Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik
2004-12-22

GRID COMPUTING SOM PLATTFORM FÖR DATA MINING

- en kvalitativ undersökning över egenskaperna hos grid computing och deras möjlighet att tillgodose kraven hos data mining

Abstrakt

Vi har utfört en studie i syfte att undersöka i vilken mån computational grids kan tillgodose kraven *processorkraft*, *processorkraft på begäran*, *stora mängder minne*, *robusthet*, *säkerhet och integritet* samt *enkel tillgång till data* som data mining ställer på ett system. Kravlistan isolerades med hjälp av litteraturstudier varpå intervjuer med företrädare för Swegrid, SAS och IBM, i kombination med litteraturstudier, användes för att undersöka hur väl computational grids kan uppfylla dessa krav. Slutsatsen blev att grid computing mycket väl kan tillgodose de flesta kraven, men att tekniken ännu inte nått det stadie av utveckling som kan anses nödvändigt för att computatonal grids fullt ut skall kunna tjäna som plattform för data mining. Bland annat bör kösystem effektiviseras och felhantering förbättras.

Nyckelord: grid computing, data mining, computational grids, scavenging grids

Författare: Johan Björk & Fredrik Pålsson
Handledare: Faramarz Agahi
Magisteruppsats 20 poäng

FÖRORD

Ett stort tack till till de på Swegrid, SAS Institute och IBM som hjälpt oss med information till denna uppsats och ställt upp på våra intervjuer så att uppsatsen kunde genomföras.

Vi tackar även vår handledare och examinator för deras input.

INNEHÅLLSFÖRTECKNING

1 INLEDNING

Detta kapitel inleds med en redogörelse för bakgrunden till det som avhandlas i uppsatsen. Vi redogör därefter för problemområde, frågeställning, vad som gjorts tidigare på området samt språkpolicy. I kapitlet ingår även en lista med definitioner av ord och koncept som förekommer i arbetet. Kapitlet avslutas med en redogörelse för rapportens disposition.

1.1 Bakgrund

Både grid computing (hädanefter benämnt som GC) och data mining (hädanefter benämnt som DM) har fått en hel del uppmärksamhet inom forskning de senaste åren. DM används redan i stor utsträckning inom näringslivet bl.a. för att upptäcka kontokortsbedrägerier och analys av kunders inköpsmönster samt processtyrning i t.ex. fabriker. Det som man märker av är att folk anser att de inte riktigt har tid till så mycket DM som de skulle önska [5]. Man kan ju då fråga sig hur ett företags strategi skulle se ut om de kunde utföra DM varje gång de önskade istället för att behöva vänta på ett bra tillfälle.
GC å sin sida är ett relativt nytt fenomen. Det talas väldigt mycket om GC i tidningar och på hemsidor, man jämför det med att få el ur vägguttaget men att det här handlar om datorkraft istället för el och man talar om att det skall finnas företag som kommer att tillhandahålla processorkraft till uthyrning [45]. Frågan är då om näringslivet kommer att ha någon nytta av GC och i så fall hur mycket eller om GC kommer att fastna i forskningsvärlden.
GC är ännu så länge dock fortfarande i utvecklingsstadiet, vilket visar sig i att mycket forskning inom området fortfarande ägnas åt att utveckla de olika beståndsdelar som anses behövas för att ett grid skall fungera som det är tänkt [14], [10]. Det förekommer dock även olika forskningsområden där försök gjorts med att utföra beräkningar och simulationer på gridsystem [3], [20]. På senare tid har även näringslivet börjat intressera sig allt mer för olika distribuerade lösningar – t.ex. kluster och grid – för att få tillgång till mer datorkraft än det som finns tillgängligt i datorer med enbart en processor. Allt fler applikationer anpassas för grid, t.ex. har Adobes senaste version av videoredigeringsprogrammet After Effects Professional en insticksmodul för redigering med hjälp av GC. Något som tycks hindra en mer utbredd användning av grid just nu är att man har svårt att komma överens om en standard. Ett annat problem är att GC fortfarande inte har en helt klar definition, något som kan tänkas förvirra ute på marknaden.
I West Virginia i USA invigs den 19:e november Global Grid Exchange. Delstaten genomför projektet tillsammans med West Virginia Technology Consortium, Verizon och Hewlett-Packard. Projektet går ut på att länka datorer från akademiska institutioner, myndighetskontor och personliga datorer som har datorkraft i överflöd. Gridet kommer att öppnas för alla genom en portal för ett ännu så länge obestämt pris. Det här är ett steg i att kunna bidra till datorkraft efter behov och att utnyttja de resurser man redan har tillgång till fullt ut.

Ett annat projekt är Eurogrid. Eurogrid är ett samarbete i Europa mellan ett antal nationella HPC-center(High Performance Computing), stora användare av HPC och teknologiföretag som tillhandahåller HPC. Samarbetsparterna kommer främst ifrån de stora länderna i Europa som England, Frankrike och Tyskland, men även Norge, Schweiz och Polen är med. Projektet går ut på att utveckla beräkningsgridinfrastrukturer i Europa. Fokus för projektet ligger på heterogena och väldigt kraftfulla beräkningsmiljöer. Man använder sig av gridteknologin för att möjliggöra en enhetlig bild och transparant åtkomst till sådana miljöer.
Det har förekommit en viss optimism i att kunna använda GC till DM ändamål [11] men frågan är då vad GC har att tillföra DM och de krav som DM ställer på systemen som DM:en skall utföras på. Detta är grunden till vårt intresse och uppsatsens inriktning.

1.2 Problemområde

En begränsande faktor för DM är att det är en resurskrävande process som därmed inte kan göras när som helst. Dessutom kan de data ett företag vill analysera vara utspridd över flera databaser. Vi vill finna sätt att övervinna eller i alla fall begränsa dessa problem genom att utnyttja de inneboende egenskaperna hos den nya gridtekniken som är under utveckling. Syftet är därför att utifrån de krav som DM ställer på systemet det skall köras på och de egenskaper som ett grid innehar, eller syftar till att inneha, undersöka vad grid kan tillföra DM.
Målet med denna rapport är att sammanfatta de DM krav som finns och de egenskaper som ett grid har eller kommer att ha så småningom och försöka föra dessa samman.

1.3 Frågeställning

I hur stor utsträckning kan beräkningsinriktade grids tillgodose de krav data mining ställer på ett system?

1.4 Var står grid computing idag?

GC har tidigare främst varit av intresse inom forskningsdiscipliner med stora behov av beräkningskapacitet, t.ex. fysik, meteorologi och bioinformatik. Globus [52] är en organisation som startats för att tillhandahålla standarder inom GC, främst för forskningsvärlden. Även näringslivet har dock börjat få upp ögonen för GC. Det finns bl.a. ett pågående projekt som helt riktar sig till näringslivet, Enterprise Grid Alliance [51], som syftar till att utveckla gridstandarder som är anpassade för affärsapplikationer för att företagen bättre skall kunna utnyttja tekniken. Det vanligaste som behandlas är dock fortfarande hur GC kan hjälpa vetenskapen. Den senaste satsningen i Sverige är Swegrid [55], som är ett samarbete mellan 6 universitet i Sverige för att få tillgång till mer datorkraft när det behövs. Istället för att köra arbeten enbart på ett universitet kan de nu få

tillgång till datorkraft från ytterligare fem universitet. Den här stora satsningen är ett led i ett större samarbete ute i Europa, där forskningsinstitutet Cern i Genève står i centrum. De har för avsikt att utnyttja datorer över hela Europa för att analysera data från ett simulerat experiment.
Ett av de företag som levererar gridinfrastrukturer är Entropia. De har publicerat en rapport från ett experiment som de utfört för att testa hur bra deras system är på att ta tillvara på överbliven datorkraft hos datorer som står passiva [6]. Detta skulle resultera i sparade pengar för företag eftersom det skulle minska nödvändigheten att investera i superdatorer för att klara av mer krävande uppgifter. I företag är superdatorer kanske mer sällsynta än i forskningsvärlden, men de finns där och då är det ännu mer frestande att slippa göra så pass stora investeringar när man kan samla ihop datorkraft hos datorer som ändå inte arbetar så mycket.
När det gäller att komma så nära vårt ämne som möjligt så finns det en artikel som heter ”Distributed data mining on the grid” [9] som är en studie om hur man har försökt bygga ett grid för att kunna utföra DM distribuerat över ett data grid. De menar på att fler och fler företag har sina data på olika platser runt om i världen och att man därför borde sammanföra alla dessa i ett stort data grid. Med hjälp av ett underliggande computational grid kan man sedan bearbeta sina data med DM på håll för att hitta relevanta mönster eller dylikt.

1.5 Avgränsning

Vi har valt att koncentrera oss på grids som är ämnade att aggregera processorkraft ur noderna - dvs. computational grids, och i något mindre utsträckning scavenging grids - och i princip utelämna data grids helt. Detta på grund av att data grids och computational grids skiljer sig så pass mycket att det i princip skulle bli två parallella undersökningar. Anledningen till att vi valde computational grids framför data grids är att vi inte såg några direkta beröringspunkter med DM - annat än just det faktum att användaren skulle få det lättare att komma åt data som ligger spridd över flera databaser. När det å andra sidan gäller computational grids försiggår själva DM-sessionen på gridet i sig och påverkas därmed mer direkt av gridets egenskaper.

Vad gäller DM har vi valt att hålla oss på en så generell nivå som möjligt. Detta betyder att de krav som presenteras i kapitel 2 inte nödvändigtvis är sådana att samtliga tillämpningsområden för DM ställer dem i alla upptänkliga situationer. Vi har helt enkelt utformat listan på ett sådant sätt att om samtliga punkter i listan uppfylls på ett fullgott sätt bör man kunna använda sig av den DM-metod man anser lämpligast för ändamålet utan att behöva oroa sig för att systemet skall orsaka problem. På ett mer allmänt plan kan sägas att uppsatsen inte är ämnad att jämföra computational grids med andra typer av HPC-system, som SMP-datorer och kluster.

1.6 Språkpolicy

Vi har valt att skriva uppsatsen på svenska, med vissa undantag. Citaten från en av intervjuerna (den med representanter för IBM) samt citateten från artiklar och böcker har vi valt att behålla på originalspråket. Vi ser det som osannolikt att en läsare som kan tillgodogöra sig innehållet i denna uppsats har uppnått den kunskapsnivå som krävs utan att ha tillräckligt goda kunskaper i engelska för att kunna förstå det som sägs i citaten. Därför anser vi att ett citat på originalspråket, utan den påverkan en översättning alltid kan sägas innebära, är bättre än en översättning av oss.

Inte helt oväntat förekommer det inom såväl GC som DM ett stort antal engelska termer och uttryck. Vi har valt att i möjligaste mån använda oss av termernas svenska motsvarigheter, men vissa uttryck har vi inte ansett att vi kunnat hitta någon fullgod motsvarighet till i det svenska språket.

De ord och uttryck vi inte ansett kunnat betraktas som självklara ens för den som är insatt i informatik har vi valt att presentera förklaringar för under rubriken ”definitioner”. Där återfinns även de termer vi inte ansett oss kunna översätta och inte förklaras i texten.

1.7 Definitioner

Cachning	Nedladdning av data i förväg, så att det den finns tillgänglig när den behövs.
Commodity Computing	Samlingsnamn för ett antal datorteknologier med kapacitet att arbeta distribuerat, bl.a. Java, CORBA och DCOM [17].
Front-end	Se *presentationslager.*
GridFTP	Filöverföringsstandard som utvecklats med avsikt att användas inom grid computing. Baserat på FTP (File Transfer Protocol).
GSI	Grid Security Infrastructure. Globus paket med funktioner för säker inloggning och dataöverföring.
High Performance Computing (HPC)	Samlingsnamn för alla former av kraftfulla datasystem såväl som forskning och utveckling runt och användande av desamma.
Mellanprogramvara	Programvara som ”översätter” mellan två eller flera program.
Middleware	Se *mellanprogramvara.*

Network File System (NFS)	Filöverföringsstandard som syftar till att emulera lokal filåtkomst vid fjärråkomst av data över ett nätverk.
Presentationslager	Den del av ett datasystem som hanterar interaktionen med användaren.
Quality of Service (QoS)	Begreppet QoS har sin grund i teorier som syftar till att mäta – och med hjälp av dessa mätningar garantera - kvaliteten på datakommunikation.
Sandlåda	Vanligtvis inkapsling av en applikation för att den skall kunna exekveras på ett system utan att riskera att applikationen skadar systemet. När det gäller GC skyddar inkapslingen även applikationen från påverkan från systemet.
Submit	Skicka in.
XRSL	eXtended Resource Specification Language. XML-baserat scriptspråk för att beskriva beräkningsjobb – dvs. bl.a. namn på filer som ingår (binär-, in- och utfiler) och resurskrav.

1.8 Disposition

I kapitlet ”Teori” redogör vi för DM i stort och de krav som DM ställer på system som det skall köras på. Vi tittar också på GC i stort och de egenskaper som GC har eller kommer att ha. Grunden bakom det tillvägagångssätt vi valt för rapporten redogörs för i kapitlet ”Metod” och i kapitlet ”Resultat” försöker vi föra samman kraven från DM med egenskaperna hos GC utifrån de intervjuer vi gjort och det övriga material vi tagit del av. I kapitlen ”Diskussion” och ”Slutsats” presenteras slutligen de slutsatser vi dragit av våra efterforskningar.

