ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

Курс: архитектура суперкомпьютеров

Лекция 2

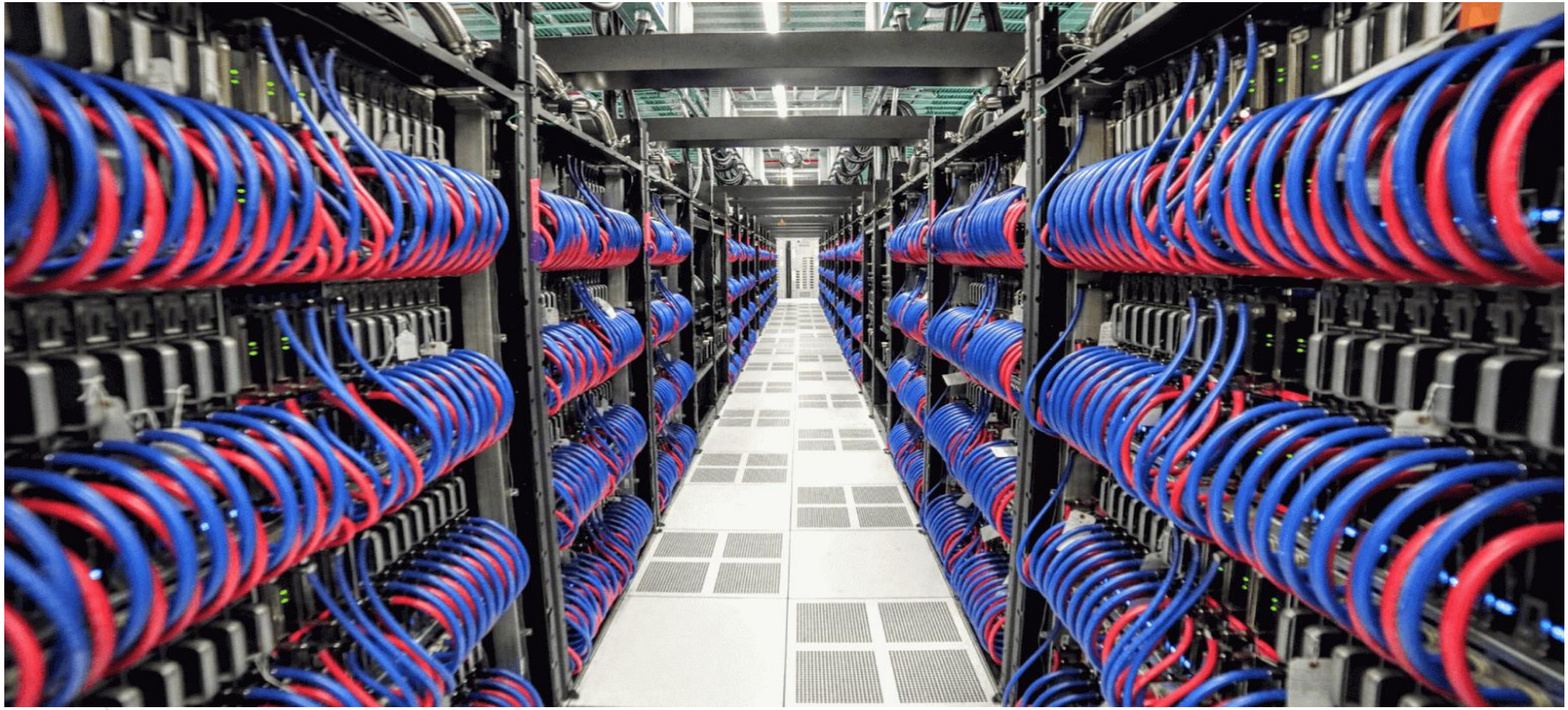
Памяти-центрические вычислительные системы

(обратные задачи алгоритмизации вычислительных процессов)

В. С. Заборовский

17 сентября
2024

СК: **2 ЭКСАФЛОПСА ИЛИ 1 600 000 000 000 000 000 ОПЕРАЦИЙ В СЕКУНДУ, 21 МВТ ПОТРЕБЛЕНИЯ ЭНЕРГИИ И ЭМУЛЯЦИЯ ОБЛАСТИ ЧЕЛОВЕЧЕСКОГО МОЗГА В ТЕЧЕНИЕ 1 СЕКУНДЫ.**



«Машинное обучение» - супер задача для суперкомпьютеров

прямая задача: «алгоритм -> данные»

обратная задача «данные -> алгоритм»

Искусственные Нейронные Сети - это вариант формальной системы, способной к «обучению» для реализации «рекурсии Геделя» с целью «конструирования аксиоматики» для описания решения выбранного класса «обратных задач»

смысл результата вычисления, согласно теоремам о неполноте Геделя, осмысление результаты вычисления МТ требует «внешнего наблюдателя». ИНС «вычисляет» числа-знаки (см. Г. Вейль), но не может их объяснить их смыслы (4 - «красный цвет»)

Естественный интеллект:
{ действия , основанные на их понимании законов природы и стремления объяснить смысл производимых действия с учетом контекста событий или гипотез }

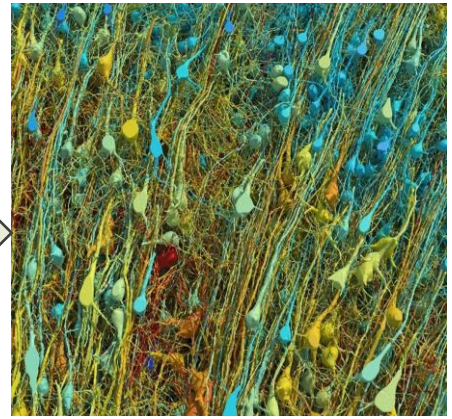
граница формальной (частичной) рациональности

ИИ - виртуальная реальность:
{ действия направленные на имитацию (моделирование) процессов на основе формальных (логически объяснимых) правил /алгоритмов }

Компьютер как машина-трансформер: «чего» во «что» и «как» ?

if we built a big enough computer, then it could compute anything we wanted it to. Is this true?
Ричард Фейнман

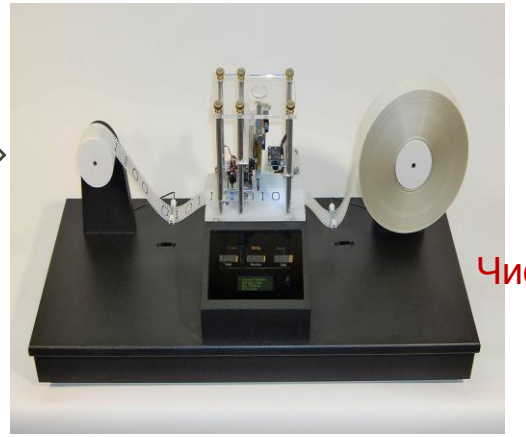
мозг



изображение одного кубического миллиметра мозга с точностью до каждой клетки

Машина Тьюринга

Лента-программа

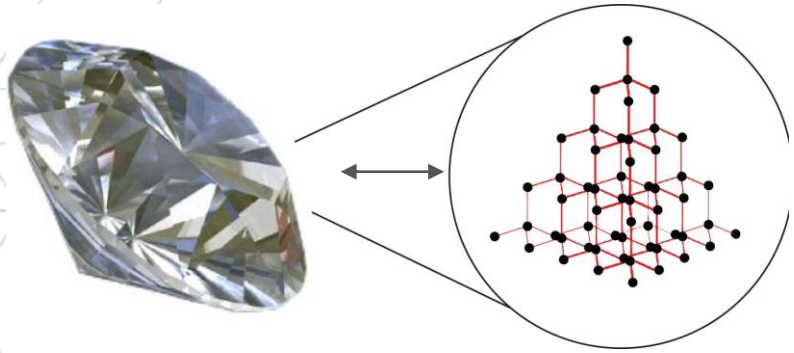


Число x_{y^2}

«три гриба»

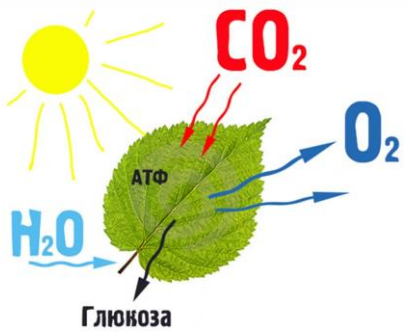
Суперкомпьютер И. Маска будет применяться для обучения чат-бота Grok из 100 000 микрочипов NVIDIA H100.

Зеттафлопсный суперкомпьютер и ЦОД с тремя ядерными реакторами от Oracle будет использоваться для обучения систем ИИ



атомарная структура трансформируется в «твердое тело» - алмаз

Фото-синтез углекислый газ и воду трансформирует в древесину



Деревон а 80% состоит из воды и воздуха

Проблемы КН: «теорема Гёделя» —арифметика неполна.

Computer science, in a sense, existed before computers themselves appeared in the mid-20th century. There was a lot of noise about the question –

what could computers calculate, in principle?

Ричард Фейнман

Введение в вычислительную физику.

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!} + \ldots = 1 + \sum_{n=1}^{\infty} \frac{1}{n!}.$$

Как выбрать n чтобы расчет выполнить с заданной точностью ?

матричная экспонента - определяется как сумма ряд

$$e^{t\mathbf{A}} = \mathbf{E} + \sum_{k=1}^{\infty} \frac{t^k \mathbf{A}^k}{k!}$$

При К > 30 график функции начинает вести себя хаотично и не имеет с экспонентой ничего общего.
Причина данного эффекта в накоплении погрешностей округления
Вводится понятие: вычислительной устойчивости

Архитектуры аппаратного и программного обеспечения СК

- **Проблема**: «точность-производительность»
- «ПРИКЛАДНАЯ» производительность и энерго-вычислительная эффективность СК при решении [задач как моделирования физических (сложных) систем …. в реальном масштабе времени .

  Цель : формализация проблем, преодоление которых позволит решить задачу повышения «прикладной производительности» и точности СК в условиях физических ограничений и требований «реального времени» .
- **Гипотеза**: Решение проблем лежит в переходе к новой архитектуре СК на основе использования:
  - декларативной программной парадигмы - «вычисления в памяти»
  - технологий описания прикладной задачи с учетом ограничений, следующих из требований «реального времени»
  - возможностей динамической реконфигурации циклов вычислений в «памяти», используемой как среда исполнения прикладных алгоритмов

  Итого: что требуется : 1) как-то менять архитектуру вычислительного поля СК «под алгоритм» и «поток входных данных» прикладной программы,

  Методология : разработка языков программирования и аппаратных компонент для реализации выдвинутой гипотезы.
  - **Ожидаемые** результаты: полиномиальный рост прикладной производительности СК
  - **Значимость исследования**: применение новых архитектур для решения задач «реального времени».

    Этапы выполнения: формализация гипотезы, патентование предлагаемых решений, апробация методов и публикация результатов.

Проблемы которые надо разрешить

- 1. **Многокритериальность:** результаты выполнения проекта могут не соответствовать правилу и критерию Парето (“20% вложений приводят к 80% результата”) , что затрудняет планирование и прогнозирование необходимых ресурсов, но позволяет использовать**лексикографический порядок** оптимизации целевых функций (в том порядке какой из критериев в данный момент важнее других для решения прикладной задачи)

2. «не нужно делать все, а необходимо делать только то, что дает результат” при этом необходимое оборудование и квалифицированные кадры всегда будут в дефиците.

- 3. **Нарушение сроков**: ошибки проектирования, сложности проведения экспериментов и других факторов вызывают задержками, что нарушает график работы, принятых в ТЗ.

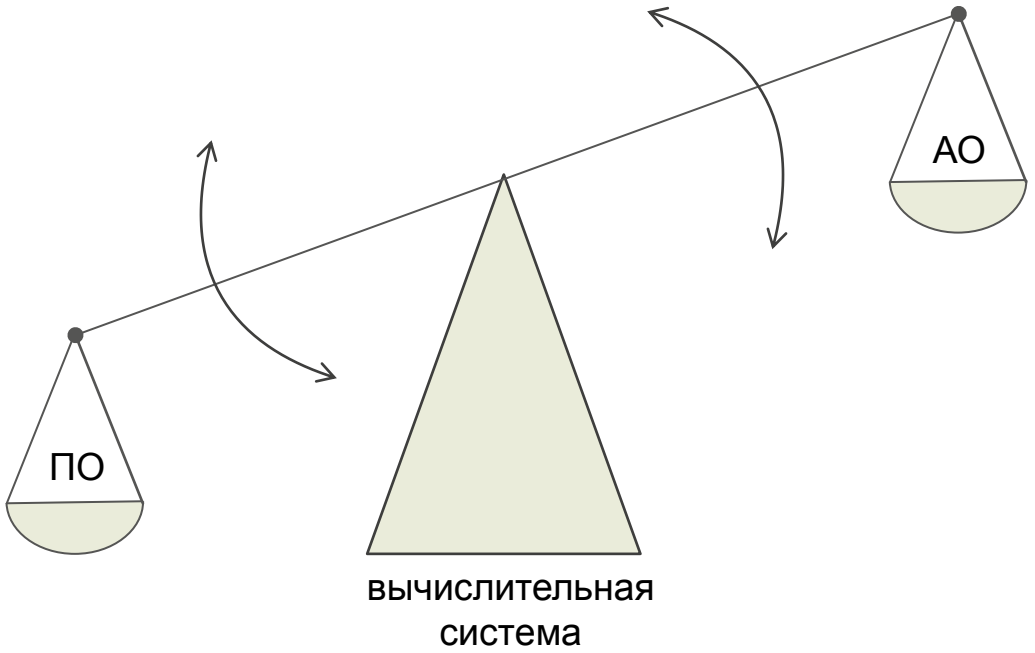
4 **Необходимость корректировки задания** : появление новых данных и уточнение потребности создает дополнительные сложности для управления..

