CHALMERS | GÖTEBORGS UNIVERSITET

Chalmers oanvända datorkraft

- Distribuering av arbete och energihantering med HTCondor

Kandidatarbete inom Data- och informationsteknik

DANIEL BERGQVIST
MARCUS KALANDER
OLIVER ANDERSSON
PONTUS JOHANSSON BERG
RURIK HÖGFELDT

Chalmers tekniska högskola
Göteborgs universitet
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, Juni 2013

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.
The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Chalmers unused computing power
- Distribution of work and power management with HTCondor

Daniel Bergqvist, Marcus Kalander, Oliver Andersson, Pontus Johansson Berg, Rurik Högfeldt

© Daniel Bergqvist, June 2013.
© Marcus Kalander, June 2013.
© Oliver Andersson, June 2013.
© Pontus Johansson Berg, June 2013.
© Rurik Högfeldt, June 2013.

Examiner: Arne Linde

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: The condor is an animal strongly associated with HTCondor due to their names.
It’s used in the HTCondor logotype.

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

Förord

Rapporten är ett kandidatarbete för Data- och informationsteknik på Chalmers tekniska högskola och har genomförts under våren 2013. Den ger ett förslag på hur Chalmers kan använda sina datorer på ett mer effektivt och miljövänligt sätt.

Vi vill tacka vår handledare, Tomas Olovsson, för den tid han lagt ner för vägledning och diskussion, Arne Linde och IT-administrationen på Data och IT institutionen som svarat på våra frågor. Vi vill även tacka för den hjälp som erhållits från fackspråk.

Sammanfattning

På Chalmers tekniska högskola finns idag många datorer som står på dygnet runt samtidigt som det finns ett behov av mer beräkningskraft för forskare som vill utföra tunga beräkningar. Vi har därför genomfört en undersökning om datorerna kan användas för beräkningskraft och i andra hand, när det inte finns något arbete, försättas i energisparläge.

Vi gjorde en noggrann studie för att jämföra olika distribuerande system där HTCondor valdes som det bästa att implementera. HTCondor är ett utmärkt system för opportunistisk användning av datorkraft, dvs. för utnyttjande av datorer som ingen annan använder för tillfället. Systemet används på flera universitet världen över med goda resultat och passar utmärkt på Chalmers där det finns behov av ett sådant system.

Vår implementation visar att HTCondor väl kan hantera outnyttjad datorkraft. HTCondor klarar också att hantera de flesta programmeringsspråken som kan tänkas köras. Systemet kan även försätta datorer i energisparläge när de inte behövs och starta dem vid behov. För att förenkla för de som använder HTCondor skapade vi ett användargränssnitt, som innehåller de funktioner som krävs för att utnyttja systemet, eftersom inget bra sådant fanns att tillgå.

Det finns tveksamheter från Chalmers gällande att stänga av datorer, de hävdar att datorerna alltid måste vara tillgängliga och att problem kan uppstå vid uppstart av datorer, exempelvis att program eller liknande inte startar. Dock visar vår undersökning att det inte är något större problem med att stänga av datorer, vilket vi visar i en energistudie där avstängningsmetoden strömsparläge utses som bästa alternativet för Chalmers.

Abstract

Chalmers University of Technology today have numerous computers which are never powered down, at the same time there is a need for more computation power for researchers. Hence, we have investigated the possibility to use the computers for computing power and secondly, if there is no work to be done, to put them into power saving mode.

We have made a thorough study where we compared different distributing systems and in the end HTCondor was chosen as the best to implement. HTCondor is an excellent system for opportunistic use of computing power, i.e. make use of computers that no one else is currently using. The system is used at several universities around the world with good results and would be excellent at Chalmers, where there is a need for such a system.

Our implementation shows that HTCondor is well capable of handling unused computing power. HTCondor can handle most file types that may be run on the system. The system can also put computers in sleep mode when they are not in use and turn them on when needed. To simplify for the users of HTCondor we have created a user interface that have all necessary functions required to make use of the system as no good such interface was available.

There are doubts from Chalmers regarding the idea of shutting down computers, they argue that the computers always must be available and that problems may arise when turning them on, for example programs that do not start. Our investigation shows that there are no bigger problems with turning on and off computers, which is shown in a energy study in which sleep mode is determined to be the best energy saving option for Chalmers.

Ordlista

Apache License 2.0	Mjukvarulisens, tillåter all slags användning och modifiering. Enda kravet är att man måste ha kvar en kopia av licensen i text format. Man ska också hålla koll på vad som modifierats och vad som är original.
ARC	Advanced Resource Connector. En grid mellanprogramvara av NorduGrid. Kan ge ut jobb till olika distribuerande system.
ARP-tabell	En tabell för att matcha IP-adress mot MAC-adress. ARP står för Address Resolution Protocol.
BIOS	Basic Input/Output System. Det mest grundläggande programmet som körs när en dator startas.
Broadcast	En term inom datorkommunikation, ett paket som skickas mottags av alla inom broadcast domänen (oftast hela lokala nätverket). Motsatsen till enkelsänding.
Broadcast domän	De datorer som nås av en broadcast är i samma broadcast domän. Routrar begränsar domänen med standardinställningar.
BOINC	Berkeley Open Infrastructure for Network Computing. Ett system för att distribuera ut arbete till datorer.
Checkpoint	I HTCondor är detta en sparning av i vilket läge programmet är så att man enkelt kan återuppta det senare eller på en annan dator.
ClassAd	Ett sätt att lagra information. Används av HTCondor för all kommunikation mellan datorer och även mellan daemons.
COLLECTOR	En av HTCondors daemons. Sköter insamlingen av information från alla datorer och jobb.
Daemon	Program som körs i bakgrunden och inte använder sig av något GUI.
DAG	Directed Acyclic Graph. En riktad graf utan cykler.

DAGman	Directed Acyclic Graph Manager. Används om man vill skicka flera jobb till HTCondor där ordningen de utförs spelar roll.
Flocking	En funktion i HTCondor. Tillåter jobb att flyttas till en annan pool för körning. Händer endast när ingen dator kan köra jobbet på ordinarie poolen. Båda poolerna måste vara rätt konfigurerade.
Globus Toolkit	Ett verktyg med öppen källkod som används för att bygga upp grids.
GSI	Grid Security Infrastructure. Specifikation för säkerhet för kommunikation mellan datorer i ett grid system.
GUI	Graphic User Interface. Grafiskt användargränssnitt.
HPC	High Performance Computing. Mycket beräkningskraft leveraras på kort tid.
HTC	High Throughput Computing. Beräkningskraft leveraras under en längre tidsperiod. Veckor och månader för att lösa ett problem är inte ovanligt.
Kerberos	System för autentisering. Används default av Windows.
Kluster	Sammankopplade datorer som delar inskickat arbete. Man ersätter en stor kraftfull dator med flera sämre. Behöver ha samma geografiska placering.
KVM	Kernel-based Virtual Machine. Programvara för att köra virtuella maskiner på en dator.
Localsystem account	Ett inbyggt system konto i Windows datorer. Samma rättigheter som administratörer.
Magiskt paket	Nätverkspaket som används i systemet Wake-on-LAN för att väcka en dator via nätverkskortet.
MASTER	En av HTCondors daemons. Håller koll på alla andra daemons.
NEGOTIATOR	En av HTCondors daemons. Sköter utdelningen av jobb på Central Manager genom att passa ihop jobb med datorer.
PBS	Portable Batch System. Sköter utgivning av jobb bland dator resurser.
Pool	En grupp datorer sammankopplade med ett nätverk.
SCHEDD	En av HTCondors daemons. Körs på datorer som kan skicka ut jobb och sköter precis den biten.
Slurm	Simple Linux Utility for Resource Management. En mycket välanvänd resource manager och job scheduler bland världens stora kluster och superdatorer. Används av Chalmers Glenn kluster.

STARTD	En av HTCondors daemons. Körs på datorer som exekverar jobb för att få den funktionaliteten.
Unicore	UNiform Interface to COmputing REsources. Distribuerar resurser över internet och intranät.
Virtual private network	Kopplar ihop två datorer över internet så att de agerar som om de är på samma nätverk.
Virtuell maskin	Programvara som skapar en virtuell miljö för användaren. I denna miljö kan man exempelvis köra program som inte påverkar resten av systemet eller installera operativsystem för test eller annat.
VMware	VMware Inc, ett företag som tillhandahåller programvara för virtuella maskiner.
VPN	Se Virtual private network.
Wake-on-LAN	En standard för att väcka datorer via nätverket m.h.a. Magic Packet.
WOL	Se Wake-on-LAN.
Xen	En teknik för virtuella maskiner där det virtuella operativsystemet är medvetet om att det är virtuellt.

Innehållsförteckning

1 Inledning

Det är vanligt att större organisationer och företag låter sina datorer stå på dygnet runt trots att de större delen av tiden inte används av någon. Det sker samtidigt som vissa organisationer behöver mer och mer datorkraft för att utföra tunga beräkningsarbeten, inom exempelvis forskningsvärlden. Bristande tekniska lösningar inom området kan bidra till att onödig datorkraft köps eller hyrs in samt att kostnaderna för driften av existerande system är högre än nödvändigt. Vilket ur miljösyn inte heller är ett bra tillvägagångssätt, att ha datorer igång dygnet runt.

En organisation som både är stor och har mängder av datorer är Chalmers tekniska högskola. På Chalmers campus finns det både datorsalar för studenter och datorer för forskare och andra anställda. Många av dessa datorer står på under nätterna utan att användas. Chalmers vill att denna oanvända datorkraft ska minska, både för att spara pengar men även för att minska miljöpåverkan.

1.1 Syfte och Mål

Syftet med projektet är att skapa ett system som ska minimera den oanvända datorkraften på Chalmers. För att uppnå det är målet ett system som i första hand utnyttjar oanvänd datorkraft för tunga beräkningar. Dessa beräkningar har kravet att de ska kunna delas upp i mindre delar och på så sätt använda flera av datorerna samtidigt. Med andra ord ska det vara samma sekventiella kod eller algoritm som körs samtidigt på olika datorer men med olika startvärden. Om varken datorn eller dess arbetskraft används ska den automatiskt försättas i energisparläge för att spara el. Det här utmynnar i två viktiga huvudmål, hur systemets distribuering av arbete ska fungera och hur systemet ska hantera datorernas energibesparingar.

Systemet ska innehålla en samling, också kallad pool, av datorer. Det ska vara möjligt för användare med rättigheter att använda datorerna i poolen som beräkningskraft. Systemet ska själv skicka ett arbete till de datorer som för tillfället är mest lämpade att hantera det. Servern i systemet ska ha allt ansvar över de datorer och användare som finns och sköts av en eller flera administratörer. Dessa administratörer ska kunna ange olika behörigheter för användare och hantera distribueringen av arbetet. Det innebär att administratören ska kunna göra allt från detaljändringar på datorer till konfigurering av servern.

Användarna av systemet ska ha möjlighet att lägga ut arbete oberoende av vilket språk det är skrivet i. Det ska heller inte krävas att användaren har någon kunskap om distribuering. Det innebär att filer som skickas till systemet inte behöver modifieras. Användaren

ska heller inte behöva veta något om de datorer som filens arbeten körs på, allt detta ska systemet sköta. För att underlätta för användaren ska också ett GUI, grafiskt användargränssnitt, finnas. Detta GUI ska tillåta användaren att skicka ut filer till systemet och att se vilken arbetskraft som erbjuds för tillfället.

När inte någon använder en dator, varken en vanlig användare eller systemet i sig, ska datorn stängas av eller försättas i det energisparläge som är bäst lämpat, viloläge eller strömsparläge. Som en följd måste systemet även kunna starta datorerna igen när de behövs till arbetskraft eller för en användare. Detta innebär att datorn alltid ska vara tillgänglig och fungera felfritt, både för studenter och personal. Arbeten ska inte köras på datorer som används då detta kan innebära att en ökad väntetid eller sämre prestanda för de som använder dem.

För att kunna sätta på en dator ska tekniken Wake-on-LAN användas som är standarden inom området. Det innebär att den här tekniken ska implementeras i systemet, antingen genom en egen programvara som är kopplad till systemet eller att Wake-on-LAN redan är integrerat i systemet.

Systemet ska vara så öppet som möjligt och det ska finnas goda möjligheter att bygga vidare och att utveckla det. Trots många valmöjligheter ska det vara enkelt att använda. En installationsguide för administratörer och ett grafiskt gränssnitt för användaren ska räcka för att komma igång och använda systemet. Det ska vara ett system som kan uppfylla Chalmers behov och det ska vara möjligt att implementera det på hela campus.

Eftersom systemet ska vara möjligt att implementera på Chalmers krävs det också att det är säkert. Utomstående som ej ha behörighet ska inte komma åt datorer och/eller arbeten. Därav är det inte bara viktigt att ha rätt rättigheter till användarna utan att hela systemet är frånskilt från obehöriga. Vid yttre manipulering av systemet så som nätverksattacker ska systemet klara sig så gott det går.

1.2 Avgränsningar

Då distribuerande system för utdelning av arbete är ett komplext område, var avsikten att använda färdiga produkter som finns på marknaden och som är licensfria då möjligheten måste finnas att vidarutveckla och bygga vidare på ett sådant system.

För att begränsa omfattningen av vårt arbete har vi valt att inte behandla mer avancerade delar av distribuerande system så som att få arbeten att köras parallellt istället för sekventiellt. Fokus ligger istället på att köra sekventiella arbeten där samma kod skickas ut till flera datorer men med olika startvärden.

För att genomföra verklighetsbaserade tester krävs det att testerna utförs i en storskalig miljö. Det är dock inte genomfört utan de tester som har gjorts är i en mindre labbmiljö, som gett möjlighet för en teoretisk analys av större skala.

Säkerheten av systemet har inte testats praktiskt, då de förutsättningar som krävs för en väl genomförd undersökning saknas. Eftersom systemen som undersökts kan välja mellan kända säkerhetsstandarder finns inte heller ett behov till en närmare undersökning. Dessa

standarder kan naturligtvis ha brister men vi valde att inte göra en närmare undersökning.

1.3 Rapportens upplägg

I kapitel 2 presenteras de olika steg som tagits under utvecklingen av detta arbete. I korthet har det varit en lång förstudie följt av implementering och testning.

Kapitel 3 innehåller förstudien som är grunden till många av de beslut som tagits. Här beskrivs de efterforskningar vi har gjort och grunden till valet av system. Det görs även en jämförelse mellan den potentiella beräkningskraften som finns på Chalmers med de kluster som i nuläget är tillgängliga för forskare.

Kapitel 4 och 5 tar upp de tekniker som används. Kapitel 4 presenterar Wake-on-LAN, hur det fungerar, vad som krävs för att används det och vilka för- och nackdelar tekniken har. Kapitel 5 beskriver HTCondor, det distribuerande system som valdes, hur systemet fungerar och dess funktionalitet.

Kapitel 6 går igenom hur HTCondor kan installeras på Windows och Linux samt hur en grundläggande konfiguration av systemet kan göras. Här beskrivs också hur en dator ska ställas in för att kunna väckas med Wake-on-LAN. Vidare beskriver kapitel 7 hur vi testat HTCondor samt hur systemet ska konfigureras och användas för att få önskad funktionalitet.

I kapitel 8 presenteras det GUI vi har implementerat för att göra det enklare att skicka ut arbete med HTCondor. Här undersöks också vad det finns för existerande GUI:n till HTCondor och vilka API:n som finns tillgängliga.

Kapitel 9 beskriver den energistudie vi har gjort. Här beskrivs hur andra universitet har sparat pengar genom att stänga av datorer, huruvida Chalmers stänger av sina datorer och till sist en undersökning av datorers energiförbrukning.

De avslutande kapitlena, 10 och 11, består av resultat, diskussion samt slutsats. Det slutgiltiga resultatet presenteras och vi diskuterar huruvida målen har uppnåtts samt övriga tankar kring projektet.

2 Projektupplägg

Projektet består av tre huvuddelar, förstudie, implementering och testning. Fokus har i stor utsträckning legat på förstudien då det är den som ligger till grund för hela projektet. Det är i förstudien som projektets tillvägagångssätt bestäms och den är även grunden till många diskussioner och beslut som tagits under projektets gång.

2.1 Förstudie

Projektet inleddes med en grundlig forskningsperiod där relevanta områden studerades. Dokumentation och undersökningar kring Wake-on-LAN, distribuerande system, kluster samt datorers energikonsumtion analyserades. Speciellt fokus har lagts på hur andra universitet utnyttjar oanvänd datorkraft.

Vi gjorde en undersökning för att se om Chalmers har ett behov av ett system som utnyttjar oanvända datorer. En uppskattning gjordes på hur mycket potentiell beräkningskraft det finns för ett distribuerande system att använda. Om det skulle vara värt arbetsinsatsen för att implementera och administrera ett sådant system granskades också.

En intervju gjordes med IT-administrationen på Data-IT avdelningen på Chalmers där behovet av systemet diskuterades samt varför de flesta datorerna på campus alltid är på. Utifrån intervjun gjordes en egen studie, om energihantering av datorer, för att se om vi kom fram till samma slutsats.

Val av distribuerande system

Vi tog beslutet att använda ett redan fungerande system för distribueringen av arbete eftersom att bygga ett från grunden inom vår tidsram skulle bli väldigt svårt. HTCondor är det system som valdes eftersom det är specialiserat på att utnyttja oanvända datorer, se kapitel 3.3 för en fullständig motivering.

2.2 Implementering

Ett testsystem har implementerats i liten skala med ett fåtal sammankopplade datorer. Datorerna var uppkopplade till Chalmers nätverk och Internet för att miljön skulle vara mer realistisk samt för att underlätta arbetet med dem. Eftersom systemet är tänkt att användas på flera olika plattformar hade datorerna i labbmiljön både Linux och Windows installerat.

Följande datorer användes:

Vi har i våra tester haft tillgång till tre stycken Dell OptiPlex 740 datorer. Detta är datorer med ett par år på nacken.

Processor:	AMD Athlon 64 x2 3800+, 2GHz (2-trådar, 64-bitar)
Grafikkort:	Nvidia GeForce 6150 LE
RAM:	4GB
Hårddisk:	120GB

Senare i projektet fick vi tillgång till ytterligare 2st bättre datorer som kallas Compustation.

Processor:	Intel Core i5-3470S CPU, 2.90GHz (4-trådar, 64-bitar)
Grafikkort:	Nvidia GTX 480
RAM:	16GB
Hårddisk:	200GB

Chalmers har framför allt två olika typer av operativsystem på datorer i labb- och datorsalar. Det är Windows 7 och Red Hat Enterprise 6. Därav installerades labbmiljön med samma eller liknande operativsystem. En Optiplex installerades med Windows 7 och de två Compustation-datorerna var redan förinstallerade med detta operativsystem.

Eftersom Red Hat Enterprise inte är gratis togs beslutet att istället installera en av OptiPlex datorerna med Fedora, ett operativsystem som är mycket likt Red Hat och även sponsras av dem (red Hat, 2011). Den sista OptiPlex installerades med Ubuntu 12.10 för att undersöka om det uppstår problem vid användning av DEB-paket vid installationen jämfört med RPM (Fedora liksom Red Hat använder sig av RPM medan Ubuntu använder DEB).