2 TEORI

I detta kapitel förklaras begreppen DM och GC. Dessutom redogörs det för de krav vi funnit att en plattform bör erbjuda för att tillhandahålla fullgott stöd för DM samt vilka egenskaper ett grid bör besitta.

2.1 Data Mining

2.1.1 Vad är Data Mining?

DM är ett sätt att utvinna information - och därigenom kunskap - ur en stor mängd data. De flesta företag samlar på sig en stor mängd data under en längre period. Dessnna datamängd kan vara mer eller mindre spridd över olika databaser och servrar. För att kunna ha någon användning av all denna data så måste man analysera den och identifiera det som kan vara viktigt och användbart. Detta är inte helt trivialt, eftersom det kan finnas samband som inte är omedelbart uppenbara; ofta helt enkelt för att man inte kan få en överblick över all data som man skall gå igenom [17]. Dessutom tar det tid, en resurs som man oftast inte har till övers. DM-verktyg hittar mönster och samband i stora mängder data och formulerar utifrån dessa mönster s.k. regler. Dessa regler kan sedan användas t.ex. i en organisations beslutsprocess.

Det finns fem olika sorters information man kan utvinna med hjälp av DM enligt [2].

- *Klassifikation*, där man klassar in någon/något beroende på vilka egenskaper som denna/detta har.
- *Klusterisering*, där grupper av data indelas beroende på vad de delar för egenskaper.
- *Association*, identifierar förhållanden mellan händelser som utspelar sig vid ett tillfälle.
- *Sekvensering*, ungefär som association förutom det att förhållandena utspelar sig under en längre tidsperiod.
- *Prognos*, estimerar framtida värden utifrån mönster inom stora datamängder.

Inom DM används ett antal metoder för att analysera data. Dessa kan t.ex. utgå från olika metoder inom statistisk dataanalys eller artificiell intelligens.
Här presenteras några av de metoder som förekommer flitigast i litteraturen.

Bayesiansk analys

Som stöd i en beslutsprocess kan man använda sig av bayesiansk analys för att bedöma den sannolika utgången av de olika alternativen. [29]

Bayesiansk analys av stora datamängder kan vara mycket krävande [23], vilket har gjort att det är först på senare tid som tillräcklig datorkraft blivit tillgänglig för att man skall kunna använda sig av den i någon större utsträckning. En fördel med bayesiansk analys är att man kan använda sig av tidigare kunskap när man utför den, vilket kan utnyttjas för att förbättra kvaliteten på nya analyser [34].

Artificiella neurala nätverk

Artificiella neurala nätverk är från början ett sätt att försöka efterlikna det sätt på vilket hjärnans neuroner arbetar (dvs. ett ”naturligt” neuralt nätverk). Tanken är att man kopplar samman ett antal noder (ej att förväxla med gridnoder) i ett nätverk med hjälp av förbindelser som var och en har en viss effektivitet (vikt). Nätverket tränas sedan att ge olika utdata beroende på vilka indata som ges.

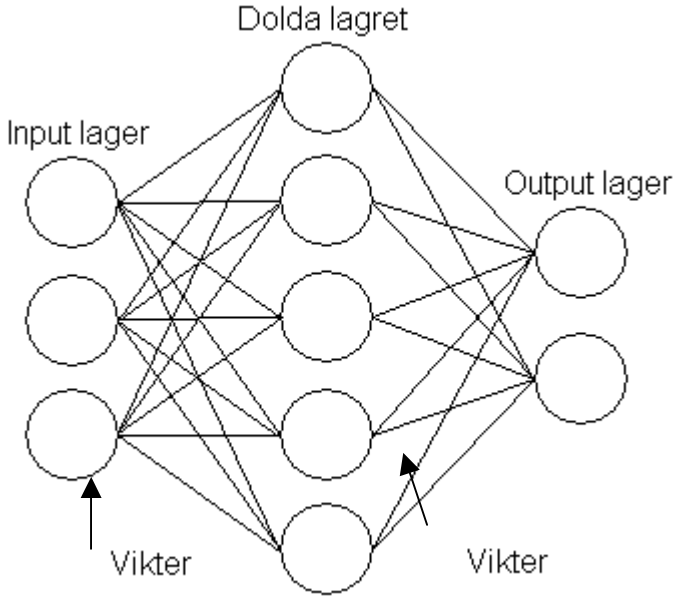

Fig. 2 Exempel på hur ett neuralt nätverk kan vara uppbyggt.

Beslutsträd

Beslutsträd syftar till att prognostisera eller klassificera objekten i en datamängd genom att stegvis genomföra tvåvägssplitar på de delpopulationer som steget innan resulterat i, baserat på någon egenskap som skiljer dem åt [23], [27]. Den resulterande regelmängden kan representeras grafiskt m.h.a. en trädliknande struktur. Inom DM skapas beslutsträd genom olika former av regression på s.k. training sets [28], [29].

2.1.2 Vilka krav ställer Data Mining?

I och med att DM kan användas för att lösa ett brett spektrum av uppgifter [32], [26], [36] varierar kraven på systemet från fall till fall. Efter långtgående studier av litteratur på området har vi isolerat följande kriterier ett system kan behöva uppfylla i större eller mindre grad för att det skall vara en lämplig platform för DM. Dessa kriterier behöver inte alltid tillfredsställas i lika stor grad hos alla DM-applikationer utan beror på dessa applikationers syfte.

- *Processorkraft*
- *Processorkraft på begäran*
- *Stora mängder minne*
- *Robusthet*
- *Säkerhet och integritet*
- *Enkel tillgång till data*

Processorkraft

För att utföra avancerade beräkningar på stora mängder data behöver man kunna göra ett stort antal beräkningar med så hög hastighet som möjligt. [16]

> *Even if the company has fairly centralised systems there are usually problems in getting sufficient resources on the computer to run a lot of tests.* [5]

> *Consistent patterns of overcharging and inappropriate early payment, leading to the discovery of thousands of pounds worth of fraudulent invoices and employee/supplier collusion; Orders for computer equipment being deliberately split so as to by-pass financial authority limits; Indications of regular customer "favouritism" uncovered by analysis of sales history and discount data; Analysis of trends in inventory write offs (previously discounted as miscoding) identifying mass pilferage. Analysis and results of the kind mentioned above often leave management wondering why this sort of information cannot be accessed on a regular basis - given enough resources, analytical tools and computing power it is of course possible but not always practical.* [5]

> *Workshop participants noted that while real-time data mining is not important for research, it is highly important for applications. Thus, the amount of effort that one puts into providing a very fast data mining system depends upon one's ultimate objective.* [25]

Processorkraft på begäran

Processorkraft på begäran handlar om att tillgodose möjligheterna till DM så fort användaren behöver det, och därmed också kunna förse användaren med de resurser som krävs för att omedelbart utföra DM operationer och dessutom få resultatet på en för användaren acceptabel tid.

> *"It's impossible these days to tell a user that you'll have their report or management chart ready in an hour or two. Our users want to have the information extremely quickly and insist on a tool that responds to their needs almost instantaneously," says Thierry Florentin, CIO Europe.* [39]

Det uppstår ibland tillfällen då detta kommer att vara nödvändigt, som indikeras av exemplet ovan. Det behandlar nödvändigheten i att få den information man behöver för att fatta beslut omgående. I sådana lägen kan man inte vänta tills helgen för att genomföra sin DM och sedan få resultatet på måndagen.

Stora mängder minne

För att kunna exekveras på ett effektivt sätt måste många algoritmer ha tillgång till en ansenlig mängd minne. Visserligen arbetas det på att ta fram mer minneseffektiva algoritmer, men för att få så stor frihet som möjligt i valet av algoritm bör tillgången till minne vara så god som möjligt. [16], [43]

> *…the basic philosophy driving RNA folding methods on massively parallel platforms is the point of view that memory is the fundamental resource bottleneck, rather than computational speed.* [16]

> *Although very efficient techniques have been presented, they still suffer from the same problem. That is, they are all inherently dependent on the amount of main memory available. Moreover, if this amount is not enough, the presented techniques are simply not applicable anymore, or significantly need to pay in performance.* [22]

Ett alternativ är naturligtvis att dela upp arbetet i mindre portioner, men det är inte alltid det är möjligt.

Robusthet

Robusthet behandlar frågan om hur pass stabilt och feltolerant systemet måste vara, att det inte händer, eller att man i alla fall begränsat chansen, att en DM-session påverkas av att det uppstår komplikationer hos systemet. I vårt fall är det frågan om stabiliteten hos systemet som man kommer att köra DM på. Kravet är inte ett universellt krav för all DM utan det handlar om, som vi redan poängterat, vilken sorts DM det är frågan om, vad DM-applikationen har hand om för uppgift.

Visst vore det önskvärt att DM-sessionen aldrig avbryts pga. komplikationer, men om man utför en DM-uppgift en gång i månaden över något som inte är tidskritiskt har det mindre betydelse.

> *Robustness. Real time business processes rely on data mining (results) (24 x 7)* [24]

> *As each control unit is produced, approximately 450 quality-control tests are performed and as many as 1.5 million data values are collected per day. Unfortunately, until recently, most of the collected data was not readily available to manufacturing engineers, technicians or operators. If a part failed a test, the results were saved and printed for later reference.* [38]

> *Tools for proactive analysis and action. An early-warning system scans operational data, ferrets out potential problems and automatically issues alerts to the right persons, so they can take action early.* [48]

För applikationstillämpningar som fungerar som ovan, dvs. att de skall kunna köras konstant samtidigt som organisationens övriga verksamhet, är robusthet synnerligen viktigt. Det är inte bara det att de skall köras hela dagen - de måste fungera under hela den tiden, då eventuella fel kan få stora ekonomiska konsekvenser.

Säkerhet och integritet

Då de data som behandlas kan vara av sådan art att det kan få allvarliga konsekvenser om den faller i orätta händer, måste systemet garantera att obehöriga inte kan avläsa eller påverka de uppgifter som behandlas. [13]

> *Many security and counter-terrorism-related decision support applications need data mining techniques for identifying emerging behavior, link analysis, building predictive models, and extracting social networks. They often deal with multi-party databases/data-streams where the data are privacy sensitive. Financial transactions, health-care records, and network communication traffic are a few examples.* [13]

Enkel tillgång till data

Beredandet av de data som skall behandlas är den mest tidskrävande delen av DM. [15]. För att underlätta detta moment är det viktigt att systemet tillåter en så *enkel tillgång till data* som möjligt, dvs. att systemet tillhandahåller funktioner för åtkomst och beredning av data. [17], [31]

2.2 Grid Computing

2.2.1 Vad är grid computing?

GC är en vidareutveckling på distribuerad databehandling. Distribuerad databehandling har tidigare förekommit främst i forskarvärlden som ett sätt att kunna utföra stora beräkningar som skulle ta alldeles för lång tid på en enskild dator. Tekniken inriktar sig på att koppla ihop flera datorer (noder) för att samarbeta och utnyttja deras processorkraft.
GC skiljer sig från distribuerad databehandling genom att ställa högre krav vad gäller generalisering [40].

En förenklad förklaring av hur GC fungerar är följande:
Användaren skapar, med hjälp av ett gränssnitt eller en applikation, en uppgift för systemet. Uppgiften skickas sedan till gridet där den s.k. mellanprogramvaran avgör vilken eller vilka noder som är lämpliga att sköta de beräkningar som skall göras. Idealt är jobbet uppdelat på flera, sinsemellan oberoende, subjobb för att man skall kunna genomföra beräkningarna på flera noder samtidigt för att därigenom minska den tid som går åt för att genomföra beräkningarna. Hur de individuella noderna ser ut kan variera kraftigt – det kan vara alltifrån en gammal arbetsstation till ett toppmodernt kluster. Användaren behöver inte ha någon kännedom om hur det fysiska gridet ser ut, utan behöver bara invänta resultatet som om hela arbetet utförts på en enda dator.

