Оркестрация транскодеров с помощью K8s

Ilia Galimianov

Привет!

- Меня зовут Илья Галимьянов
- Технический лидер команды OTT WASD.TV - подразделение eSports в MTS DIGITAL
- В сфере медиа-стриминга около 4х лет
- Занимаюсь разработкой ПО для приема\обработки\доставки аудио-видео данных
- Имел дело с китайскими видео-энкодерами

О чем буду рассказывать

- Что такое оркестрация и зачем она нужна
- Стоимость реализации оркестрации с нуля
- Kubernetes как framework для оркестрации
- Оценка и планирование ресурсов транскодеров в K8s
- Реализация собственного планировщика для K8s
- Пробный запуск и расследование проблем

WASD.TV

- Видео-стриминговая платформа МТС
- От приема сигнала до доставки зрителям
- Контент обычно очень динамичный 1080P@60FPS
- Транскодируем всех по возможности и только на CPU
- Все транскодеры запускаются на bare-metal
- Железо все разношерстное (от Intel Westmere EP до Cascade Lake)
- Мы предъявляем требования к пользовательским аудио-видео потокам (Max FPS, Bitrate, Resolution)

Зачем нужна оркестрация?

История одного сервера

- Запускаем весь софт на одном физическом\виртуальном сервере
- И еще запускаем
- И еще?
- ....
- Ресурсы на сервере кончились

Закупим еще серверов!

Закупаемся

- Сколько покупать?
- Что покупать?
- …
- Все сложно, поэтому пусть отдел закупок чего-нибудь купит да побольше!

Много серверов - много проблем

- Теперь мы можем запускать ПО на всех серверах
- Как выбирать будем где запустить?
- Что делать, если откажет один из серверов?
- Что если приложение упадет?
- А если захотим остановить приложение?
- А вообще достаточно ли северов мы купили?

Как решить проблему кучи серверов?

- Написать ПО, которое:
  - Запускает и следит за работоспособностью моего ПО
  - Знает о максимальной емкости каждого из серверов

А что такое оркестрация?

- Оркестрация - это автоматическое размещение, координация и управление ИТ системами
- Какой минимум функций оно должно выполнять?
  - Запуск\остановка\перезапуск ПО
  - Отслеживание работоспособности
  - Миграция на другой сервер в случае проблем
  - Отдача статуса о состоянии запущенного ПО
  - Распределение ПО в зависимости от загруженности серверов

Думаю над своим ПО для оркестрации

Делаю свой оркестратор

- Кто будет давать команды о запуске\остановке\перезапуске?
- А где хранить, что нужно запустить\остановить и состояние запущенного ПО?
- Что если тот самый мастер отвалится?
- Как узнать, что отвалились сервера исполняющие ПО?
- Балансировать нагрузку как?
- А это как?
- …

Мы находились примерно здесь

DevOps’ы приходят на помощь

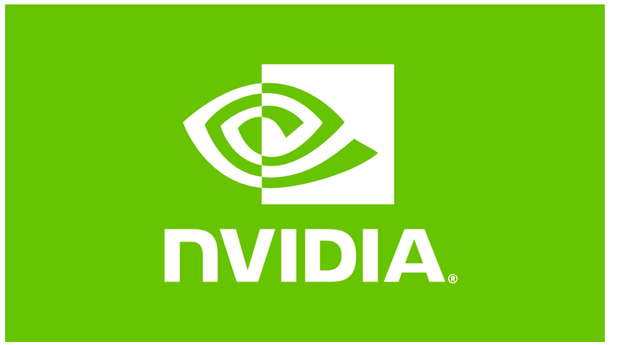
- Я стал рассказывать о своей проблеме нашему DevOps’у, в ответ он продал мне K8s и заверил, что это как робо-рука и она сейчас всем нужна

Kubernetes? Kubernetes!

Почему Kubernetes?

Плюсы

- Он уже написан!
- Умеет все то, что хотели (запуск, остановка, балансировка, etc.)
- Умеет не только то, что хотели
- Около 2766 компаний эксплуатирует K8S (по данным stackshare.io)
- Масштабируемый и отказоустойчивый
- DevOps’ы умеют с ним работать
- Расширяемый



