Declarative Data Collections for Portable Parallel Performance based on *oneAPI*[a,b,c]

oneAPI DevSummit 2023

Zhibo Li*

Zhibo.li@ed.ac.uk

[a] Funded by the Huawei-Edinburgh Joint Lab

[b] Best Paper Award, ACM SLE 2022

[c] School of Informatics, The University of Edinburgh

Call-to-Action

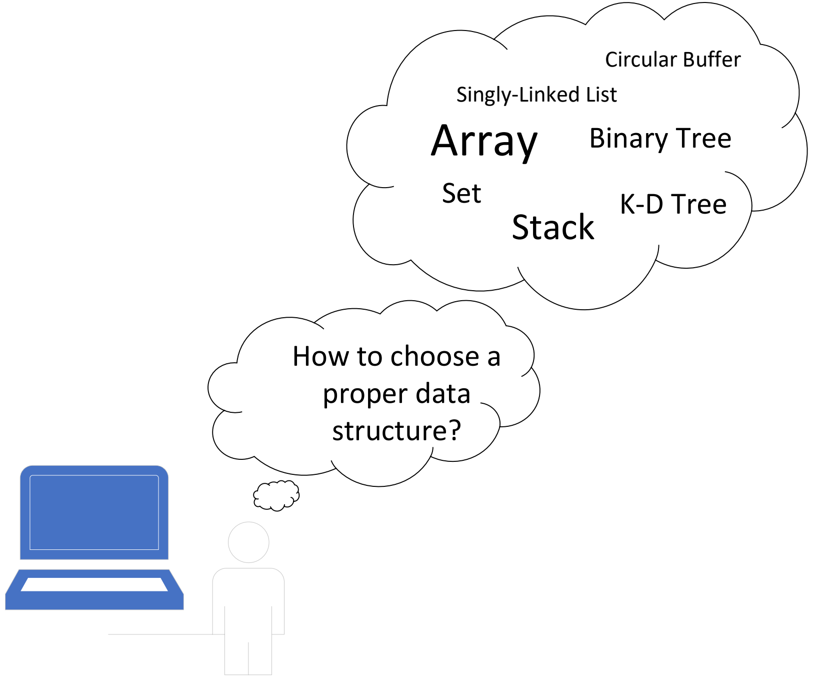
Write your 10-line-code parallel program that runs with oneAPI!

Abstract

1. Collection Skeletons - a novel, declarative approach to data collections through properties
2. Shielding the app developer from parallel implementation details
3. Parallelism shielding implicitly – concurrent implementations(oneTBB, oneDPL)
4. Parallelism shielding explicitly – parallelism algorithmic skeletons(SYCL from oneAPI)

Programming

How to choose a proper data structure?



Circular Buffer
Singly-Linked List
Array
Binary Tree
Set
Stack
K-D Tree

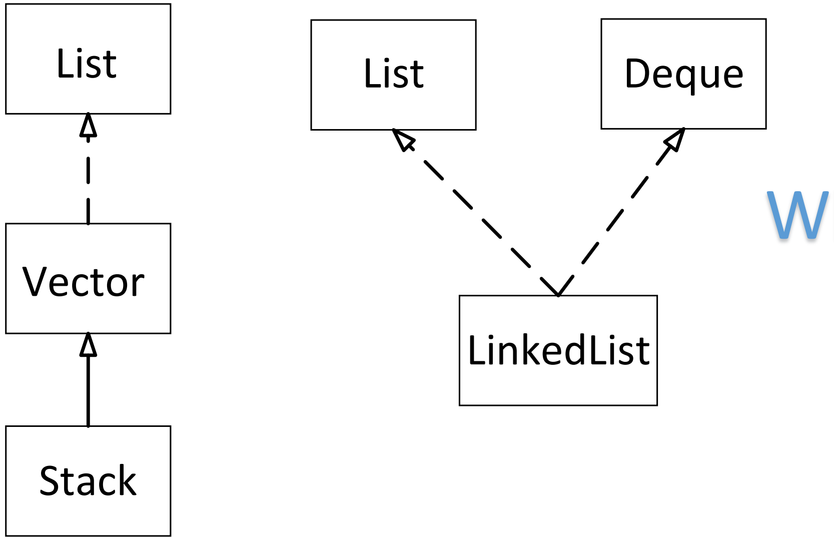
How to choose a proper data structure?

