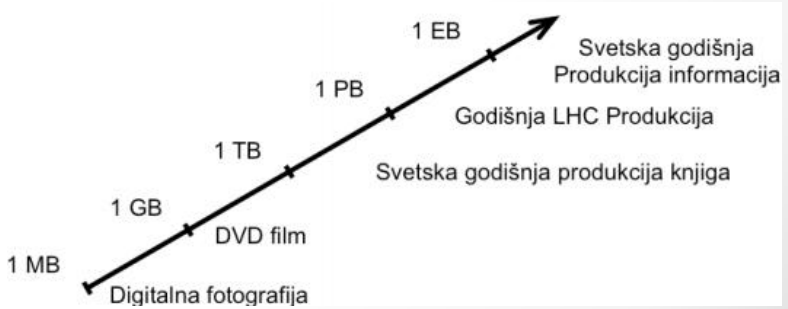
Klaster računari

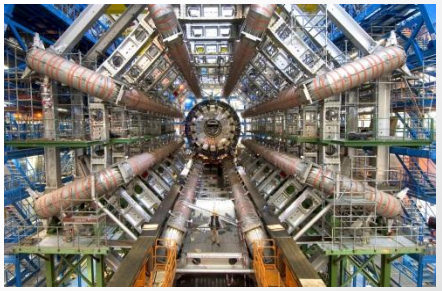
Motivacija

- Nauka postaje sve više digitalna i zahteva obradu džinovskih količina podataka (big data)
- Fizika elementarnih čestica je jedna od važnih naučnih disciplina koja koristi velike količine računarskih resursa
  - Velike međunarodne kolaboracije
  - Velike količine podataka iz eksperimenta
  - Large Hadron Collider (LHC) u CERN-u
    - 40 miliona sudara u sekundi
- Digitalni univerzum
  - 4 zetabajta ($4 * 10^{21}$) informacija
  - Do 2020. u svetu će biti 40 zetabajta



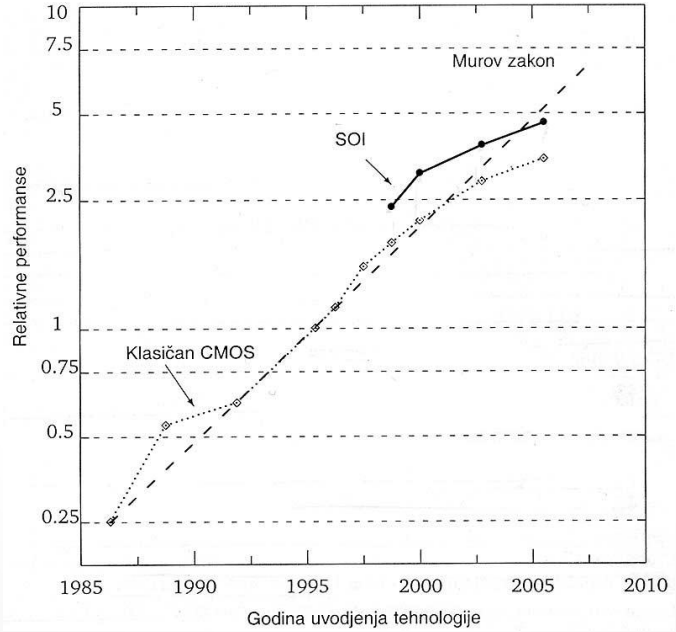
Motivacija

- Numeričke simulacije u nauci?
- Da bismo numerički rešavali teorije koje drugačije ne mogu biti rešene
- Da bismo izvršavali virtuelne eksperimente
- Da bi procenili smislenost i opravdanost ideja i novih teorija
- Za potpunu simulaciju jednog sudara (događaja) u ATLAS detektoru potrebno je ~15 min.
- Analiza podataka na LHC zahteva kompijutersku snagu ekvivalentnu 100.000 današnjih PC procesora

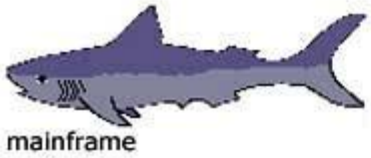


Kako ubrzati zahtevne proračune?