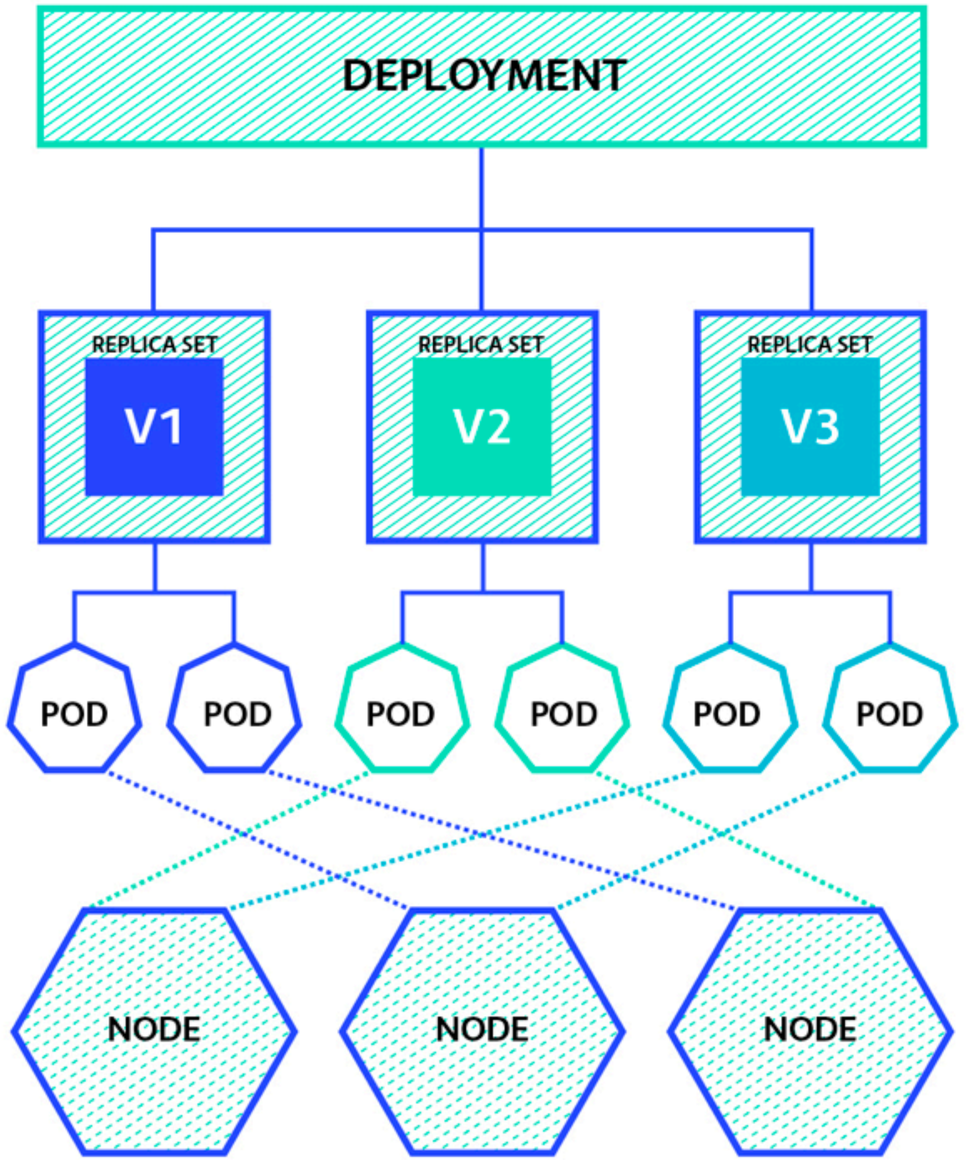
Почему Kubernetes?

Минусы

- Чуть-чуть сложный
- Кто-то не любит контейнеры
- Фич так много, что разбегаются глаза

Основные сущности

- Node - сервер
- Deployment - что хотим запустить
- Pod - экземпляр запущенного контейнера
- И еще куча всяких, но нам они не интересны



Deployment

- В Spec указывается требуемое состояние
- Основное
  - Кол-во реплик
  - Образ с ПО
  - Affinity
  - Лимиты CPU\RAM

А как запустить транскодер?

Рецепт запуска транскодера

- Подключенные Node к кластеру
- Готовый образ с транскодером
- Написанный deployment с указанным:
  - Образом транскодера
  - Лимитами CPU/RAM
    - А где их взять?

```
kind: Deployment
metadata:
  name: transcoder
spec:
  replicas: 1
    spec:
      containers:
        image: registry.io/ffmpeg:4.4.1
        name: transcoder
        resources:
          limits:
            cpu: ???
            memory: ???
          requests:
            cpu: ???
            memory: ???
```

Оцениваем требуемые ресурсы для транскодера

А зачем их оценивать?

- Прогнозируемая емкость сервера на количество одновременно работающих транскодеров живого траффика
- Избежание проблем перегрузок
  - Транскодер не сможет потреблять ресурсов больше, чем ему выдали
  - Транскодер не запустится там, где свободных ресурсов для него не хватает

В чем заключается оценка

- Определить пиковое потребление ресурсов транскодером на живом траффике в рамках определенного FPS\Resolution\Bitrate на определенной серверной конфигурации

Как буду оценивать?

- Возьму FFmpeg
- Сгенерирую видео с рандомным шумом 1080P@60FPS 10MBit/s
- Оттранскодирую видео в ABR Ladder с фильтром -re
- Удостоверюсь, что скорость всегда была >= 1x
- Соберу метрики CPU\RAM и повторю тест три раза
- Возьму пиковые значение CPU\RAM
- Повторю для других наборов видео (1080P@30, 720P@60, etc.) и на разных на серверных конфигурациях

Требования к видео для Benchmark’a

- Картинка должна быть максимально динамичной для своего битрейта
- Битрейт должен быть CBR (или быть похожим на него)
- FPS должен быть CFR, никакого VFR’a
- Главная цель - заставить транскодер потреблять максимальное кол-во ресурсов

Генерация видео для benchmark’a

- Читаем из /dev/urandom и превращаем в сырое видео и аудио
- Размер сырого видео 384x216, т.к. если оно будет больше, то кодировщик просто вырежет всю нашу динамику
- Транскодирую на CPU и выполняю upscale сырого видео в такое, которое укладывается в рекомендуемые хар-ки нашей платформы

```
ffmpeg \
    -f rawvideo -video_size 384x216 -pixel_format yuv420p -r 60 \
    -i /dev/urandom -ar 48000 -ac 2 -f s16le -i /dev/urandom \
    -c:a aac \
    -c:v copy \
    -t 60 -f matroska noise_raw.mkv

ffmpeg -i noise_raw.mkv \
    -c:v libx264 -vf scale=1920:1080 \
    -profile:v high -preset medium -r 60 \
    -force_key_frames "expr:gte(t,n_forced*1)" \
    -b:v 10000k \
    -minrate:v 9000k -maxrate:v 10000k -bufsize:v 16500k \
    -t 60 -f mpegts noise_cpu_upscale_medium.ts
```

