Michela Mapelli

N-body techniques for astrophysics:
Lecture 3 – N-body methods for collisionless systems

OUTLINE of this lecture:

1 – Softening

2 - Tree code (tree building)

3 – Particle Mesh (PM), Particle-Particle (PP)/PM and
   Fast Multipole Moment (FMM) algorithms
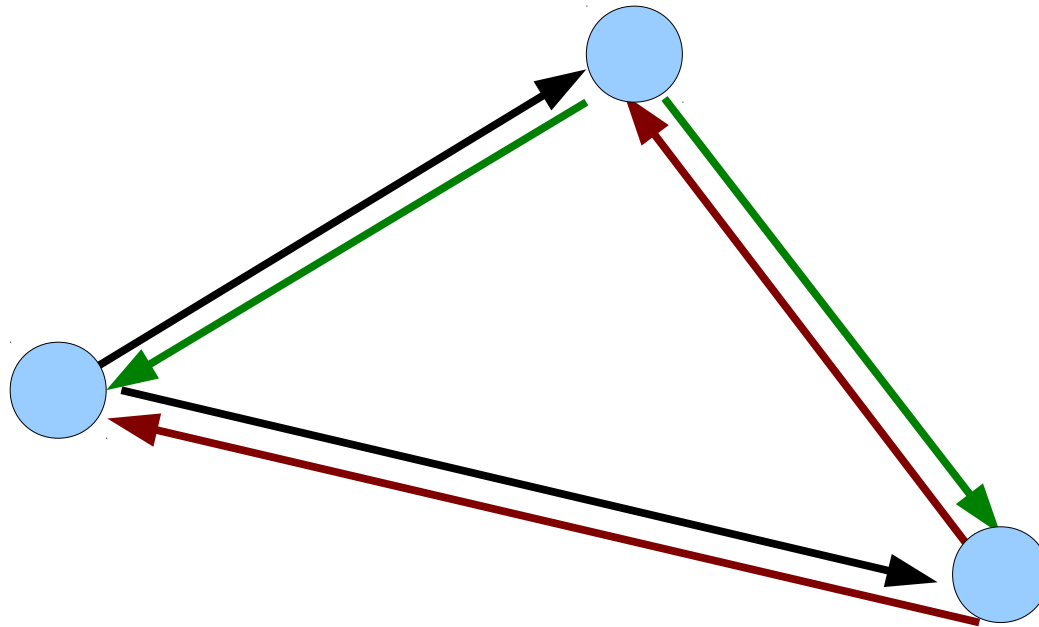
4 - Clusters of computers

5 - Example codes: ChanGa no gas

1. WHAT IS an N-Body SIMULATION?

SCALING of a NUMERICAL PROBLEM:

Numerical complexity:

$$\ddot{\vec{r}}_i = -G \sum_{j \neq i} m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3}$$

How many calculations I have to do for N-particles

3 particles
6 forces
N (N - 1)

1. WHAT IS an N-Body SIMULATION?

SCALING of a NUMERICAL PROBLEM:

Numerical complexity:

$$\ddot{\vec{r}}_i = -G \sum_{j \neq i} m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3}$$

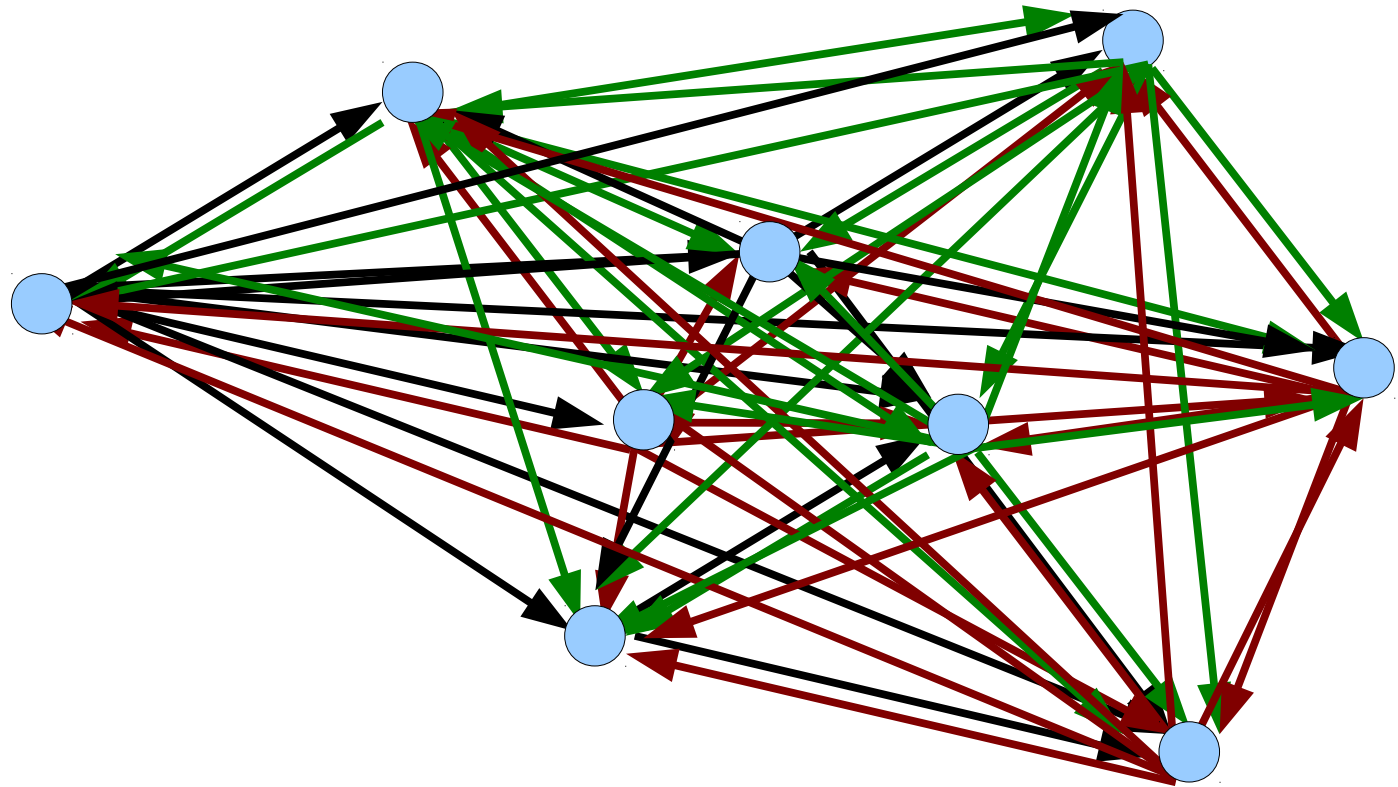**How many calculations I have to do for N-particles**

9 particles
72 forces
N (N – 1)

COMPLEXITY GROWS as N^2   - VERY FAST !!!!

1. WHAT IS an N-Body SIMULATION?

SCALING of a NUMERICAL PROBLEM:

COMPLEXITY GROWS as N^2   - VERY FAST !!!!

- CAN I REDUCE COMPLEXITY?

YES, BUT IT IS NOT ALWAYS THE RIGHT CHOICE
→ See direct vs indirect N-body LECTURE 2

- HOW CAN I REDUCE COMPLEXITY?
E.G. with BARNES-HUT TREE METHOD
and/or with MULTIPOLE EXPANSION
→ See LECTURE 3

1. SOFTENING:

In most N-body simulations (except for those in Lecture 2)

       1) MASS of a PARTICLE >> MASS of a true star, or gas, or
                         dark matter particle
       2) particles are point-like (no physical radius)

If 2 particles approach each other,
       $F(r_i\text{-}r_j) \propto (r_i - r_i)^{-2}$ becomes very large (unphysical)

→ particles spuriously scatter each other
→ wrong kinetic energy distribution in the simulation

We need a substitute of the physical radius, a radius over which the force is SOFTENED, smoothed to a smaller value :=

The SOFTENING

NB: The softening does not slow down two-body relaxation (does not affect far encounters), but the softening KILLS impulsive 3-body encounters
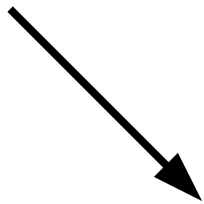→ should not be used in direct N-body (apart from limited cases),
→ must be used in large (cosmological-, galaxy cluster- galaxies, galaxy-) scale simulations

1. SOFTENING:

Calculate softening in practice:

Substituting point potential with Plummer potential (Aarseth 1963) at small scales

$$F_{ij} = -G \, m_i \, m_j \, \frac{\mathbf{r}_i - \mathbf{r}_j}{(|\mathbf{r}_i - \mathbf{r}_j|)^3}$$

$$F_{ij} = -G \, m_i \, m_j \, \frac{\mathbf{r}_i - \mathbf{r}_j}{(|\mathbf{r}_i - \mathbf{r}_j|^2 + \epsilon^2)^{3/2}}$$

Or with a more general Kernel:

$$F_{ij} = -G \, \frac{m_i \, m_j}{\epsilon^2} \, \frac{\mathrm{d}}{dt} \phi \left(\frac{|\mathbf{r}_i - \mathbf{r}_j|}{\epsilon} \right) \, \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|}$$

Dehnen 2001, MNRAS, 324, 273

1. SOFTENING:

Calculate softening in practice -2:
Which is the most reasonable value of epsilon? Sort of magic..

Rule of thumb: average distance between particles or a bit less

$$\boxed{\epsilon} = \left(\frac{4\pi}{3} R_{\mathrm{vir}}^3 \right)^{1/3} N^{-1/3}$$

↘ **Virial radius**

↘ **# of particles in simulation**

Or some more sophisticated recipe, e.g. (Dehnen 2001):

$$\epsilon = 0.017\, r_{\mathrm{s}} \left(\frac{N}{10^5} \right)^{-0.23}$$

→ **Scale radius of NFW halo**

IDEA:

We do not need to estimate the force exerted by each single particle

We can estimate the cumulative force exerted by groups of particles if they are sufficiently far

Particles that are far-off can be grouped into a single particle with mass = total mass and position/velocity = the position/velocity of the centre of mass
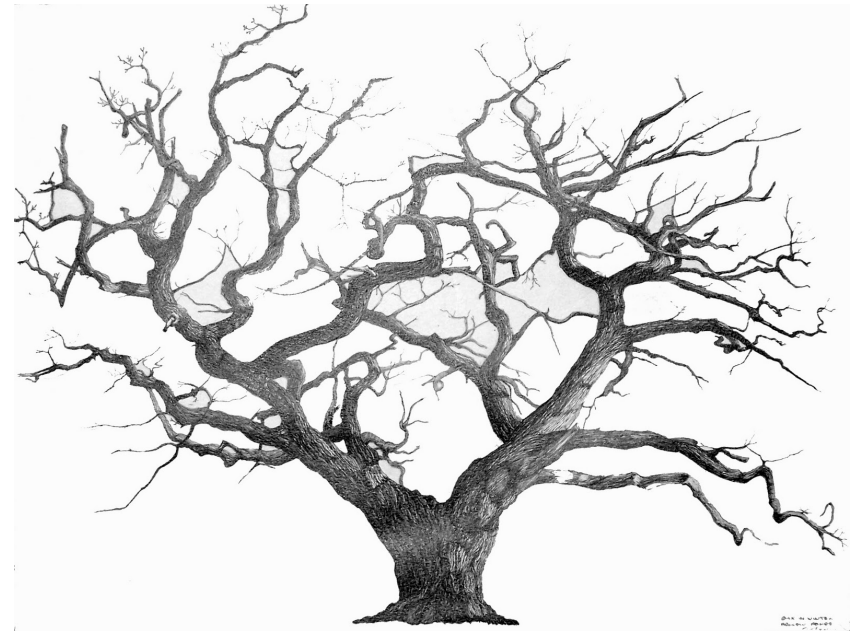
Why tree?

Divide simulation box into sub-boxes and then divide the sub-boxes into sub-sub-boxes and so on, down to the smallest sub-boxes that still have particles

