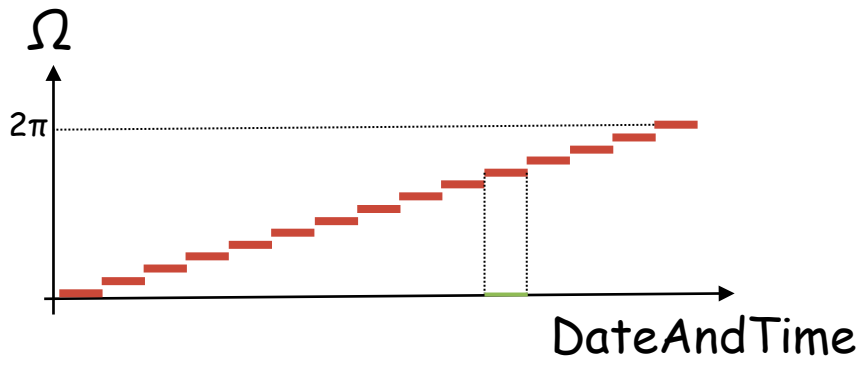
Changes to the Time-Slicing Backend

ORNL is managed by UT-Battelle, LLC
for the US Department of Energy

A Working Definition for Time Slicing

The *partition* of the time elapsed during one (or more) experiments

The partition is usually governed by measurable *environmental* quantities (temperature, electromagnetic fields, sample orientation)



Each partition element can be *labeled* by the values taken by the environmental quantities of interest

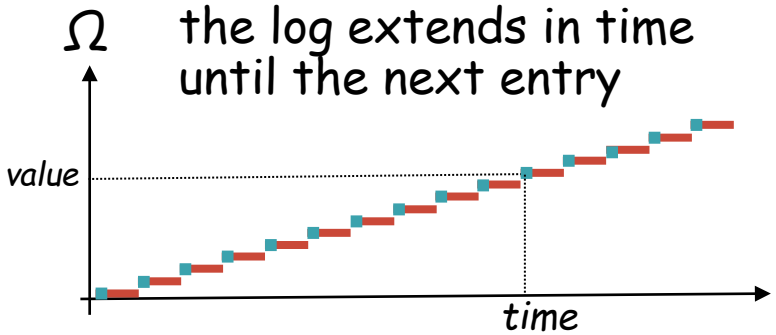
Targets of Time Slicing

Any set of time-stamped objects

- A list of events (EventList)
- A time series log (TimeSeriesProperty)

