Západočeská univerzita v Plzni
Centrum informatizace a výpočetní techniky

Informační Bulletin
CIV

PC clustery
na ZČU v Plzni

2

Prosinec 2002

Příspěvky uvedené v bulletinu jsou dílem kolektivu autorů CIV a ITI.
Publikace neprošla jazykovou ani grafickou úpravou.

Redakční rada: J. Sitera, E. Chánová, M. Otta a V. Rudolf.

Sazba písmem Bookman v systému LaTeX2$_\varepsilon$.
Vytiskl TYPOS — Digital Print s.r.o., závod Plzeň.

Vydání první, náklad 250 výtisků.
Vydala Západočeská univerzita v Plzni.

ISBN 80—7082–942–7

OBSAH

REDAKČNÍ POZNÁMKA

Tato publikace je určena studentům a zaměstnancům ZČU v Plzni a pracovníkům ITI UK v Praze, částečně i jiných organizací. Může sloužit jako základní referenční příručka pro používání PC clusterů na ZČU v Plzni a v *METACentru*.

Členem cílové skupiny čtenářů je uživatel, který má zkušenosti s počítáním a zajímá se o možnost používat PC clustery na Linuxu. Další cílovou skupinou jsou i uživatelé dosud počítající lokálně a hledající možnosti využívání dalších zdrojů.

Tato příručka byla vydána při příležitosti pořádání semináře pro uživatele z řad studentů a zaměstnanců ZČU v Plzni v prosinci 2002.

PROSINEC 2002

KAPITOLA 1

ÚVODEM NĚCO OBECNĚ O CLUSTERECH

1.1 Co jsou to clustery

Anglicko-český slovník nás poučí, že výraz „cluster“ má překlady „shluk, chumel, hlouček, seskupení“ atd. Představme si velké množství relativně „obyčejných“, ale velmi výkonných počítačů řady PC, propojených velmi rychlou sítí a vybavených vhodným komunikačním softwarem tak, aby byly schopny spolupracovat na řešení jedné úlohy podobně, jako probíhá spolupráce procesorů uvnitř klasického superpočítače. Tak to je, s trochou zjednodušení, cluster pracovních stanic, či krátce jen „cluster“. Hlavní výhodou takovéhoto výpočetního prostředku je velmi příznivý poměr ceny a výkonu, jelikož v dnešní době lze takový cluster stavět z masově vyráběných komponent.

Cluster je typicky zařízení s homogenními uzly propojenými velmi výkonnou sítí, pokud možno co nejspolehlivěji a na krátkou vzdálenost. V jedné nebo několika málo přístrojových skříních je velké množství počítačů PC (v tenkém a „rackovém“ provedení), komunikační přepínač, chlazení apod. Nejsou to tedy žádná víceúčelová zařízení, z nichž by se pro výpočty používal jen nadbytečný výkon. Typický cluster používá operační systém Linux[1] a jiné „open source“ technologie, což ho v očích uživatelů poněkud odlišuje od superpočítače obvykle vybaveného firemním operačním systémem.

[1] Existují také PC clustery pracující s operačními systémy MS Windows, ty však nejsou vzhledem k okolnostem předmětem tohoto článku

Klasický superpočítač má (přes mnoho vnitřních technologií, které toto alespoň navenek pro uživatele zajišťují) jednu sdílenou paměť, jedny periferie a jednu vysokorychlostní komunikační sběrnici, která to vše propojuje. Z hlediska uživatele (aplikačního programátora) se tak jeví jako jeden, velmi výkonný, ale homogenní počítač.

Vývoj v oblasti PC clusterů ale za poslední roky velmi pokročil a dnes již existuje celá řada speciálních SW nástrojů, obvykle označovaných pojmem middleware, které slouží pro vytvoření virtuálního superpočítače nad hardwarem clusteru a tím pomáhají aplikačním programátorům a uživatelům — umožňují jim pracovat ve známém světě, byť virtuálním.

Lze tedy říci, že trend v oblasti náročných výpočtů v poslední době jednoznačně ukazuje přesun od klasických superpočítačů ke clusterům. Zejména pro úlohy, jejichž podstata dovoluje paralelizaci na slabě vázané procesy (lze přirozeně navrhnout řešení úlohy tak, že se rozpadne na více relativně nezávislých problémů, které nepotřebují častou synchronizaci či výměnu výsledků), je tento přístup výrazně efektivnější. Teď je zřejmě ta správná doba pro všechny uživatele z oblasti velmi náročných výpočtů (pokud již tak dávno neučinili) začít se o tento trend zajímat, byť třeba jen v úvahách do budoucna.

1.2 Zkušenosti s PC clustery

K tomu, co již bylo výše řečeno o clusterech a trendu v oblasti náročných výpočtů, je možno pro potenciální, respektivě váhající zájemce o tuto technologii dodat několik povzbudivých slov, vycházejících zejména z dosavadních zkušeností získaných v různých projektech.

Rozhodně je třeba říci, že clustery nejsou doménou matematiků a informatiků. Typickým „velkým“ uživatelem jsou fyzici, speciálně fyzici vysokých energií. Projekt DataGrid, který je v tomto směru jednoznačně největším konzumentem výpočetní kapacity, vznikl na základě potřeby zpracovat obrovské množství dat vzniklých v urychlovači, který buduje evropská komunita fyziků vysokých energií v CERNu. Zpracování samo má distribuovaný charakter, v podstatě lze říci, že několik skupin fyziků hledá naprosto nezávisle na sobě a vlastními algoritmy v jedněch datech specifické znaky. Navíc většina z těchto algoritmů má charakter prohledávání hrubou silou (metoda Monte Carlo), tj. má značné nároky na celkovou výpočetní kapacitu s minimálními nároky na synchronizaci. Významná skupina fyziků v Praze napojených na DataGrid se zabývá simulací detektorů, tj. zařízení, kterými proudí částice a jež na základě

toho produkují data ke zpracovaní. Tato simulace slouží jak k návrhu detektorů, tak k pracím souvisejícím s pochopením dat, která hotový detektor produkuje.

V Brně a Praze je také silná komunita uživatelů z řad chemiků, respektivě biochemiků. Zejména pak uživatelé z Národního centra pro výzkum biomolekul v Brně již dnes pro velkou část svých výpočtů využívají clustery namísto klasických superpočítačů.

Obecněji řečeno, všechny problémy, které vedou k řešení metodami typu Monte Carlo, či jsou jinak snadno paralelizovatelné v režii řešitele, patří v první řadě k vhodným kandidátům na použití clusterů. U problémů řešených s využitím firemních softwarů je situace vždy závislá na existenci příslušné portace pro danou platformu (Linux), podpoře distribuovaného zpracování výpočtu a v neposlední řadě také na typu a množství licencí pro daný produkt.

Kapitola 2

PC clustery na ZČU v Plzni

Na ZČU v Plzni je zhruba od poloviny roku 2002 umístěn PC cluster pořízený v rámci výzkumného záměru, jehož nositelem je CESNET, zájmové sdružení vysokých škol včetně ZČU. Jedná se 16 uzlů, každý se dvěma procesory Intel Pentium III o taktovací frekvenci 1 GHz, 1 GB paměti a 18 GB pevným diskem. Označení clusteru je (podle předřazené front–endové stanice) *nympha*.

Začátkem října 2002 byl pak na ZČU uveden do provozu nový cluster, jenž byl pořízen z prostředků Institutu teoretické informatiky (ITI), společného vědeckovýzkumného pracoviště UK Praha, AV ČR a ZČU v Plzni, financovaného v rámci programu výzkumných center. Jedná se 16 uzlů, každý se dvěma procesory AMD Athlon MP o taktovací frekvenci 1.9 GHz, 1 GB paměti a 40 GB pevným diskem. Označení clusteru je (podle předřazené front–endové stanice) *minos*.

