

Технический аудит производительности сайта заказчика

На январь 2021

1С-Битрикс: Управление сайтом 20.5.399. © Битрикс, 2016
php 7.2.34
mysql Percona 5.7.32-35

В рамках аудита исследовали производительность каталога, включая:

- детальную карточку товара,
- список товаров,
- фильтры (в том числе умный фильтр),
- поиск.

Проверки средствами 1С Битрикс

Тестирование конфигурации

Подсистема	Оценка	Эталон	Примечание
Конфигурация	19.21	30	
Среднее время отклика	0.0521	0.0330	секунд
Процессор (CPU)	339.4	9.0	миллионов операций в секунду
Файловая система	23 857.7	10 000	файловых операций в секунду
Почтовая система	0.0513	0.0100	время отправки одного письма (в секундах)
Время старта сессии	0.0001	0.0002	секунд
Конфигурация PHP	оптимально	оптимально	рекомендации
База данных MySQL (запись)	16 828	5 600	количество запросов на запись в секунду
База данных MySQL (чтение)	34 171	7 800	количество запросов на чтение в секунду
База данных MySQL (изменение)	14 872	5 800	количество запросов на изменение в секунду

Тестировать конфигурацию

Монитор производительности на боевом сайте не показывает явных проблемных мест. БД и процессор в норме.

Нагруженные страницы

Замер не в часы пиковой нагрузки. Большую нагрузку составляют страницы каталога: “/catalog/”, которые берут на себя половину всей нагрузки.

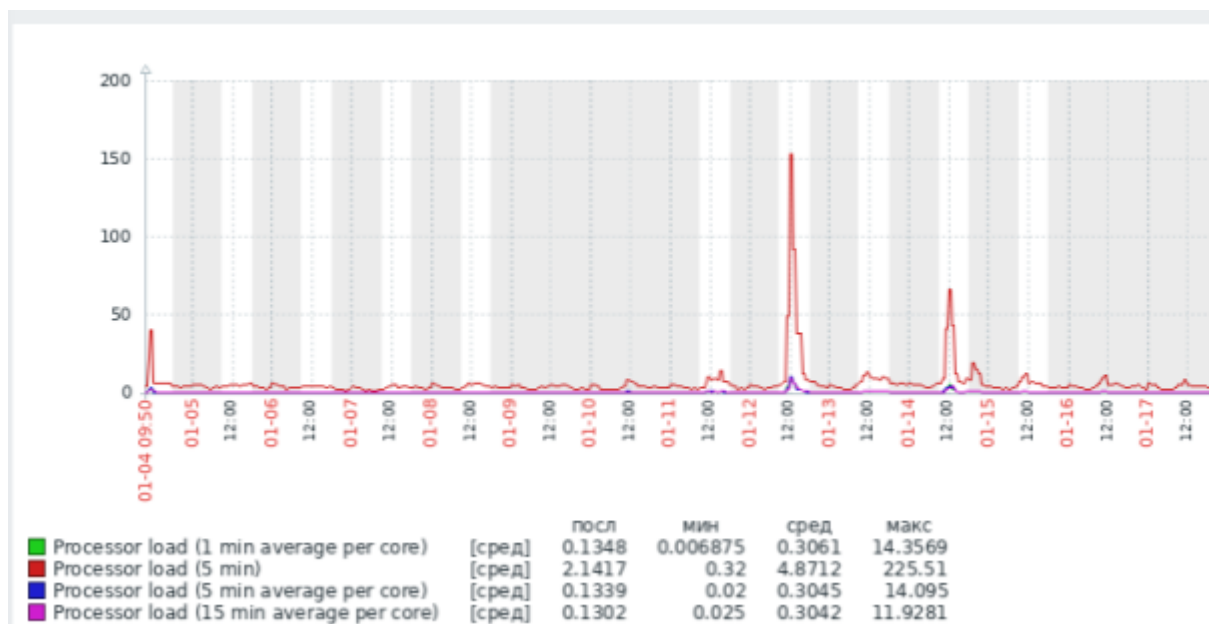
20 самых нагруженных страниц	Ошибки разработки?	Нагрузка	Количество хитов	Среднее время (сек.)
catalog/index.php	1	50.28%	17 360	1.9620
home/hit/cst_wwww.../bitrix/modules/main/include/cron_events.php	1	24.66%	339	49.7639
home/hit/cst_wwww.../localphp_interface/cron_events.php	1	16.89%	103	112.2023
home/hit/cst_wwww.../localization/insert/la.php		1.97%	6	224.6394
local/components/bitrix/catalog.section/ajax.php		1.27%	653	1.3339
/cron/modules/stayalive/index.php	1	0.78%	13	40.8495
shop/index.php	1	0.67%	402	1.1449
/dtd.php	1	0.66%	960	0.4306
local/ajax/suggests.php		0.66%	862	0.4761
search/index.php	2	0.52%	210	1.6943
local/include/catalog.product.php	1	0.26%	109	1.6357
/index.php	1	0.26%	115	1.5231
/personal/cart/index.php	1	0.19%	35	3.7837
bitrix/services/main/ajax.php		0.19%	19	6.6068
employees/detail.php	1	0.17%	238	0.4991
medcenter/index.php	1	0.13%	43	2.0498

Пиковые нагрузки

Нагрузка на сервер (замеры zabbix) за месяц. На скриншотах видно что существуют пиковые нагрузки.

Вопрос к заказчику: чем они могут быть вызваны?

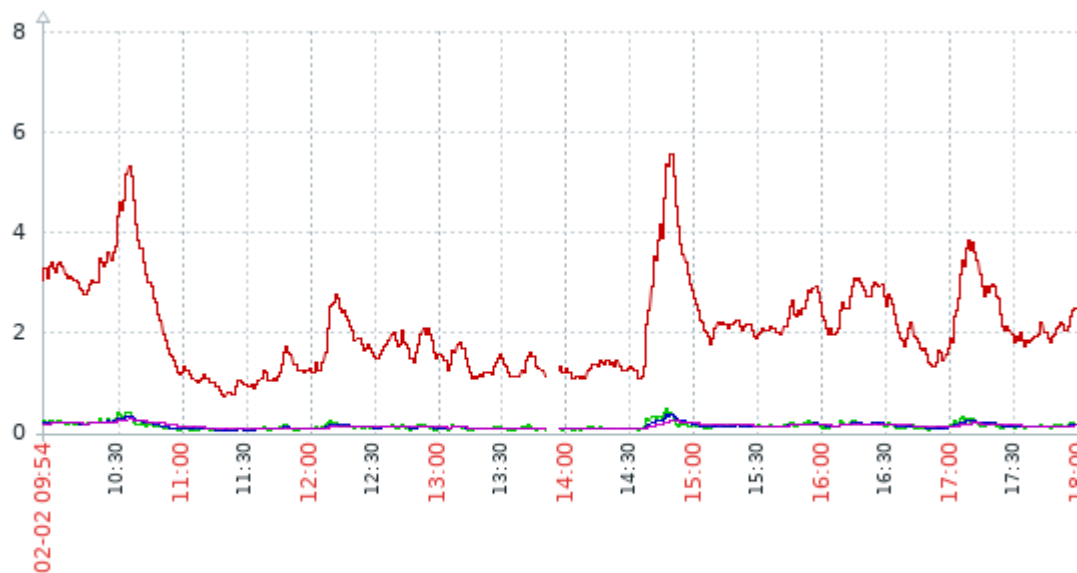
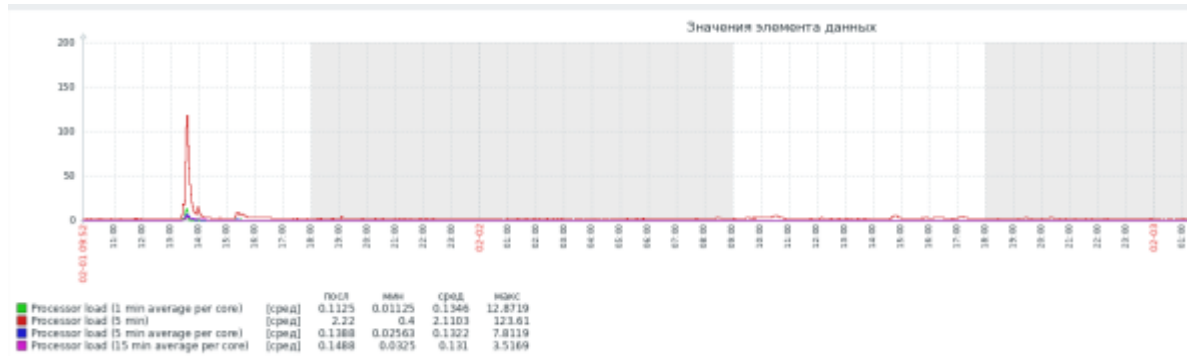
Ответ: рассылками с ссылками на каталог.



Графики ниже иллюстрируют нагрузку при маркетинговой рассылке за 02.02.

