PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Výpočetní klastr na učebně

2013 Lukáš Beran

Anotace

Výsledkem práce (na vlastní téma) je výpočetní klastr složený z pracovních stanic na jedné z počítačových učeben katedry informatiky. Řešení spočívá v implementaci a rozšíření zvoleného software Microsoft HPC Pack 2008 R2 realizujícího výpočetní klastr nad množinou počítačů a řídícím serverem o terminálové textové ovládací rozhraní a o služby automatického plánování dostupnosti počítačů pro klastr na základě rozvrhu výuky na učebně, monitorování a optimalizaci vytížení uzlů klastru výpočetními úlohami a webové rozhraní pro základní práci s výpočetními úlohami klastru.

Děkuji Mgr. Janu Outratovi, Ph.D. za odborné vedení mé práce a cenné připomínky na konzultacích. Dále děkuji Ing. Jaroslavu Maňasovi a Ing. Lukáši Vojáčkovi za poskytnutí odborných konzultací, a samozřejmě i mé rodině a přítelkyni za trpělivost.

Obsah

1. Úvod **8**

2. Základní informace o klastrech **8**
2.1. Základní vlastnosti 9
2.2. Využití klastrů 9
2.3. Typy klastrů 9
2.3.1. Výpočetní klastr 9
2.3.2. Klastr s vysokou dostupností 10
2.3.3. Klastr s rozložením zátěže 10
2.3.4. Úložný klastr 10
2.3.5. Gridový klastr 11
2.4. Implementace klastrů 11
2.4.1. MPI aplikace 12

3. Technologie klastrů a virtualizace **16**
3.1. Xen 16
3.1.1. Typy hostů 17
3.2. Citrix XenServer 17
3.2.1. Vlastnosti Citrix XenServeru 18
3.3. Red Hat Enterprise Virtualization 18
3.4. VMware 18
3.5. Microsoft Hyper-V 19

4. Dostupná řešení výpočetních klastrů **19**
4.1. Red Hat Enterprise Linux for Scientific Computing 19
4.1.1. Verze Red Hat Enterprise Linux for Scientific Computing . 20
4.1.2. Základní vlastnosti 20
4.2. ParaTools HPC Linux 20
4.3. SUSE Linux Enterprise Server 21
4.4. PelicanHPC 21
4.5. Microsoft HPC 21
4.5.1. Edice Microsoft HPC 22
4.5.2. Systémové požadavky Microsoft HPC 22
4.5.3. Hardwarové požadavky Microsoft HPC 23
4.5.4. Vlastnosti Microsoft HPC 23

5. Implementace klastru s využitím Microsoft HPC **25**
5.1. Konfigurace sítě 26
5.2. Poskytnutí přihlašovacích údajů správce 27
5.3. Konfigurace jmen výpočetních nodů 27
5.4. Vytvoření šablony pro nody 28

5.5. Instalace Microsoft HPC Pack na stanice v učebně 29
5.6. Přidání výpočetních nodů do klastru 29
5.7. Instalace a konfigurace webového rozhraní 30
5.8. Přidání oprávněných uživatelů a skupin 30
5.9. Dostupnost výpočetních nodů 31
5.10. Instalace SSH serveru 32
5.11. Nastavení systémové proměnné PATH 32
5.12. Řešení hibernace úloh na klastru 33

6. Uživatelská dokumentace 34
6.1. Probuzení výpočetních nodů 35
6.2. Ovládání klastru pomocí HPC Job Manager 35
6.2.1. Vytvoření nové výpočetní úlohy 35
6.2.2. Zobrazení stavu výpočetní úlohy 37
6.2.3. Úprava výpočetní úlohy 37
6.2.4. Zrušení výpočetní úlohy 37
6.3. Ovládání klastru pomocí PowerShellu 37
6.3.1. Kopírování dat 39
6.3.2. Vytvoření nové výpočetní úlohy 39
6.3.3. Přidání výpočetního procesu k úloze 40
6.3.4. Spuštění výpočetní úlohy 40
6.3.5. Vytvoření jednoduché výpočetní úlohy 40
6.3.6. Zjištění informací o výpočetní úloze 41
6.3.7. Zrušení výpočetní úlohy 41
6.3.8. Zjištění stavu a vytíženosti klastru 41
6.3.9. Zobrazení informací o výpočetních nodech 42
6.4. Ovládání klastru pomocí webového portálu 42
6.4.1. Vytvoření nové výpočetní úlohy 43
6.4.2. Zobrazení stavu výpočetní úlohy 43
6.5. Příklad spuštění MPI úlohy na klastru 43
6.5.1. Spuštění úlohy pomocí skriptů v PowerShellu 44
6.5.2. Spuštění úlohy pomocí HPC Job Manageru 45
6.6. Vytváření vlastních MPI úloh 46
6.7. Šablony výpočetních úloh 46
6.8. Další informace 46

7. Porovnání výsledků výpočtů na klastru 47

Závěr 50

Reference 51

A. Referenční příručka – HPC Job Manager 53

B. Referenční příručka – HPC PowerShell **59**

C. Referenční příručka – webový portál **72**

D. Obsah přiloženého DVD **74**

Seznam obrázků

1. Sktruktura MPI aplikace 13
2. HPC Job Manager 36
3. Windows HPC Server Web Portal 43
4. Výpočet čísla π s 1 000 000 000 iterací 48
5. Tvorba a naučení neuronové sítě typu SOM 48

1. Úvod

Katedra informatiky disponuje celkem čtyřmi počítačovými učebnami, z toho je jedna složena z 18 výkonných pracovních stanic. V rámci výzkumu pracovníků katedry se provádí mnoho vědeckých výpočtů, pro které by vysoká výpočetní síla měla opodstatnění. Katedra však takovou výpočetní silou nedisponuje, proto jsem se rozhodl využít výkon počítačů na jedné z učeben a vytvořit na této učebně počítačový klastr, který by zpřístupnil nevyužitý výpočetní výkon pro vědecké účely v době, kdy na učebně neprobíhá výuka.

Výsledkem bakalářské práce je výpočetní klastr, který využívá pracovní stanice na počítačové učebně 5.003. Základem výpočetního klastru je Microsoft HPC Pack 2008 R2. Jako hlavní (řídící) server byl zvolen Windows Server 2008 R2 HPC Edition, který zajišťuje obsluhu celého výpočetního klastru.

Vlastní téma práce jsem si zvolil z důvodu, že technologie Microsoftu jsou mi blízké a současně klastrovací nástroje jsou pro mě zajímavým tématem, ve kterém jsem si chtěl rozšířit své znalosti. Při výběru tématu jsem se zaměřoval také na to, aby téma mělo praktický význam a bylo dlouhodobě využitelné a přínosné, což doufám, že výpočetní klastr splní a bude hojně využívaný pro vědecké výpočty pracovníků katedry. Zadání a řešení práce tedy spočívá ve vytvoření a přípravě klastru na Katedře informatiky Univerzity Palackého v Olomouci. Vytvořený výpočetní klastr je provozován v pro klastry netypickém prostředí, jelikož dostupnost klastru musí respektovat výuku na učebně a klastr se tedy vypíná v době, kdy na učebně probíhá výuka. Součástí práce je také demonstrace možností tohoto klastru na jednoduché MPI úloze. Vytvoření vlastní složitější MPI úlohy s podrobným rozborem je nad rámec zadání i řešení této práce.

Výpočetní klastr umožňuje ovládání pomocí grafického nástroje ze vzdálené plochy řídícího serveru nebo pokročilou správu přes PowerShell, pomocí kterého jsem nachystal i sadu 15 předpřipravených skriptů, které je možné dále rozšiřovat a upravovat. Pro základní správu a monitorování výpočetních úloh je možné využít i webové rozhraní.

2. Základní informace o klastrech

Počítačový klastr se skládá z množiny volně propojených počítačů, které pracují dohromady tak, že v mnoha ohledech mohou být považovány za jeden systém. Počítače v klastru (nody klastru) jsou obvykle propojeny rychlou lokální počítačovou sítí a každý počítač v klastru má vlastní instanci operačního systému.

Počítačové klastry bývají nasazovány pro zvýšení výpočetní rychlosti nebo spolehlivosti s vyšší efektivitou, než by mohl poskytnout jediný počítač, přičemž jsou levnější než jediný počítač o srovnatelné rychlosti nebo spolehlivosti.

2.1. Základní vlastnosti

Počítačový klastr obvykle spojuje počítače pomocí rychlé lokální počítačové sítě. Celý počítačový klastr je řízený klastrovacím middleware[1], což je softwarová vrstva na nodech klastru, která umožňuje uživatelům přistupovat ke klastru jako k jednomu celku, například pomocí single system image konceptu[2].

Klastrování se opírá o přístup centralizované správy, která zpřístupňuje nody klastru jako řízené sdílené servery. Přístup centralizované správy je důležitým rozdílem například od peer-to-peer[3] přístupu, který také využívá více nodů.

Počítačový klastr může být jednoduché spojení dvou počítačů nebo se může jednat o propojení tisíců počítačů do jednoho klastru. Základním typem klastru je Beowulf klastr, který spojuje několik osobních počítačů do ekonomicky výhodné alternativy vysoce výkonných počítačů.

2.2. Využití klastrů

Klastry slouží k paralelním výpočtům složitých početních úloh (např. faktorizace na prvočísla, simulace vývoje počasí, analýza velkého množství statistických dat) nebo se používají pro zajištění vysoké dostupnosti určité služby (např. databáze, webové služby), případně k rozložení zátěže. Používají se buď specializované víceprocesorové stroje propojené sítí nebo obyčejné počítače třídy PC.

2.3. Typy klastrů

Pro různé potřeby existují různé typy počítačových klastrů.

2.3.1. Výpočetní klastr

Výpočetní klastr (anglicky High-performance computing, zkratka HPC) slouží ke zvýšení výpočetního výkonu pomocí více počítačů, které na výpočtu spolupracují. Obvykle jsou použity počítače střední cenové hladiny propojené pomocí vysokorychlostní počítačové sítě. Tímto způsobem vznikne vysoce výkonný celek, který je mnohonásobně levnější než jeden vysoce výkonný počítač.

Výpočetní klastr typicky využívá osobní počítače stejné konstrukce, které jsou běžně používány pro desktop nebo servery. Výpočet je pak prováděn pomocí paralelizace, která umožňuje rozdělit řešení na mnoho nezávislých úloh. Proto musí být při programování využívány speciální postupy.

Výpočetní výkon je mezi klastry porovnáván pomocí jednotky FLOPS[4].

[1] Software poskytující služby, které neposkytuje operační systém
[2] Více strojů, které vypadají jako jeden systém
[3] Přímé spojení počítačů
[4] Počet operací s plovoucí řádovou čárkou za jednu sekundu

Alternativou k výpočetním klastrům jsou gridové klastry.

2.3.2. Klastr s vysokou dostupností

Klastr s vysokou dostupností (anglicky High-availability cluster nebo Failover cluster) zajišťuje pomocí několika redundantních počítačů nepřetržité poskytování dané služby i při výpadku některého počítače z důvodu hardwarové závady nebo plánované údržby. Službu poskytuje jeden počítač, který je v případě výpadku automaticky zastoupen jiným počítačem bez přerušení poskytování služby nebo nutnosti zásahu administrátora.

Součástí vysoce dostupných klastrů je i vysoce dostupné síťové připojení a redundantní datové úložiště SAN[5]. Důležitou součástí vysoce dostupných klastrů je hearthbeat síť, která kontroluje dostupnost jednotlivých nodů klastru.

2.3.3. Klastr s rozložením zátěže

Klastr s rozložením zátěže (anglicky Load balancing cluster nebo Scalable cluster) snižuje možnou míru zátěže tím, že službu poskytuje několik počítačů, které mají stejný obsah (služba je poskytována paralelně). Stejný obsah je zajištěn replikací obsahu mezi všechny propojené počítače nebo existencí specializovaného centrálního úložiště.

Rozložení zátěže je realizováno pomocí specializovaného hardware nebo software, například Layer 4 Load Balancerů[6] nebo Round-robin DNS[7].

2.3.4. Úložný klastr

Úložný klastr (anglicky Storage cluster) zprostředkovává přístup k diskové kapacitě, která je rozložena mezi více počítačů z důvodu dosažení vyššího toku dat (paralelizace toku dat) nebo pro zajištění vyšší spolehlivosti pro tok dat. K zajištění služby jsou používány speciální souborové systémy, které jsou schopny zajistit rozložení zátěže, redundanci dat, pokrytí výpadků jednotlivých uzlů, distribuovaný mechanismus zamykání souborů a další doprovodné služby.

V distribuovaných systémech se používají block-level protokoly, například SCSI, iSCSI, Fibre Channel nebo InfiniBand.

Podle architektury systému mohou být metadata[8] distribuována mezi všechny nody klastru nebo metadata mohou být uložena na centralizovaných metadata serverech.

[5]Storage Area Network, dedikovaná síť poskytující konsolidované datové úložiště

[6]Router na transportní vrstvě OSI modelu posílající pakety na jedno nebo více zařízení schovaných za jednou IP adresou

[7]Jednomu doménovému jménu je přiřazeno více IP adres

[8]Strukturovaná data o datech

2.3.5. Gridový klastr

Klasický klastr je tvořen stejnými počítači, jejichž jediným úkolem je poskytovat svůj výpočetní výkon ve prospěch klastru. Gridový klastr (anglicky Grid cluster) je složen z nezávislých počítačů, které jsou typicky určeny pro jinou činnost (například desktop, server), a poskytování výpočetního výkonu pro využití v klastru je jejich doplňkový úkol. Jednotlivé počítače gridového klastru jsou proto z principu velmi různorodé (tzv. heterogenní klastr) a vzájemná součinnost je zajištěna na úrovni aplikačního software, což umožňuje využít i různých hardwarových i softwarových platforem. Jednotlivé počítače sdružené v gridovém klastru nemusí být umístěny pohromadě, ale mohou využívat pro vzájemnou komunikaci síť LAN, MAN i WAN a jsou plně pod kontrolou jejich vlastníků.

Klíčové vlastnosti gridových klastrů jsou koordinace zdrojů nepodléhajících centrální správě, použití standardních, otevřených a obecných protokolů a rozhraní a poskytování netriviální kvantity i kvality služeb.

Gridové klastry jsou často zaměňovány s distribuovanými systémy. Od distribuovaných systémů se ale gridové klastry liší především svým heterogenním prostředím a univerzálností celé platformy, zatímco distribuované systémy chápeme jako homogenní systémy s úzkým zaměřením.

Gridové klastry rozšiřují distribuované systémy o různé druhy zdrojů, možnost využití různorodého hardware, aplikací i dat, více druhů interakcí, uživatelské skupiny a aplikace interagující s gridovými klastry odlišným způsobem a dynamickou povahou celého klastru, kdy počet zdrojů a uživatelů se v čase mění.

Informace o gridových klastrech čerpají z [3].

2.4. Implementace klastrů

Klastry se od superpočítačů liší hlavně v přístupu k paměti. Zatímco superpočítače využívají sdílenou paměť, která je dostupná v rámci celého superpočítače pro všechny výpočetní jednotky (procesory), klastry mají paměť oddělenou na jednotlivé nody.

Počítačové klastry využívají klastrových souborových systémů, jako jsou například linuxový GFS, General Parallel File System od IBM, Cluster Shared Volumes od Microsoftu nebo Cluster File System od Oracle.

V počítačových klastrech se využívá specifikace Message Passing Interface (MPI) implementující stejnojmenný protokol pro podporu paralelního řešení výpočetních problémů v počítačových klastrech. Často používané označení jako knihovna (library) není v tomto případě zcela správné, jedná se totiž pouze o specifikaci, jak by taková knihovna mohla vypadat. Finální specifikace první verze MPI-1.0 byla vydána v květnu roku 1994, aktuální verze MPI-3.0 byla schvá-

lena v září roku 2012. MPI není ISO nebo IEEE standard, ale jedná se o jakýsi průmyslový standard pro HPC platformu.

MPI bylo původně navržené pro architektury s distribuovanou pamětí. Jak se ale později měnily trendy, systémy se sdílenou pamětí byly propojeny sítí a vytvářely tím hybridní distribuovanou paměť. MPI tedy podporuje všechny architektury – distribuovanou paměť, sdílenou paměť i hybridní scénáře. Z pohledu referenčního modelu ISO/OSI je protokol posazen do páté, tedy relační vrstvy, přičemž většina implementací používá jako transportní protokol TCP. Data jsou přesouvána z adresního prostoru jednoho procesu do adresního prostoru jiného procesu přes kooperativní operace obou procesů.

API Message Passing Interface je závislé na programovacím jazyce, jelikož se jedná především o síťový protokol. Nejčastěji se používají jazyky `C`, `C++`, `Java`, `Python` nebo `Fortran`, ale existují i podpory přímo na úrovni hardwaru.

2.4.1. MPI aplikace

V následující části textu zjednodušeně popíši základní vlastnosti MPI aplikací se zaměřením na jazyk `C`.

Struktura MPI aplikace Struktura MPI aplikace je vidět na obrázku č. 1..

Hlavičkové soubory Všechny MPI aplikace musí mít připojený hlavičkový soubor mpi.h.

```
#include "mpi.h"
```

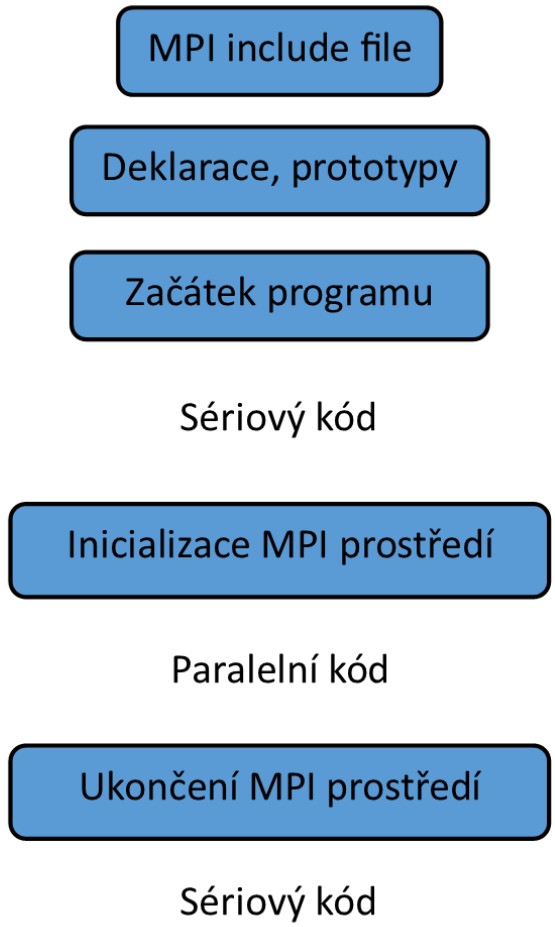
MPI volání MPI rutiny používají prefix MPI_.

```
rc = MPI_Bsend(&buf,count,type,dest,tag,comm)
```

Komunikátory a skupiny MPI používá objekty zvané komunikátory a skupiny pro určení, které kolekce procesů spolu mohou komunikovat. Většina MPI rutin požaduje specifikaci komunikátoru ve svém argumentu. Pokud není vyžadováno omezení, je možné použít předdefinovaný komunikátor MPI_COMM_WORLD, který umožní komunikaci mezi všemi MPI procesy.

Ranky Uvnitř komunikátoru má každý proces unikátní identifikátor typu integer, který je procesu přidělen systémem při jeho inicializaci. Čísla ranků začínají od nuly a jsou postupná. Ranky se používají pro určení zdroje a cíle zasílaných zpráv.

Zpracování chyb Většina MPI rutin má return/error code parametr. Podle MPI standardu je však výchozí chování volání prostřednictvím MPI zrušit, pokud dojde k chybě, což znamená, že pravděpodobně nebude možné odchytit jiné return/error code než MPI_SUCCESS (code zero).