Роль информационных и физико-энергетических воздействий

1. **Информационное воздействие**:
   - Сбор и анализ данных для принятия взвешенных решений.
   - Передача знаний и коммуникация позволяет обеспечить координацию действий и согласование планов.
   - Мониторинг и оценка: Постоянный мониторинг выполнения проекта с использованием различных метрик позволяет выявлять отклонения от плана и корректировать действия.

2. физико-энергетические **воздействия**:
   - Планирование и организация управляющех воздействий определения критериев успеха.
   - Контроль и корректировка результатов.
   - Мотивация и руководство включает координацию усилий направленных на выполнения задач, установление приоритетов и разрешение конфликтов.

Are there some questions that computer could never answer for us,
however beautifully made it might be?
Ричард Фейнман



There was a lot of noise about the question - what could computers calculate, in principle?

ИТАК

Computer science differs from physics in that it is no actually a science. It does not study natural objects.

Ричард Фейнман

Развитие компьютерных наук шло в двух направлениях

- Первое — parallel computing. Многопоточность как форма параллельного вычисления нужна для выполнения одних и тех же операций над разными блоками (частями) данных) : SISD, SIMD, MIMD, MISD.

- Второе — concurrent computing. Это одновременное выполнение множества разных операций. Например, многопоточный сервер СУБД, который одновременно принимает запросы **от разных пользователей**, строит планы их выполнения, производит операции ввода-вывода, отдает результаты запросы клиентам, обновляет статистику и т.д. Многопоточность нужна для обеспечения действительно параллельного выполнения различных операций. Формально, обеспечивать concurrency можно даже и на одном потоке , это режим т.н. квази-параллелизма.

Модельная формализация задачи (1)

Дано: Есть N «вычислимых» объектов, т.н. реальных акторов, которые соответствуют множеству отдельных процессов (потоков данных) в «физической» модели объекта в пространстве его состояний. «Носителем» акторов (то, где они «живут» и «размножаются») является поле вычислительных процессов компьютерной системы. Формально, акторных моделей у реального физического объекта может быть «счетное множество». Все N акторов «конкурируют» естественно друг с другом за вычислительные ресурсы и данные.

Нас интересует такие модели, в которой акторы способны «жить» то есть менять свое внутреннее состояние в реальном времени, другими словами, со скоростью моделируемых с их помощью физических процессов.

Конструктивная модель 1 (простая функциональная изоляция): образовать M мета-акторов (логических или виртуальных авторов), каждый из которых владеет «не разделяемыми ни с кем собственными» данными и вычислительными ресурсами. При этом ни один из таких мета-акторов не имеет доступа к данным других акторов.

Модельная формализация задачи (2)

Ситуация. Актору X для «выживания» потребовалась какая-то информация, которая есть у актора Y или если актор Y «хочет», чтобы актор Z «обновил» какие-то данные у себя.

Другими словами, акторы должны взаимодействовать между собой. Каким образом?

Конструктивная модель (2) (передача сообщений – локальная модель «ответственности»): У каждого актора есть только «локальная ответственность» в удовлетворении своих потребностей. Обычно она реализуется через асинхронный механизм сообщений – переданных и принятых. Эти сообщения образуют очередь и… хранятся в памяти. При этом, если сообщения переносят копию данных, а не только ссылку на исходные данные находящихся в разделяемой памяти вычислительной системы, то такая модель акторов дополняется функционалом «распределенности».

Считается, что актор Y свои «потребности» выражает в форме сообщений, которые образуют новую сущность - очередь сообщений и «локальное время». Считается, что передав сообщение, актор «удовлетворил» свои «локальные потребности». Для актора Y и не важно как актор Z обрабатывает сообщения после их получения, формально эти сообщения могут быть актором Z отправлены на другой узел «акторной» сети. Сообщение суть новая «не физическая» сущность, поэтому эта сущность «информационно» самодостаточна. С помощью сообщений физические потоки и их содержимое могут быть «сериализованы», то есть могут храниться в памяти компьютера до момента «обработки». **В диссертации предложена модель, в которой взаимодействие акторов происходит в РВ.**

https://en.wikipedia.org/wiki/Actor_model_theory. в языке Erlang каждый процесс может считаться актором.

Базовые понятия : Актор —агрегированная сущность, на которую можно «навесить» различные требования

Формальная модель акторов не «говорит» о том, как именно эта сущность должна быть реализована. Так:

актором может быть отдельный объект, который «связывает» один рабочий контекст задачи с другим. Могут быть реализации, в которых все акторы работают на контексте одного потока - одной единственной нити (thread) .

Актор, работающий в РВ, может быть «программно-аппаратной сущностью» (так, например, сделан актор в WhatsApp)

Классический пример модели акторов — программа «пинг-понг» на языках Erlang или Scala

*подозреваю, что пришествие настоящих **многоядерных CPU** сделает программирование параллельных систем с использованием **традиционных мьютексов и разделяемых структур данных** сложным до невозможности, и что именно обмен сообщениями станет **доминирующим способом разработки параллельных систем**.*

Джо Армстронг (создателя языка Erlang

Преимущества акторного подхода

- **простота** разработки - асинхронного обмена сообщениями сильно упрощает работу с парадигмой concurrent computing;
- **масштабирование** - позволяет создавать огромное количество акторов, каждый из которых отвечает за свою частную задачу. Сочетание принципа shared nothing и асинхронный обмен сообщениями позволяет строить распределенные приложения, горизонтально масштабируясь по мере возникновения потребностей;
- **Отказоустойчивость** -сбой одного актора может может контролироваться другими акторами, которые в состоянии предпринимать действия для восстановления ситуации (например, реализуя механизм супервизоров).

Можно показать, что вопросы отказоустойчивости и безопасности вычислительных систем, например, в управляемых средах, эффективно разрешаются при использовании акторных моделей

Таким образом разрешается **парадокс программных технологий**: чем сложнее система и язык программирования, тем проще использовать систему в повседневной работе. (пример программирование требований производственных технологий)

Модель акторов уместно использовать не всегда, хотя в прикладном смысле (например, требования РВ) за асинхронным обменом сообщений между независимыми сущностями, такими как акторы, будущее программных технологий.

Теория (1)

- Понятие **вычислимости** традиционно определялось через **машины Тьюринга**. У такой машины есть состояние – совокупность значений всех ячеек ленты. Ячейки ленты – суть память, перемещение головки машины вдоль ячеек ленты – суть «внешние шаги» вычислений.
- Итак, процесс **вычислений представляет собой последовательность** шагов машины Тьюринга, каждый из которых меняет ее состояние (состояние «**памяти машины», куда записываются данные**). Каждый шаг машины – выполнение лишь одного **неделимого действия** (операции). Традиционная моделью вычислений – выполнение одного шага ленты - одной команды.
- Такая модель вычислений нуждается в введении нового фундаментального понятия **глобального времени**, которое аналогично понятию **континуума времени** в физических науках. Суть в том, что в один и тот же момент «глобального» времени над состояниями как классического физического объекта, так и вычислительного процесса, может осуществляться **одна и только одна «атомарная операция»**.
- Это свойство **глобального времени** широко используется для доказательства свойств различных протоколов синхронизации, например в случае **многопоточного программирования на однопроцессорных машинах**
- **См.** https://www2.cs.arizona.edu/~greg/mpdbook/lectures/
  - sequential program -- process
  - concurrent program processes + communication + synchronization
  - kinds: multithreaded, parallel, distributed

Теория (2)

- Для многопроцессорных компьютеров или распределенных вычислительных систем –кластеров, требование «глобального времени», в общем случае, неприемлемо.
- Уже много лет существует необходимость обобщить понятие «вычислимости» на случай параллельных вычислений.
- Одним из первых вариантов таких обобщений (1973 г.) стала **модель акторов**, которая переносится на многопроцессорные МТ, а их ПО дополнено функциями искусственного интеллекта.
  - В статье 1973 года Carl Hewitt, Peter Bishop и Richard Steiger A Universal Modular Actor Formalism For Artificial Intelligent. показали, что многие классы приложений являются частным случаем модели акторов.

Теория (3)

Итак,

1. Актор – это абстракция , которая характеризует вычислительный процесс, так, актор в ответ на получаемое сообщение может отправить конечное число сообщений другим акторам,
2. В программе можно создать можно конечное число акторов, которые после приема сообщений могут менять свое «поведение» (еще одна новая абстракция).

Комментарии:

- Разница между классической МТ и моделью актора аналогична разнице между телефоном и отправлением сообщения по почте.
- В первом случае происходят вычисления на основе глобального времени, то есть в этом случае идет конкуренция в РВ за доступ к общему разделяемому ресурсу.
- В случае же с почтовым отправлением отправитель письма (актор) просто посылает письмо адресату без каких-либо задержек – реальном времени, так как у актора нет необходимости согласовывать свои действия с другими отправителями в рамках концепции «глобального времени», но при этом актору неизвестно, когда получатель прочтет и ответит на его сообщение

Теория (4)

- События в модели акторов образуют частично упорядоченное множество.
- Аксиоматика этого множества известная как Ordering Laws, была описана в 1977 г. Carl Hewitt и Henry Baker в статье Actors and Continous Functionals.
- Аксиом в этом описании довольно много, чтобы обосновать почему **модель акторов** годится для использования на практике, например, доказано, что в любой момент времени количество сообщений, адресованных одному получателю, конечно.
- В диссертации Gul A. Agha Actors: A Model Of Concurrent Computations in Distributed Systems описывается синтаксис **минимального языка программирования**, поддерживающего акторы, а также набор типовых задач и приемов их решения.

Должна быть обоснована структура и синтаксис языка описания акторов, которая учитывает не только временные, но и **показатели сложности программного обеспечения**. (см. Event-Based Programming without Inversion Control , где рассматривается асинхронный вызов актором соответствующего «метода» и возврат к вычислениям, когда будет получен ответ. Этот механизм всего лишь одна из возможных реализаций event-based programming, которая, однако, не реализует всю акторную модель в полном объеме и не учитывает требования сложности ПО)

Теория (5)

- Общие законы **акторных вычислений** основаны на аксиоме вычислимости **(реализуемости) выбранного** целевого комплекса прикладных задач **в модели глобальном времени.**
- Именно понятие глобального времени является ключевым ограничением для любой модели параллельных вычислений. Показано, что недетерминированный параллелизм более фундаментальный механизм, чем детерминированное последовательное вычисление. Это позволяет отказаться от необходимости строгой фиксации точек синхронизации (как это реализуется и использованием механизма тактовой частоты) и использовать механизм «стохастической синхронизации» которые позволяет решить проблему семантики для широкого класса **недетерминированных языков программирования**.
- Модель поведенческой семантики, дополненная моделью ожидающих событий, образуют так называемый **неполный семантический домен**, в котором программам, написанным на языках, основанных на акторах, присваиваются определенное значения.

Теория (5_1)

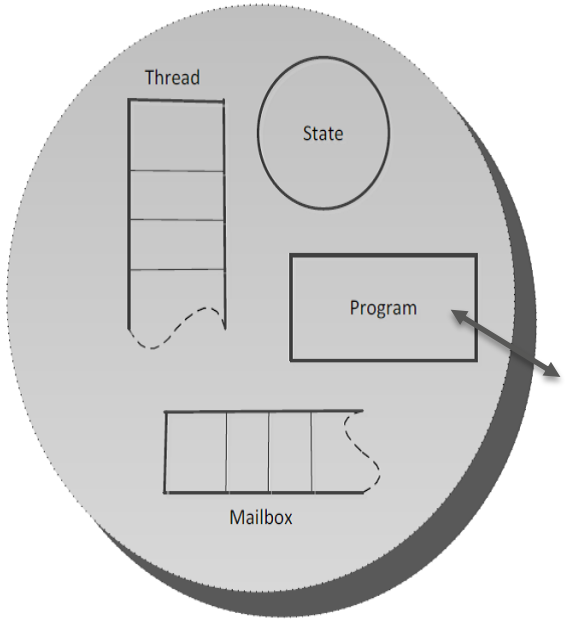
- **Денотационная семантика** (то есть семантика, **связывает каждую часть программы с математическим объектом, представляющим её значение)** совместима **с декларативной семантикой логического программирования**, в которых программы состоят из объявлений (деклараций) условий решения задачи, а не из операторов присваивания или управления.
- Декларативные объявления в действительности являются операторами символьной логики и определяют суть понятия «поведенческая семантика».
- Возможности (законы) локальной разрешимости могут быть доказаны для языка, семантика которого построена на концепции акторов. Однако, даже небольшое изменение семантики языка могут нарушать условия локальной разрешимости.

Должны быть определены достаточные требования к языку программирования, выполнение которых гарантирует условия локальной разрешимости акторной модели

Теория (6)

- В работе Foundations of Actor Semantics William Duglas Clinger показано, что акторная модель обладает неограниченным уровнем **недетерминизмом**, в то время как машина Тьюринга является ограниченно недетерминированной машиной.
- Из этого автор статьи делают вывод, что существуют алгоритмы, которые можно реализовать в акторной модели, но нельзя реализовать на машине Тьюринга. **Это относится к задачам РВ (теорема** – ее суть : итерация цикла может быть заменена на вызов функции, а функциональные языки удобно использовать там, где нужна параллельность и реакция в режиме реального времени**)**
- Поэтому любая попытка реализации акторной модели на «обычных» компьютерах, «исповедующих» вычислимость по Тьюрингу, приведет к тому, что будет реализован лишь частный случай акторной модели.

Теория (7)



Но в 1988 году Hewitt и Agha опубликовали статью Guarded Horn clause languages: are they deductive and Logical?, в которой показали, что для модели акторов текущее состояние программы может дедуктивно (логически) не следовать из предыдущего состояния. **Э**то значит что «классические последовательные технологии» отладки программ на основе модели акторов не может быть так же эффективна, как в случае последовательных (императивных) программ.

