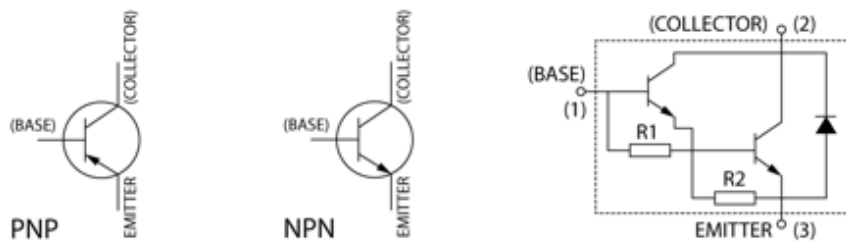


rus.time_for_d

Alexandr Kirilov (<https://github.com/alexandrkirilov>)

Время для D? 2D? 3D? Или nD?

Со времени изобретения транзистора принципиально нового ничего не было сделано, все последующие изобретения в сфере вычислительной техники тем или иным образом были связаны с транзистором. Это был технологический прорыв который позволил создать индустрию IT в том виде в котором она существует есть сейчас.



Технологии разработанные тогда потихоньку сейчас начинают тормозить развитие и не только в IT сфере. Со временем, если не предпринимать никаких действий, этот эффект будет становиться все более заметным, а потом и критическим. Почему? Ответ можно найти в структуре строения памяти компьютера или жесткого диска и процедурах чтения и записи информации совершаемых в этих носителях.

Вдаваться в подробности технической реализации той или иной памяти не является целью статьи. Более подробную информацию, с техническими деталями о том как это работает, можно найти в интернете:

- Компьютерная память
- Жёсткий диск

Если говорить кратко - то все эти устройства используют линейные алгоритмы и статическую линейную архитектуру которая не адаптируется под информацию которую обрабатывает или хранит. Это и есть основная проблема. Статическая линейность на физическом уровне организации компьютеров:

- в случае с памятью это цепь транзисторов организованная каким-то образом и не обладающая способностью перестроить сама себя в зависимости от информации которую мы хотим сохранить
- в случае с HDD это линейная последовательность секторов расположенная на огромной спирали которая начинается в центре диска и заканчивается на краю диска так же не обладающая возможностью перестроить по требованию к примеру количество витков и последовательность байт
- и т.д.

Список перечислений линейных алгоритмов используемых в физической архитектуре компьютера можно продолжать очень долго.

Основная проблема статической линейности, не в линейности как таковой, а в том что физически она не измена - мы не можем изменить состав и порядок элементов расположенных на этой линии в зависимости от того или иного условия. Структура не оптимизируется или адаптируется под информацию которую она хранит. В случае с RAM мы не можем изменить последовательность транзисторов, которые организуют последовательность байт, в случае с HDD мы не можем изменить последовательность намагниченных секторов на основании которых организуется последовательность байт, мы можем только изменить значение намагниченности статического сектора (в случае с CD мы вообще ничего не можем изменить). В случае работы с Big-data это может быть основным "горлышком бутылки" на данный момент времени при тех технологиях которыми мы обладаем.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	...	N

Простейшим примером иллюстрирующим этот принцип может быть винтовая лестница в небоскребе, где на каждом этаже расположен один офис, а вы курьер который выполняет задачи по доставке информации из одного офиса в другой. Вам дали задачу собрать подписи на документе у нескольких групп людей, где одна группа расположена на 5,34,190,1113 этажах, вторая группа расположена на 5,78,1,23456,34,7819 этажах. А теперь представьте сколько этажей вы пройдете мимо не обращая на них внимания??? Представили???

Приблизительно так же выглядит хранение данных в файлах на диске и для примера это причина почему существуют такие программы как defrag HDD (дефрагментация диска) в некоторых ОС (они просто выстраивают последовательно эти группы так, что бы все элементы групп и сами группы были последовательно расположены, тем самым уменьшая "работу ног" проходя мимо ненужного). А теперь представьте что вам нужно это сделать много миллионов раз для многих сотен тысяч групп - это есть описания проблем big-data, огромное количество энергоемких операция которые не несут никакого логического или структурного смысла а только обязательны для носителя информации основанного на транзисторных технологиях.

Представьте на секунду что будет с вашими ногами к вечеру после такого рабочего дня??? Представили??? Понравилось???

Другим примером может быть гирлянда на новогодней елке для ваших детей и вас попросили вкрутить цветные лампочки в определенном порядке. Вы это сделали, а в конце обнаружили что забыли вкрутить между 35 и 36 лампочку зеленого цвета, а всего вы вкрутили к примеру 1234 лампочки. Как результат вам придется выкрутить все с 1234 до 35 (1199), вкрутить зеленую и только потом вкрутить назад 1199 лампочки и получить в сумме 1235.

