MetaCentrum

Plánování úloh – otázky a odpovědi

Miroslav Ruda

CESNET

Praha, seminář MetaCentra, 2010

Obsah

- odpovědi na časté otázky z RT
  - proč se mi úloha nechce pustit?
  - úloha se pustila, co dál?
  - dávkový systém se mi moc nehodí, co mám dělat?
- souvislost se souborovými systémy, AA mechanismy, . . .
- změny plánovače, pilotní úlohy
- cokoli dalšího
- . . . a průběžně se ptejte

K čemu je plánování?

- vždyť přece naskládáme úlohy za sebe a pak je spustíme. . . ?
- bohužel ne!
- spousta omezujících podmínek:
  - maximální doba běhu úlohy (short/normal/long)
  - počet strojů a procesorů
  - vlastnosti strojů
  - velikost potřebné paměti
  - licence
  - velikosti místa na pomocném disku /scratch
  - . . .
- podmínky vyjadřujeme při zadání úlohy

Proč se nespustila moje úloha?

- pokud nelze podmínky nikdy splnit, uživatel se to dozví hned
- ale někdy se úloha „pouze nespustí“
  - protože to není spravedlivé
    - dle aktuální definice spravedlnosti ;)
  - protože se dělá prostor pro paralelní úlohu
  - protože uzel používá někdo s vyšší prioritou
  - protože uzel nepřijímá nové úlohy
  - protože požaduji kombinaci, která právě teď neexistuje

Úvod k plánování I

Rozdílné požadavky:

- uživatel: potřebuji úlohu spustit co nejdříve
- správce: stroje musí být co nejvíce vytížené
- poučený uživatel: ale musí to být spravedlivé
- vlastník clusteru: moje úlohy nejdříve

Požadavky vyjádřené např. jako:

- propustnost (počet dokončených úloh)
- zpoždění úlohy (čas_spuštění - čas_přijetí)
- průměrná doba odezvy (čas_dokončení - čas_přijetí)
- vážené hodnoty z předchozích, . . .

Úvod k plánování II

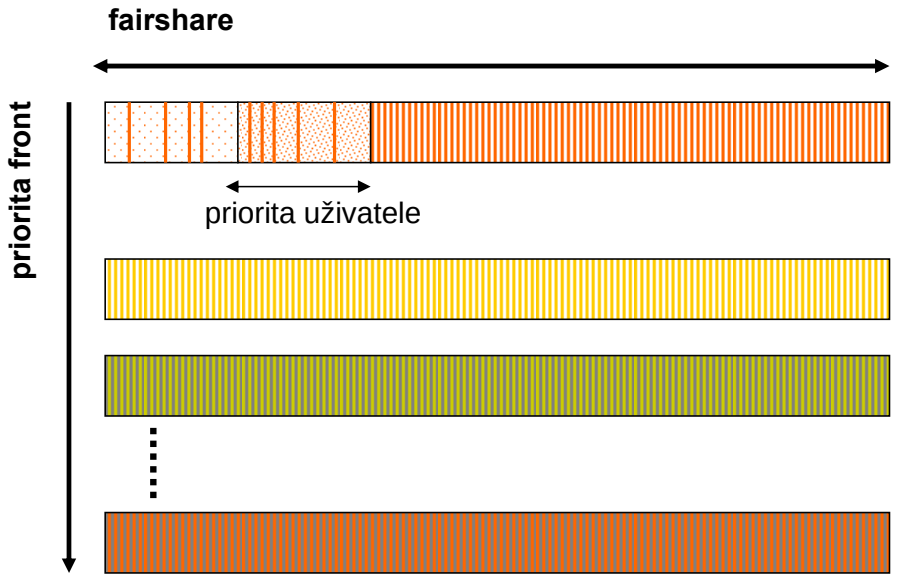
Rychlost samotného naplánování:

- v současnosti plánujeme cca 140.000 úloh za měsíc
  - 3 za minutu
- plány se průběžně mění (výpadky strojů, nové úlohy)
  - proto je obtížné dopředu určit, kdy bude úloha spuštěna

Úvod k plánování III

Fronty:

- dávkové úlohy řadíme do front
  - zde ale „fronta“ neznamená First-In, First-Out
- fronty definují
  - prioritu (řazení front)
  - práva (kdo kam smí)
  - omezující podmínky (jen na některé stroje, jen některé úlohy)



Implementace v MetaCentru

Portable Batch System (PBS):

