Tvorba počítačových clusterů pomocí Linuxu
Creation computer clusters using Linux

Bakalářská práce
Petr Ciml
Vedoucí práce: Mgr. Jiří Pech, Ph. D.
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
2008

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne

Poděkování

Rád bych poděkoval vedoucímu této práce, panu Mgr. Jiřímu Pechovi, Ph. D., za cenné rady týkající se nejen tématu, ale také úpravy práce. Dále pak děkuji zaměstnancům Výpočetního ústavu ekonomické fakulty Jihočeské univerzity za vypůjčení potřebného hardwarového vybavení, poskytnutí prostorů a další podporu.

Anotace

Mým cílem je, aby tato práce pomohla zájemci s návrhem a realizací clusteru. Od uvedení do problematiky a vysvětlení pojmů přes porovnání vhodnosti jednotlivých řešení mezi Linuxovými distribucemi i mimo ně, až ke konkrétnímu návodu, jak realizovat vybrané řešení.

Abstract

The aim of this work is to support the user hen planning and creating a computer cluster. This B.A thesis starts with an introduction of the problem and explanation of main terms. Then it compares variol possibiities using different Linux distributions and also some other systems. And finály, there are concrete instructions for realisation of the particular posibility.

Obsah

1 Úvod

Velké množství lidí na celém světě se snaží objevovat nové techniky vedoucí k dosažení většího výpočetního výkonu jejich systémů. Důvodů je mnoho, vědecké potřeby, potřeby modelování a simulování různých situací od relativně jednoduchého předvídání vývoje trhu až po předpovědi počasí, nebo simulace výbuchu atomové bomby.

Technologií, pomocí kterých lidé sestavují vysoce výkonné systémy, je také velmi mnoho.

Prvním způsobem je zdokonalování stávajících, na polovodičích založených počítačů, počínaje urychlováním výpočetních jednotek a zvyšování jejich počtu, změnami ve výrobní technologii, úpravami instrukčních sad, zvětšováni kapacity operační paměti, apod.

Druhým způsobem je hledání alternativ k polovodičovým technologiím (které postupně dosahují svých fyzikálních limitů). Na tomto poli lze sledovat pokusy o vytvoření kvantových nebo biologických počítačů.

Tyto technologické postupy jsou samozřejmě velmi žádoucí a potřebné. Bohužel jsou však také velmi nákladné a mají stále limitovaný maximální výkon. Pokud je třeba vyššího výkonu, než poskytují nejvýkonnější komponenty, je nutné jít jinou cestou.

Existuje tedy způsob, jehož pomocí je možné dosáhnout extrémně vysokého výkonu bez limitu? Odpověď zní: „Ano existuje, ale ….“.

Dokončení této věty vyplyne v průběhu práce, důležité v tuto chvíli je, že takový způsob existuje a nazývá se *cluster***.**

„Cluster je termín označující nezávislé počítače spojené sítí a softwarem do jednoho systému.“ [1].

Je velmi mnoho způsobů k realizaci clusteru, lišících se podle požadavků na cluster a hardwarového vybavení. Programy pro cluster je také možno vytvářet mnoha různými způsoby. To vše bude v této práci popsáno.

2 Cíle práce

Díky obsáhlosti a rozmanitosti celé problematiky clusterů bude nutná také rozsáhlá teoretická část. Představím jednotlivé druhy clusterů a jejich hlavní smysl a výhody, které poskytují. Pokusím se u každého z těchto druhů uvést co nejrozsáhlejší výčet, v současnosti používaných, technologií pro jeho realizaci.

Dále uvedu několik základních principů paralelního programování, aby bylo patrné, že ne každý program může být provozován na clusteru.

Praktická část práce se bude skládat z dvou částí. První částí je výběr vhodné distribuce pro realizaci. Pokusím se vybrat distribuce, které jsou v současnosti nejčastěji používány, abych podal zprávu o současném stavu problematiky a nabídce produktů. Dále navrhnu svůj způsob pro jejich hodnocení a porovnávání. Několik vybraných distribucí takto porovnám. Do hodnocení bych chtěl začlenit také jedno z komerčních řešení.

Distribuci, která získá v tomto hodnocení nejvíce bodů, pak nainstaluji a prakticky otestuji, to bude druhá polovina praktické části.

3 Literatura

Beowulf.org – Především fórum obsahuje mnoho cenných rad ohledně výpočetních clusterů.

Ze stránek svn.oscar.openclustergroup.org jsem získával informace o balíčku Oscar.

Na webu www.rocksclusters.org v sekci „Support and Docs“ se nachází celá oficiální dokumentace distribuce Rocks.

Na stránkách www.linuxhpc.org se nachází mnoho informací z oboru výpočetních clusterů.

Na www.root.cz se nachází několik článků o Mosixu, Paralelním programování a Xenu.

Domovské stránky jednotlivých produktů a jejich oficiální dokumentace.

4 Teorie

Pojem počítačový cluster[„klástr“](=shluk) chápeme jako skupinu počítačů navenek chovajících se jako jeden stroj. Ke zřízení clusteru obvykle vedou důvody uvedené v podkapitolách

4.1 Failover neboli high-availability cluster

Česky také vysoce dostupný cluster. Dnes nejčastější důvod ke konstrukci clusteru je zajištění vysoké dostupnosti kritických služeb (důležité databáze, frekventované stránky atd.).

Projekt LinuxHA se snaží učinit Linux použitelný na poli Failover clusterů. Celý projekt těží především z funkcí programu heartbeat, ten stačí spustit jako službu na obou počítačích, a vytvořit několik konfiguračních skriptů podle dokumentace a oba počítače se stávají vysoce dostupnými (přesněji služby na počítačích běžící a zapsané v konfiguračním souboru heartbeatu). Toto řešení má však mnoho problémů především v případech, kdy se jedná o služby hojně využívající filesystému nebo databází, v těchto případech (většina) je nutné zajistit replikaci dat, s čímž už nám heartbeat samozřejmě nepomůže.

4.1.1 Virtualizace

Problém vysoké dostupnosti lze však řešit lepším způsobem, než je failover cluster a tím je virtualizace.

Virtualizace je proces, kdy v jednom operačním systému (OS) spustíte pomocí virtualizačního systému jiný OS. Výhodou je, že systémy jsou od sebe odděleny a ani oprávnění roota na jednom počítači vám nepomůže k přístupu do druhého systému.[2]

V současné době existují 3 zhruba stejně kvalitní provedení. Jedná se o Hyper-V firmy Microsoft, VMware firmy Novell a XEN - open source projekt zastřešený firmou Citrix. Všechny tyto nástroje jsou spolu kompatibilní, je možné migrovat virtuální stroje mezi jednotlivými prostředími a globálně spravovat celou farmu obsahující všechna řešení pomocí jednoho nástroje.

Jak jsme již zvyklí, firma Microsoft poskytuje velmi komplexní, přehledné a ergonomické produkty. Ani u jejich virtualizačních produktů Hyper-V Server a Windows Server 2008 tomu není jinak. Pro řízení a správu prostředí slouží nástroje System Center Configuration Manager, Operation Manager a Virtual Machine Manager.

Konkurenci na poli komerčních produktů dělá Microsoftu pouze firma Novell s produktem VMware, který má dlouhou historii a mnoho spokojených uživatelů. Je multiplatformní, což je na jednu stranu výhodou, na druhou stranu je díky tomu pomalejší a mívá horší spolupráci s hardware.

Ve světě Linuxu a OSS do této trojice enterprise virtualizačních nástrojů zapadá produkt Xen. Existují však i další nástroje realizující virtuální prostředí, např. KVM nebo Qemu, o těch se však nemluví jako

o serverové vizualizaci, ale spíše jsou zařazeny do kategorie desktopových virtualizačních nástrojů, jako je např. Microsoft Virtual PC.

Vzhledem k zaměření této práce bude právě Xen prakticky otestován a popsán:

Kromě VMware a Microsoftu, kteří jsou výhradními distributory svých produktů, Xen je open source technologie dostupná od mnoha prodejců, včetně Red Hatu, Novellu a (v blízké budoucnosti) Sunu. Jeden z dalších poskytovatelů, jenž poskytuje Xen, je XenSource komerční sponzor projektu Xen (Poznámka: XenSource byl nedávno koupený Citrixem, a produkt se nyní nazývá Citrix XenServer). Produkty XenSource však nejsou open source, jako je to u ostatních poskytovatelů. Na druhou stranu XenSource má více funkcí než základní Xen. [16]

Elementární Xenovský hypervisor je základem XenSource. Xen implementuje paravirtualizační pojetí virtualizace, to znamená, že je navržen jako slabá softwarová vrstva, která řídí přístupy ke spodním hardwarovým zdrojům. Komunikace z hostovaných virtuálních strojů a zdroji spodního stroje (síť, uložiště) prochází přes privilegovaného hosta zvaného (v Xenovském názvosloví) „Domain0“ nebo „Dom0.“ Hostovaný virtuální stroj je nazývaný jako „DomainU“ nebo „DomU.“ [16]

Použití tenké hypervisor vrstvy umožňuje Xenovské virtualizaci docílit téměř nativního výkonu, důležitá výhoda, protože jedním

z hlavních problémů virtualizace byla daň v podobě výkonu, vynucená pro vloženou vrstvu virtualizačního software mezi hostovaný virtuální stroj a hardware. [16]

Jedním nedostatkem paravirtualizace bylo, že jádro hostovaného virtuálního stroje potřebovalo modifikovat pro správnou spolupráci s Xen hypervisorem. Poslední generace čipů od fy. AMD a Intel obsahuje hardwarové rozšíření, které umožňuje spouštět virtuální stroje na paravirtualizačním hypervisoru bez změn jádra.[16]

4.2 Compute cluster, HPC

HPC (High performance computing). Česky výkonový (výpočetní) cluster. Po propojení skupiny počítačů do sítě lze nejen využívat jejich celkovou diskovou kapacitu pomocí distribuovaných souborových systémů, ale také jejich celkový výkon.