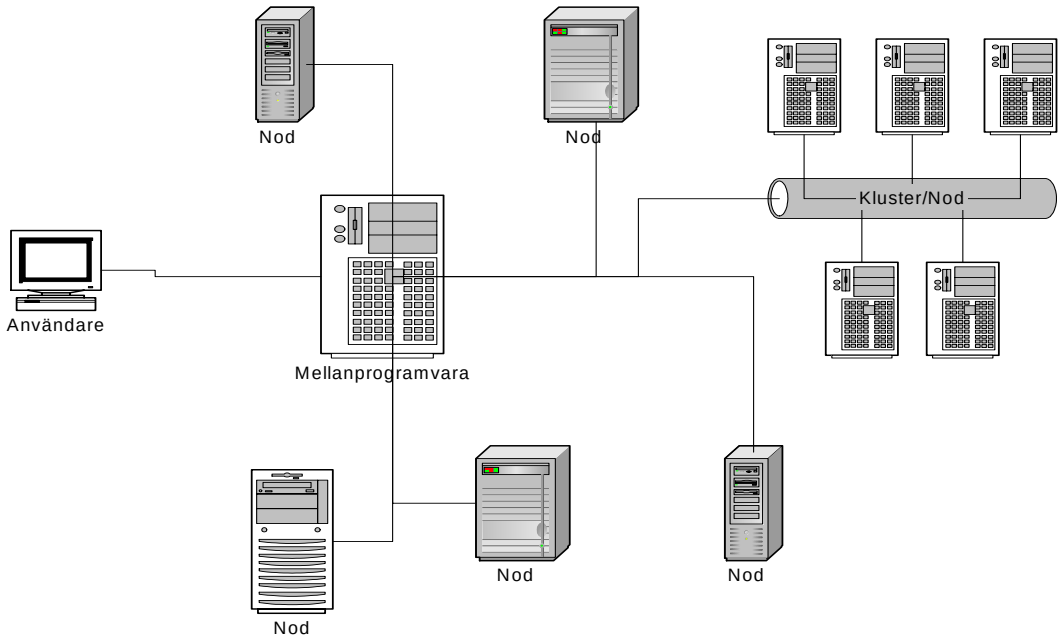

Fig. 1 Mycket förenklad bild på hur ett grid kan vara uppbyggt. Användaren använder sig av ett gränssnitt för att skapa en uppgift för gridet, som av mellanprogramvaran, vilken befinner sig i presentationslagret, delas ut till noderna utefter nodernas individuella kapacitet.

Det råder en viss brist på konsensus rörande exakt vilka system som kan betraktas som grid [44]. Vi har valt att följa [40] där grid definieras som ett system som:

- Samordnar resurser som inte är underställda central kontroll.
- Använder sig av standardiserade, öppna och generella protokoll och gränssnitt.
- Levererar icketriviala Qualities of Service.

Det finns flera sätt att kategorisera gridsystem. Jacob sorterar i [42] in gridsystem i tre olika kategorier - *computational grids, scavenging grids* och *data grids.*

- *Computational grids* är när man har kopplat ihop flera noder som har resurser åsidosatta, som alltså är dedikerade till att användas till jobb som man vill ha utförda. Computational grids skiljer sig från *kluster* genom att vara mer flexibelt vad gäller heterogenitet hos noderna.
- *Scavenging grids* liknar computational grids på så sätt att flera datorer samarbetar för att genomföra beräkningar, men till skillnad från computational grids är de i systemet ingående datorerna inte dedikerade till systemet, utan kan, t.ex., vara en arbetsstation. När en dator står overksam ställs den till systemets förfogande och genomför de arbeten som den tilldelas av systemet ända tills den används till sitt egentliga syfte igen.
- *Data grids* riktar in sig på att underlätta för s.k. *virtual organizations* att dela data med varandra, genom att erbjuda verktyg för enhetlig lagring och åtkomst av data.

Vi fastnade för ovanstående kategorisering, eftersom den ger en tydlig och teknisk indelning, som låg i linje med den tonvikt vi avsåg lägga på computational grids framför data grids.

För att en applikation ska kunna utnyttja de fördelar gridsystem erbjuder fullt ut måste det vara anpassat för dylika (man talar om *grid-aware applications* [30]). För resurskrävande uppgifter effektiviseras arbetet till exempel om man kan dela upp det på flera mindre jobb som går att utföra parallellt [33]. Ju mindre dessa jobb beror på varandra, desto bättre kan applikationen ta tillvara resurserna i ett grid.

2.2.2 Vad har Grid Computing att erbjuda?

Genom att studera litteratur på området har vi kompilerat en lista över egenskaper som det anses viktigt att ett grid innehar. Dessa egenskaper är *effektivitet, lättadministrerbarhet,* r*obusthet, Quality of Service* (QoS), *diskretion, öppenhet/enkelhet av applikationsintegrering, säkerhet* och *skalbarhet.*

Effektivitet

GC ”skördar” oanvända klockcykler för att aggregera processorkraft och effiktivitet är då hur bra den gör detta. Helst skall den skörda så gott som alla oanvända klockcykler (Entropia gridet lyckas med detta till 95%). [6]

Lättadministrerat

När man pratar om GC är det ofta i termer av ett nät som har kanske tusentals eller t.o.m. hundratusentals noder kopplade till sig. Detta nät kommer då att kräva mycket bra administrativa verktyg för att kunna hållas igång i gott skick. I ett fall med så här många noder så vore tumregeln att ha en administratör för varje 200 datorer helt omöjligt. Åsikten är den att hela gridet skall vara administrerbart av ett satt antal personer och inte kräva fler om fler noder läggs till. Det är också en fördel om det är enkelt att lägga till och ta bort noder ur nätet. En annan aspekt av den lätta administrationen är att klienten som gridet lånar resurser av skall förbli i samma skick före som efter. [6]

Robusthet

Robusthet på gridet handlar om att gridet måste kunna klara av att hantera smärre fel. Det måste klara av att behandla klart en uppgift även vid t.ex. resurs- eller nätverksfel. Det är också en fråga om att kunna förse användaren med förutsägbar prestanda trots att det är en något oförutsägbar grund som gridet är baserat på. [6]

Quality of service

QoS är ett begrepp som vanligtvis associeras med datakommunikation och handlar om hur ”bra” datakommunikationen fungerar mellan två punkter (mest typiskt, i sammanhanget, två datorer som kommunicerar med varandra i ett nätverk).
Eftersom gridsystem är relativt skiftande i sin kapacitet, i alla fall de som bygger på scavenging grid och inte har renodlade resurser åsidosatta till beräkningar, så är QoS något som har stor betydelse. I en gridsystemkontext kan man urskilja olika indelningar av QoS [11]. Det pratas om applikations-QoS (A-QoS), som behandlar användarens uppfattning av ett körande program. A-QoS påverkas därför starkt av vilken portal som används och interfacet mellan portalen och

gridsystemet. För att klara av realtidsinteraktion så måste A-QoS bibehållas inom rimliga gränser. Den andra nivån för QoS förhåller sig direkt till gridsystemet och benämns mellanprogramvaru-QoS (M-QoS). M-QoS styr hur lång tid det tar att generera nya uppgifter och exekveringen av dessa. Den här nivån av QoS måste mätas mellan gridsystemet och den eller de resurshanterare som används vid ett givet tillfälle. Den tredje och sista nivån av QoS behandlar nätverket och resurserna. Det finns gott om litteratur om hur man mäter, övervakar och i vissa fall ser till att det upprätthålls. Nätverks-QoS (N-QoS) (där man bl.a. intresserar sig för bandbredd och fördröjningar) har blivit grundligt utforskat. Man kan också se en viss möjlighet till existensen av resurs-QoS, där saker som antalet processorer eller minsta möjliga minne blir intressant att beakta. För att applikationerna skall kunna köras effektivt genom gridet blir det nödvändigt att slå samman alla nivåerna av QoS som nämnts ovan [11].

Diskretion

Använder man sig av resurser som man inte ”äger”, utan delar med, eller rent av lånar av, andra verksamheter inom organisationen, skall gridsystemet inte påverka utomstående processer. Exempelvis skall användaren av en arbetsstation i ett scavenging grid inte märka av systemet i form av minskad prestanda hos de program vederbörande använder sig av, ens om gridmjukvaran havererar. [6]

Öppenhet/enkelhet av applikationsintegrering

I grund och botten så är gridet ett system på vilket man skall köra applikationer. Antalet och de olika varianterna av applikationer man kan köra bedömer dess nytta. Gridsystemet måste stödja så många olika applikationer som möjligt, med olika behov och skapade med olika modeller, utan någon större ansträngning. [6]

Säkerhet

En viktig del i all datakommunikation är säkerhet, mycket mer så när det gäller grid computing. Eftersom tanken med grid computing är att man skall dela på resurser - processorkraft såväl som lagringsresurser - så gäller det att en viss nivå av säkerhet säkerställs. I princip kan man säga att det finns tre säkerhetsaspekter som är aktuella inom grid computing. För det första skall data transporteras mellan användare, presentationslager och noder, vilket gör att nätverkskommunikationen måste vara säker [47]. För det andra måste man säkerställa integriteten för data och applikation när de befinner sig på en nod, för att förhindra att noden, eller dess ägare, påverkar jobbet på något sätt [1], [6]. Slutligen måste man förhindra att avsiktligt skadliga applikationer kan påverka gridets resurser [1], [6]. Säkerhet blir viktigare ju känsligare den inblandade datan är.

Skalbarhet

För att ett gridsystem eller en –applikation skall kunna tillgodogöra sig de tillgängliga resurserna optimalt är skalbarhet viktigt. Prestanda per processor skall påverkas så lite som möjligt av hur många processorer som ingår i systemet.
Eftersom företag idag använder sig av en stor mängd datorer, från 1000 upp till 100000 stycken är skalbarheten en viktig egenskap hos systemet. Det skall sköta sig oavsett hur många noder som ingår i gridet. Observera att det är lika viktigt att systemet är skalbart uppåt som nedåt. [6]

3 METOD

I detta avsnitt diskutera vi metoder och metodval vi gjort, samt de ansatser som ligger till grund för materialinsamlingen.

3.1 Teoretiskt ramverk

Det finns ett antal olika synsätt och metoder man kan utgå från när man genomför en studie. Nedan redogörs de vi valde mellan.

3.1.1 Synsätt och metod

En forskningsrapport kan ha en av tre huvudsakliga synsätt vad gäller hur den skall förhålla sig till omvärlden. Den kan en traditionell rapport, där man utgår ifrån att omvärlden är objektiv, eller så kan den vara kvalitativ, där man säger att omvärlden är subjektiv [4].
Det traditionella synsättet, även kallat positivism [35], menar att det existerar en objektiv verklighet som är skild från människan. Denna verklighet kan man få kunskap om genom att iaktta den. Utifrån dessa iakttagelser försöker man sedan förklara världen.
Det kvalitativa perspektivet anser att man endast kan se omvärlden subjektivt, det går således inte att distansera sig från omvärlden och tro att man kan iaktta hur saker och ting verkligen förhåller sig. Det kvalitativa synsättet förhåller sig mer till individen än omvärlden i sig, och försöker ta reda på hur individen uppfattar sin omvärld [4].
När man har valt perspektiv måste man också fundera ut hur man skall gå tillväga för att studera det man intresserar sig för. Det är här valet av metod kommer in. Metodvalet är väldigt beroende av vilket sorts problem och vilken frågeställning forskaren har valt [35]. De två som oftast förekommer är den kvantitativa och den kvalitativa metoden.
Den kvantitativa metoden lämpar sig för när man har något mätbart där man skall ta till siffror och beräkningar för att göra t.ex. jämförelser och dylikt som kräver mätbara värden. Denna metod används mycket inom de naturvetenskapliga ämnena.
Den kvalitativa metoden använder sig inte av siffror eller tal utan av verbala formuleringar för att bedriva sin forskning.
I takt med att forskningsvärlden växer ökar behovet av att få en överblick över den kunskap som produceras. Detta har lett till etablerandet av den tredje typen av forskningsrapport – forskningsöversikten. Forskningsöversiktens mål är att sammanställa forskningen inom ett eller flera områden och presentera den kunskap som går att utvinna ur de granskade publikationerna. Det räcker dock inte att ha som enda mål att sammanställa det som andra gjort – man bör ha en mer ambitiös frågeställning än så.

Notera dock att bara för att man valt ett kvalitativt synsätt så följer det inte automatiskt att man måste använda sig av en kvalitativ metod, lika lite som det traditionella synsättet inte innebär att man måste ha en kvantitativ metod.

Den här rapporten är en positivistisk rapport med kvalitativ metod och vi ämnar lösa vår frågeställning genom intervjuer och litteraturstudier.

3.1.2 Intervjuer

Det finns många olika sätt att lägga upp en intervju på. De olika sätten bestäms genom de två olika dimensionerna en intervju har, graden av standardisering och graden av strukturering [35]. Standardisering är hur mycket som lämnas till intervjuaren vad gäller utformningen av frågorna och i vilken ordning de ställs. Graden av strukturering har att göra med hur mycket frihet intervjupersonen har att tolka frågorna utifrån sin egen inställning eller tidigare erfarenhet. När intervjun är lågt standardiserad eller helt icke-standardiserad så hittar intervjuaren på frågorna allteftersom intervjun pågår och ställer dem i den ordning han anser att de bör ställes för just den här intervjupersonen. På så sätt kan intervjuaren gräva sig djupare ner i saker som intervjupersonen råkar nämna under intervjun. Om intervjun är helt standardiserad så ställs exakt samma frågor i exakt samma ordning till alla intervjupersoner och ger därmed liten chans till att söka kunskap som man kanske inte räknat med att finna. När det gäller strukturering så är det en fråga om hur mycket svarsutrymme som intervjupersonen har. En starkt strukturerad intervju ger väldigt lite utrymme åt intervjupersonen att elaborera sitt svar. I vissa fall är det så strukturerat att svarsalternativen är givna från början och intervjupersonen bara har att välja bland dem. Detta görs oftast för att kunna jämföra svaren mellan olika intervjupersoner. I en ostrukturerad intervju så har intervjupersonen fritt spelrum, till en viss gräns naturligtvis, och uppmuntras att tala fritt inom frågan för att kanske öppna för nya frågor. Genom att kombinera de fyra ytterligheterna får vi fyra möjliga sätt att konstruera en intervju på.