  $(t_1,v_1), (t_2,v_2), (t_3,v_3), \ldots$

Convention: the value of the log extends in time until the next entry

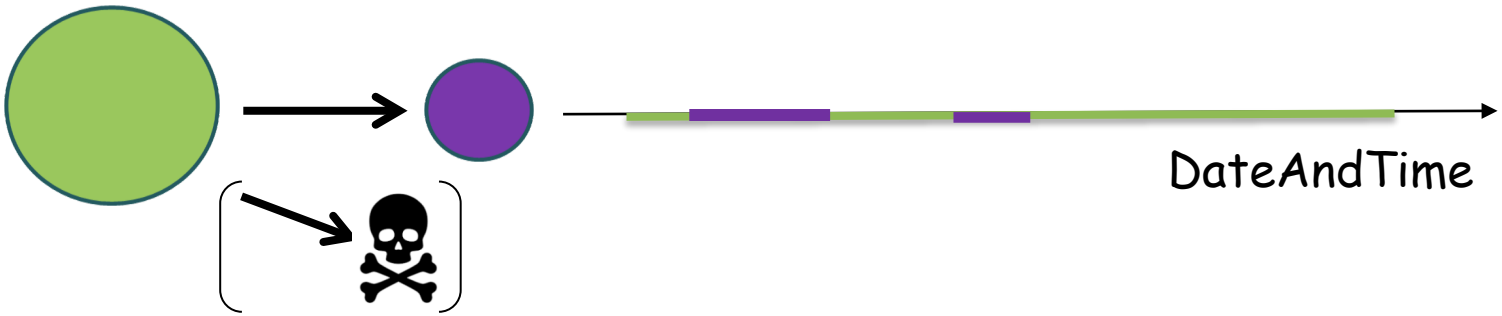


Algorithms involving Time Slicing

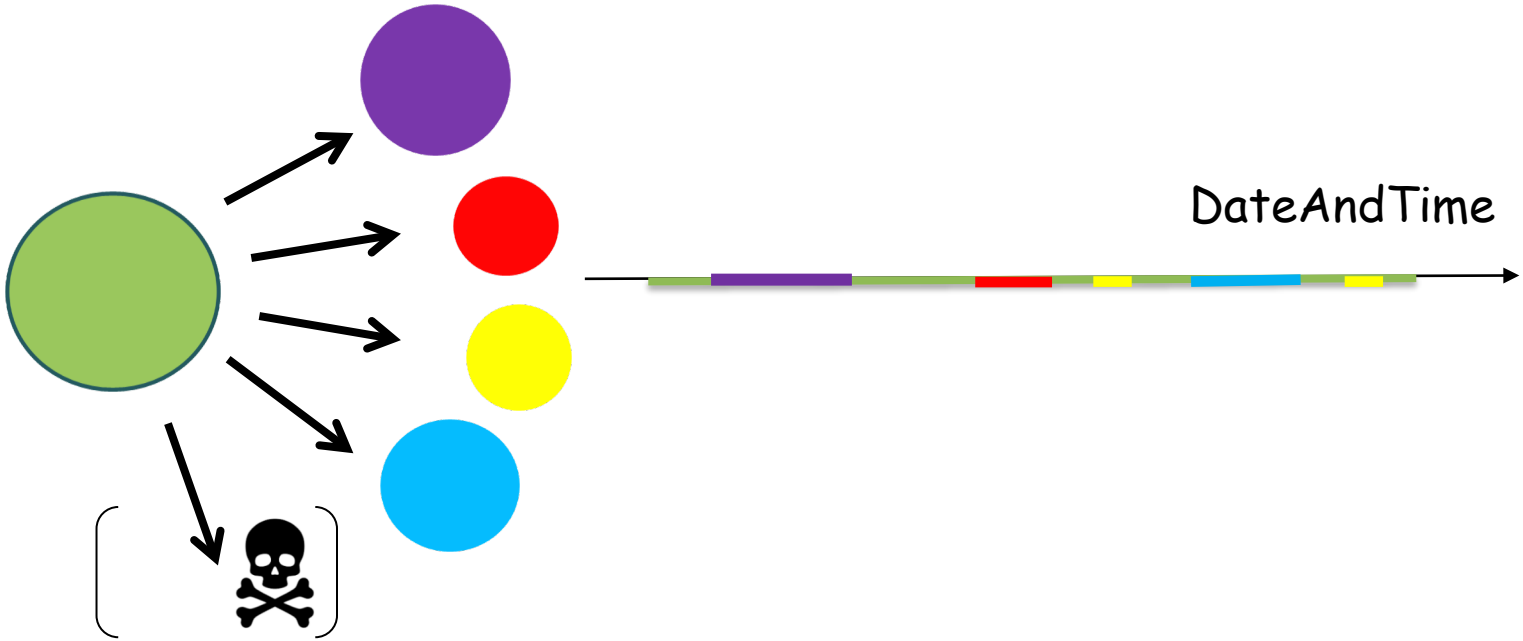
FilterBadPulses
FilterByLogValue
FilterByTime
FilterByXValue
FilterLogByTime
GetTimeSeriesLogInformation
ExportTimeSeriesLog
GenerateEventsFilter
FilterEvents
RebinByTimeAtSample

Types of Time Slicing?

