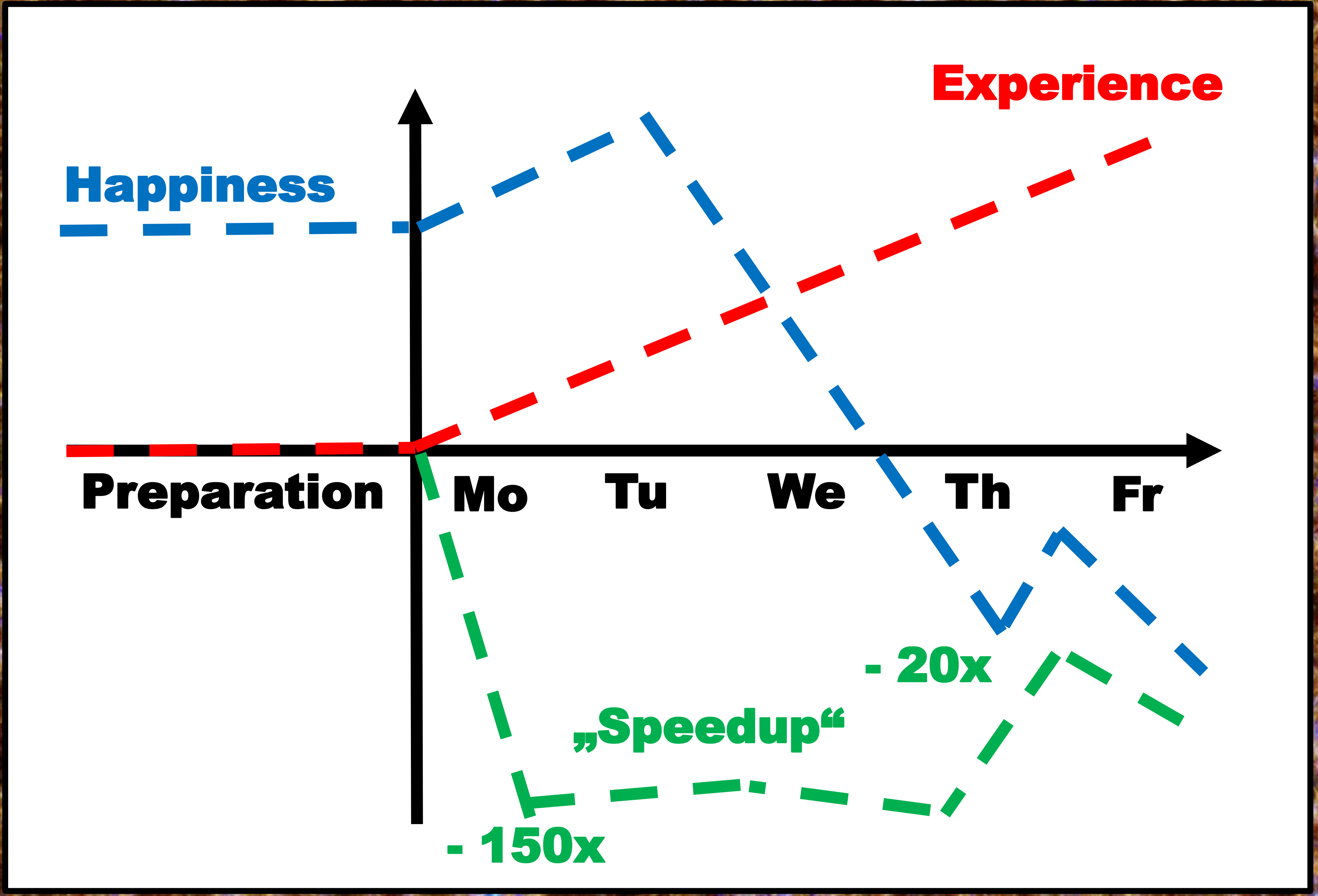
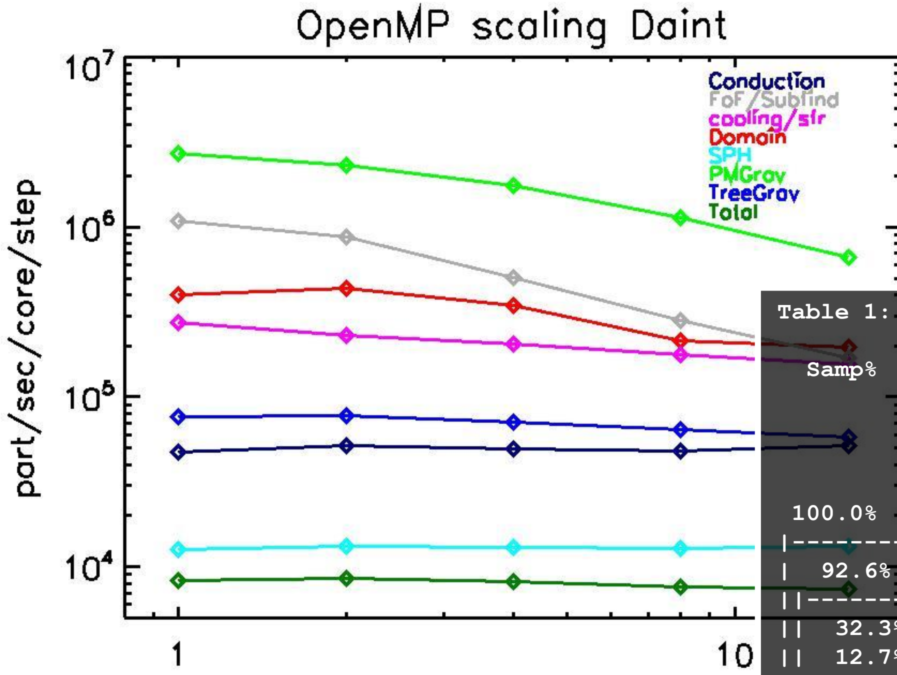
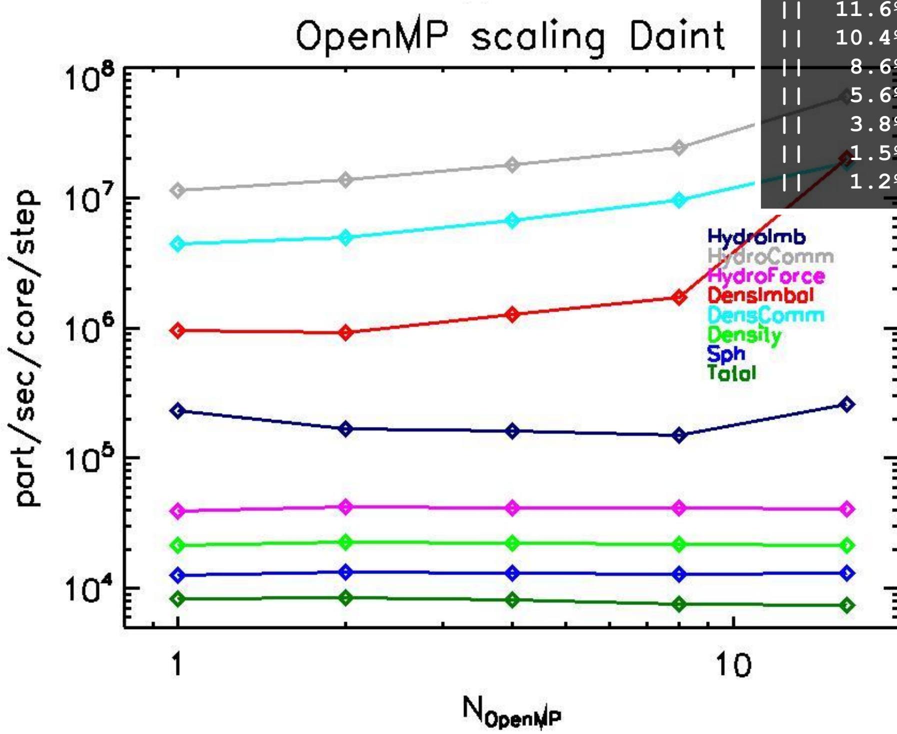
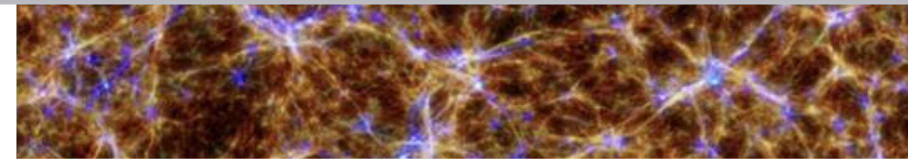
Gadget-ACC Summary