Пример актуален только для libx264 энкодера

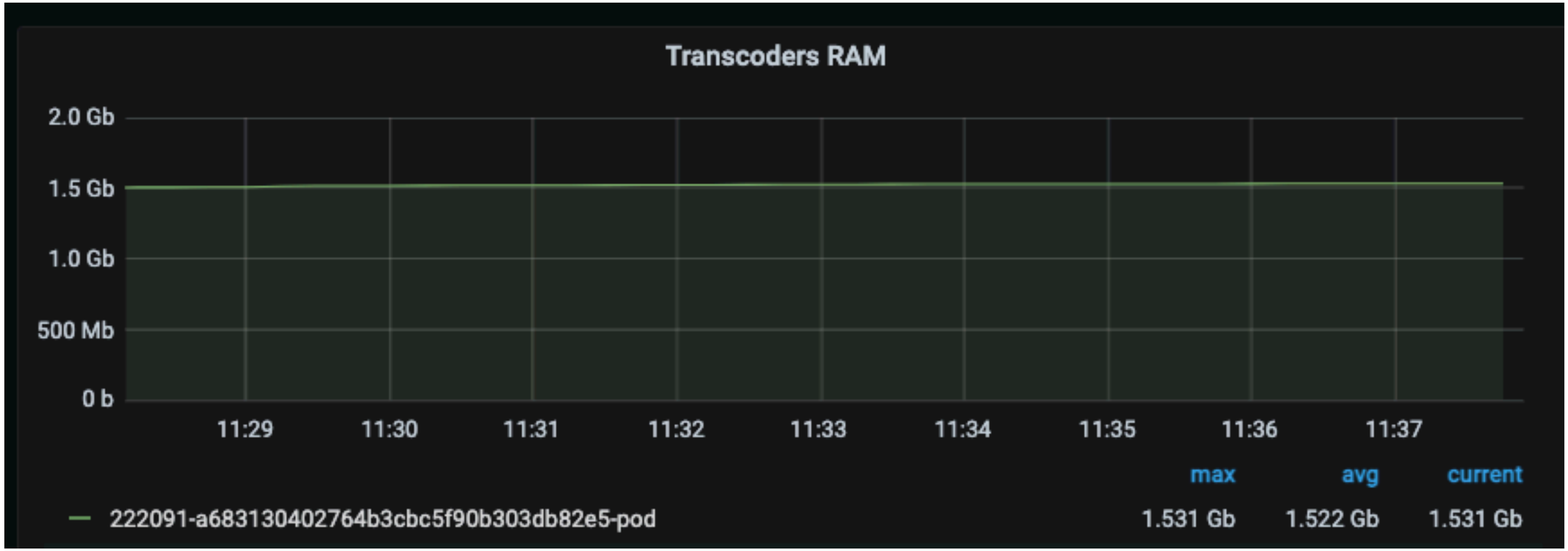
Получил кошмар эпилептика

Транскодирую

```
ffmpeg -re -i noise_cpu_upscale_medium.ts \

    -filter_complex "[0:v]split=4[out_0][m1][m2][m3];[m1]scale=1280:720[out_1];[m2]scale=854:480[out_2];[m3]scale=640:360[out_3]" \

    -map [out_0] -c:v:0 libx264 -profile:v high -preset superfast -force_key_frames "expr:gte(t,n_forced*2)" \
    -b:v:0 10000k -minrate:v:0 9000k -maxrate:v:0 11000k -bufsize:v:0 16500k \

    -map [out_1] -c:v:1 libx264 -profile:v high -preset superfast -force_key_frames "expr:gte(t,n_forced*2)" \
    -b:v:1 3500k -minrate:v:1 3150k -maxrate:v:1 3850k -bufsize:v:1 5775k \

    -map [out_2] -c:v:2 libx264 -profile:v high -preset superfast -force_key_frames "expr:gte(t,n_forced*2)" \
    -b:v:2 1500k -minrate:v:2 1350k -maxrate:v:2 1650k -bufsize:v:2 2475k \

    -map [out_3] -c:v:3 libx264 -profile:v high -preset superfast -force_key_frames "expr:gte(t,n_forced*2)" \
    -b:v:3 500k -minrate:v:3 450k -maxrate:v:3 550k -bufsize:v:3 825k \

    -af aresample=async=1 -map 0:a -c:a aac \
    -f null /dev/null -y
```

Собираю метрики потребления CPU

Собираю метрики потребления RAM



Итоги оценки

Configuration	VideoWidth	VideoHeight	FPS	BitrateKbs	CPU	Memory
Xeon-6248R	1920	1080	60	10 000	5,7	1.5GB
Xeon-6248R	1920	1080	30	5 000	3,5	1GB
Xeon-6248R	1280	720	60	4 000	3	500MB
Xeon-5118	1920	1080	60	10 000	7,5	1.5GB
Xeon-5118	1920	1080	30	10 000	4,5	1GB

* Значения только для примера, это не настоящие показатели

Планирование ресурсов

Планирование ресурсов

- Знаю сколько нужно ресурсов для всего набора принимаемых потоков
- Могу поделить емкость кластера на требуемые ресурсы потоков и получить емкость кластера в кол-ве транскодеров