3.1.3 Litteraturstudie

Inom all forskning måste man ta hänsyn till vad som redan skrivits [4]. Det är viktigt att ta reda på vad som redan har skrivits om ämnet och vad det resulterade i. Detta gör att man inte alltid behöver börja från början med ett ämne och det leder också ofta till att man ser ett möjligt utgångsläge på en ställd fråga. Man måste också ta hänsyn till vem som skrivit det, något som blir extra uppenbart i detta fall.

3.1.4 Reliabilitet och validitet

I en kvantitativ undersökning pratar man ofta om reliabilitet och validitet [35]. Reliabilitet har att göra med hur pålitliga de mätinstrument är som man använder under forskningens gång, alltså kommer de att ge samma värden varje gång man

använder dem under samma förhållanden. Validitet avser om man faktiskt har mätt det man föresatt sig att mäta. Dessa begrepp finns även i den kvalitativa forskningen, men pga. av avsaknad av faktiska mätinstrument och därmed mättekniska problem så heter de och definieras de annorlunda. Istället för reliabilitet pratar man om trovärdighet och istället för validitet pratar man om rimlighet. När man skall göra en kvalitativ undersökning så står forskaren inför lite annorlunda problem än vad en kvantitativ forskare gör. Vid en kvalitativ undersökning så är det forskaren som står i centrum både vid insamlandet av information och vid analysen [35]. Eftersom man inte har något att mäta på samma sätt som hos en kvantitativ undersökning är det tankarna och idéerna som står i centrum. Det blir då svårare att bedöma eftersom det inte finns några yttre kriterier och man tvingas då istället ställe er rad kritiska frågor [35]: Innehåller dokumentationen information som berör det centrala fenomenet? Är den teoretiska referensramen rimlig i förhållande till den allmänna kunskapen om fenomenet? Utnyttjas den totala informationen? Vilka alternativa tolkningar finns? Rimligheten, som omtalas i [35], handlar till stor del om att kritiskt granska de tolkningar som gjorts. Rimligheten kan bedömas utifrån hur mycket man utnyttjat den information man insamlat, liten utnyttjandegrad kan leda till att man dragit slutsatser som inte nödvändigtvis visar hela sanningen. Att försöka jämföra alternativa tolkningar är också ett sätt att försöka garantera rimligheten. Ett annat sätt att bedöma rimligheten på är att låta uppgiftslämnarna granska tolkningarna som gjorts utifrån deras uppgifter, så att de kan avgöra om det som står där överensstämmer med deras uppfattning av situationen som tolkningarna syftar till. Vad gäller trovärdigheten så är det en fråga om att visa att tolkningarna inte bygger på stereotypa uppfattningar, förutfattade meningar eller lättillgängliga slutsatser [35]. Trovärdigheten i tolkningarna kan t.ex. kontrolleras av uppgiftslämnaren. Denne måste dock göras medveten om de perspektiv och utgångspunkter som forskaren har och förstå dessa, för att sedan kunna ges möjlighet till att själv ge sig i kast med att pröva, acceptera eller förkasta tolkningarna.

3.2 Praktiskt genomförande

För att kunna svara på frågeställningen använde vi litteraturstudier till att finna de krav DM ställer på ett system. Vi har även använt litteraturstudier för att få en djupare förståelse för GC och dess egenskaper. Resultatet av dessa litteraturstudier fick sedan ligga till grund för den första versionen av intervjufrågorna till de olika intervjuobjekten.
Resultatet av intervjuerna har sedan använts för att ge en bild av hur GC, och GC inom DM, ser ut i praktiken i dagsläget. Denna bild har sedan fått ligga till grund för resonemanget kring frågeställningen.

3.2.1 Intervjuer

När våra intervjuer skulle konstrueras så funderade vi på vad det var vi ville få ut av dem i form av material. Eftersom vi inte hade en klar uppfattning om vad det fanns för möjliga kopplingar inom vårt ämne valde vi att göra intervjuerna icke-standardiserade och ostrukturerade. På det här sättet så kunde vi spinna vidare på de längre utläggen vi hoppades på att få som svar från våra intervjupersoner. En annan anledning till att vi valde att göra intervjuerna icke-standardiserade var att de personer vi intervjuade arbetade med vårt ämnesområde lite olika. IBM och SAS personerna arbetade visserligen med både GC och DM men de hade tyngdpunkten på var sitt håll. Den ena var mer inbegripen med GC och den andra med DM så därför vinklades frågorna olika beroende på vem vi intervjuade. Vi valde att göra dessa intervjuer över e-mail, vilket resulterade i en stadig ström av e-mail fram och tillbaka där vi skickade en mindre mängd frågor i taget. Svaren på dessa frågor resulterade i ett antal nya frågor. Detta fortsatte tills vi ansåg att vi uttömt möjligheterna till mer information. Frågorna till de här två personerna skilde sig dock åt av ovan nämnda anledning. Allteftersom svar kom in ställdes det motfrågor för att få dem att elaborera sina svar för att vi skulle kunna få ut så mycket information som möjligt, även sådant vi förbisett i våra utgångsfrågor. När det kom till Swegrid intervjuerna resonerade vi som så att alla hade ungefär samma inriktning så vi lade upp intervjuerna lite annorlunda än vad vi hade gjort tidigare. Till att börja med gjordes dessa intervjuer över en högtalartelefon och med hjälp av en diktafon, förutom i ett fall som gjordes ansikte mot ansikte. Intervjuerna gick till så att vi började med litet antal frågor som vi ställde till den första personen och efter den intervjun så lade vi till frågor som hade kommit upp under intervjun. Med dessa nya frågor samt de gamla utförde vi så nästa intervju. På det här sättet lade vi till fler och fler frågor efter varje intervju tills vi kom till den sista. Efter den sista intervjun gick vi tillbaka till början med de frågor som varje intervjuperson inte hade ställts ännu. På det här sättet har alla intervjupersonerna fått svara på samtliga frågor.
Vad gäller valet av intervjupersoner resonerade vi enligt följande. De som har närmast kontakt med ämnesområdena är de som forskar med dem alternativt de som sysslar med dem i sina jobb. I Sverige finns det just nu det pågående projektet Swegrid som nämnts ovan och vi ansåg att dessa borde ha en stor kunskapsbas som vi kunde ta del av. Eftersom vi också var intresserade av hur verkligheten såg ut lika mycket som hur forskarna såg på vårat ämne ville vi få kontakt med företag som sysslade med vårat ämnesområde. Därför intervjuade vi ett par på etablerade företag för att avgöra om det finns någon verklighetsanknytning till det som vi önskade undersöka.

Swegrids [55] verksamhet finns beskrivet ovan i avsnitt 1.4. De personer som vi har intervjuat är Swegrids kontaktpersoner för de lokala noderna i Göteborg, Lund och Linköping.

IBM [53] har intressen inom det mesta som rör datorer. De tillhandahåller produkter för både företag och privatpersoner inom hård- och mjukvara samt konsulttjänster.

När det var dags att välja ut lämpliga objekt för intervjuer tyckte vi därför att IBM föreföll vara en perfekt kandidat. Detta eftersom deras databas DB2 har ett integrerat verktyg för parallell DM i form av intelligent miner for data i kombination med det faktum att de har intressen inom GC – och t.o.m. föresatt sig att gridanpassa så stor del av sina produkter som möjligt.
Vårt första försök att kontakta IBM, via webbformulär, var föga framgångsrikt. Vi valde därefter en mer direkt metod och ringde IBM:s kontor i Stockholm, som hänvisade oss till Mats Mohlin, deras tekniska säljstöd för DB2.

SAS institute [54] tillhandahåller affärslösningar för informationshantering, t.ex. data warehousing och verksamhetsanalys.
Vi valde dem för att de är ett av de främsta företagen inom DM. När vi studerade företagets hemsida, för att bilda oss en uppfattning om de lösningar de erbjöd, hittade vi dessutom en viss indikation på att deras produkter var gridanpassade.
Vi kontaktade deras ansvarige för examensarbeten och liknande, som omgående satte oss i kontakt med Börje Edlund, produktchef på deras Stockholmskontor.

3.2.2 Litteraturstudie

I inledningsskedet hade vi bara en allmänt konceptuell uppfattning om vad GC innebär, eller på vilket sätt det skiljer sig från distribuerad databehandling. Därför riktade vi först in oss på mer lättförståeliga artiklar som fanns publicerade på Globus och IBM:s hemsidor – däribland de respektive organisationernas faq:er.

När vi sedermera gick vidare till den mer akademiska delen av våra litteraturstudier har vi mest sökt igenom de databaser som var datororienterade. Dessa databaser hittade vi genom Universitetsbibliotekets databas sök. Man skulle kunna ha tänkt sig att leta bland andra ämnesområden som utnyttjar t.ex. DM men vi utgick ifrån att dessa artiklar då skulle hamna på datororienterade databaserna i alla fall. Vi har tagit del av material angående både GC och DM i ett försök att bilda oss en uppfattning av hur möjligheten ser ut för att kunna sammanföra de båda tekniska lösningarna. Detta har gjorts genom sökningar i Göteborgs universitetsbiblioteks databaser samt genom att läsa grundläggande litteratur inom DM. Vi gjorde även en undersökning över vilka företag som sysslade med GC och DM för att hitta lämpliga intervjuobjekt på företagsfronten. De vi valde hade mycket information att tillgå på sina hemsidor och vi tillbringade en längre tid med att söka oss fram till huruvida de hade några synpunkter på det vi sysslade med. Även Swegrid hittade vi genom artiklar som vi studerat.

4 RESULTAT

Resultaten nedan följer på våra intervjuer och det material vi samlat in under litteraturstudierna. Kapitlet struktureras upp efter de krav som DM ställer och vi går igenom vad vi kommit fram till i våra efterforskningar krav för krav.

4.1 Processorkraft

Eftersom syftet med computational grids är just att producera stora mängder datorkraft, kan man tycka att det skulle vara trivialt att visa att grids kan möta DM:s behov av processorkraft. Problemet med detta är dock att det inte är helt trivialt att ta tillvara denna datorkraft då man, i dagsläget, inte kan se ett grid som en homogen pool ur vilken man kan ösa önskad mängd processorkraft. Istället är ett grid snarare en låda med olikformade klossar av processorkraft som kan kombineras för att på bästa sätt fylla användarens behov av datorkraft. Frågan om GC kan tillhandahålla tillräcklig mängd processorkraft blir då snarare om GC kan tillhandahålla tillräcklig mängd processorkraft som DM kan ta tillvara. De egenskaper man då bör se till hos GC blir då de som direkt syftar till att underlätta tillvaratagandet av processorkraften, dvs. skalbarhet och N-QoS.

Ett problem som skulle kunna uppstå är om inte skalningen fungerar ordentligt, dvs. att systemet får problem om man vill lägga till fler noder för att öka sin tillgång till processorkraft. Enligt [19] har man inom GC tidigare inte bekymrat sig så mycket om skalbarhet. För att råda bot på det startades Commodity Grid Project (CoG) [49] som en del av Globus för att på så sätt integrera kunskap inom commodity computing. En av fördelarna med detta är att commodity computing redan tidigare ägnat sig åt att säkerställa skalbarhet [19]. I ett antal rapporter, t.ex. [6] och [7], har olika tester gjorts för att mäta skalbarheten hos olika system som genererat lovande resultat.
Inom Swegrid ser man inte skalbarheten som ett problem, förutsatt att man tänker på det och vidtar lämpliga åtgärder när man skalar upp. En av de intervjuade inom Swegrid hade följande synpunkt:

> *Det är lite grann middleware relaterat, jag tror inte det är ett problem egentligen men det man kanske måste tänka på är att bygga ut front-endarna, typ dom som kör middlewareprogramvaran, att man har bra servrar där och kanske lasthanterare också, för att om det är mycket jobb på nätet så kan det bli mycket jobb som middleware får hantera.*

I övrigt uttrycktes det mest optimism för möjligheten att skala upp:

> *…Swegrid kan nog växa 10 potens till utan att det blir några tekniska problem…*

N-QoS är den aspekt av QoS som fått mest uppmärksamhet. Dels genom allmän forskning inom QoS för nätverkskommunikation [37], [12] och dels genom gridspecifik forskning [21]. Det finns därmed ett flertal lösningar för hur N-QoS skall säkerställas. Globus funktioner för QoS finns samlade i paketet GARA där en resurshanterare, som även har ansvaret för gridets övriga resurser, har ansvaret för QoS [41].
Swegrid har identifierat just nätverkskommunikationen som en av de begränsande faktorerna för utnyttjandet av gridresurser. Detta trots att Swegrid har tillgång till Sunets GigaSunet-nätverk som är på 10 Gbit/sek. En av de intervjuade inom Swegrid hade följande att säga om nätverkskommunikationen:

> *Det är nätverksförbindelserna som ställer till det, när man har stora datamängder att förflytta till höger och vänster så måste ju de här laddas upp och laddas ner… jag har märkt på vår site att dataöverföringen tar mycket kraft från front-enden…*

SAS uttryckte viss oro för scenariot där resultaten skall sammanföras för att tolkas då detta kan leda till flaskhalsar på presentationslagret.