Se zbytkem světa je připojení obou clusterů realizováno přímo prostřednictvím plzeňského bodu přítomnosti národní sítě výzkumu a vývoje CESNET2 tak, aby možnost spolupráce clusterů v rámci *META Centra* byla co nejlepší (aktuálně linka Plzeň–Praha 2.5 Gbit/s a linka Praha–Brno 10 Gbit/s).

Oba PC clustery jsou k dispozici v rámci projektu *META Centrum* všem uživatelům, kteří splní základní jednoduché podmínky pro jeho užívání. Díky vzájemnému propojení obou zařízení lze v případě potřeby pracovat na obou strojích současně — a komu ani toto nestačí, může mít k dispozici díky *META Centru* i další stroje *META Centra* (v Praze či Brně).

2.1 Kdo jsou aktéři projektu

Institut teoretické informatiky UK (ITI, `http://iti.zcu.cz`) byl založen v roce 2000 jako společné pracoviště Univerzity Karlovy v Praze, Matematického ústavu AV ČR, Ústavu informatiky AV ČR a Západočeské univerzity v Plzni v rámci grantového programu výzkumných center. Odborně je centrum zaměřeno především na základní výzkum v oblasti diskrétní matematiky a teoretické informatiky. Plzeňské pracoviště ITI pracuje při katedře matematiky FAV, na níž byl již dříve (v roce 1998) instalován cluster s 12 CPU, pořízený z grantových prostředků projektu „Lyra“ grantového programu MŠMT INFRA II. Tento cluster v současné době slouží hlavně k přípravě a ladění úloh pro výkonnější počítače a pro výuku. S clustery pracovních stanic má komunita matematiků a informatiků na ZČU v Plzni tedy již zkušenosti z minulých let.

Projekt *META Centrum* (`http://meta.cesnet.cz`) vznikl v roce 1996 s cílem vytvořit jednotné prostředí pro uživatele nad heterogenní sítí výpočetních kapacit poskytovaných superpočítačovými centry vysokých škol. V rámci projektu je toto prostředí udržováno a rozvíjeno dodnes. Z hlediska investičního rozvoje se v poslední době *META Centrum* věnuje právě clusterům. V tuto chvíli je k dispozici ještě dalších celkem 96 procesorů (Pentium III, část 700 MHz, část 1 GHz) s celkem 48 GB paměti, dislokovaných v Brně (64 CPU) a Praze (32 CPU). Navíc koncem letošního roku *META Centrum* investovalo do clusterové technologie, takže níže popisovaná konfigurace není finální.

2.2 Instalované zařízení na ZČU v Plzni

Při pohledu na PC cluster *minos* (viz obrázek 2.1) možná rozpoznáte 16 plochých PC umístěných nad sebou ve speciální přístrojové skříni (takzvaném „racku“). PC cluster *nympha* vypadá obdobně. A nyní se podívejme podrobněji na konfiguraci jednotlivých PC clusterů.

2.2.1 Cluster *nympha*

PC cluster *nympha* je umístěn na serverovně CIV-LPS na ZČU v Plzni, má 16 výpočetních uzlů HP LP1000r, kterým je předřazena řídící stanice `nympha.zcu.cz`.

Technické parametry výpočetních uzlů (HP LP1000r):
Procesory: 2 × Intel Pentium III 1 GHz, 256 Kb Cache, 133 MHz FSB

Obrázek 2.1: PC cluster *minos* při instalaci

Paměť: 1 GB SDRAM 133 MHz FSB
Disky: 18 GB SCSI-3 10 000 rpm
Síťové připojení: 100 Mb/s Fast Ethernet, 1.2 Gb/s Myrinet
Hostname: {`nympha1-nympha16`}`.zcu.cz`

Speciální vybavení:
Myrinet M2L-SW16 switch

Technické parametry řídící stanice:
Procesor: 1 × Intel Pentium 4 1.4 GHz
Paměť: 512 MB RAM
Disky: 60 GB 10 000 rpm
Síťové připojení: 100 Mb/s Fast Ethernet
Hostname: `nympha.zcu.cz`

2.2.2 Cluster *minos*

PC cluster *minos* je umístěn na serverovně CIV-LPS na ZČU v Plzni, má 16 výpočetních uzlů Dual AMD Athlon 1900+, kterým je předřazena řídící stanice `minos.zcu.cz`. Tento cluster podléhá speciálnímu režimu řízení zpracovávání úloh tak, aby jej mohli prioritně využívat zejména pracovníci ITI. Další uživatelé mají přístup pouze pomocí krátkodobých a střednědobých front v PBS, viz kapitola 3.3.

Technické parametry výpočetních uzlů:
Procesory: 2 × AMD Athlon MP 1900+ 1.6 GHz, 256Kb Cache, 266 MHz FSB
Paměť: 1 GB SDRAM ECC 266MHz FSB
Disk: 40 GB Ultra ATA IV 7 200 rpm
Síťové připojení: 1 Gb/s Gigabit Ethernet
Hostname: {`minos1-minos16`}`.zcu.cz`

Speciální vybavení:
3Com SS3 4924 Gigabit switch

Technické parametry řídící stanice:
Procesory: 2 × Pentium III Xeon 800 MHz
Paměť: 512 MB SDRAM ECC reg. 133 MHz FSB
Disky: 8 GB Ultra3 SCSI 10 000 rpm + 36 GB Ultra3 SCSI 15 000 rpm
Síťové připojení: 1 Gb/s Gigabit Ethernet
Hostname: `minos.zcu.cz`

2.3 Další PC clustery *METACentra*

2.3.1 Cluster *skirit*

PC cluster *skirit* se nachází na půdě ÚVT MU Brno. Má celkem 16 výpočetních uzlů SGI 1200 a 16 výpočetních uzlů HP LP1000r, kterým je předřazena jedna řídící stanice `skirit.ics.muni.cz`.

Technické parametry výpočetních uzlů (SGI 1200):
Procesory: 2 × Intel Pentium III 700 MHz 256 Kb Cache, 100 MHz FSB
Paměť: 1 GB SDRAM 100 MHz FSB
Disky: 9 GB Ultra2 SCSI 7 200 rpm
Síťové připojení: 1 Gb/s Gigabit Ethernet

Hostname: {skirit1-skirit16}.ics.muni.cz
Speciální vybavení:
HP ProCurve 4108 GL Gigabit switch

Technické parametry výpočetních uzlů (HP LP1000r):
Procesory: 2 × Intel Pentium III 1 GHz 256 Kb Cache, 133 MHz FSB
Paměť: 1 GB SDRAM 133 MHz FSB
Disky: 18 GB SCSI-3 10 000 rpm
Síťové připojení: 100 Mb/s Fast Ethernet, 2 Gb/s Myrinet
Hostname: {skirit17-skirit32}.ics.muni.cz
Speciální vybavení:
Myrinet M3-E32 switch

Technické parametry řídící stanice:
Procesor: 1 × Intel Pentium 4 1.4 GHz
Paměť: 512 MB SDRAM 100 MHz FSB
Disky: 18 GB Ultra2 SCSI 7 200 rpm + 2 × 36 GB Ultra2 SCSI 7 200 rpm
Síťové připojení: 1 Gb/s Gigabit Ethernet
Hostname: skirit.ics.muni.cz

2.3.2 Cluster *skurut*

PC cluster *skurut* se nachází na půdě CESNETu v Praze. Má 16 výpočetních uzlů SGI 1200, kterým je předřazena řídící stanice, která má hostname skurut.cesnet.cz.