Std::forward_list
Std::list
Self-implemented list
cds::intrusive::lazy_list
Boost::container::slist

Maybe a list is suitable?

A proper data collection

It can be equally **hard** for a programmer to even **select the most appropriate** collection from an existing collection hierarchy such as **Scala and Java Collections**.

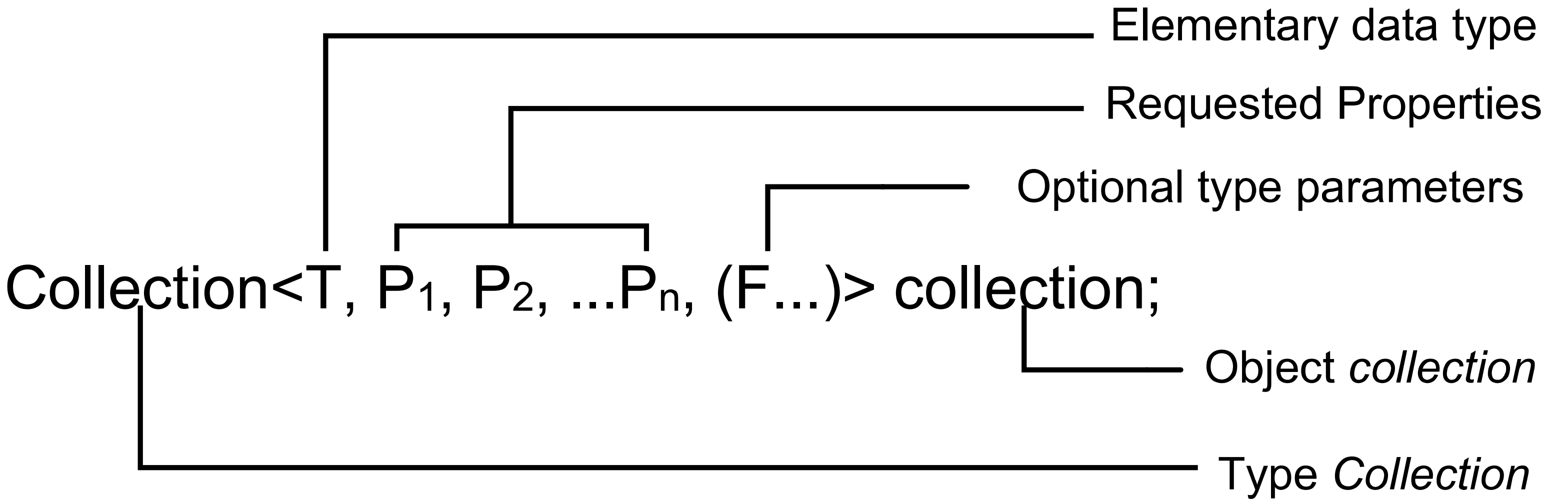


Why would a stack be a vector or a list ?

Collection Skeletons

1. Declarative API
2. Properties exposed and identified
3. **Parallelism shielding**

Programming API



Collection<T, P_1, P_2, ...P_n, (F...)> collection;

