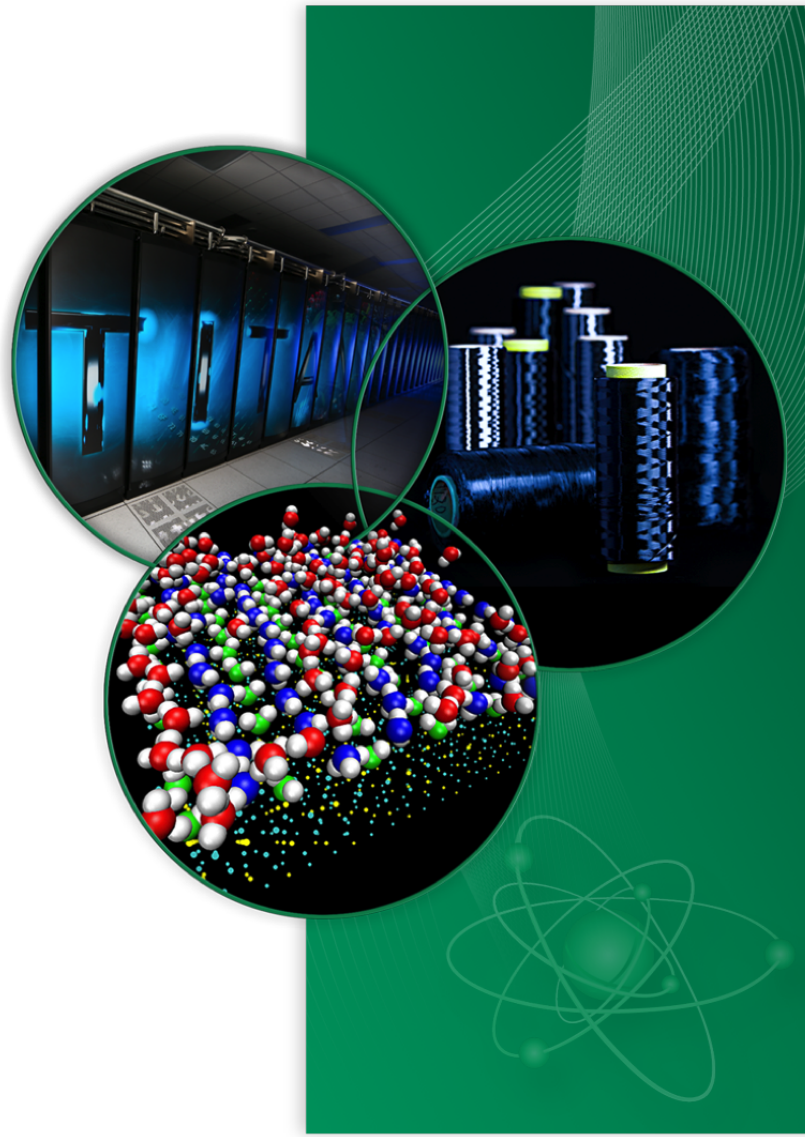
ArborX: a performance portable geometric search library

Andrey Prokopenko (ORNL)
Daniel Arndt (ORNL)
Damien Lebrun-Grandié (ORNL)
Bruno Turcksin (ORNL)



Kokkos tea-time

November 20, 2024

U.S. DEPARTMENT OF ENERGY

Introduction

OAK RIDGE National Laboratory

What is ArborX?

ArborX is an open-source **performance portable geometric search library** based on MPI+Kokkos.

Geometric search: find geometric objects that are close in some sense.

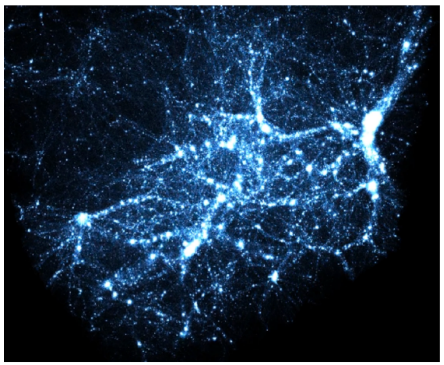
- **Search**
  - k-nearest neighbors (k-NN)
  - Range (radius search, intersections)
- **Ray Tracing**
- **Clustering algorithms**
  - Minimum spanning tree (Euclidean MST)
  - Density based clustering (DBSCAN, HDBSCAN*)
- **Interpolation**
  - Moving least squares (MLS)

`https://github.com/arborx/ArborX`

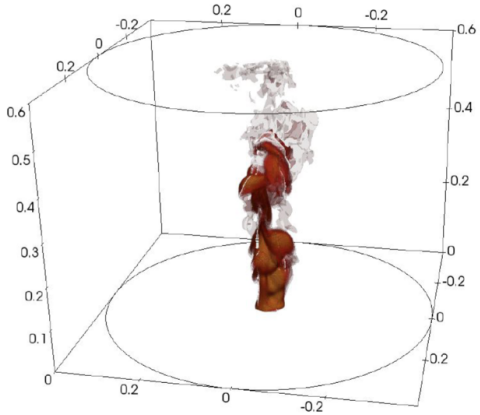
OAK RIDGE National Laboratory

Who uses ArborX?

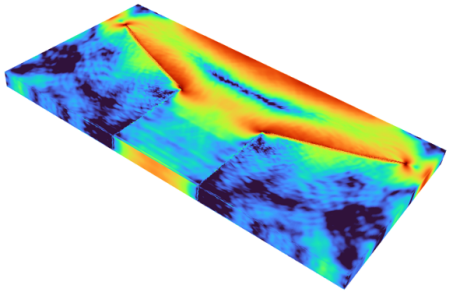
- **NimbleSM** contact mechanics
- **ALEGRA** shock hydrodynamics
- **LGRT** Lagrangian grid reconnection
- **deal.II** finite element library
- **MCRT** thermal radiation
- **Picasso** particle-in-cell
- **HACC/CosmoTools** clustering (dark matter)
- **Cabana** particle-based simulations
- **Adamantine** additive manufacturing
- **STK mesh** SIERRA/Trilinos meshing



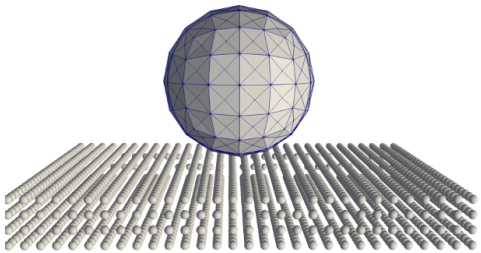
Cosmology (Credits: Nicholas Frontiere, ANL)



Combustion (Credits: Nicolas Tricard, UConn)



Additive manufacturing (Credits: Sam Reeve, ORNL)



Contact mechanicts (Credits: Nicolas Morales, SNL)

OAK RIDGE National Laboratory

Why use ArborX?

