Typy clusterů

Pod pojmem *cluster* rozumíme několik víceméně samostatných počítačů (v tomto kontextu nazývaných *uzly*) propojených do jednoho funkčního celku. Clustery lze nasazovat v několika typech úloh. Podle toho se také liší požadavky na hardwarovou a softwarovou konfiguraci celého clusteru.

Podle typu úloh dělíme clustery na tyto kategorie:

- **Výpočetní clustery** (high-performance computing). Jsou to obvykle počítače, propojené za účelem provádění distribuovaných paralelních výpočtů.
- **Clustery rozdělující zátěž** (load balancing). Slouží obvykle jako servery pro zvýšení výkonu síťových aplikací. Z hlediska klienta se celý cluster tváří jako jeden server.
- **Clustery pro vysokou dostupnost aplikací** (high availabilty). Více počítačů je zde připraveno vykonávat určitou službu, v případě havárie jednoho z nich přebírá jeho funkci další.

Výpočetní clustery

Pro urychlení výpočtů lze použít více počítačů, spojených do jedné výpočetní sítě. Podle typu úlohy lze při malé komunikační náročnosti použít i skupinu počítačů, spojených lokální sítí (*seti@home*, atd.). Pro Linuxové výpočetní clustery se používá souhrnné označení *Beowulf* podle stejnojmenného projektu NASA, který realizoval první výpočetní cluster a některé podpůrné části v jádře systému. Nejde o označení konkrétního softwaru. Pro komunikačně náročnější úlohy je obvyklé použít dedikovanou síť a jeden *front-end server*, který je zapojen i do Internetu. Ostatní *výpočetní uzly* pak obvykle nemají připojení do Internetu, systém zavádějí z front-end serveru a často nemají ani vlastní disk. U vysokovýkonných výpočtů je obvyklé, že se nepočítá se swapováním a úlohy pro jednotlivé uzly jsou dimenzovány tak, aby mohly běžet celé v operační paměti.

Síťové prostředky

Dedikovaný výpočetní cluster mívá své uzly obvykle propojené některou z následujících technologií:

- **100 Mbps ethernet, všechny uzly v jednom switchi.** V případě výkonného switche jde o dostatečně výkonné řešení z hlediska propustnosti, méně však z hlediska latence. Front-end server pak může být připojen rozhraním o vyšší kapacitě – gigabitovým ethernetem nebo několika 100 Mbps rozhraními spojenými do jednoho logického rozhraní (bonding driver).
- **100 Mbps ethernet, topologie hyperkostka.** Každý uzel má pak několik ethernetových rozhraní, které jej spojují s několika sousedními uzly. Toto řešení má menší latenci, ale obvykle neexistuje přímé spojení pro některé dvojice uzlů.
- **Myrinet, HIPPI nebo jiné specializované rozhraní.** Tato technologie může být řádově rychlejší, než 100 Mbps ethernet, ale hlavně má daleko nižší latenci, která mimo jiné spočívá v tom, že síťové rozhraní není obaleno silnou vrstvou protokolů TCP/IP. Další výhodou je to, že v případě Myrinetu umí hardware příjemce přímo zapisovat přijatá data do paměti procesu, takže zasílání zpráv mezi výpočetními uzly je rychlejší. V případě tohoto řešení mívají uzly ještě další síťové rozhraní TCP/IP (například ethernet), které slouží ke sdílení diskového prostoru, vzdálené konfiguraci, spouštění úloh a podobně.

Mosix

Jedním z prostředků pro paralelizaci aplikací je systém Mosix. Je to systém transparentní migrace procesů na uzly clusteru (tzv. *fork and forget*). Nové procesy jsou transparentně přesunovány na jiné uzly clusteru. I/O operace pak provádějí na původním uzlu, kde vznikly. Mosix umožňuje několik typů uzlů v clusteru. Například uživatelské pracovní stanice mohou procesy pouze generovat, ale nesmí na ně migrovat procesy jiných uživatelů, než uživatele příslušné stanice. Servery potom mohou přijímat procesy všech uživatelů.

Mosix je jednoduchý systém, ale pro většinu aplikací dostačující. Nepodporuje sdílenou paměť a tedy ani migrace multithreadových aplikací (například Java). Pokud jsou ale k dispozici paralelní verze aplikací, je lepší použít jinou technologii.

DIPC

Distributed IPC je systém, který mezi uzly clusteru implementuje sdílenou paměť a další synchronizační prostředky za pomocí protokolů TCP/IP. Toto je nejsnazší cesta k přenosu vícevláknových aplikací na clusterové prostředí. Nicméně běh aplikací, které předpokládají sdílenou paměť a nepočítají s vyšší časovou náročností nad DIPC, pak může být suboptimální.

