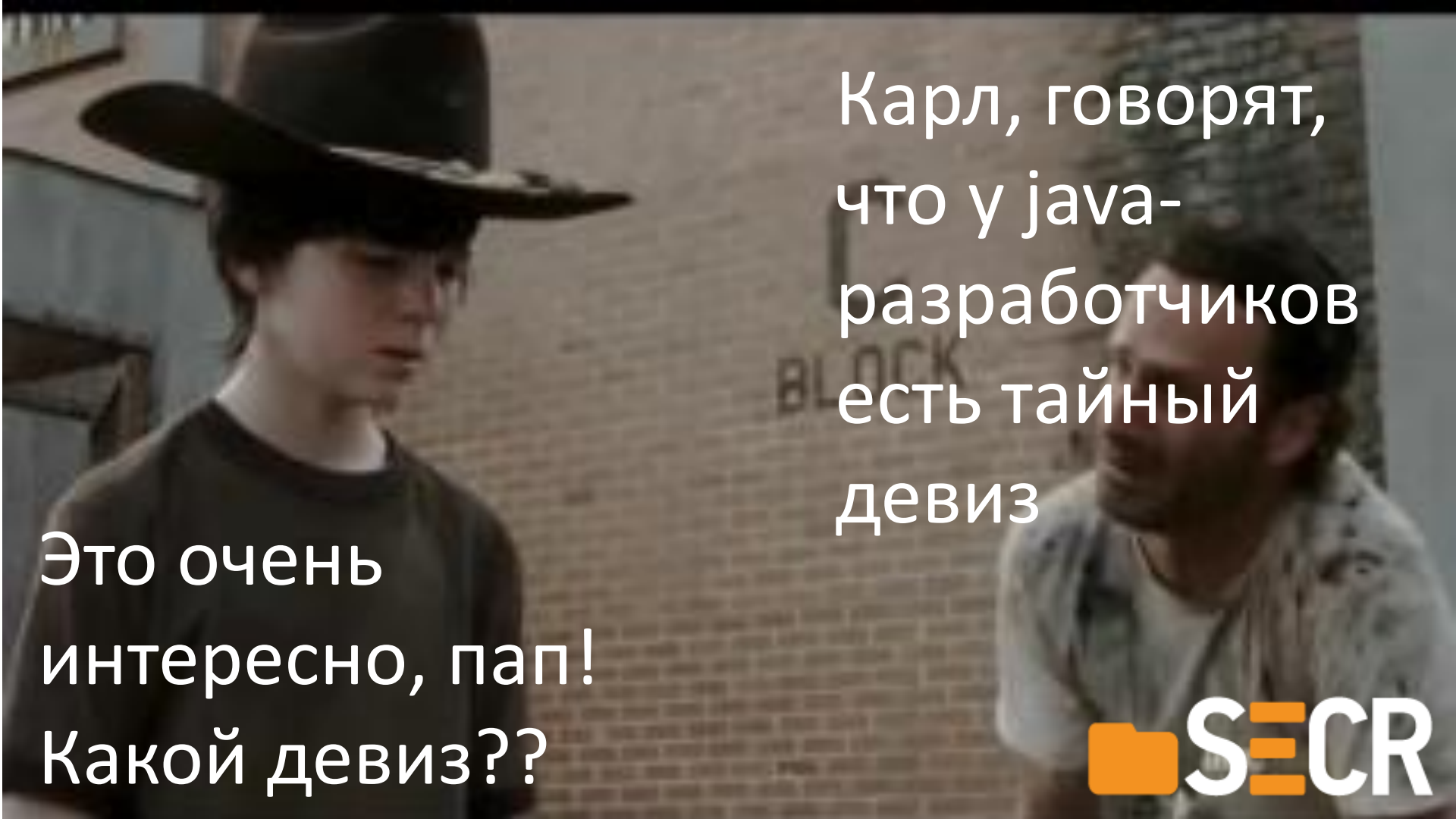


**Проектирование, разработка и эксплуатация
высоконагруженной системы онлайн репликации >500 ТБ и
1 млрд. файлов клиентов между континентами: Amazon S3
(США) – облако Mail.ru (Россия)**


Александр Сербул
Руководитель направления



Карл, говорят,
что у java-
разработчиков
есть тайный
девиз

Это очень
интересно, пап!
Какой девиз??

 **SECR**



«Зачем делать
просто, если
МОЖНО сложно?»

 **SEC**

Немного об архитектуре проекта

1. Компании пользуются закрытым облачным «сайтом» для организации работы сотрудников
2. Компания использует уже около 1000 таблиц в MySQL
3. Сотрудники компании создают и загружают тысячи файлов
4. Активных компаний – сотни тысяч
5. Данных чертовски много, бигдатой просто заваливает - информацию нужно бекапить, реплицировать, очищать...



 **SEC**

Почему мы «тусим» в облаках Amazon и mail.ru?

1. Можно быстро поднять машины любого типа на любом железе и погасить (Amazon EC2)
2. Можно легко бэкапить диски машин (Amazon EBS)
3. Удобные облачные объектные хранилища неограниченного объема (HTTP)



Структура любого AWS SDK...

AWS SDK for PHP 3.x

Namespace

Class

Namespaces

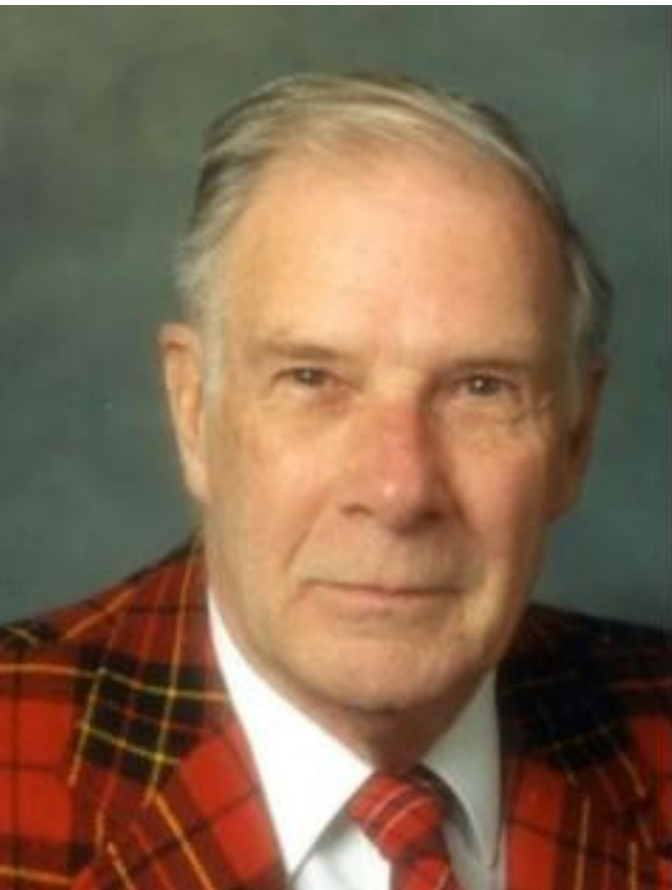
- Aws
- Acm
- ACMPCA
- AlexaForBusiness
- Api
- ApiGateway
- ApplicationAutoScaling
- ApplicationDiscoveryService
- Appstream
- AppSync
- Athena
- AutoScaling
- AutoScalingPlans
- Batch
- Budgets
- Cloud9
- CloudDirectory
- CloudFormation
- CloudFront

Service APIs

Service Name	Client Class	API Version
AWS Certificate Manager	Aws\Acm\AcmClient	2015-12-08
AWS Certificate Manager Private Certificate Authority	Aws\ACMPCA\ACMPCAClient	2017-08-22
Alexa For Business	Aws\AlexaForBusiness\AlexaForBusinessClient	2017-11-09
Amazon API Gateway	Aws\ApiGateway\ApiGatewayClient	2015-07-09
Application Auto Scaling	Aws\ApplicationAutoScaling\ApplicationAutoScalingClient	2016-02-06
AWS Application Discovery Service	Aws\ApplicationDiscoveryService\ApplicationDiscoveryServiceClient	2015-11-01
Amazon AppStream	Aws\Appstream\AppstreamClient	2016-12-01
AWS AppSync	Aws\AppSync\AppSyncClient	2017-07-25
Amazon Athena	Aws\Athena\AthenaClient	2017-05-18
Auto Scaling	Aws\AutoScaling\AutoScalingClient	2011-01-01
AWS Auto Scaling Plans	Aws\AutoScalingPlans\AutoScalingPlansClient	2018-01-06

Языки AWS SDK и их применение в классах задач

Язык SDK	Класс задач в Битрикс24
Java	для сложной и высоконагруженной, многопоточной бизнес-логики
.NET	
Node.js	часть триггеров
PHP	автоматизация бэкапов, системные скрипты, аналитика...
Python	
Ruby	
Go	
C++	
Консольная утилита	для системного администрирования



Mathematicians stand on each others' shoulders and computer scientists stand on each others' toes.

