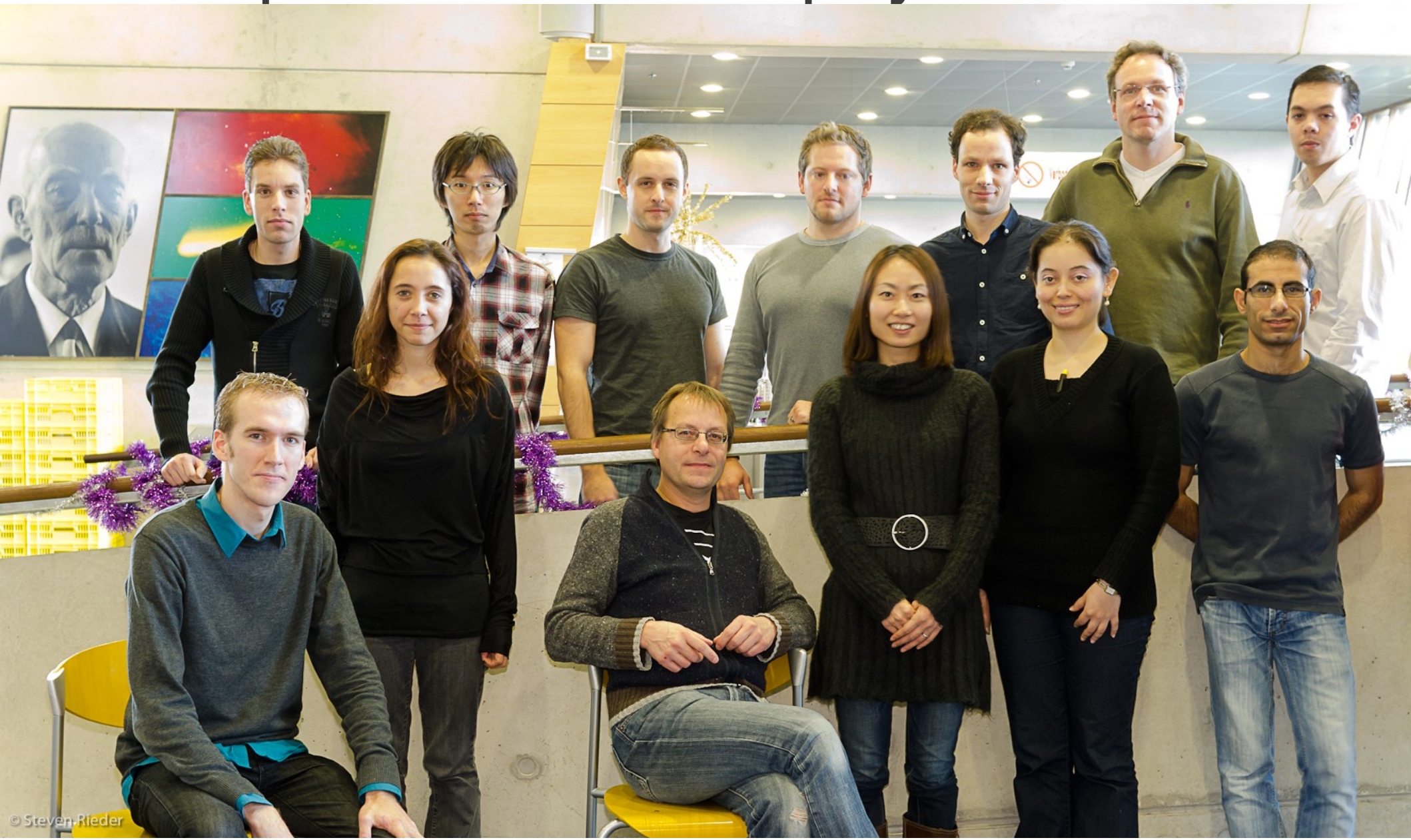
AMUSE-ing CosmoGrid

Simon Portegies Zwart
Sterrewacht Leiden

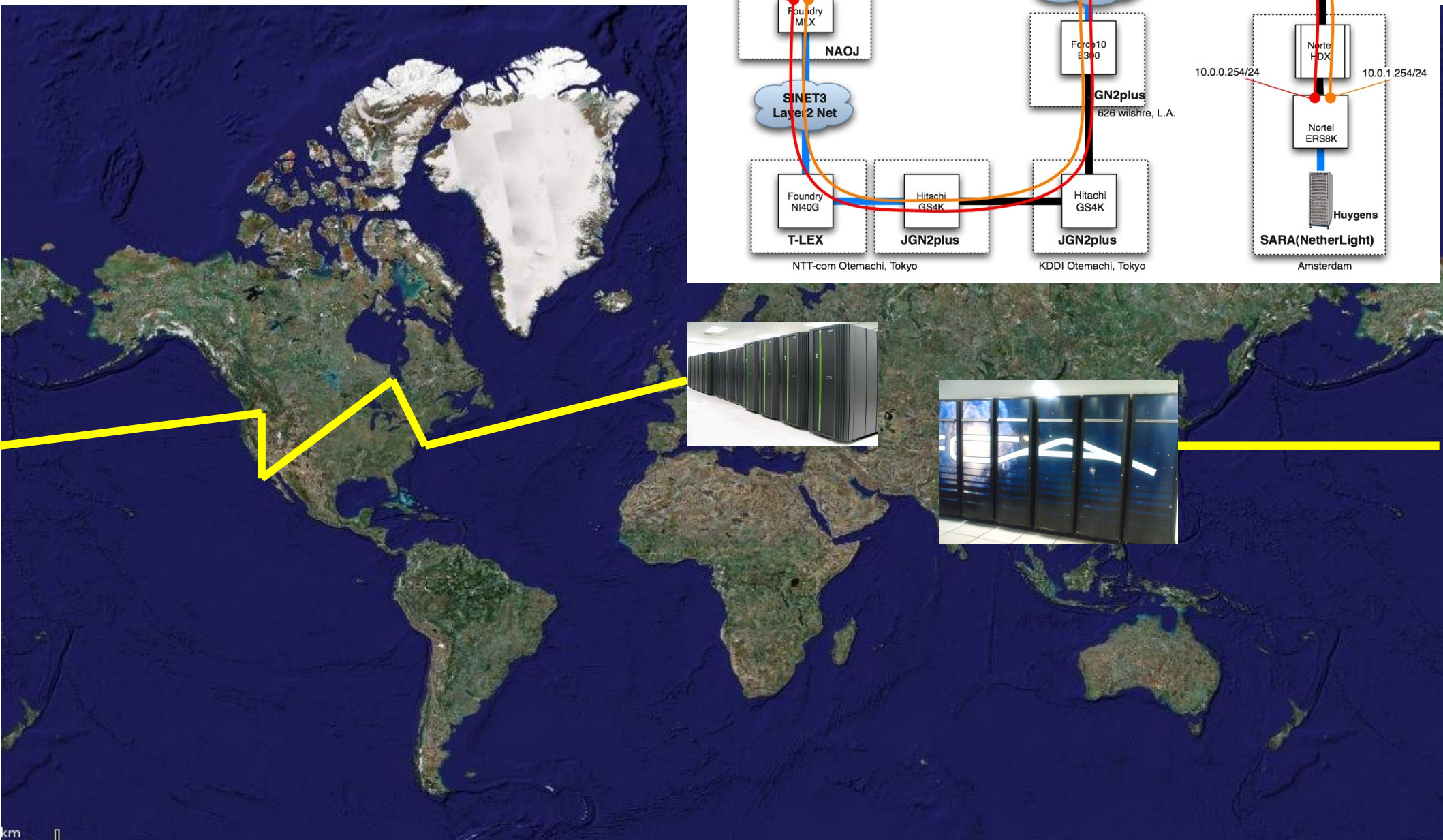
Sterrewacht Leiden

Sterrewacht Leiden

Computational Astrophysics Leiden



A MODEST Approach to MUSE

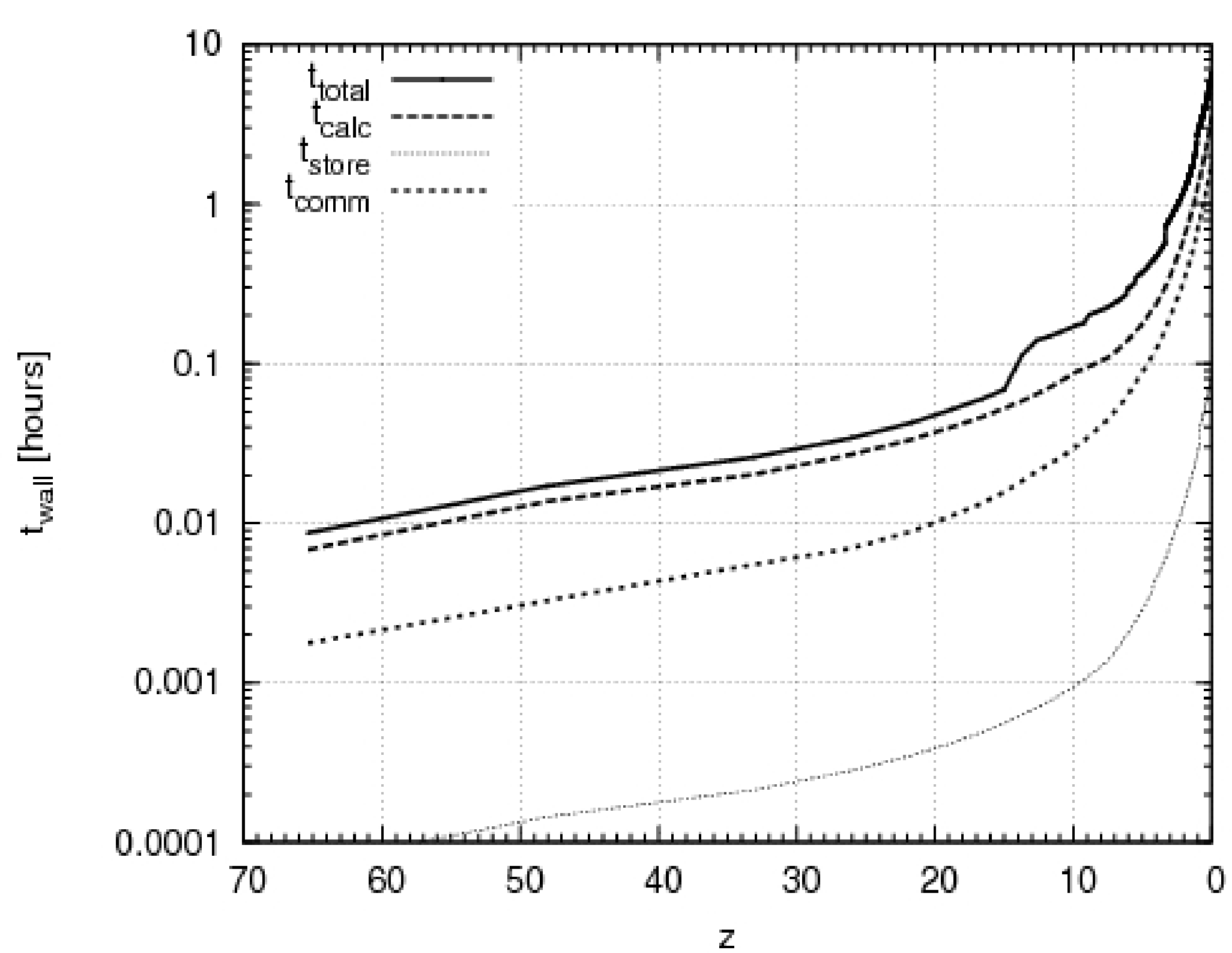

Network Topology for Cosmo Grid experiment
- Round Trip Performance Test -
Rev0 · 4 May. 21 2008
tanaka@kddnet.ad.jp
VLAN 3780: 10.0.0.0/24
VLAN 3781: 10.0.1.0/24
10G STM-64/10GE WAN-PHY
10GE LAN-PHY
Mitaka,Tokyo
Cray TX4
10.0.0.2/24
