Multiphysics/Hydrodynamical Simulations with Attached and Detached Data

Simulation of Nuclear Networks in SPH-EXA

Joseph Touzet

Joint work with Ruben Cabezon, Osman Seckin Simsek, Sebastian Keller and Florina Ciorba

Unibas DMI

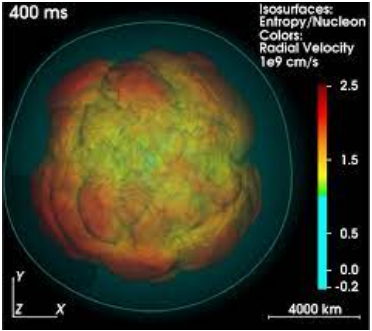
12th of January 2023, SKACH Winter Meeting

Outline

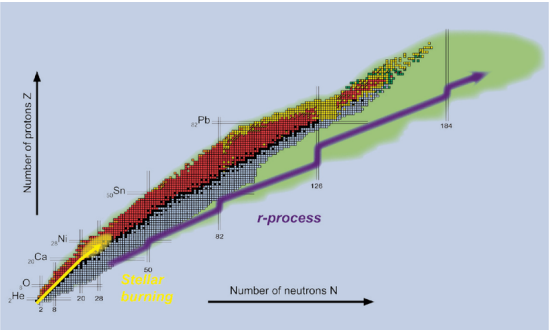
- A motivation for a multiphysics simulation framework: Supernovae simulation
- Nuclear networks
  - net14
  - net86
  - net87 - electrons
- Load balancing proposition: Detached and Attached data
- Preliminary results and future experiments

Supernovae simulation

- Turbulence $\longrightarrow$ Load imbalance
- Time scales: (10^{-3}s to 10^{-7}s for hydrodynamics, 10^{-12}s for nuclear physics).
- simulation scale: from 10^{10} up to 10^{12} particles for up to 1s of integration time.



(a) Turbulences in supernovae.



(b) Nucleosynthesis in supernovae.

Nuclear networks

Description of the reaction $nX + mY \rightarrow lZ$:

$$\begin{cases} \frac{d[X]}{dt} = -r_{nX+mY\rightarrow lZ} \frac{n}{n!m!} [X]^n [Y]^m \\ \frac{d[Y]}{dt} = -r_{nX+mY\rightarrow lZ} \frac{m}{n!m!} [X]^n [Y]^m \\ \frac{d[Z]}{dt} = +r_{nX+mY\rightarrow lZ} \frac{l}{n!m!} [X]^n [Y]^m \end{cases}$$

Energy conservation:

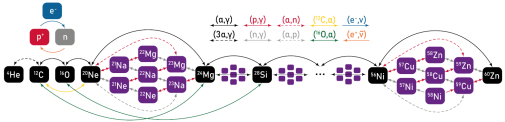
$$-\sum_i \left(BE_i + \frac{\partial U}{\partial Y_i} \right) \frac{dY_i}{dt} + \frac{1}{\rho^2} \frac{d\rho}{dt} \frac{\partial P}{\partial T} T + C_v \frac{dT}{dt} = 0$$

Solving by discretization and linearilization.

Nuclear networks - net14, net86 and net87

- 14, 86 and 87 (86 + electrons) nuclear species respectively.
- 16, 157 and 157 fusion reactions respectively.
- 13, 153 and 153 fission reactions respectively.
- Net87: 1 electronic capture reaction and 1 positron capture reaction.
- $O(n^2)$ time complexity $\Rightarrow$ net86/87 $\approx$36x slower than net14.

(a) Reactions simulated by net14.

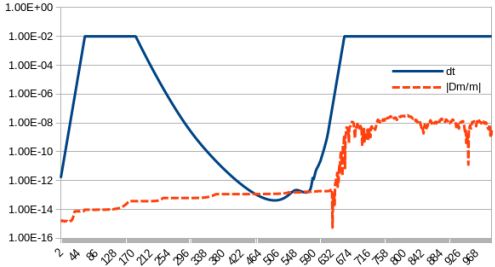


(b) Reactions simulated by net86/87.

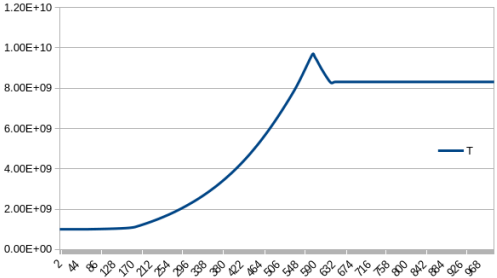
Nuclear networks - Numerical integration

- Iterative solver, solving the linearized system of equations.
- Sub-stepping over the hydro timetep.
  - Timestep variation $\Rightarrow$ Targeting a relative temperature variation $\frac{\Delta T}{T} \approx \tau_{target}$.