— *Richard Hamming* —

AZ QUOTES

Современный код





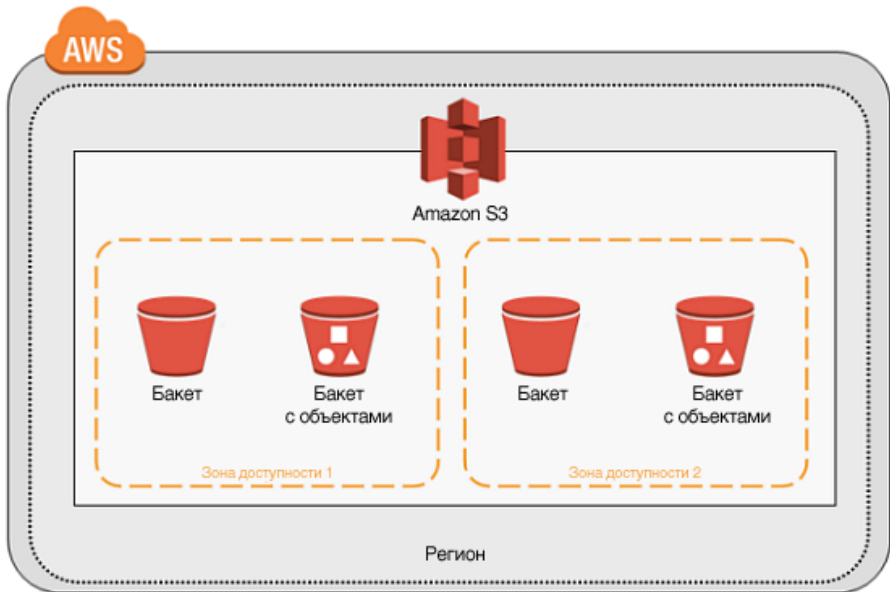
современный Код

К хорошему - привыкаешь

1. Накопилось более 500 ТБ
файлов клиентов в Amazon S3
2. Почти миллиард файлов
3. В США..., Карл (до закона о
персданных)



Amazon S3 – изнутри



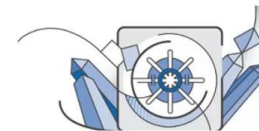
Choice of storage classes on S3



Standard



Standard - Infrequent Access



Amazon Glacier

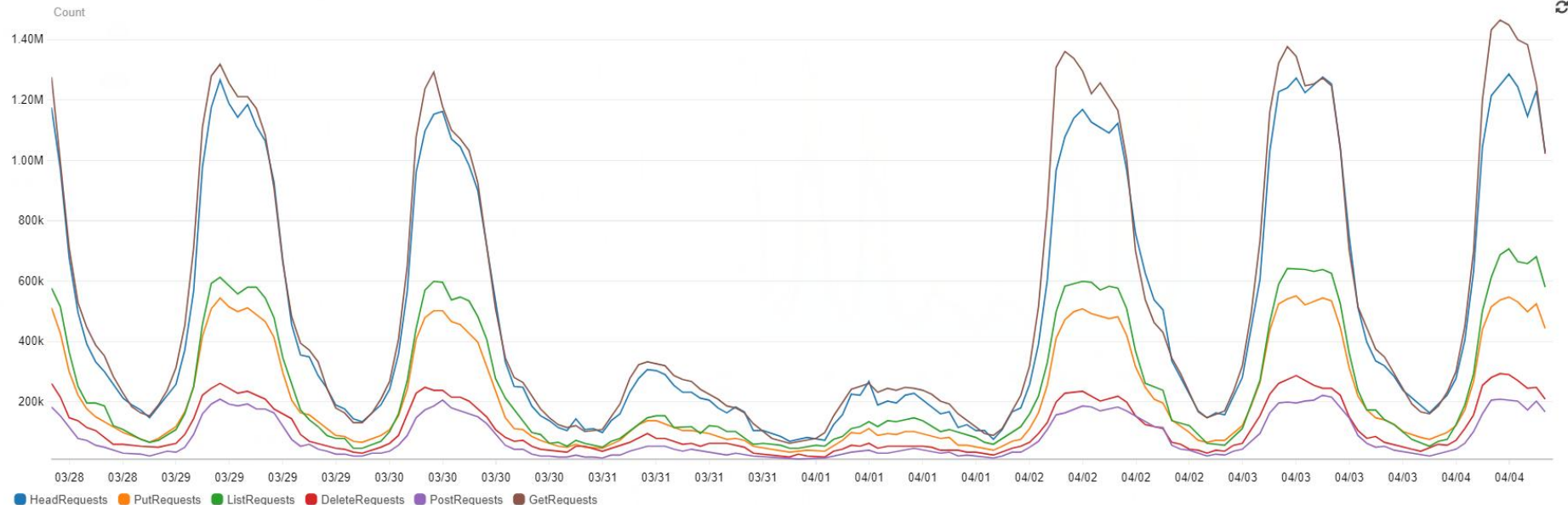
Active data

Infrequently accessed data

Archive data

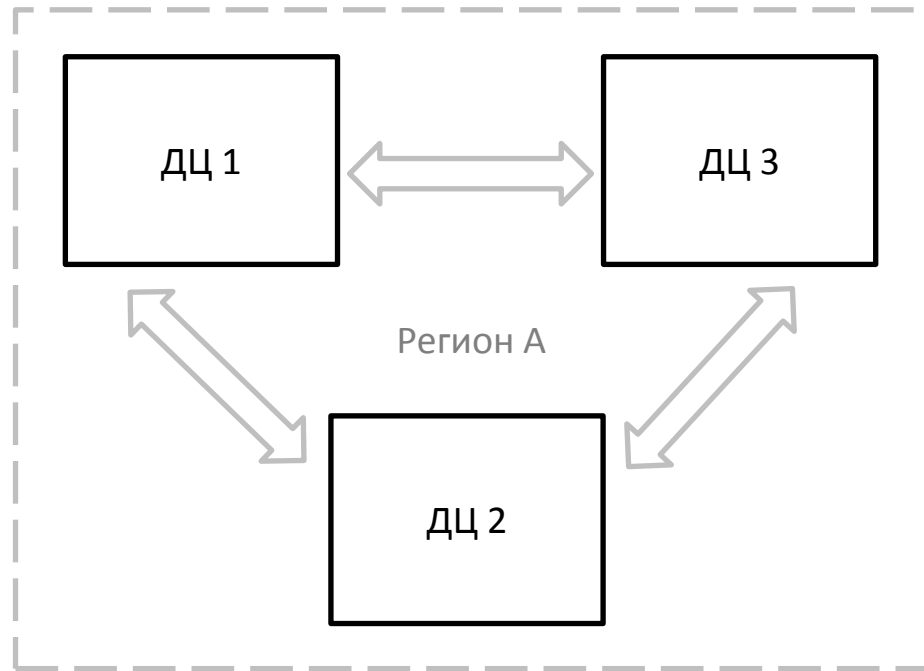
Amazon S3 – статистика по Битрикс24

Битрикс24 s3 бакет: запросы



А вдруг... катастрофа?

1. Amazon S3
2. Распределенная ФС с дублированием
3. Удар молнии в ДЦ Amazon и последствия
4. Открытые аналоги: Apache Hadoop



Если завтра – война...



Каждая цифра — это компания, цвет и



Техника копирования, версия 0.

Hadoop

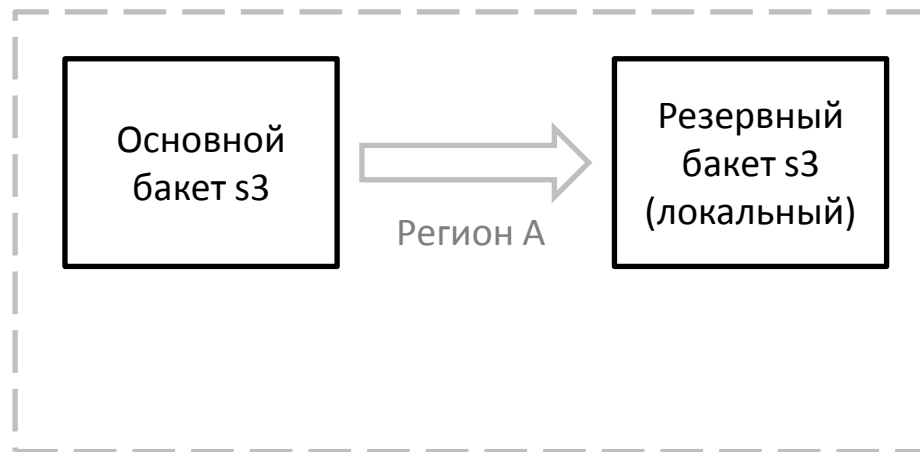
1. Выгружаем список названий файлов бакета s3 в текстовые файлы
2. Около миллиарда файлов
3. Запускаем Hadoop MapReduce jobs для параллельного копирования файлов из бакета А в бакет Б
4. Чтобы повторить: Amazon EMR или Cloudera, Hortonworks и MapR



Репликация за несколько дней, PHP.