Energistudie

En undersökning gjordes för att komma fram till vilket energisparläge som är mest lämpligt att använda på datorerna och hur mycket energi Chalmers skulle spara på det samt vad det skulle innebära i ekonomiska besparingar och miljöutsläpp.

Utveckling av GUI

Ett GUI konstruerades till HTCondor. All kodning skrevs i Java och det grafiska komponenterna gjordes främst med biblioteket Swing i Java. Ett API användes för att koppla GUI:t till det distribuerande systemet.

2.3 Testning

Tester skedde både parallellt med implementeringen och efter det att systemet blev klart för användning. Eftersom endast ett fåtal datorer var tillgängliga för implementation förblev vissa viktiga delar otestade. Hur systemet beter sig i storskalig miljö, med hundratals datorer, var inget som var möjligt att testa. Den informationen kommer inte från tester utan från rapporter och andra källor som analyserats och sammanställts. Även säkerheten är en del som lämnats otestad.

Omfattande tester gjordes på distribueringen av arbete, att resultatet av arbetet skickades

tillbaka och att systemet betedde sig på ett felfritt sätt när flera olika resultat sammanställs. Tester utfördes även på att datorer startas efter behov och stängs av när de inte behövs eller används, som var en central del av projektet. Därför har denna del testats i den mån det går med de datorer som fanns i testmiljön. I samband med detta testades även Wake-on-LAN för väckningen av datorer.

3 Förstudie

Följande kapitel presenterar de efterforskningar och undersökningar som ligger till grund för de val samt riktningar vi tagit i projektet. Först presenteras hur andra universitet har gått tillväga för att minska sin oanvända datorkraft och sedan beskrivs de distribuerande system som vi har undersökt under projektets gång. Utifrån den insamlade informationen motiveras det slutgiltiga valet av system och avslutningsvis beskrivs den nuvarande situationen på Chalmers i form av beräkningsbehov och potential.

3.1 Andra universitets lösningar

Systemet är tänkt att kunna användas främst på större organisationer som universitet. Därför har en noggrann undersökning och jämförelse gjorts mellan universitet som har liknande projekt för att dela ut arbete till oanvända resurser och spara energi. Här presenteras de universitet och deras lösningar som hittats under förstudien.

3.1.1 University of Liverpool

University of Liverpool är ett universitet i England med ett stort behov av att utföra tunga beräkningar vilka bestämde sig för att använda verktyget HTCondor för att utnyttja campusdatorernas oanvända datorkraft. HTCondor användes för att bygga upp en pool av datorer för att sedan kunna skicka ut arbete till dem från en server.

Skolan hade sedan tidigare undersökt möjligheterna att spara energi genom att inte låta datorer stå på i onödan. Enligt undersökningarna kom de fram till att datorerna i klassrummen endast användes 6% av den sammanlagda tiden på ett år och personalens datorer endast 8%. För att spara energi bestämde de sig för att använda mjukvaran PowerMAN som kan övervaka aktiviteten på en dator och sätta den i strömsparläge när den inte används. Programmet kan även samla in användbar statistik om hur datorer används och hur deras elförbrukning ser ut. Efter att ha infört en ny energibesparingspolicy och börjat försätta datorer i strömsparläge, när de inte används, har universitetet enligt beräkningar avlägsnat mellan 200 000 och 250 000 timmar av inaktivitet varje vecka och sparat in £124 000 på den årliga elräkningen.

När HTCondor infördes på skolans datorer uppstod ett problem med strömsparfunktionerna. Datorerna stängdes av efter en viss tid av inaktivitet trots att HTCondor-arbete kördes på datorn. För att lösa problemet konfigurerades PowerMAN så att datorer aldrig stängs av när HTCondors arbetsprocess kördes på datorn. PowerMAN användes också till att sätta på datorer med Wake-on-LAN för att till exempel installera uppdateringar eller köra HTCondor-arbete.

Då HTCondor redan innehåller funktioner för att stänga av och sätta på datorer efter behov så hade en extra programvara för energisparande egentligen inte varit nödvändigt. Eftersom universitetet redan använde produkten PowerMAN, och var nöjda med den, valde de att inte använda HTCondors inbyggda strömsparfunktioner utan att fortsätta med sin redan fungerande lösning (University of Liverpool, 2010).

3.1.2 Clemson University

Clemson University i South Carolina har använt sig av HTCondor och BOINC med målet att minimera den oanvända datorkraften så mycket det går på campus. Till skillnad från de flesta andra universitet vill de att datorerna ska användas dygnet runt istället för att stängas av när det inte finns något arbete att utföra. Det görs genom att använda HTCondors backfill-mekanism så att arbeten kan tas emot från BOINC när det finns lediga resurser på nätverket, men inte tillräckligt med lokala arbeten för att fylla upp dem. BOINC är en mjukvara som tar emot arbete från olika forskningsprojekt världen över med behov av extra beräkningskraft för deras forskning, vilket gör att det alltid finns arbeten att utföra på campus datorer. Under det första året med BOINC hamnade Clemson University på femte plats i världen över de användare som bidragit mest till BOINC-projekt.

Skolan är också medlem i Open Science Grid (OSG), en nationell organisation i USA där flera universitet och organisationer kopplar ihop sina nätverk så att de har tillgång till varandras resurser. På så sätt kan de ta emot arbeten från andra forskare i OSG och även skicka ut arbeten till andra medlemmars nätverk i OSG när det inte finns några lediga resurser i det egna systemet.

För att öka användandet av HTCondor-poolen fick alla studenter rättigheter att skicka ut arbeten. Det ökade användningen av poolen markant och huvudservern blev till och med överbelastad vid ett tillfälle när hela 500 000 arbeten var aktiva samtidigt.

Det gjordes också en undersökning på hur mycket extra energi som gick åt på datorer med HTCondor som arbetar dygnet runt jämfört med datorer som går i strömsparläge när de inte används. Skillnaden i elförbrukning var som väntat väldigt stor. En dator med HTCondor drog i genomsnitt 2,0534 kWh/dag med en kostnad på $104,93 per år medan en dator med strömsparfunktioner drog 0,4116 kWh/dag med en kostnad på $21,32 per år.

Trots den höga elförbrukningen vill universitetet fortsätta använda sin lösning eftersom de anser att det är viktigare att bidra till forskningsvärlden, än att spara in på elräkningen (Sepulveda, 2009).

3.1.3 University College London

University College London (UCL) är det tredje äldsta universitetet i England och ett av de tidigare engelska universiteten att implementera en HTCondor-pool på deras campus. De började undersöka möjligheterna att implementera ett distribuerande system på campus datorer då behovet av beräkningskraft var stor bland olika forskningsorganisationer i landet. De tog beslutet att sätta upp en HTCondor-pool bestående av drygt 930 processorkärnor och ge projektet eMinerals tillgång till poolen. Forskarna på eMinerals arbetar med simuleringar på molekylnivå för att lösa aktuella miljöproblem.

Valet att använda HTCondor gjordes efter att noggrant ha gått igenom kraven från användarna av poolen och skolans IT-avdelning och även kraven för applikationerna som ska kunna köras. HTCondor ansågs vara den lösning som uppfyllde de flesta av kraven. Det är en välutvecklad mjukvara som utan behov av andra programvaror fungerar som ett komplett distribuerande system.

Under de första 6 månaderna med HTCondor hade skolans datorer bidragit med ca 110 CPU-år till eMinerals och andra forskare. Då de inte har något system för att spara energi utan låter datorerna arbeta dygnet runt har energiförbrukningen ökat på skolan efter att man började använda HTCondor. Dock bidrar de till en bättre miljö genom att ge mer beräkningskraft till eMinerals vilket kan anses vara ett ganska bra alternativ jämfört med att stänga av datorer. (University College London, 2004)

3.1.4 Övriga universitet

Utöver de ovan presenterade universiteten finns det en mängd andra skolor i världen som använder distribuerande system. Speciellt bland engelska universitet har det blivit väldigt populärt att använda distribuerande system för att utnyttja campus datorer bättre. Universiteten i Cambridge, Manchester, Reading och Lancaster är bara några exempel på skolor som använder sig av HTCondor på campusdatorerna. Det finns även organisationer såsom Northwest Grid, Campus Grids Special Interest group (CS-SIG) och National Grid Service som kopplar ihop flera universitets och andra organisationers nätverk och gör resurserna tillgängliga för diverse forskningsprojekt.

University of Huddersfield, som själva använder HTCondor på campus, har gjort en undersökning bland medlemmarna i CS-SIG och Northwest Grid om deras erfarenheter med HTCondor. Enligt resultaten består de flesta HTCondor-poolerna av 1 000 till 2 000 processorkärnor med i genomsnitt 45 aktiva användare som skickar ut arbete till datorerna. Endast hälften av de undersökta organisationerna använder energibesparande funktioner på datorerna medan resten låter datorerna stå på dygnet runt. Nästan ingen använder HTCondors inbyggda energisparfunktioner utan egna lösningar används istället. Över lag är organisationerna väldigt nöjda med HTCondor och tycker det är en värdefull och mycket användbar produkt (Gubb, 2012).

Universitetet i Westminster har istället för att köpa en superdator använt BOINC på sina campusdatorer och på så sätt sparat in tusentals pund. Det är det första universitetet som har byggt upp en BOINC-pool i England där de flesta andra använder HTCondor (Evans-Toyne, 2011).

HTCondor och andra distribuerande system används även flitigt i USA där flera skolor, likt tidigare nämnda Clemson University, är medlemmar i organisationen Open Science Grid som består av flera ihopkopplade nätverk över hela landet. Ett av universiteten, Purdue University, har byggt upp en HTCondor-pool med hela 40 000 processorkärnor.

Sammanfattningsvis så har distribuerande system blivit ganska vanligt förekommande på olika universitet. Det allra vanligaste är att använda HTCondor men det finns även universitet som använder BOINC, någon annan programvara eller en kombination av flera.

3.2 Distribuerande system

Vi har undersökt vilka verktyg som skulle kunna användas för att distribuera ut arbete. De bästa kandidaterna, utöver det utvalda systemet HTCondor, presenteras i kapitlet. Dessa skulle kunna användas som alternativ till eller tillsammans med HTCondor, beroende på vilken typ av lösning som eftersöks.

3.2.1 Globus Toolkit

Globus Toolkit är ett verktyg med öppen källkod utvecklat av Globus Alliance som används för att bygga upp ett distribuerande system. Det är en så kallad mellanprogramvara (eng: middleware) som ligger emellan operativsystemet och applikationerna på det distribuerande systemet. En mellanprogramvara kan ses som ett lim mellan två mjukvarulager som möjliggör kommunikation mellan dessa lager (Objectweb, 2007).

Globus är välkänt inom många vetenskapliga organisationer och företag, det används av bland annat CERN och är sponsrat av NASA, IBM, Microsoft med flera. Mjukvaran finns tillgänglig för Linux, Windows och OS X.

Globus Toolkit består av 5 komponenter: GSI, GridFTP, GRAM, MyProxy och GSI-OpenSSH. GRAM är den komponenten som sköter distribuering av arbete över nätverket och är därför den mest intressanta för det här projektet. GridFTP används för att dela filer över nätverket och de övriga komponenterna sköter administration och säkerhet. Via kommandotolken går det att skicka ut arbeten till klustret, för att sedan följa status, integrera med arbetet samt få ut resultatet. Det finns även flera API:er för GRAM såsom GRAM API och SAGA. Med hjälp av dessa går det att t ex skriva en applikation med ett GUI som kan underlätta för användaren att skicka ut arbete.

3.2.2 BOINC

Berkeley Open Infrastructure for Network Computing (BOINC) är en mellanprogramvara för distribuerande system utvecklat av University of California, Berkeley. Mjukvaran har öppen källkod och stödjer Windows, Linux och OS X.

BOINC-projektet leds av forskaren David Anderson och utvecklades till en början för att ge mer beräkningskraft åt Andersons sidoprojekt, SETI@home, vars syfte är att söka efter utomjordiskt liv genom att analysera radiosignaler uppfångade av Arecibo-observatoriet i Puerto Rico.

BOINC gör det möjligt för forskare som har behov av tunga beräkningar att skapa ett så kallat BOINC-projekt. Vem som helst kan ladda ner och installera en BOINC-klient på sin persondator och sedan ansluta sig till ett eller flera projekt. Klienten börjar då ta emot arbete från projektet, beräknar ett svar och skickar sedan tillbaka resultatet. Den här typen av system kallas för volontär beräkning (eng: volunteer computing). För tillfället finns det ca 1,2 miljoner datorer som aktivt använder BOINC med en genomsnittlig beräkningskraft på 8,6 petaFLOPS (flyttalsoperationer per sekund) och över 40 projekt som går att ansluta sig till.

Företag och universitet kan använda BOINC till att sätta upp ett eget distribuerande system. Universitet kan skapa ett så kallat Virtual Campus Supercomputing Center (VCSC) som gör det möjligt för forskare att använda campus datorer som en superdator för beräkningar. Det går att konfigurera så att inga beräkningar körs på datorerna när någon sitter vid datorn. Det finns dock inga funktioner för att spara energi när datorerna inte används till något.

Det går även att använda BOINC tillsammans med en existerande HTCondor-pool. Genom att använda HTCondors backfill-funktion kan beräkningar hämtas från BOINC-projekt och utföras på datorer i poolen när det inte finns några lokala arbeten att utföra.

3.3 Vårt val av system

Huvudorsaken till att HTCondor valdes är för att det är ett mycket effektivt system för opportunistisk användning av datorkraft. Det innebär att arbete inte utförs på datorer som redan används och pausas eller flyttas om någon börjar använda en dator där arbetet startat. Det vill säga användaren störs eller påverkas inte av HTCondor.

Systemet används på universitet och organisationer världen över vilket har bidragit till fortsatt utveckling och förbättring. Flera olika kraftfulla verktyg erbjuds som att spara arbeten för att återuppta vid ett senare tillfälle, eller rent av på en annan dator, och då fortsätta på samma ställe som när det sparades. En annan utomordentlig funktion är möjligheten att skicka arbeten mellan olika pooler när datorer inte finns tillgängliga i den ordinarie poolen. Det finns även, något som är intressant för detta projekt, inbyggt stöd för energisparande genom att datorerna kan försättas i energisparläge och sedan väckas när de behövs.

Jämfört med de övriga system som främst studerades, Globus och BOINC, finns det flera fördelar med HTCondor. Alla tre har öppen källkod och stödjer operativsystemen Linux och Windows som används på Chalmers. Dock är Globus främst anpassat till Linux och det finns ingen manual som beskriver hur det kan installeras och konfigureras på Windows. Globus är dessutom ett större och mer komplext system med en ganska krånglig installationsprocedur och det verkar inte finnas några andra universitet som har implementerat detta system. Av dessa anledningar valde vi bort Globus ganska tidigt. Det andra systemet, BOINC, är ett bra och välanvänt verktyg men används främst för volontär beräkning och inte lika ofta för att bygga upp en pool för distribuering av arbete. Det är därför inte lika väldokumenterat och enkelt att installera som HTCondor. Ett bättre användningsområde för BOINC är förmodligen att använda det tillsammans med HTCondor för att ta emot arbeten från diverse BOINC-projekt och köra på HTCondor-poolen.

Utifrån de rapporter från andra universitet, som vi har studerade verkar HTCondor vara det bästa och utan tvekan mest populära systemet att använda för att utnyttja datorer på campus till beräkningar.

3.4 Datorkraft tillgänglig på Chalmers

Vi har undersökt huruvida Chalmers har ett behov av ett distribuerande system och hur mycket mer datorkraft som skulle kunna fås. De kluster som finns idag på Chalmers presenteras här och en jämförelse mellan dem och den potentiella kapaciteten från campus datorer har gjorts.

3.4.1 Potentiell datorkraft

På Chalmers finns det ungefär 10 000 studenter och över 2 300 anställda på de två olika campus, Johanneberg och Lindholmen (Chalmers, 2012). Detta innebär att det behövs många stationära datorer som alltid ska vara tillgängliga. Antalet datorer stående i datorsalar på Johanneberg, det största av Chalmers två campus, är 612 med Windows 7 och 229 installerade med Red Hat Enterprise Linux. På Lindholmen finns det inte lika många, 150 Windows 7 och 32 med Red Hat. Totalt sett motsvarar det hela 1 023 datorer i datorsalar som potentiellt skulle kunna utnyttjas för beräkningar eller stängas av när de inte används. Informationen om antalet datorer hämtades från (Chalmers StuDAT, 2013). Dessa siffror räknar inte med labbdatorer eller andra datorer som inte står i datorsalar. Det innebär att det finns mycket mer potentiell datorkraft än vad siffrorna pekar på.

Enligt IT-administrationen på Data och IT institutionen finns det alltid ett behov av mer beräkningskraft på Chalmers. Varje år får de in flera förfrågningar från personal och studenter som vill ha mer datorkraft. Det finns dessutom statistik från de kluster som beskrivs i nästa avsnitt som visar på att de används.

De datorer som finns på Chalmers campus stängs aldrig av, varken på nätter eller lov. Det finns undantag för vissa dator- och labbsalar. Datorerna används främst under lektions- och arbetstid som är mellan klockan 8.00 till 17.00 på vardagar. Efter denna tid får datorerna användas men görs bara det av en liten minoritet. Det innebär att större delen av tiden står de helt oanvända och drar ström vilket har en negativ verkan på miljön och Chalmers ekonomi.

Att det finns potential till att utföra beräkningar och spara energi råder det ingen tvekan om. Med mer än 1 000 datorer att utnyttja kan ett effektivt system öka beräkningskapaciteten på Chalmers avsevärt. Här kan pengar sparas i både energi och inköpskostnader av datorer till kluster.

3.4.2 Kluster tillgängliga på Chalmers

Den enda generella resursen som finns tillgänglig för beräkningar på Chalmers är C3SE enligt Arne Linde, IT-chef för Data och IT institutionerna. C3SE har hand om tre olika kluster, Glenn, Beda och Svea (C3SE, 2013). De är inköpta kluster som enbart är till för beräkningar för forskare på Chalmers. För att få lov att använda dem krävs det att användaren är med i ett projekt, har rätt behörighet och bokar en tid på något av klustren. Glenn är det största av dem, ungefär lika stor som de två andra klustren tillsammans. Klustren använder sig av olika system för att distribuera ut arbete.

Glenn är det största klustret med sina 379 noder, sammanlagt 6 080 kärnor och 18.1TB RAM. Det är endast fyra stycken av noderna som har ett grafikkort monterat. Grafikkorten

är Tesla M2050 (Nvidia, 2011) som är speciellt byggda för High Performance Computing, vilket innebär att de ger mycket beräkningskraft. Två noder erbjuder resurser som gör det möjligt att använda dem som virtuella desktops. Vilket sker med hjälp av hårdvara för att sända över grafiken, Leadtek VP200H PCoIP adaptrar. Dessutom finns tre servrar att koppla upp sig till för att komma åt systemet och att administrera det. Glenn använder sig av Slurm för att kontrollera resurserna och arbeten samt för att sköta själva distribueringen av arbete, när arbeten ska göras, vilken dator det ska göras på och så vidare (C3SE, 2013).

De två andra klustren är inte lika omfattande i storlek och erbjuder inga av de speciella funktioner som Glenn har. Beda är det större av dessa två med 268 noder, med totalt 2 144 kärnor och 7TB RAM. Svea är mycket mindre jämfört med de andra två och har även sämre hårdvara, bara 91 noder vilket ger mindre än 1 000 kärnor och 2TB RAM totalt sett.