Приблизительно так выглядит работа памяти когда вы меняете состояние вашего компьютера путем нажатия кнопок и т.д.

Почему эта проблема сейчас становится достаточно остро - потому что многократно выросли и будут расти дальше объемы информации которыми оперирует человечество, а технологии без изменения физических принципов хранения информации начинают уже отставать. Еще 10-15 лет назад размер хранилища в 1Tb был чем-то около-фантастическим, сейчас диски для домашнего использования производятся по 5-10Tb и это уже становится нормой. Некоторым пользователям уже этого не хватает. Размер корпоративных хранилищ данных исчисляются тысячами если не десятками тысяч терабайт и это далеко не предел.

Теперь несколько слов о том как можно было бы решить данную проблему. Для начала, что стоило бы сделать разработчикам, что бы понять проблему глубже:

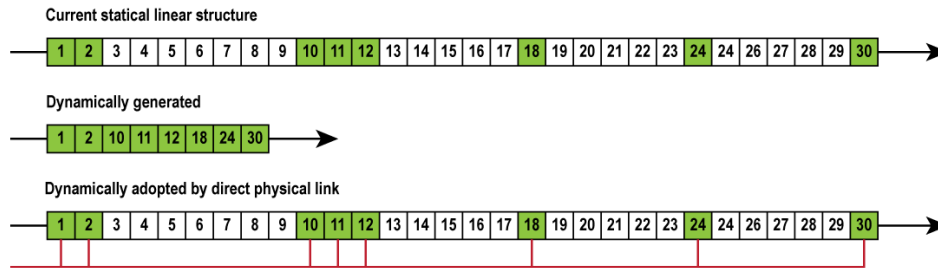
- **Самый главный пункт - понять что носитель информации всегда будет определять формат при помощи которого будет информация сохранена, статическая линейная структура это только один из множества**

вариантов и без физического развития возможностей носителей информации не будет роста индустрии IT, рано или поздно мы упремся в предел транзисторных технологий и горизонтальное масштабирование путем многопоточности, количества процессоров или дисков не спасет положение дел

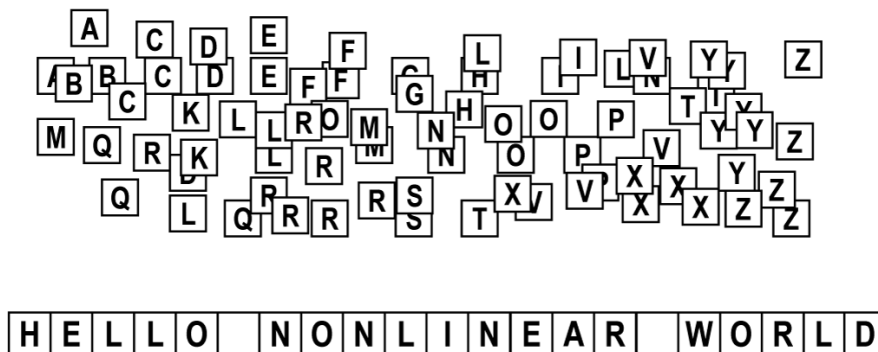
- **Сместить акцент с производства IT продуктов "по мотивам" в сторону новых физических принципов хранения информации которые расширят возможности самой информации как средство фиксации или отображения событий или процессов:** огромное количество версий одного и того же основанного на коде разработанном десятки лет назад (практически все операционные системы основаны на разработках 20-30 летней давности и многие библиотеки не переделывались не потому, что их не могут переделать, а потому что нет никакого смысла их переделывать, они написаны оптимально по отношению к физическим свойствам платформ на которых они работают - вспоминаем историю транзисторов)
- **Знать свое место и перестать на время придумывать парадигмы программирования и языки программирования написанные на одном языке C, остановиться, огляделся вокруг и понять что IT существует только потому что есть информация которой нужно управлять вне зоны IT и у всего есть предел:** сейчас сама по себе IT индустрия начинает напоминать перегревающийся двигатель который разогнан до максимальных значений, при этом путь еще предстоит очень длинный, а ресурс на пределе - придумано огромное количество всего, но это все работает на одном статическом линейном решении, возвращаемся к предидущему пункту

Можно еще добавить несколько пунктов к этому списку, но они менее значимые, чем эти три.

После того как мы изменим свое собственное отношение к тому что мы называем IT, скорей всего появятся свободные ресурсы которые можно направить на создание чего-то действительно нового, например нелинейной памяти способной адаптироваться под размер хранимого объекта и устанавливать связи между объектами по требованию тем самым избавляясь от необходимости "ходить мимо этажей в небоскребе". Это прямо противоположно тому, что происходит сейчас - мы адаптируем информацию под средства хранения.



