German Research School
for Simulation Sciences

Parallelization of fluid animate application with Intel CnC

Sergei Krestianskov, Michael Buse, Bo Zhao
The German Research School for Simulation Sciences

JÜLICH FORSCHUNGSZENTRUM
RWTH AACHEN UNIVERSITY

Agenda

- **Problem Definition**
- Approach
- Evaluation
- Conclusion

Team background

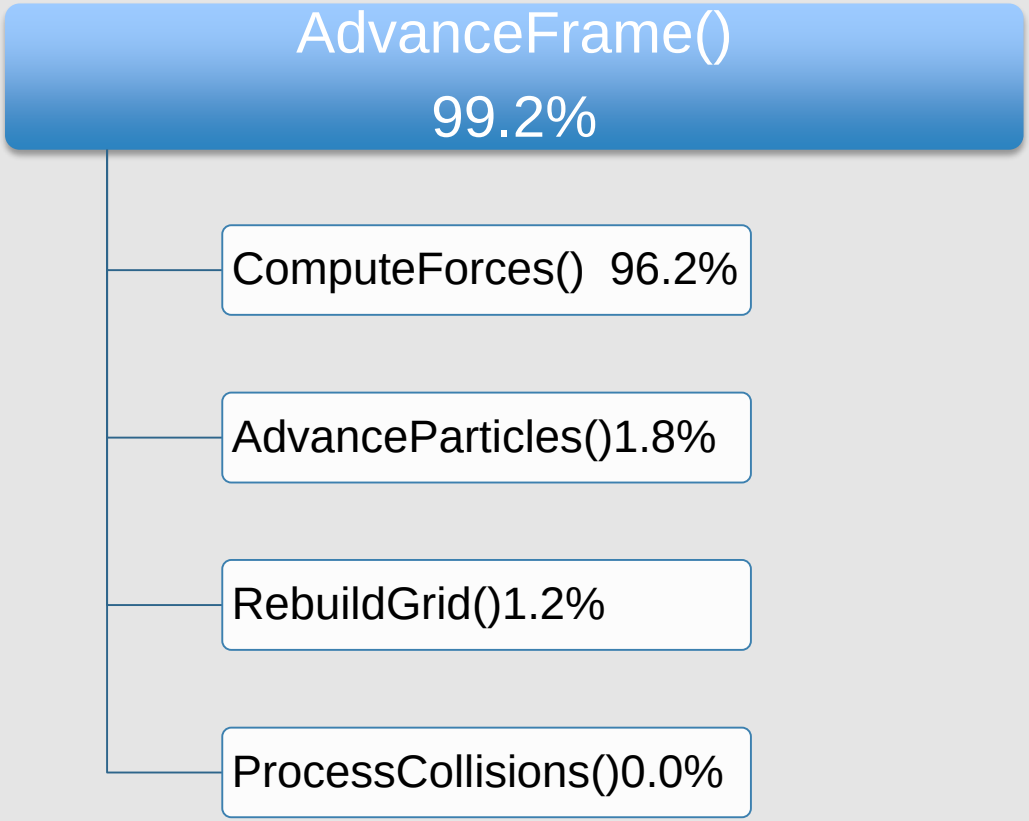
- We are not experts in CnC
- We are not the experts in this domain
- One month time