Dessa två kluster använder inte Slurm som Glenn utan bygger istället på två olika system, TORQUE för att hålla koll på resurser och arbete och Maui för själva distribueringen. Båda systemen har öppen källkod och finns att ladda ner på Adaptive Computings hemsida (Adaptive Computing, 2013).

TORQUE utvecklas fortfarande men Adaptive Computing har lagt Maui på hyllan och istället börjat med ett nytt system, Moab. Moab är ett liknande fast bättre system (Adaptive Computing, 2011) men nackdelen är att den varken har öppen källkod eller är tillgänglig gratis.

För att skicka ut arbeten till Chalmers två olika system för distribuering av arbete, Slurm och TORQUE/Maui, krävs en submit-fil där olika parametrar ska specificeras (namn, antalet noder osv.) samt det skript som ska köras. För personer som inte är särskilt insatta i datorer och programmering och vill distribuera ut arbete kan det vara svårt att komma igång, inget GUI eller liknande ''förenkling" erbjuds.

3.4.3 Jämförelse mellan den oanvända datorkraften och klustren

Följande är en mycket grov beräkning där de oanvända datorernas kapacitet jämförs med de existerande kluster som finns på Chalmers. Datorerna på Chalmers uppskattades till 2 000 stycken men det kan vara i underkant då IT-administration grovt trodde att det fanns över 3 000 datorer. Efter ett par stickprov på olika datorer beslutades ett medelvärde på 3 kärnor och 6 GB RAM per dator. Det är förvisso inte bara dessa parametrar som påverkar hur bra kluster är men de är de lättaste att mäta och jämföra.

	Chalmers	Svea	Beda	Glenn	Alla kluster
Kärnor	$\sim$ 6000	640	2144	6080	8864
RAM (TB)	$\sim$ 12	1.3	7	18.1	26.4

Tabell 3.1: Grov jämförelse mellan Chalmers oanvända datorers potential och de kluster som finns. Information om kluster från C3SEs hemsida (C3SE, 2013).

Ur tabellen går det att utläsa att den oanvända datorkraften på Chalmers är i samma storlek som det största klustret Glenn, dock finns inte samma mängd RAM. Om beräkningskapaciteten bara skulle bero av dessa två variabler skulle Chalmers den totala beräkningskapaciteten

troligtvis kunna öka med mer än 50%. Detta är inte hela sanningen, det finns en rad faktorer som påverkar kapaciteten bland annat hur snabba processorerna är. Men det går ändå att dra slutsatsen att beräkningskapaciteten utan tvivel skulle öka markant.

4 Wake-on-LAN tekniken

Wake-on-LAN, förkortat WOL, är en nätverksstandard som används för att väcka datorer. Det går att använda sig av Wake-on-LAN oavsett om datorn som ska väckas är avstängd, i vilo- eller strömsparläge. Kapitlet redogör för hur detta fungerar samt de svagheter som standarden har.

4.1 Väckning av datorer

För att väcka en dator med Wake-on-LAN skickas ett ''magiskt" paket (eng: magic packet) till datorn som ska väckas. I paketet finns MAC-adressen till nätverkskortet i måldatorn. MAC-adressen identifierar unikt den dator som ska väckas. När nätverkskortet får det magiska paketet startar den datorn.

Ett magiskt paket skickas vanligen som ett UDP-paket. Paketet kan skickas till lämplig port men går vanligtvis till port 7 eller 9. Det viktiga är innehållet där ett magiskt paket innehåller först 6 bytes med bara 1:or, hexadecimalt blir det `FF FF FF FF FF FF`. Efter det så repeteras måldatorns MAC-adress 16 gånger. Allt detta uppnår totalt till 102 byte. (AMD, 1995).

Det finns ett antal krav på hårdvaran, nätverket och den tillgängliga informationen för att Wake-on-LAN ska kunna fungera korrekt. Det mest uppenbara är att MAC-adressen för måldatorn måste vara känd. Det krävs också att hårdvaran i måldatorn stödjer Wake-on-LAN, då nätverkskortet hela tiden ska skanna av nätverkstrafiken för att veta om ett magiskt paket kommer. Detta kräver att nätverkskortet aldrig är helt avstängt utan bara i energisparläge även när resten av datorn är avstängd. Den negativa sidan är att strömförbrukningen ökar, dock marginellt. På datorer som stödjer Wake-on-LAN finns inställningar i BIOS för att aktivera funktionen (Asus, 2013).

Det krävs en trådbunden förbindelse till nätverket eftersom trådlösa nätverkskort inte kan skanna av nätverkstrafiken i energisparläge. Det finns dock en standard för att väcka datorer på trådlösta nätverk, Wake on Wireless LAN (WoWLAN) men nästan ingen hårdvara stödjer det (Andrew vonNagy, 2010).

4.2 Krav på nätverket

Vid sändning av ett magiskt paket underlättar det om måldatorn är inom samma lokala nätverk. Mer specifikt inom samma broadcast domän, dvs. att den går att nå genom att

broadcasta paketet. När broadcast används behövs inte IP-adressen till datorn som ska väckas, något som förenklar saken eftersom datorer generellt sett inte behåller sin IP-adress när de är avstängda. Utan broadcast kan inte datorn väckas eftersom routern inte vet var den ska skicka paketet. Undantaget är när IP-adressen fortfarande ligger kvar i routerns ARP-tabell.

Det här problemet försvinner om routern kan lägga in datorer som statiska i ARP-tabellen. Datorn behöver då även ha en statisk IP-adress, vilken är samma adress som läggs in i ARP-tabellen. Då vet routern alltid var den ska skicka paketet. Denna lösning är inte bra för många datorer, speciellt inte om de flyttas runt eller byts ut, eftersom datorerna måste använda statiska IP-adresser. Det finns ett antal andra sätt att lösa problemet, vilket som är bäst beror på situationen.

- Subnet-directed Broadcast (AMD, 1995), SDB, fungerar om datorerna befinner sig på olika subnät. Routrarna mellan de olika nätverken måste konfigureras till att släppa igenom broadcast-paketen.
- Portforwarding kan användas på routern. Idén är att alla paket som kommer till en speciell port skickas vidare till en IP-adress. Då skickas de magiska paketen till routerns IP-adress och den speciella porten. Därefter beroende på konfiguration, broadcastar routern paketet eller skickar vidare paketet till en annan IP-adress. Det fungerar endast om routern kan ställas in med statiska datorer i ARP-tabellen. Den här tekniken är omständig men fungerar bra för ett fåtal datorer.
- Om systemet inte behöver vara helt automatiskt och om routern själv kan skicka Wake-on-LAN paket, så finns det ett annat alternativ. Det är att logga in på routern och därifrån skicka ut WOL-paketet till subnätet.
- Ett annat sätt är att skapa ett ''Virtual private network", VPN. Då agerar datorerna som om de är inom samma nätverk fast de inte är det. Det kan bli problem med att använda broadcast med den här metoden beroende på vilken programvara som används för att skapa VPN nätverket.
- Om en dator i samma subnätet som måldatorn alltid är igång går det att använda SSH eller annan programvara för fjärrstyrning för att komma in i den. Därefter är det bara att skicka ett WOL-paket till subnätet.

4.3 Säkerhet med Wake-on-LAN

När Wake-on-LAN-tekniken är aktiverad innebär det att alla som vet om måldators MAC-adress kan väcka denna. MAC-adresser går även snabbt att ta reda på för datorer på samma nätverk, med någon programvara för att se all trafik över nätverket, exempelvis det fria programmet Wireshark (Wireshark, 2013). Men även om andra personer kan väcka datorerna så är det i sig inte någon stor säkerhetsrisk. De kan fortfarande inte göra någonting med datorerna.

Det finns teknik som blockerar andra från att väcka datorerna, men då krävs speciell hårdvara som stödjer något som heter ''SecureON" (DD-WRT, 2012). För den hårdvaran går det att ställa in ett 6 bytes lösenord och datorn sätts bara på om både MAC-adressen och lösenordet är korrekta. Det försvårar otillåten uppstart men är fortfarande inte säkert då lösenordet skickas i klartext när datorn ska väckas och det är möjligt att avlyssna nätverket och få reda på lösenordet.

5 Distribuerande systemet HTCondor

I det här kapitlet presenteras HTCondor, det distribuerande system som vi valt att använda, hur systemet är uppbyggt, hur det fungerar samt den viktigaste funktionaliteten.

Systemet är främst inriktat på HTC, High Throughput Computing, där syftet är att leverera datorkraft under långa tidsperioder, veckor eller månader är inte ovanligt. Motsatsen är HPC, High Performance Computing, där mycket beräkningskraft fås under kort tid. Vanliga kluster, däribland Chalmers egna, är inriktade på HPC.

5.1 Introduktion

HTCondor (HTCondor, 2013a) är skrivet i C++. Systemet sköter utdelandet av arbete till tillgängliga resurser, prioriteringen och schemaläggning av dessa arbeten samt resurshantering och övervakning. Användare skickar in arbeten till en kö och HTCondor bestämmer när och var de ska köras, allt efter förinställda konfigurationer.

HTCondor är ett gammalt projekt med öppen källkod som varit i utveckling sedan 1984 (Thain, 2005) och är licenserat under Apache License 2.0. Det innebär att det är tillåtet att modifiera källkoden, men det måste vara tydligt vad som modifierats.

Systemet utvecklas fortfarande aktivt av avdelningen för datavetenskap på University of Wisconsin-Madison och nya uppdateringar släpps regelbundet. Från början kallades det bara Condor. År 2012 tvingades de byta namn eftersom Phoenix Software International redan under en längre tid använt namnet på en av sina produkter. Bokstäverna i början på det nya namnet är från systemets syfte: att möjliggöra HTC, High Throughput Computing.

Det som utmärker HTCondor framför andra liknande system är att det är opportunistiskt. Det har med andra ord förmågan att mycket effektivt utnyttja oanvänd datorkraft från, för tillfället, oanvända datorer (Smith, 2010). Systemet används av hundratals organisationer och företag till att implementera både större och mindre pooler, från ett fåtal till tusentals sammanbundna datorer (HTCondor, 2013b).

5.2 Uppbyggnad av en HTCondor-pool

HTCondor har den stora fördelen att det går att installera och köra på Windows, Linuxbaserade system och även Mac-datorer mixade i samma pool (Calleja, 2004). I samma pool är det också möjligt att ha dedikerade datorer, datorer som inte ska användas till annat än beräkningar. Det går följaktligen att skapa och administrera ett traditionellt kluster med

HTCondor, traditionellt i den mening att ingen använder datorerna för annat än beräkningar.

I en HTCondor-pool har datorerna olika roller beroende på vad de har för uppgifter. Det finns alltid en Central Manager i varje pool, det är maskinen som matchar ihop tillgängliga resurser med de arbeten som är i kö för att skickas ut. Utöver denna centrala punkt finns det datorer för att utföra arbeten, executors, och datorer för att skicka ut arbeten, submitters. Det går även bra att kombinera de tre rollerna.

Det finns inte några krav på hur bra en dator behöver vara för att utföra arbeten. Förvånansvärt nog krävs det mer av de datorer som är submitters. Detta på grund av att för varje arbete som skickas ut körs en process på submittern. Samtidigt krävs även en del utrymme på hårddisken då så kallade checkpoints, sparade tillstånd av arbeten, skickas till och sparas på submittern. Om arbetet ifråga är stort kan det krävas en hel del plats. Det finns i själva verket en fjärde roll för datorer i poolen som bara har som uppgift att spara dessa checkpoints, en checkpoint server. (HTCondor Manual, 2013c).

Datorerna i poolen skickar med jämna mellanrum uppdateringar av sitt tillstånd till Central Managern. De kan vara väldigt många till antalet och fler datorer i poolen sätter högre krav på en kraftfull Central Manager och ett bra nätverk (HTCondor, 2013b). Ett alternativ är att bygga flera olika pooler som sedan kopplas samman, mer under 5.10 HTCondors skalbarhet.

5.3 Bakgrundsprocesser

En dator med HTCondor installerat kör ett par daemons, processer som körs i bakgrunden. Vilka och hur många daemons som körs beror på vilken roll datorn har i poolen. De grundläggande och viktigaste är beskrivna här nedan men det finns fler, runt 20 stycken olika (HTCondor Manual, 2013c). Många av dessa är mindre intressanta och används automatiskt av HTCondor men ett par till kommer beskrivas längre fram.

condor_master är processen som övervakar alla andra daemons som körs på maskinen och ser till att de daemons som ska vara igång är igång. Den körs alltid på alla datorer i poolen.

condor_collector sköter insamlingen av tillstånden på datorerna i poolen och vilka resurser de har att erbjuda. Processen körs bara på Central Managern.

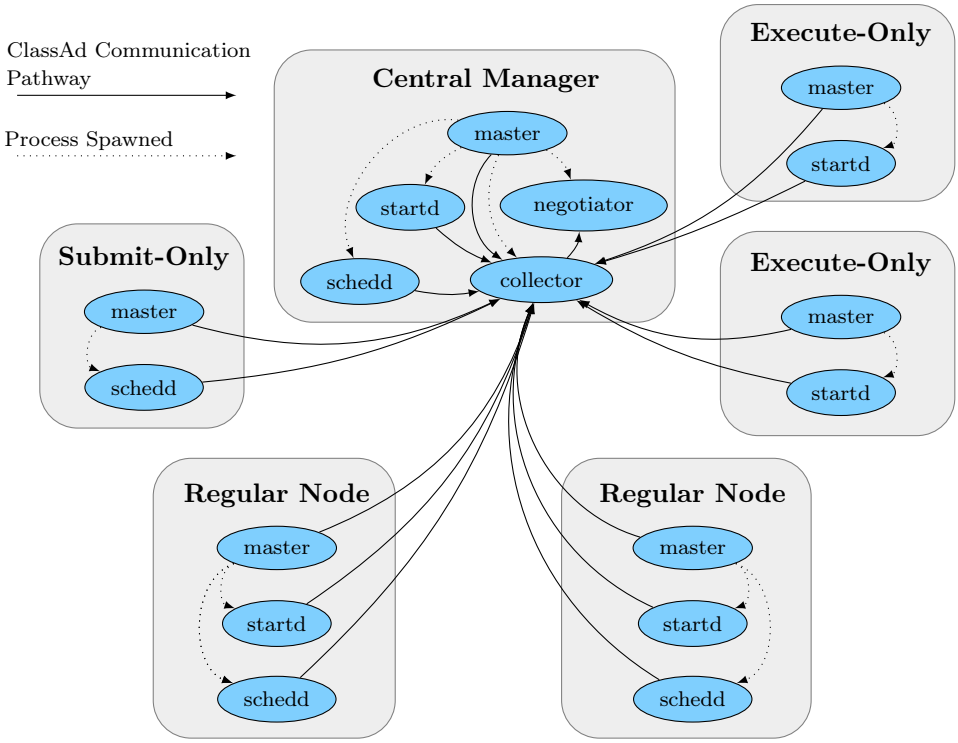
condor_negotiator har som uppgift att dela ut arbete till lämpliga resurser. Den håller också koll på vilka arbeten som har prioritet över andra och kan bestämma att byta ut arbeten som har lägre prioritet till ett som har högre. Användare som fått mycket resurser tilldelat får lägre prioritet jämfört med en användare som inte utnyttjar någon resurs. Processen körs bara på Central Managern. Det ska nämnas att det går att köra **condor_collector** och **condor_negotiator** på olika maskiner och på så sätt dela upp Central Managerns uppgifter på två datorer.

condor_startd körs av alla executors, maskiner som kan köra arbeten. Den kommunicerar med **condor_collector** om vilka resurser som erbjuds. Den ser även till att alla

inställningar om när och hur arbete får köras på datorn. När datorn befinner sig i ett läge där den är beredd att ta emot ett arbete av HTCondor startar den en ny daemon, **condor_starter**, som sköter allt som har med själva arbetet att göra.

condor_schedd är den daemon som skickar ut arbete från en submitter. För varje arbete som för tillfället körs finns det en annan process, **condor_shadow**, som håller koll på arbetet.

I figur 5.1 här nedan är ett exempel på hur en pool kan se ut. Poolen består av sex stycken datorer, varav en är Central Manager. I poolen finns en blandning av olika sorters klienter, två kan bara utföra arbeten (Execute-Only), en av dem kan bara skicka (Submit-Only) och de två sista kan båda delarna (Regular Node). Bilden visar hur master processen skapar de andra processerna samt hur kommunikationsvägarna mellan och inom datorerna ser ut.



Figur 5.1: En liten HTCondor-pool på sex datorer.

5.4 All kommunikation sker med ClassAds

Kommunikation mellan datorer samt alla olika daemons sker med ClassAds, Classified Advertisments. En dator kan med en ClassAd beskriva vilka egenskaper den har. Operativsystem, minne, om java finns, allt som är intressant vid valet om den ska tilldelas ett specifikt arbete. Även arbeten beskrivs med en ClassAd där alla de krav arbetet har beskrivs samt

olika prioriteringar. Central Managern samlar ihop ClassAds från poolens alla maskiner och matchar de mot alla ClassAds för arbete, för att se vilken dator som är lämplig för vilket arbete, om någon. De används inte bara till kommunikation utan sparas även som loggfiler för statistik och felsökning (HTCondor, 2013c).

ClassAds är ett simpelt men effektivt verktyg. Det är en länk mellan attribut om uttryck. Uttrycken kan vara allt från tal, booleska värden till text och sammansättningar av andra attribut. Kort sagt är en ClassAd en lista med egenskaper.

Det går att evaluera uttryck i en ClassAd med hjälp av en annan ClassAd. Det är så en Central Manager matchar ihop arbete med resurser. Central Managern kollar då på attributet *requirements* på både arbetet och datorn. Om båda evalueras till sant med hjälp av den andra matchas de ihop och arbetet sänds till den datorn. Ett exempel på en ClassAd för en Linux dator finns i tabell 5.4 här nedan. Ett sätt att få *requirements* att bli sann i exemplet är om *TARGET.Owner=="emil"*. Om ett arbete skickas ut med attributet *Owner* satt till ''emil" kommer exempel ClassAd:en evalueras till sann med hjälp av den.

MyType	=	"Machine"
TargetType	=	"Job"
Machine	=	"CE-PC05.net.chalmers.se"
Arch	=	"INTEL"
OpSys	=	"LINUX"
Disk	=	35882
Memory	=	128
KeyboardIdle	=	173
LoadAvg	=	0.1000
Requirements	=	TARGET.Owner=="emil" ||
		LoadAvg<=.3 && KeyboardIdle>15*60

Tabell 5.1: Exempel på en ClassAd för en Linux dator

5.5 Konfiguration av HTCondors beteende

För att datorerna i poolen ska bete sig som önskat måste de konfigureras på rätt sätt. Detta görs i filer där olika variabler kan regleras och på så sätt ändra funktionalitet. I HTCondor görs konfigurationen lämpligast i flera lager med en globalfil för alla datorer och sedan lokala filer på varje dator om det är något som skiljer den datorn från resten. De underliggande filerna skriver över parametrar av filer högre upp i hierarkin. Toppfilen, den globala filen, pekas ut med en så kallad ''environment variable", en variabel på datorn som finns kvar efter omstart.

