Дипломные работы

Темы / направления

1. Доработка визуализатора сетевой симуляции NetAnim по пожеланиям разработчиков *(C++/Qt, возможно в ходе исследования может быть доработано иное средство)*:
    - 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
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
5. Система поддержки составления плана операции и диагностики ЛОР-заболеваний (актуальные задачи на странице проекта)

На уровне идей (конференции...)

1. Оптимизация выдедения памяти в jemalloc и освобождения при делании сего из разных потоков *(C/C++)*

Текущие темы

Магистранты

Бакалавры

Защищённые темы

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

Магистры

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

Бакалавры

1. Швец А. *“Разработка системы управления наземными измерительными средствами”*, ЛЭТИ 2021
2. Цикалюк А. *“Разработка инфраструктуры развёртывания программных артефактов в системе непрерывной интеграции”*, СПбГПУ 2021
3. Шохин Е. *“Интерактивная карта университета с отображением расписания”*, ЛЭТИ 2016
4. Королёв Ю. *“Разработка подсистемы визуализации созвездия навигационных космических аппаратов”*, ЛЭТИ 2009 (+ Кафтасьев)

Архив

1. Доработка алгоритмов детекции Data race в valgrind helgrind и drd по существующим ошибкам / запросам
2. Алгоритмы, требующие реализации и доработки в libcds, 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 libcds 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 докладпо 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++)*