Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra aplikované fyziky a techniky

Diplomová práce

Realizace výpočetních úloh na MetaCentru

Vypracoval: Bc. Josef Horelica
Vedoucí práce: doc. RNDr. Petr Bartoš, Ph.D.

České Budějovice 2013

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce.

Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 29. dubna 2013

.....
Bc. Josef Horelica

Rád bych poděkoval vedoucímu diplomové práce doc. RNDr. Petru Bartošovi, Ph.D. za příkladné vedení, cenné rady a trpělivost. Dále děkuji všem pracovníkům MetaCentra za konzultace.

Anotace

Diplomová práce se zabývá problematikou realizace náročných výpočetních úloh na MetaCentru. Pro tyto potřeby byl vytvořen manuál, který by měl počátky práce na MetaCentru usnadnit. V první části práce jsou shrnuty základní poznatky o paralelním počítání, charakteristika prostředků pro počítání, projekt MetaCentrum a některé aplikace nabízené MetaCentrem. V druhé části práce jsou uvedeny praktické ukázky práce na MetaCentru. Praktickou část doplňuje přiložené CD obsahující multimediální výukový kurz.

Klíčová slova

Paralelní počítání, MATLAB, MetaCentrum, plánovací systém.

Abstract

This thesis deals with the realization of demanding computing tasks on MetaCentrum. For these purposes was created the manual, which should help beginners on MetaCentrum. The first part is about the basic knowledge about parallel computation, characterization of means for computing, project MetaCentrum and some applications offered by MetaCentrum. In the second part there are the practical examples of computing on MetaCentrum. The enclosed CD contains the multimedia tutorial.

Keywords

Parallel computing, MATLAB, MetaCentrum, planning system.

Obsah

Úvod

Tato diplomová práce je koncipována jako výukový materiál, který čtenáře seznámí s problematikou realizace náročných výpočetních úloh na MetaCentru. Projekt MetaCentrum představuje důležitý článek pro podporu rozvoje VaV v ČR tím, že cílem je vybudovat národní gridovou infrastrukturu (NGI) a propojit výzkumné týmy a instituce v ČR. Dalším cílem je připojení ČR do mezinárodních výzkumných prostředí, jako jsou např. EGEE/EGI, EMI, EUAsiaGRID, CHAIN.

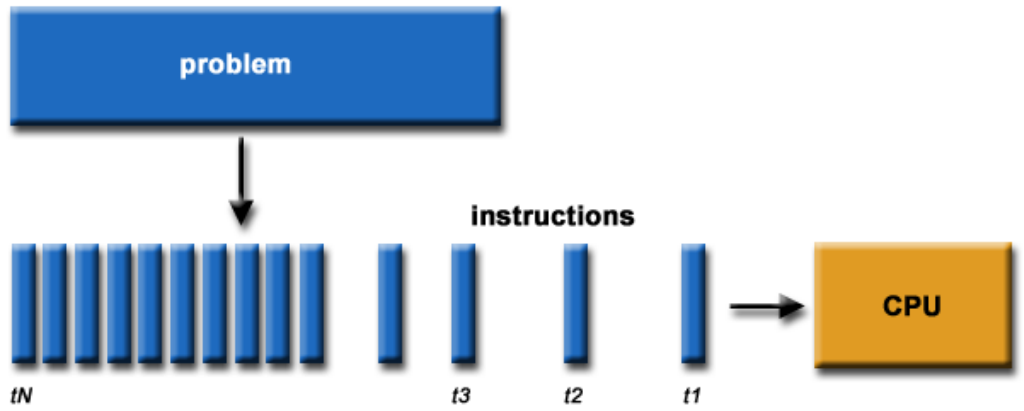
Práci je možné rozdělit do dvou částí. V první části jsou shrnuty základní poznatky o paralelním počítání, zejména v prostředí MATLAB, charakteristika prostředků pro počítání, projekt MetaCentrum a některé aplikace nabízené MetaCentrem.

V druhé části práce jsou uvedeny praktické ukázky počítání na MetaCentru pomocí aplikace MATLAB. Praktickou část ještě doplňuje přiložené CD, které obsahuje multimediální výukový kurz, představující spuštění úlohy až po získání dat z MetaCentra.

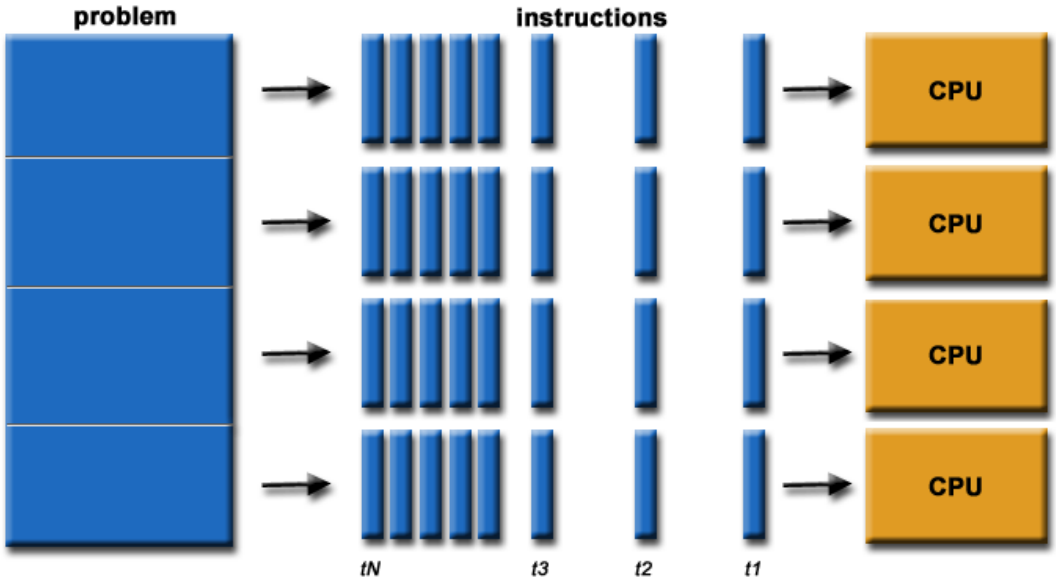
Při psaní práce byl kladen důraz na názornost, a proto jsou kapitoly v praktické části doplněny screenshoty, které ilustrují jednotlivé kroky práce na MetaCentru. Úlohy je možné vyzkoušet, protože zdrojové soubory jsou na přiloženém CD.

1. Úvod do paralelních výpočtů

Pojem paralelizace lze přiblížit typickým školním příkladem s kombajnem, který sklidí úrodu za 10 hodin a otázkou by bylo, za jak dlouho sklidí úrodu, pokud by byly dva kombajny. Analogicky tedy paralelní počítání znamená využití více než jednoho procesoru k řešení úloh, viz obrázky 1 a 2.

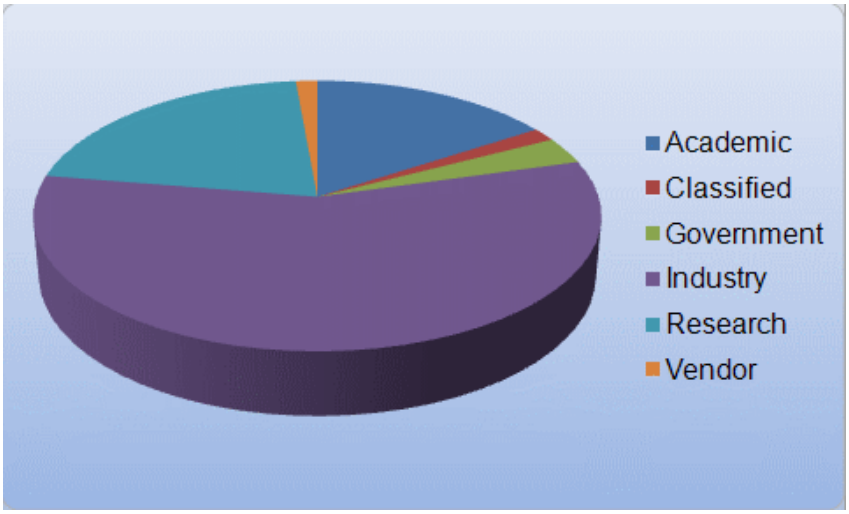


Obr. 1 Jednoprocesorové počítání, převzato a upraveno z [1]

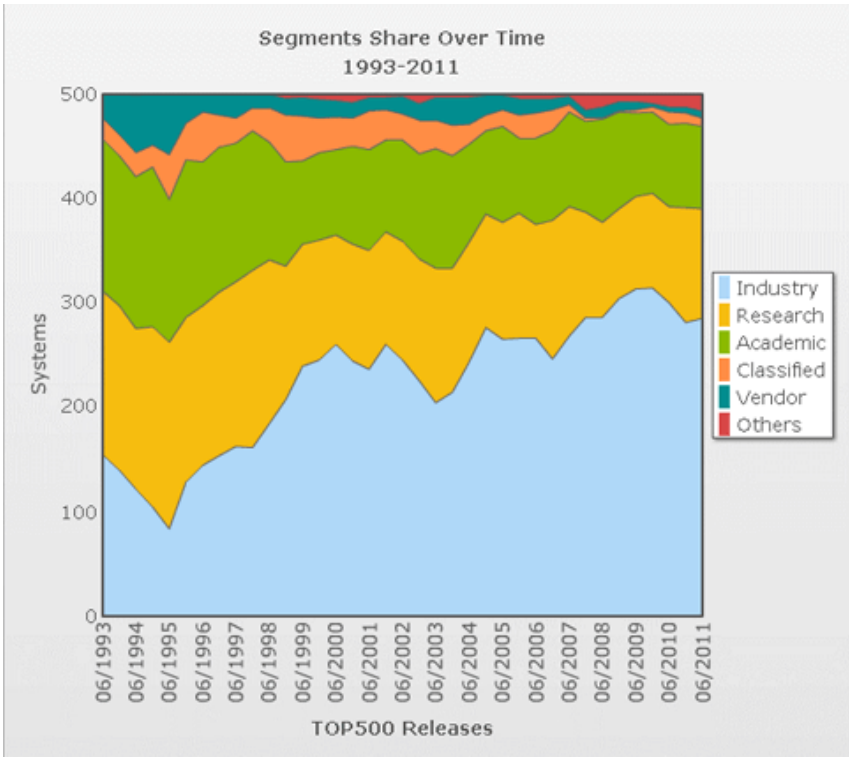


Obr. 2 Víceprocesorové počítání, převzato a upraveno z [1]

Z výše uvedených obrázků je zřejmé, že paralelizací, tj. činností několika procesů současně, lze získat větší výpočetní výkon, resp. zpracovat daleko více informací než při jednoprocesorovém výpočtu. To je velmi důležité při výpočtech složitých úloh, jako jsou např. simulace v reálném čase, inženýrské výpočty, vědecké výpočty apod. Pro přehled jsou uvedeny grafy 1 a 2, ve kterých jsou vyobrazeny oblasti využívající paralelní výpočty a přehled využívání paralelismu v čase jednotlivých oblastí.



Graf 1 Oblasti využívající paralelní počítání, převzato a upraveno z [1]



Graf 2 Využívání paralelního počítání v čase a jednotlivých oblastech, převzato a upraveno z [1]

Je několik dalších důvodů, proč paralelizaci využít. Za prvé, v budoucnu se bude muset jistě řešit také to, že jsme omezeni materiálovým limitem, resp. vývoj procesorů nelze donekonečna zvyšovat. Za druhé, v jednotlivých oblastech, které jsou zmíněny výše, vznikají požadavky na řešení daleko větších a složitějších problémů, které by se nedaly řešit bez využití paralelizace. Za třetí, díky paralelizaci je možné ušetřit jak čas, tak peníze.

V oblasti paralelních výpočtů se musí samozřejmě počítat i s některými problémy. Co se týče hardwaru, tak v současnosti lze konstruovat paralelní počítače mající společnou paměť pro maximálně 100 procesorů. Do tzv. clusterů jsou pak spojeny jednoprocesorové nebo víceprocesorové počítače, které mezi sebou komunikují pomocí switche.

Právě na tyto switche se zaměřuje vývoj, neboť umožňují téměř stejně rychlou komunikaci mezi procesory jako mezi procesorem a vnitřní pamětí počítače. Software představuje také potencionálně problém, protože se nelze spoléhat na automatickou paralelizaci (paralelní kompilátory - HPF, High Performance Fortran, či paralelní numerické knihovny). Obecně platí to, že nejlepší paralelizací je ta, kterou si uživatel udělá sám. V neposlední řadě se ukazuje, že některé algoritmy používané na jednoprocesorových počítačích nelze dost dobře paralelizovat, a to vede k vytváření nových paralelních algoritmů [2].

1.1 Více procesorů neznamená rychlejší výpočet

Touto otázkou se zabýval Gene Amdahl, jenž formuloval zákon, popisující závislost zrychlení systému poté, co je vylepšena pouze některá z jeho částí [3]. Jak se zrychlí výpočet, pokud se místo jednoho procesoru použije *n* procesorů?

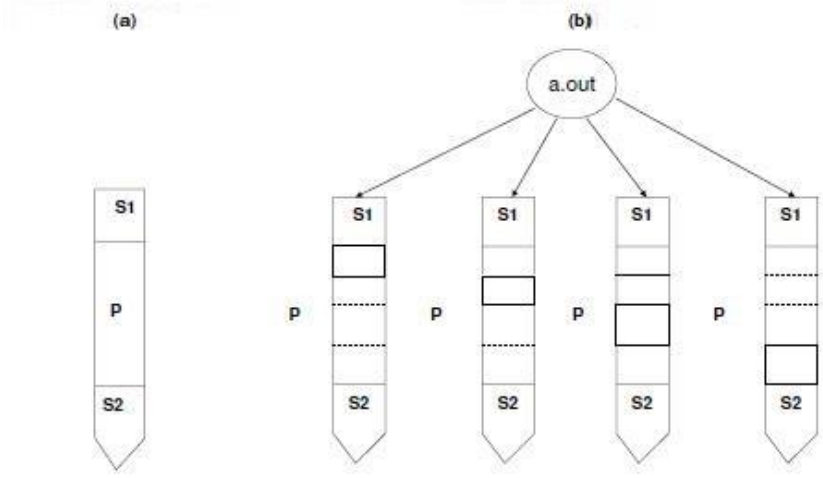
Podle Amdahlova zákona lze spočítat, kolik procent výpočetního času bude trvat paralelizovaný program oproti programu, který bude spuštěn na jednoprocesorovém počítači. Jako příklad lze uvést, kdy se sečítá 1 000 000 000 čísel. Načtení prvků probíhá 8 % celkového výpočtového času, vlastní výpočet součtu trvá 90 % celkového výpočtového času a tisk trvá 2 % celkového výpočtového času.

Podle Amdahlova vzorce

$$(100 - P) + \frac{P}{n} \qquad (1)$$

(100 – P) neparalelizovatelná část, *P* – paralelizovatelná část, *n* – počet procesorů. Při použití *n* = 10 procesorů lze vypočítat zrychlení výpočtu, které je přibližně 19 %, t.j. zrychlení výpočtu paralelizací přibližně 5krát. Z Amdahlova zákona je možné vyvodit dva závěry:

1) neplatí přímá úměra mezi počtem procesorů a urychlením výpočtu (použití 10 procesorů neznamená urychlení výpočtu 10krát),
2) pro limitní případ kdy $n \Rightarrow \infty$ je urychlení výpočtu konečné a rovno *100/(100 - P)*. Pro ilustraci je uveden obrázek 3 [3].



Obr. 3 Schéma (a) sériového a (b) paralelního programu (běžícího na 4 procesorech). S1 a S2 jsou části programu, které nelze paralelizovat a *P* je paralelizovatelná část programu, převzato a upraveno z [3]

2. MATLAB

MATLAB (MATrix LABoratory) je interaktivní programové prostředí a skriptovací programovací jazyk čtvrté generace. Program MATLAB je vyvíjen společností MathWorks a v březnu 2012 vyšla zatím poslední verze R2012a, která je k dispozici pro operační systémy Linux (32bitový, 64bitový), Windows (32bitový, 64bitový), Mac OS X (64bitový). MATLAB umožňuje počítání s maticemi, vykreslování 2D i 3D grafů funkcí, implementaci algoritmů, analýzu, paralelní výpočty, modelování a simulaci různých dějů, prezentaci dat a i vytváření aplikací včetně uživatelského rozhraní [5].

MATLAB tvoří soubory s koncovkou **.m*, tzv. *m-files*, obsahující definice funkcí, skriptů nebo tříd. Práce s těmito připravenými *m-files*, určenými k výpočtu na MetaCentru, bude řešena v kapitole 9. Je nutné ještě zmínit, že MATLAB obsahuje tzv. Toolboxy (knihovny), které budou zmíněny v kapitole 2.3 [5].

2.1 Základní vlastnosti MATLABu

- Interpretační jazyk - uživatel obdrží odpověď na svůj povel téměř okamžitě.
- Základními objekty - matice, ale podporuje i složitější typy, například vícerozměrná pole, datové struktury atd.
- Skládáním datových typů je možné vytvořit libovolně složité datové struktury.
- Grafika - umožňuje snadné zobrazení a prezentaci získaných výsledků. Lze vykreslit různé druhy grafů: dvourozměrné, třírozměrné, histogramy apod. MATLAB také umožňuje otevřít více oken pro zobrazení grafů najednou nebo zobrazit více grafů v jednom okně.
- Otevřená architektura přispěla k velkému rozšíření.
- Úplný programovací jazyk - uživatelé v něm mohou vytvářet funkce "šité na míru" pro jejich aplikace. Tyto funkce se způsobem volání nijak neliší od vestavěných funkcí a jsou uloženy v souborech v čitelné formě. Dokonce většina funkcí s MATLABem dodávaných je takto vytvořena a opravdu vestavěné jsou jen funkce základní. To má dvě velké výhody - jazyk MATLABu je téměř neomezeně rozšiřitelný a kromě toho se uživatel může při psaní vlastních funkcí poučit z algoritmů s programem dodávaných. Navíc jsou takto koncipované funkce snadno přenosné mezi různými platformami, na kterých je MATLAB implementován [4].

2.2 Stručný úvod do MATLABu

V MATLABu lze pracovat buď v interaktivním režimu, nebo v dávkovém režimu. Interaktivní režim pracuje v okně Command Window, kde se zadávají jednotlivé příkazy, které se potvrzují klávesou Enter. Dávkový režim představuje zapsání jednotlivých příkazů např. do připraveného souboru, který se najednou spustí.

Datové Typy[6]

MATLAB umožňuje pracovat s mnoha datovými typy - číselné, řetězce, pole (jednorozměrné - řádkové a sloupcové vektory a dvourozměrné - matice). Číselné hodnoty mají implicitně dvojitou přesnost a mohou být reálné nebo komplexní. Matice nemusí být pouze plné, ale lze využít také řídké reprezentace. V desetinných číslech se píše desetinná tečka. Podrobný popis lze najít v elektronické dokumentaci či použít příkaz ***help datatypes***.

Speciální proměnné[6]

ans	výsledek posledního (nikam nepřiřazeného) výrazu (funkce);
eps	počítačové epsilon;
pi	hodnota Ludolfovo číslo;
realmin	nejmenší použitelné reálné číslo;
realmax	největší použitelné reálné číslo;
i, j	imaginární jednotka ($\sqrt{-1}$);
inf	nekonečno;
NaN	Not a Number.

Další informace pomocí příkazu ***help elmat***.

Vytváření matic a vektoru[6]

Při vytváření matic a vektoru se pracuje s přiřazovacím příkazem " ***=*** ". Vlastní prvky se zapisují mezi dvě hranaté závorky a oddělují mezerami nebo čárkami (v případě prvku v řádce) a dále řádky matic mezi sebou oddělují středníky (nebo se píší každý řádek matice na zvláštní řádek). Příkaz ***a = [1 2 3 4 5 6 7 8 9 10]*** vytvoří řádkový vektor dimenze 10, zatímco příkaz ***B = [1 2 3 ; 4 5 6 ;7 8 9]*** vytvoří čtvercovou matici řádu 3 x 3.

Generování vektoru[6]

Použitím schématu [od:krok:do] lze generovat vektor s prvky od čísla od s krokem krok do čísla do. Například příkaz ***[1:0.2:2]*** vygeneruje vektor ***[1 1.2 1.4 1.6 1.8 2]***. Vynecháním kroku příkazem ***[od:do]*** vygenerujeme posloupnost čísel od čísla od do čísla do s krokem 1. Vektory lze tvořit prostým výčtem prvků, např. u = [1 2 3 4 5 6 7 8 9].

Operace s maticemi a vektory[6]

S maticemi a vektory lze provádět všechny operace známé z matematiky. Označení operátoru odpovídá konvencím (*+, -, *, /, ^*). Operátory s tečkou (například " *.** ") označují prvkové operace. V případě " *.** " by se tedy neprováděl maticový nebo vektorový součin, ale součin korespondujících prvků. Následuje seznam některých maticových funkcí. Další informace lze získat příkazy ***help ops*** a ***help elmat***.

Funkce a skripty[6]

Skript nemá hlavičku, nemá vstupní ani výstupní argumenty a přímo pracuje s daty pracovního prostoru příkazového okna (workspace). Skript je textový soubor s příponou "*m*" (m-soubor), který obsahuje posloupnost příkazů. Vytvořit skript je možné příkazem ***edit <jmeno_skriptu>***. Skript je možné spustit příkazem ***run <jmeno_skriptu>***. Všechny proměnné zůstávají po provedení skriptu v globální paměti. Skript může obsahovat komentáře uvozené znakem procenta "***%***". Jestliže několik prvních řádek skriptu jsou komentáře, lze tyto řádky vypsat příkazem ***help <jmeno_skriptu>***. To platí také pro funkce.

Funkce je skupina příkazů včetně hlavičky s formálními vstupními a výstupními parametry. Volá se jménem se skutečnými parametry, které se předávají do formálních podle pořadí. Zápis je ***function [vstupní proměnné] = jmeno_funkce(výstupní proměnné)***. Vstupní a výstupní parametry zajišťují přenos dat mezi pracovním prostorem a funkcí. Rozlišujeme funkce globální, které jsou zapsány v m-souboru se shodným jménem jako jméno funkce, a funkce lokální, které mohou být zapsány za tělem nadřazené globální funkce. Funkci lze v jejím těle ukončit příkazem ***return***. Další informace lze získat příkazem ***help function*** nebo příkazem ***help lang***.

Řídící struktury[6]

Stejně jako v každém programovacím jazyku, také v MATLABu existují řídící struktury typu různých cyklů a podmíněných příkazů. Více informací v nápovědě ***help lang***.

- podmíněný příkaz ***if***

```
if <logicky_vyraz>
  <prikazy>
elseif <logicky_vyraz>
  <prikazy>
else
  <prikazy>
end
```

  V logických výrazech je možné používat klasické relační a logické operátory (==, =, >, >, &, |, . . .). Další informace pomocí příkazu ***help ops***.

- podmíněný příkaz ***switch***

```
switch <vyraz>
 case <hodnota1>
  <prikazy>
 case <hodnota2>
  <prikazy>
...
 otherwise
  <prikazy>
end
```

  Umožňuje podle hodnoty výrazu vykonat různé příkazy.

- cyklus ***for***

```
for <promenna> = <vektor>
  <prikazy>
end
```

  Provádí určité příkazy po daný počet opakování, kde ***<vektor>*** je obyčejně nějaký vygenerovaný vektor typu [od:krok:do].

- cyklus ***while***

```
while <logicky_vyraz>
  <prikazy>
end
```

  Provádí určité příkazy, pokud platí zadaná podmínka.

Vstupně výstupní operace[4,8]

MATLAB poskytuje širokou škálu nástrojů pro správu dat. Informace lze ukládat (příkaz ***save <jmeno_souboru>***) nebo načítat (***load <jmeno_souboru>***). Tyto zmíněné příkazy lze použít pro import a export dat z textových souborů. Soubory s příponou "*.mat*" mohou být uloženy jako textové, implicitně se zapisují jako binární, to se ale nedoporučuje.

cd	změna pracovního adresáře;
copyfile	kopírování souboru nebo adresáře;
delete	odstranění souborů nebo grafických objektů;
dir	výpis obsahu adresáře;
fileattrib	nastavení atributu pro soubor nebo adresáře;
filebrowser	otevření panelu pro prohlížení běžného adresáře;
isdir	určuje, zda se jedná o adresář;
lookfor	hledá klíčové slovo ve všech položkách nápovědy;
ls	výpis obsahu adresáře;
MATLABroot	kořenový adresář;
mkdir	vytvoření nového adresáře;
movefile	přesun souboru nebo adresáře;
pwd	jméno běžného adresáře;
recycle	nastavuje volbu pro přesunutí zrušených souborů do koše;
rmdir	odstranění adresáře;
tempdir	jméno systémového dočasného adresáře;
Toolboxdir	kořenový adresář pro zadaný Toolbox;
type	zobrazí obsah souboru;
visdiff	porovná dva soubory (např. mat, binarní, zip) nebo adresáře;
fopen	otevření souboru;
fclose	uzavření souboru.

Grafy funkcí a plochy[7]

Pro vytváření grafických objektů (oken) se použije příkaz ***figure***, ve kterých MATLAB zobrazuje grafické výstupy a vrací ukazatel (handle) na vytvořený objekt. Dále příkaz ***plot*** vytváří grafy funkcí. Více informací lze získat v nápovědě MATLABu nebo na internetových stránkách MathWorks.