- fronty se prochází podle priority front
  - nejprve se zkusí spustit všechny úlohy z jedné fronty
  - pak se jde na frontu s menší prioritou
- řazení v jedné frontě podle „spravedlivosti“
- u všech úloh se pak hledají zdroje, které úloze vyhovují
  - jsou volné (nebo se dají vytvořit – virtualizace): úloha se pustí
  - jsou obsazené: úloha čeká, mohou se pro ni zdroje rezervovat
  - nejsou: úloha se odmítne (zdroje nikdy nebudou) nebo čeká (mohou se objevit)

Priority front v MetaCentru I

Priority front vyjadřují:

- obtížnost při plánování (paralelní úlohy)
- prioritní přístup vlastníka zdroje
- váhu uživatele (více publikací = lepší priorita)
- plánovací strategii (krátké versus dlouhé úlohy)

Vlastnící zdrojů, kteří poskytují svůj cluster do MetaCentra:

- přednostní právo podle dohody
  - omezení délky nebo počtu cizích úloh
  - možnost pozastavit cizí úlohy

Priority front v MetaCentru II

Fronta `privileged`

- naše vyjádření poděkování uživatelům, kteří nás citují
- podle zaregistrovaných publikací
- prozatím se započítávají napořád, bude nutno domyslet
  - zvýšení váhy ve fairshare

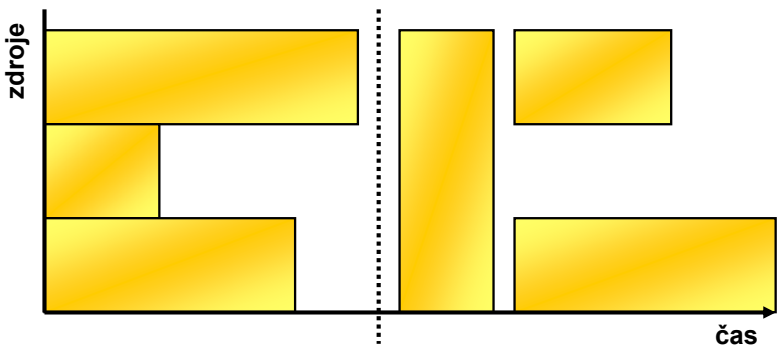
Spravedlnost

- jeden uživatel nesmí obsadit neúměrně moc zdrojů
  - existují úlohy se kterými soupeří?
- limity na počet úloh v systému, ve frontě apod.
- fairshare podle PBS
  - čas propočítaný uživatelem za poslední měsíc
    - váha v čase klesá
  - úlohy v jedné frontě jsou řazeny podle této hodnoty
- spravedlivé vyvážení krátkých a dlouhých úloh pomocí
  - uzlů přiřazených jednotlivým frontám
  - omezení počtu úloh z jednotlivých front
- jednoprocesorové versus paralelní úlohy

Plánování paralelních úloh

Úpravy plánovacího algoritmu zvýhodňující paralelní úlohy

- pokud úloha dlouho čeká (strádá), jsou pro ni rezervovány stroje, které jí vyhovují
  - žádné další úlohy se na takové uzly nenaplánují (až na priority)
- omezení počtu takto preferovaných úloh (limit fronty, limit strádajících úloh)
- při plánování by výrazně pomohly přesnější odhady délky běhu úlohy
  - může se zvyšovat náš tlak na přesnější odhady, údaje o starších úlohách často napoví



Uzly v údržbě

- s rostoucím počtem uzlů roste i počet nutných zásahů
- doinstalování, upgrade, reboot často počkají „na vhodnou chvíli“
  - často je možné nechat doběhnout rozpočítané úlohy
- uzly jsou vidět v PBS
  - úlohy na ně směřované se nevrací jako nemožné
  - nepřijímají nové úlohy
- snaha podobné úkony automatizovat, rozložit v čase tak, aby vždy alespoň část MetaCentra nové úlohy spouštěla
- vždy musí být možno úlohu zadat

Neexistující kombinace

Nejčastější problémy:

- uživatel si nepožádal o účet na dalším clusteru
  - přecházíme na nastavení, že většina clusterů se přiděluje automaticky
  - stále může platit pro clustery s omezeným přístupem
- uzly jsou rezervované pro školení, údržbu, . . .
- požadavek na počítané zdroje (paměť, scratch) přesahují možnost vybraného clusteru
- neexistující kombinace vlastností uzlů (infiniband:praha) systém odhalí, ale počítané zdroje ne
- syntaxe není hezká **-l nodes=1:praha:mem=1g** versus -l nodes=1:praha,mem=1g