10.0.1.2/24
Foundry RX4
10.0.0.1/24
10.0.1.1/24
Foundry MLX
NAOJ
SINET3
Layer2 Net
Foundry NI40G
T-LEX
Hitachi GS4K
JGN2plus
NTT-com Otemachi, Tokyo
NorthWestern Univ.
710 Lakeshore Chicago
JGN2plus
Hitachi GS4K
StarLIGHT
Force10 E1200
Nortel HDX
MLR
Layer1 WaveNet
IRNC circuit
Force10 E300
GN2plus
626 wilshre, L.A.
Hitachi GS4K
JGN2plus
KDDI Otemachi, Tokyo
Nortel HDX
10.0.0.254/24
10.0.1.254/24
Nortel ERS8K
Huygens
SARA(NetherLight)
Amsterdam

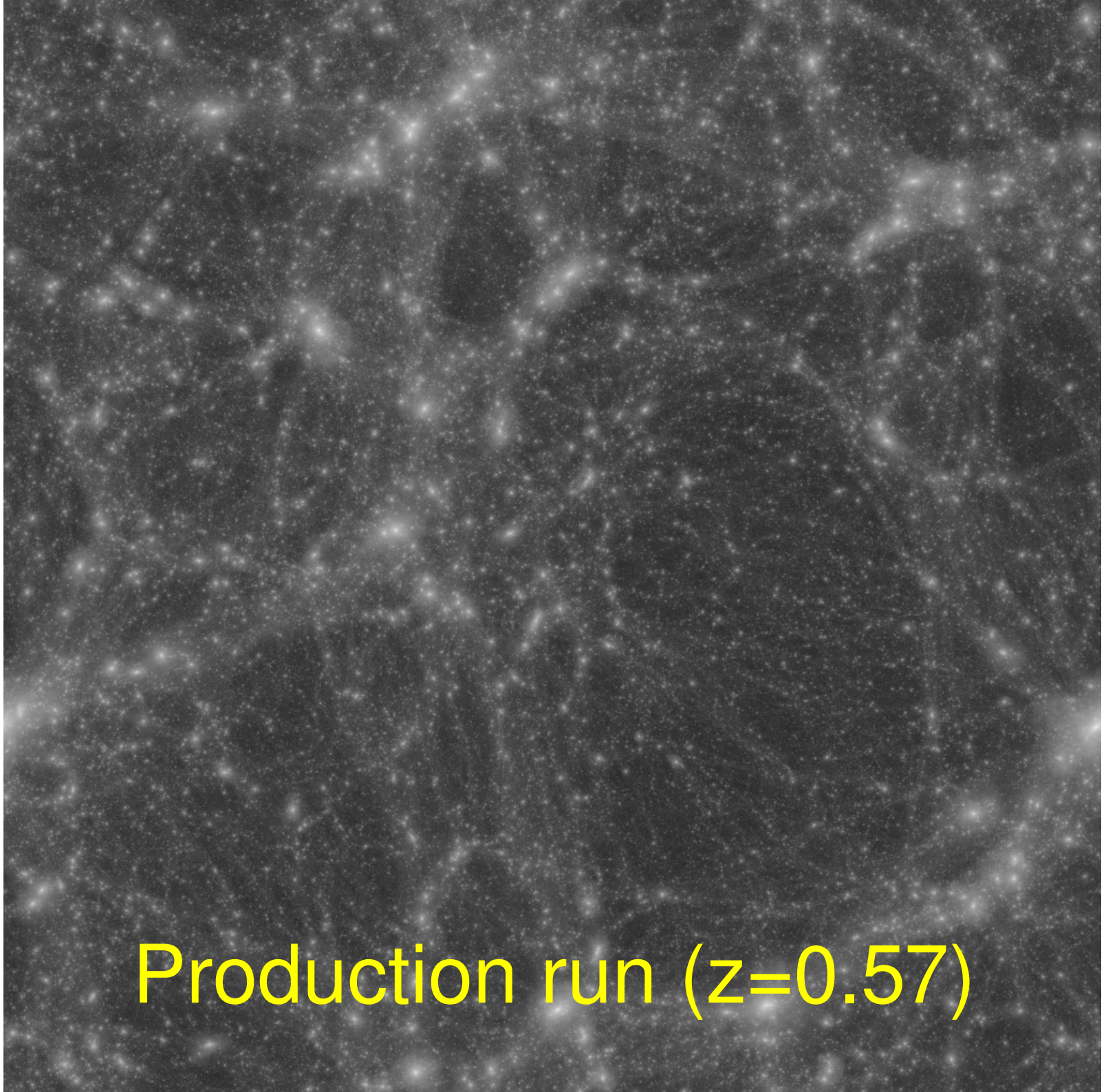
- Cray-XT 4 in Tokyo, Japan (26 TFLOPs).