- Fast
- Flexible interface
- Performance portable (Kokkos)
- Modern C++
- Actively developed
- Both on-node and distributed

OAK RIDGE National Laboratory

Who is developing ArborX?

Core development team

- Daniel Ardnt
- Damien Lebrun-Grandié
- Andrey Prokopenko
- Bruno Turcksin

Most are also Kokkos developers.

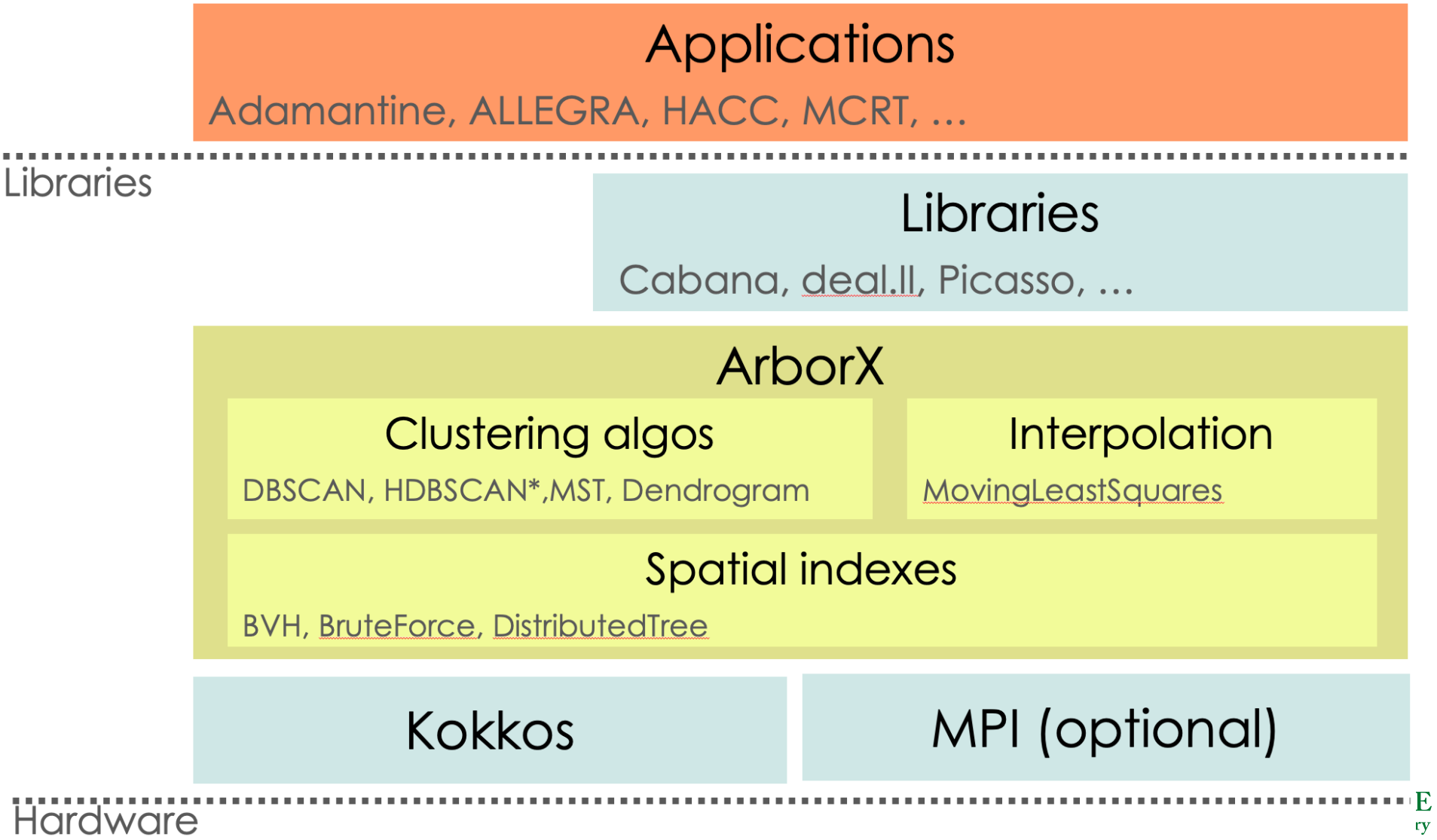



Contributors:

- Ana Gainaru
- Wenjun Ge
- Piyush Sao
- Yohann Bosqued

OAK RIDGE National Laboratory

ArborX in the scientific stack



Why Kokkos?

Context: start of US DOE Exascale Computing Project in 2017

- Facing the unknown beyond Summit (Nvidia GPUs)
- SYCL not around a that time
- RAJA? Kokkos? Roll our own?

Join forces with Kokkos

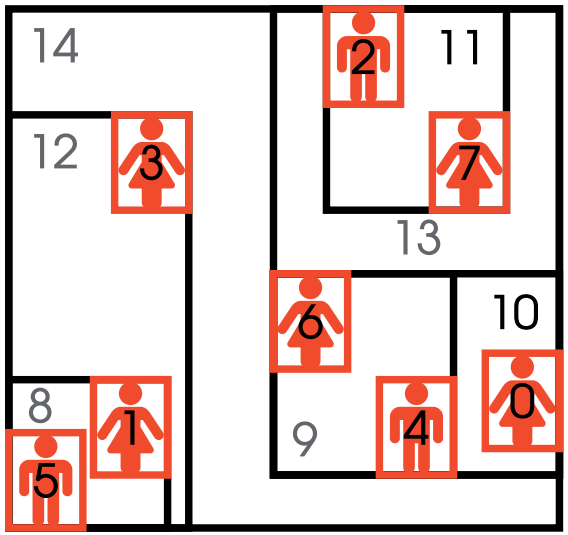
- More than a programming model
- Ecosystem with debugging and profiling tools, math libraries, etc.
- Building a community