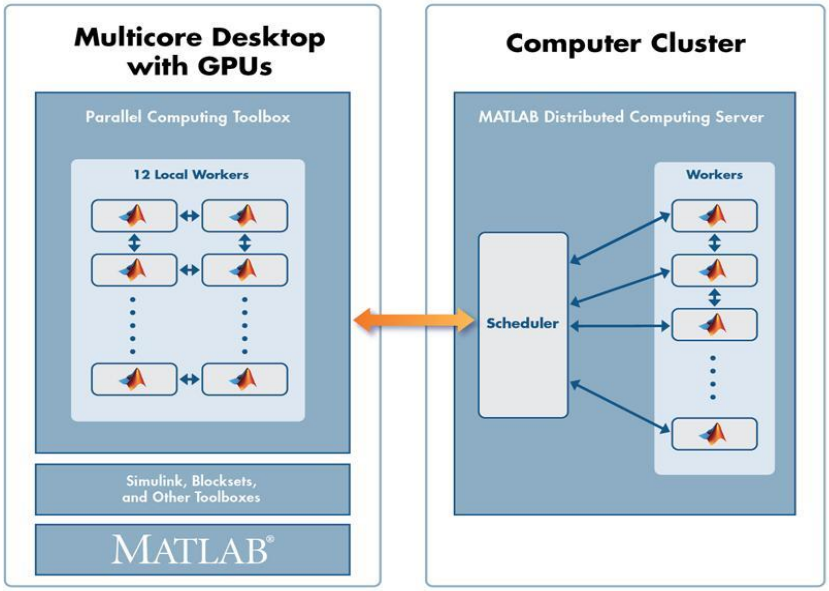
2.3 Paralelizace v MATLABu

V kapitole o paralelním počítání bylo zmíněno několik důvodů, proč paralelizaci používat. Prostředí MATLAB paralelní počítání umožňuje.

MATLAB obsahuje tzv. Toolboxy - knihovny funkcí, které rozšiřují možnosti použití MATLABu při paralelním počítání na více procesorech, a to buď na počítači samotném, nebo v kombinaci s výpočetními clustery. Paralelní počítání pomocí MATLABu je výhodné v tom, že uživatel nemusí nijak upravovat svůj kód, pokud chce počítat na svém PC nebo na clusteru. Užitím Parallel Computing Toolbox lze rozložit zátěž, při zpracování početně a datově náročných problémů, mezi více procesorů. Příkladem mohou být rozsáhlé simulace, zpracování dat, modelování, algoritmizace a testování, které mohou zabrat velmi mnoho času a mají větší nároky na hardware, resp. operační paměť. Tímto se umožní zredukovat úsilí věnované programování.

MATLAB Toolboxy rozšiřují MATLAB o konstrukce, jako jsou paralelní smyčky (parfor), speciální pole pro distribuované počítání (distributed jobs), jednoduché paralelní algoritmy (parallel jobs), metoda Single Program Multiple Data (SPMD), dále pak možnost využití GPU výpočtů na grafických kartách NVIDIA podporujících architekturu CUDA.

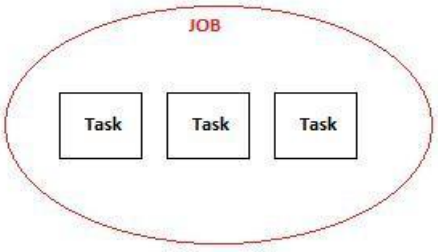
MATLAB od verze 2011 umožňuje provádět paralelní výpočty do počtu 12 procesorů pomocí Parallel Computing Toolboxu (licence v MetaCentru má název Distrib_Computing_Toolbox) na víceprocesorových strojích. Parallel Computing Toolbox společně s MATLAB Distributed Computing Server umožňuje spouštět aplikace na clusteru, viz obrázek 4, a pomocí MATLAB Distributed Computing Server lze provádět výpočty více jak s 12 procesory, jméno licence v MetaCentru je MATLAB_Distrib_Comp_Engine [9, 10, 12].



Obr. 4 Toolboxy MATLABu R2010b, převzato a upraveno z [11]

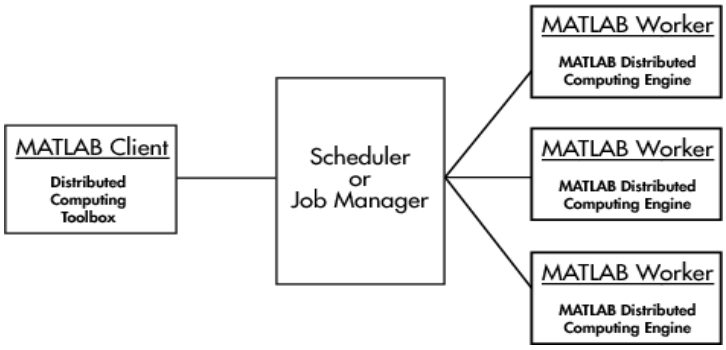
V úvodu je nutné uvést některé pojmy:

- *job* - objekt dat (tasků), které odesílá klient ke zpracování, viz obrázek 5;
- *task* - úloha (někdy označován jako Lab), kterou zpracovává worker;
- *worker* - MATLABovský stroj (engine), který zpracovává tasky;
- *job manager/scheduler* - plánuje a řídí vykonávání jednotlivých tasků, které jsou odesílány workerům. Je součástí Distributed Computing Engine;
- *results* - výsledky zpracovaných tasků, které vrací workery.

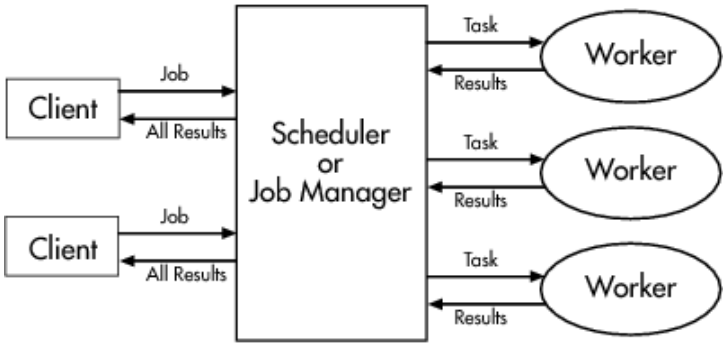


Obr. 5 Schéma Jobu, převzato a upraveno z [10]

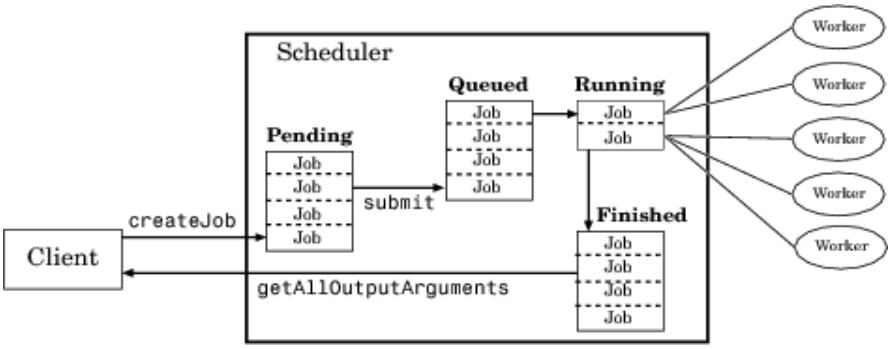
Klient vytváří objekty (joby) na svém PC, které mohou být zpracovávány na jeho počítači, nebo pomocí Distributed Computing Toolboxu se tyto joby rozdělí (dekomponují pomocí Job Manageru) na menší tasky. Job Manager je součástí MATLAB Distribubuted Computing Engine a řídí tasky, které rozesílá na jednotlivé workery (procesory). MATLAB Distribubuted Computing Engine zpracuje tasky a vrátí výsledky klientovi, viz obrázky 6, 7 a 8 [10].



Obr. 6 Schéma paralelizace v MATLABu, převzato a upraveno z [10]



Obr. 7 Komunikace mezi klientem, Job Managerem a workerem - detail, převzato a upraveno z [10]



Obr. 8 Detail stavu jobů v Job Manageru (plánovači), převzato a upraveno z [10]

2.4 Konstrukce pro paralelní počítání v MATLABu

***Parfor*[9]**

Paralelní smyčky *parfor* umožňují rozdistribuovat nezávislé tasky mezi více workerů. Podmínkou použití *parfor* je to, že žádná iterace není závislá na jiné a při zpracování smyčky mezi sebou workery nekomunikují. Paralelní smyčky zajišťují přesun dat mezi řídící session a workery a navíc automaticky detekují volné workery. V případě nedostatku workerů se prostředí vrací do normálního sériového zpracování. *Parfor* smyčka je užitečná v situacích, kde potřebuje uživatel mnoho iteračních smyček jednoduchých výpočtů, např. simulace Monte Carlo, BER testování (BER testing - Bit error rate).

Porovnání *for* a *parfor* smyček

- Klasický *for*

```
for i = 1:n
  % vypocet nejakeho kodu...
end
```

- Paralelní *parfor*

```
% nedefinovany pocet workeru
parfor (i = 1:n)
  % vypocet nejakeho kodu...
end
```

V příkladu *parfor* smyčka rozděluje tasky mezi více workerů, které nejsou pevně stanoveny, tzn., že distribuce tasků se provádí dynamicky. Místo toho, aby byl alokován fixní rozsah procesorů, je na začátku každé iterace přiřazen volný worker pro zpracování. To zajišťuje rovnoměrné rozložení zátěže. Pokud uživatel chce definovat počet workerů, MATLAB toto umožňuje použitím funkce *matlabpool.* Jejich počet je maximálně roven počtu jader, resp. virtuálních jader procesoru, anebo jich může být menší počet.

Hlavní vlastností funkce *matlabpool* je, že umožňuje interakci příkazového okna se zpracovatelskými procesy. To má za následek, že lze spojit sériový a paralelní kód bez toho, aniž by se musela spouštět dávka na clusteru. V níže uvedeném příkladu je task rozdělen mezi 4 workery a na konci výpočtu jsou výsledky sezbírány.

- Paralelní *parfor* s *matlabpool*

```
% pocet lokalnich workeru = 4
matlabpool open local 4
  parfor i = 1:1024
      A(i) = sin(i*2*pi/1024);
  end
matlabpool close
```

Distributed jobs [9, 13]

Distributed jobs (distribuované počítání) se skládají z tasků, které spolu nekomunikují a ani nemusí běžet současně. Například jeden worker může zpracovávat několik tasků za sebou. Výhodou je, že klient nemusí být po dobu výpočtu, kdy běží tasky na clusteru, online. Navíc pro specifikování zdrojů nutných pro výpočet se nemusí definovat v konfiguračním skriptu. Uživatel jej může napsat přímo do m-souboru, viz kapitola 9.3. Níže je uvedena základní konstrukce.

```
% Nalezeni JobManageru
sched = findResource('scheduler', 'type', 'torque');
% Definice typu clusteru, tyto hodnoty jsou pouzity pro kazdy job
set(sched, 'ClusterSize', 250);
set(sched, 'ClusterOsType', 'unix');
% MATLABovsky adresar s verzi MATLABu na clusteru
set(sched, 'ClusterMatlabRoot', '/server/aplikace/matlab/2010b');
% Definovani zdroju pro pocitani
% Hodnotu ^N^ lze nahradit cislem, ktere definuje pocet pozadovanych workeru
set(sched, 'ResourceTemplate', '-l select=^N^:ncpus=1:mem=2gb');
set(sched, 'SubmitArguments', '-l walltime=2:00:00');
% Definovani adresare, kde MATLAB muze ukladat souvisejici joby, podminkou je to, ze tyto soubory musi byt
% na sdilenem filesystemu, napr. AFS, nesmi byt v home - $HOME
 set(sched, 'DataLocation', '/server/nfs4/useradresar/data');
% Pozadovany pocet workeru, minum a maximum
set(job, 'MinimumNumberOfWorkers', cislo);
set(job, 'MaximumNumberOfWorkers', cislo);
% Vytvoreni job objektu a sdeleni JobManageru, kde ma najit kody, vstupni nebo vystupni soubory
job = createJob(sched, 'FileDependencies', {'/server/user_adresar/projekty/soubor.m'});
% Vytvoreni tasku pro vytvoreny job objekt
% J - objekt jobu, F - jmeno funkce, N - pocet (cislo) vystupu, {x1,..., xn} - vstupni parametry
createTask(j, F, N, {x1,..., xn});
% Potvrzeni jobu
submit(job);
% Cekej (smycka), dokud nejsou vsechny joby hotove
waitForState(job, 'finished');
% Ulozeni vysledku jobu
vysledek = getAllOutputArguments(job);
% Zobrazeni vysledku, lze modifikovat na jakem vystupu zobrazit
celldisp(vysledky);
% Smazani jobu
destroy(job);
% V pripade TRUE, jsou vystupy ze vsech uloh zaznamenavany a vraceny klientovi
tasks = get(job, 'Tasks');
set(tasks, 'CaptureCommandWindowOutput', true);
```

Parallel jobs [12]

V tomto případě jednotlivé workery spolu komunikují. Tasky jsou duplikovány na každém workeru a takto duplikované tasky jsou spuštěny současně. Každý worker může vykonávat tasky nad různými daty.

Příklad:

```
sched=findResource('scheduler', 'type', 'torque');
set(sched, 'SubmitArguments', '-l walltime=15:00 -q short ');
% Vytvoreni paralelniho jobu a ulozeni do promenne pjob s atributy sched
pjob=createParallelJob(sched);
set(pjob, 'MaximumNumberOfWorkers', 4);
set(pjob, 'MinimumNumberOfWorkers', 4);
set(pjob, 'FileDependencies', 'nejaky_soubor.m');
% Vytvoreni tasku a ulozeni do promenne t s atributy
t=createTask(pjob, @funkce, 1, {5,5})
submit(pjob);
waitForState(pjob);
results=getAllOutputArguments(pjob);
destroy(pjob);
```

Single program multiple data (SPMD) [14]

Tato metoda umožňuje spustit stejný program na všech workerech s tím, že na každém workeru budou k dispozici jedinečná data pro výpočet. Tato metoda je podobná paralelní smyčce. Na začátku je nutné použít příkaz na otevření *N* workerů pomocí příkazu ***matlabpool open N***. Těmto workerům jsou zasílána data pro výpočet.

Konstrukce je následující:

- *SPMD obecně:*

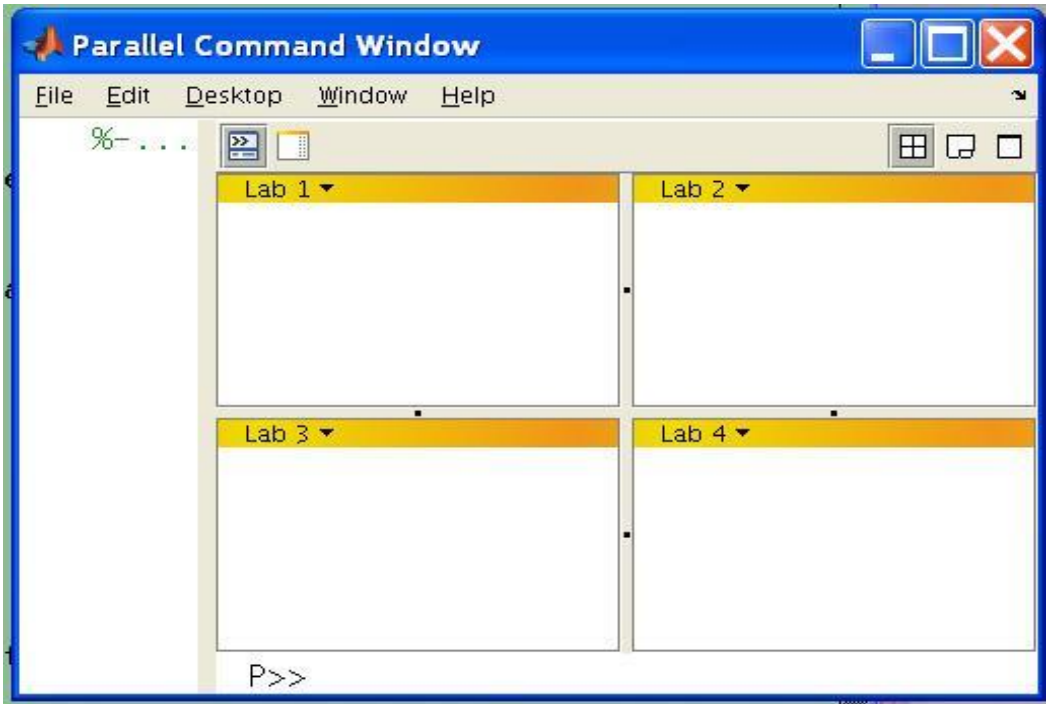
```
matlabpool open N
  spmd
    % prikazy pro jednotlive workery
  end

  spmd
    % dalsi prikazy pro jednotlivé workery
  end

matlabpool close
```

- *SPMD jednoduchý příklad:*

```
matlabpool open 4
  spmd
    A=rand(3,3);
  end
matlabpool close
```

Parallel Computing Toolbox umožňuje interaktivní paralelní režim (příkaz ***pmode***), ve kterém lze po rezervaci workerů, zobrazovat více oken, viz obrázek 9. Více o ***pmode*** na stránkách MathWorks. Níže je popsán příklad pro využití čtyř workerů, které představují čtyři okna na uživatelské stanici.

```
% Pozadani o misto vypoctu (local nebo cluster) a pocet workeru
pmode start <profile-name> <num-workers>
% Vykonani prikazu na vsech workerech
P>> X = 2*labindex;
% Nakopirovani promenne X z workeru 4 do MATLABu klienta
pmode lab2client X 4
% Nakopirovani promenne Y z MATLABu do workeru 1 az 4
pmode client2lab Y 1:4
% Ukonceni a navrat do MATLABu klienta
P>> pmode exit
```



Obr. 9 Vyvolání paralelního GUI, příkazem ***pmode***, převzato a upraveno z [14]

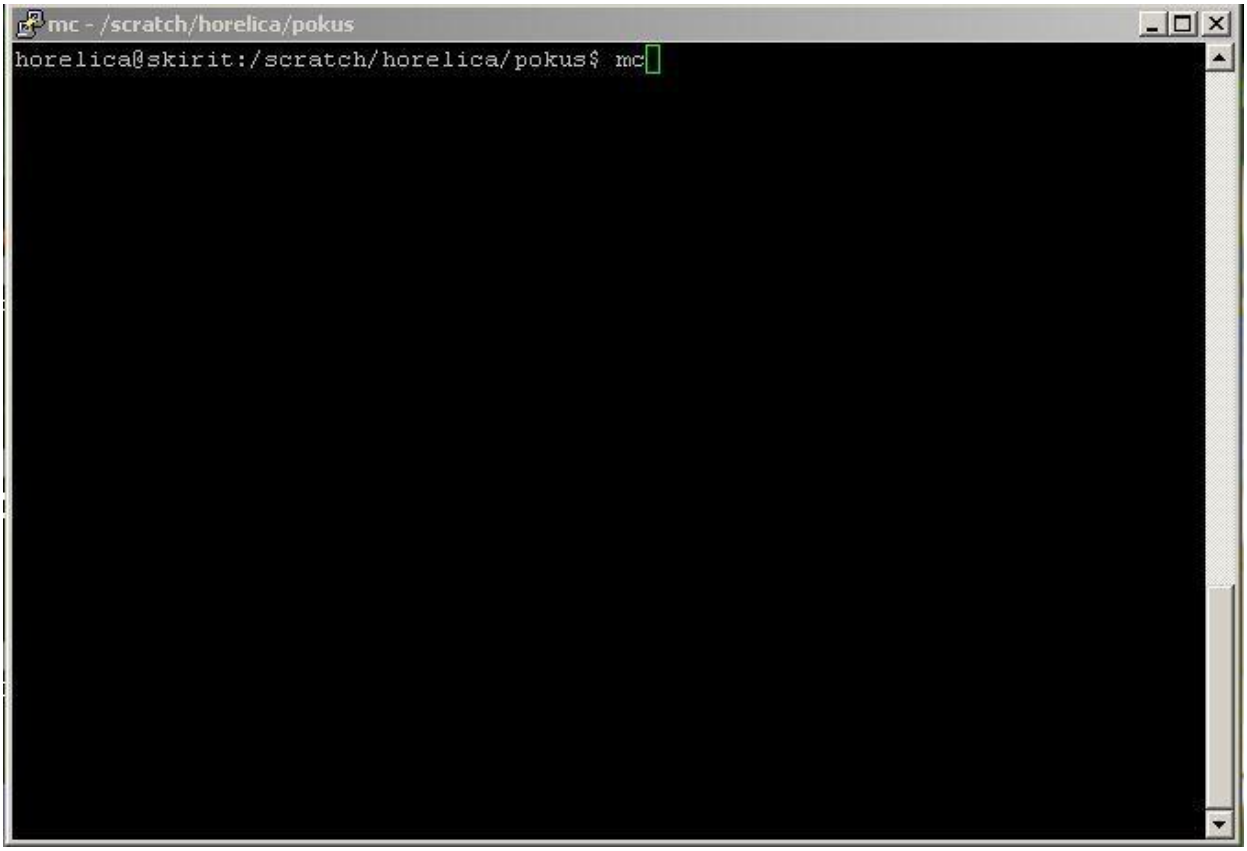
V MetaCentru mohou být problémy s přidělováním licencí Toolboxů MATLABu, protože licence se z licenčního serveru přidělí až v okamžiku použití Toolboxu, což může být i dlouho po zahájení výpočtu. Pokud se mezitím licence vyčerpá, může dojít k chybě. Řešením je umístit na začátek programu smyčku čekající na licenci, viz níže [12].

```
while 1
        [tf msg] = license('checkout','Statistics_Toolbox');
        if tf==1, break, end
        display(strcat(datestr(now),' waiting for licence '));
        pause(5);
end
```

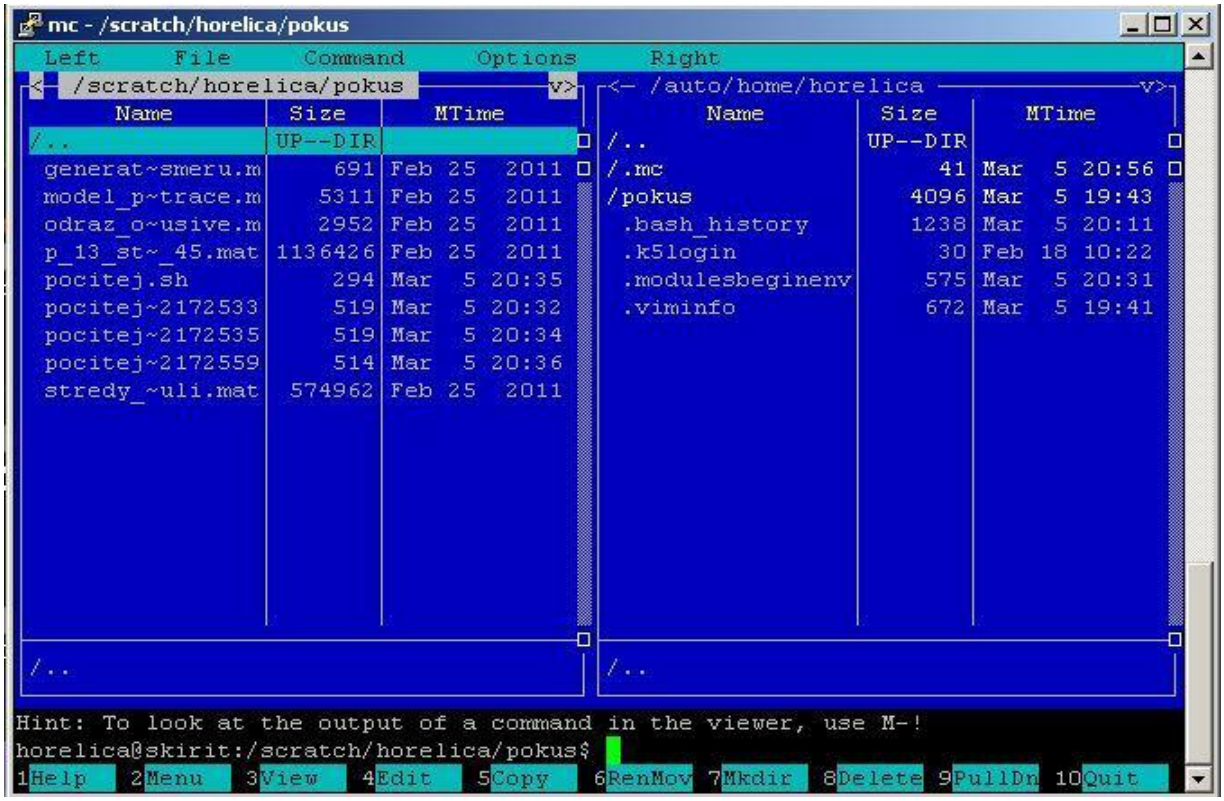
3. Příkazy systému UNIX

Na strojích MetaCentra běží UNIXové operační systémy a jejich ovládání je pomocí příkazového řádku. Uživatel operačního systému MS-Windows bude potřebovat nějaký program, který nahrazuje textový terminál, např. PuTTY. UNIXové operační systémy se liší od MS-Windows tím, že jejich příkazový řádek poskytuje mnohem lepší programovatelnost dávkových skriptů a že jsou mnohem lépe připraveny na používání v počítačové síti.