Technické parametry výpočetních uzlů (SGI 1200):
Procesory: 2 × Intel Pentium III 700 MHz 256 Kb Cache, 100 MHz FSB
Paměť: 1 GB SDRAM 100 MHz FSB
Disky: 9 GB Ultra2 SCSI 7 200 rpm
Síťové připojení: 100 Mb/s Fast Ethernet
Hostname: {skurut1-skurut16}.cesnet.cz

Technické parametry řídící stanice (SGI 1200):
Procesor: 2 × Intel Pentium III 700 MHz 256 Kb Cache, 100 MHz FSB
Paměť: 512 MB SDRAM 100 MHz FSB
Disky: 18 GB Ultra2 SCSI 7 200 rpm + 2 × 36 GB Ultra2 SCSI 7 200 rpm
Síťové připojení: 100 Mb/s Fast Ethernet
Hostname: skurut.cesnet.cz

2.4 Co nás stojí superpočítání

V realitě je to tak, že lidé, kteří mají potřebu realizace náročných vědeckotechnických výpočtů, musí řešit i jiné otázky než technické. Je lépe koupit si pro řešení našeho úkolu vlastní zařízení, mít nad ním plnou kontrolu, ale také nést všechny starosti a náklady spojené s jeho provozem, nebo je výhodnější uvažovat o využití nabízených služeb? Takové otázky musí řešit řada pracovišť. Jednou, snad nejdůležitější otázkou v tomto uvažování, je otázka finanční. Co mne, jako zaměstnance ZČU, bude stát použití clusteru? Jak to vlastně je s takovými službami v rámci rozpočtu ZČU?

Zde je třeba zdůraznit, že nová metodika rozpočtu ZČU umožňuje u investičního majetku pořízeného z grantových zdrojů eliminovat negativní efekty odpisového mechanismu. Protože obě uvedená zařízení jsou, jak již bylo řečeno, pořízena z grantových prostředků, a uvedené mechanismy na ně byly plně aplikovány (cluster *META Centra* je navíc v majetku CESNETu), nejsou jejich uživatelé zatěžováni žádnými platbami (typu „podílu na odpisech“ apod.) Prakticky to znamená, že po vyřízení minimálních formalit (registrace, zřízení konta atd.) jsou obě zařízení uživatelům z řad akademické obce ZČU k dispozici pro jejich výpočty. Jsou zde jen dvě omezení, jež jsou (nebo by měla být) při daném zdroji financování samozřejmá:

- zařízení využívat pouze pro nekomerční účely,
- v publikačních výstupech ocitovat čísla příslušných grantových projektů.

KAPITOLA 3

PŘÍSTUP KE ZDROJŮM

3.1 Elektronická agenda uživatele

Dříve než začneme pracovat na výpočetních zdrojích v rámci *META Centra*, musíme absolvovat jistou administrativní proceduru. Tato procedura je podporovaná elektronickým systémem (přes WWW rozhraní), některé úkony je však třeba realizovat písemným stykem s příslušným uzlem *META Centra*. V některých případech (např. cluster *minos*) navíc zřízení přístupu podléhá schválení zástupcem partnerské organizace (v tomto případě za ITI Ing. R. Kuželem).

Tématem tohoto sborníku jsou clustery a to zejména ty instalované na ZČU. Je proto namístě nejprve říci, že pokud jste uživatelem *META Centra* (například používáte superpočítače ZSC *kirke* a *pasifae*), stačí Vám podat žádost o přístup na clustery *nympha* a *minos* (viz dále). Vaše uživatelské konto v *META Centru* je jedno a budete jej nadále používat.

Poznámka: Všechny odkazované funkce WWW rozhraní jsou přístupné z webové stránky *META Centra*, která se nachází na URL

`http://meta.cesnet.cz`.

Níže uvedená konkrétní URL slouží pouze pro rychlejší orientaci v případě, že máte na očích tento sborník.

3.1.1 Elektronická přihláška

Pokud nejste uživatelem *META Centra*, začněte podáním přihlášky. Přihláška se vyplňuje elektronicky prostřednictvím WWW formuláře. Ten najdete na adrese

`https://scb.ics.muni.cz/active-pub/registration/`.

Zde prosím vyplňte úplně a správně požadované údaje. Pokud jste zaměstnanci ZČU, uveďte prosím do položky UČO svoje uživatelské jméno v projektu ORION. Vaše osobní číslo na ZČU nám bohužel vzhledem k problémům s propojením databází nepřináší žádnou informaci. Také jako svůj požadovaný login prosím uvádějte stejný login jako máte v ORIONu. Pokud nebude již zabraný uživatelem mimo ZČU, tak vám tato volba poněkud zjednoduší život.

Do projektu se můžete připojit dodatečně, nicméně pokud patříte mezi uživatele z řad ITI, připojte se k příslušnému projektu raději hned, zejména z důvodu získání příslušných privilegií (viz dále).

V sekci „požadované účty“ nezapomeňte označit clustery *nympha* a *minos*. Opět lze požadavek na zřízení účtu podat dodatečně, dojde však ke zbytečnému zdržení.

Žádost bude automaticky elektronicky postoupena ke schválení příslušnému pracovníkovi (v našem případě ZSC — Ing. J. Kňourek a ITI — Ing. R. Kužel). Do kolonky zdůvodnění přihlášky máte možnost volným textem uvést další informace pro tohoto pracovníka.

Po odeslání přihlášky se vytvoří její forma vhodná pro tisk. Vytiskněte ji, podepište a dle údajů na WWW zašlete na příslušnou adresu. Svým podpisem stvrzujete souhlas s pravidly *METACentra*, pověřený pracovník čeká se schválením Vaší přihlášky na doručení tohoto dokumentu.

3.1.2 Projekty

Pod pojmem projekt se v kontextu uživatelské administrativy *METACentra* myslí ucelený pracovní záměr, spojující uživatele do skupiny, která má společné vlastnosti. Typicky z členství v projektu plynou jistá přístupová oprávnění a priority. Projekty se také používají v souvislosti s politikou prodlužování účtu (viz dále) k označení úkolu, na kterém uživatel pracuje.

Z hlediska tohoto sborníku je důležitý projekt ITI, který slouží k označení pracovníků ITI. Z členství v tomto projektu plynou priority na clusteru *minos*. Každý uživatel může být ve více projektech, lze se tedy dále přihlásit k projektu dle konkrétního výzkumu (členství v projektu ITI je spíše dáno příslušností k pracovišti).

Pro vytváření projektů, přihlašování se k nim a odhlašování slouží samostatná WWW stránka pod administrativou účtu (viz dále).

3.1.3 Osobní administrativa účtu

Po vytvoření konta v *META Centru* se každému uživateli zpřístupní (po zadání uživatelského jména a hesla) část WWW prostoru, kde pracuje pod svojí osobní identitou (Osobní administrativa). Zde se provádějí dále popsané úkony i výše odkazovaná práce s projekty. Důležité je říci, že každý uživatel nese odpovědnost za aktuálnost svých osobních údajů v databázi, přičemž k jejich úpravě slouží formulář „Osobní údaje". Zejména prosím dbejte na to, aby vaše adresa elektronické pošty byla funkční.

Prakticky: Máte-li již konto v *META Centru*, zvolte „Osobní údaje" na URL

```
https://scb.ics.muni.cz/active/personal/
```

a zkontrolujte a případně opravte zde uvedené údaje. Zcela jistě došlo např. ke změně telefonních čísel pracovníků ZČU.

3.1.4 Žádost o přístup na konkrétní uzel

Jednou z funkcí osobní administrativy je i žádost o přístup na další uzly *META Centra*. Tuto funkci použije například uživatel, který již má konto v *META Centru* a chce používat clustery na ZČU. Jelikož tyto podléhají schválení přístupových oprávnění (viz výše), může operace trvat delší dobu.

Prakticky: Máte-li konto v *META Centru* a chcete používat clustery na ZČU, přihlaste se na URL

```
https://scb.ics.muni.cz/active/personal/
```

svým uživatelským jménem a heslem, zvolte „Žádost o další účty", vyberte clustery *nympha* a *minos* a žádost odešlete. V případě problémů kontaktujte výše uvedené pracovníky.