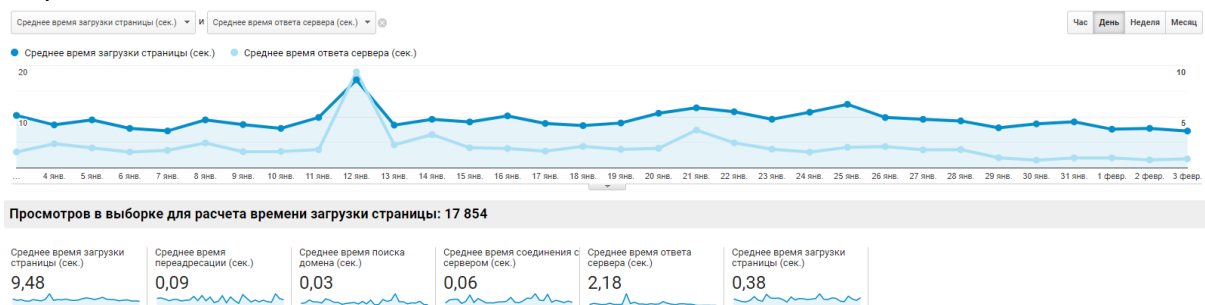
Нагрузка после проведенной рассылки в 14.00 незначительна, особенно на фоне существующих пиковых нагрузок.

Считаем, что это связано с тем, что текущая рассылка не вела непосредственно на каталог.



	тип	посл	мин	сред	макс
Processor load (1 min average per core)	[сред]	0.05813	0.01188	0.1061	0.4956
Processor load (5 min)	[сред]	1.85	0.43	1.6949	5.56
Processor load (5 min average per core)	[сред]	0.1138	0.02625	0.106	0.3481
Processor load (15 min average per core)	[сред]	0.1388	0.03438	0.1059	0.2619

Google Analytics демонстрирует, что среднее время ответа сервера колеблется в рамках 1,5-2с, а после отключения настройки “перенос js в конец страницы” (31.01) сократилось до 0.8с.



Исходя из данных zabbix и количеству пользователей по Google Analytics можно сделать вывод, что количество одновременных хитов при рассылке за 02.02 ~85 в минуту или **~1,4 хит в секунду**.

Перенос js в конец страницы

Замер времени ответа сервера **рабочего сайта** с включенной настройкой “Перенос js в конец страницы” в сравнении с отключенной настройкой:

```
[78]: test('prod.json')
#бой с галкой
```

			back_start	back_total	unload	dom	front	total
url mode								
https://	/	0	1222.4	1852.7	0.2	931.7	2239.3	3481.3
https://	/about/	0	372.2	408.6	0.0	309.8	989.3	1375.4
https://	/catalog/ortopedicheskaya-obuv/	0	2194.2	2359.4	0.9	1356.8	1833.5	4049.0

```
[79]: test('prod2.json')
#бой без галки
```

			back_start	back_total	unload	dom	front	total
url mode								
https://	/	0	399.2	1453.1	1.0	1557.5	2973.8	3427.7
https://	/about/	0	192.5	264.8	0.7	624.6	1704.8	1952.3
https://	/catalog/ortopedicheskaya-obuv/	0	480.5	1048.0	0.7	2301.2	2812.9	3335.1

Вывод: время ответа сервера на боевом сайте незначительно в рамках общего времени загрузки страницы и колеблется в пределах секунды. О настройке “Перенос js в конец страницы” подробнее в разделе [Чистая скорость загрузки бэк/фронт](#). Имеет смысл искать проблему на фронтальной части сайта.

Нагруженные запросы и кеш

Замеры *секции каталога*:

с кешем, с частичным сбросом кеша и без:
/catalog/ortopedicheskaya-obuv/

Время создания страницы: 0.2702 сек.
Всего SQL запросов: 67
Время исполнения запросов: 0.0128 сек.

Время создания страницы: 15.4216 сек.
Всего SQL запросов: 810
Время исполнения запросов: 14.4193 сек.

Время создания страницы: 52.3882 сек.
Всего SQL запросов: 3998
Время исполнения запросов: 50.3486 сек.

Разница в количестве запросов существенная, потому что большая часть логики Битрикса рассчитана на использование кеша.

Детальная

На примере *детальной страницы* видим, что полный сброс кеша на странице не показателен, так как в реальных условиях при переходе на страницу товара из рассылки будет происходить сценарий с частичным сбросом кеша (обновление товара или остатков сбрасывает только кеш компонента отображающий товар).

На скриншотах детальная:

С кешем

Время создания страницы: 0.1864 сек.
Всего SQL запросов: 62
Время исполнения запросов: 0.0126 сек.

С частичным сбросом кеша

Время создания страницы: 3.1313 сек.
Всего SQL запросов: 1886
Время исполнения запросов: 1.665 сек.

Полный сброс кеша

Время создания страницы: 11.277 сек.
Всего SQL запросов: 2725
Время исполнения запросов: 9.7856 сек.

Время создания страницы: 0.3923 сек.
Всего SQL запросов: 65
Время исполнения запросов: 0.0123 сек.

Время создания страницы: 0.8964 сек.
Всего SQL запросов: 655
Время исполнения запросов: 0.4341 сек.

Время создания страницы: 10.2578 сек.
Всего SQL запросов: 1114
Время исполнения запросов: 9.4992 сек.

Обновление товара также обновляет данные для доставки и самовывоза (функция входит в состав компонента товара), а также сбрасывает кеш для каждого отдельного расчета данных доставки торгового предложения.

`/local/templates/sitename_desktop/components/bitrix/catalog.element/.default/result_modifier.php:210`

```

// Товар в корзине или нет?
$arSku['IS_IN_BASKET'] = isInBasket($offer['ID']); // проверка предложений в цикле на наличие в кор

// Данные для доставки и самовывоза
$arSku['DELIVERY_TERMS'] = array();
$arDelivery = $orteka->getDelivery($offer['ID'], $curr_city, $arResult['TIMESTAMP_X'], $arResult["

// Если нет в салонах текущего города но есть в регионе то выводим все салоны текущего региона с в
if (empty($arResult['STOCKS_DATA']['CITIES'][$curr_city['ID']]) && !empty($arResult['STOCKS_DA
    $arResultStocksVisible = [];
    $arResultSalonsIds = [];

    foreach (GlobalData::I()->AR_IM_SHOPS as $salon) {
        if ($salon['CITY_ID'] == $curr_city['ID']) {
            $arResultSalonsIds[] = $salon['ID'];
        }
    }
}

```

При большой посещаемости сайта в связи с этим может генерироваться большая нагрузка, ~ 60% от общего времени генерации страницы.

Циклические запросы к корзине из того же компонента:

/local/templates/sitename_desktop/components/multisite/products.list/menu_catalog/template.php:74

```

67
68     <div class="b-card_bottom">
69         <?php
70         $offer = reset( &array: $product['OFFERS']);
71         // количество товаров/офферов в корзине
72         $isInBasket = 0;
73         foreach($product['OFFERS'] as $offer) {
74             $isInBasket += isInBasket($offer['ID']);
75         }
76     >?php
77     <button type="button" class="g-link g-link_primary g-link
78         data-element="<?= $product['ID'] ?>"
79         data-id="<?= $offer['ID'] ?>"

```

Рефакторинг циклических запросов позволит выиграть на текущий момент ~0.1с.

Тяжелый запрос который происходит на каждой странице:

/local/classes/general/globalsettings.php:39

```

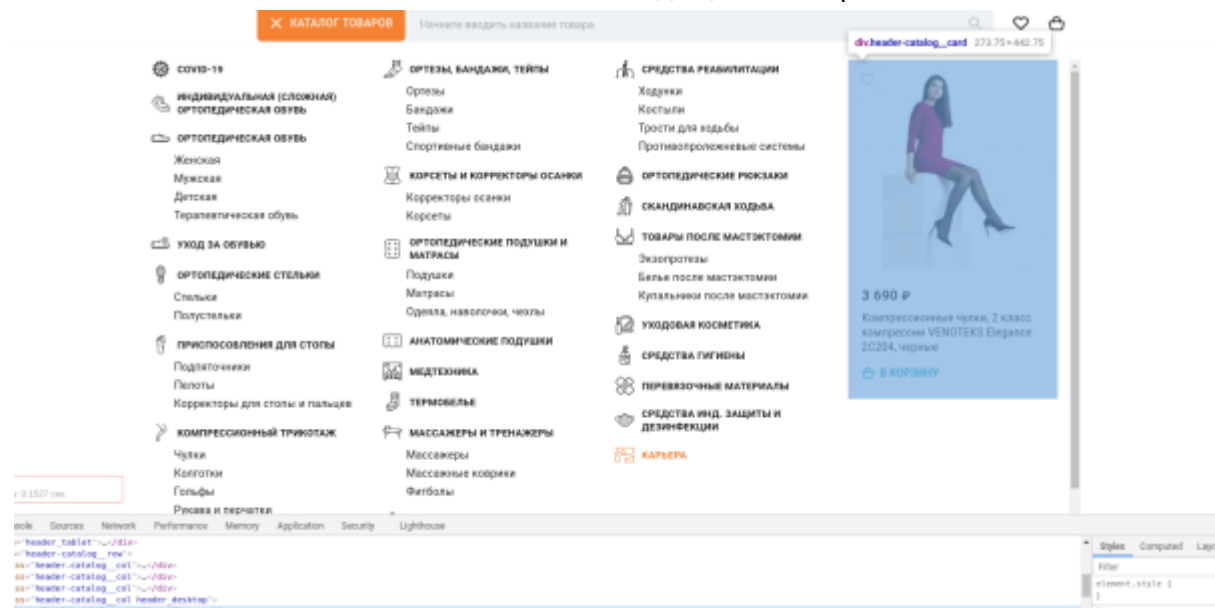
private function __construct() {
    $this->loadAllConfigurations();
    // dd($this);
    if(CModule::IncludeModule( module_name: "iblock")) {
        $groups = [];
        $rsSections = \CIBlockSection::GetList([], ['IBLOCK_ID' => 2]);
        while ($aGroup = $rsSections->Fetch()) {
            $groups[$aGroup['CODE']] = [$aGroup['IBLOCK_ID']];
        }
        $this->CATALOG_PATH_BY_IBLOCK = $groups;
    }

    global $APPLICATION;
    if(empty($_REQUEST['SECTION_CODE'][0])) {
        $url = $APPLICATION->GetCurDir();
    }
}