- Filtering

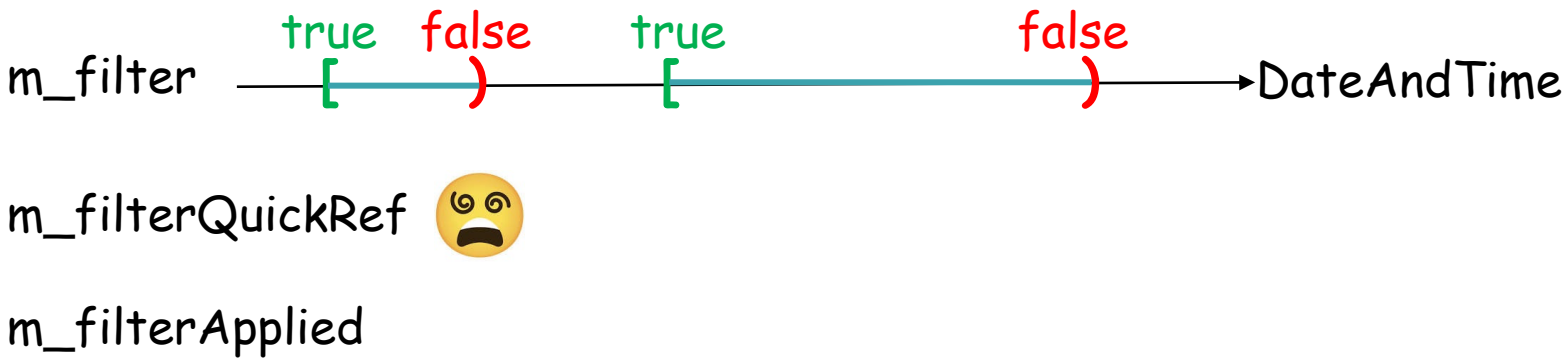


- Splitting



Status of the time-series Log class (≤ v6.6)

```
TimeSeriesProperty<TYPE>
  m_values : std::vector<TimeValueUnit<TYPE>>
  m_size : int
  m_propSortedFlag : TimeSeriesSortStatus
  m_filter : std::vector<std::pair<Types::Core::DateAndTime, bool>>
  m_filterQuickRef : std::vector<std::pair<size_t, size_t>>
  m_filterApplied : bool
```



Allows for *each log* in a workspace to have its own filter (rare case)

FilteredTimeSeriesProperty (≤ v6.6)

```
FilteredTimeSeriesProperty : public TimeSeriesProperty
    m_unfiltered : std::unique_ptr<const TimeSeriesProperty<HeldType>>
```

```
template <typename HeldType>
FilteredTimeSeriesProperty<HeldType>::FilteredTimeSeriesProperty(TimeSeriesProperty<HeldType> *seriesProp,
                                                                 const TimeSeriesProperty<bool> &filterProp)
    : TimeSeriesProperty<HeldType>(*seriesProp),
      m_unfiltered(std::unique_ptr<const TimeSeriesProperty<HeldType>>(seriesProp->clone())) {
  // Now filter us with the filter
  this->filterWith(&filterProp);
```

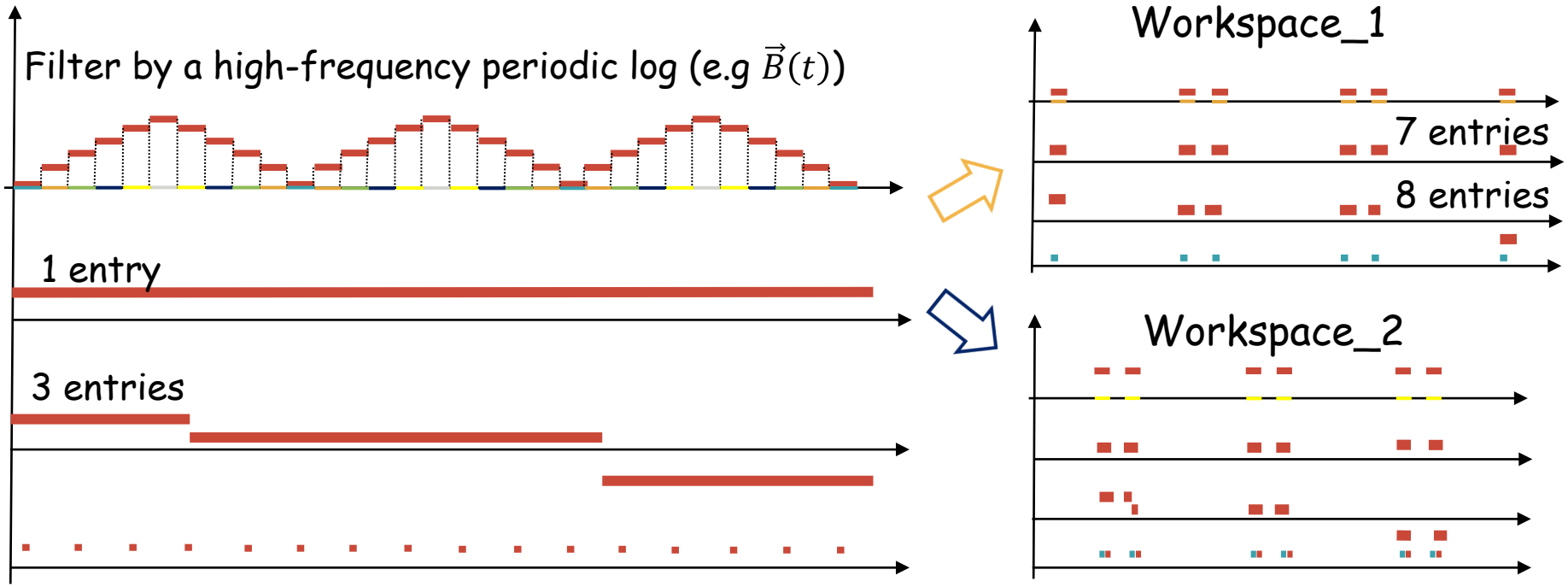
Duplicates the (t_1,v_1), (t_2,v_2), (t_3,v_3), … sequence