3.1.5 Výroční zpráva — politika prodlužování účtů

Politika prodlužování účtů v podstatě vyjadřuje politiku *META Centra* vůči uživatelům. Prakticky kdokoli, kdo splní základní podmínky akademického využití výpočetních kapacit a software, může výše popsaným postupem získat uživatelské konto. Poté má možnost volně pracovat a využívat standardním způsobem zdrojů *META Centra* a to po dobu jednoho roku. Na konci roku musí požádat o prodloužení uživatelského konta,

přičemž nutnou přílohou k této žádosti je výroční zpráva — text popisující jeho práci a dosažené výsledky. Na základě této zprávy, případně dalších vyžádaných informací, je rozhodnuto o prodloužení konta. Proces se opakuje každý rok, přičemž k dosaženým výsledkům může být přihlíženo také při přidělování nadstandardních zdrojů a stanovování priorit přístupu k nim.

Pokud uživatel pracuje na konkrétním a předem vyprofilovaném tématu, je žádoucí toto uvést formou projektu. To kromě jiného umožňuje podávat jednu souhrnnou výroční zprávu za celý projekt. Každý uživatel sice musí osobně podat žádost o prodloužení účtu (pouze elektronická forma), ale namísto přiložení výroční zprávy se může odkázat na projekt.

V běžném pracovním cyklu (mimo nových uživatelů) se předpokládá odevzdávání výročních zpráv na konci kalendářního roku. Výše popsaná aplikace politiky prodlužování účtu je zároveň prostředkem pro sběr podkladů pro tvorbu „Ročenky *META Centra* “, dokumentu, který prezentuje činnost uživatelů *META Centra* [1].

Prakticky: Máte-li konto v *META Centru* již delší dobu, využijte situace na konci roku, kdy pravděpodobně stejně musíte podávat minimálně průběžné zprávy svých výzkumných aktivit a podejte i výroční zprávu *META Centra* — na URL

```
https://scb.ics.muni.cz/active/personal/,
```

volba „Výroční zpráva a prodloužení účtu“.

3.2 Uživatelské prostředí

Řídící stanice jsou určeny k přípravě a zadávání úloh, výpočetní stanice k jejich provádění. Na řídících stanicích není dovoleno za normálních okolností spouštět úlohy. Na výpočetních uzly se uživatelé obvykle nepřihlašují.

Přihlášení k řídícím stanicím je možno provést pomocí kryptovaného spojení (například ssh) několika způsoby. Využijeme-li naše konto na strojích *META Centra* v Plzni, skládá se přihlášení z prostředí ORION z následujících příkazů:

```
add meta
metaconnect kirke
ssh nympha
```

Domovské adresáře na jednotlivých částech clusteru jsou na řídících stanicích *nympha*, *minos*, *skirit* a *skurut*, a jsou v rámci clusteru sdíleny

prostřednictvím NFS. V domovských adresářích není možno spouštět úlohy, které mají jakékoli nároky na vstupně-výstupní operace (kromě triviálních). K tomu jsou určeny adresáře `/scratch/`*`username`* na všech výpočetních stanicích clusteru. Soubory potřebné pro běh úlohy je nutno na jejím začátku zkopírovat z domovského adresáře a na konci opět uložit, nebo lépe nakopírovat prostřednictvím PBS s využitím *stagein* a *stageout* operací (viz dokumentace k PBS).

V domácím adresáři je také vytvořen link `shared` na domovský adresář v příslušné AFS buňce uživatele, tedy například v případě uživatele *knourek* ze ZČU na `/afs/zcu.cz/users/k/knourek`. Protože má po přihlášení pověření i pro buňku `zcu.cz` (ověříte příkazem `klist -T`), může přistupovat k AFS adresářům, které jsou mu běžně dostupné i v prostředí ORION.

Podobně jako v prostředí ORION je aplikační software rozdělen do balíků, tzv. *modulů*. Systém správy prostředí *modules* zajišťuje uživatelské rozhraní k modifikacím prostředí potřebným pro spouštění aplikačních programů. Umožňuje tak konfigurovat relaci uživatele nezávisle na platformě a fyzickém umístění potřebných souborů, a vyhnout se tak mnohdy komplikovaným ručním modifikacím prostředí.

Následující příkazy vypíší seznam dostupných modulů na daném systému a připojí modul *fluent60*.

```
module available
module load fluent60
```

Standardně má uživatel nastaven jako přihlašovací shell *bash*. Jeho konfigurace či další nastavení je možno provést v souboru `.bashrc` v domovském adresáři. Uživatel, zvyklý na prostředí ORION, možná změní řádku `set -o vi` na `set -o emacs` aby mohl editovat pohodlně příkazové řádky. Do souboru `.bashrc` je také vhodné umístit příkazy připojující příslušný aplikační software pro případ paralelního běhu programů.

3.3 Systém dávkového zpracování úloh — PBS

V této podkapitole bude popsán zhruba systém dávkového zpracování úloh — PBS. Uživatel najde odpovídající dokumentaci k PBS na URL `http://meta.cesnet.cz`, proto se zde systémem nebudeme zabývat podrobně. Vysvětlíme pouze některé základní principy, konkrétní použití je pak vidět v příkladech v kapitole 4. Detailní nápovědu pak uživatel najde v manuálových stránkách příslušných příkazů.

3.3.1 Jak lze cluster využívat

Z hlediska uživatelského prostředí má uživatel v zásadě dvě možnosti, jak na clusteru počítat. Uživatelsky nejpřívětivější možností je použití specializovaného software určeného pro řešení úloh z konkrétních oblastí (například FLUENT pro modelování proudění tekutin, MSC.MARC pro strukturální dynamiku či ANSYS pro simulaci obecných fyzikálních jevů). V tomto případě uživatel pracuje v prostředí, které důvěrně zná, a o distribuci jím připravené úlohy na jednotlivé výpočetní uzly clusteru se nemusí starat, protože ji za něj automaticky zajišťuje uživatelský software.

Ve druhém případě uživatel při algoritmizaci úlohy a programátorské práci (typicky program v jazyce C nebo Fortran) úlohu rozdělí na nezávislé části, a potřebnou synchronizaci mezi nimi zajišťuje jako součást své programátorské práce s využitím speciálních knihoven pro předávání zpráv (MPI, PVM).

Dva výše uvedené případy platí pro distribuované zpracování úlohy na více uzlech. Existují samozřejmě i situace, kdy úloha může využívat pouze jeden procesor uzlu clusteru nebo běžet na jednom uzlu při využití obou procesorů za použití symetrického multiprocesingu.

Ve všech případech jsou pak připravené úlohy na clusteru zpracovávány v dávkovém režimu, což znamená, že jednotliví uživatelé vloží své úlohy do fronty na clusteru a dávkový systém zajistí jejich postupné zpracování na volných uzlech podle specifikovaných kritérií.

Dávkovým systémem zpracování úloh, který je k dispozici na PC clusterech *META Centra* je PBS — Portable Batch System. Systém PBS umožňuje interaktivní i neinteraktivní běh úloh majících různé požadavky tak, aby sdílené prostředky byly zatěžovány racionálně a rovnoměrně, a aby nedocházelo k monopolizaci zdrojů určitými úlohami nebo uživateli.

3.3.2 Koncepty PBS

Základním pojmem systému PBS z pohledu uživatele je úloha (*job*). Úlohou rozumíme jeden nebo více příbuzných UNIXovských procesů, které se podílí na řešení jednoho (obvykle uceleného) úkolu zadaného uživatelem. Pod pojem úloha tedy spadají všechny tyto procesy související s výpočtem, včetně procesů režijních.