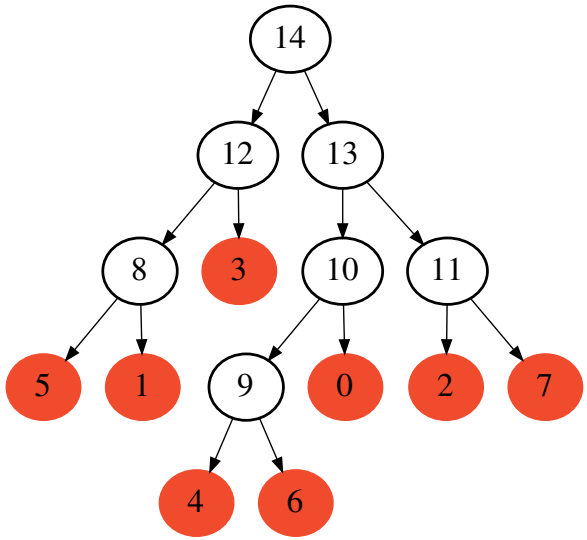
OAK RIDGE National Laboratory

Core concepts

Bounding volume hierarchy (BVH)

Bounding volume hierarchy (BVH) is a tree structure on a set of geometric objects, where each object is associated with a conservative bounding box.

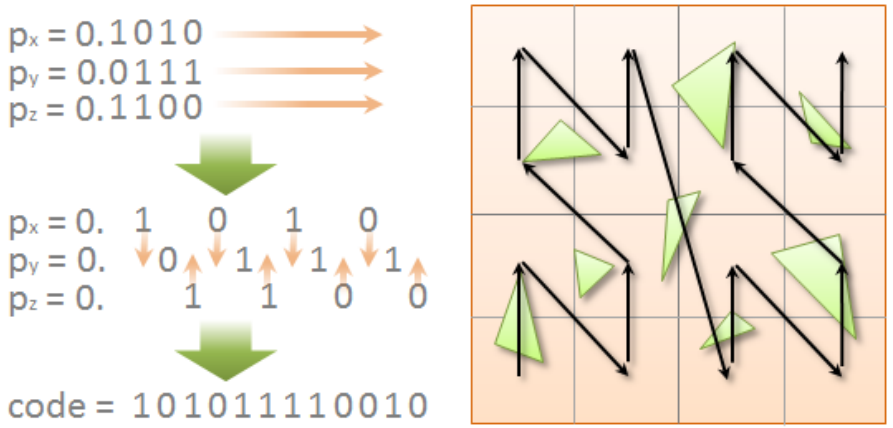




OAK RIDGE National Laboratory

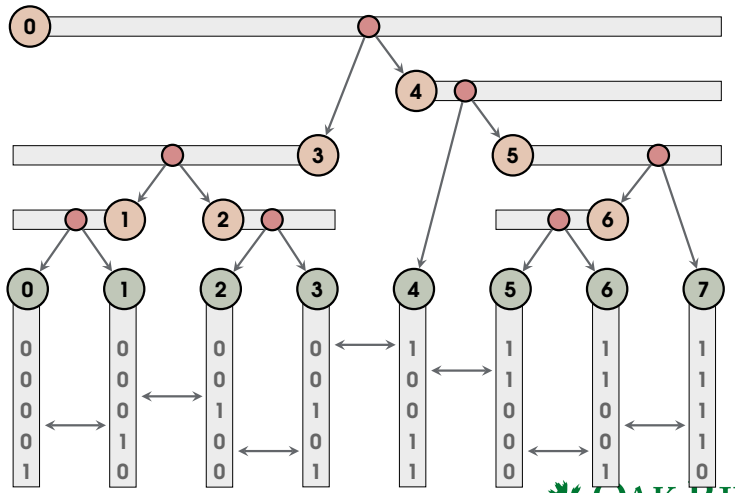
Linear BVH

Impose order in which leaf nodes appear in the tree (Z-curve/Morton codes)



(courtesy of T. Karras, NVidia)

- Each internal node is a linear range over leaf nodes
- The splits are determined according to the highest bit that differs between the Morton codes within the given range
- Can be constructed fully in parallel



OAK RIDGE National Laboratory

Two flavors of search

Range search

Find all the data that satisfies a criteria (withing certain distance, intersects, etc.)

- Optimized stackless traversal
- Do two passes (count-and-fill) as the number of found object is not known *a priori*
- Multiple knobs to speed things up:
  - Early termination
  - Half traversal for pairs

Nearest search

Find the predefined number of the closest neighbors to a given object.

- Single pass (know in advance how much memory is required)
- Use stack instead of priority queue

OAK RIDGE National Laboratory

Interface

OAK RIDGE National Laboratory

Concepts

- Search index is a container of values

  Can be anything (integers, geometries, user types)

- Values are transformed to geometries through indexable getter

  Could be user provided for custom, or default

- Bounding volumes can be customized (AABBs, kDOPs, OBBs)

  AABB by default; could be user provided

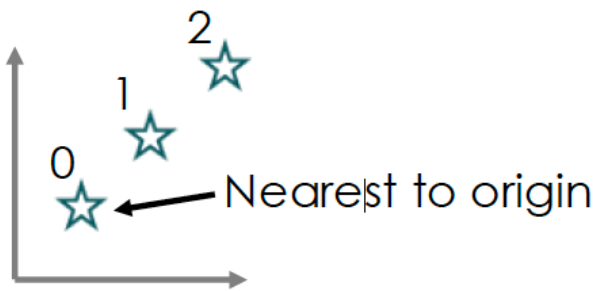
```
template<typename MemorySpace,
         typename Value,
         typename IndexableGetter,
         typename BoundingVolume>
class BVH {
  template <typename ExecutionSpace, typename Values>
  BVH(ExecutionSpace const& space,
      Values const& values,
      IndexableGetter const& indexable_getter);

  template <typename ExecutionSpace, typename Predicates,
            typename Callback>
  void query(ExecutionSpace const& space,
             Predicates const& predicates,
             Callback const& callback) const;

  template <typename ExecutionSpace, typename Predicates,
            typename Callback,
            typename Values, typename Offsets>
  void query(ExecutionSpace const& space,
             Predicates const& predicates,
             Callback const& callback,
             Values& values,
             Offsets& offsets) const;
};
```

