

Дипломные работы

Темы / направления

Медицина

3D Slicer

Технологии: Python / OpenCV / Qt / OpenFOAM / CMake / C++ ...

Пригодность: бакалавр / магистр

Во взаимодействии с клинической больницей РАН спланирован ряд задач в проекте "[Система поддержки составления плана операции и диагностики ЛОР-заболеваний](#)". Набор задач потребует достаточно разноплановые знания - от разработки плагинов и анализа изображений (как методами вычислительной геометрии, так и ML) до обеспечения необходимого уровня производительности

1. Поиск и выделение:
 - Поиск и выделение носовой перегородки
 - Полипов, аденоидов и т.п.
 - Решётчатых артерий
2. Определение:
 - Искривления носовой перегородки
 - Узкого места и расчёт площади/объёма дыхательных пазух
 - Утолщения слизистой оболочки
3. Хватает ли дыхания на основе отношения индекса массы тела и площади дыхательного просвета? (показано ли хирургическое лечение храпа)
4. Расчёт маршрута проведения инфундибулотомии
5. Учёт газодинамических эффектов и моделирование дыхательного процесса

На текущий момент в рамках 2 дипломных работ апробированы некоторые методы выделения аномалий и моделирования дыхания. Требуется доисследование с доведением до практического применения + решение остальных исследовательско-практических задач

Параллельное программирование / High performance computing

Contention-aware synchronization objects

Технологии: C++ / C / Linux / eBPF / ...

Пригодность: бакалавр / магистр

Исследования и доработка примитивов синхронизации glibc/ядра ОС для переключения стратегий синхронизации исходя из анализа текущего профиля исполнения приложения и загруженности ОС (contention level). Конечная цель - PR с адаптивной реализацией базовых примитивов в тесном взаимодействии с планировщиком ОС

- [Описание проблемы на примере Postgres](#)
- [Базовая идея определения contention level](#)

YDB

Технологии: C++ / YDB / etcd / CMake / ...

Пригодность: бакалавр / магистр / НИР

Yandex в качестве дипломных и практических работ предоставляет [ряд задач](#) по YDB. Команда [Олега Доронина](#) (ведущий практик по || программированию) из Yandex поддерживает работу с дипломником и обеспечивает экспертизу в указанной области

- Разработка новой функциональности и улучшений в SDK на языках: [Go Java Python C++](#)
- [Разработка NoSQL-адаптера к СУБД ydb на основе протокола etcd: необходимо реализовать протокол etcd на основе YDB \(активно используется GRPC\), сохранив производительность и иные заданные параметры системы](#)
- [Интеграция R7 office и YDB](#)
- [Поддержка новых федеративных источников на Go \(как PostgreSQL, MySQL, MS и т.д.\)](#)
- **Кеширование прочитанных данных из S3.** В YDB федеративных запросах поддерживается чтение из внешнего источника [S3](#). Чтение данных из такого источника может быть медленным, а также данные в нем могут храниться в неструктурированном виде json, csv, xml, raw и даже в сжатом представлении. Предлагается в качестве НИР разработать кеширование на диске прочитанных данных в уже подготовленном и сжатом виде для ускорения и минимизации накладных расходов при повторном чтении таких данных.
- **Автоскейлинг потоковых запросов.** Внутри сервиса [Yandex Query](#) поддерживаются потоковые запросы, обработка в которых является бесконечной. Источниками в таких запросах являются бесконечные очереди сообщений. Для обеспечения отказоустойчивости и гарантий доставки данных используются так называемые [чекпоинты](#). Но при этом в процессе работы таких запросов может увеличиться объем поступаемых данных и в этом случае нужно иметь механизмы для оценки и масштабирования запросов (в том числе и состояний которые сохраняются при прохождении чекпоинтов)
- **Планировщик для запуска запросов в YQ.** Некоторые системы поддерживаются запуск запросов по расписанию (для [cron](#)). Примером такой системы является [snowflake](#). В работе предлагается провести исследование существующих систем которые предлагают похожие решение. На основе этого анализа предложить варианты решения для Yandex Query и разработать прототип для предложенного решения.
- **Продвинутый планировщик который учитывает локальный плейсинг с учетом доступных ресурсов для потоковых запросов.** Текущая реализация алгоритма планирования либо умеет явно заселять запрос на один хост или же заселять с учетом доступной памяти. Такие стратегии не всегда эффективны с точки зрения использования ресурсов. Предлагается учитывать структуру графов чтобы минимизировать число пересылок и учитывать другие потребляемые ресурсы при планировании CPU/Mem/Net