Úlohy, zadávané uživateli, se zařazují do tzv. *front*, kde čekají na situaci vhodnou ke spuštění, a to především z hlediska zátěže systému. PBS server řídí PBS na dané skupině strojů (clusteru). PBS serveru jsou zasílány úlohy prostřednictvím příkazu *qsub*, a PBS server se také stará

o jejich spouštění. V případě popisovaných PC clusterů je PBS serverem `skirit.ics.muni.cz`.

Vlastnosti (*properties*) v PBS jsou formální zápisy určitých požadavků, které máme na stroj (stroje), na nichž má být úloha spuštěna. Pomocí vlastností lze vyjádřit požadavky např. na typ síťového propojení výpočetních uzlů nebo na fyzické umístění uzlů. Seznam vlastností lze (mimo jiné) zjistit příkazem *pbsnodes*.

Zdroje systému jsou ukazatele určitých vlastností počítače (např. paměť, zatížení, kapacita dočasného adresáře atd.), na němž určitá konkrétní instance PBS běží. Požadavky na tyto vlastnosti a zdroje (např. typu „Chci dva uzly každý se dvěma procesory“, „Chci nejméně 250 MB paměti“, nebo „Chci 15 hodin skutečného času“) jsou formulovány čárkami oddělenými řetězci, a jsou obvykle zadávány jako argument volby `-l` příkazu `qsub`.

3.3.3 Některé příkazy pro ovládání PBS

Podrobný popis příkazů najde uživatel na URL `http://meta.cesnet.cz` v oddíle *Dokumentace*, část *PBS*. Následující přehled je pouze informativním úvodem.

qsub

Příkaz, který nám umožní zaslat job ke zpracování. Je vhodné co nejpřesněji specifikovat nároky na zdroje. V případě přijetí jobu do fronty ke zpracování je výstupem příkazu identifikátoru jobu, *jobID*. Když specifikované požadavky překročí limity pro danou frontu, je výsledkem chybové hlášení. Syntaxi příkazu následují příklady použití:

```
qsub -q fronta -l resources job
qsub -q short -l nodes=1:ppn=1:brno,cput=2:00:00 \
>   job.script
qsub -q normal -l nodes=4:ppn=2:plzen,\
>   walltime=10:00:00,pvmem=200mb job.script
```

Definice jobu je skryta ve skriptu (dávce) *job.script*. Zde se jeho náplní nebudeme zabývat, konkrétní příklady použití s dostatečným popisem najde čtenář v kapitole 4. Druhým příkazem požadujeme spuštění jobu ve frontě *short* po dobu 2 hodin na některém ze strojů v Brně. Třetím příkazem požadujeme umístění jobu do fronty *normal*, s požadavkem na přidělení čtyř uzlů na plzeňském clusteru, na každém 1 procesor a 200 MB paměti po dobu 10 hodin. Po překročení limitu skutečného

času či jiných zdrojů (pro danou frontu, nebo specifikovaných jako zdroj prostřednictvím volby `-l`) je úloha násilně ukončena.

qstat

Tímto příkazem lze zjistit stav jobů, které zrovna spravuje PBS server. Typické použití příkazu je uvedeno níže, výstupem je informace o stavu jobu, dosud spotřebovaných zdrojích či dalších okolnostech.

```
qstat jobID
qstat -ans -u login
```

qdel

Umožní odstranění úlohy z fronty. Jeho syntaxe je taktéž jednoduchá:

```
qdel jobID
```

xpbs, xpbsmon

Grafický přehled o PBS serverech, frontách a úlohách, s možností zaslat úlohu do fronty, nebo jednoduché modifikace běžící úlohy, poskytuje program *xpbs*. Grafické znázornění vytížení jednotlivých uzlů v PBS umožňuje program *xpbsmon*. Protože to jsou programy s grafickým uživatelským rozhraním, a jejich ovládání by tedy mělo býti intuitivní, nebudeme se jimi zde zabývat.

KAPITOLA 4

UKÁZKY KONKRÉTNÍ PRÁCE PRO RŮZNÉ TYPY ÚLOH

Předpokládáme, že čtenář patří mezi uživatele UNIXových operačních systémů s alespoň základními znalostmi. V případě ukázek práce s výpočty pomocí komerčních balíků (FLUENT, MSC.MARC, ANSYS) předpokládáme, že je schopen běžnými způsoby úlohu připravit k finálnímu zpracování.

Poznámka: Některé uváděné jednořádkové příkazy, které se nevešly na šířku textu tohoto dokumentu, jsou rozdělené znakem „\“ na konci řádku a pokračují pak znakem „>“ na řádku následujícím.

4.1 Jednoduchý skript

Převzato z dokumentace k systému PBS, autorem Martin Černohorský.

Uvedeme si nejdříve triviální příklad, na němž demonstrujeme funkci PBS. Do fronty *short* zašleme úlohu, která zjistí stroj, na kterém je spuštěna a aktuální adresář. Záznam terminálové seance pak může vypadat například takto:

```
skirit$ cat pokus
# Jednoduchy pokus s pouzitim PBS.
# Vypiseme hostname stroje, na kterem se uloha spusti,
hostname
# pracovni adresar v nemz se spusti,
/bin/pwd
# a datum a cas:
```

```
date
# To je vse, konec skriptu "pokus".
skirit$ pwd
/home/cerno
skirit$ qsub -q short -l nodes=1:ppn=1:brno pokus
457.skirit.ics.muni.cz
skirit$ qstat 457.skirit.ics.muni.cz
Job id        Name        User        Time Use S Queue
------------- ----------- ----------- -------- - -----
457.skirit    pokus       cerno       00:00:00 R short
skirit$
```

Po skončení úlohy nalezneme v příslušných souborech standardní a standardní chybový výstup (stdout a stderr):

```
skirit$ ls -o *457
-rw-------  1 cerno      0 Jan 28 12:49 pokus.e457
-rw-------  1 cerno     60 Jan 28 12:49 pokus.o457
skirit$ cat pokus.o457
skirit2
/amd/skirit/home/cerno
Mon Jan 28 12:49:08 CET 2001
skirit$
```

4.2 Doma psaný kód

Uživatel má na clusterech samozřejmě také možnost spouštět vlastní paralelní programy napsané v běžném programovacím jazyce, například C, Fortran. Pro synchronizaci těchto programů je možné použít komunikační knihovny MPI (Message Passing Interface) či PVM (Parallel Virtual Machine). Oba tyto typy komunikačních knihoven jsou velmi často používané v oblasti paralelních výpočtů a také velmi dobře dokumentované.

Spuštění uživatelem napsaného programu si demonstrujeme na následujícím jednoduchém příkladě. Jedná se program „cpi.c“ pro výpočet čísla π.

Zdrojový kód programu:

```
/*
Program pro vypocet cisla PI
*/
```

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f(a)
double a;
{
   return (4.0 / (1.0 + a*a));
}

int main(argc,argv)
int argc;
char *argv[];
{
   int done = 0, n, myid, numprocs;
   long  i;
   double PI25DT = 3.14159265358979323846264 3;
   double mypi, pi, h, sum, x;
   double startwtime, endwtime;
   int  namelen;
   char processor_name[MPI_MAX_PROCESSOR_NAME];

   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD,&myid);
   MPI_Get_processor_name(processor_name,&namelen);

   fprintf(stdout,"Process %d of %d on %s\n",
         myid, numprocs, processor_name);

   n = 0;
   while (!done)
   {
      if (myid == 0)
      {

         if(n == 0) n=1000000; else n=0;
         startwtime = MPI_Wtime();
      }

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

```
      if (n == 0)
         done = 1;
      else
      {
         h   = 1.0 / (double) n;
         sum = 0.0;
         for (i∼= myid + 1; i∼≤ n; i∼+= numprocs)
         {
            x = h ∗ ((double)i − 0.5);
            sum += f(x);
         }
         mypi = h ∗ sum;

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

         if (myid == 0)
         {
            printf("pi is approximately %.16f, Error is %.16f\n",
                 pi, fabs(pi − PI25DT));
            endwtime = MPI_Wtime();
            printf("wall clock time = %f\n",
                 endwtime−startwtime);
         }
      }
   }
   MPI_Finalize();
   return 0;
}
```

Máme-li zdrojový kód programu, bude dalším krokem jeho přeložení. Jelikož jednotné prostředí clusterů obsahuje tzv. modulární systém uživatelských programů je nutné potřebné aplikace nejprve aktivovat pomocí příkazu `module`. Pro potřeby našeho příkladu budou všechny příkazy spouštěny v domovském adresáři.