EventList Responsible for Splitting itself (≤ v6.6)

```
EventList
  splitByFullTime(TimeSplitterType &, map<int, EventList *>, bool, double, double)
  splitByFullTimeHelper(TimeSplitterType &, map<int, EventList *>, vector<T> &, b
  splitByFullTimeMatrixSplitter(const vector<int64_t> &, const vector<int> &, map
  splitByFullTimeSparseVectorSplitterHelper(const vector<int64_t> &, const vecto
  splitByFullTimeVectorSplitterHelper(const vector<int64_t> &, const vector<int>
  splitByPulseTime(TimeSplitterType &, map<int, EventList *>) const : void
  splitByPulseTimeHelper(TimeSplitterType &, map<int, EventList *>, vector<T> &)
  splitByPulseTimeWithMatrix(const vector<int64_t> &, const vector<int> &, map<
  splitByPulseTimeWithMatrixHelper(const vector<int64_t> &, const vector<int> &
  splitByTime(TimeSplitterType &, vector<EventList *>) const : void
  splitByTimeHelper(TimeSplitterType &, vector<EventList *>, vector<T> &) const :
```

- **5 public and 6 private helper methods**

Splitting a TimeSeriesProperty (≤ 6.6)



Splitting the other time series introduced spurious entries in the series' values

- Increased memory footprint
- Decreased execution speed
- Overestimation of the integrated proton charge

The TimeROI Object (> 6.6)

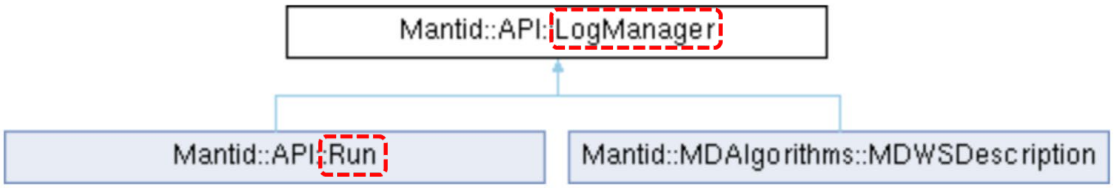
- A list of Regions-Of-Interest as time elapses


```
TimeROI
    m_roi : std::vector<Types::Core::DateAndTime>
```

even number of entries

- One TimeROI for *all* the TimeSeriesProperty in a given Run object

```
LogManager
    m_timeroi : std::unique_ptr<Kernel::TimeROI>
```



Status of the time-series Log class (> v6.6)

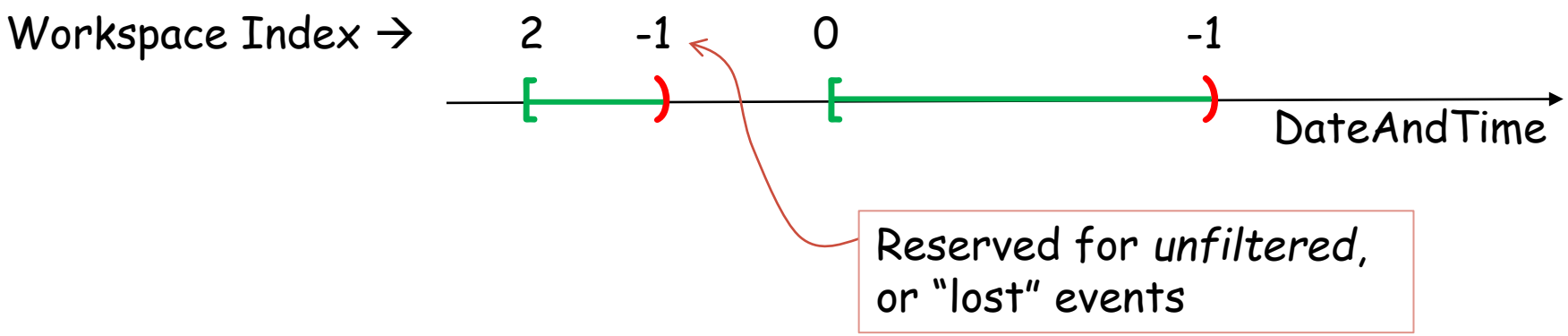
```
TimeSeriesProperty<TYPE>
    m_values : std::vector<TimeValueUnit<TYPE>>
    m_size : int
    m_propSortedFlag : TimeSeriesSortStatus
    m_filter : std::vector<std::pair<Types::Core::DateAndTime, bool>>
    m_filterQuickRef : std::vector<std::pair<size_t, size_t>>
    m_filterApplied : bool
```

```
FilteredTimeSeriesProperty : public TimeSeriesProperty
    m_filter : std::unique_ptr<TimeROI>
    m_filterMap : std::vector<size_t>
    m_filterIntervals : std::vector<TimeInterval>
    m_filterApplied : bool
```

