Unstructured Hydrodynamics on Spatial Dataflow Architectures

Democratizing AI Accelerators for HPC Applications:
Challenges, Success, and Support

SC24 Atlanta, GA | hpc creates.

CASC Center for Applied Scientific Computing

11/20/2024

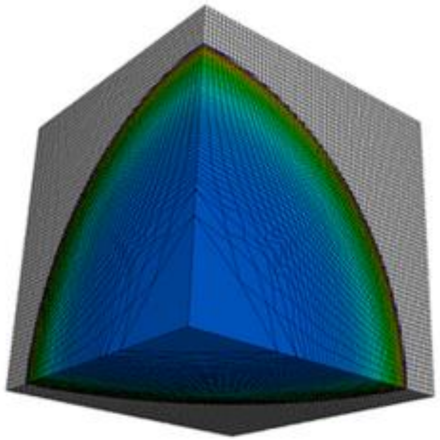
Piotr Luczynski, Tal Ben-Nun, **Brian Van Essen**
Computer Scientist

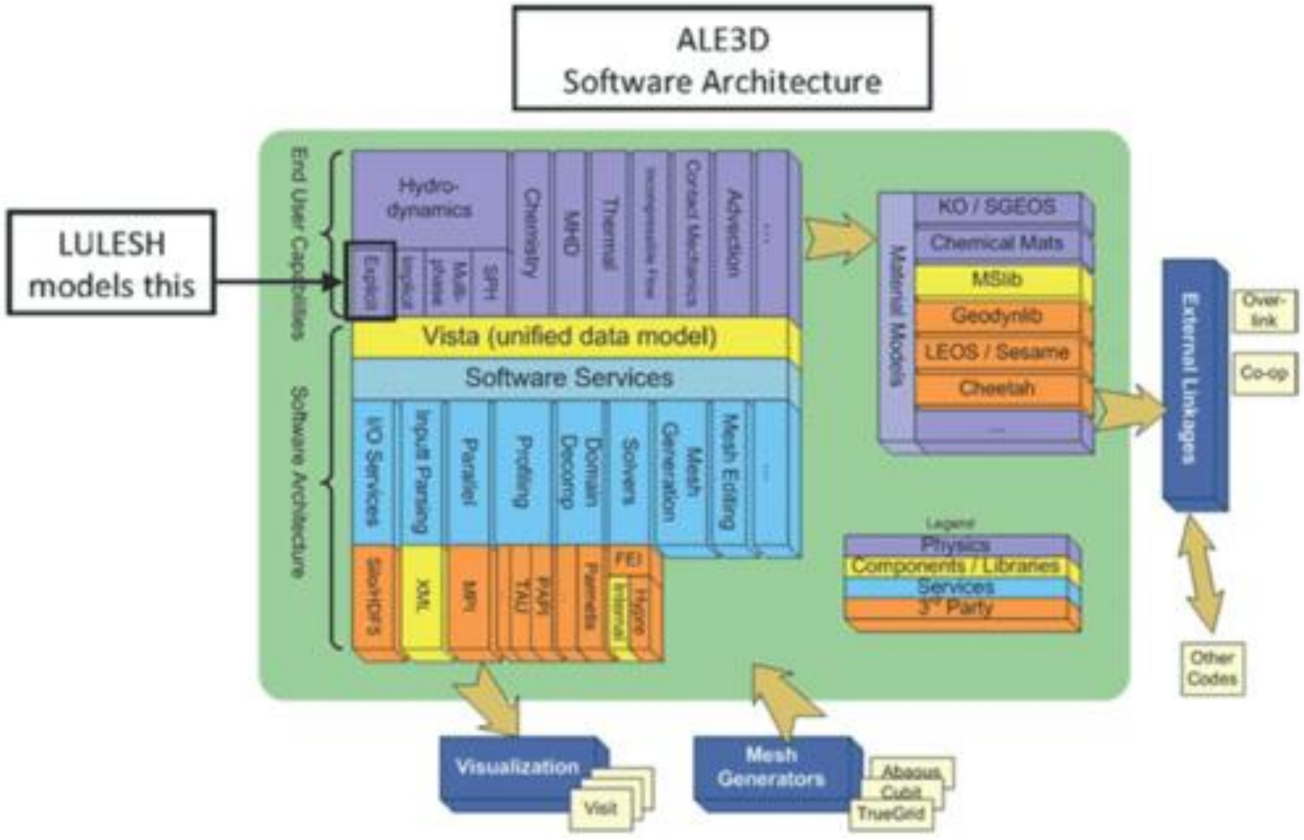
LLNL-PRES-778998
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