Det finns mycket information om vad det finns för olika parametrar och förslag på hur de ska konfigureras, se (HTCondor Manual, 2013a). En bra idé är att ha ett lager med konfigurationsfiler för olika operativsystem om poolen innehåller flera olika (HTCondor Manual, 2013b).

5.6 Distribuering av arbete

Den distribuering som HTCondor främst är lämpad för är den sorten där ett stort problem bryts ner i flera mindre delar. Varje del beräknas sedan med samma algoritm men med olika startvärden. Det är även möjligt att dela ut arbeten som består av ett flertal olika processer som behöver kommunicera med varandra. Den typen av arbete kräver emellertid att datorerna inte avbryter arbetet, följaktligen att det finns dedikerade datorer i poolen.

För att skicka ut arbete till datorerna i poolen behövs en submit-fil. I denna fil specificeras vad som ska köras, startvärden och allt annat relevant. Krav och önskemål på datorerna som arbetet ska köras på går också att sätta in, exempelvis hur mycket minne de måste eller bör ha. Med rätt inställningar på datorerna går det att få arbeten att pausas eller flyttas till en annan dator när någon rör vid mus eller tangentbord.

Ett arbete skickas egentligen inte till en dator utan till en processor i datorn. HTCondor ser de olika processorerna som separata enheter vilket gör att en dator med fyra kärnor kan utföra fyra arbeten samtidigt. De olika processorerna skickar separata ClassAds till Central Managern, det enda som visar att de är placerade i samma dator är de sakerna de får gemensamt, som MAC- och IP-adress. Det är möjligt att konfigurera dem olika, något som i de flesta fall inte är intressant.

Ett HTCondor arbete består av en fil som körs av den dator arbetet hamnar på samt, vid de tillfällen det behövs, ytterligare filer. I de flesta fall då filen som ska köras är skrivet i något programmeringsspråk, som Java eller C, är den kompilerad. Om den är det ställs i vissa fall krav på datorn som ska köra den kompilerade filen. Exempelvis måste ett C program vara kompilerad för rätt arkitektur, 32- eller 64-bitars, dessutom måste hänsyn tas till operativsystemet - det är olika för Windows och Unix-system. Det ställs inte alltid sådana krav. Kompilerade Java (.class) filer körs på en virtuell maskin, JVM (Java Virtual Machine). Detta gör att Java program går att köra på alla datorer som har JVM installerat utan att behöva ta hänsyn till operativsystem och arkitektur, dock måste versionen på de olika JVM:erna tas i beaktande.

Oftast krävs inga förändringar i koden som ska skickas ut, allting sköts automatiskt. Det är bara i vissa fall som små ändringar kan bli nödvändiga och då endast för att få tillgång till funktionalitet utöver det faktiska utskickandet.

HTCondors olika arbeten har olika prioritet beroende på användaren som skickade ut dem. Användarnas prioritet förändras hela tiden beroende på hur många resurser de använder samt under hur lång tid. Om inga eller ett fåtal resurser används ökar prioriteten för användaren och vise versa. Arbeten med högre prioritet kan byta ut arbeten som har lägre prioritet och på så sätt ta över en dator.

5.7 HTCondors verktyg för distribuering

HTCondor använder sig av ett flertal olika verktyg för att distribuering av arbete ska fungera så bra som möjligt. I kapitlet presenteras det som användare kan komma att märka av och det användaren har mest nytta av.

5.7.1 Spara undan arbeten till ett senare tillfälle

Verktyget checkpoints används om ett arbete behöver flyttas. Dess uppgift är att inget påbörjat arbete ska gå förlorat och startas om, speciellt inte när det gäller mer tidskrävande arbeten. HTCondor har därför möjligheten att spara undan arbetets tillstånd så att det senare kan startas upp på samma ställe eller på en annan maskin, om så skulle behövas (HTCondor Manual, 2013d). En sådan fil kallas för en checkpoint och skickas till datorn som gav ut arbetet eller, som nämndes tidigare, till en checkpoint server.

Det finns två olika sätt att åstadkomma detta periodiska sparande. Program som ska köras på UNIX datorer kan länkas till ett HTCondor bibliotek som sköter allting rörande sparandet. I alla andra fall krävs ändringar i koden genom att lägga in filsparande och läsande i den (Liverpool, 2012).

5.7.2 Arbeten som är beroende av varandra

Verktyget DAGman används för att ge ut en samling arbeten där de olika delarbetena är beroende av varandra på något sätt. Exempelvis när utdata i ett delarbete är indata till ett annat. För att skicka ut arbete av den sorten behövs en speciell DAG submit-fil där noderna (alla arbeten) och kanterna (beroendena) specificeras. Efter att arbetet är utskickat kan det övervakas och administreras likt ett vanligt HTCondor-arbete (HTCondor Manual, 2013f).

5.7.3 Flytta arbete till annan pool

Arbete kan inte alltid köras i poolen, antingen för att alla resurser är upptagna eller för att inga datorer uppfyller arbetets krav. Med flocking kan de istället flyttas till en annan pool (HTCondor Manual, 2013g). Detta görs bara när ingen dator i ordinarie poolen för tillfället kan utföra arbetet. För att det ska fungera måste bägge poolerna konfigureras för att tillåta flocking till respektive från den andra poolen.

Det är inte bara möjligt att flytta arbeten mellan lokala pooler med flocking. Det går även på en större skala som mellan olika universitet eller andra som använder HTCondor.

5.7.4 Systemanrop till den utskickande datorn

En annan bra funktion som erbjuds är ''Remote System Calls". Vad den gör är att få arbetet att tro att det körs på datorn som skickade ut arbetet (Thain, 2005). I själva verket är det **condor_shadow** daemonen (körs på datorn som skickade ut arbetet) som utför systemanropen och sedan skickar svaren till klienten (HTCondor Manual, 2013e). Den kan exempelvis öppna och läsa filer som bara finns på datorn som skickade ut arbetet.

5.7.5 HTCondors virtuella maskiner

Det är möjligt att utföra arbete på virtuella maskiner som körs på datorerna i poolen. De system som stöds är VMware, Xen och KVM (HTCondor Manual, 2013h). Genom att använda sig av virtuella maskiner undviks de problem som uppstår med olika operativsystem i poolen. Det blir enklare att skicka arbeten från en Linux maskin till en Windows maskin och vice versa. Ur en säkerhetssynpunkt är det en fördel att använda sig av Virtuella maskiner. En användare kan inte råka förstöra någonting på datorn och om någon obehörig får tillgång på systemet kan den personen ändå inte göra särskilt mycket skada.

Det finns två olika vägar att gå. Antingen körs HTCondor på den virtuella maskinen, då ser HTCondor den som en vanlig nod i poolen. En nackdel är att HTCondor måste installeras och konfigureras separat både på datorn och på den virtuella maskinen. Något annat att tänka på är att om det ska köras på Windows så behövs Perl (HTCondor Manual, 2013i).

Den andra vägen är att som arbete skicka en skivavbild (eng: disk image). Då startar HTCondor upp den virtuella maskinen och den går därefter att användas.

5.7.6 Arbeten som bara körs när inget annat arbete finns

HTCondor kan konfigureras till att köra backfill-arbeten. Tanken är att arbete av denna typ bara ska utföras när datorn inte har någonting annat att göra, när ingen använder den och det inte finns några arbeten till den. Backfill-arbeten är av lägsta möjliga prioritet och körs i bakgrunden tills annat arbete anländer, antingen en användare eller ett vanligt HTCondor-arbete.

BOINC är för närvarande det enda programmet som stöds för att utföra backfill-arbete (HTCondor Manual, 2013h). BOINC måste installeras och konfigureras manuellt, det kan inte göras med HTCondor. När allting fungerar kan HTCondor konfigureras med ett par variabler till att använda BOINC som backfill klient.

5.7.7 Olika typer av arbeten

Det finns flera olika miljöer HTCondor kan köras i, olika universum (eng: universe). För varje arbete som skickas ut ska ett universum väljas i respektive submit-fil. Om inget väljs blir det universumet som är inställt som standard universumet. HTCondor använder sig av olika universum för att veta vilken slags typ av arbete det är som den arbetar med. Olika typer av arbeten ska behandlas på olika sätt och därför används dessa universum.

Det finns ett flertal olika universum att välja mellan. Bland annat finns Java-universumet för Java arbeten och VM-universumet om en skivavbild för en virtuell maskin ska skickas som arbete.

Det finns även ett universum för parallella arbeten kallat ‘’parallel” (HTCondor Manual, 2013m). MPI, Message Passing Interface, kan användas när processer körandes på olika maskiner behöver kommunicera med varandra.

5.8 Energibesparing samt väckning

HTCondor kan konfigureras till att vara miljövänlig. Systemet kan stänga ner datorer eller sätta dem i energisparläge när vissa kriterier uppfyllts, och när det senare finns arbete tillgängligt kan datorerna som behövs automatiskt väckas. För att sköta detta krävs en extra daemon på Central Managern, **condor_rooster** daemonen, som sköter väckningen av datorerna genom att skicka ut Wake-on-LAN paket (HTCondor Manual, 2013j).

Innan en dator stängs eller går in i energisparläge säger den till sin Central Manager att den gör det. Den skriver i sin tur in en ClassAd med all information som behövs för att veta om

datorn lämpar sig för arbeten som kan komma in. Om ett sådant arbete kommer, och ingen annan dator som är på kan köra det, väcks datorn (Smith, 2010).

5.9 HTCondors säkerhet

Säkerhet är en viktig del i många system. Så även i HTCondor. Dock är det svårt i HTCondor då syftet med systemet är att låta användare köra godtycklig kod på andra datorer, något som potentiellt kan förstöra eller skada data. Det är därför mycket viktigt att bara betrodda personer får tillgång till poolen och tänka på vilka privilegier de får (HTCondor Manual, 2013k).

HTCondor litar på alla som har rättigheter till att använda sig av systemet. Den säkerhet som går att konfigurera försöker se till så att ingen utomstående påverkar systemet eller använder det. För autentisering används som standard användarens id. Starkare autentiserings metoder går att konfigurera om så önskas som Kerberos och GSI (HTCondor Manual, 2013k). Det är fullt möjligt att kryptera all information som skickas över nätverket, både data från användaren som arbete och HTCondors egna kommunikationer. Till detta går det att lägga till kontrolldata som visar som någon mixtrat med datan.

Det finns ingen möjlighet att via HTCondor kryptera data som sparats på hårddisken. Det erbjuds heller ingen kontroll som visar om någon ändrat på sparad data på något sätt.

5.10 HTCondors skalbarhet

Tidigare, i förstudien, visades exempel på universitet som använder sig av HTCondor, både ensam och i kombination med andra system. Det innebär att HTCondor kan fungera bra i större system och kan samverka med annan programvara. Flocking gör det möjligt att ha ett flertal pooler som samarbetar med varandra, någonting som är nödvändigt för större organisationer som universitet (en Central Manager skulle få problem att klara all data och nätverkstrafik).

Utöver detta klarar HTCondor att använda sig av datorer som inte har HTCondor installerat utan annan liknande programvara. För att uppnå detta ska universumet ''grid" anges i submit-filen. Även vilken typ av programvara som körs på resurserna som ska användas måste anges. HTCondor kan skicka ut arbete och övervaka dem på ett flertal olika sorters system. Bland annat stöds system som bygger på Globus Toolkit men även ARC av Nordugrid, Unicore och PBS med mera (HTCondor Manual, 2013l).

HTCondor utvecklas fortfarande och gränserna för vad systemet klarar i form av belastning pressas hela tiden uppåt. En jämförelse har utförts på skillnaden mellan olika versioner av HTCondor (Bradley, 2011). Jämförelsen visade på stora skillnader på hur effektivt HTCondor kunde hantera olika belastningar och stora förbättringar kan ses. Ett test som kördes var att låta en submit-maskin köra för fullt och se hur många arbeten som kunde skickas ut per sekund. Arbeten tog mindre än en sekund att utföra och datorn hade som uppgift att se till att 500 stycken alltid var igång. Från version 7.4 till 7.5.5 ökade antalet arbeten per sekund med över 30 procent. Tester kördes även på Central Managern och framsteg visades även där, bland annat gjordes tester för att se hur många arbeten som kunde paras ihop per sekund. Mer förbättringar har utförts sedan testerna utfördes och systemet har blivit

än mer effektfullt, HTCondor är uppe i version 7.9.6.

HTCondor tar inte upp mycket plats, mellan 100 till 500 MB efter installation beroende på vad som installerats och på operativsystem. Istället för att installera på varje dator ges möjligheten att ha en central installation på ett distribuerat filsystem med hjälp av HDFS, Hadoop File System (Apache Hadoop, 2012).

6 Implementering av HTCondor och Wake-on-LAN

Här beskriver vi hur ett fungerande system med HTCondor och Wake-on-LAN kan sättas upp. Först presenteras olika sätt att installera systemet på och vad som bör tänkas på. Därefter följer en beskrivning hur en grundläggande men fungerande konfigurering görs. Sist finns en kort guide till hur en dator ska ställas in för att kunna väckas med Wake-on-LAN.

6.1 Installation av HTCondor

Första steget är att ladda ner programvaran från HTCondors hemsida. Där erbjuds färdigbyggda installationsfiler för Windows och även RPM och DEB paket för Linux system. Det erbjuds också möjligheten att bygga sitt eget paket med hjälp av källfilerna. Utförandet skiljer sig sedan beroende på vilket installationssätt som valts.

Det finns några saker som måste bestämmas innan installationen av en HTCondor-pool påbörjas. Vilken dator som ska bli Central Manager är någonting att tänka på. Den maskinen borde alltid vara på, stängs den av eller kraschar kommer inte nya arbeten kunna köras. Arbeten som redan körs kommer dock inte att påverkas. Många av HTCondors verktyg och kommandon kommer också sluta att fungera för alla datorer i poolen om deras Central Manager försvinner.

Trafiken på nätverket är också någonting att ta hänsyn till. Daemons som kör på poolens datorer skickar uppdateringar om sitt tillstånd till Central Managern, som standard med 5 minuters intervall. Det innebär att en Central Manager bör placeras på en plats i nätverket där det inte blir några problem med tillgängligheten.

Hur mycket minne som tas upp av Central Managerns uppgifter beror på antalet datorer i poolen. Siffror från HTCondors hemsida: 100 datorer tar ca 25MB och 1000 datorer i poolen ca 100MB.

6.1.1 Installation på Windows

För att installera HTCondor finns det vissa krav på operativsystemet. När det gäller Windows så måste det antingen vara Windows 2000 SP4, Windows XP SP2 eller nyare. Windows behöver också ha Visual C++ 2008 installerat. HTCondor installeras och körs av LocalSystem kontot, att ändra detta kan leda till problem och bör därför inte göras. Arbeten som HTCondor genomför kör dock med normala användarrättigheter.

På en Windowsdator är det enklaste sättet att använda .msi filen. Det resulterar i en vanlig GUI installation med val av olika funktionaliteter i systemet. Allt som ställs in här går att senare att ändra på.

Det är inte alltid en installation med GUI önskas. Speciellt inte om det är något som ska installeras på många datorer. MSI-filer går att köra i en terminal där den använder en fil som inparameter med valen på alla frågorna. Detta sker med msiexec-kommandot. (Microsoft, 2011). Det går även att installera HTCondor manuellt genom bland annat att ändra i registret (HTCondor Manual, 2013b), detta har dock lämnats otestat.

Om det finns behov av att ändra i källfilerna eller någon annan orsak finns att inte använda de färdiga installationsfilerna erbjuder HTCondor källfilerna till systemet. Bygga om HTCondor med Windows är inte det enklaste och det behövs en rad olika installerade program på datorn för att klara av det. Ingen närmare undersökning i ämnet gjordes på Windows.

6.1.2 Installation på Linux

Hur installationen av HTCondor går till skiljer sig mycket beroende på vilken installationsmetod som väljs. Det enklaste är att lägga till HTCondors datakatalog (eng: repository) i sin pakethanterare genom att lägga till en rad i dess katalog. Ett lika enkelt alternativ är att ladda ner ett RPM eller DEB paket från hemsidan som sedan kan installeras med pakethanteraren.

De sista alternativen är att ladda ner ett tar paket eller källfilerna, det valet kräver mer arbete för installationen. Tar-filen måste byggas vilket ofta kan göras med ett par enkla kommandon men källfilerna behövs byggas till ett paket eller installation först, något som är betydligt svårare. Ett test av det har framgångsrikt genomförts.

För att bygga HTCondor behöver ett antal program vara installerade, viktigast är GNU gcc och g++ kompelatorerna. Några av de andra program som krävs finns listade i en fil, README.building, men konstigt nog inte alla. Om det saknas nödvändiga program kommer byggandet att krascha med, ibland, kryptiska meddelanden vilket ska tydas och åtgärdas. De program som saknas i README.building är libpcre3, libpcre3-dev, libssl-dev (för openssl), uuid-dev samt curl.

Det är rekommenderat att installera med en ny användare, condor, på varje dator som äger de filer HTCondor skapar. Det kontot ska inte gå att logga in på utan ska enbart användas av HTCondor. Ett standardsätt att åstadkomma det är att ha ett lösenord med ogiltiga tecken i. Vid installation med RPM och DEB paket löser HTCondor hjälpsamt det själv.

6.2 Grundläggande konfigurering av datorer med HTCondor

En av OptiPlex datorerna konfigurerades till server som delar ut arbeten och de andra användes huvudsakligen som klienter vars uppgift varit att utföra de arbeten som delades ut. Mer specifikt har servern varit konfigurerad som Central Manager och submitter medan de

övriga varit executers.

För att få datorerna att utföra olika uppgifter ska de konfigureras rätt i respektive dators lokala konfigurationsfil. Det som alltid krävs i dessa är: vilken dator som är deras Central Manager och vad de ska köra för daemons. Alla ändringar görs i varje dators lokala konfigurationsfil.

condor_host = Central Managerns namn eller IP-adress här. Central Managern själv kan utnyttja en av de redan fördefinierade variablerna genom att sätta *$(FULL_HOSTNAME)* här.

daemon_list = *MASTER* ska alltid vara med här. Sedan är det olika beroende på vad datorn ska göra. För att kunna utföra arbeten ska *STARTD* vara med, skicka ut arbete kräver *SCHEED* och Central Managern behöver *COLLECTOR* och *NEGOTIATOR*. För en dator som ska kunna både skicka och utföra arbeten kan listan se ut såhär: *MASTER*, *SCEED*, *STARTD*. Mer detaljer om de olika daemons som kan vara aktuella och vad de gör finns i 5.2 Uppbyggnad av en HTCondor-pool.

allow_read variabeln används för att bestämma vilka datorer som ska ha läsrättigheter, rättigheter för att se status på poolen. Om den sätts till att ge alla läsrättigheter genom att sätta den till ''*" ges det till hela världen. Bättre är att bara ge Chalmers rättigheter genom exempelvis *.chalmers.domän eller 129.16.*..

allow_write ger i sin tur rättigheter att gå med i poolen samt skicka ut arbete. Samma saker som för *allow_read* gäller här.

6.3 Inställningar för Wake-on-LAN

För att få Wake-on-LAN att fungera på sin dator behöver en del saker konfigureras. En sak måste alltid göras, det är att i BIOS ställa in så att datorn kan väckas av magiska paket. Denna inställning ligger på olika ställen beroende på vad för BIOS som används samt dess version. Stödjer inte nätverkskortet Wake-on-LAN finns här ingen inställning. När detta är gjort skiljer sig förfarandet åt beroende på operativsystem.