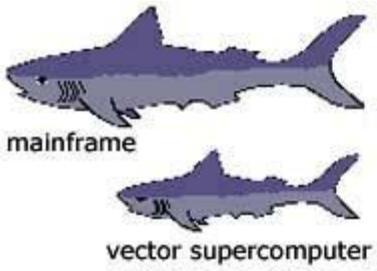
- Korišćenjem boljeg hardwere-a
- Murov zakon
- Optimizacija algoritma
- Korišćenjem distribuiranih računarskih sistema



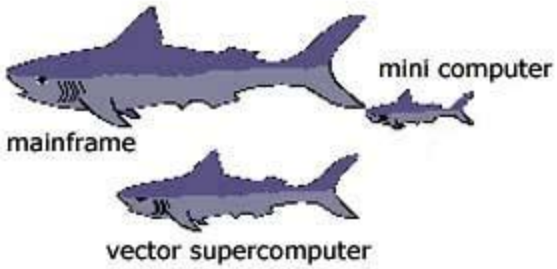
Klaster u odnosu na druge sisteme



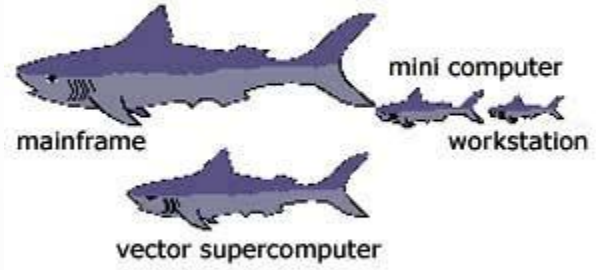
Klaster u odnosu na druge sisteme