4.2.1 Požadavky na aplikace

Pokud chceme provozovat aplikaci tak, aby mohla využít výhod výkonového clusteru, musí k tomu být přizpůsobena. O knihovnách MPI apod. a o pravidlech, která je nutné dodržet, najdete informace v kapitole paralelní programování.

4.2.2 Beowulf

Clustery typu Beowulf jsou škálovatelné, výkonné clustery založené na běžném hardware, na privátní síti s open source (Linuxovou) infrastrukturou.[16]

Všechny části clusteru se skládají z PC nebo pracovních stanic vyhrazených ke spouštění HPC úloh. Uzly v clusteru nejsou postaveny na stolech, jsou vyhrazeny pro provozování clusterových úloh.[16]

Připojit je k okolnímu světu lze jen přes jeden uzel.[16]

Některé Linuxové clustery jsou sestaveny pro zajištění spolehlivosti a ne rychlosti. Toto nejsou Beowulfy.[16]

4.3 Grid Cluster

Gridové clustery jsou ve své podstatě compute clustery, které jsou provozovány na rozlehlejší síti. Frontend celého gridu musí obsahovat komplexní nástroje pro spouštění a řízení úloh, který (např. na základě práv uživatele) stanoví na kterých a kolika počítačích z gridu může úloha běžet. Nejznámějším nástrojem tohoto druhu je SGE – Sun Grid Engine.

Gridový cluster se může rozprostírat po celém světě, jeho uzly mohou být výkonné superpočítače i běžná domácí PC. V provozu je několik takových projektů, do kterých se může každý vlastník počítače zapojit a přispět svým výkonem k objevení mimozemské civilizace v projektu SETI@Home, pomoci vědcům skládat proteiny v projektu

Folding@Home (čímž údajně lze léčit mnoho nemocí jako Alzheimer, BSE nebo Parkinson) apod..

Do gridu nemusíme jen přispívat svým výkonem, ale můžeme také využít grid pro své vlastní projekty. Jeden takový, dokonce český [17], umožňuje registrovaným uživatelům (studentům, vědcům a akademickým pracovníkům působícím na českých univerzitách) spouštět své vlastní úlohy a tak těmto uživatelům pomoci ušetřit výdaje potřebné na jejich výzkum.

4.4 Scallable Cluster

Slouží k rozdělení zátěže mezi několik strojů, většinou se však nejedná o zátěž výpočetního charakteru (to zastávají výkonové clustery). Ve většině případů se scallable cluster využívá u často navštěvovaných serverů v internetu. Každý uzel (server) je připojen na jinou linku, data se mezi všemi replikují. Tímto zároveň vzniká efekt failover clusteru.

5 Vývoj aplikací pro clustery

Aplikace určené ke zpracování na výpočetním clusteru musejí splňovat některá kriteria. Paralelní programátor má na výběr ze dvou možností.

- Naučit se nový jazyk, který je určen přímo pro paralelní programování (Ada).
- Získat některou z MPI nebo PVM knihoven pro svůj oblíbený jazyk.

5.1 Ada

Ada je programovací jazyk, který vznikl na začátku 80. let z potřeby amerického ministerstva obrany (DoD) vytvořit jeden univerzální jazyk, který by byl použitelný jak pro vložená (embedded) zařízení, tak pro velké systémy s milióny řádky kódu. Byl pojmenován po jisté Augustě Adě Byron, která spolupracovala s Charlesem Babbagem a pravděpodobně byla prvním programátorem na světě. Ada má syntaxi podobnou Pascalu a jedná se standardní nástroj zaměřený na tvorbu spolehlivých a přenositelných aplikací.[3]

Současná verze Ady se nazývá Ada95 a obsahuje mimo jiné prostředky pro OOP (jedná se o velice konzervativní návrh, funkčně na úrovni C++ s jednoduchou dědičností), jazykové konstrukce zaměřené na vícevláknové programování a generické konstrukce.[3]

Nejčastější využití nachází v mission/safety critical aplikací. Několik příkladů: řídící software k raketoplánům, jaderným elektrárnám, vlakům TGV, stíhačce Gripen atd. [3]

Ada 95 je první mezinárodně standardizovaný jazyk spojující v jednom návrhu vybavení pro multitasking i paralelní programování. Komunikace mezi distribuovanými oddíly je přes synchronní i asynchronní vzdálené volání procedur.[18]

5.2 DSM – Distributed Shared Memory Model

Snadno pochopitelný způsob paralelního programování pro programátory píšící běžné vícevláknové aplikace.

5.2.1 POSIX Threads – Pthreads

Knihovna pro podporu vícevláknových aplikací zavádí datové typy, makra a funkce potřebné pro vytvoření vícevláknové aplikace. Předpokládá se využití na jedno či víceprocesorovém stroji se sdílenou pamětí, poskytované funkce tudíž poskytují podporu pro programovací techniku (způsob interakce vláken) "data sharing". Pomocí poskytovaných prostředků je samozřejmě možné si vytvořit vlastní (uživatelské) objekty a funkce pro "message passing" ve sdílené paměti.[4]

Samotné Pthreads však neumějí pracovat na clusteru, je nutné využít některých knihoven a technik jak toho docílit.

5.2.2 THROOM

THROOM umožňuje nemodifikované POSIX (pthreads) binární soubory spouštět transparentně na clusteru. Klíčovou myšlenkou je rozšířit jednoprocesní multivláknový model na víceprocesní model, jehož vlákna jsou distribuována na procesy spuštěné na vzdálených nodech. Distribuovaná vlákna se provádějí v globálním sdíleném adresovém prostoru realizovaném logickou SW-DSM vrstvou. THROOM je také koncepce, která provozuje nemodifikované pthread binárky na virtuálním clusteru jako standardní UNIX procesy. THROOM běží na horní straně DSZOOM fine-grain SW-DSM systému, má tedy omezenou podporou operačních systémů.[5]

5.2.3 DSZOOM – Low latency software-based Shared memory

Neboli česky: „Softwarově emulovaná sdílená paměť s nízkou odezvou.“

5.2.4 PJ – Parallel Java

Paralel Java je API a middleware pro paralelní programování v Javě na SMP strojích, clusterech a hybridních clusterech složených ze SMP strojů.[19]

5.2.5 Distributed Computing in Java

Relativně jednoduchým způsobem lze v javě realizovat i vlastní paralelní aplikaci. Potřebujeme k tomu znalost několika Javových technologií:

- Threads – vlákna v Javě
- Serialization – serializace objektů
- Sockets – komunikace po síti
- RMI – Remote Methods Invocation – vzdálené volání metod
- Jini/JavaSpaces – realizace distribuované sdílené paměti

Programovací jazyk Java je stále volený jako jazyk pro implementaci enterprise a distribuovaných aplikací. To může být způsobeno nezávislostí na platformě, jednoduchým objektově orientovaným modelem, čistou syntaxí nebo vestavěnou podporou pro základní kameny (vlákna a sockety) pro vývoj distribuovaných aplikací. [20]

Tvorba distribuovaných aplikací je obtížná, protože se musíme vžít do několika situací, jako jsou částečná selhání, zvýšená odezva, distribuovaná presistence a jazyková shoda.[20]

5.2.5.1 JavaSpaces

Technologie JavaSpaces je jednoduchý a výkonný vysokoúrovňový nástroj pro vytváření distribuovaných a kolaborujících

aplikací. Je založená na konceptu sdíleného síťového prostoru, který obsahuje jak úložiště objektů, tak prostor pro výměnu, to vše nám poskytuje jednoduché API, které je snadné na naučení a zároveň mocné při budování sofistikovaných distribuovaných aplikací.[20]

Patrná odchylka od běžných distribuovaných modelů, které spoléhají na message passing (viz kapitola 5.3) nebo na RMI, JavaSpaces model vidí distribuovanou aplikaci jako kolekci procesů, které spolupracují s tokem objektů dovnitř a ven jednoho nebo více prostorů.[20]

5.2.5.2 RMI - Remote Method Invocation

Komunikaci mezi službami může být realizována za pomoci Java Remote Method Invocation (RMI). Infrastruktura pro podporu komunikace mezi službami není jediná služba, která byla objevena a používána, ale spiše je to část infrastruktury technologie Jini.[21] RMI poskytuje mechanizmy pro hledání, aktivování a garbage collect skupin objektů.[21]

Především RMI je rozšíření jazyka Java o tradiční mechanizmus volání vzdálených procedur. RMI nepodporuje jen posílání dat od objektu k objektu přes síť, ale také celé objekty obsahující kód. Velká část čistoty Jini systému je dána právě tímto důvtipným způsobem přesouvání kódu po síti, který umožňuje encapsulaci objektů.[21]

5.3 Message Passing Model

Při provádění message-passing programu každý z procesorů provozuje jeden sub-program. Všechny proměnné jsou tedy privátní. Komunikace probíhá pouze přes volání speciálních subrutin.

Ke komunikaci slouží zprávy (message), které jsou vlastně pakety putujícími mezi sub-programy. Zpráv je několik druhů: point-to-point, kolektivní, synchronní a asynchronní.

Každý z výrobců má své vlastní MPI standardy, to má za důsledek existenci mnoha různých funkcí, ale také nekompatibilitu a celkové zmatení celé problematiky. Nejznámějšími implementacemi jsou OpenMPI, MPICH, MPICH2, PVM a Microsoft MPI, existuje však mnoho dalších.

Na stránkách [22] byl neoficiálně definován standard zvaný MPI (Message Passing Interface), který je dnes ve verzi 2.1.

5.3.1 Paralel Virtual Machine - PVM

Parallel Virtual Machine (PVM) je starší, ačkoliv dosud používaná, knihovna, která byla inspirací pro návrh modelu MPI.