PVM

Parallel Virtual Machine je nástroj na vytvoření distribuovaného výpočetního stroje s předáváním zpráv a distribuovanou synchronizací procesů. PVM má podporu pro předávání zpráv mezi různými platformami (konverze endianity a jiných datových formátů). Jedná se o virtuální stroj s možností dynamické rekonfigurace (přidávání a ubírání dalších uzlů za běhu aplikace). Dále je zde možnost přizpůsobení komunikačních vrstvy PVM konkrétní topologii clusteru, takže například v případě, kdy neexistuje přímé spojení mezi všemi uzly výpočetní sítě, můžou mezilehlé uzly transparentně dělat směrování zpráv k cílovým uzlům.

MPI

Druhým často používaným prostředím kromě PVM je *Message passing interface*. Jde o API pro komunmikaci mezi výpočetními kontexty. MPI má různé implementace pro různé výpočetní architektury. Pro Linuxové clustery se nejčastěji používá implementace MPICH. MPI neřeší start jednotlivých procesů na uzlech výpočetní sítě.

Load-balancing clustery

Druhým typem clusterů je spojení více počítačů do jednoho celku za účelem rozdělení zátěže. V tomto případě se cluster tváří jako síťový server, na jednotlivé uzly clusteru jsou přesměrovány požadavky od různých klientů.

Pro rozdělení zátěže se používá nejčastěji nástroj s názvem *ipvs*. Tento prostředek se skládá ze dvou částí: modulu do jádra systému a obslužného programu `ipvsadm`. Load balancing cluster pak obvykle funguje tak, že v systému je jeden *rozdělovací server*, který má IP adresu, odpovídající příslušné síťové službě, realizované clusterem (WWW server, FTP server a podobně). Pomocí modulu *ipvs* pak tento rozdělovací server přijímá jednotlivé požadavky a přeposílá je na *pracovní servery* clusteru k vyřízení.

Rozdělování požadavků v IPVS

Jádro má několik metod, jak rozdělovat požadavky na jednotlivé servery:

- **Round Robin** – požadavky jsou rovnoměrně rozdělovány na reálné servery.
- **Weighted Round Robin** – jednotlivým serverům je navíc přiřazena váha dle výkonnosti nebo očekávané zátěže.
- **Least Connection** – další požadavky jsou přiřazeny serverům, které zpracovávají nejméně aktivních požadavků.
- **Weighted Least Connection** – podobné jako předchozí, serverům může být přiřazena váha.

Rozdělovač spojení navíc podporuje persistenci, to jest po nějakou dobu jsou požadavky z jedné IP adresy směrovány vždy na tentýž pracovní server. Dokonce umí granularitu vyšší než jen jedna IP adresa, tedy směrovat požadavky z jedné sítě na tentýž server. Toto by pak pomohlo v případě, že aplikace potřebuje udržovat například nad HTTP nějaké stavové informace. Na druhou stranu pokud povolíte persistenci, možná zjistíte, že 40% požadavků pochází z jedné velké proxy cache...

Přeposílání požadavků

Další věc, kterou je potřeba při konfiguraci load-balancing clusteru řešit, je způsob, jakým se od rozdělovače zátěže dostanou datagramy k pracovním serverům. IPVS podporuje tři metody:

- **Direct routing.** V této metodě jsou datagramy bez modifikace přeposílány na cílový server. Z toho plynou jistá omezení: protože datagram (konkrétně cílová adresa) není změněn, musí být rozdělovač zátěže i servery na stejné síti. Jinak by na routerech už nebylo známo, na který server se má datagram přeposlat. Další problém tkví v tom, že cílový server musí tento datagram přijmout za svůj, i když cílová adresa není jeho vlastní. Toto lze řešit dvěma způsoby: Udělat alias síťového rozhraní se sdílenou adresou nebo přesměrovat příslušná spojení na sebe sama mechanismem transparentní proxy v `ipchains` nebo `iptables`. První přístup je o něco rychlejší, ale je zde problém: Linux posílá odpovědi na dotazy ARP do všech rozhraní, nejen do rozhraní, které má příslušnou adresu. Pak tedy může nastat situace, kdy router dostane odpověď ARP na sdílenou adresu nikoli od rozdělovacího serveru, ale od pracovních serverů. Existují záplaty na jádro, které toto chování potlačují. Jinak ale považuji za lepší použít mechanismus transparentní proxy. Metoda direct routing je velmi výhodná v tom, že odchozí datagramy jdou z pracovních serverů přímo na směrovač se správnou zdrojovou adresou, a tedy negenerují další zátěž na rozdělovacím serveru. Toto je dobré u protokolů typu HTTP nebo FTP, kde většina dat teče od serveru ke klientovi.
- **Tunelling.** Datagramy jsou na rozdělovacím serveru zapouzdřeny protokolem IPIP a poslány na pracovní server IP tunelem. Výhoda tohoto přístupu je, že pracovní servery nemusí být na stejné síti, a tedy je možno omezit možnost výpadku i tím, že servery clusteru jsou od sebe geograficky vzdáleny.
- **Network Address Translation.** Překlad adres je další možný přístup. Rozdělovací server přeloží cílovou adresu na adresu jednoho z pracovních serverů. Odchozí packety přeloží opačným způsobem (zdrojovou adresu přepíše na svoji adresu). Tento přístup je použitelný v případě, kdy rozdělovací server je zároveň směrovačem, který připojuje do Internetu ostatní servery clusteru. Potenciální nevýhodou je, že ne všechny protokoly snášejí dobře překlad adres. Nicméně IPVS využívá modul `ip_conntrack` z netfilteru pro sledování spojení, a tedy zvládá překládat i datová spojení protokolu FTP, UDP provoz nebo ICMP zprávy, které souvisejí s TCP spojením.