Abychom mohli přeložit náš příklad, bude nutné zaktivovat příslušnou implementaci MPI (seznam dostupných implementací lze najít na `http://meta.cesnet.cz/software/index.cz.html`). V našem případě zvolíme implementaci LAM/MPI (Local Area Multicomputer, více informací viz `http://www.lam-mpi.org`). Modul pro LAM zaktivujeme příkazem:

```
nympha$ module add lam
```

Program přeložíme:

```
nympha$ mpicc -o cpi cpi.c
```

Dále budeme pro spuštění našeho programu potřebovat v home adresáři pomocný soubor `lamhosts` popisující topologii clusteru, tento soubor stačí vytvořit „jednou pro vždy“.

```
nympha$ cat lamhosts

# topologie clusteru nympha
#
nympha.zcu.cz
nympha1.zcu.cz cpu=2
nympha2.zcu.cz cpu=2
nympha3.zcu.cz cpu=2
nympha4.zcu.cz cpu=2
nympha5.zcu.cz cpu=2
nympha6.zcu.cz cpu=2
nympha7.zcu.cz cpu=2
nympha8.zcu.cz cpu=2
nympha9.zcu.cz cpu=2
nympha10.zcu.cz cpu=2
nympha11.zcu.cz cpu=2
nympha12.zcu.cz cpu=2
nympha13.zcu.cz cpu=2
nympha14.zcu.cz cpu=2
nympha15.zcu.cz cpu=2
nympha16.zcu.cz cpu=2
nympha$
```

Nyní připravíme spouštěcí skript do PBS pro naší úlohu. Skript `uloha` může vypadat například takto:

```
nympha$ cat uloha

#! /bin/sh
#PBS -m bae
#
# zaktivovani potrebnych modulu
module add lam
#
```

```
# spusteni LAM/MPI
lamboot -v lamhosts
#
# spusteni programu cpi na 32 procesorech
mpirun -np 32 cpi
#
# ukonceni LAM/MPI
lamhalt
# konec skriptu

nympha$
```

A nakonec připravenou úlohu pošleme do fronty příkazem:

```
nympha$ qsub -q short -l nodes=16:ppn=2:plzen \
> uloha
```

O spuštění, ukončení a případně o potížích při vykonávání úlohy bude uživatel automaticky informován systémem PBS prostřednictvím e-mailu.

Spuštění uživatelského paralelního programu používající PVM či jinou implementaci MPI stačí pouze zaktivovat příslušné moduly v zaváděcím skriptu úlohy. Postup pro spuštění pak bude zcela analogický výše popsanému příkladu. Na clusterech `minos` i `nympha` jsou nainstalovány GNU překladače a uživatel má tedy možnost psát své programy v jazycích C, C++ a F77. Pro psaní uživatelských programů je k dispozici řada vývojových nástrojů nástrojů, uveďme např. *vampir* — nástroj pro ladění MPI programů či debugger *totalview*.

4.3 FLUENT

Na PC clusterech v Plzni je instalován FLUENT 6.0.20. Nejsou záměrně instalovány prostředky pro přípravu úloh, tj. preprocesory GAMBIT a TGrid — clustery by měly sloužit zejména pro finální zpracování úloh, tj. pro vlastní výpočty.

FLUENT připojíme příkazem `module load fluent60`. Tento příkaz je nutné umístit do souboru `.bashrc` v domovském adresáři v případě, že chceme aplikaci používat paralelně na více výpočetních uzlech.

Na ZČU máme k dispozici 30 plovoucích licencí FLUENTu. Zde je třeba připomenout, že každé spuštění FLUENTu vyžaduje tolik volných licencí, kolik výpočetních uzlů (tj. z hlediska FLUENTu procesorů) je pro zpracování úlohy použito.

Vzhledem k omezenému počtu licencí je třeba vždy specifikovat jako parametr příkazu `qsub` vlastnost *fluent* s příslušným počtem licencí (procesorů), které FLUENT bude potřebovat. Viz příklady níže.

K software FLUENT má přístup pouze omezená skupina uživatelů. O zařazení do této skupiny (v případě, že nejste jejími členy) žádejte na adrese `zsc@service.zcu.cz`.

4.3.1 FLUENT na jednom uzlu stroje

Předpokládejme, že v adresáři `testy` máme připravenou úlohu pro FLUENT. Při používání systému PBS je nutné FLUENT spouštět v dávkovém módu bez grafického uživatelského rozhraní. Soubor `fluent.input` je vlastně seznamem příkazů, které má FLUENT vykonávat a jeho obsah může vypadat následovně:

```
file/read-case jesterka.cas
solve/initialize initialize-flow
solve/iterate 1000
file/binary-files yes
file/write-case-data jesterka2.cas
exit
```

Tento vstupní soubor pro FLUENT zajistí načtení souboru s popisem úlohy `jesterka.cas`, inicializaci řešení a výpočet po dobu 1000 iterací. Pak jsou výsledná data uložena v binární formě (je rychlejší a úspornější) do pracovního adresáře. Dále se zde detailněji popisem vstupních souborů nebudeme zabývat.

V adresáři `testy` se dále nachází soubor `jesterka.cas` a spustitelný (pozor na příslušná práva!) skript `fl.node` pro PBS. Jeho obsah je následující:

```
# zacatek
#PBS -m ae
# zmena pracovniho adresare
cd ~/testy
# spusteni FLUENTu
fluent 3d -g -i fluent.input > fluent.output
# konec
```

Jako v každém shellovském skriptu, text za dvojkřížkem (#) slouží jako komentář. Výjimku představuje konstrukce `#PBS`, za kterou můžeme vložit argumenty příkazové řádky příkazu *qsub*. V tomto případě si necháme poslat zprávu elektronickou poštou v okamžiku ukončení nebo zrušení

úlohy (volba `-m ae`). Skript provede přesun do adresáře `~/testy`, kde spustí 3D verzi FLUENTu bez grafického prostředí. Vstupní soubor se seznamem příkazů je specifikován parametrem `-i fluent.input` (je vhodnější než `< fluent.input`), výstup z běhu programu je pak přesměrován do souboru `fluent.output`.

Job umístíme do fronty *normal* následujícím příkazem. Předpokládáme, že úloha zabere 500 MB paměti a do 10 hodin času, výpočet probíhá na jednom uzlu plzeňské části clusteru.

```
qsub -q normal -l nodes=1:ppn=1:plzen,walltime= \
> 10:00:00,pvmem=300mb -l fluent=1 fl.node
```

Nyní tu samou úlohu pustíme na jednom uzlu na dvou procesorech. Předpokládejme, že definice úlohy nám to umožní (viz další kapitola). Lehce se změní pouze skript `fl.node` pro PBS:

```
# zacatek
#PBS -m ae
# zmena pracovniho adresare
cd ~/testy
# spusteni FLUENTu
fluent 3d -g -psmpi -t2 -i fluent.input > fluent.out
# konec
```

Řetězec `-psmpi -t2` zajistí spuštění paralelní verze FLUENTu s využitím dvou procesorů. Příkaz pro umístění úlohy do fronty je pak následující:

```
qsub -q normal -l nodes=1:ppn=2:plzen,walltime= \
> 7:00:00,pvmem=300mb -l fluent=2 fl.node
```

4.3.2 Paralelní síťová verze FLUENTu

Jediná cesta, jak spočítat některé náročné úlohy v reálném čase, je používat FLUENT v síťové paralelní verzi. Zda vůbec a s jakou měrou využití dostupných prostředků, záleží na rozmyslu uživatele a konkrétní úloze.