Je to systém, který umožňuje programátorům pohlížet na heterogenní soubor unixových strojů jako na jednolitý paralelní počítač. PVM pracuje na jednoduchém, ale funkčně kompletním modelu předávání zpráv (*message passing model*).[23]

Základem PVM je démon pvmd3, který běží na každém počítači. Všechny uzly se spuštěným démonem pvmd3 mohou dohromady vytvářet virtuální stroj.[23]

Druhou částí PVM je knihovna, poskytující rozhraní pro paralelní operace. V současné době jsou podporovány jazyky C, C++ a Fortran.[23]

6 Metodika

6.1 Postup zjišťování informací

Před realizací clusteru je nutné především vědět, jaké úlohy chceme na clusteru provozovat, zda se bude jednat o cluster vysoce dostupný, výpočetní, nebo gridový. Abychom se mezi těmito termíny zorientovali, musíme nastudovat některé termíny a pochopit základní principy funkce jednotlivých řešení (viz teoretická část práce). Pak můžeme rozhodnout, který cluster postavíme, jaké zhruba budou hardwarové nároky a které technologie pravděpodobně využijeme.

Mé požadavky na hardwarové vybavení byly téměř nulové, byly tedy stanovovány minimální konfigurací vybraného clusterového řešení. O konfiguraci jednotlivých clusterů se zmiňuji v praktické části práce.

Požadavků na výběr operačního systému bylo několik. V první řadě požadavek plynoucí ze zadání práce, kterým je omezení okruhu výběru operačních systému pouze na Linuxové distribuce, to mi však nebrání v tom, abych se okrajově zmínil i o jiných řešeních alespoň v teoretické části.

Mým dalším požadavkem bylo, aby vybraná distribuce byla typu all-in-one, tedy aby už po instalaci obsahovala co nejvíce clusterových technologií a pokud možno nebylo nutné doinstalovávat žádné další balíčky nebo provádět kompilace nových jader. Domnívám se, že pro první pokusy s clustery a pro akademickou půdu je tento požadavek

velmi vhodný. Při potřebě konkrétní funkce už není potřeba volit univerzální distribuci.

Dalším pochopitelným kritériem při výběru takovéto distribuce je velikost komunity a celkové uživatelské základny, pro případ některých nedokumentovaných problémů, které se ve světě Linuxových distribucí často vyskytují. Samozřejmě také velkou váhu bude mít hodnocení rozsahu oficiální dokumentace k této distribuci, nebo alespoň k distribuci, ze které vychází.

S výběrem konkrétních kandidátů mi velmi vydatně pomohl pan Mgr. Jiří Pech, Ph. D., který mi doporučil distribuci Rocks, která nakonec zvítězila v mém hodnocení a byla tedy použita k realizaci clusteru. S výběrem dalších distribucí mi pomohl především portál sloužící k vyhledávání [14] a portál se seznamem Linuxových distribucí [15], jejichž pomocí jsem našel další distribuce.

Do nejužšího výběru se dostaly tyto operační systémy:

- Rocks 5.1, x86 (dále jen Rocks)
- Oscar 5.1 beta2 na Fedora Core 8 (dále jen Oscar)
- clusterKNOPPIX 3.6
- Microsoft Windows HPC Server 2008 - 64-bit with Hyper-V Enterprise Edition Trial (dále jen WinHPC)

Rocks je distribuce cílená na vědeckou obec a na akademické pole, je velmi jednoduchá na instalaci a obsahuje velké množství již

připravených aplikací především pro biologické a genetické výpočty. Je však ochuzena po stránce enterprise funkcí, protože není cílená pro komerční využití ve firmách.

Oscar je jen balíček knihoven a programů, je kompatibilní a dostupný jen pro několik vybraných distribucí, proto se také některé body hodnocení mohou lišit podle podkladové distribuce.

ClusterKnoppix je velmi zastaralá distribuce využívající zastaralý clustrový model openMosix, který je však velmi zajímavý svou koncepcí. Dostává velmi malé bodové ohodnocení především díky špatné podpoře hardware, nainstalovat jej na moderní hardware je velmi složité. Pokud je provozován na vhodném hardware, je velice jednoduchým instantním clusterem.

WinHPC je jediným komerčním řešením v mém testu. Především bych na něm chtěl demonstrovat ohromný rozdíl komerčních řešení a komunitních, Linuxových produktů. Vyniká především přístupem jak k uživateli, tak k programátorovi, oběma skupinám nabízí nevídané nástroje. A s příchodem .NET 4.0 tento náskok velmi výrazně vzroste.

6.2 Způsob hodnocení vlastností, definice stupnice

Každá cílená distribuce či operační systém obsahuje několik funkcí, kvůli kterým se stává výjimečnou, tyto vlastnosti se u takovýchto distribucí hodnotí nejčastěji, proto i na mé hodnocení budou mít velký

vliv. Nerad bych však zanedbal hodnocení týkající se běžných požadavků na operační systém obecně.

Každé kriterium bude hodnoceno 0-5ti body.

- 0 – tuto funkci nepodporuje, nelze nebo lze velmi obtížně dodat do systému
- 1 – funkce lze dodat
- 2 – funkci systém obsahuje jako svou automatickou nebo volitelnou součást
- 3 – funkce je v systému, je dobře dokumentovaná a existuje komunita mnoha uživatelů této funkce na tomto operačním systému
- 4 – kromě funkce samotné je v systému ještě balík speciálních utilit usnadňujících práci s touto funkcí, nebo jí automaticky zpravuje v rámci vyššího kontextu (odstínění od vlastností konkrétní funkce.)
- 5 – obsahuje citelně lepší alternativu požadované funkce

Toto hodnocení bude v některých případech nahrazeno.

Každé téma má několik dílčích otázek. Součet těchto bodů odhalí, který z testovaných systémů se nejlépe hodí k realizaci nejrůznějších druhů clusterů. Z kritérií se pokusím vynechat věci, které jsou nedokazatelné, nebo diskutabilní, jako je hodnocení stability a spolehlivosti.

V následujících kapitolách jsou rozebrána jednotlivá kriteria hodnocení.

6.2.1 Hodnocení speciálních Clusterových funkcí

Vzhledem k existenci více druhů Custerových řešení, z nichž má každé své specifické požadavky na systém, bude se hlavní část tohoto hodnocení skládat ze 4 částí:

1. Možnosti pro realizaci virtuálního prostředí nebo high-availability clusteru.
2. Podpora high-performance-computingu ve formě existence knihoven MPI a PVM.
3. Podpora high performance pro vyšší jazyky (Java)
4. Možnosti pro řazení úloh, kvalita a propracovanost user managementu, příprava pro grid cluster.

Body získané v této skupině se pro celkové porovnání budou násobit konstantou 2, abychom získali relevantnější hodnocení, vzhledem k důležitosti těchto vlastností.

Tabulka 1 - Hodnocení clusterových funkcí

část	Rocks	Oscar	clusterKNOPPIX	WinHPC
1.	2	3	0	4
2.	3	3	5	5
3.	2	1-2	1	5
4.	3	3	1	4
Součet	10	10	7	18
Krát 2	20	20	14	36

Zdůvodnění Rocks:

1. Xen obsahuje jako volitelnou roli, není však dobře dokumentovaná.
2. OpenMPI i MPICH2 jsou volitelné funkce, tyto knihovny jsou dobře dokumentované.
3. Role Java obsahuje pouze běžné SDK bez rozšířených knihoven pro cluster, ty je třeba doinstalovat.
4. Sun Grid Engine je volitelnou součástí s dobrou dokumentací přímo od společnosti Sun

Zdůvodnění Oscar:

1. Xen je součástí, není v oficiální dokumentaci zmíněn, ale existuje několik článků s touto problematikou.

2. Implementace MPI funkcí je vestavěná
3. Java v balíčku není, může však být v podkladové distribuci
4. SGE stejne jako u Rocks

Zdůvodnění clusterKnoppix:

1. Virtualizace nebyla v době vydání známá
2. Kromě MPI implementací obsahuje modifikaci jádra OpenMosix.
3. Neobsahuje, je velmi obtížné dodat, vzhledem ke známým problémům Knoppixových distribucí s funkcí balíčkovacího systému apt-get.
4. Opět poplatné době, kdy ještě gridové clustery nebyly známy.

Zdůvodnění WinHPC:

1. Hyper-V je obdoba Xenu, má ale větší podporu v podobě MSDN, mnoha konferencí i placené podpory v mnoha různých podobách.
2. Vlastní implementace MPI, podpora Visual Studia, řešení přes webové služby.V budoucnu .NET 4.0 s podporou pro paralelní programování.
3. .NET framework je lepší alternativou Javy, viz předchozí zdůvodnění.

4. Vlastní nástroj začleněný v nástrojích pro správu, možnost použit i SQE i SASCG

6.2.2 Hodnocení enterprise funkcí

Cluster, pokud má najít komerční využití, musí obsahovat funkce, které usnadňují správci integraci tohoto systému do podnikové sítě a jeho správu.

1. Hromadná instalace uzlů. 0 bodů – (ne), 7 bodů – (pomocí speciálního nástroje), 10 bodů – (ano)
2. Kooperace s enterprise nástroji pro dohled a správu.
3. Možnosti propojení s existujícím user managementem (ověřování, identity management, adresářové služby).
4. Multiplatformnost (z pohledu architektury). 0 bodů – (jen speciální stroje), 1 bod – (jen PC), 3 body – (běžné platformy), 5 bodů – (široké pole platforem), 10 bodů – (široké pole platforem plus herní konzole a GPU apod.)

Tabulka 2 - Hodnocení enterprise funkcí

část	Rocks	Oscar	clusterKNOPPIX	WinHPC
1.	10	10	7	7
2.	2	1-3	1	4
3.	2	2-3	2	4
4.	5	5	1	3
součet	19	18	11	18

Zdůvodnění Rocks:

1. Insert-ethers
2. Možné dodat balíček do podkladové distribuce CentOS
3. Většinu Linuxu je možné propojit s LDAP a jsou kompatibilní s ověřováním Kerberos. AD je možné.
4. Linux je kompatibilní se širokým polem platforem.

Zdůvodnění Oscar: Vše závislé na podkladové distribuci.

Zdůvodnění clusterKnoppix:

1. OpenMosix Terminál Server + omdicd
2. Zastaralé standardy je nutné obnovit, pokud to bude možné.
3. Stejně jako u Rocks. LDAP, Kerberos.