Životní cyklus úlohy I

Zadaná

- příkaz qsub, požadavek na počet a typ uzlů, procesorů, paměti
- možno zadat z libovolného front-endu
- výkonný kód ve formě shellového skriptu

Ve frontě

- plánovač ji opakovaně zkouší naplánovat

Životní cyklus úlohy II

Stage-in

- PBS přenese případná vstupní data specifikovaná přes `-Wstagein`

Běžící

- zadaný skript je spuštěn na prvním přiděleném uzlu
- ideálně nad lokálním scratch prostorem
- NFS/AFS je vhodné použít jen pro nakopírování dat a programů

Životní cyklus úlohy III

Konec výpočtu

- na konci je možné uložit výsledek do NFS/AFS
- je nutné promazat dočasné soubory na lokálním disku
- po skončení skriptu ještě PBS přenese soubory specifikované přes `-Wstageout`
- pokud není řečeno jinak, přenese výstup na stroj, kde byla úloha zadána. Je možno to změnit přes `-o cesta` `-e cesta`

Ukončená

- zůstává vidět na několik hodin ve výpisu `qstat`

Úlohy: přístup k paralelismu

- jednoprocesorové
- paralelismus ve sdílené paměti
  - thready, OpenMP
- „ruční cluster buster“ paralelismus
- paralelizační framework
  - MPI, Boinc, . . .
- skrytý v aplikaci
  - např. Matlab – podporuje prakticky všechny zmíněné možnosti

MPI

Jak se spouští MPI úloha:

- PBS spustí skript na prvním přiděleném uzlu
  - seznam dalších uzlů je v souboru $PBS_NODEFILE
- je na uživateli, jak si spustí výpočet na dalších uzlech
- lze použít `ssh`/`rsh`, ale není to vhodné pro výpočty
  - procesy utečou z dohledu PBS
- je vhodné využít spouštění přes PBS MOM
  - pbsdsh – obdoba `rsh`
  - mpiexec – náhrada klasického `mpirun`
- `mpiexec` musí být kompilován s naší PBS
- různé druhy MPI mají trochu odlišný mpiexec
  - některé si nejdříve potřebují uzly „pospojovat“

MPI příklady I

```
#pozadam si o dva uzly, interaktivne, na kazdem jen
#jeden procesor, aby to bylo nazornejsi
$ qsub -I -lnodes=2:ppn=1
...

#krasny jednoduchy priklad je cpi.c
$ cd tmp
$ cp /software/mpich-1.2.7/
      amd64_linux26/ch_p4/examples/cpi.c .

#tady ulozi PBS seznam uzlu ktera nam pridelila
$ cat $PBS_NODEFILE
skirit50-1.ics.muni.cz
skirit51-1.ics.muni.cz
```

MPI příklady II

```
#cesta pres OpenMPI, neni treba pouzit $PBS_NODEFILE,
#je kompilovane s PBS

$ module add openmpi-1.4.1
$ mpicc cpi.c -o cpi.openmpi
$ mpiexec -np 1 ./cpi.openmpi
Process 0 on skirit50-1.ics.muni.cz
pi is approximately 3.1416009869231254,
wall clock time = 0.000105

$ mpiexec -np 2 ./cpi.openmpi
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit51-1.ics.muni.cz
pi is approximately 3.1416009869231241,
wall clock time = 0.083386
```

MPI příklady III

```
#OpenMPI z OFED balicku, kde je podpora infinibandu,
#ale ne PBS

$ module del openmpi-1.4.1
$ module add ofed-1.3-openmpi
$ mpicc cpi.c -o cpi.ofed.openmpi
#mpiexec nevi o dalsich uzlech, takze pouzije znovu
#stejny stroj

$ mpiexec -np 2 ./cpi.ofed.openmpi
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit50-1.ics.muni.cz
pi is approximately 3.1416009869231241,
wall clock time = 0.000968
```

MPI příklady IV

```
#OpenMPI z OFED balicku, seznam uzlu se musi zadat
$ mpiexec -np 2 -hostfile $PBS_NODEFILE
./cpi.ofed.openmpi
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit51-1.ics.muni.cz
pi is approximately 3.1416009869231241,
wall clock time = 0.011491
$ module del openmpi-1.4.1
```