В диссертации предложен подход к отладке акторных программ, который следует парадигме функционального программирования, которая рассматривается с точки зрения требований параллельного программирования. Такой механизм реализован на основе программного таймера (такой таймер обычно «тикает» 64 раза в секунду)**. Частое и**сполнение прерываний отнимает часть вычислительных ресурсов, поэтому увеличение их количества в единицу времени должно замедлить скорость

Таймер является глобальным ресурсом, он «тикает» с одинаковой частотой для всей системы целиком. Получается, что если какая-то программа увеличивает частоту таймера, то это сказывается на поведении всей системы….

В проекте показано, какой интервал прерываний (но не 64 раза в секунду) **для акторной программы дает реальный положительный эффект с точки зрения достижения технологических требований в нотациях понятий «разрешимости и вычислимости»**

Теория (8)

- Оценки накладных расходов, связанных с организацией параллельности в акторных моделях могут быть велики.
- Сайт benchmarksgame.alioth.debian.org предлагает измерять производительность программ, написанных на разных языках, сравнивая решения одной и той же задачи.
- Надо помнить, что затраты ресурсов на вычисления и на удовлетворение условий, где нужен мгновенный отклик, совершенно разные.
- Так, физиологически, невозможно показывать одинаково высокие результаты на спринтерских и марафонских дистанциях, просто потому, что от мышц требуются совершенно разные свойства. Аналогичные требования возникают с использованием вычислительных систем.
- Сейчас производители процессоров «уперлись» в технологические ограничения по наращиванию тактовых частот, поэтому идея параллельности на всех уровнях организации вычислений набирать популярность.

Можно показать, что есть возможности расширение существующих программных языков, чтобы писать параллельные программы в «привычном стиле» **работы с локальной памятью**

Как сформулировать системные требования к архитектуре СК

Высказывание «специалистов», что:

> будущее за процессорами с большим количеством ядер, большими данными, облачными вычислениями, интеллектуальным анализом данных,

предлагается доопределить через идею «само-применимости» вычислительных систем, а именно:

> будущее вычислительных технологий за
>
> > гибридными оперативно реконфигурируемыми системами, способными проводит «вычисления в оперативной или «около» оперативной памяти», имеющей большое количество портов для обработки данных реальном масштабе времени, используя результаты проведенных вычислений как размеченные обучающие выборки для своей непрерывной «технологической» отладки .

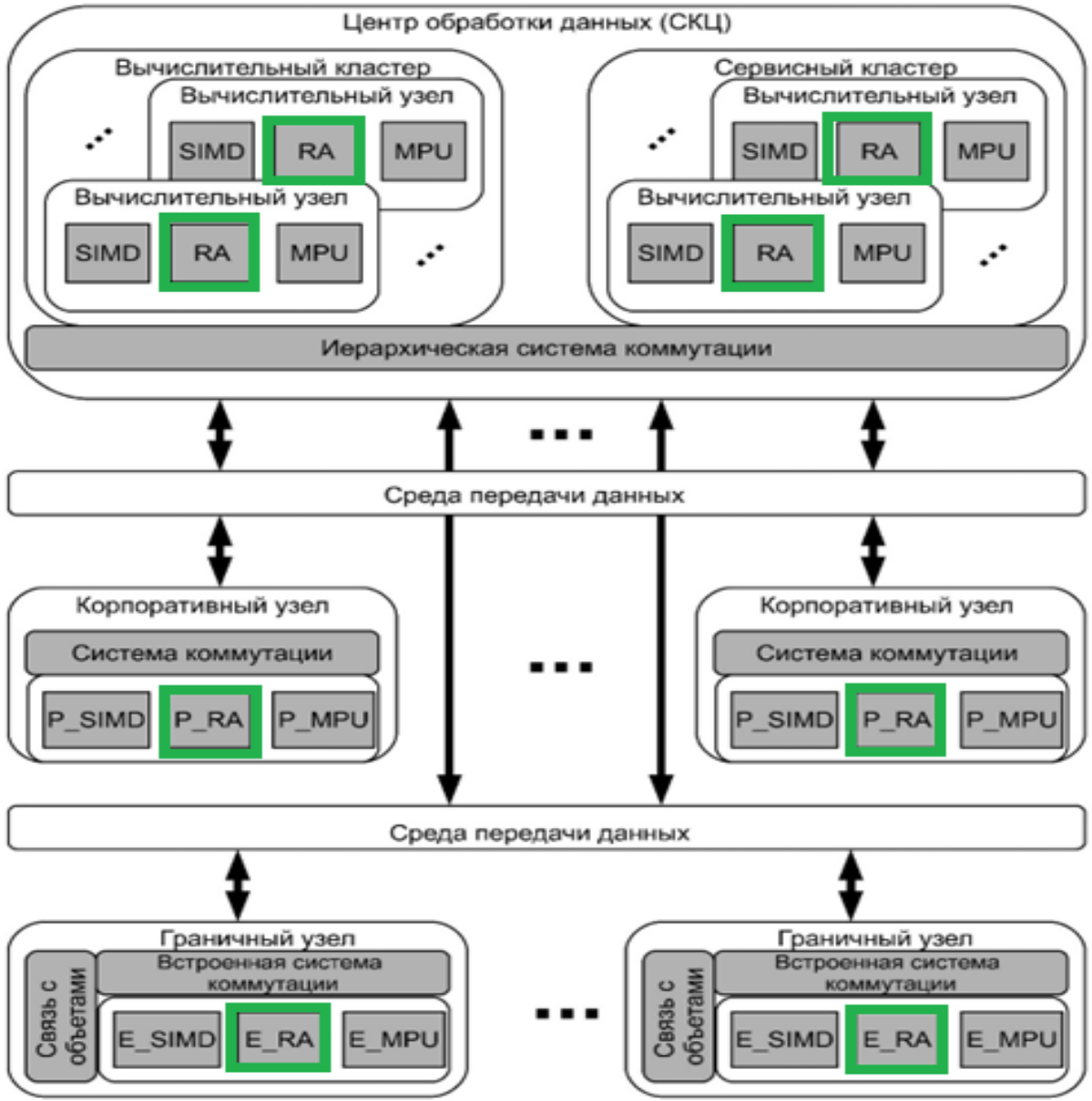
АРХИТЕКТУРА ПОДДЕРЖИВАЮЩАЯ ПРИНЦИПЫ САМОПРИМЕНИМОСТИ ВЫЧИСЛЕНИЙ

Уровень «объяснения» результатов

Энерго-вычислительная эффективность >4 Гфлопс/Вт

Уровень «машины Геделя»: механизмов «внимания» и «понимания» реализуемых процессов
Энерго-вычислительная эффективность >10 Гфлопс/Вт

Уровень «машины Тьюринга» - CPU/GPU/FPGA процессоры
Энерго-вычислительная эффективность >20 Гфлопс/Вт



Условие реализуемости алгоритмов вычисления в РВ

$$\Psi = \Omega / \tau$$

Необходимое быстродействие системы управления равно отношению сложности Ω объекта управления к длительности τ цикла управления (цикл тактовой частоты)

$$\Omega = n \cdot m^{1/2} \cdot q \cdot g^{1/2} \cdot p \cdot s^{1/2},$$

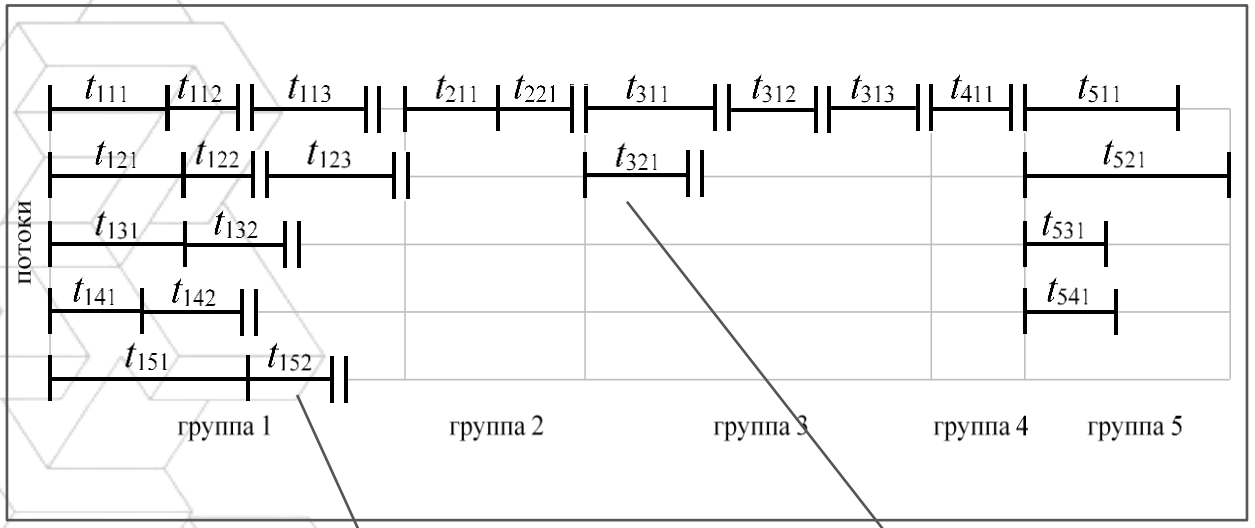
где n — число типов элементов, m — среднее число элементов одного типа, q — число типов связей, g — среднее число связей одного типа, p — среднее число контролируемых параметров, посредством которых описывается состояние отдельного элемента, s — среднее число отслеживаемых состояний контролируемых параметров

$$\tau = T_{\text{инт}} = \min\left(\frac{\Delta r}{v}\right),$$

где $T_{\text{инт}}$ — временной шаг интерполяции,
Δr — допустимая погрешность позиционирования, v — скорость движения рабочего органа

Для большей части современного технологического оборудования, в котором расчёт параметров движения осуществляется системой управления для каждого узла интерполяции, длительность цикла управления равна временному шагу интерполяции

Диаграмма оптимального цикла реализации алгоритмов управления движением

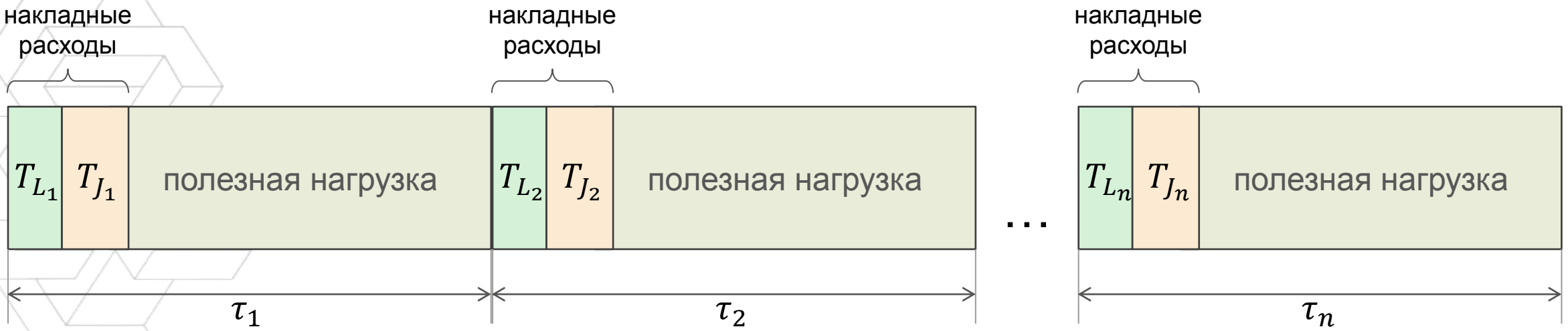


t_{111}, t_{121}, t_{131}, t_{141}, t_{151} — задержки коммуникационной сети при передаче данных от датчиков и сенсоров в модули очувствления; t_{112} — задержка системы технического зрения, СТЗ (без ИНС); t_{113} — задержка искусственной нейронной сети (ИНС) при обработки данных для СТЗ;
t_{122} — задержка системы распознавания речи, СРР (без ИНС);
t_{123} — задержка ИНС при обработки данных для СРР; t_{132}, t_{142}, t_{152} — задержки модулей обработки данных от датчиков положения, скорости и др.; t_{211}, t_{221} — задержки ядра: интерпретатора (не в каждом цикле) и задержка при обработке данных от модулей очувствления; t_{311} — задержка модуля трансформации; t_{312} — задержка модуля интерполяции; t_{313} — задержка модуля эквидистантной коррекции;
t_{321} — задержка модуля предварительного просмотра; t_{411} — задержка модуля разгона-торможения; t_{511}, t_{521}, t_{531}, t_{541} — задержки регуляторов исполнительных устройств; | | — задержка записи/чтения данных в общей памяти (δ)

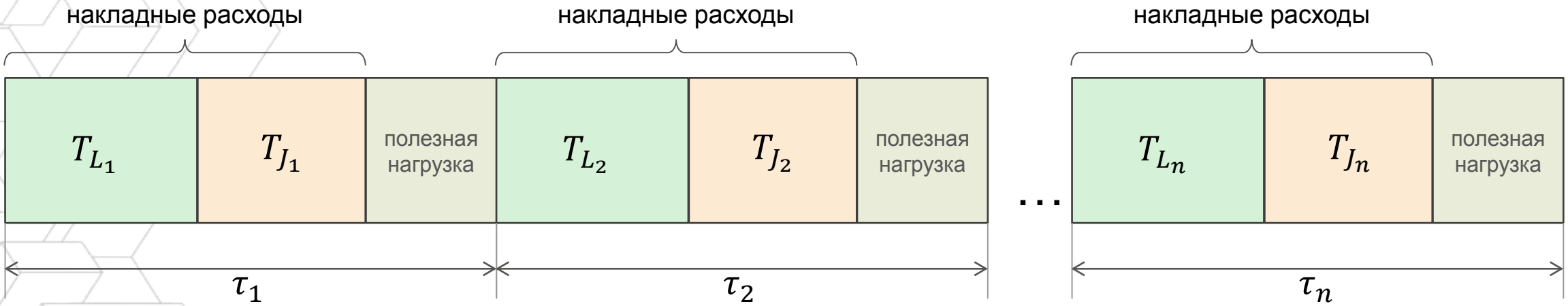
$$\tau = \max(t_{111} + t_{112} + t_{113} + 2\delta,\ t_{121} + t_{122} + t_{123} + 2\delta,\ t_{131} + t_{132} + \delta,\ t_{141} + t_{142} + \delta,$$
$$t_{151} + t_{152} + \delta) + t_{211} + t_{221} + \delta + \max(t_{311} + t_{312} + t_{313} + 3\delta,\ t_{321} + \delta) +$$
$$+ t_{411} + \delta + \max(t_{511},\ t_{521},\ t_{531},\ t_{541}) = \sum_{i=1}^{5} \max\left(\bigcup_{j=1}^{j_{i\max}} \sum_{k=1}^{k_{ij\max}} (t_{ijk} + \delta)\right) - 2\delta$$