OAK RIDGE National Laboratory

"Hello, world!" in ArborX




```
#include <ArborX.hpp>
#include <Kokkos_Core.hpp>
int main(int argc, charg *argv[]) {
  Kokkos::initialize(argc, argv);
  {
    Kokkos::DefaultExecutionSpace exec;
    // Build data structure
    ArborX::BoundingVolumeHierarchy bvh(
      exec, to-view({
        {1.f, 1.f}, // 0
        {2.f, 2.f}, // 1
        {3.f, 3.f}  // 2
      }));
    // Perform the search
    bvh.query(exec, to-view(
        ArborX::Nearest(ArborX::Point{0.f, 0.f, 0.f})
    ), KOKKOS_LAMBDA(auto /*predicate*/,
                     auto point /*value*/) {
      printf("Nearest to origin is (%f, %f)\n",
        point[0], point[1]);
    });
  }
  Kokkos::finalize();
  return 0;
}
```

Prints "Nearest to origin is (1, 1)"

OAK RIDGE National Laboratory

Access traits

- Customization point
- Opt-in mechanism to tell ArborX
  - where the data resides
  - how much of it
  - how to access
- Allowed to specialize for user-defined type

```
struct PointCloud {
  float *d_x, *d_y, *d_z;
  int N;
};

template <>
struct ArborX::AccessTraits<PointCloud>
{
  using memory_space = Kokkos::CudaSpace;

  static KOKKOS_FUNCTION
  std::size_t size(PointCloud const &cloud) {
    return cloud.N;
  }
  static KOKKOS_FUNCTION static ArborX::Point
  get(PointCloud const &cloud, std::size_t i) {
    return {{cloud.d_x[i], cloud.d_y[i], cloud.d_z[i]}};
  }
};
```

OAK RIDGE National Laboratory

User callbacks

Users may not care about the results themselves, or want a subset of the results.

For example:

- Call a function on the results

  May not need to store results, so less memory and single pass for spatial.

  Use case: finding a particle with most neighbors (defined by a length), or particles with a lowest potential (calculated as a function of friends)

- Do results pruning

  Use case: doing fine intersection search with the exact object geometry

- Use processor-local information

  Use case: compute interpolation coefficients for results using mesh parts on a processor

OAK RIDGE National Laboratory

User callbacks

```
template <typename MemorySpace>
struct Callback {
  template <typename Query, typename Value>
  KOKKOS_FUNCTION
  void operator()(Query const &query, Value const& value) const {
    auto data = ArborX::getData(query);
    // do something
  }
};

template <typename MemorySpace>
struct CallbackWithOutput {
  template <typename Query, typename Value, typename Output>
  KOKKOS_FUNCTION
  void operator()(Query const &query, Value const& value, Output const &out) const {
    auto data = ArborX::getData(query);
    // store something as a result
    out({...});
  }
};
```

OAK RIDGE National Laboratory

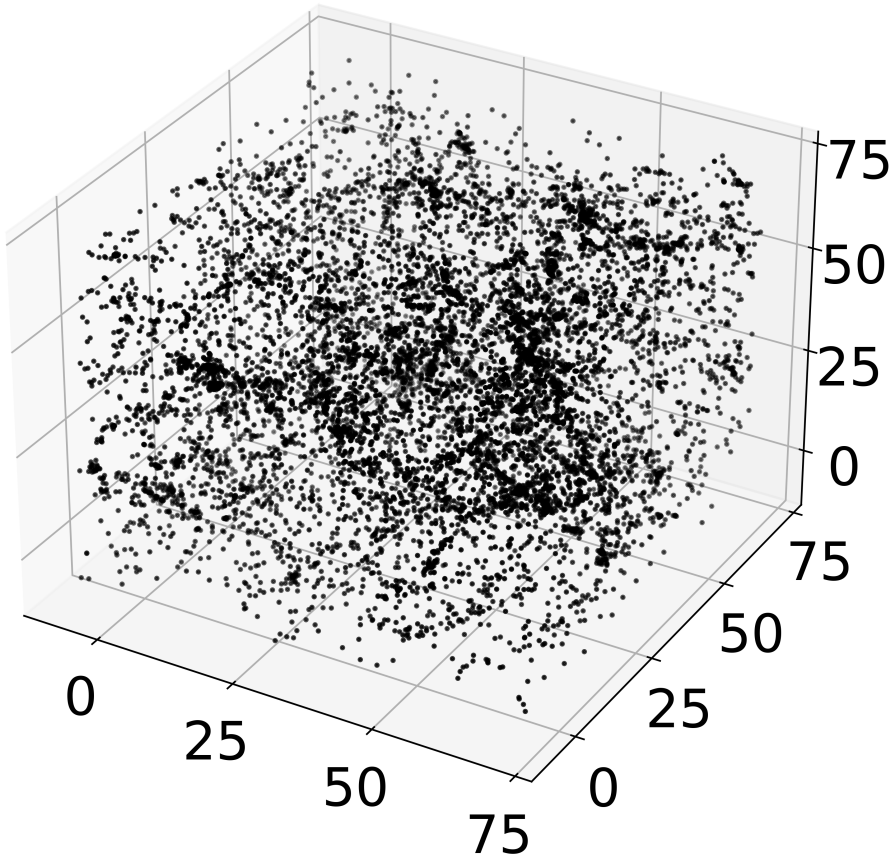
Use case 1: DBSCAN

Application

Cosmological simulation of the universe.

Data is massive. For example, 3096^3 particles running on 64 MPI ranks for a small run. In practice, 500M particles per GPU.

Goal: find dense regions of particles (halos) on each simulation time step.



OAK RIDGE National Laboratory

DBSCAN algorithm

Clustering: split a set of objects into disjoint classes (clusters), so that objects in each class are more similar to each other than to those in other classes.

