Academia Română
Institutul de Geodinamică *"Sabba S.Ştefănescu"*
Laboratorul Dinamica Globului Terestru
Str. Jean-Louis Calderon, Nr. 19-21, Bucureşti-37, România, R-020032,
fax:(4021) 317.2120, tel. (4021) 317.2126; e-mail: inst_geodin@geodin.ro
http://www.geodin.ro/~prezentare/

DIRECTOR,

Dr. Crişan DEMETRESCU
Membru corespondent al Academiei Române

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2007 - 2013

Programul Operațional Sectorial Creşterea Competitivității Economice
Axa prioritară 2: Competitivitate prin Cercetare, Dezvoltare Tehnologică şi Inovare
Operațiunea: O.2.1.2 „Proiecte CD de înalt nivel ştiințific la care vor participa specialişti din străinătate"
Proiect: Infrastructură cibernetică pentru studii geodinamice relaționate cu zona seismogenă Vrancea: ID-593, cod SMIS-CSNR 12499
Etapa 2: Construirea si vizualizarea unei serii de modele geodinamica tridimensionale de inalta rezolutie cu ajutorul sistemului HPCC/HPVC/GeoWall: Activarea si testarea sistemului hardaware HPCC si HPVC
Perioada: 18 iunie 2011-17 decembrie 2011

STUDIU

PRIVIND INSTALAREA/CONFIGURAREA/TESTAREA HPCC

Director de proiect,

Dr. Vlad Constantin Manea

A U T O R I :

Dr. Vlad Constantin Manea

Dr. Marina Manea

Drd. Mihai Pomeran

CUPRINS:

1. INTRODUCERE

Scopul principal al proiectului CYBERDYN este construirea unei infrastructuri cibernetice in cadrul Institutului de Geodinamica al Academiei Romane, Bucuresti, pentru studierea evolutiei geodinamice pe termen lung a zonei seismogene active Vrancea. Aceasta infrastructura cibernetica este formata dintr-un HPCC (High Performance Computing Cluster – Grup de Servere pentru Calcule de Inalta Peformanta), un HPVC (High Performance Visualization Cluster – Grup de Servere pentru Vizualizare de Inalta Performanta) si un sistem de Vizualizare Stereo in 3D (GeoWall).

Noua infrastructura cibernetica va ajuta la crearea unui corp de cercetatori format din experti cu inalta pregatire obtinuti prin antrenarea tinerilor oameni de stiinta in campul geodinamicii computationale, permitand generarea primului centru de excelenta in domeniu din Romania. Activitatea acestui centru de excelenta se va extinde si dupa finalizarea ultimei etape a proiectului prin formarea tinerilor specialisti si prin participarea in proiecte nationale/internationale bazata pe capacitatea si performanta oferite de o asemenea tehnologie.

In acest studiu se prevede descrierea instalarii, configurarii si testarii sistemului de calcul HPCC CyberDyn. Aditional, asa cum s-a prevazut in cererea de finantare, scopul acestui studiu este de a furniza membrilor echipei CyberDyn cunostiintele necesare in ceea ce priveste lucrul pe HPCC CyberDyn. Studiul este alcatuit din urmatoarele capitole principale:

-Administrarea sistemului de calcul HPCC-CyberDyn.

-Sistemul de administrare a lansarii programelor Sun Grid Engine integrat in infrastructura de calcul HPCC – CyberDyn.

-Testarea sistemului CyberDyn la maxima capacitate folosing pachetul HPL-Linpack.

-Instalarea, configurarea si testarea programului Matlab (MDCE (Matlab Distributed Computing Server) si Parallel Toolbox) si functionarea in paralel pe infrastructura de calcul HPCC – CyberDyn.

-Concluzii finale.

-Referinte bibliografice.

2. ADMINISTRAREA SISTEMULUI DE CALCUL HPCC-CYBERDYN

Sistemul de calcul HPCC CyberDyn este configurat astfel incat sa beneficieze de avantajele oferite de soft-ul de comanda si control Bright Cluster Manager (*http://www.brightcomputing.com*). Acesta este un soft cu licenta care ofera urmatoarele avantaje:

1) Faciliteaza instalarea, managamentul si utilizarea clusterelor cu sistem de operare Linux.
2) Ofera posibilitatea de a se adauga facil noi resurse de calcul.
3) Este un soft profesional care ofera administratorului toate instrumentele necesare administrarii intregului sistem de calcul.

Mai jos prezentam caracteristicile specifice configurarii Bright Cluster Management pe sistemul HPCC CyberDyn.

1) Comenzile de administrare se introduc de la consola de comanda, instalata in rack-ul principal (rack 03). Accesul la aceasta consola este limitat cu ajutorul a doua incuietori cu amprenta. Exista posibilitatea de a introduce aceste comenzi folosind o conexiune securizata cu server-ul Sapphire.

2) Consola de comanda poate fi conectata la mai multe sisteme, acestea aflandu-se in categoriile: Noduri principale; Noduri "worker", sau noduri de calcul, Alte servere aflate in incinta (Web-servere; file-servere). Pentru a putea sa navigam intre aceste sisteme conectate (maxim 5) se foloseste tasta Print Screen, dupa care se alege din lista care apare, sistemul pe care dorim sa-l administram.

3) Serverul central (master node) Sapphire, lucreaza sub un sistem de operare Linux (o varianta a RedHat Linux) si are instalat un sistem de administrare si de gestiune specific, inclus de catre furnizorul Cluster Vision.

Sistemul de gestionare si administrare este impartit in doua componente cu functionalitati similare.

Comanda de pornire a programului de administrare care functioneaza in mod grafic este:

$cmgui #Cluster Management Graphic User Interface

Comanda de pornire a programului de administrare care functioneaza de la consola este:

$cmsh #Cluster Management SHell

De exemplu, pentru a porni unul dintre nodurile serverului din mod grafic, se navigheaza pana la nodul respectiv in meniul de tip tree din stanga ferestrei programului de administrare si

dupa ce acesta a fost selectat, din partea dreapta se selecteaza tab-ul administrare (Tasks). Aici exista un buton care o data apasat, lanseaza comanda de pornire.

Pentru a efectua aceeasi comanda din modul consola, dupa ce se porneste programul de administare cmsh:

-Se intra in contextul "device". Pentru asta se tasteaza:

$[sapphire1]% device

-Se intra in contextul nodului respectiv de exemplu, nodul019. Pentru asta se tasteaza:

$[sapphire1->device]% use node019

-Apoi in acest context se executa comanda:

$[sapphire1->device[node019]]% power on

-Acelasi rezultat se poate obtine si daca direct in contextul device se executa comanda:

$[sapphire1->device]% power on -n node019

Lista completa de variante ale comenzii power este:

power [optiuni] status: aceasta comanda afiseaza situatia pornit/oprit a echipamentelor care pot fi interogate prin reteaua IPMI

power [optiuni] on: aceasta comanda executa o comanda de pornire a echipamentelor care sunt legate la reteaua IPMI

power [optiuni] off: aceasta comanda executa o comanda de oprire (hard) a echipamentelor care sunt legate la reteaua IPMI

power [optiuni] reset: aceasta comanda executa o comanda de resetare (hard) a echipamentelor care sunt legate la reteaua IPMI

Lista intreaga de optiuni ale comenzilor power este:

-n, --nodes lista(noduri)

Lista de noduri, ex. node001..node015,node20..node028,node030 sau ^/un/fisier/continand/noduri

-g, --group lista(grupuri)

Include toate nodurile care apartin unui grup de noduri, ex. testnodes sau test01,test03

-c, --category lista(categorii)

Include toate nodurile care apartin unei categorii, ex. default sau default,gpu

-r, --rack lista(rack-uri)

Include toate nodurile care sunt pozitionate intr-un anumit rack, ex rack01 sau rack01..rack04

-h, --chassis lista(sasiuri)

Include toate nodurile care sunt pozitionate intr-un anumit sasiu, ex chassis01 sau chassis03..chassis05 chassis

-p, --powerdistributionunitport <pdu:port>(lista)

Comanda de pornire executata direct pe PDU (power distribution units) - Prize modulare cu montare in rack. Se poate folosi "*" pentru toate porturile unui PDU.

-b, --background

Executa comanda in fundal, mesajele fiind transmise ca evenimente in log.

-d, --delay <secunde>

Asteapta <secunde> intre executia a doua comezi "power" secventiale. Aceasta optiune este ignorata in cazul variantei "status" a comenzii power.

-s, --state <status>

Filtreaza echipamentele asupra carora se ruleaza comanda "power" prin status-ul in care se afla acestea, ex. UP, "CLOSED|DOWN", "INST.*"

-f, --force

Forteaza comanda power asupra echipamentelor care au fost inchise.

Alta comanda folositoare, inrudita cu power este reboot. Modul de executare al acestei comenzi este:

reboot [OPTIUNI/nod]

Optiunile sunt:

-n, --nodes lista(noduri)

Lista de noduri, ex. node001..node015,node20..node028,node030 sau ^/un/fisier/continand/noduri

-g, --group lista(grupuri)

Include toate nodurile care apartin unui grup de noduri, ex. testnodes sau test01,test03

-c, --category lista(categorii)

Include toate nodurile care apartin unei categorii, ex. default sau default,gpu

-r, --rack lista(rack-uri)

Include toate nodurile care sunt pozitionate intr-un anumit rack, ex rack01 sau rack01..rack04

-h, --chassis lista(sasiuri)

Include toate nodurile care sunt poziţtionate intr-un anumit sasiu, ex chassis01 sau chassis03..chassis05 chassis

Din aceasta categorie de comenzi mai face parte si shutdown. Modul de executare al acestei comenzi este:

shutdown [OPTIUNI/nod]

Optiunile sunt:

-n, --nodes lista(noduri)

Lista de noduri, ex. node001..node015,node20..node028,node030 sau ^/un/fisier/continand/noduri

-g, --group lista(grupuri)

Include toate nodurile care apartin unui grup de noduri, ex. testnodes sau test01, test03

-c, --category lista(categorii)

Include toate nodurile care apartin unei categorii, ex. default sau default,gpu

-r, --rack lista(rack-uri)

Include toate nodurile care sunt pozitionate intr-un anumit rack, ex rack01 sau rack01..rack04

-h, --chassis lista(sasiuri)

Include toate nodurile care sunt pozitionate intr-un anumit sasiu, ex chassis01 sau chassis03..chassis05 chassis

Pentru a verifica status-ul echipamentelor monitorizate, se poate folosi comanda status. Modul de executare al acestei comenzi este:

status

status <nod>

status [OPTIUNI]

Optiunile sunt:

-n, --nodes lista(noduri)

Lista de noduri, ex. node001..node015,node20..node028,node030 sau ^/un/fisier/continand/noduri

-g, --group lista(grupuri)

Include toate nodurile care apartin unui grup de noduri, ex. testnodes sau test01, test03

-c, --category lista(categorii)

Include toate nodurile care apartin unei categorii, ex. default sau default,gpu

-r, --rack lista(rack-uri)

Include toate nodurile care sunt pozitionate intr-un anumit rack, ex rack01 sau rack01..rack04

-h, --chassis lista(sasiuri)

Include toate nodurile care sunt pozitionate intr-un anumit sasiu, ex chassis01 sau chassis03..chassis05 chassis

Pentru a afla mai multe informatii despre un anumit nod, se poate executa comanda „sysinfo”. Aceasta comanda ofera informatii stocate de catre producator in memoria echipamentului respectiv. Modul de executare al acestei comenzi este:

sysinfo <nod>

Exemple:

sysinfo – Ofera informatii despre nodul curent

sysinfo node001 – Ofera informatii despre nodul „node001”

Pentru a afla informatii specifice sistemului de gestiune si administrare furnizat de Cluster Vision putem folosi comanda „show”. Modul de executare al acestei comenzi este:

show <nod>

Exemple:

show – Ofera informatii despre nodul curent

show node001 – Ofera informatii despre nodul „node001”

Pentru a afla ce echipamente sunt controlate de sistemul de gestionare si administrare, putem folosi comanda „list”. Aceasta comanda va intoarce o lista cu aceste echipamente si cateva atribute importante ale acestora, cum ar fi: Tipul, Numele sistemului, Adresa MAC, Categoria (se introduce din cmsh), adresa IP, Reteaua (se introduce din cmsh). Modul de executare al acestei comenzi este:

list [OPTIUNI]

list [OPTIUNI] <tip>

Optiunile sunt:

-f, --format	Foloseste alt format decat cel uzual
-s, --sort	Foloseste alta ordonare decat cea uzuala
-d, --delimiter	Seteaza separatorul de rand
-a, --all	Afiseaza inclusiv obiectele marcate pentru stergere.