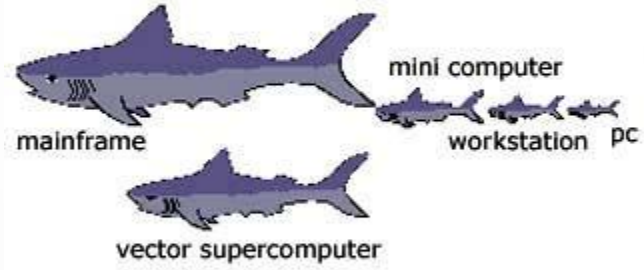
Klaster u odnosu na druge sisteme



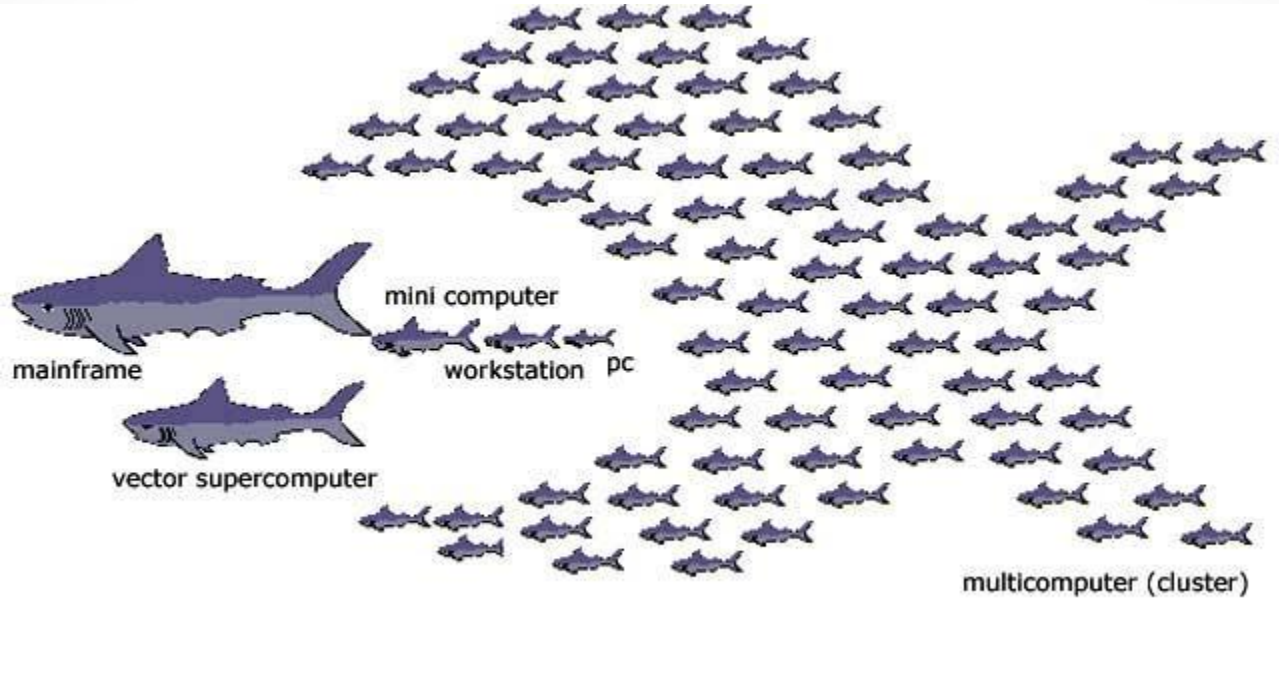
Klaster u odnosu na druge sisteme



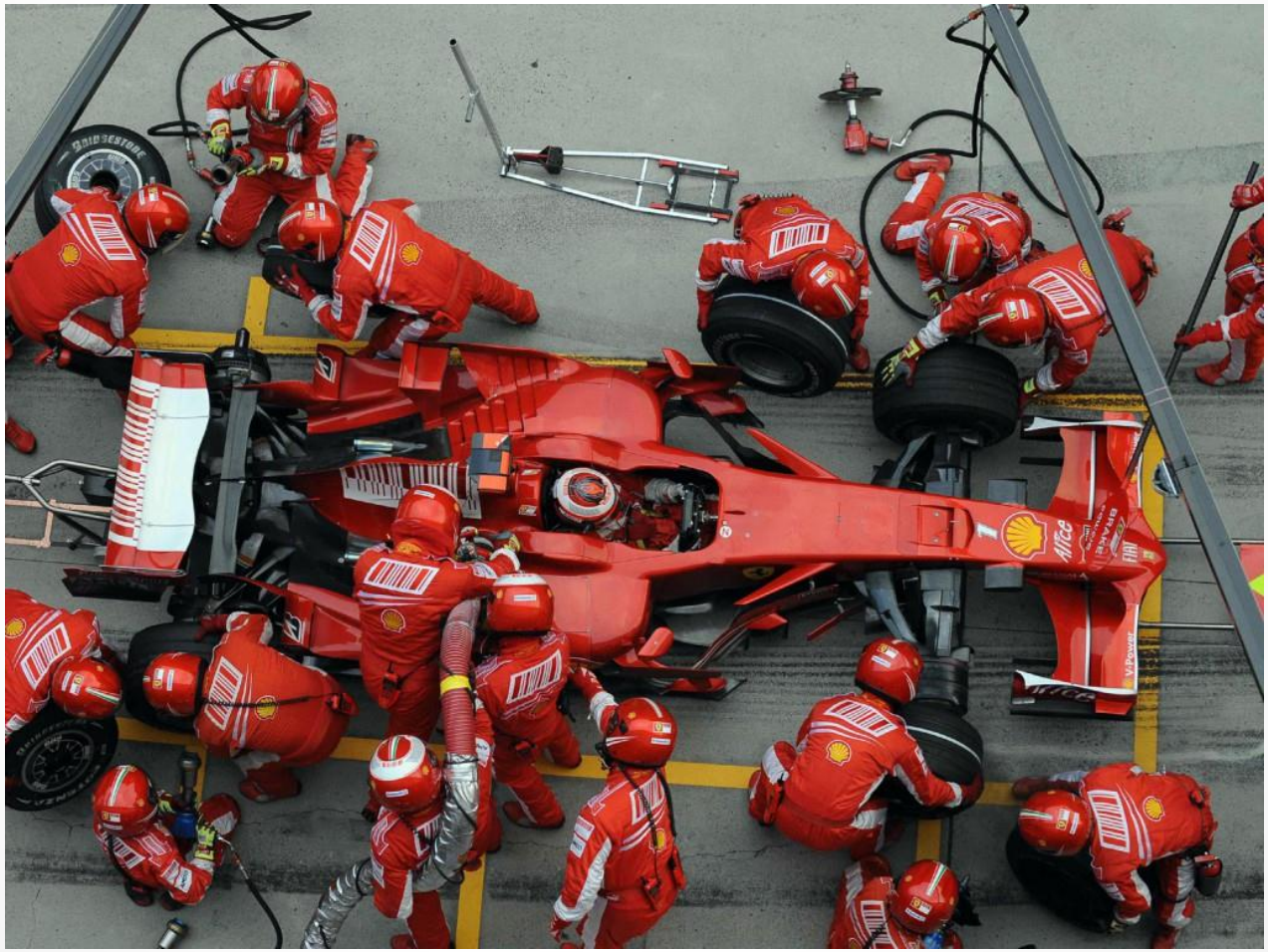
Klaster u odnosu na druge sisteme



Klaster u odnosu na druge sisteme



