HBase.Advanced

API. Advanced

BufferedMutator

- Предназначен для асинхронных пакетных изменений таблиц
- BufferedMutator сам определяет момент синхронизации и размер пакета с данными
- Возможна потеря данных.

```
BufferedMutatorParams params = new BufferedMutatorParams(TableName.valueOf("testtable"));

BufferedMutator bufferedMutator = connection.getBufferedMutator(params);

bufferedMutator.mutate(put);

bufferedMutator.flush();

bufferedMutator.close();
```

Scan

- Аналог курсоров в базах данных.
- Создает итератор : ResultScanner
- Каждый вызов ResultScanner.next возвращает объект Result, содержащий отобранные данные строки
- Можно для повышения производительности сузить возвращаемую информацию

  Scan addFamily(byte [] family)

  Scan addColumn(byte[] family, byte[] qualifier)

  Scan setTimeRange(long minStamp, long maxStamp) throws IOException

  Scan setTimeStamp(long timestamp)

  Scan setMaxVersions()

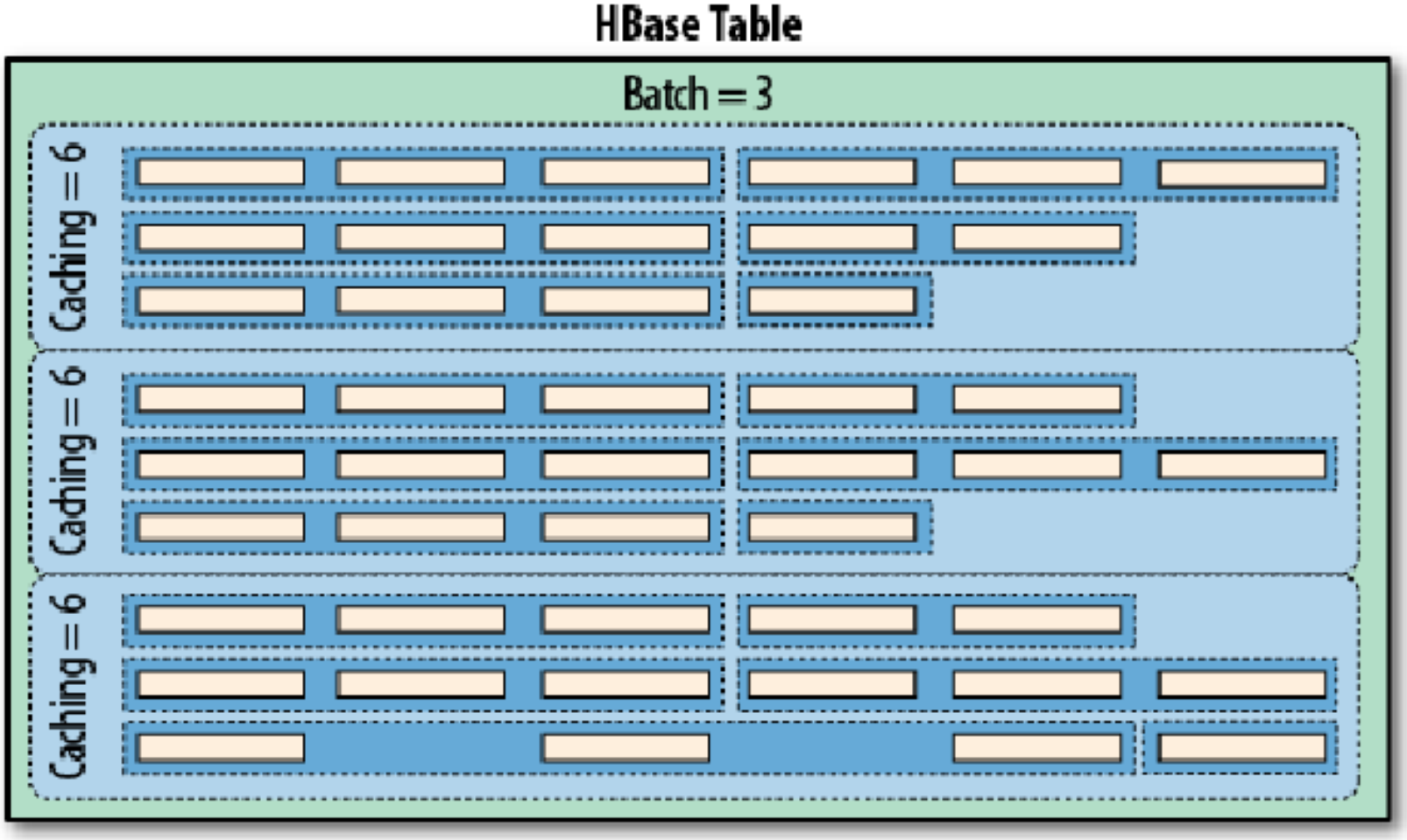
  Scan setMaxVersions(int maxVersions)
- Каждый сканер занимает часть ресурсов сервера, поэтому обязательно после использование требуется вызов close()
- Для повышения производительности основным методом является задание диапазона ключей

```
scan.setStartRow(Bytes.toBytes("row80"));
scan.setStopRow(Bytes.toBytes("row89"));
```