Argumente:

<tip> = Tipul echipamentului: physicalnode, virtualsmpnode, masternode, ethernetswitch, ibswitch, myrinetswitch, powerdistributionunit, genericdevice, racksensor,chassis, gpuunit

Sistemul de gestionare si administrare inclus de catre furnizor inregistreaza o serie de valori utile pentru administratorul de sistem. Un exemplu pentru o astfel de valoare este temperatura ambientala inregistrata la un moment dat de catre un nod al cluster-ului. Pentru a

afla o lista a valorilor inregistrate in baza de date de catre sistemul de gestionare si administrare, se foloseste comanda „metrics”. Modul de executare al acestei comenzi este:

metrics <echipament>

Exemple:

metrics – Ofera informatii despre nodul / echipamentul curent

metrics node001 Ofera informatii despre nodul / echipamentul „node001”

Un extras dintr-o lista de valori pentru un nod uzual (worker) este:

Uptime = Timp scurs de la ultima pornire / restartare
MemoryUsed = Memorie RAM folosita
SwapUsed = Memorie Swap folosita
CPUUser = CPU folosit de procese ale utilizatorilor
CPUSystem = CPU folosit de procese de sistem
CPUIdle = CPU nefolosit
BytesRecv:eth1 = Bytes primiti pe interfata eth1
BytesRecv:eth2 = Bytes primiti pe interfata eth2
BytesRecv:eth3 = Bytes primiti pe interfata eth3
BytesRecv:ib0 = Bytes primiti pe interfata infiniband
BytesSent:eth1 = Bytes trimisi pe interfata eth1
BytesSent:eth2 = Bytes trimisi pe interfata eth2
BytesSent:eth3 = Bytes trimisi pe interfata eth3
BytesSent:ib0 = Bytes trimisi pe interfata infiniband
UsedSpace:/ = Spatiu pe disc folosit
UsedSpace:/local = Spatiu pe disc folosit in partitia /local
UsedSpace:/tmp = Spatiu pe disc folosit in partitia /tmp
UsedSpace:/var = Spatiu pe disc folosit in partitia /var
FreeSpace:/ = Spatiu pe disc disponibil
FreeSpace:/local = Spatiu pe disc disponibil pe partitia /local
FreeSpace:/tmp = Spatiu pe disc disponibil pe partitia /tmp
FreeSpace:/var= Spatiu pe disc disponibil pe partitia /var
Ambient_Temp = Temperatura ambientala
FAN_1_RPM = Frecventa de rotatie a ventilatorului 1
FAN_2_RPM = Frecventa de rotatie a ventilatorului 2
FAN_3_RPM = Frecventa de rotatie a ventilatorului 3
FAN_4_RPM = Frecventa de rotatie a ventilatorului 4
FAN_5_RPM = Frecventa de rotatie a ventilatorului 5
FAN_6_RPM = Frecventa de rotatie a ventilatorului 6

3. FOLOSIREA SUN GRID ENGINE INTEGRAT IN INFRASTRUCTURA DE CALCUL HPCC – CYBERDYN.

Sun Grid Engine (SGE: gridengine.org) reprezinta un job scheduler avansat open sourse care programeaza rularea de programe intr-un sistem de calcul tip HPCC – CyberDyn. Desi exista si alte sisteme similare la nivel mondial (de ex. Torque, sau PBS), s-a decis folosirea SGE pe motivul experientei anterioare pe care o detin cativa membri ai echipei de lucru CyberDyn, care au utilizat SGE atat la Caltech (California Institute of Technology), cat si la UNAM (Universidad Nacional Autonoma de Mexico). Mai jos prezentam cateva exemple de folosire practica a sistemului SGE, cu scopul de a-i initia si pe ceilalti membrii ai echipei CyberDyn cu lucrul pe HPCC CyberDyn.

Scopul principal al unui job scheduler (sau programator de rulari) este sa permita utilizarea resurselor de calcul in cel mai eficient mod posibil. SGE este scris si distribuit de catre compania Sun Microsystems sub licenta Sun Industry Standards Source, si este disponibil gratis. Pagina web unde SGE este localizat este: *http://gridengine.sunsource.net*

3.1. CUM SE TRIMIT PROGRAME BATCH CATRE SGE

Programele sunt transferate catre SGE cu ajutorul unor mici secvente de cod numite scripts. Mai jos prezentam un exemplu de job script serial, numit sleep.sh, care executa comanda bash sleep.

```
cat sleep.sh
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#
date
sleep 10
date
```

Liniile din script-ul de mai sus care incep cu #$ sunt recunoscute de catre SGE ca optiuni.

- cwd indica SGE-ului sa ruleze programe din directorul current.

- j y indica SGE-ului sa combine iesirea standard error si iesirea standard output intr-un singur fisier in loc de doua fisiere separate.

- S /bin/bash indica SGE ca interpretatorul de comenzi este Bash shell.

Pentru a putea trimite catre executie pe nodurile de calcul ale sistemului HPCC – CyberDyn programul de mai sus, trebuie folosita comanda SGE qsub:

```
# qsub sleep.sh
```
your job 16 ("sleep.sh") has been submitted

Mai jos prezentam un exemplu de script folosit pentru a transmite catre executie pe nodurile de calcul ale sistemului de calcul HPCC – CyberDyn un job paralel numit submit_mpi.sh:

```
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#
/opt/openmpi/bin/mpirun -np $NSLOTS $HOME/test/submit_mpi
```

Comanda folosita pentru a trimite catre executie programe MPI in paralel este similara cu aceea folosita pentru a submite programe seriale, cu unica diferenta ca este nevoie sa se utilizeze urmatoarea sintaxa: *-pe orte N*. N se refera la numarul de procesoare pe care utilizatorul doreste sa le aloce programului MPI. Mai jos prezentam un mic exemplu in care se transmite catre executie un program care foloseste doua procesoare:

```
$ qsub -pe orte 2 submit_mpi.sh
```

Cand este incheiata executia, mesajele de iesire sunt scrise in fisierul submit_mpi.sh.o*, iar mesajele de eroare vor fi scrise in fisierul submit_mpi.sh.po*.

3.2. CATEVA COMENZI SGE UTILE PENTRU LUCRU

1) Pentru a putea executa un program pe un anumit nod de calcul:

 # qsub -l hostname=node001 myjob.sh

2) Pentru a trimite catre executie acelasi program de un anumit numar de ori (50 de ori de exemplu)(array of jobs):

 # qsub -t 1-50 myjob.sh

3) Pentru a sterge un anumit program care se executa pe sistemul HPCC – CyberDyn (de exemplu job ID numarul 10)

 # qdel 10

4) In cazul in care nu se poate sterge un job cu comanda de mai sus, se poate utiliza optiunea –f adica“force”:

 # qdel -f 10

5) Mai jos prezentam cateva explicatii suplimentare aferente comenzii *# qstat –f:*

- a(larm)
- A(larm)
- C(alendar suspended)
- s(uspended)
- S(ubordinate)
- d(isabled)
- D(isabled)
- E(rror)

6) In cazul in care se doreste sa se stearga mai multe programe printr-o singura comanda:

 # qdel -u $USER

7) Afisarea stadiului in care se afla programele care au fost trimise catre executie pe sistemul de calcul HPCC – CyberDyn:

 # qstat

 # qstat -f (mai multe detalii)

 # qstat -g c (mai putine detalii)

 # qstat -s r (afiseaza numai programele care ruleaza (“r”))

 # qstat -j 999 (afiseaza toate detaliile legate de programul cu identificatorul ID= 999)

8) Urmatoarea comanda afiseaza detalii legate de resursele de calcul disponibile pe sistemul HPCC – CyberDyn:

 # qhost

3.3. COMENZI AVANSATE SGE:

1) Daca se doreste sa se ruleze un program pe anumite noduri de calcul:

 # qsub -pe mpi 8 –q all.q at node001, all.q at node002, all.q at node003 ,all.q at node004 /home/vladcm/my.qsub

2) Se poate intrerupe trimiterea catre executie a unor programe pe un anumit nod de calcul:

 # qmod -d '@<hostname>'*

In acest caz, nu se vor mai trimite catre executie programe catre nodul specificat, acesta fiind evitat. In cazul in care deja exista programe care ruleaza pe aceste noduri, ele vor fi lasate sa se incheie fara probleme.

3) Urmatoarea comanda este permisa numai administratorului de sistem:

 # qconf -mhrp @allhosts

Aceasta comanda deschide configuratia @allhosts intr-un editor vi (care este editorul de texte standard pe sistemul de calcul HPCC - CyberDyn). Pentru a elimina un nod de calcul trebuie eliminat nodul din linia in care apare 'hostlist' (atentie ca aceasta variabila foloseste \ pentru a indica trecerea la o noua linie!). In acest moment se poate salva si iesi din editorul vi, efectul acestor modificari fiind imediat. Se poate verifica daca indepartarea unui anumit nod de calcul (de exemplu nodul node005) a fost facuta cu success:

 # qstat -f | grep "compute-0-5"

Daca pe nodul de calcul node005 se ruleaza deja un program, o sa afiseze urmatorul mesaj:

all.q at node005 BIP 4/4 4.03 lx26-amd64 o

Litera 'o' de la sfarsit indica faptul ca acest nod nu mai face parte din programatorul de rulaje SGE. In cazul in care nu este afisat niciun mesaj pe ecran, pe nodul node005 de fapt nu se executa niciun program in acest moment.

4) Pentru a vedea un rezumat al intregului sistem de calcul, utilizatorul poate folosi urmatoarea comanda:

 # qstat -g c

CLUSTER QUEUE	CQLOAD	USED	AVAIL	TOTAL	aoACDS	cdsuE
all.q	1.00	64	0	64	4	0

5) Urmatoarea comanda SGE blocheaza nodul node005:

 # qmod -d all.q at node005

Asa cum am mai mentionat, daca deja exista un program care ruleaza, acesta va fi lasat sa isi temine executia, apoi nodul node005 va fi blocat si nu mai primeste programe trimise de catre SGE spre executie.

 # qstat -f

all.q at node005 BIP 0/2 0.00 lx26-ia64 d

6) Pentru deblocarea nodului node005 se utilizeaza urmatoarea comanda:

 # qmod –e all.q at node005

In acest fel, nodul node005 este acum pregatit pentru a prelua programe trimise catre executie de catre SGE.

7) Pentru a elimina toate sistemele “queues”, urmatoarea comanda SGE trebuie introdusa de la tastatura:

*# qmod -d *.q*

In cazul in care exista programe care inca ruleaza pe nodurile de calcul, aceste sunt lasate sa isi termine executia in mod normal. Utilizatorii pot trimite inca programe catre nodurile de calcul dar ele vor fi puse in asteptare.

8) Pentru a bloca momentan all.q, se poate folosi urmatoarea comanda SGE:

 # qmod -d all.q

9) Pentru a debloca all.q, se poate folosi urmatoarea comanda SGE:

 # qmod -e all.q - to unlock/release all.q

Mai multe detalii despre SGE se pot gasi la urmatoarea adresa de internet:

http://wikis.sun.com/display/GridEngine/Using+Sun+Grid+Engine

4. TESTAREA SISTEMULUI CYBERDYN LA MAXIMA CAPACITATE FOLOSING PACHETUL HPL-LINPACK.

HPL (*http://www.netlib.org/benchmark/hpl*) reprezinta o implementare portabila a testului de performanta ridicata Linpack pentru supercalculatoarele cu memorie distribuita. HPL consta intr-un pachet de programe care rezolva intr-un mod aleator un sistem aritmetic dens de ecuatii liniare in dubla precizie (64 bits) utilizand supercalculatoare cu memorie distribuita asa cum este CyberDyn. Testul HPL este folosit in prezent pentru masurarea performantei si clasificarea supercalculatoarelor in topul mondial (*http://www.top500.org*). In urma aplicarii testului HPL pe supercalculatorul CyberDyn s-a obtinut o performanta maxima de 7.6 TFlopi ceea ce reprezinta o eficienta de 67% (din 11.34 TFlopi pentru performanta teoretica maxima) (a se vedea figura de mai jos).