Obrázek 1. Sktruktura MPI aplikace

Ukázka konkrétní MPI aplikace Součástí práce je i ukázková MPI aplikace na výpočet čísla π, která je na přiloženém DVD. Na této aplikaci jsem také demonstroval možnosti klastru v kapitole 7.. V následující části textu rozeberu základní části zdrojového kódu aplikace.

```
#include "stdafx.h"
#include "mpi.h"
#include <math.h>
#include <string>

int main(int argc, char* argv[])
{
        int n, rank, size, i;
        double PI25DT = 3.14159265358979323846264 3;
        double mypi, pi, h, sum, x;
```

Hlavičkové soubory, funkce main, deklarace a inicializace proměnných.

```
MPI_Init(&argc, &argv);
```

Inicializace MPI prostředí. Funkce musí být volána v každém MPI programu před jinými MPI funkcemi a může být v programu maximálně jednou.

```
MPI_Comm_size(MPI_COMM_WORLD,&size);
```

Vrací celkové množství MPI procesů ve specifikovaném komunikátoru MPI_COMM_WORLD.

```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
```

Vrací rank volaného MPI procesu ve specifikovaném komunikátoru.

```
if (rank == 0) {
                if (argc != 2)
                        return 0;
                n = atoi(argv[1]);
        }
```

Validace argumentů a převod druhého argumentu na integer.

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

Rozeslání dat (hodnoty n) všem procesům ve skupině MPI_COMM_WORLD. Zdrojovým procesem je proces s rankem 0. Další možností rozesílání dat je například MPI_Scatter, který více dat z jednoho procesu rozešle na ostatní procesy ve skupině, případně MPI_Gather, který naopak data z více procesů ve skupině pošle jednomu procesu.

```
h = 1.0 / (double) n;
        sum = 0.0;
        for (i = rank + 1; i <= n; i += size) {
                x = h * ((double)i - 0.5);
                sum += (4.0 / (1.0 + x*x));
        }
```

Výpočet velikosti intervalu.

```
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM,0,
   MPI_COMM_WORLD);
```

Sbírání částečných výsledků a jejich spojování. Posbírá částečné výsledky (hodnota mypi) a spojený výsledek (hodnota pi) uloží do jednoho nového procesu (rank 0). Alternativou k MPI_Reduce by mohlo být MPI_Allreduce, které uloží výsledek do všech procesů. Alternativou k MPI_SUM, které dělá součet, může být například MPI_MAX, který bere maximum hodnot nebo například MPI_LAND, který dělá logický AND.

```
MPI_Finalize();
```

Ukončení MPI prostředí. Tato funkce je poslední MPI rutinou zavolanou v každém MPI programu.

Celý kód aplikace pro výpočet π z přiloženého DVD vypadá následovně:

```
#include "stdafx.h"
#include "mpi.h"
#include <math.h>
#include <string>

int main(int argc, char* argv[])
{
        int n, rank, size, i;
        double PI25DT = 3.14159265358979323846264 3;
        double mypi, pi, h, sum, x;

        MPI_Init(&argc, &argv);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);

        if (rank == 0) {
                if (argc != 2)
                        return 0;
                n = atoi(argv[1]);
        }

        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

        if (n == 0)
                return 0;

        h = 1.0 / (double) n;
        sum = 0.0;
        for (i = rank + 1; i <= n; i += size) {
                x = h * ((double)i - 0.5);
                sum += (4.0 / (1.0 + x*x));
        }

        mypi = h * sum;

        MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM,0,
           MPI_COMM_WORLD);

        if (rank == 0)
                printf ("pi is approximately %1.30f, Error is
                   %1.30f\n",pi, fabs(pi-PI25DT));

        MPI_Finalize();
        return 0;
}
```

Vzhledem k tomu, že tvorba a hlubší rozbor vlastní MPI aplikace určené pro počítačové klastry je nad rámec zadání i řešení této práce, odkazuji zájemce o bližší informace a příklady na web Livermore Computing Center, případně přímo pro Microsoft HPC na web MSDN[9], konkrétně na Microsoft MPI, Microsoft Compute Cluster Pack nebo LINQ to HPC jako podpora pro vývoj aplikací pracujících s velkými datovými soubory.

Informace uvedené v této kapitole čerpají z [1] a [2].

3. Technologie klastrů a virtualizace

V této části mé práce se budu věnovat různým klastrovacím technologiím a virtualizací[10], která s klastry úzce souvisí.

Virtualizace je totiž základním nástrojem pro dosažení vysoké dostupnosti různých služeb a s tím také souvisejícím rozkladem zátěže mezi více hostitelských strojů. Díky virtualizaci je také možné výrazně šetřit náklady na provoz serverové infrastruktury i při zachování vysoké dostupnosti poskytovaných služeb.

Díky sdílenému vysoce dostupnému úložišti, na kterém jsou umístěny disky virtuálních operačních systémů, je možné v případě selhání některého hostitele přesunout běžící virtuální operační systémy na jiného hostitele bez přerušení poskytování služeb. Tím je zajištěna vysoká dostupnost poskytovaných služeb. Obdobně fungují i klastry pro rozložení zátěže, které požadavky klientů distribuují mezi více identických systémů.

3.1. Xen

Xen je hypervizor[11] s otevřeným zdrojovým kódem, který umožňuje spustit více instancí operačního systému nebo spustit více různých operačních systémů na jednom fyzickém počítači.

Xen je základem mnoha různých komerčních aplikací, jako jsou serverová a desktopová virtualizace, bezpečnostní aplikace a služby IaaS[12].

Xen používá mikrojádro, díky čemuž je zvýšena jeho robustnost a bezpečnost. Další výhodou oproti konkurenci je schopnost umožnit běh hlavního ovladače zařízení ve virtuálním stroji, což znamená, že pokud ovladač havaruje nebo je napadený, virtuální stroj obsahující ovladač může být restartovaný i s ovladačem,

[9] Microsoft Developer Network, web Microsoftu určený pro vývojáře

[10] Virtualizace operačních systémů umožňuje běh více operačních systémů na jednom fyzickém počítači

[11] Operační systém, který spravuje a přiděluje hardwarové prostředky hostovaným operačním systémům

[12] Infrastructure as a Service, poskytování infrastruktury jako služby

který neovlivní zbytek systému. Další výhodou je paravirtualizace, díky které mohou virtuální systémy běžet znatelně rychleji než s hardwarovými rozšířeními (HVM).

3.1.1. Typy hostů

Xen má dva základní typy hostů, kterými jsou paravirtualizace (PV) a plná neboli hardwarově asistovaná virtualizace (HVM). Oba typy virtualizace mohou být použity v jednu chvíli na jednom fyzickém stroji.

Paravirtualizace je efektivní a lehký typ virtualizace představené Xenem a dále adoptované dalšími platformami.

Paravirtualizace nevyžaduje virtualizační rozšíření hostitelského CPU, ale stačí Xen-PV-enabled jádro a PV ovladače. Xen-PV-enabled jádra existují pro Linux, NetBSD, FreeBSD a OpenSolaris.

Plná virtualizace využívá virtualizační rozšíření z hostitelského CPU, což vyžaduje Intel VT nebo AMD-V hardwarová rozšíření.

Xen používá Qemu pro emulaci hardware včetně BIOS, IDE diskového řadiče, VGA grafického adaptéru, USB řadiče, síťového adaptéru a dalších. Virtualizační rozšíření je použito pro zvýšení výkonu emulovaného hardware.

Plně virtualizované virtuální systémy nevyžadují žádnou podporu jádra, což znamená, že například Windows může být použit jako Xen HVM host.

Plně virtualizované systémy bývají pomalejší než hosté s paravirtualizací kvůli použité emulaci.

Paravirtualizace na plné virtualizaci umožňuje maximalizovat výkon plně virtualizovaných systémů využitím speciálních paravirtuálních ovladačů. Tyto ovladače jsou optimalizovanými PV ovladači pro HVM prostředí a obchází emulaci disku a síťových IO, což nám dává výkon na HVM jako u PV, případně i vyšší.

Informace o Xen hypervizoru čerpají z oficiálních stránek [4].

3.2. Citrix XenServer

Citrix XenServer je komerční implementací Xenu. Citrix rozšířil základní engine Xen o správcovské nástroje a zpřísnil různé komponenty spojené s implementací a správou virtuálních Windows a Linux systémů.

3.2.1. Vlastnosti Citrix XenServeru

Citrix XenServer podporuje všechny moderní techniky pro datacentra a cloudové služby, jako jsou site-to-site disaster recovery[13], dynamický rozklad zátěže, zajištění vysoké dostupnosti pomocí monitorování jednotlivých virtuálních strojů a automatických reakcí na chyby, snižování spotřeby elektrické energie pomocí konsolidace virtuálních strojů na menší počet fyzických strojů a následné vypínání nevyužívaných fyzických strojů a v případě potřeby jejich opětovné zapínání, tvorba snapshotů[14], živé migrace virtuálních strojů mezi fyzickými hostiteli za běhu systému a mnoho dalších.

Informace o Citrix XenServeru čerpají z oficiálních stránek [5].

3.3. Red Hat Enterprise Virtualization

Serverová virtualizace od Red Hat přichází v roce 2008 akvizicí firmy Qumranet, která vytvářela desktopovou virtualizaci založenou na KVM[15].

Na rozdíl od hypervizorů Microsoftu, VMware nebo Citrixu se řešení Red Hat nespoléhá na emulovaný hardware, ale používá paravirtualizaci, kdykoliv je možné připojit virtuální stroje přímo na hardware pomocí /dev/kvmkernel rozhraní.

Jelikož původní řešení Qumranetu bylo směrované na desktopovou virtualizaci, Red Hat musel přesunout koupené řešení do odvětví serverových virtualizací (Red Hat Enterprise Virtualization for Servers), ale nadále rozvíjel i desktopové virtualizace (Red Hat Enterprise Virtualization for Desktops).

Informace o Red Hat virtualizaci čerpají z oficiálních stránek [6].

3.4. VMware

Společnost VMware, Inc. je lídrem na trhu v oblasti virtualizace a cloudových technologií. Založena byla v roce 1998 v Californii a v roce 2004 se stala dceřinou společností firmy EMC Corporation.

Základní nabízenou technologií je VMware vSphere, což je virtualizační platforma pro budování cloudové infrastruktury. Spolu s Operations Managementem dávají možnost správy kompletní firemní infrastruktury pro virtualizaci.

Pod VMware vSphere spadají všechny produkty pro virtualizaci a správu, jako jsou ESXi, vCloud Suite, vCloud Director, vCloud Automation Center, vCloud Networking and Security a vSphere Storage Appliance.

[13] Proces obnovy nebo nepřerušené poskytování služeb v geograficky oddělených lokalitách
[14] Stav systému v konkrétním čase
[15] Kernel-based Virtual Machine, virtualizační infrastruktura pro linuxové jádro

VMware vSphere je postavený na hypervizoru ESXi, který je nainstalovaný přímo na fyzickém hardware a rozděluje hardwarové prostředky pro virtuální stroje.

Informace o VMware čerpají z oficiálních stránek [7].

3.5. Microsoft Hyper-V

Hyper-V od Microsoftu je z výše uvedených nejmladší. Finální verze Hyper-V byla vydána teprve 26. června 2008.

Hlavní výhodou pro mnoho zájemců může být cena, jelikož hypervizor Hyper-V je dostupný kompletně zdarma a spolu s mocným balíkem správcovských nástrojů z rodiny System Center se z této technologie stává silná konkurence pro nejrozšířenější VMware.

Microsoft Hyper-V oproti konkurenci působící na trhu delší dobu nijak nezaostává. Nabízí všechny základní vlastnosti, které jsou pro virtualizaci a obecně moderní softwarově definovaná datacentra nezbytná, jako jsou živé migrace virtuálních strojů, virtuální switche s podporou vlastních rozšíření, dynamické přidělování operační paměti virtuálním strojům, spojování síťových rozhraní do týmů pro zajištění vyšší datové propustnosti nebo vysoké dostupnosti síťového připojení a další.

Výhodou oproti konkurenci může být vyšší výkon diskového systému na neplnohodnotných diskových řadičích bez vlastní vyrovnávací paměti.

Informace o Hyper-V hypervizoru čerpají z oficiálních stránek [8].

Výše uvedené příklady klastrovacího software patří do klastrů pro zajištění vysoké dostupnosti, rozložení zátěže nebo úložných klastrů. Nepatří však do skupiny výpočetních klastrů, které jsou tématem mé bakalářské práce, a proto se jimi nebudu hlouběji zabývat. Další část textu již obsahuje popis řešení výpočetního klastru.

4. Dostupná řešení výpočetních klastrů

4.1. Red Hat Enterprise Linux for Scientific Computing

Red Hat nabízí balíčky specifické pro vědecké HPC uživatele pro jednoduché nasazení výpočetních klastrů.

Platforma Red Hat Enterprise Linux for Scientific Computing se skládá z Red Hat Enterprise Linux for HPC Head Node a z homogenního klastru Red Hat Enterprise Linux for HPC Compute Nodes.

4.1.1. Verze Red Hat Enterprise Linux for Scientific Computing

Red Hat Enterprise Linux for HPC Head Node je platforma se všemi vlastnostmi pro nasazení jako interaktivního systému v HPC klastru.

Red Hat Enterprise Linux for HPC Compute Node jsou výpočetní nody, které dědí úroveň podpory z řídícího Red Hat Enterprise Linux for HPC Head Node.

4.1.2. Základní vlastnosti

Standardizované API poskytuje funkcionalitu správy výpočetních úloh. Rozhraní příkazové řádky (CLI) poskytuje vysoce skriptovatelné rozhraní s veškerou funkcionalitou a konzistentním výstupem.

Desktop cycle harvesting umožňuje využití nevyužitého výkonu desktopů pro zvýšení výpočetní síly.

Workflow management slouží k definování závislostí mezi jednotlivými výpočetními úlohami.

Napojení na cloud díky MRG Gridu umožňuje využít výpočetní síly cloudových prostředí jako je Amazon EC2.

Centralizovaná konfigurace a administrace poskytuje flexibilní administrační politiky, úrovně práv podle jednotlivých úloh, správu licencí pro zajištění maximálního využití dostupných licencí, schopnost měnit konfiguraci bez nutnosti pozastavení výpočtů, možnost využití všech komponent systému v režimu vysoké dostupnosti a monitorování klastru v reálném čase s historií.

Informace o Red Hat Enterprise Linux for Scientific Computing čerpají z oficiálních stránek [9].

4.2. ParaTools HPC Linux

Společnost ParaTools, Inc. s pomocí partnerů vytvořila linuxovou distribuci zvanou HPC Linux, která obsahuje důležité nástroje pro vývoj paralelního software.

HPC Linux je dostupný zdarma ke stažení. Založený je na linuxové distribuci Fedora 16 a obsahuje všechny nástroje a vlastnosti z plné instalace Fedory.

Součástí HPC Linux je také PToolsWin toolkit, který umožňuje portování paralelních HPC aplikací s MPI nebo OpenMP z linuxu na Windows Azure.

Informace o Paratools HPC Linuxu čerpají z oficiálních stránek [10].

4.3. SUSE Linux Enterprise Server

SUSE Linux Enterprise Server je Linux běžící na 64bit a mainframe[16] systémech.

Mezi přednosti SUSE Linux Enterprise Server patří hlavně pokročilá správa paměti, podpora moderních procesorů, Native POSIX Thread Library (NPTL) a pokročilé MPIO[17] schopnosti.

Díky pokročilým možnostem a zapojení do programu Intel Cluster Ready je SUSE Linux Enterprise Server používaný na největších světových superpočítačích.

Pro potřeby systémů pracujících v reálném čase (real-time systems) SUSE vydalo doplněk pro SUSE Linux Enterprise, který umožňuje pro koncové uživatele odezvu v reálném čase.

SUSE a Microsoft umožňují používat mixované klastrové prostředí skrze Linux a Windows interoperabilitu, což opět umožňuje zjednodušit celé prostředí a snížit celkové náklady.

Informace o SUSE Linux Enterprise Serveru čerpají z oficiálních stránek [11].

4.4. PelicanHPC

PelicanHPC je hybridní obraz disku založený na Debian GNU/Linux Squeeze, který umožňuje zprovoznění HPC klastru během několika minut a umožňuje paralelní výpočty s použitím MPI.

PelicanHPC může být použitý na jednom stroji s více procesorovými jádry nebo může být použitý pro vytvoření výpočetního klastru z více počítačů s využitím počítačové sítě.

Hlavní server (fyzický nebo virtuální) bootuje z obrazu disku. Výpočetní nody bootují ze sítě pomocí PXE z hlavního serveru.

Instalační obrazy je možné upravovat díky Debian Live a přidávat jim balíčky pomocí apt-get, což umožňuje velice snadné upravení klastru pro vlastní potřeby.

Informace o PelicanHPC čerpají z oficiálních stránek [12].

4.5. Microsoft HPC

Řešení Microsoft HPC, které využívám ve své bakalářské práci pro realizaci výpočetního klastru, poskytuje obsáhlé a cenově dostupné řešení pro využití síly počítačů a serverů.

[16] Počítače primárně určené pro běh kritických aplikací ve velkých firmách

[17] Multipath I/O, vysoce dostupná a výkonná technika na systémech s více možnostmi propojení CPU s úložnými systémy

Microsoft HPC Pack umožňuje organizacím vytvářet řešení výpočetních klastrů s využitím stávající infrastruktury a dostupných zdrojů v rozmezí několika pracovních stanic s desktopovými Windows a nainstalovaným HPC Pack for Workstations až po tisíce výpočetních jader ve vysoce výkonných serverech. Uživatelé těchto výpočetních klastrů mohou snadno využít jejich Windows počítače pro vytváření a správu výpočetních úloh.

4.5.1. Edice Microsoft HPC

HPC Pack 2008 R2 Express je základní edice, která umožňuje nasazení a používání Microsoft HPC klastru. Obsahuje nástroje potřebné pro nasazení a správu klastru a plánovač úloh s podporou SOA a MPI.

HPC Pack 2008 R2 Enterprise and HPC Pack 2008 R2 for Workstation je nejvyšší edice obsahující všechny nástroje z edice Express, ke kterým přidává podporu pro běh Microsoft Excel úloh a možnost přidávat do klastru i pracovní stanice s desktopovými Windows.

Windows Server 2008 R2 HPC Edition je upravená verze Windows Server 2008 R2 pro použití v HPC klastru, která rozšiřuje jeho funkcionalitu o NetworkDirect RDMA, nástroje pro správu klastru, SOA správce úloh, MPI knihovnu a High Performance Computing Basic Profile.

Windows HPC Server 2008 R2 Suite je speciální balíček produktů Windows Server 2008 R2 HPC Edition a HPC Pack 2008 R2 Enterprise.

4.5.2. Systémové požadavky Microsoft HPC

Hlavní nod: Windows Server 2008 R2 Standard, Enterprise, Datacenter nebo HPC Edition

Výpočetní nod: 64bit varianty Windows Server 2008 nebo Windows Server 2008 R2 Standard, Enterprise, Datacenter nebo HPC Edition

Broker nod: Windows Server 2008 R2 Standard, Enterprise, Datacenter nebo HPC Edition

Workstation nebo Cycle Harvesting nod: Windows 7 Professional nebo Enterprise, 64bit verze Windows Server 2008 R2 Standard, Enterprise, Datacenter

Jako nody klastru jsou označovány výpočetní uzly klastru, tedy servery nebo koncové počítače zařazené do počítačového klastru.

4.5.3. Hardwarové požadavky Microsoft HPC

CPU: 64bit architektura kromě Workstation nodů, které mohou mít i 32bit

RAM: 512 MB

Pevný disk: Minimálně 50 GB diskového prostoru

Síť: Minimálně jedna síťová karta

Pro zapojení do klastru s vysokou dostupností je potřeba Enterprise edice hlavních a broker[18] nodů.

4.5.4. Vlastnosti Microsoft HPC

Integrace s Windows Azure Lokální řešení výpočetního klastru je možné propojit s cloudovým Windows Azure a pronajmout si vysoký výpočetní výkon.

Při využití Windows Azure je možné připojit VHD soubor přímo z lokálního úložiště do Windows Azure výpočetního nodu.