Gadget profiling



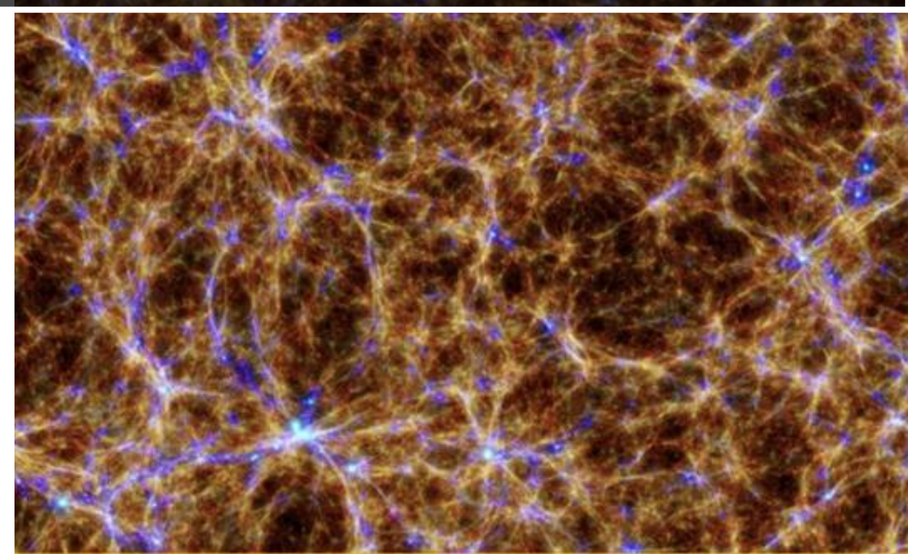




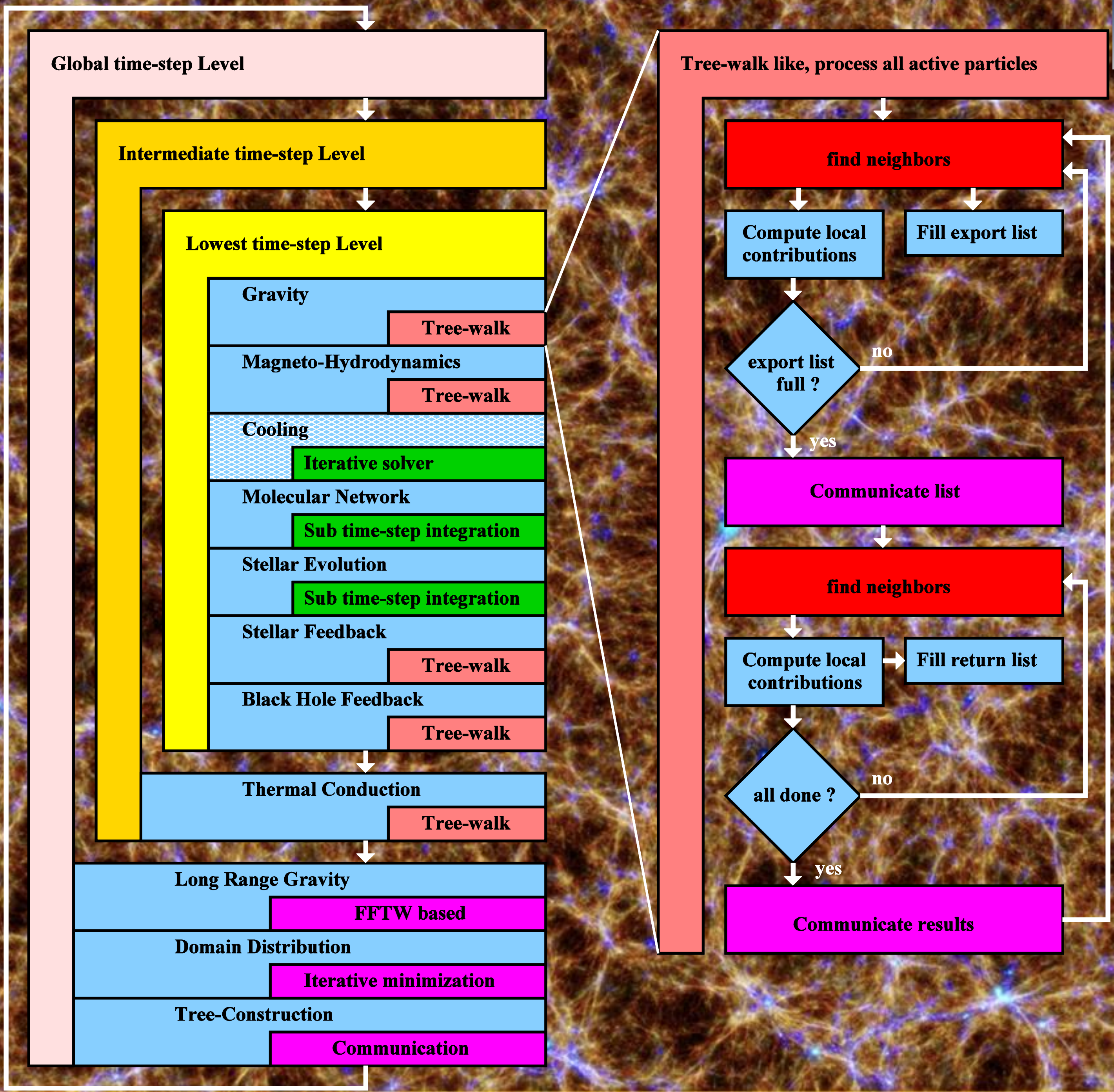
```
Table 1:  Profile by Function

  Samp% |      Samp |  Imb. |  Imb. |Group
        |           |  Samp | Samp% | Function
        |           |       |       |  PE=HIDE
        |           |       |       |   Thread=HIDE

 100.0% | 599768.5 |    -- |    -- |Total
|------------------------------------------------------------------
|  92.6% | 555260.0 |    -- |    -- |USER
||-----------------------------------------------------------------
||  32.3% | 193441.0 | 389.0 |  0.4% |ngb_treefind_variable_threads