The big box is the ROOT of the tree

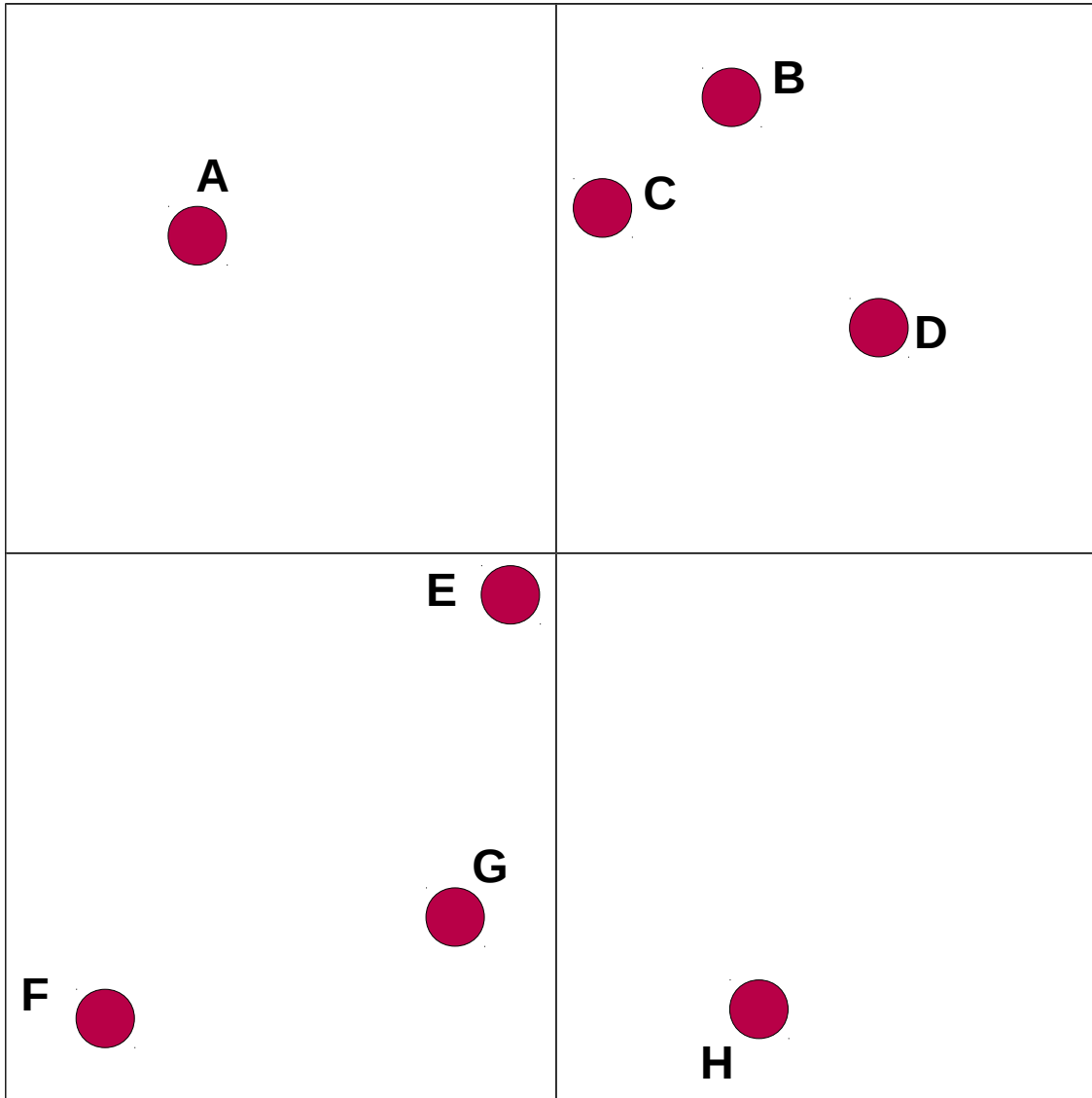**The sub-boxes are BRANCHES**
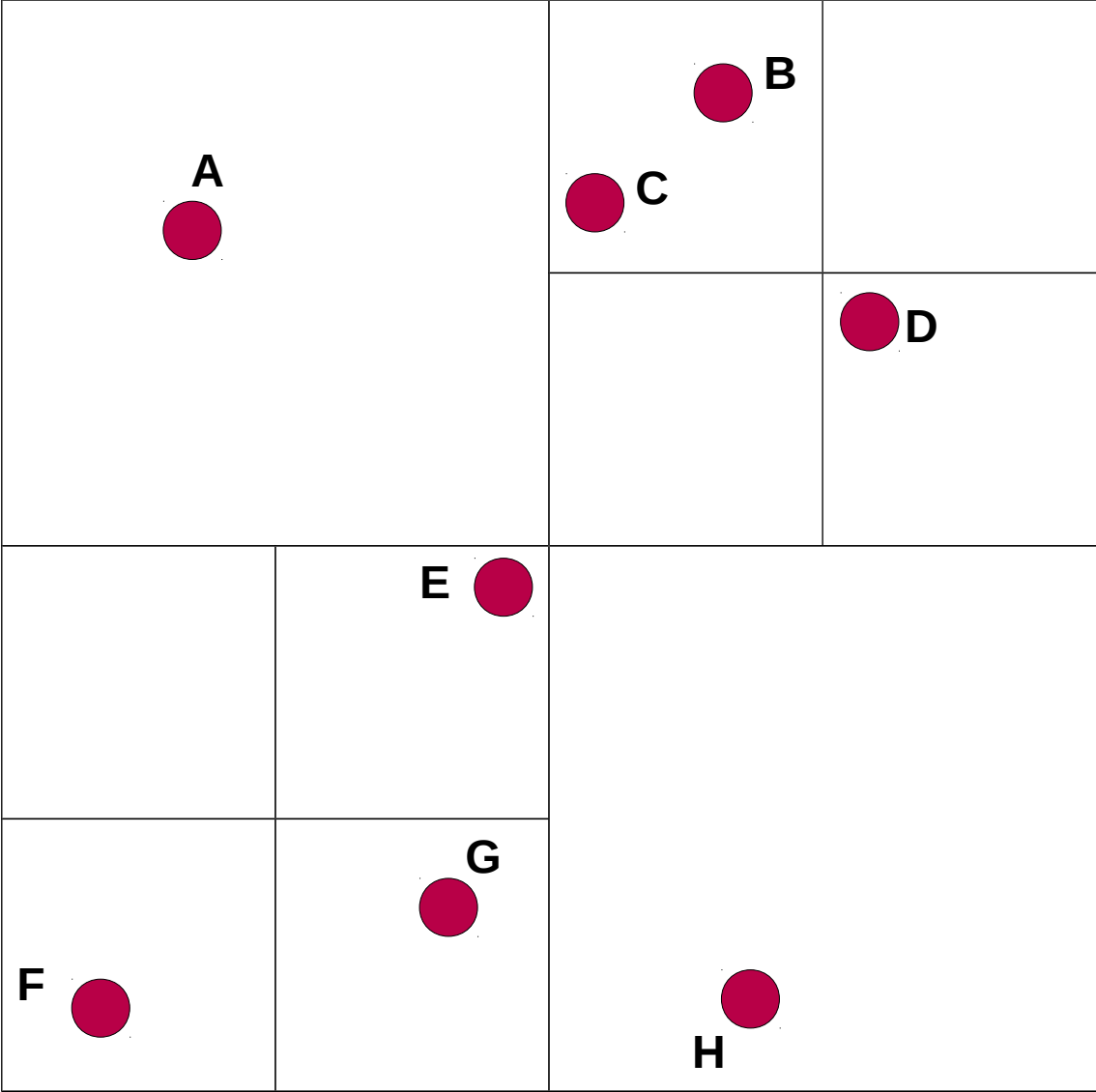
The smallest sub-boxes are LEAVES

2. TREE CODE:

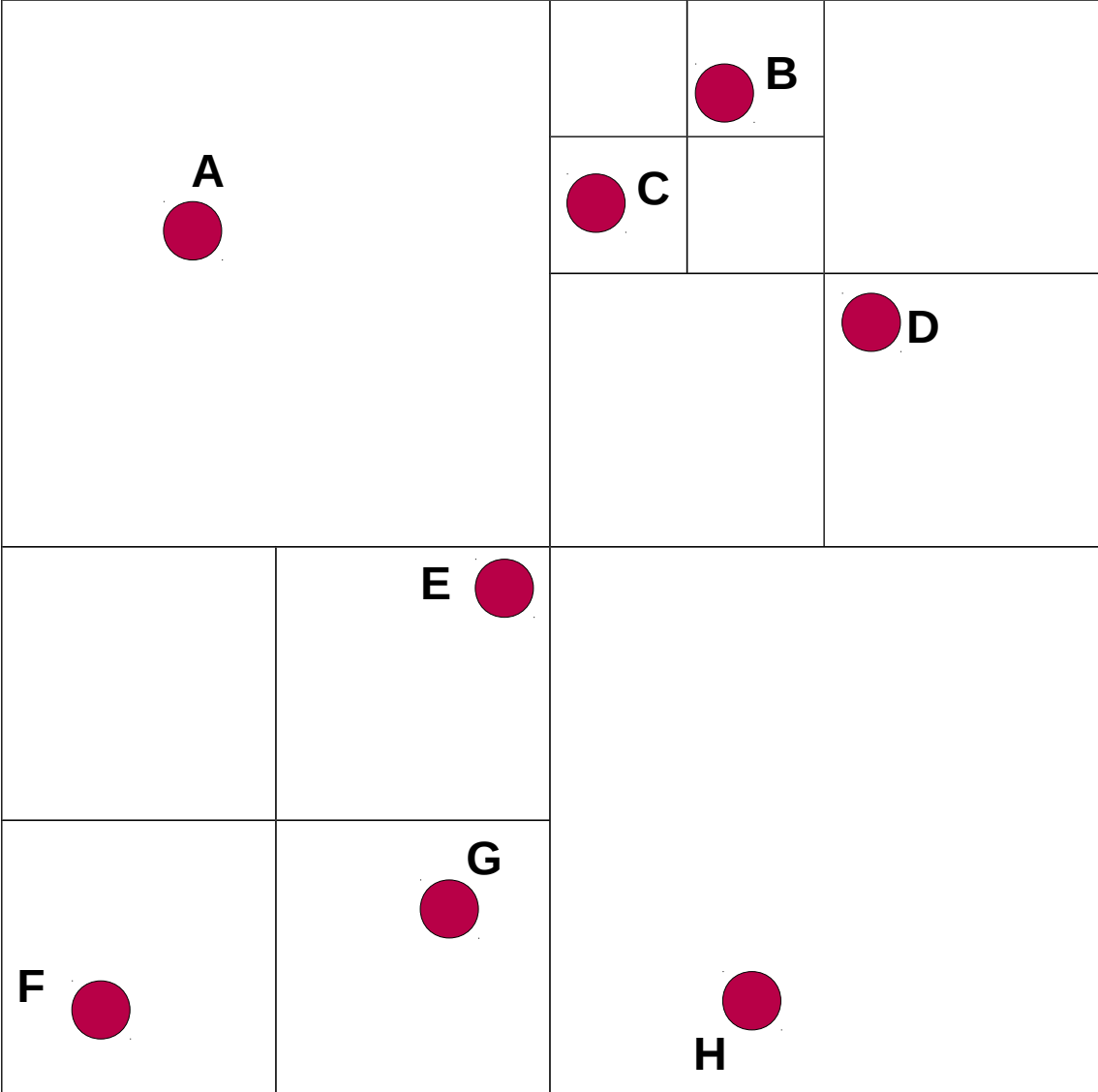## Example: quad-tree with 8 particles
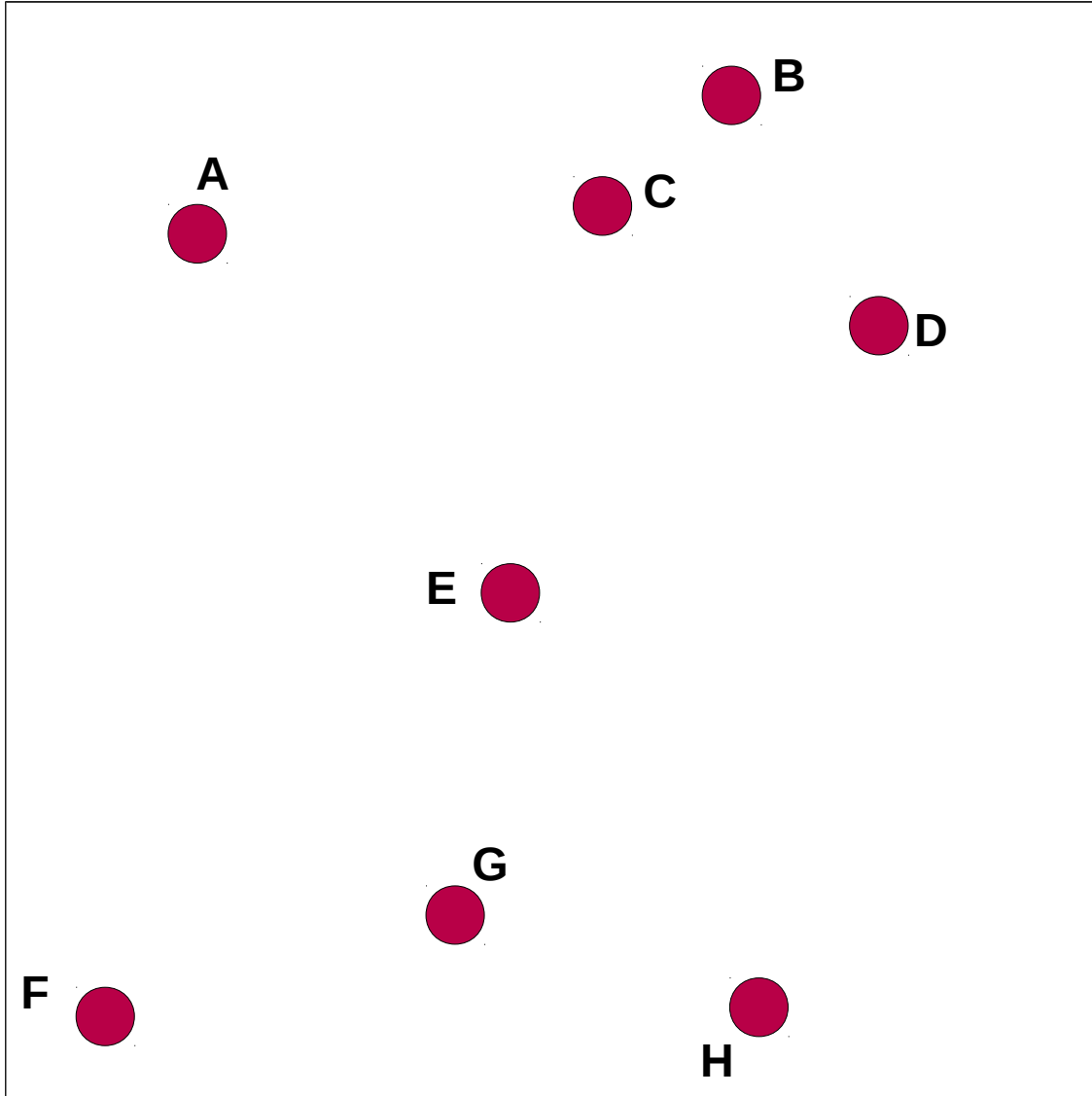
2. TREE CODE:

Example: quad-tree with 8 particles

2. TREE CODE:

Example: quad-tree with 8 particles
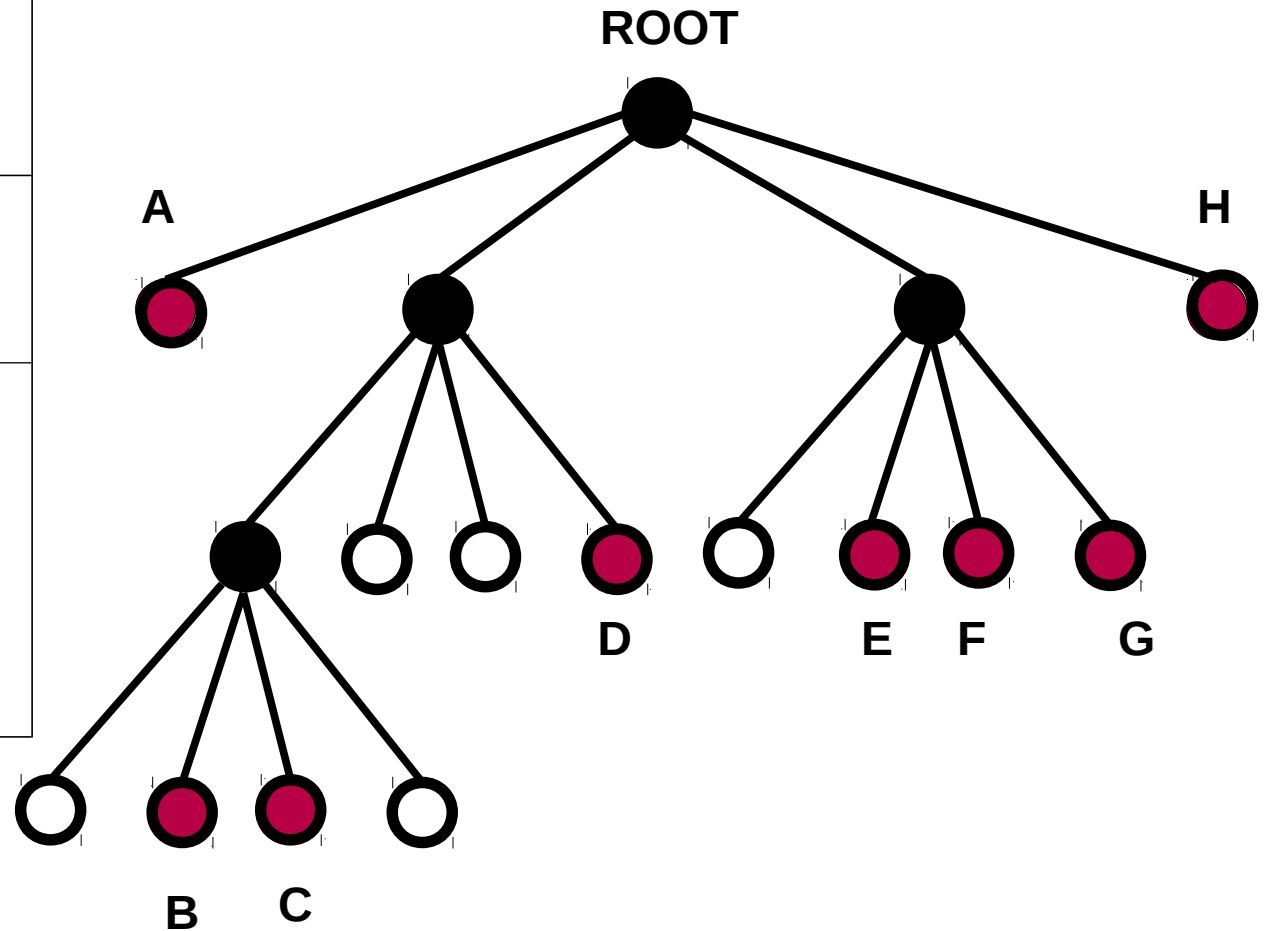
2. TREE CODE:

Example: quad-tree with 8 particles

2. TREE CODE:

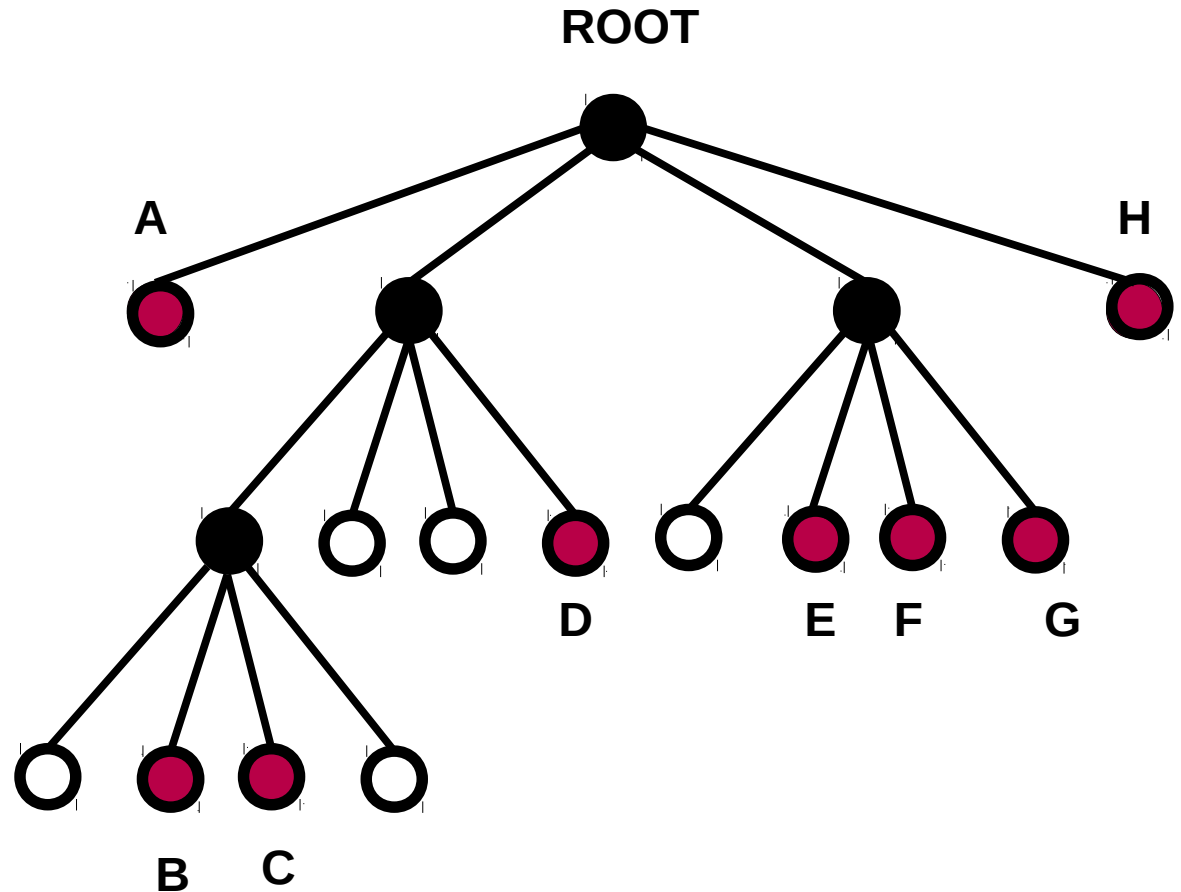## Example: quad-tree with 8 particles
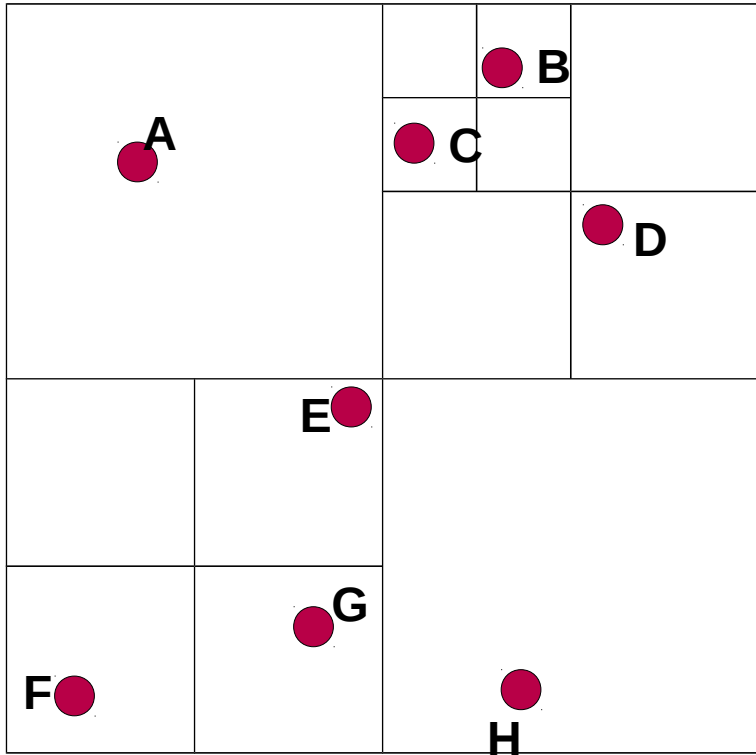
2. TREE CODE:

Force between nearby particles is calculated directly.

e.g. the force exerted by F on E

Force between distant particles is approximated with centre of mass

e.g. the force of E, F, G on D is approximated by the centre of mass of E, F and G

2. TREE CODE:

<u>HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?</u>

If   s / d   <   θ

s = width of the region represented by the considered node,
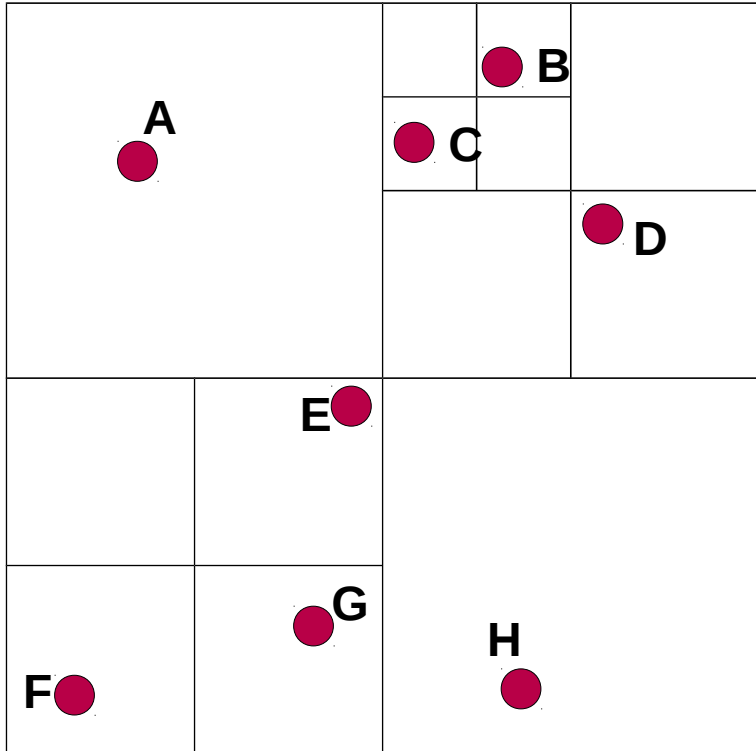d = distance between the body and the node's center-of-mass
θ = threshold value
   (θ = 0.5 commonly used in practice;
    θ = 0 means no internal node is
   treated as a single body, and the
   algorithm degenerates to brute force)

θ is called OPENING ANGLE
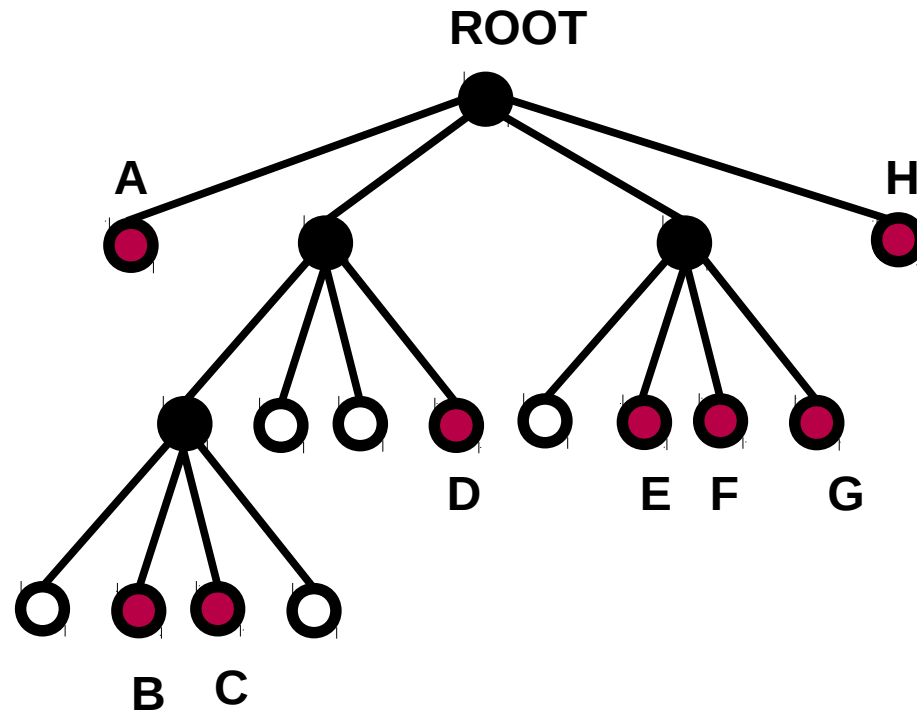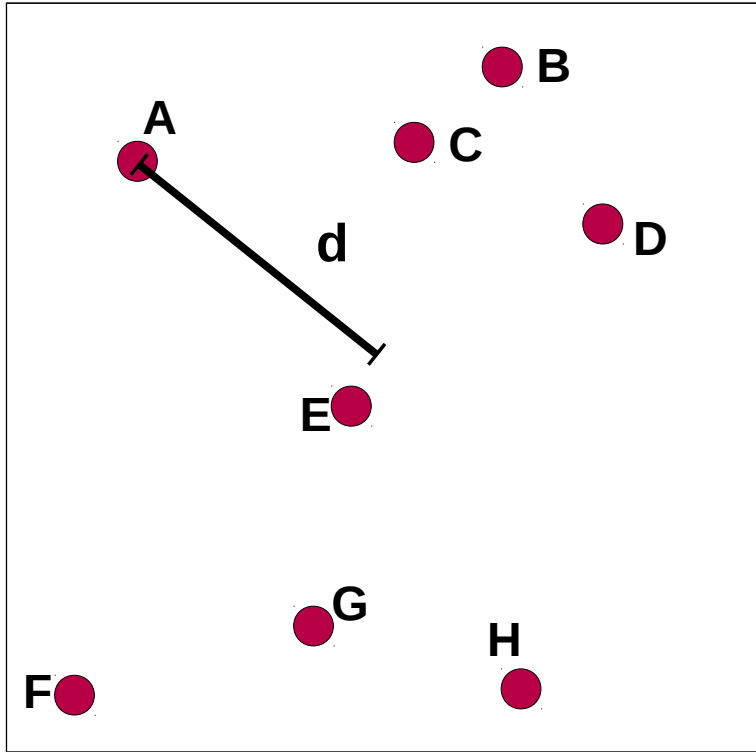
2. TREE CODE:

Example: CALCULATE FORCES ON A

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If $s / d < \theta$

2. TREE CODE:

Example: CALCULATE FORCES ON A

$s / d = 2.2 \; > \; \theta = 0.75$

I walk the tree to calculate forces (I split cells)

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If $\; s / d \; < \; \theta$