Plánování úloh Pro plánování úloh je možné využít zabezpečené webové rozhraní s pomocí klientských aplikací napsaných v .NET, které využívají API pro správu úloh, případně nástroje příkazové řádky určené pro správu úloh HPC klastru, HPC PowerShell cmdlety nebo přímo HPC Job Manager.

Webové rozhraní je rovněž možné s použitím API využít i pro sledování nodů v klastru.

Při konfiguraci plánovače úloh je možné nastavit úroveň preempce, aby se zamezilo nechtěnému ukončení některých typů úloh. Úroveň preempce je možné rozdělit na úroveň pro jednotlivé celé výpočetní úlohy (jobs) a na úroveň pro jednotlivé úkoly v rámci úloh (tasks), díky čemuž může být v rámci preempce ukočen pouze jeden úkol v celé úloze, ale nemusí se ukončovat celá úloha.

Uživatelským skupinám je možné definovat garantovanou dostupnost určitých výpočetních prostředků. Pokud uživatelská skupina nevyužívá všechny výpočetní prostředky, mohou tyto volné výpočetní prostředky využít jiné skupiny. Odebrání prostředků poté probíhá na základě politik preempcí.

[18] Přijímá požadavky od SOA klientů, distribuuje je na výpočetní nody a poté sbírá výsledky a předává je zpět klientům

Bezpečnost lze zvýšit zavedením požadavku na přihlášení uživatelů pomocí certifikátů.

S pomocí DLL knihoven je možné definovat aktivační filtry pro různé úlohy. Filtry mohou například kontrolovat dostupnost určitých licencí, které jsou potřebné pro běh spouštěné úlohy.

Úlohy v klastru mohou mít až 4000 úrovní priorit, což zajišťuje spravedlivé rozdělení výpočetních prostředků pro velké množství výpočetních úloh. Stále přitom existuje pět úrovní priorit pro základní rozložení – Lowest (0), BelowNormal (1000), Normal (2000), AboveNormal (3000), Highest (4000).

Při zadávání nové úlohy je možné zvolit výpočetní nody, na kterých úloha poběží exkluzivně. Žádná jiná úloha tedy nebude na vybraných nodech spuštěna, dokud nebude aktuální úloha dokončena. Stejně tak je možné vybrat výpočetní nody, na kterých zadaná úloha nemá běžet.

Úlohy je možné řádně ukončovat pomocí události CTRL_BREAK poslané aplikaci. Aplikace má poté správcem definovaný čas na řádné ukončení výpočtu, tedy například uložení aktuálního stavu výpočtu, zapsání zpráv do logu nebo vytvoření či smazání souborů. Po uběhnutí času na řádné ukončení (Grace Period) je výpočet ukončen násilně.

Plánování dostupnosti výpočetních nodů v klastru může být podle pevně daného rozvrhu dostupnosti, který může být rozšířený o detekci uživatelské aktivity na výpočetních nodech podle vstupů z klávesnice a myši nebo podle vytížení procesoru. Druhou možností plánování dostupnosti je ruční převod mezi online a offline stavem výpočetních nodů.

Jako výpočetní nody lze zvolit i servery, které jsou běžně využívané k jiným účelům, ale v určité době nejsou využity a jejich výpočetní výkon je tedy možné využít pro klastr. Na tyto servery se nainstaluje HPC Pack 2008 R2 for Cycle Harvesting a při splnění definovaných podmínek je server zpřístupněn výpočetním úlohám klastru.

Správa klastru probíhá přes HPC Cluster Manager. Výpočetní nody klastru nemusí být ve stejné Active Directory jako hlavní nod klastru. Pokud ale chceme přidat nody z jiné domény, musíme při přidávání nodu do klastru uvést FQDN[19] hlavního nodu.

Pro zálohu klastru je možné využít disaster recovery plánu, který umožňuje exportovat a poté opět importovat kompletní konfiguraci výpočetního klastru.

[19]Plně kvalifikované doménové jméno

Jako výpočetní nody lze zvolit i stanice bez pevných disků. Stanice bootují ze sítě s využitím iSCSI připojení na Windows HPC Server.

Nody v klastru je možné jednoduše řadit do lokací a podle těchto lokací je i spravovat.

Správci jednotlivých výpočetních úloh si mohou nastavit e-mailové notifikace při změně stavu úlohy. E-mailové adresy jednotlivých uživatelů jsou načítány z jejich profilů v Active Directory.[20]

Informace o Microsoft HPC čerpají z oficiálních stránek [13].

5. Implementace klastru s využitím Microsoft HPC

Implementaci počítačového klastru s využitím Microsoft HPC jsem začal řešit ihned po vybrání tématu bakalářské práce, tedy v září roku 2012. Instalaci a základní nastavení klastru jsem dokončil podle plánu ještě v zimním semestru.

Výpočetní nody klastru jsou tvořeny pracovními stanicemi HP Z210 (Intel Xeon E3-1240, 8 GB DDR3 ECC RAM, 1000 GB SATA III HDD, NVIDIA Quadro 2000, Windows 7 Enterprise 64bit) na počítačové učebně 5.003. Všechny počítače na učebně i hlavní nod jsou připojeny k lokální 1 Gbit/s síti, která je součástí infrastruktury fakulty.

Jako hlavní nod, který se stará o správu celého klastru, jsem použil Windows Server 2008 R2 HPC Edition, který je licenčně dostupný v rámci DreamSpark Premium předplatného katedry. Server je nainstalovaný jako virtuální.

V dalším kroku následovalo zařazení serveru do fakultní Active Directory, což je nutný krok pro vytvoření klastru, a instalace Microsoft HPC Pack 2008 R2 Enterprise[14] na hlavní nod.

V prvním kroku instalace je možné vybrat, jestli chceme instalovat HPC Pack 2008 R2 Express, který obsahuje nástroje pro nasazení a správu výpočetního klastru, plánovač událostí s podporou SOA a MPI úloh. Verze Express nevyžaduje žádné další licence. Druhou volbou je HPC Pack 2008 R2 Enterprise and HPC Pack 2008 R2 for Workstation. Enterprise verze umožňuje navíc oproti Express verzi spouštět úlohy Microsoft Excel workbooks. Volil jsem volbu Enterprise, protože licence na Enterprise verzi je dostupná v předplatném DreamSpark Premium a pokročilejší funkčnost může být využita v budoucnu.

[20] E-mailové notifikace nefungují, pokud cílový účet není v Microsoft Exchange. Všechny nové zaměstnanecké účty od roku 2013 mají již e-mailové schránky v Microsoft Exchange. Starší účty, které ještě nebyly do Microsoft Exchange převedeny, je možné nechat převést.

V dalším kroku je na výběr typ instalace pro tento počítač. První volbou, kterou jsem volil i já, je Create a new HPC cluster by creating a head node, tedy vytvoření nového HPC klastru vytvořením hlavního nodu. Dalšími možnostmi jsou připojení k existujícímu klastru vytvořením nového výpočetního nodu, připojení k existujícímu klastru vytvořením WCF[21] broker nodu, připojení k existujícímu klastru vytvořením workstation nodu a instalace pouze nástrojů pro správu klastru (HPC Cluster Manager, HPC Job Manager a HPC PowerShell).

Dalším krokem instalace je výběr umístění databáze s informacemi o klastru. Doporučení Microsoftu je použití lokální databáze na hlavním nodu, pokud se klastr skládá z méně než 256 nodů. Tohoto doporučení jsem se držel a databáze pro Cluster management, Job scheduling, Reporting i Diagnostics jsem umístil na hlavní nod. Součástí instalace byla tedy i instalace Microsoft SQL Server 2008 SP1 databáze s instancí COMPUTECLUSTER.

Po dokončení instalace je klastr aktivní a zbývá již jen dokončit další nastavení. Jméno klastru je stejné jako jméno hlavního nodu, v mém případě se tedy klastr jmenuje HPC-INF.

V posledních krocích instalačního průvodce jsem už jen potvrdil umístění databáze i všech souborů HPC Packu do výchozích umístění a spustil samotnou instalaci.

Posledním krokem je aktualizace Microsoft HPC Pack 2008 R2 postupně na SP1, SP2, SP3 a SP4.

Následně bylo nutné konfigurovat nový klastr na hlavním nodu pro potřeby katedry. Konfigurace probíhá přes HPC Manager v několika krocích.

5.1. Konfigurace sítě

Při konfiguraci sítě je možné zvolit z pěti různých topologií:

1. Compute node isolated on a private network, která se používá pro oddělenou privátní síť vyhrazenou pouze pro výpočetní nody. Výpočetní nody nejsou součástí žádné jiné sítě a jsou připojeny vyhrazeným síťovým rozhraním k serveru. Pro tuto konfiguraci je tedy nutné mít minimálně dvě síťová rozhraní na hlavním nodu.

2. All nodes on enterprise and private networks, která se používá v případě, že všechny nody jsou připojeny k privátní i neprivátní síti, přičemž privátní síť se využívá pouze pro potřeby klastru. Tato konfigurace vyžaduje minimálně dvě síťová rozhraní na hlavním nodu i na výpočetních nodech.

[21] Windows Communication Foundation, běhové prostředí a balík rozhraní (API) v .NET Frameworku pro propojené aplikace

3. Compute nodes isolated on private and application networks využívá dvě sítě pro klastr, kdy jedna síť je vyhrazena pro komunikaci aplikací (výpočtů) a druhá pro správu klastru. Konfigurace vyžaduje minimálně dvě síťová rozhraní na výpočetních nodech a minimálně tři síťová rozhraní na hlavním nodu.

4. All nodes on enterprise, private, and application networks je založená na třech sítích. Jedna síť slouží pro komunikaci aplikací (výpočtů), druhá pro správu klastru a třetí je běžná firemní síť. V této konfiguraci jsou vyžadována minimálně tři síťová rozhraní na výpočetních nodech i hlavním nodu.

5. All nodes only on an enterprise network je konfigurace, kterou jsem použil pro výpočetní klastr, jelikož počítače na učebnách mají pouze jedno síťové rozhraní, pomocí kterého jsou připojeny do fakultní infrastruktury. Všechny výpočetní nody i hlavní nod jsou tedy připojené pouze k jedné firemní (univerzitní) síti. Z tohoto důvodu nebylo možné využít žádnou jinou možnost. Na hlavním nodu jsem nastavil stejnou síť, jaká je na učebnách, tedy 158.194.92.0/24, přiřadil statickou IP adresu a nastavil doménové jméno.

Dalším krokem konfigurace sítě je zvolení síťového rozhraní serveru, které bude použité pro výpočetní klastr. Při zvolené topologii s jednou sítí stačí vybrat jedno síťové rozhraní.

Posledním krokem konfigurace sítě je nastavení firewallu. Na výběr je ze dvou možností. První doporučenou možností je automatická konfigurace firewallu na výpočetních nodech i hlavním nodu pro potřeby klastru, druhou možností je ponechání nastavení firewallu v aktuálním stavu. Zvolil jsem tedy automatickou úpravu pravidel na firewallu na stanicích i na hlavním nodu.

Následuje již jen přehled provedených nastavení a potvrzení.

5.2. Poskytnutí přihlašovacích údajů správce

V další části prvotní přípravy klastru je potřeba zadat údaje doménového účtu, který má administrátorská oprávnění pro instalaci software a správu všech stanic. Zvolil jsem tedy svůj zaměstnanecký účet.

5.3. Konfigurace jmen výpočetních nodů

Pro snadnější správu se pro všechny výpočetní nody nastavují jména, která se automaticky generují všem nově přidaným výpočetním nodům.

Vzhledem k tomu, že v současné době je využita jedna počítačová učebna s 18 počítači pro výpočetní klastr, jsem zvolil jako číslovací šablonu pro výpočetní nody %10%, která umožňuje číslování od 10 do 99. Tím je zaručený dostatečný

prostor pro nové výpočetní nody i v budoucnu, pokud by se výpočetní klastr rozšiřoval na další výpočetní stroje. Celá jmenná šablona má tvar HPC-INF%10%.

5.4. Vytvoření šablony pro nody

Šablony slouží pro zjednodušení přidávání, konfigurace a správy nodů v klastru. Pokud je nějaká šablona nodům přiřazena, jsou na tyto nody automaticky aplikována pravidla definovaná v této šabloně.

Pomocí šablon je možné například definovat dostupnost výpočetních nodů, spravovat aktualizace pomocí Windows Update nebo WSUS[22], přidávat nody do domény Active Directory, spravovat sdílené jednotky připojené na výpočetních nodech a další. Součástí šablon mohou být i předpřipravené obrazy systémů, které je díky tomu možné snadno nasadit včetně přidání aktualizací, aktivačních klíčů apod.

Existují čtyři základní typy šablon:

1. Compute node template se používá pro výpočetní nody, které jsou trvale dostupné. Tento typ nodů umí přijímat a spouštět výpočetní úlohy a akceptuje SOA požadavky od broker nodů.

2. Broker node template přijímá požadavky od Service-Oriented Architecture (SOA) klientů, distribuuje tyto požadavky na výpočetní nody klastru, sbírá odpovědi a posílá je zpět klientům. Broker nod musí být připojený do firemní sítě.

3. Workstation node template slouží pro pracovní stanice, které budou využívány pro výpočty. Tato šablona jako jediná umožňuje definovat dostupnost výpočetních nodů. Šablonu pro pracovní stanice jsem použil i pro výpočetní nody na učebně 5.003.

4. Windows Azure node template umožňuje napojení na Windows Azure a správu Windows Azure výpočetních nodů.

U zvolené Workstation node template jsem nastavil jméno podle učebny, tedy 5003, a nastavil jsem dostupnost nodů podle rozvrhu hodin na učebně tak, aby se maximalizovala dostupnost, ale přitom to nijak nezasahovalo do výuky. Před koncem dostupnosti výpočetních nodů jsem nastavil 15 minutový interval, po který již nelze spouštět žádné nové úlohy. Dostupnost výpočetních nodů je nastavována automaticky pomocí vlastního skriptu, který si stahuje údaje ze STAGu.

Volitelně je možné nastavit i detekci uživatelské aktivity na počítačích a tím ještě více maximalizovat dostupnost výpočetních nodů. Detekce aktivity funguje

[22] Windows Server Update Services, lokální server pro správu aktualizací

v online stavu výpočetních nodů, kdy umožňuje detekovat aktivitu na počítači podle vstupu z klávesnice nebo myši, případně podle využití CPU. Pokud je detekce aktivity aktivována, všechny výpočetní úlohy běží na dané stanici s prioritou BelowNormal (nebo nižší, pokud uživatel klastru zadá výpočetní úlohu s nižší prioritou), což zajišťuje, že výpočetní procesy nebudou mít vyšší prioritu než běžné procesy uživatelů. Pokud je aktivována detekce aktivity v závilosti na využití CPU, jsou výpočetní úlohy spuštěny jen tehdy, pokud vytížení CPU dané stanice klesne pod definovanou hodnotu po určitý definovaný čas. Do využití CPU se započítávají pouze procesy s prioritou Normal nebo vyšší, které nejsou procesy výpočetního klastru. Detekce uživatelské aktivity v tuto chvíli není na klastru aktivní, jelikož se využívá plánování dostupnosti podle rozvrhu na učebně, z tohoto důvodu tedy výpočty na klastru nemohou ovlivnit výuku.

Pokud výpočetní nod přechází z online stavu do offline stavu, ale na nodu běží výpočetní úlohy, je úlohám umožněno doběhnout, případně řádně uložit svůj stav a ukončit výpočet. Doba, po kterou mají úlohy čas na řádné ukončení, je definovaná pomocí Task Cancel Grace Period v nastavení klastru. Výchozí hodnota je nastavená na 15 sekund, tuto hodnotu jsem ponechal. Aby bylo možné zajistit řádné uložení a ukončení výpočtu, musí výpočetní úloha umět zpracovat událost CTRL_BREAK. Pokud úloha tuto událost zpracovat neumí, je ukončena násilně.

Pokud je na výpočetních nodech nastavený automatický přechod mezi online a offline stavem podle definovaného rozvrhu, není možné ručně změnit stav výpočetních nodů. Pro ruční změnu je nutné přiřadit nodům šablonu, která je nastavena pro manuální změny stavu výpočetních nodů.

5.5. Instalace Microsoft HPC Pack na stanice v učebně

Na pracovní stanice v učebně 5.003 bylo nutné nainstalovat HPC Pack, abych je mohl přidat do výpočetního klastru jako výpočetní nody. Stáhl jsem si tedy Microsoft HPC Pack 2008 R2 Enterprise, který jsem nainstaloval a aktivoval.

Instalace probíhá stejně jako na hlavním nodu, jen v druhém kroku typu instalace jsem vybral Join an existing HPC cluster by creating a new workstation node a v dalším kroku jsem vybral název již existujícího klastru HPC-INF. Následovalo opět jen potvrzení umístění a spuštění instalace.

Posledním krokem je opět aktualizace Microsoft HPC Pack 2008 R2 postupně na SP1, SP2, SP3 a SP4.

5.6. Přidání výpočetních nodů do klastru

Před prvním použitím klastru pro výpočty je nutné všechny nově nainstalované nody přidat do HPC Cluster Manageru a publikovat tyto nody pro výpočty.

Nově nainstalované nody jsou v HPC Cluster Manageru zařazeny do kategorie Unapproved a je potřeba jim přidělit šablonu. Šablonu jsem tedy přiřadil označením všech nodů v kategorii Unapproved a zvolením Assign Node Template, kde jsem vybral dříve vytvořenou šablonu 5003. Aplikováním šablony dojde k přesunutí všech nodů do kategorie OK. Pokud by se aplikace šablony nepodařila například z důvodu nedostupnosti daného nodu, bude nod přesunutý do kategorie Error a rozkliknutím daného nodu je možné na záložce Node Health vidět důvod chyby.

5.7. Instalace a konfigurace webového rozhraní

Microsoft HPC Pack 2008 R2 obsahuje webové rozhraní Windows HPC Server Web Portal, kde je možné sledovat právě běžící úlohy na klastru, přidávat nové výpočetní úlohy, spravovat šablony výpočetních úloh a sledovat vytížení jednotlivých výpočetních nodů. Ke správě úloh se využívá HTTP rozhraní HPC Job Scheduler Service založené na REST[23] modelu.

Před samotnou instalací webového rozhraní jsem přidal na server roli Internetové informační služby (IIS) a nainstaloval důvěryhodný certifikát podepsaný certifikační autoritou Terena SSL CA, který je dostupný v rámci služeb CESNET, z.s.p.o. pro všechny instituce připojené do národní akademické sítě CESNET2.

Instalace webového rozhraní probíhá v několika jednoduchých krocích instalačního průvodce. Poté je v Internetové informační službě potřeba jen nadefinovat cestu k domovskému adresáři webového portálu a přiřadit certifikát. Posledním krokem je spuštění portálu pomocí PowerShell skriptu Set-HPCWebComponents.ps1 s parametry –Service Portal -enable, který je součástí složky HPC Pack.

5.8. Přidání oprávněných uživatelů a skupin

Aby uživatelé mohli zadávat úlohy, musí být ve skupině Users v HPC Cluster Manageru. Do této skupiny jsem tedy přidal celou skupinu zaměstnanců katedry informatiky z fakultní Active Directory. Na vyžádání je možné přidávat i jednotlivé uživatele, kteří nejsou vedeni jako zaměstnanci, například studenti, kteří v rámci svých prací využijí možnosti výpočetního klastru. Celou skupinu zaměstnanců katedry informatiky jsem přidal i do uživatelů vzdálené plochy na serveru hpc-inf.

Všechny výpočetní úlohy musí běžet ze síťového úložiště, které je dostupné všem výpočetním nodům, případně musí být rozkopírované do složky C:\HPC na všechny výpočetní nody, na kterých úloha bude běžet. Na serveru storage-inf jsem tedy vytvořil novou sdílenou jednotku \\storage-inf\hpc$, do které mají

[23] Representational State Transfer, architektura navržená pro distribuované prostředí

přístup všichni zaměstnanci. V této složce si každý zaměstnanec vytvoří vlastní složku, ve které bude mít všechny své výpočetní úlohy. Rovněž jsem na počítačích v klastru vytvořil složku C:\HPC, kde mají zaměstnanci plná práva, ale studenti nemají žádná práva.