En annan viktig faktor för hur väl ett grids processorkraft kan tas tillvara är hur väl applikationen anpassats till en gridmiljö. Swegrid är ett typexempel på ett grid som främst lämpar sig för uppgifter som består av flera, sinsemellan oberoende, delproblem:

> *…problem där man också kan dela upp det i delproblem där delproblemen är ganska oberoende av varandra så att dom kan köra oberoende och inte behöva kommunicera lika mycket med varandra… det är den klassens problem som Swegrid är inriktat på.*

Även inom SAS är det på det sättet man utnyttjar GC:

> *Inom datamining kan man tex träna modeller eller distribuera scoring av kunder över flera noder.*

4.2 Processorkraft på begäran

Processorkraft på begäran finns det vissa svårigheter med än så länge. Hos Swegrid fungerar det så att mellanprogramvaran i första hand försöker hitta ett system där det finns tillgänglig kapacitet, alltså att den kan fås omedelbart. Om detta inte går så börjar mellanprogramvaran titta på kötider och försöker uppskatta vilken nod som jobbet skulle få starta snabbast på och så skickar den jobbet dit. Gridet i sig har inget kösystem utan delar ut jobben på noderna som sedan själva har ett kösystem. I och med detta finns det inget sätt för gridet att själv organisera och prioritera vissa jobb framför andra. Det enda användaren kan göra för att försöka få sin processorkraft på begäran är att skicka in jobbet igen och hoppas på att det landar på en nod med ingen eller i alla fall kortare jobbkö. En Swegridanknuten hade följande åsikt:

> *Det är ju en grej som är lite av grids baksida, det är just det här med schedulering på gridnivå är ju inte speciellt bra tycker jag. Det är egentligen klienten som bestämmer var det skall hamna, sen frågar den systemet efter närmast tid för en submit, men när du väl hamnat på en site så är du ju där med ditt jobb och då är det ju upp till den sitens begränsningar om du får köra och när.*

En annan Swegridanknuten påpekade följande:

> *Just nu finns det ingen möjlighet att migrera jobb. Om det är väldigt lång kö på ett centra så skulle du bara migrera ditt jobb, men det går inte nu.*

Det finns just nu inte heller någon form av prioriteringssystem hos Swegrid utan det är först till kvarn som gäller. Hos Entropia däremot finns det ett kösystem som är associerat med mellanprogramvaran. Där finns det ett prioriteringssystem som dessutom, om jobbet fallerar, ökar prioritering på jobbet när det skall köras igen [6].

Ett annat problem som kan uppstå är att du har väldigt mycket jobb i omlopp men du har ett litet nät eller inte har en tillräckligt bra infrastruktur i presentationslagret. Nätet i sig sätter begränsning på hur mycket trafik som kan passera fram och tillbaka och presentationslagret sätter begränsningar i och med att den kanske inte kan hantera alla jobb som forsar förbi och kan därför leda till att jobbbehandlingar sätts i kö hos mellanprogramvaran. En Swegridanknuten hade följande att säga:

> *…om det är mycket jobb på nätet så kan det bli mycket jobb som middlewaret får hantera. Den sköter ju både submit till noderna och hanterar informationssystemet. Så att när användare submitar går ju klienten ut och frågar noderna och då kan det bli mycket frågor som skall besvaras och snabbt. Har man då mycket jobb på maskinen då lagrar det data över längre tid och då kan det ta tid och parsa igenom det.*

4.3 Stora mängder minne

I likhet med processorkraft är minnet i ett grid inte en koherent massa som kan utnyttjas precis hur man vill. Även om minne är mer flexibelt än processorkraft när det gäller att kunna expandera de individuella "klossarna" efter behov genom att dagens operativsystem har möjlighet att använda hårddiskutrymme till att virtuellt expandera primärminnet, inverkar utnyttjande av den möjligheten menligt på en applikations beräkningshastighet. Detta gör att man inte automatiskt kan förutsätta att ett grid kan tillgodose hur stora minneskrav som helst. Om man inte kan dela upp en beräkning som kräver mer primärminne än noderna kan erbjuda får man vara beredd på att gridets förmåga att effektivt

kunna behandla jobbet blir lidande. Gridet i sig har sålunda inte möjlighet att ge dig mer minne för en applikation än vad som faktiskt någon nod i gridet har tillgång till. Däremot kan du i ett jobbeskrivingsscript ange att din applikation kräver ett minimum av en viss mängd minne och på så sätt utesluta alla de noder som inte uppfyller detta minneskrav. Detta kan du även göra med andra krav än minneskrav. En Swegridanknuten hade följande förklaring:

> *Det gör du i ditt beskrivningsscript, när du begär att få köra så gör den en fråga på nätet, eller på gridet då, om det här kravet kan tillgodoses på någon av noderna.*

Vad som är viktigt att komma ihåg är att gridet tillgodoser minimikravet och bryr sig inte om att försöka optimera kravet, alltså ta en nod som kan tillgodose kravet men inte överstiger det med alltför mycket. På frågan om man kan tilldelas en nod med 4 gb minne även om det finns en ledig med 1 gb minne och man bara behöver 1 gb, svarar ett intervjuobjekt från Swegrid så här:

> *Du kan ju få den med 4 gb även om du har ett jobb som bara kräver 1 gb.*

Gridet ser sålunda enbart till minimikravet och sen är det en frågan om vilken nod av de som klarar minimikravet som skulle låta jobbet starta snabbast.

Entropia har ett liknande system där fördelningen av jobben sköts av mellanprogramvaran som, utifrån information om vilka noder som passar bäst för jobben, delar ut jobben på passande noder. [6]

Henri Casanova presenterar i [8] ett förslag till en lösning där en agent används för att förstå vad användarens applikation har för behov. Agenten väljer sedan var applikationen skall exekveras och tar även hand om förflyttning av eventuell data, jobinitiering och kan även modifiera hur applikationen körs utifall att resursläget skulle ändra sig. Valen som agenten gör baserar sig på var de eventuella data som skall användas är belägen någonstans såväl som grundläggande egenskaper hos resursen, t.ex. CPU-kraft och även vad resursen har för id. Det som kanske är mest intressant är att den även tar hänsyn till dynamiska egenskaper så som hur belastningen på resursen ser ut för tillfället, samt hur den ändrar sig under körningen.

4.4 Robusthet

Med tanke på att ett grid består av en stor mängd noder så är det väldigt liten sannolikhet att alla dessa kommer att gå ner samtidigt eller ens någonsin vara nere simultant överhuvudtaget. Om en applikation som kör på Swegrid av någon anledning inte får den data som den behöver så dör applikationen. Om en nod i Swegrid som en applikation körs på av någon anledning går ner så dör applikationen också. Dessa två scenarion vore inte ett problem i sig men det är också så att det inte finns någon kontroll för om applikationen lyckades slutföras

och som vidtar någon lämplig åtgärd om den inte gör det. Det vill säga det finns ingen automatisk kontroll som, om en applikation inte slutförs, försöker köra den på nytt. Det som finns är dels en check för att se vad som hände och vid fel ange en felkod och dels en kontroll för datatillgången, alltså om applikationen får tillgång till den data den behöver. Om den inte får det så försöker mellanprogramvaran ett par gånger igen och försöker även testa alternativa dataförvaringsställen. Swegrid har medvetet valt att inte hantera robustheten automatiskt, utan lämnar det till användaren för hur de vill göra om deras applikation inte slutförs. Som en Swegridanknuten påpekar:

> *Användaren får hantera det. Alltså om applikationen dör så får du bestämma om du vill starta om. Jag menar om det t.ex. är så att det är ett programeringsfel och jobbet bara kraschar, vad är då egentligen meningen med att starta om det om det kommer att krascha igen? Det säkraste egentligen är ju att överlåta det till användaren att bestämma.*

Enda sättet att få sitt jobb gjort i fall som detta är att manuellt skicka in jobbet igen, detta skulle förstås innebära att man får sitta och övervaka jobben medan de kör om man vill försäkra sig om att de blir gjorda. En bieffekt är tyvärr att jobbet hamnar längst bak i kön på den noden som du blir tilldelad nästa gång. Det finns dock planer hos Swegrid på att försöka lösa det här automatiskt:

> *Det är väll en av de grejerna som man håller på och jobbar med, att det skall finnas någon form av övervakningsdemon på sin egen sida som kan hantera det i själva klientdelen. Det skall finnas en funktion som skall kunna skicka jobben igen om det har fallerat.*
>
> *Man har en högre grad jobb som fallerar just nu än vad man har om man kör på ett dedikerat kluster så man kanske måste ta in det i beräkningarna.*

Entropia å sin sida har en lösning som är mer eller mindre automatiserad. Där startas jobbet om på en annan nod om noden går ner eller om noden inte återkommer med ett svar inom en viss förväntad tid. Om jobbet inte lyckas slutföras inom ett visst antal försök så markeras det som felaktigt och skickas tillbaka till mellanprogramvaran. Mellanprogramvaran kan då välja att antingen skicka in jobbet igen eller att underrätta användaren.

4.5 Säkerhet/integritet

Studien har undersökt två aspekter på säkerhet vad gäller DM. Den ena aspekten handlar om säkerheten hos datakommunikationen. Den andra aspekten syftar till säkerheten på noderna vad gäller integriteten hos informationen som bearbetas där. Den tredje och sista aspekten som tas upp i teorin bekymrar sig inte om DM-datan utan gridets säkerhet i sig vad gäller dess noder och eventuell påverkan av

dessa från illasinnade jobb och har därför ingen direkt påverkan på DM-datan. Därför har vi inte undersökt denna aspekt så ingående som de första två.

För att tillgodose säkerheten i ett gridsystem innehåller Globus toolkit paketet Grid Security Infrastructure (GSI), ett paket för säker nätverkskommunikation [47]. De flesta system förefaller använda sig av GSI för sina säkerhetsbehov [18]. Swegrid ger ett par exempel på hur man kan säkerställa sin data när den skickas:

> *Man kan ju t.ex. hämta från valfri URL och då kan man specificera att den skall vara https. Sen har du GridFTP. Den har möjlighet att kryptera GridFTP överföringen.*

När man skickar in ett jobb så skickar man med ett certifikat som ger jobbet rättigheter till åtkomst av filer som man behöver via GridFTP. På det sättet är filerna skyddade från åtkomst. Däremot är det inte standard att kryptera själva filöverföringen på Swegrid så den är vanligtvis öppen. Man resonerar så här på Swegrid:

> *Autentiseringen sker ju med kryptomekanismer så att säga, motsvarande inloggningen sker ju inte med klartextlösenord eller så, däremot när själva dataöverföringen av filerna sker, så sker den i klartext. Normalt sätt, det är konfigurerbart av administratören som sätter FTP-servern, men normal sätt så har man inte krypteringen påslagen för att det tar för mycket cpu-kraft att kryptera de här stora dataströmmarna.*

När det gäller den andra aspekten av säkerhet, den som har med säkerheten hos data när den ligger på en nod så finns det inga funktioner för det hos Swegrid. Ett jobb på en nod kan se ett annat jobbs filer och läsa dem. Dock är det något man har tänkt på att kanske göra något åt. En person på Swegrid menade följande:

> *Det är också en annan grej som håller på att diskuteras just det här med att hur man kör grid-jobben, som det är nu så är det lite problematiskt för att man mappar ju många användare till ett unixaccount på varje maskin, det gör ju i princip att ett jobb kan se ett annat jobbs filer också och läsa dom och så vidare, det är ju något vi håller på att fundera på om man kan göra på något annat sätt.*

Förutom att ett jobb ser ett annat jobbs filer och kan läsa dessa säger man på Swegrid att administratören på noden förstås har fulla rättigheter att göra vad han vill med jobbens data något som man på Swegrid också ser som ett problem vid ett eventuellt användande av ett scavenging grid.

Det finns dock försök till lösningar på säkerhetsfrågan vad gäller data på noder. Forskning för att säkerställa att jobb och noder inte kan skada varandra bedrivs bl.a. av Entropia [6]. Entropias lösning bygger på s.k. sandlådelösning, som går ut på att man kapslar in jobbet så att det interagerar med en virtuell maskin istället

för direkt med noden. Detta hindrar både att jobbet kan påverka data och resurser hos noden och att noden påverkar jobbet eller dess data på oönskat vis.

4.6 Enkel tillgång till data

Eftersom vi valt att koncentrera vår undersökning på de gridtyper som syftar till att tillhandahålla processorkraft har undersökningarna rörande dataförsörjning koncentrerats till om computational grids sätter några begränsingar på hur en applikations dataförsörjning kan skötas eller om den på något sätt bidrar till dataförsörjningen, snarare än att se till verktyg för t.ex. datatvätt. Eftersom Swegrid enbart är ett computational grid överlåter det dataåtkomsten på den som konstruerar applikationen. För detta tillhandahåller man diverse protokoll, t.ex. GridFTP [46] och NFS. När man skickar in jobbet så behöver man inte själv hantera filöverföringen. Man anger helt enkelt i ett jobbscript var datan man behöver befinner sig. Inte heller behöver man sköta utdatafilerna manuellt, inte förrän man behöver dem i alla fall. På Swegrid ger man följande förklaring:

> *I sin XRSL-fil, det är den som specificerar jobbet, så får användaren ange vilka indatafiler han behöver och kan då peka ut dom antingen som filer som finns på hans klientmaskin eller som finns lagrade någonstans på en GridFTP-server och då kommer dom indatafilerna automatiskt att laddas ner till den maskin som jobbet skall köras på och kommer då finnas tillgängliga då jobbet startar. Och på samma sätt får han specificera vilka utdatafiler som kommer att genereras av jobbet och kan om han vill automatiskt få dom uppladdade till en GridFTP-server.*

Ett annat stöd hos Swegrid när det gäller applikationernas tillgång till data är en cachningsfunktion:

> *…t.ex. om flera jobb specificerar samma indatafiler laddas det bara ner en gång och ligger cachade på lokaldisken…*

På det sättet har de andra jobben enkel tillgång till samma data utan att behöva vänta på att den skall laddas ner igen.