6.3.1 För Windows

I Windows måste en inställning göras på nätverkskortet för att det ska gå bra att väcka datorn med Wake-on-LAN. Detta är lättast att göra genom att gå in i ''Device Manager" och sedan till egenskaper på nätverkskortet. Under ''Advanced'' finns en egenskap som heter ''Wake on Magic Packet''. Den ska vara påslagen. Det är inte alltid denna finns men då är det troligen påslaget ändå om det stöds.

Det finns ett till ställe att ändra i. Fortfarande hos egenskaperna på nätverkskortet men nu under ''Power Management". Där ska ''Allow this device to wake the computer" vara kryssad. För lite extra säkerhet går det även att kryssa i att bara ''magic packets" får väcka datorn.

6.3.2 För Linux

För att kunna använda Wake-on-LAN behövs ett ges kommando till datorn, men detta kommando håller bara tills nästa gång datorn startat. För att lösa detta kan ett skript göras som körs varje gång datorn startas.

Kommandot som ska köras är: *ethtool -s eth0 wol g*. Det första som behöver göras är att ta reda på är vilket nätverksgränssnitt som ska väcka datorn och byta ut *eth0* till det. Om flera olika gränssnitt ska kunna användas går det att köra kommandot upprepade gånger. Därefter läggs allting in i en den fil som ska köras varje gång datorn startas. Processen för att uppnå detta är olika beroende på vilket operativsystem som används, filen ska läggas på olika ställen och olika kommandon ska köras.

7 Test och användning av HTCondor

Här beskrivs det praktiska test av HTCondor som vi genomfört och rekommendationer om hur systemet bör användas. Exempel på hur önskvärd funktionalitet fås finns också i kapitlet. I all användning och alla tester har version 7.8.7 av HTCondor använts.

7.1 Distribuering av arbete

När arbete distribueras med HTCondor så skickas det till en tråd på den dator som ska utföra arbetet, vilket innebär att en dator kan utföra lika många arbeten samtidigt som den har kärnor. För att distribuera arbete så används kommandot *condor_ submit*, som tar en submit-fil som argument. En submit-fil beskriver hur arbeten ska utföras och vilka operativsystem och arkitekturer som kan utföra dessa arbeten. Exempel på olika submit-filer för C, Java och Matlab finns i Bilaga A. Gemensamt för alla submit-filer är att det måste specificeras vad som ska köras på måldatorn, vilket görs med parametern *executable*, det vill säga den körbara filen. För att ge indata till denna fil används *argument*, värdena i *argument* separeras med blanksteg. Det första värdet ska vara namnet på den körbara filen och sedan följt av dess indata. När arbetet ska utföras flyttar HTCondor den körbara filen till måldatorn och exekverar den med argumenten.

I submit-filen bör det även stå vilket universum som ska användas, detta beskriver för HTCondor vilka slags krav som ställs på arbetet. Om inget universum anges så används som standard ett som heter vanilla. Men eftersom det är möjligt att ändra vilken som är standard så är det rekommenderat att alltid ange vilket universum som ska användas. I projektet används huvudsakligen tre olika universum, dessa är java, standard och vanilla. Den största skillnaden mellan standard och vanilla är att standard använder sig av checkpoints för att kunna återuppta arbeten vid ett senare tillfälle. I de fall som arbeten inte ska eller kan återupptas bör vanilla istället användas. Det gäller för bland annat kommersiella program där källkoden inte är tillgänglig, eftersom program som ska använda standard behöver kompileras om med kommandot *condor_ compile*.

Genom att sätta ett värde efter parametern *queue* så bestäms det hur många gånger detta arbete ska utföras med de redan angivna argumenten. Det går även att upprepa *argument* och *queue* efter varandra för att slippa skapa flera submit-filer. Om argumenten på de olika arbetena följer ett mönster, exempelvis 2, 4, 6, 8, kan variabeln *process* användas. Variablen tilldelas ett nummer med start från 0 och ökar sedan med 1 för varje arbete i *queue*. Med andra ord har varje enskilt arbete ett unikt nummer tilldelat. Variabeln kan sedan multipliceras, adderas med mera för att få önskad talföljd.

För att specificera vilka krav på till exempel arkitektur och operativsystem ett arbete har används parametern *requirements*, vilket har använts för Matlab arbetet i Bilaga A. Om det inte ska vara ett krav men någonting den gärna får ha, används parametern *rank* istället.

Resultatet som skickas tillbaka när ett arbete är utfört skrivs till filen som anges av parametern *output*. Dess innehåll består då av det som skrevs till standard output (stdout). På samma sätt som för output fungerar parametern *error*, som använder standard error (stderr) istället. HTCondor skickar även tillbaka alla nya filer som skapades när arbetet kördes. För att specificera filer som ska flyttas tillbaka kan *transfer_ output_ files* användas.

HTCondor sköter automatiskt förmedlingen av arbete och ser till att arbeten inte sprider sig och skickas till fler datorer än tänkt, vilket skulle vara slöseri av kapacitet. Däremot om HTCondor tappar kontakten med datorn som utför arbetet så kan den skicka samma arbete igen, men till en annan dator.

7.1.1 Köra program skrivna i C

Arbeten skrivna i C kan använda både universumet standard och vanilla, dock om standard ska användas så behöver programmet kompileras om. För att göra detta används *condor_ compile*, som skrivs framför kommandot som annars hade använts. Som till exempel hade *gcc main.c -o main* blivit *condor_ compile gcc main.c -o main*. Se Bilaga A för exempel på en submit-fil

7.1.2 Köra program skrivna i Java

För att utföra ett arbete i Java så används ett universum som heter java, det talar om för HTCondor att alla datorer som har Java installerat kan utföra arbetet oavsett arkitektur. I *executable* anges vilken Java-klass som ska användas. I Bilaga A visas en submit-fil för Java arbete.

För att HTCondor ska kunna utföra denna typ av arbete ska Java vara installerat på datorn. Det visade sig inte räcka på Windows utan ett par rader behövde läggas till i konfigureringsfilen. På Linux fungerade det utan extra tillägg.

java = Ska peka på java.exe. Ligger vanligen i system32 mappen.
java_ classpath_ separator=";''

7.1.3 Köra program skrivna i Matlab

Att distribuera arbete som använder Matlab är något mer avancerat än för C och Java, det räcker inte bara med en submit-fil utan det krävs även en skriptfil för att starta Matlab. Denna skriptfil skiljer sig mellan Linux- och Windowsdatorer och finns beskriven i Bilaga B. Eftersom det krävs olika skript för att starta Matlab, men de delar submit-fil så behövs ett sätt att bestämma vilket skript som ska användas när arbetet skickas. Detta kan göras genom att använda *$$(OpSys)* i submit-filen i namnet på skriptet, när submit-filen tolkas så översätts *$$(OpSys)* till operativsystemet på datorn (Windows eller Linux) som blev tilldelad arbetet och rätt skript skickas med. För att detta ska fungera krävs det att skriptet har filändelsen .bat, annars tolkar inte Windows det som ett skript. Det behöver även specificeras vilka filer utöver skriptet som ska skickas till måldatorn, detta används *transfer_ input_ files*

till. Om det behöver skickas flera filer så används kommatecken för att separera dem.

Viktigt att tänka på är även att Matlab ska ha ''quit" på slutet i skriptet, annars finns det risk för att det aldrig kommer att avslutas och inte skicka något resultat av arbetet tillbaka. Det universum som ska användas är vanilla, men detta universum är väldigt grundläggande och HTCondor erbjuder då inte stöd för checkpoints. Men detta kan lösas genom att använda Matlab skript som kan återuppta avbrutna arbeten.

En sådan Matlab-fil behöver spara hur långt beräkningen kommit till en fil då och då. Som in parameter behöver den ta endast en sådan fil. Då kan beräkningen fortsätta från den sparade filen vid ett senare tillfälle eller på en annan dator.

Submit-filen fungerar nu istället så att den startar skriptet med argumenten som i sin tur startat Matlab med samma argument. För att det ska fungera korrekt att skicka Matlab arbeten till både Linux- och Windowsdatorer behövs det även talas om för HTCondor att dessa datorer kan hantera sådana arbeten. Detta sätts med parametern *requirements* som syns i exemplet i Bilaga A.

Om Matlab inte finns installerat på datorn så går det att lösa på andra sätt som till exempel att använda Octave istället, ett mindre program som kan köra Matlab-filer. Octave kan skickas med arbetet och sedan köra Matlab programmet på datorn. Hur detta uppnås beskrivs här nedan.

7.1.4 Köra program skrivna i andra språk

På liknande sätt som för Matlab så går det att skapa ett skript för att starta ett annat program och på så sätt går det utföra arbeten i andra språk än de ovan nämnda. Om den önskade programvaran eller filer saknas på de datorer som ska utföra arbetet så kan *transfer_input_files* användas till att skicka med dessa.

Praktiskt har detta testats med programmet Octave som kan köra de flesta Matlab-filer. Octave skickades med i en zip-fil och för att sedan köra arbetet packades programmet först upp, utförde arbetet med hjälp av Matlab-filen och togs sedan bort. Tillvägagångsättet är detsamma oberoende av språk och program som ska köras. Exempel på submit och exekveringsfil finns i Bilaga C.

7.2 Prioritering av de fysiska användarna

HTCondor har ett antal variabler som påverkar när och under vilka omständigheter arbeten får startas och avbrytas. Det finns främst sju stycken variabler som är intressanta. Om variabeln *start* sätts till sant är det tillåtet för HTCondor att starta ett arbete på den maskinen. De andra variablerna kan på olika sätt påverka arbetet när det väl har startat. *suspend* pausar arbetet och *continue* återupptar det igen. *preempt* avbryter arbetet och flyttar det för att köras på en annan dator och *kill* stänger bara ner det, något som kan vara bra om ett arbete hängt sig.

Det finns två variabler till, *want_suspend* och *want_vacate*. De agerar främst som komplement till *suspend* och *preempt* och bestämmer vilken variabel som ska evalueras. Ett exempel är när *want_suspend* är falsk, då kan inte variabeln *suspend* påverka systemet. En

fullständig överblick av sambanden mellan variablerna finns i figur 7.1.

I figuren finns ett tillägg till de vanliga sambanden. Det är att om ett arbete varit pausat över en viss tid, *maxSuspendTime*, så ska de flyttas eller tas bort beroende på vad *want_vacate* avgör.

Vid uppsättning och testning av systemet är det enklast om arbeten startar direkt och aldrig avbryts. Detta uppnås enklast genom att sätta *start* till sant och resterande variabler till falskt. I de flesta fall är den typen av konfiguration inte intressant. Istället finns det krav på när arbeten ska pausas, återupptas, tas bort eller flyttas. Nedan följer en redogörelse hur systemet bör fungera. Se Bilaga D för kod som uppnår dessa riktlinjer.

För att HTCondor ska få lov att starta ett arbete ska ingen ha använt datorn under en viss tid. Om det är någonting som körs som använder sig av datorkraften får arbetet dock inte starta. Detta för att inte störa längre beräkningar som skulle kunna vara igång på datorn. Det behövs en kontroll här för att se om det är HTCondor som använder datorkraften eller något annat program. Om det är HTCondor bör ett nytt arbete få starta, det gör att arbeten med högre prioritet kan byta ut det arbete som körs. Det går även att ställa in så att arbeten bara får starta på natten eller andra specifika tider.

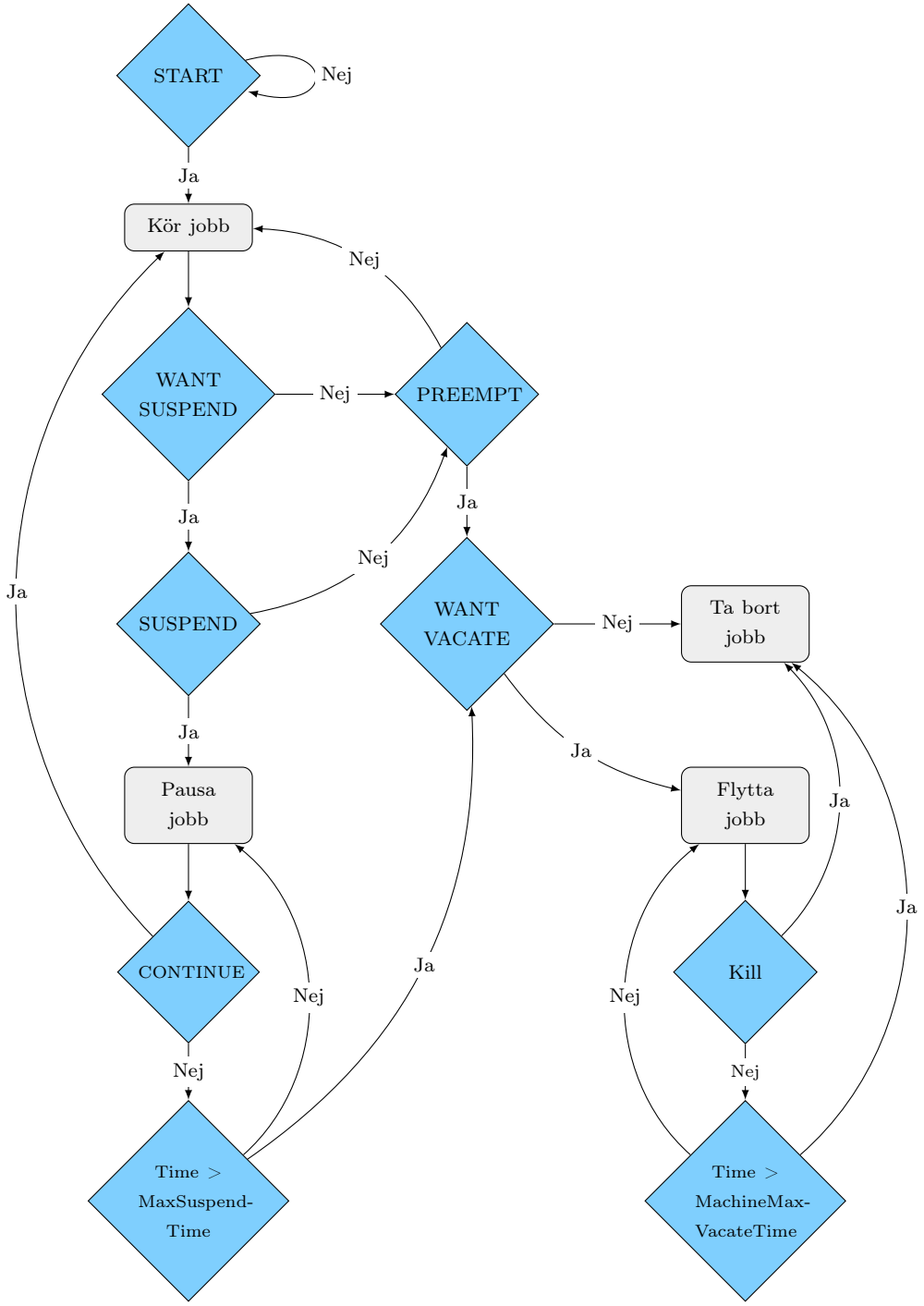
Ett arbete ska pausas när en användare rör musen eller använder tangentbordet, men om det är ett stort arbete ska det istället flyttas till en annan dator. Om arbetet varit pausat under en längre tid kommer användaren med stor sannolikhet fortsätta använda datorn. Då ska arbetet istället flyttas.

För att ett arbete ska återupptas ska användaren ha gått ifrån datorn. Enklaste sättet att se om användaren är borta är genom att se hur länge sedan det var musen eller tangentbordet användes. Detta sköter en daemon, **condor_kbdd**, som körs automatiskt på Windows men måste läggas till i listan på Linux. För att HTCondor ska få återuppta ett arbete ska processorn bara används av de processer som alltid är på normalt. Eventuella beräkningar eller dylikt som användaren kan satt på ska inte störas.

Det är aktuellt att ta bort ett arbete och börja om det från början när det knappt hunnit påbörjas. Arbetet att spara och flytta arbetet blir då mer krävande än att helt enkelt starta om. Det finns ett till tillfälle när det är aktuellt och det är när ett arbete håller på att flytta men gjort det under en lång tid. Detta är för att förhindra att arbeten fastnar i flytten och ligger kvar i det stadiet.

7.3 Energibesparing och väckning med Wake-on-LAN

För att få ett fungerande system där datorer går in i energisparläge eller stängs av när de inte används krävs en del konfigurationer, både på Central Managern och på resten av datorerna i poolen. Tanken är att datorerna själva ska gå in i energisparläge när vissa krav uppfylls och sedan ska Central Managern väcka dem när de behövs.



Figur 7.1: Variabler som kan påverka när och hur jobb utförs.

7.3.1 Central Manager

Central Managern är den som ska väcka datorer om det kommer arbeten till dem. Detta sköts av en en daemon kallad **condor_rooster** som ska läggas in i listan med alla aktiva daemons, *daemon_list*.

Inställningar ska göras med val om när datorer ska väckas. Det görs med två variabler. Den ena, *rooster_interval*, styr hur ofta daemonen ska se efter om det finns datorer att väcka. Om *rooster_unhibernate*, den andra variabeln, evalueras till sant vid en sådan kontroll skickas ett Wake-on-LAN paket till den datorn för att väcka den. Variablerna skulle kunna se ut såhär:

*rooster_unhibernate = offline && unhibernate rooster_interval = 5 * $(minute)*

Exemplet visar ett kontrolleringsintervall på 5 minuter vilket startar datorer relativt snabbt samtidigt som det inte är tillräckligt ofta för att fylla för mycket av Central Managerns tid. *offline* är sann om datorn är av och *unhibernate* en variabel som ställs in på varje enskild dator, olika datorer kan sätta olika krav på när de får väckas.

För att allting ska fungera måste dessa värden finnas tillgängliga för Central Managern när datorerna är av. För att kunna veta om ett arbete kan utföras av datorn behövs en hel del mer information utöver detta som operativsystem, om det finns java och mycket mer. För att väcka datorer med Wake-on-LAN behövs även deras Mac-adresser. När en dator går in i energisparläge skickar den en ClassAd som innehåller all relevant information. Informationen sparas sedan undan av Central Managern till en fil. Filen som ska användas måste anges i en variabel.

collector_persistent_ad_log = /path/to/file

7.3.2 Övriga datorer i poolen

Som nämndes tidigare finns en variabel vid namn *unhibernate* där varje dator kan specificera villkor för att bli väckt. Alla tester vi har gjort har haft variabeln satt till default vilket gör att datorn väcks när den tilldelats ett arbete.

På samma sätt som Central Managern hade två variabler, en för tidsinterval och en som bestämmer om något ska göras, har varje dator två variabler. Med *hibernate_check_interval* bestäms hur lång tid det ska gå mellan varje kontroll om datorn ska gå in i energisparläge eller inte. För att försätta datorn i energisparläge vid en sådan kontroll ska *hibernate* vara en sträng som representerar det energisparläge som ska användas.