AI Spatial Dataflow accelerators offer an interesting opportunity for traditional Scientific Modeling and Simulation Code

- No agreed upon abstraction
- No, PyTorch will not work and isn't a DSL
- Demonstrate mapping scientific kernel from LULESH to Cerebras CS-2



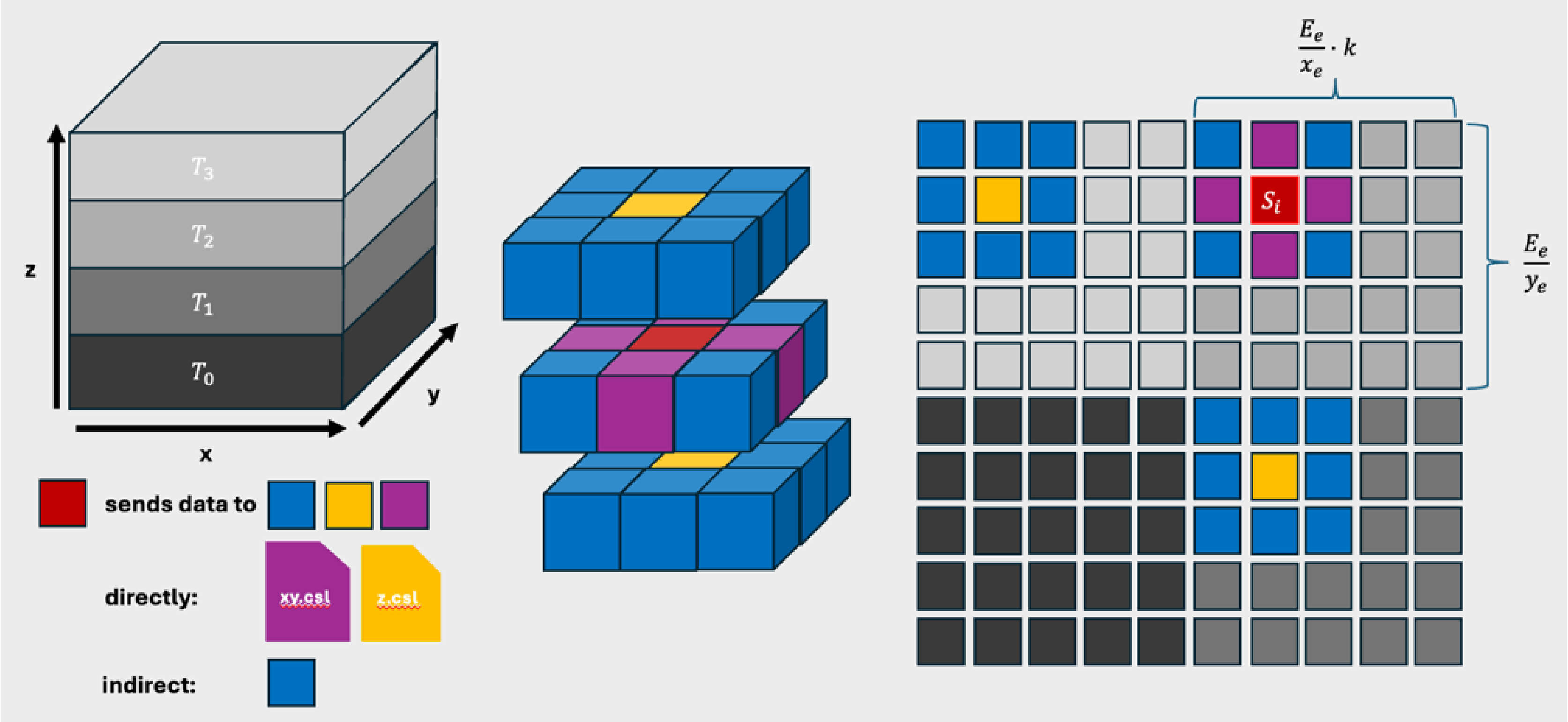


LULESH represents a typical hydrocode, like ALE3D. LULESH approximates the hydrodynamics equations discretely by partitioning the spatial problem domain into a collection of volumetric elements defined by a mesh.

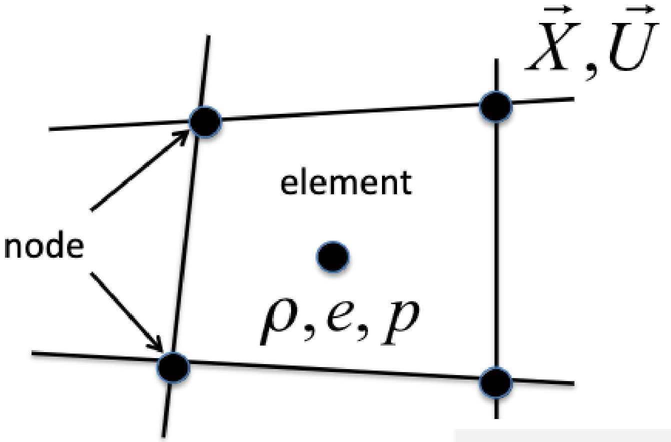
https://asc.llnl.gov/codes/proxy-apps/lulesh

Spatial mapping 3D problem onto 2D plane is challenging

- 2-D neighbors are "easy" to map onto a spatial dataflow architecture
- Mapping a 3rd dimension can either use compute-local memory, or space
- Complexity of LULESH code exceeds local configuration memory – we have to use a spatial tiling



Approximate the LULESH program code to a benchmarkable kernel



Given:

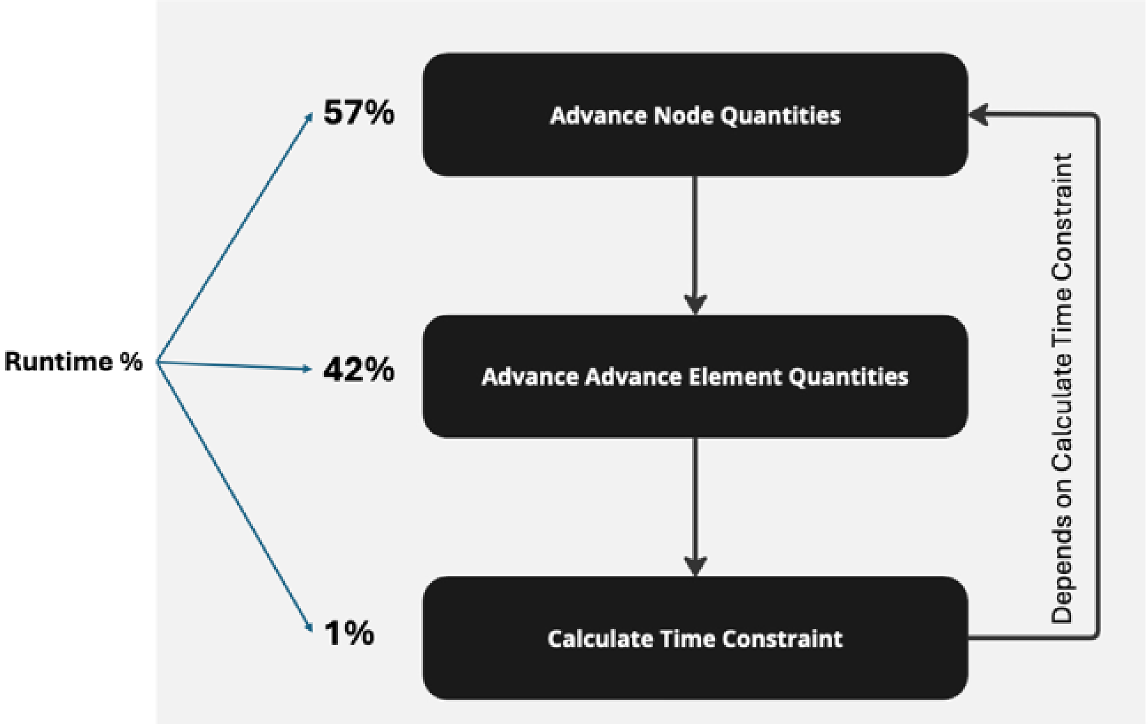
x, y, z, xd, yd, zd: [numnode]f32

deltatime: f32

How to implement:

```
x += xd * deltatime
y += yd * deltatime
z += zd * deltatime
```

Optimize for runtime and memory



https://doi.org/10.2172/1117905

Hand-written performance tuning requires expertise and time

Implementation	Time (cycles)	Instruction Size (Bytes)
Optimal	$216 \cdot 3 = 648$	N/A
Loop	15779	100
Map	4562	96
DSD	1314	48

```
fn fmacs_map(a: f32, b: f32, c: f32) f32 { return a * b + c; }
@map(fmacs_map, domain.xd_dsd, domain.deltatime, domain.x_dsd, domain.x_dsd);
@map(fmacs_map, domain.yd_dsd, domain.deltatime, domain.y_dsd, domain.y_dsd);