Нужно разработать такой механизм хранения информации, который будет выглядеть как набор пустых ячеек памяти заготовленных заранее в не связанном состоянии или механизм создающий эти ячейки памяти по требованию в зависимости от структуры информации. Или как минимум устанавливающий прямые связи между ячейками, тем самым уменьшая количество шагов до цели.



Иллюстрацией этого может быть детская игра в слова, когда ребенок из кубиков с буквами складывает слова. Его собственная идея или идея того человека который учит ребенка, определяет структуру слова, а ребенок из свободных кубиков складывает слова и воспринимает множество кубиков с буквами как единое целое - слово (механизм объединения частей в целое описан в статье "[Понимание blockchain](#)") и потом дополняет это слово связями, которые организуют предложения, которые в свою очередь могут менять значение этого слова структурой предложения.

Встает вопрос о том какой материал позволит это все реализовать? Ответ вокруг нас, нужно только оглядеться внимательно (это было упомянуто в третьем пункте). Что вокруг нас самостоятельно создается и так же

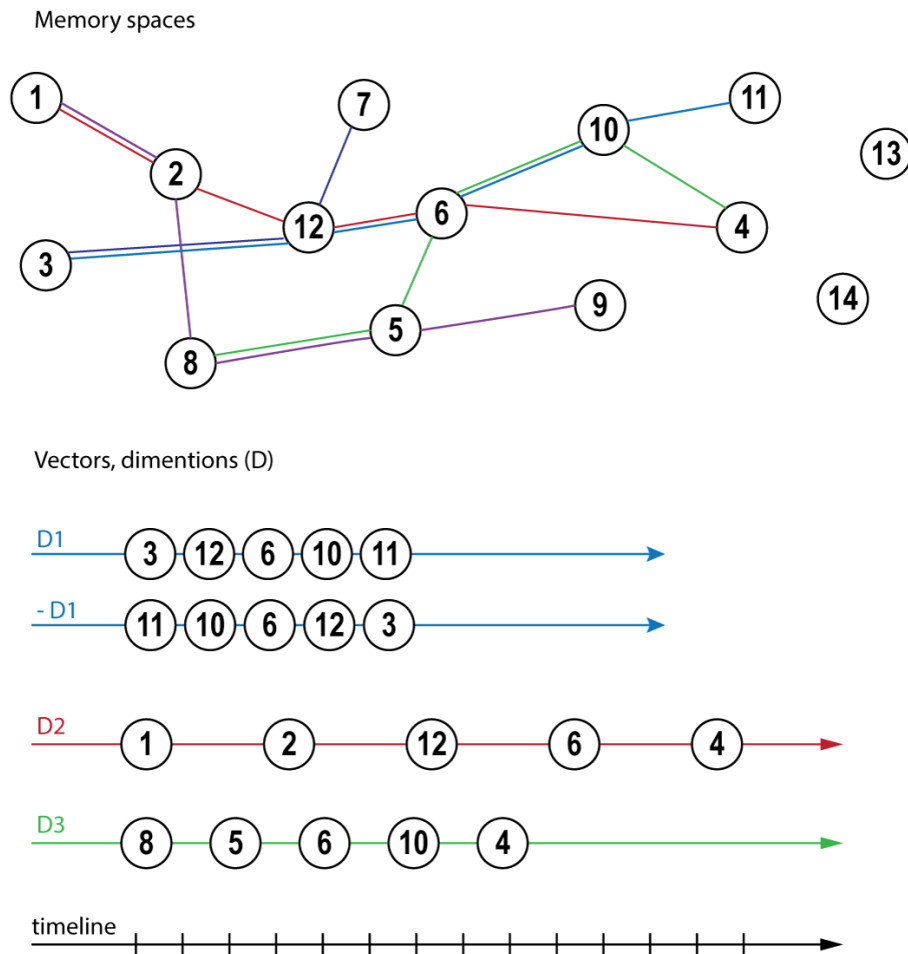
самостоятельно исчезает - клетка, живая клетка.

Уже давно доказан факт, что клетка обладает памятью. В медицине существует даже целый раздел посвященный этому. Так почему же не отработать технологию которая позволит для примера из клеточных элементов растворенных в воде при помощи электрических токов малой силы и различной частоты выращивать клетки памяти определенной структуры в которой будут храниться определенные объекты. Тут может пригодиться объектно-ориентированная парадигма программирования (ООП), потому как на физическом уровне это будут действительно объекты (а не линейная эмуляция как сейчас это реализовано) с которыми нужно будет что-то делать. При удалении объекта, клетка током большей силы или другой частоты (частично о частотах описано в статье "[Frequency](#)") будет разрушаться до уровня компонентов для новых объектов по новым схемам, тем самым обеспечивая замкнутость системы с определенной емкостью.