De strängar som kan användas för *hibernate* är:

''S1"	''STANDBY"	: CPU:n stannar och dess cache töms.
''S2"		: CPU:n stängs av.
''S3"	''RAM"	: Strömsparläge. RAM-minnena har fortfarande ström.
''S4"	''DISK"	: Viloläge. Sparar allting på disk och stänger av sig.
''S5"	''SHUTDOWN"	: Datorn stänger av sig.

Nedan är ett förslag på hur en dator skulle kunna konfigureras för att själv försätta sig i energisparläge efter lämplig tid. För att förtydliga vad som är vad i exemplet finns två

hjälpvariabler. Om *should_hibernate* sätts till sant går datorn in i det energisparläge som angivits i *hibernate_state*, om den blir falsk händer ingenting.

```
hibernate_check_interval  =  5 * $(minute)
should_hibernate          =  ( (KeyboardIdle >15 * $(minute))
                             && $(StateTimer) >15 * $(minute)
                             && (State != ''Claimed") )
hibernate_state           =  ''RAM"
hibernate                 =  ifThenElse( $(should_hibernate),
                             $(hibernate_state), ''NONE" )
```

should_hibernate visar ett fullt fungerande sätt att ställa in när en dator ska gå in i energisparläge. Den evalueras här till sant om ingen använt mus eller tangentbord på 15 minuter samt att inga arbeten från HTCondor utförts under den tiden.

På Linux datorer finns det ytterligare en sak till som måste läggas till. En till daemon, **condor_kbdd**, som håller koll på om någon använder mus eller tangentbord. Windows datorer kör denna daemon per automatik.

8 Användargränssnitt för HTCondor

Av de distribuerande system som har undersökts så har de flesta inte haft ett GUI, vilket heller inte HTCondor har, utan all konfiguration och exekvering sker via en terminal. Detta fungerar bra för de som är erfarna med systemet men för nya ovana användare kan det vara krångligt, därav är ett GUI av stor vikt för ett användarvänligt system för HTCondor. Detta ger två möjliga alternativ, antingen att implementera ett nytt GUI från början eller att använda ett redan existerande. Vi har utvärderat ett komplett GUI, CycleServer, men vi tog beslutet att inte använda det utan istället skapa ett GUI från början med bästa lämpliga API som beskrivs i sin helhet i detta kapitel. Anledningen att CycleServer valdes bort var att de inte erbjuder vidareutveckling eller modifiering av programmet (Cycle Computing, 2013).

8.1 API för GUI

För att underlätta implementeringen av ett GUI så undersöktes möjligheterna att använda ett API, för att på så sätt slippa hantering av HTCondors källkod. GUI:t ska skrivas i Java och därför måste API:t också vara skrivet i detta språk, vilket kan medföra problem då HTCondor är skrivit i C++. Tanken med API:t är att alla de funktioner som HTCondor har ska gå att anropa via Java och att då också slippa konverteringen mellan C++ och Java. Två API:er undersöktes som båda hade olika för- och nackdelar.

Det första API:t som undersöktes var HTCondors egna Web Service API, som innebar att GUI:t kunde vara tillgängligt från en central sida i nätverket. Det fanns mycket information om detta API men valet gjordes att inte använda det då API:t krävde anrop från SOAP som är ett protokoll som är använder XML och HTML-kod (HTCondor Manual, 2013n).

Det andra API:t som undersöktes var skrivet i Java av en forskare från ett "Information Technology Research Institute" i Japan (Nakada, 2012). Det fanns bra exempel och var enkelt att använda. Dock hade detta API inte lika bra dokumentation som HTCondors egna, inte heller verkade det finnas lika många funktioner. Det var helt öppet och hade bra möjligheter för fortsatt utveckling (Nakada, 2009). Därav valet att använda detta API vid implementeringen.

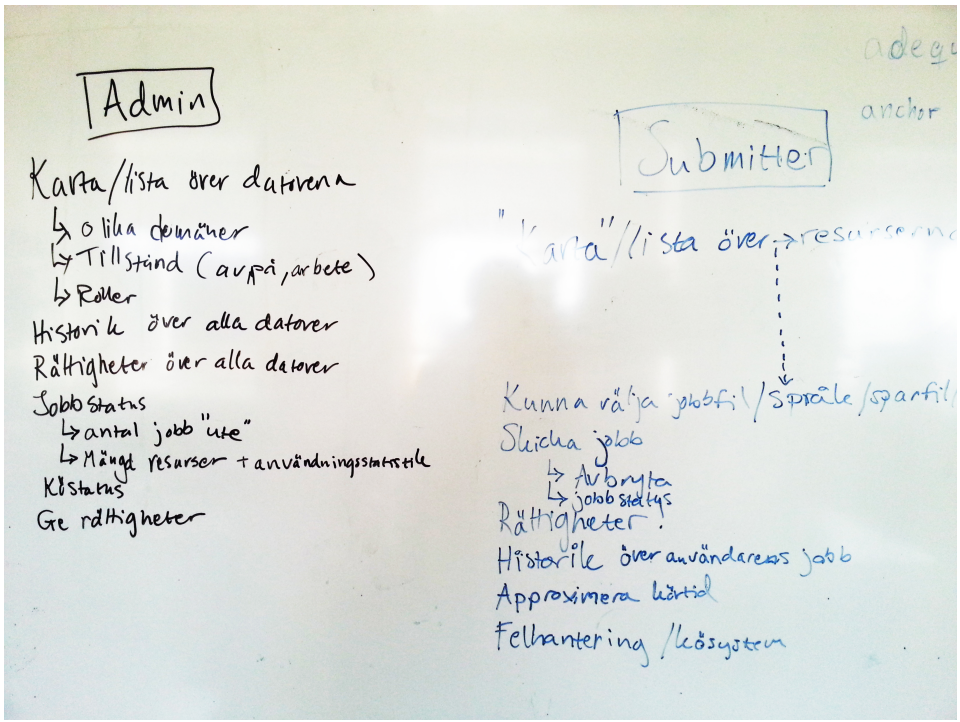
8.2 Utvecklingen av GUI:t

När ett API fanns att tillgå med de funktioner som behövdes kunde utvecklingen av GUI:t börja. Det första steget var att skissa på hur GUI:t skulle fungera och efter detta göra olika prototyper och börja implementeringen i Java. GUI:t har de funktioner som anses vara mest

användbara för den vanliga användaren. Inget GUI utvecklades för administratörer då denna förutsätts att ha goda kunskaper om hantering av HTCondors kommandon via terminalen.

8.2.1 Första skissen

Det här var en skiss med funktioner på både ett användare- och administratörsgränssnitt, se figur 8.1. Det var många funktioner med som senare togs bort för att de inte behövdes eller var för svåra att implementera. Den första tanken var att ha en lista och karta över alla datorer som fanns på Chalmers där det skulle gå att se vilka datorer som tillhörde varje datorsal eller institution. En karta hade varit snyggt grafiskt att använda men vi valde senare att lista alla datorer i en tabell istället då det är mer användbart. De viktigaste funktionerna var information om alla datorer och arbeten som kördes i systemet. Egna arbeten skulle också gå att ta bort och historik skulle finnas. Både administratörer och användare skulle ha dessa funktioner. Förutom att administratörer hade alla funktioner som en användare hade kunde också administratören lägga till och ge rättigheter till användare samt avbryta arbeten.



Figur 8.1: Första skissen på GUI:t

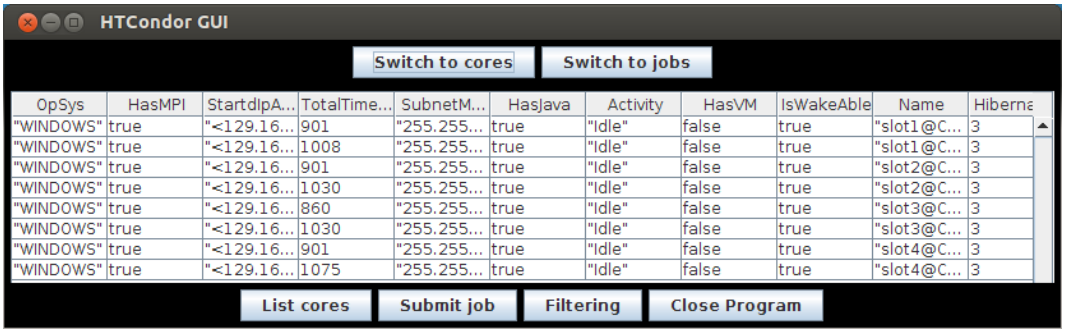
8.2.2 Prototypen

Prototypen utformades med de funktioner som fanns i skissen. Fokus för gränssnittet var, som nämnt tidigare, att det skulle vara enkelt att använda och ha de funktioner som var nödvändigast. Därför utformades den första prototypen med en stor ruta där den datorkraft som fanns i systemet listades i en tabell med tillgänglig information. Användaren av GUI:t

kunde välja att antingen visa aktiva arbeten eller den datorkraft som finns i rutan och hade även möjlighet att ta bort de arbeten som användaren körde. Skicka arbete till systemet kunde göras enkelt genom att trycka på en submit-knapp och välja den fil som ska köras i ett popup-fönster. Filtrering gjordes också genom att i ett popup-fönster välja vilken information som skulle visas. Vid start av programmet skulle programmet visa den information som är mest väsentlig.

8.2.3 Slutliga produkten

Implementation av GUI:t utformades nästan helt från prototypen, se figur 8.2 Det var bara små ändringar som gjordes som inte tidigare hade tagits i beräkning. All information och kod skrevs på engelska för vara anpassad till alla på Chalmers. HTCondor API:t, som nämnt tidigare, användes och hjälpte till mycket men det fanns en del avsaknad av metoder i API:t som då fick kodas.



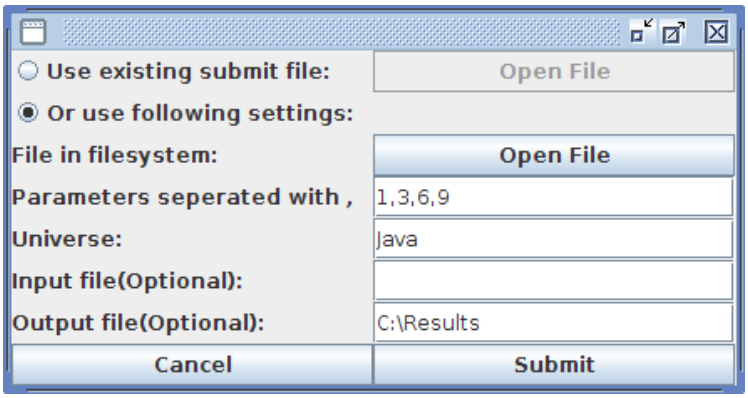
Figur 8.2: Listade datorer i HTCondor GUI:t.

Skicka ut arbeten

Ett arbete skickas ut genom att användaren väljer att skicka ett arbete genom GUI:t, se figur 8.3. Då kommer ett popup-fönster upp som ber användaren att leta upp antingen en submit-fil eller en körbarfil som Java, C eller Matlab. Därefter måste användaren ange om den laddat upp en submit-fil eller körbar fil, varav GUI:t skapar en om det behövs. Skulle fel fil väljas eller något annat fel inträffa kommer ett felmeddelande upp som ber användaren göra om proceduren.

Lista Information om tillgänglig datorkraft

Detta är huvudfönstret där antingen information om den datorkraft som finns eller arbeten visas. All datorkraft visas, både den som är igång och avstängd. När programmet först öppnas finns det ett antal kolumner med information om varje tillgänglig datorkraft, som t.ex. datornamn, möjlighet att köra Matlab, processortyp med mera. Det går att lista varje kolumn i bokstavsordning, antingen som stigande eller fallande. För att det gör det lättare för användaren att hitta hur mycket datorkraft som är tillgänglig för en viss typ av arbeten.



Figur 8.3: Utskick av arbete där GUI:t själv skapar submit-filen

Filtrering

För att underlätta listan med tillgänglig kraft ytterligare, finns filtrering. Där kan användare välja vad som ska visas i listan, där kolumner går både att lägga till och filtrera bort. All information som finns om varje dator går inte att se då mycket har gallrats bort då det inte är intressant. Om användaren vill ha mer information om varje datorkraft går detta enkelt att lägga till i källkoden.

Lista och hantering av aktiva arbeten

De arbetena som för tillfället körs i systemet kommer att listas i huvudfönstret med informationen om vilken datorkraft som används och vilka inparamterar arbetet körs med. Användaren har också möjlighet att för varje arbete ta bort detta om det inte längre är relevant. Då GUI:t hämtar information från HTCondor kommer aldrig någon information gå förlorad gällande arbeten som körs så länge inte HTCondor stängs av.

9 Energi och miljö för Chalmers datorer

I det här kapitlet presenteras först ett exempel på energihantering från ett annat universitet som följs av en intervju som vi har gjort med IT-administrationen för att undersöka varför datorer inte stängs av. Efter intervjun kommer lösningsförslag och argumentation mot de problem som uppstår vid avstängning och uppstart av datorer. Kapitlet avslutas med en studie som vi gjorde för att utreda hur datorerna skall stängas av och vilka besparingar Chalmers kan göra.

9.1 Energihantering av datorer på andra universitet

Det finns många exempel på andra skolor som med hjälp av program, stängt av datorer och fört statistik över energiåtgången för att minska kostnaden. Ett exempel är ''University East Angelia" (UEA) som är ett stort universitet som har över 5 000 datorer på campus, som studenter, lärare och annan personal äger. De ville sänka sin energikostnad för datorer som inte används för att spara pengar och minska miljöpåverkan.

Detta gjorde de med hjälp av ett program som samlade statistik och stängde av datorer som inte behövdes. Ett försök gjordes på 500 datorer och resultatet var lyckat, de sparade 40% av den ursprungliga energimängden som datorerna drog. Detta innebar en minskning på 132 000 kWh per år som gav en ekonomisk vinst på över 150 000 kr. Tack vara det positiva resultatet så försattes många fler datorer i energisparläge och den ekonomiska vinningen som alstrades ökade proportionellt mot andelen nya datorer.

UEA är bara ett av många andra universitet som installerat ett system för energiadministration och uppnått goda resultat som både gynnar skolan och miljön. De negativa effekterna som finns att hitta och kan vara skälet till att inte alla organisationer med många datorer redan har ett system för avstängning av datorer kan bero på flera faktorer. De självklaraste som finns är att någon dels behöver installera ett system som ska vara säkert och pålitligt och det bör också vara användarvänligt. Därefter behöver systemet hanteras för vardaglig administration. Detta kostar och kan kräva särskild kompetens som kanske inte finns för tillfället.

Då finns lösningen som universitet UEA och många andra gjort, att köpa en produkt och tjänst som sköter detta åt organisationen. Det betyder att skolan kan lämna över allt arbete till ett externt företag. Det innebär ofta en liten kostnad som många organisationer inte är villiga att betala, om detta beror på okunskap eller att den beräknade ekonomiska omsättning är mindre än den kostnad som uppstår vid användning av dessa produkter är svårt att säga (Mosley, 2010).

9.2 Energihantering av datorer på Chalmers

Då det finns mängder av datorer på campus kan Chalmers ha mycket att vinna, både ur miljö- och ekonomiskt perspektiv, om datorer försätts i energisparläge. Här beskriver vi först de problem som kan uppstå när Chalmers stänger av sina datorer. Därefter kommer förslag för att lösa de problemen och till sist en studie som visar hur mycket Chalmers kan spara och vilket energisparläge som är mest lämpligt att använda.

9.2.1 Anledningar till att Chalmers inte stänger av sina datorer

Orsaken till att Chalmers inte försätter sina datorer i energisparläge beror på en del problem som kan uppkomma om datorerna stängs av, hävdar IT-administration på Data och IT institutionen. De menar först och främst att Chalmers datorer måste vara tillgängliga dygnet runt till skolans studenter och forskare. Med andra ord att en användare aldrig ska behöva vänta på att en dator ska starta, för detta tar tid och kan ge upphov till andra problem t.ex. att drivrutiner eller program ej startar korrekt. En annan orsak, förklarade IT-administration, var att datorns livstid minskas avsevärt om den stängs av och sätts på istället för att konstant vara igång. Det tredje problemet som IT-administrationen nämnde var att det inte finns någon klar energipolicy på Chalmers gällande energianvändning för datorer. Ett skäl till detta kan vara att ingen institution betalar för el, utan Chalmers betalar för allt centralt. Alltså finns det inte någon vinning för IT-administrationen att skära ned på energikostnaderna.

9.2.2 Energisparlägen för datorer

För Linux- och Windowsdatorer finns det tre vanliga metoder för att försätta datorn i energisparläge: stänga av datorn, strömsparläge eller viloläge. Strömsparläge låter RAM-minnet vara igång medan resten av datorn stängs av för att komma ihåg den information som fanns när användaren försatte datorn i strömsparläge. Detta betyder att strömsparläge inte behöver starta BIOS när datorn startar och på så vis minimeras riskerna av problem vid uppstart. Viloläge behöver däremot starta BIOS vid uppstart för att den sparar RAM-minnet till hårddisken. När BIOS väl har startat ser det likadant ut som när datorn stängdes av. Skillnaden mellan dessa två är att strömsparläge drar lite ström men startar snabbare och viloläge drar ungefär lika lite ström som om datorn vore avstängd men startar inte lika fort som strömsparläge (Microsoft, 2013). På Linux operativsystem finns samma energisparlägen men med andra namn och på Linux finns också möjlighet att skapa egna energisparlägen. I arbetet är både Windows och Linux diskuterat, dock används endast Windows namn på energisparlägen (Linux Kernel, 2011).

9.2.3 Datorns livslängd

Företag och organisationer menar till skillnad från Chalmers att det är möjligt att starta och stänga av datorer tusentals gånger. Energy Star som är världens största organisation för energikonsumtion för elektronik menar att en dator har 40 000 på och av cykler, vilket skulle betyda att en dator minst ska hålla över 100 år om den stängs av och sätts på en gång per dag. Det finns också stora IT-företag som t.ex. IBM, som uppmuntrar sina anställda att stänga av datorn varje dag för att skapa en bättre ekonomi och miljö (Cureton, 1995).

9.3 Undersökning av energiåtgången av datorer

För att undersöka hur mycket Chalmers skulle kunna spara på att använda energisparläge gjordes en studie på de datorer som fanns tillgängliga på skolan. För att få realistisk och användbar data gjordes olika energimätningar på två olika typer av datorer. Det var främst förbrukningseffekten som datorerna drar i olika energisparlägen som var intressant att studera. Då det inte fanns möjlighet att göra mätningar med HTCondor på datorer i datasalarna inriktades mätningar främst på datorer som fanns tillgängliga i vår testmiljö. Vad som måste tas i beaktande är att det finns många olika datorer på Chalmers och därför finns det ingen möjlighet att göra mätningar på alla. Studier på de två datorerna som finns i testmiljö ger ändå en realistisk bild då detta är typer av datorer som återfinns på campus.