```

Шапка

В шапке в меню каталога есть компонент, выводящий 1 товар:



`/local/include/header.menu_catalog.product.php:41`

```
if (!empty($productCodes)) {
    $APPLICATION->IncludeComponent(
        componentName: 'multisite:products.list',
        componentTemplate: 'menu_catalog',
        [
            'USE_ELASTICSEARCH' => 'N',
            'MENU_CATALOG_PRODUCT' => 'Y',
            'FILTER' => $arrFilter,
        ],
        parentComponent: false
    );
}
```

При обновлении любого товара в нем так же сбрасывается кеш.

Однако, так как в нем отключена опция использования **Elasticsearch**, он делает несколько тяжелых запросов в бд.

`/local/components/multisite/products.list/class.php:214`

```
199 if (count($iblockIdToProductIds)) {
200     // для некоторых списков, например, для рекомендаций, нужно кэшировать по элементам
201     // для снижения нагрузки из-за того что товар выводится в случайном порядке
202     if (
203         $this->arParams['MENU_CATALOG_PRODUCT'] === 'Y'
204         || $this->arParams['GET_BY_ONE'] === 'Y'
205     ) {
206         foreach ($iblockIdToProductIds as $iblockId => $productIds) {
207             foreach ($productIds as $iOneProductId) {
208                 if (
209                     isset($this->arParams['USE_ELASTICSEARCH'])
210                     && $this->arParams['USE_ELASTICSEARCH'] === 'Y'
211                 ) {
212                     $products[] = ElasticElement::getFull($iOneProductId, useCache: true, inner_folder: 'product_getfull'.GlobalSettings::I()->SITE_ID);
213                 } else {
214                     $products[] = Product::getFull($iOneProductId, useCache: true, inner_folder: 'product_getfull'.GlobalSettings::I()->SITE_ID);
215                 }
216             }
217             $allIds = array_merge($allIds, $productIds);
218         }
219     } else {
220         foreach ($iblockIdToProductIds as $iblockId => $productIds) {
221             foreach ($productIds as $iOneProductId) {
222                 $products[] = ElasticElement::getFull($iOneProductId, useCache: true, inner_folder: 'product_getfull'.GlobalSettings::I()->SITE_ID);
223             }
224         }
225     }
226 } else {
227 }
```

include/component/_includePHPTemplate	61 519	75.1 %
include/cvat/www/html/components/footer/main_include/templates/default/template.php	61 574	75.1 %
include/cvat/www/html/local/include/header/menu_catalog_product.php	61 673	75.1 %
CMain->includeComponent	61 806	76.9 %
CBaseComponent->includeComponent	61 806	76.9 %
ProductList->includeComponent	61 805	76.9 %
iMainCatalogProductGetFull	9 374	11.8 %
iMainCatalogProductGetFull	8 748	10.6 %
iMainCatalogProductGetFull	8 649	10.5 %
iMainCatalogProductGetFull	8 374	10.2 %
iMainCatalogProductGetFull	8 403	10.2 %
iMainCatalogProductGetFull	8 419	10.2 %
iMainCatalogProductGetFull	8 368	10.2 %
ProductList->includeComponentTemplate	583	0.7 %

Выводы

При сбрасывании кеша генерируется много (тысячи) запросов каталога.

Проблема существует в нескольких местах, описанных выше - в запросе в шапке и в запросе в компоненте товара - доставке.

Мы рекомендуем оптимизировать эти функции, и провести нагрузочное тестирование, в ходе которого понять, насколько удалось облегчить количество запросов, а также параллельно, возможно найти дополнительные проблемные места.

Рекомендуем провести рефакторинг циклических запросов, а также проверить:

- возможно ли настроить кеш в функции доставки таким образом, чтобы он не сбрасывался каждый раз при обновлении товара и оптимизировать код функции.

Для оптимизации работы функции delivery:

- гибко настроить кеш, чтобы не было сброса при обновлении товара (3-4ч),
- или переписать функцию полностью (завязать на товар из 1С) (20ч).

Для оптимизации запроса в шапку:

- уточнить, зачем был убран Elasticsearch
- настроить тегирование, которое сбрасывает весь кеш инфоблока (до 4ч)
- отрефакторить функцию (до 15ч)

Фронт

Фронтальная часть сайта грузится 6-7с в среднем.



Это связано со сторонними подключениями маркетинговых программ, не оптимальным решением подгрузки скриптов js и скрытыми элементами dom.

Сторонние подключения

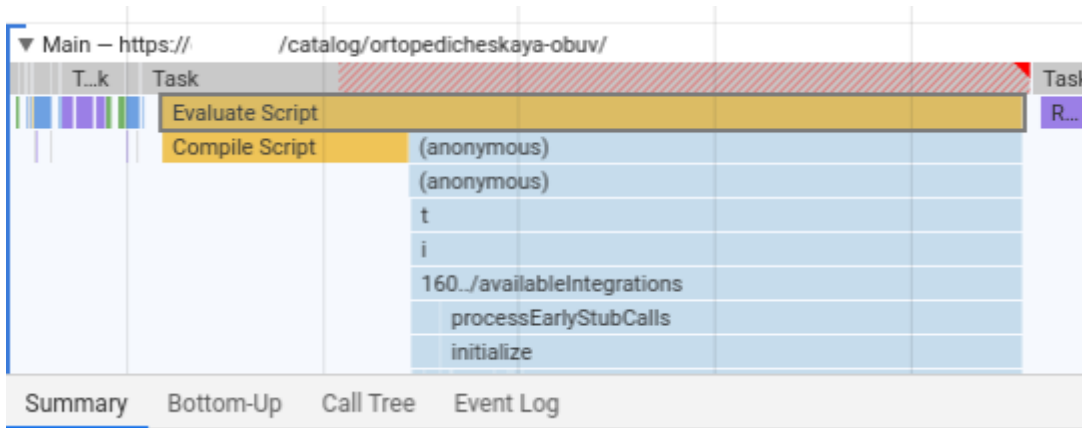
Список:

- [segmentstream.min.js](#)
- [goodmod.ru](#)
- [k50tracker2.js](#)
- [criteo.net](#) (Реклама в соответствии с покупательскими трендами.)
- большой нагрузки не замечено)
- [flocktory](#) (Маркетинговая платформа для управления жизненным циклом клиента. Большой нагрузки не замечено)
- [enpop.min.js](#) (Большой нагрузки не замечено)
- [fbevents.js](#) (Большой нагрузки не замечено)
- [vk openapi](#) (Большой нагрузки не замечено)
- [Яндекс.Метрика](#)
- [recaptcha](#)
- [sherlockcrm](#)

[segmentstream.min.js](#)

Продвинутая аналитика для маркетинга на основе данных.

