Дзен

Off-heap вместо трех JEP'ов

Дмитрий Погорелов, руководитель инфраструктуры рекомендательной системы Дзена
@pogorelych

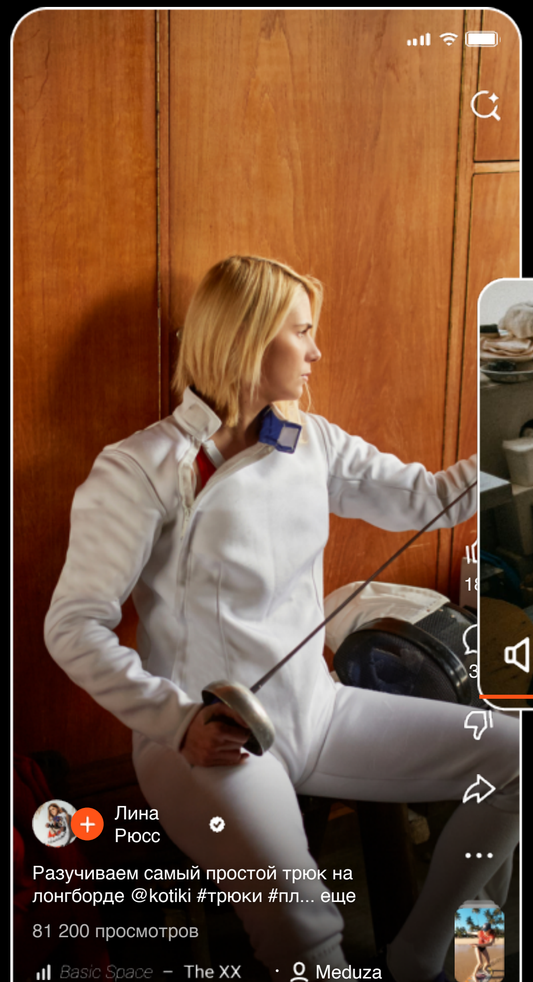
Зачем C++ программист перешел на другую сторону улицы?
Чтобы собрать мусор!