↳ Load inbalance due to a varying number of iteration and substep per particle.



(a) Timestep evolution for ^{12}C+^{16}O fusion.



(b) Temperature evolution for ^{12}C+^{16}O fusion.

Nuclear networks - from Attached to Detached data

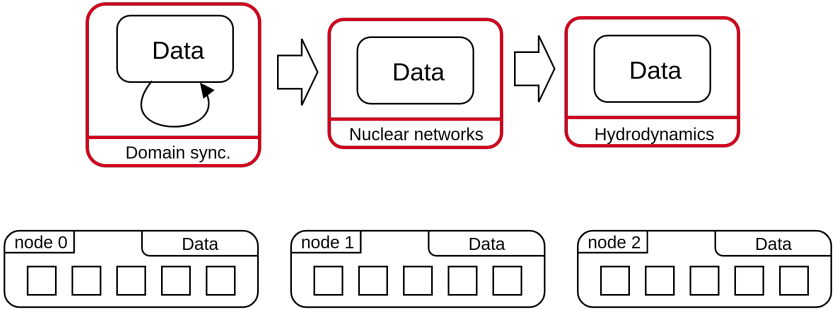


Figure: Simulation procedure and data container for attached data.

Nuclear networks - from Attached to Detached data

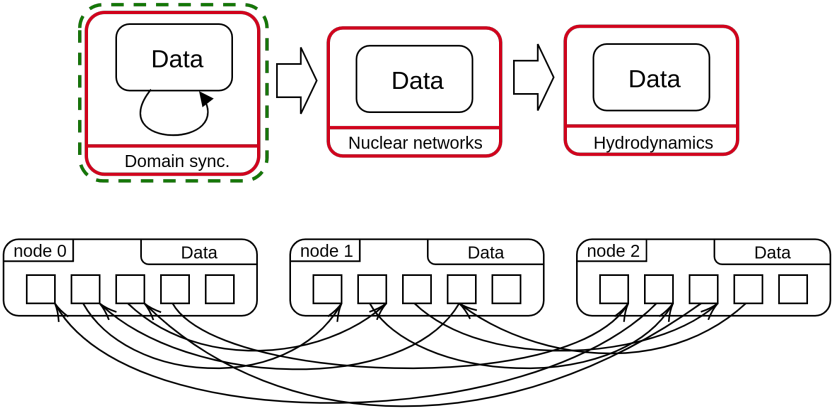
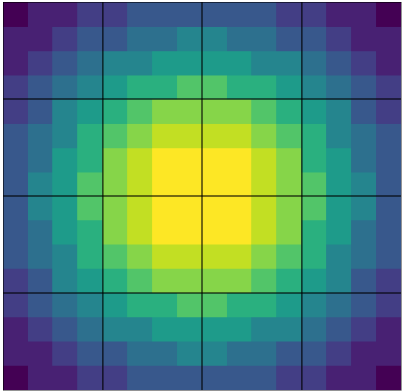


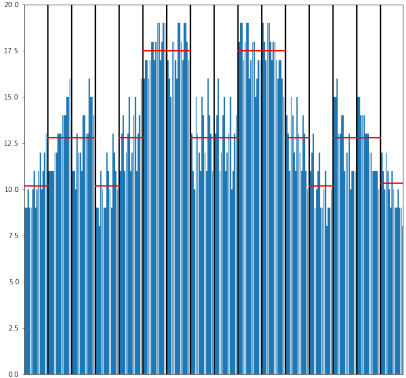
Figure: Simulation procedure and data container for attached data: Domain sync.

Nuclear networks - from Attached to Detached data

Example of (imbalanced) spatial load partition:



(a) Spatial number of iterations (load) repartition repartition.



(b) Node number of iterations (load) repartition (per node average in red).

Nuclear networks - from Attached to Detached data

- Static Random Distribution of Detached Data - SRD^3
- ↳ Random load balancing from random data (and thus load) mapping
- Communication impacts:
  - Collective for a few hydro properties (5 floats: ρ, T, c_v, c, P).
  - Avoid domain synchronization (and reordering) of abundances (14, 86 or 87 floats).

