

Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)

Студенческий практикум по обработке графов знаний на вычислительной платформе «Тераграф»

Алексей Юрьевич Попов,
к.т.н., доцент кафедры Компьютерных систем и сетей
Москва, октябрь-декабрь 2023 г.



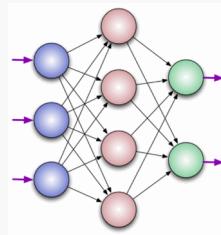
01

Графы знаний

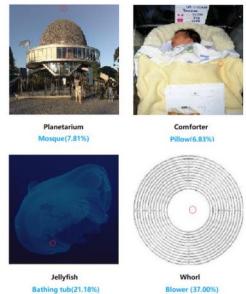


Проблемы слабого искусственного интеллекта

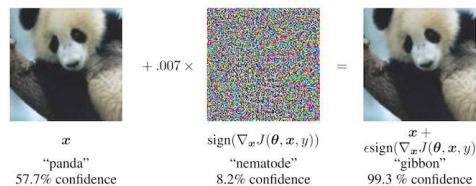
Искусственные
нейронные сети
генерируют выходные
данные для любого
входного шаблона



Результат обучения
нейронной сети не
всегда предсказуем



Внесение шума в
изображение
существенно
снижает точность
распознавания



Трансляционная
инвариантность
приводит к ошибкам

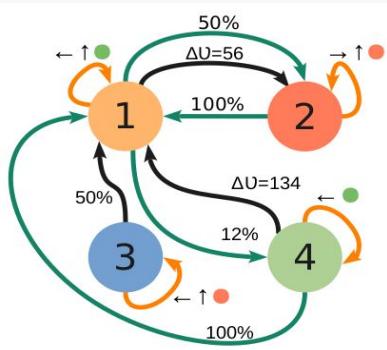


Дэниэлл Денетт,
философ из Университета
Тафтса

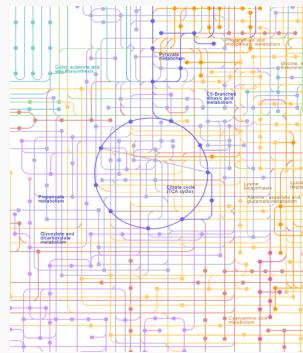
«Я считаю, что если мы собираемся использовать эти вещи и зависеть от них, тогда нужно понимать, как и почему они действуют так, а не иначе. Если они не могут лучше нас объяснить, что они делают, то не стоит им доверять».

Представление знаний

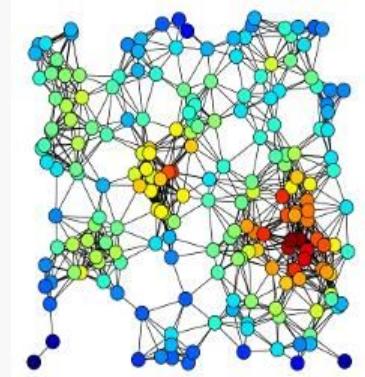
Знания представляются в виде графовых моделей, позволяющих однозначно интерпретировать результат. Вершины и ребра графа представления знаний обладают атрибутами, которые анализируются алгоритмами и позволяет делать логический вывод.



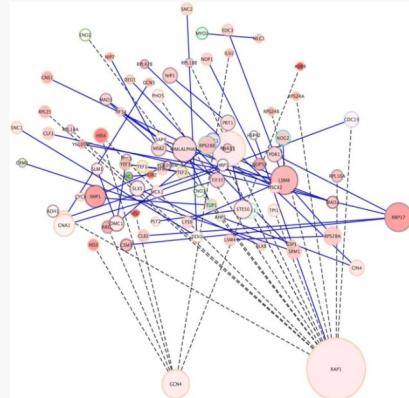
Динамический граф сцены



Фрагмент графа обмена веществ



Граф результатов анализа контрагентов для участника рынка



Граф белок-белковых взаимодействий

Аналитическая система на основе графов знаний



Финансовый сектор

Анализ участников рынка
Валютные операции
Оценка участников рынка
Обнаружение мошеннических схем
Выявление незаконных финансовых операций

Безопасность

Контроль информационных потоков
Агрегация данных видео-аналитики
Контекстное распознавание объектов на фото
Автономный автотранспорт

Биология и медицина

Анализ белок-белкового взаимодействия
Моделирование биологических систем
Персонифицированная медицина

Графы знаний и ИИ

Хранение и анализ контекстной информации
Повышение качества распознавания образов
Преобразование речевой информации в графы

Графы знаний в биологии

- Интерактомика
- Анализ проблемы индивидуальной нормы
- Моделирование и визуализация процессов в биологических системах
- Моделирование и анализ популяций и сложных сообществ



Графы знаний в системах сильного ИИ

Обработка событий

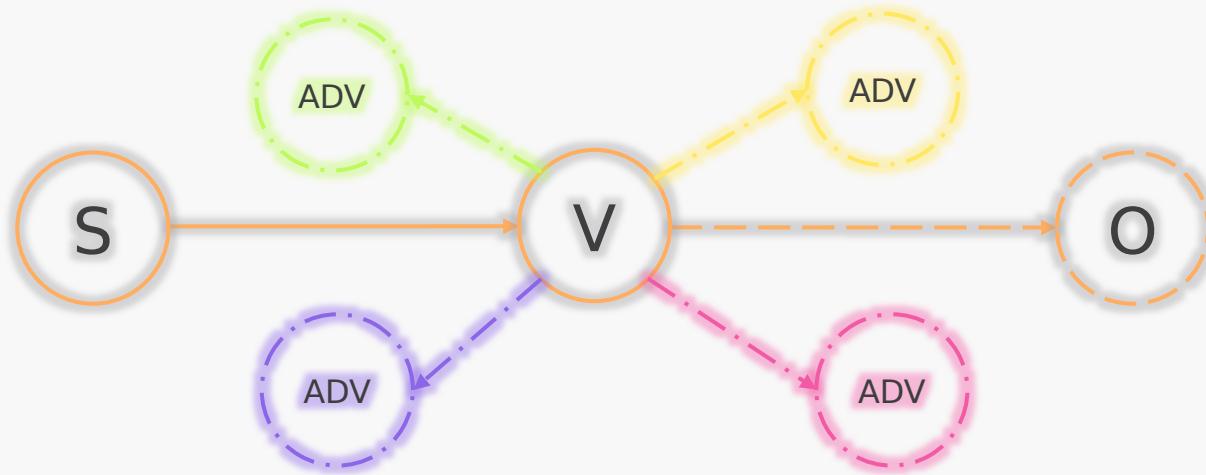
Построение алгоритмов

Принятие решений

Выдвижение гипотез

Самообучение

Образы действия, время, места, эмоциональный окрас, цели и причины непосредственно связаны с вершиной V



02

Вычислительный комплекс «Тераграф»



Существующие подходы к обработке графов

Режимы обработки

- Обработка статичных графов
- Потоковая обработка статичных графов
- Обработка динамических графов

Программные решения

- Эффективные структуры данных
- Библиотеки обработки графов
- Графовые базы данных

Аппаратные решения

- Многопоточность и многонитевость
- Графические ускорители
- Специализированная память
- Ускорители на ПЛИС
- Микропроцессоры с набором команд дискретной математики