4. Zastaralé jádro neobsahuje ovladače na žádné nové zařízení. Možné použít jen na staré sestavě.

Zdůvodnění WinHPC:

1. Přes AD, nebo komfortněji přes SCOM.
2. Ano, velmi rozmanitá. Nejlepší s těchto nástrojů pocházejí od stejného výrobce (rodina Systém Center), mnoho dokumentace.
3. AD, LDAP, NTLD 1,2,3, Kerberos atd., mnoho dokumentace.
4. S podporou platforem je na tom hůře.

6.2.3 Hodnocení z pohledu akademického

Vzhledem k faktu, že výpočetní clustery jsou stále ve velké míře využívány pouze na akademické, příp. vědecké půdě, je tento pohled také velmi důležitý.

1. Náklady na pořízení. 0 bodů – (dražší než 500 Kč za uzel), 5 bodů – (do 500 Kč/uzel), 10 bodů – (zdarma)
2. Náročnost na hardware. 0 bodů (high-end PC), 3 body – (běžný kancelářský PC), 5 bodů – (uzel může být levné PC např. z bazaru)
3. Přítomnost software pro biologické výpočty.
4. Přítomnost simulačního software.

5. Snadnost ovládání – převládající způsob. 0 bodů (textové rozhraní), 5 bodů – (grafické uživatelské rozhraní), 10 bodů – (ergonomický průvodce s nápovědou)

Tabulka 3 - Hodnocení z pohledu akademického

část	Rocks	Oscar	clusterKNOPPIX	WinHPC
1.	10	0-10	10	0
2.	3	3	0	0
3.	3	2	2	2
4.	2	2	2	2
5.	0	5	0	10
Součet	18	12	14	14

Zdůvodnění Rocks, Oscar, clusterKnoppix

1. Zdarma, jen u Oscar záleží na podkladové distribuci.
2. U Knoppix špatná podpora nového hardware limituje použití novějšího hardware.
3. U Rocks je to jedna ze stěžejních funkcí.
4. Simulační software jsem nenašel v žádné distribuci, 3rd party existuje.
5. U Oscar je přívětivější rozhraní.

Zdůvodnění WinHPC:

1. Cena je ještě výrazně vyšší, než 500 Kč/uzel
2. Náročnost na hardware je vysoká, Hyper-V vyžaduje instrukce Intel-VT nebo AMD-V obsažené jen v nejnovějším hardware.
3. 3rd party.
4. 3rd party.
5. Ergonomie, přehlednost, pohodlnost ovládání je na nejvyšší úrovni.

6.2.4 Dokumentace, komunita, podpora

Jak jsem již zmínil v úvodu, veliký důraz bude též kladen na kvalitu dokumentace, velikost komunity, či existenci placené nebo neplacené podpory.

1. Rozsah oficiální dokumentace.
2. Počet uživatelů (úměrný k velikosti komunity). Body budou rozděleny podle pořadí.
3. Další podpora.
4. Možnosti zaškolení, certifikáty.

Tabulka 4 - Hodnocení dokumentace a podpory

část	Rocks	Oscar	clusterKNOPPIX	WinHPC
1.	2	2	1	5
2.	4	3	2	5
3.	0	0	0	5
4.	0	0	0	5
Součet	6	4	4	20

Zdůvodnění Rocks, Oscar, clusterKnoppix:

1. Oficiální dokumentace jen z povinnosti, velmi nízká úroveň.
2. Fóra a diskusní skupiny jsou jediná místa, kde lze získat některé informace, vyhledávat takto zásadní informace je běžnou praxí.
3. Další podpora existuje pouze pro některé podkladové distribuce a to ve velmi omezené formě.
4. Stejně jako s podporou, jen omezeně pro podkladové distribuce.

Zdůvodnění WinHPC:

1. Propracovaná nápověda v systému i online (někdy dokonce lokalizovaná), MSDN, KB, rozsáhlé domovské stránky.
2. Ohromná komunita, vzhledem k množství společných znaků s ostatními verzemi Windows.
3. Podpora jak přímo od Microsoftu nebo od dalších poradenských certifikovaných firem, po telefonu, e-mailu nebo osobně. Mnoho odborníků a specializovaných firem ve světě i v ČR.
4. Školení, konference, workshopy, tutoriály a videotutoriály od MS i dalších partnerů. Rozsáhlé pole certifikačních programů.

6.3 Konečné hodnocení

Tabulka 5 - Výsledné hodnocení distribucí

část	Rocks	Oscar	clusterKNOPPIX	WinHPC
6.2.1.	20	20	14	36
6.2.2.	19	18	11	18
6.2.3.	18	12	14	14
6.2.4.	6	4	4	20
Součet	63	54	43	88

Z hodnocení jsou patrné přednosti a nedostatky jednotlivých řešení.

Všechny Linuxové distribuce trpí fatálním nedostatkem podpory, který snižuje jejich použitelnost.

ClusterKnoppix má rysy typické pro ukončené projekty: nepodporuje novější hardware ani nové technologie. OpenMosix však má velkou cílovou skupinu uživatelů díky svému nekonvenčnímu přístupu.

WinHPC je podle očekávání vítězem především díky výborné podpoře a možnostem, které Microsoft poskytuje vývojářům.

7 Distribuce Rocks

Rocks je open-source clusterová linuxová distribuce, která umožňuje koncovým uživatelům snadné sestavení výpočetních clusterů, zakončení gridu a znázornění TDW (tiled display walls = metoda 3d zobrazování). Stovky výzkumných pracovníků na celém světě používají Rocks k sestavení svého clusteru.[6]

Počínaje květnem 2000 byly skupině Rocks zasílány problémy ve vývoji řízených clusterů. Skupina měla jen jeden cíl: udělat clustery jednoduchými. Pod pojmem jednoduché myslíme jednoduché sestavit, spravovat, inovovat a škálovat. Byla hnána cílem pomoci využít výpočetní sílu clusteru širokému poli vědeckých pracovníků. Je jisté, že vytvoření stabilní a spravovatelné platformy dostupné širokému poli vědců může nesmírně pomoci při zlepšování nejmodernějších paralelních nástrojů.[6]

7.1 Technologie

Distribuce je vystavěna nad linuxovou distribucí CentOS (The Community Enterprise Operating Systém), která se již sama zaměřuje na enterprise záležitosti, jakými jsou právě clustery a virtualizace. Ani ona však není původní distribucí, vychází z Red Hat Enterprise linuxu. Nyní uvedu některé rysy, které CentOS má a které od něj podědila distribuce Rocks.

7.1.1 Možnosti pro instalaci přes síť

Pro cluster je velmi vhodná a potřebná možnost automatické instalace uzlů přes síť. Po nainstalování systému na hlavní server (frontend nebo také head node) se na něm spustí DHCP server, který jednotlivým compute-nodům udělí IP adresu a další nezbytná síťová nastavení. TFTP server (též PXE server), také spuštěný na frontend stroji, obsahuje soubory nezbytné pro spuštění compute-nodů. Zbylá data o konfiguraci a balíčcích mohou být umístěna na síťovém filesystému (nejčastěji NFS), na FTP serveru nebo na HTTP serveru spuštěným také na frontend stroji.

Pro pohodlnou konfiguraci slouží balíček system-config-netboot. Nutnou úpravu souboru *dhcpd.conf* patrně v CentOS musí provést uživatel, v Rocks je již automatizována.

Na jednotlivých compute-nodech musí být nastaveno bootování ze sítě (PXE).

7.1.1.1 Kickstart

Mnoho systémových administrátorů by upřednostnilo používání automatické instalační metody k instalaci Red Hat Enterprise Linuxu na své stroje. Jako odpověď vytvořil Red Hat instalační metodu kickstart. Při použití kickstart může administrátor vytvořit pouze jeden soubor obsahující odpovědi na všechny otázky, na které se za běžných okolností ptá instalátor.[7]

Soubory kickstartu mohou být umístěny na jednom serveru a čteny jednotlivými počítači v průběhu instalace. Tato instalační metoda může podporovat použití jednoho kickstart souboru pro instalaci Red Hat Enterprise Linux na mnoho strojů, což jí dělá ideální pro síťové a systémové správce. [7]

Kickstart soubor může být snadno sestaven pomocí utility system-config-kickstart.

7.2 Prvky distribuce Rocks - Rolls

Osobní zkušenost mám pouze s nyní nejaktuálnější verzí 5.1. Distribuce Rocks se skládá z tzv. Rolls neboli rolí, které obsahují skupinu balíčků zaměřených na konkrétní funkci, kterou může stroj mít. Rolí Rocks nabízí v základu několik:

7.2.1 Area51

Skládá se ze 2 balíčků, které mají za úkol kontrolovat změny v souborech a tím zvyšovat bezpečnost systému a zpřehledňovat tak správu celého clusteru. Jedná se o balíčky Tripwire a Chkrootkit.

- Tripwire je open source bezpečnostní utilita zajišťující integritu dat, je použitelná pro sledování a oznamování změn ve specifických souborech na několika systémech.[8]

- Chkrootkit slouží k odhalování rootkitů (program maskující svou existenci, existenci jiného programu nebo útok hackera tím, že mění systémové knihovny a aplikace).

7.2.2 Bio

Role Bio obsahuje balíčky potřebné pro Bio-informatiku, tedy souprava aplikací, umožňující na clusteru počítat úlohy od rozpoznávání hlasu, až po úlohy s DNA nebo proteiny.

- Balíček HMMER slouží k realizaci výpočtů založených na „skrytých Markovových modelech“ – HMM = hidden markov model (od zmíněného rozpoznávání hlasu přes vyhledávání, např. v databázích až po analýzu sekvencí proteinů).
- mpiBLAST je volně dostupná, open-source, paralelní implementace NCBI BLAST. Slouží k efektivnímu využití distribuovaných výpočetních zdrojů k fragmentaci distribuovaných databází, rozdělování dotazů, inteligentnímu plánování, paralelním I/O operacím.[9]
- Biopython je kolekce pythonových aplikací a knihoven sloužících k výpočtům v oblasti molekulární biologie.

A mnoho dalších balíčků zaměřených na biologické výpočty.