Sledování dostupnosti

Máme-li load-balancing cluster, je potřeba sledovat, které z možných pracovních serverů jsou dostupné. K tomu slouží několik možných řešení.

- **ldirectord** – démon, který sleduje jednotlivé pracovní servery podle své konfigurace. Pokud jsou dostupné, přidá je do tabulky modulu IPVS. Pokud nejsou dostupné, zase je z tabulky zruší. Pokud není ani jeden ze serverů dostupný, přesměruje požadavky na záložní server (v případě protokolu HTTP může například záložní server odpovídat ve stylu „momentálně není služba v provozu, zkuste to později“). Program `ldirectord` je psaný v Perlu a je poměrně dobře rozšiřitelný. V základní distribuci obsahuje podporu pro sledování protokolů HTTP a FTP, dopsat HTTPS není složité. Program posílá testovací požadavek a v odpovědi očekává klíčové slovo. V případě, že je najde, považuje příslušný pracovní server za funkční. Pomocí protokolu `rcmd` umí zjišťovat zátěž jednotlivých pracovních serverů a podle ní průběžně upravovat váhy v tabulce IPVS v jádře.
- **keepalived** – v podstatě podobný program, jen je psaný v C. Je více modulární a provádí kontrolu na více úrovních (3., 4., a 5. vrstva). Na 3. vrstvě posílá ICMP zprávu (ICMP echo request), pokud dostane odpověď, postupuje k vyšším vrstvám. Obsahuje podporu pro HTTP a HTTPS, další je možno dopsat.

High availability

V případě clusterů pro rozdělení zátěže zde byla jakási redundance (cluster může fungovat i po výpadku několika pracovních serverů), ale stále ještě je kritické místo (single point of failure) v rozdělovacím serveru, který používá IPVS. Jedno z řešení v tomto případě je mít takové servery dva a nakonfigurovat je tak, aby v případě výpadku jednoho z nich druhý převzal jeho povinnosti. Toto se může dít několika způsoby:

- **MAC address takeover** – při této metodě potřebujete fyzickou (MAC) adresu, která se nevyskytuje na lokální síti. Dále nakonfigurujete jeden server tak, aby tuto adresu považoval za svoji (je potřeba mít síťovou kartu, která umí mít více MAC adres nebo aspoň změnit svoji MAC adresu). Tato metoda je poměrně náročná na korektní implementaci, nicméně zajistí téměř okamžité převzetí IP adresy bez ohledu na možné rozdíly v implementacích ARP cache na směrovačích.
- **IP address takeover** – zde si oba dva servery předávají jednu sdílenou adresu pomocí IP aliasingu. Pokud záložní server zjistí, že primární server není funkční, aktivuje alias pro sdílenou adresu a pošle do sítě ARP odpověď pro tuto adresu, takže směrovače mají možnost změnu zaregistrovat téměř okamžitě (ne u všech typů se to podaří, ale možnost tu je).
- **DNS name takeover** – v případě výpadku dojde k překonfigurování DNS na jinou IP adresu. Tuto metodu lze využít pouze pro zajištění vysoké dostupnosti sítě, která je připojená přes více linek přes různé ISP (a tedy přes různé autonomní systémy). Toto řešení ale nezajistí dostatečně rychlé převzetí záložním serverem a v zásadě je nelze doporučit.

Pro zajištění kontroly dostupnosti se obvykle používá démon *heartbeat*. Umožňuje sledovat dostupnost druhého stroje pomocí různých metod – přes ARP, TCP, UDP, po sériové lince a podobně. V případě výpadku aktivního stroje pak umožní resetovat systém pomoci zařízení watchdog. Tento démon implementuje metodu IP address takeover.

