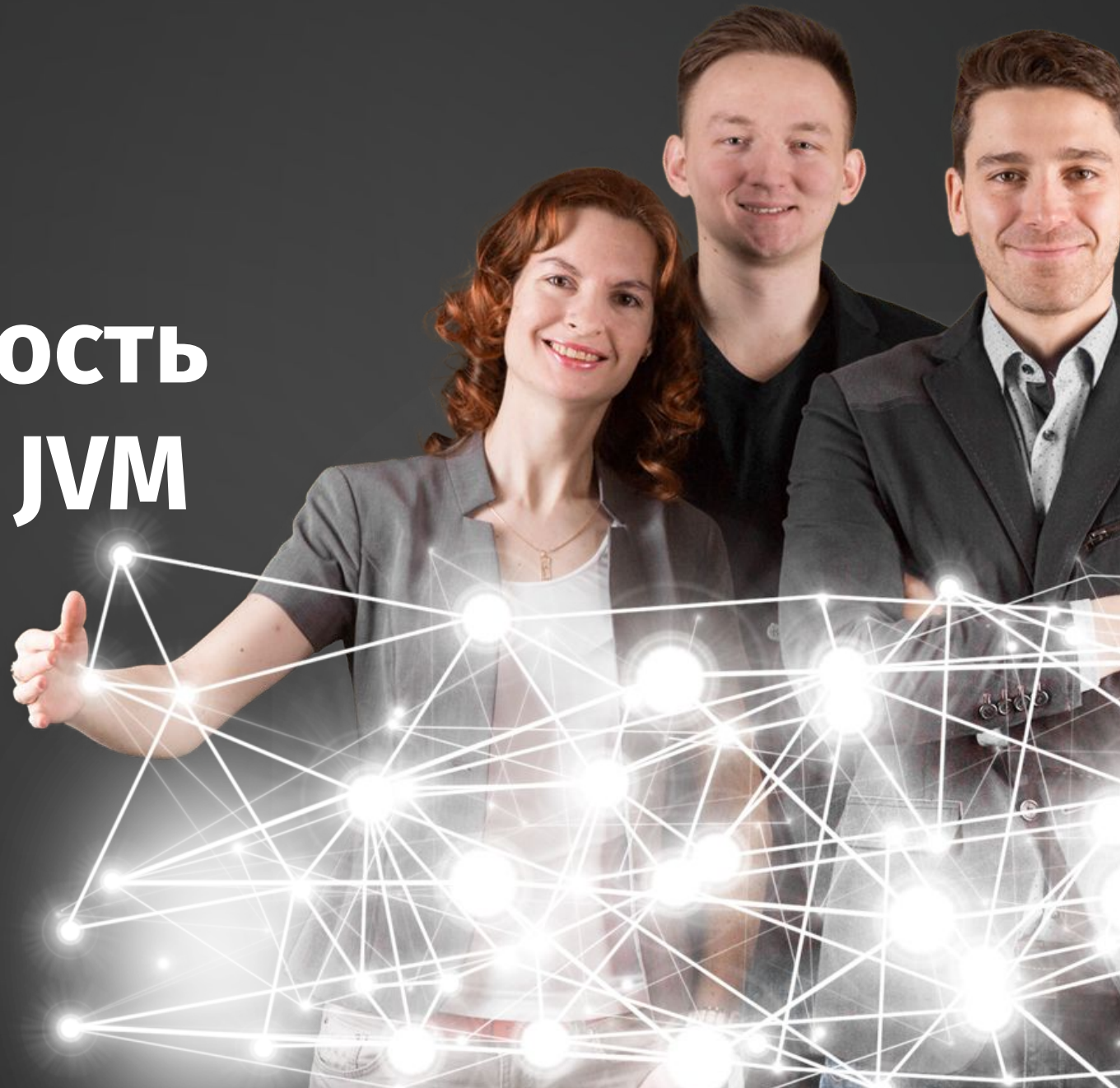
Grid Dynamics

Повышаем производительность файлового I/O в JVM на Linux

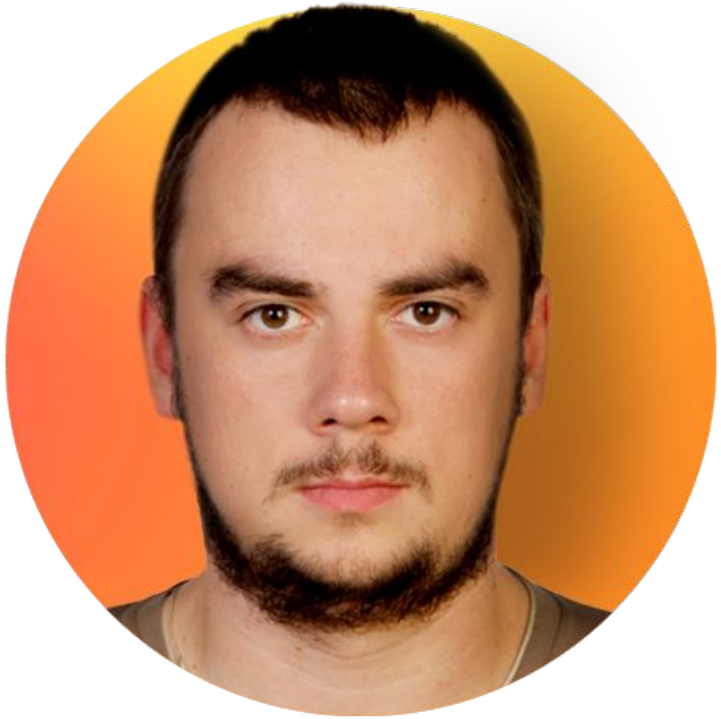


Дмитрий Бундин

Апрель 2020

Privileged and confidential

О спикере

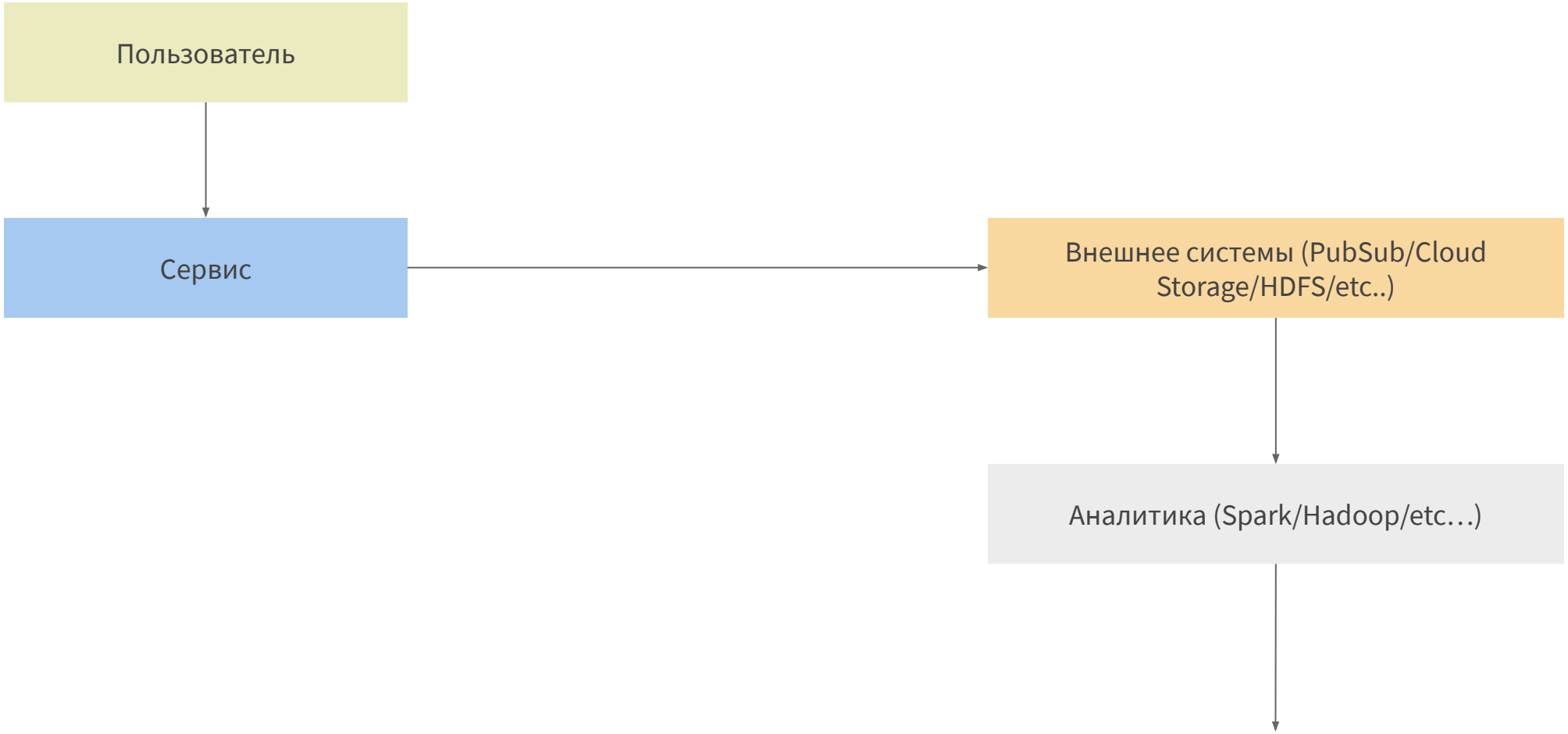


Дмитрий Бундин

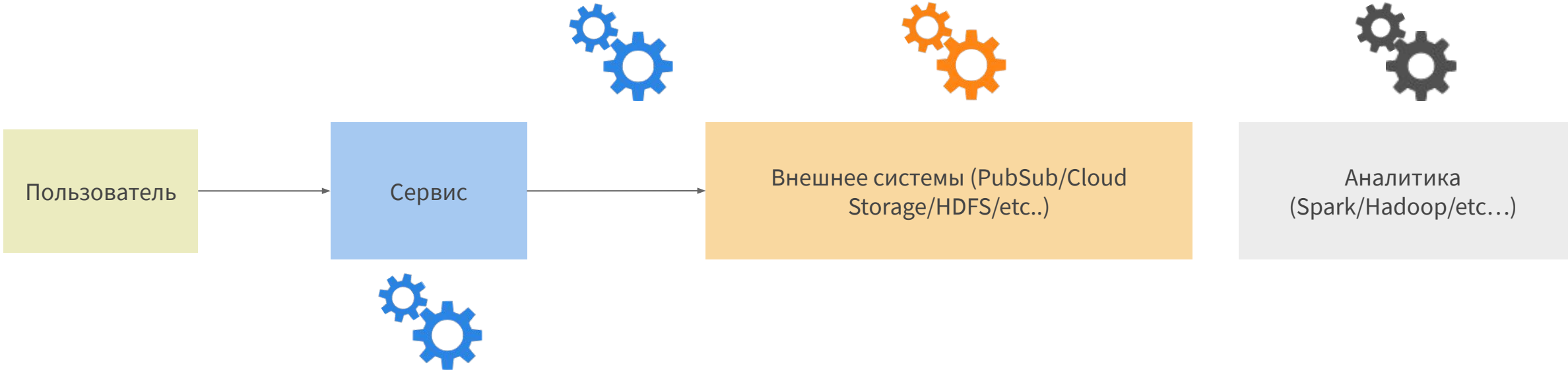
Senior Big Data developer

Дмитрий в ИТ с 2014 года. В последние несколько лет активно занимается разработкой I/O интенсивных приложений на Java, C/Linux и связанных с ними вопросами производительности. До этого разрабатывал банковский софт и платформу обработки данных на Spark в сфере рекламы. Имеет опыт работы со Scala, функциональным программированием и typelevel стеком. В настоящее время является старшим Big Data разработчиком в Grid Dynamics.

