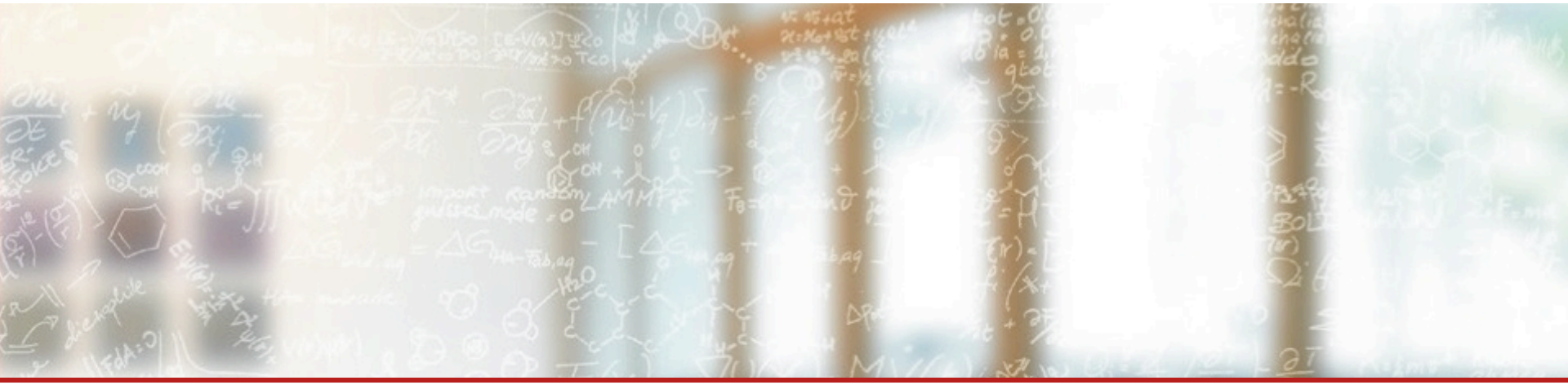
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



# Longing for Portability of Performance

Mauro Bianco

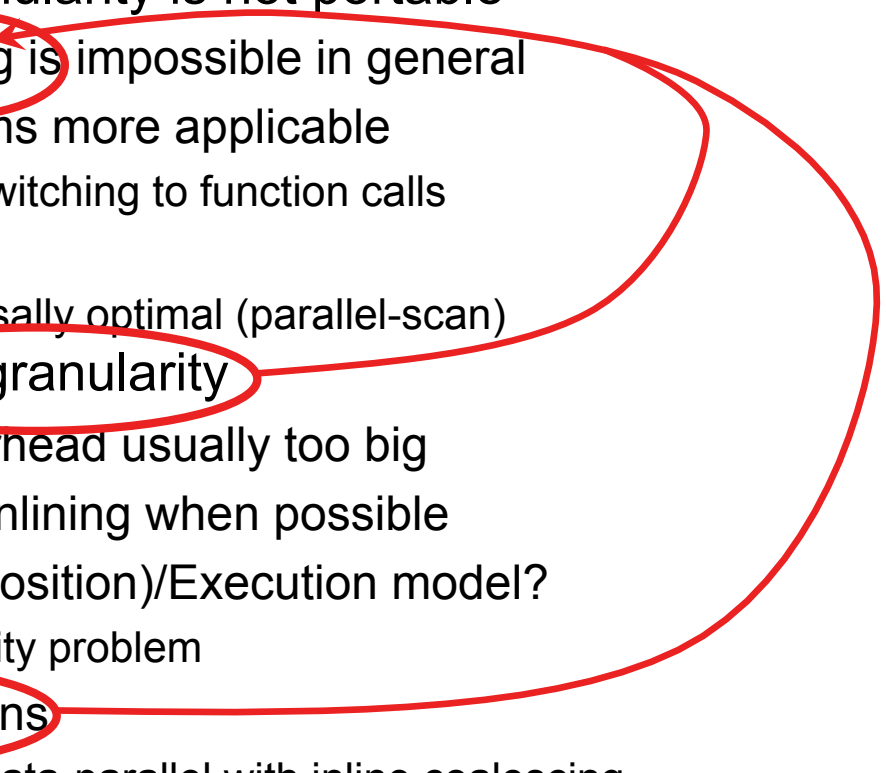
HiHAT: Hierarchy, Part II
February 20th, 2018