7.2.3 Ganglia

Ganglia je škálovatelný distribuovaný monitorovací systém určený pro výpočetní clustery nebo gridy. Základ tvoří hierarchický design cílený na společenství clusterů. Používá účinné, široce používané technologie jako XML pro reprezentaci dat, XDR pro kompaktní přenos dat a RRDtool pro ukládání dat a vizualizaci. Používá pečlivě navržené datové struktury a algoritmy pro velmi nízké zatížení jednotlivých nodů a vysokou souběžnost. Implementace je robustní a je portována na značné množství operačních systémů a architektur, dnes jí používají tisíce clusterů po celém světě. Byla použita ke spojení clusterů napříč univerzitami celého světa a umožňuje tak spravovat clustery s více než 2000 nody.[10]

Webové stránky dostupné z odkazu poskytují grafické rozhraní pro živé informace o clusteru poskytované ganglia monitory spuštěnými na všech nodech. Monitory shromažďují hodnoty pro různá měření, jako je zátěž CPU, volná paměť, zaplnění disku, komunikace po síti, verze operačního systému atd. Tyto údaje jsou zasílány přes vnitřní síť clusteru a jsou použity na frontendovém stroji pro generování grafů a historie.[11]

7.2.4 HPC

Role HPC přináší implementace OpenMPI, MPICH1 a 2, PVM atd. Popis těchto technologii (viz kapitola 5.3). Použití (viz kapitola 8.2.6).

7.2.5 Sun Grid Engine – SGE

Sun Grid Engine je systém pro správu dávkového zpracování úloh. Hlavní myšlenka těchto systémů spočívá v tom, že vámi požadovaný výpočet zapíšete do skriptu a ten předáte do SGE. Ten se na základě aktuálního stavu rozhodne, kdy a na kterém z dostupných uzlů vaši úlohu spustí. Na každém výpočetním uzlu je definována fronta s kapacitou 1-4 úlohy podle počtu procesorů. Do těchto front jsou zasílány k vykonání jednotlivé úlohy z centrálního plánovače.[12]

Kromě konzolových nástrojů pro ovládání SGE existuje také velmi komplexní a propracovaný grafický nástroj spustitelný příkazem

qmon.

7.2.6 Xen

Xen je jediným nástrojem obsaženým v distribuci Rocks, který nemá svůj hlavní účel v high-performance-computingu, je to totiž velmi mocný virtualizační systém (viz kapitola 4.1.1).

Virtualizační technologie Xen, dostupná pro Linux, je navržená ke konsolidaci několika operačních systémů pro spuštění na jednom serveru, normalizovaný přístup k hardware pro operační systémy, izolování nekorektních aplikací a možnost migrovat běžící instance operačních systémů z jednoho fyzického serveru na jiný.[24]

7.3 Konzole Rocks

Hlavním ovládacím prvkem celé distribuce Rocks je rocks konzole.

Rocks konzole byla vytvořena, aby poskytovala více uniformní rozhraní k používání základních struktur na správu systémové konfigurace a chování. Všude, kde je to možné, Rocks používá SQL databázi (v současné době MySQL) k ukládání informací o uzlech, rozdělení disků, bootovacích parametrech a mnoha dalších informací. Na základě informací v databázi jsou upravovány vhodné konfigurační soubory. Regenerování těchto souborů se provádí pokaždé, když je přidán nebo odebrán nový uzel z clusteru. Regenerování také může být urychleno. Postup pro změnu konfigurace má dva kroky: [13]

1. Použijte příkaz konzole rocks pro změnu konfigurace v databázi (např. *rocks set host*)
2. Přepište konfigurační soubory použitím příkazu *rocks sync config*

8 Praktická část - Instalace distribuce Rocks 5.1

Jak uvádím v předchozí kapitole, distribuce Rocks slouží k snadné instalaci a správě clusteru. Vybral jsem její nejnovější verzi 5.1. Mé testování probíhalo v několika fázích.

8.1 Fáze 1. Cluster na starším HW

HW konfigurace tohoto clusteru:

- Frontend
  - CPU: AMD Thurion64 X2 (2 x 1,6 GHz)
  - RAM: 1GB
  - HDD: 1x 120GB 5400ot./min.
  - 1 x Gigabit ethernet
- Uzel #1
  - CPU: Intel Pentium MMX (1 x 200MHz)
  - RAM: 32MB
  - HDD: 1x 6GB 7200ot./min.
  - 1x 100Mbit ethernet

Na této konfiguraci jsem nedokázal distribuci Rocks nainstalovat, důvodů bylo několik:

- Rocks vyžaduje (chyba instalačních skriptů) specifické rozdělení disku. Frontendem byl však můj notebook, kde jsou nainstalovány Windows Vista jako můj hlavní operační systém, tudíž nepřipadalo v úvahu nechat Rocks, aby použil celý disk.
- Uzel #1 nesplňoval HW nároky distribuce Rocks jak velikostí paměti RAM, tak velikostí diskové kapacity.

- Uzel #1 dále také nepodporoval PXE bootování (bootování ze sítě). Později jsem zjistil, že existují možnosti, jak PXE boot nasimulovat pomocí bootovací diskety či CD, tento stroj však neobsahoval ani jedno potřebné čtecí zařízení.

Kromě těchto problémů řešitelných i neřešitelných mě od instalace na tyto stroje odradil také můj předpoklad, že bilance takto nevyváženého clusteru (běžný notebook + extrémně zastaralý desktop) bude velmi nepříznivá a hodnotit kvalitu clusterového řešení na této konfiguraci by mohlo vést ke zkreslení celého hodnocení. Všechny tyto aspekty mě přesvědčily, že bude vhodnější použít pro instalaci jiný hardware.

8.2 Fáze 2. Cluster na Výpočetním ústavu EF JU

Konfigurace:

- Frontend
  - CPU: Intel Core 2 Duo E6550 (2x 2,33GHz)
  - RAM: 2 GB
  - HDD: 1x 80GB 7200ot./min.
  - 1x Gigabit + 1x 100Mbit ethernet
- Uzly #1 - #3
  - CPU: AMD Athlon64 2800+ (1x 800MHz)
  - RAM: 512 MB
  - HDD: 1x 80GB 7200ot./min.
  - 1x 100Mbit ethernet

Na této konfiguraci se po stránce hardwarových omezení nevyskytl žádný problém. Instalace tak mohla být dokončena.

8.2.1 Instalace frontendu

Po několika nevydařených pokusech o instalaci, zakončených výjimkou instalačního průvodce, zpříčiněnou chybným rozdělením disku, jsem v [12] nalezl tabulku uvádějící doporučené nastavení a zároveň nastavení skrývající pod volbou „Auto Partitioning“ v instalačním průvodci.

Tabulka 6 - Rozdělení disku [12]

Partition Name	Size
/	16GB
/var	4GB
swap	1GB
/export (symbolically linked to /state/partition1)	remainder of root disk

Využil jsem tedy volbu automatického nastavení rozložení disku.

V průvodci jsem vybral několik rolí, které jsem chtěl vyzkoušet, nebo které byly nutné pro běh systému. Byly to tyto: base, ganglia, hpc, java, kernel, os, sge, web-server, xen. Popis jednotlivých rolí je obsahem minulé kapitoly.

Dalším krokem bylo vyplnění některých informací o pojmenování počítače, doméně, geografickém umístění a adresách. Frontendu jsem dal název ClusterEF01 a adresu cluster.ef.jcu.cz.

Nyní následovala již popsaná problémová část s rozdělováním pevného disku a samotné kopírování souborů. Instalace je tedy tímto krokem dokončena.

8.2.2 Konfigurace sítí

Nastavení sítě jsem si rozvrhl takto:

Tabulka 7 - Konfigurace sítě

Rozhraní	eth1	eth0
IP Adresa	160.217.X.Y	192.168.1.1
Maska	255.255.255.0	255.255.255.0
Popis	100Mbit ethernet k připojení do vnější sítě a do internetu	1Gbit ethernet sloužící ke komunikaci s ostatními nody clusteru

Konfigurace routovacích pravidel byla také aktualizována. Konkrétně příkazem:

route add default gw 160.217.X.Z

Výsledná tabulka routovacích pravidel vypadala takto:

Tabulka 8 - Konfigurace routovací tabulky

Destination	Gateway	Genmask	Iface
255.255.255.255	0.0.0.0	255.255.255.255	eth0
160.217.X.Y	192.168.1.1	255.255.255.255	eth0
160.217.X.0	0.0.0.0	255.255.255.0	eth1
192.168.1.0	0.0.0.0	255.255.255.0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	virbr0
169.254.0.0	0.0.0.0	255.255.0.0	eth0
0.0.0.0	160.217.X.Z	0.0.0.0	eth1

Do souboru */etc/resolv.conf* byl dopsán záznam o nameserveru používaném v síti ekonomické fakulty

- nameserver 127.0.0.1
- *nameserver 160.217.X.W*
- search local ef.jcu.cz

Dále jsem požádal správce místní sítě o přidání záznamu cluster.ef.jcu.cz = 160.217.X.Y do doménových záznamů na výchozím nameserveru s adresou 160.217.X.W, tato úprava byla nutná pro provoz webového rozhraní ganglia.

K distribuci systému na jednotlivé uzly je ještě nutné zapnout službu DHCP serveru, tedy spuštění deamonu dhcpd, který na Linuxu tuto službu realizuje. Slouží k tomu příkaz

service dhcpd start

Nyní byla dokončena úvodní konfigurace frontendu a zbývalo rozdistribuovat systém na jednotlivé uzly.

8.2.3 Hromadná instalace uzlů clusteru

Dalším krokem při instalaci clusteru byla instalace uzlů. V distribuci Rocks k tomuto účelu, jak jsem popsal v předchozí kapitole, slouží celá skupina systémů, ať už přejatých z distribuce CentOS (např. anakonda nebo kickstart), tak i vlastních funkcí (konzole rocks a nástroj *insert-ethers*).

V první řadě však bylo nutné na jednotlivých uzlech nastavit v BIOSu možnost bootování z PXE (pokud je to možné), u strojů, které jsem měl k dispozici, to byla volba „NVIDIA Agent“. Toto nastavení je závislé na použitém hardware a může se tedy lišit.