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4      250000   128    32    42            1407.96              7.398e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0011130 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4      300000   128    32    42            2376.51              7.574e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0012504 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4      350000   128    32    42            3887.73              7.352e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0011484 ...... PASSED
================================================================================
```

Extras din testul HPL rulat pe supercalculatorul CyberDyn. A se observa ca performanta maxima obtinuta in timpul testului a fost de 7.574 TFlopi.

5. INSTALAREA, CONFIGURAREA SI TESTAREA PROGRAMULUI MATLAB (MDCE - MATLAB DISTRIBUTED COMPUTING SERVER SI PARALLEL TOOLBOX) SI FUNCTIONAREA IN PARALEL PE INFRASTRUCTURA DE CALCUL HPCC – CYBERDYN.

Infrastructura de calcul HPCC - CyberDyn ofera posibilitatea rularii de coduri Matlab in paralel. Matlab paralel reprezinta un mediu de dezvoltare de soft al companiei MathWorks. Mai jos am sintetizat mai multe informatii si exemple in vederea rularii de programe Matlab folosind pachetele Parallel Computing Toolbox (PCT) si Matlab Distributed Computing Server (MDCS) instalate pe High Performance Computing Cluster – CyberDyn. CyberDyn ofera posibilitatea rularii in paralel de programe scrise in Matlab folosind un numar maxim de 8 nuclee de calcul. Nodul principal (sau de log-in) are instalat doar o singura licenta de Matlab, si ca atare doar un singur utilizator/sesiune poate sa foloseasca Matlab intr-o sesiune interactiva. MDCS permite ca programul scris in Matlab sa fie distribuit pe 8 nuclee de calcul folosing sistemul de fisiere comun al intregului sistem. Utilizatorul poate sa ruleze programe Matlab in mod interactive sau folosind programatorul de programe oferit de Matlab (modul batch).

5.1. ARHITECTURA MATLAB PARALEL

Matlab paralel este alcatuit din doua module esentiale: Parallel Computing Toolbox (PCT) si MATLAB Distributed Computing Server (MDCS). Matlab Distributed Computing Engine (DCE) permite utilizatorilor sa lanseze doua tipuri de programe (job-uri): cu distributie simpla (de ex. mai multe job-uri folosind doar un singur nucleu de calcul) si complexe in paralel (de ex. job-uri matlab in paralel). Un job se considera de tip distribuit atunci cand mai multe task-uri ruleaza independent si fara sa existe comunicatie intre nucleele de calcul pe care se efectueaza rulajele. In modul batch acest tip de rulaj se numeste job multiplu folosind un singur procesor. Un job este considerat paralel atunci cand un singur task ruleaza simultan pe mai multe nuclee de calcul, iar aceste nuclee de calcul comunica intre ele. In modul batch, acest tip de rulaj se numeste job cu procesoare multiple, sau job tip data-parallel.

5.2. ELEMENTE SI CAPABILITATI ALE MODULULUI PARALLEL COMPUTING TOOLBOX

Modulul PCT ruleaza direct pe nodul de log-in al HPCC CyberDyn, si ofera o serie de capabilitati importante:

1. Loop-uri for in paralel (parfor)
2. Matrici distribuite pe un numar maxim de 8 nuclee de calcul.
3. Blocuri SPMD (single program, multiple data) care executa coduri in paralel de o maniera similara cu MPI; un cod MATLAB amplasat intr-un bloc SPMD se executa simultan pe un numar de nuclee de calcul stabilite de utilizator (pool de nuclee); fiecare proces poate fi identificat cu ajutorul unei variabile tip labindex.

Nota: utilizatorul trebuie sa decida de la inceput, in functie de programul pe care trebuie sa-l ruleze, daca are nevoie de un job distribuit sau paralel. De exemplu, daca aplicatia contine un set mare de date pe care utilizatorul doreste sa execute o serie de calcule simultane, atunci se poate folosi un job paralel (se poate folosi si modul “pmode”, care este prezentat mai tarziu). Daca aplicatia contine calcule repetitive care pot fi executate independent unul fata de altul, atunci un job distribuit este recomandat.

5.3. CONECTAREA, CONFIGURAREA SI TESTAREA MATLAB IN PARALEL

Pentru a putea rula Matlab pe HPCC – CyberDyn, utilizatorul trebuie sa se conecteze la CyberDyn folosind numele de utilizator si parola oferite de catre administratorul sistemului de calcul. In acest sens trebuie folosit exclusiv sistemul SSH (secure shell). De exemplu:

$ ssh –l –Y user sapphire

Optiune "-l" se refera la utilizator, iar optiune "-Y" deschide un canal grafic.

Odata conectat la sistem, utilizatorul este directionat automat catre directorul /home/$USER unde poate sa stocheze rutinele de calcul precum si datele de intrare si iesire.

Pentru a initia o sesiune interactiva de Matlab, utilizatorul trebuie sa introduca de la tastatura: matlab. In cateva secunde interfata grafica a Matlab o sa apara pe ecran. Primul lucru care trebuie facut odata ce interfata grafica Matlab este complet incarcata, este sa se testeze conectivitatea in paralel a Matlab cu nodurile de calcul.

Nota: nu folositi optiunea "local", aceasta implica rulaje de Matlab pe nodul de login si ca atare duce la o diminuare a randamentului intregului sistem de calcul.

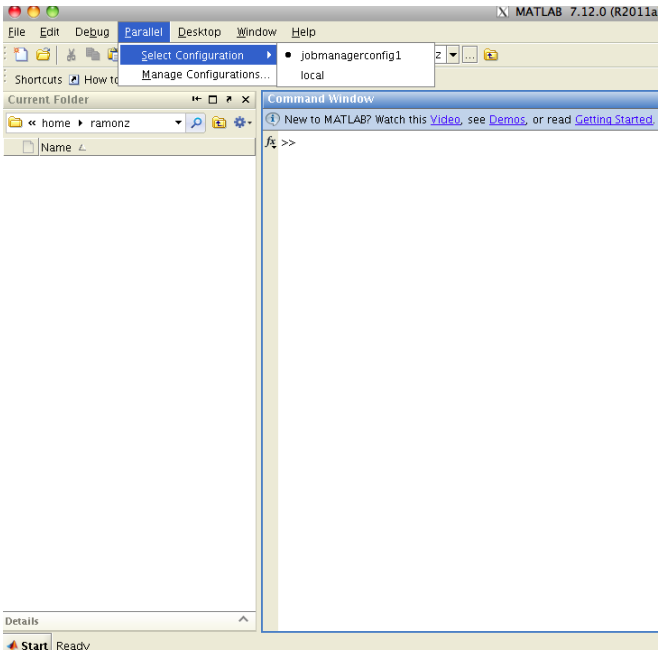


Figura 1. Matlab paralel are o optiune in meniul principal "Parallel".

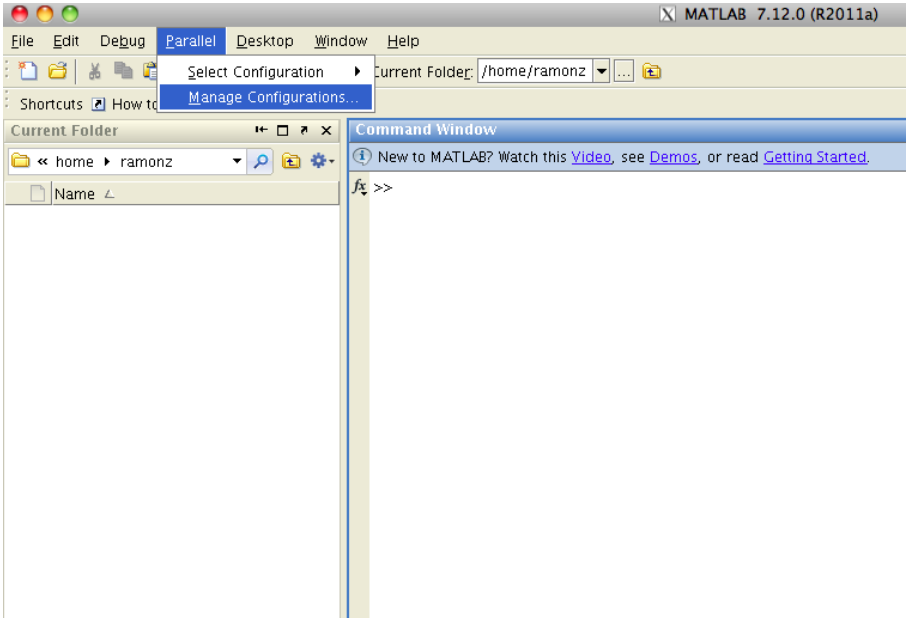


Figura 2. Selectand "Manage Configurations …" utilizatorul poate sa creeze un "jobmanager" cu un numar prestabilit de pana la 8 workers.

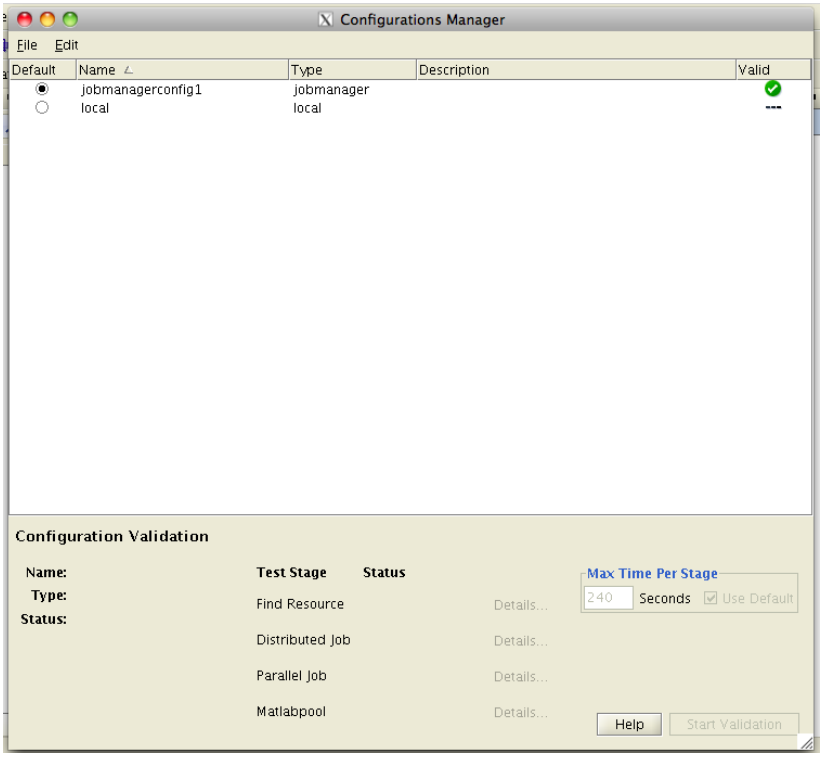


Figura 3. Dupa crearea unui "jobmanager" trebuie validata configuratia. Daca nu se obtin erori, atunci utilizatorul poate sa foloseasca Matlab in paralel. In cazul in care apar erori, trebuie instiintat administratorul HPCC – CyberDyn.

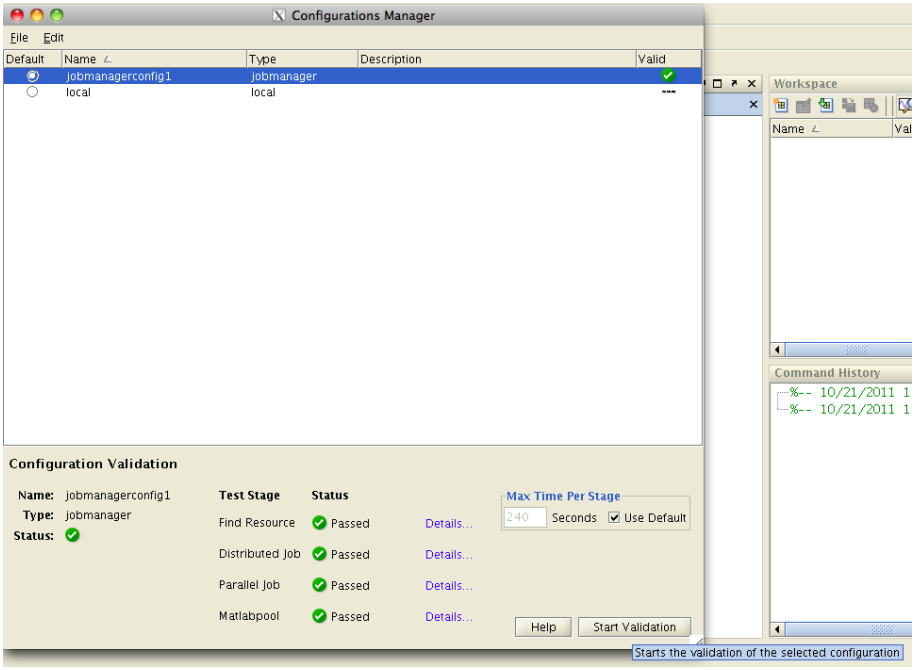


Figura 4. Un exemplu de validare a unui “jobmanager” fara erori.

5.4. EXEMPLE DE CODURI MATLAB UTILIZAND PARFOR SI SPMD

5.4.1. PARFOR: un FOR Loop in parallel

parfor reprezinta comanda Matlab care transforma for-loop serial in for-loop paralel. Practic, **parfor** inlocuieste **for** in cazul in care un for-loop se poate paraleliza.

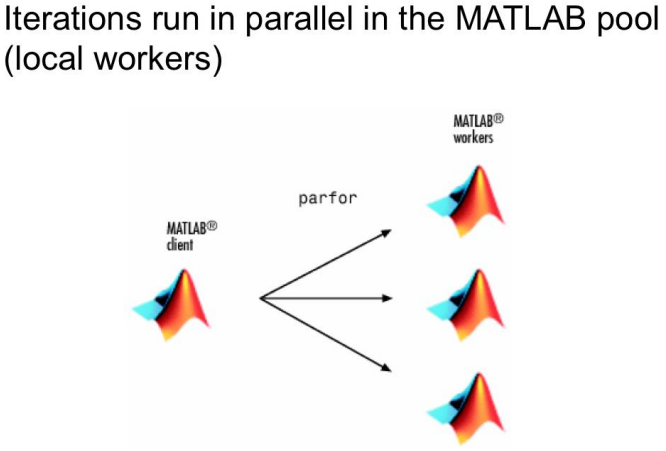


Figura 5. Folosind **parfor**, un ciclu **for** este distribuit pe 3 workers (computing cores).

Iteratiile din ciclul for sunt impartite si distribuite pe diferiti workers (computing cores). Operatiile din ciclul for sunt executate intr-o ordine aleatoare (si necunoscuta) iar rezultatul final este trimis inapoi codului de Matlab principal. Acest tip de paralelizare este foarte comun in simularile tip Monte Carlo.

Nota: Din cauza ca iteratiile din cadrul unui ciclu paralelizat cu parfor sunt executate intr-o ordine necunoscuta, toate datele necesare trebuie sa existe inainte ca ciculul paralelizat sa fie apelat. Practic, o iteratie nu trebuie sa aiba legatura cu rezultatul dintr-o iteratie anterioara. De exemplu, urmatorul cod nu poate fi paralelizat folosind **parfor**.