- IBM Power6 in Amsterdam, NL (60 TFLOPs).

Cray-XT 4/5 in Espoo, Finland (76.5 TFLOPs).

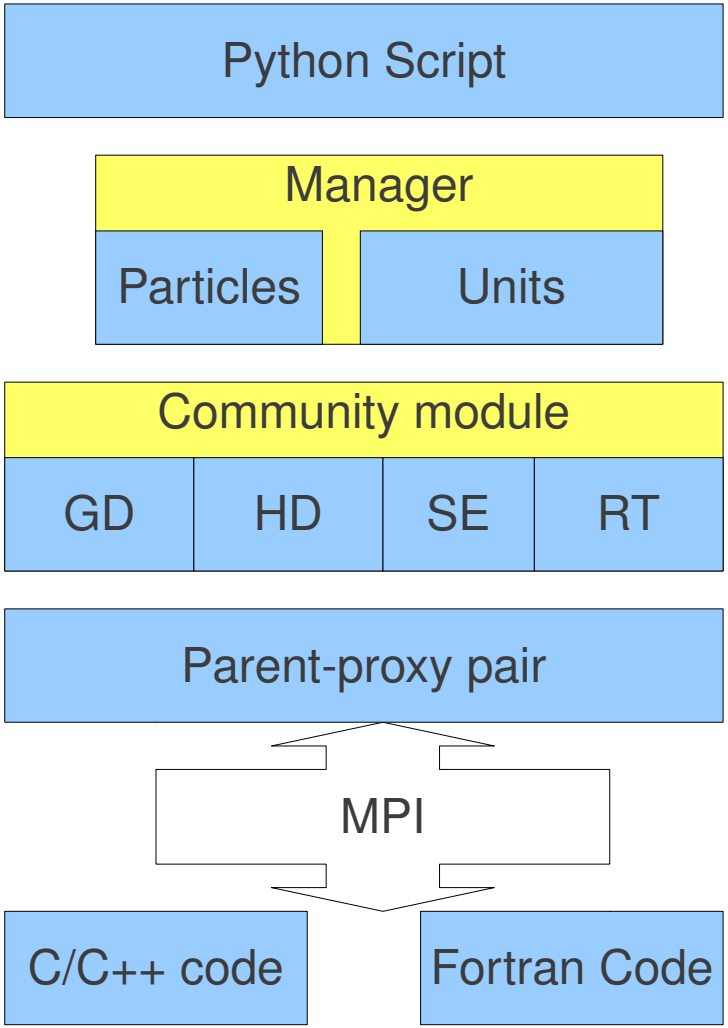
Cray-XT 4/5 in Edinburgh, UK (174 TFLOPs).


10
1
0.1
0.01
0.001
0.0001
t_{wall} [hours]
t_{total}
t_{calc}
t_{store}
t_{comm}
70
60
50
40
30
20
10
0
z

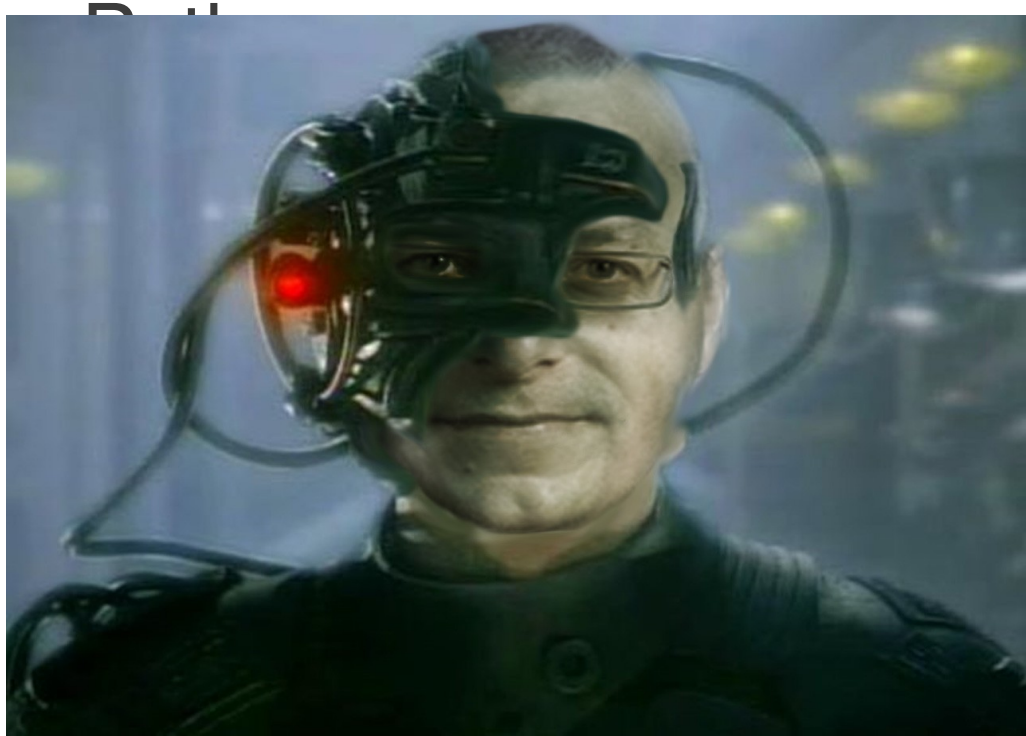

Production run (z=0.57)

