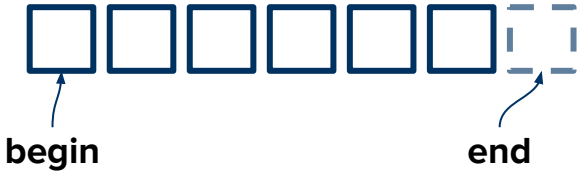
Distributed Ranges Spec.

Ranges

C++ 20 introduces **ranges**

A **range** is a **collection of values**

Range concepts provide a **standard way** to iterate over values




```
// Iteration
for (auto&& value : range) {
  printf("%d\n", value);
}
```

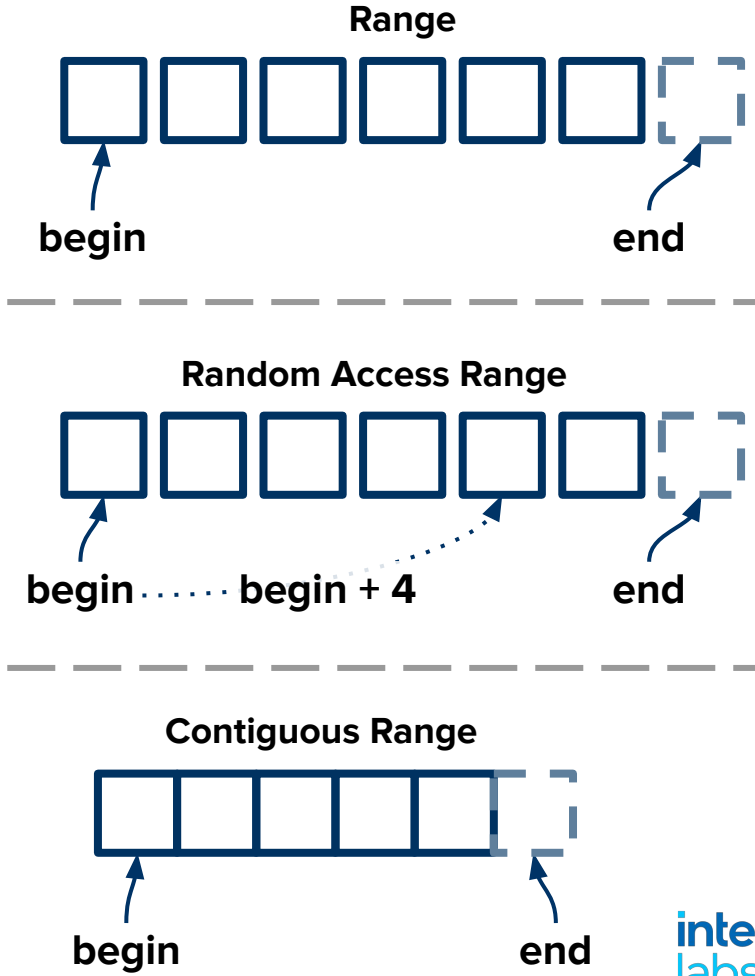
```
// Algorithms
auto r = std::ranges::reduce(range);
auto r = std::ranges::partial_sum(range);
```

```
// Views
auto add_two = [](auto v) { return v + 2; };
auto view =
     std::ranges::transform_view(range, add_two);
```

intel labs

Ranges API

- Have a **`begin()`** and **`end()`**
- Have a **`size()`** (usually)
- **Random access**: can access any element at random in **constant time**
- **Contiguous**: represents a **contiguous block of memory**



intel labs

There are lots of different **standard data structures**

—but they all expose the **ranges API**.

Distributed Data Structures

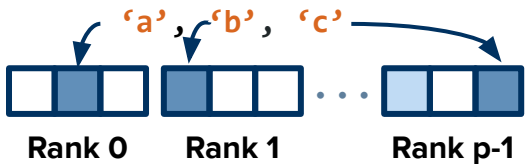
Distributed data structures **split up** data across multiple **segments**

Segments may be stored in **different memory regions**

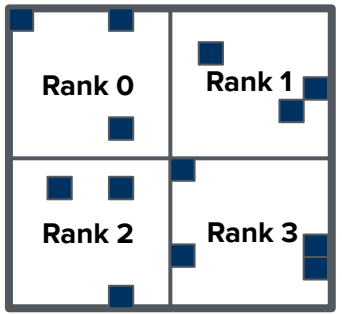
We need a unified API for accessing these distributed data structures!