Операции, кванторы и функции

| Discrete math operations | Description | DISC instructions |
|---------------------------------------|---|--------------------------|
| $A = \langle A_1, \dots, A_n \rangle$ | - store function of n sets as an A tuple | Insert |
| $R(A_i, x, y), x \in A_i, y \in A_i$ | - relationship between the x and y in the set A_i | Next/Previous/Neighbors |
| $ A_i , i = 1, n$ | - cardinality of the A_i set | Cardinality |
| $x \in A_i, x \notin A_i, i = 1, n$ | - check the inclusion/exclusion of the x in the set A_i | Search |
| $A_i \cup x, i = 1, n$ | - inserting the x into the set | Insert |
| $A_i \setminus x, i = 1, n$ | - removing an element x from the set | Delete, Delete structure |
| $A \setminus A_i$ | - removing the set A_i from the tuple A | Delete structure |
| $A_i \subset A_j$ | - inclusion relation of the set A_i in A_j | Slices |
| $A_i \equiv A_j$ | - equivalence relation operation | Slices |
| $A_i \cup A_j$ | - union operation of two sets | OR |
| $A_i \cap A_j$ | - intersection operation of two sets | AND |
| $A_i \setminus A_j$ | - difference operation | NOT |
| $A_i \Delta A_j$ | - symmetric difference | — |
| \bar{A} | - complement of the A_i | NOT |
| $A_i \times A_j$ | - Cartesian product operation | — |
| 2^{A_i} | - Boolean operation | — |

Набор команд дискретной математики DISC

Операции, основанные на поиске

- Поиск по ключу *SRCH*
- Поиск минимального *MIN*
- Поиск максимального *MAX*
- Поиск следующего *NEXT*
- Поиск предыдущего *PREV*
- Ближайший больший *NGR*
- Ближайший меньший *NSM*

Операции добавления/удаления

- Вставка *INS*
- Удаление *DEL*
- Удаление множества *DELS*

Операции И-ИЛИ-НЕ

- Объединение множеств *OR*
- Пересечение множеств *AND*
- Дополнение множеств *NOT*

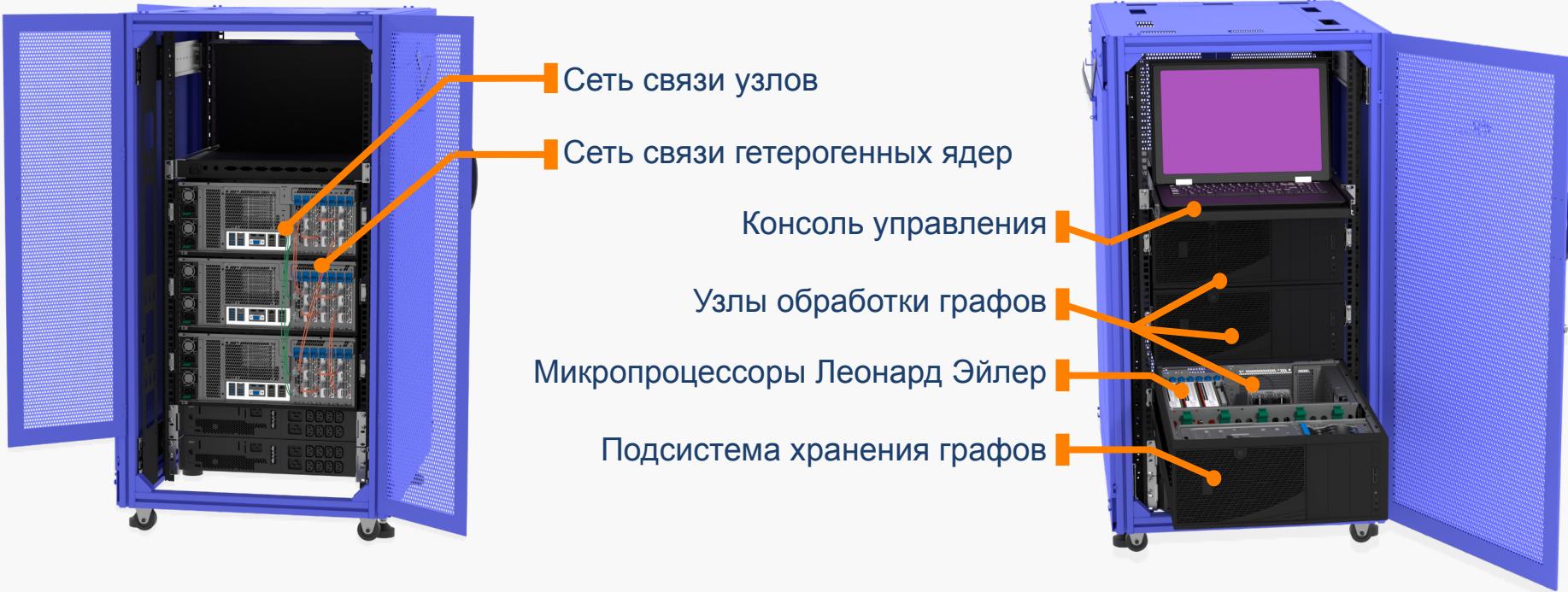
Операции среза

- Срез больше *GR*
- Срез больше или равно *GREQ*
- Срез меньше *LS*
- Срез меньше или равно *LSEQ*
- Срез меньше/больше *GRLS*