Problem Definition

- **Fluid Animate**
  - Parallelize with CnC
  - Analyze the performance gains

German Research School for Simulation Sciences

Problem Definition: Hotspots



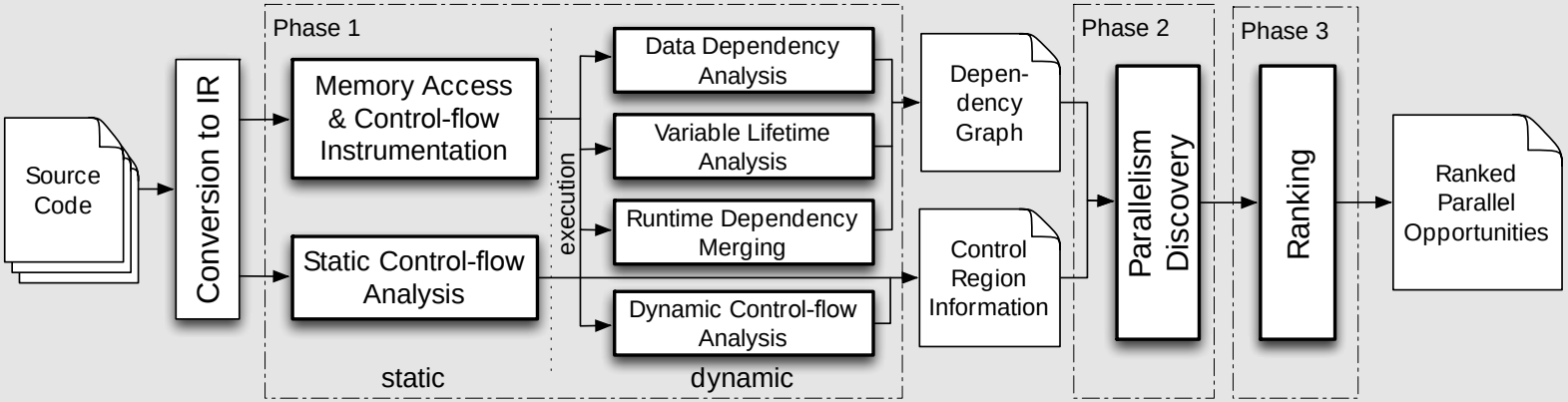
JÜLICH FORSCHUNGSZENTRUM RWTH AACHEN UNIVERSITY

DiscoPoP: Discovery of Potential Parallelism

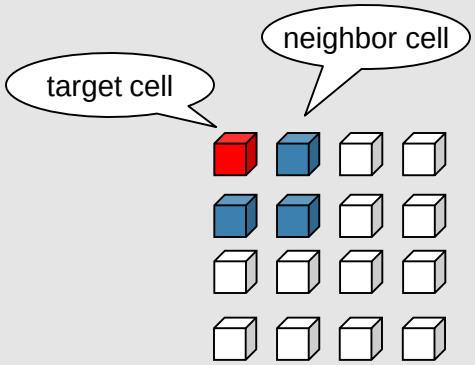
- Target parallelism:
  - DOALL loops
  - Pipeline
  - taskingReveal concrete data dependences that prevent parallelization
- Efficient in time and space