Pro práci se soubory se doporučuje, ze začátku, program Midnight Commander (obrázek 10), který lze spustit příkazem ***mc*** (potvrzení Enter). Je velice podobný programům Norton Commander, Total Commander a podobným. Obsahuje i jednoduchý textový editor, který se spustí klávesou F4. Program Midnight Commander se ukončuje klávesou F10, obrázek 11.



Obr. 10 Vyvolání programu Midnight Commander v terminálovém okně



Obr. 11 Program Midnight Commander v terminálovém okně

Úlohy se v MetaCentru zadávají ve formě shellových skriptů, které jsou ideově podobné souborům *.bat* z MS-Windows, ale používají mnohem silnější programovací jazyk. Manuál k příkazům lze získat po zadání příkazu ***man***. Například dokumentace k shellu bash se zobrazí příkazem ***man bash***, dokumentace k příkazu ***qsub*** zase příkazem ***man qsub*** atd.

3.1 Základní UNIXové nástroje pro práci na MetaCentru

Příkazy v prostředí UNIX se potvrzují klávesou Enter.

PWD
Příkaz ***pwd*** zobrazí cestu a název právě aktuálního adresáře. V domovském adresáři uživatele *horelica* příkaz ***pwd*** zobrazí:

/home/horelica

CD
Slouží pro změnu adresáře. Lze použít absolutní i relativní cestu, takže do podadresáře *pokus* aktuálního adresáře *horelica* lze přejít z tohoto adresáře jednou z těchto možností:

cd pokus
cd /home/horelica/pokus
cd ./pokus

Příkaz ***cd*** bez parametru způsobí přechod do domovského adresáře uživatele.

LS

Pro vypsání souborů v adresáři lze použít příkaz ***ls***. Bez parametrů zobrazí stručný seznam názvů souborů v adresáři. Při použití parametru ***-l*** se získá detailní výpis, přidá-li se ještě parametr ***-a***, ve výpisu se objeví i skryté soubory. Lze samozřejmě uvést specifikaci souborů, které se mají zobrazit, včetně cesty. Pokud není uvedena, ***ls*** vypíše všechny soubory v aktuálním adresáři.

Příklady:

ls m*.m	vypíše soubory s příponou **.m*, které začínají písmenem m;
ls p??	vypíše soubory, které začínají **.m* a za ním jsou právě libovolné 2 znaky;
ls /home	vypíše obsah adresáře */home*;
ls -l -R	vypíše adresář včetně podadresářů;
ls -l	vypíše adresář včetně přístupových práv k souborům.

K této skupině příkazů lze připojit příkaz ***du***, který vypíše zaplnění diskového prostoru.

MKDIR

Příkaz ***mkdir*** (make dir - vytvoř adresář) vytvoří nový adresář zadaného jména.

Příklad:

mkdir dokumenty	vytvoří adresář *dokumenty* v aktuálním adresáři.

CP

Příkaz ***cp*** („copy“ - kopíruj) vytvoří kopii zadaného souboru.

Příklad:

cp dokument kopie udělá přesnou kopii souboru *dokument* a pojmenuje ho *kopie* a soubor *dokument* bude stále na svém místě. Když se použije ***mv***, tak původní soubor nebude nadále existovat, když se použije ***cp***, tak původní soubor zůstává a je vytvořena nová kopie.

MV

Podobnou syntaxi má příkaz ***mv***, který soubor přenese do jiného adresáře, užívá se i pro přejmenování souboru.

Příklady:

mv aaa.txt bbb.txt	přejmenuje z *aaa.txt* na *bbb.txt*;
mv ccc.txt dokumenty	přenese *ccc.txt* do podadresáře *dokumenty* (adresář musí existovat).

RM

Příkaz ***rm*** (remove - odstranit) odstraní zadaný soubor. Nebude fungovat na neprázdných adresářích. ***rm -r*** smaže rekurzivně všechny soubory i adresáře v daném adresáři a nakonec samotný adresář.

FUSER

Zjištění, kdo se souborem pracuje.

Příklad: ***fuser jméno_souboru***

CAT
Slouží k vypsání obsahu souborů. Pro prosté vypsání obsahu souboru na obrazovku:

cat soubor.txt
nebo soubor v adresáři *home*
cat /home/soubor.txt

nebo vypsání obsahu podle velikosti terminálového okna, protože soubor může být jakkoli veliký a celý obsah se nemusí vejít do terminálového okna.
cat soubor.txt | more

Další možností je sloučení více souborů do jednoho:
cat soubor_A soubor_B > soubor_AB

HEAD
Vypíše začátek souboru:
head /home/pokus/dokument.txt

TAIL
Vypíše konec souboru:
tail /home/pokus/dokument.txt

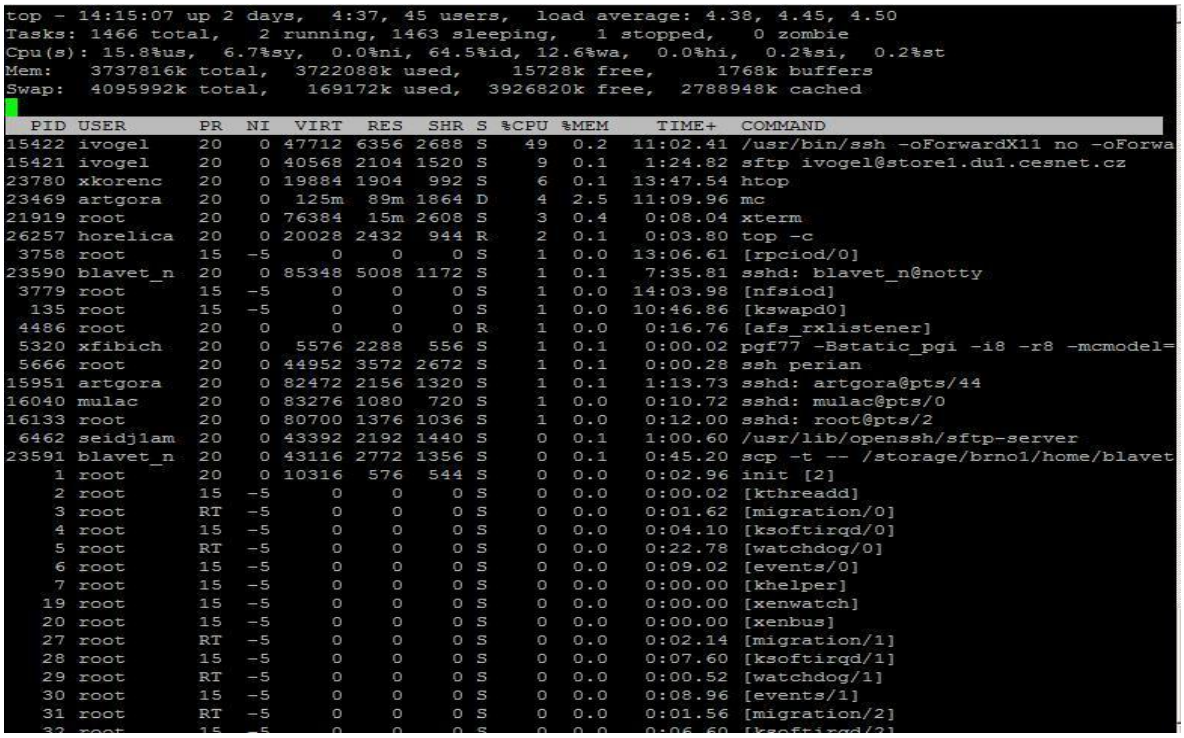
FIND
Vyhledává soubory rekurzivně, vyhledání souboru *david.txt* a podobných hledání začne v nejvyšší úrovni a bez rozlišení velikosti písmen:
find / -iname „*david*“

PS
Výpis běžících procesů na stroji. Sloupce, PID číslo procesu, identifikace terminálu, na kterém běží, využití procesorového času, jméno procesu.

Příklady:

ps a	zobrazení všech procesů;
ps ux	zobrazení všech procesů uživatele, pod kterým je tento příkaz spuštěn;
ps aux	zobrazení všech procesů všech uživatelů.

TOP
Zobrazuje informace o UNIXovém systému, běžící procesy a systémové prostředky, včetně CPU, RAM & swap a počet právě běžících úkolů. Díky tomu se může v reálném čase sledovat změna stavů procesů, jejich "boj" o procesor, aktuální velikost paměti, kterou procesy alokují a spousta dalších užitečných informací, viz obrázek 12. Defaultní časový interval změny výpisu je nastaven na 5 sekund. Pro ukončení zobrazení aktuálních procesů lze použít klávesu ***q***.



Obr. 12 Seznam všech procesů vypsaných příkazem ***top***

KILL

Nástroj sloužící k poslání signálu daným PID. Název programu je anglické slovo, které znamená „zabít“ proces. V tomto kontextu je to myšleno jako „nuceně ukončit“ proces, což dělá pouze jeden ze signálů, které ***kill*** může vyslat.

Příkladem může být to, že příkazem ***ps aux*** si lze vypsat následující informace:

```
$ ps aux
USER       PID  %CPU %MEM  VSZ RSS     TTY   STAT START   TIME COMMAND
timothy  29217  0.0  0.0 11916 4560 pts/21   S+   08:15   0:00 pine
root     29505  0.0  0.0 38196 2728 ?        Ss   Mar07   0:00 sshd: can [priv]
can      29529  0.0  0.0 38332 1904 ?        S    Mar07   0:00 sshd: can@notty
```

Obr. 13 Seznam procesů vypsaných příkazem ***ps aux***.

A cílem je ukončit proces běžící s PID=29529, takže příkaz k ukončení bude vypadat:
kill 29529

Dále je možné ukončit všechny instance programu MATLAB:
killall -s 9 matlab

Pro práci na MetaCentru je tento výčet příkazů dostačující a případné další informace lze získat např. z internetových stránek http://www.linuxsoft.cz nebo http://doors.stanford.edu/~sr/computing/basic-unix.html.

4. MetaCentrum

4.1 Historie, současnost a budoucnost [15]

MetaCentrum vzniklo jako reakce na konkrétní situaci v akademické komunitě České republiky na počátku roku 1996 v rámci pilotního projektu Fondu rozvoje vysokých škol České republiky z roku 1994. Šlo o zřízení prvních počítačových center pro podporu výkonných výpočtů. Nejdříve vznikla tři centra a v konečném důsledku pak pět center podpory výkonných výpočtů. Jednalo se o Ústav výpočetní techniky Univerzity Karlovy, Centrum výpočetních a informačních služeb Vysokého učení technického v Brně, Ústav výpočetní techniky Masarykovy univerzity v Brně, Západočeskou univerzitu v Plzni a Výpočetní centrum Českého vysokého učení v Praze. Počátkem roku 1996 byl na uvedených školách instalován hardware firem SGI, Digital (Plzeň) a IBM (SP2 na VC ČVUT).

Jednalo se o programové vybavení z oblastí výpočetní chemie a fyziky, které byly primárně zastoupeny na ÚVT MU a ÚVT UK, s určitým přesahem i na VC ČVUT, zatímco technicky orientované výpočty pokrývalo programové vybavení na ZČU, VC ČVUT, VUT a též program Fluent na ÚVT UK. Rovněž přístup k systému MATLAB byl zajištěn především na ČVUT, ZČU a MU a bylo možno nalézt i další rozdíly. Všechny počítače byly zpřístupněny všem zájemcům z akademického prostředí České republiky, tj. v podstatě všem učitelům a vědeckým pracovníkům vysokých škol a Akademie věd a do značné míry i všem studentům vysokých škol. V roce 1996 MŠMT vyhlásilo program TEN-34 CZ podpory vysokorychlostních sítí a aplikací, který otevřel možnost dalšího kvalitativního růstu center, založených a vytvořených v předcházejících letech. V rámci tohoto programu byl financován projekt MetaCentrum, jehož účelem bylo vytvoření výpočetní mřížky.

Od roku 1999 je projekt MetaCentrum začleněn do výzkumného záměru sdružení CESNET. V devadesátých letech minulého století začal rozvoj Gridů, které zajišťují vysokorychlostní síťovou infrastrukturu. Pojem Grid se objevil jako nová forma distribuovaného zpracování informací a řešení složitých problémů. Gridy se zpočátku soustředily primárně na oblast náročných výpočtů, ale postupně pronikaly do jiných oblastí, které mohly profitovat z řešení, která byla vytvořena v rámci rozvoje Gridů. Vyjmenovat lze např. oblast digitálních knihoven, e-learningu nebo kvalitních videokonferenčních či multimediálních nástrojů. V tomto roce se podařilo dokončit jednu etapu výstavby bezpečnostní infrastruktury MetaCentra - přechod na Kerberos 5 v implementaci heimdal, na jejímž rozvoji se MetaCentrum podílí. Současně byla vyřešena otázka zálohování dat v rámci MetaCentra zakoupením a následným zprovozněním páskového robota.

V letech 2000 a 2001 došlo k výrazné změně orientace MetaCentra v oblasti technického vybavení, což byl jednak důsledek omezených finančních zdrojů (s výjimkou ÚVT Praha žádný uzel výrazně do výpočetních zdrojů neinvestoval), jednak i stále pokračujícího příklonu k tzv. "commodity" řešením i v oblasti superpočítačů - zde formou orientace na clustery počítačů, zejména s procesorovou architekturou IA32 a operačním systémem Linux. Základní myšlenkou MetaCentra je nebudovat jediný uzel a naopak využívat v maximální možné míře možnosti poskytované vysokorychlostními sítěmi. Kromě zvyšování výpočetního výkonu se práce v těchto letech soustředila především na oblast bezpečnosti, plnou integraci clusterů (včetně vývoje nezbytného programového vybavení), založení portálu MetaCentra jako jednotné informační brány pro všechny uživatele i administrátory MetaCentra, rozvoj systému Perun pro správu uživatelských účtů a postupný přechod na nový dávkový systém PBS. Významné začaly být i aktivity MetaCentra na mezinárodní úrovni.

V letech 2002 a 2003 bylo ve 4 centrech k dispozici 192 procesorů (96 uzlů) s 1 GB paměti na uzel. Disková kapacita se zvýšila na 9-36 GB/uzel. Vznikl zárodek národního Gridu, tj. virtuálního distribuovaného počítače, který umožnil jak synchronní využití výpočetních zdrojů, tak i možnost používat jednotlivé uzly bez přesné znalosti umístění a do jisté míry i architektury konkrétních počítačů.

V roce 2004 bylo ve 3 uzlech - na ZČU v Plzni, v Praze v prostorách CESNETu a na MU v Brně - celkem 262 procesorů Intel Pentium (od verze III pracující na 700 MHz až pro 3 GHz Xeony) s operačním systémem Debian Linux. Část kapacit byla propojena vysokorychlostní sítí Myrinet, což umožnilo realizovat výpočty s vysokými nároky na rychlost a kapacitu přenosu mezi uzly clusteru. Uživatelům byly k dispozici i alternativní výpočetní prostředí, především 64bitové systémy IBM Power4+ a AMD Opteron, s operačním systémem SuSe Linux. MetaCentrum dále ve spolupráci se ZČU, UK a MU spravovalo i výkonné výpočetní systémy firem SGI třídy Origin, osazené procesory MIPS (v Praze a Brně) a HP/Compaq AlphaServer s procesory EV7 (v Plzni) a velkoobjemovou páskovou knihovnu s kapacitou 12 TB (systém NetWorker firmy Legato, resp. IBM), která sloužila k zálohování všech uzlů a průběžnému zálohování videoarchivu. Významnou roli v roce 2004 hrála i úzká spolupráce s projektem EGEE, mimo jiné i proto, že část zdrojů MetaCentra se stala součástí celoevropského EGEE Gridu.

V roce 2005 byly veškeré výpočetní i datové zdroje MetaCentra rozmístěny do čtyř lokalit - v sídle sdružení (skurut - primárně pro projekt EGEE, v Ústavu výpočetní techniky UK (výpočetní systémy HAL, Mat a Acharon (všechny SGI) a související diskové kapacity), na Západočeské univerzitě Pasifae (DEC/Compaq/HP) a clustery Nympha (vlastní cluster MetaCentra) a Minos (cluster patřící ITI ZČU spravovaný provozní skupinou MetaCentra), na ÚVT MU Eru, Grond a Gandalf a clustery Skirit (vlastní systém MetaCentra) a Perian (majetek NCBR MU ve správě MetaCentra). Pro experimentální účely byl vyhrazen dual CPU Power4 a počítač od IBM (majetek MetaCentra). Řada činností byla ovlivněna rozhodnutím přechodu na PKI autentizaci s využitím USB tokenů.

Roku 2006 bylo celkem k dispozici přes 600 procesorů spravovaných ve 4 uzlech. V první polovině roku do prostředí MetaCentra byla plně integrována nová pásková knihovna NEO8000 firmy Overland Storage, umístěná v uzlech v Plzni a Brně, s celkovou nekomprimovanou kapacitou 400 TB a předřazeným diskovým polem s kapacitou 8 TB. A koncem roku byla zavedena první verze zálohovací politiky, která garantovala minimální dobu dostupnosti záloh v délce 3 měsíce. Všechna zálohovaná data byla ukládána duálně v Plzni i Brně. V roce 2006 pokračovala úzká spolupráce v rámci bezpečnosti, která se kromě již tradičního využívání služeb Certifikační autority zaměřila především na oblast federativních přístupů a jejich propojení s gridovými autentizačními a postupně i autorizačními službami. Na mezinárodní úrovni pokračovala úzká spolupráce s projektem EGEE a návazným projektem EGEE II, které odpovídají za budování celoevropské gridové infrastruktury.

V roce 2007 bylo integrováno BM Blade řešení do infrastruktury MetaCentra. Ve všech případech byla nasazena virtualizační technologie Xen. Hlavní činností MetaCentra v roce 2007 byly první kroky tvorby virtualizované gridové infrastruktury. Realizované činnosti lze rozdělit do následujících oblastí: vlastní virtualizace infrastruktury, spočívající jednak v instalaci virtuálních počítačů, jednak v modifikaci plánovacího systému a dalších součástí provozu MetaCentra tak, aby s virtualizovaným prostředím mohly pracovat; nasazení IPv6 jako základního protokolu komunikace mezi uzly MetaCentra a zahájení přechodu vybraných služeb na IPv6; řešení úložných kapacit, integrovatelných do virtualizovaného prostředí. Pro uživatele MetaCentra byla zřízena autentikace Identity Provider.

V roce 2008 MetaCentrum spravovalo na 577 fyzických strojů (více než 1 200 jader). Průměrné využití novějších strojů (zejména 16jaderné počítače manwe a stroje skirit 49-83 a skurut 33-66 s procesory s frekvencí 3 a více GHz) bylo kolem 50 %, což je srovnatelné nebo lepší než mezinárodně udávané hodnoty pro podobná prostředí. Zájem o starší, méně výkonné stroje, je menší, využití se pohybuje mezi 10 a 30 %. Stav infrastruktury MetaCentra je sledován systémem Nagios, který byl po předchozím zkušebním využití nasazen v roce 2008 do plného provozu. Přechod na virtualizované prostředí probíhal taktéž v roce 2008.

Rok 2009 znamenal některé investiční aktivity, které se soustředily především do nákupu nového clusteru, který byl náhradou starého clusteru s 32bitovými procesory umístěného v sídle CESNETu. Nový cluster byl vybaven uzly s procesory Intel Nehalem (28 uzlů po dvou čtyřjádrových procesorech, celkem 224 jader) propojenými technologií Infiniband 4×QDR (40 Gb/s). Dále byly pořízeny dva stroje Sun X4600, každý s 32 jádry. Byla zavedena nová verze nástroje *Pbsmon*, který na portálu MetaCentra zobrazuje aktuální stav vytížení virtualizované infrastruktury. Rok 2009 byl v oblasti virtualizace ve znamení cesty virtualizačních nástrojů blíže uživatelům.

Technicky to znamenalo integraci existujících řešení do nového celku sloužícího k poskytování nové služby. Využití virtualizace jako nástroje optimalizace infrastruktury a její správy samozřejmě zůstalo důležitou strategickou orientací. V roce 2009 byla proto navržena a implementována autorizační infrastruktura, která umožňovala centralizované nastavení přístupových politik a jejich jednotnou správu. Služba je plně integrována se systémem pro správu zdrojů Perun, ze kterého přebírá informace o dostupných službách a uživatelích a jejich identitách.

V roce 2010 byla navýšena disková kapacita pole na 124 TB. Byl implementován systém Pakiti, který slouží k monitorování softwarových balíků, zejména s důrazem na identifikaci bezpečnostních problémů a nedostatečně aplikovaných záplat. Bylo provedeno mnoho úprav na plánovacím systému z důvodu efektivnějšího využívání výpočetních zdrojů. Byl pořízen nový cluster Tarkil (CESNET), kterým byl nahrazen cluster Skurut (CESNET) v Praze. Byl obnoven stroj Ajax v Plzni (ZČU) a došlo k rozšíření clusteru Alela (FEEC VUT) a Hermes (JČU). Na žádost vlastníků byly odstaveny clustery Perian17-68 a Perian 69-76 (oba PřF MU). Na Západočeské univerzitě v Plzni byl zprovozněn cluster vybavený grafickými koprocesory (GPU) jako prostředek k urychlení výpočtů.

V roce 2011 byly pořízeny dva clustery, každý o kapacitě přes 600 jader a s úložnou kapacitou 100 TB – z projektu eIGeR byl pořízen cluster hostovaný na Západočeské univerzitě v Plzni, tvořený klasickými výpočetními uzly se dvěma procesory (24 jader) v jednom uzlu, zatímco z prostředků velké infrastruktury byl pořízen cluster větších SMP strojů (56 jader) hostovaný na Masarykově univerzitě v Brně. V tomto roce pokračovalo připojování clusterů dalších organizací. Během roku byl připojen cluster pořízený FAV ZČU, ve kterém jsou dostupné vlastníkům i dalším uživatelům i výpočetní grafické karty NVIDIA (deset uzlů) a cluster Luna, převzatý od FZÚ AV ČR. Z prostředků projektu CEITEC byl povýšen cluster perian (320 nových jader), který je nadále připojen do MetaCentra a jehož části jsou zpřístupněny dalším uživatelům. Na konci roku byl integrován první cluster CERIT-SC, čímž se zdroje MetaCentra rozrostly o dalších 8 strojů se SMP architekturou (640 jader). Dále v závěru roku 2011 byla ve spolupráci s CERIT-SC zprovozněna i první pilotní instalace cloudového rozhraní, přes které je dostupno dalších 10 výpočetních uzlů s celkovou kapacitou 100 jader. V roce 2011 MetaCentrum kompletně opustilo komerční systém PBSPro, používaný pro správu úloh, a přešlo na volně dostupný systém Torque, který byl výrazně rozšířen pro potřeby MetaCentra. Ročenka za rok 2012 bohužel není ještě zpracována, z tohoto důvodu není uvedena.

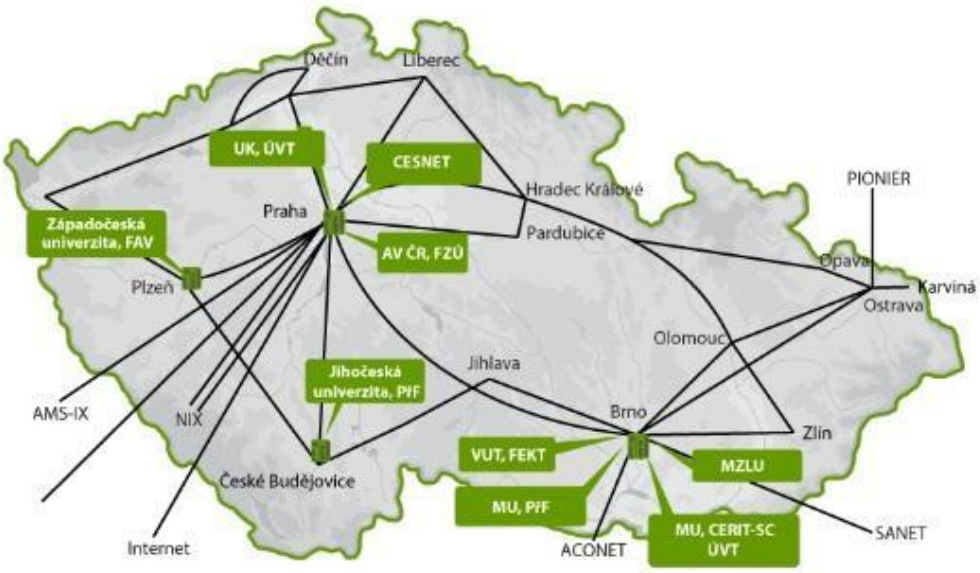
Co se týče současnosti a budoucnosti MetaCentra, tak lze čerpat ze stránek MetaCentra), kde jsou uvedeny projekty, které v současnosti běží (http://www.metacentrum.cz/cs/about/projects.html). Stručně lze zmínit následující:

- ***EPIKH - Exchange Programme to advance e-Infrastructure Know-How***. Hlavní cíle projektu jsou posilovat vliv e-infrastruktury ve vědeckém výzkumu definováním a uskutečňováním podpůrných programů na vzdělávacích akcích, jako jsou Gridové školy a kurzy High-performance computing. Rozšiřovat zapojení v aktivitách e-Science a spolupráci jak geograficky, tak mezi jednotlivými obory. Březen 2009 až březen 2013.
- ***CHAIN-REDS - Coordination and Harmonisation of Advanced e-Infrastructures for Research and Education Data Sharing***. Pokračování projektu CHAIN je koordinace současného gridového úsilí a jeho výsledků s vizí harmonického a optimalizovaného interakčního modelu e-infrastruktury a specificky gridových rozhraní mezi Evropou a zbytkem světa dle modelu definovaného projektu CHAIN. Prosinec 2012 až červen 2015.
- ***CHAIN - Coordination and Harmonisation of Advanced e-INfrastructures***. Cílem projektu CHAIN je koordinace současného gridového úsilí a jeho výsledků s vizí harmonického a optimalizovaného interakčního modelu e-infrastruktury a specificky gridových rozhraní mezi Evropou a zbytkem světa. Projekt bude definovat koherentní operační a organizační model, v němž řada evropských zemí/regionů bude působit - ve spolupráci s projektem EGI - jako mosty/brány k dalším regionům a kontinentům. Prosinec 2010 až listopad 2012.