AMUSE

http://amusecode.org

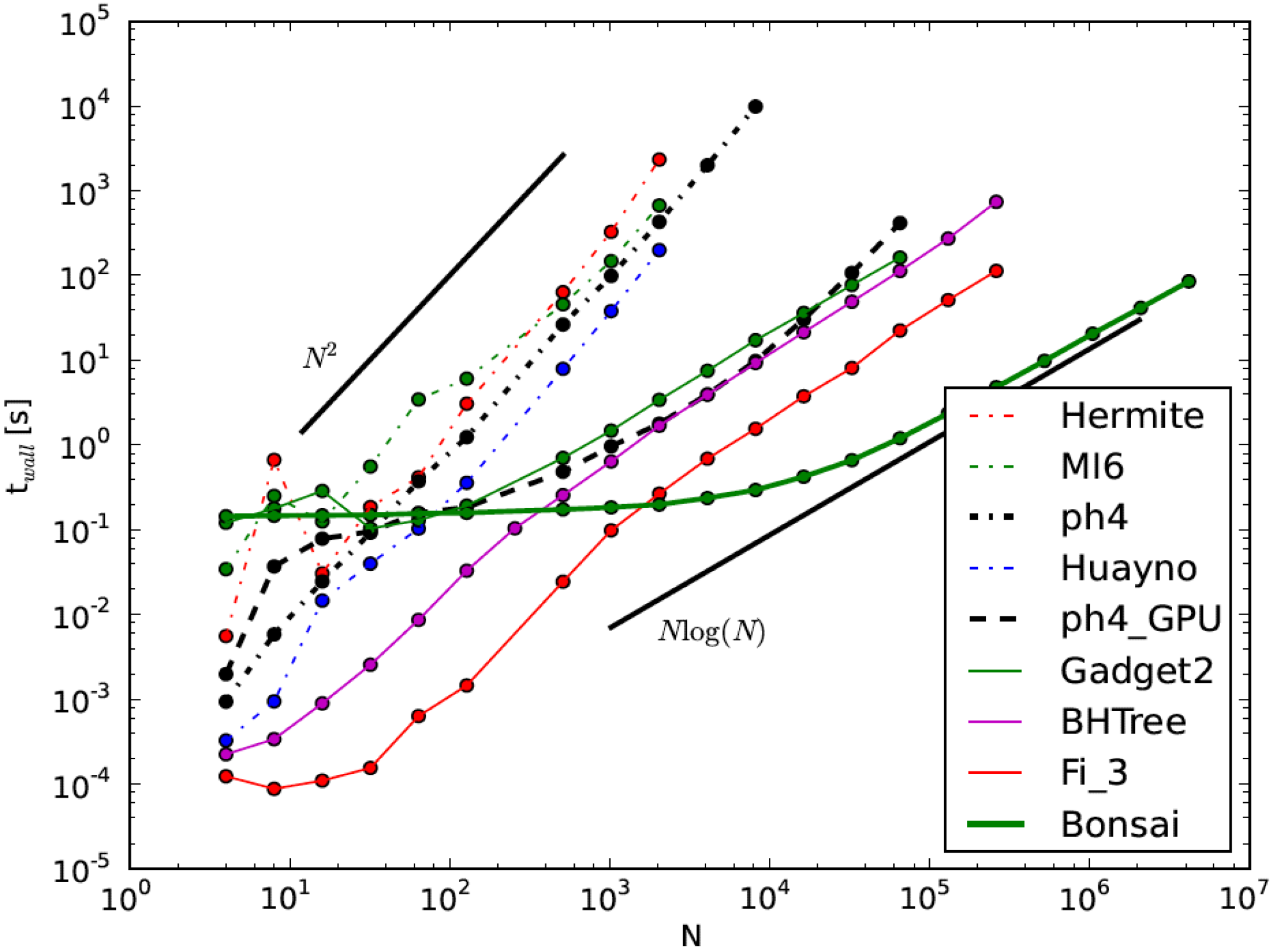


- Layers having different roles
- Written in C/C++, Java Fortran and



AMUSE - philosophy

- Build on community codes
- Standardized interfaces
- Automate as much as possible
- Core Team:
  - Inti Pelupessy (researcher)
  - Arjen van Elteren (software engineer)
  - Nathan de Vries (scientific programmer)
  - David Jansen (user support)

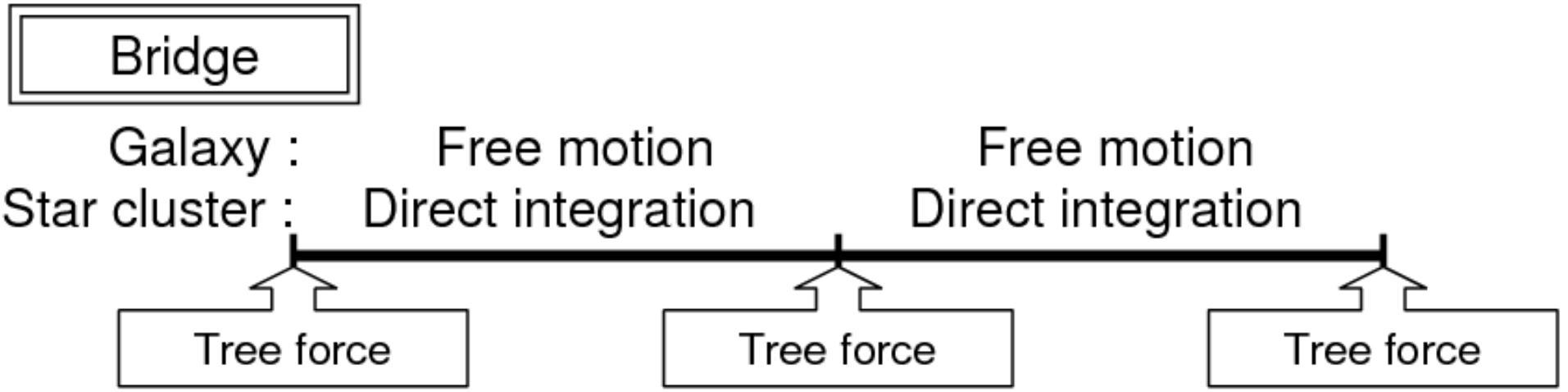

10^5
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
t_{wall} [s]
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
N
N^2
$N\log(N)$
Hermite
MI6
ph4
Huayno
ph4_GPU
Gadget2
BHTree
Fi_3
Bonsai

```
def main(CG_File="cg.hdf5", Ncl=65536, Rvir=1|parsec, W0=7, t_end=10|Gyr):

    masses = new_salpeter_mass_function(Ncl)
    converter=nbody_system.nbody_to_si(masses.sum(),Rvir)
    cluster_particles = new_king_model(len(masses), W0, converter)
    cluster_particles.scale_to_standard(convert_nbody=converter)
    cluster = Bonsai(converter)
    cluster.particles.add_particles(cluster_particles)
    to_framework = cluster.particles.new_channel_to(cluster_particles)

    cosmo_grid = read_set_from_file(CG_File)
    cluster.position = cosmo_grid.globular_cluster[0].pos
    cluster.velocity = cosmo_grid.globular_cluster[0].vel

    universe = bridge.Bridge(use_threading=False)
    universe.add_system(cluster, (cosmogrid,) )

    print_diagnostics(cluster, cosmogrid)
    universe.timestep = 1|Myr
    while universe.model_time < 10|Gyr:
        universe.evolve_model(universe.model_time + 100|Myr)
        to_framework.copy()

        print_diagnostics(cluster, cosmogrid)
        write_set_to_file(universe, "cluster_snapshots.hdf5", 'hdf5')
    cluster.stop()
```

```
def main(CG_File="cg.hdf5", Ncl=65536, Rvir=1|parsec, W0=7, t_end=10|Gyr):
    masses = new_salpeter_mass_function(Ncl)
    converter=nbody_system.nbody_to_si(masses.sum(),Rvir)
    cluster_particles = new_king_model(len(masses), W0, converter)
    cluster_particles.scale_to_standard(convert_nbody=converter)
    cluster = Bonsai(converter)
    cluster.particles.add_particles(cluster_particles)
    to_framework = cluster.particles.new_channel_to(cluster_particles)

    cosmo_grid = read_set_from_file(CG_File)
    cluster.position = cosmo_grid.globular_cluster[0].pos
    cluster.velocity = cosmo_grid.globular_cluster[0].vel

    universe = bridge.Bridge(use_threading=False)
    universe.add_system(cluster, (cosmogrid,) )

    print_diagnostics(cluster, cosmogrid)
    universe.timestep = 1|Myr
    while universe.model_time < 10|Gyr:
        universe.evolve_model(universe.model_time + 100|Myr)
        to_framework.copy()

        print_diagnostics(cluster, cosmogrid)
        write_set_to_file(universe, "cluster_snapshots.hdf5", 'hdf5')
    cluster.stop()
```

