Количество блоков в пуле должно быть ограничено, мы будем отталкиваться от избыточного объёма памяти накопителя (overprovisioning). В среднем overprovisioning составляет 10% от общего объёма памяти [3], половину от него выделим для поддержания пула (5%), что приближённо составляет 100 блоков на plane (подмножество блоков, физически размещённых на одной интегральной схеме), или 1 600 на всё устройство. Имеет смысл равномерно распределять блоки из пула по plane, так как большинство контроллеров NAND поддерживают операцию обратного копирования (copy-back), которая позволяет копировать данные из одной страницы в другую на том же plane [1]. Эта операция читает данные во внутренний регистр данных и записывает из него же, обеспечивая лучшую производительность за счёт отсутствия передачи данных по шине. Если в пуле не остаётся чистых блоков, выбирается наиболее заполненный блок из пула, после чего объединяется с соответствующим блоком данных и очищается.

Необходимое количество памяти рассчитывается исходя из размера одного элемента таблицы и количества записей. Для страничной стратегии достаточно 3Б памяти на элемент таблицы, чтобы хранить номер физической страницы. Количество записей таблицы равно количеству страниц. Для блочной стратегии – 2Б, чтобы хранить номер физического блока. Количество записей равно количеству блоков. Для гибридной стратегии одна таблица идентична блочной. Каждая запись второй таблицы хранит номер логического и физического блоков (2Б + 2Б) и 64 однобайтовых смещения страниц в физическом блоке (64Б). Количество записей второй таблицы, как было показано ранее, равно 1 600. Общий объём памяти двух таблиц: 64КБ + 106КБ = 170КБ.

Таблица 2 – Потребление памяти разных стратегий.

	Страничная	**Блочная**	**Гибридная**
Размер элемента таблицы	3Б	2Б	2Б, 68Б
Количество записей	2 097 152	32 768	32 768, 1 600 (~5%)
Размер таблицы	6МБ	64КБ	170КБ

Из таблицы 2 видно, как и ожидалось, меньше всего памяти потребляет блочная стратегия. Также можно отметить, что гибридная стратегия обеспечивает более эффективные записи размера страницы относительно блочной стратегии при гораздо меньшем потреблении памяти относительно страничной стратегии.

К недостаткам можно отнести бóльшую задержку доступа к памяти, так как надо произвести поиск в двух таблицах в случае если log-block и data-block ещё не были объединены, и потребность в дополнительной памяти накопителя для поддержания пула блоков.

Для более детальной оценки рассмотренных стратегий в дальнейшем планируется реализовать их в системе моделирования флэш-памяти “MQSim”. Иначе сложно оценить такие факторы, как задержка выполнения разных операций, сборка мусора, выравнивание износа. Даже те объёмы памяти, которые были рассчитаны в статье, в большинстве случаев не могут храниться в динамической памяти полностью, поэтому необходимо решать задачу кэширования таблиц.

Список использованных источников:

1. Design Tradeoffs for SSD Performance / Nitin Agrawal [et al] // USENIX Technical Conference – June 2008.
2. Reconfigurable FTL (Flash Translation Layer) Architecture for NAND Flash-Based Application / Chanik Park [et al] // ACM Transactions on Embedded Computing Systems – July 2008 – Vol. 7 – No. 4 – Article 38.
3. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives / Yu Cai [et al] // IEEE – September 2017.

ПРИЛОЖЕНИЕ ДЛЯ УВЕЛИЧЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ТРУДА НА ОСНОВЕ ТЕХНИКИ POMODORO

Чачура Р.В., Соболевский С.Е.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Жвакина А.В. – к.т.н.,доцент

Во время разработки произведены исследования в области информационных технологий, в частности инструментов для разработки серверной и пользовательской части приложения по улучшению умственной производительности человека.

Increaser − это приложение для тех, кому для выполнения работы необходима умственная концентрация: студенты, инженеры, программисты, математики и другие технические работники. Эти люди хотят делать больше глубокой, целенаправленной работы и меньше мелкой, полной

отвлекающих факторов. Таким образом, они найдут время, чтобы заниматься тем, что имеет значение в их жизни. Отвлечение является главной проблемой, когда мы говорим о Increaser. Из-за отвлекающих факторов мы не можем выполнять работу качественно, испытываем стресс и не находим времени на важные для нас вещи.