Загружается в течении 0.2с



■ Evaluate Script

Total Time 219.23 ms

Self Time 0.18 ms

Script segmentstream.min.js?1610394792323:1

goodmod.ru

<https://goodmod.ru/>

Блокировщик Советника Яндекс.Маркета v2.5.

```

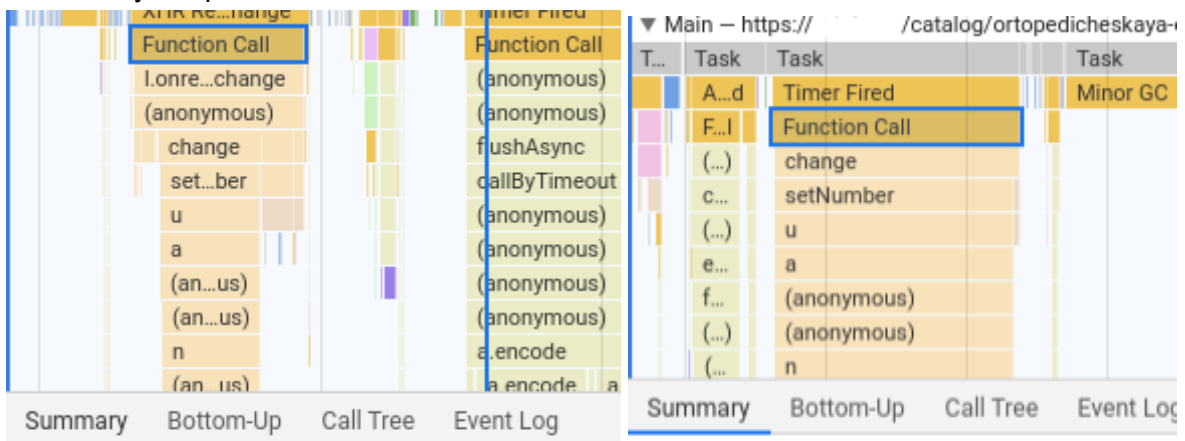
<!-- goodmod блокировка яндекс.советника -->
<script defer data-skip-moving="true" src="https://goodmod.ru/scripts/7651fd27b18b8585c220db02449cacba/api.js"></script>
<!-- //goodmod -->

<!-- SegmentStream snippet UPD -->

```

k50tracker2.js

~170 мс суммарно



■ Function Call

Total Time 30.59 ms

Self Time 0.39 ms

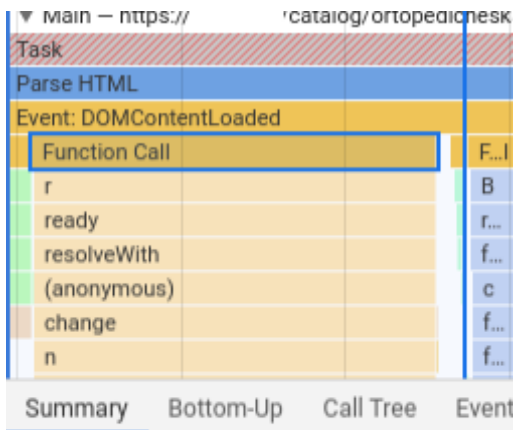
Function l.onreadystatechange @ k50tracker2.js:1

■ Function Call

Total Time 17.54 ms

Self Time 0.15 ms

Function change @ k50tracker2.js:1



Function Call

Total Time 123.27 ms

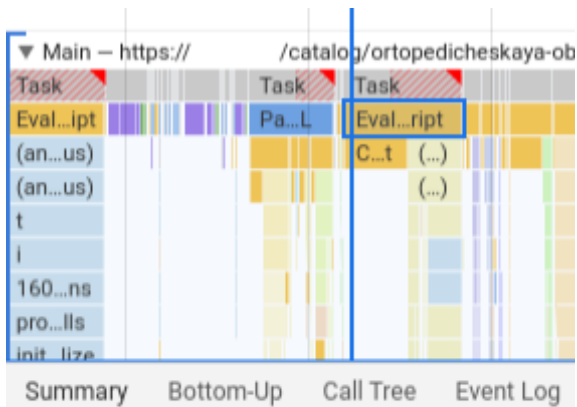
Self Time 0.11 ms

Function r@k50tracker2.js:1

Яндекс.Метрика

<https://mc.yandex.ru/metrika/tag.js>

Прогружается 1-1,5с. Часто блокирует поток маленькими вставками для Вебвизора.

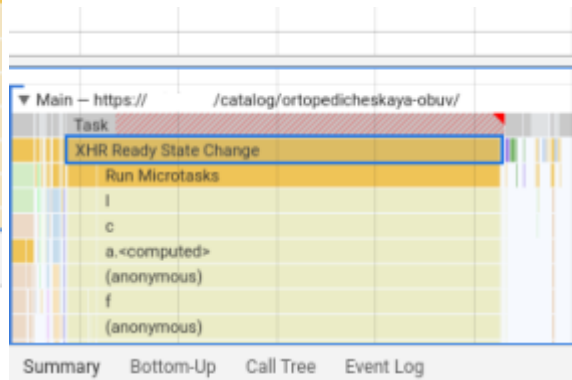


Evaluate Script

Total Time 127.01 ms

Self Time 0.27 ms

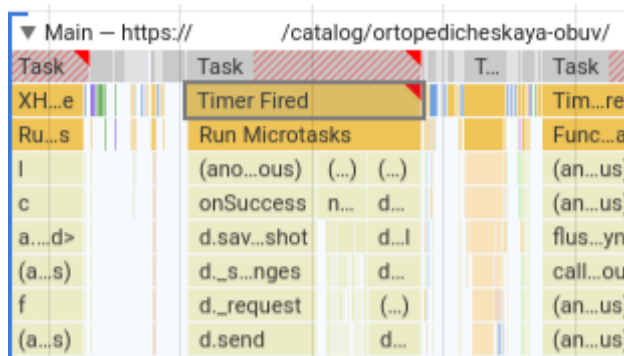
Script tag.js:1



XHR Ready State Change

Total Time 440.42 ms

Self Time 0.14 ms



Summary Bottom-Up Call Tree Event Log

Timer Fired

Warning Recurring handler took 179.40 ms

Total Time 179.40 ms

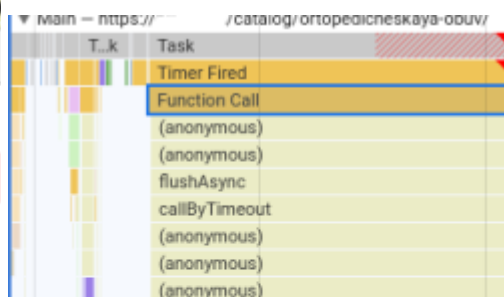
Self Time 94 μs

Timer ID 38

Pending for 494.2 ms

Initiator [Reveal](#)

Timer Installed I @ [tag.js:1](#)



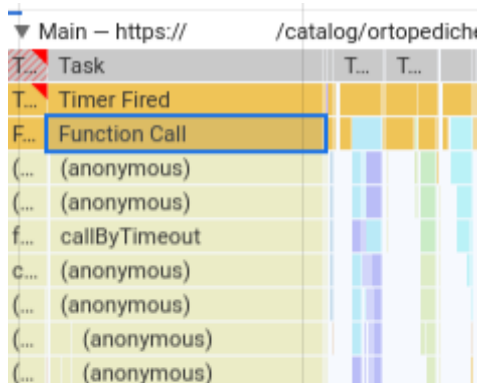
Summary Bottom-Up Call Tree Event Log

Function Call

Total Time 79.23 ms

Self Time 0.25 ms

Function (anonymous) @ [tag.js:1](#)



Summary Bottom-Up Call Tree Ev

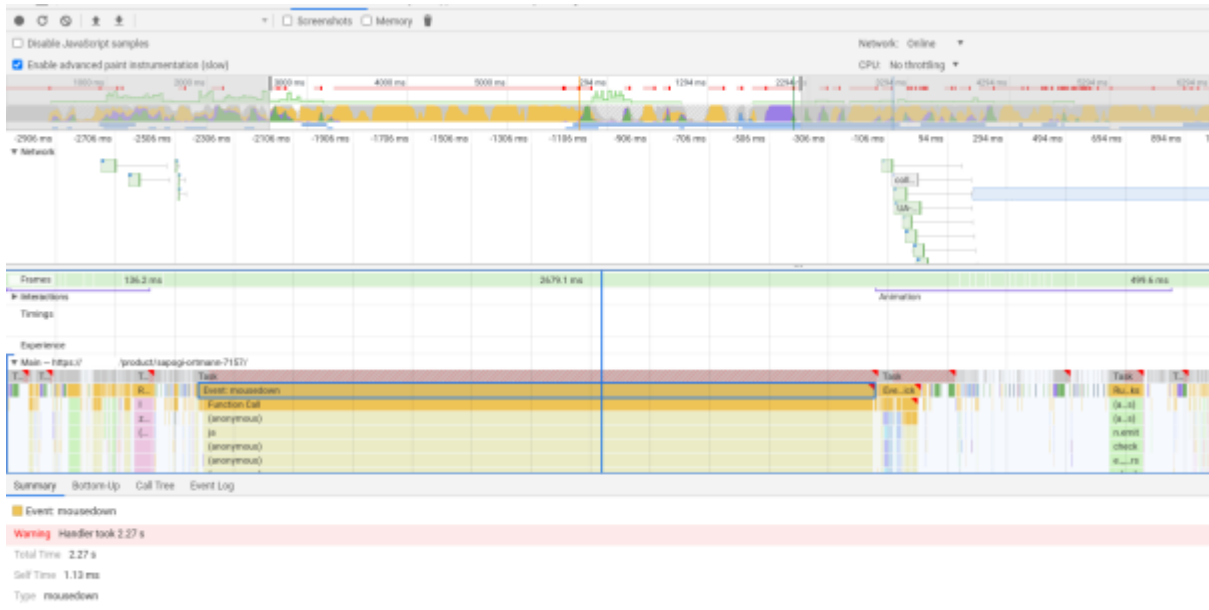
Function Call

Total Time 48.57 ms

Self Time 0.17 ms

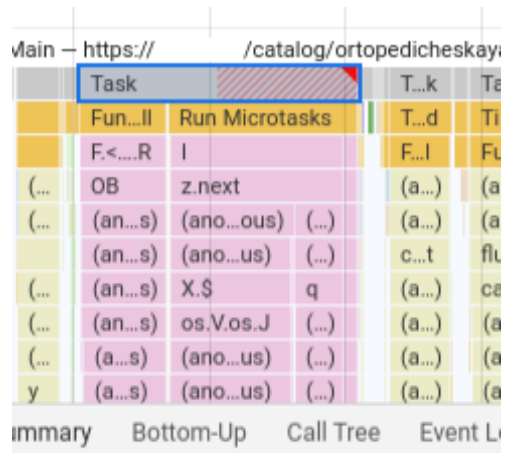
Function (anonymous) @ [tag.js:1](#)

При выгрузке страницы или переходе между страницами, могут возникать большие задержки - отловили ожидание в 2с.