||  12.7% |  76407.5 | 436.5 |  1.1% |density_evaluate
||  11.6% |  69755.5 |  74.5 |  0.2% |ngb_treefind_pairs_threads
||  10.4% |  62577.5 | 724.5 |  2.3% |density_evaluate_extra_physics_gas
||   8.6% |  51671.5 |  76.5 |  0.3% |force_treeevaluate_shortrange
||   5.6% |  33488.0 | 214.0 |  1.3% |kernel_main
||   3.8% |  22614.5 |  71.5 |  0.6% |hydro_evaluate
||   1.5% |   9084.5 |   8.5 |  0.2% |DMAX
||   1.2% |   6966.5 |  13.5 |  0.4% |conduction_evaluate
```



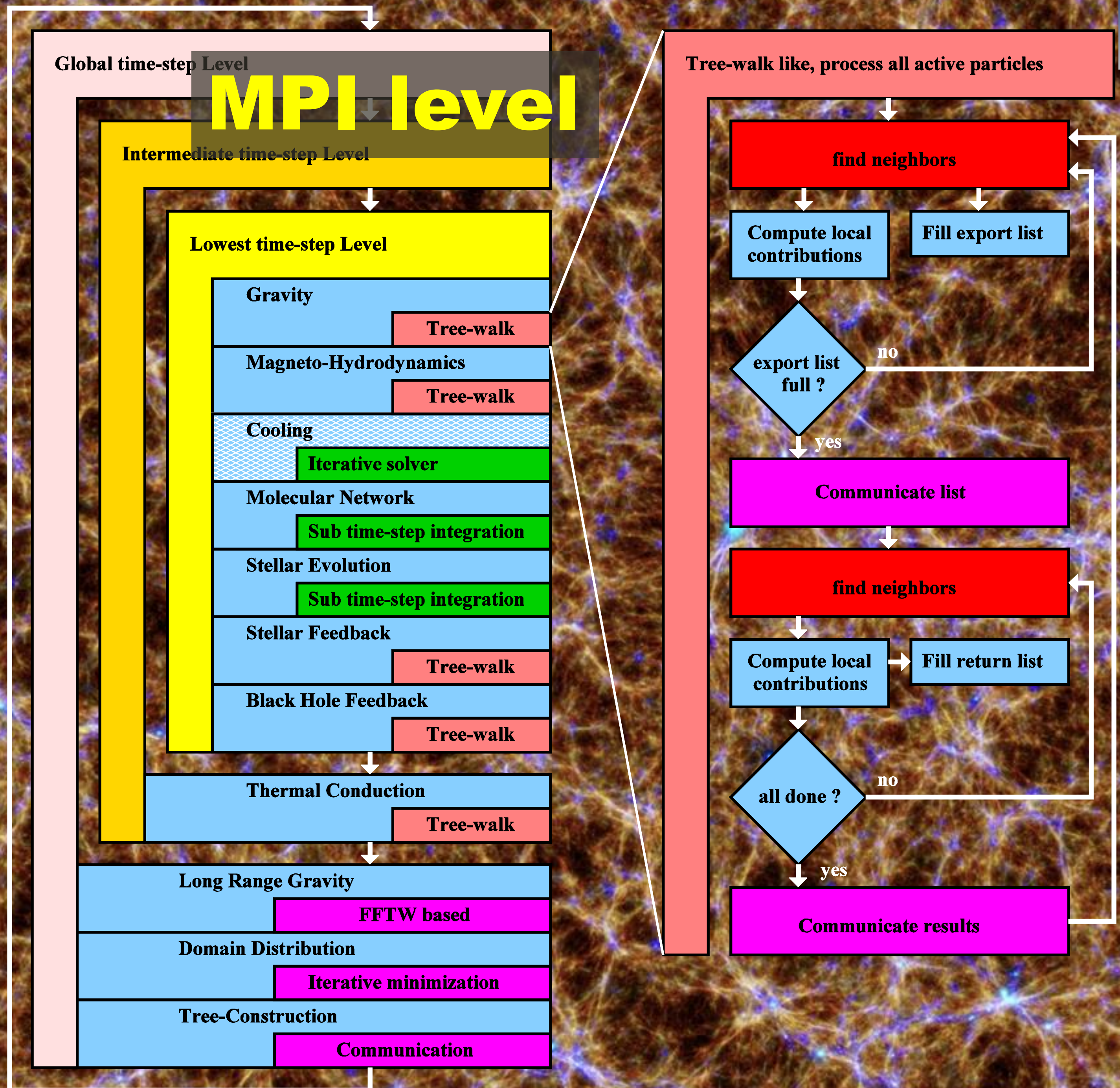
Interplay of different physics modules in Gadget3