2. TREE CODE:

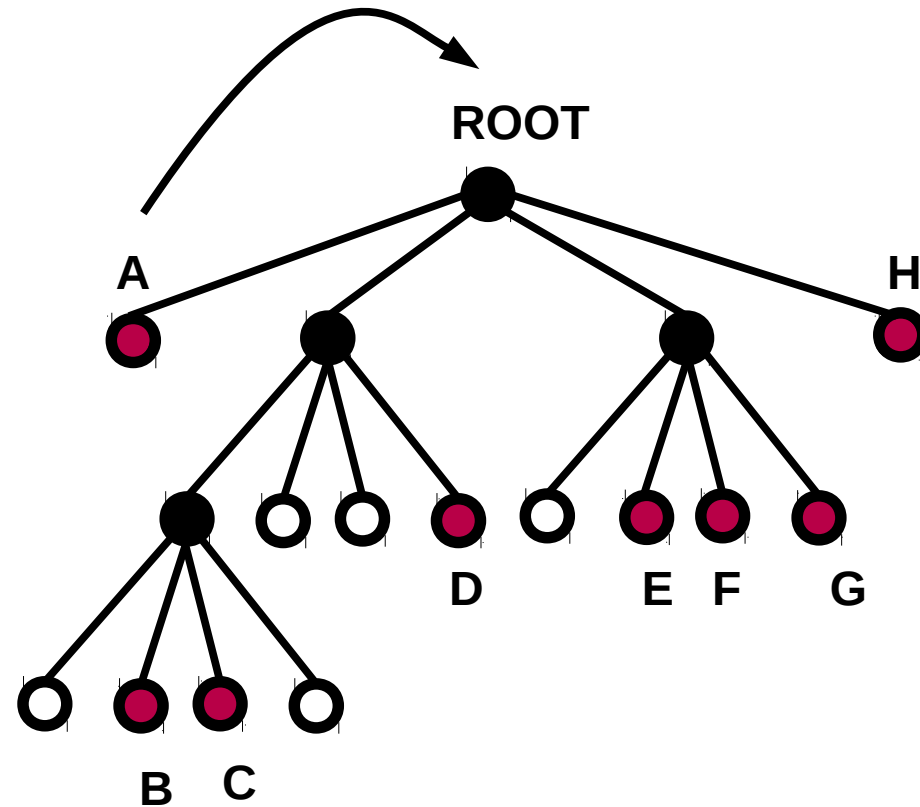*Example: CALCULATE FORCES ON A*
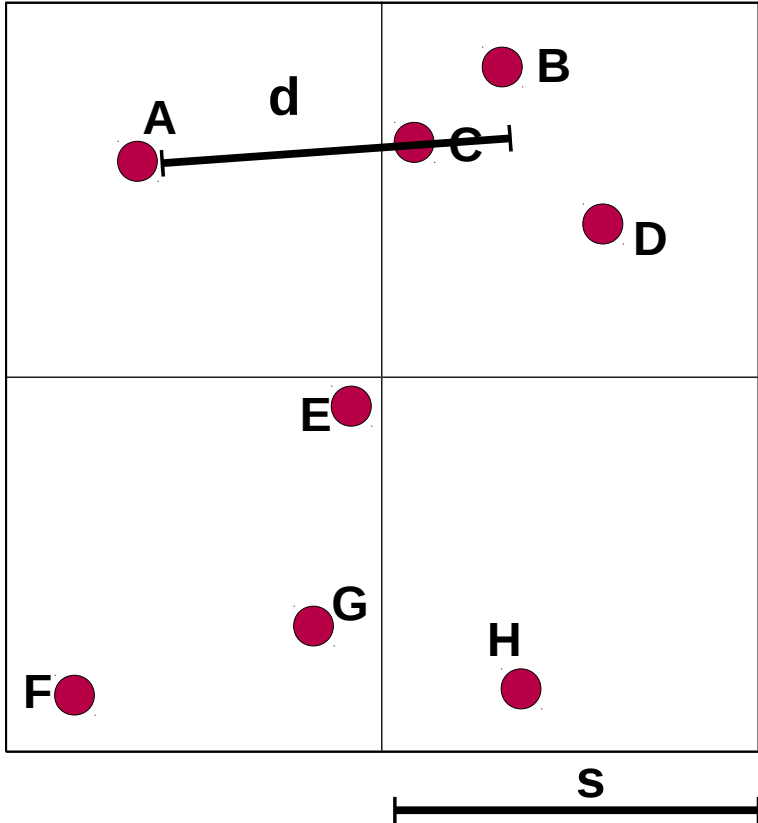
$$s / d = 1.1 > \theta = 0.75$$

I walk the tree to calculate forces (I split cells)

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If   s / d   <   θ

2. TREE CODE:

Example: CALCULATE FORCES ON A
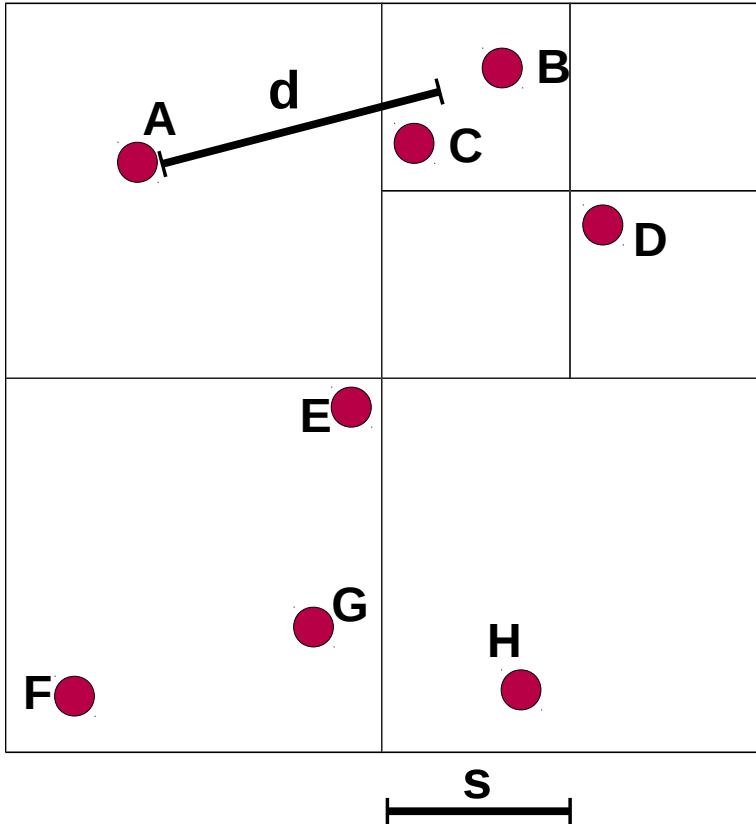
$s / d = 0.64 < \theta = 0.75$

I STOP WALKING the tree and CALCULATE the FORCE between A and the centre-of-mass of B+C

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If $s / d < \theta$

B and C are sufficiently far from A

2. TREE CODE:

Example: CALCULATE FORCES ON A
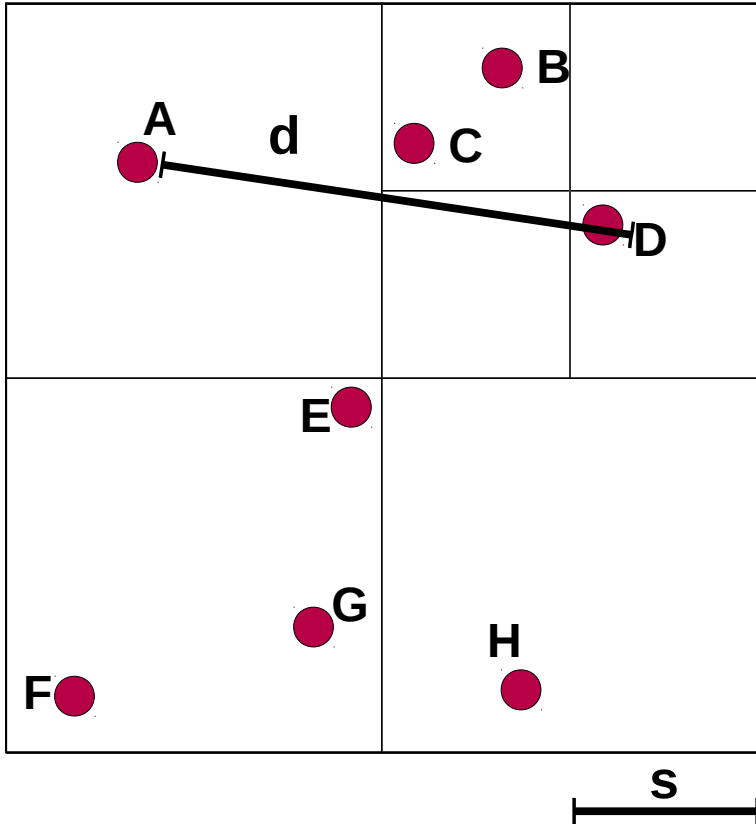
$s / d = 0.77 > \theta = 0.75$

I should split a cell, but D is already a single particle → I calculate force between A and D

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?
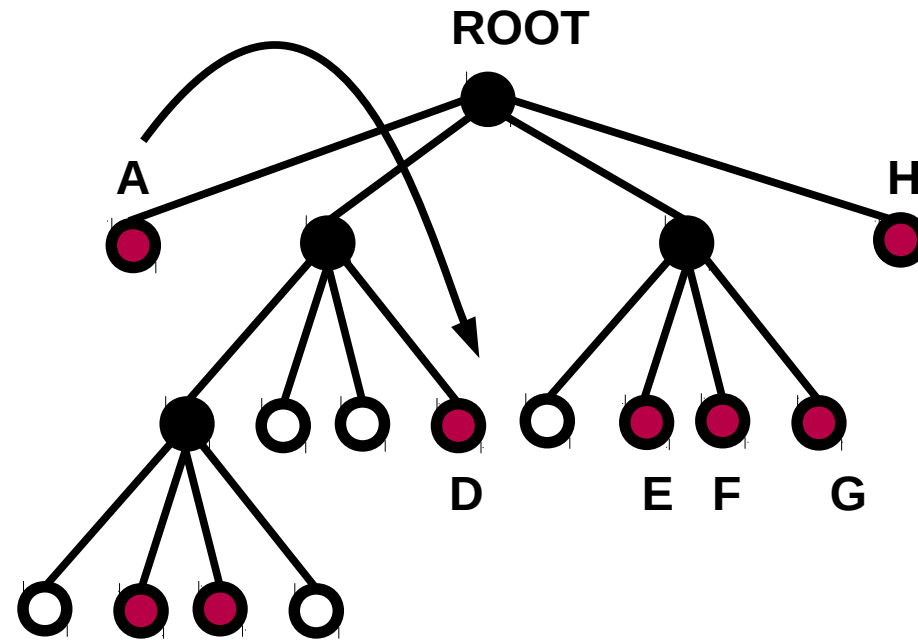
If $s / d < \theta$

2. TREE CODE:

Example: CALCULATE FORCES ON A

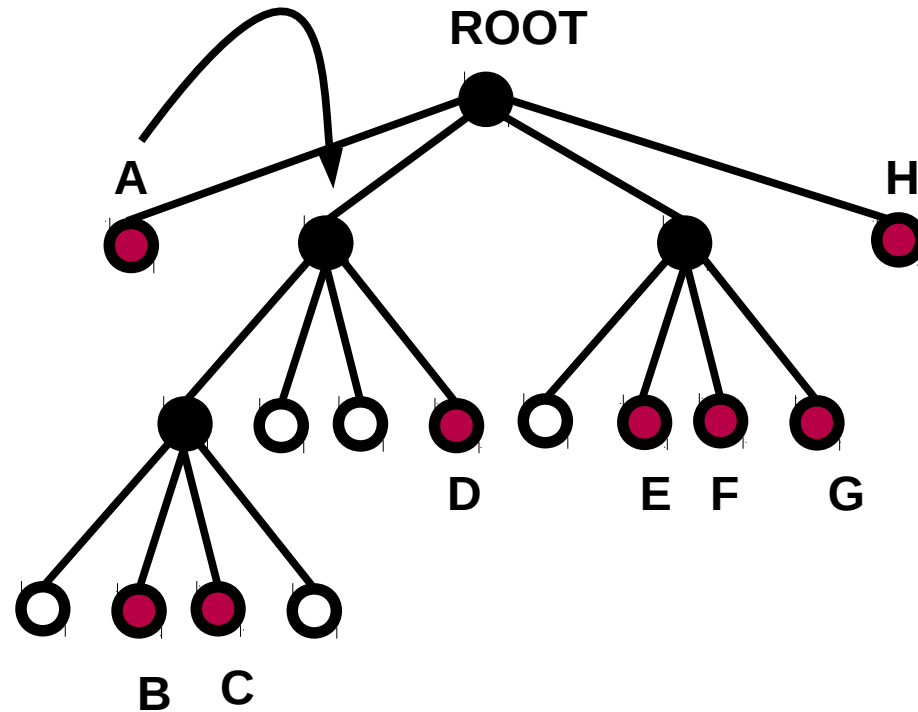s / d = 0.90 > θ = 0.75

I walk the tree to calculate forces (I split cells)

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If   s / d   <   θ
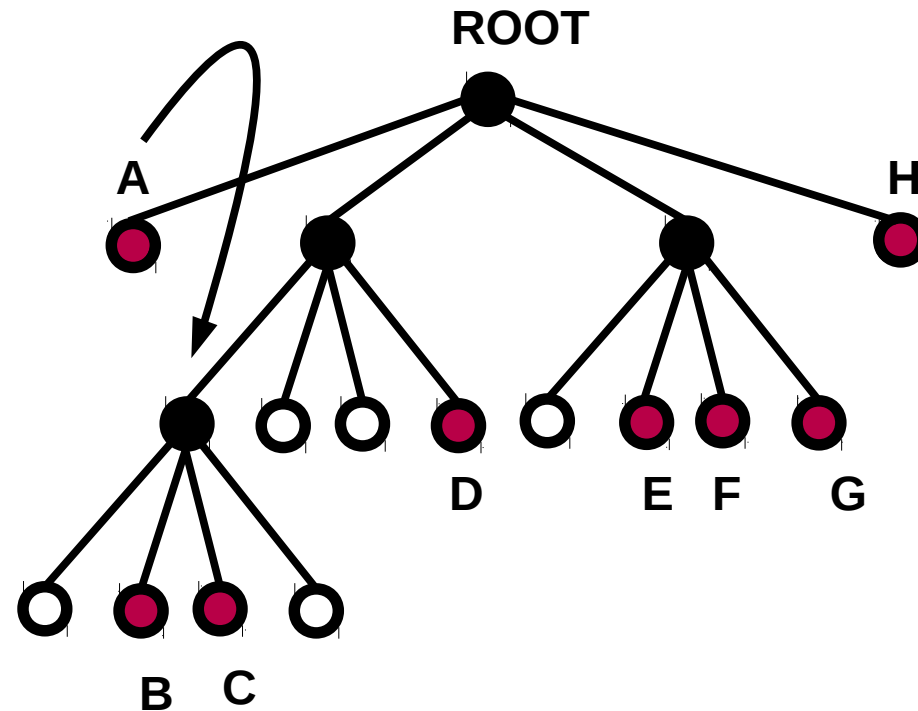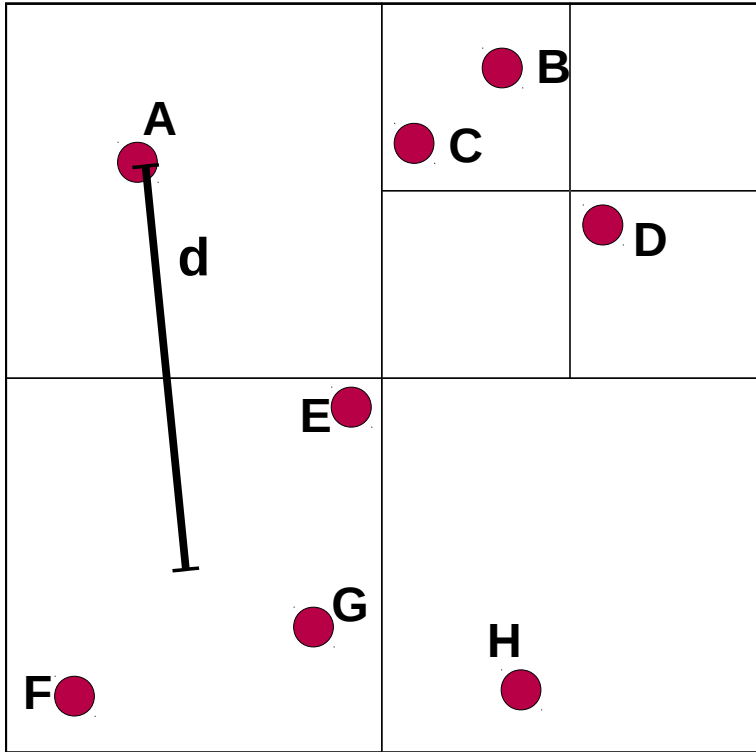
2. TREE CODE:

Example: CALCULATE FORCES ON A

s / d = 0.63 < θ = 0.75
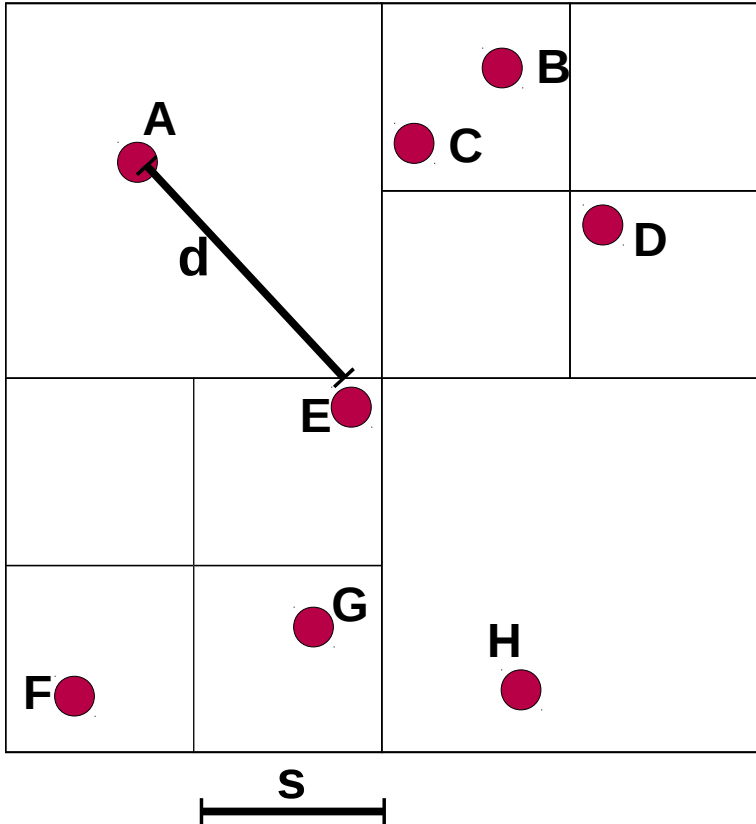
I should stop splitting cells, and besides E is a single particle: I calculate force between A and E

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If s / d < θ

2. TREE CODE:

Example: CALCULATE FORCES ON A

s / d = 0.25 < θ = 0.75

I should stop splitting cells, and besides F is a single particle: I calculate force between A and F

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?

If   s / d < θ

2. TREE CODE:

Example: CALCULATE FORCES ON A
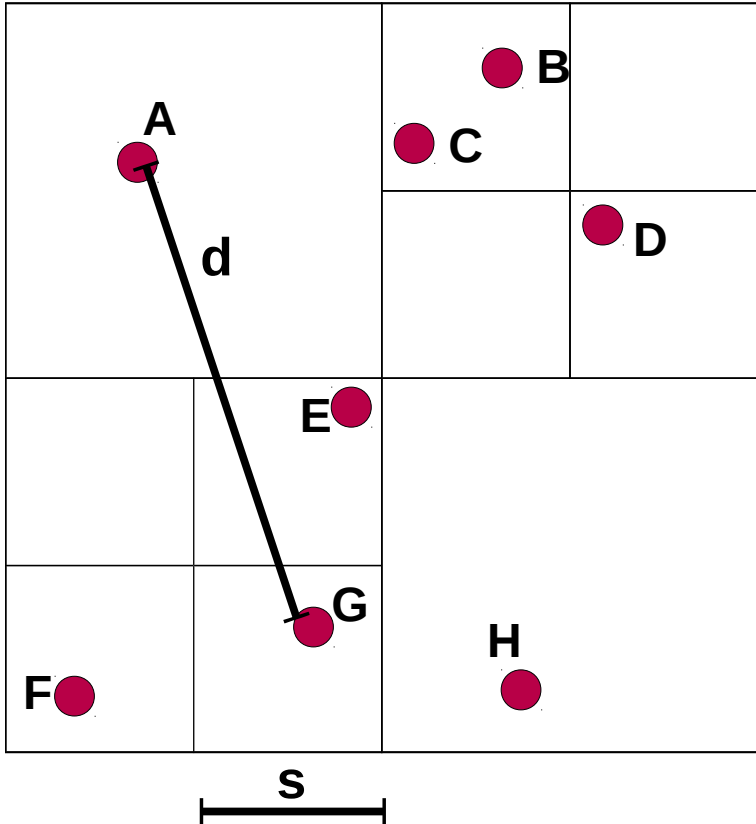
$s / d = 0.35 < \theta = 0.75$

I should stop splitting cells, and besides G is a single particle: I calculate force between A and G

HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?
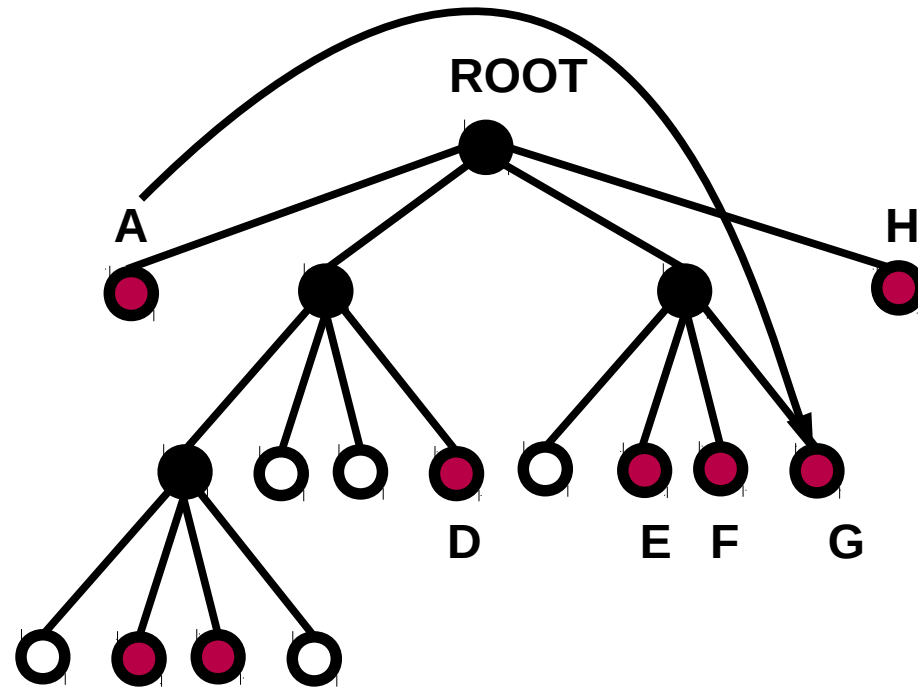
If $s / d < \theta$
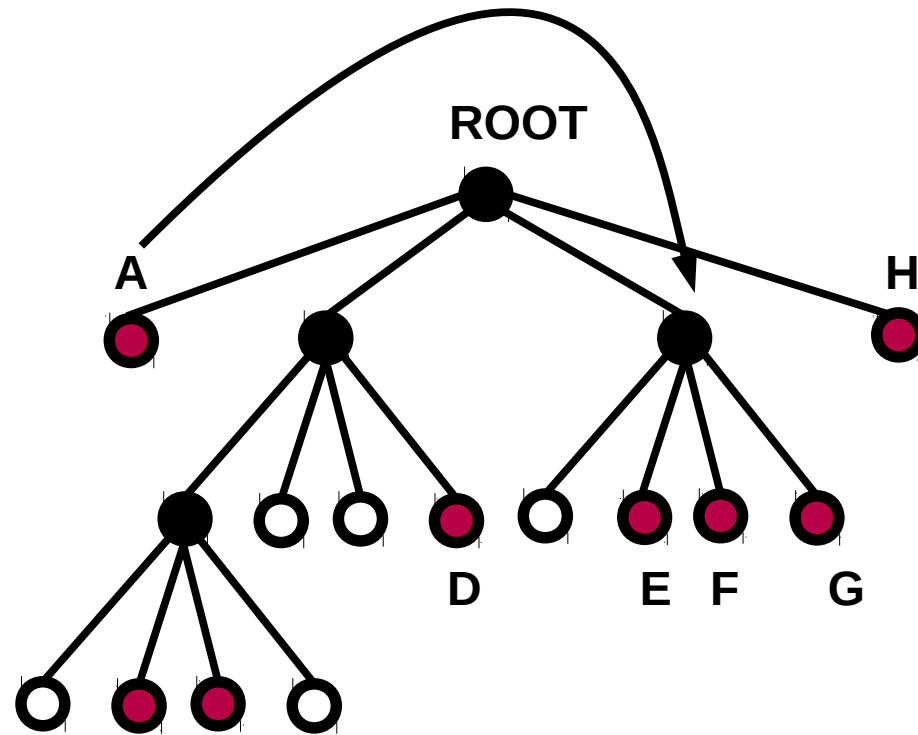
2. TREE CODE:

Example: CALCULATE FORCES ON A

s / d = 0.58 < θ = 0.75
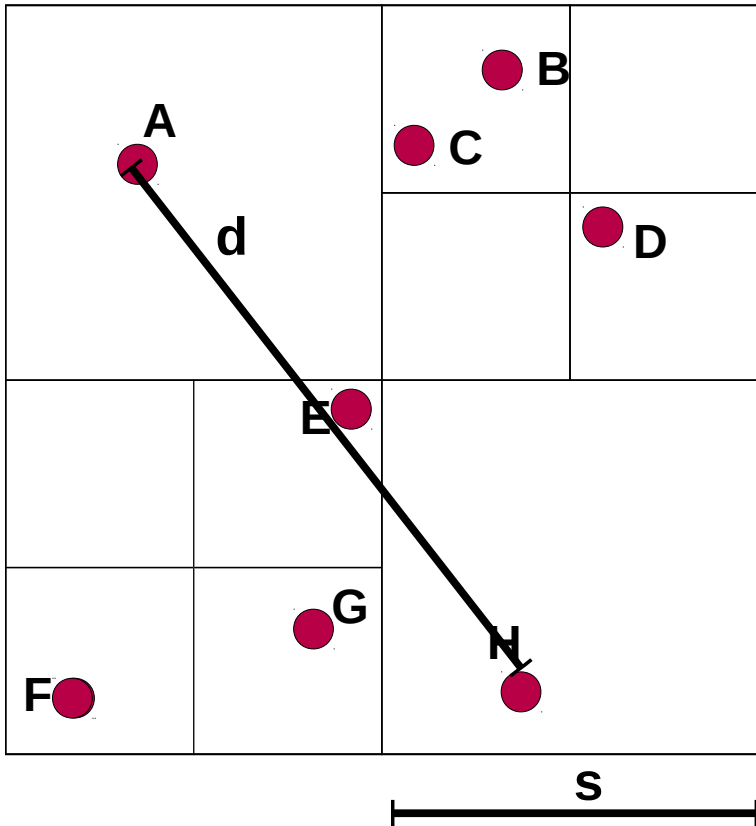
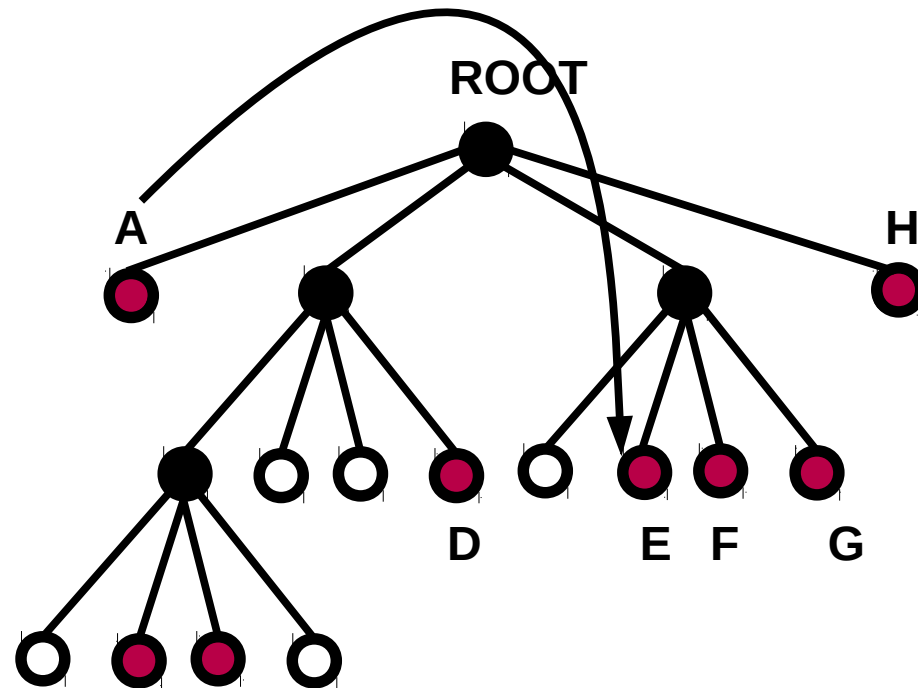I should stop splitting cells, and besides H is a single particle: I calculate force between A and H