MPI příklady V

```
#Starsi MPICH, kde mame vlastni mpiexec
$ module add mpich-p4
$ mpicc cpi.c -o cpi.mpichp4
$ mpiexec -np 2 ./cpi.mpichp4
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit51-1.ics.muni.cz
pi is approximately 3.1416009869231241,
wall clock time = 0.003906
$ mpirun -machinefile
  $PBS_NODEFILE -np 2 ./cpi.mpichp4
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit51-1.ics.muni.cz
pi is approximately 3.1416009869231241,
wall clock time = 0.000000
$ module del mpich-p4
```

MPI příklady VI

```
#nektere MPI si nejdriv potrebuje postavit svet,
#napr. MVAPICH2
$ module add ofed-1.3-mvapich2
$ mpicc cpi.c -o cpi.mvapich2
$ mpdboot -n 2 -f $PBS_NODEFILE
$ mpiexec -np 2 ./cpi.mvapich2
pi is approximately 3.1416009869231241,
wall clock time = 0.000488
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit51-1.ics.muni.cz

$ mpdallexit
$ module del ofed-1.3-mvapich2
```

MPI příklady VII

```
#podobne LAM
$ module add lam-7.1
$ mpicc -L /software/pbs-7.0.0/lib cpi.c -o cpi.lam
$ lamboot
LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University

$ mpiexec -np 2 ./cpi.lam
Process 0 on skirit50-1.ics.muni.cz
Process 1 on skirit51-1.ics.muni.cz
pi is approximately 3.1416009869231241,
wall clock time = 0.000819

$ lamhalt
```

Kerberos a obnovování lístků

Kerberos

- systém pro jednotná hesla přes celou organizaci (centrální kontrolní bod)
- bezpečný i v prostředí přes celé MetaCentrum
- integrovaný s filesystémy, které chceme používat pro přístup k datům (AFS, NFSv4)
- omezená platnost lístku

PBS a paralelní úlohy

- PBS zařídí a obnovuje lístky na prvním uzlu
  - v Torque bude udržovat lístky na všech uzlech
- ssh/rsh lístky používá a přenáší, neprodlužuje
- komunikace přes PBS MOM Kerberos nepotřebuje, ani nepřenáší

Kerberos FAQ

Kerberos mimo PBS (front-end) přes renewable lístky

```
$ kinit -l 1m -r 1d
ruda@META's Password:
odin$ klist
Credentials cache: FILE:/tmp/krb5cc_11623_tGEAn9
        Principal: ruda@META
  Issued           Expires          Principal
Sep 29 16:30:09  Sep 29 16:31:09  krbtgt/META@META
...
odin$ kinit -R
odin$
```

Pozor na screen, je nutno před tím použít pagsh a zde udělat separátní lístek

Jaké zdroje PBS plánuje

Nejen podle uzlů (počet, vlastnosti), procesorů a paměti

- můžeme použít lokální zdroje (na každém uzlu)
- globální zdroje
  - např. licence Matlab, Fluent, Ansys, Marc
- uměli bychom hlídat např. maximálně dvě úlohy s požadavkem na GPU na uzel
  - ohlídání fyzického přístupu ke kartě ale ne

Hlídání použité paměti

- používejte `-l mem`, default je hodně malý (400mb)
- je dobré požadavek nepřehánět, bude větší šance že se najde volný stroj
- pro skončené úlohy je maximální použitá velikost vidět v historii (qstat, webové rozhraní)
- PBS hlídá využití, aplikaci používající víc **zabíjí**

Scratch I

Scratch, volné místo na lokálním, dočasném disku

- začíná být palčivý problém, budeme muset řešit lépe
- je možné použít -l scratchsize=10G
  - to je porovnáno proti aktuálnímu volnému místu
  - neřeší problém dvou souběžně spuštěných úloh
- bude nutné zavést -l scratch=10G
  - které se opět bude sčítat na serveru
- obtížnější hlídání obsazení na uzlu
  - dvě souběžné úlohy jednoho uživatele
  - zřejmě bude nutné hlídat přes quotas