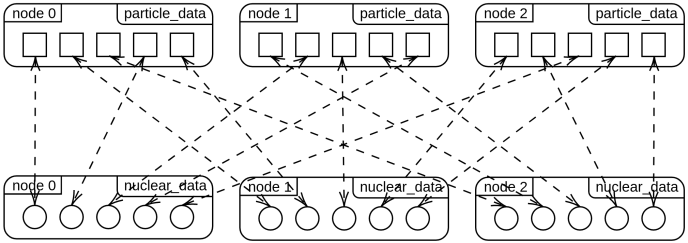


Figure: Example of a Static Random Distribution of Detached Data (SRD^3).

Nuclear networks - from Attached to Detached data

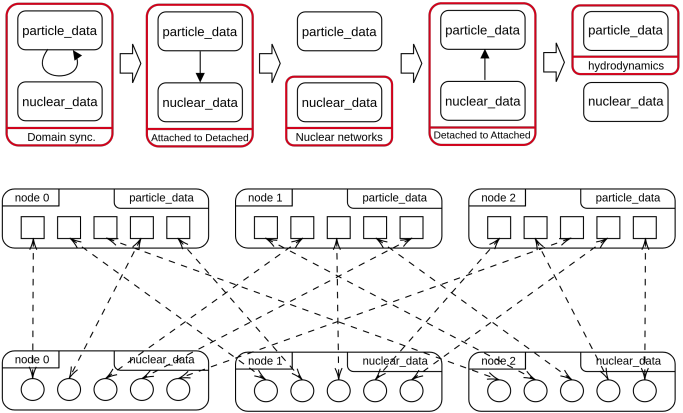


Figure: Simulation procedure and data containers for detached data.

Nuclear networks - from Attached to Detached data

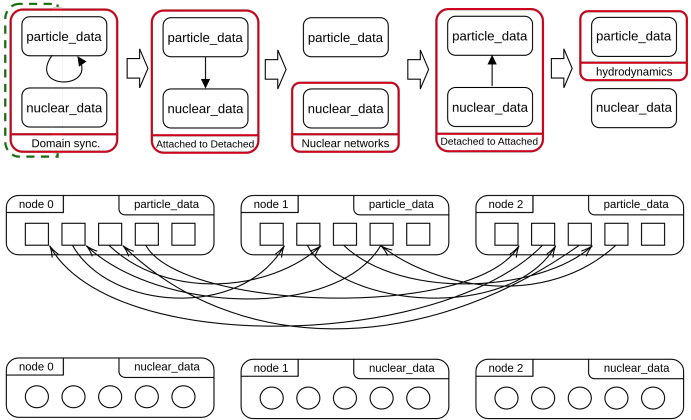


Figure: Simulation procedure and data containers for detached data: domain sync.

Nuclear networks - from Attached to Detached data

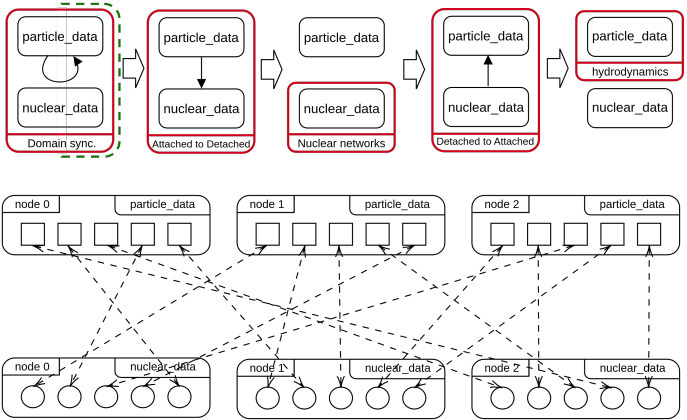


Figure: Simulation procedure and data containers for detached data: partition re-decomposition

Nuclear networks - from Attached to Detached data

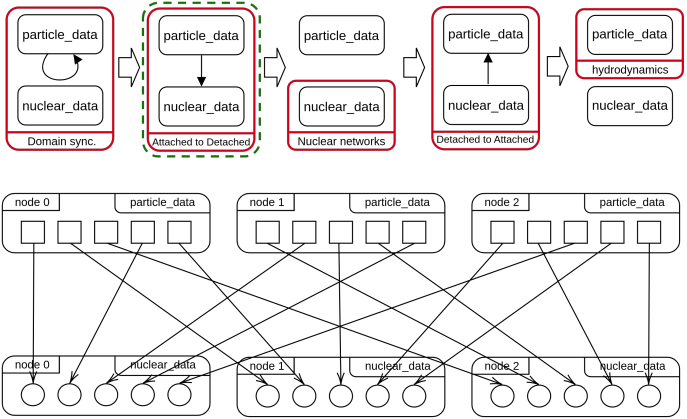


Figure: Simulation procedure and data containers for detached data: Attached to Detached sync.