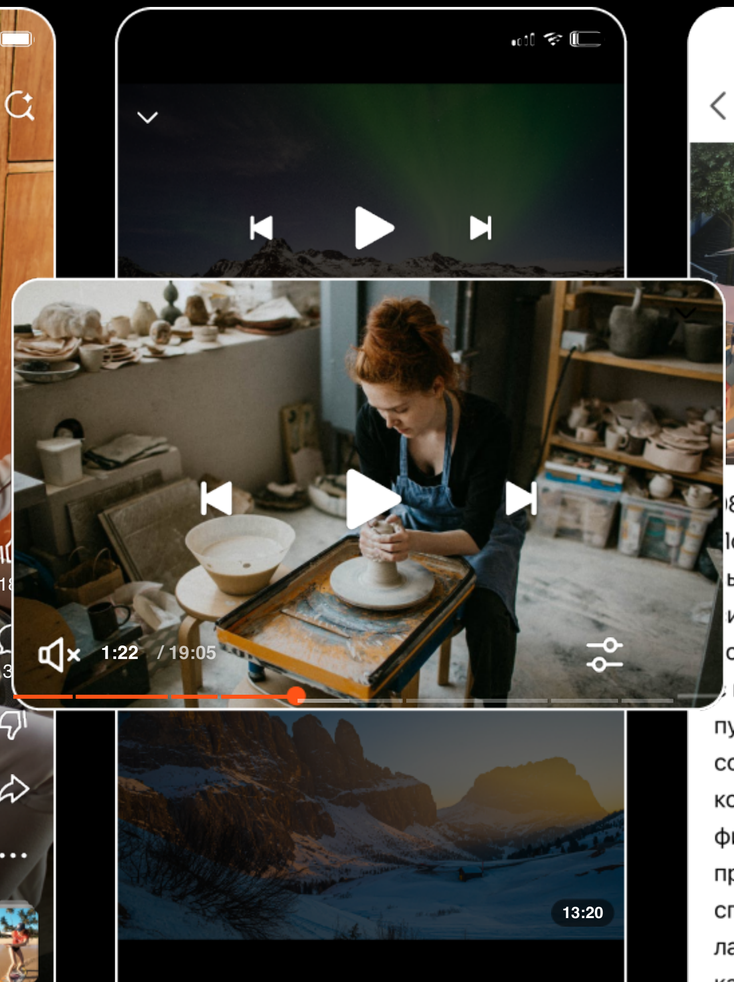
Дзен — это большая контентная платформа

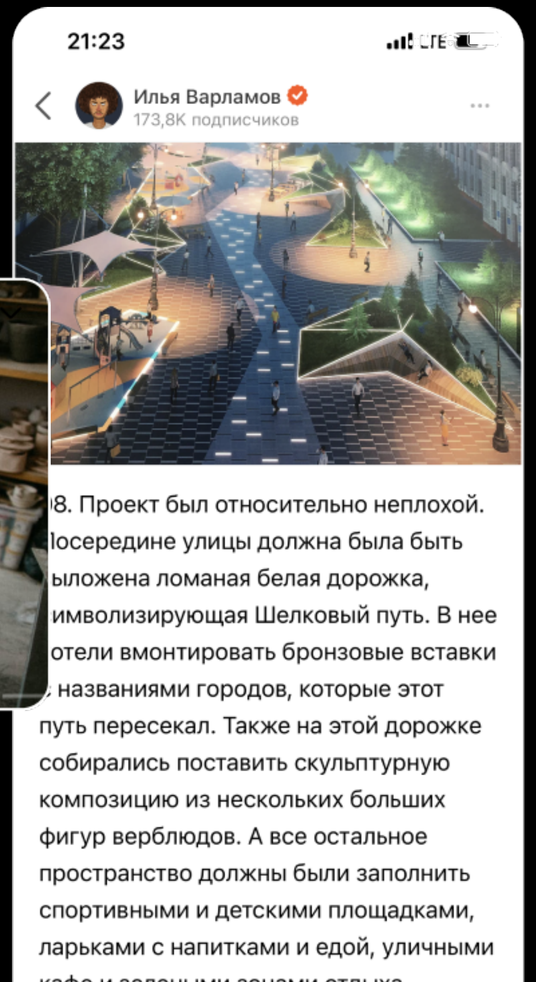
Вертикальные короткие ролики

Длинные видео

Тексты: лонгриды и посты







Дзен

Дзен в цифрах

DAU 30M+
пользователей

MAU 70M+
пользователей

100K+
активных авторов

150K
RPS

100+
микросервисов

9999
отказоустойчивость

В сердце Дзена — рекомендательная система

Рекомендации работают в реальном времени

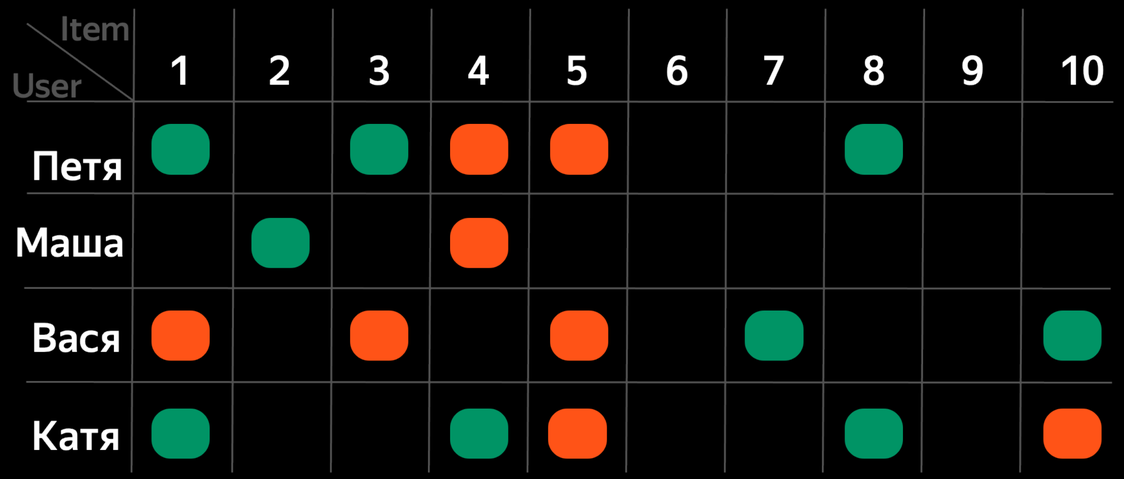
- 10 000 RPS
- 300 ms в 99_prc

10 млн документов

На запрос отскорим 50 тыс документов

Дзен

… и много математики



≈

-0.82	-0.47	-1.05
-0.82	0.00	-0.00
-0.00	-1.28	0.77
0.82	-0.47	-1.05

×

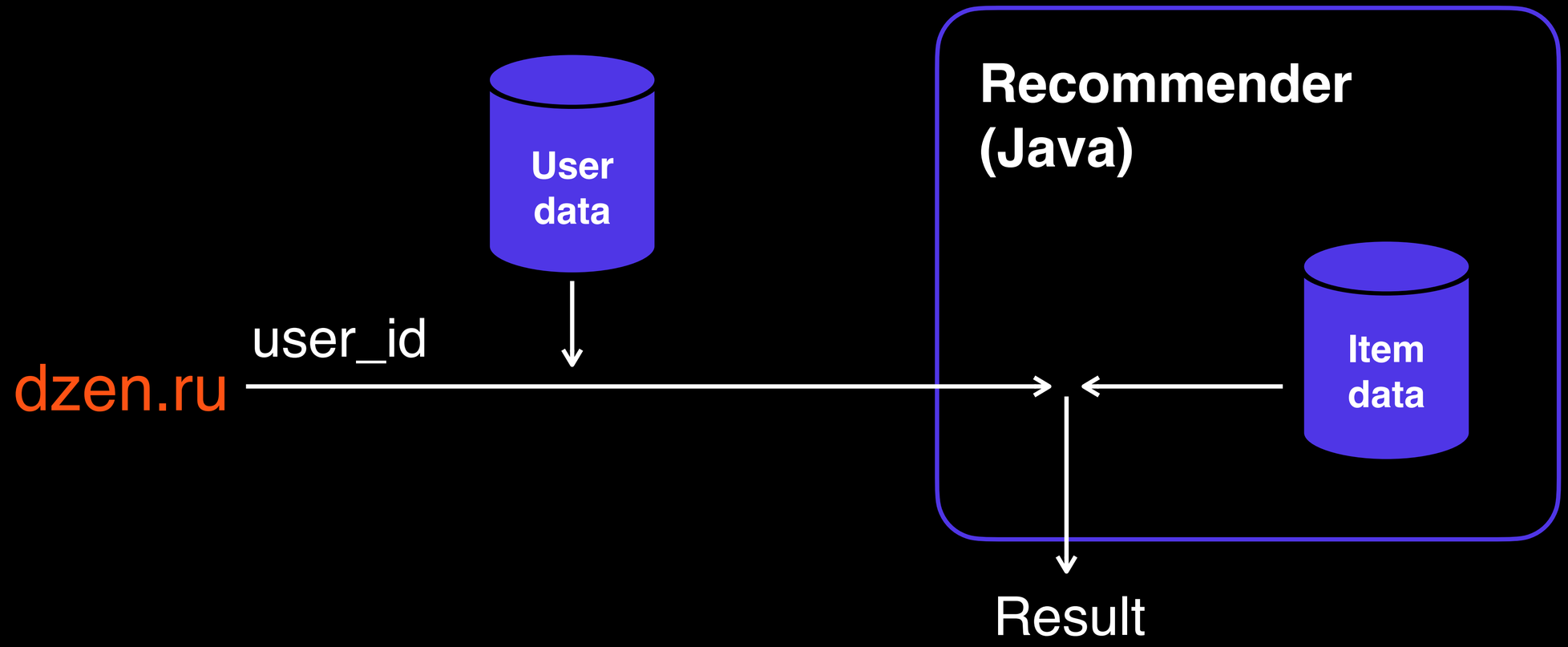
0.00	-0.41	-0.41	1.22	-0.00	0.00	0.00	0.00	0.00	-0.41
0.17	-0.00	0.39	0.00	1.07	0.00	-0.62	-0.45	0.00	-0.39
-1.03	0.00	-0.65	0.00	0.48	0.00	0.28	-0.75	0.00	0.65

=

Item / User	1	2	3	4	5	6	7	8	9	10
Петя	1.00	0.33	0.83	-1.00	-1.00	0.00	0.00	1.00	0.00	-0.17