Scan.Caching.Batch

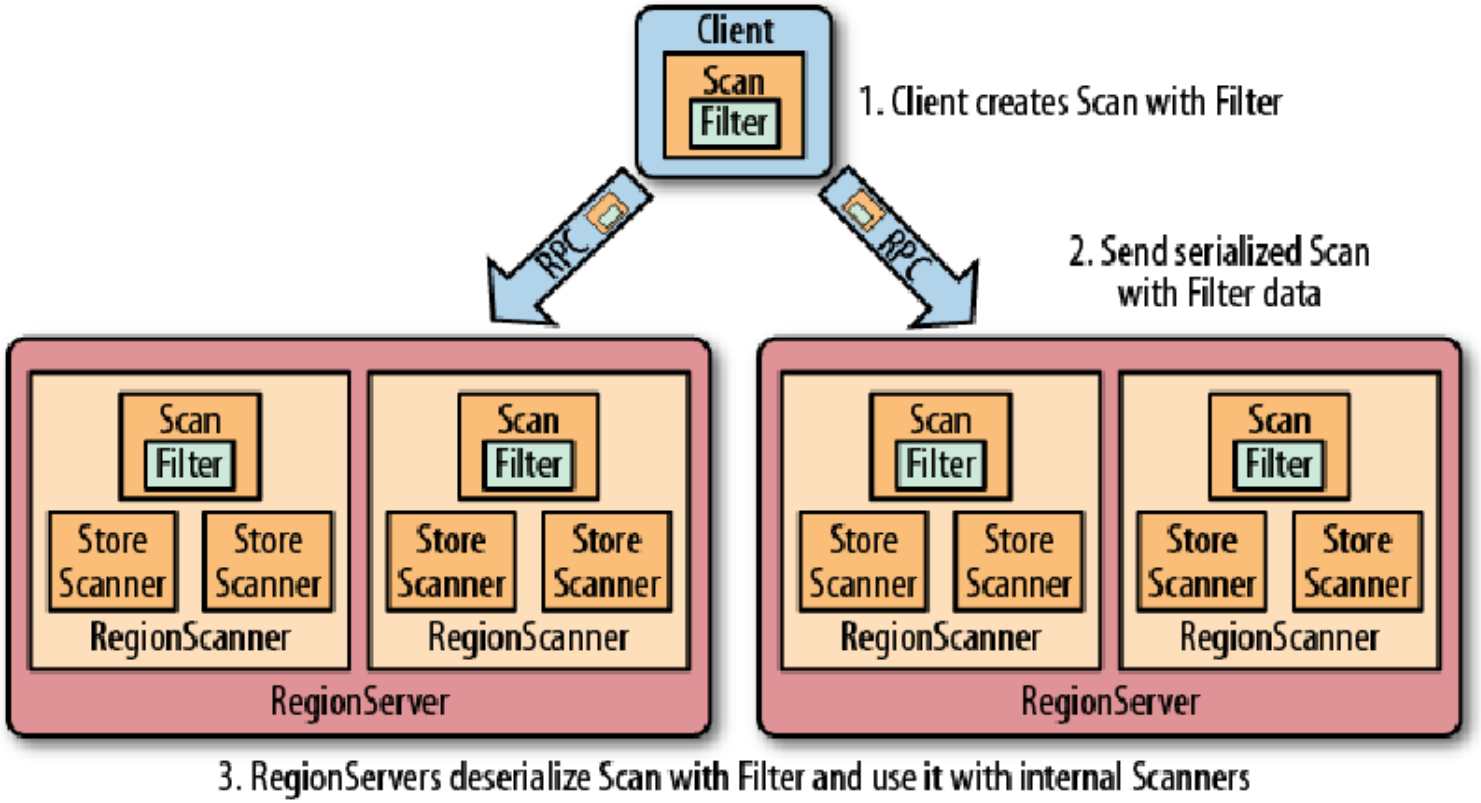
- SetCaching задает количество упреждающих запросов RPC к серверу
- Batch задает количество столбцов которое передается за один вызов.
- Комбинация setCaching и setBatch задает порядок выборки данных с сервера.