Elementary data type

Requested Properties

Optional type parameters

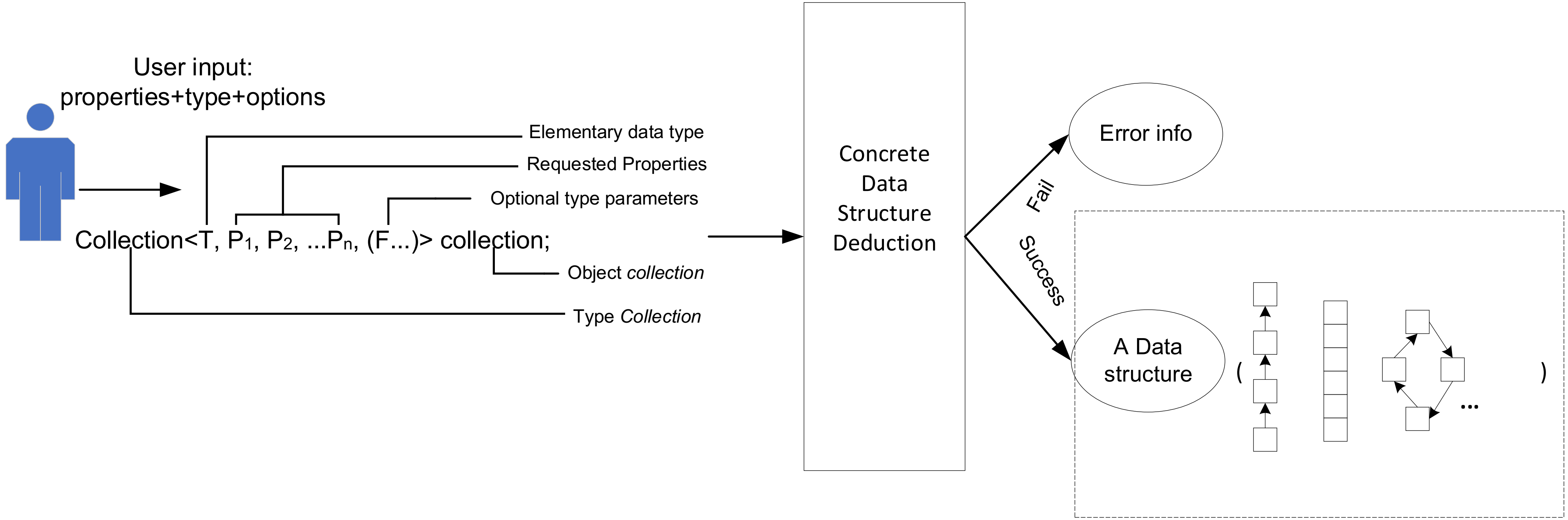
Object *collection*

Type *Collection*

Properties

1. Semantic Properties
2. Interface Properties
3. Non-functional Properties

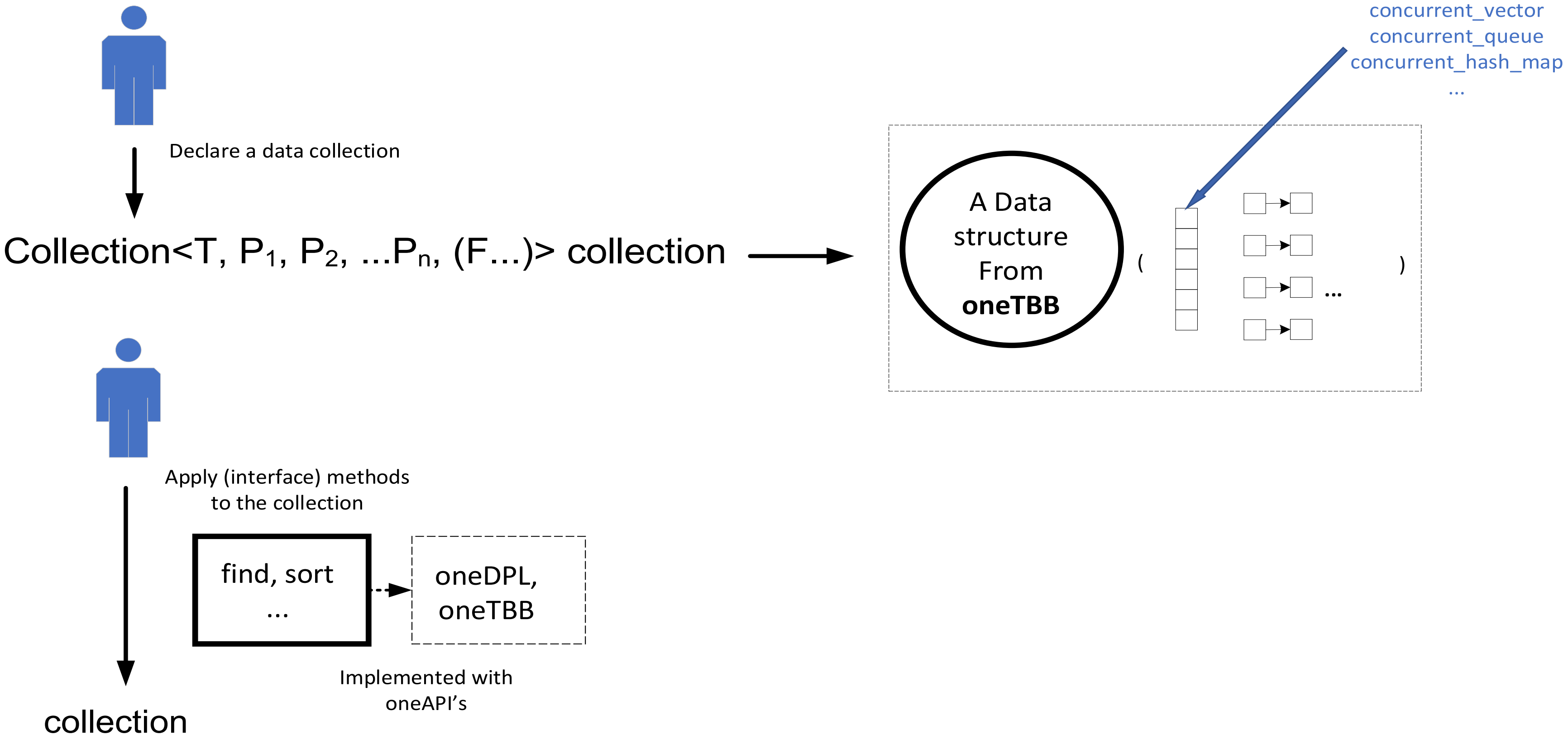
Library overview



Parallelism Shielding

1. Implicit Parallelism
2. Explicit Parallelism

Implicit Parallelism



Declare a data collection

Collection<T, P_1, P_2, ...P_n, (F...)> collection

A Data structure From **oneTBB**

concurrent_vector
concurrent_queue
concurrent_hash_map
...