Маша	-0.00	0.33	0.33	-1.00	0.00	0.00	-0.00	-0.00	0.00	0.33
Вася	-1.00	-0.00	-1.00	0.00	-1.00	0.00	1.00	0.00	0.00	1.00
Катя	1.00	-0.33	0.17	1.00	-1.00	0.00	0.00	1.00	0.00	-0.83

Архитектура

Храним документы там же, где и ранжируем



Вызовы перед инфраструктурой

40+ гб

данных на один под

5 млрд

умножений векторов в секунду

Рецепт ранжирования


```
1  float[] user = {1f, 0f, 0.28f, .....}
2  float[] item = {0.3f, 1f, 0.121f, .....}
3
4  float feature_i = dotProduct(user, item)
5
6  float score(user,item) = matrixnet(float[] features)
7
8  float dotProduct(float[] a, float[] b) {
9      float sum = 0;
10     for (int i = 0; i < a.length; i++) {
11         sum += a[i] * b[i];
12     }
13     return sum;
14 }
```

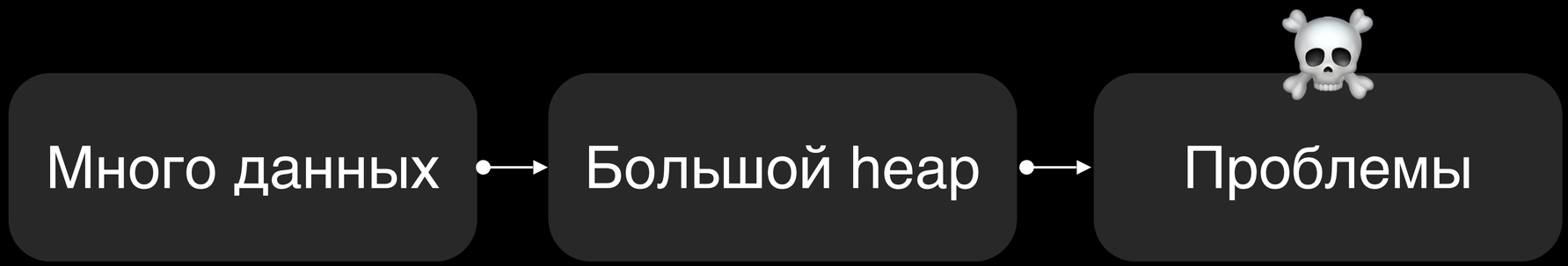
Project Panama,
JEP-424: Foreign Function & Memory API

SIMD, AVX

JEP-448: Vector API

В off-heap проще


```
1  float*  user = 0x434234
2  float*  item =  0x434234
3  float feature = Eigen_JNI.dotProduct();
4  score = matrixnet(float*)
```