- ***EMI - European Middleware Initiative***. Projekt EMI je spojenectvím tří hlavních poskytovatelů middlewaru v Evropě, konkrétně ARC, gLite, Unicore a dalších konsorcií. Cílem projektu EMI je dodávka konsolidované sady komponent middlewaru vhodných pro nasazení v EGI, PRACE a dalších distribuovaných výpočetních infrastrukturách, rozšíření interoperability mezi gridy a jinými výpočetními infrastrukturami a ustavení trvale udržitelného modelu pro údržbu a vývoj middlewaru splňujícího požadavky uživatelských komunit. Květen 2010 až duben 2013.
- ***EGI InSPIRE - European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe***. Projekt EGI InSPIRE pokračuje v přechodu k trvale udržitelné panevropské e-infrastruktuře iniciované sérií EGEE projektů pomocí podpory Gridů v oblastech high-performance a high-throughput výpočtů. Projekt poskytne centrální koordinující organizaci nutnou k integraci a vzájemné interoperabilitě individuálních národních gridových infrastruktur (NGI), ke sběru požadavků a poskytování uživatelské podpory pro současné a nové uživatele. Navíc bude sloužit pro definici, ověření a integraci UMD, middlewaru od externích poskytovatelů, potřebného pro přístup na infrastrukturu. Květen 2010 až duben 2014.

4.2 Infrastruktura MetaCentra VO[16]

MetaCentrum VO je tzv. “catch-all” virtuální organizace, která sdružuje všechny registrované uživatele do MetaCentra. Je součástí Národní Gridové iniciativy (NGI_CZ). Nabízí všem akademickým pracovníkům, zaměstnancům a studentům nástroje (výpočetní, úložné kapacity a aplikační programy) pro řešení jejich problémů z oblastí výpočetní chemie, materiálové a strukturní simulace, simulace proudění plynů a kapalin, rozpoznávání a generování řeči, fyzikální geodézie, ekologické modelování, zpracování videa, data mining, analýzy lékařských obrazů atd. Členství v MetaCentru je bez omezení, všem osobám z akademického prostředí ČR a jedině za účelem výzkumu. Zahraniční studenti na českých vysokých školách mohou MetaCentrum využívat po dobu svého studia, stejně tak zahraniční zaměstnanci českých akademických institucí mohou MetaCentrum používat po dobu svého zaměstnaneckého poměru. Tito uživatelé mají po vyplnění přihlášky přístup k výpočetním zdrojům, avšak musejí počítat s tím, že vlastníci těchto zdrojů mají výhradní právo na jejich užívání.



Obr. 14 Infrastruktura MetaCentra VO, převzato a upraveno z [16]

4.3 Statistiky MetaCentra[17]

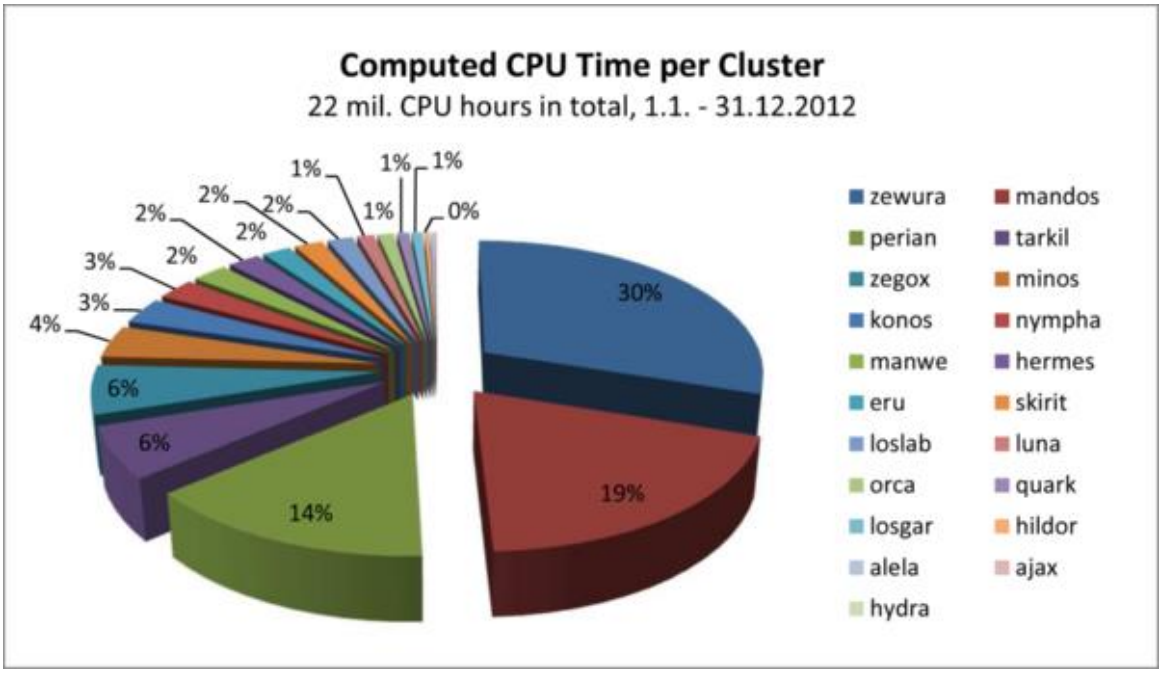
Ve statistikách MetaCentra lze najít přehled o propočítaném čase, počtu propočítaných úloh, době čekání, době běhu úloh, složení jednotlivých institucí, používaných aplikací, nejaktivnějších uživatelů, vytížení strojů, využití volně přístupných clusterů a strojů institucemi.

V diplomové práci jsou k dispozici statistiky provozu MetaCentra VO za rok 2011 a 2012, viz tabulka 1.

Tabulka 1 Statistiky MetaCentra VO za období 2011 a 2012, převzato a upraveno z [17]

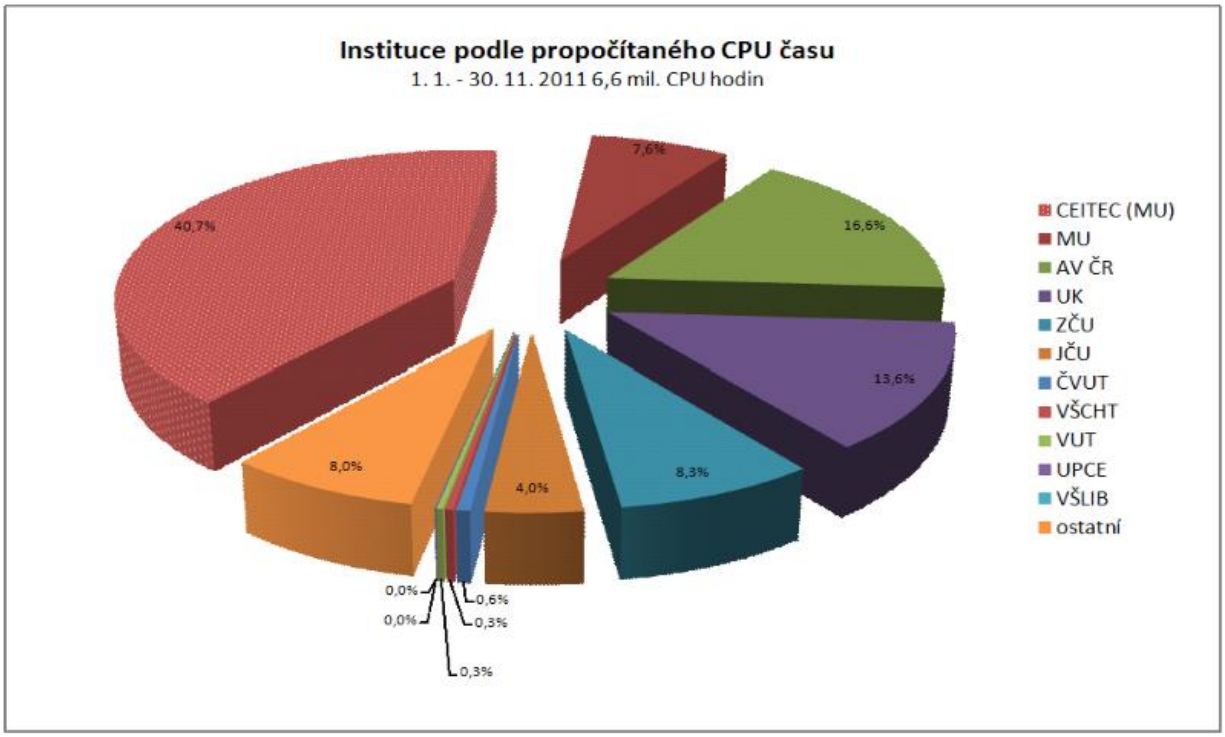
Název	**Rok 2011**	**Rok 2012**
Počet úloh	609 tis.	108 tis.
Celkový propočítaný čas	742 CPU let	2.5 tis. CPU let
Počet aktivních uživatelů	491	613
Počet prodloužených účtů	314	301
Počet nově získaných uživatelů	177	312
Počet uživatelů, kteří spustili alespoň jednu úlohu	240	322
Objem dat na diskovém poli	85 TB	350 TB

V období leden až prosinec 2012 uživatelé propočítali 22 milionů CPU hodin (2,5 tis. CPU let) ve více než milionu úloh, viz graf 3. Ve srovnání s rokem 2011 trojnásobně narostl celkový propočítaný čas, což koresponduje s trojnásobným nárůstem dostupné výpočetní kapacity. Průměrné vytížení volně dostupných strojů v MetaCentru se pohybovalo mezi 60 a 95 %, vytížení okolo 70 % je optimální, vyšší vytížení již znamená faktickou saturaci a způsobuje delší doby čekání úloh ve frontě.



Graf 3 Celkový propočítaný CPU čas na jednotlivých clusterech MetaCentra za rok 2012, převzato a upraveno z [17]

Pro zajímavost je uveden ještě přehled propočítaného CPU času jednotlivých institucí, graf 4.



Graf 4 Instituce podle propočítaného CPU času, převzato a upraveno z [18]

Podrobnější statistiky za jednotlivé roky lze dohledat na stránkách MetaCentra VO, v menu, stav zdrojů, statistiky využití (http://metavo.metacentrum.cz/cs/state/stats/index.html). Roční a technické zprávy, ročenky, publikace apod. lze dohledat na http://www.metacentrum.cz/cs/about/results/index.html.

5. Hardware MetaCentra

Výpočetní zdroje MetaCentra se neustále mění, procesy, paměť, diskové pole apod. V této kapitole bude stručně uveden přehled poskytovatelů, druh hardwaru a vizuální představa, co je čelní uzel, výpočetní uzel.

5.1 Hardware[19]

Níže je uveden přehled hardwaru jednotlivých institucí k datu 1. 4. 2013.

CERIT-SC (2184 CPU)

- zegox.cerit-sc.cz (576 CPU, 48 uzlů)
- zewura.cerit-sc.cz (1600 CPU, 20 uzlů)
- kudu.cerit-sc.cz (8 CPU)

CERIT-SC, CESNET (112 CPU)

- manwe.ics.muni.cz (112 CPU, 7 uzlů)

CESNET (2912 CPU)

- gram.zcu.cz (160 CPU, 10 uzlů)
- dukan.ics.muni.cz (240 CPU, 10 uzlů)
- ramdal.ics.muni.cz (32 CPU)
- hildor.metacentrum.cz (416 CPU, 26 uzlů)
- mandos.ics.muni.cz (896 CPU, 14 uzlů)
- minos.zcu.cz (600 CPU, 50 uzlů)
- tarkil.cesnet.cz (224 CPU, 28 uzlů)
- eru.ruk.cuni.cz (64 CPU, 2 uzly)
- nympha.zcu.cz (160 CPU, 20 uzlů)
- skirit.ics.muni.cz (120 CPU, 30 uzlů)

FEKT VUT (96 CPU)

- alela.feec.vutbr.cz (96 CPU, 12 uzlů)

FI MU (58 CPU)

- quark.video.muni.cz (58 CPU, 13 uzlů)

FZÚ AV ČR (70 CPU)

- luna.fzu.cz (70 CPU, 5 uzlů)

Loschmidt Laboratories (264 CPU)

- loslab.ics.muni.cz (168 CPU, 14 uzlů)
- losgar.ics.muni.cz (96 CPU, 2 uzly)

NCBR (568 CPU)

- orca.ics.muni.cz (72 CPU, 18 uzlů)
- perian1-10.ncbr.muni.cz (80 CPU, 10 uzlů)
- perian11-20.ncbr.muni.cz (64 CPU, 8 uzlů)
- perian21-40.ncbr.muni.cz (160 CPU, 20 uzlů)
- perian41-56.ncbr.muni.cz (192 CPU, 16 uzlů)

PřF JČU (96 CPU)

- hermes.metacentrum.cz (96 CPU, 12 uzlů)

ZČU (128 CPU)

- konos.fav.zcu.cz (120 CPU, 10 uzlů)
- ajax.zcu.cz (8 CPU)

Hardware Skirit

Čelní uzel představuje místo, kam se přihlašují uživatelé a připravují si zde úlohy, v tomto případě skirit.ics.muni.cz, obrázek 15.

Obr. 15 Čelní uzel skirit.ics.muni.cz, převzato a upraveno z [19]

Konfigurace čelního uzlu clusteru

DNS jméno	skirit.ics.muni.cz;
procesor	2x Intel Xeon 3.06 GHz;
paměť	1 GB;
disk	135 GB;
síť	Ethernet 1 Gb/s;
příslušnost k	CESNETu.

Výpočetní uzel představuje místo, kde se provádí jednotlivé výpočty, v tomto případě skirit 17-48.ics.muni.cz, obrázek 16.

Obr. 16 Výpočetní uzel Skirit - uzly č. 17 až 48 (celkem 32), převzato a upraveno z [19]

Konfigurace každého uzlu

DNS jméno	skirit 17-48.ics.muni.cz;
procesor	2x Intel Xeon 2.4 GHz;
paměť	1 GB;
disk	66 GB;
síť	Ethernet 1 Gpbs + Myrinet;
příslušnost k	CESNETu.

Obr. 17 Výpočetní uzel Skirit - uzly č. 49 až 83 (celkem 35), převzato a upraveno z [19]

Konfigurace každého uzlu

DNS jméno	skirit 49-83.ics.muni.cz;
procesor	2x Dual Core Xeon 5160 3 GHz, 4 MB cache;
paměť	4 GB;
disk	73 GB;
síť	1 Gbps Ethernet;
příslušnost k	CERIT-SC/MU;

speciální fyzický stroj se dvěma virtuálními stroji.

Obr. 18 Skirit - uzel č. 84 (celkem 1), převzato a upraveno z [19]

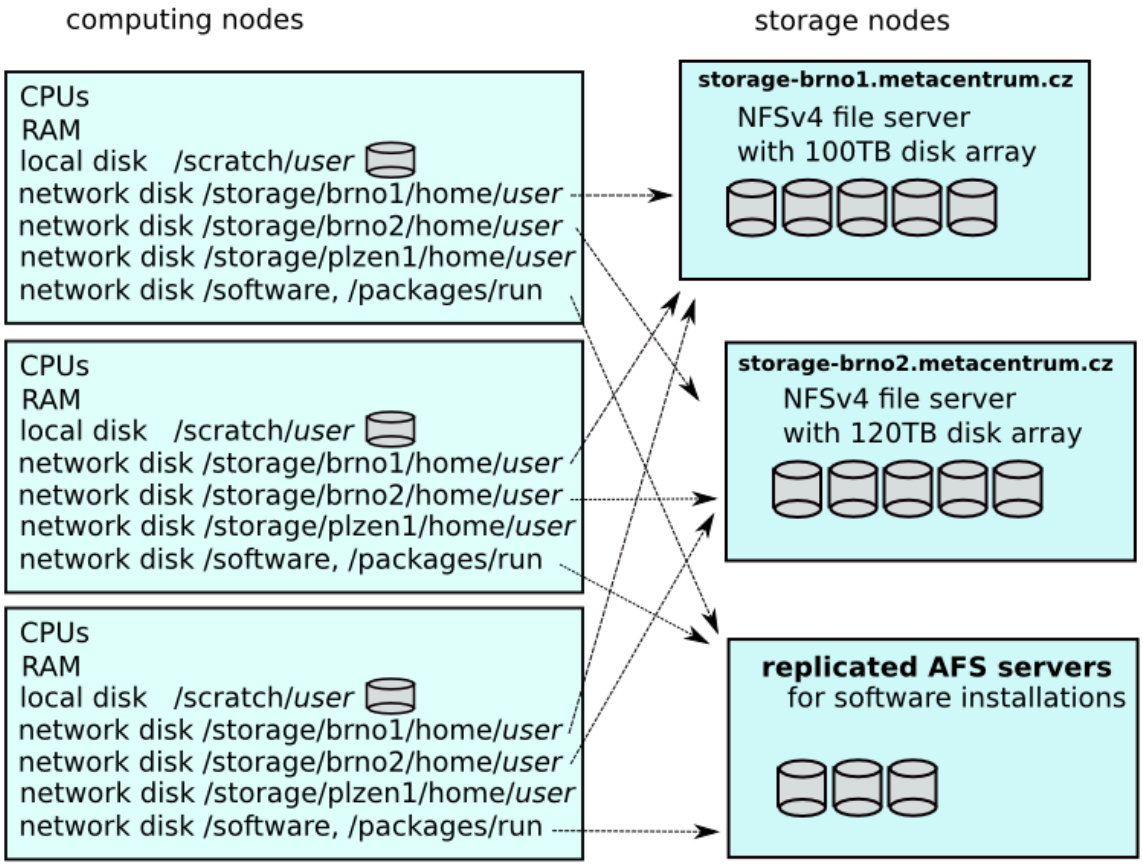
Konfigurace každého uzlu

DNS jméno	skirit84.ics.muni.cz;
procesor	2x Quard Core X5355 2,6 GHz;
paměť	8 GB;
disk	146 GB;
síť	1 Gbps Ethernet;

speciální fyzický stroj se dvěma virtuálními stroji Infiniband.

6. Souborové systémy v MetaCentru

Schéma souborových systémů MetaCentra zobrazuje obrázek 19.



Obr. 19 Schéma souborových systémů MetaCentra, převzato a upraveno z [20]

V MetaCentru jsou k dispozici na Computing nodes (výpočetních uzlech) a Storage nodes (čelních uzlech) tyto disky a svazky:

- rychlý lokální disk určený proměnnou *$SCRATCHDIR*, což může být:
  - rychlý lokální HDD disk mapovaný do adresáře */scratch*;
  - ultrarychlý, ale malý SSD disk mapovaný do adresáře */ssd*;
  - rychlý síťový svazek sdílený pro všechny uzly daného clusteru mapovaný do adresáře */scratch.shared*;
- několik velkých diskových polí určených pro dlouhodobé držení dat, mapovaných do adresáře */storage* přes souborový systém NFSv4;
- síťový replikovaný souborový systém určený pro instalaci software, mapovaný do adresářů */software* a */packages* přes souborový systém AFS.

6.1 Druhy svazků v MetaCentru[20]

Svazky scratch (svazky pro dočasná data)
Tyto svazky jsou rychlé úložné svazky, které slouží pro pracovní data běžících úloh a jsou umístěny na lokálních discích jednotlivých uzlů (SSD nebo HDD disky). Výhradně slouží pro ukládání dat běžících úloh, tedy výsledků nebo mezivýsledků úlohy do doby, než úloha doběhne. Po skončení úlohy by měl uživatel data z tohoto disku smazat. Jistou nevýhodou je, že data umístěná v tomto svazku se nedají sdílet s jinými stroji popřípadě přes všechny uzly daného clusteru.

Identifikace vyhrazeného prostoru je vždy dostupná skrze systémovou proměnnou *$SCRATCHDIR*. Při zadání úlohy musí uživatel požádat o volný prostor parametrem *-l scratch*, a podle velikosti požadovaného prostoru bude přidělen nejrychlejší typ scratche s dostatečným místem, tj. nejdřív */ssd*, pak */scratch* a pak */scratch.shared*. Jako příklad je uvedeno to, že uživatel si chce zarezervovat 1 MB místa na tomto svazku:

qsub -l nodes=1:ppn=1,-l scratch=1024,mem=500mb,matlab=1

Pro kopírování dat do a z scratche lze použít skript, ve kterém je pomocí příkazu ***cp*** zajištěno nakopírování dat pro nastávající výpočet. Opět po skončení výpočtu se doporučuje data z tohoto adresáře odstranit, např. příkazem ***rm -r $SCRATCHDIR***.

Svazky /home[20]
Svazky /home, tedy domovské adresáře uživatelů, jsou postaveny na systému NFSv3. Protože autentizace v NFSv3 je slabá (postavená pouze na číslu uživatele a skupiny), jsou svazky */home* sdíleny pouze v rámci jednoho clusteru. Výjimkou jsou clustery v Brně, které sdílejí jeden */home*. Je tedy dobré mít na mysli, že například */home* na skiritech v Brně je jiný souborový systém, než */home* na clusteru nympha. Hlavní motivací pro svazky */home* je fyzická blízkost diskového úložiště a konkrétního clusteru, což zejména zvyšuje spolehlivost. Cluster bez /home adresářů by byl totiž pro uživatele naprosto nepoužitelný. Tyto svazky mají kapacitu v jednotkách TB a poměrně restriktivní kvóty. Byly zamýšleny jako prostor pro data, se kterými uživatelé právě pracují zejména na konkrétním clusteru („právě rozpracovaný projekt“), kde ono „právě“ má být chápáno ve smyslu „v posledním roce jsem s těmito daty pracoval“.

Svazky /storage[20]
Svazky */storage* jsou postaveny na NFSv4. Svazek */storage* slouží pro ukládání dat uživatelů, typicky do adresářů */storage/home/<login>*. Je dostupný na strojích s vlastností NFS4. Pro řízení přístupu k němu se používá systém Kerberos (tzn., že uživatel potřebuje tzv. „lístky“) a lze si jej připojit z libovolného vlastního stroje. Disková pole poskytující */storage* jsou umístěna v Brně a nově i v Plzni. Svatým grálem souborových systémů je, aby taková informace nebyla vůbec důležitá. Bohužel svět není ideální, takže na vzdálenosti záleží při operaci zápisu. Zápis na */storage* v Brně bude nejrychlejší ze strojů v Brně, potom z Prahy a nejpomalejší bude z Plzně. Vliv bude větší při zápisu malých souborů, menší pro velké. Pro čtení jsou rozdíly zanedbatelné.

Označení svazků (platné k 1. 4. 2013):

/storage/brno1/home	(původní /storage/home v Brně)
/storage/brno2/home	(nové diskové pole v Brně)
/storage/brno3-cerit/home	(nové diskové pole centra CERIT-SC v Brně)
/storage/plzen1/home	(nové diskové pole v Plzni)

Svazky AFS[20]

AFS (Andrew File System) je distribuovaný souborový systém, který umožňuje poskytovat data přes síť (internet). V případě MetaCentra lze sdílet data mezi stroji zapojenými do sítě MetaCentra. Výhodou je tedy to, že data jsou dostupná všude v rámci MetaCentra, nevýhodou je, že se nehodí pro kopírování dat přesahujících velikost 300 MB a také velikost svazku by neměla přesahovat 2 GB. Přístupy k AFS jsou řízeny autentizačním systémem Kerberos, který využívá tzv. lístky, které lze přirovnat k certifikátům (více na internetových stránkách https://wiki.metacentrum.cz). Problematika AFS je poněkud složitější a cílem není zde uvádět veškeré informace, a proto veškeré další informace lze najít v dokumentaci na internetových stránkách MetaCentra (https://wiki.metacentrum.cz/wiki/AFS). Uživatel by měl mít v podvědomí to, že existují přístupová práva a kvóty na těchto svazcích.