5 DISKUSSION

I kapitlet börjar vi med att diskutera hur GC tillfredställer de sex krav DM ställer på ett system som det skall köras på genom att analysera det som framkom i resultatkapitlet. Därefter har vi en generell diskussion över hur väl vi anser att GC i nuläget lämpar sig som plattform när man skall genomföra DM för att därefter avsluta med att besvara vår frågeställning.

5.1 Processorkraft

Om vi ser till möjligheten till processorkraft så spelar det föga roll vad du har för noder om du inte kan få den data du tänker bearbeta att hamna på noderna du vill utnyttja. I resultaten kom det upp att det trots allt handlar om väldigt stora datamängder som skall förflyttas från det ena stället till det andra. Swegrid har tillgång till Sunets GigaSunet-nätverk och har därför mindre problem med detta men hur ser det ut på företagen som skall använda sig av gridteknologin för sin DM? Detta är något som man inte kan ignorera utan måste se till att det finns de resurser som krävs för att inte man skall gå miste om fördelarna med GC.

Skalbarheten på gridet tycks vara någorlunda bra efter vad vi har fått höra. Men om du vill åt mer processorkraft genom att addera noder så måste du räkna med att få bygga ut presentationslagret som bl.a. handhar mellanprogramvaran. Vilken kapacitet på presentationslagret som krävs i förhållande till antal noder har dock flera beroendefaktorer. Bl.a. så beror det på hur många anrop det får ta emot. Det fanns i ett tidigt stadium problem med att Swegrids presentationslager fick så många anrop med förfrågningar på systemets status att det inte klarade av att sköta att ta emot jobb. Sen tar presentationslagret också hand om nedladdningen av filerna som jobben har ansökt om vilket också skapar stress på presentationslagret.

Det är fortfarande inte helt trivialt att utnyttja processorkraften även om du löst ovanstående problematik. Det handlar naturligtvis också om hur väl din applikation lämpar sig att köras på ett grid. Som det nämndes i resultatet så är det fördelaktigt om applikationen eller uppgiften du vill köra kan delas upp i mindre, oberoende jobb. Som exempel på en sådan uppgift i DM har vi träningen av neurala nätverk där man måste starta om träningen med olika utgångslägen på parametrarna i nätverket för att få ett sådant optimalt nätverk som möjligt. Detta skulle enkelt kunna köras som många småjobb där varje jobb var ett nätverk med en viss parameteruppsättning som skulle tränas.

Kan man däremot inte dela upp en uppgift försvinner många av fördelarna med GC då systemet – i bästa fall - reduceras till ett sätt att sköta åtkomsten av enskilda noder som råkar vara kraftfulla nog att klara av uppgiften i fråga, snarare än ett sätt att aggregera datorkraft ur en större mängd noder. Detta är förvisso inte samma sak som att GC är olämpligt för dylika uppgifter – Swegrid är ju t.ex.

ämnat att just agera samlad presentationslager för forskare som behöver använda sig av kluster.

5.2 Processorkraft på begäran

Huruvida grid kan tillhandahålla processorkraft på begäran är en svår fråga att ge ett generellt svar på. Bland de faktorer som spelar in finns i hur hög grad andra applikationer används på gridet, gridets kapacitet, hur många noder som kan tillfredsställa jobbets - eller dess subjobbs - krav samt hur effektivt kösystemet är. Något som saknas just nu och som nämndes i resultatet är möjligheten att migrera jobb. Detta skulle öka möjligheterna till att få tillgång till processorkraft om det blir tillgängligt någonstans i gridet medan ett jobb står i en lång kö till en annan nod som just vid tillfället som jobbet kom in verkade vara det bästa alternativet.

Under de omständigheter som nämns under processorkraft på begäran i avsnitt 2.1.2 skulle det kunna uppstå problem i ett system liknande Swegrid, där stora beräkningar hanteras enligt "först till kvarn", eftersom det inte finns något mer dynamiskt prioriteringssystem. Om användaren behöver något klart om 15 minuter så är det nästintill omöjligt i nuläget att förvissa sig om att det kommer att gå att genomföra. Användaren får helt enkelt ställa sig i kö som alla andra. Här tycks Entropia ha löst det ganska smidigt genom sitt kösystem med innehållande prioriteringssystem där varje jobb har en prioritet.

En annan sak som kom upp i resultatet var det här med att kunna begära att gridet tillgodoser vissa resurskrav som ett jobb har genom XRSL-scriptet. Ett problem som vi anser kommer upp då är begränsandet av noder för andra jobb. Om systemet bara ser till minimikraven och sen tar första bästa nod som leder till kortast exekvering av jobbet så finns det här en potentiell risk, förutsatt att noderna är heterogena, att låsa resurser för jobb som behöver dessa resurser för att överhuvudtaget kunna exekveras. Om du har ett jobb som kräver 1 GB minne och detta tilldelas en nod med 4 GB minne för att denna nod kommer att exekvera jobbet först och det sedan kommer ett jobb som behöver 4 GB så har du nu tillsatt ett jobb på en nod med 4 GB minne som nästa jobb skulle behöva. Det blir förstås en fråga om balansering här. Vad bör prioriteras, att jobb görs så fort som möjligt i den ordning de tillkommer eller att få den genomsnittliga jobbexekveringen så bra som möjligt?

För att kunna vara riktigt säker på att användaren kan få tillgång till processorkraft när den behövs krävs det dock trots allt tillräckliga beräkningsresurser för att klara av den belastning som råder på gridet. Förutsatt att hårdvaruresurserna är tillräckliga och att man finner lämpliga lösningar på ovanstående problematik bör processorkraft på begäran inte vara omöjligt att uppnå inom GC.

5.3 Stora mängder minne

Tillfredställandet av mycket minne får man egentligen inte hjälp med i och med GC-införande. Det som resultatet visar är att du kan tillfredställa minneskravet genom att GC letar upp en nod med de minnesresurser som du behöver men finns det inte en nod i gridet som uppfyller kraven så är det inte något mer du kan göra. Vad minneskravet hos DM däremot gör är att påverka de andra kravens problem. Tittar man t.ex. på det exempel som nämns i avsnitt 5.2 så är ju minneskravet en begränsande faktor till hur många noder du har möjlighet att utnyttja i ditt grid och därmed hur mycket processorkraft du kan begära vid ett visst tillfälle för just den applikationen som har minneskravet. Även i de fall där en uppgift går att dela upp i subjobb ger inte det någon fördel åt grid, vad gäller minneskrav. Även om en uppdelning kan resultera i lägre minneskrav för varje enskilt jobb är slutresultatet enbart förbättrade förutsättningar att tillvarata processorkraften i gridet, då subjobben, förhoppningsvis, får tillgång till ett större antal noder som lämpar sig för uppgiften. Ur ren minnessynpunkt ger inte grid några fördelar jämfört med att exekvera uppgiftens subjobb ett efter ett på en enda dator.

5.4 Robusthet

Om man ser på grid som att man får tillgång till massor med datorer som skulle kunna köra jobben man har, kan man kanske tycka att det skulle garantera jobbens utförande. Riktigt så enkelt är det emellertid inte. Ett system som fungerar som t.ex. Swegrid, där jobben dör ut utan omstart om de får problem, antingen med data som de behöver men inte får eller att noden går ner, har en väldigt liten grad av robusthet. Gridet bidrar i det här fallet inte alls till att lösa kravet på robusthet som vissa DM-tillämpningar kräver. Liksom minneskravet var kopplat till processorkraft på begäran ovan, är robustheten kopplad till dataförsörjningen. I och med att jobb dör om de inte får den data de behöver så finns det en stark koppling däremellan.

Å andra sidan har vi den lösning som Entropia föreslår, där jobbet startas om ett visst antal gånger om det av någon anledning uppstår komplikationer. Vi ser även ett stort problem med detta, något som också uppmärksammades i resultatet, nämligen det att du inte vet varför jobbet kraschade. Var det p.g.a. att jobbet inte fick sin data, att det var något fel på noden eller att jobbet helt enkelt inte var programmerat riktigt? Vi ser i värsta fall ett scenario där det visar sig att jobbet är felprogrammerat, väldigt stort och att Entropiasystemet vill köra om det ett antal gånger för att kontrollera att det verkligen inte kan gå igenom. Om man nu tänker sig att det inte blir fel förrän absolut i slutet, så finns det en potentiell risk för att mycket tid och resurser slösas på något som ändå i slutändan inte kommer att fullföljas. Detta vore högst olämpligt. Vi ser här ett starkt behov av att man utvecklar bättre feldiagnostik. Är det gridet som fallerat så kan man tänka sig att köra om jobbet på en annan nod men om det inte är det så borde man kanske följa Swegrids modell där det är användarsidan som bestämmer hur man skall hantera det icke slutförda jobbet.

Skulle man välja att köra ett scavenging grid så ser vi ett annat potentiellt problem vad gäller robusthetsproblematiken. Om gridanvändaren har ett jobb som exekveras på en nod som är en arbetsstation någonstans som det sitter en användare på, så finns det risk att gridanvändarens jobb helt plötsligt måste migreras till, eller startas om på, en annan nod. Eftersom Entropia garanterar diskretion så innebär det att användaren som sitter på noden när som helst kan få för sig att öppna upp en resurskrävande applikation på sin arbetsstation, en applikation som behöver nodens resterande resurser – dvs. samma resurser som gridjobbet tilldelats och som därmed leder till att gridjobbet slängs ut. Ju större jobbet är, ju mer resurser tar det upp på en nod och ju större chans är det att användaren gör något som kräver tillräckligt mycket resurser för att jobbet måste bort.

5.5 Säkerhet/integritet

Säkerheten delades in i tre delar i teorin. Dels har vi säkerheten när data skall överföras över nätverket och dels har vi två aspekter av säkerhet när data har hamnat på en nod och där bearbetas, säkerheten av datan och säkerheten av noden. Huruvida noden är skyddad från DM-jobbet är egentligen inte intressant för just datasäkerhetskravet som finns hos DM. De andra två aspekterna är dock av stort intresse eftersom det är datan som DM är intresserad av att skydda, och nyttan med att skydda en applikations integritet är såväl universell som uppenbar.

I resultatet framkom det två möjligheter till att säkerställa datan när det är fråga om närverkskommunikation. Det ena var gridspecifika standarder, som GridFTP, och det andra var mer universella standarder, som https, och båda begär man som användare när man genererar ett XRSL-script, eller motsvarande, för ett jobb. Problemet med krypterad dataöverföring är att det tar beräkningsresurser i anspråk, som annars hade kunnat användas till att utföra det aktuella jobbet .

När det gäller säkerheten på noderna – för jobb, såväl som nod, förefaller Entropias lösning med att kapsla in jobben och exekvera dem med hjälp av en virtuell maskin vara lämplig – även om virtuella maskiner i allmänhet även de gör att en applikation blir mer resurskrävande.
Att inte ha någon lösning alls kanske går an om man litar på alla som är kopplade till gridet och dess noder men i ett företag eller annan verksamhet där det kan finnas data som man inte vill att vem som helst skall kunna få reda på är detta helt enkelt inte realistiskt. Nu är Swegrid fortfarande i utvecklingsstadiet och man hade som det visade sig i resultatet tänkt på möjligheten att faktiskt säkerställa data även när den väl har kommit fram till noderna. Hur denna lösning kommer att se ut är bara att vänta och se.

Eftersom scavenging grids bygger på resurser där varje nod, i princip, har en unik ägare blir säkerhet extra viktigt då sannolikheten att någon av dessa av någon anledning föresätter sig att kompromettera ett jobbs integritet ökar. Detta gör att man teoretiskt sätt kan tvingas använda så stor del av de resurser gridet får

tillgång till på varje nod för att säkerställa säkerheten att det inte blir tillräckligt mycket över för att faktiskt genomföra jobbet inom rimlig tid.

Den tredje aspekten av gridsäkerhet, den som har med nodens säkerhet gentemot illasinnade jobb att göra, har inte någon direkt bäring på den data vi är intresserad av att skydda. Dock skall det påpekas att det finns anledning, tack vare det nuvarande upplägget på robusthet, att säkerställa även denna säkerhetsaspekt eftersom det i nuläget fungerar så att om noden sänks av ett illasinnade jobb och ett DM-jobb också körs där försvinner det DM-jobbet utan någon omstart eller migrering.