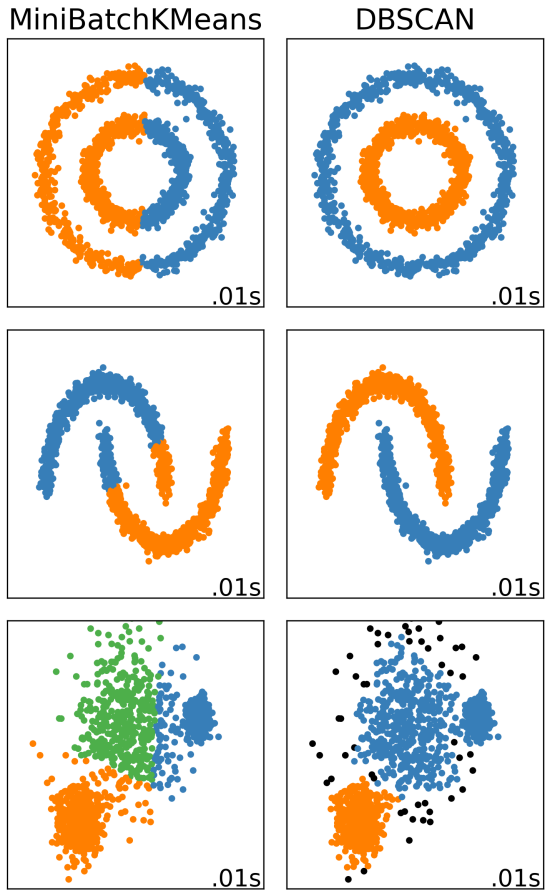
DBSCAN is a clustering algorithm based on density.

Main advantages:

- does not require to specify the number of clusters
- can find arbitrary shaped clusters
- has a notion of noise and is robust to outliers

It also has some disadvantages:

- not fully deterministic (border points)
- does not cope well with clusters of variable density
- choosing its parameters can be difficult



Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In Kdd, 96(34), pp. 226-231. 1996.

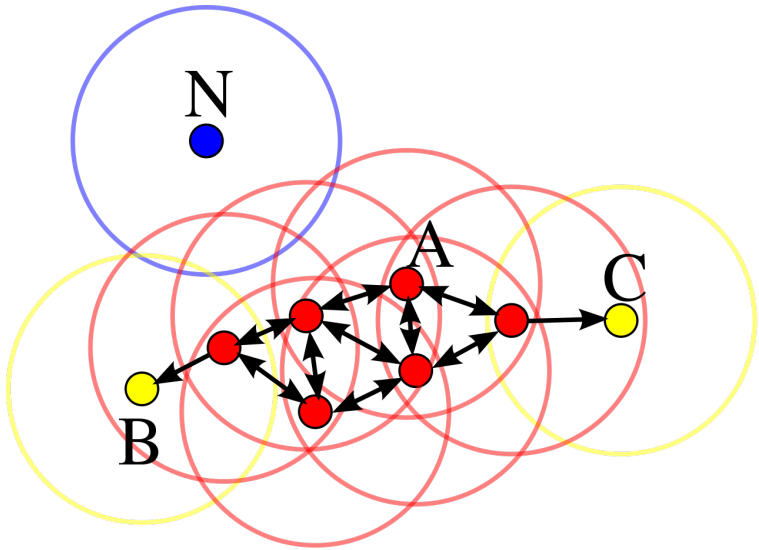
OAK RIDGE National Laboratory

DBSCAN algorithm

Given two parameters, ε and ***minpts***, DBSCAN separates all points into three classes:

- **core points**: have at least *minpts* neighbors within ε
- **border points**: not core points, but have a core point within ε
- **noise**: all other points

Any cluster consists of a combination of core points (at least one) and border points (possibly, none). Example below show *minpts* $= 4$.



OAK RIDGE National Laboratory

Image by Chire, Wikimedia Commons, distributed under CC BY-SA 3.0

F-DBSCAN (sparse)

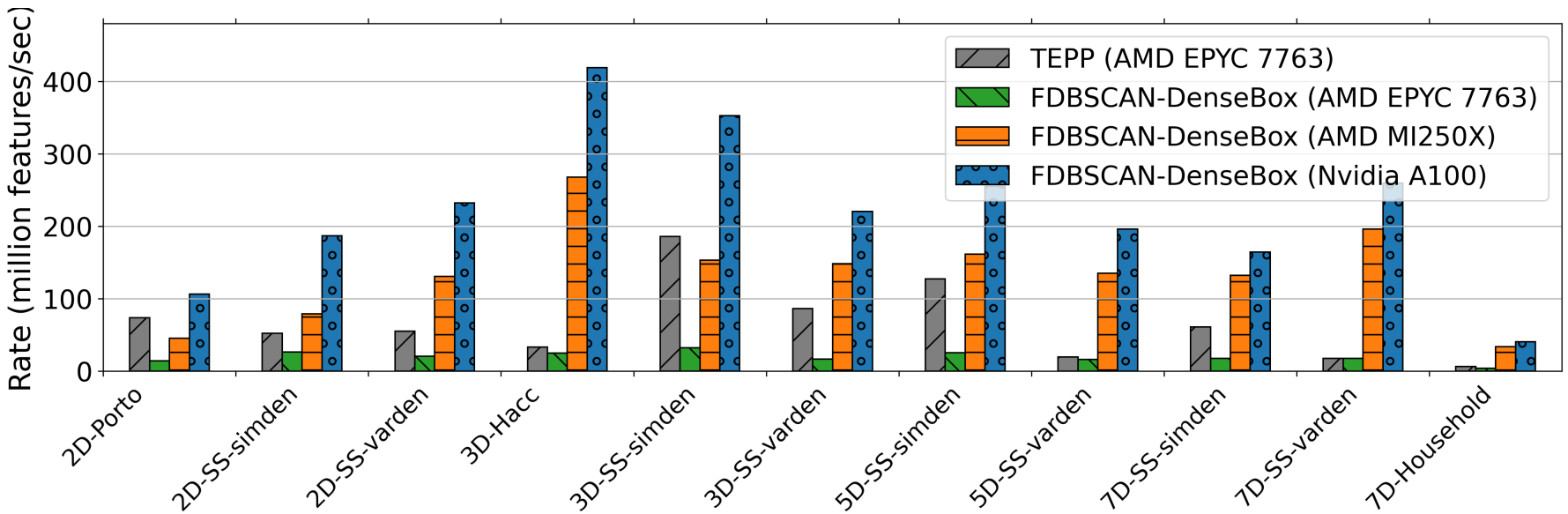
Focus: use parallel neighbor computation to minimize data and thread divergence on GPUs. Two phases: *preprocessing* and *main*.