Свойства множеств

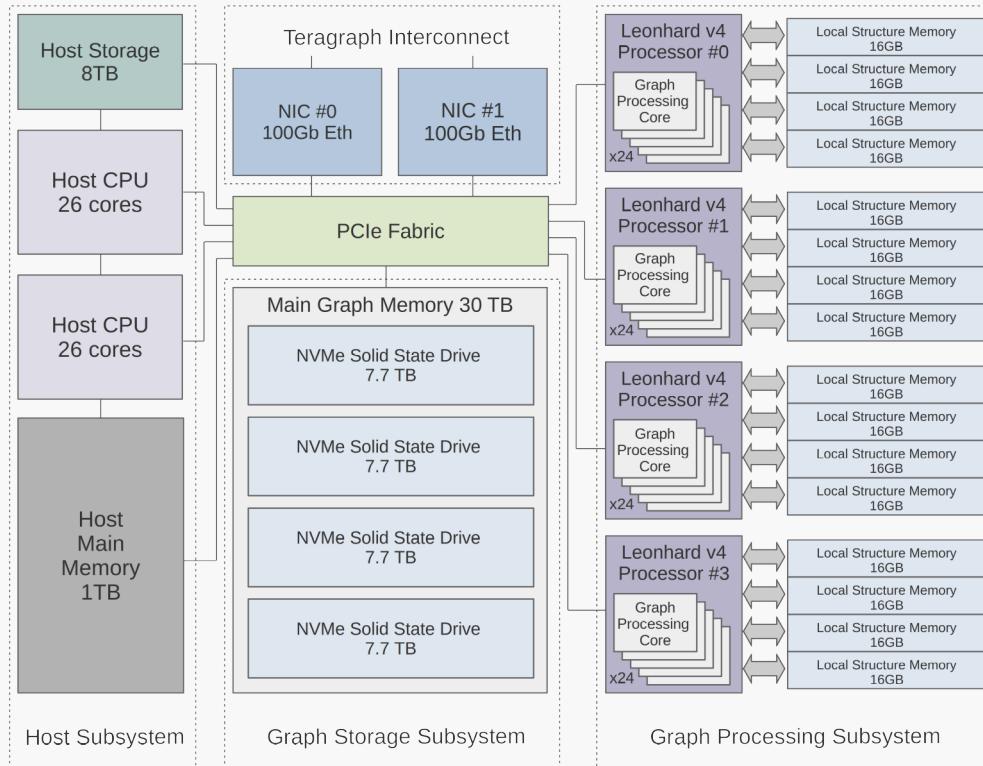
- Мощность множества *CNT*

Вычислительный комплекс Тераграф



Вычислительный узел комплекса Тераграф

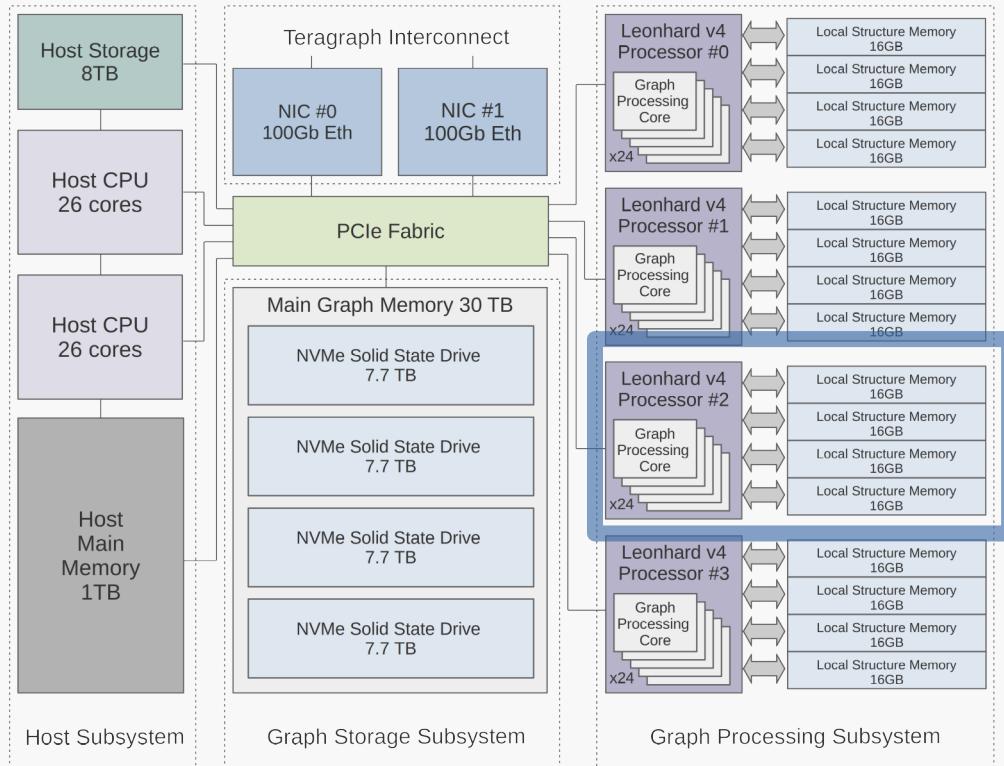
Teragraph node #1



- Предусмотрено длительное размещение графов в оперативном доступе.
- Используется ассоциативная память большого объема (2.5ГБ на одно ядро Graph Processing Core, GPC)
- Ядра GPC являются весокоэффективными гетерогенными системами, взаимодействующими через единой адресное пространство PCIe
- GPC самостоятельно обращается в локальное графовое хранилище 30ТБ и графовые хранилища других узлов
- Хост система выполняют второстепенные функции (инициализация, распределение и т.д.)

Вычислительный узел комплекса Тераграф

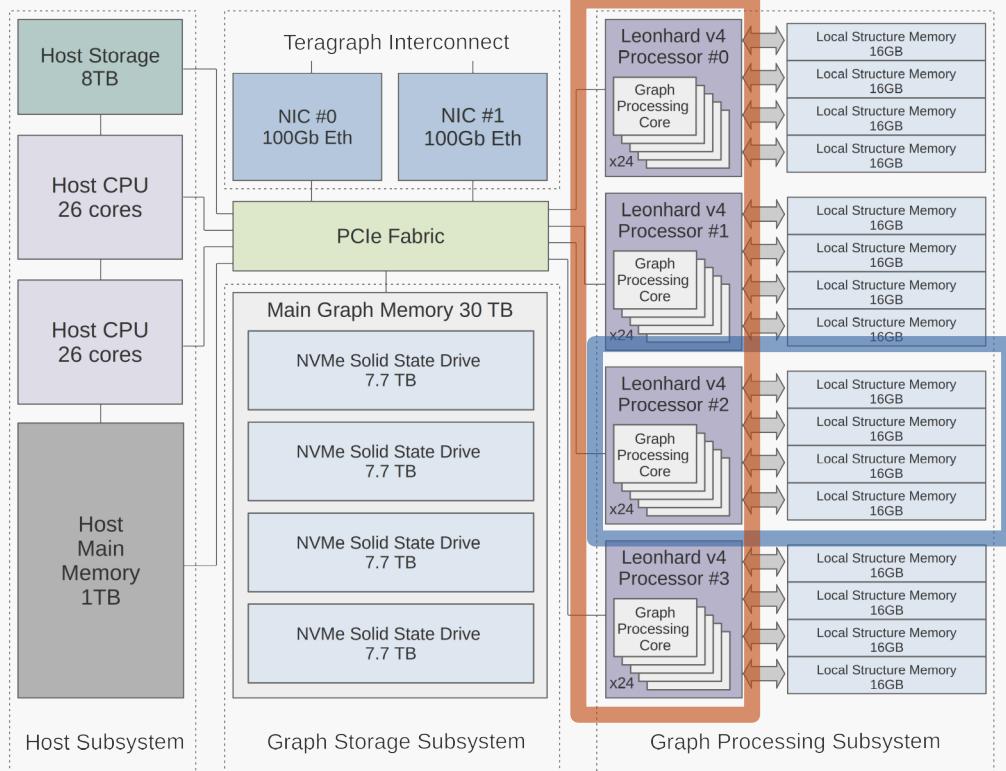
Teragraph node #1



- Предусмотрено длительное размещение графов в оперативном доступе.
- Используется ассоциативная память большого объема (2.5ГБ на одно ядро Graph Processing Core, GPC)
- Ядра GPC являются весокоэффективными гетерогенными системами, взаимодействующими через единой адресное пространство PCIe
- GPC самостоятельно обращается в локальное графовое хранилище 30ТБ и графовые хранилища других узлов
- Хост система выполняют второстепенные функции (инициализация, распределение и т.д.)