Ресарча

Подключается в коде, т.е. есть возможность вынести после загрузки страницы. Однако, капча 3 поколения, скорее всего переносить ее на отложенную загрузку не стоит.



Task

Warning [Long task](#) took 101.66 ms.

Total Time 101.66 ms

Self Time 0.34 ms

Sherlockcrm

<http://sherlock.im/>

Коммуникации в цифровых каналах, мессенджеры и чат боты.

На сайт внедрен чатбот:

chatBot.js

<https://sitename.sherlockcrm.ru/api/scripts/chatplatform/chatBot.js>

/local/templates/sitename_desktop/js/widget/sherlock_plus/include.php

```
1 <?php
2
3 use ...
6
7 $asset = Asset::getInstance();
8 $asset->addCss('https://...sherlockcrm.ru/api/Scripts/ChatPlatform/chatBot.css');
9 $asset->addCss(SITE_TEMPLATE_PATH . '/js/widget/sherlock_plus/style.css');
10 $asset->addJs('https://...sherlockcrm.ru/api/scripts/chatplatform/chatBot.js');
11 //$asset->addJs(SITE_TEMPLATE_PATH . '/js/widget/sherlock_plus/chatBot.js');
12
13 $settings = [
14     'channelsIds' => Config::get('sherlock.widget/channelsIds'),
15     'logoUrl' => SITE_TEMPLATE_PATH . "/js/widget/sherlock_plus/img/..._logo.svg",
16     'theme' => [
17         'type' => "Default",
18         'closeIconUrl' => SITE_TEMPLATE_PATH . "/js/widget/sherlock_plus/img/close_kras.svg",
19         'attachIconUrl' => SITE_TEMPLATE_PATH . "/js/widget/sherlock_plus/img/attach.svg"
20     ]
21 ];
22
23 $settings = Web\json::encode($settings);
24
25 <script>
26 /**
27  * Sherlock Plus
28  */
29 !function () {
30     function cb() {
31         if (typeof window.SherlockChat !== "undefined") {
32             var widget = new window.SherlockChat('https://...SherlockCRM.ru/api', {
33                 externalId: null,
34                 settings: Object.assign({}, {
35                     chatFormMode: 1,
36                     channelsMode: 1,
37                     editorMode: 1,
```

Task	Task
Evaluate Script	Anir
(anonymous)	Fun
(anonymous)	(anc
t	c_e
(anonymous)	(anc
t t	(a...) e.pr
(...) (anonymous)	C...t fast
(...) t	(anc
r (anonymous)	(an..

Summary Bottom-Up Call Tree Event Log

Evaluate Script

Total Time 192.77 ms

Self Time 0.14 ms

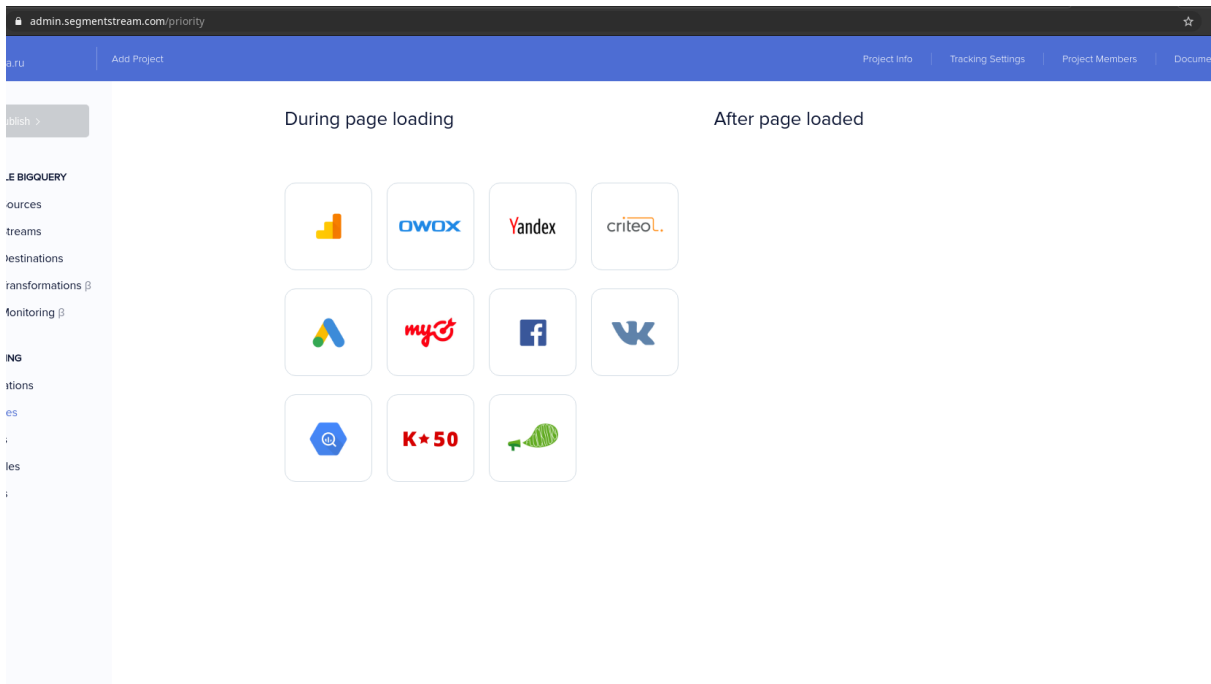
Script [chatBot.js:1](#)

Рекомендуем перенести на отложенную загрузку.

Segmentstream

<https://admin.segmentstream.com/integrations>

[Панель управления](#) Segmentstream - здесь можно руками переключить время подгрузки трекеров.



Изменить время загрузки большинства трекеров из кода нельзя, так как они подключаются через SegmentStream и управлять их загрузкой можно только из приложения.

Project Info

Project name

Project ID

HTTP API token

Website snippet code

```
<!-- SegmentStream snippet -->
<script type="text/javascript">
(function(h,d){d=d||"cdn.segmentstream.com";var a=window
.segmentstream=window.segmentstream||[];window.ddListener
=window.ddListener||[];var b=window.digitalData=window
.digitalData||{};b.events=b.events||[];b.changes=b
.changes||[];if(!a.initialize)if(a.invoked)window
.console&&console.error&&console.error("SegmentStream
snippet included twice.");else{a.invoked=!0;a.methods
="initialize addIntegration persist unpersist on once off
getConsent setConsent".split(" ");a.factory=function(k
){return function(){var c=Array.prototype.slice.call
(arguments);c.unshift(k);a.push(c);return a}};for(b=0;b<a
.methods.length;b++){var f=a.methods[b];a[f]=a.factory(f)}a
.load=function(a){var c=document.createElement("script");c
.type="text/javascript";c.charset="utf-8";c.async=!0;c.src
```

Save

Поддержка

Segmentstream подключает в коде один раз с помощью запроса выше. Это не позволяет нам самостоятельно управлять отложенной загрузкой маркетинговых инструментов.