Пример использования Caching+ Batch



Filter

- Предназначены для предварительной фильтрации информации на стороне сервера.



Пример работы с фильтром

```
Scan scan = new Scan();
scan.addColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("col-0"));
Filter filter1 = new
RowFilter(CompareFilter.CompareOp.LESS_OR_EQUAL,
                new BinaryComparator(Bytes.toBytes("row-22")));
scan.setFilter(filter1);
ResultScanner scanner1 = table.getScanner(scan);
for (Result res : scanner1) {
    System.out.println(res);
}
scanner1.close();
```

CompareFilter

- Проверяют row/column family/column/value на соответствие условию

  CompareFilter(CompareOp valueCompareOp, ByteArrayComparable valueComparator)

- Операторы :

  LESS,LESS_OR_EQUAL,EQUAL,NOT_EQUAL,GREATER_OR_EQUAL,GREATER, NO_OP

- Сравниваемые значения :

  BinaryComparator,BinaryPrefixComparator.NullComparator,BitComparator,RegexStringComparator, SubstringComparator

- Фильтры :
  - RowFilter,FamilyFilter,QualifierFilter,ValueFilter,DependentColumnFilter

Специальные фильтры

- SingleColumnValueFilter – пропускает записи если в них есть столбец удовлетворяющий условию
- PrefixFilter – пропускает записи row key которых имеет заданный префикс
- KeyOnlyFilter – возвращает только Key из каждого KeyValue
- FirstKeyOnlyFilter – возвращает только key первого столбца строки
- И т.д.

Нестандартные фильтры

- Для написания фильтра надо реализовать интерфейс org.apache.hadoop.hbase.filter.Filter
- Основные методы

  reset - сбрасывает фильтр

  filterAllRemaining - true *остановить сканирование*

  filterRowKey(Cell firstRowCell) – true если требутеся убрать строку из результата

  filterKeyValue(Cell v) - вызывается для каждого KeyValue, можем вернуть команду что делать дальше(пропуск строки, добавить KeyValue и т.д.)