5.6 Enkel tillgång till data

GC har som vi såg i resultatet ett antal funktioner för att underlätta dataåtkomst för de jobb som skall köras på gridet. Om man först tittar på XRSL-scriptet så är det ganska enkelt att se till att man får den datan som man behöver genom att helt enkelt ange var den ligger någonstans, sedan sköter gridet hämtningen av datan själv. Detta är ett utmärkt stöd för DM när man har data som ligger utstrött lite var stans eftersom gridet då ganska enkelt kan samla ihop datan utan att du behöver göra mer än att ange alla de ställen som den finns på.

Det andra uppenbara stöd för DM som vi ser är cachningsfunktionen hos gridet. Genom att du bara behöver ladda ner datan en gång så har jobben omedelbar åtkomst till datan och det blir därför kortare arbetstider på jobben. Vi ser också en omedelbar användning för cachningsfunktionen vad gäller en specifik inriktning av DM, nämligen neurala nätverk och träningen av dessa. Eftersom neurala nätverk tränas genom att man startar sitt nätverk med olika utgångslägen på startvärdena i nätverkat men man använder samma data i alla träningstillfällen så har även här cachningsfunktionen en tidssparande effekt, speciellt om man lägger ut alla träningarna så man kan köra dem samtidigt. Cachningsfunktionen gör även att det blir mindre stress på nätverket med tanke på att data bara behöver skickas en gång.

Ett problem som vi ser i nuläget är ett liknande problem som vi hade på säkerhetsavsnittet, nämligen ett problem som är kopplat till den nuvarande robusthetessituationen. Om man i sitt XRSL-script anger ett ställe som gridet skall hämta data ifrån som för tillfället är nere så dör applikationen precis som i säkerhetsexemplet ovan.

5.7 Generell diskussion

Under våra efterforskningar har det framkommit flera intressanta punkter som inte hamnar direkt under någon av de övriga punkterna vi gått igenom ovan.

Applikationer som hämtar extern data under körning riskerar att slösa ansenliga mängder processorkraft eftersom jobbet står stilla under den tid det tar för begäran att göra en round-trip – dvs. begära och få den data som behövs.

Med dålig eller ingen inriktning på parallellism så kan man ändå använda gridet till att göra t.ex. saker som arbetar med genomströmning av data, t.ex. kreditkortsbedrägeri och tillverkningsövervakningen/kontrollen. Men frågan är hur lämpat GC är för tidskritiska uppgifter som skall vara kontinuerligt aktiva. I ett grid där flera applikationer samsas kan det mycket väl hända att ett jobb fastnar i en kö – oavsett hur högt prioriterat det är.

Något man måste ta med i beräkningen är att processorkraft på begäran kan bli väldigt svårt att tillgodose och att uppehålla. Med tanke på den nuvarande dåliga robustheten blir det inte bara en fråga om att lyckas få processorkraften när man vill ha den utan också att lyckas behålla den när man väl har fått den, något som försvåras av att det, om jobbet skulle dö, inte finns något prioriteringssystem för jobb. Så bara för att du har lyckas få din processorkraft när du vill ha den en gång innebär inte det att du får den igen om något går fel.

En annan synpunkt vi har är att man i nuläget som användare oftast behöver veta hur man skriver ett XRSL-script för sitt jobb, något som vi anser att man måste ändra på innan man kan tillämpa det här i företag där man inte kan förvänta sig att de anställda har förmågan eller viljan att lära sig detta. Man måste förmodligen göra ett webbgränssnitt för de applikationer som finns på ett företag så att de går att köra utan att användaren behöver veta om vad ett XRSL-script är, något som man på Swegrid redan har tillämpat för vissa applikationer.

5.8 Grid computings förmåga att tillgodose kraven

I kapitel 5 har vi nu diskuterat hur GC förhåller sig till de krav som DM ställer på systemet det skall köras på och vi har även diskuterat lite intressanta punkter som kom upp i samband med huvuddiskussionen. Nedan följer de slutsatser vi har dragit av materialet i resultatet och diskussionen och dessa slutsatser skall därmed utgöra svaret på vår frågeställning.

Vi anser att GC skulle kunna vara en utmärkt plattform för ett flertal former av DM, under förutsättning att vissa brister åtgärdas. Redan idag kan GC uppfylla kraven på processorkraft och enkel tillgång till data, förutsatt att infrastrukturen är adekvat och applikationerna gridanpassats tillfredsställande. Genom att ha tillgång till en stor mängd datorer som besitter en stor mängd processorkraft så kan man potentiellt påskynda sina applikationers slutförande och kanske sluta dra sig för att köra applikationerna pga. tidsåtgången eftersom denna inte skulle vara så stor längre. Applikationerna som kräver data från olika ställen blir lätta att köra pga. den smarta lösningen som återfinns i XRSL-scriptet där man enkelt kan ange var datan man behöver finns. Säkerheten för DM finns det i stort sett möjlighet att tillgodose, förutsatt att man är beredd att offra en del resurser på kryptering och andra former av säkerhetsåtgärder. Dock bör man tänka extra mycket på

resurskraven om man har funderingar på att sätta upp ett scavenging grid eftersom det mycket väl kan sluta med att man använder större delen av kraften som gridet innehar till att säkerställa data.

Ett annat krav som har potential att tillgodoses är processorkraft på begäran. Vi skulle vilja se förbättrade kösystem med en migreringsmöjlighet också för att kunna flytta på jobb om noder blir lediga eller om man lägger till noder. Frågan är hur man gör en sådan genomgång av jobben och förflyttningen av dem att det inte tar mer kraft än vad du tjänar. Förutom detta tillägg tycks Entropias lösning vara på god väg att vara en lösning till processorkraft på begäran-problematiken.

Tillgodoseendet av en stor mängd minne är inte något som gridet i sig egentligen kan bidra med. Det är snarare så att man kan få hjälp med att hitta en nod i gridet som kanske har den mängden man behöver. Dock ställer tillgodoseendet av minne, som poängterats ovan, till med lite problem vad gäller processorkraft på begäran. Därmed blir man, om uppgiften kräver både processorkraft på begäran och mycket minne, tvungen att basera sin infrastruktur på noder med stora mängder primärminne. Om robusthet är en kritisk faktor för det användningsområde som man har i åtanke så skall man nog titta på andra lösningar än just GC - i alla fall i nuläget. Varken Swegrids eller Entropias lösning på ett jobbs fallerande är särskilt tillfredsställande. Swegrids för att det innebär att ditt jobb bara dör och du måste själv se till att få det omstartat och då hamnar du, vid en eventuell kö, sist i kön. Entropias för att det kan potentiellt stå och köra om ett väldigt stort och resurskrävande jobb flera gånger trots att det är ett ordentligt fel på jobbet. Entropia har visserligen sin prioriteringsökning på fallerade jobb så att om det skulle vara något fel på t.ex. noden man kör på så kan man få jobbet utfört utan att förlora tid på att vara tvungen att ställa sig längst bak i kön. Vi anser däremot att gridet inte kan lastas för att en applikation genererar ett felaktigt jobb och att Entropia därmed har den bättre robustheteslösningen i sammanhanget om än inte perfekt. Det skall också påpekas att man nog inte heller skall använda sig av ett scavenging grid om robusthet är viktigt då noden du har ditt jobb på när som helst kan belastas så hårt av ägaren till noden och ditt jobb måste flyttas för att det inte skall påverka ägarens arbete.

De begränsande faktorerna som vi upptäckt kommer att påverka gridet och dess möjlighet att tillgodose kraven som DM ställer är; nätet som noderna i gridet är ihopkopplade med, nätet som leder in till gridet, presentationslagrets infrastruktur, alltså vilka servrar som finns där för att sköta jobbhanteringen, samt vad det är för datorer som utgör gridets noder. Dessa begränsningar är rena hårdvarubegränsningar och inget man direkt kan göra något åt annat än att byta ut och uppgradera dem. I mjukvaruväg är det utformningen av kösystemet, migreringsmöjligheterna hos noderna och hur man hanterar fallerade jobb som begränsar. Lättaste sättet att misslyckas med ett grid tycks därmed vara att ha ett inadekvat presentationslager, både hårdvaru- och mjukvarumässigt.

6 SLUTSATS

I slutsatsen ges en kort redogörelse för fördelarna med GC, samt potentiella användningsområden inom DM för GC. Därefter ges förslag på områden för framtida studier.

Som det sades i diskussionen anser vi att GC mycket väl kan tjäna som plattform för ett stort antal användningsområden inom DM. Dock kan man inte generellt säga att GC per definition är ett bra val, utan man måste noga se över vilka krav som kommer ställas av just den uppgift applikationen skall utföra. En av fördelarna med GC är att det är avsett att fungera med hårdvaruresurser som är både dynamiska och heterogena, vilket gör att det kan lämpa sig för uppgifter vars krav skiftar på ett sätt som är svårt att förutse i ett systems designfas. Om t.ex. ett kluster visar sig vara otillräckligt dimensionerat för sin uppgift måste man uppgradera hela klustret, medan ett grid enkelt kan byggas ut genom att utöka det med ett antal kraftfullare noder, eller uppgradera ett antal befintliga noder.

De DM-områden/-applikationer som man skulle kunna tänka sig att GC skulle kunna bidra till är bl.a., som vi redan nämnt, träning av neurala nätverk men också mönsterigenkänning och dylika områden. Vi ser dock att man nog inte skall använda sig av GC om man har applikationer som kräver mycket respons från t.ex. en användare, s.k. fråga-svar applikationer, då man inte utnyttjar sig av gridet på ett optimalt sätt. Tack vare att robustheten inte är garanterad skall man också undvika applikationer som körs kontinuerligt, t.ex. produktionsövervakning.

Ett faktum vi observerat, och som inte får glömmas bort, är att GC på intet sätt är en teknologi som är färdigutvecklad. På många områden krävs det fortfarande forskning och utveckling innan GC kan sägas ha uppnått ett tillräckligt mått av fullständighet. I dagsläget är, som tidigare nämnts, inte ens terminologi och definitioner enhetliga.

Framtida studier

Något som vi känner skulle kunna vara intressant är en mer genomgående kartläggning över vilka specifika DM-områden/applikationer som skulle kunna ha nytta av GC annat än de som redovisats i den här studien. Man skulle även kunna göra en motsvarande studie som inriktar sig på data grids istället för computational grids för att se om även detta GC-område skulle kunna vara till någon nytta för DM.

7 METOD- OCH KÄLLKRITIK

I detta kapitel utvärderar vi vårt tillvägagångssätt och diskuterar alternativa metoder.

Vi har haft för avsikt att undersöka huruvida det finns en möjlighet att GC skulle kunna tillgodose de krav som DM ställer på systemet som det skall köras på. Vi anser att den metod vi valt, med semistrukturerade intervjuer, egentligen är den enda kvalitativa metod hade varit möjlig att genomföra på ett produktivt sätt. Intervjuerna har även visat sig nödvändiga för att få fram den specifika information vi sökt, då det som publiceras i artiklar och på hemsidor sällan ger utttryckliga svar på just det vi ansett oss behöva veta.

Det enda alternativa tillvägagångssättet vi kan se är att vi skulle ha kunnat sätta upp ett eget grid och testat att genomföra en DM-session på det. Detta hade dock krävt både resurser vi inte har tillgång till och teknologi som ännu inte är fullt utvecklad för att kunna ge ett användbart svar.

Något som vi hade en viss skepsis till var Entropia-artikeln eftersom det är ett kommersiellt företag som naturligtvis vill sälja sin idé. Den är dock publicerad i "Journal of Parallel and Distributed Computing" [50] och måste därför ha granskats av forskare inom området som artikeln gäller.

8 REFERENSER

Nedan följer de referenser som vi har använt oss av i vår studie.

Böcker och artiklar

[1] Adabala S., Butt A.R., Figueiredo R.J., Fortes J.A.B. & Kapadia N.H (2003). Grid-computing portals and security issues, *Journal of Parallel and Distributed Computing*, 63, 1006-1014.

[2] Aronson J.E. och Turban E. (2001) *Decision support systems and intelligent systems (6th ed.)*. Upper Saddle River: Prentice Hall.

[3] Ayache N., Pennec X. & Stefanescu R. (2004) Grid powered nonlinear image registration with locally adaptive regularization, *Medical Image Analysis*, 8, 325-342.

[4] Backman J. (1998) *Rapporter och uppsatser*. Lund: Studentlitteratur.

[5] Behnke J. & Dobinson E. (2000) NASA workshop on issues in the application of data mining to scientific data, *ACM SIGKDD Explorations Newsletter*, 2, 70-79.

[6] Bhatia K., Calder B., Chien A. & Elbert S. (2003) Entropia: arcitecture of an enterprise desktop grid system, *Journal of Parallel and Distributed Computing*, 63, 597-610.

[7] Brightwell R., Fisk L.A., Greenberg D.S, Hudson T., Levenhagen M, Maccabe A.B. & Riesen R. (2000) Massively parallel computing using commodity components, *Parallel Computing*, 26, 243-266.

[8] Casanova H. (2002) Distributed computing research issues in grid computing, *ACM SIGACT News*, 33, 50-70.

[9] Cannataro M., Talia D. & Trunfio P. (2002) Distributed data mining on the grid, *Future Generation Computer Systems*, 18, 1101-1112.