Пример: Влияние сложности программного обеспечения на цикл управления

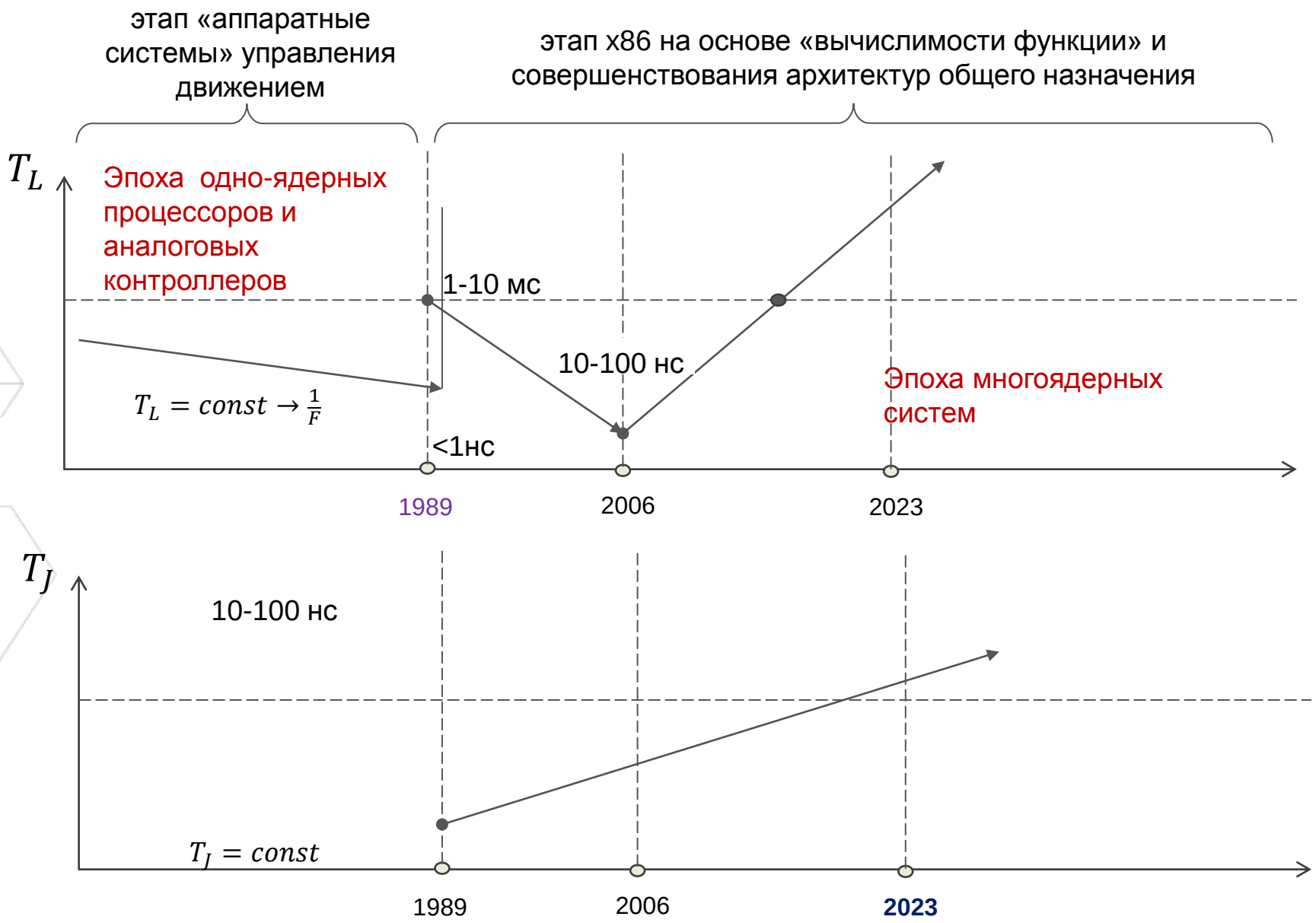
Одноядерные процессоры (до 2006 г.)



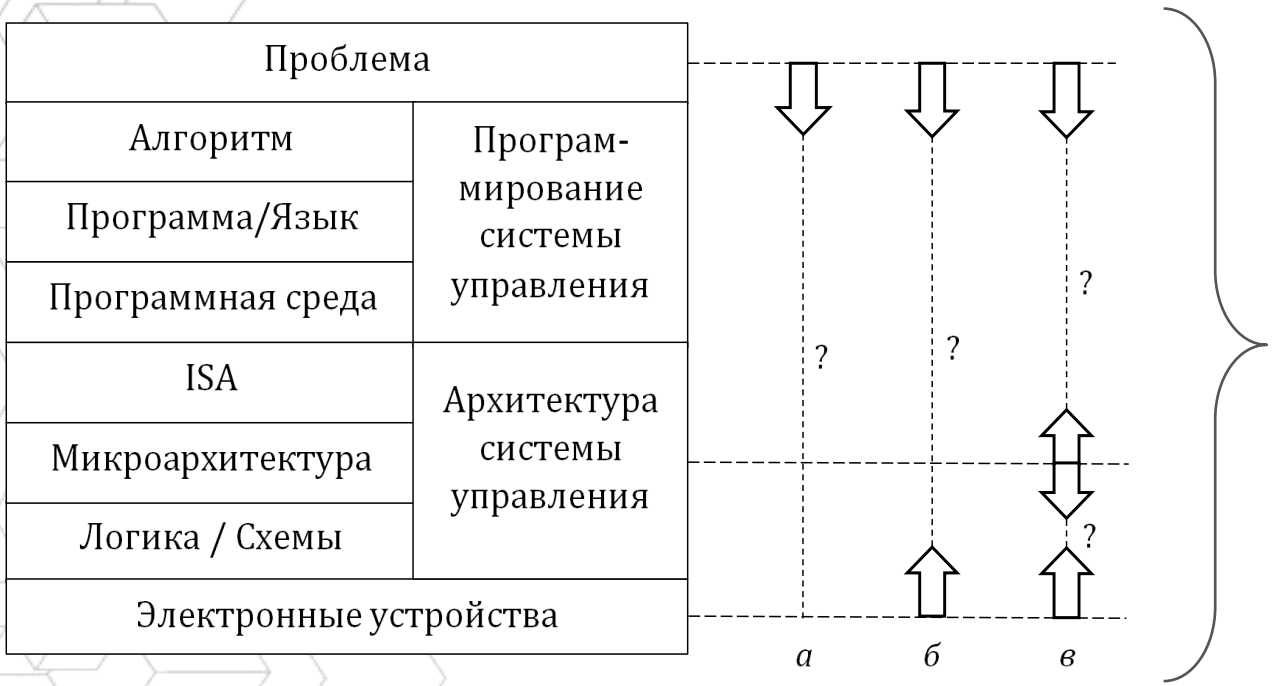
Многоядерные процессоры (до н/в)



Влияние архитектуры вычислительных систем на реализацию алгоритмов моделирования процессов в реальном масштабе времени



Подходы к решению задач синтеза алгоритмов вычислений в РВ



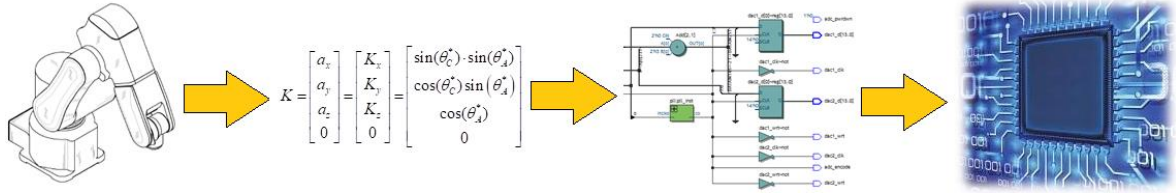
КОМПЛЕКСНАЯ ПРОБЛЕМА СИНТЕЗА :

1. **Средства реализации СУД (аппаратные/программные и др.);**
2. **Определение реализуемости СУД;**
3. **Обеспечение доверия к СУД;**
4. **Комплексная методология синтеза быстродействующих доверенных систем управления движением:**
   a) **Определение архитектуры СУД;**
   b) **Методы синтеза подсистем СУД;**
   c) **Методология программирования СУД.**

***а* — синтез системы, исходя из сформулированной проблемы;**

***б* — синтез СУД исходя из поставленной проблемы и доступных электронных устройств;**

***в* — синтез СУД исходя из поставленной проблемы, доступных устройств и выбранной концептуальной модели архитектуры**

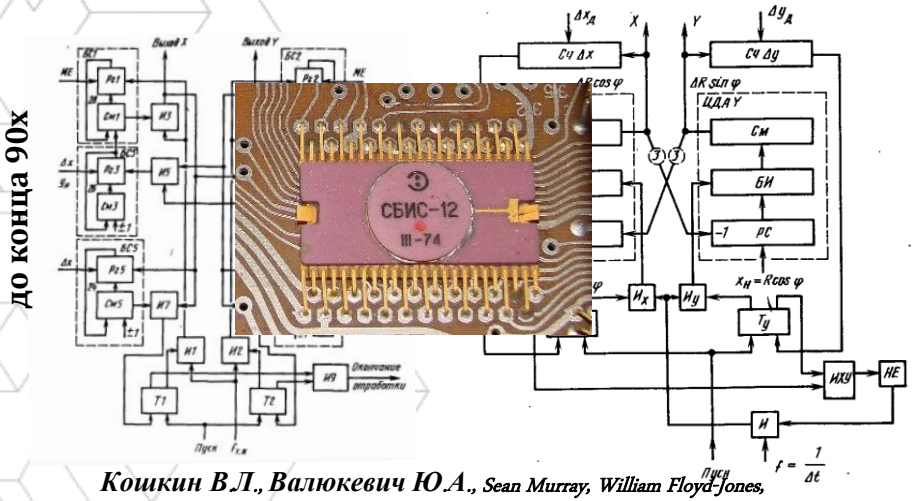


Синтез СУД исходя из поставленной проблемы (вариант *а*)

Средства реализации систем вычисления в реальном времени

Аппаратные

Бурное развитие до конца 90х



Кошкин В.Л., Валюкевич Ю.А., Sean Murray, William Floyd-Jones, Konrad Gac, Shiqi Lian, Yinhe Han и др.

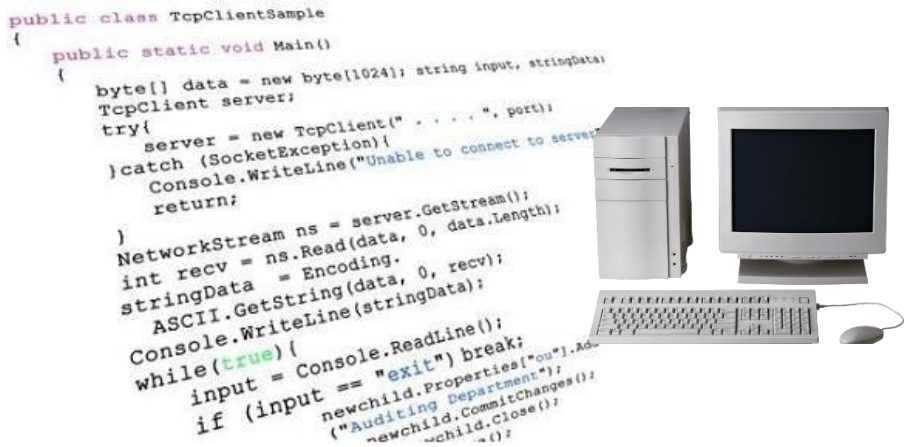
В основе проектирования <u>аппаратных</u> систем управления движением и СЧПУ лежит принципы цифровой схемотехники. В целом вся логика строится на теории комбинационной и последовательной логике и их производных.

<u>Ключевое преимущество:</u> Сверхвысокое быстродействие,. Полная детерминированность реакции на событие. Возможность реализации на отечественном технологическом базисе.

<u>Ключевые недостатки</u>: - существенная переработка при незначительном дополнении функционала; - трудоемкий процесс синтеза; - высокие требования к разработчикам; - централизованная разработка и др.

Процессоро-центрические (Архитектура фон Неймана)

Бурное развитие с начала 90х



Сосонкин В.Л., Мартинов Г.М., Suk-Hwan Suh, Ian Stroud, Seong-Kyoon Kang и др.