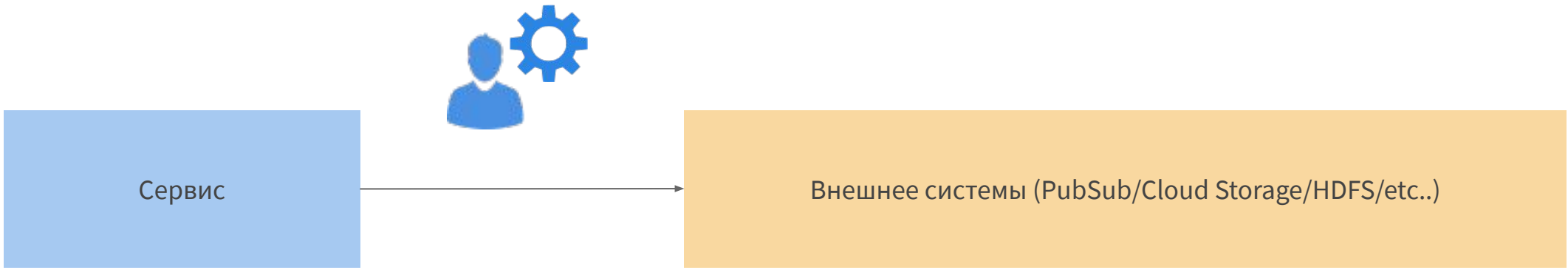
Схема потока данных Web проектов



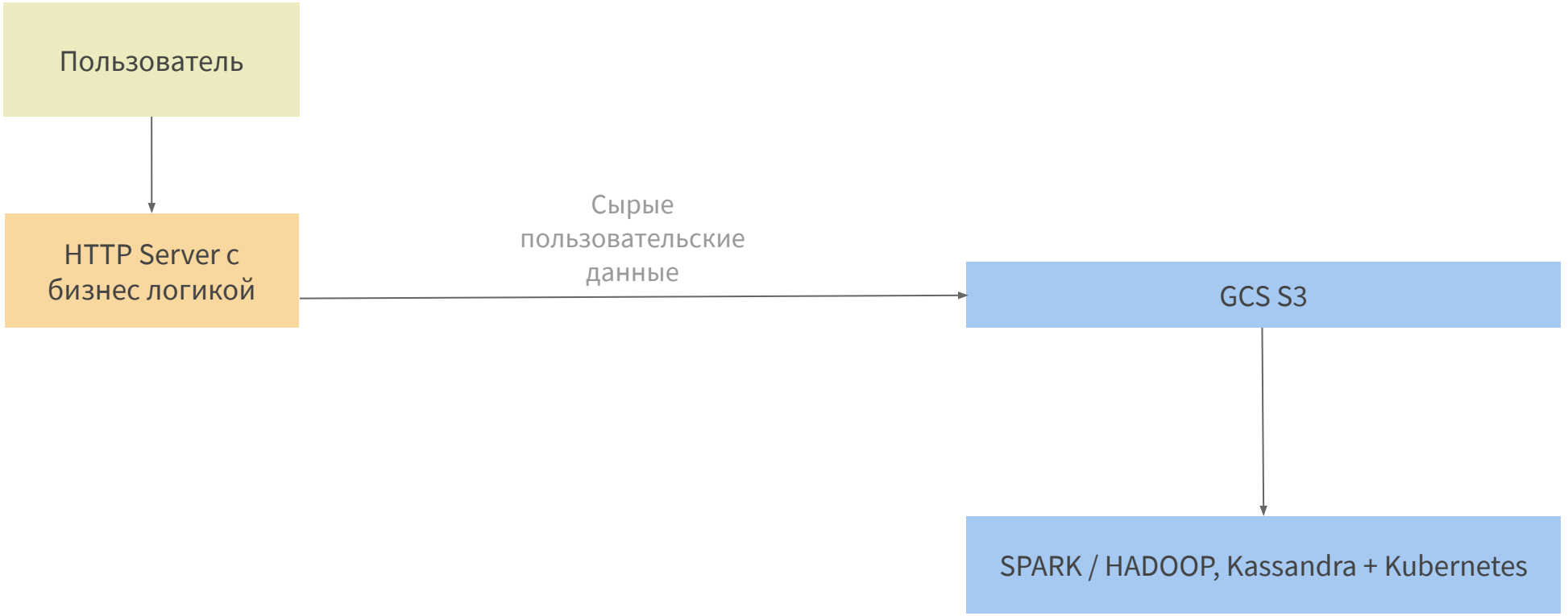
Оптимизация производительности



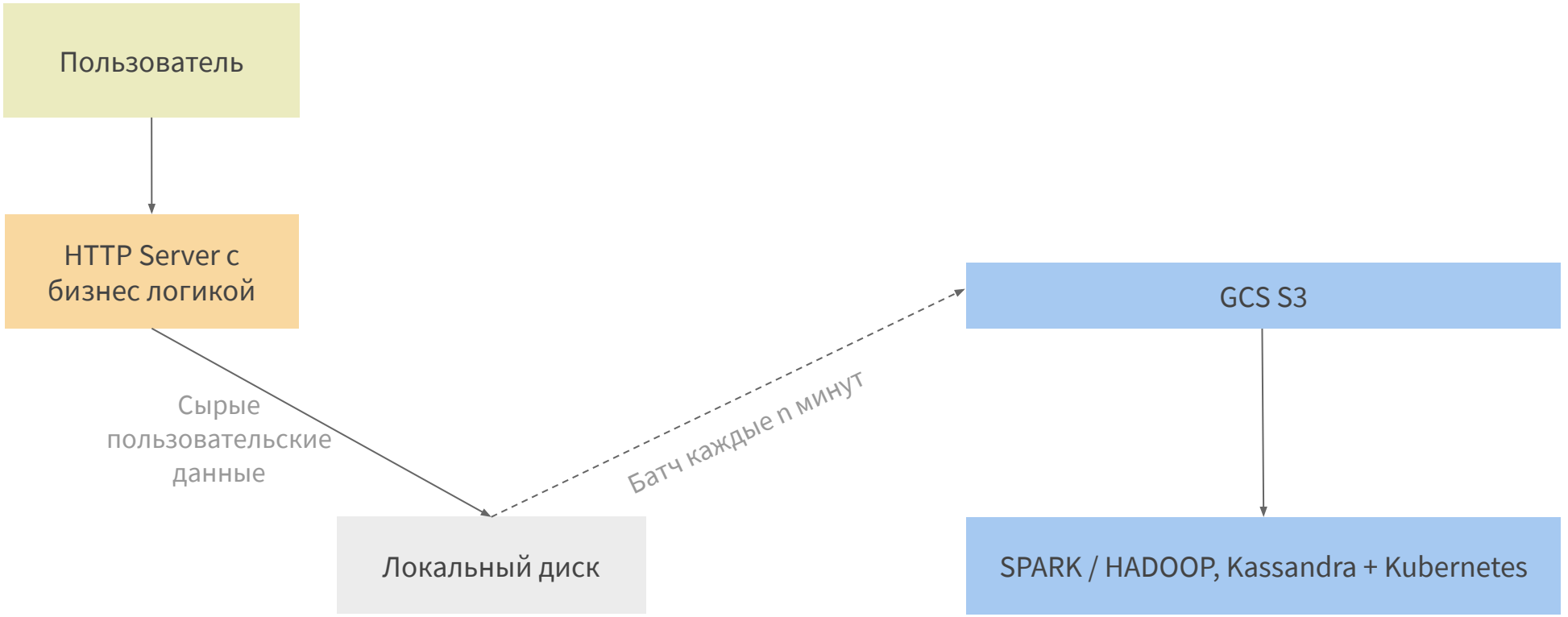
Оптимизация производительности



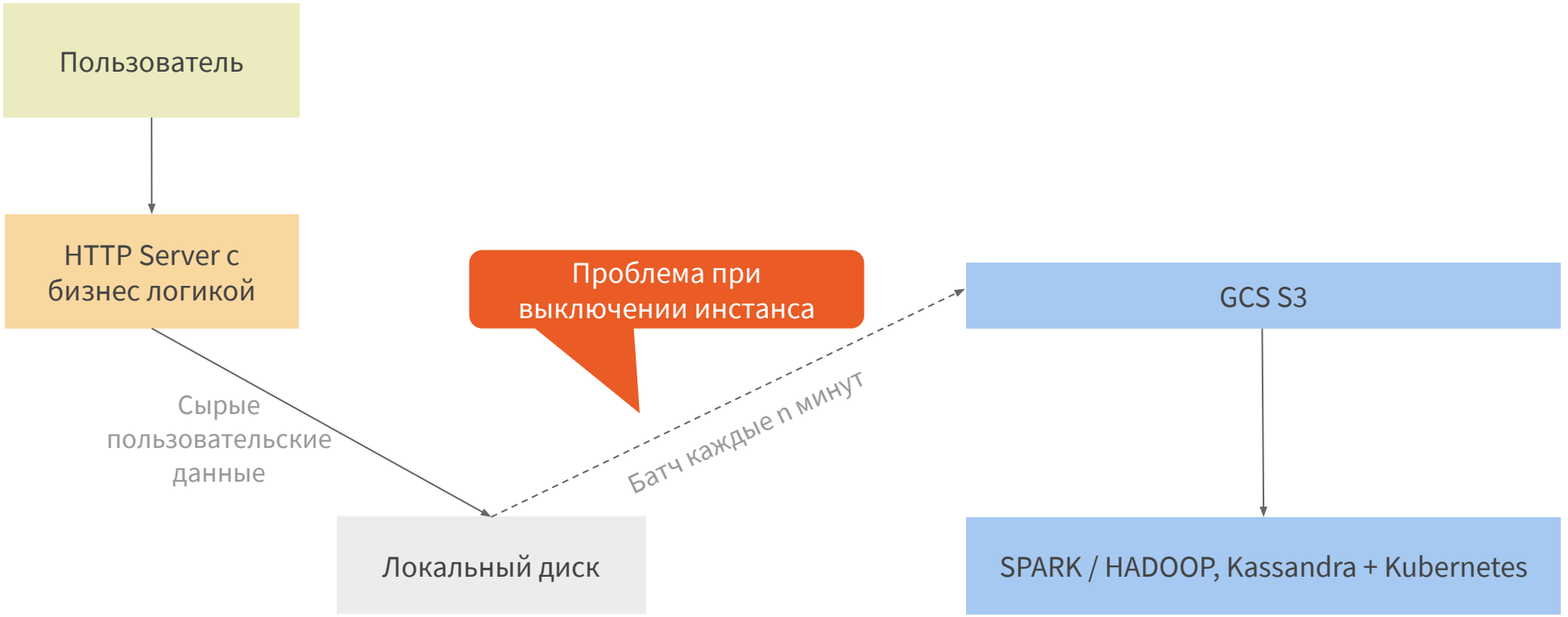
Проект в сфере рекламы



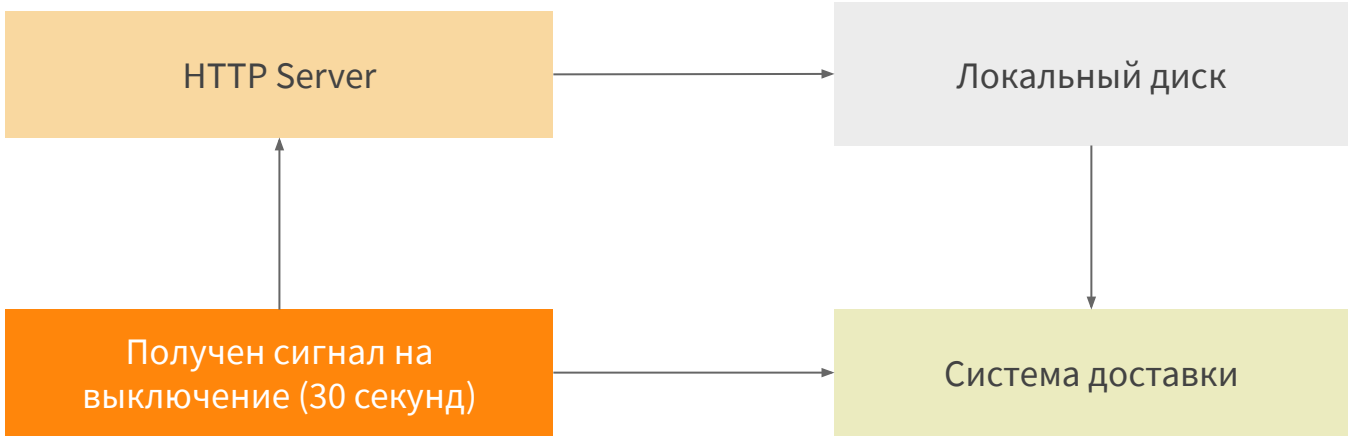
Проект в сфере рекламы



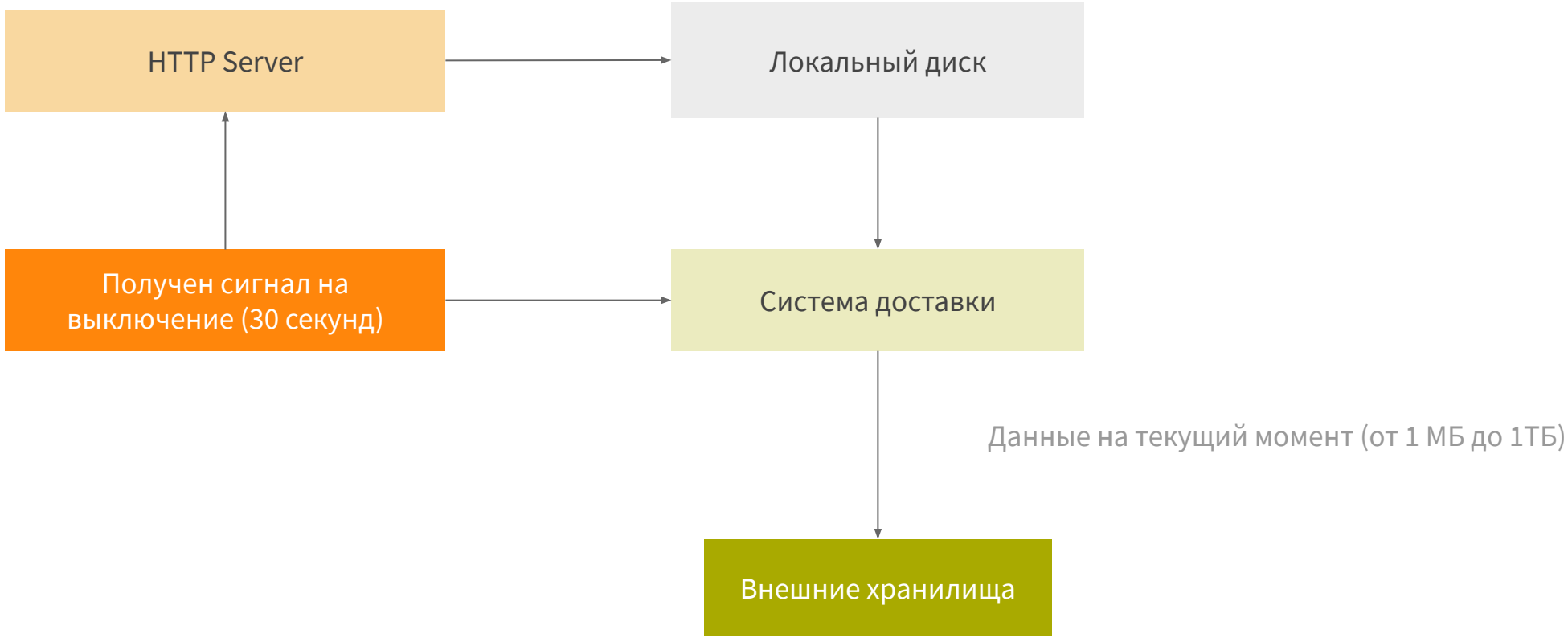
Проект в сфере рекламы



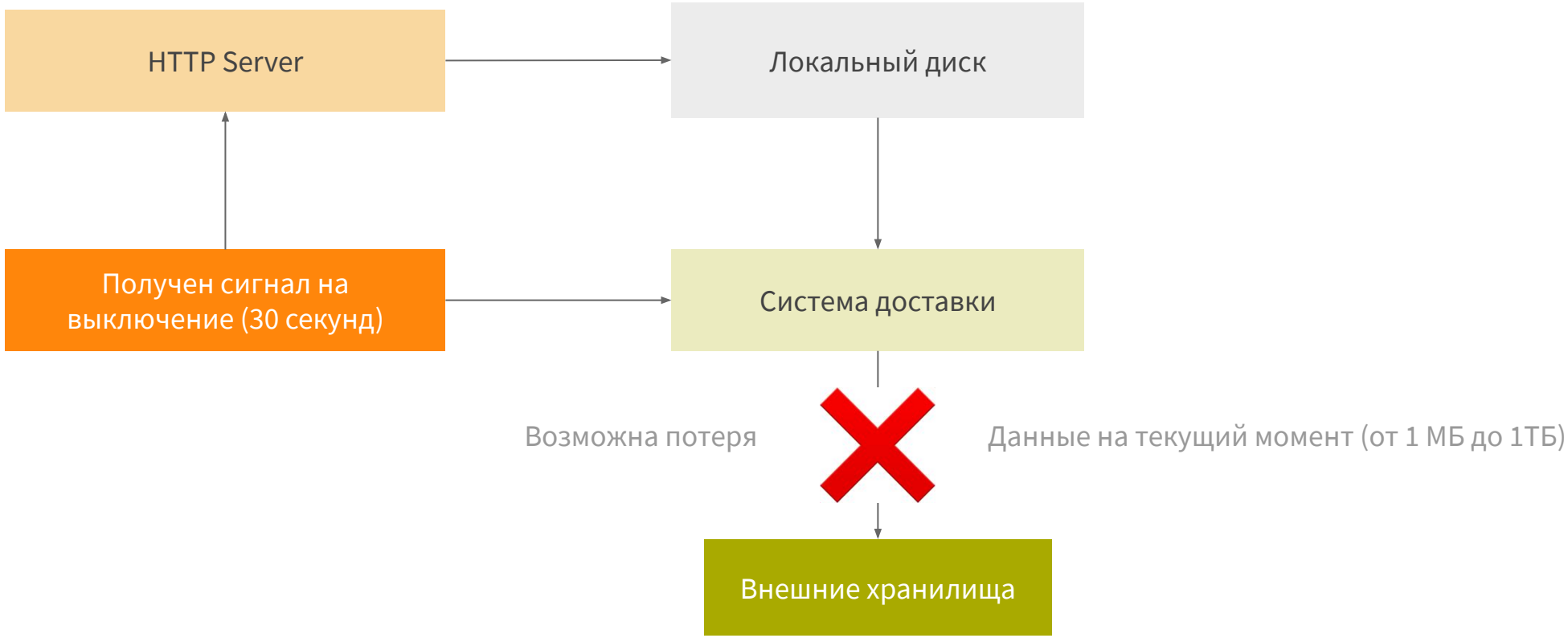
Проект в сфере рекламы: выключение инстанса



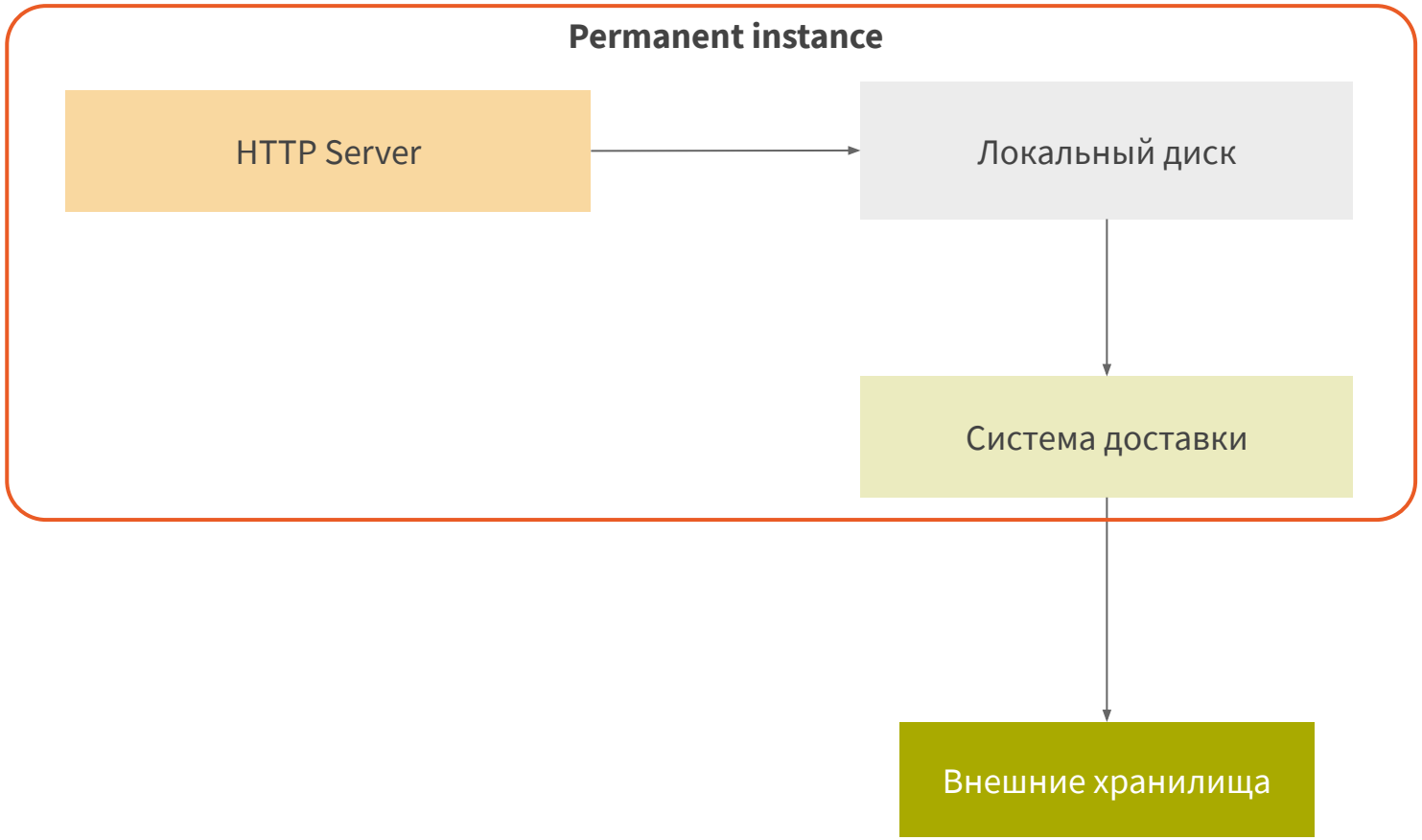
Проект в сфере рекламы: выключение инстанса



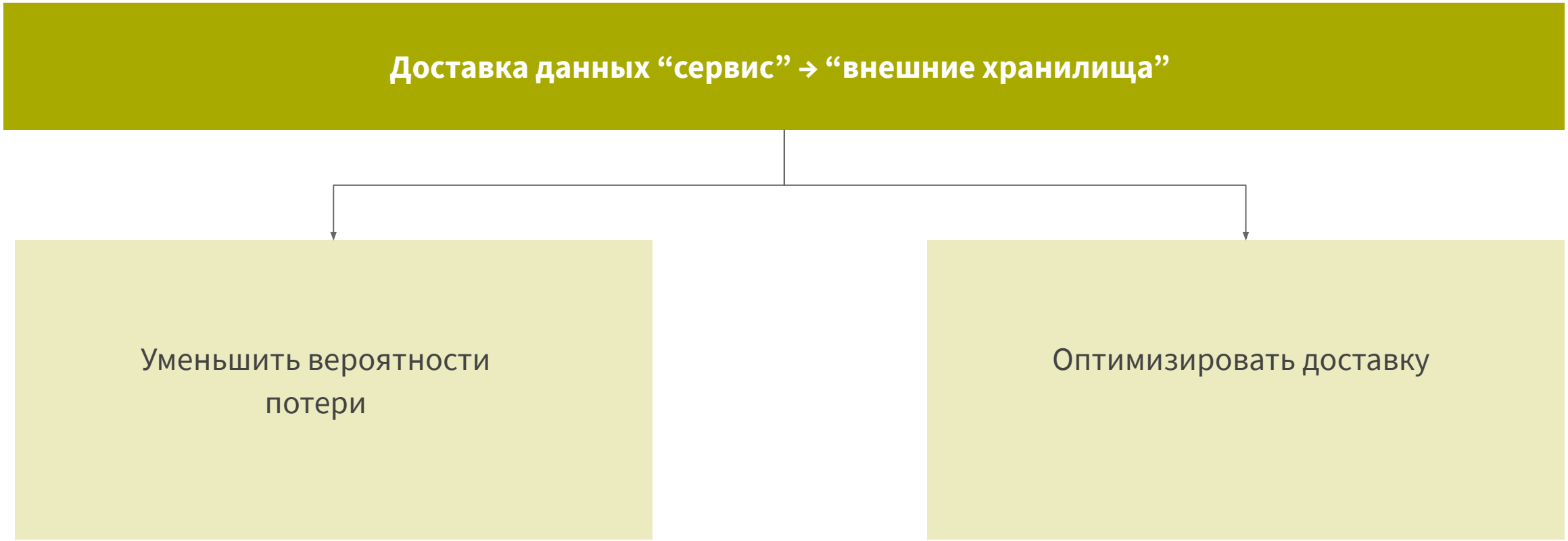
Проект в сфере рекламы: выключение инстанса