Вычислительный узел комплекса Тераграф

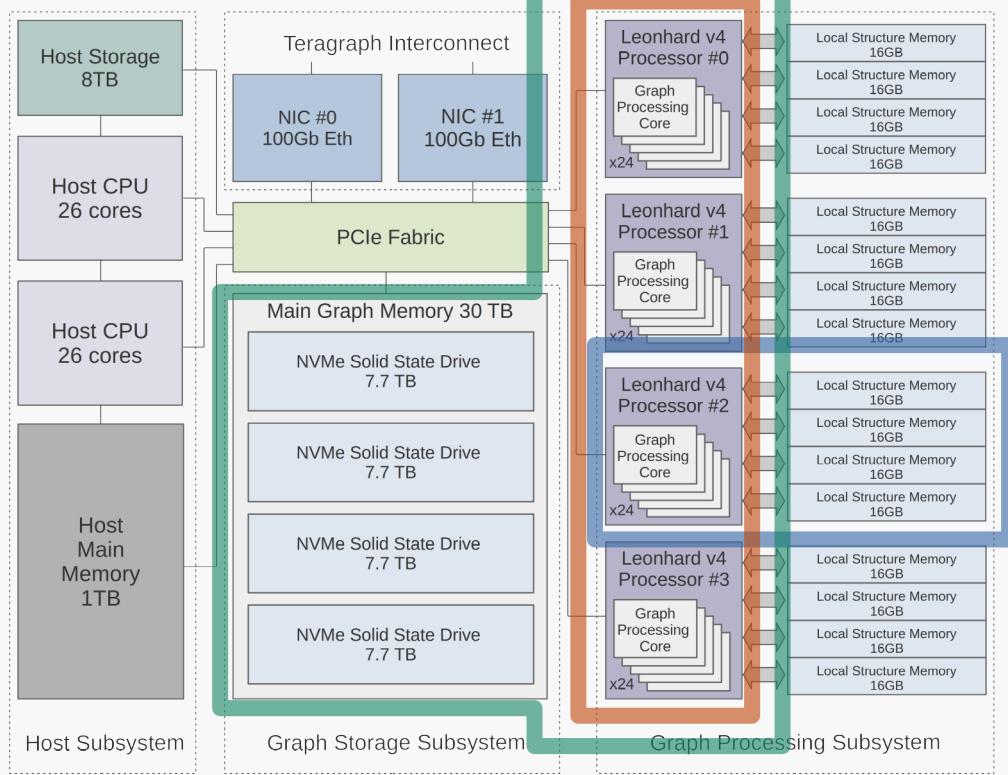
Teragraph node #1



- Предусмотрено длительное размещение графов в оперативном доступе.
- Используется ассоциативная память большого объема (2.5ГБ на одно ядро Graph Processing Core, GPC)
- Ядра GPC являются весокоэффективными гетерогенными системами, взаимодействующими через единой адресное пространство PCIe
- GPC самостоятельно обращается в локальное графовое хранилище 30ТБ и графовые хранилища других узлов
- Хост система выполняют второстепенные функции (инициализация, распределение и т.д.)

Вычислительный узел комплекса Тераграф

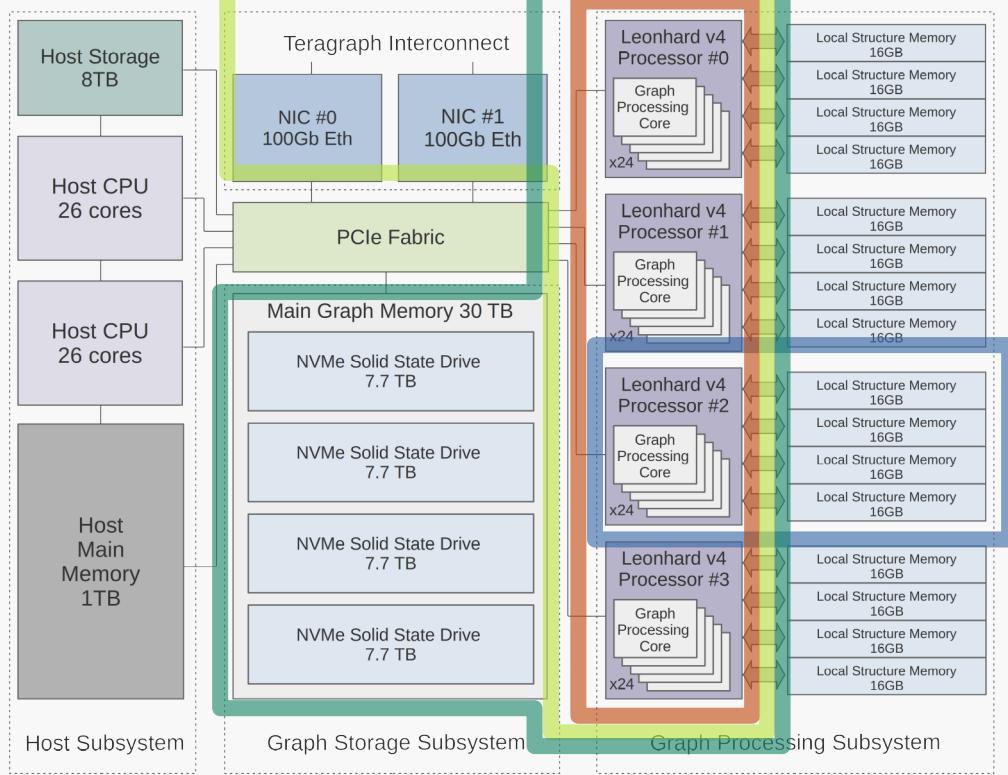
Teragraph node #1



- Предусмотрено длительное размещение графов в оперативном доступе.
- Используется ассоциативная память большого объема (2.5ГБ на одно ядро Graph Processing Core, GPC)
- Ядра GPC являются весокоэффективными гетерогенными системами, взаимодействующими через единой адресное пространство PCIe
- GPC самостоятельно обращается в локальное графовое хранилище 30ТБ и графовые хранилища других узлов
- Хост система выполняют второстепенные функции (инициализация, распределение и т.д.)

Вычислительный узел комплекса Тераграф

Teragraph node #1



- Предусмотрено длительное размещение графов в оперативном доступе.
- Используется ассоциативная память большого объема (2.5ГБ на одно ядро Graph Processing Core, GPC)
- Ядра GPC являются весокоэффективными гетерогенными системами, взаимодействующими через единой адресное пространство PCIe
- GPC самостоятельно обращается в локальное графовое хранилище 30ТБ и графовые хранилища других узлов
- Хост система выполняют второстепенные функции (инициализация, распределение и т.д.)

