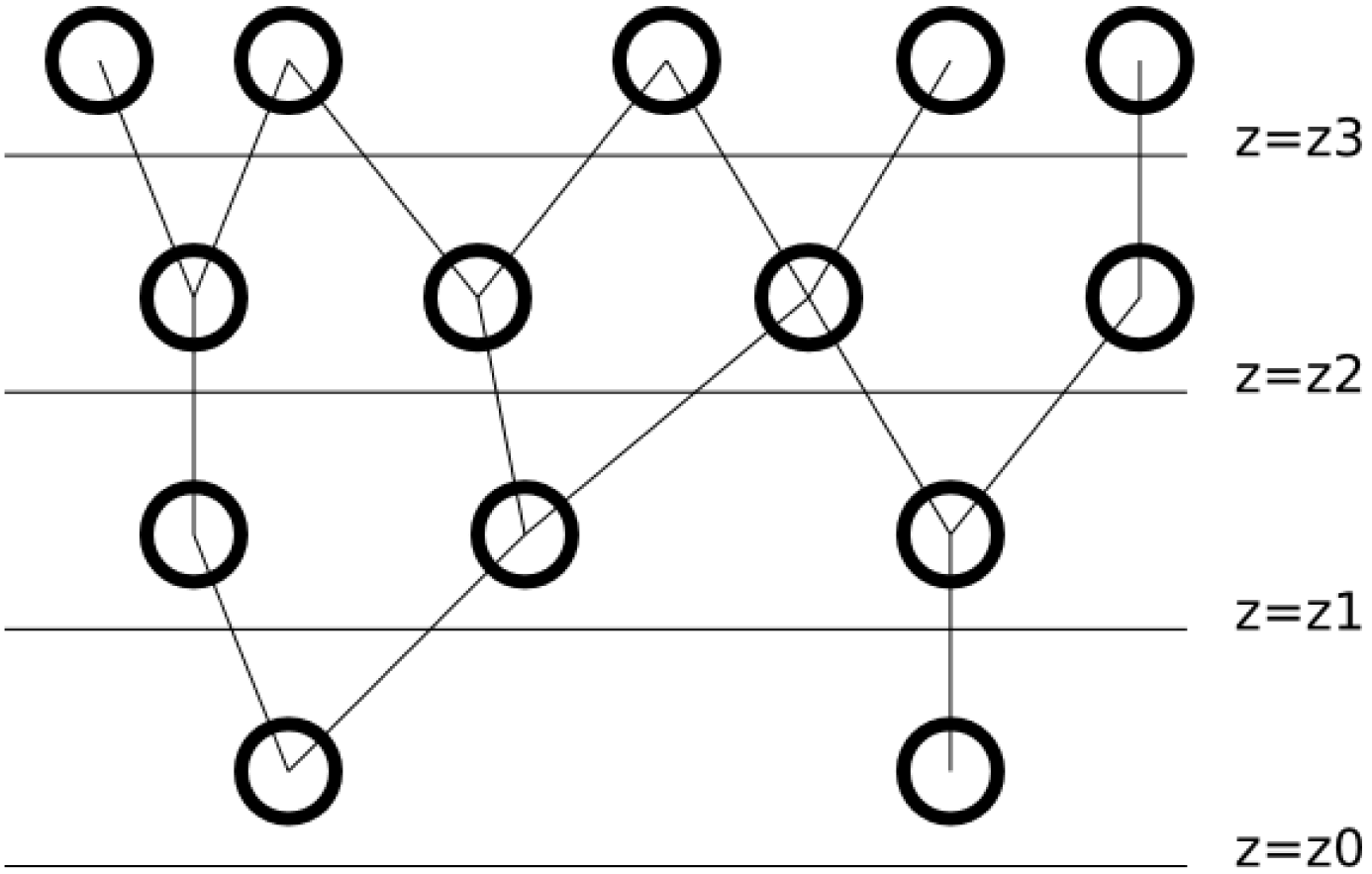
MergerTreeMPI

Edoardo Carlesi
(Universidad Autonoma de Madrid)