Distributed Array

Distributed Hash Table



Distributed Matrix

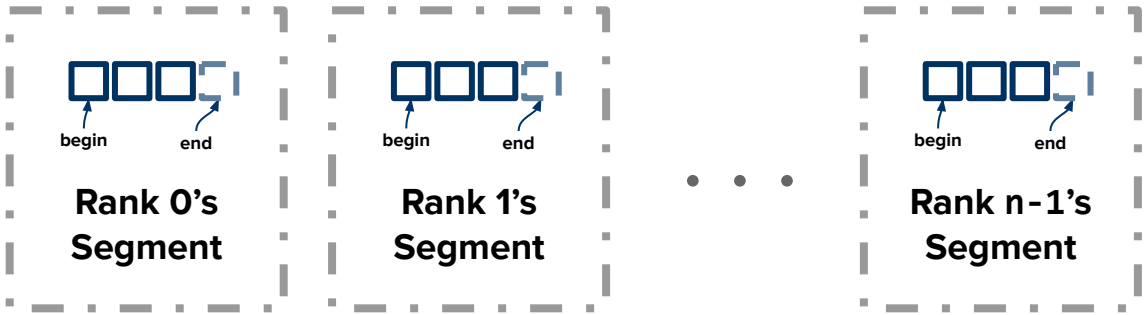


intel labs

Distributed Ranges

A distributed range is distributed across **segments**

Segments may be located on different **ranks**



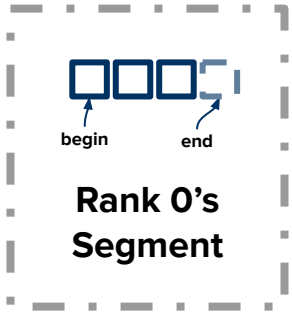
intel labs

Remote Range

Each of the segments in a distributed range is a **remote range**

A remote range is a **standard range**

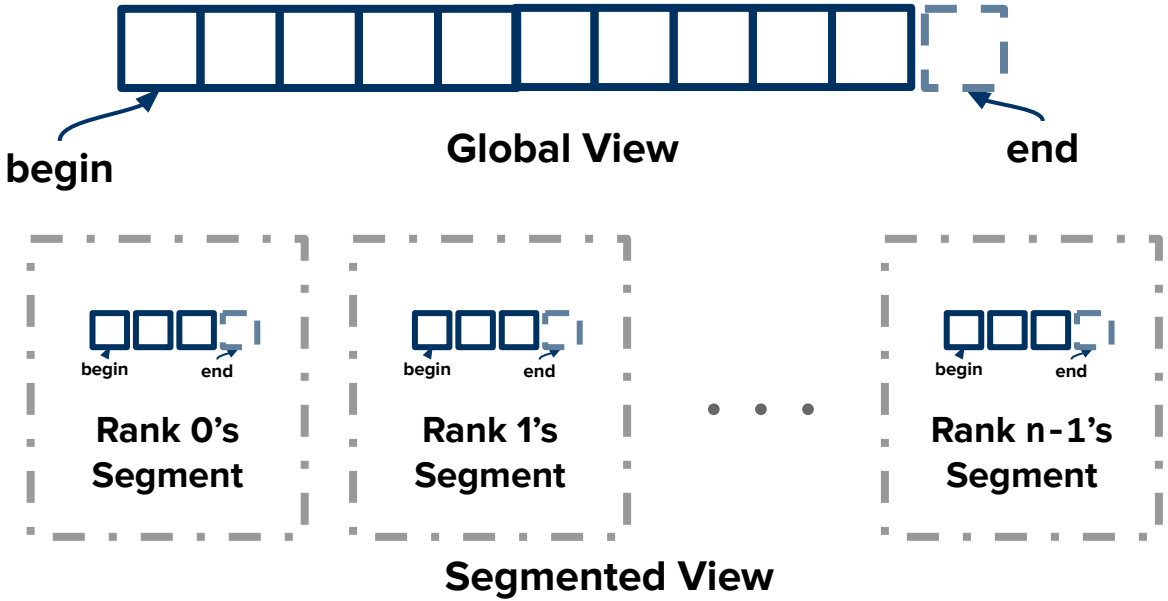
—Plus it has a **rank**



Our Proposal for Distributed Ranges

A distributed range is composed of **multiple remote ranges**

This creates both a **global view** and a **segmented view**



intel labs

Our Proposal for Distributed Ranges

- Is a **range**
- Has a `**segments()**` method
  - Each segment is a remote range

Remote ranges can be **different types** (**contiguous**, **random access**, etc.)