6.2 Přístupová práva a kvóty[20]

Na všech svazcích (např. výše uvedené AFS) a souborových systémech v MetaCentru jsou aplikována přístupová práva, tzn., že uživatelé si navzájem nemohou smazat obsah domovských adresářů nebo měnit obsah apod. Zároveň si ale mohou vzájemně nastavit určitá práva přístupu do těchto adresářů nebo souborů. V první řadě je důležité si zjistit, jaká práva uživatel má na svůj domovský adresář. To lze zjistit příkazem ***ls –l***.

práva	**id uživatele**	**vlastník**	**skupina**	**velikost**	**datum**	**soubor/adresář**
drwxr-xr-x	1	josef	admin	2657	2013-03-07 13:49	home/
-rw-r--r--	1	josef	admin	46	2013-03-07 18:10	vysledek.mat

Přístupová práva k adresáři *home*, např: drwxr-xr-x. Tento zápis si lze pro větší přehlednost rozdělit do čtyř sloupců:

typ	**práva vlastníka**	**práva skupiny**	**práva ostatních**
d	rwx	r-x	r-x

První sloupec je jednoznakový a určuje, o jaký typ souboru se jedná:

- **-** = soubor\
- **c** = znaková zařízení (tiskárna…)
- **b** = bloková zařízení (disky…)
- **d** = adresář
- **l** = link

Další tři sloupce již označují skupiny, kterým lze práva přiřazovat. Práva vlastníka určují, co se souborem může dělat vlastník souboru. Práva skupiny udávají, co se souborem můžou provádět ostatní členové skupiny, které soubor náleží. A nakonec jsou práva ostatních, kteří nepatří do skupiny, jíž soubor přísluší.

Význam r, w, x

práva	**význam**	**pro soubor**	**pro adresář**
r	Read	číst soubor	vypsat obsah adresáře
w	Write	zapisovat do souboru	vytvářet nebo mazat soubory či adresáře
x	eXecute	spouštět soubor	procházet adresářem

Ke změně práv k souborům, adresářům nebo svazkům slouží příkaz ***chmod***, který má tvar:
chmod o+rx jmeno_souboru

Prvním znakem se určí, čí práva se budou měnit:

u	user	vlastník
g	group	skupina
o	others	ostatní
a	all	všichni

Následně se specifikuje typ práva:

+	přidání práva;
-	odebrání práva;
=	kompletní změna práv na nastavenou hodnotu.

Příkladem může být přidělení práva na soubor *vysledek.mat*:
chmod a=rwx vysledek.mat

Vypsáním práv souboru (příkaz ***ls - l***) se zobrazí: ***-rwxrwxrwx vysledek.mat***

Kvóty slouží zejména jako mechanismus sledování zaplnění diskového prostoru pro správce a jako ochrana před chybou, která by zaplněním disků znemožnila práci všem uživatelům. Kvóty jsou různé pro různé uživatele. Přehled svých kvót lze zjistit na webové stránce MetaCentra v menu, přehled kvót, obrázek 20.

user	dir	space				files			
		used	soft quota	hard quota	grace	used	soft limit	hard limit	grace
horelica	/storage/brno1/home/horelica	4kB	5TB	5TB		2	-	-	
horelica	/storage/brno2/home/horelica	12kB	3TB	5TB		4	-	-	
horelica	/storage/brno3-cerit/home/horelica	128kB	1TB	3TB	none	2	-	-	none
horelica	/storage/plzen1/home/horelica	704kB	1TB	3TB	none	11	-	-	none

Obr. 20 Přidělené diskové kvóty na úložištích, převzato a upraveno z [21]

7. Aplikace MetaCentra

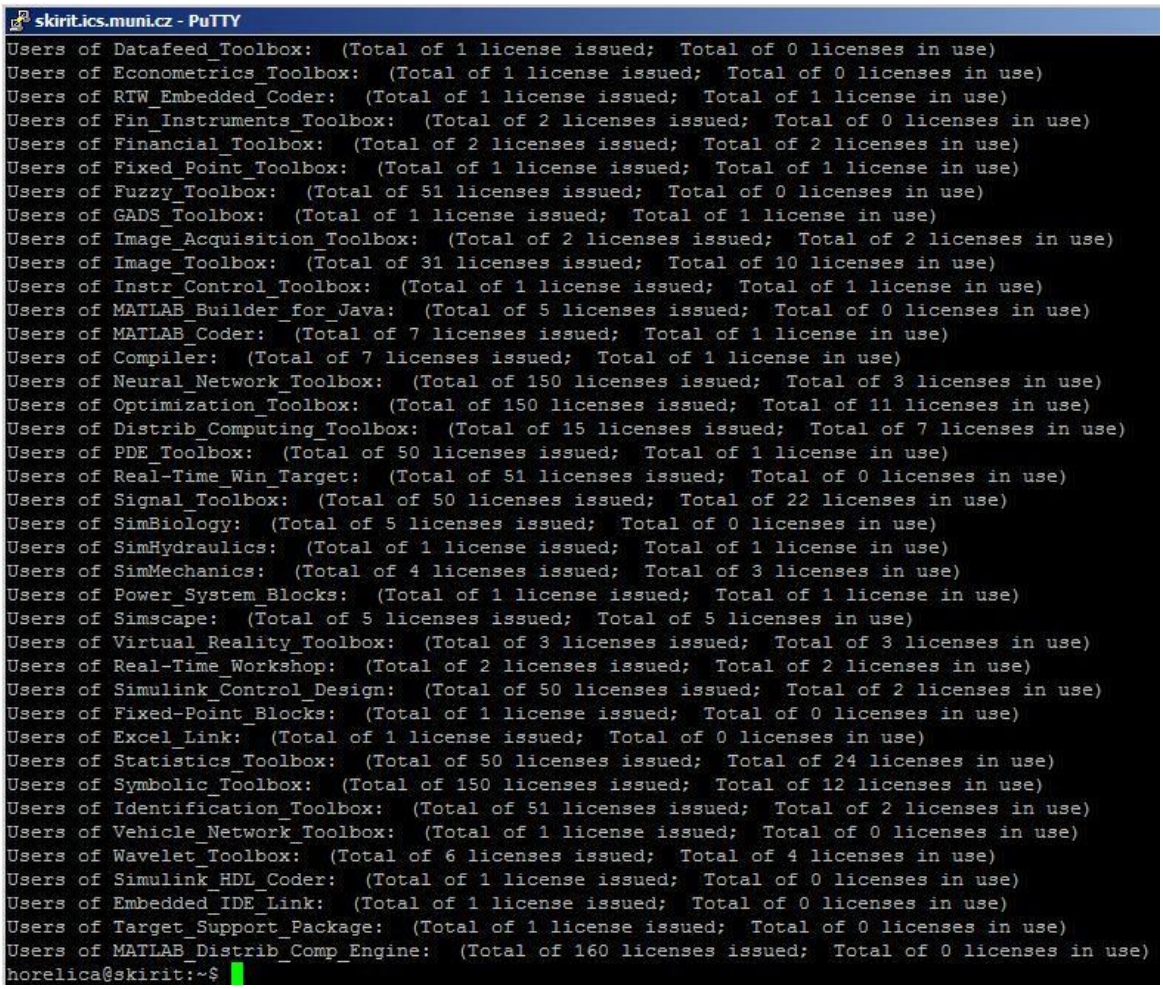
Seznam aplikací, které MetaCentrum nabízí, lze dohledat na internetových stránkách MetaCentra v kategorii aplikace. V této kapitole bude uveden MATLAB a stručně budou představeny aplikace Maple a Ansys CFD.

7.1 MATLAB a jeho používání v MetaCentru[12]

V této části bude obecně popsáno, jak postupovat při využívání MATLABu v MetaCentru.

Na všech strojích je možné si připojit MATLAB aktuální dostupnou verzi s využitím softwarových modulů. V současné době je na MetaCentru dostupná verze MATLAB 8.0 s tím, že je možnost využívat i starší verze. Uživatel si dostupné používané licence zobrazí příkazem:

/software/matlab-8.0/etc/lmstat -a|grep "in use"



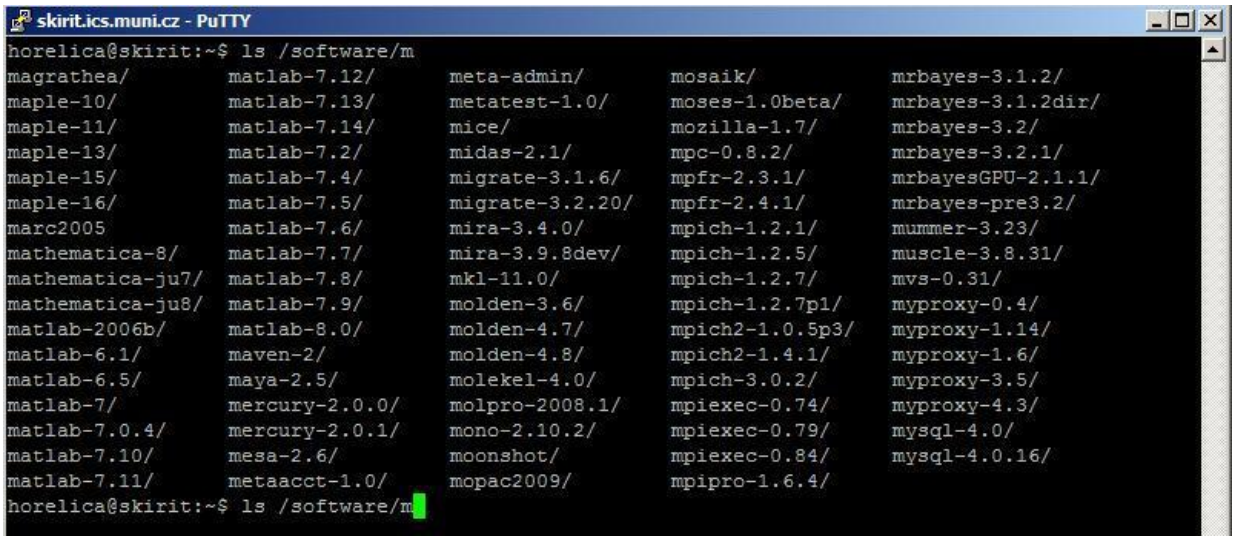
Obr. 21 Vypsání dostupných licencí programu MATLAB verze 8

Pokud uživatel neví přesný název aplikace, tak si jej může vypsat pomocí příkazu:

ls /software/

Případně zadáním počátečních znaků z názvu programu. V uvedeném příkladu si lze vypsat programy začínající na „*m*“. Po zadání příkazu musí uživatel namísto ***Enter*** stisknout klávesu ***Tab***.

ls /software/m



Obr. 22 Vypsání dostupných programů na MetaCentru začínajících na „*m*“

Nápověda MATLABu je dostupná v adresáři *$MATLABROOT/help* a online je na stránkách firmy MathWorks. Licence je možné dohledat i na stránkách MetaCentra.

MetaCentrum umožňuje interaktivní práci, nebo práci pomocí dávkových souborů. Dále je možnost v interaktivním režimu využívat GUI MATLABu, toto grafické prostředí MATLABu je přenášeno na počítač uživatele. Avšak je nutné, aby na uživatelském PC běžel nějaký X-server pro MS-Windows, například Xming, Cygwin/X, X-Win32, ReflectionX nebo Exceed. Pro SSH (Secure Shell - zabezpečená komunikace mezi počítači) s podporou tunelování X-protokolu lze použít program PuTTY nebo Cygwin. Nicméně interaktivní způsob spuštění pravděpodobně nepřinese zásadní zvýšení výkonu oproti spuštění MATLABu přímo na stroji, dokud se nepoužije paralelizace. Interaktivní spuštění lze využít pro vývoj aplikace a po dokončení vývoje ji spustit dávkově. Interaktivní režim představuje následující kroky.

Interaktivní použití MATLABu

Uživatel má připravené zdrojové MATLABovské soubory, které si následně nakopíruje do svého adresáře na MetaCentru. Než tyto MATLABovské soubory spustí, musí požádat plánovací systém o výpočetní zdroje.

#Pozadavek na rezervaci vypocetnich zdroju, fronta, uzel (node), procesor… atd., licence MATLABu
qsub -I -q short-l nodes=1:ppn=1,matlab=1

#Inicializace prostredi
module add matlab

#Prikaz pro interaktivni textove rozhrani
matlab –nosplash –nodisplay –nodesktop

#Nebo jen prikaz pro interaktivni graficke rozhrani
matlab

V případě interaktivního grafického rozhraní je zapotřebí mít spojení na X-server, které je na obrázku 29.

Dávkové použití

Dávkové spouštění MATLABovských úloh na MetaCentru znamená, že uživatel má připravené zdrojové MATLABovské soubory, které si následně nakopíruje na MetaCentrum a pomocí shellových skriptů (přípona **.sh*) tyto soubory spouští na MetaCentru. Shellový skript obsahuje informace pro plánovací systém, tzn. požadavky na výpočetní zdroje, licence atd., více kapitola 9.

Příklad shellového skriptu:

#Jako interpreter prikazu se pouzije Bash (UNIXovy program)
#!/bin/bash

#Fronta typu Short
#PBS -q short

#Typ uzlu, procesor, licence
#PBS -l nodes=1:ppn=1,matlab=1

#Nastaveni promenne PATH (cesty), aby se nasel MATLAB
. /packages/run/modules-2.0/init/bash

#Inicializace prostredi
module add matlab

#MATLAB vstoupi do adresare, kde se nalezaji soubory pro vypocet
cd /nejaky_adresar/

#Spusti MATLAB bez grafického rozhrani, v davkovem rezimu, MATLABovsky soubor bez pripony
matlab -nodisplay -r "nazev_m_souboru_bez_pripony"

V příkazové řádce uživatel uvede následující příkaz: ***qsub start.sh***, tím dojde ke spuštění výpočtu.

Shellových skriptů může být i více, lze je spouštět hromadně pomocí hlavního skriptu, ve kterém jsou uvedeny příkazy:

#Jako interpreter prikazu se pouzije Bash (UNIXovy program)
#!/bin/bash

#Spust v adresari home shellovy soubor davka1.sh
qsub ./home /davka1.sh

#Spust v adresari home shellovy soubor davka2.sh
qsub ./home/davka2.sh

V kapitole 9 je přehledně uvedeno, jak používat MATLAB při spouštění interaktivních, dávkových, paralelních nebo distribuovaných úloh na MetaCentru.

7.2 Maple v MetaCentru[22]

Maple je systém počítačové algebry pro výuku a využití matematiky v přírodovědných, technických a ekonomických oborech. Slouží pro symbolické výpočty, řešení vědeckých a inženýrských problémů, matematické zkoumání, vizualizaci dat a tvorbu technických publikací. V dnešní době je verze Maple 17.

Na MetaCentru je v současnosti dostupná verze Maple 16 (umí využít pouze více vláken, tj. více procesorů na jednom stroji). Při požadavku na úlohu je nutné předat plánovači informaci o tom, jakou licenci chce uživatel používat (úloha bude spuštěna až poté, kdy bude nějaká licence opravdu dostupná). Toto se provádí zadáním vlastnosti Maple tak, jak je znázorněno na následujících příkladech.

Interaktivní použití

Požádání plánovacího systému o interaktivní režim a výpočetní zdroje.

 # pozadavek na 1 licenci a 1 uzel s X vypocetnimi jadry
qsub -I -l nodes=1:ppn=X:... -l maple=1

 # inicializace prostredi
module add maple

 # prikaz pro interaktivni textove rozhrani
maple nebo

 # prikaz pro interaktivni graficke rozhrani
xmaple

V případě interaktivního grafického rozhraní je zapotřebí mít spojení na X-server, které lze zajistit postupem uvedeným v hesle X-Window (https://wiki.metacentrum.cz/wiki/X-Window).

Dávkové použití

Ukázka spouštěcího skriptu:

#Jako interpreter prikazu se pouzije Bash (UNIXovy program)
#!/bin/sh

 # inicializace systemu modulu
. /packages/run/modules-2.0/init/sh

 # inicializace modulu Maple
module add maple

 # predani vstupnich dat programu Maple
maple < mymaplefile.{txt,mpl}

Poté zadáním požadavku plánovacímu systému:
 # pozadavek na 1 licenci a 1 uzel s X vypocetnimi jadry
qsub -l nodes=1:ppn=X:... -l maple=1 skript.sh

Integrace Maple s MATLABem

Instalovaný Maple Toolbox for MATLAB umožňuje vzájemnou integraci prostředí Maple s prostředím MATLABu. Toolbox tak kombinuje symbolické výpočty v Maple s numerickými v MATLABu, které se dají společně s výhodou použít pro velmi složité matematické analýzy výsledků. Více informací na stránkách MetaCentra (https://wiki.metacentrum.cz/wiki/Maple).

Dokumentace je dostupná na stránkách výrobce Maple (www.maplesoft.com) v angličtině a v českém jazyce jsou dostupné na http://www.maplesoft.cz/navody-publikace.

7.3 Ansys CFD v MetaCentru[22]

Ansys CFD je produkt společnosti Ansys kombinující dva základní nástroje určené pro modelování a simulaci toků/proudění různých veličin – Ansys Fluent a Ansys CFX.

- Ansys Fluent je program obsahující fyzikální modely postihující široké možnosti potřebné k modelování proudění, turbulence, přenosu tepla a reakcí pro průmyslové aplikace. Ty sahají od proudění vzduchu kolem leteckých profilů po spalování v pecích, od modelování probublávání po ropné plošiny, od toku krve po výrobu polovodičů a od návrhu ventilace místností po úpravu a čištění vody;
- Program Ansys CFX představuje komplexní program pro simulaci problémů spojených s prouděním tekutin (vícefázových, reagujících) se zahrnutím vlivu tepla (vedení, konvekce, radiace).

Oba integrované nástroje dokáží při výpočtu úlohy využít maximálně čtyřy procesory lokálního stroje a pro výpočty náročnějších úloh v rámci víceprocesorového/víceuzlového prostředí (např. gridů) je nutné využití doplňkového nástroje - Ansys HPC - jehož každá licence umožňuje distribuci výpočtu na další dostupný (lokální nebo vzdálený) procesor.

Ansys CFD (Ansys Fluent + Ansys CFX) ve verzi 14.5 v množství 25 souběžných spuštění a modul je dostupný pomocí příkazu ***module add ansys-14***. Ansys Fluent lze spouštět jak v interaktivním režimu, tak i v dávkovém. Pro názornost budou uvedeny oba způsoby. Pozor, pro spuštění uživatel musí využívat pouze stroje s OS Debian 6.0 a novější (na Debian 5.0 produkty Ansys nefungují).

Interaktivní režim

Například pomocí programu PuTTY se uživatel přihlásí na čelní uzel, kde může požádat o režim a) textový nebo b) grafický, analogie MATLABovského prostředí.

a) textový
qsub -I -l nodes=X:ppn=Y:debian60 -l mem=Zgb
fluent -g

b) grafický
qsub -I -X -l nodes=X:ppn=Y:debian60 -l mem=Zgb
fluent

Dávkový režim

Taktéž pomocí programu PuTTY se uživatel přihlásí na čelní uzel s tím, že má možnost provádět a) sériový nebo b) paralelní/distribuovaný výpočet.

a) sériový

 # inicializace prostredi

module add ansys-14

 # seriove spusteni Fluentu

fluent <version> -g -i input_file

b) paralelní/distribuovaný

 # inicializace prostredi

module add ansys-14

 # paralelni/distribuovane spusteni Fluentu

fluent <version> -t${cpus} -p -cnf=$PBS_NODEFILE -g < flow.input

Takto vytvořený skript úlohy se předá plánovači spolu s požadavkem na počet uzlů/procesorů a velikost potřebné paměti:

qsub -l nodes=X:ppn=Y:debian60 -l mem=Zgb popisnyskript.sh

Podobným způsobem lze pracovat i s programem Ansys CFX, viz základní dokumentace na MetaCentru (lokálně v adresáři programu /software/ansys-14/doc/readme.html) nebo spíše na stránkách výrobce programu, avšak pouze pro registrované uživatele.

8. Plánovací systém

Každý uživatel si musí uvědomit, že není jediný, který využívá prostředky (procesory, paměť, prostor na disku, softwarové licence apod.) MetaCentra. Efektivní hospodaření s těmito prostředky má na starosti plánovací systém Torque, dříve PBS (Portable Batch System). Na začátku budou uvedeny některé pojmy jako výpočetní zdroje, operační systémy, příkazové řádky, skripty apod.

Prvním krokem je podání přihlášky pro vstup do MetaCentra a seznámení se s pravidly využívání služeb MetaCentra. Standardní účet v MetaCentru je zřizován na kalendářní rok, obvykle expiruje k 1. 1. Jedenkrát ročně, obvykle na konci kalendářního roku, uživatel žádá o prodloužení svého účtu na další rok, při kterém dokládá, k jakému účelu zdroje MetaCentra použil a jakých výsledků ve svém oboru dosáhl a vloží seznam svých publikací za uplynulý rok. Při neprodloužení účtu na konci roku jsou účtu odebrána práva až do jeho opětovné reaktivace. Čtyři neprodloužení v řadě za sebou má za následek smazání uživatelského účtu a nemožnost dostání se k případně uloženým datům [23].

Prvním pojmem je úloha, která představuje rezervaci prostředků, tj. procesory, paměť, prostor na disku, softwarové licence, které uživatel bude využívat po určený čas. Plánovací systém tyto úlohy eviduje a snaží se jim co nejefektivněji vyhovět podle vnitřních algoritmů, které hlídají veškeré dostupné výpočetní zdroje. Úlohy se mohou zadávat interaktivně, tzn. příkazy zadávat postupně do příkazové řádky, nebo pomocí dávky ve formě připraveného skriptu, více v praktické části.

Úlohy čekající na spuštění jsou v tzv. frontách (obrázek 23), kterých je mnoho a liší se svojí prioritou, množinou výpočetních uzlů, maximálním časem úloh, počtem zároveň spuštěných úloh a přístupovými právy uživatelů. Úlohy si po spuštění pamatují, ze které fronty byly spuštěny, protože to ovlivňuje, kolik času má spuštěná úloha k dispozici a kolik úloh zároveň bude spuštěno jednomu uživateli. Z hlediska časové náročnosti se úlohy přiřazují do následujících obvyklých typů front:

- short úloze stačí méně než 2 hodiny času;
- normal úlohy běží nanejvýš 24 hodin;
- long více jak 24 hodin až 720 hodin (30 dní);
- backfill velké množství úloh, až tisíce na jednoho uživatele.

Pak existuje mnoho dalších front s různým účelem, většinou jsou to fronty s vyšší prioritou vyhrazené vlastníkům určitých strojů, aby na svých strojích měli přednost před ostatními uživateli. Uživatel má možnost použít i běžně nepřístupné zdroje, které se dají využívat pomocí fronty preemptible s tím, že musí počítat s možným dočasným zastavením svých úloh [23].

Server arien.ics.muni.cz - Produkční prostředí

fronta	priorita	časové limity	nutná vlastnost	úloh ve frontě	běžících/max	hotových	celkem	max. na uživatele
private	99	0 - 0		0	0 /	0	0	
monitoring	99	0 - 0		3	0 /	0	3	
maintenance	99	0 - 0		0	0 / 1000	34	34	1000
priority	91	0 - 24:00:00		0	0 /	0	0	
reserved	90	0 - 0		0	24 / 1000	2	26	1000
globus	80	0 - 720:00:00	globus	0	0 / 4	0	0	4
pa177	80	0 - 24:00:00	pa177	0	0 / 4	0	0	15
xentest	80	0 - 720:00:00		0	0 / 4	0	0	4
jcu2	71	0 - 720:00:00	q_jcu2	0	0 /	0	0	
orca16g	71	0 - 720:00:00	orca16g	0	0 / 120	0	0	80
ncbr	70	0 - 720:00:00	q_ncbr	0	0 / 1200	0	0	32
quark	70	0 - 720:00:00	q_quark	0	2 /	366	371	
ncbr_long	70	0 - 720:00:00	q_ncbr_long	0	17 / 20	7	24	5
interactive	70	0 - 04:00:00	q_normal	0	0 / 6	0	0	6
jcu	70	0 - 720:00:00	jcu	0	0 /	0	0	
feec	70	0 - 720:00:00	q_feec	0	0 /	0	0	
loslab	70	0 - 720:00:00	q_loslab	0	0 /	0	0	
zsc	70	0 - 720:00:00	zsc	0	0 /	0	0	
orca	70	0 - 720:00:00	orca	0	0 / 120	0	0	80
mufin	70	0 - 720:00:00	q_mufin	0	0 /	0	0	
iti	70	0 - 720:00:00	q_iti	0	0 / 500	0	0	96
mikroskop	69	0 - 720:00:00	q_quark	0	0 /	43	43	
gpu	65	0 - 24:00:00	q_gpu	0	10 / 20	64	74	
ncbr_medium	65	0 - 120:00:00	q_ncbr_medium	2	27 / 1000	70	99	32
privileged	65	0 - 720:00:00	q_privileged	0	31 / 1000	16	54	50
ncbr_single	64	0 - 48:00:00	q_ncbr_single	0	62 / 1000	87	150	200
long	62	24:00:01 - 720:00:00	q_long	115	149 / 1000	124	389	50
preemptible	61	00:00:00 - 720:00:00	q_preemptible	7	55 /	174	236	400
short	60	0 - 02:00:00	q_short	3	67 / 1000	454	528	250
gpu_long	55	0 - 168:00:00	q_gpu_long	0	0 / 20	0	0	
debian6_long	51	02:00:01 - 720:00:00	q_debian6_long	44	2 / 1000	13	60	50
normal	50	02:00:01 - 24:00:00	q_normal	118	26 / 1000	99	256	100
debian6	50	02:00:01 - 24:00:00	q_debian6	0	0 / 1000	0	0	50
backfill	20	00:00:01 - 24:00:00	q_backfill	11	328 / 2000	1498	1840	1000
MetaSeminar	0	0 - 0	q_metaseminar	0	0 /	0	0	
default	0	0 - 0		0	0 /	0	0	

Obr. 23 Fronty - server arien.ics.muni.cz - produkční prostředí, převzato a upraveno z [24]

V prvním sloupci je název fronty a kliknutím na její název lze zjistit, k jakému účelu je fronta určena. Po kliknutí na sloupec priorita lze získat informaci o tom, kteří uživatelé, skupiny uživatelů patří do fronty. Plánovací systém umožňuje vyhradit některé uzly pro uživatele, skupiny uživatelů, kteří mají určité výhody, např. to, že skupina má vyhrazené právo spouštět úlohy ve stanovený čas.