CLUES Meeting 2014
AIP, Potsdam

Merger Tree (Graph)



Merger Tree: Howto

- Input data: halos' particle content (ID)

- Pairwise catalog comparison:

z_0 -> z_1, z_1 -> z_2; ... z_{N-1} -> z_N

- When $HALO_j$ of z_0 and $HALO_k$ of z_1 have more than MIN_COMMON_PART in common we identify a pair descendant/progenitor

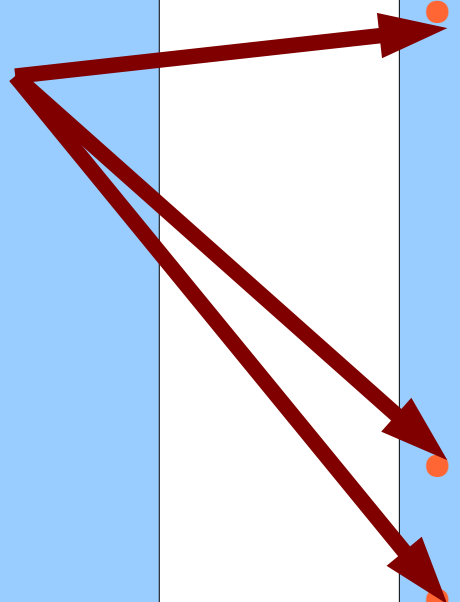
Halo Particle Content Comparison

Snapshot z_J

- Halo 1
  - Part_ID = 12345
  - Part_ID = 34567
  - .....
- .....
- Halo N
  - Part_ID = 56789
  - Part_ID = 78901
  - .....

Snapshot z_{J+1}

- Halo 1
  - Part_ID = 10235
  - Part_ID = 56789
  - .....
- .....
- Halo N_1
  - Part_ID = 36784
  - Part_ID = 58903
  - .....



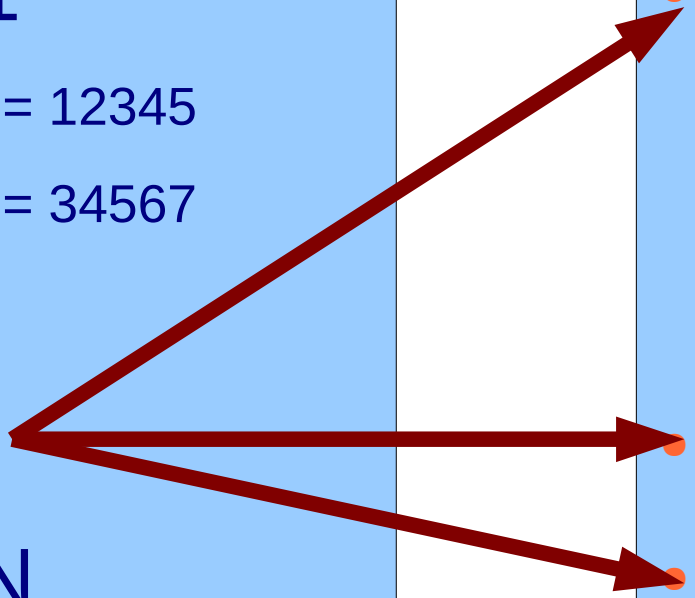
Halo Particle Content Comparison

Snapshot z_J

- Halo 1
  - Part_ID = 12345
  - Part_ID = 34567
  - .....
- .....
- Halo N
  - Part_ID = 56789
  - Part_ID = 78901
  - .....

Snapshot z_{J+1}

- Halo 1
  - Part_ID = 10235
  - Part_ID = 56789
  - .....
- .....
- Halo N_1
  - Part_ID = 36784
  - Part_ID = 58903
  - .....