DiscoPoP: Discovery of Potential Parallelism

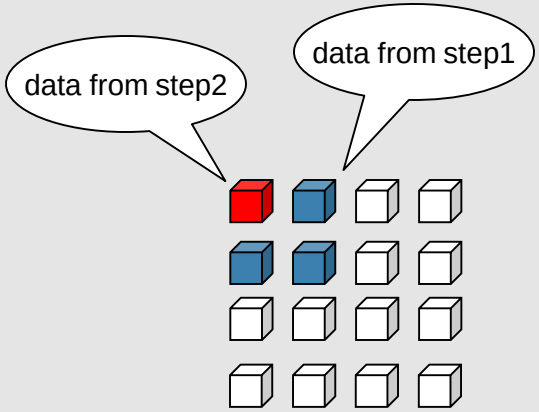
- Architecture



Fluid Animate Dependencies



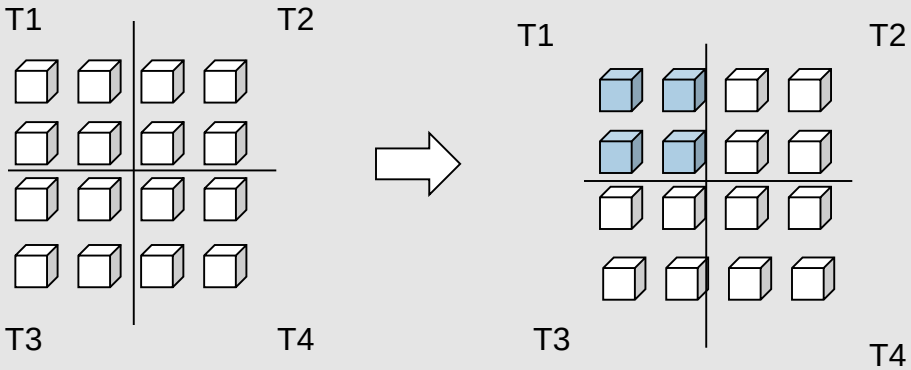
Step 1



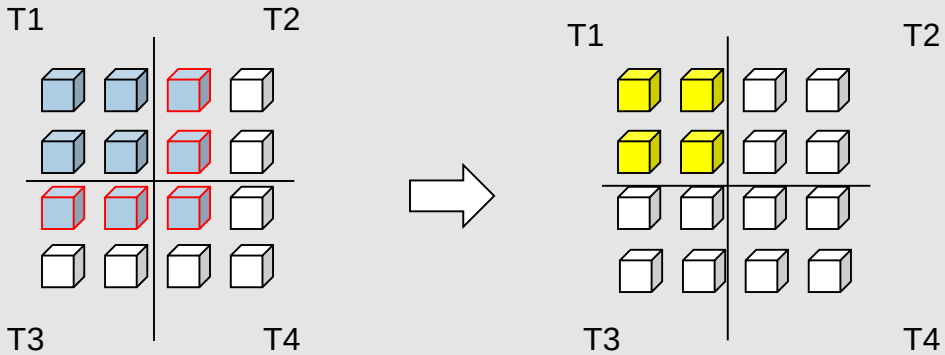
Step 2

Fluid Animate Dependencies

Function 1



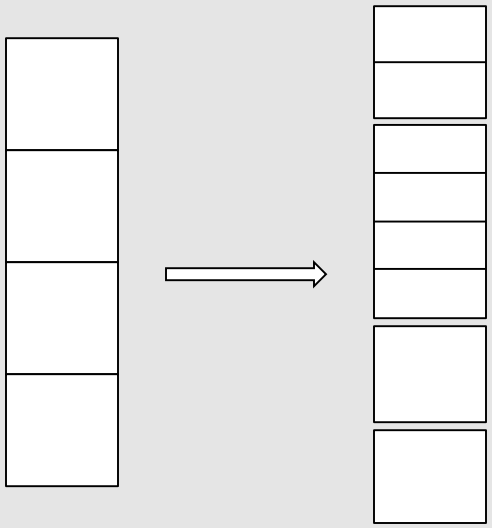
Function 2



Agenda

- Problem Definition
- **Approach**
- Evaluation
- Conclusion

TBB

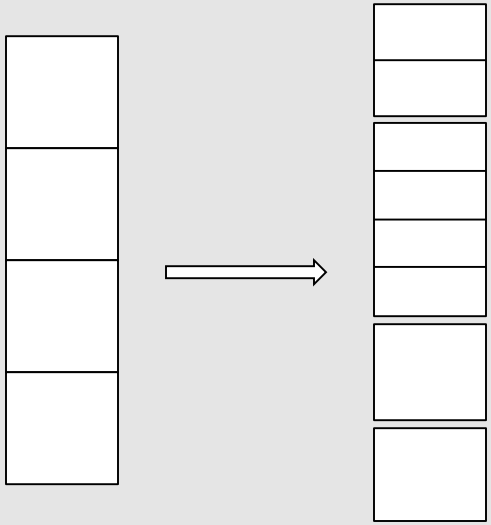


```
struct Grid {
  union {
    struct {
      int sx, sy, sz;
      int ex, ey, ez;
    };
  };
} *grids;
```

In functions:

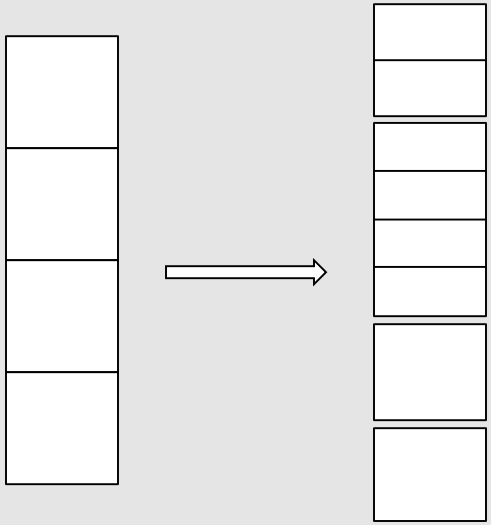
```
    for(int iz = start + block*pid; iz < sz + block; ++iz)
      for(int iy = grids[tid].sy; iy < grids[tid].ey; ++iy)
        for(int ix = grids[tid].sx; ix < grids[tid].ex; ++ix)
        {
          int index = (iz*ny + iy)*nx + ix;
          …
```