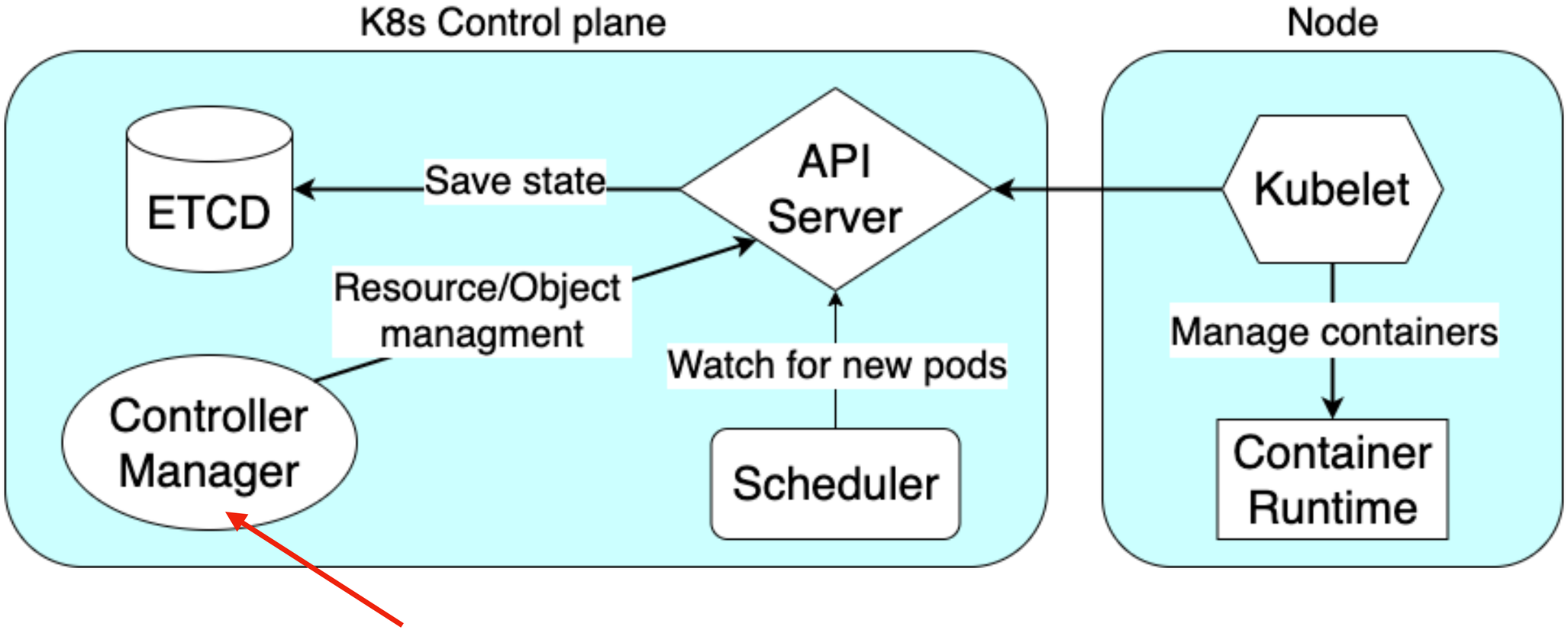
Планирую ресурсы в K8S

- Если придет 1080P@60FPS 10MBit/s, то знаю сколько ему ресурсов нужно
- Могу указать лимиты в Deployment’е
- Лимит можно указать один, а серверов много…
- …
- K8s не подходит?

```
kind: Deployment
metadata:
  name: transcoder
spec:
  replicas: 1
    spec:
      containers:
        image: registry.io/ffmpeg:4.4.1
        name: transcoder
        resources:
          limits:
            cpu: ???
            memory: ???
          requests:
            cpu: ???
            memory: ???
```

Делаю пользовательский контроллер!

K8s изнутри



Этот контроллер
можно написать самим

Деятельность контроллера

- Наблюдает за ресурсами K8s, получая события изменения (CUDP) с помощью kubeapi
- При любых изменениях наблюдаемого ресурса, пытается довести текущее состояние кластера до желаемого (т.е. того, что указано в Spec’е)
- Получает только событие об изменении, а не само изменение

Custom Resource Definition

- Такой же API объект как и Deployment\Node и тд
- Custom - потому что мы сами можем описать, что он будет хранить
- Состоит из Spec и Status
- Spec - требуемое состояние кластера
- Status - текущее состояние объекта