Архитектура комплекса Тераграф



Характеристика

Значение

Количество процессоров Леонард Эйлер **9**

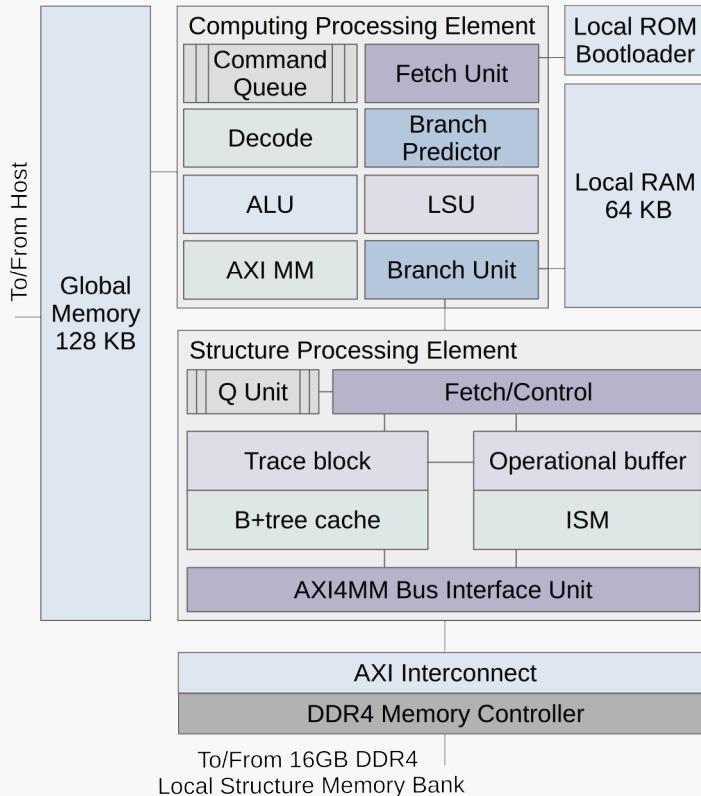
Количество GPC **216**

Кэш память (DDR4, ГБ) **576**

Оперативная память GPC (ТБ) **48**

Количество хранимых ключей **1 триллион**

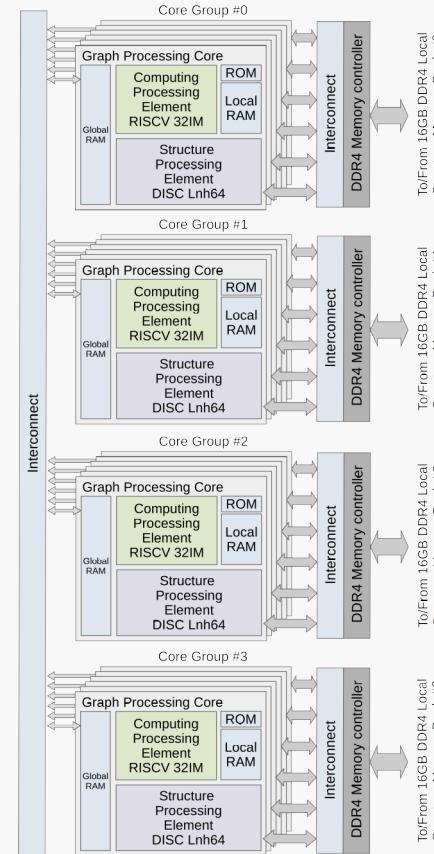
Гетерогенное ядро обработки графов



- GPC состоит из двух тесно связанных микропроцессоров: Computing Processor Element (CPE) и Structure Processing Element (SPE).
- CPE реализован на базе микропроцессора с набором команд riscv32im.
- SPE представляет собой микропроцессор с набором команд дискретной математики DISC
- SPE подключен, как ускорительное ядро кшине памяти CPE.
- Производительность GPC сопоставима с производительностью одного ядра Intel Xeon Platinum v8 при 10x меньшей частоте (267 МГц) и 40x меньшем количестве вентилей (2.5 млн.)

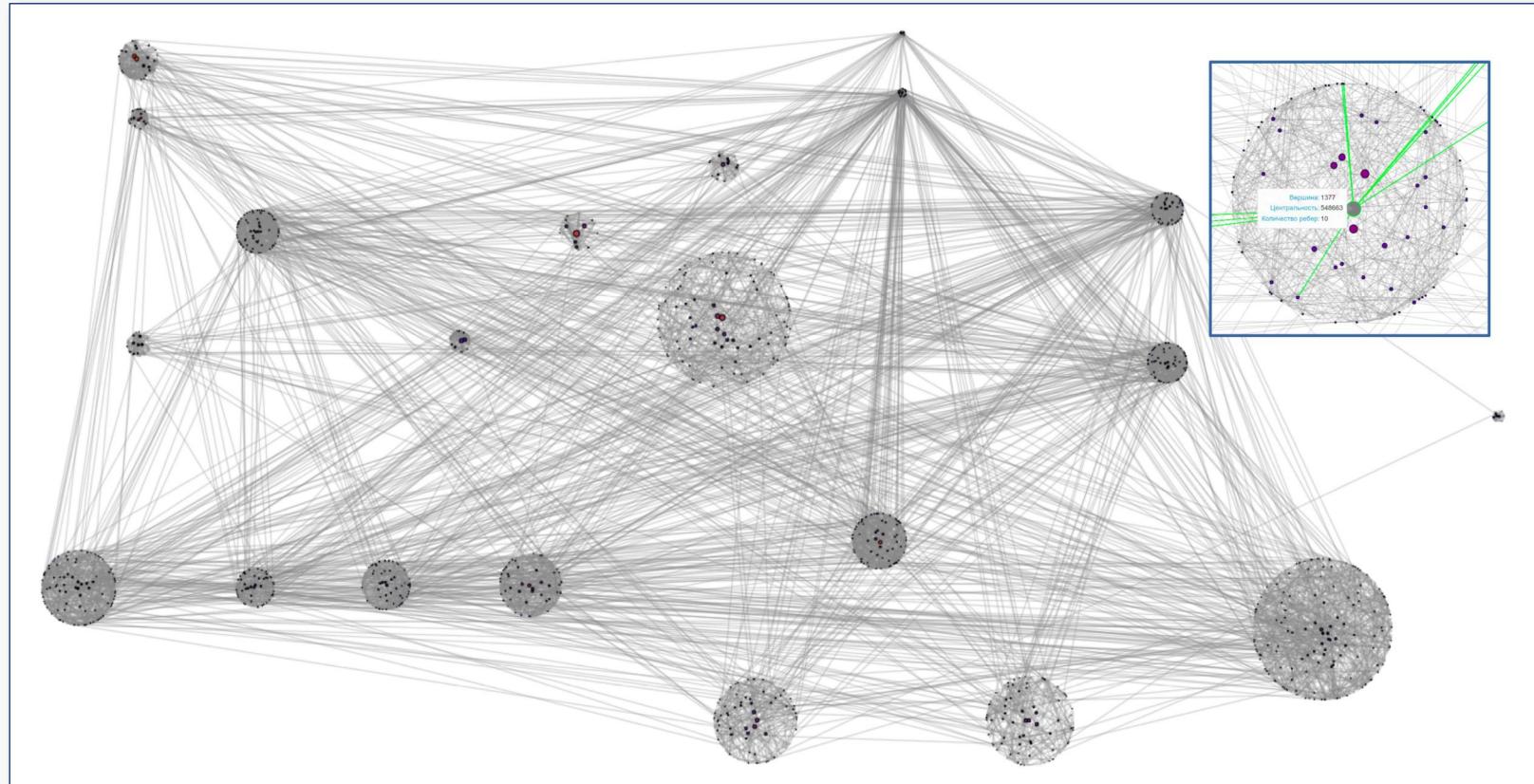
Микропроцессор Леонард Эйлер

- Ядра GPC объединяются в группы ядер (до 6 ядер в группе)
- В каждой группе предусмотрена глобальная память 128КБ для обмена данными между хост-подсистемой и СРЕ.
- В каждом ядре СРЕ предусмотрены аппаратные очереди сообщений Host2GPC и GPC2Host на 512 записей по 32 бит каждая.
- Все GPC в одной группе подключены к одной шине памяти DDR4 (16ГБ).
- Хост-подсистема может независимо управлять каждым GPC в отдельности.
- Основным программным компонентом программного ядра является обработчик (подобно шейдеру), который написан на языке C и загружается по запросу хост-системы.