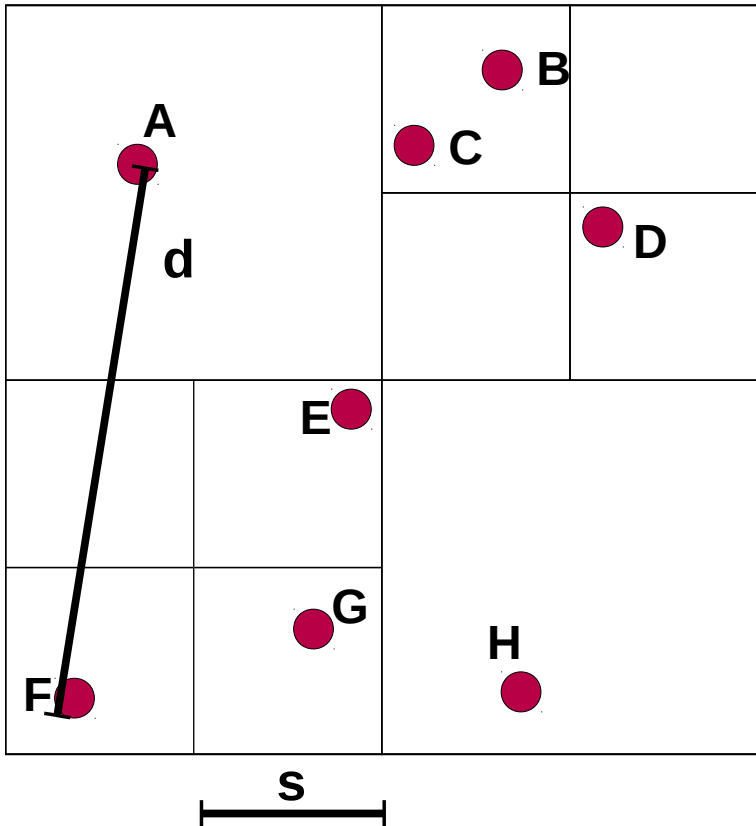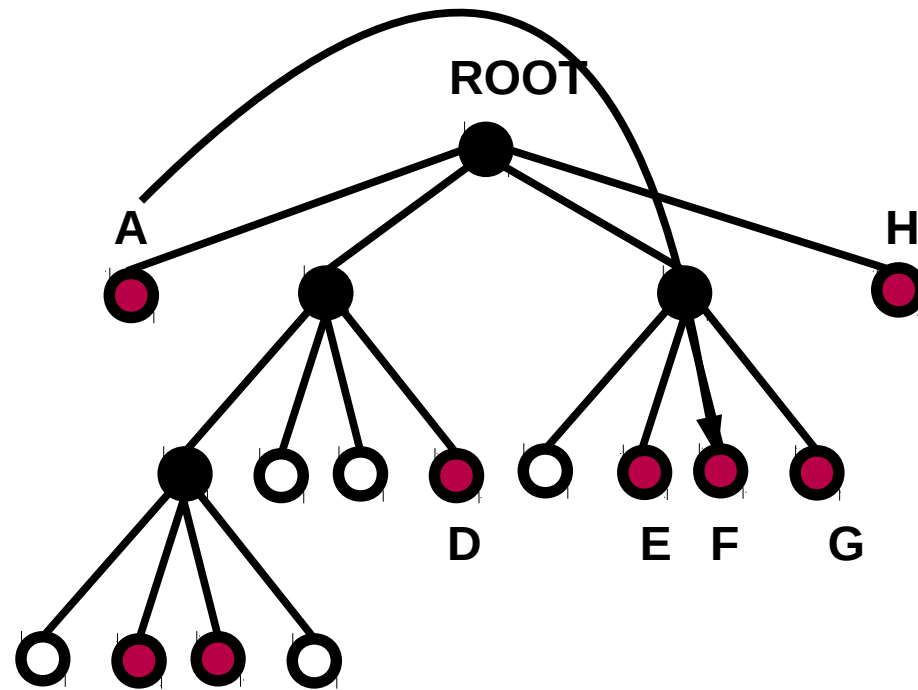<u>HOW DO I DECIDE THAT A PARTICLE IS FAR ENOUGH TO AVOID DIRECT CALCULATION?</u>

If   s / d   <   θ
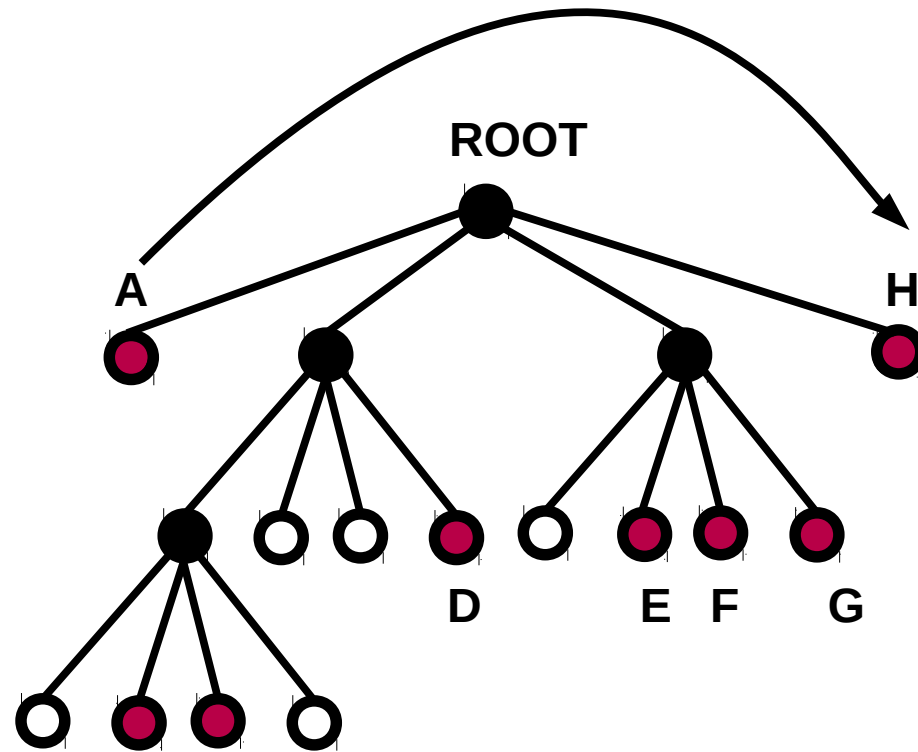
2. TREE CODE:

What is the advantage of building a tree?
Evaluation of force scales as O (N logN)
rather than O(N^2) as an effect of cell opening criterion

Which kind of integrator for force?
In most codes a LEAPFROG (see Lecture 1)

Which kind of time-stepping criterion?
Generally same time-step for all particles (since the system is collisionless) but multiple time-stepping is possible. Most used criterion

$$\Delta t = \min \left(\Delta t_{max}, \left(\frac{2\,\eta\,\epsilon}{|a|} \right)^{1/2} \right)$$

where η = accuracy parameter,   ε = softening,
A = acceleration of particle, Δt$_{max}$ = max. timestep

Tree codes can be pure Barnes-Hut trees, or combined with Particle-Mesh and Fast Multipole algorithms.
See next discussion

3. PARTICLE MESH (PM) algorithm

Not only time but also positions are discretized

$$\nabla^2 \Phi = 4\pi\rho$$

Positions of single particles are remapped to a MESH (GRID)

Potential (Poisson) equation is solved on mesh points
→ **problem scales as**
  G log G,
where G is the number of grid cells
Advantage: G < N

Then evaluate force exerted by this potential on each particle

To find the force on a particle not located at a meshpoint we can either use the force at the nearest meshpoint, or interpolate the force from the closest meshpoints

→ **Complexity O(N G log G) ~ O(N)**
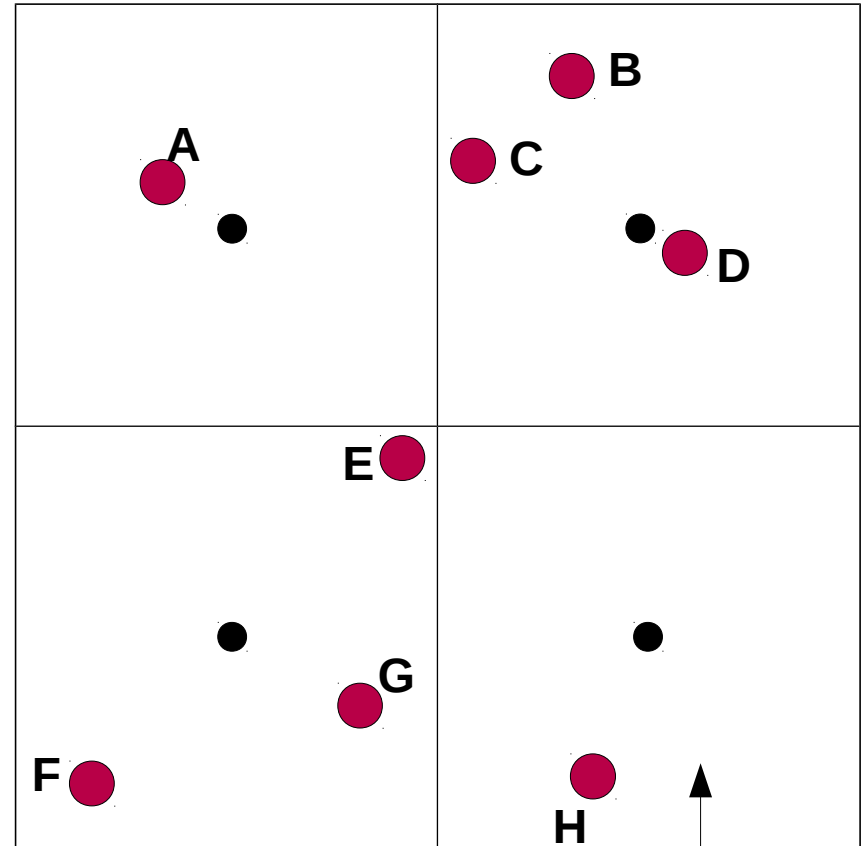
● **Meshpoints (G=4)**

● **Particles (N=8)**

Mesh cell

3. PARTICLE-PARTICLE (PP)/ PARTICLE MESH (PM) algorithm

PM algorithm is fast – O(N) - but inaccurate for nearby particles

PP PM is a HYBRID algorithm:

-PP on short range forces: within a critical distance particle-particle forces are calculated directly

-PM on long-range forces: outside critical distance particle-mesh algorithm is used

- critical distance~ 2 – 3 cell sizes
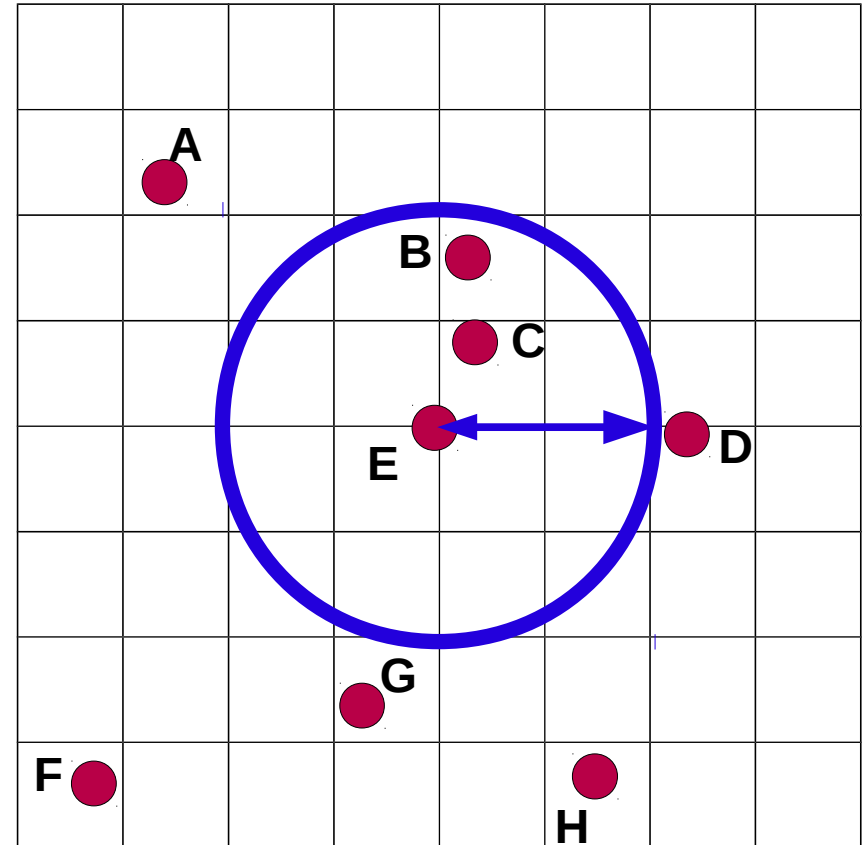
Critical distance for particle E
PP used for force of B and C on E
PM for all other forces on E

3. FAST MULTIPOLE MOMENT (FMM) algorithm

Similar to tree code but based on POTENTIAL INSTEAD OF FORCE

<u>Implications:</u>

1) Multipole expansions of the potential of a box is <u>more accurate</u> than center of mass substitution, but computationally <u>heavier,</u>

2) leaves can have > 1 particle (2 compensates 1),

3) force needs to be evaluated only at leaf level
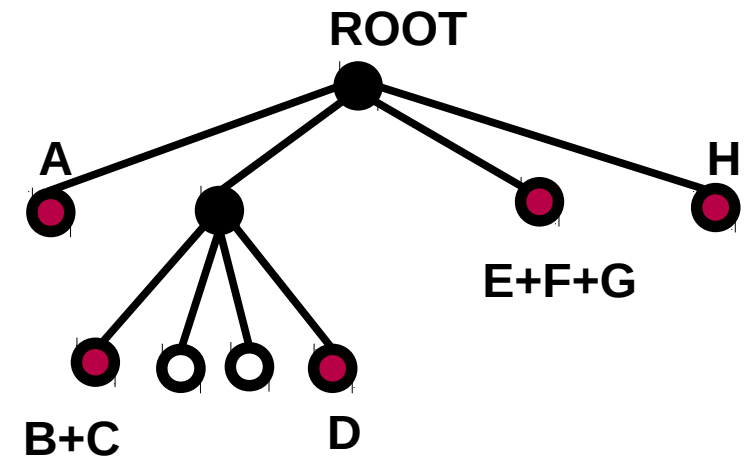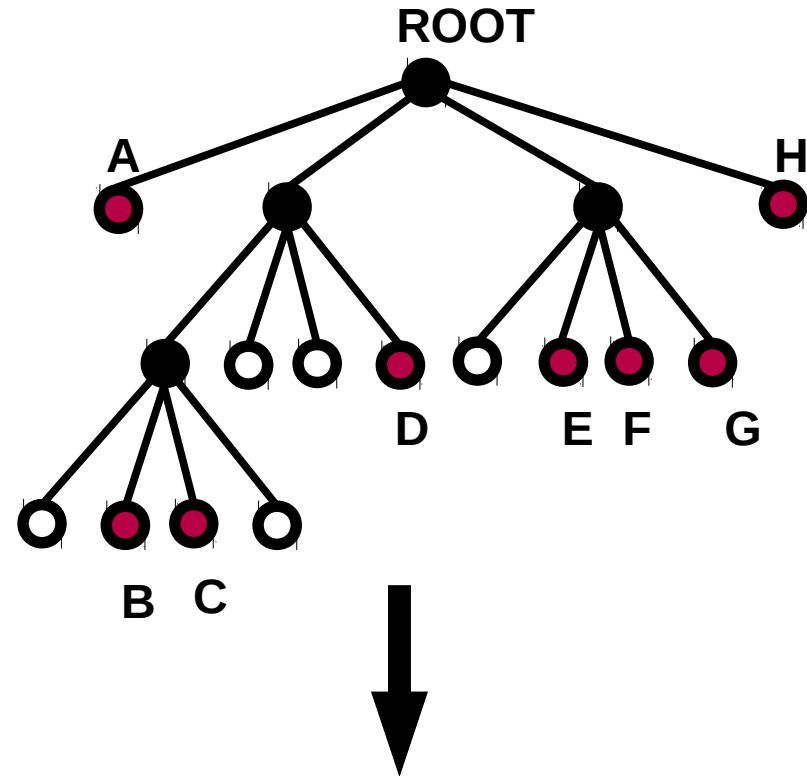
4. COMPUTER CLUSTERS: problems of N-body codes

- The need for parallel computing

- HPC architectures

- MPI: basic concepts and main issues

- Intrinsic problems of scaling

- The queues and the scheduler (example: PBS)