Что хотим в своем controller’е

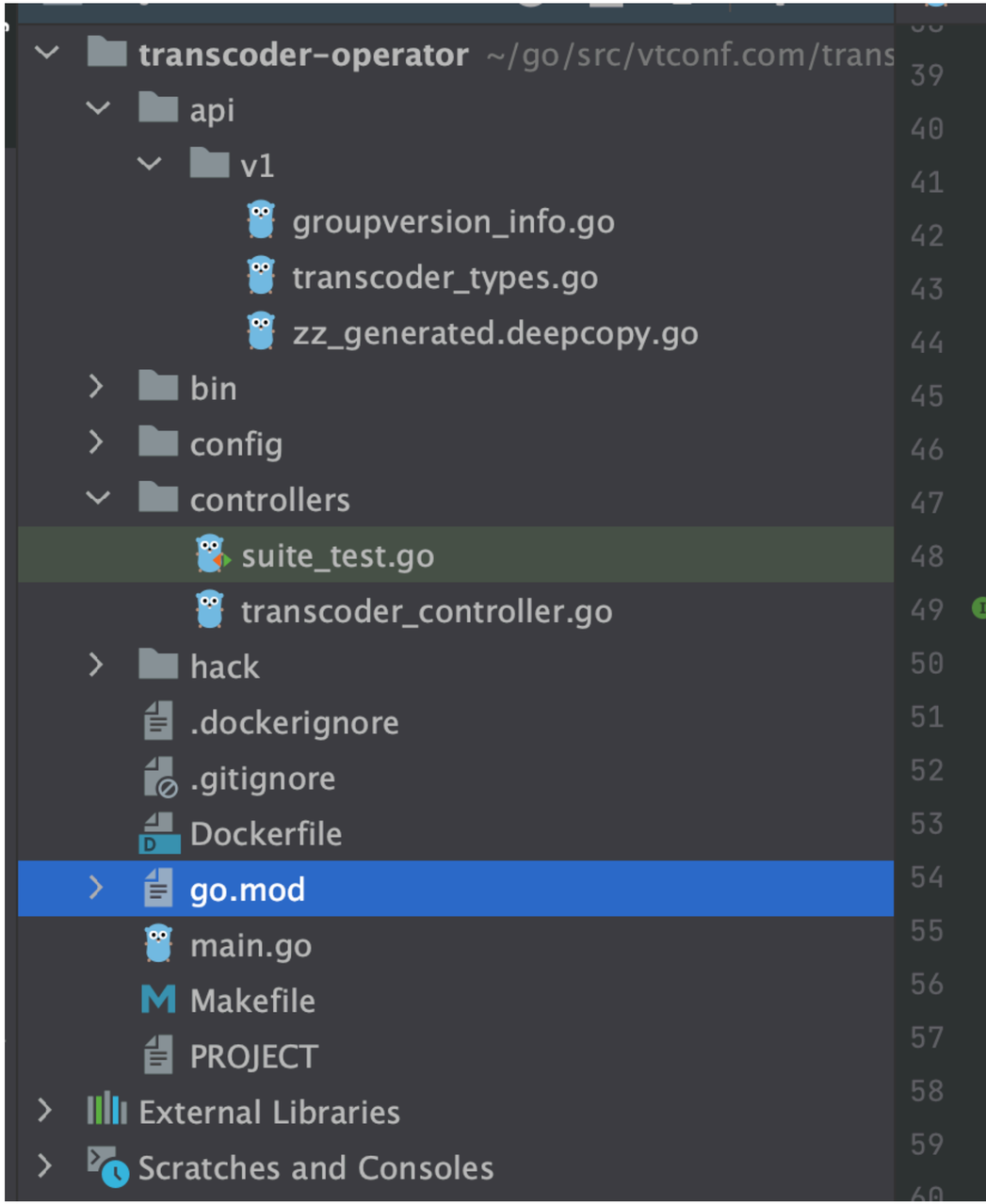
- Возможность указывать требуемые ресурсы для разных серверов
- Проверять, что наши транскодеры живы и справляются с нагрузкой

На чем писать можно?

- OperatorSDK (Go, Ansible, Helm)
- Kubebuilder (Go)
- Kubeapi (практически все, даже PHP есть)

Kubebuilder

- Генерируем костяк приложения
  - kubebuilder init --domain vtconf.com --repo vtconf.com/trans-operator
- Генерируем API и контроллер
  - kubebuilder create api --group transcoder --version v1 --kind Transcoder
- Костяк готов, осталось написать логику



Transcoder CRD

- Создаю свой CRD взамен Deployment’у
- В CRD мне нужно создать карту в Spec’е из запрашиваемых ресурсов для каждой конфигурации

```
apiVersion: transcoding.vtconf.ru/
v1alpha1
kind: Transcoder
metadata:
  generation: 1
  name: transcoder-vtconf
spec:
  node_pool_resources_requests:
    Xeon-6248R:
      cpu: "5.7"
      memory: "1.5Gi"
    Xeon-5118:
      cpu: "7.5"
      memory: "1.5Gi"
```

Концепция работы

Обрабатывает изменения

Подписывается
на изменения Pod'а и CRD

```go
type Result struct {
    Requeue      bool
    RequeueAfter time.Duration
}
type Request struct {
    types.NamespacedName
}
type Reconciler interface {
    Reconcile(context.Context, Request) (Result, error)
}
func (r *TReconciler) SetupWithManager(mgr ctrl.Manager) error {
    return ctrl.NewControllerManagedBy(mgr).
        For(&transcoderv1.Transcoder{}).
        Watches(&source.Kind{Type: &corev1.Pod{}}, &handler.EnqueueRequestForOwner{
            IsController: true,
            OwnerType:    &transcoderv1.Transcoder{},
        }).
        Complete(r)
}
```

Базовая логика Controller’а

- Получаем CRD нашего транскодера
- Получаем Pod, который создался для нашего транскодера
- Если Pod найден и его статус Running, то просто выходим
- Если Pod не найден или его статус Failed, то создаем новый

```go
func (r *Reconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
    tcoder := &transcoderv1.Transcoder{}
    err := r.Client.Get(ctx, req.NamespacedName, tcoder)
    if err != nil {
        if k8sErrors.IsNotFound(err) {
            return reconcile.Result{}, nil
        }
        return reconcile.Result{}, errors.Wrap(err, message: "get transcoder")
    }

    transcoderPod := &corev1.Pod{}
    podKey := types.NamespacedName{Name: "pod_name", Namespace: tcoder.Namespace}
    if err = r.Client.Get(ctx, podKey, transcoderPod); err != nil {
        if k8sErrors.IsNotFound(err) {
            if err := CreatePod(ctx, tcoder); err != nil {
                return reconcile.Result{}, errors.Wrap(err, message: "create pod")
            }

            return reconcile.Result{}, nil
        }

        return reconcile.Result{}, errors.Wrap(err, message: "get pod")
    }

    return ctrl.Result{}, nil
}
```