Scratch II

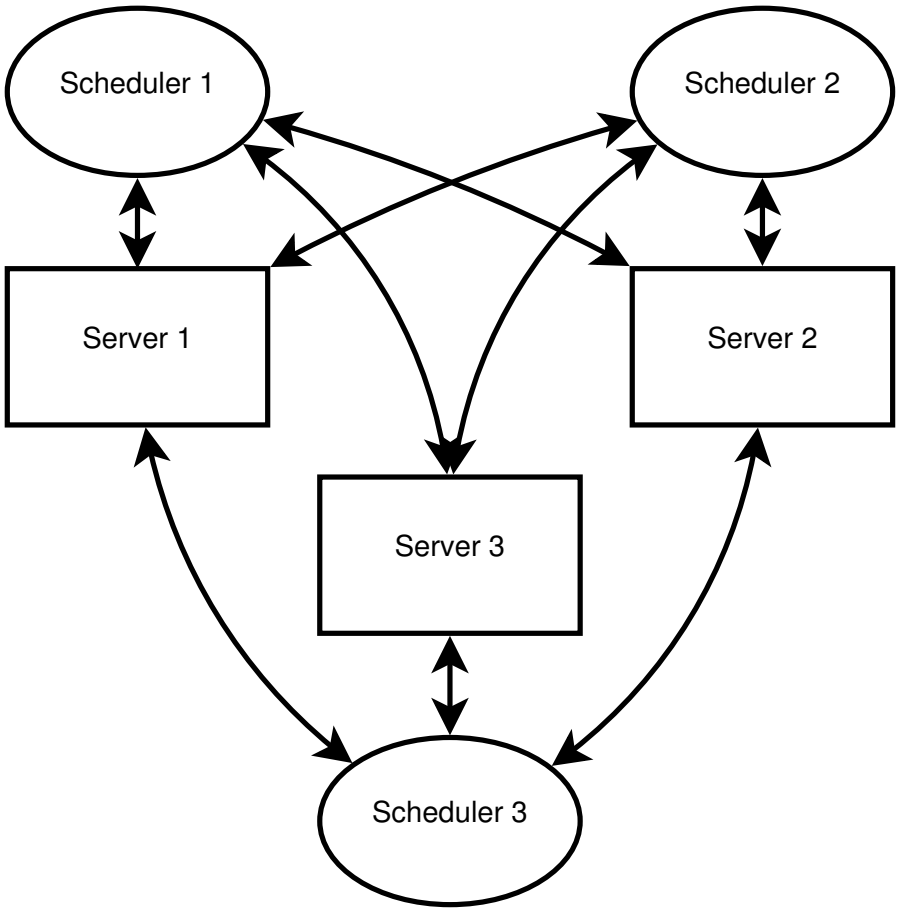
Uklízení

- po doběhnutí úlohy by měla být smazána všechna data
  - lze snadno vyřešit ve spuštěném skriptu, problémy nastávají při pádu úlohy
- zpřísnění pravidel pro automatické čištění
  - pokud je zaplněno více jak 10 % scratch prostoru, smaž vše, co je starší 14 dní
  - v případě zaplnění nad 50 % po týdnu
  - neplatí pro uživatele s běžící úlohou
- další postupné kroky
  - jiná pravidla pro `/scratch/login` a `/scratch/login/PBSJOBID`
  - automatické přenesení `/scratch/login/PBSJOBID` z lokálního disku do úložiště

Další vývoj – Torque I

- MetaCentrum roste
  - počet clusterů, úloh, připojených organizací
- nevýhody centrální instalace začínají převažovat nad výhodami
- směrem ke kooperativním plánovačům, několika instalacím Torque
- zachování globálních vlastností
  - fairshare, účtování
  - viditelnost všech uzlů a úloh pro plánovače
  - možnost použít libovolný front-end pro správu úloh

Další vývoj – Torque II



- server spravuje jen lokální uzly (stabilita v případě výpadku)
- plánovač komunikuje i s ostatními servery (mají přehled o celém gridu)
- úlohy se mohou přesouvat mezi servery
- složitější podpora paralelních úloh přes více center

V dohledné době bude nasazeno na experimentální cluster

Další vývoj - jiné typy úloh

Podporované

- interaktivní – základní podpora v PBS, více přes virtualizaci
- požadavky na odlišné prostředí – virtuální clustery
- pilotní úlohy – další průsvitka

Postupně se budeme věnovat i dalším

- backfill – velké množství úloh s nejnižší prioritou
- BOINC – knihovna pro vývoj aplikací využívajících volných cyklů (seti@home)
- Hadoop – prostředí pro vývoj aplikací od Apache, zejména MapReduce úlohy, přichází i s vlastním filesystémem