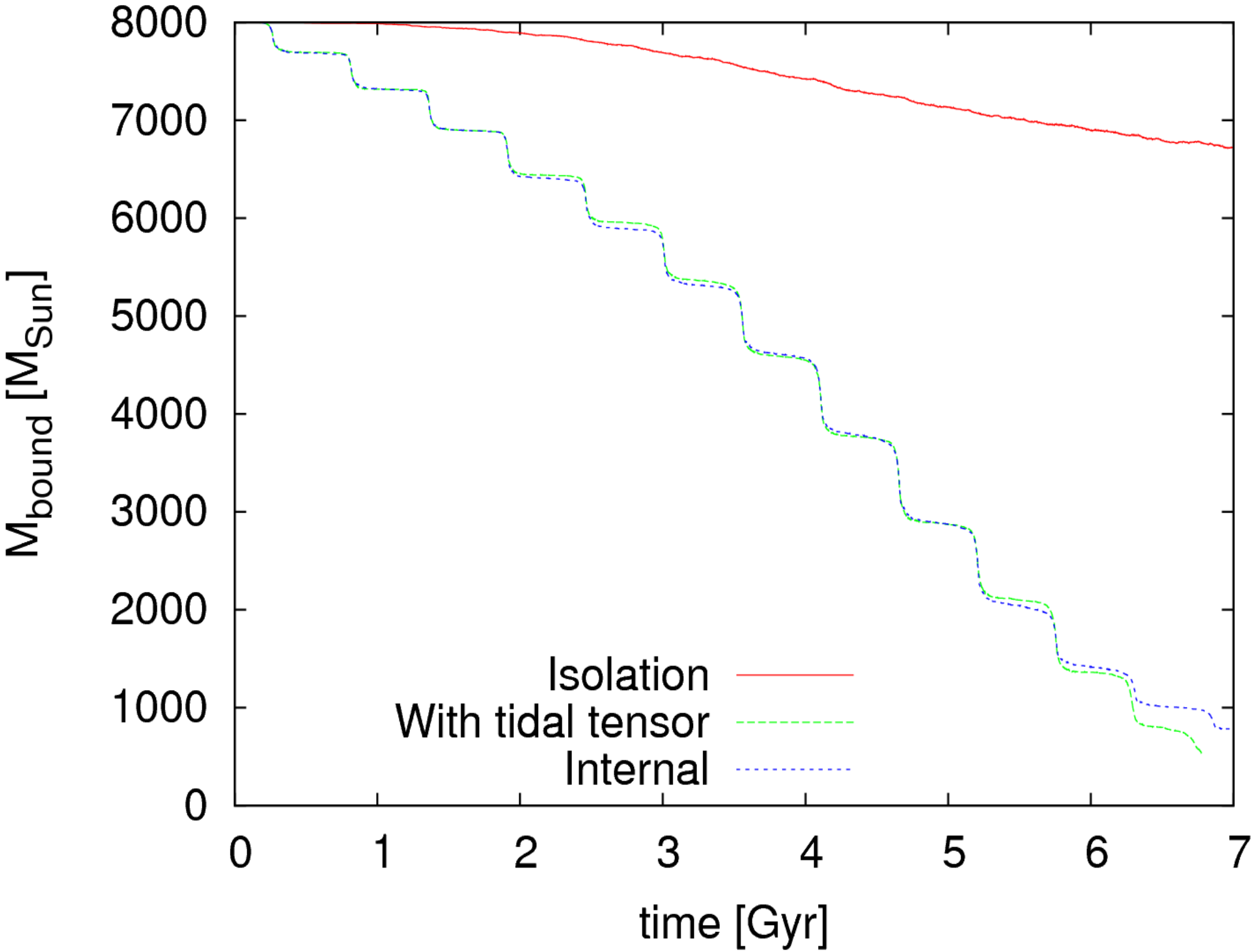
Bridge N-body integrator

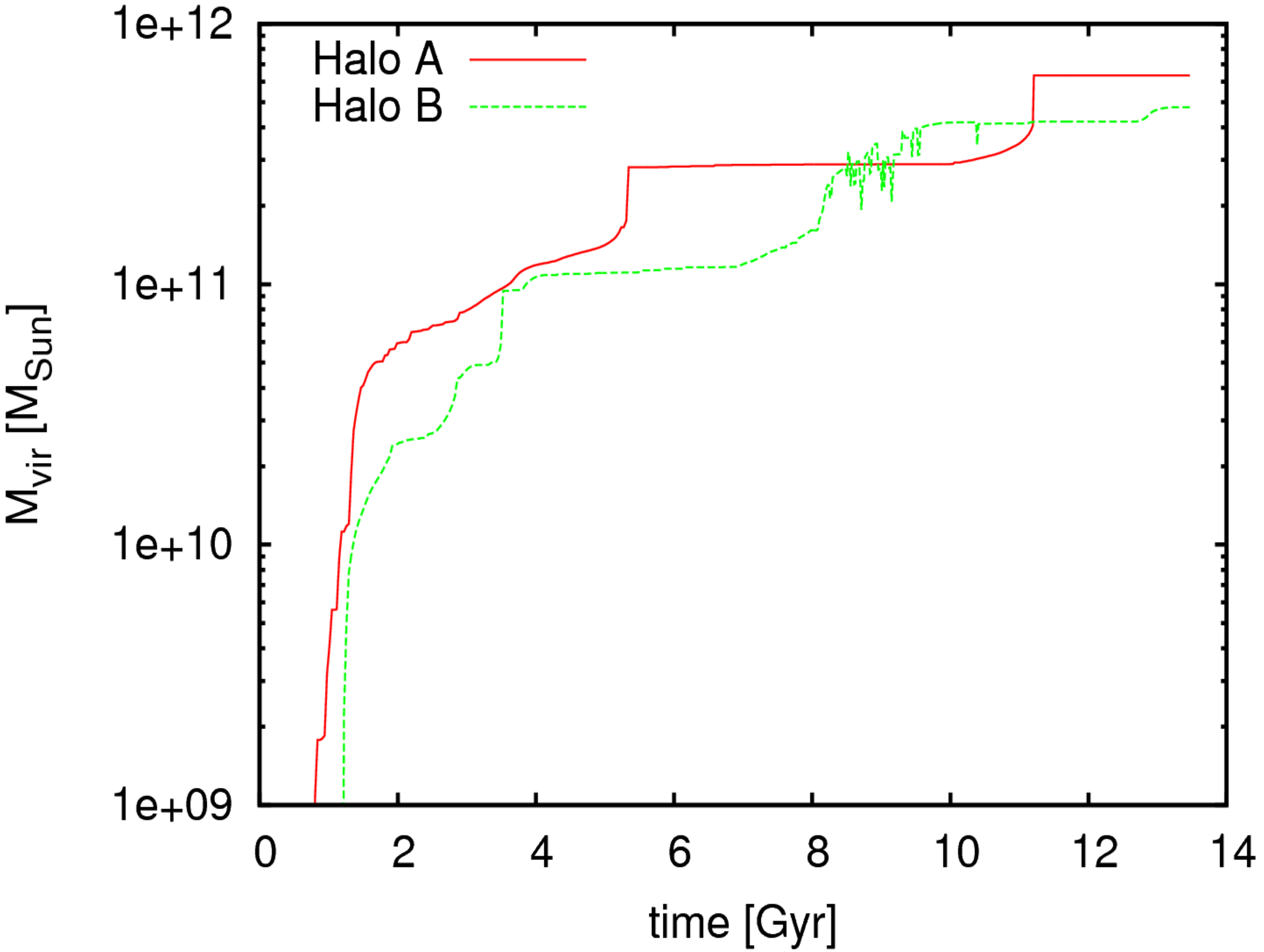
Fujii et al. 2007