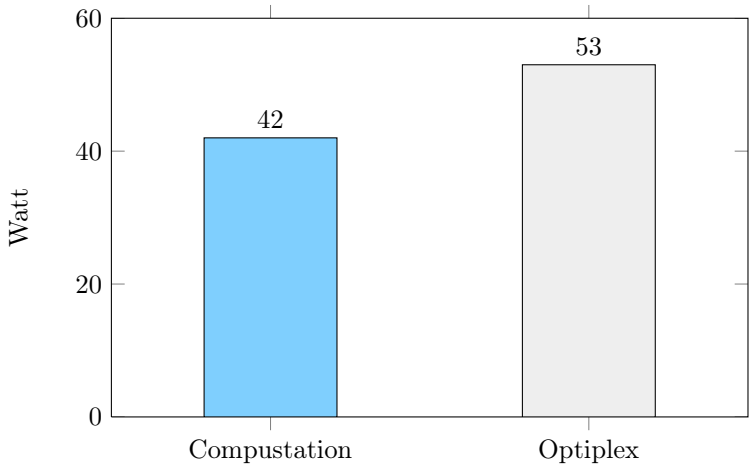
9.3.1 Metod för undersökningen

Viloläge, strömsparläge och avstängd är som tidigare nämnt de olika energilägena som finns för Windows och i Linux. Dessa tre olika energisparlägena testades och mättes flera gånger på de två olika datorer som vi hade tillgång till för att få ett trovärdigt resultat. Resultaten visas i form av tabeller och diagram.

Mätinstrumentet som användes var Rubicson's elmätare KGS02-01/67058. Vid mätningarna användes storheten watt med max en decimal för beräkning av effekt. Om effekten ändrade sig vid mätning togs medeltalet.

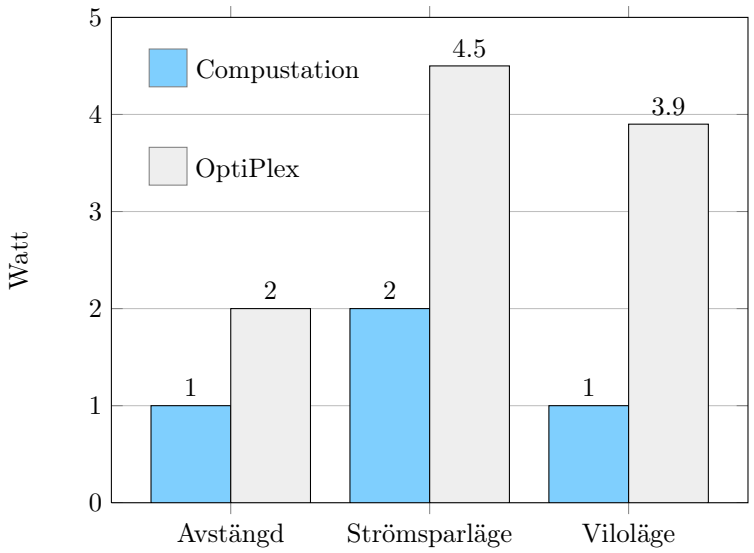
9.3.2 Skillnad mellan datorers energiåtgång

Jämförelsen av två olika typer av datorer gav ett intressant resultat. Det som kan observeras är att de äldre datorerna Dell OptiPlex fick ganska skilda resultatet mot Compustation datorerna. Vad som stack ut mest var den höga effekten av strömsparläge och viloläge för OptiPlex jämfört med Compustation. Som framgår ur undersökningen konsumerade OptiPlex 4.5 watt och något mindre 3.9 watt i strömsparläge jämfört med viloläge. Detta är en stor skillnad mot de nyare som hade effekten 2 jämfört med 1 watt. Alltså en skillnad mellan datorerna på minst det dubbla. Skillnaden var inte lika stor när datorerna jämfördes när de var igång, då konsumerade OptiPlex runt 10 watt mer, se figur 9.1. Detta visar att äldre datorer har en högre energikonsumtion speciellt i energisparlägena.



Figur 9.1: Jämförelse mellan datorerna när de är på.

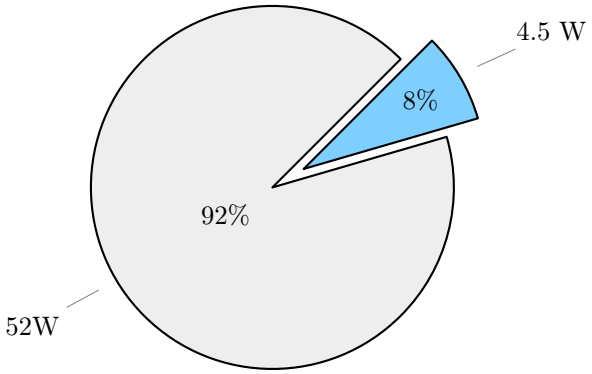
Resultatet, se figur 9.2, visar att alla energisparlägen har en låg effekt, mellan 1-4.5W. Siffrorna påvisar också att en dator förbrukar energi när den är avstängd och att skillnaden mellan strömsparläge, avstängd och viloläge inte är så stor, och ännu mindre på nyare datorer som Compustation. Det är viktigt att uppmärksamma att en äldre dator som Optiplex drar mer i viloläge än om den är vanligt avstängd jämfört med nyare datorer som Compustation som viloläge och avstängd drar lika lite. Detta betyder att mängden energi en dator drar i olika energisparlägen beror till stor del på hårdvaran.



Figur 9.2: Jämförelse mellan de två datorerna.

9.3.3 Besparingar som kan göras.

Det är inte förvånande att strömsparläge drar mest utav de olika energisparlägena. Det är dock en liten mängd energi jämfört med om datorn är på då detta konsumerar minst 10 gånger mer energi, se fig 9.3.



Figur 9.3: Dator på jämfört viloläge.

Det finns inga exakta siffror på hur många och hur mycket datorerna används men som nämnts tidigare används datorerna mest mellan 8 och 17 på vardagar. Alltså i stort sätt ingen användning av datorer på lov och helger. Detta ger en arbetstid på 9 timmar 5 dagar i veckan som totalt är 45 timmar per läsvecka. Då det finns 36 läsveckor summeras detta till 1 620 timmar av datoranvändning totalt för en dator på ett år. Alltså står en dator uppskattningsvis på 8 760 timmar per år där ungefär 7 140 timmar är inaktiv tid.

Effekt för en dator på ett år	**Timmar**	**Compustation (kWh)**	**OptiPlex (kWh)**
På under arbetstid	1620	84.24	69.66
Strömsparläge resten av tiden	7140	14.28	32.13
Alltid på	8760	455.52	376.68
Besparingar om strömspar aktiveras		357	274.89
Energispar konsumerar jämfört med på		22%	27%
Beräkningar på alla datorer			
Totalt med 2000 datorer (kWh)		714000	549780
Pris 0.80 (kr) per kWh		571200	43924

Tabell 9.1: Effektförbrukning av datorer per år

Tabellen, se tabell 9.1, visar hur mycket energi som kan sparas på att försätta datorn i strömsparläge när de inte används jämfört med om de hade varit på dygnet runt. Som går att utläsa ur tabellen kan alltså den totala effekten minska med ca 400 KWh per år, vilket är en minskning med tre fjärdedelar. Detta betyder att besparingar kan göras på 400-600 tusen kronor per år om det är ca 2 000 datorer och elpriset ligger på 0,80kr/KWh. I miljösynpunkt blir detta totalt mellan 13 750 till 17 850kg/CO2 utsläpp (Svensk energi, 2013).

10 Resultat och Diskussion

I det här kapitlet presenteras resultaten vi kommit fram till och vår diskussion kring dem. Först presenteras resultaten och diskussionen av distribueringsdelen av projektet och sedan energidelen. Till sist diskuterar vi vad som skulle kunna göras i framtiden för att ytterligare förbättra resultatet.

10.1 Distribuering med HTCondor

Vårt mål har varit att antingen skapa ett system för distribuering från grunden eller ta fram ett som redan var tillgängligt på marknaden och använda oss av det. Prioritering föll på att Chalmers skulle kunna få ett system som var så bra som möjligt att använda på hela campus. Efter studier på den typen av system och deras konstruktion, kom vi till insikt med att vi själva inte har möjligheten att konstruera en bra produkt inom den tidsram arbetet har. Beslutet att använda ett existerande system togs och det är någonting vi inte har ångrat.

Vi valde att använda oss av systemet HTCondor. Möjligheten att använda liknande system fanns, dock var det svagheter hos de systemen vilket gjorde att vi ansåg de mindre lämpade för Chalmers. HTCondor var det bästa distribuerande systemet utav de som vi jämförde som användes på andra universitet.

10.1.1 Utskickning av arbete

Processen för att skicka ut arbete är inte så enkel som den skulle kunna vara. För en användare som inte har erfarenhet av liknande system kan processen se komplicerad ut. Användarvänligheten var någonting vi prioriterade högt och därför skapades ett GUI för att förenkla inlärningsprocessen. Dock kräver komplicerade arbeten fortfarande användning av terminalen.

Vilken dator som är bäst anpassad för ett arbete beror på många olika faktorer. Avstängda datorer prioriteras lägre än datorer som redan är på, en prioritering som känns naturlig. Användaren som skickar ut arbete får själv ställa in de krav och preferenser som arbetet har. Det är någonting som inte systemet själv klarar av då det på förhand inte vet vad arbetet är och därmed inte vilka datorer som är bäst lämpade.

Vi har testat att köra arbeten skrivna i C, Java och Matlab vilket HTCondor har klarat utan problem. Det går även att skicka med den miljö som ett arbetet behöver för att kunna köras och på det sättet går alla typer av arbeten att köras, oberoende av vad som är installerat på datorn. Dock så spelar det fortfarande roll hur filerna som ska köras är kompilerade om inte arbetet själv kompilerar filerna innan de körs. Att skicka med miljön kan

vara besvärligt och i somliga fall opraktiskt om filerna som behövs är flera hundra megabyte, vilket både tar tid att föra över och utrymme på hårddisken.

10.1.2 Användarvänligheten

Det viktigaste med distribueringen var att de fysiska användarna inte skulle störas. Med HTCondor och den konfiguration som beskrivs i kapitel 7.2 Prioritering av de fysiska användarna kan arbeten som utförs pausas eller flyttas när någon börjar använda datorn. Användaren störs inte av HTCondor efter det att arbetet har avslutats, pausats eller flyttats, vilket sker snabbt efter upptäckt av användare. HTCondor kräver att det hela tiden körs ett par processer i bakgrunden på datorn men de tar inte upp mycket minne och använder i stort sett ingen datorkraft.

Att användaren som skickar ut arbete inte ska behöva ha någon kunskap om distribuering och inte heller behöva ändra i några filer var något vi gärna ville uppnå. Användaren behöver bara veta hur en submit-fil ska se ut och hur den skall skrivas. För enklare arbeten krävs inte ens det, eftersom GUI:t som gjordes klarar av att skapa en submit-fil om användaren fyller i de parametrar som behövs. Om någonting går fel vid något steg i processen av utskickningen av arbetet och det måste felsökas, behövs en del kunskap för att åtgärda det. Skulle detta hända är antingen Central Managern eller datorn som används felkonfigurerad och det kräver en administratörs kunskap att rätta till.

Användaren behöver inte ändra något i filerna som ska köras, det kan däremot krävas en del ändringar för att få all funktionalitet som exempelvis Matlab-kod som behöver ändras en del i för att kunna utnyttja checkpoints.

Hur enkelt det blir att administrera beror på hur systemet sätts upp. Servern är enkel att administrera, det är bara en dator, men resten av poolen är svårare. Den enklaste lösningen för att sätta upp poolen är att ha lokala konfigureringsfiler på varje enskild dator, något som kan bli väldigt svårt att administrera. Vad vi borde ha testat är att samla alla konfigureringsfiler på ett och samma ställe med ett delat filsystem. Om alla styrfiler hade lagts på exempelvis servern skulle de förmodligen bli enklare att administrera.

10.1.3 Energisparande med HTCondor

När vi upptäckte att HTCondor också stödjer uppväckning och avstängning av datorer passade det utmärkt att använde oss av de funktionerna, istället för att använda ett program parallellt för att fylla samma funktion. Eftersom HTCondor väcker datorer när de behövs för beräkningar skulle ett sådant program ha behovet av att vara starkt integrerat med HTCondor. Det skulle varit både krångligt och då det redan finns, onödigt. De inställningar som vi ändrade för att möjliggöra HTCondor att väcka datorer, var att i varje dator aktivera Wake-on-LAN och konfigurera HTCondor för den datorn.

Vi upptäckte att HTCondors uppväckningssystem inte är helt perfekt. Datorer som inte stängs av med HTCondor går inte att väcka med systemet när de behövdes för arbete. Orsaken är att datorn inte hinner skicka information till Central Managern om att den går in i energisparläge. HTCondor tror då att den helt enkelt tappat kontakten med datorn och försöker inte väcka den eller tilldela den arbete. Utöver det har HTCondor fungerat utan problem.

Det är inte bara HTCondor som ska kunna väcka datorer i energisparläge, en person som ska använda en dator måste själv kunna starta den. Eftersom datorerna inte alltid står bredvid resten av utrustningen bör datorerna ställas in så att de går att väcka med mus och tangentbord.

10.1.4 Konstruktion av GUI till HTCondor

Den implementering vi gjorde av GUI var lyckad, den var både enkel att använda och förstå. Dock gjordes inga verkliga tester på användarvänligheten. Vi valde att inte implementera alla funktioner som en vanlig användare kan göra via terminalen då många av dessa nästan aldrig används och vi bestämde oss för att lägga ner mer tid på andra delar som kändes mer relevanta. Möjligheten att skicka ett arbete till systemet känns som den bästa funktionen, för då slipper användaren av systemet skriva submit-filen som annars behövs. Valet att lista alla datorer var vi lite tveksamma till först, men vi valde att implementera det då vi kan tänka oss att vissa användare vill veta hur mycket kraft det finns att tillgå för en viss typ av arbete som för t.ex. Matlab arbeten.

Funktioner som inte implementerades

GUI:t hade kunnat utvecklats mycket mer och det finns ett antal funktioner som vi valde att inte implementera. Detta på grund av tidbrist och att vissa av dessa funktioner inte kändes nödvändiga. Något som hade varit intressant att implementera är statistik, möjlighet att samla information från alla användare och alla de arbeten som körs i HTCondor. Insamling av statistik skulle ha inbringat en större möjlighet att analysera användningen av distribuering för att t.ex. få kunskap om vilken typ av arbete som är vanligast och när datorerna belastas som mest mm. En intressant tanke var att använda ett webbgränssnitt, alltså att användare kan ha behörighet till en viss sida inom nätverket där det går att ladda upp och hantera arbeten. Det för att slippa ha klienter på varje dator där användare kan tänkas använda HTCondor. För att implementera eller använda ett webbgränssnitt krävs som tidigare nämnts kunskap om språken HTML och XML. Därför valde vi att inte göra denna typ av implementering.

API

Valet att använda ett API var enkelt då det förenklade arbetet avsevärt med GUI:t. Om vi inte gjort det valet skulle vi behövt en mycket djupare förståelse om HTCondors källkod. Det tog ett bra tag innan ett API hittades och det fanns knappt någon information alls om det. Det gjorde oss först kritiska till att använda det, men då vi testade API:t och fick det att fungera bra valde vi att fortsätta med det. Vi behövde en del metoder som inte fanns i API:t som vi själva enkelt kunde skriva och lägga till. Det var något som vi såg som positivt, en vidareutveckling av API:t. Det kan finnas en god nytta av att ha ett bra API till HTCondor för att i framtiden utveckla mer avancerade GUI:n.

10.1.5 Säkerhet och skalbarhet

Det finns en hel del som inte testats praktiskt, bland dessa är de viktigaste två områdena säkerheten och skalbarheten. Skalbarheten är omöjlig att testa med det antal datorer som varit tillgängliga. Däremot skulle flocking med fördel kunnat testats.

Vi använde oss av en HTCondor server men det kan krävas flera vid en implementation på Chalmers. Vid stora nätverk kan det vara intressant att dela upp större subnät i olika HTCondor-pooler vilket vi tror skulle underlätta för administratörerna.

HTCondor har stöd för kända standarder inom säkerhet. Beslutet att inte utföra säkerhetstester baserade sig på projektgruppens dåliga kunskaper inom säkerhetsområdet. Om något test skulle ha genomförts på säkerhet skulle det med stor sannolikhet varit av för dålig kvalité för att ge nyttig information.

10.2 Energisparlägen och miljöaspekten

En fråga som tagits upp är om Chalmers verkligen ska använda energisparlägen i HTCondor när de redan kan ha använt datorernas egna inställningar för energibesparing. Det var skälet till att vi gjorde en energistudie som underlag för att reflektera över hur mycket energi som kan sparas. Vi undersökte också varför Chalmers inte stänger av datorer och om de gör rätt i det valet. Om det ändå är så att Chalmers eller de som använder sig av HTCondor inte vill använda energisparläge så finns möjligheten att bara ta bort denna funktion och enbart använda distribuering.

10.2.1 Problem med avstängning

Våra resultat visar att det är möjligt att stänga av datorer som inte används, dock menar Chalmers IT-administration att det finns ett antal problem som inte gör det möjligt. Det största problemet som de nämnde var att datorn alltid måste vara tillgänglig för alla på Chalmers. Vi tycker att en dator i strömsparläge är tillgänglig, då den slipper att starta BIOS och alla program vid uppstart, vilket dels innebär att datorn startar snabbare och att riskerna med problem minskas vid uppstart.

Ett annat problem som nämndes var slitage som en dator får när den startas. Dock pekade allt material som vi hittade i förstudien på att det slitaget är försumbart, då en dator ska klara minst 40 000 uppstarter. Därför tycker vi att Chalmers argument om att livslängen förkortas är vag, det känns orimligt att Chalmers skulle äga såpass gamla datorer att detta problem fortfarande skulle vara en risk.

Det sista problemet som IT-administrationen nämnde, var att administratörer måste ha möjlighet att komma åt datorerna när de t.ex. ska installera ny programvara. Det var inget som undersöktes något närmare, då de som arbetar med denna typ av administration har egna verktyg som vi inte hade tid att lära oss. Vad vi däremot är ganska säkra på är att de verktyg som de använder kan med stor sannolikhet stödja uppväckning av datorer. Skulle det inte vara möjligt så går det alltid att väcka datorer med HTCondor.

Något annat som förvånade oss från förstudien var den avsaknad av miljösyn som verkar råda kring miljöfrågor om energi på institutionerna. Det verkar som att Chalmers som har ambition att vara i framkant när det gäller miljöfrågor inte har några policys för datorers energiförbrukning. Om institutionen som vi intervjuade betalade elräkningen ur egen ficka hade de nog med stor sannolikhet varit mer måna om att hålla energikostnaderna nere.

10.2.2 Energistudien

Resultatet från energistudien visar att strömsparläge är det bästa alternativet om datorer ska stängas av på Chalmers. Skälet till det valet beror på att datorn startar väldigt snabbt igen och drar bara 1-2 watt mer än en dator som är avstängd eller i viloläge. Det var lite förvånande då vi trodde att strömsparläge skulle dra bra mycket mer än vad det gör. Vi blev också överraskande över att en dator som är helt avstängd ändå drar 1-2 watt. Det gjorde valet om energisparläge ännu enklare. Även med risken för felmarginaler i våra mätningar kan vi säga att skillnaden i energiförbrukning mellan en dator som är igång och en dator i strömsparläge är tillräckligt stor för att det ska vara värt att införa energisparfunktioner på Chalmers datorer.

