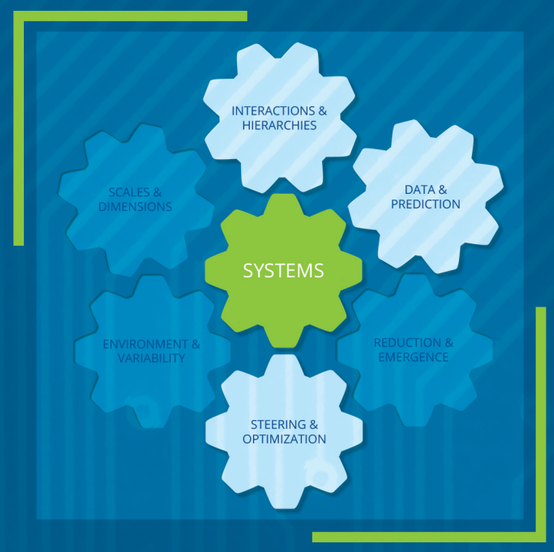
Bernhard Manfred Gruber

bernhardmgruber@gmail.com

Professional support under extreme conditions

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING



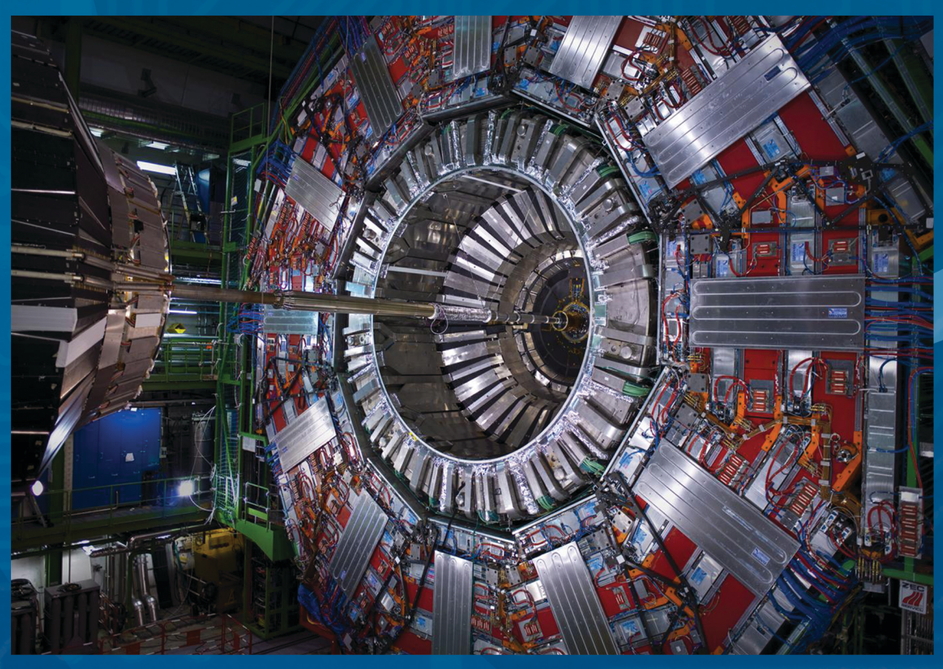
LLAMA: COMPILE TIME AUTOMATIC MEMORY LAYOUT OPTIMIZATION. WRITE CODE ONCE AND PERFORM WELL ON MANY SYSTEMS.

GOAL

Current and future hardware architectures are heterogeneous. Performance portability with a single code base is an increasingly relevant challenge.
Performance portable parallelism, to exhaust multi-, manycore and GPU systems, is largely addressed, e.g. Alpaka or Kokkos.
Efficient use of a system's memory and cache hierarchies is equally crucial. First attempts like AoS/SoA containers and std::mdspan exist, but general solutions do not exist yet.

We propose a novel software library called LLAMA to address this gap. LLAMA will:

- Build infrastructure for memory layout transformation
- Provide means for automatic memory layout optimizations for various hardware architectures.