Nuclear networks - from Attached to Detached data

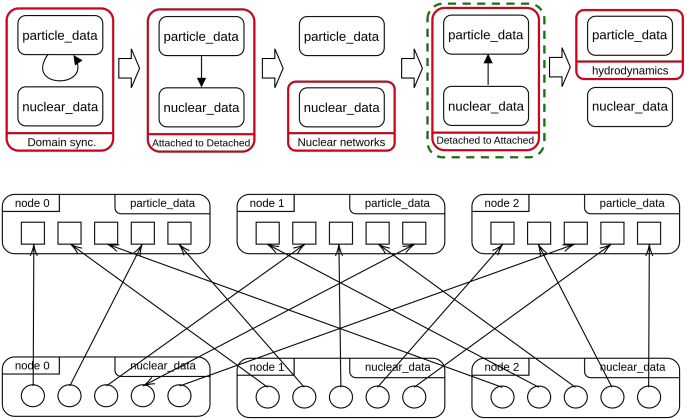
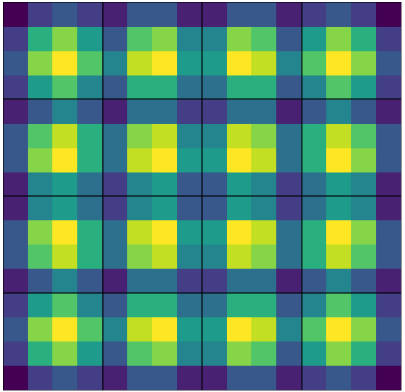
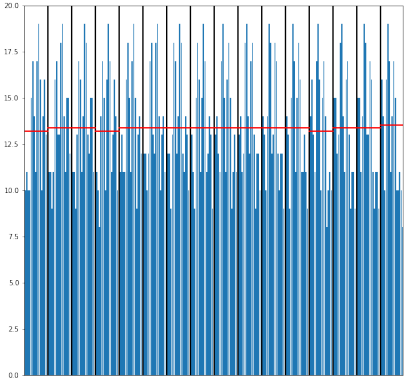


Figure: Simulation procedure and data containers for detached data: Detached to Attached sync.

Nuclear networks - from Attached to Detached data



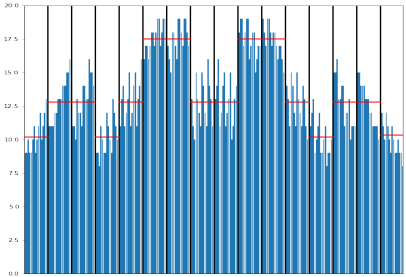
(a) Node number of iterations (load) repartition (2D) with detached data.



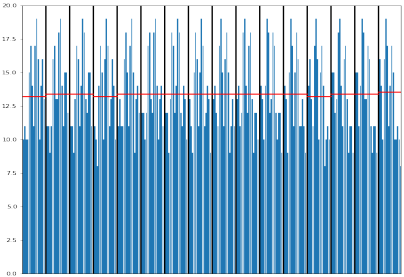
(b) Node number of iterations (load) repartition with detached data (per node average in red).

Nuclear networks - from Attached to Detached data

type	max. load	c.o.v.	$\frac{max-avg}{avg}$	$\frac{max-avg}{max}$	comments
spatial load	17.5	20%	31%	24%	
detached data	14	0.6%	1.4%	1.4%	20% speedup



(a) Attached data (spatial load).



(b) Detached data.

Nuclear networks - Impact of Detached data

- Tested for 3 nodes (6 MPI ranks) on Scicore.
- ”Attached data” simulated by *initializing* nuclear particle on the same node as their hydro counterpart.
- 25 iteration for a constant 4.10^{-7}s timestep.
- $^{12}C + {}^{16}O$ fusion, Gaussian heat distribution (sedov test case).

metric	attached data	detached data	comments
runtime	398s	353s	13% speedup
c.o.v.	0.3-24%	0.5-3.3%	
$\frac{max-avg}{avg}$	0.5-26%	0.8-6%	
$\frac{max-avg}{max}$	0.4-33%	0.8-6.4%	

Preliminary results and future experiments

- Performance results:
  - Tested for 5 millions (net14) particles per A100 (40GB) GPU $\longrightarrow$ $\sim$50 million particles per LUMI-G node.
  - 1x-5x speedup for GPU (A100) vs CPU (intel skylake xeon, AMD zen2/zen3).
  - 13% (and more ?) speedup due to load balancing.
- Future experiments:
  - Supernovae simulation.
  - From 10^{10} to 10^{12} particles simulation.
  - Simulations on LUMI-G.