Pilotní úlohy I

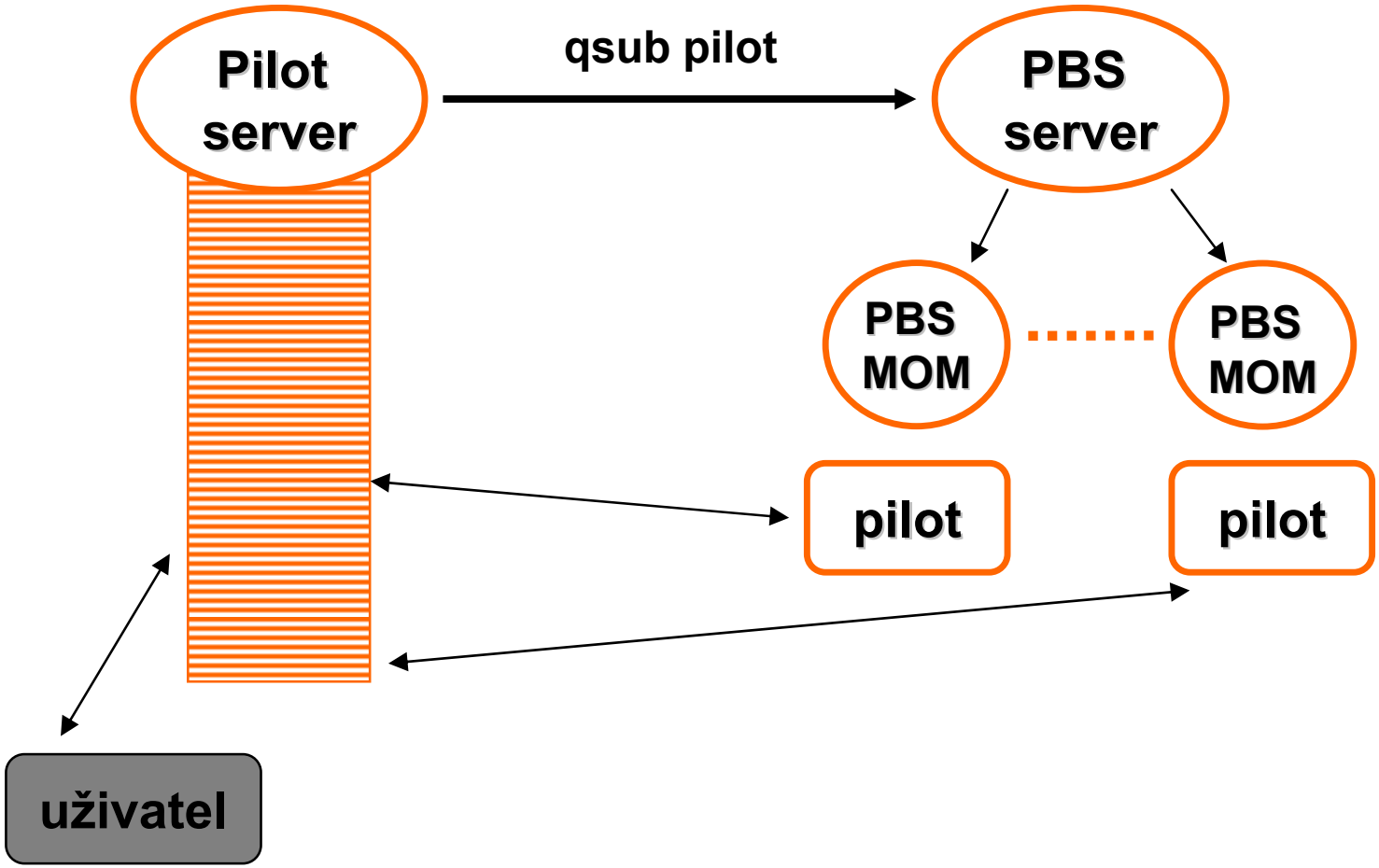
Velké množství úloh, které je vhodné spravovat mimo PBS

- velice krátké úlohy
- potřeba složitějších závislostí, priorit
- správa úloh přes více prostředí

Instalovaný systém Diane, experimentální použití několika skupinami

- úlohy se zadávají do odděleného serveru
- systém udržuje v PBS zadaný počet pilotních úloh
- ty slouží jen jako „obálky“, po spuštění si vyzvednou reálné zadání ze svého serveru
- pilot takto provádí úlohy do vypršení času vyhrazeného v PBS

Pilotní úlohy II



Děkuji za pozornost!