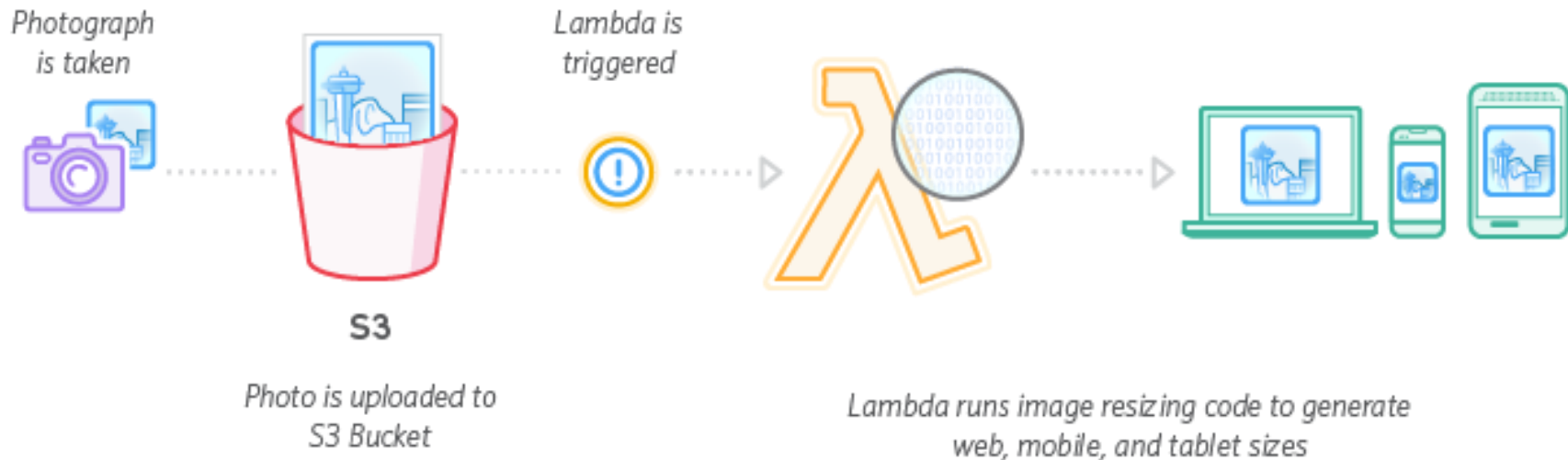
Грабли: ошибка в коде приложения

1. Ошибки в приложении
(ошибка с «очисткой по префиксу»)
2. Копирование в локальный бакет s3 - спасло



Amazon Lambda

Example: *Image Thumbnail Creation*



Basic settings

Description

Make bkp of created files in different s3 bucket in the same region

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.



192 MB

Timeout [Info](#)

2 min 0 sec

Доступные языки для Lambda: **Node.js, Java, Python, Go, C#**

Amazon Lambda – мониторинг

bx24_s3_bkp

Throttle

Qualifiers ▾

Actions ▾

Select a test event.. ▾

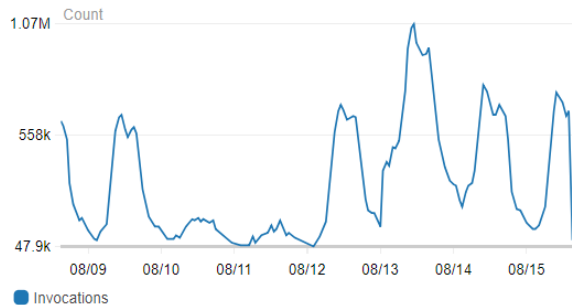
Test

Save

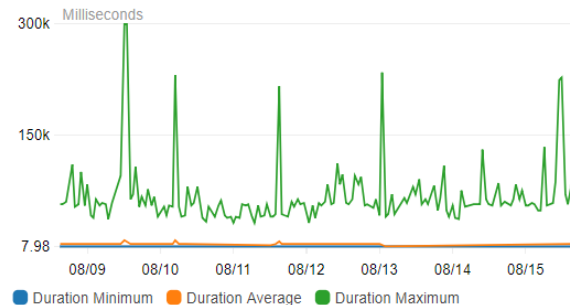
1h 3h 12h 1d 3d 1w custom ▾



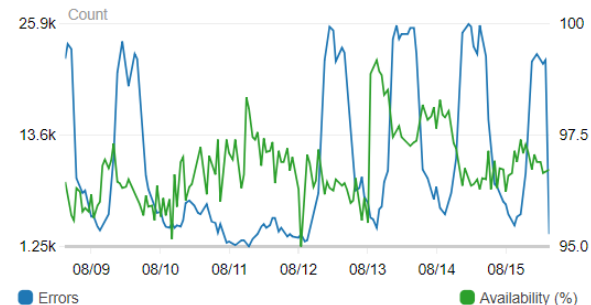
Invocations



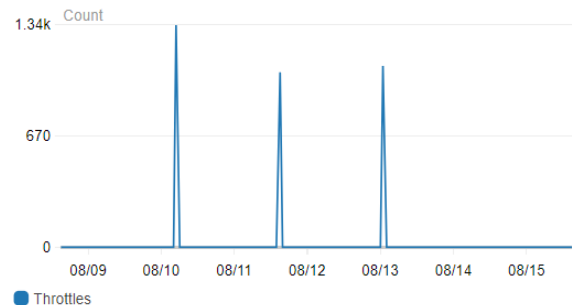
Duration



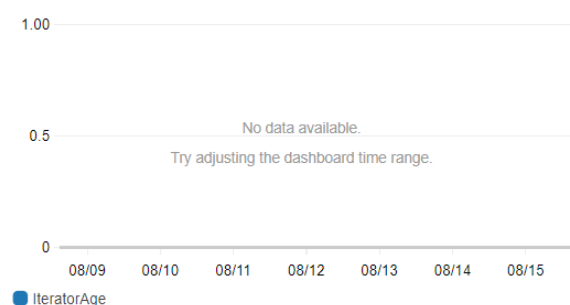
Errors, Availability (%)



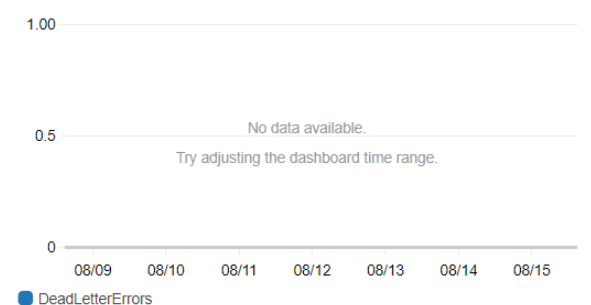
Throttles



IteratorAge

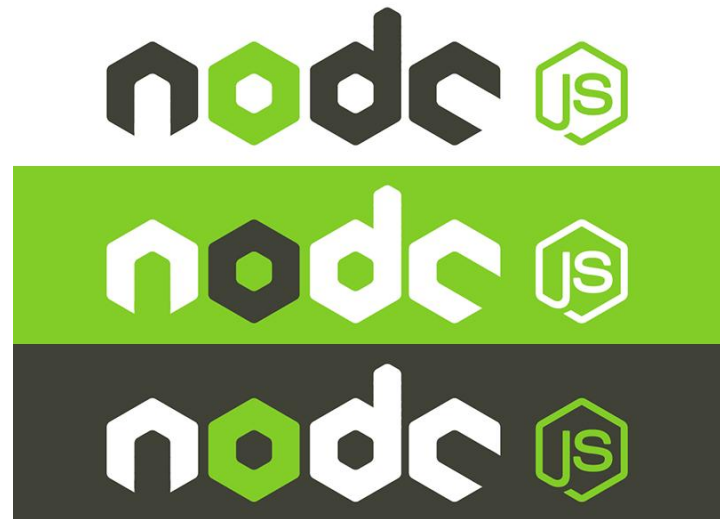


DeadLetterErrors



Amazon Lambda – версия 1, Node.JS

1. Кратко про технологию, асинхронность
2. Пакеты (npm)
3. Просто
4. Быстро
5. Глючит окружение в Амазоне
6. Спонтанное падение воркеров
7. Теряются события



Node.JS, npm и... потроха



Amazon Lambda – версия 2, Java

1. Строгая типизация, отточенные промышленные стандарты
 2. Великолепные среды разработки
 3. Скорость
 4. Много кода
 5. Продолжают теряться события, но гораздо меньше
-



Amazon Lambda – версия 2, Java

aws
Services ▾
Resource Groups ▾
serbul @ Su

bx24_s3_bkp

Throttle
Qualifiers ▾
Actions ▾
Select a test event.. ▾
Test
Save

Function code Info

i The deployment package of your Lambda function "bx24_s3_bkp" is too large to enable inline code editing. However, you can still invoke your function right now.

Code entry type

Upload a .ZIP or JAR file
▾

Runtime

Java 8
▾

Handler Info

com.bitrix24.backup.s3.lambda.aws.Lambda

Function package*

📁 Upload

For files larger than 10 MB, consider uploading via S3.