4. COMPUTER CLUSTERS: THE NEED FOR PARALLEL COMPUTING

Systems with >10^5 particles subject to gravity
(long-range force, we cannot neglect forces on distant particles)

→ **beyond clock and RAM performance of a single CPU**

→ **need to do same calculation in parallel**

BASIC CONCEPTS:
      - *N* particles must be distributed between *p* processors with a
           suitable algorithm (simplest one : *N/p*)
      - each processor calculates a portion of forces (those on local
           particles)
      - since gravity is long-range, information about calculated
           forces must be sent to all other processors

1. the more processors I have the better (unless *N~p*)

2. <u>communication</u> between processors is critical:
      must be fast, otherwise is <u>bottleneck</u>
           → **useless to increase *p* if communication saturates**

4. COMPUTER CLUSTERS: HPC architectures

Very fast internal network (eg infiniband)
- high transmission velocity (>>10Gbit/s)
- low latency (<<10 micro sec, latency = time delay
between stimulation and response)

Internet

node01   node02   node03   node04   node05   node06   node07   node08

(frontend and storage)

Many computing nodes (slaves):
`normal' CPUs but connected by fast network
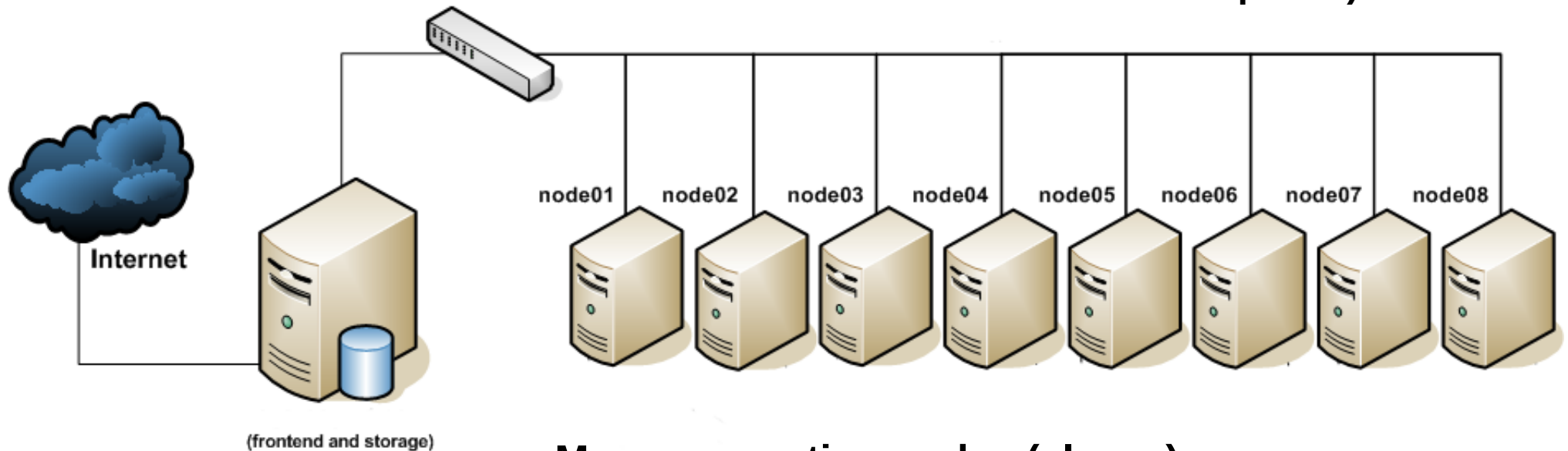
One front-end
(master) for
-user access (via
internet)
-compiling
-launching jobs

GPU cluster : GPUs connected to nodes (usually 2-4)

SOME EXAMPLES of CLUSTERS: GALILEO @ CINECA

System Architecture

```
Model: IBM NeXtScale
Architecture: Linux Infiniband Cluster
Nodes: 516
Processors: 8-cores Intel Haswell 2.40
GHz (2 per node)
Cores: 16 cores/node, 8256 cores in
total
Accelerators: 2 Intel Phi 7120p per node
on 384 nodes  (768 in total)
+ 2 NVIDIA K80 per node on 40 nodes
RAM: 128 GB/node, 8 GB/core
Internal Network: Infiniband with 4x QDR
switches
```

SOME EXAMPLES of CLUSTERS: FERMI @ CINECA

System Architecture

Model: IBM-BlueGene /Q
Architecture: 10 BGQ Frame
Front-end Nodes OS: Red-Hat EL 6.2
Compute Node Kernel: lightweight Linux-like kernel
Processor Type: IBM PowerA2, 1.6 GHz
Computing Nodes: 10240 with 16 cores each
Computing Cores: 163.840
RAM: 16GB / node; 1GB/core
Internal Network: Network interface with 11 links ->5D Torus

SOME EXAMPLES of CLUSTERS: FERMI @ CINECA

vs 8256

vs 2.4 GHz

System Architecture

```
Model: IBM-BlueGene /Q
Architecture: 10 BGQ Frame
Front-end Nodes OS: Red-Hat EL 6.2
Compute Node Kernel: lightweight Linux-
like kernel
Processor Type: IBM PowerA2, 1.6 GHz
Computing Nodes: 10240 with 16 cores
each
Computing Cores: 163.840
RAM: 16GB / node; 1GB/core
Internal Network: Network interface with
11 links ->5D Torus
```

vs 128 GB

4. COMPUTER CLUSTERS: MPI basic concepts

How does MPI work?

Library that organizes communication of commands, variables, information between nodes
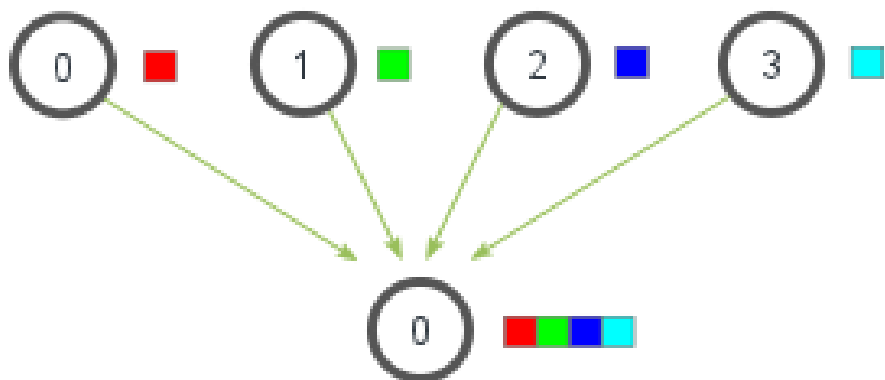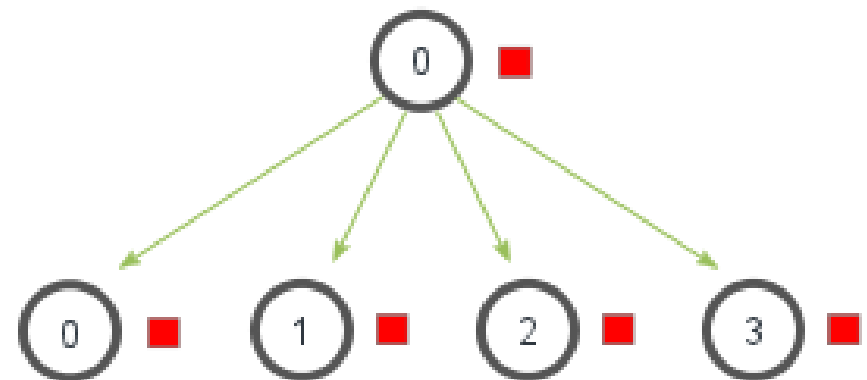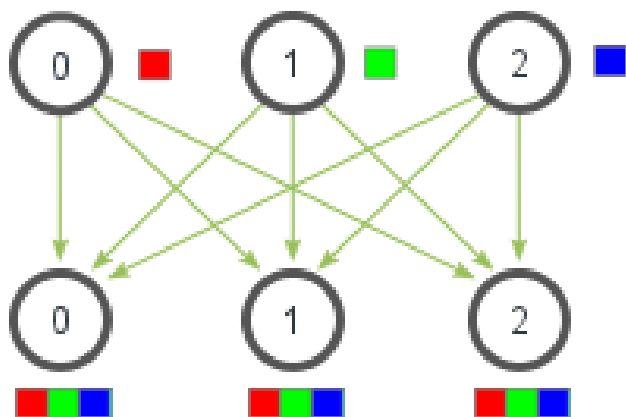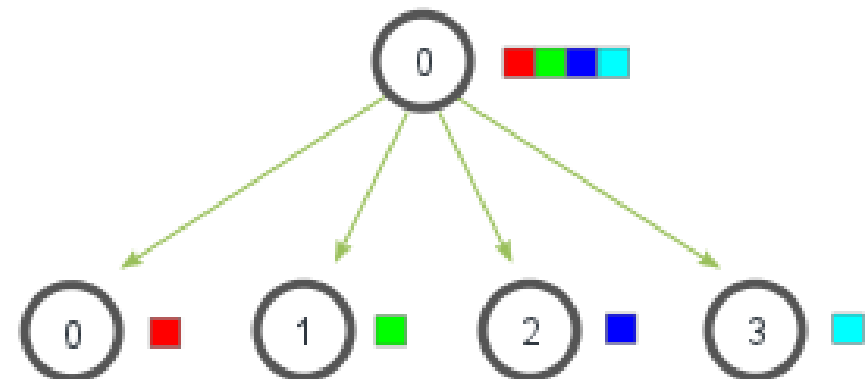
4. COMPUTER CLUSTERS: intrinsic problems of scaling

How do I measure the scaling?
Time for running a job versus number of cores
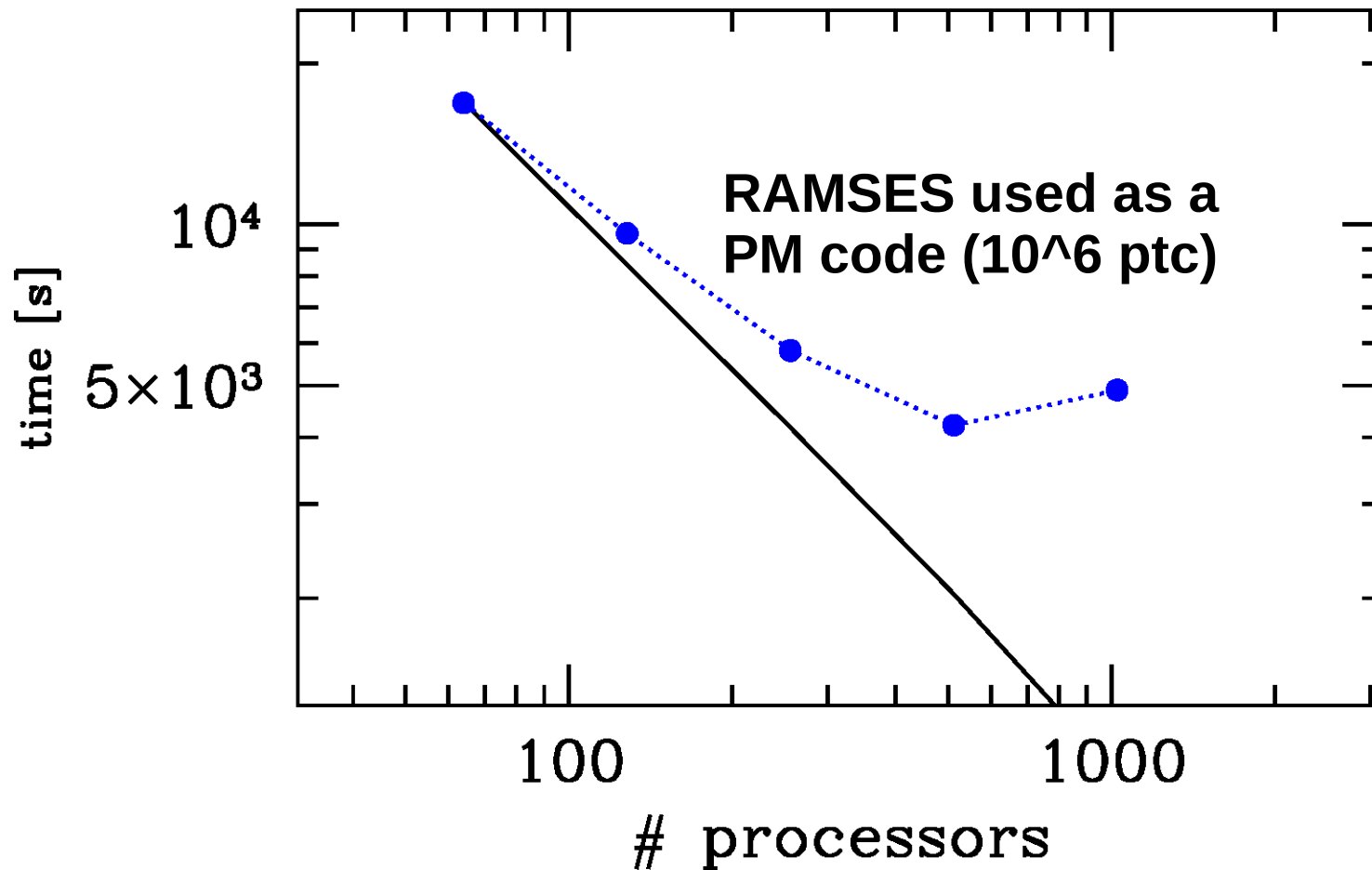
4. COMPUTER CLUSTERS: intrinsic problems of scaling

How do I measure the scaling?