Primer problema gde su neophodne visoke performanse



Klasteri

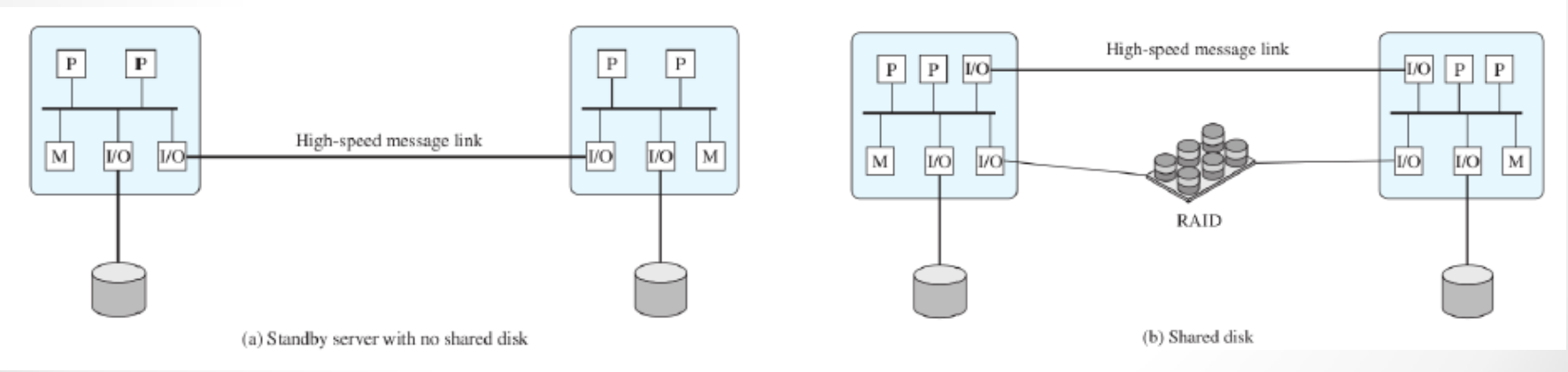
- Skup međusobno povezanih potpunih računara koji rade zajedno kao ujedinjeni resurs, stvarajući iluziju kao da su jedan računar
  - Potpun računar?
- Svaki računar u klasteru obično se označava kao čvor

Prednosti?