Další nástroje

Pro budování clusterů jsou někdy potřebné i další nástroje. U aplikací, které potřebují sdílená data mezi uzly clusteru, je nutné mít sdílený souborový systém. V případě výpočetních clusterů obvykle stačí sdílet systém přes NFS s řídícím počítačem clusteru. V případě rozdělení zátěže nastává obvykle problém, jak udržet jednotlivé servery s konzistentní konfigurací. Zde lze doporučit synchronizaci souborových systémů protokolem *rsync*. Pokud jednotlivé uzly clusteru potřebují opravdu sdílená data v reálném čase, nezbude než použít specializované prostředky (nebo specializované databáze, například Oracle parallel server nebo replikaci v MySQL).

Intermezzo

Jde o distribuovaný souborový systém s možností replikace, odpojené práce, persistentní cache a dalších vlastností. Ideově vychází ze souborového systému Coda a je vyvíjen jedním z jeho autorů.

Global File System

GFS je souborový systém pro clustery. V tomto souborovém systému se počítá s tím, že k jednomu úložnému zařízení (disk, diskové pole, atd.) je přímo (přes sdílené SCSI, přes Fibrechannel nebo jinou *storage area network*) připojeno více počítačů. Souborový systém GFS umožní všem uzlům clusteru, aby přistupovali k diskovému prostoru jako k lokálnímu svazku. Podporuje žurnálování prováděných operaci a synchronizaci více serverů. Je plně symetrický, takže zde není jeden centrální server, který by byl úzkým místem z hlediska výkonu nebo kritickým místem z hlediska dostupnosti.

Piranha

Jde o několik programů pro vytváření load-balancing a high-availability clusterů. Jednotlivé moduly tohoto systému obsahují obdoby programů `ldirectord` (s dynamickým rozdělováním požadavků podle zátěže pracovních serverů) a `heartbeat`. Dále Piranha obsahuje grafický nástroj na konfiguraci clusterů a jednotný konfigurační soubor, který všechny moduly využívají. Systém je vyvíjen firmou Red Hat software.

Příklady nasazení výpočetních clusterů

Asi největším clusterem s rozdělením zátěže je Internetový vyhledávač *Google* – používá okolo 6000 počítačů s Linuxem, z nichž některé slouží přímo pro vyhledávání, jiné stahují z Internetu stránky a jiné dělají indexování stažených stránek.

Zřejmě největším výpočetním clusterem je *Cplant cluster Alaska* ze Sandia National Laboratories. Tento cluster obsahuje 580 pracovních stanic s procesorem Alpha, propojených sítí Myrinet. Operační systém je Linux/Alpha. Teoretický špičkový výkon je 580 GFlops, výkon při testu Linpack je 232.6 GFlops. Tento systém je v současné době 84. nejsilnější superpočítač na světě (podle seznamu Top500).

Na Fakultě informatiky Masarykovy univerzity jsme vyzkoušeli nasazení Linuxového clusteru pro rozdělení zátěže. Na clusteru běží informační systém univerzity. Cluster obsahuje jeden rozdělovací stroj (CPU AMD Athlon 700 MHz) – tento kromě rozdělování zátěže vykonává ještě některé další činnosti, šest pracovních serverů (AMD Athlon 800 MHz) – zde běží Apache a skripty v Perlu a jeden databázový server (Sun E450 se čtyřmi procesory UltraSparc, OS Solaris). Cluster používá metodu direct routing pro směrování na pracovní servery.

Odkazy

Beowulf
http://www.beowulf.org/

Cplant cluster
http://www.cs.sandia.gov/cplant/

Distributed IPC
http://wallybox.cei.net/dipc/dipc.html

Global File System
http://www.sistina.com/gfs/

Google
http://google.com/

Heartbeat
http://www.linux-ha.org/

Informační systém MU
http://is.muni.cz/

Intermezzo
http://www.inter-mezzo.org/

IP virtual server
http://www.linuxvirtualserver.org/

Keepalive daemon
http://keepalived.sourceforge.net/

Ldirector daemon
http://ultramonkey.sourceforge.net/ultramonkey-1.0.1/ldirectord.html

Message Passing Interface
http://www-unix.mcs.anl.gov/mpi/index.html

MOSIX

http://www.mosix.cs.huji.ac.il/

MPICH

http://www-unix.mcs.anl.gov/mpi/mpich/

Netfilter

http://netfilter.kernelnotes.org/

Piranha

http://www.redhat.com/support/wpapers/piranha/

Parallel Virtual Machine

http://www.epm.ornl.gov/pvm/pvm_home.html

Síť Myrinet

http://www.myrinet.com/

500 nejrychlejších superpočítačů

http://www.top500.org/