Получаем CRD транскодера

Пытаемся получить Pod транскодера

Если не нашли Pod, то создаем его, что спровоцирует вызов Reconcile

Проверяем, что Pod жив

```go
transcoderPod := &corev1.Pod{}
podKey := types.NamespacedName{Name: "pod_name", Namespace: tcoder.Namespace}
if err = r.Client.Get(ctx, podKey, transcoderPod); err != nil {
    if k8sErrors.IsNotFound(err) {
        if err := CreatePod(ctx, tcoder); err != nil {
            return reconcile.Result{}, errors.Wrap(err, message: "create pod")
        }
        return reconcile.Result{}, nil
    }

    return reconcile.Result{}, errors.Wrap(err, message: "get pod")
}

if isDeadPod(transcoderPod) {
    if err := DeletePod(ctx, transcoderPod); err != nil {
        return reconcile.Result{}, errors.Wrap(err, message: "delete pod")
    }
}

return ctrl.Result{}, nil
}
```

Pod найден, проверяем, что жив

Pod мертв, удаляем и провоцируем Reconcile

```go
func isDeadPod(pod *corev1.Pod) bool {
    isDead := pod.Status.Phase == corev1.PodFailed ||
        pod.Status.Phase == corev1.PodSucceeded
    if isDead {
        return true
    }

    if pod.Status.Phase != corev1.PodPending {
        return false
    }

    if len(pod.Status.Conditions) == 0 {
        return false
    }

    for _, c := range pod.Status.Conditions {
        if c.Reason == corev1.PodReasonUnschedulable {
            return true
        }
    }

    return false
}
```

Проверяем, что завершился или упал

Проверяем, что он еще ожидает запуска

Проверяем, что для Pod'а нашлись ресурсы

Запуск Pod’a

- Получаю список всех Node
- Получаю из Spec’и кол-во ресурсов, которые будут нужны на каждой из них
- Нахожу все Node, у которых достаточно свободных ресурсов
  - K8s SDK позволяет это сделать одним методом
- При создании Pod’a, указываю в Affinity все найденные Node
- K8s Scheduler сам выберет лучшую по его мнению Node и запустит там транскодер

Проверка работоспособности транскодеров

- Буду проверять, что наши транскодеры работают:
  - Скорость транскодирования >= 1x
  - Входящая очередь фреймов не превышает пороговое значение
  - Количество обработанных фреймов монотонно возрастает
- Для этого смотрю в Prometheus и если что-то не нравится - удаляю Pod

Итоги пользовательского контроллера

- Транскодеры запускаются
- Планирование работает
- Транскодеры мигрируют в случае проблем
- K8S имеет богатый Framework для построения своей логики

Не упадет ли все это?

Отказоустойчивость

- Сам K8S отказоустойчивый
- Controller из коробки умеет в leader-election
- Даже если контроллер умрет, запущенные Pod’ы будут жить

А как всем этим управлять?

Control-plane

- Имеем полный CRUD через kubectl утилиту
- Вспоминаем, что CRD такой же API объект

```
→  ~ kubectl get transcoders.transcoding.wasd.tv
NAME                                       AGE
3180-4ba63b36d8f9425f81817296e5dc1cfa      74m
78905-cd8d4104eba14f0ebadfd3fad587f073     72m
```

Время пускать фейерверки?

А точно все работает?

- Определили, что емкость кластера с транскодерами это 100 одновременных потоков с 1080P@60 10MBit\s потоков
- Застримили все 100 потоков
- Все 100 транскодеров запустились
- Транскодеры постоянно пытаются мигрировать из-за низкой скорости транскодирования
- …
- Коробку с фейерверками убрали обратно

Игра в детектива

Конфигурация #1

- Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz x2
- 24 ядра и 48 потоков в сумме
- 64GB RAM
- Канал 6Gbit/s симметричный
- Должно влезать 7 транскодеров с 1080P@60FPS 10MBit/s с нужными профилями
- Влезло 3….

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
Address sizes:         46 bits physical, 48 bits virtual
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
Stepping:              4
CPU MHz:               2699.995
CPU max MHz:           3200.0000
CPU min MHz:           1000.0000
BogoMIPS:              4600.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              16896K
NUMA node0 CPU(s):     0-11,24-35
NUMA node1 CPU(s):     12-23,36-47
```

Метрики и еще раз метрики

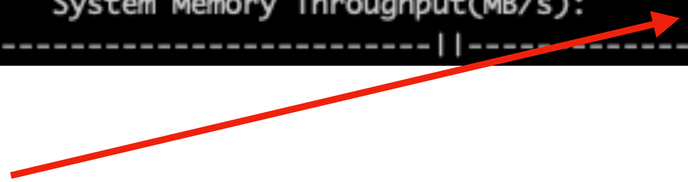
- Смотрю на метрики CPU, RAM, I/O
- CPU Idle на уровне 40%
- Free RAM 60%
- Утилизация I/O 5%
- Транскодеры не утилизируют весь лимит ресурсов
- Что-то точно не так

А может какая-то шина памяти?

- Может где-то упирается в DRAM Bandwidth?
- Чем проверить?
  - Intel - pcm
  - AMD - AMD uProf

Запускаем и замеряем