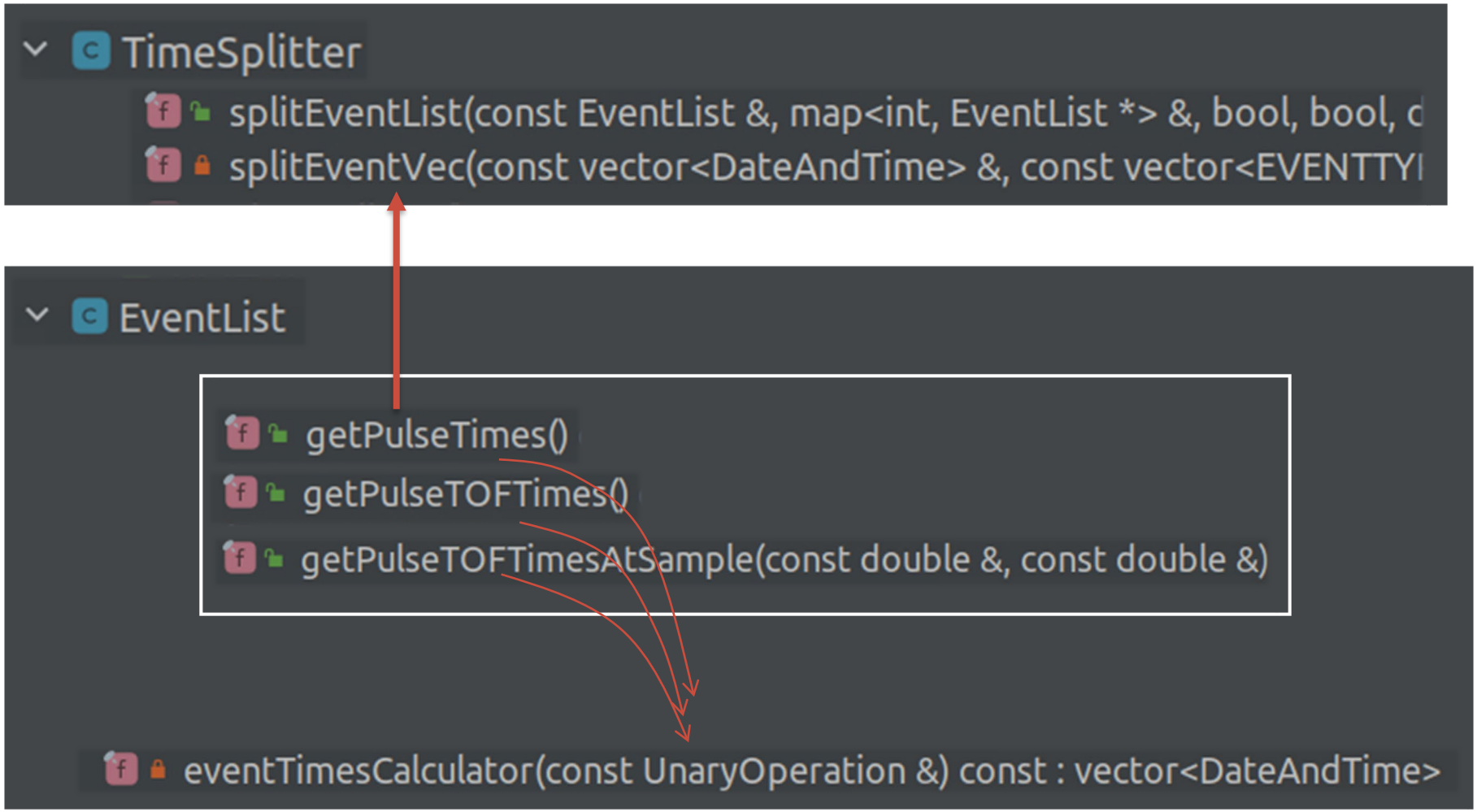
TimeSplitter object (> 6.6)