Sloupec priorita představuje číselnou hodnotu, resp. čím větší číselná hodnota, tím dřívější běh úlohy. Každý uživatel si stanovuje priority úloh sám, pokud má více úloh, tak si může tyto priority stanovit dle jejich důležitosti.

Časové limity znamenají minimální a maximální dobu běhu úlohy. Je na uživateli, aby si vybral frontu podle toho, jak dlouho předpokládá, že bude trvat její zpracování. Po překročení časového limitu je úloha plánovačem ukončena. Sloupec nutná vlastnost představuje identifikátor o tom, že pro tento stroj se požadují určité vlastnosti, aby se mohly spouštět úlohy do ní zařazené [23, 24].

Sloupec úloha je rozdělen na dílčí sloupce, všechny svými informacemi vztažené k úlohám zařazeným ve frontě (kolik jich je právě ve frontě, kolik jich je běžících a kolik jich může být maximálně, kolik je z nich hotových, kolik je jich celkem, kolik jich může maximálně zadat jeden uživatel).

Plánovací systém je zodpovědný za zařazení uživatelových úloh do příslušných front a také za životní cyklus těchto úloh. **Fronty, resp. priority front** (Queue priority, By Queue) jsou tedy jedním z plánovacích mechanismů, který obsahuje ještě další plánovací mechanismus, viz dále.

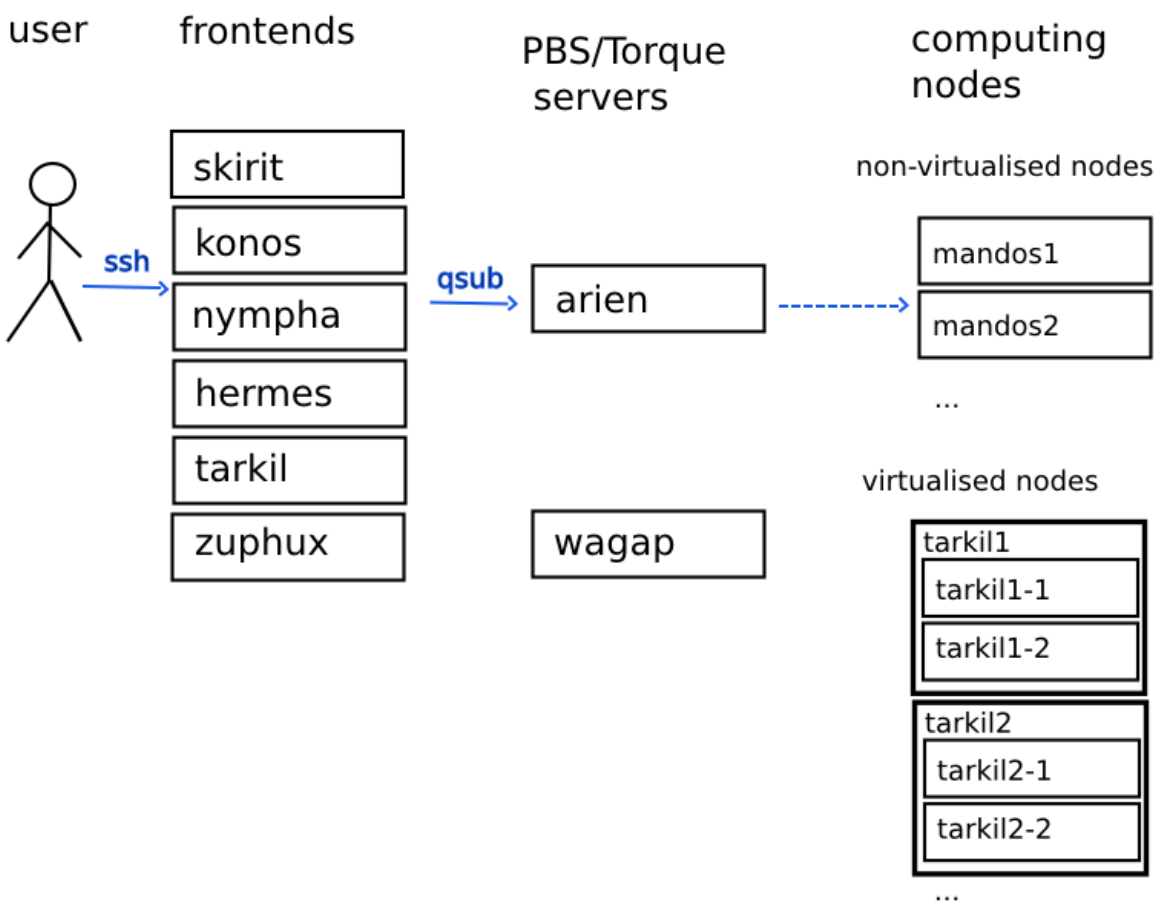
Systém kontroluje všechny zdroje a férovost vytížení výpočetních zdrojů uživatelem při každém průchodu plánovacího cyklu, který se nazývá **slábnoucí Fair-share algoritmus** (Decaying FS algorithm). Algoritmus Fair-share je označen jako slábnoucí, protože při pohledu na uživatelem zpracovávané úlohy na časové ose, mají starší úlohy menší váhu než novější úlohy. Váha úloh slábne s přibývajícím časem a pro Fair-share se nastavuje tzv. half-time, což je doba, po které má propočítaná úloha poloviční váhu než novější úlohy stejného uživatele. Tento princip se používá na MetaCentru s tím, že se neaplikují počítadla na skupiny uživatelů, tzn. bez využití hierarchického Fair-share stromu (Hierarchical Fair-share Tree). Každý uživatel má své vlastní počítadlo určující, kolik výpočetních zdrojů za poslední období využil. Half-time je nastaven na 194 hodin. To je čas, po kterém se váha uživatelem využitých zdrojů sníží zhruba na polovinu. Pokud jsou ve stejné frontě dvě úlohy se stejným Fair-share, je upřednostněna ta úloha, která má od uživatele nastavený kratší odhad času běhu (pokud tento údaj úloha zadán nemá, bere se v potaz maximální možná doba běhu v příslušné frontě) [25].

Dále je důležité zmínit, že v MetaCentru uživatelé, kteří v posledním měsíci hodně počítali, dostávají nižší prioritu a zvýhodněni jsou ti uživatelé, kteří vykázali určitý počet publikací nebo přispěli do ročenky MetaCentra. Určitý problém představují paralelní úlohy, které požadují rozličná množství procesorů (a ještě kombinované s jednoprocesorovými úlohami). Aby se dostaly k výpočetním zdrojům, plánovací systém je nastaven tak, že čeká-li paralelní úloha již příliš dlouho, plánovač pro ni rezervuje zdroje a na nich běžící úlohy nechá dokončit, nicméně nespouští tam další menší úlohy. Proto uživatel občas uvidí volné zdroje a je překvapen, že se zrovna jeho úloha nespustila [23, 25].

Životní cyklus úlohy:

- *zadaná*: byla zadána příkazem ***qsub*** z libovolného frontendu, plánovací systém ji zařadí do fronty, výkonný kód úlohy je popsán ve formě shell skriptu;
- *ve frontě*: plánovač se pokouší úlohu naplánovat;
- *stage-in*: příprava na spuštění, je vybráno, na jakých strojích se úloha spustí, plánovač spouští kopírování vstupních dat na tyto uzly;
- *běžící*: výkonný kód je vykonáván;
- *končící*: na konci výpočtu by úloha měla zkopírovat výsledky na datové úložiště a smazat zbylé dočasné soubory na výpočetním uzlu, pokud není řečeno jinak, standardní a chybový výstup spouštěného programu budou zkopírovány na stroj, odkud byla úloha zadána;
- *ukončená*: informace o úloze bude ještě několik hodin vidět ve výpisu příkazu ***qstat***.

8.1 Spouštění úloh[23]



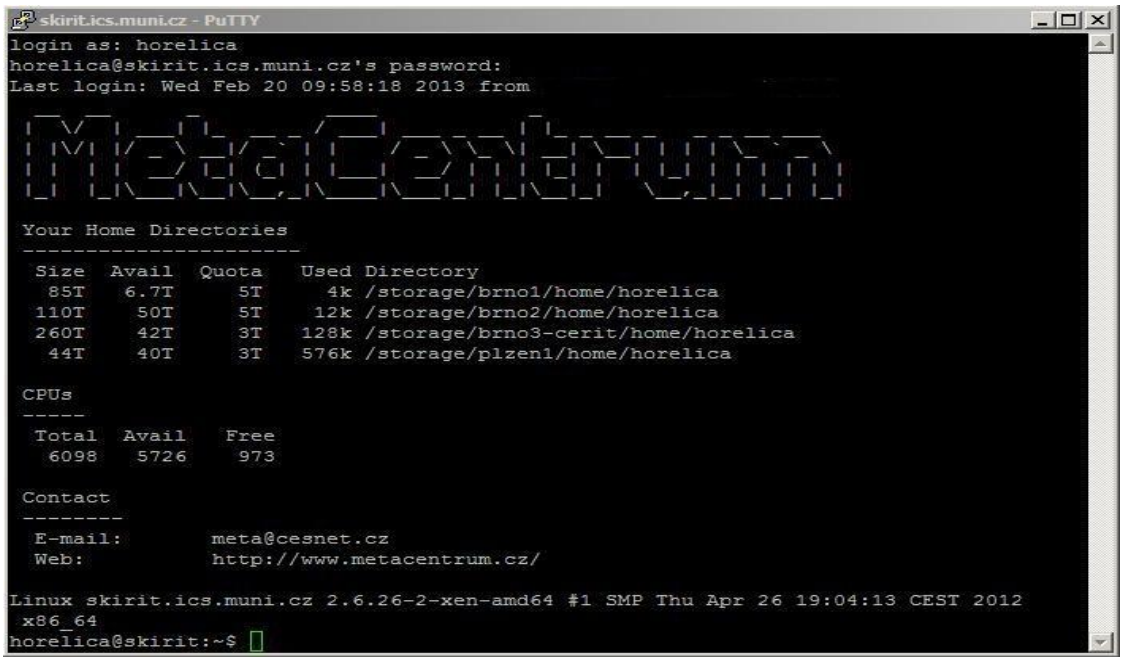
Obr. 24 Schéma fungování MetaCentra VO, převzato a upraveno z [23]

Obrázek 24 ilustruje schéma fungování MetaCentra VO. K tomu, aby mohl uživatel zadávat a spouštět úlohy na MetaCentru, je nutné přihlášení na některý čelní uzel neboli frontend (např. skirit.ics.muni.cz, arien.ics.muni.cz, nympha.zcu.cz, hermes.metacentrum.cz, tarkil.cesnet.cz, minos.zcu.cz a další, které je možné najít na stránkách Wiki MetaCentra) pomocí nejpoužívanějšího programu PuTTY, který simuluje textový terminál běžící v prostředí MS-Windows. V tomto případě bude použit program PuTTY (obrázek 25). Plánovací systém je označen zkratkou PBS/Torque servers, který přistupuje na nevirtualizované nebo virtualizované výpočetní uzly. Virtualizovaný výpočetní uzel narozdíl od nevirtualizovaného představuje např. stroj, který může mít jiný operační systém (Windows), než který je v MetaCentru běžně používán. Záleží na požadavcích uživatele.

V prostředích postavených na UNIXu se využívají zabudované terminály, kde se uživatel hlásí příkazem ***ssh username@nazev_frontendu*** a ***heslem***. Přihlašování lze také realizovat autentizačním systémem Kerberos, který využívá tzv. *lístky*, které lze přirovnat k certifikátům. Výhoda Kerberosu je v tom, že stačí jednou denně získat Kerberos lístek, který umožní po dobu 10 hodin přistupovat ke strojům MetaCentra, aniž by se muselo zadávat heslo.

Obr. 25 Úvodní okno programu PuTTY

Na obrázku 25 je zvýrazněná část pro zadávání adresy frontendu skirit.ics.muni.cz v programu PuTTY. Toto spojení si je možné uložit pro příští přístup. Port spojení je 22, tzn., že toto spojení mezi počítačem uživatele a serverem je šifrováno. Po úspěšném přihlášení se objeví terminálové okno připravené pro zadávání příkazů, obrázek 26.



Obr. 26 Terminálové okno programu PuTTY po přihlášení na frontend skirit.ics.muni.cz

Ještě než začne uživatel zadávat úlohy, měl by si stanovit požadavky na stroje pro výpočet. Ty nejdůležitější jsou popsány níže.

8.2 Požadavky na výpočetní zdroje

- *Geografické umístění* - mající vliv na zpoždění (latenci) sítě. Klient musí při určitých operacích čekat na potvrzení, což se projeví zejména při výpočtech, které zakládají mnoho malých souborů, jako je například rozbalení archívu nebo kompilace jádra systému apod., více na https://wiki.metacentrum.cz, část rychlosti při použití souborových systémů. Umístění např. Brno, Praha, Plzeň, České Budějovice.
- *Umístění v clusteru* - umožňuje spustit úlohu v rámci jednoho clusteru, např. cl_mandos - stroje clusteru mandos a naopak vyloučí spuštění úlohy na clusteru mandos ^cl_mandos.
- *Dosah /home adresáře* - vlastnost home_cluster říká, že na stroji s touto vlastností je sdílený */home* s daným clusterem, např.: home_skirit - na uzlu s touto vlastností je sdílený */home* se strojem skirit.
- *Typ CPU* - udává instrukční sadu procesorů: amd64, x86, xeon, x86_64.
- *Počet CPU* - je vhodné v případě, že uživatel potřebuje vyhradit celý stroj: nodecpus1, nodecpus2, nodecpus4, nodecpus8, nodecpus16, nodecpus32, tj. počet CPU na uzlu (1, 2, 4, 8, 16, 32). Najde stroj právě s požadovaným počtem procesorů.
- *Operační systém* - stroje s operačním systémem Linux (Debian, Debian verze 5.0/6.0).
- *Nodes* - počet nebo typ výpočetních uzlů/strojů, které se mají rezervovat pro úlohu.
- *Walltime* - maximální množství skutečného času, po který může být úloha ve stavu běžící.
- *Cput* - maximální množství času CPU spotřebovaného všemi procesy náležejícími k úloze.
- *Fronta* - viz kapitola 8.
- *Paměť* - stanovení požadavků na paměť se zadává parametrem *mem*. Jeho pomocí se zarezervuje a zároveň omezí určité množství paměti pro každý uzel požadovaný v zadání úlohy (node). Plánovač nikdy nepoužije jeden fyzický stroj jako více uzlů ze zadání úlohy, tj. i pokud bude k dispozici stroj se čtyřmi nebo více procesory, tato úloha na něm dostane přiděleny jen dva procesory. Nebo jinak řečeno, když uživatel požaduje *nodes=N*, pak dostane skutečně *N* různých strojů, i když by bylo možné dosáhnout stejného množství procesorů s menším počtem strojů. Bohužel ani použití *mem* nedává skutečnou záruku, že úloha dostane od operačního systému celou požadovanou paměť, systém může spotřebovat nějakou paměť na režijní účely. Proto je dobré žádat o trochu více paměti, než úloha potřebuje, ale zase ne o moc, aby nedošlo k omezení počtu strojů, na které může být úloha naplánována. Pokud parametr *mem* není zadán, tak se použije standardní hodnota, která je poměrně nízká, a bude úlohu zbytečně omezovat. Proto je vhodné parametr vždy zadávat.
- *Licence pro software* - např. MATLAB, Fluent, Ansys atd.

V MetaCentru je základním příkazem ***qsub*** (submit a job), který se dále kombinuje s dalšími parametry, viz dále. Vlastnosti zdrojů se zadávají s volbou ***-l*** a jsou formulovány čárkami oddělenými řetězci. Tedy příkaz by měl následující syntaxi: ***qsub -l***.

Uživatel by měl znát některé další příkazy, například jak se úloha zruší, provede výpis stavu uzlů atd., viz tabulka 2.

Tabulka 2 Seznam základních příkazů PBS, převzato a upraveno z [26]

Syntaxe	Popis
qsub	Zadání úlohy do fronty
qdel	Zrušení čekající nebo běžící úlohy
qfree	Aktuální stav uzlů a jejich vlastnosti
xpbs	Grafický přehled o PBS serverech, frontách a úlohách
xpbsmon	Grafické znázornění vytížení jednotlivých uzlů v PBS
qstat	Výpis všech úloh
qstat -q	Výpis všech úloh běžících na systému
qstat -r	Výpis všech běžících úloh
qstat -f [cislo_ulohy]	Vypíše aktuální stav úlohy
qstat -u [user]	Vypíše všechny úlohy uživatele
qsub -?	Nápověda

Jak již bylo zmíněno, tak jednotlivé příkazy lze zadávat buď interaktivně, nebo pomocí připravených skriptů spolu s dalšími soubory. Interaktivní úlohy se zadávají s přepínačem velké **I** (**I**nteractive) a příkaz má tvar:
qsub -I

Uživateli se přidělí náhodně stroj a s příkazovým řádkem může operovat. Ukončení je příkazem ***exit***, čímž se uživatel dostane zpět na příkazový řádek čelního uzlu. Pokud zadá příkaz ***qsub*** bez dalších parametrů, bude mu přidělen jeden procesor a 400 MB paměti na 24 hodin. Pozor, úloha nemusí být spuštěna na výpočetním uzlu stejného clusteru, z jehož čelního uzlu byla zadána. Obecně může být úloha spuštěna na libovolném výpočetním uzlu libovolného clusteru, případně na některém samostatném stroji, který není v žádném clusteru. Při spuštění úlohy je na prvním z přidělených strojů spuštěn shell, buď interaktivně u interaktivní úlohy, nebo dávkově se zadaným shellovým skriptem. Úloha končí sama, pokud je ukončen tento shell, nebo je ukončena násilně plánovačem, pokud překročí přidělený čas.

V MetaCentru je možné díky systému X-Window přenášet grafické rozhraní aplikací (např. MATLAB, Maple) přes síť. Takže uživatel má dojem, jako by aplikace běžela na jeho počítači. Příkaz má pak tvar: ***qsub -X –I***.

8.3 Příklady práce v interaktivním režimu[12]

Pokud by uživatel chtěl použít interaktivní práci s grafickým prostředím, frontu typu short (max. 2 hodiny), jeden uzel, tak příkaz by měl tvar:
qsub -X -I -q short -l nodes=1

Exkluzivní rezervace slouží k tomu, když uživatel neví, kolik procesorů může úloha zabírat. Je vhodné používat v kombinaci s vlastností pro počet CPU stroje:
qsub -X -I -q short -l nodes=1:ppn=2#excl

Pro určení konkrétního uzlu, např. v Brně, který se má použít, bude příkaz:
qsub -X -I -q short -l nodes=1:ppn=2:brno

Pro určení typu procesou, který se má použít, se definuje příkaz:
qsub -X -I -q short -l nodes=1:ppn=2:brno:amd64

Rozšíření o potřebné množství požadované paměti bude definováno příkazem:
qsub -X -I -q short -l nodes=1:ppn=2:brno:amd64,mem=1gb

Pokud bude uživatel používat konkrétní aplikaci (MATLAB, Fluent apod.), musí uvést typ licence. Kolik licencí je celkem se dá dočíst na stránkách příslušných aplikačních programů. Kolik licencí je volných se dá zjistit příkazem ***qfree -i***. K dispozici jsou tyto licence:

- Matematické a statistické modelování - MATLAB, Maple.
- Výpočetní chemie / Molekulové modelování - Amber, Babel, Gaussian/GaussView.
- Strukturní biologie, biologie a bioinformatika - MrBayes, QUEEN, X-PLOR.
- Technické a materiálové simulace - ANSYS (včetně modulu LS-DYNA), Fluent, MSC. Marc.
- Vývojářské nástroje a prostředí - Vývojové prostředí SGI, PGI CDK, SICStus Prolog.

Kompletní přehled všech aktuálních aplikačních programů na MetaCentru. V následujícím příkladu je uveden MATLAB s jednou licencí:
qsub -I -q short -l nodes=1:ppn=2:amd64,mem=1gb,matlab=1

Doposud bylo v příkazu používáno několik parametů ***-l***:
qsub -I nodes=1:ppn=2:amd64 -l mem=1gb -l matlab=1

Nebo lze příkaz zjednodušit pomocí čárky:
qsub -I nodes=1:ppn=2:amd64,mem=1gb,matlab=1

Dalším případem může být to, že uživatel přesně ví, jaké maximální množství skutečného času může být úloha ve stavu běžící, tzn. nemusí použít typ fronty, ale může zadat maximální dobu parametrem ***walltime [HH:MM:SS]***, tzn.:

qsub -I walltime=10:00:00 -l nodes=1:ppn=2:brno:amd64

Uživatel by měl umět pracovat s nápovědou, kterou poskytuje plánovací systém. Zobrazení je příkazem ***man pbs_resources***.

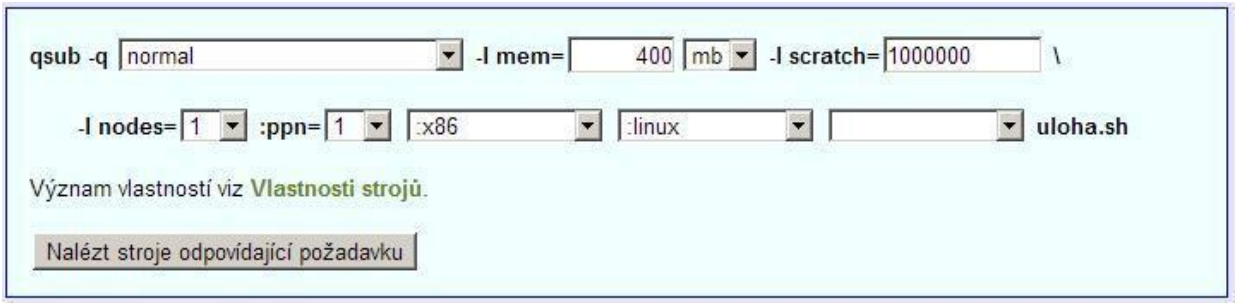
9. Spouštění MATLABovských úloh na MetaCentru

Jak již bylo zmíněno v kapitole 8.2, na začátku si uživatel musí stanovit požadavky pro výpočet. Pro oživení je uvedena stručná forma.

- *Fronta,* parametr ***-q*** + typ fronty:
  - ***short*** - do 2 hodin,
  - ***normal*** - do 24 hodin,
  - ***long*** - více jak 24 hodin až 720 hodin (30 dní).

  Pokud přesně uživatel ví, jak dlouho bude úloha trvat: ***-q normal -l walltime=5:00:00***, ***l walltime=hodiny:minuty:sekundy***.
- *Paměť* - parametr ***mem=***číslo [MB, GB].
- *Nodes* - počet a možný typ výpočetních strojů (uzlů) s počtem procesorů, které se mají rezervovat pro úlohu, parametr ***-l nodes=***číslo***:ppn=***číslo.
  - ***-l nodes=1:ppn=1*** – jeden procesor na jednom stroji,
  - ***-l nodes=10:ppn=2*** – deset dvouprocesorových strojů,
  - uživatel neví kolik jich potřebuje, tedy zadá příkaz: ***-l nodes=1:ppn=2:nodecpus4#excl***,
  - stroj v Brně s procesorem AMD Opteron:***-l nodes=1:ppn=1:brno:amd64***.
- *Licence* - počet licencí programu, parameter ***-l nazev_licence=***počet, např. uživatel chce použít jednu licenci MATLABu: ***-l nodes=1:ppn=1 -l mem=500mb -l matlab=1***.
- Velikost místa na rychlém úložišti pracovních dat, parametr ***-l scratch***=*číslo*, kde *číslo* je počet kibibajtů (1 KiB = 1 024 B). Například uživatel vyžaduje v úloze 1 MiB místa: ***-l scratch=1mb***.

Novému neznalému uživateli je na MetaCentru umožněno využít tzv. *„Sestavovač příkazů qsub“*, který je na webu MetaCentra v menu zdroje, osobní pohled, který poskytuje pomoc při sestavování příkazu, obrázek 27. Pokud uživatel zadá požadavky, tak tento sestavovač nalezne navíc i volné stroje, které jsou mu ihned k dispozici a nemusí vyčkávat, až tyto stroje budou volné, obrázek 28.