Environment variables

You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more.](#)

BIG_FILES_QUEUE_REGION	eu-central-1	Remove
BIG_FILES_QUEUE_URL	https://sqs.eu-central-1.amazonaws.com/	Remove
DD_MAX_RETRIES	15	Remove
DELETED_FILES_DD_TABLE	bx24_s3_backup_deleted-files	Remove
DELETED_FILES_SHARDS_CNT	100000	Remove



Многопоточные приложения, ... их за ногу

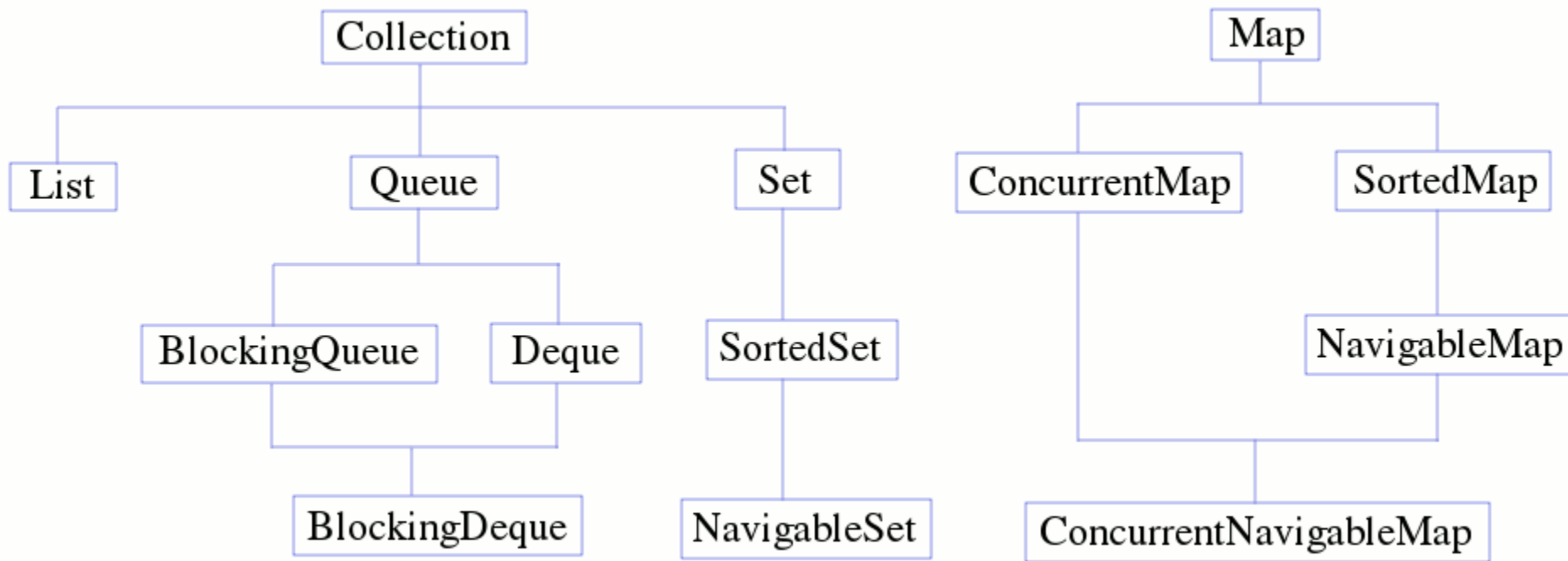
1. Haskell?
2. Erlang?
3. Go?
4. Scala?
5. Kotlin?
6. Java?



Haskell

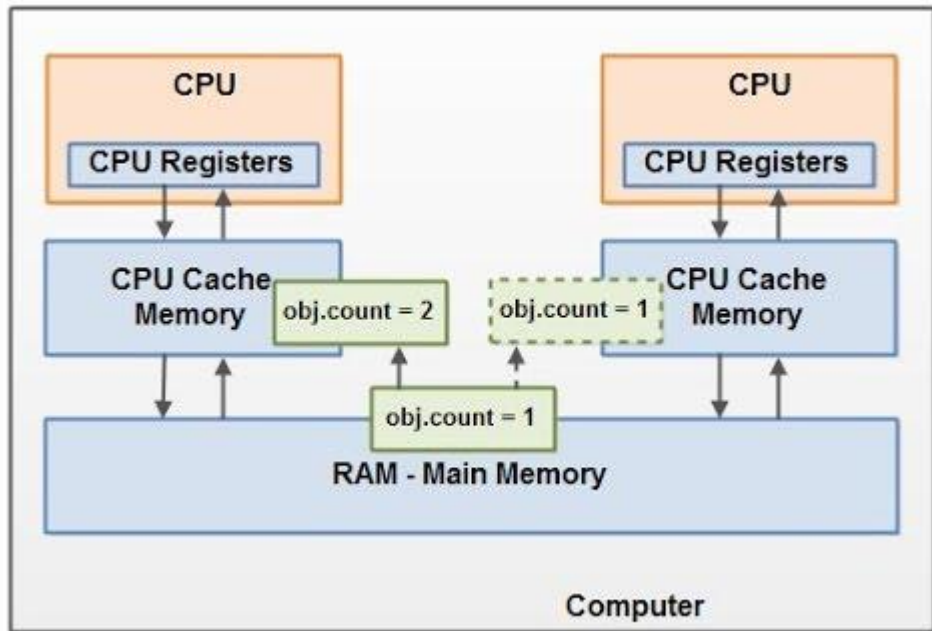
Freedom
from
state

Почти все уже готово, зачем рисковать?

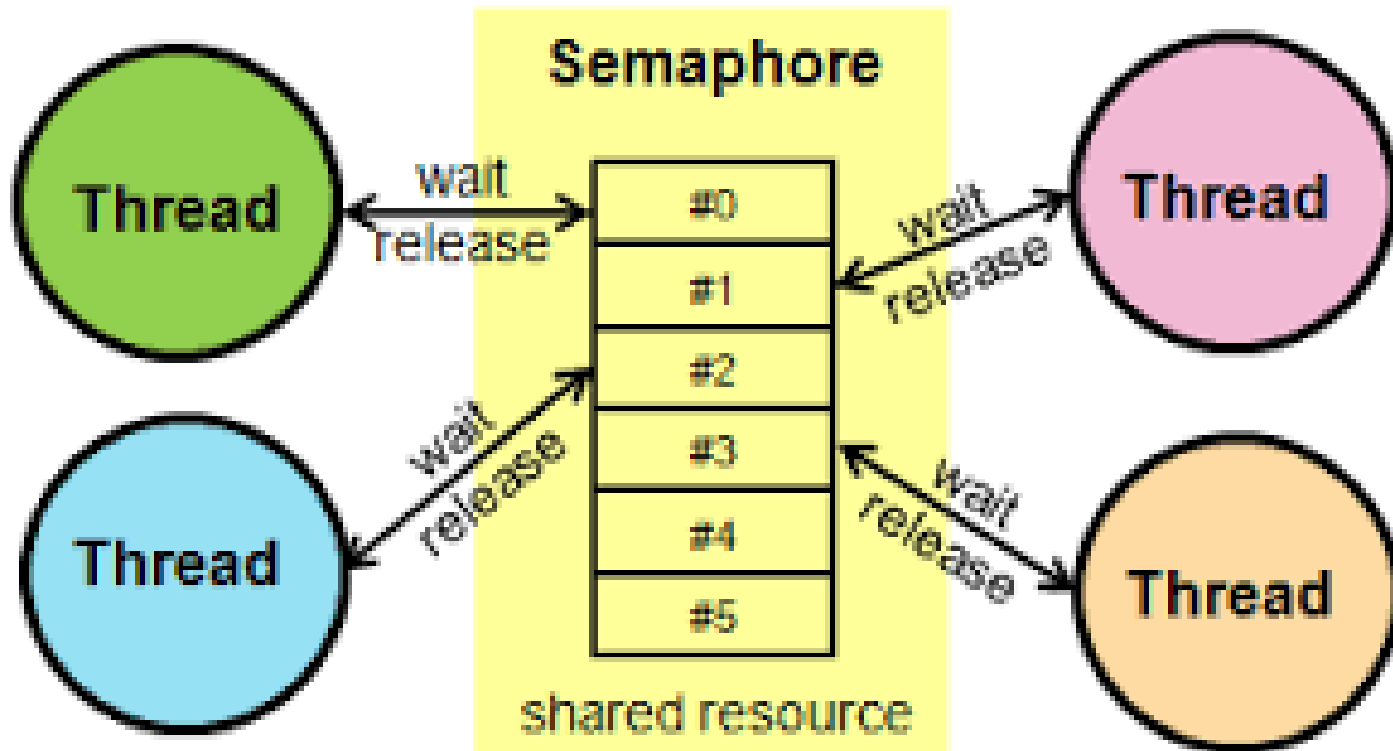


Примитивы синхронизации – полно

1. На уровне языка
2. Синхронизаторы
3. Семафоры
4. Барьеры
5. Модель памяти



Семафоры и другие полезные примитивы



AWS SDK for Java v.1 и 2

1. Хорошая документация
2. Много полезных билдеров и фабрик
3. Потокбезопасность
4. Прозрачная отладка
5. В версии 2 – неблокирующий асинхронный ввод-вывод на Netty (NodeJS – «по-взрослому»)



AWS Java SDK :: Services :: Amazon S3

The AWS Java SDK for Amazon S3 module holds the client classes that are used for communicating with Amazon Simple Storage Service

License	Apache 2.0
Tags	aws s3 amazon
Used By	27 artifacts

Central (11)

Version	Repository	Usages	Date
2.0.0-preview-11	Central	13	Jul, 2018
2.0.0-preview-10	Central	19	May, 2018
2.0.0-preview-9	Central	16	Mar, 2018
2.0.0-preview-8	Central	18	Feb, 2018
2.0.0-preview-7	Central	16	Dec, 2017



AWS Java SDK For Amazon S3

The AWS Java SDK for Amazon S3 module holds the client classes that are used for communicating with Amazon Simple Storage Service