Проект в сфере рекламы: permanent instances



Задачи проекта

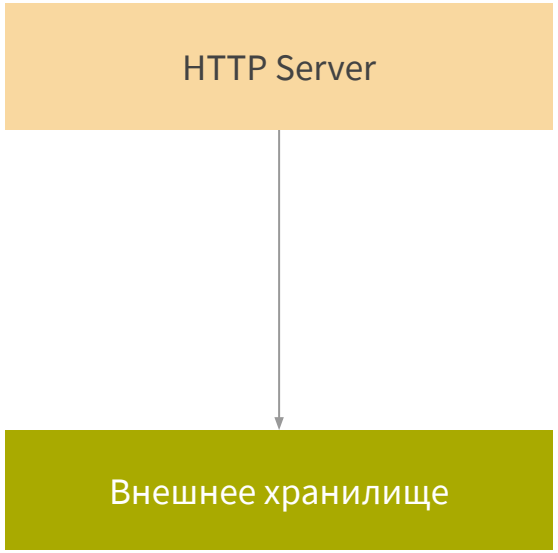


Grid Dynamics

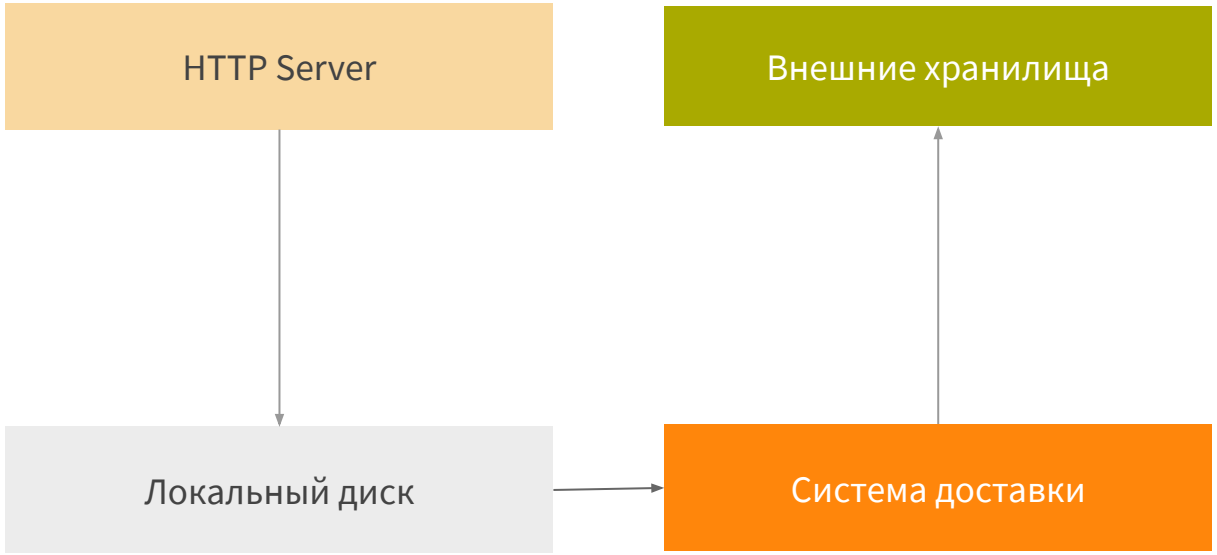
Уменьшение вероятности потери данных

Стриминг новых данных

Стриминг данных напрямую

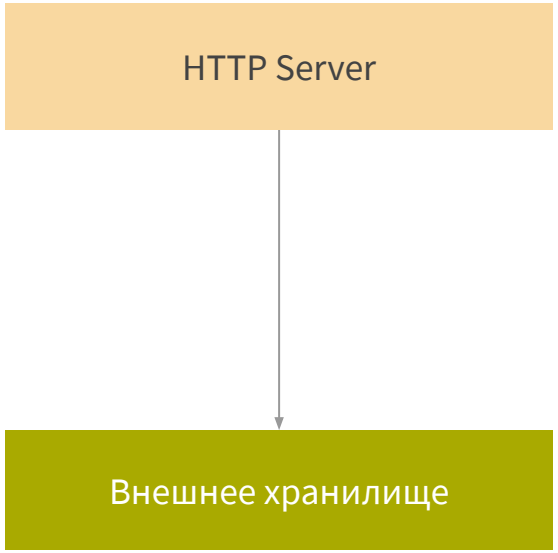


Стриминг данных с локальных дисков

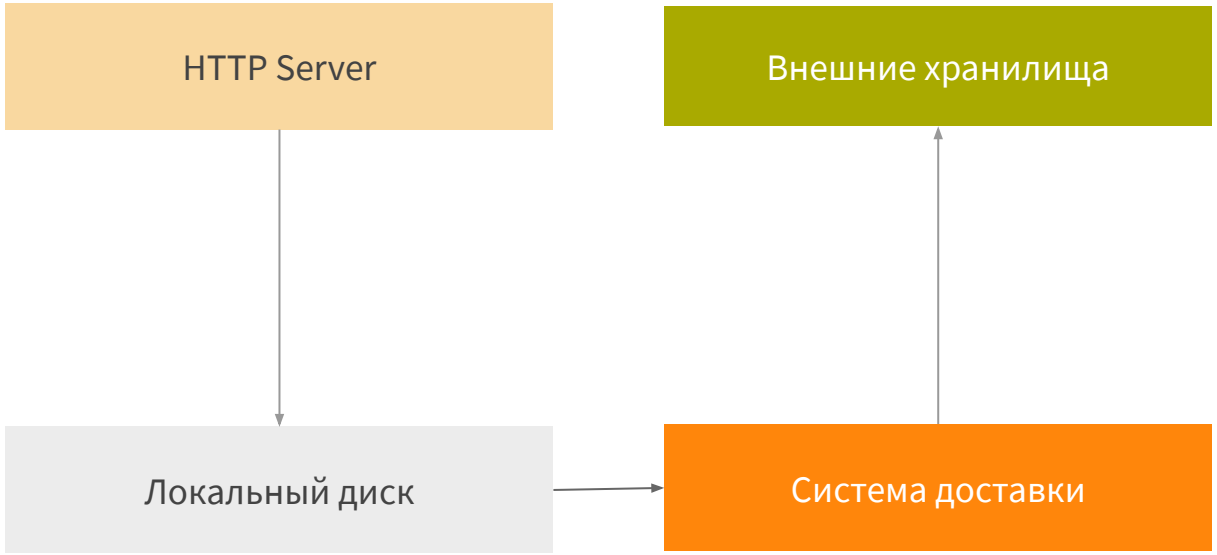


Стриминг новых данных

Стриминг данных напрямую



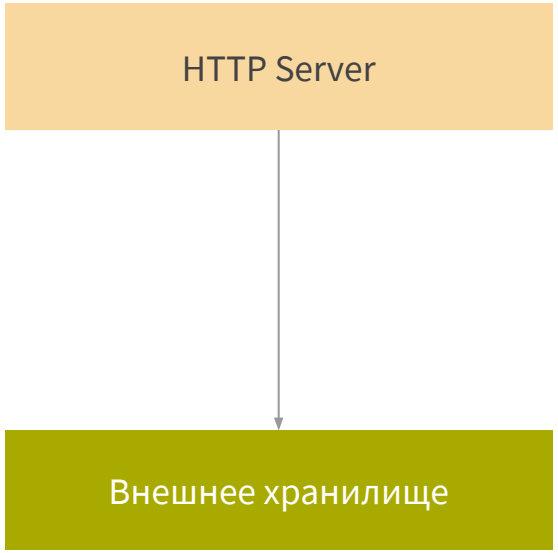
Стриминг данных с локальных дисков



#1 Стриминг данных напрямую

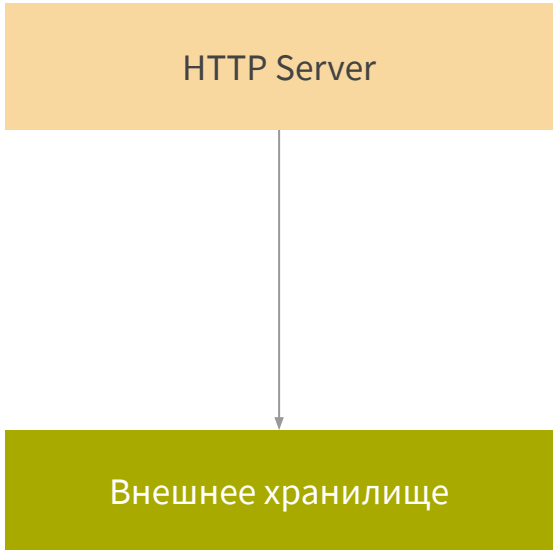
Недостатки:

- Сложность реализации
- Вероятность потери сохраняется

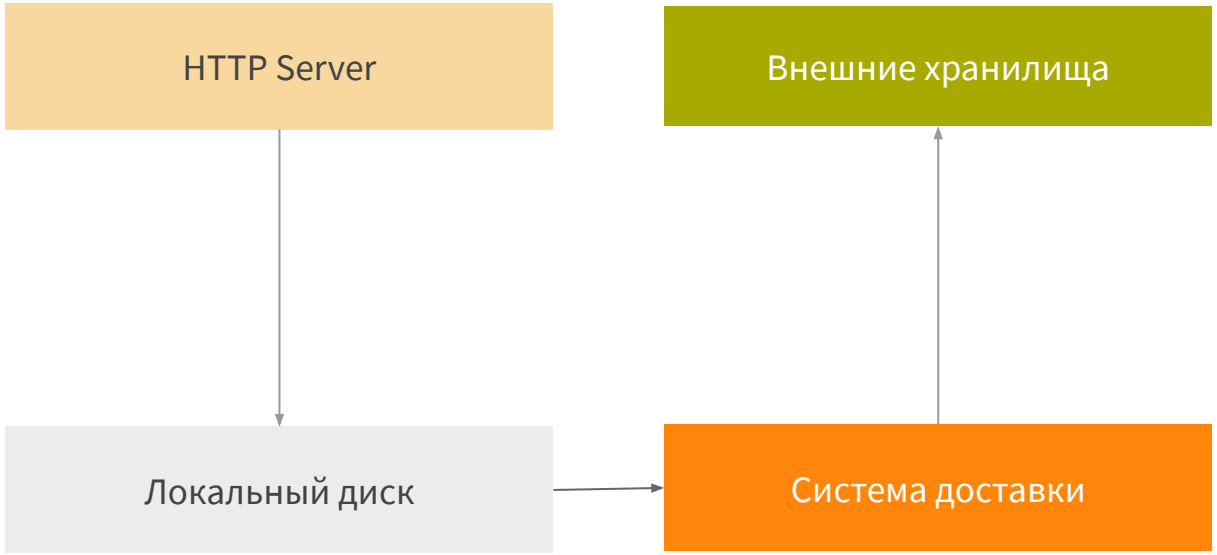


Стриминг новых данных

Стриминг данных напрямую



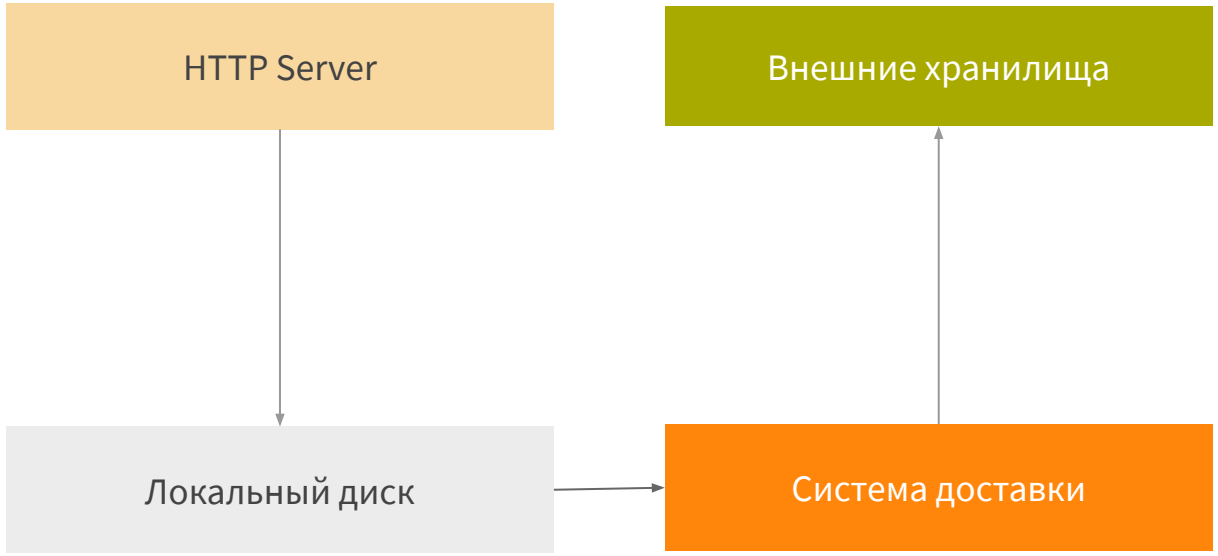
Стриминг данных с локальных дисков



#2 Стриминг данных с локальных дисков

Преимущества:

- Гибкость



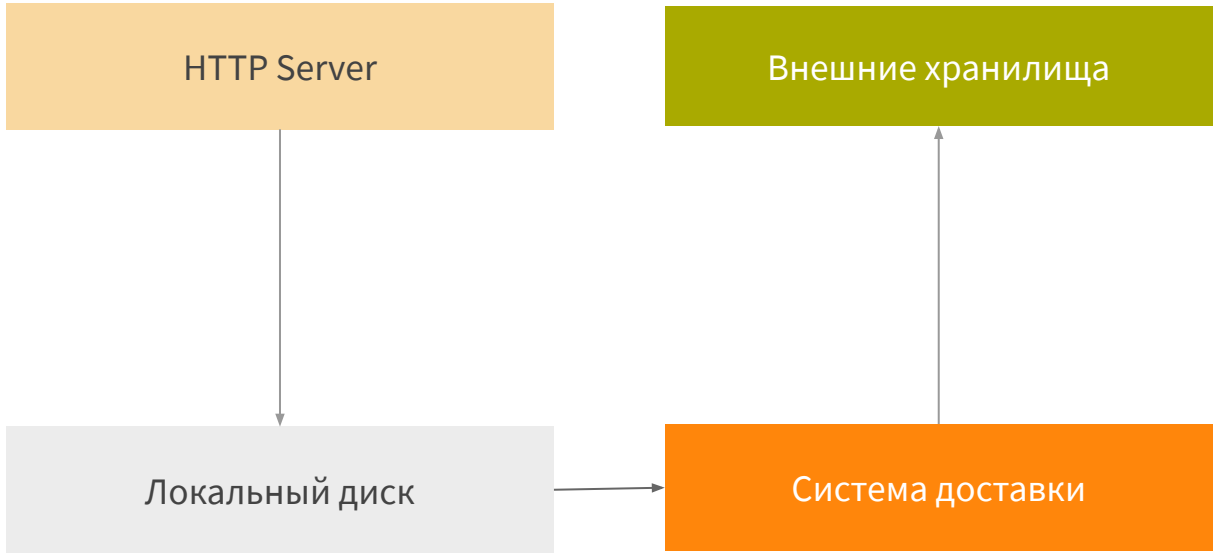
#2 Стриминг данных с локальных дисков

Преимущества:

- Гибкость

Недостатки:

- ~~Сложность реализации~~
- ~~Вероятность потери сохраняется~~



Grid Dynamics

Оптимизация доставки данных

Архитектура системы стриминга

Файловое I/O

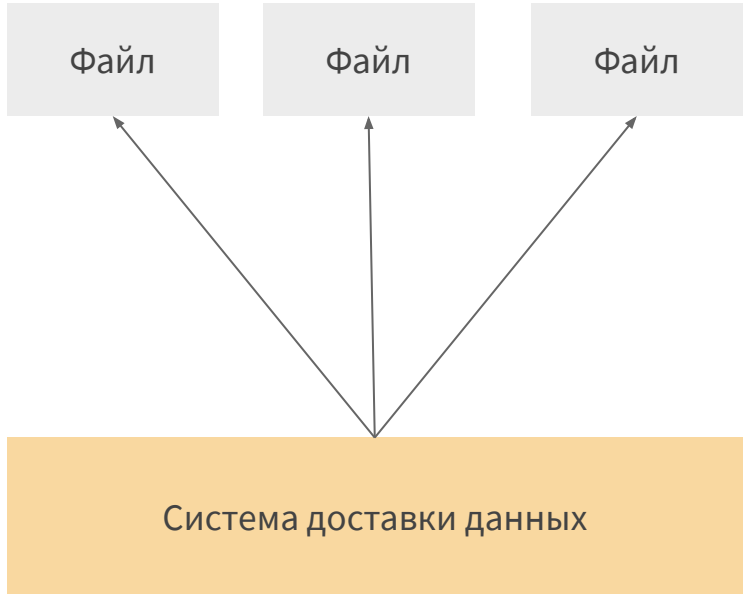
- Определение изменившихся файлов
- Определение предыдущей позиции стриминга
- Чтение новых данных

Файловое I/O

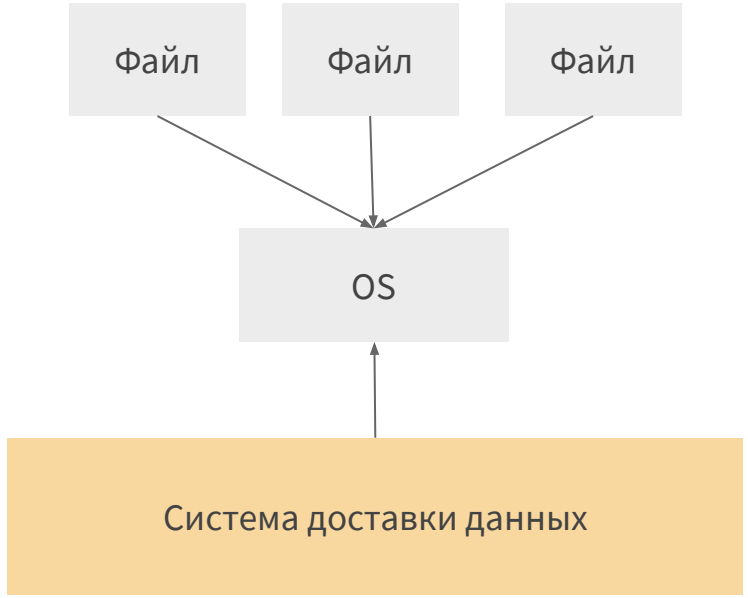
- **Определение изменившихся файлов**
- Определение предыдущей позиции стриминга
- Чтение новых данных

Определение изменившихся файлов

- Поллинг

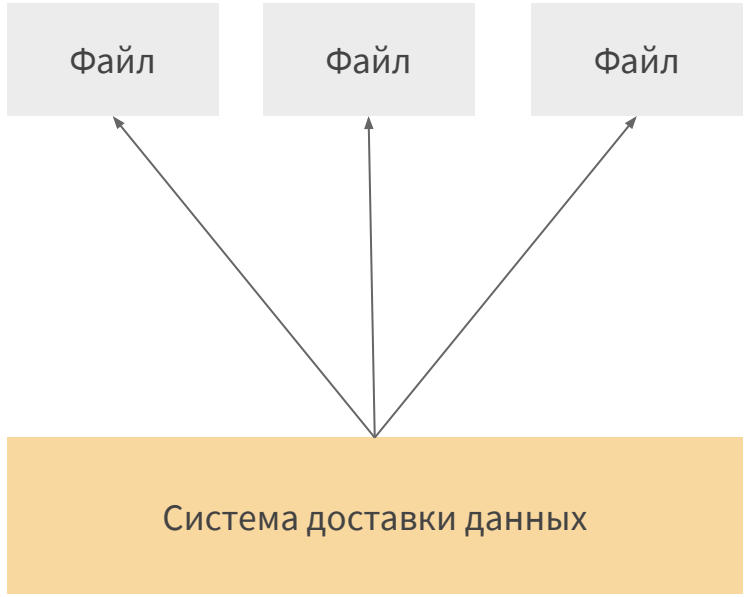


- Подписка на события

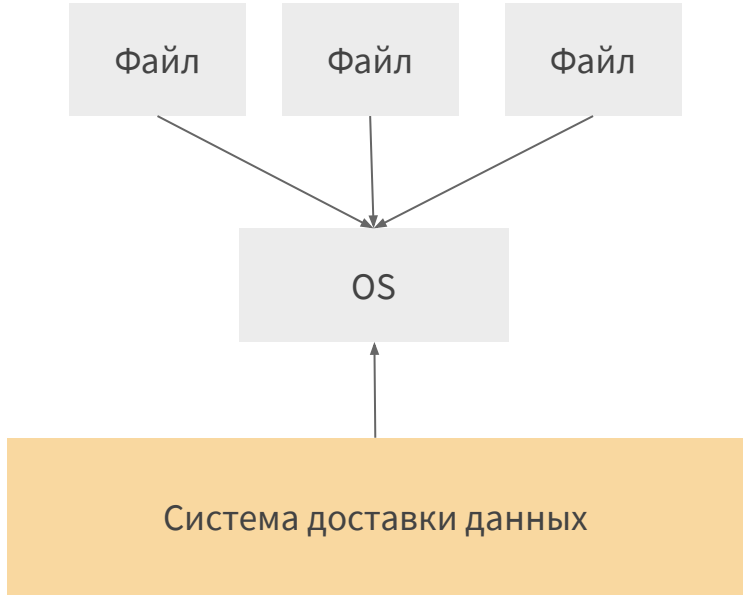


Определение изменившихся файлов

- **Поллинг**

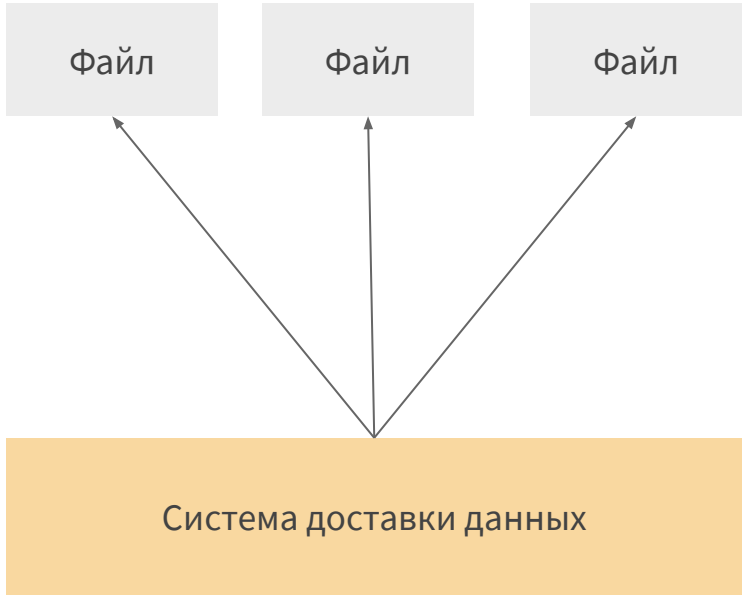


- Подписка на события



Поллинг

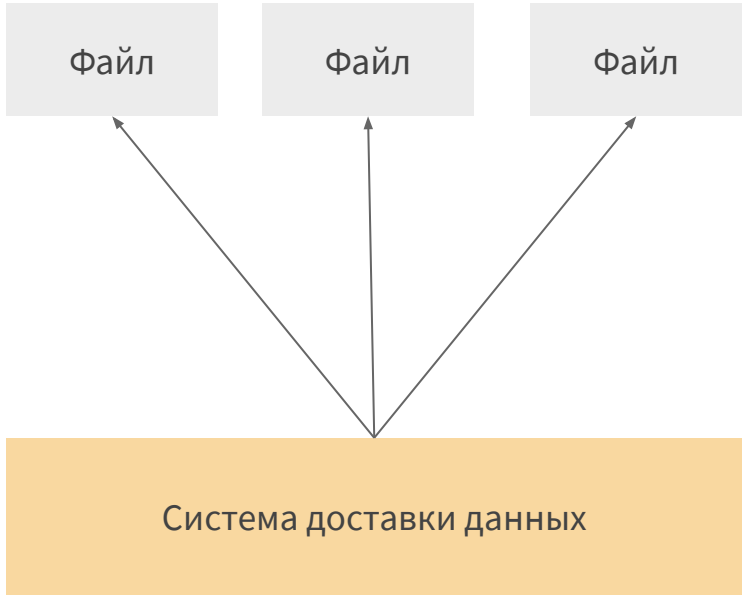
```
long pollInterval;
List<String> watchedFiles;
//...
while(...){
      for(String file: watchedFiles){
            //Проверка наличия новых данных в файле
      }
      Thread.sleep(pollInterval); //throws InterruptedException
}
```



Поллинг

Преимущества:

- Простота



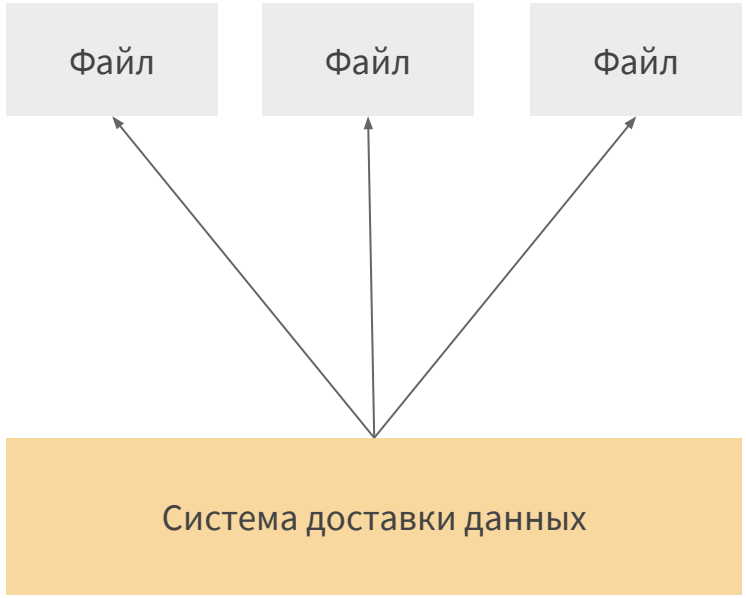
Поллинг

Преимущества:

- Простота

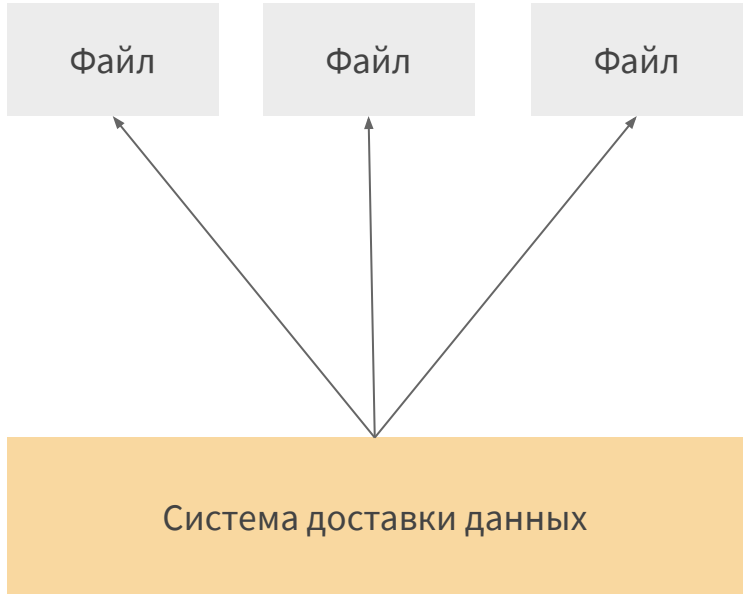
Недостатки:

- Отсутствие гибкости
- Потребление ресурсов

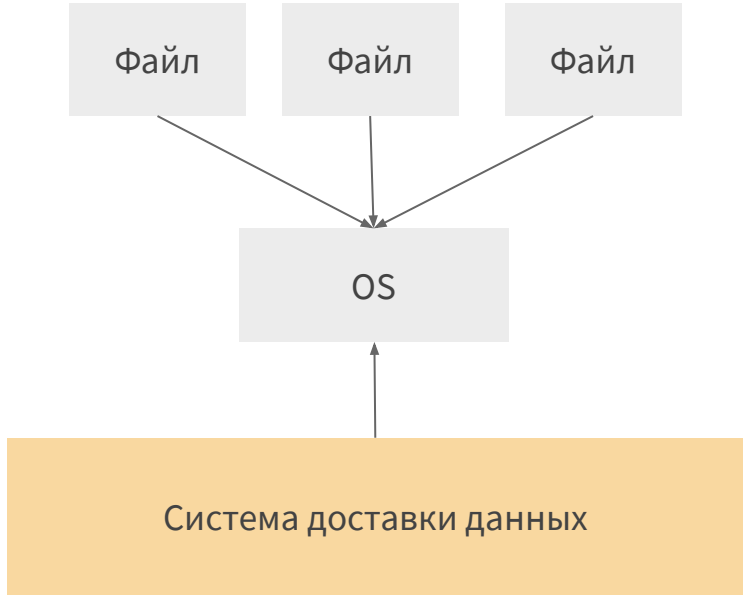


Определение изменившихся файлов

- Поллинг

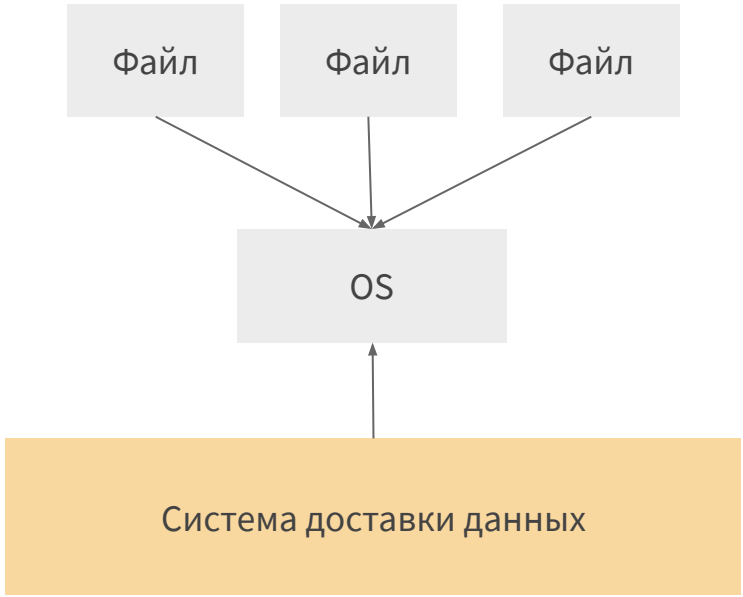


- **Подписка на события**

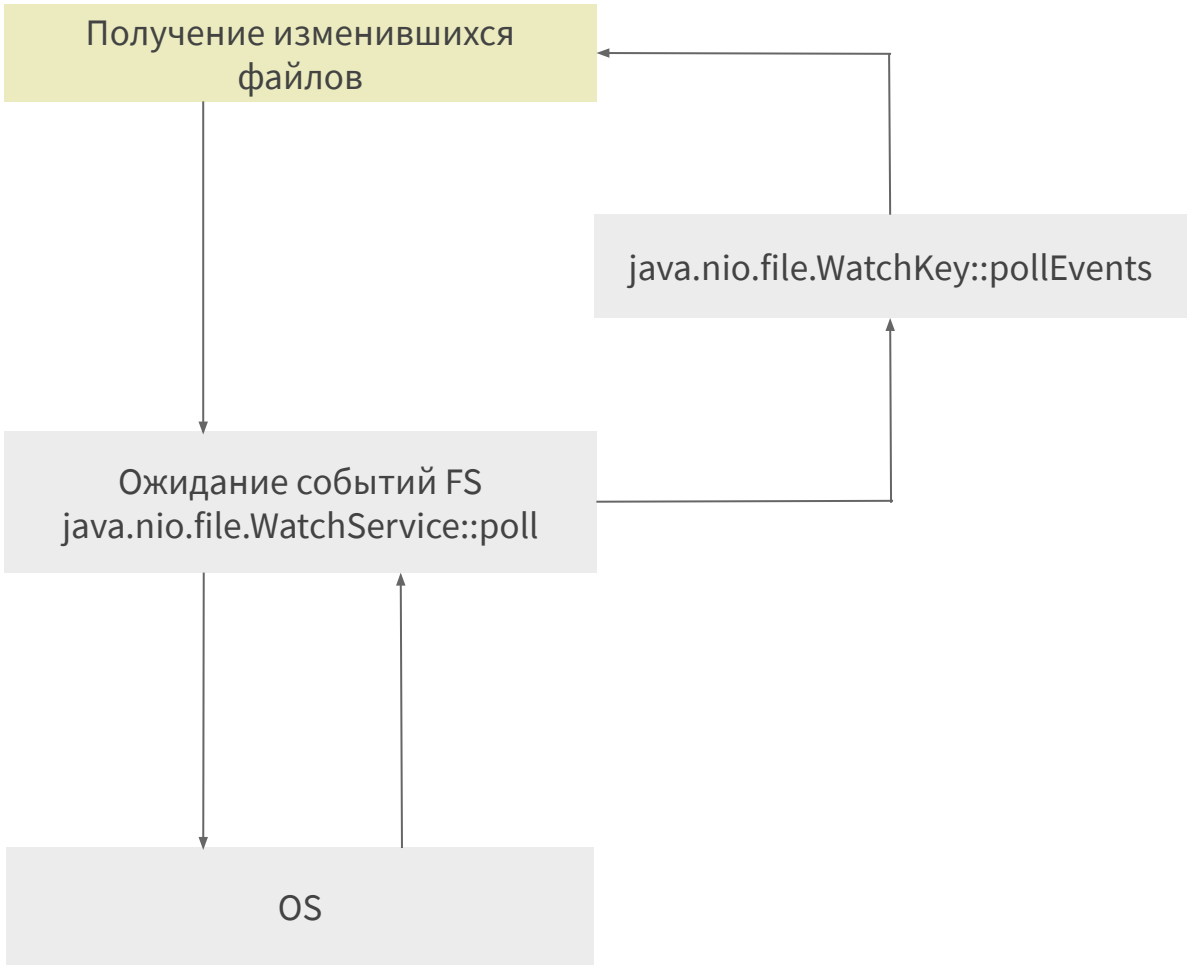


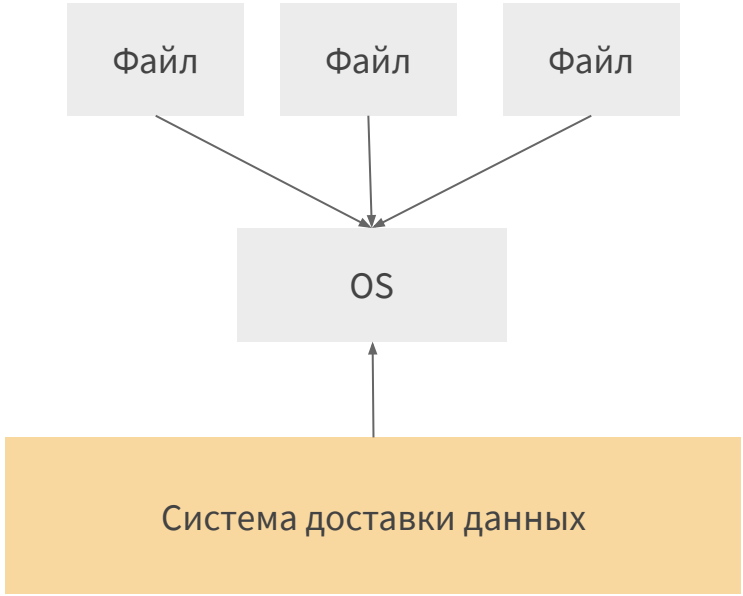
Подписка на события: JDK java.nio.file

```
1     package java.nio.file;
2
3     public interface WatchService extends Closeable {
4           WatchKey poll(long timeout, TimeUnit unit)
5                 throws InterruptedException;
6
7           WatchKey take() throws InterruptedException;
8
9           WatchKey poll();
10    }
11
12    public interface WatchKey {
13
14          List<WatchEvent<?>> pollEvents();
15
16          Watchable watchable();
17          //...
18    }
19
20    public interface WatchEvent<T> {
21          T context();
22
23          //...
24    }
```



JDK java.nio.file: схема использования

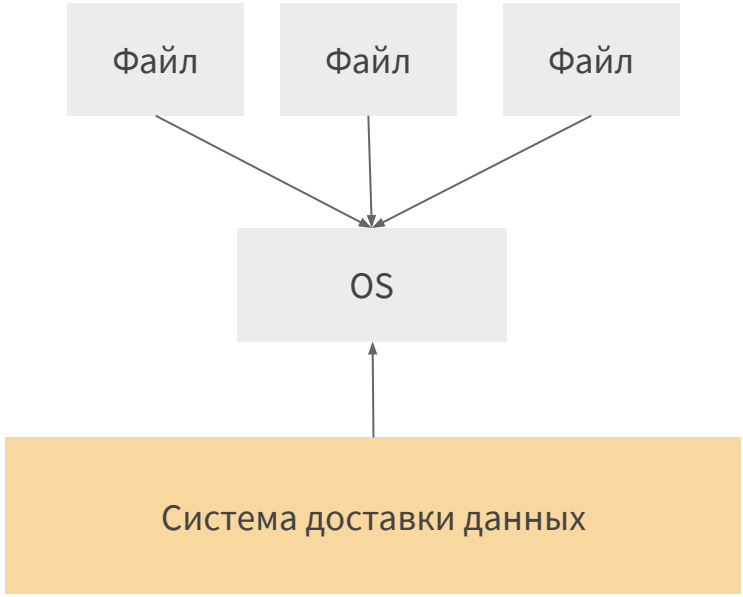




Подписка на события: JDK java.nio.file

Преимущества:

- Обработка только изменившихся файлов
- Кроссплатформенность



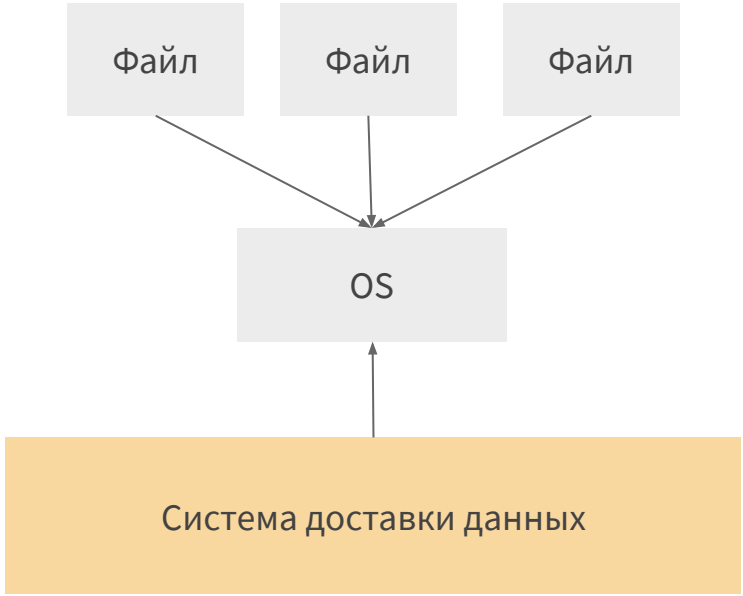
Подписка на события: JDK java.nio.file

Преимущества:

- Обработка только изменившихся файлов
- Кроссплатформенность

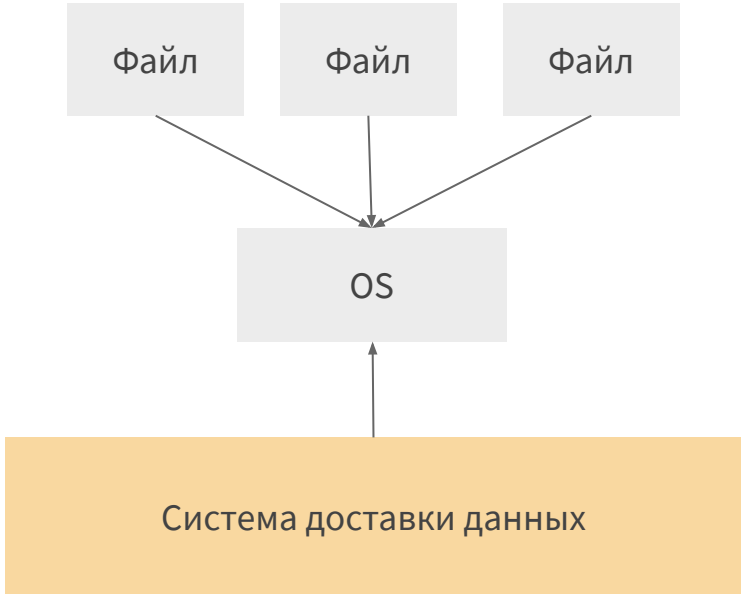
Недостатки:

- Скудное множество доступных эвентов



Подписка на события: JDK java.nio.file

```
1    package java.nio.file;
2
3    public final class StandardWatchEventKinds {
4
5         public static final WatchEvent.Kind<Object> OVERFLOW = //
6
7         public static final WatchEvent.Kind<Path> ENTRY_CREATE = //
8
9         public static final WatchEvent.Kind<Path> ENTRY_DELETE = //
10
11        public static final WatchEvent.Kind<Path> ENTRY_MODIFY = //
12   }
```



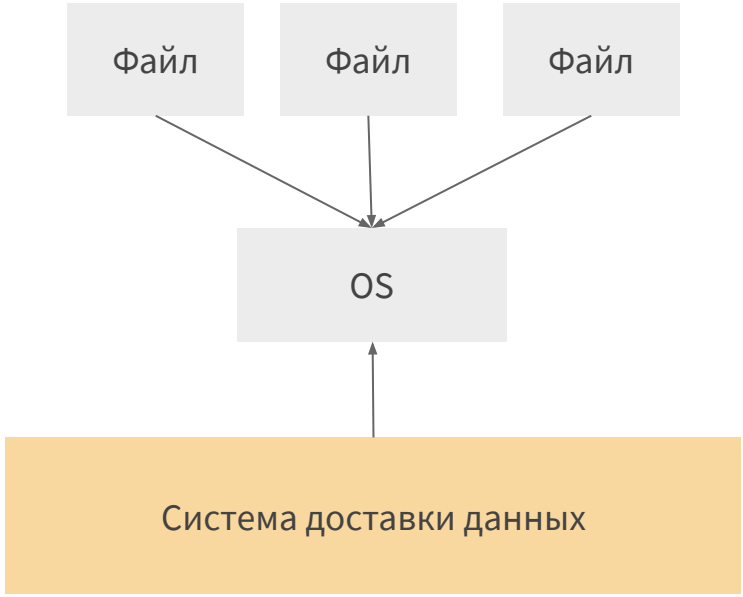
Подписка на события: JDK java.nio.file

Преимущества:

- Обработка только изменившихся файлов
- Кроссплатформенность

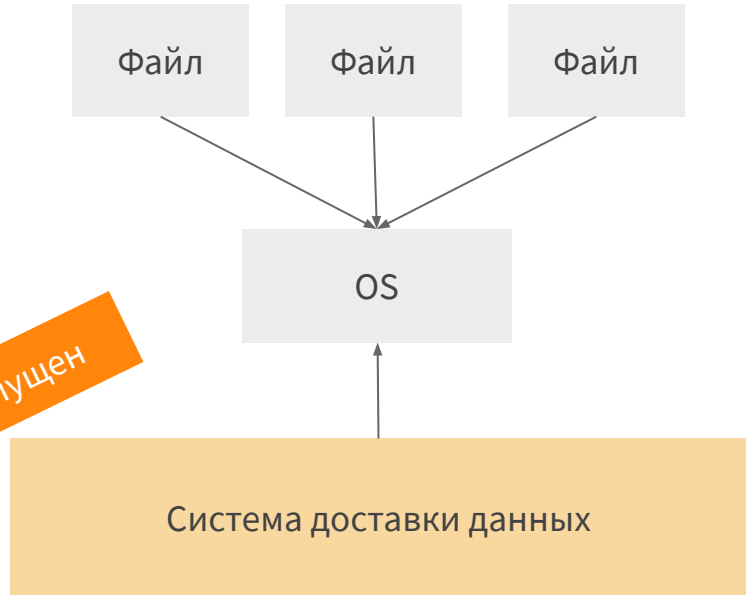
Недостатки:

- Скудное множество доступных эвентов
- Игнорируется информация о перемещении



Подписка на события: JDK java.nio.file

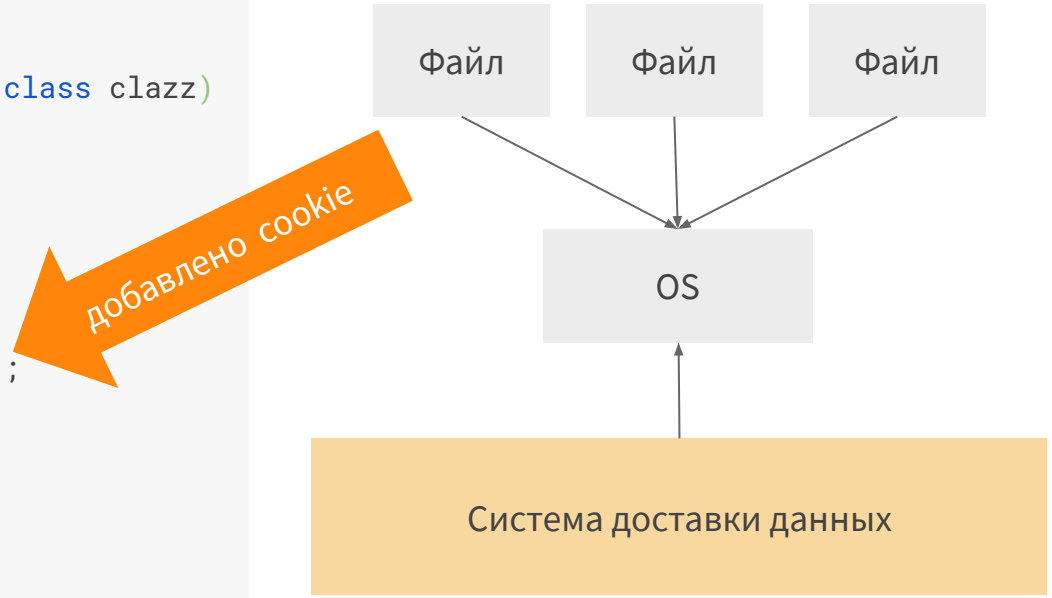
```
1   package sun.nio.fs;
2
3   class LinuxWatchService extends AbstractWatchService {
4       //...
5       private static class Poller extends AbstractPoller {
6           /**
7            * struct inotify_event {
8            *     int          wd;
9            *     uint32_t     mask;
10           *     uint32_t     len
11           *     char name    __flexarr;  //present if len > 0
12           * } act_t;
13           */
14           public static final int SIZEOF_INOTIFY_EVENT = eventSize();
15           public static final int[] offsets            = eventOffsets();
16           public static final int OFFSETOF_WD          = offsets[0];
17           public static final int OFFSETOF_MASK        = offsets[1];
18           public static final int OFFSETOF_LEN         = offsets[3];
19           public static final int OFFSETOF_NAME        = offsets[4];
20           //...
21      }
22      public static native int[] eventOffsets();
23  }
```



Подписка на события: JDK java.nio.file

LinuxWatchService.c *

```
1       JNIEXPORT jintArray JNICALL
2       Java_sun_nio_fs_LinuxWatchService_eventOffsets(JNIEnv *env, jclass clazz)
3       {
4           jintArray result = (*env)->NewIntArray(env, 5);
5           if (result != NULL) {
6               jint arr[5];
7               arr[0] = (jint)offsetof(struct inotify_event, wd);
8               arr[1] = (jint)offsetof(struct inotify_event, mask);
9               arr[2] = (jint)offsetof(struct inotify_event, cookie);
10              arr[3] = (jint)offsetof(struct inotify_event, len);
11              arr[4] = (jint)offsetof(struct inotify_event, name);
12              (*env)->SetIntArrayRegion(env, result, 0, 5, arr);
13          }
14      }
```



* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/linux/native/libnio/fs/LinuxWatchService.c#l54

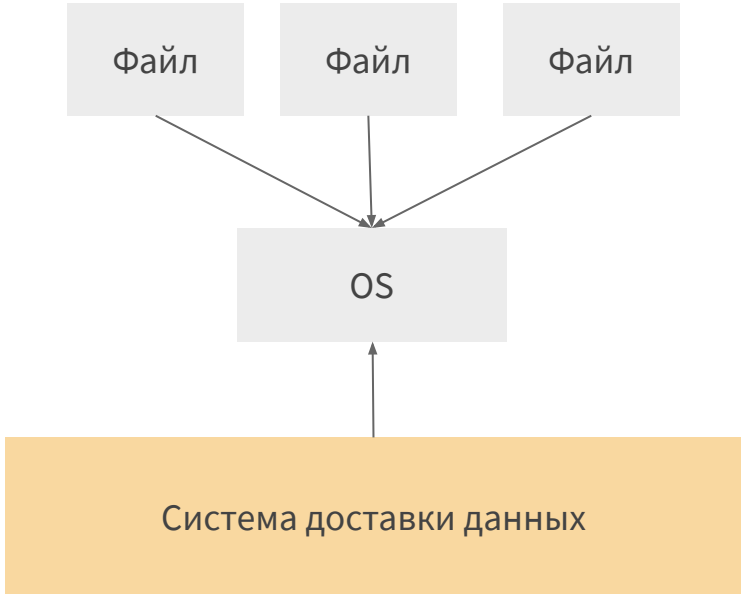
Подписка на события: JDK java.nio.file

Преимущества:

- Обработка только изменившихся файлов
- Кроссплатформенность

Недостатки:

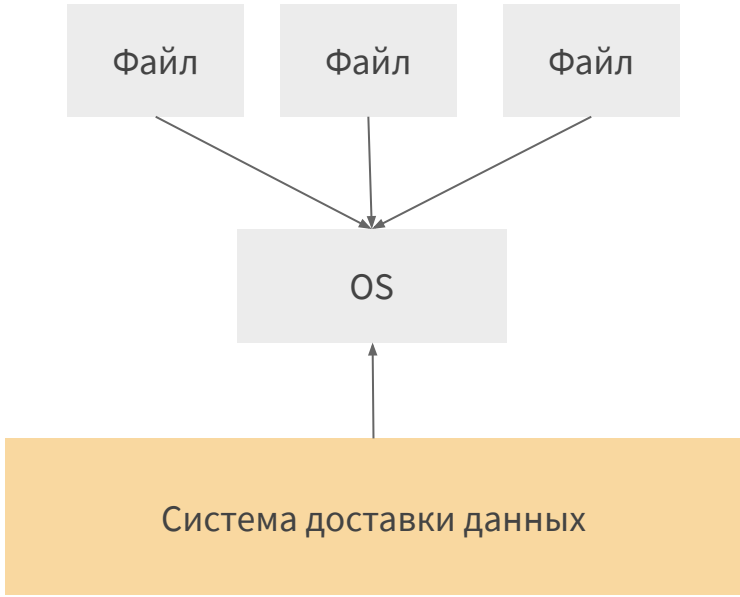
- Скудное множество доступных эвентов
- Игнорируется информация о перемещении



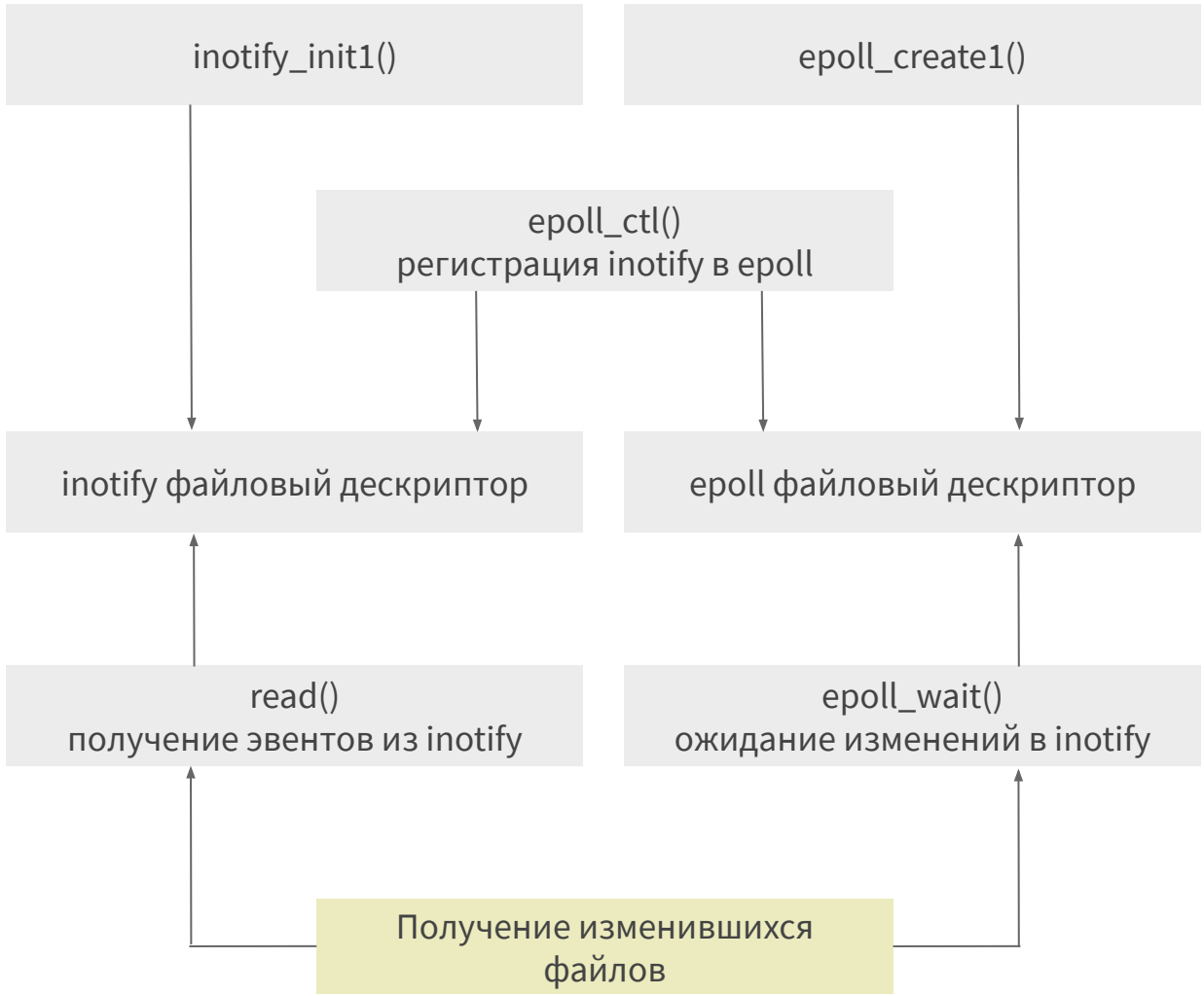
Вывод: при необходимости использования всех эвентов предоставляемых OS java.nio.file не подходит

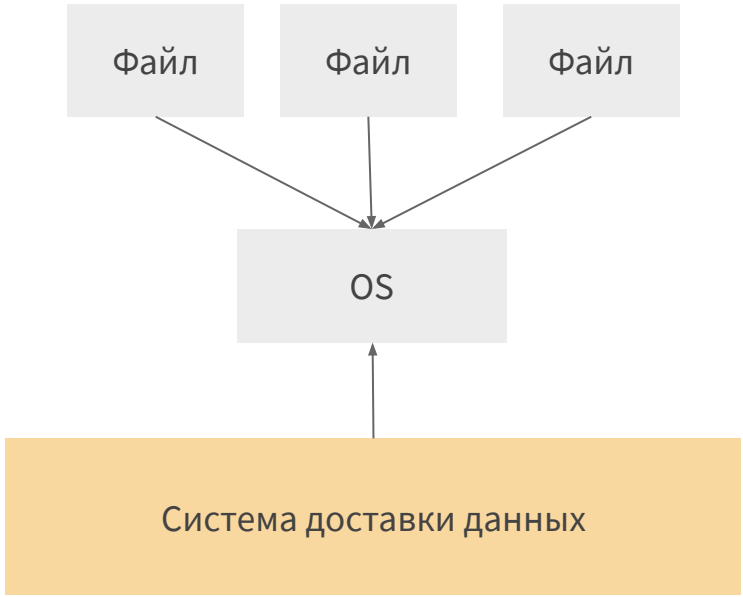
Подписка на события: inotify/Linux + epoll/Linux

```c
#include <sys/inotify.h>

struct inotify_event {
    int      wd;       /* Watch descriptor */
    uint32_t mask;     /* Mask describing event */
    uint32_t cookie;   /* Unique cookie associating related
                          event for rename(2) */
    uint32_t len;      /* Size of name field */
    char     name[];   /* Optional null-terminated name */
}

int inotify_init(void);

int inotify_add_watch(int fd, const char *pathname, uint32_t mask);

int inotify_rm_watch(int fd, int wd);

#include <sys/epoll.h>

int epoll_create1(int flags);

int epoll_ctl(int epfd, int op, struct epoll_event *event);

int epoll_wait(int epfd, struct epoll_event *events,
               int maxevents, int timeout);
```



inotify/Linux + epoll/Linux: схема использования

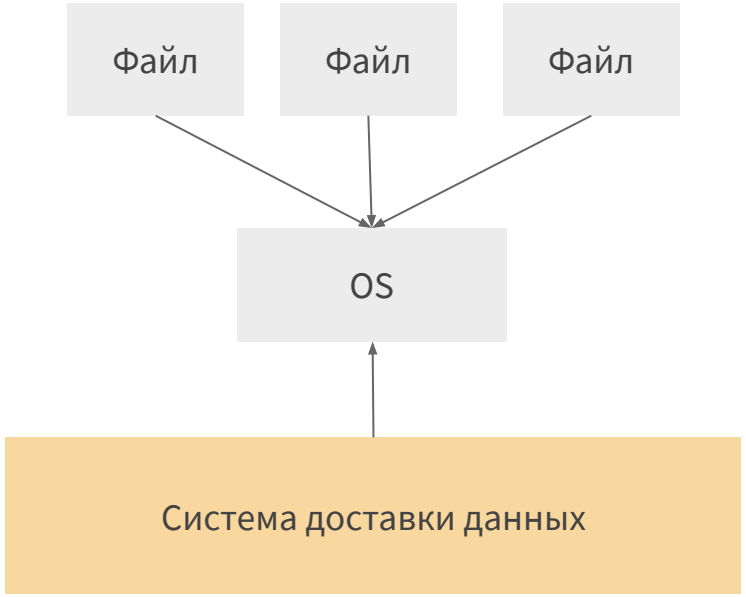




Подписка на события: inotify/Linux + epoll/Linux

Преимущества:

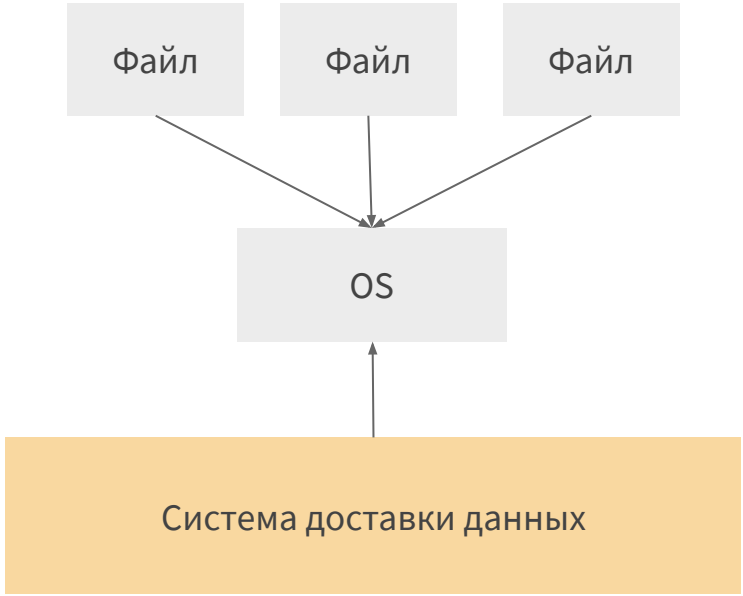
- Полный набор событий OS



Подписка на события: inotify/Linux + epoll/Linux

inotify.h *

```
1       #define IN_ACCESS           0x00000001  /* File was accessed */
2       #define IN_MODIFY           0x00000002  /* File was modified */
3       #define IN_ATTRIB           0x00000004  /* Metadata changed */
4       #define IN_CLOSE_WRITE      0x00000008  /* Writtable file was closed */
5       #define IN_CLOSE_NOWRITE    0x00000010  /* Unwrittable file closed */
6       #define IN_OPEN             0x00000020  /* File was opened */
7       #define IN_MOVED_FROM       0x00000040  /* File was moved from X */
8       #define IN_MOVED_TO         0x00000080  /* File was moved to Y */
9       #define IN_CREATE           0x00000100  /* Subfile was created */
10      #define IN_DELETE           0x00000200  /* Subfile was deleted */
11      #define IN_DELETE_SELF      0x00000400  /* Self was deleted */
12      #define IN_MOVE_SELF        0x00000800  /* Self was moved */
```



* https://elixir.bootlin.com/linux/v5.3/source/include/uapi/linux/inotify.h#L29

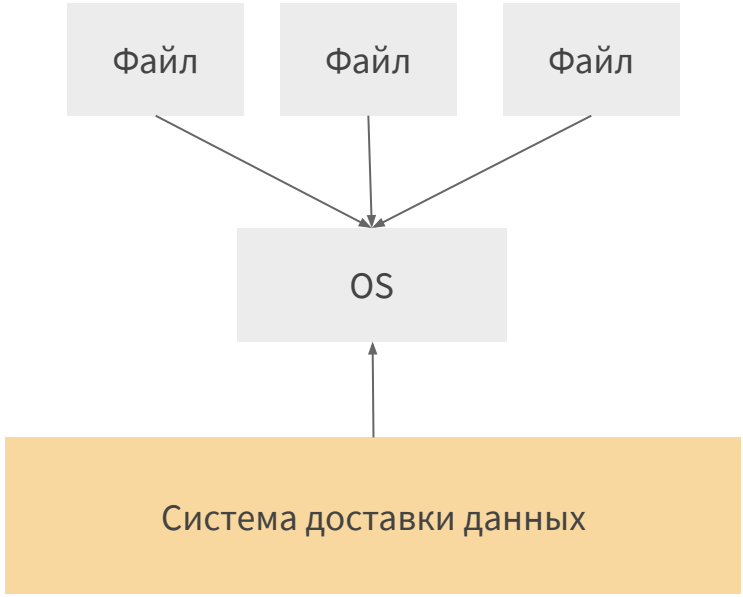
Подписка на события: inotify/Linux + epoll/Linux

Преимущества:

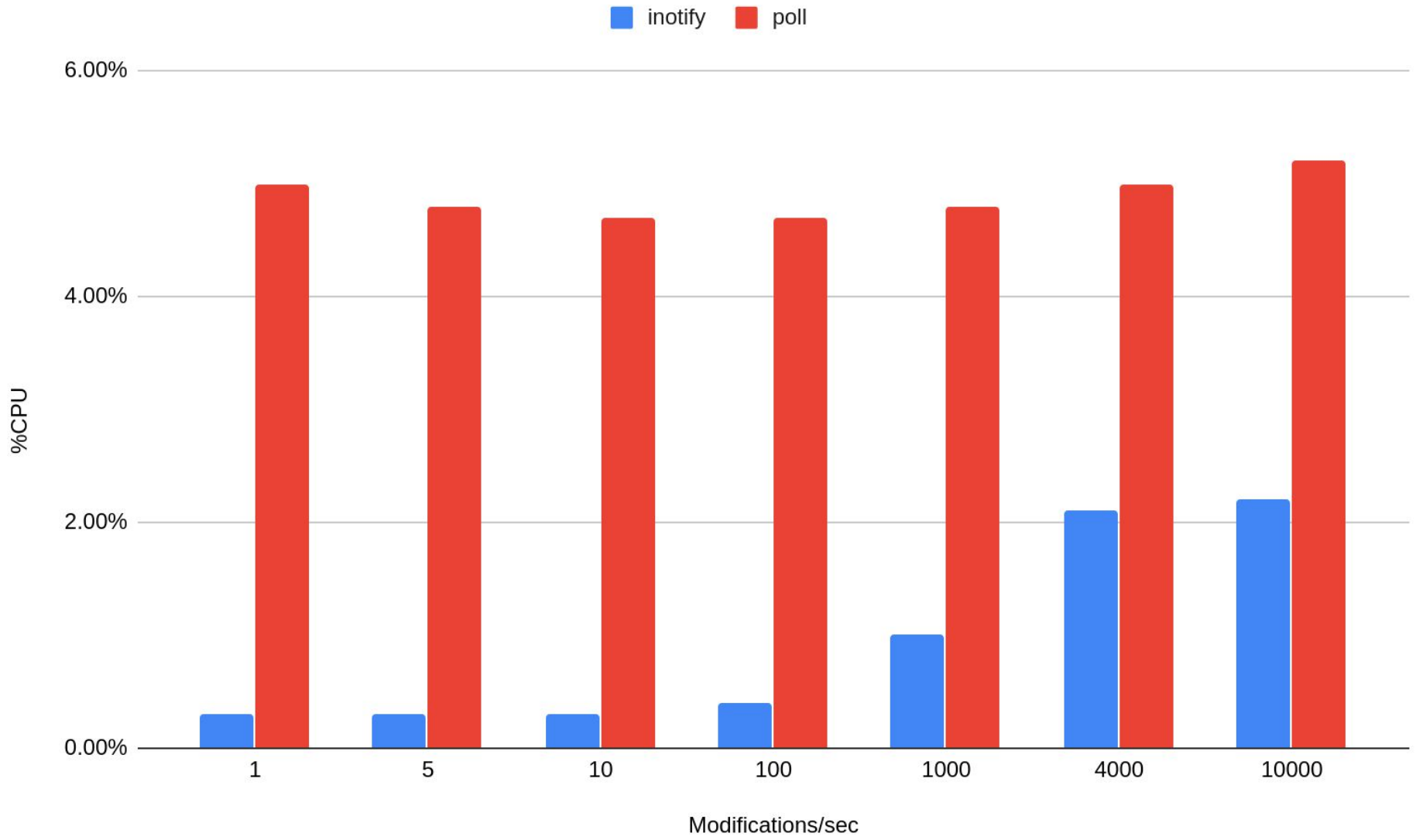
- Полный набор событий OS

Недостатки:

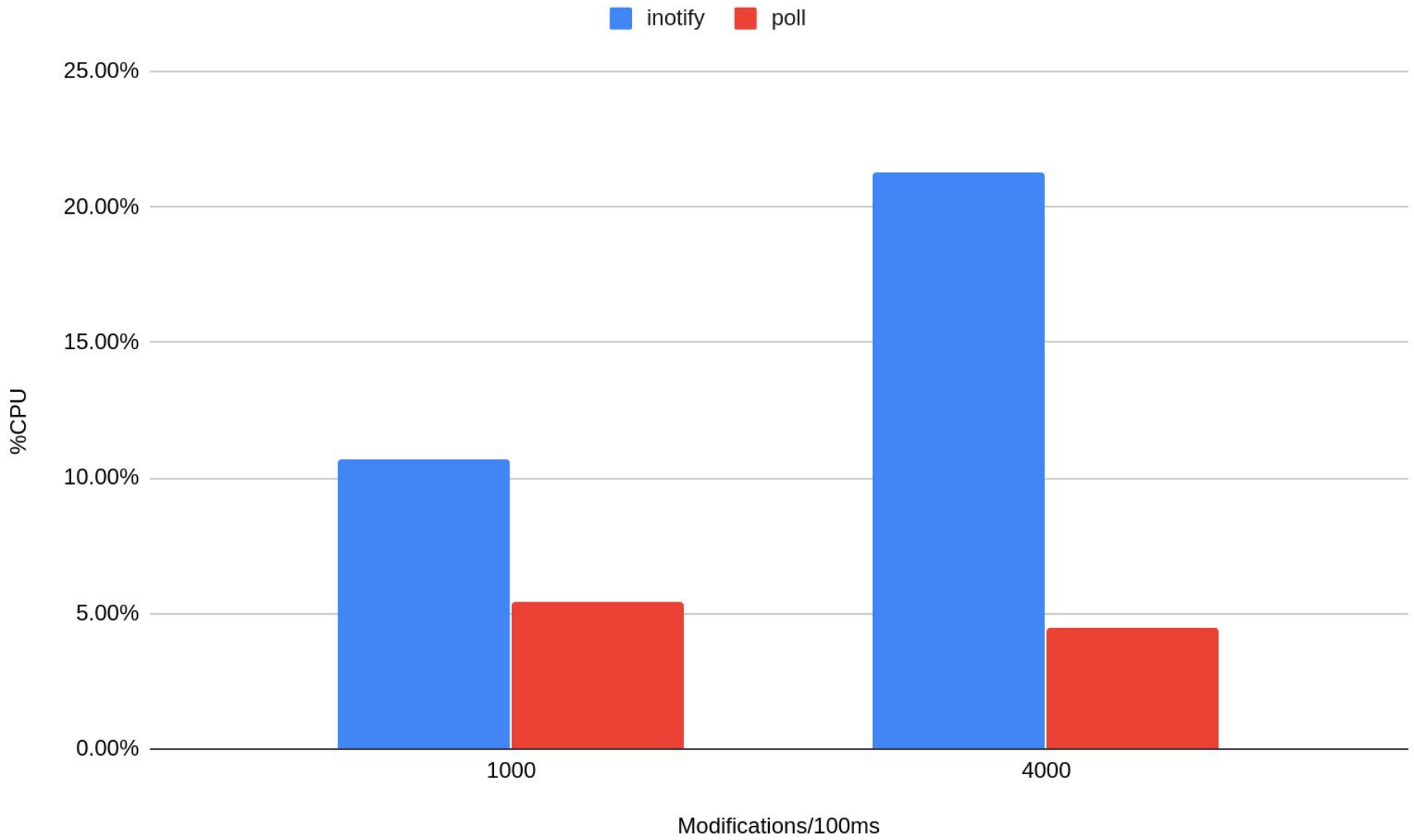
- Не кроссплатформенно / Не POSIX
- Сложность использования



Потребление CPU: Модификации каждую секунду

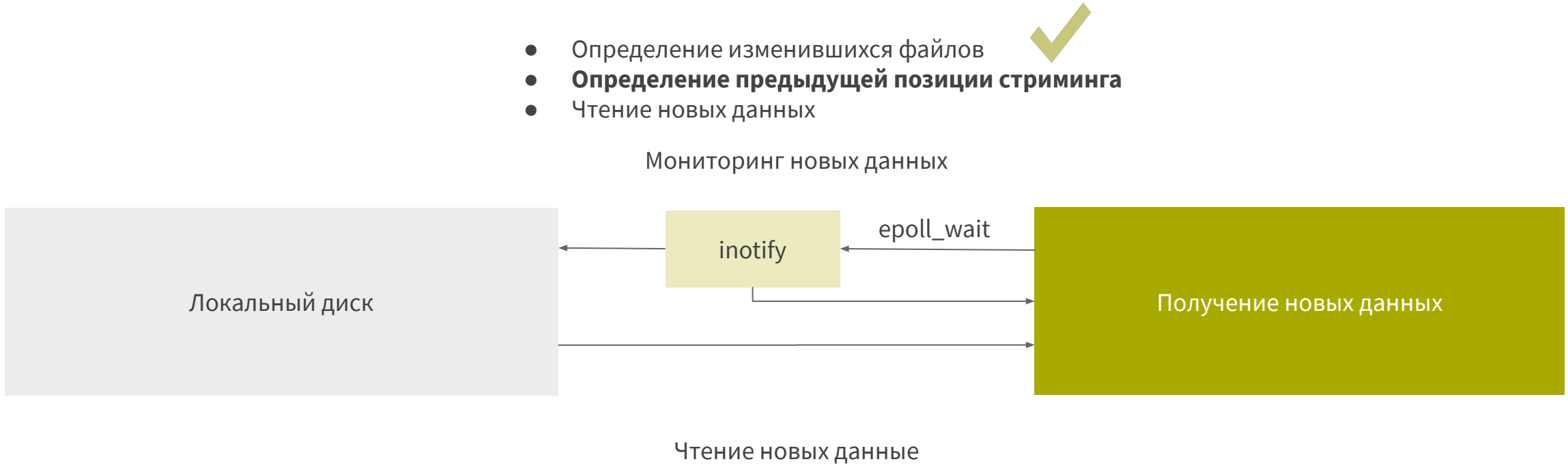


Потребление CPU: Модификации каждые 100ms



Файловое I/O

- Определение изменившихся файлов
- **Определение предыдущей позиции стриминга**
- Чтение новых данных



Определение предыдущей позиции стриминга

Идентификация файла: абсолютный путь

Преимущества

- Простота использования

Идентификация файла: абсолютный путь

Преимущества

- Простота использования

Недостатки:

- Не идецифицирует файл

Идентификация файла: хэш

Преимущества

- Портируемость
- Простота реализации

Идентификация файла: хэш

Преимущества

- Портируемость
- Простота реализации

Недостатки:

- Одинаковые файлы неразличимы
- Производительность

Идентификация файла: POSIX file serial number

Документация POSIX*:

3.176 File Serial Number

A per-file system unique identifier for a file.

* https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html

Идентификация файла: POSIX file serial number

Документация POSIX*:

The *<sys/stat.h>* header shall define the structure of the data returned by the functions *fstat()*, *lstat()*, and *stat()*.

The **stat** structure shall contain at least the following members:

[...]
ino_t st_ino File serial number.

* https://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/stat.h.html

Идентификация файла: inode number

man stat(2):

```
struct stat {
    //...
    ino_t st_ino;    /* Inode number */
    //...
};
```

man inode(7):

```
stat.st_ino; statx.stx_ino
```

Each file in a filesystem has a unique inode number. **Inode numbers are guaranteed to be unique only within a filesystem** (i.e., the same inode numbers may be used by different filesystems, which is the reason that hard links may not cross filesystem boundaries). This field contains the file's inode number.

Inode number: JDK java.nio.file

Схема API:

```
1    package java.nio.file;
2
3    public final class Files {
4
5         public static Object getAttributes(Path path, String attribute,
6                                            LinkOption... options)
7              throws IOException
8
9    }
```

Пример:

```
1    public static Long getInodeNumber(Path path) throws IOException{
2        return (Long) Files.getAttributes(path, “unix:ino”);
3    }
```

Inode number: JDK java.nio.file

UnixNativeDispatcher.c: *

```
1       JNIEXPORT void JNICALL
2       Java_sun_nio_fs_UnixNativeDispatcher_stat0(JNIEnv *env, jclass this,
3           jlong pathAddress, jobject attrs)
4       {
5           int err;
6           struct stat64 buf;
7           const char *path = (const char *)jlong_to_ptr(pathAddress);
8
9           RESTARTABLE(stat64(path, &buf), err);
10          if (err == -1) {
11              throwUnixException(env, errno);
12          } else {
13              prepAttributes(env, &buf, attr);
14          }
15      }
```

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/unix/native/libnio/fs/UnixNativeDispatcher.c#l498

Inode number: JDK java.nio.file

Преимущества:

- Уникальный файловый идентификатор
- Простота

Inode number: JDK java.nio.file

Преимущества:

- Уникальный файловый идентификатор
- Простота

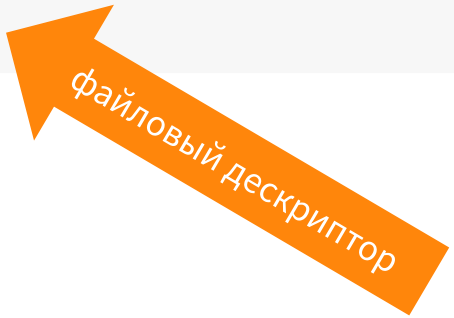
Недостатки:

- **Используется путь к файлу**
- Портируемость

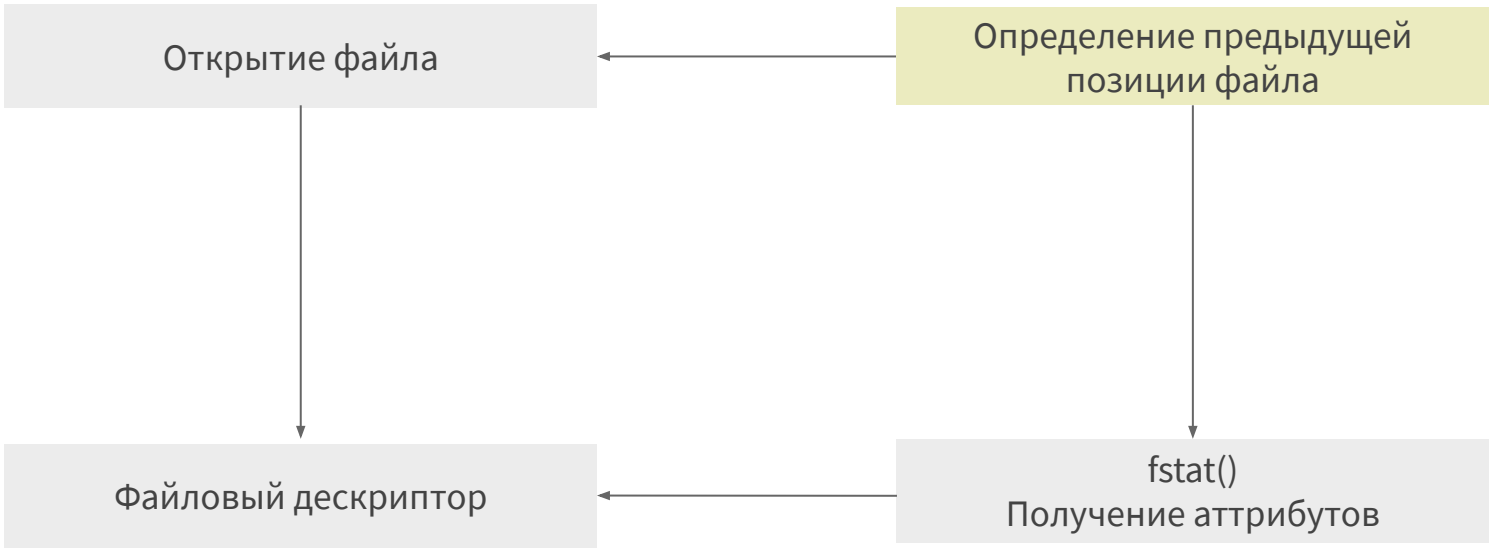
Inode number: POSIX fstat

Схема API:

```
1   #include <sys/stat.h>
2
3   int stat(const char *pathname, struct stat *statbuf);
4
5   int fstat(int fd, struct stat *statbuf);
```



POSIX fstat: схема использования



Inode number: POSIX fstat

Преимущества:

- Отсутствие Race Condition

Inode number: POSIX fstat

Преимущества:

- Отсутствие Race Condition

Недостатки:

- Требует файловый дескриптор

POSIX fstat: файловый дескриптор

Приватное поле:

```
1     package java.io;
2
3     public final class FileDescriptor {
4         private int fd;
5     }
```

Реализация в one-nio:

```
1     package one.nio.os;
2
3     public final class Mem {
4         private static final Field fdField = JavaInternals.getField(FileDescriptor.class, “fd”);
5
6         public static int getFd(FileDescriptor fd){
7             try {
8                 return fdField.getInt(fd);
9             } catch (IllegalAccessException e) {
10                throw new IllegalStateException(“Should not happen”);
11            }
12        }
13    }
```

POSIX fstat: файловый дескриптор

Нативный метод:

```
1    package com.company;
2
3    public class PosixUtils {
4        public static native int openRO(String path);
5    }
```

Реализация:

```
1    JNIEXPORT void JNICALL
2    Java_com_company_PosixUtils_openRO(JNIEnv *env, jclass jc, jstring path){
3        const char *native_path = (*env)->GetStringUTFChars(env, path, NULL);
4        int fd = open(native_path, O_RDONLY);
5        (*env)->ReleaseStringUTFChars(env, path, native_path);
6        return fd;
7    }
8
```

POSIX fstat: производительность

```
#include <immintrin.h>
#include <x86intrin.h>

#define warmup(iters, foo, ...)                                 \
do {                                                            \
    unsigned _iterations = iters;                               \
    while(_iterations --> 0) {                                  \
        foo(__VA_ARGS__);                                       \
    }                                                           \
} while (0)

#define single_shot_measure(foo, ...)                           \
({                                                              \
    unsigned _tsc_aux_ignored = 0;                              \
    uint64_t _start = __rdtsc();                                \
    foo(__VA_ARGS__);                                           \
    uint64_t _end = __rdtscp(&_tsc_aux_ignored);                \
    _end - _start;                                              \
})

static inline void cache_flush(const void *ptr, size_t sz) {
    size_t line_offset = 0;
    while(line_offset < sz) {
        _mm_clflush((const char *) ptr + line_offset);
        line_offset += 64;
    }
}
```

POSIX fstat: производительность

```c
#include <immintrin.h>
#include <x86intrin.h>

#define warmup(iters, foo, ...)                                 \
do {                                                            \
    unsigned _iterations = iters;                               \
    while(_iterations --> 0) {                                  \
        foo(__VA_ARGS__);                                       \
    }                                                           \
} while (0)

#define single_shot_measure(foo, ...)                           \
({                                                              \
    unsigned _tsc_aux_ignored = 0;                              \
    uint64_t _start = __rdtsc();                                \
    foo(__VA_ARGS__);                                           \
    uint64_t _end = __rdtscp(&_tsc_aux_ignored);                \
    _end - _start;                                              \
})

static inline void cache_flush(const void *ptr, size_t sz) {
    size_t line_offset = 0;
    while(line_offset < sz) {
        _mm_clflush((const char *) ptr + line_offset);
        line_offset += 64;
    }
}
```

i7-8550U KbL, Linux Kernel 5.3.0,
горячий dentry cache

taskset -c 1 ./benchmarks

	Reference cycles
stat_noflush	2010
stat	2486
open_fstat_noflush	4265
open_fstat	4699
fstat_noflush	1326
fstat	1803
pread_8KB_noflush	2042
pread_8KB	10282
pread_8B_noflush	1648
pread_8B	1680

POSIX fstat: производительность

```
-   34,74%     0,26%  bin      [kernel.kallsyms]  [k] user_path_at_empty
   - 34,47% user_path_at_empty
      - 20,97% filename_lookup
         - 18,95% path_lookupat
            + 11,02% link_path_walk.part.33
            + 2,64% walk_component
            + 2,34% complete_walk
            + 1,77% path_init
         + 1,00% putname
      + 13,19% getname_flags
```

Обход каждого из компонентов пути

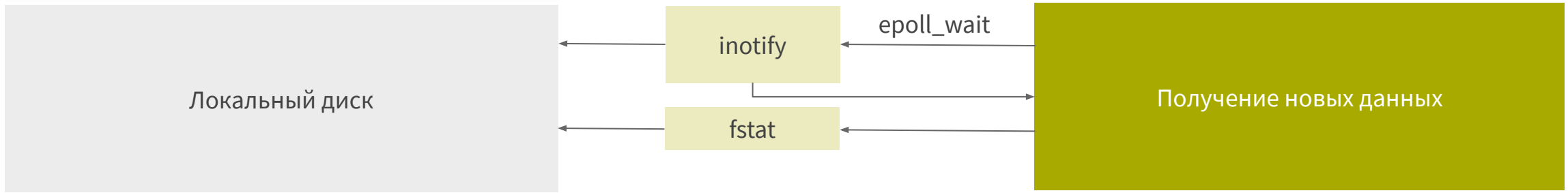
i7-8550U KbL, Linux Kernel 5.3.0,
горячий dentry cache

taskset -c 1 ./benchmarks

	Reference cycles
stat_noflush	2010
stat	2486
open_fstat_noflush	4265
open_fstat	4699
fstat_noflush	1326
fstat	1803
pread_8KB_noflush	2042
pread_8KB	10282
pread_8B_noflush	1648
pread_8B	1680

Файловое I/O

- Определение изменившихся файлов
- Определение предыдущей позиции стриминга
- **Чтение новых данных**



Чтение новых данные

Чтение новых данных

- Пропускная способность
- Использование on-heap byte[] для последующей обработки
  - GZIP до JDK11
- Использование on-heap byte[] для последующей отправки
  - GCP connector

Чтение новых данных

Zero-copy file

java.nio.channels.FileChannel

java.io.InputStream

Чтение новых данных

Zero-copy file

Чтение новых данных: Zero-copy file

Схема API:

```
package java.nio.channels;

public abstract class FileChannel {

      public abstract long transferTo(long position, long count,
                                      WritableByteChannel target)
          throws IOException;

}
```

Пример:

```
FileChannel from = new FileInputStream("/path/to/file").getChannel();
SocketChannel to = SocketChannel.open(new InetSocketAddress("host", 7777));
long position = 0;
long count = 8192;
from.transferTo(position, count, to);
```

Чтение новых данных: Zero-copy file

FileChannelImpl.c: *

```
1       JNIEXPORT void JNICALL
2       Java_sun_nio_ch_FileChannelImpl_transferTo0(JNIEnv *env, jobject this,
3                                                   jobject srcFD0,
4                                                   jlong position, jlong count,
5                                                   jobject srcFD0)
6       {
7           jint srcFD = fdval(env, srcFD0);
8           jint dstFD = fdval(env, dstFD0);
9
10          #if defined(__linux__)
11              off64_t offset = (off64_t) position;
12              jlong n = sendfile64(dstFD, srcFD, &offset, (size_t)count);
```

Zero-copy file

* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/unix/native/libnio/ch/FileChannelImpl.c#l124

Чтение новых данных: Zero-copy file

Преимущества:

- Позволяет избежать копирования

Чтение новых данных: Zero-copy file

Преимущества:

- Позволяет избежать копирования

Недостатки:

- Передает файлы as-is

Чтение новых данных

java.nio.channels.FileChannel

Чтение новых данных: FileChannel.read

Схема API:

```
1    package java.nio.channels;
2
3    public abstract class FileChannel {
4
5         public abstract int read(ByteBuffer dst, long position) throws IOException;
6
7    }
```