<u>RISC</u>

Байкал Т (MIPS) – Микрос/Ресурс
Байкал М (ARM-A57) – Ресурс
Скиф (ARM-A53) – Ресурс

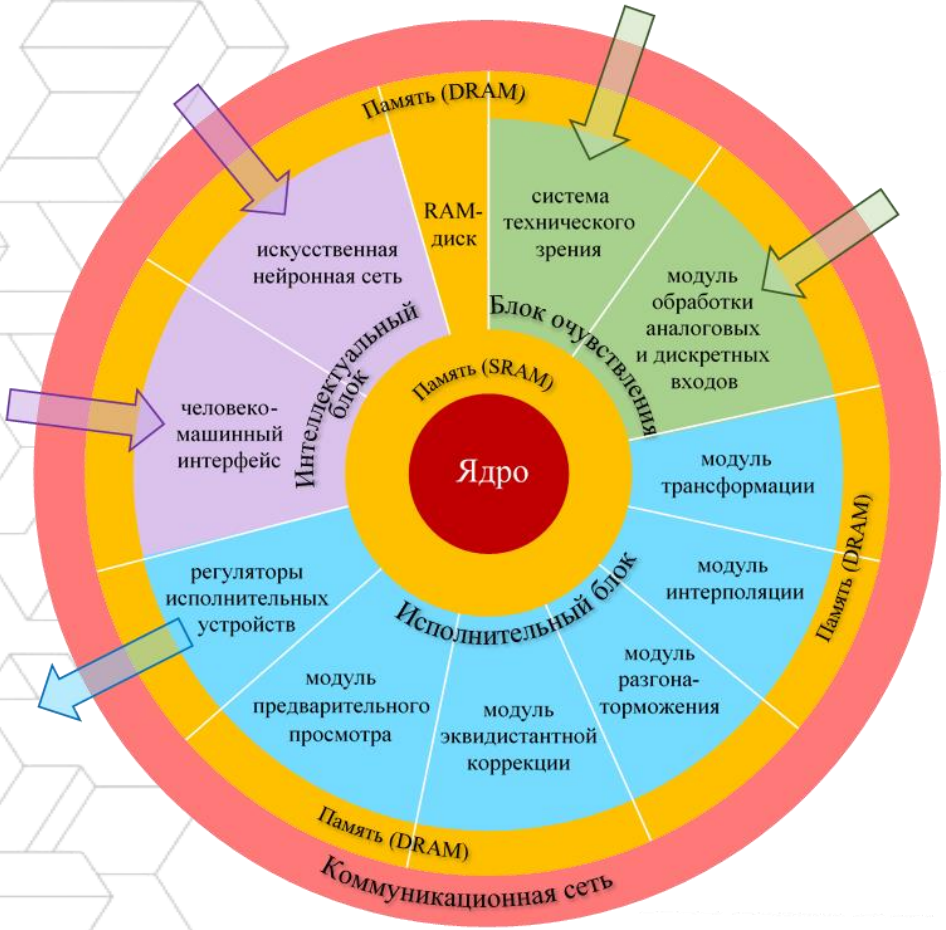
<u>CISC</u>

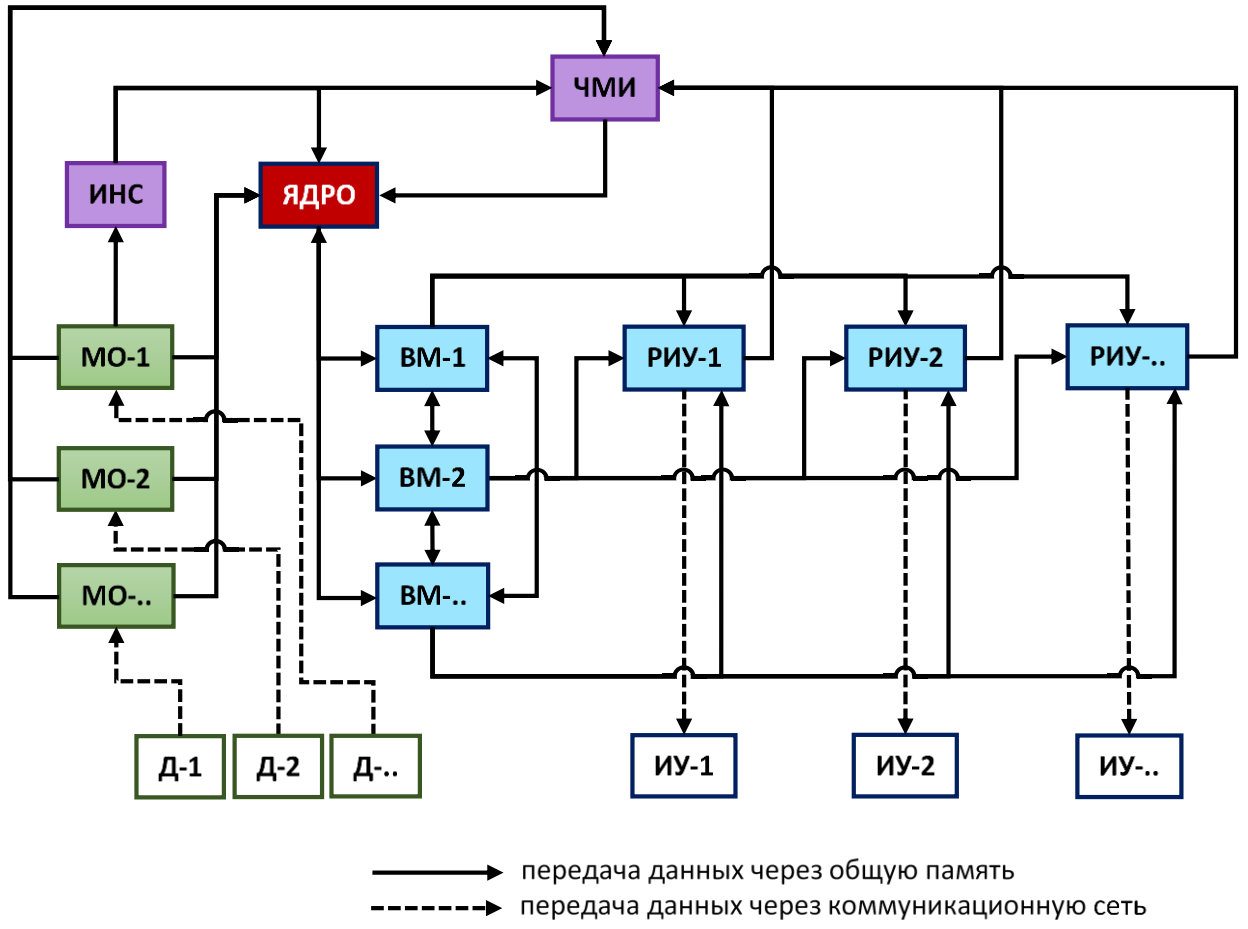
Intel/AMD – Аксиома/LinuxCNC/ Mach3/PUMOTIX и др.
Эльбрус 8С – Аксиома

<u>Ключевое преимущество:</u> Гибкость проектирования и модернизации ПО СУД. Синтез системы управления происходит на уровне прикладного ПО с использованием языков высокого уровня с обязательным присутствием операционной системы.

<u>Ключевые недостатки</u>: - низкое время динамической реакции на события (высокие latency, Jitter, определенная архитектурой приоритетность прерываний); процессорно-центрическая подход в совокупности архитектурой фон Неймана не позволяет реализовывать нагруженные алгоритмы в цикле управления; - Невозможность адаптировать аппаратную архитектура под объект управления; - проблемы обеспечения доверия и др.

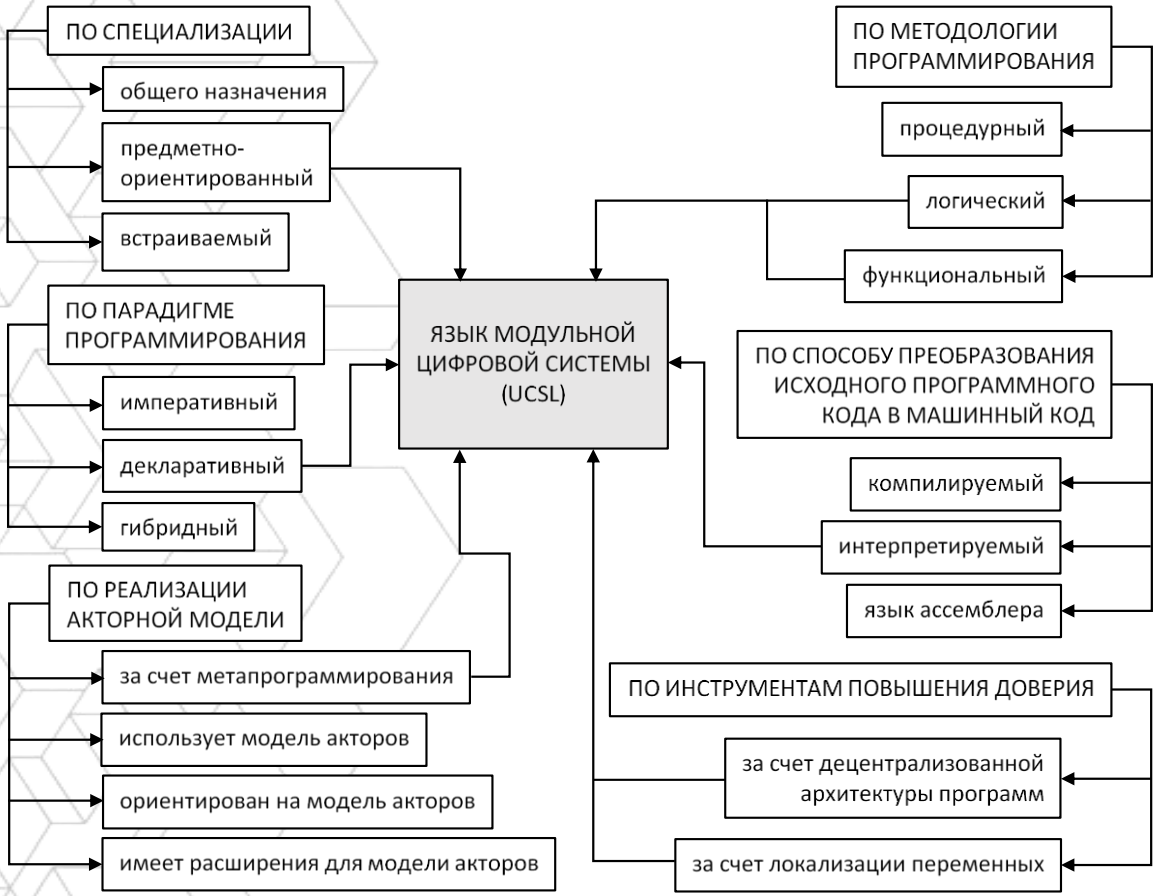
Концептуальная и программно-инструментальная архитектура память-ориентированных вычислительных систем





Декларативный акторный язык программирования память-ориентированной вычислительной системы

Классификационная модель языка



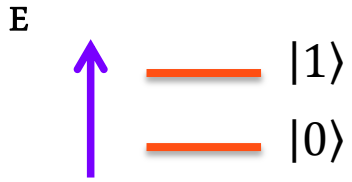
Модель применения языка

- Выполнение кода в стандарте **ISO**
- Объявление переменных стандартных типов **BOOL, BYTE, WORD, INT, REAL**
- Объявление структур и массивов
- Встроенные математические функции **ABS, ACOS, ASIN, ATAN, ATAN2, CEIL, COS, EXP, FLOOR, LOGN, LOG, POW, SIN, SQRT, TAN**
- Организация условного ветвления **IF...THEN...ELSE**
- Организация выбора **SELECT...CASE**
- Организация циклов **DO WAIL...LOOP , DO...LOOP WAIL, FOR...NEXT**
- Определение подпрограмм с параметрами

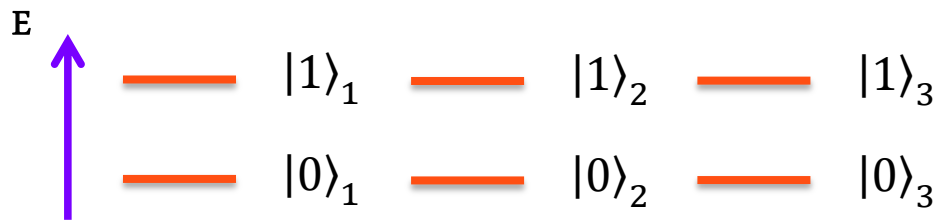
```
// Переменные
INT count = 7                    // Счётчик
REAL step = 2.273/3              // Шаг

// Простой код ISO
F200 S900 T1 D2 M3
G0 X100 Y100                     // Подвод
N40 G1 X150 G91                  // Прямоугольник
Y120
X-150
Y-120
IF count GOTO40                  // Првтор
G0 X0 Y0 Z0 G90                  // Отвод

// Условный оператор
IF count == 0 THEN               // Если 0:
G1 Z20.6
ELSE IF count == 1 THEN          // Если 1:
Z40 X2.8
ELSE                             // Иначе:
Z0 X0
END IF                           // Конец

// Цикл
DO WHILE count > 0               // Пока > 0:
CALL1001(step, count)            // Подпрограмма
LOOP
```

Квантовые памяти-центрические парадигмы vs классические принципы вычислений (машина фон Неймана)

$S = 0 \vee 1$



$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$



Bit **вычисления в памяти** **Q-Bit**

До акта измерения кубит находится в состоянии суперпозиции своих возможных состояний

Новое свойство – запутанность

Нет запутанности

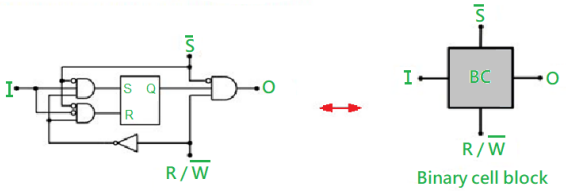
$\alpha_1|0\rangle_1 + \beta_1|1\rangle_1 + \alpha_2|0\rangle_2 + \beta_2|1\rangle_2 + \alpha_3|0\rangle_3 + \beta_3|1\rangle_3$ – **2N** чисел

Есть запутанность

$\alpha_{000}|000\rangle + \alpha_{001}|001\rangle + \alpha_{010}|010\rangle + \alpha_{011}|011\rangle +$