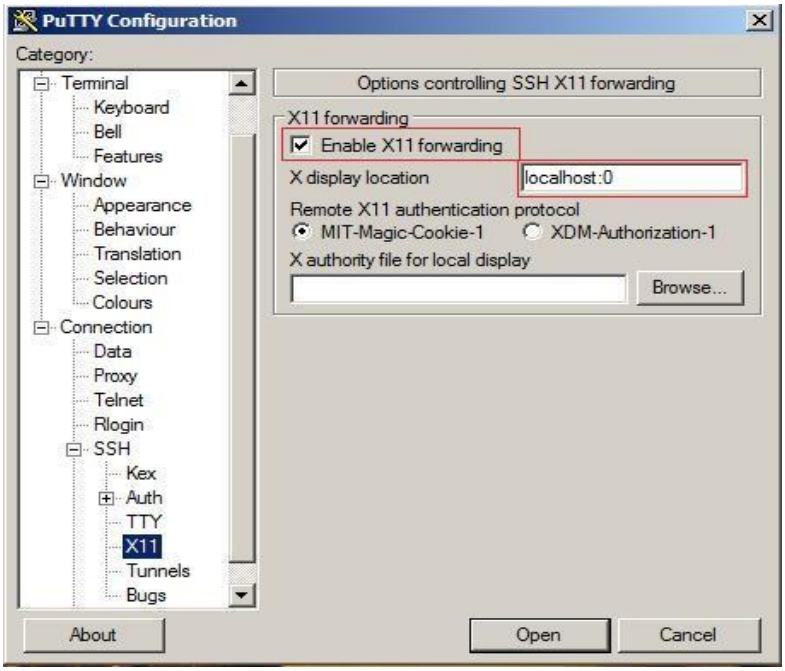
Obr. 27 Sestavovač příkazů qsub, převzato a upraveno z [21]

Obr. 28 Výsledek sestavovače příkazů qsub, převzato a upraveno z [21]

9.1 Interaktivní spouštění[12]

a) V grafickém režimu

Pokud uživatel používá na svém počítači MS-Windows, potřebuje nějaký X-server pro MS-Windows, např. Xming, Cygwin/X, X-Win32, ReflectionX nebo Exceed. Pro SSH s podporou tunelování X-protokolu lze použít PuTTY nebo Cygwin. V tomto případě je PuTTY s nastavením, které je na obrázku 29 [12].



Obr. 29 Nastavení X-Window v PuTTY

Po nastavení se uživatel přihlásí pomocí PuTTY na frontend MetaCentra, požádá o prostředky příkazem ***qsub -I -q short -l nodes=1:ppn=1:amd64,mem=2gb,matlab=1***, obrázek 30. Poté má tyto prostředky zarezervované a spustí si MATLAB v interaktivním režimu s GUI, obrázek 30.

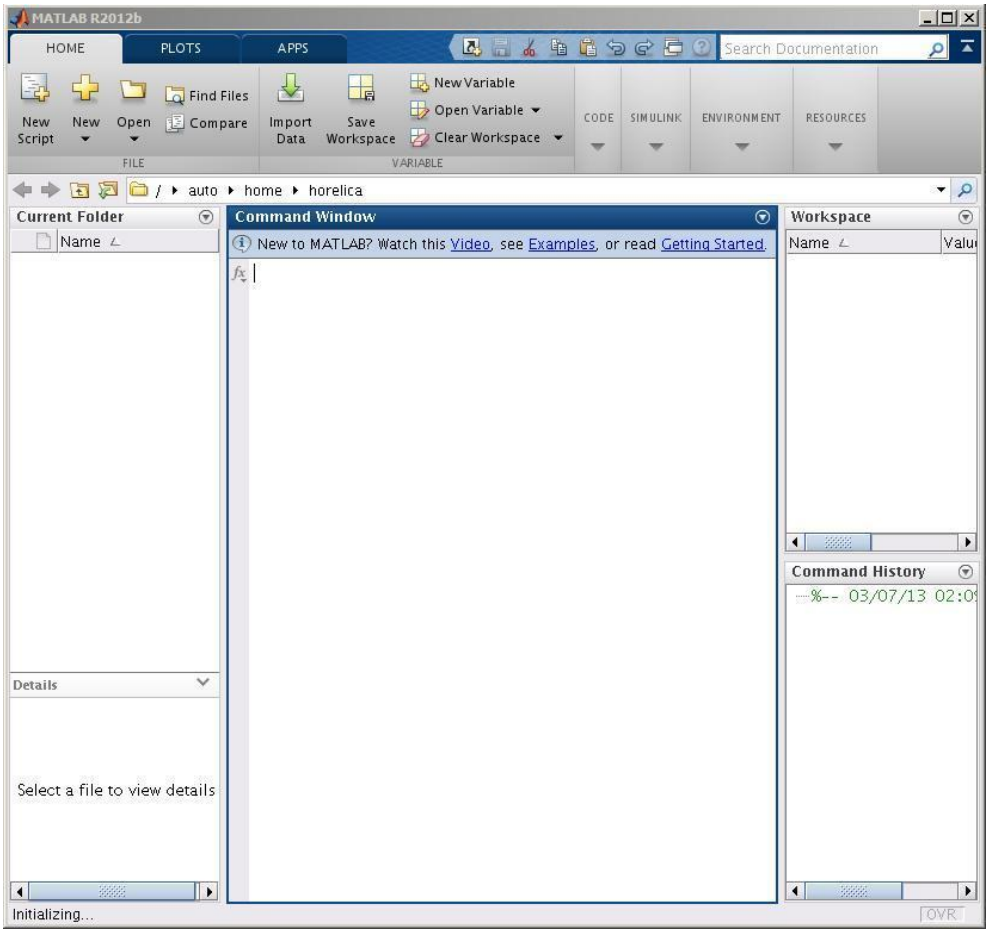
Pokud uživatel nezadá v příkazu volbu ***-singleCompThread***, MATLAB bude využívat při výpočtu všechny dostupné CPU, proto je nutné požádat o přidělení celého stroje, jinak bude úloha využívat více CPU, než má zarezervováno.

Přidělení celého stroje pomocí parametru ***#excl***, resp.:
qsub -I -q short -l nodes=1:ppn=1#excl:amd64,mem=2gb,matlab=1

```
horelica@skirit:~$ qsub -I -q short -l nodes=1:ppn=1#excl:amd64,mem=2gb,matlab=1
qsub: waiting for job 2184716.arien.ics.muni.cz to start
qsub: job  2184398 arien.ics.muni.cz ready

horelica@apollo2:~$ module add matlab
horelica@apollo2:~$ matlab
```

Obr. 30 Žádost o prostředky a modul MATLAB v PuTTY

Po uplynutí času (záleží na rychlosti linky, HW počítače) se objeví GUI MATLABu připravené pro práci, obrázek 31.

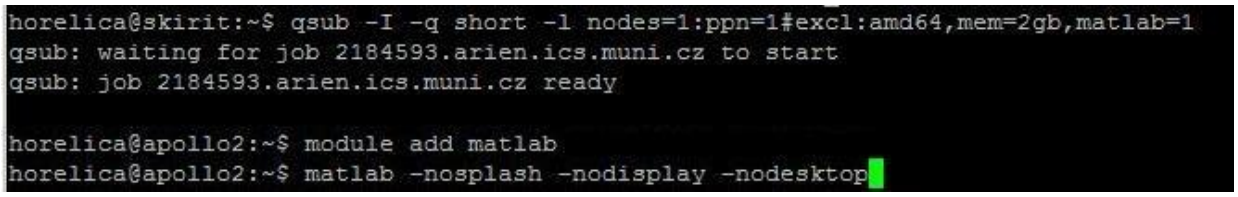


Obr. 31 Připravené prostředí GUI MATLABu

Prostředí MATLAB lze ukončit uzavřením GUI, spojení bude ukončeno a terminálové okno bude připravené k novému zadávání. Spouštění úloh v grafickém režimu je spíše jednou z možností, kterou lze využívat, avšak záleží na uživateli, jestli se pro něj rozhodne. Interaktivní spuštění lze použít pro vývoj nové aplikace a po dokončení vývoje ji spustit dávkově, viz dávkové spouštění úloh. Interaktivní způsob spuštění pravděpodobně nepřinese zásadní zvýšení výkonu oproti spuštění MATLABu přímo na počítači uživatele, dokud se nepoužije paralelizace přes více strojů. Přesněji řečeno, pokud má uživatel osobní počítač s jedním nebo dvěma procesory a spustí MATLAB v MetaCentru na mnohaprocesorovém stroji, zrychlení bude citelné, protože MATLAB využívá při maticových operacích všechny dostupné procesory. Ale ještě mnohem většího zrychlení lze dosáhnou spuštěním více dávkových úloh vedle sebe [12].

b) V textovém režimu

MATLAB lze spustit i bez GUI, resp. jen v textovém režimu, ve kterém není potřeba spouštět X-server. Nejdříve opět musí uživatel požádat o prostředky (obrázek 32), po přidělení musí požádat o modul MATLABu (obrázek 32) a nakonec se zobrazí terminálové okno MATLABu, obrázek 33. Ukončení terminálového okna MATLABu lze provést příkazem ***exit***.



Obr. 32 Žádost o softwarový modul MATLAB v PuTTY

Obr. 33 Terminál MATLABu, připravený k zadávání příkazů

9.2 Dávkové spouštění[12]

MATLAB lze spouštět i bez grafického rozhraní, a to v dávkovém režimu. K tomu slouží přepínače ***-nodisplay -r***. Dávkové spuštění je výhodné, pokud chce uživatel spustit více úloh najednou, nebo si nechce blokovat svůj stroj dlouho běžící úlohou. Uživatel si vytvoří soubor s příponou **.sh*, např. v Notepadu, kde uvede seznam příkazů pro shell, je to analogie vytváření *.bat* souborů v DOSu. Skript pro shell se skládá z jednotlivých příkazů, které se píší do příkazové řádky, viz níže.

Příklad shellového souboru s názvem **pocitej.sh**

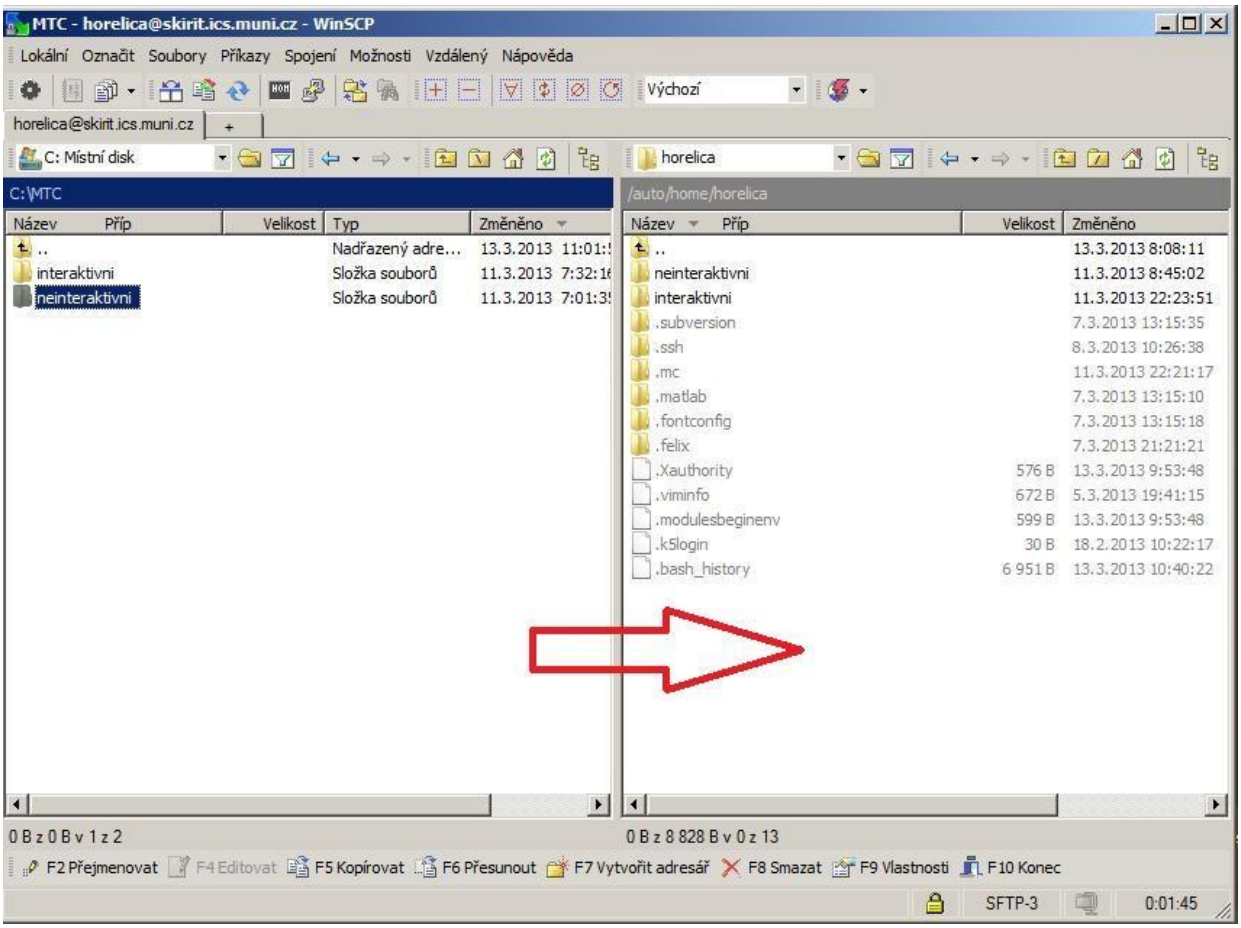
#!/bin/bash
#PBS -q short
#PBS -l nodes=1:ppn=1:#excl:amd64,mem=2gb,matlab=1
#PBS -j oe
#PBS -m n
 # Nastaveni promenne PATH, aby se nasel MATLAB
. /packages/run/modules-2.0/init/bash
module add matlab
 # MATLAB vstoupi do adresare, kde se nalezaji soubory pro vypocet
cd /scratch/horelica/neinteraktivni/
 # Spusti MATLAB bez grafického rozhrani, v davkovem rezimu, MATLABovsky soubor bez pripony
matlab -nodisplay -r "nazev_m_souboru_bez_pripony"

Řádka *#!/bin/bash* se musí psát na začátek všech skriptů, protože tím se informuje shell, že má jako interpreter spustit /bin/bash. Řádky začínající #PBS jsou příkazy pro plánovací systém, tzn. typ fronty, počet uzlů, procesorů, paměti, licence apod. Volba *-j oe* znamená, že standardní chyby úlohy budou předány do standardního výstupu (obrazovka). Volba *-m n* znamená, že nebude odeslán e-mail uživateli po skončení úlohy, v opačném případě, pokud je zadán e-mail, bude odeslán uživateli. Výčet těchto příkazů představuje základ, záleží na uživateli, jak chce možnosti dávkového spouštění využívat, pro detailnější informace lze čerpat z nápovědy PBS, kterou si uživatel vyvolá zadáním příkazu ***man pbs***.

Tabulka 3 PBS příkazy ve skriptu, převzato a upraveno z [26]

Syntaxe	**Popis**
#PBS -N jobname	Definování názvu úlohy
#PBS -M xyz@server.cz	E-mail uživatele
#PBS -m b	Odešle e-mail pokud začne úloha
#PBS -m e	Odešle e-mail pokud úloha skončí
#PBS -m a	Odešle e-mail pokud úloha skončí chybou
#PBS -o /users/username/output	Zapíše událost do souboru
#PBS -e /users/username/errors	Zapíše chybovou událost do souboru

Níže je uveden příklad s postupem. Po vytvoření souboru pro shell a ostatních MATLABovských souborů jsou nakopírovány do domovského adresáře na MetaCentru, např. pomocí WinSCP, obrázek 34.



Obr. 34 Kopírování souborů na MetaCentru

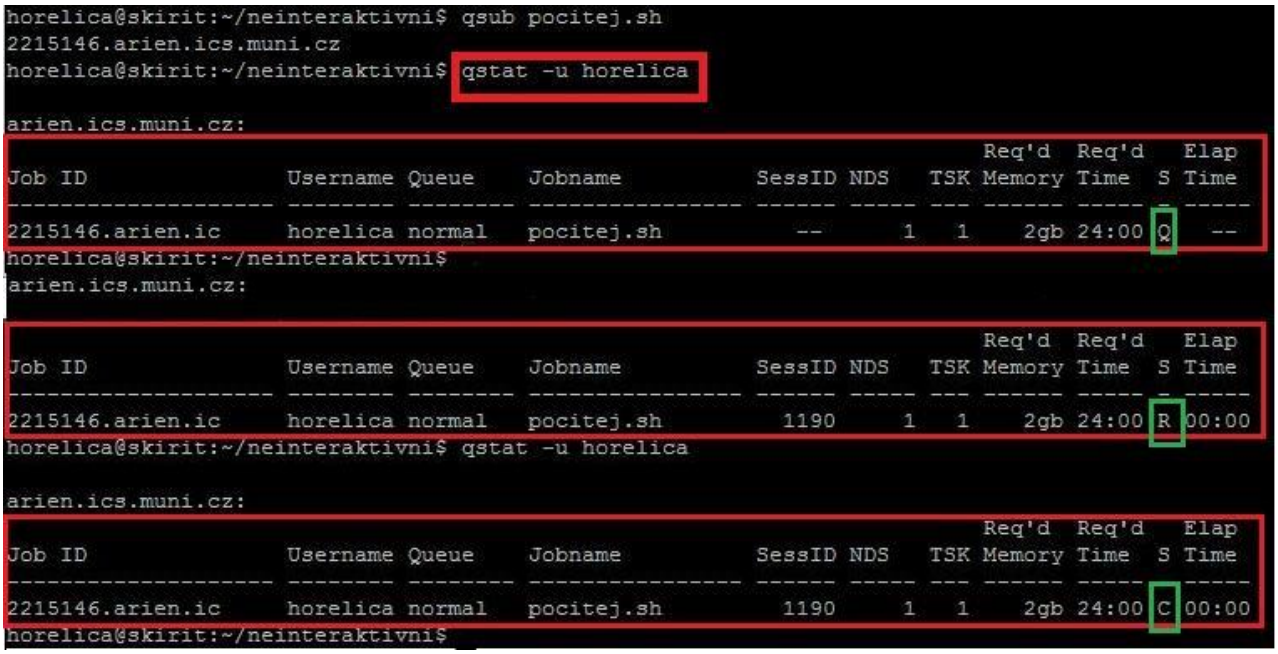
Po přihlášení na čelní uzel se musí uživatel dostat do adresáře, kde je umístěn shellový soubor. V jakém adresáři se uživatel nalézá lze zjistit příkazem ***pwd***.

Pak pomocí dalších příkazů zmíněných v kapitole 3.1 se uživatel dostane do složky, kde jsou shellové (**.sh*) a MATLABovské (**.m*) soubory. Poté stačí pouze zadat do příkazové řádky příkaz ***qsub pocitej.sh*** a tím dojde k tomu, že se začnou vykonávat jednotlivé příkazy v souboru *pocitej.sh*, obrázek 35.

```
Linux skirit.ics.muni.cz 2.6.26-2-xen-amd64 #1 SMP Fri Mar 1 10:46:35 CET 2013 x86_64
horelica@skirit:~$ pwd
/home/horelica
horelica@skirit:~$ ls
interaktivni  neinteraktivni
horelica@skirit:~$ cd neinteraktivni/
horelica@skirit:~/neinteraktivni$ ls -l
total 12
drwxr-xr-x 2 horelica meta  39 2013-03-11 08:31 01
drwxr-xr-x 2 horelica meta  39 2013-03-11 08:31 02
-rw-r--r-- 1 horelica meta  40 2013-03-11 08:32 generuj.m
-rw-r--r-- 1 horelica meta 316 2013-03-13 11:07 pocitej.sh
-rw-r--r-- 1 horelica meta 165 2013-03-08 14:46 pocitej_vse.sh
horelica@skirit:~/neinteraktivni$ qsub pocitej.sh
```

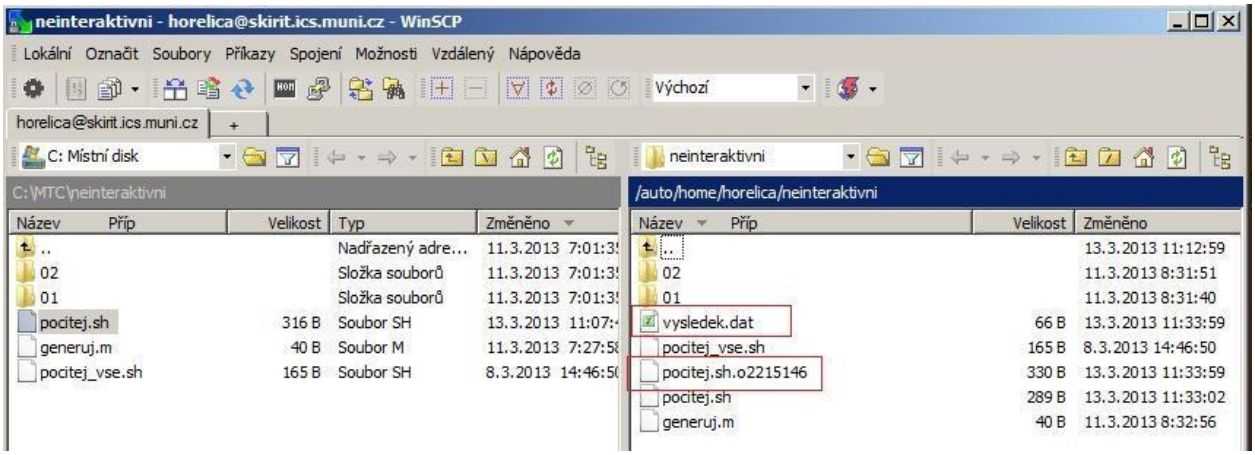
Obr. 35 Spuštění dávkového souboru na MetaCentru

Po zadání úlohy do fronty je možné sledovat její stav, a to buď na stánkách MetaCentra v menu úlohy, nebo pomocí příkazu ***qstat -u username***. Úloha nabývá stavů: Q - ve frontě, R - běží, E - končí, C - dokončena, W - čeká, H - pozdržena, T - přesunována, S – pozastavena, viz obrázek 36.



Obr. 36 Stavy úlohy na MetaCentru

Přehled všech úloh lze získat z webového rozhraní nebo příkazy, které byly zmíněny v tabulce 2, lze úlohy zobrazovat, ukončovat apod. Po ukončení výpočtů se vytvořil soubor s výsledky *vysledek.dat* a soubor *pocitej.sh.o2215146*, ve kterém je možné najít informace o přidělení strojů apod. Kdyby se navíc vytvořil soubor *pocitej.sh.e2215146*, tak by to znamenalo, že byly při výpočtu nějaké chyby. Veškeré soubory si uživatel musí stáhnout, protože po určité době dochází k jejich smazání, obrázek 37.