```
x(1) = 0.0;
for i = 2 : n
x(i) = x(i-1) + dt * f ( x(i), t );
end
```

Un alt exemplu de cod in care parfor nu poate fi utilizat este prezentat mai jos. Este vorba despre un ciclu for care aproximeaza a doua derivata.

parfor i = 2 : n - 1

*d2(i) = **(a(i-1)** - 2 * **a(i)** + **a(i+1)**) / (2 * dx);*

end

Din cauza ca variabila a() este indexata cu i-1, i, si i+1, acest ciclu nu poate fi paralelizat iar utilizarea parfor NU este posibila.

In continuare prezentam cateva exemple de ciclu **parfor** perfect functional.

In cazul in care o variabila este indexata de catre index-ul din cicluri parfor, atunci indexarea trebuie facuta in asa fel incat variabila tip array poate sa fie impartita, fiecare dintre elementele matricei avand un idex unic. Acest tip de matrice poate fi impartita si distribuita catre diferite procesoare (nuclee de calcul sau workers), se pot face operatii pe fiecare parte, iar apoi rezultatul final este obtinut prin combinatia fiecarei parti.

parfor i = 2 : n

*a(i,2) = b(i+1) + a(i,2) * func (c(i-3)) + d(j);*

end

De exemplu, la iteratia 17 se obtine A(17,2), B(18), C(14), si in nicio parte din program in timpul ciclului parfor nu exista aceste date, in afara de pasul 17.

Alte cateva exemple corecte sunt prezentate mai jos.

Un ciclu **parfor** simplu:

```
>> for i=1:1024
A(i)=sin(i*2*pi/1024);
end
>> plot (A)
>>
>> matlabpool open
Starting matlabpool using the 'local' configuration ... connected to 2 labs.
>> parfor i=1:1024
A(i)=sin(i*2*pi/1024);
end
>> plot (A)
>> matlabpool close
Sending a stop signal to all the labs ... stopped.
>>
```

Crearea unei functii pcalc() care foloseste **parfor**.

```
function pcalc()
N = 60;
a = zeros(N,1);

% Time consuming loop
tic
parfor (i = 1:N)
    a(i) = iFunctionTakesLongTime();
    % Other computations;
end
toc
%
plot(a);
figure(1)
end

function o = iFunctionTakesLongTime()
    o = max(abs(eig(rand(300))));
end
```

Aproximarea unei integrale.

Mai jos este prezentata varianta seriala a codului pentru estimarea unei integrale:

```
function q = quad ( n, a, b )
q = 0.0;
dx = ( b - a ) / ( n - 1 );
for i = 1 : n
        x = a + ( i - 1 ) * dx;
        fx = x^3 + sin ( x );
        q = q + fx;
end
q = q * ( b - a ) / n;
return
end
```

Functia quad aproximeaza integrala unei functii pe un interval finit [a, b]. Practic, se estimeaza integral a pe un numar de “n” intervale, in fiecare interval evaluandu-se aria unui dreptunghi delimitat superior de functia de integrat, inferior de axa x iar lateral de limitele intervalului finit (a-b)/n (Figura 6). Suma tuturor ariilor reprezinta o aproximare a integralei. Ceea ce este important de mentionat este ca se poate face evaluarea acestor arii in orice ordine, si ca atare putem sa folosim comanda **parfor**.

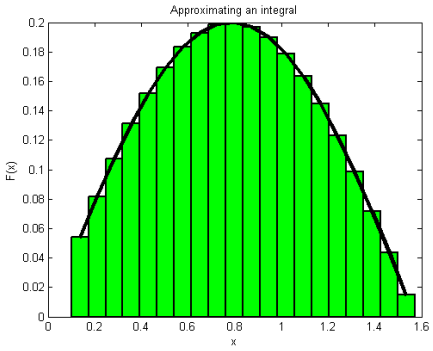


Figura 6. Aproximarea unei integrale cu suma finita a ariilor unor dreptunghiuri de dimensiuni mici (infinitezimale).

Varianta paralela a codului pentru aproximarea unei integrale este prezentat mai jos:

```
function q = quad ( n, a, b )
q = 0.0;
dx = ( b - a ) / ( n - 1 );
parfor i = 1 : n
        x = a + ( i - 1 ) * dx;
        fx = x^3 + sin ( x );
        q = q + fx;
end
q = q * ( b - a ) / n;
return
end
```

Mai jos prezentam in detaliu care sunt etapele de calcul paralel pentru exemplul de mai sus: functia incepe ca un cod serial care ruleaza pe un singur procesor. Atunci cand intalneste comanda **parfor** codul serial se opreste pe moment, iar executia este transferata nodurilor de calcul (computing cores sau workers). Fiecare worker proceseaza un anumit numar de iteratii din ciclul **parfor**, si dupa caz, poate sa preia o anumita cantitate de date de intrare. Fiecare worker transmite codului serial rezultatul final, iar la terminarea cicului **parfor**, controlul este preluat de codul serial. Codul serial, in modul batch, se ruleaza pe unul dintre nodurile de calcul, ca un worker aditional. Programul Matlab asigura functionarea corecta a programului, fie ca este serial ori paralel.

Mai devreme am mentionat ca un ciclu for nu poate fi paralelizat daca rezultatul unei iteratii depinde de o iteratie anterioara. In codul de mai sus, aparent aceasta conditie nu este indeplinita: q = q + fx. In mod normal, pentru a putea calcula valoarea lui q la iteratia numarul 10, este nevoie de valoarea de la iteratia numarul 9. MATLAB "stie" ca variabila q nu este folosita in cadrul ciclului parfor decat pentru a produce suma finala. Recunoaste aceasta operatie ca pe o **operatie de reducere** si ca atare acest tip de operatie este paralelizata automat. Alte operatii de reducere includ: suma, produsul, suma si produsul logic, minimul si maximul unui calcul.

5.4.2. SPMD: SINGLE PROGRAM, MULTIPLE DATA

In cazul SPMD exista doar un proces care ruleaza pe client si care supervizeaza fiecare lab (worker sau nucleu de calcul) care coopereaza intre ele si care sunt parte dintr-un singur program. Fiecare worker poseda un identificator unic, un fel de ID unic. Acesti workeri au urmatoarele caracteristici:

- fiecare worker ruleaza pe un nucleu de calcul separat;
- fiecare worker foloseste un workspace diferit;
- un program comun este folosit;
- programul principal (client) are capacitatea de a examina si modifica date aferente fiecarui worker;
- oricare doi workeri pot comunica direct prin intermediul mesajelor.

Daca se doreste specificarea unui anumit numar de workers, SPMD primeste ca argument acest numar (n):

```
spmd (n)
   <statements>
end
```

Nota: daca n este 0, spmd nu foloseste workers si ruleaza direct pe client, la fel ca in cazul in care nu exista un pool de workers deschis.

In exemplul de mai jos, se creaza o matrice aleatoare folosind 3 workers (sau labs):

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

Se poate specifica un numar minim si maxim de workers:

```
spmd (m, n)
   <statements>
end
```

In cazul de mai sus, **spmd** are nevoie de un numar minim m de labs si un numar maxim n de labs.

MATLAB prescrie un worker (sau lab) special numit client. MATLAB prescrie numarul cerut de workers, pe fiecare ruland o copie a programului. Fiecare worker "stie" ca este un worker, si are acces la doua functii speciale:

- **numlabs(),** numarul de workers.
- **labindex()**, un identificator unic cu valori cuprinse intre 1 si numlabs().

Nota: In mod current parantezele sunt optionale, dar este important de mentionat ca acestea sunt functii si nu variable. In cazul in care clientul apeleaza aceste functii, amandoua returneaza valoarea 1, pentru ca atunci cand clientul functioneaza, workerii nu si vice versa.

Spre deosebire de un ciclu **parfor**, workerii folositi in cadrul unui bloc **spmd** au un identificator unic cu o valoare data de labindex. In acest fel, se poate rula un cod pe anumiti workeri. De exemplu se poate crea o matrice in functie de identificatorul worker-ului (labindex):

```
spmd (3)
   if labindex==1
      R = rand(9,9);
    else
      R = rand(4,4);
   end
end
```

Un alt exemplu important este legat de matricile codistribuite (codistributed arrays) utilizate intr-un bloc **spmd**:

```
spmd (3)
   RR = rand(30, codistributor());
end
```

In exemplul de mai sus, fiecare lab (sau worker) contine un segment de 30/10 din matricea codistribuita RR. Clientul poate sa determine numarul disponibil de workers cu ajutorul urmatoarei comenzi:

n = matlabpool ('size')

Clientul si workerii impart un singur program in care anumite comenzi sunt specificate in interiorul blocurilor care sunt declarate cu **spmd** si terminate cu **end**. Clientul executa comenzile pana la intalnirea blocului spmd, si in acest moment se opreste; de aici workerii executa comenzile cuprinse in blocul spmd, iar cand acesta se termina, controlul este preluat de client. Clientul si workerii au spatii de lucru diferite, dar exista posibilitatea de a comunica si de a schimba informatii. Valoarea unei variabile definite in programul client poate sa fie apelata de catre workeri, dar nu si modificata. In schimb, variabilele definite de catre workeri pot fi atat apelate cat si schimbate de catre client. In figura 7 se prezinta fluxul de informatii in cazul unui program care contine 2 workeri si un client.

	Client			Worker 1			Worker 2		
	a	b	e	c	d	f	c	d	f
a = 3;	3	-	-	-	-	-	-	-	-
b = 4;	3	4	-	-	-	-	-	-	-
spmd									
c = labindex();	3	4	-	1	-	-	2	-	-
d = c + a;	3	4	-	1	4	-	2	5	-
end									
e = a + d{1};	3	4	7	1	4	-	2	5	-
c{2} = 5;	3	4	7	1	4	-	5	6	-
spmd									
f = c * b;	3	4	7	1	4	4	5	6	20
end									

Figura 7. Fluxul de informatii in cazul unui program care contine 2 workeri si un client

Un program Matlab poate sa contina cateva blocuri **spmd**. Cand executia unui bloc **spmd** este incheiata, workerii intra in stadiul de pauza si nu dispar, spatiul lor de lucru ramane intact. O variabila definita in cadrul unui bloc **spmd** isi conserva valoarea in cazul in care un nou bloc **spmd** este apelat de catre program. Practic, clientul si workerii isi alterneaza executia. In MATLAB, variabilele definite in cadrul unei functii dispar odata ce functia respectiva nu mai este apelata. Acelasi lucru este valabil si pentru un program care apeleaza o functie care contine un bloc **spmd**. In timp ce in cadrul functiei, datele aferente unui worker sunt conservate de la un bloc la altul, odata ce functia se incheie, datele definite pe workeri dispar si nu mai pot fi apelate.

5.4.3. CATEVA EXEMPLE DE PROGRAME CARE INCLUD BLOCURI SPMD

Exemplul 1

```
>> matlabpool
Starting matlabpool using the 'local' configuration ... connected to 2 labs.
>> spmd(2)
R=rand(4,4);
end
>> R(:)

ans =

    [4x4 double]
    [4x4 double]

>> R{1}

ans =

    0.9173    0.4612    0.2155    0.4621
    0.6839    0.1562    0.4978    0.9846
    0.8661    0.4626    0.2904    0.9587
    0.4809    0.8009    0.9071    0.5795

>> R{2}

ans =

    0.2951    0.7010    0.9143    0.7375
    0.0990    0.3821    0.2740    0.5407
    0.3277    0.9602    0.6484    0.6348
    0.6902    0.7780    0.2781    0.0948
```

Exemplul 2

```
>> spmd (2)
   if labindex==1
      R = rand(4,4);
   else
      R = rand(2,2);
   end
end
>> R(:)

ans =

    [4x4 double]
    [2x2 double]

>> R{1}

ans =

    0.5324    0.9413    0.6332    0.8776
    0.1019    0.2410    0.0554    0.9100
    0.5341    0.3698    0.7112    0.5573
    0.2081    0.0299    0.0759    0.1785

>> R{2}

ans =

    0.3802    0.7370
    0.1215    0.3625
```

Exemplul 3 – codistributor()

```
>> matlabpool
Starting matlabpool using the 'local' configuration ... connected to 2 labs.
>> spmd
a=rand(500,500);
end
>> a

a =

   Lab 1: class = double, size = [500  500]
   Lab 2: class = double, size = [500  500]

>> spmd
b=rand(800,800, codistributor())
end
Lab 1:

  This lab stores b(:,1:400).

