Refreshing

Es existieren zwei Möglichkeiten, wie die Zustandsinformation des Clusters auf den aktuellen Stand gebracht werden kann. Eine Möglichkeit besteht darin, dass man ein Update manuell ausführt, indem man den Refresh-Button am unteren Fensterrand betätigt. Als zweite Möglichkeit bietet das Tool ein automatisches Refreshing an, welches in periodischen Zeitabständen ein Update ausführt. Das Refreshing erfolgt immer über alle Rechner im Cluster in zeitlich sequentieller Folge. Von welchem Rechner im Augenblick gerade Informationen geholt werden, lässt sich daran erkennen, dass der entsprechende Hostname vertieft dargestellt wird.

3.3 Administrations-Fenster

Das Administrationsfenster ist in Abbildung 3.2 dargestellt. Mit den Registerkarten auf der rechten Seite lassen sich die verschiedenen Administrationsfunktionen anwählen. In der Liste der Hostnamen auf der linken Seite lässt sich jeweils festlegen, auf welche Rechner sich die Operation beziehen soll. Es können auch mehrere Rechner selektiert werden. Die Operation wird dann auf jedem dieser Rechner ausgeführt. Diese Multiselektionen werden durch Selektion mit der Maus bei gleichzeitigem Drücken der Ctrl- bzw. Shift-Taste ermöglicht. Mit Hilfe der Shift-Taste lässt sich ein Bereich selektieren, während mit der Ctrl-Taste einzelne Rechner selektiert bzw. deselektiert werden können.

3.3.1 Booten

Die Boot-Funktion bietet dem Benutzer die Möglichkeit, einen oder mehrere Rechner im Cluster neu aufzustarten. Welches der auf dem Zielrechner installierten Betriebssysteme gebootet werden soll, lässt sich dabei wählen. Für den Fall, dass mehrere Rechner selektiert wurden, kann man natürlich nur unter Betriebssystemen wählen, welche auf allen diesen Rechnern installiert sind[3].

3.3.2 Kopieren von Files

Im der Copy-Registerkarte lassen sich Files vom lokalen Rechner auf andere Rechner im Cluster kopieren. Das Ausführen dieser Operation ist nur nach Eingabe eines

[2] Die Ausgabe der idle time ist zur Zeit nur im Linux-Daemon implementiert.

[3] Damit diese Multiselektion funktioniert, muss darauf geachtet werden, dass bei der Konfiguration die Namen der installierten Betriebssysteme auf allen Rechnern exakt übereinstimmen.

Abbildung 3.2: Administrations-Fenster

Passwortes möglich. Die Schreiboperationen auf den Zielrechnern werden im Console-Fenster bestätigt.

3.3.3 Log-File anzeigen

Mit Hilfe der WatchLog-Registerkarte lässt sich unter Linux das Logfile `messages` im Verzeichnis `/var/log` ansehen. Es werden die mit dem Schieberegler eingestellte Anzahl Zeilen am File-Ende angezeigt.

3.4 Console

Dieses Fenster protokolliert aufgetretene Fehler (sowohl lokal als auch beim Abarbeiten eines Kommandos auf dem Zielhost) oder informiert über ausgeführte Aktionen auf anderen Rechnern. Es ist von Vorteil dieses Fenster während dem Ausführen von Administrationsoperationen geöffnet zu haben.

3.5 Settings

Im Settings-Fenster, welches in Abbildung 3.3 zu sehen ist, lassen sich einige Einstellungen zum Refreshing und dem Erscheinungsbild der Applikation machen. Die Einstellung der Schriftart erfolgt über das Programm `xfontsel`[4], welches aufgerufen wird, wenn man den *Select Font* Button drückt. Damit die gewählte Schriftart aktiviert wird, muss das Tool allerdings neu aufgestartet werden. Auf der rechten Seite des Fensters lässt sich die Breite der verschiedenen Spalten des Hauptfensters verändern. Von den Fonteinstellungen abgesehen, werden alle anderen Einstellungen aktiv, sobald der *Apply*-Button gedrückt wird.

[4]Nähere Informationen zu `xfonstsel` stehen in der manual page: `man xfonstsel`

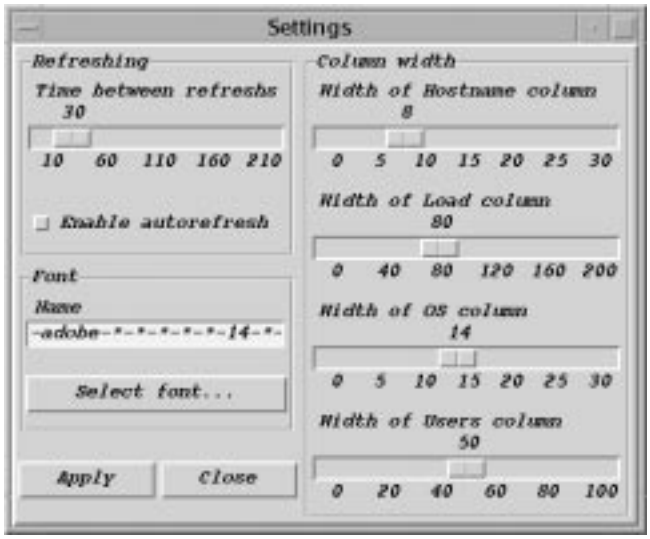

Abbildung 3.3: Das Settings-Fenster