- Apsolutna skalabilnost
- Rastuća skalabilnost
- Visoka raspoloživost
- Superiorni odnos cena/performanse

Klasifikacija klastera

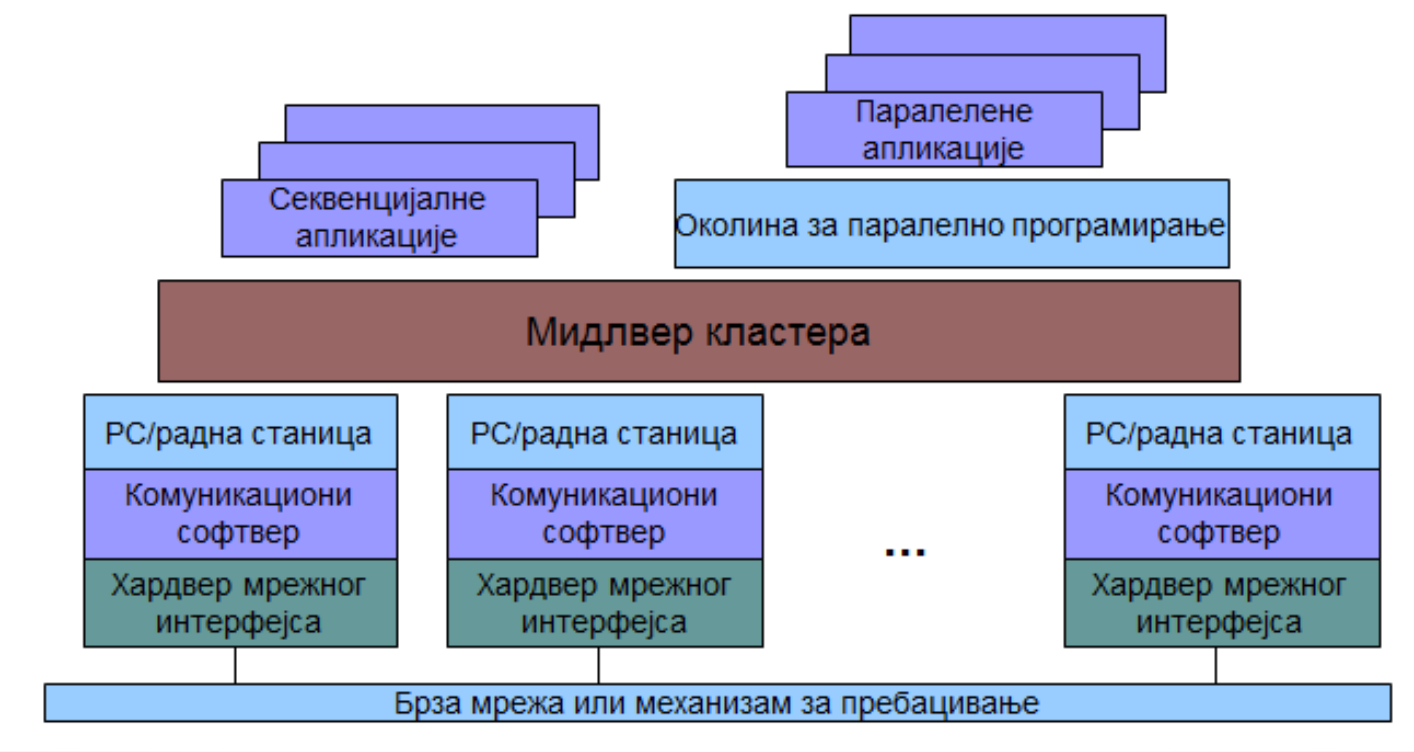
- Mogu se klasifikovati na više načina
- Najjednostavnija klasifikacija je zasnovana na tome da li računari u klasteru dele pristup istim diskovima