@map(fmacs_map, domain.zd_dsd, domain.deltatime, domain.z_dsd, domain.z_dsd);
```

```
@fmacs(domain.x_dsd, domain.x_dsd, domain.xd_dsd, domain.deltatime);
@fmacs(domain.y_dsd, domain.y_dsd, domain.yd_dsd, domain.deltatime);
@fmacs(domain.z_dsd, domain.z_dsd, domain.zd_dsd, domain.deltatime);
```

Stateful Dataflow Multigraphs (SDFG): A Data-Centric IR

Directed graph of multigraphs

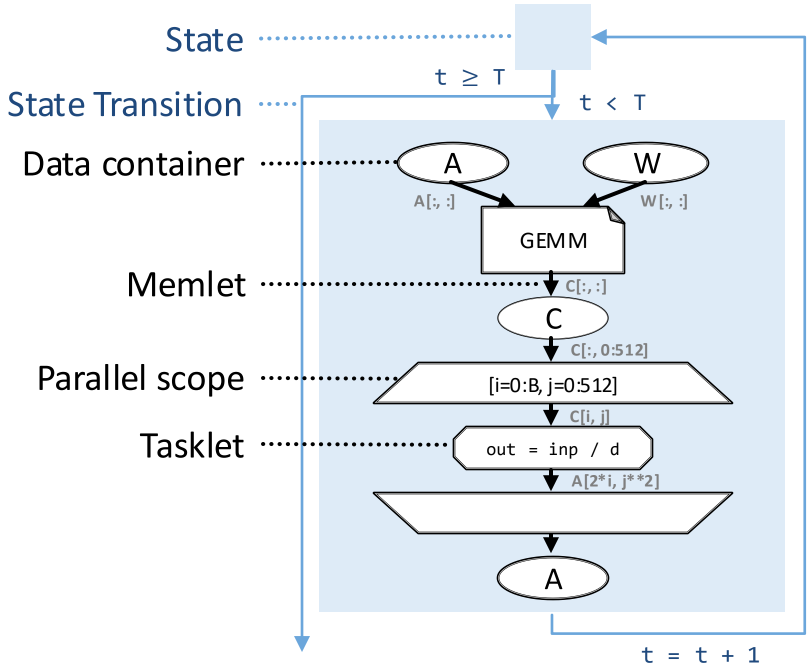
- Data containers unique but not single-assigned
  - Allocation can be controlled
- Parametric data movement and parallelism
- State machine exposes control-flow data dependence