         LocalPart: [800x400 double]
      Codistributor: [1x1 codistributor1d]

Lab 2:

  This lab stores b(:,401:800).

         LocalPart: [800x400 double]
      Codistributor: [1x1 codistributor1d]
```

Exemplul 4 – codistributor()

```
>> spmd
c=b'
end
Lab 1:

  This lab stores c(1:400,:).

         LocalPart: [400x800 double]
     Codistributor: [1x1 codistributor1d]

Lab 2:

  This lab stores c(401:800,:).

         LocalPart: [400x800 double]
     Codistributor: [1x1 codistributor1d]
```

```
>> spmd
d=c*b
end
Lab 1:

  This lab stores d(:,1:400).

         LocalPart: [800x400 double]
     Codistributor: [1x1 codistributor1d]

Lab 2:

  This lab stores d(:,401:800).

         LocalPart: [800x400 double]
     Codistributor: [1x1 codistributor1d]
```

5.5. RULAREA PROGRAMELOR MATLAB IN MODUL "BATCH"

Exista posibilitatea ca anumite programe in Matlab sa dureze ore sau zile pana cand sa produca rezulatele dorite. In acest caz nu se poate mentine deschisa o sesiune grafica de Matlab, dar exista posibilitatea ca rutinele Matlab sa ruleze in plan secund, fara sa fie nevoie sa se mentina deschisa sesiunea grafica. Acest mod de rulare se numeste mod "**batch**". Mai jos prezentam pasii (5 la numar) care trebuiesc urmati pentru a putea rula un program Matlab in modul **batch**:

Pasul 1: pregatiti un fisier numit (de exemplu) mybatch_par.m:

```
for i=1:10000000
  A(i)=sin(i*2*pi/1024);
end
```

Pasul 2: de la promptul din fereastra unei sesiuni Matlab se poate transmite spre rulare in mod **batch** programul de mai sus folosind 2 workeri (un client si 2 workeri, in total este nevoie 3 workeri disponibili):

```
>> job=batch('mybatch_par', 'Configuration', 'My_Job_Manager_8_Workers', 'matlabpool', 2)
```

Pe ecran or sa apara urmatoarele mesaje informative:

```
job =

MatlabPool Job ID 19 Information
================================

                  UserName : vladycm
                     State : running
                SubmitTime : Mon Oct 31 18:50:27 CST 2011
                 StartTime :
           Running Duration :

- Data Dependencies

           FileDependencies : /home/vladycm/MATLAB/TESTS/mybatch_par.m
           PathDependencies : {}

- Associated Task(s)

            Number Pending  : 3
            Number Running  : 0
            Number Finished : 0
           TaskID of errors :

- Scheduler Dependent (MatlabPool Job)

    MaximumNumberOfWorkers : 3
    MinimumNumberOfWorkers : 3
>> wait(job)
>> load(job)
>> plot(A)
```

Pasul 3: optional se poate astepta ca rulajul sa se termine:

>> wait(job)

Pentru a vedea daca programul inca mai ruleaza pe nodurile de calcul, este necesar sa se introduca urmatoarea comanda:

```
>> s = findResource('scheduler', 'type', 'jobmanager');
>> findJob(s)

ans =

    Jobs: 3-by-1
    ============

  #  Job ID          Type       State       FinishTime  UserName  #tasks
 ---------------------------------------------------------------------------
  1      13    matlabpool    finished  Oct 31 18:37:16   vladycm       2
  2      17    matlabpool    finished  Oct 31 19:14:51   vladycm       7
  3      18    matlabpool     running                -   vladycm       7
```

In exemplul de mai sus se poate vedea ca programul cu inca ruleaza (in acest exemplu am folosit 6 workeri si un client, in total 7 nuclee de calcul).

Cand rulajul este terminat, pe ecran apare urmatorul mesaj:

```
>> findJob(s)

ans =

    Jobs: 3-by-1
    ============

  #  Job ID          Type       State       FinishTime  UserName  #tasks
 ------------------------------------------------------------------------
  1      13     matlabpool    finished  Oct 31 18:37:16   vladycm       2
  2      17     matlabpool    finished  Oct 31 19:14:51   vladycm       7
  3      18     matlabpool    finished  Oct 31 19:16:48   vladycm       7
```

Pasul 4: pentru a incarca rezultatele rulajului si pentru a prezenta grafic rezultatele se introduc de la tastatura urmatoarele comenzi:

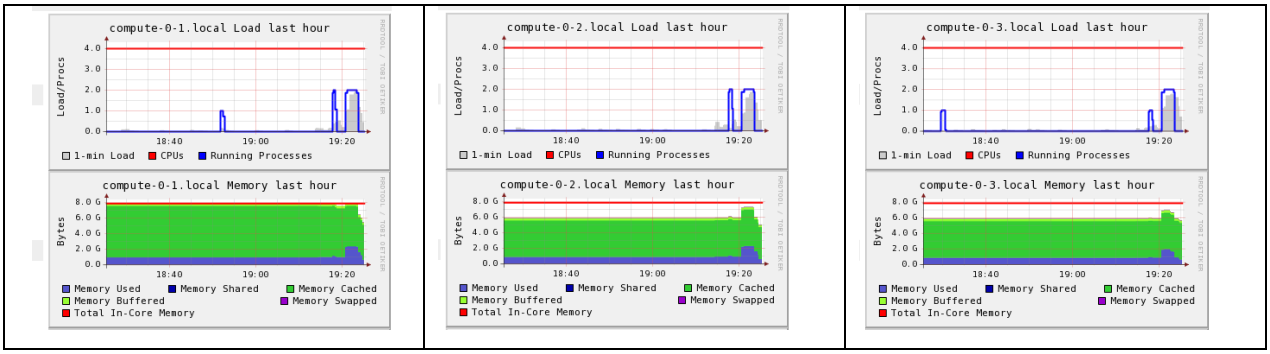
>> load(job)

>> plot(A)

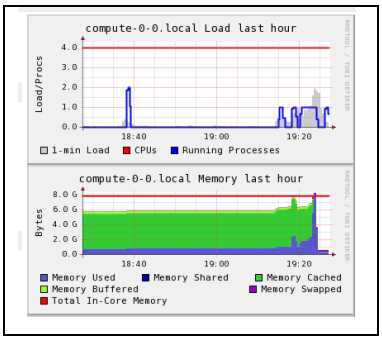
Pasul 5: Acesta este ultimul pas, si este vorba de “distrugerea” rulajului:

>> destroy(job)

Nota: Utilizatorul poate verifica stadiul rulajelor folosind unul dintre instrumentele de monitorizare instalate pe HPCC – CyberDyn. Dupa cum puteti vedea programul ruleaza pe 3 noduri de calcul (nodul 1, 2 si 3), folosind cate doua nuclee de calcul pe fiecare server si in plus clientul foloseste un singur nucleu de calcul aferent unui nod de calcul separat.



Nodul 0 foloseste doar un singur nucleu de calcul pentru client.



5.6. UN EXEMPLU DE CALCUL SPMD: REGULA TRAPEZULUI PENTRU APROXIMAREA UNEI INTEGRALE

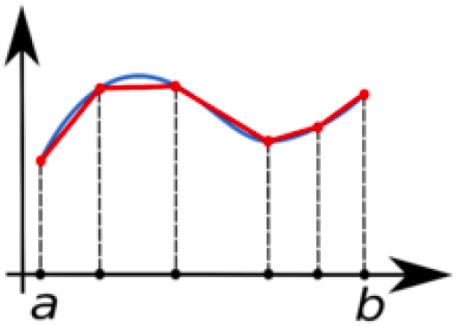


Figura 8. Aproximarea unei integrale pe un interval [a, b] folosind regula trapezului.

Formularea matematica a regulii trapezului este urmatoarea:

$$\int_a^b f(x)\,dx \approx \left(\frac{1}{2}f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right) * \frac{b-a}{n-1}$$

Intervalul cosiderat este [0,1], iar acest interval o sa fie impartit in mod egal in 4 workeri (de exemplu worker-ul numarul 3 o sa aiba un subinterval [1/2; 3/4]). Primul pas care trebuie facut, esta ca fiecare worker sa isi delimiteze propriul interval:

```
fprintf ( 1, ' Integration limits for each worker:\n' );

spmd
        a = ( labindex - 1 ) / numlabs;
        b = labindex / numlabs;
end
```

Nota: a{1} reprezinta modul in care clientul recunoaste o variabila aferenta worker-ului numarul 1. Clientul poate sa citeasca si sa rescrie aceasta variabila. In MATLAB numele acestei variabile este “composite variable”. Workerii por sa citeasca acest tip de variabila, dar nu o pot modifica.

Urmatoarea secventa de cod printeaza intervalul de calcul al integralei pentru fiecare worker in parte:

```
spmd
        a = ( labindex - 1 ) / numlabs;
        b = labindex / numlabs;
        fprintf ( 1, ' A = %f, B = %f\n', a, b );
end
```

Urmatoarea secventa de cod este folosita de catre client, care printeaza toate subintervalele de calcul:

```
spmd
        a = ( labindex - 1 ) / numlabs;
        b = labindex / numlabs;
end
        for i = 1 : 4 <-- "numlabs" wouldn't work here!
        fprintf ( 1, ' A = %f, B = %f\n', a{i}, b{i} );
end
```

In acest moment, fiecare worker poate sa isi calculeze propria parte din integrala:

```
spmd
        x = linspace ( a, b, n );
        fx = f ( x );
        quad_part = ( b - a ) / ( n - 1 ) * ( 0.5 * fx(1) + sum(fx(2:n-1)) + 0.5 * fx(n) );
        fprintf ( 1, ' Partial approx %f\n', quad_part );
end
```

Ultima parte a calculului este facuta de catre client, care combina rezultatele partiale ale workerilor:

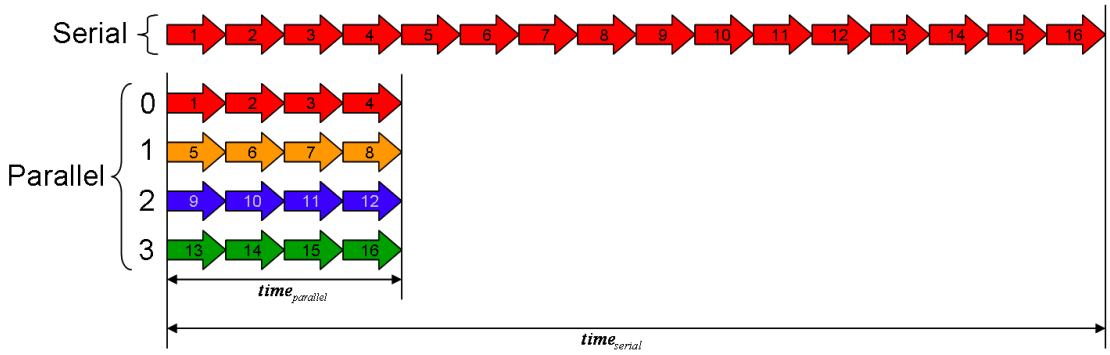
```
quad = sum ( quad_part{1:4} );
fprintf ( 1, ' Approximation %f\n', quad );
function value = quad_integral ( n )
fprintf ( 1 , ' Compute limits \n ' ) ;
spmd
        a = ( labindex - 1 ) / numlabs ;
        b = labindex / numlabs ;
        fprintf ( 1 , ' Lab %d works on [%f ,%f ] .\ n ' , labindex , a , b ) ;
end
fprintf ( 1 , ' Each lab estimates part of the integral .\n ' ) ;
spmd
        x = linspace ( a , b , n ) ;
        fx = f ( x ) ;
        quad_part = ( b - a ) * ( fx ( 1 ) + 2 * sum ( fx ( 2 : n-1) ) + fx ( n ) ) . . .
        / 2.0 / ( n - 1 ) ;
        fprintf ( 1 , ' Approx %f\n ' , quad_part ) ;
end
quad = sum ( quad_part {:} ) ;
fprintf ( 1 , ' Approximation = %f\n ' , quad )
return
end
```

Pentru a executa aceasta functie, programele SMPD se executa in acelasi mod ca si programele PARFOR:

```
matlabpool open local 4
        n = 10000;
        value = quad_integral ( n );
matlabpool close
```

5.7. MAI MULTE EXEMPLE DE PROGRAME MATLAB FOLOSIND SPMD

In continuare, prezentam o modalitate de calcul a numarului pi cu ajutorul metodei Monte-Carlo. In imaginea de mai jos se prezinta care este diferenta dintre un cod serial Monte-Carlo si un cod paralel Monte-Carlo. Fiecare sageata reprezinta o singura simulare Monte-Carlo. Dupa cum se poate vedea prin paralelizarea unui cod se obtine o diminuare a timpului total de calcul, proportional cu numarul de procesoare (sau workeri in cazul Matlab) specificat (4 workeri in exemplul de mai jos).



Exemplul 1: schema de calcul seriala

cat piMCserial.m

```
function [time]=piMCserial(N)
tic
n=0;
for j=1:N
   x=rand; y=rand;
   if (x^2+y^2)<=1, n=n+1; end
end
mypi=4*n/N
error=abs(pi-mypi)
time=toc;
```

Acest cod se poate rula astfel (de la promptul unei sesiuni interactive matlab)

```
piMCserial(1000000000)
```

Exemplul 1: schema de calcul paralela

cat piMCparallel.m

```
function [time]=piMCparallel(N)
tic
matlabpool open 4
matlabpool('addfiledependencies', {'piMCparallel.m'})
spmd
   n=0;
   for j=(labindex-1)*(N/numlabs)+1:labindex*(N/numlabs)
      x=rand; y=rand;
      if (x^2+y^2)<=1, n=n+1; end
   end
mypi=4*gplus(n)/N %global sum
error=gop(@max, abs(pi-mypi)) %global max
end
matlabpool close
time=toc;
```

Acest cod se poate rula astfel (de la promptul unei sesiuni interactive matlab)

```
piMCparallel (1000000000)
```

Rezultatul acestui exercitiu este sintetizat in tabelul de mai jos. Dupa cum se poate vedea, folosind un numar de workeri din ce in ce mai mare, timpul de calcul se diminueaza considerabil. Acesta este avantajul major al programarii in paralel.

Number of workers	Wall time (sec.)
1	1488
2	795
4	409
8	204

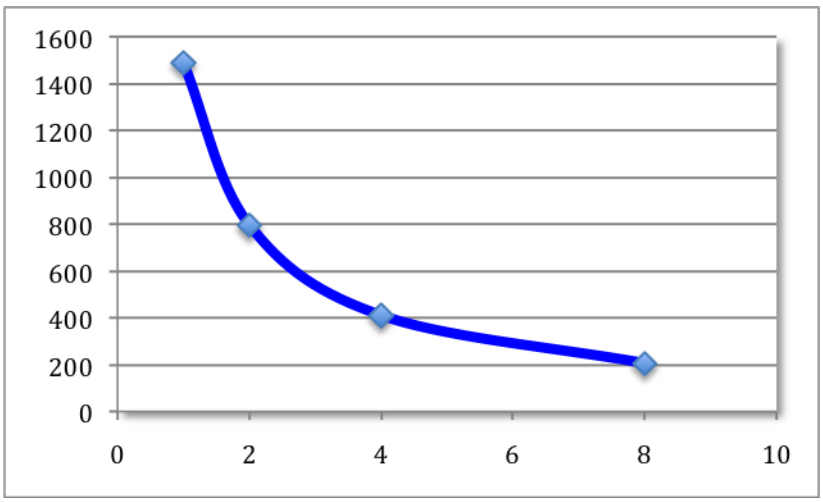


Figura 9. Reducerea timpului de calcul (axa vericala in secunde) pentru aproximarea numarului pi prin metoda Monte-Carlo folosind un numar mare de workeri (axa orizontala).

5.8. MATLAB SI INTEGRAREA CU SUN GRID ENGINE (SGE) PENTRU A SUBMITE JOBURI DISTRIBUITE TIP BATCH FOLOSIND HPCC – CYBERDYN

In continuare, o sa prezentam in detaliu procedura corecta de a trimite catre rulare pe nodurile de calcul ale HPCC – CyberDyn un cod serial scris in Matlab.

Pasul 1: Acesta etapa cade in principal in sarcina administratorului de sistem. Este nevoie sa se copieze functia decode, submit si scriptul wrapper in directorul in care este instalat Matlab. Daca Matlab este instalat intr-o locatie impartita (de exemplu /cm/share/Matlab) pe toate nodurile sistemului de calcul HPCC – CyberDyn, atunci aceste functii trebuie copiate acolo.

 # cd /usr/local/MATLAB/R2011a/toolbox/local/
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeWrapper.sh .
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeDecodeFunc.m .
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeSubmitFcn.m .
 # cd /export/apps/MATLAB/R2011a/toolbox/local/
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeWrapper.sh .
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeDecodeFunc.m .
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeSubmitFcn.m .
*# ls /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeParallel**
sgeParallelDecode.m sgeParallelSubmitFcn.m sgeParallelWrapper.sh
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeParallel /usr/local/MATLAB/R2011a/toolbox/local/*
 # cp /state/partition1/apps/MATLAB/R2011a/toolbox/distcomp/examples/integration/old/sge/sgeParallel /export/apps/MATLAB/R2011a/toolbox/local/*

Nota: in cazul utilizarii SGE a programului de submitere de job-uri, este nevoie sa existe workeri liberi. In acest sens, este nevoie sa se contacteze administratorul de sistem.

Pasul 2: In MATLAB, utilizatorul normal, poate incerca testul simplu de trimitere a unui job catre nodurile de calcul:

>> sched = findResource('scheduler', 'type', 'generic')

Pe terminal o sa apara urmatorul mesaj:

---

sched =
GENERIC Scheduler Information
==========

Type : generic
ClusterOsType : unix
ClusterSize : Inf
DataLocation : /home/vladcm/MATLAB
HasSharedFilesystem : true

- Assigned Jobs

Number Pending : 0
Number Queued : 0
Number Running : 0
Number Finished : 0

- Scheduler Specific Properties

ClusterMatlabRoot :
ParallelSubmitFcn : [ ]
SubmitFcn : [ ]
GetJobStateFcn : [ ]
CancelJobFcn : [ ]
CancelTaskFcn : [ ]
DestroyJobFcn : [ ]
DestroyTaskFcn : [ ]

Introduceti apoi urmatoarele comenzi de la tastatura:

```
>> set(sched,'DataLocation','/home/vladcm/MATLAB')
>> set(sched,'HasSharedFilesystem',true)
>> set(sched,'ClusterMatlabRoot','/share/apps/MATLAB/R2011a')
>> set(sched,'SubmitFcn',@sgeSubmitFcn)
>> j = createJob(sched);
>> createTask(j, @sum, 1, {[1 1]});
>> createTask(j, @sum, 1, {[2 2]});
>> createTask(j, @sum, 1, {[3 3]});
>> submit(j);
```

Mai jos puteti vedea mesajele afisate pe terminal in cazul in care submiterea unui job pe nodurile de calcul folosind SGE s-a facut cu success:

Submitting task 1
Job output will be written to: /home/vladcm/MATLAB/Job1_Task1.out
QSUB output: Your job 5050 ("Job1.1") has been submitted

Submitting task 2
Job output will be written to: /home/vladcm/MATLAB/Job1_Task2.out
QSUB output: Your job 5051 ("Job1.2") has been submitted

Submitting task 3
Job output will be written to: /home/vladcm/MATLAB/Job1_Task3.out
QSUB output: Your job 5052 ("Job1.3") has been submitted

Pasul 3: In momentul in care programul s-a incheiat, este necesara aducerea rezultatului programului pe nodul principal:

```
>> results = getAllOutputArguments(j);
>> results
```

results =

[4]

[2]

[6]

Pentru a monitoriza rulajele este nevoie sa folositi in terminal comanda “qstat”. In capitolele urmatoare o sa prezentam in detaliu comezile SGE printer care si comanda qstat.

5.9. CATEVA EXEMPLE IN CARE UTILIZATORUL ISI CREAZA PROPRIA FUNCTIE SI APOI FOLOSESTE SGE PENTRU A TRIMITE CATRE RULARE ACEASTA FUNCTIE PE NODURILE DE CALCUL ALE SISTEMULUI HPCC – CYBERDYN.

Pasul 1: creati o functie matlab numita, de exemplu, **test_submit_Matlab_SGE_CyberDyn.m** in felul urmator:

```
function test_submit_Matlab_SGE_CyberDyn()
processors = 1
sched = findResource('scheduler', 'type', 'generic');
set(sched,'HasSharedFilesystem',true);
set(sched,'SubmitFcn',@sgeSubmitFcn);
set(sched,'ClusterMatlabRoot','/share/apps/MATLAB/R2011a');
% Aici schimbati urmatoarele linii in functie de directorul home de pe cyberdyn
set(j,'PathDependencies',{'/home/vladcm/MATLAB/TESTS'});
set(sched,'DataLocation','/home/vladcm/MATLAB/TESTS');
j = createJob(sched);
t = createTask(j,@test_function_Matlab,1,{16});
 j.FileDependencies={'test_function_Matlab.m'};
set(t,'CaptureCommandWindowOutput',true);
submit(j)
j.waitForState
o = j.getAllOutputArguments;
o{:}
```

Fisierul test_function_Matlab.m arata in felul urmator:

```
function a = test_function_Matlab(N)
    a = zeros(N,1);
    for(i=1:N)
      a(i) = i*i
    end
```

Pasul 2: In directorul utilizatorului normal, o sa apara o multime de fisiere de genul “Job…”. Aceste fisiere sunt create in timpul rularii programului in mod batch. Fisierul care trebuie verificat este cel cu extensia “out”. Trebuie sa arate ca mai jos in cazul in care nu sunt erori la executie:

```
                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
              Version 7.12.0.635 (R2011a) 64-bit (glnxa64)
                              March 18, 2011


 To get started, type one of these: helpwin, helpdesk, or demo.
 For product information, visit www.mathworks.com.

2011-10-27 19:19:14 | About to construct the storage object using constructor
}:"
2011-10-27 19:19:15 | Logging to /home/vladycm/MATLAB/TESTS/Job18/Task1.java.l
2011-10-27 19:19:15 | About to find job proxy using location "Job18"
2011-10-27 19:19:15 | About to find task proxy using location "Job18/Task1"
2011-10-27 19:19:15 | Completed pre-execution phase
2011-10-27 19:19:16 | About to pPreJobEvaluate
2011-10-27 19:19:16 | About to pPreTaskEvaluate
2011-10-27 19:19:16 | About to add job dependencies
2011-10-27 19:19:17 | About to call jobStartup
2011-10-27 19:19:17 | About to call taskStartup
2011-10-27 19:19:17 | About to get evaluation data
2011-10-27 19:19:17 | Begin task function
2011-10-27 19:19:17 | End task function
2011-10-27 19:19:17 | About to call taskFinish
2011-10-27 19:19:17 | About to call pPostTaskEvaluate
2011-10-27 19:19:17 | About to call pPostJobEvaluate
2011-10-27 19:19:17 | About to set job state to finished
2011-10-27 19:19:17 | About to exit MATLAB normally
```

Pasul 3: In cazul in care apare o eroare de genul celei descrise mai jos, explicatia este ca nu exista licente suficiente pentru a putea rula in paralel codul de matlab pe sistemul de calcul HPCC – CyberDyn. Este nevoie sa contactati administratorul de sistem al HPCC – CyberDyn.

```
Executing: /share/apps/MATLAB/R2011a/bin/worker
License checkout failed.
License Manager Error -4
Maximum number of users for MATLAB_Distrib_Comp_Engine reached.
Try again later.
To see a list of current users use the lmstat utility or contact your License Administrator.

Troubleshoot this issue by visiting:
http://www.mathworks.com/support/lme/R2011a/4

Diagnostic Information:
Feature: MATLAB_Distrib_Comp_Engine
```

In continuare prezentam un exemplu util in cazul in care utilizatorul doreste sa scrie datele rezultate in urma unui rulaj, intr-un fisier *.txt.