# Performance Portability in Weather/Climate Codes

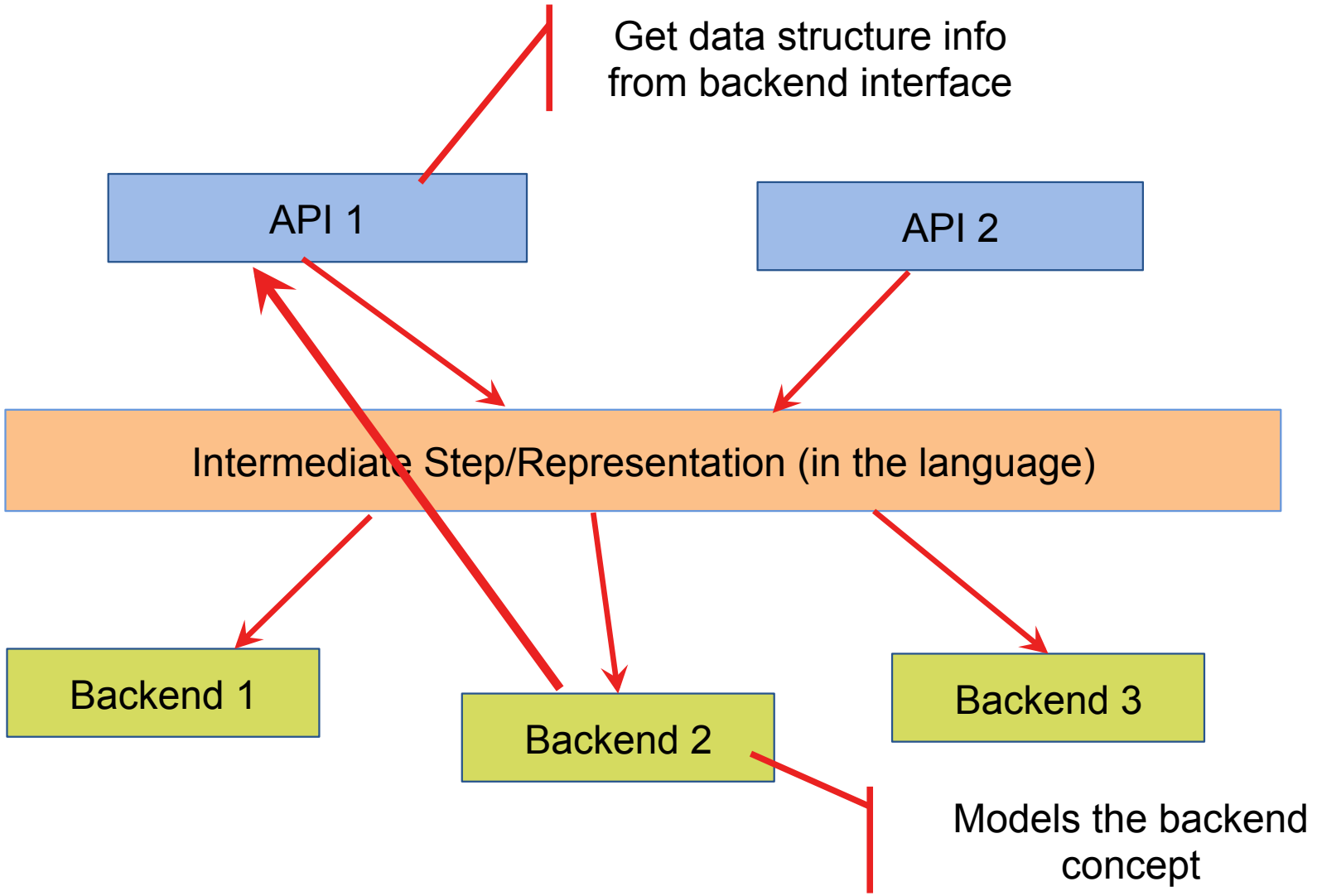
- Single source code
  - Defined as the code a programmer needs to maintain
- Stencil computations with complex dependencies
  - Tens-hundreds stencils / time step
  - Conditional execution
  - Limited number of halo-lines
    - Communication is needed
  - Stencils are reasonably big tasks
    - But maybe not filling up a GPU, for instance
    - Stencil scheduling can help
  - Stencils are uniform data-parallel computations
    - They will consume more that a single thread