Сети

NetAnim

Технологии: Qt / C++ / CMake

Пригодность: бакалавр

Доработка [визуализатора](#) сетевой симуляции [NetAnim](#):

- Отображение загруженности сетевых очередей, буферов и т.п.
- Отображение узлов на карте при известном географическом положении
- Анимация объёма передаваемых данных
- Возможность анимации в реальном времени с управлением через python-консоль
- Отображение направленности антенн

Средство развивается мало с 2017 года, но применяется в связке с NS-3. Хорошая практика работы со стекком технологий со средним порогом вхождения

На уровне идей (конференции...)

1. Оптимизация выделения памяти в [jemalloc](#) и освобождения при делании сего из разных потоков (C/C++)
2. Продолжение [проекта Thread Sanitizer](#) (применение [Google TSAN](#) через [LLVM](#)) для [OpenJDK](#):
 - Изучение проблем приостановки работ в 2020 году
 - Доведение до готовности к слиянию в основную ветку
3. Доработка [Google Thread Sanitizer](#), в части:
 - Уменьшение числа false/positive при поиске data/race
 - [Поддержка явных барьеров памяти](#)
 - [Добавление fuzzing потоков](#) (сейчас есть только на данных)
4. Fuzzing:
 - Участие в разработке [centipede](#) в части распределённости и привнесения учёта многопоточности в анализ трасс

Текущие темы

Магистранты

1. Шишкин А., *“Разработка алгоритма адаптивного управления стратегиями синхронизации в зависимости от уровня конкуренции за ресурсы”*, ЛЭТИ 2026

Бакалавры

Защищённые темы

Магистры

1. Кашин Г., *“Разработка алгоритма адаптивного управления стратегиями синхронизации в зависимости от уровня конкуренции за ресурсы”*, ИТМО 2025

2. Талашенко П. *“Разработка NoSQL-адаптера к СУБД ydb на основе протокола etcd”*, ЛЭТИ 2024
3. Самсонов П. *“Система поддержки составления плана операции и диагностики ЛОР-заболеваний”*, ИТМО 2024
4. Егорычев А. *“Система поддержки составления плана операции и диагностики ЛОР-заболеваний”*, ИТМО 2024
5. Шахов А. *“Разработка стратегии планирования вычислительных потоков с целью проверки линейризуемости lock-free алгоритмов”*, ЛЭТИ 2021
6. Надежин Н. *“Разработка комплекса скоростной видеосъемки с возможностью автосопровождения летящих объектов”*, 2020
7. Ёров С. *“Доработка алгоритмов Google Thread Sanitizer”*, АУ 2018
8. Доронин О. *“Автоматическое fuzzy-планирование потоков с помощью relacy для обнаружения ошибок в многопоточном коде”*, АУ 2018
9. Карулин Н. *“Исследование и разработка методов обеспечения заданной производительности системы анализа последовательностей генома”*, ЛЭТИ 2017
10. Яцык А. *“Разработка протокола формирования и передачи вектора состояния ОЭС ТИК-М.”*, ИТМО 2017
11. Галимуллин М. *“Разработка адаптивной стратегии синхронизации потоков в конкурентных структурах данных, основанных на flat-combining”*, ЛЭТИ 2016
12. Рапоткин Н. *“Разработка стратегий flat-combining для конкурентных структур данных на примере библиотеки libcds»”*, ЛЭТИ 2015 (+ Балтрашевич)
13. Леснова О. *“Разработка методов балансировки нагрузки для платформы моделирования сетей NS-3”*, ЛЭТИ 2013 (+ Балтрашевич)
14. Алексеева А. *“Разработка алгоритма маршрутизации беспроводных Mesh-сетей в условиях ограничения на энергопотребление узлов”*, ЛЭТИ 2013 (+ Балтрашевич)
15. Александрова С. *“Разработка средства моделирования пространственной структуры белковых молекул”*, АУ 2010