Примеры, исходные коды библиотек: <https://alexbmstu.github.io/2022>

Пример работы одного гетерогенного ядра Тераграф



Определение сообществ и центральности
~4K вершин, 16 млн кратчайших путей

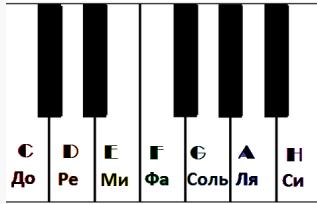
03

Тема
практикума:

создание
МУЗЫКИ!



Ноты и Midi формат:



\flat - бемоль(понижение на полутон)

\sharp - диез(повышение на полутон)

| Октава | Нота | | | | | | | | | | | |
|--------|------------|-----|-----|-----|-----|-----|-----|------|-------|-----------|-----|-----|
| | C | C# | D | D# | E | F | F# | G | G# | A | A# | B |
| | До | До# | Ре | Ре# | Ми | Фа | Фа# | Соль | Соль# | Ля | Ля# | Си |
| -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 0 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 1 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| 2 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 3 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| 4 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 5 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 |
| 6 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 7 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 |
| 8 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 9 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | | | | |

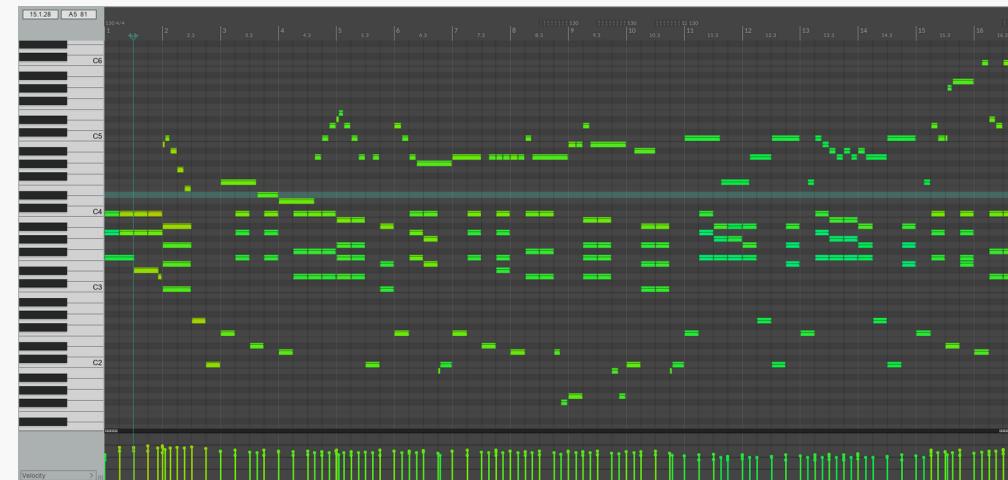
Ноты и Midi формат:

Последовательность сообщений:

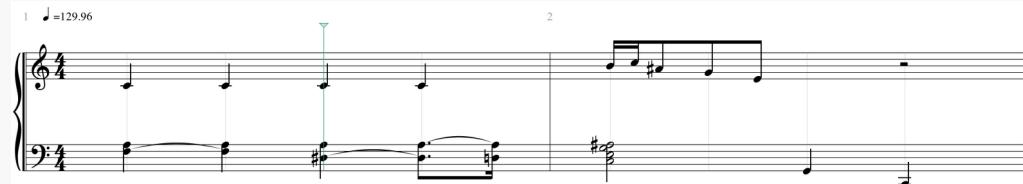
Track 0: Acoustic Guitar

```
<meta message track_name name='Acoustic Guitar' time=0>
note_on channel=0 note=58 velocity=72 time=0
note_off channel=0 note=58 velocity=64 time=50
note_on channel=0 note=60 velocity=72 time=0
note_off channel=0 note=60 velocity=64 time=30
note_on channel=0 note=61 velocity=72 time=0
note_off channel=0 note=61 velocity=64 time=30
note_on channel=0 note=60 velocity=72 time=0
note_off channel=0 note=60 velocity=0 time=30
note_on channel=0 note=70 velocity=72 time=0
note_off channel=0 note=70 velocity=64 time=100
note_on channel=0 note=66 velocity=72 time=0
note_off channel=0 note=66 velocity=64 time=33
note_on channel=0 note=72 velocity=72 time=0
note_off channel=0 note=72 velocity=64 time=38
note_on channel=0 note=66 velocity=72 time=0
note_off channel=0 note=66 velocity=64 time=21
note_on channel=0 note=70 velocity=72 time=0
note_off channel=0 note=70 velocity=64 time=41
note_on channel=0 note=66 velocity=72 time=0
note_off channel=0 note=66 velocity=64 time=33
note_on channel=0 note=70 velocity=72 time=0
...
...
```

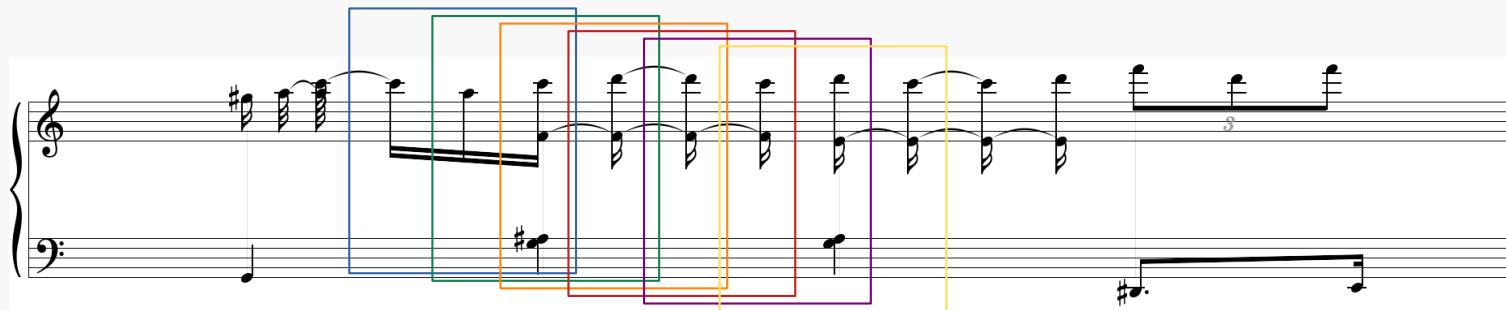
Матричное представление:



Нотный стан:

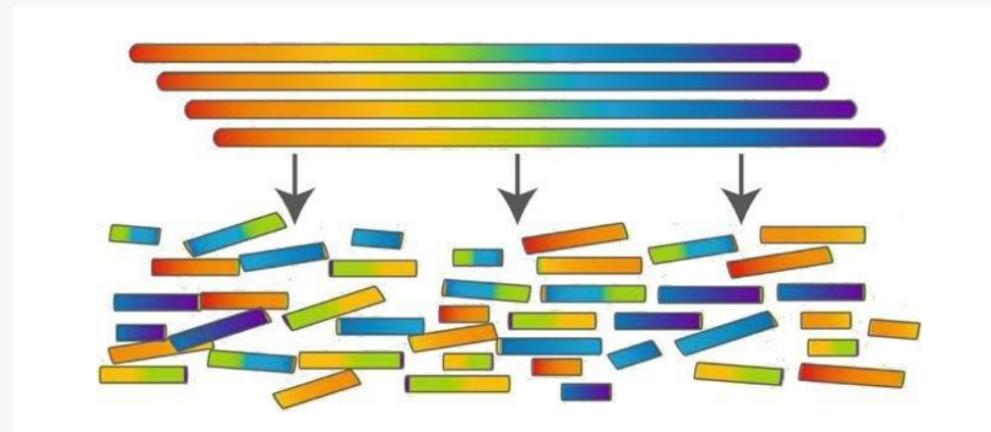
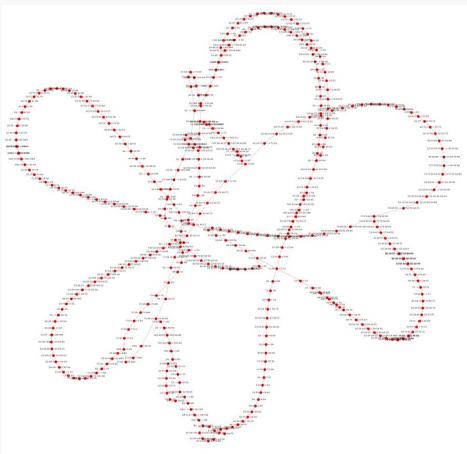


Граф Де Брюйна:



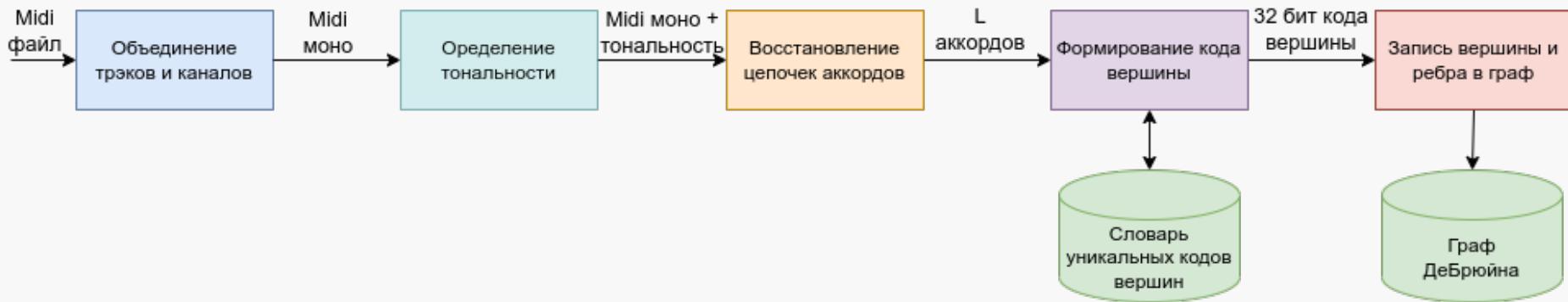
Графов Де Брюйна прелюдии И.С.Баха

Пример применения графов Де Брюйна для восстановления последовательности участков генома



Конвейер генерации музыки:

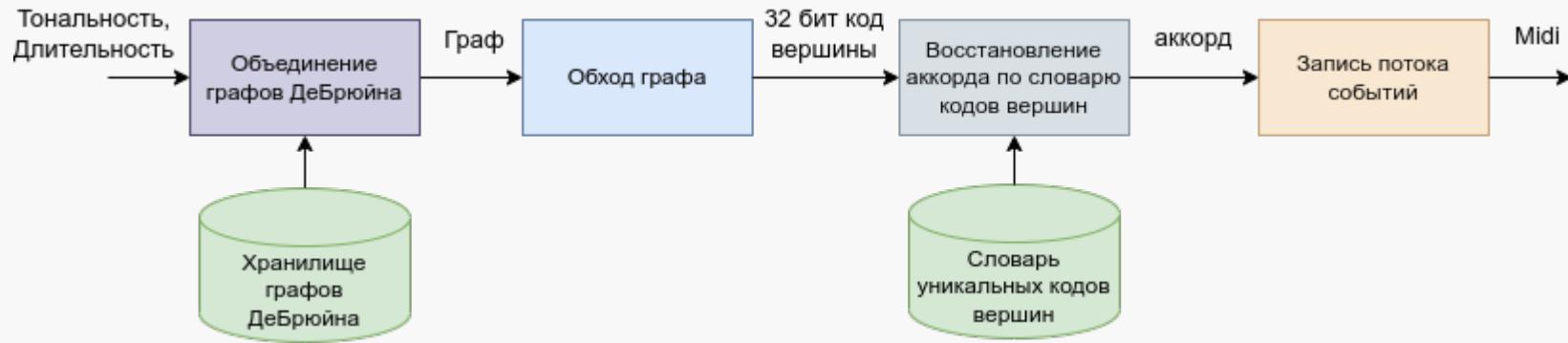
Этап 1: создание графов де Брюйна



- Объединение треков и каналов** — инструменты исходного midi сводятся в один голос.
- Определение тональности** — используется алгоритм на основе Байесовского классификатора (TemperleyKostkaPayne алгоритм)
- Восстановление цепочек аккордов** — последовательность событий преобразуется в состояния
- Формирование кода вершины** — каждое состояние кодируется в виде последовательности нот. Для состояния из Словаря уникальных кодов вершин получается уникальный ключ вершины.
- Запись вершины и ребра в граф** - ключ вершины добавляется в граф дебрюйна и соединяется ребром с предыдущей вершиной.

Конвейер генерации музыки:

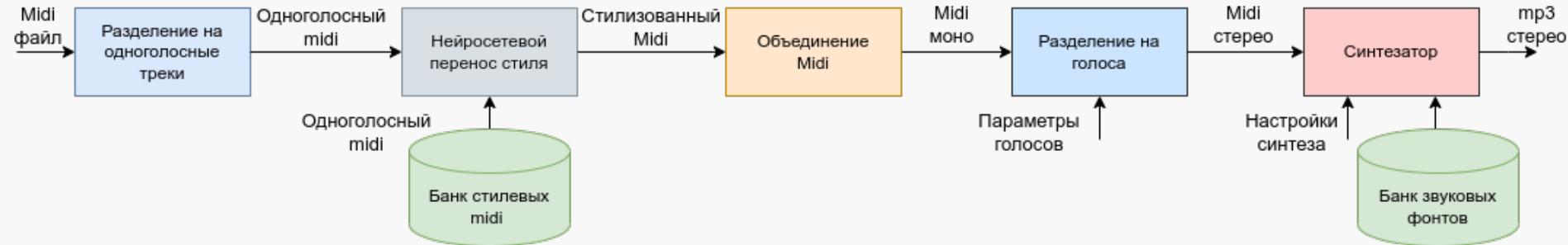
Этап 2: обход графов де Брюйна



- Объединение графов ДеБрюйна** — поиск в директории всех графов с заданной тональностью. Граф записывается в GPC на суперЭВМ Тераграф
- Обход графа** — проход по графу ДеБрюйна по случайному маршруту с заданным количеством шагов
- Восстановление аккорда** — выборка полной записи о вершине из словаря аккордов, хранимого в GPC суперЭВМ Тераграф
- Запись потока событий** — аккорд преобразуется в последовательность событий midi.