CERN CMS DETECTOR (1)

CHALLENGES

- Programmers use incredibly diverse data structures.
- No user code changes when switching memory layout or target hardware.
- [Automatically] find the optimal layout for a target hardware, resulting in fast, efficient code.
- User facing API should feel like a native data structure.
- Aim for a standard C++ library solution.

```cpp
struct Particle {
  struct Pos {
    float x;
    float y;
    float z;
  } pos;
  float mass;
};

auto particles =
  std::vector<Particle>(N);
for (auto& p : particles) {
  p.pos.x = ...;
  p.pos.y = ...;
  p.pos.z = ...;
  p.mass  = ...;
}
```

NAIVE C++ VERSION

```cpp
struct Pos{};
struct X{};
struct Y{};
struct Z{};
struct Mass{};

using Particle = llama::DS<
  llama::DE<Pos, llama::DS<
    llama::DE<X, float>,
    llama::DE<Y, float>,
    llama::DE<Z, float>
  >>,
  llama::DE<Mass, float>
>;

auto particles = llama::allocView(
  llama::Mapping<Particle>(N));
for (int i = 0; i < N; ++i) {
  auto p = particles(i);
  p(Pos{}, X{}) = ...;
  p(Pos{}, Y{}) = ...;
  p(Pos{}, Z{}) = ...;
  p(Mass{})     = ...;
}
```

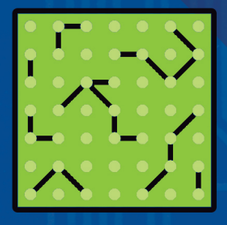
LLAMA VERSION

METHOD

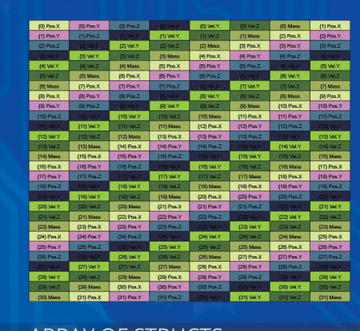
Based on some theory around index spaces and linear mappings we build a C++ template meta programming library that optimizes a memory layout at compile time. The implementation will be verified in PIConGPU and ROOT and tested on heterogeneous HPC systems and conventional desktop workstations.

SIMULATIONS

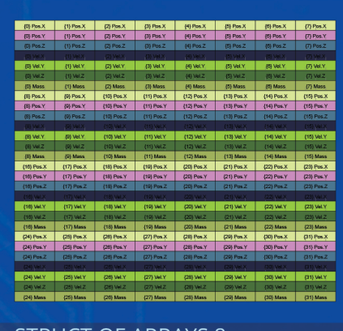
HIGH-PERFORMANCE COMPUTING (HPC)



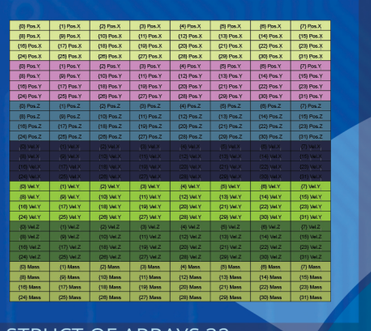
BIG DATA



ARRAY OF STRUCTS



STRUCT OF ARRAYS 8



STRUCT OF ARRAYS 32

PERSPECTIVE

We expect a prototypic C++17 software library that scientists can gradually adopt into their existing code bases achieving cross platform and cross architecture performance portability on the use of memory. We expect great synergy with performance portable parallelization frameworks such as Alpaka and Kokkos.

CROSS-DISCIPLINARY

We aim at providing fundamental software infrastructure technology without bias for a specific domain. Any computationally and memory intensive application should benefit.

REFERENCES: ALPAKA: GITHUB.COM/ALPAKA-GROUP/ALPAKA; KOKKOS: GITHUB.COM/KOKKOS/KOKKOS; STD::MDSPAN: WG21.LINK/P0009; PICONGPU: GITHUB.COM/COMPUTATIONALRADIATIONPHYSICS/PICONGPU; ROOT: ROOT.CERN