- Use tree structure for search (e.g., kd-tree or BVH)
- Determine core points in preprocessing (terminate early)
- Implicit edges, execute UNION on collision

procedure F-DBSCAN$(X, \mathit{minpts}, \varepsilon)$
if $\mathit{minpts} > 2$ **then**
  for each point $x \in X$ **in parallel do**
    determine whether x is a core point
for each pair of points x, y
  such that $dist(x, y) \leq \varepsilon$ **in parallel do**
  if x is a core point **then**
    if y is a core point **then**
      $Union(x, y)$
    else if y is not yet a member of any cluster **then**
      critical section:
        mark y as a member of a cluster
      mark y as a member of a cluster
      $Union(x, y)$

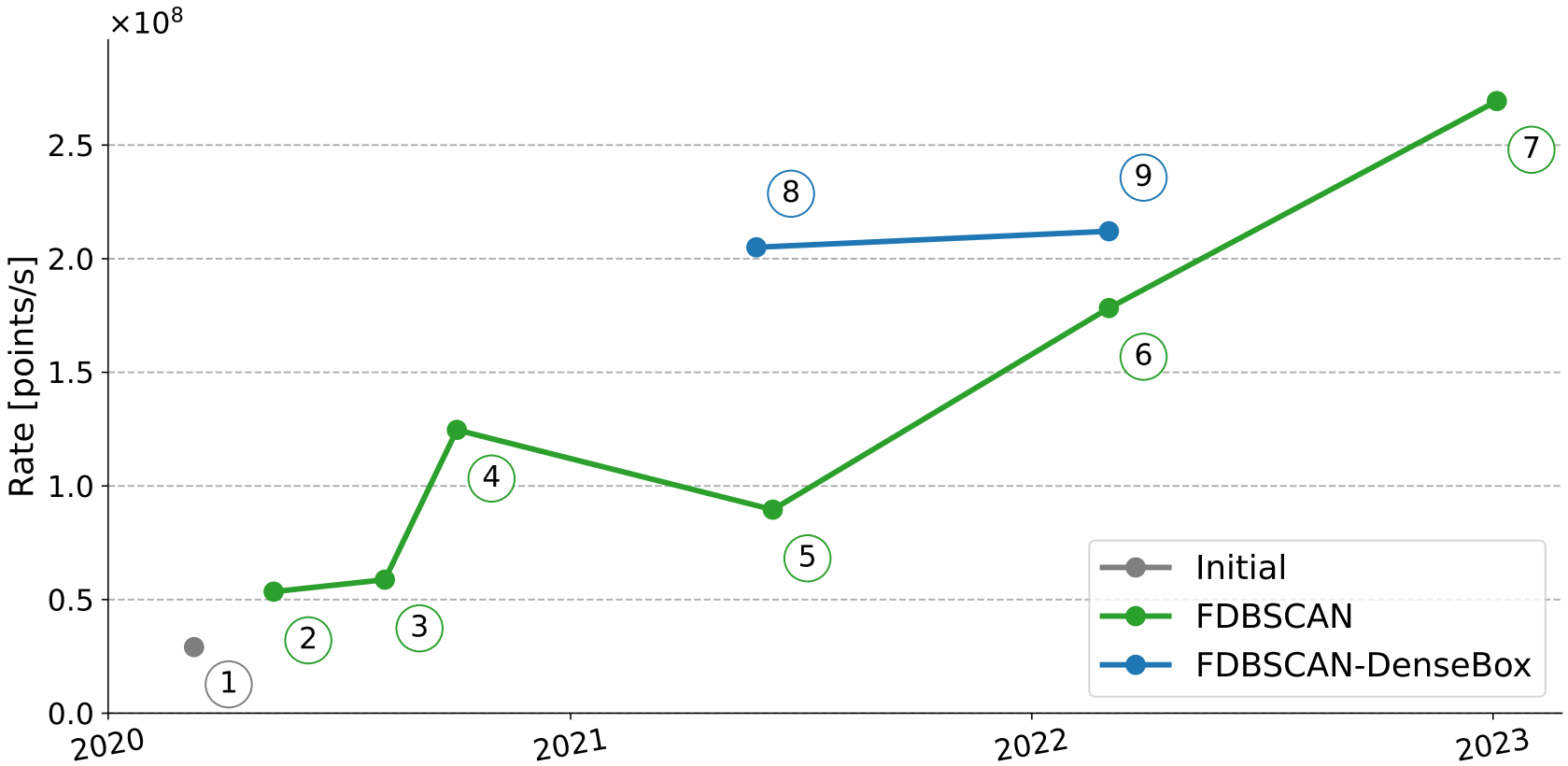
OAK RIDGE National Laboratory

Results: performance portability



OAK RIDGE National Laboratory

Results: application proxy performance improvement



OAK RIDGE National Laboratory

Use case 2: Euclidean minimum spanning tree (EMST)

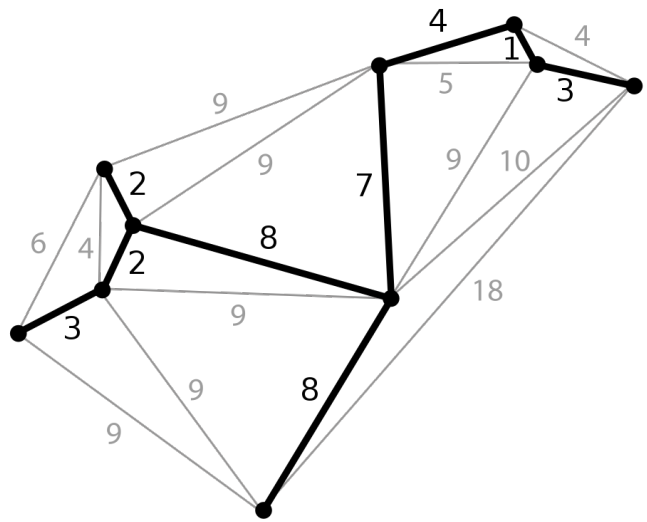
OAK RIDGE National Laboratory

Euclidean minimum spanning tree

A **minimum spanning tree** (**MST**) is a subgraph of a weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

A **Euclidean MST** (**EMST**) is a MST of the distance graph of a set of points, *i.e.*, a graph where each pair of vertices are connected by an edge of weight equal to the Euclidean distance between them.