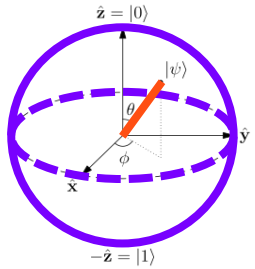
$+\alpha_{100}|100\rangle + \alpha_{110}|110\rangle + \alpha_{101}|101\rangle + \alpha_{111}|111\rangle$ – $\mathbf{2^N}$ чисел

Число возможных (логически разделенных) состояний n-разрядного двоичного регистра 2^n



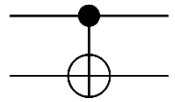
Элементарная однокубитная операция – поворот на сфере Блоха

$$|\psi\rangle = \cos\frac{\Theta}{2}|0\rangle + e^{i\phi}\sin\frac{\Theta}{2}|1\rangle$$



Двухкубитные операции генерируют запутаность кубитов

CNOT

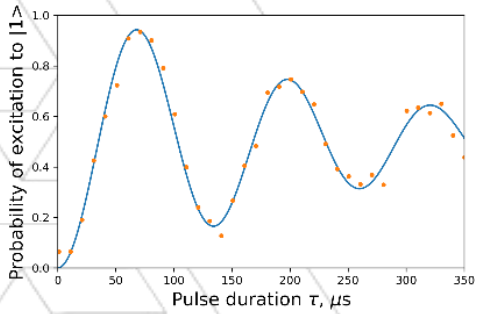


$|00\rangle \rightarrow |00\rangle$
$|01\rangle \rightarrow |01\rangle$
$|10\rangle \rightarrow |11\rangle$
$|11\rangle \rightarrow |10\rangle$

Особенности носителей информации и новых компьютерных архитектур «вычисления в памяти»

Время когерентности

Характерное время на котором сохраняются квантовые (логические) свойства (состояния) системы



Число кубитов N

Число двухуровневых квантовых систем на которых можно проводить однокубитные и двухкубитные операции

Достоверность операции

$$F = \frac{\text{Число успешных операций}}{\text{Число проведенных операций}}$$

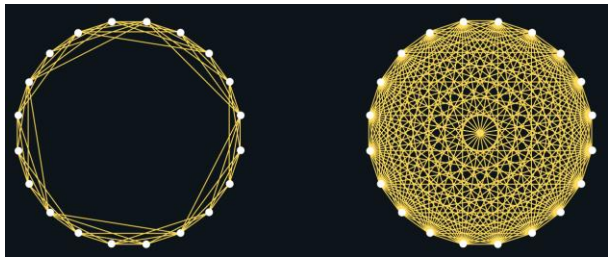
регистры и счетчики - два компонента, которые играют важные, но разные роли в хранении данных и манипулировании ими

Доступная «глубина» цепочки «L» логических вентелей/кубитов

- $F_{\text{алгоритм}} = F_1 F_2 \dots F_L > 50\ \%$
- **Пример:** $F_i = 99.9\% \rightarrow L \approx 500$
- **Пример:** $F_i = 99\% \rightarrow L \approx 50$

Связность

Между какими кубитами можно провести двухкубитную операцию



Квантовый объем (число возможных состояний, находящихся в суперпозиции) = 2^N

- **Если, число кубитов не меньше чем N то на этом массиве можно реализовать квантовый алгоритм состоящий из двухкубитных операций глубиной N**

Потенциал вычислений над «запутанными состояниями»

новые алгоритмы

- https://quantumalgorithmzoo.org/ - имеется 50 квантовых алгоритмов, превосходящих классические. Сколько есть алгоритмов, реализуемых в режиме «вычисления в памяти»?
- Так, алгоритм Шора позволяет раскладывать число на простые множители за $O(N^3)$ вместо $O(2^{N/4})$ операций
- Быстрый поиск по базам данных (алгоритм Гровера)
  $O(\sqrt{N})$ вместо $O(N)$ операций
- Быстрое решение систем линейных уравнений
  $O(\mathbf{Log}(N))$ вместо $O(N)$ операций

Области применения

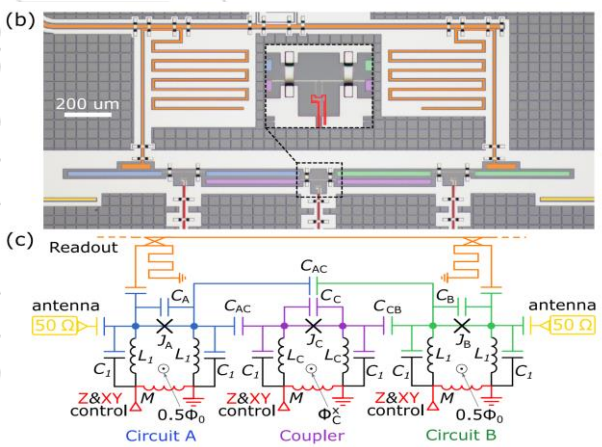
- Взлом классической криптографии
- Синтез новых химических соединений/лекарств
- Решение логистических задач
- Моделирование динамики сложных систем
- Моделирование ранее недоступных физических процессов (например, динамика атмосферы)
- Машинное обучение, искусственный интеллект (сильный искусственный интеллект?)
- ???

Требования

- Пример: для взлома RSA 1024 (алгоритм Шора) нужно примерно 3000 идеальных (логических)(кубит.
  Или 3 млн кубит с ошибкой 0.01%.
  Или 10 млн кубит с ошибкой 0.1%.
- Сейчас есть
  15 кубит с ошибкой 0.5 % (ионы)
  433 кубита с ошибкой 2-5 % (сверхпроводники)

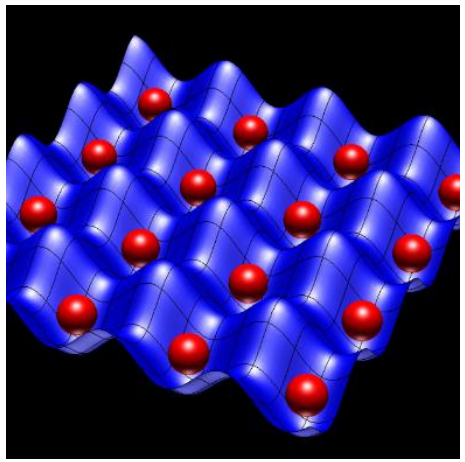
А.К. Федоров, Е.О. Киктенко, Н.Н. Колачевский “Вычислимое и невычислимое в квантовом мире: утверждения и гипотезы” , УФН 2024, DOI: 10.3367/UFNr.2024.07.039721

Три платформы для организации вычислений в «запутанной» памяти: «архитектура без разделения процессоров и памяти) (+ полупроводники + фотоны + ?)



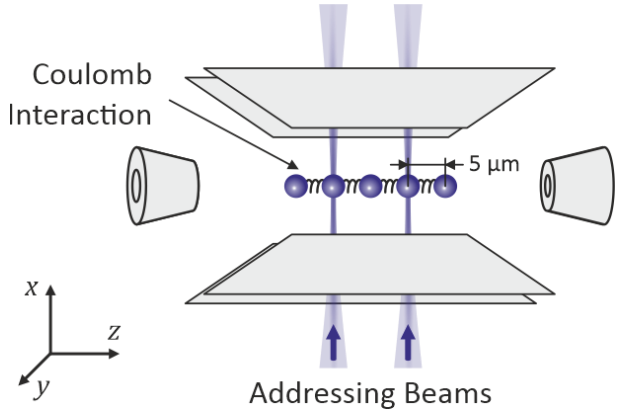
Сверхпроводники

- **Сверхпроводящая электрическая цепь в криостате**
- **Кубит – коллективном состоянии многих электронов**



Атомы

- **Атомы в оптической решетке**
- **Кубит – в электронных состояниях атома**
- **Двухкубитная операция – через ридберговские состояния атома**



Ионы

- **Ионы в электромагнитных ловушках**
- **Кубит – в электронных состояниях иона**
- **Двухкубитная операция – через кулоновское взаимодействие**

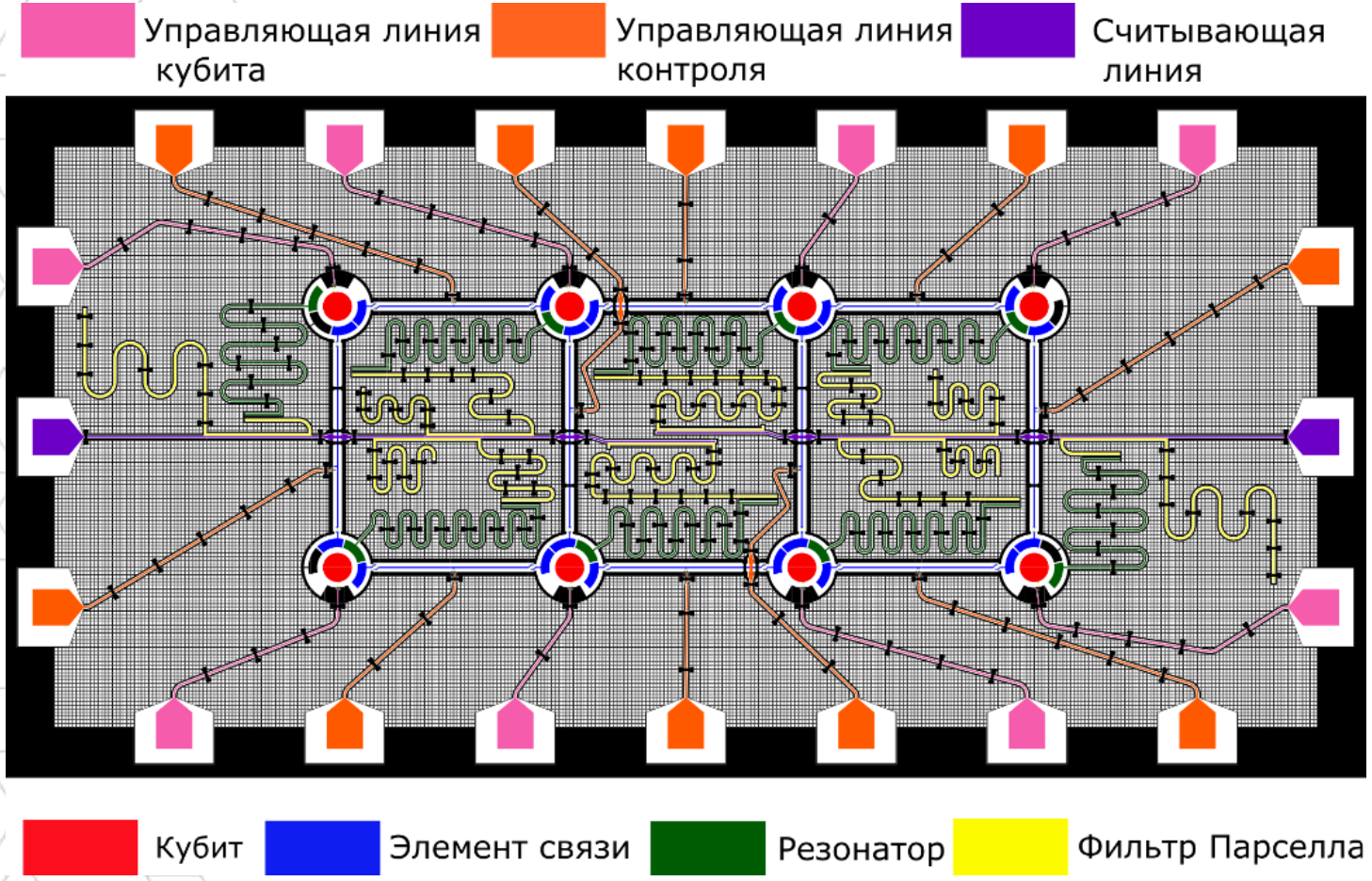
Сверх-проводниковые квантовые компьютеры в России