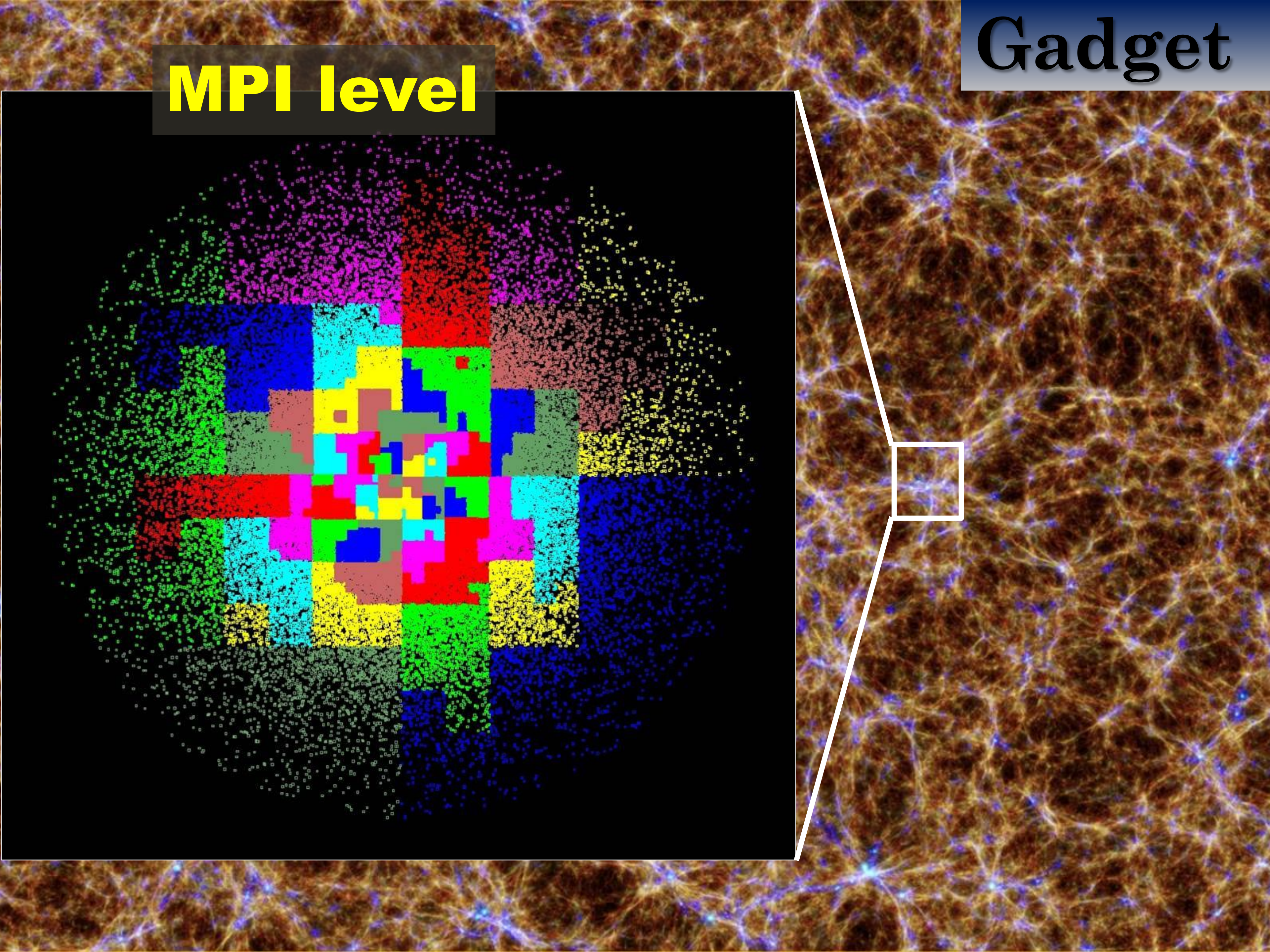


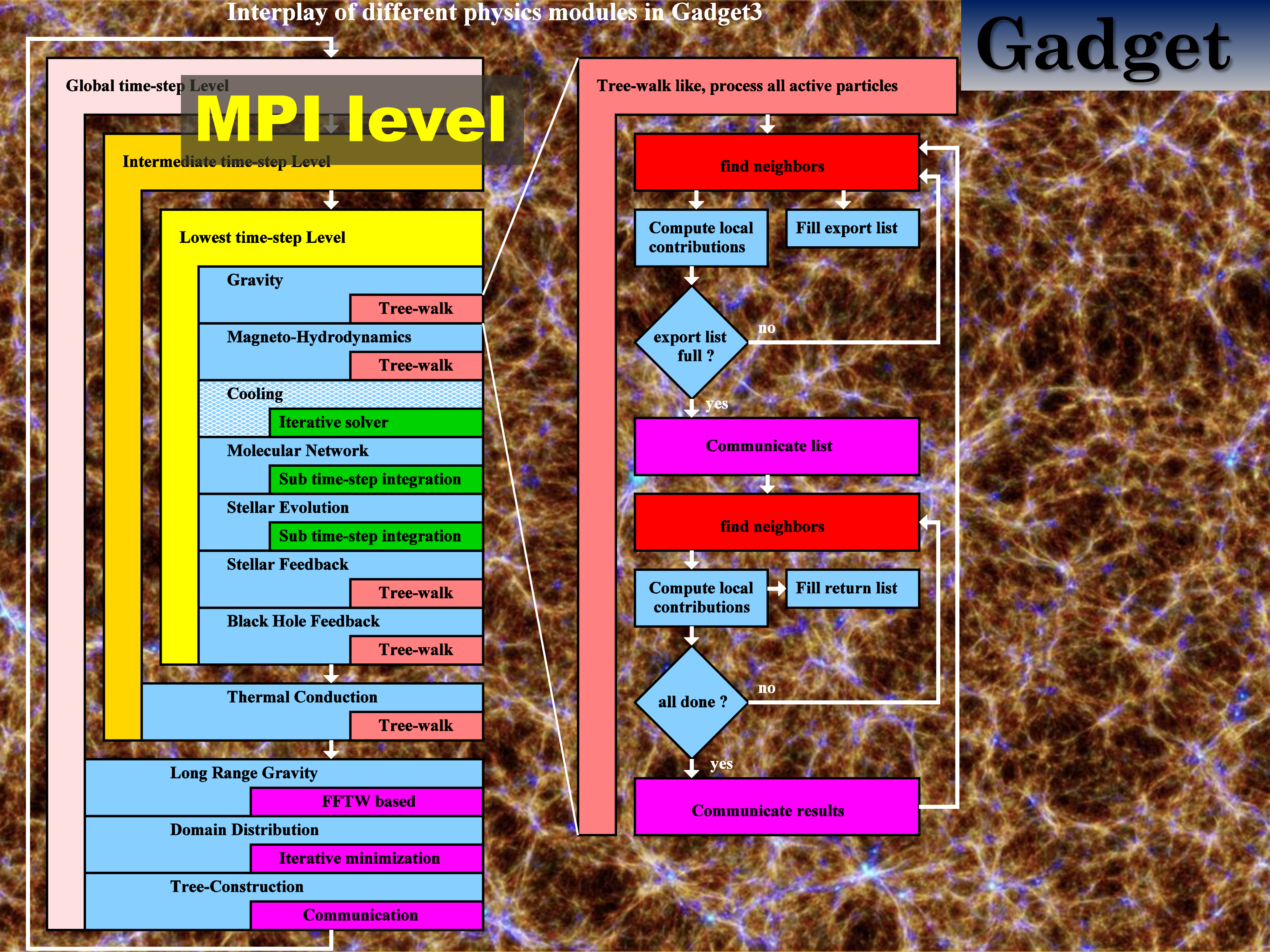
Gadget

Interplay of different physics modules in Gadget3

Gadget

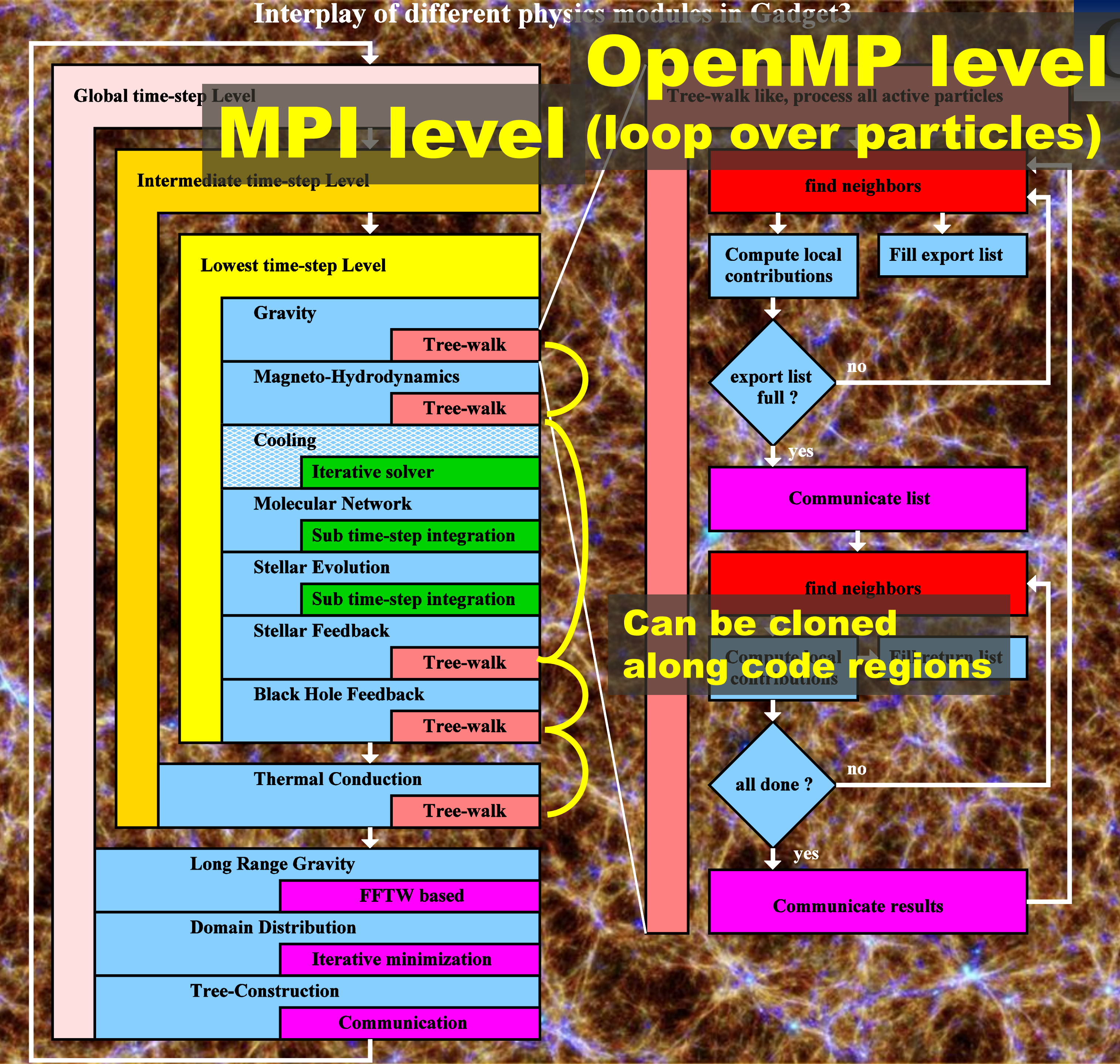



Gadget
MPI level

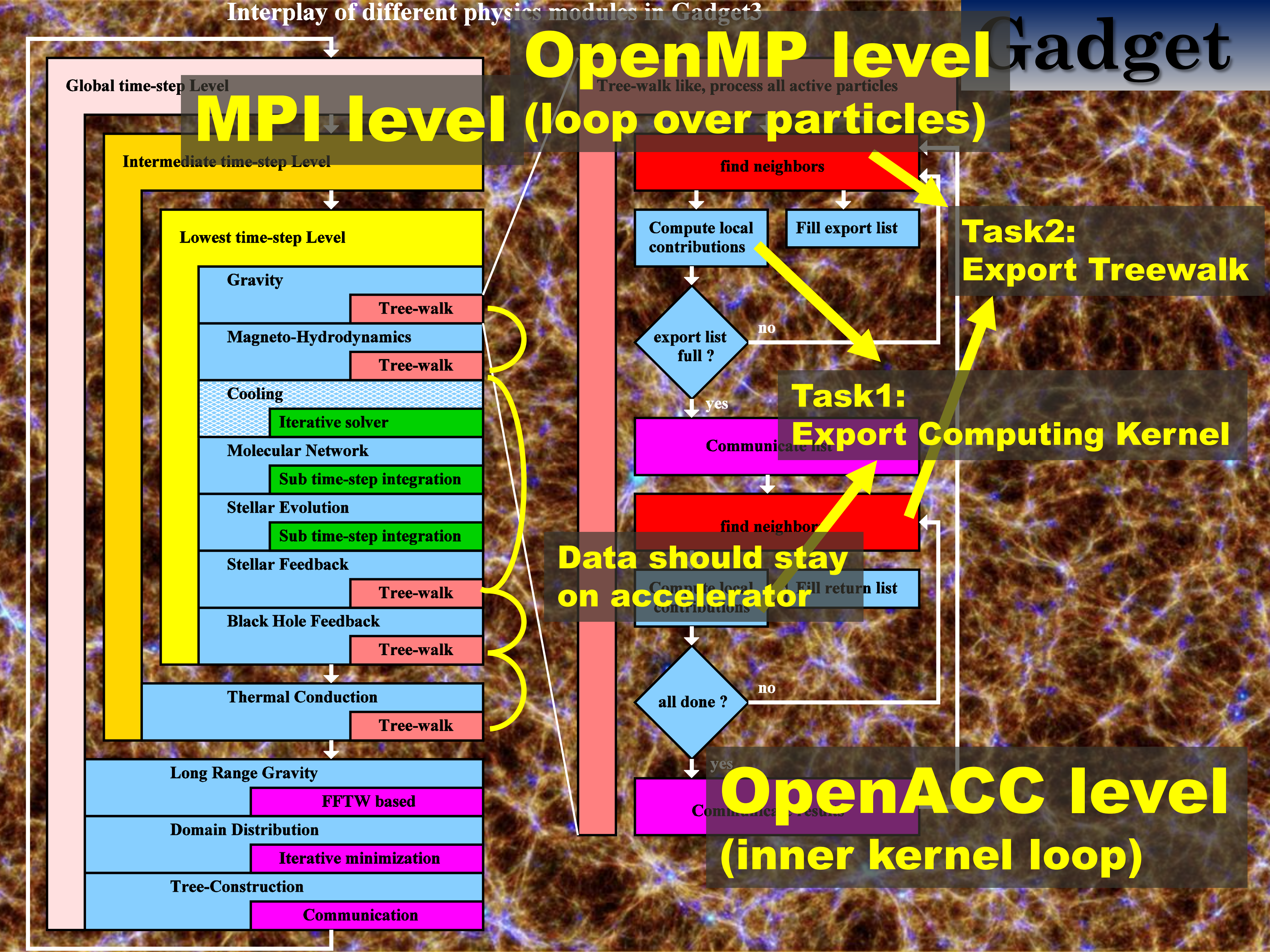


Interplay of different physics modules in Gadget3
Gadget
Global time-step Level
MPI level
Intermediate time-step Level
Lowest time-step Level
Gravity
Tree-walk
Magneto-Hydrodynamics
Tree-walk
Cooling
Iterative solver
Molecular Network
Sub time-step integration
Stellar Evolution
Sub time-step integration
Stellar Feedback
Tree-walk
Black Hole Feedback
Tree-walk
Thermal Conduction
Tree-walk
Long Range Gravity
FFTW based
Domain Distribution
Iterative minimization
Tree-Construction
Communication
Tree-walk like, process all active particles
find neighbors
Compute local contributions
Fill export list
export list full ?
no
yes
Communicate list
find neighbors
Compute local contributions
Fill return list
all done ?
no
yes
Communicate results