# Data Parallel and Task Oriented Programming

- From a developer PoV
- The context switch problem
  - User managed granularity is not portable
    - Automatic splitting is impossible in general
    - Aggregation seems more applicable
      - From thread switching to function calls
      - Inlining
      - Still not universally optimal (parallel-scan)
  - Express the finest granularity
    - Function call overhead usually too big
    - Coarsening with inlining when possible
    - Inspector/(Transposition)/Execution model?
      - Still a granularity problem
    - Algorithmic patterns
      - This become data-parallel with inline coalescing



# Our Approach: Generic Library in C++

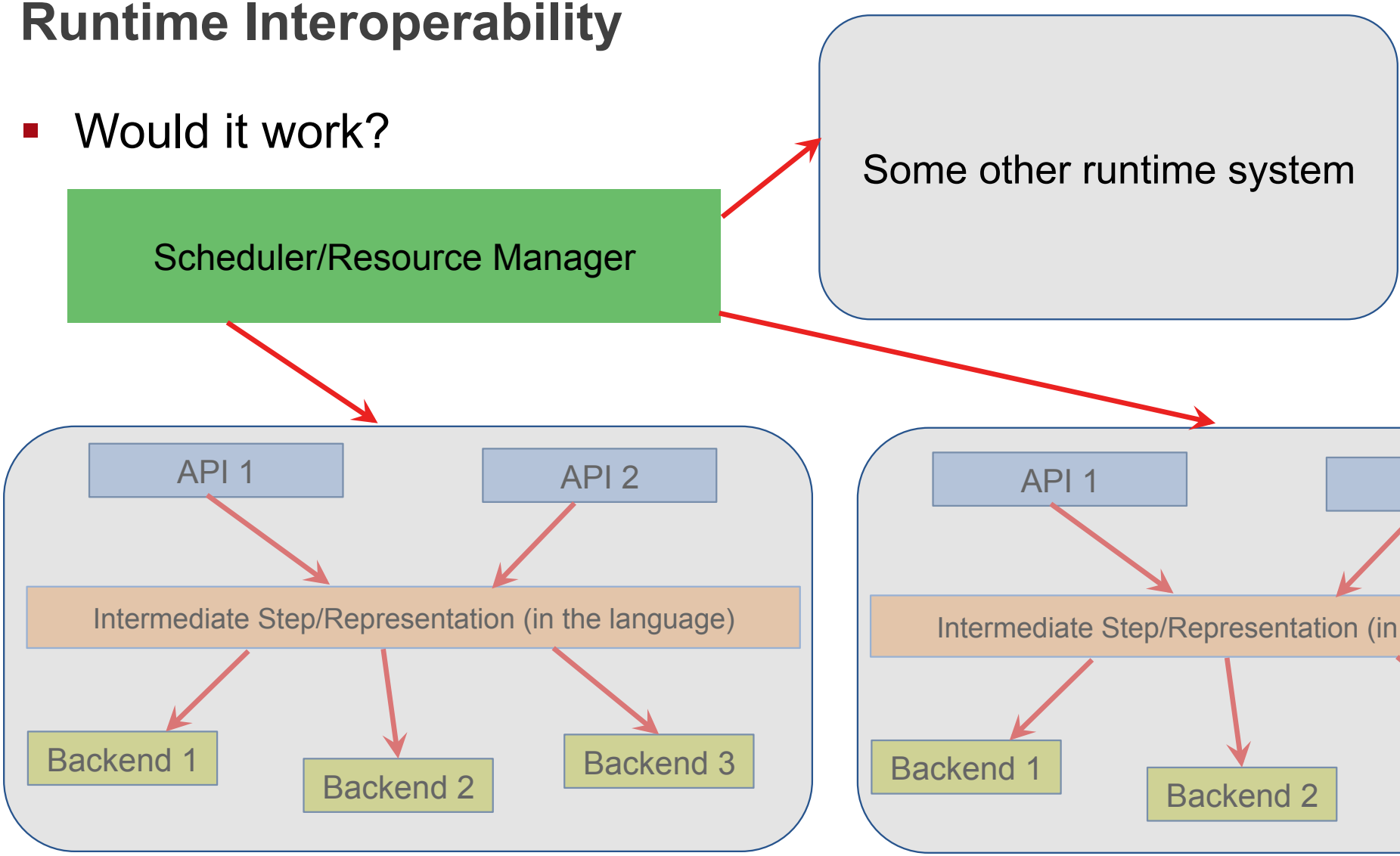


# Non-Uniform Applications in HPC

- Meaning: Many computational patterns
  - Different platforms optimize different patterns differently
- HPC tend to favor uniform applications
  - The simulations are steps in a workflow managed by the user
- There are good reasons for that
  - Layout transformations needed?