Если представлять графически архитектуру то скорей всего она будет очень похожа на нейроны в мозгу человека, некие объекты с большим количеством связей между собой и электрическим током протекающем по связям между ними.

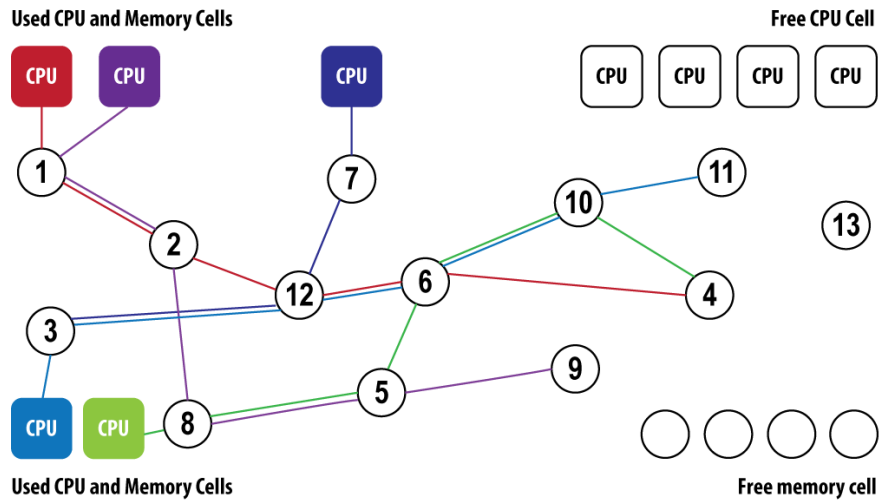
Немного футурологии. Представим, что все то, что описано выше - реализовано. С точки зрения сегодняшнего пользователя должна забавно выглядеть просьба продать 3 литра памяти в компьютерном магазине.

Теперь несколько слов о том, почему статья названа именно так "Время для D? 2D? 3D? Или nD?" и никак иначе. D - от английского слова dimension, если быть совсем точным, то от математического определения вектора.



То, что на этом изображении - есть описание основной причины почему эта статья названа именно так как она названа.

Есть некоторое пространство в памяти, в котором выращены клетки содержащие объекты (информацию об объектах). Пространство памяти неизменно и не требует вычислительных мощностей, все объекты статичны. Единственное что происходит - организация векторов через связи между клетками определенной последовательности и направлении тем самым адаптируя память под конкретную задачу. В контексте задачи адресное пространство не больше и не меньше чем количество объектов объединенных вектором. Мы вернулись к линейности но с возможностью перестраивать элементы линейного алгоритма.



Если подключить к одному вектору один процессор и настроить его на уникальную частоту (схема уникальности частот для каждого процесса может быть взята из сотовой связи, когда есть диапазон но для каждого абонента используется своя частота) и синхронизировать с этой частотой ток проходящий по связям между объектами вектора, то в большинстве случаев можно будет избегать коллизий в один момент времени при использовании одного и того же объекта различными векторами (данное решение может позволить снять проблемы в программировании связанные с разрешением конфликтов использования одной ячейки памяти различными процессами). Через какое-то время если проводить работу по анализу и унификации объектов то можно получить своего рода "Таблицу Менделеева" для информации, в которой будут свои "металлы" с определенными свойствами, свои инертные газы с другими свойствами, свои минералы и т.д. Данная таблица в ключе big-data сможет сильно уменьшить объемы хранения сместив акцент от объектов как таковых в сторону контекста их использования и связями между ними.

У данной технологии может быть огромное количество преимуществ относительно сегодняшних. Их можно перечислять очень долго ... И статья может перерасти в огромный IT-талмуд в котором будут описаны закономерности процессов ...

Может пришло время для D?

PS ...

Диалог описанный ниже наиболее ярко иллюстрирует принцип на основе которого могут разрабатываться устройства обработки и хранения информации.

- A: Why the information is information?
- B: Because the information - is the energy-in-formation.
- A: But why are we mentioning only about the "in formation" only?
- B: Because energy is energy, but the formation creating the forms and fixing it by formulas for being able to repeat action or making copy within another portion of energy ...

Следите за обновлениями автора в [**Linkedin**](#).

Следите за AR|BO|RE|US обновлениями в [**Twitter**](#) в [**Linkedin**](#).