- Meant to substitute std::vec<SplittingInterval>
- Takes on the responsibility of filtering lists of events (*EventList*)

```
TimeSplitter
    m_roi_map : std::map<DateAndTime, int>
```

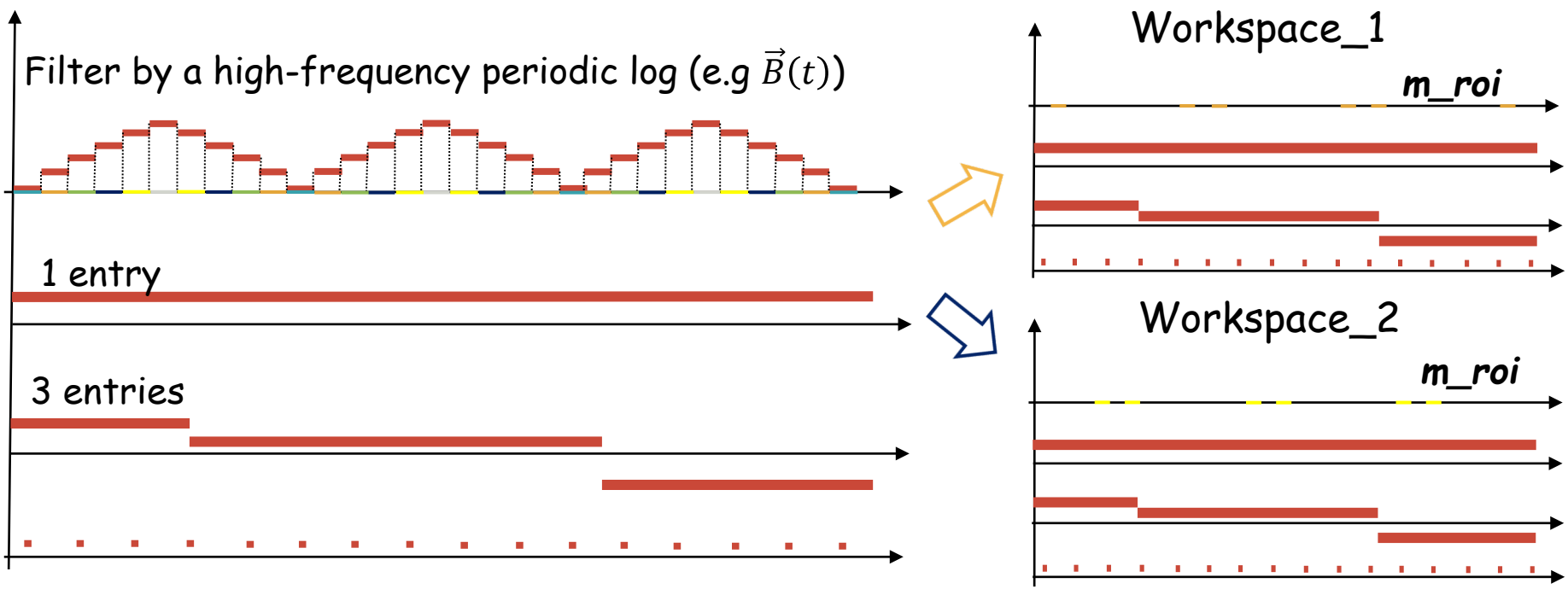


TimeSplitter Responsible for Splitting EventList



TimeSeries Filtering (>6.6)