TBB



```
if(border[cNeigh]) {
    tbbMutex::scoped_lock lock(mutex[cNeigh][iparNeigh % MUTEXES_CELL]);
    neigh->a[iparNeigh % PARTICLES_PER_CELL] -= acc;
}
else {
    neigh->a[iparNeigh % PARTICLES_PER_CELL] -= acc;
}
```

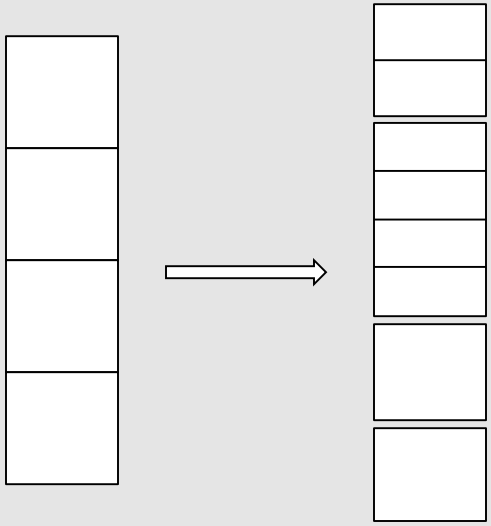
1st CnC Version



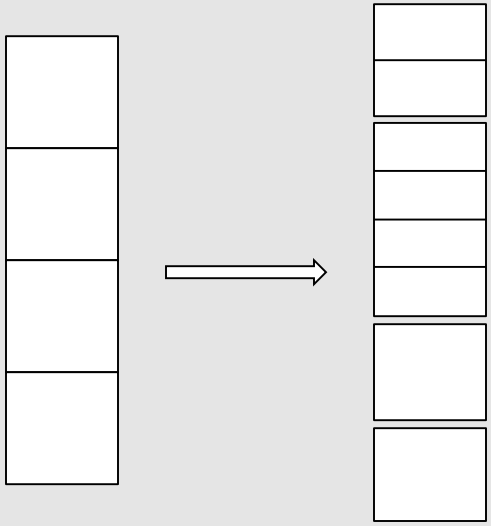
- **4 functions but only 1 hotspot**
  - RebuildGrid
  - ComputeForces (hotspot!)
  - ProcessCollissions
  - AdvanceParticles
- **RebuildGrid and ComputeForces**
  - Can be split into parts, but each part depends on prior part
- **Also have to use TBB locks**

1st CnC Version

```
void ComputeForces()
{
    for(int i = 0; i < num_procs; ++i) ctxt.tagsCF1.put(i);
    ctxt.wait();

    for(int i = 0; i < num_procs; ++i) ctxt.tagsCF2.put(i);
    ctxt.wait();

    for(int i = 0; i < num_procs; ++i) ctxt.tagsCF3.put(i);
    ctxt.wait();

    for(int i = 0; i < num_procs; ++i) ctxt.tagsCF4.put(i);
    ctxt.wait();
}
```



1st CnC Version



In the functions:

```
int sx = (tag%XDIVS)*dx;
int sz = (tag/XDIVS)*dz;
int ex = std::min(sx+dx,nx);
int ez = std::min(sz+dz,nz);
for(int ck = sz; ck < ez; ++ck)
  for(int cj = 0; cj < ny; ++cj)
    for(int ci = sx; ci < ex; ++ci)
    {
      int cindex = (ck*ny + cj)*nx + ci;
      …
```