Gadget



Interplay of different physics modules in Gadget3
OpenMP level
MPI level
(loop over particles)
Global time-step Level
Intermediate time-step Level
Lowest time-step Level
Gravity
Tree-walk
Magneto-Hydrodynamics
Tree-walk
Cooling
Iterative solver
Molecular Network
Sub time-step integration
Stellar Evolution
Sub time-step integration
Stellar Feedback
Tree-walk
Black Hole Feedback
Tree-walk
Thermal Conduction
Tree-walk
Long Range Gravity
FFTW based
Domain Distribution
Iterative minimization
Tree-Construction
Communication
Tree-walk like, process all active particles
find neighbors
Compute local contributions
Fill export list
export list full ?
no
yes
Communicate list
find neighbors
all done ?
no
yes
Communicate results
Can be cloned along code regions



Interplay of different physics modules in Gadget3
Gadget
OpenMP level
MPI level
(loop over particles)
Global time-step Level
Intermediate time-step Level
Lowest time-step Level
Gravity
Tree-walk
Magneto-Hydrodynamics
Tree-walk
Cooling
Iterative solver
Molecular Network
Sub time-step integration
Stellar Evolution
Sub time-step integration
Stellar Feedback
Tree-walk
Black Hole Feedback
Tree-walk
Thermal Conduction
Tree-walk
Long Range Gravity
FFTW based
Domain Distribution
Iterative minimization
Tree-Construction
Communication
Tree-walk like, process all active particles
find neighbors
Compute local contributions
Fill export list
export list full ?
no
yes
Communicate list
find neighbors
Fill return list
all done ?
no
yes
Task2:
Export Treewalk
Task1:
Export Computing Kernel
Data should stay on accelerator
OpenACC level
(inner kernel loop)

Gadget

```
Strategy:
Cores guided OpenACC calculation
```

```
#pragma acc data copy(P[0:NumPart])

:

fac=P[j].something;
ngb=find_neighbours(j,*list);
double dv;
#pragma acc parallel loop reduction(+dv)\
             copyin(fac,list[0:ngb])
for(i=0;i<ngb;i++)
   dv+=fac*(P[i].x-P[j].x);
P[j].dv=dv;

:
```

```
In principle: All could stay on GPU !
```

Gadget

But:
Large losses by non needed data transfere

```
method=[ memcpyHtoD ] gputime=[ 4.192 ] cputime=[ 13.325 ] memtransfersize=[ 19432 ]
method=[ memcpyHtoD ] gputime=[ 7805.088 ] cputime=[ 7865.931 ] memtransfersize=[ 46766808 ]
method=[ memcpyHtoD ] gputime=[ 1.120 ] cputime=[ 8.096 ] memtransfersize=[ 168 ]
method=[ memcpyHtoD ] gputime=[ 0.864 ] cputime=[ 6.331 ] memtransfersize=[ 108 ]
method=[ memcpyHtoD ] gputime=[ 1.184 ] cputime=[ 6.451 ] memtransfersize=[ 1184 ]
method=[ memcpyHtoD ] gputime=[ 0.864 ] cputime=[ 5.387 ] memtransfersize=[ 32 ]
method=[ memcpyHtoD ] gputime=[ 0.864 ] cputime=[ 5.159 ] memtransfersize=[ 48 ]
method=[ memcpyHtoD ] gputime=[ 0.896 ] cputime=[ 5.327 ] memtransfersize=[ 4 ]
method=[ hydro_evaluate$ck_L2376_1 ] gputime=[ 49.792 ] cputime=[ 16.593 ] occupancy=[ 0.375 ]
method=[ memcpyDtoH ] gputime=[ 2.432 ] cputime=[ 18.836 ] memtransfersize=[ 32 ]
method=[ memcpyDtoH ] gputime=[ 2.368 ] cputime=[ 13.374 ] memtransfersize=[ 48 ]
```

```
ACC: Start transfer 2 items from hydra_acc.c:1009
ACC:       allocate, copy to acc 'All' (19432 bytes)
ACC:       allocate, copy to acc 'HydroNgbGet' (46766808 bytes)
ACC: End transfer (to acc 46786240 bytes, to host 0 bytes)
ACC: Start transfer 14 items from hydra_acc.c:2376
ACC:       present 'All' (19432 bytes)
ACC:       present 'HydroNgbGet' (46766808 bytes)
ACC:       allocate, copy to acc 'kernel' (168 bytes)
:
ACC: End transfer (to acc 1544 bytes, to host 0 bytes)
ACC: Execute kernel hydro_evaluate$ck_L2376_1 blocks:3 threads:128 async(auto)
                                                        from hydra_acc.c:2376
ACC: Wait async(auto) from hydra_acc.c:3331
```

In principle: All could stay on GPU !