[10] Crawford J., Lim S.G. & Waters G. (2004) Optimising multicast structures for grid computing, *Computer Communications*, 27, 1389-1400.

[11] Cunha J.C., Medeiros P.D., Ranab O.F. (i tryck). Future trends in distributed applications and problem-solving environments, Future *Generation Computer Systems* – article in press.

[12] Daigle J.N. & Duan Q. (2004) Resource allocation for quality of service provision in multistage buffered crossbar switches, *Computer Networks*, 46, 147-168.

[13] Datta S., Dutta H., Kargupta H. & Sivakumar K. (2003) Analysis Of Privacy Preserving Random Perturbation Techniques: Further Explorations, *Workshop On Privacy In The Electronic Society archive - Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, 31-38.

[14] Di Martino V. & Mililotti M. (2004) Sub optimal scheduling in a grid using genetic algorithms, *Parallel computing*, 30, 553-565.

[15] Fayyad U.M., Piatetsky-Shapiro G. & Uthurusamy R. (2003) Summary from the KDD-03 panel: data mining: the next 10 years, *ACM SIGKDD Explorations Newsletter*, 5, 191-196.

[16] Fayyad U. & Stolorz P. (1997) Data mining and KDD: Promise and challenges, *Future generation computer systems*, 13, 99-115.

[17] Fayyad U. & Uthurusamy R. (2002) Evolving data mining into solutions for insights: Evolving data into mining solutions for insights, *Communications of the ACM*, 45, 28-31.

[18] Ferrari A.J., Grimshaw A.S., Humphrey M.A., Karpovich J.F., Lewis M.J., Morgan M.M., Natrajan A., Nguyen-Tuong A. & Wasson G.S. (2003) Support for extensibility and site autonomy in the Legion grid system object model, *Journal of Parallel and Distributed Computing*, 63, 525-538.

[19] Foster I., Gawor J. & von Laszewski G. (2000) CoG kits: a bridge between commodity distributed computing and high-performance grids, *Proceedings of the ACM 2000 conference on Java Grande*.

[20] Foster I., Fredian T., Greenwald M., Keahey K., McCune D., Peng Q., Schissel D. P. & Thompson M. (2002) Computational Grids in action: the National Fusion Collaboratory, *Future Generation Computer Systems*, 18, 1005-1015.

[21] Galis A., Guo X., Liu D., Yang B. & Yang K. (2003) Towards efficient resource on-demand in Grid Computing, ACM *SIGOPS Operating Systems Review*, 37, 37-43.

[22] Goethals B. (2004) Memory issues in frequent itemset mining, *2004 ACM Symposium on Applied Computing*, 530-534.

[23] Hand D., Mannila H. & Smyth P. (2001) *Principles of data mining*. Cambridge: The MIT press.

[24] Holsheimer M. (1999) Data mining by business users: integrating data mining in business processes, *Conference on Knowledge Discovery in Data - Tutorial notes of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 266-291.

[25] Iyer N. (1996) SAP R/3 — Fraud monitoring and data-mining techniques, *Computer Audit Update*, 1996, 23-25.

[26] Jiao J. & Zhang Y. (2005) Product portfolio identification based on association rule mining, *Computer-Aided Design*, 37, 149-172.

[27] Koh H.C. (2004) Going concern prediction using data mining techniques, *Managerial Auditing Journal*, 19, 462-476.

[28] Kohavi R. & Quinlan J.R. (2002) Decision-tree discovery. I Klösgen W. & Zytkow J.M. (red.) *Handbook of data mining and knowledge discovery* (pp. 267-276). Oxford: Oxford university press.

[29] Körner S. & Wahlgren L. (2000) *Statisktisk dataanalys (3:e uppl.)*. Lund: Studentlitteratur.

[30] Laforenza D. (2002). Grid programming: some indications where we are headed, *Parallel computing*, 28 1733-1752.

[31] Lau H.C.W., Jiang B., Lee, W.B. & Lau, K.H. (2001) Development of an intelligent data-mining system for a dispersed manufacturing network, *Expert Systems*, 18, 175-185.

[32] Leigh W., Hightower R. & Modani N. (2005) Forecasting the New York stock exchange composite index with past price and interest rate on condition of volume spike, *Expert Systems with Applications*, 28, 1-8.

[33] Luksch P. (2000) Parallel and distributed implementation of large industrial applications, *Future Generation Computer Systems*, 16, 649–663.

[34] Madigan D. & Ridgeway G. (2003) Bayesian data analysis. I Ye N. (Red.) *the Handbook of data mining* (pp. 365-391). Mahwah: Lawrence Erlbaum associates.

[35] Patel R., & Tebelius U. (1987) *Grundbok i forskningsmetodik : kvalitativt och kvantitativt*, Lund, Studentlitteratur.

[36] Seifert J.W. (I tryck) Data mining and the search for security: Challenges for connecting the dots and databases, *Government Information Quarterly* – article in press.

[37] Zhu A. (2004) Analysis of queueing policies in QoS switches, *Journal of Algorithms*, 53, 137-168.

WWW-dokument

[38] Podd K.C., The Drive for Quality, *SAS Success Stories* [www-dokument] http://www.sas.com/success/robertbosch.html [2004-11-12].

[39] Florentin T., Seeing Data More Clearly, *SAS Success Stories* [www-dokument] http://www.sas.com/success/essilor.html [2004-11-12].

[40] Foster I. (2002) What is the Grid – a Three Point Checklist, *Grid Today* [www-dokument] http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf [2004-02-23].

[41] Foster I., Roy A. & Sander V. (2000) A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation, *8th International Workshop on Quality of Service*, [www-dokument] http://www.globus.org/documentation/incoming/iwqos_adapt1.pdf [2004-07-21].

[42] Jacob B., (2003) Grid computing: What are the key components? [www-dokument] http://www-106.ibm.com/developerworks/library/gr-overview/ [2004-03-05].

[43] Palmerini P. (2002) High Performance Distributed Systems for Data Mining, *Univerit`a Ca' Foscari* [www-dokument] http://www.dsi.unive.it/~palmeri/doc/report.pdf [2004-03-24].

[44] Roberts M.L. (2003) Does the iSeries Play in Grid? It Depends on the Definition, *Iseries network* [www-dokument] http://www.iseriesnetwork.com/news/nwn/story.cfm?ID=16593&channel=news [2004-12-06].

[45] Shread P. (2002) Gateway Offers Computing On Demand, *Grid Computing Planet* [www-dokument] http://www.gridcomputingplanet.com/news/article.php/3281_1555061 [04-12-01].

[46] The Globus Alliance: GridFTP: Key Concepts [www-dokument] http://www-unix.globus.org/toolkit/docs/3.2/gridftp/key/index.html [2004-11-25].

[47] The Globus Alliance: GSI: Key Concepts [www-dokument] http://www-unix.globus.org/toolkit/docs/3.2/gsi/key/index.html [2004-07-15].

[48] SAS Process Intelligence for Manufacturing [www-dokument] http://www.sas.com/industry/mfg/pri/index.html [2004-11-12].

WWW-adresser till organisationer

[49] http://www.cogkit.org/ [2004-09-15].

[50] http://www.elsevier.com/

[51] http://www.gridalliance.org [2004-02-25].

[52] http://www.globus.org [2004-02-25].

[53] http://www.ibm.com [2004-03-24].

[54] http://www.sas.com [2004-03-24].

[55] http://www.swegrid.se [2004-03-10].

APPENDIX 1 - INTERVJU MED REPRESENTANTER FÖR SWEGRID

Här presenteras de frågor som ställdes via telefonintervju, samt i ett fall ansikte mot ansikte, till Swegridrepresentanterna.

Frågorna som de såg ut vid sista intervjutillfället:

1. Vad är swegrid designat att göra?

2. Är det lätt att lägga till och ta bort noder?

3. Har det gjorts något för att underlätta för användaren att gridanpassa vederbörandes applikation? t.ex. verktyg för att generera xRSL-script.

4. Vi noterade att det på hemsidan för swegrid står att swegrid inte är byggt för parallellism. Vi undrar vad det får för konsekvenser för applikationer i tort? Kan middlewaret dela upp ett jobb i subjobb eller måste man dela upp det själv och submitta subjobben som en massa egna jobb? T.ex. om du har en applikation som är väldigt parallelliserbar, innebär det då att den inte går att dela upp på ett enkelt sätt som det är tänkt att den skall kunna?

5. Hur hanteras dataåtkomsten? Finns det inbyggda funktioner i gridet, eller får användaren hantera det helt på egen hand?

6. När vi intervjuade en av de andra ansvariga för swegrid fick vi uppfattningen att det inte var lämpligt att ha applikationer vars data skickades i "klumpar", utan att swegrid var anpassat för dataströmmar. Vad innebär det här exakt eller förstod vi fel? Vilka begränsningar sätter det här på applikationer?

7. Har gridet funktioner för kryptering och annan säkerhetshantering, eller får användaren sköta dylikt själv?

8. Är data som ligger på en nod skyddade från åtkomst?

9. Hur är det med loadbalancing på swegrid? Är det så att middlewaret försöker dela ut arbete så jämt som möjligt eller tar man alltid första nod man får tag på?

10. Finns det också så att middlewaret kollar efter minimikrav och delar ut på det sättet så att inte applikationer som inte tar så mycket resurser får mycket resurser så att det sedan kommer applikationer som verkligen behöver mycket resurser och då inte kan få det?

11. Hur hanterar gridet problem som upstår vid exekvering av en applikation?

12. Är gridet skyddat från eventuella skadliga applikationer? Hur stora rättigheter har användaren?

13. Hur bra är swegrid på att ta tillvara på så mycket processorkraft som möjligt? Har ni t.ex. kört någon form av benchmark för att se hur mycket processorkraft ni kan få ut?

14. Hur stor är den genomsnittliga belastningen på gridet?

15. Är möjligheten till datorkraften åtkomlig jämt?

16. Hur mycket påverkas systemet av hur många noder som ingår?

17. Sen undrar vi också varför swegrid överhuvudtaget är ett computing grid och inte ett scavenging grid med tanke på hur många datorer som universiteten har som oftast bara står där och inte gör så mycket, absolut inte så mycket att all deras kapacitet är uttnyttjad?

18. När filer som mer än ett jobb använder cachas - var cachas de?

APPENDIX 2 - INTERVJU MED REPRESENTANT FÖR SAS

Här presenteras de frågor som ställdes via e-post till en representant för SAS Institute.

Första omgången frågor:

1. På vilket sätt - och i hur stor utsträckning - är era applikationer grid-aware?

2. Fungerar era applikationer på alla standardiserade grid?

3. Vilka egenskaper hos griden är det applikationerna kan utnyttja?

4. Kan man uppfatta en markant skillnad om man använder sig av ett grid när man kör applikationen?

5. Har ni stött på några grid-specifika problem vid implementation? Om så, vilka?

6. I hur stor utsträckning är era dataminingapplikationer skalbara? När slutar storleken på gridet att spela roll för applikationens prestanda? Eller är det t.o.m. ett problem när gridet blir för stort?

7. Har det funnits ett intresse hos kunderna att applikationerna är grid-aware eller är det SAS som tagit initiativet?

8. Finns det några speciella kännetecken för vilken sorts företag som väljer att köra era applikationer på ett gridsystem i förhållande till de som väljer en traditionell lösning?

Andra omgången frågor:

9. När ni pratar om computing grids menar ni då att de datorer som applikationen körs på är dedikerade till ändamålet eller innefattar det även scavenging grids (cycle harvesting)?

10. Om det enbart handlar om dedikerade grids; finns det någon speciell anledning till detta?

Tredje omgången frågor:

11. Bygger ni/kunderna ett nytt grid varje gång ni/de vill utföra en datamininguppgift?

12. Använder ni er av globus toolkit och om ja, i hur stor utstäckning?

APPENDIX 3 - INTERVJU MED REPRESENTANT FÖR IBM

Här presenteras de frågor som ställdes via e-post till en representant för IBM. Intervjun företogs på engelska, då de frågor representanten inte själv kunde svaret på vidarebefordrades till andra anställda inom IBM.

Första omgången frågor:

1. In what way(s) - and to what extent - is Intelligent Miner for Data grid-aware?

2. Which properties of the grid is Intelligent Miner for Data able to utilize?

3. Is there a substantial change in efficiency when running Intelligent Miner for Data on a grid?

4. Have there been any grid-specific problems during implementation? If so, what kind of problems?

5. To what extent is Intelligent Miner for Data scalable? At what point does increasing the size of the grid seize to add efficiency to the application? Can there even be problems associated with the grid being too large?

6. Was grid-enabling Intelligent Miner for Data something the customers requested, or did IBM do it on their own initiative?

7. Is there anything that distinguishes customers who choose to run Intelligent Miner for Data on a grid from those who opt for a more traditional solution?

8. Are there any customers in Sweden who have chosen to run Intelligent Miner for Data on a grid?

Andra omgången frågor:

9. Can you see any possible advantages in grid-enabling IM (for example faster response times or lower hardware investment cost)?

10. Do the advantages of grid computing, whichever they are that data mining can utilise, not outweigh the disadvantages of having to deal with the difficulties surrounding grid computing?