le savoir vivant

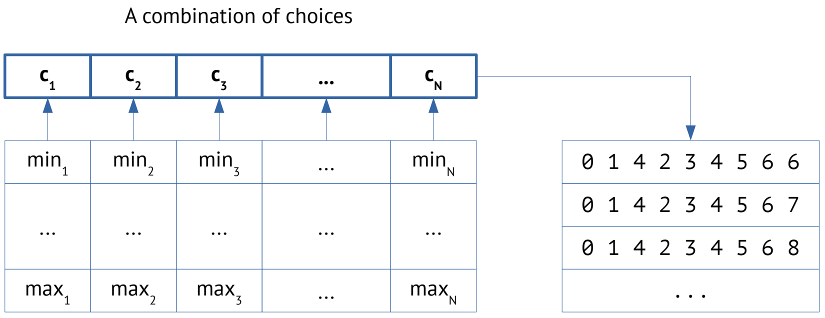
HPC Multi-iterator Engine Design & Implementation

Dmitry Mikushin Simon Scheidegger
Philipp Eisenhauer Moritz Mendel

May 4, 2021

UNIL | Université de Lausanne
HEC Lausanne

Multi-Iterator: the purpose



Each choice has its own constraints (runtime consts)

A client program receives combinations one-by-one without storing anything

Multi-Iterator: motivation

TODO Trend link video by Hal Finkel

Simple example

```
#include <multiit/multiit.h>

multiit::runtime::MultiIterator mi({ 2, 3, 4 });
// OR: multiit::compiletime::MultiIterator<2, 3, 4> mi;

int niters = 0;
while (1)
{
    niters++;

    bool next = mi.next();
    if (!next) break;

    const auto& current = mi.getCurrent();

    // TODO Use the current combination of choices in a target app.
}

printf("%d iterations visited\n", niters);
```

Supported types of multi-iterators

- `MultiIterator`: A group of indexes that iterate from 0 to the given upper value
- `LimitedMultiIterator`: A group of indexes that iterates only through indexes with total sum no greater than limit
- `GenericMultiIterator`: A hosting group of indexes, whose indexes are themselves groups of indexes

$\Rightarrow$ A `GenericMultiIterator` of `MultiIterator` and `LimitedMultiIterator` elements can be used to express many kinds of combinatorial iterations

Programming package structure

- `multiit`: general-purpose multi-iterators in C++, with unit tests
- `kernit`: multi-iterator *kernel* generator based on JIT-compilation located in the cloud (web service)
- `respyit` : subjected multi-iterators, with Python API, which quiries `kernit` to provide a kernel with required specifics

Cloud serivce + end-user application pipeline

TODO Pipeline figure

End-user example

```python
import respyit

# TODO
```