5.9. Dostupnost výpočetních nodů

Všechny stanice na učebnách mají dostupnost definovanou podle rozvrhu na učebně a mimo výuku je možné stanice využívat pro výpočty. Z tohoto důvodu je v doménových politikách nastaveno, aby studenti nemohli vypínat počítače. Počítače jsou nastavené tak, aby se po 30 minutách nečinnosti uspaly (režim S3, Suspend to RAM) a bylo umožněno je v případě potřeby i vzdáleně probudit. K probuzení počítačů slouží aplikace NetScan uložená ve složce WakeOnLan na disku C hlavního nodu. V aplikaci NetScan je v menu Options položka Wake-On-Lan Manager, kde jsou definované MAC adresy všech počítačů v učebně 5.003 a pomocí této aplikace je možné budit jednotlivé počítače nebo všechny počítače zároveň.

Pro účely automatického plánování dostupnosti výpočetních nodů jsem si napsal vlastní skript v PowerShellu, který stahuje informace o výuce na učebně 5.003 ze STAGu v XML souboru, který dále parsuje a převádí data do formátu šablony výpočetních nodů klastru, kde jsou data o dostupnosti uložena v blocích skládajících se ze čtyř třicetiminutových úseků. Tyto bloky jsou v XML šabloně výpočetních nodů uloženy v řadě znaků 0 až 9 a A až F, kde každý znak symbolizuje jeden dvouhodinový blok (čtyři třicetiminutové úseky) podle toho, které třicetiminutové úseky jsou v tom bloku využity, z čehož vyplývá značení pomocí 0 až F, jelikož se jedná o 2^4 možností. Zápis začíná nedělí 00:00 a končí sobotou 24:00.

```
<Parameter Name="SerializedAvailabilitySchedule"
           Value="FFFFFFFFFFFFFFF001C3 ..." />
```

Výše uvedený příklad ukazuje zadání hodnoty dostupnosti do šablony výpočetních nodů klastru. Písmenem F je označený blok čtyř třicetiminutých úseků, kdy je klastr dostupný pro výpočty. Při zadávání dostupnosti od neděle 00:00 je z příkladu patrné, že klastr je dostupný pro výpočty celou neděli (dvanáct znaků F, každý dvě hodiny) a dále v pondělí od půlnoci do 8:00 (další čtyři znaky F), kdy začíná výuka. Celkově tedy 16 znaků F na začátku dostupnosti. Znak 0 značí blok čtyř třicetiminutových úseků nedostupnosti klastru pro výpočty. Dva znaky 0 tedy značí čtyři hodiny nedostupnosti v kuse, v tomto příkladě tedy nedostupnost od 8:00 do 12:00. Další znak 1 značí dostupnost klastru pouze ve čtvrtém třicetiminutovém úseku dvouhodinového bloku, tedy prodloužení nedostupnosti o další tři třicetiminutové úseky až do 13:30 a od 13:30 do 14:00 dostupnost klastru. Znak C značí dostupnost v prvním a druhém třicetiminutovém úseku

v dvouhodinovém bloku, tedy dostupnost od 14:00 do 15:00 a od 15:00 do 16:00 nedostupnost. Následující znak 3 značí dostupnost v třetím a čtvrtém třicetiminutovém úseku dvouhodinového bloku, což nám tedy udává nedostupnost klastru od 16:00 do 17:00 a od 17:00 do 18:00 dostupnost.

Tento vlastní skript zaokrouhluje začátky nedostupnosti na nejbližší předchozí třicetiminutový úsek a konce nedostupnosti na nejbližší následující třicetiminutový úsek. Skript záměrně nebere v úvahu jednorázové rozvrhové akce na učebně ani víkendové akce, jelikož se tyto typy akcí mohou často měnit a jedná se o potenciální zdroj problémů, což by mohlo mít za následek negativní vliv na dostupnost výpočetních nodů a tedy i vědeckých výpočtů.

Skript je spouštěn v plánovači každé pondělí v 9:00, jelikož v tuto dobu by již měla probíhat výuka, výpočetní klastr by tedy měl být nedostupný pro výpočty a nemělo by dojít k nečekané změně stavu výpočetních nodů z Online do Offline. Dalším důvodem volby tohoto času byly různé údržby a plánované odstávky univerzitních systémů v nočních hodinách, které by mohly v případě spouštění skriptu v noci negativně ovlivnit i jeho výsledky a tím pádem i celý výpočetní klastr.

5.10. Instalace SSH serveru

SSH[24] server slouží pro multiplatformní textové vzdálené ovládání. Po přihlášení se automaticky spustí Windows PowerShell, ze kterého je možné přímo spouštět skripty. Aby bylo možné klastr ovládat pohodlně a přitom plnohodnotně i z jiných operačních systémů, nainstaloval jsem na hlavní nod PowerShell Server. Tento software je v základní verzi zdarma pouze s omezením na 1 současné spojení. Při testování jiných SSH serverů pro Windows jsem však došel k závěru, že i přes omezení na jedno současné spojení se jedná o nejlepší možnost, jelikož je to aktuální a stále vyvíjený a podporovaný software s plnou kompatibilitou s operačním systémem Windows Server 2008 R2 a podporou PowerShell 3.0.

U alternativního software jsem se setkal s ukončeným vývojem a tedy nekompatibilitou, zcela nebo částečně nefunkčním Windows PowerShell, nepodporou pro doménové účty nebo pouze komerční verzí software. Z testovaného software mohu zmínit například OpenSSH for Windows, Copssh, freeSSHd, SilverSHielD, MobaSSH nebo Bitvise SSH Server.

5.11. Nastavení systémové proměnné PATH

Aby bylo možné vlastní PowerShell skripty spouštět odkudkoliv, nastavil jsem cestu C:\PowerShell scripts do systémové proměnné PATH v Control Panel\System and Security\System\Advanced system settings\Environment Variables.

[24] Secure Shell, zabezpečený komunikační protokol

5.12. Řešení hibernace úloh na klastru

Výpočetní klastr měl podle zadání práce umožňovat také hibernaci úloh. Hibernace úloh slouží k tomu, aby při ukončení dostupnosti výpočetního nodu mohla být celá úloha uložena a poté spuštěna na jiném výpočetním nodu ze stavu, ve kterém skončila, případně na stejném výpočetním nodu po obnovení jeho stavu. Stejně tak by hibernace úloh klastru umožňovala přesouvat výpočetní úlohy mezi výpočetními nody.

Hibernaci úloh klastrovací software Microsoft HPC nepodporuje. U tohoto typu software se předpokládá, že klastr poběží trvale dostupný a hibernace úloh tedy není potřebná funkcionalita. Vytvoření klastru pouze z běžných počítačů není standardní scénář využití, předpokládá se využití primárně z trvale dostupných výpočetních serverů, které jsou případně o běžné počítače pouze rozšířeny. Přidání běžných počítačů do klastru je také novinkou v aktuálně nainstalované verzi Microsoft HPC Pack 2008 R2 a v dřívějších verzích tato možnost nebyla vůbec dostupná.

Aby bylo možné zavést hibernaci úloh, je potřeba, aby podpora byla přímo v operačním systému, na kterém klastr běží. Na učebně, kde je klastr nainstalovaný, je použitý běžný desktopový operační systém Microsoft Windows 7, který tuto podporu nemá. Vzhledem k tomu, že počítače na učebně primárně slouží k výuce, není možné, aby byl nainstalovaný jiný operační systém, který by podobnou podporu poskytoval.

Důvod, proč není možné v běžném systému bez této podpory proces hibernovat a obnovit v systému na jiném nodu, je například ten, že v systému je stav, který není součástí procesu. Těmito stavy jsou například otevřené soubory (a pozice v těchto souborech), otevřená síťová připojení (TCP sockety), nastavené CPU afinity, používaná synchronizační primitiva jako semafory, mutexy apod. Konkrétní příklad s mutexem – pokud program uzamkl mutex a proces by v této chvíli měl být hibernovaný, došlo by k uvolnění mutexu (kvůli chybějící podpoře uložení a obnovení stavu mutexu, případně i v systému na jiném nodu, při hibernaci). Pokud by mutex chránil nějaký stav procesu, po obnovení procesu z hibernace by se domníval, že je mutex stále uzamčený a stav by tedy měl být chráněný před neoprávněnou manipulací. Jenže mutex byl hibernací uvolněn a stav je nechráněný a může být poškozený. Mutex tedy neplní svoji funkci. Podobným příkladem by mohla být práce se soubory s výhradním přístupem. Pokud by si proces otevřel soubor s výhradním přístupem a proces byl poté hibernován, vyhradní přístup by byl ukončen (ze stejného důvodu jako u mutexu) a po obnovení procesu by mohl přístup získat jiný proces. Původní proces by se tak dostal do nekonzistentního a neočekávaného stavu a mohl by havarovat.

Řešením by teoreticky mohlo být ponechat otevřené všechny objekty patřící k hibernovanému procesu. Zde ale nastávají další problémy s tím, že tyto objekty by byly zamčené pro jakoukoliv další manipulaci. A vzhledem k tomu, že

proces by mohl být hibernovaný i po dlouhou (teoreticky nekonečně dlouhou) dobu, způsobovalo by to dalekosáhlé problémy. S těmito problémy souvisí i další problémy s přesunem procesu na jiný výpočetní nod, protože tam nemusí být dostupné všechny soubory a další prostředky, poběží tam jiné procesy, které budou jinak využívat systémové prostředky apod., proces by tedy po obnovení stavu opět končil pravděpodobně v nekonzistentním stavu. Podpora hibernace úloh je tedy nutná přímo v operačním systému.

Pro operační systém Linux jsou projekty, které se tímto zabývají. Většina projektů je založena na principu checkpoint & restart. Aktuálně použitelný je například CryoPID. Pro Windows však takové projekty neexistují vzhledem k uzavřenosti systému.

Pro uložení stavu výpočtu by teoreticky bylo také možné použít například výrazně ořezaný virtuální systém, na kterém by běžel pouze výpočetní proces a nezbytně nutné systémové služby, a který by poté byl hibernovaný a případně přesunutý jako celek na jiný nod. Hibernaci virtuálního operačního systému a spuštění na jiném fyzickém počítači umožňují všechny rozšířené hypervizory. Hlavní nod by tedy kromě obsluhy výpočetních procesů musel obsluhovat i virtualizaci a jako příkaz pro hibernaci úlohy by byl příkaz pro hibernování celého virtuálního operačního systému i s úlohou. Popisované řešení je ale již poměrně náročné a přesahuje rámec zadání i řešení práce.

Z výše uvedených důvodů není hibernace úloh ze zadání práce implementována.

6. Uživatelská dokumentace

Součástí bakalářské práce je i uživatelská dokumentace k výpočetnímu klastru.

Ovládání klastru je v základu možné třemi způsoby:

1. HPC Job Manager
2. PowerShell
3. Windows HPC Server Web Portal

Všechny výpočetní úlohy musí mít formu spustitelného souboru (tj. nejen EXE, ale i BAT, CMD, COM, MSI) z příkazové řádky operačního systému Microsoft Windows s volitelně zadanými parametry. Každá výpočetní úloha se spouští z příkazové řádky výpočetního klastru a všechny výpočetní úlohy včetně dalších nutných dat musí být uloženy na síťovém úložišti, které je dostupné pro všechny výpočetní nody, případně rozkopírované do adresáře C:\HPC na všechny výpočetní nody, které se budou na výpočtu podílet. K uložení dat výpočetních úloh

i samotných výpočetních aplikací slouží síťové úložiště \\storage-inf\hpc$ nebo lokální adresář HPC na všech výpočetních nodech. Zde si každý zaměstnanec vytvoří vlastní adresář podle svého přihlašovacího jména do počítačové sítě a volitelně si může změnit oprávnění ke své složce pro ostatní uživatele.

Výpočetní úlohy musí být samostatně fungující celky, které nevyžadují další aplikace nebo nástroje. Není tedy možné například spouštět Matlab nebo vývojová prostředí.

V následujících částech textu přidávám uživatelskou příručku, která má formu přímočarého návodu.

6.1. Probuzení výpočetních nodů

Před samotným zahájením výpočtů je nutné probudit výpočetní nody, které po 30 minutách nečinnosti automaticky usínají. Výpočetní nody je možné probudit pomocí předpřipravené aplikace NetScan, která je uložená ve složce WakeOnLan na disku C hlavního nodu. V aplikaci NetScan je v menu Options položka Wake-On-Lan Manager, kde jsou definované MAC adresy všech počítačů v učebně 5.003 a pomocí této aplikace je možné budit jednotlivé počítače nebo všechny počítače zároveň.

6.2. Ovládání klastru pomocí HPC Job Manager

HPC Job Manager je základním grafickým nástrojem spustitelným ze vzdálené plochy hlavního nodu výpočetního klastru a slouží pro správu výpočetních úloh.

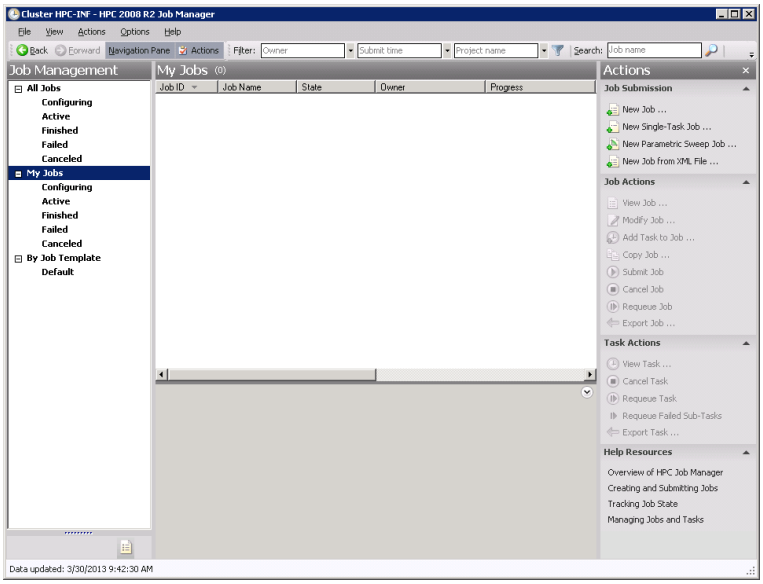
HPC Job Manager je možné spustit ze vzdálené plochy serveru hpc.inf.upol.cz. Po připojení na vzdálenou plochu serveru pomocí uživatelského účtu pro přístup do počítačové sítě spusťte aplikaci HPC Job Manager, která se nachází ve složce Microsoft HPC Pack 2008 R2 v programech nabídky Start.

V levé části aplikace se nachází nabídka úloh, kde je možné vidět úlohy podle jejich stavu. Ukončené úlohy jsou uchovávány v historii po dobu 7 dnů. Pravá část aplikace obsahuje nabídku akcí, které je možné s úlohami provádět, viz obrázek 2.

6.2.1. Vytvoření nové výpočetní úlohy

V pravém menu akcí vyberte New Job. V nově vytvořeném okně zadejte do políčka Job name jméno výpočetní úlohy pro pozdější snadnější orientaci. Job template ponechejte Default, Project nechte prázdný a Priority zůstane Normal.

Můžete si nastavit odeslání notifikací při spuštění úlohy a/nebo při ukončení úlohy pomocí zatržítek u položky Send a notification. Při zatržení alespoň jedné



Obrázek 2. HPC Job Manager

volby se zobrazí políčko s výchozí e-mailovou adresou načtenou z přihlášeného účtu. Tuto adresu je možné libovolně změnit, ale berte v úvahu problém s doručováním notifikací na e-maily, které nejsou na Microsoft Exchange.

Ve spodní části okna ponechejte typ Job resources jako Node kvůli paralelizaci pomocí MPI a volitelně můžete změnit minimální a maximální počet nodů přidělených výpočetní úloze. Pokud nastavíte Auto, výpočetní klastr sám určí počet přiřazených nodů pro úlohu vzhledem k celkovému vytížení klastru.

V levé části okna se přepněte na záložku Edit Tasks a poté v pravé části okna zvolte Add. Do políčka Task name můžete zadat jméno přidávaného výpočetního procesu. Do políčka Command line zadejte název spustitelného souboru výpočetního procesu včetně případných argumentů. Pokud spouštíte MPI úlohu, zadejte jako první mpiexec, což je aplikace, která umožní spouštět proces na více výpočetních nodech, a až za mpiexec název spustitelného souboru výpočtu s případnými argumenty. Pokud se spustitelný soubor výpočetního procesu nenachází přímo v pracovním adresáři nastaveném níže v políčku Working directory, zadejte i relativní cestu vzhledem k pracovnímu adresáři nebo absolutní cestu. Cesta ke spustitelnému souboru musí být dostupná pro všechny výpočetní nody.

Do Working directory zadejte cestu k pracovnímu adresáři výpočetního procesu. Tato cesta musí být dostupná pro všechny výpočetní nody. Do pracovního adresáře se poté relativně zadávají cesty spustitelných souborů výpočtu v políčku Command line výše a standardní vstupy, výstupy i chybové výstupy výpočetního procesu v políčkách níže.

Jako poslední můžete specifikovat minimální a maximální počet výpočetních zdrojů přiřazených procesu. Typ zdroje je určen typem vybraným pro celou úlohu

v předchozí části na záložce Job Details.

Na záložce Resource Selection můžete ručně vybrat výpočetní nody, na kterých má daná úloha běžet. Pokud nevyberete žádné výpočetní nody, úloha bude spuštěna na nejméně vytížených výpočetních nodech, které jsou dostupné.

Tlačítkem Submit přidejte novou výpočetní úlohu. Pokud budou dostupné výpočetní zdroje pro běh úlohy, je výpočet spuštěn okamžitě. Pokud výpočetní zdroje dostupné nebudou, bude úloha zařazena do fronty a při uvolnění výpočetních zdrojů bude automaticky spuštěna. V případě nastavených notifikací budete o spuštění a případně ukončení úlohy informováni na zadaný e-mail.

6.2.2. Zobrazení stavu výpočetní úlohy

Úlohy můžete filtrovat podle jejich stavu nebo vlastníka pomocí levého menu v Job Manageru.

Základní stav úlohy je možné vidět již v samotném výpisu úloh klastru ve sloupci State. Pokud chcete zobrazit bližší podrobnosti o úloze a výpočetních procesech úlohy, dvojklikem na vybranou úlohu si můžete otevřít podrobnosti.

Na záložce Job Progress můžete vidět stav otevřené úlohy a případně zprávy úlohy s podrobnostmi o aktuálním stavu.

Záložka Job Details ukazuje počáteční nastavení úlohy.

Na záložce View Tasks můžete zobrazit výpočetní procesy úlohy a vidět podrobnosti jednotlivých procesů.

Záložka Resource Selection umožňuje zobrazit vybrané zdroje pro danou úlohu a záložka Allocated Nodes zobrazuje výpočetní nody přidělené úloze.

6.2.3. Úprava výpočetní úlohy

Chcete-li úlohy upravovat, označte si úlohu ve výpisu úloh a v pravém menu akcí zvolte Modify Job.

Při úpravě výpočetních úloh se možnosti editace liší podle toho, v jakém stavu se úloha nachází. Z pochopitelných důvodů není možné například upravovat věci, které již proběhly, jako například notifikace při spuštění úlohy, když úloha již běží.

6.2.4. Zrušení výpočetní úlohy

Pokud chcete zrušit některou výpočetní úlohu, můžete to provést pomocí Cancel Job v menu akcí.

6.3. Ovládání klastru pomocí PowerShellu

Prostředí Windows PowerShell je založeno na Microsoft .NET Framework a obsahuje integrované příkazy, které se nazývají rutiny. Pomocí Windows PowerShell je možné spravovat jak lokální počítač, tak i vzdálené počítače.