Apply (interface) methods to the collection

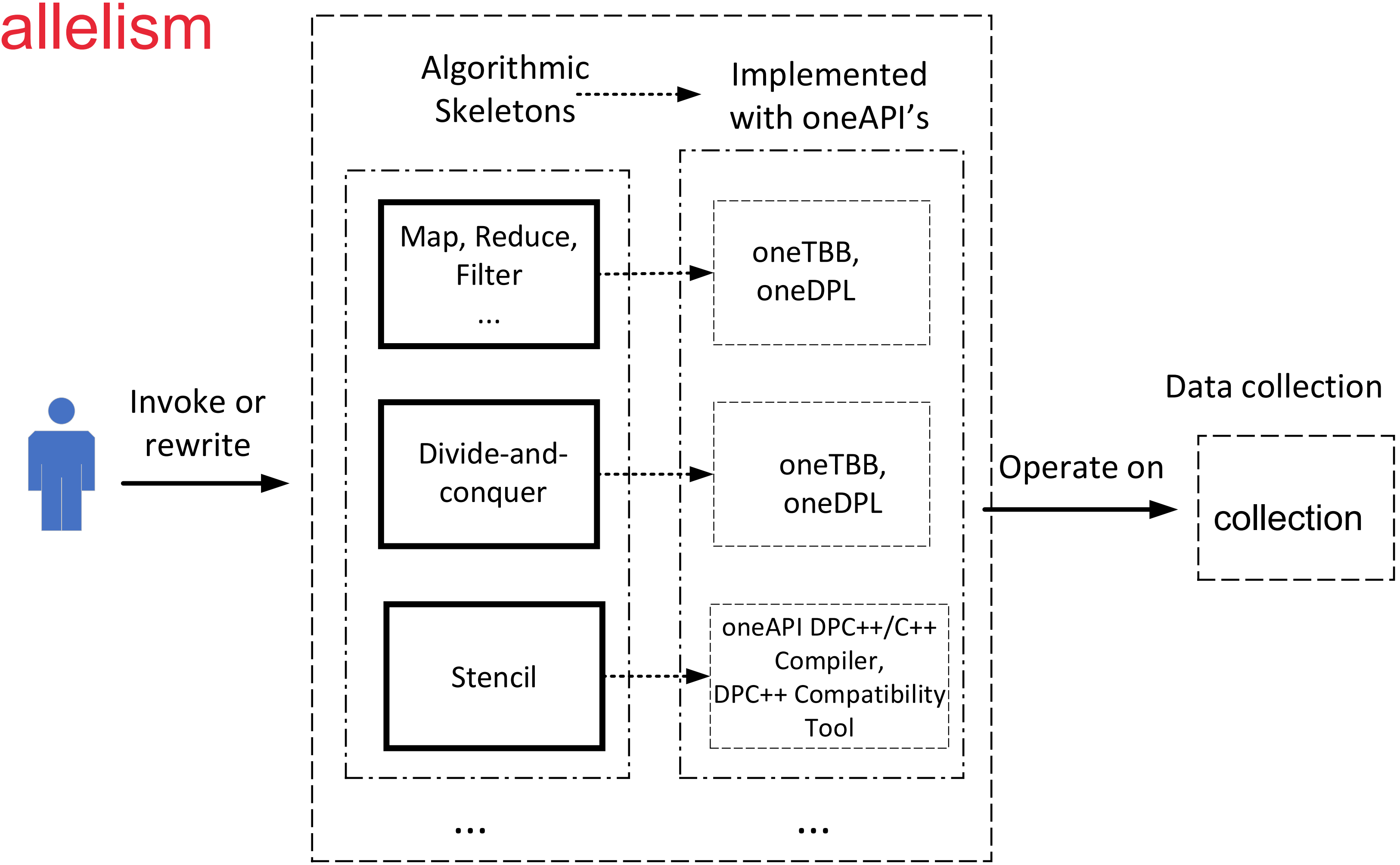
find, sort
...

oneDPL, oneTBB

Implemented with oneAPI's

collection

Explicit Parallelism



Explicit Parallelism

Parallelised algorithmic skeletons implemented with oneAPI

1. Data parallel algorithmic skeletons: map, reduce, filter, zip
2. Task based algorithmic skeletons: divide-and-conquer(splitable)
3. The library also **checks** if a collection can be done with algorithmic skeletons at compile time

Coding time – write your 10-line toy program that runs a parallel **map** on a GPU with oneAPI

```
#include<PropAbs.h>
using collection = Collection<int, Rnd>;
int main(){
  collection c{1,2,3,4};
  print_collection(c);
  //parallel map, implemented with oneAPI and can be
ported to GPUs
  auto c_new = map(c, [](int a){return a+1;});
  print_collection(c_new);
  return 0;
}
```

Thanks for listening! Q&A

Contacts and Social Media

Email: zhibo.li@ed.ac.uk

Personal page: https://libz.at

Devmesh: https://devmesh.intel.com/users/zhibo

Linkedin: https://www.linkedin.com/in/libz/

Tweet me @Zhibo_CS Find me on Techhub.social @User001

SLE 22 paper

JSS paper(preprint)