tf.contrib.data.sloppy_interleave

```
sloppy_interleave(
    map_func,
    cycle_length,
    block_length=1
)
```

Defined in `tensorflow/contrib/data/python/ops/sloppy_ops.py`.

A non-deterministic version of the `Dataset.interleave()` transformation.

`sloppy_interleave()` maps `map_func` across `dataset`, and non-deterministically interleaves the results.

The resulting dataset is almost identical to `interleave`. The key difference is that if retrieving a value from a given output iterator would cause `get_next` to block, that iterator will be skipped, and consumed when next available. If consuming from all iterators would cause the `get_next` call to block, the `get_next` call blocks until the first value is available.

If the underlying datasets produce elements as fast as they are consumed, the `sloppy_interleave` transformation behaves identically to `interleave`. However, if an underlying dataset would block the consumer, `sloppy_interleave` can violate the round-robin order (that `interleave` strictly obeys), producing an element from a different underlying dataset instead.

Example usage:

```
# Preprocess 4 files concurrently.
filenames = tf.data.Dataset.list_files("/path/to/data/train*.tfrecords")
dataset = filenames.apply(
    tf.contrib.data.sloppy_interleave(
        lambda filename: tf.data.TFRecordDataset(filename),
        cycle_length=4))
```

WARNING: The order of elements in the resulting dataset is not deterministic. Use `Dataset.interleave()` if you want the elements to have a deterministic order.

Args:

- `map_func`: A function mapping a nested structure of tensors (having shapes and types defined by `self.output_shapes` and `self.output_types`) to a `Dataset`.
- `cycle_length`: The number of threads to interleave from in parallel.
- `block_length`: The number of consecutive elements to pull from a thread before advancing to the next thread. Note: sloppy_interleave will skip the remainder of elements in the block_length in order to avoid blocking.

Returns:

A `Dataset` transformation function, which can be passed to `tf.data.Dataset.apply`.

Last updated November 2, 2017.