Windows PowerShell kromě základních rutin umožňuje i skriptování a pokročilé parsování, díky čemuž je možné procesy automatizovat a upravovat. Konkrétně lze například jednotlivé skripty pro práci s úlohami propojovat, kdy jeden skript bude pracovat s jiným skriptem, plánovat spouštění skriptů a na nich navázaných úloh v plánovači událostí, automaticky reagovat na stav ostatních úloh apod.

Jednotlivé části Windows PowerShellu jsou rozděleny do tématických celků, které se nazývají cmdlety[25].

Kromě ovládání klastru pomocí grafického nástroje HPC Job Manager je možné klastr ovládat i pomocí cmdletů PowerShellu. Ovládání pomocí cmdletů má rozšířené možnosti oproti ovládání přes grafické rozhraní nástroje HPC Job Manager nebo Windows HPC Server Web Portal a je tudíž vhodné pro definování pokročilejších parametrů úloh.

Windows PowerShell je možné spustit přímo na hlavním nodu. První možností je spuštění přímo pomocí HPC PowerShellu ve složce Microsoft HPC Pack 2008 R2 v nabídce Start, alternativně pomocí běžného Windows PowerShellu a načtením cmdletu Add-PsSnapin Microsoft.HPC pro ovládání HPC klastru. Další možností je spuštění Windows PowerShell na vlastním počítači zařazeném do fakultní Active Directory, připojením na vzdálený server pomocí Enter-PSSession -ComputerName hpc-inf a načtením cmdletu Add-PsSnapin Microsoft.HPC pro ovládání HPC klastru. Poslední možností je spuštění připraveného skriptu. Skript je možné spustit z prostředí Windows PowerShell nebo pomocí možnosti Run with PowerShell po kliknutí pravým tlačítkem myši na skript.

Na hlavním nodu je nainstalovaný software PowerShell Server, který umožňuje připojení pomocí SSH (verze 2.0) a je tedy vhodný pro multiplatformní textové ovládání klastru.

Adresa SSH serveru je hpc.inf.upol.cz, port standardní 22. Připojení je z bezpečnostních důvodů možné pouze ze sítě univerzity (IP rozsah 158.194.0.0/16). Pro přihlášení slouží zaměstnanecký univerzitní účet, který se používá i pro přihlášení na vzdálenou plochu serveru nebo na počítače na učebnách. Při přihlášení je nutné zadat i doménu PRFAD, přihlašovací jméno bude tedy ve tvaru prfad\login. Po přihlášení je dostupný klasický Windows PowerShell.

Součástí PowerShell Serveru je i SCP[26] a SFTP[27] server. SFTP má domovský adresář nastavený na C:\a běží na portu 22. SCP je druhou možností pro

[25] Jednoduchý skript v prostředí Windows PowerShell

[26] Secure copy, zabezpečený přenos dat mezi počítači založený na SSH

[27] SSH File Transfer Protocol, pokročilejší verze SCP umožňující nejen přenos dat, ale i pokročilejší práci s daty

kopírování dat, běží také na portu 22.

Z licenčních důvodů je možné pouze jedno současné SSH (a tedy i SCP a SFTP) spojení na server. Při pokusu o další spojení nebude spojení se serverem navázáno. Z tohoto důvodu je také na SSH serveru nastaven Idle Session Timeout na 20 minut, aby se zamezilo nechtěnému blokování dalších spojení.

Všechny předpřipravené PowerShell skripty se nachází na serveru hpc-inf ve složce C:\PowerShell scripts.

Skripty v případě potřeby můžete editovat například v poznámkovém bloku nebo jiném textovém editoru podle referenční příručky. Při ukládání nezapomeňte zachovat příponu souboru ps1. Před úpravou skriptu si ho nejprve zkopírujte do jiného vlastního umístění. Z důvodu zachování předpřipravených skriptů pro ostatní uživatele klastru jsem zakázal editaci ve složce C:\PowerShell scripts.

Některé skripty je možné spouštět přímo z PowerShellu i se zadáním všech hodnot, které skript vyžaduje ke své činnosti. Pokud je skript spuštěn bez parametrů, spustí se jednoduchý průvodce, ve kterém uživatel vyplní požadované informace. Aby bylo možné skripty plnohodnotně využít i v rámci jiných skriptů a navazovat na ně další automatické akce, přidal jsem u většiny skriptů možnost spuštění skriptu s parametry, kde tyto parametry plně nahrazují průvodce a v případě správného vyplnění všech parametrů skript proběhne bez dotazování v průvodci. Bližší informace jsou popsané přímo u jednotlivých skriptů v referenční příručce.

6.3.1. Kopírování dat

Pro zkopírování dat na síťové úložiště nebo kopírování výsledků do vlastního počítače slouží složka Jobs na disku C: hlavního nodu. Tato složka je pomocná pro kopírování přes SFTP/SCP, jinak je samozřejmě možné připojit přímo síťovou složku hpc na serveru storage-inf. Po nakopírování dat z počítače do složky Jobs přes SCP/SFTP je možné v PowerShellu (tedy i SSH) připojit síťovou jednotku hpc pomocí příkazu New-PSDrive -Name H -Root \\storage-inf\hpc$ -PSProvider FileSystem -Credential login, kde místo login zadejte své uživatelské jméno včetně domény prfad ve tvaru prfad\login. Následně budete vyzváni k zadání hesla. Po zadání hesla je jednotka připojena pod písmenem H, stačí tedy zadat H: a aktuální adresář se změní na jednotku H:, kde je namapovaná síťová jednotka hpc. Data je možné přesouvat pomocí Copy-Item -Path zdroj -Destination cíl.

6.3.2. Vytvoření nové výpočetní úlohy

Pro vytvoření nové výpočetní úlohy využijte předpřipraveného skriptu NewHPCJob ve složce C:\PowerShell scripts na hlavním nodu.

Ve skriptu si v prvním kroku nastavte název nově vytvářené úlohy. Název je nepovinný.

Skript je nastavený, aby zaslal notifikace při spuštění i ukončení výpočetní úlohy, a aby využíval všechny výpočetní nody, pokud jsou dostupné.

Skript můžete spustit přímo z okna PowerShellu nebo pomocí možnosti Run with PowerShell po kliknutí pravým tlačítkem myši na skript.

6.3.3. Přidání výpočetního procesu k úloze

Dříve vytvořená výpočetní úloha nemá definované žádné výpočetní procesy a nachází se ve stavu Configuring.

Výpočetní proces můžete k úloze přidat pomocí předpřipraveného skriptu NewHPCTask ve složce C:\PowerShell scripts na hlavním nodu.

V prvním kroku si nastavte název nově vytvářeného procesu. Poté vyberte podle výpisu všech úloh na klastru úlohu, ke které chcete výpočetní proces přidat, a zadejte její ID. V dalším kroku zadejte název pracovního adresáře a případně relativní cestu vzhledem ke sdílené jednotce hpc. Dalším krokem je zadání jména spustitelného souboru výpočetního procesu a případně relativní cesty vzhledem k pracovnímu adresáři. Poslední dva kroky slouží k zadání názvu výstupního souboru a názvu souboru pro chybový výstup včetně případné relativní cesty vzhledem k pracovnímu adresáři.

Ve skriptu je přednastavené, aby využíval všechny dostupné výpočetní nody.

Skript můžete spustit přímo z okna PowerShellu nebo pomocí možnosti Run with PowerShell po kliknutí pravým tlačítkem myši na skript.

6.3.4. Spuštění výpočetní úlohy

Dříve zadanou výpočetní úlohu včetně výpočetních procesů je nutné ještě spustit, jelikož úloha se nachází ve stavu Configuring. Pro spuštění výpočetní úlohy slouží skript SubmitHPCJob ve složce C:\PowerShell scripts na hlavním nodu.

V tomto skriptu zadejte pouze ID úlohy podle výpisu všech úloh na klastru, kterou chcete spustit.

6.3.5. Vytvoření jednoduché výpočetní úlohy

Pro vytvoření nové jednoduché výpočetní úlohy využijte předpřipraveného skriptu NewSingle-TaskJob ve složce C:\PowerShell scripts na hlavním nodu.

Ve skriptu si v prvním kroku nastavte název nově vytvářené úlohy. Název je nepovinný.

V dalším kroku zadejte název pracovního adresáře a případně relativní cestu vzhledem ke sdílené jednotce hpc. Dalším krokem je zadání jména spustitelného souboru výpočetního procesu a případně relativní cesty vzhledem k pracovnímu adresáři. Poslední dva kroky slouží k zadání názvu výstupního souboru a názvu souboru pro chybový výstup včetně případné relativní cesty vzhledem k pracovnímu adresáři.

Skript je nastavený tak, aby zaslal notifikace při spuštění i ukončení výpočetní úlohy, a aby využíval všechny výpočetní nody, pokud jsou dostupné. Úloha je automaticky spuštěna.

Skript můžete spustit přímo z okna PowerShellu nebo pomocí možnosti Run with PowerShell po kliknutí pravým tlačítkem myši na skript.

6.3.6. Zjištění informací o výpočetní úloze

Ke zjištění stavu o úloze slouží skript GetHPCJob ve složce C:\PowerShell scripts na hlavním nodu, který vypíše všechny úlohy na klastru.

```
Id  Name    State       Owner           Priority  NumberOfTasks
--  ----    -----       -----           --------  -------------
379 s~Canceled    PRFAD\beralu00  Normal    1
380 ukazka  Finished    PRFAD\beralu00  Normal    1
381 neco    Running     PRFAD\beralu00  Normal    2
382 s~Failed      PRFAD\beralu00  Normal    0
```

6.3.7. Zrušení výpočetní úlohy

Výpočetní úlohu můžete zrušit pomocí skriptu StopHPCJob ve složce C:\PowerShell scripts na hlavním nodu.

Ve skriptu zadejte ID výpočetní úlohy podle výpisu všech úloh na klastru, kterou chcete zrušit. Úloha bude ukončena násilně.

6.3.8. Zjištění stavu a vytíženosti klastru

Aktuální stav klastru a využití jednotlivých výpočetních nodů můžete zjistit pomocí skriptu GetHPCMetric ve složce C:\PowerShell scripts na hlavním nodu.

Pokud je skript spuštěn ze vzdálené plochy serveru, musí být skript spuštěn z okna PowerShellu s elevovanými právy, tedy pravým tlačítkem myši na aplikaci Windows PowerShell a zvolit možnost Run as administrator. Pokud je skript spuštěn ze vzdáleného počítače, ať už přes SSH nebo vzdálený PowerShell, není potřeba před spuštěním provádět žádnou akci, skript je spouštěn s elevovanými právy automaticky.

```
NodeName      Metric            Counter               Value
--------      ------            -------               -----
              HPCSchedulerCores Number of busy cores  6
              HPCSchedulerCores Number of idle cores  68
...
77-5003-01-PC HPCCoresInUse                           4
77-5003-01-PC HPCJobsRunning                          1
77-5003-01-PC HPCTasksRunning                         1
77-5003-02-PC HPCCoresInUse                           2
77-5003-02-PC HPCJobsRunning                          1
77-5003-02-PC HPCTasksRunning                         1
77-5003-03-PC HPCCoresInUse                           0
77-5003-03-PC HPCJobsRunning                          0
77-5003-03-PC HPCTasksRunning                         0
...
```

6.3.9. Zobrazení informací o výpočetních nodech

Zobrazení informací o výpočetních nodech a o jejich stavu je možné pomocí skriptu GetNodeInformation ve složce C:\PowerShell scripts na hlavním nodu.

Pokud je skript spuštěn ze vzdálené plochy serveru, musí být skript spuštěn z okna PowerShellu s elevovanými právy, tedy pravým tlačítkem myši na aplikaci Windows PowerShell a zvolit možnost Run as administrator. Pokud je skript spuštěn ze vzdáleného počítače, ať už přes SSH nebo vzdálený PowerShell, není potřeba před spuštěním provádět žádnou akci, skript je spuštěn s elevovanými právy automaticky.

```
NetBiosName   HealthState NodeState ProcessorCores  Memory
-----------   ----------- --------- --------------  ------
77-5003-01-PC Error       Online    4               8151
77-5003-02-PC Error       Online    4               8151
77-5003-03-PC OK          Online    4               8151
77-5003-04-PC OK          Online    4               8151
...
```

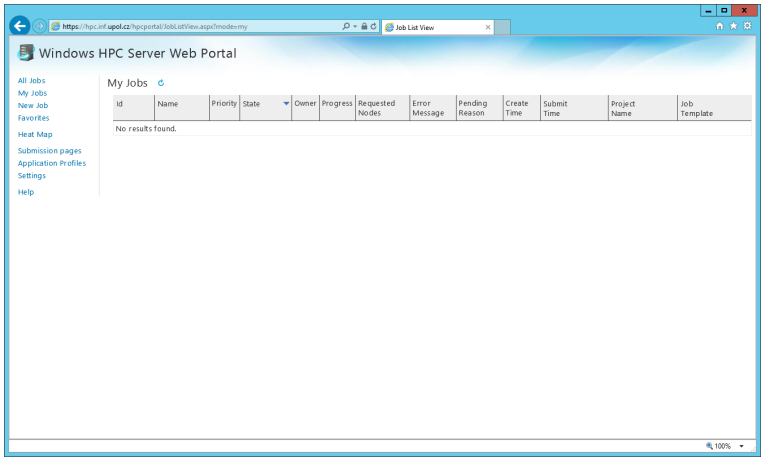
6.4. Ovládání klastru pomocí webového portálu

Pomocí webového rozhraní klastru je možné sledovat běžící výpočetní úlohy a přidávat nové úlohy. Jedná se o plnohodnotnou náhradu HPC Job Manageru co se týče přidávání výpočetních úloh a sledování jejich stavu. Není možné však třídit výpočetní úlohy podle jejich stavu, editovat a kopírovat výpočetní úlohy a procesy a rušit výpočetní úlohy.

Adresa webového rozhraní klastru je hpc.inf.upol.cz a je dostupná pouze ze sítě univerzity (158.194.0.0/16). Webové rozhraní je zabezpečené a běží na protokolu HTTPS s certifikátem podepsaným certifikační autoritou Terena SSL CA.

Pro přihlášení slouží uživatelský účet pro přístup do počítačové sítě. Pokud se přihlašujete z počítače ve fakultní Active Directory, stačí zadat jméno a heslo. Pokud se přihlašujete z počítače mimo fakultní Active Directory, je nutné zadat i doménu, tedy prfad\login.

V levé části obrazovky se nachází hlavní menu, viz obrázek 3.



Obrázek 3. Windows HPC Server Web Portal

6.4.1. Vytvoření nové výpočetní úlohy

Novou výpočetní úlohu vytvoříte pomocí položky New Job v levém menu a následně pomocí odkazu Full Job.

Možnosti zadání jsou nyní stejné jako u vytvoření úlohy pomocí Job Manageru.

6.4.2. Zobrazení stavu výpočetní úlohy

Všechny vlastní výpočetní úlohy můžete zobrazit pomocí položky My Jobs v levém menu. Základní stav úlohy je možné vidět již v samotném výpisu úloh klastru ve sloupci State. Pokud chcete zobrazit bližší podrobnosti o úloze a výpočetních procesech úlohy, klikněte na ID nebo jméno úlohy.

Zobrazené informace o úlohách jsou stejné jako u Job Manageru.

6.5. Příklad spuštění MPI úlohy na klastru

Jako ukázkový příklad zvolím spuštění MPI úlohy pro výpočet čísla π, který je na přiloženém DVD.

6.5.1. Spuštění úlohy pomocí skriptů v PowerShellu

V následujícím příkladu předpokládám spouštění úlohy z vlastního počítače, který není zařazený ve fakultní Active Directory, a který používá libovolný operační systém s SSH klientem. Počítač se však musí nacházet v univerzitní síti, ať už fyzicky nebo pomocí VPN[28] připojení.

V SSH klientovi vytvořte nové připojení. Adresa serveru je hpc.inf.upol.cz, port standardní 22. Při prvním připojení bude nutné potvrdit, že se připojujete k serveru s neznámým klíčem.

Jako uživatelské jméno zadejte svůj účet, kterým se přihlašujete na počítače ve fakultní Active Directory. Účet se zadává i se jménem domény, tedy ve tvaru prfad\login. Dále zadejte heslo ke svému účtu.

Po přihlášení máte automaticky načtený Windows PowerShell.

Pro zkopírování dat na síťové úložiště nebo kopírování výsledků slouží složka Jobs na disku C: hlavního nodu. Zkopírujeme tedy soubor PiCalculator.exe do této složky pomocí SFTP. Poté je možné v PowerShellu (tedy i SSH) připojit síťovou jednotku hpc pomocí příkazu New-PSDrive -Name H -Root \\storage-inf\hpc$ -PSProvider FileSystem -Credential login, kde místo login zadejte své uživatelské jméno včetně domény prfad ve tvaru prfad\login. Zadávám tedy *New-PSDrive -Name H -Root \\storage-inf\hpc$ -PSProvider FileSystem -Credential prfad\beralu00*. Následně jsme vyzváni k zadání hesla. Po zadání hesla je jednotka připojena pod písmenem H, stačí tedy zadat *H:* a aktuální adresář se změní na jednotku H:, kde je namapovaná síťová jednotka hpc. Data je možné přesouvat pomocí Copy-Item -Path zdroj -Destination cíl, v mém případě tedy *Copy-Item -Path C:\Jobs\PiCalculator.exe -Destination beralu00*.

Nyní si můžeme vytvořit novou výpočetní úlohu. K tomu použijeme skript NewHPCJob, který spustíme pomocí *NewHPCJob.ps1*. Jméno úlohy nastavíme například na *pi*. Tím je úloha vytvořena.

V dalším kroku přidáme k nově vytvořené úloze výpočetní proces pomocí skriptu NewHPCTask zadáním *NewHPCTask.ps1*. Nejprve je nutné zadat jméno výpočetního procesu, nastavíme tedy například *vypocet pi*. Dále vybereme a napíšeme ID úlohy, ke které chceme výpočetní proces přidat. Nově přidané úlohy jsou na konci seznamu. Dalším krokem je zadání cesty pro pracovní adresář úlohy relativně vzhledem ke sdílené jednotce hpc, zadávám tedy svůj adresář *beralu00*. Dále zadáme název spustitelného souboru výpočtu včetně relativní cesty vzhledem k pracovnímu adresáři a parametrů. Zadáme tedy *mpiexec PiCalculator.exe 1000000000*, aby výpočet probíhal paralelně na více nodech (mpiexec na začátku) a s 1 000 000 000 iterací. Výstupní soubor v dalším kroku nastavíme na *out.txt* a chybový výstup na *err.txt*. Tím je výpočetní proces přidán k úloze.

[28] Virtuální privátní síť, propojení privátních sítí skrz veřejnou síť

Zbývá už jen úlohu spustit na klastru. K tomu slouží skript SubmitHPCJob, který spustíme zadáním *SubmitHPCJob.ps1*, kde napíšeme ID úlohy, kterou chceme spustit, a potvrdíme. Tím je výpočet spuštěn, pokud jsou dostupné výpočetní nody, jinak je zařazen do fronty na spuštění. Při spuštění i ukončení výpočtu přijde na univerzitní e-mail potvrzení o změně stavu úlohy.

Stav můžeme zkontrolovat pomocí skriptu GetHPCJob spuštěním *GetHPCJob.ps1*.

```
Id  Name    State       Owner           Priority  NumberOfTasks
--  ----    -----       -----           --------  -------------
379 s~Canceled    PRFAD\beralu00  Normal    1
380 ukazka  Finished    PRFAD\beralu00  Normal    1
381 neco    Running     PRFAD\beralu00  Normal    2
382 s~Failed      PRFAD\beralu00  Normal    0
383 pi      Queued      PRFAD\beralu00  Normal    1
```

6.5.2. Spuštění úlohy pomocí HPC Job Manageru

V následujícím příkladu předpokládám spouštění úlohy z vlastního počítače, který není zařazený ve fakultní Active Directory, a který používá libovolný operační systém s klientem pro vzdálenou plochu Microsoft Windows (protokol RDP). Počítač se však musí nacházet v univerzitní síti, ať už fyzicky nebo pomocí VPN připojení.