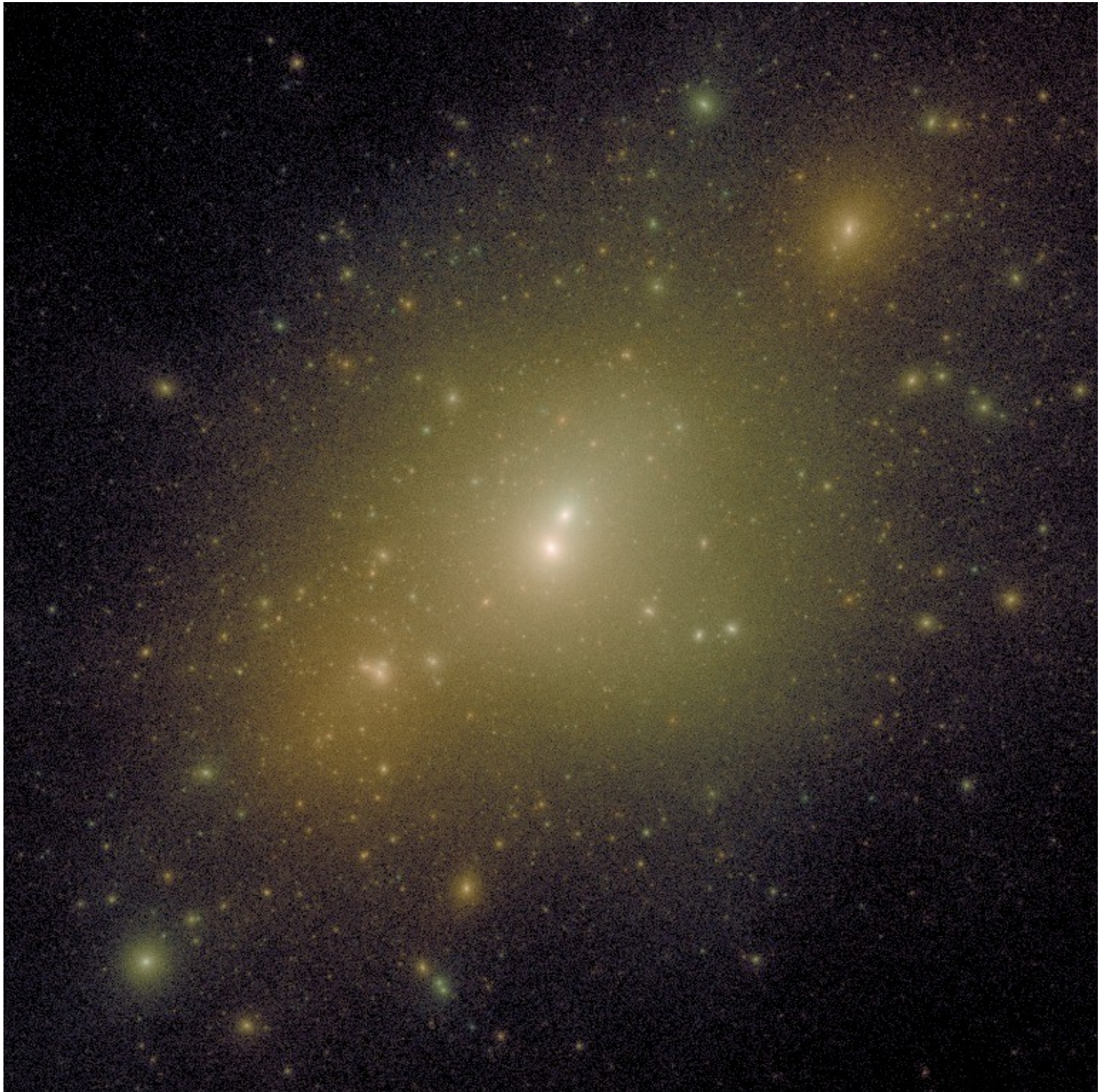


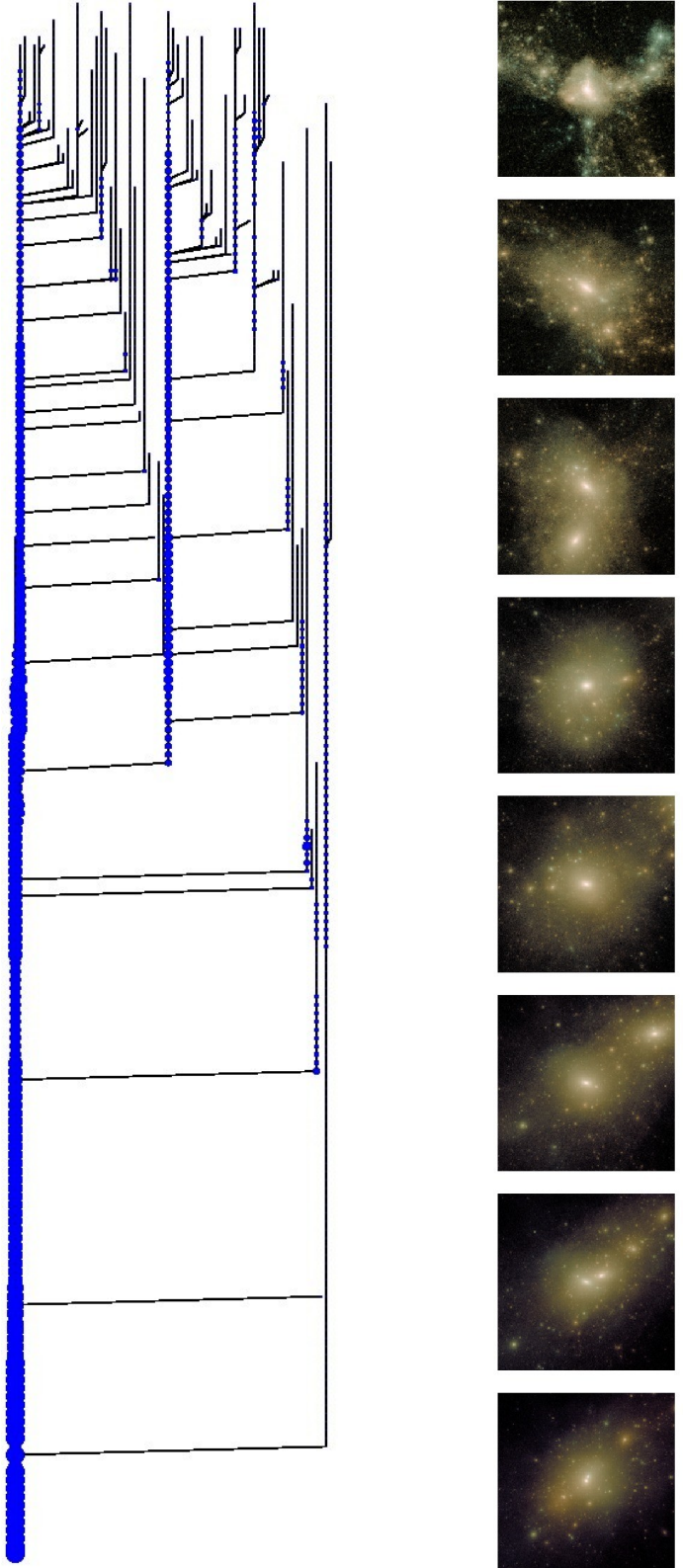
Almost symplectic N-body integrator- implementation in AMUSE:

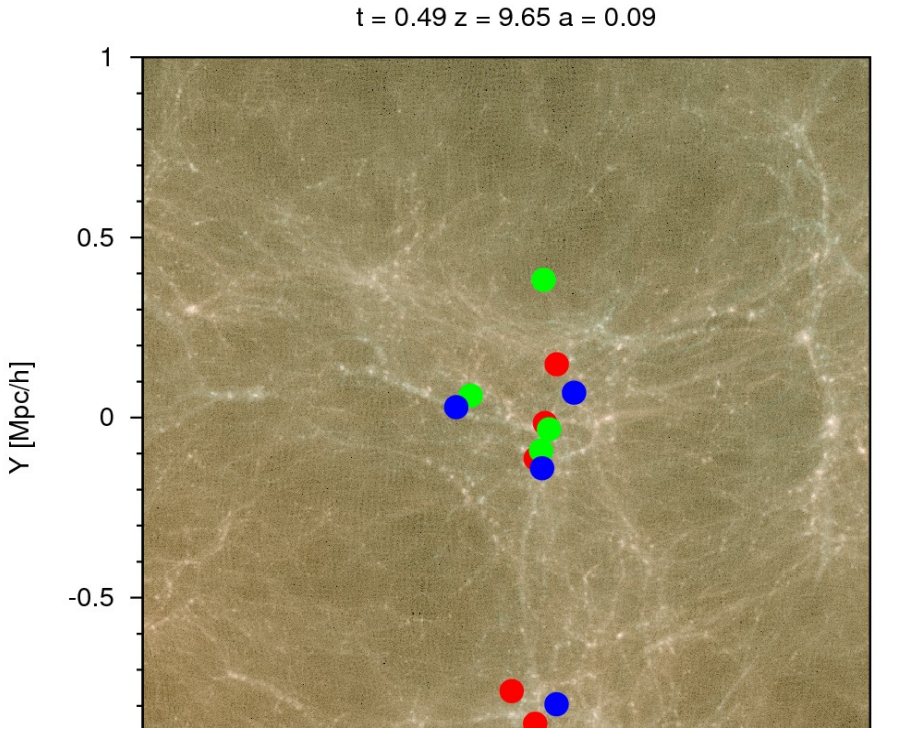
- interface includes function for gravity query
- implemented in python space
- combination of different integrators possible:
treecode, directe hermite, etc

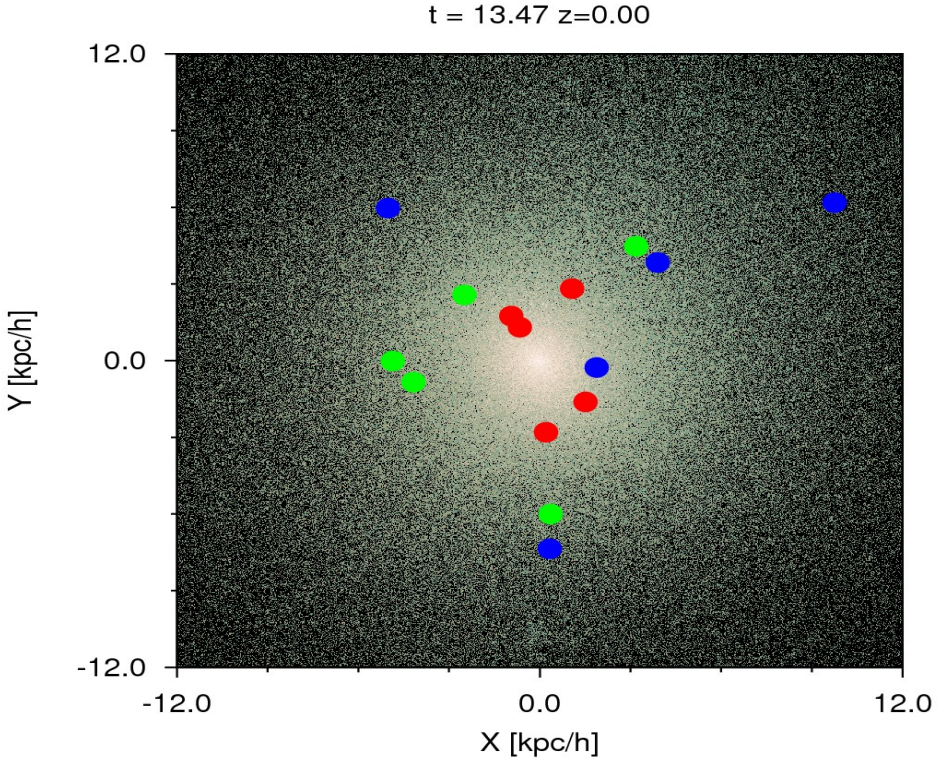

8000
7000
6000
5000
4000
3000
2000
1000
0
M_{bound} [M_{Sun}]
0
1
2
3
4
5
6
7
time [Gyr]
Isolation
With tidal tensor
Internal