Pitanja projektovanja OS-a

- Da bi se u punoj meri iskoristio hardver klastera potrebno je izvršiti neka poboljšanja u OS-ima za samostalne računare
  - Upravljanje otkazima
  - Uravnotežavanje opterećenja (treba imati u vidu inkrementalnu skalabilnost klastera)
  - Paralelizacija izračunavanja
    - Paralelizovan kompajler
    - Paralelizovana aplikacija
    - Parametrizovano izračunavanje

Arhitektura klastera



Arhitektura klastera

- Midlver klastera obezbeđuje korisniku uniformnu sliku sistema (slika jedinstvenog sistema)
- Pozeljno je da midlver pruža sledeće usluge i funkcije:
  - Jedisntvena ulazna tačka
  - Jedinstvena hijerarhija fajlova
  - Jedinstvena upravljačka tačka
  - Jedinstvena virtuelna mreža
  - Jedinstveni sistem za upravljanje poslovima (job manager, npr. Torque)
  - Jedinstveni UI prostor: svaki čvor može da pristupi U/I uređaju ili disku pri čemu ne mora da zna njegovu fizičku lokaciju
  - Jedinstveni prostor procesa (na nekim sistemima)
  - Preseljenje procesa

Klasteri

- Na duže staze, prednosti klastera će verovatno rezultirati situacijom da će klasteri biti dominantni na tržištu servera visokih performansi
- Klasteri su superiorni u pogledu pouzdanosti, jer se sve komponente sistema mogu načiniti visoko redudantnim

Primer 1.1

- job.sh

```
#!/bin/bash

date
hostname
pwd
sleep 10
```

- job.sub

```
#!/bin/bash
#PBS -N PrviJob
#PBS -q batch
#PBS -l nodes=1:ppn=1
#PBS -l walltime=00:00:40
#PBS -e ${PBS_JOBID}.err
#PBS -o ${PBS_JOBID}.out

cd $PBS_O_WORKDIR

chmod +x job.sh

./job.sh >> istorija.txt
```

Torque
batch_processing_tutorial.html
pbs_user_guide.html
http://kb.iu.edu/data/avmy.html

Primer 1.2

- Parametarski posao

```
#!/bin/bash
#PBS -N PrviJob
#PBS -q batch
#PBS -l nodes=1:ppn=1:intel
#PBS -l walltime=00:00:40
#PBS -e ${PBS_JOBID}.err
#PBS -o ${PBS_JOBID}.out

cd $PBS_O_WORKDIR

chmod +x job.sh

./job.sh >> istorija${PBS_JOBID}${PBS_ARRAYID}.txt
```