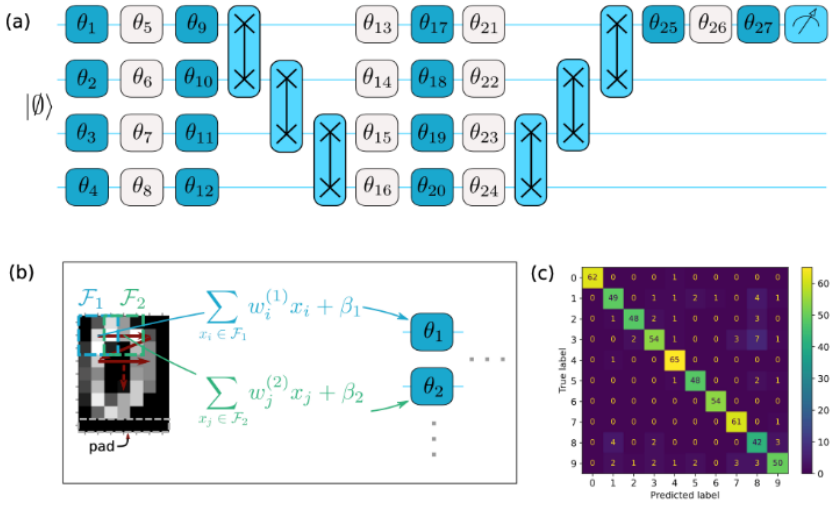
МФТИ

МИСИС УНИВЕРСИТЕТ

РКЦ Российский Квантовый Центр

МГТУ им Баумана





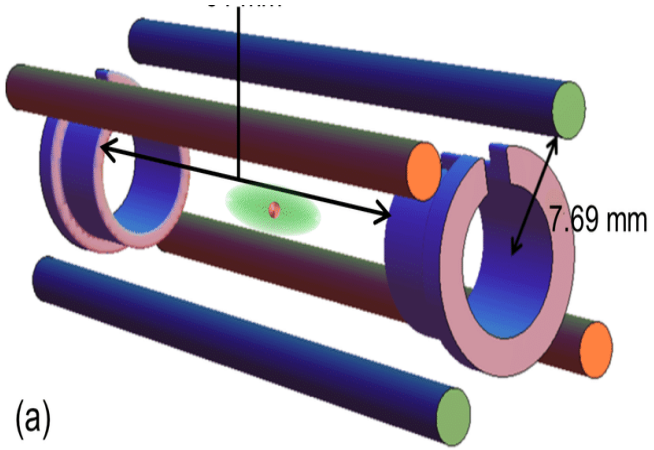
Характерные параметры

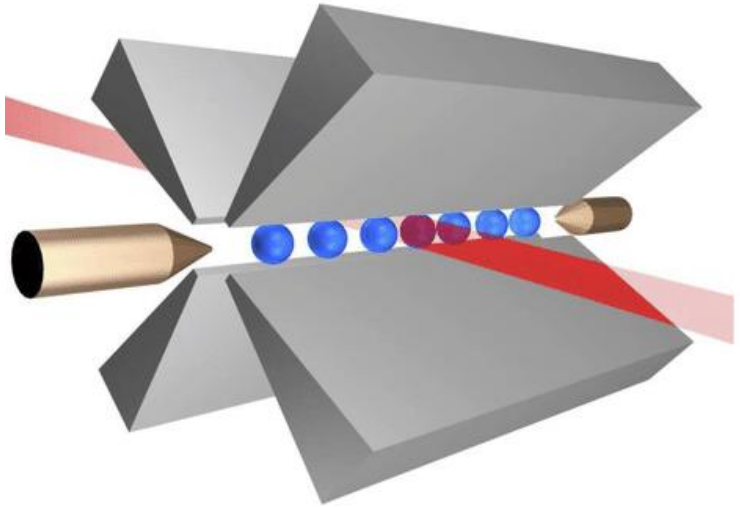
- **Рекордное время когерентности до 1 часа** [1]
- **Рекордные достоверности операций в многокубитных системах** [2]:
  - F_{1Q} = 99.99916(7)%
  - F_{2Q} = 99.97%
- **Рекордный квантовый объем 2^{20}**
- **Количество кубит на сегодняшний день – до 56**
- **Связность «все со всеми»**
- **Характерное время однокубитной операции** – 1-10 мкс
- **Характерное время двухкубитной операции** – **50-200 мкс**

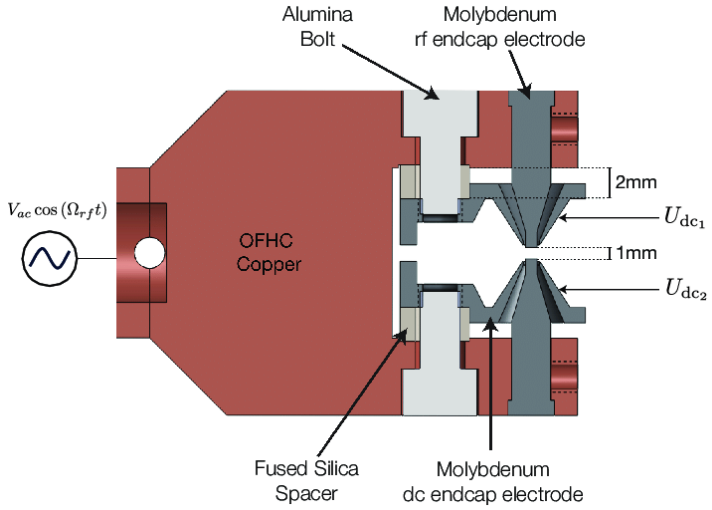
[1] Wang P. et al. //Nature communications. – 2021. – Т. 12. – №. 1. – С. 233.
[2] Löschnauer C. M. et al. //arXiv preprint arXiv:2407.07694. – 2024.

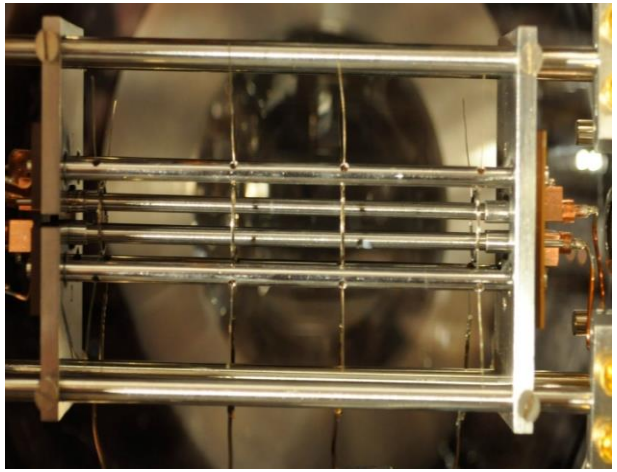
Проблема устойчивости квантовых носителей информации: «ловушка» квантов (ионов) Пауля

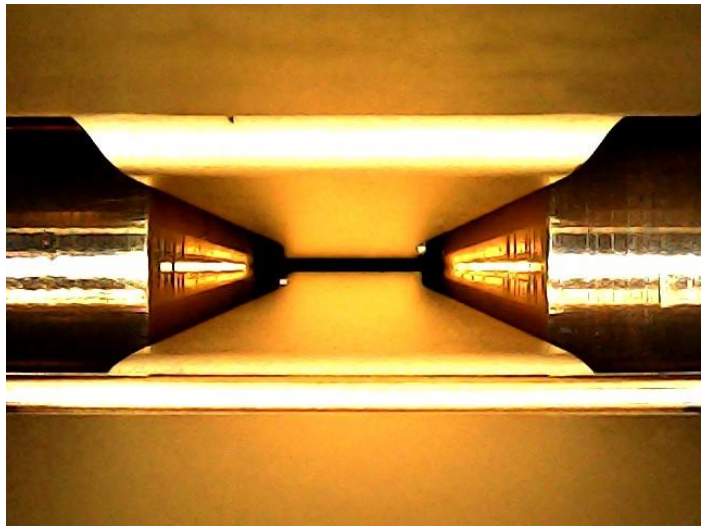
Реализация «ловушек»