- -XX:UseCompressedOops не работают
- Медленный GC
- Долгий старт

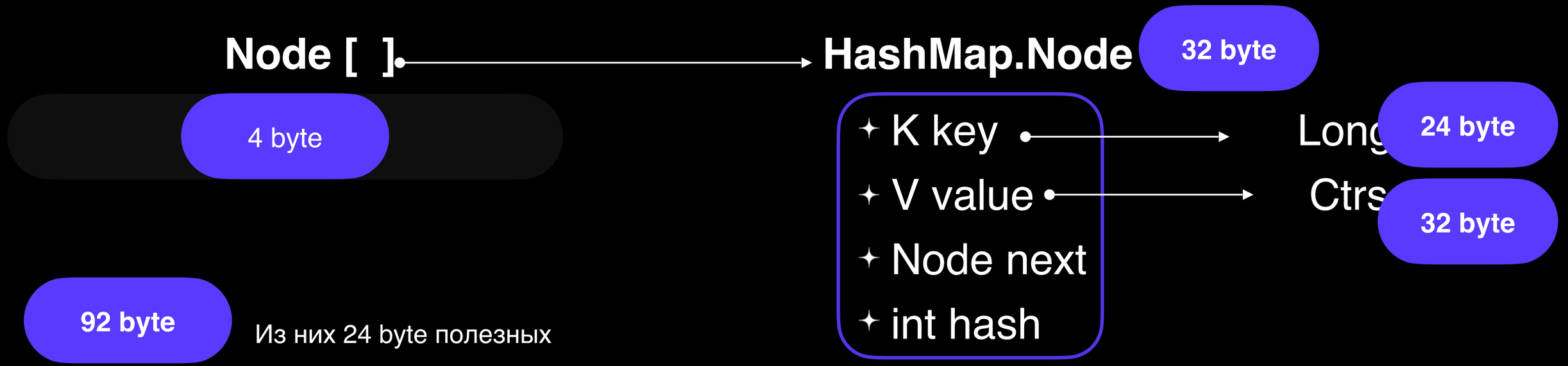
Off-heap коллекции

Не только вектора


```
1  Map<Long, Ctrs> ctrs;
2  record Ctrs (long clicks, long shows) {}
```

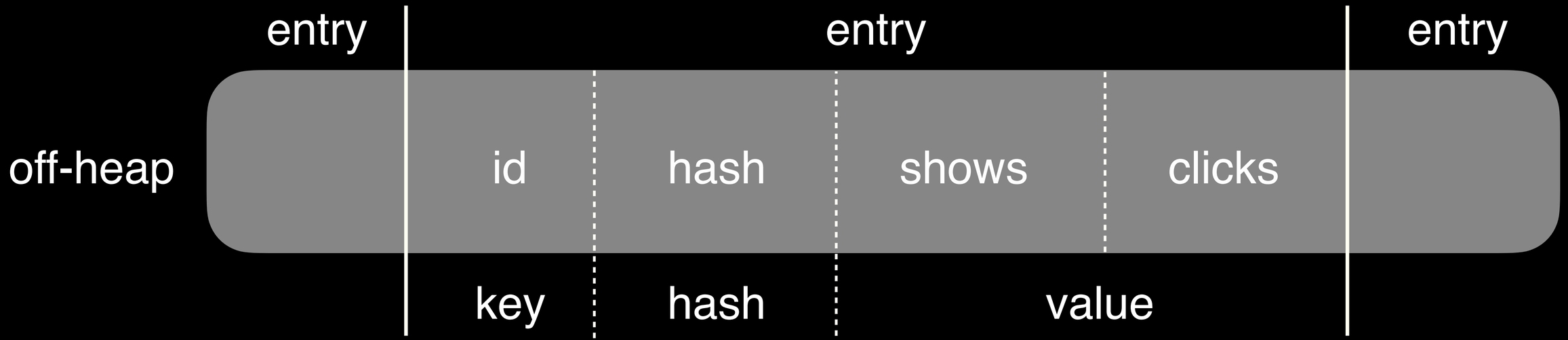
java.util.HashMap


```
1  Map<Long, Ctrs> ctrs = new HashMap ()
```



JEP 401: Value Classes and Objects

OffheapOpenHashMap



28 byte

Из них 24 byte полезных

OffheapOpenHashMap


```
1 V get (K key) {
2     long addr = findPosition(key);
3     V result = valueMemManager.get(addr);
4 }
```

OffheapOpenHashMap.get(key) —> аллокация

OffheapOpenHashMap

- Маленький heap —> проще GC
- Плотно пакуем данные без overhead на заголовки
- Большие коллекции (больше 2 гб)
- Локальность данных

Мусорим при обращении

One-nio

implementation 'ru.odnoklassniki:one-nio:1.7.1'

github.com/odnoklassniki/one-nio

Свои велосипеды

- float[] храним с выравниванием
- Ленивая десериализация Map<Long, Map<Long, …>>
- Коллекции примитивов