/local/templates/sitename_desktop/header.php:139

```

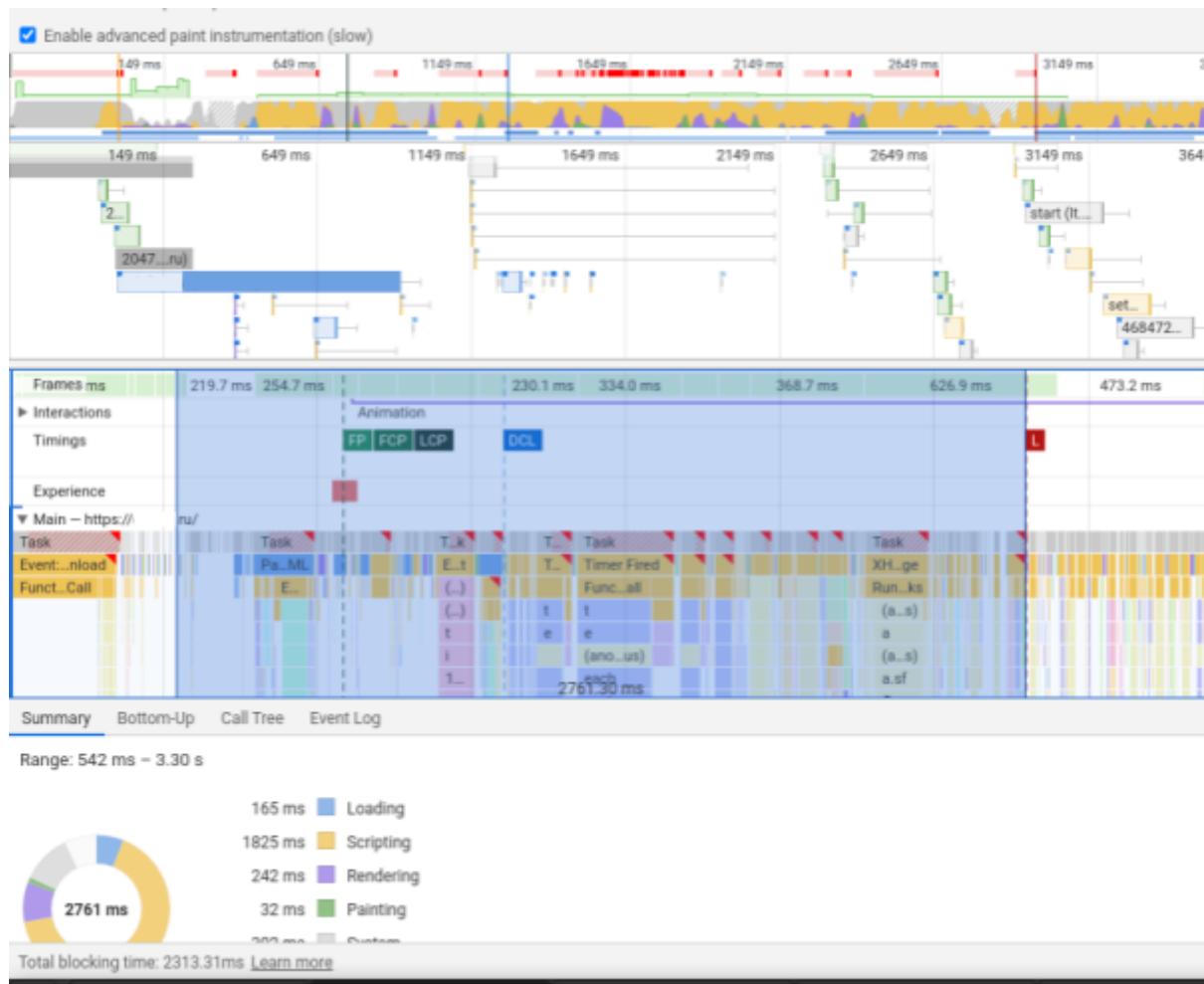
27
28 <script>
29     window.user = {
30         auth : <?=$USER->isAuthorized() ? 'true' : 'false'?>
31     };
32 </script>
33
34 <!-- goodmood блокировка яндекса.советника -->
35 <script defer data-skip-moving="true" src="https://goodmod.ru/scripts/7651fd27b18b8585c2
36 <!-- //goodmood -->
37
38 <!-- SegmentStream snippet UPD -->
39 <script data-skip-moving="true">
40     (function(h,d){d=d||"cdn.segmentstream.com";var a=window.segmentstream=window.segmentstream||[];
41 </script>
42 <!-- //SegmentStream snippet -->
43
44 </head>
45 <body>
46 <div id="panel"><?php $APPLICATION->ShowPanel() ?></div>

```


Вывод

Перенести трекары на отложенную загрузку, что можно - руками в Segmentstream, что можно - в коде.

после переноса части сервисов в отложенную загрузку, уменьшилось общее время загрузки страницы

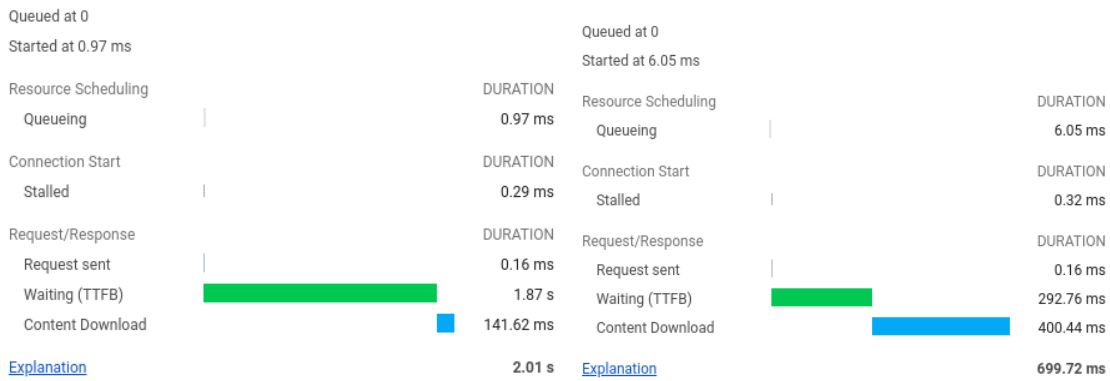


Оптимизация JS / CSS

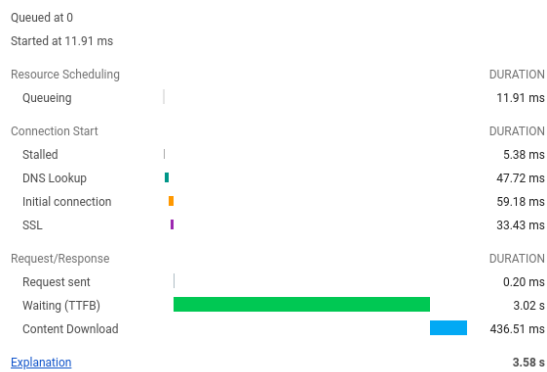
Отключение CSS сильно снижает нагрузку на сервер, тк оптимизация тратит много времени на парсинг страницы.

Оптимизация CSS	
Объединять CSS файлы:	<input checked="" type="checkbox"/>
Объединять JS файлы:	<input checked="" type="checkbox"/>
Подключать минифицированные версии CSS и JS файлов:	<input checked="" type="checkbox"/>
Переместить весь Javascript в конец страницы:	<input checked="" type="checkbox"/>
Создавать сжатую копию объединенных CSS и JS файлов:	<input checked="" type="checkbox"/>

Время, затрачиваемое на запрос, на тестовой среде (включено и отключено соответственно):

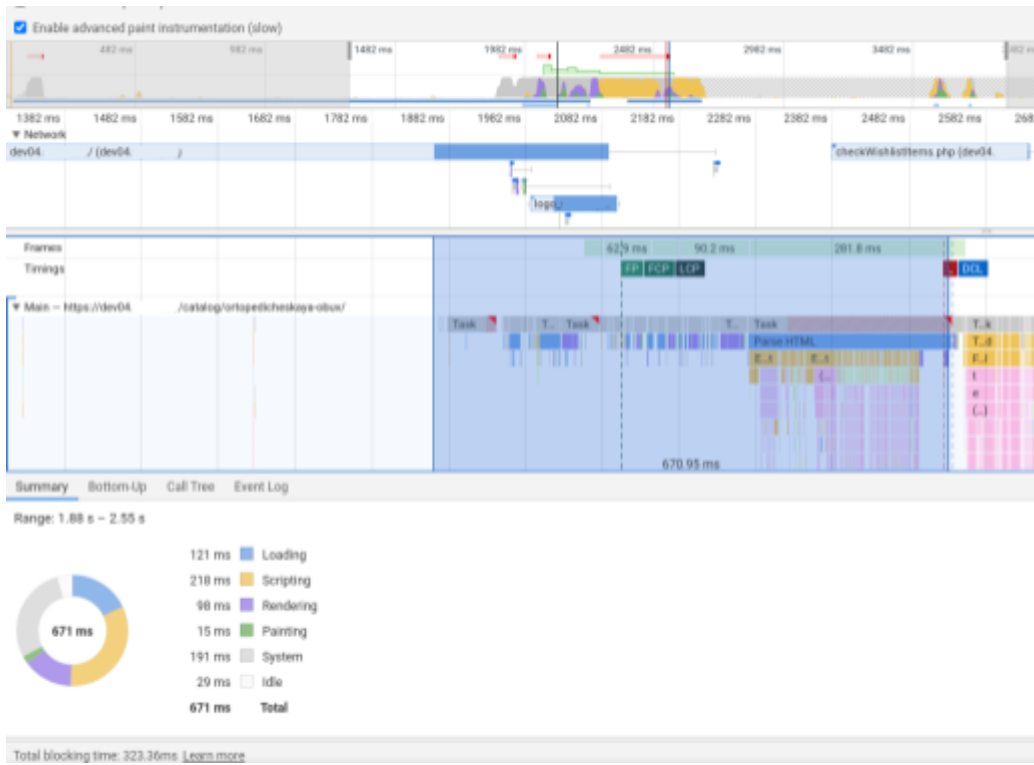


Время затрачиваемое на запрос, на боевом сайте, которое ранее [демонстрировали](#) в виде таблицы (включено):



Тесты проводились с отключенным segmentstream, гесарча и чатботом на тестовой среде dev04.sitename.ru

С включенной оптимизаций: получение страницы от сервера ~1,8 сек, обработка страницы браузером ~0,65 сек, суммарно ~2,5 сек.