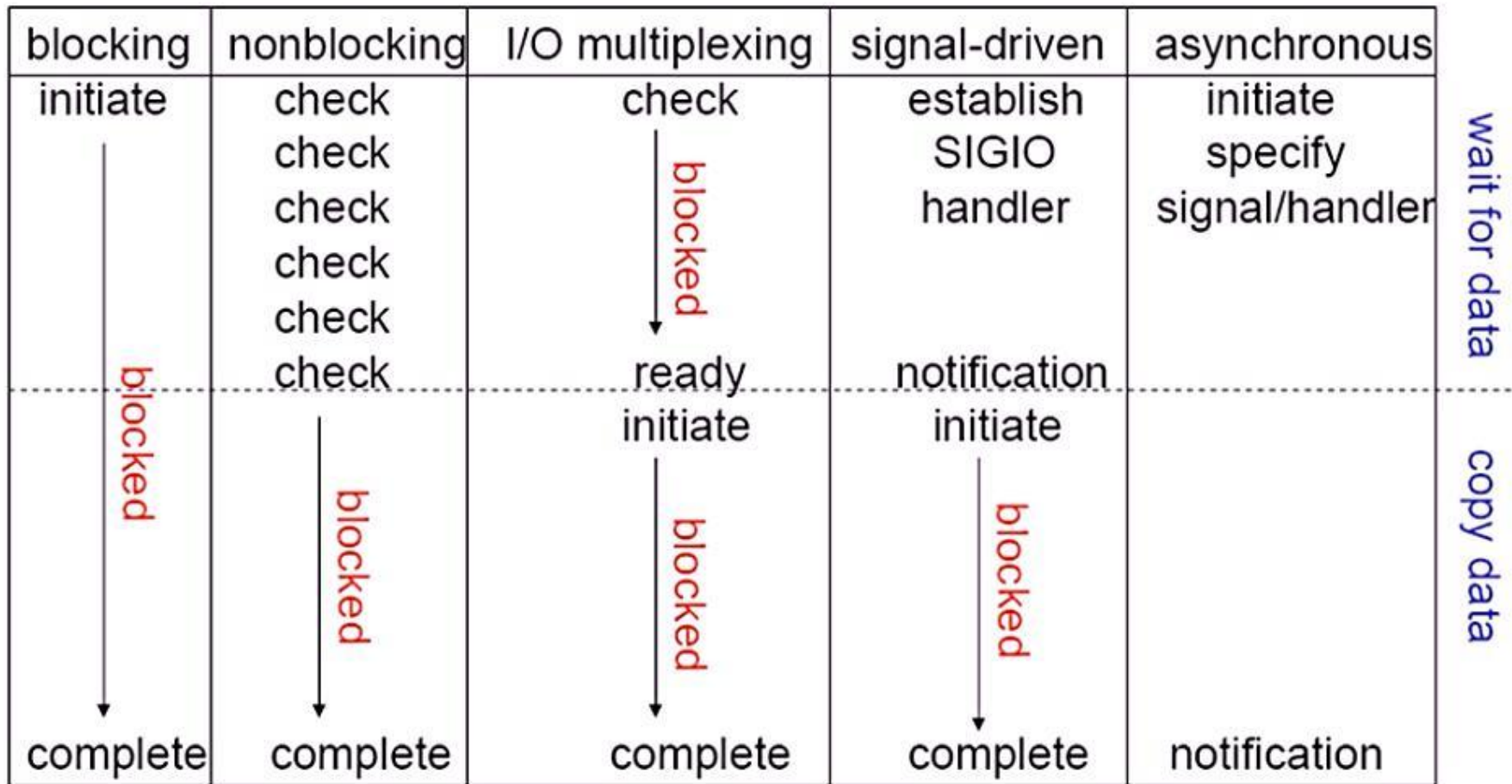
License	Apache 2.0
Categories	S3 Clients
Tags	amazon aws s3 sdk
Used By	456 artifacts

Central (514) Atlassian 3rd-P Old (5)

Version	Repository	Usages	Date
1.11.387	Central	3	Aug, 2018
1.11.386	Central	3	Aug, 2018
1.11.385	Central	3	Aug, 2018
1.11.384	Central	3	Aug, 2018
1.11.383	Central	4	

Сравнение возможностей AWS SDK for Java v.1 и v.2

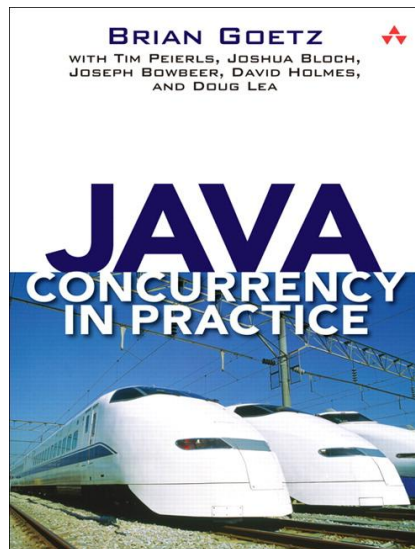
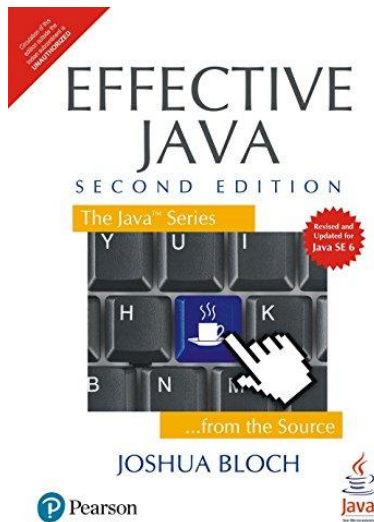
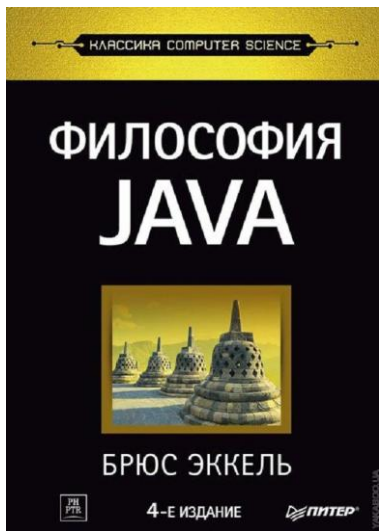
Возможность	AWS SDK for Java v.1	AWS SDK for Java v.2
Согласованное именование классов и методов	-	+
Билдеры вместо конструкторов – почти повсеместно	-	+
Асинхронная работа с сервисами (через пул потоков)	+	+
Асинхронный неблокирующий ввод-вывод через мультиплексирование	-	+ (в т.ч. Netty)
Иммутабельные объекты (POJO)	-	+



← synchronous I/O

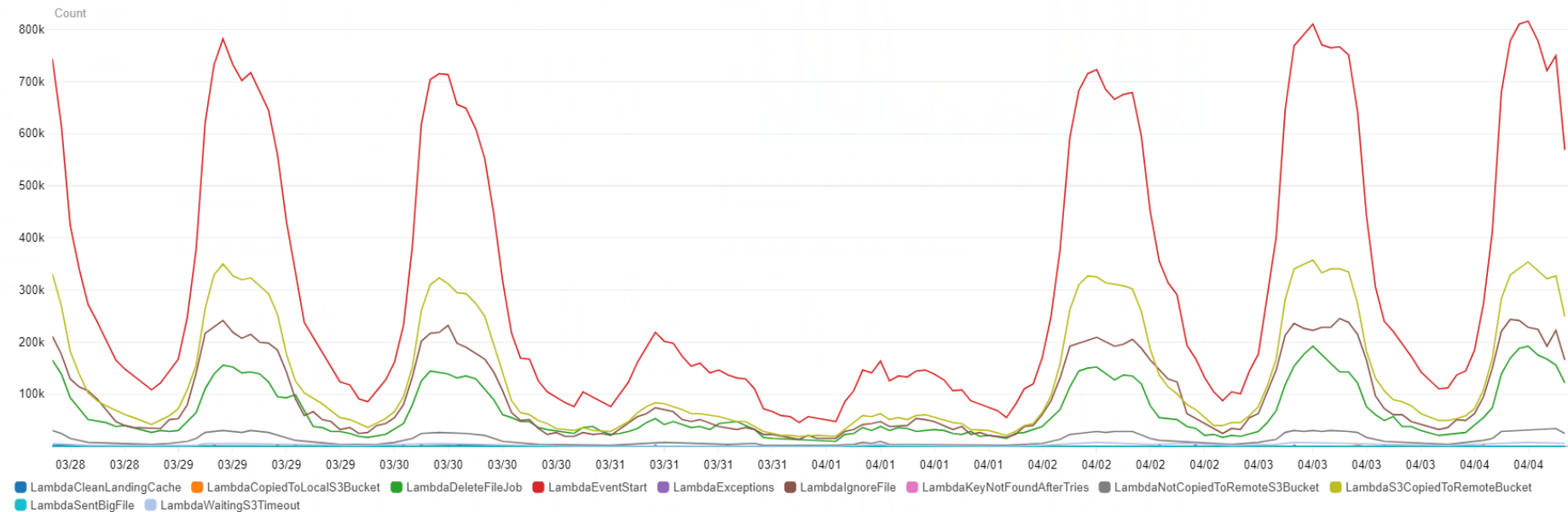
asynchronous I/O →

Книжки по программированию, которые стоит прочитать хоть раз



Amazon Lambda – аналитика

Битрикс24 s3 lambda: запросы



Amazon Lambda – подводные камни

1. Нет SLA
 2. Мониторинг
 3. Спонтанные падения воркеров (Node.JS)
 4. Версии и обновление сервисов, непрозрачность документации
 5. Потеря пакетов, сверка и синхронизация остатков
 6. Есть ли альтернатива?
-

Amazon Lambda – dead letter queue (DLQ)