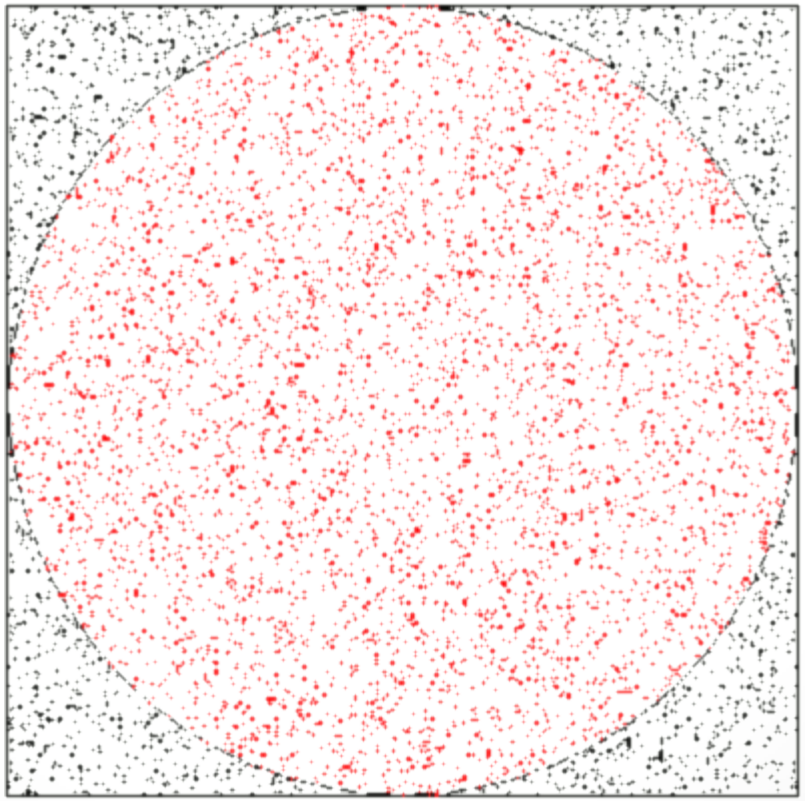
Primer 2

- job.sh

```
#!/bin/sh
#PBS -N OS2-nproc
#PBS -r n
#PBS -q batch
#PBS -l nodes=2:ppn=2

echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR

echo Running on host `hostname`
echo Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
echo `cat $PBS_NODEFILE`
# Define number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS cpus

RSLT=rslt.txt

touch $RSLT
```

Primer 3.1

- Monte carlo metod za izračunavanje broja Pi



Primer 3.2

- krug.c

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>

int main(int argc,char *argv[])
{
        long iCount = atol(argv[1]);
        long iterator;
        long hits = 0;
        double range,dx,dy;
        srand(time(NULL));
        for(iterator = 0; iterator < iCount; iterator++)
        {
                dx = ((double)rand()) / RAND_MAX - 0.5;
                dy = ((double)rand()) / RAND_MAX - 0.5;
                range = sqrt(dx*dx + dy*dy);
                if(range < 0.5) hits++;
        }

        //printf("%lf",(double)hits / iCount * 4.0);
        printf("%ld",hits);

        return 0;
}
```

Primer 3.3

- krug.sub

```
#!/bin/sh
#PBS -N ADAM_KRUG_TEST
#PBS -r n
#PBS -q batch
#PBS -l nodes=2:ppn=2

echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR
echo Running on host `hostname`
echo Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
echo `cat $PBS_NODEFILE`

# Define number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS cpus

i=1
sample=22000000

for HOST in `cat $PBS_NODEFILE`; do
FILENAME=$i.txt
ssh $HOST "cd $PBS_O_WORKDIR; gcc krug.c -o krug -lm;  touch $FILENAME; ./krug $sample > $FILENAME"
i=`echo "$i+1"|bc`
done

chmod +x collect.sh
./collect.sh $NPROCS $sample > result.txt
```

Primer 3.4

- collect.sh

```
#!/bin/sh

COUNT=$1
SAMPLE=$2
SUM=0
for I in `seq 1 1 $COUNT`
do
        echo "working on $I"
        ITEM=`cat $I.txt`
        echo "item is $ITEM"
        SUM=`echo "$ITEM + $SUM" | bc -l`
done

RESULT=`echo "$SUM / ($COUNT*$SAMPLE) * 4" | bc -l`

echo $RESULT
```

Primer 4.1 - MPI (Message Passing Interface)

- hello.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
        FILE *f;
        int id;
        int p;
        char cid[2];
        cid[1] = '\0';
        char filename[6] = {'t','e','x','t',' ','\0'};
        char filename1[6];
        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &id);
        MPI_Comm_size(MPI_COMM_WORLD, &p);
        sprintf(filename1,"%s%d",filename,id);
        f = fopen(filename1,"w");
        fprintf(f,"Hello, world, from process %d\n", id);
        fclose(f);
        MPI_Finalize();
        return 0;
}
```

Primer 4.2

- hello.sub

```
#!/bin/sh
#PBS -N HELLO_MPI
#PBS -r n
#PBS -q batch
#PBS -l nodes=4

echo Working directory : $PBS_O_WORKDIR
cd $PBS_O_WORKDIR

echo Running on host `hostname`
echo Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
echo `cat $PBS_NODEFILE`
# Define number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS cpus

module load openmpi-pmf-x86_64

mpicc hello.c -o hello.exe
mpiexec hello.exe
```