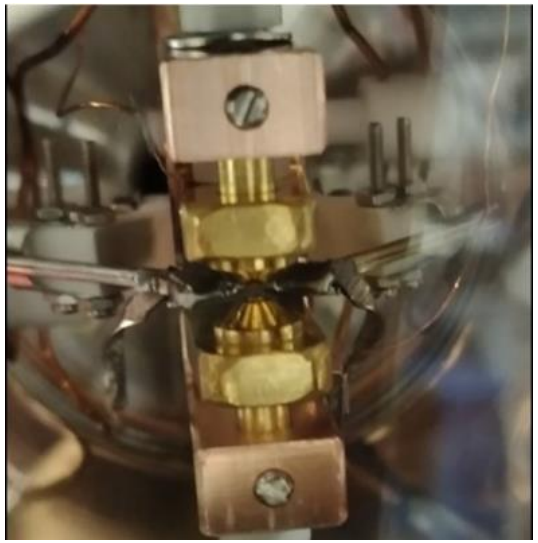




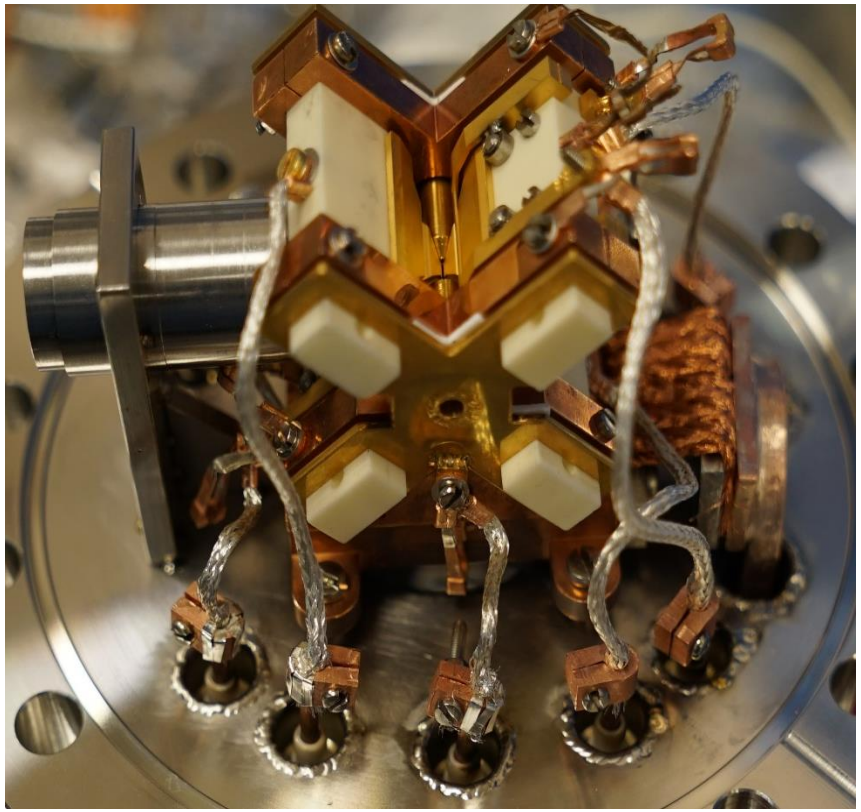


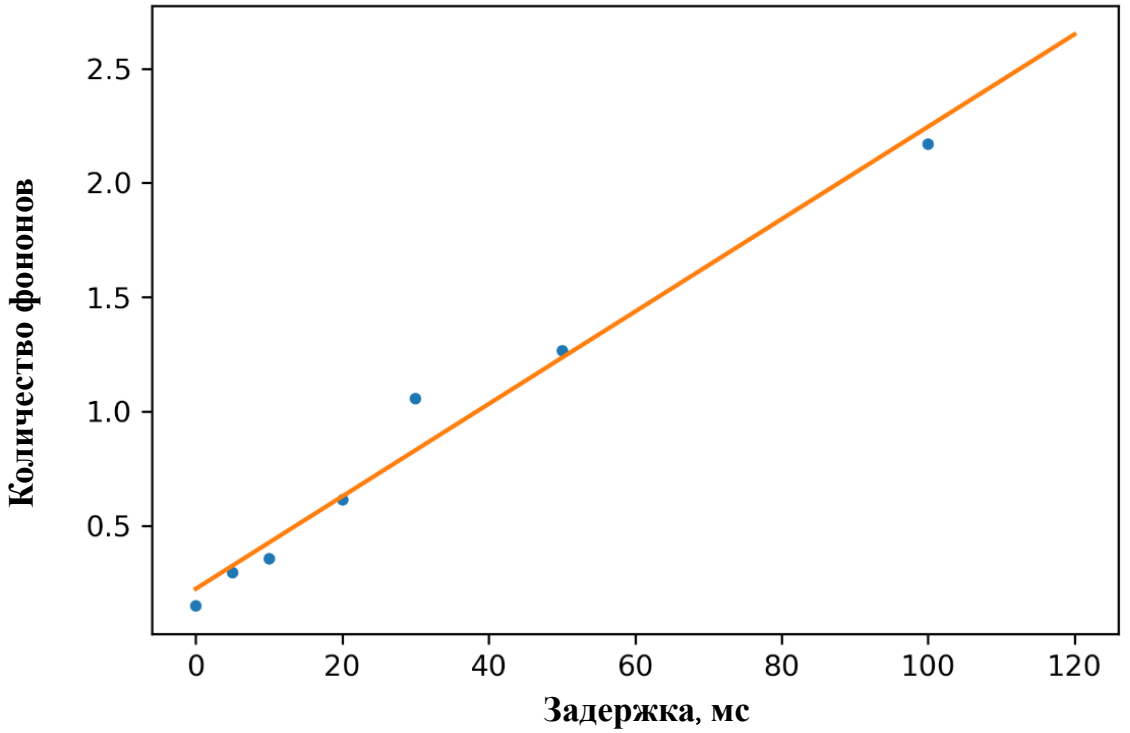






Скорость образования носителей в установки ФИАН - 20 фононов/с

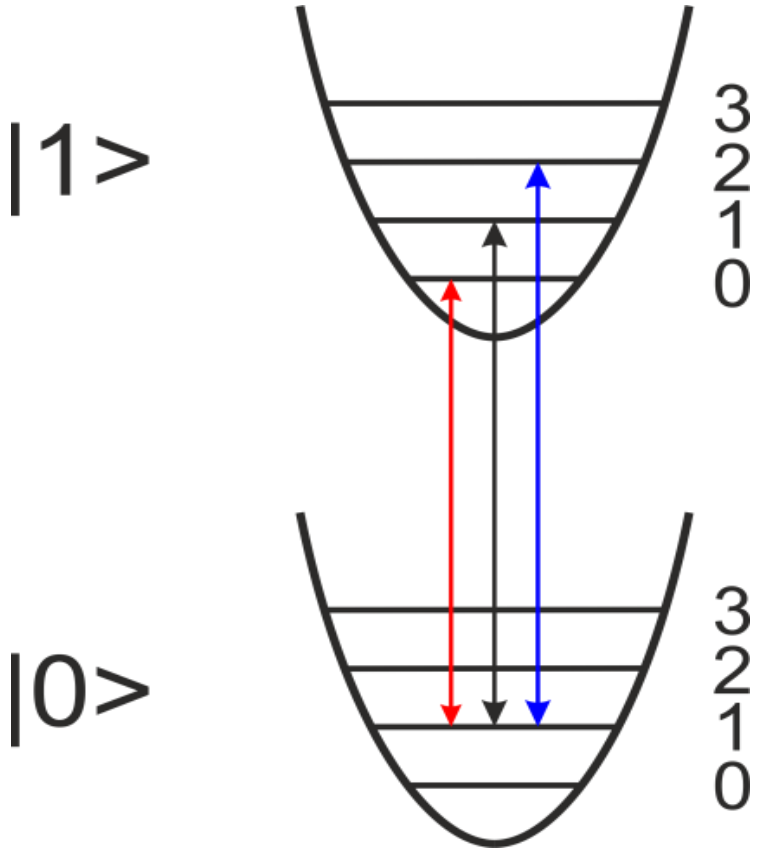


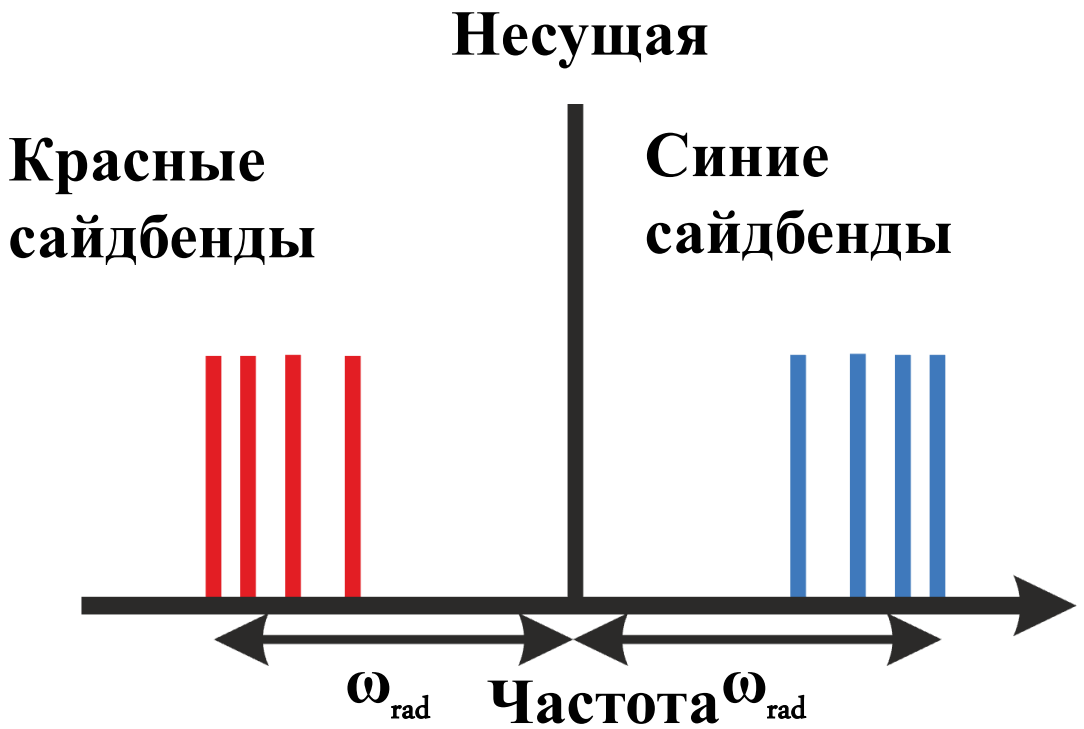


$$\omega_{rad} = 2\pi \times 3.8 \text{ MHz}$$

Многочастотные колебательные моды – шина данных

Квантовые асинхронные колебания, которыми можно когерентно управлять (осуществлять запись данных непосредственно в память, где происходят вычисления)



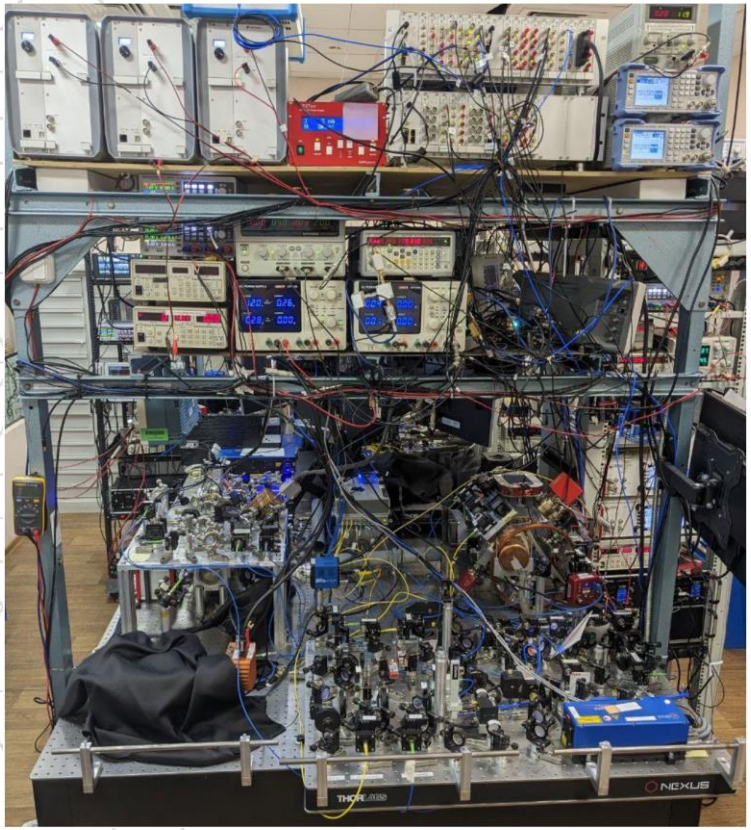


Экспериментальная установка

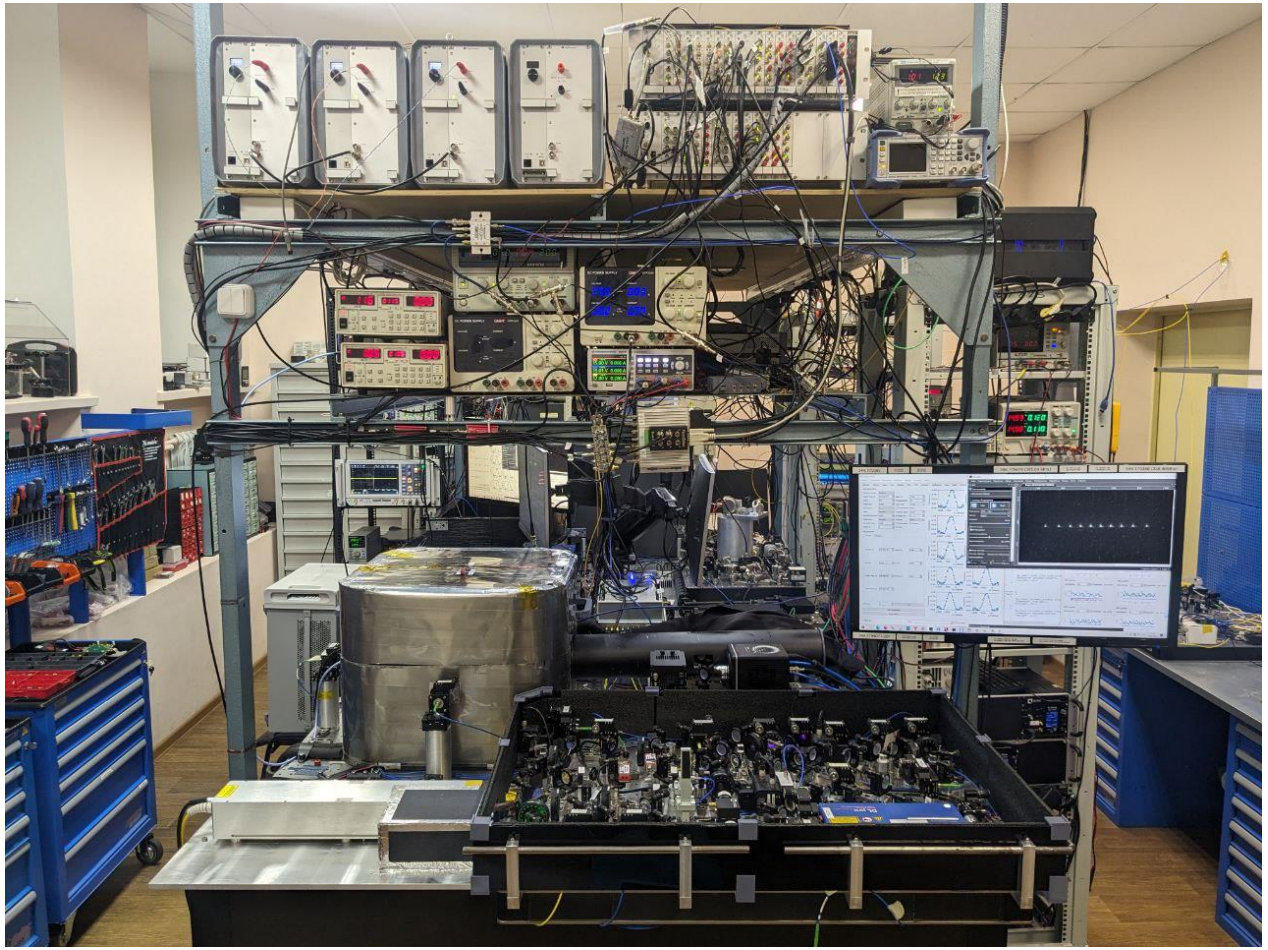
ФИАН

РКЦ

2022 год



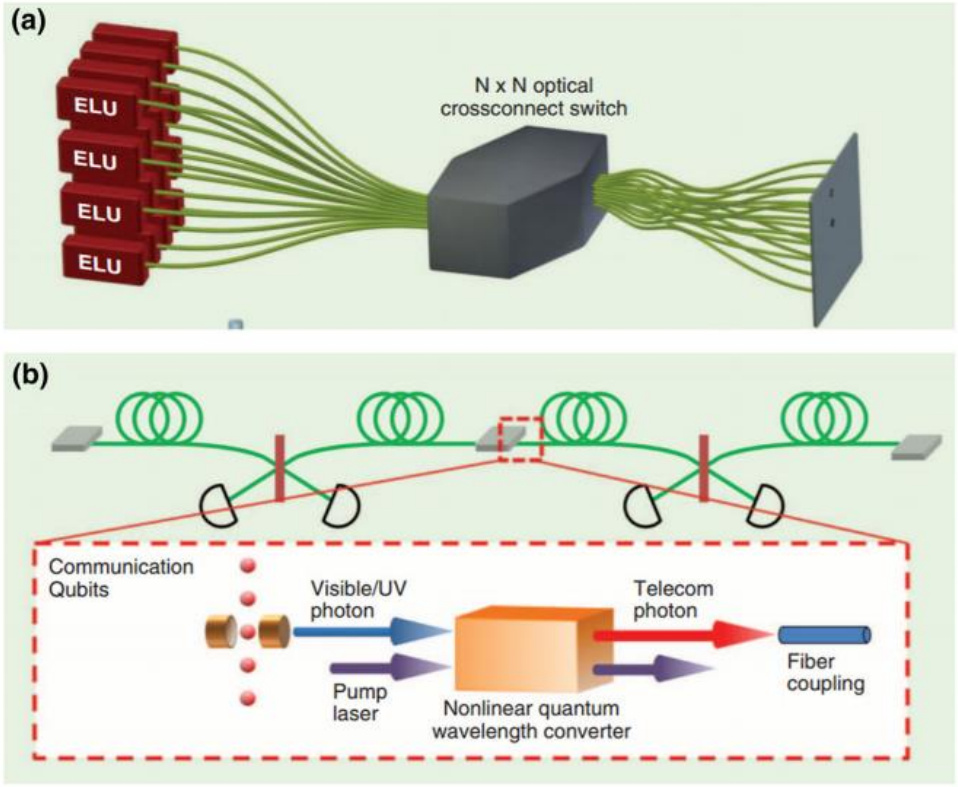
2023 год



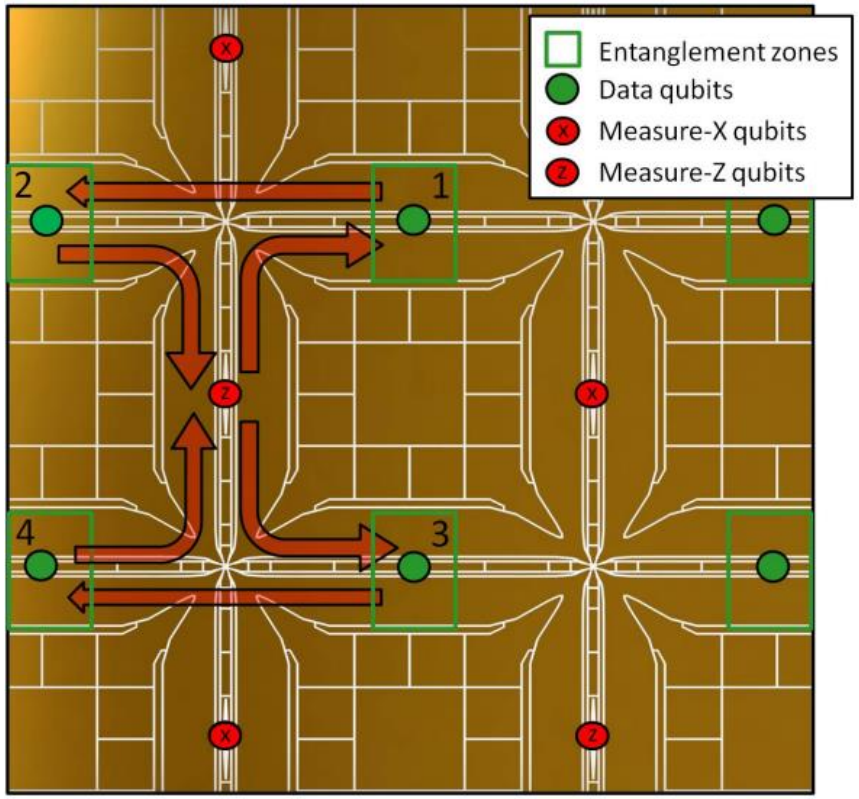
Подходы к масштабированию

<u>Главная проблема , которая должна быть решена в рамках проекта</u>: сложность работ
Все медленно и сложно.

<u>Базовая идея</u>: перейти на работу с несколькими цепочками малого размера, одновременно заполняя их данными. Архитектура «вычисления в памяти»



Фотонные интерконнекты



QCCD