Коллекции примитивов

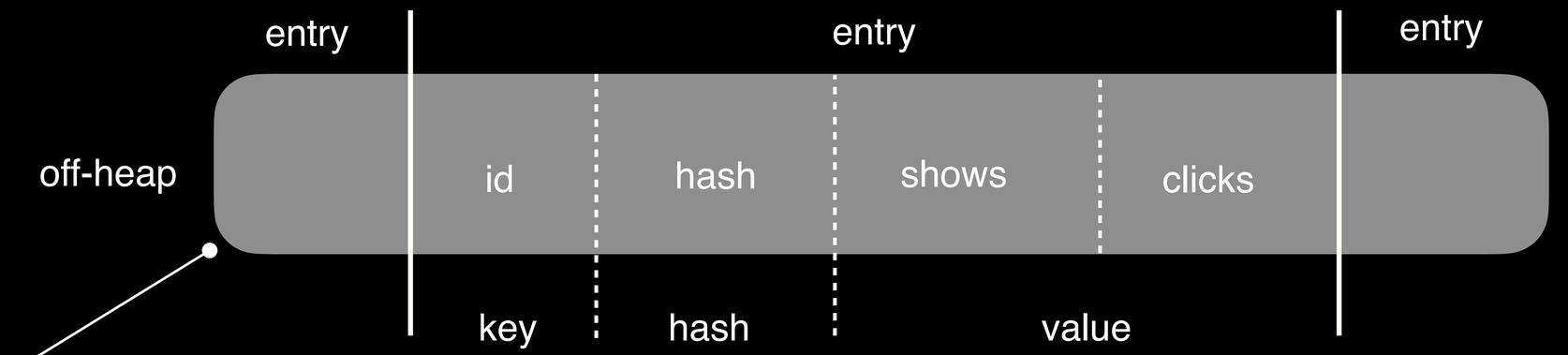

```
1  LongLongHashMap
2
3  public long get(long key) {}
4
5  public void put(long key, long value) {}
```

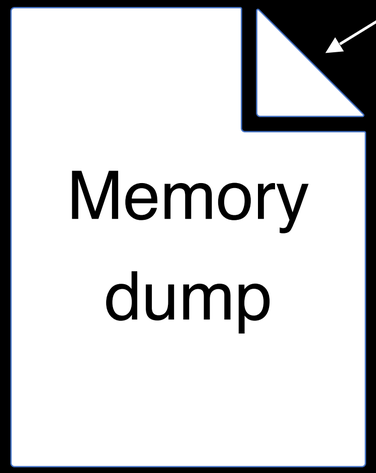
Коллекции примитивов

Практическое применение

id документа	→	id автора
id документа	→	статус модерации
id документа	→	список похожих