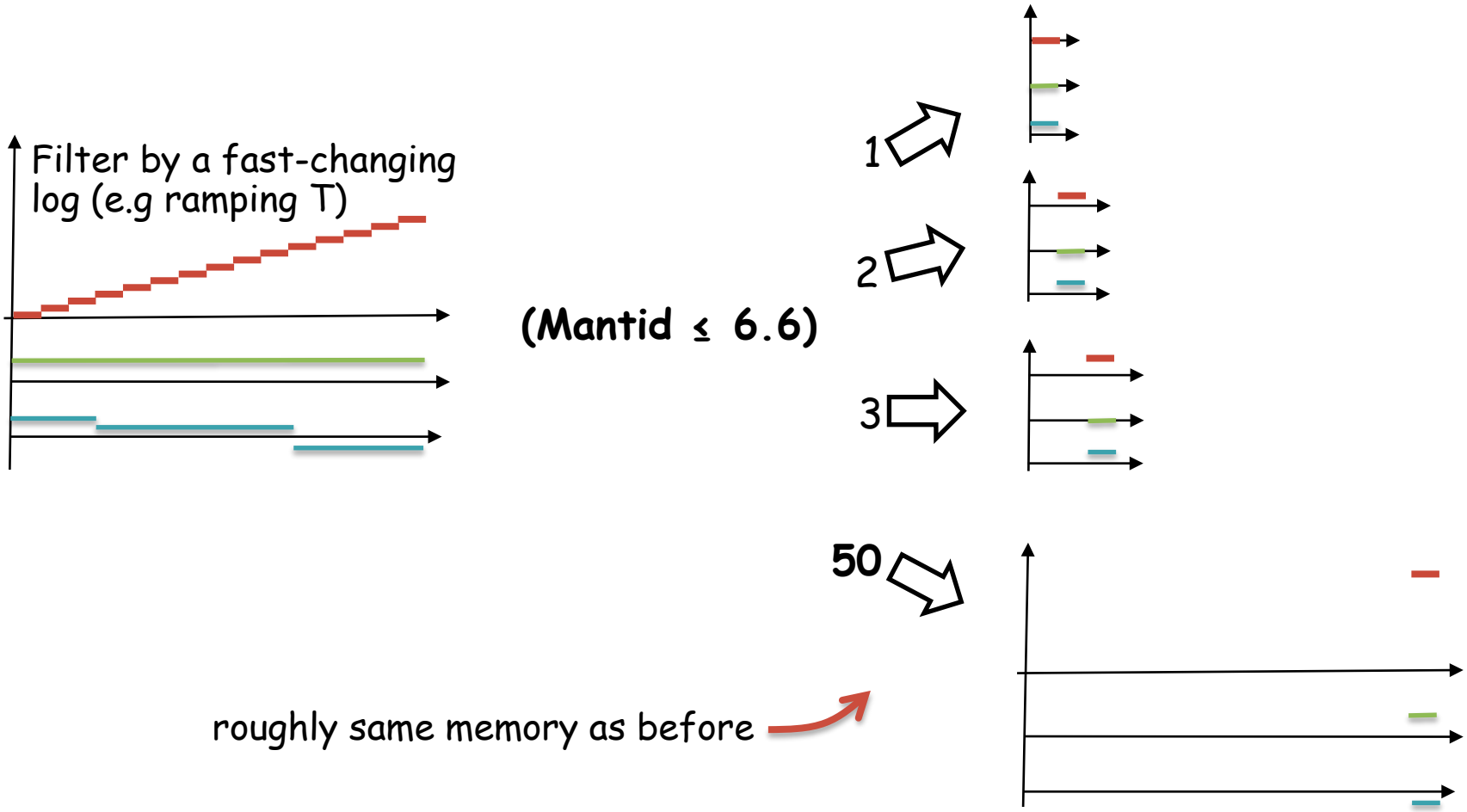
- Each Workspace ends up with cloned logs and its own TimeROI