German Research School for Simulation Sciences

2nd CnC Version

```
struct ComputeForcesCnC { int execute(const int &tag, fa_context &ctxt) const; };
struct fa_context : public CnC::context<fa_context> {
    CnC::step_collection<ComputeForcesCnC>  stepsCF4;
    CnC::tag_collection<int>                tagsCF4;
};
fa_context::fa_context() : CnC::context<fa_context>(), stepsCF4(*this), tagsCF4(*this),
{
    tagsCF4.prescribes(stepsCF4, *this);
}
for(int i = 0; i < num_procs; ++i)
    ctxt.tagsCF4.put(i);
ctxt.wait();

int ComputeForces::execute(const int &tag, fa_context &ctxt) const { … }
```

```
CnC::parallel_for(0, num_progs, 1, ComputeForces(), CnC::pfor_tuner< false >());

int ComputeForces::operator()(int tag) const { … }
```


JÜLICH FORSCHUNGSZENTRUM RWTH AACHEN UNIVERSITY

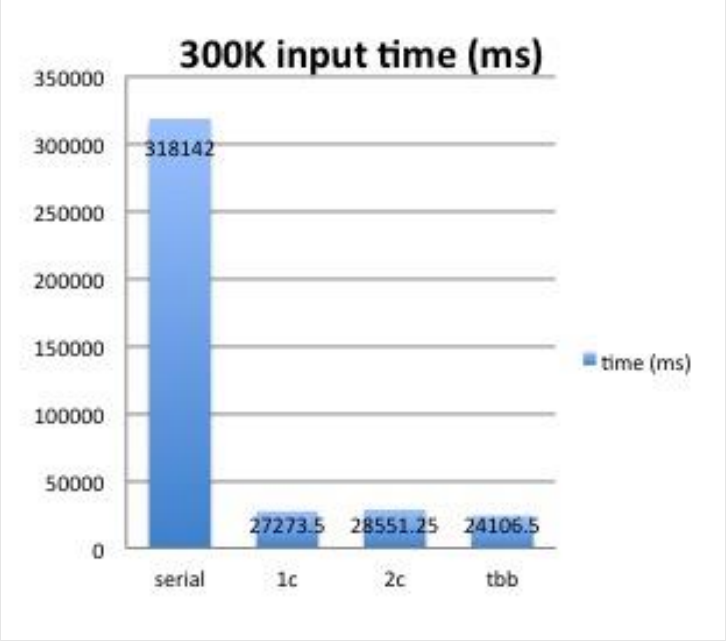
Agenda

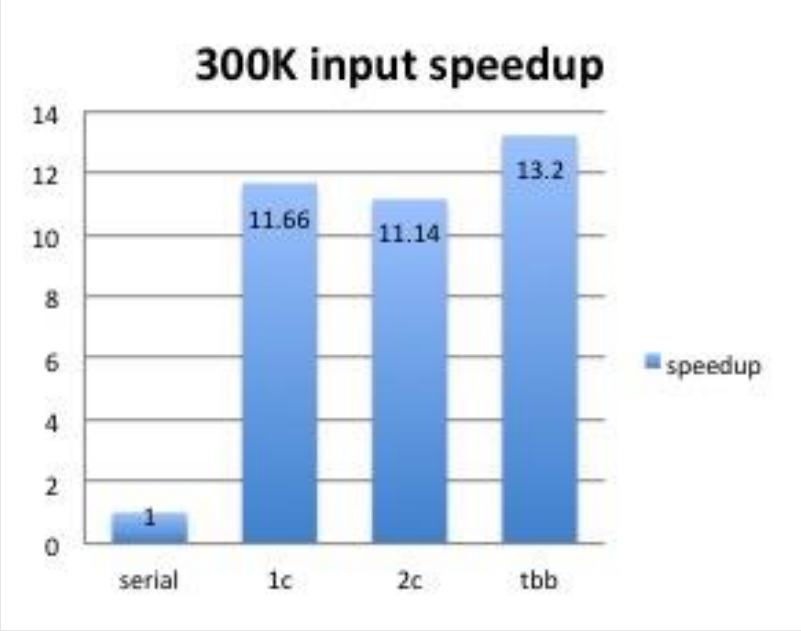
- Problem Definition
- Approach
- **Evaluation**
- Conclusion