Velice zhruba řečeno, každých 100 tisíc buněk výpočetní sítě úlohy zabere asi 100 MB paměti. Při rozdělení úlohy na více strojů se pak potřebná paměť dělí (v podstatě lineárně) mezi jednotlivé stroje. Výpočetní čas klesá (v rozumných mezích) také lineárně s rostoucím počtem použitých procesorů.

Přechod od jednoprocesorového zpracování úlohy k paralelnímu výpočtu nemusí být nijak složitý, záleží to ale na konkrétní úloze (použitých fyzikálních modelech, typu výpočetní sítě apod.) Ve většině případů

se uživatel nemusí o nic starat, FLUENT umí úlohy automaticky načíst i do paralelní verze, kdy podle předem zadaných kritérií rozděluje části výpočetních sítí přímo na jednotlivé výpočetní uzly.

Výjimku představují úlohy, kde je použito modelu *sliding mesh*, nebo kde jsou definovány nekonformní *grid-interfaces*, na kterých chceme provádět adaptaci během výpočtu. Pak nezbývá, než výpočetní síť „ručně" rozdělit v jednoprocesorové verzi FLUENTu na požadovaný počet *partitions* například pomocí příkazu `parallel/partition`. Tolik tedy manuál. Jak se ale ukazuje, je vhodné rozdělit výpočetní síť i v případě, že obsahuje jakékoli nekonformní *grid-interfaces* (i když na nich nechceme provádět adaptaci).

Dosavadní znalosti a zkušenosti dále ukazují, že nic nezkazíme, provedeme-li dělení výpočetní sítě v každém případě. Vhodné je rozdělení na například 12 *partitions*, pak můžeme bez obav pro výpočet použít podle aktuální situace 2, 3, 4, 6 nebo 12 procesorů (ehm, s těmi 12 procesory pozor, jsme omezeni počtem volných licencí), FLUENT rovnoměrně načte podle potřeby na každý výpočetní uzel více *partitions*.

Předpokládejme, že v adresáři `testy` máme opět připravenou úlohu `jesterka.cas` a soubor `fluent.input`. Nyní chceme spustit paralelní FLUENT na části PC clusteru. K tomu nám pomůže následující skript `fl.net` pro PBS:

```
# zacatek
#PBS -m ae
#
JOBID=‘echo $PBS_JOBID  | awk -F . ’{print $1}’’
mkdir /scratch/knourek/$JOBID
#
cd ~/testy
cp jesterka.cas /scratch/knourek/$JOBID
cp fluent.input /scratch/knourek/$JOBID
cd /scratch/knourek/$JOBID
#
cat $PBS_NODEFILE > fluent.hostlist.$JOBID
#
fluent 3d -g -cnf=fluent.hostlist.$JOBID -pnet -t0 \
>   -i fluent.input > fluent.output.$JOBID
#
mv * ~/testy/
cd
rm -rf /scratch/knourek/$JOBID
# konec
```

Následujícím příkazem zařadíme job do fronty *normal*, požadujeme alokaci 2 uzlů, na každém 2 procesory.

```
qsub -q normal -l nodes=2:ppn=2:plzen,walltime= \
> 4:00:00,pvmem=300mb -l fluent=4 fl.net
```

Job má v systému PBS identifikaci složenou z čísla jobu a řetězce (například `10457.skirit.ics.muni.cz`). Výše uvedený skript nejprve z tohoto identifikátoru vyseparuje číslo úlohy (tedy 10457), bude se nám hodit. Jeho hodnota je uložena v proměnné `JOBID`. Dále se vytvoří na stroji, kde úloha běží pracovní adresář `/scratch/knourek/10457` na lokálním disku `/scratch`. Toto se hodí pro případ, že FLUENT často přepisuje či zapisuje data — nejsme vázáni na pomalejší NFS domovské adresáře. Do pracovního adresáře jsou pak přesunuta potřebná data.

Dále je zkopírován seznam alokovaných uzlů, který je v souboru, jehož umístění je dáno obsahem systémové proměnné `$PBS_NODEFILE`, do (textového) souboru v pracovním adresáři (`fluent.hostlist.10457`). Následuje spuštění FLUENTu v paralelní síťové verzi. Seznam uzlů pro výpočet je dán obsahem souboru `fluent.hostlist.10457`. Všimněte si, že není třeba ve skriptu určovat, na kolika procesorech a v jaké konfiguraci uzlů bude FLUENT spuštěn — to určí systém PBS. Proto je třeba vždy v příkazu *qsub* specifikovat, kolik uzlů a procesorů na uzel pro svůj job požadujeme. Po skončení výpočtu jsou všechna data z pracovního adresáře přesunuta zpět do adresáře `~/testy`.

4.3.3 Použití dalších síťových komunikátorů

FLUENT je kromě výše uvedené paralelní verze, která komunikuje pomocí standardního síťového rozhraní, také ve verzích, které umí používat *Network MPI* a *Myrinet MPI*, tedy výkonnější formy komunikace. Myrinet je instalován na clusteru *nympha* v Plzni, Network MPI je možno použít na obou clusterech.

Použití *Network MPI* a *Myrinet MPI* přináší rychlejší start FLUENTu a rychlejší načítání úlohy a distribuování dat na jednotlivé výpočetní uzly.

Následuje příklad skriptu pro použití Network MPI, rozdílné je použití příkazu spouštějícího FLUENT.

```
# zacatek
#PBS -m ae
#
cd ~/testy
```

```
#
NP=‘wc -l $PBS_NODEFILE | awk ’{print $1}’’
cat $PBS_NODEFILE > fluent.hostlist
#
fluent 3d -g -pnmpi -t$NP -cnf=fluent.hostlist \
>   -i fluent.input > fluent.out
# konec
```

Příklad skriptu pro použití Myrinetu je uveden níže. Rozdílná je struktura konfiguračního souboru *gm-file* se seznamem výpočetních uzlů. Ten obsahuje kromě seznamu výpočetních uzlů, na nichž úloha běží, ještě číslo portu myrinetové karty (2 nebo 4) a v úvodu celkový počet použitých procesorů.

```
# zacatek
#PBS -m ae
#
cd ~/testy
#
NP=‘wc -l $PBS_NODEFILE | awk ’{print $1}’’
cat $NP > gm-file
cat $PBS_NODEFILE | awk ’{if(old!=$1) n=2; \
>  print $1, n; old=$1; n+=2;}’ >> gm-file
export GMPICONF=gm-file
#
fluent 3d -g -pgmpi -t$NP -i fluent.input > fluent.out
# konec
```

Do fronty zašleme úlohy podobně jako v předchozích případech. Nesmíme zapomenout, že myrinet je pouze na některých clusterech (například na clusteru *nympha*).

4.4 MSC.MARC

Na PC clusterech v Plzni je instalován MSC.MARC 2001. Není záměrně instalován preprocesor MSC/Mentat — clustery by měly sloužit zejména pro finální zpracování úloh, tj. pro vlastní výpočty.

MSC.MARC připojíme příkazem `module load marc2001`. Tento příkaz je nutné umístit do souboru `.bashrc` v domovském adresáři v případě, že chceme aplikaci používat paralelně na více výpočetních uzlech.