Конвейер генерации музыки:

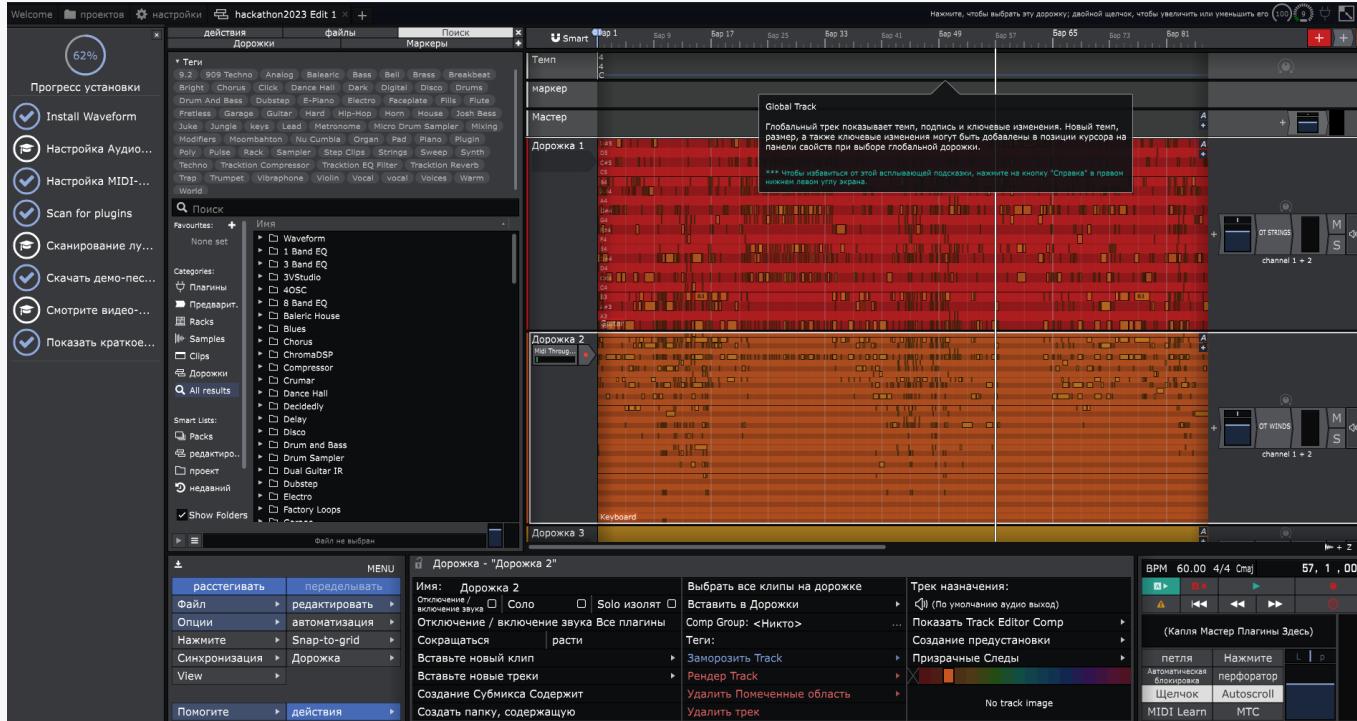
Этап 3: Стилизация, многоголосие и синтез звука



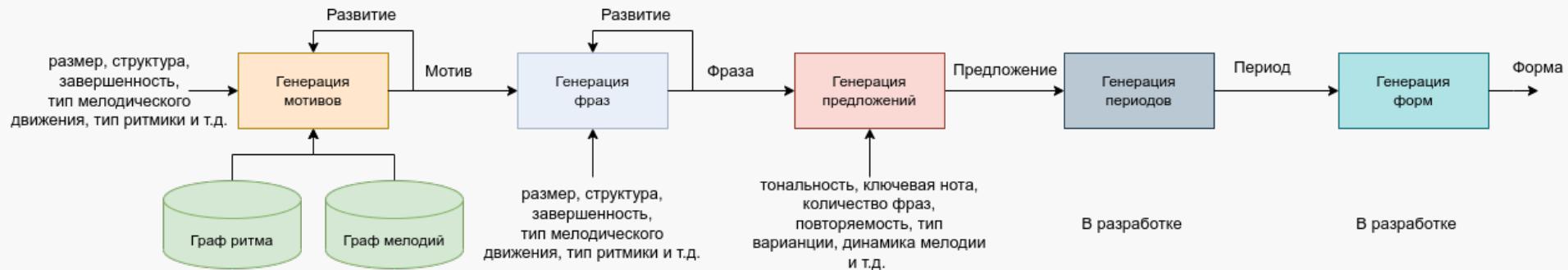
- 1. Разделение на одноголосные треки** — выделяются ноты в каждом аккорде: нижняя, верхняя, 2-я, 3-я, и т. д. для стилевой обработки одноголосных партий.
- 2. Нейросетевой перенос стиля** — используется стилевая композиция для определения вектора коэффициентов. Далее применяется итерационное изменение параметров Velocity обрабатываемого midi для минимизации ошибки
- 3. Объединение midi** — стилизованные голоса сводятся вместе
- 4. Разделение на голоса** — по настройкам пользователя выделяются верхний, нижний голоса, или голоса по заданному диапазону нот. Каждый голос записывается в отдельный канал.
- 5. Синтезатор** — для каждого голоса задается soundfont (.sf2) и номер инструмента. Звук генерируется с использованием timidity, далее wav кодируется в mp3 с использованием ffmpeg.

Конвейер генерации музыки:

Этап 4: Аранжировка в DAW (ауд 803, по желанию) LogicPro, Cubase, Waveform, ...



Технология структурного синтеза музыкальных произведений



- Мотив** — является элементарной ритмо-мелодической единицей. Состоит из одного либо двух тактов.
- Фраза** — последовательность, состоящая из двух мотивов.
- Предложение** — единая музыкальная мысль, состоит из 2, реже 3, еще реже 4, фраз. Последняя фраза называется каденцией - завершением предложения.
- Период** — форма изложения законченной или относительно законченной музыкальной мысли, завершённая кадансом. Является наибольшей синтаксической и наименьшей композиторской единицей в музыке.
- Форма** — воплощение музыкального содержания, под которым понимается целостная организация музыкальной пьесы как совокупности мелодики (мотивов, фраз, предложений и др.), гармонии, метра и ритма, склада и фактуры, текста (стихотворного, молитвословного, прозаического), тембров и кластеров и др.

План практикума:

Часть 1: Изучение принципов работы микропроцессорного ядра RISC-V (8 часов)

Часть 2: Разработка и отладка программ в вычислительном комплексе Тераграф (8 часов)

Часть 3: Обработка и визуализация графов в вычислительном комплексе Тераграф (4 часа)

Часть 3: Командный практикум. Создание музыкальных произведений (14 часов)

Спасибо!

alexpopov@bmstu.ru
Москва, сентябрь 2023 г.