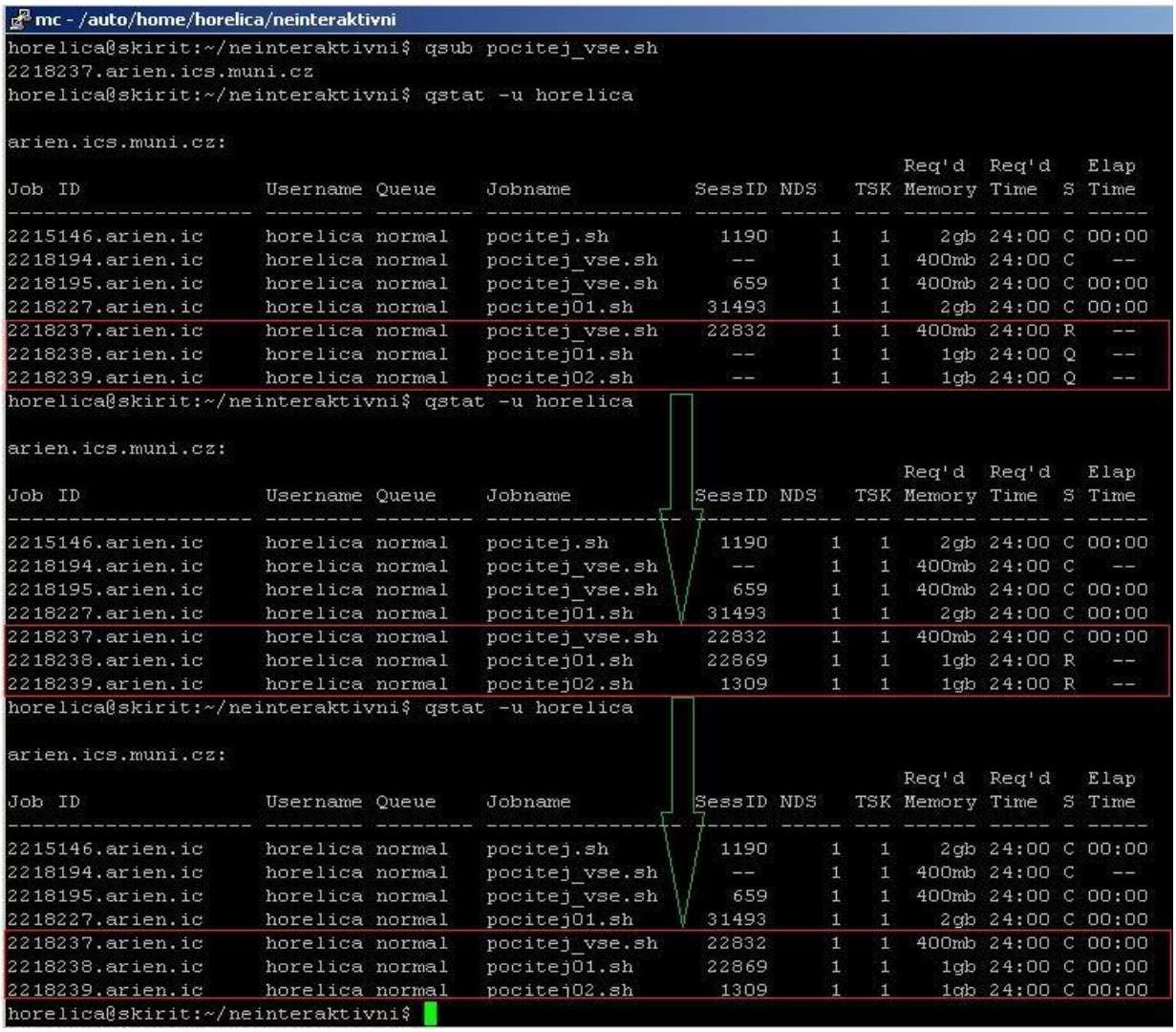


Obr. 37 Zkopírování výsledků z MetaCentra

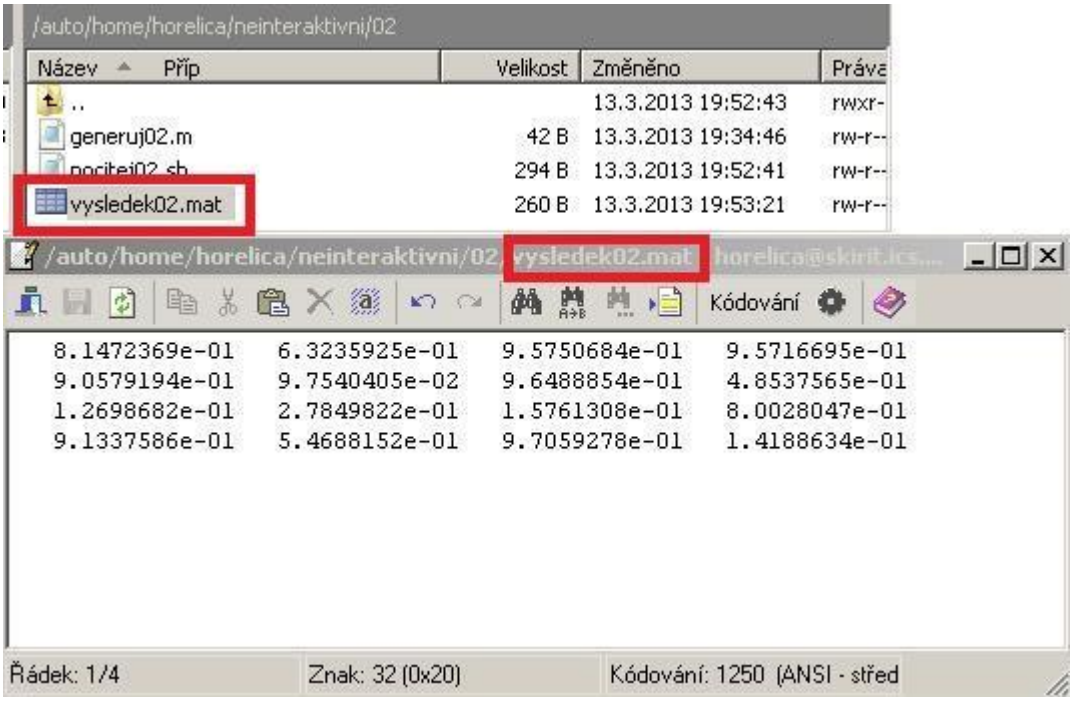
Výše uvedený příklad se týkal spouštění jedné úlohy, ale ve většině případů uživatelé spouštějí několik dávek. Proto je nutné uvést ještě případ, že se spouští několik úloh najednou. V uvedeném případě je na serveru v cestě home/horelica/neinteraktivni adresář *01* (uvnitř jsou *pocitej01.sh* a *generuj01.m*) a 02 (uvnitř jsou *pocitej02.sh* a *generuj02.m*). Spuštění všech dávek lze zajistit souborem např. *pocitej_vse.sh*, který může být umístěn kdekoliv, v uvedeném případě v adresáři *neinteraktivni.* Zde se soubor spustí příkazem ***qsub pocitej_vse.sh***. Tento soubor zajistí spuštění dávek v jednotlivých adresářích tak, jak jdou postupně za sebou.

Tímto způsobem lze spouštět i více úloh. Na obrázku 38 je vyobrazen průběh stavů úloh a obrázek 39 představuje výsledek výpočtu, resp. vygenerovanou matici o rozměru 4 x 4, vytvořené soubory s názvy *vysledek01.mat* a *vysledek02.mat*.

#!/bin/bash
qsub ./neinteraktivni/01/pocitej01.sh
qsub ./neinteraktivni/02/pocitej02.sh



Obr. 38 Průběh stavů úloh na MetaCentru



Obr. 39 Výsledek výpočtu

Pro získání detailnějších informací o proběhlých úlohách se lze podívat na stránky MetaCentra v menu úlohy, zobrazit moje úlohy, úlohy v PBS a kliknutím na číslo úlohy se zobrazí všechny údaje, např. start úlohy, délka úlohy použitý stroj atd., více obrázky 40 a 41.

Úlohy v PBS

uživatel	Počet úloh					Počet CPU úloh				
	celkem	ve frontě	běžících	dokončených	ostatních	celkem	ve frontě	běžících	dokončených	ostatních
horelica	7	0	0	7	0	7	0	0	7	0

úloha	CPU	vyhraz. paměť	použitá paměť	jméno	CPU čas	čas běhu	stav	fronta	čas vytvoření
2215146.arien.ics.muni.cz	1	2048mb	40mb	pocitej.sh	00:00:02	00:00:34	C - dokončena	normal	13.3.13 11:33
2218194.arien.ics.muni.cz	1	400mb	0mb	pocitej_vse.sh			C - dokončena	normal	13.3.13 19:30
2218195.arien.ics.muni.cz	1	400mb	0mb	pocitej_vse.sh	00:00:00	00:00:04	C - dokončena	normal	13.3.13 19:36
2218227.arien.ics.muni.cz	1	2048mb	12mb	pocitej01.sh	00:00:02	00:00:32	C - dokončena	normal	13.3.13 19:41
2218237.arien.ics.muni.cz	1	400mb	0mb	pocitej_vse.sh	00:00:00	00:00:02	C - dokončena	normal	13.3.13 19:52
2218238.arien.ics.muni.cz	1	1024mb	57mb	pocitej01.sh	00:00:04	00:00:53	C - dokončena	normal	13.3.13 19:52
2218239.arien.ics.muni.cz	1	1024mb	123mb	pocitej02.sh	00:00:03	00:00:36	C - dokončena	normal	13.3.13 19:52

Obr. 40 Úlohy v PBS, převzato a upraveno z [21]

Úloha 2218239.arien.ics.muni.cz

Základní informace

úloha	CPU	vyhraz. paměť	použitá paměť	jméno	uživatel	CPU čas	čas běhu	stav	fronta
2218239.arien.ics.muni.cz	1	1024mb	123mb	pocitej02.sh	**horelica**	00:00:03	00:00:36	C - dokončena	**normal**

požadované stroje	nodes=1:ppn=1
vytvořena	Středa, 13. březen 2013 19:52:41
způsobilá k běhu	Středa, 13. březen 2013 19:52:41
spuštěna	Středa, 13. březen 2013 19:52:47
skončí do	Čtvrtek, 14. březen 2013 20:02:58
dokončena	Středa, 13. březen 2013 19:53:26
poslední změna stavu	Středa, 13. březen 2013 19:53:25
komentář	Job started on Wed Mar 13 at 19:52
pracovní adresář	/auto/home/horelica
použitý stroj/cpuDesc	**skirit61-1/2**
seznam proměnných	PBS_O_QUEUE=normal PBS_O_HOME=/home/horelica PBS_O_LANG=C PBS_O_LOGNAME=horelica PBS_O_PATH=/usr/local/bin:/usr/bin:/bin:/usr/games:/packages/run/modules-2. PBS_O_SHELL=/bin/bash PBS_O_HOST=skirit58-1.ics.muni.cz PBS_SERVER=arien.ics.muni.cz PBS_O_WORKDIR=/auto/home/horelica

Obr. 41 Detailní informace úlohy 2218239 v PBS, převzato a upraveno z [21]

9.3 Distribuované počítání[13, 12]

Distribuované počítání může představovat práci jednoho workeru, který může zpracovávat několik tasků za sebou. Jednotlivé tasky spolu nekomunikují a nemusí běžet současně. Příkladem může být to, kdy uživatel chce počítat úlohu dávkově na jednom uzlu, s maximální délkou trvání 2 hodiny, na jednom procesoru s využitím jedné licence MATLABu.

Soubor distrib.sh

```
#!/bin/bash
# Nastavi PATH, aby se nasel MATLAB
./packages/run/modules-2.0/init/bash
module add matlab
# Vstoupi do naseho adresare
cd /home/horelica/distributivni_pocitani/
# Spusti davkove MATLAB
echo "exit(1)"|matlab -nosplash -nodisplay -nodesktop -r "distrib,exit(0)" >$PBS_JOBID.out 2>&1
if [ $? -eq 1 ] ; then
# Pokud MATLAB skoncil chybou, tak odesle e-mail
tail -n 30 $PBS_JOBID.out | mail -s "chyba v uloze $PBS_JOBID" username@server.xyz fi
```

MATLABovský soubor distrib.m

```
sched = findResource('scheduler','type','torque');          % Nalezeni Job manageru/scheduleru
set(sched,'ClusterMatlabRoot',matlabroot);                  % Odtud je MATLAB dostupny
set(sched,'ClusterOsType', 'unix');
set(sched,'DataLocation',['/home/horelica/matlab']);
set(sched,'HasSharedFilesystem', true);
set(sched,'SubmitArguments','-q short');
set(sched,'ResourceTemplate','-l nodes=^N^:ppn=1,matlab=1, matlab_MATLAB_Distrib_Comp_Engine=^N^');
job = createJob(sched);                                     % Vytvoreni distributivniho Jobu
createTask(job, @rand, 1, {2,2});                           % První nahodna matice 2x2
createTask(job, @rand, 1, {2,2});                           % Druha nahodna matice 2x2
submit(job);                                                % MATLAB interne zavola qsub
waitForState(job);
results = getAllOutputArguments(job);                       % Sber dat od jednotlivych tasku
celldisp(results);                                          % Zobrazeni vysledku nebo je mozne tyto
                                                            % vysledky ukladat do predem nadefinovanych
                                                            % formatu - osetreno ve zdrojovem kodu
```

Vysvětlivky:

- ***set(sched,'ClusterMatlabRoot',matlabroot)***;
- **DataLocation** je adresář, kam si bude MATLAB ukládat data, která bude odesílat jednotlivým workerům a také si sem bude ukládat výsledky a logy;
- **SubmitArguments** a **ResourceTemplate** jsou parametry pro příkaz ***qsub***, který MATLAB interně zavolá na spuštění jednotlivých tasků;
- Literál **^N^** - sem bude dosazeno MATLABem, automaticky, počet uzlů, které bude úloha potřebovat;
- **MATLAB_Distrib_Comp_Engine** - ^N^ licencí;
- **waitForState** - smyčka, která čeká na dokončení všech tasků - stav finished;
- Příkazem **submit**(xyz) se jednotlivé tasky stanou Torque (PBS) joby.

DataLocation, ResourceTemplate a další parametry je dobré volit dle doporučení konzistentně, tj. tak, aby adresář opravdu existoval na všech potenciálních strojích, kde se úloha může spustit.

9.4 Paralelní počítání[12]

Zde je uveden jednoduchý příklad, kdy uživatel chce počítat paralelní úlohu na čtyřech workerech s definovanými požadavky na zdroje: fronta typu short, jeden uzel (node), s jedním procesorem a s licencemi MATLABu.

MATLABovský soubor parallel_job.m

```
sched = findResource('scheduler','type','torque');
set(sched,'HasSharedFilesystem', true);
set(sched,'ClusterMatlabRoot',matlabroot);
set(sched,'ClusterOsType', 'unix');
set(sched,'DataLocation',['/home/horelica/matlab']);
set(sched,'SubmitArguments','-q short');
set(sched,'ResourceTemplate','-l nodes=1:ppn=1,matlab=1, Matlab_Distrib_Comp_Engine=^N^');
pjob = createParallelJob(sched);                % Vytvoreni paralelniho Job objektu
set(pjob, 'MinimumNumberOfWorkers', 4);
set(pjob, 'MaximumNumberOfWorkers', 4);
task = createTask(pjob,@rand, 1, {2,2});        % Vytvoreni tasku s parametry v zavorce
submit(pjob);
waitForState(pjob);
results = getAllOutputArguments(pjob);
destroy(pjob);
```

Shellový soubor pocitej_par.sh

```
#!/bin/bash
./packages/run/modules-2.0/init/bash
module add matlab
cd /home/horelica/matlab/
matlab -nodisplay -r "parallel_job"
```

Počítání se spustí opět dávkovým souborem *pocitej_par.sh*. Sledování stavu úlohy je možné opět přes webové rozhraní MetaCentra nebo příkazem ***qstat -u username***.

Na přiloženém CD diplomové práce je videoprezentace, kde je postup, jak spouštět úlohu na MetaCentru od začátku až do konce počítání s odkopírováním dat na počítač uživatele.

10. Řešení nejčastějších problémů[27]

Problémy, se kterými se uživatel může setkat, je opravdu mnoho a není možné je zde postihnout všechny, proto je uveden jen konkrétní výčet s tím, že podrobnější a aktuálnější informace lze dohledat na stránkách MetaCentra v sekci často kladené otázky z oblastí účtů, hesel, přístupů, výpočtů, používání aplikací.

1. *Je pro využívání kapacit MetaCentra vyžadován od uživatelů nějaký finanční příspěvek?*
   Využívání kapacit MetaCentra je bezplatné. Jedinou podmínkou je členství v akademické sféře České republiky.

2. *Jak obnovit platnost účtu, který jsem dlouhodobě nepoužíval, u něhož jsem nepožádal o prodloužení, a proto jsem o přístup k němu přišel?*
   Pokud si pamatujete Vaše heslo, požádejte o prodloužení účtu standardním způsobem. Pokud jste heslo zapomněli, pak postupujte podle návodu v sekci Změna hesla. Zvolíte si nové heslo a následně požádáte o prodloužení účtu.

3. *Jak funguje mechanismus prodlužování účtu?*
   Na konci kalendářního roku jsou všichni uživatelé MetaCentra vyzváni k prodloužení účtu. To se provede v sekci Výroční zpráva a prodloužení účtu selekcí období, na něž má být účet prodloužen a přiložením zprávy o tom, k jakým cílům zdroje MetaCentra v daném kalendářním roce uživatel využíval.

4. *Musím podat výroční zprávu?*
   Samozřejmě nemusíte, nicméně se nepodáním zprávy zbavujete možnosti využívat výpočetní kapacity MetaCentra.

5. *Co se stane, když výroční zprávu nepodám?*
   Když nepodáte výroční zprávu, dojde k zablokování všech Vašich aktivních účtů. Ty lze případně v budoucnu opět odblokovat, ale pouze v případě, že podáte výroční zprávu za poslední rok své aktivní činnosti v MetaCentru.

6. *Máme tedy účet v MetaCentru napořád?*
   Existence Vašeho účtu v MetaCentru je vázána pouze na Vaše působení v akademické sféře. Z Vaší strany je pouze třeba dbát na to, abyste si včas (před koncem kalendářního roku) prodloužili účet v MetaCentru a podali výroční zprávu za uplynulý kalendářní rok. Pokud jste již jednou během svého fyzického života žádali o účet v MetaCentru a ten získali, zpravidla lze Váš účet prodloužit i po delší době nečinnosti.

7. *Jak změnit heslo pro přístup ke zdrojům MetaCentra?*
   Příkazem ***kpasswd*** na libovolném stroji MetaCentra.

8. *Jak zadat joby tak, aby se spustily až po určitém časovém úseku? (například pokud si je chci připravit dopředu)?*
   Zadejte: ***qsub -a date_time***

9. *V dokumentaci se píše, že se programy mají spouštět ze svazku scratch. Jak toho mám docílit, když tam nemám právo zápisu?*
   Myslí se úloze přidělený podadresář na scratch svazku, tj.
   /scratch/vase_uzivatelske_jmeno/job_JOBID
   (identifikován proměnnou *$SCRATCHDIR*).
   Tento adresář máte jen na výkonných uzlech, ne na řídícím uzlu clusteru (skirit.ics.muni.cz apod.).

   Skript úlohy může vypadat např. takto:
   cd $SCRATCHDIR
   cp /storage/brno1/home/vase_uzivatelske_jmeno/.../vstup*
   # vlastni vypocet
   cp vystup* /storage/brno1/home/vase_uzivatelske_jmeno/.../
   # uklid
   rm -rf $SCRATCHDIR

10. *Jak můžu vymazat svoji úlohu z fronty?*
    Úlohu smažete příkazem ***qdel CISLO***, kde CISLO je označení Vaší úlohy (např. 123456.arien.ics.muni.cz).

11. *Pokud spustím dvě úlohy na tom samém nodu a každá úloha má přiřazeno jedno GPU, jak poznat, které GPU patří které úloze?*
    Informaci zjistíte následujícím příkazem:
    /usr/sbin/list_cache arien gpu_allocation | grep $PBS_JOBID

Závěr

Cílem diplomové práce bylo poskytnout uživatelům stručný návod, jak realizovat výpočetně náročné úlohy na MetaCentru. V práci byly shrnuty základní poznatky o paralelních výpočtech, výpočetních zdrojích MetaCentra a jeho charakteristika. V práci bylo využito softwarového produktu MATLAB, který je právě určen pro tyto účely. Praktickým úkolem v druhé části práce bylo vytvořit ukázku (videokurz), který ilustruje jednotlivé kroky od začátku až po konec výpočtu.

Použitá literatura a zdroje

[1] *Blaise Barney, Introduction to Parallel Computing*, *Lawrence Livermore National Laboratory,* [online] 2013, [cit. 2013-3-1].
Dostupné z WWW: **https://computing.llnl.gov/tutorials/parallel_comp**

[2] *doc. Ing. Lísal Martin DSc., Paralelní počítání, ISBN 80-7044-784-2,* 2006, Studijní opora, strana 5, Univerzita Jana Evangelisty Purkyně v Ústí nad Labem, Přírodovědecká fakulta, [cit. 2013-3-2].
Dostupné z WWW: **http://physics.ujep.cz/~mlisal/par_progrm/pprg-web.pdf**

[3] *Jan Zapletal*, *Amdahlův a Gustafsonův zákon,* Vysoká škola báňská - Technicka univerzita Ostrava, [online] 2013, [cit. 2013-3-2]. Dostupné z WWW: **http://homel.vsb.cz/~zap150/pa/ref/pa_ref.htm**

[4] *Ing. Tomáš Ťoupal, Úvod do programu MATLAB,* strana 1, Západočeská univerzita, Fakulta aplikovaných věd, [online] 2013, [cit. 2013-3-3].
Dostupné z WWW: **http://home.zcu.cz/~ttoupal/www-kma/STAV/stav_10.pdf**

[5] *Wikipedie, MATLAB,* [online] 2013, [cit. 2013-3-4].
Dostupné z WWW: **http://cs.wikipedia.org/wiki/MATLAB**

[6] *Ing. Michal Hajžman, Ph.D., Základy programování v MATLABu,* Západočeská univerzita, Fakulta aplikovaných věd*, KME,* [online] 2013, [cit. 2013-3-7].
Dostupné z WWW: **http://home.zcu.cz/~mhajzman/matl_zaklad.pdf**

[7] *Ivan Nagy*, *Úvod do MATLABu,* [online] 2013, [cit. 2013-3-16].
Dostupné z WWW: **http://www.fd.cvut.cz/personal/nagyivan/MatUvod.pdf**

[8] *doc. RNDr. Josef Blažek, CSc., Kurz MATLABu,* Jihočeská univerzita v Č. Budějovicích, Pedagogická fakulta, Katedra aplikované fyziky a techniky, [online] 2013, [cit. 2013-3-19].
Dostupné z WWW:
http://www.eamos.cz/amos/kat_fyz/modules/external/index.php?kod_kurzu=kat_fyz_1272

[9] *Rostislav Holec, Paralelní programování v prostředí MATLAB*, strana 1,
Vysoká škola báňská - Technická univerzita Ostrava, [online] 2013, [cit. 2013-3-22].
Dostupné z WWW: **http://homel.vsb.cz/~hol527/pds/ref/preklad.html**

[10] *The MathWorks, Parallel Computing Toolbox,* [online] 2013, [cit. 2013-3-26]. Dostupné z WWW:
http://http://www.mathworks.com/help/distcomp/index.html

[11] *The MathWorks, MATLAB*, [online] 2013, [cit. 2013-3-27].
Dostupné z WWW: **http://www.mathworks.com/cmsimages/62006_wl_mdcs_fig1_wl.jpg**

[12] *Wikipedie MetaCentra VO, MATLAB,* [online] 2013, [cit. 2013-3-28].
Dostupné z WWW: **https://wiki.metacentrum.cz/wiki/Matlab**

[13] *Princeton university, MATLAB, Department of Computer Science Computing Guide,* [online] 2013, [cit. 2013-3-29]. Dostupné z WWW:
https://csguide.cs.princeton.edu/software/matlab?destination=node%2F167

[14] *Boston university, How to use pmode, Information services & Technology,* [online] 2013, [cit. 2013-3-30]. Dostupné z WWW: **http://www.bu.edu/tech/research/training/scv-software-ackages/matlab/pct/how_to_use_pmode/**

[15] *MetaCentrum, Projekt MetaCentrum,* [online] 2013, [cit. 2013-3-31]. Dostupné z WWW: **http://www.metacentrum.cz/cs/about/meta/**

[16] *Wikipedie MetaCentra VO, O MetaCentru VO* [online] 2013, [cit. 2013-4-1]. Dostupné z WWW: **http://metavo.metacentrum.cz/**

[17] *Wikipedie MetaCentra VO, Statistiky 2012,* [online] 2013, [cit. 2013-4-2]. Dostupné z WWW: **http://metavo.metacentrum.cz/cs/state/stats/2012/index.html**

[18] *Wikipedie MetaCentra VO, Statistiky 2011,* [online] 2013, [cit. 2013-4-2]. Dostupné z WWW: **http://metavo.metacentrum.cz/cs/state/stats/2011/index.html**

[19] *MetaCentrum VO, Hardware* [online] 2013, [cit. 2013-4-6]. Dostupné z WWW: **http://metavo.metacentrum.cz/pbsmon2/hardware;jsessionid=9815AA630F91294885282DD1E4B8043A**

[20] *Wikipedie MetaCentra VO, Souborové systémy v MetaCentru,* [online] 2013, [cit. 2013-4-4]. Dostupné z WWW: **https://wiki.metacentrum.cz/wiki/Souborov%C3%A9_syst%C3%A9my_v_MetaCentru**

[21] *MetaCentrum VO, Můj účet,* [online] 2013, [cit. 2013-3-13/14]. Dostupné z WWW: **http://metavo.metacentrum.cz/cs/myaccount/index.html**

[22] *Wikipedie MetaCentra VO, Aplikace,* [online] 2013, [cit. 2013-4-6]. Dostupné z WWW: **https://wiki.metacentrum.cz/wiki/Kategorie:Aplikace**

[23] *Wikipedie MetaCentra VO, Úlohy a plánovač,* [online] 2013, [cit. 2013-3-10]. Dostupné z WWW: **https://wiki.metacentrum.cz/wiki/Kategorie:%C3%9Alohy_a_pl%C3%A1nova%C4%8D**

[24] *MetaCentrum VO*, *Fronty - server arien.ics.muni.cz - Produkční prostředí*, [online] 2013, [cit. 2013-3-1]. Dostupné z WWW: **http://metavo.metacentrum.cz/pbsmon2/queues/list**

[25] *Michal Kafka, Odhad času spuštění úlohy v plánovacím systému MetaCentra,* strana 4, Masarykova univerzita, Fakulta informatiky, [online] 2013, [cit. 2013-3-26]. Dostupné z WWW: **http://is.muni.cz/th/143303/fi_m/dp-print.pdf**

[26] *Altair Engineering, Inc., High performance computing - PBS Professional 9.1, User's Guide,* [online] 2013, [cit. 2013-3-28]. Dostupné z WWW: **http://www.hpc.maths.unsw.edu.au/sites/www.hpc.maths.unsw.edu.au/files/PBSPro-UserGuide-9.1.pdf**

[27] *Wikipedie MetaCentra VO, Často kladené otázky,* [online] 2013, [cit. 2013-4-20]. Dostupné z WWW: **https://wiki.metacentrum.cz/wiki/%C4%8Casto_kladen%C3%A9_ot%C3%A1zky**

Seznam obrázků

Seznam tabulek

Seznam grafů