Бакалавры

1. Манцева Т., *“Разработка алгоритма адаптивного управления стратегиями синхронизации в зависимости от уровня конкуренции за ресурсы”*, ЛЭТИ 2025
2. Мосин К., *“Разработка приложения информационного взаимодействия с радиотехническими комплексами”*, ЛЭТИ 2025
3. Никитин Д. *“Разработка системы развёртывания программных продуктов в рамках конвейера системы непрерывной интеграции”*, ЛЭТИ 2024
4. Швец А. *“Разработка системы управления наземными измерительными средствами”*, ЛЭТИ 2021
5. Цикалюк А. *“Разработка инфраструктуры развёртывания программных артефактов в системе непрерывной интеграции”*, СПбГПУ 2021
6. Шохин Е. *“Интерактивная карта университета с отображением расписания”*, ЛЭТИ 2016
7. Королёв Ю. *“Разработка подсистемы визуализации созвездия навигационных космических аппаратов”*, ЛЭТИ 2009 (+ Кафтасьев)

Специалисты

1. Лагутчев Н. *“Разработка системы непрерывной интеграции для целевых платформ защищённых ОС”*, РГГМУ 2021
2. Шмыгин Е. *“Разработка конвейерной системы поддержки жизненного цикла сборки”*

программных продуктов”, РГГМУ 2021

3. Кузнецов Н. “Проектирование безопасности оптико-электронной станции”, РГГМУ 2020

Архив

1. Доработка алгоритмов детекции Data race в valgrind [helgrind](#) и [drd](#) по существующим ошибкам / запросам
2. Алгоритмы, требующие реализации и доработки в [libcdfs](#), обычно есть что улучшить и ускорить по сравнению с реализацией из статей
3. [Основанное на анализе кода fuzzy-планирование потоков с применением lincheck для обнаружения ошибок в многопоточном коде](#)
4. Пересмотр текущих чистых lock-free алгоритмов в сторону Hardware Transactional Memory
5. Итераторы в lock-free контейнерах (Multi Array): в структуре данных “Feldman's Multi Array” в [libcdfs](#) добавить возможность работы с ключами переменной длины через список коллизий в узлах. Тип lock-free списка должен задаваться извне (в Traits) - сейчас есть три реализации - MichaelList, LazyList, IterableList
6. [Доработки hpx](#)
7. Сравнительный анализ:
 - Производительность STM для Java / Haskell / Closure
 - Производительность HTM в OpenJDK
 - Производительность [HTM](#)
 - Производительность scalable аллокаторов (C++)
8. *[В проработке]* Доработка применения HTM в OpenJDK ([вводная статья](#) и [от того же автора](#), небольшой [вводный доклад](#) по TM в принципе). Сама реализация была сделана в 1.8 по [этой задаче](#)
9. Решение [задач](#) по расширению функциональности моделирования сетевых протоколов в симуляторе [NS-3](#) (C++)
10. Сравнительный анализ:
 - RCU примитивы по отношению к RWLock для различных структур данных (тут придётся немного пару структур может на rcu перевести, но это не сложно) (C++)

From:

<http://wiki.osll.ru/> - **Open Source & Linux Lab**

Permanent link:

<http://wiki.osll.ru/doku.php/etc:users:kel:diplomas?rev=1751016361>

Last update: **2025/06/27 12:26**