Nyní může začít instalace uzlů. V nástroji insert-ethers spuštěném na frontendu zvolíme možnost „compute node“ pro vytvoření uzlu pro výpočetní cluster. Nyní každý z bootujících uzlů získá své síťové nastavení z DHCP serveru a začne provádět PXE boot z frontendu. Takto, bez zásahu uživatele, projde celou instalací a v ideálním případě po několika desítkách minut provede restart a stane se uzlem clusteru.

Málokdy se však povede instalace bez problémů nebo chyb v konfiguraci[1]. Ani u mě se tomu tak nestalo a po přepojení monitoru a klávesnice na konkrétní uzel jsem našel následující problém.

Instalační program vyhodnotil, že by bylo vhodné k bootovacím parametrům přidat příkaz

dom0_mem=1024M,

který zapříčinil chybu zavádění „Error 28: Selected item cannot fit into memory.“ Chyba nastala z důvodu nedostatku paměti, uzly mají totiž jen 512MB RAM. Po stisku libovolné klávesy vstoupíme do rozhraní zavaděče grub, kde pomocí stisku klávesy ‘a‘ („to modify the kernel arguments before booting“) zeditujeme tyto parametry. Hodnotu u parametru dom0_mem změníme z 1024M na 512M a po stisku enteru nabootujeme stroj.

Instalace jednotlivých uzlů se provádí postupně, bohužel jsem nedokázal spustit proces instalace na několika uzlech naráz, ačkoli se o této chybě v dokumentaci autoři nezmiňují.

[1] Z několika zdrojů jsem se dozvěděl, že verze 5.0 byla spolehlivější a méně problémová.

Po nainstalování a nabootování všech uzlů (u mě 3) se můžeme přesvědčit o jejich funkčnosti několika způsoby. Buďto pomocí příkazu

rocks list host,

nebo pomocí webového rozhraní ganglia (web server spuštěný na frontendu). Případně je možné použít příkaz

ping compute-0-x,

kde x je pořadové číslo uzlu indexované od 0 a přidělované ve stejném pořadí v jakém probíhala instalace.

Nyní je cluster funkční. Stále však má chybu v bootovacích parametrech, která způsobí, že se pro restartu jednotlivé uzly nespouštějí.

8.2.4 Odstranění problému s bootováním uzlů

Pomocí

rocks report host bootflags

získáme výpis bootovacích parametrů u jednotlivých uzlů.

Příkaz:

rocks set host bootflags %host% flags="%flags%"

slouží k úpravě těchto parametrů.

Provedeme tedy

rocks set host bootflags flags="dom0_mem=512M"

a

rocks set host bootflags cluster flags="dom0_mem=1024M"

Tyto změny však mají vliv pouze na stav v lokální databázi na frontendu, pro použití těchto změn je nutné provést reinstall jednotlivých uzlů.

To provedeme takto:

Pomocí příkazu

rocks set host pxeboot compute-0-0 action="install"

sdělíme 1. uzlu, že má po restartu pokračovat režimem instalace. A provedeme samotný restart pomocí:

rocks run host compute-0-0 reboot.

Vyčkáme několik minut, znovu nainstalovaný stroj by se již měl zavést správně. Nyní pokud se stroj opravdu nastartoval správně, můžeme tuto akci provést i pro ostatní uzly.

8.2.5 Konfigurace vzdáleného přístupu

Cluster je tedy plně funkční. Nyní přistoupíme k jednomu z posledních kroků, umožníme vzdáleným uživatelům pracovat s clusterem a přistupovat na něj. K těmto účelům slouží služby ssh a vnc.

Službu ssh má server již nastavenou a spuštěnou, realizuje ji deamon sshd, ujistit se o jeho běhu můžeme pomocí příkazu

ps –A | sshd.

Službu vnc je nutné zapnout. Nejprve nastavme heslo pomocí příkazu

vncpasswd,

poté lze server spustit pomocí příkazu

vncserver.

Na všechny tyto služby je však nutno povolit přístup, který omezuje firewall iptables. Konfigurační soubor iptables obsahuje všechna pravidla a výjimky, podle kterých se firewall řídí. Jeho otevření lze provést příkazem:

nano /etc/sysconfig/iptables

a můžeme v něm provést několik změn. Já jsem jej pro testovací účely vypnul vyřazením z init skriptů.

Konfigurace vzdáleného přístupu je tímto dokončena, nyní se můžeme odhlásit od frontendu. Nadále se ke clusteru lze připojovat přes zabezpečenou konzoli ssh odkudkoliv bude potřeba, a z libovolného operačního systému (ve Windows slouží k připojení na ssh program putty)

8.2.6 Testování clusteru pomocí OpenMPI

Spouštění programů vytvořených pro běh na clusteru za pomoci knihovny MPI (v implementaci OpenMPI i MPICH), provádí příkaz

mpirun.

Balíčky OpenMIP a MIPCH jsou obsahem role hpc. Při instalaci jsem zvolil, že tuto roli chci nainstalovat, proto je příkaz mpirun připraven k používání.

Nejprve musí být vytvořen nový uživatel, pojmenoval jsem ho „mpiuser“, to provedeme příkazem

useradd mpiuser.

Z rootovského účtu provedeme

su mpiuser

a pomocí

passwd

změníme heslo tohoto nového uživatele.

Aby se tato změna promítla do konfigurace celého clusteru, musí být provedena synchronizace účtů pomocí

rocks sync users.

Po přihlášení jako mpiuser se musí vytvořit lokální konfigurační soubor se seznamem všech uzlů, kterých bude tento uživatel využívat. Pomocí editoru nano jsem vytvořil soubor *machines* obsahující řádky

compute-0-0

compute-0-1

compute-0-2

compute-0-3

Na základě návodu na stránkách [6] jsem napsal testovací program na výpočet Jacobiho matice (zdrojový kód, viz příloha A) a uložil jsem jej jako jacobi.c, napsal jsem Makefile soubor (viz příloha B) a pomocí příkazu

make

tento program zkompiloval. Po úspěšné kompilaci jsem jej spustil příkazem

mpirun –np 4 –hostfile machines jacobi.

Na webovém rozhraní Ganglia bylo možné sledovat, do jaké míry vytížil tento program který uzel. Potvrdilo se zde, že cluster je opravdu funkční, na výpočtu se podílely všechny uzly.

Po několika minutách běhu úlohy však jeden z uzlů přestal reagovat, což bylo patrné z rozhraní Ganglia (viz příloha D). Následoval pád celého clusteru a zastavení prováděné operace. Po ručním restartu inkriminovaného uzlu se automaticky uvedl do režimu instalace a reinstaloval se (výchozí nastavení Rocks – po použití násilného vypnutí nebo restartu uzlu). Na důvod výpadku uzlu jsem nepřišel.

8.2.7 Xen virtualizace

Jak je popsáno v předchozí kapitole, vizualizační nástroj Xen je také součástí distribuce rocks. Pro jeho správu, kromě rocks konzole, slouží program Virtual Machine Manager spustitelný příkazem virt-manager srceenshot viz Příloha C.

8.2.7.1 Instalace Debianu na virtuální stroj

Xen v distribuci Rocks je především určen k virtualizaci jednotlivých výpočetních uzlů, aby nedošlo k zastavení výpočtů při hardwarové závadě na některém z nich (viz další kapitola). Lze na něm však provozovat i jiný operační systém, pro test jsem vybral, pro mě důvěrně známou, distribuci Debian ve verzi testing i386 s jádrem 2.6.26. Kroky instalace:

- Stáhneme .iso soubor distribuce, kterou chceme instalovat.
- Spustíme Virtual Machine Manager.
- Připojíme se na localhost (dvojklikem), tím se povolí tlačítko „New“.

- Po stisku tlačítka „New“ se spustí průvodce konfigurace nového virtuálního stroje.
- Pojmenujeme jej (v mém případě „TestMachine“).
- Pokud máme vyhovující hardware a distribuci, vybereme paravirtualizaci, já jsem vybral plnou virtualizaci.
- Do položky ISO image location vložíme cestu ke staženému .iso souboru (podle druhu operačního systému zvolíme položky OS Type příp. OS Variant.
- Jako úložiště jsem zvolil soubor, tedy možnost Simple File.
- Vzhledem k absenci další síťové karty v počítači jsem v dalším kroku vybral možnost Virtual network.
- Zbývá jen nastavit kolik paměti a procesorových jader novému virtuálnímu stroji chceme přidělit.
- Tlačítkem finish spustíme instalaci.
- Dále pokračujeme podle instalačního průvodce vybrané distribuce.

8.2.7.2 Instalace virtuálního clusterového prostředí

Jak již bylo řečeno, role Xen v distribuci Rocks slouží především k realizaci virtuálního výpočetního clusteru, který má za úkol nedovolit případným hardwarovým problémům, aby přerušily průběh výpočtu.

Instalace probíhá v několika krocích.

Pomocí PXE bootu řízeného přes program insert-ethers nainstalujeme nové uzly jako VM Container (podrobnosti o instalaci uzlů pomocí nástroje insert-ethers a PXE bootování jsou v kapitole 8.2.3). Stejně tak jako v předchozím případě můžeme takto nainstalovaným uzlům upravit parametry bootování atd. pomocí konzole rocks. Já jsem se rozhodl pro realizaci virtuálního clusteru na nových strojích, nechal jsem tedy své výpočetní uzly běžet a obstaral jsem další 2 stroje Dell Optiplex s konfigurací:

- Uzly #4, #5
  - CPU: Intel Core 2 Duo E6550 (2x 2,33GHz)
  - RAM: 2 GB
  - HDD: 1x 250GB 7200ot./min.
  - 1x Gigabit ethernet

Po dokončení instalace těchto hostujících strojů se promocí konzole rocks vytvoří virtuální prostředí.

rocks add cluster fqdn="virtual.cluster.ef.jcu.cz"
ip="160.217.X.T" num-computes=2.