  transformCell(Cell v) – Преобразовать KeyValue в другое значение

  filterRowCells(List<Cell> kvs)– можем изменить значения столбцов

  filterRow - пропустить строку на основании всех значений стобцов

Пример простого фильтра

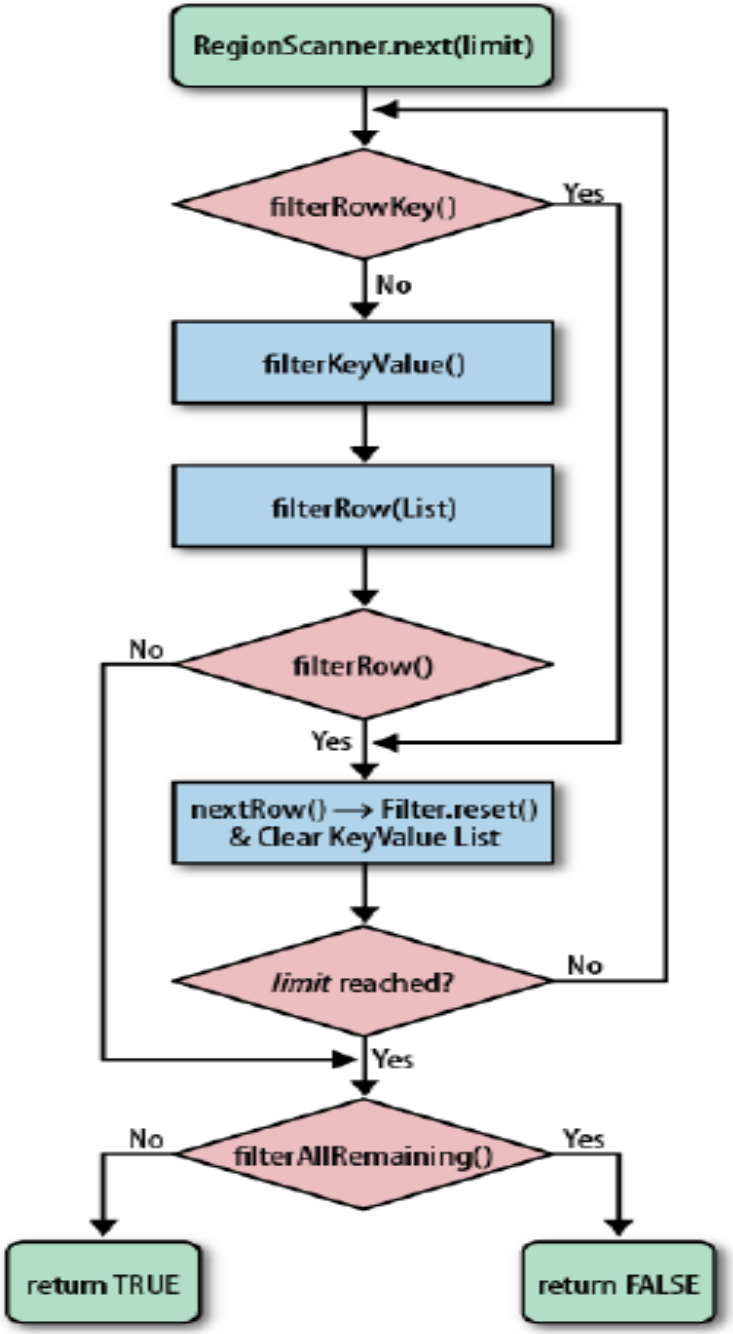
```
public class CustomFilter  extends FilterBase{
    String selectRowKey;
    public CustomFilter() {
        super();
    }
    public CustomFilter(String selectRowKey) {
        this.selectRowKey = selectRowKey;
    }
    @Override
    public boolean filterRowKey(byte[] buffer, int offset, int length) throws IOException {
        String key = new String(buffer, offset, length);
        return !key.equals(selectRowKey);
    }
@Override
    public byte[] toByteArray() throws IOException {
        return selectRowKey.getBytes("UTF-8");
    }
@Override
    public ReturnCode filterKeyValue(Cell cell) throws IOException {
        return ReturnCode.INCLUDE;
    }
    public static Filter parseFrom(byte[] pbBytes) throws DeserializationException {
        try {
            System.out.println("create filter!");
            return new CustomFilter(new String(pbBytes, "UTF-8"));
        } catch (UnsupportedEncodingException e) {
            throw new DeserializationException(e);
        }
    }
}
```

Пример использования

```
Scan scan = new Scan();
scan.addFamily("colfam1".getBytes());
scan.setFilter(new CustomFilter( "rawkey1"));

ResultScanner scanner = table.getScanner(scan);
for (Result res : scanner) {
    System.out.println("row=" + res);
}
```