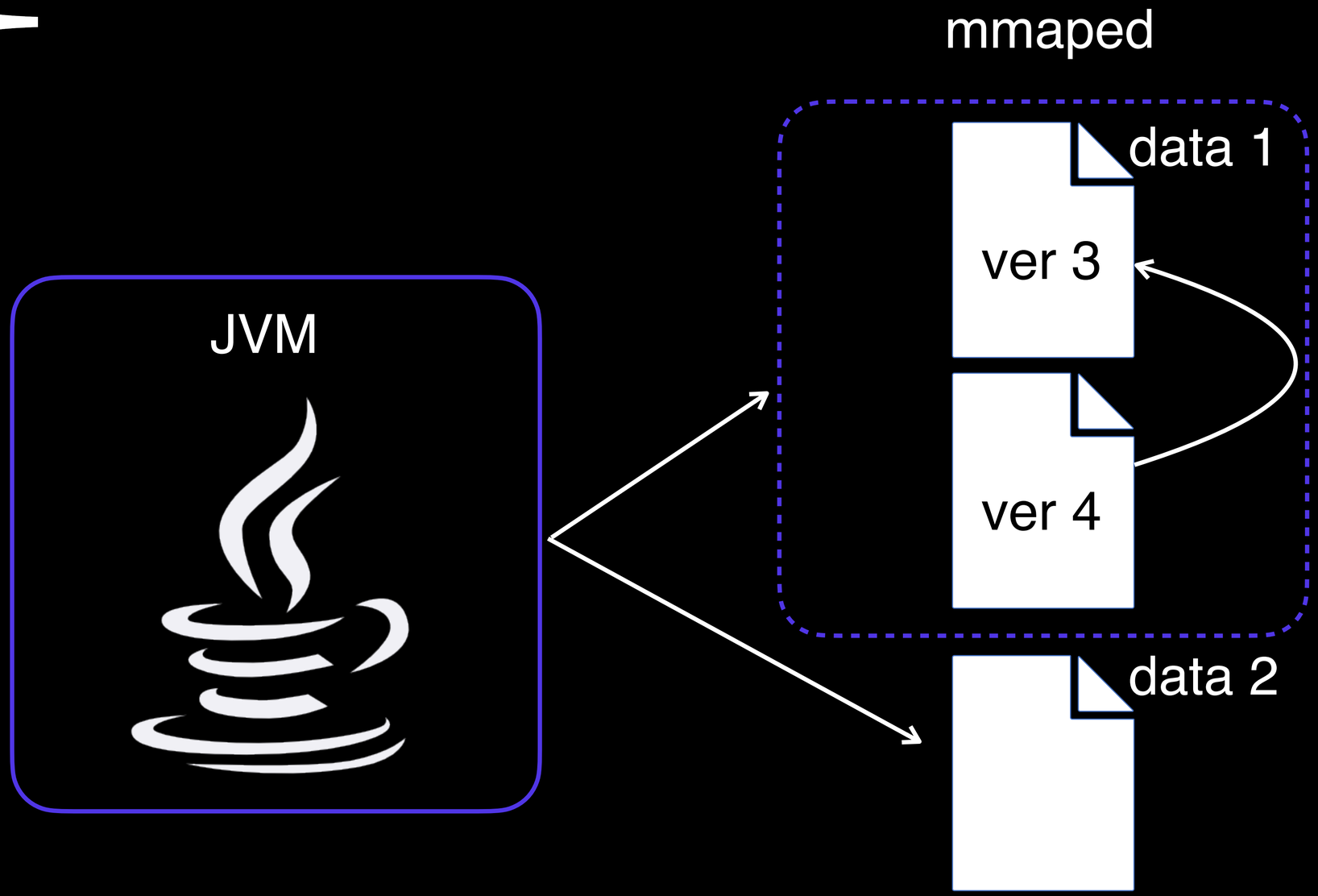
Dump памяти = бесплатная сериализация



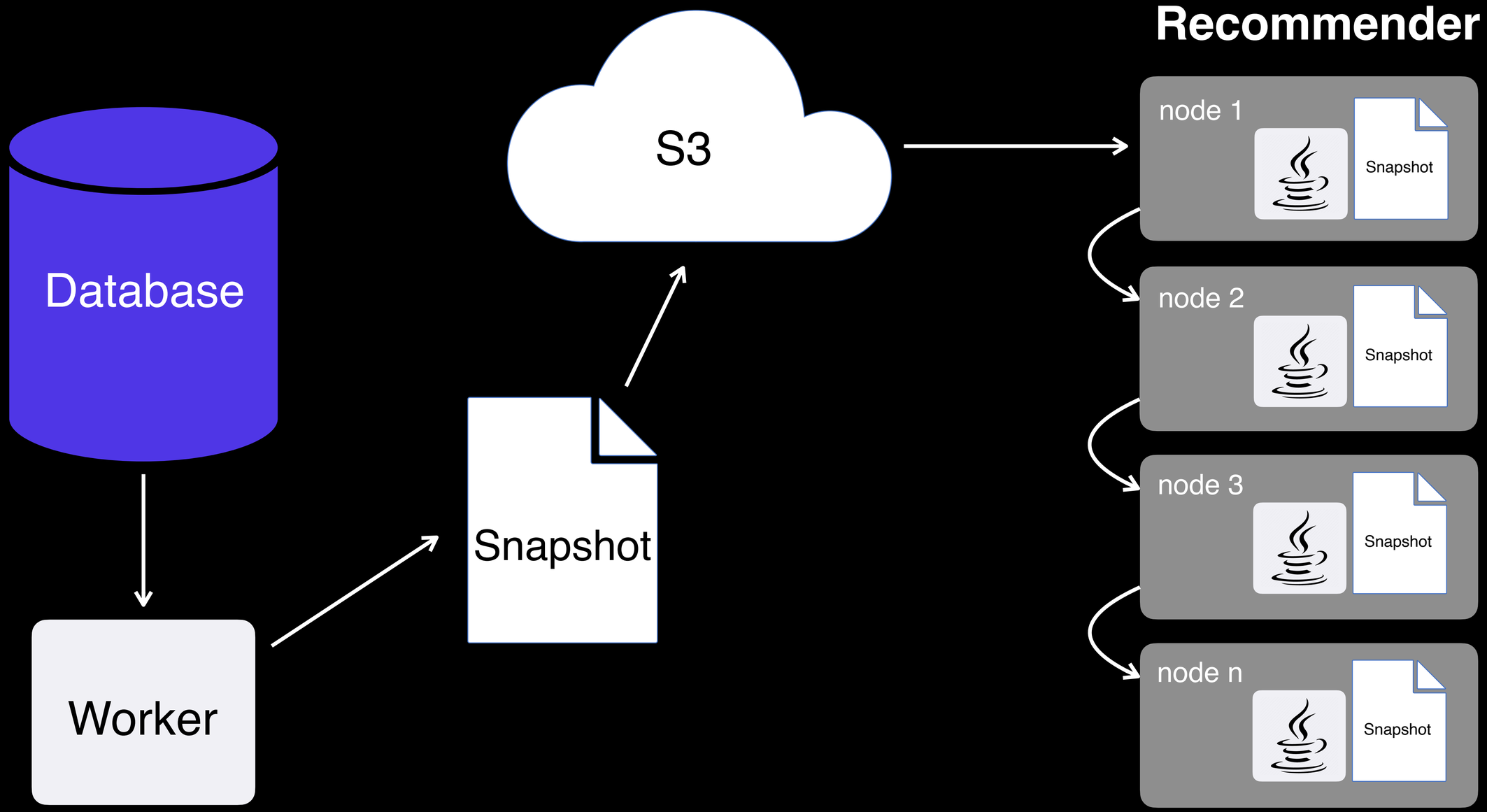


Дзен

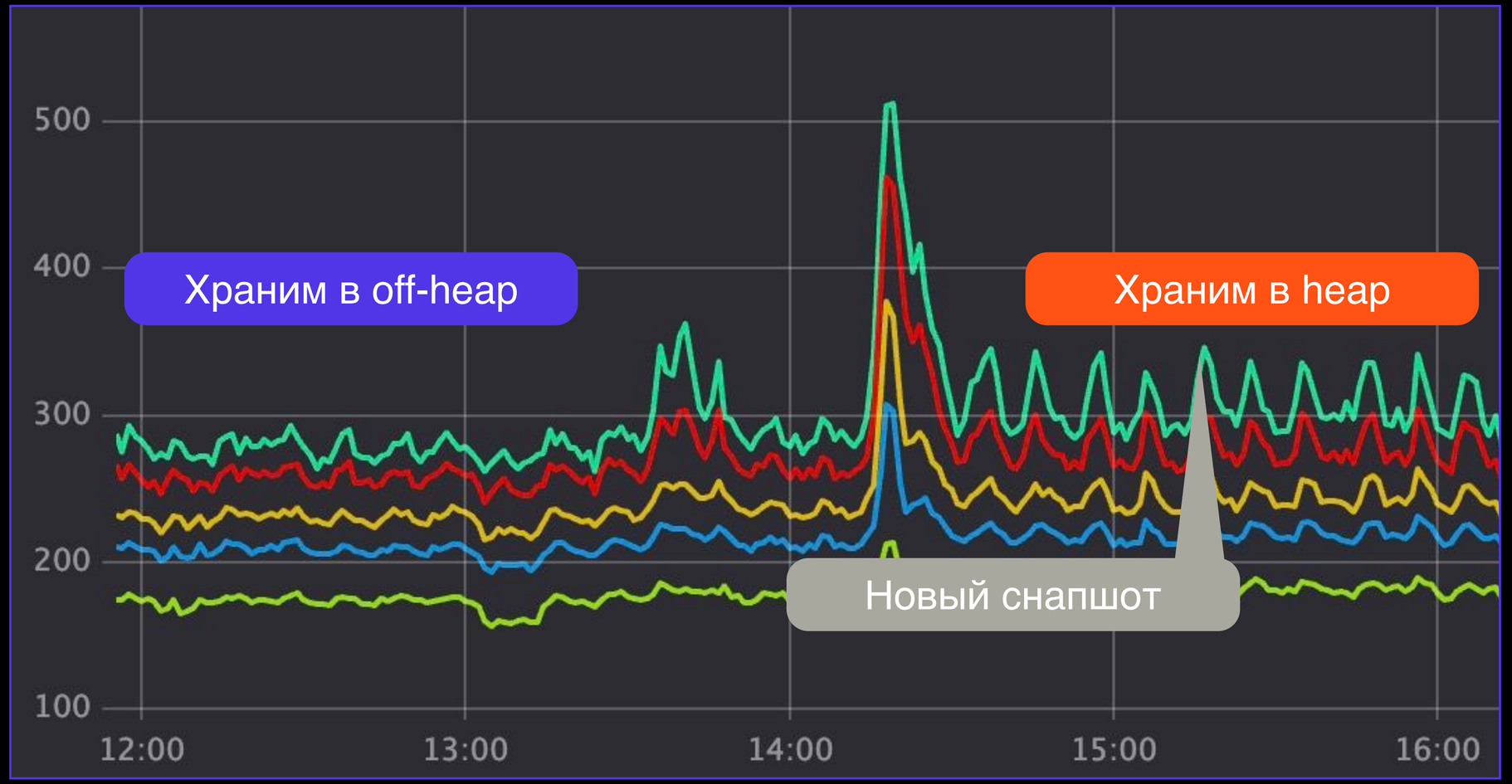
Снапшоты данных



Снапшоты на кластере



Переложили 3 гб в heap



Технология снапшотов

- Бесплатно загружаем на нодах
- Моментальный рестарт с горячим кешом
- 40+ гб на 2 тыс нод за 15 минут…
  или быстрее, упираемся только в сеть

Как работать с off-heap

Добываем off-heap


```
long addr = Unsafe.allocateMemory(size)

ByteBuffer buffer = ByteBuffer.allocateDirect(…)

MappedByteBuffer javammaped = file.getChannel().map(…)

MMapBuffer buf = new MMapBuffer(file, …)
```

Добываем Unsafe


```java
public static final sun.misc.Unsafe UNSAFE = initializeUnsafe();

private static sun.misc.Unsafe initializeUnsafe() {
    try {
        Field field = sun.misc.Unsafe.class.getDeclaredField("theUnsafe");
        field.setAccessible(true);
        return (sun.misc.Unsafe) field.get(null);
    } catch (NoSuchFieldException | IllegalAccessException e) {
        throw new Error(e);
    }
}
```

Unsafe


```java