V klientovi vytvořte nové připojení. Adresa serveru je hpc.inf.upol.cz, port standardní 3389.

Jako uživatelské jméno zadejte svůj účet, kterým se přihlašujete na počítače ve fakultní Active Directory. Účet se zadává i se jménem domény, tedy ve tvaru prfad\login. Dále zadejte heslo ke svému účtu.

V nabídce Start najdeme HPC Job Manager, který spustíme. V pravé horní části okna vybereme možnost New Job.

V novém okně zadáme jméno úlohy *pi*, označíme odeslání notifikací při startu i ukončení úlohy a jako Job resources ponecháme Node.

V levém menu se přepneme na záložku Edit Tasks, kde klikneme na tlačítko Add. Jako jméno vyplníme *vypocet pi*, do políčka Command line napíšeme *mpiexec PiCalculator.exe 1000000000*, abychom úlohu spustili paralelně na více výpočetních nodech, a aby výpočet čísla π probíhal s 1 000 000 000 iterací. Working directory nastavíme na síťový adresář hpc a vlastní uživatelskou složku, ve které má výpočet probíhat, v mém případě tedy *\\storage-inf\hpc$\beralu00*. Standard output nastavíme na *out.txt* a Standard error nastavíme na *err.txt*. Maximální počet zdrojů přidělených výpočetnímu procesu nastavíme na 18 a potvrdíme.

Úlohu spustíme tlačítkem Submit. Stav úlohy vidíme přímo v okně My Jobs (případně All Jobs).

6.6. Vytváření vlastních MPI úloh

Jako šablona pro vytváření vlastních MPI úloh může být použita ukázková úloha na výpočet čísla π napsaná v jazyce `C`, která je součástí přiloženého DVD v adresáři src a podadresáři PiCalculator.

Při vytváření MPI úloh ve Visual Studiu je důležité, aby ve vlastnostech projektu v části C/C++ – General settings byla přidána složka C:\Program Files\Microsoft HPC Pack 2008 R2\Inc do Additional Include Directories, aby kompilátor věděl, kde hledat hlavičkový soubor mpi.h.

Dále ve vlastnostech projektu přidejte v části Linker – General cestu C:\Program Files\Microsoft HPC Pack 2008 R2\Lib\i386 (případně amd64 pro 64bit programy) do Additional Library Directories, aby linker věděl, kde hledat MPI knihovnu.

Jako poslední přidejte v Linker – Input msmpi.lib do Additional Dependencies, aby linker věděl, že musí použít MPI.

Složka C:\Program Files\Microsoft HPC Pack 2008 R2 je součástí instalace Microsoft HPC Pack 2008 R2, který je také na přiloženém DVD. Případně je možné využít pro vývoj aplikací počítače na učebně 5.003, kde je Microsoft HPC Pack 2008 R2 nainstalovaný a dostupné je i Visual Studio.

6.7. Šablony výpočetních úloh

Šablony umožňují definovat výchozí nastavení úloh. Šablona tedy má stejné možnosti nastavení jako samotná vytvářená úloha. Pokud je úloha vytvořena z šablony, jsou u úlohy nastaveny hodnoty nastavené v šabloně. Použití šablon je tedy vhodné tehdy, pokud uživatel vytváří úlohy se stejným nebo velice podobným nastavením.

Šablony výpočetních úloh může vytvářet pouze správce klastru. Pokud někdo bude mít zájem o vytvoření vlastní šablony výpočetních úloh, musí požádat správce klastru o vytvoření. Následně je možné nastavit oprávnění pro konkrétní uživatele na jednotlivé šablony, aby si je mohli sami spravovat.

6.8. Další informace

E-mailové notifikace nefungují, pokud příjemce nemá schránku na Microsoft Exchange, jak jsem již zmiňoval v kapitole 4.5.4.. Zaměstnanecký poštovní účet je však možné nechat na Microsoft Exchange převést.

Jeden výpočetní proces ve výchozím stavu může běžet pouze na jednom výpočetním nodu, jelikož výpočetní klastry nepracují se sdílenou operační pamětí, která je využívána u superpočítačů, jak jsem již zmiňoval v kapitole 2.4.. Pro paralelizaci výpočtů na více výpočetních nodů je potřeba vytvořit více samostatných procesů (i v rámci jedné výpočetní úlohy), které poté mohou běžet na více výpočetních nodech, nebo výpočetní proces paralelizovat pomocí MPI a v rámci každého výpočetního nodu použít vlákna a úlohu poté nespouštět s asociací na procesorová jádra (Cores), ale s asociací na celé výpočetní nody (Nodes), jak jsem již uváděl u příkladů, a před samotným názvem výpočetního procesu musí být uveden název aplikace *mpiexec*, kdy tato aplikace zajistí běh na více výpočetních nodech. V rámci jednoho výpočetního nodu tedy budou použita vlákna, díky čemuž budou využita všechna procesorová jádra tohoto nodu. O paralelizaci a komunikaci mezi výpočetními nody se poté postará aplikace mpiexec s využitím MPI ve výpočetním procesu.

Při výpadku uzlu z klastru dojde k násilnému ukončení výpočtu. Výpočet tedy končí ve stavu Failed a uživatel je o této skutečnosti informován e-mailovou notifikací, pokud byla u úlohy povolena. Samotný restart úlohy není v základu možný. Díky PowerShellu je ale možné takovou akci vytvořit například v plánovači událostí, kdy se v pravidelných intervalech bude kontrolovat stav výpočtu, a pokud se změnil na Failed, skript by zareagoval zkopírováním této výpočetní úlohy a opětovným spuštěním.

Za tímto účelem jsem tedy jeden takový skript připravil. Po spuštění nové výpočetní úlohy může uživatel spustit skript DefineRestartHPCJob.ps1, ve kterém nastaví ID úlohy, kterou chce sledovat. Tento skript je nutné z důvodu zásahu do plánovače událostí spouštět z PowerShellu s elevovanými právy. Skript následně přidá do plánovače na serveru naplánovanou úlohu, která v pravidelných hodinových intervalech bude kontrolovat stav zadané výpočetní úlohy. V případě, že se úloha ocitne ve stavu Failed, dojde k automatickému zkopírování a spuštění výpočetní úlohy znovu s původním nastavením. Pokud se výpočetní úloha ocitne ve stavu Finished, skript se postará o odstranění naplánované úlohy z plánovače.

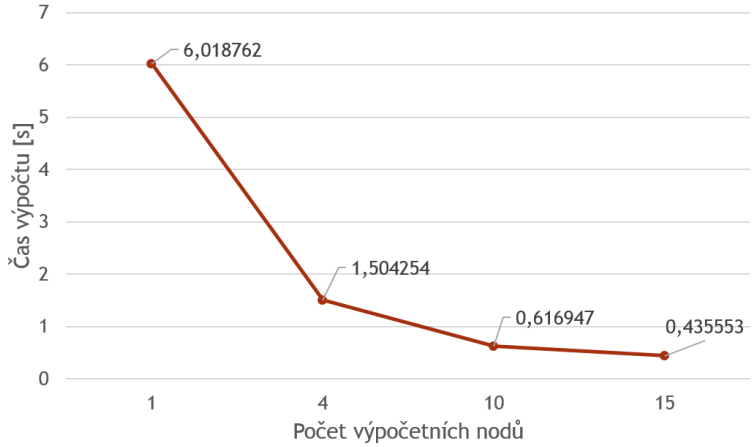
7. Porovnání výsledků výpočtů na klastru

K demonstraci síly klastru a možností jeho využití jsem použil jednoduchou MPI úlohu pro výpočet čísla π s 1 000 000 000 iterací. Jak je vidět z obrázku 4., při využití většího počtu výpočetních nodů dochází k téměř ideálnímu zkrácení doby výpočtu vzhledem k počtu použitých výpočetních nodů.

Zdrojový kód testovací úlohy v jazyce `C` je součástí přiloženého DVD. Tento kód je možné použít i jako vzor pro implementaci vlastních MPI úloh.

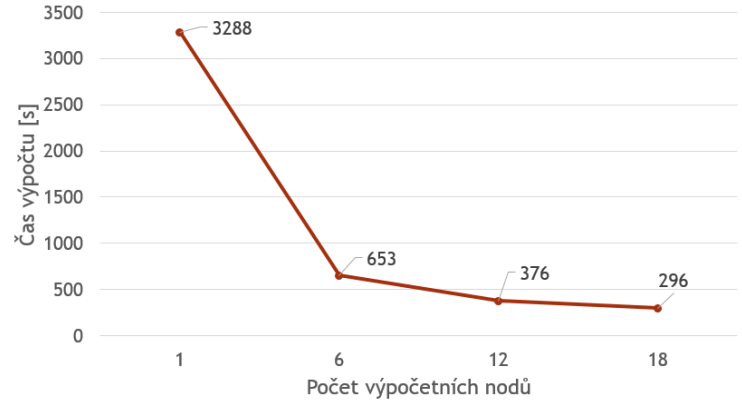
Druhým příkladem je tvorba a naučení neuronové sítě typu SOM[29] o velikosti

[29]Self-Organizing Maps, známé také jako Kohonenovy mapy, které patří do skupiny samou-



Obrázek 4. Výpočet čísla π s 1 000 000 000 iterací

100 x 100, který je vidět na obrázku 5.. U tohoto příkladu se pracuje s externím souborem s daty o velikosti ∼10 MB. Na příkladu je vidět již ne zcela ideální zrychlení výpočtu jako v případě výpočtu čísla π, jelikož se zde pracuje se souborem, který se musí synchronizovat mezi výpočetními nody, což stojí nějaký čas. I tak je ale zrychlení celého výpočtu s přidáním dalších výpočetních nodů velmi citelné, tudíž i v tomto případě je výpočetní klastr přínosem. U tohoto typu úloh je však vhodné nastavit výpočet tak, aby k synchronizaci docházelo co nejméně často, tedy aby si výpočetní nody vzaly dostatečné množství dat, se kterými poté budou co nejdelší dobu pracovat.



Obrázek 5. Tvorba a naučení neuronové sítě typu SOM

Spustitelný soubor výpočtu neuronové sítě i s ukázkovými daty je na přiloženém DVD. Soubor s daty generator.txt se předává jako argument ke spustitelnému souboru výpočtu SOM.exe. Ve stejném adresáři musí být i přiložená

čících se neuronových sítí

knihovna MPI.dll, jelikož výpočet je určen pro MPI.NET.

Závěr

Cílem práce bylo vytvoření výpočetního klastru na jedné z počítačových učeben, který by umožňoval využít dostupný výpočetní výkon počítačů na učebně v době, kdy neprobíhá výuka.

Výpočetní klastr je možné ovládat pomocí grafického rozhraní na vzdálené ploše hlavního nodu klastru, kde je možné sledovat stav běžících úloh, zadávat nové výpočetní úlohy a další věci popsané v kapitole 6.. Další možností ovládání klastru je pomocí webového rozhraní, které poskytuje téměř stejné možnosti jako grafické rozhraní na vzdálené ploše hlavního nodu. Poslední možností ovládání je pomocí PowerShellu s využitím HPC cmdletů. Tento způsob ovládání poskytuje nejširší možnosti, ale je také nejméně komfortní. Z tohoto důvodu je součástí práce sada 15 předpřipravených skriptů, které by měly alespoň základní práci s klastrem usnadnit a poskytnout šablonu pro další rozšiřování. Dokumentace ke skriptům včetně dalších možností pro vlastní rozšíření je rovněž součástí práce.

Hibernaci výpočetních úloh při vyjmutí uzlu (nodu) z klastru podle zadání práce nebylo možné v rámci bakalářské práce splnit, jelikož použité řešení Microsoft HPC tuto akci nepodporuje v žádné verzi a celé řešení Microsoft HPC je uzavřené a tudíž není možné ho dále rozšiřovat. Ze stejného důvodu není možné ani přesouvat běžící výpočty mezi výpočetními uzly. Bližší informace jsou popsány v kapitole 5.12.. Řešením je pouze podpora události CTRL_BREAK přímo ve výpočetním procesu, na kterou proces zareaguje uložením svého stavu a řádným ukončením výpočtu.

Aktuálně je výpočetní klastr složený z 18 pracovních stanic na jedné počítačové učebně. V budoucnu je však možné výpočetní klastr rozšířit na další počítače v učebnách, případně do klastru zařadit trvale dostupné výpočetní servery a dále tím zvyšovat výpočetní výkon a možnosti využití.

Reference

[1] Buyya, Rajkumar. *High Performance Cluster Computing: Architectures and Systems.*
ISBN 0-13-013784-7, Prentice Hall PTR, NJ, USA, 1999.

[2] Blaise Barney, Lawrence Livermore National Laboratory. *Message Passing Interface (MPI).*
https://computing.llnl.gov/tutorials/mpi/, 2013.

[3] Kmuníček, Jan. *Gridy jako klíčový fenomén informačních technologií nového tisíciletí.*
Zpravodaj ÚVT MU, 2005.

[4] Xen.org. *Xen.org.*
http://xen.org, 2013.

[5] Citrix Systems, Inc. *Citrix XenServer.*
http://www.citrix.com/products/xenserver, 2013.

[6] Red Hat, Inc. *Red Hat Enterprise Virtualization.*
http://www.redhat.com/products/virtualization, 2013.

[7] VMware, Inc. *VMware vSphere.*
http://www.vmware.com/cz/products/datacenter-virtualization/vsphere, 2013.

[8] Microsoft Corporation. *Microsoft Virtualization.*
http://www.microsoft.com/en-us/server-cloud/virtualization/default.aspx, 2013.

[9] Red Hat, Inc. *Red Hat Enterprise Linux for Scientific Computing.*
http://www.redhat.com/products/enterprise-linux/scientific-computing, 2013.

[10] ParaTools, Inc. *ParaTools HPC Linux.*
http://www.paratools.com/HPCLinux, 2013.

[11] SUSE Linux GmbH. *SUSE Linux Enterprise Server.*
https://www.suse.com/products/server/hpc.html, 2013.

[12] Universitat Autonoma de Barcelona. *PelicanHPC GNU Linux.*
http://pareto.uab.es/mcreel/PelicanHPC, 2013.

[13] Microsoft Corporation. *Microsoft HPC Pack.*
http://www.microsoft.com/hpc, 2013.

[14] Microsoft Corporation. *Microsoft HPC Pack – TechNet Library.* http://technet.microsoft.com/en-us/library/cc514029.aspx, 2013.

[15] Microsoft Corporation. *What Is Windows Communication Foundation – MSDN Library.* http://msdn.microsoft.com/en-us/library/ms731082.aspx, 2013.

[16] Microsoft Corporation. *Windows HPC Cmdlets for Windows PowerShell – TechNet Library.* http://technet.microsoft.com/en-us/library/ff950195.aspx, 2013.

A. Referenční příručka – HPC Job Manager

Vytvoření nové výpočetní úlohy se základním nastavením

Novou jednoduchou výpočetní úlohu můžeme vytvořit pomocí volby New Single-Task Job v nabídce akcí na pravé straně HPC Job Manageru. Tato volba je vhodná tehdy, pokud chceme vytvořit jen jednoduchou úlohu s jedním výpočetním procesem, kde nepožadujeme bližší možnosti specifikace.

V nově otevřeném okně aplikace můžeme zvolit šablonu Job Template, podle které má být úloha vytvořena. Další možností je nastavení e-mailových notifikací s volbou, jestli má být notifikace odeslána při spuštění výpočtu a/nebo při ukončení výpočtu. Výchozí e-mailová adresa pro notifikace je načtena z informací u přihlášeného účtu. Tuto adresu je možné změnit v sousedním políčku.

Do políčka Task name můžeme zadat název výpočetní úlohy. Název jednotlivých výpočetních procesů není v tomto případě možné zadat. Název může zůstat prázdný.

V políčku Command line specifikujeme název spustitelného souboru výpočtu včetně případných parametrů. Název spustitelného souboru musí být definovaný. Pokud chceme spustit paralelní úlohu, přidáme před název spustitelného souboru *mpiexec*.

Working directory je umístění, ve kterém bude výpočetní úloha pracovat a od tohoto umístění se udávají další relativní cesty včetně spustitelného souboru výpočetního procesu zadaném v políčku Command line. Working directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Do Standard input je možné zadat relativní cestu vzhledem k Working directory pro stdin soubor procesu.

Standard output slouží k zadání relativní cesty vzhledem k Working directory pro zadání stdout souboru procesu.

Standard error umožňuje zadat relativní cestu vzhledem k Working directory pro stderr soubor procesu.

Posledním nastavením je zadání počtu výpočetních nodů, které mají být výpočetní úloze neexkluzivně přiděleny.

Vytvoření nové výpočetní úlohy s pokročilým nastavením

Chceme-li vytvořit úlohu s pokročilejším nastavením parametrů běhu, případně chceme-li v rámci jedné úlohy mít víc výpočetních procesů, využijeme volbu New Job v pravém menu akcí.

Job Details V nově otevřeném okně průvodce na záložce Job Details můžeme nastavit jméno úlohy (není povinné), vybereme šablonu pomocí roletky Job Template, z které úlohu chceme vytvářet, a volitelně zadáme jméno projektu, do kterého bude výpočetní úloha patřit.

Priorita úlohy umožňuje nastavit, kolik daná úloha dostane výpočetních prostředků relativně k ostatním úlohám v případě, že na jednom výpočetním nodu běží více výpočetních úloh.

V části Job run options je možné nastavit, jak dlouhou dobu maximálně může úloha běžet (po uplynutí času je úloha násilně ukončena), jestli má úloha běžet až do zrušení nebo uplynutí přiděleného času (úloha po celou dobu drží výpočetní zdroje, i když už výpočet neběží). Další možností je, jestli má být úloha ukončena, pokud havaruje některý výpočetní proces v rámci úlohy. Jako poslední je možné nastavit e-mailové notifikace s volbou zaslání e-mailu při spuštění a/nebo ukončení výpočtu a případně zvolit jinou e-mailovou adresu, na kterou mají být notifikace zaslány.

Ve spodní části záložky Job Details je možné nastavit zdroje pro danou výpočetní úlohu. Výběr zdrojů je dělen do tří skupin – Core (procesorová jádra), Socket (procesorové sokety, v našem případě mají všechny výpočetní nody jeden soket a zadání počtu soketů se tedy rovná zadání počtu nodů) a Node (počet výpočetních nodů). Jeden výpočetní nod má u nás čtyři procesorová jádra. Pro spouštění paralelních úloh se jako zdroj používají nody. Jako poslední je možné zvolit, jestli mají být výpočetní zdroje úloze přiděleny exkluzivně, kdy přidělené zdroje není možné přidělit jiným výpočetním úlohám, dokud úloha zdroje neuvolní.

Edit Tasks Při zadávání jednotlivých výpočetních procesů dané úlohy je možné vybrat z šesti druhů.

Basic Task Basic Task je základní typ výpočetního procesu, který je možné na klastru spouštět. Jedná se o běžný výpočetní proces a tento typ úlohy bude pravděpodobně vhodný pro většinu případů.

Do políčka Task name je možné napsat jméno výpočetního procesu pro následnou snadnější orientaci. Jméno není povinné.

V políčku Command line specifikujeme název spustitelného souboru výpočtu včetně případných parametrů. Název spustitelného souboru musí být definovaný. Pokud je spouštěna paralelní úloha, před samotným názvem výpočetního procesu musí být zadané *mpiexec*.

Working directory je umístění, ve kterém bude výpočetní proces pracovat, a od tohoto umístění se udávají další relativní cesty včetně

spustitelného souboru výpočetního procesu zadaném v políčku Command line. Working directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Do Standard input je možné zadat relativní cestu vzhledem k Working directory pro stdin soubor procesu.

Standard output slouží k zadání relativní cesty vzhledem k Working directory pro zadání stdout souboru procesu.

Standard error umožňuje zadat relativní cestu vzhledem k Working directory pro stderr soubor procesu.

Posledním nastavením je zadání počtu výpočetních zdrojů (typ zdroje se řídí typem zadaným pro celou úlohu), které mají být výpočetnímu procesu neexkluzivně přiděleny.