```
function test_submit_Matlab_SGE_CyberDyn()
   processors = 1
   sched = findResource('scheduler', 'type', 'generic');
   set(sched,'DataLocation','/home/vladcm/MATLAB/TESTS');
   set(sched,'HasSharedFilesystem',true);
   set(sched,'ClusterMatlabRoot','/share/apps/MATLAB/R2011a');
   set(sched,'SubmitFcn',@sgeSubmitFcn);
   j = createJob(sched);
   set(j,'PathDependencies',{'/home/vladcm/MATLAB/TESTS'});
   t = createTask(j,@test_function_Matlab,1,{16})
   j.FileDependencies={'test_function_Matlab.m'};
   set(t,'CaptureCommandWindowOutput',true);
   submit(j)
   j.waitForState
   o = j.getAllOutputArguments;
   o{:}
```

Functia 'test_function_Matlab.m' arata ca mai jos. Cel mai important lucru aici este faptul ca trebuie specificata explicit toata calea catre fisierul de iesire *.txt. Altfel, datele vor fi scrise in /home/$USER (de exemplu /home/vladcm).

```
function a = test_function_Matlab(N)
    fid = fopen('/home/vladcm/MATLAB/TESTS/testparallel_output.txt', 'w');
    a = zeros(N,1);
    parfor(i=1:N)
   a(i) = i*i
    end
    fprintf(fid, '%d\n', a(N));
    fclose(fid);
```

Urmatorul sript nu apeleaza functia, ci ruleaza o functie interna numita ‘system’, care este folosita pentru a afisa numele complet al nodului de calcul pe care ruleaza aplicatia trimisa spre rulare prin intermediul organizatorului de rulaje SGE:

```
function test_submit_Matlab_SGE_CyberDyn()
   processors = 1
   sched = findResource('scheduler', 'type', 'generic');
   set(sched,'DataLocation','/home/vladcm/MATLAB/TESTS');
   set(sched,'HasSharedFilesystem',true);
   set(sched,'ClusterMatlabRoot','/share/apps/MATLAB/R2011a');
   set(sched,'SubmitFcn',@sgeSubmitFcn);
   j = createJob(sched);
   set(j,'PathDependencies',{'/home/vladcm/MATLAB/TESTS'});
   t = createTask(j,@system,2,{{'hostname'}})
   j.FileDependencies={'test_function_Matlab.m'};
   set(t,'CaptureCommandWindowOutput',true);
   submit(j)
   j.waitForState
   o = j.getAllOutputArguments;
   o{:}
```

Pe terminal vor aparea mesajele din figura de mai jos. Cum puteti vedea, codul se executa pe nodul de calcul numarul 26.

```
t =

Task ID 1 from Job ID 6 Information
==================================

                    State : pending
                 Function : @system
                StartTime :
          Running Duration :

- Task Result Properties

           ErrorIdentifier :
              ErrorMessage :
Submitting task 1
Job output will be written to: /home/vladycm/MATLAB/TESTS/Job6_Task1.out
QSUB output: Your job 5163 ("Job6.1") has been submitted
```

Distrugerea task-urilor si crearea unora noi

Daca se doreste crearea unor task-uri aditionale, trebuie creat in primul rand un job. Daca se doreste crearea unui nou task si asignarea lui catre "j", atunci acesta se va adauga listei de task-uri "j". Trimiterea catre rulare a unui job in acest fel rezulta in trimiterea unui task now si vechi in acelasi timp. Cel mai simplu este sa se creeze un nou job, de exemplu "new_job", asignarea acestui job unui nou task, sau utilizatorul poate distruge (sterge) task-ul "j" inainte de a crea unul nou. Comanda pentru a sterge un job este: *destroy(j).*

Limitarea marimii unui job

Sistemul de calcul HPCC – CyberDyn ofera pe moment doar 8 licente de workeri. In cazul in care utilizatorul submite un job care contine mai mult de 8 task-uri, acele task-uri peste 8 or sa intre in lista de asteptare pana cand workerii se elibereaza. Acest lucru se aplica si in cazul in care alti utilizatori submit joburi si nu exista workeri disponibili. CyberDyn ofera numai 8 task-uri folosind Distributed Computing Engine asa cum a fost descris in capitolele precedente. Toate joburile care sunt peste 8, or sa intre in lista de asteptare pana cand ceilalti workeri se elibereaza.

Nota: Functia "findJob" are ca argument un scheduler. In cazul in care se foloseste scheduler 'local', utilizatorul poate sa testeze urmatoarea comanda:

```
s = findResource('scheduler', 'type', 'local');
```

```
ans =

    Jobs: 9-by-1
    ============

  #  Job ID        State       FinishTime  UserName  #tasks
 ------------------------------------------------------------
  1       6     finished  Oct 20 20:41:17   vladycm       1
  2       7     finished  Oct 20 20:42:36   vladycm       3
  3       8      pending                -   vladycm       0
  4       9     finished  Oct 20 20:44:19   vladycm       1
  5      13      pending                -   vladycm       1
  6      14      pending                -   vladycm       0
  7      15     finished  Oct 25 11:09:27   vladycm       1
  8      16     finished  Oct 25 11:09:52   vladycm       1
  9      17     finished  Oct 25 11:11:01   vladycm       1
```

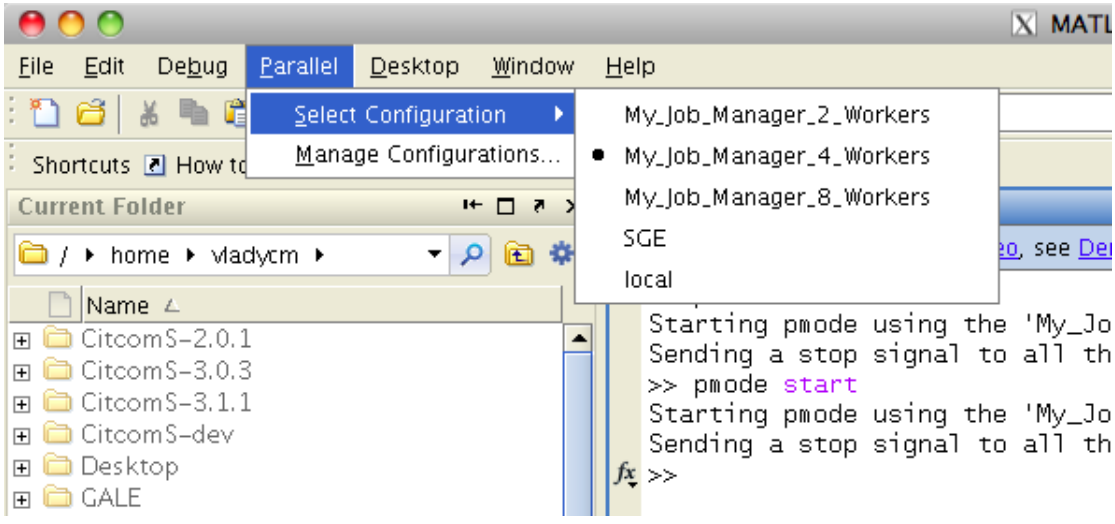
In acest fel se pot sterge rulaje care sunt in asteptare (pending). Acelasi lucru se poate aplica si pentru joburile care s-au terminat (finished).

```
>> pending_jobs = findJob(s,'State','pending','UserName','vladycm')

pending_jobs =

    Jobs: 3-by-1
    ============

  #  Job ID        State       FinishTime  UserName  #tasks
 ------------------------------------------------------------
  1       8      pending                -   vladycm       0
  2      13      pending                -   vladycm       1
  3      14      pending                -   vladycm       0


>> destroy(pending_jobs)
>> clear pending_jobs
>> findJob(s)