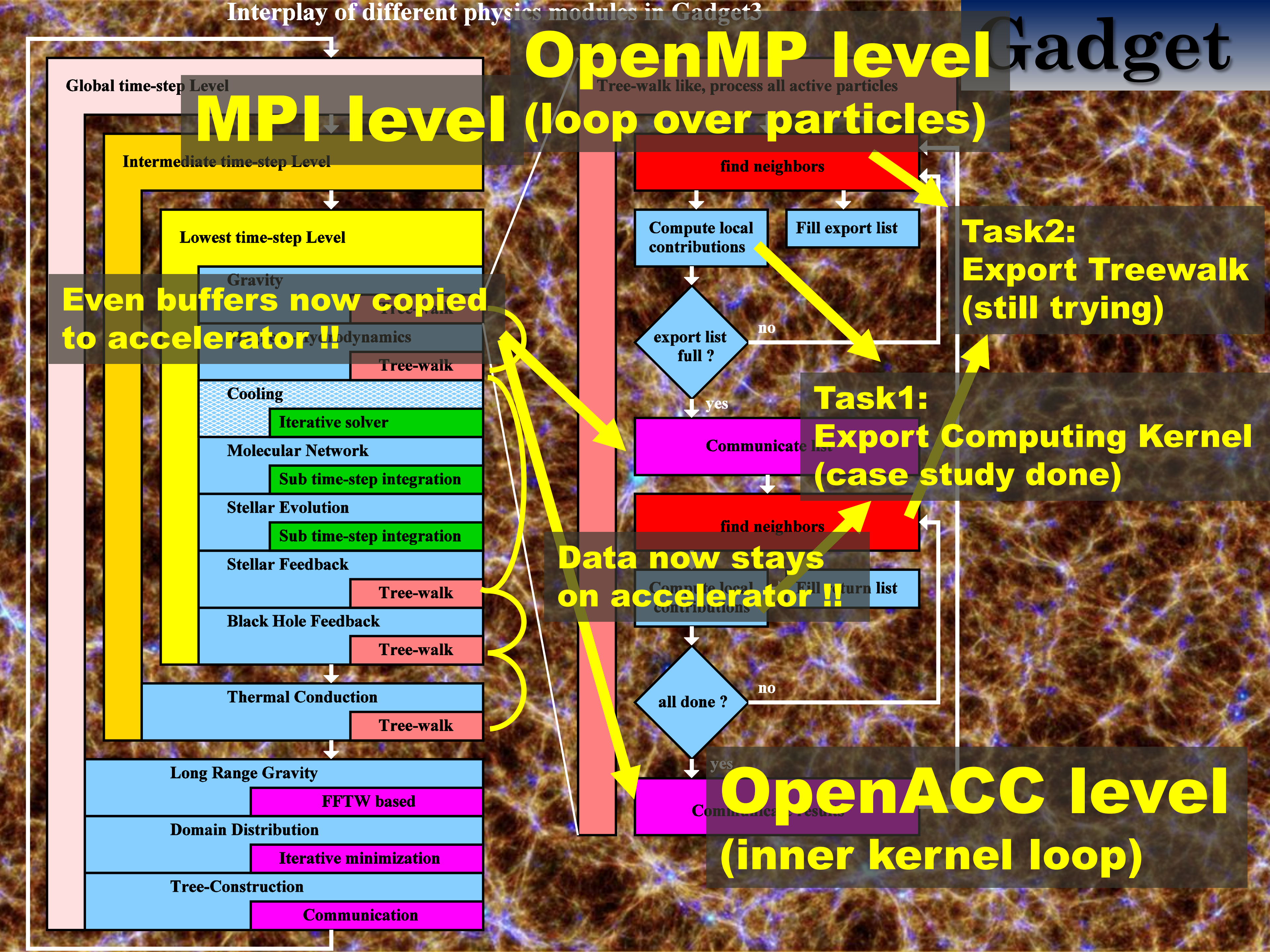
Gadget

Solution: (working but false results)

```
:
ngb=find_neighbours(j,*list);
#pragma acc parallel copyin(list[0:ngb])
{
fac=P[j].something;
double dv;
#pragma acc parallel loop reduction(+dv)
for(i=0;i<ngb;i++)
   dv+=fac*(P[i].x-P[j].x);
P[j].dv=dv;
}
:
```

```
:
method=[ hydro_evaluate_ngblist$ck_L1932_4 ] gputime=[ 66.304 ] cputime=[ 9.000 ] occupancy=[ 0.250 ]
method=[ memcpyHtoD ] gputime=[ 1.216 ] cputime=[ 7.187 ] memtransfersize=[ 1140 ]
method=[ hydro_evaluate_ngblist$ck_L1932_4 ] gputime=[ 64.928 ] cputime=[ 7.709 ] occupancy=[ 0.250 ]
method=[ memcpyHtoD ] gputime=[ 1.056 ] cputime=[ 11.176 ] memtransfersize=[ 1132 ]
:
```

Problem: No reduction possible within parallel region !


Interplay of different physics modules in Gadget3
Gadget
OpenMP level
(loop over particles)
MPI level
Global time-step Level
Intermediate time-step Level
Lowest time-step Level
Gravity
Tree-walk
Hydrodynamics
Tree-walk
Cooling
Iterative solver
Molecular Network
Sub time-step integration
Stellar Evolution
Sub time-step integration
Stellar Feedback
Tree-walk
Black Hole Feedback
Tree-walk
Thermal Conduction
Tree-walk
Long Range Gravity
FFTW based
Domain Distribution
Iterative minimization
Tree-Construction
Communication
Even buffers now copied
to accelerator !!
Tree-walk like, process all active particles
find neighbors
Compute local contributions
Fill export list
export list full ?
no
yes
Communicate list
find neighbors
Fill return list
all done ?
no
yes
Task2:
Export Treewalk
(still trying)
Task1:
Export Computing Kernel
(case study done)
Data now stays
on accelerator !!
OpenACC level
(inner kernel loop)

Gadget-ACC working summary

```
PGI (not working):
*) first private in loop
*) address not present in table in loop (acc+OpenMP)
⇒ No OpenMP+OpenACC possible
*) compiler bug #1 (can't pass struct by value)
*) compiler bug #2 (can't de-reference struct)
```

```
Cray (not working):
*) Internal compiler error:
#pragma novector
  for(i = nr + 1; i < Nblocks; i++)
    length += BlockSize[i];
*) async (with OpenMP) does not work (compiler bug)
⇒ No GPU speedup when switcheing on OpenMP+OpenACC
*) copyin(P[0,N])  (not defined N not catched)
```

Gadget-ACC wishlist

```
#pragma acc parallel loop reduction(+P[i].acc[0])
```

```
struct particle_data {
  double Pos[3];
  short int Type;
#pragma acc deepcopy(none)
  MyIDType ID;
: } *P
```

```
#pragma acc data copyin(P[0:numPart])
```

```
#pragma acc parallel {
double dv=0;
#pragma acc parallel loop reduction(+dv)
for(i=0;i<ngb;i++)
   dv+=(P[i].x-P[j].x)
#pragma acc reduce(dv)
P[j].dv=dv }
```

```
#pragma acc single
dv = P[j].dv;
```