Appendix 1 - GPU implementation details

- c++-17 (header only, except CUDA/HIP part)
- CPU parallelism with OPENMP
- Multi-node support with MPI
- GPU parallelism with CUDA/HIP, avoids code duplication.
- CUDA work sharing across (shared memory) thread block.

```
    template<typename Float>
    HOST_DEVICE_FUN Float inline ggt1(const Float x)
```

Figure: CUDA/host function decorator.

```
    /// constant atomic number values
    DEVICE_DEFINE(inline static const std::array<double COMMA 14>, Z, = {
        2 COMMA 6 COMMA 8 COMMA 10 COMMA 12 COMMA 14 COMMA 16 COMMA 18 COMMA 20 COMMA 22 COMMA 24 COMMA 26 COMMA 28 COMMA 30
    };)

    /// constant number of masses values
    DEVICE_DEFINE(inline static const std::array<double COMMA 14>, A, = {
        4 COMMA 12 COMMA 16 COMMA 20 COMMA 24 COMMA 28 COMMA 32 COMMA 36 COMMA 40 COMMA 44 COMMA 48 COMMA 52 COMMA 56 COMMA 60
    };)
```

Figure: Defining a CUDA/host hybrid constant.

```
                    const double gamp = gam*std::pow(constants::DEVICE_ACCESS(Z)[i], 5./3.);
```

Figure: Accessing a CUDA/host hybrid constant.

```
#define COMMA ,

#if defined(__CUDACC__) || defined(__HIPCC__)
    #define COMPILE_DEVICE

    #define HOST_DEVICE_FUN __host__ __device__

    #define DEVICE_DEFINE(type, symbol, definition) \
                    type        symbol definition    \
        __device__ type dev_##symbol definition
#else
    #define HOST_DEVICE_FUN

    #define DEVICE_DEFINE(type, symbol, definition) type symbol definition
#endif


#if defined(__CUDA_ARCH__) || defined(__HIP_DEVICE_COMPILE__)
    #define DEVICE_CODE

    #define DEVICE_ACCESS(symbol) dev_##symbol
#else
    #define DEVICE_ACCESS(symbol) symbol
```

Figure: CUDA prepossessing macros.

Appendix 2 - Numerical integration

- sub-stepping: iterative time-stepping to integrate over the hydro timestep.
- Iterative method to solve the non-linear system from its lineralized approximation.

Algorithm 1: RÉSEAU NUCLÉAIRE

```
t ← 0
while t < dt_hydro do
    Y_next = Y
    T_next = T
    for i ← 0 to N_iteration,max do
        T_theta ← (1 − θ)T + θT_next
        Y_theta ← (1 − θ)Y + θY_next

        [r, dr/dT, BE] ← COMPUTE-RATES(Y_theta, T_theta, ρ)
        [M, RHS] ← COMPUTE-SYSTEM(Y_theta, T_theta, ρ, r, dr/dT, BE)
        [dY/dt, dT/dt] ← SOLVE-SYSTEM(M, RHS)

        Y_theta ← Y + dt(dY/dt)
        T_theta ← T + dt(dT/dt)

        if CONVERGED() then
            break
    if not FAILED() then
        Y = Y_next
        T = T_next
        t ← t + dt_nuclear
    dt_nuclear ← UPDATE-TIMESTEP(dt_nuclear, dT)
```

Appendix 3 - Parallel computation

- Parallel computation over every particles.
- Taking advantage of massive parallelism GPU.
- Load balancing between node.

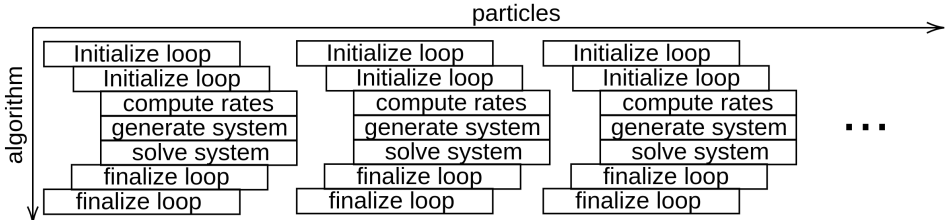


Figure: Demonstration of parallel computation.