ans =

    Jobs: 6-by-1
    ============

  #  Job ID        State       FinishTime  UserName  #tasks
 ------------------------------------------------------------
  1       6     finished  Oct 20 20:41:17   vladycm       1
  2       7     finished  Oct 20 20:42:36   vladycm       3
  3       9     finished  Oct 20 20:44:19   vladycm       1
  4      15     finished  Oct 25 11:09:27   vladycm       1
  5      16     finished  Oct 25 11:09:52   vladycm       1
  6      17     finished  Oct 25 11:11:01   vladycm       1
```

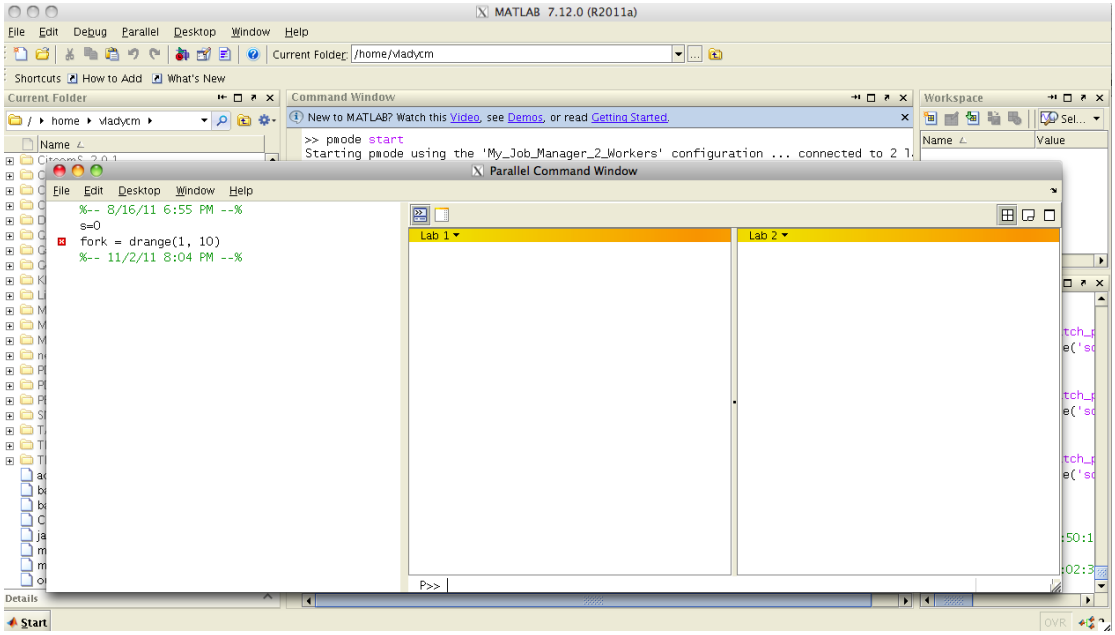
5.10. MODUL DE LUCRU PMODE

PMODE permite executia in paralel a programelor Matlad de o maniera interactiva. PMOD realizeaza acest lucru definind si ruland un program in paralel si deschide o serie de ferestre de lucru (Parallel Command Window) care sunt conectate la fiecare worker in parte. Workerii primesc comenzi in Parallel Command Window, le executa si trimit inapoi o serie de mesaje catre Parallel Command Window.

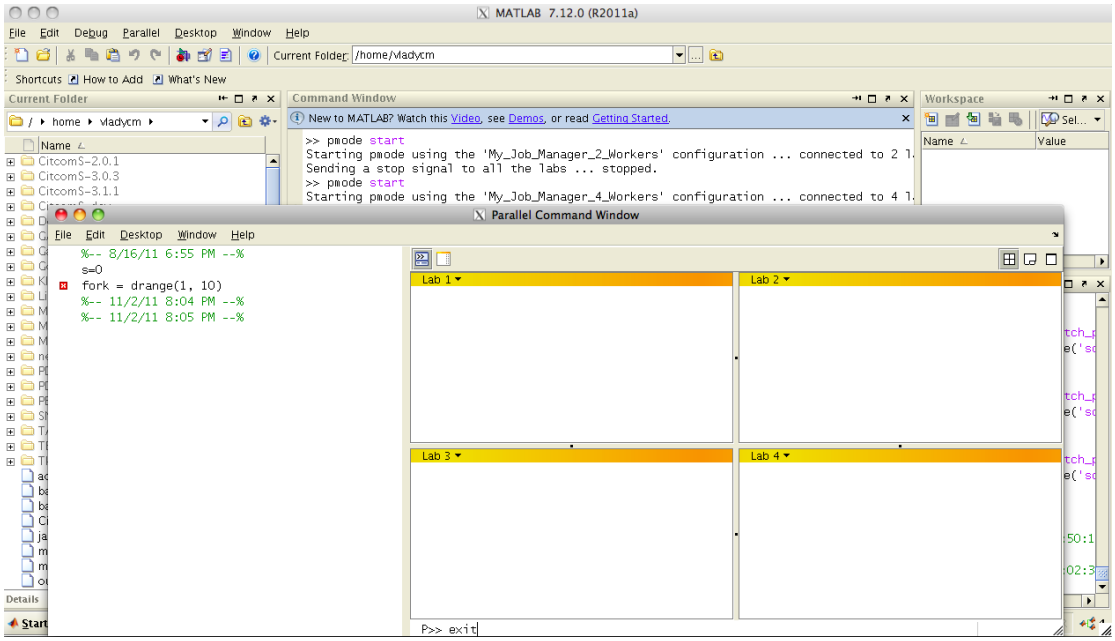
De exemplu, comanda *pmode start 'local' 2* initiaza modul pmode folosind doar 2 workeri. Comanda *pmode exit* sterge programele in paralel si inchide sesiunea pmode. In cele ce urmeaza o sa exemplificam modul de lucru cu pmode. Mai intai utilizatorul trebuie sa selecteze o configuratie paralela. Mai jos este ilustrat un exemplu in care sunt folositi doar 2 workeri (din cei 8 disponibili pe sistemul de calcul HPCC – CyberDyn).



In momentul in care se executa comanda “pmode start” de la terminalul Matlab, se initiaza o sesiune pmode cu doar 2 workeri, asa cum se poate vedea in imaginea de mai jos.

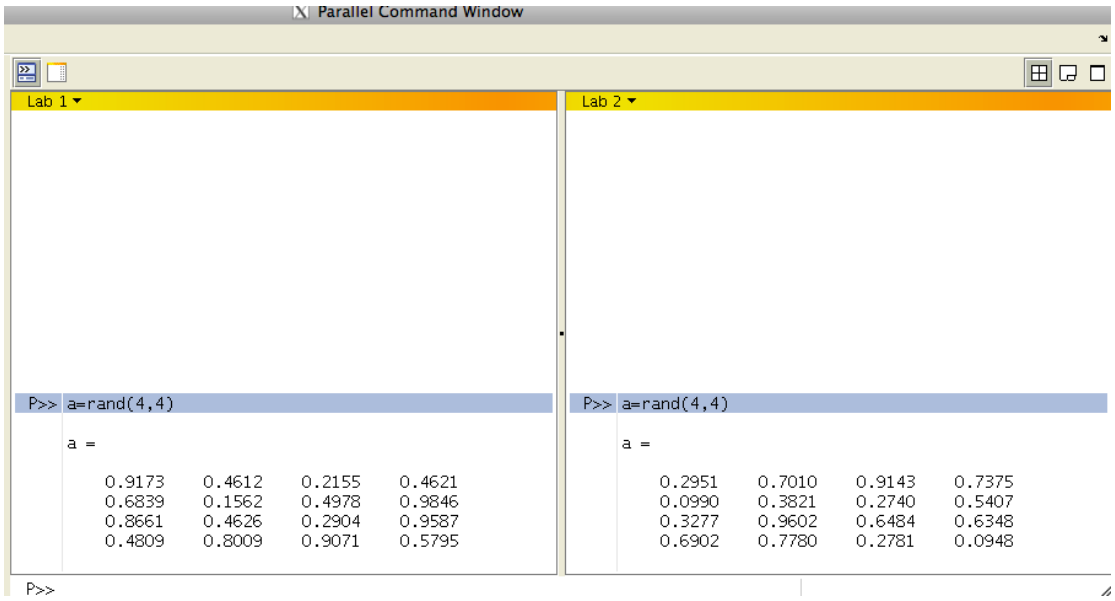


Daca am selectat un job manager cu 4 workeri, atunci comanda “pmode start” va initia o sesiune cu 4 workeri, asa cum se poate vedea in figura de mai jos.

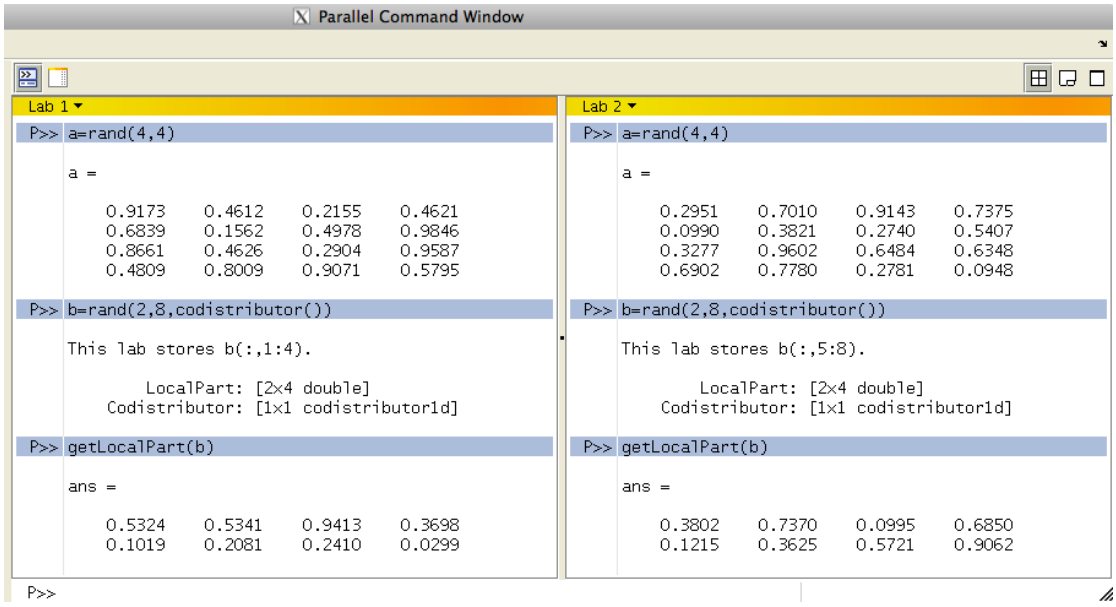


In exemplele ce urmeaza o sa folosim doar 2 workeri intr-o sesiune pmode.

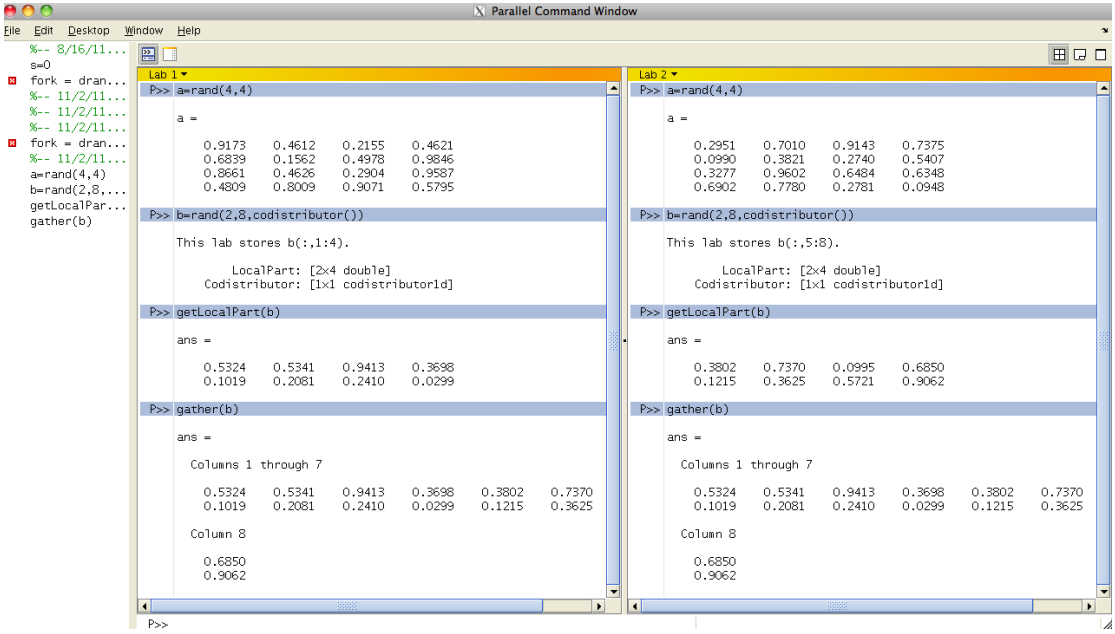
Daca se introduce de la tastatura comanda care generaza o matrice aleatoare de 4x4 elemente, se observa ca fiecare worker detine o matrice distincta.



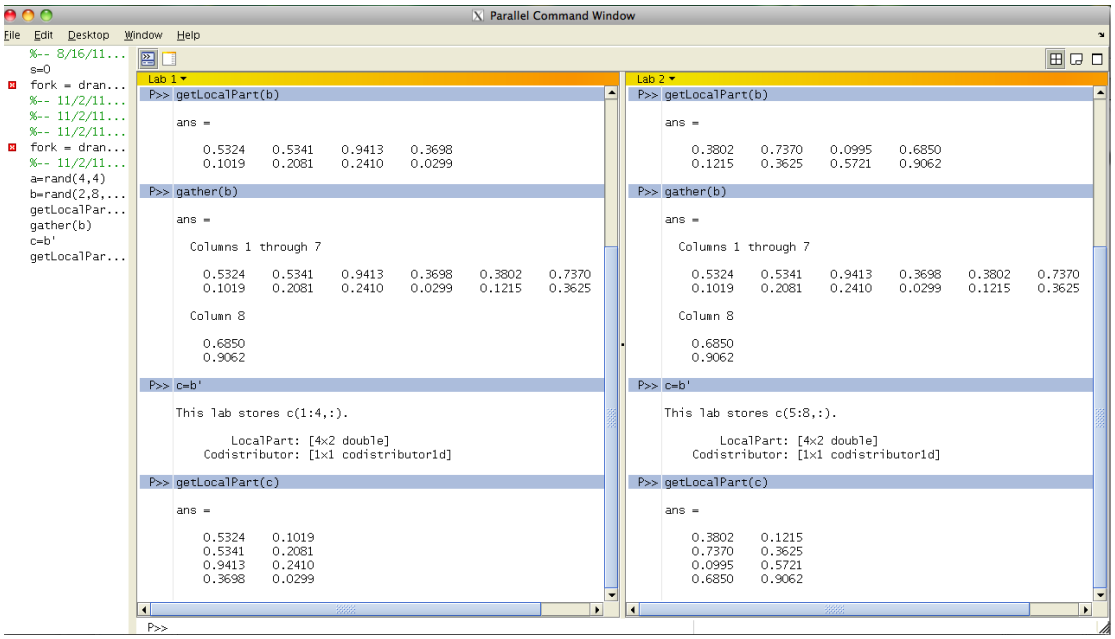
Daca se foloseste optiunea **codistributor()**, atunci fiecare worker o sa detina cate o parte din matricea 4x4. Cu alte cuvinte, primul worker (lab1) o sa stocheze primele doua randuri din matricea “a”, iar worker-ul numarul 2 (lab2) o sa stocheze celelalte 2 randuri.



Comanda “getLocalPart(b)” afiseaza pe ecran pentru fiecare worker partea de matrice care ii corespunde. Comanda “gather(b)” combina cele doua parti si formeaza o singura matrice, asa cum se poate vedea in imaginea de mai jos.



O facilitate oferita de SPMD este ca in cazul in care se doreste sa se calculeze inversa unei matrici distribuite pe cei doi workeri, inversa este calculata separat pe cei 2 workeri, asa cum se poate vedea in imaginea de mai jos.



6. CONCLUZII

In urma acestui studiu au reiesit urmatoare concluzii principale:

- Sistemul HPCC CyberDyn a fost configurat astfel incat sa poata sa beneficieze de cele mai recente dezvoltari in ceea ce priveste administrarea de sisteme de calcul in paralel tip cluster HPCC. Sistemul de management instalat si configurat pe HPCC CyberDyn este Bright Cluster Management, unul dintre cele mai avansate soft-uri din domeniu la ora actual,

- In urma testelor de performanta (cu ajutorul testului Linpack) s-a obtinut un randament real de aproape 70%, comparabil cu sisteme similare din lista www.top500.org,

-HPCC CyberDyn foloseste ca soft de programare de rulaje soft-ul liber SGE, unul dintre cele mai robuste soft-uri libere din domeniu,

-Pe HPCC CyberDyn s-a instalat, configurat si testat soft-ul Matlab in paralel. Testele de performanta au aratat ca folosind 8 nuclee in paralel, timpul de executie al unui cod paralelizat a scazut de la 1455 de secunde (1 nucleu) la 204 secunde (8 nuclee),

-Toate testele de mai sus arata ca sistemul HPCC CyberDyn este configurat pentru a functiona la capacitate optima, si este pregatit pentru a se incepe pregatirea/rularea modelelor numerice cu elemente finite in vederea studierii originii geodinamice a zonei seismogene Vrancea.

7. REFERINTE BIBLIOGRAFICE

Bright Cluster Management: http://www.brightcomputing.com

HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers: http://www.netlib.org/benchmark/hpl/

Sun Grid Engine: http://gridengine.sunsource.net

Matlab: www.mathworks.com