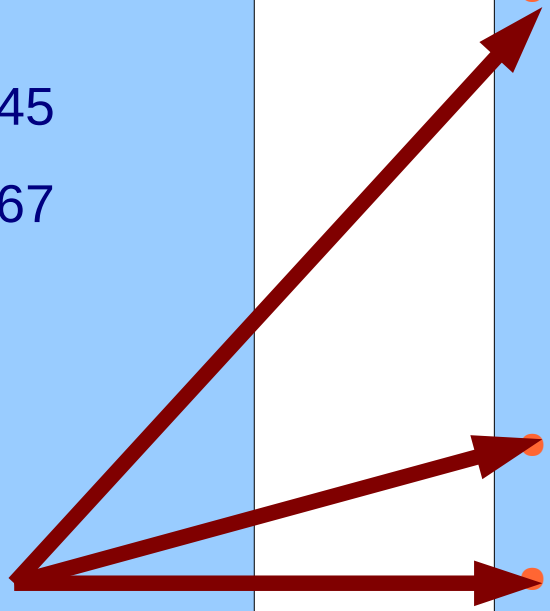
Halo Particle Content Comparison

Snapshot z_J

- Halo 1
  - Part_ID = 12345
  - Part_ID = 34567
  - .....
- .....
- Halo N
  - Part_ID = 56789
  - Part_ID = 78901
  - .....

Snapshot z_{J+1}

- Halo 1
  - Part_ID = 10235
  - Part_ID = 56789
  - .....
- .....
- Halo N_1
  - Part_ID = 36784
  - Part_ID = 58903
  - .....



Serial Merger Tree

Snapshot z_J

- PartID 1
  - Halo_ID = 12345
  - .....
- .....
- PartID 12345
- ....
- PartID N
  - Halo_ID = 56789
  - .....

Snapshot z_{J+1}

- PartID 1
  - Halo_ID = 10235
  - .....
- .....
- PartID 12344
- .....
- PartID N_1
  - Halo_ID = 36784
  - .....

Serial Merger Tree

PRO:

- Very efficient algorithm (approx linear in N_part)

CONTRA:

- Memory demanding: e.g. 2048^3 particles simulation require >10 TB on a single node

- OpenMP parallel only, limited by cores on a single node

Parallel Merger Tree

- Two halo catalogs: SNAP_00 and SNAP_01

- Distribute SNAP_00 and SNAP_01 across N MPI tasks (load balancing):