SPEED UP: sequential execution time / parallel execution time
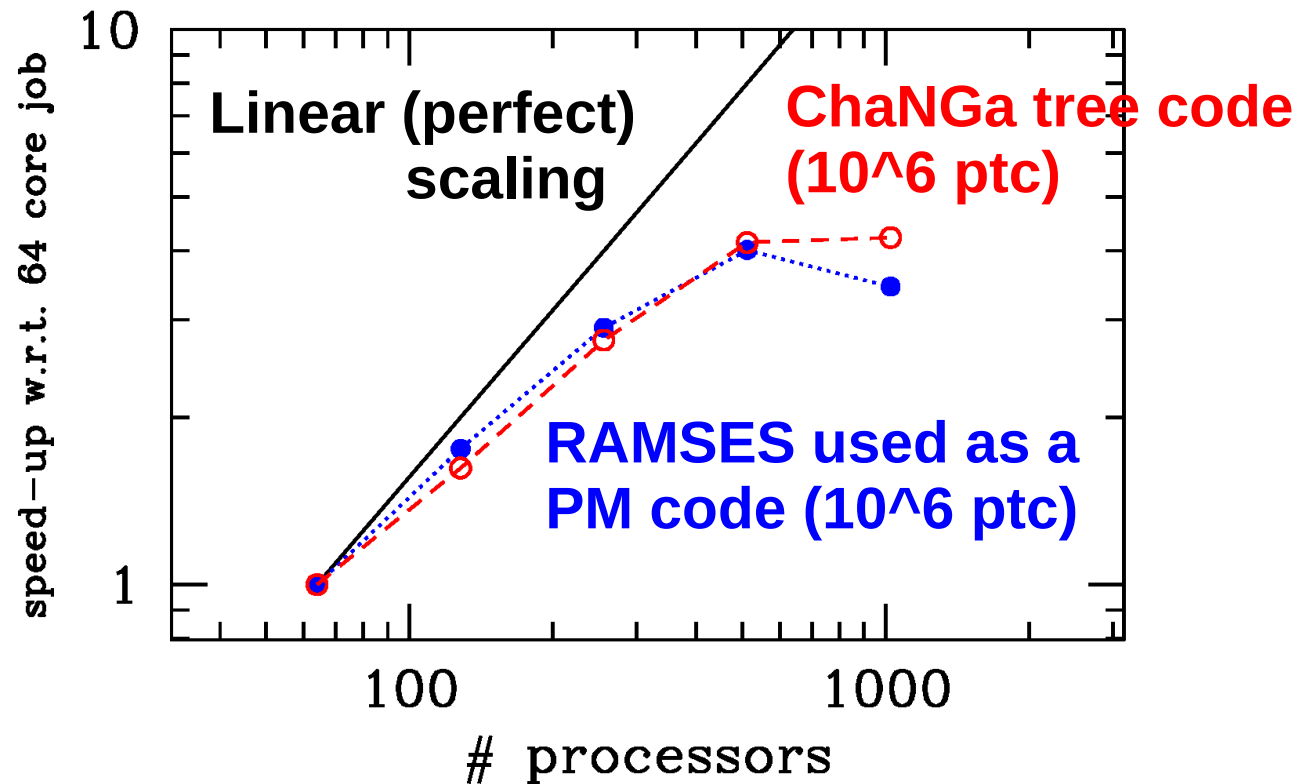
EFFICIENCY or LINEARITY: ratio of speed up to number of processors

The problems are
1. gravity is long-range → communication is necessary
2. gas physics, sub-grid physics, binaries or other things that require even more communication!

4. COMPUTER CLUSTERS: queues and scheduler

on a cluster you cannot run by typing
                mpirun -np 16 mycode
because there are other 100s of users

  1. You submit a job to a scheduler

  2. The scheduler assigns a priority to the job

  3. The job is QUEUED according to this priority

  4. The run starts when priority is higher than any other job

HOW TO SUBMIT A JOB?

Note: the University of Padova pays CPU time at CINECA
You might want to use such hours....

4. COMPUTER CLUSTERS: queues and scheduler

PBS (Portable Batch System) is the most used scheduler

1. prepare a submission script, e.g.

```
#!/bin/bash
#PBS -A NAME-OF-YOUR-PROJECT
#PBS -N changaprova2
#PBS -l walltime=00:59:00
#PBS -l select=1:ncpus=2


cd /gpfs/scratch/userexternal/mmapelli/sphere/dm_changa/
NPROCS=`wc -l < $PBS_NODEFILE`

./charmrun -machinefile $PBS_NODEFILE -np $NPROCS
./ChaNGa  ./newpar.par
```

This is the example pbs_script.sh

4. COMPUTER CLUSTERS: queues and scheduler

PBS (Portable Batch System) is the most used scheduler

2. submit job with

> **qsub pbs_script.sh**

3. check the status

> **qstat -u username**

4. delete wrong job

> **qdel job_identifier**

NOW we can go back to section 5. of Lecture 2 and understand it better..

5. ChANga

a smokable form of ayahuasca, containing dimetiltriptamina (DMT)

5. ChANga (Charm N-body GrAvity solver)
http://librarian.phys.washington.edu/astro/index.php/Research:ChaNGa
http://www-hpcc.astro.washington.edu/tools/changa.html

– Barnes-Hut tree code (where leaves can be > 1 particle and there is multipole expansion for far away particles)

– Softening not as Plummer but cubic spline

– In gravity only mode seems to work well for > 128k cores

– Multi-stepping for timesteps allowed (which does not mean it is smart to use it)

– Excellent work-load balance (= integrations and particles are well distributed between the used CPUs) thanks to
CHARM++ runtime system
http://charm.cs.uiuc.edu/software

– GAS treatment through smoothed-particle hydrodynamics (see Lecture 4)

– Lot of extra-physics (star formation, supernovae, … see Lecture 5)

5. ChANga

OBTAIN CHARM AND CHANGA:
git clone http://charm.cs.illinois.edu/gerrit/cosmo/changa.git
git clone http://charm.cs.illinois.edu/gerrit/cosmo/utility.git
git clone http://charm.cs.uiuc.edu/gerrit/charm

COMPILE CHARM AND CHANGA:
You need gnu & openmpi or intel and intelmpi
ONLY 64 bit architecture!!!!

cd charm/
./build ChaNGa mpi-linux-x86_64   mpicxx

export
CHARM_DIR="/gpfs/scratch/userexternal/mmapelli/charm/"

cd changa/
./configure --enable-cooling=cosmo

make

RUN CHARM AND CHANGA:
./charmrun -np 2 ./ChaNGa ./newpar.par

5. ChANga

EXAMPLE of a PARAMETER FILE:

```
bPeriodic       = 0    # periodic boundaries: use only for cosmo
bParaRead       = 0    # read input file in parallel (usually safe)
bParaWrite      = 0    # write outputs in parallel

dTheta          = 0.55  # opening angle for gravity (standard value)
nBucket         = 16    # number of particles in a "bucket", i.e. leaf
of the gravity tree

iStartStep      = 0    # starting step number of simulation (for
output numbering)
achInFile       = MW_gas_intrud.std #input file
achOutName      = out  #output file
dDelta  = 0.01          # length of base timestep ("Rung 0") in Gyr

#dHubble0        = 2.894405 # hubble constant
#dOmega0         = 0.268        # Omega_0
#dLambda = 0.732
dMsolUnit       = 2.225e5    # mass unit in solar masses
dKpcUnit        = 1.0      # box size in kpc (NOT kpc/h!)
#dRedTo          = 0.0          # final redshift of simulation
```

5. ChANga

EXAMPLE of a PARAMETER FILE:

```
iCheckInterval  = 1        # do a checkpoint every N timesteps
dExtraStore     = 3        # extra memory storage per node

bKDK            = 1        # kick-drift-kick, leave like this
iMaxRung        = 15       # max number of "rungs", i.e. sub-timestep


dEtaCourant     = 0.3         # accuracy of SPH

bStandard       = 1        # snapshots in standard binary
bDoGravity      = 1        # ok
#bComove         = 1          # use comoving coordinates for cosmo
```

5. ChANga

EXAMPLE of a PARAMETER FILE:

iGasModel   = 2   # gas thermodynamics
0 = adiabatic (CHECK!)
1 = isothermal (CHECK!)
2 = cooling

dConstGamma   = 1.666667   # gamma of equation of state
dMeanMolWeight = 0.59259   # for equation of state

nSteps   = 5000   # total number of timesteps
iOutInterval   = 1   # do tipsy snapshot every N dDeltas
iLogInterval   = 1   # do logfile output every N dDeltas
bOverwrite   = 1   # do you want to overwrite outputs?
bVDetails   = 1   # verbose details in logfile (?)
bDoDensity   = 1   # do you want an external tipsy array
                                with dens
#iBinaryOutput   = 1   # write the external tipsy arrays in
                                binary

nSmooth   = 32   # number of neighbors for SPH
bViscosityLimiter = 1   # do you want the Balsara switch?

5. ChANga

EXAMPLE of a PARAMETER FILE:

#The parameters below should improve computational speed
bFastGas   = 1

#dFracNoDomainDecomp = 0.2   # below this fraction, do not
                                             decompose domain

nTruncateRung   = 32   # do not use small timestep rungs with
                                             less

#dhMinOverSoft limits the SPH smoothing length
#dhMinOverSoft = 0.05

#options for physical softening
bPhysicalSoft   = 1   # use physical (not comoving) softening
#bSoftMaxMul   = 1   # factor used in scaling comoving softening
#dSoftMax   = 10   # (1+z) after which use physical softening

5. ChANga

EXAMPLE of a PARAMETER FILE:

#star formation parameters

#if you want to activate the UV just put bUV=1 and then will read the UV
#file - but obly works with cooling on!

```
bDoIOrderOutput   = 1        # produce .iord and .igasorder files
bUV               = 0        # read UV background file
bStarForm         = 1        # do you want star formation?
#dOverDenMin      = 2.63         # blocks star formation in unvirialized
dPhysDenMin  = 5.0           # density threshold for SF in atoms/cc
dTempMax          = 3e4          # max temperature for SF in K
dCStar        = 0.05   # SF efficiency
bFeedBack       = 1        # do you want FB?
dESN          = 0.8e51     # SN
dMinGasMass  = 0.1     # min mass of gas particle before it gets deleted 20%
dInitStarMass   = 0.1   # min mass of new star particles
bLowTCool       = 1        # do you want metal cooling below 8000 K?
bSelfShield       = 0        # do you want self-shielding of dense gastoUV
```

5. ChANga

EXAMPLE of RUN:

**RUN A BLOB OF DARK MATTER WITH UNIFORM DENSITY
(velocity field is Gaussian, with a power spectrum)
10^5 DM particles for 5 Myr**

./charmrun -np $NPROCS ./ChaNGa ./newpar_dm.par

visualize with tipsy

5. ChANga

Analysis tool:

TIPSY
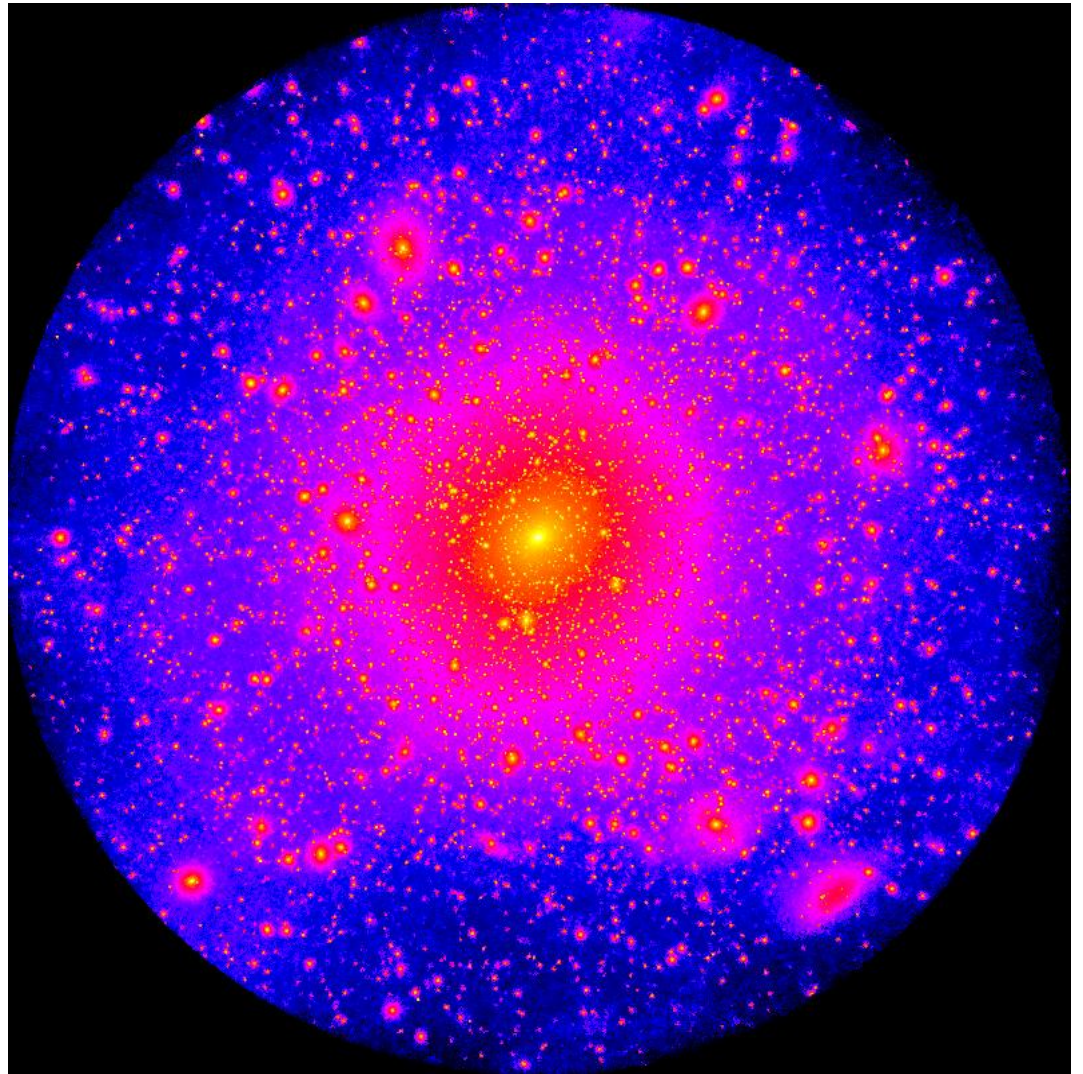http://www-hpcc.astro.washington.edu/tools/tipsy/tipsy.html

Installation:
follow readme on the
website

Usage:
Type tipsy on the terminal

openb namefile
loads 1
xall
boxstat 0 all

and so on..

5. ChANga

EXAMPLE of RUN:

RUN A BLOB OF DARK MATTER WITH UNIFORM DENSITY
(velocity field is Gaussian, with a power spectrum)
10^5 DM particles for 5 Myr

./charmrun -np $NPROCS ./ChaNGa ./newpar_dm.par

visualize with tipsy

```
openb out.000040
loads 1
zall                             # xall or yall depending on projection
boxstat 0 all
gasify 1 1 10 1.001 0 0 0        # to use  plot options of gas for DM
viewgas logrho 2 10              # to see projected density in code units,
                                    log scale
hard movie out40                 # create an image in tipsy image format
shell convert out40 out40.jpg  # convert image to jpg
```

5. ChANga

EXAMPLE of RUN:

**RUN A BLOB OF DARK MATTER WITH UNIFORM DENSITY
(velocity field is Gaussian, with a power spectrum)
10^5 DM particles for 5 Myr**

./charmrun -np $NPROCS ./ChaNGa ./newpar_dm.par

visualize with tipsy

```
openb out.000040
loads 1
zall                          #
boxstat 0 all
gasify 1 1 10 1.001 0 0 0      #
viewgas logrho 2 10            #

hard movie out40              #
shell convert out40 out40.jpg  #
```


0 Myr

5. ChANga

EXAMPLE of RUN:

RUN A BLOB OF DARK MATTER WITH UNIFORM DENSITY
(velocity field is Gaussian, with a power spectrum)
10^5 DM particles for 5 Myr

./charmrun -np $NPROCS ./ChaNGa ./newpar_dm.par

visualize with tipsy

openb out.000040
loads 1
zall   #
boxstat 0 all
gasify 1 1 10 1.001 0 0 0   #
viewgas logrho 2 10   #

hard movie out40   #
shell convert out40 out40.jpg #

4 Myr