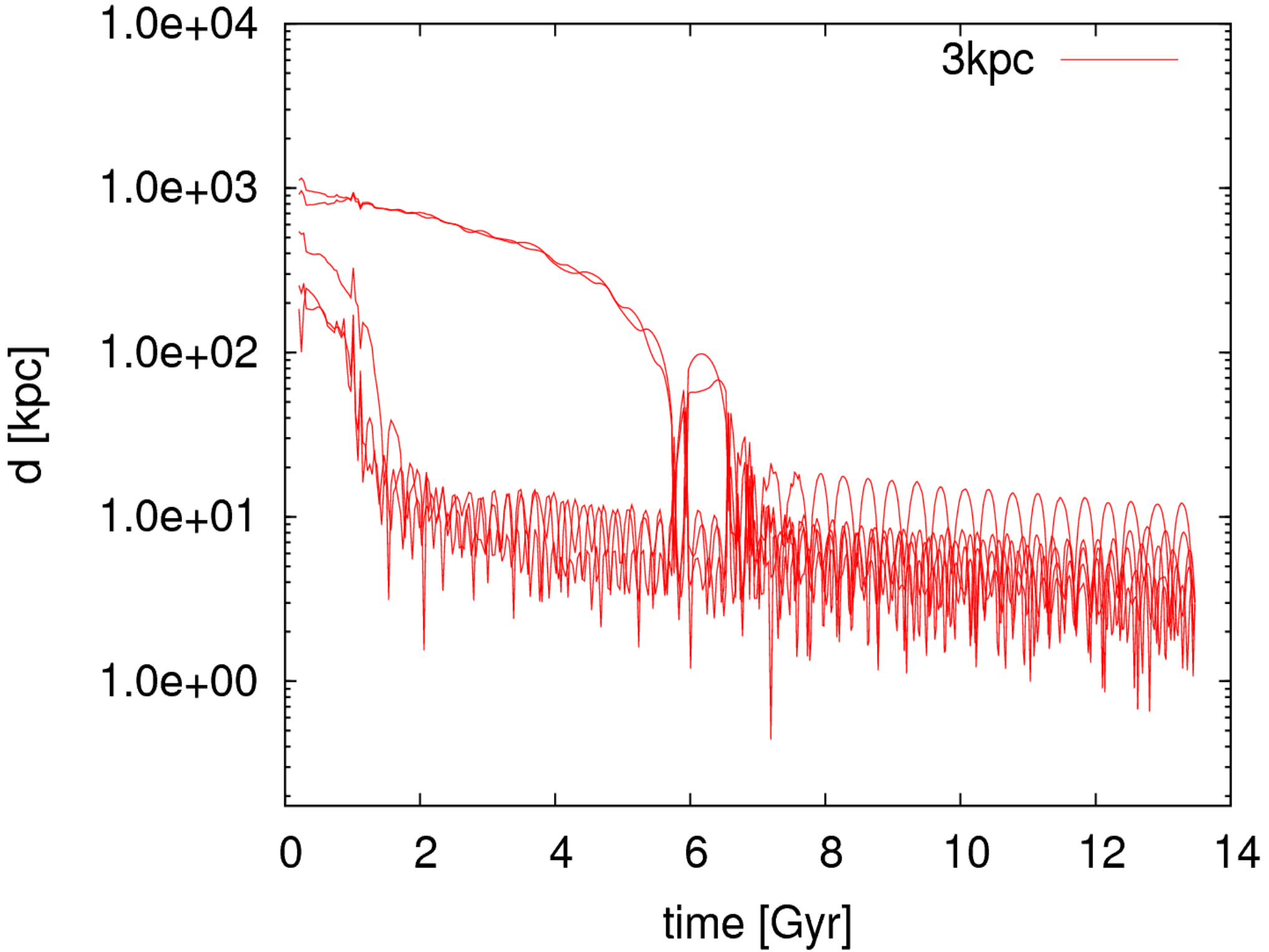

Halo A
Halo B
1e+12
1e+11
1e+10
1e+09
M_{vir} [M_{Sun}]
0
2
4
6
8
10
12
14
time [Gyr]

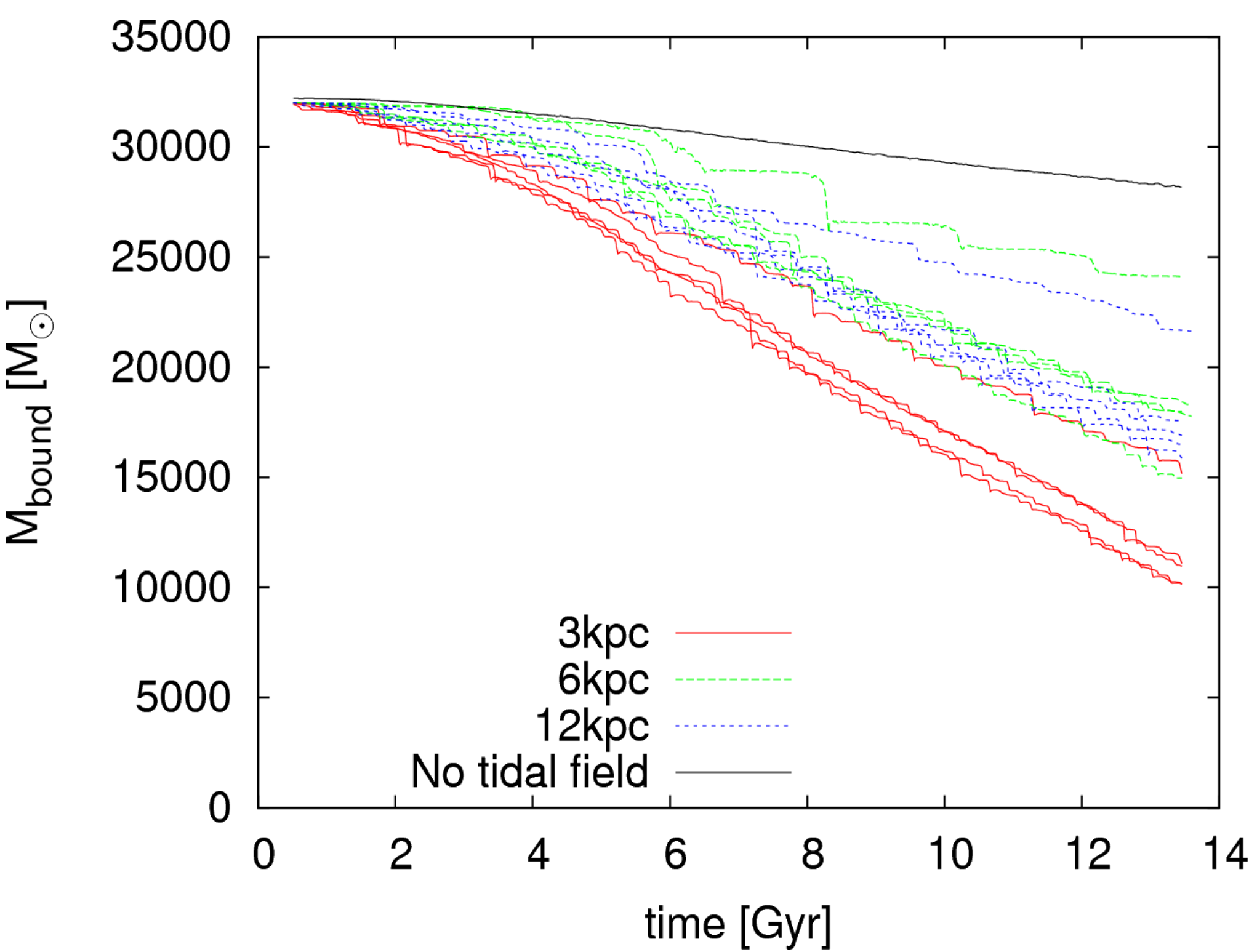





t = 0.49 z = 9.65 a = 0.09
1
0.5
0
-0.5
Y [Mpc/h]


t = 13.47 z=0.00
12.0
0.0
-12.0
Y [kpc/h]
-12.0
0.0
12.0
X [kpc/h]


1.0e+04
1.0e+03
1.0e+02
1.0e+01
1.0e+00
d [kpc]
3kpc
0
2
4
6
8
10
12
14
time [Gyr]


35000
30000
25000
20000
15000
10000
5000
0
M_{bound} [$M_\odot$]
0
2
4
6
8
10
12
14
time [Gyr]
3kpc
6kpc
12kpc
No tidal field

Conclusions

- We run a cosmological model over an intercontinental grid of supercomputers
- The data was used to simulate the background potential against which we evolved star clusters
- This was done with the AMUSE package
- We make the distinction between native and immigrant globular clusters
- Native clusters tend to lose mass more quickly than immigrants

WWW.NBabel.org

- Write a simple gravitational N-body code
- In any (new) language
- Predictor-corrector leapfrog
- Shared time-step
- Read-in initial snapshot (m, r, v)
- Integrate from t=0 to t=1
- Output (relative) energy error
- Current implementations: Python, C++, CUDA