```
|---------------------------------------||---------------------------------------|
|--             Socket  0             --||--             Socket  1             --|
|---------------------------------------||---------------------------------------|
|--     Memory Channel Monitoring     --||--     Memory Channel Monitoring     --|
|---------------------------------------||---------------------------------------|
|-- Mem Ch  0: Reads (MB/s):  9904.83 --||-- Mem Ch  0: Reads (MB/s): 10204.96 --|
|--            Writes(MB/s):  4263.47 --||--            Writes(MB/s):  4099.78 --|
|-- NODE 0 Mem Read (MB/s) :  9904.83 --||-- NODE 1 Mem Read (MB/s) : 10204.96 --|
|-- NODE 0 Mem Write(MB/s) :  4263.47 --||-- NODE 1 Mem Write(MB/s) :  4099.78 --|
|-- NODE 0 P. Write (T/s):    6245610 --||-- NODE 1 P. Write (T/s):    6694815 --|
|-- NODE 0 Memory (MB/s):    14168.30 --||-- NODE 1 Memory (MB/s):    14304.74 --|
|---------------------------------------||---------------------------------------|
|---------------------------------------||---------------------------------------|
|--                System Read Throughput(MB/s):      20109.80                  --|
|--               System Write Throughput(MB/s):       8363.25                  --|
|--             System Memory Throughput(MB/s):      28473.04                  --|
|---------------------------------------||---------------------------------------|
```

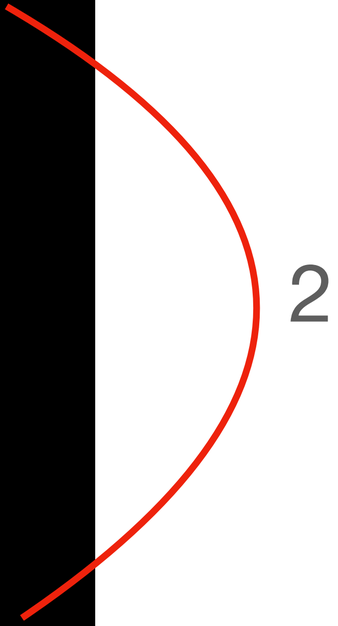


И сколько транскодеров не запускай,
выше этого значения не подымается

А как вообще дела с памятью?

- Xeon 5118 имеет 6 каналов памяти на частоте 2400MHz, на 2 сокета 12 каналов
- Теоретическая макс пропускная способность
  - Общая: 2400 x 64 * 12 / 8 = 230.4GB/s
  - На сокет: 115.5Gb/s
  - На канал: 19.2GB/s
- Как-то не сходится 28.4GB/s vs 230.4GB/s
- А сколько у нас всего планок памяти установлено?

```
description: System Memory
   description: DIMM DDR4 Synchronous Registered (Buffered) 2666 MHz (0.4 ns)
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM DDR4 Synchronous Registered (Buffered) 2666 MHz (0.4 ns)
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
   description: DIMM Synchronous [empty]
description: Memory controller
```




ВСЁ!

Замеряю фактическую скорость

- Исходя из фактической емкости смогу определить настоящую емкость транскодеров
- Хочется узнать:
  - Пропускную способность на сокет
  - Общую пропускную способность
- Чем замерять буду?
  - Intel Memory Latency Checker

Итоги замеров

- Общая
  - Чтение: 34.7GB/s
  - Stream-Triad: 28.6GB/s
- На сокет
  - Чтение: 17.35GB/s
  - Stream-Triad: 14,3GB
- Local Access 17.4GB/s
- Remote Access 16.1GB/s

```
Measuring Peak Injection Memory Bandwidths for the system
Bandwidths are in MB/sec (1 MB/sec = 1,000,000 Bytes/sec)
Using all the threads from each core if Hyper-threading is enabled
Using traffic with the following read-write ratios
ALL Reads        :      34713.9
3:1 Reads-Writes :      32672.4
2:1 Reads-Writes :      32403.9
1:1 Reads-Writes :      31406.7
Stream-triad like:      28623.8

Measuring Memory Bandwidths between nodes within system
Bandwidths are in MB/sec (1 MB/sec = 1,000,000 Bytes/sec)
Using all the threads from each core if Hyper-threading is enabled
Using Read-only traffic type
                Numa node
Numa node            0       1
       0        17494.6 16121.8
       1        16128.1 17457.2
```

Encode/Decode DRAM bitrate

- А сколько памяти прогоняет прогоняет каждый из потоков?
- Замеряю каждый поток с помощью Intel PCM

Type	720P@60 3.6Mbit	1080P@60 10Mbit	4K@60 25Mbit
Only decode	350 MB/s	1.1 GB/s	4.8 GB/s
Encode to ABR Ladder	3.2 Gbit	7.4 GB/s	36.4 GB/s

Все равно не сходится

- Макс. пропускная способность 34.7GB/s
- Потребление шины 1080P@60 потоком 7.4GB/s
- Должно влезать 34,7 / 7,4 = 4,…
- Влезло 3…
- Почему не влазит еще один поток?

NUMA и UPI

- При помощи Intel PCM узнаю, как много данных гоняется между сокетами
- Процессы на сокете пытаются ходить на Remote Node, где шина уже перегружена
- Получается, что NUMA стреляет в ноги