- Time-weighted computations (such as average) consider the TimeROI

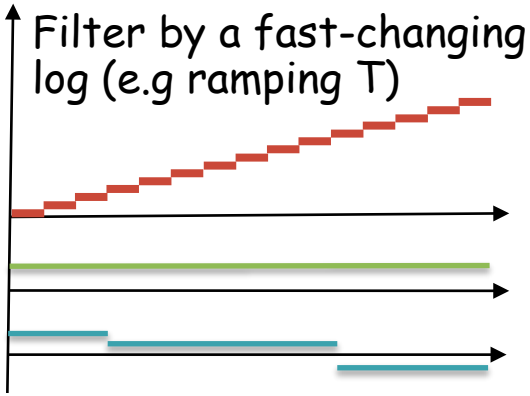
The Log Replication Problem

- Non-period, fast changing logs are ideally split into **very many** chunks

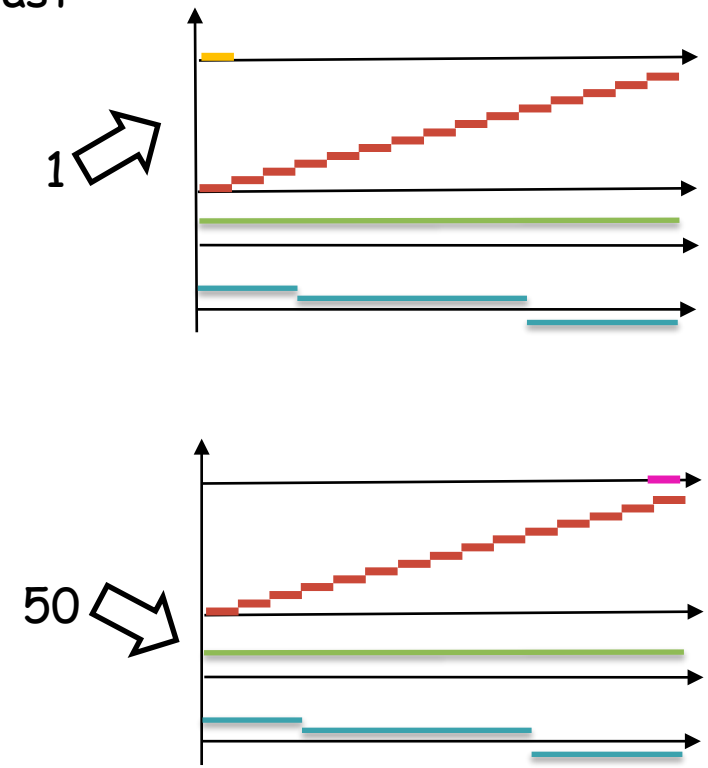


The Log Replication Problem

- Keeping full logs for each Workspace significantly increases the allocation of memory when splitting according to non-periodic, fast changing logs

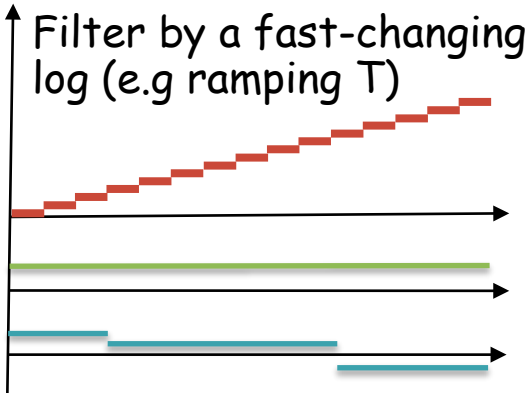


(Mantid > 6.7)

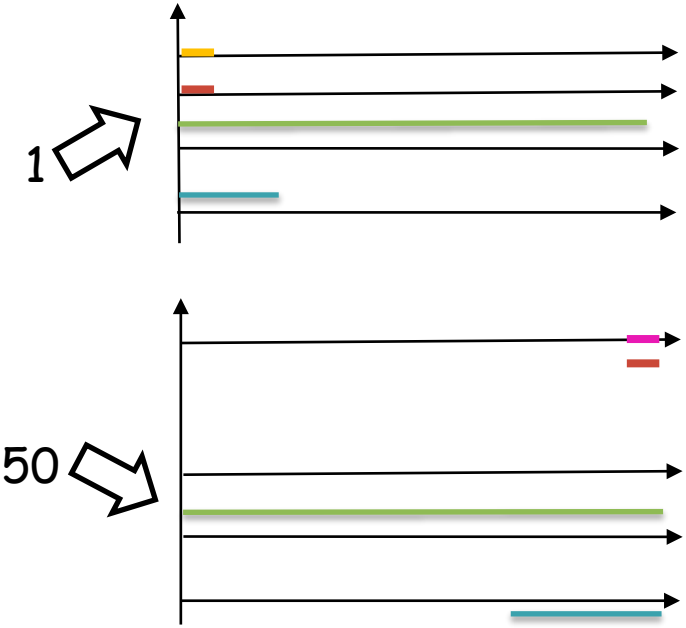


Log Capping

- For each time series, copy only the values within the ROIs, plus the values immediately before and after each ROI



(Mantid > 6.7)



..a work in progress

- Optimizations are underway to speed up the generation of TimeSplitter objects (**GenerateEventsFilter**) and the filtering+splitting of events (**FilterEvents**)