long addr = UNSAFE.allocateMemory(size);
UNSAFE.setMemory(addr, size, (byte) 0);
UNSAFE.copyMemory(…);
UNSAFE.getLong(addr);
UNSAFE.putByte(addr);
UNSAFE.freeMemory(addr);
```

DirectByteBuffer


```
1  int size = …;
2  ByteBuffer buffer = ByteBuffer.allocateDirect(size);
3  byte a = buffer.get(index);
4  long b = buffer.getLong(index);
5  buffer.putLong(index, value);
```

MappedByteBuffer


```
1  RandomAccessFile file = new RandomAccessFile("file","rw");
2
3  MappedByteBuffer javammaped =
4          file.getChannel().map(FileChannel.MapMode.READ_WRITE, 0, f.length());
5
6  javammaped.force();
```

DirectByteBuffer
MappedByteBuffer

Подчищаются GC

Не больше 2 гб

XX:MaxDirectMemory


```
-XX:MaxDirectMemorySize
-XX:MaxRAM

Exception in thread "main" java.lang.OutOfMemoryError: Cannot reserve 200000000 bytes of direct buffer memory
 at java.base/java.nio.Bits.reserveMemory(Bits.java:178)
 at java.base/java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:121)
 at java.base/java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:332)
 at Main.main(Main.java:42)
```