  - Does the implementation depends on the input size?
- Even with uniform applications there are problems
  - Different penalties for different operations
    - Branching, floating-point, …
- The fewer the computational patterns the better
  - Higher chances of being performance-portable

# Runtime Interoperability

- Would it work?

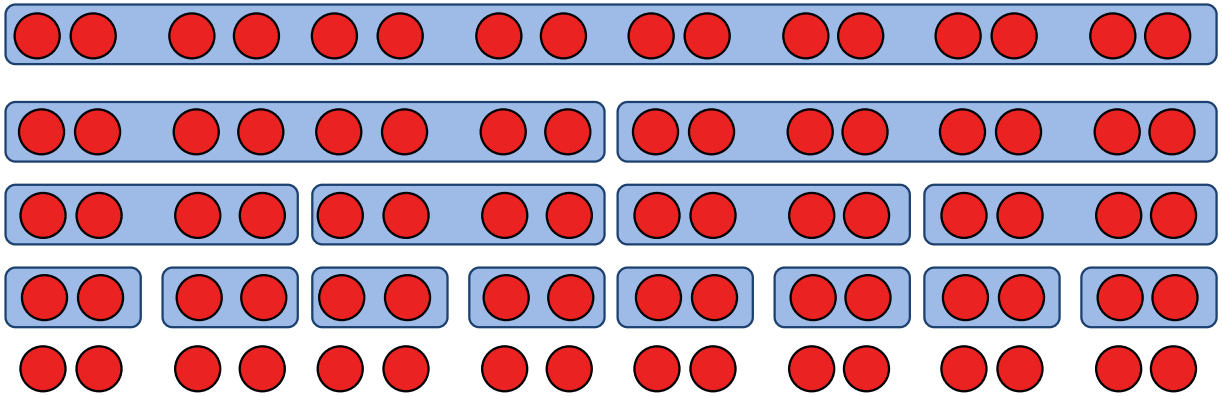


# Solutions?

- Holistic approach
  - All backends needs a common runtime/semantics interfaces
    - HiHAT
  - Need to make runtime systems interoperable universally
    - E.g., OpenMP and TBB
- C++ Executors
  - Through HiHAT interfaces?
  - This is still kind of holistic
- Right now a backend does whatever
  - Ideally re-implement backends to lay on top of common interfaces

# Disputing Universality

- Is there a solution that works optimally for all cases?
- Network-oblivious algorithms provide a negative result
  - Transposition cannot be implemented optimally in D-BSP



- Can generalize to hierarchical asynchronous tasking?
  - Not easy to do mathematically but…
  - Latency is getting worse