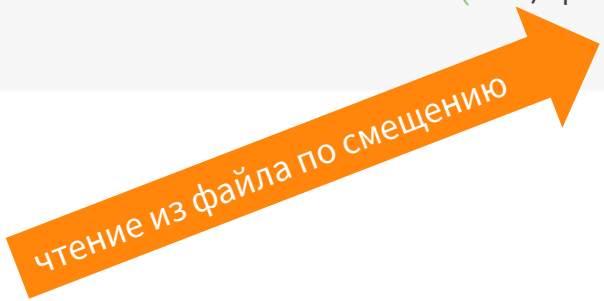
Пример:

```
1    FileChannel fileChannel = new FileInputStream("/path/to/file").getChannel();
2    ByteBuffer byteBuffer = ByteBuffer.allocateDirect(8192);
3    //byteBuffer.flip();
4    long readFrom = 0;
5    fileChannel.read(byteBuffer, readFrom);
```

Чтение новых данных: FileChannel.read

FileDispatcherImpl.c: *

```
1       JNIEXPORT void JNICALL
2       Java_sun_nio_ch_FileDispatcherImpl_pread0(JNIEnv *env, jclass clazz, jobject fdo,
3                                 jlong address, jint len, jlong offset)
4       {
5           jint fd = fdval(env, fdo);
6           void *buf = (void *)jlong_to_ptr(address);
7
8           return convertReturnVal(env, pread64(fd, buf, len, offset), JNI_TRUE);
9       }
```



* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/unix/native/libnio/ch/FileDispatcherImpl.c#l88

Чтение новых данных: FileChannel.read

IOUtil.java: *

```
1     package sun.nio.ch;
2
3     public class IOUtil {
4
5          static int read(FileDescriptor fd, ByteBuffer dst, long position,
6                          boolean directIO, int alignment, Native Dispatcher nd){
7
8               if(dst instanceOf DirectBuffer)
9                    return readIntoNativeBuffer(fd, dst, position, directIO, alignment, nd);
10
11              ByteBuffer bb;
12              int rem = dst.remaining();
13
14              bb = Util.getTemporaryDirectBuffer(rem);
15
16              int n = readIntoNativeBuffer(fd, bb, position, directIO, alignment,nd);
17
18              dst.put(bb);
19              //...
20         }
21    }
```

* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/share/classes/sun/nio/ch/IOUtil.java#l226; часть нерелевантного кода опущена

Чтение новых данных: FileChannel.read

Преимущества:

- Чтение данных в ByteBuffer

Чтение новых данных: FileChannel.read

Преимущества:

- Чтение данных в ByteBuffer

Недостатки:

- Копирование при использовании HeapByteBuffer
- *Возможно* дополнительное выделение нативной памяти

Чтение новых данных

java.io.InputStream

Чтение новых данных: java.io.InputStream

Схема API:

```
package java.io;

public abstract class InputStream implements Closeable {

     public int read(byte[] dst, int off,int len) throws IOException {
         //...
     }
}
```

Пример 1:

```
byte[] buf = new byte[8192];
int bytesRead = Files.newInputStream(Paths.get("/path/to/file")).read(buf);
```

Пример 2:

```
byte[] buf = new byte[8192];
int bytesRead = new FileInputStream("/path/to/file").read(buf);
```

Чтение новых данных: java.io.InputStream

io_util.c: *

```c
/* The maximum size of a stack-allocated buffer.
 */
#define BUF_SIZE 8192

jint
readBytes(JNIEnv *env, jobject this, jbyteArray bytes,
          jint off, jint len, jfieldID fid)
{
    char stackBuf[BUF_SIZE];
    char *buf = NULL;

    if (len == 0) {
         return 0;
    } else if (len > BUF_SIZE) {
        buf = malloc(len);
        if (buf == NULL) {
            JNU_ThrowOutOfMemoryError(env, NULL);
            return 0;
        }
    } else {
        buf = stackBuf;
    }
    //...
```

Указатель для чтения из файла

Выделение памяти в C heap

Использования памяти на стеке

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/share/native/libjava/io_util.c; часть нерелевантного кода опущена

Чтение новых данных: java.io.InputStream

Преимущества:

- Использует byte[] для чтения

Чтение новых данных: java.io.InputStream

Преимущества:

- Использует byte[] для чтения

Недостатки:

- Выделение нативной памяти (C heap/C stack)
- Дополнительное копирование в byte[]

Чтение новых данных

Zero-copy file

\+

- Наиболее эффективный

\-

- Передает данные as is

java.nio.channels.FileChannel

\+

- Напрямую читает по сырому off-heap указателю

\-

- Копирование в java heap

java.io.InputStream

\+

- Читает в byte[]

\-

- Копирование в java heap

Чтение новых данных

Zero-copy file

\+

- Наиболее эффективный

\-

- Передает as is без обработки

java.nio.channels.FileChannel

\+

- Напрямую читает по сырому off-heap указателю

\-

- Копирование в java heap

java.io.InputStream

\+

- Читает в byte[]

\-

- Копирование в java heap

???

\+

- Читает в byte[]

\-

- ~~Копирование в java heap~~

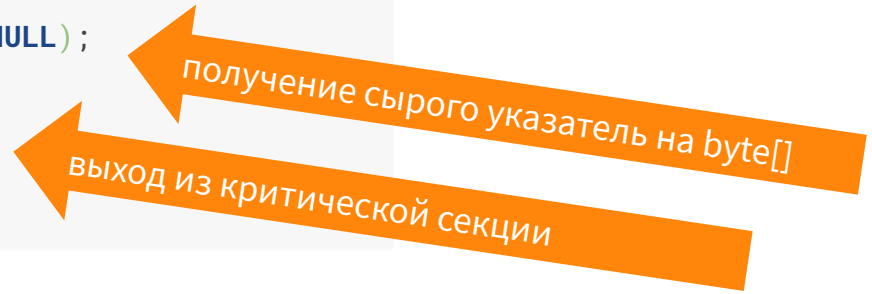
Критическая секция JNI.

Java API:

```
1    package com.company;
2
3    public class PosixUtils {
4        public static native int read(int fd, byte buf[]);
5    }
```

Реализация:

```
1    JNIEXPORT void JNICALL
2    Java_com_company_PosixUtils_read(JNIEnv *env, jclass jc, jint fd, jbyteArray buf){
3        size_t sz = (size_t) (*env)->GetArrayLength(env, buf);
4        void *buf_ptr = (*env)->GetPrimitiveArrayCritical(env, buf, NULL);
5        ssize_t bytesRead = read(fd, buf_ptr, sz);
6        (*env)->ReleasePrimitiveArrayCritical(env, buf, buf_ptr, 0);
7        return (jint) bytesRead;
8    }
```



Критическая секция JNI.

Преимущества:

- Не требует дополнительного копирования

Критическая секция JNI.

Преимущества:

- Не требует дополнительного копирования

Недостатки:

- Требует файловый дескриптор
- Неконтролируемый STW, отсутствие predictable latency

Критическая секция JNI: Неконтролируемый STW

jni.cpp: *

```
1     static oop lock_gc_or_pin_object(JavaThread* thread, jobject obj){
2         if(Universe::heap()->supports_object_pinning()){
3             const oop o = JNIHandles::resolve_non_null(obj);
4             return Universe::heap()->pin_object(thread, o);
5         }
6         //...
7     }
```

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/share/prims/jni.cpp#l3155; часть нерелевантного кода опущена

Критическая секция JNI: Неконтролируемый STW

collectedHeap.hpp: *

```
1     // CollectedHeap
2     //   GenCollectedHeap
3     //     SerialHeap
4     //     CMSHeap
5     //   G1CollectedHeap
6     //   ParallelScavengeHeap
7     //   ShenandoahHeap
8     //   ZCollectedHeap
9     //
10    class CollectedHeap : public CHeapObject<mtInternal> {
11        virtual bool supports_object_pinning() const;
12    }
```

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/share/gc/shared/collectedHeap.hpp#l87; часть нерелевантного кода опущена

Критическая секция JNI: ShenandoahGC

shenandoahHeap.hpp: *

```
1    class ShenandoahHeap : public CollectedHeap {
2        // Shenandoah supports per-object (per-region) pinning
3        bool supports_object_pinning() const { return true; }
4    }
```

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/share/gc/shenandoah/shenandoahHeap.hpp#l147; часть нерелевантного кода опущена

Критическая секция JNI.

Преимущества:

- Не требует дополнительного копирования

Недостатки:

- Требует файловый дескриптор
- Неконтролируемый STW, отсутствие predictable latency
- Баги GCLocker

Критическая секция JNI: Производительность

```
1   package com.company;
2
3   public class PosixUtils {
4       public static native int read(int fd, byte buf[]);
5       public static native int openRO(String path);
6   }
7
8   public static byte buffer[] = new byte[1024 * 128];
9
10  public long measureCriticalRead(String path){
11      int fd = PosixUtils.openRO(path);
12      long start = System.nanoTime();
13      while(PosixUtils.read(fd, buffer) > 0){ }
14      return System.nanoTime() - start;
15  }
16
17  public long measureInputStream(String path){
18      //InputStream is = Files.newInputStream(Paths.get(path));
19      InputStream is = new FileInputStream(path);
20      long start = System.nanoTime();
21      while(is.read(buffer) != -1){ }
22      return System.nanoTime() - start;
23  }
```

Критическая секция JNI: Производительность

```java
package com.company;

public class PosixUtils {
    public static native int read(int fd, byte buf[]);
    public static native int openRO(String path);
}

public static byte buffer[] = new byte[1024 * 128];

public long measureCriticalRead(String path){
    int fd = PosixUtils.openRO(path);
    long start = System.nanoTime();
    while(PosixUtils.read(fd, buffer) > 0){ }
    return System.nanoTime() - start;
}

public long measureInputStream(String path){
    //InputStream is = Files.newInputStream(Paths.get(path));
    InputStream is = new FileInputStream(path);
    long start = System.nanoTime();
    while(is.read(buffer) != -1){ }
    return System.nanoTime() - start;
}
```

i7-8565U WhL, Linux Kernel 5.3.0,
горячий page cache

	InputStream	criticalRead
2 GB	0.36 sec.	0.27 sec.
4 GB	0.59 sec.	0.47 sec.
6 GB	0.95 sec.	0.72 sec.

Критическая секция JNI: анализ.

```
$ sudo perf record --call-graph dwarf -F 9123 -p <app_pid>
```

```
+   99,18%     0,04%  java            [kernel.kallsyms]   [k] entry_SYSCALL_64_after_hwframe
+   99,10%     0,06%  java            [kernel.kallsyms]   [k] do_syscall_64
+   99,03%     0,03%  java            [kernel.kallsyms]   [k] __x64_sys_read
+   99,00%     0,05%  java            [kernel.kallsyms]   [k] ksys_read
+   98,37%     0,07%  java            [kernel.kallsyms]   [k] vfs_read
+   97,54%     0,01%  java            [kernel.kallsyms]   [k] __vfs_read
+   97,53%     0,08%  java            [kernel.kallsyms]   [k] new_sync_read
+   97,42%     0,05%  java            [kernel.kallsyms]   [k] ext4_file_read_iter
+   97,19%     8,41%  java            [kernel.kallsyms]   [k] generic_file_read_iter
+   80,04%     2,25%  java            [kernel.kallsyms]   [k] copy_page_to_iter
-   77,08%    76,78%  java            [kernel.kallsyms]   [k] copy_user_enhanced_fast_string
     76,78% 0x7f23001e1b9a
        JavaCritical_com_test_Main_read
        read (inlined)
        __GI___libc_read (inlined)
        entry_SYSCALL_64_after_hwframe
        do_syscall_64
        __x64_sys_read
        ksys_read
        vfs_read
        __vfs_read
        new_sync_read
        ext4_file_read_iter
        generic_file_read_iter
      - copy_page_to_iter
           76,33% copy_user_enhanced_fast_string
+    7,44%     0,72%  java            [kernel.kallsyms]   [k] pagecache_get_page
+    6,72%     3,34%  java            [kernel.kallsyms]   [k] find_get_entry
+    3,37%     2,84%  java            [kernel.kallsyms]   [k] xas_load
+    1,16%     0,47%  java            [kernel.kallsyms]   [k] _cond_resched
     0,84%     0,41%  java            [kernel.kallsyms]   [k] copyout
+    0,65%     0,54%  java            [kernel.kallsyms]   [k] xas_start
+    0,63%     0,59%  java            [kernel.kallsyms]   [k] rcu_all_qs
+    0,59%     0,51%  java            [kernel.kallsyms]   [k] mark_page_accessed
```

Копирование из ядра пользователю

Копирование памяти в ядре

copy_user_64.S: *

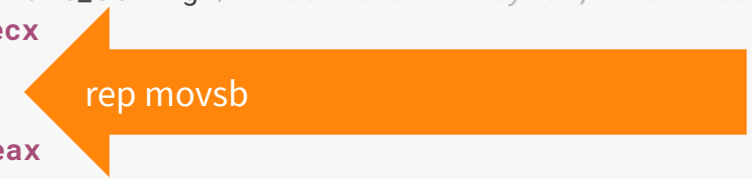
```
1       ENTRY(copy_user_enhanced_fast_string)
2           ASM_STAC
3           cmpl $64,%edx
4           jb .L_copy_short_string /* less then 64 bytes, avoid costly `rep` */
5           movl %edx,%ecx
6       1:  rep
7           movsb
8           xorl %eax,%eax
9           ASM_CLAC
10          ret
11
12          .section .fixup,“ax”
13      12: movl %ecx,%edx          /* ecx is zerorest also */
14          jmp copy_user_enhanced_tail
15          .previous
16          _ASM_EXTABLE_UA(1b, 12b)
17      ENDPROC(copy_user_enhanced_fast_string)
```

* https://elixir.bootlin.com/linux/v5.3/source/arch/x86/lib/copy_user_64.S#L200

Копирование памяти в ядре

copy_user_64.S: *

```
1       ENTRY(copy_user_enhanced_fast_string)
2           ASM_STAC
3           cmpl $64,%edx
4           jb .L_copy_short_string /* less then 64 bytes, avoid costly `rep` */
5           movl %edx,%ecx
6       1:  rep
7           movsb
8           xorl %eax,%eax
9           ASM_CLAC
10          ret
11
12          .section .fixup,“ax”
13      12: movl %ecx,%edx          /* ecx is zerorest also */
14          jmp copy_user_enhanced_tail
15          .previous
16          _ASM_EXTABLE_UA(1b, 12b)
17      ENDPROC(copy_user_enhanced_fast_string)
```



* https://elixir.bootlin.com/linux/v5.3/source/arch/x86/lib/copy_user_64.S#L200

Копирование памяти в ядре: SysV ABI

Спецификация AMD64 SysV ABI: *

A system-call is done via the syscall instruction. The kernel destroys registers `%rcx` and `%r11`.

* https://github.com/hjl-tools/x86-psABI/wiki/X86-psABI

Сравнение rep movsb и AVX2

System.arraycopy *:

```
1      else if (UseAVX == 2) {
2          __ vmovdqu(xmm0, Address(end_from, qword_cout, Address::times_8, -56));
3          __ vmovdqu(Address(end_to, qword_cout, Address::times_8, -56), xmm0);
4          __ vmovdqu(xmm1, Address(end_from, qword_cout, Address::times_8, -24));
5          __ vmovdqu(Address(end_to, qword_cout, Address::times_8, -24), xmm1);
6      }
```

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/cpu/x86/stubGenerator_x86_64.cpp#l1263

Сравнение rep movsb и AVX2

System.arraycopy *:

```
    else if (UseAVX == 2) {
        __ vmovdqu(xmm0, Address(end_from, qword_cout, Address::times_8, -56));
        __ vmovdqu(Address(end_to, qword_cout, Address::times_8, -56), xmm0);
        __ vmovdqu(xmm1, Address(end_from, qword_cout, Address::times_8, -24));
        __ vmovdqu(Address(end_to, qword_cout, Address::times_8, -24), xmm1);
    }
```

memcpy:

```
(gdb) disas __memmove_avx_unaligned_erms
#...
0x000000000018eb1c <+76>:  mov rcx,rdx
0x000000000018eb1f <+79>:  rep movs BYTE PTR es:[rdi],BYTE PTR ds:[rsi]
0x000000000018eb21 <+81>:  ret
#...
0x000000000018eb82 <+178>: cmp rdx,0x1000
0x000000000018eb89 <+185>: ja  0x18eafb <__memmove_avx_unaligned_erms+43>
#...
```

rep movsb

Если размер 4КБ и более

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/cpu/x86/stubGenerator_x86_64.cpp#l1263

Сравнение memcpy и System.arraycopy

```
public class Memcpy {
    public static native void arrayMemcpy(byte src[], int srcOff,
                                           byte dst[], int dstOff,
                                           int len);
}
//8KB, 128KB, 8MB, 16MB, 32MB
@Param({“8192”, “131072”, “8388608”, “16777216”, “33554432”})
public int size;

public byte src[], dst[];

@Setup
public void generateData(){
    //Например, чтение из /dev/urandom
}

@Benchmark
public void arraycopy(){
    System.arraycopy(src, 0, dst, 0, size);
}

@Benchmark
public void memcpy(){
    Memcpy.arrayMemcpy(src, 0, dst, 0, size);
}
```

Сравнение memcpy и System.arraycopy

Реализация Memcpy.arrayMemcpy

```
1     JNIEXPORT void JNICALL
2     Java_com_company_Memcpy_arrayMemcpy(JNIEnv *env, jclass jc,
3                                         jbyteArray src, jint src_off,
4                                         jbyteArray dst, jint dst_off,
5                                         jint len){
6         void *native_arr_src = (*env)->GetPrimitiveArrayCritical(env, src, NULL);
7         void *native_arr_dst = (*env)->GetPrimitiveArrayCritical(env, dst, NULL);
8         memcpy((char *) native_arr_dst + dst_off, (char *) native_arr_src + src_off, (size_t) len);
9         (*env)->ReleasePrimitiveArrayCritical(env, dst, native_arr_dst, 0);
10        (*env)->ReleasePrimitiveArrayCritical(env, src, native_arr_src, 0);
11    }
12
13    JNIEXPORT void JNICALL
14    JavaCritical_com_company_Memcpy_arrayMemcpy(jint src_len, jbyte *src, jint src_off,
15                                                jint dst_len, jbyte *dst, jint dst_off,
16                                                jint len){
17        memcpy((char *) dst + dst_off, (char *) src + src_off, (size_t) len);
18    }
```

Сравнение memcpy и System.arraycopy

```
1   public class Memcpy {
2       public static native void arrayMemcpy(byte src[], int srcOff,
3                                              byte dst[], int dstOff,
4                                              int len);
5   }
6   //8KB, 128KB, 8MB, 16MB, 32MB
7   @Param({"8192", "131072", "8388608", "16777216", "33554432"})
8   public int size;
9
10  public byte src[], dst[];
11
12  @Setup
13  public void generateData(){
14      //Например, чтение из /dev/urandom
15  }
16
17  @Benchmark
18  public void arraycopy(){
19      System.arraycopy(src, 0, dst, 0, size);
20  }
21
22  @Benchmark
23  public void memcpy(){
24      Memcpy.arrayMemcpy(src, 0, dst, 0, size);
25  }
```

i7-8565U WhL

	memcpy, µs	System.arraycopy, µs
8 KB	0.172	0.132
128 KB	3.439	4.171
8 MB	542.921	959.101
16 MB	1393.213	2209.657
32 MB	3128.653	4702.299

60-90 %

Non-Temporal Stores

```
(gdb) p __x86_shared_non_temporal_threshold
$1 = 6291356 #6MB
```

```
(gdb) disas __memmove_avx_unaligned_erms
0x000000000018eafb <+43>:  cmp rdx,QWORD PTR [rip+0x261b76] #0x3f0678 <__x86_shared_non_temporal_threshold>
0x000000000018eb1f <+79>:  jae 0x18ec2d
#...
0x000000000018edb1 <+737>:   vmovdqu  ymm0,YMMWORD PTR [rsi]
0x000000000018edb5 <+741>:   vmovdqu  ymm1,YMMWORD PTR [rsi+0x20]
0x000000000018edba <+746>:   vmovdqu  ymm2,YMMWORD PTR [rsi+0x40]
0x000000000018edbf <+751>:   vmovdqu  ymm3,YMMWORD PTR [rsi+0x60]
0x000000000018edc4 <+756>:   add      rsi,0x80
0x000000000018edcb <+763>:   sub      rdx,0x80
0x000000000018edd2 <+770>:   vmovntdq YMMWORD PTR [rdi],ymm0
0x000000000018edd6 <+774>:   vmovntdq YMMWORD PTR [rdi+0x20],ymm1
0x000000000018eddb <+779>:   vmovntdq YMMWORD PTR [rdi+0x40],ymm2
0x000000000018ede0 <+784>:   vmovntdq YMMWORD PTR [rdi+0x60],ymm3
0x000000000018ede5 <+789>:   add      rdi,0x80
0x000000000018edec <+796>:   cmp      rdx,0x80
0x000000000018edf3 <+803>:   ja       0x18ed95 <__memmove_avx_unaligned_erms+709>
0x000000000018edf5 <+805>:   sfence
```

vmovntdq - Non-Temporal Store

sfence обязателен

Non-Temporal Stores

Официальное описание Non-Temporal записей *

8.4.1 The Non-temporal Store Instructions

This section describes the behavior of streaming stores and reiterates some of the information presented in the previous section.