Frontends for various languages:

- Python/NumPy, C, Fortran 90 (+'08)

Backends for various architectures:

- CPU, GPU, FPGA



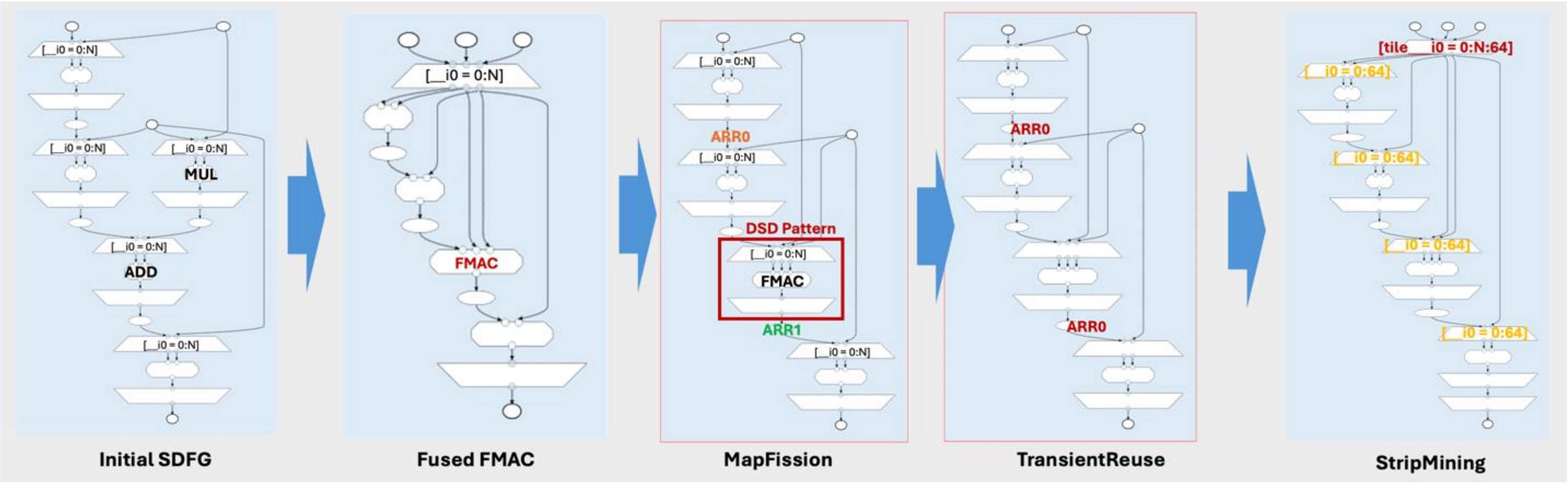
https://github.com/spcl/dace

T. Ben-Nun et al., "Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures", SC'19.

DaCe Spatial DataFlow Graph has a good alignment with spatial dataflow accelerators

```
def calc_position_for_nodes(domain: Domain, dt: float):
    domain.x += domain.xd * dt
    domain.y += domain.yd * dt
    domain.z += domain.zd * dt
```

```
@fmacs(domain.x_dsd, domain.x_dsd, domain.xd_dsd, domain.deltatime);
@fmacs(domain.y_dsd, domain.y_dsd, domain.yd_dsd, domain.deltatime);
@fmacs(domain.z_dsd, domain.z_dsd, domain.zd_dsd, domain.deltatime);
```



https://github.com/tbennun/pylulesh/blob/master/lulesh.py#L425

Aligning DSL abstraction with hardware capabilities enables more efficient porting of compute kernels

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.