Då vi inte haft tillgång till skolans vanliga datorer så kan vi inte fullt ut förlita oss på den data som samlades. Mätinstrumentet som fanns till förfogade kan också kritiseras då dessa instrument kan ge en liten felmarginal. Så vid t.ex. låg effekt som 2 watt kanske det egentligen är 1 watt. I det stora hela kan denna felmarginal och den kritiska datan nästan försummas då vi fick fram resultat som pekade på en rejäl minskning av energi, vilken dator det än var som testades.

10.3 Fortsatt utveckling

Något som vi ofta tänkte på under projektets gång var om Chalmers inte kan stänga av datorer och det inte finns tillräckligt med arbeten för att hålla alla datorer aktiva, så borde datorer utnyttjas på något annat sätt. Vi såg i förstudien att universitet har använt BOINC tillsammans med HTCondor för att utnyttja alla datorer även när det inte finns tillräckligt med HTCondor arbeten. Detta är något som Chalmers skulle kunna implementera i HTCondor för att köra arbeten från olika BOINC-projekt på skolans datorer och på så sätt hjälpa forskare världen över. Dock skulle det innebära att Chalmers elräkning kommer att öka, det kostar en del att ha tusentals datorer arbetandes dygnet runt.

Som nämnts under GUI avsnittet i diskussion finns det mycket att arbeta vidare på när det gäller användarvänligheten. Implementation av en statistik del, webgränssnitt och mer funktioner är något som skulle vara relevant i framtiden.

Om HTCondor skulle installeras på campus alla datorer kan det vara aktuellt att låta de existerande kluster som finns styras av HTCondor. Då skulle det finnas ett system för all distribuering av arbete och inte tre olika som det skulle finnas annars, HTCondor medräknat.

11 Slutsatser

Vi har analyserat hur Chalmers bäst kan använda outnyttjad datorkraft genom distribuering av arbete samt bättre energihantering. HTCondor valdes som mest lämplig kandidat för dessa uppgifter. Systemet skulle vara möjligt att implementera på hela Chalmers och drivas utan att märkas av de som aktivt arbetar med datorerna. Inte heller ska systemet alstra dyra omkostnader för Chalmers utan snarare tvärtom.

Vi lyckades bra med att implementera HTCondor i vår labbmiljö och det fungerade felfritt att skicka olika typer av arbeten till systemet där HTCondor-poolen bestod av både Windows och Linux datorer. HTCondor klarar både att starta de datorer som behövs och försätta de som inte används i strömsparläge. Ingen typ av databas var nödvändig då HTCondor alltid har kontroll på datorerna som finns i poolen, även när de är avstängda. Vid de tillfällen en dator börjar användas samtidigt som ett av HTCondors arbeten körs på den, pausas eller flyttas arbetet till en andra dator i poolen. Vilket innebär att en inloggad användare på datorn aldrig märker av systemet. För att förenkla för nya användare har ett enkelt GUI konstruerats, med vars hjälp de kan genomföra enkla uppgifter och lära känna systemet.

Vi rekommenderar starkt att Chalmers börjar använda ett system för att bättre ta till vara på oanvänd datorkraft då det är högt tryck på de kluster som redan finns på campus. Tillvägagångssättet med att i första hand distribuera ut arbete med HTCondor och i andra hand sätta datorer i energisparläge fungerade utmärkt. Det energisparläge som genom studie visades vara bäst var strömsparläge, stora besparningar kan göras på Chalmers genom att börja använda det.

Om Chalmers av någon anledning inte väljer att använda ett kombinerat system utan bara en av teknikerna, skulle det inte innebära några problem då systemen fungerar väl var för sig. Vi förespråkar emellertid ett kombinerat system där de båda teknikernas styrkor kan komplementera varandra.

Källförteckning

Adaptive Computing (2011) Ten Reasons to Switch from Maui Cluster Scheduler to Moab HPC Suite. Adaptive Computing.
http://www.adaptivecomputing.com/ (2013-04-22)

Adaptive Computing (2013) Open-Source. Adaptive Computing.
http://www.adaptivecomputing.com/products/open-source/ (2013-05-07)

AMD (1995) Magic Packet Technology White Paper. AMD.
http://support.amd.com/us/Embedded_TechDocs/20213.pdf (2013-03-25)

Andrew vonNagy (2010) Wake on Wireless LAN. Revolution Wi-fi.
http://revolutionwifi.blogspot.se/2010/11/wake-on-wireless-lan.html (2013-02-15)

Apache Hadoop (2012) HDFS Architecture. The Apache Software Foundation.
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html (2013-03-10)

Asus (2013) FAQ, How do you enable Wake On LAN (WOL) in the BIOS?. Asus Support.
http://support.asus.com/Search/KDetail.aspx(2013-03-25)

Bradley, D. et al. (2011) An update on the scalability limits of the Condor batch system. Journal of Physics: Conference Series, vol. 331, nr 6. s. 062002.

Calleja, M. et al. (2004) CamGrid: Experiences in constructing a university-wide, Condor-based grid at the University of Cambridge. I Proceedings of the UK e-Science All Hands Meeting 2004; 31 augusti - 3 september 2004, Nottingham. s. 173 -178.

C3SE (2013) Wiki. Charmers Centre for Computational Science and Engineering
http://www.c3se.chalmers.se (2013-02-20)

Chalmers (2012) Facts and figures. Chalmers tekniksa högskola
http://www.chalmers.se/en/about-chalmers/Pages/facts-and-figures.aspx (2013-05-08)

Chalmers StuDAT (2013) Datorsalsinformation. Chamlers StuDAT
http://www.studat.chalmers.se/mofit/studat/labb.xml (2013-03-22)

Cureton, M (1995) Home Energy Brief, Computers & Peripherals. Rocky Mountain

institute.
http://infohouse.p2ric.org/ref/32/31146.pdf (2013-3-23)

Cycle Computing (2013). Cycle Server. Cycle Computing.
http://www.cyclecomputing.com/cycleserver/overview (2013-04-12)

DD-WRT (2012) Wake on LAN (Tutorial). DD-WRT Wiki.
http://www.dd-wrt.com/wiki/index.php/Wake-on-LAN_(tutorial) (2013-02-20)

Gubb, D. et al. (2012) Implementing a Condor pool using a Green-IT policy. University of Huddersfield.
http://eprints.hud.ac.uk/15839/2/DR2012-18.pdf (2013-04-28)

HTCondor

(a) HTCondor (2013) HTCondor - High Throughput Computing.
http://research.cs.wisc.edu/htcondor/ (2013-03-10)

(b) HTCondor (2013) What is HTCondor?. HTCondor - High Throughput Computing.
http://research.cs.wisc.edu/htcondor/ (2013-03-10)

(c) HTCondor (2013) Classified Advertisments. HTCondor - High Throughput Computing.
http://research.cs.wisc.edu/htcondor/classad/classad.html (2013-04-26)

HTCondor Manual

(a) HTCondor Manual (2013) HTCondor Version 7.9.5 Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/ref.html (2013-04-18)

(b) HTCondor Manual (2013) Installation. Administrator's Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/3_2Installation.html (2013-04-18)

(c) HTCondor Manual (2013) Introduction. Administrator's Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/3_1Introduction.html (2013-04-18)

(d) HTCondor Manual (2013) HTCondor's Checkpoint Mechanism. Miscellaneous Concepts.
http://research.cs.wisc.edu/htcondor/manual/v7.9/4_2HTCondor_s_Checkpoint.html (2013-04-28)

(e) HTCondor Manual (2013) Road-Map for Running Jobs. Users' Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/2_4Road_map_Running.html (2013-04-28)

(f) HTCondor Manual (2013) DAGMan Applications. Users' Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/2_10DAGMan_Applications.html (2013-05-01)

(g) HTCondor Manual (2013) Connecting HTCondor Pools with Flocking. Grid Computing.
http://research.cs.wisc.edu/htcondor/manual/v7.9/5_2Connecting_HTCondor.html (2013-05-02)

(h) HTCondor Manual (2013) Setting Up for Special Environments. Administrator's Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/3_12Setting_Up.html (2013-05-03)

(i) HTCondor Manual (2013) Virtual Machines. Administrator's Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.8/3_14Virtual_Machines.html (2013-05-03)

(j) HTCondor Manual (2013) Power Managerment. Administrator's Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.9/3_15Power_Management.html (2013-05-04)

(k) HTCondor Manual (2013) Security. Administrator's Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.8/3_6Security.html (2013-05-03)

(l) HTCondor Manual (2013) Grid Universe. Grid Computing.
http://research.cs.wisc.edu/htcondor/manual/v7.9/5_3Grid_Universe.html (2013-05-05)

(m) HTCondor Manual (2013) Parallel Applications (Including MPI Applications). Users' Manual.
http://research.cs.wisc.edu/htcondor/manual/v7.8/2_9Parallel_Applications.html (2013-05-05)

(n) HTCondor Manual (2013). Application Program Interfaces. Web Service.
http://research.cs.wisc.edu/htcondor/manual/v7.8/4_5Application_Program.html (2013-04-01)

Linux Kernel (2011). System Power Management States. The Linux Kernel Archives.
https://www.kernel.org/doc/Documentation/power/states.txt (2013-05-13)

Liverpool (2012) Efficient use of the Condor Pool through Application Checkpointing. University of Liverpool.
http://www.liv.ac.uk/csd/escience/condor/checkpoint.htm (2013-04-28)

Microsoft (2011) The Command-Line Options for the Microsoft Windows Installer Tool Msiexec.exe. Microsoft Support.
http://support.microsoft.com/kb/314881/en-us (2013-03-24)

Microsoft (2013) Sleep and hibernation: frequently asked questions. Windows Microsoft.
http://windows.microsoft.com/en-sg/windows7/sleep-and-hibernation-frequently-asked-questions (2013-03-20)

Mosley, S., Bear, H. och Totten, N. (2010) UEA Desktop Computer, Power Mointoring and Management. University of East Anglia.
http://www.uea.ac.uk/polopoly_fs/1.161952!desktop-power-mgt-report.pdf (2013-05-02)

Nakada, H. (2009). Condor Java API: Java Interface for Condor. Information Technology Research Institute.
http://staff.aist.go.jp/hide-nakada/condor_java_api/ (2013-04-17)

Nakada, H. (2012). Curriculum Vitae. Information Technology Research Institute.
http://staff.aist.go.jp/hide-nakada/cv.html (2013-04-23)

Nvidia (2011) TESLA M-CLASS GPU COMPUTING MODULES ACCELERATING SCIENCE. Nvidia.
http://www.nvidia.com/docs/IO/105880/DS-Tesla-M-Class-Aug11.pdf (2013-03-23)

Objectweb (2007) What is Middleware. ObjectWeb.
http://middleware.objectweb.org/ (2013-05-17)

Red Hat (2011) What is the relationship between Fedora and Red Hat Enterprise Linux? Resource Library.
http://www.redhat.com/resourcelibrary/articles/relationship-between-fedora-and-rhel (2013-04-10)

Sepulveda, D (2009) Deploying and Maintaining a Campus Grid at Clemson University. Clemson University.
http://etd.lib.clemson.edu/documents/1253044158/Sepulveda_clemson_0050M_10229.pdf (2013-04-30)

Smith, I.C (2010) Towards a greener Condor pool: adapting Condor for use with energy-efficinent PCs. University of Liverpool.
http://www.liv.ac.uk/csd/escience/condor/power_save.pdf (2013-04-18)

Svensk energi (2013). Klimatpåverkan och växthusgaser. Svensk energi.
http://www.svenskenergi.se/Elfakta/Miljo-och-klimat/Klimatpaverkan/ (2013-04-26)

Thain, D., Tannenbaum, T. och Livny, M. (2005) Distributed computing in practice: the Condor experience. Concurrency - Practice and Experience, vol 17, nr 2-4, ss. 323-356.

Evans-Toyne, S., Bradley M. och Robinson L. (2011) New DIY supercomputer saves £1,000s. University of Westminster.
http://www.westminster.ac.uk/news-and-events/news/2011/university-of-westminster-launches-new-diy-supercomputer-saving-hundreds-of-thousands-of-pounds (2011-03-29)

Wilson, P., Emmerich, W. och Brodholt, J. (2004) Leveraging HTC for UK eScience with Very Large Condor pools: Demand for transforming untapped power into results. University College London.
http://eprints.ucl.ac.uk/701/1/9.8_131.pdf (2013-05-02)

Wireshark (2013) About Wireshark. Wireshark.
http://www.wireshark.org/about.html (2013-02-20)

A Exempel på submit-filer

Submit för C

```
Universe    = standard
Executable  = program
Log         = log
Error       = error.$(Process)
Output      = output.$(Process)
Queue       1
```

Submit för Matlab

```
Executable      = matlabRun.$$(OpSys).bat
Requirements    = ((Target.OpSys=="LINUX" && Target.Arch=="X86_64") ||
                (Target.OpSys=="WINDOWS" && Target.Arch=="INTEL"))
Transfer_Input_Files    = testPi.m
Should_Transfer_Files   = YES
When_To_Transfer_Output = ON_EXIT

Log         = log
Error       = error.$(Process)
Output      = output.$(Process)
Arguments   = testPi 100
Queue       1
```

Submit för Java

```
Universe    = Java
Executable  = Job.class

Log     = log
Error       = error.$(Process)
Output      = output.$(Process)
Arguments   = Job 5 7
Queue       1
```

B Matlab-körfil för skript

Konfigureringsfil för Linux (*matlabRun.LINUX.bat*)

```
#!/bin/sh
script=$1
arguments=$2
executable=/usr/local/MATLAB/R2012a/bin/matlab
${executable} -nodisplay -nosplash -logfile result
       -r "${script}(${arguments});quit"
```

Konfigureringsfil för Windows (*matlabRun.WINDOWS.bat*)

```
@echo off
set SCRIPT=%1
set ARGUMENT=%2

set TMP=%CD%
set TEMP=%CD%
set USERPROFILE=%CD%
set USERPATH=%CD%

mkdir "My Documents"
mkdir "My Documents\MATLAB"
set MATLAB_PREFDIR=%CD%\My Documents\MATLAB

start /wait matlab -wait -nojvm -nodesktop -nosplash -noFigureWindows
       -r "%SCRIPT%(%ARGUMENT%);quit;" -logfile result
```

C Köra Matlabkod med medskickad Octave på Windows

```
Vad som behövs
En fil som kan öppna zip-filer. Här kallad 7za.exe.
Zippad Octave miljö för att köra Matlab filen.

Submit-fil
Universe  = standard
Executable = run.bat
Arguments = run.bat $$([ 60000 * ( 1 + $(PROCESS))])

Transfer_input_files = 7za.exe, Octave.7z, run.bat
Should_trasfer_files = yes

Output = $(CLUSTER)_$(PROCESS).out
Log = $(CLUSTER)_$(PROCESS).log

Körfil
@echo off
7za.exe x -aoa -y Octave.7z > NUL
Octave\bin\octave.exe -H -q %*
rmdir /S /Q Octave
del /F /Q Octave*
```

D Konfigurering av arbetsregler

```
# Start
# Startar ett nytt jobb om: Datorn inte använts på 10 minuter och om
# CPU:n inte används av #någonting annat än HTCondor. Nya jobb får
# startas om ett jobb redan körs, det gör att #HTCondor kan byta till
# högre prioriterade jobb.
StartIdleTime   = 10 * $(MINUTE)
CPUIdle = LoadAvg - CondorLoadAvg <= 0.3
START   = (KeyboardIdle > $(StartIdleTime) && ( $(CPUIdle) ||
          (State != "Unclaimed" && State != "Owner"")))

# Want_suspend
# Om ett jobb ska pausas eller flyttas. Här pausas det om SUSPEND är
# sann samt om jobbet är #litet ( <20MB).
SmallJob        = TARGET.ImageSize < (20 * 1024)
WANT_SUSPEND    = $(SmallJob) && $(SUSPEND)

# Suspend
# När ett jobb ska pausas om någon använder tangentbordet.
KeyboardBusy = KeyboardIdle < $(MINUTE)
SUSPEND      = $(KeybordBusy)

# Continue
# Jobbet återupptas om ingen använt daotrn på 5 minuter samt att CPU:n
# inte används.
ContinueIdleTime    = 5 * $(MINUTE)
CONTINUE        = $(CPUIdle) && (KeyboardIdle > $(ContinueIdleTime))

# Want_vacate
# Jobb flyttas istället för att tas bort om jobbet körts längre än 10
# minuter.
ActivationTimer =(time() - EnteredCurrentState)
WANT_VACTE = $(ActivationTimer) > 10 * $(MINUTE)

# Preempt
# Ett jobb flyttas om det varit pausat i mer än 15 minuter eller om
# kraven för SUSPEND uppnås #fast WANT_SUSPEND är falsk, med andra ord
# om någon använder datorn när ett jobb större #än 20MB körs.
MaxSuspendTime  = 15 * $(MINUTE)
PREEMPT         = ( ((Activity == "Suspend") && ($(ActivityTimer) >
           $(MaxSuspendTime))) || (SUSPEND && (WANT_SUSPEND == False)) )

# Ett jobb tas bort om det tar längre än 10 minuter på sig att flyttas.
```

```
MachineMaxVacateTime = 10 * $(MINUTE)

# Kill
# Ett jobb ska aldrig avbryta sin flytt innan det tagit längre tid än
# MachineMaxVacateTime att flyttas.
KILL                  = False
```

E Intervju med Data och IT institutionen

EDIT-huset på Chalmers Datum: 18/4 - 13

De intervjuade var anställda på Data och IT institutionen på Chalmers
Arne Linde, IT-chef.
Rune Ljungbjörn, Systemadministratör för Linux.
Peter Helander, Administratör och IT-support.

Intervjun genomfördes av kandidatgrupp 31
Daniel Bergqvist, Rurik Högfeldt, Marcus Kalander och Oliver Andersson.

Används datorerna till vettiga saker när de inte används ex. natten osv?

- Nej

Behövs det beräkningskraft?

- Ja finns alltid behov på en organisation som chalmers
- Det klustret som finns används men det krävs att personen har ett projekt och bokar upp en tid, alltså krångligt.
- Arne får ett par förfrågningar per år om användning av datorsalarnas datorer

Varför stängs datorerna inte av?

- Administration måste enkelt kunna uppdatera datorerna på Chalmers och då förenklar det om datorerna är på.
- Datorer går sönder när man stänger av dem. Power cycling är det som sliter på datorer.
- Rent praktiskt är det jobbigt.
- Chalmers betalar alla elräkningar så att institutionen som finns behöver inte lägga någon budget på detta, därav blir det bara en kostnad om vi måste börja administrera datorer. Skulle vi själva betalat hade vi nog varit mer måna om att dra ner på elförbrukningen. Det verkar som att det inte finns någon klar miljöpolicy på Chalmers gällande detta

Hur många datorer finns det på Chalmers?

- IT: 100 labbdatorer
- Chalmers: 600 linux datorer fastsittande (inte laptops)
- Snitt: 2 datorer per anställd (merparten bärbara). gissningsvis 200-250 datorer som inte är laptops.
- Arne gissar ett par tusen datorer totalt sett, 3000

Vad för system används för att Administrera datorer?

- Linux: Hobbit, globalt filsystem. 2 tjänster.

Hur ser nätverket ut?

- Logiskt subnät. Alla datorer kommer åt varandra med ip-adresser.
- Måste vidarebefodra broadcasts genom routrar

Finns det fysiska datorer vid skärmarna?

- Alltid en dator per skärm och tangentbord, datorerna är inte alltid åtkomliga av studenter.
- Finns superdatorer, används främst som servrar.