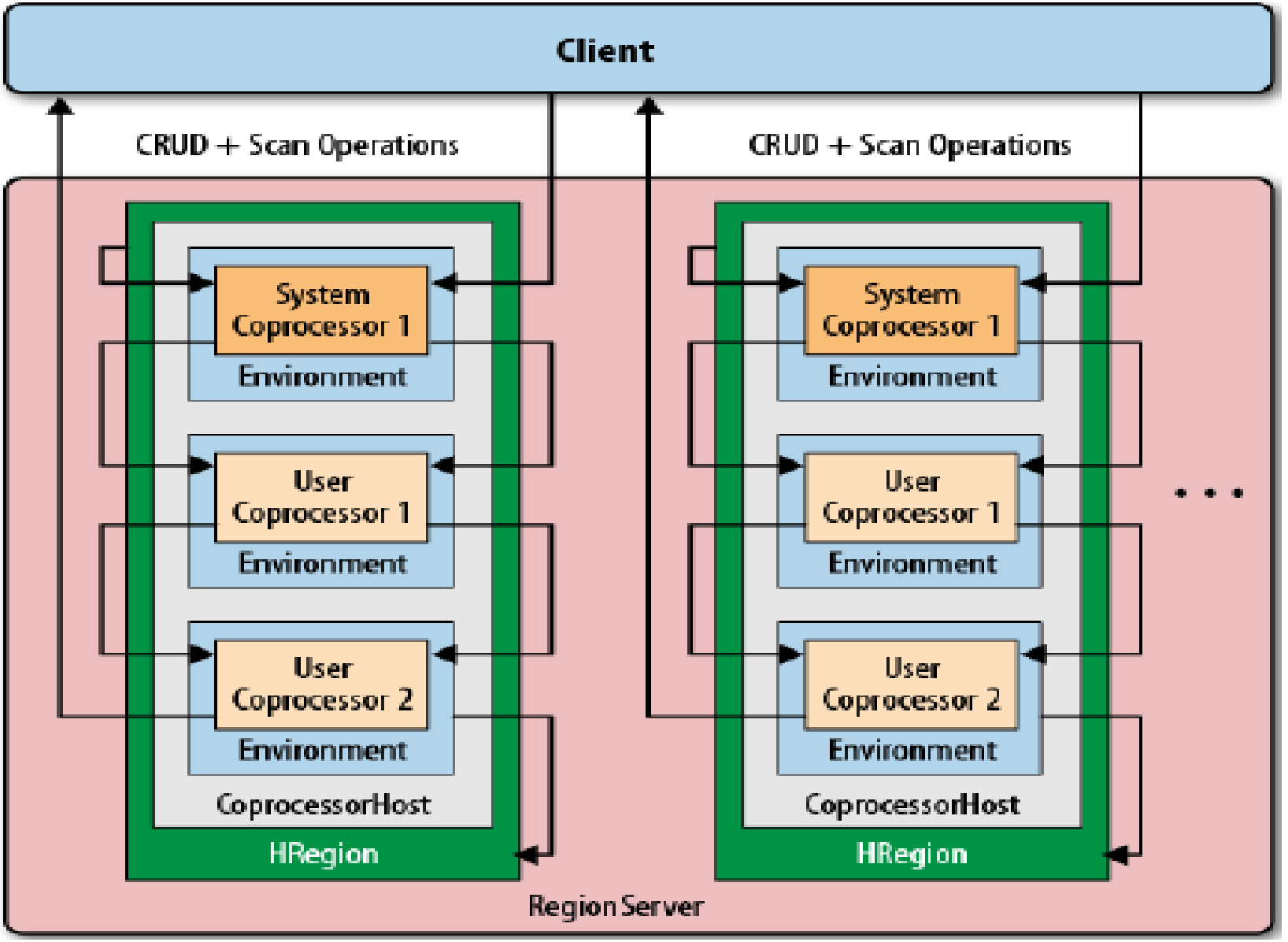
Последовательность вызовов



Coprocessor Observer

- Аналогичны триггерам RDBMS
- Классы размещены на сервере HBase и при выполнении какого-либо события встраиваются в ход выполнения
- Типы observer-ов :
  - RegionObserver, обрабатывает события связанные с данными
  - MasterObserver, обрабатывает события свзяанные с DDL
  - WALObserver, работает с Write Ahead Log

Работа Coprocessor



Разработка RegionObserver

- Расширяем класс RegionObserver
- Реализуем требуемый метод preXX или postXX
- Для доступа к Hbase используется параметр
  - ObserverContext<RegionCoprocessorEnvironment> e)
- Регистрирем RegionObserver либо в настройках hbase (hbase-site.xml) либо в метаданных таблицы

Пример RegionObserver

```
public class RegionObserverExample extends BaseRegionObserver {
    public static final byte[] FIXED_ROW = Bytes.toBytes("@@@GETTIME@@@");
    @Override
  public void preGetOp(
     ObserverContext<RegionCoprocessorEnvironment> e,
     Get get,
     List<Cell> results
    ) throws IOException {
        if (Bytes.equals(get.getRow(), FIXED_ROW)) {
            KeyValue kv = new KeyValue(get.getRow(), FIXED_ROW, FIXED_ROW,
                    Bytes.toBytes(System.currentTimeMillis()));
            results.add(kv);
        }
    }
}
```