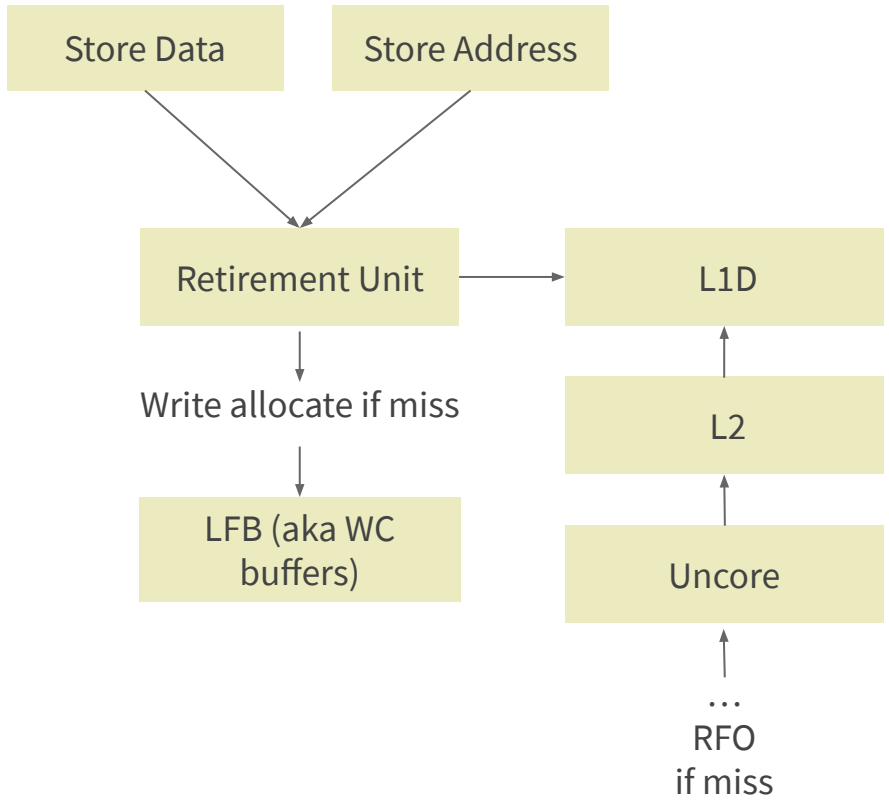
In Streaming SIMD Extensions, the MOVNTPS, MOVNTPD, MOVNTQ, MOVNTDQ, MOVNTI, MASKMOVQ and MASKMOVDQU instructions are streaming, non-temporal stores. With regard to memory characteristics and ordering, they are similar to the Write-Combining (WC) memory type:

- Write combining — Successive writes to the same cache line are combined.
- Write collapsing — Successive writes to the same byte(s) result in only the last write being visible.
- Weakly ordered — No ordering is preserved between WC stores or between WC stores and other loads or stores.
- Uncacheable and not write-allocating — Stored data is written around the cache and will not generate a read-for-ownership bus request for the corresponding cache line.

* Intel Software Optimization Manual/8.4.1

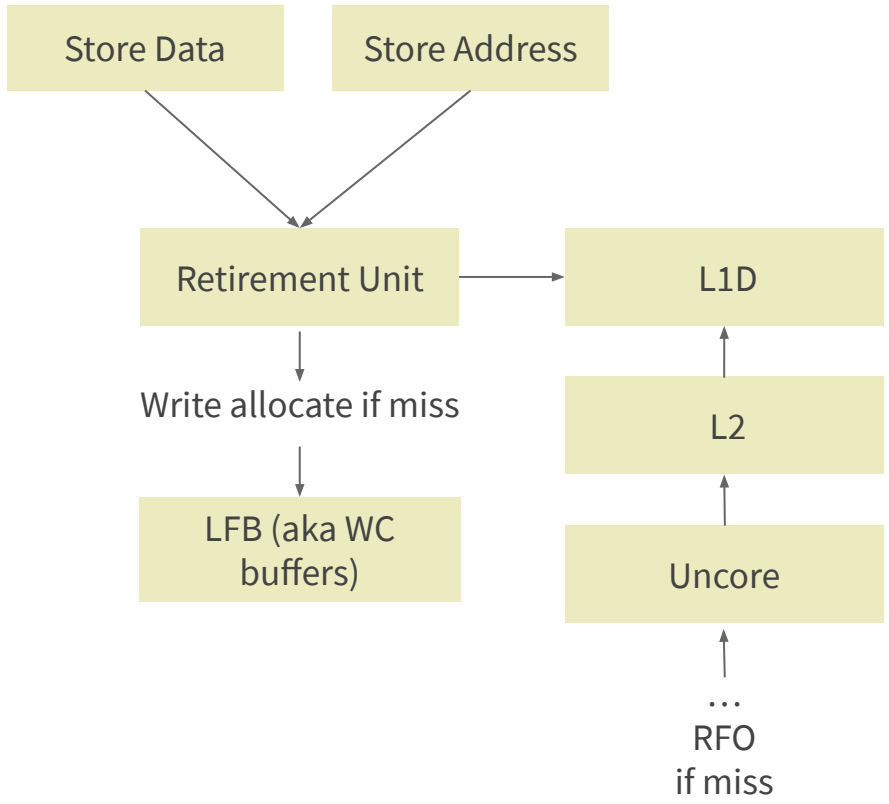
Non-Temporal Stores: Сравнение с Write-Back

- Write-Back Stores

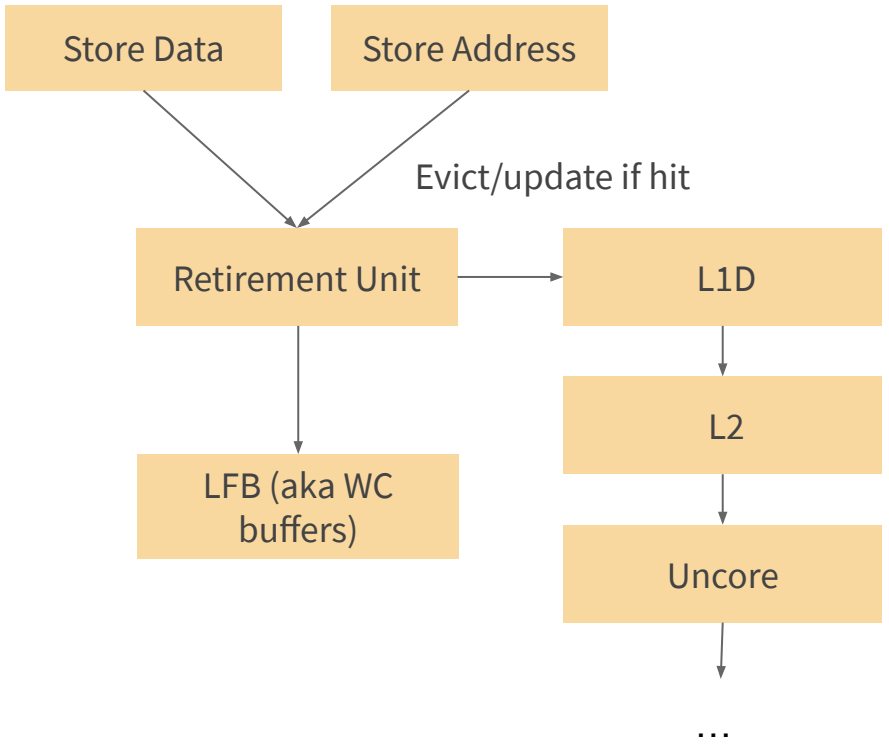


Non-Temporal Stores: Сравнение с Write-Back

- Write-Back Stores

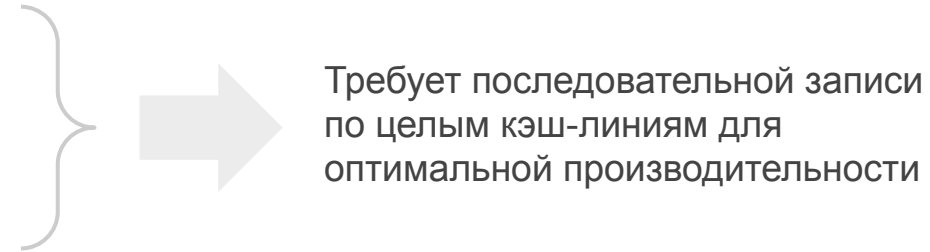


- Non-Temporal Stores



Non-Temporal Stores: Summary

- Не кешируются
  - Нет RFO трафика
  - Evict/Update в случае write hit
- Write-Combining
  - Записываются в память по целым кэш линиям
  - Записываются в память по кускам 8 байт если целую кэш линию собрать не удалось и LFB переполнились
- Weakly-ordered
  - Требует sfence для гарантий видимости



Non-Temporal Stores + mmap: Производительность

```
1   private static final String FILE_NAME = "/path/to/file.txt";
2   private static final int SIZE = 8 * 1024 * 1024;
3   private static final byte buffer[] = new byte[SIZE];
4
5   private static native void memcpy(long addrSrc, byte bufferDst[]);
6
7   public static long copyMmap(String path) throws Exception {
8       long alignedSize = Files.size(Paths.get(FILE_NAME)) & -SIZE;
9       one.nio.mem.MappedFile mf = new one.nio.mem.MappedFile(
10            FILE_NAME,
11            alignedSize,
12            one.nio.mem.MappedFile.MAP_RO
13      );
14      long addr = mf.getAddr();
15      long bytesRead = 0;
16      long start = System.nanoTime();
17      while(bytesRead < alignedSize) {
18          memcpy(addr + bytesRead, buffer);
19          bytesRead += SIZE;
20      }
21      long end = System.nanoTime();
22      mf.close();
23      return end - start;
24  }
```

Вызов sys_mmap

Вызов sys_munmap

Non-Temporal Stores + mmap: Производительность

```
1   private static final String FILE_NAME = "/path/to/file.txt";
2   private static final int SIZE = 8 * 1024 * 1024;
3   private static final byte buffer[] = new byte[SIZE];
4
5   private static native void memcpy(long addrSrc, byte bufferDst[]);
6
7   public static long copyMmap(String path) throws Exception {
8       long alignedSize = Files.size(Paths.get(FILE_NAME)) & -SIZE;
9       one.nio.mem.MappedFile mf = new one.nio.mem.MappedFile(
10            FILE_NAME,
11            alignedSize,
12            one.nio.mem.MappedFile.MAP_RO
13      );
14      long addr = mf.getAddr();
15      long bytesRead = 0;
16      long start = System.nanoTime();
17      while(bytesRead < alignedSize) {
18          memcpy(addr + bytesRead, buffer);
19          bytesRead += SIZE;
20      }
21      long end = System.nanoTime();
22      mf.close();
23      return end - start;
24  }
```

Вызов sys_mmap

Вызов sys_munmap

i7-8565U WhL, Linux Kernel 5.3.0,
горячий page cache

- Не учитывая mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.72	0.72

Non-Temporal Stores + mmap: Производительность

```
1   private static final String FILE_NAME = "/path/to/file.txt";
2   private static final int SIZE = 8 * 1024 * 1024;
3   private static final byte buffer[] = new byte[SIZE];
4
5   private static native void memcpy(long addrSrc, byte bufferDst[]);
6
7   public static long copyMmap(String path) throws Exception {
8       long alignedSize = Files.size(Paths.get(FILE_NAME)) & -SIZE;
9       long start = System.nanoTime();
10      one.nio.mem.MappedFile mf = new one.nio.mem.MappedFile(
11            FILE_NAME,
12            alignedSize,
13            one.nio.mem.MappedFile.MAP_RO
14      );
15      long addr = mf.getAddr();
16      long bytesRead = 0;
17      while(bytesRead < alignedSize) {
18          Memcpy.copyAddr(addr + bytesRead, buffer);
19          bytesRead += SIZE;
20      }
21      mf.close();
22      long end = System.nanoTime();
23      return end - start;
24  }
```

Вызов sys_mmap

Вызов sys_munmap

i7-8565U WhL, Linux Kernel 5.3.0,
горячий page cache

- Не учитывая mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.72	0.72

- Учитывая mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.82	0.72

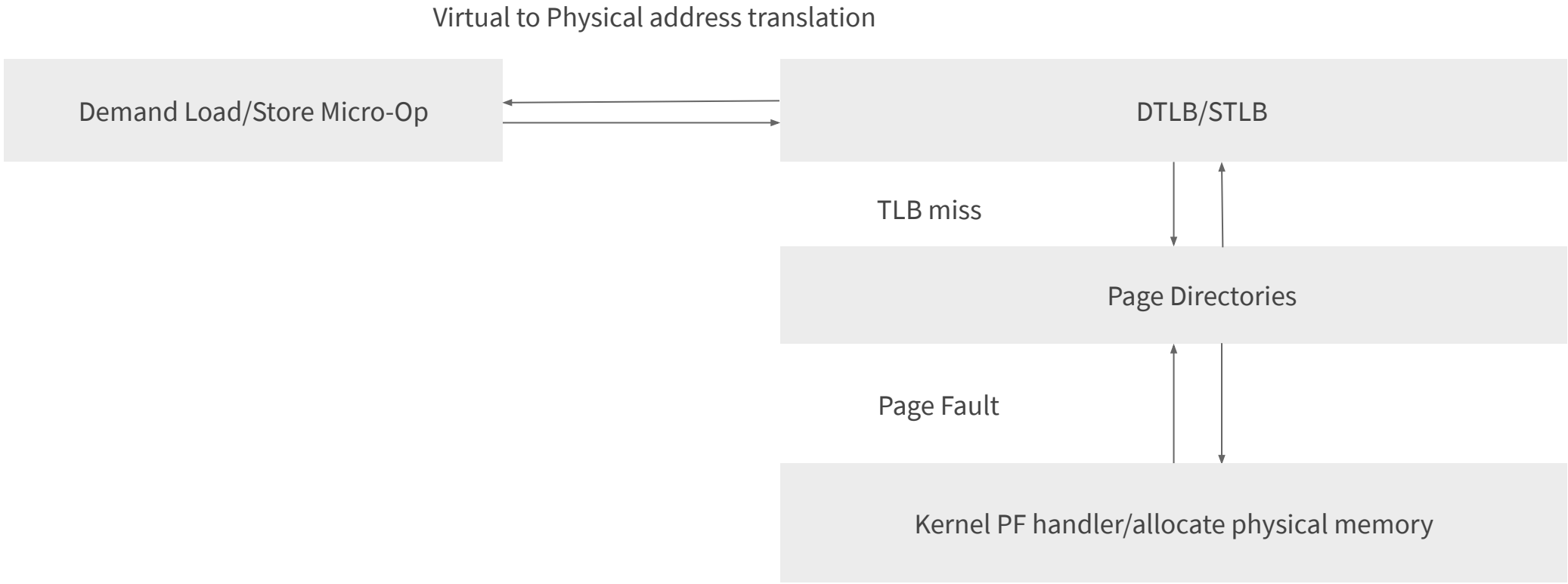
Non-Temporal Stores + mmap: Производительность

```
$ sudo perf record --call-graph dwarf -F 9123 -p <app_pid>
```

```
-   86,90%    65,47%  java             libc-2.27.so          [.] __memmove_avx_unaligned_erms
   - 62,60% 0x7f93c7394e99
        Java_com_test_Memcpy_copyAddr
        memcpy (inlined)
        __memmove_avx_unaligned_erms
   - 21,43% __memmove_avx_unaligned_erms
      - 18,90% page_fault
         + 18,71% do_page_fault
      + 0,98% error_entry
      2,86% page_fault
+   84,04%     0,00%  java             [JIT] tid 10832       [.] 0x00007f93c7394e99
+   84,03%     0,00%  java             libmemcpy_java.so     [.] Java_com_test_Memcpy_copyAddr
+   84,03%     0,00%  java             libmemcpy_java.so     [.] memcpy (inlined)
+   21,77%     0,18%  java             [kernel.kallsyms]     [k] page_fault
+   18,72%     0,03%  java             [kernel.kallsyms]     [k] do_page_fault
```

Page Fault при первом доступе

Non-Temporal Stores + mmap: Page Fault



Non-Temporal Stores + mmap: Производительность

```
1   private static final String FILE_NAME = "/path/to/file.txt";
2   private static final int SIZE = 8 * 1024 * 1024;
3   private static final byte buffer[] = new byte[SIZE];
4
5   public static long copyMmapNoPageFault(long size,
6                                          long addr) throws Exception {
7       long start = System.nanoTime();
8       long bytesRead = 0;
9       while(bytesRead < alignedSize) {
10          Memcpy.copyAddr(addr + bytesRead, buffer);
11          bytesRead += SIZE;
12      }
13      mf.close();
14      long end = System.nanoTime();
15      return end - start;
16  }
```

i7-8565U WhL, Linux Kernel 5.3.0,
горячий page cache

- Не учитывая mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.72	0.72

- Учитывая mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.82	0.72

- Многократный доступ

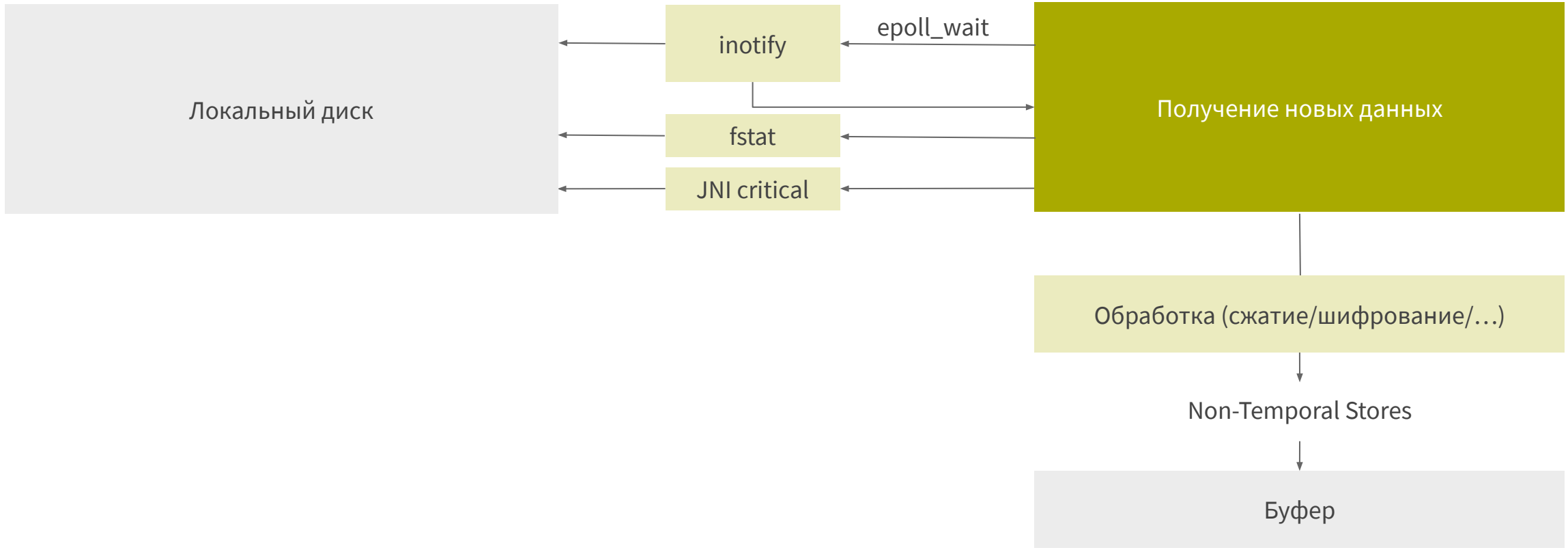
	mmap+NT, sec	criticalRead, sec
5.5GB	0.51	0.72

Non-Temporal Stores + mmap: Summary

- Не подходит для чтения из файла в общем случае
  - Page Fault
  - Cache miss при последующем доступе
- *Может быть* полезен при многократном доступе

Файловое I/O

- Определение изменившихся файлов
- Определение предыдущей позиции стриминга
- Чтение новых данных



Заключение

- Для максимально эффективной реализации I/O-интенсивного сервиса могут потребоваться платформо-зависимые вставки
- При замерах производительности необходимо учитывать влияние железа, ядра ОС и JVM.

Q&A

Grid Dynamics

Thank you!

www.griddynamics.com