С отключенной : получение страницы от сервера ~0,3 сек, обработка страницы браузером ~1,1 сек, суммарно ~1,46 сек.



Вставки js на странице

примеры вставок js скриптов в тело страницы которые могут замедлять работу сервера при включенной опции “перенос js в конец страницы”

скрипт
https://dev04.sitename.ru/catalog/ortopedicheskaya-obuv/
https://dev04.sitename.ru/
https://dev04.sitename.ru/product/botinki-berkemann-2018/

В каталоге присутствуют вставки js, реализующие стандартные обработчики битрикс. Ниже приведено несколько примеров нагруженных участков.

Каталог

/local/templates/sitename_desktop/components/bitrix/system.pagenavigation/.default/template.php:84

повторяется на каждый товар, много данных

/local/templates/sitename_desktop/components/bitrix/catalog.item/.default/template.php:283

повторяется на каждый товар

/local/templates/sitename_desktop/components/bitrix/catalog.item/.default/card/template.php:244

/local/templates/sitename_desktop/components/bitrix/catalog.smart.filter/.default/template.php:380

/local/templates/sitename_desktop/components/bitrix/catalog.smart.filter/.default/template.php:227

/local/templates/sitename_desktop/components/bitrix/favorite.line/version2/template.php:17

/local/components/sitename/auth.registration/templates/header/template.php:73

/local/templates/sitename_desktop/header.php:128

много данных

/local/templates/sitename_desktop/components/bitrix/catalog.section/.default/template.php:269

/local/templates/sitename_desktop/footer.php:76

Детальная страница

/local/templates/sitename_desktop/components/bitrix/catalog.element/.default/template.php:503

много данных

/local/templates/sitename_desktop/components/bitrix/catalog.element/.default/template.php:534

Главная страница

много данных

/local/templates/sitename_desktop/components/bitrix/news.list/salons_mainpage/template.php:163

Настройка “Перенос js в конец страницы”

Замер рабочего сайта со включенной настройкой “перенос js в конец страницы” в сравнении с отключенной.

```
[78]: test('prod.json')  
#бой с галкой
```

```
[78]:
```

				back_start	back_total	unload	dom	front	total
	url	mode							
	https://	/	0	1222.4	1852.7	0.2	931.7	2239.3	3481.3
	https://	/about/	0	372.2	408.6	0.0	309.8	989.3	1375.4
	https://	/catalog/ortopedicheskaya-obuv/	0	2194.2	2359.4	0.9	1356.8	1833.5	4049.0

```
[79]: test('prod2.json')  
#бой без галки
```

```
[79]:
```

				back_start	back_total	unload	dom	front	total
	url	mode							
	https://	/	0	399.2	1453.1	1.0	1557.5	2973.8	3427.7
	https://	/about/	0	192.5	264.8	0.7	624.6	1704.8	1952.3
	https://	/catalog/ortopedicheskaya-obuv/	0	480.5	1048.0	0.7	2301.2	2812.9	3335.1

Вывод: отключение настройки дает уменьшение среднего времени загрузки страницы.



Все пользователи
Просмотры страниц: +0,00 %

29 янв. 2021 г. - 1 февр. 2021 г.
Сравнить с: 22 янв. 2021 г. - 25 янв. 2021 г.



+ Добавить сегмент

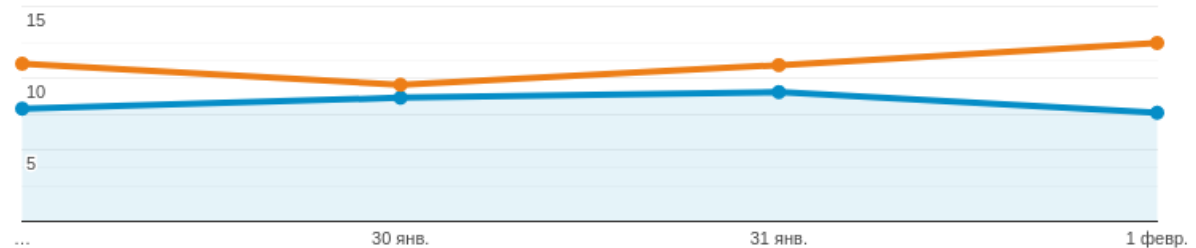
Обзор

Среднее время загрузки страницы (сек.) И Выбор показателя

Час День Неделя Месяц

29.01.2021 - 01.02.2021: ● Среднее время загрузки страницы (сек.)

22.01.2021 - 25.01.2021: ● Среднее время загрузки страницы (сек.)

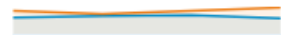


Просмотров в выборке для расчета времени загрузки страницы: 2 247

Среднее время загрузки страницы (сек.)

-24,90 %

8,16 и 10,87



Среднее время переадресации (сек.)

13,50 %

0,10 и 0,08



Среднее время поиска домена (сек.)

-14,81 %

0,03 и 0,03



Среднее время соединения с сервером (сек.)

-40,19 %

0,05 и 0,09



Среднее время ответа сервера (сек.)

-53,90 %

0,94 и 2,05

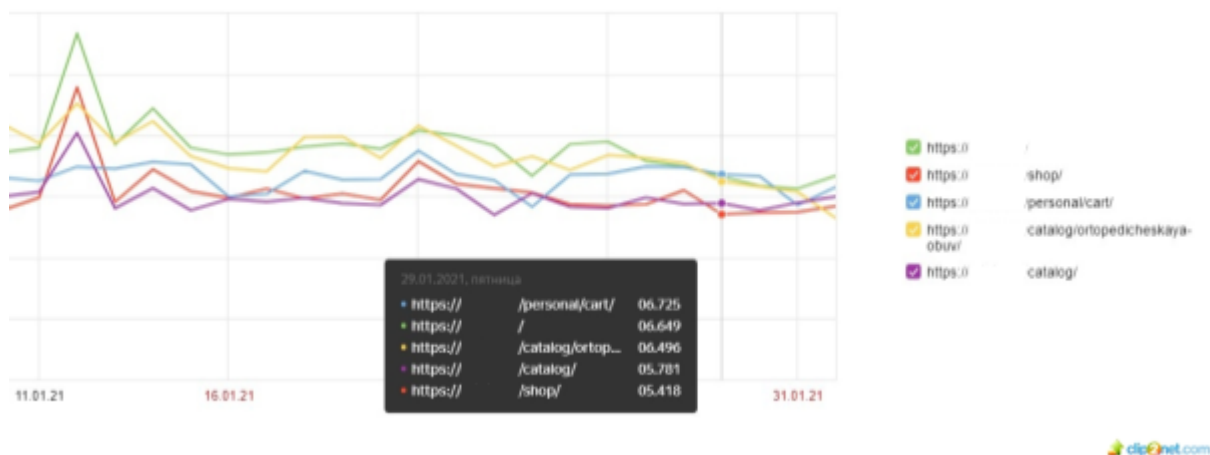


Среднее время загрузки страницы (сек.)

-18,05 %

0,36 и 0,44





Вывод

Все вышеперечисленное говорит в пользу решения - выключить настройку “Перенос js в конец страницы”, за счет чего удастся сократить время обработки этих скриптов.

Чистая скорость загрузки бэк/фронт

Замер чистой производительности страниц каталога: отключили **маркетинговые трекеры**, а затем еще и **html скрипты и скрипты js перенесли в конец** (чат бот, капча на отложенной загрузке) для определения максимально достижимых показателей ответа сервера и фронта.

Из таблицы видно, что сервер при выключенной настройке “перенос js в конец страницы” в совокупности с отложенной загрузкой трекеров и облегченным dom, отвечает менее чем за 0.3с, что является хорошим показателем.