K dispozici je 7 plovoucích licencí MSC.MARCu a 24 licencí pro paralelní běh kódu. V případě paralelního běhu se alokuje 1 základní licence

MSC.MARCu a počet přídavných licencí pro paralelní běh odpovídající počtu procesorů. Vzhledem k převaze licencí pro paralelní běh je vhodné úlohy počítat pokud možno paralelně. V MSC.MARCu je ale v paralelním běhu omezeno použití některých řešičů či fyzikálních modelů.

Vzhledem k omezenému počtu licencí je třeba vždy specifikovat jako parametr příkazu `qsub` vlastnost *marc* s příslušným počtem licencí (procesorů), které FLUENT bude potřebovat. Viz příklady níže.

K software MSC.MARC má přístup pouze omezená skupina uživatelů. O zařazení do této skupiny (v případě, že nejste jejími členy) žádejte na adrese `zsc@service.zcu.cz`.

4.4.1 Jednoprocesorová verze MSC.MARCu

Nechť se v našem adresáři `marc` nachází soubor `test.dat`, který tvoří vstupní data pro jednoprocesorovou verzi MSC.MARCu. Níže je výpis scriptu `marc.node` pro spuštění úlohy prostřednictvím PBS.

```
# zacatek
#PBS -m ae
#
module load marc2001
#
cd ~/marc
mkdir /scratch/knourek/marc
cp test.dat /scratch/knourek/marc
cd /scratch/knourek/marc
#
marc2001 -jid test -v no -b no > output.log
#
mv -f * ~/marc
# konec
```

Protože MSC.MARC většinou zapisuje průběžně nemalý objem dat do pracovního adresáře, je za něj vhodné zvolit lokální diskový prostor `/scratch` na výpočetním uzlu. Parametry příkazu `marc2001` jsou následující: `-v no` pro neinteraktivní spuštění výpočtu, `-b no` pro běh na popředí, následuje přesměrování standardního výstupu do souboru `output.log`. Po dokončení výpočtu se výsledná data přesunou zpět do adresáře `~/marc`.

Následujícím příkazem zařadíme job `marc.node` do fronty *normal*:

```
qsub -q normal -l nodes=1:ppn=1:plzen,walltime= \
> 4:00:00,pvmem=300mb -l marc=1 marc.node
```

4.4.2 Paralelní verze MSC.MARCu

Přechod od jednoprocesorové úohy k paralelní není nijak složitý. Patřičné změny je třeba zadat v preprocesoru MSC.MENTAT. Předpokládejme, že máme popis úlohy v souboru `test.mud`. Následuje popis seance v MSC.MENTATu, kde vytvoříme popis úlohy pro běh na 4 procesorech. Je uveden záznam příkazů zadávaných z příkazové řádky programu, je jasné, že mají i svoje ekvivalenty při ovládání pomocí GUI MSC.MENTATu.

```
open_model test.mud
domains_generate 4
job_option dynamic:newmark
update_job
job_option parallel:on
job_write_input yes
```

Po ukončení MSC.MENTATu vidíme, že v pracovním adresáři byly vytvořeny soubory `test_job1.dat`, `1test_job1.dat`, `2test_job1.dat`, `3test_job1.dat` a `4test_job1.dat`. Ty jsou zdrojovými soubory pro paralelní běh MSC.MARCu. Všiměte si, že MSC.MARC neumí v paralelním běhu řešit explicitní dynamiku (mimo jiného), je proto zvolen implicitní řešič.

Výše uvedené soubory nechť se nacházejí v adresáři `marc` v našem domovském adresáři. Následuje script `marc.net` pro paralelní zpracování úlohy pomocí PBS na čtyřech výpočetních uzlech clusteru. V souboru `hosts.m` je uvedena specifikace výpočetních uzlů, které jsou pro zpracování úlohy vybrány PBS. Struktura souboru může být složitější, pro detaily nahlédněte do manuálů systému MSC.MARC.

```
# zacatek
#PBS -m ae
cd ~/marc
cp *test_job1.dat /scratch/knourek
cd /scratch/knourek
cat $PBS_NODEFILE | awk '{print $1 " 1"}' > hosts.m
NP='wc -l hosts.m | awk '{print $1}''
#
marc2001 -jid test -v no -b no -nproc $NP \
>  -host hosts.m > output.log
#
mv -f * ~/marc
# konec
```

Jako pracovní adresář je zvolen opět pro každý výpočetní uzel lokální `/scratch/knourek`. MSC.MARC automaticky před započetím výpočtu přenese na pracovní adresáře (`/scratch/knourek`) jednotlivých výpočetních uzlů příslušné vstupní soubory a po skončení výpočtu je shromáždí zpět. Příkaz, který umístí job do PBS fronty *normal* je:

```
qsub -q normal -l nodes=4:ppn=1:plzen,walltime= \
> 6:00:00,pvmem=700mb -l marc=4 marc.net
```

V případě, že použijeme pouze dvou procesorů na jednom výpočetním uzlu clusteru, je paralelní spuštění úlohy jednodušší. Předpokládáme, že v preprocesoru MSC.MENTAT byla úloha rozdělena na 2 části a v adresáři `~/marc` se tedy nacházejí soubory `test_job1.dat`, `1test_job1.dat` a `2test_job1.dat`. Následuje skript `marc.2` pro PBS:

```
# zacatek
#PBS -m ae
#
module load marc2001
cd ~/marc
mkdir /scratch/knourek/marc
cp *test_job1.dat /scratch/knourek/marc
cd /scratch/knourek/marc
#
marc2001 -jid test -v no -b no -nproc 2 > output.log
#
mv -f * ~/marc
# konec
```

Úlohu pak umístíme ke zpracování do PBS fronty příkazem:

```
qsub -q normal -l nodes=1:ppn=2:plzen,walltime= \
> 4:00:00,pvmem=300mb -l fluent=2 marc.2
```

4.5 ANSYS

Na PC clusterech je instalován ANSYS 6.1. Je dostupný modul ANSYSRF v počtu 30 plovoucích licencí. Přídavné licence pro paralelní běh nejsou zakoupeny, takže uživatelé jsou omezeni na používání ANSYSu pouze v jednoprocesorovém běhu.

ANSYS 6.1 připojíme příkazem `module load ansys61`.

Vzhledem k omezenému počtu licencí je třeba vždy specifikovat jako parametr příkazu `qsub` vlastnost *ansys* s příslušným počtem licencí (procesorů), které ANSYS bude potřebovat (zde vždy 1).

Následující script `ansys.node` pro PBS zařídí spustění výpočtu v ANSYSu, předpokládejme, že máme v podadresáři `ansys` našeho domovského adresáře úlohu popsanou v souboru `test1.dat`.

```
# zacatek
#PBS -m ae
#
cd ~/ansys
#
ansys61 -b -p ansysrf < test1.dat > output.log
# konec
```

Následujícím příkazem pak předáme úlohu ke zpracování PBS serveru:

```
qsub -q normal -l nodes=1:ppn=1:plzen,walltime= \
> 4:00:00,pvmem=300mb -l ansys=1 ansys.node
```

KAPITOLA 5

ZÁVĚREM

Doufáme, že tato publikace přispěla k informovanosti o nových možnostech výkonného počítání a seznámila uživatele s konkrétními způsoby používání PC clusterů.

5.1 Dostupná dokumentace

Hlavními body pro získání informací jsou tato dvě URL:

http://meta.cesnet.cz
http://zsc.zcu.cz

Najdete zde popisy softwarových produktů, příklady, návody a podrobnou dokumentaci, odkazy na další stránky a konkrétní kontaktní adresy pro řešení vašich problémů.

LITERATURA

[1] Ročenka *METACentra* pro roky 2000–2001, kterou najdete na URL: `http://meta.cesnet.cz/yearbooks/2001_cz.pdf`

[2] Dokumentace příslušných SW produktů

Místo pro Vaše poznámky