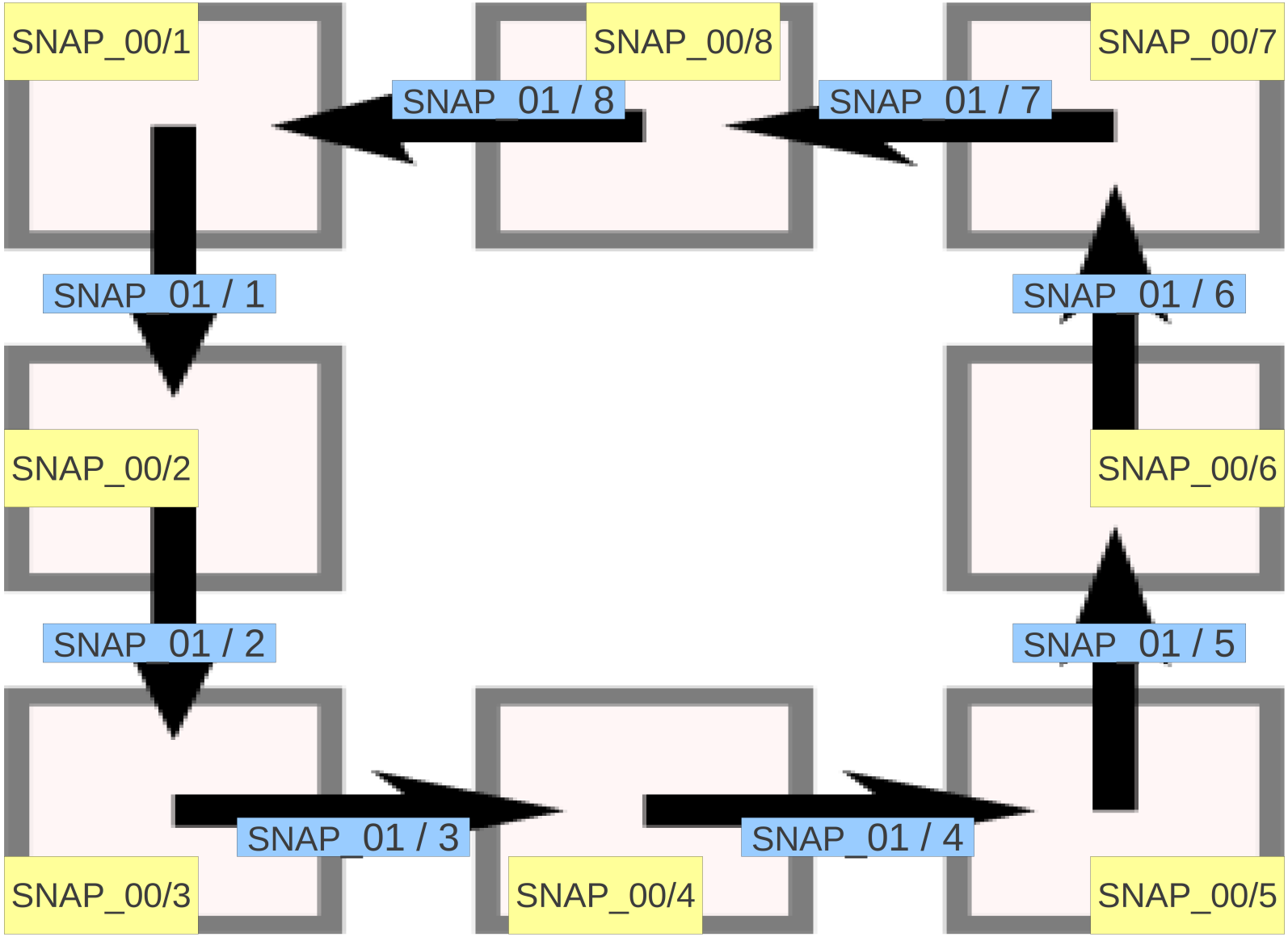
  - Task 1 = SNAP_00/1, SNAP_01/1
  - Task 2 = SNAP_00/2, SNAP_01/2

  [...]

  - Task N = SNAP_00/N, SNAP_01/N

- Swap SNAP_01/j across tasks

Data Swapping Across Tasks



Parallel vs. Serial

PRO:

- Possibility to use distributed memory machines and several cores

CONTRA:

- Very expensive: each tasks performs approx. $\mathbf{N_j} \times \mathbf{N_j}$ operations (N_j = parts per task), serial version is prop. to $\mathbf{N_j}$

Improvements

- Load Balancing: minimize the number of idle tasks, better workload distribution (N_part * sqrt(N_halos))

- Add OpenMP

- Compare only haloes whose c.o.m distance is smaller than MAX_HALO_DIST

TEST 1

- Three CLUES-WDM snapshots (approx. $4x10^7$ particles each)

- Without MAX_HALO_DIST, total runtime for one step on 4 tasks = **1130** *s*

- With MAX_HALO_DIST = 3 Mpc/h, total runtime for one step on 4 tasks = **83** *s*

- Serial runtime on the same machine = **79** *s*

- 100% match with the serial results

TEST 2

- For 1024^3 particles in 250 Mpc/h box

- With MAX_HALO_DIST = 5 Mpc/h, total runtime for one step on 512 tasks = **103** *s*

- Total time estimated for 60 catalogs is approx. 10^3 core hours. Serial version requires 1 week (approx. 200 core hours)

Ongoing Work

- Use of *MPI_Pack / MPI_Unpack* functions to swap data will **reduce up to 50%** the memory required per task

- Introduce an HALO_MIN_PART threshold to remove smaller haloes reduces less significan comparison