Zvolené jméno a IP adresa musí být zaznamenáno v /etc/hosts, to je nutné provést ručně. Stav clusteru si můžeme ověřit pomocí příkazu rocks list cluster, výsledek by měl být obdobný jako je v následující tabulce:

Tabulka 9 - Přehled o clusteru

FRONTEND	CLIENT NODES	TYPE
cluster.ef.jcu.cz	----------	physical
	vm-container-0-0	physical
	vm-container-0-1	physical
virtual.cluster.ef.jcu.cz	----------	VM
	hosted-vm-0-0-0	VM
	hosted-vm-0-1-0	VM

Nyní je vše připraveno a můžeme spustit virtuální frontend a nainstalovat ho. To provedeme příkazem:

rocks start host vm virtual.cluster.ef.jcu.cz install=yes

Je však třeba si dát pozor na nastavení paměti a nastavení parametrů dom0_mem, které stanovují velikost paměti, kterou si zabere hostující operační systém. Pokud tedy máme na frontendu nastavenou hodnotu dom0_mem na 1024M a frontend má jen 2GB fyzické paměti, zbývá nám pro virtuální fronend jen 1GB paměti. Pokud se však o tento prostor má dělit s dalšími virtuálními stroji (u mě debian), je třeba pomocí příkazů konzole rocks toto rozdělení upravit. V případě, že musíme tomuto virtuálnímu stroji přidělit méně než 768MB paměti

musíme upravit parametr dom0_min_mem na tuto hodnotu (cesta: */etc/xen/*xen-config.sxp*).

Pokud se start virtuálního frontendu podařil, můžeme otevřít virt-manager a dvojklikem na položku frontend-0-0 otevřít konzoli tohoto stroje. Pak už stačí projít instalační program podle návodu z kapitoly 8.2.1. S jednou výjimkou, role, které budou instalovány, již nelze získat z CD-Rom media, místo toho využijeme možnost Nework-based Rolls a do textového pole popsaného jako Hostname of Roll server napíšeme hodnotu stejnou, jako jsme napsali do parametru FQDN při vytváření virtuálního prostředí (u mě virtual.cluster.ef.jcu.cz). S tímto parametrem by měla korespondovat i položka Fully-Qualified Host Name.

Nyní máme nainstalovaný virtuální frontend (po chvíli čekání na dokončení instalace). A můžeme začít s instalací virtuálních uzlů.

Na virtuálním frontendu tedy spustíme utilitu insert-ethers a zvolíme hodnotu Compute, pak nastartujeme první virtuální uzel příkazem:

rocks start host vm hosted-vm-0-0-0 install=yes

V tomto okamžiku se objevila jedna z mnoha chyb verze 5.1, díky které se mi nepodařilo dokončit instalaci virtuálního prostředí. Jediné co jsem dokázal o této chybě zjistit, bylo to, že ve verzi 5.0 nebyla. Pro provoz virtuálního clusteru tedy doporučuji použít tuto starší verzi.

9 Závěr

Závěrem bych chtěl říci, že motivací ke studiu clusterových technologií je mnoho, stejně tak, jako je mnoho různých témat okolo clusterů, mnoho pohledů na problematiku. Existuje ohromné množství různých implementací, od velkých firem jako Microsoft, IBM, Oracle atd., zakázková řešení pro superpočítače, až po malé projekty mnohdy i jednotlivců.

Úkolem této práce nebylo udělat přehled všech těchto technologií, mnohdy jsou přehnaně náročné, drahé a vyžadují specifický hardware, nebo naopak staré neudržované a nepoužitelné. Úkolem bylo podat přehled o nejčastěji využívaných technologiích a hlavně o operačních systémech, na kterých je možné tyto technologie využívat.

Do hodnocení jsem schválně zapojil jeden komerční produkt (Windows HPC Server). Dle mého názoru by mnoho organizací raději zvolilo produkt, který je placený, avšak spolehlivý a svou ergonomií, jednoduchostí a propracovaností ušetří čas svým uživatelům i administrátorům. Zvláště v komerční sféře je tento názor nesporný.

Druhým úkolem bylo zprovoznit některé z dostupných Linuxových řešení a teoretické znalosti si na něm otestovat. Ačkoliv jsem vybral vítěze mého testu (když neuvažuji Windows HPC Server, který jsem použít nemohl), byl jsem výrazně nespokojen s dokumentací, stabilitou i spolehlivostí systému. Hodnocení těchto parametrů jsem

i přesto nezahrnul do testu, protože se mohou vyskytovat jen při různých konfiguracích a jen v určitých verzích. Jsou tedy neprokazatelné.

Pokud bych tedy znovu stál před volbou vhodného řešení pro realizaci clusteru, zcela jistě bych volil mezi komerčními produkty, především z důvodu výrazně kvalitnější podpory.

Pokud bych však měl doporučit zájemcům na poli akademickém či vědeckém některé z řešení, byl by to pronájem procesorového času na některém z gridových clusterů (např. [17]), odpadá tak nutnost vlastnit hardware a všechny problémy z toho plynoucí, jako je údržba, administrace, energie atd.

Citovaná literatura

[1] MULAČ, Miloš. *Výpočetní clustery v METACentru* [online]. 2004, 9. prosince 2004 [cit. 2008-12-13]. V češtině. Dostupný z WWW: <http://home.zcu.cz/~mulac/clusteryKMA.pdf>.

[2] SUCHÝ, Miroslav. *Root.cz : Úvod do virtualizace pomocí XENu* [online]. 2007, 8. 3. 2007 [cit. 2008-12-13]. Text v češtině. Dostupný z WWW: <http://www.root.cz/clanky/uvod-do-virtualizace-pomoci-xenu/>.

[3] PAŠKA, Marek. *Root.cz : Bezpečné programování ala Ada* [online]. 2003, 2. 10. 2003 [cit. 2008-12-17]. Český. Dostupný z WWW: <http://www.root.cz/clanky/bezpecne-programovani-ala-ada/>.

[4] ČÍRTEK, Pavel, RACEK, Stanislav, KOUTNÝ, Tomáš. *Stručný popis vláken POSIX* [online]. KIV FAV ZČU, c2007 [cit. 2008-12-17]. Český. Dostupný z WWW: <http://www.kiv.zcu.cz/studies/predmety/ppr/mat_pthread_navod.php>.

[5] LÖF, Henrik, RADOVIC, Zoran, HAGERSTEN, Erik. *THROOM - Supporting POSIX Multithreaded Binaries on a Cluster* [online]. Uppsala University, Department of Information Technology, c2008 , 2004-09-08 [cit. 2008-12-17]. Anglický. Dostupný z WWW: <http://www.it.uu.se/research/group/uart/publications/lof_2003_aug>.

[6] *Rocks Clusters. Rocks Clusters : About [online]. [2005] [cit. 2008-12-07]. Text v angličtině. Dostupný z WWW: <http://www.rocksclusters.org/wordpress/?page_id=57>.*

[7] Red Hat, Inc.. *CentOS : Red Hat Enterprise Linux Installation Guide* [online]. [2008] [cit. 2008-12-07]. Text v angličtině, volný překlad. Dostupný z WWW: <http://www.centos.org/docs/5/html/5.2/Installation_Guide/index.html>.

[8] *SourceForge.net : Open Source Tripwire* [online]. [2000], 2007 [cit. 2008-12-13]. Text v angličtině, volný překlad. Dostupný z WWW: <http://sourceforge.net/projects/tripwire>.

[9] *MpiBLAST : Open-Source Parallel BLAST* [online]. [2008] [cit. 2008-12-17]. Text v angličtině. Dostupný z WWW: <http://www.mpiblast.org/>.

[10] *Ganglia Monitoring System : What is Ganglia?* [online]. c2007 [cit. 2008-12-17]. Text v angličtině. Dostupný z WWW: <http://ganglia.info/>.

[11] *Ganglia Roll: Users Guide: : Chapter 3. Using the ganglia Roll* [online]. University of California, c2008 [cit. 2008-12-17]. Text v angličtině. Dostupný z WWW: <http://www.rocksclusters.org/roll-documentation/ganglia/5.1/using-ganglia.html>.

[12] *Fakulta informačních technologií VUT : Sun Grid Engine* [online]. Fakulta informačních technologií VUT, c2007 , 30. srpna 2007 [cit. 2008-12-17]. Text v češtině. Dostupný z WWW: <http://www.fit.vutbr.cz/CVT/linux/sge.html.cs.iso-8859-2>.

[13] *Base Roll: Users Guide* [online]. The Regents of the University of California, c2000-2008 [cit. 2008-12-17]. Text v angličtině. Dostupný z WWW: <http://www.rocksclusters.org/roll-documentation/base/5.1/index.html>.

[14] *Google* [online]. Dostupný z WWW: <http://www.google.com>.

[15] *DistroWatch* [online]. c2001, 26-12-2008 [cit. 2008-12-26]. Dostupný z WWW: <http://distrowatch.com/>.

[16] GOLDEN, Bernard. *XenExpress tutorial: Introduction and installation* [online]. [2007], 12. 20. 2007 [cit. 2008-12-30]. Text v angličtině. Dostupný z WWW: <http://searchservervirtualization.techtarget.com/tip/0,289483,sid94_gci1287016,00.html>.

[17] *METACentrum: superpočítače, výpočetní clustery, gridy* [online]. [2007], 2008-12-19 [cit. 2008-12-30]. Text v češtině. Dostupný z WWW: <https://meta.cesnet.cz/cs/>.

[18] KEMPE, Magnus. *Ada FAQ: Programing with Ada* [online]. c1994-1998 [cit. 2009-01-03]. Dostupný z WWW: <http://www.adahome.com/FAQ/programming.html>.

[19] KAMINSKY, Alan. *Parallel Java Library : Overview* [online]. c2005-2008, 24-Dec-2008 [cit. 2009-01-03]. Text v angličtině. Dostupný z WWW: <http://www.cs.rit.edu/~ark/pj.shtml>.

[20] MAMOUD, Qusay. *Sun Developer Network (SDN) : Getting Started With JavaSpaces Technology: Beyond Conventional Distributed Programming Paradigms* [online]. 2005, July 12, 2005 [cit. 2009-01-03]. Text v angličtině. Dostupný z WWW: <http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/>.

[21] *Jini : Jini Architecture Specification* [online]. [2006], 4 September 2006 [cit. 2009-01-03]. Text v angličtině. Dostupný z WWW: <http://www.jini.org/wiki/Jini_Architecture_Specification>.