Experiment Setup

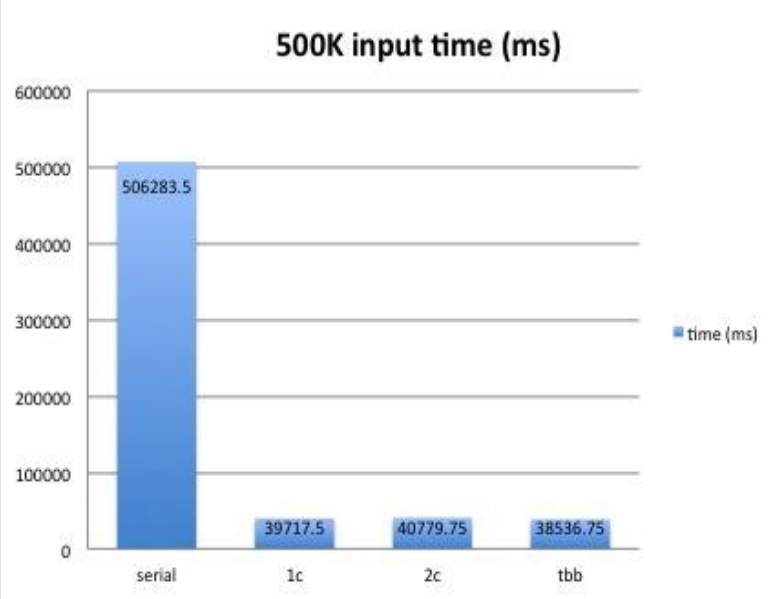
- 2 x 8-core Intel Xeon E5-2650 2 GHz processors with
- 32 GB memory
- Ubuntu 12.04 (64-bit server edition)
- Average of 4 measurements was calculated for each version

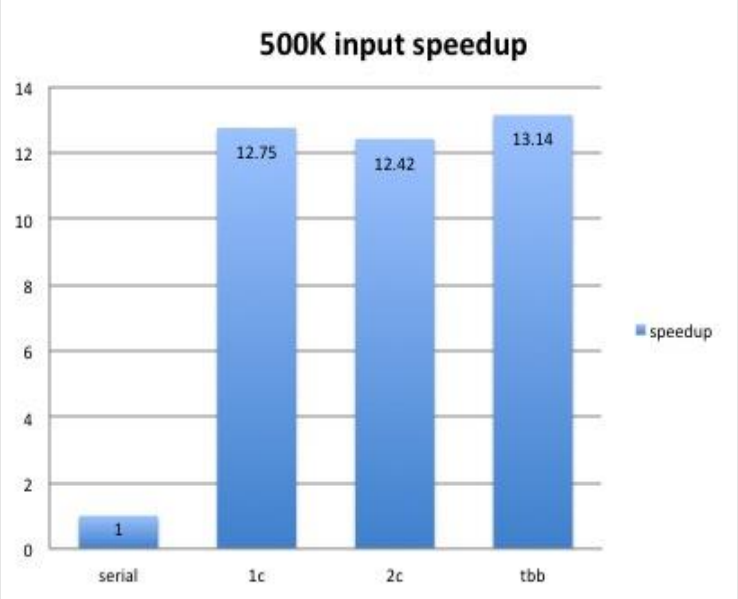
Speedup


300K input time (ms)
350000
300000
250000
200000
150000
100000
50000
0
318142
27273.5
28551.25
24106.5
serial
1c
2c
tbb
time (ms)


300K input speedup
14
12
10
8
6
4
2
0
1
11.66
11.14
13.2
serial
1c
2c
tbb
speedup

Speedup


500K input time (ms)
600000
500000
400000
300000
200000
100000
0
506283.5
39717.5
40779.75
38536.75
serial
1c
2c
tbb
time (ms)


500K input speedup
14
12
10
8
6
4
2
0
1
12.75
12.42
13.14
serial
1c
2c
tbb
speedup

Agenda

- Problem Definition
- Approach
- Evaluation
- **Conclusion**

Conclusions

- We are beginners and had only one month, but still we got a reasobable speedup
    - CnC is good for beginners
- In terms of performance, TBB and CnC showed comparable results