Applications: data clustering (e.g., HDBSCAN*), Euclidean traveling salesman problem, wireless network connectivity, computational fluid dynamics, etc.

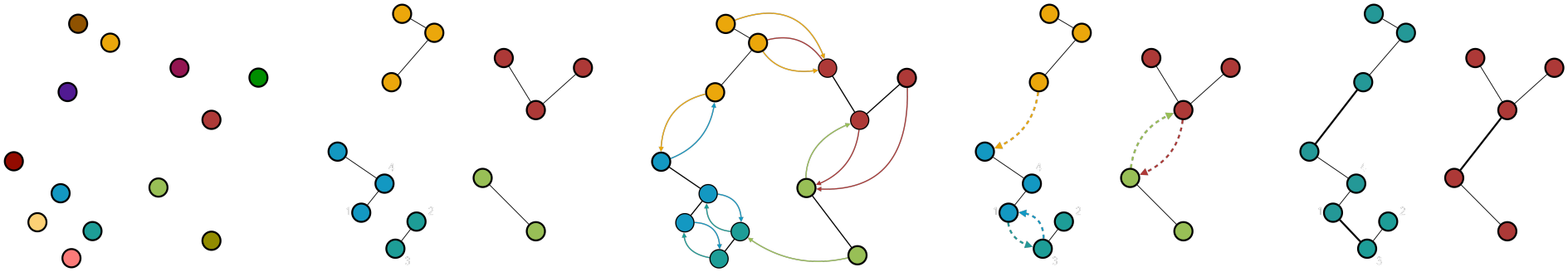


c/o Wikipedia

Our goal: fast EMST computation for large (10M+) dataset of low-dimensional data.

OAK RIDGE National Laboratory

Borůvka's algorithm



a. Initial state (each component having a single vertex).

b. The state after a few Borůvka iterations.

c. Closest neighbors from a different component for each vertex.

d. The shortest outgoing edge for each component.

e. The new components after the merge (the initial state of the next Borůvka iteration).

OAK RIDGE National Laboratory

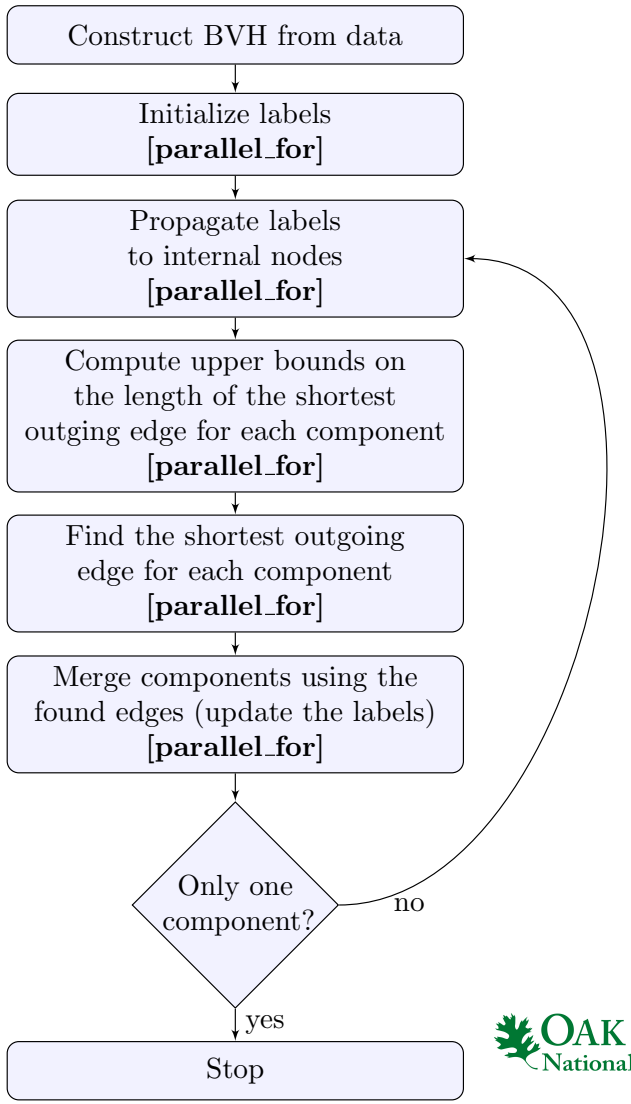
The single-tree algorithm overview

The Borůvka's algorithm is iterative. Each iteration consists of two phases:

1. find the shortest outgoing edge for each component (**expensive!**)
2. merge components

Finding the shortest outgoing edge: a **nearest-neighbor** problem with a **constraint** (the neighbor is from a different component). **Two optimizations** are necessary to prune the number of distance calculations:

- skipping nodes in the same component (**subtree skipping**)
- maintaining an **upper bound** on the distance of the outgoing edge candidates

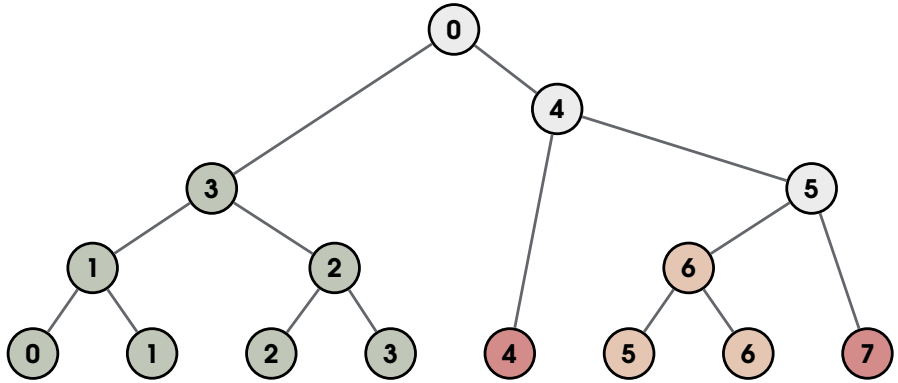


OAK RIDGE National Laboratory

Subtree skipping

Goal: Reduce the number of tree nodes encountered during the traversal.