```
               UPI0     UPI1     |  UPI0   UPI1
-------------------------------------------------------------------------------------------------------
 SKT    0     2114 M   2110 M   |    9%     9%
 SKT    1     2175 M   2176 M   |    9%     9%
-------------------------------------------------------------------------------------------------------
Total UPI outgoing data and non-data traffic: 8577 M
```

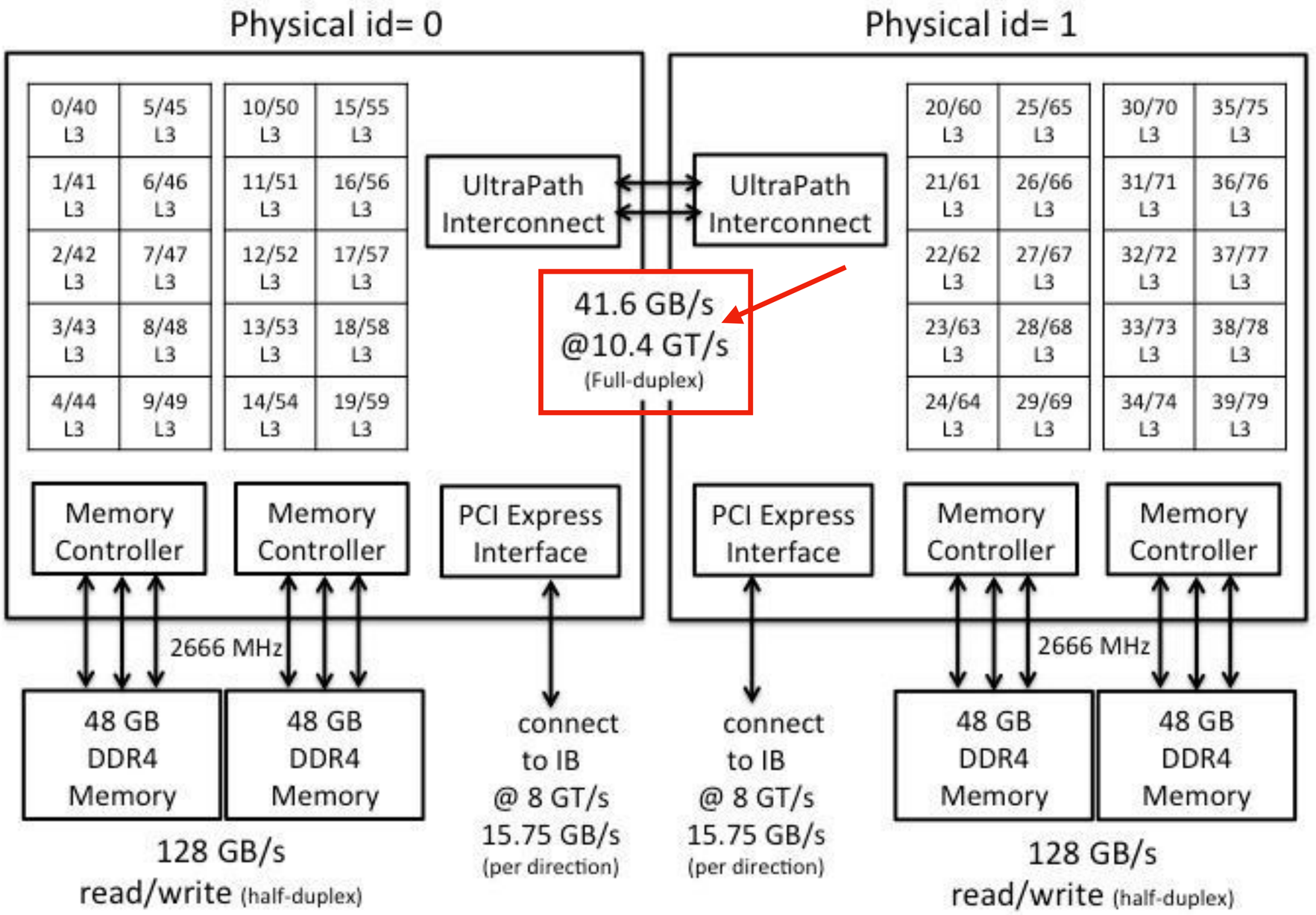
Заколачиваем гвозди

- NUMA виновата, а значит надо исключить её влияние
- Приколачиваю транскодеры каждый к своей NUMA Node
- Запускаю по два транскодера на Node
  - numactl --cpubind=0 --membind=0 ffmpeg….
- Все 4 транскодера работают штатно со скоростью 1x
- Победа!

А если доступен весь канал?

- А нужно ли прикреплять транскодеры к NUMA Node, если бы был доступен весь канал?
- Теор. емкость всех каналов памяти 230.4GB/s
- Должно влезть 230.4 / 7.4 = 31,… потоков 1080P@60 10Mbit/s
- Получается проблема только из-за плохой конфигурации железа?

Configuration of a Skylake-SP Node



Виновата конфигурация?

- Теоретическая пропускная способность к соседнему сокету 41.6GB/s и 83.2GB/s в сумме
- Фактическая пропускная способность между сокетом по тестам Intel MLC на процессоре Xeon Gold 6248R равняется 34GB/s
- Т.е. 34 / 7.4 = 4,… потока влезет, если гонять память меж. сокетами
- Даже правильная конфигурация может выстрелить

Но что там с K8s?

- На конфигурации с 2мя планками, нужно ограничить кол-во одновременно работающих транскодеров
  - В таблице потребления поменяю для 1080 потоков с 7,4 до 12 ядер, чтобы влезало только ровно 4
- Но как прикрепить их к определенным NUMA Node?

Configuration	VideoWidth	VideoHeight	FPS	BitrateKbs	CPU	Memory
Xeon-5118	1920	1080	60	10 000	7,5 -> 12	1.5GB

Kubernetes выручает

Kubernetes CPU && Memory Manager

- Аллоцирует контейнеры на конкретной NUMA Node
- Предоставляет эксклюзивные ресурсы CPU
- Аллоцирует только если ресурсы на Node есть
- Не совместим с Burst

Подвожу итоги

Выводы

- Оркестрация это просто, хоть и пришлось писать код (но совсем немного и быстро!)
- K8s мощный инструмент для построения своей логики оркестрации
- Прозрачный подсчет TCO транскодеров
  - Бизнес знает текущую емкость кластера и сколько железа купить еще, чтобы удовлетворить спрос
- Максимальная утилизация железа без потери качества сервиса
- Стоит уделять внимание физической и виртуальной конфигурации памяти

Спасибо за внимание!

Ilia Galimianov
Telegram: @zloydyadka
ilyaxa1@gmail.com

Links

- Intel MLC - https://www.intel.com/content/www/us/en/developer/articles/tool/intelr-memory-latency-checker.html
- Intel PCM - https://github.com/opcm/pcm
- AMD uProf - https://developer.amd.com/amd-uprof/