Parametric Sweep Task Výpočetní procesy typu Parametric Sweep Task slouží pro zadání výpočetních procesů, které se mají opakovat.

Do políčka Task name je možné napsat jméno výpočetního procesu pro následnou snadnější orientaci. Jméno není povinné.

V kroku 1 zadáme první a poslední hodnotu v opakování výpočetního procesu.

Krok 2 umožňuje zadat, o kolik se má v každém opakování hodnota zvednout.

V políčku Command line specifikujeme název spustitelného souboru výpočtu včetně případných parametrů a pomocí hvězdičky (*) zadáme umístění inkrementované hodnoty. Název spustitelného souboru musí být definovaný.

Working directory je umístění, ve kterém bude výpočetní proces pracovat, a od tohoto umístění se udávají další relativní cesty včetně spustitelného souboru výpočetního procesu zadaném v políčku Command line. Working directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Do Standard input je možné zadat relativní cestu vzhledem k Working directory pro stdin soubor procesu.

Standard output slouží k zadání relativní cesty vzhledem k Working directory pro zadání stdout souboru procesu.

Standard error umožňuje zadat relativní cestu vzhledem k Working directory pro stderr soubor procesu.

V kroku 4 vidíme náhled, jak budou jednotlivé výpočetní procesy za sebou spouštěny.

Node Preparation Task Tento typ výpočetní úlohy slouží k přípravě výpočetních nodů na následující výpočty například nakopírováním dat na všechny výpočetní nody, vytvořením adresářů na výpočetních nodech pro potřeby dalších výpočtů apod. Tento typ procesu je spouštěn na všech nodech, které jsou dané úloze přiděleny ještě před tím, než jsou spuštěny zbylé výpočetní procesy v této úloze. Každá úloha může obsahovat pouze jednu Node Preparation Task.

Do políčka Task name je možné napsat jméno výpočetního procesu pro následnou snadnější orientaci. Jméno není povinné.

V políčku Command line specifikujeme název spustitelného souboru výpočtu včetně případných parametrů. Název spustitelného souboru musí být definovaný.

Working directory je umístění, ve kterém bude výpočetní proces pracovat, a od tohoto umístění se udávají další relativní cesty včetně spustitelného souboru výpočetního procesu zadaném v políčku Command line. Working directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Do Standard input je možné zadat relativní cestu vzhledem k Working directory pro stdin soubor procesu.

Standard output slouží k zadání relativní cesty vzhledem k Working directory pro zadání stdout souboru procesu.

Standard error umožňuje zadat relativní cestu vzhledem k Working directory pro stderr soubor procesu.

Node Release Task Node Release Task slouží k „uklizení“ po skončení výpočtu. Tento proces je spuštěn po dokončení výpočtu na všech výpočetních nodech, na kterých běžela daná úloha. Může sloužit například pro odstranění dočasných adresářů nebo souborů z jednotlivých výpočetních nodů nebo pro kopírování výsledků výpočtů do jiného umístění apod. Každá úloha může obsahovat maximálně jeden proces tohoto typu.

Do políčka Task name je možné napsat jméno výpočetního procesu pro následnou snadnější orientaci. Jméno není povinné.

V políčku Command line specifikujeme název spustitelného souboru výpočtu včetně případných parametrů. Název spustitelného souboru musí být definovaný.

Working directory je umístění, ve kterém bude výpočetní proces pracovat, a od tohoto umístění se udávají další relativní cesty včetně

spustitelného souboru výpočetního procesu zadaném v políčku Command line. Working directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Do Standard input je možné zadat relativní cestu vzhledem k Working directory pro stdin soubor procesu.

Standard output slouží k zadání relativní cesty vzhledem k Working directory pro zadání stdout souboru procesu.

Standard error umožňuje zadat relativní cestu vzhledem k Working directory pro stderr soubor procesu.

Service Task Service Task umožňuje hostovat vlastní WCF služby[15]. V rámci jedné úlohy může být se Service Task procesem přidán pouze proces Node Release Task nebo Node Preparation Task.

Do políčka Task name je možné napsat jméno výpočetního procesu pro následnou snadnější orientaci. Jméno není povinné.

V políčku Command line specifikujeme název spustitelného souboru výpočtu včetně případných parametrů. Název spustitelného souboru musí být definovaný.

Working directory je umístění, ve kterém bude výpočetní proces pracovat a od tohoto umístění se udávají další relativní cesty včetně spustitelného souboru výpočetního procesu zadaném v políčku Command line. Working directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Do Standard input je možné zadat relativní cestu vzhledem k Working directory pro stdin soubor procesu.

Standard output slouží k zadání relativní cesty vzhledem k Working directory pro zadání stdout souboru procesu.

Standard error umožňuje zadat relativní cestu vzhledem k Working directory pro stderr soubor procesu.

From Task File Poslední volba From Task File umožňuje zadat výpočetní proces z XML souboru.

Kromě zadávání nových procesů je možné v záložce Edit Tasks také procesy upravovat, kopírovat, mazat, ukládat do XML souborů pro pozdější načtení nebo nastavovat závislosti jednotlivých procesů, kde se procesy seskupují do skupin a výpočet se může do další skupiny procesů přesunout až tehdy, když jsou ukončeny všechny procesy v aktuální skupině.

Resource Selection V části Resource Selection je možné specifikovat, na kterých výpočetních nodech bude úloha spuštěna.

Ve vrchní části okna je možné vybírat skupiny výpočetních nodů (volba Run this job only on nodes that are members of all the following groups) a podle toho filtrovat samotné výpočetní nody. Vzhledem k tomu, že výpočetní nody jsou pouze z jedné počítačové učebny 5.003, nejsou zde definované žádné bližší skupiny výpočetních nodů kromě výchozích ComputeNodes, UnmanagedServerNodes, AzureNodes a WorkstationNodes a všechny výpočetní nody patří do skupiny WorkstationNodes. Tuto volbu tedy nemá smysl vybírat.

Pokud bychom chtěli vybrat konkrétní počítače z učebny 5.003, zatrhneme volbu Run this job only on nodes in the following list a vybereme konkrétní počítače podle jména.

V části Hardware preferences je možné blíže specifikovat, jaké minimální požadavky musí výpočetní nod splňovat, aby na něm mohla být výpočetní úloha spuštěna. Vzhledem k tomu, že všechny výpočetní nody mají stejné hardwarové parametry, tato volba postrádá pro naše potřeby smysl.

Licenses Záložka Licenses slouží k definování názvů a počtů licencí, které jsou pro úlohu potřeba. Tato volba slouží k omezení počtu spuštěných instancí výpočetního procesu, pokud by byl počet instancí omezený licenčně. Ve většině případů tuto položku nebude nutné vyplňovat.

Environment Variables V této záložce je možné definovat proměnné, které chceme používat v kontextu výpočetní úlohy. U této části opět nepředpokládám, že by byla pro výpočetní úlohy využívána.

Nově zadanou úlohu můžeme potvrdit tlačítkem Submit nebo ji uložit do XML souboru tlačítkem Save Job XML File pro pozdější použití.

Vytvoření nové výpočetní úlohy z XML souboru

Další možností vytvoření nové výpočetní úlohy je načtení XML souboru úlohy. XML soubor je možné vytvořit exportováním nějaké úlohy nebo ručním vytvořením.

Vytvoření nové výpočetní úlohy zkopírováním jiné úlohy

Poslední možností vytvoření nové výpočetní úlohy je kopírováním nějaké již vytvořené úlohy pomocí označení úlohy, kterou chceme zkopírovat, a v menu akcí vybráním Copy Job, které otevře okno zadání nové úlohy s předvyplněnými informacemi z kopírované úlohy.

Zobrazení informací o výpočetní úloze

Informace o běžících i ukončených úlohách je možné zobrazit označením úlohy a vybráním View Job v menu akcí. V nově otevřeném okně je možné vidět aktuální stav úlohy včetně zpráv, pokud to bylo zadáno, konfigurační detaily úlohy, informace o běžících procesech (alokované výpočetní nody, počet výpočetních jader přiřazených procesu atd.) a další.

Zde je také možné uložit úlohu do XML souboru pomocí tlačítka Save Job XML File ve spodní části okna.

Úprava běžících výpočetních úloh

Běžící výpočetní úlohy je možné do jisté míry upravovat i za běhu. K tomu slouží tlačítko Modify Job v menu akcí.

U výpočetních úloh je možné například upravovat název, jméno projektu, prioritu výpočetní úlohy, e-mailovou adresu pro notifikace a zasílání notifikací při ukončení úlohy nebo přidávat další výpočetní procesy k úloze.

U běžících výpočetních úloh však nelze upravovat šablonu úlohy, přiřazené výpočetní prostředky nebo upravovat výpočetní procesy úlohy.

B. Referenční příručka – HPC PowerShell

Seznam všech níže uvedených vlastních skriptů pro ovládání výpočetního klastru čerpá z Technet knihovny[16], kde je také dostupný seznam všech cmdletů použitelných v Microsoft HPC Pack 2008 R2.

NewHPCJob

Skript New HPC Job využívá cmdlet New-HpcJob a slouží pro vytvoření nové HPC úlohy.

Skript je možné zavolat s již předdefinovaným jménem úlohy ve formě parametru, tedy NewHPCJob.ps1 jmenoulohy. Pokud je skript zavolaný se jménem úlohy, nespustí se průvodce a úloha s definovaným jménem je rovnou vytvořena.

-Name Parametr -Name specifikuje jméno vytvářené úlohy. Maximální délka je 80 znaků.

-NotifyOnCompletion Parametr -NotifyOnCompletion specifikuje, jestli má být při ukončení úlohy zasláno upozornění na e-mail vlastníka dané úlohy. Jako ukončení úlohy jsou brány stavy Canceled, Failed a Finished. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-NotifyOnStart Parametr -NotifyOnStart specifikuje, jestli má být při spuštění úlohy zasláno upozornění na e-mail vlastníka dané úlohy. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-NumNodes Parametr -NumNodes specifikuje celkový počet výpočetních nodů, na kterých má úloha běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumSockets, není možné nastavit -NumNodes.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-EmailAddress Parametr -EmailAddress slouží pro definování e-mailové adresy, na kterou má být zaslané oznámení o změně stavu definované úlohy. Výchozí adresa pro zasílání oznámení je adresa uživatele v univerzitní databázi. Maximální délka adresy je 256 znaků.

-Exclusive Parametr -Exclusive specifikuje, jestli má být úloha na zvolených výpočetních nodech spuštěna exkluzivně, tedy že žádná jiná úloha na těchto nodech nepoběží. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-FailOnTaskFailure Parametr -FailOnTaskFailure specifikuje, jestli má být celá úloha ukončena, pokud zhavaruje některý výpočetní proces. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-Job Parametr -Job vybírá úlohu, která má být zkopírována do nově vytvářené úlohy. Zadává se ID úlohy. ID úlohy je zobrazeno při zadání úlohy, případně je možné ho dodatečně zjistit pomocí cmdletu Get-HpcJob.

-JobFile Parametr -JobFile specifikuje jméno a cestu XML souboru, který obsahuje nastavení pro danou úlohu.

-NumCores Parametr -NumCores specifikuje celkový počet procesorových jader, na kterých má úloha běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumNodes nebo -NumSockets, není možné nastavit -NumCores.

-NumSockets Parametr -NumSockets specifikuje celkový počet soketů, na kterých má úloha běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumNodes, není možné nastavit -NumSockets.

-Priority Parametr -Priority určuje prioritu, s jakou má být úloha spuštěna. Priorita může být zadána číslem 0 až 4000, kde 0 znamená nejnižší prioritu a 4000 nejvyšší prioritu. Další možností zadání priority je slovní zadání pomocí Highest, AboveNormal, Normal, BelowNormal a Lowest, tyto slovní hodnoty odpovídají číselným hodnotám 4000, 3000, 2000, 1000 a 0. Poslední možností zadání priority je pojmenovanáhodnota+offset nebo pojmenovanáhodnota-offset. Výsledek tohoto zadání ale musí být v rozmezí 0 až 4000. Výchozí hodnota je Normal.

-Project Parametr -Project specifikuje jméno projektu spouštěné úlohy. Jméno projektu se může hodit při sledování stavu úloh. Maximální délka jména projektu je 80 znaků.

-TemplateName Parametr -TemplateName specifikuje jméno šablony použité u dané úlohy. Maximální délka jména je 80 znaků. Výchozím jménem je Default.

SetHPCJob

Skript Set HPC Job využívá cmdlet Set-HpcJob a slouží k nastavení parametrů výpočetní úlohy.

Skript je možné zavolat s již předdefinovaným ID úlohy, jménem, e-mailovou adresou, notifikací po startu úlohy a notifikací po skončení úlohy ve formě parametrů, tedy SetHPCJob.ps1 idulohy jmenoulohy mail notifikacepristartu notifikaciprikonci v tomto pořadí, případně s parametry -Id, -Name, -Mail, -StartNotify a -CompNotify s jejich argumenty v libovolném pořadí, tedy například SetHPCJob.ps1 -Name jmenoulohy -StartNotify $true -CompNotify $false -Id idulohy -Mail mail. Pokud je skript zavolaný s ID úlohy, jménem, e-mailovou adresou, notifikací po startu úlohy i notifikací po skončení úlohy, nespustí se průvodce a výpočetní úloha je upravena podle zadaných parametrů. Pokud je skript spuštěn bez parametrů nebo jen s prvními parametry ve správném pořadí, je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

-Id Parametr -Id slouží ke specifikování ID úlohy, kterou chceme upravit.

-EmailAddress Parametr -EmailAddress slouží k úpravě e-mailové adresy, na kterou mají být posílány upozornění o změně stavu výpočetní úlohy.

-Name Parametr -Name specifikuje jméno vytvářené úlohy. Maximální délka je 80 znaků.

-NotifyOnCompletion Parametr -NotifyOnCompletion specifikuje, jestli má být při ukončení úlohy zasláno upozornění na e-mail vlastníka dané úlohy. Jako ukončení úlohy jsou brány stavy Canceled, Failed a Finished. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-NotifyOnStart Parametr -NotifyOnStart specifikuje, jestli má být při spuštění úlohy zasláno upozornění na e-mail vlastníka dané úlohy. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-NumNodes Parametr -NumNodes specifikuje celkový počet výpočetních nodů, na kterých má úloha běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumSockets, není možné nastavit -NumNodes.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Exclusive Parametr -Exclusive specifikuje, jestli má být úloha na zvolených výpočetních nodech spuštěna exkluzivně, tedy že žádná jiná úloha na těchto nodech nepoběží. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-FailOnTaskFailure Parametr -FailOnTaskFailure specifikuje, jestli má být celá úloha ukončena, pokud zhavaruje některý výpočetní proces. Přípustné hodnoty jsou $true nebo $false. Výchozí hodnota je $false.

-NumCores Parametr -NumCores specifikuje celkový počet procesorových jader, na kterých má úloha běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumNodes nebo -NumSockets, není možné nastavit -NumCores.

-NumSockets Parametr -NumSockets specifikuje celkový počet soketů, na kterých má úloha běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumNodes, není možné nastavit -NumSockets.

-Priority Parametr -Priority určuje prioritu, s jakou má být úloha spuštěna. Priorita může být zadána číslem 0 až 4000, kde 0 znamená nejnižší prioritu a 4000 nejvyšší prioritu. Další možností zadání priority je slovní zadání pomocí Highest, AboveNormal, Normal, BelowNormal a Lowest, tyto slovní hodnoty odpovídají číselným hodnotám 4000, 3000, 2000, 1000 a 0. Poslední možností zadání priority je pojmenovanáhodnota+offset nebo pojmenovanáhodnota-offset. Výsledek tohoto zadání ale musí být v rozmezí 0 až 4000. Výchozí hodnota je Normal.

-Project Parametr -Project specifikuje jméno projektu spouštěné úlohy. Jméno projektu se může hodit při sledování stavu úloh. Maximální délka jména projektu je 80 znaků.

-TemplateName Parametr -TemplateName specifikuje jméno šablony použité u dané úlohy. Maximální délka jména je 80 znaků. Výchozím jménem je Default.

SubmitHPCJob

Skript Submit HPC Job využívá cmdlet Submit-HpcJob a slouží pro spuštění vybrané úlohy na klastru, případně pro zařazení dané úlohy do fronty, pokud není dostatek prostředků pro okamžité spuštění úlohy.

Skript je možné zavolat s již předdefinovaným ID úlohy ve formě parametru, tedy SubmitHPCJob.ps1 idulohy. Pokud je skript zavolaný s ID úlohy, nespustí se průvodce a úloha s definovaným ID je spuštěna na klastru. Pokud je skript spuštěn bez parametru, je spuštěn průvodce, který požaduje vyplnění ID úlohy.

-Id Parametr -Id slouží pro definování ID úlohy, která má být spuštěna nebo zařazena do fronty. ID úlohy je zobrazeno při zadání úlohy, případně je možné ho dodatečně zjistit pomocí cmdletu Get-HpcJob.

GetHPCJob

Skript Get HPC Job využívá cmdlet Get-HpcJob, který slouží k vypsání úloh podle zadaných parametrů nebo úloh, které běží na specifikovaných výpočetních nodech.

-State Parametr -State vypíše výpočetní úlohy podle zadaného stavu. Přípustné hodnoty jsou Configuring, Submitted, Validating, ExternalValidation, Queued, Running, Finishing, Finished, Failed, Canceled, Canceling a All. Výchozí hodnota je Queued,Running.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-BeginSubmitDate Parametr -BeginSubmitDate vypíše všechny výpočetní úlohy, které byly spuštěny po zadaném čase. Čas se zadává v americkém formátu měsíc/den/rok a případně i s časem ve formátu hodiny:minuty:sekundy AM/PM.

-EndSubmitDate Parametr -EndSubmitDate vypíše všechny výpočetní úlohy, které skončily před zadaným časem. Čas se zadává v americkém formátu měsíc/den/rok a případně i s časem ve formátu hodiny:minuty:sekundy AM/PM.

-TemplateName Parametr -TemplateName vypíše výpočetní úlohy podle zadaného jména šablony úloh.

-Project Parametr -Project vypíše všechny výpočetní úlohy patřící zadanému jménu projektu.

-Owner Parametr -Owner vypíše všechny výpočetní úlohy patřící zadanému vlastníkovi. Vlastník úlohy se zadává i s doménou, tedy prfad\login.

-NodeName Parametr -NodeName vypíše všechny výpočetní úlohy, které běží na zadaném výpočetním nodu. Zadává se jméno výpočetního nodu.

-Name Parametr -Name vypíše výpočetní úlohy podle zadaného jména úlohy.

-Id Parametr -Id vypíše výpočetní úlohy podle zadaného ID úlohy.

StopHPCJob

Skript Stop HPC Job využívá cmdletu Stop-HpcJob a slouží k ukončení výpočetní úlohy.

Skript je možné zavolat s již předdefinovaným ID úlohy ve formě parametru, tedy StopHPCJob.ps1 idulohy. Pokud je skript zavolaný s ID úlohy, nespustí se průvodce a úloha s definovaným ID je násilně ukončena. Pokud je skript spuštěn bez parametru, je spuštěn průvodce, který požaduje vyplnění ID úlohy.

-Id Parametr -Id specifikuje ID výpočetní úlohy, kterou chceme ukončit.

-Force Parametr -Force ukončí úlohu okamžitě a násilně bez využití Grace Periody pro řádné ukončení úlohy.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Message Parametr -Message umožňuje nastavit zprávu, která má být zobrazena jako důvod ukončení úlohy.

-State Parametr -State umožňuje nastavit stav ukončené úlohy. Výchozí stav je Canceled a pomocí parametru -State je možné nastavit stav Finished. Žádný jiný stav není možné nastavit.

ExportHPCJob

Skript Export HPC Job využívá cmdlet Export-HpcJob a slouží k exportování výpočetních úloh do XML souboru.