XX:MaxDirectMemory

MaxRam = Xmx + MaxMetaspaceSize +
ReservedCodeCacheSize + MaxDirectMemorySize

-XX:MaxRam -XX:MaxDirectMemory только для DirectByteBuffer

XX:MaxDirectMemory

-XX:MaxDirectMemorySize=1m
-XX:MaxRAM=100m

```
long size = 1000_000_000L;
long addr = unsafe.allocateMemory(size);
RandomAccessFile f = new RandomAccessFile(“my_200mb_file","rw");
MappedByteBuffer javammaped =
        f.getChannel().map(FileChannel.MapMode.READ_WRITE, 0, f.length());
```

com.indeed:util-mmap


```
implementation 'com.indeed:util-mmap:1.0.52-3042601'

MMapBuffer buf = new MMapBuffer(file, FileChannel.MapMode.READ_ONLY
buf.mlock(0, bufferLength);
long addr = buf.memory().getAddress()
```

com.indeed:util-mmap

- Файлы любого размера
- mlock

Troubleshooting


```
java.lang.OutOfMemoryError: null
  at com.indeed.util.mmap.NativeMemoryUtils.mlock(NativeMemoryUtils.java:58)
  at com.indeed.util.mmap.MMapBuffer.mlock(MMapBuffer.java:186) ~[util-mmap-1
```

Troubleshooting

Ставим капы на Java
setcap cap_ipc_lock=epi $JAVA_BINARY

Получаем ошибку
java: error while loading shared libraries: libjli.so: cannot open shared object file: No such file or directory

Решаем
echo "$JAVA_HOME/lib" > /etc/ld.so.conf.d/java.conf
ldconfig

unix.stackexchange.com/a/616511

Troubleshooting

Что делать в k8s?

k8s использует rlimit с ноды, на которой запущен контейнер, и это не конфигурируется для контейнеров

unix.stackexchange.com/a/616511


Off-heap
GC

Он вам не Exception


```
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x0000000102ac9260, pid=20968, tid=8707
#
# JRE version: OpenJDK Runtime Environment Corretto-17.0.6.10.1 (17.0.6+10) (build 17.0.6+10-LTS)
# Java VM: OpenJDK 64-Bit Server VM Corretto-17.0.6.10.1 (17.0.6+10-LTS, mixed mode, sharing, tiered,
# Problematic frame:
# V  [libjvm.dylib+0x9d5260]  Unsafe_PutLong(JNIEnv_*, _jobject*, _jobject*, long, long)+0x118
#
# No core dump will be written. Core dumps have been disabled. To enable core dumping, try "ulimit -c
#
# If you would like to submit a bug report, please visit:
#   https://github.com/corretto/corretto-17/issues/
```

-XX:ErrorFile=/var/log/my_app/java_error%p.log

Try-with-resources


```
1  public class MyOffheapObject
2                      implements Closeable {
3
4      private final long addr;
5
6      public MyOffheapObject() {
7          this.addr =  UNSAFE.allocateMemory(size);
8      }
9
10     @Override
11     public void close() {
12         UnsafeUtils.UNSAFE.freeMemory(addr);
13     }
14 }
15 public static void main(String[] args) {
16         try (var obj = new MyOffheapObject()) {
17             //do some stuff
18         }
19     }
```

Try-with-resources — не лучший вариант, если…

- Объекты в глобальном контексте
- Хотим спрятать за интерфейс
- Лень про это помнить

~~Finalize —> fanthom ref —>~~ JDK9

JDK9 Cleaner


```java
public class MyOffheapObject implements AutoCloseable {
    private final static Cleaner cleaner = Cleaner.create();

    private final long addr;
    private final Cleaner.Cleanable cleanable;

    public MyOffheapObject(long size) {
        this.addr = UNSAFE.allocateMemory(size);
        this.cleanable = cleaner.register(
                    this, new Deallocator(addr));
    }

    @Override
    public void close()  {
        cleanable.clean();
    }
}
```

```java
private static class Deallocator implements Runnable {
        final long addr;

        private Deallocator(long addr) {
            this.addr = addr;
        }

        @Override
        public void run() {
            UnsafeUtils.UNSAFE.freeMemory(addr);
        }
    }
```

JDK9 Cleaner…

… но GC ничего не знает про off-heap

Try-with-resources


```
try (resource1 = ... ; resource2 = ...) {
    ResourceContext context = new ResourceContext(resource1, resource2, ...)
    ...

    doSomeStuff(context);

    ...
    doAnotherStuff(context);
    ...
}
```

“Отличный” способ получить связность в коде

Try-with-resources, но волшебный


```
1  try () {
2      var resource1 = ...;
3      ....
4      doSomeStuff();
5      ....
6
7  }
8
9  void doSomeStuff() {
10
11     var resource2 = ...;
```

Прикапываем созданное в thread-local или spring request context

Вывод

Велосипед уместен, если он едет в нужную сторону :)

- Быстро работает математика
- Быстро доезжают данные
- Помещается большая база

Дзен

Спасибо! Задавайте вопросы

Дмитрий Погорелов, руководитель инфраструктуры рекомендательной системы Дзена
@pogorelych