Секция каталога <https://dev04.sitename.ru/catalog/ortopedicheskaya-obuv/>

	back_start		back_total		dom		front		total	
	с	без	с	без	с	с	с	с	с	с
	переносим js	переноса js	переносим js	переноса js	переносим js	без переноса js	переносим js	без переноса js	переносим js	без переноса js
песок, без модификаций	342.7	2862.8	791.0	3092.2	2345.2	1275.1	2893.7	2006.8	3291.9	4910.3
без чатбота и капчи	334.3	2449.5	492.7	2584.6	2014.3	939.7	2205.2	1838.2	2584.1	4333.7
без чатбота и капчи, легкий dom	338.7	338.7	451.0	1861.3	1471.8	222.2	1684.6	1158.6	2069.8	2992.3
без чатбота и капчи и segmentstream	301.2	2391.6	412.0	2489.6	281.0	260.3	288.8	264.5	612.6	2680.1
без чатбота и капчи и segmentstream, легкий dom	310.8	1768.9	402.1	1825.1	221.9	211.8	231.3	212.5	566.4	2005.2

Детальная товара <https://dev04.sitename.ru/product/botinki-berkemann-2831/>

	back_start		back_total		dom		front		total	
	с	без	с	без	с	без	с	без	с	без
	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js
песок, без модификаций	216.4	819.2	309.6	903.8	972.8	401.1	1600.4	1467.7	1878.8	2342.8
без чатбота и капчи	214.1	576.9	255.0	621.5	414.2	390.9	1280.0	1212.4	1537.7	1830.4
без чатбота и капчи, легкий dom	206.6	484.1	243.8	515.8	243.5	244.9	955.9	934.5	1217.9	1458.6
без чатбота и капчи и segmentstream	189.9	585.9	216.8	622.4	151.5	162.9	151.7	163.1	363.7	771.2
без чатбота и капчи и segmentstream, легкий dom	185.8	489.3	217.0	528.2	126.5	131.0	128.9	131.2	330.5	640.4

Умный фильтр: [много условий](#)

	back_start		back_total		dom		front		total	
	с	без	с	без	с	без	с	без	с	без
	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js
песок, без модификаций	395.9	925.0	463.2	986.8	844.0	353.4	1545.2	1589.6	1990.9	2553.8
без чатбота и капчи	415.6	728.2	440.0	757.7	684.5	378.5	1122.4	1194.9	1578.6	1968.7
без чатбота и капчи, легкий dom	384.0	616.8	398.0	623.6	477.3	270.1	926.0	859.8	1345.0	1513.5
без чатбота и капчи и segmentstream	364.7	731.9	399.2	743.2	141.6	146.0	144.2	161.1	526.8	912.0
без чатбота и капчи и segmentstream, легкий dom	387.5	616.9	393.6	623.7	130.1	150.1	143.9	153.3	550.9	788.3

Умный фильтр: [мало условий](#)

	back_start		back_total		dom		front		total	
	с	без	с	без	с	без	с	без	с	без
	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js	переносом js	переноса js
песок, без модификаций	450.6	2758.7	755.5	2864.0	2224.9	980.3	2713.9	1880.2	3223.2	4686.5
без чатбота и капчи	422.9	2279.3	537.6	2387.4	1999.4	761.1	2215.9	1439.6	2684.2	3763.3
без чатбота и капчи, легкий dom	439.8	1726.0	501.5	1810.3	1442.0	285.3	1652.1	1229.3	2134.7	3012.7
без чатбота и капчи и segmentstream	394.5	2279.5	477.9	2359.8	271.8	240.4	273.4	244.7	691.2	2550.1
без чатбота и капчи и segmentstream, легкий dom	404.6	1681.5	454.5	1726.0	201.5	226.1	220.6	227.8	649.0	1930.9

При этом, если вернуться назад и посмотреть на замеры нагруженного рабочего сайта с выключенной настройкой js - ясно, что время сервера изначально в приемлемых величинах до 0.5с в некоторых случаях.

```
[79]: test('prod2.json')
      #бой без галки
```

[79]:	uri	mode	back_start	back_total	unload	dom	front	total	
	https://	/	0	399.2	1453.1	1.0	1557.5	2973.8	3427.7
	https://	/about/	0	192.5	264.8	0.7	624.6	1704.8	1952.3
	https://	/catalog/ortopedicheskaya-obuv/	0	480.5	1048.0	0.7	2301.2	2812.9	3335.1

Вывод: проблема на стороне отрисовки фронта.

Предлагаем отключить настройку js, облегчить dom и перенести трекеры на отложенную загрузку.

Ниже - несколько вариантов замеров на **тестовой среде** в разных комбинациях.

https://dev04._____/

	back_start		back_total		dom		front		total	
	с переносом js	без переноса js	с переносом js	без переноса js	с переносом js	без переноса js	с переносом js	без переноса js	с переносом js	без переноса js
песок, без модификаций	335.6	1609.7	1263.1	2325.6	1345.3	1441.4	2760.7	2665.1	3158.9	4329.9
без чатбота и капчи	315.7	1597.7	867.4	1873.9	1061.5	973.5	2058.2	1926.2	2420.8	3572.7
без чатбота и капчи, легкий dom	280.3	1141.4	830.2	1480.2	927.3	1000.9	1947.5	1863.4	2272.6	3038.2
без чатбота и капчи и segmentstream	235.4	1190.7	412.5	1366.9	204.6	194.2	503.7	1101.3	762.2	2317.1
без чатбота и капчи и segmentstream, легкий dom	234.9	962.1	416.2	1097.9	191.6	171.0	485.1	834.2	742.6	1823.6

https://dev04._____/about/

	back_start		back_total		dom		front		total	
	с переносом js	без переноса js	с переносом js	без переноса js	с переносом js	без переноса js	с переносом js	без переноса js	с переносом js	без переноса js
песок, без модификаций	200.4	451.8	255.5	495.5	482.5	192.3	1312.1	1420.2	1574.0	1916.5
без чатбота и капчи	187.5	305.1	195.2	312.7	185.2	146.8	804.1	851.0	1035.4	1197.9
без чатбота и капчи, легкий dom	191.3	254.4	201.8	259.3	146.2	162.9	781.9	911.1	1024.6	1201.7
без чатбота и капчи и segmentstream	158.6	277.5	163.3	303.8	126.3	147.4	838.2	463.8	1014.8	758.7
без чатбота и капчи и segmentstream, легкий dom	172.9	249.4	177.2	253.5	100.0	143.9	315.4	431.9	505.1	699.2

Замер на тестовой среде при выносе html, чатбота, на отложенную загрузку (~2.6с при выключенной галке js). В сравнении с нагруженной песочницей (2.1с при выключенной настройкой “Перенос js в конец страницы”) скорость загрузки страницы на 15-20 % выше.

```

: test('sand_lite_track.json')
#песок с трекерами и без лишней html, без чатбота, без капчи

:                                     back_start  back_total  unload  dom  front  total
url mode
https://dev04._____/catalog/ortopedicheskaya-obuv/ 29    331.7    470.9    0.9  1667.4  1872.1  2275.3
:                                     31    1867.0    1953.4    0.1  292.8  1217.5  3134.2

```

```

: test('sand_lite_track2.json')
#песок с трекерами и без лишней html, с чатбот, с капчей

:                                     back_start  back_total  unload  dom  front  total
url mode
https://dev04._____/catalog/ortopedicheskaya-obuv/ 29    333.6    570.2    0.5  1355.0  1788.8  2175.1
:                                     31    2225.6    2343.9    0.7  847.4  1535.3  3809.2

```

Резюме

В рамках данного аудита решалась **задача**:

Найти причины медленной работы каталога сайта (включая умный фильтр и детальную страницу товара) и повысить производительность.

Цель - достичь 3с загрузки страницы каталога.

По итогу аудита был выявлен ряд проблем, как на стороне бэка, так и фронта.

Средняя скорость загрузки страницы каталога на 30.01.2020 составляла 9с.

3 из них - отработывал в среднем каталог, остальная часть нагрузки приходилась на фронт.

Выводы и предложения

#	Предложение	Оценка (ч)
Фронтальная часть		
1	Перенос на отложенную загрузку маркетинговых инструментов, в том числе из Segmentstream <ul style="list-style-type: none">- goodmod.ru- k50tracker2.js- citeo.net- flocktory- enpop.min.js (Большой нагрузки не замечено)- fbevents.js (Большой нагрузки не замечено)- vk openapi (Большой нагрузки не замечено)- sherlockcrm	до 4ч
2	Облегчить dom <ul style="list-style-type: none">- поиск- превью страниц- всплывающие окна	20ч
3	Отключение настройки "перенос js в конец страницы"	готово
Серверная часть		
4	Для достижения цели: Выдерживание нагрузки: В день 100 000 пользователей (сейчас 17 – 20 тыс.) Онлайн 1500-2000 (сейчас 200-250) Хитов: 50 в сек (сейчас 10 в секунду)	20-24ч

	<p>Нагрузочное тестирование</p> <ul style="list-style-type: none"> - формирование сценария нагрузочного тестирования по основному конверсионному пути; - написание скрипта по основным пользовательским сценариям; - поиск проблемных мест на рефакторинг (параллельно); - запуск тестирования; - выводы. <p>предлагаем провести замер максимально возможной нагрузки сервера: нагрузочное тестирование на текущей конфигурации для основного конверсионного сценария на сайте, чтобы выявить предел.</p> <p>При подготовке скрипта для тестирования могут быть дополнительно выявлены особо нагруженные запросы и узкие места на сайте.</p> <p>Тестирование будет проводиться инструментом https://locust.io, он позволяет гибко настраивать скрипты и с его помощью удобно проводить аналитику результатов.</p>	
5	<p>Рефакторинг и исследование кеширования</p> <p>Для оптимизации работы функции delivery:</p> <ul style="list-style-type: none"> - гибко настроить кеш, чтобы не было сброса при обновлении товара (3-4ч), - или переписать функцию полностью (завязать на товар из 1С) (20ч). <p>Функция вызова товара в шапке:</p> <ul style="list-style-type: none"> - настроить тегирование, которое сбрасывает весь кеш инфоблока (до 4ч) - отрефакторить функцию (до 15ч) 	20ч на delivery 15ч на шапку