Skript je možné zavolat s již předdefinovaným ID úlohy a cestou pro export ve formě parametrů, tedy ExportHPCJob.ps1 idulohy cesta v tomto pořadí, případně s parametry -UserPath a -Id s jejich argumenty v libovolném pořadí, tedy například ExportHPCJob.ps1 -UserPath cesta -Id idulohy. Pokud je skript zavolaný s ID úlohy i cestou pro export, nespustí se průvodce a úloha s definovaným

ID je exportována do definovaného souboru. Pokud je skript spuštěn bez parametrů nebo s prvním parametrem, je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

-Id Parametr -Id specifikuje ID výpočetní úlohy, kterou chceme exportovat.

-Path Parametr -Path specifikuje jméno souboru a cestu, kam se má výpočetní úloha exportovat. Cesta je zadávána vzhledem k hlavnímu nodu a je doporučené používat síťové umístění.

NewHPCTask

Skript New HPC Task využívá cmdletu Add-HpcTask a slouží k vytvoření nového výpočetního procesu výpočetní úlohy.

Skript je možné zavolat s již předdefinovaným jménem, ID úlohy, pracovním adresářem, jménem spustitelného souboru výpočetního procesu, standardním výstupem a standardním chybovým výstupem ve formě parametrů, tedy NewHPCTask.ps1 jmenoprocesu idulohy pracovniadresar jmenosouboruprocesu vystup chybovyvystup v tomto pořadí, případně s parametry -Name, -JobId, -WorkDir, -AppName, -Out a -Err s jejich argumenty v libovolném pořadí, tedy například NewHPCTask.ps1 -JobId idulohy -WorkDir pracovniadresar -AppName jmenosouboruprocesu -Name jmenoprocesu -Err chybovyvystup -Out vystup. Pokud je skript zavolaný se jménem, ID úlohy, pracovním adresářem i jménem spustitelného souboru výpočetního procesu, nespustí se průvodce a výpočetní proces s definovanými parametry je přidán k definované úloze. Parametry standardního výstupu a standardního chybového výstupu jsou nepovinné. Pokud je skript spuštěn bez parametrů nebo jen s prvními parametry ve správném pořadí, je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

-CommandLine Parametr -CommandLine slouží k definování názvu spustitelného souboru výpočetního procesu a případnému definování parametrů pro tento proces. Název spustitelného souboru je povinný a definuje se spolu s relativní cestou vzhledem k Working Directory.

-JobId Parametr -JobId definuje ID výpočetní úlohy, ke které má být výpočetní proces přidán.

-Name Parametr -Name slouží k nastavení jména výpočetního procesu pro snadnější orientaci.

-NumNodes Parametr -NumNodes specifikuje celkový počet výpočetních nodů, na kterých má proces běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumSockets, není možné nastavit -NumNodes.

-Stderr Parametr -Stderr specifikuje relativní cestu vzhledem k Working Directory a název souboru pro standard error výstup výpočetního procesu.

-Stdout Parametr -Stdout specifikuje relativní cestu vzhledem k Working Directory a název souboru pro standardní výstup výpočetního procesu.

-WorkDir Parametr -WorkDir specifikuje umístění, ve kterém bude výpočetní proces pracovat, a od tohoto umístění se udávají další relativní cesty. Working Directory tedy může mít tvar například \\storage-inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Stdin Parametr -Stdin specifikuje relativní cestu vzhledem k Working Directory a název souboru pro standardní vstup výpočetního procesu.

-NumCores Parametr -NumCores specifikuje celkový počet procesorových jader, na kterých má proces běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumNodes nebo -NumSockets, není možné nastavit -NumCores.

-NumSockets Parametr -NumSockets specifikuje celkový počet soketů, na kterých má proces běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumNodes, není možné nastavit -NumSockets.

-Depend Parametr -Depend umožňuje nastavit závislosti mezi procesy. Nově vytvářený proces nebude spuštěn dřív, než bude ukončen výpočet všech procesů, které jsou definovány jako argumenty parametru -Depend. Argumentem jsou názvy procesů, na které má vytvářený proces čekat.

-Exclusive Parametr -Exclusive specifikuje, jestli má vytvářený proces přidělené výpočetní prostředky zabrat jen pro sebe.

-TaskFile Parametr -TaskFile umožňuje vytvořit výpočetní proces z XML souboru. Argumentem parametru -TaskFile je absolutní cesta k souboru a název souboru.

SetHPCTask

Skript Set HPC Task využívá cmdletu Set-HpcTask, který slouží k dodatečnému nastavování parametrů výpočetních procesů.

Skript je možné zavolat s již předdefinovaným ID úlohy, ID procesu, jménem procesu, pracovním adresářem, jménem spustitelného souboru výpočetního procesu, standardním výstupem a standardním chybovým výstupem ve formě parametrů, tedy SetHPCTask.ps1 idulohy idprocesu jmenoprocesu pracovniadresar jmenosouboruprocesu vystup chybovyvystup v tomto pořadí, případně s parametry -JobId, -TaskId, -Name, -WorkDir, -AppName, -Out a -Err s jejich argumenty v libovolném pořadí, tedy například SetHPCTask.ps1 -JobId idulohy -WorkDir pracovniadresar -AppName jmenosouboruprocesu -Name jmenoprocesu -Err chybovyvystup -Out vystup -TaskId idprocesu. Pokud je skript zavolaný s ID úlohy, ID procesu, jménem procesu, pracovním adresářem, jménem spustitelného souboru výpočetního procesu, standardním výstupem i standardním chybovým výstupem, nespustí se průvodce a výpočetní proces je upraven podle zadaných parametrů. Pokud je skript spuštěn bez parametrů nebo jen s prvními parametry ve správném pořadí, je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

-TaskId Parametr -TaskId slouží k definování ID výpočetního procesu, který chceme upravit.

-CommandLine Parametr -CommandLine slouží k definování názvu spustitelného souboru výpočetního procesu a případnému definování parametrů pro tento proces. Název spustitelného souboru je povinný a definuje se spolu s relativní cestou vzhledem k Working Directory.

-Name Parametr -Name slouží k nastavení jména výpočetního procesu pro snadnější orientaci.

-JobId Parametr -JobId definuje ID výpočetní úlohy, ke které má být výpočetní proces přidán.

-NumNodes Parametr -NumNodes specifikuje celkový počet výpočetních nodů, na kterých má proces běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumSockets, není možné nastavit -NumNodes.

-Stderr Parametr -Stderr specifikuje relativní cestu vzhledem k Working Directory a název souboru pro standard error výstup výpočetního procesu.

-Stdout Parametr -Stdout specifikuje relativní cestu vzhledem k Working Directory a název souboru pro standardní výstup výpočetního procesu.

-WorkDir Parametr -WorkDir specifikuje umístění, ve kterém bude výpočetní proces pracovat, a od tohoto umístění se udávají další relativní cesty. Working Directory tedy může mít tvar například \\storage-

inf\hpc$\beralu00\mojeuloha. Zadaná cesta by neměla obsahovat mezery ani diakritiku.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Stdin Parametr -Stdin specifikuje relativní cestu vzhledem k Working Directory a název souboru pro standardní vstup výpočetního procesu.

-NumCores Parametr -NumCores specifikuje celkový počet procesorových jader, na kterých má proces běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumNodes nebo -NumSockets, není možné nastavit -NumCores.

-NumSockets Parametr -NumSockets specifikuje celkový počet soketů, na kterých má proces běžet. Přípustný formát je [minimum]-[maximum]. Pokud je zadána jedna hodnota, je na danou hodnotu nastaveno minimum i maximum. Pokud je nastavený parametr -NumCores nebo -NumNodes, není možné nastavit -NumSockets.

-Depend Parametr -Depend umožňuje nastavit závislosti mezi procesy. Nově vytvářený proces nebude spuštěn dřív, než bude ukončen výpočet všech procesů, které jsou definovány jako argumenty parametru -Depend. Argumentem jsou názvy procesů, na které má vytvářený proces čekat.

-Exclusive Parametr -Exclusive specifikuje, jestli má vytvářený proces přidělené výpočetní prostředky zabrat jen pro sebe.

GetHPCTask

Skript Get HPC Task využívá cmdletu Get-HpcTask, který slouží k vypsání výpočetních procesů přiřazených k zadané výpočetní úloze nebo vypsání informací o procesu.

-JobId Parametr -JobId slouží k zadání ID výpočetní úlohy, z které chceme získat informace o výpočetních procesech.

-State Parametr -State určuje typ výpočetních procesů úlohy, které chceme získat. Možné hodnoty jsou Configuring, Submitted, Validating, Queued, Dispatching, Running, Finishing, Finished, Failed, Canceled, Canceling nebo All. Výchozí hodnota je All.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-TaskId Parametr -TaskId zadáním ID výpočetního procesu umožňuje vypsat informace o zadaném procesu.

StopHPCTask

Skript Stop HPC Task využívá cmdletu Stop-HpcTask, který slouží k ukončení výpočetního procesu úlohy.

Skript je možné zavolat s již předdefinovaným ID úlohy a ID výpočetního procesu ve formě parametrů, tedy StopHPCTask.ps1 idulohy idprocesu v tomto pořadí, případně s parametry -JobId a -TaskId s jejich argumenty v libovolném pořadí, tedy například StopHPCTask.ps1 -TaskId idprocesu -JobId idulohy. Pokud je skript zavolaný s ID úlohy i ID procesu, nespustí se průvodce a proces s definovaným ID v rámci definované úlohy je násilně ukončen. Pokud je skript spuštěn bez parametrů nebo s prvním parametrem, je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

-JobId Parametr -JobId specifikuje ID úlohy, jejíž výpočetní proces chceme ukončit.

-TaskId Parametr -TaskId specifikuje ID výpočetního procesu v rámci úlohy, který chceme ukončit.

-Force Parametr -Force specifikuje, že výpočetní proces bude ukončen násilně bez využití Grace Periody pro řádné ukončení.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Message Parametr -Message umožňuje nastavit zprávu, která má být zobrazena jako důvod ukončení procesu.

ExportHPCTask

Skript Export HPC Task využívá cmdletu Export-HpcTask, který slouží k exportování výpočetního procesu do XML souboru.

Skript je možné zavolat s již předdefinovaným ID úlohy, ID výpočetního procesu a cestou pro export ve formě parametrů, tedy ExportHPCTask.ps1 idulohy idprocesu cesta v tomto pořadí, případně s parametry -UserPath, -JobId a -TaskId s jejich argumenty v libovolném pořadí, tedy například ExportHPCJob.ps1 -UserPath cesta -JobId idulohy -TaskId idprocesu. Pokud je skript zavolaný s ID úlohy, ID výpočetního procesu i cestou pro export, nespustí se průvodce a proces s definovaným ID je exportován do definovaného souboru. Pokud je skript spuštěn bez parametrů nebo s prvními několika parametry (ve správném pořadí), je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

-JobId Parametr -JobId slouží pro zadání ID úloh, jejíž výpočetní proces chceme exportovat.

-TaskId Parametr -TaskId slouží k zadání ID výpočetního procesu v rámci úlohy, který chceme exportovat.

-Path Parametr -Path specifikuje jméno souboru a cestu, kam se má výpočetní proces exportovat. Cesta je zadávána vzhledem k hlavnímu nodu a je doporučené používat síťové umístění.

GetHPCMetric

Skript Get HPC Metric využívá cmdletu Get-HpcMetricValue, který slouží k zobrazení využití klastru.

-Name Parametr -Name slouží pro zadání jmen metrik, které chceme zobrazit.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Counter Parametr -Counter umožňuje zobrazit čítače v rámci metriky. Například metrika HPCSchedulerJobs obsahuje čítače jako Total number of jobs nebo Number of canceled jobs.

-MetricTarget Parametr -MetricTarget specifikuje lokace, ze kterých chceme generovat metriky. Přípustné hodnoty jsou HeadNode, ComputeNode nebo Cluster.

-Node Parametr -Node specifikuje jeden nebo více objektů typu HpcNode, ze kterých chceme generovat metriky.

-NodeName Parametr -NodeName specifikuje jména výpočetních nodů, ze kterých chceme generovat metriky.

-Type Parametr -Type umožňuje vybrat typ metrik, které chceme zobrazit. Přípustné hodnoty jsou Performance, Hardware nebo Calculated.

GetNodeInformation

Skript Get Node Information využívá cmdletu Get-HpcNode, který slouží k vypsání nodů a informací o nich.

Skript vypíše všechny nody a zobrazí jejich jméno, stav, dostupnost pro výpočty, počet procesorových jader a velikost operační paměti.

Další možné rozšíření skriptu může spočívat v následujících parametrech:

-Health Parametr -Health vypíše nody podle jejich stavu. Přípustné hodnoty jsou OK, Unreachable, OngoingOperation, DiagnosticFailed nebo ProvisioningFailed.

-Name Parametr -Name vypíše nody podle zadaných jmen.

-State Parametr -State vypíše nody podle jejich stavu dostupnosti pro výpočty. Přípustné hodnoty jsou Unknown, Provisioning, Offline, Starting, Online, Draining, Rejected nebo Removing.

NewSingle-TaskJob

Skript New Single-Task Job využívá cmdletů New-HpcJob, Add-HpcTask a Submit-HpcJob a slouží pro vytvoření jednoduché výpočetní úlohy.

Skript je možné zavolat s již předdefinovaným jménem, pracovním adresářem, jménem spustitelného souboru výpočetního procesu, standardním výstupem a standardním chybovým výstupem ve formě parametrů, tedy NewSingle-TaskJob.ps1 jmenoulohy pracovniadresar jmenosouboruprocesu vystup chybovy-vystup v tomto pořadí, případně s parametry -Name, -WorkDir, -AppName, -Out a -Err s jejich argumenty v libovolném pořadí, tedy například NewSingle-TaskJob.ps1 -WorkDir pracovniadresar -AppName jmenosouboruprocesu -Name jmenoulohy -Err chybovyvystup -Out vystup. Pokud je skript zavolaný se jménem, pracovním adresářem i jménem spustitelného souboru výpočetního procesu, nespustí se průvodce a výpočetní úloha s definovanými parametry je vytvořena a spuštěna. Parametry standardního výstupu a standardního chybového výstupu jsou nepovinné. Pokud je skript spuštěn bez parametrů nebo jen s prvními parametry ve správném pořadí, je spuštěn průvodce, který požaduje doplnění nevyplněných parametrů.

Popis parametrů u tohoto skriptu včetně možností rozšíření je tedy stejný jako u výše uvedených skriptů NewHPCJob, NewHPCTask a SubmitHPCJob.

DefineRestartHPCJob

Skript Define Restart HPC Job slouží k monitorování úlohy běžící na klastru. Pokud definovaná úloha skončí ve stavu Failed, je automaticky znovu spuštěna s původním nastavením.

Ve skriptu je nutné v prvním kroku nastavit jen ID úlohy pro sledování. Skript následně vytvoří v plánovači událostí na serveru naplánovanou úlohu, která se stará o monitorování zvolené výpočetní úlohy v hodinových intervalech. Pokud se výpočetní úloha ocitne ve stavu Failed, dojde k automatickému spuštění výpočetní úlohy znovu s původním nastavením. Nová úloha je opět monitorována v plánovači. V případě úspěšného ukončení výpočtu je skript z plánovače událostí odstraněn.

Skript je možné zavolat s již předdefinovaným ID úlohy ve formě parametru, tedy DefineRestartHPCJob.ps1 idulohy. Pokud je skript zavolaný s ID úlohy, nespustí se průvodce a úloha s definovaným ID je rovnou zařazena pro sledování.

C. Referenční příručka – webový portál

All Jobs Stránka All Jobs je výchozí a na této stránce jsou zobrazeny všechny výpočetní úlohy všech uživatelů, které jsou uloženy v historii výpočetního klastru. Historie obsahuje úlohy za posledních 7 dní.

Pokud klikneme na jméno úlohy nebo ID úlohy, otevře se nám stránka s informacemi o vybrané úloze, kde můžeme vidět na záložce Job Progress stav celé úlohy a jednotlivých výpočetních procesů, na záložce Job Details informace o konfiguraci dané úlohy, na záložce View Tasks je možné vidět informace o výpočetních procesech a jejich výpočetních prostředcích, na záložce Resource Selection je možné vidět výpočetní zdroje přidělené vybrané úloze a záložka Allocated Nodes zobrazuje informace o alokovaných výpočetních nodech pro vybranou úlohu.

Každý uživatel může zobrazit pouze informace o úloze, jejíž je vlastníkem. Zobrazení bližších informací o cizích výpočetních úlohách není možné.

My Jobs Stránka My Jobs poskytuje stejné možnosti, jako stránka All Jobs, pouze s tím rozdílem, že zobrazuje jen úlohy aktuálně přihlášeného uživatele.

New Job Na stránce New Job můžeme vytvořit novou výpočetní úlohu. Vytvoření nové výpočetní úlohy je možné importováním z XML souboru, vytvořením nové jednoduché úlohy nebo vytvořením nové úlohy s pokročilým nastavením.

Basic Job Pokud chceme vytvořit novou výpočetní úlohu se základními možnostmi nastavení, vybereme volbu Basic Job. Tato volba umožňuje nastavit stejné parametry výpočetní úlohy jako vytvoření nové výpočetní úlohy se základním nastavením v HPC Job Manageru. Navíc nabízí možnost uložit si šablonu úlohy do oblíbených (Favorites) a později tuto úlohu snadno a rychle spustit znovu.

Full Job Pokud chceme vytvořit novou výpočetní úlohu s pokročilými možnostmi nastavení, vybereme volbu Full Job. Tato volba umožňuje stejné volby nastavení jako výpočetní úloha s pokročilým nastavením vytvořená přes HPC Job Manager. Jediné, co není možné zadat, je výpočetní proces typu Service Task.

New job from XML file Pokud chceme vytvořit novou výpočetní úlohu z XML souboru, vybereme možnost New job from XML file a načteme XML soubor s informacemi o výpočetní úloze. Vytvoří se nám nová výpočetní úloha s předdefinovanými parametry z XML souboru. Další úpravy je možné provádět jako u vytváření nové výpočetní úlohy s pokročilým nastavením.

Favorites Stránka Favorites obsahuje jednoduché úlohy uložené jako oblíbené. Otevřením úlohy uložené do oblíbených je možné načíst uložená nastavení a rychle tím spustit novou výpočetní úlohu podle dříve uložené úlohy.

Heat Map Na stránce Heat Map je možné zobrazit vytížení jednotlivých výpočetních nodů vlastními výpočetními úlohami. Změnu výchozího zobrazení je možné provést pomocí Configure heat map, kde je možné zvolit zobrazení Cores in use, CPU usage, Disk throughput, Memory paging nebo Network usage a u každého změnit měřítko zobrazení. U vytížení je na učebně 5.003 možné zobrazit pouze využitá procesorová jádra, jelikož HPC Pack 2008 R2 for workstations neumožňuje zobrazit žádné další informace. Pokud by se však do výpočetního klastru přidaly servery nastavené jako Compute Nodes, bylo by možné u nich zobrazit všechny parametry.

Help Poslední položkou v menu je nápověda, která odkazuje na stránky Technet knihovny.

D. Obsah přiloženého DVD

`bin/`
: Adresář obsahuje obraz instalačního disku Microsoft HPC Pack 2008 R2 Enterprise (x64).

`doc/`
: Adresář obsahuje text práce v PDF formátu v souboru vypocetni-klastr-beran.pdf a kompletní zdrojový text práce v souboru beran-vypocetni-klastr.zip.

`src/`
: Adresář obsahuje podadresář Availability Scripts s PowerShell skripty pro automatické nastavení dostupnosti klastru, podadresář Cluster Configuration s XML soubory šablon výpočetních nodů a výpočetních úloh a textovým souborem s informacemi o konfiguraci sítě klastru, podadresář PowerShell Scripts s 15 předpřipravenými skripty v PowerShellu pro ovládání klastru, podadresář PiCalculator s vzorovou MPI aplikací na výpočet čísla π včetně zdrojových kódů, podadresář SOM s vzorovou MPI aplikací na tvorbu a naučení neuronové sítě typu SOM a podadresář WakeOnLan s aplikací NetScan pro probouzení výpočetních nodů.

`readme.txt`
: Soubor obsahuje stručný popis obsahu DVD.