В представляемой работе реализована серверная часть для приложения по улучшению производительности с использованием AWS сервисов. Основной API для пользовательской части написан на языке программирования NodeJS. Пользовательская часть написана с использованием React и Styled-Components для UI, Redux для управления состоянием приложения и Redux-Saga для асинхронных операций.

В процессе реализации серверной части приложения решен широкий спектр задач, начиная с описания инфраструктуры через код и заканчивая созданием запросов к базе данных.

Использованы сервисы AWS, такие как ElasticBeanstalk, DynamoDB, S3, Route5S, Certificate Manager, CloudFront, CodePipeline, ECR. Для описания инфраструктуры использовалась технология Terraform. Рассмотрены лучшие практики для организации Terraform модулей и их зависимостей. Написана конфигурация для CI/CD сервисов с помощью CodePipeline. Сборка сервисов происходит в Docker контейнерах. Для хранения пользовательских данных используется NoSQL база данных DynamoDB, для хранение редко извлекаемых данных − Amazon S3.

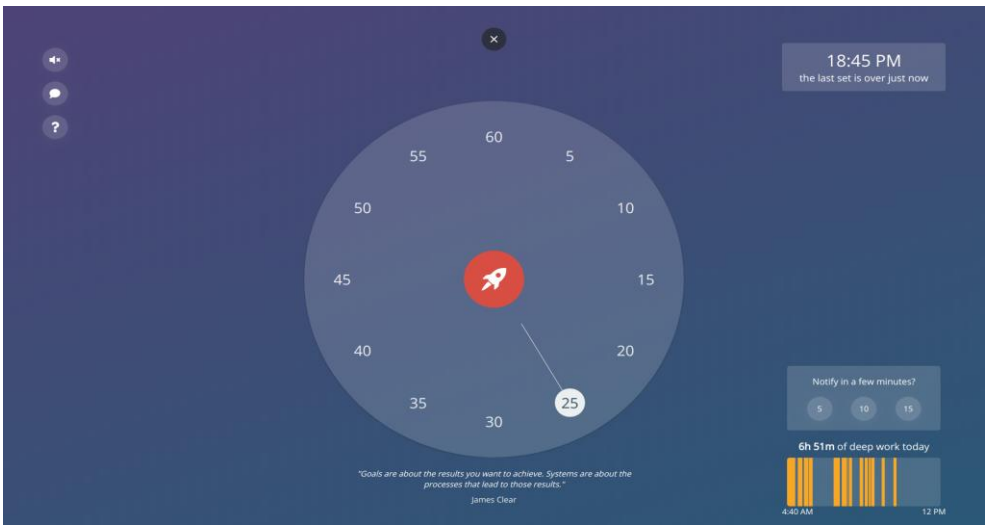
Во время разработки приложения реализовано второе приложение pomodoro.increaser.org благодаря тому, что UI вынесены в отдельные репозитории и NPM модули. Для написания серверной части использовался аналогичный стек. Единственное отличие заключается в использовании GraphQL для обмена данными между клиентом и сервером.

Процесс использования приложения можно представить в цикле из трех шагов. Выбор промежутка времени для работы, выполнение задачи без отвлечений, перерыв. В конце дня можно провести анализ производительности.

Приложения доступны по адресам increaser.org и pomodoro.increaser.org.

Аналитика связанная с количеством активных пользователей

Важным преимуществом приложения является качественное UX и доступность приложения в браузере. Работа над приложением продолжается, мы внимательно прислушиваемся к отзывам пользователей и адаптируем функционал с учетом их пожеланий.



Интерфейс главной страницы pomodoro.increaser.org

Список использованных источников:

1. Kief Morris. Infrastructure as Code: Managing Servers in the Cloud/ K. Morris // Physica Status Solidi. A. – 2016. – Vol. 2014, № 12. – P. 2765–2771.
2. AWS Whitepapers. Architecting for the AWS Cloud: Best Practices / AWS Whitepapers. – 5. Aufl. – Canda, 2016. – 232 S.
3. Dan Sullivan.NoSQL for Mere Mortals / D. Sullivan – UK, 2015. – 234-552 S.