1. Если сообщение не доставлено в Lambda 2 раза
 2. Попробуй его записать в DLQ SQS
 3. Если не получилось (?) – увеличь счетчик ошибок
DLQ
 4. Недоставленные сообщения нужно разбирать
отдельно
 5. Кейс – ошибка при обновлении Lambda
-

Amazon Lambda – сервис поддержки

1. Что делать с «большими» файлами, которые сильно удорожают использование Lambda?
 2. Сервис «поддержки» Amazon Lambda:
 - копирование больших файлов
 - до-отправка DLQ-записей
 - отложенное удаление
 - ревалидация по внутренним файловым событиям ядра Битрикс24
-

Сервис поддержки – «железо»

c3.xlarge: 4 ядра, 7.5 ГБ ОЗУ

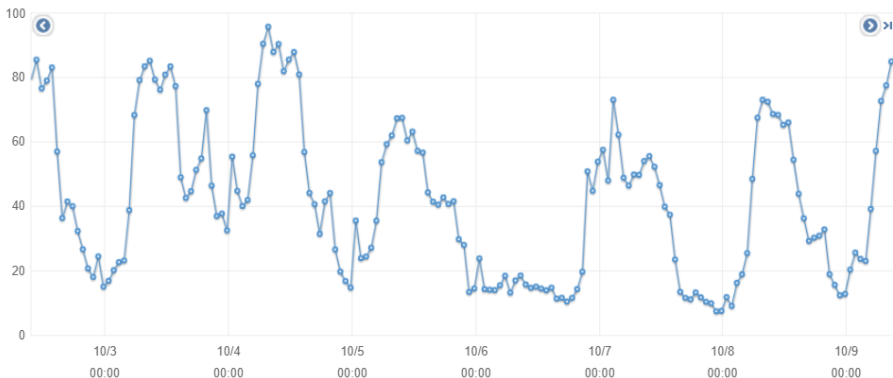
CloudWatch Monitoring Details ✕

CPU Utilization (Percent)

Statistic: Average

Time Range: Last 1 Week

Period: 1 Hour



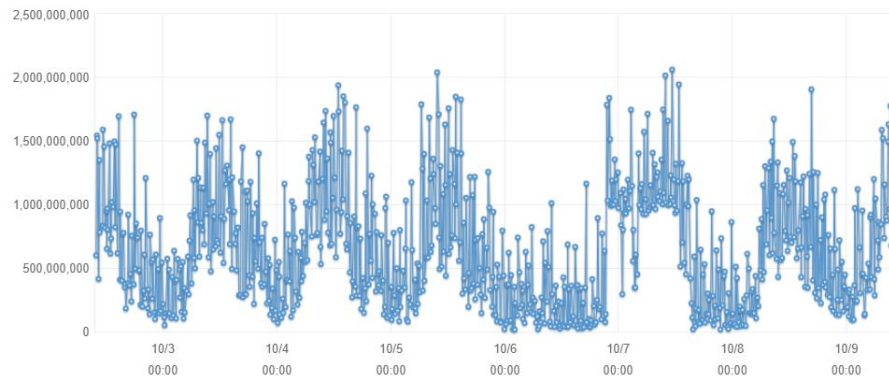
CloudWatch Monitoring Details ✕

Network In (Bytes)

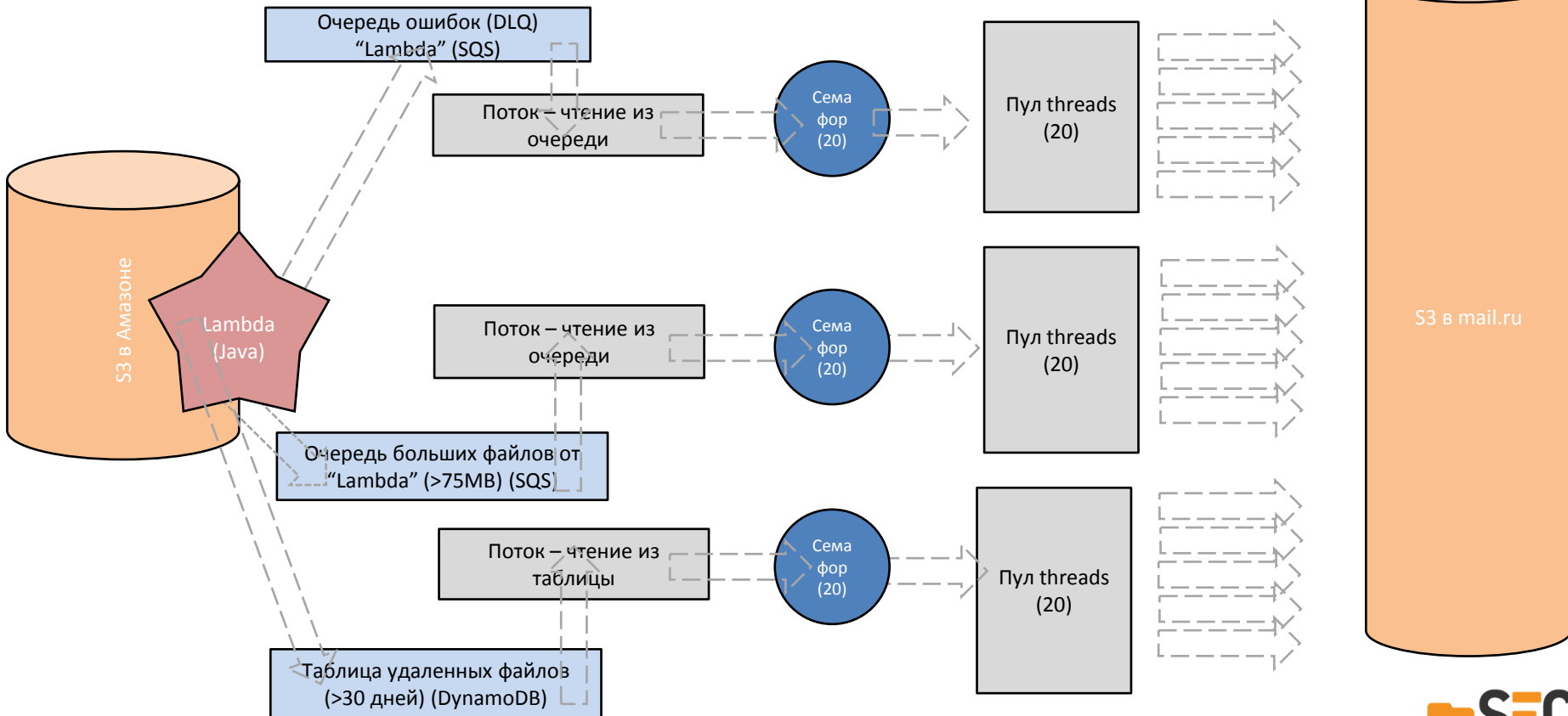
Statistic: Average

Time Range: Last 1 Week

Period: 5 Minutes



Архитектура репликатора файлов клиентов Битрикс24 между континентами



Сервис поддержки – структура кода

```

bigdata-bx24-s3-bigfile-remote-backuper
├── src/main/java
│   ├── com.bitrix24.backup.s3
│   │   ├── AwsUtils.java
│   │   ├── BigFileDownloadUploadTask.java
│   │   ├── BigFileQueueReader.java
│   │   ├── BigFilesRemoteBackuper.java
│   │   ├── Bx24PHPEngineEventFileInfo.java
│   │   ├── Bx24PHPEngineEventProcessingTask.java
│   │   ├── Bx24PHPEngineEventQueueReader.java
│   │   ├── DerbyRegistry.java
│   │   ├── DftEventProcessingTask.java
│   │   ├── DftTableReader.java
│   │   ├── DfqEventProcessingTask.java
│   │   ├── DfqFileInfo.java
│   │   ├── DfqQueueReader.java
│   │   ├── FileInfo.java
│   │   ├── HashMapRegistry.java
│   │   └── Lambda.java
│   ├── LocalS3FilesRegistry.java
│   ├── S3BucketDeleter.java
│   ├── S3BucketDumper.java
│   ├── S3BucketStat.java
│   ├── S3Differ.java
│   ├── S3KeyData.java
│   ├── S3KeyDataProcessor.java
│   ├── S3KeyInventoryJsonDataSource.java
│   ├── SftEventProcessingTask.java
│   └── SftTableReader.java
├── src/test/java
├── JRE System Library [JavaSE-1.8]
├── Maven Dependencies
├── src
├── target
└── pom.xml
    
```

1. Apache Maven
2. Интенсивное использование “concurrency framework” и “collections framework”
3. Параноидальный код, продуманная работа с exceptions
4. Разные уровни логирования с slf4j + log4j
5. Юнит-тесты и частично интеграционные с Mockito
6. В ближайший план – кластер с Apache ZooKeeper