[22] *Message Passing Interface Forum* [online]. [cit. 2009-01-03]. Text v angličtině. Dostupný z WWW: <http://www.mpi-forum.org/>.

[23] BURDA, Michal. *Root.cz : Povídky paralelistické: programování pro paralelní počítače* [online]. 2003, 21. 8. 2003 [cit. 2009-01-03]. Text v češtině. Dostupný z WWW: <http://www.root.cz/clanky/povidky-paralelisticke-programovani-pro-paralelni-pocitace-/>.

[24] *An Overview of Xen Virtualization* [online]. c2005 , August 2005 [cit. 2009-01-04]. Text v angličtině. Dostupný z WWW: <www.dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf >.

Příloha A

```
#include <stdio.h>
#include <math.h>
#include "mpi.h"
/* This example handles a 12 x 12 mesh, on 4 processors only. */
#define maxn 1000

int main( argc, argv )
int argc;
char **argv;
{
   int       rank, value, size, errcnt, toterr, i, j, itcnt;
   int       i_first, i_last;
   MPI_Status status;
   double    diffnorm, gdiffnorm;
   double    xlocal[(1000/4)+2][1000];
   double    xnew[(1000/3)+2][1000];
   MPI_Init( &argc, &argv );
   MPI_Comm_rank( MPI_COMM_WORLD, &rank );
   MPI_Comm_size( MPI_COMM_WORLD, &size );
   if (size != 4) MPI_Abort( MPI_COMM_WORLD, 1 );
   /* xlocal[][0] is lower ghostpoints, xlocal[][maxn+2] is upper */
   /* Note that top and bottom processes have one less row of interior
      points */
```

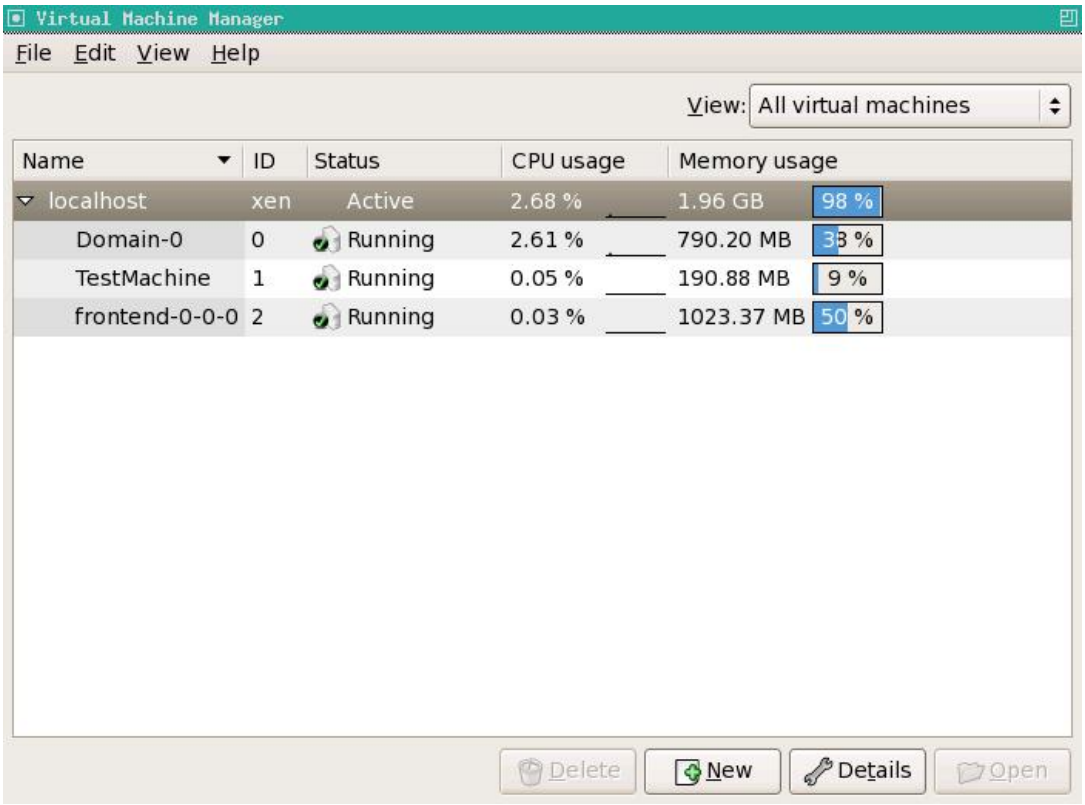
```
i_first = 1;
i_last  = maxn/size;
if (rank == 0)        i_first++;
if (rank == size - 1) i_last--;
/* Fill the data as specified */
for (i=1; i<=maxn/size; i++)
        for (j=0; j<maxn; j++)
           xlocal[i][j] = rank;
for (j=0; j<maxn; j++) {
        xlocal[i_first-1][j] = -1;
        xlocal[i_last+1][j] = -1;
}
itcnt = 0;
do {
        /* Send up unless I'm at the top, then receive from below */
        /* Note the use of xlocal[i] for &xlocal[i][0] */
        if (rank < size - 1)
           MPI_Send( xlocal[maxn/size], maxn, MPI_DOUBLE,
rank + 1, 0, MPI_COMM_WORLD );
        if (rank > 0)
           MPI_Recv( xlocal[0], maxn, MPI_DOUBLE, rank - 1, 0,
                     MPI_COMM_WORLD, &status );
        /* Send down unless I'm at the bottom */
        if (rank > 0)
           MPI_Send( xlocal[1], maxn, MPI_DOUBLE, rank - 1, 1,
```

```
                    MPI_COMM_WORLD );
        if (rank < size - 1)
           MPI_Recv( xlocal[maxn/size+1], maxn, MPI_DOUBLE,
rank + 1, 1,MPI_COMM_WORLD, &status );
        /* Compute new values (but not on boundary) */
        itcnt ++;
        diffnorm = 0.0;
        for (i=i_first; i<=i_last; i++)
           for (j=1; j<maxn-1; j++) {
                xnew[i][j] = (xlocal[i][j+1] + xlocal[i][j-1] +
                             xlocal[i+1][j] + xlocal[i-1][j]) / 4.0;
                diffnorm += (xnew[i][j] - xlocal[i][j]) *
                          (xnew[i][j] - xlocal[i][j]);}
        /* Only transfer the interior points */
        for (i=i_first; i<=i_last; i++)
           for (j=1; j<maxn-1; j++)
                xlocal[i][j] = xnew[i][j];
        MPI_Allreduce( &diffnorm, &gdiffnorm, 1, MPI_DOUBLE,
MPI_SUM, MPI_COMM_WORLD );
        gdiffnorm = sqrt( gdiffnorm );
        if (rank == 0) printf( "At iteration %d, diff is %e\n", itcnt,
gdiffnorm );
   } while (1==1);
   MPI_Finalize( );
   return 0;
}
```

Příloha B

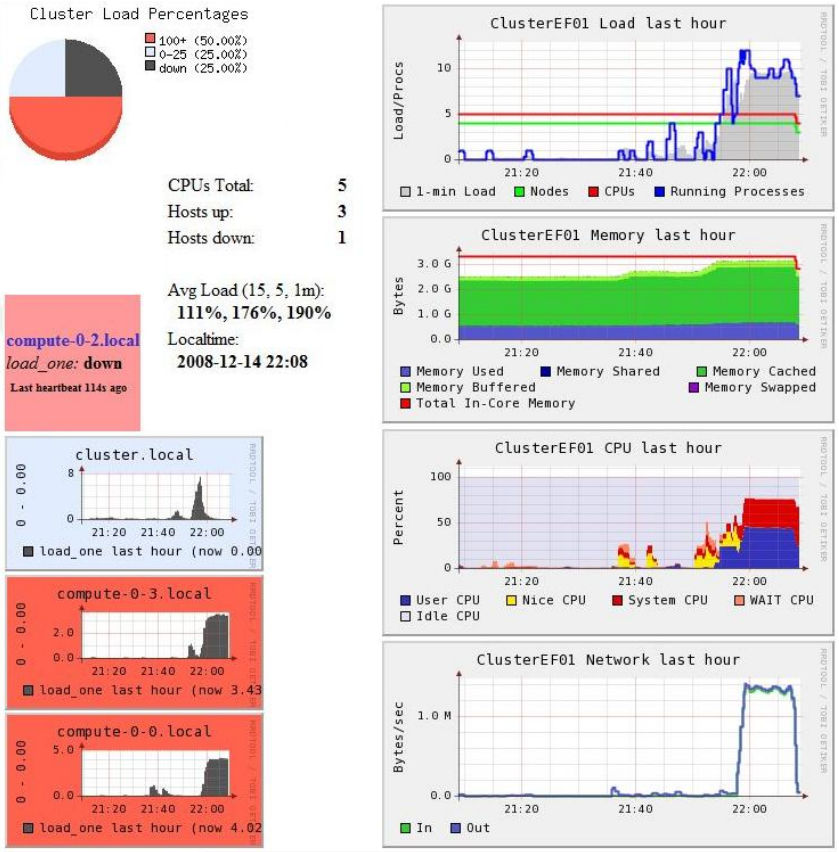
```
ALL: jacobi
SHELL = /bin/sh
DIRS =
jacobi: jacobi.c
	mpicc -o jacobi jacobi.c -lm
profile.alog: jacobi.c
	mpicc -o jacobi.log -mpilog jacobi.c -lm
	mpirun -np 4 jacobi.log
	/bin/mv jacobi.log_profile.log profile.alog
clean:
	/bin/rm -f jacobi jacobi.o jacobi.log
```

Příloha C



Obrázek 1 - Virtual Machine Manager

Příloha D



Ganglia Node Image Legend	
Node Image	**Meaning**
Red	Over 100% Utilization. Utilization is: (1 min load) / (number of CPUs) * 100%.
Orange	75-100%
Yellow	50-74%
Green	25-49%
Blue	0-24%
Crossbones	The node is dead. We consider a node dead when the reporting node has not heard from it in 60 sec.

Obrázek 2 - Upravené rozhraní Ganglia