- Leaf node labels are component membership of the data points.
- Propagate the labels from the leaf nodes to the internal nodes.
  Same labels of children $\rightarrow$ parent label. Different labels of children $\rightarrow$ invalid parent label. Done in a single bottom up traversal
- Skip subtrees of the same label as query during the traversal.



Very important on the **later** Borůvka iterations, when the components are large.

OAK RIDGE National Laboratory

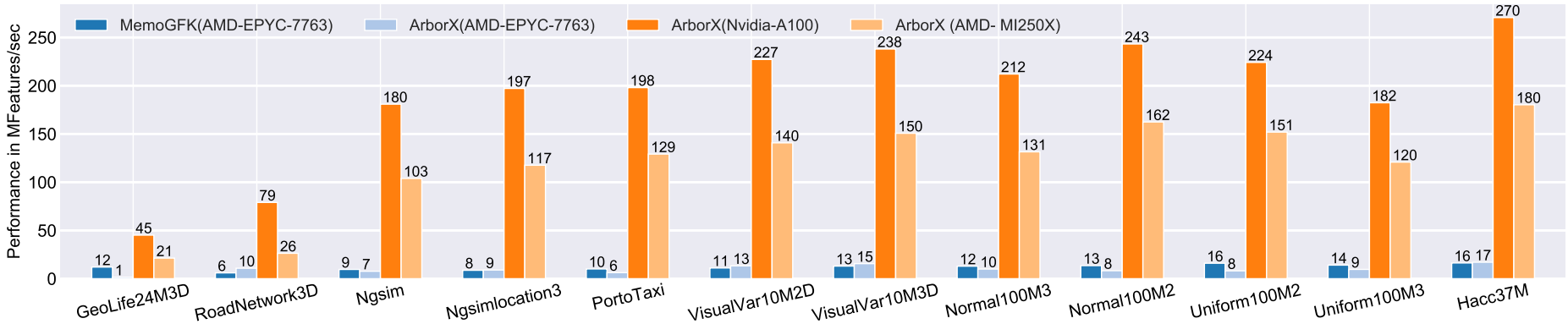
Datasets

Name	Dim	Size	Description
Ngsim	2	∼12,000,000	Car trajectories on three highways
Ngsimlocation3	2	∼6,400,000	A subset of *Ngsim* (one highway)
PortoTax	2	∼1,710,000	Taxi trajectories in Porto, Portugal
RoadNetwork3	2	∼400,000	Road network of a Denmark province
GeoLife24M3	3	∼24,000,000	User location data
Hacc37M	3	∼37,000,000	Cosmological data from a simulation
Hacc497M	3	∼497,000,000	Cosmological data from a simulation
VisualVar10M2	2	10,000,000	Synthetic data (Gan-Tao generator)
VisualVar10M3	3	10,000,000	Synthetic data (Gan-Tao generator)
Normal100M2	2	100,000,000	Normally distributed points
Normal300M2	2	300,000,000	Normally distributed points
Normal100M2	3	100,000,000	Normally distributed points

OAK RIDGE National Laboratory

Parallel performance

Performance comparison of the EMST implementations using AMD EPYC 7763 (64 cores), Nvidia A100 and AMD MI250X (single GCD[1]).



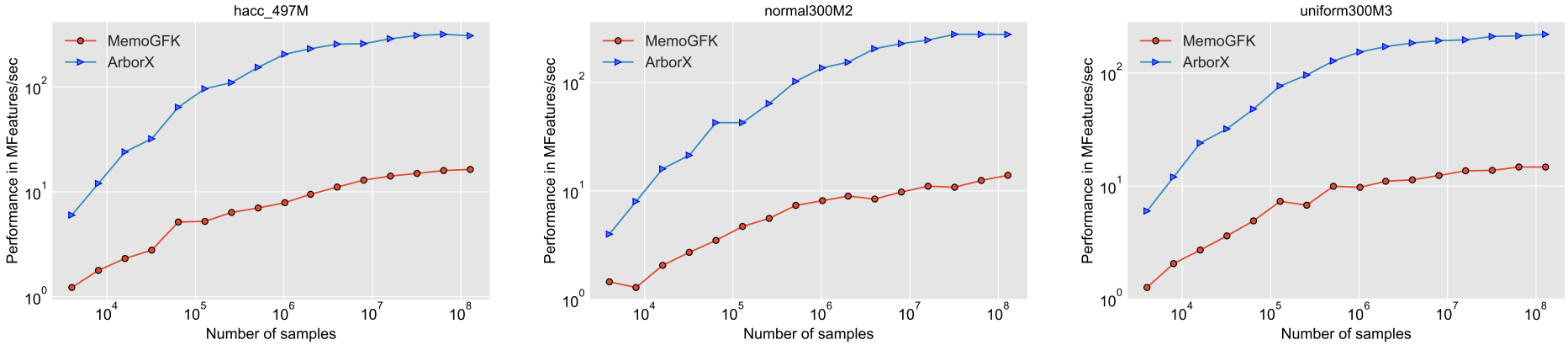
- ArborX implementation is faster by 4-24× than MemoGFK on Nvidia A100
- ArborX multi-threaded implementation is within factor 0.5-2× of MemoGFK (the fastest available multi-threaded implementation) (except *GeoLife24M3D*)
- ArborX optimized for Nvidia A100 but not for AMD MI250X

OAK RIDGE National Laboratory

[1] GCD = Graphics Complex Die

Scaling

Effect of the dataset size on the parallel performance using AMD EPYC 7763 and Nvidia A100.



- Performance increases with the number of samples until reaching saturation
- Asymptotically linear complexity of the algorithms

OAK RIDGE National Laboratory

Future

What does future hold?

- ArborX 2.0 (new interface!)
- Documentation!
- Performance improvements

  OBB hierarchy, hierarchy structure optimization, low precision
- New indexes

  Octree, kd-tree
- Approximate search
- Very high dimensional (dim > 10) search

OAK RIDGE National Laboratory

Questions?

https://github.com/arborx/ArborX

prokopenkoav@ornl.gov

OAK RIDGE National Laboratory