Сервис поддержки – техники программирования

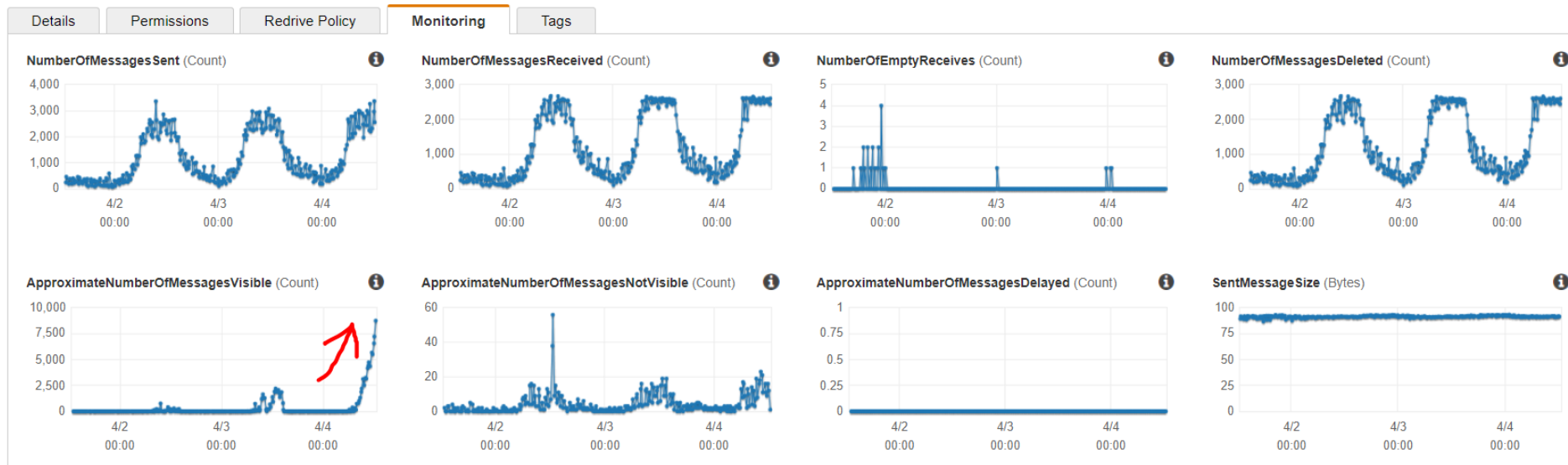
1. Готовые библиотеки, где можно: maven shade, commons-cli, commons-validator, commons-io, org.json, junit, org.mockito, com.opencsv
2. Иммутабельные объекты, где возможно
3. Билдеры и фабрики, где удобно, вместо конструкторов
Инкапсуляция/делегирование вместо наследования
4. Константность, где только можно – через “final”
5. Инкапсуляция, где имеет смысл и удобно
6. Проверка входных параметров в публичных и... остальных методах
7. Интенсивное использование Enums – для перечислений и сопоставлений с образцом
8. Unit-тесты – где можно и разумно

Сервис поддержки – копирование «больших» файлов

1. Читает SQS-очередь «больших» файлов, получает задание
2. Захватывает семафор над пулом копирующих потоков (ExecutorService)
3. Пытается скопировать файл в облако mail.ru напрямую через сокет-сокет, без заливки на диск
4. Удаляет SQS-задание из очереди
5. Освобождает семафор

Сервис поддержки – копирование «больших» файлов

1 SQS Queue selected



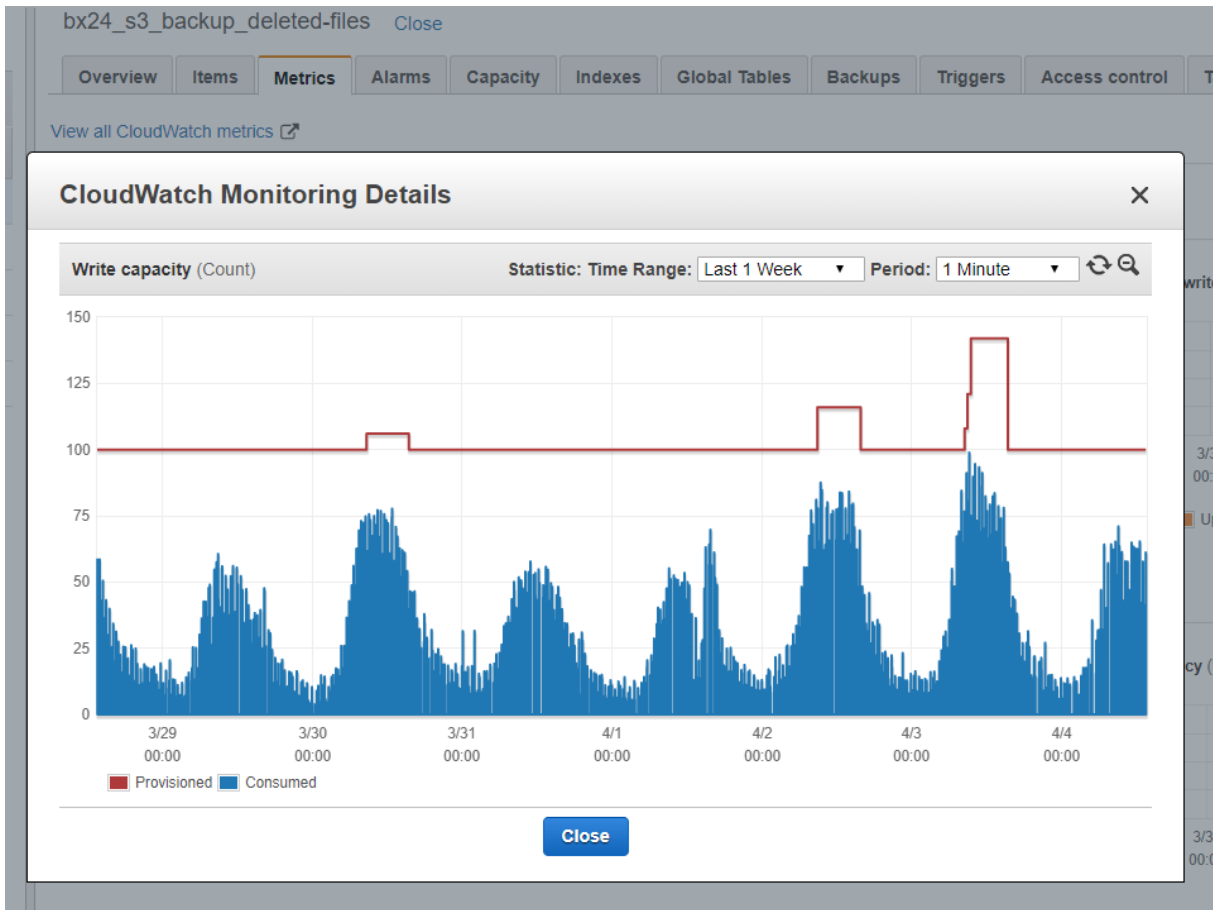
Сервис поддержки – дозаливка событий из DLQ

1. Читает SQS-очередь «dead letter queue» с именами недоставленных Amazon Lambda-файлов, получает задание
2. Захватывает семафор над пулом копирующих потоков (ExecutorService)
3. Пытается скопировать файл в облако mail.ru напрямую через сокет-сокет, без заливки на диск
4. Удаляет SQS-задание из очереди
5. Освобождает семафор

Сервис поддержки – отложенное удаление файлов

1. Читает Amazon DynamoDB-таблицу (выборка событий месячной давности), получает задание
2. Захватывает семафор над пулом удаляющих потоков (ExecutorService)
3. Пытается удалить файл из бакета mail.ru
4. Удаляет задание из таблицы
5. Освобождает семафор

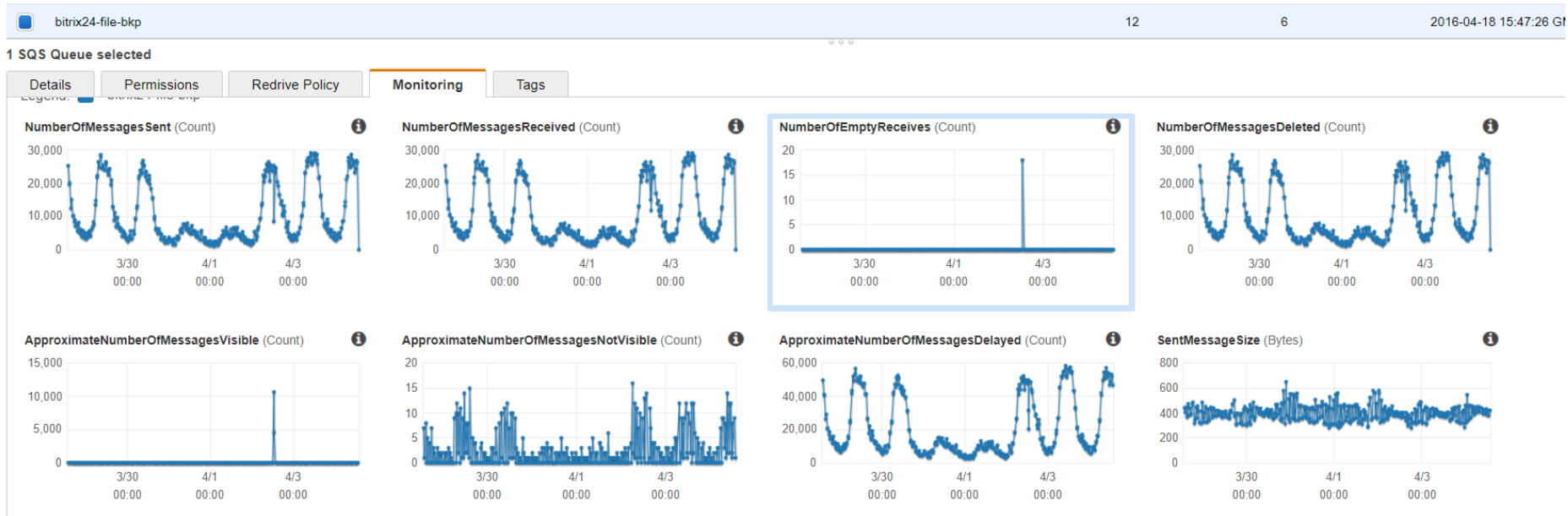
Сервис поддержки – отложенное удаление файлов



Сервис поддержки – корректировка файловыми событиями ядра Битрикс24

1. Читает Amazon SQS-очередь файловых событий ядра Битрикс24, получает задание
2. Захватывает семафор над пулом копирующих потоков (ExecutorService)
3. Пытается скопировать файл в бакет mail.ru
4. Удаляет задание из SQS-очереди
5. Освобождает семафор

Сервис поддержки – корректировка файловыми событиями ядра Битрикс24

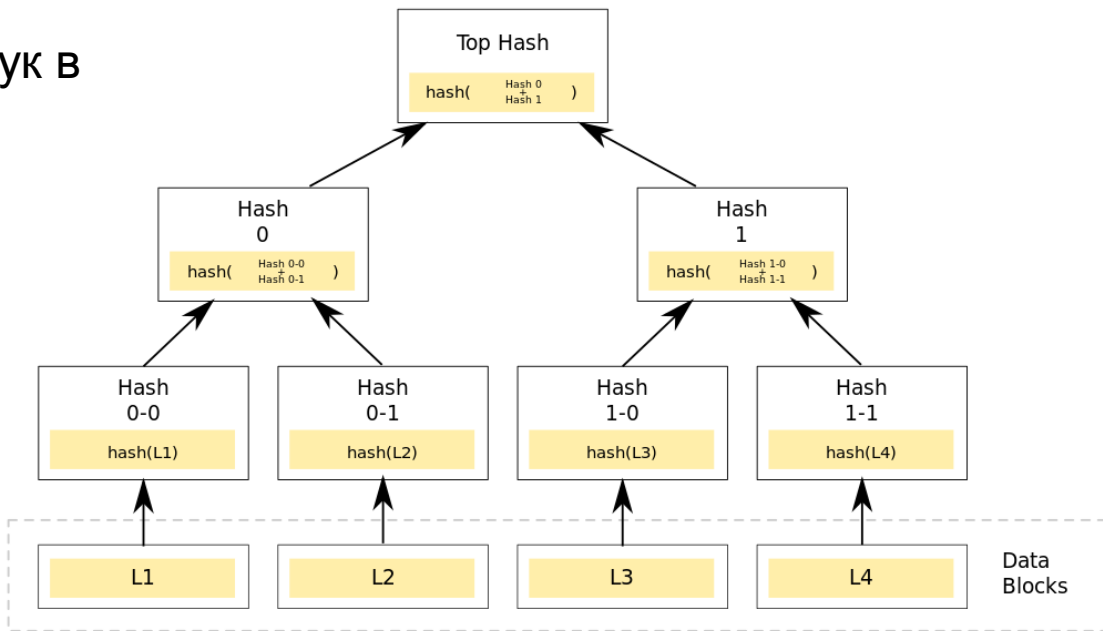


Проект по миграции 1 млрд. файлов, >500 ТБ

1. Сканируются все записи в s3 Битрикс24 в один поток
2. Имена файлов записываются в DynamoDB-таблицу на разные шарды для повышения скорости записи
3. Модуль в один поток читает записи с рандомной шарды и увеличивает номер шарды (при достижении предельной шарды начинает с рандомной шарды)
4. Часть чтений начинается с 0 шарды (чтобы захватить начальный «хвост»)
5. Модуль захватывает семафор и ставит задание в многопоточный ExecutorService
6. Задание на копирование выполняется в отдельном потоке. Успешное выполнение – удаление записи в DynamoDB-таблице
7. Модуль освобождает семафор

S3->mail.ru – ресинхронизация ВОЗМОЖНЫХ ПОТЕРЬ

1. Список файлов в 1 млрд штук в двух s3-хранилищах
2. BerkeleyDB
3. SQLite
4. LevelDB
5. LMDB
6. Amazon Inventory
7. Merkle tree?
8. Варианты развития архитектур



S3->mail.ru: ресинхронизация


1. Читаем список файлов каждого портала в бакете s3 и сохраняем в конкурентной хэш-таблице в памяти (ConcurrentHashMap)
2. Читаем список файлов этого же портала в бакете mail.ru и пересекаем с хэш-таблицей в памяти
3. Пункты 2-3 – делаются многопоточно
4. Оставшиеся файлы в памяти – копируем в одном направлении:
s3 -> mail.ru
5. Ведем детальное логирование
6. Запускаем раз в неделю

mail.ru->S3: обратная репликация

1. В mail.ru бакете нет аналогов Amazon Lambda.
2. Передаем события ядра Битрикс24 в SQS очередь
3. Поток (в перспективе - несколько) в асинхронно-неблокирующем режиме обрабатывает события с файлами и копирует их.
4. При успешном завершении операции – событие удаляется из SQS-очереди
5. Используем AWS SDK for Java v.2 – поддержка асинхронно-неблокирующих операций. Улучшение производительности – **на порядок.**

Асинхронно-неблокирующая техника: скачиваем файл

```
final CompletableFuture<GetObjectResponse> future = BigFilesRemoteBackuper.getS3DstV2AsyncClient().getObject(  
    GetObjectRequest.builder()  
        .bucket(BigFilesRemoteBackuper.getS3DstBucket())  
        .key(fi.getKey())  
        .build(),  
    AsyncResponseHandler.toFile( tmpUniqFileName )  
);  
  
future.whenComplete((resp, err) -> {
```



1. Не ждем, регистрируем обработчик и идем дальше
2. Поток не простаивает

Асинхронно-неблокирующая техника: заливаем файл

```

future.whenComplete((resp, err) -> {

    if (resp != null) {

        //Download OK handler body

        log.info("{}: Downloaded OK: {}->{}", Thread.currentThread().getName(), fi.getKey(), tmpUniqFileName)

        ///// Upload
        ///

        final Map<String, String> meta = new HashMap<String, String>();
        meta.put(BigFilesRemoteBackuper.REPLICATION_S3_KEY_META_PARAM_NAME, S3BucketReplicationCode.MAIL_RU_1)

        final CompletableFuture<PutObjectResponse> futureUpl = BigFilesRemoteBackuper.getS3SrcV2AsyncClient()
            .putObjectRequest
                .builder()
                .bucket(BigFilesRemoteBackuper.getS3SrcBucket())
                .key(fi.getKey())

                .metadata(meta)
                .build(),
            AsyncRequestProvider.fromFile( tmpUniqFileName )
        );

        futureUpl.whenComplete((resp2, err2) -> {
            try {

```

Асинхронно-неблокирующая техника: завершение транзакции

1. Если перелили файл, удаляем задание из очереди, иначе – повторяем через определенное время
2. «Простота – залог надежности» (*Edsger W. Dijkstra*)

```
futureUpl.whenComplete((resp2, err2) -> {
    try {
        if (resp2 != null) {
            //Upload OK handler body
            log.info("{}: Uploaded: {}->{}", Thread.currentThread().getName(), tmpF
                sqsSafeDeleter.jobWasExecutedSuccessfully(m.getReceiptHandle());
        } else {
            //Upload error handler body
            log.error("{}: Upload error handler body: {}", Thread.currentThread().get
        }
    } catch ( Exception e ) {
        log.error("{}: Upload handler exc.: {}", Thread.currentThread().getName(),
    } finally {
        try {
            //always free lock!
            log.info("{}: Releasing Semaphore: {}",
                Thread.currentThread().getName(), BigFilesRemoteBackuper.getS3L
            BigFilesRemoteBackuper.getS3MailRu2AwsS3LambdaSemaphore().release();
            Files.deleteIfExists(tmpUniqFileName);
        }
    }
});
```

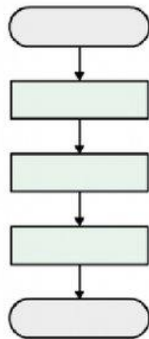
Опасайтесь спагетти-кода с промисами и подтанцовками!

Структурное программирование

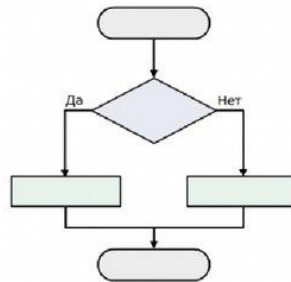
— методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 1970-х годах Э. Дейкстрой и др.

Основные типы алгоритмических структур:

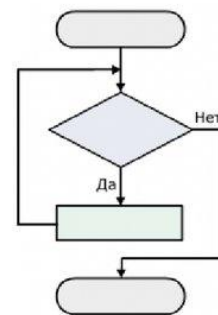
Линейная



Разветвляющаяся



Циклическая



Ближайшие планы

1. Внедрить систему кластерной координации Apache ZooKeeper – для масштабирования решения на множество машин
 2. Реализовать средства межмодульного взаимодействия: конфигурация, выбор лидера, очереди, блокировки, двухфазный коммит
 3. Быть готовыми масштабироваться на 2-100-1000 spot-серверов!
-

Спасибо за
внимание!
Вопросы?

Александр Сербул

 @AlexSerbul

 Alexandr Serbul

serbul@1c-bitrix.ru

